

PEOPLE'S DEMOCRATIC REPUBLIC OF
ALGERIA



Ministry of Higher Education and Scientific Research



UNIVERSITY OF EL OUED
FACULTY OF EXACT SCIENCES
Computer Science Department

Thesis

submitted in partial fulfillment of the requirements for the degree of

ACADEMIC MASTER

Field: Mathematics and Computer Science

Option: Computer Science

Specialty: Internet of things & Cybersecurity

Project Title:

**Design and Implementation of an Intelligent UAV-
Based System for Real-Time Plant Disease Detection**

Board of Examiners

- ◆ Chairman: Dr. Mohammed Mounir
Bouhamed
- ◆ Examiner: Dr. Mostefa Kara
- ◆ Supervisor: Dr. Abdelkader Hima

Prepared by

Badrelboudour Lamouri
Manar Sama Bendebka

Academic Year: 2025 / 2026

Table of contents

شكر وتقدير	VIII
إهداء	IX
إهداء	X
Abstract	XI
ملخص	XII
General Introduction	XIII
I Smart agricultural drones and artificial intelligence	1
Introduction	1
I.1 Historical and conceptual evolution of agriculture	1
I.1.1 Traditional agriculture	1
I.1.2 Precision agriculture (PA)	2
I.1.3 Smart agriculture	2
I.2 Detailed explanation of core technologies	3
I.2.1 Surveying technologies and aerial platforms	4
I.2.2 Artificial intelligence and image processing	4
I.2.3 Management and simulation technologies	5
I.2.4 Implementation and resource optimization	5
I.3 The pivotal role of drones and artificial intelligence	5
I.3.1 Drones	5

I.3.2	Artificial intelligence	6
I.4	Comparison of uav systems for precision agriculture	6
I.5	Comparative analysis of object detection models	7
I.6	Challenges and limitations	8
I.6.1	Technical challenges	8
I.6.2	Regulatory and economic challenges:	9
I.7	Related works	9
	Conclusion	11
II	The open source computer vision library	12
	Introduction	12
II.1	OpenCV library	12
II.2	Advantages of OpenCV library	12
II.3	The role of the OpenCV library in the proposed system	13
II.4	Image preprocessing techniques	14
II.4.1	Image resizing using OpenCV	14
II.4.2	Image color space conversion in OpenCV	14
II.4.3	Noise reduction	18
II.4.4	Contrast enhancement	18
II.5	Integration of OpenCV with YOLOv8	19
	Conclusion	20
III	System design and implementation	21
	Introduction	21
III.1	Project description	21
III.1.1	System overview	21
III.2	System implementation	22
III.2.1	Software tools	22
III.2.2	Hardware components	23
III.3	System architecture	25

III.3.1 Architecture description	25
III.4 Dataset preparation	27
III.4.1 Handling class imbalance	28
III.4.2 Code training	29
III.5 Esp32 telemetry and data transmission code	30
III.6 Communication and system integration	33
III.6.1 Backend workflow flowchart	34
III.7 Hardware architecture and system interconnection diagram	34
III.8 Security mechanisms	36
Conclusion	37
IV Experimental results and discussion	38
Introduction	38
IV.1 Results of training	38
IV.1.1 Class distribution analysis	38
IV.2 Training and validation performance analysis	40
IV.2.1 Training loss analysis	40
IV.2.2 Validation metrics analysis	41
IV.2.3 Overall model behavior	41
IV.3 Inference results	41
IV.4 Experimental results and deployment	42
IV.4.1 Physical implementation	42
IV.4.2 Login interface	43
IV.4.3 Desktop website interface	44
IV.4.4 Mobile web interface	44
IV.4.5 System latency measurements	45
IV.5 Decision making	46
IV.5.1 Handling false positives and false negatives	46
IV.6 Limitations and future work	47
Conclusion	49

General Conclusion**xiv****References****xv**

List of Figures

I.1	Traditional agriculture practices	2
I.2	Precision agriculture technologies	2
I.3	Smart agriculture	3
II.1	The image used	15
II.2	Result of BGR to grayscale conversion	16
II.3	Result of BGR to HSV conversion	16
II.4	Result of BGR to RGB conversion	17
III.1	Esp32 microcontroller	23
III.2	Esp32 cam	24
III.3	Global positioning system(GPS)	24
III.4	Sensor DHT11	25
III.5	System architecture	25
III.6	Flowchart of the backend system architecture and data flow	34
III.7	Electronic composition	35
IV.1	Dataset analysis and annotation distribution report	39
IV.2	Training loss and detection performance	40
IV.3	Visual inspection of plant disease detection results via the trained YOLOv8 model	42
IV.4	Physical implementation of the embedded hardware architecture	43
IV.5	Screenshot of the login interface	44
IV.6	Screenshot of the website on a desktop client	44
IV.7	Screenshot of the website on a mobile device	45

List of Tables

I.1	Comparison between traditional, precision and smart agriculture	3
I.2	Comparison of the most important vegetation indices for precision agriculture .	4
I.3	Comparison of uav systems for precision agriculture	7
I.4	Object detection models comparison	8
II.1	Common interpolation methods in OpenCv	14
II.2	Common color space conversion codes in OpenCV	17
III.1	Distribution of annotated instances per class	29
III.2	Explanation of the training code	30
III.3	Comprehensive hardware wiring and pin configuration for the drone subsystems	35
IV.1	Overall model accuracy results	41
IV.2	End-to-end system latency measurements (based on 34 samples)	46

Abbreviations

AI artificial intelligence

AWS amazon web services

GPIO general purpose input/output

GPS global positioning system

HSV hue

NIR near infra-red

ONNX open neural network exchange

OpenCV open source computer vision library

PA precision agriculture

RGB red

SAVI soil adjusted vegetation index

TensorRT tensor regression tree

UART universal asynchronous receiver-transmitter

UAV unmanned aerial vehicle

URL uniform resource locator

Wi-Fi wireless fidelity

YOLOv8 you only look once version 8

شكر و تقدير

الحمد لله حمداً يليق بجلال وجهه وعظيم سلطانه، الذي أنار لنا درب العلم والمعرفة، ووفقنا لإتمام هذا العمل المتواضع، سائلين المولى عز وجل أن يجعله علماً نافعاً وعملاً صالحاً. نتقدم بأسمى عبارات الشكر والتقدير والامتنان إلى الأستاذ المشرف د. هيمة عبد القادر الذي كان لنا نعم الموجه والمرشد لما قدمه لنا من دعم وتوجيهات قيمة ونصائح سديدة ومتابعته المستمرة لنا طوال مراحل إنجاز هذه المذكرة، فكان لتشجيعه الأثر الكبير في إتمام هذا العمل. كما نتوجه بخالص الشكر وعظيم الاحترام إلى كافة أساتذة قسم الإعلام الآلي الذين أفاضوا علينا من علمهم ومعرفتهم وكان لهم الفضل في تكويننا العلمي والأكاديمي طيلة مشوارنا الدراسي. ونتقدم كذلك بجزيل الشكر والتقدير إلى الأساتذة الأفاضل أعضاء لجنة المناقشة رئيس اللجنة د. محمد منير بوحامد و المناقش د. مصطفى قارة لتفضلهم بقبول مناقشة هذه المذكرة وعلى ما قدموه من ملاحظات وتوجيهات قيمة تثري هذا العمل وترتقي به. ولا يفوتنا أن نعبر عن امتناننا لكل من ساهم من قريب أو بعيد في مساعدتنا ودعمنا لإنجاز هذا العمل. وفي الأخير نسأل الله عز وجل أن يوفق الجميع لما فيه الخير والنجاح، وأن يجعل هذا العمل خالصاً لوجهه الكريم.

إهداء

بسم الله الرحمن الرحيم
 ((يرفع الله الذين آمنوا منكم والذين أوتوا العلم درجات والله بما تعملون خبير)) صدق الله العظيم
 ها أنا اليوم أتوج لحظاتي الأخيرة في هذا الطريق الذي كان يحمل في باطنه العثرات، ورغماً عنها ظلت
 قدمي تخطو بكل صبر وطموح وعزيمة وتفأؤل وحسن ظن بالله.
 أهدي بكل حب تخرجي:
 إلى ذلك الرجل العظيم، إلى من كان لي العمود الفقري الذي ساندني بكل حب في ضعفي، الذي أخرج
 أجمل ما في داخلي وشجعني دائماً للوصول إلى طموحاتي، إلى من انتظر هذه اللحظات ليفتخر بي...
 والذي العزيز أدامك الله ظلاً لنا
 إلى من جعل الله الجنة تحت أقدامها وسهلت لي الشدائد بدعائها... إلى الإنسانية العظيمة التي طالما تمتنت
 أن تقر عينها في يوم كهذا...
 أمي الغالية حفظك الله
 إلى ضلعي الثابت الذي لا يمل، يا من شددت عضدي به، وإلى الينبوع الذي استسقى به لأرتقي... إلى
 سندي ومسندي واتكأئي...
 أبي الثاني (أخي عبد السلام)
 إلى تلك النجوم التي تثير حياتي، إلى من حبهم يعلو فوق كل حب، إلى من كانوا عوناً وسنداً لي في هذا
 المشوار...
 إخوتي رعاكم الله
 لكل من كان عوناً وسنداً في هذا الطريق... للأصدقاء الأوفياء...
 شكراً لكم جميعاً
 فالحمد لله على ما وهبني، وأن يجعلني مباركاً وأن يعينني أينما كنت، فمن قال "أنا لها" نالها، وأنا لها وإن أبت
 رغماً عنها أتيت بها، فالحمد لله شكراً وحباً وامتناناً.
 ((وآخر دعوانا أن الحمد لله رب العالمين))

إهداء

لَنَا فِي مَرَاتِبِ الشَّرَفِ مَرْتَبَةٌ
تَأْتِي إِلَيْنَا وَإِنْ طَالَ بِنَا الدَّهْرُ
وَإِنْ ظَنَنْتُمْ أَنَّ نَدْنُو مِنْهَا مَنْزِلَةً
فَإِنَّهَا كَانَتْ لَنَا وَبِالصَّبْرِ نَفْتَخِرُ

إلى عائلتي، سندي ومصدر فخري.. شكراً لأنكم كنتم دائماً الدافع والملجأ.

إلى كل من شاركني هذا الدرب وساندني، شكراً لكم جميعاً من القلب.

Abstract

This memorandum aims to develop an intelligent system for monitoring agricultural crops and early detection of plant diseases using Internet of Things (IoT), computer vision, artificial intelligence (AI), and unmanned aerial vehicles (UAVs). Our system utilizes an ESP32 module to read environmental sensor data, such as temperature and humidity, and to determine geolocation using GPS. The ESP32-CAM module is used to capture images of plants and crops while the UAV flies over the fields.

The captured data and images are transmitted to a web server via HTTPS for real-time storage and processing. The processing stage utilizes the OpenCV library to enhance image quality before passing it to the YOLOv8 AI model, which is used to detect plant diseases and pinpoint affected areas with high accuracy. The system also allows for the display of analysis results, sensor data, and geolocation information via an interactive web dashboard, enabling continuous monitoring and rapid, efficient decision-making. The proposed model was trained using a database containing 28 different plant disease classes with a total of 25,274 images within the PlantVillage Dataset. The evaluation results demonstrated the system's effectiveness in detecting plant diseases with good accuracy, achieving a Recall value of approximately 0.6722, a Precision value of 0.7426, a Standard Accuracy of 0.7319, and a Comprehensive Metric of 0.5683, reflecting the model's efficiency in real-time classification and detection. The final results demonstrated the effectiveness of the proposed system in detecting plant diseases with good accuracy and in real time, highlighting the importance of integrating artificial intelligence, the Internet of Things, and drones into smart agriculture to improve the quality of agricultural production and reduce losses resulting from plant diseases.

The system achieved an average inference time of 243.75 ms and a processing rate of 4.07 frames per second, confirming its suitability for real-time agricultural applications.

Keywords: Plant disease detection, deep learning, UAV, smart agriculture, IoT, computer vision, real-time monitoring.

ملخص

تهدف هذه المذكرة إلى تطوير نظام ذكي لمراقبة المحاصيل الزراعية والكشف المبكر عن أمراض النباتات بالاعتماد على تقنيات إنترنت الأشياء (IoT)، والرؤية الحاسوبية، والذكاء الاصطناعي، والطائرات بدون طيار (UAV). يعتمد نظامنا على استخدام وحدة ESP32 لقراءة بيانات الحساسات البيئية، مثل درجة الحرارة والرطوبة، بالإضافة إلى تحديد الموقع الجغرافي بواسطة نظام GPS، في حين تُستخدم وحدة ESP32-CAM لالتقاط صور النباتات والمحاصيل الزراعية أثناء تحليق الطائرة بدون طيار فوق الحقول الزراعية.

