

Smart Condominium Simulation Using Low Code Programming

By

TEE YU XUAN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMMUNICATIONS
AND NETWORKING

Faculty of Information and Communication Technology

(Kampar Campus)

FEBRUARY 2025

COPYRIGHT STATEMENT

© 2025 Tee Yu Xuan. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Technology (Honours) Communications and Networking at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Ts Dr. Goh Hock Guan, who has given me this bright opportunity to engage in the Smart Condominium Simulation project. It is my first step to establish a career in smart condominium and IoT field. A million thanks to you.

Furthermore, I would like to give a very big thanks to my dearest friend in my University Life, See Keng Lek, for his patience, assistance, guidance of the Node-RED system when encountered issue, and helping me reformatting my laptop. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

This project provides a smart condominium simulation architecture with a goal to reduce the gap between user interactions, real-time sensor data, and automated functionalities within the simulated environment. This architecture focusses the accessibility and user-friendliness by using low-code programming.

One of the key contributions is the development of a scalable smart condominium model. The platform's modular design allows the combining of multiple devices, sensors, and functionality, which ensuring its ability to grow and develop with the development of smart condominium technology. Additionally, the use of Scalable Vector Graphics (SVG) helps to design pleasant and flexible simulation object representations.

Then, another way to point out the user-friendliness is the applying of the low-code programming platform Node-RED. With this platform, individuals with different levels of coding experience can simulate out the functions and interactions between the smart home devices. It brings the ability to instantly interact with the simulation, operate virtual equipment, establish the automation rules based on simulated sensor data, and track the effects of their actions, the platform places a high priority on an interactive user experience.

In the nutshell, this project enhances the field by developing an architecture for smart home simulation that is more flexible. This platform helps to develop a more sustainable and effective smart home technologies to explore the potential of smart homes.

Area of Study (Minimum 1 and Maximum 2): Modelling, Simulation

Keywords (Minimum 5 and Maximum 10): Smart Condominium, Simulation, Low-Code Programming using Node-RED, Scalable Vector Graphics (SVG), Smart IoT Devices.

TABLE OF CONTENTS

TITLE PAGE	i
DECLARATIN OF ORIGINALITY	ii
ACHKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENT	v
LIST OF FIGURES	viii
LIST OF TABLES	xii
LIST OF ABBREVIATIONS	xiii
CHAPTER 1 INTRODUCTION	1
1.1 Motivation	1
1.2 Problem Statement	2
1.3 Project Objectives	3
1.4 Project Scope	5
1.5 Contribution	7
1.6 Report Organization	8
 CHAPTER 2 LITERATURE REVIEW	 9
2.1 Review of the Technologies	9
2.1.1 Firmware/OS	9
2.1.2 Database	10
2.1.3 Programming Language	11
2.1.4 Summary of the Technologies Review	12
2.2 Review of the Existing System/Application	14
2.2.1 Smart Home Simulation using A.I and Java Programming Language	14
2.2.1.1 Limitation of Smart Home Simulation Using A.I and Java Programming Language	15
2.2.2 Smart Home Simulation Using OpenSHS Architecture	16
2.2.2.1 Limitation of Smart Home Simulation Using OpenSHS Architecture	19
2.2.3 Smart Home Simulation Using Map Editor and WSN Simulator	20

2.2.3.1 Limitation of Smart Home Simulation Using Map Editor and WSN Simulator	21
CHAPTER 3 SYSTEM METHODOLOGY	22
3.1 System Development Models	22
3.1.1 System Development Model 1: Agile Model	22
3.1.2 System Development Model 2: Waterfall Model	23
3.1.3 System Development Model 3: Iterative Model	25
3.1.4 System Development Model 4: Spiral Model	26
3.1.5 Selected Model	27
3.2 System Design	27
3.2.1 Hardware	27
3.2.2 Software	29
3.3 Functional Requirements	33
3.4 Project Milestones	34
3.5 Concluding Remark	36
CHAPTER 4 SYSTEM DESIGN	37
4.1 System Architecture	37
4.2 Functional Modules in the System	38
4.3 GUI Design	39
4.4 Concluding Remark	48
CHAPTER 5 SYSTEM IMPLEMENTATION	49
5.1 Software Setup	49
5.1.1 Software Setup – Node-RED	49
5.1.2 Software Setup - InfluxDB	52
5.1.3 Software Setup (Mobile) – IoT MQTT Panel	55
5.1.4 Software Setup (Mobile) – Telegram	58
5.2 Setting and Configuration	61
5.2.1 Setting and Configuring InfluxDB node in Node-RED	61
5.2.2 Setting and Configuring MQTT Panel Application and node in Node-RED	64
5.2.3 Setting and Configuring Telegram in Node-RED	70
5.3 System Operation	71

5.3.1 System Operation of Main Door	71
5.3.2 System Operation of Living and Kitchen	74
5.3.3 System Operation of Master Room	77
5.3.4 System Operation of Bedrooms	79
5.4 Concluding Remark	80
 CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	 81
6.1 System Testing and Performance Metrics	81
6.2 Testing Setup and Result	82
6.2.1 Main Door	82
6.2.2 Living Hall	84
6.2.3 Kitchen	86
6.2.4 Master Room	88
6.2.5 Bedrooms Control Tab	90
6.2.6 Bedroom 1 and Bedroom 2	91
6.3 Project Challenges	93
6.4 Objective Evaluation	94
6.5 Concluding Remark	94
 CHAPTER 7 CONCLUSION AND RECOMMENDATION	 95
7.1 Conclusion	95
7.2 Recommendation	96
 REFERENCES	 97
POSTER	99

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1.2.1	MongoDB Logo	9
Figure 2.2.2.1	Author's design phases	15
Figure 2.2.2.2	Author's simulation phase	16
Figure 2.2.2.3	Author's fast-forwarding features	16
Figure 3.1.1	Agile Model	23
Figure 3.1.2	Waterfall Model	24
Figure 3.1.3	Iterative Model	25
Figure 3.1.4	Spiral Model	26
Figure 3.2.1.1	Acer Nitro 5 AN515-46-R20B	27
Figure 3.2.1.2	Apple iPhone 13	28
Figure 3.2.1.3	Xiaomi 14T	28
Figure 3.2.2.1	Node-RED Logo	29
Figure 3.2.2.2	Interface of Node-RED	29
Figure 3.2.2.3	Inkscape Logo	30
Figure 3.2.2.4	Interface of Inkscape	30
Figure 3.2.2.5	InfluxDB Logo	31
Figure 3.2.2.6	Interface of InfluxDB	31
Figure 3.2.2.7	IoT MQTT Panel Application Logo	32
Figure 3.2.2.8	Telegram Logo	32
Figure 3.2.2.9	Telegram BotFather	33
Figure 4.1.1	System Architecture Diagram	37
Figure 4.3.1	Original Floorplan for Smart Condominium	39
Figure 4.3.2	Node-RED dashboard of Main Door	40
Figure 4.3.3	Colour of Door Handle turns green when unlock with correct password	40
Figure 4.3.4	Node-RED dashboard for Living and Kitchen Tab (Before)	41
Figure 4.3.5	Node-RED dashboard for Living and Kitchen Tab (After)	41
Figure 4.3.6	"Where To" input field with hint	42
Figure 4.3.7	Node-RED dashboard for Master Room (Before)	42
Figure 4.3.8	Node-RED dashboard for Master Room (After)	43

Figure 4.3.9	Node-RED dashboard for Bedrooms Tab (Before)	43
Figure 4.3.10	Node-RED dashboard for Bedrooms Tab (After)	44
Figure 4.3.11	User Interface of IoT MQTT Panel Application (Living Hall)	45
Figure 4.3.12	User Interface of IoT MQTT Panel Application (Master Room)	46
Figure 4.3.13	Interface of Telegram Chatbot	47
Figure 5.1.1.1	Type “node-red” to execute Node-RED	49
Figure 5.1.1.2	Node-RED has successfully executed	50
Figure 5.1.1.3	Node-RED Flow Editor Panel	50
Figure 5.1.1.4	Palette to be installed in Node-RED	51
Figure 5.1.2.1	Download InfluxDB in Windows	52
Figure 5.1.2.2	Command Prompt running InfluxDB	52
Figure 5.1.2.3	InfluxDB server	53
Figure 5.1.2.4	API Token	53
Figure 5.1.2.5	InfluxDB Dashboard	54
Figure 5.1.3.1	IoT MQTT Panel Application in Apple Play Store	55
Figure 5.1.3.2	Press the “+” Symbol to add new connections	56
Figure 5.1.3.3	Connection for Smart Condo Simulation System	57
Figure 5.1.4.1	BotFather with blue tick beside	58
Figure 5.1.4.2	“/start” to activate BotFather	59
Figure 5.1.4.3	“/newbot” and follows the instruction to create new bot	60
Figure 5.2.1.1	Flows for InfluxDB to get data of Temperature and Humidity	62
Figure 5.2.1.2	Configuration Setting for influxdb node	62
Figure 5.2.1.3	Configuration for Temperature Measurement	63
Figure 5.2.1.4	Configuration for Humidity Measurement	63
Figure 5.2.1.5	Temperature Measurement shown in InfluxDB	64
Figure 5.2.1.6	Humidity Measurement shown in InfluxDB	64
Figure 5.2.2.1	Flows for “mqtt in” connection with json and switch node	65
Figure 5.2.2.2	Flow for “mqtt in” connected with json and directly without switch node	65
Figure 5.2.2.3	Configuration for mqtt-broker node	66

Figure 5.2.2.4	Configuration for “mqtt in” node	67
Figure 5.2.2.5	Configuration for switch node	67
Figure 5.2.2.6	Panel configuration with switch node linked and without switch node linked	69
Figure 5.2.3.1	Flow for Telegram “callback_query” and Telegram sender node	70
Figure 5.2.3.2	Configuration for Telegram bot node	70
Figure 5.3.1.1	Door Locked	71
Figure 5.3.1.2	Door Unlocked after entering the correct password	72
Figure 5.3.1.3	Notification shows incorrect password	72
Figure 5.3.1.4	Camera and notification that captured image send via Telegram Bot	73
Figure 5.3.2.1	Dashboard that shows floorplan and control tab of Living and Kitchen	74
Figure 5.3.2.2.	Status of environment or Temperature of some Smart Furniture	74
Figure 5.3.2.3	Type “main/master/bed” to jump to specific tab	75
Figure 5.3.2.4	Control button in IoT MQTT Panel Application	75
Figure 5.3.2.5	Telegram warning message or callback query	76
Figure 5.3.3.1	Dashboard that shows the floorplan and control tab of Master Room	77
Figure 5.3.3.2	Status of environment and Temperature of Smart Air-Conditioner	78
Figure 5.3.3.3	Control Button in IoT MQTT Panel for Master Room	78
Figure 5.3.4.1	Dashboard that shows the floorplan and control tab of Bedrooms	79
Figure 5.3.4.2	Status slot to show each Room status and availability of washroom	79
Figure 6.2.1.1	Captured Image from Webcam send by Telegram Bot	83
Figure 6.2.1.2	Dashboard notification when enter incorrect password and when door is locked	83
Figure 6.2.2.1	Notification about Sliding Door sends by Telegram Bot	85
Figure 6.2.2.2	Control buttons in IoT MQTT Panel Application	85

Figure 6.2.3.1	Telegram Notification for Smoke Sensor	87
Figure 6.2.3.2	Telegram Notification for Refrigerator Powered OFF	87
Figure 6.2.3.3	Telegram Notification for Oven Overheated	87
Figure 6.2.3.4	Telegram Notification and Mode Selection for Washing Machine	87
Figure 6.2.4.1	Colour of Light changes according to colour picker	89
Figure 6.2.4.2	Control Buttons in IoT MQTT Panel Application for Master Room	89
Figure 6.2.5.1	Washroom Status Display	90
Figure 6.2.6.1	Dashboard of Bedrooms	92

LIST OF TABLES

Table Number	Title	Page
Table 2.1.4.1	Summary of Firmware/OS	11
Table 2.1.4.2	Summary of Database	11
Table 2.1.4.3	Summary of Programming Language	12
Table 2.2.2.1	Dataset that authors recorded	17
Table 3.2.1.1	Specification of Laptop	26
Table 3.2.1.2	Specification of Mobile Phone	27
Table 3.3.1	Description of Functional Requirements	32
Table 3.4.1	Milestone of FYP 1	33
Table 3.4.2	Milestone of FYP 2	34
Table 6.1.1	Description and Expected Outcome of Key Testing Metrics	81
Table 6.2.1	System Testing and Result for Main Door	82
Table 6.2.2	System Testing and Result for Living Hall	84
Table 6.2.3	System Testing and Result for Kitchen	86
Table 6.2.4	System Testing and Result for Master Room	88
Table 6.2.5	System Testing and Result for Bedrooms Control Tab	90
Table 6.2.6	System Testing and Results for Bedroom 1 and Bedroom 2	91

LIST OF ABBREVIATIONS

<i>IoT</i>	Internet of Things
<i>SME</i>	Small and Medium Enterprise
<i>AI</i>	Artificial Intelligence
<i>API</i>	Application Program Interface
<i>GUI</i>	Graphical User Interface
<i>PM</i>	Pressure Mat
<i>PIR</i>	Passive Infrared
<i>XML</i>	Extensible Markup Language
<i>LOC</i>	Loss of Consciousness
<i>SVG</i>	Scalable Vector Graphic
<i>OS</i>	Operating System
<i>DBMS</i>	Database Management System
<i>RDBMS</i>	Relational Database Management System
<i>TSDB</i>	Time Series Database
<i>OOP</i>	Object-Oriented Programming Language
<i>SDLC</i>	Software Development Life Cycle
<i>API</i>	Application Program Interfaces
<i>MQTT</i>	Message Queuing Telemetry Transport
<i>UI</i>	User Interface
<i>QoS</i>	Quality of Service

CHAPTER 1

Project Background

This project, "Smart Condominium Simulation using Low Code Programming," is motivated by the desire to the way to engage with our living spaces in the age of the digital transformation. With the development of the technology being more and more advances, our reliance on internet devices has increased, and the concept of smart homes has become as a key solution to enhance our living standards, efficiency of energy, and security. However, the development and simulation of smart home environments have often been constrained by technical barriers, requiring extensive programming knowledge and expertise. With the growing demand for intuitive and accessible smart home solutions, this research aims to bridge this gap by leveraging low code programming and scalable simulation techniques. By enabling users without extensive technical backgrounds to design, simulate, and optimize smart home models, we empower a broader audience to innovate and create personalized living experiences.

1.1 Motivation

The main motivation of this project, "Smart Condominium Simulation using Low Code Programming," is to allow individuals and businesses, especially SMEs (Small and Medium Enterprise), with an easy and cost-effective for smart condominium development and simulation. The management and innovation in smart home technology have become important for staying competitive and relevant in the market. However, there are many barriers such as high costs and technical complexity when trying to implement smart home solutions. This will reduce smart home technology's potential and hurt its ability to compete.

This project aims to remove these obstacles and open up smart condominium development and virtual simulation to a more audience by developing a Smart Condominium Simulation Platform using low-code programming techniques. Which it can helps individuals that wanted to innovate and enhance their living spaces in a smart home environment.

The initiative is also driven by the possibility of improving the competitiveness, cost-effectiveness, and operational efficiency of enterprises and individuals operating in the household automation market. The project goal is to allows users to make well-informed decisions and motivation of the innovation in the smart home market throughout the virtual

simulation and development of smart condominiums and enhancing decision-making processes through living models.

1.2 Problem Statement

Although smart condominium simulation or smart home simulation platforms have improved a lot, there is still lack of seamless connection between automated capabilities in the simulated environment and user interactions as well as real-time sensor data. This is particularly relevant when simulating complicated situations that's involves multiple linked devices and user behavior patterns.

The traditional smart home simulation or smart condominium simulation systems usually focus on static settings and device representations. They are lacking the process of real-time sensor data that indicating the changes in the home or respond quickly according to user interactions. This disadvantage makes the simulation become more difficult to fully explore the potential of IoT smart technologies and to faithfully represent real-world smart home or smart condominium experiences.

The discrepancy between user interactions and simulated responses can lead to a less realistic and informative user experience. Users may not be able to fully understand how different IoT smart devices interact with each other or how their actions influence the simulated environment. This limits the value of these simulations for user education, design exploration, and testing of new smart home or smart condominium functionalities.

A less realistic and informative user experience may result from the differences between actual user experiences and simulated responses. It's possible that users won't be able to fully understand how different smart home or smart condominium appliances work together or how their actions will affect the simulated environment. This will reduce the value of the simulations for testing new features of smart home or smart condominium, design exploration, and user education.

Therefore, connecting real-time sensor data to user interactions in smart condominium simulation platforms is a major challenge. Overcoming this challenge leads to the development of smart condominium simulations that are more lively, realistic, and interactive. By allowing people to explore the possibilities of smart condominium technologies and make knowledgeable decisions regarding their own smart condominium structure, these enhanced

simulations could lead to a more practical, effective, and user-friendliness smart condominium experience.

1.3 Project Objectives

This project focuses on the development of a scalable smart condominium simulation model, which aims to integrate a comprehensive range of interconnected devices and functionalities. The primary objective is to design a simulation that is not only functional but also adaptable to future enhancements. Scalability is a key consideration, ensuring that as new smart condominium scenarios emerge, additional components can be seamlessly integrated into the existing model. This flexibility is vital to maintaining the relevance and utility of the simulation over time, allowing for continuous updates and modifications as technology and user needs evolve.

A significant component of this project is the application of Scalable Vector Graphics (SVG) to enhance the visual representation of the smart condominium elements within the simulation. SVGs offer numerous advantages that make them ideal for this purpose. One of the most notable benefits is scalability; SVG graphics can be resized without any loss of quality, making them perfect for visual components that need to be displayed on various screen sizes and resolutions. This ensures that the simulation remains visually consistent and clear, regardless of the device or platform being used. Additionally, SVGs provide a high degree of flexibility, enabling the design of complex and dynamic objects. This capability allows for the creation of visually appealing and informative representations of the smart condominium components, enhancing user engagement and understanding.

Moreover, SVGs are platform-independent, which means they are compatible with a wide range of devices and operating systems. This characteristic ensures that the smart condominium simulation tool can be accessed and utilized across different platforms, broadening its accessibility and appeal to a diverse audience. Whether viewed on a mobile device, tablet, or desktop computer, the SVG-based visuals will maintain their integrity and effectiveness, making them a crucial part of the project's overall design strategy.

To simulate the interactions and functionalities of the IoT smart devices within the model, the project employs Node-RED, a low-code programming platform. Node-RED is particularly well-suited for this task due to its user-friendly interface, which simplifies the

development process and makes it accessible to users with varying levels of coding expertise. The visual programming environment of Node-RED allows developers to create complex interactions through an intuitive drag-and-drop interface, significantly reducing the time and effort required to build and test the simulation.

The flow-based design of Node-RED provides a clearer visual representation of how the IoT smart devices interact with each other, making it easier to understand the relationships between different components and to troubleshoot any issues that arise. This clarity is particularly valuable in a complex system like a smart condominium simulation, where multiple devices and interactions must work together seamlessly. Furthermore, Node-RED facilitates faster prototyping, enabling developers to quickly iterate on different automation scenarios within the simulation. This capability is crucial for exploring various smart condominium configurations and refining the model to better meet user needs and expectations.

1.4 Project Scope

The project scope involves the comprehensive development of a smart condominium simulation model that utilizes Scalable Vector Graphics (SVG) for object design and incorporates low-code programming through the Node-RED platform. This project aims to create a versatile platform where users can design, simulate, and optimize smart condominium models, taking into account human interactions, various automation scenarios, and the integration of Internet of Things (IoT) devices. The platform is envisioned as a modular simulation environment, structured to include a wide range of rooms, devices, sensors, and interactions that closely resemble those found in contemporary smart home settings. For instance, specific design elements, such as the fan animation—where the fan spins when operational—will be integrated to enhance the realism and interactivity of the simulation. The design of the platform will prioritize scalability and flexibility, allowing for future expansion and customization to meet evolving user requirements and technological advancements.

In addition to the foundational structure, the project will incorporate IoT sensors into the simulation platform, which will play a critical role in collecting real-time environmental data essential for smart home operations. These sensors will monitor key environmental factors, including temperature, humidity, light levels, occupancy, and other relevant indicators, thereby simulating real-world conditions with high accuracy. This data collection capability will be integral to the operation and automation of smart home functionalities within the simulation. By closely mirroring actual smart condominium environments, the simulation will enable users to understand and optimize the interactions between various devices and environmental conditions.

The scope of the project also includes the implementation of control mechanisms and advanced automation features within the simulation platform. Users will be able to control and adjust IoT devices directly through the platform based on real-time environmental data and interactions between users and devices. The automation features will allow for the creation of complex scenarios that mimic real-life smart condominium operations. For example, the platform will be able to automatically adjust lighting based on the detected ambient light levels, lowering the air-conditioning temperature when the room temperature rises above a certain threshold, or activating security systems in response to specific triggers. These automation features will provide users with a realistic and interactive experience, allowing them to explore and optimize smart condominium scenarios effectively.

Moreover, the project aims to ensure that the simulation platform is both user-friendly and adaptable, catering to a broad range of users with varying levels of technical expertise. The use of Node-RED's low-code programming environment will facilitate this, enabling users to design and implement smart home functionalities without needing extensive programming knowledge. The visual programming interface of Node-RED will simplify the development process, making it easier for users to create, modify, and test their smart home simulations. By integrating SVG for visual design and Node-RED for functionality, the project will deliver a powerful tool that not only simulates a smart home environment but also allows for continuous enhancement and adaptation in response to changing user needs and technological innovations.

1.5 Contributions

The project has 3 main which the first contribute are contributes to the Development of a Comprehensive Smart Condominium Simulation Platform System. This project will use Node-RED's low-code programming methodology to create and build a comprehensive smart condominium simulation platform, where this platform allows the development, simulation, and improvement of smart condominium models that can combine a range of IoT devices, automation scenarios, and user experiences. Users will be benefit by a user-friendly interface that allows for easy navigation, visualization of data, control over simulations, and testing of scenarios.

Other than that, the project will also contribute about the accessibility and democratization of smart condominium development will also by using Node-RED low-code programming techniques, this project aims to democratize the development and simulation of smart condominiums so that people with multiple levels of computing skill can use it. The wider range of people such as homeowners and developers are able to participate on the designing of the smart condominium environment that suits their own requirements and needs.

Lastly, the project will contribute to the enhanced of efficiency and innovation. The Smart Condominium Simulation Platform will enhance the operational efficiency by adding automated simulation processes, minimizing the human errors, and allows the faster design and the testing of the smart home scenarios. This automated simulation process will lead to cost savings, improved the resource usage, and also increase the innovation of smart home technologies, and can benefits both users and businesses in the home automation industries.

1.6 Report Organization

There are total of seven chapters in this report which introduction will be represents the first chapter of the report. In the Chapter 1, Problem statements, project goals, inspiration, project scope, and anticipated contributions are all covered in this section. After that, Chapter 2's Literature Review examines the hardware and software technologies pertinent to the project and looks at four current systems that are relevant to the project's setting. System Methodology which discusses about the system development model along with technologies involved which includes hardware and software. System Design will be discussed about the design aspects of the system, including functional modules and system flows in Chapter 4. Chapter 5 will be discussed about the system implementation of the project. Chapter 6 covered the system evaluation and discussion about the system testing and performance metrics, testing setup and results, project challenges and the evaluation of the objectives. Lastly Chapter 7 is the conclusion and the recommendation for the whole project and system.

CHAPTER 2

Literature Review

2.1 Review of the Technologies

2.1.1 Firmware/OS

Windows OS

Windows OS is developed by Microsoft Corporation. Windows OS is a popular Operating System (OS) that are widely used in all around the world because of the Windows OS ease of use. This makes Windows OS is installed on around 90% of personal computer in all around the world [1]. There are few versions of the Windows Operating System right now, and the latest version of Windows OS is Windows 11. Windows OS also have 2 types of editions which is Windows Home and Windows Professional. Windows Home are the most basics operating system that includes Windows' basic features such as web browsing, internet connectivity, playing video games, etc. While the Windows Professional which also known as Windows Pro is a more powerful version that includes all the feature in Windows Home and in addition with the following features such as: Remote Desktop, Hyper -V, Bitlocker and Trusted Boot [1].

Unix OS

Unix OS is an operating system that design for adaptability and flexibility. Based on the article in [2] Unix OS is a operating system that written in C programming language. Unix OS is made up of few components including the kernel, shell, file system and core set of utilities or programs. Kernel is the heart of the Unix OS which acts as a master control of the program that provide services from start to end of the programs. User normally interact with Unix OS through the shell of the OS. The CLI commands will be pass to kernel for execution [2]. Operating System like Mac OS and Linux are Unix-based operating system.

2.1.2 Database

MongoDB

MongoDB is a Database Management System (DBMS) that used documents instead of tables to process and store data. According to [3], MongoDB is a database that does not need any Relational Database Management System (RDBMS), which means that it provides a flexible data storage model that allows users to store and query data types easily.



Figure 2.1.2.1 MongoDB Logo

InfluxDB

InfluxDB is a Time Series Database (TSDB) system that written in Go language. InfluxDB offers a high availability retrieval of the data, faster storage time which makes InfluxDB becomes a popular database option for monitoring operation, used for application metrics, storing IoT sensor data and perform real-time analysis [4].

2.1.3 Programming Language

Java

Java is created by James Gosling is a high-level Object-Oriented Programming Language (OOP). Java primary goal is “write once, run anywhere” this means that it is a program that allow user to be moved to a mobile device easily [5]. Because of the easiness to compile, Java now is the most popular programming languages in the world, and it is suited for building mobile applications, support IoT devices, creating games or simulations and etc [6].

Node-RED

According to [7], Node-RED is a visual programming stream-based tool that is mainly focuses on the IoT (Internet of Things). Node-RED is an open-source programming tool that developed by IBM Emerging Technology. The flow of Node-RED operates by passing the coding between each node. The coding are simple JavaScript objects that allows any of the properties. Node-RED provides browser-based flow editor for user which make the user easy to link the flows together using the nodes given in the palette [8]. Since Node-RED is built on Node.js, means it taking the advantages by ideally allows the low-cost hardware such as Raspberry Pi or cloud able to run at the edge of the network. Lastly the flows that created in Node-RED can be saved using JSON format which it can be easily imported or exported to share with others [8].

2.1.4 Summary of the Technologies Review

Firmware/OS

Description	Strength	Weakness
Windows OS	<ul style="list-style-type: none"> • Wide compatibility and software availability • User-friendly Interface • Backward Compatibility 	<ul style="list-style-type: none"> • High licensing cost • Require higher system performance
Unix OS	<ul style="list-style-type: none"> • Stability and Reliability • Flexibility and customization 	<ul style="list-style-type: none"> • Complex of learning curve • Limited software compatibility

Table 2.1.4.1 Summary of Firmware/OS

Database

Description	Strength	Weakness
MongoDB	<ul style="list-style-type: none"> • Simplified Performance Optimization • Schema Not Required • Efficient Memory Utilization 	<ul style="list-style-type: none"> • Limited Transaction Scope • Duplicated Data and Memory Usage
InfluxDB	<ul style="list-style-type: none"> • Store time-series data • Faster storage time • High availability retrieval of the data 	<ul style="list-style-type: none"> • Memory Usage • Limited SQL functionality

Table 2.1.4.2 Summary of Database

Programming Language

Description	Strength	Weakness
Java	<ul style="list-style-type: none"> • Object-Oriented Programming support • Platform Independence • Strong IDE support 	<ul style="list-style-type: none"> • Complexity in large system • Limited Low-Level Programming
Node-RED	<ul style="list-style-type: none"> • Ease of use • Extensive node library • Low-Code development 	<ul style="list-style-type: none"> • Limited language support • Complex to perform debugging

Table 2.1.4.3 Summary of Programming Language

2.2 Review of the Existing System/Application

2.2.1 Smart Home Simulation using A.I and Java Programming Language

The authors of [9] says that their implementation for this work is not just only a smart home simulation that simulates the smart home only, they also added an A.I (Artificial Intelligence) to create a realistic person that acts like human being. To make this simulation more realistic, they added total of 5 basic needs for the A.I which is:

- Energy
- Hunger
- Hygiene
- Bladder
- Fun

These 5 needs can be satisfied by activities that related to, for example, Hunger can be satisfied with food, washing hand, or having shower satisfied Hygiene, Playing video games for fun and etc. They used an Artificial Intelligence Algorithm which called Graphplan Algorithm which is an algorithm used to plan a sequence of an events to reach a specific goal. They also use PLPlan which is an Open-Source Artificial Intelligence Planner that develop by Philippe Fournier - Viger, Assistant Professor at University of Moncton, Canada by implementing a wrapper over the library just to include their needs system and provides an easier API (Application Program Interfaces) in order to use the algorithm.

