Smart Basil Greenhouse: An IoT-Based Automated Monitoring and Control System

Rafid Hasan
Bachelors in Computing Science
Thompson Rivers University
Kamloops, BC, Canada
rafidhasan202@gmail.com

Arshit Bansal
Bachelors in Computing Science
Thompson Rivers University
Kamloops, BC, Canada
T00729472@mytru.ca

Manpreet Singh
Bachelors in Computing Science
Thompson Rivers University
Kamloops, BC, Canada
T00729604@mytru.ca

Rajveet Singh
Bachelors in Computing Science
Thompson Rivers University
Kamloops, BC, Canada
T00729537@mytru.ca

Abstract*—This project presents an IoT-based smart greenhouse system designed to automatically monitor and regulate key environmental conditions for growing basil plants. The primary objective was to create a low-cost, real-time solution integrating sensors, cloud connectivity, automation logic, and a 3D digital twin. The system employs temperature, humidity, soil-moisture, light, and water-level sensors connected to an Arduino MKR1010, which communicates with Firebase Realtime Database. A custom web dashboard displays live sensor data, historical trends, plant health indicators, and manual/automatic control options for actuators including an exhaust fan, grow light, and water pump. A Unity-based 3D model mirrors real-time states as a digital twin for enhanced visualization and remote monitoring. Testing demonstrated reliable data streaming with response times of 200-500ms, functional automation thresholds, and stable real-time control through the dashboard. The system shows practical potential for small-scale farming, home gardening, and remote plant monitoring applications. Future improvements include adding horticultural LEDs, integrating advanced sensors for CO_2 and pH monitoring, and implementing AI-driven predictive analytics. This project demonstrates how IoT and cloud technologies can significantly simplify greenhouse management while improving plant growth conditions.

***Index Terms*—IoT, smart greenhouse, Arduino MKR1010, Firebase, digital twin, automation, precision agriculture, real-time monitoring.**

I. INTRODUCTION

The integration of Internet of Things (IoT) technologies in agriculture has experienced significant growth in recent years, driven by increasing demands for precision, efficiency, and remote monitoring capabilities [1]. Traditional greenhouse management requires continuous observation and manual adjustment of environmental parameters including temperature, humidity, soil moisture, and water levels factors that directly impact plant health and crop yield. For culinary herbs such as basil, maintaining a stable microenvironment is essential for yield quality, aroma development, and overall growth consistency [2]. However, manual supervision remains time consuming, prone to human error, and often inconsistent.

Recent advances in sensor technology, cloud computing, and microcontroller platforms have enabled the development of affordable smart agriculture systems that address these limitations [3]. The emergence of digital twin technology has further enhanced remote monitoring capabilities by creating virtual representations of physical agricultural systems [4].

This project addresses these challenges by developing a low-cost, fully connected IoT-based smart greenhouse system dedicated to basil cultivation. The system collects real-time environmental data using multiple sensors—DHT22, soil moisture probe, LDR, and ultrasonic water-level sensor—and communicates these readings to Firebase Realtime Database through an Arduino MKR1010. A web-based dashboard displays live readings, provides notifications, and enables users to control devices such as an exhaust fan, grow light, and water pump. A key innovation in this project is the integration of a Unity 3D digital twin, allowing users to visualize real-time changes inside a virtual greenhouse representation.

II. LITERATURE REVIEW

IoT-BASED smart agriculture has been extensively studied in recent years, with researchers developing increasingly sophisticated systems for greenhouse monitoring and control. Singh et al. [1] conducted a comprehensive study examining monitoring, control, and communication techniques in IoT-based greenhouse systems, highlighting the importance of integrating multiple sensor types for comprehensive environmental assessment. Their research emphasized that smart greenhouses typically comprise three key components working collaboratively: sensors for data collection, actuators for environmental control, and microcontrollers for processing and communication.

Song et al. [2] designed and implemented an intelligent monitoring system for agricultural environments using IoT, combining sensors, actuator, and cloud platform technologies. Their system demonstrated that environmental parameters such as temperature, humidity, and light can be effectively obtained through sensors and uploaded to cloud platforms for storage and analysis, enabling automatic control of greenhouse conditions.

