Proyecto Fin de Máster
Máster en Ingeniería Industrial

Arquitectura y Desarrollo Software de un Robot Terrestre no Tripulado para Agricultura

Autor: Víctor Quesada Conejero

Tutor: Ángel Rodríguez Castaño

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Proyecto Fin de Máster
Máster en Ingeniería Industrial

Arquitectura y Desarrollo Software de un Robot Terrestre no Tripulado para Agricultura

Autor:
Víctor Quesada Conejero

Tutor:
Ángel Rodríguez Castaño
Profesor Contratado Doctor Interino

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Proyecto Fin de Máster: Arquitectura y Desarrollo Software de un Robot Terrestre no Tripulado para Agricultura

Autor: Víctor Quesada Conejero
Tutor: Ángel Rodríguez Castaño

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

A mi familia y amigos

Resumen

La agricultura de precisión representa el futuro de la producción agrónoma. Mediante la implementación de sistemas inteligentes se puede mejorar la calidad del terreno, realizar una detección temprana de plagas y enfermedades o incluso, realizar una recolección autónoma de los frutos maduros. Por ello representa un área del desarrollo tecnológico perfecto para realizar transferencia de tecnología de otros ámbitos.

El objetivo de este trabajo ha sido diseñar y desarrollar una arquitectura software, así como varios de sus módulos para un caso de uso concreto de la agricultura de precisión, la recolecta autónoma de brócoli ecológico. Mediante experimentos, tanto en simulación como en terreno bajo unas condiciones similares, se han probado los algoritmos desarrollados y se han evaluado sus características principales. Entre las sugerencias para trabajos futuros se incluye: el estudio de nuevas maniobras para la realización de giros, la integración de visión en el sistema de navegación mediante algoritmos tipo vSLAM así como la exploración de nuevos sensores para la detección de surcos en el terreno.

Abstract

Precision agriculture is becoming the future of agricultural production. By utilizing intelligent systems, it is possible to enhance the quality of fields, detect pests and diseases earlier, and even achieve autonomous harvesting of ripe fruits. Given its potential for technological advancement, precision agriculture is an ideal field for knowledge transfer from other domains.

The primary objective of this work was to design and develop software architecture along with several modules for a specific use case of precision agriculture, the autonomous harvesting of organic broccoli. Through simulation and field experiments conducted under similar conditions, the developed algorithms were tested and evaluated for their main features. Future directions for research could include: exploring new techniques for turning, integrating vision into the navigation system through vSLAM-like algorithms, and investigating new sensors for detecting furrows in the field.

Índice

1 Introducción

Yo es que he pensado que a mí también me gustaría ser intelectual, como no tengo nada que perder.

AMANECE QUE NO ES POCO, 1989

La agricultura ha representado siempre un pilar indiscutible en el desarrollo económico de una nación. La revolución neolítica tuvo lugar hace más de 9000 años, momento en el que la humanidad sufre un cambio radical y evoluciona de un comportamiento nómada a sedentaria productora con el inicio de la agricultura y la ganadería. Gracias a la tecnología y a la experiencia hemos mejorado nuestros procesos y aún tenemos un largo camino lleno de retos e hitos que superar.
Como se puede observar en la Figura 1.1, el aporte de la agricultura al PIB español representa, aproximadamente, un 4 % del total. Acorde al mapa provincial, Figura 1.2, esta cifra puede ascender hasta más del 10 % del aporte de algunas comunidades autónomas [3]. En los últimos años se puede observar un descenso de la producción nacional; esto puede ser atribuido a destinar los terrenos a procesos productivos más lucrativos como la instalación de sistemas de generación eléctrica sostenible o a la incapacidad de generar beneficios bajo las condiciones competitivas del sector. Para poder mejorar la capacidad competitiva de este y otros sectores una solución posible es la evolución tecnológica de los procesos productivos en pos de una mejora en la eficiencia.

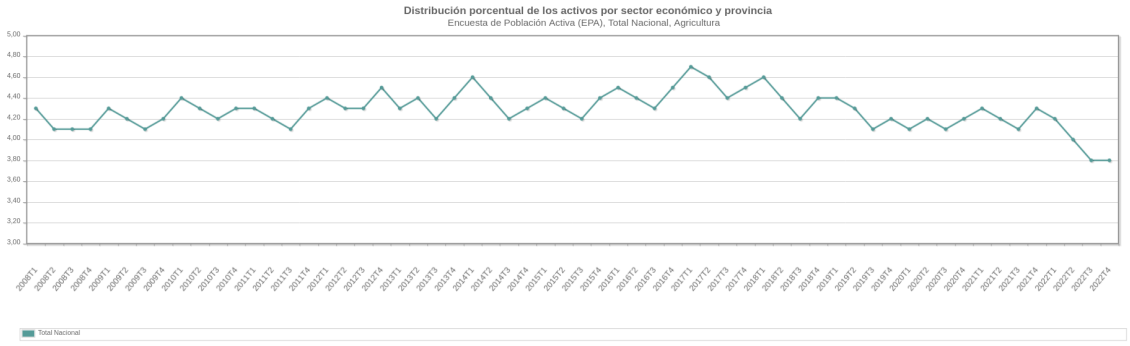

Figura 1.1 Evolución temporal de la representación porcentual de la agricultura en el PIB de España [3].

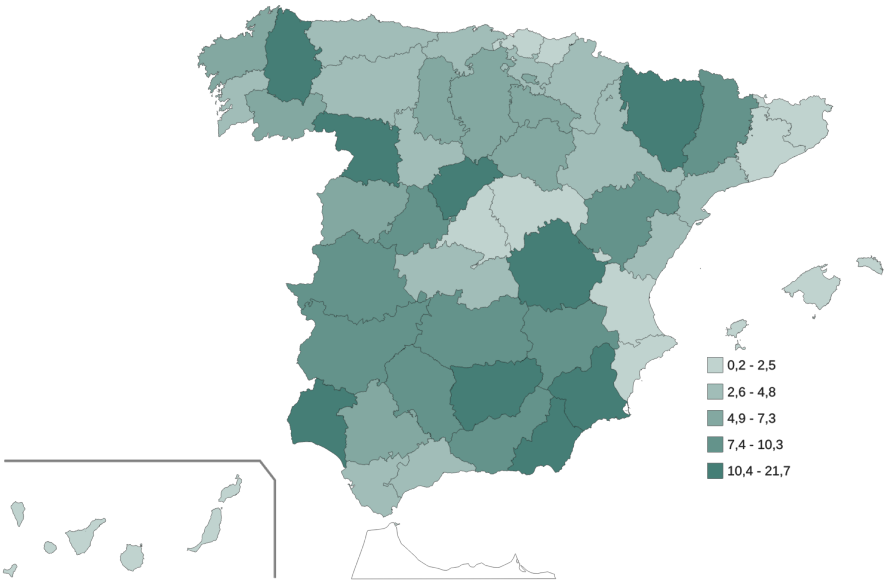

Figura 1.2 Mapa de color con la representación porcentual de la agricultura en el PIB provincial [3].

En concreto, dentro del sector agrícola se encuentran las plantaciones de olivos, un producto patrio que ha sufrido una gran evolución en su recolecta gracias a la industrialización. Tradicionalmente, la cosecha de estos se ha realizado a mano por los agricultores mediante la técnica del vareado, pero debido a la evolución de la producción extensiva emergió un nuevo mercado de maquinaria adaptada para la recolecta de olivas mediante técnicas novedosas. Un ejemplo de ello es la cosechadora industrial Bobcat T770, Figura 1.3, equipada con paraguas con pinza vibradora, capaz de cosechar una media de 30 olivos por hora, siendo esta una cifra equivalente al trabajo realizado por unos 15 recolectores [2].

Figura 1.3 Imagen de archivo de la cosechadora Bobcat T770 realizando tareas de cosechado de olivos [2].

1.1 Trabajo realizado

Este trabajo de fin de máster plantea una arquitectura y desarrollo software para robots terrestres autónomos especializados en tareas agrícolas, en concreto en la recolección de brócoli ecológico. De esta forma se

busca mejorar los procesos productivos y una aportación a la evolución tecnológica de un sector básico como es la agricultura. Aunque la arquitectura desarrollada se aplica al caso de uso de recolección, esta ha sido concebida para poder ser aplicada a distintos casos de uso.

