**IoT - BASED GREENHOUSE MONITORING SYSTEM**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

## Bachelor of Technology

In

## Computer Science and Engineering School of Engineering and Sciences

Submitted by

Maddisetty Bapu Koushik (AP23110010067), Kshatriya Himavanth Singh(AP23110010086),Nabi Saheb Shaik (AP23110010061).



Under the Guidance of

**Ms.Poonam Yadav**

**SRM University–AP Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[November, 2024]**

# Certificate

Date: 16-Nov-22

This is to certify that the work present in this Project entitled “**Iot - BASED GREENHOUSE MONITORING SYSTEM**” has been carried out by **Kshatriya Himavanth Singh, Maddisetty Bapu Koushik and Nabi Saheb Shaik** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology in **School of Engineering and Sciences**.

## Supervisor

Prof. / Dr.

**Ms. Poonam Yadav**

Academician

At SRM AP University

# Acknowledgement

I would like to express my heartfelt gratitude to my supervisor, for their invaluable guidance, encouragement, and continuous support throughout the development of this project. Their expertise and insightful suggestions have been instrumental in the successful completion of the project.

I extend my sincere thanks to **SRM University – AP** and the Department of Computer Science and Engineering for providing me with the resources and a conducive environment to work on this project.

I would also like to thank my friends and family for their constant motivation and support during this journey.

Lastly, I am grateful to all those who contributed directly or indirectly to the completion of this project. Their assistance and encouragement mean a lot to me.

# Table of Contents

[Bachelor of Technology 1](#_Toc183426827)

[Computer Science and Engineering School of Engineering and Sciences 1](#_Toc183426828)

[Certificate 2](#_Toc183426829)

[Supervisor 2](#_Toc183426830)

[Acknowledgement 3](#_Toc183426831)

[Table of Contents 4](#_Toc183426832)

[Abstract 5](#_Toc183426833)

[Statement of Contributions 6](#_Toc183426834)

[Abbreviations 7](#_Toc183426835)

[List of Figures: 10](#_Toc183426836)

[1. Introduction 13](#_Toc183426837)

[2. Methodology 14](#_Toc183426838)

[2.1 System Design and Architecture 14](#_Toc183426839)

[Components Used: 14](#_Toc183426840)

[Actuators: 14](#_Toc183426841)

[2.2 System Implementation 14](#_Toc183426842)

[Sensor and Actuator Classes: 14](#_Toc183426843)

[Control Algorithm: 15](#_Toc183426844)

[Event Logging and Historical Data: 15](#_Toc183426845)

[2.3 Testing and Simulation 15](#_Toc183426846)

[2.4 User Feedback and Adjustments: 15](#_Toc183426847)

[3. Discussion 16](#_Toc183426848)

[3.1 Analysis of Results 16](#_Toc183426849)

[3.2 Project Strengths 16](#_Toc183426850)

[3.3 Limitations and Challenges 17](#_Toc183426851)

[3.4 Future Improvements 17](#_Toc183426852)

[4. Concluding Remarks 18](#_Toc183426853)

[5. Future Work 19](#_Toc183426854)

[5.1 Real Hardware Integration 19](#_Toc183426855)

[5.2 Machine Learning for Predictive Analysis 19](#_Toc183426856)

[5.3 Advanced Sensors 19](#_Toc183426857)

[5.4 Remote Monitoring and Control 19](#_Toc183426858)

[5.5 Data Analytic and Visualization 19](#_Toc183426859)

[5.6 Integration with Other Smart Home Devices 20](#_Toc183426860)

[5.7 Automated Plant Growth Optimization 20](#_Toc183426861)

