INTELLIGENT DRYING RACK CONTROL SYSTEM WITH WEATHER CONDITION PREDICTION USING FUZZY LOGIC

NUR DARWISYAH FAQIHAH BINTI LUTFI

Bachelor Of Computer Science
(Software Engineering) With Honours

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	: Nur Darwisyah Faqihah Binti Lutfi
Date of Birth	
Title	: Intelligent Drying Rack Control System with weather condition prediction using Fuzzy Logic
Academic Session	: Semester II Academic Session 2022/2023

I declare that this thesis is classified as:

- ☐ CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*
- ☐ RESTRICTED (Contains restricted information as specified by the organization where research was done)*
- ☐ OPEN ACCESS I agree that my thesis to be published as online open access (Ful Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

[illegible]	
(Student's Signature)	(Supervisor's Signature)
New IC/Passport Number	Name of Supervisor
Date: 2nd July 2023	

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons	(i)
	(ii)
	(iii)

Thank you.

Yours faithfully,

____
(Supervisor's Signature)

Date: 25 July 2023

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

SUPERVISOR's DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Science (Software Engineering) with Honours.

____

(Supervisor's Signature)

Full Name	: TS. DR. MOHD IZHAM BIN MOHD JAYA
Position	: HEAD OF PROGRAM (SOFTWARE ENGINEERING)
Date	: 25 July 2023

____

(Co-supervisor's Signature)

Full Name	:
Position	:
Date	:

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

[illegible]

(Student's Signature)

Full Name	: NUR DARWISYAH FAQIHAH BINTI LUTFI
ID Number	: CB20080
Date	: 2nd July 2023

INTELLIGENT DRYING RACK CONTROL SYSTEM WITH WEATHER CONDITION PREDICTION USING FUZZY LOGIC

NUR DARWISYAH FAQIHAH BINTI LUTFI

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor Of Computer Science (Software Engineering) With Honours

Faculty of Computing

UNIVERSITI MALAYSIA PAHANG

JULY 2023

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to the following individuals and organizations who have contributed to the completion of this thesis titled "Intelligent Drying Rack Control System with Weather Condition Prediction using Fuzzy Logic".

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr Ts. Mohd Izham bin Mohd Jaya, for their invaluable guidance, support, and expertise throughout the entire research process. Their patience, encouragement, and constructive feedback have been instrumental in shaping this thesis.

I would like to acknowledge the assistance and cooperation received from the participants who took part in the data collection phase of this study. Their willingness to provide information and insights has significantly contributed to the validity and reliability of the research findings.

Furthermore, I am grateful to my family and friends for their unwavering support, understanding, and encouragement throughout this academic journey. Their love and encouragement have been a constant source of motivation and inspiration.

Lastly, I would like to express my gratitude to all the researchers, scholars, and authors whose works and studies have provided valuable references and insights for this thesis. Their contributions have enriched the theoretical framework and helped shape the methodology and analysis of this research.

ABSTRAK

Kebanyakan orang mengikut rutin harian termasuk mencuci dan mengeringkan pakaian. Namun, kebanyakan orang mengatakan bahawa proses mengeringkan pakaian adalah bahagian yang paling mencabar disebabkan cuaca yang tidak menentu di Malaysia. Projek ini memperkenalkan Sistem Kawalan Rak Pengering Pintar dengan ramalan keadaan cuaca menggunakan logik Kabur yang boleh mengesan secara automatik kehadiran hujan, cahaya matahari, suhu dan kelembapan. Sistem rak pengering ini akan ditarik masuk dan dikeluarkan apabila sistem mengesan perubahan cuaca sekeliling. Projek ini dibangunkan untuk membantu orang menjalankan tugas harian di luar dan mencegah pakaian mereka daripada menjadi lembap akibat hujan. Tujuan projek ini adalah untuk membantu orang menguruskan pakaian mereka ketika mereka jauh dari rumah, mengurangkan tekanan mereka dan membebaskan perhatian mereka untuk tugas harian yang lain. Prototaip model ini menggunakan peranti keras seperti Arduino Uno, sensor hujan, sensor cahaya, sensor DHT11, dan motor servo sebagai penggerak. Dengan menggunakan logik Kabur, apabila cuaca mendung, sensor akan menurunkan rak pengering ke bumbung, dan apabila cuaca cerah, rak akan dipindahkan semula. Inisiatif ini membantu orang menguruskan pakaian mereka ketika mereka jauh dari rumah, mengurangkan tekanan mereka dan membebaskan perhatian mereka untuk tugas harian yang lain.

ABSTRACT

Most people's daily routines include washing and drying clothes. However, most of the people claimed that the process of drying up the clothes is the most challenging part due to the unpredictable weather in Malaysia. This project introduces an Intelligent Drying Rack Control System with weather condition prediction using Fuzzy logic that can automatically detect the presence of rain, sunlight and temperature, humidity. This drying rack system will be retrieved and pulled out when the system detects a change in surrounding weather. This project was developed to help people stay busy and focus on daily tasks outside and prevent their clothes from getting damp from the rain. The prototype model used hardware such as Arduino Uno, rain sensor, light sensor, DHT11 sensor and servo motor as actuator. Using Fuzzy logic showed that when it is cloudy, the sensor will pull the drying rack down to the roof, and when it gets sunny, it will be moved again. This initiative helps people manage their clothes when they are away from home, which reduces their stress and frees up their attention for other everyday duties.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

MQTT	Message Queuing Telemetry Transport
IoT	Internet of Thing
AI	Artificial Intelligent
MySQL	My Structured Query Language
LED	Light-Emitting Diode
DHT	Digital Humidity and Temperature
API	Application Programming Interface

CHAPTER 1

INTRODUCTION

1.1 Introduction

An increase in interest in smart home technology that makes our lives simpler and more effective has been seen in recent years. The '*intelligent drying rack control system with weather condition prediction using fuzzy logic'* which is intended to automate and optimize the process of drying clothing indoors, is one example of such an innovation. More individuals are resorting to indoor drying options as urban living becomes more prevalent and outside space becomes increasingly scarce (S.Chaihang, 2021). Traditional drying racks, on the other hand, might be ineffective, occupying valuable space and using a lot of energy (Kharisma, 2019).

This project background part examines the literature on automated drying rack provides an IoT architecture. It will have four layers which are sensing layer, network layer, data processing layer and application layer. For the sensing layer it uses four sensors which are DHT11 humidity and temperature sensor, photocell sensor and rain sensor. It will give the movement to actuators which are servo motors and give the output LED will bright. For the microcontroller devices, there are WIFI Uno ESP32 and Raspberry Pi 3B+. Then, for the network layer, the gateway which connects with raspberry pi via Wi-Fi connection to the server by using MQTT server. In the data processing layer, data will be stored in MySQL. WebSocket server, AI server, and fuzzy logic has been implemented in this system. '*Intelligent Drying Rack Control System with weather condition prediction using fuzzy logic*' addresses this issue by incorporating features such as servo motors move the suspension in and out from the side of the home, remote monitoring, and control via web dashboard. By using sensors and timers, intelligent drying racks can optimize the dying process and reduce energy consumption compared to traditional drying methods. The three variables used in this study's fuzzy system are temperature, rain, and light intensity. The input value will be transformed into a linguistic value for fuzzy input (Tatyantoro, 2022). In the application layer, there is a web application which is Laravel. It will display data visualization such as real time graphs and real time sensors data. In addition, the dashboard allows telegrams to inform the user about the system and provides control buttons to operate the actuators.

This project explores the concept of an intelligent drying rack system using fuzzy classification in depth. It will review the current state of the technology, identify the benefits and limitations of using intelligent drying rack systems, and provide recommendations for practical and efficient system design and implementation. Additionally, the project will consider the environmental impact and economic viability of using intelligent drying racks, as well as the potential for future development and innovation in this area.

1.2 Problem Statement

1.2.1 Unpredictable weather conditions

Unpredictable weather circumstances might make it more difficult for items to dry, such as rain or too much sun. Rain can soak the clothing and make it difficult to adequately remove the moisture, resulting in lengthy drying times. Furthermore, clothes left outside in a rainfall may end up dirty or even have their colour fade. On the other side, too much sunshine can cause clothing to dry too rapidly, which could lead to fabric that is stiff or wrinkled. In addition, extended exposure to direct sunlight can weaken the fabric's fibres and fade colours, shortening the life of the garments. Why it happens? Photodegradation is the cause, according to the librarian of Congress (2010). How does that work? Because of the chromophores in the dyes, some chemical linkages can be found in our clothing. The UV rays from too much sunshine can damage the chromophores' chemical linkages, causing the clothing to fade as though a bleaching treatment had been used.

Therefore, the unpredictable nature of weather conditions poses challenges for drying clothes efficiently, necessitating adaptability in drying methods and consideration of protective measures like using drying racks indoors or monitoring weather forecasts to minimize potential damage or delays.

1.2.2 Inefficiency and inconvenience of traditional drying methods

Due to the inefficiency and inconvenience of traditional drying techniques, consumers typically forget about their drying clothes. The weather is a major factor when hanging clothing outside on a clothesline. Unpredictable rain or high gusts might hinder or even harm the drying process and the garments. Similar to outside drying racks, interior drying racks and clotheslines occupy space and might restrict movement inside the living space. Additionally, it takes time and energy to physically flip or rotate the garments to guarantee even drying. These elements add to the annoyance and ineffectiveness of conventional drying techniques, leading users to frequently forget about their clothing, leaving it moist for prolonged periods of time, and sometimes resulting in unpleasant odours or mildew. Thus, if no one was home to retain the clothing, there is a good probability that they were still wet when the owner returned. In that case, they would need to wash the clothes again or spin the water out of the clothing once more. As a result, there is a need for more practical and automated drying solutions that can resolve these problems, offer people a hassle-free experience, and guarantee that their clothing will dry quickly and effectively.

1.3 Objective

There are 3 objectives which aim to be achieved at the end of the project:

1. To study the requirement for IoT architecture that provides real-time data and provides insightful information about weather.
2. To develop an intelligent drying rack system that uses rain sensors, light sensors, temperature, and humidity sensors with fuzzy logic.
3. To evaluate the effectiveness of an intelligent drying rack system based on fuzzy.

1.4 Scope

The scope of the project consists of user, system, and development. The scopes are specified as below:

User Scope:

I. House owner or household usage.

System Scope:

I. Use sensors to collect the data which are the temperature and humidity of weather, rain detection and light presence.
II. Provide alerts to users when the rain sensors detect the water rain.
III. Provide precise classification skills, improving the drying rack using fuzzy logic.
IV. Communication protocols using MQTT and WebSocket

Development Scope:

I. The system will be developed using Visual Studio Code, Laravel framework and MySQL as database server.
II. This system uses MQTT server, web server and WebSocket server.

1.5 Thesis Organization

This thesis consists of five chapters, starting from introduction, literature review, methodology, implementation and results, and conclusion.

Chapter 1 mostly discusses the general information regarding the Intelligent Drying Rack Control System such as the introduction to the project, problem statement, main objectives, scope and the thesis organization.

The goal of chapter 2, which is the literature review section, is to examine the specifics of current market-available system solutions that are relevant to this project and to compare them in order to learn more about each of their advantages and disadvantages that should be taken into account when this project is being developed.

The methodology component of the project is covered in chapter 3. Here, the project's selected method of development is described in detail, and the system's flow is elaborated upon using flowcharts, IoT architecture and wireframe dashboard. Any specialised software and hardware that will be employed in the system's development would also fall under this category.

Once the system has been created and is prepared for deployment, the report moves on to chapter 4, where it is shown dashboard and fuzzy for results and discussion for the project.

Finally, chapter 5 will bring this thesis to a close. This section contains a summary of the development project and conclusion of the project.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Before creating the system, it is necessary to do comparisons between relevant current systems in order to comprehend each of its capabilities as well as other defining characteristics like the processes and technologies employed that can be incorporated into the project. The '*Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic*' can then be developed by evaluating the concrete strengths and weaknesses of each of the systems.

2.2 Existing Systems/Works

This section will analyze various past systems and techniques for intelligent drying rack control systems. The main focus of the review will be on comprehending the design, implementation, and performance of these systems. The analysis will take into account data collection techniques, monitoring tactics, data processing algorithms, and decision-making procedures. By examining the benefits and drawbacks of various systems, substantial insights will be gained to improve the suggested solution.

2.2.1 Intelligent Drying Rack System based on Internet of Things

The traditional outdoor drying racks for clothes face inconveniences due to weather changes, making it difficult to collect and dry clothes. To address this issue, a smart drying rack based on the Internet of Things (IoT) is designed (X.Xing, 2021). It is suitable for various settings such as homes, hotels, hospitals, and laundry shops where large amounts of clothing require frequent washing and outdoor drying. The smart drying rack offers features like automatic drying, recycling, drying, and disinfection of clothing. It addresses the limitations of traditional drying methods by incorporating remote one-button drying, automatic induction recycling, one-button undressing, and remote sterilization. The system utilizes an STM32F103C8T6 microcontroller as the main control chip and the OneNET cloud platform as the information port. Data from temperature and humidity sensors in the drying rack sensor module enable intelligent perception of the environment and clothing recycling. The system can be controlled manually through a touch screen interface or remotely monitored and controlled via a mobile phone app. This user-friendly operation enhances the convenience and portability of smart homes.

