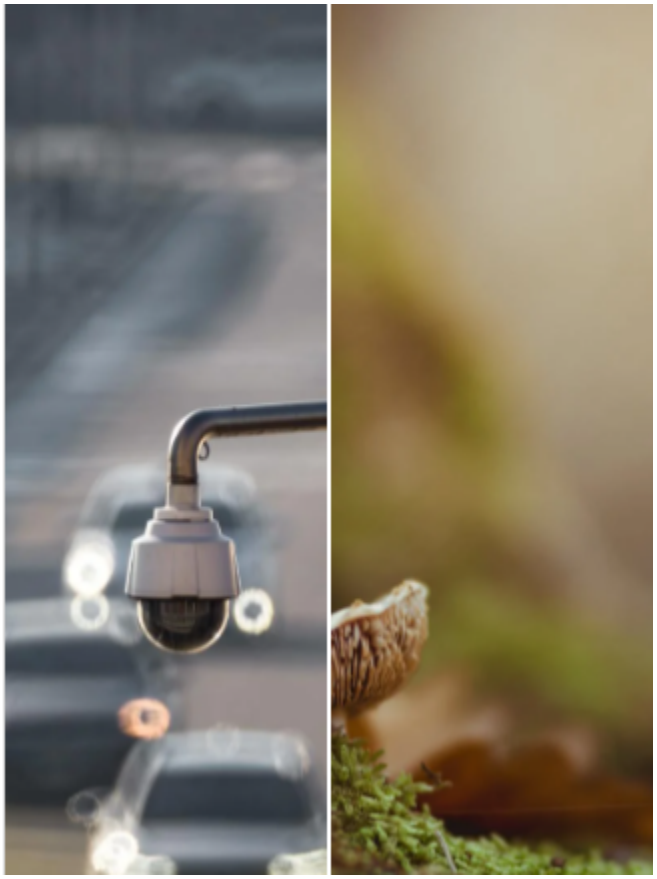
Enoki Cultivation House



Software Engineering for Internet of Things
University of L'Àquila, Italy
Professor Davide Di Ruscio

Marcia Ramalho Rodrigues
Thinakone Louangdy

February, 2025

Table of Contents

1. Introduction

Growing Enoki mushrooms requires precise environmental conditions, including low temperatures, high humidity, and controlled CO_2 levels. In traditional farming, manual monitoring and adjustments often lead to inefficiencies, inconsistencies in growth, and increased labor costs. Smart agriculture offers solutions through automation and IoT-driven control, which ensures optimal growing conditions with minimal human intervention.

To address these challenges, we propose an IoT-based automated cultivation system that integrates real-time monitoring and automated climate control to optimize growing conditions. This system enables precise adjustments to the cultivation environment, leveraging advanced sensor networks and data analytics. By automating climate regulation and utilizing real-time data visualization, this approach improves efficiency, enhances resource management, and optimizes the overall yield of Enoki mushrooms.

2. Objective

The objective of this project is to develop an automated IoT system for Enoki mushroom cultivation.

The system goal is to maintain ideal environmental conditions with minimal human intervention. In order to do that, the system continuously monitors and adjusts temperature, humidity, and CO_2 levels using an MQTT-based network and automated actuators. In addition, the application provides real-time alerts and data visualization to facilitate decision-making.

Additionally, the system is designed to meet scalability, efficiency, and adaptability requirements, ensuring that it can be expanded for larger-scale mushroom cultivation while maintaining cost-effectiveness and ease of deployment.

3. Functional Requirements

3.1. Sensor Integration

The system integrates simulated sensors developed using a Python script. These sensors generate realistic environmental data, including temperature, humidity, and CO_2 levels, to mimic real-world mushroom cultivation conditions.

The data is structured with timestamps and unique identifiers for accurate tracking. The script ensures that gradual fluctuations and occasional anomalies in environmental conditions are introduced to simulate real-world variability.

3.2. Communication Protocols

The system uses MQTT as the primary communication protocol, allowing efficient and lightweight messaging between components.

