iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XII **Month of publication:** December 2025

DOI: https://doi.org/10.22214/ijraset.2025.76574

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

Smart Irrigation Control Using IOT Sensors and Machine Learning for Optimized Water Management

Upangshu Basak[1], Swarnendu Chakraborty[2]
[1]*Heritage Institute of Technology, Kolkata*
[2]*Institute of Engineering and Management, Kolkata*

Abstract: ***With the rapid advancement of digital technologies and the growing global demand for sustainable agricultural practices, traditional irrigation systems are undergoing a transformative evolution. Agriculture, being one of the most water-intensive sectors, faces increasing pressure to optimize water usage while ensuring maximum crop yield. Smart irrigation systems—leveraging sensors, microcontrollers, and predictive algorithms—have emerged as a solution to this problem, offering automated water management based on real-time soil and environmental data. In this project, an intelligent irrigation system was developed that integrates Internet of Things (IoT) components with a machine learning (ML) model to automate irrigation decisions. A hardware prototype was constructed using a soil moisture sensor and a controller capable of activating a water pump based on environmental conditions. The system supports both automatic and manual modes, allowing flexibility in operation depending on the context and user preference. On the software side, a supervised ML model—specifically a K-Nearest Neighbors (KNN) classifier—was trained using historical sensor data (soil moisture, temperature, and humidity) to predict whether irrigation should be turned ON or OFF. The dataset was preprocessed and normalized, and the model achieved approximately 66% accuracy in its predictions. The prototype successfully demonstrated automated irrigation behavior that responds intelligently to varying soil conditions, conserving water and eliminating the need for manual monitoring. This project not only showcases the feasibility of combining ML and IoT for smart agriculture but also presents a practical framework for developing low-cost, scalable irrigation systems that can be tailored to small-scale and large-scale farms alike. The integration of real-time sensing with predictive analytics holds significant promise in addressing water scarcity and improving farming efficiency. Future enhancements may include more complex models, weather integration, and remote access via mobile applications to further increase system robustness and adaptability.***

Keywords: Smart Irrigation, IoT, Machine Learning, KNN, Soil Moisture Sensor, Automated Agriculture, Water Conservation

I. INTRODUCTION

Efficient water management is critical for sustainable agriculture. Smart irrigation uses sensors (e.g. soil moisture, temperature, humidity) and networked controllers to supply water only when needed. By automatically measuring soil moisture and weather factors, these systems avoid overwatering and adapt to real-time conditions. For example, automated irrigation has been shown to **save water and time** compared to manual methods, helping farmers apply the *optimal* water amount for maximum yield. Modern implementations often use microcontrollers (Arduino, Raspberry Pi) to process sensor data and drive pumps or valves. Machine learning (ML) further enhances these systems by enabling predictive irrigation decisions: prior work reports ML models (KNN, SVM, ANN, Random Forest) achieving **>98% accuracy** in irrigation scheduling. This project combines these ideas into a single system: an IoT prototype reads soil moisture and environment data, and an ML classifier predicts irrigation state (ON/OFF) to control the pump automatically..

II. AIM AND OBJECTIVES

This project aims to develop an intelligent, automated irrigation system that leverages IoT-based sensor data and machine learning algorithms to optimize water usage and improve agricultural efficiency.

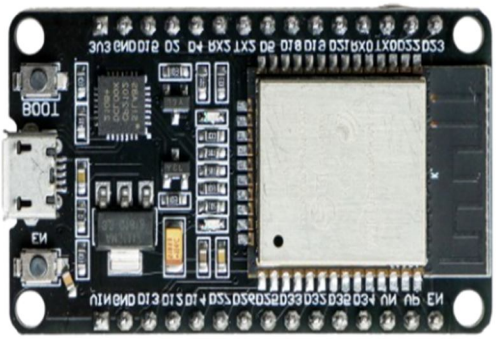
The relevant research objectives of this project are:

1) To design and implement a hardware prototype that collects real-time environmental data such as soil moisture, temperature, and humidity using IoT sensors.

2) To build and train a K-Nearest Neighbors (KNN) machine learning model that predicts whether irrigation should be ON or OFF based on sensor inputs.
3) To integrate the ML model with the microcontroller-based hardware to enable automatic control of the water pump, with an optional manual override mode.
4) To evaluate the performance of the ML model using appropriate accuracy metrics (e.g., confusion matrix, classification accuracy).
5) To analyze the potential benefits of the system in terms of water conservation, cost efficiency, and scalability for precision agriculture.

III. COMPONENTS AND TECHNOLOGY USED

1) *Microcontroller (ESP 32 Devkit V1)*



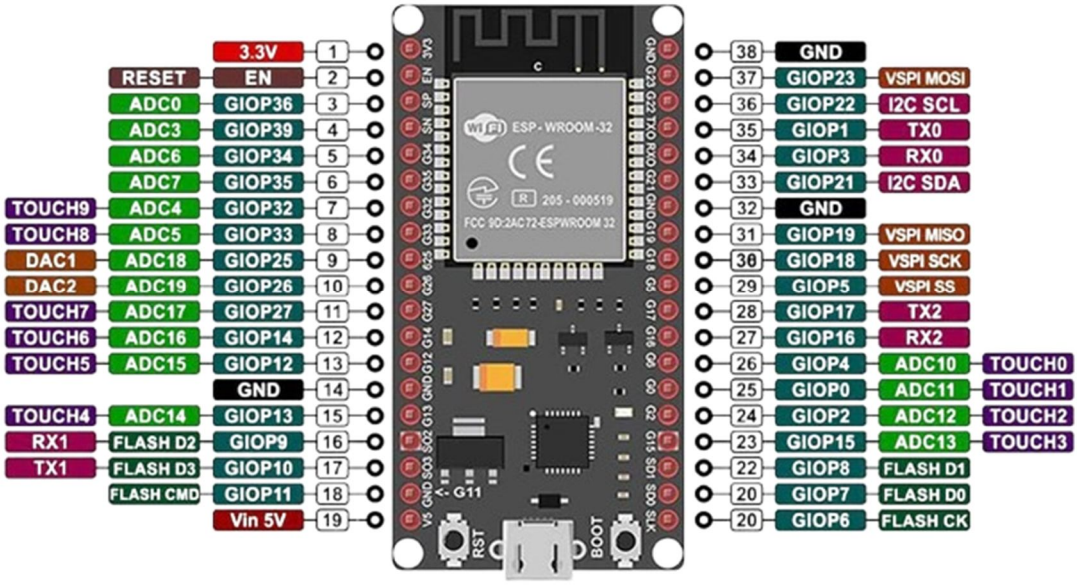
ESP32 is a family of low-cost, low-power System on Chip (SoC) microcontrollers developed by Espressif Systems that integrate Wi-Fi and Bluetooth connectivity, making them ideal for a wide range of Internet of Things (IoT) and embedded projects. Unlike NodeMCU (which strictly refers to firmware originally for the ESP8266 with later support for ESP32), the term "ESP32" commonly refers to the SoC chip itself, the modules built around it, and the development boards that make prototyping easy.

