Universidad de la República
Facultad de Ingeniería
Instituto de Computación
Uruguay

Gestión de Datos en Sistemas Ciberfísicos Orientados a la Agricultura Inteligente.

Agustín Gallego

Trabajo Final
Licenciatura en Computación
Universidad de la República

Montevideo, Uruguay. Agosto 2022.

Supervisores: Dra. Ing. Lorena Etcheverry
Dr. Ing. Facundo Benavides
Universidad de la República

Resumen

Durante su historia, la agricultura ha evolucionado para mejorar la utilización de los recursos disponibles, y así poder brindar productos de mejor calidad en mayor cantidad. Actualmente, el foco no es solamente optimizar la producción, sino hacerlo de una manera ecológicamente sustentable. Para ello se introducen metodologías de agricultura de precisión –orientadas al manejo de cultivos de manera más atómica– y de agricultura inteligente –utilizando técnicas de gestión y procesamiento de datos como soporte a la toma de decisiones. En este trabajo se llevó a cabo una revisión de literatura para evaluar el estado del arte de la gestión de datos en la agricultura inteligente, tomando como fuentes artículos publicados en revistas y conferencias con revisión de pares. La metodología utilizada plantea una serie de pasos a seguir para que la experiencia pueda ser repetible, objetiva y con foco en una temática previamente establecida. Además, se desarrolló un caso de estudio práctico, en el cual se simuló un sistema ciberfísico orientado a la toma de datos en una plantación de pinos y robles. La información obtenida se guarda en un gestor de base de datos relacional para su posterior procesamiento y análisis mediante el cálculo de índices de vegetación. Como resultado, se obtuvo una simulación en la que se exploran las distintas etapas correspondidas en un caso de uso de agricultura inteligente, mostrando una potencial aplicación de las distintas herramientas relevadas previamente en la revisión de la literatura. Si bien la revisión mostró que es un área con creciente interés y desarrollo en los últimos años, no se pudo encontrar trabajos prácticos públicamente accesibles que presenten esta temática de una manera holística. El trabajo aquí desarrollado se comparte en su totalidad como *software* libre y de código abierto, de manera que pueda ser reutilizado en el futuro.

Palabras clave: Gestión de Datos, Sistemas Ciberfísicos, Agricultura Inteligente

Índice general

1 Introducción

La agricultura es uno de los pilares de la subsistencia del ser humano; no solamente produce la materia prima necesaria para la alimentación de la población mundial, sino que en ciertos países y regiones es además un importante motor de la economía. Históricamente, ha sufrido importantes cambios tecnológicos y socioeconómicos, destacándose cuatro grandes períodos o "revoluciones" (Liu et al. (2021)). El primero, se estima, comenzó alrededor del año 10.000 a.C. y se lo conoce como la Revolución Neolítica. En ella se dan los primeros indicios de comunidades humanas capaces de cultivar plantas y domesticar animales. Con el advenimiento del motor de vapor en el siglo XVIII es que se entra en el segundo período, en el cual la productividad aumenta gracias a la mecanización de las distintas tareas –como ser arado, siembra y cosecha. Luego de la segunda revolución industrial, se aumenta aún más la eficiencia con la creación del motor de combustión interna y la aparición del tractor. También se destacan el uso de fertilizantes y pesticidas químicos. Luego, en 1950 la llamada Revolución Verde le sigue, marcando el tercer período. En él se utilizan nuevas variedades de plantas genéticamente modificadas, y se introduce tecnología de posicionamiento global y aplicación variable de pesticidas, fertilizantes, e irrigación –en contraposición a técnicas en las cuales la aplicación era homogénea en toda el área sembrada. Finalmente, en el último período se utilizan tecnologías de la información, vehículos no tripulados, dispositivos *IoT*[1] (por su sigla en inglés para *Internet of Things*), y almacenamiento y análisis de datos a gran escala para dar paso a la denominada Agricultura Inteligente.

La Agricultura Inteligente se vale de sistemas ciberfísicos para recolectar datos sobre los cultivos, como ser humedad del suelo o cantidades de ciertos nutrientes, mediante sensores montados en vehículos o estaciones fijas. Estos datos se podrán utilizar ya sea para tomar lecturas en tiempo real y actuar acorde a los valores presentes, para realizar análisis históricos sobre datos pasados y actuar acorde a predicciones a futuro, o para ambas. A su vez, estos datos podrán ser utilizados por otros sistemas ciberfísicos para actuar sobre los cultivos, o para recolectar datos más concretos sobre un área en particular. En cualquiera de estos casos se precisan técnicas de gestión de datos para almacenar y procesar la información de una manera eficiente. Las potenciales áreas de aplicación son muchas y muy variadas y, como mencionan los autores en Navarro et al. (2020), es una temática que en años recientes –en particular del 2016 en adelante– ha visto un desarrollo

[1] "Internet de las cosas (*IoT*) es la red de objetos físicos conectados mutuamente con el propósito de intercambio de datos mediante internet." – https://www.oracle.com/internet-of-things/what-is-iot/

significativo en el ámbito académico.

En la actualidad, la agricultura se enfrenta a un gran desafío: continuar incrementando la productividad con los suelos y sistemas hídricos disponibles. Según el informe de la *FAO* sobre el estado de los recursos para la agricultura mundial, no solo se está decrementando el margen disponible para nuevas superficies de tierra productiva, sino que la presión sobre los sistemas hídricos está aumentando (FAO (2021)). En el reporte se hace mención a que actualmente un 98 % de las calorías consumidas mundialmente provienen de actividades agrícolas que utilizan un modelo de intensificación que no es sostenible, y que ha llevado a los sistemas de tierras y aguas a una degradación y utilización límites. En las Fig. 1.1 y 1.2 se pueden ver datos sobre la degradación de la tierra, y la presión sobre los sistemas hídricos debido a la utilización de este sector productivo. Por último, el reporte subraya cómo el cambio climático aumentará los niveles de evapotranspiración y la cantidad y distribución de las precipitaciones, con lo cual habrá incluso mayores cambios y presiones en los sistemas agrícolas. Es en este contexto que la Agricultura Inteligente busca brindar nuevas herramientas y soluciones a las problemáticas modernas.

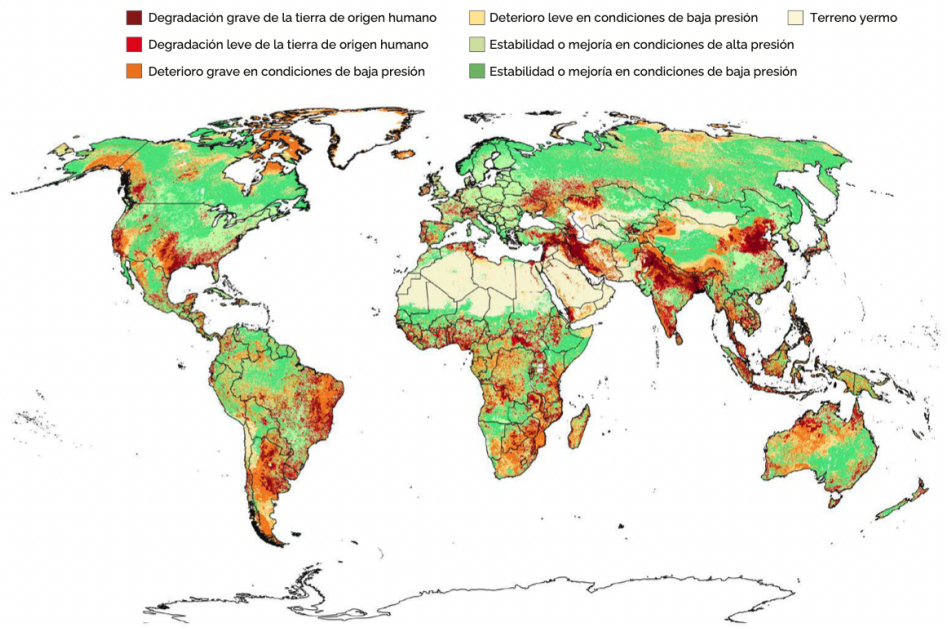


Figura 1.1: Degradación de la tierra (erosión del suelo, agotamiento de los nutrientes y aumento de la salinidad) según la gravedad de las presiones ejercidas por el ser humano, y las tendencias de deterioro. (FAO (2021))

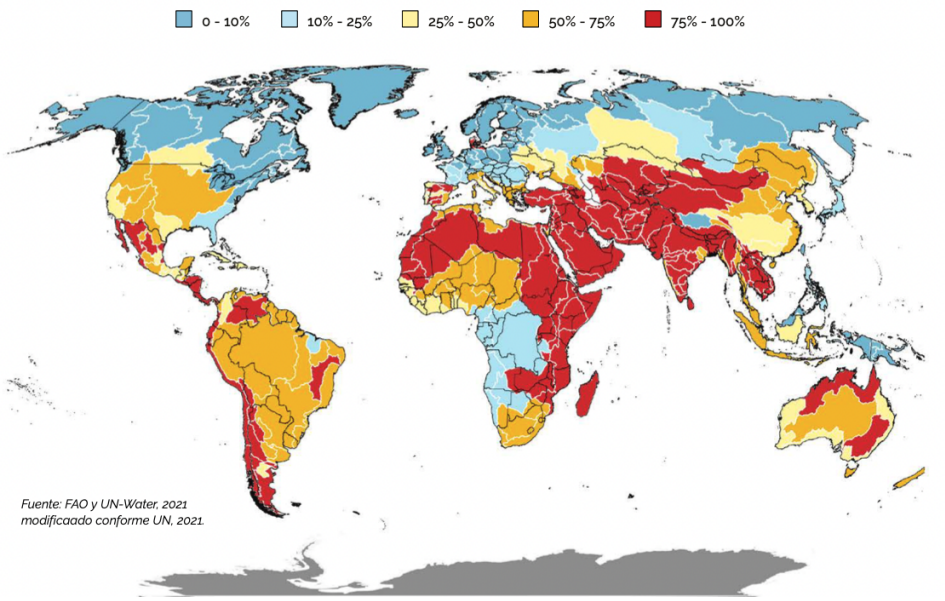


Figura 1.2: Nivel de estrés hídrico debido al sector agrícola, por cuenca. El indicador se define como la razón entre el total de agua dulce extraída y el total de recursos de agua dulce renovables, después de haber considerado las necesidades de caudal ambiental. (FAO (2021))

1.1. Objetivos y resultados esperados

El objetivo de este trabajo es comprender el estado del arte y los principales desafíos de la gestión de datos en entornos de Agricultura Inteligente. En particular, al ser un área tan vasta y rica en publicaciones, se tendrá como prioridad la evaluación de artículos que mencionen el uso de vehículos aéreos no tripulados o redes de sensores (*IoT*).

Los resultados esperados son el relevamiento de la literatura –producto del relevo de información de distintas propuestas, herramientas y tecnologías utilizadas– y el desarrollo de una prueba de concepto simulada que permita aproximarse al problema real. La simulación se compondrá de un sistema que genere datos, otro que los recolecte y otro que los almacene y procese.

1.2. Organización del documento

Este documento está organizado de la siguiente manera. En el Capítulo 2 se presenta la revisión de la literatura, se describe la metodología utilizada para realizarla, se definen los conceptos relevantes y se presentan los resultados de la misma. En el Capítulo 3 se desarrolla el caso de estudio y se describen los componentes de la solución planteada. En el Capítulo 4 se detallan los resultados obtenidos de la revisión de la literatura y la prueba de concepto implementada. Finalmente, en el Capítulo 5 se concluye la tesis y señalan los potenciales trabajos a futuro.

2 Revisión de la literatura

En este capítulo se describen en profundidad los procedimientos y procesos llevados a cabo para efectuar la revisión de la literatura, y los resultados obtenidos. En la Secc. 2.1 se describe la metodología utilizada, y en la Secc. 2.2 se presentan los resultados.

2.1. Metodología de revisión

Para la revisión de la literatura se optó por seguir la metodología propuesta en Okoli (2015). La misma se plantea en ocho pasos, detallando de manera explícita cómo operar en cada una de las etapas. A los propósitos de este trabajo de investigación, se efectuó el siguiente subconjunto de pasos:

1. Identificación del propósito: plantear las preguntas que se quieren responder.
2. Búsqueda y filtrado de literatura: elegir las palabras clave y los motores de búsqueda, y efectuar la búsqueda inicial. En este paso también se eliminan artículos que no son vistos como relevantes, a juzgar por el título y resumen de la publicación.
3. Extracción de información: analizar los artículos elegidos. Es posible que publicaciones no relevantes hayan pasado la etapa anterior, en cuyo caso se descartan en esta etapa, luego de una lectura con mayor detenimiento.
4. Síntesis de los estudios: recolectar la información extraída –ya sea cualitativa, o cuantitativa– en un repositorio común que permita su fácil interpretación.
5. Escritura de la revisión: además de la revisión en sí, se deben plasmar en la documentación los pasos seguidos, para que la actividad pueda ser reproducida en caso de ser necesario.

2.1.1. Identificación del propósito

Como se mencionó en el Cap. 1, el principal objetivo es relevar el estado del arte de la gestión de datos en sistemas ciberfísicos orientados a la Agricultura Inteligente. En particular se desea relevar información sobre sistemas ciberfísicos que se valgan de aeronaves

no tripuladas para la obtención de los datos. Dentro de la gestión de datos, interesa hacer hincapié en algunos aspectos en particular como los tipos de datos utilizados, las políticas de gestión de datos de alto nivel (usuarios, permisos, etc.), el volumen de los datos (generados, transmitidos y almacenados), la volatilidad de los datos y las arquitecturas utilizadas (procesamiento centralizado o distribuido, presencia de alta disponibilidad, etc.). A continuación se enumeran estos aspectos en su totalidad, planteando algunas de las preguntas que se buscó responder durante el relevamiento de la bibliografía.

Adquisición de los datos.
¿De qué dispositivos o fuentes provienen los datos?

Transferencia de los datos.
¿Cómo se transfieren los datos desde los nodos a los servidores de bases de datos?

Procesamiento de los datos.
¿De qué manera se manipulan los datos?

Almacenamiento de los datos.
¿Cómo y dónde se persisten los datos?

Características de los datos.
¿Qué tipos de datos se utilizan, y quiénes los pueden acceder?

Temporalidad de los datos.
¿Se utilizan los datos en tiempo real, o se almacenan para consultas históricas?

Arquitectura para la gestión de datos.
¿Los datos se procesan de manera centralizada o distribuida?

Escenario de aplicación.
¿El artículo refiere a un escenario de aplicación en particular?

Tipos de licencias.
¿Existe *software* desarrollado en el marco del artículo? En caso que se encuentre disponible, ¿con qué tipo de licencia?

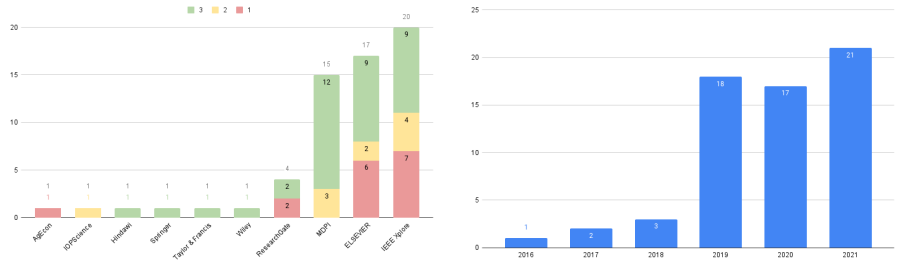
2.1.2. Búsqueda y filtrado de la literatura

Los motores de búsqueda utilizados fueron *Google Scholar*[1] y Timbó Foco[2]. *Google Scholar* provee acceso a publicaciones de variadas fuentes, y se vale de un indexador web (*web crawler*) para obtener nuevos resultados; en él se encuentran nucleadas publicaciones de sitios como *IEEE Xplore*, *Elsevier*, *MDPI* y *Wiley*. La principal desventaja es que los artículos indexados no siempre están publicados de manera abierta, con lo cual es

[1] Página principal del buscador *Google Scholar* – https://scholar.google.com/
[2] Página principal del buscador Timbó Foco – https://foco.timbo.org.uy/

necesario contar con una suscripción para la editorial que lo publicó –o pagar una tarifa por descargarlos. Por otra parte, si bien precisa de una cuenta de usuario, Timbó Foco oficia de *proxy* de suscripciones a editoriales como *IEEE Xplore*, *Elsevier* y *Springer*, y se utilizó como complemento para la búsqueda de artículos no disponibles gratuitamente.

Para compilar la lista inicial de artículos, se definieron búsquedas con combinaciones de las siguientes palabras clave: "smart farming", "precision agriculture", "data" y "data management". En una primer etapa de filtrado se analizaron los títulos de las publicaciones y los resúmenes, aceptando como válido cualquier artículo que hiciera mención a datos y sistemas ciberfísicos en ámbitos agrícolas. En esta etapa, se seleccionaron 62 artículos –de un total de aproximadamente 250 evaluados–, de los cuales 4 fueron descartados por no pertenecer a sitios con exigencias de revisión de pares[3]. En la siguiente etapa de filtrado se procedió a una lectura rápida de los artículos, y se asignó un índice de relevancia a cada uno: 1- poco relevante (no trata gestión de datos ni aeronaves no tripuladas), 2- relevante (no trata gestión de datos, pero sí de aeronaves no tripuladas), 3- muy relevante (trata ambos temas). En esta etapa se obtuvo también información sobre el tipo de artículo –revisión bibliográfica o caso de estudio–, el año de publicación y la cantidad de citas, y se ordenaron los artículos en función de estas nuevas categorías. En la Fig. 2.1 se muestran datos sobre las editoriales, el índice de relevancia y los años de publicación de los 62 artículos preseleccionados. Luego de ordenar los artículos por año y relevancia –en orden descendiente–, se compila la lista final de artículos a analizar. La cantidad final de artículos a relevar –compuesta de 33 publicaciones– se tomó estimando el tiempo total dedicado de 75 horas. En el Cuad. 2.1 se puede consultar la lista exhaustiva de los artículos analizados, con sus respectivos identificadores a utilizar en los distintos cuadros y referencias de la Secc. 2.2, e identificación de tipo de artículo –caso de estudio (CE) o revisión (R)– y referencia bibliográfica.



(a) (b)

Figura 2.1: Estadísticas de los artículos relevados, sus fuentes y años de publicación. En 2.1a se muestran artículos por editorial y la graduación inicial acorde al índice de relevancia. En 2.1b se muestran cantidad de artículos seleccionados publicados por año.

[3]En particular, los artículos fueron publicados en *ResearchGate*.

ID	Título	Tipo	Referencia
1	A Generic ROS-Based Control Architecture for Pest Inspection and Treatment in Greenhouses Using a Mobile Manipulator	CE	Martin et al. (2021)
2	Design of a Data Management Reference Architecture for Sustainable Agriculture	CE	Giray and Catal (2021)
3	Precision Agriculture Workflow, from Data Collection to Data Management Using FOSS Tools: An Application in Northern Italy Vineyard	CE	Belcore et al. (2021)
4	Data management and internet of things: A methodological review in smart farming	R	Debauche et al. (2021)
5	Big Data Storage and Analysis for Smart Farming	CE	Fote et al. (2020)
6	Smart data in innovative farming	R	Rakhra and Singh (2021)
7	Digitalization and Big data in smart farming – a review	R	Iaksch et al. (2021)
8	Agricultural data management and sharing: Best practices and case study	R / CE	Moore et al. (2021)
9	From Smart Farming towards Unmanned Farms: A New Mode of Agricultural Production	R	Wang et al. (2021)
10	A Review of Applications and Communication Technologies for Internet of Things (IoT) and Unmanned Aerial Vehicle (UAV) Based Sustainable Smart Farming	R	Islam et al. (2021)
11	Smart Farming in Europe	R	Moysiadis et al. (2021)
12	Unmanned Aerial Vehicles in Smart Agriculture: Applications, Requirements and Challenges	R	Maddikunta et al. (2021)
13	IoT-based data management for Smart Agriculture	CE	Martínez et al. (2020)
14	Big Data Processing Architecture for Smart Farming	CE	Roukh et al. (2020)
15	A Study of Smart Farming Based on IOT	R	Kanimozhi et al. (2020)
16	Applications of Remote Sensing in Precision Agriculture: A Review	R	Sishodia et al. (2020)
17	Smart Farming Introduction in Wine Farms: A Systematic Review and a New Proposal	R	Sarri et al. (2020)
18	A Systematic Review of IoT Solutions for Smart Farming	R	Navarro et al. (2020)

Cuadro 2.1 (continúa de la página anterior).

19	Internet of Things (IoT) and Agricultural Unmanned Aerial Vehicles (UAVs) in smart farming: A comprehensive review	R	Boursianis et al. (2022)
20	From Smart Farming towards Agriculture 5.0: A Review on Crop Data Management	R	Saiz-Rubio and Rovira-Más (2020)
21	A compilation of UAV applications for precision agriculture	R	Radoglou-Grammatikis et al. (2020)
22	CAN't – An ISOBUS Privacy Proxy for Collaborative Smart Farming	CE	Helmké et al. (2019)
23	A Review on UAV-Based Applications for Precision Agriculture	R	Tsouros et al. (2019)
24	The Digitisation of Agriculture: a Survey of Research Activities on Smart Farming	R	Bacco et al. (2019)
25	A Survey on the Role of IoT in Agriculture for the Implementation of Smart Farming	R	Farooq et al. (2019)
26	Unmanned Aircraft System (UAS) Technology and Applications in Agriculture	R	Hassler and Baysal-Gurel (2019)
27	An Architecture model for Smart Farming	CE	Triantafyllou et al. (2019)
28	Data Acquisition and Analysis Methods in UAV-based Applications for Precision Agriculture	R	Tsouros et al. (2019)
29	Big Data Transformation in Agriculture: From Precision Agriculture Towards Smart Farming	CE	Rodríguez et al. (2019)
30	Unmanned Aerial Vehicles in Agriculture: A Review of Perspective of Platform, Control, and Applications	R	Kim et al. (2019)
31	mySense: A comprehensive data management environment to improve precision agriculture practices	CE	Morais et al. (2019)
32	A Smart Decision System for Digital Farming	CE	Cambra Baseca et al. (2019)
33	Big Data in Smart Farming – A review	R	Iaksch et al. (2021)

Cuadro 2.1: Artículos pertenecientes a la lista de 33 publicaciones finales, con identificador –para referencia con tablas de la Secc. 2.2–, tipo de artículo –caso de estudio o revisión– y cita bibliográfica.

2.1.3. Extracción de información y síntesis de los estudios

Si bien en la metodología estas dos etapas se plantean como disjuntas, fueron realizadas en una misma iteración. Es decir, se analizó con más detenimiento los artículos preseleccionados, y se procedió a extraer la información –acorde a los puntos tratados en la Secc.

