uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática
2017-2018

Trabajo Fin de Grado

"Análisis de datos radar para detección y evasión de obstáculos en vehículo autónomo"

Luis Miguel Pinto Maguiña

Tutor

Jesús García Herrero

Lugar y fecha de presentación prevista: 5 octubre 2018.
Campus de Colmenarejo, Salón de grados.

RESUMEN

El presente documento tiene como objetivo explicar mediante la información obtenida de distintas fuentes y experiencias personales, por medio de ilustraciones y tablas, cómo se ha llevado a cabo la realización del proyecto. El objetivo del proyecto es desarrollar un sistema que permita adquirir y analizar los datos procedentes de un sensor radar. De forma que en futuros proyectos puedan usarse para la evasión de obstáculos en un vehículo autónomo.

Todas las medidas que se toman para alcanzar el objetivo del documento se realizan en 3 etapas distintas:

1. Configuración de los componentes. El primer paso para que todo funcione, es tener claro el flujo de información y la conexión que tendrán los componentes del proyecto. Para su mejor comprensión, se ha diseñado la siguiente estructura.

Ilustración 1. Resumen. Esquema de las conexiones del proyecto [30] [31]

Los cuadrados explican los cables que conectan los componentes entre ellos. Mientras que los números indican el orden del flujo de la información en el sistema.

2. Programación del microcontrolador y la creación de los scripts para el análisis y la creación de gráficas en Matlab. En este caso, se usará la herramienta Atollic TrueStudio para programar el microcontrolador para que envíe los datos del radar en un formato que se especificará. Estos datos, los guardaremos en ficheros .csv para su posterior análisis con los scripts de Matlab. Quedan incluidas opciones para cambiar entre los tipos de envío del radar.

3. Por último, la realización de las pruebas y su correspondiente análisis. Las pruebas realizadas simulan la circulación en una carretera. Dispone de objetos estáticos y otros en movimiento que simulan un acercamiento peligroso que puede concluir en un accidente. Una vez analizados los datos, se comprueba que tienen el resultado esperado.

Los resultados de todas las pruebas han finalizado correctamente y verifican el correcto funcionamiento de todo el sistema. Gracias a la creación de diferentes gráficas para entender cada prueba, ha sido más fácil la labor de verificar que las pruebas cumplen con todos los requisitos del sistema. Así como explicar en este documento de una forma fácil y clara los resultados obtenidos.

La conclusión de este proyecto es que efectivamente, se han cumplido las expectativas y el correcto funcionamiento del sistema. Se han cumplido en gran medida la planificación impuesta al comienzo y los problemas encontrados durante el desarrollo se han solucionado personalmente.

Palabras clave: sensor, radar, procesado, análisis, vehículo autónomo.

AGRADECIMIENTOS

A mis padres, especialmente a mi madre, por el gran apoyo que me han brindado, no solo durante la realización del grado, sino durante toda mi vida, y que me ha permitido llegar a este preciso instante. A mi familia y a mi pareja, aunque ya sea parte de la familia, por haber estado a mi lado, ofreciéndome toda la ayuda posible en los momentos más duros que han acontecido en esta época, incluso en este proyecto. A mis amigos, los cuales me han ofrecido siempre varios puntos de vista en aquellas situaciones en las que necesitaba consejos para seguir adelante. Y, por último y no por ello menos importante, a mis profesores, los cuales han sido un gran ejemplo a seguir y un referente de éxito conseguido a través de la perseverancia, la constancia y una buena actitud. Pero especialmente a mi tutor, que durante toda la realización de este proyecto ha sido un apoyo constante y que me ha permitido finalizar esta etapa con determinación y entusiasmo.

A todos vosotros, gracias. Porque gracias a vosotros todo esto y lo que está por venir ha sido posible.

Luis Miguel Pinto Maguiña

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

1.1. VISIÓN GENERAL

El primer rastro de un vehículo autónomo se remonta a 1939. Donde General Motors mostró un vehículo eléctrico embebido en el pavimento de una carretera. A partir de este momento, varias empresas mostraron un gran interés y empezaron a realizar inversiones de grandes cantidades en este ámbito. Hasta el punto de que numerosos gobiernos, incluida la Unión Europea, invirtiesen en varios proyectos. Hasta la actualidad, donde se siguen desarrollando varios prototipos e incluso varias empresas han sacado sus modelos a la venta al público.

Un vehículo autónomo (VA de ahora en adelante), es un vehículo capaz de realizar las habilidades humanas de manejo y control. Puede percibir todo su entorno y navegar según se considere oportuno en cada ocasión. Un usuario (el conductor), elegirá un destino y no será necesario que realice ninguna operación mecánica del vehículo.

El funcionamiento de los VA se basa en percibir el entorno mediante distintos componentes y técnicas como láser, radar, sistema de posicionamiento global y visión computarizada. Usan esta información para identificar una ruta apropiada evitando todos los obstáculos que puedan encontrarse durante el camino, así como la identificación de las señales de tráfico establecidas.

En nuestro caso, nos basaremos en la utilización y manejo de la información proporcionada por un radar para la conducción autónoma del vehículo y la evasión de los posibles obstáculos.

Un radar es un sistema que utiliza ondas electromagnéticas para medir distancias, altitudes y velocidades de los distintos objetos al alcance de este sensor.

1.2. MOTIVACIÓN

Los VA, pertenecen a un sector que está en auge. Como hemos mencionado anteriormente, todas las empresas automovilísticas apuestan por crear su propio VA en lo que se podría considerar hoy en día como una carrera. Estas empresas, buscan fabricar un VA capaz de circular de forma totalmente libre sin necesidad de ninguna operación mecánica por parte del conductor.

La mayor dificultad y a la vez el mayor miedo que incluyen estos vehículos es la seguridad vial de la que disponen y de la cual hacen uso. Si son capaces de evitar accidentes como haría una persona, si el juicio del código programado equivale o supera al humano y si es capaz de respetar todas y cada una de las señales viales impuestas en cada región.

Todas estas empresas, se unen a la gran búsqueda de fabricar un vehículo que sea totalmente eléctrico y, por ello, contribuir a la sostenibilidad del medio ambiente. Más

aún con las nuevas polémicas de grandes tasas de contaminación en las principales ciudades mundiales. Pero este es un tema que no será tratado en este documento a pesar de lo interesante que resulta.

En este documento, se tratará uno de los puntos más importantes para que estos vehículos tengan un gran éxito: la seguridad en el transcurso de su ruta.

Por último, mencionar que en España, el uso de un VA está prohibida siempre que este circule por carreteras públicas (no en circuitos privados). Por lo que es una motivación extra contribuir a que España acepte estas medias como ya han hecho muchos países.

Todo esto, en conjunto, ha sido la motivación a la hora de escoger el proyecto de fin de grado. Consistirá en el uso de un radar que será implantado en el VA de forma que se recogerán los datos que proporcione y se estudiarán para que el VA pueda tomar decisiones en caso de encontrarse un peligro, una posible colisión o un humano al que pudiera atropellar de forma involuntaria, así como cualquier posible obstáculo en su trayectoria. [1] [2]

1.3. OBJETIVOS

El principal objetivo del desarrollo es realizar un sistema que accede y utiliza los datos radar para evasión de obstáculos. Todo esto mediante la recogida de los datos proporcionados por el radar y, que tras procesarlos y analizarlos, pueda obtenerse la información necesaria para que pueda saber si se trata de un peligro y actuar en consecuencia para finalizar el trayecto de la ruta marcada por el conductor.

Este diseño puede dividirse en dos grupos de objetivos:

1.3.1. Objetivos del proyecto

- Lograr una comunicación con el sensor. De forma que se pueda enviar información de configuración y de esta forma, recibir todos los datos que envía según la configuración establecida.
- Configuración, modificación (si fuese necesaria) y conexión de los componentes hardware para su correcto funcionamiento.
- Procesar la información relativa a la detección de posibles obstáculos en la trayectoria.
- Recoger los datos en un archivo .csv que nos permita poder trabajar con esta información.
- Realizar todas las pruebas necesarias para asegurar el correcto funcionamiento del radar según los requisitos establecidos.
- Analizar los datos recibidos para saber que está pasando dentro del campo de visión del radar.

1.3.2. Objetivos personales

- Aprender el funcionamiento y la configuración de los distintos dispositivos utilizados a lo largo del proyecto: radar, ST Link, fuente de alimentación regulable y PixHawk.
- Profundizar en diferentes leguajes de programación. Algunos que ya conocía y otros totalmente nuevos para mí pero que son esenciales en el transcurso del proyecto.
- Entender las diferentes maneras de tráfico de datos. En este caso, el CAN Bus.
- Ganar experiencia con la realización de la documentación de un proyecto de estas características.

1.4. ESTRUCTURA DEL DOCUMENTO

El presente documento se estructura en los siguientes apartados:

1. Introducción: este primer apartado consistirá en realizar una pequeña explicación del entorno en el que vamos a trabajar. Explicando qué son los VA y los componentes que usaremos. Además, incorpora una motivación personal para la realización de este proyecto y los objetivos del mismo.

2. Marco teórico: este apartado contiene información referente a todos los componentes del proyecto. Expone información como la legislación actual y las responsabilidades penales y jurídicas, la propiedad intelectual del sistema y los posibles riesgos en cuanto a un ciberataque al sistema. Por último, incluirá un apartado con el impacto socioeconómico explicado al detalle.

3. Estado del arte: aquí se explicará el actual estado de los componentes que engloban o que estén relacionados con lo que se va a desarrollar.

4. Organización y gestión del proyecto: se exponen la planificación y control de las tareas del proyecto, la identificación de los requisitos (y sus pruebas correspondientes) y todas las tecnologías que han sido empleadas.

5. Desarrollo del proyecto: se explica la forma en la que se ha realizado todo el proyecto. La programación, la configuración del microcontrolador, la extracción de los datos y su análisis.

6. Soluciones previas propuestas: se explicarán todas las soluciones posibles que se encontraron para resolver los distintos problemas del proyecto.

7. Ampliaciones futuras: posibles mejoras del sistema y su instalación en otros sistemas o componentes.

8. Análisis de los resultados: tras todo el desarrollo, se comprueba que cumplen todos los requisitos impuestos al comienzo del proyecto.

9. Conclusiones y problemas encontrados: Se explican los distintos problemas encontrados a lo largo del proyecto. También se exponen las conclusiones del mismo.

10. Bibliografía: es este apartado estarán todos los enlaces a las páginas web de donde se ha obtenido la información y que se ha usado en este documento.

11. Acrónimos y definiciones: se incluyen las definiciones de las palabras técnicas que contiene el documento de forma que toda persona sin altos niveles en la materia pueda entender su contenido.

12. Project summary: un breve resumen del proyecto en una lengua extranjera, inglés en este caso.

2. MARCO TEÓRICO

En este apartado, se expondrá la información existente referente a los componentes del proyecto.

2.1. MARCO REGULADOR

Para que el proyecto sea completo, es necesario conocer y tener claras aquellas regulaciones legales relacionadas con este. Por ello, como se ha mencionado antes, en el caso de España hay diversas limitaciones en lo que a conducción autónoma se refiere.

Con esta información, se conseguirá que el sistema propuesto esté dentro de la legalidad vigente y no incumpla ninguna ley que pueda suponer un problema en el futuro.

2.1.1. Marco legal en España y la Unión Europea

Actualmente en España, la legalidad prohíbe el uso de VA a particulares. Ya que hay una gran cantidad de interrogantes legales. Desde cómo debería estar programada la toma de decisiones que tenga vidas en juego hasta las responsabilidades en caso de accidente.

Debido a la falta de legislación al respecto, en este apartado se informará de las medidas que se están planteando tomar y como afectarían al producto final que tenemos pensado realizar con este proyecto.

Un resumen rápido sobre la legislación española sobre los VA sería: "*solo fabricantes e instituciones pueden probar vehículos sin conductor y únicamente en determinadas vías*" según autobild.

Se pronostica que alrededor de 2020, se podría disponer de una nueva ley para regular todos los casos mencionados anteriormente. Aunque con la aparición de accidentes de diversas empresas y sus vehículos en el extranjero, puede retrasarse algo más ya que ponen en duda la madurez de los VA. Empieza a ser una carrera contrarreloj el buscar un marco legal que estipule las pautas a seguir para la utilización de estos medios antes de que se impongan en las carreteras sin que haya dado tiempo a elaborar una normativa. No solamente ocurre en España, sino en la Unión Europea.

En abril de 2016, se llevó a cabo la **Declaración de Ámsterdam sobre cooperación en el capo de la conducción automatizada y conectada.** Este documento, plateaba promover un marco legal para la conducción autónoma que debería estar disponible en 2019. Además, incluiría un marco legal específico para esta tecnología como la privacidad y la protección de datos o la ciberseguridad.

En España, lo más cerca que se ha estado de una regulación sobre la materia, fue a través de una **Instrucción de la Dirección General de Tráfico.** Que autorizó pruebas de investigación de conducción autónoma en vías abiertas con tráfico general. [3]

2.1.2. Responsabilidad civil

"*La implantación de los vehículos autónomos deberá ir precedida de la necesaria regulación administrativa*". Esto dice la Memoria de la Fiscalía General del Estado publicada en 2016.

Entre las normas que se prevén necesarias, está la **normativa de responsabilidad civil y su aseguramiento**. Sobre la que se deberán realizar varias modificaciones ya que actualmente la responsabilidad recae sobre el conductor, figura inexistente en algunos casos de conducción autónoma.

Por este supuesto, se plantea si desplazar las responsabilidades del conductor al fabricante del VA bajo el título de responsabilidad objetiva o principio de riesgo o incluso a los fabricantes del software que utiliza el vehículo y que impulsa a que tome las acciones en cada ocasión. Puede extenderse hasta la industria que lo explota y a los creadores de los mapas que han sido utilizados. Llegando a responsabilizar a la Administración Pública si los vehículos autónomos forman parte de los servicios públicos. [3] [4]

2.1.3. Responsabilidad penal

En lo que se refiere a la responsabilidad penal en caso de que hubiese un accidente de tráfico, el Ministerio, piensa que no podrá atribuirse al conductor la culpabilidad por falta de dominio en el suceso. Ya que incluso en algunos casos, ni siquiera habrá un conductor.

Por ello, insiste en que "*habrá que explorar si con la doctrina de la imputación objetiva pueden examinarse eventuales responsabilidades penales atribuibles a los elaboradores del software por graves deficiencias o al fabricante del vehículo por relevantes irregularidades cuando, con conocimiento de los riesgos que conllevan y de los probables resultados lesivos, hayan omitido deberes elementales de cuidado permitiendo la circulación de los vehículos automatizados en esas condiciones*".

Por otra parte, dado que los VA ofrecen la posibilidad de intercambiar entre conducción autónoma y manual, sería de gran utilidad instalar cajas negras en los VA para informar si en el momento del accidente conducía el sistema autónomo o el conductor físico. [3] [4]

2.1.4. Estándar técnico

También es importante puntualizar los estándares técnicos que regulan el desarrollo y de los cuales haremos uso.

Para el proyecto en cuestión, vamos a utilizar el estándar de Métrica Versión 3. Ofrece un instrumento útil para la sistematización de las actividades del ciclo de vida del software (planificación, desarrollo y mantenimiento de sistemas de información). Cuya propiedad intelectual pertenece al Ministerio de Hacienda y Función Pública. [5]

Por otro lado, para la realización progresiva del proyecto, también utilizaremos técnicas ágiles de desarrollo de software. De forma que el desarrollo y las pruebas del sistema, se harán de forma iterativa, en varios sprints (entregas al cliente) donde se crearán prototipos del sistema y cuando se tenga la versión deseada, se finalizará con un sprint final que es básicamente la creación del producto final.

2.1.5. Posibles vulnerabilidades ante ciberataques

Lo que se busca en este proyecto es la creación y mejora de los vehículos autónomos lo más seguros y fiables posibles. Pero hay un problema muy grande respecto a los vehículos autónomos: los ciberataques.

Obviamente, a un conductor humano, no es posible de controlar mediante un ciberataque y conseguir que haga algo a voluntad del atacante. Pero en el caso de los VA, este estará tripulado la mayor parte del tiempo por un sistema, lo que implica que el cerebro del coche será un ordenador. Esto significa que es vulnerable a ciberataques

que pueden alterar el funcionamiento del VA e incluso concluir en accidentes no contemplados.

Es un aspecto a tener en cuenta, ya que se han investigado actualmente varios casos peligrosos debidos a estos riesgos.

2.1.6. Propiedad intelectual

En cuanto a la propiedad de este sistema en concreto. Todo aquel que adquiera o haga uso del sistema tratado en este documento, tendrá que dejar constancia del grupo desarrollador del sistema. Indicando a los miembros del grupo como creadores o colaboradores en caso de que se incluya este sistema en otros.

2.2. IMPACTO SOCIOECONÓMICO

2.2.1. Beneficios del proyecto

Los beneficios derivados de la elaboración de este sistema tienen tanto beneficios en la innovación como beneficios en el sector empresarial y la sociedad. Por lo que se explicarán de forma separada.

Por un lado, los beneficios de innovación. El servicio que se va a desarrollar resulta una novedad en la sociedad actual. Muchas grandes empresas están del todo centradas en el desarrollo de productos similares al que se trata en este proyecto, pero ninguna ha conseguido llegar a un sistema perfecto y libre de errores. Por lo que puede significar tanto una novedad como una mejora en la tecnología actual. Ofrecerá una garantía de los datos gracias a un análisis exhaustivo de los mismos por lo que se ganará en seguridad y prestaciones del sistema.

También será necesario formar a la plantilla en los conocimientos del hardware y el software que se necesita para el proyecto. Ya que todas las herramientas tienen conexión entre ellas y debe conocerse a la perfección su funcionamiento.

En cuanto al coste, una vez que se monte el sistema en un vehículo, el precio del mismo aumentará considerablemente debido a las nuevas funcionalidades que incluye respecto a un vehículo tradicional.

En cuanto al impacto laboral y ambiental: se crearán nuevos empleos para perfiles expertos en la materia y que puedan instruir a la plantilla actual. Puesto que se trabaja tanto en componentes hardware como en sistemas software, serán dos equipos diferentes los que se encarguen de esto una vez comience el proceso de producción en una empresa. En el ámbito ambiental, se pretende incluir esta propuesta en vehículos autónomos que utilicen motores eléctricos principalmente. Por lo que es una gran ayuda para los altos niveles de contaminación detectados en todas las ciudades mundiales debido a la emisión de gases producidos por los coches que utilizan combustible.

Por todo esto, las relaciones empresariales aumentarán en gran medida. Estos productos se podrán internacionalizar de forma que cualquier país pueda disfrutar de esta tecnología siguiendo las normas estipuladas para su uso. Se accederá a nuevos

mercados debido a la inclusión de nuevas tecnologías, tanto movilísticas como informáticas por el uso de inteligencia artificial en el procesado y el análisis de los datos.

Por otro lado, nos encontramos con los beneficios en el sector empresarial y la sociedad. En este aspecto, se notará un aumento significativo en el empleo. Estos nuevos integrantes deben contar con conocimientos tecnológicos asociados al área en el que vayan a colaborar.

Gracias a esto, conseguirá ser un componente competitivo en el sector automovilístico y en el sector de la inteligencia artificial procedente del análisis y resultado de los datos.

La calidad de vida mejorará ya que libera a las personas de realizar una labor que, a mucha gente a diario, quita varias horas y puede producir estrés en horas puntas: la conducción. Junto con la mejora medioambiental que conlleva (explicada en este mismo punto más arriba), proporcionará a la sociedad una vida más cómoda y saludable.

2.2.2. Plan de difusión y divulgación de los resultados

Ya que la finalidad de este sistema es incluirlo en un vehículo y, posteriormente conseguir el mayor número de ventas con el mismo (junto con la contribución tecnológica y medioambiental a la sociedad), la divulgación de este sistema se realizará a través de los siguientes medios:

1. Publicación en revistas científicas y del sector: se publicarán los resultados de la investigación y el desarrollo del sistema en revistas para poner en conocimiento de todo el sector tecnológico lo que se ha conseguido.

2. Actividades de formación: como se ha especificado anteriormente, todo el manejo de los componentes del sistema es un tanto complicado. Por lo que habrá jornadas de formación para nuevos y empleados que estaban ya en la plantilla.

3. Divulgación al público: ya que una de las finalidades de este sistema (una vez montado y funcionando junto al resto de sistemas en un VA) es la venta al público, se informarán de las prestaciones que ofrece en ruedas de prensa, anuncios y folletos explicativos.

4. Difusión en la web: se crearán apartados en las páginas webs de venta para una primera vista de lo que ofrece e incluso alguna jornada de demostración para interesados.

5. Acuerdos previos con empresas o entidades públicas: los gobiernos y empresas saben que la concienciación con el medio ambiente en la actualidad significa mucho para la sociedad. Por lo que se crearán acuerdos con estas empresas para la implantación de vehículos con nuestro sistema entre sus filas. (Taxis, vehículos del estado, etc.).

2.2.3. Plan de explotación de los resultados

Para llevar a cabo la explotación de los resultados de esta investigación, se seguirán los siguientes aspectos:

1. El sistema final, es capaz de capturar los datos capturados por el radar de forma que se pueden analizar para identificar posibles obstáculos y crear alertas por colisión. Dada la investigación actual en el sector, se trata de un avance en una novedad mundial que se abre más y más camino en el entorno tecnológico actual. El objetivo de este proyecto es crear un sistema que se pueda incluir en un prototipo y realizar las pruebas necesarias para su garantía de completo funcionamiento.

2. Se pueden identificar 4 participantes diferentes en el proyecto: desarrolladores del sistema, desarrolladores del vehículo, desarrolladores del resto de sistemas y los empresarios que compran todas estas tecnologías. Cada participante tiene una idea para el producto final con las integraciones de todas las partes. Todos los desarrolladores, buscan crear un sistema compacto que funcione perfectamente uniendo todos los pilares. Mientras que los empresarios buscan un producto final útil con el que puedan conseguir el mayor número de ventas y de beneficios.

3. Como se ha indicado anteriormente, la propiedad intelectual del sistema desarrollado en este proyecto pertenece al grupo descrito más adelante en la organización del proyecto. Siempre que algún sistema conjunto o un componente externo que utilice este sistema deberá incluir al equipo como creadores del sistema que han utilizado (el desarrollado en este proyecto).

4. Los usuarios a los que se debe dirigir el mercado del sistema son los propietarios de investigaciones referentes a los VA. También se incluirán en este punto usuarios que desarrollen sistemas similares a este y que lo necesiten para perfeccionar el suyo.

3. ESTADO DEL ARTE

3.1. DEFINICIÓN

Para dar una definición que englobe todo lo que se quiere enseñar en este documento, se explicarán las tareas que puede realizar un vehículo autónomo, que son las siguientes:

1. Ruta: el VA debe ser capaz de, una vez establecido un punto de destino, llevar al usuario desde la ubicación actual a ese punto final.

2. Entorno: el VA debe ser capaz de controlar todo lo que le rodea. Obstáculos (pueden ser personas, objetos inmóviles o móviles como coches, etc.). Todo esto gracias al radar que se usará en este proyecto.

3. Ubicación: para desempeñar la función de VA, tiene que saber en todo momento en que posición se encuentra para conocer las rutas principales o prioritarias que debe tomar en su recorrido hasta llegar al destino.

Una definición de Alex Barredo en programarfacil.com sería: *"es un ordenador con cuatro ruedas y un habitáculo donde van los viajeros y el conductor"*

Dentro de este sector, se pueden diferenciar varios niveles de coches según su tipo de conducción autónoma. Desde el coche tradicional más conocido, hasta coches capaces de repostar en una gasolinera por ellos mismos.

Para entender todos los niveles de los que disponemos, nos ayudaremos de las reglas determinadas por la Autoridad de Tráfico de Estados Unidos.

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Ilustración 2. Niveles de automatización según SAE [33]

3.1.1. Nivel 0: no automatización

En este nivel, entran todos los coches tradicionales, los cuales no disponen de ninguna medida de automatización. No se considera que tener un sistema de luces o limpiaparabrisas automático sea un símbolo de automatización.

3.1.2. Nivel 1: asistente de conducción

En este nivel, entran vehículos con equipamiento de diferentes ayudas a la hora de conducir. Como puede ser el asistente de frenado o el control por crucero. Por ejemplo, para el asistente de frenado se utilizarían sensores como ultrasonidos o algún tipo de radar y en algunos casos cámaras.

3.1.3. Nivel 2: semi-automático

El nivel 2 incorpora varias funcionalidades del nivel 1 pero mejoradas. Por ejemplo, al usar el control de crucero, podemos establecer la velocidad deseada a 70km/h, pero en caso de que encuentre algún obstáculo u otro vehículo este disminuirá la velocidad o frenará en caso de ser necesario.

3.1.4. Nivel 3: autonomía controlada (autopistas y autovías)

En este nivel, están los coches autónomos capaces de mantenerse por sí mismos en un carril de una autopista sin necesidad de que actúe el conductor. Puede mantener una velocidad constante o frenar según precise. Además, un apartado extra sería el aparcado automático.

Se pueden encontrar coches de varias marcas a la venta en el mercado con estas prestaciones orientadas a la conducción automática.

3.1.5. Nivel 4: alto nivel de autonomía

Actualmente se está trabajando e intentando mejorar dentro de este nivel. Están incluidos aquellos vehículos capaces de llevar a los pasajeros a la ruta establecida y encontrar por sí mismos (de forma autónoma) un aparcamiento cerca del punto de destino. La tecnología de la que se está hablado, ya está disponible en empresas como Tesla, Waymo y Cruise.

Prácticamente lleva a cabo todo el trabajo de la conducción de inicio a fin. Reconoce todo el tráfico que encuentra, entiende las señales y se comunica con varios sistemas externos al coche para conocer mejor aún el entorno en el que circula.

Ilustración 3. Ejemplo de conducción en VA Tesla [6]

3.1.6. Nivel 5: autonomía completa

La única diferencia que tiene respecto a los vehículos de nivel 4 es que estos son capaces de repostar de forma autónoma, sin ayuda del conductor incluso para esto. También son capaces de circular bajo cualquier tipo de condición meteorológica. [6] [7]

3.2. HARDWARE DE UN VEHÍCULO AUTÓNOMO

Una de las partes más importantes para el funcionamiento de este sector es el Hardware. Un coche autónomo se nutre de todo tipo de sensores para circular de forma autónoma por las calles.

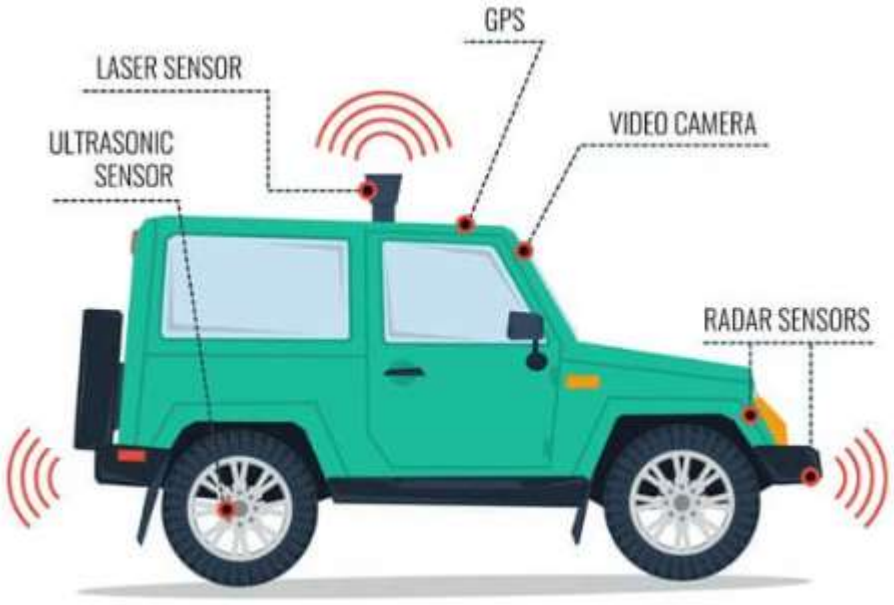

Ilustración 4. Hardware en un vehículo autónomo. Por programarfacil.com [6]

Su funcionamiento depende de dos elementos imprescindibles, al igual que los humanos; **cerebro** y **visión**.