The core of ESP32 is a 32-bit microcontroller that may feature dual-core or single-core CPUs, typically based on the Tensilica Xtensa LX6 architecture (and in some newer variants RISC-V cores), running at clock speeds up to around 240 MHz. These chips include on-chip Wi-Fi (802.11 b/g/n) and Bluetooth Classic & BLE (Bluetooth Low Energy) radios, along with a range of built-in peripherals such as GPIOs, ADCs, DACs, UART, SPI, I²C, PWM, capacitive touch sensors, and more — all integrated on a single chip. ESP32 modules (such as the popular ESP32-WROOM-32) take the bare ESP32 SoC and add flash memory, an antenna interface, and support components required for wireless operation, packaged in a compact form that can be easily embedded on development boards or custom PCBs.

Development boards built around these modules (like the ESP32 DevKit V1) extend these capabilities further by including a USB-to-serial interface, voltage regulation, and headers for all important pins — allowing rapid prototyping on breadboards or custom rigs without complex soldering.

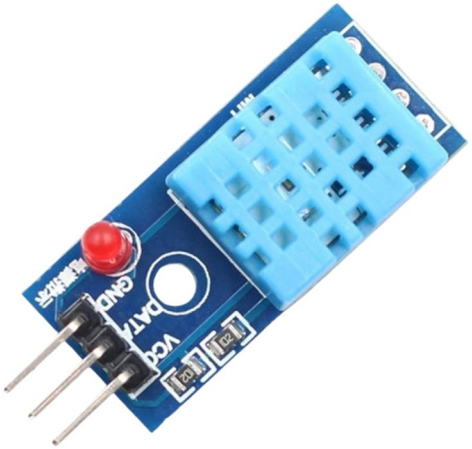
Because of this rich feature set, low cost, and broad software support, the ESP32 is widely used in IoT devices, home automation, wearable electronics, robotics, sensor networks, and other embedded applications. Developers can program ESP32 boards using multiple frameworks and languages, including *Arduino C/C++*, *Espressif's ESP-IDF*, *MicroPython*, *Lua*, and more, giving flexibility depending on project needs and expertise.

2) ESP 32 PIN Configuration



ESP32	GPIOs Connected Sensors/Components
D34	Capacitive moisture sensor AOUT pin
D32	Push-Button to turn ON/OFF pump
D33	Push-Button for changing MODE
D25	TIP122 base pin (controlling pump)
D26	BC547 base pin (controlling Buzzer)
D14	DHT11 sensor O/P pin
D15	LED indicator for MODE
D2	LED indicator for WiFi
D21	SDA of OLED
D22	SCL of OLED

3) DHT11 Sensor



The digital temperature and humidity sensor DHT11 is a composite sensor that contains a calibrated digital signal output of temperature and humidity. The technology of a dedicated digital modules collection and the temperature and humidity sensing technology are applied to ensure that the product has high reliability and excellent long-term stability. The sensor includes a resistive sense of wet component and an NTC temperature measurement device, and is connected with a high-performance 8-bit microcontroller.

4) Soil Moisture Sensor

Soil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighing of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content. The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Reflected microwave radiation is affected by the soil moisture and is used for remote sensing in hydrology and agriculture. Portable probe instruments can be used by farmers or gardeners. Soil moisture sensors typically refer to sensors that estimate volumetric water content. Another class of sensors measure another property of moisture in soils called water potential; these sensors are usually referred to as soil water potential sensors and include tensiometers and gypsum blocks.

5) 0.96" OLED Display

The 0.96-inch OLED display is a small, low-power graphical display module commonly used in embedded systems and IoT projects. It typically features a resolution of 128 × 64 pixels and uses Organic Light-Emitting Diode (OLED) technology, which means each pixel emits its own light and no backlight is required. This results in high contrast, wide viewing angles, and low power consumption compared to LCD displays.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

Most 0.96″ OLED modules are based on the SSD1306 display controller and communicate with microcontrollers using the I2C interface (and sometimes SPI). Due to its small size, readability, and minimal pin usage, it is widely used with boards such as ESP32, ESP8266 (NodeMCU), Arduino, and Raspberry Pi to display sensor data, text, icons, and simple graphics.
The display typically operates at 3.3 V or 5 V, making it compatible with most modern microcontrollers. Its compact form factor and open-source library support make it ideal for rapid prototyping and low-power applications.

6) 1 kΩ 0.25 W Resistors

1 kΩ (1000 ohm), 0.25-watt resistors are fixed passive electronic components used to limit current, divide voltage, and protect sensitive components in electronic circuits. The 0.25 W (¼-watt) power rating indicates the maximum power the resistor can safely dissipate without damage.
These resistors are commonly used in microcontroller-based projects (such as ESP32 and NodeMCU) for applications including pull-up/pull-down configurations, LED current limiting, signal conditioning, and biasing. Their small size, low cost, and reliability make them ideal for general-purpose prototyping and breadboard use.

7) 1N4007 diode

The 1N4007 is a general-purpose silicon rectifier diode commonly used for AC-to-DC conversion, reverse polarity protection, and flyback protection in electronic circuits. It belongs to the 1N400x diode family, with the 1N4007 offering the highest reverse voltage rating among them.
The diode can handle an average forward current of 1 A and a peak reverse voltage of up to 1000 V, making it suitable for power supply and protection applications in low-frequency circuits. Due to its reliability, low cost, and wide availability, the 1N4007 is frequently used in microcontroller-based and power electronics projects.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

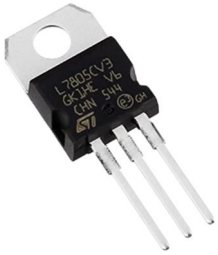
8) BC547 NPN Transistor

The BC547 is a general-purpose NPN bipolar junction transistor (BJT) widely used for signal amplification and low-power switching applications. It is designed to operate with low current and low voltage, making it suitable for interfacing microcontrollers such as ESP32 and NodeMCU with external components.
The transistor can handle a maximum collector current of approximately 100 mA and a collector-emitter voltage of up to 45 V. Due to its small size, reliability, and ease of use, the BC547 is commonly used in switching circuits, amplifier stages, relay drivers (with proper current limiting), and signal conditioning.

9) TIP122 NPN Transistor with heat sink

The TIP122 is a high-power NPN Darlington transistor used for high-current switching applications. It consists of two internal transistors in a Darlington configuration, providing very high current gain, which allows low-current signals from microcontrollers such as ESP32 or NodeMCU to control high-power loads.
The TIP122 can handle collector currents of up to 5 A and collector-emitter voltages up to 100 V. Due to significant power dissipation at high currents, a heat sink is required to prevent overheating and ensure reliable operation. It is commonly used for driving motors, relays, solenoids, lamps, and other high-current devices.