2.1.1– en una planilla de datos para su consulta e interpretación.

2.2. Revisión del estado del arte

En las siguientes secciones se detallan los resultados de la revisión de la literatura de acuerdo con los aspectos relevantes mencionados en la Secc. 2.1.1.

2.2.1. Adquisición de los datos

En los sistemas ciberfísicos en general, y en aquellos presentes en la Agricultura Inteligente en particular, las fuentes de los datos pueden ser muchas y de muy variadas características. Desde cámaras convencionales de tipo *RGB*[4], a cámaras multiespectrales o hiperespectrales[5], hasta sensores desplegados en la plantación y estaciones meteorológicas remotas. No hay un límite claro que defina el alcance de qué tipos de datos pueden ser recabados por los distintos sistemas, pero sí queda claro que se pueden agrupar en categorías que comparten en gran parte características similares. Es por ello que se optó por clasificarlos en categorías (o familias) de datos a más alto nivel. Las familias de datos resultantes son: sensores, imágenes satelitales, imágenes de vehículos terrestres y aeronaves, estaciones meteorológicas y otras fuentes –una especie de contenedor para el resto de los datos, que no pueden ser fácilmente agrupados o descritos con mayor profundidad. En cada familia se comparten características en común en cuanto a resolución temporal y espacial, es decir, cada cuánto se generan datos, y de qué tipos son y qué dimensiones tienen, respectivamente.

Los sensores recolectan datos directamente en la fuente, es decir, se ubican en la parcela de tierra con los cultivos y toman mediciones de manera constante. Típicamente, los datos recolectados son numéricos y su resolución temporal es alta, y si bien no se espera gran variabilidad segundo a segundo, tienen la capacidad de recolectar datos a ese nivel, en caso de ser necesario. Dado que los sensores no son unidades autosuficientes, precisan de un ambiente de cómputo que gestione la recolección y el almacenamiento, o el envío de datos hacia centros de cómputo remotos. Los computadores de una sola placa (o *SBCs*, por su sigla en inglés *single-board computers*) son muy populares en este respecto, brindando un entorno de desarrollo que se adapta a las distintas necesidades de los ambientes *IoT*. Por ejemplo, es necesario que los requerimientos energéticos sean mínimos, ya que en la mayoría de los casos estos sensores estarán desplegados en zonas remotas, sin acceso directo a un tendido eléctrico, y deberán valerse de baterías o generadores de electricidad situados in situ –mediante celdas solares u otros generadores de bajo porte. Por otro lado,

[4] Las cámaras *RGB* captan luz dentro del espectro visible por el ojo humano.

[5] Las cámaras multi o hiperespectrales captan luz fuera del espectro visible, en longitudes de onda mayores al rojo y menores al violeta.

deberán contar con la habilidad de conectarse con una base de cómputos remota, y también de almacenar temporalmente los datos en caso de desconexión. Existen varios tipos de *SBCs*, pero según Navarro et al. (2020) y Triantafyllou et al. (2019) los más populares son *Arduino* –un microcontrolador programable– y *Raspberry Pi v.3* –una computadora con soporte a sistemas operativos compilados para la arquitectura *ARM*.

Las cámaras pueden recolectar datos de tres maneras: en la parcela de tierra desde vehículos terrestres, sobre la parcela de tierra desde aeronaves, o desde satélites que orbitan la tierra. A su vez, los vehículos terrestres y aeronaves podrán ser tripulados o no tripulados, y dentro de estos últimos podrán ser piloteados remotamente o autónomos. Los tipos de datos recolectados son mapas de bits. Existen cámaras de tipo *RGB*, multiespectrales e hiperespectrales; cuál se deba utilizar dependerá de qué tipo de información se quiera obtener. En cuanto a la variabilidad en el tiempo, la cadencia de las imágenes provenientes de satélites depende por un lado de la velocidad de órbita del satélite que las recolecta, y por otro del nivel de acceso que se tenga a los datos, y suele medirse en días o semanas. Las provenientes de vehículos terrestres o aeronaves dependen mayormente del nivel de acceso que se tenga a dicha infraestructura, pudiendo ser medida en horas, días o semanas.

Las estaciones meteorológicas proveen datos numéricos variados, como ser velocidad y dirección del viento, cantidad de precipitaciones y su probabilidad, humedad, temperatura y nubosidad. Su variabilidad en el tiempo es baja, pudiendo ser consultada diaria o semanalmente.

Por último, se pueden recolectar datos de otras bases de datos –existen proyectos que comparten públicamente datos, como *Agro API*[6] o la Infraestructura de Datos Espaciales (IDEuy)[7]–, de otros tipos de sensores –como los sensores *LiDAR*, que pueden ser utilizados para la construcción de modelos *3D* o mapas topográficos del terreno–, o incluso de otras fuentes como el *web scraping*[8]. En este caso, dado que se engloba al resto de los datos en una familia, no se puede plantear una generalización sobre su tipo o variabilidad en el tiempo.

En el Cuad. 2.2, se muestran las familias de datos mencionadas en cada artículo, y sus principales características de resoluciones temporales y espaciales. La resolución temporal refiere a la potencial rapidez con la que se pueden obtener los nuevos datos: alta (segundos o minutos), media (horas) y baja (días o semanas). Por otro lado, la resolución espacial es cuánto ocupan esos datos: alta (Gb), media (Mb) y baja (Kb).

2.2.2. Transferencia de los datos

La transferencia de los datos se puede caracterizar de dos maneras: transferencia física y lógica. La transferencia física refiere a cuál es el medio que se utiliza y cómo se envían los

[6] Página principal de *Agro API* – https://agromonitoring.com/
[7] Página principal de IDEuy – https://www.gub.uy/infraestructura-datos-espaciales/
[8] El *web scraping* es una técnica que permite obtener información de manera automatizada de sitios web.

Familia de datos	Resolución Temporal	Resolución Espacial	Artículo
Sensores	Alta	Baja	2-7, 9-13, 15-16, 18-20, 23-25, 27, 29, 31, 32
Imágenes satelitales	Baja	Media	11, 13, 15, 16, 20, 24, 3
Imágenes de vehículos terrestres o aéreos	Media	Alta	1, 4, 9-12, 15, 16, 18-21, 23-28, 30
Otras fuentes	N/A	N/A	2, 5, 11, 13, 14, 18, 20, 24, 27

Cuadro 2.2: Familias de datos mencionadas por artículo, y sus principales características.

datos en este. En particular, interesó relevar información sobre transferencias inalámbricas, dado que los entornos se despliegan típicamente en ambientes rurales remotos en los que no es posible –o rentable– contar con una infraestructura completamente cableada. En Debauche et al. (2021) se muestra un esquema con una arquitectura genérica utilizada en entornos *IoT* de Agricultura Inteligente, en el cual la comunicación entre los dispositivos y los servidores se efectúa exclusivamente de manera inalámbrica (ver Fig. 2.2).

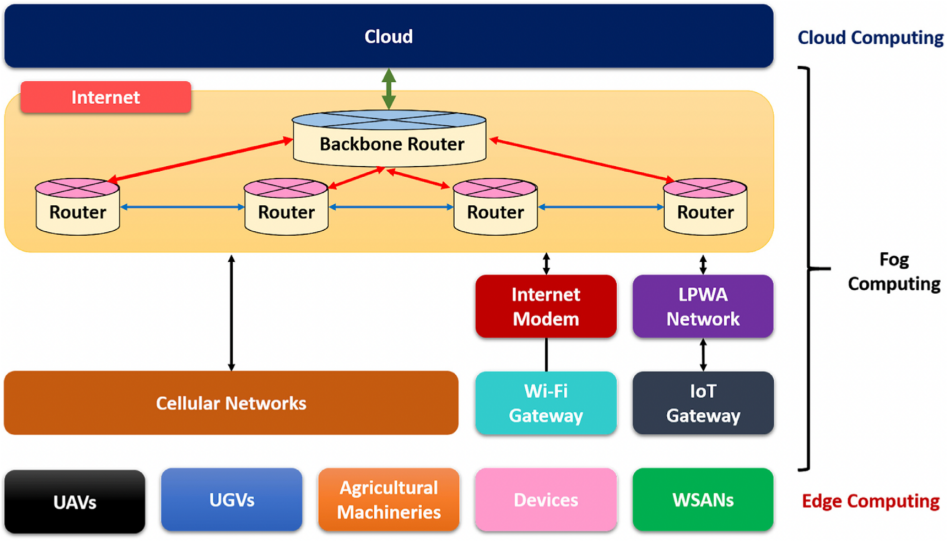


Figura 2.2: Arquitectura genérica propuesta para entornos de Agricultura Inteligente implementados con dispositivos *IoT*. Todas las comunicaciones entre los nodos *Edge* y el resto de los componentes es mediante protocolos de comunicación inalámbrica. (Debauche et al. (2021))

Por otro lado, la transferencia lógica refiere a cómo se agrupan los datos en unidades atómicamente transferibles –o paquetes–, sabiendo que la capa física resuelve el envío

de los mismos en el medio. Tomando el modelo de interconexión de sistemas abiertos (o modelo *OSI*, por su sigla en inglés *Open Systems Interconnection*) como referencia, diremos que la transferencia física de los datos corresponde a las primeras tres capas –física, enlace de datos y red– y la transferencia lógica corresponde a las últimas cuatro –transporte, sesión, presentación y aplicación.

Para entender qué tipos de protocolos físicos o de red se podrán utilizar en un entorno dado, es necesario conocer las características de los datos que se deberán comunicar. Es decir, en términos de lo discutido en la Sección 2.2.1, qué tipos de datos y con que resolución en el tiempo. Además, cada protocolo tendrá asociadas otras características como rango efectivo de comunicación, utilización de recursos –como batería– y calidad de la red –tolerancia a desconexiones o garantías de consistencia.

Cabe también mencionar que se puede complementar alguna de las carencias o desventajas de los protocolos elegidos con decisiones en la arquitectura de la red. Por ejemplo, es cada vez más común ver utilizadas redes inalámbricas de sensores (o *WSNs*, por su sigla en inglés *Wireless Sensor Networks*) en entornos en los que la conectividad o la infraestructura de la red son problemáticas o inadecuadas. En una *WSN*, los nodos son ellos mismos enrutadores de tráfico de otros nodos, formando de esta manera una red interconectada de nodos de sensores[9]. Algunas de las ventajas de una arquitectura de este estilo son la independencia de infraestructura dedicada a la comunicación de datos –como antenas celulares cercanas o conexión satelital disponibles– y la posibilidad de contar con una arquitectura diseñada específicamente para las necesidades de los nodos de los que se dispone. Por otro lado, entre las desventajas se encuentran mayor gasto energético y complejidad en cada nodo, y potencial introducción de puntos únicos de falla en la red –si falla un nodo que conecta dos grupos de nodos inconexos, los grupos quedarán incomunicados uno del otro.

Transferencia física

A nivel físico se cuenta con los siguientes protocolos de red: *Bluetooth* y *BLE* (*Bluetooth Low Energy*), Celular (2G, 3G, 4G, 5G), *LR-WPAN* (*Low-Rate Wireless Personal Area Network*), *LoRa WAN* (*Long Range Wide Area Network*), *NB-IOT* (*Narrowband IoT*), *NFC* (*Near-field Communication*), *RFID* (*Radio-frequency Identification*), *SigFox*, *WiFi* (*Wireless Fidelity*), *WiMAX* (*Worldwide Interoperability for Microwave Access*) y *ZigBee*. La elección del protocolo a utilizar dependerá de los requerimientos de ancho de banda, consumo energético, rango efectivo de comunicación y costo de implementación.

En Boursianis et al. (2022) se destacan los protocolos *NB-IoT* y *LoRa WAN* como prometedores dentro de la Agricultura Inteligente. En caso de contar con infraestructura desplegada en interiores –como ser en un invernadero– los autores recomiendan *NB-IoT* que cuenta con bajos costos de operación, bajo consumo energético y soporte a mayor

[9] Definición de *WSN* según *ScienceDirect* – https://www.sciencedirect.com/topics/engineering/wireless-sensor-network

cantidad de dispositivos conectados, ofreciendo velocidades de hasta 200 Kbps. En caso contrario, para entornos en los que haya sensores desplegados en locaciones remotas, recomiendan implementar la red utilizando *LoRa WAN*, que no solamente tiene requerimientos energéticos muy bajos, sino que posee un gran rango efectivo de comunicación –llegando a los 20 kilómetros de alcance. El ancho de banda disponible es limitado, soportando velocidades de hasta 50 Kbps, pero remarcan que debería ser suficiente para transmitir datos de los sensores típicamente encontrados en este tipo de entornos. En el Cuad. 2.3 se muestra un resumen de las capacidades y principales características de cada uno, así como las menciones por artículo.

Protocolo	Alcance	Ancho de banda	Frecuencia	Consumo	Artículo
Bluetooth/BLE	100 m	1 - 24 Mbps	2.4 GHz	Muy Bajo	4[10], 11, 19, 27
Celular	>100 m[11]	64 Kbps - 1 Gbps	0.9 - 8 GHz	Medio	11, 19, 27
LR-WPAN	30 m	20 - 250 Kbps	0.9 - 2.4 GHz	Bajo	19, 27
LoRa WAN	20 km	0.3 - 50 Kbps	900 MHz	Muy Bajo	10, 19, 27
NB-IOT	1 - 10 km	200 Kbps	800 - 1800 MHz	Bajo	10, 19
NFC	20 cm	400 Kbps	13.5 MHz	Bajo	4
RFID	10 cm - 200 m	400 Kbps	0.125 - 10 GHz	Bajo	4, 27
SigFox	10 - 40 km	0.1 - 100 Kbps	433 - 915 MHz	Muy Bajo	4, 10, 11, 19
Wi-Fi	100 m	0.2 - 1.3 Gbps	2.4 - 5 GHz	Alto	4, 11, 19[12], 27
WiMAX	50 km	2 - 66 GHz	1 Gbps	Alto	4, 19
ZigBee	100 m	20 - 250 Kbps	0.9 - 2.4 GHz	Medio	4, 11, 27

Cuadro 2.3: Protocolos de red mencionados por artículo y sus principales características.

En el Cuad. 2.4 se puede ver una posible agrupación de distintos protocolos de transferencia de datos por alcance y periodicidad de la comunicación. Además, en Navarro et al. (2020) se puede consultar una lista de protocolos de red utilizados por proyecto que puede ser de utilidad.

	Periodicidad de la comunicación		
	Continua	**Regular**	**Evento**
<10 m	BLE	BLE	RFID-NFC
10-100 m	Wi-Fi	Wi-Fi	Wi-Fi
		Zigbee	Zigbee
>100 m	NB IoT	NB IoT	Sigfox
	LoRa WAN	LoRa WAN	LoRa WAN
	WiMAX	WiMAX	WiMAX

Cuadro 2.4: Protocolos de red mencionados, agrupados por rango efectivo y periodicidad de la comunicación.

[10] El artículo [4] menciona el rango como hasta 10 m.
[11] Depende de la cobertura en esa zona.
[12] El artículo [19] menciona la frecuencia máxima como de 60 GHz.

Transferencia lógica

A nivel lógico, se mencionan los siguientes protocolos: *AMQP* (*Advanced Message Queuing Protocol*), *CoAP* (*Constrained Application Protocol*), *HTTP* (*Hypertext Transport Protocol*) y *MQTT* (*MQ Telemetry Transport*). En el Cuad. 2.5 se muestran referencias a los protocolos mencionados por artículo, así como su tipo de arquitectura y protocolos de capa de transporte sobre los que operan. En Farooq et al. (2019) se puede encontrar una breve reseña, destacando las particularidades de cada uno y mencionando que la característica común a todos es la de ser protocolos livianos –por ello su popularidad en ambientes *IoT*. En particular, *CoAP* corre sobre *UDP*[13] y sigue una arquitectura de tipo pedido/respuesta. *HTTP* es también utilizado en modalidad pedido/respuesta, y soporta tanto *UDP* como *TCP*[14] como protocolos de capa de transporte. Por otro lado, *AMQP* y *MQTT* utilizan una arquitectura de tipo publicación/suscripción, y dependen de *TCP* como protocolo de capa de transporte. En todos los casos, estos protocolos soportan la utilización de funcionalidades de transporte cifrado, mediante la utilización de *TLS*[15].

Si bien no fue parte de la revisión de la literatura, en Glaroudis et al. (2020) se puede leer con más detalle un análisis de protocolos de transferencia para entornos orientados a *IoT* que puede ser de utilidad.

Protocolo	Tipo de arquitectura	Capa de transporte	Artículo
AMQP	Publicación/suscripción	*TCP*	1, 5, 25
CoAP	Pedido/respuesta	*UDP*	5, 13, 18, 24, 25
HTTP	Pedido/respuesta	*UDP* / *TCP*	5, 18, 25, 31, 32
MQTT	Publicación/suscripción	*TCP*	5, 13, 14, 18, 24, 25

Cuadro 2.5: Protocolos de transferencia lógica mencionados por artículo.

2.2.3. Procesamiento de los datos

Según lo relevado, el procesamiento de los datos puede darse en dos etapas: previo al momento de ingresarlos a la base de datos –como parte del proceso de calidad de datos–, o en una etapa de procesamiento posterior –como parte de un proceso más complejo de análisis de datos y extracción de conocimiento e información.

En Debauche et al. (2021), Giray and Catal (2021) y Wolfert et al. (2017), los autores hacen hincapié en la importancia de la calidad de los datos en los sistemas orientados al

[13] *UDP* (por su sigla en inglés *User Datagram Protocol*) es el protocolo de capa de transporte definido en el RFC 768 – https://datatracker.ietf.org/doc/html/rfc768

[14] *TCP* (por su sigla en inglés *Transmission Control Protocol*) es otro protocolo de capa de transporte, definido en el RFC 675 – https://datatracker.ietf.org/doc/html/rfc675

[15] *TLS* (por su sigla en inglés *Transport Layer Security*) es un protocolo de encriptación de comunicaciones. Su última versión es la 1.3, definida en el RFC 8446 – https://datatracker.ietf.org/doc/html/rfc8446

soporte de toma de decisiones (o *DSSs*, por su sigla en inglés *Decision Support Systems*), y mencionan que una baja calidad de datos afecta directamente la calidad de las conclusiones a las que se puede llegar a partir de ellos[16]. Entre otros, mencionan como posibles puntos a tratar en este aspecto a la detección de valores atípicos (*outliers*) o errados, a la inferencia de datos faltantes mediante interpolación y al manejo de valores duplicados (*deduplication*). En la Fig. 2.3 se muestra una secuencia de operaciones posible para el manejo de la calidad de los datos en una etapa de preprocesamiento. La calidad de los datos es mencionada en los siguientes artículos: 2, 4 y 33.

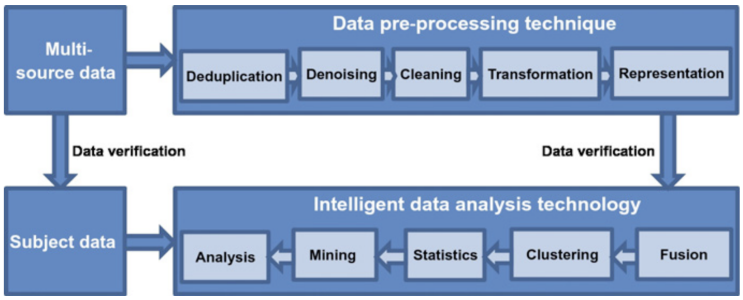


Figura 2.3: Cadena de eventos en procesos de calidad de datos (arriba –"*Data pre-processing technique*"–) y posprocesamiento (abajo –"*Intelligent data analysis technology*"–). (Wolfert et al. (2017))

Por otro lado, en Saiz-Rubio and Rovira-Más (2020) se habla de los sistemas de soporte a la toma de decisiones como una plataforma ideada para, a partir de los datos provenientes de las parcelas de tierra y mediante el uso de inteligencia artificial, arribar a una decisión que permita actuar sobre el entorno monitorizado. De esta manera, se puede además retroalimentar un sistema de toma de decisiones ideado para mejorar los procesos y productos. Además, se plantea que los datos se pueden transformar de manera que pueda ser más conveniente su uso. En particular, supongamos que se tienen datos de un viñedo, en el cual cada planta en particular fue catalogada utilizando un índice de vegetación. Interesa generar un nuevo mapa con los datos consolidados en grupos homogéneos más grandes, para poder tomar decisiones que puedan ser utilizadas más eficientemente. En este caso, los datos se están transformando para desplegar la información que brindan de una manera más conveniente y entendible. En la Fig. 2.4 se muestra un ejemplo de cómo se pueden agrupar los datos para generar una nueva representación de la realidad. Qué tipo de decisión se tome dependerá del caso en particular, pero es claro que hay dos grandes regiones en las que el índice de vegetación es similar según el criterio de agrupamiento utilizado.

En general, en la bibliografía relevada se plantea el uso de Inteligencia Artificial –en particular de Aprendizaje Automático (*Machine Learning*) y Aprendizaje Profundo (*Deep*

[16] *"Garbage In, Garbage Out"* – https://www.worldwidewords.org/qa/qa-gar1.htm

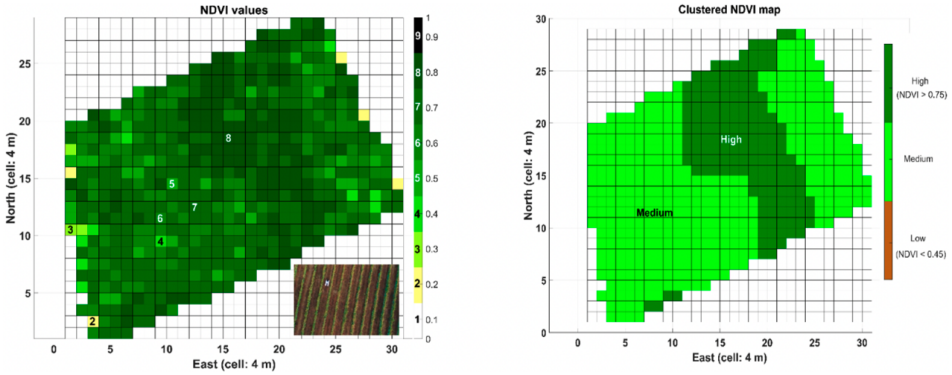


Figura 2.4: A la izquierda, valores atómicos de cada zona en la plantación. A la derecha, la transformación de los datos luego de aplicar un algoritmo de agrupamiento. (Saiz-Rubio and Rovira-Más (2020))

Learning)– como herramientas cada vez más explotadas en sistemas de esta índole, en donde existe una gran cantidad de datos, o donde la relación entre los datos disponibles y la información que se desea extraer de ellos no es necesariamente directa o depende de distintos factores –en el ejemplo anterior, ¿la planta muestra un índice bajo por falta de riego o nutrientes, o por enfermedades?. Se pueden utilizar estas técnicas en diversas áreas: para predecir métricas a futuro, analizar imágenes en busca de objetos (frutos, pestes, etc.), para interpretar índices de vegetación, e incluso para generar nuevos. Si bien no se ahonda en las técnicas utilizadas en particular, en Moysiadis et al. (2021), Navarro et al. (2020) y Hassler and Baysal-Gurel (2019) los autores dedican una sección a presentar trabajos relacionados al uso de Inteligencia Artificial en entornos de Agricultura Inteligente. Por último, se encuentran además –en menor medida– referencias a otras técnicas como Lógica Difusa (*Fuzzy Logic*) y Visión Artificial (*Computer Vision*). En el Cuad. 2.6 se muestran referencias a tipos de transformaciones de datos citadas por artículo.