El cerebro debe ser capaz de saber lo que ha de hacer en cada circunstancia y centralizar el funcionamiento del resto de componentes y sistemas que utiliza el vehículo. De modo que puede acceder y utilizar en la medida apropiada los elementos del coche como pueden ser el freno, el acelerador, los intermitentes o la caja de cambios del coche.

Por otro lado, la visión es algo crucial para que el resto funcione. Ya que el vehículo debe ser capaz de visualizar todas las señales de la vía por la que circula, el tráfico del que dispone a su alrededor e incluso, en una intersección, saber cuándo es su turno para acceder a ella y realizar el giro correspondiente. Por ejemplo, el coche de Google prioriza este elemento con su detector mediante luces infrarrojas LIDAR.

3.2.1. Ordenador central

Se considera el cerebro del VA. Recopilará todas las señales provenientes de los sensores (radar en nuestro caso). El software de este ordenador central (todo en un portátil personal en nuestro caso) deberá ser el encargado de procesar la información recogida para la toma de decisiones.

Este cerebro, debe procesar toda la información lo más rápido posible. El tiempo de frenado estimado en una persona es de 2 segundos, mientras que el de un VA se estima en 3 décimas de segundo.

3.2.2. LIDAR

Es el acrónimo de Laser Imaging Detection And Ranging. Consiste en un haz de luces infrarrojas con visión de 360º y un rango de visión de alrededor de 100 metros. Este dispositivo óptico gira sobre sí mismo pudiendo acceder a todo el entorno del vehículo. Lanza un haz de luces que, al impactar con un objeto, rebota y los devuelve al origen donde, por medio de una cámara, son captados. De esta forma, permite dibujar el entorno de forma tridimensional.

Es uno de los componentes más precisos ya que cuenta con una precisión del orden de +- 2 centímetros.

Ilustración 5. LIDAR en el techo de vehículo de Google. [33]

3.2.3. Cámaras de visión

Son las encargadas de hacer el trabajo de los “ojos” del vehículo. Identifican otros vehículos, luces, semáforos, colores, peatones, etc. Por sí solos no serían de mucha utilidad, pero al combinarlos con algoritmos de visión artificial, tenemos unos ojos ideales para un VA.

Ilustración 6. Cámara estereoscópica. [7]

3.2.4. Radares

La funcionalidad de este componente es similar a la de un sensor LIDAR, pero funciona de distinta manera. El radar manda señales que rebotan en los objetos a los que detecta y de esta forma puede saber en todo momento la distancia y la posición relativa en la que se encuentra.

Se pueden colocar varios en distintas posiciones del vehículo, aunque suele ir instalado en la parte frontal.

Este es el único componente que utilizaremos en este proyecto en concreto.

Ilustración 7. Posible ubicación de radares para la detección de obstáculos. [7]

3.2.5. GPS/IMU

Para poder tener conocimiento de la ubicación del vehículo en todo momento, se incorporan dos sistemas de control de ubicación:

Por un lado un sistema GPS que está integrado en la mayoría de los dispositivos electrónicos que conocemos.

Por otro lado, la IMU (Intertial Measuremente Unit). Es básicamente un giroscopio junto con un acelerómetro que permite conocer la dirección, orientación y velocidad en un espacio tridimensional.

Se complementan los dos elementos, por ejemplo para el caso de que el VA entre a un túnel. El GPS en la mayoría de los casos perdería la señal y el coche estaría perdido. Pero aquí entra en juego la IMU ya que sabiendo la dirección, la orientación y la velocidad que mantiene, sabemos donde situar el vehículo en todo momento.

Ilustración 8. Localizador GPS [34]

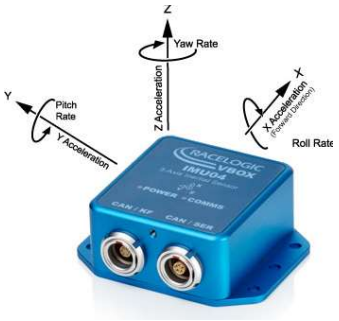

Ilustración 9. IMU [35]

3.2.6. Sensores de sonido

Este es otro sentido del que debe estar dotado un VA (el auditivo). Debe estar atento a lo que ocurre a su alrededor a su nivel acústico. Además, toda esta información aportará a los algoritmos una mejor toma de decisiones.

3.2.7. Sensores y cámaras interiores

Por último, también existen componentes que se instalan en el interior del vehículo para monitorizar constantemente el estado del conductor. Puede ir desde sensores biométricos que estudien las acciones faciales del conductor hasta sensores que midan las constantes vitales del conductor.

Ilustración 10. Sensor biométrico para evitar la somnolencia. [36]

3.3. SOFTWARE

3.3.1. Redes neuronales y aprendizaje automático

Se basa en un set de parámetros de entrada que van reajustando el propio sistema.

Como hemos dicho anteriormente, un VA es parecido a una persona por lo que debe ser capaz de aprender por sí solo.

Los vehículos utilizarán la inteligencia artificial para utilizar los datos que recogen del entorno y hacer predicciones sobre los posibles sucesos en las carreteras. El VA será capaz de detectar a ciclistas y peatones, a la vez que podrán evaluar situaciones, de forma que se introduce totalmente en la conducción junto con el resto de los vehículos. Eso sí, pudiendo reconocer los peligros y anticipándose a ellos con más tiempo.

3.3.2. Visión artificial

Consiste en la capacidad de analizar y reconocer el entorno del VA mediante componentes destinados a la captura de imágenes. No necesita una alta inversión en hardware, pero sí en experimentación.

Se considera una de las partes más complejas del software en un VA ya que es un sector que no ha sido muy desarrollado por las empresas. Pero que ahora está considerado uno de los puntos más importantes para mejorar el sistema de visión artificial que instalan en sus vehículos. [39]

3.3.3. Limitaciones del funcionamiento

Como es de esperar, estos vehículos adolecen de varias limitaciones. Por ejemplo, son capaces de llevar una conducción eficiente y segura solo en rutas programadas y mapeadas anteriormente.

Otro posible problema es el factor medioambiental, ya que en días con lluvia, niebla o luz excesiva/insuficiente, podrían verse en problemas todos los sensores y las cámaras.

Pero la principal limitación de que tiene hoy en día el vehículo autónomo es sobre el terreno legal. Ya que, como hemos mencionado anteriormente, las leyes en España no están preparadas para la inserción de estos VA en las carreteras.

La principal inquietud se refiere al dilema ético ya que el vehículo (en caso de colisión segura con otras personas o vehículos), deberá escoger si salvar la vida a las otras personas por ser un mayor número (por ejemplo) o salvar al conductor del vehículo autónomo. Aunque, por otra parte, se ha demostrado que un VA es capaz de detectar accidentes inminentes y actuar al respecto anticipándose a los hechos y llevando a cabo

las contingencias que tiene programadas, algo que la mayoría de los conductores habituales no podrían ser capaces de prever. [6] [7] [8]

3.4. ESTADO DEL ARTE: RADAR

Puesto que este proyecto va a contener información únicamente sobre este componente que es el que se usará junto con el coche autónomo, se va a dedicar un apartado en concreto para la explicación del estado actual de los sensores radar incluidos en los vehículos autónomos.

El sistema radar, utiliza ondas de radio que rebotan en los objetos en su "zona de visión" obteniendo distancia, velocidad, ángulo del objeto que ha detectado. Dado que tienen un menor nivel de absorción que los sensores que funcionan enviando láser, pueden medir a unas distancias más largas.

El radar, es más fiable en condiciones meteorológicas desfavorables como niebla, nieve, lluvia o tormentas.

Actualmente, el radar no cuenta con una media angular tan amplia como sus competidores y puede llegar a perder en ciertos momentos visión del vehículo objetivo en las curvas. Además, también puede darse la ocasión de que al encontrarse dos coches de pequeñas dimensiones, puede equivocarse y considerar a los dos objetos como un único objeto más grande y enviar una información errónea de proximidad.

Aunque si algo está claro, es que todos los sistemas para los coches autónomos funcionan mejor cuando están en sincronía entre ellos. Ya que el radar funciona de una forma mucho más fiable con cámaras u otros sensores de apoyo. [2]

Ilustración 11. Sensor RADAR Continental SRR200 [27]

4. ORGANIZACIÓN Y GESTIÓN DEL PROYECTO

Este apartado, tiene como principal finalidad el seguimiento, la planificación y el control de las actividades. También de los recursos humanos empleados y del material necesario durante toda la realización del proyecto. Gracias a este control, se puede saber en todo momento en qué etapa del proyecto se está y cuáles son los siguientes pasos a realizar.

4.1. PLANIFICACIÓN DEL PROYECTO

En primer lugar, se mostrarán las tareas que se han especificado al comienzo del proyecto y la duración que tendrá cada una.

Posteriormente, se mostrará el diagrama de Gantt correspondiente a las tareas especificadas en primera instancia.

4.1.1. Planificación

La tabla mostrada a continuación, muestra las diferentes tareas que se realizan a lo largo de todo el proyecto. Especificando la duración de cada una mediante su fecha de inicio y final. Así como los responsables en cada caso.

Cabe destacar que la jornada de realización de las tareas, es de lunes a viernes con una carga diaria de 4 horas (20h semanales), lo que se ajusta a un horario de media jornada. Los fines de semana a lo largo del proyecto serán libres por lo que no se realizará ningún tipo de avance en el proyecto. Aunque, aun disponiendo de esa jornada, dependiendo del puesto del trabajador, puede trabajar entre 1 y 4 horas diarias.

Nombre de la tarea	Fecha inicio	Fecha Fin	Responsable
Realización Documentación	Lunes 08/01/2018	Viernes 14/09/2018	Luis Miguel Pinto
- **Introducción**	Lunes 08/01/2018	Lunes 05/03/2018	Luis Miguel Pinto
- **Marco regulador**	Martes 06/03/2018	Viernes 16/03/2018	Luis Miguel Pinto
- **Estado del arte**	Martes 19/03/2018	Viernes 30/03/2018	Luis Miguel Pinto
- **Organización del proyecto**	Lunes 02/04/2018	Viernes 06/04/2018	Luis Miguel Pinto

- **Desarrollo del proyecto**	Lunes 09/04/2018	Viernes 31/08/2018	Luis Miguel Pinto
- **Soluciones previas propuestas**	Lunes 03/09/2018	Miércoles 05/09/2018	Luis Miguel Pinto
- **Análisis de los resultados**	Jueves 06/09/2018	Viernes 07/09/2018	Luis Miguel Pinto
- **Conclusiones y problemas encontrados**	Lunes 10/09/2018	Martes 11/09/2018	Luis Miguel Pinto
- **Apartado en inglés**	Miércoles 12/09/2018	Viernes 14/09/2018	Luis Miguel Pinto
Revisión de la Documentación	Lunes 17/09/2018	Jueves 20/09/2018	Luis Miguel Pinto
Sprint 1	Jueves 01/02/2018	Viernes 23/02/2018	Luis Miguel Pinto
- **Análisis de la situación**	Jueves 01/02/2018	Lunes 05/02/2018	Luis Miguel Pinto
- **Diseño del prototipo**	Martes 06/02/2018	Viernes 09/02/2018	Luis Miguel Pinto
- **Implementación del prototipo**	Lunes 12/02/2018	Miércoles 21/02/2018	Luis Miguel Pinto
- **Pruebas del prototipo**	Jueves 22/02/2018	Viernes 23/02/2018	Luis Miguel Pinto
Evaluación de la solución	Lunes 26/02/2018	Miércoles 28/02/2018	Luis Miguel Pinto
Sprint 2	Jueves 01/03/2018	Miércoles 28/03/2018	Luis Miguel Pinto
- **Análisis de la situación**	Jueves 01/03/2018	Viernes 02/03/2018	Luis Miguel Pinto
- **Diseño del prototipo**	Lunes 05/03/2018	Viernes 09/03/2018	Luis Miguel Pinto
- **Implementación del prototipo**	Lunes 12/03/2018	Viernes 23/03/2018	Luis Miguel Pinto
- **Pruebas del prototipo**	Lunes 26/03/2018	Miércoles 28/03/2018	Luis Miguel Pinto
Evaluación de la solución	Jueves 29/03/2018	Viernes 30/03/2018	Luis Miguel Pinto
Sprint 3	Lunes 02/04/2018	Jueves 26/04/2018	Luis Miguel Pinto
- **Análisis de la situación**	Lunes 02/04/2018	Miércoles 04/04/2018	Luis Miguel Pinto
- **Diseño del prototipo**	Jueves 05/04/2018	Martes 10/04/2018	Luis Miguel Pinto
- **Implementación del prototipo**	Miércoles 11/04/2018	Viernes 20/04/2018	Luis Miguel Pinto
- **Pruebas del prototipo**	Lunes 23/04/2018	Jueves 26/04/2018	Luis Miguel Pinto
Evaluación de la solución	Viernes	Lunes	Luis Miguel Pinto

	27/04/2018	30/04/2018	
Sprint 4	Martes 01/05/2018	Martes 28/04/2018	Luis Miguel Pinto
- **Análisis de la situación**	Martes 01/05/2018	Viernes 04/05/2018	Luis Miguel Pinto
- **Diseño del prototipo**	Lunes 07/05/2018	Viernes 11/05/2018	Luis Miguel Pinto
- **Implementación del prototipo**	Lunes 14/05/2018	Miércoles 23/05/2018	Luis Miguel Pinto
- **Pruebas del prototipo**	Jueves 24/05/2018	Lunes 28/05/2018	Luis Miguel Pinto
Evaluación de la solución	Martes 29/05/2018	Jueves 31/05/2018	Luis Miguel Pinto
Sprint 5	Viernes 01/06/2018	Martes 26/06/2018	Luis Miguel Pinto
- **Análisis de la situación**	Viernes 01/06/2018	Martes 05/06/2018	Luis Miguel Pinto
- **Diseño del prototipo**	Jueves 07/06/2018	Martes 12/06/2018	Luis Miguel Pinto
- **Implementación del prototipo**	Miércoles 13/06/2018	Viernes 22/06/2018	Luis Miguel Pinto
- **Pruebas del prototipo**	Lunes 25/06/2018	Martes 26/06/2018	Luis Miguel Pinto
Evaluación de la solución	Miércoles 27/06/2018	Viernes 29/06/2018	Luis Miguel Pinto
Sprint 6	Lunes 02/07/2018	Viernes 27/07/2018	Luis Miguel Pinto
- **Análisis de la situación**	Lunes 02/07/2018	Jueves 05/07/2018	Luis Miguel Pinto
- **Diseño del prototipo**	Viernes 06/07/2018	Martes 10/07/2018	Luis Miguel Pinto
- **Implementación del prototipo**	Miércoles 11/07/2018	Miércoles 25/07/2018	Luis Miguel Pinto
- **Pruebas del prototipo**	Jueves 26/07/2018	Viernes 27/07/2018	Luis Miguel Pinto
Evaluación de la solución	Lunes 30/07/2018	Martes 31/07/2018	Luis Miguel Pinto
Sprint Final	Miércoles 01/08/2018	Lunes 10/09/2018	Luis Miguel Pinto
- **Análisis de la situación**	Miércoles 01/08/2018	Miércoles 08/08/2018	Luis Miguel Pinto
- **Diseño del producto final**	Jueves 09/08/2018	Viernes 17/08/2018	Luis Miguel Pinto
- **Implementación del producto final**	Lunes 20/08/2018	Miércoles 05/09/2018	Luis Miguel Pinto
- **Pruebas del producto final**	Jueves 06/09/2018	Lunes 10/09/2018	Luis Miguel Pinto

Evaluación del producto final	Martes 11/09/2018	Viernes 14/09/2018	Luis Miguel Pinto

4.1.2. Diagrama Gantt

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Realización del documento**	**180 días**	**lun 08/01/18**	**vie 14/09/18**
2	Introducción	42 días	lun 08/01/18	mar 06/03/18
3	Marco regulador	9 días	mar 06/03/18	vie 16/03/18
4	Estado del arte	10 días	lun 19/03/18	vie 30/03/18
5	Organización del proyecto	5 días	lun 02/04/18	vie 06/04/18
6	Desarrollo del proyecto	105 días	lun 09/04/18	vie 31/08/18
7	Soluciones previas propuestas	3 días	lun 03/09/18	mié 05/09/18
8	Análisis de los resultados	2 días	jue 06/09/18	vie 07/09/18
9	Conclusiones y problemas encontrados	2 días	lun 10/09/18	mar 11/09/18
10	Apartado en inglés	3 días	mié 12/09/18	vie 14/09/18
11	Revisión de la documentación	4 días	lun 17/09/18	jue 20/09/18
12	**Sprint 1**	**17 días**	**jue 01/02/18**	**vie 23/02/18**
13	Análisis de la situación	3 días	jue 01/02/18	lun 05/02/18
14	Diseño del prototipo	4 días	mar 06/02/18	vie 09/02/18
15	Implementación del prototipo	8 días	lun 12/02/18	mié 21/02/18
16	Pruebas del prototipo	2 días	jue 22/02/18	vie 23/02/18
17	Evaluación de la solución	3 días	lun 26/02/18	mié 28/02/18
18	**Sprint 2**	**20 días**	**jue 01/03/18**	**mié 28/03/18**
19	Análisis de la situación	2 días	jue 01/03/18	vie 02/03/18
20	Diseño del prototipo	5 días	lun 05/03/18	vie 09/03/18
21	Implementación del prototipo	10 días	lun 12/03/18	vie 23/03/18
22	Pruebas del prototipo	3 días	lun 26/03/18	mié 28/03/18
23	Evaluación de la solución	2 días	jue 29/03/18	vie 30/03/18
24	**Sprint 3**	**19 días**	**lun 02/04/18**	**jue 26/04/18**
25	Análisis de la situación	3 días	lun 02/04/18	mié 04/04/18
26	Diseño del prototipo	4 días	jue 05/04/18	mar 10/04/18
27	Implementación del prototipo	8 días	mié 11/04/18	vie 20/04/18
28	Pruebas del prototipo	4 días	lun 23/04/18	jue 26/04/18
29	Evaluación de la solución	2 días	vie 27/04/18	lun 30/04/18
30	**Sprint 4**	**20 días**	**mar 01/05/18**	**lun 28/05/18**
31	Análisis de la situación	4 días	mar 01/05/18	vie 04/05/18
32	Diseño del prototipo	5 días	lun 07/05/18	vie 11/05/18
33	Implementación del prototipo	8 días	lun 14/05/18	mié 23/05/18
34	Pruebas del prototipo	3 días	jue 24/05/18	lun 28/05/18
35	Evaluación de la solución	3 días	mar 29/05/18	jue 31/05/18
36	**Sprint 5**	**18 días**	**vie 01/06/18**	**mar 26/06/18**
37	Análisis de la situación	3 días	vie 01/06/18	mar 05/06/18
38	Diseño del prototipo	4 días	mié 06/06/18	lun 11/06/18
39	Implementación del prototipo	8 días	mar 12/06/18	jue 21/06/18
40	Pruebas del prototipo	2 días	vie 22/06/18	lun 25/06/18
41	Evaluación de la solución	3 días	mié 27/06/18	vie 29/06/18
42	**Sprint 6**	**20 días**	**lun 02/07/18**	**vie 27/07/18**
43	Análisis de la situación	4 días	lun 02/07/18	jue 05/07/18
44	Diseño del prototipo	3 días	vie 06/07/18	mar 10/07/18
45	Implementación del prototipo	11 días	mié 11/07/18	mié 25/07/18
46	Pruebas del prototipo	2 días	jue 26/07/18	vie 27/07/18
47	Evaluación de la solución	2 días	lun 30/07/18	mar 31/07/18
48	**Sprint Final**	**29 días**	**mié 01/08/18**	**lun 10/09/18**
49	Análisis de la situación	6 días	mié 01/08/18	mié 08/08/18
50	Diseño del producto final	7 días	jue 09/08/18	vie 17/08/18
51	Implementación del producto final	13 días	lun 20/08/18	mié 05/09/18
52	Pruebas del producto final	3 días	jue 06/09/18	lun 10/09/18
53	Evaluación del producto final	4 días	mar 11/09/18	vie 14/09/18

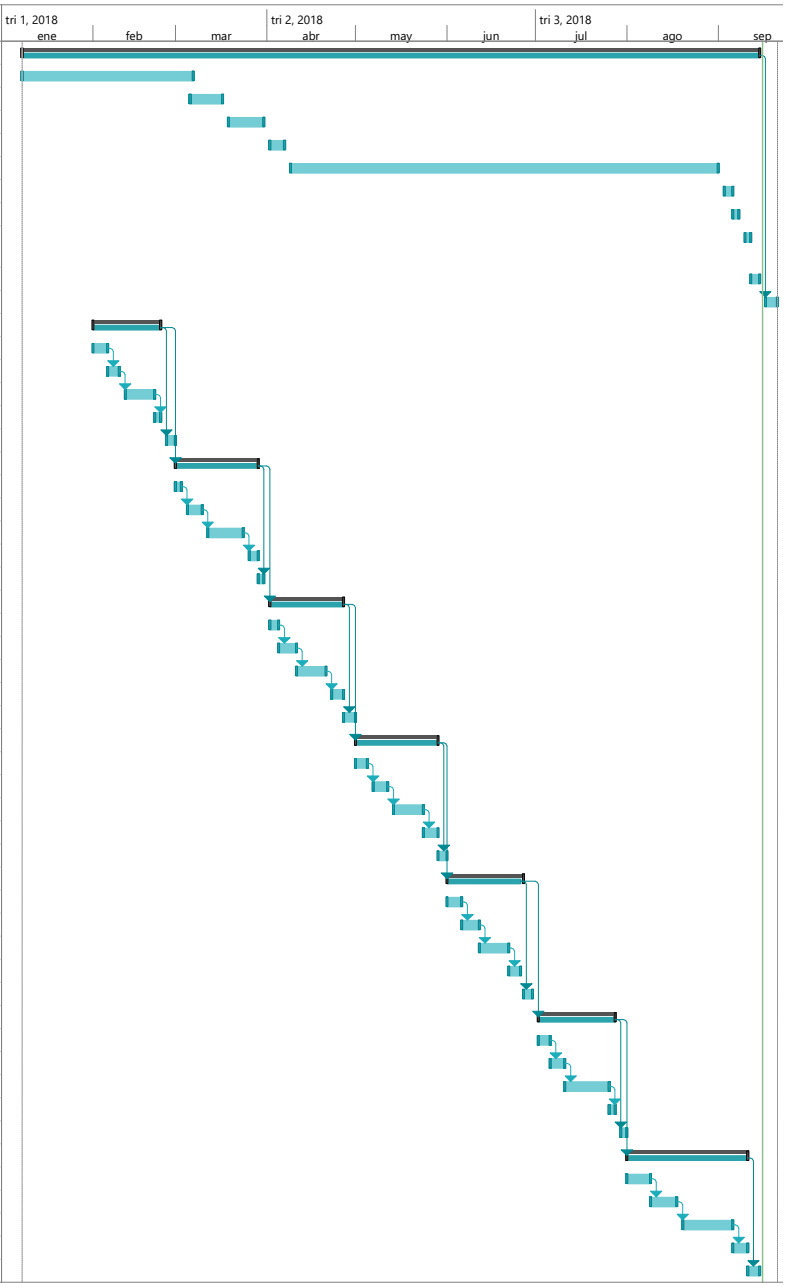

Ilustración 12. Diagrama Gantt de la planificación del proyecto

En el diagrama de la figura superior, se puede observar todas las tareas disponibles en el proyecto y los días necesarios para cada una. También se puede observar las dependencias de cada tarea con sus anteriores.

Tabla I. Planificación del proyecto

4.2. ORGANIZACIÓN DEL PROYECTO

Se expone un esquema que permite visualizar la jerarquía del proyecto incluyendo todos los recursos humanos que serán necesarios.

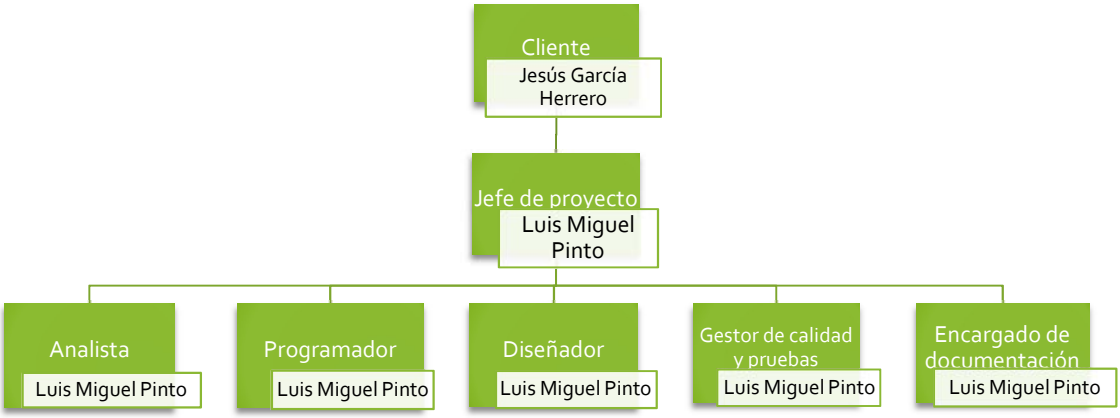

Ilustración 13. Organización del proyecto

4.3. PRESUPUESTO

En este apartado, se expondrán los gastos estimados para la realización del proyecto. Se incluirán todos los gastos de los empleados, los gastos para el material hardware y el coste de todas las licencias de productos software que también serán necesarios.

4.3.1. Recursos humanos

Como se ha mencionado anteriormente, la jornada de los trabajadores no incluye trabajar durante los fines de semana y se trabajarán como máximo 4 horas diarias lo que significa 20 horas semanales, que equivale a un horario de media jornada. Aunque algunas tareas, requerirán menos tiempo de unos responsables que de otros por lo que en algunos casos no se llevarán a cabo las 20 horas semanales en cada puesto de trabajo.

El apartado anterior, consistía en un esquema que mostraba la jerarquía de los puestos que se ocupan en este proyecto. Ahora, se procese a explicar en qué consiste cada uno de estos puestos:

1. **Cliente:** el cliente en este caso será Jesús García Herrero ya que será el encargado de dejar las pautas para la realización del proyecto. Estipula los requisitos que debe cumplir el sistema y llevará a cabo la supervisión continuada de los sprints dando el visto bueno en cada una o aportando nuevas ideas que quiera implantar. Este puesto no estará asalariado de ninguna forma durante el proyecto.

2. **Jefe de proyecto:** esta tarea se llevará a cabo por Jesús García Herrero y Luis Miguel Pinto. Se encargarán de supervisar el proyecto en todo momento, tomando las decisiones necesarias para el correcto funcionamiento del mismo. Guiarán al equipo para conseguir terminar las tareas de manera correcta en los plazos estimados.

3. **Analista:** esta función será llevada a cabo por Luis Miguel Pinto. Será el responsable de conocer el funcionamiento del radar y del software con el que se complementa casi a la perfección, para asegurarse de que el sistema y las pruebas avanzan en la dirección adecuada. Aunque no es un jefe como tal del resto del equipo (por eso no se refleja en el diagrama de la jerarquía), en ausencia del Jefe de proyecto, será el encargado de liderar al resto de los componentes.

4. **Programador:** puesto ocupado por Luis Miguel Pinto. Se encargará de implementar en todos los sprints el diseño actual para el prototipo, o en caso de ser el último sprint, el producto final. Configurará el radar para recibir los datos que necesitamos y parsearlos para que podamos entenderlos y trabajar con ellos. Se encargará de realizar todo el código sobre el radar (principalmente en lenguaje C) y, con los datos obtenidos, los scripts correspondientes en Matlab para su análisis.

5. **Diseñador:** Luis Miguel Pinto se encargará de diseñar una mejora para el prototipo del sprint anterior de forma que el producto avance de forma continua en cada iteración. Debe estar en constante contacto con el programador ya que debe explicarle exactamente como realizar la implementación en el caso de que surja alguna duda para no incurrir en errores no deseados.

6. **Gestor de calidad y pruebas:** será Luis Miguel Pinto el encargado de este rol. Se asegurará que el producto va en todo lugar en la dirección deseada y cumpliendo los requisitos que son las características marcadas por el cliente para su sistema software. También se encargará de llevar a cabo todas las pruebas del producto para englobar todos los sucesos posibles y los casos más extremos. Dejando constancia de todas estas pruebas para el conocimiento del resto del equipo.