10) 7805 voltage regulator with heat sink

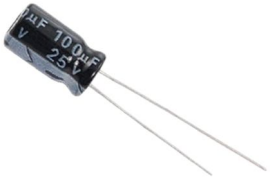


The 7805 is a fixed linear voltage regulator that provides a stable 5 V DC output from a higher input voltage, typically in the range of 7 V to 35 V. It is commonly used to supply regulated power to electronic circuits and microcontroller-based systems.
When the input voltage is significantly higher than 5 V or when supplying higher load currents, the 7805 dissipates excess power as heat. Therefore, a heat sink is required to maintain safe operating temperature and ensure reliable performance. The 7805 is widely used in power supply circuits, embedded systems, and prototyping applications.

11) 5V DC Buzzer

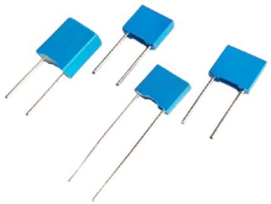
A 5 V DC buzzer is an electroacoustic device used to produce audible alerts and warning sounds in electronic circuits. It operates directly from a 5 V DC supply and is commonly available as an active buzzer (with an internal oscillator) or a passive buzzer (requires an external signal).
5 V buzzers are widely used in microcontroller-based projects such as ESP32 and NodeMCU for applications including alarms, notifications, status indication, and user alerts. Due to their low cost, simple interfacing, and reliable operation, they are ideal for embedded and IoT systems.

12) 100uF 25V capacitor



The 100 µF, 25 V capacitor is a polarized electrolytic capacitor commonly used for filtering, smoothing, and decoupling in power supply and electronic circuits. It helps reduce voltage fluctuations and ripple, providing a more stable DC output.
This capacitor is widely used in voltage regulator circuits (such as with the 7805), microcontroller power rails, and general-purpose electronics. The 25 V voltage rating indicates the maximum safe operating voltage, which should always be higher than the circuit's working voltage to ensure reliability and long service life.

13) 100nF 330nF AC capacitors



The 100 nF and 330 nF capacitors are non-polarized ceramic capacitors commonly used for decoupling, noise filtering, and signal coupling in electronic circuits. Being non-polarized, they can be connected in either direction, making them suitable for AC and high-frequency applications.

These capacitors are typically placed near power supply pins of integrated circuits to suppress high-frequency noise and improve voltage stability. They are widely used in microcontroller-based systems, power regulation circuits, and general electronic applications.

14) 12V DC pump or solenoid valve

A 12 V DC pump or solenoid valve is an electromechanical device used to control the flow of liquids or gases in automated systems. The DC pump is used to move or circulate fluids, while the solenoid valve is used to open or close fluid flow paths when electrically energized.

Both devices operate on a 12 V DC power supply and typically require a driver circuit (such as a transistor or relay) when interfaced with microcontrollers like ESP32 or NodeMCU, due to their higher current requirements. These components are widely used in automation, irrigation systems, fluid control, and IoT-based monitoring applications.

IV. PROPOSED METHODOLOGY

A. Overview of System Architecture

The smart irrigation system developed in this project is designed to autonomously manage water distribution for crops based on real-time environmental conditions. It integrates multiple components including sensors, a microcontroller, and a machine learning model that collaborates to make intelligent irrigation decisions. The system operates in two modes: **automatic**, driven by machine learning predictions, and **manual**, allowing for user override when needed.

The core architecture consists of the following functional units:

1) Sensing Unit
- A soil moisture sensor continuously measures the volumetric water content in the soil.
- Additional sensors like temperature and humidity sensors (e.g., DHT11 or DHT22) monitor atmospheric conditions that impact irrigation needs.
- These sensors are directly connected to an ESP32 microcontroller that serves as the central processing unit.

2) Processing Unit (ESP32 Microcontroller)
- The ESP32 collects data from the sensors at regular intervals.
- It acts as the bridge between physical sensing and the digital logic.
- The microcontroller either uses predefined threshold logic or invokes a trained K-Nearest Neighbors (KNN) model (hosted externally or interpreted in Python for testing) to determine whether the irrigation system should be turned ON or OFF.

3) Machine Learning Decision Logic
- The machine learning model is trained on historical environmental data (soil moisture, temperature, humidity) stored in *TARP.csv*.

- The model outputs a binary prediction (0 = OFF, 1 = ON) which is used by the ESP32 to make real-time decisions.
- In practical deployment, the ESP32 receives the prediction either directly from onboard logic or wirelessly via a connected edge device.

4) Actuation Unit (Pump Controller)

- Based on the model's prediction, the ESP32 triggers a relay module to control a 12V DC water pump or solenoid valve.
- This pump irrigates the soil for a predefined duration or until the moisture crosses a specified threshold.
- The pump status is also displayed in real time on an OLED display for user monitoring.

5) Manual Override Module

- Two push buttons are incorporated: one to manually turn the pump ON, and another to stop it.
- This ensures control even in the absence of model confidence or connectivity, enhancing system reliability.

The system is powered via a regulated 5V–12V DC power source, protected by filtering capacitors and diodes to ensure hardware safety. Components like the 7805 voltage regulator, BC547 NPN transistor, and 1N4007 diode are used to manage voltage regulation and current flow through the circuit.