Técnica de procesamiento de datos	**Artículo**
Inteligencia Artificial[17]	9, 15, 16, 18-20, 24
Aprendizaje Automático	2, 9, 11, 13, 16, 18-21, 23-26, 28, 33
Aprendizaje Profundo	1, 2, 9, 11, 16, 18-20, 23, 26, 28, 30
Otros	1, 2, 9, 11, 18-21, 23, 26, 28, 33

Cuadro 2.6: Técnicas de procesamiento de datos mencionadas por artículo.

[17] Si bien Inteligencia Artificial es un término que engloba el resto de las técnicas mencionadas, se agrega por completitud en las menciones por artículo.

2.2.4. Almacenamiento de los datos

El almacenamiento de datos es un área que puede ser muy amplia: desde la utilización de estructuras de datos en memoria, pasando por datos guardados en simples archivos de texto, a la utilización de herramientas diseñadas expresamente a los efectos de la persistencia de los mismos. En esta revisión interesó relevar datos sobre la utilización de sistemas de gestión de bases de datos[18] (o *DBMS*, por su sigla en inglés *Database Management System*) en particular. Existe una gran variedad de ellos, pero entre los más utilizados se encuentran los relacionales, de documentos, de búsqueda, de series de tiempo, de grafos, los columnares y los de clave-valor. En el ranking *DB-Engines*[19] se puede ver un listado con los manejadores más populares, así como también una sección con un glosario que muestra los distintos tipos que existen. Qué tipo de manejador se adapta mejor a los datos dependerá de cómo se modelen, de las restricciones que se tengan y de las funcionalidades que se deseen utilizar. Históricamente, los motores de bases de datos relacionales han dominado el mercado, aunque en tiempos recientes es cada vez más común ver entornos heterogéneos en donde se utilizan, además, otros manejadores para un subconjunto de los datos.

Dentro del total de artículos relevados, siete (21 %) hicieron mención explícita al manejador de base de datos utilizado, y es interesante notar que todos ellos son herramientas de código abierto. Además, tres de esos proyectos mencionaron utilizar tanto uno relacional como otro no relacional al mismo tiempo, demostrando que los entornos heterogéneos son de hecho utilizados en la práctica. Los manejadores relacionales nombrados fueron *MySQL* y *PostgreSQL*, y los no relacionales *Cassandra*, *Elasticsearch* y *MongoDB*. En el Cuad. 2.7 se muestran los motores de bases de datos utilizados por artículo, y sus principales características.

Gestor de base de datos	Tipo	Distribuido	Artículo
Elasticsearch	de Búsqueda	Sí	1
Cassandra	Columnar	Sí	5, 14
MongoDB	de Documentos	Sí	31
MySQL	Relacional	No	27, 31, 32
PostgreSQL	Relacional	No	3, 5, 14

Cuadro 2.7: Sistemas de gestión de bases de datos por artículo y sus principales características.

Un sistema de soporte a la toma de decisiones (*DSS*) para entornos de Agricultura Inteligente es desarrollado en Cambra Baseca et al. (2019), en el cual se utiliza *MySQL* –un manejador de bases de datos relacional– para la gestión de datos. El manejador es encargado de gestionar todos los datos generados por el entorno: desde la información proveniente de sensores e imágenes terrestres y aéreas –con sus respectivos índices computados–, a

[18] También llamados manejadores o administradores de bases de datos.

[19] Ranking de manejadores de bases de datos – https://db-engines.com/en/ranking

los datos de estaciones meteorológicas cercanas e interacciones entre usuarios. Dado que no se mencionan particularidades de la implementación, o sobre el manejo de datos y metadatos, no es posible ahondar en su estudio.

PostgreSQL es un manejador de bases de datos relacional que soporta la adición de funcionalidades mediante módulos externos (o extensiones). Entre ellas, una de las más populares es *PostGIS*, que implementa los tipos de datos y funciones necesarias para manipular información en un sistema de información geográfica (SIG). En Belcore et al. (2021) se formula una implementación holística –desde la recolección de datos, hasta la visualización de los mismos– enteramente basada en herramientas de código abierto, que utiliza *PostgreSQL* y *PostGIS* para procesar datos espaciales de un viñedo. Si bien se muestra información sobre la estructura de las tablas utilizadas, solamente se describe una de ellas en profundidad, por lo que no es posible evaluar con mayor profundidad la solución planteada.

En Morais et al. (2019), se presenta también una implementación de un sistema de soporte a decisiones que utiliza dos tipos distintos de manejadores: *MySQL* (relacional) y *MongoDB* (de documentos). La arquitectura fue diseñada teniendo en cuenta que los terminales ubicados en la plantación no disponen de una conexión confiable, y por lo tanto deben poder guardar datos localmente en caso de problemas en la red. Para ello, se despliega localmente un servidor *MySQL* que se encarga de persistir temporalmente los datos hasta que sea posible enviarlos a la unidad central de procesamiento. Dentro de la unidad central de procesamiento se utiliza tanto *MySQL* como *MongoDB*. *MySQL* es encargado de gestionar los datos cuasiestáticos, como metadatos de los terminales, usuarios y reportes, y *MongoDB* de los datos provenientes de los sensores. *MongoDB* es un gestor de bases de datos distribuido orientado al manejo de documentos, sin requerimientos de esquemas definidos y con escalado horizontal (*sharding*) implementado de forma nativa. Los documentos se conforman acorde al estándar de texto *JSON*[20], que en términos simples se define como una agrupación de uno o más campos de tipo clave-valor, y dado que no requiere que las tablas utilicen un esquema definido, es útil para operar con datos no estructurados. En este caso, el mayor beneficio de utilizar un manejador con estas prestaciones es la facilidad para manejar grandes datos que provee el *sharding* nativo, permitiendo fácilmente desplegar más instancias para elevar la capacidad de almacenamiento y procesamiento de datos. Esto no es un problema en los terminales de la plantación, ya que el volumen de datos generado por los sensores no amerita tales consideraciones.

En Martin et al. (2021), los autores utilizan un vehículo no tripulado para la detección y tratamiento de plagas en invernaderos. La arquitectura en capas diseñada utiliza *ROS*[21] para controlar el robot, que a su paso por el invernadero genera datos de la capa de registro (o *logging*), documentando no solo el estado de las plantas, sino que también generando

[20] Página principal del formato de texto *JSON (JavaScript Object Notation)* – https://www.json.org/json-en.html

[21] *Robot Operating System*, o *ROS*, es un entorno diseñado para el trabajo con robots. Provee un conjunto de bibliotecas y herramientas que pueden ser utilizadas para controlar sistemas robóticos (o incluso simularlos en su totalidad).

datos sobre su propio funcionamiento interno. Para gestionar estos datos se utiliza *Elasticsearch*, que es un manejador de base de datos de búsqueda con foco en almacenamiento y recuperación de documentos (con formato *JSON*). En particular, su fortaleza radica en búsquedas en texto completo (*full-text*) en entornos distribuidos.

Si bien *MySQL* es mencionado en Triantafyllou et al. (2019) como parte de la capa de gestión de información, no se proveen mayores indicios de cómo se utiliza. Los autores se limitan a mencionar a la base de datos como un repositorio central de información, sin ahondar en más detalles.

Por último, en Fote et al. (2020) y Roukh et al. (2020), si bien se mencionan *PostgreSQL* y *Cassandra* como manejadores utilizados, se limitan a catalogar su uso como motores de datos estructurados y no estructurados, respectivamente, sin mencionar particularidades de la implementación. *Cassandra* es un manejador de base de datos columnar, y al igual que *MongoDB* está orientado al procesamiento de grandes volúmenes de datos –ya que es un manejador distribuido que provee escalado horizontal nativo. La gran diferencia radica en cómo organiza internamente la información de cada registro, agrupando con mayor afinidad los datos de cada columna (o familia de columnas) para un rápido acceso de lectura en tipos de datos que requieren operaciones sobre las mismas. Un ejemplo de esto puede ser computar el promedio de precipitaciones y temperaturas máxima y mínima para un sensor dado, en un período de tiempo.

Big Data

Además de mencionar motores de bases de datos en particular, en la literatura se hizo referencia a requerimientos de manejar datos a gran escala, más conocido como *Big Data*. Los datos a gran escala están caracterizados por varias dimensiones, no solamente por el tamaño del conjunto de datos (*dataset*), sino además por otros factores como la velocidad de arribo de nuevos datos, o la variedad de las fuentes que los originan. En Rodríguez et al. (2019) se definen los datos a gran escala como conjuntos de datos de gran volumen, generados a gran velocidad y de una variedad de fuentes, de los cuales se puede extraer conocimiento (o valor) con exactitud (o veracidad). A esta definición se la conoce como las cinco "V", ya que en inglés se pueden definir estas dimensiones como: *volume, velocity, variety, value* y *veracity*. Si bien esta definición puede ser considerada como vaga, ya que no cuantifica de ninguna manera las dimensiones presentadas, se podría decir que se está ante datos a gran escala si no es posible procesarlos eficientemente mediante técnicas y herramientas de gestión de datos tradicionales[22] [23] –esencialmente, manejadores de base de datos relacionales no distribuidos. Se puede extender la cantidad de dimensiones a diez, como se muestra en Fote et al. (2020)[24], pero en términos generales basta

[22] Definición de *Big Data* según *IBM* – https://www.ibm.com/hk-en/analytics/hadoop/big-data-analytics

[23] Definición de *Big Data* según *Oracle* – https://www.oracle.com/big-data/what-is-big-data/

[24] *Venue, validity, variability, vagueness* y *vocabulary*.

con utilizar las cinco "V" mencionadas anteriormente. En Jacobs (2009), el autor plantea los principales problemas que se pueden encontrar al enfrentarse a datos a gran escala, y discute posibles soluciones a ellos, como ser la desnormalización de los datos en pos de optimizar la lectura, o la distribución de los datos en múltiples nodos para el procesamiento en paralelo. Si bien en la literatura se hace referencia a *Big Data* en un gran número de artículos, no se encontraron mayores detalles sobre su implementación, o las dimensiones que estos datos a gran escala tomaron. Tanto en Rakhra and Singh (2021) como en Wang et al. (2021) se subraya este hecho, y se concluye que, si bien presente, *Big Data* –como disciplina– está todavía en sus etapas iniciales de crecimiento en lo que respecta a entornos de Agricultura Inteligente.

2.2.5. Características de los datos

El objetivo principal de esta categoría es el de identificar las principales características de los datos en el contexto de las bases de datos; por ejemplo, qué tipos de datos atómicos se utilizan para almacenar la información proveniente de las fuentes, o qué clase de segregación de permisos de acceso tiene cada tipo de usuario sobre ellos. Debido a la falta de información en la literatura relevada, no fue posible hacer un estudio del estado del arte en esta categoría. Se destaca el trabajo en Iaksch et al. (2021) en este aspecto, en el cual se arriba a la misma conclusión de falta de referencias, y se plantea como potencial área para trabajar a futuro el desarrollo de un modelo de datos universal para la implementación de sistemas de Agricultura Inteligente.

2.2.6. Temporalidad de los datos

En general, cuando se monitoriza un sistema existen dos posibles maneras de utilizar los datos resultantes. En tiempo real, para poder mejorar los tiempos de reacción ante ciertos eventos, o en diferido, para optimizar un proceso, aprender de experiencias pasadas, o examinar potenciales fuentes de errores o problemas. Para la utilización en tiempo real interesa modelar el sistema de tal manera de minimizar el tiempo de reacción ante los eventos que generaron los datos. En estos casos la retención de los datos no es necesariamente un problema, dado que luego de haber actuado sobre el evento se pueden descartar. Contrariamente, para la utilización en diferido –o de análisis histórico–, en general no importa tanto la rapidez de arribo de los datos como el poder de retención de los mismos. En este tipo de sistemas el problema es cómo escala la solución en el tiempo, dado que es costoso mantener datos históricos con gran precisión en el tiempo. La solución es, típicamente, utilizar técnicas de agregación de datos para minimizar la cantidad a almacenar –por ejemplo, datos que se generaron cada cinco minutos pueden ser promediados y agrupados por hora. Qué tipo de agregación se haga dependerá del tipo de dato y de la información que se quiera extraer de ellos, y cada caso en particular ameritará un estudio de la realidad a modelar.

Como es de esperar, en entornos de Agricultura Inteligente interesan tanto los datos en tiempo real como los históricos. Tanto es así que en la mayoría de los artículos se hizo mención no solo a una de estas técnicas, sino que a las dos por igual. Los artículos en los que se hizo mención al manejo de datos en el tiempo son los siguientes: 1, 2, 4-7, 9, 11, 13-16, 18-20, 23, 25-29 y 31-33. En particular, en Morais et al. (2019) se muestran las dos instancias encargadas de gestionar los datos en el diagrama de arquitectura de la base de datos. En una de ellas se guardan datos de los últimos treinta días, y en la otra datos históricos semanales (ver Fig. 2.5). Además, se detalla un módulo de alertas en tiempo real encargado de enviar notificaciones con evaluaciones de riesgo de enfermedades o pestes que afectan a la plantación. En particular, se presenta el mildiu de la vid[25] como ejemplo, aunque no se menciona qué datos o reglas componen las alertas. Otro ejemplo de agregación de datos históricos es planteado en Cambra Baseca et al. (2019), en el cual los autores muestran promedios diarios de temperatura y velocidad del viento para los últimos diez meses.

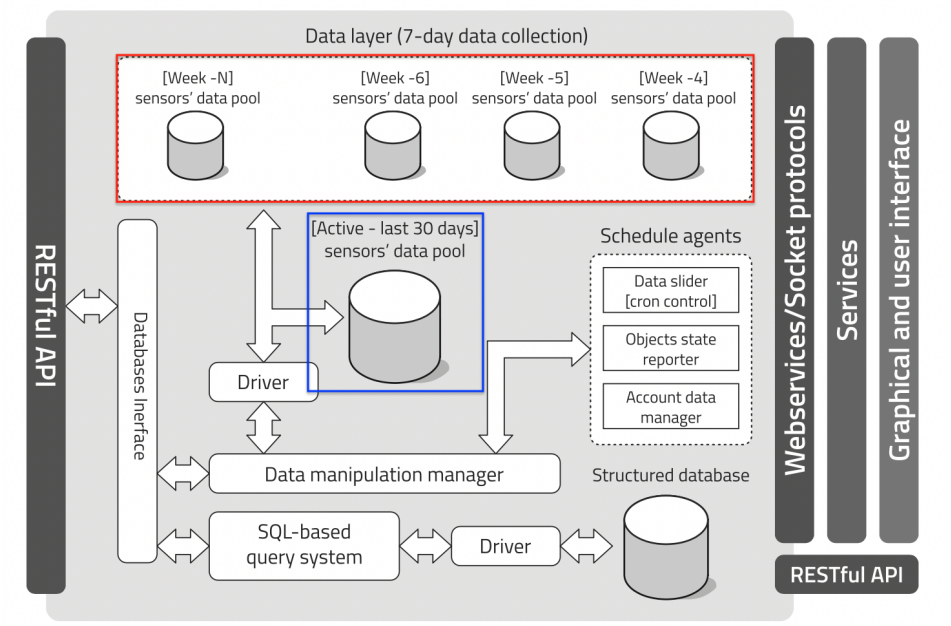


Figura 2.5: Separación de datos para análisis en tiempo real y diferido. En azul se muestra la base de datos destinada a persistir datos recientes, y en rojo se muestran las distintas bases de datos para persistencia histórica semanal. (Morais et al. (2019)).

[25] "El mildiu de la vid es una enfermedad potencialmente muy peligrosa, [...] que afecta a todas las partes verdes de la vid, principalmente hojas, inflorescencias y bayas." – https://www.certiseurope.es/noticias/detalle/news/mildiu-de-la-vid-plasmopara-viticola-sintomas-danos-y-tratamiento

2.2.7. Arquitectura para la gestión de datos

El relevamiento se enfocó principalmente en los aspectos de distribución de los datos; es decir, interesó evaluar si se utilizaron modelos con arquitecturas centralizadas, en las cuales todo el conjunto de datos se guarda en un nodo, o arquitecturas distribuidas, en las cuales un conjunto de nodos comparten la carga de trabajo[26]. Los artículos que hicieron referencia a arquitecturas utilizadas fueron mínimos, por lo que no se pudo evaluar tendencias en esta temática.

En una arquitectura centralizada, el gestor de base de datos es desplegado y opera en su totalidad en una sola instancia. En ella se persiste toda la información requerida para que las aplicaciones puedan funcionar correctamente sin necesidad de consultar otros nodos –ya sea implícita o explícitamente. Esto no quiere decir que no puedan existir otros nodos, pero, en el caso de existir, no se contará con funcionalidades para manejar la distribución de operaciones consistentes de escritura y lectura. Un ejemplo de esto es la replicación asincrónica de datos en *MySQL* y *PostgreSQL*, en la que se cuenta con un nodo primario –en el cual se efectúan todas las escrituras– y uno o más nodos secundarios. En los nodos secundarios se pueden efectuar lecturas, pero debido a la naturaleza asincrónica del envío de datos, no se puede garantizar la consistencia de los mismos –al menos no sin consultar en el nodo primario. Incluso si se forzara el sincronismo en el envío de datos del nodo primario a las réplicas, al tener todos los nodos todo el conjunto de datos se puede decir que es de igual manera un tipo de arquitectura centralizada, dado que estos nodos proveen solamente herramientas de alta disponibilidad y no de partición de datos.

Por otro lado, en una arquitectura distribuida los datos son particionados (o segregados) en los distintos nodos que la componen. En estos casos se debe llevar un control sobre qué datos están en qué nodos, para poder manejarlos eficientemente y así evitar consultas innecesarias en nodos que no contengan los datos requeridos. Si bien puede ser resuelto desde la capa de aplicación, la complejidad asociada al manejo de datos distribuidos llama a la utilización de manejadores que proveen esta funcionalidad de forma nativa, como *MongoDB* o *Cassandra*. Así como en las arquitecturas centralizadas, pueden existir también nodos dedicados a incrementar la disponibilidad mediante replicación; sin embargo, en este caso no necesariamente requiere de nodos adicionales, ya que se puede implementar en los mismos nodos utilizados para las particiones[27].

En Triantafyllou et al. (2019) se menciona el uso de *MySQL* como manejador de base de datos, pero no queda clara la arquitectura utilizada. Se hace referencia a un "servidor principal" en el cual la base de datos está alojada, por lo cual se intuye que es de tipo centralizada. Además, no solo no es común ver entornos de *MySQL* que cuenten con una arquitectura distribuida, sino que cuando se habla de *sharding* en *MySQL* se le llama ex-

[26] Página de documentación de *Oracle* con definiciones y conceptos sobre distintas arquitecturas posibles – https://docs.oracle.com/cd/B10501_01/server.920/a96521/ds_concepts.htm

[27] *Cassandra* es un ejemplo, en el cual se utiliza el "factor de replicación" para determinar cuántas copias de la tupla se deben tener en el *cluster* – https://docs.datastax.com/en/cassandra-oss/3.x/cassandra/architecture/archDataDistributeAbout.html

plícitamente *MySQL NDB Cluster*[28], o se mencionan herramientas específicas diseñadas sobre *MySQL*, como *Vitess*[29].

Como se mencionó anteriormente en la Secc. 2.2.4, *MySQL* es utilizado en conjunto con *MongoDB* en Morais et al. (2019), siendo este un caso de arquitectura distribuida. *MongoDB* persiste los datos de los sensores y *MySQL* persiste metadatos e información de usuarios. Desafortunadamente no se detalla la arquitectura utilizada internamente en *MongoDB*, que al ser un manejador con soporte nativo a *sharding* es, a su vez, naturalmente utilizado en ambientes distribuidos. De particular interés en estos casos es conocer el esquema de datos y las claves (o campos) que se utilizan para efectuar la partición de los datos en los distintos nodos.

Cómputo y almacenamiento en la nube

Si bien los detalles de arquitecturas utilizadas fueron escasos, las menciones a entornos desplegados en la nube abundaron: 23 de los 33 artículos hicieron mención a ello, ya sea siendo explícitamente utilizada en la solución planteada, o en alguna sección del artículo dedicada al estado de la tecnología. Los artículos son los siguientes: 1, 2, 4-7, 9, 11-16, 18-20, 24, 25, 27, 29, 31-33.

La computación en la nube es la disponibilidad a demanda de servidores o servicios remotos accesibles vía red[30]. Existen prestadores de servicios orientados al público en general, así como también entornos de nube privados dedicados a suplir la demanda interna en una organización. Los servicios se pueden discriminar en tres grandes categorías, dependiendo del nivel de abstracción que se ofrece (de menor a mayor): infraestructura como servicio (o *IaaS*, por su sigla en inglés *Infrastructure as a Service*), plataforma como servicio (o *PaaS*, por *Platform as a Service*) y software como servicio (o *SaaS*, por *Software as a Service*). Conforme aumenta el nivel de abstracción, se pierde control sobre las funcionalidades del servidor, pero se gana en facilidad de manejo y mantenimiento –por ejemplo, ya no es necesario saber cómo administrar un servidor de base de datos, el usuario solamente debe entender cómo utilizarlo para guardar y recuperar información. En *IaaS* se tiene acceso al sistema operativo, teniendo la posibilidad de configurarlo e instalar paquetes de software a discreción; en *PaaS* ya no es posible configurar parámetros del sistema operativo, pero se pueden instalar los paquetes de software necesarios para la aplicación final; por último, en *SaaS* ya no es posible configurar el sistema operativo o elegir software en particular, es decir, se tiene acceso a la aplicación final y el único control que se tiene es sobre los datos. En la Fig. 2.6 se muestra a qué nivel se sitúa cada servicio respecto a las capas que componen una aplicación.