After the A.I simulation, the authors started to develop the smart home simulation. According to the explanation of [9], the reason why author choose Java because Java is a “Write once, run everywhere” language. The simulations use Swing Framework for the GUI (Graphical User Interface) to presents a simulation map where user can observe what did the A.I do inside the virtual house. They use a MySQL Database to store the A.I and the map locations, connections to the nodes. Those possibilities of the activity will by read from JSON file and the PLPlan will be the planner which this PLPlan is a Java library that implements the GraphPlan Algorithm.

2.2.1.1 Limitation of Smart Home Simulation using A.I and Java Programming Language

The combination of Artificial Intelligence (A.I) and Java Programming in the smart home simulation indeed is a new era of innovation. However, it still comes with its own set of limitations that need to be attention in make further improvement.

Based on the article [9], there are total of two limitation and one of the limitations of in this article is the reliance on a simplified human activity. This model only applies five basic needs, such as energy, hunger, hygiene, bladder, and fun. Although this framework useful for studying human behaviour in the simulated environments, but it is not ideal at capturing the intricacies and the complexity of human action and the making of decision action in the real word. This limitation could affect the simulation's accuracy and realism, especially in the complexness of the human reaction.

Moreover, the difficulty of implementation included the needs to train AI models accurately based on the simulated scenarios. Training an AI model required a big amount of time, resources, and expertise. This process involves feeding the model with massive amounts of data, refining algorithms, and fine-tuning parameters to ensure the performance. This will bring challenges, especially developers and organizations with limited resources or technical capabilities.

Another limitation is the resource-hungry because training a comprehensive AI model requires a lot of time and computer resources. The requirements for the specification of a computer just to trains a comprehensive AI model can be significant, requiring powerful hardware, extensive processing capabilities, and substantial energy consumption. This resource demand not only will increase operation costs but also will limits the scalability and accessibility of smart home simulations.

2.2.2 Smart Home Simulation Using OpenSHS Architecture

This paper author [10] used OpenSHS which is a new hybrid, open-source, and cross-platform 3D smart home simulation tool for dataset. This OpenSHS architecture is based on Blander and Python. In the author research they separate into two entities which is researcher and participant, which research basically focuses on working on OpenSHS while the participants will be anyone that volunteers to simulate their activities.

This report will mainly focus on author [10] researcher entity which the author divided the working progress into three main phases which is design phase, simulation phase and aggregation phase. In design phase, the mainly on designing and building the virtual environment, insert the smart home devices, and assign the label of activity and the context.

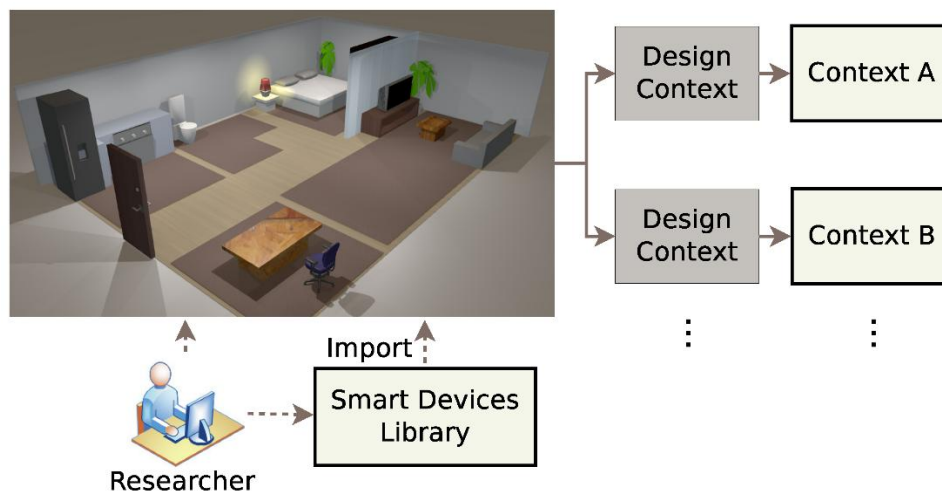


Figure 2.2.2.1 Author's design phases

Then, the researcher used Blender to design a floor plan in 3D. Blender allows researchers to simulate the architecture of house and control different aspects according to their own perspective. After designing the floor plan, researchers will start implementing smart devices or sensors from the smart device library offered by the OpenSHS which includes Pressure sensors, Door sensors, Appliance switches (Television, oven, fridge, etc) and Light control sensors. These sensors and devices are extensible because it is programmed with Python and allows researchers to customise their own devices or sensors. Finally in the design phase, researchers can start to assign the activity labels and design the context which the labels can be “eat”, “work” and “other” and the context can be specific in time frames such as morning, afternoon, evening, and night context.

After the design phase, researchers will start going into simulation phase which this phase can start the simulation using the tool from the OpenSHS interface module. In the simulation phase participants can select their own context and apply into the simulation mode. The goals of this simulation phase in this [10] report is to capture the real-life interactions of the participants and collect the data. But it is a massive job for the participants and the researchers, that's why they used OpenSHS because this application provides a fast-forwarding option to fast forward the simulation.

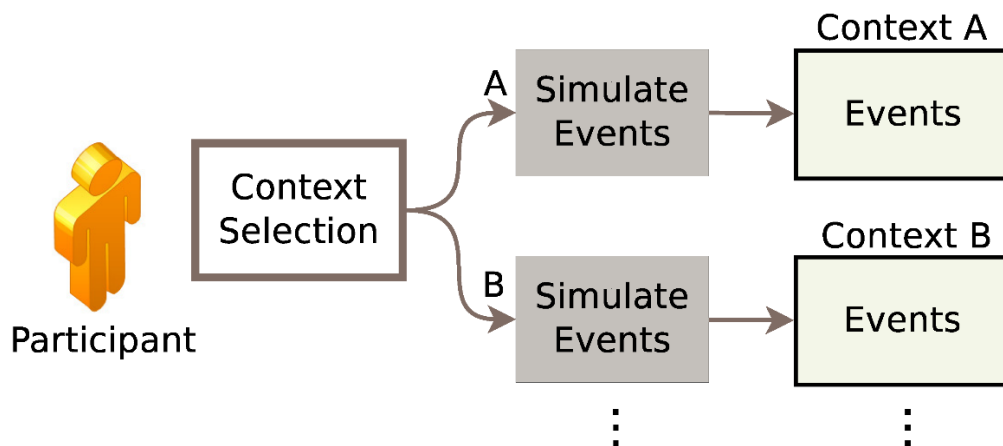


Figure 2.2.2.2 Author's simulation phase

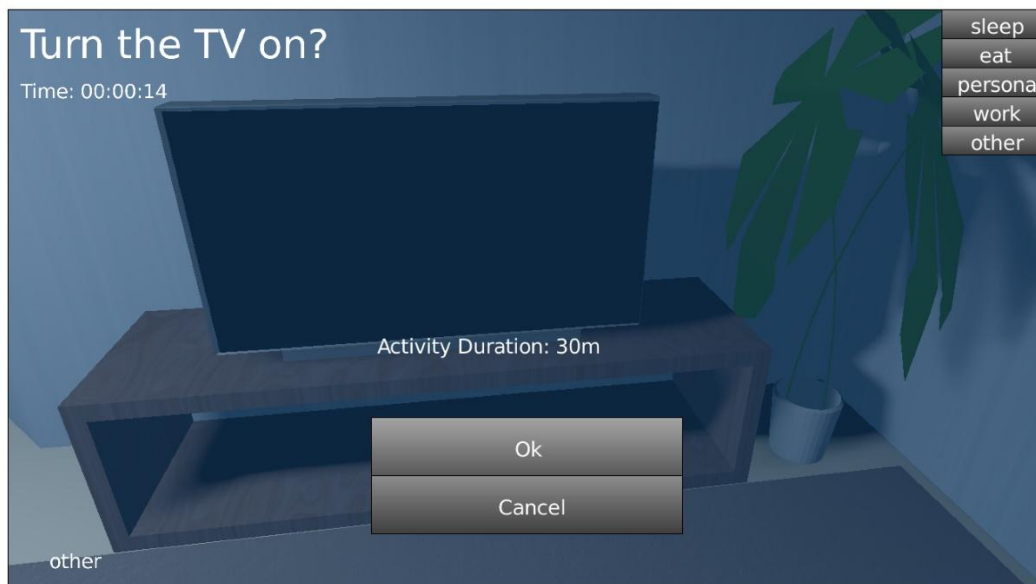


Figure 2.2.2.3 Author's fast-forwarding features.

Lastly in the aggregation phase, the researcher will aggregate the participant's sample activity after the simulation phase to produce the final dataset for the result of the whole simulation process.

Samples		Activities			
1	sleep	personal	work	eat	other
2	sleep	personal	other		
3	sleep	personal	other		
4	sleep	eat	personal	other	
5	sleep	eat	personal	other	

Table 2.2.2.1 Dataset that authors recorded.

2.2.2.1 Limitation of Smart Home Simulation using OpenSHS Architecture

The limitation for using OpenSHS Architecture is the difficulty to implement. This is because it uses both Blender and Python, and it is challenging for individuals without a technical background because the setting up and configuring the simulation environment requires a nice technical background. Even though these tools offer advanced capabilities for designing and implementing simulation environments.

One of the limitations is the difficulty of implementation combine with Blender and Python. Blender is a powerful 3D modelling and animation software that requires a certain level of expertise and familiarity with design principles. Implementing Blender into smart home simulations requires a skill in modelling, texturing, and animating the objects in order to create realistic virtual environments. While Python scripting is to add the interactive functionalities and logic to these simulations. However, mastering Python scripting and understanding its integration with Blender can be challenging, especially for non-technical users.

Another limitation is the usability issues, including a steep learning curve for new users and limited user-friendly features in the simulation interface. Blender's complex interface and multitude of tools might confuse beginners and making it challenging to navigate and utilize effectively. Similarly, Python can be frightening for those users that unfamiliar with programming concepts and syntax. This steep learning curve can stop the potential users from participate in the smart home simulations and exploring their full potential.

Moreover, the UI design of Blender-based simulations might not have a easier visual indicators and intuitive elements that can navigate and interact with users. The complexity of the interface and the less of user-friendly design elements might cause user to feel frustrated in the simulation tasks.

2.2.3 Smart Home Simulation using Map Editor and WSN Simulator

In this article [11], the authors used the Map Editor to create the floor plan and the sensors such as Pressure Mat (PM) sensors and Passive Infrared (PIR) sensors. The authors also says that the drawing on Map Editor can be convert into XML (Extensible Markup Language) code which easier to save the project afterwards. The authors thinks that the simulator must have the ability to allow users to create a resident profile and list of the activities for each member in the home environment, so WSN Simulator will be the best simulator for the research.

In the simulation testing, the author's implements simulator's signal to create an algorithm that can identify three different kinds of falls and post-events in a single or multi-person household. The system used a single PIR sensor for each room to do the tracking of the daily activity of the resident. The fall algorithm works under the concept that if all sensors are turned off for longer than five minutes, which this algorithm's accuracy was 55.6% [11].

On the second stage of the research of this article, the authors modified the previous algorithm with enhancing the graph theory. The graph theory let the system able to track different group of people and perform the monitoring of fall events. A short trial has been conducted into a two-bedroom apartment to assess how well was the algorithm works to identify a fall resulting in loss of consciousness (LOC) and a "long lie" event. Three women and two men between the ages of 25-35 were recruited for this trial, and a volunteer and a researcher performed two activities, three fall events with recovery, three "long lie" events, and three fall events with loss of consciousness during each session. Thus, there are total of 70 sessions will be conducted and the accuracy was improved from 55.6% to 82.9% [11]. However, due to the resident movements on the floor will keep on activates the PIR sensor, the algorithm was still unable to detect the entire "long lie" event.

To differentiate between activities that occurred in the upper and lower portion of each room, the third stage of the system combines two PIR sensors into one. 360 fall events and 90 of daily activities were simulated using the simulator. Three undirected graphs and several time criteria for identifying between a static event, a "long lie" event, and a fall event with lost off consciousness were added to the algorithm to improves its performance and the accuracy was 89.3% [11] right now.

2.2.3.1 Limitation of Smart Home Simulation using Map Editor and WSN Simulator

There are few limitations for this smart home simulation using Map Editor and WSN Simulator, which these limitations make quite a big problem to author [11]. The first limitation is the potential of the hardware issues especially sensors. The problem brings by sensor may cause the accuracy of the performance of the simulator. This limitation highlighted the importance of the sensor accuracy and reliability in order to maintain the integrity of smart home simulations.

Another limitation is this simulation will be simplified human behaviors according to the resident profile. Although this technique provides a fundamental comprehension of user interactions, it still might ignore the specific details and behavior differences. This limitation highlighted the need for more advanced behavior simulating methods that can accurately represent a wider range of human behavior and reactions in the simulation setting.

CHAPTER 3

System Methodology

3.1 System Development Models

To make the development of the project to be successful, implementing Software Development Life Cycle (SDLC) is crucial for the project development. Software Development Life Cycle (SDLC) is a structured process that used for planning, creating, and implementing software applications. It can ensure that each stage of the project is carefully managed and executed. By following to the SDLC, the development can be ensured about the project meet the requirements and able to finish on time while maintaining high quality of the project.

3.1.1 System Development Model 1: Agile Model

According to [13], Agile is one of the most popular SDLC models. Agile is a combination of both incremental and iterative development methods. It involves continuously releasing the product and improving and testing iteratively. Unlike traditional SDLC models, which often follow a linear sequence of phases, Agile mainly focuses on flexibility and responsiveness to change, allowing for continuous delivery and improvement of the product throughout the development process.

One of the key principles for Agile is “fast failure”. This means that Agile encourages software developers to fail quickly in the early of the development process. This is because in the early of the development process, if the process has an error, the developer is able to solve it faster instead of the last moment of the project which the early of the failure start, the problem is easier to solve it by the developers. But if the error failed in the end of the project or nearly end of the project, developers will be harder to solve the problem due to the massive work has done for the project [14].



Figure 3.1.1 Agile Model

3.1.2 System Development Model 2: Waterfall Model

Waterfall methodology is a linear, sequential SDLC that is commonly used in software engineering and product development [15]. Waterfall methodology model used a logical progression step of the project that similar to the direction of water flows from top to bottom of the cliff. It normally will set a goal point to achieve in each phase of the project and complete of each phase of the project then only proceed to the next phase of the project.

There are total of 7 phases in the waterfall methodology based on [15]. Which the 7 phases are: Requirements, Analysis, Design, Coding and Implementation, Testing, Operation and Deployment and Maintenance. Each of the phases provides it own process. Below is the process of each phase:

- **Requirements:**

Get to know the requirements, deadline, and the guidelines of the project, which this phase will be planning the whole project workflow and what to do or what the project needed including developing or hardware.

- **Analysis:**

Analyze the system specifications needed to generate the product to the guideline of the project given. Which mostly the project will be analyze includes analyzing the finance or the resources needed.

- **Design:**

A design specification document contains the technical design requirements for the hardware, programming language, data sources, architecture, and services.

- **Coding and Implementation:**

In this phase, developer will start the project system coding and the implementation for the project. Normally the code will be developed according to the logic and requirements of specification that has been done in the previous phase. The system is normally coded in a smaller component before they merge it together for the flows.

- **Testing:**

This phase is to do testing for the coding that coded in the previous phase, ensuring that the system or the projects run smoothly and also the quality of the project. This phase might be jumped back to the previous phase for the debugging process right until the system and process does not have any problem.

- **Operation and Deployment:**

The product has been fully developed and is ready to show in the real world.

- **Maintenance:**

Improving the projects from the feedback given by other people after the operation and deployment process to make the product be more user friendly or easy to use.

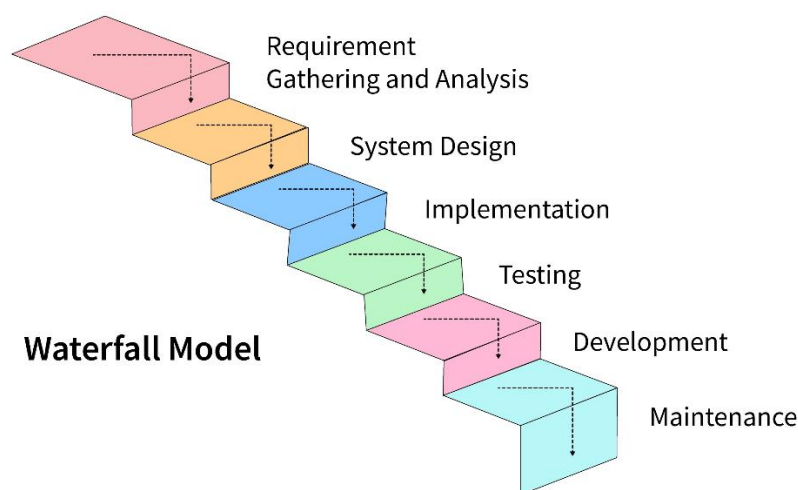


Figure 3.1.2 Waterfall Model

3.1.3 System Development Model 3: Iterative Model

Iterative Model are not a traditional SDLC like the others, it starts implementing in development specifically just the part of the software which will be reviewed in order to check the further requirements [16]. Iterative Model development is typically used in the conjunction of incremental development, which separates the massive software development progress into smaller part that interconnected with each other. The phases will be repeated over and over again once the initial planning is completed.

There are several advantages for the iterative model, which the major advantage of the interactive model is able to implement in the early stage of the software development. This will allow developers to able to find the error of the function or design in the early stages, which allows them to have plenty of time to correct the error function or redesign to a better design. Some of the function of the system can be developed in the early of the development phase of the SDLC. Iterative model is also easy to adapt according to the changing needs of the project and it is more cost effective to change the scope or the requirements during the development [16].

Although Iterative model brings a lot of advantages to the development of projects, but things are not having advantages only, it also contains disadvantages. The disadvantages of Iterative model are there cannot be extend over each of the iteration phases. Other than that, the system architecture or design issue might be enlarged once they combine due to the different requirements in each phase. Iteration model also not suitable for smaller project and require more resources including high skill resources for skill analysis [16].

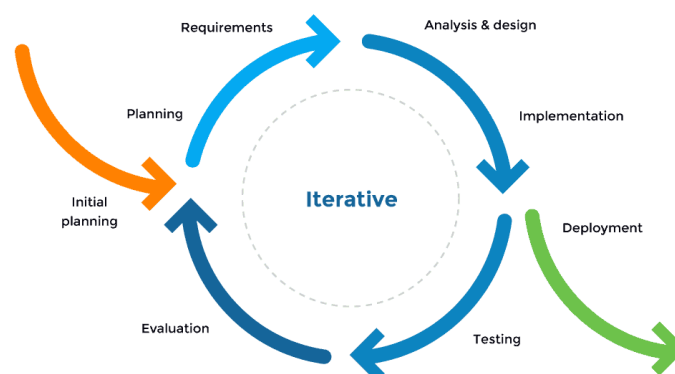


Figure 3.1.3 Iterative Model

3.1.4 System Development Model 4: Spiral Model

According to [17] Spiral Model is a systematic and iterative Software Development Life Cycle (SDLC) to software development. It will be named spiral model due to its diagrammatic representation which it looks like a spiral that contains many of loops. The exact number of loops is unknown and is different depending on different projects. Each phase of the software development process represents each loop of the spiral. Each of the iteration of spiral representing the completion of the software development cycle. The phase of the spiral model includes Objective Defined, Risk Analysis, Engineering, Evaluation and Planning.

The advantages of the spiral model include Risk Handling. Some of the project may contains many of unknown risk as the development ongoing, but projects that using spiral model have includes risk analysis and risk handling in each phase. Spiral models are also suitable for large and complex project according to its function. Customers are also able to monitor the development of the product at the early stages of project. The spiral model also allows flexibility and adaptability to response to the changing of requirements or any problems that unexpected [17].

The spiral model also having it disadvantages even though there are many advantages in it. The complexity of spiral model is more complex than other SDLC. It is also much more expensive for the small projects. Spiral model projects are over relied on the Risk Analysis and without the experienced risk analyst, the project might end up being fail of developing of the project. Other than that, since the number of phases is unknown at the starting of the project, spiral model is hard to do the time estimation which increased the time management for the project. Spiral model also can be time-consuming because it requires many evaluations and reviews [17].

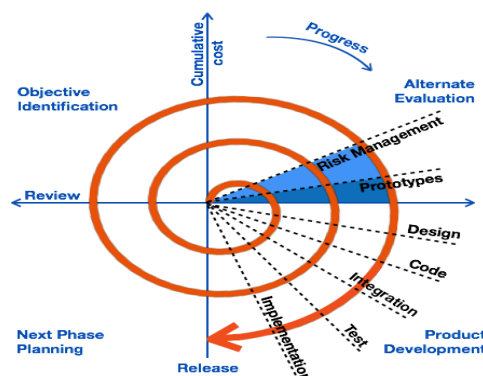


Figure 3.1.4 Spiral Model

3.1.5 Selected Model

After the researched for the 4 Software Development Model, the most suitable Software Development Life Cycle (SDLC) for this project is the Agile Model. Since agile model is an incremental and iterative development methods, it is able to proceed the project continuously by testing the project regularly to know which part of the project should be improved. Most importantly, the key principle of the Agile Model is “fast failure”, which is suitable for this project flows which developer can fail in the early process to decrease the work of debugging when in the testing of the project. This project is also flexible enough to change the design and function flows during the development process. So, Agile Model is the most suitable SDLC for this project.

3.2 System Design

3.2.1 Hardware

The hardware that uses in this project will be computer to develop and runs the smart condominium simulation. Other than laptop, this project will also use mobile phone to runs the smart condominium simulation.



Figure 3.2.1.1 Acer Nitro 5 AN515-46-R20B

Description	Specification
Model	Acer Nitro 5 AN515-46-R20B
Processor	AMD Ryzen 7 6800H with Radeon Graphics 3.20 GHz
Operating System	Windows 11 Home Single Language
Graphic Card	NVIDIA GeForce RTX 3060 6GB GDDR6
Memory (RAM)	16GB DDR5 RAM
Storage (ROM)	1TB PCIe NVMe Gen4 SSD

Table 3.2.1.1 Specifications of Laptop



Figure 3.2.1.2 Apple iPhone 13

Description	Specification
Model	Apple iPhone 13
Processor	Apple A15 Bionic (5 nm)
Operating System	iOS 17.6.1
Graphic	Apple GPU (4-core graphics)
Memory (RAM)	4GB RAM
Storage (ROM)	256GB

Table 3.2.1.2 Specifications of Mobile Phone 1



Figure 3.2.1.3 Xiaomi 14T

Description	Specification
Model	Xiaomi 14T
Processor	Mediatek Dimensity 8300-Ultra (4nm)
Operating System	Android 15, HyperOS 2
Graphic	Mali G615-MC6
Memory (RAM)	12GB RAM
Storage (ROM)	256GB

Table 3.2.1.3 Specifications of Mobile Phone 2

3.2.2 Software

Node-RED

Node-RED is a programming tool that designed for connecting the hardware device with the APIs (Application Program Interfaces), and online services specifically for IoT (Internet of Things). It is a flow-based programming so that users without high experience in coding also manage to create the application. Node-RED will be the most suitable platform to develop the entire project of smart condominium simulation.

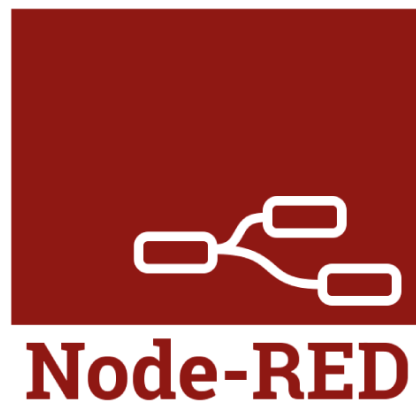


Figure 3.2.2.1 Node-RED Logo

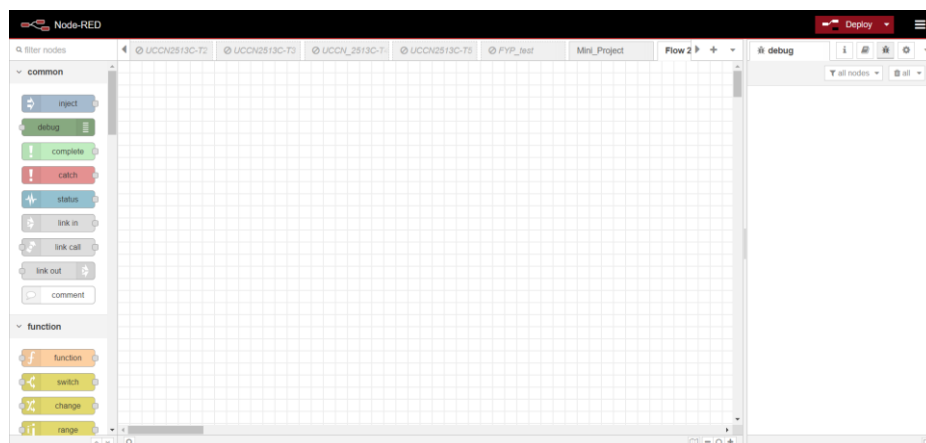


Figure 3.2.2.2 Interface of Node-RED

Inkscape

Inkscape is a free and powerful vector drawing program for Windows and Mac that works similarly to Adobe Illustrator. Inkscape also let user to design their own drawings and convert into SVG (Scalable Vector Graphic)



Figure 3.2.2.3 Inkscape Logo

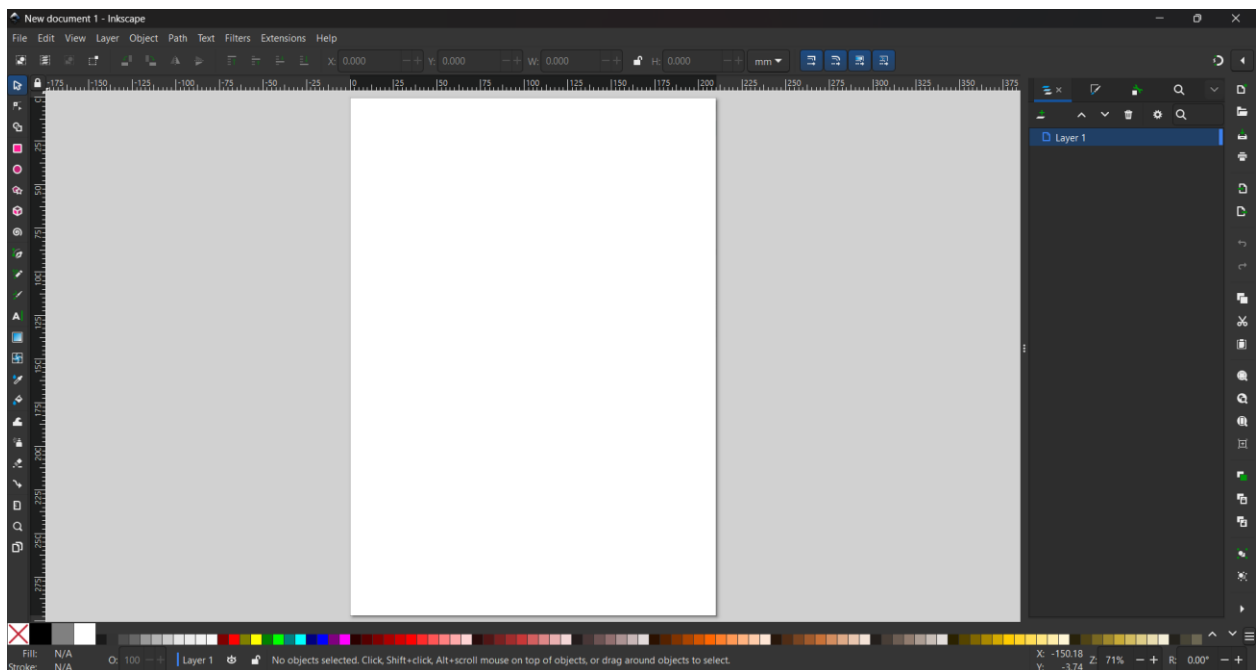


Figure 3.2.2.4 Interface of Inkscape

InfluxDB

InfluxDB is an open-source Time Series Database (TSDB). Its area of specialization includes real-time analytics, Internet of Things (IoT) sensor data, application metrics, and monitoring. It is intended for high-performance and high-efficiency storage and is primarily written in the Go programming language.



