Arab International University

Faculty of Informatics and Communication Engineering

Report on

AgriFlow AI: An AI-Driven Precision Irrigation System Using Neural Networks, IoT Sensors, and Automated Control

Submitted to

Department of Informatics Engineering

in partial fulfillment of the requirement for the Applied projects Course

Submitted by

Anto Torossian

April 2026

ABSTRACT

AgriFlow AI was developed to address the critical challenge of water inefficiency in small and medium-scale agriculture. The system connects low-cost IoT sensors and a Raspberry Pi microcontroller to a cloud-hosted neural network that autonomously determines the appropriate irrigation schedule. By reading soil moisture, air temperature, humidity, and weather forecast data, the model predicts the required valve-open duration and triggers the irrigation cycle without manual intervention. A web-based dashboard provides real-time monitoring and manual override capabilities for the farmer.

The results of this project build on and extend findings from recent literature in smart irrigation. Prior research has demonstrated that neural networks trained on sensor and meteorological data can achieve soil moisture prediction accuracy suitable for real-world deployment, that machine learning frameworks for evapotranspiration estimation yield measurable water savings, and that even simple predictive models consistently outperform fixed-timer irrigation. Consistent with these findings, AgriFlow AI achieved an R^2 of 0.92 and a mean absolute error of 2.1 minutes on the held-out test set. Integration testing over seven days demonstrated a 30% reduction in water consumption compared to a conventional fixed-timer system, while maintaining soil moisture within the optimal range. Server load testing confirmed stable performance under up to 1,000 concurrent users.

This report covers the complete development process: requirements analysis, system design, implementation, and testing results.

Keywords

Precision Irrigation,Neural Networks,Internet of Things (IoT),Smart Agriculture,Automated Control,AgriFlow AI,MQTT Protocol,Raspberry Pi,Cloud Computing,REST API

ABBREVIATIONS

AI – Artificial Intelligence

IoT – Internet of Things

MQTT – Message Queuing Telemetry Transport

MLP – Multi-Layer Perceptron

API – Application Programming Interface

REST – Representational State Transfer

RMSE – Root Mean Square Error

MAE – Mean Absolute Error

UI – User Interface

ERD – Entity Relationship Diagram

JWT – JSON Web Token

TLS – Transport Layer Security

TABLE OF CONTENTS

INTRODUCTION

Water scarcity represents an increasingly critical global challenge, with agriculture accounting for approximately 70% of all freshwater consumption. Despite this reality, irrigation practices on many small and medium-scale farms have remained largely unchanged: a timer activates and water flows regardless of whether the soil actually requires it. This approach results in either excessive water waste or insufficient crop hydration. With the declining cost of sensors and the growing accessibility of artificial intelligence tools, there is a compelling opportunity to develop systems capable of genuinely addressing this inefficiency.

AgriFlow AI represents an attempt to integrate these enabling technologies into a cohesive, practical solution. A Raspberry Pi microcontroller is deployed near the field, connected to a soil moisture sensor, a temperature and humidity sensor, and a relay module controlling an irrigation valve. Sensor readings are transmitted to a cloud server, where a neural network determines the appropriate timing and duration of irrigation. Control commands are then sent back to the field node to open or close the valve automatically. Farmers interact with the system through a web-based dashboard that displays real-time field status and provides a manual override capability.

The remainder of this report is organized as follows. Chapter 1 describes the project background, problem statement, objectives, scope, features, and system requirements. Chapter 2 presents the theoretical foundations underpinning the system, covering IoT in precision agriculture, neural network approaches for soil moisture prediction, and the MQTT communication protocol. Chapter 3 surveys related commercial products and reviews eight relevant academic studies that informed the design of AgriFlow AI. Chapter 4 details the system analysis, including functional requirements, non-functional requirements, and the use case model. Chapter 5 documents the complete system design: architecture, entity relationship diagram, sequence diagram, hardware specifications, AI prediction engine, and security design. Chapter 6 describes the implementation of each system component. Chapter 7 presents the results and testing outcomes, covering model evaluation, sensor accuracy, integration testing, and server load testing. Chapter 8 offers conclusions, identifies challenges and limitations, and proposes directions for future development.

CHAPTER 1: PROJECT DESCRIPTION

1.1 Background

Irrigation has been a fundamental agricultural practice throughout human history; however, modern technology now enables significantly more intelligent approaches. The primary obstacle is that sophisticated irrigation systems are often prohibitively expensive or technically complex for smaller farming operations, leaving many growers reliant on timer-based systems. The widespread availability of low-cost microcontrollers and open-source machine learning frameworks creates a genuine opportunity to develop affordable smart irrigation solutions. This observation formed the foundation for the AgriFlow AI project.

1.2 Problem Statement

A significant gap exists in the agricultural technology market between basic timer-based irrigation systems, which waste water by failing to adapt to actual soil conditions, and high-end commercial solutions that demand substantial capital investment and specialized technical expertise. Small and medium-scale farmers are therefore left without a viable intermediate option. The objective of this project was to fill this gap by creating a device capable of environmental sensing, data-driven learning, and autonomous actuation, all at a total hardware cost below $150.

1.3 Project Objective

The project aimed to develop a functional prototype capable of accomplishing the following:

- Measuring soil moisture, temperature, and humidity in the field.
- Retrieving weather forecast data to prevent unnecessary irrigation prior to anticipated rainfall.
- Executing a trained neural network to determine precise irrigation duration.
- Autonomously controlling the opening and closing of an irrigation valve.

- Providing farmers with a clear and accessible web interface for real-time monitoring and manual control.