[28] Manual de referencia de NDB Cluster – https://dev.mysql.com/doc/refman/8.0/en/mysql-cluster.html

[29] Página principal de *Vitess* – https://vitess.io/

[30] Página de documentación de *Amazon AWS* – https://aws.amazon.com/es/what-is-cloud-computing/

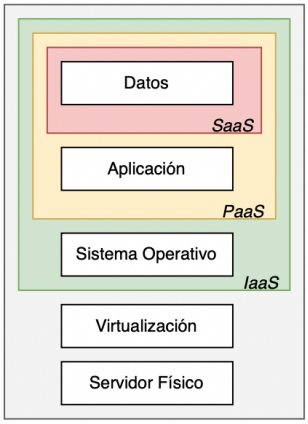


Figura 2.6: Niveles de abstracción en los que operan los distintos servicios prestados en la nube en relación a las capas que componen una aplicación. (Elaboración propia.)

Entre las ventajas de este modelo de servicio se pueden destacar la flexibilidad que brinda al momento de elegir qué solución se adapta mejor al proyecto, la habilidad de escalar fácilmente –e incluso a demanda, y de manera automática–, el acceso a una gran capacidad de cómputo sin necesidad de invertir en *hardware*, el fácil acceso a los datos –en proveedores públicos, el entorno se accede vía internet– y el manejo automático de parches, actualizaciones y respaldos en el caso de servicios de tipo *SaaS*. Por otro lado, algunas de las desventajas son la potencial dependencia del proveedor (*vendor lock-in*), los costos elevados –a mayor capa de abstracción, mayor costo tendrán los servicios–, el alojamiento de datos críticos fuera de la esfera de acción del usuario y la falta de control sobre la infraestructura –en especial en modalidades de tipo *SaaS*. En SRG (2022) se puede ver un reporte sobre los mayores proveedores de servicios en la nube en la actualidad, y en Gartner (2022) se puede ver una predicción sobre la adopción de los servicios para los años 2022 y 2023 –que se estima crecerá globalmente en un 20,3 % y 21,2 % respectivamente. Actualmente, *Amazon Web Services*[31] (*AWS*) lidera el grupo, con un 33 % del valor de mercado, seguido por *Microsoft Azure*[32] con el 21 % y *Google Cloud Platform*[33] (*GCP*) con el 10 %.

Como resultado de la revisión de literatura en Navarro et al. (2020), se mencionan las siguientes plataformas en la nube como las más utilizadas: *ThingSpeak*, *FIWARE*, *Ubidots*, *SmartFarmNet*, *AWS IoT* y *Thinger.io*. En particular, *ThingSpeak*[34] fue la más mencionada, debido a que es una plataforma que cuenta con una aplicación (cliente) de código

[31] Página principal de *AWS* – `https://aws.amazon.com/`
[32] Página principal de *Azure* – `https://azure.microsoft.com/`
[33] Página principal de *GCP* – `https://cloud.google.com/`
[34] Página principal de *ThingSpeak* – `https://thingspeak.com/`

abierto[35] y con bajos requerimientos de infraestructura, que se adapta bien a las necesidades de este tipo de entornos. Se destaca también el uso de *AWS* como plataforma en la que se utiliza una mayor variedad de técnicas de almacenamiento y procesamiento de los datos. Por último, los autores notan un marcado incremento en la mención de entornos desplegados en la nube conforme avanzan los años, mostrando una adopción que tiende a favorecer su uso.

2.2.8. Escenario de aplicación

La esfera de acción de la Agricultura Inteligente comprende una gran cantidad de etapas dentro del ciclo de producción. Desde la preparación de la tierra, la siembra, el cuidado de los cultivos y su posterior cosecha, a la logística involucrada en el almacenamiento, venta y distribución de la producción. Como es de esperar, los trabajos relevados reflejan en gran medida esta realidad, describiendo en muchos casos múltiples etapas a la vez. Como se mencionó en la Secc. 2.2.3, en Saiz-Rubio and Rovira-Más (2020) se presenta un sistema de soporte a la toma de decisiones orientado a mejorar los procesos y productos. En particular, se muestra cómo se puede aplicar a los cultivos, tomando datos de sensores y procesándolos de tal manera que como resultado se generen acciones –los cuales brindarán nuevos datos, retroalimentando el sistema. En la Fig. 2.7 se muestran las distintas etapas de este proceso, que empieza y termina con el cultivo (*crop*). La etapa descrita como "plataforma" (*platform*) está compuesta por los sensores y medios físicos con los que se toman datos de los cultivos, para luego ser recolectados en un repositorio central en la siguiente etapa (*data*). Mediante la aplicación de técnicas de inteligencia artificial, se interpretan los datos y se extrae información que es de utilidad para la toma de decisiones, las cuales son finalmente llevadas a cabo en la etapa de actuación. El proceso se repite hasta la cosecha, cerrando el ciclo de ingesta y procesamiento de datos.

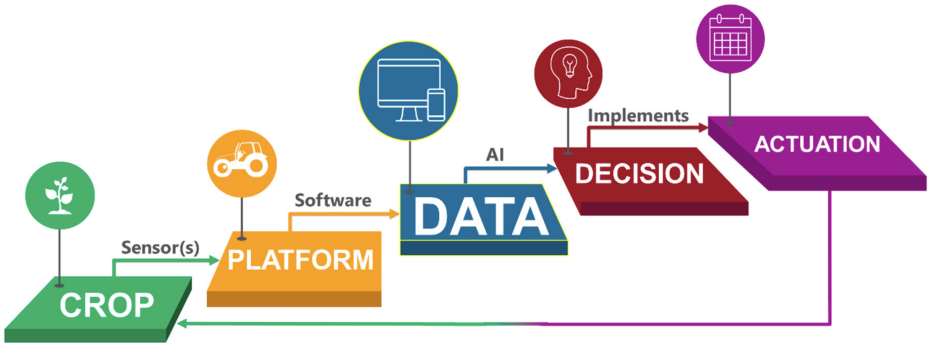


Figura 2.7: Etapas del ciclo de administración de cultivos en Agricultura Inteligente. (Saiz-Rubio and Rovira-Más (2020))

[35] Código fuente del cliente *ThingSpeak* – https://github.com/iobridge/thingspeak

Dentro de la monitorización de los cultivos existen varias categorías de datos de las cuales se puede extraer información. En el Cuad. 2.8 se muestra una lista no exhaustiva de categorías de datos y áreas de interés mencionadas por artículo. Se puede concluir que las más populares son las concernientes a cultivos y suelos, destacándose particularmente las orientadas al manejo de enfermedades, pestes e irrigación, y a la detección de hierbas y rendimiento de la cosecha.

Área de monitorización		**Artículo**
Categoría	**Subcategoría**	
Clima	Humedad	9, 10, 12, 15, 31
	Luz solar	9, 31
	Precipitaciones	31
	Temperatura	9, 10, 12, 15, 19, 31
	Velocidad del viento	10, 19
	No especifica	7, 25
Maquinaria	No especifica	7, 17
Planta	Manejo de enfermedades	10, 11, 16, 17, 21, 24, 25, 26
	Manejo de fertilizantes	7, 11, 12, 15, 16, 21, 24, 25, 26
	Conteo	26
	Control de plagas	1, 7, 10, 21, 30
	Evapotranspiración	16
	Modelado 3D	19, 26, 30
	Niveles de biomasa	21, 23, 24, 26
	Polinización artificial	30
	Rendimiento de la cosecha	7, 11, 15, 16, 19, 23, 25, 28
	Salud	7
	No especifica	17, 23, 25, 30
Suelo	Manejo de riego	7, 9, 11, 16, 17, 19, 23, 24, 25, 30
	Conductividad eléctrica	19
	Detección de hierbas	11, 15, 16, 19, 23, 24, 26, 28
	Humedad	9, 15, 16, 19, 21, 24, 26, 31
	Nivel de pH	12, 15, 19
	Niveles de nutrientes	10
	No especifica	7, 17, 24
Trazabilidad	No especifica	17

Cuadro 2.8: Dentro de la monitorización de variables en los cultivos, existe una gran variedad de categorías de las cuales extraer información. En este cuadro se muestra una lista no exhaustiva de potenciales casos de uso, y los artículos que hacen referencia a ellos.

2.2.9. Tipos de licencias

En lo que a licencias de *software* respecta, el paisaje está dominado por proyectos propietarios de código cerrado. Si bien se encuentran menciones a trabajos que utilizan de manera subyacente herramientas de código abierto (*open source*), los detalles de las implementaciones en particular no son –en general– compartidos. El *software* libre y de código abierto (*FOSS*, por su sigla en inglés *Free and open-source software*) hace posible el acceso universal a las herramientas, y facilita la reproducción y validación de los sistemas desarrollados dado que, a diferencia del código propietario, es en general accesible gratuita y públicamente. No todo el *software* libre es de código abierto, y viceversa, y existe además una plétora de licencias en las cuales un proyecto se puede basar. Es necesario estudiar cuidadosamente los términos de cada una al momento de elegir una herramienta para un proyecto, dado que algunas licencias imponen restricciones en los términos que rigen los trabajos derivados que las utilizan. Un gran impulsor detrás de la adopción de *software* de código abierto fue la plataforma *GitHub*[36], que provee de manera gratuita la posibilidad de publicar proyectos y herramientas *open source*. Cuenta con funcionalidades que facilitan la interacción entre los usuarios y los desarrolladores, no solamente permitiendo crear reportes de problemas (*bugs*) o peticiones de nuevas funcionalidades (*feature requests*), sino que también de aportar código directamente –mediante los denominados *Pull Requests*. Se vale de la herramienta de control de versiones distribuido *git*[37] para el manejo del código fuente y sus versiones.

Un claro ejemplo de trabajos que utilizan herramientas libres y de código abierto es el descrito en Belcore et al. (2021), el cual es presentado como "flujo de trabajo para agricultura de precisión utilizando herramientas *FOSS*". Si bien se comparte en gran detalle cada componente y procedimiento llevados a cabo, carece de referencias donde se pueda acceder a inspeccionar el código fuente resultante. Los autores remarcan en las conclusiones y trabajos futuros las bondades del código abierto como facilitador de la replicación y validación de un sistema, pero carecen proveer –al menos para el público en general– las herramientas necesarias a tales efectos.

En Saiz-Rubio and Rovira-Más (2020) se listan 36 aplicaciones de *software* para el manejo de información agrícola, de las cuales solamente dos son *open source*: *ADAPT*[38] (de *AgGateway*) y *AgroSense*[39] (de *Corizon*). *ADAPT* es un entorno de trabajo (*framework*) que ofrece interoperabilidad entre aplicaciones basadas en *hardware* o *software* distintos, y puede ser de utilidad para reducir la brecha en sistemas heterogéneos. Por otro lado, *AgroSense* es descrito como una solución para la administración de entornos de agricultura de precisión, en la cual se pueden importar y exportar datos desde y hacia otras herramientas, e incluso compartirlos con los diferentes actores involucrados –sean empleados u otras empresas contratadas. Lamentablemente, parece no estar más disponi-

[36] Página principal de *GitHub* – https://github.com/
[37] Página principal de *git* – https://git-scm.com/
[38] Página principal de *ADAPT* – https://adaptframework.org/
[39] Página principal de *AgroSense* – https://agrosense.eu/

ble como código abierto, ya que el enlace asociado al proyecto[40] retorna un error de "no encontrado" ("*404 - Repository not found*").

El único artículo cuyo código fuente es accesible es, lamentablemente, el menos relacionado a los objetivos de esta revisión de literatura. En Helmké et al. (2019), los autores desarrollaron un *proxy* de privacidad para dispositivos conectados mediante el protocolo *ISOBUS*. *ISOBUS* es utilizado para la comunicación entre ciertos componentes electrónicos –llamados unidades de control– en maquinaria agrícola, cuya función es encargarse de tareas como el manejo de sensores y actuadores, o el control de la maquinaria en su entorno. Dada su poca relevancia a los propósitos de esta revisión, no se analizó el código fuente con mayor detenimiento. De cualquier manera, y por completitud, se puede acceder al mismo mediante el siguiente enlace: `https://github.com/rhelmke/cant`.

Finalmente, es de destacar que cuando se habla de *open source* se tiende a pensar exclusivamente en *software*, aunque también existe su contraparte en el *hardware*, e incluso en los conjuntos de datos. Los autores en Debauche et al. (2021) mencionan dos *SBCs* en particular que son ampliamente utilizados en entornos *IoT*: *Arduino* y *BeagleBone*. Si bien no es común ver modificaciones de componentes de *hardware* en la literatura, es destacable que existan proyectos que permitan al usuario tener el control y las libertades provistas por las licencias abiertas. Por último, los autores en Moore et al. (2021) describen la importancia de contar con *datasets* abiertos, tanto para los productores como para los investigadores, y mencionan cómo instituciones públicas y privadas pueden aportar valor mediante la liberación de los mismos.

[40] Enlace al código fuente de *AgroSense* – `https://bitbucket.org/corizon/agrosense/`

3 Caso de estudio

Como ejemplo de caso de estudio se desarrolló una interacción entre un nodo robótico que captura datos y los guarda en una base de datos remota para su posterior procesamiento. El nodo robótico es representado por un drone, y la base de datos remota por una instancia *PostgreSQL*. Como resultado de la exploración del drone, se generan imágenes que podrán ser utilizadas para evaluar el estado de la plantación en un momento dado. Además, se generan nuevas imágenes a partir del cálculo de índices de vegetación, y se comparan los resultados obtenidos.

El desarrollo del caso de uso tiene como objetivo principal validar empíricamente los conocimientos adquiridos mediante la revisión de la literatura, y como objetivo secundario el de proveer un prototipo funcional que pueda servir como referencia en futuros trabajos. Para ello se optó por utilizar un ambiente simulado, que cuenta con numerosos beneficios sobre experiencias de otra índole, en las cuales se debe tener acceso no solo a la infraestructura necesaria –como ser un dron con una cámara *RGB*–, sino que también al entorno del cual extraer información –una parcela de tierra con cultivos. Otros beneficios son un menor costo de implementación, la libertad de ejecutar pruebas sin temor a ocasionar daños por mal funcionamiento o errores, la reproducibilidad y la de contar con una herramienta que puede ser utilizada por terceros en su propio código (o incluso para duplicar la experiencia aquí documentada y realizar una revisión de la misma). Por otra parte, las desventajas vienen dadas en su mayoría por la calidad de los datos y del modelo utilizados. La validez de los resultados en experiencias análogas no simuladas estará sujeta a qué tan precisamente se haya modelado la realidad en la simulación.

Para el desarrollo del caso de estudio se optó por utilizar *Robot Operating System* (*ROS*[1]) como entorno de trabajo de aplicaciones robóticas. Es un buen entorno para trabajar con prototipos y simulaciones dado que, como se verá más adelante, cuenta con herramientas especialmente desarrolladas para tales fines. *ROS* es un conjunto de bibliotecas y aplicaciones de software libre diseñado para uso en entornos ciberfísicos –tanto simulados como físicos–, proveyendo una arquitectura modular confiable y escalable. En particular, para la simulación y control del dron se utilizan *ArduPilot* y *QGroundControl*, y para la visualización del modelo 3D *Gazebo*. *PostgreSQL* oficia de motor de base de datos, y dos módulos se escriben en *Python*, uno para consumir las imágenes de la cámara del dron e insertarlas en la base de datos y el otro para el posprocesamiento y generación de nueva información a partir del cálculo de índices de vegetación. En la Fig. 3.1 se puede ver un

[1] Página principal de la documentación en línea de *ROS* – https://docs.ros.org/

diagrama que muestra las distintas interacciones entre estas herramientas.

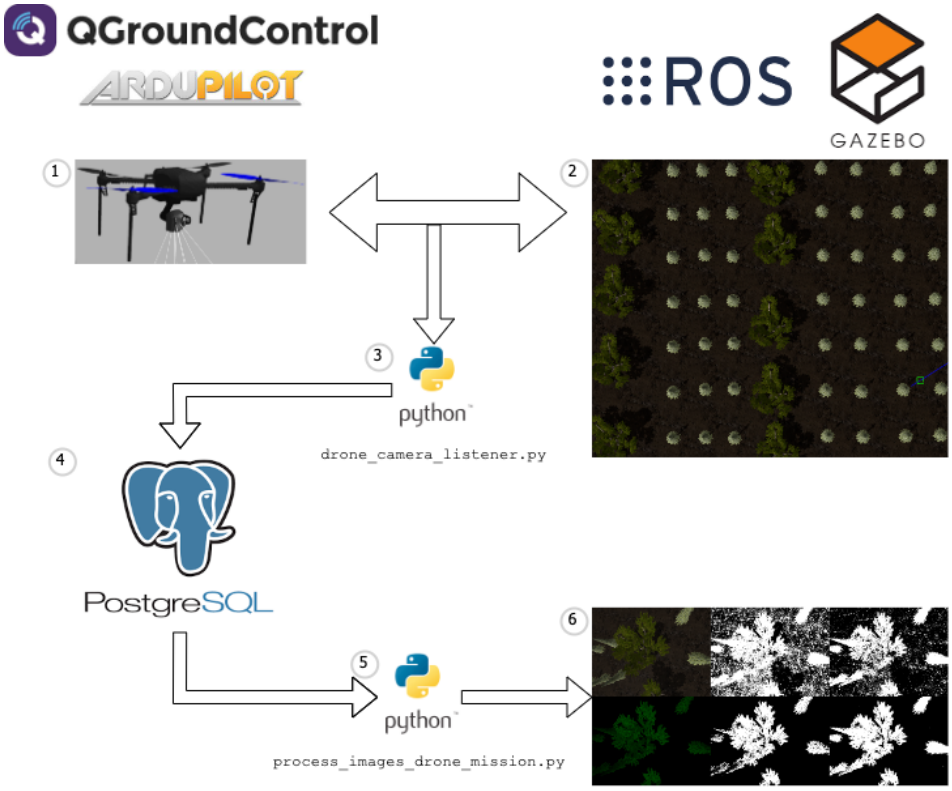


Figura 3.1: Diagrama de interacciones entre las distintas herramientas que componen la prueba de concepto desarrollada. Desde el dron (1) y mapa (2) simulados, al módulo *Python* que consume las imágenes (3) y las guarda en la base de datos (4), al módulo *Python* que posprocesa las imágenes (5) para generar la imagen resultante del cálculo de los índices de vegetación (6).

El resto de este capítulo está compuesto de la siguiente manera. En la Secc. 3.1 se presentan los distintos tipos de vehículos aéreos no tripulados disponibles en la actualidad, y en la Secc. 3.2 se discuten en mayor profundidad los índices de vegetación y sus potenciales usos. Finalmente, en la Secc. 3.3 se detallan las particularidades de la implementación: los paquetes de *software* instalados, las configuraciones utilizadas y las pruebas realizadas.

3.1. Vehículos aéreos no tripulados

Fue de interés relevar datos sobre sistemas ciberfísicos que utilizaran vehículos aéreos no tripulados (drones o *UAVs*, por su sigla en inglés *Unmanned Aereal Vehicles*) para la captura de datos. En particular, se hizo hincapié en artículos que presentaran casos de uso utilizando *UAVs*, dado que existe una gran variedad, y sus entornos de aplicación son muy diversos.

A grandes rasgos, se pueden clasificar en función de su uso como militares o civiles, o en función de su método de control como autónomos, monitorizados, supervisados, preprogramados o controlados remotamente (ver Unidad Reguladora y de Control de Datos Personales (2018)). Además, se pueden categorizar por las características de vuelo, siendo dirigibles, planeadores, de ala fija, de alas giratorias, o híbridos. Un dirigible es un aeronave que se vale de gases de menor densidad que los que componen la atmósfera –como el hidrógeno o el helio– para lograr la sustentación aerostática, y cuenta con un timón y propulsores para controlar su traslación. El resto son aeronaves de sustentación aerodinámica (o aerodinos), debido a que son más densos que el aire circundante y se valen de sus propios medios para la sustentación y traslación en el aire. Los planeadores carecen de motor, por lo que necesitan ayuda de otros medios para el despegue, y una vez en el aire dependen de corrientes de aire para poder operar. Finalmente, tanto las aeronaves de ala fija como las giratorias –y las híbridas– son completamente independientes en ambos sentidos, proveyendo sus propios medios tanto para la sustentación como para la traslación. Estas últimas son las plataformas más utilizadas dentro de ambientes de Agricultura Inteligente, como se menciona en Hassler and Baysal-Gurel (2019). A su vez, dentro de las aeronaves de alas giratorias, se puede diferenciar respecto a la cantidad de hélices: dos (helicóptero), tres (tricóptero), cuatro (cuadricóptero), seis (hexacóptero) y ocho (octocóptero) son las configuraciones más comunes. Otras categorizaciones posibles son: velocidad de desplazamiento, carga útil, autonomía, costo, tamaño y peso.

(a) (b) (c)

Figura 3.2: Distintos tipos de aeronaves de alas giratorias (o drones): en 3.2a se muestra un cuadricóptero, en 3.2b un hexacóptero y en 3.2c un octocóptero.

Las aeronaves de ala fija tienden a desplazarse más rápidamente, y pueden cubrir más área que las de alas giratorias, pero no son capaces de desarrollar tareas de precisión o de permanecer estacionarias en un lugar fijo sobre el terreno. Generalmente precisan de una pista de aterrizaje y despegue, por lo que las potenciales áreas de aplicación son diferentes en unas u otras. Las de ala fija se adaptan mejor a tareas como mapeo de grandes

superficies, o de aplicación de fertilizantes, pesticidas o riego a gran escala. Por otro lado, las de alas rotatorias están mejor adaptadas a trabajos en áreas más reducidas, en las que se desea tener un mayor control, como ser: detección de estrés en plantas, estimación de nutrientes y biomasa, manejo de plagas, polinización y aplicaciones de precisión (como aplicación localizada de pesticidas –por planta). Inclusive dentro de la categoría de alas giratorias, existen diferencias entre las posibles aplicaciones dependiendo de la cantidad de rotores y su tamaño.

Entre los tipos de sensores disponibles se encuentran los de localización, de imágenes –radiación electromagnética visible, multiespectrales e hiperespectrales–, térmicas y de distancia o profundidad; y entre los mecanismos agregados se encuentran los de aplicación de productos, irrigación y de manipulación.

