CAPSTONE PROJECT REPORT

PROJECT TITLE

Smart Home Automation System

TEAM MEMBERS

Ch.Venkatesh (192210253)

K.Tejesh (192210210)

COURSE CODE / NAME

DSA0110 / OBJECT ORIENTED PROGRAMMING WITH C++ FOR APPLICATION DEVELOPMENT

SLOT A

DATE OF SUBMISSION

12.11.2024

SIMATS ENGINEERING

Engineer to Excel

SIMATS

Saveetha Institute of Medical And Technical Sciences

(Declared as Deemed to be University under Section 3 of UGC Act 1956)

BONAFIDE CERTIFICATE

Certified that this project report SMART HOME AUTOMATION is the bonafide work of Ch.Venkatesh (192210253) and K.Tejesh (192210210) who carried out the project work under my supervision.

SUPERVISOR

Dr. S. Sankar

ABSTRACT

Smart home automation is an emerging field that integrates various IoT (Internet of Things) devices to create efficient, convenient, and secure home environments. This project uses Cisco Packet Tracer, a network simulation tool, to demonstrate a smart home automation system. The simulation includes essential components of a modern automated home, such as lighting, security cameras, motion detectors, and temperature control systems, connected through a network to a central control unit.

In C++, abstraction is achieved by defining classes with pure virtual functions, creating a blueprint for derived classes to implement specific functionality. For instance, in a smart home automation system, an abstract base class `Device` can be created with pure virtual functions like `turnOn` and `turnOff`, which represent actions applicable to all smart devices. This enables the `SmartHomeController` to manage various devices seamlessly, triggering device-specific behaviour without needing to know the exact type of device. Through abstraction, the code becomes modular, extendable, and easier to maintain, as new device types can be added with minimal changes to the existing code structure.

The smart devices in the simulation communicate over a local network, managed by a router and controlled by a central controller, representing either a smartphone app or a web interface. This setup allows users to interact with devices remotely, control home appliances, receive alerts, and monitor security in real time. Cisco Packet Tracer enables visualisation of device communication protocols, network traffic, and data flow between devices, simulating realistic scenarios of remote access and automation.

INTRODUCTION

Smart home automation is a technology-driven approach that enhances convenience, energy efficiency, security, and comfort within modern households. By integrating various IoT (Internet of Things) devices, smart home systems enable remote monitoring and control over home appliances, lighting, temperature, security cameras, and more, often through a centralised platform accessible via smartphones, tablets, or voice commands. These systems use sensors and automation algorithms to respond to environmental changes or user preferences, such as adjusting lighting when motion is detected, regulating the thermostat, or locking doors remotely. Smart home automation integrates IoT (Internet of Things) devices and advanced networking to create interconnected systems within households, enabling remote monitoring, control, and automation of appliances, security, and environmental settings.

As the demand for convenience, security, and energy efficiency grows, smart home technology has gained momentum, making homes more responsive and adaptive to their inhabitants' needs. In a typical setup, devices like smart thermostats, lights, locks, cameras, and sensors communicate over a network and are accessible through centralised controllers, often via mobile apps or voice-activated assistants like Alexa or Google Assistant. The context for this technology lies in the rising adoption of IoT in everyday life, which is driven by advances in wireless connectivity, AI, and data processing. The concept now plays a vital role in smart cities and sustainability initiatives, as it optimises resource usage and improves overall quality of life. The industry continues to innovate with new devices, standards, and protocols that make smart homes safer, more reliable, and adaptable to future technologies.

The primary purpose of smart home automation is to make homes more **efficient, secure, and adaptable** to user needs and preferences. By automating repetitive tasks like adjusting lights or regulating temperatures, it saves time and energy while also contributing to sustainability. Furthermore, smart home systems enhance **security and safety**, protecting against potential risks such as unauthorised access, fire, and environmental hazards. Another important purpose is to promote **personalised comfort and convenience**. Smart homes can adapt to individual routines, such as dimming lights in the evening or pre-heating the home before arrival, creating a seamless and comfortable living experience. Lastly, smart home automation plays a significant role in **health monitoring and elderly care** by enabling in-home safety systems that detect emergencies, supporting aging-in-place for seniors, and providing caregivers with crucial health and safety information remotely.

LITERATURE REVIEW

Smart Home Automation Systems (SHAS) have become increasingly popular in recent years, offering homeowners convenience, security, and energy efficiency. However, the current generation of wireless technology, such as 4G LTE, has speed, latency, and connectivity limitations, which can hinder the performance of smart home devices. With the introduction of 5G technology, there is a potential for significant improvements in the functionality and reliability of SHAS.(Bassey et al. 2024).Each automated system's primary objective is its ability to reduce human labour, effort, time, and mistakes brought on by carelessness. The objective of this study is to provide in-depth evaluation of the newly developed home automation system. Moreover, state-of-the-art(Aydin et al. 2024) home automation topologies such as ZigBee, Z-wave, Wi-Fi and Bluetooth are also discussed here. The authors are optimistic that this study would have a major impact on the present advances in home automation technology.(Habib et al.).