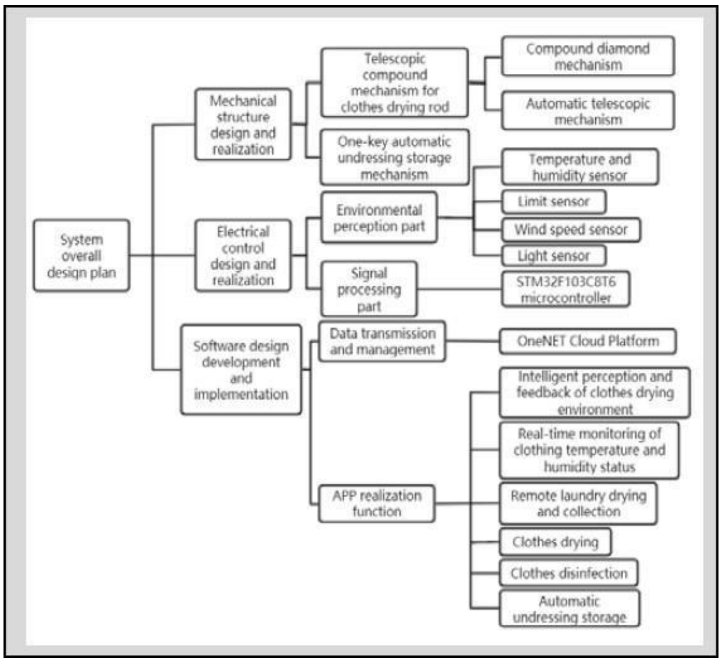

System overall design plan
Mechanical structure design and realization
Electrical control design and realization
Software design development and implementation
Telescopic compound mechanism for clothes drying rod
One-key automatic undressing storage mechanism
Environmental perception part
Signal processing part
Data transmission and management
APP realization function
Compound diamond mechanism
Automatic telescopic mechanism
Temperature and humidity sensor
Limit sensor
Wind speed sensor
Light sensor
STM32F103C8T6 microcontroller
OneNET Cloud Platform
Intelligent perception and feedback of clothes drying environment
Real-time monitoring of clothing temperature and humidity status
Remote laundry drying and collection
Clothes drying
Clothes disinfection
Automatic undressing storage

Figure 2.2.1 - System overall

2.2.2 Propose design of smart clothesline with the tree diagram approach analysis and quality function deployment method for Indonesia weather

In tropical countries like Indonesia, the unpredictable rainy season causes anxiety for people when hanging clothes outdoors. Electric clotheslines are an option, but they require more power and are costly to operate, impacting household budgets. To address this issue, a new design of an automatic clothes dryer is proposed, considering the seasons in Indonesia, particularly in the Pekanbaru area (Kharisma, 2019). Previous research has been conducted on clothes dryers using light sensors (LDR) and rain sensors to protect clothes from rain. The proposed design aims to be more ergonomic and energy-efficient by utilizing the Quality Function Deployment (QFD) method to align consumer needs with the product design. The QFD method ensures that the new design meets customer wants and needs. The research aims to provide innovative solutions in developing clothing dryer products and reduce reliance on electric power. The proposed design will offer convenience and effectiveness in drying clothes while considering the unpredictable weather conditions in Indonesia.

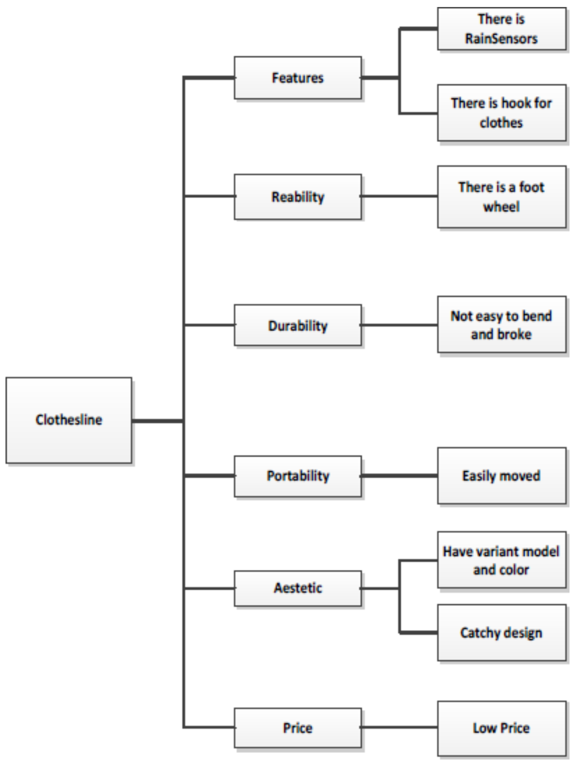


Figure 2.2.2 - Clothesline tree diagram analysis

2.2.3 Simulation and design of smart clothesline using fuzzy for weather forecast

Weather conditions strongly influence human activities in various sectors such as agriculture, tourism, and aviation. Having accurate weather information is crucial for decision-making and anticipating the impact of weather changes. However, predicting climate change is challenging. Fuzzy logic systems provide an effective and accurate solution for weather prediction by considering imprecise or vague linguistic variables (Tatyantoro, 2022). Fuzzy logic has been successfully applied in various fields, including industrial process control, medical instrumentation, and decision support systems. In weather prediction, fuzzy logic has been used to estimate water levels, predict rain to prevent flooding, and develop weather prediction applications using methods like ANFIS (Adaptive Neural Fuzzy Inference System). The advantages of fuzzy logic lie in its ability to handle complex and imprecise variables, making it suitable for modeling real-world phenomena. This project aims to utilize fuzzy logic to develop a weather prediction system that can assist in decision-making and improve the accuracy of weather forecasts.

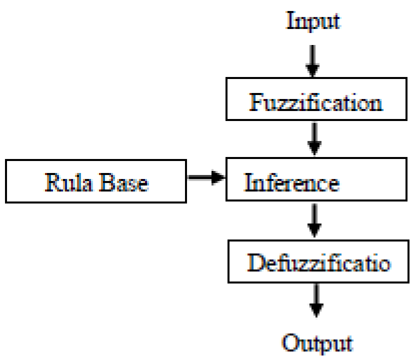


Figure 2.2.3 - Process fuzzy logic algorithm

2.3 Analysis/ Comparison of Existing System

Only after completing the evaluation of the three current systems can comparisons be made based on each system's description, development process, technologies used, and advantages and disadvantages. There will be obvious distinctions made between these systems based on their importance to the project's target objectives, and significant components that could be incorporated into the design of the *'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* can be identified.

2.3.1 Analysis of comparison on existing systems

Title/Criteria	**Intelligent Drying Rack System based on Internet of Things**	**Propose design of smart clothesline with the tree diagram approach analysis and quality function deployment method for Indonesia weather**	**Simulation and design of smart clothesline using fuzzy for weather forecast**
Description	Design of a smart clothes rack with a focus on recycling and automatic one-button fall off of clothes.	Design of an automatic clothes dryer that protects clothes from rain.	Simulation of a weather prediction system
Technique/Method	Fuzzy logic	Fuzzy logic	Fuzzy logic
Tools/Technology	-microcontroller: STM32F103C8T6 -sensors: DHT11 sensor and light sensor -stepping motors: 57-type and 42-type	-microcontroller: ATMEGA8525 -sensors: rain sensors	-microcontroller: Arduino Mega -sensors: DHT22 sensor and LDR sensor

Table 2.3.1 - analysis comparison on existing system

2.3.2 Relevance of comparison with project title

System / Comparison	**Intelligent Drying Rack System based on Internet of Things**	**Propose design of smart clothesline with the tree diagram approach analysis and quality function deployment method for Indonesia weather**	**Simulation and design of smart clothesline using fuzzy for weather forecast**
Advantages	Offers a telescopic mechanism for expanding the drying area based on demand.	Protects clothes from rain and provides an ergonomic design for consumers.	Utilizes fuzzy logic for more accurate weather predictions.
	Provides real time weather updates for users to choose appropriate clothes for drying.	Using QFD methods to meet consumer demand and reduce energy consumption.	Real-time updates
Disadvantages	specific technical details or limitations not mentioned.	requires additional sensors and components for rain detection and protection.	relies on simulation rather than real-world implementation for testing.
	potential complexity and cost associated with the use of multiple components and mechanism	need electric power	limited information on the validation and effectiveness of the fuzzy logic system for weather prediction in practical

Table 2.3.2 – Advantages and Disadvantages comparison between 3 existing system

2.4 Summary

After reviewing the three existing systems, it was possible to clearly distinguish the differences between the systems in terms of their many features, the way each one employs IoT technologies and a host of other factors. After that, a comparison of these three systems can be done in order to identify each one's advantages and disadvantages in relation to the '*Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic*' system that has been proposed.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter discusses the overall approach or framework of '*Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic*'. It should cover method/technique or approach to be used whereas the methodology in details to accomplish the project. The content for this chapter can contain: Introduction, architecture design, dashboard design, analytic feature design, potential use of proposed solution and reference.

The architecture design section delves into the technical aspects of the automated drying rack system. It outlines the overall structure and components involved in the system, such as sensors, actuators, control units, and communication interfaces. The section also discusses the integration of fuzzy logic into the architecture, explaining how it enables intelligent decision-making and adaptive control based on real-time inputs. Then, the dashboard design section focuses on the user interface aspect of the system. It describes the design and layout of the graphical user interface (GUI) that allows users to interact with the automated drying rack. This includes features such as displaying real-time information on weather, monitoring drying rack and send the alarm trigger. The section may also discuss the considerations taken into account for creating a user-friendly and intuitive interface.

The analytic feature design section explores the analytical capabilities of the system. It explains how data collected from sensors can be processed and analysed to derive insights and optimize the drying process. This may involve techniques such as data visualization, pattern recognition, and predictive modelling. The section highlights the value of analytics in improving efficiency, identifying potential issues, and providing actionable recommendations for further optimization. The potential use section discusses the practical applications and benefits of the proposed '*Intelligent Drying Rack Control*

System with Weather Condition Prediction Using Fuzzy Logic'. It explores the various real-time situations where the solution can be implemented, such as residential households, commercial laundries, and the hospitality industry. The section emphasizes how the system's adaptability, energy efficiency, and high-quality drying results make it suitable for a wide range of users and contexts.

Overall, this chapter provides a comprehensive overview of the intelligent drying rack with fuzzy logic system, covering the approach, methodology, and potential use cases. It sets the foundation for understanding the subsequent chapters that delve into more specific details, implementation techniques, and experimental results.

3.2 Project Management Framework : Rapid Application Development

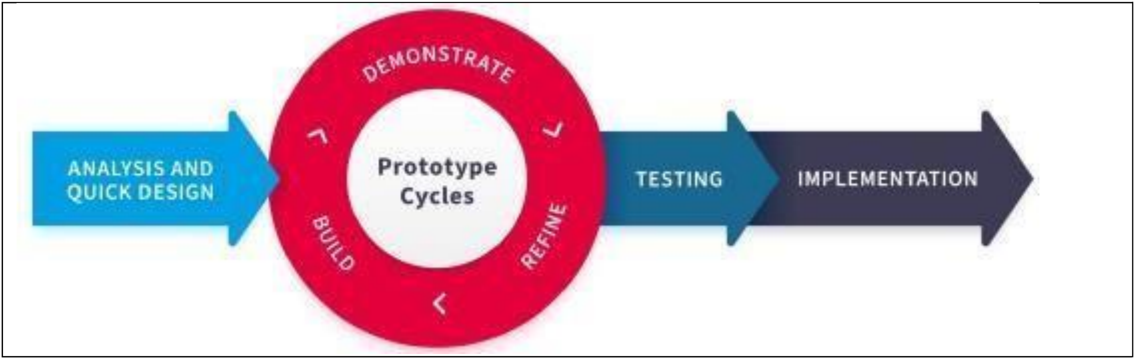


Figure 3.2 – Rapid Application Development SDLC

Figure 3.2 clearly illustrates the phases conducted within the RAPID APPLICATION DEVELOPMENT (RAD) methodology. This was determined to be the best SDLC method for the ‘*Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic*’ system since it would allow for a system to be produced much more quickly and with a higher level of product quality. This is because the RAD technique is highly flexible and responsive to changes, making it ideal for the development of this system because of the many requirements that would always call for minor system adjustments to be made quickly. It is also an iterative process, allowing for simple revisiting of various development phases like the refining, building, and demonstrating process, allowing for enhanced revisions throughout the project's development until it is finished.

3.2.1 PHASE 1 : Analysis and Quick Design

Every required aspect of the system, such as the project's functional and non-functional requirements, along with the software and hardware requirements, must be gathered and analysed within this process. This process is called the ANALYSIS AND QUICK DESIGN process. It is the first phase of the development of *the 'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* system. Only then will it be possible to create a system design that is specifically designed to address all the objectives and needs that have been put forth.

As mentioned in earlier chapters, the *'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* system's reachable goals and scope have already been determined. Additionally, we have examined currently available systems on the market and have effectively pinpointed significant variations in regard to each of their strengths and shortcomings that must be taken into account throughout the construction of this project.

Additional needs, such as the previously mentioned functional and non-functional requirements as well as software and hardware specifications required to correctly construct this specific system and listed in PROJECT needs Chapter 3.3, can be found here.

3.2.2 PHASE 2 : Prototype Cycling

To properly illustrate the inner workings of the *'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* system, additional elements are refined in this phase and will be further discussed in the PROPOSED DESIGN Chapter 3.4. These elements include the proposed system and data design as well as a defined IoT architecture.

Following that, the build process—which marks the official beginning of the system's development—can be started. This phase involves all the labor-intensive tasks, such as setting up the sensors, connecting the gear through networking, and creating the overall database. Only when it is finished will it be possible to go on to the demonstration procedure, in which a system prototype that has been produced is assessed before the actual testing of the finished product. Here, any future updates to the system requirements that result in the addition of new modules.

This procedure can be repeated in order to improve the system's updates, monitor the prototype, and plan swifter building until a working model is ultimately finished and prepared for testing in real-world scenarios.

3.2.3 PHASE 3 : Testing

During the testing phase, the *'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* will be put through its paces to ensure that every feature is working properly and any faults or glitches will be fixed. This phase may be repeated as necessary to accommodate additions and modifications.

End users' comments on the user interface and functionality will be gathered in order to further enhance the product overall. This is accomplished by allowing people to provide thoughtful feedback during this phase, offering adjustments, revisions, or fresh concepts that address issues as they arise. If the feedback is only positive, the last step can be carried out; otherwise, the prototype cycling process is resumed.

3.2.4 PHASE 4 : Implementation

The RAD methodology's last step involves resolving any technical difficulties that cropped up during early prototype in order to optimize implementation for increased stability and maintainability as we move closer to the launch of the finished product. The *'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* system's important components are all transferred to a live production environment, where extensive testing is conducted to find any remaining product flaws. Before finally providing the client a finished product, thorough documentation and other essential maintenance chores must be carried out.

3.3 Project Requirements

The requirements for the *'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* system are further discussed in this section. These will comprise explanations of the project's software and hardware needs as well as its functional and non-functional requirements, which will act as guidelines for the system's design process.

3.3.1 Functional & Non-Functional Requirements

To guarantee that the *'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* system functions as intended, both the functional and non-functional requirements must be clearly stated from the beginning of development. This necessitates understanding the distinctions between those two types of needs as well as how to define each.

While NON-FUNCTIONAL REQUIREMENTS describe how the system should operate, primarily in terms of system usability to ensure a well performing system, FUNCTIONAL REQUIREMENTS specify what the system must do in response to various inputs and can be made up of both product features and user requirements.

These definitions state that the *'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* system must adhere to the functional and non-functional requirements listed below:-

FUNCTIONAL REQUIREMENTS	**NON-FUNCTIONAL REQUIREMENTS**
1. Intelligent drying rack control system must display all recorded values into simple understandable data visualizations that are integrated into a web-based dashboard application. 2. Intelligent drying rack control system must allow for both automated and control of	1.Intelligent drying rack control system should be developed using a Laravel framework for the web-based dashboard application fit for Python language, HTML and JavaScript for web page development and microprocessor operation of the sensors and actuators.

actuators such as servo motors and LED via event triggers on dashboard. 3. Intelligent drying rack control system must enable dashboard interface controls to be used to operate the button.	2. A local MySQL database should be used by the intelligent drying rack control system for secure data storage and query activities. 3. Intelligent drying rack control system should use the WebSocket connection protocol for real-time alerts and important event triggers and the MQTT connection protocol for sensor readings. 4. Intelligent drying rack control system should display on the dashboard for the latest 1-minute update intervals.