The application of cloud platforms for IoT greenhouse systems has been investigated by multiple researchers. Zaguia [3] developed a smart greenhouse management system with cloud-based platform and IoT sensors, demonstrating the feasibility of real-time remote monitoring and control. Similarly, Kumar et al. [5] provided a comprehensive review of smart and sustainable agriculture using IoT technologies, emphasizing the importance of cloud connectivity for data synchronization and remote access.

Digital twin technology has emerged as a promising advancement in smart greenhouse applications. Peladarinos et al. [4] conducted a comprehensive review of digital twins in smart agriculture, noting that virtual replicas of physical

farms can simulate various scenarios, test different strategies, and predict outcomes accurately. More recently, Kumar and Islam [6] proposed a digital twin framework for smart greenhouse management using next-generation mobile networks and machine learning, demonstrating the integration of Unity 3D simulation environments with IoT monitoring systems.

This project builds upon existing work by: (1) integrating a 3D real-time digital twin developed in Unity, (2) implementing cloud synchronization with Firebase Realtime Database for low-latency data exchange, (3) conducting empirical threshold testing for environmental automation specific to basil cultivation, and (4) providing toast notifications and health scoring for enhanced user guidance.

III. METHODOLOGY

I. A. System Architecture

The greenhouse automation system consists of four interconnected layers designed to provide comprehensive monitoring and control capabilities. The Physical Layer encompasses all greenhouse components including sensors (DHT22 temperature/humidity sensor, soil-moisture probe, LDR light sensor, ultrasonic distance sensor) and actuators (exhaust fan, water pump, LED grow light placeholder). The Processing Layer utilizes an Arduino MKR1010 Wi-Fi-enabled microcontroller that manages sensor inputs and activates actuators, with automation rules running locally to ensure fast response times. The Cloud Layer employs Firebase Realtime Database for low-latency data synchronization between the physical system and user interfaces. The Interface Layer includes both a web dashboard developed with HTML/CSS/JavaScript and a Unity 3D digital twin for enhanced visualization.

II. B. Hardware Components

Table I summarizes the hardware components used in this project along with their purposes and relevant implementation notes.

TABLE I: Hardware Components Summary

Component	Purpose	Notes
DHT22	Temperature & humidity	±0.5°C, ±2% RH
Soil Moisture	Soil wetness detection	Calibrated trials
LDR	Light intensity	Grow light trigger
Ultrasonic	Water tank level	3–11 cm range
Arduino MKR1010	Central controller	WiFi + cryptochip
Exhaust Fan	Climate regulation	Humidity/tempcon-trol
Water Pump	Automated irrigation	3s safety limit
LED	Grow light	Digital twin sync

III. C. Software Workflow

The software workflow operates through a coordinated multi-layer communication loop between the Arduino, Firebase, the web dashboard, and the Unity digital twin. The Arduino MKR1010 continuously reads all sensor values at 2-second intervals and uploads the data to Firebase using REST API calls. These updates are written to predefined database paths corresponding to temperature, humidity, soil moisture, light intensity, water-level percentage, and actuator states.

The web dashboard—implemented using HTML, CSS, JavaScript, and Firebase Web SDK—subscribes to these database paths in real time. Each change triggers automatic updates to the dashboard's display cards, historical charts, toast notifications, and plant-health scoring system. Users may also issue manual commands from the dashboard (e.g., toggling fan, pump, or auto mode), which are written back to Firebase and subsequently read by the Arduino, enabling seamless interaction between physical hardware and software interface.

The Unity digital twin operates in parallel with the dashboard. Using Firebase's Unity SDK, the digital twin subscribes to the same database nodes, updating its 3D animations, materials, and UI overlays whenever sensor or actuator values change. This ensures that all three systems, the physical greenhouse, web dashboard, and Unity environment—remain perfectly synchronized, providing a unified and responsive IoT control ecosystem.