Sensors publish data to specific MQTT topics, which are then subscribed to by Node-RED for processing and automation. The following MQTT topics are used:

- **Temperature Sensors:**
  - `enoki/house_1/sensor/temperature/device_1`
  - `enoki/house_1/sensor/temperature/device_2`
  - `enoki/house_1/sensor/temperature/device_3`
- **Humidity Sensors:**
  - `enoki/house_1/sensor/humidity/device_1`
  - `enoki/house_1/sensor/humidity/device_2`
  - `enoki/house_1/sensor/humidity/device_3`
- **CO_2 Sensors:**
  - `enoki/house_1/sensor/co2/device_1`
  - `enoki/house_1/sensor/co2/device_2`

Each topic follows a structured naming convention, categorizing data by house, sensor type, and individual device. This structure facilitates scalability by enabling the easy addition of new sensors or houses while maintaining clear organization.

The hierarchical topic format allows efficient data retrieval and filtering, reducing network congestion and improving system performance. Additionally, this design supports modular expansion, where multiple mushroom houses can be integrated into the system with minimal configuration adjustments, making it adaptable to larger-scale farming operations.

This structured topic hierarchy ensures efficient organization and retrieval of data, supporting scalability and modular expansion within the system.

3.3. Data Processing

Data processing occurs within a predefined flow in Node-RED. Since the data is generated by a Python script, it is already pre-formatted before entering the flow.

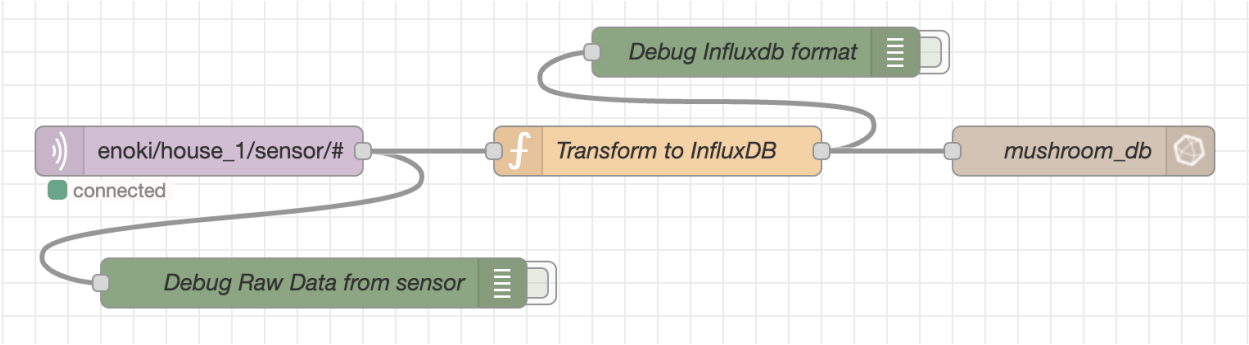
Here, Node-RED receives and processes this structured data, ensuring that it is correctly parsed and forwarded to the appropriate components for storage, visualization, and actuation.

The figure below illustrates the data processing flow in Node-RED:

1. **MQTT Input Node (`enoki/house_1/sensor/#`)**: This node subscribes to all sensor topics under `enoki/house_1/sensor/#`, collecting real-time temperature, humidity, and CO_2 data.
2. **Debug Node (`Debug Raw Data from sensor`)**: This node is used to inspect raw sensor data before transformation, ensuring correctness and identifying potential anomalies.
3. **Function Node (`Transform to InfluxDB`)**: This node processes incoming sensor data, formatting it appropriately for storage in InfluxDB. It ensures the data includes timestamps and structured fields for efficient querying.
4. **Debug Node (`Debug InfluxDB format`)**: This node inspects the transformed data before insertion into the database, verifying proper formatting.
5. **InfluxDB Output Node (`mushroom_db`)**: This node sends the formatted data to the InfluxDB instance for long-term storage and analysis.

This flow ensures that sensor data is efficiently processed, validated, and stored, enabling robust monitoring and historical analysis of environmental conditions in the Enoki mushroom cultivation system.

Figure 1. Data processing and data storage flow.



3.4. Data Storage

Node-RED sends the processed sensor data to an instance of InfluxDB running locally within a Docker container. This database efficiently handles time-series data, enabling efficient storage, retrieval, and historical analysis of environmental conditions within the Enoki cultivation house.

3.5. Visualization

Node-RED facilitates the retrieval of stored data from InfluxDB and shares it with Grafana for visualization. Grafana dashboards display real-time and historical sensor data, enabling farmers to track environmental trends, identify patterns, and make data-driven decisions to optimize mushroom cultivation.

