Available online at www.ijournalse.org

Emerging Science Journal

(ISSN: 2610-9182)

Vol. 9, No. 3, June, 2025

Smart Irrigation System with IoT, Machine Learning, and Solar Power for Efficient Plant Care

Justin M. A. Capcha-Ochoa [1], Jefferson A. Chahua-Benito [1], Miguel A. Serafin-Cayllahua [1], Sebastian E. Mamani-Martinez [1], Jesus G. Mendivil-Imbertis [1], Jordan I. Mendoza-Fernandez [1], Roberto J. M. Casas-Miranda [1], Maritza Cabana-Cáceres [1], Cristian Castro-Vargas [1*]

[1] *Facultad de Ingeniería y Arquitectura, Escuela de Profesional de Ingeniería de Sistemas, Universidad César Vallejo, Lima, Perú.*

Abstract

Efficient irrigation in green areas and homes is essential for environmental sustainability and water conservation. This study aims to develop an intelligent irrigation system based on the Internet of Things (IoT) and machine learning to optimize water use, improve plant monitoring, and enhance security. Two ESP32 microcontrollers and an ESP32-CAM were deployed to manage humidity, temperature, light sensors, and irrigation automation using a solenoid valve. A modified Yolov3-tiny model detects signs of dehydration and chlorosis in plants, while facial recognition restricts access to authorized users. Data is processed through IoT platforms such as Adafruit IO and Telegram, ensuring continuous solar-powered monitoring. Furthermore, integration with YouTube Live and Dropbox enables remote monitoring and intrusion detection. Experimental results indicate a 43.7% reduction in water consumption, efficient detection of plant problems (93.86% accuracy), and increased security. Based on AI and renewable energy, this innovative approach surpasses traditional systems and represents a scalable and sustainable solution for innovative irrigation management.

Keywords:

Smart Irrigation; ESP32; ESP32-CAM; Face Detection; Object Recognition; Solar Energy; IOT; Plant.

Article History:

Received:	18	December	2024
Revised:	16	May	2025
Accepted:	20	May	2025
Published:	01	June	2025

1- Introduction

Green areas are crucial in environmental sustainability and quality of life in urban and rural settings. However, the efficient maintenance of these areas faces significant challenges, such as water waste due to outdated irrigation systems and the lack of advanced technological strategies for optimizing water consumption [1]. According to the FAO, agriculture and traditional irrigation systems account for approximately 70% of freshwater consumption worldwide [2]. Water efficiency is a critical issue in Latin America, with studies reporting losses of 30% to 50% of water used in irrigation due to inefficient techniques [3]. In Peru, this problem is exacerbated by limited access to automated irrigation technologies, increasing water use costs and waste [4].

Various approaches have been proposed to improve irrigation efficiency through the Internet of Things (IoT) and machine learning. Previous research has shown that integrating sensors with microcontrollers such as the ESP32 allows irrigation to be optimized based on real-time environmental data [5]. However, most of these systems have three main limitations: (1) they rely solely on predefined thresholds to trigger irrigation without advanced plant health analysis [6]; (2) they lack AI-based visual sensing capabilities, which limits their ability to identify specific vegetation problems [7]; and (3) they do not effectively integrate renewable energy, which reduces their long-term sustainability [8-10].

* **CONTACT**: ccastrov13@ucvvirtual.edu.pe

DOI: http://dx.doi.org/10.28991/ESJ-2025-09-03-012

Several studies have attempted to address these issues. Majid et al. [11] designed an automatic water flow control system using the ESP32, achieving 92% accuracy and reducing water waste. Rao et al. [12] developed a water quality monitoring system using the ESP32 and Blynk, meeting WHO standards and promoting consumption reduction. Pereira et al. [13] implemented an innovative drip irrigation system using ESP32, demonstrating its effectiveness in crops such as spring onions. Lin et al. [14] used ESP32 to monitor water quality in aquaculture, highlighting the system's accuracy.

Additionally, Jhun et al. [15] developed a low-cost ESP32-based weather station with satisfactory results in environmental measurement. Chang et al. [16] proposed a system to detect pipe leaks using ESP32, highlighting its security through data encryption. Sneineh & Shabaneh [17] designed a hydroponic system using ESP32 and Blynk, achieving efficiency in water management. Toridi et al. [18] created a wireless water level detection system using ESP32, validating its accuracy with an R^2 de 0,99.

In agricultural monitoring, Korlepara et al. [19] developed a system based on humidity sensors to monitor irrigation valves, reducing environmental impact. Satriyo et al. [20] proposed an IoT-powered sprinkler irrigation system, validating its performance in modern agriculture. Faysal & Mohammed [21] created an intelligent irrigation system using ESP32, highlighting its efficiency in crops and horticulture. Mehta et al. [22] optimized agricultural production using a monitoring system with ESP32-WROOM. Abd et al. [23] designed an irrigation monitoring system for chili plantations using ESP8266, achieving optimization in water consumption. Finally, Idris et al. [24] developed a fertigation system using ESP32, reducing operating costs in eggplant cultivation.

While these studies have provided significant advances, none have integrated IoT, machine learning, and renewable energy into a single system for smart irrigation. This study proposes an autonomous irrigation system based on IoT, AI, and solar energy to address this gap. The system architecture consists of two ESP32 microcontrollers and an ESP32-CAM, which manage soil moisture, temperature, ambient humidity, and light intensity sensors. Additionally, a modified Yolov3-tiny model identifies early signs of chlorosis and dehydration, while a facial recognition system restricts access to unauthorized users. The collected data is transmitted to IoT platforms such as Adafruit IO and Telegram, ensuring remote monitoring and real-time notifications.

Unlike conventional systems, this approach integrates artificial intelligence, cloud storage, and solar energy, providing a sustainable and scalable solution for modern irrigation. This paper is structured as follows: Section II presents the theoretical approach; Section III details the methodology used; Section IV presents the experimental results; Section V analyzes the findings and compares them with previous work; and Section VI presents the conclusions, highlighting the impact of the research and proposals for future work.

2- Theoretical Approach

Efficient water management in agriculture is essential to ensure sustainability and productivity in the agricultural sector. The application of emerging technologies, such as the Internet of Things (IoT) and Artificial Intelligence (AI), has revolutionized traditional irrigation practices, enabling significant optimization in the use of water resources.

2-1- Internet of Things (IoT) in Irrigation Systems

The IoT facilitates the interconnection of devices and sensors that collect and transmit data in real-time, enabling more precise and automated irrigation management. Sensors for soil moisture, temperature, and other environmental parameters provide key information for determining the specific water needs of crops, optimally adjusting irrigation, and reducing water waste. Recent studies indicate that the implementation of IoT-based irrigation systems can reduce water consumption in agriculture by up to 50%, contributing significantly to the sector's environmental and economic sustainability [25].

2-2- Artificial Intelligence and Machine Learning in Precision Agriculture

AI and machine learning allow the analysis of large volumes of agricultural data to identify patterns and predict future conditions, facilitating informed decision-making. In the context of irrigation, these systems can predict crop water needs based on variables such as weather conditions, soil type, and plant phenological stage. Machine learning models have been used to predict crop health based on information on pesticide use and other crop variables, which has led to improved efficiency and reduced environmental impact [26].

2-3- Integration of Renewable Energy in Smart Irrigation Systems

Incorporating renewable energy sources, such as solar energy, into innovative irrigation systems promotes sustainability and reduces dependence on fossil fuels. Using solar panels to power IoT devices and water pumping systems lowers operating costs and minimizes agricultural operations' carbon footprint [27].

2-4- Practical Applications and Benefits

Combining IoT, AI, and renewable energy in irrigation systems has improved water use efficiency and crop health. Implementing IoT sensors in drip irrigation systems allows water flow to be adjusted to individual plants or specific rows based on detected needs, optimizing the use of water resources [28]. Furthermore, using AI-based predictive models facilitates the early detection of pests and diseases, enabling timely interventions and reducing the need for pesticides [29].

In summary, this research's theoretical approach focuses on integrating advanced and sustainable technologies to develop a smart irrigation system that optimizes water use, improves agricultural productivity, and contributes to environmental sustainability.

3- Research Methodology

For the development of this project, the waterfall methodology was used, which consists of five sequential phases that must be completed in an orderly manner [30]. This approach guarantees a clear and well-defined structure during all stages of the project. In addition, the TDD (Test-Driven Development) methodology was implemented to write tests before code development and ensure that the implemented functionalities meet the specified requirements [31]. Figure 1 presents the phases of the waterfall methodology, while Figure 2 describes the stages of the TDD approach applied to the project.

Figure 1. Phases of the Cascade Methodology

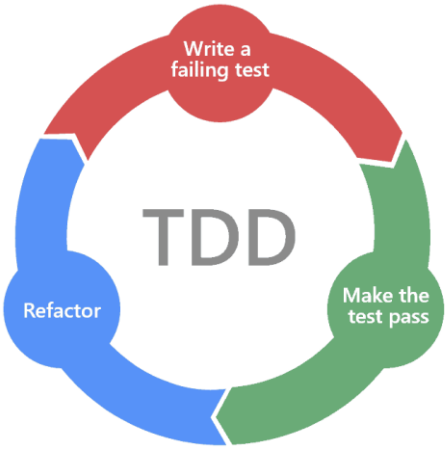

Figure 2. Phases of the TDD Methodology

3-1-Requirements and Design

The development of the project requires identifying and defining multiple key aspects. The first step was to determine the System's implementation area using official statistics from the Peruvian government [32]. This analysis allowed us to contextualize the problem and prioritize the specific needs related to irrigation and maintenance of green areas. To better understand the challenges associated with manual irrigation systems, an Ishikawa diagram was used, as shown in Figure 3, which identified the leading causes of inefficiencies, such as water waste and high maintenance costs. In addition, Figure 4 and Table 1 detail the analysis of square meters of green areas per inhabitant in the districts of Lima. This analysis highlights that the World Health Organization (WHO) recommends at least 9 square meters of green areas per inhabitant, a figure that has yet to be reached in most districts evaluated. Table 2 also provides a comparison between the benefits of the proposed smart irrigation system and those of a manual system, addressing aspects such as irrigation efficiency, resource use, long-term operating costs, and environmental impact. This comparison highlights the significant advantages that automation and the use of IoT technologies bring to the management of water and energy resources.

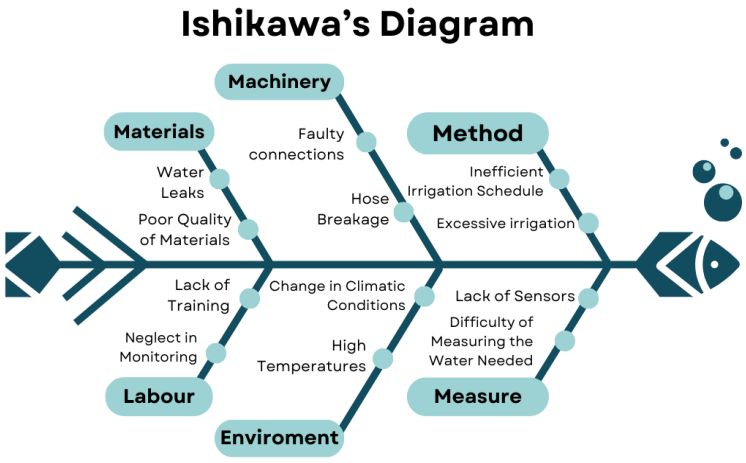

Figure 3. Ishikawa diagram

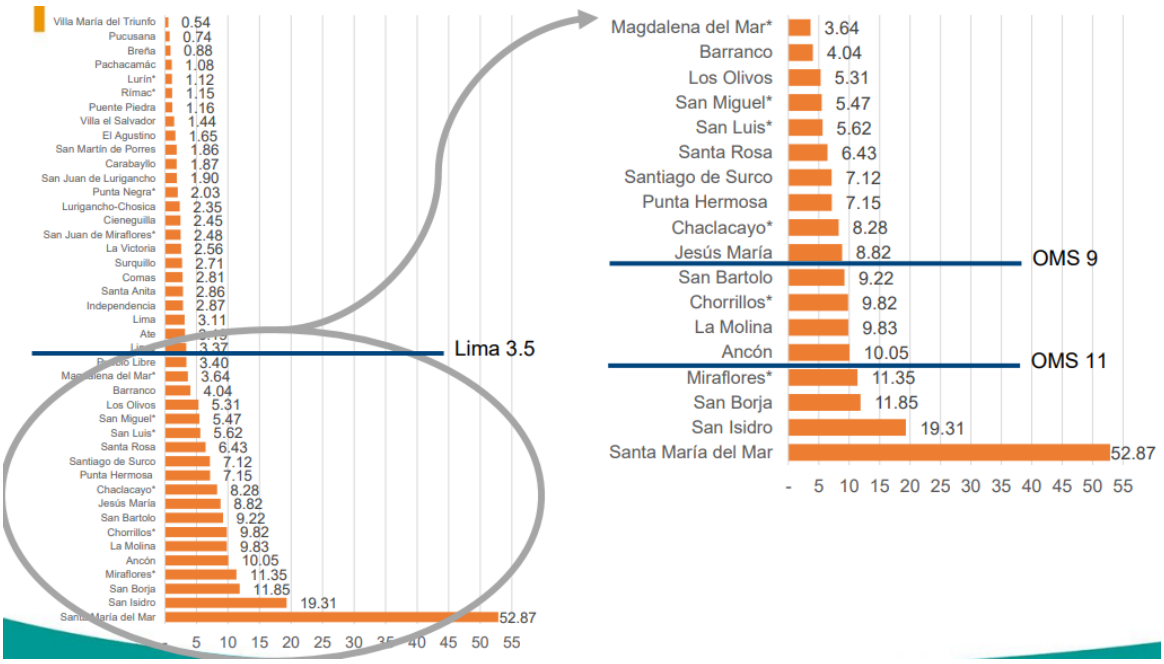

Figure 4. Square meters of green areas per inhabitant in Perú

Table 1. Square meters of green areas in the districts of Lima, Perú

District	Total Green Areas per inhabitant(m²)
San Miguel	5.47
Los Olivos	5.31
Ate	3.15
Lima	3.11
Comas	2.81

Note: Only a few districts were referred.