The concept of a "smart home" has gained attention recently, facing challenges like decision-making, secure IoT device identification, continuous connectivity, and privacy. Existing systems address some of the issues, but a truly effective smart home needs built-in security and analytical capabilities. This work proposes a novel smart home using Z-wave and Wi-Fi, with the Dynamic Analysis and Replanning(Giwa et al. 2024) Tool (DART) for maximum security. The system employs a support vector machine (SVM) classifier to determine device status ("OFF" or "ON"). The setup includes Raspberry Pi, a 5 V relay circuit, and sensors.Measures that can be taken by a store owner to deter fraudulent(Sayeduzzaman et al. 2023) activity include an electronic door lock system with a six-digit keypad which allows for safety and notifies the owner when someone or something approaches. Through the Internet, it enables remote device monitoring. The user can find and shut down the devices while away to reduce energy usage and management (Supriya et al.). This prototype had micro-controller-based smart security and automated home support with various sensors.

RESEARCH PLAN

The research plan for developing a smart home automation system involves designing, simulating, and implementing a comprehensive solution for monitoring and controlling IoT devices. This project's primary goal is to enable real-time management of home devices, such as lights, fans, and thermostats, using C++ programming. We will leverage Cisco Packet Tracer to simulate the network environment and device setup, and use CSV files to store and retrieve device data. The project begins with defining the scope, objectives, and key stakeholders while outlining the tasks and milestones to keep the development on track.For the requirement analysis, we'll identify the necessary devices and configurations for the smart home setup, planning for network design in Packet Tracer and defining how device status data will be handled in CSV format.

A robust system architecture will be created, including modules for CSV parsing, device control logic, and an interface for Packet Tracer integration. The core development phase focuses on simulating IoT devices in Packet Tracer and enabling communication with the C++ program. This involves setting up a home network topology and configuring devices to export status data to CSV files. C++ will be used to parse these files, updating device states in real-time and implementing control functions, such as turning devices on/off and adjusting settings based on environmental conditions. For the user interface, a simple GUI in C++ or a C++ library like Qt will allow users to interact with devices and view real-time status updates. The GUI will also display graphical data, such as bar graphs or charts, representing device statuses (e.g., Off, Low, High) through a plotting library, like Matplotlib C++. Documentation of development stages, deployment preparations, and stakeholder feedback sessions will be essential final steps to ensure usability, functionality, and reliability in this smart home automation project. This structured plan aims to create an intuitive and efficient automation system, meeting user needs and supporting real-time control over simulated IoT devices.

SL. No	Description	07/10/2024-11/10/2024	12/10/2024-16/10/2024	17/10/2024-20/10/2024	21/10/2024-29/10/2024	30/10/2024-05/11/2024	07/10/2024-10/11/2024
1.	Problem Identification						
2.	Analysis						
3.	Design						
4.	Implementation						
5.	Testing						
6.	Conclusion						

Fig.1-Timeline Chart

Day 1: Project Initiation and Planning

- Define the project's scope and objectives, focusing on developing an efficient and user-friendly smart home automation system.
- Conduct preliminary research to gather insights on IoT devices, automation protocols, and best practices in smart home integration.
- Identify key stakeholders (e.g., developers, designers, users, and security experts) and establish communication channels for efficient collaboration.
- Create a comprehensive project plan with detailed tasks and milestones for each development phase, ensuring all team members have a clear understanding of the project timeline.

Day 2: Requirement Analysis and System Design

- Conduct a thorough requirement analysis to identify user needs, essential functionalities, and automation capabilities for the smart home system.
- Define system specifications, including device compatibility, communication protocols (e.g., Wi-Fi, Zigbee, Bluetooth), and security requirements.
- Develop detailed architecture and system design, including user interface mock-ups, device connection layouts, and data flow diagrams.
- Confirm hardware and software requirements, ensuring compatibility with the chosen IoT devices and integration framework.

Day 3: Core Development and Device Integration

- Begin coding the core modules of the system, including device discovery, control functions, and communication protocols.
- Implement features for adding, removing, and managing connected devices (e.g., lights, fans, thermostats) within the home automation network.
- Ensure reliable communication between devices and central control via a responsive and stable backend, and integrate security measures for data protection.
- Conduct unit testing on each component to verify functionality and smooth interaction between devices.

Day 4: GUI Design and Prototyping

- Design and develop an intuitive and user-friendly interface for the smart home system, allowing users to easily control and monitor their devices.
- Implement real-time data visualisation for device status, energy usage, and system notifications.
- Ensure that the GUI is responsive and optimised for multiple device types (e.g., smartphones, tablets, desktops).
- Conduct iterative testing and usability evaluations, making necessary adjustments to improve user experience and functionality.

Day 5: Automation Scenarios and Testing

- Define and develop automation scenarios (e.g., scheduling, motion-triggered actions, energy-saving modes) to enhance the system's functionality.
- Integrate machine learning or rule-based algorithms to enable smart decision-making for predictive automation.
- Conduct thorough testing of various scenarios to ensure reliability and accurate responses to user inputs and environmental changes.
- Fine-tune system performance based on test results to maximise responsiveness and efficiency.

Day 6: Documentation, Deployment, and Feedback

- Document the entire development process, including architectural choices, code structure, and key decisions made.
- Prepare for deployment, ensuring the system is compatible with real-world conditions and meets industry standards for IoT security and data privacy.
- Deploy the system for beta testing and organise feedback sessions with stakeholders and end-users to gather insights on usability, performance, and potential improvements.