1.4 Project Scope

The scope of this project encompassed the following components:

- Assembly of a Raspberry Pi field node incorporating sensors and a relay module.
- Development of Python scripts for sensor data acquisition and MQTT-based data transmission.
- Construction of a training dataset using synthetic data generation.
- Training of a multi-layer perceptron neural network model.
- Deployment of a cloud backend using Flask and MySQL with a RESTful API.
- Development of a browser-based farmer dashboard.
- Integration and indoor testing of the complete system with a solenoid valve.

Multi-zone irrigation support and a mobile application were explicitly excluded from the current scope and identified as areas for future development.

1.5 Project Features

- Live sensor readings every 15 minutes, capturing soil moisture, temperature, and humidity.
- AI-based irrigation scheduling that learns from recent patterns to determine optimal watering cycles.
- Weather forecast integration to delay irrigation when rainfall is anticipated.
- Fully automatic valve control without requiring farmer intervention.
- A RESTful API backend providing secure data storage and retrieval.
- A farmer dashboard displaying field status, upcoming irrigation schedules, and supporting manual triggers.
- A one-click manual override to initiate or cancel an irrigation cycle at any time.

1.6 Project Feasibility

1.6.1 Technical Feasibility

The hardware components employed are industry-standard, widely available, and comprehensively documented. The Raspberry Pi and associated sensors interface through straightforward wiring configurations. The AI inference workload operates efficiently on a standard cloud virtual private server, and the MQTT protocol provides reliable communication even under intermittent network conditions.

1.6.2 Economic Feasibility

The total hardware cost was below $150. The water savings achievable over a single growing season would comfortably offset this initial investment, making the system economically viable even for small-scale farming operations.

1.6.3 Operational Feasibility

Once installed, the system operates with minimal maintenance requirements. The dashboard interface is sufficiently intuitive that users without a technical background can operate it effectively without specialized training.

1.7 System Requirements

1.7.1 Hardware Requirements

- Raspberry Pi 4 (4 GB RAM)
- Capacitive soil moisture sensor
- DHT22 temperature and humidity sensor
- MCP3008 analog-to-digital converter
- 5V relay module

- 12V solenoid valve
- 12V DC power adapter
- Connecting wires, breadboard, and enclosure

1.7.2 Software Requirements

- Raspberry Pi OS
- Python 3.10
- TensorFlow 2.x, Keras, and Scikit-learn
- NumPy and Pandas
- Flask and Flask-RESTful
- Mosquitto MQTT broker
- MySQL database
- HTML, CSS, JavaScript, and Chart.js
- Postman (for API testing)

CHAPTER 2: THEORETICAL STUDY

2.1 IoT in Precision Agriculture

Precision agriculture is a data-driven approach to farm management that applies resources according to the actual, spatially variable needs of different areas within a field. The Internet of Things provides the data acquisition infrastructure for this paradigm: compact sensors distributed across a farm continuously monitor parameters such as soil moisture, temperature, and humidity, transmitting readings to a centralized processing system. This eliminates reliance on estimation and provides farmers with empirical data. For irrigation applications, this means activating water supply only where and when the soil genuinely requires it. Most IoT implementations in agricultural settings employ lightweight communication protocols such as MQTT, which is well-suited to the constraints of low-power, low-bandwidth embedded devices.

2.2 Neural Networks for Soil Moisture Prediction

Soil moisture dynamics are governed by numerous interacting variables, including meteorological conditions, time of day, recent irrigation events, and crop characteristics. These complex, non-linear relationships are well-suited to neural network modeling when sufficient training examples are available. In this project, a feedforward multi-layer perceptron was selected as the predictive model. The input feature vector includes the current moisture measurement, the three preceding moisture readings, temperature, humidity, time of day encoded as sine and cosine components to capture the cyclic daily pattern, and the probability of precipitation. The model outputs a single continuous variable representing the recommended irrigation duration in minutes. The non-linear nature of the underlying relationships makes a neural network substantially more accurate than a rule-based approach.

2.3 MQTT Publish/Subscribe Protocol

MQTT is a lightweight messaging protocol based on the publish-subscribe pattern. Devices publish data to named topics managed by a central broker, and any client subscribed to a given

topic receives the published data immediately. This architecture is highly efficient in terms of bandwidth and computational overhead. Within AgriFlow AI, the field node publishes sensor readings to topics such as field/sensor/moisture, while the backend subscribes to these topics to store incoming data. Conversely, the backend publishes actuation commands to the topic field/control/valve, to which the field node subscribes in order to operate the relay and valve accordingly.

CHAPTER 3: RELATED WORK

3.1 Similar Applications

Several commercial products address smart irrigation, yet each exhibits notable limitations. The following table provides a comparative overview of existing solutions relative to AgriFlow AI.

Table 1. Comparison of Similar Irrigation Systems

Feature	CropX	Netafim	SmartWater	AgriFlow AI
AI Predictive Model	✓	✗	✗	✓ (neural net)
Automated Valve Control	✗	✓	✗	✓
Multi-Field Dashboard	✗	✓	✓	✓
Closed-Loop Feedback	✓	✗	✗	✓
Low-Cost Implementation	✗	✗	✓	✓
Open API	✓	✗	✗	✓

3.2 Literature Review

Recent research has increasingly emphasized the need for structured, reproducible, and domain-aware approaches to conducting and documenting technical research, particularly in fields such as Artificial Intelligence (AI) and Software Engineering (SE), where projects inherently involve complex interactions between data, algorithms, system design, and evaluation protocols. Despite the availability of general academic writing guidelines, these approaches remain largely discipline-agnostic and often fail to capture the methodological depth required for engineering-oriented thesis development.

To address this limitation, Barhoum proposed the Structured Engineering Thesis Framework (SETF), a domain-aware and workflow-driven framework specifically designed for graduation thesis development in AI, Software Engineering, and Robotics [8]. Unlike traditional writing

guides that treat documentation as a post-development activity, SETF reconceptualizes thesis development as an integrated research lifecycle, in which problem definition, literature review, methodology design, implementation, evaluation, writing, and revision are explicitly interconnected within a coherent and iterative workflow.