7. **Encargado de la documentación:** el responsable será Luis Miguel Pinto. Como el nombre del puesto indica, se encargará de realizar durante todo el avance del proyecto la documentación del mismo. De forma que todo lo realizado en cada tarea quede plasmado en un informe que podrán leer en cualquier momento para tener conocimiento de todo lo acontecido.

Para establecer el salario de cada empleado, se acude al BOE, publicado el 6 de marzo de 2018 del que se obtiene una tabla con los salarios del área de Consultoría, Desarrollo y Sistemas.

Nuevo Sistema de Clasificación Profesional			Equivalencia con las categorías del anterior convenio solo a efectos de trasposición	Salarios Nuevo Convenio antes de incrementos		
Area de Actividad	Grupo	Nivel		Salario Base	Plus Convenio	Salario Total
AREA 3: Consultoría, Desarrollo y sistemas	A			23.362,50	1.637,50	25.000,00
	B	I		22.673,75	1.576,25	24.250,00
	B	II	Analista; Analista de sistemas (Grupo III)	21.969,50	1.536,22	23.505,72
	C	I	Analista programador; Diseñador pagina web (Grupo III)	21.555,66	1.438,08	22.993,74
	C	II		20.091,75	1.408,25	21.500,00
	C	III		18.222,75	1.277,25	19.500,00
	D	I	Programador senior, Jefe de Operación, Programador en Internet (Grupo III)	15.442,56	1.089,20	16.531,76
	D	II	Deliniante-proyectista (Grupo III)	14.295,54	998,48	15.294,02
	D	III		14.013,90	986,10	15.000,00
	E	I	Programador junior; Técnico de mantenimiento web; Operador de ordenador (Grupo III)	13.827,66	973,00	14.800,66
	E	II	Administrador de Test (Grupo III)	11.773,15	828,39	12.601,54
	E	II	Tabulador de Ordenador,.....Operador de periféricos (Grupo IV)			
	E	III	Codificador informático, Perforista, Verificador, Clasificador y Grabador (Grupo IV)	9.811,06	687,26	10.498,32
	E	III	Calcador (Grupo IV)	9.644,85	675,30	10.320,15

Ilustración 14. Salarios según el BOE para el año 2018

Con esta información, se procede a distribuir el sueldo a los empleados según las horas ejercidas a lo largo del proyecto.

Queda incluir que el Sueldo Total, se refiere al sueldo bruto ya que, tras incluir las retenciones de la Seguridad Social, se obtendrá el sueldo que recibirá el empleado. Aunque lo que costará ese empleado al proyecto será el suelto bruto en sí.

De forma que los sueldos brutos y netos de los empleados serían los siguientes (coste/hora brutos):

Puesto de trabajo	Horas al día en el puesto de trabajo
Jefe de proyecto	1
Analista	1
Programador	2
Diseñador	2
Gestor de calidad y pruebas	2
Encargado documentación	1

Tabla II. Horas que cada empleado trabaja por día

Puesto de trabajo	Grupo-Nivel	Coste/ hora (bruto)	Coste/ hora (neto)	Horas en proyecto	Sueldo bruto (euros)	Retenciones
Jefe de proyecto	B - I	12,6	10,35	164	2.066,4	14,13%
Analista	B - II	12,2	9,83	30	366	13,29%
Programador	E - I	7,71	6,6	136	1048,5	7,22%
Diseñador	C – I	11,9	9,6	64	761,6	13,09%
Gestor de calidad y pruebas	C - I	11,9	9,6	40	476	13,09%
Encargado documentación	D - III	7,8	6,7	164	1279,2	7,64%
TOTAL					5997,7	

Tabla III. Presupuesto para los recursos humanos

Como se puede observar, personal con un puesto muy por debajo de otros tienen unos ingresos mayores. Esto es porque el proyecto necesitará más horas y empeño de estas personas para que el proyecto sea eficaz y eficiente.

A este precio, habría que sumarle los viajes y las dietas. A cada empleado se le proporcionará 6 euros al día para las comidas (a los empleados que trabajen por lo menos las 4 horas diarias) y 2 euros diarios (a todos los empleados que se desplacen, en este caso a la Universidad Carlos III desde Las Rozas) por cada viaje que tenga que realizar o, en algunos casos, si el trabajador tuviera que desplazarse a otro lugar, incluso alguna provincia diferente. Para estos desplazamientos, se reserva una cantidad de dinero. Puesto que no se consideran necesarios gastos de alojamiento, solo habrá gastos de manutención y transporte. En este caso, el empleado siempre es el mismo para todos los puestos, pero supongamos que son personas distintas para calcular las dietas de una forma real.

Puesto de trabajo	Días en el proyecto	Manutención	Transporte	Total (horas)
Jefe de proyecto	164	0	328	328
Analista	30	0	60	60
Programador	68	408	136	544
Diseñador	32	0	64	64
Gestor de calidad y pruebas	20	228	76	304
Encargado documentación	164	0	328	328
Viajes de larga duración				2000
TOTAL				3628

Tabla IV. Dietas y transporte en el proyecto

4.3.2. Costes del hardware

HARDWARE	MODELO	COSTE (euros)
Ordenador portátil	MSI GE60 2PE Apache Pro	0
RADAR	SRR 208 Continental	3500
ST Link	ST-Link/V2	19,30
PixHawk	PX4	102,95
Fuente de alimentación regulable	CSI3005EIII	189,54
Cables	JTAG, USB-MicroUSB x2, CAN bus.	27,80
TOTAL		3839,59

Tabla V. Costes del hardware

El ordenador portátil tiene un coste de 1200 euros, pero ya había sido adquirido antes del comienzo del proyecto. El resto de los componentes, han sido adquisiciones puntuales para este sistema.

4.3.3. Costes del software

SOFTWARE	MODELO	COSTE (euros)
Sistema operativo	Ubuntu Linux 16.04 LTS	0
Editor de código	Sublime Text 3	0
Entorno de programación del microcontrolador	Atollic Studio	0
Ofimática	Microsoft Office 365 Personal	69
Alojamiento de archivos	Google Drive	0
MATLAB	Licencia MatLab Educación	35
TOTAL		104

Tabla VI. Costes del software

El sistema operativo que se usará es Ubuntu. El cual es de código libre y por lo tanto de libre acceso. Lo mismo sucede con el editor de texto elegido y el entorno para programación y pruebas del sistema.

Respecto a la realización del documento, se usará un paquete de Microsoft Office el cual tiene un precio de 69 euros anuales.

Por último, para el almacenamiento de los distintos ficheros para que pueda verse y modificarse por todos los usuarios, se utilizará Google Drive, servicio que también es gratuito.

4.4. IDENTIFICACIÓN DE REQUISITOS

4.4.1. Justificación de los requisitos

La clasificación de los requisitos se llevará a cabo haciendo dos grupos diferentes de ellos. Por un lado, estarán los requisitos funcionales (los cuales indican qué hay que hacer) y los requisitos no funcionales (los cuales indican cómo hacerlo).

A su vez, habrá varios subapartados para la diferenciación de los requisitos que serán los siguientes: control, componentes hardware, código software e interfaz.

4.4.2. Justificación de la plantilla de requisitos

La plantilla para la especificación de los requisitos sobre la que trabajaremos será la siguiente:

ID	RF-XX | RNF-XX
Nombre	Nombre descriptivo del requisito.
Origen	Origen del requisito (cliente, analista…).
Prioridad	Prioridad del requisito (alta, baja).
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	Descripción detallada del requisito.
Fecha creación	DD-MM-YYYY
Fecha modificación	DD-MM-YYYY
Responsable	Nombre del responsable del requisito

Tabla VII. Ejemplo requisitos

- ID: en el encabezado de cada tabla, se detallará un identificador para reconocer cada requisito. Por una parte, podrán ser requisitos funcionales (RF) seguidos de un número cronológico para distinguirlos entre ellos (XX). Por otro lado, de la misma forma tendremos los requisitos no funcionales (RNF) que tendrán un identificador numérico al igual que los funcionales.

- Nombre: cada requisito obtendrá un nombre lo más descriptivo posible para que no haga falta leer todo el requisito para saber de qué trata.

- Prioridad: cada requisito tendrá un campo acorde a la prioridad de este en el proyecto. Puede ser Alta o Baja y esto determina la importancia de este en el desarrollo y por lo tanto, el orden en el que se deben realizar.

- Pruebas de verificación: en este apartado, se incluirán los identificadores de todas las pruebas que se harán para probar que cada requisito ha sido realizado y ha finalizado con las expectativas esperadas.

- Descripción: se incluirá una descripción detallada del requisito. Qué debe hacer específicamente dejando claro todo. Sin ambigüedades.

- Fecha de creación: fecha de creación del requisito con el formato (DD-MM-YYYY).

- Fecha de modificación: en el caso de que algún requisito necesite una modificación posterior a su creación, habrá que dejar indicada la fecha de la misma en el mismo formato que la fecha de creación.

- Responsable: nombre del responsable de realizar el requisito para, en caso de algún fallo o cuestión importante, acudir rápidamente al responsable del requisito.

Para verificar el cumplimiento de todos los requisitos, una vez finalizado el proyecto, se realizarán una serie de pruebas tanto del software como del hardware que corroborarán que los requisitos han sido cubiertos correctamente.

La plantilla de estas pruebas es la siguiente:

ID	**PF-XX | PNF-XX**
Requisito	Nombre del requisito al que hace alusión.
Precondición	Estado antes de la prueba.
Acción	Acción para garantizar que la prueba funciona.
Postcondición	Estado tras la prueba.

Tabla VIII. Ejemplo prueba de requisitos

Puesto que los requisitos no funcionales no siempre pueden demostrarse, algunos pueden no contener una prueba verificadora.

4.4.3. Requisitos funcionales

4.4.3.1. Control

En este apartado, se detallarán los requisitos funcionales los cuales tratan sobre el control del usuario sobre el sistema.

ID	RF-01
Nombre	Configuración de datos RADAR
Origen	Cliente
Prioridad	Alta
Pruebas de verificación	PF01, PF02
Descripción	El sistema debe permitir al usuario modificar la configuración el radar, de forma que elija la forma en la que envía los datos. Ya sean en cluster o en track.
Fecha creación	23-03-2018
Fecha modificación	23-03-2018
Responsable	Luis Miguel Pinto

Tabla IX. Requisito Funcional 01

ID	PF-01
Requisito	RF-01
Precondición	Radar utilizando la configuración de datos de salida “SendTracks”
Acción	Presionar la tecla 0 mientras el PixHawk estaba enviando datos de tracks.
Postcondición	El radar modifica su configuración de datos de salida a “SendCluster” y comienza a enviar mensajes de clusters.

Tabla X. Prueba Requisito Funcional 01

ID	PF-02
Requisito	RF-01
Precondición	Radar utilizando la configuración de datos de salida “SendCluster”
Acción	Presionar la tecla 1 mientras el PixHawk estaba enviando datos de clusters.
Postcondición	El radar modifica su configuración de datos de salida a “SendTracks” y comienza a enviar mensajes de tracks.

Tabla XI. Prueba Requisito Funcional 02

ID	RF-02
Nombre	Encendido apagado RADAR
Origen	Cliente
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	El cliente podrá parar en todo momento el paso de datos del radar. No mediante la anulación de alimentación al radar, sino incluyendo un mecanismo que pueda realizar esta operación.
Fecha creación	23-03-2018
Fecha modificación	23-03-2018
Responsable	Luis Miguel Pinto

Tabla XII. Requisito Funcional 02

ID	PF-03
Requisito	RF-02
Precondición	El radar se encuentra enviando información a través del PixHawk al ordenador.
Acción	Presionar el botón incorporado manualmente al PixHawk.
Postcondición	El envío de datos se cancela y se regresa al menú.

Tabla XIII. Prueba Requisito Funcional 03

4.4.3.2. Código software

En este apartado, incluiremos los requisitos del código.

ID	RF-03
Nombre	Optimización del código
Origen	Analista
Prioridad	Baja
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	El sistema debe recoger información del radar en todo momento. Será envío de información a tiempo real de forma que nunca podrá tener demasiado procesamiento y poder perder algún dato. Por lo que el sistema debe estar optimizado de forma que el tiempo máximo de espera entre datos sea de 0,2s.
Fecha creación	23-03-2018
Fecha modificación	23-03-2018
Responsable	Luis Miguel Pinto

Tabla XIV. Requisito Funcional 03

ID	PF-04
Requisito	RF-03
Precondición	Código sin estructurar. Medición del tiempo de compilación.
Acción	Incluir medidas de optimización de código.
Postcondición	El tiempo final de configuración del software disminuye notablemente.

Tabla XV. Prueba Requisito Funcional 04

ID	RF-04
Nombre	Parseo de datos bits a hexadecimal
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	Dado que el radar envía los datos como bits y esto es algo muy difícil de comprender para el ojo humano, el sistema deberá parsear todos estos datos a formato hexadecimal.
Fecha creación	23-03-2018
Fecha modificación	23-03-2018
Responsable	Luis Miguel Pinto

Tabla XVI. Requisito Funcional 04

ID	PF-05
Requisito	RF-04
Precondición	El radar envía la información de lo que detecta y el PixHawk lo entrega al ordenador con los datos en crudo.
Acción	Se aplica el software perteneciente al primer sprint.
Postcondición	El radar modifica su configuración de datos de salida a “SendTracks” y comienza a enviar mensajes de tracks.

Tabla XVII. Prueba Requisito Funcional 05

ID	RF-05
Nombre	Parseo de datos hexadecimal a datos en formato decimal
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	Una vez tenemos los datos en formato hexadecimal, ya se sabrá a qué tipo de dato corresponde cada uno. Por lo que se debe identificar primero a qué se refiere y darles el significado en metros, grados o letras (de un mensaje) según corresponda y, por último, convertirlos a base decimal para poder trabajar con ellos.
Fecha creación	23-03-2018
Fecha modificación	23-03-2018
Responsable	Luis Miguel Pinto

Tabla XVIII. Requisito Funcional 05

ID	PF-06
Requisito	RF-05
Precondición	El PixHawk envía los datos al ordenador en hexadecimal
Acción	Programar el microcontrolador con el software perteneciente al sprint final.
Postcondición	Los datos recibidos por el ordenador son los valores decimales de estos datos.

Tabla XIX. Prueba Requisito Funcional 06

ID	RF-06
Nombre	Guardado de los datos en un fichero .csv
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	Para el posterior procesamiento y análisis de los datos, una vez transformados los datos en crudo a datos decimales, deben escribirse en un fichero .csv.
Fecha creación	23-03-2018
Fecha modificación	23-03-2018
Responsable	Luis Miguel Pinto

Tabla XX. Requisito Funcional 06

ID	PF-07
Requisito	RF-06
Precondición	Se reciben los datos y se visualizan por pantalla.
Acción	Se ejecuta el comando 'screen -L -Logfile archivo.csv /dev/ttyACM0 115200
Postcondición	Todos los datos que se reciben se guardan en el archivo especificado.

Tabla XXI. Prueba Requisito Funcional 07

ID	RF-07
Nombre	Análisis de datos
Origen	Cliente
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	Se creará un script en el entorno de Matlab para analizar los datos del radar y crear gráficas con las que poder trabajar.
Fecha creación	23-03-2018
Fecha modificación	23-03-2018
Responsable	Luis Miguel Pinto

Tabla XXII. Requisito Funcional 07

ID	PF-08
Requisito	RF-07
Precondición	Se tiene el fichero .csv con todos los valores de los datos
Acción	Se ejecuta el script de Matlab creado
Postcondición	Se generan las gráficas correspondientes a los datos introducidos.

Tabla XXIII. Prueba Requisito Funcional 08

4.4.3.3. Componentes Hardware

ID	RF-08
Nombre	Modificación PixHawk para compatibilidad con el radar
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	El componente, deberá modificarse para tener un puerto CAN del que recibirá los datos del radar y mediante un puerto micro USB a USB pasará los datos al ordenador. Por otro lado, se habilitará un input para un cable JTAG para que el ST-link.
Fecha creación	28-03-2018
Fecha modificación	28-03-2018
Responsable	Luis Miguel Pinto

Tabla XXIV. Requisito Funcional 08

ID	PF-09
Requisito	RF-08
Precondición	El PixHawk no dispone de un puerto para conectar el programador ST-Link.
Acción	Se conecta el ST-Link al puerto creado.
Postcondición	El PixHawk puede ser programado correctamente por el ST-Link.

Tabla XXV. Prueba Requisito Funcional 09

ID	RF-09
Nombre	Regulación de la fuente de alimentación
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	Las medidas para la conexión de la fuente de alimentación y el radar deben ser de 12V y 0,5A. En ningún caso puede exceder dichos valores.
Fecha creación	28-03-2018
Fecha modificación	28-03-2018
Responsable	Luis Miguel Pinto

Tabla XXVI. Requisito Funcional 09

ID	PF-10
Requisito	RF-09
Precondición	El radar se encuentra sin suministro de energía.
Acción	Se conecta la fuente de alimentación al radar con unos valores de 12 voltios y 0,5 amperios.
Postcondición	El radar funciona correctamente.

Tabla XXVII. Prueba Requisito Funcional 10

4.4.4. Requisitos no funcionales

4.4.4.1. Componentes Hardware

ID	RNF-01
Nombre	Funcionamiento dependiente del ordenador
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	El radar deberá estar conectado mediante la conexión USB al ordenador ya que será la única forma de traspaso de datos.
Fecha creación	28-03-2018
Fecha modificación	28-03-2018
Responsable	Luis Miguel Pinto

Tabla XXVIII. Requisito No Funcional 01

ID	PNF-01
Requisito	RNF-01
Precondición	El radar envía los datos al PixHawk, pero este no los traslada a ningún lado.
Acción	Se conecta el PixHawk al ordenador
Postcondición	El PixHawk empieza a enviar todos los mensajes del radar al ordenador.

Tabla XXIX. Prueba Requisito No Funcional 01

ID	RNF-02
Nombre	Coste bajo de los componentes.
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	Excluyendo el radar, el resto de los componentes deben ser lo más estrictamente ajustados en relación calidad-precio para que el coste total del proyecto no se exceda.
Fecha creación	28-03-2018
Fecha modificación	28-03-2018
Responsable	Luis Miguel Pinto

Tabla XXX. Requisito No Funcional 02

ID	RNF- 03
Nombre	Sistema compacto
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	Sin que afecte al sistema, este deberá ocupar el menor espacio posible ya que su finalidad es ir instalado en un coche.
Fecha creación	28-03-2018
Fecha modificación	28-03-2018
Responsable	Luis Miguel Pinto

Tabla XXXI. Requisito No Funcional 03

ID	PNF-02
Requisito	RNF-02
Precondición	Se trabaja con los componentes en un aula de la universidad.
Acción	Se traslada el equipo fácilmente a una vivienda para continuar con el desarrollo.
Postcondición	El equipo sigue funcionando correctamente.

Tabla XXXII. Prueba Requisito No Funcional 02

ID	RNF- 04
Nombre	Gráficas con curva de entendimiento poco pronunciada
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	Las gráficas deberán ser simples de forma que el cliente pueda entenderlas rápidamente y verificar si los datos se corresponden con lo esperado.
Fecha creación	28-03-2018
Fecha modificación	28-03-2018
Responsable	Luis Miguel Pinto

Tabla XXXIII. Requisito No Funcional 04

ID	PNF-03
Requisito	RNF-03
Precondición	Se ve la gráfica por primera vez.
Acción	Se realiza una breve explicación.
Postcondición	Se entiende totalmente la gráfica y su significado.

Tabla XXXIV. Prueba Requisito No Funcional 03

4.4.4.2. Código software

ID	RNF- 05
Nombre	Funcionamiento en Ubuntu
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	Todo el sistema, deberá funcionar en el sistema operativo Linux, específicamente en la distribución de Ubuntu.
Fecha creación	28-03-2018
Fecha modificación	28-03-2018
Responsable	Luis Miguel Pinto

Tabla XXXV. Requisito No Funcional 05

ID	PNF-04
Requisito	RNF-05
Precondición	El software ha sido creado.
Acción	Se prueba a ejecutarlo en un sistema operativo Linux (distribución Ubuntu)
Postcondición	El programa funciona correctamente.

Tabla XXXVI. Prueba Requisito No Funcional 04

ID	RNF- 06
Nombre	Funciones del código atómicas
Origen	Analista
Prioridad	Alta
Pruebas de verificación	Pruebas para el requisito en cuestión.
Descripción	El código debe estar bien estructurado. Todas las funciones deben ser atómicas y con nombres descriptivos.
Fecha creación	28-03-2018
Fecha modificación	28-03-2018
Responsable	Luis Miguel Pinto

Tabla XXXVII. Requisito No Funcional 06

4.5. TECNOLOGÍAS EMPLEADAS

En este apartado, se especificarán todas las tecnologías que se utilizarán en el transcurso del proyecto. Estas tecnologías, pertenecen a uno de los 2 campos que trata este proyecto: recogida de datos y análisis de datos.

4.5.1. PixHawk

Esta herramienta es un controlador de hardware libre pensada para vehículos autopilotados e investigaciones en comunidades académicas. Ya sean drones o coches autónomos, por ejemplo. Destacan entre otras cosas por su bajo coste, gran rendimiento, alta disponibilidad y flexibilidad en los distintos componentes externos (sensores) a los que se puede conectar. También tiene como beneficio clave la capacidad de ser altamente personalizable y por disponer de una actualización automatizada al firmware más reciente a través de QGroundControl.

Se han creado varias versiones de estas placas. Pero versiones más recientes, no tienen por qué significar mejor capacidad o procesamiento. Simplemente se diferencian en el cableado de los conectores, ya que tienen distintos objetivos finales.

Ilustración 15. Pixhawk versión PX4 [28]

En este caso, se hará uso de PixHawk 4. Al ser orientado para instituciones académicas, no hace falta saber mucho sobre las versiones.

Por un lado, QGroundControl descarga automáticamente el firmware correcto para el autopilot conectado.

Y, por otro lado, el único conocimiento requerido es el conocimiento sobre los conectores de la placa ya que las restricciones son básicamente físicas.

La licencia de este componente permite la utilización de casi cualquier forma deseada siempre que se compartan los cambios y se publique con la misma licencia de código.

El autopilot PX4 que se usará en este proyecto, dispone de sensores internos para medir altitud, posición y provee algoritmos de guía, navegación y control para los distintos tipos de posibles vehículos y otros algoritmos con funciones de procesamiento de datos que incluyen posibles filtros programados.

Sensor	Type	Axes	Scale	ADC accuracy	Data rate
L3GD20H	gyroscope	3	2000 dps	16 bits	760 Hz
LSM303D	accelerometer/ magnetometer	6	± 16g / ± 2gauss	16 bits	1600 Hz/ 100 Hz
MPU-6000	accelerometer/ gyroscope	6	± 16g / 2000 dps	16 bits	1000 Hz/ 8000 Hz
MS5611	barometer	1	1200 mbar	24 bits	1000 Hz

Ilustración 16. Sensores internos integrados en PixHawk [38]

Además de los distintos sensores internos integrados mostrados en la ilustración anterior, cuenta con varios puertos para la conexión de los componentes externos o periféricos que harán que el vehículo sea más fiable en cuanto a los datos recibidos.

Por un lado, cuenta con algunos conectores específicos para ciertos periféricos como: conectores para comunicaciones telemétricas, un socket para recibir espectros y un conector para GPS.

Por otro lado, tiene conectores más genéricos como: receptores asíncronos universales (puertos UART), interfaces periféricas seriales (SPI), conector para circuito inter-integrado (I2C), un conector de bus universal en serie (USB), un puerto de "Controller Area Network" (CAN), que es el que se usará juntos al puerto USB en nuestro proyecto, y dos conectores ADC de 3,3 y 6,6V respectivamente.

En este caso, se incluye el radar que se conectará al puerto CAN del PixHawk que a la vez irá conectado por medio de USB al ordenador. De forma que enviará los datos que recibe del radar realizando este recorrido. Además, para programar el PixHawk y elegir cómo queremos que nos envíe la información con los datos, conectaremos el ST-Link por medio de un puerto que ha sido creado a base de soldar distintos pines y adaptándolo para que se pueda incluir un cable JTGA que irá conectado a ambos extremos (uno al ST-Link y otro al PixHawk).

El radar aportará al vehículo más precisión con las distancias y la situación de los objetos que tiene delante. Mayor precisión en los ángulos y velocidades de estos objetos y detección de posibles colisiones y de esta forma, poder evitarlas. [9] [10] [38]

4.5.2. ST-Link

Es un depurador y programador en el circuito para la familia de microcontroladores. En este caso, este componente se usará para programar el PixHawk y conseguir que envíe los datos de la forma que se necesita. Incluso pasar un filtrado para recoger solo los datos que se consideran necesarios.

Mediante un cable JTGA, estará conectado al PixHawk y, mediante un cable con puertos USB-microUSB estará conectado al ordenador que enviará el código que se quiere implementar en el controlador. El ST-Link programará el PixHawk con el algoritmo deseado y, de esta forma, se recibirá la información tal y como se ha establecido. [11]

Ilustración 17. ST-Link [29]

4.5.3. Microsoft Office

Como herramienta de ofimática, se ha utilizado Microsoft Office. Este paquete ofrece un gran número de programas muy útiles, pero en este caso utilizaremos los siguientes:

- Microsoft Word: este programa será el elegido para la realización de toda la memoria del proyecto. Se puede aplicar cualquier formato deseado en cualquier parte del documento y dispone de varias herramientas de texto y procesamiento del mismo. Dando la oportunidad de utilizar un lenguaje mucho más estructurado.

- Microsoft Excel: aplicación de hojas de cálculo utilizada mayormente para tareas contables y financieras. Ya que dispone de innumerables formulas, gráficos y es hasta programable. En este caso, solo se utilizará para los ficheros en los que guardarán los datos procedentes del RADAR gracias a su facilidad de guardar datos en columnas.

- Microsoft Project: aplicación de administración y gestión de proyectos. Se basa en diseño y desarrollo de proyectos y asignación de recursos humanos para una correcta utilización de los mismos. En este caso, el diagrama Gantt incluido en el documento, estará realizado con esta herramienta.

4.5.4. Atollic TrueStudio

Esta herramienta está diseñada en el marco de Eclipse IDE. Admite diferentes tipos de proyectos ofreciendo una gran flexibilidad.

Gracias a la potente interfaz gráfica, se puede crear un proyecto y ejecutar el código de una forma rápida. Mientras que los potentes motores para compilación y depuración brindan la posibilidad de tener estructuras de código muy complejas.

Dispone de una infinidad de herramientas para la edición/navegación/refactorización del código. Y un potente indexador para lenguajes C y C++. Por ello, cuenta con un compilador C/C++ muy optimizado y varias herramientas de línea de comandos.

A la hora de procesar, cuenta con un depurador profesional de C/C++. También ofrece soporte de depuración para sistemas fuera de Atollic TrueSudio como por ejemplo, para ST-Link el cual es nuestro caso. [12] [13] [14]

4.5.5. Matlab

Se trata de una herramienta para procesado de datos muy potente. En este caso, será utilizada para realizar gráficas con los datos proporcionados por el radar para un entendimiento más fácil por parte de los usuarios de qué ha ocurrido.

4.5.6. Lenguaje programación C

Se utilizará este lenguaje de programación para programar el controlador PixHawk por medio del ST-Link. Es un lenguaje muy útil y que dispone de varias librerías muy útiles para estos casos.

4.5.7. Librería stm32f4xx_stdPeriph_Driver

Librería disponible para Atollic TrueStudio. Consiste en una serie de drivers que cubren periféricos a bajo nivel.

Cubre 3 niveles de abstracción y nos da la posibilidad de un mapeo de direcciones con todos sus registros y sus bits declarados en C. Además, incluye un archivo de mapeo que ayuda a que el debugeo sea más rápido.

Por último, incluye varias colecciones de rutinas y estructuras de datos que cubren todas las funciones de los periféricos y drivers con una API en común. Está especialmente diseñado para controladores de la familia de nuestro PixHawk (serie STM32F4). Se compone de varias clases y haremos uso de las siguientes:

Gpio

Pin genérico en un chip cuyo comportamiento se puede controlar por el usuario. Esto es lo principal al conectar el dispositivo.

Permite inicializarlo, configurarlo y, adicionalmente, funciones de lectura y escritura.

- GPIO_InitStructure.