METHODOLOGY

The methodology for implementing a smart home automation system using Cisco Packet Tracer and C++ involves a structured approach across three main phases: system design, network simulation, and code integration. In the system design and planning phase, the project's overall architecture is established, defining the types of devices and functionalities to be included, such as lights, thermostats, security cameras, motion detectors, and door locks. Each device's role and interactions within the network are outlined, specifying control mechanisms such as automatically turning on lights when motion is detected or adjusting the thermostat based on temperature changes. In the network simulation phase, Cisco Packet Tracer is used to create a virtual smart home network. This network setup includes routers, switches, and IoT-enabled devices that represent each component in the smart home.

Network protocols, IP addressing, and communication channels between devices are configured to simulate real-world connectivity. Packet Tracer’s tools help visualise data flow, device communication, and response to various scenarios, such as sending alerts or triggering devices remotely. The final code integration phase involves programming device-specific behaviours in C++. Each device is represented by a class, with abstract methods for common actions like `turnOn`, `turnOff`, and specific functions, such as temperature sensing or motion detection. The C++ code defines how devices respond to inputs, process commands, and communicate with the network, as established in Packet Tracer. This integration allows the simulation to perform automated tasks, giving a realistic demonstration of a smart home automation system’s core functions, including control, monitoring, and remote access, in a virtual environment.

Using software like Excel or a data visualisation tool, the CSV file is imported, and a bar graph is plotted to show each device's activity across scenarios. This bar graph provides an at-a-glance view of device behaviour over time, highlighting patterns such as peak usage times or active security monitoring periods. The methodology effectively combines CSV data management with visual analysis, helping evaluate automation patterns and system efficiency in the simulated smart home setup.

RESULTS

Figure 1 depicts the dashboard view after opening Cisco Packet Tracer. Begin by dragging the "Home Gateway" from the available sources below, placing it in the DLC100 section. Next, review the programming of the gateway, noting the SSID, and verify that all attributes are set to their correct positions for proper functionality.

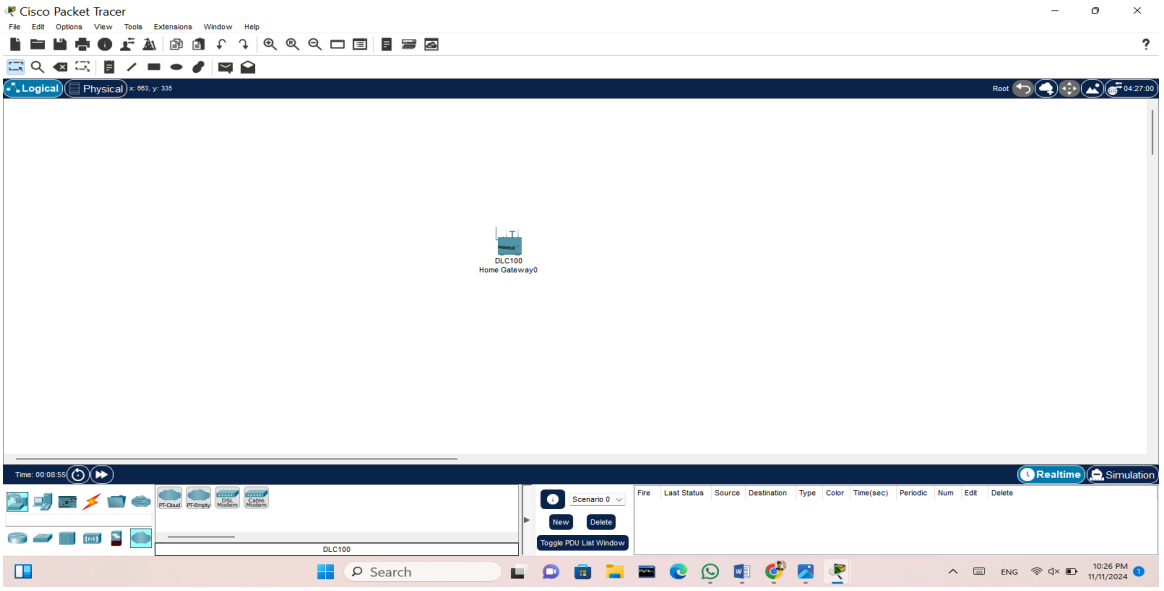


Fig.1: Dashboard of Packet Tracer

Figure 2 demonstrates the process of locating a fan using the search bar at the bottom of the dashboard. Drag one fan from the search results, then repeat the process to add the remaining seven fans. Once all fans are placed, they will automatically connect to the gateway, establishing a connection as shown in the image.