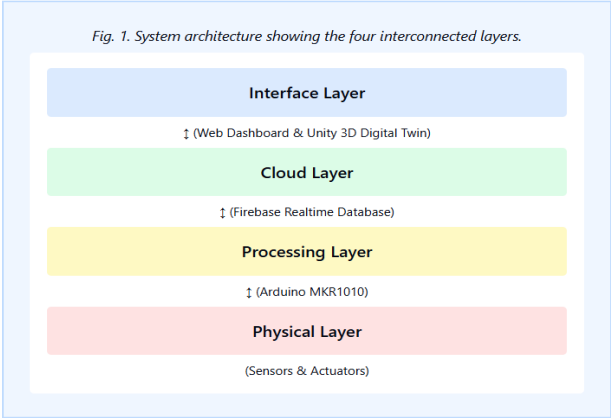


Fig. 1: Four-layer architecture showing Physical, Processing, Cloud, and Interface components of the smart greenhouse system.

D. Unity Integration

The Unity-based digital twin was integrated using the official Firebase Unity SDK, which enabled real-time two-way data synchronization using Firebase Realtime Database listeners. The Firebase database structure mirrored in Unity followed the same schema defined in the system's cloud architecture (e.g., temperature, humidity, soil moisture, water level, fan/pump states, and automation toggles), consistent with the hardware JSON layout used by both Arduino and the web dashboard
hardware
. This allowed the Unity scene to act as an exact virtual replica of the physical greenhouse.

The 3D environment itself was constructed using a combination of **custom Blender models** and **Unity Asset Store resources**. Blender was used to create custom meshes such as the soil bed, plant pot, and water reservoir to match the physical prototype's dimensions. Additional elements like environmental lighting, fan housing, and decorative greenhouse structures were imported from the Unity Asset Store to accelerate the modelling process. These were

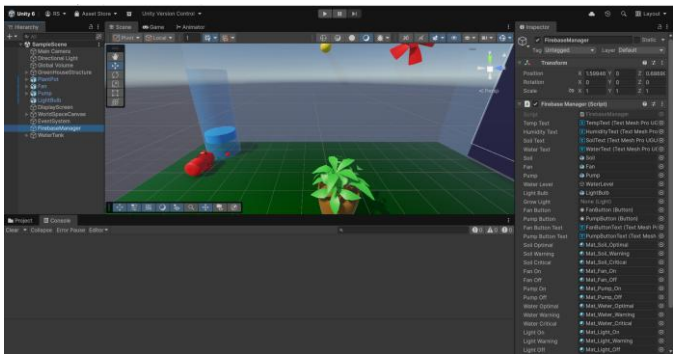
organized in a structured scene hierarchy and assigned object references to be controlled via scripts.

Once the scene was prepared, the FirebaseManager.cs script was added to a central controller object in the Unity hierarchy. Public variables in the script—such as 3D objects (fan, pump, soil, water cylinder, grow light), UI fields (temperature, humidity, soil moisture), and material sets—were exposed to the Unity Inspector and linked manually by dragging their corresponding scene objects. The script initializes Firebase, checks dependencies, then connects to database paths such as greenhouse/temperature, greenhouse/humidity, greenhouse/soil_moisture, greenhouse/light, and greenhouse/water_level_percent. Each of these database paths is continuously monitored, and whenever a value changes, dedicated update methods (e.g., UpdateWaterLevel(), UpdateFanVisual(), UpdateLightIntensity()) are triggered to adjust materials, animations, or UI text accordingly

FirebaseManager

.

Furthermore, actuator interactions were synchronized to match the system's automation logic. When the user manually toggles fan or pump controls within the Unity UI, the script writes updated values into database paths such as greenhouse/control/fan and greenhouse/control/pump. These write operations immediately propagate to the Arduino and to the web dashboard interface (implemented in index.html), ensuring consistent state synchronization across all platforms index. This unified structure allows the Unity digital twin to serve as both a live visualization tool and a functional control interface, accurately reflecting sensor data and actuator states of the physical greenhouse in real time.