Estructura para la inicialización del GPIO. Sus atributos son los siguientes: pines que se quieren configurar, el modo de operación para estos pines, la velocidad de los pines, el tipo de salida de los pines y la forma de levantar y tirar los pines seleccionados.

- GPIO_Init(CAN_GPIO_PORT, &GPIO_InitStructure).

Inicializa el periférico GPIO según los parámetros establecidos en la estructura de inicialización anterior.

- GPIO_PinAFConfig(GPIO, PinSource, AFSelection).

Función que cambia el mapeo del pin especificado.

- RCC_AHB1PeriphClockCmd(CAN_GPIO_CLK, ENABLE).

Activa o desactiva el reloj periférico. [24] [25]

CAN

Es un protocolo de comunicaciones para la administración de mensajes en entornos distribuidos. Ofrece alta inmunidad ante las interferencias.

Esta es otra clase disponible en la librería. Contiene firmware para realizar conexiones mediante el CAN BUS disponible en nuestro PixHawk.

Para ello, se comienza igual que con Gpio; definiendo su estructura e inicializando:

- CAN_InitStructure.

Estructura para la inicialización del CAN bus. Sus atributos son los siguientes: longitud de un quantum, el modo del CAN, el número máximo de tiempo que el CAN tiene permitido para la resincronización, numero de tiempo en bits, activa o desactiva el modo de comunicación, el manejo automático del bus, el modo de arranque automático, el modo automático de retransmisión y la configuración FIFO.

- CAN_Init(Pix_CAN_Periph, &CAN_InitStructure).

Inicializa el periférico CAN según los parámetros establecidos en la estructura de inicialización anterior.

- CAN_FilterInitStructure

Estructura para el filtro CAN. Se compone de un id, la máscara del filtro, el FIFO que se le asignará al filtro, el filtro a inicializar (del 0 al 13), el modo del filtro, la escala del filtro y si se activa o desactiva el filtro.

- CAN_FilterInit(CAN_FilterInitStructure)

Inicializa la recepción de CAN según el filtro especificado en los parámetros.

- CanTxMsg

Estructura que define los mensajes transmitidos CAN. Sus atributos son: id, identificador del mensaje, marco del mensaje que será retransmitido, longitud del marco y un array con los datos que se transmitirán.

- CanRxMsg

Estructura que define los mensajes recibidos CAN. Sus atributos son: id, identificador del mensaje, marco del mensaje que será retransmitido, longitud del marco, un array con los datos que se transmitirán y un índice del filtro del mensaje almacenado en el buzón.

Usbd_cdc_vcp

Esta clase contiene una función que permite escoger la información que se va a enviar a través del cable USB al ordenador.

- VCP_DataTx(buf, len)

Mediante esta clase, se podrá introducir un buffer con la información que se quiere transmitir por medio del USB hasta el ordenador y la longitud del buffer. Por ejemplo, se puede almacenar en este buffer el estado constante del PixHawk y hacer que lo envíe por medio del USB. Por lo que, en la pantalla del ordenador, la información que se verá procedente del PixHawk será el estado constante del componente. [23] [25] [26]

4.6. ANTECEDENTES DEL PROYECTO Y SOFTWARE ALTERNATIVOS

En este apartado se mostrarán distintas alternativas para el programado del microcontrolador y para el procesado de los datos. Serán una seria de programas que disponen de la misma capacidad para resolver las necesidades del proyecto. Se explicará porque no han sido escogidas, cuáles son las principales características y para qué sirve cada una de ellas. También se hablará sobre los antecedentes tecnológicos en la sociedad en el periodo anterior a este proyecto.

Como ya se conoce en el sector, este apartado en la investigación tecnológica es bastante famoso actualmente. Pocas firmas automovilísticas no invierten en la investigación de estos mercados, por lo que se está desarrollando a un ritmo sorprendente. Varias marcas, las más avanzadas actualmente (Waymo, Volkswagen y Ford), cuentan con software similar al que se va a desarrollar, los cuales piensan introducir en sus modelos más novedosos en lo que es ya “una carrera” por el mejor VA. Se han obtenido ideas de las noticias de estos grandes para la realización de este proyecto.

Ahora, se va a explicar los distintos softwares alternativos disponibles en el mercado y una breve explicación de por qué no han sido escogidos en este caso.

Por un lado, hay diferentes entornos de programación. El ejemplo más famoso para alguien como un ingeniero informático puede ser Eclipse. Es un entorno que ofrece soporte para varios lenguajes de programación. A primera vista, ambos entornos parecen ser idénticos como se muestran en las siguientes figuras:

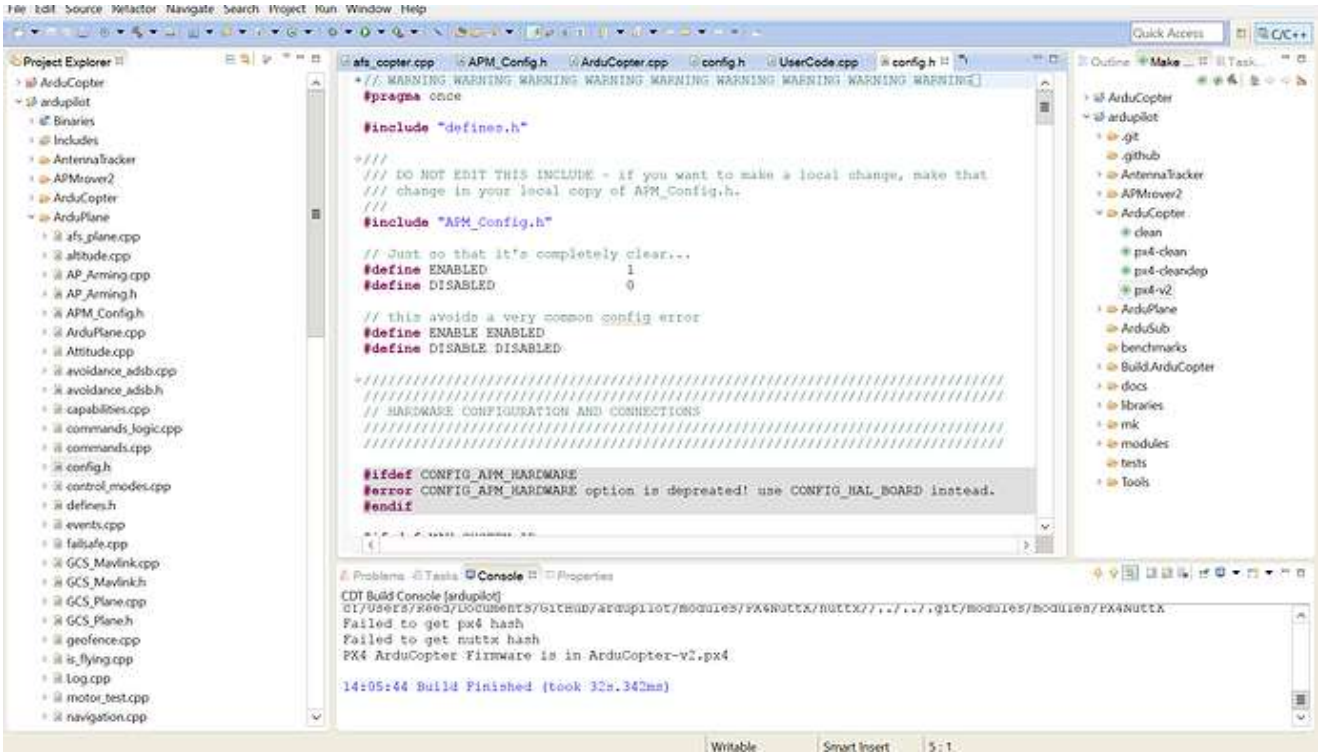

Ilustración 18. Ejemplo Eclipse [15]

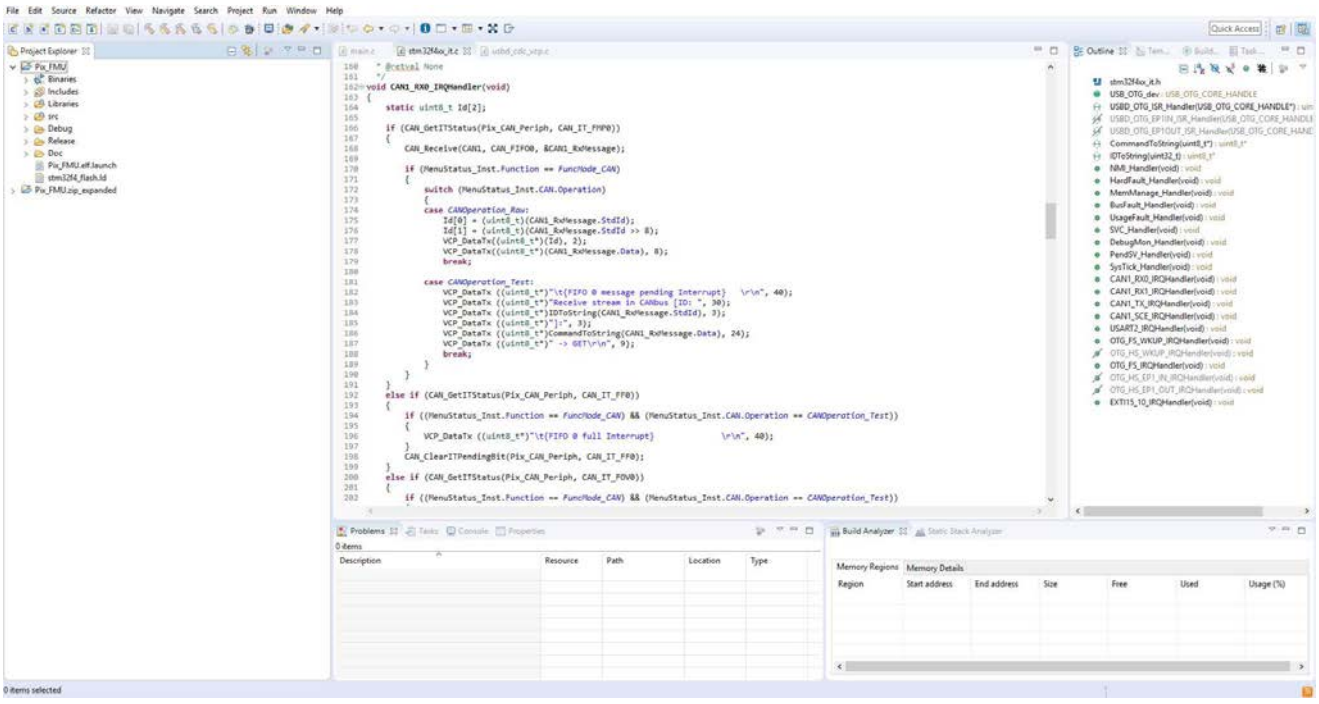

Ilustración 19. Ejemplo Atollic TrueStudio

Lo que diferencia a estos dos programas, son las facilidades de uso para el programador. Eclipse es un entorno que nos permite configurar varios aspectos. Pero en este caso, al trabajar con un microcontrolador, es una tarea más difícil ya que su software no está destinado para estos casos. Sin embargo, Atollic TrueStudio (software basado en eclipse) está destinado para desarrolladores de microcontroladores que necesitan herramientas potentes. Además de incorporar funciones profesionales para el manejo del código y un sistema avanzado de análisis del sistema. Además de un soporte técnico casi inmediato en la herramienta.

En este aspecto, pueden incluirse innumerables programas que realicen la misma función principal que eclipse y Atollic TrueStudio. Muchas más similares a eclipse y otras, como Atollic, más centradas en el trabajo con microcontroladores. Pero se ha decidido decantarse por esta herramienta al tener todo lo que se necesita y por tener una versión de software totalmente gratuito. [16] [17]

Por otro lado, se pueden encontrar varias formas de visualizar los datos provenientes del radar. Hay varias pequeñas herramientas que permiten llevar a cabo estas funciones con distintas características.

Se pueden encontrar herramientas como Termite o como Putty. Termite es un terminal que dispone de una interfaz similar a un chat en el que se puede visualizar todos los datos entrantes y una línea de comandos para introducir lo que se desea transmitir. Simplemente se basa en configurar el puerto del que debe escuchar estipulando el “baudrate” (tasa de baudios) que es el número de unidades de señal por segundo.

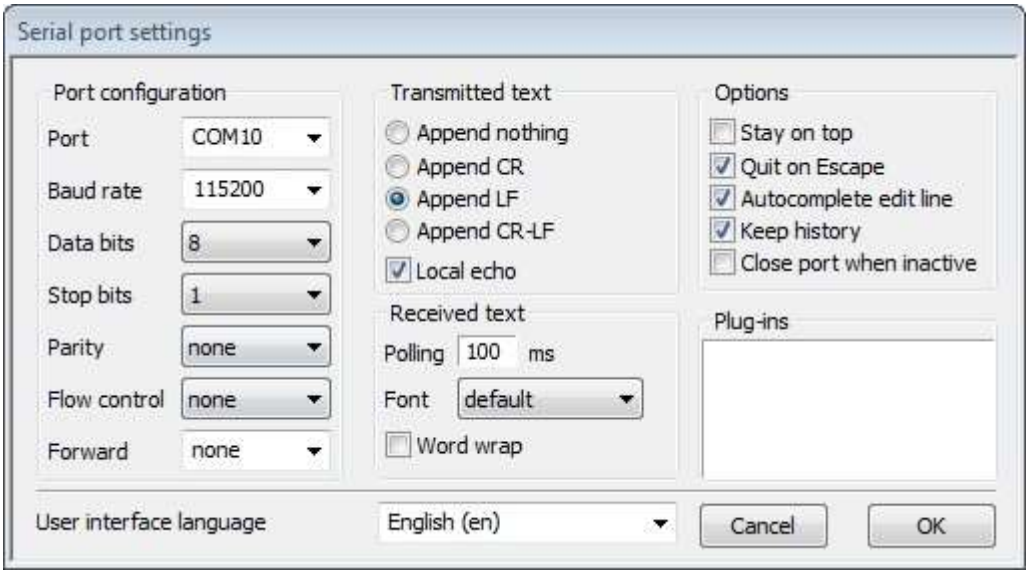

Ilustración 20. Ejemplo Termite [18]

Ilustración 21. Ejemplo Termite 2 [18]

También existe una herramienta muy similar a Termite, esta es Putty. Es una herramienta idéntica a Termite pero incluye más funcionalidades, las cuales no nos sirven de nada en nuestro caso.

Ilustración 23. Herramienta Putty [19]

Por otro lado, se disponen de programas los cuales se pueden incorporar a la terminal en Linux y que permiten visualizar el contenido en la pantalla de la misma terminal, e incluso la posibilidad de guardar estos datos en un fichero. Entre ellos: cu, minicom, screen y tip. Se ha elegido el comando screen ya que es el más sencillo y que incluye todas las funcionalidades que se necesitan (ver la información del puerto sin interrupciones, interactuar con el puerto y guardar toda la información en un fichero).

Por último, para el análisis de los datos y la creación de las gráficas, hay varias herramientas que pueden compararse con Matlab (la elección):

Scilab: programa de cálculo científico. Su sintaxis difiere un poco de la de Matlab. Permite hacer esquemas acústicos, figuras y montajes superficiales. [21] [22]

GNU Octave: es un potente programa orientado a sintaxis matemática y creación de gráficos y herramientas de visualización. Utiliza sintaxis similar a Matlab por lo que es altamente compatible con sus scripts.

Ambas opciones son gratuitas y pueden llevar a cabo las funciones de Matlab de una forma casi igual de eficiente y, puesto que Matlab es de pago, son candidatos muy buenos. Pero gracias a la Universidad Carlos III se dispone de licencia gratis para Matlab por lo que se ha elegido finalmente la opción más potente y eficiente.

Para finalizar, incluir un aporte a una alternativa al hardware, al radar. Una posible alternativa al radar, puede ser un sensor LIDAR o un sensor ultrasónico. Pero se van a exponer las principales características de cada uno para explicar la elección del radar:

1. Ultrasónico vs radar:

Precio: la media de precio para un sensor Ultrasónico es de entre 500 a 2000 euros. Mientras que un radar puede costar alrededor de los 4000 euros, aunque en la actualidad están sobre los 1000 y 2000 euros los más básicos.

Distancia: un sensor ultrasónico, puede llegar a medir hasta a 18 metros de distancia. Mientras que un radar es capaz de alcanzar los 90 metros.

Limitaciones medioambientales: el sensor ultrasónico se puede ver limitado por las condiciones meteorológicas de agua o polvo. Mientras que nada de esto afecta al radar.

Educación: se ha impartido mucha educación sobre los sensores ultrasónicos gracias a su similitud con los murciélagos. Mientras que, del radar, la educación ha sido poca (esto significa que la mayoría de las empresas se centran en el ultrasónico sin pensarse cambiar al radar aunque este sea mejor).

Monitorización: ultrasónico baja y radar alta.

2. LIDAR vs radar:

Coste: un sensor LIDAR cuesta más que un sensor radar por varios motivos. El LIDAR realiza mediciones con receptores ópticos, motores y láseres. También precisan de elementos electrónicos de alta velocidad que cuestan más que los que incluye el radar. Pero nada de esto es esencial para el desarrollo de este proyecto.

Simplicidad: el LIDAR se centra en enviar una gran cantidad de datos por cada punto encontrado con el láser. Mientras que el radar no envía tanta información y solo devuelve información relacionada a la posición del objeto y alguna característica más. Por lo que para este proyecto es ideal.

5. DESARROLLO DEL PROYECTO

En este apartado, se explicará cómo se ha llevado a cabo la realización del proyecto entero. Se expondrá el desarrollo desde la parte principal de programación y configuración hasta la parte final que consiste en realizar las pruebas y analizar los datos obtenidos de las mismas.

Para ello, lo primero es repasar los pasos de los que está estructurado el proyecto:

- Primero, se llevará a cabo la conexión de todos los componentes necesarios para conseguir los datos. Un fallo en este proceso significará un error total por lo que el sistema no funcionará y en el peor de los casos, puede suponer una pérdida o rotura de algún material. Estos componentes son: el radar, el PixHawk, el ST-Link, un ordenador y la fuente de alimentación.

- Después, es el turno de la programación del microcontrolador (PixHawk). Para ello, se utilizará el entorno de Atollic TrueStudio con la librería que se ha mencionado anteriormente la cual provee de todas las funcionalidades que se van a necesitar. Entre ellas, la inicialización de todos los parámetros y configuración de los diferentes pines de la placa que se van a usar. También permitirá configurar el puerto CAN por el cual se envían mensajes para configurar el radar y, a través de este puerto, también se recibirán los datos que deja el PixHawk procedentes del radar. En este punto también se incluirá la función de guardar los datos en un fichero .csv para que su posterior estudio sea más simple al estar dividido en filas y columnas.

- Por último, se estudiarán los datos obtenidos por el radar. Se utilizará la herramienta Excel para la observación primaria de los datos y su reestructuración para su posterior procesado en Matlab. Donde se realizan las distintas gráficas para todos los casos que se han probado y poder ver de una forma más sencilla y visual que ha sucedido en el transcurso de la prueba.

5.1. DISEÑO DE LA SOLUCION FINAL PROPUESTA

En este apartado, se explicará el diseño de la solución final propuesta para el sistema. Se dice final, porque como se ha explicado anteriormente en la planificación del proyecto, este estará dividido en varios sprints en los que se irán entregando un prototipo de una solución al sistema y, a partir de este, y la revisión por parte del responsable y del mismo cliente, se estipularán los cambios oportunos al prototipo de forma que el siguiente solucione los errores o carencias que tenía el anterior.

De esta manera, se ha llegado a una solución final del proyecto que cubre todos los requisitos establecidos inicialmente y que lleva el software realizado en el proyecto que intenta cumplir con las expectativas iniciales.

Cabe mencionar que dicha propuesta final, aún no está al nivel de competir con las grandes firmas del sector ya que no se dispone de los mismos recursos e infraestructuras que ellos. Este proyecto es meramente de investigación y desarrollo por lo que aún le faltarían varios puntos en los que evolucionar y avanzar. Pero todo esto se explicará más adelante en el apartado de Futuras ampliaciones.

Tras esta breve explicación de lo que consta el apartado, se dará comienzo a la explicación de la solución final. Esta explicación irá en las partes que se han mencionado en el apartado anterior: conexión de los componentes, programación y configuración de los componentes y análisis de datos.

5.1.1. Conexión de los componentes

Primero, se hará un resumen de los distintos componentes de los que se dispone y los cables mediante los que se conectan.

1. Por un lado, está el componente principal: el **radar**. El radar dispone de un cable CAN bus por el que transmite toda la información sobre los objetos que detecta. También dispone de dos cables: uno positivo (rojo) y otro negativo (negro) los cuales irán conectados a la fuente de alimentación la cual dotará de energía al radar.

2. **ST-Link**. Este componente es un programador para microcontrolador. Se utilizará para programar el PixHawk. Tiene un puerto microUSB por el que debe conectarse al Ordenador y un puerto de salida para un cable JTAG que irá conectado a un puerto construido para él en el PixHawk.

3. **PixHawk.** Aquí está el microcontrolador. En este caso, se tiene un puerto de salida microUSB que irá conectado con el ordenador y es el cable por donde enviará los datos que recibe del radar. Estos datos irán procesados y llegarán en el formato definido. Pero esto se explicará en el la siguiente parte de la solución final. También tiene un puerto CAN bus por el que podrá conectarse el cable del radar y por el que recibirá toda la información que este capte en tiempo real. Por último, está el puerto JTAG. Este puerto no estaba configurado inicialmente, por lo que ha sido necesario rediseñar uno de los puertos que no iban a ser utilizados para construir un puerto que pueda ser usado con esta finalidad. Para ello, con ayuda de una lima, se amplía un poco más el agujero de serie del PixHawk y se incluyen unos pines más a este puerto para que pueda funcionar con un cable JTAG.

4. **Ordenador**. Este componente es esencial ya que en él se realizarán las distintas funciones necesarias para el desarrollo del proyecto. Enviará esta información

al ST-Link que programará el PixHawk a la vez que recibe los datos del PixHawk para poder verlos en pantalla y guardarlos posteriormente.

5. **Fuente de alimentación**. Este es el último de los componentes. Su función será proveer de energía al radar para que pueda transmitir los datos que detecta en todo momento. Sus valores deben ser constantes durante todo el uso de radar ya que unos valores más bajos pueden provocar el no funcionamiento del sistema y unos valores mayores pueden llegar a estropear o quemar el radar. Estos valores serán de 12 voltios y 0,5 amperios.

Ilustración 24. Esquema de las conexiones del proyecto [30] [31]

Como se puede observar en la imagen que se ha diseñado, los rectángulos indican los cables que conectan los componentes entre sí.

El ordenador irá conectado al programador ST-Link por medio de un cable con un extremo microUSB (ST-Link) y otro extremo USB (ordenador).

La conexión entre el ST-Link y el PixHawk se realizará a través de un cable JTAG.

Por otro lado, el radar envía los datos por medio del cable CAN bus del que dispone que irá conectado al PixHawk también.

El radar, recibirá la energía de la fuente de alimentación por medio de los cables negativos y positivos que se corresponden con los cables rojo y negro tan comunes.

Por último, el PixHawk se conectará con el ordenador por otro cable microUSB. Al igual que pasaba con el ST-Link, el extremo microUSB irá conectado al PixHawk y el extremo USB al ordenador.

También, se puede observar en el esquema que se ha creado, una serie de números que junto con el sentido de las flechas explican cuál es el flujo de la información en el proyecto.

1. Primero, el código que se ha diseñado y todos sus algoritmos se pasan al ST-Link gracias a la herramienta de debugeo de Atollic TrueStudio.

2. Una vez que el dispositivo de programación recibe el código desde el ordenador, es capaz de programar el microcontrolador (PixHawk) por medio del cable JTAG que los une. Ahora el PixHawk tiene que seguir una serie de pautas para enviar la información que recibe del radar al ordenador.

3. Este paso es se realiza una vez que el microcontrolador está programado, lo que significa que es capaz de enviar información al ordenador y por lo tanto el radar puede empezar a enviar datos. Este paso consiste en conectar el radar por medio de los cables positivo y negativo de alimentación para que reciban la energía necesaria. La cual debe ser obligatoriamente de 12 voltios y 0,5 amperios.

4. Una vez que el radar empieza a recibir la energía requerida, empieza a enviar constantemente y en tiempo real datos en crudo al PixHawk sobre todos los obstáculos que es capaz de detectar y los atributos sobre estos (distancia, velocidad, ángulos y desplazamientos entre otros). Toda esta información se enviará por medio del cable CAN bus que estará conectado al PixHawk. En este caso, hay una flecha bidireccional. Esto significa que como se ha explicado, el radar envía información al PixHawk pero, por otro lado, el PixHawk también envía información al radar para configurarlo (solo le llega información al radar por medio del PixHawk). Como resumen, la relación consiste en: el PixHaw configura el radar con el mensaje que recibe del ordenador y, por otro lado, recibe los mensajes del radar y los entrega al ordenador. Todo el procesado y comunicación con el radar se lleva a cabo en el siguiente punto.

5. Por último, el paso final del flujo de información entre los distintos componentes. Una vez que el PixHawk empieza a recibir información desde el radar, hace el procesamiento programado con los datos en crudo y envía al ordenador los datos de la forma que ha sido estipulada anteriormente en los pasos 1 y 2.

Ahora que hemos terminado con el flujo de datos, todo el trabajo de análisis y creación de graficas se llevarán a cabo en el ordenador.

5.1.2. Recogida de los datos en crudo

En este apartado, se comienza con la explicación de los algoritmos definidos para el tratamiento de los datos que transmite el radar. Todo este código, programará el PixHawk gracias al ST-Link de forma que enviará sólo los datos que se desean.

Para empezar, lo primero que se incluirá será un menú en el que por medio de teclas establecidas transmitiremos al PixHawk que se desea empezar a recibir datos.

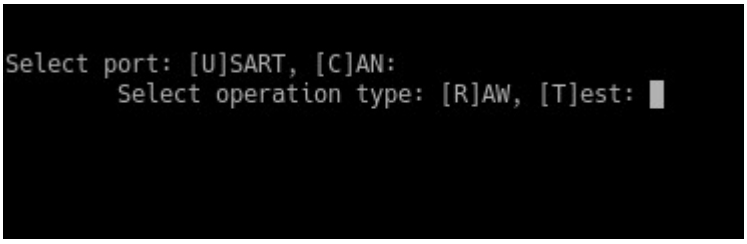

Ilustración 25. Menú de la interfaz programada en el PixHawk

La primera opción que se elige es el la del puerto que vamos a utilizar en el PixHawk, el puerto CAN, a la que se accede pulsando la letra c en el teclado del ordenador.

A continuación, hay 2 opciones:

- Una es la opción de datos en crudo (RAW) la cual se ha utilizado en los primeros sprints para verificar que los datos obtenidos en crudo y casi inentendibles para el ojo humano.

- En esta opción (TEST) se procesan los datos obtenidos en crudo. Tras el paso de los sprints, se ha conseguido obtener solo los campos de los mensajes que interesan: información de la posición, velocidad y ángulo de los objetos.

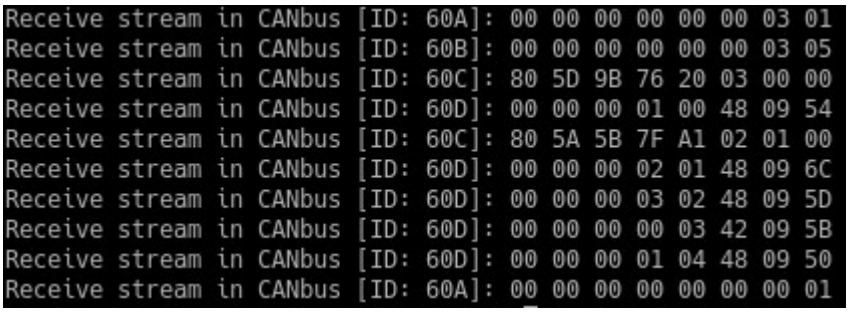

Ilustración 26. Datos parseados. ID y contenido de todo el mensaje

Como se puede observar en la figura, el primer dato que se obtiene en cada transmisión del PixHawk es un dato en nomenclatura hexadecimal. Este se corresponde con el ID del mensaje que transmite el radar para poder conocer el cuerpo de este mensaje y la estructura correspondiente. Ya que cada mensaje utiliza los bytes de manera diferente.

