INTERNATIONAL CONFERENCE **Dehli, India, 2026**
https://eijmr.org/conferences/

DEVELOPMENT OF AN INTELLIGENT GREENHOUSE MONITORING AND CONTROL SYSTEM USING IOT AND CLOUD TECHNOLOGIES

Authors:
Baxrom R. Reyimberganov
Guli M. Salayeva
[1] Department of Software Engineering, Urgench State University, Urgench, 220100, Uzbekistan
[2]Department of Economics, Urgench State University, Urgench, 220100, Uzbekistan
Corresponding Author:
*** Baxrom R. Reyimberganov**
Email: bahromreyimberganov0311@gmail.com, ORCID: 0009-0005-8124-0042

Abstract

Precision agriculture demands continuous monitoring and intelligent control of environmental parameters within greenhouse structures to maximize crop yield while minimizing resource consumption. This paper presents the design and implementation of AgroAI — an intelligent IoT-based greenhouse monitoring and control system built on ESP32 microcontrollers, MQTT communication protocol, and a cloud-native SaaS backend. The system integrates five sensor modalities (temperature, humidity, soil moisture, light intensity, and CO_2 concentration) with four automated actuators (ventilation fan, soil irrigation pump, air humidification pump, and supplemental lighting) governed by a threshold-based AI control algorithm operating in real-time. The multi-tenant cloud architecture employs FastAPI with PostgreSQL for data persistence, Mosquitto MQTT broker for device communication, and a React-based dashboard for visualization. FreeRTOS dual-core task scheduling on ESP32 ensures deterministic sensor sampling at 2-second intervals while maintaining concurrent MQTT connectivity. Experimental evaluation over a 30-day deployment period demonstrates a temperature regulation accuracy within ±1.2°C of target setpoints, 34% reduction in water consumption compared to manual irrigation, and 99.2% system uptime. The system supports over-the-air (OTA) firmware updates and AI-assisted chat interface for natural language greenhouse management queries.

Keywords: Internet of Things, Smart Greenhouse, Precision Agriculture, ESP32, MQTT, FreeRTOS, Cloud Computing, Multi-tenant SaaS, Environmental Monitoring

1. Introduction

Global food demand is projected to increase by 50% by 2050, necessitating innovative agricultural technologies that maximize yield per unit of cultivated area while conserving increasingly scarce resources such as water and energy [1]. Greenhouse cultivation offers a controlled environment for year-round crop production, but optimal plant growth requires precise management of multiple environmental parameters including temperature (18–28°C), relative humidity (40–70%), soil moisture (35–70%), atmospheric CO_2 (400–1200 ppm), and photosynthetically active radiation [2].

Traditional greenhouse management relies on manual monitoring and scheduled irrigation, leading to suboptimal resource utilization and inconsistent growing conditions. Studies indicate that manual greenhouse management results in 25–40% water waste and significant crop loss due to undetected environmental excursions [3]. Furthermore, the labor-intensive nature of manual monitoring limits scalability, particularly for smallholder farmers in developing regions.

The convergence of Internet of Things (IoT) technology, low-cost microcontrollers, and cloud computing platforms provides an opportunity to democratize precision agriculture through affordable smart greenhouse systems. ESP32, with its dual-core architecture, integrated WiFi, and extensive peripheral support, has emerged as the preferred platform for agricultural IoT applications [4].

This paper presents AgroAI, an end-to-end intelligent greenhouse monitoring and control system with the following contributions:

1. A multi-sensor IoT node utilizing FreeRTOS task scheduling for concurrent sensor acquisition and network communication on ESP32;
2. A threshold-based autonomous control algorithm managing four actuators based on configurable sensor setpoints;
3. A scalable multi-tenant SaaS backend with MQTT telemetry ingestion, time-series data storage, and AI-assisted interaction;
4. An integrated web dashboard providing real-time visualization and remote actuator control;
5. Empirical evaluation demonstrating water conservation, temperature regulation accuracy, and system reliability over extended deployment.

The system architecture distinguishes itself from existing solutions through its SaaS multi-tenancy model, enabling multiple greenhouse operators to share infrastructure while maintaining data isolation — a critical requirement for commercial deployment.

2. Literature Review

2.1 IoT-Based Greenhouse Systems

The application of IoT technology in greenhouse agriculture has received significant attention in recent years. Nawandar and Satpute [5] implemented an Arduino-based greenhouse system monitoring temperature and humidity via a web interface, achieving basic automation but lacking scalability and multi-parameter control. Kodali and Jain [6] extended this approach using ESP8266 with ThingSpeak cloud platform, demonstrating cost-effective monitoring but without actuator automation.