Los sensores de localización se encuentran en todos los vehículos descritos en la literatura, y son parte fundamental de entornos en los cuales se depende de la geolocalización para llevar a cabo las tareas –delimitar áreas de trabajo, predefinir caminos a recorrer o habilitar puntos de "retorno al origen"–, o para poder utilizarla como metadatos en etapas de procesamiento posteriores –para agregar información georreferencial a imágenes, por ejemplo. Típicamente se hace uso de un sistema global de navegación por satélite (o *GNSS* por su sigla en inglés *Global Navigation Satellite System*), siendo el *GPS* uno de los más utilizados. El sistema *GPS* (por su sigla en inglés *Global Positioning System*) fue desarrollado por Estados Unidos y puesto en funcionamiento en 1994, y consta de 32 satélites. Existen otros sistemas, también populares, como el *GLONASS* (por su sigla en ruso *Global'naya Navigatsionnaya Sputnikovaya Sistema*), desarrollado por Rusia y desplegado en 1995, que consta de 24 satélites. BeiDou, desarrollado por China, está en funcionamiento desde 2018 y consta de 35 satélites. Y finalmente Galileo (compatible con *GPS*), desarrollado por la Unión Europea, está en funcionamiento desde 2014 y consta de 24 satélites.

Las cámaras utilizadas pueden ser de distintos tipos, siendo las más comunes aquellas que captan imágenes en el espectro de luz visible –también conocidas como cámaras *RGB*– y en espectros con longitudes de onda inferiores y superiores –conocidas como cámaras multi e hiperespectrales. Las cámaras *RGB* almacenan datos de cada color –rojo, verde y azul– para cada pixel de la imagen. Tienen la bondad de ser más accesibles, de proveer imágenes de alta calidad sin necesidad de configuraciones complejas, y son livianas y sencillas de operar. En contraposición, los puntos negativos son la falta de datos para generar índices de vegetación adecuados –ya que muchos de ellos precisan, además, datos del espectro no visible– y la variabilidad de las imágenes dependiendo de las condiciones climáticas o la hora del día en las cuales fueron tomadas. Por otro lado, las cámaras multi e hiperespectrales cuentan con sensores capaces de detectar luz en espectros infrarrojos y ultravioletas, siendo la mayor diferencia entre estos dos tipos cómo se capta y presenta la información. En las cámaras multiespectrales, las bandas espectrales son más anchas que en las hiperespectrales, agrupando más datos en menos puntos de referencia. Como se menciona en Lu et al. (2019) y Lucieer et al. (2014), en las cámaras multiespectrales es común tener un grupo limitado de bandas –agrupando

decenas o centenas de nanómetros por banda– e incluso haber brechas entre las mismas. En cambio, en las cámaras hiperespectrales se tienen cientos de bandas –teniendo cada banda a lo sumo decenas de nanómetros– logrando una mayor precisión y continuidad en la información registrada. En la Fig. 3.3 se puede ver un ejemplo de esta agrupación por bandas, y cómo las cámaras hiperespectrales pueden dar una noción más clara de continuidad, en contraposición a los grupos de datos más discretos de las multiespectrales. Las cámaras hiperespectrales son más complejas de configurar, y al generar una gama más amplia de datos el posprocesamiento será también más demandante: como mencionan los autores en Tsouros et al. (2019), las imágenes resultantes se deben calibrar y procesar para aplicar correcciones y mejoras antes de ser utilizadas. Como se verá más adelante, no siempre es necesario contar con información tan granular, ya que algunos índices de vegetación sumamente eficientes pueden ser computados con unas pocas bandas.

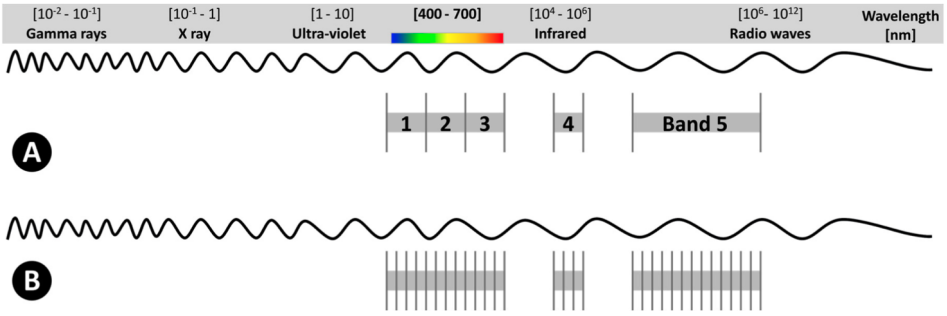


Figura 3.3: Representación del espectro de luz: (A) Ejemplo multiespectral utilizando cinco bandas; y (B) ejemplo hiperespectral utilizando mayor cantidad de bandas (usualmente se extiende a los cientos de bandas). (Adão et al. (2017))

Además, se puede obtener datos de cámaras termales y sensores de proximidad. Las cámaras termales son casos particulares de cámaras hiperespectrales, capturando información de espectros comprendidos entre 3.000 y 15.000 nm de longitud de onda. Su funcionamiento se basa en que todo objeto que tenga una temperatura superior al cero absoluto emana energía de manera proporcional a la misma. Generalmente, los datos termales son utilizados para medir el estrés hídrico del suelo y la evapotranspiración de las plantas, aunque también tienen aplicaciones de detección de plagas y enfermedades, y predicción de rendimiento. Finalmente, los sensores de proximidad (o *LiDARs*, por su sigla en inglés *Light Detection and Ranging*) se valen de pulsos láser para medir distancias. Se utilizan para trazar mapas y modelar parcelas de tierra en 3D, y como ayuda en operaciones de vuelo para detectar obstáculos, evitar colisiones, y regular la altura de la aeronave en misiones de aplicación de fertilizantes, pesticidas o riego. En Belcore et al. (2021) se puede ver, a modo de ejemplo, el resultado de utilizar un *LiDAR* para generar una imagen de nube de puntos en 3D.

Las ventajas de contar con aeronaves no tripuladas en entornos de Agricultura Inteligente son numerosas. En contraposición a aeronaves tripuladas de mayor porte –como ser

helicópteros o avionetas– los drones son más accesibles al público en general, siendo su costo de compra, operación y mantenimiento órdenes de magnitud menor. Esto permite no solo que los productores puedan asumir el costo como parte de su operativa propia, sino que haya menos barreras de entrada a potenciales proveedores de servicios o trabajos de investigación. A su vez, con una mayor utilización, el ecosistema entorno a sistemas de Agricultura Inteligente basados en *UAVs* se retroalimenta, generando nuevas oportunidades, aplicaciones y demanda en general. Además, introduce nuevas dimensiones en los datos disponibles, como ser mayor resolucion temporal –ya no se depende de que el satélite orbite sobre el mismo punto para tomar mediciones– y espacial –un dron puede efectuar vuelos rasantes o incluso permanecer estacionario mientras se capturan los datos, generando imágenes con resoluciones en el orden de los milímetros por pixel–, y tiene una mayor tolerancia a interferencia por parte de nubes. Dado que son altamente personalizables, se puede decidir qué tipos de cámaras o sensores se utilizan en cada vuelo, ofreciendo mayor versatilidad en las funcionalidades y los tipos de datos que capturan. Los satélites en particular cuentan con un gran número de sensores, pero no siempre son datos de público acceso, o pueden incluso contar con ciertas restricciones de uso –como ser ancho de banda o número de imágenes descargadas por unidad de tiempo– o costos asociados a licencias. También se tiene control sobre los caminos a seguir en cada vuelo, pudiendo trazar recorridos automatizados mediante coordenadas *GPS*, lo que permite hacer un seguimiento preciso de una parcela de tierra en el tiempo para monitorizar la evolución de los cultivos. En caso de ser necesario, se puede incluso interactuar con los cultivos mediante modificaciones que permitan, por ejemplo, irrigar o esparcir agroquímicos, o en casos más complejos recolectar frutos (ver Hassler and Baysal-Gurel (2019)). Por último, en el caso de aeronaves de alas rotatorias, no es necesario contar con una pista de despegue y aterrizaje, minimizando la infraestructura necesaria para su operación.

Naturalmente, existen también desventajas vinculadas a la utilización de *UAVs*. Cada país tiene sus propias regulaciones en lo que refiere a operar aeronaves no tripuladas: desde el tamaño, la altura máxima y zonas permitidas para el vuelo, a los tipos de licencias necesarias y costos asociados a permisos. Al ser un ámbito relativamente nuevo, es de esperar que las políticas y leyes asociadas estén sujetas a modificaciones en el corto y mediano plazo, por lo que se requiere un trabajo adicional de seguimiento de las mismas. Las cámaras multi e hiperespectrales y demás sensores necesarios pueden tener costos elevados, dependiendo qué datos se desee capturar y con qué precisión. Además, si no se cuenta con personal capacitado capaz de operar y mantener la infraestructura, se deben subcontratar empresas especializadas, con lo que se puede incurrir en problemas de gobernanza de datos y dependencia de proveedor. Por último, existen limitaciones físicas en el desempeño de los drones, tales como la vida útil de la batería, la autonomía de vuelo, la carga con la que puede operar la aeronave y el poder de cómputo disponible para realizar tareas en paralelo –como ser evasión de obstáculos, o cálculos dinámicos de índices de vegetación sobre los datos recolectados. En general, se deberá balancear las funcionalidades disponibles con la autonomía de vuelo, buscando maximizar el desempeño sin sacrificar la toma de datos necesarios. En Matese et al. (2015) se menciona que el rango efectivo de aplicaciones que utilicen *UAVs* es entre 5 y 50 hectáreas, dado que en áreas mayores la

utilización de imágenes satelitales provee un costo/beneficio que permanece competitivo.

Cabe destacar los trabajos en Radoglou-Grammatikis et al. (2020) y Kim et al. (2019), en los cuales se puede ahondar en la utilización de vehículos aéreos no tripulados en entornos de Agricultura Inteligente. En ellos, los autores describen en mayor profundidad tanto los distintos tipos de vehículos disponibles, como las aplicaciones en las que fueron utilizados, proveyendo una buena guía del estado del arte de los mismos.

3.2. Índices de vegetación

Los índices de vegetación son valores numéricos calculados en base al reflejo de la radiación electromagnética por parte de los cultivos, y suelen ser utilizados para monitorizar la salud y el desarrollo de los mismos. Se valen de fórmulas matemáticas que toman valores de algunas bandas del espectro electromagnético para definir de manera cuantitativa cualidades de la planta como: propiedades físicas o bioquímicas –biomasa o contenido de clorofila–, efectos del entorno –escasez de agua o nutrientes– o enfermedades. Son calculados a partir de los datos recabados por las cámaras y sensores, y son típicamente computados en una etapa posterior a la de la recolección[2]. Existe una gran variedad de índices generados a partir de una gran variedad de fuentes, los cuales se pueden incluso combinar para generar nuevos índices. En el Cuad. 3.1 se puede ver un breve resumen de algunos de ellos; además, en Sishodia et al. (2020) se puede ver una tabla con más índices de vegetación disponibles, en donde se muestran las fórmulas para calcularlos y sus principales aplicaciones en entornos de Agricultura Inteligente. Las bandas más utilizadas son: azul (*B*) –430-500 nm–, verde (*G*) –500-560 nm–, rojo (*R*) –620-670 nm–, *red edge (RE)* –670-720 nm– e infrarrojo cercano (*NIR)* –720-1.500 nm–, siendo suficiente una cámara multiespectral de cinco bandas, como se presenta en la Fig. 3.4b.

Algunos ejemplos de índices basados en imágenes de cámaras *RGB* son: *Normalized Difference Index (NDI)*, *Excess Red (ExR)*, *Excess Green (ExG)* y *Excess Blue (ExB)*. El *NDI* se vale de las bandas *G* y *R* para segregar las plantas del suelo y demás ruido, y el *ExG* utiliza las tres bandas para el cálculo, basándose en que las plantas reflejan en mayor grado la luz en el espectro verde. En Lottes et al. (2017) se puede ver como los autores utilizan el índice *ExG* para detectar y clasificar vegetación cuando no se cuenta con información multiespectral, y en Hassler and Baysal-Gurel (2019) se muestra la imagen fuente y el resultado de aplicarle los índices *Ex*. Si bien existen aplicaciones para estos índices, los más utilizados son los calculados a partir de información multi o hiperespectral. Algunos ejemplos son: *Normalized Difference Vegetation Index (NDVI)*, *Ratio Vegetation Index (RVI)*, *Normalized Difference Red Edge Index (NDRE)*, *Green Normalized Difference Vegetation Index (GNDVI)*, *Soil Adjusted Vegetation Index (SAVI)*, *Canopy-Chlorophyll Content Index (CCCI)*, *Leaf Area Index (LAI)* y *Crop Water Stress Index (CWSI)*. Muchos

[2]Existen dispositivos que tienen la funcionalidad de computar ciertos índices de manera dinámica. Un ejemplo de ello es el drone *DJI P4 Multispectral* (ver Fig. 3.4), que tiene la habilidad de mostrar imágenes con valores *NDVI* calculados durante el vuelo.

(a) (b)

Figura 3.4: En 3.4a se puede ver un ejemplo de drone con soporte a captura de imágenes multiespectrales: el *DJI P4 Multispectral*. En 3.4b se muestra la cámara multiespectral en mayor detalle, describiendo las bandas de luz y los rangos en los que captura imágenes.

de estos índices se basan en el mismo concepto de normalización, utilizando dos bandas y computándolos como el cociente entre su diferencia y su suma, resultando en valores comprendidos entre -1 y 1:

$$\frac{Banda_1 - Banda_2}{Banda_1 + Banda_2}$$

Por ejemplo, el índice *NDVI* utiliza el infrarrojo cercano (*NIR*) como primer banda y el rojo como segunda banda, y se basa en el hecho de que una planta saludable refleja luz en grandes cantidades en la banda *NIR* y absorbe luz –debido a la clorofila– en la banda *R*. Por otro lado, una planta estresada o enferma refleja en menor cantidad luz en la banda (*NIR*). En la Fig. 3.5 se muestra un ejemplo ilustrado de cómo una hoja en diferentes condiciones de salud refleja las diferentes bandas de luz en los espectros visibles e infrarrojo cercano. En la escala de *NDVI*, valores negativos indican la presencia de un cuerpo de agua, valores cercanos a cero indican suelos sin cultivos –o cultivos muertos–, y valores cercanos a uno indican cultivos saludables. En Cambra Baseca et al. (2019), Martínez et al. (2020), Belcore et al. (2021) y Kim et al. (2019) se pueden ver ejemplos de trabajos en donde se generan mapas con información proveniente de índices *NDVI* para seguimiento del crecimiento y la salud de distintos tipos de plantaciones, y en la Fig. 3.4 se puede ver un dron que tiene la funcionalidad de no solamente calcular valores *NDVI* de manera dinámica, sino también de desplegarlos en vivo en la interfaz gráfica, ayudando a la toma de decisiones durante el vuelo. En Islam et al. (2021) se incluye una tabla con una gran variedad de índices de vegetación utilizados en entornos de Agricultura Inteligente.

Índice de vegetación	**Fórmula**
Excess Greenness Index (ExG)	$2 * G - R - B$
Normalized Difference Index (NDI)	$\frac{G-R}{G+R}$
Ratio Vegetation Index (RVI)	$\frac{NIR}{R}$
Normalized Difference Vegetation Index (NDVI)	$\frac{NIR-R}{NIR+R}$
Normalized Difference Red Edge Index (NDRE)	$\frac{NIR-RE}{NIR+RE}$
Green Normalized Difference Vegetation Index (GNDVI)	$\frac{NIR-G}{NIR+G}$

Cuadro 3.1: Algunos de los índices de vegetación más utilizados y sus fórmulas. (Tsouros et al. (2019))

3.3. Desarrollo del caso de uso

El caso de uso se desarrolla entorno a una plantación de pinos y robles –modelos 3D que se importaron fácilmente, tomando como referencia proyectos preexistentes–, sobre la cual se sobrevuela un dron que captura imágenes. La misión del dron es efectuar un vuelo de cubrimiento a 10 metros de altura, a la vez que envía información a la estación de control situada en las inmediaciones del predio. Luego de terminar la misión, el dron se dirige al punto de salida y aterriza. La recorrida del dron se planifica con antelación, y se carga el plan de vuelo desde la estación de control. Al finalizar la captura de datos, se procesan las imágenes guardadas en la base de datos para generar nueva información a partir del cálculo del índice de vegetación *Excess Greenness* (*ExG*). Con esta nueva información se podrá, por ejemplo, obtener la cantidad de árboles en el predio delimitado por la misión de vuelo.

Las siguientes secciones se componen de la siguiente manera. En la Secc. 3.3.1 se describe el entorno de trabajo utilizado, y en la Secc. 3.3.2 se muestra cómo se instalaron y configuraron las herramientas que lo componen. En la Secc. 3.3.4 se detallan las pruebas funcionales realizadas y, finalmente, en la Secc. 3.3.5 se muestran los resultados de las pruebas de rendimiento.

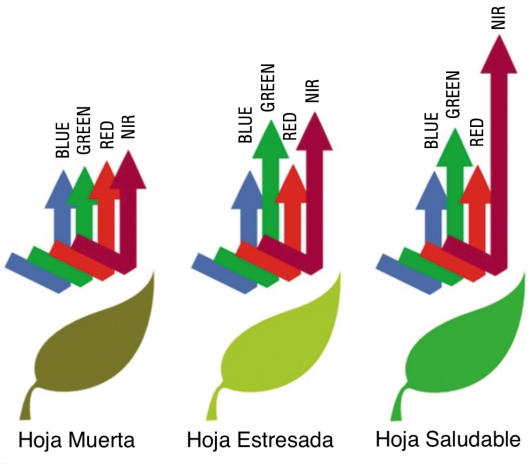


Figura 3.5: Diferencias en el reflejo de radiación electromagnética (luz) en las bandas *RGB* y *NIR* por hojas muertas, estresadas y saludables. El índice de vegetación *NDVI* se basa en este principio. (Sylvester (2018))

3.3.1. Entorno de trabajo

Entorno robótico

Como principal componente de *software* se decidió utilizar el marco de trabajo robótico de código abierto *Robot Operating System (ROS)*. Existen actualmente dos versiones mayores de *ROS*: *ROS 1* y *ROS 2*, cada una con sus respectivas versiones menores, a las que se les asigna un nombre código cuya primer letra es tomada en orden alfabético creciente. Si bien *ROS 2* es recomendada (ver Macenski et al. (2022)) se decidió utilizar *ROS 1 (Noetic Ninjemys)*, ya que se cuenta con experiencia previa trabajando en esta versión del entorno. *ROS* se puede instalar tanto en *Ubuntu* como *Debian*, y de manera experimental (no enteramente soportado) en *Windows* y *Arch Linux*. Aunque actualmente *Ubuntu 22.04 (Jammy)* es la versión más actual, *Ubuntu 20.04 (Focal)* sigue siendo la versión recomendada en la documentación de *ROS 1*[3], y fue por ende la versión utilizada en este proyecto.

En *ROS*, los distintos módulos se comunican entre sí mediante la publicación de mensajes en tópicos (*topics*), siguiendo el modelo publicador-suscriptor. En este contexto, un módulo puede ser cualquier componente: desde una cámara –ya sea simulada o real– a un programa diseñado para procesar datos de varios tópicos y publicarlos en uno nuevo. Cuando un módulo genera información, la publica en una cola de mensajes previamente definida, con un formato de mensaje bien establecido –e inmutable. Los módulos que

[3]Guía de instalación de *ROS 1 (Noetic)* – https://wiki.ros.org/noetic/Installation

quieran consumir esa información se suscriben al tópico, y reciben estos mensajes de manera asíncrona mediante la definición de funciones de retrollamada (*callback*). Los mensajes pueden ser definidos como simples tipos atómicos de datos –como ser números reales en punto flotante de 64 bits–, como tipos complejos de datos –como ser las coordenadas en el espacio de un objeto– e incluso como tipos de datos compuestos por un conjunto de otros tipos de datos –como ser la posición de un objeto, dada por sus coordenadas y su rotación. Además, *ROS* brinda soporte para la definición de nuevos tipos de datos, de manera que cada aplicación puede extenderlos acorde a sus necesidades.

Para la simulación del dron se utilizó la componente de *software in the loop (SITL)* de *Ardupilot*[4], la cual despliega varias consolas en pantalla con el estado de los distintos sistemas simulados (*GPS*, batería, etc.). En la Fig. 3.6 se muestran las distintas consolas utilizadas: a la izquierda se ven las lanzadas por *Ardupilot* para mostrar el estado de la simulación, y a la derecha se ve la terminal utilizada para ejecutarla, en la cual se pueden ingresar comandos para interactuar con el dron –cual si fuera una estación de control.

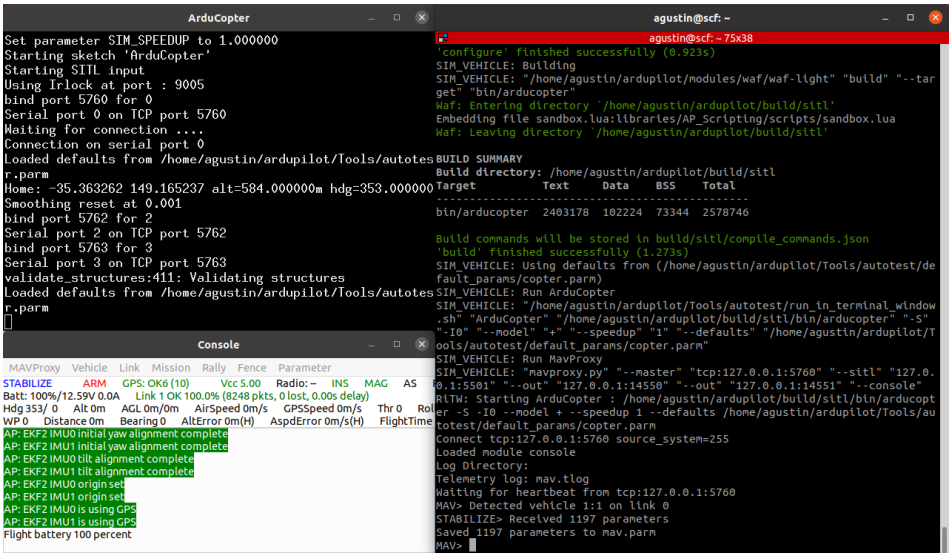


Figura 3.6: Consolas desplegadas luego de ejecutar la simulación *SITL* de *Ardupilot*.