Table 3.3.1 – List functional and non-functional requirements

3.3.2 Software Requirements

SOFTWARE	PURPOSE OF UTILIZATION
Microsoft Visual Studio Code	To allow coding for :- • Building the PHP, HTML, and JavaScript pages for the Intelligent drying rack control system web application. • A Raspberry Pi microprocessor's operating instructions for reading and transmitting sensor data values and operating actuators. • Ensuring that the server and devices follow the network connection instructions for data transmission.
VNC viewer	Enables Artificial Intelligence (AI) to send messages via WebSocket to activate the actuators controller by the Python code.
XAMPP - phpMyAdmin	Enables database storing and query within local MySQL environment.
Composer	Manages library dependencies in PHP.
Laravel PHP framework	Providing a framework structure for Intelligent drying rack control system web application development.
Spyder IDE	Platform for Artificial Intelligence (AI) development and debugging.
Google Chrome	Web browser for Intelligent drying rack control system web application implementation.

Table 3.3.2 – List and description of required software

3.3.3 Hardware Specifications

NO.	DEVICE TYPE & PURPOSE	DEVICE
1.	**CLIENT/SERVER COMPUTER** • Server Computer : To host Intelligent drying rack control system web application for multiple users simultaneously, local MySQL database as well as AI fuzzy program • Client Computer : To allow for display and interaction of Intelligent drying rack control system web application.	**SERVER COMPUTER** **CLIENT COMPUTER**
2.	**MICROPROCESSOR** • To collect data from DHT11 temperature and humidity sensor to be sent over MQTT and WebSocket connection to be used by the system.	**RASPBERRY PI 3B+**
3.	**MICROPROCESSOR** • To collect data from photocell sensor and rain sensor to be sent over MQTT and WebSocket connection to be used by the system.	**ESPDUINO 32**

4.	**SENSOR** • To measure the weather environmental temperature and humidity.	**DHT11 TEMPERATURE & HUMIDITY**
5.	**SENSOR** • To measure the light intensity of the ambient light.	**PHOTOCELL SENSOR**
6.	**SENSOR** • To detects the rain drops.	**RAIN SENSOR**
7.	**ACTUATOR** • To pull in and out the drying rack	**SERVO MOTOR**
8.	**OUTPUT** • To act as alarm when the rain sensor detects the rain drops.	**LED**

3.4 Proposed Design

The *'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* system's proposed design will be further developed in this chapter. Here, more information will be provided on subjects like the general system sketch and operation.

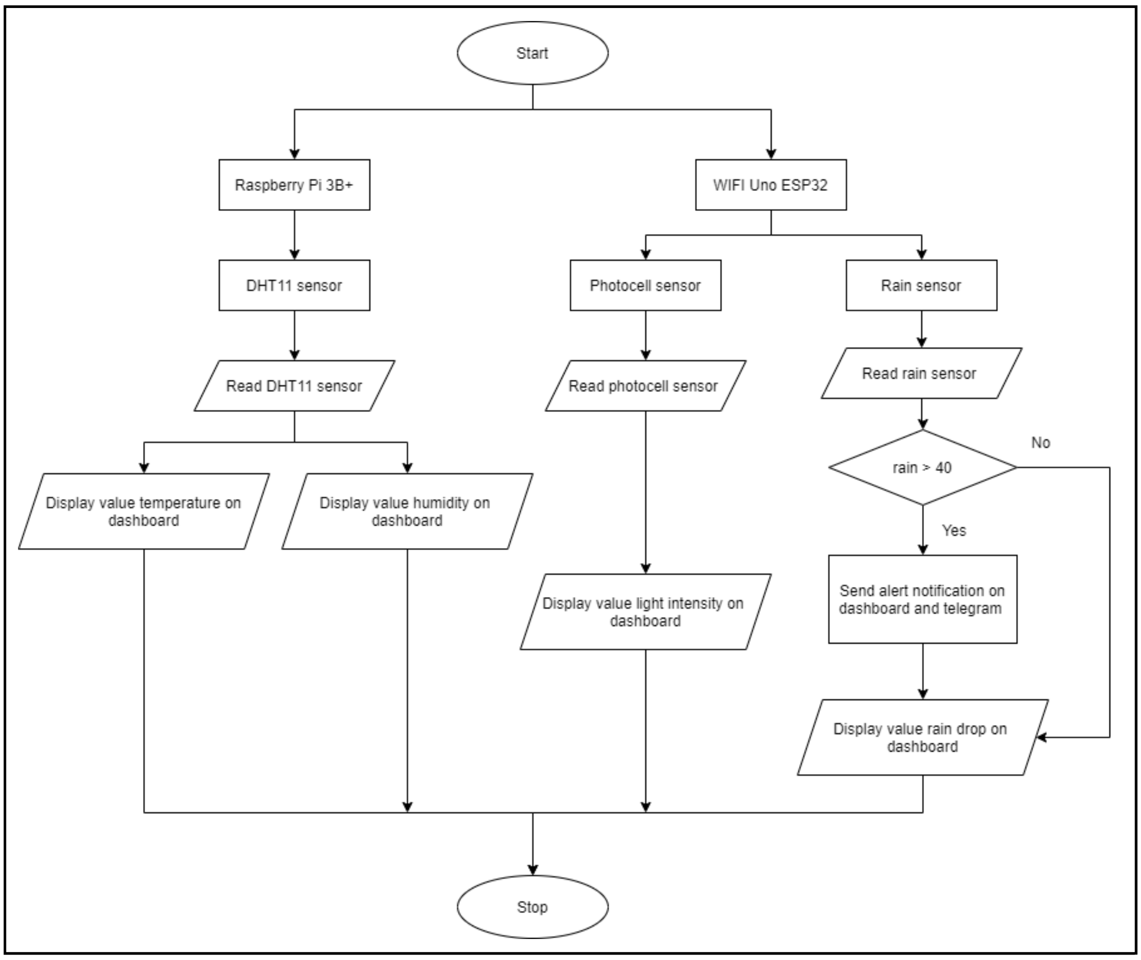

Start
Raspberry Pi 3B+
WIFI Uno ESP32
DHT11 sensor
Photocell sensor
Rain sensor
Read DHT11 sensor
Read photocell sensor
Read rain sensor
rain > 40
No
Yes
Display value temperature on dashboard
Display value humidity on dashboard
Display value light intensity on dashboard
Send alert notification on dashboard and telegram
Display value rain drop on dashboard
Stop

Figure 3.4 - Flowchart Architecture

The intelligent drying rack architecture utilizes two microcontrollers, namely the Raspberry Pi 3B+ and the ESPDuino 32, to create a comprehensive system. The flowchart begins with the Raspberry Pi, which directly connects to a DHT11 sensor. This sensor is responsible for measuring temperature and humidity levels. The ESP32, on the other hand, connects to both the photocell and the rain sensor via Wi-Fi.

The photocell and rain sensor detect light levels and rainfall respectively. These sensors provide readings of the sensed values, which are then transmitted to the respective microcontrollers. The readings are then processed and displayed on a dashboard. This dashboard serves as a user interface, allowing users to monitor and visualize the data collected by the sensors.

Furthermore, in the case of the rain sensor, if the detected rain value exceeds 40 (presumably a predefined threshold), an alert notification is triggered. This notification is sent to both the dashboard and the messaging platform Telegram, providing real-time updates about the rain condition. This alert system ensures that users are promptly informed of significant rainfall, allowing them to take appropriate actions to protect their belongings on the drying rack.

3.5 Architecture Design

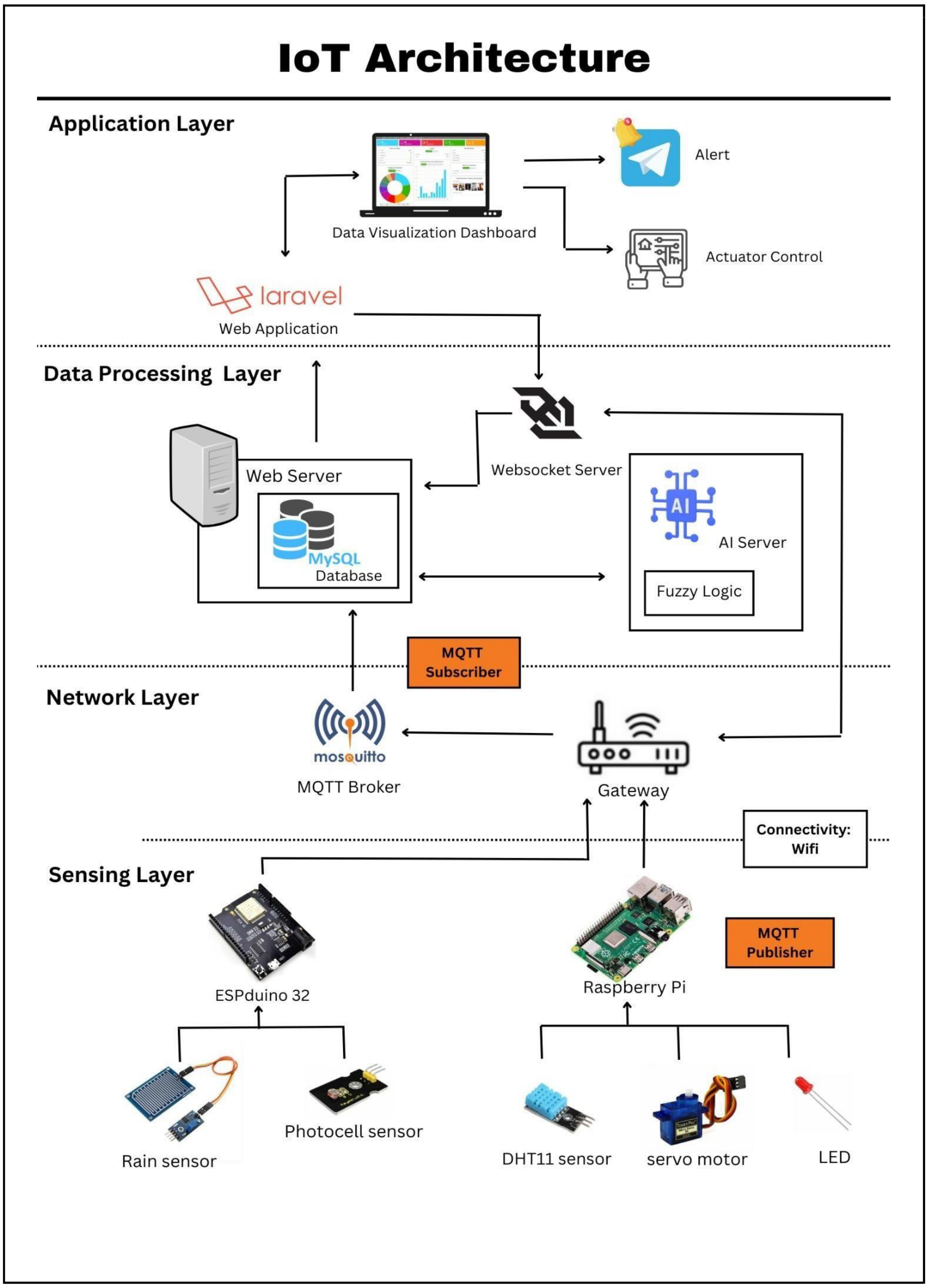


Figure 3.5 - IoT Architecture

The proposed design for the *'Intelligent Drying Rack Control System with Weather Condition Prediction Using Fuzzy Logic'* incorporates several components and technologies to create an intelligent and connected drying system. Here's a breakdown of the design elements. For the sensing layers, the sensors that use are DHT11 sensor, photocell sensor and rain sensor and for the microcontrollers that use are raspberry pi 3B+ and WIFI Uno ESP32. The DHT11 sensor measures temperature and humidity levels in the environment. It provides data about the ambient conditions that can be used to optimize the drying process. The photocell sensor detects the intensity of light in the surroundings. It helps determine whether the drying rack is exposed to direct sunlight or if it's located in a shaded area, allowing the system to adjust the drying parameters accordingly. The rain sensor detects the presence of rain. It ensures that the drying process is paused or adjusted when rain is detected to prevent damage to the clothes.

In terms of connectivity, the photocell sensor and rain sensor are connected to WIFI Uno ESP32 via a wired connection. On the other hand, the DHT11 sensor is directly connected to the raspberry pi 3B+. The WIFI Uno ESP32 and raspberry pi takes on the role of an MQTT publisher, sending data to the MQTT broker through a gateway. For the actuator use is servo motor. The servo motor controls the movement of the drying rack. It can be used to adjust the position of the rack or to rotate it for even drying. Then, the output is LED. The LED serves as an output indicator, providing visual feedback to the user about the status of the drying rack.

For the network layer, connection protocols that are used in this automated drying rack are MQTT and WebSocket. The MQTT stands for Message Queuing Telemetry Transport protocol used for efficient and reliable communication between the drying rack and other devices or services. It enables real-time data exchange and control commands. In this project, the gateway is connected via a Wi-Fi connection to create connectivity. Data transmission from the microcontroller to the MQTT broker is handled by the gateway. It will collect data and send it as JSON format to the web server by using MQTT server. The MQTT broker serves as the network layer's main hub for data transport. A subscriber receives the information from the gateway when it subscribes to a specific user and subject. The data is kept by the MQTT broker, who makes sure the right subscribers receive it. On the Internet of Things system, effective and dependable data transfer and communication are provided by the gateway, the MQTT broker, and the network layer. The gateway enables connection between the MQTT publisher and the broker, while the MQTT broker stores and disperses data to subscribers in accordance with their subscriptions to specific users and subjects.

While in the data processing layer, the MySQL database is used to store the collected sensor data and other relevant information. It provides a structured and reliable storage solution for historical data and enables data analysis and reporting. Besides that, this layer consists of two servers which are AI server (Artificial Intelligence) and WebSocket server. In the AI server, the fuzzy logic system has been implemented. The three variables used in this study's fuzzy system are temperature, rain, and light intensity. Cold, mild and hot are the three linguistic input variables for temperature. Dry, light rain and heavy rain are the three linguistic variables that make up the input variable for rain. Dark, moderate, and bright are the three language variations of light intensity. A condition for the weather output variable with a sunny, cloudy and rainy. The values of the fuzzy logic algorithm are derived from each variable. The module will transfer numerical values to fuzzy sets by fuzzification. The input value will be transformed into a linguistic value for fuzzy input.