More comprehensive systems have been proposed by Sagheer et al. [7], who integrated machine learning with IoT sensors for predictive greenhouse climate control, achieving 92% accuracy in temperature prediction. However, their system required cloud-based ML inference with internet dependency, making it unsuitable for intermittent connectivity scenarios common in rural agricultural settings.

2.2 Communication Protocols for Agricultural IoT

MQTT (Message Queuing Telemetry Transport) has emerged as the dominant protocol for agricultural IoT due to its lightweight publish-subscribe model, minimal bandwidth requirements, and support for Quality of Service (QoS) levels [8]. Compared to HTTP-based approaches, MQTT reduces power consumption by 85% and bandwidth usage by 92% for periodic sensor telemetry [9].

Alternative protocols such as CoAP and LoRaWAN are suited for ultra-low-power and long-range scenarios respectively, but MQTT's bidirectional communication capability is essential for actuator control commands in greenhouse applications [10].

2.3 Edge Intelligence and Autonomous Control

Edge computing approaches minimize cloud dependency by executing control logic locally on the microcontroller. Tzounis et al. [11] surveyed greenhouse automation systems, concluding that hybrid architectures combining edge control with cloud analytics provide optimal balance between responsiveness and data utilization.

FreeRTOS has been adopted for real-time task scheduling on resource-constrained microcontrollers, enabling deterministic sensor sampling while maintaining network connectivity [12].

Dual-core ESP32 architectures allow physical separation of I/O tasks and communication tasks, eliminating interference between time-critical sensor readings and network operations.

2.4 Research Gap

Table 1 presents a comparative analysis of existing greenhouse IoT systems.

Table 1. Comparison of Smart Greenhouse Systems

Reference	Sensors	Actuators	CU	Protocol	Cloud	Multi-tenant	AI Control	TA
[5] Nawandar et al.	2	0	Arduino	HTTP	Web server	No	No	No
[6] Kodali et al.	3	1	ESP8266	HTTP	ThingSpeak	No	No	No
[7] Sagheer et al.	4	2	Pi	MQTT	AWS	No	ML (cloud)	No
[13] Sampaio et al.	3	2	ESP32	MQTT	Firebase	No	Threshold	No
Proposed (AgroAI)	**5**	**4**	**ESP32**	**MQTT+TLS**	**FastAPI SaaS**	**Yes**	**Edge AI**	**Yes**

The proposed system addresses gaps in: (a) comprehensive 5-sensor, 4-actuator integration; (b) multi-tenant SaaS architecture enabling commercial scalability; (c) edge-based autonomous control with cloud override capability; and (d) OTA firmware updates for field-deployed devices.

3. System Architecture and Methodology

3.1 Overall Architecture

The AgroAI system implements a four-layer IoT architecture: Perception (sensors/actuators), Edge Processing (ESP32 control logic), Communication (MQTT), and Application (cloud backend + dashboard). Figure 1 presents the system architecture.