System Architecture Diagram

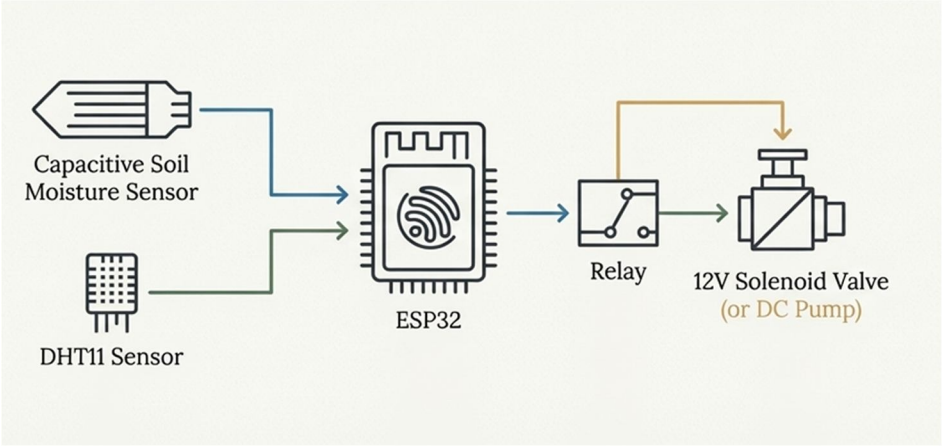


Figure 1: Hardware System Architecture of Smart Irrigation System

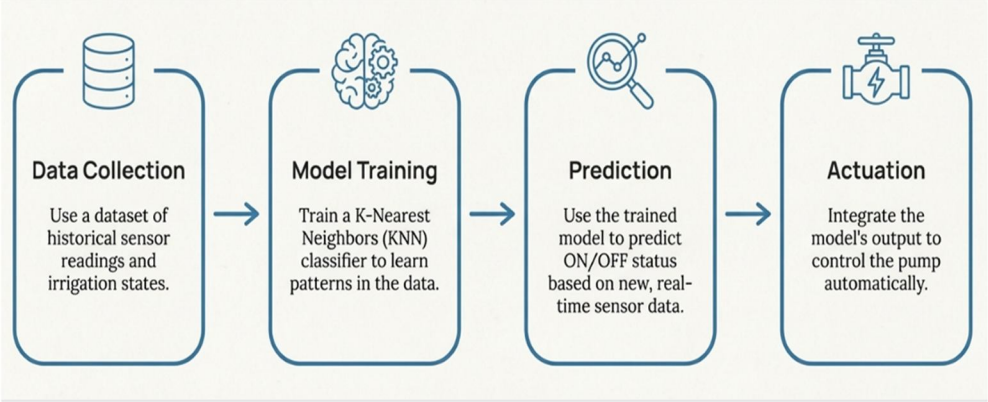


Figure 2: Machine Learning Workflow for Smart Irrigation Prediction

B. Dataset Description

The dataset used in this project serves as the foundational input for training and evaluating the machine learning model responsible for irrigation decision-making. It captures environmental and soil-specific parameters that influence irrigation requirements and records whether irrigation was applied at each time step. The dataset is stored in the file **TARP.csv** and consists of multiple observations collected over time.

1) Features in the Dataset

The dataset contains the following key columns:

- Moisture: Represents the soil moisture level measured by a capacitive soil moisture sensor. Values are typically scaled or normalized to fit within the model input range.
- Temperature: The ambient temperature in degrees Celsius, measured by a DHT11 sensor.
- Humidity: The relative humidity in the air, also recorded by the DHT11 sensor.
- Pump *(Target Variable)*: A binary label indicating whether the pump should be turned **ON (1)** or **OFF (0)** based on the environmental conditions.

Each row in the dataset corresponds to a unique timestamped observation of environmental conditions and the associated irrigation decision. This data simulates real-time monitoring and is used for supervised training of the KNN model.

2) Data Collection and Generation

The data was generated through a combination of simulated and sensor-driven methods. Initially, a set of readings was manually or programmatically created to represent common environmental states encountered during irrigation cycles. These included:

- Low soil moisture and high temperature → irrigation needed (Pump = 1)
- High soil moisture and low-to-moderate temperature → irrigation not needed (Pump = 0)

The readings were compiled to form a labeled dataset, sufficient for training a machine learning model that learns the relationship between environmental inputs and irrigation decisions. In future implementations, this dataset can be expanded with live sensor readings logged over time and stored in CSV format for real-world adaptability.

This structured and labeled dataset is essential in enabling the KNN classifier to learn meaningful patterns that guide the system's ON/OFF irrigation decisions.

C. Data Preprocessing

Load Dataset

```
[5]: df = pd.read_csv("TARP.csv")
     df.head()
```

[5]:

	Soil Moisture	Temperature	Soil Humidity	Time	Air temperature (C)	Wind speed (Km/h)	Air humidity (%)	Wind gust (Km/h)	Pressure (KPa)	ph	rainfall	N	P	K	Status
0	54	22	70	21	19.52	2.13	55.04	6.30	101.50	6.502985	202.935536	90.0	42.0	43.0	ON
1	12	20	40	104	19.49	2.01	55.17	10.46	101.50	7.038096	226.655537	85.0	58.0	41.0	OFF
2	34	26	35	62	19.47	1.90	55.30	14.63	101.51	7.840207	263.964248	60.0	55.0	44.0	ON
3	7	44	44	93	19.54	2.28	54.20	16.08	101.51	6.980401	242.864034	74.0	35.0	40.0	OFF
4	50	38	23	92	19.61	2.66	53.09	17.52	101.51	7.628473	262.717340	78.0	42.0	42.0	OFF

Figure 3: Sample Records from the Smart Irrigation Dataset Showing Soil, Weather, and Irrigation Status Parameters

Before training the machine learning model, several preprocessing steps were performed to ensure that the data was clean, consistent, and suitable for supervised learning. The dataset contained 100,000 records and 15 columns, including sensor readings, environmental data, and the target label for irrigation control.

1) Handling Missing Values

The dataset includes a mix of fully populated columns and others with substantial missing data:

- Columns like Soil Moisture, Temperature, Soil Humidity, and Time have no missing values and are retained.
- Columns such as Air temperature, Wind speed, Air humidity, ph, rainfall, and nutrient values (N, P, K) have many missing entries or are only partially populated.
- These columns were excluded from the training features to maintain dataset consistency.

2) Feature Selection

The following columns were selected for training the KNN classifier:

- Soil Moisture
- Temperature
- Soil Humidity

These three features serve as the environmental inputs for the model. The Status column is the target label indicating whether the pump should be ON or OFF.

3) Label Encoding

The target column Status contains categorical string values ON and OFF. These were encoded to numerical format as follows:

- ON → 1
- OFF → 0

4) Normalization

To ensure equal weighting and improve model performance, all input features were normalized to a 0–1 range using Min-Max scaling:

Figure 4: Feature Selection and Target Variable Encoding for the Irrigation Dataset

This transformation ensures that the distance-based KNN algorithm does not become biased toward features with higher magnitude values.

5) Final Preprocessed Dataset

After these steps:

- The final dataset includes three normalized features and one binary label.
- All records are now free from missing data and are suitable for training and testing a supervised classification model.

D. Machine Learning Model

1) Algorithm Used: K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm was selected as the primary machine learning model for this study. KNN is a supervised, instance-based learning algorithm that classifies data points based on the majority class of their *k* closest neighbors in the feature space, typically using a distance metric such as Euclidean distance.

2) Justification for Model Choice

KNN was chosen for the following reasons:

- Simplicity and interpretability: The algorithm is easy to understand and implement, making it suitable for baseline model evaluation.
- No explicit training phase: KNN does not build a parametric model, which allows it to adapt well to complex decision boundaries.

- Effectiveness on small-to-medium datasets: KNN performs well when the dataset size is manageable and features are properly scaled.
- Baseline comparison: KNN serves as a strong baseline model to compare against more complex algorithms.

Prior to training, feature scaling was applied to ensure that all features contributed equally to the distance calculations.