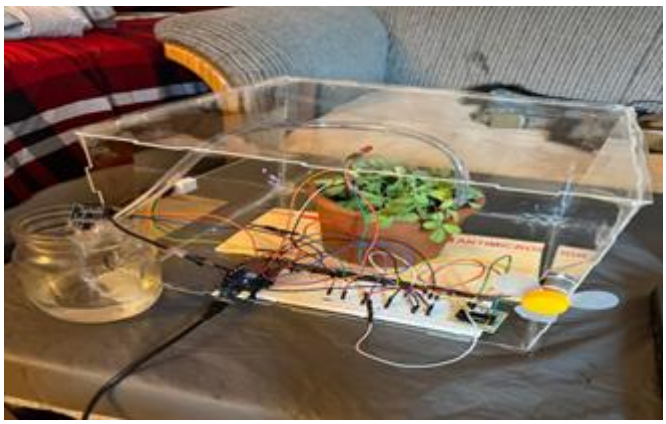


Fig. 2: Complete hardware assembly showing sensors, Arduino MKR1010, and actuators integrated in the greenhouse prototype.

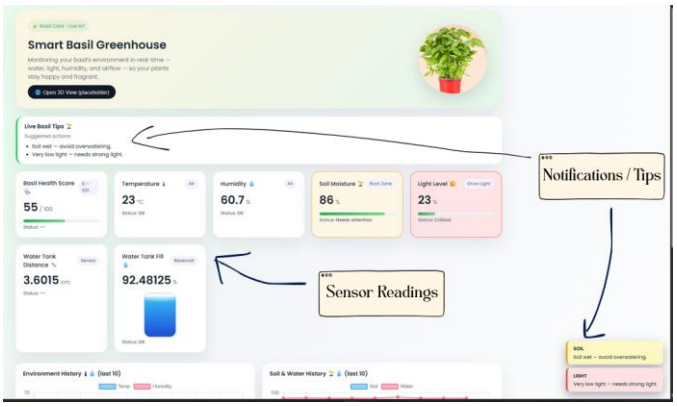


Fig. 3: Real-time web dashboard showing sensor readings, health score, control panel, and historical graphs.

IV. D. Threshold Testing Process

Multiple experiment rounds were conducted to determine optimal automation thresholds specific to basil cultivation requirements. Table II summarizes the tested conditions and final selected thresholds.

TABLE II: Threshold Testing Results

Parameter	Range Tested	Threshold
Temperature	27–32°C	Fan > 30°C
Humidity	60–80%	Fan > 70%
Soil Moisture	25–50%	Pump < 40%
Light Level	20–50%	LED < 40%
Water Level	10–30%	Disable < 20%

IV. RESULTS AND DISCUSSION

V. A. Real-Time Monitoring Performance

The web dashboard successfully reflected sensor changes within 200–500ms of Firebase updates, demonstrating low latency data synchronization suitable for real-time monitoring applications. Color-coded display cards clearly indicated status levels categorized as OK (green), Warning (yellow), and Critical (red), providing intuitive visual feedback for users monitoring greenhouse conditions.

VI. B. Automation Evaluation

Automation rules were thoroughly tested across multiple controlled scenarios by artificially manipulating temperature, humidity, soil moisture, light intensity, and water-tank levels. When humidity increased using a humidifier, the exhaust fan correctly triggered ON at the 70% threshold and automatically switched OFF once conditions returned to the safe range. Soil moisture behavior was validated by both drying and saturating the soil; when moisture rose above 40%, the pump correctly remained OFF, and when moisture dropped below the configured range, it activated for its 3-second safety interval before shutting down. Light-based automation was verified by blocking the LDR sensor—causing light readings to fall—

and the grow light correctly turned ON, while exposure to direct light reversed the action. Water-level automation was also validated: when the ultrasonic sensor detected tank levels below the 20% threshold, the system safely disabled pump operation to prevent dry-running damage. Together, these tests demonstrated consistent actuator responses, correct hysteresis behavior, and stable synchronization with both the web dashboard and the Unity digital twin, confirming the robustness and reliability of the threshold-based control logic.