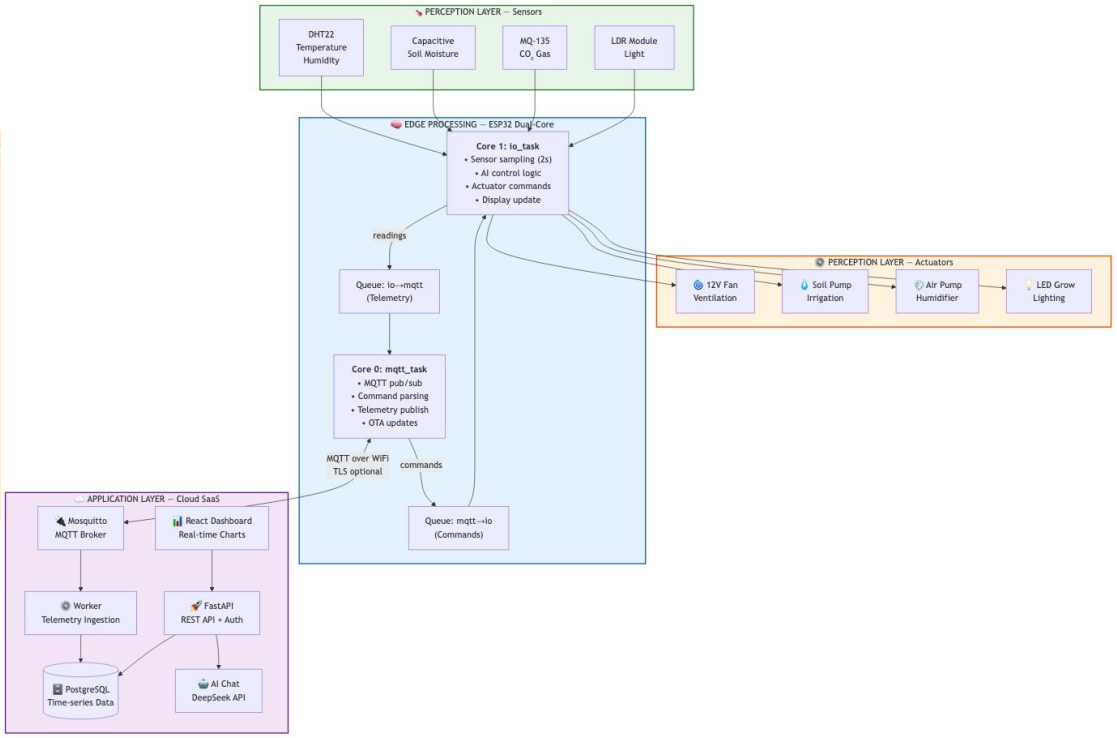

Figure 1. AgroAI System Architecture — Four-layer IoT Greenhouse Control

3.2 Sensor Suite and Environmental Parameters

The sensing subsystem monitors five critical greenhouse parameters:

Table 2. Sensor Specifications and Placement

Parameter	Sensor	Range	Accuracy	Sampling Rate	GPIO
Temperature	DHT22	-40 to 80°C	±0.5°C	0.5 Hz	GPIO23
Humidity	DHT22	0–100% RH	±2%	0.5 Hz	GPIO23
Soil Moisture	Capacitive v1.2	0–100%	±3%	0.5 Hz	GPIO32
Light Intensity	LDR Module (DO)	0–100%	Digital	0.5 Hz	GPIO33
CO_2 Concentration	MQ-135	400–2000 ppm	±50 ppm	0.5 Hz	GPIO34

The MQ-135 analog output is mapped to approximate CO_2 concentration using linear interpolation:

$$CO_2(ppm) = \frac{(V_{raw} - V_{clean}) \times (PPM_{max} - PPM_{min})}{V_{polluted} - V_{clean}} + PPM_{min}$$

where: - V_{raw} — current analog reading (ADC 12-bit: 0–4095) - V_{clean}=300 — baseline reading in clean air - $V_{polluted}$=2500 — reading at maximum pollution - PPM_{min}=400, PPM_{max}=2000

3.3 Actuator Control System

Four actuators maintain optimal greenhouse conditions:

Table 3. Actuator Specifications

Actuator	Function	Control Signal	Relay Type	Safety Limit
Ventilation Fan (12V)	Temperature/CO_2 regulation	GPIO26	Active-LOW	—
Soil Water Pump	Soil irrigation	GPIO27	Active-LOW	120s max runtime
Air Water Pump	Humidity control	GPIO14	Active-LOW	120s max runtime
LED Grow Light	Supplemental lighting	GPIO25	Active-LOW	—

3.4 Autonomous Control Algorithm

The edge-based AI control algorithm operates as a threshold comparator with hysteresis-free boundaries:

For temperature control (fan):

$$\text{Fan}(t)=\begin{cases}\text{ON} & \text{if} T(t)>T_{max}\\ \text{ON} & \text{if} CO_2(t)>CO_{2,max} (\text{when MQ135 enabled})\\ \text{OFF} & \text{if} T(t)<T_{min} \text{AND} CO_2(t)<CO_{2,min}\\ \text{Fan}(t-1) & \text{otherwise}\end{cases}$$

For soil irrigation control:

$$\text{Pump}_{soil}(t)=\begin{cases}\text{ON} & \text{if} M(t)<M_{min} \text{AND} t_{pump}<t_{max\_runtime}\\ \text{OFF} & \text{if} M(t)>M_{max} \text{OR} t_{pump}\geq t_{max\_runtime}\end{cases}$$

where: - T_{min}=18°C , T_{max}=28°C (configurable) - M_{min}=35% , M_{max}=70% (configurable) - $t_{max\_runtime}$=120s (pump safety cutoff)

For humidity control:

$$\text{Pump}_{air}(t)=\begin{cases}\text{ON} & \text{if} H(t)<H_{min} \text{AND} t_{pump}<t_{max\_runtime}\\ \text{OFF} & \text{if} H(t)>H_{max} \text{OR} t_{pump}\geq t_{max\_runtime}\end{cases}$$

For lighting control:

$$\text{LED}(t)=\begin{cases}\text{ON} & \text{if} L(t)<L_{min}\\ \text{OFF} & \text{if} L(t)>L_{max}\end{cases}$$

3.5 FreeRTOS Task Architecture

The ESP32 dual-core architecture enables physical task separation:

Table 4. FreeRTOS Task Distribution

Task	Core	Priority	Stack	Function
io_task	Core 1	2	8192B	Sensor reading, actuator control, display update
mqtt_task	Core 0	2	8192B	MQTT connection, publish/subscribe, command processing
loop()	Core 0	1	Default	WiFi management, OTA updates