Figure 3.2.2.5 InfluxDB Logo

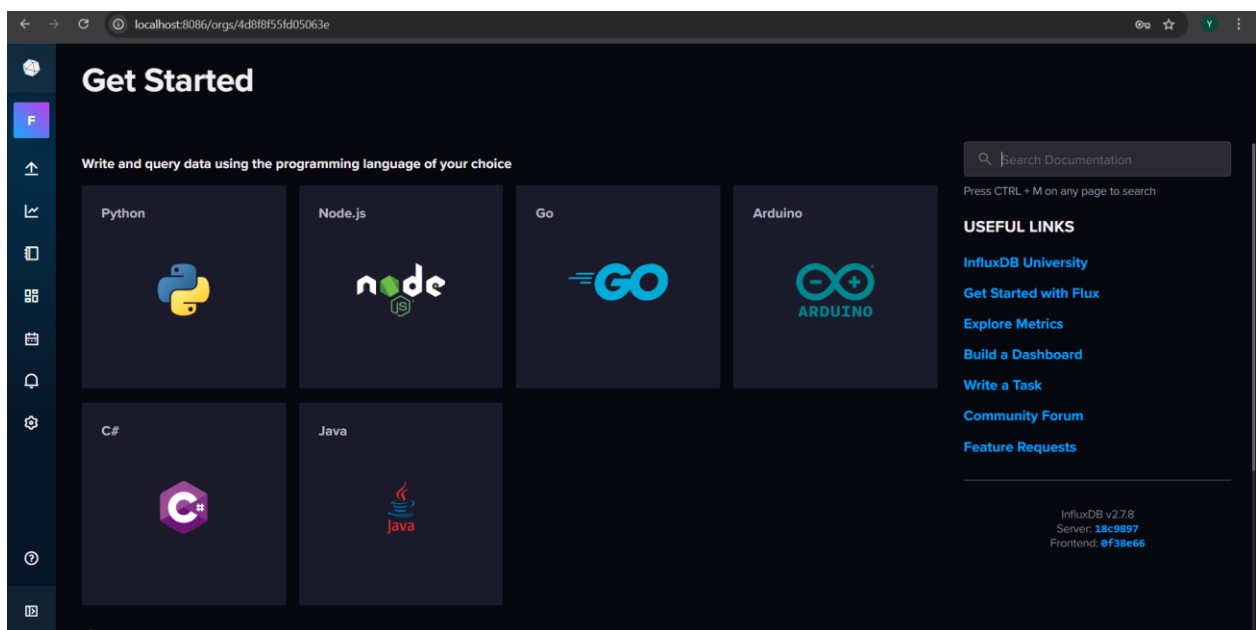


Figure 3.2.2.6 Interface of InfluxDB

IoT MQTT Panel

IoT MQTT Panel is a mobile application that allow users to control and manage their IoT smart devices in mobile phone based on the MQTT (Message Queuing Telemetry Transport) protocol.



Figure 3.2.2.7 IoT MQTT Panel Application Logo

Telegram Chatbot

Telegram Chatbot is a bot application that runs in Telegram. It can set to be interact with users and run specific task automatically or provide notification or information. Users can interact with Chatbot using the specific commands.



Figure 3.2.2.8 Telegram Logo



Figure 3.2.2.9 Telegram BotFather

3.3 Functional Requirements

No.	Description
1	The Node-RED dashboard should display the SVG graphic that saved in the SVG node.
2	The SVG IoT Smart Furniture/Devices in the dashboard should be able to respond by the function after triggering by button or slider.
3	The IoT MQTT Panel Application should be able to control and responded by the SVG graphic that deploy in Node-RED dashboard.
4	Telegram Chatbot should be able to send the notification of the specific furniture function and the SVG graphic in the dashboard should be responded after the command sent in Telegram Chatbot.
5	InfluxDB should be able to store the real time temperature data that changes from the slider of the temperature which is treat is as “virtual sensor” of the smart condominium.
6	Node-RED dashboard InfluxDB tab should be able to retrieve and display the real time data from the InfluxDB database.

Table 3.3.1 Description of Functional Requirements

3.4 Project Milestones

FYP1

Task Description	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
FYP1														
Project Analysis														
Refine Project Objective and Scope														
Define the project objective and scope														
Determine the hardware/software that will be used														
Project Implementation														
Draws the floorplan and import the SVG furniture graphic in Inkscape then apply in the SVG node in Node-RED														
Create the function flow of the system in Node-RED														
Apply MQTT node and connect to the MQTT Panel App in mobile phone														
Apply Telegram node to the system and activate Telegram Chatbot														
Apply InfluxDB database in Node-RED system														
Project Testing														
Run the system to make sure the SVG are functionable														
Run the IoT MQTT Panel, Telegram Chatbot, InfluxDB make sure it able to run														
Observe the system														

Table 3.4.1 Milestone of FYP1

FYP2

Task Description	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
FYP2														
Project Analysis														
Review the weaknesses of FYP 1 systems														
Project Implementation														
Re-Draw the floorplan to a better design														
Implement more virtual sensor into the system														
Apply all the sensors in the system and read in InfluxDB database														
Telegram Bot should be able to control and receive specific notification														
IoT MQTT Panel/ Telegram able to check the real-life status of the smart home														
Project Testing														
Run the system to make sure it is functionable														
Observe the system														
Prepare for presentation														

Table 3.4.2 Milestone of FYP2

3.5 Concluding Remark

In conclusion, the Agile model's strategic selection and application—which provided the required flexibility and adaptability to meet changing requirements was key to the development of this smart condominium simulation project. Through usage of diverse software tools including Inkscape, InfluxDB, IoT MQTT Panel, Node-RED, and Telegram Chatbot, the project is able to simulate smart condominium features in virtually. A reliable and effective system was ensured by the iterative method, which enabled continual testing, improvement, and early detection of possible problems. This project not only shows how Node-RED and IoT integration can be used effectively, but it also emphasizes how crucial a structured, yet adaptable development approach is to the development process of any project.

CHAPTER 4

System Design

4.1 System Architecture

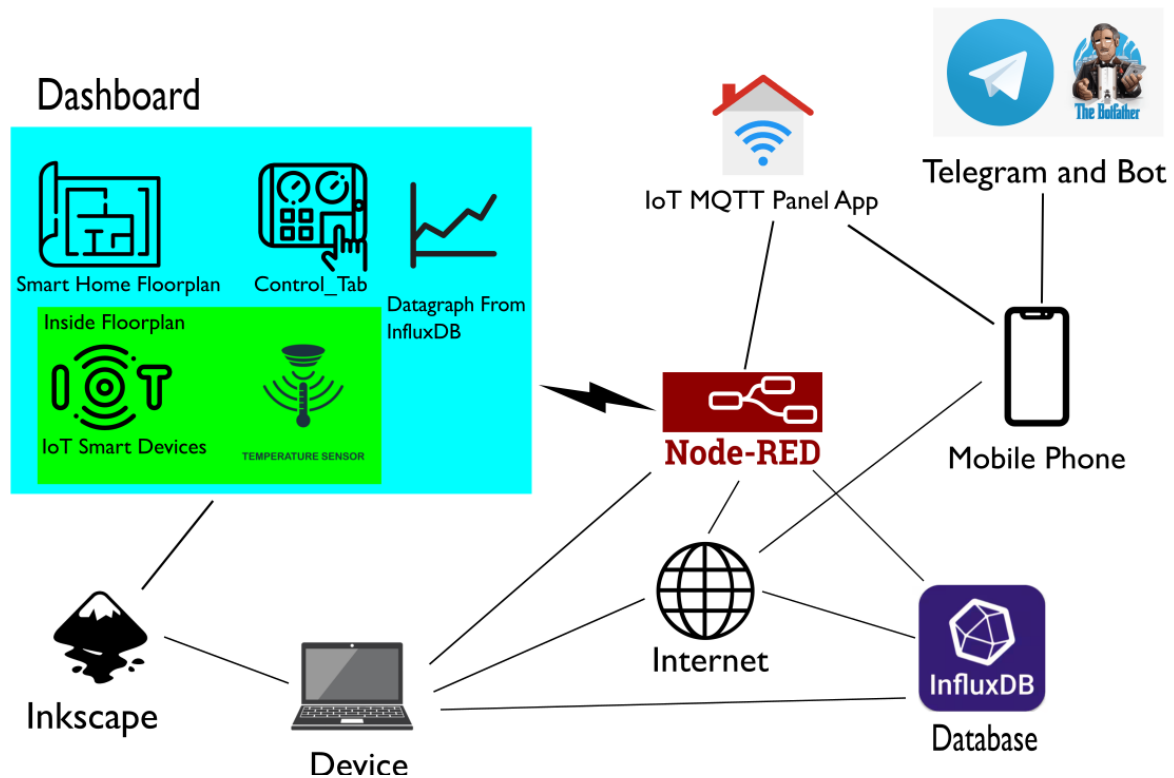


Figure 4.1.1 System Architecture Diagram

In this project, Node-RED plays the main software for creating flow-based automation and displaying a user-friendly dashboard. The Node-RED dashboard integrates the floorplan that was drawn in Inkscape to produce a virtual condominium environment and IoT smart devices in SVG to let the devices can be interact with the command when apply in Node-RED dashboard. InfluxDB, a time-series database, smoothly integrates with Node-RED by collecting the data from the virtual sensor in the smart condominium simulation, used to monitoring of the smart condominium environment.

The IoT MQTT Panel works as a mobile controller that lets users to interact and operate the devices that are connected to a system in the Node-RED flows. The direct MQTT topic-to-Node-RED connectivity of this panel ensures real-time updates and control. Additionally, a Telegram bot that created with BotFather also connected with Node-RED flows to enhance overall user experience and system responsiveness. This allows users to receive notifications, check environmental conditions, automate controls, and send commands to control devices and receive messages.

4.2 Functional Modules in the System

Module 1 (Automation and Dashboard)

Node-RED covers all the system automation flows and providing a user-friendly dashboard User Interface (UI). In the dashboard applied the SVG floorplan and smart device, which allows user to interact and control in the virtual smart condominium environment.

Module 2 (Data Collection and Monitoring)

InfluxDB will plays a role in handling the collected time-series data from the virtual sensor in the smart condominium simulation. The collected data will show in the InfluxDB bucket list and Node-RED are able to pull the collected data from database and show in the dashboard for the monitoring of the environment condition and device status.

Module 3 (Mobile Controller)

Users are able to interact and control the IoT devices that connected in the Node-RED flows. This ensures the real-time update and control through the MQTT topic that connected with Node-RED.

Module 4 (Telegram Chatbot)

The integration between Telegram and Node-RED able to enhance the whole user experience by sending notification and automation control. User can send specific command to control the devices and receive the notification through the Telegram Chatbot.

4.3 GUI Design

The Node-RED dashboard offers a user-friendly platform to show the virtual smart condominium floor plan. The floor plan is divided into three main sections which are Master Room, Living & Kitchen, and Bedrooms. This allows more detailed interaction with smart devices in each area. Additionally, an additional dashboard tab for the main door enhances the overall realism of the system by simulating access control and entry point automation, contributing to a more realistic smart condominium experience.



Figure 4.3.1 Original Floorplan for Smart Condominium

In the **Figure 4.3.2** shows the Main Door's dashboard for the smart condominium simulation. On the left side, the SVG graphic that simulates as the wall and the door structure of the condominium. On the right side, the Control Tab allows users to control and simulate door related functions which include door camera, lock and unlock switch and a password input field to unlock the door.

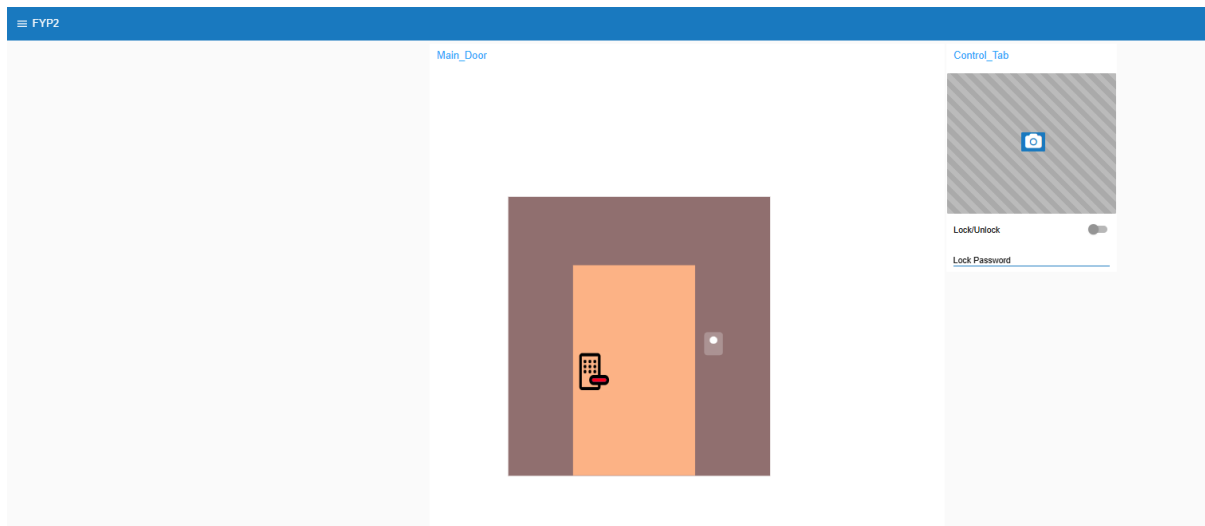


Figure 4.3.2 Node-RED dashboard of Main Door

To let the simulation process look more real, after user entered the correct password, the door handle will turn green to simulate unlock and automatically jump Living & Kitchen tab after a few seconds.

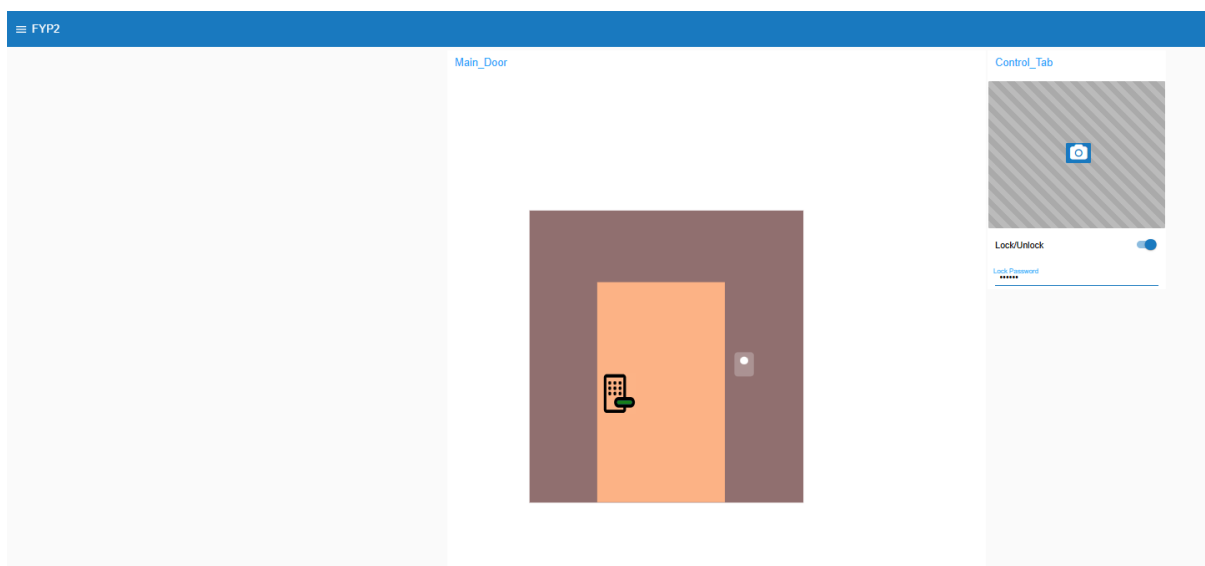


Figure 4.3.3 Colour of Door Handle turns green when unlock with correct password

In the Living & Kitchen tab, the left side of the dashboard displays the floor plan of the living hall and kitchen, which provides a clearer visual layout of the smart condominium. On the right side, the Control Tab is organized into separate sections for the living hall and kitchen controls. This separation enhances the clarity of the Control Tab, making the interface look more user-friendly, and less confusing.

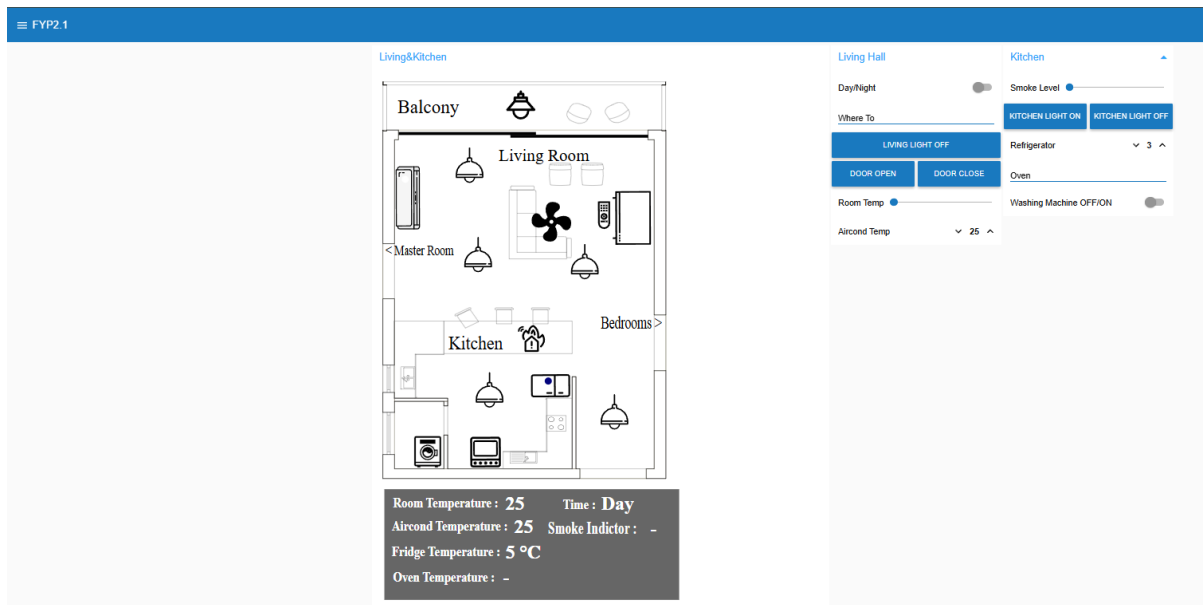


Figure 4.3.4 Node-RED dashboard for Living and Kitchen Tab (Before)

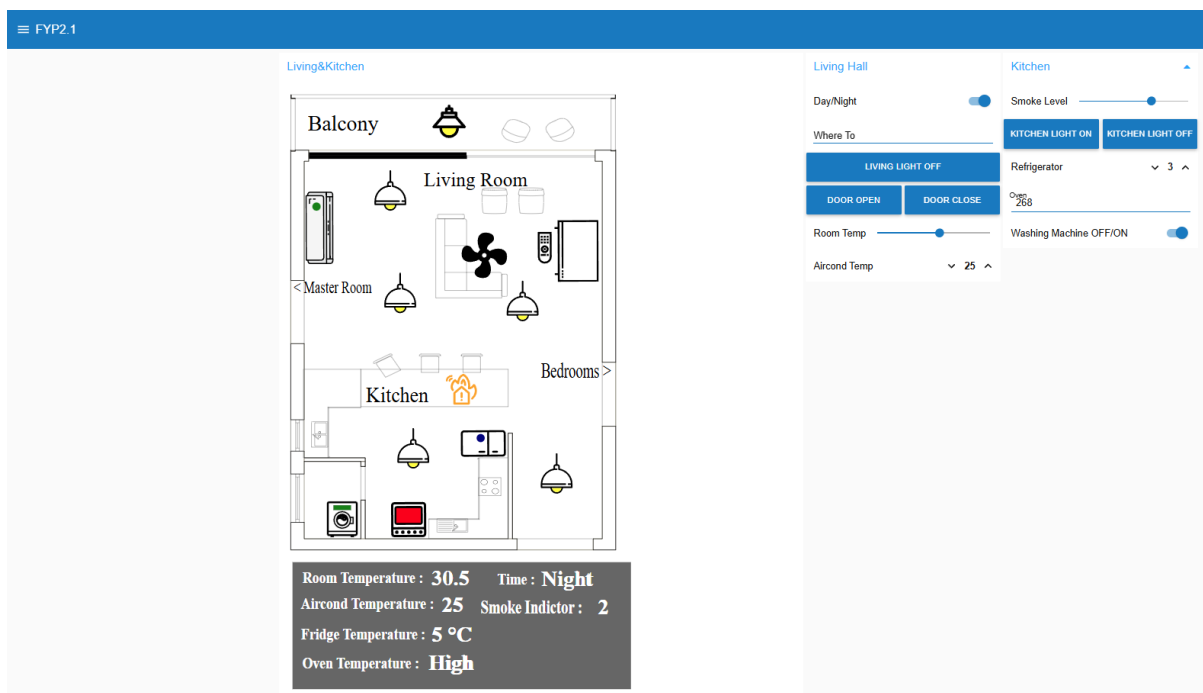


Figure 4.3.5 Node-RED dashboard for Living and Kitchen (After)

In the Control Tab of the Living Hall, there is a "Where To" input field that allows users to quickly navigate to other dashboard sections by entering keywords such as "main", "master", or "bed", as indicated by the provided hint (*Figure 4.3.6*).

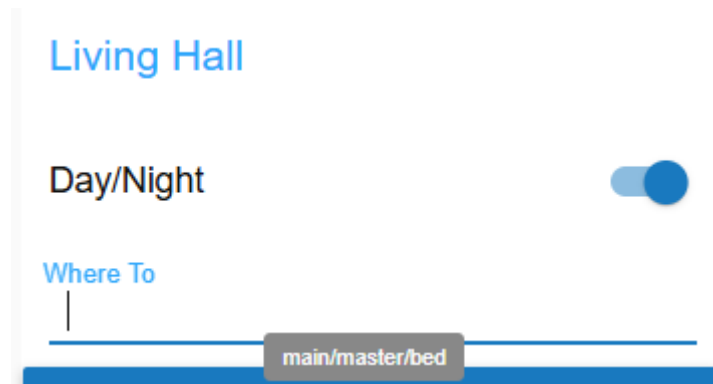


Figure 4.3.6 “Where To” input field with hint

Moving to the Master Room tab, the left side of dashboard displays the floor plan of the master room, and the right side is the Control Tab to control smart furniture inside the master room.

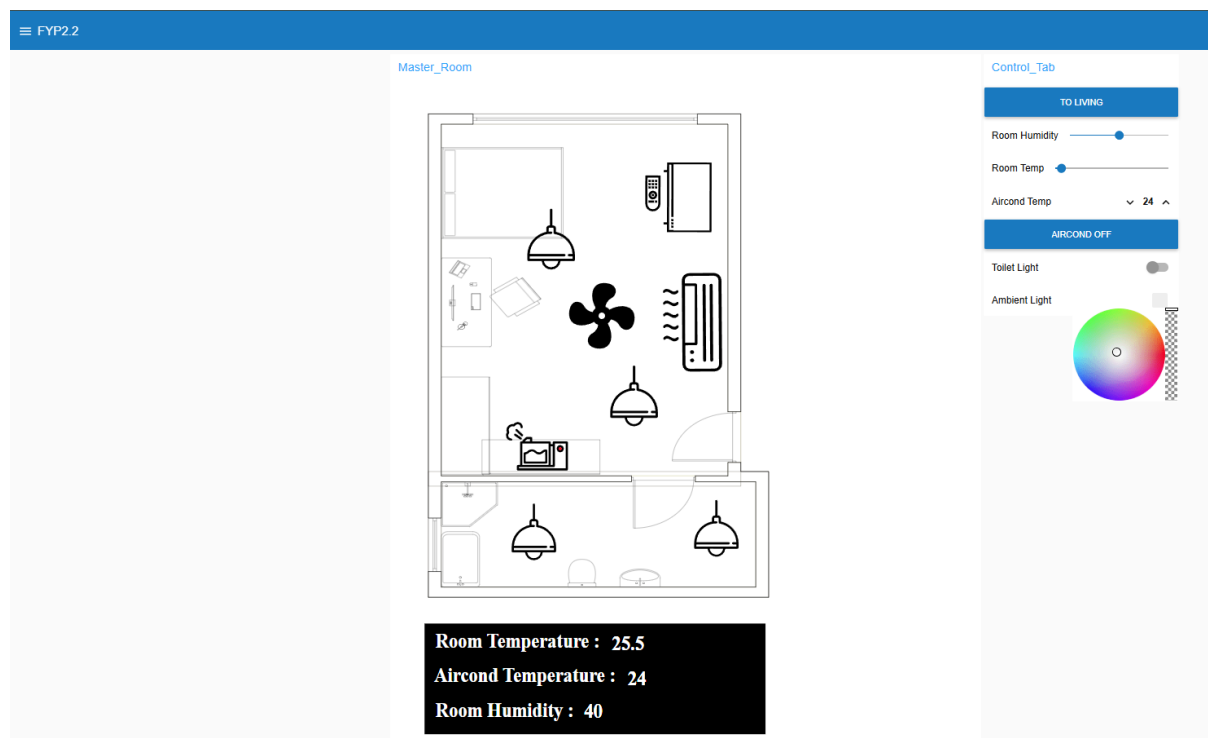


Figure 4.3.7 Node-RED dashboard for Master Room (Before)

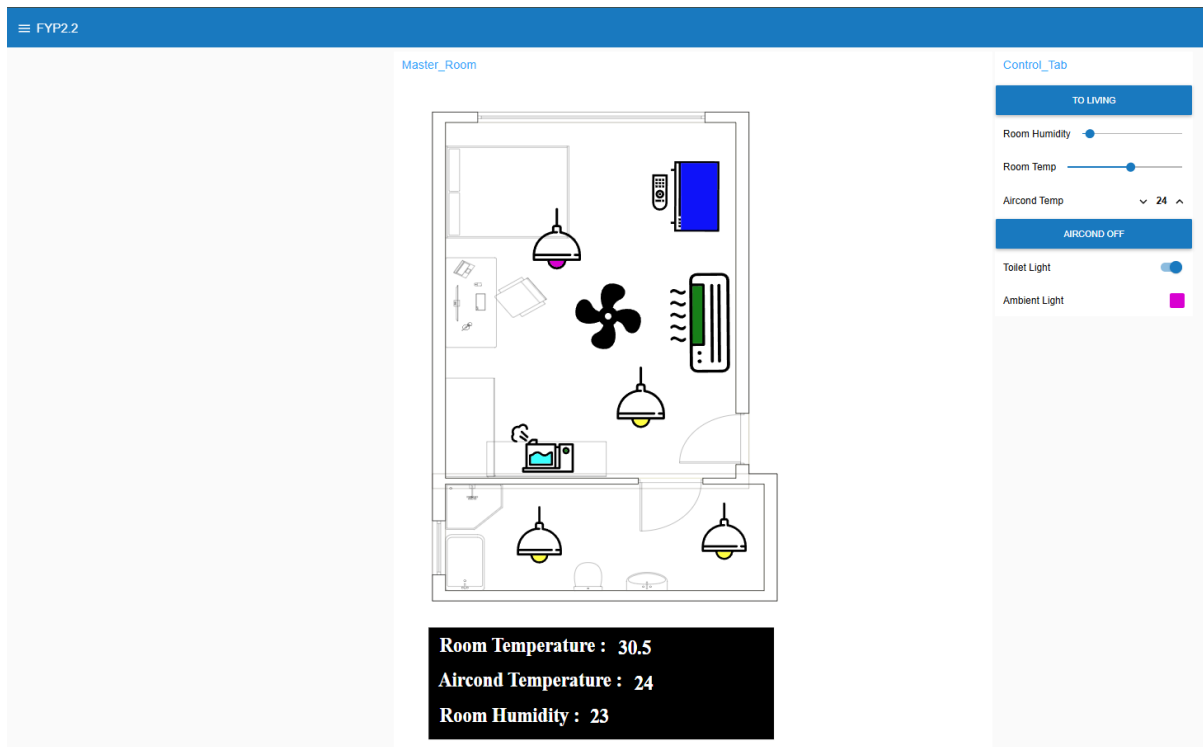


Figure 4.3.8 Node-RED dashboard for Master Room (After)

The final tab in the Smart Condominium Simulation system is the Bedrooms tab. On the left side of the dashboard, it displays the floor plan with two bedrooms and a shared washroom located between them. The right side of the dashboard is the Control Tab, which is divided into three sections to clearly separate the controls for each bedroom and the public washroom.