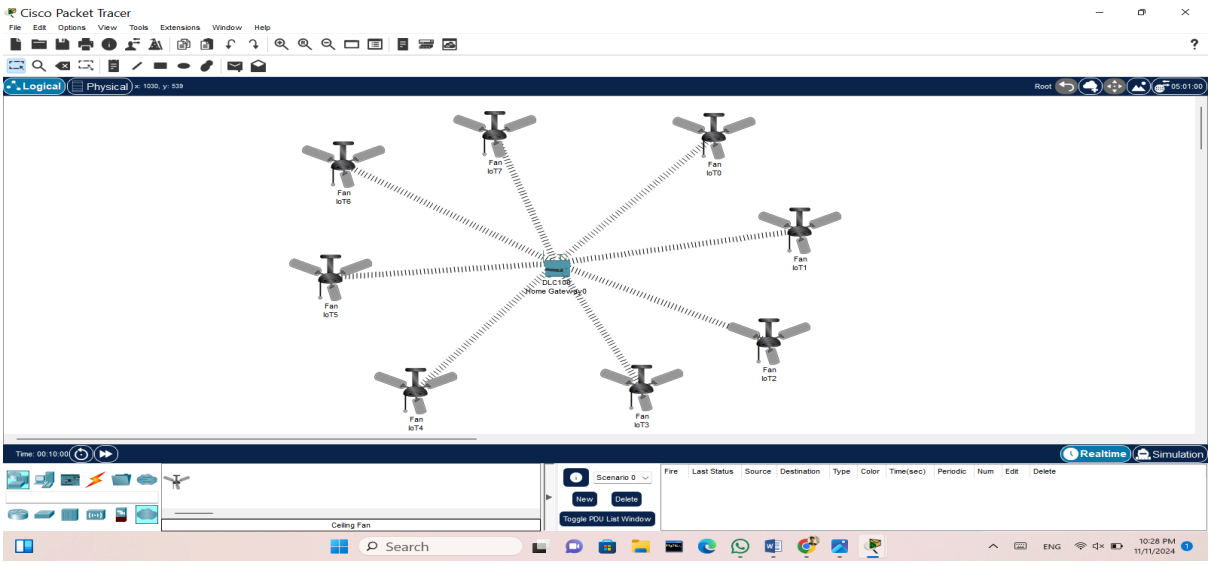


Fig.2: IoT Fans are Connected

Figure 3 illustrates the steps to connect a smartphone to the network. Begin by selecting a smartphone from the search bar at the bottom of the dashboard. To connect the smartphone to the gateway, retrieve the SSID from the Home Gateway's programming section. Copy the SSID and paste it into the smartphone's wireless programming settings. This action will establish the connection.

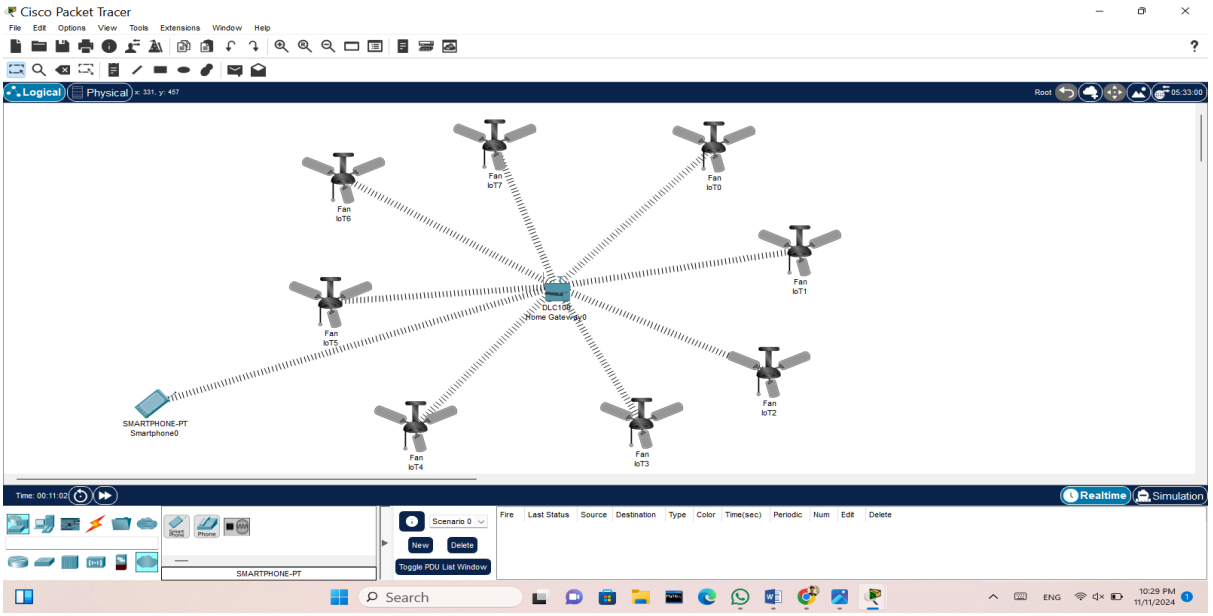


Fig.3: Smart phone is connected to Gateway

Figure 4 shows the options available when you click on the smartphone icon. Several domains are displayed, including "Physical," "Config," and "Desktop." Navigate to the "Desktop" section, where you will find various applications such as Bluetooth, IP Config, and IoT Monitor. Select "IoT Monitor" and tap on it to proceed.