Table 2. The comparison of the benefits of a Smart Irrigation System versus a Traditional one (Manual)

Aspect	Smart Irrigation System	Traditional Irrigation System
Irrigation Efficiency	Adjust the amount of water according to the soil moisture, reducing water waste.	Water at fixed intervals which can lead to over- or under-watering.
Water Saving	Use sensors and automatic control to optimize water consumption	Generally requires more water due to constant watering
Remote Control	Allows you to monitor and control irrigation from Telegram	It does not have a remote control
Use of Renewable Energy	The application of renewable energy based on solar panels was given	It requires the use of labor for its operation
Long Term Costs	Long-term cost reduction thanks to more efficient use of water and energy.	Higher operating costs due to increased water and labor consumption.
Environmental Conservation	Minimizes resource use by precisely adjusting irrigation, reducing environmental impact.	Contributes to the depletion of water resources and increased carbon emissions

3-2-System Architecture

Figure 5 illustrates the overall architecture of the proposed System, which integrates various sensors and microcontrollers for environmental monitoring and control of the irrigation system. Among the sensors used are the LDR, YL-83, DHT22, and the Capacitive Soil Moisture Sensor, which collects data on environmental conditions. These data are sent to the control units, composed of two ESP32 and one ESP32-CAM, programmed in the Arduino IDE and Python environments.

The first ESP32 manages the sensors, while the second controls the irrigation system's actuators, including a relay and a solenoid valve. The ESP32-CAM, on the other hand, performs visual monitoring tasks using machine learning, such as face recognition and detection of critical events in crops. The entire System is powered by rechargeable lithium batteries using solar panels and uses voltage regulators to ensure a continuous power supply. In addition, notifications,

real-time data, and remote control of the System are managed through applications such as Telegram, Dropbox, YouTube, and Adafruit IO.

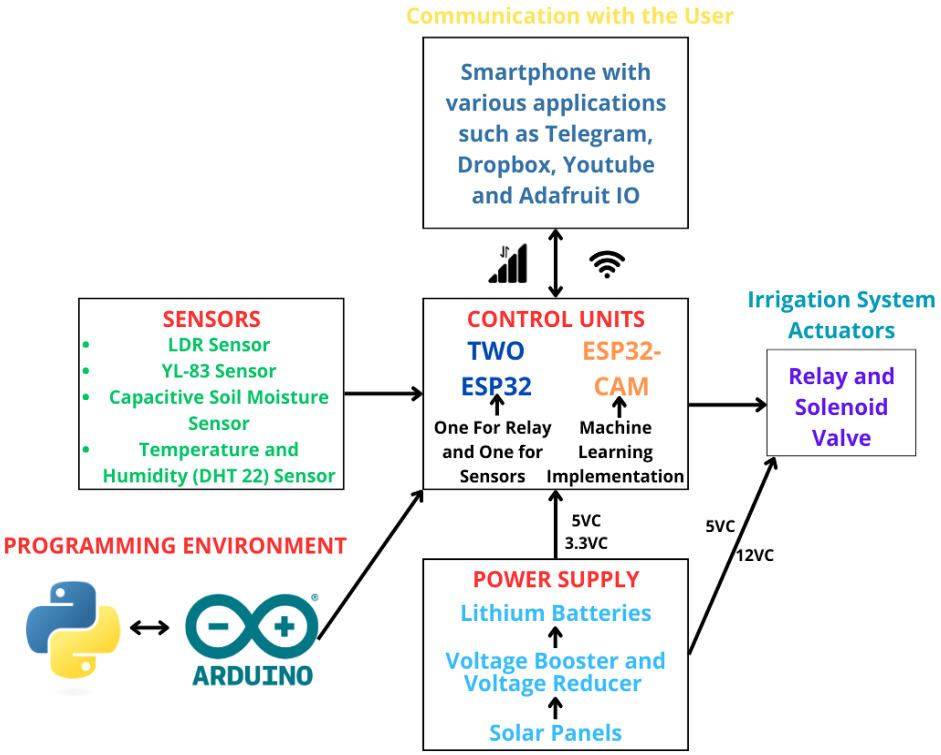

Figure 5. Smart Irrigation System Architecture

3-3-System Components

The fundamental components of the smart irrigation system are two ESP32 [33] microcontrollers and one ESP32-CAM, configured to ensure data processing, automation, and efficient monitoring of the System. Figure 6 and Figure 7 show the physical layout and schematics of the ESP32 and ESP32-CAM, respectively, illustrating the configurations required to efficiently integrate sensors, actuators, and communication modules. The synergy between the ESP32 and the ESP32-CAM [34] is crucial, combining local processing and control with advanced visual monitoring, making the System a comprehensive solution for plant and green area care. Furthermore, this modular and scalable design allows future expansions to incorporate more sensors, actuators, or other modules—functionalities adapting to the user's needs.

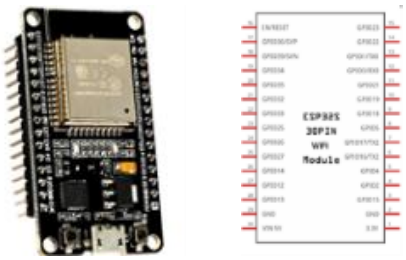

Figure 6. ESP32 Microcontroller Design: physical view (left), schematic view (right)

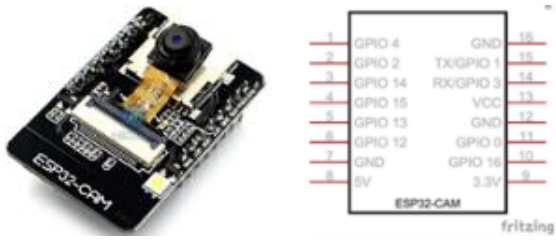

Figure 7. ESP32-CAM Microcontroller Design: physical view (left), schematic view (right)

3-4-System Block Diagram

Figure 8 presents the block diagram of the System, highlighting the integration of renewable energy, sensors, and actuators. The solar energy captured by the panels is stored in lithium batteries and regulated by voltage bucks and boosters to meet the specific requirements of the electronic components. The data flow starts with the ESP32 in charge of the sensors, which collects information on temperature, humidity, light, rain, and soil moisture. This data is sent to the second ESP32, which activates the Relay to control the solenoid valve and operate the irrigation system. In parallel, the ESP32-CAM performs visual monitoring tasks and sends notifications via Telegram and Dropbox in case of detection of critical events.

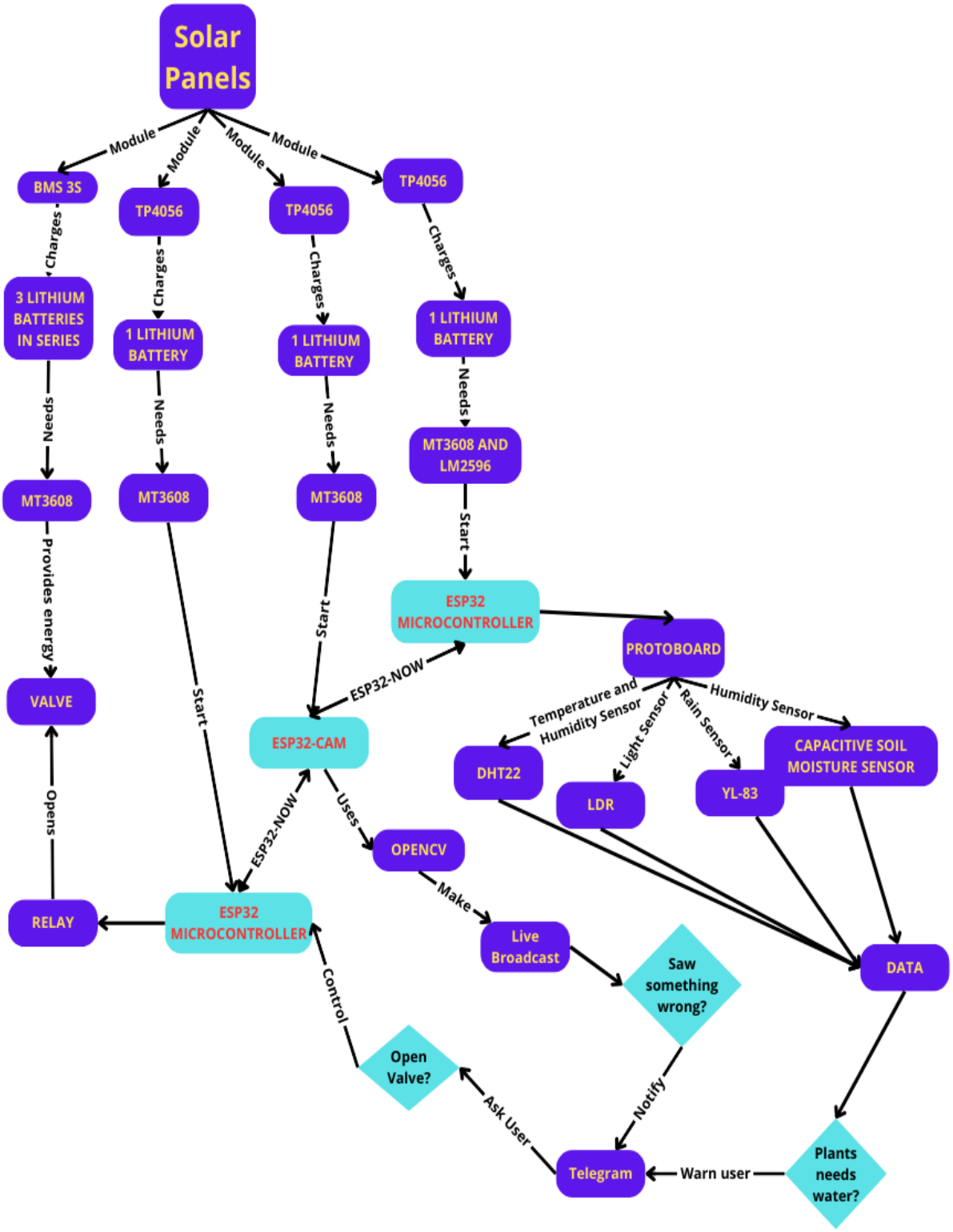

Figure 8. System Block Diagram

3-5- Communication Model

Figure 9 describes the communication model between the devices that make up the innovative irrigation system, highlighting the interaction between sensors, microcontrollers, and cloud applications. The sensors connected to the first ESP32 collect environmental data such as temperature, humidity, rainfall, and light intensity, which are processed locally and transmitted to the second ESP32 using the ESP-NOW protocol. This second ESP32 is responsible for activating the solenoid valve through a relay based on the parameters received or the commands sent by the user through platforms such as Telegram or Adafruit IO. On the other hand, the ESP32-CAM plays a critical role in visual monitoring, streaming live video to YouTube Live, and uploading images to Dropbox to detect unknown people or relevant environmental events. In addition, this module sends automatic notifications via Telegram to alert the user about potentially dangerous situations, such as the presence of intruders or problems with the plants. The integration of these technologies enables an efficient and real-time communication flow, ensuring dynamic control and continuous monitoring of the System, regardless of the user's location.