3) Model Training and Accuracy

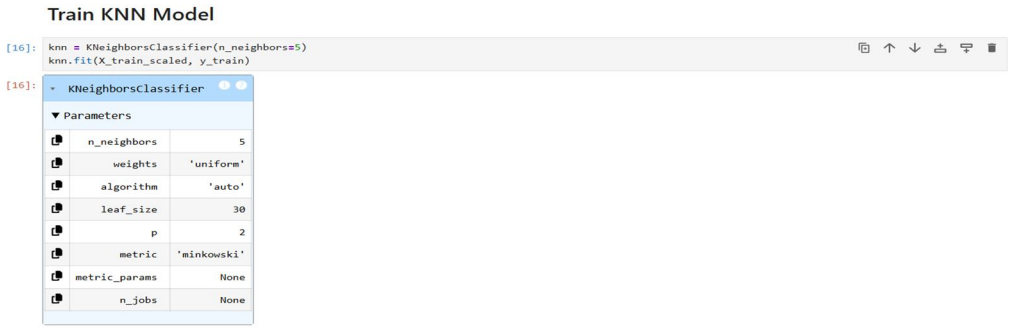


Figure 5: . Configuration and training of the K-Nearest Neighbors classifier with k = 5.

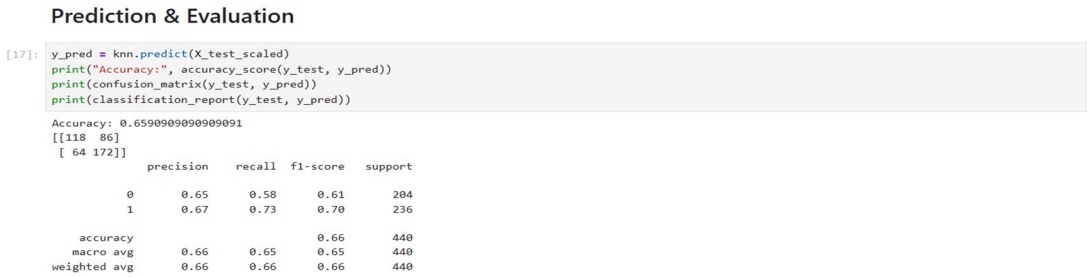


Figure 6: Prediction accuracy and classification report of the KNN model on the test dataset.

E. Hardware Implementation

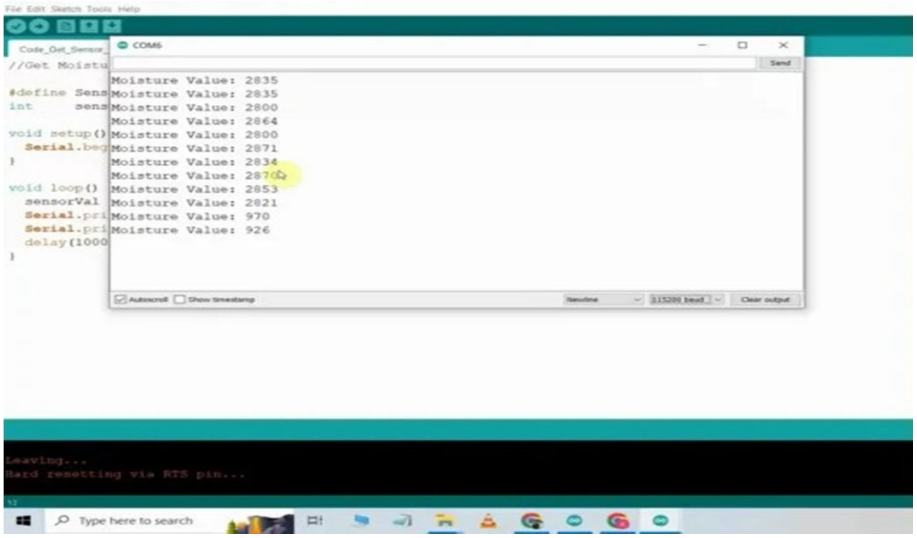


Figure 7: Arduino IDE serial monitor displaying real-time soil moisture sensor values used for system calibration and data acquisition.

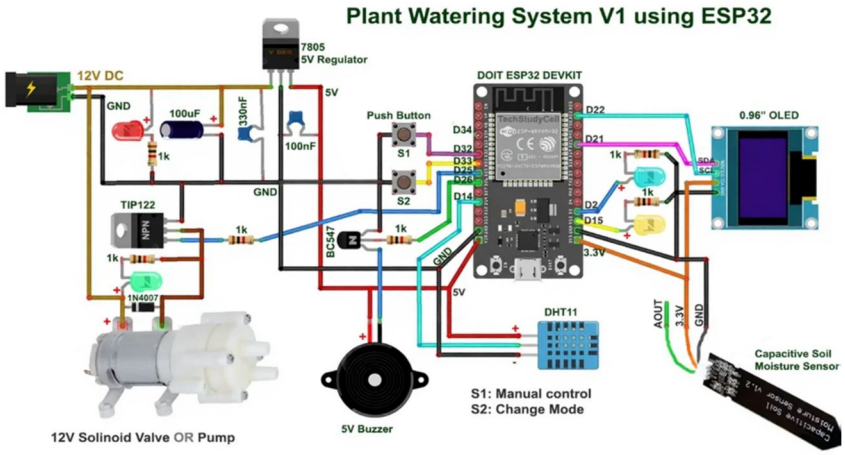


Figure 8: Detailed wiring diagram showing electrical connections among ESP32, sensors, actuators, power supply, and display modules.

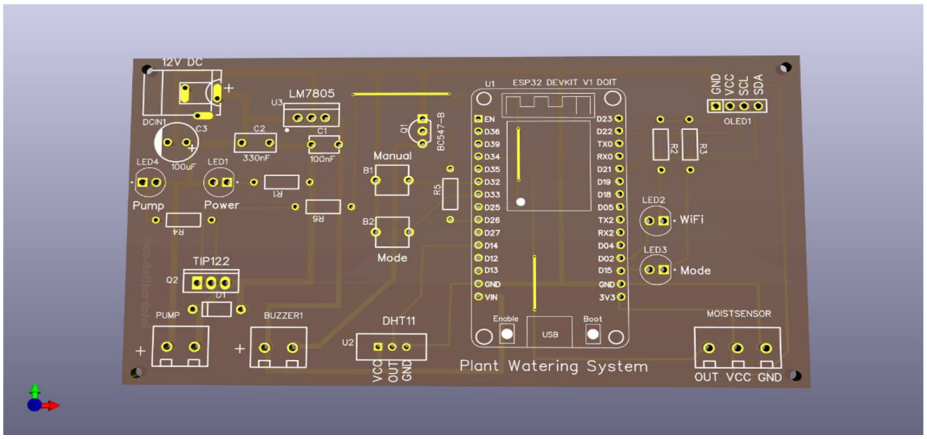


Figure 9: 3D PCB render of the finalized ESP32-based plant watering system.