For the Application layer, software that is used in dashboard and notifications Laravel and telegram. Laravel is a PHP-based web application framework that can be used to develop a user-friendly dashboard. It provides a robust and scalable platform for managing and visualizing the drying rack's data, settings, and controls. Telegram is a messaging platform that can be utilized to send notification alerts to the user. It can deliver real-time updates about the drying process, completion status, or any issues that require attention.

The overall architecture of this proposed design involves the sensors (DHT11, photocell sensor, rain sensor) collecting data about the environment, which is processed by the fuzzy logic system within the microcontroller. The fuzzy logic system uses the sensor data to make intelligent decisions regarding the drying process. The actuator (servo motor) controls the movement of the drying rack based on the decisions made by the fuzzy logic system. The LED provides visual feedback to the user about the current status of the drying rack. The MQTT and WebSocket protocols enable communication between the drying rack and the dashboard/user interface, allowing for real-time data exchange and control commands. The MySQL database stores the collected data for further analysis and reporting. The Laravel-based dashboard provides a user-friendly interface to monitor and control the drying rack, while Telegram serves as a platform for sending notification alerts to the user, keeping them informed about the drying process. This integrated design combines sensor data, fuzzy logic decision-making, actuator control, connectivity protocols, database storage, and user interfaces to create an intelligent and connected smart drying rack system.

3.6 Data Collection

The data will be collected depending on fuzzy logic classification. The data will be collected from temperature value, light intensity and rain drop value. The sensors that will be used to collect the data are DHT11 sensor, photocell sensor and rain sensor. To facilitate the interaction with these sensors and gather the necessary data, a functional model will be developed. This model serves the purpose of autonomously collecting the data, removing the need for external interventions. Continuous data collection ensures the acquisition of a comprehensive dataset that accurately represents the dynamic behaviour of the drying rack variables over time. Specifically, the data being collected in this thesis pertains to temperature value, light intensity and rain drop value, which serve as input or membership functions for the fuzzy logic analysis of the drying rack operations. Figure 3.6.1, 3.6.2, 3.6.3 below shows the fuzzy input membership function of the temperature, humidity and light intensity from the previous journal (Tatyantoro, 2022).

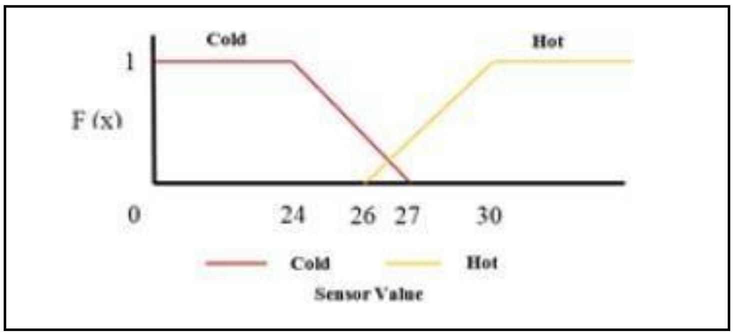


Figure 3.6.1 - member function graphic of temperature

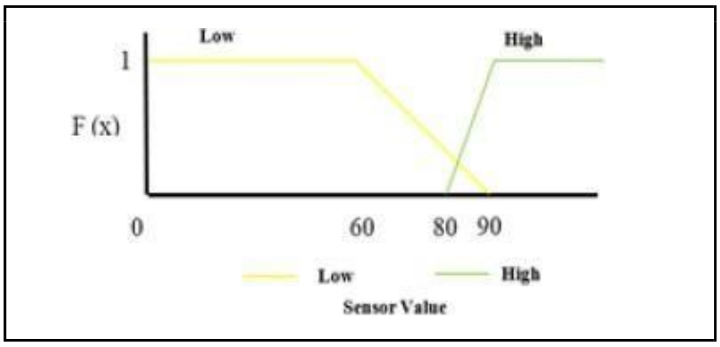


Figure 3.6.2 - member function graphic of humidity

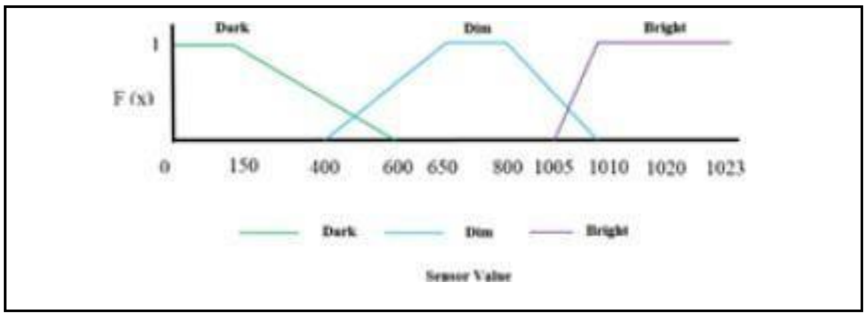


Figure 3.6.3 - member function graphic of light intensity

In figure 3.6.4 is the membership function of the fuzzy output from defuzzification from the existing system.

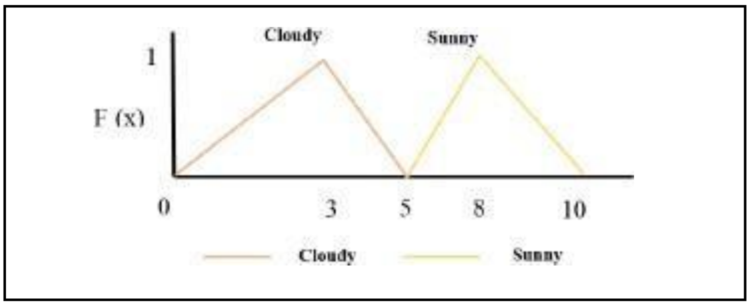


Figure 3.6.4 - member function graphic of out

In the inference module of a fuzzy logic system, simulated decision making takes place based on fuzzy concepts using knowledge rules. These rules are determined by combining linguistic variables and the system's knowledge. The output of the inference process is a feasibility value, which represents the degree of certainty or confidence in a particular decision or action (Tatyantoro, 2022).

After the inference process, the feasibility values can be further processed through the defuzzification step to obtain a crisp output value or a set of crisp values that represent the final decision or prediction based on the fuzzy logic system's rules and inputs. It's important to note that the specific fuzzy rules, linguistic variables, and membership functions used in a weather prediction system may vary depending on the system's design and requirements.

3.7 Dashboard Design

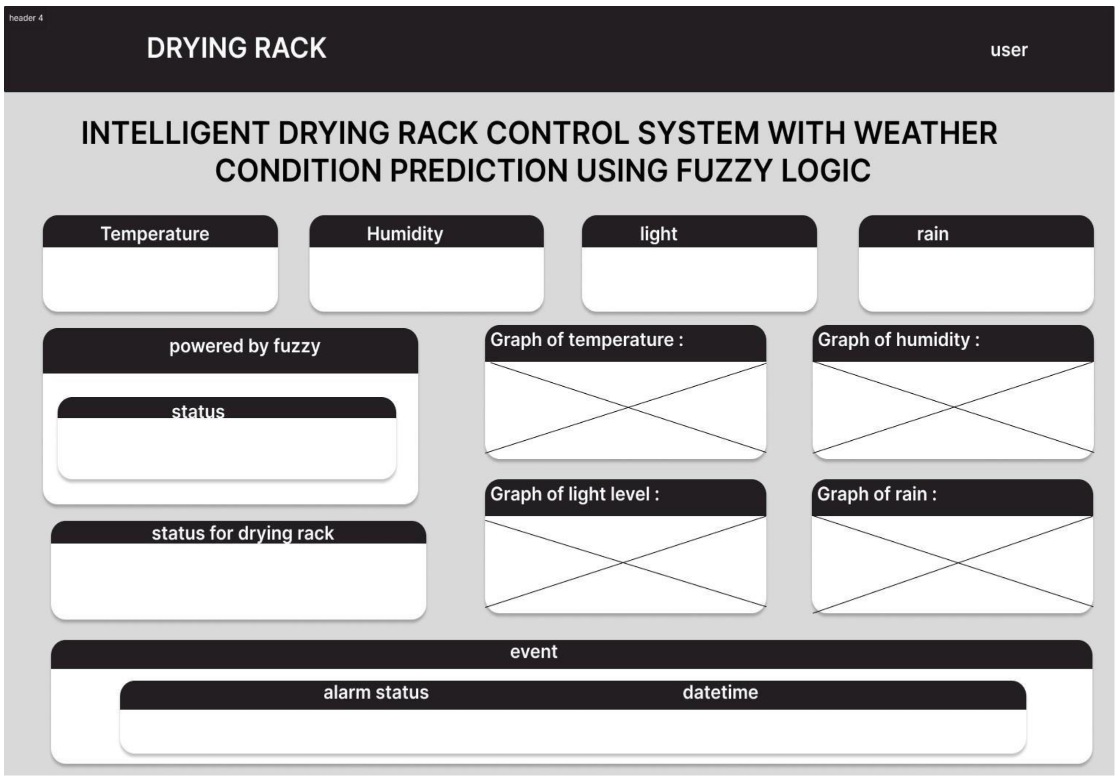


Figure 3.7 - Wireframe Dashboard

The figure 3.7 shows wireframe dashboard for the Smart drying rack system. The home dashboard for a smart drying rack with a fuzzy logic system offers a user-friendly interface. The home dashboard serves as a centralized hub, providing real-time information and control options. In the home section, users can quickly access real time information about humidity, temperature, light intensity, and the presence of rain. This display enables users to assess the current drying conditions at a glance, allowing them to make informed decisions and adjustments as needed. The analytics section provides deeper insights into the drying process. The control section empowers users to personalize settings and preferences, enabling a customized and optimized drying experience. Finally, the alarm status alerts the user if the rain sensor was detected.

3.8 Analytic Feature Design

The intelligent drying rack control system incorporates fuzzy logic to enhance its efficiency and adaptability in response to changing environmental conditions. Fuzzy logic is a computational approach that deals with uncertainty and imprecise data by utilizing linguistic variables and rules. In this system, the inputs are temperature, rain, and light levels, while the output is the weather condition classification.

The temperature input is divided into three linguistic variables: cold, mild, and hot, with respective ranges of 0-16, 17-33, and 34-40. The rain input is classified into dry, lightrain, and heavyrain, corresponding to the ranges of 2-30, 31-45, and 46-100. Similarly, the light input is categorized as dark, moderate, and bright, spanning the ranges of 0-30, 31-50, and 51-100. Using fuzzy logic, the control system processes these inputs and assigns degrees of membership to each linguistic variable based on their measured values. These memberships represent the degree of truth or relevance of each input to a particular linguistic variable. The system then applies a set of predefined fuzzy rules to determine the output, which is the weather condition classification.

The output linguistic variable is divided into three categories: sunny, cloudy, and rainy. Each category has a defined range, with sunny ranging from 0 to 67.5, cloudy from 68 to 80.0, and rainy from 81.0 to 100. The fuzzy logic system combines the degree of membership of the input variables and applies fuzzy inference rules to determine the appropriate weather condition classification based on the inputs. By employing fuzzy logic, the intelligent drying rack control system can effectively handle imprecise and uncertain data, allowing it to adjust its drying operation based on the inferred weather conditions. This enables the system to optimize the drying process by making informed decisions about the duration and intensity of drying, ensuring efficient energy usage and protecting clothes from adverse weather conditions.

Based on the given information about the input and output variables, here are some fuzzy rules that can be defined:

```
rule1 = temperature['cold'] & rain['dry'] & light['dark'], weather['cloudy']
```

```
rule2 = temperature['cold'] & rain['dry'] & light['moderate'], weather['sunny']

rule3 = temperature['cold'] & rain['dry'] & light['bright'], weather['sunny']

rule4 = temperature['cold'] & rain['lightrain'] & light['dark'], weather['rainy']

rule5= temperature['cold'] & rain['lightrain'] & light['moderate'], weather['rainy']

rule6 = temperature['cold'] & rain['lightrain'] & light['bright'], weather['rainy']

rule7 = temperature['cold'] & rain['heavyrain'] & light['dark'], weather['rainy']

rule8 = temperature['cold'] & rain['heavyrain'] & light['moderate'], weather['rainy']

rule9 = temperature['cold'] & rain['heavyrain'] & light['bright'], weather['rainy']

rule10 = temperature['mild'] & rain['dry'] & light['dark'], weather['cloudy']

rule11 = temperature['mild'] & rain['dry'] & light['moderate'], weather['sunny']

rule12 = temperature['mild'] & rain['dry'] & light['bright'], weather['sunny']

rule13 = temperature['mild'] & rain['lightrain'] & light['dark'], weather['rainy']

rule14 = temperature['mild'] & rain['lightrain'] & light['moderate'], weather['rainy']

rule15 = temperature['mild'] & rain['lightrain'] & light['bright'], weather['rainy']

rule16 = temperature['mild'] & rain['heavyrain'] & light['dark'], weather['rainy']

rule17 = temperature['mild'] & rain['heavyrain'] & light['moderate'], weather['rainy']

rule18 = temperature['mild'] & rain['heavyrain'] & light['bright'], weather['rainy']

rule19 = temperature['hot'] & rain['dry'] & light['dark'], weather['cloudy']
```

```
rule20 = temperature['hot'] & rain['dry'] & light['moderate'], weather['sunny']

rule21 = temperature['hot'] & rain['dry'] & light['bright'], weather['sunny']

rule22 = temperature['hot'] & rain['lightrain'] & light['dark'], weather['rainy']

rule23 = temperature['hot'] & rain['lightrain'] & light['moderate'], weather['rainy']

rule24 = temperature['hot'] & rain['lightrain'] & light['bright'], weather['rainy']

rule25 = temperature['hot'] & rain['heavyrain'] & light['dark'], weather['rainy']

rule26 = temperature['hot'] & rain['heavyrain'] & light['moderate'], weather['rainy']

rule27 = temperature['hot'] & rain['heavyrain'] & light['bright'], weather['rainy']
```

3.9 Potential Use of Proposed Solution

The proposed automatic drying rack solution with a fuzzy logic system has many potential uses in real-time situations. It can be used at home to dry clothes efficiently, adjusting the appropriate time and time for drying based on factors such as humidity, temperature, light intensity and the presence of rain. In commercial laundries, this system optimizes the drying process for large volumes of laundry, reducing energy consumption and improving drying quality. The hospitality industry can benefit from this solution by offering guests a reliable and convenient drying solution with customizable settings. Garment manufacturers can integrate the system into their production process, ensuring efficient and high-quality garment drying. In addition, the solution finds value in research and development settings, enabling detailed analysis of drying characteristics. In short, automatic drying racks with fuzzy logic enable efficient, safe and accurate drying across a wide range of applications, from residential to commercial and industrial environments.