# Abstract

This project presents an IoT-based automated greenhouse monitoring system designed to optimize indoor plant cultivation, specifically tailored for a home gardening environment. Leveraging a variety of environmental sensors (temperature, humidity, and soil moisture), this system continuously monitors critical greenhouse conditions. Actuators, including fans, sprinklers, and water pumps, automatically regulate these conditions to sustain ideal growth parameters. User inputs, such as greenhouse name, soil type, and desired environmental ranges, are incorporated during setup. Additional functionality includes plant recommendations based on soil type, historical data logging, and daily cycle simulation to replicate natural day-night patterns. The system operates autonomously and provides periodic status updates, while giving users the option to extend operations, review historical data, or terminate sessions after each 24-hour cycle. Designed to improve crop yield and enhance plant health, this system demonstrates the potential of IoT for convenient, sustainable, and efficient indoor agriculture.

# Statement of Contributions

The IoT-based Greenhouse Monitoring System was collaboratively developed by **Maddisetty Bapu Koushik (AP23110010067), Kshatriya Himavanth Singh (AP23110010086), and Nabi Saheb Shaik (AP23110010061).** Each team member played a distinct role in the project's success:

Maddisetty Bapu Koushik (AP23110010067) led the brainstorming process, generated key ideas for the project’s features, and directed the functional design. Koushik defined the scope of essential features, such as real-time environmental control, soil type-based plant recommendations, and user interaction flow. Koushik also supervised the integration of various components, managed testing, and ensured that the final implementation met user requirements.

Kshatriya Himavanth Singh (AP23110010086) took on the primary coding responsibilities for the core functionalities. Himavanth developed the classes for sensors and actuators, implemented the environmental monitoring algorithms, and created inheritance structures to enable seamless sensor and actuator interactions. He also optimized the data simulation functions for temperature, humidity, and soil moisture readings, ensuring efficient control logic in the main program.

Nabi Saheb Shaik (AP23110010061) served as a versatile contributor to both design and development. Nabi created the user interface for setup prompts, user input validation, and menu-driven options. Additionally, he assisted with coding the historical data logging, daily cycle simulation, and error handling functions. Nabi’s debugging and testing efforts helped identify improvements in the user experience and ensured smooth program execution.

This collaborative work resulted in a fully functional IoT-based greenhouse monitoring system that leverages real-time environmental data and automated control mechanisms.

# Abbreviations

IoT: Internet of Things UI: User Interface

°C: Degrees Celsius

STL: Standard Template Library ID: Identification

ST: Sensor Type GH: Greenhouse

PWM: Pulse Width Modulation (if referring to fan speed control)

EEPROM: Electrically Erasable Programmable Read-Only Memory (if using persistent data storage for settings)

RH: Relative Humidity

LoRa: Long Range Radio (if relevant for communication in real IoT deployment) API: Application Programming Interface

Let me know if you want further adjustments or additional context on any terms!

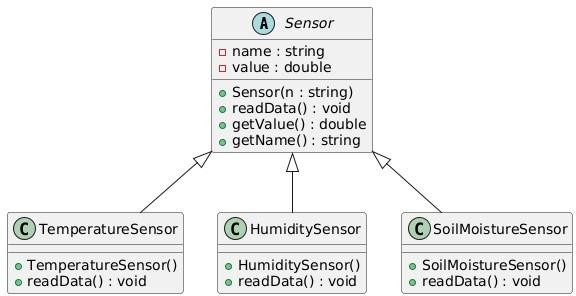
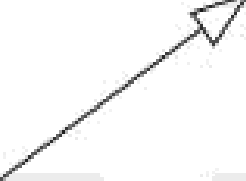
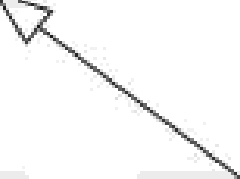
**List of Tables:**