CAN	Frame Format	Message ID[1]	Message Name	Content	Section
1	CAN 2.0A (11 Bit)	0x200	RadarConfiguration	Sensor configuration	6
1	CAN 2.0A (11 Bit)	0x60A	RadarStatus	Sensor Status output	7
1	CAN 2.0A (11 Bit)	0x70B	CAN1_ClusterStatus	Cluster status	8.1
1	CAN 2.0A (11 Bit)	0x70C	CAN1_Cluster_1	Cluster information 1	8.1
1	CAN 2.0A (11 Bit)	0x60B	CAN1_TrackStatus	Track status	8.2
1	CAN 2.0A (11 Bit)	0x60C	CAN1_Track_1	Track information 1	8.2
1	CAN 2.0A (11 Bit)	0x60D	CAN1_Track_2	Track information 2	8.2

Ilustración 27. Mensajes disponibles en el radar. [SRR 208 Software Vers.:1.0.3.0].

Esta figura, proviene del manual del radar (SRR 208 Continental Technical Documentation). Se va a explicar brevemente el contenido de cada mensaje:

- **0x200**: este mensaje se envía desde el ordenador al radar por medio del PixHawk. Mediante este mensaje, es posible configurar el radar. De forma que se puede escoger entre el modo de detección de objetos. Este modo puede ser “SendTracks” o “SendClusters”. También se puede configurar el ID del sensor. El digito intermedio en todos los IDs de los mensajes es el 0. Eso es porque por defecto, el radar viene configurado con este ID del sensor. Dispone de un rango de 0-7 IDs posibles, por lo que en caso de configurar el ID del sensor a 1, el mensaje RadarStatus (por ejemplo) llegaría con el ID de mensaje 0x61A.

- **0x60A**: este mensaje indica el estado constante del radar. Ofrece información de estado interno del sensor (funciona bien, hay algún fallo en el hardware o si la temperatura es demasiado alta), el estado de la configuración del radar (la configuración que está usando o si el nuevo ID del sensor se está usando).

- **0x70B, 0x70C**: Estos dos mensajes, se corresponden con el envío de datos cuando está configurado en el tipo “SendCluster”.
  El primer mensaje 0x70B, contiene información general sobre la lista disponible en el cluster en este momento. Por ejemplo, si el radar detecta 8 objetos distintos,

enviaría un 8 (todo esto en bits) para informar de que la lista de cluster contiene actualmente 8 clusters.
Por otro lado, el mensaje 0x70C contiene toda la información relativa al cluster. Incluye: un índice que indica el cluster al que corresponde esta información dentro de la lista de clusters comentada anteriormente, un valor RCS del cluster en cuestión, la distancia radial, el ángulo y la velocidad relativa del mismo.

- **0x60B, 0x60C, 0x60D**: Estos tres mensajes, se corresponden con el envío de datos cuando el radar está configurado con "SendTracks".
  El primero (0x60B), al igual que en el caso anterior, contiene el número de trazas que se han medido, las cuales se identifican mediante un id. Aquí se almacenará toda la lista de todos los tracks.
  En el segundo mensaje 0x60C, contiene la información de estos tracks: id del track (dentro de la lista de tracks mencionada anteriormente), desplazamiento longitudinal, desplazamiento lateral, velocidad relativa lateral y velocidad relativa longitudinal.
  Por último, el mensaje 0x60D, contiene más información relativa de los tracks en la lista. Incluye: el id del track en cuestión, su valor RCS y un valor con el tiempo que lleva este track en la lista.

Todos estos mensajes, tienen en común que incluyen un mismo campo: rollcount. Este campo es un pequeño contador (0-3) para asegurar que los mensajes llegan en estricto orden y sin pérdidas. Otra cosa en común de todos estos mensajes es que tendrán un tamaño total de 8 bytes con sus correspondientes 64 bits. Cada mensaje los usará según necesite pudiendo estar algunos bits sin utilizar en algunos casos.

5.1.3. Recogida de los datos procesados

Para el caso de este proyecto y, tras el transcurso de todas las iteraciones, se configurará el PixHawk para que solo envíe datos de los mensajes 0x60C y 0x70C los cuales corresponden a la información que se necesita de los objetos que detecta el radar en ambos modos.

Tanto los "clusters" como los "tracks" son transmitidos en CAN1. Los cluster son determinados en cada ciclo por los objetos detectados por el radar. Mientras que los tracks se evalúan mediante el seguimiento de los clusters durante el tiempo, de forma que incluyen información de varios ciclos.

5.1.3.1. Mensaje CAN1_Track1 (0x60C)

El primer mensaje que se va a recibir y con el que se va a trabajar, va a ser el mensaje 0x60C, correspondiente a la información de los tracks enviados por el radar sobre los objetos que ha encontrado en su trayectoria.

```
60C,2,128,1750,490,36
60C,0,128,1750,511,20
60C,1,128,1750,520,40
60C,2,128,1750,490,36
60C,0,128,1750,511,20
60C,1,128,1750,520,40
60C,2,128,1750,490,36
60C,0,128,1750,511,20
60C,1,128,1750,520,40
60C,2,128,1750,490,36
60C,0,128,1750,511,20
60C,1,128,1750,520,40
60C,2,128,1750,490,36
60C,0,128,1750,511,20
60C,1,128,1750,520,40
60C,2,128,1750,490,36
60C,0,128,1750,511,20
60C,1,128,1750,520,40
60C,2,128,1750,490,36
60C,0,128,1750,511,20
60C,1,128,1750,520,40
60C,2,128,1750,490,36
60C,0,128,1750,511,20
60C,1,128,1750,520,40
60C,2,128,1750,490,36
60C,0,128,1750,511,20
60C,1,128,1750,520,40
60C,2,128,1750,490,36
60C,0,128,1750,511,20
```

Ilustración 28. Datos del mensaje 0x60C

Como se puede observar en la captura, está programado que los datos de este mensaje sean enviados de la siguiente manera. Cada línea contiene la misma información, pero con diferentes valores según la ocasión que se esté dando en ese momento.

Primero, se obtiene el id del mensaje para asegurar que se está recibiendo la información que se ha requerido en todo momento.

A continuación, se recibe el id del track del que se envía dicha información. Como se puede observar en la imagen, en ese momento estaba detectando un total de 3 objetos (IDs 0, 1, 2).

En tercer y cuarto lugar, se encuentra la información referente a las velocidades lateral y longitudinal respectivamente.

Y, por último, en quinto y sexto lugar se dispone la información de los desplazamientos lateral y longitudinal respectivamente.

Todos estos campos irán separados mediante una coma ya que la finalidad es guardar estos datos en un fichero .csv (hoja de cálculo) y poder usar dicho fichero para almacenar la información conjuntamente en columnas y analizarla de una forma mucho más fácil y eficiente.

Puede parecer algo raro ver los datos de todos estos campos con unos valores tan raros y faltos de significado. Pero el programador de microprocesadores que utilizamos no soporta funciones en coma flotante (lo que conocemos como variables de tipo float, para poder trabajar con decimales), por lo que el cálculo exacto se realizará más adelante en las hojas de cálculo en las que se guardan los datos extraídos del radar.

Para la realización de dichos cálculos, se han seguido las instrucciones del fabricante:

Ilustración 29. Estructura del mensaje 0x60C [SRR 208 Software Vers.:1.0.3.0]

Signal	Start	Len	Byte Order	Value Type	Res	Offset	Value Range
Track_ID	8	16	Motorola	Unsigned	1	0	0 -> 65535
Track_LongDispl	29	9	Motorola	Unsigned	0.1 m	0	0 -> 51.1
Track_LatDispl	46	10	Motorola	Unsigned	0.1 m	-51.1	-51.1 -> 51.2
Track_Index	24	5	Motorola	Unsigned	1	0	0 -> + 24
Track_VrelLong	50	12	Motorola	Unsigned	0.02 m/s	-35	-35 -> + 35
Track_VrelLat	56	8	Motorola	Unsigned	0.25 m/s	-32	-32 -> 31.75
Track1_RollCount	48	2	Motorola	Unsigned	1	0	0 -> 3 tpcl.: 0->3

Ilustración 30. Informacion para el mensaje 0x60C [SRR 208 Software Vers.:1.0.3.0]

En la primera imagen, está explicada la estructura del mensaje. La matriz es un símil a los 8 bytes de los que está compuesto y los 8 bits que, a su vez, componen cada byte. Las filas corresponden a los bytes y las columnas a los bits. Para coger los datos que necesitamos de estás direcciones específicas, se ha realizado un proceso de conversiones de los datos hasta que se ha llegado al resultado deseado.

```
while (count < 24) {//pasamos el mensaje en hex a un array
    data[count] = *(((uint8_t*)CommandToString(CAN1_RxMessage.Data))+count);
    auxdata[count] = *(((uint8_t*)CommandToString(CAN1_RxMessage.Data))+count);
    count++;
}
```

Ilustración 31. Inserción de los datos en hexadecimal en arrays.

Con la función CommandToString(), se coge la información del mensaje (quitando el id) y se transforma en un String que será guardado en 2 arrays. Uno que se mostrará por pantalla y otro que será usado para seguir con la conversión de estos datos.

El siguiente paso, es convertir estos datos en hexadecimal en formato binario ya que, para seguir el manual, hay que acceder a las direcciones en unidades de bits.

```
while(count <24){//convertimos el mensaje de hex a binario y lo introducimos en una matriz, para que tenga
                 //el mismo aspecto que la matriz del manual

    //guardamos los datos en binario.
    if(count == 0 || count%3==0 || strcmp(auxdata[count], " ") == 0){
        bina[countbina]=auxdata[count];
    }else{

        switch (auxdata[count]) {
        case '0':
            bina[countbina]='0';
            auxbina[countbina]='0';
            countbina++;
            bina[countbina]='0';
            auxbina[countbina]='0';
            countbina++;
            bina[countbina]='0';
            auxbina[countbina]='0';
            countbina++;
            bina[countbina]='0';
            auxbina[countbina]='0';
            break;
        case '1':
            bina[countbina]='0';
            auxbina[countbina]='0';
            countbina++;
            bina[countbina]='0';
            auxbina[countbina]='0';
            countbina++;
            bina[countbina]='0';
            auxbina[countbina]='0';
            countbina++;
            bina[countbina]='1';
            auxbina[countbina]='1';
            break;
        .
        .
        .
        .
        .
        case 'F':
            bina[countbina]='1';
            auxbina[countbina]='1';
            countbina++;
            bina[countbina]='1';
            auxbina[countbina]='1';
            countbina++;
            bina[countbina]='1';
            auxbina[countbina]='1';
            countbina++;
            bina[countbina]='1';
            auxbina[countbina]='1';
            break;

            default:
                break;
        }
    }
```

Ilustración 32. Conversión de datos hexadecimales en datos binarios.

Esta conversión es muy sencilla. Dependiendo del carácter que lea, lo convierte en su correspondiente valor binario. De esta forma, se dispone de un array con toda la información del mensaje en binario.

Después, se introducen los datos del array en una matriz 8x8 que se asemeje al esquema de la Ilustración 29 (ilustración del esquema 0x60C).

Ahora que existe una matriz con todos los datos con una estructura idéntica a la especificada en el manual del radar, se puede proceder a recoger la información que se necesita. Por ejemplo, en el caso del mensaje 0x60C, se necesitan los datos relacionados las velocidades y los desplazamientos laterales y longitudinales.

Para el caso de la velocidad relativa lateral. Esta información ocupa el byte 8 del mensaje. En la matriz, correspondería con los campos fila 7, columna 7 hasta fila 7, columna 0. Cabe destacar que en este caso, se ha cambiado la numeración de las filas y las columnas. Cuando en la imagen aparece la fila 0 la primera fila, en nuestro caso, la fila 0 se corresponde con la última fila.

	0	1	2	3	4	5	6	7
7								
6								
5								
4								
3								
2								
1								
0								

Ilustración 33. Esquema para guardar los bits especificado en el código del proyecto

```
    //cogemos las posiciones que deben corresponder al VrelLat.
uint8_t VrelLat [8];//la velocidad relativa lateral ocupa 8 bits

mi=0;
mj=0;

for(int i = 0; i<8; i++){
    VrelLat[i] = matrix[mi][mj];

    if(mj==7){
        mi++;
        mj = 0;
    }else{
        mj++;
    }

}
```

Ilustración 34. Extracción de posiciones para el valor de la Velocidad Relativa Lateral

De esta forma, se empieza a leer en la posición 0,0 que es donde comienzan los bits para este dato y se pone un contador de 8 (la longitud en bits de este dato). Por lo que leerá bits hasta la posición 0,7 y lo guardará en el vector que tiene programado.

Otro ejemplo, en este caso de un campo que tenga una longitud de más de un byte, sería el caso de la velocidad relativa longitudinal. Según el esquema de la Ilustración 33, esta información comienza en las posiciones 2,2 y termina en las posiciones 1,5.

```
      //cogemos las posiciones que deben corresponder al VrelLong.
uint8_t VrelLong [12];//la velocidad relativa longitudinal ocupa 12 bits

mi=2;
mj=2;

for(int i = 0; i<12; i++){
    VrelLong[i] = matrix[mi][mj];

    if(mj==7){
        mi--;
        mj = 0;
    }else{
        mj++;
    }

}
```

Ilustración 35. Extracción de posiciones para el valor de la Velocidad Relativa Longitudinal

Como se puede observar, en este caso el bucle se ejecutará un total de 12 veces puesto que este campo es más grande que el anterior.

En estos momentos, se dispone de una serie de arrays con los datos que se necesitan de este mensaje. El siguiente paso, será realizar una conversión de estos datos de binario a decimal. Para ello, se ha realizado una función que hace esta conversión fácilmente. Devolviendo el número decimal del resultado.

```
int conv(uint8_t array [], int binarios [], int len){
    int j = 11;
    int suma=0;

    for (int i = len; i > 0; i--) {
        if(array[i-1] == '1'){
            suma = suma + binarios[j];
        }
        j--;
    }
    return suma;
}
```

Ilustración 36. Función para convertir array de bits en un valor decimal

El funcionamiento de esta función es bastante sencillo. Necesita 3 parámetros: el array con los bits, un array con los valores binarios equivalentes a decimal (00 = 0, 01 = 1, 10=2, etc.) y un valor de la longitud del array de bits.

El array de los valores binarios equivalentes a decimal sería el siguiente:

```
int binarios [12];
    binarios [0]=2048;
    binarios [1]=1024;
    binarios [2]=512;
    binarios [3]=256;
    binarios [4]=128;
    binarios [5]=64;
    binarios [6]=32;
    binarios [7]=16;
    binarios [8]=8;
    binarios [9]=4;
    binarios [10]=2;
    binarios [11]=1;
```

Ilustración 37. Array de valores binario-decimal

Con este array, dependiendo de la posición donde se encuentre un 1, habrá que sumar la cantidad decimal establecida.

Por lo que, continuando con la explicación de la función de la Ilustración 37 (código de conv), se recorre el array de bits empezando por la última posición del array ya que se corresponde con el bit menos significativo. Cada vez que se encuentre un 1, se sumará la cantidad asociada a su posición, obteniendo así el valor decimal del array seleccionado.

Por último, ya se disponen de todos los datos en sus variables correspondientes por lo que lo único restante es hacer que se envíen al ordenador.

```
VCP_DataTx ((uint8_t*)idd, 3);
VCP_DataTx ((uint8_t*)",", 1);
VCP_DataTx ((uint8_t*)fTIndex, s);
VCP_DataTx ((uint8_t*)",", 1);
VCP_DataTx ((uint8_t*)fVrelLat, q);
VCP_DataTx ((uint8_t*)",", 1);
VCP_DataTx ((uint8_t*)fVrelLong, p);
VCP_DataTx ((uint8_t*)",", 1);
VCP_DataTx ((uint8_t*)fLatDispl, n);
VCP_DataTx ((uint8_t*)",", 1);
VCP_DataTx ((uint8_t*)fLongDispl, m);
```

Ilustración 38. Valores a mostrar en el ordenador para el mensaje 0x60C.

La función VCP_DataTx (), es una función que recibe como parámetro un buffer con los datos que se quiere que el PixHawk envíe a través del USB al ordenador y un valor numérico con la longitud de los datos que debe enviar. Por ejemplo, para enviar un 1 la sintaxis sería:

```
uint8_t ejemplo = 1;
VCP_DataTx ((uint8_t*)ejemplo, 1);
```

Ilustración 39. Ejemplo de la función VCP_DataTx ().

De esta forma, y tras todo este trabajo, se reciben los datos de la forma indicada en la Ilustración 28 (Datos del mensaje 0x60C).

5.1.3.2. Mensaje CAN1_Cluster (0x70C)

El otro mensaje que se desea obtener es el relacionado con la transmisión de mensajes con datos de clusters detectados. Esta información llega a través del mensaje con ID = 0x70C.

```
70C,3,700,45,31
70C,4,701,86,53
70C,0,700,45,10
70C,1,700,51,21
70C,2,700,30,21
70C,3,700,45,31
70C,0,700,45,10
70C,1,700,51,21
70C,2,700,30,21
70C,3,700,45,31
70C,0,700,45,10
70C,1,700,51,21
70C,2,700,30,21
70C,3,700,45,31
70C,0,700,45,10
70C,1,700,51,21
70C,2,700,30,21
70C,3,700,46,32
70C,0,700,45,10
70C,1,700,52,21
70C,2,700,30,21
70C,3,700,46,31
70C,0,700,45,10
70C,1,700,51,20
70C,2,700,30,21
70C,3,700,45,28
70C,0,700,45,10
70C,1,700,51,21
70C,2,700,30,21
```

Ilustración 40. Datos del mensaje 0x70C

Como se puede observar, la forma de envío de la información es muy similar a la recibida en el mensaje 0x60C. En ambos, está determinado que los distintos campos

vayan separados mediante una coma. De forma que puedan guardarse fácilmente en un fichero .csv y su análisis sea mucho más sencillo al estar distribuido en tablas.

En cuanto a los datos que se reciben:

Primero se obtiene el ID del mensaje para asegurar que toda la información recibida es del tipo que se ha solicitado y que va a ser tratada correctamente.

A continuación, se recibe el identificador del cluster al que hace alusión toda la información incluida en el mensaje.

En tercer lugar, se obtiene la velocidad del cluster en cuestión.

En penúltimo lugar se tiene el valor del ángulo al que está el cluster respecto al radar.

Y, por último, la distancia del cluster al radar.

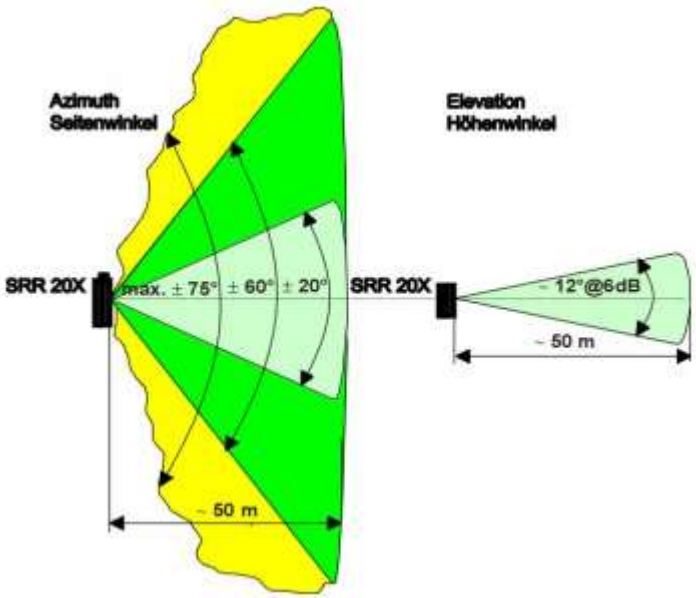

Ilustración 41. Información de las medidas de ángulo y distancia máximas del radar [SRR 20X /-2 /-2C /-21 datasheet]

La figura muestra que la medida del ángulo máxima puede llegar a ser de 75º hacia cada lado del radar. Y que la distancia de medida máxima es de alrededor de 50 metros.

Para la obtención de estos datos, igual que en el caso anterior, lo primero que se debe hacer es acudir al manual del radar para conocer la estructura del mensaje y los datos que envía.

Ilustración 42. Estructura del mensaje 0x70C [SRR 208 Software Vers.:1.0.3.0]

Signal	Start	Len	Byte Order	Value Type	Res	Offset	Value Range
Cluster_Index	0	8	Motorola	Unsigned	1	0	0 -> 127
Cluster1_RollCount	38	2	Motorola	Unsigned	1		0 -> 3 tpcl.: 0->3
Cluster_RCSValue	8	8	Motorola	Unsigned	0.5 dBm^2	-50	-50 -> 30
Cluster_Range	16	8	Motorola	Unsigned	0.2 m	0	0 -> + 51
Cluster_Azimuth	24	7	Motorola	Unsigned	2 deg	-90	-90 -> + 90
Cluster_Vrel	40	11	Motorola	Unsigned	0.05 m/s	-35	-35 -> +35

Ilustración 43. Información para el mensaje 0x70C [SRR 208 Software Vers.:1.0.3.0]

Como explica la estructura, los datos necesarios se encuentran comprendidos en los bits marcados para cada tipo.

Como se ha comentado anteriormente, el proceso hasta la obtención de la matriz con todos los bits incluidos en el mensaje será el mismo para los mensajes 0x60C y 0x70C.

Por lo que se comenzará a explicar este apartado a partir de ese punto. Datos necesarios:

1. Cluster índex: este dato se corresponde con el cluster actual al que referencian el resto de los datos. Es importante disponer de este dato ya que podría confundirse medidas para un objeto que se encuentre en otra ubicación.

   Para la obtención de este valor, se acudirá a las posiciones de memoria 7,0 e iremos recogiendo bits hasta la posición 7,7. Recordemos la asignación de la matriz que se ha establecido en el algoritmo: Ilustración 33.

```
    //cogemos las posiciones que deben corresponder al indice que indica el cluster de la lista actual.
uint8_t CIndex [8];//la index ocupa 8 bits

mi=7;
mj=0;

for(int i = 0; i<8; i++){
    CIndex[i] = matrix[mi][mj];

    if(mj==7){
        mi--;
        mj = 0;
    }else{
        mj++;
    }

}
```

Ilustración 44. Extracción de posiciones para el valor del Cluster Index

2. Velocidad del cluster: este dato como su nombre indica, contiene el valor de la velocidad del cluster que está siendo detectado por el radar.

   Este dato se obtiene acudiendo a la posición 3,5 de la matriz y leyendo los 11 bits siguientes. Esto detallado en el código sería tal que así:

```
    //cogemos las posiciones que deben corresponder a la velocidad del cluster.
uint8_t CVel [11];//la velocidad relativa del cluster ocupa 11 bits

mi=3;
mj=5;

for(int i = 0; i<11; i++){
    CVel[i] = matrix[mi][mj];

    if(mj==7){
        mi--;
        mj = 0;
    }else{
        mj++;
    }

}
```

Ilustración 45. Extracción de posiciones para la velocidad relativa del cluster

3. Ángulo del cluster: con este valor, se puede conocer en qué posición lateral se encuentra teniendo al radar como eje 0. Se puede obtener este valor acudiendo a las posiciones 4,1 de la matriz y leyendo los 7 bits siguientes.

```
    //cogemos las posiciones que deben corresponder al angulo del cluster.
uint8_t CAngle [7];//el angulo ocupa 7 bits

mi=4;
mj=1;

for(int i = 0; i<7; i++){
    CAngle[i] = matrix[mi][mj];

    if(mj==7){
        mi--;
        mj = 0;
    }else{
        mj++;
    }

}
```

Ilustración 46. Extracción de posiciones para el valor del ángulo del cluster

4. Y, por último, la distancia del cluster al radar: este dato ocupa un total de 8 bits. Para recoger su valor, hay que acceder a las posiciones 5,0 de la matriz y leer los 8 bits siguientes.

```
    //cogemos las posiciones que deben corresponder al Range (distancia).
uint8_t CRange [8];//la distancia del cluster ocupa 8 bits

mi=5;
mj=0;

for(int i = 0; i<8; i++){
    CRange[i] = matrix[mi][mj];

    if(mj==7){
        mi++;
        mj = 0;
    }else{
        mj++;
    }

}
```

Ilustración 47. Extracción de posiciones para el valor de la distancia del cluster al radar.

Ahora que ya están disponibles y en sus respectivas variables todos los datos necesarios, se hará uso de la función VCP_DataTx () para que sean enviados mediante el USB desde el PixHawk hasta el ordenador.

```
VCP_DataTx ((uint8_t*)idd, 3);
VCP_DataTx ((uint8_t*)",", 1);
VCP_DataTx ((uint8_t*)fCIndex, b);
VCP_DataTx ((uint8_t*)",", 1);
VCP_DataTx ((uint8_t*)fCVel, d);
VCP_DataTx ((uint8_t*)",", 1);
VCP_DataTx ((uint8_t*)fCAngle, c);
VCP_DataTx ((uint8_t*)",", 1);
VCP_DataTx ((uint8_t*)fCRange, a);
```

Ilustración 48. Valores a mostrar en el ordenador para el mensaje 0x70C

De esta forma, recibiremos los datos como hemos indicado al principio de este apartado en la Ilustración 40.

5.1.3.3. Configurar radar para cambiar entre Track y Cluster (0x200)

Como se ha comentado anteriormente, el radar también ofrece la posibilidad de trabajar enviando los clusters detectados mediante el tipo de envío “SendCluster”.

Para ello, se acude a la información del manual para saber qué campos hay que rellenar para informar al radar de que se quiere modificar su configuración.

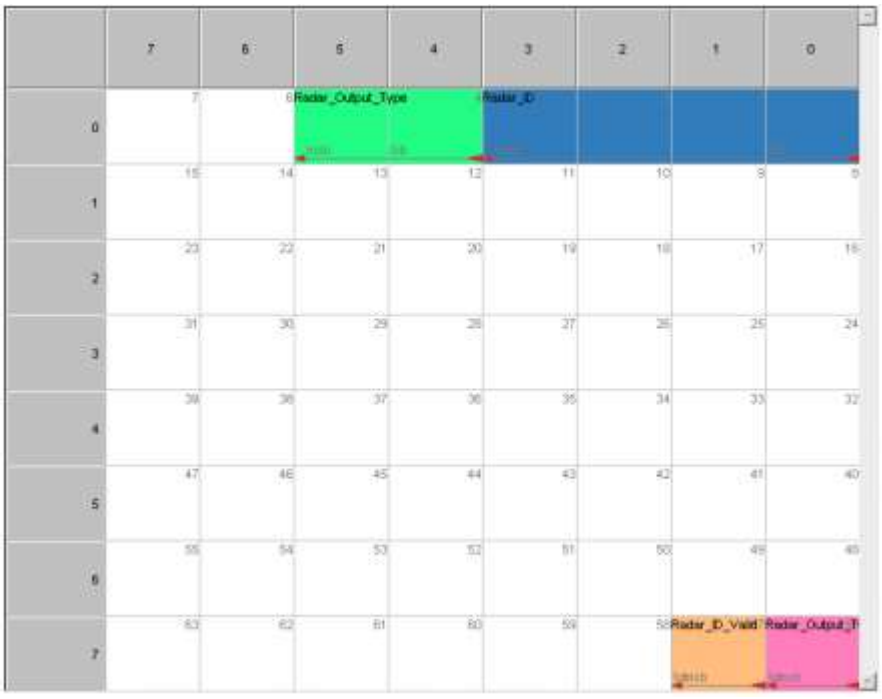

Ilustración 49. Estructura del mensaje de configuración (0x200) [SRR 208 Software Vers.:1.0.3.0]

Signal	Start	Len	Byte Order	Value Type	Res	Value Range
Radar_Output_Type_Valid	56	1	Motorola	Unsigned	1	▪ 0 -> Invalid ▪ 1 -> Valid
Radar_Output_Type	4	2	Motorola	Unsigned	1	▪ 0 -> (0 x 0) SendTracks ▪ 1 -> (0 x 1) SendCluster
Radar_ID_Valid	57	1	Motorola	Unsigned	1	▪ 0 -> False ▪ 1 -> True
Radar_ID	0	4	Motorola	Unsigned	1	0 ... 7

Ilustración 50. Mensaje de configuración del radar. [SRR 208 Software Vers.:1.0.3.0]

Según el manual, para cambiar a modo “SendCluster”, se debe colocar un ‘01’ en las posiciones de los bits 5 y 4 del byte menos significativo. Por otro lado, el resto de las variables, irán a 1 a excepción del radar_ID que se mantendrá a 0. Para ello, se utilizará el siguiente fragmento de código:

```
case FuncMode_CAN:
    switch (MenuStatus_Inst.CAN.Operation)
    {
    case CANOperation_Test:
        switch (c)
        {
        case 'h':
            VCP_DataTx ((uint8_t*)"help ->\r\n", 9);
            break;
        case '0':
            CAN_SendCommand(StringToCommand((uint8_t*)"03 00 00 00 00 00 00 10"));
            break;
        case '1':
            CAN_SendCommand(StringToCommand((uint8_t*)"03 00 00 00 00 00 00 00"));
            break;
```

Ilustración 51. Código para configurar el radar

En este caso, la función CAN_SendCommand (), recibe un comando en hexadecimal y lo transmite al radar por medio del CAN bus del PixHawk mediante el mensaje de configuración 0x200.