يتم إرسال البيانات والصور الملتقطة إلى خادم ويب عبر بروتوكول HTTPS من أجل تخزينها ومعالجتها في الزمن الحقيقي. تعتمد مرحلة المعالجة على استخدام مكتبة OpenCV لتحسين جودة الصور قبل تمريرها إلى نموذج الذكاء الاصطناعي YOLOv8، والذي يُستخدم للكشف عن أمراض النباتات وتحديد المناطق المصابة بدقة عالية. كما يتيح النظام عرض نتائج التحليل وبيانات الحساسات والموقع الجغرافي عبر لوحة تحكم ويب تفاعلية، مما يسمح بالمراقبة المستمرة واتخاذ القرارات المناسبة بسرعة وكفاءة.

تم تدريب النموذج المقترح باستخدام قاعدة بيانات تحتوي على 28 صنفاً مختلفاً من أمراض النباتات بإجمالي 25,274 صورة ضمن مجموعة البيانات (PlantVillage Dataset). وقد أظهرت نتائج التقييم فعالية النظام في الكشف عن الأمراض النباتية بدقة جيدة، حيث بلغت قيمة Recall حوالي 0.6722، بينما وصلت قيمة Precision إلى 0.7426، كما حقق النموذج Standard Accuracy قدرها 0.7319، إضافة إلى Comprehensive Metric بقيمة 0.5683، مما يعكس كفاءة النموذج في التصنيف والكشف في الزمن الحقيقي.

أظهرت النتائج النهائية فعالية النظام المقترح في الكشف عن أمراض النباتات بدقة جيدة وفي الزمن الحقيقي، مما يبرز أهمية دمج تقنيات الذكاء الاصطناعي وإنترنت الأشياء والطائرات بدون طيار في مجال الزراعة الذكية لتحسين جودة الإنتاج الزراعي وتقليل الخسائر الناتجة عن الأمراض النباتية. حقق النظام متوسط زمن استدلال قدره 243.75 ميلي ثانية ومعدل معالجة يبلغ 4.07 إطار في الثانية، مما يؤكد ملاءمته للتطبيقات الزراعية في الوقت الفعلي.

الكلمات المفتاحية: الكشف عن أمراض النبات، التعلم العميق، الطائرات بدون طيار، الزراعة الذكية، إنترنت الأشياء، رؤية الكمبيوتر، المراقبة في الوقت الحقيقي.

General Introduction

Agriculture has historically been the backbone of the Algerian economy, evolving through three distinct phases: traditional subsistence farming before independence (pre-1962), the state-led agricultural revolutions of the 1970s and 1980s, and the national agricultural development plans of the 2000s. Despite the achievements of each phase, persistent challenges remain, particularly the lack of real-time, data-driven crop monitoring tools.

Over the past two decades, the Wilaya of El Oued has transformed from a traditional date-producing oasis into a national leader in potato production. However, this rapid expansion into sandy, high-evaporation terrains has outpaced the development of phytosanitary surveillance systems, making crops increasingly vulnerable to diseases. A field study conducted by Fezzai et al. (2021) revealed that late blight disease (*Phytophthora infestans*) affects approximately 54% of potato production in the region, alongside a wide range of fungal, bacterial, and viral diseases [1]. Traditional diagnostic approaches rely primarily on manual visual inspection, which is labor-intensive, time-consuming, and often detects infections only after visible symptoms appear, allowing diseases to spread and causing significant economic losses.

Recent advancements in Artificial Intelligence (AI), Computer Vision, the Internet of Things (IoT), and Unmanned Aerial Vehicles (UAVs) have opened new perspectives for smart agricultural solutions capable of real-time monitoring. In particular, deep learning models such as YOLOv8 have demonstrated remarkable performance in real-time object detection and disease recognition tasks, making them well-suited for precision agriculture applications.

Motivated by these challenges, this research focuses on the development of an intelligent agricultural payload designed for integration into a UAV platform. The proposed system combines image acquisition, geolocation, edge processing, and AI-based disease detection, while incorporating secure HTTPS communication for data transmission to a remote server for processing, storage, and visualization.

The central research question addressed is: *How can a low-cost intelligent UAV payload be developed to enable real-time plant disease detection, accurate localization of infected areas, and secure data transmission in smart agricultural environments?*

Chapter I

Smart agricultural drones and artificial intelligence

Introduction

The agricultural sector is undergoing a profound transformation, shifting from traditional practices based on uniform field management to smart and precision agriculture, which relies on localized, data driven strategies. This evolution is fundamentally rooted in the integration of unmanned aerial vehicles (UAVs) as advanced data acquisition tools and artificial intelligence (AI), which serves as the analytical brain for this data.

This chapter aims to provide the theoretical framework for the concepts that enable intelligent monitoring, ranging from the principles of remote sensing to deep learning algorithms. It highlights the process of converting aerial imagery into digital intelligence, paving the way for advanced applications in crop growth monitoring and health assessment.

I.1 Historical and conceptual evolution of agriculture

Agricultural practices have evolved through three fundamental generations, each representing a paradigm shift in how land and crops are managed:

I.1.1 Traditional agriculture

This is the legacy pattern that relies entirely on inherited human experience and manual labor. It is characterized by treating the field as a single unit, with irrigation, fertilization, and spraying applied uniformly across the entire area, without accounting for variations among individual plants. This method lacks precise data, leading to significant resource waste and lower productivity [2].



Figure I.1: Traditional agriculture practices

I.1.2 Precision agriculture (PA)

Emerging with the advent of global positioning system (GPS) and remote sensing, precision agriculture (PA) is a management approach aimed at monitoring and responding to intrafield variability. The core principle is providing the right resource, in the right place, at the right time. It relies on soil maps and satellite imagery to identify areas requiring specific care, though the final decision often remains with the farmer [3].



Figure I.2: Precision agriculture technologies

I.1.3 Smart agriculture

This is the latest and most comprehensive umbrella (Industry 4.0 in Agriculture). It goes beyond spatial precision by adding intelligence and big data ,it integrates iot and ai to create autonomous,

predictive systems, in smart agriculture, data is automatically analyzed to support proactive decisions on disease and yield [3], [4].



Figure I.3: Smart agriculture

Table I.1 illustrates the transition from traditional methods, which lack data and precision, to smart agriculture, which represents the current state of industry 4.0 in farming. While precision agriculture introduced the concept of doing the right thing in the right place [3], smart agriculture which is the core of this research adds an extra layer of intelligence artificial intelligence. This allows for automated disease diagnosis and real time decision making without constant human intervention [4], [5].

Table I.1: Comparison between traditional, precision and smart agriculture

Comparison factor	Traditional agriculture	Precision agriculture	Smart Agriculture
Concept	Based on inherited experience and manual labor.	Focuses on managing intra field spatial variability.	Focuses on data driven automation and prediction.
Main Technologies	Simple tools and direct human observation.	GPS, sensors, and satellite imagery [3].	AI, IoT, big data, and digital twins [3, 4].
Data Collection	Periodic monitoring by human eye.	Spatial mapping and remote sensing [6].	Continuous real-time data streaming [7].
Decision Making	Human-centric (based on intuition/experience).	Human-centric (supported by data maps) [3].	Autonomous (AI-driven) or predictive analytics [5].
Resource Efficiency	Low; high waste of water and chemicals [2].	High; optimized via variable rate technology (VRT) [2].	Maximum; predictive analytics [5].
Key Advantage	Low initial cost, but low productivity.	Optimized resource management and yield.	Real-time monitoring and disease forecasting [8].

I.2 Detailed explanation of core technologies

The technologies used in smart and precision agriculture fall under five main technical pillars:

I.2.1 Surveying technologies and aerial platforms

Unmanned aerial vehicles (drones) function as autonomous surveying platforms capable of flying at low altitudes to provide images with very high spatial resolution [6].

Complementing drone platforms, multispectral technology measures light reflectance in near infra-red (NIR) and red edge bands to calculate vegetation indices such as NDVI, NDRE, and soil adjusted vegetation index (SAVI) [4]. This process involves mounting a multispectral camera on a UAV to capture aerial images of agricultural fields across different spectral bands, enabling vegetation condition analysis with higher accuracy than traditional satellite imagery.

Multispectral sensors measure light reflectance in specific divisions of the electromagnetic spectrum, including Green (520--600 nm), Red (620--700 nm), and NIR (760--900 nm). These sensors also utilize the red edge band (700--740 nm) to detect early chlorophyll changes and plant stress indicators. Using these bands allows for the generation of precise vegetation indices such as NDVI, NDRE, and SAVI, which are essential for high-accuracy analysis of vegetation density and health.

As shown in Table I.2, **NDVI** is the most widely used index for general vegetation health monitoring, although it remains sensitive to soil background interference. **NDRE** provides better sensitivity to early plant stress by exploiting the red edge band, while **SAVI** introduces a soil adjustment factor (L) to improve accuracy in fields with sparse vegetation [9].

Table I.2: Comparison of the most important vegetation indices for precision agriculture

Index	Mathematical formula	Spectral bands	Measured parameter	Main advantage	Limitation
NDVI	$\frac{NIR - Red}{NIR + Red}$	NIR and Red	Vegetation density and health	Simple to compute and widely adopted	Sensitive to soil background
NDRE	$\frac{NIR - Red_{edge}}{NIR + Red_{edge}}$	NIR and Red _{edge}	Early plant stress detection	Higher sensitivity at early growth stages	Requires a red edge sensor
SAVI	$\frac{NIR - Red}{NIR + Red + L} \times$	NIR and Red	Vegetation with soil correction	Suitable for sparse vegetation areas	Requires calibration of L (soil brightness)

I.2.2 Artificial intelligence and image processing

Artificial intelligence plays a central role in transforming raw aerial imagery into actionable agricultural insights. YOLOv8 is a state-of-the-art real-time object detection model particularly well-suited for UAV and IoT applications, enabling instantaneous and accurate identification of plant diseases from drone-captured images. However, training such deep learning models from scratch requires substantial computational resources and large annotated datasets. To address this, transfer learning techniques are employed, leveraging pre-trained model weights

from large-scale architectures such as ResNet [4]. Additionally, super-resolution algorithms are applied to reconstruct fine details from noisy or low-quality drone imagery, further improving detection reliability under challenging field conditions [10]. The seamless integration of these AI capabilities with IoT infrastructure, connecting UAVs with ground-based sensors via protocols such as LoRaWAN or 5G, enables continuous data exchange across the field [7]. Furthermore, edge computing allows data to be processed directly on-board the drone, minimizing transmission latency and enabling timely response without dependence on remote servers [8].

I.2.3 Management and simulation technologies

At the management level, digital twin technology creates a virtual model of the farm that mirrors real-world conditions, enabling predictive analysis of disease spread and crop behavior under varying environmental scenarios [3]. This is further enhanced by GIS and GPS mapping, which links AI model outputs to precise geographic locations, allowing farmers to identify and target infected zones with spatial accuracy, thereby optimizing resource allocation and intervention strategies [11].

I.2.4 Implementation and resource optimization

Variable Rate Technology (VRT) represents the operational layer of smart agriculture, where intelligent diagnostic data is translated into precise field actions. VRT-enabled machinery receives AI-generated infection maps and applies pesticides exclusively to affected spots, minimizing chemical waste, reducing environmental impact, and lowering operational costs for farmers [2].

I.3 The pivotal role of drones and artificial intelligence

I.3.1 Drones

Unmanned aerial vehicles provide critical data acquisition capabilities that form the foundation of intelligent crop monitoring systems. Their ability to fly at low altitudes enables high-resolution imaging of individual plants, making subtle changes in plant health detectable at early stages, long before they become visible to the naked eye [6]. Equipped with multispectral cameras, UAVs can capture spectral data beyond the visible range, revealing hidden stress indicators such as chlorophyll depletion or water deficiency before they become visually apparent [4]. The integration of GPS modules further allows the generation of precise geospatial maps of field conditions, linking detected anomalies to exact geographic coordinates for targeted intervention [11, 7]. Collectively, these capabilities establish UAVs as indispensable tools for

precision agriculture, providing the spatial and spectral resolution necessary for effective crop health monitoring.

I.3.2 Artificial intelligence

While UAVs serve as the sensory organs of the system, collecting vast amounts of visual and spectral data, artificial intelligence serves as the analytical engine, transforming raw drone-collected imagery into structured, actionable intelligence. Detection models trained on extensive annotated datasets learn to recognize complex visual patterns associated with specific plant diseases, achieving competitive levels of accuracy in complex multi-class classification scenarios [5]. Prior to inference, image preprocessing using OpenCV ensures noise reduction, color normalization, and contrast enhancement, significantly improving the quality of input data fed into the detection pipeline and reducing false positives [12]. Together, these components form a coherent AI-driven framework capable of supporting early, accurate, and scalable plant disease diagnosis in precision agriculture environments, ultimately contributing to reduced crop losses and more sustainable farming practices.