Si bien se puede interactuar con el dron mediante la terminal, es provechoso contar con una herramienta diseñada específicamente a tales efectos, es decir, una estación de control (o *GCS*, por su sigla en inglés *Ground Control Station*). Para ello se utilizó *QGroundControl*, una estación de control compatible con drones que soporten el protocolo de mensajería *MAVLink*[5]. *QGroundControl* ofrece una interfaz sencilla e intuitiva, y todas las fun-

[4]Documentación de la componente de *SITL* de *Ardupilot* – https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

[5]Página principal de *MAVLink* – https://mavlink.io/en/

cionalidades necesarias tanto para controlar el dron manualmente como para crear planes de vuelo autónomo. En la Fig. 3.7a se puede ver su funcionamiento como controlador manual de vuelo, y en 3.7b el editor de planes de vuelo autónomo. En este caso se muestra cómo se puede delimitar un rectángulo manualmente para que la herramienta planifique un vuelo que lo cubra –acorde a ciertos parámetros de distancia entre caminos y altura especificados. Además, en 3.7a se muestran estadísticas e información del dron como: carga de la batería (95 %), conteo de satélites *GPS* disponibles (10) y su precisión (1.2 según el índice *HDOP*[6]), altura (10.0 m), velocidad de traslación (17.5 km/h), distancia recorrida (57.9 m), tiempo de vuelo (25 segundos), y dirección (Norte), entre otras.

(a) (b)

Figura 3.7: Interfaz gráfica de la estación de control *QGroundControl*. En 3.7a se muestra su funcionamiento de control manual, en el cual se pueden controlar parámetros de vuelo como altura y destino. En 3.7b se muestra la herramienta de creación de planes de vuelo autónomo.

Luego de contar con un dron que puede ser controlado remotamente, es deseable poder visualizarlo en su entorno, para lo que se precisa de un simulador de entornos físicos tridimensionales. Para ello se utilizó *Gazebo*[7], que cuenta con un motor físico capaz de simular ambientes tanto interiores como exteriores, y gráficos que incluyen proyecciones de luz y sombras y renderizado de texturas. Se integra de forma nativa con *ROS* mediante el meta paquete `ros-noetic-gazebo-ros-pkgs`[8], facilitando su instalación y uso. Además, es necesario contar con un *plugin* para la comunicación entre *Gazebo* y *Ardupilot*[9], para lo que se utiliza el módulo *ardupilot_gazebo*[10]. En la Fig. 3.8 se puede ver una simulación de la pista de aterrizaje del aeropuerto mostrado previamente en la Fig. 3.7a,

[6]El índice *HDOP* (por su sigla en inglés *Horizontal Dilution of Precision*) es una medida del error en la medición de las coordenadas *GPS* – http://wiki.gis.com/wiki/index.php/Dilution_of_precision_(GPS)

[7]Página principal del simulador *Gazebo* – https://classic.gazebosim.org/

[8]Página que muestra la integración de *Gazebo* y *ROS* – https://classic.gazebosim.org/tutorials?tut=ros_overview

[9]Página que muestra la integración de *Gazebo* y *Ardupilot* – https://ardupilot.org/dev/docs/using-gazebo-simulator-with-sitl.html

[10]Página del proyecto *ardupilot_gazebo* con documentación y código fuente – https://github.com/khancyr/ardupilot_gazebo

en la cual se logra representar en un entorno tridimensional la información del mapa de dos dimensiones desplegado en *QGroundControl* –si el dron recibe una orden de vuelo, esta será reflejada también en el simulador. En la barra de estado situada por debajo de la renderización se muestran estadísticas de la simulación como duración, factor de tiempo real y cuadros por segundo (*FPS*). El factor de tiempo real es una medida de qué tan bien se puede simular el entorno con los recursos disponibles, siendo un factor de tiempo real con valor 1 el ideal. También se puede ver una proyección saliendo del dron, que representa en este caso las imágenes que están siendo procesadas por la cámara integrada en el mismo. Como se verá más adelante, estas imágenes son publicadas como tópicos, los cuales se pueden consumir fácilmente gracias a la integración de *Gazebo* con *ROS*.

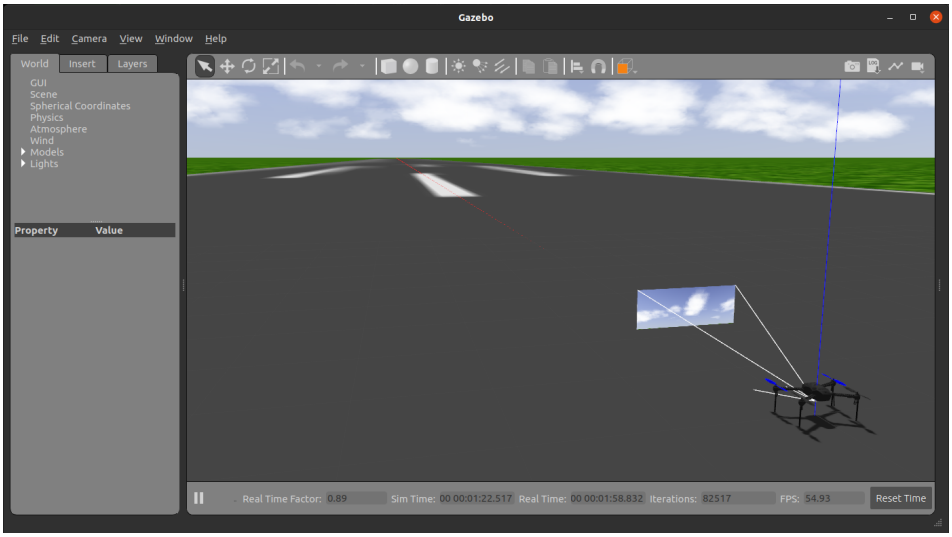


Figura 3.8: Simulación de una pista de aterrizaje en *Gazebo*. La pista representa el aeropuerto mostrado en la Fig. 3.7a.

Finalmente, la comunicación de datos entre el dron y los nodos *ROS* se efectúa mediante los tópicos publicados por un nodo *proxy*: *mavros*[11]. Los datos se podrán utilizar como metadatos asociados a las imágenes provenientes de la cámara, como ser coordenadas *GPS* y altura relativa al suelo.

En lo que al *hardware* respecta, se utilizó una computadora de escritorio con las siguientes especificaciones:

- Placa madre: *MSI* H77MA-G43
- Procesador: *Intel*® *Core*™ i7-3770 CPU @ 3.40GHz (4 núcleos)
- Tarjeta de video: *NVIDIA GF108* (2Gb *RAM*, 96 núcleos *CUDA*)

[11] Documentación del nodo *mavros* – https://wiki.ros.org/mavros

- Memoria: *Kingston HyperX DDR3* 24Gb @ 1333MT/s
- Disco duro: *Intel® SSD 540s Series* (*SATA* 6Gb/s)
- Red: tarjeta de red *onboard, full duplex* de 10Gbit/s

Entorno de base de datos

En Soto et al. (2019) y Amado et al. (2021) los autores describen dos versiones de un dron de vuelo autónomo (Termodron II y III, respectivamente) que utiliza una estación de control desarrollada en una *Raspberry Pi 3*, por lo que se decidió tratar de emular este entorno, como una potencial aplicación de un caso previamente desarrollado. Fue así que la base de datos se instaló sobre una *Raspberry Pi 3*[12], que se encuentra en la misma red local que la computadora de escritorio utilizada para la simulación.

En la base de datos se guarda toda la información perteneciente a las misiones de cada dron, es decir, las imágenes acompañadas de metadatos como ID del dron y la misión, las coordenadas *GPS*, altura y *timestamp –secs* y *nsecs* para segundos y nanosegundos, respectivamente. Cada imagen se guarda como una secuencia de bytes utilizando el tipo de dato `bytea`, y –si bien transparente para el usuario– serán guardadas en tablas secundarias llamadas *TOAST*[13] (por su sigla en inglés *The Oversized-Attribute Storage Technique*). Se agrega un índice para mejorar los tiempos de búsqueda de imágenes para un dron y misión dados, y dado que fue ideada para uso en un ambiente controlado de pruebas, por el momento no se aplican restricciones como claves únicas o primarias. El esquema utilizado se muestra a continuación:

```
gdscf=# \d drone_camera_blob;
                Table "public.drone_camera_blob"
   Column    |       Type       | Collation | Nullable | Default
-------------+------------------+-----------+----------+---------
 drone_id    | integer          |           |          |
 mission_id  | integer          |           |          |
 secs        | integer          |           |          |
 nsecs       | integer          |           |          |
 latitude    | double precision |           |          |
 longitude   | double precision |           |          |
 altitude    | double precision |           |          |
 image       | bytea            |           |          |
Indexes:
    "idx_dcb_1" btree (drone_id, mission_id)
```

[12] Página con especificaciones completas de la *Raspberry Pi 3* – https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/

[13] Página de documentación de *TOAST* en *PostgreSQL* – https://www.postgresql.org/docs/current/storage-toast.html

Se optó por el motor relacional de código abierto *PostgreSQL*[14] para la capa de base de datos, y para el manejo de las coordenadas *GPS* la extensión *PostGIS*[15], que brinda soporte a sistemas de información geográfica. Si bien la versión actual de *PostgreSQL* es la 15, se decidió utilizar la versión disponible en los repositorios, para facilitar la instalación de los distintos componentes. Cabe mencionar que al momento de escribir los datos no es necesario contar con la extensión *PostGIS* instalada, dado que las coordenadas *GPS* se guardan como números en punto flotante. *PostGIS* se utiliza solamente en la etapa de posprocesamiento, que puede incluso ser efectuada en otro nodo luego de haber migrado los datos. Finalmente, para validar los datos geográficos generados por el dron se utilizó el administrador *pgAdmin*[16], que cuenta con una interfaz gráfica capaz de desplegar datos de posicionamiento global en una renderización de mapa en 2D mediante el motor *OpenStreetMap*[17]. En la Fig. 3.9 se puede ver la interacción entre estos componentes para desplegar en pantalla un conjunto de puntos de un vuelo de cubrimiento sobre el aeropuerto simulado.

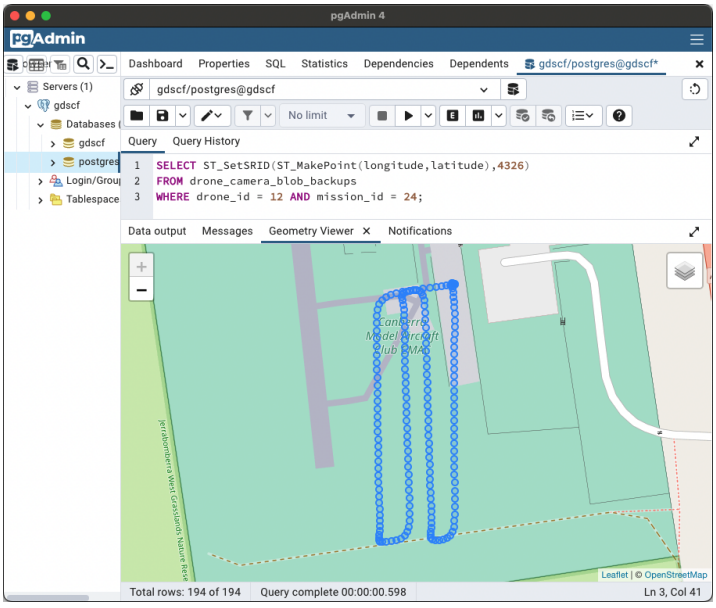


Figura 3.9: Interacción entre *pgAdmin*, *PostgreSQL* y *PostGIS* para desplegar en un mapa un conjunto de puntos de un vuelo de cubrimiento sobre el aeropuerto simulado.

Las especificaciones de la *Raspberry Pi* son las siguientes:

[14]Página principal de *PostgreSQL* – https://www.postgresql.org/
[15]Página principal de *PostGIS* – https://postgis.net/
[16]Página principal de *pgAdmin* – https://www.pgadmin.org/
[17]Página principal de *OpenStreetMap* – https://www.openstreetmap.org

- Modelo: *B Plus Rev 1.3*
- Procesador: *Broadcom BCM2837B0*, *Cortex-A53* (*ARMv8*) @ 1.4GHz
- Memoria: *LPDDR2 SDRAM* 1Gb @ 800MT/s
- Tarjeta micro SD: *Samsung* (*MB-ME32GA/AM*) 32GB (95Mb/s)
- Red: tarjeta de red *onboard, full duplex* de 100Mbit/s

Entorno de monitorización

Por último, se instaló una herramienta de monitorización de recursos para contar con la mayor cantidad de información posible en la etapa de pruebas de rendimiento. Para ello se utilizó la herramienta de código abierto *Percona Monitoring and Managament (PMM)*[18] que es capaz de recolectar métricas tanto del sistema operativo como del gestor de base de datos. Soporta de forma nativa la recolección de datos de *Linux* y *PostgreSQL* (y también de *MySQL* y *MongoDB*) con una resolución de hasta un segundo, por lo que se puede obtener información realmente precisa sobre los sistemas. *PMM* es implementada siguiendo una arquitectura de tipo cliente-servidor, en la cual se tiene un nodo central encargado de recibir y persistir los datos de todas las métricas y consultas, y uno o más clientes que las recolectan localmente. El servidor corre sobre *Docker* –una aplicación de virtualización orientada al manejo de contenedores–, y los clientes cuentan con binarios compatibles con sistemas operativos *Linux* con manejadores de paquetes de tipo *.deb* y *.rpm*. El servidor se ejecutó en una *MacBook Pro* con procesador *Apple M1 Pro*, ubicada también en la red local. Las especificaciones en este caso no son relevantes, dado que es simplemente el nodo de monitorización.

3.3.2. Instalación de herramientas y configuraciones utilizadas

Los comandos utilizados para instalar y configurar las herramientas necesarias se encuentran documentados en el proyecto GitLab accesible en `https://gitlab.fing.edu.uy/agustin.gallego/gdscf`.

Entorno robótico

Los pasos seguidos para instalar gran parte de las herramientas necesarias se encuentran documentados en Johnson (2022). El proyecto segrega entre versiones de *Ubuntu* para algunas herramientas, documentándose para la 18.04 y la 20.04 cuando aplique. Luego de

[18] Página principal de *PMM* – `https://www.percona.com/software/database-tools/percona-monitoring-and-management`

completar los pasos de instalación y configuración, se tendrán las siguientes herramientas disponibles: *Ardupilot*, *MAVProxy*, *QGroundControl*, *Gazebo*, *Ardupilot Gazebo Plugin*, y *ROS*. La única diferencia con los pasos documentados fue la instalación de *mavros* desde el repositorio de *ROS* como paso final.

Una vez se tuvo un entorno de simulación funcional, se procedió a clonar el repositorio inicial de datos[19] y su posterior compilación en *ROS*. El repositorio incluye *scripts* para ejecutar las herramientas y el archivo de mundo necesarios para una simulación básica.

Por último, se instalaron los módulos de *Python* necesarios para la comunicación con la base de datos, el procesamiento de imágenes, la construcción de gráficas y la ejecución de algoritmos de *k-means*[20]. Para ello se utilizaron *psycopg2*, *pil*, *matplotlib* y *scikit-learn*, respectivamente.

Entorno de base de datos

Para instalar *PostgreSQL* en la *Raspberry Pi 3* se utilizaron los paquetes disponibles en los repositorios oficiales de *Ubuntu*. Como se mencionó anteriormente, se optó por la versión 12 de *PostgreSQL* y la 3.0 de *PostGIS* por ser las más recientes que se encuentran actualmente en ellos.

Por defecto, *PostgreSQL* está configurado para recibir conexiones locales solamente, por lo que precisa de ciertos cambios en las configuraciones. Primero, se debe editar el archivo de configuración (`postgresql.conf`) para cambiar el valor de la opción llamada `listen_addresses`. Valores posibles son: asterisco (`*`) para escuchar en todas las interfaces –que es típicamente la variante más común–, o direcciones *IP* en particular. También se deben efectuar cambios en el archivo de autenticación de clientes (`pg_hba.conf`), agregando entradas para las nuevas reglas de conexión. Para ahondar en cómo se deben generar las reglas de conexión se puede consultar la documentación en línea del archivo de configuración[21].

Además, se instaló el paquete de contribuciones externas (*postgresql-contrib*) para poder contar con información sobre las sentencias ejecutadas (*queries*) y así poder realizar un posterior análisis en la herramienta de monitorización. En particular, se configurará la extensión llamada *pg_stat_statements* para la recolección de información relacionada a la ejecución de las *queries*. Se debe editar nuevamente el archivo de configuración para modificar la opción llamada `shared_preload_libraries` y reiniciar el servicio.

En cuanto a otras configuraciones, *PostgreSQL* tiene por defecto valores aptos para eje-

[19] Página del código fuente del cual se creó el *fork* inicial – https://github.com/Intelligent-Quads/iq_sim

[20] Introducción al *clustering* mediante el algoritmo de *k-means* – https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

[21] Página de documentación del archivo de configuración `pg_hba.conf` – https://www.postgresql.org/docs/current/auth-pg-hba-conf.html

cutar en una instancia con recursos reducidos, como lo es la *Raspberry Pi*, por lo que a priori no se efectuarán cambios, a no ser por las modificaciones en el subsistema de sincronización a disco. Para esta aplicación en particular no se busca minimizar el tiempo de respuesta ante una falla catastrófica, por lo cual se relajan las condiciones requeridas para efectuar los *checkpoints* (o puntos de sincronización) a disco: `checkpoint_timeout` y `max_wal_size`.

Entorno de monitorización

Al servidor *PMM* contar con una imagen de contenedor *Docker* disponible, solamente hace falta contar con un entorno en el cual se puedan desplegar este tipo de contenedores[22], bajar la imagen y ejecutar el comando `docker run` apropiado.

Para instalar el cliente en la terminal del simulador, se utilizó el repositorio disponible para manejadores de paquetes de tipo *.deb*, y se siguieron los pasos descritos en la documentación del producto[23]. Luego, para registrarlo en el servidor se ejecutó el comando `pmm-admin config`.

Dado que no hay paquetes disponibles para la arquitectura *ARM*, no se cuenta con binarios previamente compilados para la *Raspberry Pi*. Siguiendo los pasos documentados en el proyecto GitLab se pudo compilar los binarios requeridos, para luego registrar la instancia en el servidor, y de esta manera contar con métricas del sistema operativo. Como último paso se conecta *PMM* a la base de datos. Para ello se utiliza la funcionalidad de conexión remota, la cual no precisa de un cliente ejecutándose localmente en la instancia, y se puede configurar directamente desde la interfaz gráfica web del servidor[24].

3.3.3. Desarrollo

Las funcionalidades fueron implementadas en cinco etapas, siguiendo un modelo iterativo e incremental. En una primer etapa se desarrollaron los cambios en el mapa 3D para obtener una simulación de un dron en un ambiente orientado a la agricultura. Luego, se agregó un plan de vuelo autónomo, en el cual el dron es guiado por la estación de control trazando un recorrido en cubrimiento de la parcela. La tercer etapa estuvo compuesta por la generación de imágenes con metadatos del vuelo –coordenadas *GPS* y altura relativa al suelo–, y la cuarta etapa por la implementación de la comunicación con la base de datos. De esta manera se logra tener toda la información necesaria para trabajar con la

[22] Página de instalación de *Docker* en distintas plataformas – https://docs.docker.com/engine/install/

[23] Página de instalación del cliente *PMM* vía paquetes *.deb* – https://docs.percona.com/percona-monitoring-and-management/setting-up/client/#package-manager

[24] Página de documentación del cliente remoto para *PostgreSQL* – https://docs.percona.com/percona-monitoring-and-management/setting-up/client/postgresql.html#with-the-user-interface

generación de nueva información. En la quinta etapa de desarrollo se consume la información almacenada en la base de datos, y se procesa para generar nuevos datos, dando por finalizado el proceso de desarrollo del caso de uso.

La primer etapa en particular representó un gran desafío, dado que previamente no se tenía experiencia en la creación y el modelado de ambientes gráficos en 3D. Se tomó un archivo preexistente de "mundo" provisto por el proyecto en el cual se basa esta simulación, y se modificó para incluir dos modelos de árboles y un modelo de piso de tierra obtenidos del proyecto de código abierto `agribot`[25]. Se exploraron también otros proyectos, pero debido a que no fue posible integrarlos de manera exitosa en un plazo de tiempo razonable, se decidió continuar con este modelo simple. Entre los proyectos investigados se destaca `fields-ignition`[26], que aleatoriamente genera mapas con plantas de tomates en filas paralelas, situadas sobre un piso de tierra.

Luego de generar el mapa, se cambió la definición del dron para tener una cámara orientada verticalmente hacia abajo y otra en diagonal hacia abajo y adelante. La primera es la encargada de tomar las imágenes de la plantación durante el vuelo, y si bien la segunda no cumple un rol activo en la toma de datos, se decidió dejar para casos futuros en los que pueda ser de utilidad. Ambas cámaras generan imágenes con resolución de 640x480 pixeles. En la Fig. 3.10 se muestra el resultado de este proceso, en el cual se tiene un dron que es capaz de sobrevolar –guiado manualmente– una extensión de tierra con árboles. Por último, se modificó la velocidad de generación de imágenes a una por segundo, cambio que se actualizará luego en las pruebas funcionales, a modo de probar el cubrimiento de las configuraciones para una misión dada. Esta configuración se encuentra en el archivo `worlds/agro.world`, en la sección `update_rate` de la cámara llamada `webcam_down`.

Con estos cambios implementados, se pasó a la segunda etapa de desarrollo, en la cual se agregó un plan de vuelo al dron de manera que pueda desplazarse de forma autónoma. Para ello se utiliza la estación de control *QGroundControl*, como se mencionó anteriormente, y se trazan de manera manual los puntos del mapa por donde se desea que el dron vuele. Además, se establecen la altura de vuelo (10 metros) y la velocidad de desplazamiento (18 km/h) deseadas. Una vez se tiene el plan de vuelo trazado, se lo carga al dron para su almacenamiento y ejecución, y se guarda un respaldo en el archivo `mission.plan`.

Una vez se obtuvo una simulación exitosa del dron sobrevolando el mapa, se pasó a generar el archivo de lanzamiento automatizado en *ROS* (`agro.launch`), para utilizar con el comando `roslaunch` y poder ejecutar las distintas herramientas necesarias de manera desatendida. En una primer instancia se agregaron las siguientes componentes: el simulador *SITL*, *QGroundControl*, *Gazebo* con el nuevo mapa y el traductor *mavros*.