1.2 Proyecto asociado

Este trabajo de fin de máster ha sido desarrollado gracias a ELISA (1.4), un proyecto liderado por la empresa AND&OR junto con la colaboración de la Universidad de Sevilla para la concepción de un robot terrestre autónomo desarrollado y fabricado en territorio nacional para el cosechado de brócoli ecológico. Las capacidades de este robot incluyen la teleoperación, la detección y cosechado automático del brócoli, así como la descarga de la cosecha en pallots.

Figura 1.4 Imagen de la cosechadora autónoma ELISA.

1.3 Estructura del documento

La estructura de la memoria se divide en los siguientes apartados: primero se ha presentado una introducción del trabajo realizado, seguido por un segundo capítulo centrado en el estado del arte; el tercer capítulo aborda la arquitectura software planteada, mientras que los capítulos cuarto y quinto tratan sobre los dos módulos desarrollados para dicha arquitectura. Para finalizar, un capítulo con los experimentos realizados y otro con las conclusiones obtenidas del trabajo realizado, así como líneas de trabajo futuras.

2 Estado del arte

- ¿Es un niño o una niña?
- Creo que es un poco pronto para empezar a imponerle roles, ¿no cree?

El sentido de la vida, 1983

En este capítulo se expone una breve introducción al desarrollo de arquitectura software moderna y al desarrollo de software orientado a robótica, así como el estado del arte en la robótica autónoma para entornos agrícolas.

2.1 Desarrollo de arquitectura software

Para poder hablar sobre el estado del arte en arquitectura software primero es conveniente dar una definición concisa de este concepto, lo cual torna a ser complicado. Como toda la tecnología relacionada con el mundo de la computación, esta ha sufrido una evolución nunca vista en otros campos de conocimiento. Además, se trata de una ciencia creada por el ser humano, por lo que encontrar una definición que sea atemporal puede ser imposible. Una definición de la arquitectura software es simplemente una sucesión de decisiones desde el más alto al más bajo nivel, buscando, como objetivo principal, minimizar los recursos humanos necesarios para desarrollar y mantener el sistema requerido [12]. Adoptando dicha definición es fácilmente reconocible cuando un sistema está bien diseñado y cuando no. Si las tareas y esfuerzos necesarios aumentan en cada actualización del sistema estamos ante un sistema posiblemente mal diseñado, en cambio, si la carga de trabajo se mantiene relativamente baja y constante a lo largo de la vida útil del sistema estaríamos hablando de un sistema bien diseñado.
Un concepto importante a tener cuenta durante el proceso de diseño de una arquitectura software es su razón de ser. Se define el diseño de una arquitectura para poder mejorar su entendimiento y paliar las limitaciones humanas al desarrollar problemas con muchas variables y alta complejidad. Una estructura demasiado modular o compleja no produce un mejor software, sino todo lo contrario a lo deseado. Este caso es un claro ejemplo de sobreingeniería que provoca una disminución de la eficiencia del software y se debe evitar. Algunas claves que pueden usarse para detectar cuando algo va mal en el diseño de una arquitectura pueden ser las siguientes:

- Rigidez - Es complejo realizar cualquier cambio en el sistema, cada uno de los cambios fuerza cambios en partes distintas del software.
- Fragilidad - Cambios provocan que el sistema deje de funcionar por errores en funcionalidades donde no existe una relación conceptual con los cambios realizados.
- Inmovilidad - Es complicado seccionar el sistema en componentes reutilizables en otros sistemas.
- Viscosidad - Realizar cambios de la manera correcta es más complicado que realizarlos mal.
- Complejidad innecesaria - El sistema posee estructuras demasiado complejas que no aportan ninguna funcionalidad directa.

- Repetición innecesaria - El diseño contiene estructuras semejantes que pueden ser abstraídas.
- Opacidad - El código es difícil de leer o entender su funcionalidad.

Un ejemplo de metodología establecida en la industria es el diseño ágil, un proceso de continua aplicación de los principios, patrones y prácticas para mejorar la estructura y la claridad del software. Originalmente, esta metodología está asociada al paradigma de programación orientada a objetos, pero muchos de sus principios pueden extrapolarse a otros paradigmas [4].

2.1.1 Paradigmas de programación

Un paradigma de programación puede definirse como un estilo de desarrollo, un enfoque sistemático a un problema y la manera de resolverlo. Existen multitud de paradigmas en programación, cada uno de ellos con sus ventajas e inconvenientes, como síntesis de estos se presentan dos de los enfoques más conocidos y utilizados en la actualidad.

Programación orienta a objetos

Dicho paradigma fue descubierto en el año 1966 por Ole Johan Dahl y Kristen Nygaard. Este paradigma está basado en el concepto de objetos que contienen información, generalmente nombrado atributos, y procedimientos, generalmente llamados métodos. Algunas características principales que se le atribuyen a este paradigma son las siguientes:

- Encapsulación - Mediante el constructor de objeto es fácilmente delineable la agrupación de información y procedimientos que tengan cohesión entre sí. Visto desde un punto de vista externo podemos ocultar la información en el interior de un objeto y mostrar únicamente algunas funcionalidades.
  Este concepto de encapsulación puede contrargumentarse debido a que, al usar una función, el usuario debe saber bajo qué condiciones otorgará un resultado u otro y debido a la encapsulación a su vez debe ser conocedor de todos los atributos del objeto. En caso de que el usuario no sepa de la existencia de todos los atributos el resultado puede llegar a ser errático.
- Herencia - La herencia es el mecanismo por el cual se puede definir un objeto que deriva de otro, compartiendo así los atributos y métodos. Representa la base sobre lo que se sustenta la abstracción en programación orientada a objetos.
- Polimorfismo - Mediante el polimorfismo tenemos la capacidad de modificar el comportamiento de métodos mediante la sobrecarga de funciones o sobrescribiéndolos. Junto con la herencia permiten definir métodos genéricos en capas altas de abstracción y definir un comportamiento sobre objetos que hereden dicho objeto abstracto.

Programación funcional

La programación funcional es un paradigma definido en 1977 por John Backus [5]. Este paradigma está basado en el intento de resolver todo mediante un enfoque basado en funciones matemáticas, donde dadas unas entradas a la función el resultado no variará independientemente del momento en el que sea llamada dicha función. Algunas características principales que se le atribuyen a este paradigma son las siguientes:

- Funciones puras - Las funciones siempre devolverán el mismo resultado bajo los mismos argumentos de entrada. Todas las funciones deben de ser deterministas y no depender de variables globales. Esto permite resolver cálculos en tiempo de compilación o paralelizar instrucciones.
- Recursividad - Debido al enfoque mediante funciones deterministas se evitará el uso de bucles en el flujo del programa, se optará por la llamada recursiva de funciones.
- Transparencia - Las variables, una vez definidas, no tendrán permitido cambiar el valor que contienen. La asignación está desaconsejada en la programación funcional, para almacenar nuevos valores se debe crear una nueva variable. De esta forma se afianza el concepto determinista de este paradigma.

Arquitectura orientada a robótica

En concreto, si hablamos de arquitectura software aplicada a robótica podemos establecer una serie de objetivos específicos [9]:

- Independencia de la plataforma - La arquitectura software no debe depender de una plataforma hardware en particular. Esto ayuda a poder ejecutar dicho software en distintas plataformas de testeo o bajo simulación en ordenadores personales con un cambio mínimo en el sistema.

- Modularidad hardware - La arquitectura software debe definir reglas y distintos framework de trabajo entre los distintos módulos hardware, pero dichas comunicaciones no deben depender del elemento hardware del módulo si no de la abstracción de su funcionalidad.
- Productividad estructural - La arquitectura software debe tener bien definido el proceso para añadir nuevas funcionalidades.
- Mantenibilidad del código - La estructura software debe favorecer la mejora y mantenimiento del código desarrollado.
- Testeable - La estructura software ha de contar con una interfaz de uso para el usuario final, y de esta forma favorecer los test unitarios a cada módulo.
- Diagnóstico ágil - Detectar errores y fallos funcionales en una estructura modular es una tarea que resulta relativamente sencilla. Una estructura software debería incluir métodos de diagnóstico y depuración de los distintos módulos funcionales.
- Adaptable para desarrollo en equipo - Cada desarrollador o tester deberá tener la capacidad de trabajar en módulos de forma independiente. Esto se consigue mediante una correcta definición de las interfaces de comunicación durante la definición de la arquitectura.