Como se puede observar, al tener siempre el byte más significativo como ‘03’ se consigue poner los campos Radar_Output_Type_Valid y Radar_ID_Valid a 1 y el Radar_ID a 0. Por otro lado, con la opción 0, cambiaría el radar a la configuración de “SendCluster” y con la opción 1 a “SendTracks”. Ya que con estas opciones se escriben un 1 y un 0 respectivamente en el campo Radar_Output_Type.

```
60C,0,128,1748,511,19
60C,1,128,1757,520,43
60C,2,128,1752,489,37
60C,3,128,1738,513,61
Send command 0 to CANbus [ID: 200]: 03 00 00 00 00 00 00 10
        {Transmit mailbox empty Interrupt}
60C,0,128,1748,511,19
60C,1,128,1757,520,43
60C,2,128,1752,489,37
```

Ilustración 52. Ejemplo cambio de SendTracks a SendClusters

```
70C,0,700,45,10
70C,1,700,51,20
70C,2,700,30,21
Send command 0 to CANbus [ID: 200]: 03 00 00 00 00 00 00 00
        {Transmit mailbox empty Interrupt}
60C,0,128,1748,511,19
60C,1,128,1757,520,43
```

Ilustración 53. Ejemplo cambio de SendClusters a SendTracks

De esta forma, se podrá cambiar entre los dos modos y recoger los datos que se necesiten en cada momento y situación.

Por último, la única medida que faltaría por tomar, es el guardado de todos estos datos en un fichero .csv. Para ello, el comando screen que se utiliza, incorpora una opción para guardar todo lo que envía el PixHawk; en el fichero que le indiquemos. Para ello, se ejecutará el siguiente comando:

```
[luis@luis-GE60-2PE ~ ]$ screen -L -Logfile caminando-hacia-radar--cluster.csv /dev/ttyACM0 115200
```

Ilustración 54. Ejemplo comando screen

5.1.3.4. Cálculo y creación de gráficas con los datos recogidos

En este apartado, se tratarán los cálculos realizados para obtener los valores exactos para cada tipo de dato (velocidad, distancia, etc.). Ya que, como se comentó anteriormente, la herramienta de programación del microprocesador no tiene soporte sobre variables float. Por lo que se harán estos cálculos sobre la hoja de cálculo.

Para el caso del mensaje de tracks (0x60C), se obtienen los siguientes datos: Velocidad relativa lateral, velocidad relativa longitudinal, desplazamiento lateral y desplazamiento longitudinal.

```
60C,0,128,1750,511,20
```

Ilustración 55. Ejemplo de datos recibidos en el mensaje 0x60C

Según el manual del radar, para obtener el valor exacto de estas variables hay que realizar las siguientes funciones matemáticas:

X = valor obtenido para velocidad lateral desde el PixHawk.

Y = valor obtenido para velocidad longitudinal desde el PixHawk.

Z = valor obtenido para desplazamiento lateral desde el PixHawk.

W = valor obtenido para desplazamiento longitudinal desde el PixHawk.

1. $\mathrm{VrelLat} = \left(\mathrm{X\ x}\ \frac{0{,}25\mathrm{m}}{\mathrm{s}}\right) - \frac{32\mathrm{m}}{\mathrm{s}}$

Ecuación 1. Velocidad relativa lateral

Como se puede observar, la medida utilizada para la velocidad es de metros/segundo. Si el valor resulta negativo, significa que el objeto se está moviendo hacia la izquierda. Y, por el contrario, si el resultado es positivo, significa que se mueve hacia la derecha (siempre con el radar como observador).

2. $\mathrm{VrelLong} = \left(\mathrm{Y\ x}\ \frac{0{,}02\mathrm{m}}{\mathrm{s}}\right) - \frac{35\mathrm{m}}{\mathrm{s}}$

Ecuación 2. Velocidad relativa longitudinal

En este caso, como también se trata de un valor correspondiente a una velocidad, la unidad de medida será el metro/segundo. En este caso, si el valor es negativo, significa que el track en cuestión se está acercando al radar. Mientras que si este valor es positivo, significa que se está alejando.

3. $\mathrm{LatDispl} = (\mathrm{Z\ x}\ 0{,}1\mathrm{m}) - 51{,}1\mathrm{m}$

Ecuación 3. Desplazamiento lateral

Para el caso del desplazamiento lateral, se usará esta fórmula. Esto indicará la posición lateral en la que se encuentra el track. En caso de ser un resultado negativo, significa que está a la izquierda del radar. Y, si el dato es positivo, este se encontrará a la derecha. La unidad de medida utilizada en este caso será el metro.

4. $\mathrm{LongDispl} = (\mathrm{W\ x}\ 0{,}1\mathrm{m})$

Ecuación 4. Desplazamiento longitudinal

Por último, la fórmula necesaria para el valor del desplazamiento longitudinal. Este valor en ningún caso puede ser negativo por lo que simplemente se realiza la multiplicación. También se usará el metro como unidad de medida.

Por ejemplo, para el caso de la Ilustración 55. Ejemplo de datos recibidos en el mensaje 0x60C, obtendríamos el siguiente resultado:

a. $\mathrm{VrelLat} = \left(128 \text{ x } \frac{0{,}25\mathrm{m}}{\mathrm{s}}\right) - \frac{32\mathrm{m}}{\mathrm{s}} = 0\mathrm{m/s}$

Ecuación 5. Ejemplo velocidad relativa lateral

Esto significa que el objeto no se mueve lateralmente. Ahora se verá si se mueve longitudinalmente con la siguiente operación.

b. $\mathrm{VrelLong} = \left(1750 \text{ x } \frac{0{,}02\mathrm{m}}{\mathrm{s}}\right) - \frac{35\mathrm{m}}{\mathrm{s}} = 0\mathrm{m/s}$

Ecuación 6. Ejemplo velocidad relativa longitudinal

Tampoco se mueve longitudinalmente por lo que el track en cuestión es un objeto que está totalmente quieto. El siguiente paso es detectar donde está situado.

c. $\mathrm{LatDispl} = (511 \text{ x } 0{,}1\mathrm{m}) - 51{,}1\mathrm{m} = 0\mathrm{m}$

Ecuación 7. Ejemplo desplazamiento lateral

Este resultado indica que el desplazamiento lateral del track es de 0. Como se ha explicado anteriormente, si el resultado fuese negativo, significa que el objeto está situado en el lado izquierdo del radar y, si fuese un resultado positivo, estaría situado en el lado derecho. Ya que el resultado obtenido ha sido de 0, el objeto está justo en frente del radar.

d. $\mathrm{LongDispl} = (20 \text{ x } 0{,}1\mathrm{m}) = 2\mathrm{m}$

Ecuación 8. Ejemplo desplazamiento longitudinal

Se puede observar que el resultado ha sido de 2 metros. Lo que significa que el objeto está a 2m del radar.

Como resumen de estos cálculos, se puede conocer que el objeto no está en movimiento y se encuentra a 2 metros justo en frente de la posición del radar.

Por otro lado, están los datos del mensaje de clusters, el mensaje 0x70C. Del cual se obtienen los siguientes datos: Velocidad relativa del cluster, ángulo del cluster respecto al radar y distancia del cluster al radar.

```
70C,0,700,45,10
```

Ilustración 56. Ejemplo de datos recibidos en el mensaje 0x70C

Al igual que en el caso de los mensajes track (0x60C), el manual también indica qué operaciones seguir para poder obtener los datos exactos de la medición que ha realizado el radar.

Para ello, se van a exponer las fórmulas que han sido utilizadas para la obtención de los resultados de cada tipo de dato.

X = valor obtenido para la velocidad relativa desde el PixHawk.

Y = valor obtenido para el ángulo del cluster desde el PixHawk.

Z = valor obtenido para la distancia al cluster desde el PixHawk.

1. $\mathrm{Vrel} = \left(\mathrm{X\ x}\ \frac{0{,}05\mathrm{m}}{\mathrm{s}}\right) - \frac{35\mathrm{m}}{\mathrm{s}}$

   Ecuación 9. Velocidad relativa

   La medida utilizada para la velocidad relativa, serán los metros/segundo. Al igual que con el mensaje de track, si se recibe un valor negativo, significará que el cluster está acercándose hacia el radar. Mientras que, si el resultado es positivo, significa que este se está alejando.

2. $\mathrm{Azimuth} = (\mathrm{Y\ x\ 2\ grados}) - 90\ \mathrm{grados}$

   Ecuación 10. Valor Azimuth (ángulo)

   Esta fórmula, se utilizará para la obtención del valor del ángulo del cluster. Si el ángulo fuese negativo, este se encontraría a la izquierda del radar. Mientras que si fuese positivo estaría a su derecha. Tiene una función similar a los datos de desplazamiento lateral de los que disponían los tracks en el mensaje 0x60C.

3. $\mathrm{Range} = (\mathrm{Z\ x}\ 0{,}2\mathrm{m})$

   Ecuación 11. Distancia en cluster mode

   Esta es la fórmula para calcular la distancia del cluster al radar. Su resultado solo puede ser un número positivo e indica la distancia en metros hasta el cluster en cuestión con el radar como origen.

Para la mejor comprensión de dichas fórmulas, se va a reflejar un ejemplo de cálculo con los datos de la Ilustración 56. Ejemplo de datos recibidos en el mensaje 0x70C. Los resultados deberían de ser idénticos a los recogidos en el mensaje de tracks 0x60C ya que se los datos han sido recogidos en el mismo entorno.

a. $\text{Vrel} = \left(700 \text{ x } \frac{0{,}05\text{m}}{\text{s}}\right) - \frac{35\text{m}}{\text{s}} = 0\text{m/s}$

Ecuación 12. Ejemplo velocidad relativa

Con este resultado, puede entenderse que el cluster no está en movimiento.

b. $\text{Azimuth} = (45 \text{ x } 2 \text{ grados}) - 90 \text{ grados} = 0º$

Ecuación 13. Ejemplo azimuth

Puesto que los grados respecto al radar son 0, el objeto sigue estando en frente del radar.

c. $\text{Range} = (10 \text{ x } 0{,}2\text{m}) = 2\text{m}$

Ecuación 14. Ejemplo distancia

Y por último, el valor de la distancia al objeto se mantiene en 2m ya que el objeto es un objeto inmóvil que no se ha movido ni modificado en ninguna de las dos recogidas de datos (0x60C y 0x70C).

Una vez han sido explicadas todas las medidas que han sido utilizadas para obtener el valor exacto de cada tipo de dato, se procede a la creación de una serie de gráficas para estos datos de forma que su comprensión sea mucho más sencilla por parte del usuario.

Para la creación de gráficas, se usará el software Matlab. En este entorno, se creará un script que recoja los datos de la hoja de cálculo donde se almacenan los datos y cree graficas con ellos. De forma que puedan explicarse todos los escenarios escogidos de una forma breve y sencilla.

```
figure(101)
plot(status_1.tiempo, status_1.velat);
xlabel('time')
ylabel('velocidad lateral')
hold on
Image = getframe(figure(101));
imwrite(Image.cdata, 'velocidad lateral.jpg');

figure(102)
plot(status_1.tiempo, status_1.velong);
xlabel('time')
ylabel('velocidad longitudinal')
hold on
Image = getframe(figure(102));
imwrite(Image.cdata, 'velocidad longitudinal.jpg');
```

Ilustración 57. Porción de script para la creación de una gráfica en Matlab

Como se puede observar en la imagen de arriba, es una porción de un script de Matlab. Este fragmento, genera dos figuras utilizando dos columnas del fichero .csv. Luego nombra los dos ejes de la gráfica y la crea con los datos seleccionados.

5.2. EVALUACIÓN DE LA SOLUCION FINAL PROPUESTA

En este apartado, se va a llevar a cabo una evaluación de la solución final para asegurar que se cumplen todos los requisitos y verificar la validez de la misma.

Para este proceso de evaluación, se van a implantar una serie de pruebas para todos los casos que recoge el algoritmo y, mediante las gráficas, se corroborará que la información detectada por el radar se corresponde con la realidad.

Todas las pruebas que se han realizado a continuación, debido a la limitación de tener el radar conectado al PixHawk y a una fuente de alimentación un tanto grande, solo han sido posible realizarse en entornos cerrados y con personas como objetos en movimiento.

Para ello, se ha diseñado un esquema que representa como ha sido el entorno de cada prueba y los objetos que participan.

Lo primero, será recibir la información proveniente del radar y hacer que pase por todos los filtros que se han explicado anteriormente. Cuando estos datos están depositados en la hoja de cálculo, el programa Matlab entra en acción y crea todas las gráficas necesarias.

5.2.1. Objeto acercándose hacia el radar (Cluster)

La primera prueba, será haciendo que el sujeto camine hacia el radar. El pensamiento previo para esta prueba fue que la finalidad de este sistema es ir montado en un vehículo y detectar posibles peligros. Por lo que el sujeto trabajaría en forma de vehículo externo que se aproxima al vehículo del sistema y puede provocar un accidente.

Primero, se observarán las gráficas producidas con los datos de los mensajes cluster (0x70C).

La prueba va a consistir en un sujeto caminando desde un extremo de la habitación hasta el radar. Caminará en una trayectoria recta sin desviarse de su objetivo (el radar).

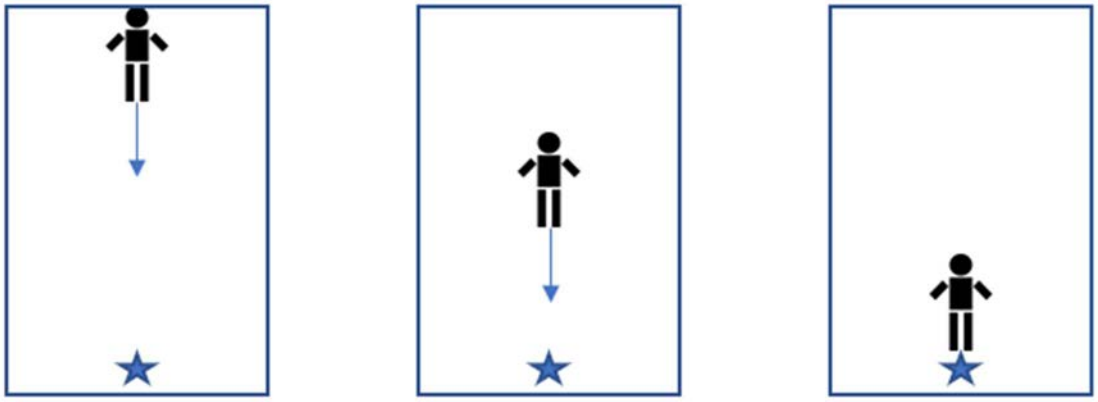

Ilustración 58. Entorno de pruebas "acercándose hacia el radar"

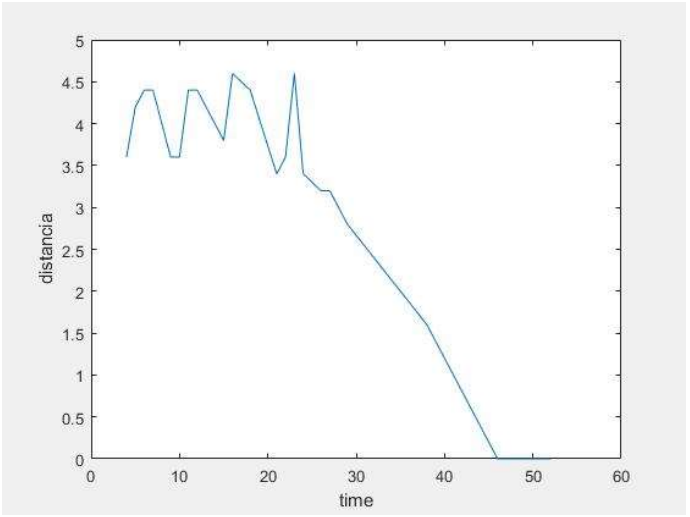

Ilustración 59. Gráfica Distancia para prueba acercándose al radar 0x70C

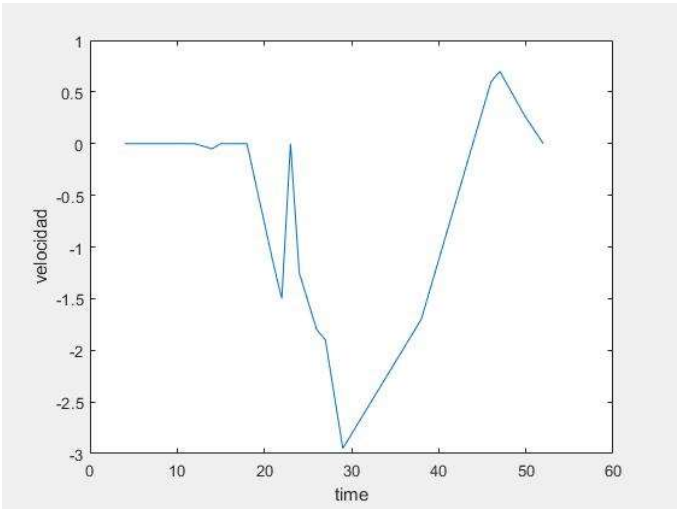

Ilustración 60. Gráfica Velocidad para prueba acercándose al radar 0x70C

Como se puede comprobar, el radar no es preciso al 100% por lo que puede tener pequeñas vibraciones en los datos.

Se ha diseñado un pequeño esquema del entorno de pruebas. Como se puede observar, se incluye un sujeto, una dirección marcada por la flecha y una estrella que es la posición actual del radar.

En la primera gráfica, la de la distancia del cluster, puede observarse que la distancia del mismo es de 4 metros al principio de la prueba. Una vez que este comienza a acercarse al radar, dicha distancia comienza a bajar mientras sigue acercándose a la señal, hasta que está tan junto al radar que la distancia en ese momento será de 0.

En la segunda gráfica, se estudia la velocidad del cluster. Al comienzo de la prueba, este permanecía inmóvil por lo que su velocidad es de 0. Cuando comienza a acercarse al radar, esta velocidad empieza a tener valores negativos puesto que la dirección que tiene es hacia la señal. Puede verse que la velocidad tiene picos en los que aumenta y se acerca a 0. Esto es porque el cluster disminuye su velocidad de acercamiento o, que al frenar de forma brusca, lo interpreta mínimamente como que empieza a alejarse del radar (aunque no sea así) y finalmente, al detenerse en frente del radar, la velocidad termina en 0.

Con toda la información extraída gracias a las gráficas, se ha considerado dar esta prueba como buena ya que demuestra el correcto funcionamiento del sistema.

5.2.2. Objeto acercándose hacia el radar (Track)

Ahora, se procede a la visualización de los gráficos procedentes de la misma prueba pero utilizando los datos incluidos en el mensaje de tracks, el mensaje 0x60C. Para este caso, se dispone de un total de 4 gráficas correspondientes a los 4 tipos de datos provenientes del mensaje.

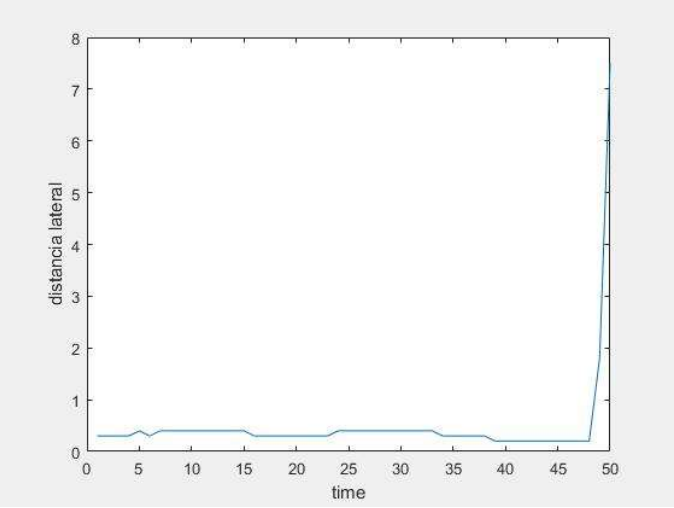

Ilustración 61. Gráfica distancia lateral para prueba acercándose al radar. 0x60C

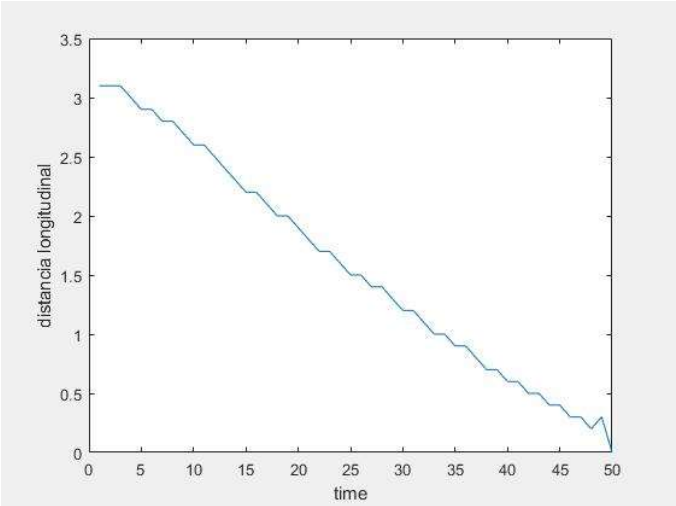

Ilustración 62. Gráfica distancia longitudinal para prueba acercándose al radar. 0x60C

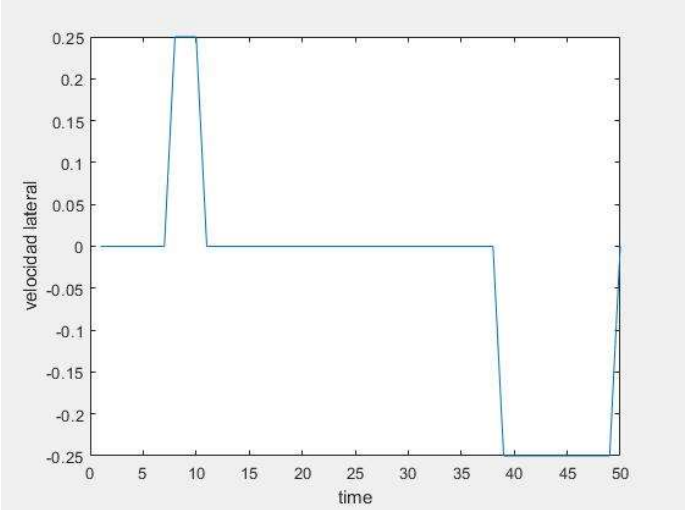

Ilustración 63. Gráfica velocidad lateral para prueba acercándose al radar. 0x60C

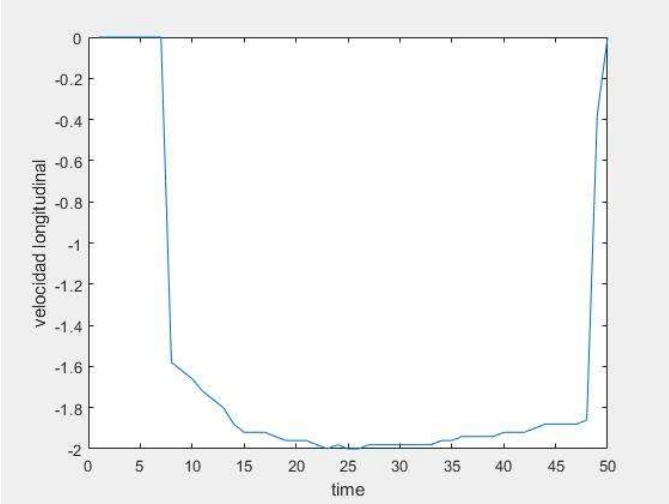

Ilustración 64. Gráfica velocidad longitudinal para prueba acercándose al radar. 0x60C

En el caso de la información de los tracks, se puede verificar que los datos tienen sentido con lo que está sucediendo y que corresponde con lo especificado.

En la primera gráfica (distancia lateral) se entiende que el sujeto se aproxima sin hacer movimientos hacia los lados hasta el momento en el tiempo en el que está muy próximo al radar. En este momento, al ser este casi el único track que detecta (ya que abarca toda la señal enviada por el radar), el valor del desplazamiento lateral se ve afectado. Aunque en realidad, no se haya movido lateralmente.

En la siguiente gráfica, que corresponde a la distancia longitudinal, se puede observar que el sujeto comienza a una distancia de 3 metros y se acerca de forma constante al radar hasta llegar a un rango de 0 metros.

En la tercera gráfica (velocidad lateral), hay datos un tanto diferentes a lo que se espera. Hay una pequeña variación al comienzo ya que marca un aumento en la velocidad lateral, aunque el sujeto solo se esté desplazando longitudinalmente. Esto se debe a que, al ser una persona el sujeto de pruebas, puede interpretar un balanceo del cuerpo como una pequeña señal de desplazamiento hacia un lado. Ya que como se puede observar al final de dicha gráfica, hay un valor exactamente igual, pero con distinto signo que se produce debido a un balanceo del cuerpo hacia el otro lado.

Y, por último, la gráfica correspondiente a la velocidad longitudinal. La gráfica que más información puede dar junto a la gráfica de la distancia longitudinal (para este caso en concreto). Se observa como el sujeto incrementa la velocidad hasta que comienza a llegar al radar por lo que la disminuye hasta pararse por completo.

Al igual que la prueba anterior, se considera que los resultados obtenidos son gratificantes por lo que se da el visto bueno a la prueba.

5.2.3. Prueba del entorno de pruebas

Esta prueba consiste en una imagen de los objetos que detecta el radar. Todos estos objetos se encuentran quietos por lo que el valor de velocidad en ellos será de 0 en todo momento. Por ello, se utilizará el modo “SendTracks” para obtener los datos de desplazamiento lateral y desplazamiento longitudinal y poder situar estos objetos en un mapa del entorno.

Un esquema visual del entorno de pruebas es el siguiente:

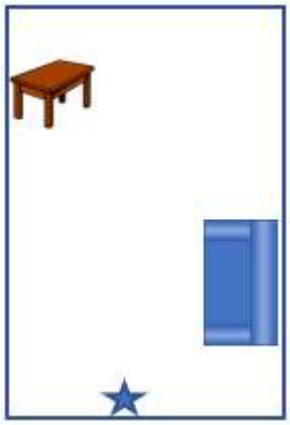

Ilustración 65. Entorno de pruebas con objetos estáticos

Como se puede observar, el radar solo tiene que distinguir los siguientes objetos: un sofá cercano a su derecha, la mesa más alejada ligeramente a la izquierda y la pared de la habitación.

Los resultados, se muestran en un gráfico un tanto distinto a los anteriores. En este caso haremos una representación de los objetos en el espacio:

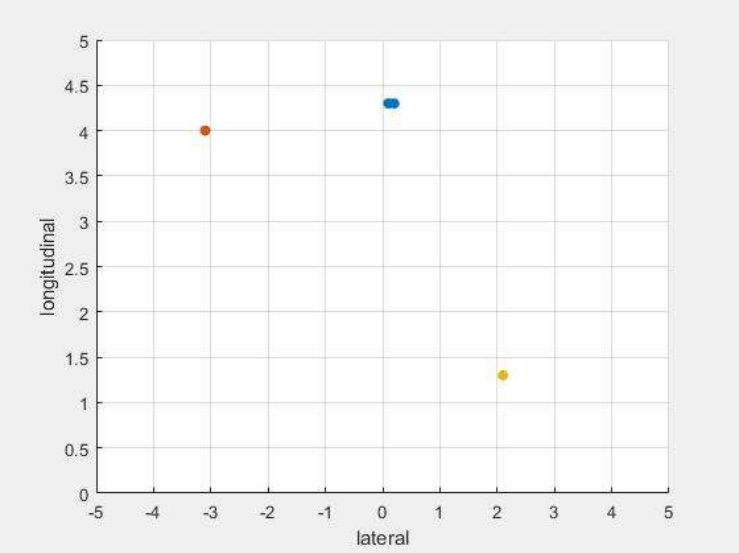

Ilustración 66. Mapa de los objetos estáticos

Como se puede ver en la figura, detecta 3 objetos distintos. El punto azul corresponde a la pared que tiene en frente, por lo que la distancia lateral corresponde a un valor cercano a 0 al ser el punto de la pared que tiene en frente el que va a tomar como referencia.