I.4 Comparison of uav systems for precision agriculture

Table I.3 details a side by side comparison of key uav systems used in agriculture:

Table I.3: Comparison of uav systems for precision agriculture

System Type	Flight Time	Maneuverability	Cost	payload Capacity	Notes
Fixed-Wing	Long (extended area)	Limited (low hover ability)	Moderate to High	High, but bulky sensors	Excellent for large-scale mapping; lacks real-time processing
VTOL	Moderate	Improved over fixed-wing	High	Moderate	Combines vertical take-off with efficient forward flight; mechanically complex
Quadcopter	Short to Moderate	High (precision hover)	Low to Moderate	Limited by design constraints	Ideal for localized monitoring; best for real-time ai integration

I.5 Comparative analysis of object detection models

Recent studies on object detection in computer vision have evaluated several models based on detection speed, accuracy, and computational efficiency.

Table I.4 summarizes key characteristics of leading object detection models relevant to this project:

Table I.4: Object detection models comparison

Model	Advantages	Disadvantages
yoloV8	• Real-time processing • High accuracy with moderate complexity • Easy retraining for custom datasets	• Requires careful anchor box tuning for very small objects
SSD (Single Shot Multibox Detector)	• Efficient one-shot detection • Good balance between speed and accuracy	• Struggles with low-quality or compressed images; sensitive to parameter tuning
Faster R-CNN	• Superior accuracy in static or detailed scenes • Robust in research applications	• Significantly slower; requires high computational power
MobileNet + SSDLite	• Extremely efficient for low-resource devices • Designed for edge computing	• Lower precision in highly critical tasks such as plant disease identification

I.6 Challenges and limitations

Despite the immense potential of smart agriculture, the adoption of drone and artificial intelligence (AI) technologies faces several core challenges.

I.6.1 Technical challenges

- **Power and flight constraints:** Battery life remains a significant bottleneck; most commercial drones have a flight endurance of 20 to 40 minutes, limiting the ability to survey vast areas in a single mission.
- **Big data processing:** High-resolution and multispectral imagery require high performance computing (HPC) and complex software to analyze and extract results in a timely manner.
- **Environmental conditions:** Wind speed, lighting intensity, and humidity in rural areas affect drone stability and data quality, which may lead to diagnostic errors in crop health assessment.

I.6.2 Regulatory and economic challenges:

- **Legal restrictions (The Algerian case):** The use of drones in Algeria is strictly regulated under Law No. 98-06 relating to civil aviation (amended by Law 15-11). Specifically, article 6 bis mandates prior authorizations from security and civil authorities, which can represent an administrative barrier to immediate and widespread adoption by farmers.
- **Data privacy:** Strict laws govern the filming of open areas and the storage of geospatial data to protect privacy and national security, requiring specific security protocols for handling crop imagery.
- **Economic cost:** The acquisition cost of drones equipped with advanced sensors remains high for small scale farmers, in addition to the need for specialized technical expertise to operate them.

I.7 Related works

Recent studies in the field of smart agriculture have focused on leveraging advanced technologies, such as Unmanned Aerial Vehicles (UAVs), computer vision systems, and the Internet of Things (IoT), for the early detection of plant diseases and efficient crop management. In this context, a field study conducted by Wageningen University in coordination with the University of Hamma Lakhdar in El Oued addressed the practical aspects of potato cultivation in the sandy soils of El Oued, Algeria, proposing an engineering solution to optimize subsurface irrigation and fertigation. Although highly significant for environmental sustainability, that study was confined to ground-level mechanical infrastructure without integrating aerial monitoring [13]. In a related context regarding aerial control and autonomous scanning, the first global study explored the integration of smart drone imaging with mobile applications for automated disease detection through an autonomous inspection system that operates without pre-planned routing. However, its application scope was restricted to indoor environments, leaving a clear gap when transitioning to open-field cultivation and expansive desert areas [8].

Concerning the optimization and lightweight deployment of deep learning algorithms, the third global study directed its focus toward processing drone imagery in open fields to detect cotton plant diseases using the *MobileNetV2* model, which is tailored specifically for lightweight edge computing. While this method succeeded in overall crop classification, the utilization of the YOLOv8 algorithm offers a distinct alternative, as it goes beyond coarse leaf classification to provide real-time object detection and precise localization of disease hotspots directly on the plant [4].

When examining integrated systems and data governance, the SmartAgriChain framework presented a comprehensive model combining multiple IoT sensors, machine learning for irrigation forecasting, and Convolutional Neural Network (CNN) algorithms for pest detection, se-

cured by blockchain technology to ensure data traceability. Although this framework introduces substantial engineering complexity and prohibitive capital costs, hardware simplicity and cost-efficiency can be prioritized by eliminating dense ground-sensor networks in favor of an exclusive, intelligent reliance on an ESP32-CAM and a GPS module, thereby providing an affordable and highly effective alternative [2].

These studies clearly highlight how drone and artificial intelligence based solutions help bridge the technical gaps in traditional agriculture, paving the way for increased production efficiency and resource sustainability.

Conclusion

In conclusion, this chapter has reviewed the qualitative shift from traditional to smart agriculture. The synergy between drone eyes and ai Brains represents an innovative solution to overcome the limitations of human supervision.

Despite technical and regulatory hurdles, ai is the primary engine for transforming aerial images into intelligent agricultural decisions.

Building upon these theoretical foundations, the next chapter will transition to the applied phase using python and OpenCv to automate image analysis and achieve early disease detection.

Chapter II

The open source computer vision library

Introduction

Smart agriculture has emerged as a modern field that relies on advanced technologies such as artificial intelligence, computer vision, and the Internet of Things (IoT) to enhance agricultural productivity and crop quality. In this work, an intelligent payload has been developed for agricultural monitoring, consisting of an ESP32-CAM module for image acquisition, an ESP32 microcontroller for environmental sensing and GPS geolocation, and a DHT11 sensor for temperature and humidity measurements. The captured images are transmitted via an encrypted HTTPS connection to a ground-based server for processing and analysis. In this chapter, we will present the image preprocessing techniques that are applied to the captured images before they are passed to the YOLOv8 deep learning model for disease detection.

II.1 OpenCV library

The open source computer vision library (OpenCV) is a software library primarily designed for computer vision development. It was developed by Intel and is a free, open-source library released under the BSD license. OpenCV can be used on all major operating systems and focuses mainly on real-time image processing, making it suitable for a wide range of computer vision applications [14].

II.2 Advantages of OpenCV library

This library possesses several features that make it a leader in the field of digital image processing, the most important of which are:

1. **Large collection of computer vision algorithms:** OpenCV provides a comprehensive library containing over 500 functions and algorithms dedicated to computer vision and image processing tasks [12].

2. **Real-time image and video processing:** One of the most important advantages of OpenCV is its ability to process images and video streams in real time, making it suitable for applications such as robotics, surveillance systems, and drones [12].
3. **Easy to use framework:** OpenCV offers a user friendly framework that allows developers to build computer vision applications quickly without implementing complex algorithms from scratch [12].
4. **Image capture from cameras:** The library supports capturing images and videos from various types of cameras, enabling the development of real-time vision and monitoring systems [12].
5. **Advanced image processing functions:** OpenCV includes many functions for image transformation, segmentation, filtering, and shape matching, which are essential for image analysis and understanding [12].
6. **Pattern recognition and object detection:** The library provides algorithms for pattern recognition and object detection, including techniques such as face detection and feature extraction [12].
7. **Motion tracking and video analysis:** OpenCV allows tracking objects and analyzing motion in 2D and 3D, useful for surveillance systems, autonomous robots, and UAV applications [12].
8. **3D vision support:** The library includes tools for 3D reconstruction using stereo vision, enabling the creation of 3D models from images [12].
9. **Integration with machine learning algorithms:** OpenCV integrates various machine learning techniques, allowing the development of intelligent systems for classification and recognition tasks [12].

II.3 The role of the OpenCV library in the proposed system

The OpenCV library plays a pivotal role in enhancing and preparing visual data within the smart agriculture system. It performs image processing operations that improve image quality and make it more amenable to accurate analysis by artificial intelligence models.

The library is used to manipulate digital images, extracting crucial visual information and minimizing noise and effects caused by various shooting conditions, such as changes in lighting or movement during capture. It also contributes to improving the visual representation of the image, ensuring higher accuracy in subsequent stages of analysis.

Therefore, OpenCV's role is to support the preprocessing phase within the system, thereby increasing the reliability of the final results and improving the performance of the disease detection model.

II.4 Image preprocessing techniques

The image processing stage is a crucial step to improve image quality and increase the accuracy of artificial intelligence models.

II.4.1 Image resizing using OpenCV

Image resizing is a common preprocessing operation in computer vision used to adjust image dimensions before analysis or model training. In OpenCV, resizing is performed using the `cv2.resize()` function, which allows scaling an image either to a fixed size or by a scaling factor. Different interpolation methods can be selected depending on the resizing objective. For example, `INTER_AREA` is generally preferred for image downscaling, while `INTER_CUBIC` provides smoother results when enlarging images [15].

Resizing is particularly important in plant disease detection systems to standardize input image dimensions before feature extraction or deep learning inference, ensuring consistency across the dataset.

Common interpolation methods:

Table II.1: Common interpolation methods in OpenCv

Method	Typical Use	Description
<code>INTER_AREA</code>	Downscaling	Reduces distortion when shrinking images
<code>INTER_LINEAR</code>	General resizing	Good trade off between speed and quality
<code>INTER_CUBIC</code>	Upscaling	Produces smoother enlarged images

This table II.1 presents the most common interpolation methods used in OpenCV for image resizing and their typical applications.

Example:

```
resized = cv2.resize(image, (640, 640), interpolation=cv2.INTER_AREA)
```

II.4.2 Image color space conversion in OpenCV

`cv2.cvtColor()` is a function in the OpenCV library used to convert images from one color space to another. The most common conversions include:

- **red (RGB) → Grayscale** (simplifies the image to shades of gray)

- **BGR** → **hue (HSV)** (Hue, Saturation, Value)
- **BGR** → **RGB** (used for correct display in Matplotlib) [16].

General Syntax:

```
cv2.cvtColor(src, code[, dst[, dstCn]])
```

- **src:** The input image to be converted.
- **code:** The color conversion code, e.g., `cv2.COLOR_BGR2GRAY`.
- **dst (optional):** The output image with the same size and depth as the input.
- **dstCn (optional):** Number of channels in the destination image; if 0 it is determined automatically.

Examples of `cv2.cvtColor()` method

For the examples, we are using the below image:



Figure II.1: The image used

Examples of color conversion

Convert BGR to grayscale:

```
# Convert to Grayscale
gray_image = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
# Display
cv2.imshow("Grayscale Image, gray_image)
```

Result:

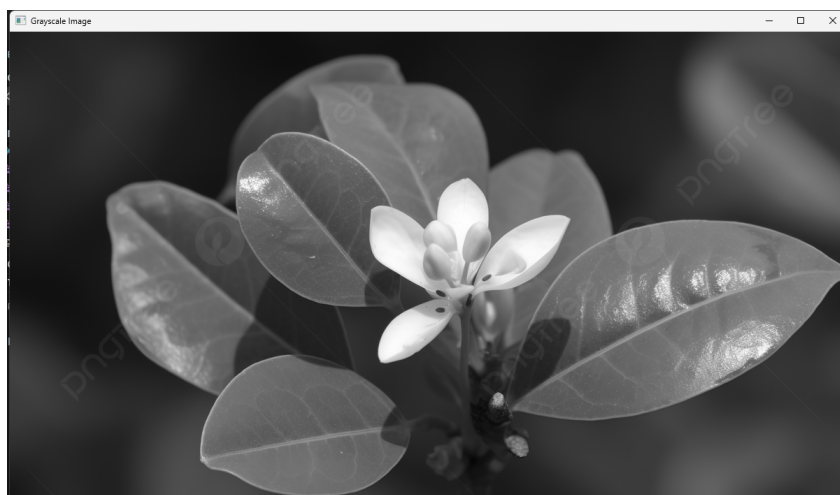


Figure II.2: Result of BGR to grayscale conversion

This conversion simplifies the image and speeds up processing, useful for face detection or OCR (text recognition) [16].

Convert BGR to HSV:

```
# Convert to HSV
hsv_image = cv2.cvtColor(src, cv2.COLOR_BGR2HSV)
cv2.imshow("HSV Image", hsv_image)
```

Result:

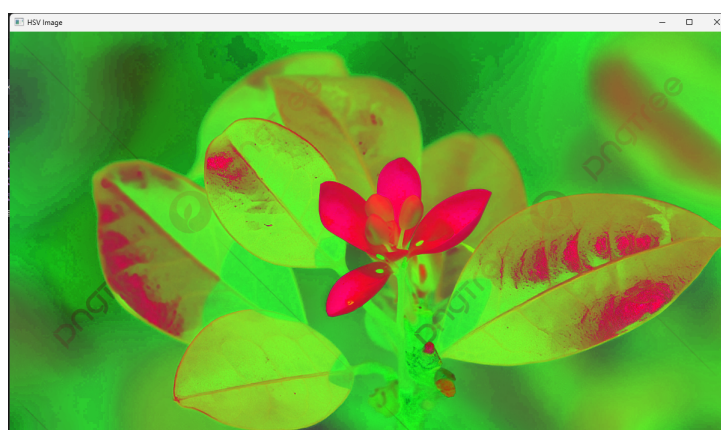


Figure II.3: Result of BGR to HSV conversion

The HSV color space is useful for object tracking or color based analysis in images and videos.