En la actualidad el estándar en comunicación para software orientado a robots está basado en ROS (Robotic operation system) [16] y su evolución ROS 2 [11]. ROS es un sistema operativo de código abierto que proporciona un entorno estructurado y robusto para el desarrollo de software para robots. Se puede definir como una colección de herramientas, librerías y convenciones que facilitan el desarrollo de software enfocado a la robótica.

2.2 Robótica en la agricultura

Actualmente, existe un término atribuido al concepto de robótica para agricultura llamado agricultura de precisión. La agricultura de precisión se puede definir como el tipo de agricultura que se centra en incrementar el número de decisiones por unidad de terreno y unidad temporal asociado a un incremento del beneficio económico [13]. Otra definición menos abstracta sería una estrategia de gestión que utiliza elementos electrónicos y otras tecnologías para recoger, procesar y analizar información espacio-temporal con el objetivo de guiar las acciones que mejoren la eficiencia, productividad y sostenibilidad de las operaciones agrarias [10].
Según la funcionalidad del robot podemos determinar los siguientes tipos [14]:

2.2.1 Preparación del terreno para la plantación

La preparación del terreno es la primera tarea en la agricultura. Mediante este proceso se eliminan malas hierbas, se oxigena el terreno, se aplican los primeros fertilizantes y se prepara para el sembrado del cultivo. Esto implica la necesidad de diseñar robots robustos para poder realizar la actividad de forma eficiente.
Un ejemplo podemos encontrarlo en el robot para fertilizar terreno AgBot, un robot equipado con tracción a dos ruedas capaces de portar distintos productos químicos para la fertilización de terreno. Su sistema de navegación lleva equipado un sistema GNSS-RTK, una cámara RGB así como una IMU. [8].

2.2.2 Plantación

Tradicionalmente, la tarea de plantado se ha realizado utilizando maquinaria especializada acoplada a un tractor. Sin embargo, mediante el uso de la robótica se plantea el uso de maquinaria más precisa, ágil y de menor peso, reduciendo así el coste, el combustible consumido y minimizando el daño al terreno durante la plantación.
Un ejemplo podemos encontrarlo en la Figura 2.1, el robot Di-Wheel, desarrollado para la plantación de distintos tipos de semillas, equipado con un sistema de tracción a dos ruedas, reduciendo el tamaño, peso y complejidad mecánica, además de tener un tiempo estimado de despliegue de únicamente 15 minutos. El robot cuenta con la capacidad de variar la distancia entre sus ruedas para poder adaptarse a distintas plantaciones. Para reducir costes en su despliegue cuenta con la capacidad de fijar un teléfono móvil y utilizar todos los sensores de este como: linterna, cámaras RGB, giróscopos, acelerómetros o GNSS, entre otros [17].

Figura 2.1 Imagen del robot Di-Wheel realizando tareas de plantado.

2.2.3 Cuidado de la plantación

La prevención de plagas y enfermedades es un elemento clave para la agricultura extensiva, esto implica un esfuerzo inabarcable por parte de los agricultores en plantaciones de grandes dimensiones. Gracias a la robótica se brinda la oportunidad de monitorizar la plantación y realizar una detección temprana de posibles plagas y enfermedades, lo cual proporciona la capacidad de actuar y minimizar los daños sufridos.
Un ejemplo podemos encontrarlo en el robot eAGROBOT, un robot terrestre diseñado para realizar tareas de vigilancia autónoma de plantaciones. Equipado con una cámara RGB y mediante técnicas de inteligencia artificial basada en redes neuronales artificiales y clasificadores K-measn tiene la capacidad de detectar enfermedades y rociar con pesticidas de forma autónoma [15].

2.2.4 Cosechado

Por último, quedaría hablar del cosechado de la plantación. Dicha tarea es difícilmente generalizable a distintos casos debido a la disparidad de formas de recolección que existen dependiendo de la planta a cosechar, es por ello por lo que, para cada tipo de plantación se presentan un caso de uso distinto y, por lo tanto, un diseño hardware distinto. Debido a la baja automatización de esta tarea representa más de la mitad del coste total de producción, además de suponer un trabajo físico muy alto para quienes lo realizan.
Un ejemplo de automatización en tareas de cosechado lo podemos encontrar en Agrobot E-Series, Figura 2.2, un robot con hasta 24 brazos cartesianos independientes diseñados para la cosecha autónoma de fresas en invernadero. Equipado con cámaras e inteligencia artificial para la detección de la maduración de las fresas, así como un sistema LiDAR para evitar la colisión con trabajadores [1].

Figura 2.2 Imagen del robot Agrobot E-Series [1].

3 Arquitectura del sistema e implementación

K.I.S.S.

~~Kelly Johnson, 1960~~ Fran Real, 2022

Tras exponer en los capítulos anteriores aspectos generales del trabajo y el estado del arte actual en el diseño de arquitecturas software para robótica y la agricultura de precisión, se continúa con la arquitectura propuesta e implementación realizada. Para ello, este capítulo se dividirá en las siguientes secciones: primero, un planteamiento general de la arquitectura, tras el cual, se detallarán las herramientas utilizadas y la justificación de su uso.

3.1 Arquitectura software

Para poder decidir el contenido de la arquitectura es necesario primero saber qué se está solicitando al sistema y qué elementos tiene, tanto de entrada como de salida. Enumerando las funcionalidades requeridas en el caso de estudio propuesto tenemos las siguientes:

1. **Seguimiento de líneas**: Seguimiento de los surcos en el suelo, modo principal para la recolección autónoma de brócoli.
2. **Giro**: Realización de giro para distintas funcionalidades. Algunas de ellas pueden ser cambiar de calle o colocar la máquina para realizar el acercamiento a los pallots.
3. **Acercamiento a pallot**: Maniobra autónoma para colocar la máquina sobre los pallots.
4. **GOTO**: Funcionalidad que permite a la máquina dirigirse en línea recta a un punto objetivo definido por su latitud y longitud.

Por otro lado, tenemos las fuentes de información del sistema:

- Receptor GNSS, configurado para proporcionar medidas integrando correcciones NTRIP, que le ayuda a posicionarse en el espacio respecto a unos ejes de referencia globales.
- Una estimación de la velocidad a la que avanzan cada una de las orugas, también llamada odometría, que con la ayuda de un modelo cinemático del robot permitirá afinar la estimación de la posición y orientación del robot.
- LIDAR 3D enfocado hacia delante para la detección de las líneas a seguir para la recolección.
- LIDAR 2D enfocado hacia atrás para la detección del pallot sobre el que descargar lo cosechado.

Siguiendo lo aprendido en el estudio del estado del arte referente al desarrollo de arquitecturas software se presenta un esquema distribuido en capas para el caso de uso que se presenta. Como se puede observar en la Figura 3.1, el sistema completo consta de una gran variedad de aplicaciones independientes; cada una de ellas ha sido desarrollada para realizar una tarea concreta, lo cual facilita la depuración de errores del sistema completo.