Con estas funcionalidades desarrolladas y probadas, se da paso a la iteración de desarrollo

[25] Código fuente de `agribot` – https://github.com/PRBonn/agribot
[26] Código fuente de *fields-ignition* – https://github.com/azazdeaz/fields-ignition

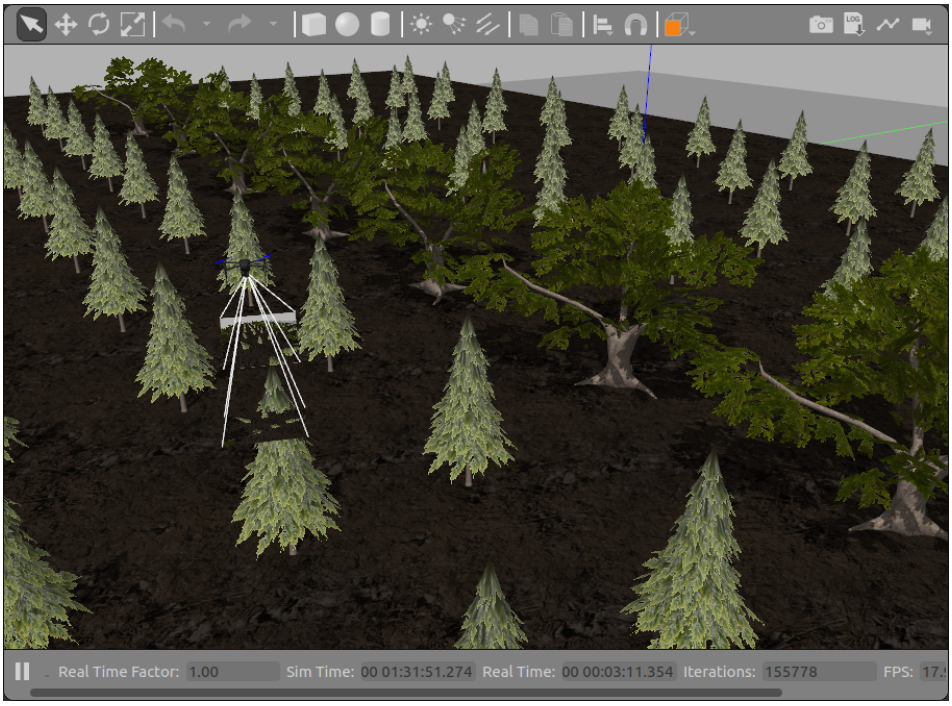


Figura 3.10: Dron guiado manualmente, sobrevolando un modelo 3D con pinos y robles.

de los módulos *Python*, en los que se integran la lógica de persistencia en la base de datos, y del posprocesamiento de los mismos. El primer paso fue la creación de un nuevo tipo de mensaje –o estructura de datos– *ROS* en el cual se componen los datos de imagen, coordenada *GPS* y altura relativa al suelo publicados respectivamente en los siguientes tópicos *ROS*: `/webcam_down/image_raw`, `/mavros/global_position/global`, y `/mavros/global_position/rel_alt`. Al ser generados en tres tópicos distintos, es necesario desarrollar un nuevo nodo *ROS* encargado de la sincronización. Las coordenadas *GPS* y alturas se coordinan mediante la primitiva de sincronización de tópicos `ApproximateTimeSynchronizer`[27], que permite definir un pequeño desfasaje entre la detección de mensajes publicados en tópicos distintos. La unión de estos datos se guarda –y actualiza con cada llamado– como variable global, a la cual luego se tiene acceso desde la *callback* definida en el suscriptor del tópico de imágenes. De esta forma, se logra no perder datos de imágenes en caso que la actualización del *GPS* o la altura no existan en un momento dado; es decir, se prioriza tener metadata potencialmente desactualizada sobre las imágenes, antes que perder imágenes. Se publica la nueva información

[27] Página de documentación de *ApproximateTimeSynchronizer* – https://docs.ros.org/en/api/message_filters/html/python/index.html?highlight=slop#message_filters.ApproximateTimeSynchronizer

en el tópico `/sync_drone_image`, y el nodo encargado de esta tarea es el definido en `scripts/gdscf/sync_drone_image.py`.

Por otro lado, para la persistencia de los datos se hizo uso del conector de base de datos *psycopg2* en el nodo `scripts/gdscf/drone_camera_listener.py`. En una primer instancia se obtiene la conexión con la base de datos, la cual se mantiene abierta durante todo el ciclo de vida del módulo. Luego de la conexión, el módulo se suscribe al tópico de imágenes con metadatos en `/sync_drone_image` y espera por nuevos llamados en la *callback* mediante la primitiva de *ROS* `rospy.spin()`. El módulo cuenta con una variable global para detectar cuándo comienza y termina la misión, y en caso de no encontrarse activamente en un recorrido se descartan las imágenes. La lógica para ello es simple, pero efectiva en el caso de la simulación controlada: cuando el dron supera los 8 metros de altura se considera que la misión ha comenzado, y por el contrario, cuando la altura es menor a 8 metros se considera que ha terminado. Al contar también con las coordenadas *GPS*, es fácil ver cómo se puede extender esta lógica para contemplar casos más complejos sin mucha dificultad. Una vez se considera que el dron está activamente en vuelo, las imágenes y metadatos entrantes se guardan en la tabla correspondiente en la base de datos mediante la ejecución de sentencias `INSERT INTO`. Este proceso se repite hasta que el módulo se detiene, y se procede a cerrar la conexión con la base de datos.

La quinta etapa comprendió el posprocesamiento de la información almacenada para una misión dada, y fue implementada en `process_images_drone_mission.py`. En este caso, la conexión con la base de datos se cierra inmediatamente luego de obtener las tuplas, dado que se utiliza `fetchall()` para obtener todo el resultado de la consulta. Podría ser necesario cambiar esta lógica si se decidiera operar de manera iterativa sobre el conjunto de datos, por ejemplo, en casos en los que exceda los límites de la memoria disponible. De cada imagen interesa obtener el valor de la componente verde de cada pixel, el valor del índice *ExG* y los histogramas de cada uno de ellos. Con esta información se validó la hipótesis planteada sobre la distribución de los verdes en las imágenes, para luego definir la cantidad de grupos que se desea segregar mediante el algoritmo de *k-means*. Se observó que hay dos grandes grupos de pixeles: los que cuentan con "poco verde" y los que no, resultado que se mantuvo tanto para el histograma de verdes como para el de valores *ExG*. En la Fig. 3.11 se puede ver un ejemplo de histogramas computados a partir de una imagen tomada por el dron en la que se distinguen estos dos grupos. En el histograma de la derecha, utilizando el índice de vegetación *ExG*, se puede apreciar de mejor manera la distinción entre estos dos grupos.

Utilizando `k=2` para el cálculo de los centros se tomó el punto medio y se obtuvo el valor de corte para generar imágenes en blanco y negro, en donde un pixel negro representa la ausencia de vegetación y uno blanco lo contrario. Dado que el algoritmo de *k-means* arroja como resultado la cantidad de puntos en cada conjunto, se decidió también ponderar el valor de corte respecto a la población de cada uno, brindando un nuevo valor de corte que tenderá a estar proporcionalmente más cerca del centro con mayor población[28]. Con

[28] Esto es equivalente a ejecutar *k-means* con k=1. En el *script* `process_images_drone_mission.py`

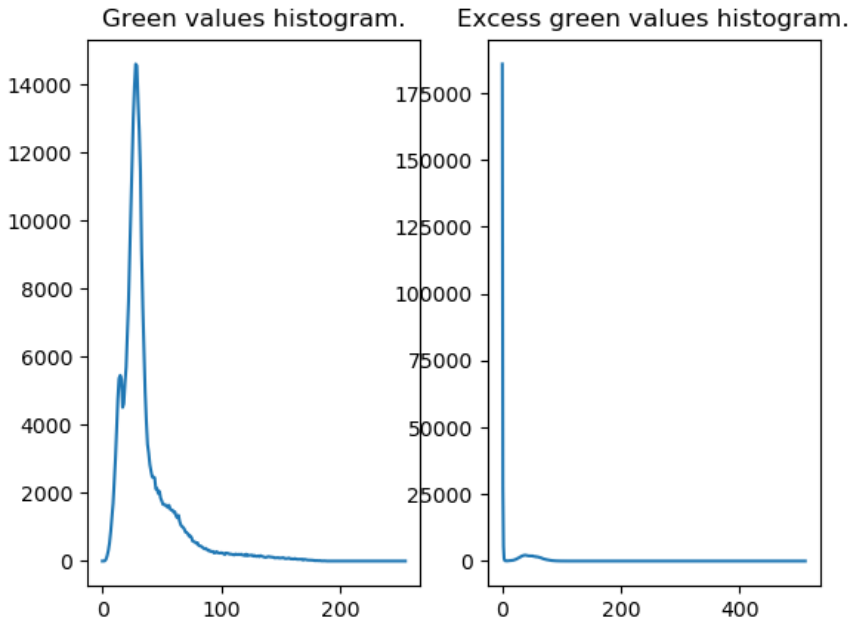


Figura 3.11: Histogramas generados para una imagen tomada por el dron, en los que se pueden ver los dos grandes grupos de pixeles: verdes y "no verdes" en el histograma de la izquierda, y *ExG* a la derecha. Es posible ver cómo la utilización del índice de vegetación *ExG* mejora la distinción de estos dos grupos.

este resultado se genera una segunda ronda de imágenes en blanco y negro, componiendo de esta manera el resultado final que se observa en la Fig. 3.12. Se eligió una disposición en columnas para poder comparar las diferencias en cada paso de manera más clara. Por ejemplo, las imágenes desplegadas en la segunda columna son las equivalentes al mismo paso computacional, partiendo de distintas fuentes –*RGB* arriba versus *ExG* abajo. En la primer iteración a partir de la imagen *RGB*, se puede ver como hay un gran número de falsos positivos, es decir, pixeles que se toman como vegetación que no lo son. También se ve una gran influencia de las sombras e incluso la detección de troncos y ramas como vegetación. Este fenómeno no está presente en las calculadas a partir del índice *ExG*, en ninguna de las iteraciones. En la tercer columna se muestra el refinamiento de los centroides por población, lo cual en ambas instancias mejora la calidad de los resultados. En el caso de *RGB* ayuda a disminuir la detección de falsos positivos, y en el caso del índice *ExG* ayuda a disminuir los falsos negativos por influencia de las sombras. Habiendo efectuado una inspección visual sobre todas las imágenes generadas, se concluye que los mejores re-

se agregan salidas en pantalla para mostrar estos resultados.

sultados se obtienen con los índices *ExG* con centroides corregidos por población. Tanto las imágenes como los histogramas generados se guardan en un directorio –elegido por el usuario– para su posterior análisis, y se incluyen los resultantes de la simulación en el directorio `data/outputs/` del proyecto en *GitLab*.

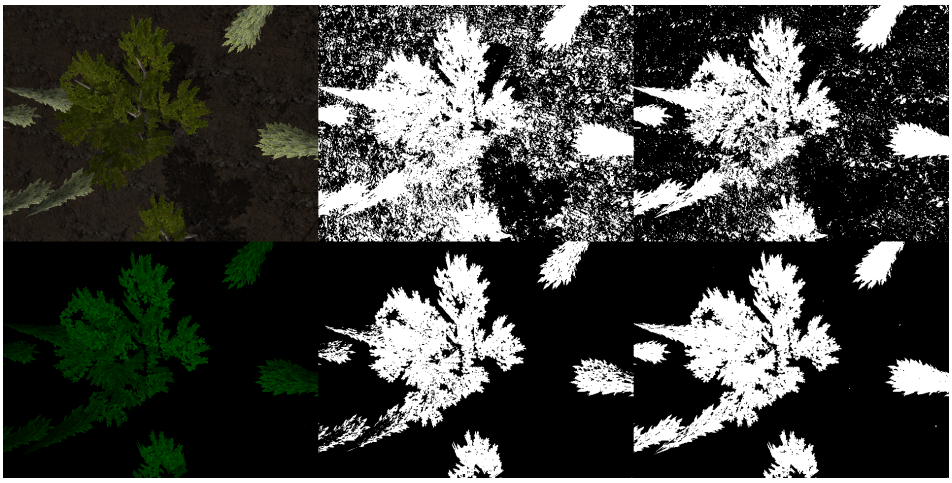


Figura 3.12: Imágenes generadas en la etapa de posprocesamiento. Arriba: a la izquierda la imagen *RGB* original, al centro y a la derecha imágenes en blanco y negro con valores de corte sin y con corrección proporcional a la población. Abajo: a la izquierda la imagen en escala de verdes del índice *ExG*, al centro y a la derecha imágenes en blanco y negro con valores de corte sin y con corrección proporcional a la población.

Como parte de los resultados de la simulación se decidió también guardar los datos de todos los tópicos *ROS*. Para ello se utilizó una *ROS Bag*[29], que es un repositorio de tópicos que puede ser utilizado para reproducir experiencias previamente guardadas, para intercambio de información entre distintos desarrolladores, o simplemente como respaldo. Esta información se guarda automáticamente en `data/rosbag/` y, dado que puede generar archivos grandes rápidamente, se encuentra comentada por defecto. Por ejemplo, una simulación de 3 minutos con 10 fotos por segundo resultó en un archivo de aproximadamente 20Gb.

Finalmente, es deseable poder lanzar todas estas componentes de manera desatendida –exceptuando el *script* de posprocesamiento–, para lo que se creó un archivo de lanzamiento (*launch file*) *ROS*. En él se definen mediante lenguaje *XML* todos los nodos con sus argumentos, y es la herramienta la encargada de ejecutarlos sin intervención del usuario. Con este fin se creó el archivo `launch/agro.launch`, el cual se puede ejecutar con el comando `roslaunch iq_sim agro.launch drone_id:=X`

[29] Página de documentación de *ROS Bag* – https://wiki.ros.org/Bags

`mission_id:=Y`, donde *X* es el ID asignado al dron y el ID de la misión es *Y*.

3.3.4. Pruebas funcionales

Las pruebas funcionales –al igual que el desarrollo– también siguieron un modelo iterativo e incremental. A medida que las funcionalidades se fueron implementando, se generaron pruebas y mensajes de impresión en pantalla para validarlas. Para futura referencia, gran parte de los mensajes desplegados se dejaron en el código, ya sea en forma de comentario o como parte de los mensajes de salida controlados por la variable `debug`. Por defecto se le asigna el valor "falso", para que la salida sea lo menos verbosa posible, pero se puede cambiar la asignación en el código en caso de querer reproducirlos.

El primer paso fue validar que la cámara estuviera correctamente publicando las imágenes en el tópico deseado. Para ello se utilizó el nodo `image_view`[30], que toma las imágenes publicadas en un tópico a elección y las despliega en pantalla a medida que se publican. Se agregó en el archivo de lanzamiento en forma de comentario, para documentar su modo de ejecución y tener una clara referencia de cómo se llevó a cabo la validación. Luego de una inspección visual de una recorrida completa, se probó el suscriptor *ROS* desarrollado en *Python*, que imprime el código *RGB* del primer pixel de cada imagen en pantalla, para luego ser corroborado manualmente con un generador de colores en línea[31]. Además, se volvieron a generar las imágenes para realizar una inspección visual de todos los pixeles procesados. Para ello se utilizó la función `Image.show()`[32] de la biblioteca *pil*, que despliega en pantalla una imagen generada, en este caso, a partir de un arreglo de tuplas *RGB*.

Luego se probó que la información en la base de datos fuera consistente con estos datos. En cada inserción se imprime un mensaje en consola con los metadatos asociados y los valores del primer pixel, de manera de poder validarlos ejecutando consultas directamente en la base de datos. Si bien fue un proceso manual –y parcial, dado que no se comprobó en extensión para todos los pixeles–, en etapas posteriores se regeneraron las imágenes *RGB* a partir de los datos recuperados de la base de datos, y se verificó que se generaron las mismas imágenes que fueron publicadas en los tópicos (un ejemplo de esto es la imagen de arriba a la izquierda en la Fig. 3.12).

La siguiente etapa de pruebas involucró el armado de histogramas y cómputo de centros del algoritmo de *k-means*. Para ello se crearon gráficas con los histogramas tanto para valores de verdes como para valores de índice *ExG*, y se imprimieron en pantalla los resultados de todos los centros y las poblaciones de cada grupo para validación manual. Por último, se generaron las imágenes compuestas y se inspeccionaron visualmente como forma de validación de los resultados finales obtenidos.

[30] Página de documentación de `image_view` – https://wiki.ros.org/image_view

[31] Como el que puede encontrarse en la siguiente página – https://rgbcolorcode.com/

[32] Página de documentación de `show()` – https://pillow.readthedocs.io/en/latest/reference/Image.html#PIL.Image.Image.show

3.3.5. Pruebas de rendimiento

El objetivo de las pruebas es medir la utilización de los recursos disponibles en busca de potenciales mejoras en el rendimiento. Medir no solo permite descubrir áreas en las que se puede mejorar –o si se puede de hecho mejorar–, probar cambios en configuraciones y algoritmos utilizados o brindar datos de soporte a la toma de decisiones, también permite comparar la calidad de las distintas herramientas desarrolladas. Se debe notar que optimizar la utilización de un recurso no es siempre sinónimo de reducir su uso; por ejemplo, en la fase de posprocesamiento, es deseable poder maximizar el uso del procesador, mientras que en la fase de recolección de datos por el dron es deseable minimizarlo.

En la simulación se relevaron datos de la misión y el dron, con el objetivo principal de cuantificar el rendimiento de la batería. En un dron, es deseable poder maximizar el tiempo de vuelo, y para ello es necesario optimizar los recursos que hagan uso de la batería. En esta instancia el foco fue meramente observar su utilización con el objetivo de documentarla, y no tratar de obtener datos para un trabajo de optimización. En la Fig. 3.13 se puede ver un ejemplo de la utilización de la batería en dos pruebas distintas, siendo la única diferencia entre ellas la frecuencia de actualización de la cámara – una foto por segundo versus diez. Se observó que no hay grandes diferencias en la utilización de la batería (0,52 %, dado que la capacidad total de la batería es de 3300 mAh), por lo que a priori se puede concluir que la frecuencia de la cámara no influye negativamente en la duración de una misión. Sin embargo, en estas pruebas, el módulo de procesamiento de imágenes y escritura en la base de datos (`drone_camera_listener.py`) no es tenido en cuenta como parte de la utilización de recursos del dron. En una implementación no simulada este módulo podría ser un factor determinante en este sentido, si el procesamiento de cada imagen tiene un costo no despreciable (por ejemplo, por la comunicación remota con la base de datos en la estación de control).

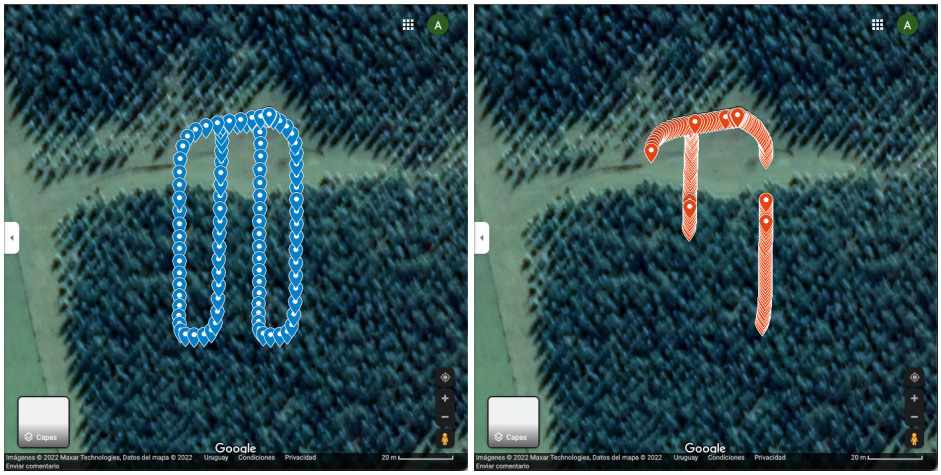
Al ejecutar pruebas con frecuencias de diez fotos por segundo, se notó que la cantidad de imágenes guardadas en la base de datos no se correspondía con la esperada. Luego de consultar los metadatos de las imágenes, se notó que no todos los segundos tenían diez imágenes asociadas, por lo que se procedió a inspeccionar la información de manera visual en un mapa. Como se muestra en la Fig. 3.14, el resultado de haber incrementado excesivamente la cantidad de imágenes a recolectar fue la pérdida de datos en gran parte del tramo de la misión. En la misión con una foto por segundo se guardaron 105 imágenes, y en la misión con diez fotos por segundo 378. Si bien hay más de tres veces la cantidad de fotos, a partir de la reconstrucción de los datos se puede ver que el recorrido del dron no es representativo de la misión real.

Para entender mejor cuánto tiempo lleva cada escritura en la base de datos, se agregaron en el módulo `drone_camera_listener.py` impresiones en pantalla con los tiempos de cada operación, y se ejecutó nuevamente una simulación –esta vez con una foto por segundo, para evitar tomar mediciones en un sistema saturado. Los resultados arrojaron tiempos de escritura a la base de datos en el orden de los 500 ms., al menos un orden de magnitud mayor al esperado. Como se puede ver en la Fig. 3.15, al inspeccio-

(a) (b)

Figura 3.13: Pruebas de utilización de la batería para un mismo recorrido. En 3.13a se muestran datos de la misión con una foto por segundo, y en 3.13b con diez fotos por segundo.

nar el desempeño de la *Raspberry Pi* en *PMM* se notó que las métricas de latencia de las escrituras a disco se correlacionaron con los tiempos vistos desde la aplicación. En este caso, el cuello de botella se encuentra en el subsistema de E/S (entrada/salida) del nodo de base de datos, ya sea por su latencia implícita o por la latencia resultado de forzar el sincronismo de la escritura a disco. No se detectan diferencias en la utilización de memoria, y si bien el procesador está notoriamente más exigido en las pruebas con diez imágenes por segundo, se puede ver cómo gran parte del tiempo está dado por esperas al subsistema de E/S (en rojo en la Fig. 3.15a). Se probó deshabilitando el sincronismo forzado a los *logs* de escritura (`synchronous_commit='off'`), pero no hubo cambios considerables en los tiempos de escritura reportados. Se probó también incrementar las variables `shared_buffers` (a 512Mb) y `bgwriter_lru_maxpages` (a 500), sin mejoras aparentes en los resultados. La variable `shared_buffers` es utilizada para asignar la cantidad de memoria *RAM* que puede utilizar *PostgreSQL* para dimensionar el *cache* interno, de manera de minimizar la presión de E/S ejercida sobre el *page cache* del sistema operativo. Además, incrementar `bgwriter_lru_maxpages` ayuda a que los procesos encargados de escribir las páginas a disco puedan procesar una mayor cantidad de páginas en cada bucle de ejecución, liberando a los clientes de aplicación de esta tarea bloqueante. Esto podría ayudar a que los procesos de conexión de clientes ejecuten más rápidamente, decrementando los tiempos de ejecución vistos del lado de la aplicación.



(a) (b)

Figura 3.14: Datos de posición *GPS* por imagen subidos a *Google Maps* para su visualización. En 3.14a se muestra una misión exitosa con una foto por segundo. En 3.14b se muestra una misión con el mismo recorrido (pero tomando diez fotos por segundo) en la que hubo pérdida de información.