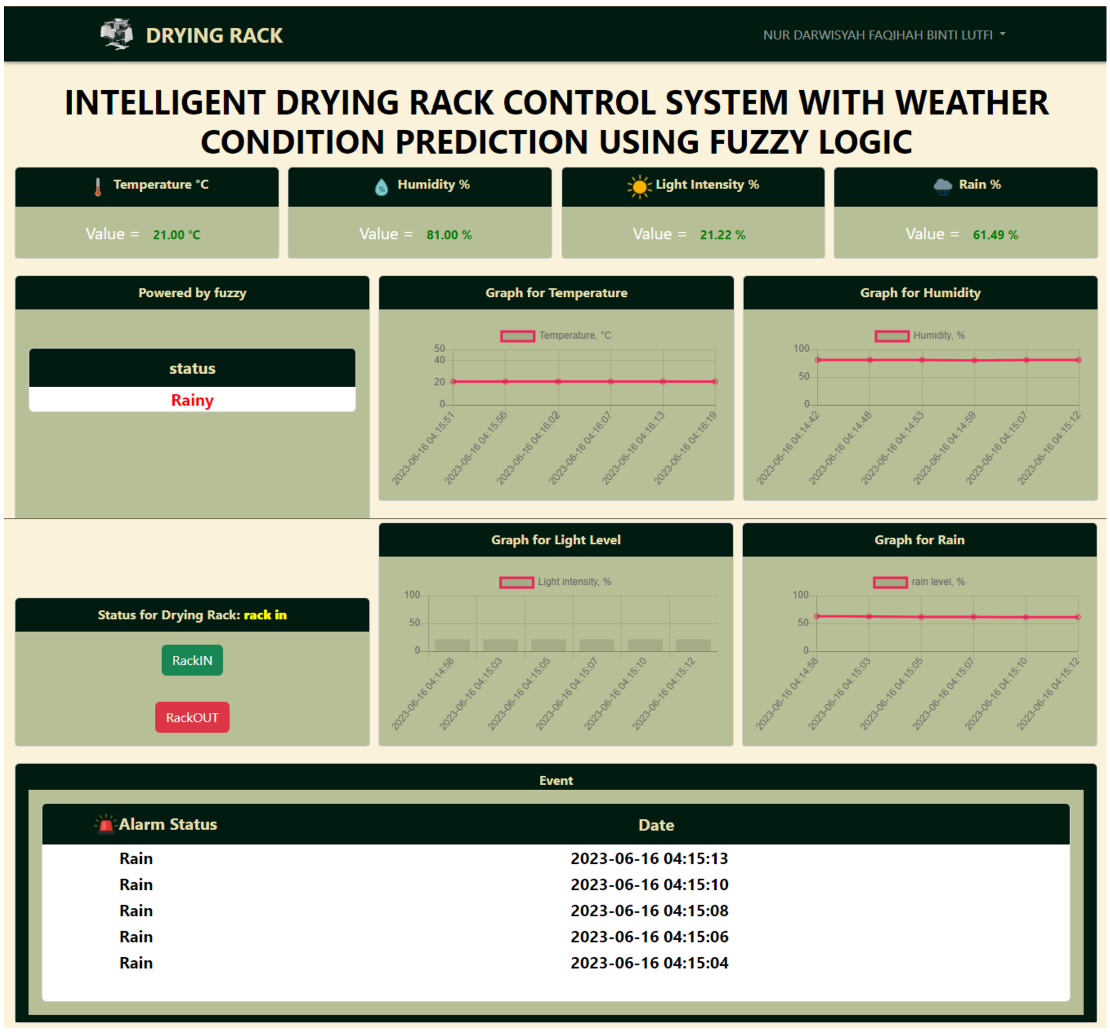
CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter discusses the results and discusses the '*Intelligent Drying Rack Control System with weather condition prediction using fuzzy logic'*. It should show implementation and results of the project. The content for this chapter can contain Introduction, dashboard results and discussion and fuzzy results and discussion. It also provides a full study of the outcomes while delving into the particulars of the developed dashboard and the fuzzy graphs that were generated. At the beginning of the chapter, the characteristics and capabilities of the dashboard are discussed. It displays techniques for data visualisation, user interface layout, and the utilisation of multiple sensors and data gathering devices. Screenshots or other visual representations of the dashboard may be provided to offer readers a thorough understanding of its design and usefulness.

4.2 Dashboard Results and Discussion



INTELLIGENT DRYING RACK CONTROL SYSTEM WITH WEATHER CONDITION PREDICTION USING FUZZY LOGIC

Temperature °C	Humidity %	Light Intensity %	Rain %
Value = 21.00 °C	Value = 81.00 %	Value = 21.22 %	Value = 61.49 %

status: Rainy

Status for Drying Rack: rack in

Event

Alarm Status	Date
Rain	2023-06-16 04:15:13
Rain	2023-06-16 04:15:10
Rain	2023-06-16 04:15:08
Rain	2023-06-16 04:15:06
Rain	2023-06-16 04:15:04

Figure 4.2.1 - Dashboard of the system

A variety of significant functions and functionalities are available on the dashboard of the intelligent drying rack control system, ensuring a user-friendly and secure experience. Users can register, log in, and log out of the system to gain access, which ensures secure access and protects their personal data. The dashboard shows data in several cards and gives consumers access to crucial weather-related information. The values of temperature (in degrees Celsius), humidity (in percentage), light intensity (in percentage), and precipitation (in percentage) are shown on these cards. Users may quickly judge whether the climate is appropriate for drying their garments by keeping an eye on these characteristics. The system's most important component is the rain sensor, which activates an alert status and promptly notifies users of the unfavourable weather conditions if the rain value surpasses 40 (classified as rainy). Users can utilise this to act quickly and keep their clothes from getting wet.

Based on the data gathered, the "Powered by Fuzzy" card forecasts the weather. However, it does not disclose the precise fuzzy rules or variables utilised in the calculation; it merely shows the outcome status. As a result, the user interface is made simpler and people aren't overloaded with technical information while still getting a good grasp of the weather. Users can examine temperature, humidity, light intensity, and rain presence trends over time by viewing real-time data in the graph card. This gives them the information they need to assess the drying process and modify their plans as necessary.

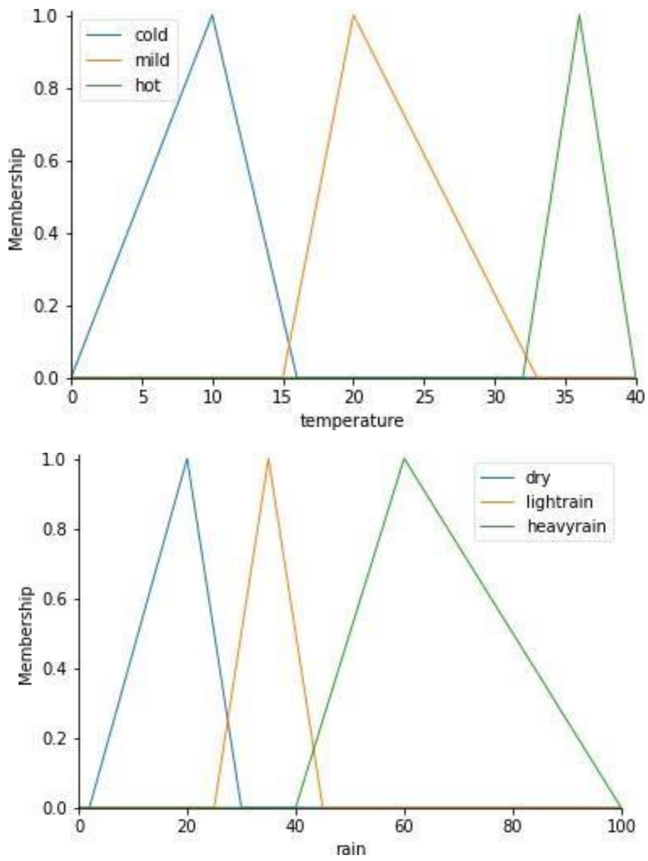
The dashboard has "Rack In" and "Rack Out" buttons for convenience and control. Depending on the climate, users can easily adjust where the drying rack is placed. When necessary, this feature makes sure that clothing is shielded from the rain or intense sunshine. Last but not least, if rain is detected, the alert status card shows the date and time that correlate to the alarm status. This function alerts consumers to any rain events that can affect their drying process so they can react quickly.

Overall, the intelligent drying rack control system's dashboard offers a wide range of capabilities and functionalities. It provides real-time data visualisation, fuzzy logic-powered weather condition analysis, user-friendly controls, and alarms for unfavourable weather. This guarantees the drying rack's effective and convenient operation while keeping users informed and in charge of the drying process.

4.3 Fuzzy Results and Discussion

The intelligent drying rack control system integrates weather condition prediction using fuzzy logic, allowing it to determine the prevailing weather conditions based on inputs from sensors such as the DHT11 sensor, photocell sensor, and rain sensor. The fuzzy logic system encompasses linguistic variables, membership functions, and a set of rules, enabling accurate weather classification.

The linguistic variables, including temperature, light intensity, and rain intensity, are defined within specific ranges and assigned membership functions that depict their degrees of membership. These membership functions facilitate the assignment of linguistic terms to the inputs, such as "cold," "mild," "hot," "dry," "lightrain," "heavyrain," "dark," "moderate," and "bright," based on their measured values. By utilizing linguistic terms, the system can effectively capture and represent the imprecision and uncertainty associated with real-world weather conditions. Shown as Figure 4.3.1 below.



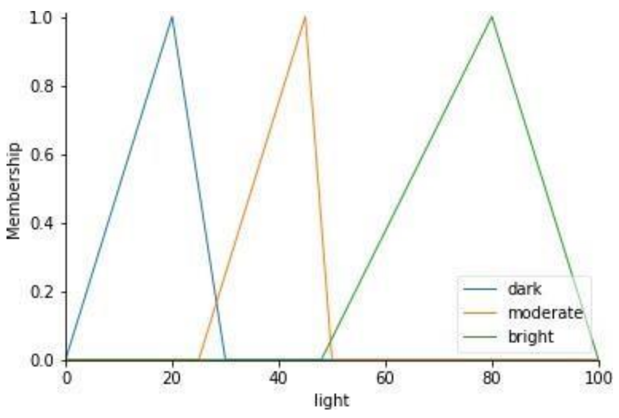


Figure 4.3.1 - Membership for temperature, rain and light

The fuzzy logic system employs a set of predefined rules that establish the relationships between the input variables and the output variable, which in this case are the weather conditions, including "sunny," "cloudy," and "rainy." These rules define how the system should respond to various combinations of temperature, light intensity, and rain intensity, ultimately determining the condition of the weather. The condition shown in Figure 4.3.2 below.

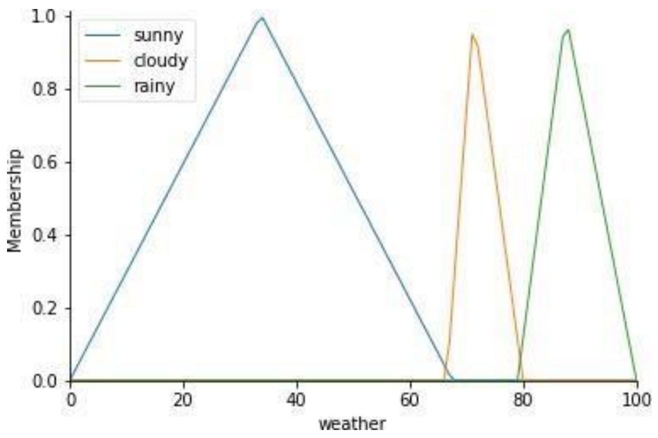


Figure 4.3.2 - Output / Condition of fuzzy

By applying fuzzy inference, which involves combining the membership values of the linguistic terms and applying fuzzy logic operators, the system can make accurate predictions about the weather conditions. This inference process takes into account the rules and the fuzzy sets associated with the inputs to generate a crisp output, representing the most probable weather condition.

The fuzzy logic-based weather condition prediction in the intelligent drying rack control system provides a robust and adaptive approach to respond to changing environmental factors. By incorporating linguistic variables, membership functions, and a set of rules, the system can effectively handle imprecise and uncertain data, enabling accurate weather classification. This, in turn, allows users to make informed decisions regarding the operation of the drying rack and adjust their drying strategies accordingly, ensuring optimal drying outcomes and protection of clothes based on the prevailing weather conditions.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

This chapter provides a complete summary of the intelligent drying rack system, highlighting its achievements in meeting the project's objectives and scope. Throughout the project, the primary goal was to develop a reliable and efficient system for assessing their conditions.

5.2 Conclusion of the project

In conclusion, the proposed design for a smart drying rack with a fuzzy logic system offers a comprehensive and intelligent solution for efficient and optimized drying processes. By integrating various components and technologies, this system enhances the user experience while providing protection for clothes and adapting to changing environmental conditions.

The sensing layers of the drying rack utilize sensors such as the DHT11, photocell sensor, and rain sensor. These sensors collect data on temperature, humidity, light intensity, and rain presence. This information is crucial for the fuzzy logic system to make informed decisions about the drying parameters and adapt to the prevailing weather conditions. The actuator, in the form of a servo motor, controls the movement of the drying rack, enabling position adjustments and rotation for even drying. The LED serves as a visual indicator, providing real-time feedback to the user about the status of the drying rack.

The network layer employs MQTT and websocket protocols for efficient and reliable communication between the drying rack and other devices or services. This allows for real-time data exchange and control commands, enabling seamless integration with a web server or other connected devices. The data processing layer incorporates a MySQL database to store the collected sensor data and relevant information. This facilitates historical data storage, analysis, and reporting, enabling users to gain insights and make informed decisions about the drying process.

The AI server in the data processing layer implements the fuzzy logic system, which utilizes linguistic variables, membership functions, and a set of rules to accurately predict weather conditions. By considering imprecise and uncertain data, the fuzzy logic system adapts to changing environmental factors and optimizes the drying process accordingly.

The application layer includes a user-friendly dashboard developed using Laravel, which allows users to monitor and control the drying rack. Additionally, the Telegram messaging platform sends real-time notification alerts to keep users informed about the drying process and any issues requiring attention.

Overall, this intelligent drying rack control system provides a robust and adaptive solution for efficient drying while considering the prevailing weather conditions. By incorporating fuzzy logic, users can optimize their drying strategies, ensure optimal drying outcomes, and protect clothes effectively. This project demonstrates the potential of integrating smart technologies, fuzzy logic, and connectivity to create innovative and intelligent solutions in everyday household appliances.

REFERENCES

[1] Chaihang, S., & Puengsungwan, S. (2021). Smart moving-spiral-clothesline for urban society. *ASEAN Journal of Science and Engineering*, *2*(3), 267-272.