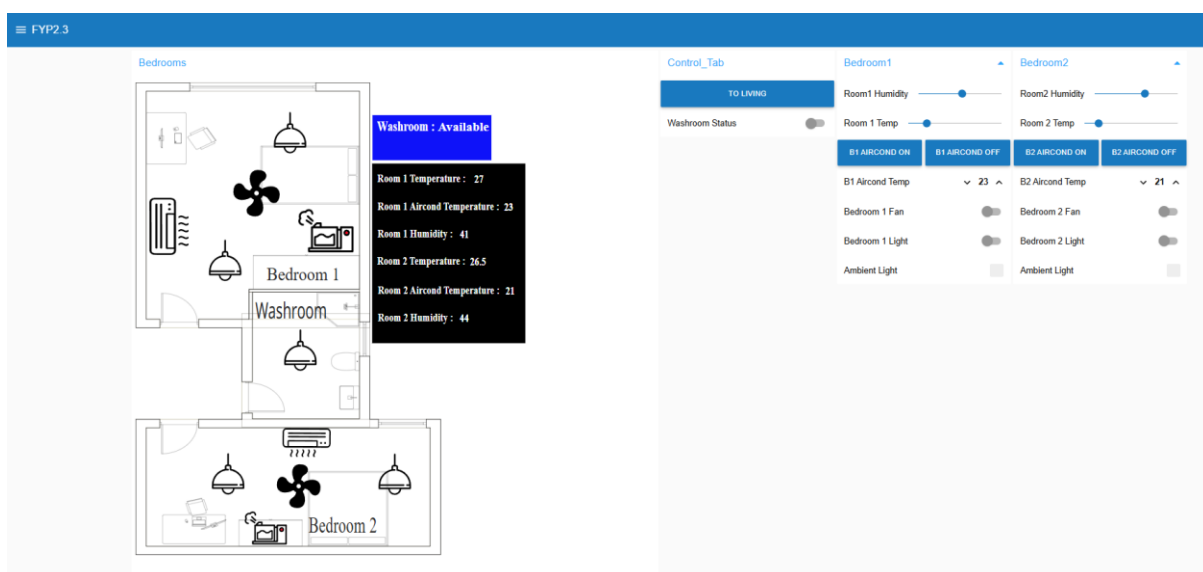


Figure 4.3.9 Node-RED dashboard for Bedrooms Tab (Before)

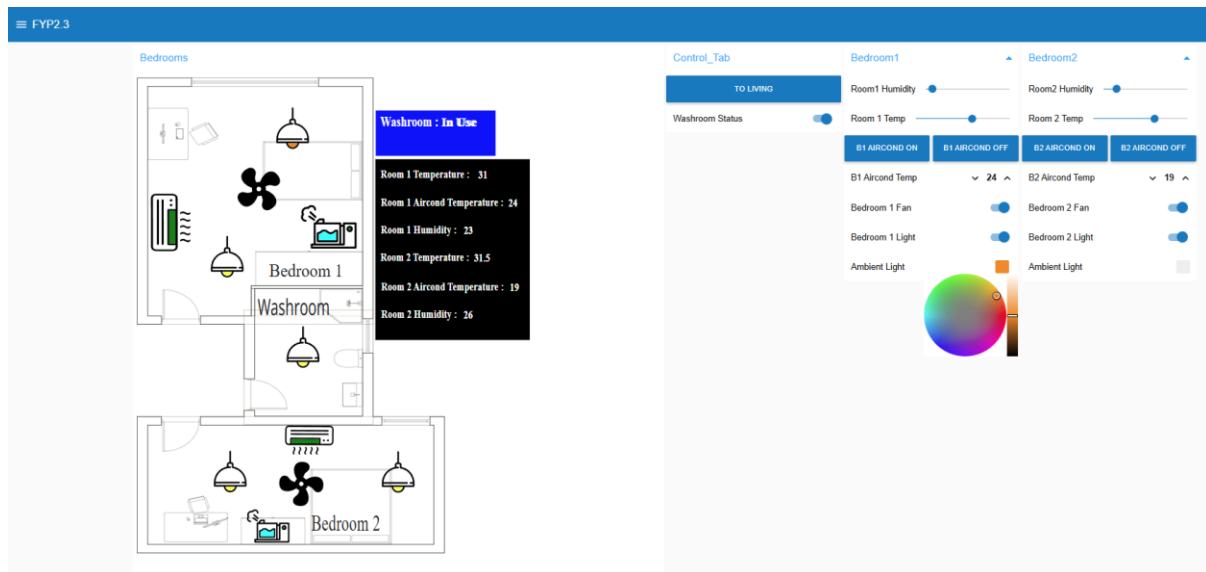


Figure 4.3.10 Node-RED dashboard for Bedrooms Tab (After)

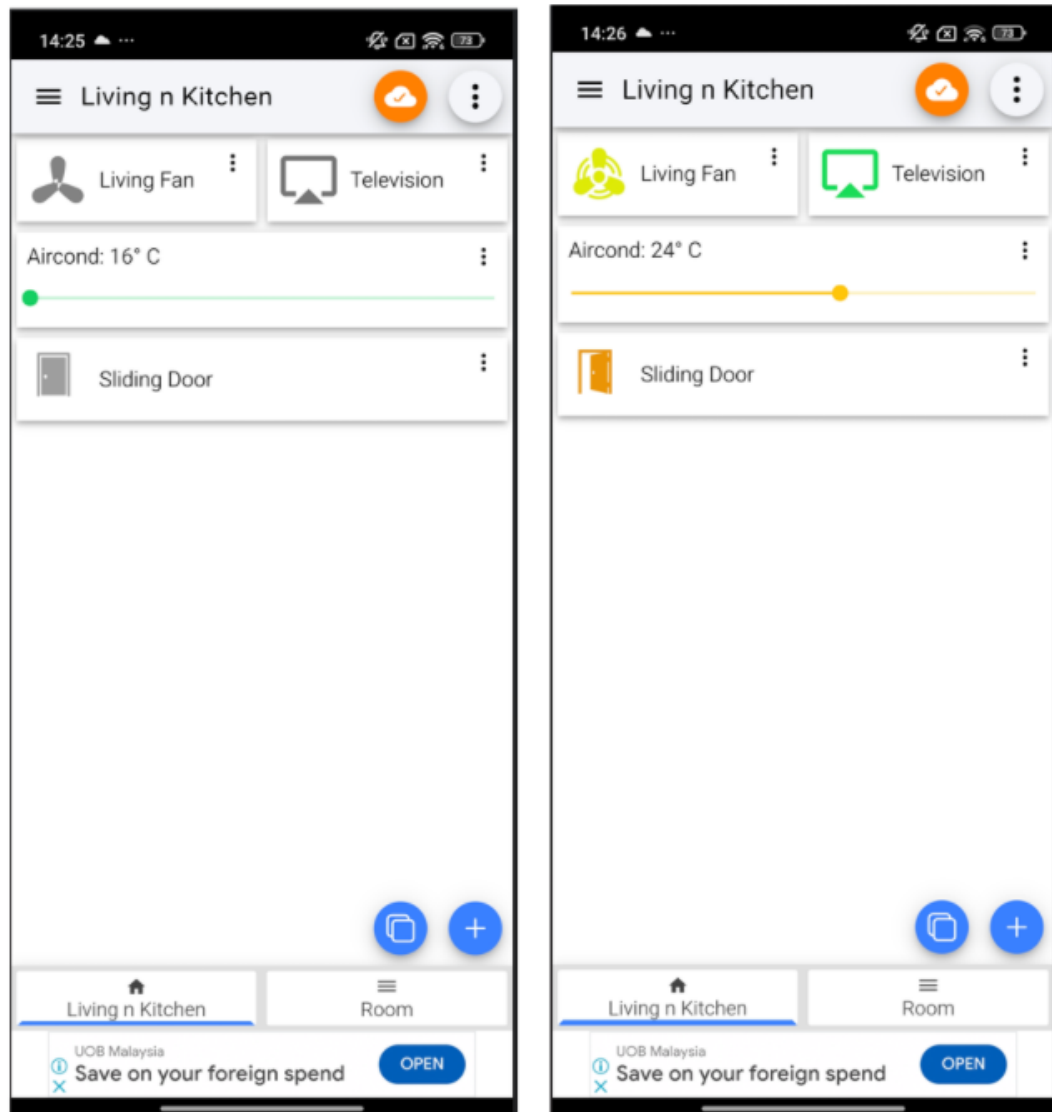


Figure 4.3.11 User Interface of IoT MQTT Panel Application (Living Hall)

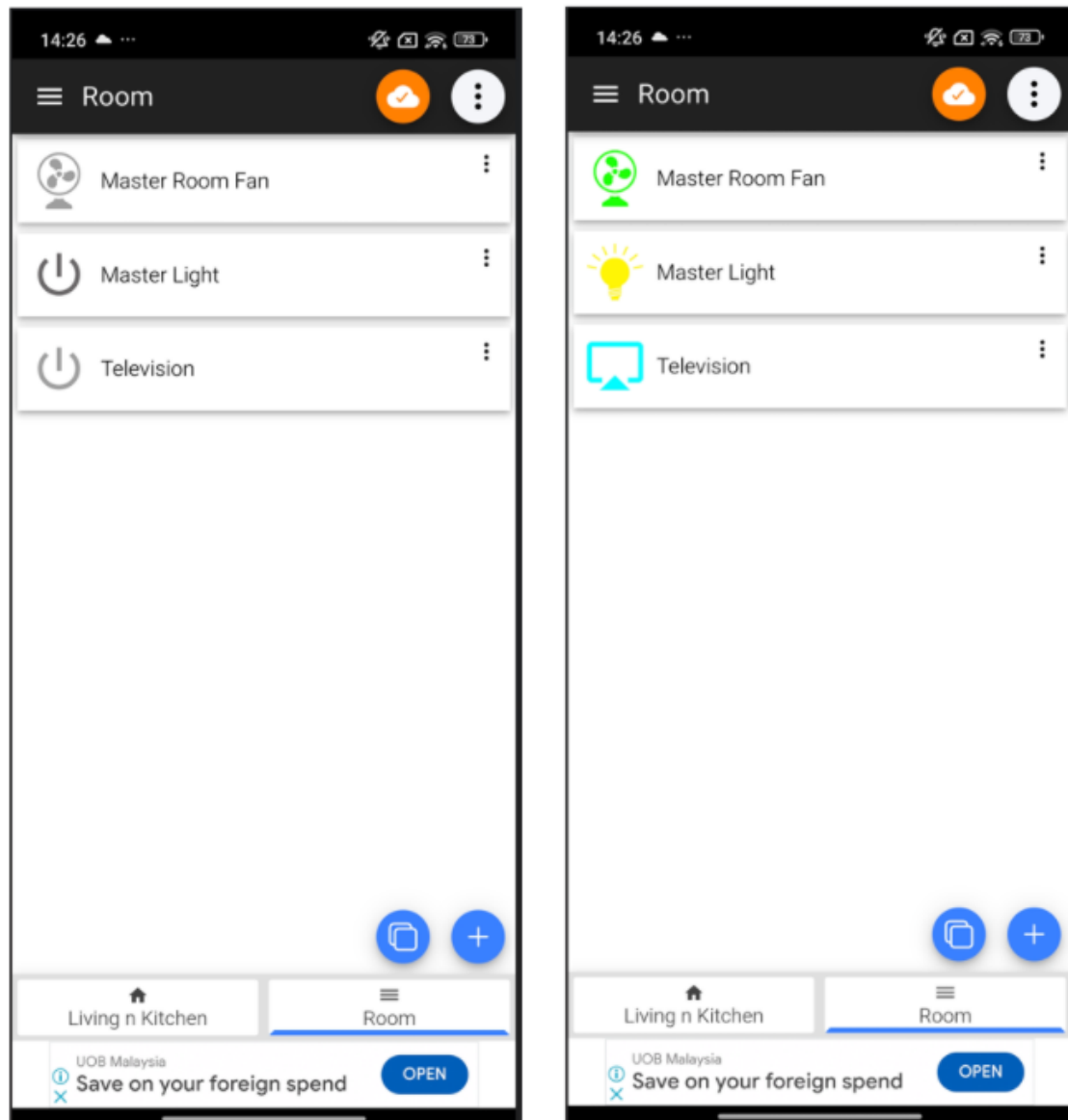


Figure 4.3.12 User Interface of IoT MQTT Panel Application (Master Room)

Furthermore, users can through mobile phones to control some of the IoT Smart Devices to turn on or turn off using the IoT MQTT Panel Application. **Figure 4.3.11** shows the User Interface (UI) of the IoT MQTT Panel Application for Living Hall, the icon in grey color means those specific smart devices are turned off while the icon that changed color shows that specific smart devices have been turned on. The slider also allows users to control the temperature of the air conditioner. **Figure 4.3.12** shows the User Interface (UI) of the IoT MQTT Panel Application for Master Room, the icon in grey colour means the specific smart devices are turned off while the icon changed color shows that the smart devices have been turned on.



Figure 4.3.13 Interface of Telegram Chatbot

Lastly, Telegram Chatbot serves as the notification sender to inform users about the smart door camera image capture, smoke level, sliding door condition, and smart oven temperature so that users can get to know the oven temperature to prevent the overheating of the oven. Other than that, Telegram Bot will also send notification if the refrigerator power has been turned off and provides a call back query regarding washing machine to let user select wash mode or dryer mode for washing machine. The connection of the Telegram node to the flow of the Node-RED enhances the user experience by offering a more efficient instant notification sender and an easier and popular mobile application to users.

4.4 Concluding Remark

This chapter analyses the smart condominium simulation system's architecture, giving readers a solid understanding of the layout and operational concept. The key software integration is emphasizing the smooth interaction between the Node-RED platform and the virtual smart condominium including the smart IoT devices. Every functional module is reviewed thoroughly, defines its distinct function in enhancing the effectiveness and responsiveness of the virtual smart condominium environment.

The system architecture diagram gives reader a visual exploration of the detailed process of the smart condominium automation, data collection and control. This diagram illustrates how those components are being connected and provide an interested and easy smart condominium experience. It provides a clear and understandable summary of the system's features. The simplicity of the design makes the controls, environmental monitoring and interaction to the device become easier to use, which ensures that every component of the system function is functional.

Lastly, this chapter covers not only functional module of the system, but also introduced each component of the user interface, such as Node-RED dashboard that user-friendly to controls, IoT MQTT Panel Application to interact with the smart devices and also Telegram Chatbot. These user interfaces gives users to access to the virtual smart condominium by just using mobile phone. Telegram Chatbot make the use of the smart condominium become easier to manage by offering real-time notification and control commands, which improves the user experience.

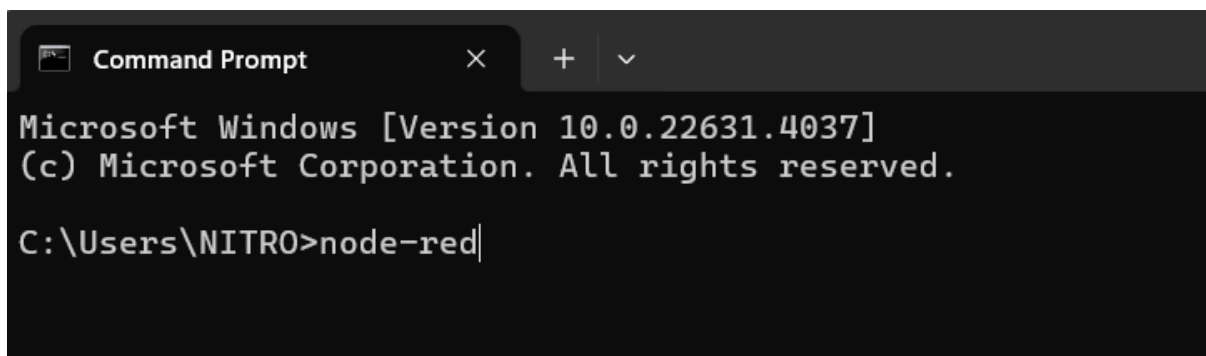
CHAPTER 5

System Implementation

5.1 Software Setup

5.1.1 Software Setup - Node-RED

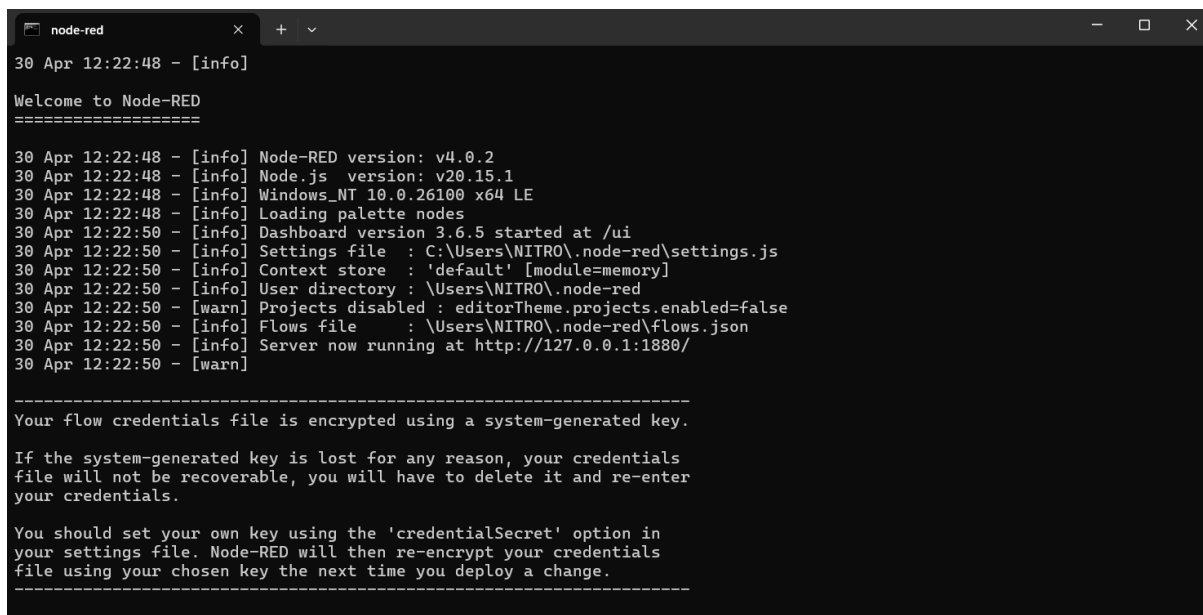
To develop the smart condominium simulation system, it requires to install Node-RED into computer. First go the the official website, <https://nodejs.org/en/> to download the latest version of Node.js. After the installation of Node.js, open Command Prompt and type the following command “**npm install -g --unsafe-perm node-red**” and Node-RED will be successfully installed. After that, user can open a new Command Prompt or use in the same Command Prompt and type “**node-red**” to execute the Node-RED server (*Figure 5.1.1.1 and Figure 5.1.1.2*).

A screenshot of a Windows Command Prompt window. The title bar reads 'Command Prompt' with standard window controls. The text inside shows the Windows version '10.0.22631.4037' and copyright information for Microsoft Corporation. The command prompt shows the current directory as 'C:\Users\NITRO' and the command 'node-red' has been entered, with a cursor at the end of the line.

```
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\NITRO>node-red|
```

Figure 5.1.1.1 Type “node-red” to execute Node-RED



```

node-red
x + v
30 Apr 12:22:48 - [info]
Welcome to Node-RED
=====
30 Apr 12:22:48 - [info] Node-RED version: v4.0.2
30 Apr 12:22:48 - [info] Node.js version: v20.15.1
30 Apr 12:22:48 - [info] Windows_NT 10.0.26100 x64 LE
30 Apr 12:22:48 - [info] Loading palette nodes
30 Apr 12:22:50 - [info] Dashboard version 3.6.5 started at /ui
30 Apr 12:22:50 - [info] Settings file : C:\Users\NITRO\.node-red\settings.js
30 Apr 12:22:50 - [info] Context store : 'default' [module=memory]
30 Apr 12:22:50 - [info] User directory : \Users\NITRO\.node-red
30 Apr 12:22:50 - [warn] Projects disabled : editorTheme.projects.enabled=false
30 Apr 12:22:50 - [info] Flows file : \Users\NITRO\.node-red\flows.json
30 Apr 12:22:50 - [info] Server now running at http://127.0.0.1:1880/
30 Apr 12:22:50 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

```

Figure 5.1.1.2 Node-RED has successfully executed.

After the launching in command prompt, go to web browser and type “**localhost:1880/**” to access into the Node-RED panel **Figure 5.1.1.3** the Node-RED Flow Editor Panel. To develop the system, users are required to install the following palettes shown in **Figure 5.1.1.4**.

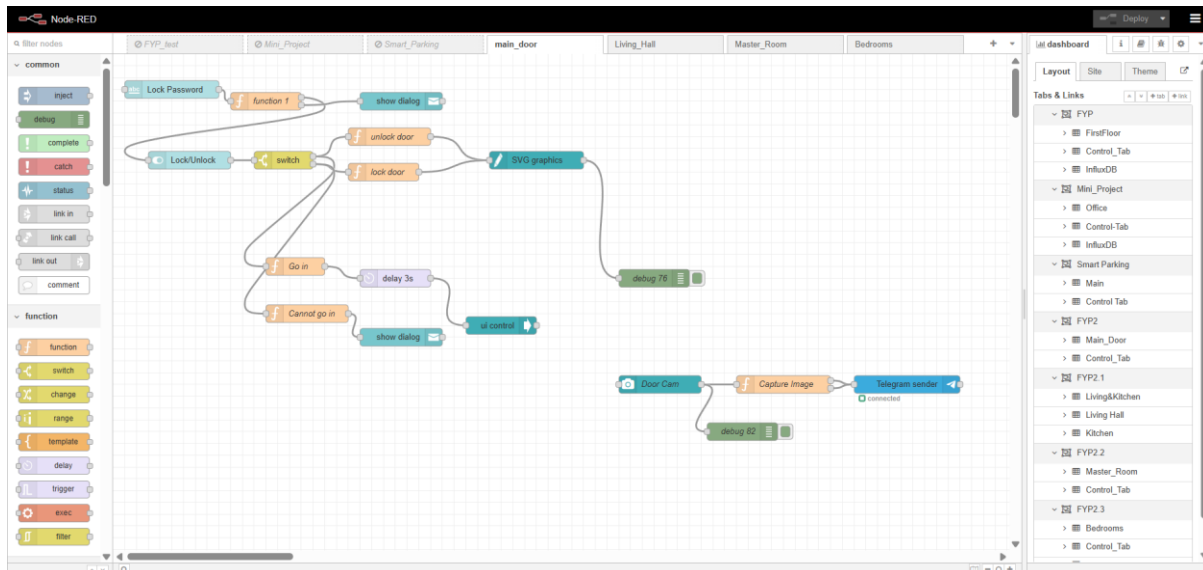


Figure 5.1.1.3 Node-RED Flow Editor Panel

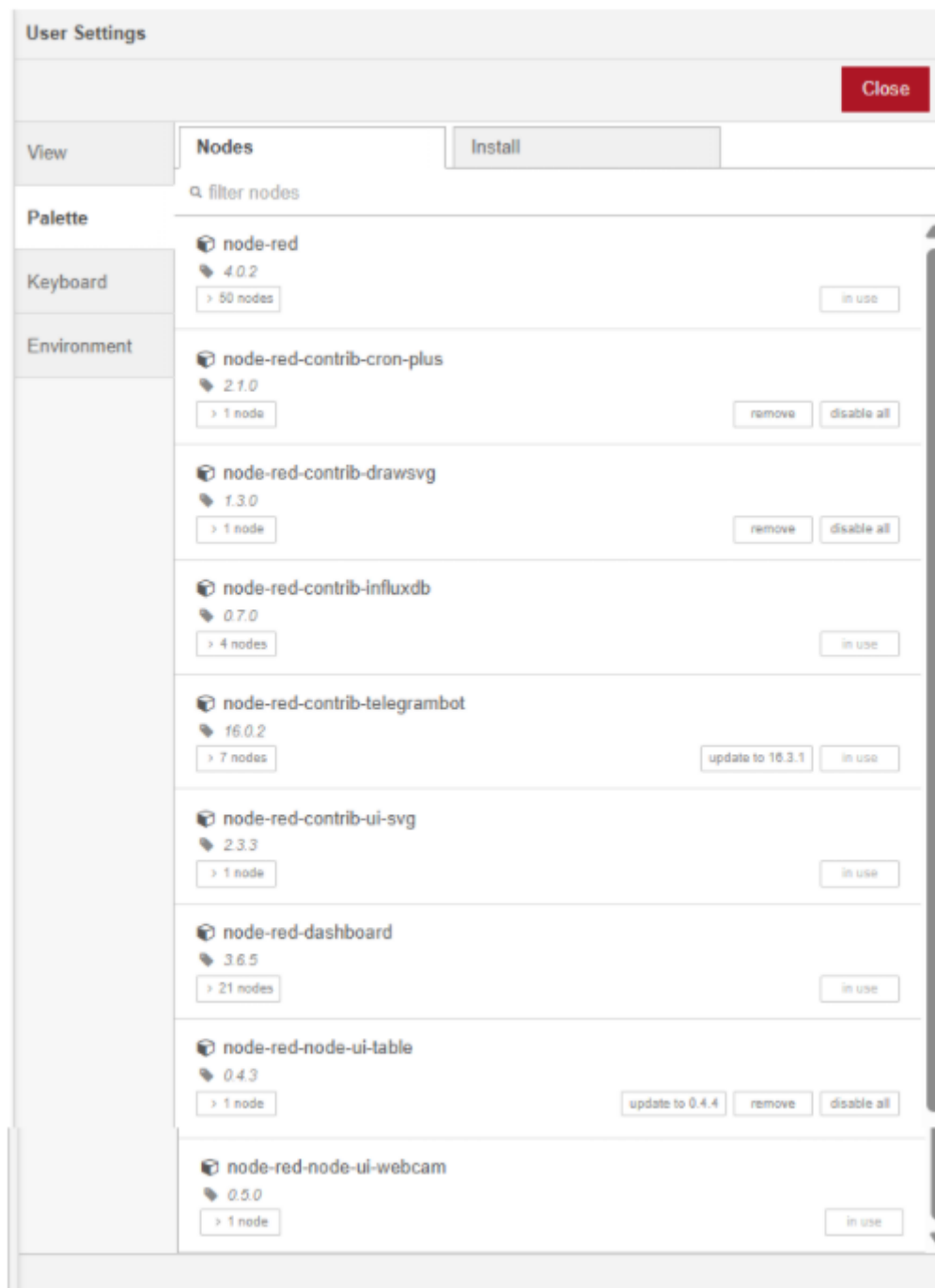


Figure 5.1.1.4 *Palette to be installed in Node-RED*

5.1.2 Software Setup – InfluxDB

The installation process for the InfluxDB is similar to installing Node-RED. First, visit the URL: <https://docs.influxdata.com/influxdb/v2/install/> to install InfluxDB. Then select Windows and click install (*Figure 5.1.2.1*).