Convert BGR to RGB (for Matplotlib):

```
# Convert from BGR to RGB
rgb_image = cv2.cvtColor(src, cv2.COLOR_BGR2RGB)
```

Result:

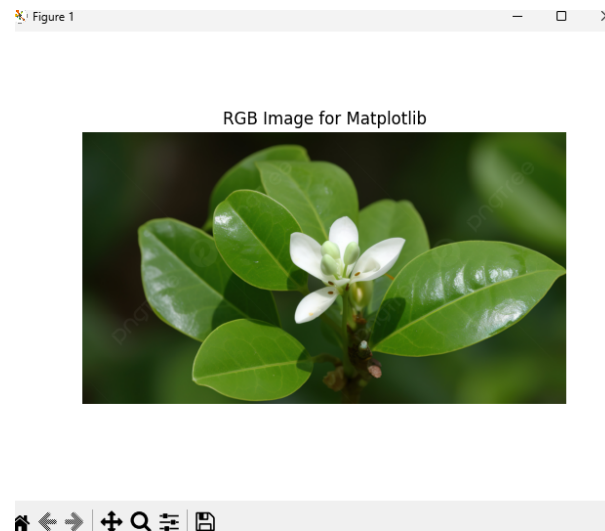


Figure II.4: Result of BGR to RGB conversion

images in BGR format by default, but most other libraries like Matplotlib use RGB, so this conversion is needed for correct color display [16].

Common Color Conversion Codes

Table II.2: Common color space conversion codes in OpenCV

Code	Description
<code>cv2.COLOR_BGR2GRAY</code>	BGR $\rightarrow$ Grayscale
<code>cv2.COLOR_BGR2RGB</code>	BGR $\rightarrow$ RGB
<code>cv2.COLOR_BGR2HSV</code>	BGR $\rightarrow$ HSV
<code>cv2.COLOR_BGR2LAB</code>	BGR $\rightarrow$ LAB
<code>cv2.COLOR_BGR2YCrCb</code>	BGR $\rightarrow$ YCrCb (used for video compression & skin detection)

Practical applications: Different color representation systems are used in many image processing and computer vision applications, with each color model being used depending on the task at hand. For example, grayscale is used in facial recognition and optical character recognition (OCR) applications because it simplifies the image and focuses on structure and shape rather than color. The HSV model is used in object tracking and color recognition because it separates color from light intensity, making it suitable for varying lighting environments. The LAB model is used in color enhancement and skin tone detection because it better reflects human color perception. The YCrCb model is used in video compression and skin tone detection in images and videos because it separates lighting components from color information, thus facilitating processing [16].

II.4.3 Noise reduction

Gaussian blur:

Gaussian Blur is one of the most important image processing techniques used to reduce noise before analysis. This method applies a filter based on the Gaussian function, where higher weights are assigned to pixels near the center of the kernel, while lower weights are given to pixels farther away. This results in a smoother image and reduces unwanted small details. Gaussian Blur is widely used as a preprocessing step before tasks such as edge detection or image segmentation because it improves the accuracy of results by minimizing image noise. [12] In OpenCV, it is implemented using the function: `cv2.GaussianBlur()`

Example in OpenCV:

```
blur = cv2.GaussianBlur(image, (5,5), 0)
```

This code applies a gaussian blur to the image using a 5×5 filter to smooth it and reduce noise. The value 0 allows OpenCV to automatically calculate the standard deviation for better results.

II.4.4 Contrast enhancement

CLAHE (Contrast Limited Adaptive Histogram Equalization) is a contrast enhancement technique used in digital image processing. This technique aims to improve image clarity and highlight fine details, especially in areas with low light or contrast.

In this system, the image is first converted from the BGR color space to the LAB space using the function:

```
cv2.cvtColor(frame_resized, cv2.COLOR_BGR2LAB)
```

The LAB space is well suited for contrast enhancement because it separates lightness information from color information. Channel L represents the lightness level, while channels A and B represent the color information.

Next, the image is split into separate channels using:

```
`l, a, b = cv2.split(lab)`
```

Then, CLAHE is applied to the lighting channel only:

```
`clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))`  
`l = clahe.apply(l)`
```

CLAHE relies on localized contrast enhancement within small areas of the image, rather than processing the entire image at once. This allows for balanced detail enhancement across different parts of the image. The technique also includes a contrast limiting feature (Clip Limit) to prevent excessive noise or lighting.

After optimizing the illumination channel, the three channels are merged again, and the image is converted to BGR space:

```
`lab = cv2.merge((l,a,b))`  
`frame_resized = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)`
```

This procedure improves the quality of plant images and highlights disease symptoms more clearly, thus increasing the accuracy of the YOLOv8 model in detecting plant diseases.

II.5 Integration of OpenCV with YOLOv8

This work integrates OpenCV's image processing techniques with the YOLOv8 model for accurate, real-time early detection of plant diseases. Images captured by an ESP32-CAM camera are first processed using OpenCV to enhance quality, reduce noise, and clarify important details. These images are then fed into the YOLOv8 model, which analyzes plant leaves and identifies affected areas using selection and classification boxes. YOLOv8 was chosen for this study due to its high detection accuracy, fast processing speed, compatibility with drone imagery, and ease of training on custom databases. This integration of OpenCV and YOLOv8 improves the study's performance and enhances the reliability of the proposed intelligent plant disease detection system.

Conclusion

In this chapter, we have presented the image preprocessing techniques employed in the developed agricultural payload to enhance the quality of images captured by the ESP32-CAM module. We introduced the OpenCV library and its advantages for real-time image processing. We also explained how these preprocessing steps are applied on the ground-based server before the images are fed into the YOLOv8 model for disease detection. The integration of OpenCV with YOLOv8 significantly improves detection accuracy and reliability. The following chapter will present the complete system implementation, including hardware components, software tools, dataset preparation, model training and evaluation.

Chapter III

System design and implementation

Introduction

After presenting the theoretical foundations and related concepts in the previous chapters, this chapter focuses on the practical design and implementation of the developed intelligent payload for plant disease detection using deep learning and Internet of Things technologies. This chapter describes the different stages followed in developing the system, including hardware and software components, system architecture, dataset preparation, and communication protocols.

III.1 Project description

The primary objective of this project is to develop an integrated, autonomous unmanned aerial vehicle (UAV) system for precision agriculture, aimed at overcoming the inherent limitations of traditional crop monitoring methods. The drone platform is engineered to execute multi-modal tasks concurrently, ranging from flight control and image-based plant health analysis to real-time environmental data acquisition. This provides farmers and agronomists with actionable insights to optimize crop management strategies.

III.1.1 System overview

The system is developed based on a quadcopter platform, selected for its superior maneuverability and suitability for localized surveillance. The UAV is equipped with advanced components enabling it to navigate agricultural fields while capturing high resolution imagery and sensor telemetry. This holistic approach integrates several key subsystems:

1. **Artificial intelligence and image processing:** An esp32cam module, mounted on the unmanned aerial vehicle (UAV), captures aerial imagery of the crops, which is then transmitted via wifi to a ground processing unit .

The system leverages the yolov8 (you only look once) deep learning model for real time image inference. The model identifies phytopathological issues, such as leaf chlorosis and fungal infections, by generating bounding boxes associated with confidence scores

to determine the probability of infection.

This AI dn framework facilitates early disease detection, enabling rapid intervention, minimizing crop loss, and maximizing overall agricultural productivity.

2. **IoT integration for environmental:** Monitoring Complementing the visual analysis, a DHT11 sensor is integrated into the system to measure ambient temperature and relative humidity. This environmental telemetry is transmitted over wireless fidelity (Wi-Fi) , providing a real visualization interface via a web dashboard or a . This seamless integration of sensor data ensures continuous monitoring of microclimatic conditions, allowing for timely adjustments in crop management practices, such as precision irrigation and fertilization scheduling.

III.2 System implementation

The implementation of the proposed system depends on a set of interconnected hardware and software components.

III.2.1 Software tools

A set of software tools were used, which are as follows:

Google colab: Used as a cloud-based platform for training and testing the YOLOv8 model. It provided access to high-performance GPU resources such as NVIDIA Tesla T4, which accelerated the training process and reduced the need for powerful local hardware. [17].

Arduino ID: Used for writing, compiling, and uploading code to the ESP32-CAM module. It provided the required environment for configuring the camera, Wi-Fi connection, and hardware level functionalities [18].

ESP32 libraries (within arduino IDE) : These libraries enabled proper initialization and operation of the ESP32-CAM module, including camera setup, Wi-Fi communication, and server connectivity [18].

Ultralytics + you only look once version 8 (YOLOv8): The YOLOv8 model was trained and executed using the Ultralytics library, which offers a simple and efficient API for training, evaluation, and deployment of deep learning models [19].

OpenCV: Used extensively for image and video processing operations. It took care of camera frames, image conversion, and pre/post-processing of object detection [20].

NumPy: Used for numerical computations and efficient manipulation of image arrays and pixel values [21].

Time & OS: Standard Python libraries used for time management, delays, timestamps, file handling, and operating system operations [22].

Socket: Used to establish communication between the ESP32-CAM module and the Python

server application through network sockets, enabling real-time transmission of image frames and sensor data [22].

Ngrok: Used as a secure tunneling and reverse proxy service to expose the locally hosted Python server to the public internet through an encrypted HTTPS endpoint. This facilitated remote communication between the server and edge devices across different networks [23].

III.2.2 Hardware components

The physical deployment of the system relies on a combination of compact, high-efficiency hardware tools designed for edge computing and remote sensing. Each component plays a vital role in capturing visual data, tracking flight coordinates, or monitoring environmental factors. The following section details functions of the hardware tools.

Esp32 microcontroller: A powerful, low-cost development board equipped with a dual-core processor, integrated wifi, and bluetooth capabilities. It was utilized as the main control unit and central processing hub of the drone's payload to manage data flow and coordinate between peripheral modules. The microcontroller handled real-time data acquisition from the GPS module and managed system telemetry, optimal and stable edge performance during the flight [24].

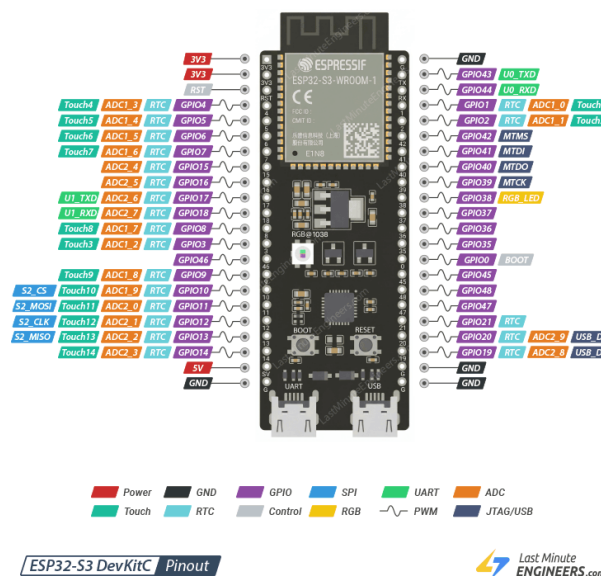


Figure III.1: Esp32 microcontroller

Esp32 cam: A compact development board with a builtin esp32 chip that includes a camera onboard. It was used to capture live photos and transmit video for object detection. The module served as the edge device and sent data to the central processing system over wireless connectivity [25].

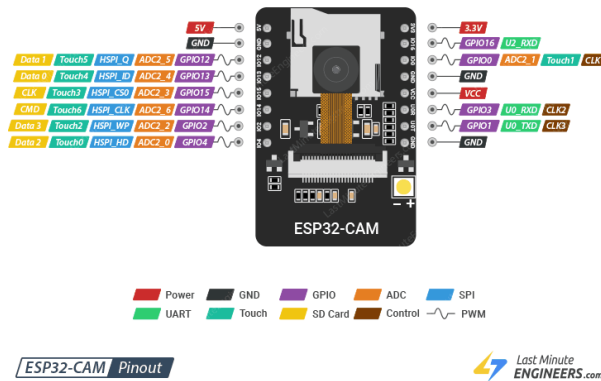


Figure III.2: Esp32 cam

Global positioning system(GPS): A high precision positioning module integrated into the system to provide real time geographic coordinates. It was utilized to pinpoint the exact location of the drone during its flight over the agricultural fields ,by acquiring latitude and longitude data, the module enabled the system to geotag every detected plant disease, allowing the central processing unit to map infection hotspots accurately [26].