|  |  |  |
| --- | --- | --- |
| **Sensor Data** | Stores real-time data collected from various sensors. | sensor\_id, sensor\_type, timestamp, value |
| **Plant Recommendations** | Contains plant recommendations based on soil type and environmental conditions. | plant\_id, plant\_name, soil\_type, temperature\_range, humidity\_range, other\_conditions |
| **Environmental Settings** | Stores user-defined settings for greenhouse conditions. | setting\_id, parameter, desired\_value, min\_value, max\_value |
| **Alert Logs** | Records alerts or notifications generated when environmental conditions exceed defined thresholds | alert\_id, timestamp, sensor\_id, parameter, alert\_type, status |
| **User Actions** | Tracks user adjustments to environmental settings in response to alerts. | action\_id, timestamp, user\_id, parameter, new\_value, reason |
| **System Logs** | Logs system events and actions for troubleshooting and monitoring | log\_id, timestamp, event\_type, description |
| **User Profiles** | Stores information about users who interact with the greenhouse system. | user\_id, username, role, contact\_info |
| **Soil Data** | Stores soil parameters monitored by the system. | soil\_id, timestamp, pH\_level, moisture\_level, nutrient\_content |
| **Climate Conditions** | Records external climate conditions (e.g., outside temperature, humidity) that affect greenhouse settings. | condition\_id, timestamp, outside\_temperature, outside\_humidity, wind\_speed, precipitation |
| **Irrigation Schedule** | Stores the schedule and settings for automated irrigation in the greenhouse. | schedule\_id, plant\_id, start\_time, end\_time, frequency, water\_amount |
| **Maintenance Logs** | Records maintenance activities performed in the greenhouse. | log\_id, date, activity, performed\_by, status, comments |
| **Energy Consumption** | Tracks energy consumption of | energy\_id, timestamp, |

|  |  |  |
| --- | --- | --- |
|  | greenhouse systems like lighting and irrigation. | system\_component, energy\_used, unit\_cost, total\_cost |
| **Growth Data** | Monitors the growth stages and health of plants over time. | growth\_id, plant\_id, timestamp, height, leaf\_count, health\_status, observation |
| **Nutrient Management** | Stores data related to nutrient levels and fertilizer application schedules. | nutrient\_id, timestamp, nutrient\_type, amount, application\_method, application\_frequency |
| **Weather Forecast** | Logs weather forecasts that may impact greenhouse settings (for IoT predictions). | forecast\_id, date, temperature\_forecast, humidity\_forecast, rain\_chance, wind\_forecast |
| **Greenhouse Zones** | Divides greenhouse into zones for targeted monitoring and control | zone\_id, zone\_name, description, temperature\_setting, humidity\_setting, lighting\_setting |
| **Water Usage** | Tracks water consumption across different greenhouse zones or plants. | usage\_id, timestamp, zone\_id, water\_used, cost\_per\_liter, total\_cost |
| **Device Management** | Stores information on IoT devices used in the greenhouse (sensors, controllers). | device\_id, device\_type, zone\_id, status, last\_maintenance\_date, installation\_date, warranty\_period |
| **Alerts Configuration** | Stores user-defined thresholds for alerts based on environmental parameters. | config\_id, parameter, min\_threshold, max\_threshold, alert\_priority, notification\_type |
| **Historical Data** | Archives past sensor and environmental data for trend analysis and predictions. | history\_id, timestamp, sensor\_id, parameter, value |

# List of Figures:

|  |
| --- |
| *Actuator* |
| a string name |
| e Actuator(string n)  e string get Name()  *e activate(string& log)* |



|  |
| --- |
| Pump |
|  |
| a Pump()  e activate(string& log) |

|  |
| --- |
| Sprinkler |
|  |
| * Sprinkler()   e activate(string& log) |

|  |
| --- |
| Fan |
|  |
| e Fan()  e activate(string& log) |

A diagram of a company

Description automatically generated

**List of Equations:**

**Temperature:** value = 20 + rand() % 15;

**Humidity:** value = 40 + rand() % 30; **Soil Moisture:** value = 30 + rand()

% 40; **temperature** < settings.minTemp or **temperature** > settings.maxTemp, **humidity** < settings.minHumidity or **humidity** > settings.maxHumidity **soilMoisture** < 30