Por otro lado, el punto naranja corresponde a la posición de la mesa. Esta se encuentra 40 centímetros más cerca del radar que la pared y está situada ligeramente a la izquierda del radar.

Por último, el punto amarillo pertenece al sofá. Este se encuentra más cerca del radar y a la derecha, por lo que marca una distancia lateral de ~ 2 metros.

Esta prueba, como se ha mencionado corresponde con una especie de foto que muestra el entorno en el que se realizan todas las pruebas.

5.2.4. Objeto acercándose con movimiento de zigzag

La siguiente prueba, consistirá en un sujeto acercándose hacia el radar, pero haciendo movimientos de zigzag. En este caso, solo se harán las pruebas con el modo “SendCluster” ya que realizarlo de las dos formas arrojaría los mismos resultados, pero con distintos campos.

El radar debería ser capaz de detectar al sujeto en todo momento en su nueva ubicación y, con los datos del ángulo que debería variar constantemente entre valores positivos y negativos, demostrar que el sujeto se está moviendo hacia los lados.

Para esta prueba, incluiremos el hecho de que, detectando al sujeto moviéndose, también es capaz de detectar a los distintos objetos estáticos que se encuentran en la habitación. Y, que a pesar de pasar por detrás o delante de ellos, no pierde en ningún momento el foco de cada cluster y la información referente a este.

El esquema del entorno de pruebas en este caso sería el siguiente:

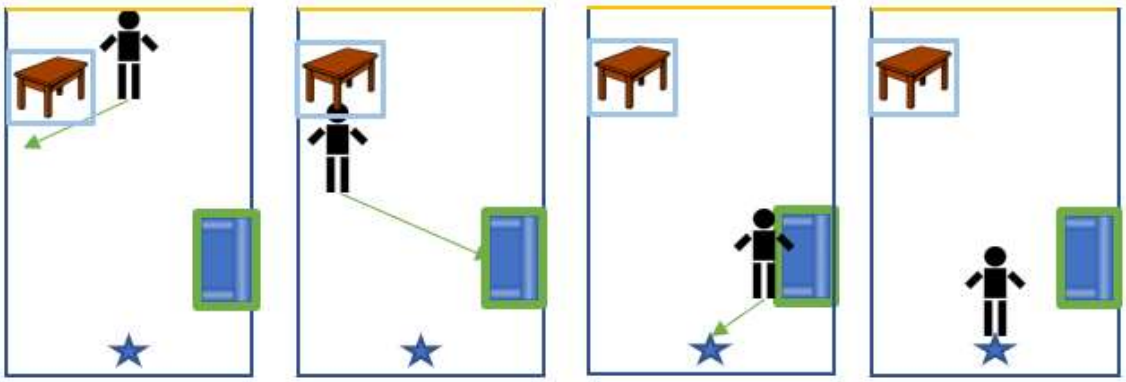

Ilustración 67. Entorno de pruebas "acercándose con movimiento zigzag"

Y los datos obtenidos por el radar:

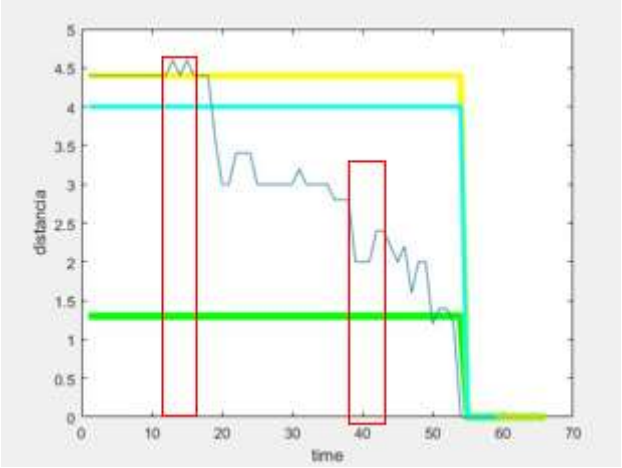

Ilustración 68. Gráfica distancia para prueba con movimiento de zigzag

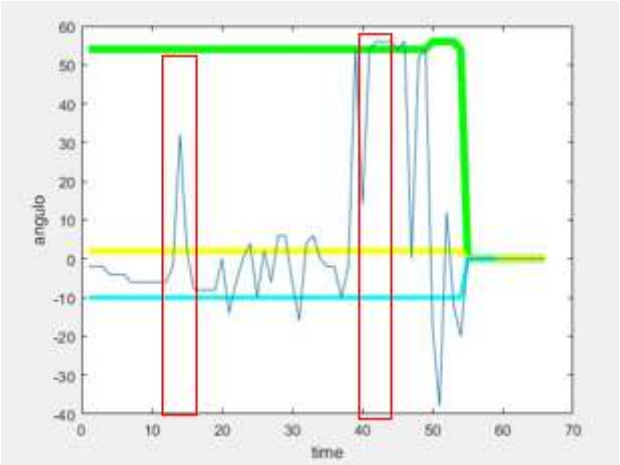

Ilustración 69.Gráfica ángulo para prueba con movimiento de zigzag

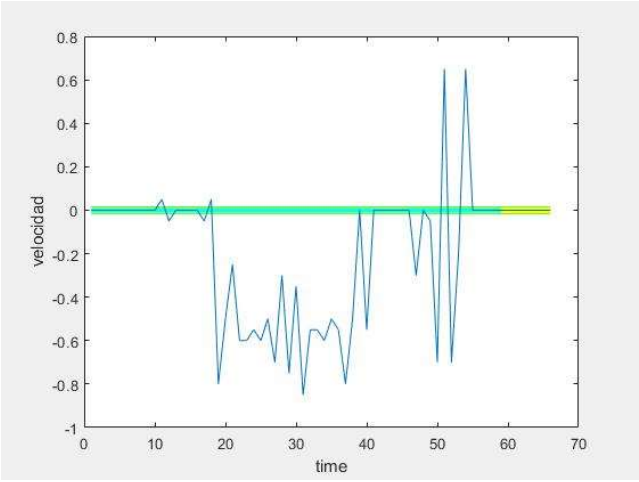

Ilustración 70. Gráfica velocidad para prueba con movimiento de zigzag

Se han indicado en el esquema los colores que corresponden a cada objeto inmóvil de la habitación.

El objeto de color amarillo corresponde a la pared del final. Consultando los datos de las 3 gráficas, se puede verificar que el objeto permanece inmóvil durante toda la prueba (gráfica de la velocidad). Por otro lado, la gráfica de la distancia muestra que se encuentra a 4,4 metros hasta el momento en el que el sujeto se acerca tanto al radar que este solo es capaz de detectar sus medidas. Y, por último, como indica la gráfica del ángulo, la pared se encuentra a ~0 grados. Lo que significa que detecta la parte que tiene directamente delante de la pared y que no varía en ningún momento, por lo que lo tomará como referencia.

Otro objeto que será identificado es la mesa (celeste), que se encuentra en la parte del fondo a la izquierda en la habitación. Corroborando estos datos con las gráficas, efectivamente se encuentra en una posición a -10º del radar, lo que significa que se encuentra a su izquierda (gráfica del ángulo) y a una distancia de 4 metros (gráfica de distancia). Y, como es un objeto inmóvil, su velocidad en todo momento es de 0 metros por segundo.

Por último, el objeto restante, el cual corresponde al color verde, será el sofá. Al igual que los dos anteriores objetos, su velocidad será de 0 metros por segundo durante toda la prueba. Su situación: se encuentra a ~50º del radar, lo que significa que está a su derecha. El valor del ángulo es mucho mayor que el de la mesa ya que al estar más cerca de radar (~1,4 metros según la gráfica de la distancia) influye en este valor.

Para terminar con la prueba, se encuentra la línea fina de color azul. Esta se corresponde con el sujeto moviéndose con movimientos de zigzag hasta llegar a la posición del radar. Se puede observar que su velocidad desde que empieza a moverse en dirección al radar es negativa en todo momento (ya que se acerca) hasta el punto en el que está tan cerca del radar que su frenado lo interpreta como una pequeña aceleración positiva.

Por otro lado, el ángulo es lo que más interesa en este experimento. Se puede observar como el ángulo va intercalando entre los valores positivos y negativos. Lo que significa que el sujeto pasa por delante del radar en varias ocasiones, cambiando su dirección numerosas veces. También pasa por las posiciones de los distintos objetos estáticos y el radar es capaz de detectar a este sujeto de pruebas incluso bajo esas circunstancias, ya que podría confundirse y al estar pegado al sofá (por ejemplo) pensar que es un nuevo objeto o que es el sofá. Por último, puede verse que la distancia va disminuyendo de forma notoria, aunque con alguna pequeña subida en algunos casos. Esto se debe a que en los momentos en los que el sujeto se encuentra en un ángulo con valor elevado (ya sea positivo o negativo) este se encuentra a una distancia del radar mayor que si se encontrase justo delante. Se han insertado una serie de marcas rojas sobre los gráficos que muestran este hecho.

Con todos los detalles de esta prueba y lo que se esperaba de ellas, se da la prueba por buena.

6. SOLUCIONES PREVIAS PROPUESTAS

En este apartado, se van a explicar las diferentes soluciones que se han considerado para el proyecto antes de llegar a la solución final y definitiva que se decidió tomar.

Al haber seguido una planificación en el proyecto basada en entregas de sprints a lo largo del mismo, cada sprint tendrá una serie de diversas soluciones a los problemas que iban encontrando, de entre las cuales se tuvieron que descartar varias. Dado que ya se sabe cuál ha sido la elección final, queda por explicar cuáles han sido esas alternativas que fueron descartadas.

Se hará una explicación por cada sprint, de forma que se pueda tener una visión de las fases del proyecto y cuándo, dónde y por qué aparecieron y por qué fueron descartadas.

6.1.1. Soluciones alternativas Sprint 1

En el primer sprint, se abarcaron las diversas formas de coger la información del radar. Al principio simplemente se recibían datos en crudo sin ningún tipo de procesamiento sobre estos datos. El formato de los datos era código ASCII por lo que era totalmente incomprensible para el ojo humano. En este punto, había tres posibilidades diferentes:

1. Una de ellas, fue almacenar todos los datos en un archivo aparte. Sobre el que, posteriormente, se ejecutaría un script que convertiría estos datos en crudo en algo comprensible por nosotros para poder comenzar con el análisis de los datos. Se descartó esta opción ya que tener que ejecutar por detrás un script se consideró demasiada carga de trabajo en la eficiencia del proyecto.

2. Otra idea fue de igual forma, ejecutar un script sobre el puerto serial que parsee estos datos, guardando la información necesaria en un fichero para su posterior análisis. Al igual que en el caso anterior, se descartó esta opción por la carga que sería tener un script corriendo en todo momento.

3. La última idea, fue intentar programar directamente el microcontrolador de forma que enviase la información de esos datos en crudo tal y como era necesaria.

En este caso, la elección fue la opción número 3. Descartando las dos primeras opciones.

6.1.2. Soluciones alternativas Sprints 2 y 3

Entre estos dos sprints no se detectó ningún problema en especial. Todo funcionó según lo acordado en el sprint 1. El falló más puntual que se detectó fue a la hora de recoger el valor real de los datos. Ya que datos como la velocidad o las distancias podían contener valores decimales y no solamente números enteros, por lo que se manejaron las siguientes posibilidades:

1. Investigar una solución para programar el microcontrolador de forma que pueda trabajar con variables en coma flotante, ya que el Atollic TrueStudio no tenía soporte en este tema. Se descartó porque la investigación sobre el tema no podía tener una fecha de finalización estimada y podía llevar a un cambio de fecha en varias tareas del proyecto.

2. Utilizar la herramienta de Microsoft Office. Más concretamente el libro de cálculo Excel para realizar dichos cálculos y obtener los valores con decimales. Esta es la opción que se tomó para la solución final.

6.1.3. Soluciones alternativas Sprints 4 y 5

Estos sprints comprenden la época del proyecto en la que se trabajó la forma de depositar los datos en el fichero .csv que se necesitaba. Para este fin, se tuvieron tres ideas:

1. Tratar de configurar esta función en el microprocesador también de forma que envíe y guarde los datos directamente en el fichero seleccionado. Esta era la principal solución para el problema de estos sprints. Pero tras intentarlo, se descubrió que el microcontrolador no es capaz de realizar las funciones de guardados en ficheros del sistema de ficheros del ordenador.

2. Ejecutar un script que recoja todos los datos que son enviados por el puerto serial y guardarlos en un fichero .csv que se le indique. Esta opción fue descartada al igual que las anteriores que incluían un script ya que la idea de tener un script adicional corriendo por detrás no era una solución principal. Aunque si fuese la única solución posible, se acataría.

3. Utilizar el comando “screen” con el flag “-L -Logfile” el cual permite indicar la ruta del fichero en la que se desea guardar lo que se envía al puerto elegido. Se optó por esta opción ya que no requería tiempo en programar nada adicional y funciona a la perfección para lo que se necesitaba en ese momento.

6.1.4. Soluciones alternativas Sprint 6

Este corresponde al último sprint antes de la solución final que recoge todas las soluciones tomadas en los anteriores sprints y, por último, las de este.

En este caso, ya se disponían de todos los datos en un fichero .csv que puede ser leído y utilizado fácilmente por la herramienta Matlab. Como para este sprint solo se llevó a cabo la creación de un script de Matlab capaz de general las gráficas no se consideraron más opciones para este fin.

Como aporte final a este apartado, en cuanto a la captación de datos y los entornos de pruebas, hubo varias ideas las cuales se descartaron hasta dar con la mejor idea dentro de las posibilidades que se disponían.

1. Utilizar un objeto como sujeto de muestras e ir realizando movimientos con este delante del radar. Se descartó esta opción ya que el radar no dispone de mucha fiabilidad con objetos que se sitúen a menos de 20 centímetros.

2. Ejecutar el comando para empezar a recibir datos del radar y luego desplazarse hasta el lugar de comienzo de la prueba. Después, borrar los datos producidos por el comienzo de la prueba ya que no corresponden a esta. Se descartó ya que no sería posible conocer qué datos habría que borrar exactamente sin borrar algo importante de la prueba.

3. Por último, la opción de realizar las pruebas con ayuda. Una persona se encargaría de ejecutar el comando “screen” mientras que la otra sería el sujeto de pruebas. Se escogió esta opción ya que desde el primer momento funcionó a la perfección.

7. AMPLIACIONES FUTURAS

En este apartado, se explicarán las posibles ampliaciones futuras del proyecto, su inclusión en otros sistemas e incluso aplicaciones adicionales que pueda tener todo lo realizado este proyecto.

7.1.1. Instalación en otros sistemas

Se comienza explicando cómo podría llevarse a cabo la instalación de nuestro sistema en otros.

Como se ha explicado al comienzo del documento y, se ha dejado entender a lo largo del mismo, este proyecto va dirigido a ir instalado en la estructura de un vehículo. Y, que junto con otros muchos sensores y componentes, pueda estar dotado de una gran autonomía como para poder considerarse un vehículo autónomo y ser competitivo con sus semejantes.

Pero no es el único fin que puede tener este sistema. Al igual que resulta del todo útil en un vehículo de cuatro ruedas, puede resultarlo montado en un dron. Los drones disponen de varios componentes capaces de generar los datos que produce el radar, el LIDAR por ejemplo, pero el radar tiene propiedades que hacen que en ciertas ocasiones sea preferente tener este componente en lugar de un LIDAR.

A parte de la incorporación en vehículos, puede incorporarse en cualquier sistema que precise de un componente que informe de distancias, velocidades o ubicaciones de lo detectado en su entorno para desarrollar su función.

7.1.2. Aplicaciones adicionales del sistema

A pesar de que este proyecto está enfocado a recoger los datos de velocidad, distancias y ángulo del entorno del radar, no es su única aplicación en nuestras vidas.

Por ejemplo, un estudio de la Universidad de Ulm en Alemania, dirigido por Markus Horn, Ole Schumann, Markus Hahn, Jürgen Dickmann y Klaus Diermayer, demuestra que, mediante las mediciones de un radar y la aplicación de diversas fórmulas, es posible conocer que están haciendo los peatones que detecta el radar. Puede saber si estos están caminando, haciendo deporte e incluso hacerse con la estatura de estos. [37].

7.1.3. Ampliaciones futuras del sistema

En el punto actual del sistema, falta mucho por hacer para dar la tarea que quiere abordar el proyecto como terminada. Por ejemplo, podrían añadirse más datos para su análisis y tener así una mejor comprensión de todo el entorno. Podrían realizarse mediciones de la altura de los objetos del entorno (usando una combinación de los datos existentes).

Por otro lado, a la configuración ya disponible, se pueden incorporar avisos o señales al sistema. De forma que cuando detecte que un objeto se dirige hacia el origen (el radar) a una velocidad considerable o que se encuentra demasiado cerca y puede producir un accidente, pueda evitarse antes de que se produzca.

A raíz de esto, una mejora alineada con las alertas por colisión en un vehículo autónomo sería la realización de una funcionalidad de esquiva o reacción. Para ello, deberían incluirse más sensores alrededor del vehículo. De forma que tenga conocimiento de lo que sucede en el entorno que le rodea con una visión completa de 360 grados. De esta forma, en caso de posible colisión, este pueda llevar a cabo un cambio de trayectoria o en caso de detectar mediante el resto de sensores que no es posible evitar dicha colisión, frenar o intentar reducir las consecuencias y los riesgos al mínimo.

Pero para llevar a cabo esta funcionalidad, habría que introducir inteligencia artificial en el sistema y, para la realización de la misma, son necesarias varias pruebas para que el desarrollador y, más adelante el sistema, sepa cómo actuar en estos casos.

8. ANÁLISIS DE LOS RESULTADOS

Este apartado, contendrá información referente a todo el trabajo realizado durante el proyecto y un análisis del tiempo que comprende el inicio y fin para este.

Tras haber realizado todas las pruebas en los escenarios que se han creído oportunos para verificar el correcto funcionamiento del sistema, se puede observar que hay una diferencia notable entre los datos captados con el modo cluster y los captados con el modo track.

Dado que el tipo de envío cluster envía datos de los clusters que detecta el radar en cada ciclo, puede conducir a mediciones fuera de lo normal en algunos casos, por pequeños que sean, pero dando como resultado final unas gráficas no tan limpias como son las de los tracks. Un track, se envía cada varios ciclos de forma que sigue y rastrea

los clusters durante estos ciclos y envían los datos de tracks con un poco más de afinidad ya que sus valores son más consistentes. Algo que se puede ver fácilmente con las gráficas y que se ha explicado anteriormente.

Podemos verificar el correcto funcionamiento del sistema con una prueba que engloba todos los objetos estáticos que detecta el radar y un objeto móvil que pasa por delante y por los lados de los objetos estáticos. El radar es capaz de identificar al objeto en movimiento y diferenciarlo de los objetos estáticos.

Inicialmente, se establecieron una serie de objetivos que fueron: lograr la comunicación con el radar de forma que podamos recoger y enviar datos (para configurar el radar, la configuración de los componentes a un nivel hardware, procesado y conversión de la información relacionada con los obstáculos captados por el radar, recogida de estos datos en una hoja de cálculo .csv, realización de pruebas para verificar que cumple con su función y analizar los datos para conocer qué pasa en todo momento en el área de detección del radar). Como se ha ido explicando a lo largo del documento, todos y cada uno de los objetivos se han ido cumpliendo a medida que avanzaba el proyecto, pudiendo verificar este cumplimiento mediante el estudio de las pruebas realizadas.

También cabe destacar que todos los objetivos personales han sido cumplidos satisfactoriamente. Se ha adquirido un conocimiento muy robusto acerca de todos los componentes que utilizamos en el proyecto pudiendo modificar la programación en ellos para cualquier labor necesaria. También, respecto al documento, se han adquirido varias nociones. Pero esto es algo de lo que hablaremos más adelante.

Como autocrítica, añadir que la investigación de alguna fuente de alimentación similar a la que se ha usado, pero de un peso mucho menor, habría permitido realizar pruebas más diversas. Como pruebas en las que el radar estuviese en movimiento detectando los objetos que tiene en su entorno.

En el caso de la planificación del proyecto, cabe mencionar que se ha seguido tal y como se había especificado en la mayor parte de las tareas definidas. Cumpliendo con todos y cada uno de los sprints correspondientes y entregando los prototipos en cada caso, para una revisión de estos y poder incluir mejoras.

Cabe explicar por qué algunas tareas no se han realizado dentro del periodo que fueron asignadas. En algunas ocasiones, las tareas dentro de un mismo sprint se han realizado antes de lo previsto y se ha podido dedicar más tiempo a otras tareas dentro de este sprint que requerían más gasto temporal y de esfuerzo. Esto es algo a tener en cuenta en futuros proyectos ya que, por ejemplo, los meses de vacaciones en los que las responsabilidades del alumno son menores, se ha aportado más tiempo al proyecto que en otras épocas.

Para asegurar que todos los requisitos establecidos inicialmente han sido cumplidos, se va a realizar un listado con las pruebas que se han llevado a cabo para verificar el cumplimiento de los requisitos:

Requisito - Prueba	Requisito	Prueba de verificación
RF01– PF01 y PF02	El sistema debe permitir al usuario modificar el radar, de forma que elija la forma en la que envía los datos. Ya sean en cluster o en track.	El radar se puede configurar presionando las teclas ‘0’ y ‘1’ de forma que cambia la configuración y comienza a transmitir los datos que se requieren.
RF02- PF03	El cliente podrá parar en todo momento el paso de datos del radar. No quitando la alimentación al radar, sino incluyendo un mecanismo que pueda realizar esta operación.	Se presiona el botón incorporado en el PixHawk de forma que se vuelve al menú y el radar deja de transmitir datos.
RF03- PF04	El sistema debe recoger información del radar en todo momento. Será envío de información a tiempo real de forma que nunca podrá tener demasiado procesamiento y poder perder algún dato. Por lo que el sistema debe estar optimizado de forma que el tiempo máximo de espera entre datos sea de 0,2 segundos.	Una vez que el algoritmo esté finalizado, se aplicarán medidas de optimización de código. Tras esto, se verificará que no haya ningún momento en el que el procesamiento tenga algún retraso o que el sistema espere por más de 0,2 segundos.
RF04- PF05	Dado que el radar envía los datos como bits y esto es algo muy difícil de comprender para el ojo humano, el sistema deberá parsear todos estos datos a formato hexadecimal.	Aplicamos el software correspondiente al sprint 1 y obtenemos los datos en base hexadecimal.
RF05- PF06	Una vez se tienen los datos en formato hexadecimal, ya se conoce que son cada uno. Por lo que se debe identificar primero a qué se refiere y darles el significado en metros, grados o metros por segundo (de un mensaje) según corresponda y luego convertirlos a base decimal para poder trabajar con ellos.	Se utilizan los algoritmos del sprint final. De forma que se obtienen los datos en base decimal pertenecientes a cada tipo de dato.

RF06-PF07	Para el posterior procesamiento y análisis de los datos, una vez transformados los datos en crudo a datos decimales, deben escribirse en un fichero csv.	Se ejecuta el comando 'screen -L -Logfile archivo.csv /dev/ttyACM0 115200' y se guardan los datos en el archivo indicado
RF07-PF08	Se creará un script en el entorno de Matlab para analizar los datos del radar y crear gráficas con las que poder trabajar.	Se ejecutará el script creado (Ilustración 57). De forma que este leerá los ficheros .csv y creará las gráficas correspondientes en cada caso.
RF08-PF09	El componente, deberá modificarse para tener un puerto CAN del que recibirá los datos del radar y mediante un puerto micro USB a USB pasará los datos al ordenador. Por otro lado, se habilitará un input para un cable JTAG para que el ST-link.	Se crea un nuevo puerto para poder conectar el cable JTAG del ST-Link. De forma que al conectarlo al PixHawk este pueda programarse con los algoritmos realizados.
RF09-PF10	Las medidas para la conexión de la fuente de alimentación y el radar deben ser de 12V y 0,5A. En ningún caso puede exceder dichos valores.	Se adquiere una fuente de alimentación regulable. Se establecen los valores de 12 voltios y 0,5 amperios de forma que se puede usar el radar de forma segura.

Tabla XXXVIII. Tabla de verificación de requisitos

9. CONCLUSIONES Y PROBLEMAS ENCONTRADOS

En este apartado, se explicarán los distintos problemas que se han encontrado a lo largo del proyecto y las complicaciones y soluciones que han tenido.

Por otro lado, se expondrán las conclusiones sobre el sistema.

9.1. PROBLEMAS ENCONTRADOS

Durante la realización del sistema que concierne este documento, se han encontrado varios problemas, los cuales se han ido resolviendo a medida que avanzaba el desarrollo.

El primer problema, apareció al comienzo del proyecto. Debido al escaso conocimiento sobre el tema, al principio fue bastante complicado realizar una explicación tan exhaustiva y completa sobre el estado actual de la tecnología que incluye el sistema. Este problema la única solución que podía tener era la dedicación de más tiempo para investigar y aprender el sobre tema a tratar.

El siguiente problema identificado, fue a la hora de utilizar la librería que permitía programar el microcontrolador. La labor realizada para solucionar este problema fue muy similar a la anterior. A base de varias pruebas y de varias fases de ensayo y error, se pudo entender cómo funcionan las librerías y así proceder con el uso de las funciones disponibles en las mismas.

También hubo un problema más técnico y de hardware. El microprocesador tuvo un problema durante una prueba y dejó de funcionar completamente, de forma que el resto de las pruebas y la continuación del proyecto se vio en serios problemas. Tras la investigación en distintos foros y varios intentos día tras día, se solucionó el problema y con eso, se continuó con el desarrollo del proyecto hasta su finalización.

Por último, varios problemas con el documento. Ya que es la primera vez que se realiza un proyecto tan profesional y completo como este, ha sido complicado entender como estructurar el contenido del documento. Además, nunca se había intentado realizar en solitario ya que, normalmente, durante el grado, este tipo de trabajos se han exigido hacer en grupos de varias personas.

Estos problemas engloban los puntos más complicados y duros del proyecto en los que más tiempo y dedicación se tuvieron que invertir.

9.2. CONCLUSIONES

El proyecto que se trata en este documento ha consistido en el desarrollo del software para un radar y el análisis de los datos que se reciben del mismo. Es una tecnología pensada para estar integrada en un coche autónomo y ayudar en sus funciones.

El resultado final, es un sistema capaz de traducir los datos en bits en crudo que proceden de un sensor radar, guardar estos datos en un archivo que no solo sirva para guardarlos, sino que desempeñe funciones de separación de los datos para que finalmente, sean analizados para conocer los casos de las pruebas y lo que ha ocurrido en el entorno de una forma precisa y detallada. Añadir que se ha concluido la finalidad del radar en las pruebas realizadas, validando en una serie de casos la capacidad de detección y discriminación de objetos en situaciones de prueba previas a la siguiente fase que evaluará su funcionamiento en condiciones reales (exterior, con el equipo en una plataforma en movimiento).

Los objetivos del proyecto impuestos no solo por el cliente, sino que también por mí mismo se han cumplido en su totalidad de forma que puede decirse que el proyecto actual está terminado. Pero esto no significa que tenga que ser el fin de la investigación en este ámbito. Actualmente, existen muchos más sistemas similares o incluso mejor desarrollados por las grandes firmas automovilísticas que pueden hacer el mismo trabajo que este proyecto, pero llevándolo a un entorno práctico y con situaciones totalmente reales.