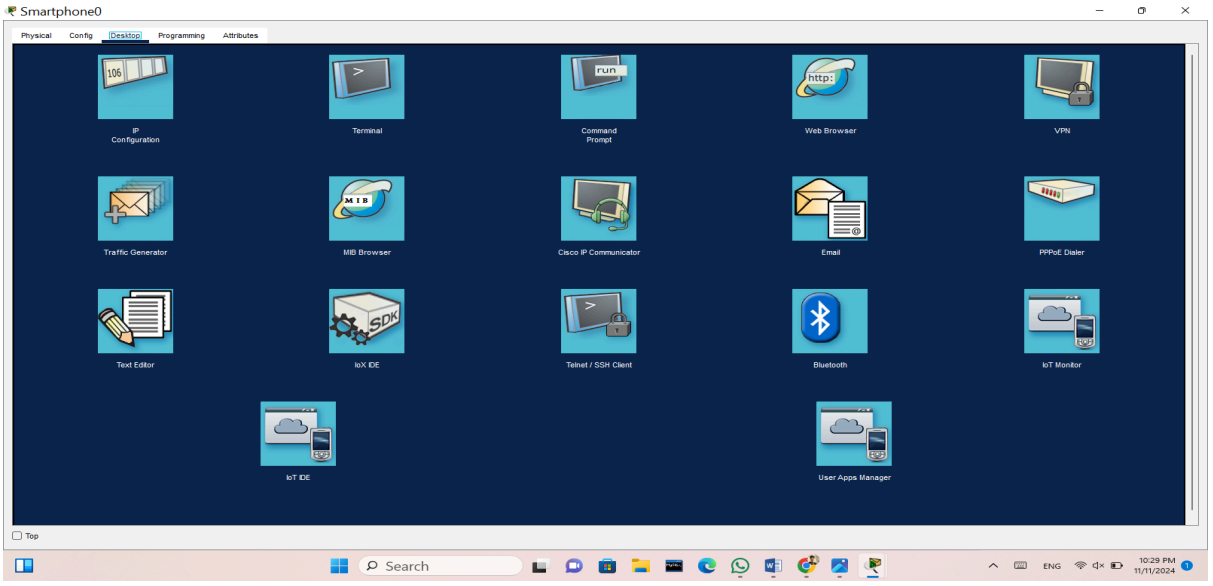


Fig.4: Desktop of Smart Mobile

Figure 5 illustrates the login screen of the IoT monitoring application designed for smartphones and desktop use. The page prompts the user to enter a username and password, both of which are preset to "admin" by default. Once these credentials are entered, the user needs to click the "Login" button to access the application. This initial login step is essential for verifying user access and maintaining system security.

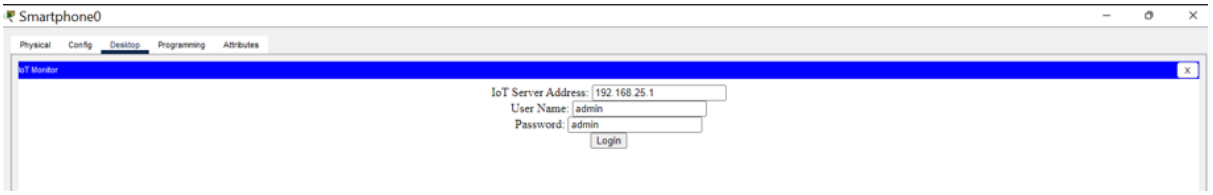


Fig.5:Login Page for Mobile

Figure 6 shows the screen displayed after logging into the IoT monitoring application. This page lists all devices connected to your smartphone, allowing you to verify their connection status. Check to ensure each device is connected properly, and confirm that all devices are also linked to the gateway for seamless monitoring.

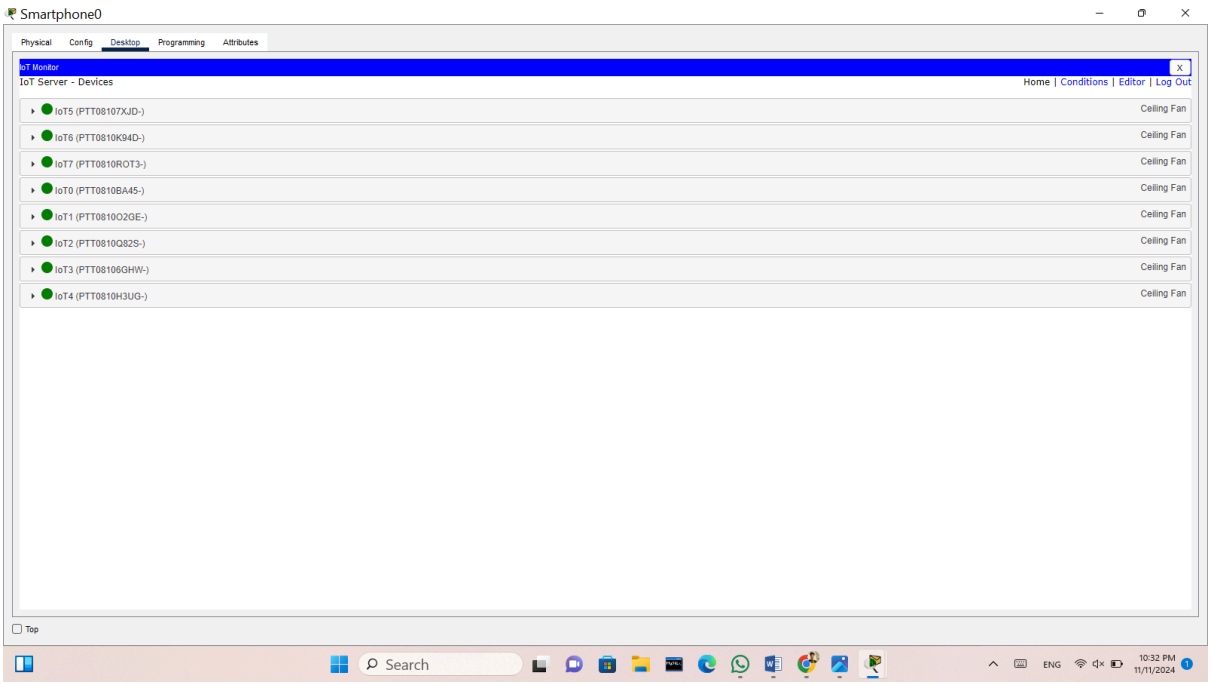


Fig.6: Smartphone connected Devices

Figure 7 provides an overview of the status of IoT devices, displaying all device connections along with options for different settings, such as Off, Low, and High. Select the desired setting for each IoT device based on your requirements. While choosing a setting, verify that the device is connected to ensure changes take effect. After making a selection, you can adjust the settings as needed.