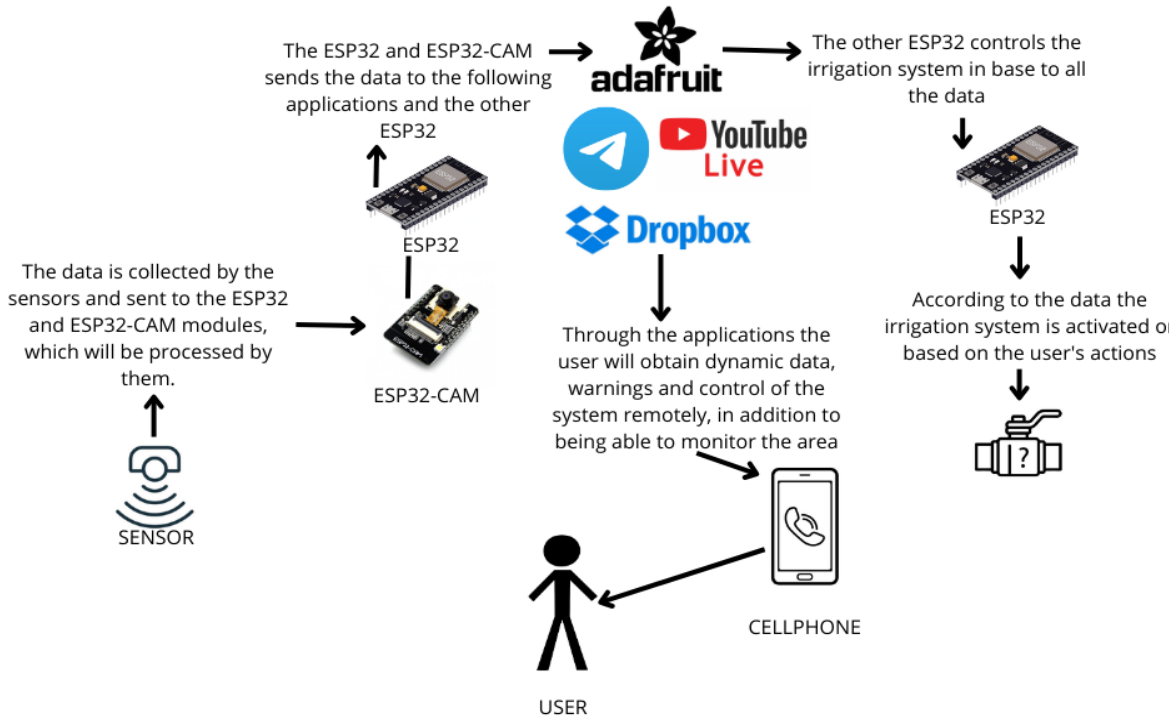

Figure 9. Device Communication Diagram

3-6- System Flow

Similarly, Figure 10 shows the flow chart of the proposed System, which goes through the following steps sequentially:

- **Obtaining Solar Energy:** The solar panels charge the lithium batteries, which will power the two ESP32s and the ESP32-CAM through voltage boosters and reducers.
- **Environmental Monitoring:** Sensors measure variables such as temperature, humidity, light, rain, and soil moisture to assess environmental conditions.
- **Monitoring the Area:** The ESP32-CAM can detect problems in crops or plants, identify faces, and alert about the presence of strangers.
- **Decision Making:** Sensor data and machine learning models determine the necessary actions, such as activating the solenoid valve or sending notifications, such as taking photos or alerting via Telegram.
- **Automated Actions:** The System features a computerized function for watering plants. This function, using the data collected, can automatically determine whether watering is necessary. This function is present if the user does not interact with the System during the notices of watering requests.
- **User-System Interaction:** The user receives alert notifications, either by Telegram, Dropbox, or Adafruit IO, and can also interact with the System through Telegram and Adafruit IO commands.
- **Continuous Cycle:** The System repeats this process to ensure constant monitoring.

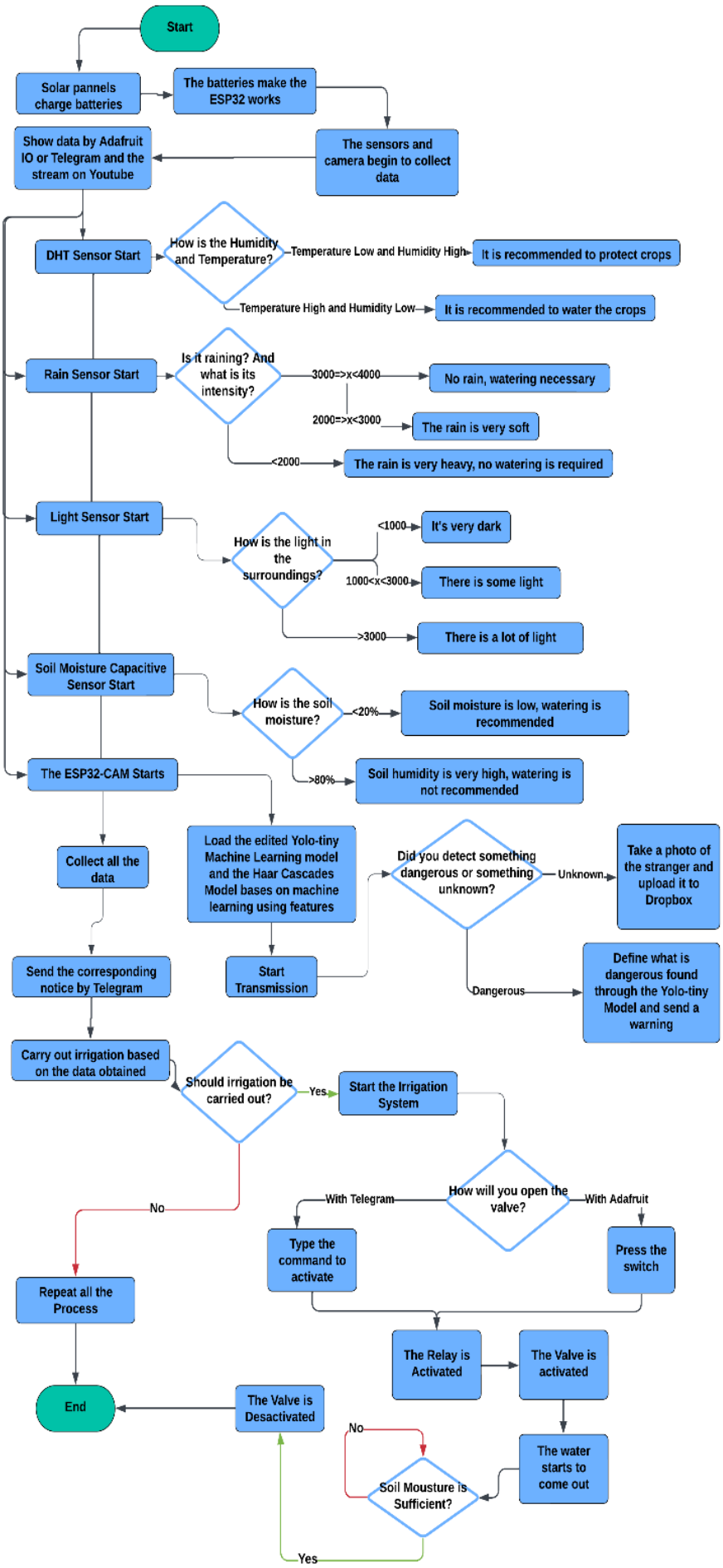

Figure 10. System Flowchart

3-7-Housing Design

To protect the electronic components from dust and moisture and facilitate their transport, a compact enclosure measuring 47 cm long and 27 cm wide was designed. This structure provides sufficient space to house the ESP32 and ESP32-CAM microcontrollers, environmental sensors, the relay module, and the solar energy management system. It also features strategic openings that allow the sensors to interact appropriately with the environment, ensuring efficient and stable system operation. Figure 11 shows the design and internal layout of the enclosure, highlighting the arrangement of key components for optimal performance.

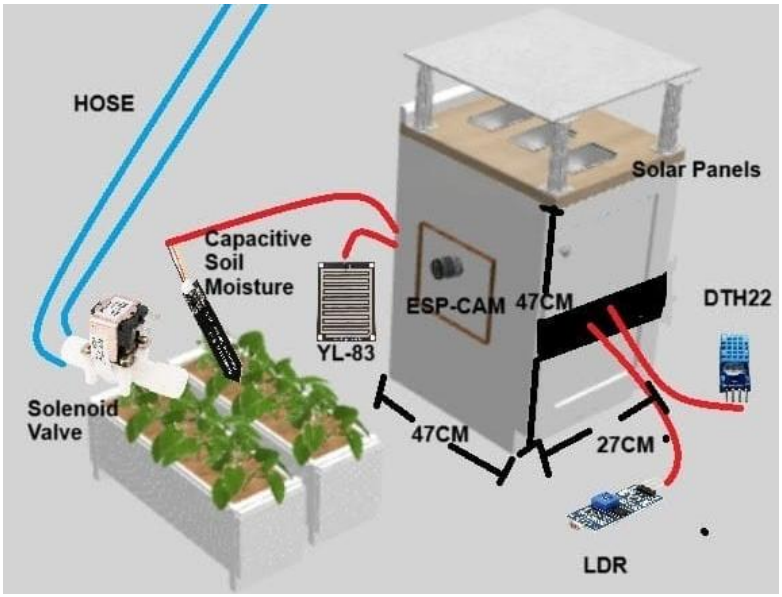

Figure 11. Sample of the housing where the components will be placed

Figure 12 details the modular design of the internal wiring, where each ESP32 is connected to a different functional block. One ESP32 manages the sensors and collects environmental data, transmitting it to another ESP32 via ESP-NOW, which controls the solenoid valve. Additionally, an ESP32-CAM performs visual analysis with artificial intelligence (AI), detecting plant problems and unfamiliar faces. The collected data is integrated with cloud platforms such as Telegram, Adafruit IO, Dropbox, and YouTube Live, enabling real-time monitoring. For power, the solar power system incorporates MT3608 (step-up) and LM2596 (step-down) voltage regulators to stabilize the power supply to the components. Finally, the system design and validation were carried out in the Fritzing simulation environment, ensuring its correct operation before physical implementation.

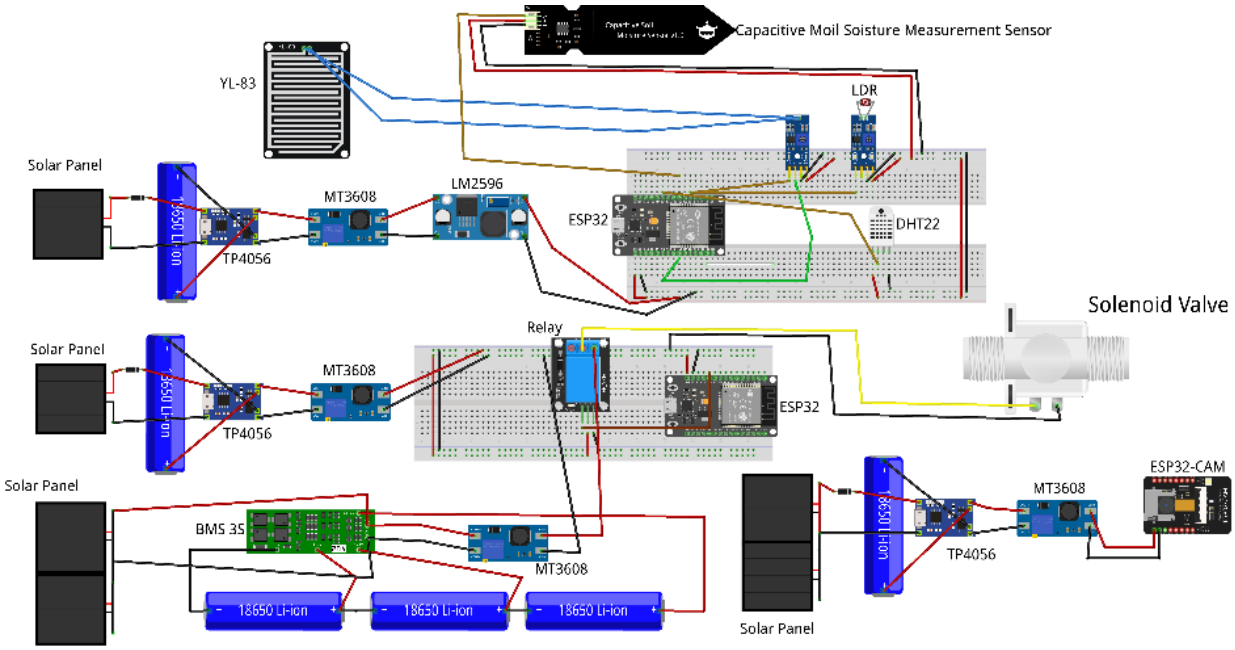

Figure 12. Design of the corresponding connections for the system prototype

Figure 13 illustrates the complete schematic of the system, which integrates sensors, automation, artificial intelligence (AI)-based monitoring, and solar energy. Its architecture consists of three main modules: an ESP32 that collects environmental data such as temperature, soil moisture, light, and rainfall, transmitting this information to the irrigation module via ESP-NOW for an immediate response; a second ESP32 that manages the opening or closing of the solenoid valve via a relay and enables remote irrigation management through Telegram commands; and an ESP32-CAM, which implements a YOLOv3-Tiny model for early detection of plant problems such as chlorosis and dehydration, in addition to performing facial recognition to differentiate between authorized and unknown users. Additionally, the system integrates with cloud platforms to improve its functionality: Dropbox, which stores images captured by the ESP32-CAM when an unknown person is detected; Telegram, which sends notifications and enables remote control of the irrigation system; Adafruit IO, which monitors and visualizes sensor data in real-time; and YouTube Live, which facilitates live streaming of the monitored environment for efficient remote monitoring.