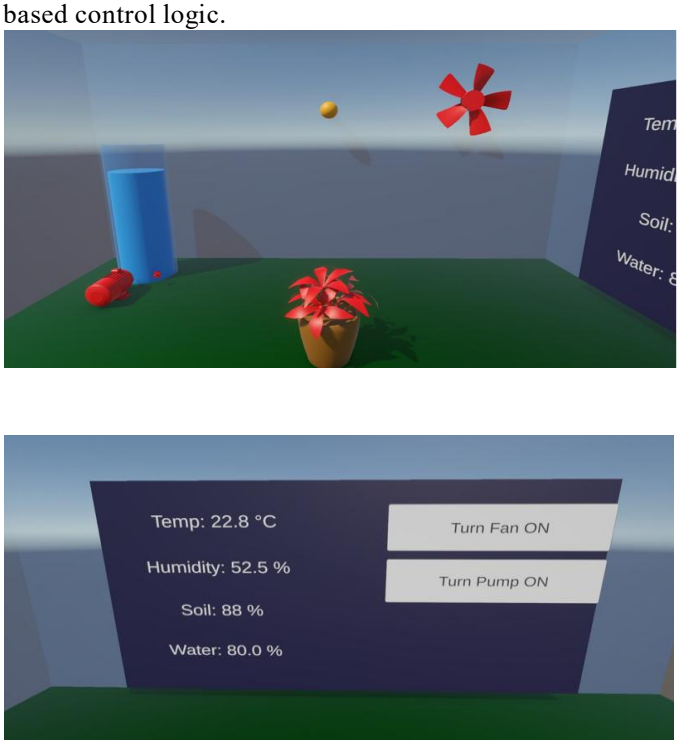


Fig. 4: Unity-based 3D digital twin showing real-time visualization of greenhouse environment and actuator states.

VII. C. Firebase Performance

Firebase Realtime Database provided stable and reliable real-time communication throughout the testing phase, with sensor updates typically reaching both the web dashboard and Unity digital twin within 200–500 ms. Data propagation remained smooth even during high-frequency streaming, with only minor delays observed when refresh intervals were pushed below one second—an expected limitation due to Firebase's built-in event throttling. The database successfully handled concurrent read/write operations from three independent sources: the Arduino MKR1010, the web dashboard (JavaScript listeners), and the Unity digital twin (C# listeners). Despite this simultaneous access, no data collisions, overwrites, or synchronization conflicts were observed. Firebase's event-driven architecture ensured that every change in actuator state or sensor reading instantly triggered updates across all clients, maintaining a consistent global state. The hierarchical structure of the database, combined with targeted listeners for specific paths (e.g., /greenhouse/fan_state, /greenhouse/light, /control/auto_mode), also minimized bandwidth usage and allowed each subsystem to receive only the data relevant to its function. Overall, Firebase proved to be a highly effective backbone for real-time IoT communication in the smart greenhouse system.

VIII. D. Digital Twin Synchronization

The Unity-based digital twin consistently mirrored the live state of the greenhouse environment, providing an accurate visual representation of both sensor readings and actuator behavior. Actuator states—including fan rotation speed, pump activation indicators, soil material changes, water-level scaling, and grow-light color transitions—updated seamlessly as values changed in Firebase. Synchronization delays were typically under one second, even during rapid sensor fluctuations, enabling true near–real-time interaction. The digital twin also responded correctly to manual user commands issued through the web dashboard or Unity interface, demonstrating full bidirectional communication. This ensured that any change in fan state, pump activation, or light level triggered immediate visual adjustments within the 3D scene. The use of Firebase listeners in Unity allowed each component of the virtual greenhouse to subscribe only to the specific database paths required for its behavior, reducing overhead and improving responsiveness. Overall, the digital twin provided an intuitive, immersive, and highly synchronized visualization layer that significantly enhanced remote monitoring, system testing, and user understanding of the greenhouse's real-time operations.