Como se mencionó anteriormente, en ninguno de los casos se vio mejoras sustanciales en los tiempos reportados en `drone_camera_listener.py`, con lo que se concluye que la limitación está dada por la latencia de escritura a la tarjeta *microSD* utilizada.

Por último, para la ejecución del algoritmo de *k-means*, se probó cambiar la variable de paralelismo (`n_jobs`), y el tipo de algoritmo utilizado (`algorithm`) en diferentes ejecuciones, pero no se logró mejorar los tiempos de ejecución reportados por imagen logrados con sus valores por defecto. Se probó también decrementar la cantidad de veces que se ejecuta el algoritmo para distintos centros iniciales, definida por `n_init`, y en ese caso sí se logró reducir los tiempos de ejecución, sin sacrificar la precisión de las soluciones encontradas. A diferencia de las otras variables, reducir `n_init` podría afectar la calidad de la solución encontrada, por lo que puede no ser deseable sacrificar precisión en pos de una ejecución más rápida.

(a) (b)

Figura 3.15: Métricas de utilización de recursos del nodo de base de datos. En 3.15a se ven métricas de utilización de *CPU* y memoria, y en 3.15b de tasa y latencia de E/S. Entre las 11:12 y 11:15 se ejecutó una simulación con 10 imágenes por segundo, y entre las 11:17 y 11:19 con una imagen por segundo.

4 Resultados obtenidos

4.1. Revisión de la literatura

El relevamiento del estado del arte permitió ahondar en numerosas etapas del ciclo de vida de los datos, desde la obtención mediante sensores, la transferencia y persistencia en un manejador de base de datos, a su procesamiento y posterior utilización como parte de un sistema de soporte a la toma de decisiones.

Los datos se generan a partir de sensores, cámaras y otras fuentes –como estaciones meteorológicas cercanas– para brindar información sobre el estado de los cultivos. Esta información se transfiere a unidades de procesamiento central, encargadas de la persistencia y gestión. Típicamente se utilizan tecnologías de transferencia inalámbrica para el envío de los datos, y se favorecen herramientas y protocolos ligeros, dado que la vida útil de la batería es una restricción en este tipo de entornos. En las etapas previas al ingreso de los datos en el manejador, se pueden aplicar técnicas de calidad de datos para reducir el error de las inferencias generadas en etapas posteriores. Además, en las etapas de posprocesamiento es común aplicar técnicas de inteligencia artificial sobre los datos, ya sea para generar nueva información o interpretar la existente de manera más efectiva.

En cuanto al almacenamiento y gestión de los datos, la totalidad de los manejadores de bases de datos mencionados son de código abierto, y si bien predominan los relacionales, algunos proyectos utilizan también no relacionales –habiendo incluso menciones a sistemas híbridos. Las temáticas de datos masivos y la computación en la nube están presentes en la literatura, aunque son tratadas únicamente de manera superficial.

En la mayoría de los casos, los datos son utilizados tanto en tiempo real como para su análisis a posteriori, y en general no hay exigencias en los tiempos de arribo de nueva información. Los escenarios de aplicación varían desde monitorización de la salud de las plantas, al análisis de riesgos y predicciones de las cosechas, además de ser parte integral de los procesos y sistemas de soporte a la toma de decisiones. Los datos son utilizados durante todo el ciclo de vida del cultivo, alimentando un bucle de toma de decisiones que tiene como objetivo la optimización de la producción.

Finalmente, se pudo ver también la influencia de las licencias *open source*, tanto para las herramientas de *software* como para algunas piezas de *hardware* utilizadas. Si bien no fueron muchos los casos en los que se compartió el código fuente, se espera que estos

aumenten a medida que se desarrollen más trabajos de investigación en el área.

Solamente dos temáticas –de las nueve propuestas inicialmente– permanecieron inconclusas por falta de referencias: las características de los datos y la arquitectura de gestión de datos. En particular, no se encontró información sobre posibles arquitecturas de referencia a utilizar en entornos de Agricultura Inteligente.

4.2. Caso de estudio

El principal resultado obtenido a partir del desarrollo del caso de estudio fue el de presentar una simulación funcional, implementada utilizando solamente herramientas de código abierto, en la que se pueden explorar la completitud de las etapas correspondidas en una posible aplicación de Agricultura Inteligente, desde la captura de datos, a su almacenamiento en una de base de datos y posterior posprocesamiento para la extracción de información. No se tiene conocimiento de otro proyecto de código abierto específicamente diseñado para entornos de Agricultura Inteligente que cuente con estas funcionalidades en la actualidad.

En particular, se permite visualizar imágenes a partir de índices de vegetación calculados sobre las imágenes recolectadas por un dron. Si bien el índice utilizado (*ExG*) no es de gran utilidad para medir la salud de los cultivos, es útil para separar vegetación de su entorno, lo cual lo convierte en un índice adecuado para el conteo o seguimiento del crecimiento de áreas verdes. Al utilizar información del espectro visible únicamente, no es necesario contar con equipamiento costoso –como ser cámaras multiespectrales–, lo que lo convierte en un buen candidato para realizar pruebas de campo no simuladas, de manera de corroborar los resultados de la simulación. Las pruebas simuladas demostraron que el índice *ExG* arroja mejores resultados que el procesamiento basado únicamente en la componente verde del espectro *RGB* de una cámara convencional. Al generar información con los centroides ponderados por población, resultado de aplicar el algoritmo de *k-means*, se puede segregar más efectivamente la vegetación del suelo. Si bien los resultados de los centroides sin ponderar (por población) son también útiles, se pierden algunas zonas de verdes producto de las sombras proyectadas. En ambos casos –centroides con y sin corrección de población– los resultados de utilizar el espectro verde contienen un mayor porcentaje de "ruido" que puede dificultar la interpretación de los resultados, es decir, tienen mayor número de falsos positivos.

Además, se pudo probar que con las especificaciones mencionadas, la *Raspberry Pi 3* no es un buen candidato para nodo de base de datos, ya que la latencia del subsistema de E/S no permite escalar por encima de dos imágenes de 640x480 por segundo, aproximadamente. Teniendo en cuenta que una cámara *RGB* hoy en día tiene comúnmente soporte para resoluciones de al menos 1280×720, es posible que no sea suficiente inclusive para una imagen por segundo en estos casos. Se debe tener en cuenta que además la *Raspberry Pi* oficia como estación de control, por lo que no es recomendable saturar los recursos con

la funcionalidad de persistencia de datos.

5 Conclusiones y trabajos futuros

5.1. Conclusiones

La Agricultura Inteligente es un campo cada vez más relevante, y como tal, tiene gran potencial para continuar su estudio en trabajos futuros. El relevamiento del estado del arte reveló que si bien en los últimos años se ha visto un incremento en la cantidad de trabajos académicos con foco en entornos de Agricultura Inteligente, todavía hay temáticas que no han sido desarrolladas en profundidad. En particular, en el área de gestión de datos no se pudo relevar información sobre arquitecturas de referencia para este tipo de entornos, o de los tipos de datos y estructuras a utilizar. Por otro lado, hay una gran cantidad de estudios e información sobre otras áreas relevadas, como la generación y transferencia de los datos, la temporalidad de los datos y los escenarios de aplicación. Por último, si bien no hubo un desarrollo en profundidad, se estudiaron también distintas herramientas y técnicas de procesamiento y almacenamiento de los datos.

La prueba de concepto aquí desarrollada muestra una visión completa de las técnicas a aplicar en un posible caso de uso, y si bien existen mejoras y funcionalidades adicionales que se pueden implementar, se pudo validar que la utilización de técnicas de Agricultura Inteligente –en particular, la creación de nueva información a partir de índices de vegetación– puede generar buenos resultados. El caso de estudio mostró una posible implementación de un sistema de detección de copas de árboles, en la cual se utilizó el índice de vegetación *ExG* para aislar los árboles –o la vegetación– del suelo.

5.2. Trabajos futuros

En lo que a la simulación respecta, se puede mejorar el mundo 3D provisto para contener mayor variedad de objetos. En particular, agregar modelos que sean de utilidad para estudiar plantaciones típicas de la región –como ser soja, vid o arroz–, o generar modelos aleatoriamente[1] para cada tipo, de manera de poder simular más fielmente la realidad. Además, se debe incrementar la resolución de la cámara utilizada, y experimentar con imágenes generadas a partir de información multi e hiperespectral. En Willis et al. (2022)

[1] Como el trabajo propuesto en https://github.com/azazdeaz/fields-ignition mencionado previamente.

se describe un proyecto que genera, a partir de imágenes disponibles vía *Google Earth Engine*[2], modelos de mundo en *Gazebo* con información multiespectral. Si bien se realizaron pruebas con la intención de utilizar la herramienta, al generar las imágenes se notó que la resolución de la información multiespectral que brinda el motor de *Google Earth* es muy baja (de 10 metros por pixel la más precisa), por lo que no es adecuada para el caso de uso que se quería demostrar. En la Fig. 5.1 se puede ver la diferencia en la cantidad de información obtenida de dos fuentes distintas –una cámara *RGB* y una multiespectral que capta el infrarrojo cercano– utilizando los mismos parámetros de ubicación y área cubierta (media hectárea), es decir, ambas imágenes describen exactamente la misma extensión de tierra, pero una tiene 10 veces menos resolución. Puede ser útil para simulaciones que abarquen estudios en cultivos de extensión como la soja, por ejemplo, por lo que no se descarta su inclusión en trabajos futuros.

En el área de procesamiento de imágenes, es cada vez más notoria la utilización de técnicas de inteligencia artificial. En el caso particular de la simulación presentada, se puede continuar explorando resultados de extracción de información a partir de los índices generados, como ser conteo de árboles detectados por imagen o metro cuadrado, o área de verde detectada por árbol. Es también interesante poder generar una sola imagen a partir de las distintas imágenes capturadas durante la misión, para generar de esta manera un mapa completo del predio evaluado. Además, es necesario contar con información sobre las coordenadas GPS proyectadas en cada vértice de las imágenes, tanto para situar el mapa como los objetos de interés ubicados en él. Para ello se pueden utilizar técnicas de transformación de marcos de referencia en el robot[3] en conjunción con proyecciones de *camera pinhole*[4]. Con esta información, se tiene la posibilidad de mejorar los planes de vuelo para maximizar el área cubierta por misión, o garantizar el cubrimiento y solapamiento necesarios para generar mapas de mayor calidad.

Se debe, además, investigar en mayor profundidad las diferentes temáticas concernientes a la gestión de los datos. En el área de características de los datos, ¿qué tipos de datos y estructuras de tablas se deben utilizar? Y en cuanto a la distribución de los datos, ¿el volumen amerita utilizar herramientas que cuenten con *sharding* nativo? ¿Cambia ésto las consideraciones que se deben tener sobre los tipos de datos y demás estructuras a utilizar? También puede resultar interesante investigar los tipos de acceso que los diferentes actores involucrados deben tener a la información almacenada y, como se mencionó anteriormente, la falta de una arquitectura de referencia para gestión de datos en este tipo de entornos es notoria y amerita un desarrollo en mayor profundidad. Por último, dos áreas en las cuales tampoco hay suficiente información –que fueron nombradas en reiteradas ocasiones en la literatura– son los datos a gran escala (*Big Data*) y el cómputo y almacenamiento en la nube (*Cloud Computing*).

[2] Página principal de *Google Earth Engine* – https://earthengine.google.com/

[3] Página de documentación de *ROS TF* para el manejo de transformadas – https://wiki.ros.org/tf

[4] Página de documentación del módulo *ROS* de *camera pinhole* – https://docs.ros.org/en/jade/api/image_geometry/html/c++/classimage__geometry_1_1PinholeCameraModel.html

Figura 5.1: Imágenes obtenidas de *Google Earth Engine* mediante `ros_georegistration`. Se puede ver información de cámaras *RGB* y *NIR* (en el recuadro rojo) centradas en la posición de comienzo del dron simulado, y de 500m de lado. Si bien la dimensión utilizada es la misma, la resolución de los datos es 10 veces menor en la imagen *NIR* (no se alteró el tamaño de las imágenes).

Bibliografía

Y. Liu, X. Ma, L. Shu, G. P. Hancke, A. M. Abu-Mahfouz, From Industry 4.0 to Agriculture 4.0: Current Status, Enabling Technologies, and Research Challenges, IEEE Transactions on Industrial Informatics 17 (2021) 4322–4334. doi:10.1109/TII.2020.3003910.

E. Navarro, N. Costa, A. Pereira, A systematic review of IoT solutions for smart farming, Sensors 20 (2020) 4231.

FAO, The State of the World's Land and Water Resources for Food and Agriculture – Systems at breaking point. Synthesis report 2021., Technical Report, Food and Agriculture Organization, 2021. URL: https://doi.org/10.4060/cb7654en, Último acceso: 2022/05/03.

C. Okoli, A Guide to Conducting a Standalone Systematic Literature Review, Communications of the Association for Information Systems 37 (2015). URL: https://hal.archives-ouvertes.fr/hal-01574600.

J. Martin, A. Ansuategi, I. Maurtua, A. Gutierrez, D. Obregón, O. Casquero, M. Marcos, A Generic ROS-Based Control Architecture for Pest Inspection and Treatment in Greenhouses Using a Mobile Manipulator, IEEE Access 9 (2021) 94981–94995.

G. Giray, C. Catal, Design of a Data Management Reference Architecture for Sustainable Agriculture, Sustainability 13 (2021) 7309.

E. Belcore, S. Angeli, E. Colucci, M. A. Musci, I. Aicardi, Precision agriculture workflow, from data collection to data management using FOSS tools: An application in northern Italy vineyard, ISPRS International Journal of Geo-Information 10 (2021) 236.

O. Debauche, J.-P. Trani, S. Mahmoudi, P. Manneback, J. Bindelle, S. A. Mahmoudi, A. Guttadauria, F. Lebeau, Data management and internet of things: A methodological review in smart farming, Internet of Things 14 (2021) 100378.

F. N. Fote, S. Mahmoudi, A. Roukh, S. A. Mahmoudi, Big data storage and analysis for smart farming, in: 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), IEEE, 2020, pp. 1–8.

M. Rakhra, R. Singh, Smart data in innovative farming, Materials Today: Proceedings (2021).

J. Iaksch, E. Fernandes, M. Borsato, Digitalization and Big data in smart farming–a review, Journal of Management Analytics 8 (2021) 333–349.

E. K. Moore, A. Kriesberg, S. Schroeder, K. Geil, I. Haugen, C. Barford, E. M. Johns, D. Arthur, M. Sheffield, S. M. Ritchie, et al., Agricultural data management and sharing: Best practices and case study, 2021.

T. Wang, X. Xu, C. Wang, Z. Li, D. Li, From smart farming towards unmanned farms: a new mode of agricultural production, Agriculture 11 (2021) 145.

N. Islam, M. M. Rashid, F. Pasandideh, B. Ray, S. Moore, R. Kadel, A review of applications and communication technologies for internet of things (Iot) and unmanned aerial vehicle (uav) based sustainable smart farming, Sustainability 13 (2021) 1821.

V. Moysiadis, P. Sarigiannidis, V. Vitsas, A. Khelifi, Smart farming in Europe, Computer science review 39 (2021) 100345.

P. K. R. Maddikunta, S. Hakak, M. Alazab, S. Bhattacharya, T. R. Gadekallu, W. Z. Khan, Q.-V. Pham, Unmanned aerial vehicles in smart agriculture: Applications, requirements, and challenges, IEEE Sensors Journal 21 (2021) 17608–17619.

J. V. Y. Martínez, A. F. Skarmeta, M. A. Zamora-Izquierdo, A. P. Ramallo-Gonzlez, IoT-based data management for Smart Agriculture, in: 2020 Second International Conference on Embedded & Distributed Systems (EDiS), IEEE, 2020, pp. 41–46.

A. Roukh, F. N. Fote, S. A. Mahmoudi, S. Mahmoudi, Big data processing architecture for smart farming, Procedia Computer Science 177 (2020) 78–85.

J. Kanimozhi, G. Pavithra, K. Pooja, S. Saranya, A study of smart farming based on iot, in: 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), IEEE, 2020, pp. 1–6.

R. P. Sishodia, R. L. Ray, S. K. Singh, Applications of remote sensing in precision agriculture: A review, Remote Sensing 12 (2020) 3136.

D. Sarri, S. Lombardo, A. Pagliai, C. Perna, R. Lisci, V. De Pascale, M. Rimediotti, G. Cencini, M. Vieri, Smart farming introduction in wine farms: A systematic review and a new proposal, Sustainability 12 (2020) 7191.

A. D. Boursianis, M. S. Papadopoulou, P. Diamantoulakis, A. Liopa-Tsakalidi, P. Barouchas, G. Salahas, G. Karagiannidis, S. Wan, S. K. Goudos, Internet of things (IoT) and agricultural unmanned aerial vehicles (UAVs) in smart farming: a comprehensive review, Internet of Things 18 (2022) 100187.

V. Saiz-Rubio, F. Rovira-Más, From smart farming towards agriculture 5.0: A review on crop data management, Agronomy 10 (2020) 207.

P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, I. Moscholios, A compilation of UAV applications for precision agriculture, Computer Networks 172 (2020) 107148.

R. Helmké, J. Bauer, A. Bothe, N. Aschenbruck, CAN't–An ISOBUS privacy proxy for collaborative smart farming, in: 2019 IEEE 38th international performance computing and communications conference (IPCCC), IEEE, 2019, pp. 1–3.

D. C. Tsouros, S. Bibi, P. G. Sarigiannidis, A review on UAV-based applications for precision agriculture, Information 10 (2019) 349.

M. Bacco, P. Barsocchi, E. Ferro, A. Gotta, M. Ruggeri, The digitisation of agriculture: a survey of research activities on smart farming, Array 3 (2019) 100009.

M. S. Farooq, S. Riaz, A. Abid, K. Abid, M. A. Naeem, A Survey on the Role of IoT in Agriculture for the Implementation of Smart Farming, IEEE Access 7 (2019) 156237–156271.

S. C. Hassler, F. Baysal-Gurel, Unmanned aircraft system (UAS) technology and applications in agriculture, Agronomy 9 (2019) 618.

A. Triantafyllou, D. C. Tsouros, P. Sarigiannidis, S. Bibi, An architecture model for smart farming, in: 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), IEEE, 2019, pp. 385–392.

D. C. Tsouros, A. Triantafyllou, S. Bibi, P. G. Sarigannidis, Data acquisition and analysis methods in uav-based applications for precision agriculture, in: 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), IEEE, 2019, pp. 377–384.

M. A. Rodríguez, L. Cuenca, Á. Ortiz, Big Data Transformation in Agriculture: From Precision Agriculture Towards Smart Farming, in: Working Conference on Virtual Enterprises, Springer, 2019, pp. 467–474.

J. Kim, S. Kim, C. Ju, H. I. Son, Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications, Ieee Access 7 (2019) 105100–105115.

R. Morais, N. Silva, J. Mendes, T. Adão, L. Pádua, J. A. López-Riquelme, N. Pavón-Pulido, J. J. Sousa, E. Peres, Mysense: A comprehensive data management environment to improve precision agriculture practices, Computers and Electronics in Agriculture 162 (2019) 882–894.

C. Cambra Baseca, S. Sendra, J. Lloret, J. Tomas, A smart decision system for digital farming, Agronomy 9 (2019) 216.

D. Glaroudis, A. Iossifides, P. Chatzimisios, Survey, comparison and research challenges of IoT application protocols for smart farming, Computer Networks 168 (2020) 107037.

S. Wolfert, L. Ge, C. Verdouw, M.-J. Bogaardt, Big data in smart farming–a review, Agricultural systems 153 (2017) 69–80.

A. Jacobs, The Pathologies of Big Data: Scale up Your Datasets Enough and All Your Apps Will Come Undone. What Are the Typical Problems and Where Do the Bottlenecks Generally Surface?, Queue 7 (2009) 10–19. URL: https://doi.org/10.1145/1563821.1563874. doi:10.1145/1563821.1563874.

S. R. G. SRG, Huge cloud market still growing at 34 % per year; amazon, microsoft & google now account for 65 % of the total, Online, 2022. URL: https://www.srgresearch.com/articles/huge-cloud-market-is-still-growing-at-34-per-year-amazon-microsoft-ar

Gartner, Gartner forecasts worldwide public cloud end-user spending to reach nearly $500 billion in 2022, Online, 2022. URL: https://www.gartner.com/en/newsroom/press-releases/2022-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending

U. Unidad Reguladora y de Control de Datos Personales, Guía de drones, Technical Report, AGESIC, 2018. URL: https://www.gub.uy/unidad-reguladora-control-datos-personales/comunicacion/publicaciones/guia-de-drones.

B. Lu, Y. He, P. D. Dao, Comparing the performance of multispectral and hyperspectral images for estimating vegetation properties, IEEE Journal of selected topics in applied earth observations and remote sensing 12 (2019) 1784–1797.

A. Lucieer, Z. Malenovskỳ, T. Veness, L. Wallace, HyperUAS—Imaging spectroscopy from a multirotor unmanned aircraft system, Journal of Field Robotics 31 (2014) 571–590.

T. Adão, J. Hruška, L. Pádua, J. Bessa, E. Peres, R. Morais, J. J. Sousa, Hyperspectral imaging: A review on UAV-based sensors, data processing and applications for agriculture and forestry, Remote sensing 9 (2017) 1110.

A. Matese, P. Toscano, S. F. Di Gennaro, L. Genesio, F. P. Vaccari, J. Primicerio, C. Belli, A. Zaldei, R. Bianconi, B. Gioli, Intercomparison of UAV, aircraft and satellite remote sensing platforms for precision viticulture, Remote Sensing 7 (2015) 2971–2990.

P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, C. Stachniss, UAV-based crop and weed classification for smart farming, in: 2017 IEEE international conference on robotics and automation (ICRA), IEEE, 2017, pp. 3024–3031.

G. Sylvester, E-agriculture in action: drones for agriculture, Food and Agriculture Organization ofn the United Nations and International . . . , 2018.

S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, Robot Operating System 2: Design, architecture, and uses in the wild, Science Robotics 7 (2022) eabm6074. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074. doi:10.1126/scirobotics.abm6074.

R. d. Soto, F. Díaz, M. Mendivil, Termodron 2: dron autónomo de reconocimiento por termografía (2019).

M. Amado, G. Fischer, K. Sosa, Termodron III: Dron autónomo de reconocimiento por termografía. (2021).

E. Johnson, Intelligent Quads Tutorials, 2022. URL: https://github.com/Intelligent-Quads/iq_tutorials#software-development-tutorials.

A. R. Willis, K. Brink, K. Dipple, ROS georegistration: Aerial Multi-spectral Image Simulator for the Robot Operating System, arXiv preprint arXiv:2201.07863 (2022).