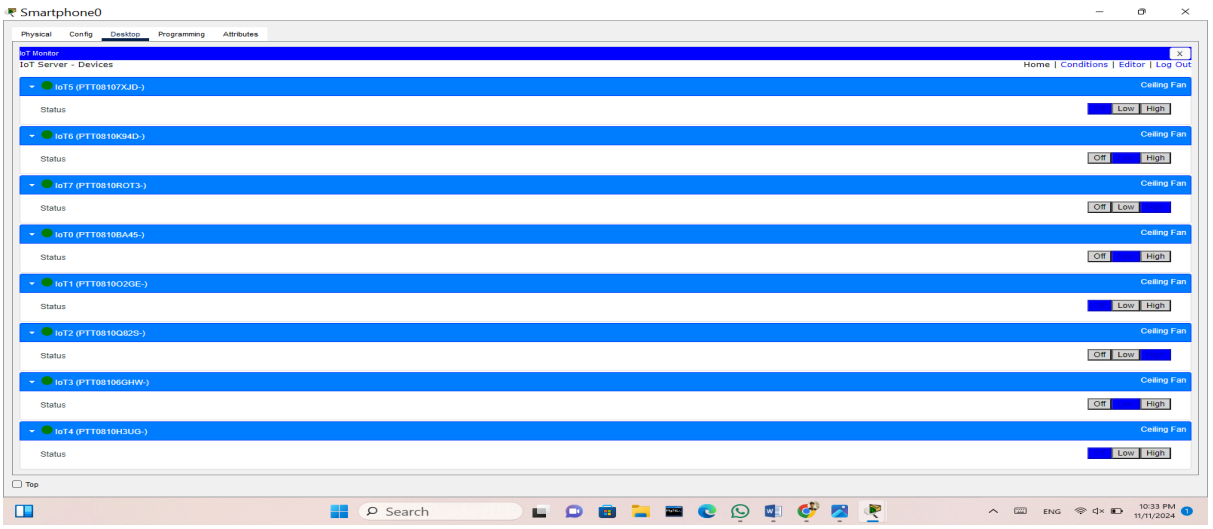


Fig.7: Status of Devices

Figure 8 presents the final stage of IoT automation, displaying the operational status of all connected IoT devices. The figure highlights that not all fans are running at the same speed, with some variation observed. Specifically, two lines indicate that certain fans are set to High, one line shows a fan set to Low, and the remaining fans are in the Off position. This illustration represents the completed automation setup, confirming each device's status and performance level.