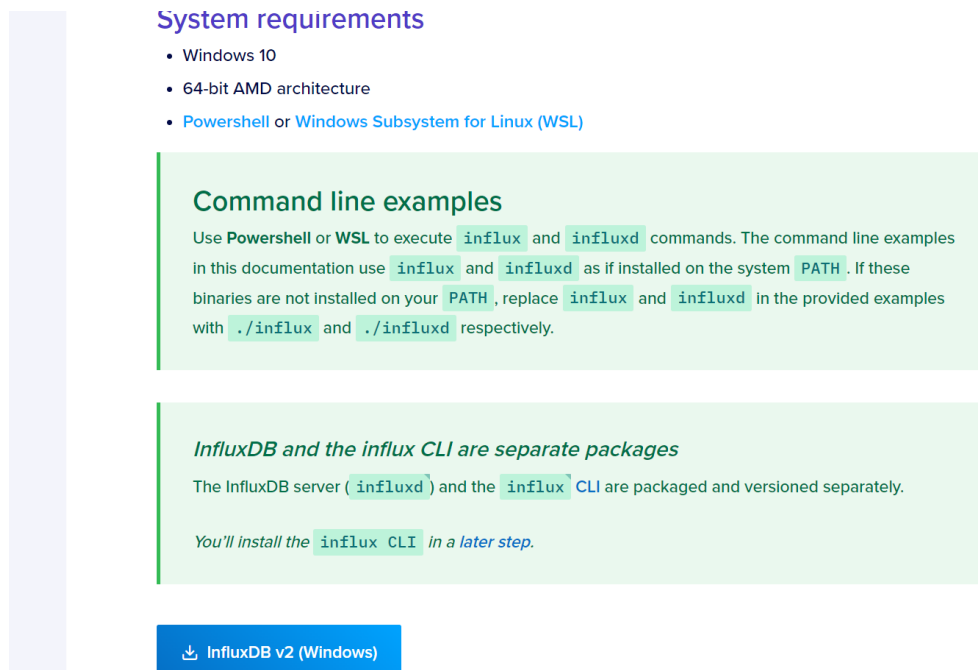


Figure 5.1.2.1 Download InfluxDB in Windows

After installed, open Command Prompt and find the downloaded InfluxDB file, drag the “**influx.exe**” into the Command Prompt and press enter to start InfluxDB server. Then go to web browser and type **http://localhost:8086/** to enter the InfluxDB server.

```
C:\Windows\System32\cmd.exe
": 1}
2025-04-30T06:53:31.459957Z    info    TSM compaction (end) {"log_id": "0wDNOrQ0000", "service": "storage-engine", "engine": "tsm1", "tsm1_strategy": "full", "tsm1_optimize": false, "op_name": "tsm1_compact_group", "op_event": "end", "op_elapsed": "17.487ms"}
2025-04-30T06:53:31.518454Z    info    Compacted file {"log_id": "0wDNOrQ0000", "service": "storage-engine", "engine": "tsm1", "tsm1_strategy": "full", "tsm1_optimize": false, "op_name": "tsm1_compact_group", "tsm1_index": 0, "tsm1_file": "C:\\Users\\NITRO\\.influxdbv2\\engine\\data\\f755b01d695918b9\\autogen\\I\\000000005-000000002.tsm.tmp"}
2025-04-30T06:53:31.518454Z    info    Finished compacting files {"log_id": "0wDNOrQ0000", "service": "storage-engine", "engine": "tsm1", "tsm1_strategy": "full", "tsm1_optimize": false, "op_name": "tsm1_compact_group", "tsm1_files_n": 1}
2025-04-30T06:53:31.518982Z    info    TSM compaction (end) {"log_id": "0wDNOrQ0000", "service": "storage-engine", "engine": "tsm1", "tsm1_strategy": "full", "tsm1_optimize": false, "op_name": "tsm1_compact_group", "op_event": "end", "op_elapsed": "76.512ms"}
2025-04-30T06:53:40.519525Z    info    index opened with 8 partitions {"log_id": "0wDNOrQ0000", "service": "storage-engine", "index": "tsi"}
2025-04-30T06:53:40.520613Z    info    loading changes (start) {"log_id": "0wDNOrQ0000", "service": "storage-engine", "engine": "tsm1", "op_name": "field indices", "path": "C:\\Users\\NITRO\\.influxdbv2\\engine\\data\\f755b01d695918b9\\autogen\\3\\fields.idx", "op_event": "start"}
2025-04-30T06:53:40.521158Z    info    loading changes (end) {"log_id": "0wDNOrQ0000", "service": "storage-engine", "engine": "tsm1", "op_name": "field indices", "path": "C:\\Users\\NITRO\\.influxdbv2\\engine\\data\\f755b01d695918b9\\autogen\\3\\fields.idx", "op_event": "end", "op_elapsed": "0.545ms"}
2025-04-30T06:53:40.521698Z    info    Reindexing TSM data {"log_id": "0wDNOrQ0000", "service": "storage-engine", "engine": "tsm1", "db_shard_id": 3}
2025-04-30T06:53:40.521698Z    info    Reindexing WAL data {"log_id": "0wDNOrQ0000", "service": "storage-engine", "engine": "tsm1", "db_shard_id": 3}
2025-04-30T06:53:40.543226Z    info    saving field index changes (start) {"log_id": "0wDNOrQ0000", "service": "storage-engine", "engine": "tsm1", "op_name": "MeasurementFieldSet", "op_event": "start"}
2025-04-30T06:53:40.544881Z    info    saving field index changes (end) {"log_id": "0wDNOrQ0000", "service": "storage-engine", "engine": "tsm1", "op_name": "MeasurementFieldSet", "op_event": "end", "op_elapsed": "2.187ms"}
```

Figure 5.1.2.2 Command Prompt running InfluxDB

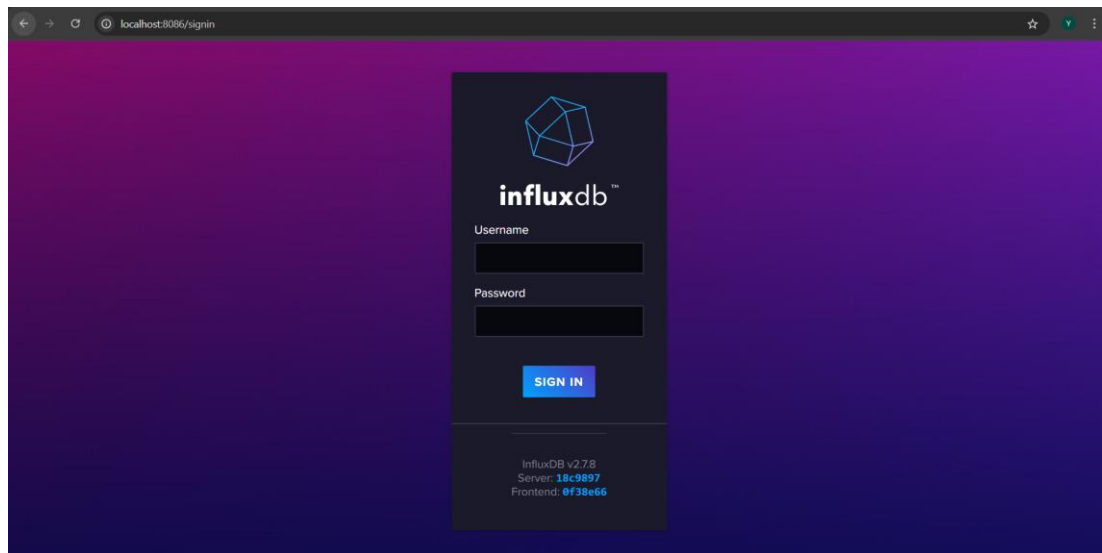


Figure 5.1.2.3 InfluxDB server

After login, go to the API Tokens tab and generate a new API token (*Figure 5.1.2.4*). Make sure to store properly for the API token as the token will only appear once. This token allows Node-RED access to the InfluxDB database.

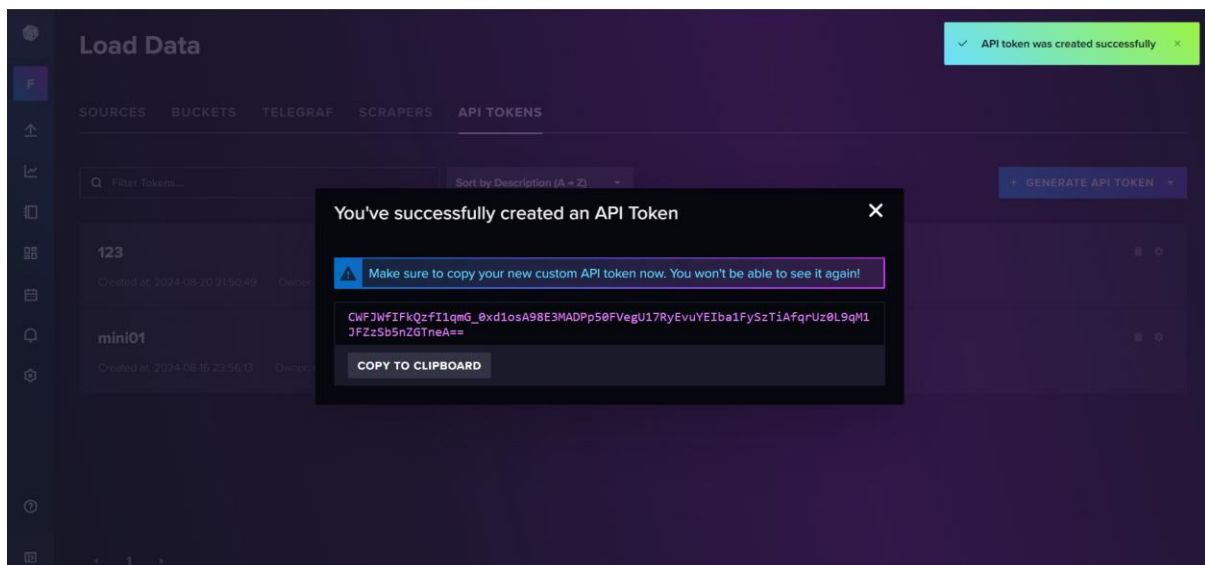


Figure 5.1.2.4 API Token

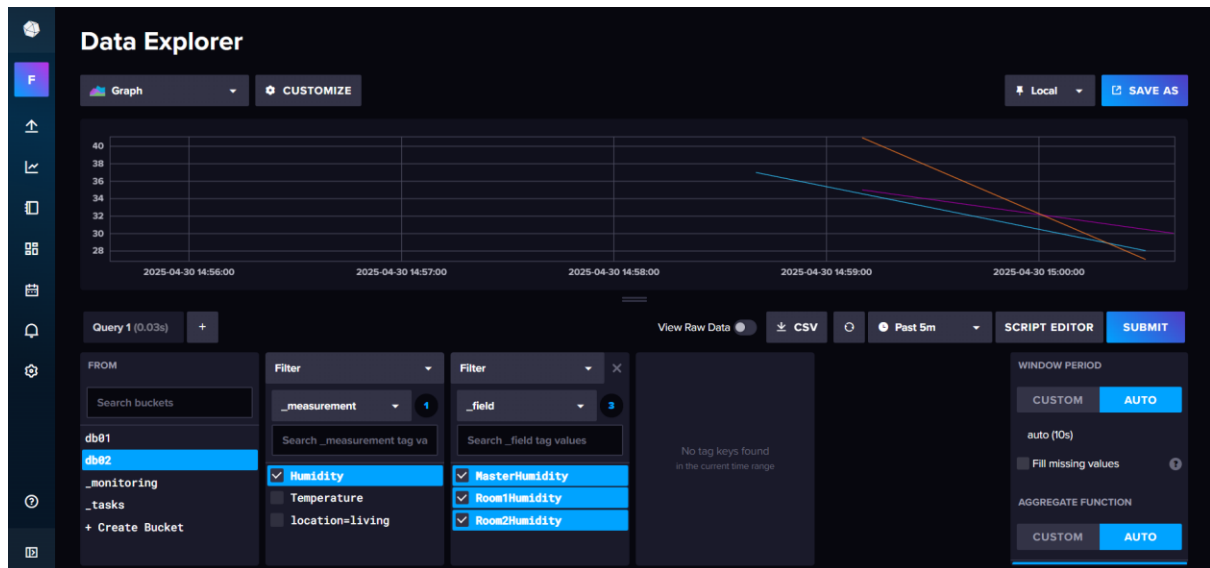


Figure 5.1.2.5 InfluxDB Dashboard

5.1.3 Software Setup (Mobile) – IoT MQTT Panel

Install the IoT MQTT Panel via Google App Store/Apple Play Store. If Huawei users, they are required to download mobile simulator in computer as this application currently still doesn't support in Huawei App Gallery. **Figure 5.1.3.1** shows the correct version of IoT MQTT Panel Application.

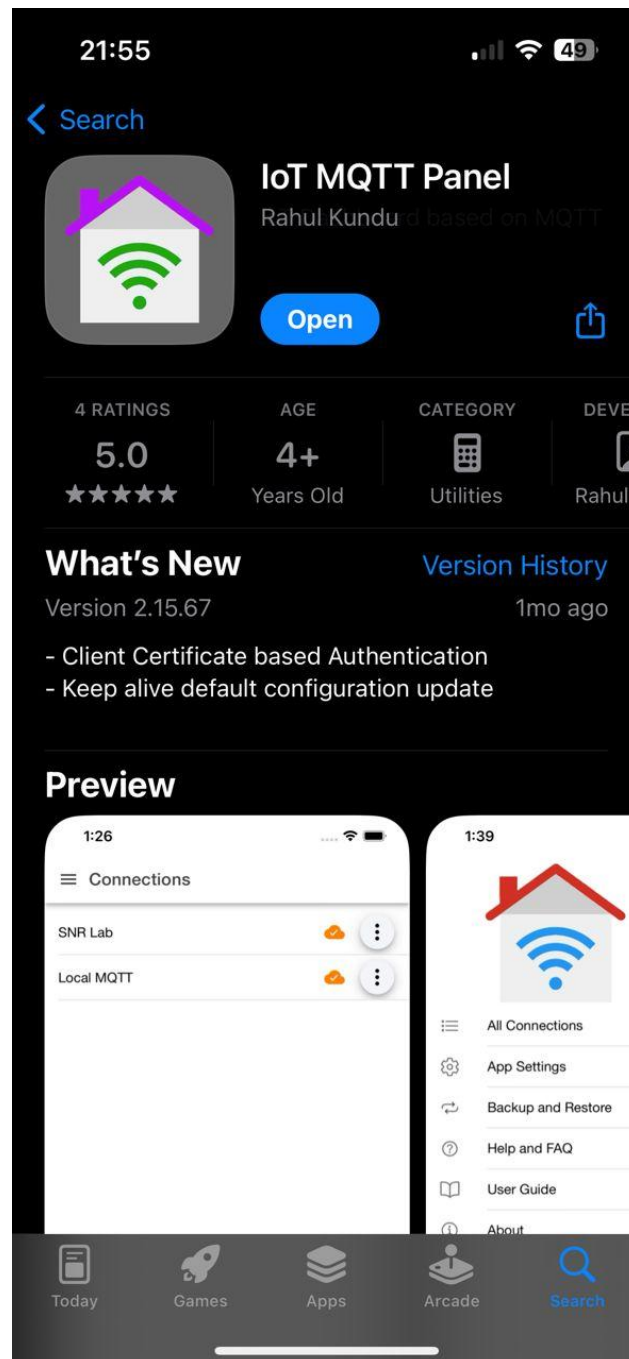


Figure 5.1.3.1 IoT MQTT Panel Application in Apple Play Store

After the installation, open the application and press the “+” symbol at the bottom right of the screen, then enter the Connection Name and Broker Web. **Figure 5.1.3.3** shows the connection for the smart condo simulation system.

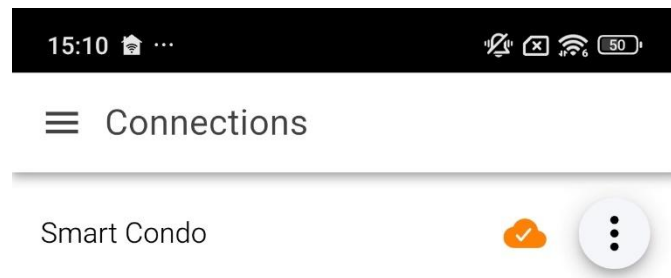


Figure 5.1.3.2 Press the “+” Symbol to add new connections

15:10

← Edit Connection

Connection name *
Smart Condo

Client ID
k4Dc25zmbn

Broker Web/IP address *
broker.hivemq.com

Port *
1883

Network protocol
TCP

Add Dashboard

Living n Kitchen

Room

Additional options

CANCEL SAVE

We provide Direct Bank-In to your account

Figure 5.1.3.3 Connection for Smart Condo Simulation System

5.1.4 Software Setup (Mobile) – Telegram

Download Telegram in Google App Store/Apple Play Store/Huawei App Gallery. Once downloaded, launch Telegram and search for **@BotFather**, be careful and make sure to look for the correct which has a blue tick beside to prevent any problems.



Figure 5.1.4.1 BotFather with blue tick beside

After that type “/start” and send to activate BotFather (*Figure 5.1.4.2*). Then type “/newbot” to create a new telegram bot with the following instruction given by BotFather (*Figure 5.1.4.3*).

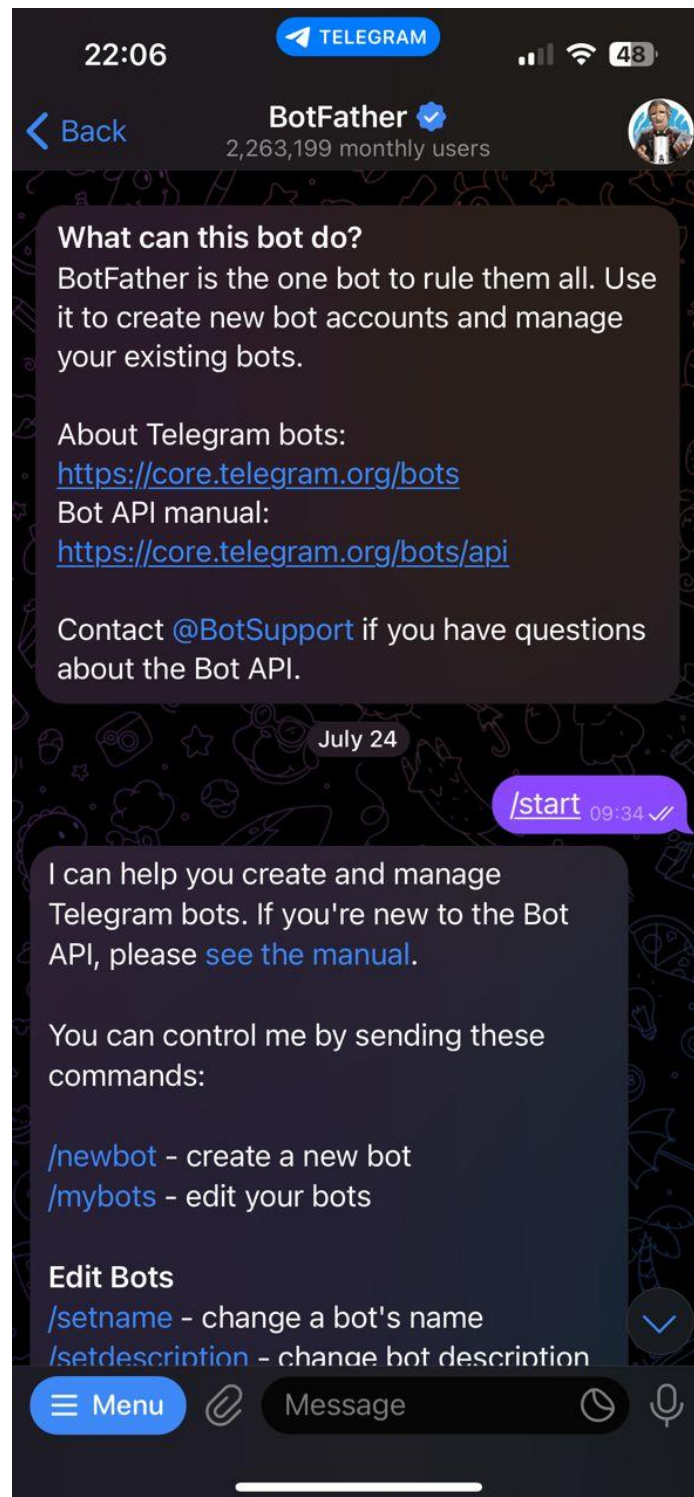


Figure 5.1.4.2 “/start” to activate BotFather

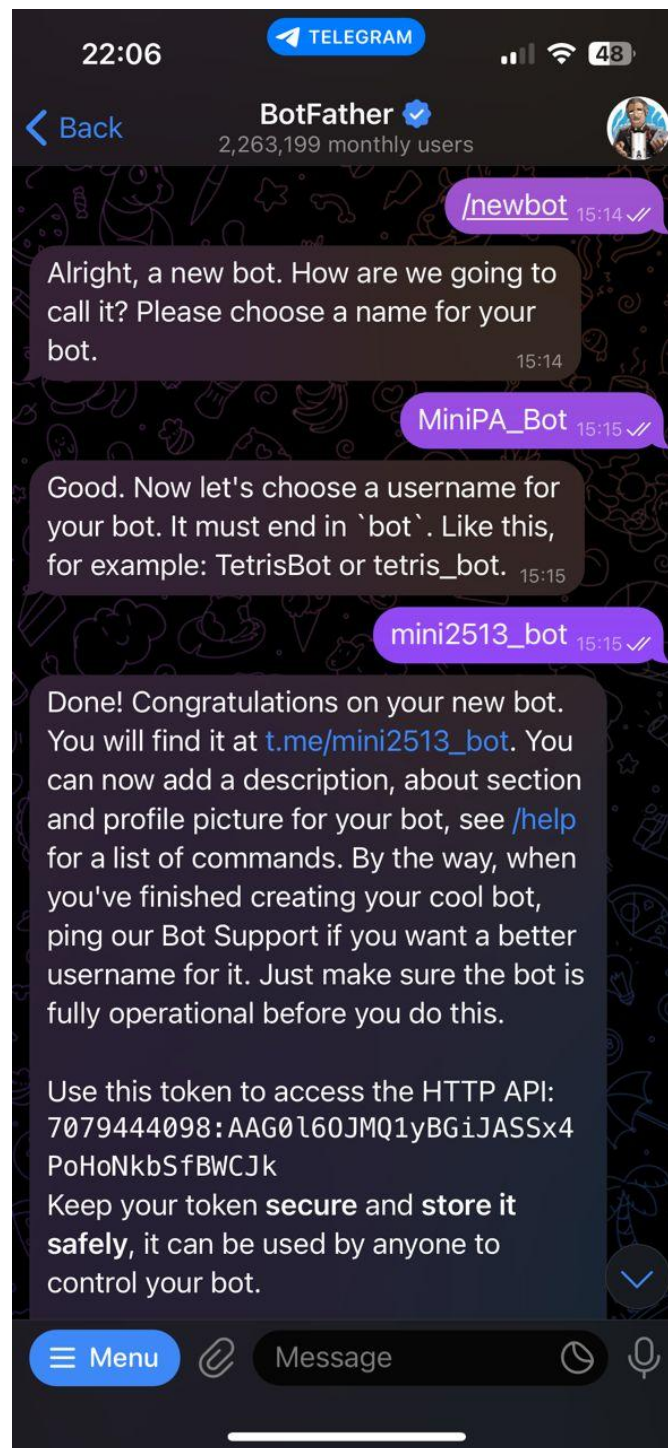


Figure 5.1.4.3 “/newbot” and follows the instruction to create new bot

Keep the API given by BotFather as this API will be used in the Node-RED to enable the interaction with Telegram Bot that user created. Lastly, search for the bot’s name that user created type “/start” command and enter to activate the Telegram Bot that user created.

5.2 Setting and Configuration

To ensure the proper functionality, the system cannot just be configured at the application level only, each component needs to be integrated properly into the Node-RED environment. Node-RED serves as the main platform to manage the logic and communication of the flow between different services.

Special nodes such as InfluxDB nodes, MQTT Panel nodes and the Telegram Bot nodes must be configured properly within Node-RED. This includes the connection parameters, authentication settings and the logic of the flow to ensure these services interact as required. The system will be unable to function if the linking of these nodes and function nodes in Node-RED is not properly configured.

5.2.1 Setting and Configuring InfluxDB node in Node-RED

To let InfluxDB to receive data from the system, the “influxdb out” node must be added and connected to the “Get Temp/Humid Status” function node within Node-RED (*Figure 5.2.1.1*). Once connected, double-click the “**influxdb out**” node to access its properties. In the configuration settings, specify the appropriate parameters such as the connection name, InfluxDB version, URL, and access token. The version should correspond to the installed InfluxDB instance which typically is **version 2.0**, and the default URL is usually **http://localhost:8086**, the access token should be the API token generated during the initial InfluxDB setup process.

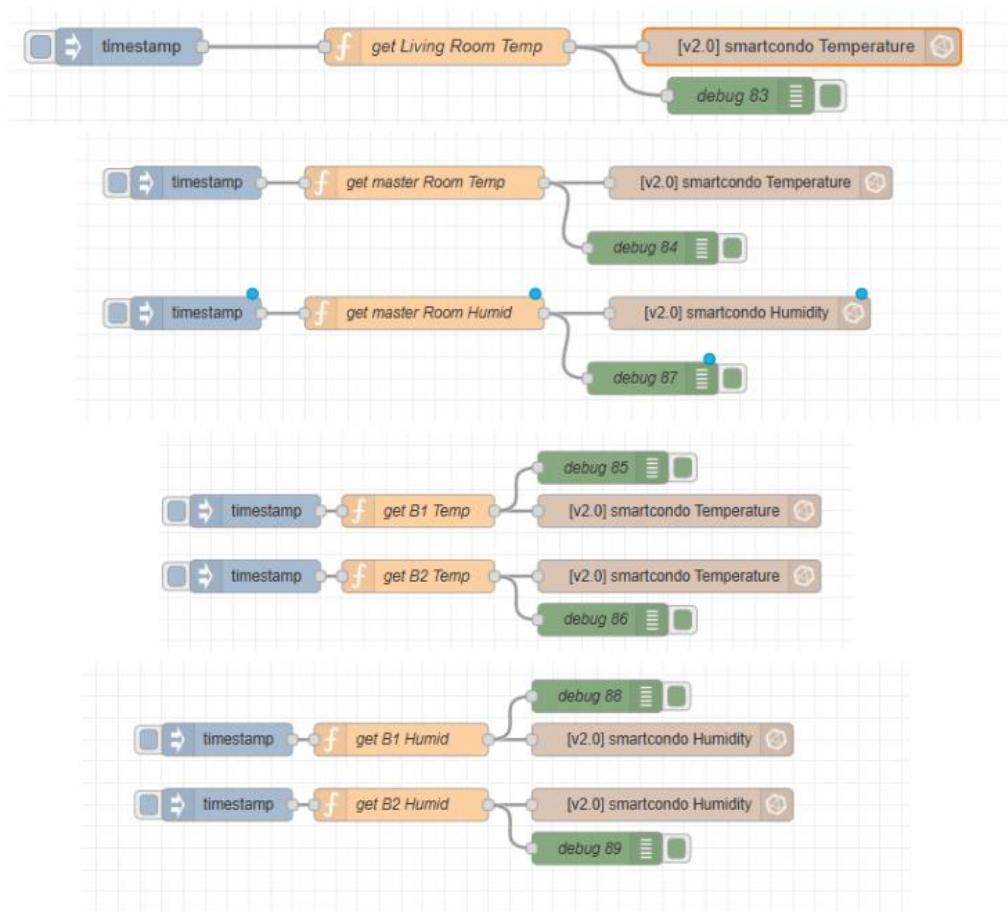


Figure 5.2.1.1 Flows for InfluxDB to get data of Temperature and Humidity

Edit influxdb out node > Edit influxdb node

Delete Cancel Update

Properties

Name smartcondo

Version 2.0

URL http://localhost:8086

Token

Connection timeout (seconds) 10

☒ Verify server certificate

Figure 5.2.1.2 Configuration Setting for influxdb node

After configuring the InfluxDB node properties, the system successfully established a connection with the server using the provided API token. Next, the **"influxdb out"** node must be further configured by specifying the Organization, Bucket, and Measurement fields. It is important to note that the Organization and Bucket values can be fixed as per the initial InfluxDB setup, while the Measurement field can be customized depending on the type of data.