La finalidad una vez llegados a este punto, es dejar un nuevo punto de partida al próximo que se vea interesado por este ámbito. Hay una infinidad de tareas pendientes por hacer hasta llegar a lo que recalcamos al principio del documento: un vehículo totalmente autónomo y capaz de tomar decisiones por sí mismo, siempre teniendo en cuenta las vidas humanas y la evasión de accidentes como primera opción en cualquier situación.

Personalmente, antes del comienzo de este proyecto no contaba con muchos conocimientos de los vehículos autónomos y de su legislatura actual en España. Y, mucho menos conocimiento sobre la configuración de un radar o un microprocesador al que va conectado. Pero es un tema realmente interesante y que capta toda atención de inmediato y puede sumergir a alguien en un mundo de ideas, todas ellas verdaderas y posibles, para el desarrollo de esta tecnología.

Para finalizar, decir que la investigación en este tema es algo que en nuestra carrera debería ser algo fundamental para conocer de una forma más cercana componentes hardware así de complejos e interesantes como es el caso. De esta forma, mediante educación sobre el tema, podremos avanzar con un paso más firme en la evolución de estos sistemas autónomos.

10. BIBLIOGRAFÍA

[1] "Vehículo autónomo." 2018 [Online]. Available: https://es.wikipedia.org/w/index.php?title=Veh%C3%ADculo_aut%C3%B3nomo&oldid=110361553.

[2] "Radar." 2018 [Online]. Available: https://es.wikipedia.org/w/index.php?title=Radar&oldid=110012346.

[3] N. López, "Legislación sobre el coche autónomo en España: queda mucho por hacer," 2017. [Online]. Available: https://www.autobild.es/reportajes/legislacion-coche-autonomo-espana-queda-mucho-hacer-179660.

[4] A. Vigil, "¿Qué leyes habría que cambiar antes de que llegue el coche autónomo?," 28-Aug-2017. [Online]. Available: https://cincodias.elpais.com/cincodias/2017/08/07/legal/1502093470_669276.html.

[5] "Metrica." [Online]. Available: https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html.

[6] "Coche autónomo, el estado del arte: Hardware y Software [PODCAST]," 2017. [Online]. Available: https://programarfacil.com/podcast/coche-autonomo-estado-del-arte/.

[7] A. M. Monografias.com, "Vehículos autónomos - Monografias.com." [Online]. Available: https://www.monografias.com/trabajos109/estado-del-arte-vehiculos-autonomos/estado-del-arte-vehiculos-autonomos.shtml.

[8] A. M. Monografias.com, "Lidar y radar" 5-Abr-2018 [Online]. Available: http://www.conecband.com/entrada/1036/lidar-y-radar-sensores-clave-del-vehiculo-autonomo/.

[9] "Introducción al Controlador de Vuelo de Drones PIXHAWK - Hardware libre." [Online]. Available: http://gidahatari.com/ih-es/introduccion-al-controlador-de-vuelo-de-drones-pixhawk-hardware-libre.

[10] "Pixhawk series." [Online]. Available: http://docs.px4.io/en/flight_controller/pixhawk_series.html.

[11] "ST-Link-V2." [Online]. Available: https://www.st.com/en/development-tools/st-link-v2.html.

[12] "Features." [Online]. Available: https://atollic.com/truestudio/features/.

[13] "TrueSTUDIO." [Online]. Available: https://atollic.com/truestudio/.

[14] "Home." [Online]. Available: https://atollic.com/. [Accessed: 16-Sep-2018]

[15] "Pixhawk stm32f427vit6 replacement to rev. 3." [Online]. Available: http://discuss.px4.io/t/pixhawk-stm32f427vit6-replacement-to-rev-3/3385/11.

[16] M. Norlander, “Migrating from Eclipse/GCC to Atollic TrueSTUDIO,” 25-Mar-2015. [Online]. Available: http://blog.atollic.com/migrating-from-eclipse-/-gcc-to-atollic-truestudio.

[17] “TrueSTUDIO.” [Online]. Available: https://www.st.com/en/development-tools/truestudio.html

[18] “Termite Alternatives and Similar Software - AlternativeTo.net.” [Online]. Available: https://alternativeto.net/software/termite/.

[19] “Comunicaciones serie con PuTTY.” 2014 [Online]. Available: https://polaridad.es/usb-serie-terminal-putty/.

[20] “5 Linux / Unix Commands For Connecting To The Serial Console.” 2012 [Online]. Available: https://www.cyberciti.biz/hardware/5-linux-unix-commands-for-connecting-to-the-serial-console/.

[21] “Scilab.” [Online]. Available: https://es.ccm.net/download/descargar-10670-scilab.

[22] “Home.” [Online]. Available: http://scilab.io/. [Accessed: 16-Sep-2018]

[23] “STSW-STM32065.” Oct 2015[Online]. Available: https://www.st.com/en/embedded-software/stsw-stm32065.html

[24] “GPIO.” 2018 [Online]. Available: https://es.wikipedia.org/w/index.php?title=GPIO&oldid=105523451.

[25] “Library 53- GPIO for STM32F4.” 2015 [Online]. Available: http://stm32f4-discovery.net/2015/03/library-53-gpio-for-stm32f4/.

[26] “Bus CAN.” 2018 [Online]. Available: https://es.wikipedia.org/w/index.php?title=Bus_CAN&oldid=109764446.

[27]

“Continental Automotive.” [Online]. Available: https://www.continental-automotive.com/. [Accessed: 16-Sep-2018]

[28] “3D Robotics Pixhawk Flight Controller - Not Chinese Clone.” [Online]. Available: https://www.uavsystemsinternational.com/product/3d-robotics-pixhawk-flight-controller/.

[29] “STSW-STM32065.” Ene 2013 [Online]. Available: https://www.st.com/en/development-tools/st-link.html [Accessed: 16-Sep-2018]

[30] G. Molina, “TEMA 3.2 FUENTE DE ALIMENTACION.” 2015 [Online]. Available: http://cursoreparacionmoviles.blogspot.com/2015/09/tema-32-fuente-de-alimentacion.html.

[31] “PORTATIL DELL E5430.” [Online]. Available: https://www.imposivle.es/tienda/portatiles-reestreno/portatil-dell-e5430-4/.

[32] “El papel de los sensores y los actuadores en vehículos autónomos | CLR.” 2017 [Online]. Available: https://clr.es/blog/es/sensores-actuadores-vehiculos-autonomos/.

[33] “Qué es un LIDAR, y cómo funciona el sensor más caro de los coches autónomos,” 28-Sep-2017. [Online]. Available: https://www.motorpasion.com/tecnologia/que-es-un-lidar-y-como-funciona-el-sistema-de-medicion-y-deteccion-de-objetos-mediante-laser.

[34] “Localizador GPS Coches / Barcos / Motos.” [Online]. Available: http://seguridad.tecnovisionmalaga.com/localizador-gps-coches/.

[35] “IMU,” 2017. [Online]. Available: https://racelogic.support/VBOX_Mining/Hardware_Info/IMU. [Accessed: 16-Sep-2018]

[36] “SENSOR DE FATIGA – LAXGPS.” [Online]. Available: http://laxgps.cl/webwp/sensor-de-fatiga/.

[37] “Motion Classification and Height Estimation of Pedestrians Using Sparse Radar Data”. Markus Horn, Ole Schumann, Markus Hahn, Jürgen Dickmann and Klaus Dietmayer
Sen. 12th Symposium Sensor Data Fusion. Trends, Solutions, and Applications 9 - 11 October 2018

[38] “Real Evaluation for Designing Sensor Fusion in UAV Platforms.” Jesús García, Jose Manuel Molina, Jorge Trincado. Information Fusion Journal, 2018 (in press)

[39] “Así aprenden a conducir los coches autónomos.” 16-Mar-2017 [Online]. Available: https://www.autocasion.com/actualidad/reportajes/asi-aprenden-conducir-los-coches-autonomos.

11. ACRÓNIMOS Y DEFINICIONES

Sensor: componente con el que se pueden medir varios parámetros del entorno del mismo.

Radar: sensor que mide los objetos de su entorno a través del envío de señales que rebotan en estos e informan de su ubicación contante respecto al radar.

Vehículo autónomo: es un vehículo capaz de tomar decisiones de forma autónoma. Sin interacción del piloto.

VA: vehículo autónomo.

Microcontrolador: circuito capaz de ejecutar las órdenes dadas por el algoritmo que tiene grabado.

Matlab: herramienta software matemática que permite la realización de las gráficas en el proyecto.

Atollic TrueStudio: entorno de programación para el microcontrolador y el programador ST-Link.

CSV: (Comma separated values). Formato de fichero que permite guardar los datos en una hoja matemática.

LIDAR: (Light Detection and Ranging). Es otro tipo de sensor que funciona mediante la emisión de un haz láser.

GPS: (Global Positioning System). Componente utilizado para determinar la posición de este sensor en cualquier lugar de la tierra.

Hardware: elementos físicos que forman el sistema.

Hardware libre: componentes cuya información es abierta de cara al público para su libre configuración.

ST-Link: programador para microcontroladores.

PixHawk: microcontrolador que puede programarse para funcionar según el algoritmo grabado en su memoria.

Caja negra: dispositivo que registra la actividad sucedida. Se usa mayormente en caso de accidentes para saber que ha sucedido antes o durante el mismo.

Métrica versión 3: metodología utilizada para la sistematización del ciclo de vida del proyecto.

Sprint: iteración en el proyecto. Es una pequeña entrega de una parte del proyecto, la cual se estudiará para encontrar posibles mejoras de cara al próximo sprint.

Ciberataque: ataque que se lleva a cabo sobre un sistema informático.

Software: programas y algoritmos que componen el sistema.

Gantt: gráfica para exponer la planificación del proyecto.

Parsear: traducir unos datos de un lenguaje o formato a otro.

Lenguaje C: lenguaje de programación orientado a la implementación de sistemas operativos.

Interfaz: conexión entre dos sistemas mostrada de forma visual, por lo que el usuario tiene una concepción mejor de lo que está pasando entre ellos.

Azimuth: medida del ángulo de la orientación sobre la superficie de una esfera.

CAN: (Controller Area Network). Es un protocolo de comunicaciones.

Linux: sistema operativo de código abierto.

Ubuntu: distribución del sistema Linux.

Cable JTAG: cable para programadores y debuggeadores de microcontroladores.

Debugear: depurar un sistema para encontrar fallos.

Mapear: reducir un gran bloque de información a uno más pequeño que solo contiene la información útil para el tema a tratar.

Bit: (Binary digit). Es una unidad de almacenamiento de la memoria digital.

Quantum: periodo de tiempo que dura un ciclo en el sistema.

ID: código identificador.

Array: zona contigua de memoria que contiene varios elementos.

Float: tipo de variable donde se pueden almacenar números con decimales.

12. PROJECT SUMMARY

12.1. ABSTRACT

The objective of this document is to explain, through information obtained from different sources and personal experiences and the use of illustrations and tables, how the realization of the project has been carried out. The project objective is to receive and analyze the data provided from a radar sensor. So, in future projects, it can be used for the evasion of obstacles in an autonomous vehicle.

All the measures taken to achieve the objective of the document are carried out in 3 different stages:

1. Component configuration. The first step for everything to work is to have clear the flow of information and the connection that the components of the project will have. For your best understanding, the following structure has been designed.

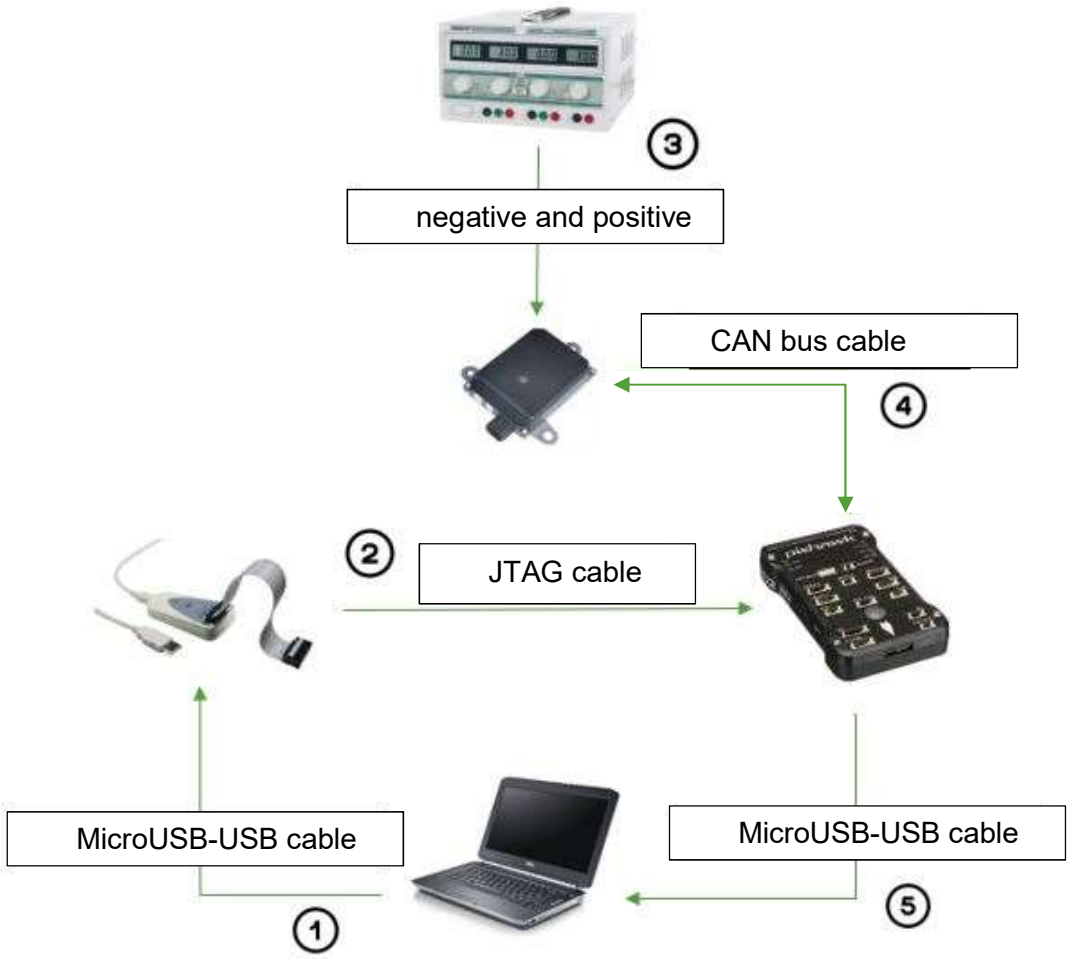

Ilustración 71. Schema of project connections and information flow [30] [31]

The squares explain the wires connecting the components between them. While the numbers indicate the order of the flow of information in the system.

2. Programming the microcontroller and the creation of scripts for the analysis and creation of graphs in Matlab. In this case, the tool used to program the microcontroller will be Atollic TrueStudio. So it can send the radar data the specified format. These data, will be saved in. csv files for later analysis with Matlab scripts. There are included options to switch between the types of radar data output.

3. Finally, the performance of the tests and their corresponding analysis. Tests simulate road traffic. It has static and other moving objects that simulate a dangerous approach that can end in an accident. Once the data has been analyzed, it is verified that they have the expected results.

The results of all the tests have been successfully completed and they verify the correct functioning of the entire system. Thanks to the creation of different graphics to understand each test, it has been easier to verify that the tests meet all the requirements of the system. Also, to explain in this document in an easy and clear way the results obtained.

The conclusion of this project is that indeed, the expectations and the correct functioning of the system have been fulfilled. The planning imposed at the beginning has been largely fulfilled and the problems encountered during the development have been solved personally.

Keywords: sensor, radar, processing, analysis, autonomous vehicle.

12.2. OBJETCIVES

The main objective of the development is to develop a system that access and uses the data from the radar for the evasion of obstacles. This will be made by processing and analyzing the data received from the radar. Once it is processed and analyzed, it will be able to know if it is a dangerous situation and react to finish the path specified by the user.

This design can be divided into two target groups:

12.2.1. Project objectives

- Achieve a communication with the sensor. So the radar can be configured and thus receive all the data it sends according to the set configuration.
- Configuration, modification (if necessary) and connection of the hardware components for their correct operation.
- Process information obtained of possible obstacles in the trajectory.
- Collect data in a .csv file that allows us to work with this information.
- Perform all the necessary tests to ensure the correct functioning of the radar according to the requirements established.
- Analyze the data received to know what is going on within the radar field of view.

12.2.2. Personal objectives

- Learn the operation and configuration of the different devices used throughout the project: radar, ST Link, adjustable power supply and PixHawk.
- Go deeper into different programming languages. Some that I already knew and others totally new to me but that are essential in the course of the project.
- Understand the different ways of data traffic. In this case, the CAN Bus.
- Gain experience with the realization of the documentation on a project of these characteristics.

12.3. SOCIO-ECONOMIC IMPACT

12.3.1. Project benefits

The benefits derived from the development of this system have as much benefits in innovation as benefits in the business sector and society. So they will be explained separately:

On the one hand, the benefits of innovation. The service that will be developed is a novelty in today's society. Many large companies are entirely focused on the development of similar products to this project, but none have managed to reach a perfect and error-free system. So it can mean both a novelty and an improvement in current technology. It will offer a guarantee of the data thanks to an exhaustive analysis of it, so it will gain in security and benefits of the system.

It will also be necessary to train the staff in the knowledge of the hardware and software that is needed for the project. Since all the tools have connection between them and its operation must be known perfectly.

In terms of costs, once the system is included into a vehicle, the price will increase considerably due to the new features that it includes with respect to a traditional vehicle.

Work and environmental impact: new jobs will be created for expert profiles on the subject and for those who can educate the current employees. Since the work includes both hardware components and software systems, there will be two different teams that take care of this once the production process begins in a company. In the environmental field, it is intended to include this proposal in autonomous vehicles that use electric motors mainly. So it is a great help for the high levels of pollution detected in all the main cities of the world due to the emission of gases produced by cars using fuel.

As result, business relations will increase greatly. These products can be internationalized in such a way that any country can enjoy this technology according to the rules stipulated for its use. New markets will be accessed due to the inclusion of new technologies, automotive and informatics (for the use of artificial intelligence in the processing and analysis of data).

On the other hand, we find the benefits in the business sector and society. In this sector, a significant increase in employment will take place. These new members must have technological knowledge associated with the area in which they will collaborate.

Thanks to this, it will be a competitive component in the automotive sector and in the artificial intelligence sector from the analysis and results of the data.

The quality of life will improve as it frees people from doing a job that, to many people every day, takes several hours and can produce stress at peak times: driving. Also with the environmental improvement that it entails (explained at this point above), it will provide society with a more comfortable and healthy life.

12.3.2. Plan for dissemination and disclosure of results

Since the purpose of this system is to include it in a vehicle and, subsequently, to obtain the largest number of sales with it (with the technological and environmental contribution to the company), the dissemination of this system will be realized through the following media:

1. Publication in scientific and technology sector journals: the results of the research and the development of the system will be published in magazines to let know the whole technological sector what has been achieved.

2. Training activities: as previously specified, all system components management is very complicated. So there will be training days for new and old employees.

3. Public dissemination: since one of the purposes of this system (once assembled and working together with other systems in a VA) is the sale to the public, they will be informed of the features offered in press conferences, advertisements and explanatory pamphlet.

4. Dissemination on internet: there will be created sections in the websites for sale for a first view of what it offers and even some kind of demonstration for interested.

5. Previous agreements with companies or public entities: governments and companies know that awareness of the environment today means a lot to society. That is why, agreements will be created with these companies for the implantation of vehicles with our system in their cars. (Taxis, State vehicles, etc.).

12.3.3. Plan of exploitation of the results

To carry out the exploitation of the results of this research, the following aspects will be followed:

1. The final system is capable of capturing the data captured by the radar so that it can be analyzed to identify possible obstacles and create collision alerts. Given the current research in the sector, this is a great progress in a global novelty that expands the options in the current technological environment. The objective of this project is to create a system that can be included in a prototype and perform the necessary tests for your full performance guarantee.

2. It is possible to identify four different roles in the project: system developers, vehicle developers, developers of the rest of systems and entrepreneurs who buy all these technologies. Each participant has an idea for the final product with the integrations of all the parties. All the developers are looking to create a compact system that works perfectly by joining all the parts. While entrepreneurs are looking for a useful product with which they can achieve the greatest number of sales and benefits.

3. As indicated above, the intellectual property of the system developed in this project belongs to the group described later in the organization of the project. Whenever a joint system or external component uses this system will have to include the team as creators or collaborators of the system that included it (the one developed in this project).

4. The users to whom the system market should be directed to the owners of researches referring to the autonomous vehicle. It will also include in this point users who develop systems similar to this and need it to perfect theirs.

12.4. ANALYSIS OF THE RESULTS

This section will contain information concerning all the work done during the project and an analysis of the time that includes the beginning and the end.

After having done all the tests in the scenarios that have thought opportune to verify the correct functioning of the system, it can be observed that there is a notable difference between the data captured with the cluster mode and those captured with the track mode.

Since the SendCluster type sends data from the clusters that detects the radar in each cycle, it can lead to out of the ordinary measurements in some cases, normally small, but giving as a result some not so clean graphics such as the tracks. A track is sent every several cycles so that it follows and tracks the clusters during these cycles and sends track data with a little more affinity because their values are more consistent. Something that you can easily see with the graphs and that has been explained above.

We can verify the correct functioning of the system with a test that includes all the static objects that detects the radar and a mobile object that passes in front and on the sides of the static objects. The radar is able to identify the moving object and differentiate it from static objects.

Initially, a series of objectives were established that were: to configure a communication with the radar so we could collect and send data (to configure the radar, the configuration of the components to a hardware level, processing and conversion of the information related to the obstacles captured by the radar, collecting these data in a .csv file, realization of tests to verify that it fulfills its function and analyze the data to know what happens all the time in the area of radar detection). As has been explained throughout the document, each one of the objectives have been fulfilled as the project progressed, being able to verify this fulfillment by studying the tests that have been realized.

Additionally, personal objectives have been satisfactorily fulfilled. I have acquired a very robust knowledge about all the components that are used in the project, being able to modify by programming them for any necessary work. Also, referring to the document, several notions have been acquired. But this is something we will talk about later.

As self-criticism, the investigation of some power supply like the one which has been used, but much lighter, would have allowed to carry out more diverse tests. As tests in which the radar was in motion detecting the objects it has in its environment.

In the case of project planning, it has been followed as specified in most of the tasks defined. Fulfilling each and every one of the corresponding sprints and delivering the prototypes in each case, for a review of these and to include improvements.

It should be explained why some tasks have not been done within the period they were planned. On some occasions, the tasks within the same sprint have been done ahead of schedule and this made possible to devote more time to other tasks within this sprint that required more temporary and effort spending. This is something to consider in future projects because, for example, the months of holidays in which the responsibilities of the pupil were minor, has meant more time to the project than in other times.

To ensure that all the requirements initially established have been fulfilled, there is a list with the tests that have been carried out to verify the fulfillment of the requirements:

Requirement-proof	Requirement	Verification Test
RF01 – PF01 and PF02	The system must allow the user to modify the radar, so he can choose the way in which the data is sent. Whether it's cluster or track.	The radar can be configured by pressing the ' 0 ' and ' 1 ' keys so it changes the settings and starts transmitting the required data.
RF02-PF03	The client will be able to stop at all times the data passing of the radar. Not by removing the power to the radar but including a mechanism that can perform this operation.	The built button on the PixHawk is pressed so it returns to the menu and the radar stops transmitting data.
RF03-PF04	The system must collect radar information at all times. It will be sending information in real time, so it can never have too much processing that can finish on data losing. So the system must be optimized and the maximum wait time between data is 0.2 seconds.	Once the algorithm is finished, code optimization measures will be applied. After this, it will be verified that the processing has not any delay or that the system waits for more than 0.2 seconds.

RF04-PF05	Since the radar sends the data as bits and this is a very difficult to understand for the human eye, the system should parse all of this data to hexadecimal format.	We apply the software corresponding to sprint 1 so we can obtain the data in hexadecimal base.
RF05-PF06	Once there is data in hexadecimal format is easy to identify the data that belongs to each type of data (velocity, range, etc.). Therefore, it must first identify what type of data it is referring to and give them the meaning in meters, degrees or meters per second (of a message) and in order to work with them, convert the data into decimal base.	The final sprint algorithms are used. So each fragment belonging to the different data types are received in decimal base.
RF06-PF07	For the subsequent processing and data analysis, once the raw data has been transformed into decimal data, it must be written in a csv file.	The command 'Screen –L -Logfile . csv File/Dev/ttyACM0 115200 ' is executed and it saves the incoming data in the indicated file.
RF07-PF08	A script will be created, in Matlab environment, to analyze the radar data and create graphs that will be used for the explanation of the test.	The created Matlab script will be executed (illustration 57). So this will read the .csv files and create the corresponding graphs in each case.
RF08-PF09	The component, must be modified to have a CAN port where it will receive the data from the radar and, through a micro USB to USB cable will pass the data to the computer. On the other hand, an input for a Cable JTAG to connect the ST-link will be made.	A new port is created to allow connecting the ST-Link's JTAG cable. So when ST-Link is connected to the PixHawk it can be programmed with the algorithms performed.
RF09-PF10	The measurements for the connection of the power supply and the radar must be of 12v and 0,5a. These values must not change in any case.	An adjustable power supply is acquired. The values of 12 volts and 0.5 amps are set so that the radar can be safely used.

Tabla XXXIX. Table for requirements verification

12.5. CONCLUSIONS AND PROBLEMS ENCOUNTERED

This section, will explain the different problems that have been encountered throughout the project and the complications and solutions they have had. Also, the conclusions about the system will be exposed.

12.5.1. Problems encountered

During the implementation of the system that concerns this document; several problems have been found and resolved as development progressed.

The first problem appeared at the beginning of the project. Due to the lack of knowledge on the subject, it was very difficult at the beginning to make such an exhaustive and complete explanation of the current state of the technology included in the system. This problem the only solution it could have was the dedication of more time to investigate and learn about the topic.

The next problem identified, was when using the library that allowed programming the microcontroller. The work done to solve this problem was very similar to the previous one. After several tests and phases of trial and error, it was possible to understand how the libraries worked and thus proceed with the use of the functions available in them.

There was also a more technical and hardware problem. The microprocessor had a problem during a test and stopped working completely, so the rest of the tests and the continuation of the project was in a serious trouble. After investigating in different forums and several attempts day after day, the problem was solved. That made possible again to continue with the development of the project until its completion.

Finally, several problems with the document. Since this is the first time a project as professional and complete as this is done, it has been difficult to understand how to structure the content of the document. In addition, I had never attempted to do this alone because, normally, during the degree, this type of work has been demanded to do in groups of several persons.

These problems encompass the most complicated and hard points of the project in which more time and dedication had to be invested.

12.5.2. Conclusions

The project in this document consists on the development of the software for a radar and the analysis of the data received from it. It is a technology designed to be integrated into an autonomous car and assist in its functions.

The final result, is a system capable of converting the data from raw bits, that come from a radar sensor, into decimal base numbers. Save this data in a file that not only is used to save them, also performs data separation functions. So finally, they are analyzed to know about the test cases and what has happened in the environment in a precise and detailed way. The purpose of the radar has been completed in the tests carried out, validating in a series of cases the capacity of detection and discrimination of objects in test situations before the next phase that will evaluate its operation in real conditions (outside, with the equipment on a moving platform).

The objectives of the project imposed not only by the client, also by myself have been fulfilled in its entirety in a way that can be said that the current project is finished. But this does not mean that it has to be the end of research in this area. Nowadays, there are many similar systems or even better developed by the big automobile companies that can do the same work as this project but, taking it to a more practical environment and with totally real situations.

The goal once reached at this point is to leave a new starting point to the next person that is interested in this area. There are infinity tasks to reach what we explained at the beginning of the document: a totally autonomous vehicle, able to make decisions for itself, always taking into account the human lives and the evasion of accidents as first option.

Personally, before the beginning of this project I did not have much knowledge of the autonomous vehicles and their current legislature in Spain. And, much less knowledge about the configuration of a radar or a microprocessor to which it is connected. But it is a really interesting subject and captures all attention immediately, so it can immerse anyone in a world of ideas, all of them true and possible, for the development of this technology.

Finally, I must say that the investigation on this topic is something that should be fundamental in our career to know in a closer way the hardware components as complex and interesting as in this case. In this way, through education on the subject, we will be able to move forward with bigger steps in the evolution of these autonomous systems.