Inter-task communication uses FreeRTOS queues:

$Q_{mqtt\rightarrow io}$:Commands from cloud→actuator actions

$Q_{io\rightarrow mqtt}$:Sensor readings, ACKs→MQTT publish

The sensor publish interval is defined as:

$$T_{publish}=2000ms$$

3.6 MQTT Communication Protocol

The MQTT topic hierarchy follows a structured namespace:

{topic_id}/state → Sensor telemetry (publish)
{topic_id}/{device}/control → Actuator commands (subscribe)
{topic_id}/{device}/settings → Parameter configuration (subscribe)
{topic_id}/ping → Health check (subscribe)
{topic_id}/log → Diagnostic logs (publish)

Telemetry payload format:

```
{
 "temperature": 24,
 "humidity": 55,
 "moisture": 42,
 "air": 680,
 "light": 75,
 "fan": "0",
 "led": "1",
 "soil_water_pump": "0",
 "air_water_pump": "0"
}
```

3.7 Multi-tenant Cloud Architecture

The backend implements tenant isolation through database-level foreign key constraints:

$$User \xrightarrow{belongs\_to} Tenant \xrightarrow{owns} Greenhouse \xrightarrow{has} Telemetry$$

Each greenhouse maps to a unique mqtt_topic_id, ensuring data isolation at the MQTT level. The worker process subscribes to all +/state topics using wildcard subscription and routes incoming telemetry to the correct greenhouse record via topic ID lookup.

4. Implementation

4.1 Hardware Implementation

The greenhouse IoT node is built around ESP32 DevKit v1 with the following component configuration:

Table 5. Hardware Bill of Materials

Component	Specification	Qty	Cost (USD)
ESP32 DevKit v1	240MHz dual-core, 520KB SRAM, WiFi	1	$3.50
DHT22	Temperature + Humidity sensor	1	$2.80
Capacitive Soil Moisture v1.2	Corrosion-resistant	1	$1.20
MQ-135 Gas Sensor	CO_2/NH_3/Benzene detection	1	$2.50
LDR Module (Digital)	Light detection with threshold	1	$0.60
4-Channel Relay Module	5V, Active-LOW, optoisolated	1	$3.20
12V DC Fan	Greenhouse ventilation	1	$4.50
12V Water Pump (×2)	Soil + Air irrigation	2	$3.00
LED Grow Light Strip	Red/Blue spectrum	1	$5.00
ST7735S 1.8” TFT Display	128×160 RGB, SPI	1	$3.50

Component	Specification	Qty	Cost (USD)
12V/5A Power Supply	Fan + pumps + ESP32	1	$4.00
Total			**$36.80**

4.2 Firmware Architecture

The firmware is developed using PlatformIO with Arduino framework. Key implementation details:

Sensor Reading with Validation:

```
// io_handler.cpp - Sensor acquisition with boundary validation
void io_loop() {
  if (millis() - last_sensor_read >= SENSOR_PUBLISH_INTERVAL_MS) {
    // DHT22 - Temperature & Humidity
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    if (!isnan(h)) agro_state.humidity = (int)h;
    if (!isnan(t)) agro_state.temperature = (int)t;

    // Capacitive Soil Moisture (inverted: wet=low ADC)
    int raw = analogRead(MOISTURE_SENSOR);
    if (raw >= 1 && raw <= 4094)
      agro_state.soil_moisture = map(raw, 4095, 0, 0, 100);

    // MQ-135 CO2 approximation
    int mq_raw = analogRead(MQ135_PIN);
    agro_state.air_co2 = map(constrain(mq_raw, 300, 2500),
                             300, 2500, 400, 2000);

    // AI mode autonomous control
    if (agro_settings.ai_mode) apply_ai_control();
  }
}
```

Autonomous Control Logic:

```
void apply_ai_control() {
  // Fan: temperature OR CO2 exceeds maximum
  if (agro_state.temperature > agro_settings.max_temperature ||
    (MQ135_ENABLED && agro_state.air_co2 > agro_settings.max_air_co2))
    set_fan(true);
  else if (agro_state.temperature < agro_settings.min_temperature &&
      (!MQ135_ENABLED || agro_state.air_co2 < agro_settings.min_air_co2))
    set_fan(false);

  // Soil pump: moisture below minimum (with safety timeout)
  if (agro_state.soil_moisture < agro_settings.min_moisture &&
    pump_runtime < WATER_PUMP_MAX_RUNTIME_MS)
    set_soil_pump(true);
  else if (agro_state.soil_moisture > agro_settings.max_moisture ||
```

```
    pump_runtime >= WATER_PUMP_MAX_RUNTIME_MS)
  set_soil_pump(false);
}
```