The screenshot shows the 'Edit influxdb out node' configuration window. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below these is a 'Properties' section with a gear icon and three sub-panels (Properties, Data, Visuals). The 'Properties' sub-panel is active, showing the following fields:

- Name:** A text input field containing 'Name'.
- Server:** A dropdown menu showing '[v2.0] smartcondo' with a plus icon to the right.
- Organization:** A text input field containing 'FICT'.
- Bucket:** A text input field containing 'db02'.
- Measurement:** A text input field containing 'Temperature'.
- Time Precision:** A dropdown menu showing 'Seconds (s)'.

Figure 5.2.1.3 Configuration for Temperature Measurement.

The screenshot shows the 'Edit influxdb out node' configuration window, similar to the previous one, but with the 'Measurement' field set to 'Humidity'. The other fields remain the same:

- Name:** A text input field containing 'Name'.
- Server:** A dropdown menu showing '[v2.0] smartcondo' with a plus icon to the right.
- Organization:** A text input field containing 'FICT'.
- Bucket:** A text input field containing 'db02'.
- Measurement:** A text input field containing 'Humidity'.
- Time Precision:** A dropdown menu showing 'Seconds (s)'.

Figure 5.2.1.4 Configuration for Humidity Measurement

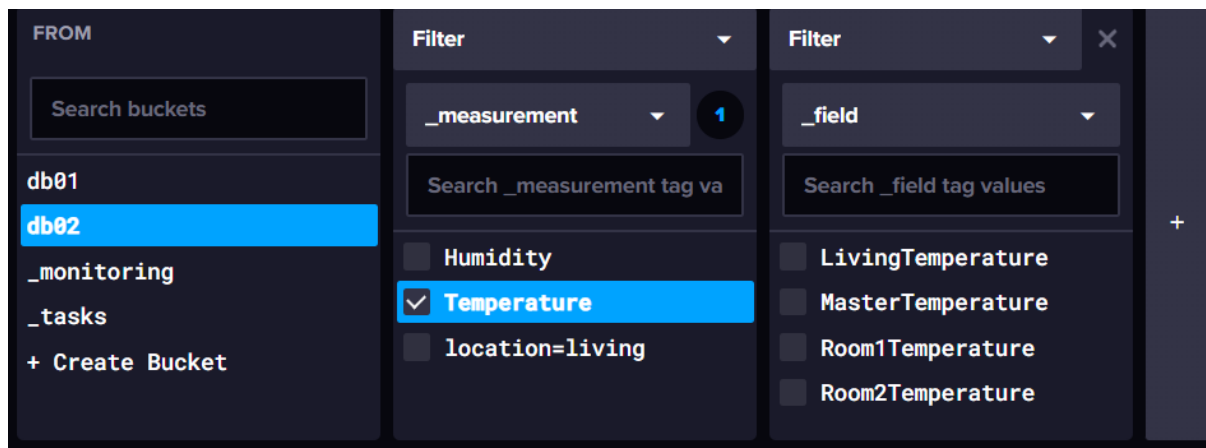


Figure 5.2.1.5 Temperature Measurement shown in InfluxDB

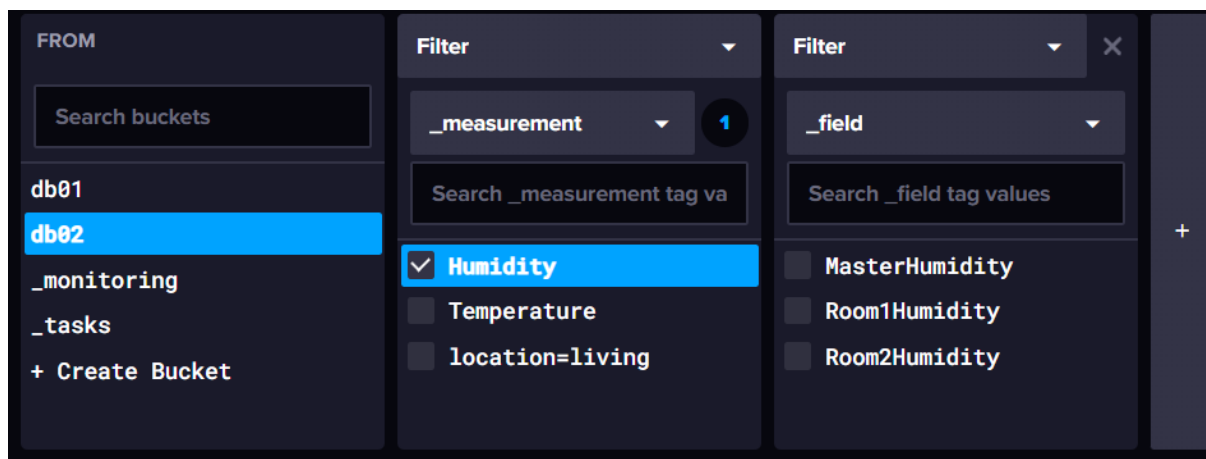


Figure 5.2.1.6 Humidity Measurement shown in InfluxDB

5.2.2 Setting and Configuring MQTT Panel Application and node in Node-RED

To let MQTT able to send command from the application to the system, the “**mqtt in**” node must be added and connected to the “**json**” node and link to **switch node** (*Figure 5.2.2.1*). or directly to the node that wanted to connect within Node-RED (*Figure 5.2.2.2*). Once connected, double-click the “**mqtt in**” node to access to broker node properties. In the configuration settings, specify the appropriate parameters such as the connection name, connection server, and protocol. The protocol is typically set as **MQTT V3.1.1**, and the server set as “**broker.hivemq.com**”.

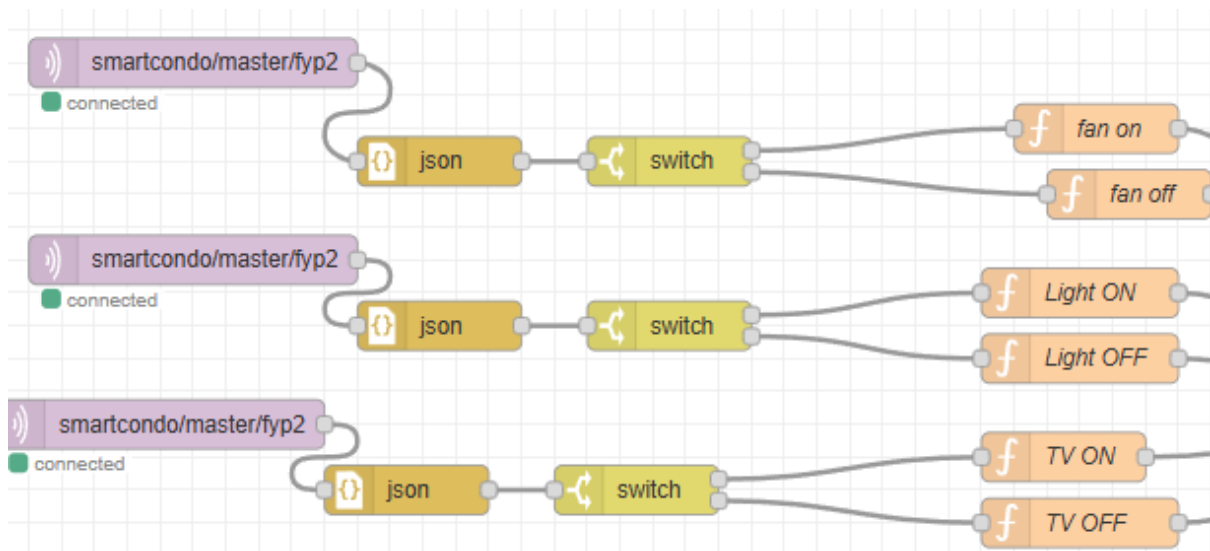


Figure 5.2.2.1 Flows for “mqtt in” connection with json and switch node

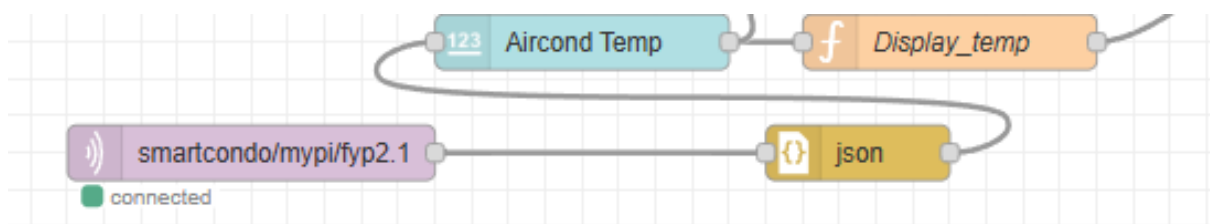


Figure 5.2.2.2 Flow for “mqtt in” connected with json and directly without switch node

Edit mqtt in node > Edit mqtt-broker node

Delete Cancel Update

Properties

Name Hivemq Smart Condo

Connection Security Messages

Server broker.hivemq.com Port 1883

☒ Connect automatically

☐ Use TLS

Protocol MQTT V3.1.1

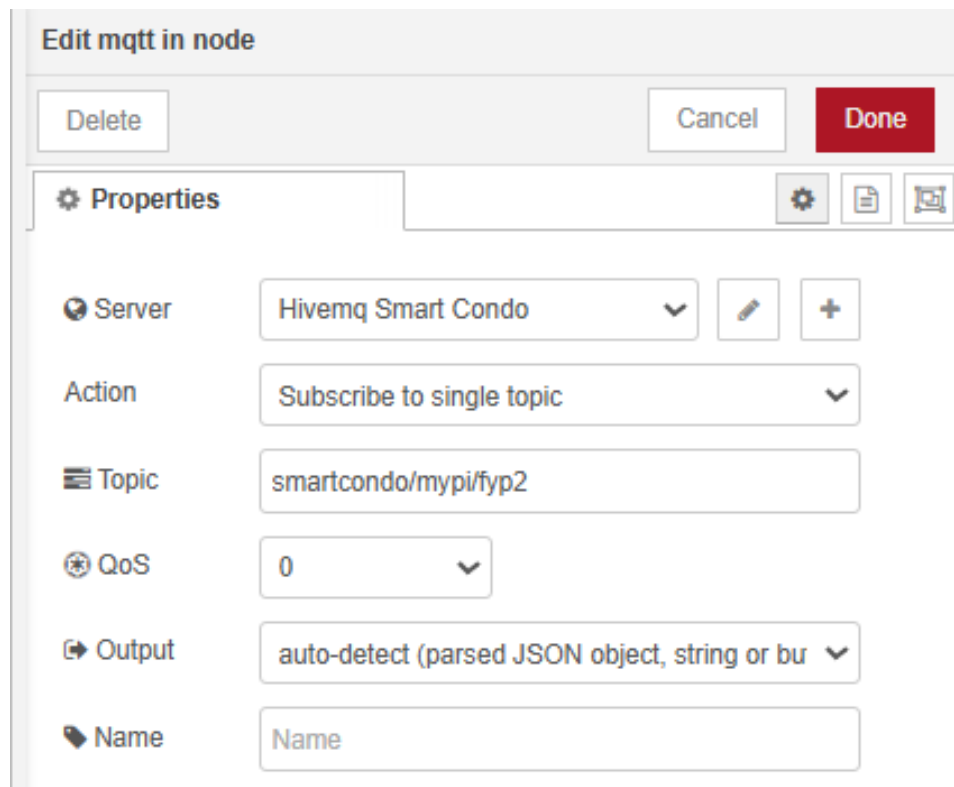
Client ID Leave blank for auto generated

Keep Alive 60

Session ☒ Use clean session

Figure 5.2.2.3 Configuration for mqtt-broker node

The “**mqtt in**” node must be configured to subscribe to the topic “**smartcondo/myypi/fyp2**”. The **Quality of Service (QoS)** level should be set to **0**, and the **Output** option can remain as **Auto-detect** to allow automatic handling of the incoming message format (*Figure 5.2.2.4*). Following this, a **Switch** node should be added to evaluate the MQTT message payload. Set the property to **msg.payload.act** and define two conditional outputs: one for the value “**on**” and another for “**off**”. This setup enables the flow to branch based on the received command status.



Edit mqtt in node

Delete Cancel Done

Properties

Server Hivemq Smart Condo

Action Subscribe to single topic

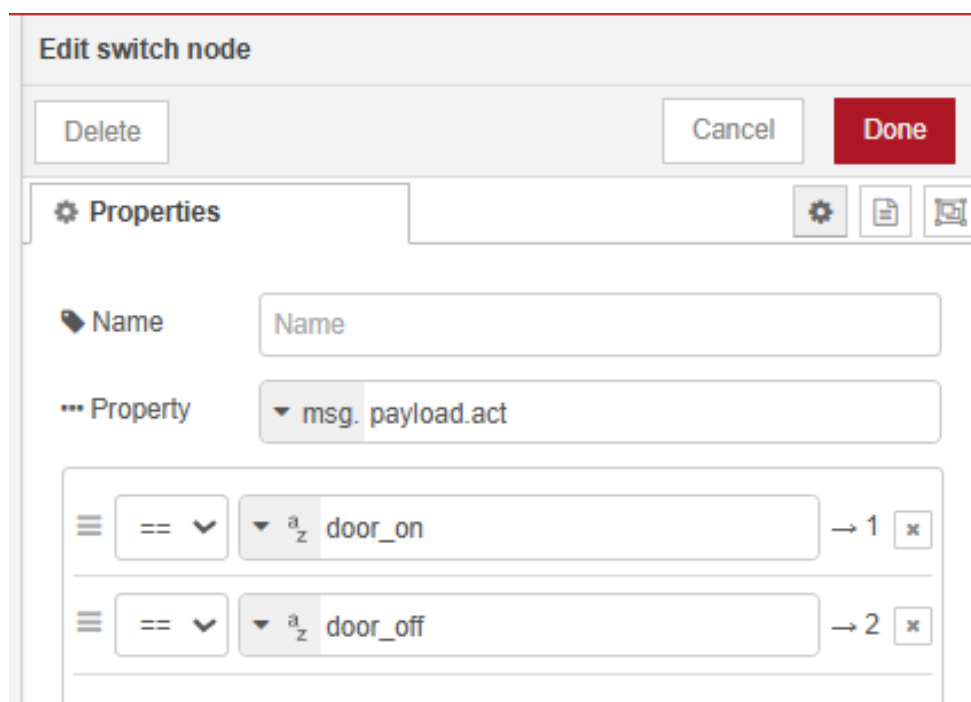
Topic smartcondo/mypii/fyp2

QoS 0

Output auto-detect (parsed JSON object, string or bu)

Name Name

Figure 5.2.2.4 Configuration for “mqtt in” node



Edit switch node

Delete Cancel Done

Properties

Name Name

Property msg.payload.act

door_on → 1

door_off → 2

Figure 5.2.2.5 Configuration for switch node

In the IoT MQTT Panel application, select the previously configured connection. Then, tap the “+” icon to add a new panel and choose the appropriate panel type based on the desired functionality. Assign a relevant name to the panel and set the topic to match the one configured in the “**mqtt in**” node within Node-RED.

If the panel is configured as a **Switch Panel**, it is typically linked to a corresponding **Switch Node** in the flow. In this case, define the payload accordingly:

- **Payload On:** {"act": "payload_on"}
- **Payload Off:** {"act": "payload_off"}

These payload structures ensure the communication between the mobile application and the Node-RED flow. However, if the panel is not linked to a switch node, simply configure the panel by assigning a name and matching the topic to match the one configured in the “**mqtt in**”, without additional payload formatting.

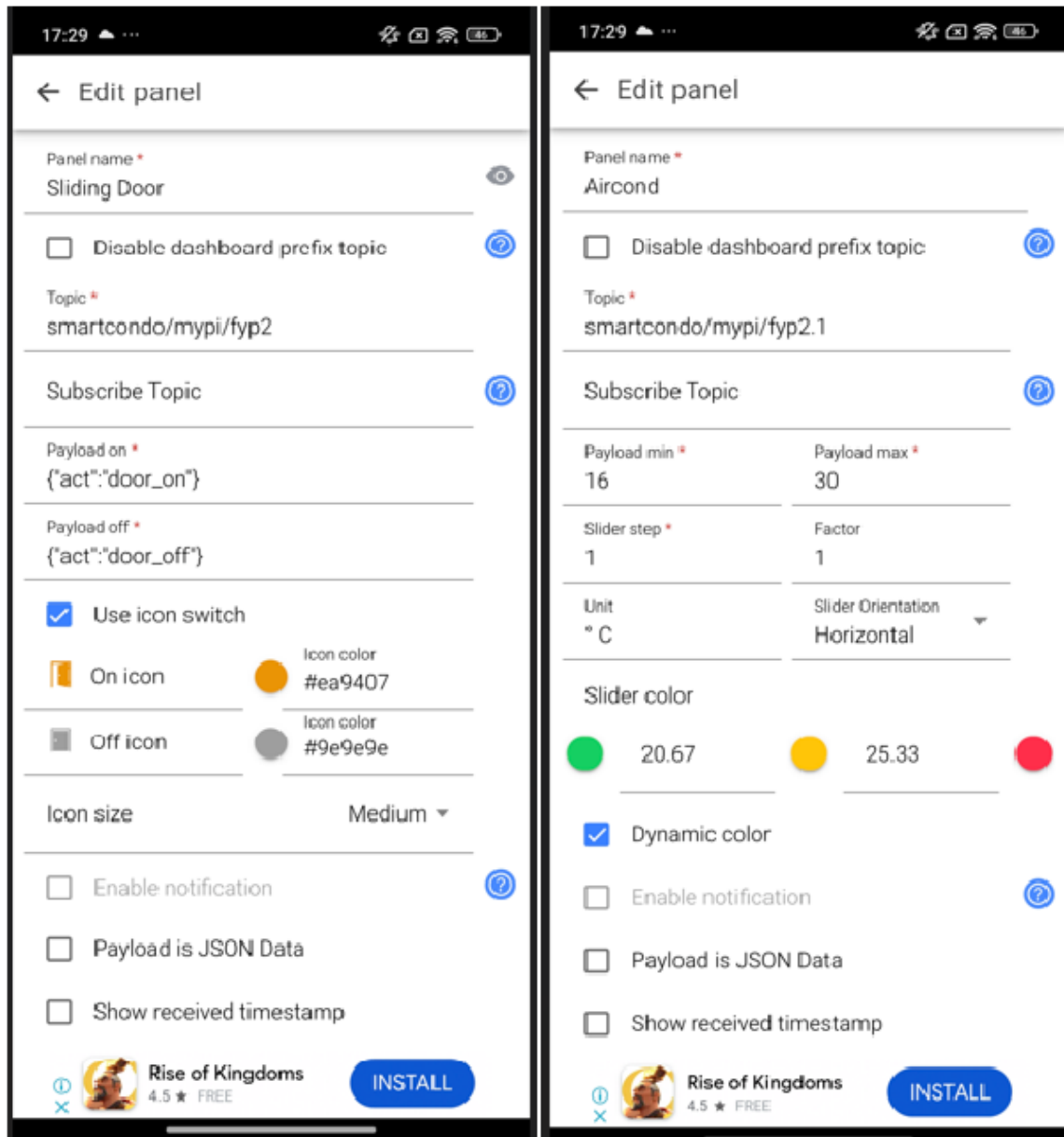


Figure 5.2.2.6 Panel configuration with switch node linked and without switch node linked

5.2.3 Setting and Configuring Telegram in Node-RED

To enable the Telegram Bot to send notification from the system, the “**Telegram Sender**” node must be added and connected to the corresponding function node that generate the message content. Additionally, a “**Callback Query**” node will be included and linked to a **switch** node to handle user interactions through callback queries. Once the nodes are properly linked, double-click on any **Telegram-related node** to open the **Telegram Bot configuration panel**. In the configuration settings, enter the **Bot Name** and **Token** that were previously generated during the initial setup of the Telegram Bot.

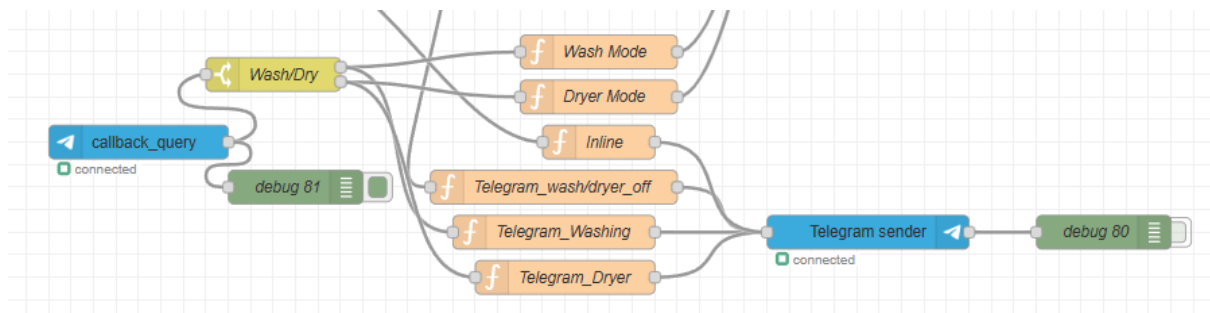


Figure 5.2.3.1 Flow for Telegram “callback_query” and Telegram sender node

The screenshot shows the 'Edit telegram bot node' configuration panel. At the top, there are 'Delete', 'Cancel', and 'Update' buttons. Below is a 'Properties' section with a gear icon and a document icon. The 'Bot-Name' field contains 'MiniPA_Bot'. The 'Token' field contains '7079444098:AAG0I6OJMQ1yBGiJASSx4PoHoNkbSfBWCJk'. A yellow tip box says: 'Tip: If you don't have a token yet, you can create a new one here: @BotFather.' Below this are three optional fields: 'Users' (with placeholder '(Optional list of authorized user names e.g.: hugo,sepp,egon)'), 'ChatIds' (with placeholder '(Optional list of authorized chat-ids e.g.: -1234567,2345678,-3456789)'), and 'Server URL' (with placeholder '(Optional URL for proxying and testing e.g.: https://api.telegram.org)').

Figure 5.2.3.2 Configuration for Telegram bot node

5.3 System Operation

5.3.1 System Operation of Main Door

This system uses the Node-RED dashboard with SVG graphics to perform the simulation process. **Figure 5.3.1.1** shows the Main Door of the Smart Condo with the “Control Tab”. The color of the handle will show red to indicate the door is **in lock**. **Figure 5.3.1.2** shows after entering the password “**123456**” the door handle will become green color to indicate the door has been unlocked. It also has a camera to capture down any suspicious person and send via Telegram Bot.

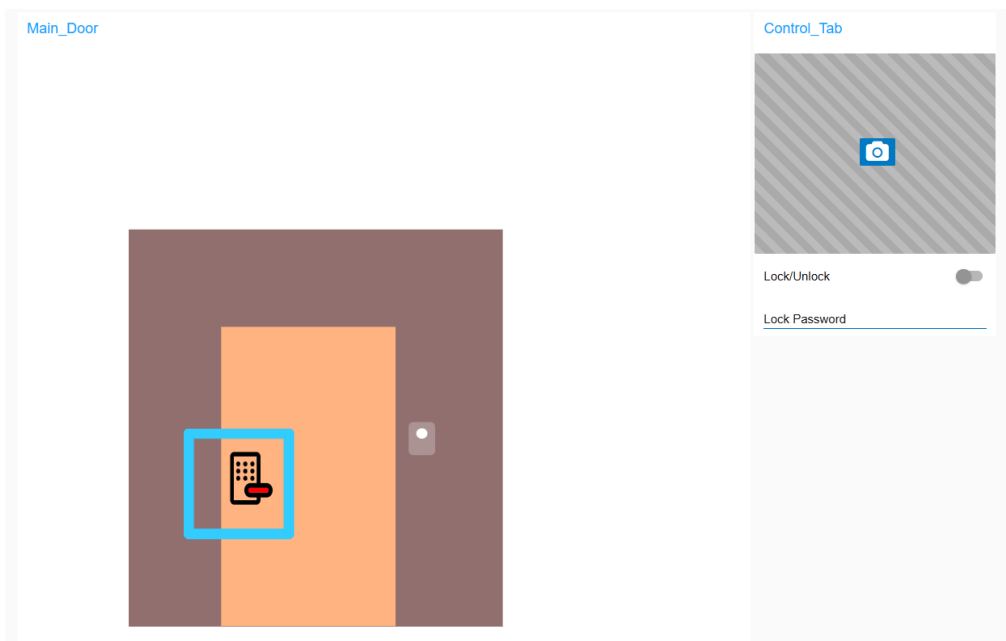


Figure 5.3.1.1 Door Locked

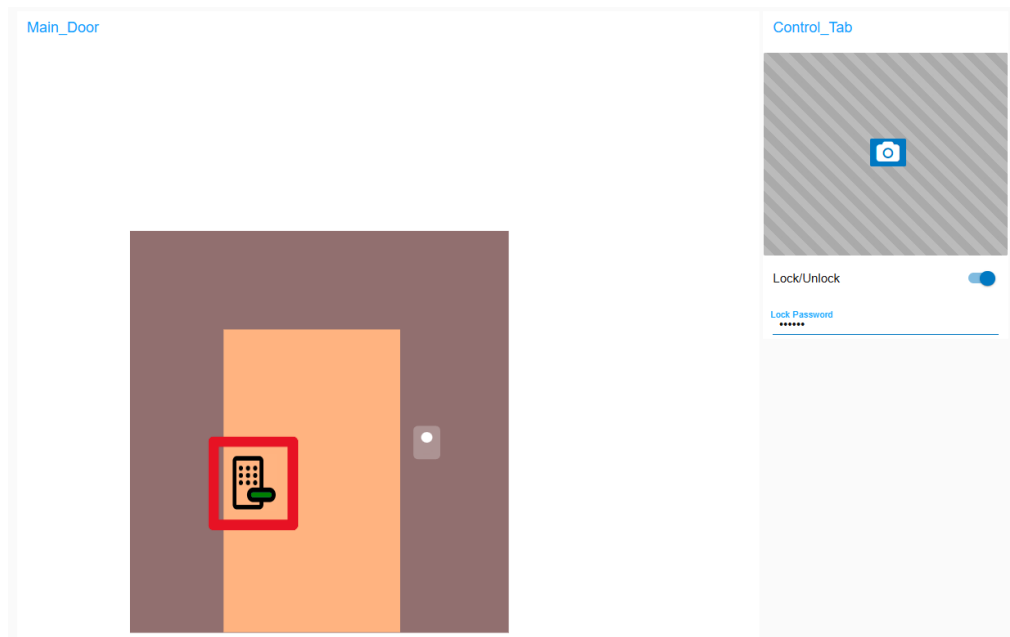


Figure 5.3.1.2 Door Unlocked after entering the correct password

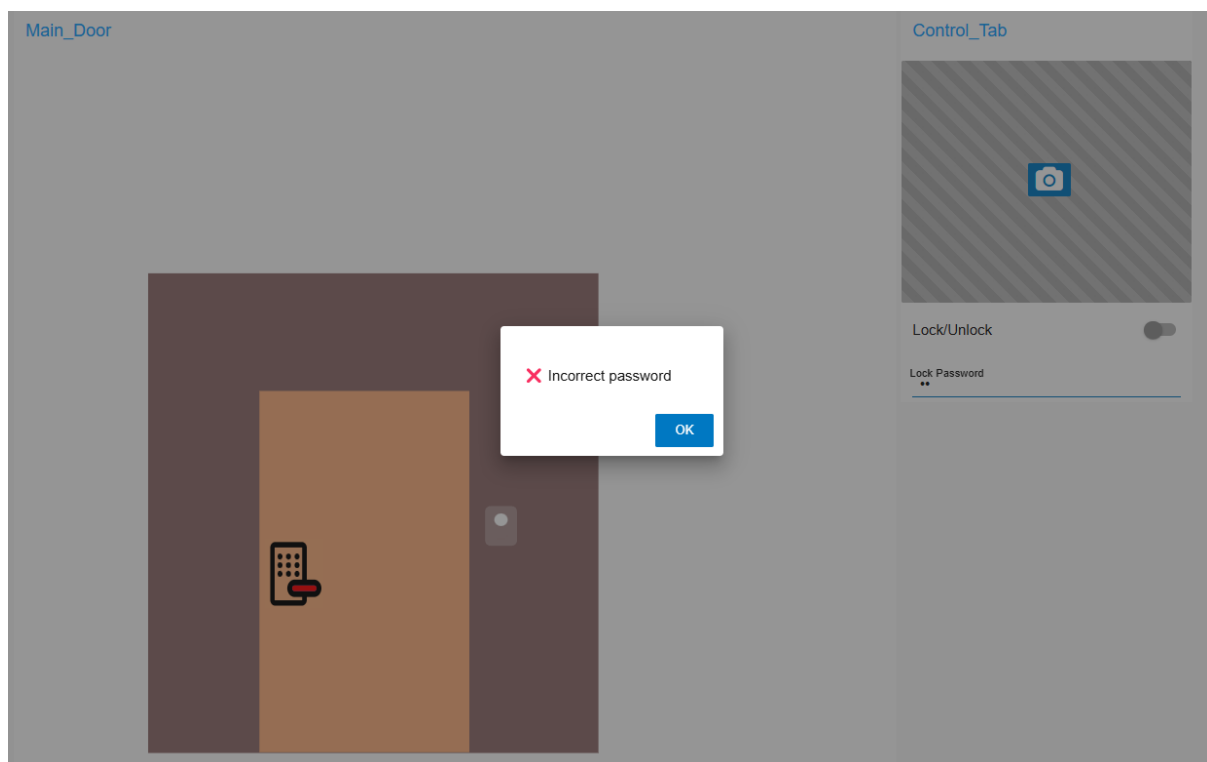


Figure 5.3.1.3 Notification shows incorrect password

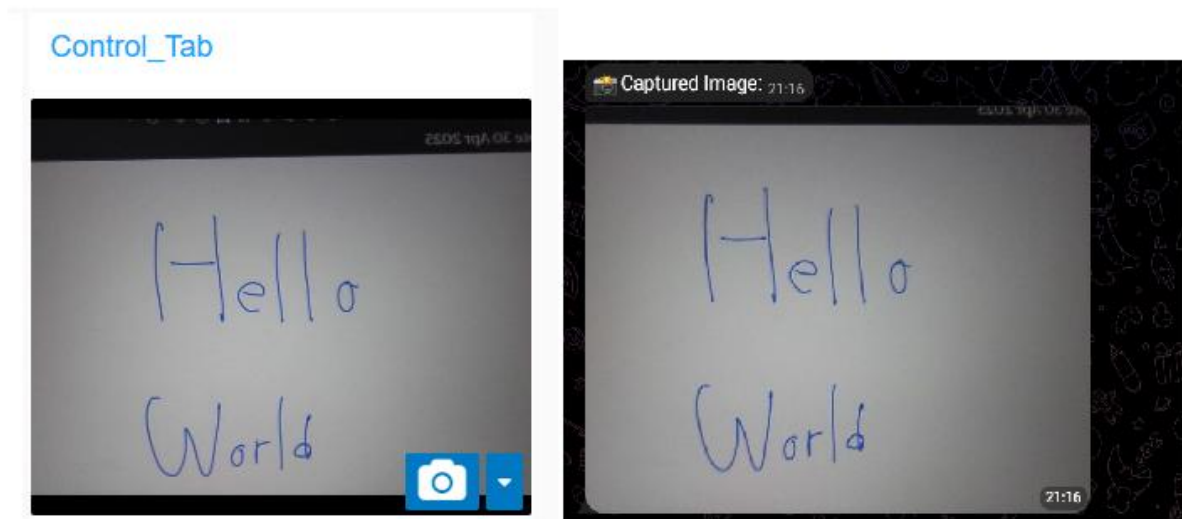


Figure 5.3.1.4 Camera and notification that captured image send via Telegram Bot

5.3.2 System Operation of Living and Kitchen

Moving into Living and Kitchen Tab, **Figure 5.3.2.1** shows the floorplan of Living Hall and Kitchen with SVG of the smart furniture. Users can control smart furniture accordingly with the separate of the control tab. **Figure 5.3.2.2** shows the status of the environment and each important smart furniture such as the temperature of air conditioner, refrigerator and oven. below the floorplan. **Figure 5.3.2.3** allow user to jump to Master Room Tab, Bedroom Tab or Main Door Tab by typing the “main, master or bed” to jump to relevant tab.