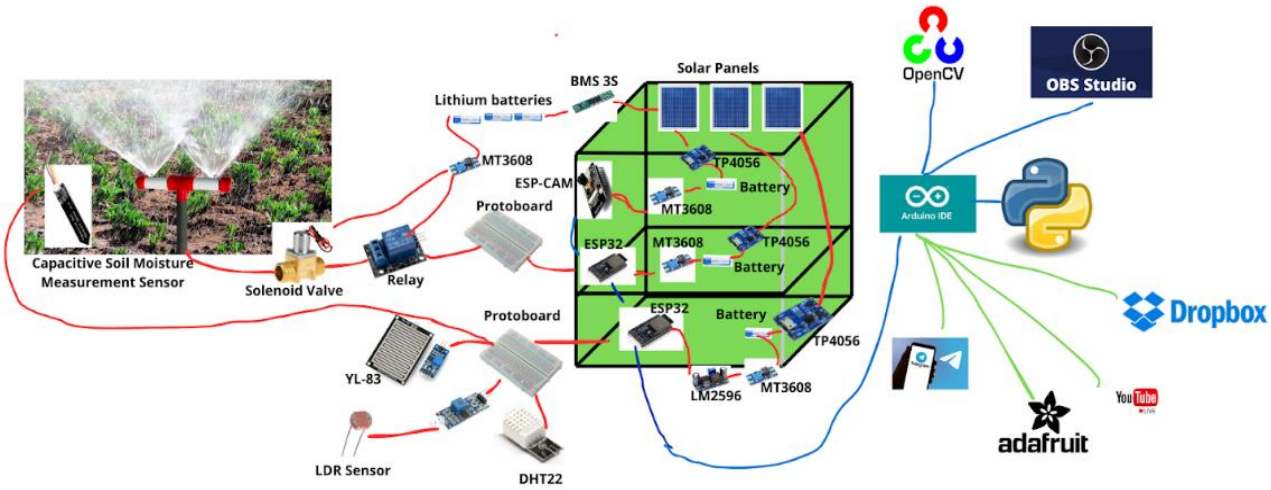

Figure 13. The final draft of the entire System

3-8- Implementation with TDD Integration

3-8-1- Simulation and Initial Validation

During the system implementation phase, the Test-Driven Development (TDD) methodology was applied, which allowed testing and validation of the codes necessary to ensure the correct operation of the components before their physical implementation. To do so, two simulation environments were used: Wokwi and Proteus. Figure 14 shows the Wokwi environment, where the DHT22 sensor code was tested, while Figure 15 shows the Proteus environment, used to validate the operation of the YL-83 sensor.

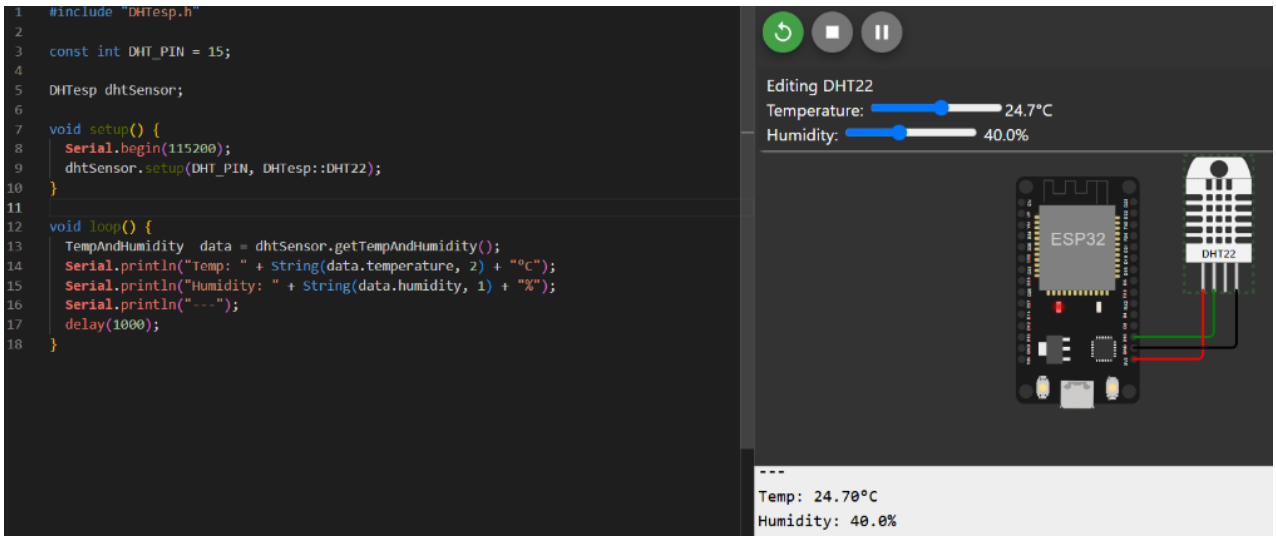

Figure 14. Wokwi environment and testing for DHT22

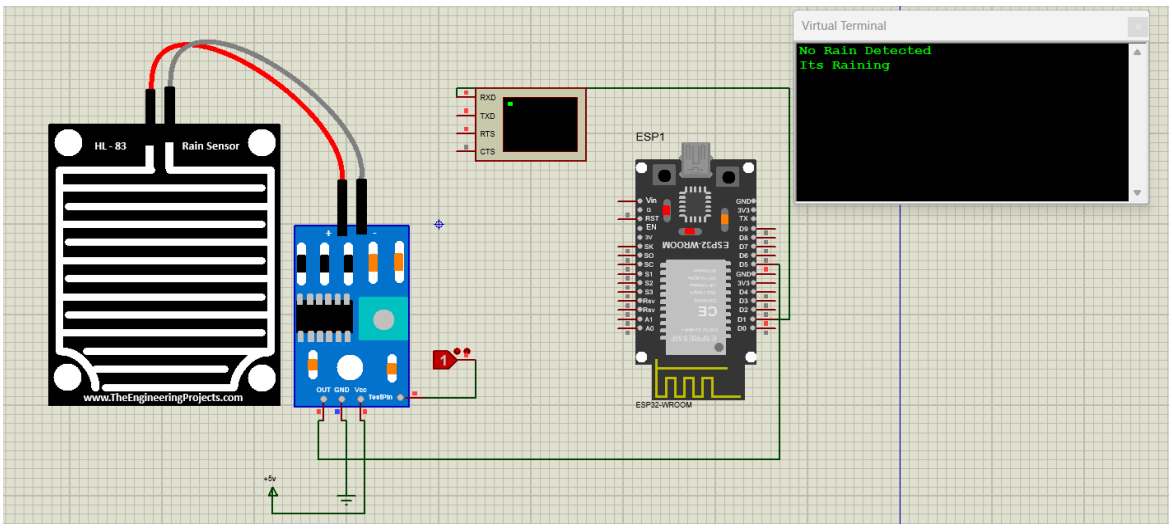

Figure 15. Proteus environment and testing for the YL-83

3-8-2- Implementation in Real Electronic Components

Once the code had demonstrated its functionality on the simulators, it was transferred to the real electronic components. The chosen programming environment was Arduino IDE, where the libraries necessary for the operation of the sensors and actuators were integrated. Following TDD's principles, code improvements were iteratively implemented, including advanced functions such as alert systems and connections to communication platforms such as Telegram, Dropbox, and Adafruit IO. Figure 16 shows a fragment of the code used to define the alert and warning ranges.

```
void checkRain() {
    int rainValue = digitalRead(rainDigitalPin);
    if (rainValue == LOW && !isRaining) {
        isRaining = true;
        String message = "⚠️ Rain Alert: It is raining.";
        bot.sendMessage(chatID, message, "");
    } else if (rainValue == HIGH && isRaining) {
        isRaining = false;
    }
}

void checkTemperature() {
    float temp = dht.readTemperature();
    if (!isnan(temp)) {
        if (temp > highTempThreshold && !tempAlertSent) {
            String message = "🔥 Alert: The temperature is very high (" + String(temp) + "°C).";
            bot.sendMessage(chatID, message, "");
            tempAlertSent = true;
        } else if (temp <= highTempThreshold && tempAlertSent) {
            tempAlertSent = false;
        }
    }
}

void checkSoilHumidity() {
    int capacitiveValue = analogRead(soilMoisturePin);
    float humidityPercentage = map(capacitiveValue, 2650, 900, 0, 100);
    if (humidityPercentage < lowSoilMoistureThreshold && !soilMoistureAlertSent) {
        String message = "⚠️ Alert: Soil moisture is very low (" + String(humidityPercentage) + "%).";
        bot.sendMessage(chatID, message, "");
        soilMoistureAlertSent = true;
    } else if (humidityPercentage >= lowSoilMoistureThreshold && soilMoistureAlertSent) {
        soilMoistureAlertSent = false;
    }
}
```

Figure 16. Code in Arduino IDE for notification systems

3-8-3- Calculation of Soil Moisture Percentage

An essential part of the programming was the calculation of the soil moisture percentage based on the values measured by the capacitive moisture sensor. The range was calibrated from 900 (0% moisture) to 2650 (100% moisture), and this formula, derived from experimental tests, made it possible to obtain precise data on the state of the soil, receiving the formula:

$$\text{Soil Moisture \%} = ((\text{Sensor Value}) - 900)/ (2650-900) \times 100\% \quad (1)$$

3-8-4- Programming the Second ESP32

In the second ESP32, the activation of the Relay responsible for controlling the solenoid valve for the water flow was programmed. Communication between both ESP32s was carried out using the ESP-NOW protocol, which requires configuring the MAC addresses of each device. Figure 17 shows a fragment of the code of this ESP32, focused on automating irrigation according to the parameters received.

```
#include <WiFi.h>
#include <esp_now.h>
const int relayPin = 14;
bool relayState = false;
// Structure for the ESP-NOW message
typedef struct __attribute__((packed)) {
    char command[32];
    int soilMoisture;
} esp_now_message;
// MAC address of the issuer
uint8_t senderMacAddress[] = {0x3C, 0x71, 0xBF, 0x19, 0xA6, 0xC4};
const char* ssid = "2.4G_Luna Benito";
const char* password = "pepe01..";
void setup() {
    Serial.begin(9600);
    pinMode(relayPin, OUTPUT);
    digitalWrite(relayPin, LOW);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi...");
    }
    Serial.println("Connected to WiFi");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
    esp_now_init();
    esp_now_add_peer(senderMacAddress, ESP_NOW_ROLE_COMBO, 1, NULL, 0);
    esp_now_register_recv_cb(onReceive);
}

void loop() {
    delay(1000);
}
// Message reception callback
void onReceive(const uint8_t* mac_addr, const uint8_t* incomingData, int len) {
    esp_now_message message;
    memcpy(&message, incomingData, sizeof(message));
    if (strcmp(message.command, "activate") == 0) {
        int humidityPercentage = message.soilMoisture;
        if (humidityPercentage < 80) {
            activateRelay();
        } else {
            Serial.println("⚠️ Soil humidity is high. Irrigation is not activated.");
        }
    } else if (strcmp(message.command, "deactivate") == 0) {
        deactivateRelay();
    }
}
// Features
void activateRelay() {
    if (!relayState) {
```

Figure 17. Sample ESP32 code for the Relay

3-8-5- Advanced ESP32-CAM Programming

The ESP32-CAM was programmed in two complementary environments to implement its advanced functionalities. In the Arduino IDE environment, video capture from the camera was configured, while in Python, artificial intelligence libraries for object detection and facial recognition were integrated. These capabilities are based on a modified Yolov3-tiny model, which was trained with project-specific images to adapt to the requirements. The customization process included modifying configuration files, training with new images, and defining specific measures for detecting healthy, dehydrated, or chlorotic plants. The Haar-Cascades approach was also used for facial recognition, with a database containing photographs of authorized users with various expressions. These configurations allow the ESP32-CAM to distinguish authorized users from unknown ones. Additional functions include uploading images to Dropbox when detecting unrecognized objects, sending automatic alerts via Telegram, and live video streaming via YouTube Live.