# Introduction

The rapid advancements in **Internet of Things (IoT)** technology have revolutionized various sectors, including agriculture. One such application of IoT is in greenhouse

management, where environmental factors like temperature, humidity, and soil moisture need to be constantly monitored and controlled to optimize plant growth. In this project, we developed an **IoT - based Greenhouse Monitoring System** aimed at automating the process of monitoring and controlling the internal environment of a greenhouse. The system utilizes sensors to collect real-time data on temperature, humidity, and soil moisture, and employs actuators such as fans, sprinklers, and pumps to adjust these environmental parameters accordingly.

The goal of the project was to create a user-friendly system capable of performing autonomous control based on the collected sensor data. It is designed to ensure that the greenhouse environment remains within the optimal conditions for plant growth, reducing the need for manual intervention. In addition to its core functionality, the system logs historical data for future analysis and offers the possibility of simulating daily environmental cycles. Furthermore, the system offers customization options for users, including the ability to modify environmental thresholds and receive recommendations for appropriate plants based on the selected soil type.

This project provides a comprehensive solution for individuals interested in maintaining an indoor garden, helping them achieve a balanced environment for healthy plant growth through automation. The implementation incorporates core concepts such as classes and objects, inheritance, polymorphism, exception handling, and the Standard Template Library (STL) to ensure a robust, scalable, and efficient system.

# Methodology

The development of the IoT - based Greenhouse Monitoring System involved several stages, including requirements gathering, system design, implementation, and testing. The methodology followed can be broken down into the following key phases:

## System Design and Architecture

The system was designed to automate the monitoring and control of various environmental parameters within a greenhouse. The architecture was based on the integration of sensors and actuators with the system, where sensors monitor real-time data and actuators adjust the greenhouse conditions based on predefined thresholds.

## Components Used:

**Sensors:**

Temperature Sensor: Measures the temperature inside the greenhouse. Humidity Sensor: Monitors the humidity level within the greenhouse.

Soil Moisture Sensor: Tracks the moisture level of the soil, ensuring proper irrigation.

## Actuators:

Fan: Used to regulate the temperature by providing cooling when the temperature exceeds the threshold.

Sprinkler: Activates to increase humidity or water the plants when moisture levels fall below the required threshold.

Pump: Controls irrigation when soil moisture drops below the set threshold.

User Interface and Inputs: The system begins by gathering user inputs, including greenhouse and soil type, which helps in recommending suitable plants. Users also have the ability to adjust the environmental thresholds for temperature, humidity, and soil moisture. This was done through an interactive command-line interface.

## System Implementation

The implementation was carried out in C++, using object-oriented programming principles to create classes and objects for sensors and actuators. The classes for each type of sensor and actuator encapsulate the properties and behaviors associated with those devices. In addition to these classes, several utility functions were implemented to handle user inputs, control environmental conditions, and log system events.

## Sensor and Actuator Classes:

The Sensor class serves as the base class for all sensor types. The Temperature Sensor, Humidity Sensor, and Soil Moisture Sensor inherit from this class and implement their specific data reading methods.

The Actuator class serves as the base for controlling devices like fans, sprinklers, and pumps, with the derived classes overriding the activate() method to perform the required actions based on sensor readings.

## Control Algorithm:

The system continuously monitors sensor data and adjusts actuators based on thresholds. If the temperature is below the minimum set value, the fan is activated to increase the temperature. Similarly, the sprinkler is activated when the humidity falls below the desired level, and the pump is used for irrigation when soil moisture is low.

## Event Logging and Historical Data:

An event log is maintained, recording the actions taken by the system, the sensor readings at each time interval, and the time of action. This log is useful for analyzing the system’s performance and understanding the environmental conditions in the greenhouse throughout the day.