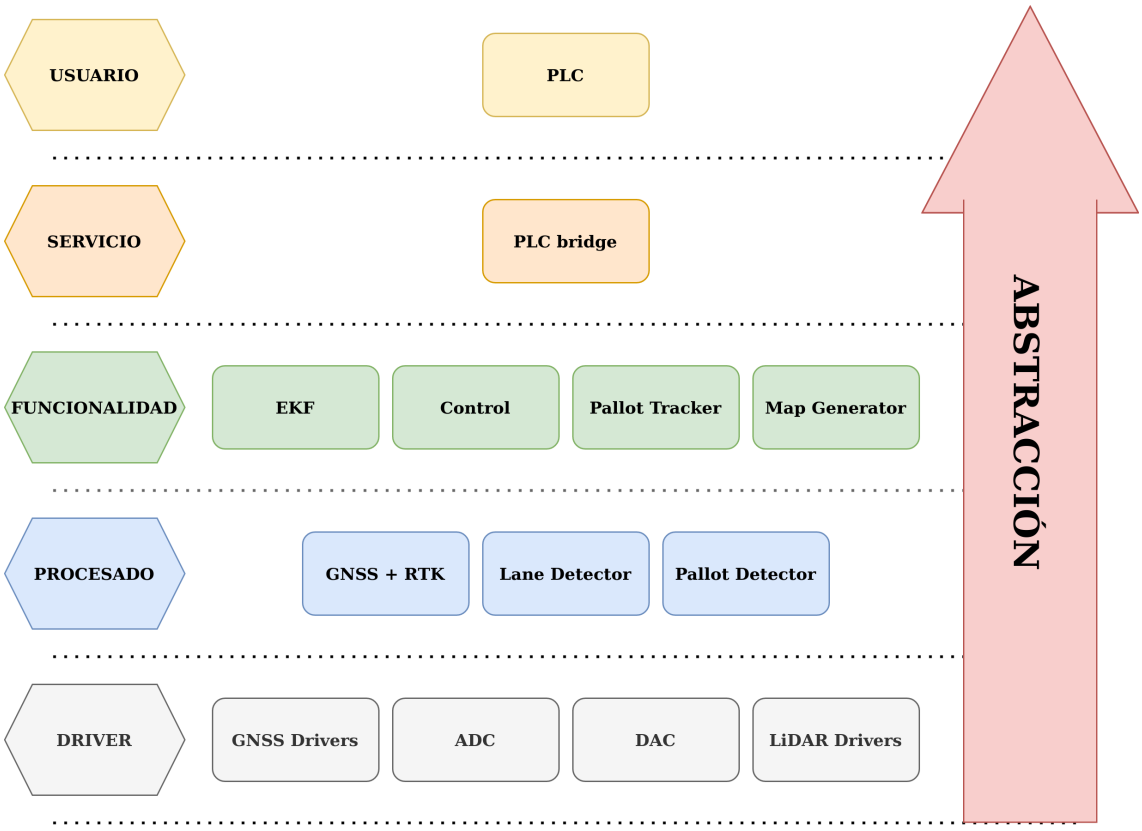

Figura 3.1 Esquema de la arquitectura implementada.

3.1.1 Capa driver

Representa la capa de abstracción más baja, en ella se encuentran las aplicaciones dedicadas íntegramente a la comunicación de la información con los medios físicos. Podemos encontrar aplicaciones genéricas como convertidores analógicos digitales o más específicas como la comunicación con el sistema GNSS o los drivers comerciales de los sistemas LiDAR equipados.

3.1.2 Capa de procesado

Esta capa representa el primer procesado a la información recibida por los sensores, adapta la información obtenida y le da un contexto de uso. Por ejemplo, en los casos de la información LiDAR, transforma dicha información de un sistema genérico a algo concreto como un pallot o un surco en el suelo, abstrayendo al sistema de elementos como puntos en el espacio.

3.1.3 Capa de funcionalidad

En esta sección se encuentra el valor añadido del software, puede verse como un paralelismo al "business layer" de una arquitectura clásica de software. Mediante la información procesada se plantean algoritmos para resolver el problema para el que está ideado el sistema.

3.1.4 Capa de servicio

Mediante este elemento de la arquitectura se crea un elemento que brinda una serie de funcionalidades al usuario final. En ámbitos más generalistas del software encontraríamos una API a la que realizar solicitudes. Dado el contexto industrial del proyecto se trata de un módulo de comunicación con el usuario final, el PLC.

3.1.5 Capa de usuario

Por último, en la capa más alta de abstracción tenemos al usuario. En esta sección se brinda al usuario final una forma de utilizar y depurar cada una de las funcionalidades desarrolladas. Típicamente, podemos encontrar sistemas HMI o GUI.

3.2 Medios utilizados

En esta sección se detallarán las herramientas utilizadas para llevar a cabo el trabajo, así como una justificación del uso de estas. Se ha tratado de usar, siempre que ha sido posible, herramientas de software libre.

3.2.1 Lenguajes utilizados

Para el desarrollo del proyecto se han utilizado distintos lenguajes de programación, dependiendo de las necesidades de cada tarea:

1. **C++** representa un estándar en el desarrollo software donde la eficiencia y la robustez son elementos críticos. Para el desarrollo de los módulos de este proyecto se ha utilizado dicho lenguaje.
2. **Python** ha sido utilizado como lenguaje auxiliar para desarrollar aplicaciones test y de interfaz para realizar simulaciones del sistema completo. Gracias a la facilidad y flexibilidad del lenguaje es idóneo para este tipo de tareas que no requieren un alto coste computacional.

3.2.2 Programas utilizados

Los programas utilizados durante el desarrollo del proyecto han sido los siguientes:

1. **ROS** es un sistema operativo robótico de código abierto que proporciona un entorno estructurado y robusto para el desarrollo de software para robots. Se puede definir como una colección de herramientas, librerías y convenciones que facilitan el desarrollo de software enfocado a la robótica [16].
2. **Git** es un software referente en las aplicaciones para control de versiones software.

4 Módulo de control

¡Mira detrás de ti! ¡Un mono con tres cabezas!.

GUYBRUSH THREEPWOOD, 1990

El módulo de control tiene como objetivo determinar las consignas de velocidad que serán transmitidas, a través de la comunicación con el PLC, a los controladores de las orugas. Dichas consignas serán determinadas con el objetivo de navegar a través de la plantación, siguiendo los surcos detectados en el suelo o aproximándose a un pallot para la descarga de lo cosechado, mientras se evita dañar la plantación.

Se ha implementado un modelo de control Pure Pursuit. Como se puede observar en Figura 4.1, dicho algoritmo de seguimiento de rutas calcula un punto objetivo, look ahead, dentro de la recta de seguimiento que asegura un establecimiento sobre la misma. Se ha decidido utilizar dicho algoritmo debido a su bajo coste computacional y a su robustez, además de la fácil adaptación al caso de uso del proyecto.

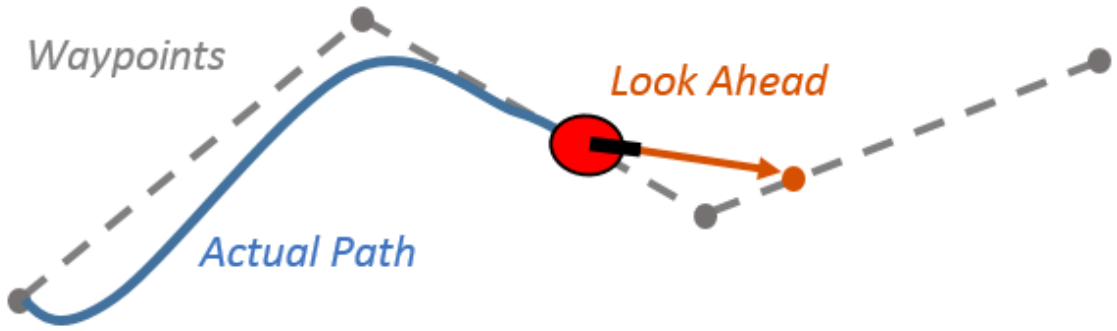

Figura 4.1 Ejemplo de un seguimiento de rectas mediante el algoritmo Pure Pursuit .

Las entradas del módulo de control son: el estado del vehículo determinado por el módulo de estimación de posición, las consignas establecidas por el PLC, las rectas detectadas por el detector de líneas y los pallots detectados por el detector de pallots. Las salidas del módulo de control serán las consignas en velocidad para las orugas y las señales de comunicación establecidas con el PLC para la transición de estados.

La recta de seguimiento queda definida de distinta forma según la tarea que se esté desempeñando:

1. Seguimiento de líneas: Se define la recta de seguimiento como la recta paralela a los dos surcos detectados más cercanos a cada lado del robot y situada, teóricamente, sobre la línea de brócoli a recolectar.
2. Giro: El proceso de giro se gestiona mediante una maniobra predefinida, por lo que no usará este método de control.

3. Acercamiento a pallot: Se define la recta de seguimiento como la recta perpendicular al segmento detectado por el sistema de detección de pallots.

4. GOTO: Se define la recta de seguimiento como la recta definida por el punto objetivo y la posición del robot.

La implementación del método puede ser resumida siguiendo los siguientes pasos:

Primero se calcula el error lateral a la recta de seguimiento mediante las siguientes ecuaciones:

$$Ax + By + C = 0 \tag{4.1}$$

$$e_l = \left| \frac{Ax_p + By_p + C}{\sqrt{A^2 + B^2}} \right| \tag{4.2}$$

Siendo A, B, C los parámetros de la ecuación general de la recta de seguimiento y x_p, y_p la posición del robot. Mediante el error lateral se calcula tanto la velocidad de avance del robot como el desplazamiento del look ahead sobre la recta, para ello se han definido las siguientes funciones logísticas:

$$desplazamiento := d = d_{min} + \frac{d_{max} - d_{min}}{1 + e^{(18*(e_l - 0.25))}} \tag{4.3}$$

$$avance := u = u_{min} + \frac{u_{max} - u_{min}}{1 + e^{(10*(e_l - 0.25))}} \tag{4.4}$$

Siendo d_{min} y d_{max} dos parámetros de ajuste y u_{max} la consigna de avance en el PLC y u_{min} un 10 % de esta. Evaluadas para la configuración por defecto del controlador y unos errores laterales posibles en el caso de uso se muestran en la Figura 4.2 y 4.3.