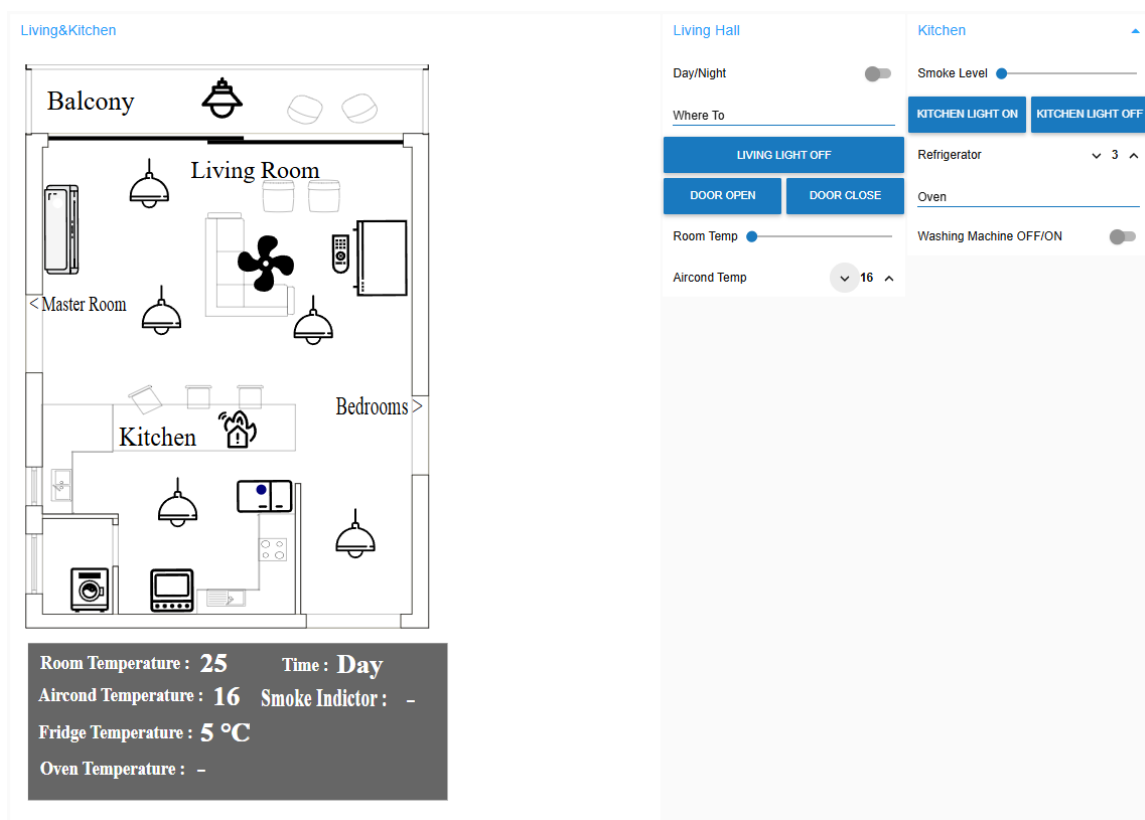


Figure 5.3.2.1 Dashboard that shows floorplan and control tab of Living and Kitchen

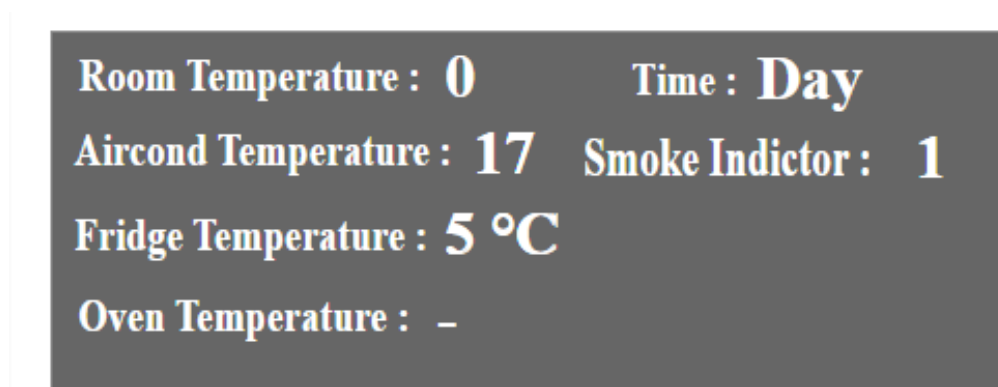


Figure 5.3.2.2 Status of environment or Temperature of some Smart Furniture

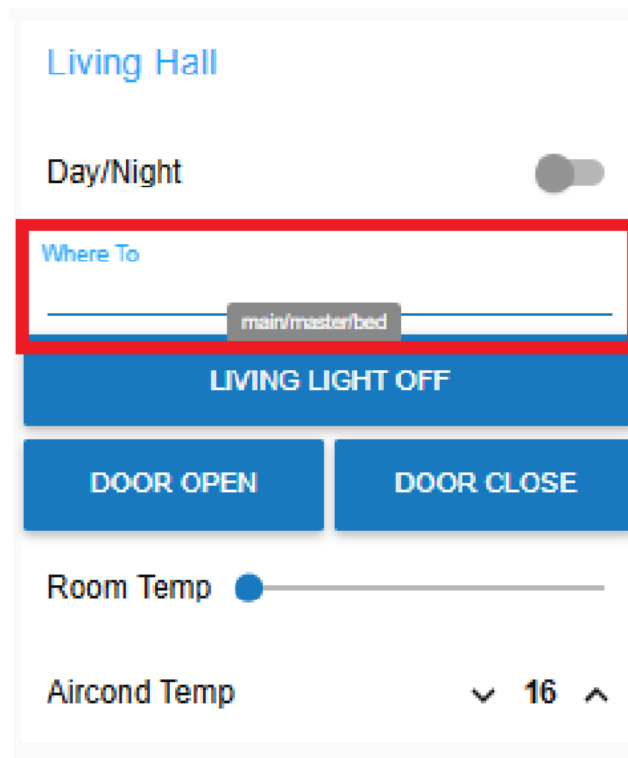


Figure 5.3.2.3 Type “main/master/bed” to jump to specific tab

There are some of the controls in the IoT MQTT Panel Application, which allow users to control and simulate the smart condo system through mobile phone (**Figure 5.3.2.4**). Telegram Bot will send warning messages or status for specific smart furniture or able to perform callback queries to control washing machine.

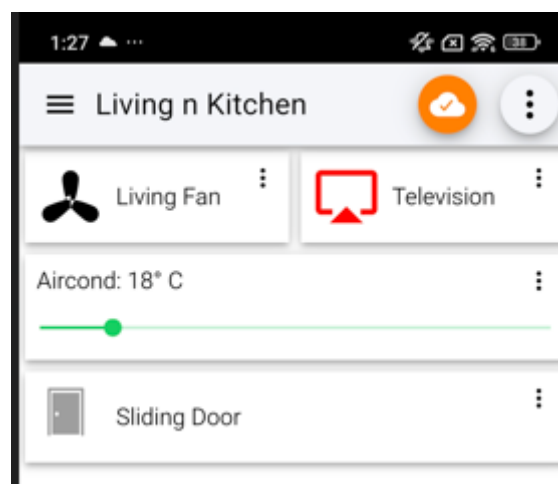


Figure 5.3.2.4 Control button in IoT MQTT Panel Application

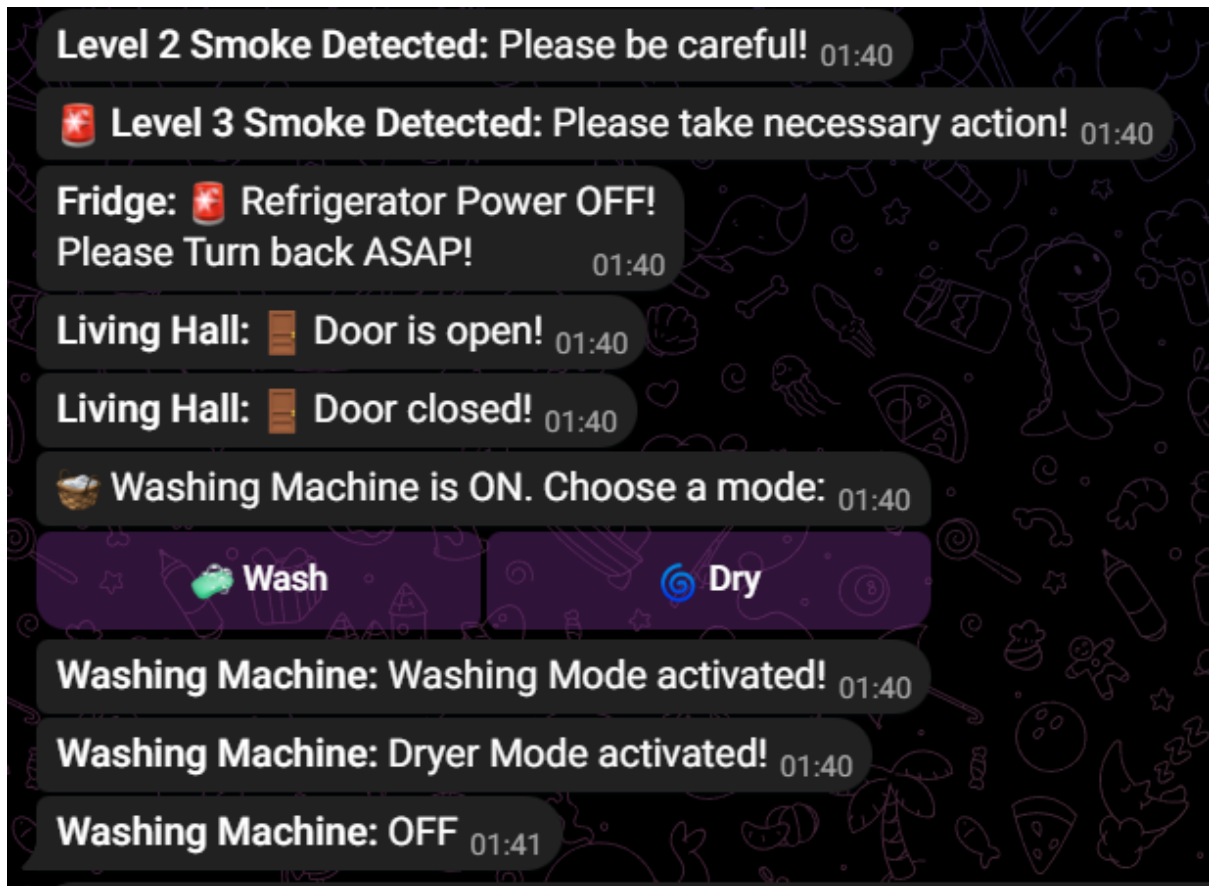


Figure 5.3.2.5 Telegram warning message or callback query

5.3.3 System Operation of Master Room

The “Master_Room” dashboard in **Figure 5.3.3.1** shows the floorplan of master bedroom of the smart condo, with the smart furniture, users can control the smart furniture to simulate in the control tab. Similar like Living & Kitchen tab, it also has its own status of environment showing room and air-conditioner temperature and room humidity (**Figure 5.3.3.2**). There are also control buttons available in the IoT MQTT Panel Application to control specific smart furniture in the master room (**Figure 5.3.3.3**).

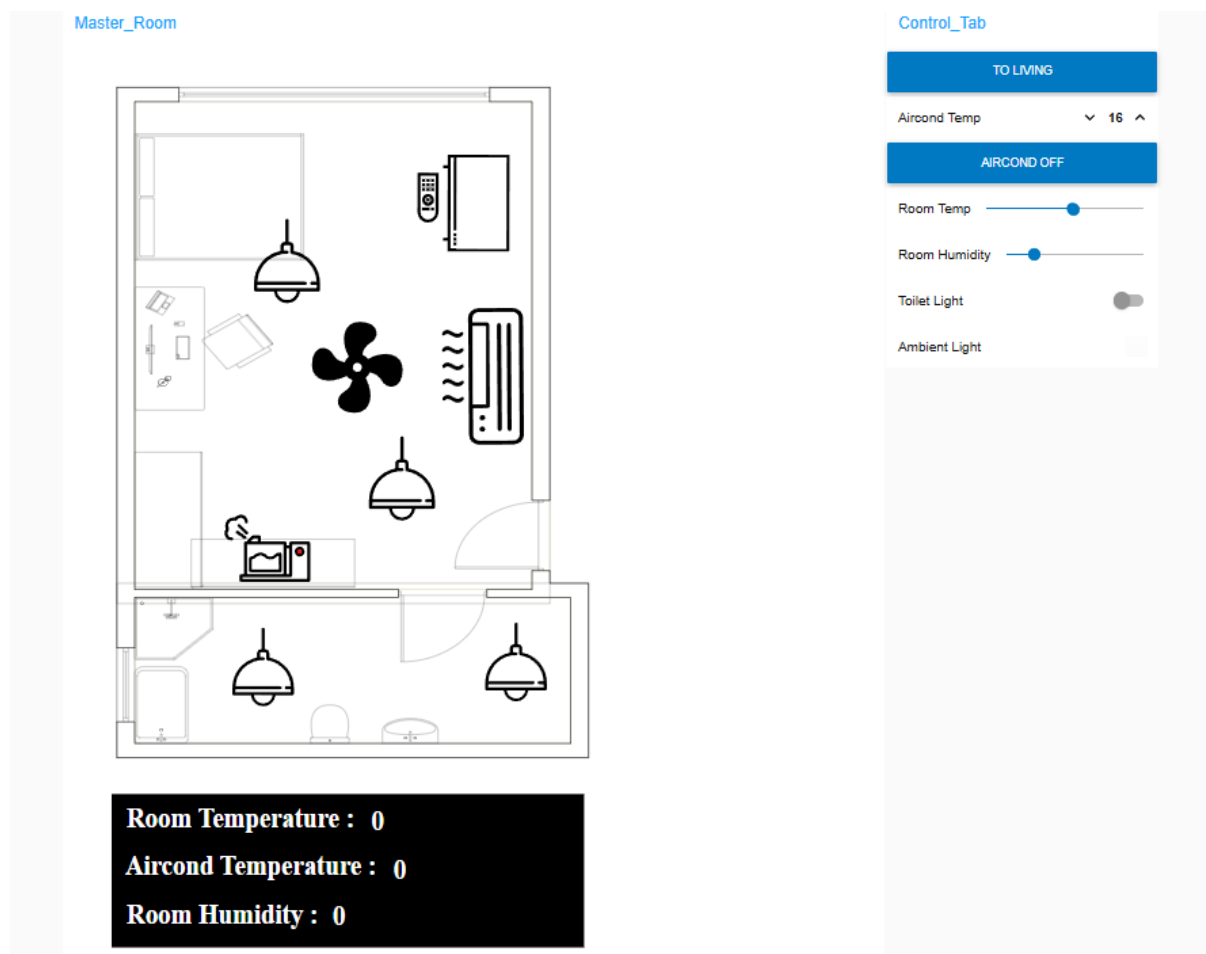


Figure 5.3.3.1 Dashboard that shows the floorplan and control tab of Master Room

Room Temperature : 0
Aircond Temperature : 0
Room Humidity : 0

Figure 5.3.3.2 Status of environment and Temperature of Smart Air-Conditioner

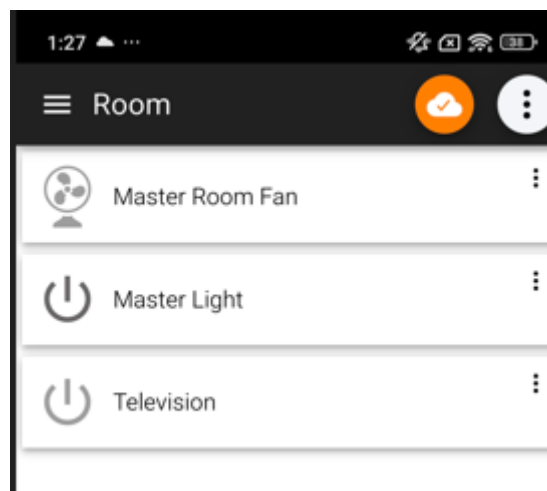


Figure 5.3.3.3 Control Button in IoT MQTT Panel for Master Room

5.3.4 System Operation of Bedrooms

Figure 5.3.4.1 shows the last tab of the dashboard is the Bedroom tab. Like the previous tab, it shows the floor plan of 2 bedrooms and a washroom. Each of the rooms has its own control panel in the control tab to prevent confusion. It also has a status slot to show each room temperature and show the availability of the shared washroom (**Figure 5.3.4.2**).

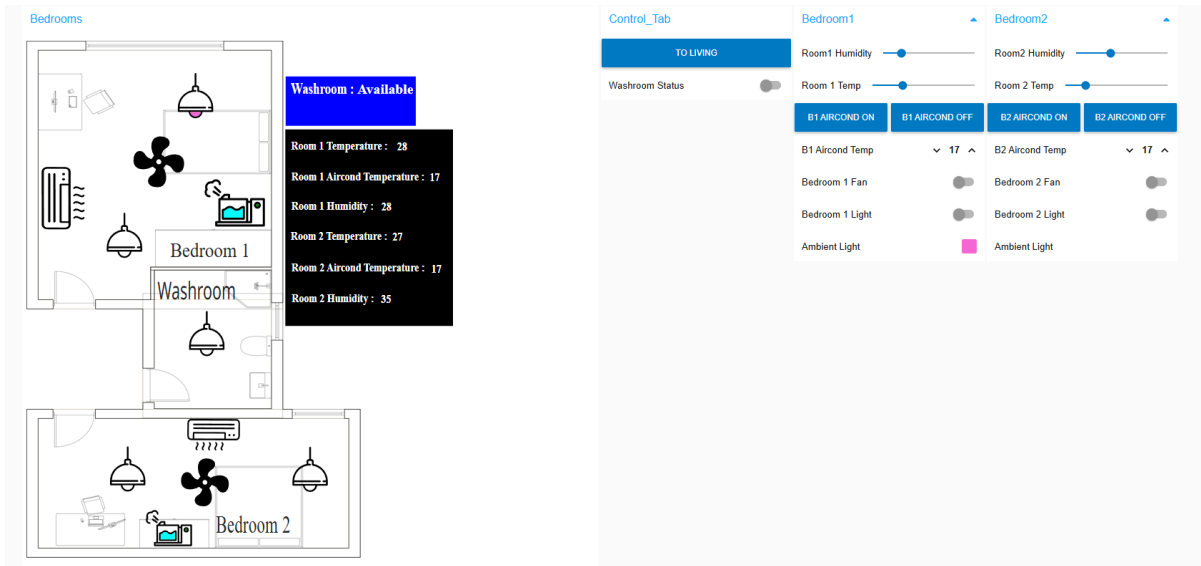


Figure 5.3.4.1 Dashboard that shows the floorplan and control tab of Bedrooms

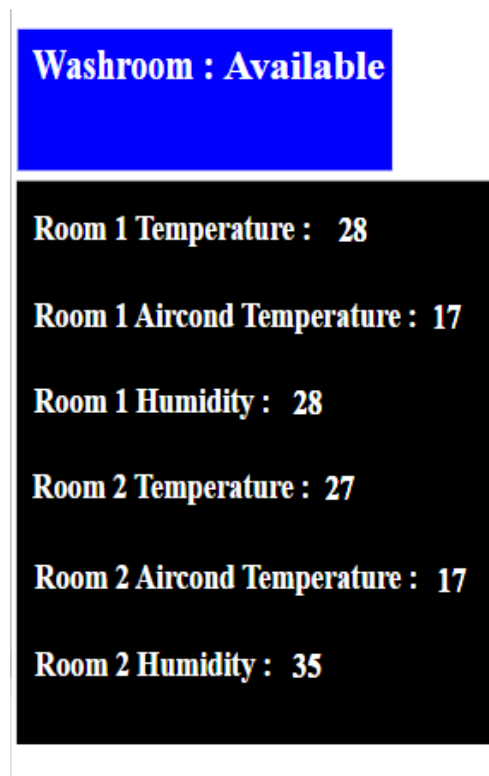


Figure 5.3.4.2 Status slot to show each Room status and availability of washroom

5.4 Concluding Remark

This chapter covered the steps that involved setting up and configuring the software components of the Smart Condominium Simulation system. It began with the installation and setup of the core platform for the system interaction and controls which are Node-RED, InfluxDB, IoT MQTT Panel application and the Telegram Bot. Various nodes in Node-RED such as “influxdb out”, “mqtt in”, “Telegram Sender and Callback Query” were configured to handle the data flow, command processing and real-time notification.

The connection of Telegram Bot allows the system to send out notifications and react to user requests via messaging. While IoT MQTT Panel application that installed in mobile phone allow users to control and simulate smart home functionalities remotely. In order to make the simulation to be easy and clear to simulate, the dashboard was divided into several areas and each of them has its own SVG graphic, status indicators and control interface.

Lastly, this chapter also concludes with the demonstration of the system operational to show each components work together to simulate an interactive and automated smart living environment.

CHAPTER 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

The performance and functionality of the Smart Condominium Simulation system having Four primary testing metrics which are virtual sensor accuracy, dashboard responsiveness, mobile app integration and notification reliability. To make sure that the system not only works as planned but also provides an ideal user experience, these metrics were chosen based on the system's key components and operational requirements.

Testing Metrics	Description	Expected outcome
Sensor Accuracy	To test the accuracy of the virtual sensor based on the slider inputs	The displayed value matches the slider inputs correctly.
Dashboard Responsiveness	Test how SVG graphic react according to the user controls.	All SVG elements able to react and give responds according to the user controls.
Mobile App Integration	To test the effectiveness of IoT MQTT Panel controls on mobile device	SVG graphics can work with the control in IoT MQTT Panel accordingly
Telegram Bot function Reliability	Test the reliability and responsiveness of Telegram Bot	Telegram Bot able to function normally by giving message instantly and react to the callback queries given by users

Table 6.1.1 Description and Expected Outcome of Key Testing Metrics

These performance metrics are important because they serve as the foundation for the whole smart condominium simulation system. Their successful performance shows that the simulation not only functions properly but also provide user-friendly experience which is important for smart home technology.