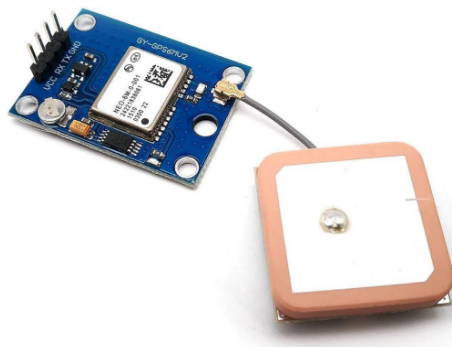


Figure III.3: Global positioning system(GPS)

Sensor DHT11: A digital environmental sensor used to measure ambient temperature and relative humidity within the agricultural microclimate, it was integrated into the system to monitor the atmospheric conditions that directly influence the proliferation of plant diseases, such as high humidity levels which often trigger fungal infections [27].

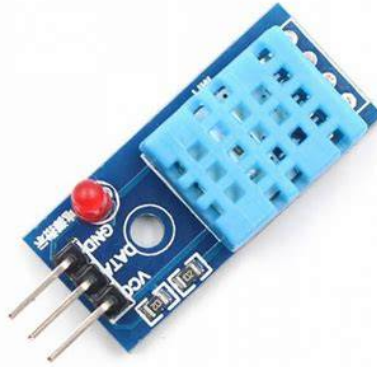


Figure III.4: Sensor DHT11

III.3 System architecture

The proposed architectural diagram represents an integrated intelligent system for agricultural crop monitoring and plant disease detection in real time, under the name AgriDrone AI. The system relies on the principle of decoupled, concurrent acquisition, where each microcontroller operates entirely independently of the other without any direct physical or programmatic link between them. Data is processed numerically and streamed in real time, eliminating the need for continuous storage unless strictly necessary (i.e., only when an infection is detected).

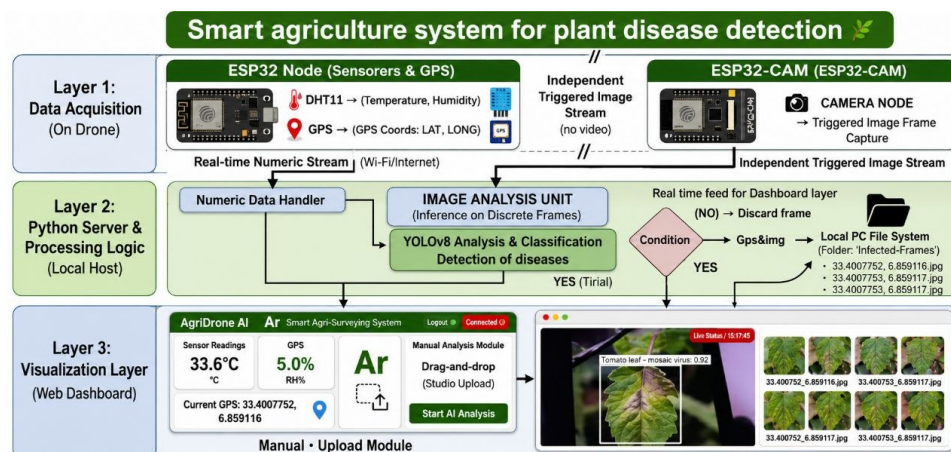


Figure III.5: System architecture

III.3.1 Architecture description

The technical architecture of the system is divided into three integrated main layers:

1. Data acquisition layer (Layer 1)

The data collection process takes place on board the drone via two independent pathways operating in parallel to transmit data over the wireless network (Wi-Fi/Internet):

Sensor node (ESP32 node): This node is connected to the DHT11 climate sensor and the GPS module. It is responsible for reading environmental data (temperature and humidity) and determining the current geographical location. This data is then sent as a continuous, synchronized real-time numeric stream to the server.

Camera node (ESP32-CAM node): Operating as a completely separate system, its sole function is to capture individual discrete frames of plant leaves upon request or at intermittent intervals. It transmits these frames immediately as an independent triggered image stream to the server without processing them as continuous video, thereby minimizing network bandwidth consumption.

2. Python server & processing logic layer (Layer 2)

Hosted locally on a computer (Local host), this layer serves as the core processing engine that coordinates the incoming field data through two separate software pipelines:

Numeric data handler: It receives the temperature, humidity, and location data from the ESP32 node and passes it immediately as a real-time passthrough to the visualization layer without saving it anywhere on the server's permanent storage.

Image analysis unit: It receives the discrete image frames from the ESP32-CAM node and forwards them directly to the YOLOv8 artificial intelligence model for disease detection and classification.

Condition check gateway: Based on the model's inference output, the server executes a conditional save logic as follows:

If no plant disease is detected (NO): The current image frame is discarded and ignored entirely to optimize storage capacity.

If a plant disease is detected (YES): A triggered save command is activated. The server instantly pairs the infected image frame with the parallel numeric GPS coordinates received at that exact timestamp. The image file is then saved directly into the local PC file system. Each saved image file is named using its corresponding digital geographical coordinates as a textual string identifier (e.g., 33.400752_6.859116.jpg) to serve as a fixed digital reference for the infection.

3. Visualization layer (Layer 3)

This layer consists of an interactive Web dashboard designed as a real-time passthrough display to monitor the ongoing status of the field, without retrieving or displaying any previously archived files. The dashboard is divided into the following functional components:

Live stream & inference display: It displays the incoming frames from the ESP32-CAM node in real time, with the YOLOv8 bounding boxes, disease classifications, and confi-

dence scores superimposed on the current image. These images update automatically as the drone moves, leaving no footprint on the dashboard interface once they pass.

Real-time metrics dashboard: It provides a live textual and graphical display of the current climate metrics (temperature and humidity from the DHT11 sensor) paired with the current digital geographical coordinates (Lat, Long) received from the GPS module.

Decoupled storage logic: When an infection is detected, the automated backend script quietly archives the image using its coordinates as the file name inside the local PC folder. This historical log or saved files are never displayed on the web dashboard interface, keeping the user interface extremely lightweight and entirely focused on live, high-speed field surveying.

Manual upload module: An independent interactive module that allows users to manually upload plant images from their computer (via a drag-and-drop interface) to pass them directly to the YOLOv8 model for isolated, on-demand analysis.

III.4 Dataset preparation

In this section, we define the default configuration which the yolov8 model will adhere to during training. The data.yaml file is used to tell the model where the dataset will be found and what classes the model should detect.

Here's the content of our data.yaml file:

```
train: "/content/d/Myd/models/train/images"
val: "/content/d/Myd/models/valid/images"
test: "/content/d/Myd/models/test/images"
nc: 29
names: ['Apple Scab Leaf', 'Apple leaf', 'Apple rust leaf', '
  Bell_pepper leaf spot', 'Bell_pepper leaf', 'Blueberry leaf', '
  Cherry leaf', 'Corn Gray leaf spot', 'Corn leaf blight', 'Corn
  rust leaf', 'Peach leaf', 'Potato leaf early blight', 'Potato
  leaf late blight', 'Potato leaf', 'Raspberry leaf', 'Soyabean
  leaf', 'Squash Powdery mildew leaf', 'Strawberry leaf', 'Tomato
  Early blight leaf', 'Tomato Septoria leaf spot', 'Tomato leaf
  bacterial spot', 'Tomato leaf late blight', 'Tomato leaf mosaic
  virus', 'Tomato leaf yellow virus', 'Tomato leaf', 'Tomato mold
  leaf', 'Tomato two spotted spider mites leaf', 'grape leaf black
  rot', 'grape leaf']
```

Code explanation:

1. **Dataset paths:** These three lines tell us where the dataset is in Google d (since I was working with Google Colab):

train: Path to the training images.

val: Path to the validation images, which are utilized for validating the model's performance while training.

test: Path to the test images, which are used for final evaluation after training is finished.

2. Number of classes:

```
nc:29
```

(nc) stands for "number of classes". In this project, I'm training the yolov8 model to recognize 29 different categories, so I set it to 29.

3. Class names: These are the names of the object classes the model will learn to detect.

```
names: ['Apple Scab Leaf', 'Apple leaf', 'Apple rust leaf', '
    Bell_pepper leaf spot', 'Bell_pepper leaf', 'Blueberry leaf',
    'Cherry leaf', 'Corn Gray leaf spot', 'Corn leaf blight', '
    Corn rust leaf', 'Peach leaf', 'Potato leaf early blight', '
    Potato leaf late blight', 'Potato leaf', 'Raspberry leaf', '
    Soyabean leaf', 'Squash Powdery mildew leaf', 'Strawberry
    leaf', 'Tomato Early blight leaf', 'Tomato Septoria leaf spot
    ', 'Tomato leaf bacterial spot', 'Tomato leaf late blight', '
    Tomato leaf mosaic virus', 'Tomato leaf yellow virus', '
    Tomato leaf', 'Tomato mold leaf', 'Tomato two spotted spider
    mites leaf', 'grape leaf black rot', 'grape leaf']
```

Each line represents a different category.

This naming helps YOLO label each detection with the correct class name while training or running inference.

III.4.1 Handling class imbalance

As shown in Table III.1, the dataset exhibits class imbalance. For example, Blueberry leaf contains 2,177 samples while Corn rust leaf contains only 399 samples. To mitigate this issue, the following strategies were applied during training:

- **Class weighting:** The built-in class weighting mechanism in YOLOv8 was used to assign higher importance to underrepresented classes during loss calculation.
- **Data augmentation:** Additional augmentation techniques (random rotation, horizontal flipping, scaling, and brightness adjustment) were applied specifically to minority classes to artificially increase their representation.

Despite these efforts, class imbalance remains a limitation of the current study. Future work will focus on collecting more field images for underrepresented disease categories, particularly those affecting crops in the El Oued region.

Table III.1: Distribution of annotated instances per class

ID	Class Name	Instances Count
0	Apple scab leaf	474
1	Apple leaf	666
2	Apple rust leaf	573
3	Bell_pepper leaf spot	750
4	Bell_pepper leaf	819
5	Blueberry leaf	2,177
6	Cherry leaf	591
7	Corn gray leaf spot	201
8	Corn leaf blight	1,128
9	Corn rust leaf	399
10	Peach leaf	1,833
11	Potato leaf early blight	51
12	Potato leaf late blight	981
13	Potato leaf	729
14	Raspberry leaf	1,743
15	Soyabean leaf	738
16	Squash powdery mildew leaf	753
17	Strawberry leaf	1,440
18	Tomato early blight leaf	692
19	Tomato septoria leaf spot	1,131
20	Tomato leaf bacterial spot	992
21	Tomato leaf late blight	807
22	Tomato leaf mosaic virus	666
23	Tomato leaf yellow virus	930
24	Tomato leaf	2,366
25	Tomato mold leaf	741
26	Tomato two spotted spider mites leaf	9
27	Grape leaf black rot	594
28	Grape leaf	300
Total Annotated Labels		25,274

III.4.2 Code training

```
from google.colab import d
```

```

d.mount('/content/d')
!pip install ultralytics
from ultralytics import YOLO
model = YOLO("yolov8n.pt")
model.train(
    data="/content/d/Myd/dataset/data.yaml",
    epochs=50,
    imgsz=640,
    batch=16,
    name="plant_disease_model",
    project="/content/d/Myd/runs"
)

```

This table provides III.2 an explanation of the training code.

Table III.2: Explanation of the training code

Parameter / Command	Explanation
<code>from google.colab import d</code>	Imports the google drive module.
<code>d.mount('/content/d')</code>	Mounts Google drive to the environment.
<code>!pip install ultralytics</code>	Installs the ultralytics package.
<code>model = YOLO("yolov8n.pt")</code>	Loads the YOLOv8 Nano pre-trained model.
<code>data="/content/d/Myd/data.yaml"</code>	Path to the dataset configuration file.
<code>epochs=50</code>	Number of complete passes during training.
<code>batch=12</code>	Number of images processed per step.
<code>imgsz=640</code>	Images are resized to 640×640 pixels.
<code>project="/d/Myd/training"</code>	Directory where results will be stored.
<code>name="plant_disease_model"</code>	Subfolder name for this specific run.

III.5 Esp32 telemetry and data transmission code

This code executes on the standard esp32 microcontroller, operating as the telemetry gateway for the drone's payload. It interfaces with the dht11 sensor to capture microclimate metrics and reads raw geospatial streams from the gps module via hardware serial communication. The microcontroller is programmed to handle data synchronization asynchronously through a non-blocking timer interval. Upon data retrieval, the esp32 encapsulates the identical coordinates (latitude and longitude) and environmental data (temperature and humidity) into the https headers of an encrypted post request, transmitting them wirelessly over wifi to the central Python

server via an ngrok tunnel, facilitating real time spatial logging and synchronization alongside the yolov8 vision framework.