3-8-6- Codes and Database

Figures 18 and 19 show code snippets implemented in the Arduino IDE and Python, respectively. Figure 20 illustrates the database used to train the YOLOv3-Tiny model, which contains images of faces and healthy plants and plants affected

by dehydration or chlorosis. One thousand, two hundred images per class were collected and stored for training, representing 3,600 images. The open-source tool LabelImg was then used to label the pictures in YOLO format (.txt). Once the corresponding configuration files (obj.names and obj.data) were generated, the model was trained on the Darknet, an environment specialized in convolutional neural networks. Training was performed on a PC with an RTX 3060 GPU (12GB VRAM), running 4,000 epochs over 39 hours. Following this process, the model's accuracy was evaluated using the Mean Average Precision (mAP) metric, yielding 93.86%. This result guarantees highly effective detection of plant problems using computer vision. The accuracy of the YOLOv3-Tiny model was calculated using the following equation:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} APi \tag{2}$$

where, N represents the number of classes evaluated and AP is the area under the Precision-Recall curve for each class (Healthy Plant, Dehydrated Plant, and Chlorotic Plant).

The Haar Cascade model was used in conjunction with the LBPH (Local Binary Patterns Histogram) algorithm for facial recognition. For training, images of one of the authors with various expressions and head movements were taken. These images were converted to grayscale to improve processing. The Haar Cascade model achieved an accuracy of 95% in well-lit conditions, decreasing to 88% in low-light environments. Accuracy was calculated using the following standard equation:

$$Accuracy\ (\%): \frac{TP}{TP+FN} \times 100 \tag{3}$$

where TP represents true positives and FN false negatives.

Thanks to all of that, this machine learning-based approach optimizes the detection of plant problems and ensures the safety of the monitored environment.

```
  Serial.println(WiFi.localIP());
  server.begin();
}
void loop() {
  WiFiClient client = server.available();
  if (client) {
    while (!client.available()) {
      delay(1);
    }
    String request = client.readStringUntil('\r');
    client.flush();
    if (request.indexOf("/stream") != -1) {
      sendMjpegStream(client);
    } else {
      client.println("HTTP/1.1 404 Not Found");
      client.println("Content-Type: text/plain");
      client.println("Connection: close");
      client.println();
      client.println("Page not found");
    }
  }
  if (millis() - lastPhotoTime >= photoInterval) {
    takeAndUploadPhoto();
    lastPhotoTime = millis();
  }
  if (millis() - lastDeleteTime >= deleteInterval) {
    deleteOldPhotos();
    lastDeleteTime = millis();
  }
}
void sendMjpegStream(WiFiClient client) {
  camera_fb_t * fb = NULL;
  String boundary = "frame";
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: multipart/x-mixed-replace; boundary=" + boundary);
  client.println("Connection: close");
  client.println();
  while (true) {
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Error capturing the frame");
      return;
    }
    client.println("--" + boundary);
    client.println("Content-Type: image/jpeg");
    client.println("Content-Length: " + String(fb->len));
    client.println();
    client.write(fb->buf, fb->len);
    client.println();
```

Figure 18. ESP32-CAM code in Arduino IDE

```
import cv2
import os
import numpy as np

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
net = cv2.dnn.readNet("yolov3-tiny.weights", "yolov3-tiny.cfg")
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))

output_folder = 'capturas'
image_files = os.listdir(output_folder)

known_face_images = []
known_face_names = []

for image_file in image_files:
    if image_file.endswith('.jpg'):
        img_path = os.path.join(output_folder, image_file)
        image = cv2.imread(img_path)
        known_face_images.append(image)
        name = image_file.split('_')[0]
        known_face_names.append(name)
def compare_faces(known_images, detected_face):
    for i, known_image in enumerate(known_images):
        # Convertir ambas imágenes a escala de grises
        gray_known = cv2.cvtColor(known_image, cv2.COLOR_BGR2GRAY)
        gray_detected = cv2.cvtColor(detected_face, cv2.COLOR_BGR2GRAY)
        known_faces = face_cascade.detectMultiScale(gray_known, 1.3, 5)
```

Figure 19. ESP32-CAM code in Python with the implementation of AI Libraries and Machine Learning

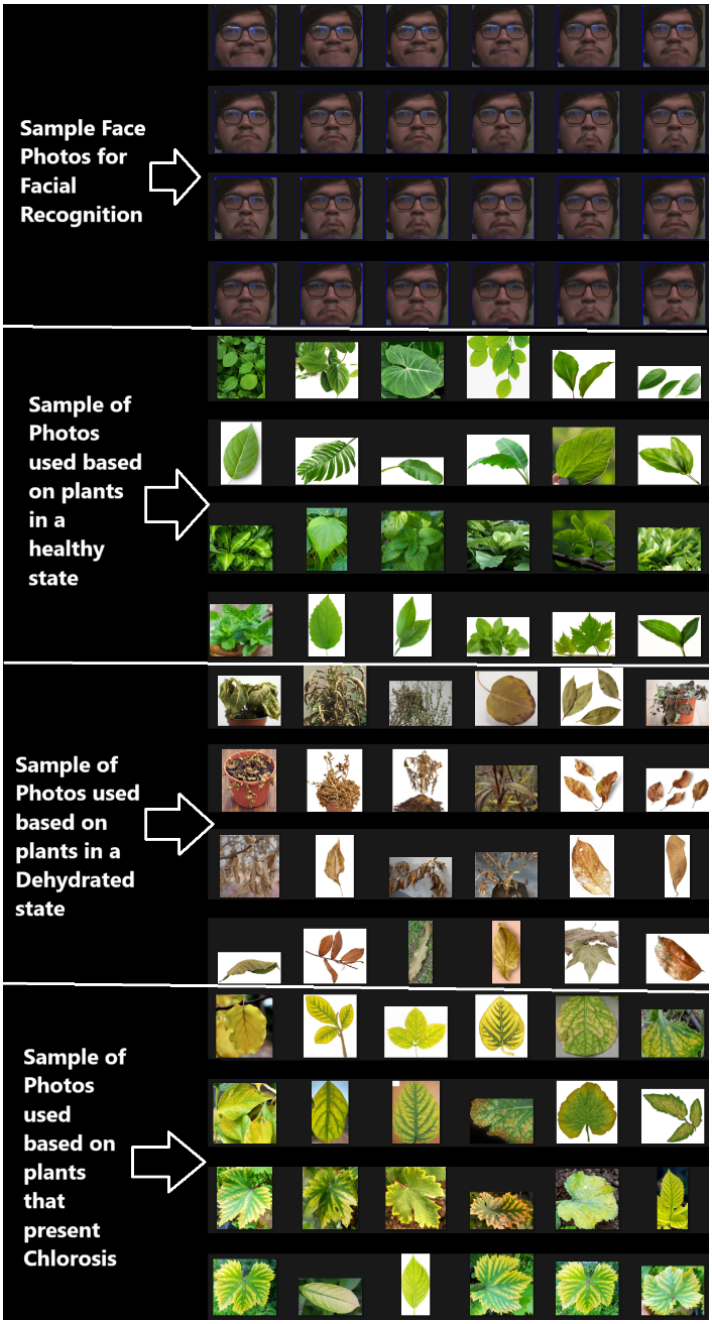

Figure 20. Sample of the Photos that were used for machine learning

4- Results

4-1- Verification and Maintenance

Verification and maintenance tests were performed using the cascade methodology to ensure proper system operation. Figure 21 shows the component connections, while Figure 22 presents the wiring of the solar charging system. The solar power system uses TP4056 and 3BMS as charge controllers for lithium batteries, an LM2596 as a step-down voltage regulator that adjusts the output to 3.3V for the sensors and ESP32, and an MT3608 as a voltage amplifier, optimizing power delivery.

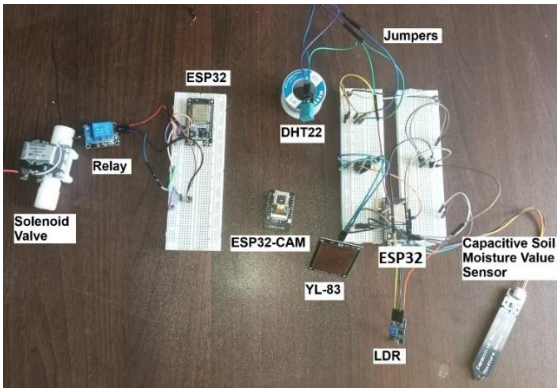

Figure 21. Control Stage

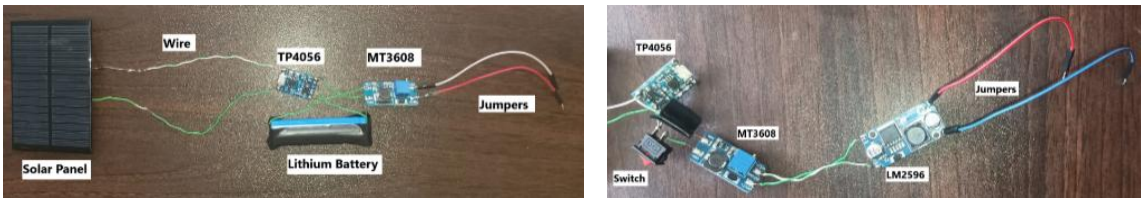

Figure 22. Power Stages: Solar Panel System view (left), System between MT3608 to LM2596 view (right)

Figure 23 shows the validation of the solar power system using a multimeter, obtaining an output of 10.49V, as seen in the image on the left side. On the right side, the correct operation of the LM2596 regulator is verified, providing a stable 3.3V output to power the sensors and one of the ESP32s, ensuring efficient system operation.

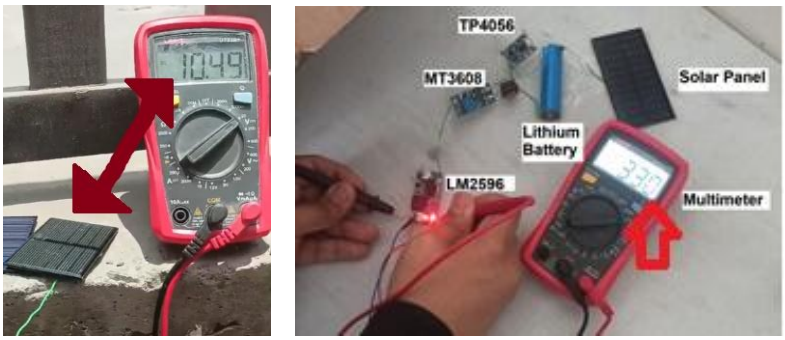

Figure 23. Operation of: Solar Panels view (left), Power System Sensors view (right)

4-2- System Energy Sustainability

The innovative irrigation system operates on solar power, ensuring its long-term sustainability. A rechargeable lithium battery system was implemented for each ESP32 and ESP32-CAM to maintain its operation on cloudy or low-light days. During testing, it was verified that the system could operate between 12 and 14 hours without receiving power from the solar panels, thanks to energy storage. Furthermore, the use of charge controllers prevents battery overcharging, prolonging their lifespan. Unlike conventional irrigation systems, this solution eliminates dependence on the electrical grid, reduces operating costs, and ensures continuous operation even in adverse weather conditions.

4-3- Prototype Assembly and Testing

To protect the components and facilitate assembly, modular housing was designed with internal partitions for sensors and electronics. The system was tested in green park areas and indoor plants (pots), as shown in Figure 24. In both cases,

a notable improvement in turf quality and plant health was observed, reducing signs of dehydration. To evaluate the system's impact on irrigation optimization, water consumption was compared between a manual system based on fixed schedules and the proposed system, which adapts irrigation based on sensors. Manual measurements were taken using a water tank for 20 days in green areas (2 m²) and 10 days in a pot with an indoor plant, as shown in Table 3.

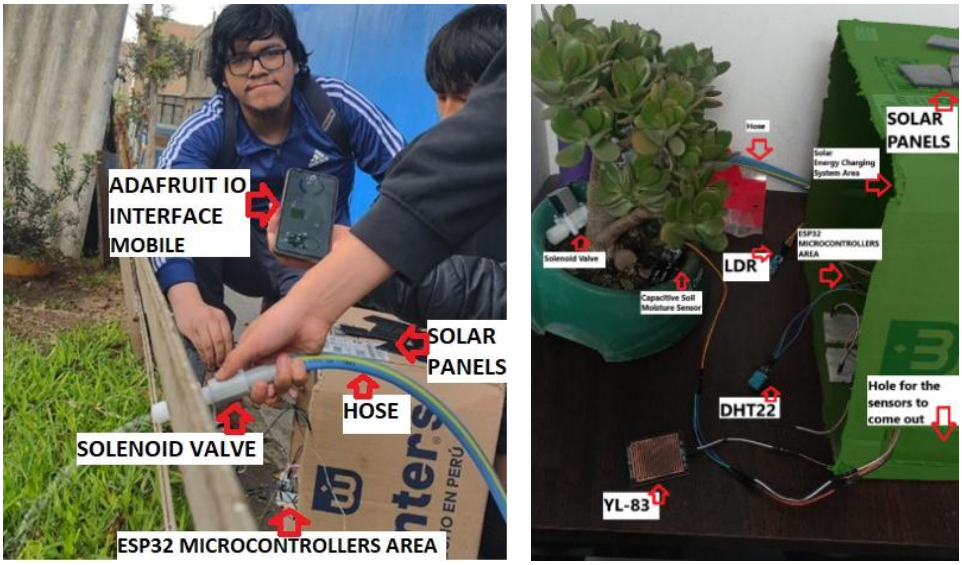

Figure 24. The prototype shows improvements in the green areas' view (left) and from a pot with a plant view (right)

Table 3. Water Consumption Comparison

Testing Site	Liters of Water Used by Manual Irrigation	Liters of Water Used by the Proposed Smart Irrigation System	Ahorro de Agua (%)
Green Areas of a Park (20 Days of Testing)	380L	214L	43.68%
Potted Plant (10 Days of Testing)	2L	1.13L	43.5%

With the water consumption values of each irrigation system, the percentage of water savings can be calculated using the following formula:

$$Water\ Savings\ (\%) = \left(\frac{Manual\ Irrigation\ Consumption - Smart\ System\ Consumption}{Manual\ Irrigation\ Consumption}\right) \times 100 \quad (4)$$

The proposed system was confirmed to reduce water consumption by 43.68% in lawns and 43.5% in pots, with slight variation due to sun exposure, soil type and water retention capacity, and the frequency of irrigation system use. Unlike traditional systems that operate on fixed schedules, this approach dynamically adapts irrigation, optimizing water consumption without affecting plant health.