Challenges Encountered

Throughout the development of the smart greenhouse system, several technical challenges were encountered. Sensor calibration proved difficult, particularly due to noise fluctuations in the soil-moisture and LDR readings, which required repeated smoothing and threshold adjustments. Maintaining stable ultrasonic distance measurements was another issue, as small environmental changes or angle variations caused inconsistent tank-level readings. Power management also became a concern, especially when operating the pump and fan simultaneously, which placed noticeable load on the system and required careful control timing. A major non-technical challenge was learning Unity from scratch to build the 3D digital twin, as none of the team members had prior experience with 3D modeling or real-time data integration. Finally, handling Firebase's online/offline synchronization behavior required additional logic to ensure readings were uploaded reliably without data loss.

CONCLUSION AND RECOMMENDATIONS

The IoT-based Smart Basil Greenhouse system achieved all major objectives, including real-time monitoring, threshold-based automation, cloud integration via Firebase, web-based UI visualization, and creation of a functional 3D digital twin using Unity. The project demonstrated that affordable hardware combined with cloud services can significantly improve plant care efficiency and accessibility for small-scale greenhouse operations.

Future enhancements to the smart greenhouse system could significantly improve its efficiency, accuracy, and scalability. Integrating additional sensors for CO_2, pH, and nutrient-level monitoring would provide a more comprehensive understanding of plant health and soil conditions. The current placeholder LED should be replaced with a full-spectrum horticultural grow light to support actual plant growth requirements. Advanced features such as AI-driven prediction models could enable intelligent watering, lighting, and ventilation decisions based on historical patterns and machine learning algorithms. Long-term analytics dashboards would allow users to monitor trends over extended periods, improving decision-making and crop planning. Expanding the project into a mobile application would increase accessibility and enable remote control from any device. Finally, scaling the system to manage multiple greenhouses simultaneously would make it suitable for larger agricultural deployments.

ACKNOWLEDGMENT

The authors would like to express sincere gratitude to the course instructor for providing the necessary hardware and continuous guidance throughout the development of this project. We also acknowledge the support of online technical communities and academic resources that assisted in troubleshooting and refining the system. Special thanks to all team members for their collaboration in integrating sensors, actuators, cloud systems, and the Unity-based digital twin.

IX. REFERENCES

[1] N. Singh, A. K. Sharma, I. Sarkar, S. Prabhu, and N. Chadaga, "IoT-based greenhouse technologies for enhanced crop production: A comprehensive study of monitoring, control, and communication techniques," *Systems Science & Control Engineering*, vol. 12, no. 1, Art. no. 2306825, Jan. 2024.

[2] Y. Song, J. Bi, and X. Wang, "Design and implementation of intelligent monitoring system for agricultural environment in IoT," *Internet of Things*, vol. 25, Art. no. 101029, Mar. 2024.

[3] A. Zaguia, "Smart greenhouse management system with cloud-based platform and IoT sensors," *Spatial Information Research*, vol. 31, no. 5, pp. 559–571, Oct. 2023.

[4] N. Peladarinos, D. Piromalis, V. Cheimaras, E. Tserepas, R. A. Munteanu, and P. Papageorgas, "Enhancing smart agriculture by implementing digital twins: A comprehensive review," *Sensors*, vol. 23, no. 16, Art. no. 7128, Aug. 2023.

[5] V. Kumar, K. V. Sharma, N. Kedam, A. Patel, T. R. Kate, and U. Rathnayake, "A comprehensive review on smart and sustainable agriculture using IoT technologies," *Smart Agricultural Technology*, vol. 8, Art. no. 100487, Aug. 2024.

[6] R. Kumar and A. K. M. Najmul Islam, "Digital twin framework for smart greenhouse management using next-gen mobile networks and machine learning," *Future Generation Computer Systems*, vol. 156, pp. 191–205, Jul. 2024.

[7] K. M. Hosny, W. M. El-Hady, and F. M. Samy, "Technologies, protocols, and applications of Internet of Things in greenhouse farming: A survey of recent advances," *Internet of Things and Cyber-Physical Systems*, vol. 4, pp. 1–15, Apr. 2024.

[8] S. J. Soheli, N. Jahan, M. B. Hossain, A. Adhikary, A. R. Khan, and M. Wahiduzzaman, "Smart greenhouse monitoring system using Internet of Things and artificial intelligence," *Wireless Personal Communications*, vol. 124, no. 4, pp. 3603–3634, Jun. 2022.