1. Global configurations and hardware pins mapping

```
char ssid[] = "your ssid";
char pass[] = "your password";
const char* serverName = "https://splendidly-intervocalic-isabel.
    ngrok-free.dev/upload_esp";
BlynkTimer timer;
TinyGPSPPlus gps;
HardwareSerial SerialGPS(2);
#define DHTPIN 15
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
float temperature = 0;
float humidity = 0;
```

- **Network and gateway credentials:** Global variables that store the local wifi router network name (SSID), password, and the static endpoint uniform resource locator (URL) generated by the secure Ngrok tunnel.
- **SerialGPS(2):** Initializes the second independent hardware serial peripheral inside the esp32 chip (UART2), keeping the GPS stream separated from the default computer serial interface (UART0).
- **Sensor Pins Assignment:** Binds the DHT11 climate data line to digital pin general purpose input/output (GPIO) 15 and explicitly defines the microclimate sensor model type.

2. **3. Asynchronous telemetry processing (void sendData()):** This function serves as the central data aggregator and transmission engine, executing three main processes:

- **Environmental microclimate acquisition**

```
float h = dht.readHumidity();
float t = dht.readTemperature();
if (!isnan(h) && !isnan(t)) {
    temperature = t;
    humidity = h;
}
```

The microcontroller reads relative humidity and ambient temperature values. It utilizes the academic safety function `isnan()` (is not a number) to filter out corrupted readings caused by physical wire vibrations during drone flight.

- **Geospatial coordinates processing**

```
if (gps.location.isValid()) {  
    float lat = gps.location.lat();  
    float lng = gps.location.lng();  
    Serial.printf("GPS Fix! Lat: %.6f, Lng: %.6f\n", lat,  
        lng);  
}
```

The *Valid()* method evaluates whether the GPS module has established a stable connection with orbiting satellites. Once a lock is confirmed, it extracts latitude and longitude values up to 6 decimal places of precision.

- **Network packet compression and HTTPS post routine**

```
if (WiFi.status() == WL_CONNECTED) {  
    WiFiClientSecure client;  
    client.setInsecure();  
    HTTPClient http;  
    http.begin(client, serverName);  
    http.addHeader("Temperature", String(temperature, 1));  
    http.addHeader("Humidity", String(humidity, 1));  
    http.addHeader("Lat", gps.location.isValid() ? String(gps.  
        location.lat(), 6) : "0.0");  
    http.addHeader("Lng", gps.location.isValid() ? String(gps.  
        location.lng(), 6) : "0.0");  
    http.addHeader("ngrok-skip-browser-warning", "true");  
    http.addHeader("Content-Type", "application/json");  
    int httpResponseCode = http.POST("{}");  
    http.end();  
}
```

client.setInsecure(): A critical performance optimization that instructs the micro-controller to bypass the heavy computation of SSL certificate chain verification, minimizing transmission delay.

HTTP headers injection: Rather than consuming the drone's edge processing power to compile complex JSON files, this design injects the coordinates and weather metrics directly into the HTTP Headers. The request is then dispatched as a lightweight empty body payload *POST("")*. This methodology significantly shrinks the data packet size over the wireless connection and simplifies real-time server-side unpacking.

3. System initialization (void setup())

```
void setup() {  
  Serial.begin(115200);  
  SerialGPS.begin(9600, SERIAL_8N1, 16, 17);  
  dht.begin();  
  timer.setInterval(5000L, sendData);  
}
```

SerialGPS.begin(...): Activates the GPS universal asynchronous receiver-transmitter (UART) peripheral at a standard baud rate of 9600 and configures the physical pin connections to $RX = 16$ and $TX = 17$.

timer.setInterval(5000L, sendData): Implements a non blocking scheduling routine. It establishes a fixed interval of 5 seconds (5000 ms) to invoke the transmission function, completely replacing the traditional *delay()* function which would otherwise cause critical network data loss.

4. Continuous execution loop (void loop())

```
void loop() {  
  timer.run();  
  
  while (SerialGPS.available() > 0) {  
    gps.encode(SerialGPS.read());  
  }  
}
```

timer.run(): Continuously monitors the internal clock cycles of the CPU to determine when to trigger the scheduled transmission function.

gps.encode(...): A highly critical processing loop that constantly listens for incoming byte streams sent by the GPS hardware. It feeds these bytes into the encoder function immediately, guaranteeing that the drone's spatial mapping coordinates remain fully synchronized in real-time with zero data latency during flight. Figure III.6 shows the parallel hardware architecture of the drone payload, illustrating how the esp32 captures telemetry (GPS and DHT11) while the esp32cam streams real time images, both sending data concurrently to the web server via an Ngrok tunnel over wifi.

III.6 Communication and system integration

To facilitate user interaction and data visualization, a dedicated web application was developed for real-time image processing and telemetry monitoring. The web platform establishes an asynchronous framework to simultaneously receive live image streams from the ESP32-CAM and

environmental data from the standard ESP32 microcontroller. To bridge these edge devices with the server across different networks, cloud communication via a secure cloud tunneling service (ngrok) was implemented. This setup exposes the local Flask backend via an encrypted HTTPS endpoint, bypassing local firewalls and Network Address Translation (NAT). Upon receiving the data, the system processes the streams using the YOLOv8 model for instantaneous disease detection, displaying the live feed with dynamic bounding boxes alongside synchronized real-time sensor logs.

III.6.1 Backend workflow flowchart

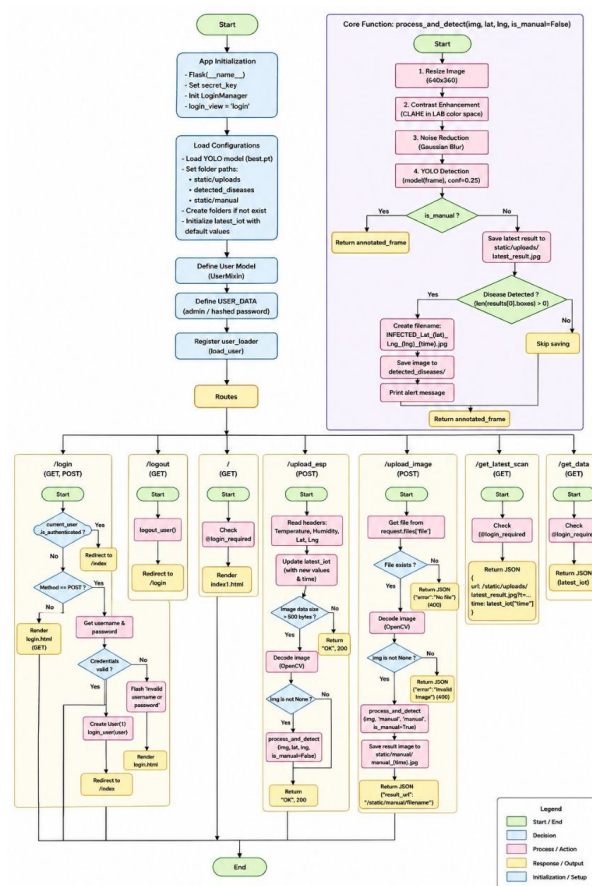


Figure III.6: Flowchart of the backend system architecture and data flow

III.7 Hardware architecture and system interconnection diagram

Figure III.7 shows the parallel hardware architecture of the drone payload, illustrating how the esp32 captures telemetry (GPS and DHT11) while the esp32 cam streams real time images, both sending data concurrently to the web server via an Ngrok tunnel over wifi.

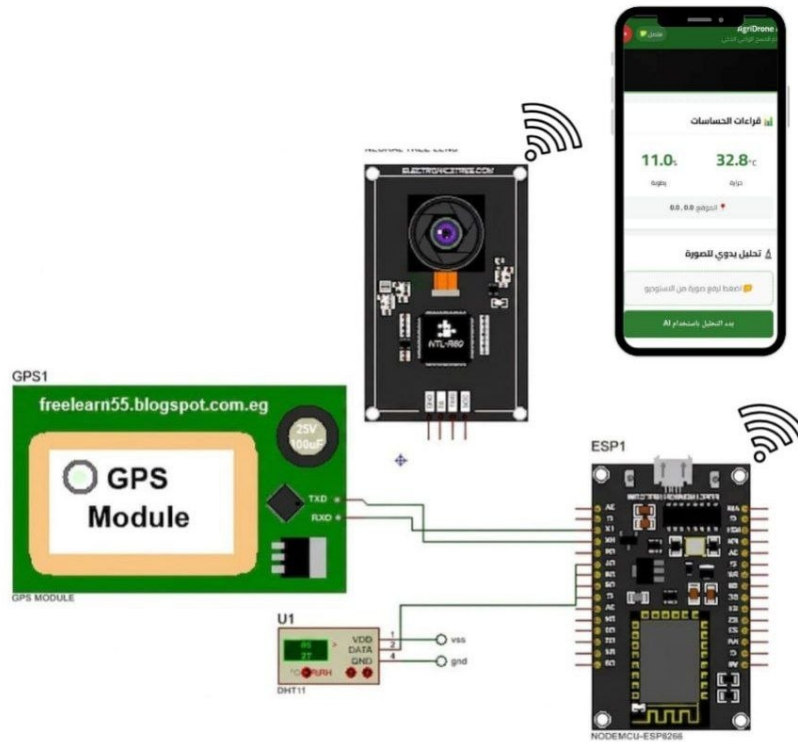


Figure III.7: Electronic composition

The comprehensive hardware architecture and pin configurations for the integrated drone subsystems are detailed in table III.3. As shown, the GPS module is interfaced via the dedicated hardware serial peripheral (UART2) on pins GPIO 16 and GPIO 17 to ensure continuous geospatial tracking. Simultaneously, the DHT11 climate sensor occupies the digital pin GPIO 15 for environmental telemetry. Lastly, the esp32 cam (OV2640 camera sensor) operates via the high speed internal parallel bus of the esp32 cam board, eliminating the need for external wiring and optimizing power routing.

Table III.3: Comprehensive hardware wiring and pin configuration for the drone subsystems

Subsystem	Function	From (Component pin)	To (Microcontroller)
GPS module	Power supply	VCC	5V / VIN (ESP32)
	Ground	GND	GND (ESP32)
	UART transmission	TX	GPIO 16 / RX2 (ESP32)
	UART reception	RX	GPIO 17 / TX2 (ESP32)
DHT11 sensor	Power supply	VCC	3.3V / 5V (ESP32)
	Ground	GND	GND (ESP32)
	Digital data signal	DATA / OUT	GPIO 15 (ESP32)
ESP32-CAM	Power supply	VCC	3.3V / 5V (Internal board power)
	Ground	GND	GND (Internal board ground)
	Timing synchronization	VSYNC, HREF, PCLK	GPIO 25, 23, 22 (ESP32-CAM)

III.8 Security mechanisms

To ensure the confidentiality and integrity of agricultural data, the proposed system implements multiple layers of security. First, all communications between the drone payload (ESP32-CAM and ESP32) and the backend server are encrypted using the HTTPS protocol via a secure ngrok tunnel. This prevents eavesdropping and man-in-the-middle attacks on sensitive data such as plant images, GPS coordinates, temperature, and humidity readings.

Second, user authentication is protected by hashing the administrator password using the `werkzeug.security` library. The password is never stored in plain text; instead, a cryptographic hash (PBKDF2 with SHA-256) is saved on the server. During login, the submitted password is verified against this hash without ever decrypting the original password.

Third, session management is secured using Flask-Login with a secret key that encrypts session cookies, preventing session hijacking and cookie tampering. These combined mechanisms establish a robust security posture suitable for smart agricultural applications where data privacy is paramount.

Conclusion

In this chapter, we have presented the complete practical design and implementation of the proposed intelligent payload. We have described the hardware components (ESP32, ESP32-CAM, GPS, DHT11) and software tools (Google Colab, Arduino IDE, Ultralytics, OpenCV) used in the system. We have explained the system architecture and its three layers (data acquisition, processing logic, and visualization), as well as the communication protocols including HTTPS encryption via ngrok tunnel. The programming codes for the ESP32-CAM image streaming and ESP32 telemetry transmission have been detailed. The experimental results, including model training performance, inference accuracy, latency measurements, and user interface evaluation, are presented in the following chapter.

Chapter IV

Experimental results and discussion

Introduction

This chapter presents the experimental results obtained from the implementation of the proposed intelligent UAV payload for plant disease detection. It provides a comprehensive evaluation of the YOLOv8 model's training performance, inference accuracy, and real-time deployment capabilities. Furthermore, this chapter analyzes the end-to-end system latency, user interface responsiveness across desktop and mobile platforms, and the practical handling of false positives and false negatives in agricultural field conditions.

IV.1 Results of training

The figure IV.1 presents an exploratory analysis of the dataset used to train the proposed plant disease detection model. This analysis aims to evaluate dataset quality, examine class balance, and analyze the spatial and geometric characteristics of annotated objects prior to the model training phase.