[2] Kharisma, O. B., & Laumal, F. E. (2019, March). Propose design of smart clothesline with the tree diagram approach analysis and quality function deployment method for indonesia weather. In *Journal of Physics: Conference Series* (Vol. 1175, No. 1, p. 012125). IOP Publishing.

[3] Xing, X., Zhang, C., Gu, J., Zhang, Y., Lv, X., & Zhuo, Z. (2021, June). Intelligent drying rack system based on internet of things. In *Journal of Physics: Conference Series* (Vol. 1887, No. 1, p. 012002). IOP Publishing.

[4] Andrasto, T., & Joko, T. (2022). Simulation and design of smart clothesline using fuzzy for weather forecast. In *IOP Conference Series: Earth and Environmental Science* (Vol. 969, No. 1, p. 012058). IOP Publishing.

[5] Ma, Y., Chang, Y. C., Cui, Z., Rothwell, D., & Bykoriz, A. (2023, April). Blooming: Changing Laundry Habits and Opening Windows to Brighter Cities. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (pp. 1-8).

[6] Daund, S. R. V. (2022). INTERNET OF THINGS (IoT) BASED ROOFTOP FOR CLOTHING STAND.

[7] Yusoff, Z. M., Muhammad, Z., Abidin, A. F. Z., Dalila, K. N., Razali, N. F., Majid, M. A., & Hasan, K. K. (2018). Smart Clothline System Based on Internet of Thing (IoT). In *MATEC Web of Conferences* (Vol. 248, p. 02002). EDP Sciences.

[8] Lin, Y. H., Lee, Y. C., & Chang, C. P. (2020, November). Establishing an intelligent laundry drying rack using system innovation theory. In *IOP Conference Series: Earth and Environmental Science* (Vol. 603, No. 1, p. 012047). IOP Publishing.

[9] Xiao, Q., Chen, J., Ouyang, S., Shao, P., & Qin, F. (2014, December). Design and realization of the hardware for an intelligent solar drying system. In *2014 13th international conference on Control Automation Robotics & Vision (ICARCV)* (pp. 1234-1238). IEEE.

[10] Leng, Y., Qi, J., Liu, Y., & Zhu, F. (2020). Design of dry-type transformer temperature controller based on internet of things. *International Journal of Embedded Systems*, *12*(3), 380-392.

APPENDIX

VNC viewer

a. mqttsenderDHT11.py

```
import paho.mqtt.client as mqtt
from random import randrange, uniform
import Adafruit_DHT
import time

port = 1883
mqttBroker = "10.26.30.33"
client = mqtt.Client("ampaian")
client.username_pw_set("umpfk", "u4h%w1Tr12")
client.connect(mqttBroker,port)

# Set the pin connected to the DHT11 sensor
gpio = 4

# Set up the DHT11 sensor
sensor = Adafruit_DHT.DHT11

while True:

    # Read the temperature and humidity from the sensor
    humidity, temperature = Adafruit_DHT.read_retry(sensor, gpio)

    print(temperature)
    print("Just publish data to topic data to TEMPERATURE")
    print(humidity)
    print("Just publish data to topic data to HUMIDITY")

    client.publish("dryingrack/temp",temperature)
    client.publish("dryingrack/humid",humidity)

    time.sleep(5)
```

b. controlButton.py

```
import paho.mqtt.client as mqtt
from random import randrange, uniform
import Adafruit_DHT
import time
```

```
port = 1883
mqttBroker = "10.26.30.33"
client = mqtt.Client("ampaian")
client.username_pw_set("umpfk", "u4h%w1Tr12")
client.connect(mqttBroker,port)

# Set the pin connected to the DHT11 sensor
gpio = 4

# Set up the DHT11 sensor
sensor = Adafruit_DHT.DHT11

while True:

    # Read the temperature and humidity from the sensor
    humidity, temperature = Adafruit_DHT.read_retry(sensor, gpio)

    print(temperature)
    print("Just publish data to topic data to TEMPERATURE")
    print(humidity)
    print("Just publish data to topic data to HUMIDITY")

    client.publish("dryingrack/temp",temperature)
    client.publish("dryingrack/humid",humidity)

    time.sleep(5)
```

Arduino IDE

projectM3esp32.ino

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Chakepopo"; // Enter your WiFi name
const char* password = "mimiperi"; // Enter WiFi password
const char* mqttServer = "103.53.35.135"; // MQTT server IP
address
const int mqttPort = 1883;
const char* mqttUser = "umpfk";
const char* mqttPassword = "u4h%w1Tr12";
const char* topicPrefix = "dryingrack/"; // Prefix for MQTT
topics
```

```
const int rainPin = 34; // Pin for rain sensor (analog input)
const int lightPin = 39; // Pin for light sensor (analog input)

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(9600);
  pinMode(rainPin, INPUT);
  pinMode(lightPin, INPUT);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to the WiFi network");

  client.setServer(mqttServer, mqttPort);

  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");

    if (client.connect("ESP8266Client", mqttUser, mqttPassword))
{
      Serial.println("Connected to MQTT server");
    } else {
      Serial.print("Failed with state ");
      Serial.print(client.state());
      delay(1000);
    }
  }
}

void loop() {
  int rainLevel = analogRead(rainPin);
  rainLevel = 4095 - rainLevel; // Invert the rain sensor
reading
  float rainPercentage = (rainLevel / 4095.0) * 100; // Convert
to percentage
```

```
  int lightLevel = analogRead(lightPin);
  float lightPercentage = (lightLevel / 4095.0) * 100; //
Convert to percentage

  // Convert float values to strings
  String rainData = String(rainPercentage);
  String lightData = String(lightPercentage);

  client.loop();

  if (!client.connected()) {
    reconnect();
  }

  char rainTopic[30];
  char lightTopic[30];
  snprintf(rainTopic, sizeof(rainTopic), "%s%s", topicPrefix,
"rain");
  snprintf(lightTopic, sizeof(lightTopic), "%s%s", topicPrefix,
"light");

  //Serial.print("Rain: ");
  Serial.print(rainPercentage);
  Serial.print(", ");
  //Serial.print("Light: ");
  Serial.println(lightPercentage);
  //Serial.println("%");

  // Publish data as strings
  client.publish(rainTopic, rainData.c_str()); // Publish rain
data
  client.publish(lightTopic, lightData.c_str()); // Publish
light data

  delay(2000);
}

void reconnect() {
  while (!client.connected()) {
    Serial.println("Attempting MQTT connection...");
    if (client.connect("ESP8266Client", mqttUser, mqttPassword))
{
        Serial.println("Connected to MQTT server");
      } else {
        Serial.print("Failed with state ");
        Serial.print(client.state());
        delay(2000);
      }
    }
}
```

Visual Studio Code

a. subMQTT.py (subscribe)

```
#subscriber
import json
import mysql.connector
import paho.mqtt.client as mqtt
import requests
import pusher

pusher_client = pusher.Pusher(app_id='1', key=u'umpfkpusher',
secret=u'u%M15z2h%3A', cluster=u'mt1', ssl=False, host=u'10.26.30.32',
port=6001)

MQTT_Broker = "10.26.30.33"
MQTT_Port = 1883
Keep_Alive_Interval = 45
MQTT_Topic = "dryingrack/#"

# Connect to MySQL database
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="sensors"
)

# Create cursor for executing SQL queries
mycursor = mydb.cursor()

# Function to save distance data to DB
def save_temp(jsonData):
    value = float(json.loads(jsonData))
    sql = "INSERT INTO temp (value) VALUES (%s)"
    val = (value,)
    mycursor.execute(sql, val)
    mydb.commit()
    print("temp data stored")

# Function to save number data to DB
def save_humid(jsonData):
    value = float(json.loads(jsonData))
    #tele
    sql = "INSERT INTO humid (value) VALUES (%s)"
    val = (value,)
    mycursor.execute(sql, val)
    mydb.commit()
    print("humid data stored")

# Function to save number data to DB
def save_light(jsonData):
    value = float(json.loads(jsonData))
    #tele
    sql = "INSERT INTO light (value) VALUES (%s)"
    val = (value,)
    mycursor.execute(sql, val)
    mydb.commit()
    print("light data stored")

# Function to save number data to DB
def save_rain(jsonData):
    Value = float(json.loads(jsonData))
    sql = "INSERT INTO rain (id,Value) VALUES (%s,%s)"
    val = ('',Value)

    mycursor.execute(sql, val)
    mydb.commit()
    print("Rain Data Stored")

# Master function to handle incoming MQTT messages
def on_message(client, userdata, msg):
    print("MQTT Data Received...")
    print("MQTT Topic: " + msg.topic)
    print("Data: " + str(msg.payload))

    if msg.topic == "dryingrack/temp":
        save_temp(msg.payload)
    elif msg.topic == "dryingrack/humid":
        save_humid(msg.payload)
    elif msg.topic == "dryingrack/light":
        save_light(msg.payload)
    elif msg.topic == "dryingrack/rain":
        save_rain(msg.payload)

# Connect to MQTT broker
mqtt_client = mqtt.Client()
mqtt_client.username_pw_set("umpfk", "u4h%w1Tr12")
mqtt_client.on_message = on_message
mqtt_client.connect(MQTT_Broker, MQTT_Port, Keep_Alive_Interval)

# Subscribe to MQTT topic
mqtt_client.subscribe(MQTT_Topic, qos=1)

# Start the MQTT network loop
mqtt_client.loop_forever()
```

b. subAlarm.py (telegram)

```
#unntuk tele
import json
import mysql.connector
import paho.mqtt.client as mqtt
import requests
import pusher

pusher_client = pusher.Pusher(app_id='1', key=u'umpfkpusher',
secret=u'u%M15z2h%3A', cluster=u'mt1', ssl=False, host=u'10.26.30.32', port=6001)

MQTT_Broker = "10.26.30.33"
MQTT_Port = 1883
Keep_Alive_Interval = 45
MQTT_Topic = "dryingrack/#"

# Connect to MySQL database
mydb = mysql.connector.connect(
   host="localhost",
   user="root",
   password="",
   database="sensors"
)

# Create cursor for executing SQL queries
mycursor = mydb.cursor()

# Function to save number data to DB
def save_rain(jsonData):
   Value = float(json.loads(jsonData))
   print(Value)

   if (Value > 40.00):
      # if there is alarm trigger, data will go this flow
      try:
         pusher_client.trigger(u'DeviceAlarm', u'App\Events\AlarmStatus',
{u'DeviceID': 'Rain'})
      except:
         print("Rainy")
      TOKEN = "5894384751:AAGoRxOIYRzyi9TwNDb8A-W_fzs8YJxCd6w"
      chat_id = "-814857072"
      message = "System Alert : Alert Rain"
      url =
f"https://api.telegram.org/bot{TOKEN}/sendMessage?chat_id={chat_id}&text={mes
sage}"
      requests.get(url).json()
      sql = "INSERT INTO alarm (id,DeviceID) VALUES (%s,%s)"
      val = ("",'Rain')
      mycursor.execute(sql, val)
      mydb.commit()
      print('Alarm trigger')
   else:
   # send the sensor value
      sql = "INSERT INTO rain (id,Value) VALUES (%s,%s)"
      val = (",Value)

      mycursor.execute(sql, val)
      mydb.commit()
      print("Rain Data Stored")

# Master function to handle incoming MQTT messages
def on_message(client, userdata, msg):
   print("MQTT Data Received...")
   print("MQTT Topic: " + msg.topic)
   print("Data: " + str(msg.payload))

   if msg.topic == "dryingrack/rain":
      save_rain(msg.payload)

# Connect to MQTT broker
mqtt_client = mqtt.Client()
mqtt_client.username_pw_set("umpfk", "u4h%w1Tr12")
mqtt_client.on_message = on_message
mqtt_client.connect(MQTT_Broker, MQTT_Port, Keep_Alive_Interval)

# Subscribe to MQTT topic
mqtt_client.subscribe(MQTT_Topic, qos=1)

# Start the MQTT network loop
```

```
mqtt_client.loop_forever()
```

c. subfuzzy.py

```
#untuk AI
import paho.mqtt.client as mqtt
import json
import mysql.connector
import numpy as np
from datetime import datetime
from skfuzzy import control as ctrl
import skfuzzy as fuzz
import pickle
import joblib
import time
import pusher
import requests

pusher_client = pusher.Pusher(app_id='1', key=u'umpfkpusher',
secret=u'u%M15z2h%3A', cluster=u'mt1', ssl=False, host=u'10.26.30.32', port=6001)



# Function to save  to DB Table
def goToAPI():
    try:
        # api-endpoint
        URLtemperature = "http://127.0.0.1:8000/api/myTemperatureLink"
        rtemperature = requests.get(URLtemperature)
        datatemperature = rtemperature.json()
        myresult1 = datatemperature['blocks']
        myresult1 = np.float32(myresult1)

        URLrain = "http://127.0.0.1:8000/api/myRainLink"
        rrain = requests.get(URLrain)
        datarain = rrain.json()
        myresult2 = datarain['blocks']
        myresult2 = np.float32(myresult2)

        URLlight = "http://127.0.0.1:8000/api/myLightLink"
        rlight = requests.get(URLlight)
        datalight = rlight.json()
        myresult3 = datalight['blocks']
        myresult3 = np.float32(myresult3)

        objectRep = open("fuzzydar.picl", "rb") #nama kena sama (subfuzzy & temp)
        mynet = pickle.load(objectRep)

        # # # Pass inputs to the control system using Antecedent labels with Pythonic
API
        # # # Note: if you like passing many inputs all at once, use .inputs(dict_of_data)
        mynet.input['temperature'] = myresult1 # / 500
        mynet.input['rain'] = myresult2
        mynet.input['light'] = myresult3

        mynet.compute()

        if str(myresult2[0]) != '0' :
            # # # Print the result in a human-friendly form
            result = mynet.output['weather']
            print (str(result[0]))
            print (str(myresult1[0]))
            print (str(myresult2[0]))
            print (str(myresult3[0]))

            pusher_client.trigger(u'ToControlDar', u"App\Events\FuzzyEventDar",
{u'Pred0': str(myresult1[0]), u'Pred1': str(myresult2[0]), u'Pred2': str(myresult3[0]),
u'Pred3': str(result[0])})
            print('publish to socket')

    except:
        print('retry')

while 1:
    goToAPI()
    time.sleep(10)
```

d. env.

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:ZOv5hkw1o2Jv9LyKP7VeZBbIEqlIVmoekkr22Upn2Po=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
```

```
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=sensors
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=pusher
CACHE_DRIVER=file
FILESYSTEM_DRIVER=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=1
PUSHER_APP_KEY=umpfkpusher
PUSHER_APP_SECRET=u%M15z2h%3A
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

e. bootstrap.js