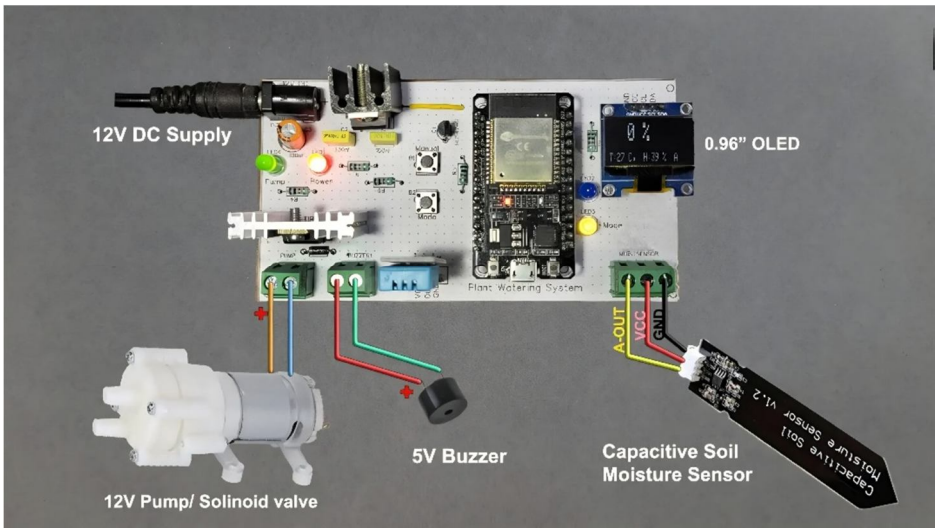


Figure 10: Overall hardware setup showing ESP32 controller, capacitive soil moisture sensor, 12V water pump/solenoid valve, OLED display, buzzer, and power supply.

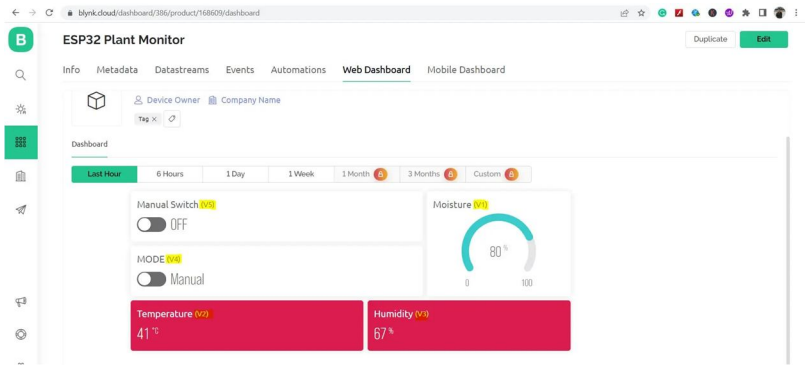


Figure 11: Web-based dashboard displaying soil moisture, temperature, humidity, and irrigation control status.

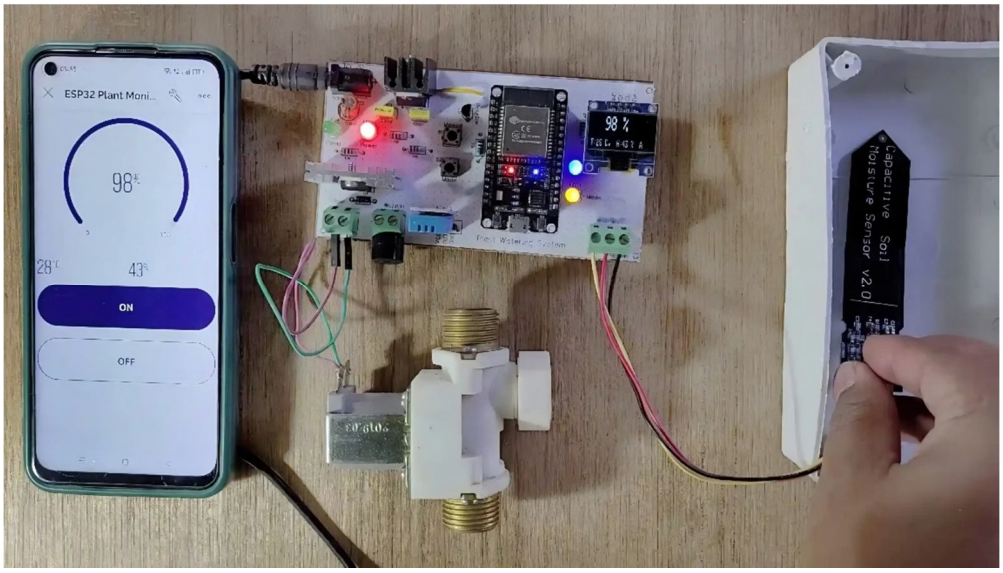


Figure 12: Overall hardware setup showing ESP32 controller, capacitive soil moisture sensor, 12V water pump/solenoid valve, OLED display, buzzer, and power supply.

V. RESULT ANALYSIS & DISCUSSION

This section presents a critical evaluation of the machine learning model used for automating irrigation decisions based on sensor data. The classifier predicts whether the water pump should be turned ON or OFF, using inputs from soil moisture, temperature, and humidity readings.

A. Model Performance

The implemented model, based on the **K-Nearest Neighbors (KNN)** algorithm, was trained on the cleaned and normalized dataset. A standard train-test split of 80:20 was used.

Prediction & Evaluation

```
[17]: y_pred = knn.predict(X_test_scaled)
      print("Accuracy:", accuracy_score(y_test, y_pred))
      print(confusion_matrix(y_test, y_pred))
      print(classification_report(y_test, y_pred))

Accuracy: 0.6590909090909091
[[118  86]
 [ 64 172]]
              precision    recall  f1-score   support

           0       0.65      0.58      0.61       204
           1       0.67      0.73      0.70       236

    accuracy                           0.66       440
   macro avg       0.66      0.65      0.65       440
weighted avg       0.66      0.66      0.66       440
```

Figure 13: Prediction Results and Performance Evaluation of the KNN Classifier

Key performance metrics:

- Accuracy: 65.9%
- Precision (Pump ON): 67%
- Recall (Pump ON): 73%
- F1-Score (Pump ON): 70%
- Precision (Pump OFF): 65%
- Recall (Pump OFF): 58%
- F1-Score (Pump OFF): 61%

The classifier demonstrates moderate accuracy, with better performance in predicting the "Pump ON" state. This prioritization helps avoid under-irrigation, which is often more critical in agricultural settings.