IV.1.1 Class distribution analysis

The bar chart illustrates the number of annotated instances per class within the dataset. The distribution reveals noticeable variation in sample counts across different disease categories, indicating a partially imbalanced dataset. Class imbalance is a critical factor in deep learning based object detection tasks, as models may become biased toward highly represented classes. Therefore, understanding class distribution enables the application of appropriate mitigation strategies such as data augmentation, class weighting, or balanced sampling during training to improve model generalization.

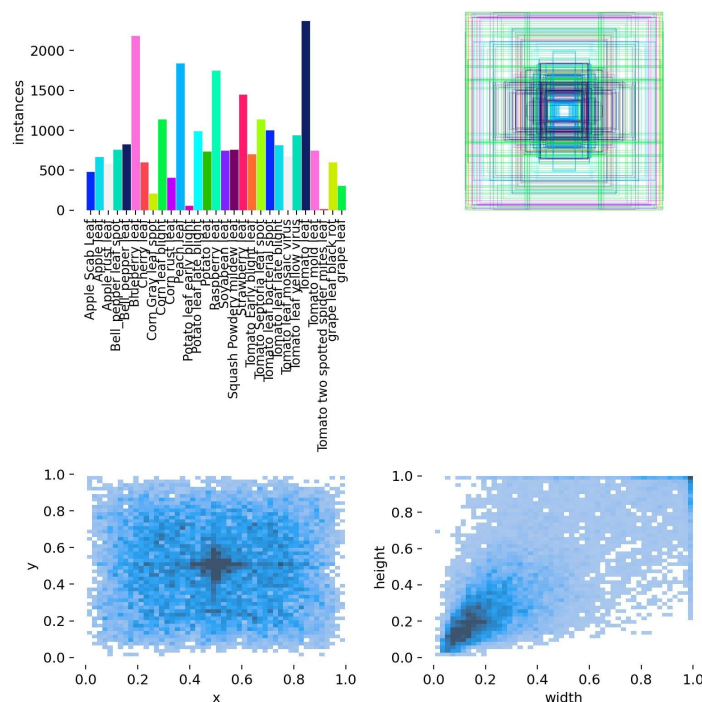


Figure IV.1: Dataset analysis and annotation distribution report

1. **Bounding box overlay visualization:** The upper-right visualization shows all normalized bounding boxes superimposed into a single coordinate space. The overlay indicates that most annotated objects are concentrated near the center of the images. This spatial consistency suggests that image acquisition was performed under semi-controlled conditions, where the target leaf or infected region was intentionally centered within the frame. Such consistency can facilitate faster model convergence and improve detection stability during training.
2. **Object center spatial distribution:** The heatmap presented in the lower-left plot represents the distribution of bounding box center coordinates (X, Y). A high density region appears around the image center, confirming that objects predominantly occupy central areas rather than image boundaries. Reduced spatial variance simplifies the localization task for the detection network and contributes to improved robustness and localization accuracy.
3. **Bounding box size distribution:** The lower right heatmap analyzes the relationship between normalized bounding box width and height. The results indicate that most annotated objects fall within small to medium size ranges, while large scale objects are less frequent. Understanding object scale distribution is essential for selecting appropriate detection parameters, including anchor box sizes, input resolution, and multiscale feature extraction settings, which collectively enhance detection performance across varying object scales.

IV.2 Training and validation performance analysis

To evaluate the learning behavior and detection performance of the proposed object detection model, the training process was analyzed over 50 epochs using loss functions and standard object detection metrics.

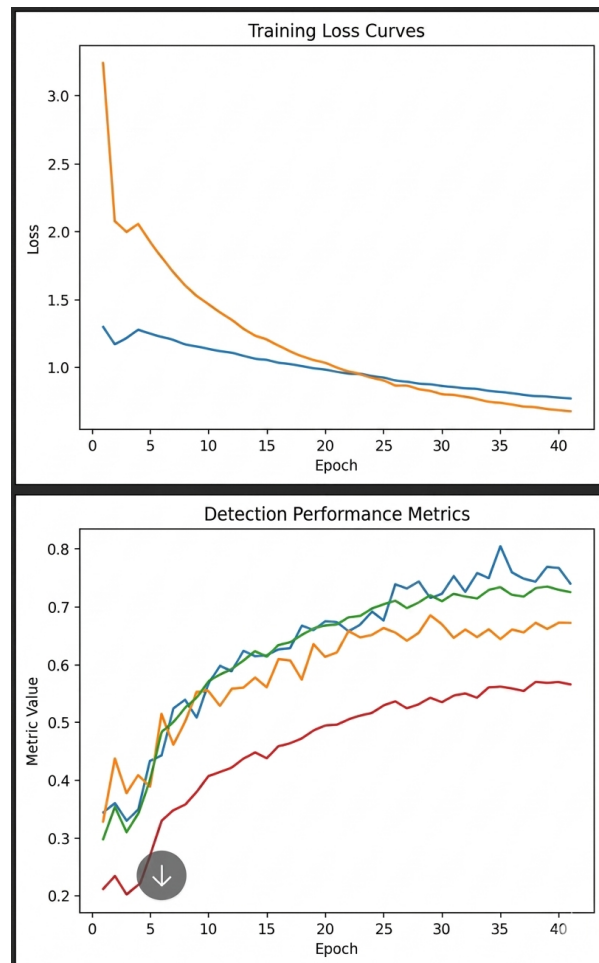


Figure IV.2: Training loss and detection performance

IV.2.1 Training loss analysis

The empirical evaluation of the model's optimization pathway, as depicted in the Training loss curves graph, demonstrates a highly stable and successful learning process. Throughout the 50 training epochs, all loss components exhibit a consistent, monotonic decay, signifying effective gradient descent and parameter optimization. Critically, the continuous convergence of these loss functions indicates that the network is systematically minimizing errors in object localization, class prediction, and bounding box regression without experiencing gradient instabilities. This steady downward trajectory confirms that the model is efficiently capturing and encoding foundational visual features from the training dataset.

IV.2.2 Validation metrics analysis

The technical analysis of the performance curves in the second graph reveals a steady and progressive evolution of the key object detection evaluation metrics across the training epochs. This upward trajectory is further validated by the empirical results consolidated in Table IV.1. Specifically, the precision metric exhibits a continuous increase, culminating in a value of 0.7426, which underscores the detector's capability to minimize false positives and generate highly accurate predictions.

Concurrently, the recall metric demonstrates a robust upward trend, reaching 0.6722, thereby indicating an enhanced capacity for target discovery and an optimal disease detection rate. This balanced learning behavior is directly reflected in the continuous improvement of the mean Average Precision; the standard metric (mAP@50) achieves a solid accuracy score of 0.7319, while the more stringent and comprehensive metric (mAP@50-95) shows a gradual, stable increment to 0.5683, proving highly robust localization performance. Ultimately, the simultaneous advancement of both precision and recall yields a well-balanced optimization trade-off, demonstrating that the model successfully enhances detection confidence while expanding its overall object coverage.

Table IV.1: Overall model accuracy results

Metric	Value
Precision	0.7426
Recall	0.6722
mAP@50	0.7319
mAP@50-95	0.5683

IV.2.3 Overall model behavior

Based on the obtained results, the model successfully converges during training. No instability or overfitting symptoms are observed, as detection accuracy improves consistently with training progression. The model shows strong potential for real-world computer vision deployment.

IV.3 Inference results

Figure IV.3 presents a compilation of inference results generated by the trained YOLOv8 model, demonstrating its practical capability in detecting and localizing various plant leaf diseases. The model successfully processes multiple distinct plant categories simultaneously including tomato, corn, bell pepper, grape, strawberry, raspberry, and peach leaves under diverse environmental conditions, lighting variances, and background complexities. The high density of precise bounding boxes, such as those accurately mapping out multiple overlapping regions of Tomato leaf yellow virus and Corn leaf blight, visually confirms the model's high localization

accuracy. This robust visual output validates the prior statistical performance metrics, proving that the YOLOv8 architecture has effectively generalized across multi-class plant pathology datasets and holds strong potential for field-ready automated agricultural diagnosis.



Figure IV.3: Visual inspection of plant disease detection results via the trained YOLOv8 model

IV.4 Experimental results and deployment

This section presents the practical validation and deployment phase of the developed system. It provides an empirical evaluation of the YOLOv8 deep learning model's performance, alongside a comprehensive overview of the agricultural monitoring platform's user interfaces.

IV.4.1 Physical implementation

Figure IV.4 illustrates the preliminary physical implementation of the embedded hardware architecture, developed strictly as a laboratory bench prototype for proof-of-concept (PoC) validation. The dual-microcontroller setup utilizes a standard ESP32 core paired with an integrated ESP32-CAM module, both mounted on a temporary prototyping breadboard to verify parallel firmware execution and multi-sensor synchronization. The framework links a serial-UART GPS module for simulated spatial mapping and a DHT11 digital sensor for microclimate telemetry. Concurrently, the ESP32-CAM captures and encodes visual streams to evaluate real-time transmission stability over local wireless links. Crucially, this structural arrangement is a non-airborne engineering model built exclusively for laboratory debugging and data latency optimization; it remains fundamentally distinct from the finalized, weight-optimized, and PCB-fabricated configuration designed for actual drone deployment.

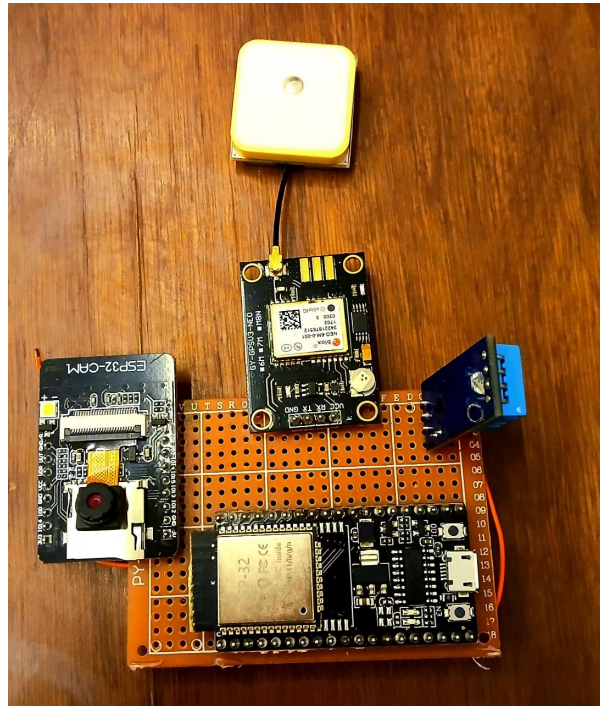


Figure IV.4: Physical implementation of the embedded hardware architecture

IV.4.2 Login interface

The login page represents the first layer of security for the proposed agricultural monitoring system. As illustrated in Figure IV.5, the user must provide a valid username and password. The credentials are verified server-side against a securely hashed password using the `werkzeug.security` library. Upon successful authentication, Flask login establishes a user session, granting access to the real-time monitoring dashboard, disease detection results, and manual image upload features. This authentication mechanism ensures that only authorized personnel (e.g., farmers, agronomists, or system administrators) can access sensitive agricultural data, drone telemetry, and infected plant records, thereby protecting the confidentiality and integrity of the system.

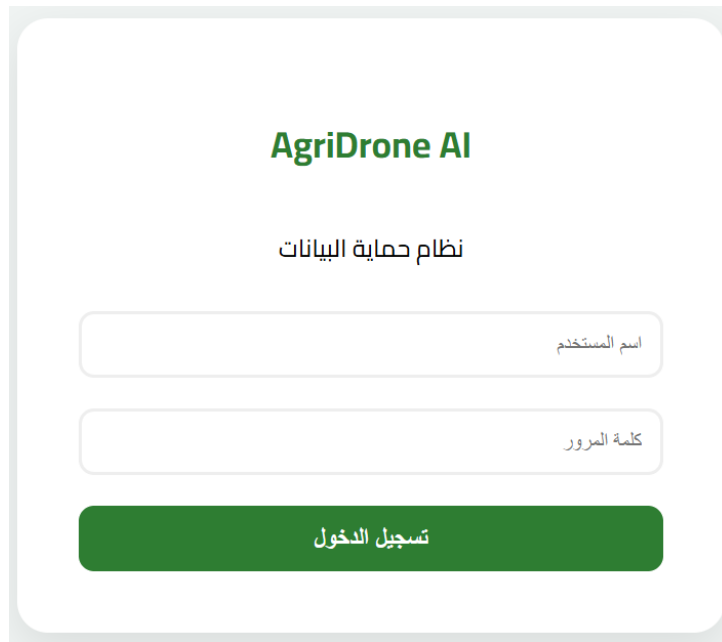


Figure IV.5: Screenshot of the login interface

IV.4.3 Desktop website interface

Figure IV.6 presents a screenshot of the developed ground control station dashboard (AgriDrone AI) captured from a desktop client interface. The web platform visually integrates incoming data streams for real-time monitoring; the left panel provides automated data logging for environmental telemetry (Temperature at 33.6°C and Humidity at 5.0%) synchronized alongside active GPS coordinate tracking. Simultaneously, the main streaming window outputs the live HTTPS video stream from the airborne payload, showing an instantaneous YOLOv8 inference where a Tomato leaf mosaic virus is successfully isolated with a confidence score of 0.77.