```
window._ = require('lodash');

try {
    require('bootstrap');
} catch (e) {}

/**
 * We'll load the axios HTTP library which allows us to easily issue requests
 * to our Laravel back-end. This library automatically handles sending the
 * CSRF token as a header based on the value of the "XSRF" token cookie.
 */

window.axios = require('axios');

window.axios.defaults.headers.common['X-Requested-With'] = 'XMLHttpRequest';

/**
 * Echo exposes an expressive API for subscribing to channels and listening
 * for events that are broadcast by Laravel. Echo and event broadcasting
 * allows your team to easily build robust real-time web applications.
 */

import Echo from 'laravel-echo';

window.Pusher = require('pusher-js');

window.Echo = new Echo({
    broadcaster: 'pusher',
    key: process.env.MIX_PUSHER_APP_KEY,
    cluster: process.env.MIX_PUSHER_APP_CLUSTER,
    wsHost: '10.26.30.32',
    wsPort: 6001,
    forceTLS: false
});
```

f. broadcasting.php

```
<?php

return [

    /*
    |--------------------------------------------------------------------------
    | Default Broadcaster
    |--------------------------------------------------------------------------
    |
    | This option controls the default broadcaster that will be used by the
    | framework when an event needs to be broadcast. You may set this to
    | any of the connections defined in the "connections" array below.
    |
    | Supported: "pusher", "ably", "redis", "log", "null"
    |
    */

    'default' => env('BROADCAST_DRIVER', 'null'),

    /*
    |--------------------------------------------------------------------------
    | Broadcast Connections
    |--------------------------------------------------------------------------
    |
    | Here you may define all of the broadcast connections that will be used
    | to broadcast events to other systems or over websockets. Samples of
    | each available type of connection are provided inside this array.
    |
    */

    'connections' => [

        'pusher' => [
            'driver' => 'pusher',
            'key' => env('PUSHER_APP_KEY'),
            'secret' => env('PUSHER_APP_SECRET'),
            'app_id' => env('PUSHER_APP_ID'),
            'options' => [
                'cluster' => env('PUSHER_APP_CLUSTER'),
                'encrypted' => true,
                'host' => '10.26.30.32',
                'port' => 6001,
                'scheme' => 'http'
            ],
        ],

        'ably' => [
            'driver' => 'ably',
            'key' => env('ABLY_KEY'),
        ],

        'redis' => [
            'driver' => 'redis',
            'connection' => 'default',
        ],

        'log' => [
            'driver' => 'log',
        ],

        'null' => [
            'driver' => 'null',
        ],

    ],

];
```

g. web.php

```
<?php

use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});

Route::group(['middleware' => 'auth'], function ()
{

    Route::get('/page1', function () {
        return view('dashboard');
    });

    Route::post('/InServo',function(){
        $message = 'on';
        event(new App\Events\AlarmStatus($message));
        return redirect('page1');
    });
    Route::post('/OutServo',function(){
        $message = 'off';
        event(new App\Events\AlarmStatus($message));
        return redirect('page1');
    });

    Route::get('route/temproute', 'App\Http\Controllers\dbcontroller@getTemp');
    Route::get('route/humidroute', 'App\Http\Controllers\dbcontroller@getHumid');
    Route::get('route/lightroute', 'App\Http\Controllers\dbcontroller@getLight');
    Route::get('route/rainroute', 'App\Http\Controllers\dbcontroller@getRain');
    Route::get('route/alarmroute', 'App\Http\Controllers\dbcontroller@getAlarm');

});
Auth::routes();

Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])-
>name('home');
```

h. dbcontroller.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

use App\Models\temp;
use App\Models\humid;
use App\Models\rain;
use App\Models\light;
use App\Models\alarm;

class dbcontroller extends Controller
{
   public function__construct()
   {
      $this->temp = new temp();
      $this->humid = new humid();
      $this->rain = new rain();
      $this->light = new light();
      $this->alarm = new alarm();
   }
   public function getTemp()
   {
      $blocks = DB::table('temp')
          ->select('Value')
          ->latest('datetime')
          ->limit(6)
          ->pluck('Value'); //give value
       $blocks2 = DB::table('temp')
          ->select('Value','datetime')
          ->latest('datetime')
          ->limit(6)
         ->pluck('datetime');
      return (compact('blocks','blocks2'));
   }

   public function getHumid()
   {
      $blocks = DB::table('humid')
          ->select('Value')
          ->latest('datetime')
          ->limit(6)
          ->pluck('Value'); //give value
      $blocks2 = DB::table('humid')
          ->select('Value','datetime')
          ->latest('datetime')
          ->limit(6)
          ->pluck('datetime');
      return (compact('blocks','blocks2'));
   }

   public function getLight()
   {
      $blocks = DB::table('light')
          ->select('Value')
          ->latest('datetime')
          ->limit(6)
          ->pluck('Value'); //give value
       $blocks2 = DB::table('light')
          ->select('Value','datetime')
          ->latest('datetime')
          ->limit(6)
          ->pluck('datetime');
       return (compact('blocks','blocks2'));
   }

   public function getRain()
   {
      $blocks = DB::table('rain')
          ->select('Value')
          ->latest('datetime')
          ->limit(6)
          ->pluck('Value'); //give value
       $blocks2 = DB::table('rain')
          ->select('Value','datetime')
          ->latest('datetime')
          ->limit(6)
          ->pluck('datetime');
       return (compact('blocks','blocks2'));
   }

   public function getAlarm()
   {
      $blocks = DB::table('alarm')
          ->select('DeviceID')
          ->latest('datetime')
          ->limit(5)
          ->pluck('DeviceID'); //give value
       $blocks2 = DB::table('alarm')
          ->select('DeviceID','datetime')
          ->latest('datetime')
          ->limit(5)
          ->pluck('datetime');
       return (compact('blocks','blocks2'));
   }

}
```

i. app.blade.php

```
<!doctype html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <title>{{ config('app.name', 'Smart Drying Rack') }}</title>

    <!-- Scripts -->
    <script src="{{ asset('js/app.js') }}"></script>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

    <script src="https://unpkg.com/leaflet/dist/leaflet-src.js"></script>
    <script src="https://unpkg.com/esri-leaflet"></script>
    <script src="https://unpkg.com/esri-leaflet-geocoder"></script>

    <!-- routing -->
   <link rel="stylesheet" href="https://unpkg.com/leaflet@1.2.0/dist/leaflet.css" />
   <link rel="stylesheet" href="https://unpkg.com/leaflet-routing-
machine@latest/dist/leaflet-routing-machine.css" />

   <!-- <script src="https://unpkg.com/leaflet@1.2.0/dist/leaflet.js"></script> -->
   <script src="https://unpkg.com/leaflet-routing-machine@latest/dist/leaflet-routing-
machine.js"></script>

    <!-- Fonts -->
    <link rel="dns-prefetch" href="//fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css?family=Nunito" rel="stylesheet">

    <!-- Styles -->
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQS
Ze" crossorigin="anonymous"></script>
    <link href="{{ asset('css/sidebars.css') }}" rel="stylesheet">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9
```

```
+nJOZ" crossorigin="anonymous">

    <!-- Scroll Bar  -->
    <link href="{{ asset('css/sidebars.css') }}" rel="stylesheet">

</head>
<body>
    <style>
        body
        {
            background-color: #FAF2DA;
        }

        .h1
        {
            color: #000000;
            font-weight: bold;
            text-align: center;
        }

        .grid-container
        {
            display: grid;
            grid-gap: 10px;
            grid-template-columns: repeat(12, minmax(0, 1fr));
            /* background-color: #BEBDB8; */
        }

        @media only screen and (max-width: 865px)
        {
            .grid-container {
                grid-template-columns: 1fr;
            }
        }

        .grid-item
        {
            width: 100%;
            height: 100%;
        }

        .card
        {
            background-color: white;
            color: black;
        }

        .card-header
        {
```

```
   background-color: #011B10;
   color: #E4DEAE;
   font-weight: bold;
   text-align: center;
}

.card-body
{
   font-size: 20px;
   text-align: center;
   background-color: #B7BF96;
   color: white;
}

.mycard1
{
   grid-row-start: 1;
   grid-row-end: 3;
   grid-column-start: 1;
   grid-column-end: 4;
   text-align: center;
}

.mycard2
{
   grid-row-start: 1;
   grid-row-end: 3;
   grid-column-start: 4;
   grid-column-end: 7;
   text-align: center;
}

.mycard3
{
   grid-row-start: 1;
   grid-row-end: 3;
   grid-column-start: 7;
   grid-column-end: 10;
   text-align: center;
}

.mycard4
{
   grid-row-start: 1;
   grid-row-end: 3;
   grid-column-start: 10;
   grid-column-end: 13;
   text-align: center;
}
```

```
.fuzzy
{
   grid-row-start: 4;
   grid-row-end: 9;
   grid-column-start: 1;
   grid-column-end: 5;
   text-align: center;
}

.mycard5
{
   grid-row-start: 4;
   grid-row-end 8;
   grid-column-start: 5;
   grid-column-end: 9;
   text-align: center;
}

.mycard6
{
   grid-row-start: 4;
   grid-row-end: 8;
   grid-column-start: 9;
   grid-column-end: 13;
   text-align: center;
}

.mycard7
{
   grid-row-start: 8;
   grid-row-end: 12;
   grid-column-start: 5;
   grid-column-end: 9;
   text-align: center;
}

.mycard8
{
   grid-row-start: 8;
   grid-row-end: 12;
   grid-column-start: 9;
   grid-column-end: 13;
   text-align: center;
}

.ServoMotor
{
   grid-row-start: 9;
```

```
          grid-row-end: 12;
          grid-column-start: 1;
          grid-column-end: 5;
          text-align: center;
        }

        .itemAlarm
        {
          grid-row-start: 13;
          grid-row-end: 16;
          grid-column-start: 1;
          grid-column-end: 13;
          text-align: center;
        }

    </style>

    <div id="app">
        <nav class="navbar navbar-expand-md navbar-dark" style="background-color:
#011B10; box-shadow: 0 2px 4px rgba(0, 0, 0, 0.4);">
            <div class="container">
                <a class="navbar-brand" href="#">
                    <img src="ampaian.png" alt="Logo" width="40" height="40" class="d-
inline-block align-text-top">
                    <a class="nav-link active" aria-current="page" href="/page1"
style="font-size: 25px; color: #E4DEAE;"><b>DRYING RACK</b></a>
                </a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="{{ __('Toggle navigation') }}">
                    <span class="navbar-toggler-icon"></span>
                </button>

                <div class="collapse navbar-collapse" id="navbarSupportedContent">
                    <!-- Left Side Of Navbar -->
                    <ul class="navbar-nav me-auto">

                    </ul>

                    <!-- Right Side Of Navbar -->
                    <ul class="navbar-nav ms-auto">
                        <!-- Authentication Links -->
                        @guest
                            @if (Route::has('login'))
                                <li class="nav-item">
                                    <a class="nav-link" href="{{ route('login') }}">{{ __('Login')
}}</a>
                                </li>
                            @endif
                            @if (Route::has('register'))
                                <li class="nav-item">
                                    <a class="nav-link" href="{{ route('register') }}">{{
    __('Register') }}</a>
                                </li>
                            @endif
                        @else
                            <li class="nav-item dropdown">
                                <a id="navbarDropdown" class="nav-link dropdown-toggle"
href="#" role="button" data-bs-toggle="dropdown" aria-haspopup="true" aria-
expanded="false" v-pre>
                                    {{ Auth::user()->name }}
                                </a>

                                <div class="dropdown-menu dropdown-menu-end" aria-
labelledby="navbarDropdown">
                                    <a class="dropdown-item" href="{{ route('logout') }}"
                                       onclick="event.preventDefault();
                                                 document.getElementById('logout-
form').submit();">
                                {{ __('Logout') }}
                                    </a>

                                    <form id="logout-form" action="{{ route('logout') }}"
method="POST" class="d-none">
                                @csrf
                                    </form>
                                </div>
                            </li>
                        @endguest
                    </ul>
                </div>
          </div>
        </nav>


        <main class="py-4">
          @yield('content')
        </main>
    </div>
</body>
</html>
```