The figure below illustrates the visualization of sensor data within Grafana:

1. **CO_2 Levels**: The top-left graph presents the CO_2 levels (ppm) recorded by two devices over time. Different colors distinguish data from each device, allowing for easy comparison and analysis of variations.
2. **Temperature Readings**: The top-right graph tracks the temperature (°C) from multiple sensors. This helps in monitoring fluctuations and ensuring the environment remains within the desired range.
3. **Humidity Trends**: The bottom-left graph displays the humidity (%) from three different sensors, showing variations over time and highlighting any potential deviations from optimal conditions.
4. **Gauge Displays**: The bottom-right section consists of three gauge indicators representing the latest values recorded for CO_2 (ppm), humidity (%), and temperature (°C). These provide a quick overview of the current environmental conditions within the cultivation house.

This structured visualization approach ensures that critical environmental parameters are easily monitored, providing users with actionable insights for maintaining ideal growing conditions.

Figure 2. Grafana dashboard - CO_2 (ppm), temperature (°C) and humidity (%) time-series plot and real-time measurements.

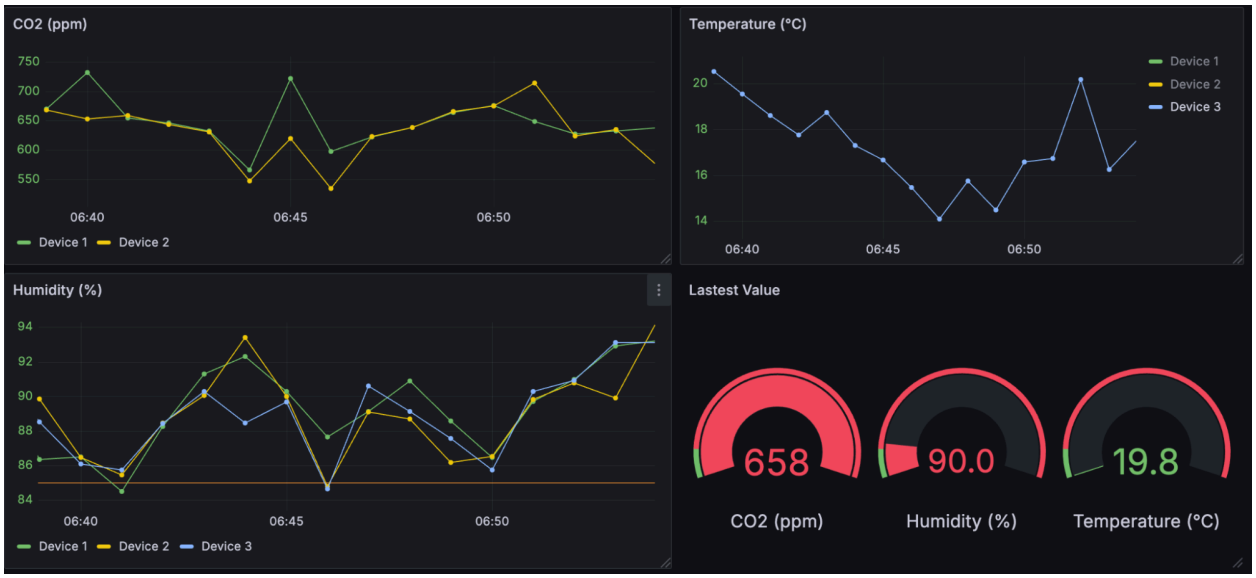


Figure 3. Grafana dashboard - Time-series for CO_2 (ppm), temperature (°C) and humidity (%).

3.6. Actuators

When environmental parameters exceed predefined thresholds, Node-RED triggers specific actuator subscribers named `actuator_temperature`, `actuator_humidity`, and `actuator_co2`.

These subscribers control actuators within the Enoki cultivation house to take corrective actions, such as adjusting ventilation, activating humidifiers, or regulating CO_2 levels to maintain optimal growing conditions.