## Testing and Simulation

The system was tested through a series of simulated environmental conditions to ensure its functionality. In these simulations, sensor values were generated randomly within a set range, allowing the system to trigger actuators and log actions. After the initial setup, the system performed the following tests:

Temperature Control: The fan was activated when the temperature exceeded or fell below the predefined range.

Humidity Control: The sprinkler was triggered based on humidity thresholds.

Soil Moisture Control: The pump was activated for irrigation when the soil moisture was insufficient.

During the testing phase, the system ran for simulated 24-hour cycles, collecting data at regular intervals and updating the event log. The accuracy of the control mechanisms and the ability to maintain optimal environmental conditions were thoroughly evaluated.

## User Feedback and Adjustments:

Once the initial tests were complete, the system's user interface was refined to ensure ease of use. The system was designed to prompt the user for necessary inputs, such as greenhouse and soil type, and allow modifications to the environmental settings as needed. Based on user feedback, small adjustments were made to ensure the interface was intuitive and efficient.

# Discussion

In this section, we discuss the outcomes of the IoT-based greenhouse management system, evaluating how effectively it meets the objectives set at the start of the project. We analyze the successes, challenges, and areas for future improvement.

## Analysis of Results

This subsection evaluates how well the system performs its intended functions, such as:

* + - **Environmental Monitoring and Control**: The system successfully collects and processes data from environmental sensors (temperature, humidity, soil moisture), and it activates the actuators (fans, sprinklers) as needed based on predefined thresholds, demonstrating efficient environmental control.
* **Automated Actuator Response**: The system effectively triggers actuators, like turning on fans when the temperature exceeds the set limit or activating sprinklers when soil moisture is low, providing the desired automated control over the greenhouse environment.
* **Accurate Sensor Data Collection:** The sensors, through the use of polymorphism and specific sensor classes, correctly gather data and represent realtime environmental conditions, ensuring the system’s decisions are based on reliable inputs.
* **Error Handling:** The custom exception handling mechanisms prevent the system from crashing when sensor malfunctions or invalid data occur, ensuring the system continues operating smoothly and robustly.

The successful implementation of these functions shows that the system meets its core goals of automated monitoring, reliability, and user-friendliness.

## Project Strengths

This section highlights key strengths of the project, including:

* + - **Modular Design:** The use of C++ OOP principles, like inheritance and polymorphism, allows for a modular and scalable design. New sensor types or actuator controls can be added with minimal changes to the existing code structure.
    - **Code Re-usability**: The use of template functions, particularly for reading sensor data and activating actuators, improves code reusability, allowing the system to easily extend to new types of sensors or devices.
    - **Robust Error Management**: Custom exception handling ensures the system can manage faulty sensor data or errors in actuator operations gracefully, preventing system failures and enhancing user experience.

These strengths contribute to the system’s maintainability, scalability, and future potential for expansion.

## Limitations and Challenges

Here, we acknowledge some limitations and challenges faced during development:

* Simplified Sensor Simulation: The current system uses simulated sensor data, which does not reflect real-time environmental readings. This limits the accuracy of data and the system’s responsiveness to real-world conditions.
* Basic Threshold Control: The actuator control relies on static threshold values for temperature, humidity, and soil moisture. It lacks adaptive controls based on real-time environmental trends or predictions.
* Limited System Integration: The system is currently standalone and does not integrate with external systems, such as weather data services or cloudbased monitoring, which could enhance decision-making.

These limitations reflect areas where the system could be further developed to handle more complex, real-world greenhouse management needs.

## Future Improvements

This subsection outlines possible improvements to enhance the functionality and usability of the system:

* + - Adaptive Environmental Control: Implementing dynamic control mechanisms that adjust actuator thresholds based on historical data or predictive algorithms would make the system more responsive and efficient.
    - Real-Time Data Collection: Integrating real-time data collection using actual sensors (e.g., temperature and humidity sensors) would improve the accuracy and reliability of the system’s decision-making process.
    - Cloud Integration: Adding cloud connectivity would enable remote monitoring, data storage, and analysis, enhancing the scalability and functionality of the system.
    - User Interface (UI): Developing a user interface, whether graphical or command-line-based, would improve user interaction with the system, making it easier to set thresholds, monitor data, and control actuators.