B. Confusion Matrix

The confusion matrix (Figure 14) illustrates the model's classification of the test set:

	Predicted OFF	Predicted ON
Actual OFF	118	86
Actual ON	64	172

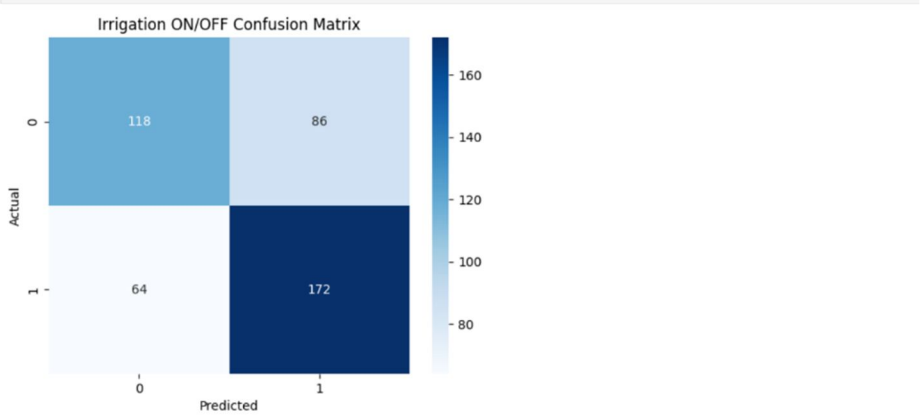


Figure 14: Confusion Matrix for KNN Irrigation Classifier

C. Real-Time Simulation and Prediction Snapshots

Using simulated inputs, the trained model was able to return pump control predictions in real time:

- Test Input 1: [18, 30, 55] → Prediction: OFF
- Test Input 2: [12, 20, 25] → Prediction: ON

Screenshots from the notebook validate that the system reacts correctly to changing sensor values.

Figure 15: Test Output: Pump Status OFF

Figure 16: Test Output: Pump Status ON

D. Strengths of the System

- Real-Time Inference: The model provides rapid predictions that can be directly used to control hardware components like pumps via a microcontroller (ESP32), ensuring timely irrigation.
- Simple and Lightweight Algorithm: KNN is computationally efficient and easy to implement, even on constrained hardware.
- Low-Cost Deployment: The system is built using affordable sensors and components, making it accessible for small-scale farmers or academic use.
- Scalable Design: The ML model can be retrained on updated datasets to adapt to new crops, soil conditions, or climatic zones.
- User-Friendly: The prediction outputs are simple (ON/OFF), reducing complexity in decision-making and making the solution easy to adopt.

E. Limitations of the System

- Sensor Dependency: The quality and calibration of sensors (e.g., soil moisture sensors) heavily influence prediction accuracy. Low-cost sensors may produce noisy or inconsistent readings.
- Generalization Challenge: The model is trained on data from a specific soil and climate environment. Its performance may degrade in different geographies or with different crops unless retrained with new data.
- No Dynamic Learning: The current model does not adapt in real time. New data must be manually collected and used to retrain the model offline.
- Binary Prediction Only: The model currently supports only ON/OFF states. It does not offer graded irrigation levels or time-based scheduling.
- Warning Due to Input Format: During simulation, a user warning was observed because the prediction input did not include feature names. While it doesn't impact predictions, best practices would involve passing inputs with named columns to prevent compatibility issues in the long run.

VI. CONCLUSION

This project successfully demonstrates the integration of IoT hardware with a machine learning-based decision system for smart irrigation. By leveraging real-time data from soil moisture, temperature, and humidity sensors, the system effectively predicts the irrigation requirement and automates the pump's ON/OFF control using a K-Nearest Neighbors (KNN) classifier. The model achieved an accuracy of approximately 66%, with higher reliability in detecting irrigation needs than withholding them, aligning with agricultural risk priorities.

The simplicity of the model and its compatibility with low-cost hardware like the ESP32 make it a promising solution for resource-constrained environments. Additionally, the user-friendly binary prediction approach ensures accessibility even for non-technical users. Overall, the project demonstrates how machine learning, when paired with embedded systems, can offer practical, automated solutions for sustainable farming and water conservation.

VII. FUTURE SCOPE

While the current implementation serves as a robust prototype, several enhancements can elevate its performance, adaptability, and scalability:

1) Real-Time Learning: Implementing online learning algorithms or periodic re-training can help the system adapt to seasonal or crop-based changes.

2) Sensor Calibration & Accuracy: Replacing basic sensors with calibrated industrial-grade ones can reduce noise and improve reliability in predictions.
3) Fuzzy Control or Multi-Level Irrigation: Instead of a binary ON/OFF system, integrating fuzzy logic or regression models can help modulate irrigation intensity based on exact moisture deficits.
4) Cloud Integration: Adding cloud-based dashboards or remote alerts can enable real-time monitoring, analytics, and control for farmers.
5) Geolocation-Based Models: Creating datasets for different agro-climatic zones and training specialized models can improve generalizability and accuracy.
6) Solar-Powered Systems: Integrating solar energy for powering the sensors and ESP32 can make the setup more sustainable and off-grid friendly.
7) Crop-Specific Models: Tailoring the ML model per crop type can optimize irrigation strategies and yield outcomes.

This project lays a solid foundation for an intelligent and efficient irrigation system. With iterative improvements, it holds significant potential for real-world deployment in precision agriculture.