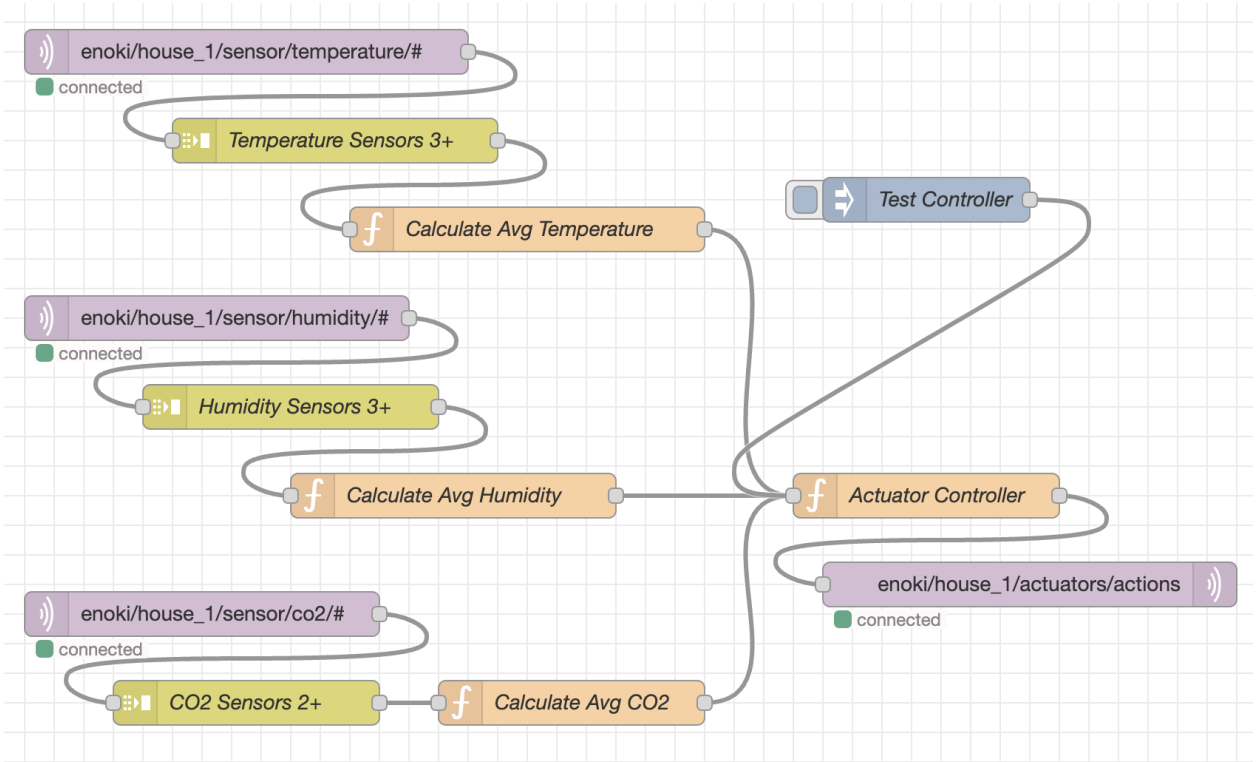
The figure below illustrates the actuator control flow within Node-RED:

1. **Sensor Data Input**:
    - Temperature, humidity, and CO_2 data is received from MQTT topics (`enoki/house_1/sensor/temperature/#`, `enoki/house_1/sensor/humidity/#`, and `enoki/house_1/sensor/co2/#`).
    - The input nodes ensure the system continuously monitors environmental conditions.
2. **Sensor Aggregation**:
    - Temperature readings from multiple sensors are processed in the `Temperature Sensors 3+` node, followed by an average calculation (`Calculate Avg Temperature`).

    - Humidity and CO_2 levels are similarly aggregated and averaged in their respective processing nodes.
3. **Actuator Control Logic**:
    - The `Actuator Controller` node receives the calculated average values for temperature, humidity, and CO_2.
    - Based on predefined threshold conditions, it determines whether any corrective actions are needed.
4. **Actuator Action Execution**:
    - If adjustments are required, the actuator control signals are published to the MQTT topic `enoki/house_1/actuators/actions`.
    - This ensures that the necessary devices (ventilation, humidifiers, or CO_2 regulators) respond appropriately to maintain optimal growing conditions.
5. **Testing and Debugging**:
    - The `Test Controller` node allows manual testing and validation of the actuator control logic before deployment.