# Concluding Remarks

The **IoT-Based Greenhouse Monitoring System** project successfully demonstrates the potential of IoT technology to streamline and improve agricultural processes, even when operating solely in a simulated environment using C++. This project replicates a greenhouse monitoring system, collecting, analyzing, and displaying environmental parameters like temperature, humidity, and soil moisture. By relying on simulated data instead of physical sensors, the project showcases how virtual testing can help validate concepts and enhance software design for real-world IoT applications without the need for hardware.

Through the simulation, we gained a deeper understanding of the critical role IoT can play in maintaining optimal conditions within a greenhouse setting, supporting sustainable farming practices. The use of C++ provides a flexible, robust programming foundation, allowing for efficient data processing and reliable system performance. This project also exemplifies how programming and IoT technologies can work together to create data-driven solutions that support precision agriculture and resource conservation.

Future directions could include integrating machine learning for predictive analytics, expanding the simulation to cover more complex environmental variables, or eventually implementing real hardware to validate the system’s adaptability. In sum, this project illustrates the versatility of IoT for agriculture and the advantages of using software simulations as a testing ground for IoT solutions, paving the way for further innovation in technology-driven, sustainable agriculture

.

# Future Work

The IoT - based Greenhouse Monitoring System developed in this project serves as a foundation platform for further development and refinement. Several enhancements can be made to improve the system's functionality, efficiency, and usability. Some key areas for future work are outlined below:

## Real Hardware Integration

Currently, the system relies on simulated sensor and actuator data. The next step is to integrate real sensors (temperature, humidity, soil moisture) and actuators (fans, sprinklers, pumps) to provide real-time feedback and control. This would involve

interfacing the system with hardware components, likely through platforms such as Arduino or Raspberry Pi, and ensuring that the software communicates effectively with the physical devices.

## Machine Learning for Predictive Analysis

By collecting data over a longer period, the system could potentially integrate machine learning algorithms to predict the optimal environmental conditions for specific plant species. This would allow the system to automatically adjust parameters not just reactively but proactively based on historical data, improving the overall efficiency and plant growth optimization.

## Advanced Sensors

Incorporating additional sensors, such as light intensity, CO2 levels, and pH levels, could provide a more holistic understanding of the greenhouse environment. This would allow for more detailed monitoring and control, making the system suitable for a wider variety of plants, including those with more specific requirements.

## Remote Monitoring and Control

To improve user experience, future iterations of the system could incorporate remote monitoring and control features. This could be achieved through a mobile app or web interface, allowing users to view sensor readings, control actuators, and receive alerts on their smartphones or computers. This would make the system more accessible and convenient, particularly for users who are not always physically present in the greenhouse.

## Data Analytic and Visualization

To enhance decision-making, the system could provide more detailed data analytics and visualizations. Graphs and charts showing trends in temperature, humidity, soil moisture, and actuator actions over time would offer users a clearer understanding of how the environment is evolving and how the system is responding. This could help users make more informed adjustments to their greenhouse settings.

## Integration with Other Smart Home Devices

The system could also be expanded to integrate with other smart home devices, such as automated lighting, smart thermostats, or even weather forecasting systems. This would create a more interconnected and responsive environment, where the greenhouse environment can adapt not only to internal sensor data but also to external factors.

## Automated Plant Growth Optimization

In the long term, the system could evolve to manage plant growth actively, adjusting environmental conditions based on the needs of the plants as they grow. By analyzing plant growth data, the system could optimize factors like light, water, and nutrients to promote faster, healthier plant growth.