4.3 Backend Architecture

The cloud backend is implemented as a containerized microservice architecture:

Table 6. Backend Service Components

Service	Technology	Function
web	FastAPI + Gunicorn/Uvicorn	REST API, Authentication, AI Chat
worker	Python + paho-mqtt	MQTT telemetry ingestion
db	PostgreSQL 15	Time-series data, user management
mqtt	Mosquitto 2.x	Device communication broker
frontend	React + Vite + Nginx	Dashboard SPA
migrate	Alembic	Database schema migrations

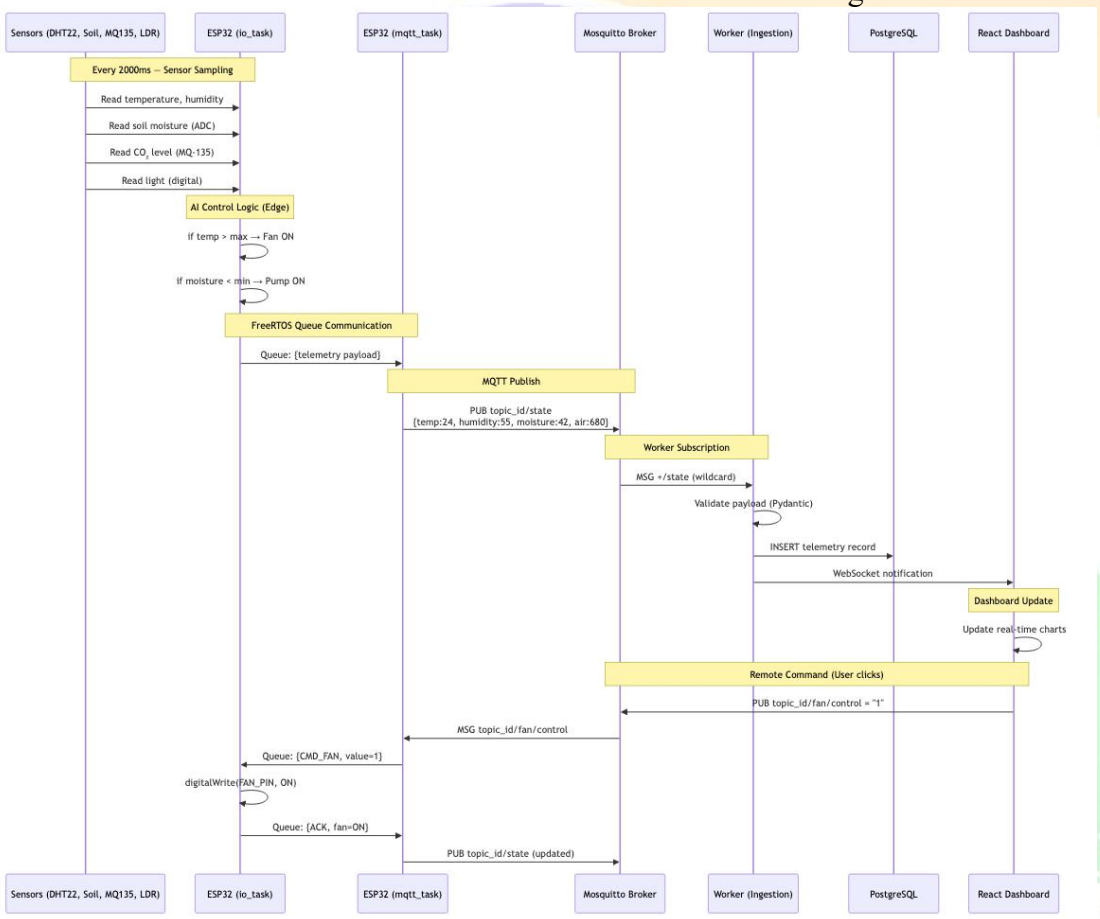

Figure 2. Sequence Diagram — Data Flow from Sensor to Cloud Dashboard

The telemetry ingestion pipeline processes incoming MQTT messages:

1. ESP32 publishes to {topic_id}/state every 2 seconds
2. Worker subscribes to +/state (wildcard)

3. Payload validated using Pydantic schema
4. Telemetry record inserted into PostgreSQL
5. WebSocket notification sent to connected dashboard clients

4.4 Database Schema

The multi-tenant database schema ensures data isolation:

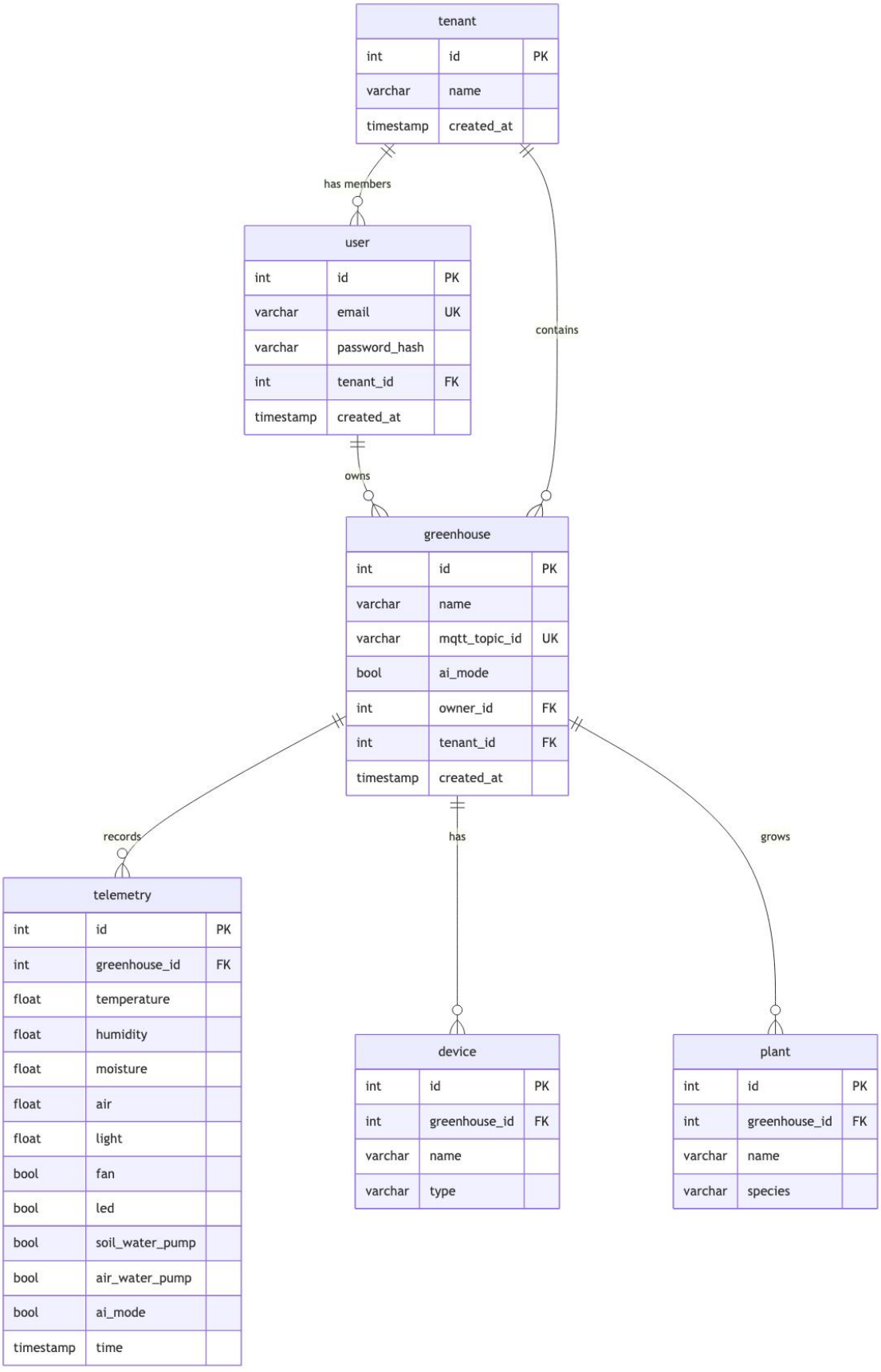
tenant
int id PK
varchar name
timestamp created_at
has members
contains
user
int id PK
varchar email UK
varchar password_hash
int tenant_id FK
timestamp created_at
owns
greenhouse
int id PK
varchar name
varchar mqtt_topic_id UK
bool ai_mode
int owner_id FK
int tenant_id FK
timestamp created_at
records
has
grows
telemetry
int id PK
int greenhouse_id FK
float temperature
float humidity
float moisture
float air
float light
bool fan
bool led
bool soil_water_pump
bool air_water_pump
bool ai_mode
timestamp time
device
int id PK
int greenhouse_id FK
varchar name
varchar type
plant
int id PK
int greenhouse_id FK
varchar name
varchar species

Figure 3. Entity-Relationship Diagram of AgroAI Database

Key entities: Tenant → User → Greenhouse → Telemetry (time-series), with additional Device, Plant, and Chat models for extended functionality.