4-4-Software and Application Validation

Tests were conducted on Telegram, Dropbox, YouTube Live, and Adafruit IO to validate the system's software and connectivity. The Telegram bot allows users to query sensor data (/rain, /temperature, /humidity, /light, /soil), activate or deactivate irrigation (/active, /deactivate), and access real-time monitoring via direct links to Dropbox and YouTube Live. To enhance security, access control was implemented that restricts bot responses to pre-authorized users only, preventing third parties from accessing sensitive data. Figure 25 shows the available commands, and Figure 26 illustrates the automatic sending of alerts. Unlike other IoT systems, this integration with Telegram allows responses in less than 1.5 seconds, ensuring real-time monitoring.

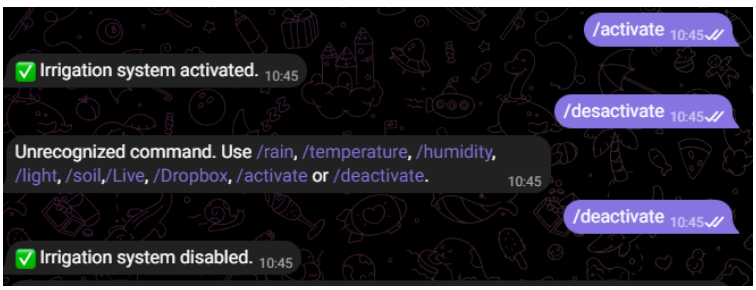

Figure 25. Shows the existing commands and operation of activating and deactivating the irrigation system

Figure 26. Demonstrates how commands and notifications work through the Telegram environment

4-5- ESP32-CAM Testing

The ESP32-CAM module was configured for object detection and facial recognition using the YOLOv3-Tiny model and the Haar cascade algorithm. Tests were conducted with static images and real-time video, achieving an accuracy of 93.86% in plant disease detection based on the Mean Average Precision (mAP) metric. For facial recognition, the accuracy rate reached 95% in good lighting conditions and 88% in low-light environments.

Figure 27 presents the object detection results, with the left image showing the identification of a dog and the correct image showing the facial recognition of an authorized user. Figure 28 illustrates the system's ability to detect plant dehydration (left) and chlorosis (right).

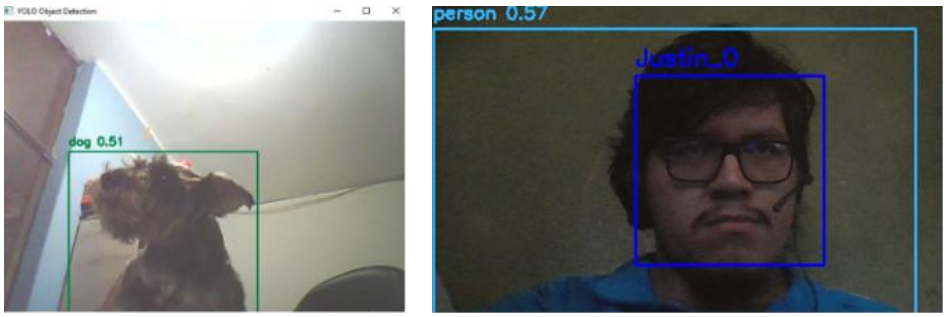

Figure 27. Object detection (left) and facial recognition (right)

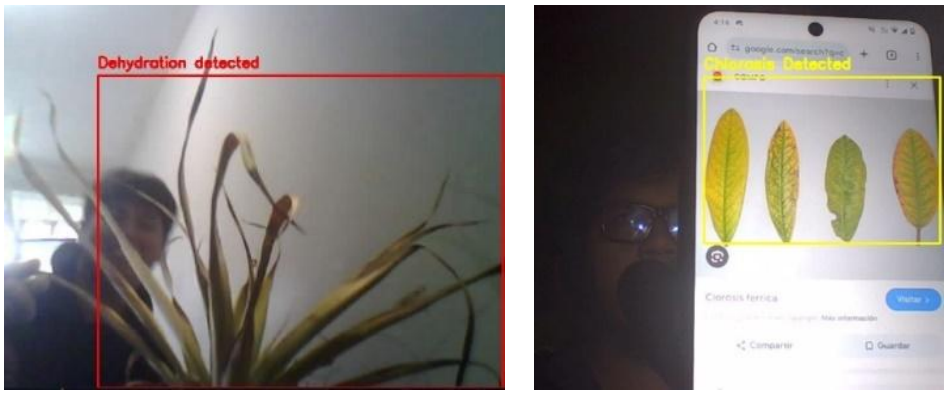

Figure 28. Plant dehydration detection (left) and chlorosis (right)

These results validate the system's effectiveness in identifying plant problems early and facilitating preventive intervention to improve plant health. Furthermore, facial recognition helps restrict unauthorized access to the system, although its accuracy decreases in low-light conditions. Finally, the system operates in real-time without relying on external servers, optimizing its efficiency and autonomy.

4-6- System Security with Unknown Person Detection

4-6-1- Facial Recognition Implementation

To improve the security of the system and the monitored environment, a facial recognition module based on Haar cascades with LBPH (Local Binary Pattern Histogram) was incorporated, allowing for the identification of registered users and the detection of unknown individuals in real-time. The model was trained with multiple images of authorized

users, achieving an accuracy of 95% in good lighting conditions and 88% in low-light environments, with a response time of less than 1.5 seconds. Unlike conventional security systems, this implementation processes the data directly on the ESP32-CAM, ensuring a rapid and autonomous response without relying on external servers.

4-6-2- Detection and Notification Process

When the ESP32-CAM identifies an unknown face, it captures an image of the individual, labels it as "Unknown," and automatically uploads it to Dropbox for storage. Simultaneously, the system sends an alert to the user via Telegram in less than 2 seconds, enabling an immediate response to potential intrusions. This process, illustrated in Figure 29, Figure 30, and Figure 31, reinforces the security of the monitored area by providing visual records and real-time notifications.

4-6-3- Periodic Monitoring and Intrusion Prevention

To complement security, the ESP32-CAM captures periodic images every 5 minutes, recording environmental changes and providing a visual history of the monitored area. Compared to previous studies, Chang et al. [16] developed a pipeline leak detection system using ESP32 without integrating AI-based security, while Toridi et al. [18] proposed real-time water level monitoring without identity verification. In contrast, our system is among the first to integrate facial recognition with Telegram alerts and cloud storage, significantly improving security in innovative irrigation systems with IoT.

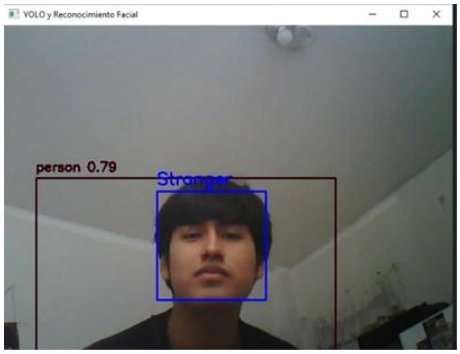

Figure 29. Demonstrates Unknown Detection via ESP32-CAM

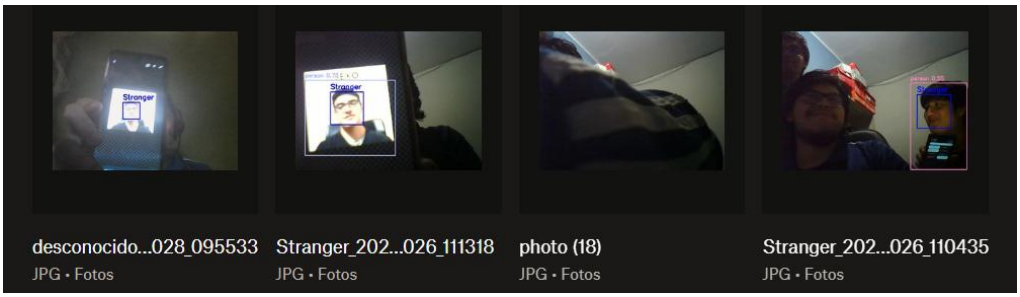

Figure 30. Shows uploading photos to the Dropbox environment

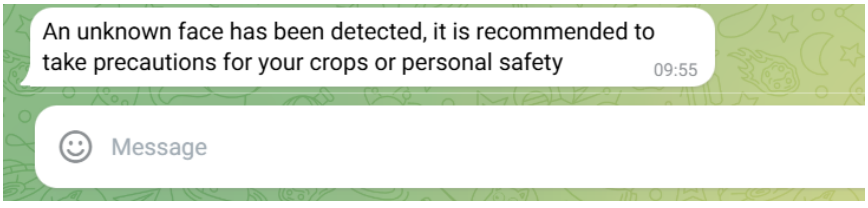

Figure 31. Shows uploading photos to the Dropbox environment

4-7- Live Broadcast

To improve real-time system monitoring, a live stream was implemented using OBS Studio, integrating the video captured by the ESP32-CAM with YouTube Live. This feature allows users to monitor the system status from any location with internet access. Figure 32 shows the configuration in OBS, while Figure 33 illustrates the live stream of a plant in a state of dehydration.

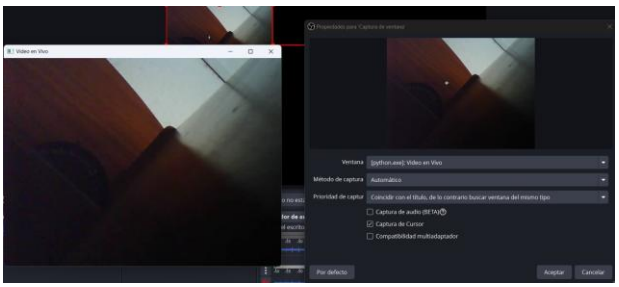

Figure 32. Capturing Video in OBS Environment

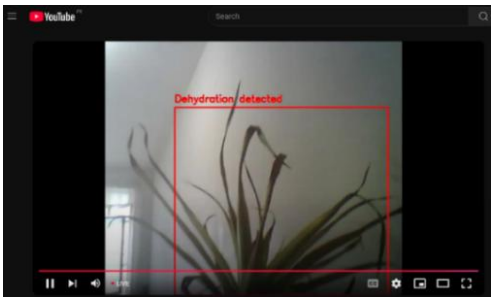

Figure 33. Sample of the Live Broadcast in the YouTube Live environment

During a 4-hour test period, transmission delay times were analyzed, obtaining the following values, as shown in Table 4:

Table 4. Transmission Delay Analysis in Different Environments

Location	Average Delay (s)
Indoors	1.3 s
Park (open area, greater distance from the router)	5.0 s

Transmission delay was observed to be greater outdoors due to the ESP32-CAM's distance from the WiFi router. However, the signal remained stable without interruptions or image freezing, demonstrating the system's reliability in various environments. To mitigate potential connectivity failures, an automatic reconnection system was implemented that allows the ESP32-CAM to re-establish connection in the event of signal loss, improving system resilience and ensuring continuous monitoring.

4-8-Sample of the Live Broadcast in the YouTube Live Environment

Dynamic dashboards were designed in Adafruit IO to display real-time data collected by the sensors. Figure 34 presents two views of the interface: on the left, typical sensor values, and the right, a warning mode where critical values are highlighted to alert the user. Data updates occur every 15 seconds, with a 1.2-second reception delay, allowing for near-real-time viewing. To ensure accurate interpretation of measurements, specific ranges were defined for each sensor, as shown in Table 5. This functionality facilitates the rapid identification of anomalies and allows users to take corrective action in a timely manner.