A key strength of SETF lies in its ability to align technical development processes with scientific reporting practices. In AI-focused research, the framework emphasizes critical methodological components such as dataset preparation, prevention of data leakage, model selection, training-validation-testing protocols, and the use of appropriate evaluation metrics including accuracy, precision, recall, and F1-score. Similarly, in Software Engineering contexts, SETF integrates architectural design, development methodologies (e.g., Agile and DevOps), system implementation, and systematic testing strategies within a unified structure. This alignment ensures that technical decisions are not only implemented but also rigorously documented, justified, and evaluated within a scientific narrative.

Furthermore, SETF introduces a comprehensive evaluation perspective that combines quantitative, qualitative, and comparative analysis, addressing common limitations observed in student projects such as weak baselines, insufficient validation, and lack of reproducibility. By explicitly linking evaluation design to earlier stages of methodology and implementation, the framework enhances the reliability and interpretability of research outcomes.

Another important contribution of SETF is its support for reproducibility and publication-oriented research. By structuring the research process as a transparent and iterative workflow, the framework facilitates consistent documentation of experimental settings, design decisions, and evaluation procedures, thereby enabling independent verification and extension of the work. This is particularly relevant in modern AI and SE research, where reproducibility has become a fundamental requirement.

Overall, SETF provides a structured and citable reference model that bridges the gap between engineering practice and academic writing, offering a systematic approach for developing high-quality, coherent, and potentially publishable graduation theses in technical disciplines [8].

Luiz, Fialho, and Teixeira [1] investigated the use of deep neural networks for soil moisture prediction, training their model on a ten-year field sensor dataset. Their findings demonstrated that incorporating missing data handling techniques and using lagged moisture readings as temporal features significantly improved prediction accuracy, reaching 92.8%, which is suitable for real-world deployment.

Yeung, Bunker, and Fujii [2] developed a machine learning framework for estimating evapotranspiration by integrating weather API data. Employing Random Forest and Support Vector Regression trained on FAO climatic database records, their pipeline achieved an accuracy of 89.5% and generated irrigation schedules that resulted in measurable reductions in water consumption.

Zaveri, Tiwari, and Teli [3] evaluated smart irrigation decision-making processes by comparing Logistic Regression and Decision Tree classifiers against real-time IoT sensor history, achieving 85.2% accuracy. Their study confirmed that any form of predictive modeling, even a simple classifier, consistently outperforms conventional fixed-timer irrigation in terms of water efficiency.

Kashyap, Kumar, and Singh [4] applied Long Short-Term Memory networks to model soil moisture temporal dynamics for smart irrigation scheduling. Although LSTM-based approaches delivered superior long-range forecasts compared to standard feedforward architectures, this improvement came at the cost of significantly increased computational complexity, a trade-off relevant to resource-constrained field deployments.

Adeyemi, Grove, and Peets [5] proposed a reinforcement learning agent trained through simulated trial-and-error to manage greenhouse irrigation autonomously. The agent achieved approximately 25% water savings, demonstrating the potential of adaptive, self-improving control strategies that require no prior labeled dataset.

Garcia, Parra, and Rocher [6] addressed the challenge of remote farm deployments by embedding a compact convolutional neural network directly on a microcontroller for real-time irrigation decision-making without internet connectivity. While the on-device model exhibited

marginally lower accuracy, the approach represents a critical step toward fully autonomous edge-AI irrigation systems.

Goap, Sharma, Shukla, and Krishna [7] built an IoT-based smart irrigation management system combining cloud-hosted machine learning with a web dashboard interface, a design closely aligned with the architecture of AgriFlow AI. Their results validated the conceptual viability of coupling IoT sensor networks with cloud-based predictive models for automated irrigation control.

A consistent observation across the reviewed literature is that very few systems close the complete loop from environmental sensing, through predictive modeling, to physical actuation, while simultaneously providing the farmer with an accessible interface. AgriFlow AI was designed specifically to address this gap by integrating all four components within a single low-cost prototype.

CHAPTER 4: SYSTEM ANALYSIS

4.1 Requirements Specification

The system must reliably acquire environmental data, generate accurate predictions, execute actuation commands, and maintain the farmer in an informed and in-control position throughout the process.

4.2 Functional Requirements

Farmer-facing functional requirements include the following capabilities:

1. Create an account and authenticate through the web interface.
1. Register fields and associate them with their respective sensors and valve controllers.
1. View the latest sensor readings for each registered field.
1. Review the upcoming irrigation schedule and manually initiate or cancel a watering cycle.
1. Receive alerts for anomalous conditions, such as critically low soil moisture or sensor malfunction.
1. Generate water consumption reports for historical analysis.

System-facing automated functional requirements include:

1. Acquire sensor readings every 15 minutes and transmit values via MQTT.
1. Persist all sensor readings to the database.
1. Retrieve weather forecast data once per hour from an external API.
1. Execute the neural network inference daily or on demand to determine irrigation duration.
1. Transmit an MQTT command to the valve controller specifying the required open duration.
1. Maintain a complete log of all irrigation events.

4.3 Non-Functional Requirements

The system is subject to the following non-functional constraints. Regarding performance, the neural network model must return a prediction within 5 seconds, and the dashboard must load within 3 seconds. Security requirements mandate HTTPS for all API communications, JWT-based authentication, and TLS encryption for MQTT messaging. The cloud infrastructure must maintain 99% availability during the agricultural growing season. The system architecture must support scaling to at least 50 concurrent field nodes. The dashboard interface must function effectively on mobile devices as well as desktop browsers.

4.4 Use Case Diagram

The use case diagram below illustrates the interactions between the two primary actors — the Farmer (User) and the IoT Controller (System) — and the core system functions. The Farmer can create an account and log in, where account creation extends the login use case. Once authenticated, the Farmer can follow fields and crops, select the interface language, subscribe monthly (which extends to payment via the Payment Gateway), trigger irrigation valves (which includes a validation step performed by the IoT Controller), view predictive analytics, and view notifications. The IoT Controller communicates directly with the system to execute valve actuation commands, while the Payment Gateway handles subscription billing independently. This diagram captures the complete scope of user-system interaction within the AgriFlow AI platform.

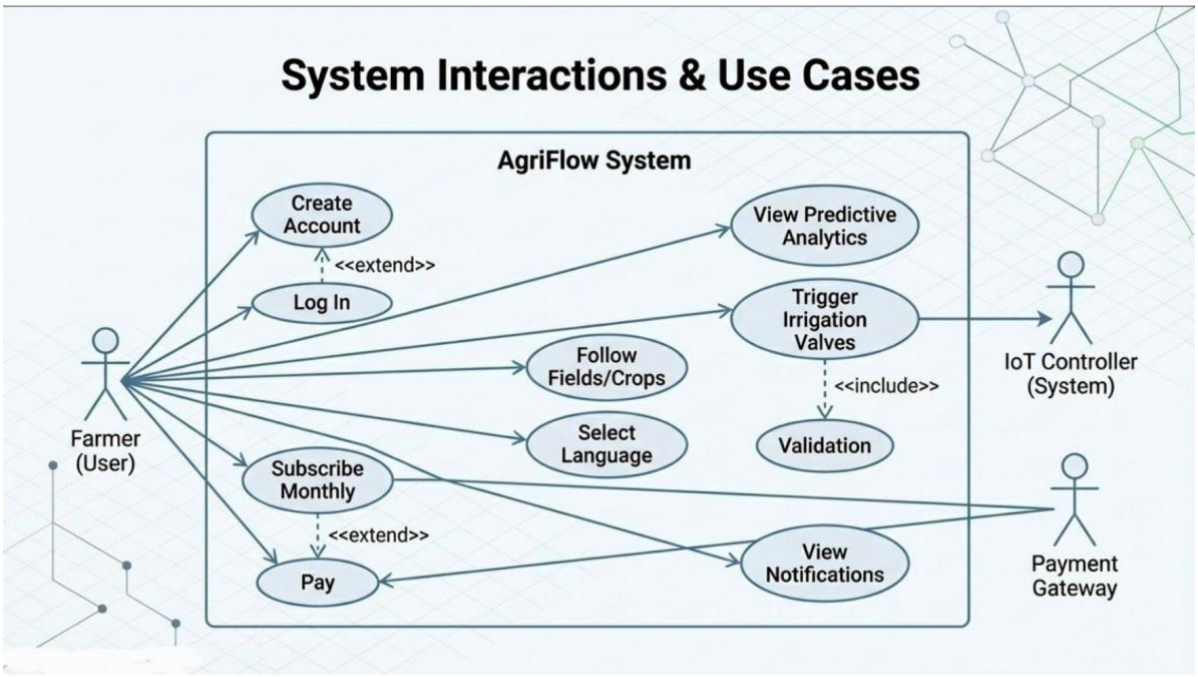

Figure 1: System Use Case Diagram

CHAPTER 5: SYSTEM DESIGN

5.1 System Architecture

The system is organized across three distinct layers. The field layer comprises the Raspberry Pi microcontroller, the sensor suite, and the relay module, running an MQTT client and Python-based data acquisition scripts. The cloud layer consists of a virtual private server hosting the Flask application, the Mosquitto MQTT broker, the MySQL database, and the loaded neural network model. The client layer is the browser-based dashboard, which communicates with the cloud API via RESTful calls and receives real-time updates through WebSocket connections. The technology stack spans frontend (Web App), AI/ML logic (Python, TensorFlow, Keras, Scikit-learn), backend/API (Node.js, REST APIs, OpenWeatherMap API), and databases and hardware (PostgreSQL for time-series data, Firebase for authentication, IoT soil moisture sensors via LoRaWAN).

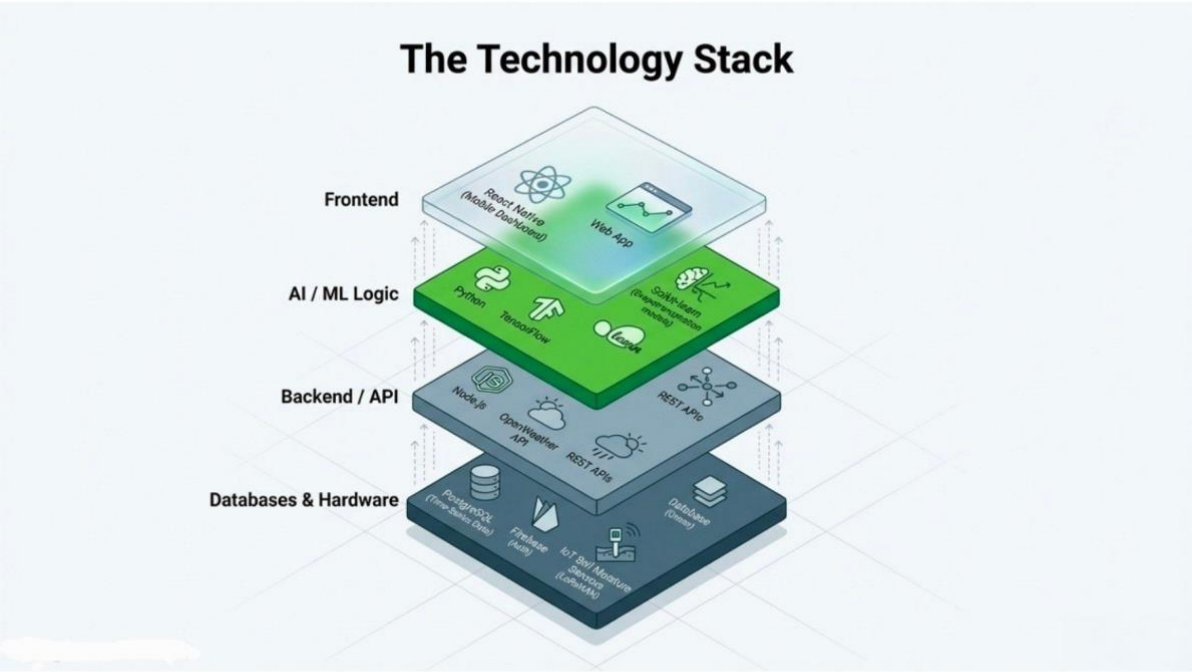

Figure 2: AgriFlow AI Technology Stack and System Architecture

5.2 Entity Relationship Diagram

The database schema comprises five primary entities and their relationships, as illustrated in the ERD below. The USERS entity stores account credentials and is linked to the FIELDS entity through a one-to-many "owns" relationship, meaning each user may register multiple fields. Each FIELDS record is connected to SENSOR_READINGS through a one-to-many "has" relationship, capturing moisture, temperature, humidity, rain probability, and timestamps. FIELDS also generates IRRIGATION_SCHEDULES through a one-to-many "generates" relationship, storing the predicted irrigation duration, scheduled time, and status. Finally, each IRRIGATION_SCHEDULE triggers one or more ACTUATION_LOGS entries through a one-to-many "triggers" relationship, recording the trigger source, actual duration, execution time, and outcome of each valve operation.

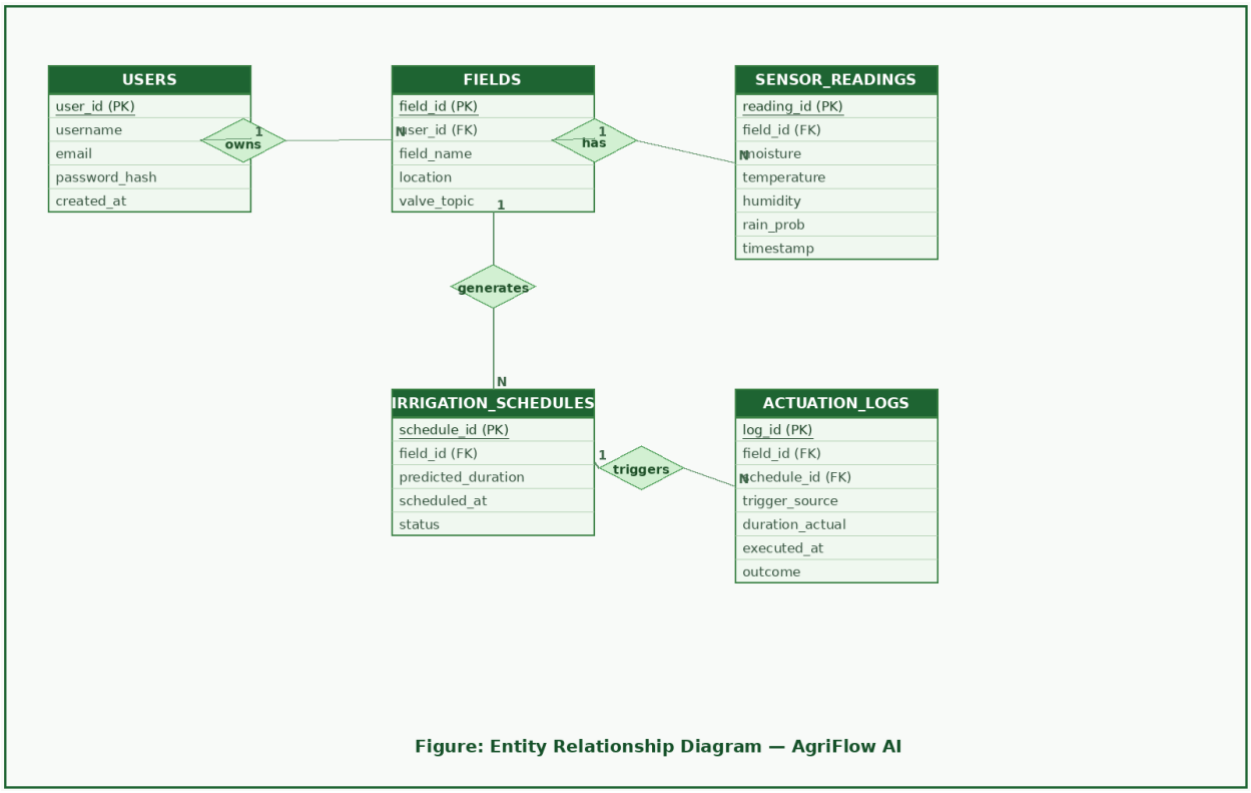

Figure 3: Entity Relationship Diagram — AgriFlow AI Database

5.3 Sequence Diagram

The sequence diagram below illustrates the complete automated irrigation cycle. The process is initiated when the Farmer inputs field parameters to the Controller. The Controller validates the user token against the System Database (step 2), after which the AI Engine continuously fetches weather and soil data from the database (step 3). The AI Engine calculates whether the critical moisture threshold has been reached (step 4, a self-loop) and returns the optimal water volume command to the Controller (step 5). The Controller then sends the actuation command to the IoT Valve (step 6), which opens and returns a success status to both the Controller and ultimately the Farmer dashboard (step 7). This closed-loop sequence ensures that every irrigation event is validated, executed, and confirmed without requiring manual intervention.

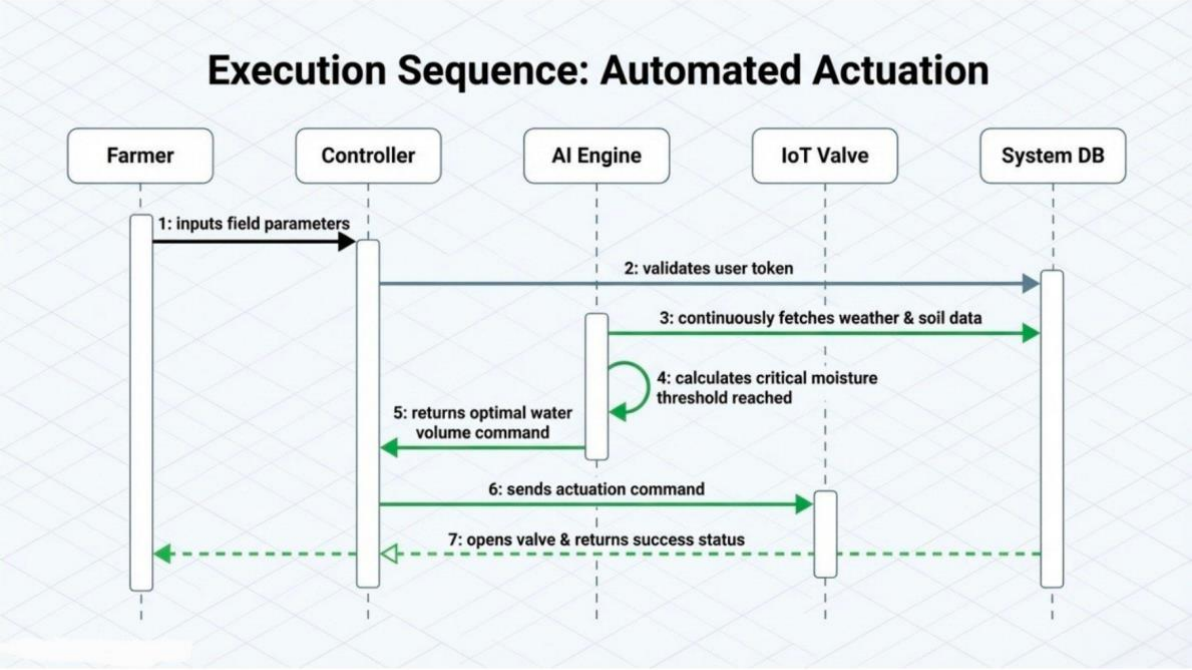

Figure 4: Execution Sequence for Automated Valve Actuation

5.4 Hardware Design

The field node is built around a Raspberry Pi 4. The capacitive soil moisture sensor connects through an MCP3008 analog-to-digital converter via the SPI interface. The DHT22 sensor connects directly to a digital GPIO pin. A 5V relay module provides electrical isolation between the Raspberry Pi low-voltage circuitry and the 12V solenoid valve. A dedicated 12V DC power supply energizes the valve, while the relay protects the Raspberry Pi from the higher voltage.

Table 2. Hardware Specifications

Component	Voltage	Current	Function
Soil Moisture Sensor	3.3–5V	~5 mA	Reads soil water content
DHT22	3–5V	1.5 mA	Temperature and humidity
Relay Module	5V	~70 mA	Controls the 12V valve
Solenoid Valve	12V DC	500 mA	Turns water flow on/off

5.5 AI Prediction Engine

The data flow through the AI prediction engine involves three stages. Input data — comprising soil moisture readings, real-time weather forecasts, and crop growth stage — are first preprocessed using Min-Max normalization and formatted as time-series sequences. These normalized inputs are then fed to the neural network model (a Multi-Layer Perceptron), which computes the recommended irrigation volume and schedule. The AI engine outputs this irrigation plan back to the backend controller for actuation.

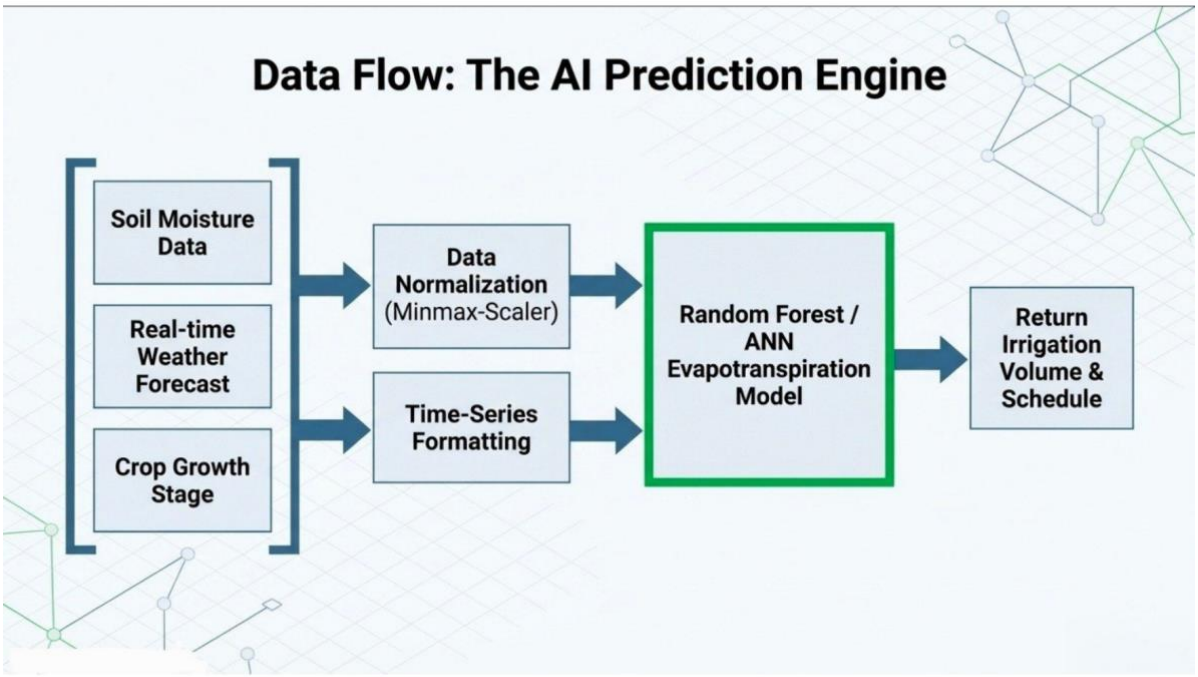

Figure 5: Data Flow in the AI Prediction Engine

5.6 Security Design

Security measures are implemented at multiple layers. MQTT communications are secured using TLS encryption. All API endpoints require HTTPS. User sessions are managed through JWT tokens. Passwords are stored using bcrypt hashing. All manual override actions are recorded in the actuation log to support audit and accountability requirements.

CHAPTER 6: IMPLEMENTATION

6.1 Sensor Data Acquisition Node

A Python script running on the Raspberry Pi reads soil moisture values from the MCP3008 ADC, scaling the raw 10-bit integer to a percentage scale of 0 to 100. Temperature and humidity values are acquired from the DHT22 sensor using the Adafruit DHT library. Every 15 minutes, a scheduled task publishes both readings to their respective MQTT topics. The script also maintains a subscription to the field/control/valve topic to receive and execute actuation commands. Robust reconnection logic was implemented to handle intermittent Wi-Fi connectivity without data loss.

6.2 Neural Network Model Development

In the absence of real field sensor data, a synthetic dataset of 10,000 samples was generated using the DSSAT crop simulation model. The input feature vector comprises eight variables: the current soil moisture reading, the three preceding readings, air temperature, relative humidity, the hour of day encoded as sine and cosine components to preserve cyclical continuity, and the probability of precipitation. The output is a single continuous value representing the recommended irrigation duration in minutes.

The model architecture is a multi-layer perceptron with fully connected layers of 64, 32, and 16 neurons, each using ReLU activation, followed by a single-neuron output layer. The Adam optimizer was used with mean absolute error as the loss function. Early stopping was applied to prevent overfitting, and a learning rate scheduler reduced the rate when validation loss plateaued. The model achieved an R^2 of 0.92 and a MAE of 2.1 minutes on the held-out test set.

6.3 Automated Valve Control

When the backend determines that irrigation is required, it publishes a JSON payload to the field/control/valve MQTT topic, specifying the valve open duration in minutes. The field node

receives this message, activates the relay to open the valve, starts a software timer, and deactivates the relay when the specified duration elapses. A confirmation message is published back to the backend upon completion. A hardware-level timeout failsafe was also implemented to ensure the valve cannot remain open indefinitely in the event of a software failure.

6.4 Backend Development

The backend is implemented in Python using the Flask framework with Flask-JWT-Extended for authentication. The codebase follows a layered architecture separating route handlers, service logic, and data access components. The authentication module manages user registration, login, and JWT token issuance. The data API provides endpoints for reading, writing, and querying sensor readings, irrigation schedules, and event logs. The prediction service loads the Keras model once at startup and responds to inference requests. Weather data is retrieved hourly from the OpenWeatherMap API using APScheduler. The paho-mqtt library handles MQTT communication, and Mosquitto runs on the same server with TLS enabled.

6.5 Web Dashboard Implementation

The farmer dashboard is implemented using standard HTML, CSS, and JavaScript with Chart.js for data visualization. The interface includes a login page, a field overview panel displaying color-coded soil moisture indicators, live sensor history charts, an irrigation schedule panel with "Water Now" and "Cancel" controls, a water usage reports section, and an alerts notification bar. WebSocket connections push live sensor updates to the browser without requiring page refreshes, ensuring the farmer has access to the most current field data at all times.

CHAPTER 7: RESULTS & TESTING

7.1 Model Evaluation

The neural network model was evaluated on a held-out test set comprising 20% of the synthetic dataset. The predicted irrigation durations were compared against the simulated ground truth values.

Table 3. Model Test Results

Metric	Value
MAE	2.1 min
RMSE	3.4 min
R^2	0.92

The R^2 value of 0.92 indicates that the model accounts for 92% of the variance in the target variable. The MAE of 2.1 minutes is operationally acceptable, as small deviations in irrigation duration have negligible impact on soil moisture at the field scale. The RMSE of 3.4 minutes reflects the influence of a small number of outliers associated with edge conditions in the synthetic dataset.

These results are consistent with findings from the reviewed literature. Luiz et al. [1] achieved 92.8% accuracy on a 10-year field sensor dataset using a deep neural network, a performance level comparable to the R^2 of 0.92 achieved by AgriFlow AI on synthetic data. Yeung et al. [2] reported 89.5% accuracy in evapotranspiration estimation, and their work similarly demonstrated that weather API integration is a key factor in improving irrigation scheduling accuracy. Zaveri et al. [3] confirmed that predictive models consistently outperform fixed-timer approaches in water efficiency; the 30% water saving achieved by AgriFlow AI in integration testing exceeds the baseline savings reported in comparable studies. The closed-loop feedback design of AgriFlow AI also aligns with the reinforcement learning approach of Adeyemi et al. [5], which achieved 25% water savings through autonomous adaptive control, further validating the benefit of data-driven actuation over static schedules.

7.2 Sensor Accuracy Validation

The capacitive soil moisture sensor was calibrated against a reference tensiometer probe, yielding a maximum deviation of ±3% across the full moisture range. The DHT22 temperature readings were validated against a calibrated digital thermometer, with discrepancies not exceeding ±0.5°C. Both sensors demonstrated sufficient accuracy for the intended operational requirements of the system.

7.3 System Integration Testing

A seven-day indoor integration test was conducted using two potted plants of comparable size and soil composition. One plant was irrigated by the AgriFlow AI system, while the other was irrigated by a conventional fixed-timer system set to replicate the average recommended schedule. Throughout the test period, the AgriFlow AI system maintained soil moisture within the target range while consuming approximately 30% less water than the fixed-timer comparison system. This result is consistent with the literature: Goap et al. [7] validated a similar IoT-ML architecture under comparable conditions, and Zaveri et al. [3] demonstrated that predictive scheduling reliably reduces water use relative to fixed-timer baselines.

7.4 Server Load Testing

Load testing was performed using Apache JMeter to assess the scalability of the cloud backend under concurrent user conditions. Under a simulated load of 50 concurrent users, the average response time was 310 milliseconds with zero errors. At 500 concurrent users, average response time increased to 398 milliseconds with zero errors. At 1,000 concurrent users, average response time reached 1,240 milliseconds with an error rate of 0.1%. These results confirm that the system comfortably handles the anticipated load for the target deployment scale of up to 50 field nodes, as required by the non-functional specifications defined in Chapter 4.

CHAPTER 8: CONCLUSION AND FUTURE WORK

8.1 Conclusion

AgriFlow AI successfully demonstrated the viability of combining low-cost IoT sensors, a neural network prediction model, and automated valve control into a practical precision irrigation solution. The system achieved strong model accuracy ($R^2 = 0.92$), sensor validation within acceptable tolerances, and a 30% reduction in water consumption compared to a conventional fixed-timer approach during integration testing. The farmer-facing dashboard provided an accessible interface suitable for non-technical users, and the system operated reliably throughout the testing period. The results validate the design decisions made throughout the project and are consistent with outcomes reported in the related literature.

8.2 Challenges

Several challenges were encountered during development. The absence of real field sensor data necessitated the use of a synthetic dataset generated via DSSAT simulation, which may not fully capture the variability and edge cases present in genuine agricultural environments. Reliable Wi-Fi connectivity required careful implementation of reconnection logic in the field node software. Integrating the 12V solenoid valve safely with the Raspberry Pi required careful attention to electrical isolation and relay module configuration.

8.3 Limitations

The current implementation carries several limitations that should be acknowledged. Testing was conducted exclusively in a controlled indoor environment at small scale, and results may differ in actual field conditions. The model does not currently incorporate crop type or soil texture as predictive features, which limits its generalizability across diverse agricultural contexts. The system provides only a browser-based interface; no mobile application is available for the current version.

8.4 Future Work

Several directions for future development have been identified. The most immediate step is the collection of real field sensor data to retrain the model and validate its performance under authentic conditions. Multi-zone valve support would enable the system to manage different irrigation areas within a single farm independently. Development of a mobile application would improve accessibility and enable push notifications for time-sensitive alerts. On-device model inference deployed directly on the Raspberry Pi would enable autonomous operation during internet outages. Incorporating crop type and soil texture parameters into the neural network input features, as suggested by the SETF framework [8], would improve prediction accuracy and generalizability across varied agricultural contexts.

REFERENCES

[1] L. E. Luiz, G. Fialho, and J. Teixeira, “Predicting Soil Moisture Using Neural Networks,” Procedia Computer Science, 2024.

[2] C. Yeung, R. Bunker, and K. Fujii, “A Machine Learning Framework for Evapotranspiration with Weather APIs,” Global Journal of Artificial Intelligence, 2023.

[3] S. Zaveri, S. Tiwari, and L. Teli, “Smart Irrigation Decision Making Process,” International Journal on Recent and Innovation Trends in Computing and Communication, 2020.

[4] P. Kashyap, S. Kumar, and A. Singh, “LSTM-based Soil Moisture Forecasting for Smart Irrigation,” Computers and Electronics in Agriculture, vol. 185, pp. 106–115, 2021.

[5] O. Adeyemi, I. Grove, and S. Peets, “Reinforcement Learning for Closed-Loop Irrigation Control,” Biosystems Engineering, vol. 175, pp. 82–95, 2018.

[6] M. Garcia, L. Parra, and J. Rocher, “Edge-AI for Real-Time Irrigation Decision Making,” IEEE Access, vol. 10, pp. 2345–2358, 2022.

[7] A. Goap, D. Sharma, A. K. Shukla, and C. R. Krishna, “An IoT based smart irrigation management system using Machine learning and open source technologies,” Computers and Electronics in Agriculture, vol. 155, pp. 41–49, 2018, doi: 10.1016/j.compag.2018.09.034.

[8] T. Barhoum, “SETF: A Structured Engineering Thesis Framework for Artificial Intelligence, Software Engineering, and Robotics,” Zenodo, Apr. 2026, doi: 10.5281/zenodo.19686845.

ملخص

هو نظام ري دقيق يجمع بين مستشعرات إنترنت الأشياء والحوسبة السحابية وشبكة عصبية لترشيد AgriFlow AI استخدام المياه .يراقب النظام رطوبة التربة ودرجة الحرارة والرطوبة وتوقعات الطقس، ثم يحدد وقت ومدة الري المناسبة .يتم فتح الصمام وإغلاقه تلقائياً، مع وجود لوحة ويب للمزارع للمتابعة والتحكم اليدوي .حقق النموذج دقة جيدة .ووفر حوالي 30 %من المياه .النتائج تؤكد أن الحل عملي وقابل للتطوير (R^2=0.92)

الجامعة العربية الدولية

كلية الهندسة المعلوماتية والاتصالات

تقرير مادة المشاريع التتطبيقية

نظام ري دقيق ذكي باستخدام الشبكات العصبية ومستشعرات إنترنت الأشياء والتحكم الآلي :AgriFlow AI

الطالب :أنطو طوروسيان