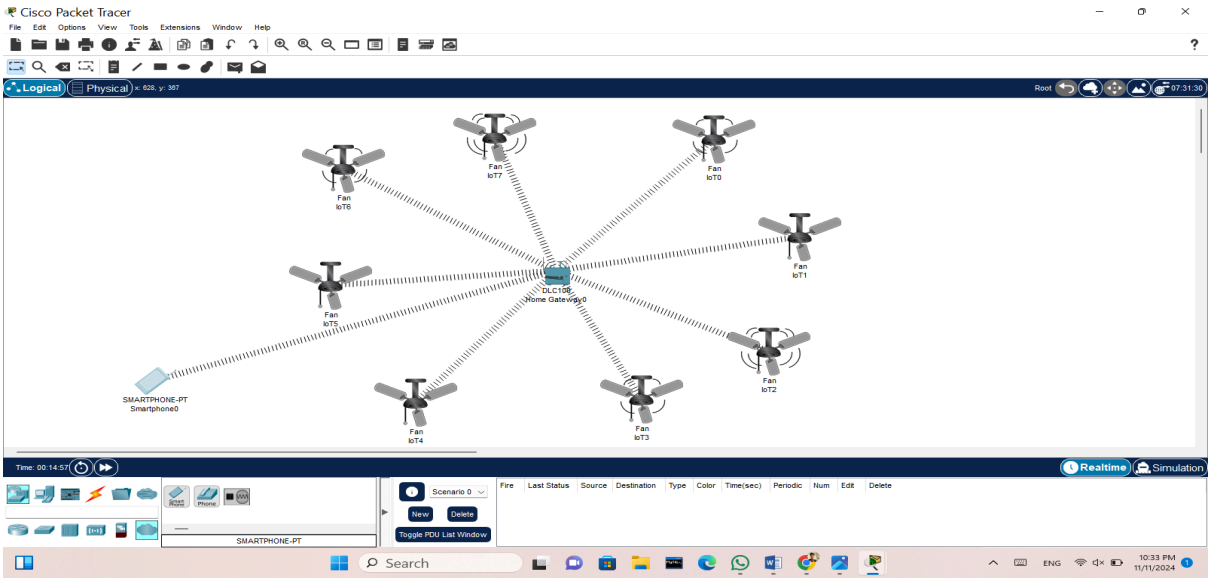
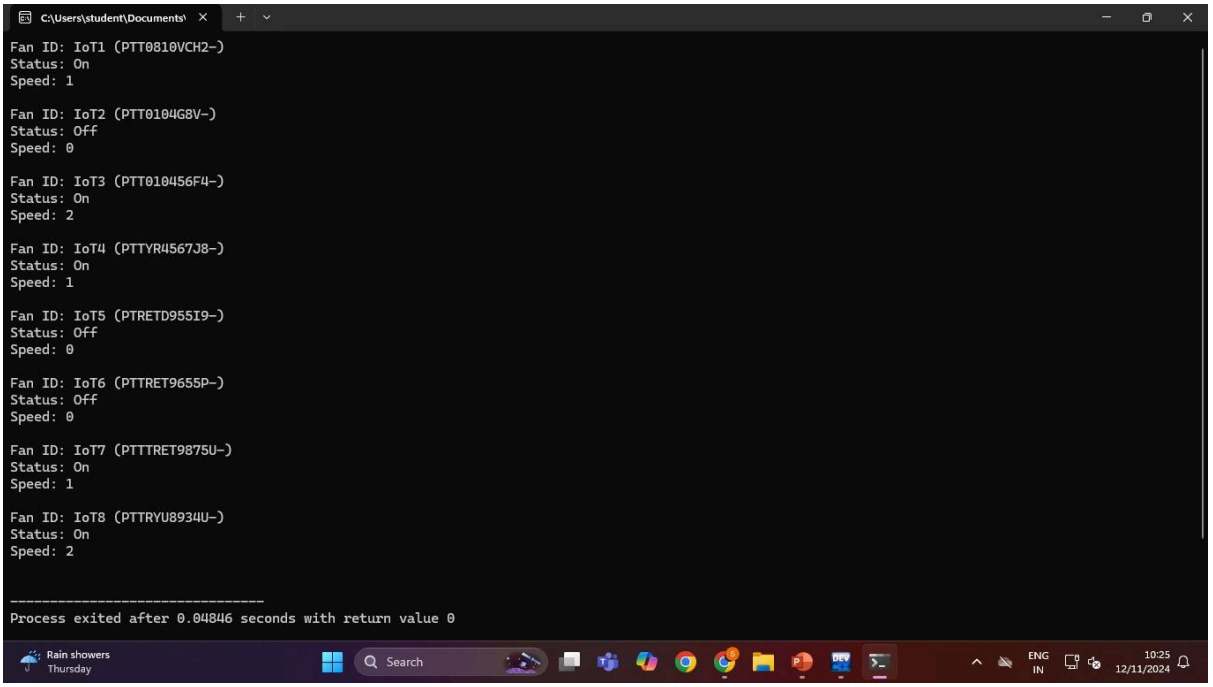


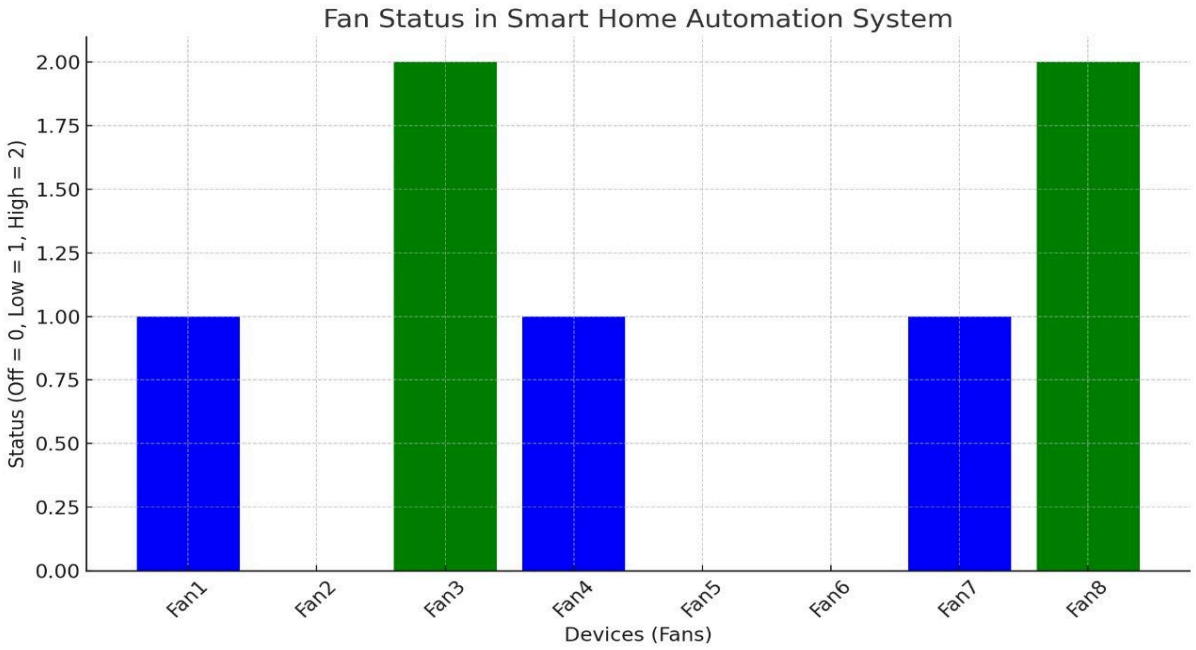
Fig.8: Final Status for Iot Automation

OUTPUT



GRAPH

The graph represents the operational status of fans across eight IoT-connected devices. It uses a numerical code where 0 indicates "Off," 1 represents "Low" speed, and 2 signifies "High" speed. Each fan's status is displayed along the Y-axis, while the X-axis lists each device (fan) individually. The graph's scale is set so that each unit equals 0.25 cm, providing a clear, proportional visualisation of fan performance across the devices. This layout allows for quick assessment of the fan status in the IoT system at a glance.