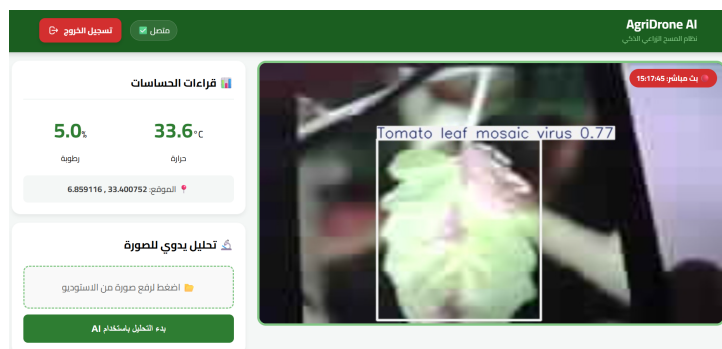


Figure IV.6: Screenshot of the website on a desktop client

IV.4.4 Mobile web interface

Figure IV.7 illustrates a screenshot of the web application's responsive interface rendered on a mobile screen via a secure Ngrok tunnel. The frontend framework dynamically rearranges

the system elements to preserve visual clarity on smaller edge-device screens. The architectural pipeline remains fully synchronized, maintaining the live image feed from the drone while concurrently updating the climate metrics (Temperature at 34.8°C and Humidity at 5.0%) and spatial coordinates without any observed latency.



Figure IV.7: Screenshot of the website on a mobile device

IV.4.5 System latency measurements

To evaluate the real-time performance of the proposed system, end-to-end latency was measured over 34 consecutive image transmissions from the ESP32-CAM to the server. Table IV.2 summarizes the average processing times for each stage.

Table IV.2: End-to-end system latency measurements (based on 34 samples)

Parameter	Average Value
Preprocessing Time (resizing)	1.04 ms
YOLOv8 Inference Time	243.75 ms
Total Processing Time	245.85 ms
Total Request Time (including network)	287.41 ms
Frames per Second (FPS)	4.07 fps

The results show that the YOLOv8 model processes each image in approximately 244 ms on a standard CPU without GPU acceleration, which is acceptable for real-time agricultural monitoring applications where a delay of less than one second is sufficient for early disease detection. The network transmission overhead adds approximately 42 ms, resulting in a total request time of 287 ms. This enables a frame rate of about 4 frames per second, allowing the drone to cover large agricultural areas efficiently.

IV.5 Decision making

Agricultural personnel (e.g., engineers or farmers) can make timely, effective decisions based on the analysis results, such as:

- Quarantine of infected plants to prevent disease transmission.
- Specific use of pesticides.
- Adjustment of irrigation, fertilization, or other treatment procedures as needed.

Benefits of computer vision for this application:

High accuracy: Capable of detecting diseases at early stages, improving treatment efficiency.

Real-time response: Enables instant feedback and response in field operations.

Wide-area monitoring: One drone is able to monitor vast agricultural areas in a short period of time.

Operational consistency: Unlike human inspectors, the system is not subject to fatigue, prejudice, or inconsistencies in judgment.

IV.5.1 Handling false positives and false negatives

Despite the model achieving good accuracy (Precision: 0.7426, Recall: 0.6722), false positives (healthy leaves incorrectly classified as diseased) and false negatives (diseased leaves missed) may still occur in real-world conditions. To address this:

- **Confidence threshold adjustment:** The system uses a confidence threshold of 0.25 for detection. Detections with confidence below 0.7 are flagged for manual verification by an agronomist.

- **Human-in-the-loop approach:** Pesticide application based solely on model predictions without human confirmation is not recommended. A qualified agronomist should verify high-confidence detections before any treatment.
- **Continuous improvement:** All detected cases (both true and false) are logged for future model retraining and fine-tuning.

This hybrid approach ensures crop safety, reduces unnecessary pesticide use, and improves the reliability of the automated system.

IV.6 Limitations and future work

The thesis title refers to an agricultural explorer drone as the target deployment platform. Nevertheless, the practical implementation presented in this work is confined to the development and validation of an intelligent payload (sensors, processing unit, and communication modules) intended for subsequent drone integration. Two primary factors justify this scoping decision. From a technical perspective, independent bench validation of the payload is a prerequisite for reliable flight integration, since airborne conditions introduce non-linear disturbances (mechanical vibrations, electromagnetic interference, and thermal stress) that would obscure the interpretation of baseline performance metrics such as inference latency and detection accuracy. From a regulatory perspective, Algerian civil aviation law (Law No. 98-06, amended by Law 15-11) mandates prior security authorization for any UAV flight in agricultural zones. Consequently, physical drone integration and in-flight validation are addressed as future work. To build upon the verified proof-of-concept prototype developed in this thesis, several engineering enhancements and strategic research directions are proposed for future development:

- **strategic focusing on saharan agriculture development:** Future research will heavily focus on specializing this system for Saharan agro-ecosystems, which currently represent the largest and most critical agricultural frontiers requiring technological modernization. Special emphasis will be placed on the El-Oued region, which has emerged as a national leader in the production of strategic crops (such as potatoes and dates). Despite its immense agricultural yield, this region faces severe environmental and logistical challenges, including extreme thermal stress, high soil and groundwater salinity, rapid water evaporation in sandy terrains, and the spread of endemic pests. Therefore, optimizing the drone's AI capabilities and expanding the IoT sensor network will be tailored specifically to mitigate these Saharan challenges, ensuring sustainable water management and early disease intervention to protect this vital economic zone.
- **Edge computing optimization:** Utilizing hardware-accelerated edge devices (such as NVIDIA Jetson Nano or Raspberry Pi) on the drone to run compressed versions of the

AI model (e.g., converted to tensor regression tree (TensorRT) or open neural network exchange (ONNX) formats). This enables on-board inference, significantly reducing network bandwidth consumption and transmission latency.

- **Autonomous airborne integration:** Upgrading the non-airborne laboratory benchmark prototype into a compact, weight-optimized Printed Circuit Board (PCB) integrated within an autonomous quadcopter. Future iterations will implement automated flight path planning (waypoint navigation) using advanced GPS modules.
- **Expansion of the IoT sensor framework:** Future work will focus on integrating advanced telemetric sensors to optimize desert farming management. This includes deploying specialized digital Soil Moisture sensors to track water retention in sandy soils, Electrical Conductivity (EC) sensors to monitor groundwater salinity levels, and Solar Radiation/UV sensors to measure heat stress on crops under extreme Saharan climate conditions.
- **Complete cloud infrastructure migration:** Transitioning the system from the current local-tunneling architecture (via ngrok) to a fully dedicated cloud computing infrastructure (e.g., amazon web services (AWS) or Google Cloud). This will facilitate big data analytics, continuous database logging, and multi-drone fleet management.

Conclusion

This chapter presented the experimental validation of the proposed UAV payload for plant disease detection. The results confirmed that the YOLOv8 model successfully detects multiple disease classes with reliable accuracy, and the integrated hardware, including the ESP32-CAM tested during actual flight, demonstrated feasible real-time operation despite challenges related to vibration, wireless range, and lighting conditions. The web dashboard provided secure and responsive visualization across devices. These findings collectively validate the proof-of-concept. However, the remaining subsystems (GPS, DHT11, and telemetry ESP32) await full flight integration, and the identified challenges provide clear directions for future work. The roadmap outlined in this chapter charts a realistic path toward a fully deployable system for Saharan agriculture.

General conclusion

In conclusion, this research demonstrates that the integration of digital image processing and Deep Learning (DL) for plant disease detection represents a promising paradigm shift toward intelligent agricultural solutions. The synergy between these technologies, the Internet of Things (IoT), and Unmanned Aerial Vehicles (UAVs) enables an automated and highly efficient crop monitoring framework.

This project successfully developed a robust model capable of analyzing foliar imagery and diagnosing pathologies with high precision, thereby facilitating early intervention, mitigating agricultural losses, and optimizing production yields. Furthermore, the work underscores the pivotal role of UAVs in streamlining large-scale data acquisition compared to conventional methods.

While the results are encouraging, future work should focus on expanding the model's diagnostic scope to include a broader range of crops and diseases, enhancing its robustness under diverse environmental conditions, and implementing real time edge processing for immediate feedback to farmers.

Ultimately, this project serves as a foundational step toward an integrated, AI-driven smart farming ecosystem that supports precision agriculture and global food security.

References

- [1] K. Fezzai, A. Moussa, and A. Lekhouimes, ``Enquête des ravageurs et maladies de la culture de pomme de terre dans la région d'el oued," mémoire de master, Université d'El-Oued, El-Oued, Algérie, 2021.
- [2] G. Gupta and S. K. Pal, ``Applications of ai in precision agriculture," *Springer*, 2025.
- [3] D. O. Alakuş, ``Smart agriculture, precision agriculture, digital twins in agriculture," *IEEE*, 2024.
- [4] W. Shafik *et al.*, ``Using transfer learning-based plant disease classification," *BMC Plant Biology*, 2024.
- [5] P. K. Dutta *et al.*, eds., *Harnessing AI to Reshape the Future of Agriculture*. Springer Nature, 2026.
- [6] V. C. Mahavishnu *et al.*, ``Drone based precision agriculture technique," *IEEE*, 2025.
- [7] G. P. Panem *et al.*, ``An analytical study of iot architectures for precision agriculture," *IEEE*, 2025.
- [8] M. Lishchynskyi *et al.*, ``Iot and uav imaging for indoor plant disease detection," *IEEE*, 2024.
- [9] J. A. J. Berni, P. J. Zarco-Tejada, L. Suárez, and E. Fereres, ``Evaluating multispectral images and vegetation indices for precision farming applications from uav images," *Remote Sensing*, vol. 7, no. 4, pp. 4026--4047, 2015.
- [10] V. A. Palan *et al.*, ``Leveraging super-resolution technology in drone imagery," *IEEE Access*, 2025.
- [11] G. P. Ardinata and A. Pratiarso, ``Mapping of agricultural land based on drone," *IEEE*, 2025.
- [12] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

-
- [13] W. University, Research, and U. H. L. d'El Oued, "Sustainable water use in potato production in algeria," tech. rep., Wageningen University & Research, 2020. Field study on potato cultivation in sandy soils, El Oued, Algeria. Reference file: `sustainable_water_use_in_potato_production_in_algeria.pdf`.
- [14] K. Al-Dabash, *Simply OpenCV: Your Way to Learning Computer Vision and Digital Image Processing*. Self-Published / Independent, 2017.
- [15] GeeksforGeeks, "Image resizing using opencv | python." <https://www.geeksforgeeks.org/python/image-resizing-using-opencv-python/>, 2025. Retrieved March 11, 2026.
- [16] GeeksforGeeks, "Python opencv - cv2.cvtColor() method." <https://www.geeksforgeeks.org/python-opencv-cv2-cvtColor-method/>, 2025. Retrieved March 11, 2026.
- [17] Google Research, "Google colaboratory frequently asked questions." <https://research.google.com/colaboratory/faq.html>, 2026. Accessed: 2026-05-21.
- [18] Arduino CC, "Arduino official documentation and reference." <https://docs.arduino.cc>, 2026. Accessed: 2026-05-21.
- [19] Ultralytics Inc., "Ultralytics yolov8 documentation." <https://docs.ultralytics.com>, 2026. Accessed: 2026-05-21.
- [20] OpenCV Documentation, "Contours: Getting started." https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html, 2024.
- [21] NumPy Developers, "Numpy documentation reference." <https://numpy.org/doc/>, 2026. Accessed: 2026-05-21.
- [22] Microsoft, "Visual studio code documentation." <https://code.visualstudio.com/docs>, 2026. Accessed: 2026-05-21.
- [23] Ngrok Inc., "Ngrok: Secure ingress as a service for developers." <https://ngrok.com/docs>, 2026. Accessed: 2026.
- [24] Espressif Systems, *ESP32 Technical Reference Manual*. Espressif Systems, 2022. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.
- [25] R. Santos and S. Santos, *AI-Powered IoT Edge Devices: Deploying Machine Learning Models on ESP32 and ESP32-CAM*. Hands-On Internet of Things Series, Porto, Portugal: Maker Publishing, 2023.

-
- [26] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle, *GNSS—Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Vienna, Austria: Springer Science & Business Media, 2012.
- [27] Aosong Electronics, "Dht11 humidity & temperature sensor product manual," 2015. Available: <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>.