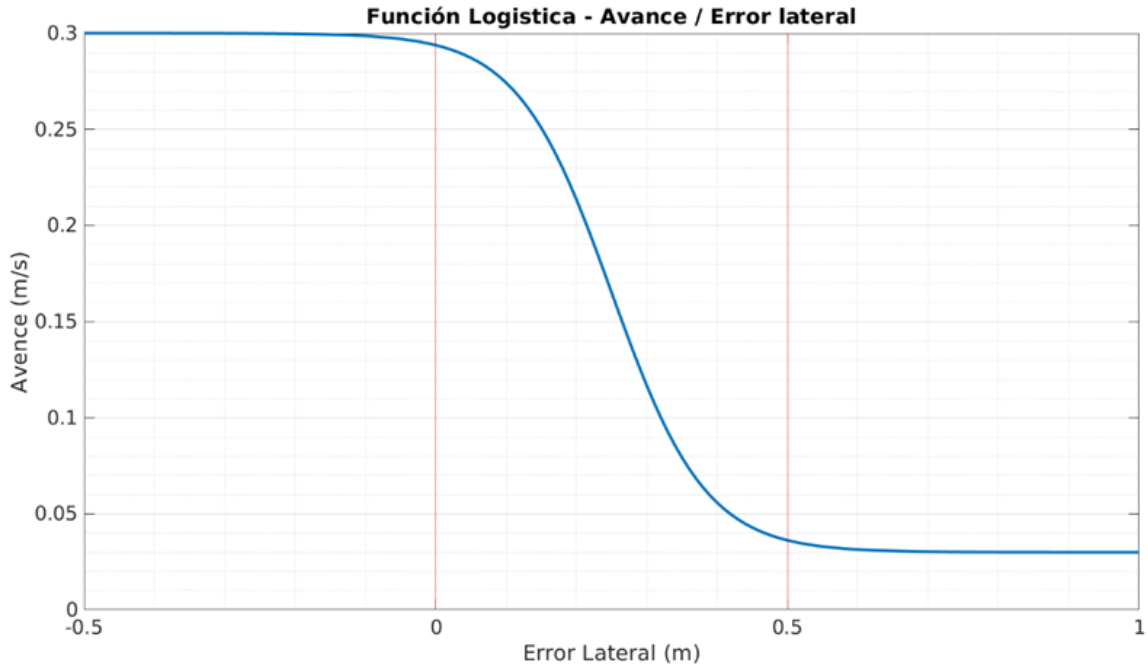

Figura 4.2 Evaluación de la función logística para el caso de uso, estableciendo la consigna de avance en 30 centímetros por segundo.

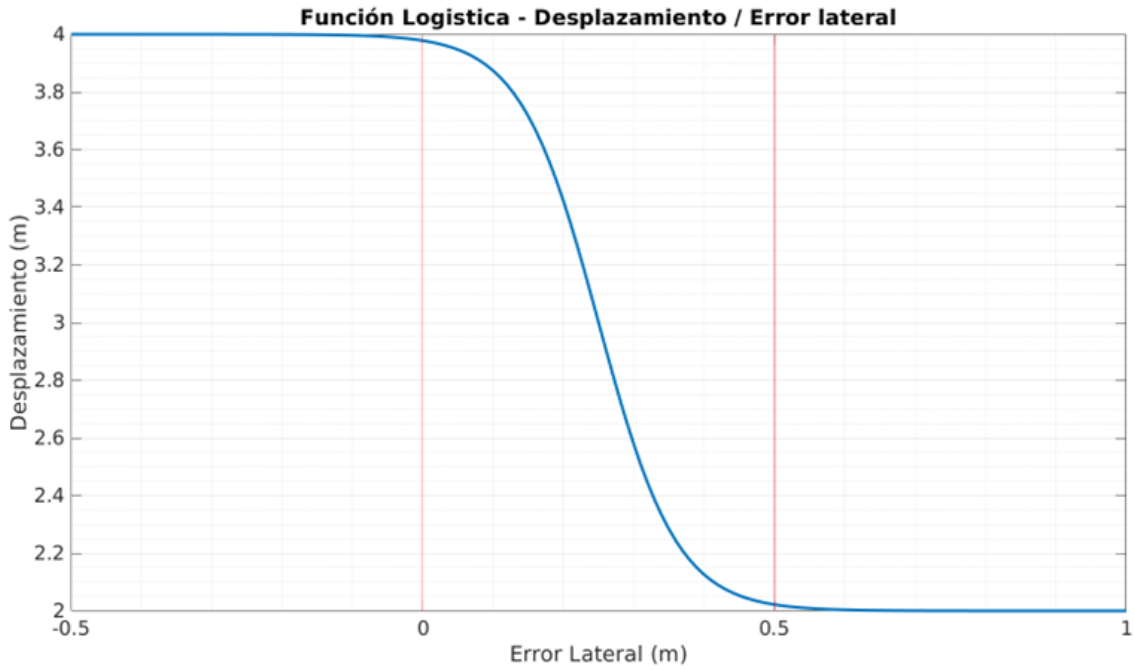

Figura 4.3 Evaluación de la función logística para el caso de uso, estableciendo el desplazamiento máximo en 4 metros y el mínimo en 2 metros.

Una vez determinado el avance y el desplazamiento del look ahead solo quedaría resolver las ecuaciones cinemáticas del robot para determinar los valores de consiga:

$$\begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 1/D & -1/D \end{bmatrix} \begin{bmatrix} v_r \\ v_l \end{bmatrix} \tag{4.5}$$

$$w = \tan^{-1}\left(\frac{y_g}{x_g}\right) \tag{4.6}$$

$$u \in [u_{min}, u_{max}] \tag{4.7}$$

$$w \in [w_{min}, w_{max}] \tag{4.8}$$

Siendo D la distancia entre ambas orugas motoras del robot, v_r, v_l definen las consignas de velocidad a las orugas y w define la velocidad angular del robot. Dichas consignas de velocidad están limitadas, pudiendo establecer los límites modificando los parámetros de configuración.

5 Módulo de navegación

¡Mira detrás de ti! ¡Un mono con tres cabezas!.

GUYBRUSH THREEPWOOD, 1990

El módulo de estimación de posición tiene como objetivo determinar la posición y orientación del robot dentro de la plantación a recolectar. Dicha información es necesaria para el módulo de control, en concreto para realizar las acciones de giros entre surcos y asegurar la navegación a través de los surcos durante el proceso de recolección.
Para dicha tarea se ha implementado el algoritmo EKF (Extended Kalman Filter). EKF es un algoritmo genérico para la estimación de variables a través de dos etapas: predicción y corrección [7]. Las entradas de este módulo son: posición GNSS y la odometría de las orugas.

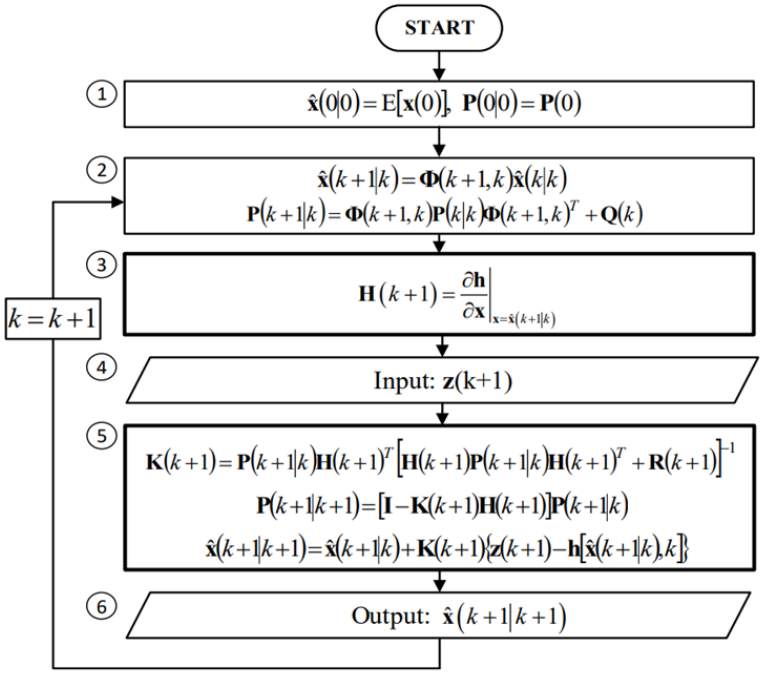

Figura 5.1 Flujo de trabajo de un EKF [7].