Figure 34. Adafruit IO Interface with Normal Sensor Statesview (left) and Warning modeview (right)

Table 5. Ranges used for programming the sensors

Sensors	Ranges to take into account	
DHT22	Environmental Humidity	Environmental Humidity
	Very High: >90%	Very High: >35º
	Acceptable: 60%-80%	Acceptable: 20º-30º
	Very Low: <40%	Very Low: <15º
Capacitive Soil Moisture Sensor	Soil Moisture	
	Very High: >80%	
	Acceptable: 40%-60%	
	Very Low:<20%	
YL-83	Presence of Rain	
	Heavy rain: <1000	
	Moderate Rain:1000<x<2000	
	Light Rain: 2000<x<3000	
	There is no rain: >3000	
LDR	Light Intensity	
	Bright: >3000	
	Normal: 2000<x<3000	
	Dark: 1000<x<2000	
	Very Dark: <1000	

4-9- Data Analysis in Adafruit IO

4-9-1- Temperature Analysis in Adafruit IO

The DHT22 sensor recorded temperature data for a month, which was stored in Adafruit IO and displayed in a line graph (Figure 35). In this graph, the Y-axis represents temperature in degrees Celsius (°C), while the X-axis shows the real-time data collection with timestamps automatically generated by the platform. The values obtained ranged from 26.2°C to 27.6°C, staying within the optimal range for healthy plant growth. Temperatures below 15°C can slow growth and increase susceptibility to disease, while temperatures above 35°C can cause dehydration and leaf cell damage. Visualizing trends in Adafruit IO allows users to anticipate changes in environmental conditions, optimizing irrigation and thermal control strategies.

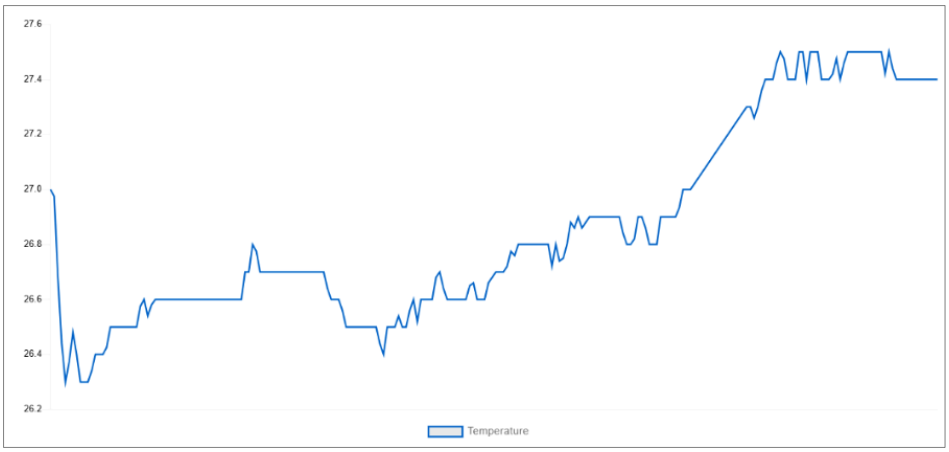

Figure 35. Temperature Sensor Line Diagram

4-9-2- Humidity Analysis in Adafruit IO

Figure 36 illustrates the relative humidity trends recorded by the DHT22 sensor, where the Y-axis expresses humidity in percentage (%) and the X-axis represents real-time data acquisition, with detailed timestamps in Table 6. The recorded values ranged from a minimum of 50% to a maximum of 65%, which are suitable for plant development as they promote nutrient uptake and reduce the risk of disease and water stress. However, levels below 20% can hinder pollination, cause salt buildup, and increase vulnerability to pests. In comparison, levels above 80% can lead to water buildup, fungal growth, deficiencies in root oxygenation, and pollination problems.

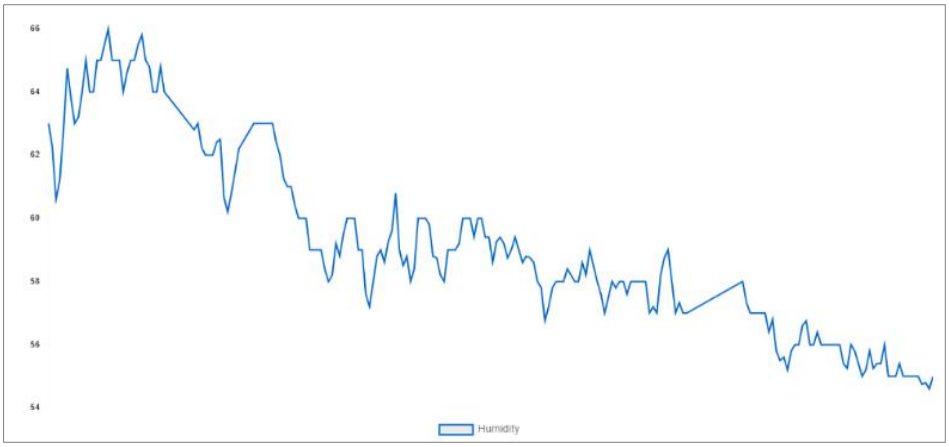

Figure 36. Humidity Sensor Line Diagram

The data collected by the sensors is sent to Adafruit IO, where it is stored along with the date and time of each recording. To complement the graphical representation, Table 6 provides the exact timestamps of the values selected in Figures 35 and 36. This information facilitates a detailed analysis of environmental conditions over time, allowing users to better interpret the graphs and make informed decisions to ensure an optimal environment for plants and avoid problems arising from weather changes or adverse conditions.

Table 6. Data was recorded within the Adafruit IO environment based on Temperature and Humidity sensors.

Date	Hour	Temperature (°C)	Humidity (%)
4/12/2024	11:50:04 AM	26.4	65
4/12/2024	12:07:31 PM	26.6	64
4/12/2024	12:37:30 PM	26.7	63
4/12/2024	1:09:04 PM	26.5	61
4/12/2024	1:41:16 PM	26.79	60
4/12/2024	2:01:47 PM	26.9	59
4/12/2024	2:25:19 PM	27.0	58
4/12/2024	2:48:09 PM	27.4	57
4/12/2024	3:03:11 PM	27.4	56
4/12/2024	3:18:12 PM	27.5	56
4/12/2024	3:33:19 PM	27.4	55

4-10- System Maintenance

Preventive maintenance protocols, detailed in Table 7, have been established to ensure the system's continuous and efficient operation.

Table 7. Component Maintenance Frequency

Component	Maintenance Frequency
Lithium Batteries	Replacement every 2-3 years to ensure a stable power supply.
Sensor Cleaning	Every 6 months to prevent dust and dirt buildup.
Wi-Fi Connectivity	Continuous monitoring to prevent unexpected disconnections.
Facial Recognition	Update authorized users as needed.
Irrigation Expansion	Addition of additional pipes and sprinklers as needed.

4-10-1- Contingency Plans

In addition to scheduled maintenance, automatic recovery mechanisms have been developed to ensure system functionality in the event of potential failures. A backup ESP32 has been configured with the same code and settings, including the MAC address for ESP-NOW, allowing it to be activated in the event of a failure of the main microcontroller. Periodic sensor calibration ensures accurate measurements and prevents errors in decision-making. A relay circuit was

implemented to protect the solenoid valve, preventing electrical overloads and extending the lifespan of the irrigation system. The system also features automatic Wi-Fi and YouTube Live reconnection, detecting network interruptions and reestablishing the connection without manual intervention. Finally, in the event of cloud failures, such as the inoperability of Telegram or Adafruit IO, the system automatically activates irrigation when soil moisture drops below a critical threshold, ensuring plant health even without remote monitoring.

5- Discussion

The results obtained during testing the proposed system demonstrated significant improvements compared to other IoT-based irrigation systems, particularly in water efficiency, plant health monitoring, security, and remote accessibility. This section compares our findings with previous studies, highlighting the innovations and advantages implemented.

5-1- Water Efficiency and Savings

Efficient water management in agriculture has been widely studied, with approaches including the use of IoT and soil moisture sensors. Majid et al. [11] designed an ESP32-based automated flow control system, achieving 92% efficiency in water supply management. Similarly, Korlepara et al. [19] developed a platform for water release valves, reducing water waste. Pereira et al. [13] and Faysal & Mohammed [21] integrated ESP32 and Blynk into drip irrigation systems, reporting improvements in water conservation.

However, most of these studies applied predefined irrigation schemes based on timers or fixed rules, which limits the ability to adapt to unexpected climate changes. In contrast, our system dynamically monitors soil moisture levels, adjusting irrigation in real-time, as shown in Table 8, which compares it with previous studies.

Table 8. Comparison with previous studies

Estudio	Método de control	Fuente de energía	Ahorro de agua (%)
Majid et al. [11]	Flow control with ESP32	Electricity	92 % (water management efficiency)
Pereira et al. [13]	Drip irrigation with ESP32	Electricity	Not reported
Proposed system	Real-time irrigation control with IoT + AI	Solar energy	43.68 % (parks), 43.5 % (flower pots)

Unlike these studies, our solution operates exclusively with solar energy, eliminating dependence on the electrical grid and improving long-term sustainability. Furthermore, by integrating artificial intelligence and real-time monitoring, the system optimizes water distribution and minimizes waste, offering a solution adaptable to different environmental conditions.

5-2- Plant Health Monitoring Using AI

The use of machine learning in agriculture has improved early disease detection. Mehta et al. [22] proposed an ESP32-WROOM-based system for agricultural monitoring, providing numerical sensor data to support decision-making; however, this system did not implement real-time image processing, limiting its ability to identify plant visual problems. In contrast, the proposed system integrates YOLOv3-Tiny into an ESP32-CAM, enabling real-time classification of plants into three categories (healthy, dehydrated, and chlorotic), achieving an accuracy of 93.86%. Unlike previous studies, this system processes images directly on the ESP32-CAM, eliminating the need for external servers and reducing latency and operating costs. Furthermore, the model was trained with 1,200 images per class, which improves its accuracy in real-world scenarios. Compared to Rao et al. [12], whose solution relied on cloud processing, this approach enables autonomous monitoring adapted to diverse agricultural conditions. These results validate the feasibility of machine learning on low-cost hardware, facilitating the implementation of intelligent monitoring in urban and rural agrarian environments.

5-3- Security and Real-Time Monitoring

Security in IoT-based irrigation systems is an aspect that has not been addressed in previous studies. Chang et al. [16] proposed a system with data encryption to prevent unauthorized access, while Toridi et al. [18] developed a wireless water level monitoring system; however, neither of these studies incorporated real-time authentication methods. In contrast, our proposal integrates facial recognition with YOLOv3-Tiny and Haar cascades, achieving 95% accuracy in detecting authorized users and sending alerts via Telegram with a 1.5-second response time, enabling rapid reaction to unauthorized access. Another innovative aspect is live streaming using ESP32-CAM and OBS, which allows remote monitoring via YouTube Live. As shown in Table 9, the proposed system outperforms previous research by combining data encryption, water level monitoring, facial recognition, and real-time streaming, significantly improving security and accessibility compared to other studies.

Table 9. Comparison of security and real-time monitoring with previous studies

Characteristics	Chang et al. [16]	Toridi et al. [18]	Proposed system
Data encryption	Yes	No	Yes
Water level monitoring	No	Yes	Yes
Facial recognition	No	No	Yes (95 % accuracy)
Live streaming	No	No	Yes (YouTube Live)

5-4- User Feedback on System Setup and Operation

To validate the system's accessibility and ease of use, a study was conducted with 10 users living near the park. Key aspects were evaluated, such as installation, integration with cloud services, and the usefulness of real-time notifications and monitoring.

The results are presented in Table 10, which shows high acceptance regarding ease of installation and intuitive design. However, opportunities for improvement were identified, such as incorporating images into Telegram alerts and optimizing Dropbox storage to avoid unnecessary captures.

Table 10. Results of the opinion survey on the system's usability

Aspect Evaluated	% Acceptance	User Comments
Ease of Installation	100%	Quick and easy installation.
Telegram Alerts	80%	Useful but suggest including images in notifications.
Dropbox Integration	90%	Beneficial, but capturing photos every 5 minutes is unnecessary.
Adafruit IO interface	100%	Intuitive and straightforward, with color coding.
Live streaming	70%	Useful, but some consider it dispensable.

This feedback provides valuable information for future system improvements, optimizing its functionality and user experience.

6- Conclusion

This study presented an intelligent irrigation system based on IoT, machine learning, and solar energy, optimizing water consumption, real-time monitoring, and system sustainability. Two ESP32 microcontrollers and an ESP32-CAM enabled irrigation automation using sensor data, with AI-based detection using a modified YOLOv3-Tiny model. Additionally, the integration of facial recognition improved security by restricting unauthorized access, while platforms such as Telegram, Adafruit IO, Dropbox, and YouTube Live facilitated real-time remote monitoring. Unlike traditional irrigation systems, which operate on fixed schedules and with manual interventions, the proposed system dynamically adapts irrigation based on environmental conditions, achieving water savings of 43.68% in parks and 43.5% in flowerpots. Implementing solar energy reduces dependence on the electrical grid, ensuring self-sufficient and sustainable operation.