This structured approach ensures precise, automated environmental control while allowing real-time monitoring and adjustments based on sensor data.

Figure 4. Actuators control flow on node-red.



3.7. Alerting Mechanisms

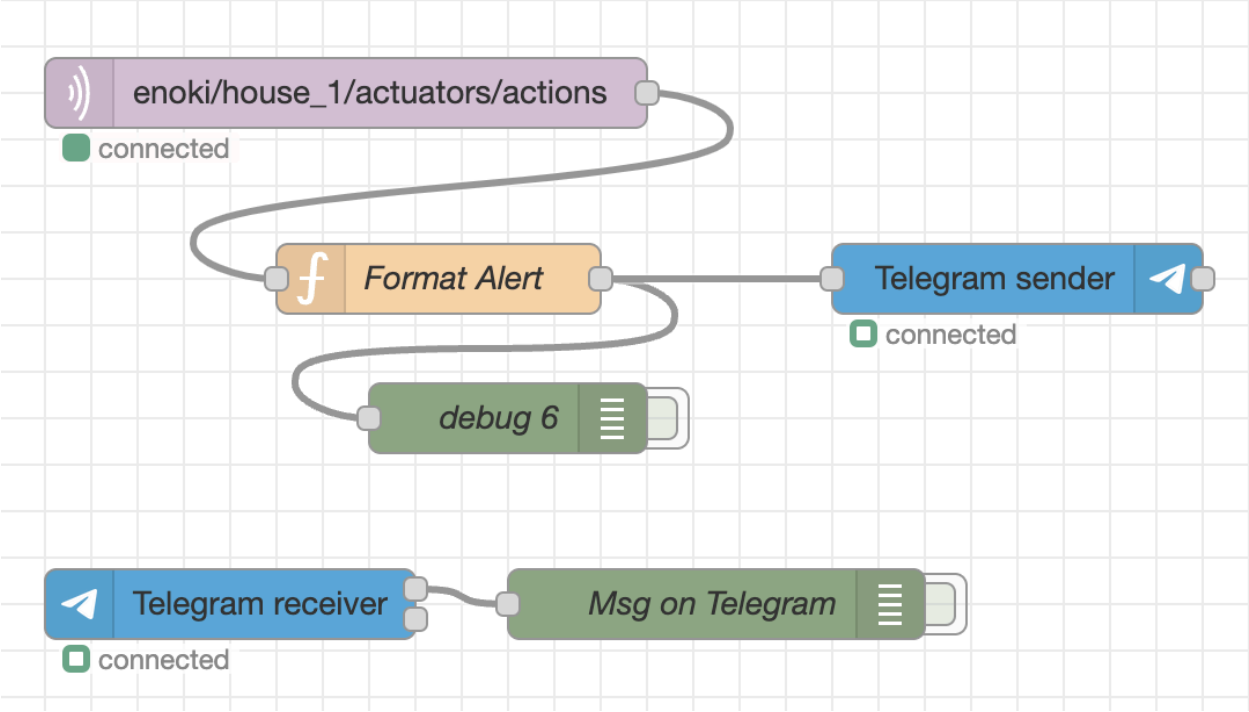
Node-RED manages the alerting mechanism by monitoring actuator activity and sending notifications when necessary. If sensor values exceed acceptable thresholds and actuators are activated, an alert is sent to a Telegram bot to notify users in real-time.

The figure below illustrates the alerting flow within Node-RED:

1. **Triggering Event**:
    - Alerts are triggered when actions are published to the `enoki/house_1/actuators/actions` MQTT topic.
    - This indicates that an environmental condition exceeded predefined limits, requiring actuator intervention.
2. **Formatting the Alert**:
    - The `Format Alert` node processes the raw data received from the actuator topic, structuring it into a readable message for end-users.
3. **Sending Notifications**:
    - The formatted alert is sent to the `Telegram sender` node, which transmits the message to a designated Telegram bot.
    - A `debug` node is also included to log the formatted messages, ensuring correct message formatting before dispatch.
4. **Receiving User Input**:
    - The `Telegram receiver` node listens for responses or commands from users interacting with the bot.
    - The received message is processed by the `Msg on Telegram` node, allowing users to acknowledge alerts or send commands.

This alerting mechanism ensures that users are promptly informed about critical environmental changes, enabling them to take necessary actions to maintain optimal mushroom cultivation conditions.