Las salidas del módulo será el siguiente vector de estado:

$$x = \begin{pmatrix} x_p \\ y_p \\ \theta \\ u \\ \omega \end{pmatrix} \quad \begin{pmatrix} x_p := posición\ eje\ x \\ y_p := posición\ eje\ y \\ \theta := orientación \\ u := velocidad \\ \omega := velocidad\ angular \end{pmatrix} \tag{5.1}$$

El vector de medidas queda definido por:

$$z = \begin{pmatrix} x_m \\ y_m \\ u \\ \omega \end{pmatrix} \quad \begin{pmatrix} x_m := posición\ eje\ x \\ y_m := posición\ eje\ y \\ u := velocidad \\ \omega := velocidad\ angular \end{pmatrix} \tag{5.2}$$

Los sensores quedan definidos a través de la matriz de observación H y la matriz de varianza asociada R, siendo las del GNSS y la edometría las siguientes:

$$H_{GNSS} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad R_{GNSS} = \begin{pmatrix} 0.05 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{5.3}$$

$$H_{odom} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad R_{odom} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix} \tag{5.4}$$

La matriz de transición de estados Φ y su matriz de varianza asociada Q definen la etapa de estimación del filtro:

$$\Phi = \begin{pmatrix} 1 & 0 & 0 & \cos(\theta) & -\sin(\theta) * 1.836 \\ 0 & 1 & 0 & \sin(\theta) & \cos(\theta) * 1.836 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{5.5}$$

$$Q = \begin{pmatrix} 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0.01 \end{pmatrix} \tag{5.6}$$

Para terminar, solo restaría definir la matriz jacobiana F para tener completamente determinado el estimador:

$$F = \begin{pmatrix} 1 & 0 & -(sin(\theta) * u - \cos(\theta) * \omega * 1.836) * dt & \cos(\theta) * dt & -\sin(\theta) * 1.836 * dt \\ 0 & 1 & (\cos(\theta) * u - \sin(\theta) * \omega * 1.836) * dt & \sin(\theta) * dt & \cos(\theta) * 1.836 * dt \\ 0 & 0 & 1 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.7}$$

6 Experimentos

El Señor habló diciendo: primero sacaréis el santo sello, luego contaréis hasta tres, no más no menos. Tres es el número al que habréis de llegar y el número del conteo deberá ser tres. No contaréis hasta cuatro, ni tampoco hasta dos, salvo que preceda al tres. El cinco está descartado. Una vez que el tres sea alcanzado lanzaréis la Santa Granada de Mano de Antioquía hacia vuestro enemigo, quien al no ser agradable a mis ojos, se desintegrará.

Los caballeros de la mesa cuadrada, 1975

En este capítulo se expondrán los experimentos para la validación de los módulos desarrollados. Debido a la gran cantidad de experimentos realizados para la validación, únicamente se presentarán en este capítulo los últimos realizados en el campo. La estructura del capítulo será de una sección para cada experimento, donde se indicarán: descripción general del experimento, la motivación del experimento, condiciones, resultados y conclusiones extraídas.

6.1 Entornos de validación

Antes de realizar pruebas experimentales es importante haber realizado pruebas previas en entornos de simulación que avalen los módulos y minimicen la posibilidad de sufrir accidentes durante la experimentación. Para la validación de los módulos se han utilizado un entorno en simulación y otro real.

6.1.1 Simulación

Para las pruebas en simulación se ha recreado, mediante el software Gazebo, un entorno semejante a una plantación de brócoli real. Además del entorno, se ha desarrollado un módulo con el objetivo de recrear la información obtenida por los distintos sensores embarcados en el sistema.
Como se puede observar en la Figura 6.1, se trata de una plantación con varios linios de brócoli, así como un grupo de pallots para realizar la descarga de la cosecha realizada.

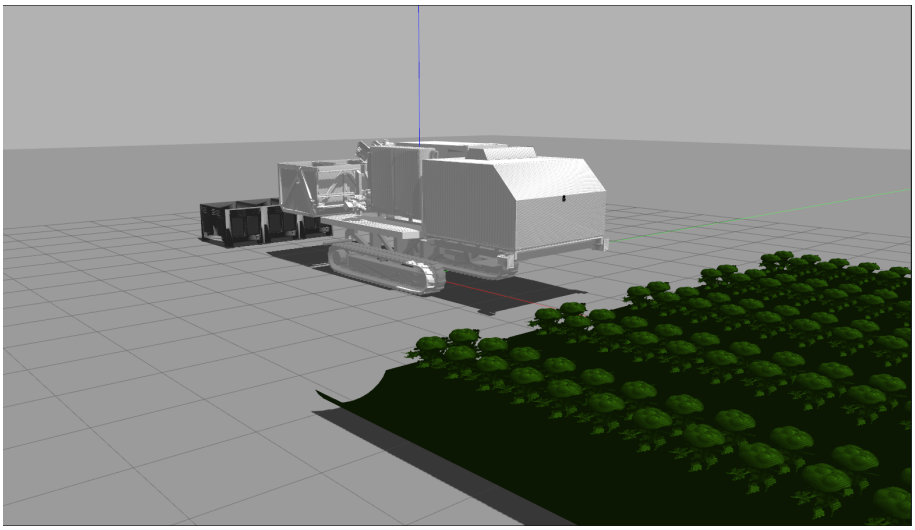

Figura 6.1 Captura del entorno de simulación desarrollado para el proyecto.

Entorno real

Las pruebas de experimentación se realizaron en un terreno propiedad de la empresa AND&OR llamado "La Merlina" (Figura 6.2) situado a la entrada del municipio de Coria del Río en la provincia de Sevilla. El entorno cuenta con el suficiente espacio para realizar multitud de pruebas sin el peligro de dañar ninguna plantación.

Figura 6.2 Foto tomada durante los experimentos en La Merlina.

6.2 Acercamiento a pallot

Una de las funcionalidades solicitadas al sistema es la capacidad autónoma de posicionarse sobre los pallots para realizar la descarga de las cubas cuando estas estén llenas de brócoli. En esta sección se detalla el experimento realizado para validar dicha funcionalidad.

6.2.1 Motivación

La motivación principal de este experimento es evaluar la capacidad del sistema para, dada una posición cercana a los pallots, calcular y realizar la maniobra necesaria para posicionarse sobre estos.

6.2.2 Condiciones

Se posiciona la máquina orientando la parte trasera hacia los pallots, ligeramente descentrada y desalineada de estos para comprobar la capacidad de maniobrar. La distancia a la que se sitúa la máquina no puede exceder los 10 metros debido a limitaciones en el sistema LiDAR 2D.

6.2.3 Resultados

Tras la realización del experimento bajo distintas condiciones se muestran los resultados en las Figuras 6.3, 6.4 y 6.5.

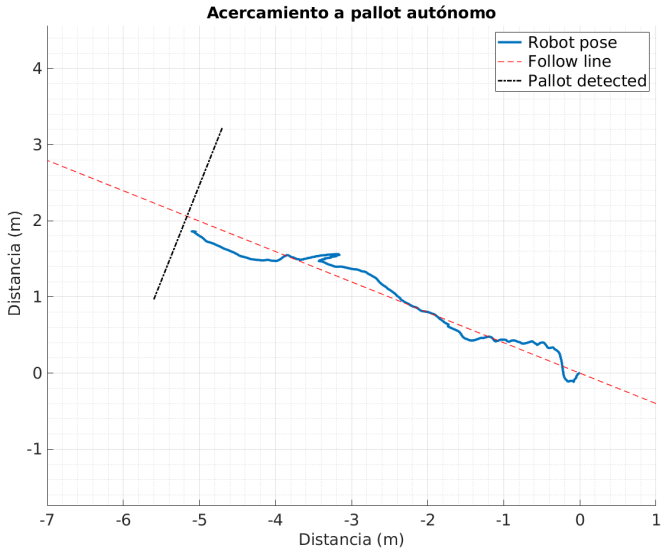

Figura 6.3 Gráfica de la trayectoria en coordenadas locales descrita en el experimento 1 para la validación del acercamiento a pallot.