Compared to other IoT irrigation solutions, this system automates irrigation and analyzes plant health using AI, achieving 93.86% accuracy in detecting dehydration and chlorosis. Furthermore, system security is enhanced with facial recognition and Telegram alerts, ensuring comprehensive control. These findings confirm the system's viability in precision agriculture and urban environments, promoting efficient water use and better crop management.

6-1- Future Works

Several development lines are proposed to improve the system, including expanding the coverage area through more humidity sensors and microcontrollers to manage larger irrigation areas. Likewise, the implementation of ESPHome with Home Assistant is proposed to offer advanced monitoring and overcome the limitations of Adafruit IO. In terms of security, the incorporation of audible alarms triggered by the ESP32-CAM to alert in case of intruders is being considered. Finally, the training of the YOLOv3-Tiny model is intended to expand to detect more complex diseases, such as fungal infections, pests, and leaf burn. These improvements will make the system more scalable, robust, and adaptable, consolidating it as an innovative solution for smart irrigation and sustainable agriculture.

7- Declarations

7-1- Author Contributions

Conceptualization, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F, R.C.M., M.C.C., and C.C.V.; methodology, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; software J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; validation, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; formal analysis, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; investigation,

J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; resources, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; data curation, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; writing—original draft preparation, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; writing—review and editing, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; visualization, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; supervision, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; project administration, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V.; funding acquisition, J.C.O., J.C.B., M.S.C., S.M.M., J.M.I., J.M.F., R.C.M., M.C.C., and C.C.V. All authors have read and agreed to the published version of the manuscript.

7-2- Data Availability Statement

Data sharing is not applicable to this article.

7-3- Funding

The authors received financial support from Universidad César Vallejo for the research, authorship, and/or publication of this article.

7-4- Acknowledgements

The authors would like to thank their families for their support.

7-5- Institutional Review Board Statement

Not applicable.

7-6- Informed Consent Statement

The authors declare that only images of the authors and one of their pets are presented in the manuscript. No individuals or animals belonging to third parties are presented.

7-7- Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancies have been completely observed by the authors.

8- References

[1] Addo-Bankas, O., Wei, T., Zhao, Y., Bai, X., Núñez, A. E., & Stefanakis, A. (2024). Revisiting the concept, urban practices, current advances, and future prospects of green infrastructure. Science of the Total Environment, 954, 176473. doi:10.1016/j.scitotenv.2024.176473.

[2] Ingrao, C., Strippoli, R., Lagioia, G., & Huisingh, D. (2023). Water scarcity in agriculture: An overview of causes, impacts and approaches for reducing the risks. Heliyon, 9(8), 18507. doi:10.1016/j.heliyon.2023.e18507.

[3] Rodríguez, C., García, B., Pinto, C., Sánchez, R., Serrano, J., & Leiva, E. (2022). Water Context in Latin America and the Caribbean: Distribution, Regulations and Prospects for Water Reuse and Reclamation. Water (Switzerland), 14(21), 3589. doi:10.3390/w14213589.

[4] Aquise, B. C., Lopez, G. W. L., Meza, J. E. Z., & Jesús Talavera, S. (2024). Optimizing Agricultural Irrigation in Arequipa - Peru, Through an IoT-Enable Automated Sprinkler Irrigation System. SSRG International Journal of Electrical and Electronics Engineering, 11(8), 256–263. doi:10.14445/23488379/IJEEE-V11I8P123.

[5] El-Khozondar, H. J., Mtair, S. Y., Qoffa, K. O., Qasem, O. I., Munyarawi, A. H., Nassar, Y. F., Bayoumi, E. H. E., & Halim, A. A. E. B. A. El. (2024). A smart energy monitoring system using ESP32 microcontroller. E-Prime - Advances in Electrical Engineering, Electronics and Energy, 9, 100666. doi:10.1016/j.prime.2024.100666.

[6] Waldburger, T., Anken, T., Cockburn, M., Walter, A., Hatt, M., Chiang, C., & Nasser, H. R. (2025). Automated irrigation of apple trees based on dendrometer sensors. Agricultural Water Management, 311, 109398. doi:10.1016/j.agwat.2025.109398.

[7] Upadhyay, A., Chandel, N. S., Singh, K. P., Chakraborty, S. K., Nandede, B. M., Kumar, M., Subeesh, A., Upendar, K., Salem, A., & Elbeltagi, A. (2025). Deep learning and computer vision in plant disease detection: a comprehensive review of techniques, models, and trends in precision agriculture. Artificial Intelligence Review, 58(3), 1–64. doi:10.1007/s10462-024-11100-x.

[8] Zakariazadeh, A., Ahshan, R., Al Abri, R., & Al-Abri, M. (2024). Renewable energy integration in sustainable water systems: A review. Cleaner Engineering and Technology, 18, 100722. doi:10.1016/j.clet.2024.100722.

[9] Sheline, C., Grant, F., Gelmini, S., Pratt, S., & Winter V., A. G. (2025). Designing a predictive optimal water and energy irrigation (POWEIr) controller for solar-powered drip irrigation systems in resource-constrained contexts. Applied Energy, 377, 124107. doi:10.1016/j.apenergy.2024.124107.

[10] Chen, L., Hu, Y., Wang, R., Li, X., Chen, Z., Hua, J., Osman, A. I., Farghali, M., Huang, L., Li, J., Dong, L., Rooney, D. W., & Yap, P. S. (2024). Green building practices to integrate renewable energy in the construction sector: a review. Environmental Chemistry Letters, 22(2), 751–784. doi:10.1007/s10311-023-01675-2.

[11] Majid, A., Jamaaluddin, Wiguna, A., Setiawan, H., & Farihah, A. (2023). Development of an Automatic Water Flow Sensor System Using ESP32 for Efficient Water Control. IOP Conference Series: Earth and Environmental Science, 1242(1), 12016. doi:10.1088/1755-1315/1242/1/012016.

[12] Rao, T. S., Pranay, P., Narayana, S., Reddy, Y., Sunil, & Kaur, P. (2021). ESP32 Based Implementation of Water Quality and Quantity Regulating System. Proceedings of the 3rd International Conference on Integrated Intelligent Computing Communication & Security (ICIIC 2021), 4, 122–129. doi:10.2991/ahis.k.210913.016.

[13] Pereira, G. P., Chaari, M. Z., & Daroge, F. (2023). IoT-Enabled Smart Drip Irrigation System Using ESP32. Internet of Things, 4(3), 221–243. doi:10.3390/iot4030012.

[14] Lin, J. Y., Tsai, H. L., & Lyu, W. H. (2021). An integrated wireless multi-sensor system for monitoring the water quality of aquaculture. Sensors, 21(24), 8179. doi:10.3390/s21248179.

[15] Mamat, N. H., Shazali, H. A., & Othman, W. Z. (2022). Development of a Weather Station with Water Level and Waterflow Detection using Arduino. Journal of Physics: Conference Series, 2319(1), 12020. doi:10.1088/1742-6596/2319/1/012020.

[16] Chang, C. C., Lee, Y. Y., Hou, T. Y., & Yu, C. C. (2022). Information Security in Wireless Water Flow and Leakage Alarm System. Sensors and Materials, 34(6), 2189–2197. doi:10.18494/SAM3811.

[17] Abu Sneineh, A., & Shabaneh, A. A. A. (2023). Design of a smart hydroponics monitoring system using an ESP32 microcontroller and the Internet of Things. MethodsX, 11, 102401. doi:10.1016/j.mex.2023.102401.

[18] Mat Toridi, N., Amri, M. I. H., Mohd Hisham, M. S. S., Ismail, N. S., Kupo, A. D. A., Mohamed Ramli, N., Khairudin, N., & Abd Aziz, S. (2024). Wireless Water Level Detection System. Advances in Agricultural and Food Research Journal, 5(1), 1-11. doi:10.36877/aafrj.a0000490.

[19] Prakash Korlepara, N. S. D., Narasimha Raju, V. S. N., Satyanarayana, P. V. V., Sunil Kumar, V., Jahnavi Priya, Y., & Harsha Vardan, D. (2024). Real-Time Precision Irrigation System for Optimal Crop Yield and Water Conservation. IOP Conference Series: Earth and Environmental Science, 1375(1), 12019. doi:10.1088/1755-1315/1375/1/012019.

[20] Satriyo, P., Nasution, I. S., & F'Alia, S. (2024). IoT-enable smart agriculture using multiple sensors for sprinkle irrigation systems. IOP Conference Series: Earth and Environmental Science, 1290(1), 12027. doi:10.1088/1755-1315/1290/1/012027.

[21] Faysal, Z., & Mohammed, G. (2021). Remote Farm Monitoring and Irrigation System. AL-Rafidain Journal of Computer Sciences and Mathematics, 15(2), 123–138. doi:10.33899/csmj.2021.170016.

[22] R. Mehta, K., Jayant Naidu, K., Baheti, M., Parmar, D., & Sharmila, A. (2023). Internet of Things Based Smart Irrigation System Using ESP WROOM 32. Journal on Internet of Things, 5, 45–55. doi:10.32604/jiot.2023.043102.

[23] Zid, C., Kasim, N., Raza Soomro, A., & Laidoune, A. (2020). The discrepancy in the construction industry of Malaysia: One of the most contributing industries in Malaysia's economy and the highest contributor of the fatal accidents. IOP Conference Series: Materials Science and Engineering, 788(1), 12034. doi:10.1088/1757-899X/788/1/012034.

[24] Idris, F., Latiff, A. A., Buntat, M. A., Lecthmanan, Y., & Berahim, Z. (2024). IoT-based fertigation system for agriculture. Bulletin of Electrical Engineering and Informatics, 13(3), 1574–1581. doi:10.11591/eei.v13i3.6829.

[25] Lakhiar, I. A., Yan, H., Zhang, C., Wang, G., He, B., Hao, B., Han, Y., Wang, B., Bao, R., Syed, T. N., Chauhdary, J. N., & Rakibuzzaman, M. (2024). A Review of Precision Irrigation Water-Saving Technology under Changing Climate for Enhancing Water Use Efficiency, Crop Yield, and Environmental Footprints. Agriculture (Switzerland), 14(7), 1141. doi:10.3390/agriculture14071141.

[26] van Klompenburg, T., Kassahun, A., & Catal, C. (2020). Crop yield prediction using machine learning: A systematic literature review. Computers and Electronics in Agriculture, 177, 100718. doi:10.1016/j.compag.2020.105709.

[27] Rumbayan, M., Pundoko, I., Sompie, S. R., & Ruindungan, D. G. (2025). Integration of smart water management and photovoltaic pumping system to supply domestic water for rural communities. Results in Engineering, 25, 103966. doi:10.1016/j.rineng.2025.103966.

[28] Routis, G., & Roussaki, I. (2023). Low Power IoT Electronics in Precision Irrigation. Smart Agricultural Technology, 5, 100310. doi:10.1016/j.atech.2023.100310.

[29] Shafique, R., Khan, S. H., Ryu, J., & Lee, S. W. (2025). Weather-Driven Predictive Models for Jassid and Thrips Infestation in Cotton Crop. Sustainability (Switzerland), 17(7), 2803. doi:10.3390/su17072803.

[30] Rumetna, M. S., Lina, T. N., Rajagukguk, I. S., Pormes, F. S., & Santoso, A. B. (2022). Payroll Information System Design Using Waterfall Method. International Journal of Advances in Data and Information Systems, 3(1), 1–10. doi:10.25008/ijadis.v3i1.1227.

[31] Roman, A., & Mnich, M. (2021). Test-driven development with mutation testing – an experimental study. Software Quality Journal, 29(1), 1–38. doi:10.1007/s11219-020-09534-x.

[32] INEI. (2022). A glimpse of Peru in figures. Instituto Nacional de Estadística e Informática - Plataforma del Estado Peruano. Gobierno Del Perú. Available online: https://www.gob.pe/en/institucion/inei/informes-publicaciones/6421190-una-mirada-al-peru-en-cifras (accessed on May 2025).

[33] Kurniasari, A. A., Puspitasari, P. S. D., Perdanasari, L., Yuana, D. B. M., & Jumiatun. (2025). Enhancing Hydroponic Systems with ESP32: An IoT Approach to Real-Time Monitoring and Automation. IOP Conference Series: Earth and Environmental Science, 1446(1), 12010. doi:10.1088/1755-1315/1446/1/012010.

[34] Elhattab, K., Abouelmehdi, K., & Elatar, S. (2023). New Model to Monitor Plant Growth Remotely using ESP32-CAM and Mobile Application. Proceedings - 10th International Conference on Wireless Networks and Mobile Communications, WINCOM 2023, 10322939. doi:10.1109/WINCOM59760.2023.10322939.