Figure 5. Alerting mechanism control flow on node-red.



4. Non-Functional Requirements

The system was designed to meet several non-functional requirements that ensure reliability, scalability, and security. The following table outlines these requirements and how they are addressed:

Requirement	Description
Portability	The system is containerized using Docker, making it deployable across multiple platforms with minimal configuration changes.
Scalability	The MQTT-based architecture and structured topic hierarchy allow seamless addition of new sensors, actuators, and cultivation units without significant modifications.
Resilience	The system ensures uninterrupted operation by handling network failures and enabling automatic recovery mechanisms within Node-RED and InfluxDB.
User-Friendly Design	Grafana dashboards provide intuitive, real-time visualizations, and the Telegram alerting system ensures that users receive timely notifications.

Security	Secure MQTT communication with authentication and access control prevents unauthorized access to data and actuator commands.
Real-time Monitoring	Continuous data acquisition from sensors allows immediate adjustments through automated actuators and real-time alerts to maintain optimal conditions.

5. System Architecture

The system follows a modular and scalable architecture, designed to facilitate real-time monitoring, automated control, and alerting for Enoki mushroom cultivation. The architecture consists of multiple interconnected components, ensuring seamless data flow and automation:

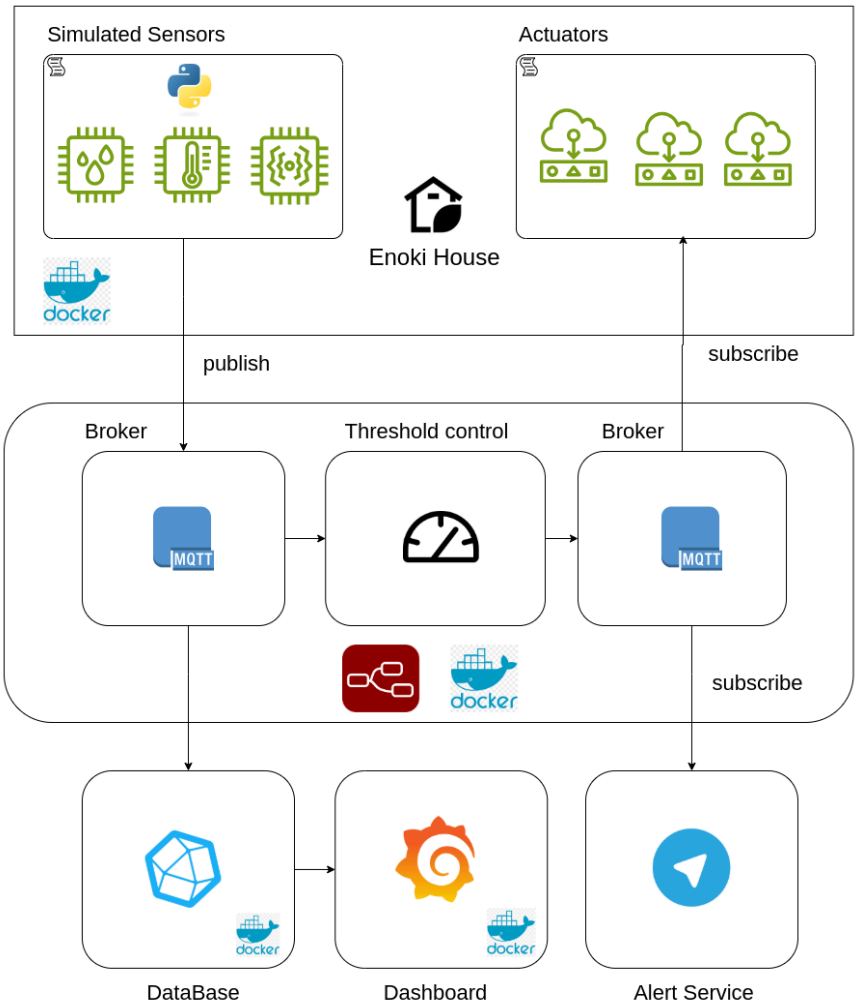
1. **Simulated Sensors**:
    - Virtual sensors generate data for temperature, humidity, and CO_2 levels within the Enoki house.
    - These sensors publish data to an MQTT broker, making the information available to other components.
2. **Message Broker (MQTT)**:
    - Acts as the central communication hub, receiving and distributing sensor data.
    - Sensors publish data to specific topics, and other system components subscribe to these topics to receive relevant updates.
3. **Threshold Control**:
    - Node-RED processes sensor data and applies threshold-based logic.
    - If environmental parameters exceed predefined limits, appropriate control actions are triggered.
4. **Actuators**:
    - Subscribes to the MQTT broker to receive control commands.
    - Responsible for regulating conditions inside the Enoki house, such as adjusting ventilation, activating humidifiers, or controlling CO_2 levels.
5. **Database (InfluxDB)**:
    - Stores time-series data for historical analysis and visualization.
    - Enables trend tracking and optimization of environmental conditions.
6. **Dashboard (Grafana)**:
    - Retrieves data from the database and presents real-time and historical visualizations.
    - Provides insights into environmental conditions and system performance.
7. **Alert Service (Telegram Bot)**:
    - Subscribes to MQTT messages to detect critical environmental changes.
    - Sends alerts to users via Telegram when thresholds are exceeded, ensuring prompt intervention.