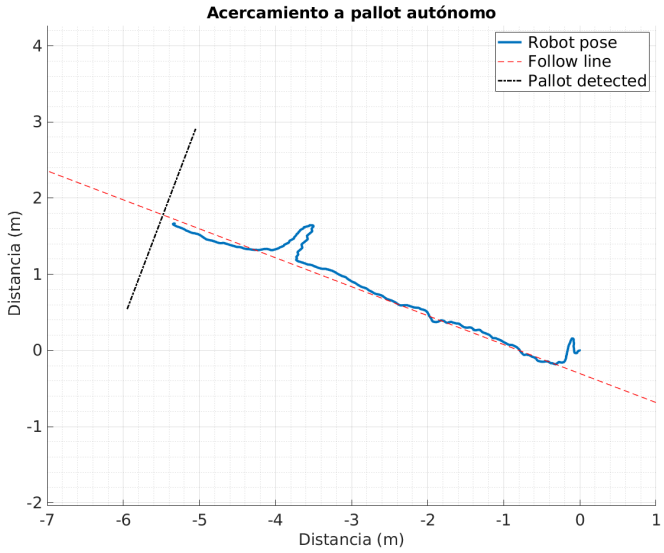

Figura 6.4 Gráfica de la trayectoria en coordenadas locales descrita en el experimento 2 para la validación del acercamiento a pallot.

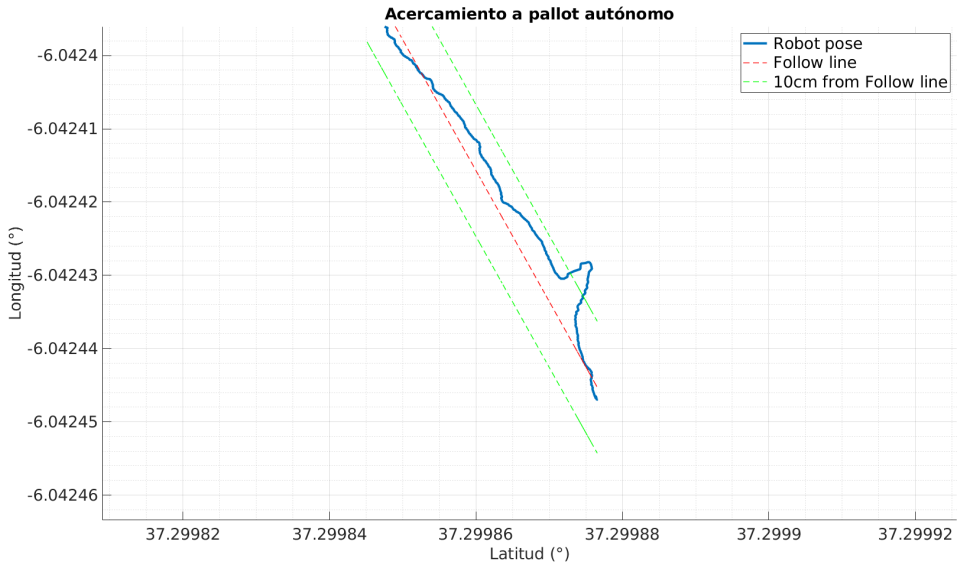

Figura 6.5 Gráfica de la trayectoria en coordenadas geodésicas descrita en el experimento 3 para la validación del acercamiento a pallot.

6.2.4 Conclusiones

Mediante mediciones en campo se ha observado una desviación de 10 centímetros respecto al centro de los pallots. Esto puede ser debido a diversos motivos como: irregularidades del terreno, errores en la detección del pallot o el rango de funcionamiento de las orugas. Se considera que el funcionamiento del módulo es correcto y los errores están dentro de tolerancias.

6.3 Seguimiento de surcos

La funcionalidad principal del sistema es su capacidad para conducir la máquina entre los surcos detectados. Es importante reducir al máximo el error lateral durante esta tarea debido a que una desviación demasiado grande provocaría que la máquina pisara la plantación.

6.3.1 Motivación

La motivación principal de este experimento es evaluar la capacidad del sistema para, dada una posición cercana a los surcos, calcular y realizar la maniobra necesaria para atravesar la plantación.

6.3.2 Condiciones

Se posiciona la máquina orientando la parte delantera hacia los surcos, ligeramente descentrada y desalineada de estos para comprobar la capacidad de maniobrar. La distancia a la que se sitúa la máquina no puede exceder los 6 metros debido a limitaciones en el sistema LiDAR 3D.

6.3.3 Resultados

Tras la realización del experimento bajo distintas condiciones se ejemplifica los resultados mediante las Figuras 6.6 y 6.7.

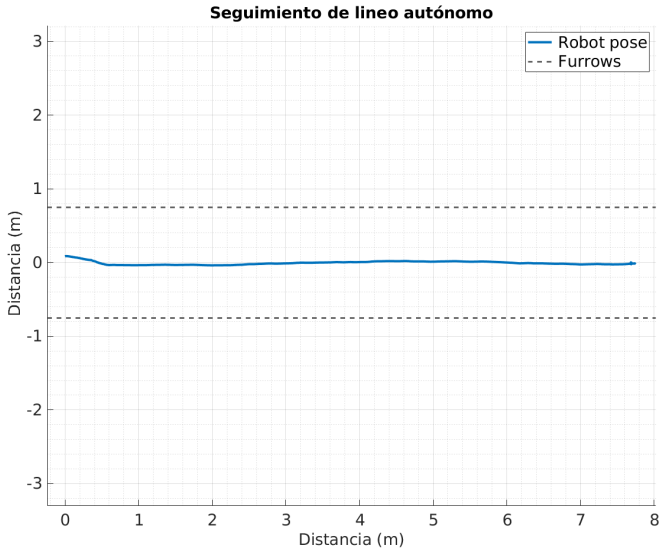

Figura 6.6 Gráfica de la trayectoria en coordenadas locales descrita en el experimento 1 para la validación del seguimiento de surcos.

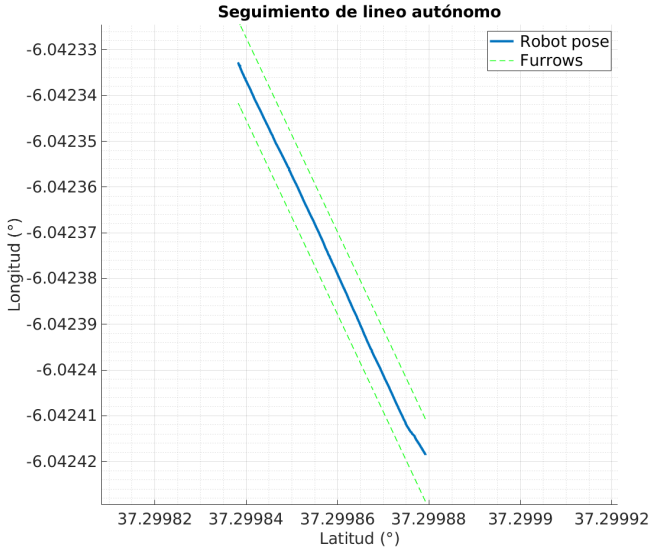

Figura 6.7 Gráfica de la trayectoria en coordenadas locales descrita en el experimento 2 para la validación del seguimiento de surcos.

6.3.4 Conclusiones

Debido a la falta de plantación con brócoli en el lugar donde se desarrollaron las pruebas, el sistema debió validarse únicamente con los surcos del terreno, siendo esta tarea más compleja que la real a realizar en una plantación. Se considera que el funcionamiento del módulo es correcto y los errores están dentro de tolerancias.

6.4 Giros

Por último, tenemos la funcionalidad del sistema para realizar giros en modo autónomo. Este modo permite al sistema continuar la cosecha tras finalizar una hilera completa de la plantación, así como realizar el

posicionamiento antes de activar el modo de acercamiento a pallot.

6.4.1 Motivación

La motivación principal de este experimento es evaluar la capacidad del sistema para realizar giros de forma autónoma dada una orientación deseada.