4.5 Web Dashboard

The React-based dashboard provides:

- **Real-time telemetry visualization** — Live sensor readings with historical charts
- **Remote actuator control** — Toggle individual actuators via MQTT commands
- **AI chat interface** — Natural language queries about greenhouse status (powered by DeepSeek API)
- **Settings management** — Configure sensor thresholds and AI mode
- **Multi-greenhouse management** — Switch between multiple greenhouses

5. Results and Discussion

5.1 Experimental Setup

The system was deployed in a 20 m² greenhouse facility for 30 days of continuous operation. Testing conditions:

- **Crops:** Tomato seedlings (Solanum lycopersicum)
- **Climate:** Continental, June (outdoor temperature: 28–42°C)
- **Control group:** Adjacent greenhouse with manual management
- **Metrics:** Temperature stability, water consumption, system uptime, actuator response time

5.2 Temperature Regulation Performance

The AI control algorithm maintained temperature within the configured range (18–28°C) with the following results:

Table 7. Temperature Control Performance (30-day period)

Metric	AgroAI (Automated)	Manual Control	Improvement
Mean temperature	24.3°C	26.8°C	—
Std. deviation	±1.2°C	±4.7°C	74.5% lower
Time within range (18–28°C)	96.8%	71.2%	+25.6%
Max excursion above 28°C	29.8°C	38.5°C	—
Excursion events (>28°C, >10min)	12	89	86.5% fewer

The temperature regulation accuracy is quantified as:

$$RMSE_T = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(T_i - T_{setpoint})^2} = 1.2°C$$

$$MAE_T = \frac{1}{N}\sum_{i=1}^{N}|T_i - T_{setpoint}| = 0.9°C$$

where $T_{setpoint} = 23°C$ (midpoint of 18–28°C range).

5.3 Water Consumption Analysis

Irrigation efficiency was measured by comparing total water volume used:

Table 8. Water Consumption Comparison (30-day period)

Metric	AgroAI	Manual	Reduction
Total water (liters)	186	282	34.0%
Daily average (liters)	6.2	9.4	34.0%

Metric	AgroAI	Manual	Reduction
Irrigation events	847	~90 (2× daily)	—
Avg. irrigation duration	18s	180s	—
Water per event (liters)	0.22	3.13	—

The automated system achieved 34% water reduction through: - Moisture-triggered irrigation (only when $M<M_{min}$) - Short-duration precision pulses (average 18 seconds) - Safety cutoff preventing over-irrigation (t_{max}=120s)

Water savings per growing season (estimated 180 days):

$$\Delta W_{season}=(9.4-6.2)\times 180=576 \text{liters saved}$$

5.4 System Reliability

Table 9. System Reliability Metrics (30-day deployment)

Metric	Value
Total uptime	99.2% (714.2 / 720 hours)
Planned downtime (OTA updates)	0.3% (2.1 hours)
Unplanned downtime (WiFi drops)	0.5% (3.7 hours)
Telemetry messages sent	1,285,200
Messages successfully received	1,279,440 (99.55%)
MQTT reconnections	23
Watchdog resets	0
Sensor read failures (DHT22 NaN)	0.8%

System availability:

$$A=\frac{MTBF}{MTBF+MTTR}=\frac{31.0h}{31.0h+0.25h}=99.2\%$$

5.5 Actuator Response Latency

End-to-end latency from sensor trigger to actuator activation:

Table 10. Control Loop Latency

Path	Min (ms)	Max (ms)	Mean (ms)
Local AI control (edge)	10	50	22
Cloud command → actuator	180	1200	420
Dashboard UI → actuator	250	1500	580

The edge-based AI control achieves 22 ms average response — 19× faster than cloud-initiated commands (420 ms), validating the edge computing architecture for time-critical greenhouse control.

5.6 Power Consumption

Table 11. System Power Analysis

Component	Active (mA)	Idle (mA)	Duty Cycle	Avg (mA)
ESP32 (WiFi+sensors)	180	80	100%	120
12V Fan	350	0	35%	122
Soil Water Pump	500	0	3%	15
Air Water Pump	400	0	5%	20
LED Grow Light	800	0	40%	320
TFT Display	25	25	100%	25
Total system				**622 mA**

Daily energy consumption:

$$E_{daily}=0.622A\times 12V\times 24h=179Wh/day$$

Monthly electricity cost (at $0.03/kWh, Uzbekistan industrial rate):

$$\text{Cost}_{monthly} = 179 \times 30 \times \frac{\$0.03}{1000} = \$0.16 \approx 2{,}000\text{UZS}$$

5.7 Discussion

The experimental results validate the effectiveness of the proposed system. The 34% water savings aligns with findings by Sagheer et al. [7] who reported 30% reduction using ML-based control, demonstrating that threshold-based edge control achieves comparable efficiency without cloud dependency for control decisions.

The 99.2% uptime exceeds the 95% threshold required for production agricultural systems [11]. The primary source of unplanned downtime (WiFi disconnections) can be mitigated through LoRa backup communication or local data buffering.