System Flow

1. Sensors publish environmental data to the MQTT broker.
2. The broker forwards data to the threshold control system for processing.
3. Threshold control determines if any corrective actions are needed and sends commands to actuators.
4. The database logs all sensor data for monitoring and analytics.
5. Grafana retrieves and visualizes stored data for real-time decision-making.
6. The alert service notifies users in case of abnormal conditions, ensuring proactive responses.

This architecture ensures efficient environmental management, providing a robust, scalable, and automated approach for Enoki mushroom cultivation. The key components and their interactions are illustrated in the diagram below.

Figure 6. System architecture diagram.



6. Technologies Used

The system leverages various technologies to ensure efficient data processing, automation, and visualization. The table below outlines the key technologies used and their purpose:

Technology	Purpose
Python	Simulates sensor data for temperature, humidity, and CO_2 levels.
MQTT (Mosquitto)	Facilitates lightweight and efficient real-time messaging between sensors, actuators, and the control system.
Node-RED	Manages data flow, processes sensor values, triggers actuators, and integrates system components.
InfluxDB	Stores sensor data in a time-series format for historical analysis and real-time monitoring.
Grafana	Visualizes environmental data and system performance for easy monitoring.
Telegram Bot	Sends real-time alerts to notify users when environmental thresholds are exceeded.
Docker	Provides containerization for easy deployment and scalability of all system components.

7. System Functionality

The system performs various automated tasks to ensure optimal environmental conditions for Enoki mushroom cultivation. The table below summarizes the key functionalities:

Functionality	Description
Real-time Monitoring	Continuously collects and processes data from simulated sensors.
Automated Actuation	Controls ventilation, humidifiers, and CO_2 regulators based on predefined thresholds.
Threshold-based Alerts	Sends notifications to users via Telegram when environmental conditions exceed safe limits.
Historical Data	Stores and retrieves time-series sensor data for trend analysis and

Analysis	decision-making.
Visualization	Displays real-time and historical data using Grafana dashboards.
Scalability	Designed to integrate additional sensors and actuators for larger deployments.
User Interaction	Allows users to monitor conditions and receive alerts for proactive management.

This structured approach ensures integration of technologies and functionalities to support efficient and automated mushroom cultivation

8. Conclusion

This IoT-based Enoki mushroom cultivation system successfully automates climate control and monitoring, enhancing efficiency and sustainability in smart agriculture. By integrating MQTT for communication, Node-RED for data processing, InfluxDB for structured storage, and Grafana for visualization, the system ensures real-time monitoring and automatic regulation of environmental conditions.

The automation of actuators minimizes manual intervention, while the Telegram alerting mechanism provides timely notifications for immediate corrective actions. The modular and scalable architecture allows seamless expansion with additional sensors and actuators, making it adaptable to larger farming operations.

Future improvements could be focused on enhancing IoT components, including optimizing MQTT topic structures for better efficiency, implementing edge computing for faster processing and reduced latency, and integrating enhanced security mechanisms for data integrity and device authentication. These advancements will further improve system reliability, responsiveness, and scalability, ensuring long-term sustainability and effectiveness in automated agriculture.