6.2 Testing Setup and Result

6.2.1 Main Door

System Testing	Expected Result	Result
Deploy the dashboard	Door indicates locked with red colour	Door indicates locked with red colour
Enter Password	Door will indicate green colour and unlock if correct password, but show error message with wrong password	Door indicates green colour and unlock when entered correct password and an error message shows up when enter the wrong password
Trigger Camera	Webcam of laptop successfully triggered	Laptop's webcam successfully triggered
Lock Door	Door will indicate red colour as locked and show notification that door has been locked	Door indicates red colour as locked and show notification that door has been locked
Camera Capture	Image from webcam will be captured and send notification and the captured image through Telegram Bot	Image from webcam successfully captured and the captured image send in Telegram Bot
After door unlocked	Automatic jump to Living & Kitchen dashboard after few seconds.	Successfully jump to Living and Kitchen dashboard automatically with few seconds of delay after the door unlocked

Table 6.2.1 System Testing and Result for Main Door



Figure 6.2.1.1 Captured Image from Webcam send by Telegram Bot

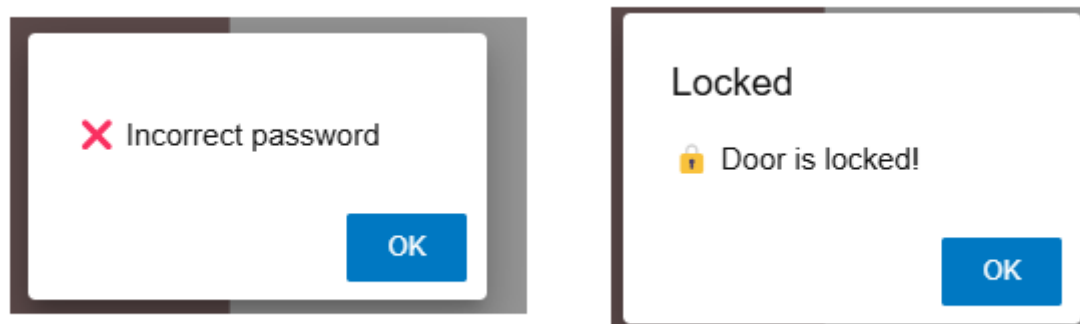


Figure 6.2.1.2 Dashboard notification when enter incorrect password and when door is locked

6.2.2 Living Hall

System Testing	Expected Result	Result
Day/Night switch in Control Tab	Lights (Living Room, Balcony, Foyer) should be turned on and status changes to "Night" and turned off when status in "Day"	Lights turned on and status updated to "Night" and turned off and status updated to "Day" as expected
"Where To" Tab	Typing "main", "master", or "bed" switches to the respective dashboard tab	Switched to the correct tab based on user input
Living Light Off button	Lights in living room will be switched off when triggered	Lights in living room switched off successfully
Door Open and Close button	Triggers sliding door animation (open/close) and telegram notification	Animation played correctly on trigger and notification sends accurately
Room Temp Slider	Displays selected temperature; turn on aircond if above 28°C.	Temperature displayed accurately; aircond activated when higher than 28°C.
Aircond Temp (In Control Tab)	Status reflects value set in Aircond Temp in Control Tab	Synchronized correctly the controls in Control Tab
Aircond Temp (MQTT Panel App)	Status reflects slider value from Aircond Temp slider in IoT MQTT Panel Application	Successfully synchronized with MQTT Panel Aircond Temp slider.
Living Fan (MQTT Panel App)	Fan animation in dashboard should be triggered	Fan animation triggered as expected
Television (MQTT Panel App)	Television will change colour to indicates turned on and back to normal when turn off	Colour in television changes when turned on and changed back when turned off
Sliding Door (MQTT Panel App)	Sliding door animation should be triggered (open/close) and telegram notification sends accordingly	Animation triggered as expected and telegram notification send accordingly

Table 6.2.2 System Testing and Result for Living Hall

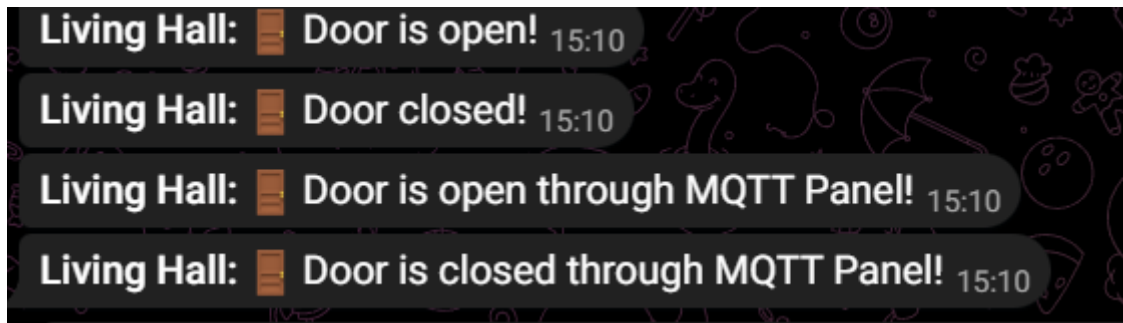


Figure 6.2.2.1 Notification about Sliding Door sends by Telegram Bot

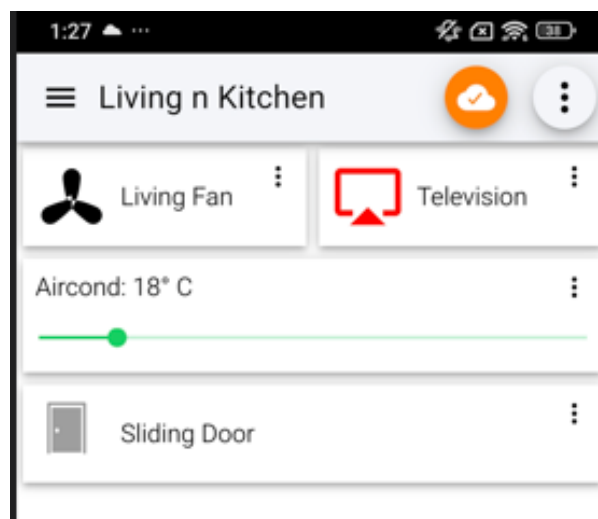


Figure 6.2.2.2 Control buttons in IoT MQTT Panel Application

6.2.3 Kitchen

System Testing	Expected Result	Result
Smoke Level Slider	<ul style="list-style-type: none"> • Status and indicator should be updated with slider • Color: orange at level 3, red at level 4 • Telegram alerts should be sent at level 3 and 4 	<ul style="list-style-type: none"> • Display and color changed correctly • Telegram warning sent as expected.
Kitchen Light ON/OFF button	Lights in kitchen should responds accordingly	Lights is kitchen responded accordingly
Refrigerator	<ul style="list-style-type: none"> • Temperature display updates • Telegram alert sent if it drops to 0. • Colour of SVG graphic of refrigerator should change 	<ul style="list-style-type: none"> • Display updated • Telegram alert sent at 0°C. • Colour changed accordingly
Oven Temperature	<ul style="list-style-type: none"> • Status shows “Low”, “Medium”, “High”, or “Over” • SVG changes color • Telegram alert at “Over”. 	All outputs functioned correctly, including SVG colour and Telegram alert.
Washing Machine OFF/ON	<ul style="list-style-type: none"> • Telegram message on activation • A selection to let users to choose “Wash” or “Dryer”. • SVG turn blue when select wash, turn red when select Dryer 	<ul style="list-style-type: none"> • Telegram Bot send message accordingly • Mode selection works • SVG colour changes as expected

Table 6.2.3 System Testing and Result for Kitchen

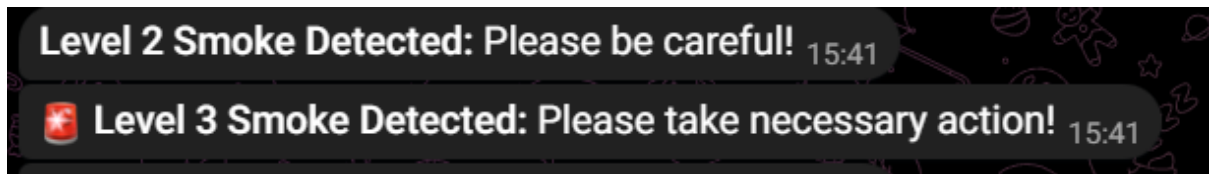


Figure 6.2.3.1 Telegram Notification for Smoke Sensor

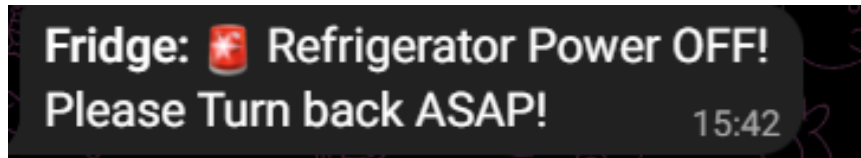


Figure 6.2.3.2 Telegram Notification for Refrigerator Powered OFF

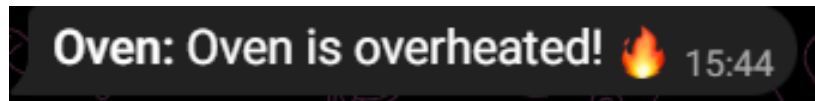


Figure 6.2.3.3 Telegram Notification for Oven Overheated

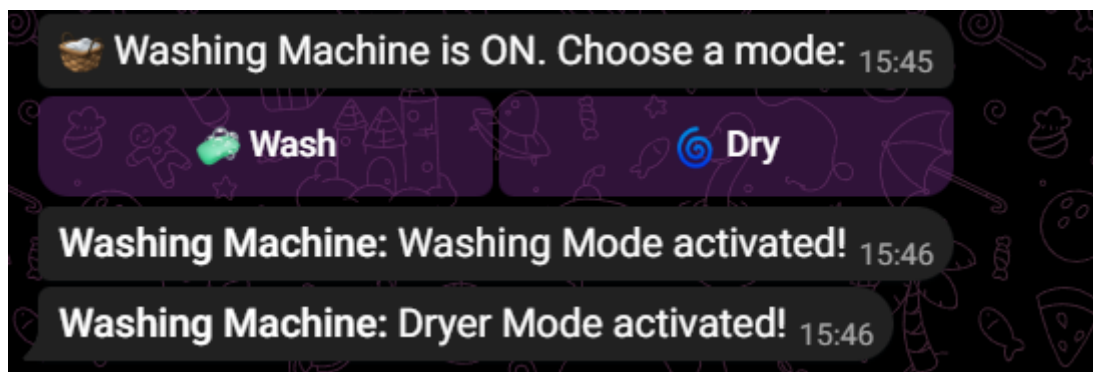


Figure 6.2.3.4 Telegram Notification and Mode Selection for Washing Machine

6.2.4 Master Room

System Testing	Expected Result	Result
To Living button	Expected to switch to Living & Kitchen dashboard	Switched to Living & Kitchen dashboard as expected
Room Humidity Slider	<ul style="list-style-type: none"> • Status display will change it value according to slider • Humidifier will be triggered and change colour when humidity lower than 40 	<ul style="list-style-type: none"> • Status display changed the value accordingly • Humidifier triggered by changing colour when humidity lower than 40 as expected
Room Temp Slider	<ul style="list-style-type: none"> • Status display will change the value according to the slider • Air conditioner will be turned on when temperature higher than 28°C 	<ul style="list-style-type: none"> • Status display changed the value accordingly • Air-conditioner turned on by changing colour when temperature higher than 28°C.
Aircond Temp	Expected status display changed the value accordingly	Status display changed the value accordingly
Aircond OFF button	Expected colour reverts to show turn off	Colour reverted, indicating aircond turned off
Toilet Light switch	Expected lamps in washroom light up and back to normal when turned off	Lamps in washroom successfully light up and back to normal when turned off
Ambient Light Picker	Expected Room lamps colour changed according to the picker	Colour of lamps in room changed accordingly
Master Room Fan (MQTT Panel App)	Expected animation of fan triggered and spins	Animation of fan triggered, and spin as expected
Master Room Light (MQTT Panel App)	Expected Lamp turn on/off with colour changes	Lamps lights up normally and back to normal when turned off
Television (MQTT Panel App)	Expected colour of television will change when ON and back to normal when OFF	Television changed colour when ON and back to normal when OFF

Table 6.2.4 System Testing and Results for Master Room

Master_Room

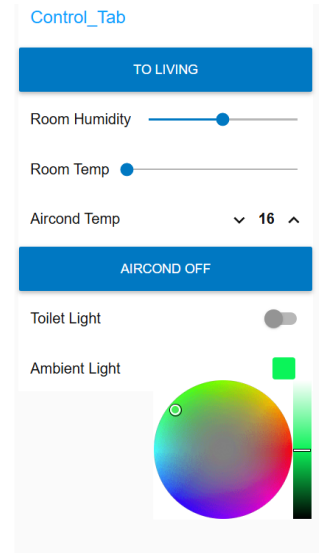
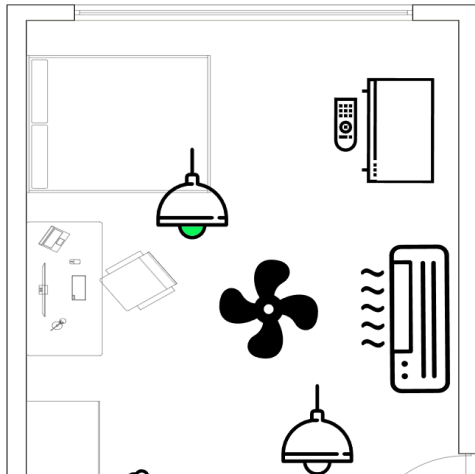


Figure 6.2.4.1 Colour of Light changes according to colour picker

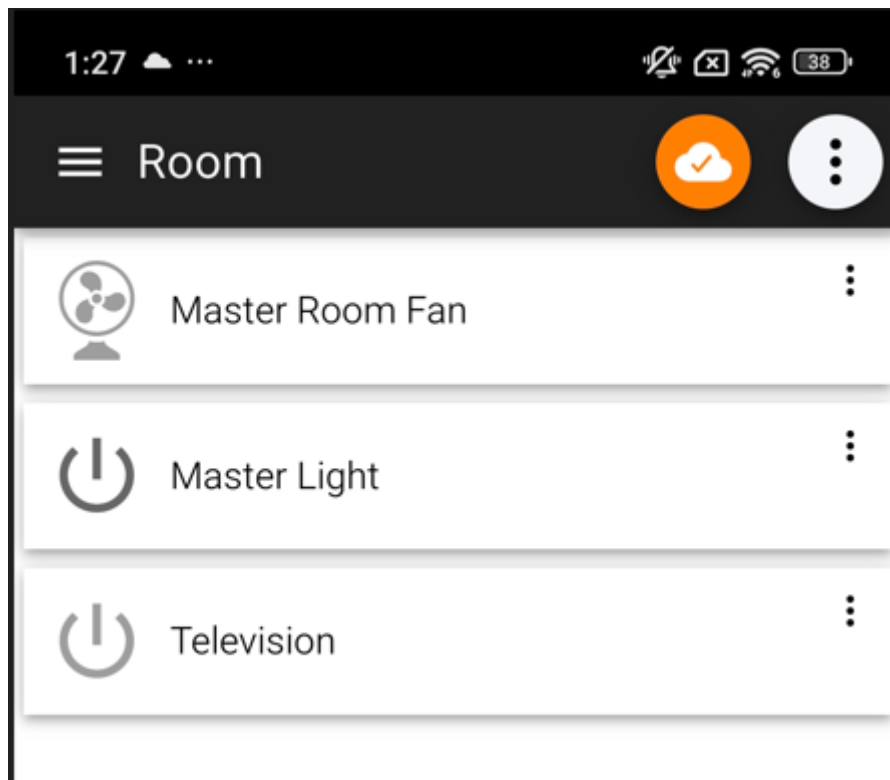
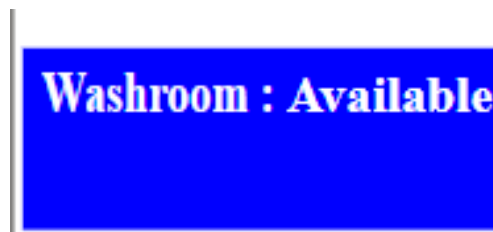


Figure 6.2.4.2 Control Buttons in IoT MQTT Panel Application for Master Room

6.2.5 Bedrooms Control Tab

System Testing	Expected Result	Result
To Living button	Expected to switch to Living & Kitchen dashboard	Switched to Living & Kitchen dashboard as expected
Washroom Status Switch	<ul style="list-style-type: none"> • Washroom status changes to “In Use” when active, and back to “Available” when turned off. • Lamp in washroom lights up 	<ul style="list-style-type: none"> • Washroom status changed accordingly • Lamps lights up as expected

Table 6.2.5 System Testing and Results for Bedrooms Control Tab*Figure 6.2.5.1 Washroom Status Display*

6.2.6 Bedroom 1 and Bedroom 2

Room Humidity Slider	<ul style="list-style-type: none"> • Status display will change its value according to slider • Humidifier will be triggered and change colour when humidity lower than 40 	<ul style="list-style-type: none"> • Status display changed the value accordingly • Humidifier triggered by changing colour when humidity lower than 40 as expected
Room Temp Slider	<ul style="list-style-type: none"> • Status display will change the value according to the slider • Air conditioner will be turned on when temperature higher than 28°C 	<ul style="list-style-type: none"> • Status display changed the value accordingly • Air-conditioner turned on by changing colour when temperature higher than 28°C.
Aircond ON/OFF button	Expected colour changed then turned on, reverts to show turn off	<ul style="list-style-type: none"> • Colour changed indicating aircond turned on, • Colour reverted, indicating aircond turned off
Aircond Temp	Expected status display changed the value accordingly	Status display changed the value accordingly
Room Fan Switch	Expected animation of fan triggered and spins	Animation of fan triggered, and spin as expected
Room Light Switch	Expected Lamp turn on/off with colour changes	Lamps light up normally and back to normal when turned off
Ambient Light Picker	Expected Room lamps colour changed according to the picker	Colour of lamps in room changed accordingly

Table 6.2.6 System Testing and Results for Bedroom 1 and Bedroom 2

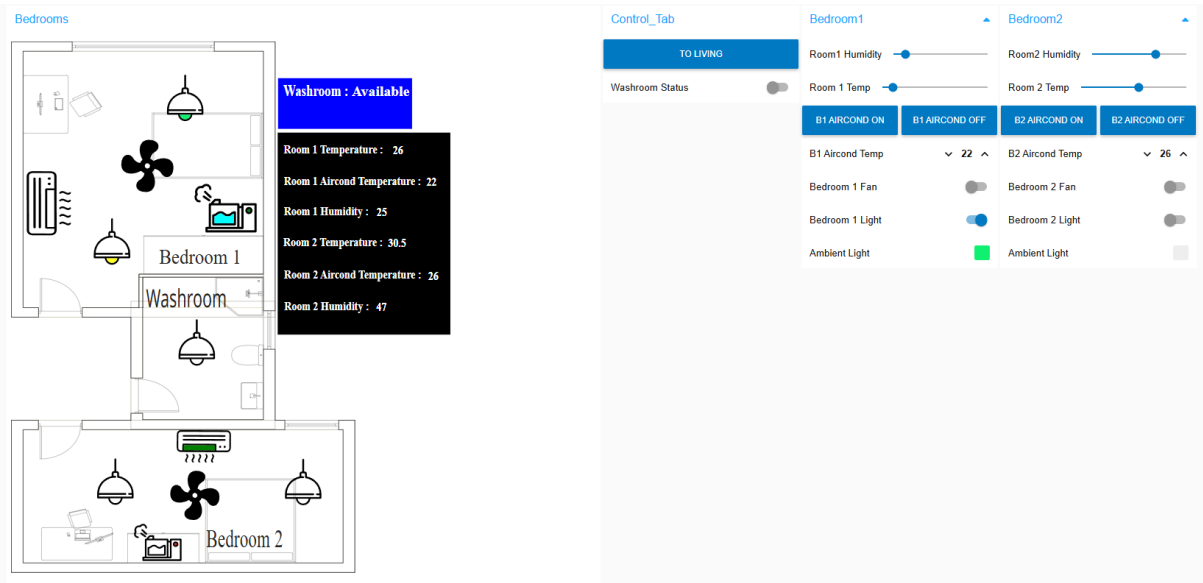


Figure 6.2.6.1 Dashboard of Bedrooms

6.3 Project Challenges

During the development process of the Smart Condominium Simulation system, there are several challenges that were encountered. The first challenge was the software setup. The accidental corruption of the Node-RED server, which causes the Node-RED unable to use. This situation forced to reformat laptop, requiring the reinstallation of all software tools and start over the system development. The credentials to access the InfluxDB database were forgotten and without a recovery method. The only method to solve is uninstall and reinstall InfluxDB entirely. This causes the loss of existing databases and API that have been set previously, need to regenerate over the API and apply in Node-RED.

The second challenge that this project encountered during the development of the system is the unsuccessful attempts to integrate the webcam node which limits the automation of the simulation. On the other hand, animating SVG components in Node-RED also caused challenges for the project development when aligning the values on the x-axis and y-axis. This challenge causes the fan in the dashboard will not rotate statically but run everywhere in the dashboard, The animation for the sliding door also will run to other places if the x-axis and y-axis are configured incorrectly.

Creating the floorplan with the smart furniture inside for the entire condominium for simulation was another time-consuming task. This process required not only designing the layout using Inkscape but also ensuring that each component matched accurately with its functional label to representation in Node-RED. While searching for SVG images for the project, an accidental download of adware corrupted the web browser, forcing a full cache reset and resulting in the loss of previously researched data.

Lastly, the major challenge was the stability of HiveMQ server as the MQTT broker. The unexpected disconnection of the server causes the disconnections of the MQTT broker, this affected the connection between Node-RED and the broker disconnected, and IoT MQTT Panel are not able to function.

6.4 Objective Evaluation

The Smart Condominium Simulation project has successfully met its main objectives. The system has been achieved to be scalable, allowing the integration of various smart devices across different rooms, and allowing to add more smart furniture in the future. The use of Scalable Vector Graphics (SVG) enhanced the visual clarity and interactivity of the dashboard, with responsive animations like fan rotation and door sliding.

The project also achieved platform independence with working smoothly across desktop and mobile devices, including integration with the IoT MQTT Panel and Telegram Bot. Using Node-RED as a low-code platform allows fast creation and simplified automation flows, making the system easy to use and understand even users have limited coding experience.

Overall, the project fulfilled its goals by delivering a functional, adaptable, and user-friendly simulation platform.

6.5 Concluding Remark

This chapter shows the Smart Condominium Simulation system has been proven to be a successful implementation using low-code programming through Node-RED, integration of visual SVG elements and real-time control mechanism via mobile and chatbot platform. The system testing shows the functionality of the system, including device automation, real-time monitoring, and remote access, changing the dashboard to simulate across different rooms and control methods.

This project faces several technical challenges such as software reinstallation due to the corruption of Node-RED, SVG animation issues, and the instability of the MQTT broker. Luckily the project was able to overcome these challenges and deliver to a more responsive and interactive simulation environment.

Lastly, the project has achieved all the main objectives, which are scalability, platform independence, visual interactivity and demonstrating the possibility of developing a smart condominium simulation system that is both scalable and easy to use. This system provides a strong foundation for future improvements, including the integration with real sensor data or automation controlled by AI, and serves as a practical model for simulating smart living environments using low-code programming techniques.

CHAPTER 7

Conclusion and Recommendation

7.1 Conclusion

The development of Smart Condominium Simulation system was successful with the use low-code programming techniques through Node-RED, integrating SVG graphics and real-time control and interaction with IoT interfaces such as MQTT and Telegram Bot. The main objective of this project was to provide a scalable, interactive, and accessible simulation platform for smart home environments. These objectives were fully achieved, as demonstrated by the system's ability to integrate with multiple virtual smart devices and simulate real-life automation scenarios across various condominium spaces.

Throughout the project, significant works were done to ensure the functionality, responsiveness, and user-friendliness of the simulation system. The system also allowing users to interact with virtual floor plan, control smart devices in real time, monitor environmental conditions, and receive alerts through mobile interfaces. Testing of the system was confirmed that key components such as virtual sensor accuracy, dashboard responsiveness, mobile app integration, and notification reliability are able to function normally as expected.

During the project development, several challenges have encountered such as system crashes, sensor integration issues, and animation alignment difficulties. But this project was able to overcome these obstacles by using iterative development and problem-solving techniques. The use of the Agile methodology proved effectively by allowing flexibility and continuous improvement throughout the development process.

In conclusion, this project has successfully demonstrated the potential of low-code development in building smart condominium simulation platforms. It also provides a valuable platform for users, developers, and small enterprises to design, test, and experience smart living environments without requiring professional or deep technical knowledge. The system's modular design and platform compatibility also helps to make the project have a strong foundation for future enhancements, such as integration with physical sensors, AI automation, or cloud-based analytics.

7.2 Recommendation

Even though this Smart Condominium Simulation system had successfully met its objectives, several enhancements can also be considered for future development to improve realism, functionality, and user engagement.

The first recommendation is the integration of an AI module for automation. With applying the machine learning algorithms, this system is able to analyze user behavior and environmental data to make predictive decisions, such as automatically adjusting the brightness of lighting, air conditioner temperature, or other device usage patterns based on historical habits. In the future development, the enhance security of the smart condominium with the implementation of live image processing through Telegram for the door camera is also suggested. This would allow real-time monitoring and facial recognition directly from the chatbot, offering a more responsive and secure simulation of access control. Additionally, integrating a live floorplan preview via either the Telegram Bot or the MQTT Panel App would allow users to monitor device status and movement in real time from mobile interfaces, improving the remote visibility.

Another recommendation for future improvement is the addition of voice communication features, including voice chat and audio playback, which would allow two-way communication between the user and the simulation system which similar to intercoms in real life. For visual realism, replacing the 2D SVG floorplan with a 3D-rendered module would offer a more immersive and life-like representation of the smart condominium living environment, further decrease the gap between simulation and real-world interaction. System with the real life sensors such as using NFC or RFID for secure door access and light sensors to control indoor lighting based on ambient brightness levels. These enhancements would allow the system to react to real physical inputs, not just virtual simulations.

Lastly, integrating real-time weather data and a real-time temperature sensor would allow the simulation to respond to actual environmental conditions. For example, the air conditioning could activate automatically based on both indoor and outdoor temperatures, simulating realistic control behavior. With these improvements the system's capability would significantly improve, making the system to be more practical and realistic and also a more effective tool for simulating and testing smart IoT technologies in real-world scenarios.

REFERENCES

- [1] Bhumika, “What is windows operating system?” Goseeko blog, <https://www.goseeko.com/blog/what-is-windows-operating-system/> (accessed Aug. 25, 2024).
- [2] R. Sheldon and E. Mixon, “What is unix?,” Data Center, <https://www.techtarget.com/searchdatacenter/definition/Unix> (accessed Aug. 25, 2024).
- [3] “What is mongodb?,” IBM, <https://www.ibm.com/topics/mongodb> (accessed Aug. 25, 2024).
- [4] “Introduction to influxdb: A time-series database,” Community Platform, <https://wearecommunity.io/communities/india-java-user-group/articles/891> (accessed Aug. 25, 2024).
- [5] “What is java?,” IBM, <https://www.ibm.com/topics/java> (accessed Aug. 30, 2024).
- [6] “What is Java used for?,” Coursera, <https://www.coursera.org/articles/what-is-java-used-for> (accessed Aug. 30, 2024).
- [7] GeeksforGeeks, “Node-red,” GeeksforGeeks, <https://www.geeksforgeeks.org/node-red/> (accessed Aug. 25, 2024).
- [8] “Red,” Node, <https://nodered.org/> (accessed Aug. 25, 2024).
- [9] Smart Home Simulation System | IEEE conference publication | IEEE xplore, <https://ieeexplore.ieee.org/document/7753134> (accessed Apr. 19, 2024).
- [10] N. Alshammari, T. Alshammari, M. Sedky, J. Champion, and C. Bauer, “OpenSHS: Open smart home simulator,” MDPI, <https://www.mdpi.com/1424-8220/17/5/1003> (accessed Apr. 19, 2024).
- [11] Simulation of a smart home environment | IEEE conference publication | IEEE xplore, <https://ieeexplore.ieee.org/document/6698459/> (accessed Apr. 19, 2024).

REFERENCES

- [12] Bugejajoseph, “Open-source smart home simulators,” Joseph Bugeja, <https://fullcirclesecurity.org/2021/01/25/open-source-smart-home-simulators/> (accessed Apr. 18, 2024).
- [13] GeeksforGeeks, “Top 5 sdlc(software development life cycle) methodologies,” GeeksforGeeks, <https://www.geeksforgeeks.org/top-5-sdlcsoftware-development-life-cycle-methodologies/> (accessed Aug. 24, 2024).
- [14] MichaelPageUK, “The top 7 SDLC methodologies,” Michael Page, <https://www.michaelpage.com.au/advice/career-advice/productivity-and-performance/top-7-sdlc-methodologies> (accessed Aug. 24, 2024).
- [15] B. Lutkevich and S. Lewis, “What is the waterfall model? - definition and guide,” Software Quality, <https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model#:~:text=The%20waterfall%20model%20is%20a,the%20edge%20of%20a%20cliff.> (accessed Aug. 24, 2024).
- [16] “What is iterative model?,” Iterative Model: Advantages and Disadvantages |Professionalqa.com, <https://www.professionalqa.com/iterative-model> (accessed Aug. 24, 2024).
- [17] GeeksforGeeks, “What is spiral model in software engineering?,” GeeksforGeeks, <https://www.geeksforgeeks.org/software-engineering-spiral-model/> (accessed Aug. 24, 2024).

POSTER

Smart Condominium Simulation Using Low Code Programming



INTRODUCTION

Democratize smart condominium innovation through low code programming, enabling a broader audience to simulate and optimize smart condominium models without requiring extensive technical expertise.

SOFTWARE/ TOOLS

Node-RED,
Inkscape, InfluxDB,
IoT MQTT Panel,
Telegram

PROJECT OBJECTIVES

1. Develop a scalable smart condominium simulation model
2. Apply SVG to enhance the visual representation of smart condominium elements.
3. Simulation employs Node-RED, to create and manage interactions between IoT smart devices.

CONCLUSION

The project creates a flexible, scalable smart condominium simulation to improving user comfort and energy efficiency, enhances real-time control, and contributes valuable insights to the smart condominium field, laying the groundwork for future improvements.

Developed by : Tee Yu Xuan

Supervised By: Ts Dr Goh Hock Guan