CONCLUSION

Smart home automation is revolutionising the way we live by seamlessly integrating technology into our daily routines. By connecting household devices and appliances to a central network, it enables users to remotely monitor, control, and automate functions such as lighting, security, heating, and entertainment. This connectivity not only enhances convenience but also promotes energy efficiency and cost savings, as systems can be optimised to use resources more effectively.

Additionally, smart home automation offers significant security benefits, providing real-time monitoring and alerts to keep homes safer. With voice commands, smartphone apps, and AI-driven assistants, these systems create a highly personalised and intuitive environment. While the technology is still evolving, smart homes represent a major step toward more responsive, efficient, and comfortable living spaces.

IMPLEMENTATION

C++ CODE:

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
class Fan {
public:
    string id;
    bool isOn;
    int speed;
    Fan(string id) {
        this->id = id;
        this->isOn = false;
        this->speed = 0;
    }
    void turnOn(int speed) {
        isOn = true;
        this->speed = speed;
    }

    void turnOff() {
        isOn = false;
        speed = 0;
```

```
    }
    void printStatus() const {
        cout << "Fan ID: " << id << endl;
        cout << "Status: " << (isOn ? "On" : "Off") << endl;
        cout << "Speed: " << (isOn ? to_string(speed) : "0") << endl;
        cout << endl;
    }
};
int main() {
    vector<Fan> fans;
    fans.push_back(Fan("IoT1 (PTT0810VCH2-)"));  // Fan 1
    fans.push_back(Fan("IoT2 (PTT0104G8V-)"));    // Fan 2
    fans.push_back(Fan("IoT3 (PTT010456F4-)"));   // Fan 3
    fans.push_back(Fan("IoT4 (PTTYR4567J8-)"));   // Fan 4
    fans.push_back(Fan("IoT5 (PTRETD955I9-)"));   // Fan 5
    fans.push_back(Fan("IoT6 (PTTRET9655P-)"));   // Fan 6
    fans.push_back(Fan("IoT7 (POTRET 9875-)"));  // Fan 7
    fans.push_back(Fan("IoT8 (POTTERY 8934-)"));   // Fan 8
    // Setting specific fan statuses based on requirements
    fans[0].turnOn(1); // Fan 1: Low
    fans[1].turnOff(); // Fan 2: Off
    fans[2].turnOn(2); // Fan 3: High
    fans[3].turnOn(1); // Fan 4: Low
    fans[4].turnOff(); // Fan 5: Off
    fans[5].turnOff(); // Fan 6: Off
```

```
    fans[6].turnOn(1); // Fan 7: Low
    fans[7].turnOn(2); // Fan 8: High
    // Print the status of all fans
    for (const Fan& fan : fans) {
        fan.printStatus();
    }
    return 0;
}
```

REFERENCES

- Ghoul Y, Naifar O (2024) IoT based applications for healthcare and home automation. Multimed Tools Appl 83:29945–29967. https://doi.org/10.1007/s11042-023-16774-z
- FakhrHosseini S, Lee C, Lee SH et al (2024) A taxonomy of home automation: Expert perspectives on the future of smarter homes. Inf Syst Front. https://doi.org/10.1007/s10796-024-10496-9
- Peng Z, Li X, Yan F (2020) An Adaptive Deep Learning Model for Smart Home Autonomous System, International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), Vientiane, Laos, 2020, pp. 707–710, https://doi.org/10.1109/ICITBS49701.2020.00156
- Benadda B, Benabdellah A (2022) Hardware Design and Integration of Low-cost Edge AI Smart Power Management and Home Automation, International Conference on Artificial Intelligence of Things (ICAIoT), Istanbul, Turkey, 2022, pp. 1–5, https://doi.org/10.1109/ICAIoT57170.2022.10121892
- Fakhruldeen HF, Saadh MJ, Khan S et al (2024) Enhancing smart home device identification in WiFi environments for futuristic smart networks-based IoT. Int J Data Sci Anal. https://doi.org/10.1007/s41060-023-00489-3
- Stalin GPA, Anand S (2022) Intelligent Smart Home Security System: A Deep Learning Approach, IEEE 10th Region 10 Humanitarian Technology Conference (R10-HTC), Hyderabad, India, 2022, pp. 438–444, https://doi.org/10.1109/R10-HTC54060.2022.9929516
- Aggarwal S, Sharma S, Voice Based Deep Learning Enabled User Interface Design For Smart Home Application System, (2021) 2nd International Conference on Communication, Computing and Industry 4.0 (C2I4), Bangalore, India, 2021, pp. 1–6, https://doi.org/10.1109/C2I454156.2021.9689435