6.4.2 Condiciones

Se posiciona el robot con una orientación arbitraria y se solicita el giro deseado. El sistema de navegación debe converger antes de poder realizar acciones de giro autónomas.

6.4.3 Resultados

Tras la realización del experimento bajo distintas condiciones se ejemplifica el resultado de este mediante la Figura 6.8.

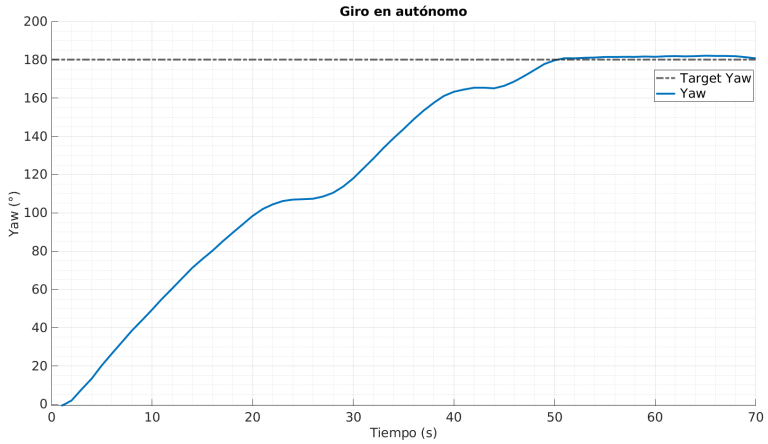

Figura 6.8 Gráfica de la evolución de la orientación respecto al tiempo en el experimento para la validación de los giros.

6.4.4 Conclusiones

Debido a limitaciones del sistema hidráulico de la máquina y la poca compactación del terreno, al realizar giros el sistema se entierra y debía realizar paradas para no apagar el sistema electrógeno que daba suministro. Se considera que el funcionamiento del módulo es correcto y los errores están dentro de tolerancias.

7 Conclusiones y trabajos futuros

Always look on the bright side of life.

La vida de Bryan, 1979

Como capítulo final de este trabajo de fin de máster se presentan las conclusiones recogidas a lo largo de su desarrollo y validación tras realizar los experimentos, así como un apartado final dedicado a posibles líneas de trabajos futuros.

7.1 Conclusiones

Tras evaluar los experimentos realizados, los siguientes hitos del proyecto han sido superados:

1. **Desarrollo de la arquitectura**: Se ha implementado una arquitectura correcta para el problema que se quiere resolver. Permitiendo alternar entre simulación y experimentación real de una forma cómoda y ágil. Además de permitir una fácil adaptación a casuísticas de la experimentación real no previstas en simulación.
2. **Módulo de navegación**: Se ha implementado un módulo de navegación que cumple su funcionamiento de forma correcta. Es importante destacar que gracias a la fusión sensorial se ha obtenido una estimación de la orientación, la cual permite realizar los giros y acciones GOTO.
3. **Módulo de control**: Se ha implementado un módulo de control que genera trayectorias coherentes bajo las condiciones de trabajo experimentadas. A falta de realizar pruebas en campo con plantaciones de brócoli real los experimentos realizados determinan que el sistema de control mantiene el error lateral durante el seguimiento de surcos dentro de las cotas de tolerancia, evitando así que se dañe la plantación.

7.2 Trabajos futuros

Debido a la limitación de tiempo, algunas ideas se han ido descartando o aplazando a lo largo del desarrollo del proyecto en pos de centrar los recursos en elementos más importantes. Tras el estudio del estado del arte de la agricultora de precisión y la experiencia ganada tras el desarrollo de este proyecto surgen las siguientes líneas de trabajo futuro:

1. **ROS 2**: Existen diversas razonas por la que la evolución de todas las aplicaciones desarrolladas en ROS deben de migrar a ROS 2, pero la más importante de ellas es la proximidad al EOL (End of life) de ROS. La última versión, Noetic, tiene fijado su descontinuación para el uno de mayo del 2025. La migración del software es un esfuerzo que todo mantenedor de software debe realizar en pos de una mejor compatibilidad del software.
   Cabe destacar que debido al planteamiento de la arquitectura propuesta la evolución de dicho sistema a múltiples robots se puede realizar con relativa simpleza y la migración a ROS 2 es uno de los primeros procesos a realizar.

2. **Extensión a otros cultivos**: Se plantea la posibilidad de, bajo la misma arquitectura, desarrollar otras aplicaciones para la detección de otras plantaciones y así ampliar la utilidad de la máquina.
3. **Mejora de la maniobrabilidad**: Durante la experimentación se ha detectado que el sistema hidráulico no tiene la capacidad motriz esperada. Se plantea la posibilidad de estudiar la implementación de otro sistema motriz al robot o la integración de maniobras de giro complejas que minimicen la cantidad de terreno desplazado.
4. **Integración de la visión en el módulo de navegación**: El sistema cuenta con una cámara cuyo objetivo es reconocer el estado del brócoli y posicionarlo en el espacio para su cosecha. Se plantea la implementación de odometría visual o algoritmos del tipo vSLAM para mejorar la capacidad del sistema de navegación.
5. **Estudio del sistema de detección**: El sistema de detección de surcos cuenta, actualmente, con un LiDAR 3D de 4 haces. Se plantea el estudio de sistemas de detección alternativos que proporcionen más información del entorno como los Livox Horizon.

Índice de Figuras

Bibliografía

[1] Agrobot e-series, *https://www.agrobot.com/*, Accessed: 2023-02-20.

[2] Bobcat_770, *https://www.bobcat.com/eu/es/company/news-and-media/job-stories/bobcat-a-benchmark-in-olive-harvesting*, Accessed: 2023-02-20.

[3] Instituto nacional de estadística, *https://www.ine.es/index.htm*, Accessed: 2023-02-20.

[4] Manifesto for agile software development, *https://agilemanifesto.org/*, Accessed: 2023-02-20.

[5] John Backus, Can programming be liberated from the von neumann style? a functional style and its algebra of programs, Commun. ACM **21** (1978), no. 8, 613–641.

[6] P.U. Chavan, M. Murugan, and P.P. Chavan, A review on software architecture styles with layered robotic software architecture, 2015 International Conference on Computing Communication Control and Automation, 2015, pp. 827–831.

[7] Simon J. Julier and Jeffrey K. Uhlmann, New extension of the Kalman filter to nonlinear systems, Signal Processing, Sensor Fusion, and Target Recognition VI (Ivan Kadar, ed.), vol. 3068, International Society for Optics and Photonics, SPIE, 1997, pp. 182 – 193.

[8] Nazmuzzaman Khan, Gregory Medlock, Scott Graves, and Sohel Anwar, Gps guided autonomous navigation of a small agricultural robot with automated fertilizing system, SAE Technical Papers **2018-April** (2018).

[9] David Kortenkamp, Reid Simmons, and Davide Brugali, Robotic systems architectures and programming, pp. 283–306, Springer International Publishing, Cham, 2016.

[10] James Lowenberg-DeBoer and Bruce Erickson, Setting the record straight on precision agriculture adoption, Agronomy Journal **111** (2019), no. 4, 1552–1569.

[11] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall, Robot operating system 2: Design, architecture, and uses in the wild, Science Robotics **7** (2022), no. 66.

[12] Robert C. Martin, Clean architecture: A craftsman's guide to software structure and design, Robert C. Martin Series, Prentice Hall, Boston, MA, 2017.

[13] Alex McBratney, Brett Whelan, Tihomir Ancev, and Johan Bouma, Future directions of precision agriculture, Precision Agriculture **6** (2005), 7–23.

[14] Luiz F. P. Oliveira, António P. Moreira, and Manuel F. Silva, Advances in agriculture robotics: A state-of-the-art review and challenges ahead, Robotics **10** (2021), no. 2.

[15] Sai Kirthi Pilli, Bharathiraja Nallathambi, Smith Jessy George, and Vivek Diwanji, eagrobot - a robot for early crop disease detection using image processing, 2014 International Conference on Electronics and Communication Systems (ICECS), 2014, pp. 1–6.

[16] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng, Ros: an open-source robot operating system, Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics (Kobe, Japan), may 2009.

[17] Salah Sukkarieh, Mobile on-farm digital technology for smallholder farmers, 2017: Transforming Lives and Livelihoods: The Digital Revolution in Agriculture, 7-8 August 2017 266635, Crawford Fund, August 2017.