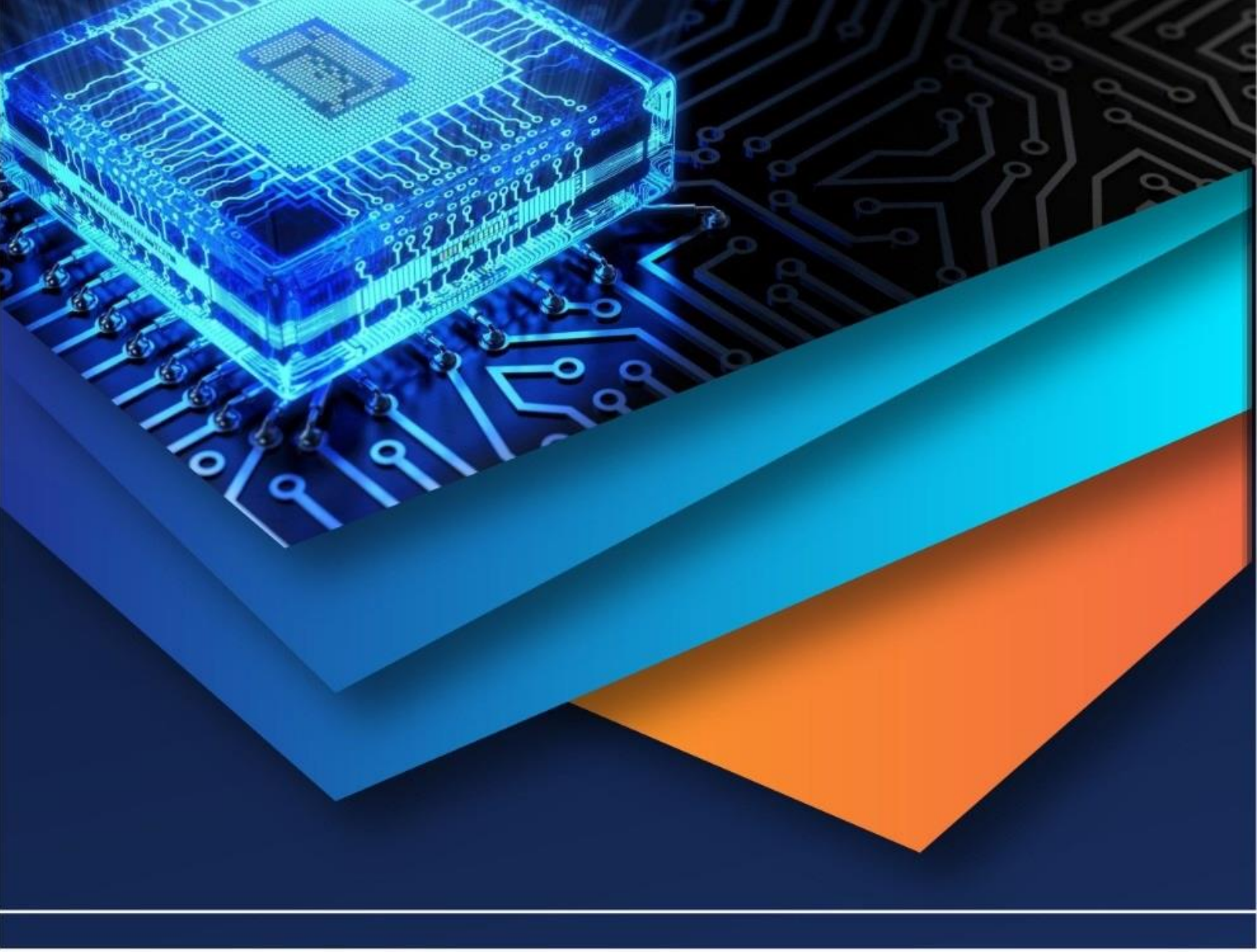
VIII. ACKNOWLEDGEMENT

We would like to express our sincere appreciation for the support and resources that enabled the successful completion of this research work. The collaborative effort on this project reflects the equal contributions made by both authors in their respective areas of development, implementation, and analysis.

We are also grateful for the academic infrastructure, tools, and technical environments that facilitated the execution of our work. Lastly, we extend our heartfelt thanks to our families and peers for their continued encouragement and support throughout this journey.

BIBLIOGRAPHY

[1] Allen, R. G., Pereira, L. S., Raes, D., & Smith, M., 1998. Crop evapotranspiration — Guidelines for computing crop water requirements (FAO Irrigation and Drainage Paper No. 56). Food and Agriculture Organization (FAO). — Standard reference for evapotranspiration and water-requirement calculations used in irrigation design and discussion of climatic inputs. FAOHome. [Accessed on: 22-Dec-2025].
[2] Del-Coco, M., et al., 2024. Machine Learning for Smart Irrigation in Agriculture, Information (MDPI) — Recent survey focusing on ML approaches for irrigation scheduling, useful for literature review and justification of ML use. MDPI. [Accessed on: 22-Dec-2025].
[3] Younes, A., Elamrani Abou Elassad, Z., et al., 2024. The application of machine learning techniques for Smart Irrigation Systems: A systematic literature review, Smart Agricultural Technology / Elsevier — Systematic review of ML models used in smart irrigation (KNN, RF, SVM, ANN) and their empirical performance. Good for comparing model choices. ScienceDirect. [Accessed on: 22-Dec-2025].
[4] Tace, Y., et al., 2022. Smart irrigation system based on IoT and machine learning, (Elsevier / ScienceDirect) — Implementation-focused paper showing an IoT+ML pipeline and performance metrics; helps support architecture and methodology choices. ScienceDirect. [Accessed on: 22-Dec-2025].
[5] Espressif Systems, ESP32-WROOM datasheet (module technical documentation) — Official datasheet and technical reference for the ESP32 module used as the microcontroller/edge device in the hardware. Use this for pinout, power, and wireless specs. Espressif Systems. [Accessed on: 22-Dec-2025].
[6] Espressif Systems, ESP-IDF Programming Guide — Official programming/deployment guide for ESP32 (useful if you flash models or telemetry code to the ESP32). Espressif Systems. [Accessed on: 22-Dec-2025].
[7] DHT11 Technical Data Sheet (manufacturer / distributor PDF) — Datasheet describing electrical characteristics, timing, and measurement accuracy for the DHT11 temperature & humidity sensor used in the project. Use when describing sensor accuracy and calibration limitations. Mouser Electronics+1. [Accessed on: 22-Dec-2025].
[8] DFRobot (Capacitive Soil Moisture Sensor SKU SEN0193) — Product documentation describing capacitive sensing principles, corrosion resilience and recommended wiring (good for hardware / sensing-unit section). wiki.dfrobot.com. [Accessed on: 22-Dec-2025].
[9] SSD1306 OLED Controller — SSD1306 datasheet (Adafruit / S Systech) for the 0.96″ I²C OLED modules. Use this for display interfacing details in hardware description. Adafruit+1. [Accessed on: 22-Dec-2025].
[10] Firebase Realtime Database — Official Firebase docs describing realtime data sync, security rules, and client APIs (useful if your system logs sensor data to the cloud or supports remote control). Firebase+1. [Accessed on: 22-Dec-2025].
[11] scikit-learn documentation — KNeighborsClassifier reference (algorithm description, parameters) and preprocessing (StandardScaler / MinMaxScaler): use these as authoritative technical references for model selection, scaling rationale, and code-level implementation. Scikit-learn+2Scikit-learn+2. [Accessed on: 22-Dec-2025].
[12] LastMinuteEngineers — "Interfacing Capacitive Soil Moisture Sensor with Arduino" (tutorial) — Practical wiring and code examples for capacitive soil sensors and microcontrollers; good for hardware wiring diagrams and code snippets in the Methodology. lastminuteengineers.com. [Accessed on: 22-Dec-2025].
[13] Recent implementations & student prototypes on Smart Irrigation + KNN (examples & proceedings) — several conference and journal implementations (e.g., AIP/ICITMSEE 2024, IJCRT 2021) — good to cite as closely-related prior work showing KNN and RF comparisons in irrigation tasks. Representative sources: Reddy et al. (2024/2025) and IJCRT (2021). pubs.aip.org+1. [Accessed on: 22-Dec-2025].



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY