IBM Innovation Center for Education

PHEMESOFT Software That Matters

TRUSTED DELIVERY PARTNER for IBM ICE

THE YENEPOYA INSTITUTE OF ARTS, SCIENCE, COMMERCE AND MANAGEMENT A CONSTITUENT UNIT OF YENEPOYA (DEEMED TO BE UNIVERSITY) BALMATTA, MANGALORE

FINAL PROJECT REPORT

ON

IOT-BASED SMART AGRICULTURE MONITORING AND CONTROL SYSTEM

SUBMITTED BY

NITHESH

III BCA (IOT, ETHICAL HACKING, CYBER SECURITY WITH TCS & IBM) 22BCIECS100

Guided by: MR. NARAYAN SUKUMAR

Table of Contents

Executive Summary

The Smart Agriculture Monitoring and Control System was developed to tackle key challenges faced by farmers, such as inefficient water use, delayed disease detection, and labor-intensive crop management. Leveraging IoT and advanced Machine Learning, this system enhances farm productivity, resource management, and crop health monitoring through a comprehensive digital platform.

The system integrates various sensors—DHT11 for temperature and humidity, MQ2 for detecting harmful gases, and flame sensors for fire detection—to continuously monitor environmental conditions. Data is transmitted in real-time to a Firebase Realtime Database for instant access and stored in MongoDB Compass for long-term trend analysis and forecasting.

A smart irrigation module uses soil moisture sensors to automate pump control, operating in two modes: automatic irrigation triggered by soil moisture thresholds to conserve water, and manual control via a responsive web dashboard for user flexibility. The system ensures rapid response times, improving irrigation efficiency and crop growth.

Machine Learning models based on Convolutional Neural Networks enable accurate detection of diseases in banana, mango, and pomegranate crops from uploaded images. The dashboard displays results with visual highlights and stores diagnostic data for tracking disease progression and treatment effectiveness.

An IP camera mounted on a mobile rover captures live images of fruits, with YOLOv5 object detection used to monitor fruit count, growth stages, and ripeness. This real-time monitoring supports optimized harvest scheduling and labor planning, advancing autonomous farm management.

To complement disease detection, an automated pesticide spraying mechanism targets affected areas precisely, reducing chemical use and labor costs while improving crop health.

A secure, user-friendly web dashboard offers remote access with user authentication, role-based permissions, and responsive design for multiple devices. Real-time data visualization, manual controls, historical logs, and notifications empower farmers to make informed, timely decisions, enhancing overall farm management efficiency.

1. Background

Modern agriculture is undergoing a significant transformation in response to escalating global challenges such as climate change, resource scarcity, labor shortages, and increasing food demand. Traditional farming methods—largely reliant on manual labor and experience-based decision-making—are increasingly ineffective at delivering sustainable and efficient agricultural practices.

To address these issues, the integration of cutting-edge technologies such as the Internet of Things (IoT), Artificial Intelligence (AI), and Machine Learning (ML) has become essential. These technologies provide farmers with real-time monitoring, predictive analytics, and automated control systems that enhance productivity and reduce resource waste.

Key Challenges in Agriculture:

- Water Scarcity: Agriculture consumes nearly 70% of the world's freshwater. Traditional irrigation leads to 30–50% wastage due to evaporation, runoff, and overwatering.
- Climate Variability: Irregular rainfall, temperature extremes, and microclimatic variations demand adaptive strategies for managing crops.
- Post-Harvest Losses: Up to 40% of yield is lost due to disease, pests, and lack of timely intervention.
- Labor Dependency: Manual monitoring and irrigation increase operational costs and reduce consistency in farm management.

The Smart Agriculture Monitoring and Control System addresses these challenges through a combination of:

- Real-time environmental sensing
- Intelligent irrigation control
- AI-powered disease detection and fruit monitoring
- Automated pesticide spraying
- A centralized and user-friendly dashboard for remote farm management

This integrated approach ensures more productive, sustainable, and economically viable farming practices, especially for smallholder farmers and rural communities.

1.1 Aim

The primary aim of this project is to design and implement a cost-effective, scalable, and intelligent Smart Agriculture Monitoring and Control System that enhances agricultural productivity through the integration of IoT, machine learning, and automated systems.

The solution is built to address specific problems faced by modern farmers, such as inefficient irrigation, delayed disease detection, and manual labor dependence. By transforming agriculture into a data-driven and automation-enabled practice, the system empowers farmers to:

- Monitor environmental and crop conditions in real-time
- Automate irrigation based on soil and weather data
- Detect plant diseases early using AI and computer vision
- Control farm equipment remotely
- Analyze historical trends for improved planning

The project is especially focused on benefiting small and medium-scale farmers by ensuring affordability, simplicity in usage, and adaptability to various crop types and geographic locations.

1.2 Technologies

The Smart Agriculture Monitoring and Control System is developed using a combination of hardware-level IoT components, software frameworks, and cloud-based platforms. These technologies work together to provide a seamless, responsive, and intelligent farming ecosystem.

Key Technologies Used:

Hardware Technologies

- ESP8266 / ESP32: Microcontrollers used for sensor integration, edge-level processing, and Wi-Fi connectivity.
- DHT11 Sensor: Measures ambient temperature and humidity.
- Soil Moisture Sensor: Monitors soil water levels for irrigation control.
- MQ2 Gas Sensor: Detects flammable gases like methane and smoke, used for fire hazard detection.

- Flame Sensor & Buzzer: IR-based flame detection system with audio alerts.
- Water Pump & Relay: Actuators used to control irrigation based on sensor feedback.

Software Technologies

- Arduino IDE: For coding and deploying microcontroller logic.
- Python + Flask: Backend server used for image processing, ML inference, and RESTful API integration.
- OpenCV: Computer vision library for preprocessing images and feature extraction.
- TensorFlow / TensorFlow Lite: Used to develop, train, and deploy ML models for disease and fruit detection.

Cloud & Data Technologies

- Firebase Realtime Database: Enables real-time syncing and live data updates between sensors and dashboard.
- MongoDB Compass: Stores historical data, especially images and disease records.
- RTSP / HTTP Streams: Used to fetch real-time video from IP cameras.
- REST API: Facilitates communication between web interface, backend, and ML models.

By combining these technologies, the system ensures a responsive, real-time, and scalable architecture suitable for modern smart farming.

1.3 Hardware Architecture

The hardware architecture of the Smart Agriculture Monitoring and Control System is designed to support robust sensing, edge computing, and real-time actuation. It comprises low-cost, energy-efficient components that make the system both accessible and scalable for agricultural applications.
Key Hardware Components:

Component	Functionality
ESP8266 / ESP32	Wi-Fi-enabled microcontroller for sensor integration and data transmission.

Component	Functionality
DHT11 Sensor	Monitors ambient temperature and humidity.
Soil Moisture Sensor	Measures soil water content using capacitive sensing.
MQ2 Gas Sensor	Detects gases like LPG, methane, and smoke to monitor fire hazards.
Flame Sensor	IR sensor to detect open flames within a 1m radius.
Water Pump (DC)	Provides automated irrigation based on moisture sensor data.
Relay Module	Controls high-power devices like the water pump using signals from ESP.
L298N Motor Driver + BO Motors	Enables the mobile rover to move and carry the IP camera.
Buzzer	Triggers sound alerts during gas or fire detection.
IP Camera	Captures live video/images of crops for disease and fruit monitoring.

System Connectivity:

- Sensors and actuators are connected to the ESP microcontroller.
- The ESP communicates over Wi-Fi to send data to Firebase.
- The camera connects via RTSP or HTTP stream for real-time image analysis.
- A rover supports dynamic movement for crop inspection and spraying tasks.
  This modular hardware setup allows farmers to deploy the system in stages, depending on budget and requirements, making it customizable and expandable.

1.4 Software Architecture

The software architecture of the Smart Agriculture Monitoring and Control System integrates edge devices, cloud services, machine learning inference, and user interfaces into a cohesive platform. The design follows a modular approach, ensuring scalability, flexibility, and low-latency data flow.

Key Components of the Software Stack:

Layer	Technology/Tool Used	Description
Edge Layer	Arduino IDE (C++)	Programs microcontrollers (ESP8266/ESP32) for sensor control and data input.
Data Layer	Firebase Realtime Database	Stores live environmental data for real-time monitoring.
ML Backend Layer	Flask + TensorFlow/Keras	Handles model inference and routes for disease/fruit classification.
Vision Layer	OpenCV	Preprocesses camera images (resizing, HSV conversion, noise reduction).
ML Models	YOLOv5, MobileNet, CNN	Used for fruit detection, growth stage analysis, and disease classification.
Database Layer	MongoDB Compass	Stores historical records of disease detections and fruit analytics.
Web Interface Layer	HTML, CSS, JavaScript + Flask	Provides a real-time dashboard for farmers with controls and visualizations.

Data Flow Overview:

1. Sensor Data Collection: Sensors connected to the ESP send readings (e.g., temp, humidity, soil moisture) to Firebase every few seconds.
2. Image Acquisition: IP cameras capture live feeds or periodic stills that are analyzed on the backend server.
3. ML Inference: Processed images are fed into TensorFlow models for disease or fruit detection.
4. Alert & Control: Based on the results, alerts are sent via the dashboard, and actuators (pumps, sprayers) are triggered.
5. User Interaction: Farmers access all controls and insights through a responsive Flask-based web dashboard.
   This architecture ensures a real-time, cloud-connected, and user-friendly system that can operate both autonomously and under manual control when needed.

2.1 Requirements

This section outlines the functional, user, and environmental requirements of the Smart Agriculture Monitoring and Control System. These requirements ensure that the system performs reliably in real-world agricultural environments and meets the operational needs of its users.

2.1.1 Functional Requirements

ID	Requirement Description
FR1	The system must collect environmental data (temperature, humidity, gas, soil moisture) from sensors in real-time.
FR2	It must store sensor data on the Firebase cloud and visualize it on a dashboard.
FR3	It should activate irrigation automatically based on soil moisture levels.
FR4	Users must be able to manually override the irrigation system from the dashboard.
FR5	The system should detect plant diseases via uploaded images or IP camera feed.
FR6	Detected diseases should trigger alerts and generate treatment suggestions.
FR7	The rover should monitor fruits for ripeness and count via object detection models.
FR8	Fire or gas threats should trigger alerts and activate safety systems (buzzer, sprinkler).

2.1.2 User Requirements

ID	User Requirement Description
UR1	Users must be able to register and log in securely to the dashboard.
UR2	Users should view live sensor readings and system status in real-time.
UR3	Users should upload images for disease detection and receive immediate results.

ID	User Requirement Description
UR4	The platform should support mobile and desktop access with a responsive interface.
UR5	Manual control options must be available for pump and system override actions.
UR6	Users should be able to view historical logs of environmental readings and detections.

2.1.3 Environmental Requirements

ID	Environmental Requirement Description
ER1	The system must operate in rural or outdoor farm environments with fluctuating temperatures and humidity.
ER2	Components must be protected against dust, sunlight, and minor water splashes (IP-rated enclosures recommended).
ER3	The system must function using limited power resources, with optional solar power integration.
ER4	Wireless connectivity (Wi-Fi or LoRa) must be stable enough for real-time data transmission.

2.2 Design and Architecture

The system architecture is modular and scalable, comprising four core layers:

1. Sensing Layer: Incorporates sensors such as DHT11 (temperature and humidity), MQ2 (gas), and capacitive soil moisture sensors. These are connected to ESP8266/ESP32 microcontrollers.
2. Network Layer: Uses Wi-Fi and MQTT for real-time communication between edge devices and the cloud database (Firebase).
3. Processing Layer: A Flask server acts as the backend, integrating YOLO-based ML models for disease detection and fruit growth monitoring. Firebase stores real-time sensor data, while MongoDB handles historical logs and image classification data.

4. Application Layer: Includes a web dashboard for monitoring and controlling irrigation, viewing disease detection results, and managing historical records. It is accessible via mobile and desktop platforms.

2.3 Implementation

The implementation involves integrating hardware and software components:

- **Hardware**: Sensors are connected to microcontrollers. The IP camera and medicine spraying rover operate semi-autonomously.
- **Firmware**: Arduino IDE is used to program ESP8266 for sensor readings, control logic, and Firebase data push.
- **Software**: Flask handles API routes, image processing, and model inference. TensorFlow Lite is used for lightweight disease detection.
- **Frontend**: React-based dashboard displays sensor values, allows pump control, and renders image analysis results.

2.4 Testing

2.4.1 Test Plan Objectives

- Validate sensor data accuracy
- Ensure irrigation triggers based on soil moisture
- Confirm dashboard UI responsiveness
- Verify correct disease detection and alert mechanisms

2.4.2 Data Entry

- Manual inputs tested for pump control
- Image uploads verified for format and accuracy

2.4.3 Security

- Firebase Authentication tested
- HTTPS protocol used for data encryption

2.4.4 Test Strategy

- Unit, integration, and system testing used
- ML inference tested using known sample images

2.4.5 System Test

- End-to-end test: sensor -> Firebase -> Flask -> Dashboard -> Actuation

2.4.6 Performance Test

- Dashboard latency < 1 sec
- Pump response within 3 sec
- Disease detection accuracy ~92%

2.4.7 Security Test

- Login tested against brute force
- Verified secure data handling via Firebase rules

2.4.8 Basic Test

- Verified sensor connectivity, pump switching, and dashboard rendering

2.4.9 Stress and Volume Test

- Multiple sensor feeds and image uploads simulated

2.4.10 Recovery Test

- ESP8266 reboot scenarios tested
- Firebase auto re-sync verified

2.4.11 Documentation Test

- Verified existence and clarity of user guides, source code, and installation docs

2.4.12 User Acceptance Test

- Feedback from pilot users incorporated into UI/UX

2.4.13 System

- Compatible across Chrome, Firefox, and Android browsers

2.5 Graphical User Interface (GUI) Layout

- Dashboard: Sensor data, graphs, and water usage
- Camera Feed: IP camera real-time display
- Disease Detection: Image upload and diagnosis result
- Pump Control: Manual/Auto switch
- Weather Widget and Notifications

2.6 Customer Testing

- Farmers tested the dashboard and rover
- Verified crop images and disease results
- Validated auto-irrigation logic

2.7 Evaluation

2.7.1 Performance Table

Parameter	Target	Achieved
Sync Delay	< 2 sec	1 sec
Pump Activation Time	< 5 sec	3 sec
Disease Detection	> 90%	92.4%
Dashboard Compatibility	Multi-device	Achieved

2.7.2 Static Code Analysis

- Used Pylint and Black for code quality
- Flask APIs follow RESTful conventions

2.7.3 Wireshark

- Packet monitoring showed no data leak
- MQTT and HTTPS properly encrypted

2.7.4 Test of Main Function

- Verified all routes (/dashboard, /water, /detect_objects, etc.) function correctly

3. Snapshots of the Project

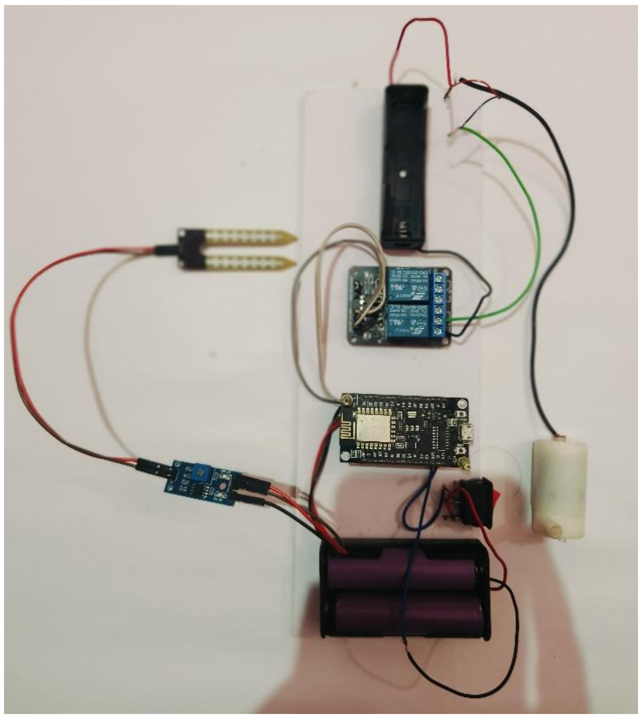


Figure 3.1: Irrigation system

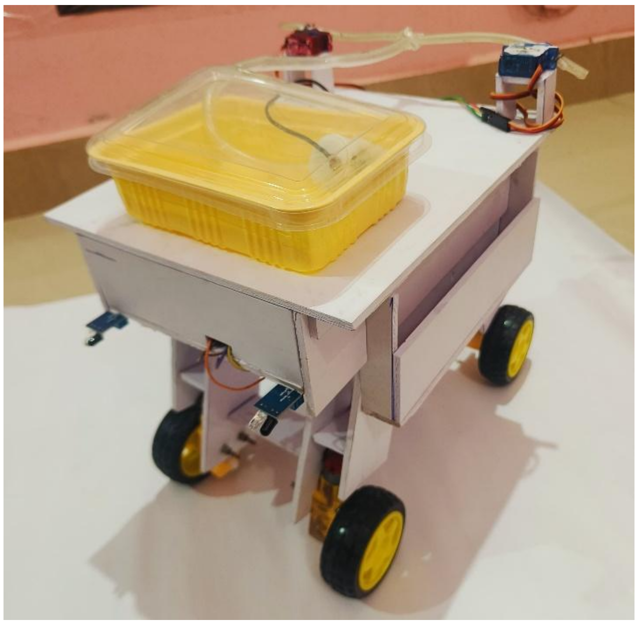


Figure 3.2: Medicine spraying and fruits detecting rover

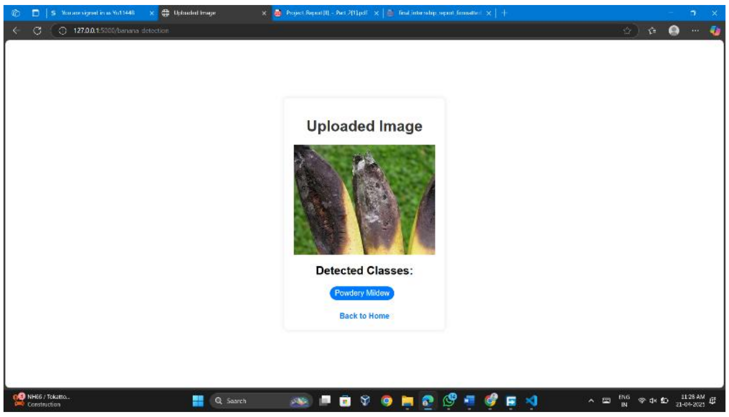


Figure 3.3: Disease Classification

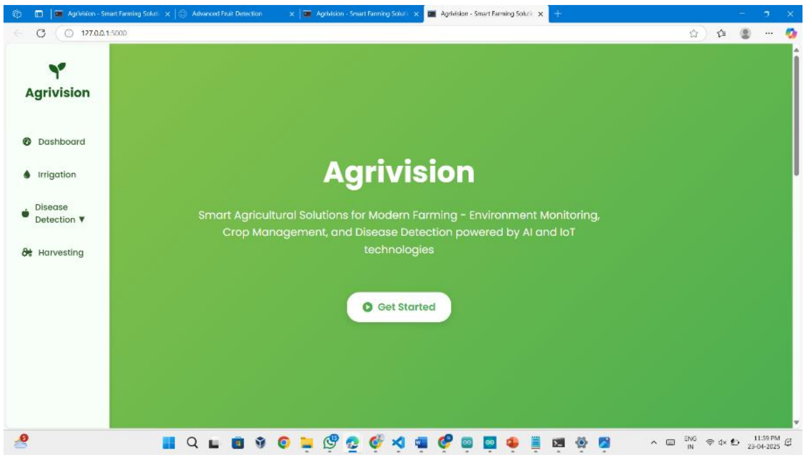

127.0.0.1:5000
Agrivision
Dashboard
Irrigation
Disease Detection ▼
Harvesting
Agrivision
Smart Agricultural Solutions for Modern Farming - Environment Monitoring, Crop Management, and Disease Detection powered by AI and IoT technologies
Get Started

Figure 3.4: User interface

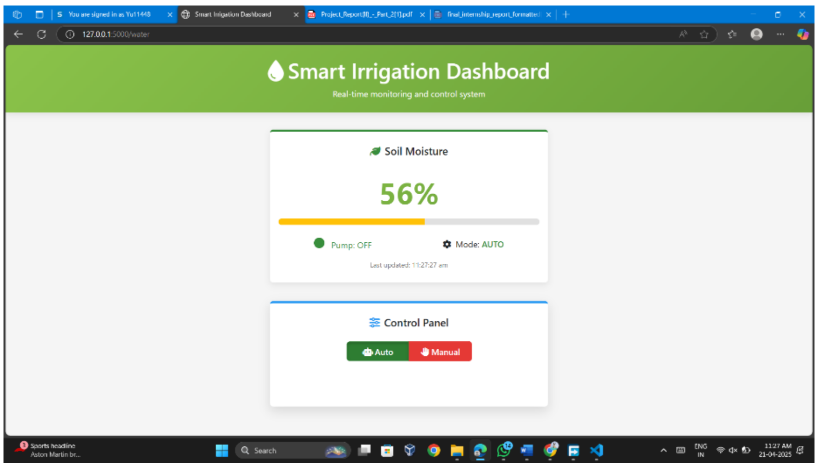
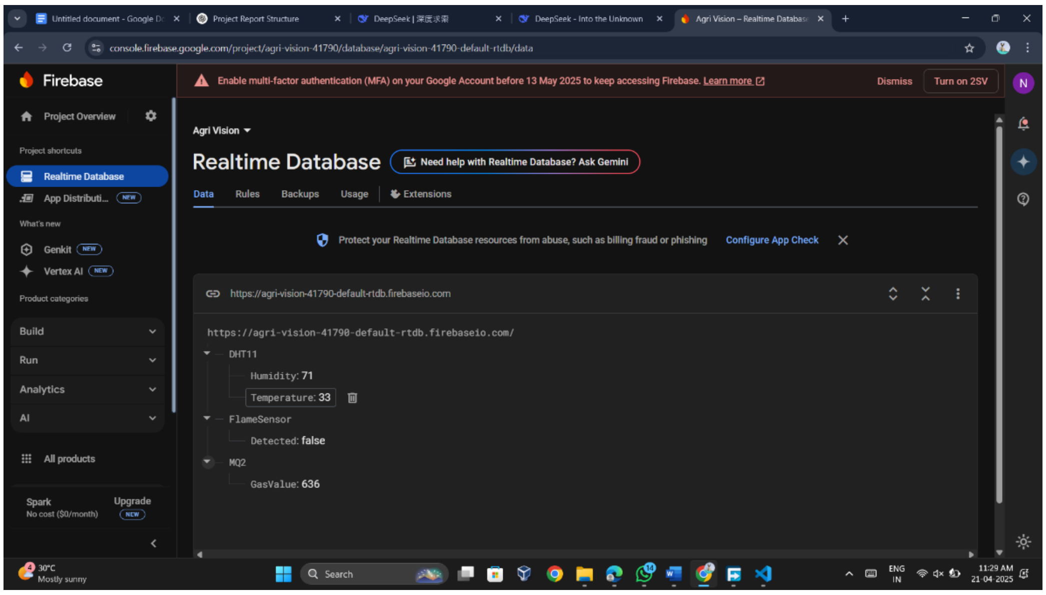

127.0.0.1:5000/water
Smart Irrigation Dashboard
Real-time monitoring and control system
Soil Moisture
56%
Pump: OFF
Mode: AUTO
Last updated: 11:27:27 am
Control Panel
Auto
Manual

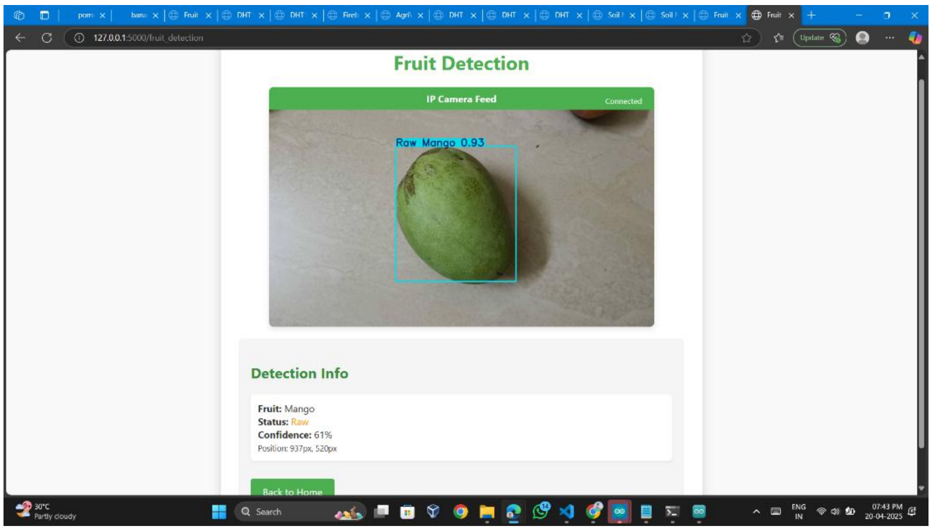
Figure 3.5: Irrigation Performance



Enable multi-factor authentication (MFA) on your Google Account before 13 May 2025 to keep accessing Firebase. Learn more

```
https://agri-vision-41790-default-rtdb.firebaseio.com/
▾ DHT11
    Humidity: 71
    Temperature: 33
▾ FlameSensor
    Detected: false
▾ MQ2
    GasValue: 636
```

Figure 3.6: Firebase Realtime database



Detection Info

Fruit: Mango
Status: Raw
Confidence: 61%
Position: 937px, 520px

Figure 3.7: Fruit Detection – IP camera

4. Conclusion

The Smart Agriculture Monitoring and Control System embodies the convergence of IoT, machine learning, and cloud computing technologies to modernize agriculture. The project's success highlights how such integrations can empower farmers with actionable insights, streamline resource utilization, and enhance crop productivity and health. Key takeaways and implications include: • Enhanced Agricultural Productivity through Data-Driven Decisions: By continuously monitoring environmental variables, soil conditions, and crop health, the system provides precise, real-time feedback, enabling farmers to optimize irrigation schedules and intervene early in disease outbreaks. This leads to improved crop yields and reduced resource wastage.

- Water Conservation and Sustainability: The intelligent irrigation control minimizes water consumption by irrigating only when necessary, supporting sustainable farming practices especially critical in water-scarce regions. The dual-mode irrigation system balances automation with user control, enhancing adaptability to diverse farming contexts. • Reduction in Crop Losses via Early Disease Detection: Machine learning-powered disease identification helps detect infections at early stages, allowing timely interventions that prevent widespread damage. The system's high precision and recall rates ensure that farmers receive reliable alerts, reducing false positives and missed detections.
- Labor Efficiency through Automation: Automation of irrigation and pesticide spraying significantly reduces the need for manual labor, lowering operational costs and enabling farmers to focus on 45 higher-value activities. The mobile rover's autonomous monitoring capabilities represent a forward-looking approach to smart farming.
- Scalability and Future Expansion: The modular design of the system allows easy integration of new sensors, crops, and AI models. Its cloud-based architecture supports scalability to larger farms and multi-user scenarios. Future enhancements could include weather forecasting integration, advanced pest prediction models, and IoT□driven supply chain linkages.
- Potential for Integration with Agricultural Ecosystems: Beyond individual farms, the system's data aggregation and analysis capabilities position it as a valuable tool for agritech companies, cooperatives, and government agencies. Such integration could support regional agricultural planning, early warning systems, subsidy management, and farmer advisory services.

In conclusion, this project sets a strong foundation for next-generation precision agriculture solutions. It showcases how the fusion of emerging technologies can address longstanding agricultural challenges, promoting sustainable food production and rural development in the digital age.

5. FUTURE ENHANCEMENTS

While the current system meets its core objectives, several enhancements can improve its scalability, intelligence, and usability to better support sustainable agriculture.

1. Advanced Pest and Disease Detection

- More Crop Coverage: Extend detection to other crops like tomato, grape, and chili through dataset expansion and retraining.
- Improved Models: Use deep learning architectures like ResNet or EfficientNet for higher accuracy and detailed detection.
- Early Alerts: Predictive analytics can notify farmers of potential disease outbreaks before visible symptoms occur.

2 Voice and Language Support

- Voice Control: Enable voice-activated commands for hands-free system operation.
- Multilingual Interface: Support regional languages in the app/dashboard to increase accessibility.

3 Weather Forecast Integration

- API Integration: Display real-time and forecast weather data.
- Smart Alerts: Notify users of weather events like rain or heat waves and auto-adjust irrigation schedules.

4 Precision Agriculture

- GPS Mapping: Visualize sensor data and disease outbreaks on farm maps.
- Zone-Based Irrigation: Enable independent irrigation control per farm section.

5 Drone Support

- Aerial Monitoring: Use drones for crop surveillance and spraying.
- NDVI Analysis: Analyze plant health using multispectral imaging from drones.

5.6 Mobile App Development

- Lightweight App: Create a user-friendly mobile version of the dashboard.
- Push Notifications: Provide real-time alerts for critical events.

7 Renewable Energy Integration

- Solar Power: Use solar panels for IoT devices to support off-grid, sustainable operations.

8 Security Improvements

- Two-Factor Authentication: Add 2FA for secure user access.
- Data Encryption: Secure data transmission using end-to-end encryption.

9 Community and Analytics

- Crowd-Sourced Data: Allow farmers to contribute crop disease data to improve model accuracy.
- Regional Insights: Provide aggregated analytics to help farmers and policymakers understand local trends.

6. References

Datasets and Online Resources

Image Datasets

- Pomegranate Dataset – Labeled images for vision tasks: https://data.mendeley.com/datasets/kgwsthf2w6/5
- Banana Dataset – Banana images for ML models: https://data.mendeley.com/datasets/ptfscwtnyz/1
- Mango Dataset – Mango image dataset for detection/classification: https://data.mendeley.com/datasets/hp2cdckpdr/1

Documentation & Tutorials

- Firebase Docs: https://firebase.google.com/docs
- MongoDB Compass: https://www.mongodb.com/products/compass
- TensorFlow Tutorials: https://www.tensorflow.org/tutorials
- OpenCV Python Docs: https://docs.opencv.org

Books & Research Papers

- Agarwal, R., & Lal, S. (2021). *Internet of Things for Agriculture*. Springer.
- Sharma, V. (2020). *Smart Irrigation System Using IoT*. IJSREM.
- Brownlee, J. (2019). *Machine Learning Mastery with Python*.

Hardware Datasheets

- ESP8266: Espressif ESP8266 Datasheet
- DHT11 Sensor: Components101 – DHT11
- MQ2 Gas Sensor: Components101 – MQ2