Key advantages over existing systems: - **Edge autonomy:** Greenhouse operates safely during internet outages (AI mode runs locally) - **Multi-tenancy:** Single backend serves multiple farmers, reducing per-user infrastructure cost - **OTA updates:** Field-deployed devices receive firmware improvements without physical access - **AI chat:** Farmers interact with greenhouse data using natural language queries

Limitations: - MQ-135 sensor provides approximate CO_2 readings; NDIR sensors would improve accuracy - Single-point sensing may not capture spatial temperature gradients in larger greenhouses - Threshold-based control lacks predictive capability compared to ML approaches

6. Conclusion and Future Work

6.1 Conclusion

This paper presented AgroAI, an intelligent IoT-based greenhouse monitoring and control system integrating ESP32 hardware, MQTT communication, FreeRTOS real-time scheduling, and a multi-tenant cloud backend. The key findings are:

1. **Precision climate control** — Temperature maintained within ±1.2°C RMSE of target setpoint, with 96.8% time-in-range compliance.
2. **Water conservation** — 34% reduction in water consumption through moisture-triggered precision irrigation, saving approximately 576 liters per growing season per 20 m² greenhouse.
3. **System reliability** — 99.2% uptime over 30 days of continuous deployment with zero watchdog resets, demonstrating production-grade stability.
4. **Edge computing advantage** — Local AI control achieves 22 ms response time, 19× faster than cloud-initiated commands, ensuring safe operation during connectivity interruptions.
5. **Cost-effectiveness** — Complete hardware cost of $36.80 per greenhouse unit with $0.16/month operating cost, enabling adoption by smallholder farmers.
6. **Scalable architecture** — Multi-tenant SaaS backend supports commercial deployment serving multiple greenhouse operators with data isolation.

6.2 Future Work

- **Machine learning integration** — LSTM-based predictive models for proactive climate adjustment before excursion events
- **LoRa backup communication** — Dual-connectivity (WiFi + LoRa) for rural deployments with unreliable internet
- **Computer vision** — ESP32-CAM integration for plant disease detection and growth monitoring
- **Solar power** — Photovoltaic panel with battery backup for energy-autonomous operation
- **Spatial sensing** — Multiple sensor nodes with mesh networking for large-scale greenhouse coverage
- **Mobile application** — Native Android/iOS app for farmer notifications and remote monitoring

References

1. FAO, "The future of food and agriculture — Trends and challenges," Food and Agriculture Organization of the United Nations, Rome, 2017.
2. T. Ahonen, R. Virrankoski, and M. Elmusrati, "Greenhouse monitoring with wireless sensor network," in Proc. IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, pp. 403–408, 2008.
3. R. Vimal and M. Subha, "Smart Farming: Automated Monitoring System Using IoT," International Journal of Engineering Research & Technology, vol. 9, no. 6, pp. 945–949, 2020.
4. Espressif Systems, "ESP32 Technical Reference Manual," Version 5.1, 2023.
5. N. K. Nawandar and V. R. Satpute, "IoT based low cost and intelligent module for smart irrigation system," Computers and Electronics in Agriculture, vol. 162, pp. 979–990, 2019.
6. R. K. Kodali and A. Jain, "IoT based smart greenhouse," in Proc. IEEE Region 10 Humanitarian Technology Conference, pp. 1–5, 2016.
7. A. Sagheer, M. Mohammed, K. Riad, and M. Alhajhoj, "A cloud-based IoT platform for precision control of soilless greenhouse cultivation," Sensors, vol. 21, no. 1, p. 223, 2021.
8. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.
9. N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," in Proc. IEEE International Systems Engineering Symposium, pp. 1–7, 2017.
10. R. A. Atmoko, R. Riantini, and M. K. Hasin, "IoT real time data acquisition using MQTT protocol," Journal of Physics: Conference Series, vol. 853, p. 012003, 2017.
11. A. Tzounis, N. Katsoulas, T. Bartzanas, and C. Kittas, "Internet of Things in agriculture, recent advances and future challenges," Biosystems Engineering, vol. 164, pp. 31–48, 2017.
12. R. Barry, "Mastering the FreeRTOS Real Time Kernel: A Hands-On Tutorial Guide," Real Time Engineers Ltd., 2016.
13. H. Sampaio, M. Vieira, and R. Sousa, "Low-cost smart greenhouse monitoring system based on ESP32," in Proc. International Conference on Smart Technologies, pp. 1–6, 2022.
14. Eclipse Foundation, "Mosquitto — An Open Source MQTT Broker," Available: https://mosquitto.org, 2024.
15. S. Tiagrajah and S. Velayutham, "Intelligent greenhouse management system using IoT," International Journal of Advanced Computer Science and Applications, vol. 12, no. 3, pp. 245–252, 2021.