j. dashboard.blade.php

```
@extends('layouts.app')
@section('content')
<div class="h1">INTELLIGENT DRYING RACK CONTROL SYSTEM WITH
WEATHER CONDITION PREDICTION USING FUZZY LOGIC</div>
<div class="container-fluid">
    <div class="dashboard">
        <div class="grid-container">

            <!--value sensors-->
            <div class="grid-item mycard1"> <!--value g1-->
                <div class="card">
                    <div class="card-header"><img src="temp.png" alt="Logo" width="30"
height="30" class="d-inline-block align-text-top">
                        Temperature °C<span class="badge badge-info float-right"></span>
                    </div>
                    <div class="card-body">Value = <span class="badge badge-info float-
right" id="tempvalue"></span></div>
                </div>
            </div>

            <div class="grid-item mycard2"> <!--value g2-->
                <div class="card">
                    <div class="card-header"><img src="humid.png" alt="Logo"
width="30" height="30" class="d-inline-block align-text-top">
                        Humidity %<span class="badge badge-info float-right"></span>
                    </div>
                    <div class="card-body">Value = <span class="badge badge-info float-
right" id="humidvalue"></span></div>
                </div>
            </div>

            <div class="grid-item mycard3"> <!--value g3-->
                <div class="card">
                    <div class="card-header"><img src="sun.png" alt="Logo" width="30"
height="30" class="d-inline-block align-text-top">
                        Light Intensity %<span class="badge badge-info float-right"></span>
                    </div>
                    <div class="card-body">Value = <span class="badge badge-info float-
right" id="lightvalue"></span></div>
                </div>
            </div>

            <div class="grid-item mycard4"> <!--value g4-->
            <div class="card">
               <div class="card-header"><img src="rain.png" alt="Logo" width="30"
height="30" class="d-inline-block align-text-top">
                  Rain %<span class="badge badge-info float-right"></span>
               </div>
               <div class="card-body">Value = <span class="badge badge-info float-
right" id="rainvalue"></span></div>
            </div>
         </div>

         <!--fuzzy logic-->
         <div class="grid-item fuzzy">
            <div class="card">
               <div class="card-header">Powered by fuzzy<span class="badge badge-
info float-right"></span></div>
               <div class="card-body">
                  <!-- <div class="card">
                     <div class="card-header">value<span class="badge badge-info
float-right"></span></div>
                     <div class="card-bodyV"><span class="badge badge-info float-
right" id="Pred3"></span></div>
                  </div>  -->
                  <br>
                  <div class="card">
                     <div class="card-header">status<span class="badge badge-info
float-right"></span></div>
                     <div class="card-bodyV"><span class="badge badge-info float-
right" id="fuzzyresult"></span></div>
                  </div>
                  <br>
                  <br>
                  <br>
                  <br>
               </div>
            </div>
         </div>

         <!--graph sensors-->
         <div class="grid-item mycard5"> <!--graph g1-->
            <div class="card">
               <div class="card-header">Graph for Temperature<span class="badge
badge-info float-right"></span></div>
               <div class="card-body d-flex justify-content-center allign-items-center"
>
                  <canvas id="myfirstgraph" class="chartjs" style="position: relative;
height:40vh;width:60vw"></canvas>
                  <!--script under here -->
               </div>
            </div>
```

```
          </div>

          <div class="grid-item mycard6"> <!--graph g2-->
             <div class="card">
                <div class="card-header">Graph for Humidity<span class="badge
badge-info float-right"></span></div>
                <div class="card-body d-flex justify-content-center allign-items-center"
>
                   <canvas id="mysecondgraph" class="chartjs" style="position: relative;
height:40vh;width:60vw"></canvas>
                   <!--script under here -->
                </div>
             </div>
          </div>

          <div class="grid-item mycard7"> <!--graph g3-->
             <div class="card">

                <div class="card-header">Graph for Light Level<span class="badge
badge-info float-right"></span></div>
                <div class="card-body d-flex justify-content-center allign-items-center"
>
                   <canvas id="mythirdgraph" class="chartjs" style="position: relative;
height:40vh;width:60vw"></canvas>
                   <!--script under here -->
                </div>
             </div>
          </div>

          <div class="grid-item mycard8"> <!--graph g4-->
             <div class="card">

                <div class="card-header">Graph for Rain<span class="badge badge-info
float-right"></span></div>
                <div class="card-body d-flex justify-content-center allign-items-center"
>
                   <canvas id="myfourthgraph" class="chartjs" style="position: relative;
height:40vh;width:60vw"></canvas>
                   <!--script under here -->
                </div>
             </div>
          </div>

          <!--Card servo motor -->
          <div class="grid-item ServoMotor">
             <div class="card">
                <!--ON/OFF switch SM-->
                <div class="card-header">Status for Drying Rack: <span class="badge
badge-info float-right" id="myDeviceStat1"></span></div>
```

```
                <div class="card-body d-flex justify-content-center align-items-center">
                    <form action="/InServo" method="post">
                        <input type="submit" value="RackIN" class="btn btn-success">
                        {{csrf_field()}}
                    </form>
                </div>
                <div class="card-body d-flex justify-content-center align-items-center">
                    <form action="/OutServo" method="post">
                        <input type="submit" value="RackOUT" class="btn btn-danger">
                        {{csrf_field()}}
                    </form>
                </div>
            </div>
        </div>

        <!--Card Alert -->
        <div class="grid-item itemAlarm">
            <div class="card">
                <div class="card-header">Event<span class="badge badge-info float-
right" id="myAlert"></span>
                    <div class="card-body d-flex justify-content-center align-items-center"
>
                        <div class="card" style="width: 100%; height: auto;">
                            <div class="card-header" style= "height: 50px;">
                                <table id="myTablehead" style="width: 100%">
                                    <tr>
                                        <td><img src="alarm.png" alt="Logo" width="30"
height="30" class="d-inline-block align-text-top">Alarm Status</td>
                                        <td style="width: 80%">Date</td>
                                    </tr>
                                </table>
                                <br>

                            </div>

                            <div class="card-bodyV" style="overflow: auto; max-height:
200px;"> <!-- Adjust max-height as needed -->
                                <table id="myTable" style="width: 100%">
                                    <tr>
                                        <td> </td>
                                        <td> </td>
                                    </tr>
                                </table>
                                <br>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
```

```
         </div>
      </div>
   </div>
</div>

<script>
   setInterval(ajaxcall,5000);
   function  ajaxcall(){

   $.getJSON('/route/temproute', function(blocksall){
      var datas = blocksall.blocks.map(Number);
      datas = datas.reverse();
      console.log(datas)

      var datasx = blocksall.blocks2.map(String);
      datasx = datasx.reverse();
      console.log(datasx)
      var temperatureValue = datas[datas.length - 1].toFixed(2);
      document.getElementById("tempvalue").innerHTML = temperatureValue + "
°C";
      document.getElementById("tempvalue").style.color="green";

      var chart = new Chart(document.getElementById('myfirstgraph'), //mesti sama
dgn id
      {
         type: 'line',
         data:
         {
            labels : datasx,
            datasets:
            [{
               label: 'Temperature, °C',
               data: datas,
               fill: false,
               borderColor: '#FA255E',
               backgroundcolor: 'rgb(255, 255, 0)',
               tension: 0.1
            }]
         },
         options: {
            animation: {
               duration: 100,
               easing: 'easeOutBounce' //movement
            },
            scales: {
               yAxes: [{
                  display:  true,
                  stacked: true,
                  ticks: {
```

```
                min: 0, //minimum value
                max: 50 //maximum value
              }
            }]
          }
        }
      });
  });
  }

  $.getJSON('/route/humidroute', function(blocksall){
    var datas = blocksall.blocks.map(Number);
    datas = datas.reverse();
    console.log(datas)

    var datasx = blocksall.blocks2.map(String);
    datasx = datasx.reverse();
    console.log(datasx)
    var humidValue = datas[datas.length - 1].toFixed(2);
    document.getElementById("humidvalue").innerHTML = humidValue + " %";
    document.getElementById("humidvalue").style.color="green";

    var chart = new Chart(document.getElementById('mysecondgraph'), //mesti
sama dgn id
    {
      type: 'line',
      data:
      {
        labels : datasx,
        datasets:
        [{
          label: 'Humidity, %',
          data: datas,
          fill: false,
          borderColor: '#FA255E',
          backgroundcolor: 'rgb(255, 255, 0)',
          tension: 0.1
        }]
      },
      options: {
        animation: {
          duration: 100,
          easing: 'easeOutBounce' //movement
        },
        scales: {
          yAxes: [{
            display:  true,
            stacked: true,
            ticks: {
              min: 0, //minimum value
              max: 100 //maximum value
            }
          }]
        }
      }
    });
  });

  $.getJSON('/route/lightroute', function(blocksall){
    var datas = blocksall.blocks.map(Number);
    datas = datas.reverse();
    console.log(datas)

    var datasx = blocksall.blocks2.map(String);
    datasx = datasx.reverse();
    console.log(datasx)
    var lightValue = datas[datas.length - 1].toFixed(2);
    document.getElementById("lightvalue").innerHTML = lightValue + " %";
    document.getElementById("lightvalue").style.color="green";

    var chart = new Chart(document.getElementById('mythirdgraph'), //mesti sama
dgn id
    {
      type: 'bar',
      data:
      {
        labels : datasx,
        datasets:
        [{
          label: 'Light intensity, %',
          data: datas,
          fill: false,
          borderColor: '#FA255E',
          backgroundcolor: 'rgb(255, 255, 0)', //'#FA255E',
          tension: 0.1
        }]
      },
      options: {
        animation: {
          duration: 100,
          easing: 'easeOutBounce' //movement
        },
        scales: {
          yAxes: [{
            display:  true,
            stacked: true,
            ticks: {
              min: 0, //minimum value
              max: 100 //maximum value
            }
          }]
        }
      }
    });

  });

  $.getJSON('/route/rainroute', function(blocksall){
    var datas = blocksall.blocks.map(Number);
    datas = datas.reverse();
    console.log(datas)

    var datasx = blocksall.blocks2.map(String);
    datasx = datasx.reverse();
    console.log(datasx)
    var rainValue = datas[datas.length - 1].toFixed(2);
    document.getElementById("rainvalue").innerHTML = rainValue + " %";
    document.getElementById("rainvalue").style.color="green";

    var chart = new Chart(document.getElementById('myfourthgraph'), //mesti sama
dgn id
    {
      type: 'line',
      data:
      {
        labels : datasx,
        datasets:
        [{
          label: 'rain level, %',
          data: datas,
          fill: false,
          borderColor: '#FA255E',
          backgroundcolor: '#FA255E',
          tension: 0.1
        }]
      },
      options: {
        animation: {
          duration: 100,
          easing: 'easeOutBounce' //movement
        },
        scales: {
          yAxes: [{
            display:  true,
            stacked: true,
            ticks: {
              min: 0, //minimum value
                    max: 100 //maximum value
                }
            }]
        }
    }
  });

});


//button
Echo.channel('AppStatus1').listen('AlarmStatus', (e) => {
   //console.log(e)
   if (e['status'] ==1) {
      document.getElementById("myDeviceStat1").innerHTML = 'on';
      document.getElementById("myDeviceStat1").style.color = "green";
   } else{
      document.getElementById("myDeviceStat1").innerHTML = 'off';
      document.getElementById("myDeviceStat1").style.color = "blue";
   }
}); //AppStatus1,DeviceStatus=VNC , myDeviceStat1=dkt table button

Echo.channel('ToControlDar').listen('FuzzyEventDar', (e) =>
{
   console.log(e['Pred3'])
   if (e['Pred3'] > 0 && e['Pred3'] <= 67.5) {
      document.getElementById("fuzzyresult").innerHTML = 'Sunny';
      document.getElementById("fuzzyresult").style.color = "green";
   }
   else if (e['Pred3'] > 67.5 && e['Pred3'] <= 80) {
      document.getElementById("fuzzyresult").innerHTML = 'Cloudy';
      document.getElementById("fuzzyresult").style.color = "blue";
   }
   else if (e['Pred3'] > 80) {
      document.getElementById("fuzzyresult").innerHTML = 'Rainy';
      document.getElementById("fuzzyresult").style.color = "red";
   }
});

//alarm
$.getJSON('/route/alarmroute', function(blocksall){
   //console.log(blocksall.blocks)
   var datas = blocksall.blocks.map(String);
   datas = datas.reverse();
   // console.log(datas)
   var datasx = blocksall.blocks2.map(String);
   datasx = datasx.reverse();
```

```
        var table = document.getElementById("myTable");

        for (let step = 0; step < datas.length; step++)
        {
            // console.log(datas[step]);
            var row = table.insertRow(0);
            var cell1 = row.insertCell(0);
            var cell2 = row.insertCell(1);
            cell1.innerHTML = datas[step];
            cell2.innerHTML = datasx[step];
        }
    });

    Echo.channel('DeviceAlarm').listen('AlarmStatus', (e) =>
    {
        var table = document.getElementById("myTable");
        var row = table.insertRow(0);
        var cell1 = row.insertCell(0);
        var cell2 = row.insertCell(1);
        cell1.innerHTML = e['DeviceID'];
        cell2.innerHTML = new Date().toLocaleString('en-CA', {hour12: false,});
        document.getElementById("myAlert").innerHTML = 'Alarm Trigger!!';
        document.getElementById("myAlert" ).style.color="red";

        // This is for timecheck
        first = new Date(datasx[datasx.length - 1]); // Get the first date epoch object
        // document.write((first.getTime())/1000); // get the actual epoch values
        second = new Date(); // Get the first date epoch object
        // document.write((second.getTime())/1000); // get the actual epoch values
        diff= second.getTime() - first.getTime() ;
        // console.log(diff);
        if (diff < 10000) {
            document.getElementById("myAlert").innerHTML = "online";
            document.getElementById( "myAlert" ).style.color="green";
            document.getElementById( "myAlert" ).style.backgroundColor = "white";
        } else{
            document.getElementById("myAlert").innerHTML = "offline";
            document.getElementById( "myAlert" ).style.color="red";
            document.getElementById( "myAlert" ).style.backgroundColor = "white";
        }
    });

</script>
@endsection
```

k. api.php

```
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});
Route::get('myTemperatureLink',
'App\Http\Controllers\APIcontroller@APIlisttemp');
Route::get('myRainLink', 'App\Http\Controllers\APIcontroller@APIlistrain');
Route::get('myLightLink', 'App\Http\Controllers\APIcontroller@APIlistlight');
```

l. apicontroller.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

use App\Models\temp;  #refer to table in dbase
use App\Models\rain;
use App\Models\light;


class APIcontroller extends Controller
{
    public function __construct()
    {
    $this->temp = new temp();
    $this->rain = new rain();
    $this->light = new light();
    }

    public function APIlisttemp()
    {
       $blocks = DB::table('temp')
       ->select('Value')
       ->latest('datetime')
       ->limit(1)
       ->pluck('Value'); //give value
       return (compact('blocks'));
    }

    public function APIlistrain()
    {
       $blocks = DB::table('rain')
       ->select('Value')
       ->latest('datetime')
       ->limit(1)
       ->pluck('Value'); //give value
       return (compact('blocks'));
    }

    public function APIlistlight()
    {
       $blocks = DB::table('light')
       ->select('Value')
       ->latest('datetime')
       ->limit(1)
       ->pluck('Value'); //give value
       return (compact('blocks'));
    }

}
```

m. alarmstatus.php

```
<?php

namespace App\Events;

use Illuminate\Broadcasting\Channel;
use Illuminate\Broadcasting\InteractsWithSockets;
use Illuminate\Broadcasting\PresenceChannel;
use Illuminate\Broadcasting\PrivateChannel;
use Illuminate\Contracts\Broadcasting\ShouldBroadcast;
use Illuminate\Foundation\Events\Dispatchable;
use Illuminate\Queue\SerializesModels;

class alarmstatus implements ShouldBroadcast
{
   use Dispatchable, InteractsWithSockets, SerializesModels;

   public $message;
   /**
    * Create a new event instance.
    *
    * @return void
    */
   public function_construct($message)
   {
      $this->message = $message;
   }

   /**
    * Get the channels the event should broadcast on.
    *
    * @return \Illuminate\Broadcasting\Channel|array
    */
   public function broadcastOn()
   {
      return new Channel('ToControlDar');
   }
}
```

n. alarm.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class alarm extends Model
{
   use HasFactory;
}
```

o. temp.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class temp extends Model
{
   use HasFactory;
}
```

p. humid.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class humid extends Model
{
   use HasFactory;
}
```

q. rain.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
```

```
class rain extends Model
{
    use HasFactory;
}
```

r. light.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class light extends Model
{
    use HasFactory;
}
```

Spyder

a. fuzzytrain.py

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 11 22:22:10 2023

@author: darwisyah
"""

import numpy as np
from skfuzzy import control as ctrl
import skfuzzy as fuzz
import pickle

# Define the fuzzy sets for the inputs
temperature = ctrl.Antecedent(np.arange(0, 41, 1), 'temperature') #range = 0-40
rain = ctrl.Antecedent(np.arange(0, 101, 1), 'rain') #range = 0-2000
```

```
light = ctrl.Antecedent(np.arange(0, 101, 1), 'light') #range = 0-4095

# Define the fuzzy sets for the output
weather = ctrl.Consequent(np.arange(0, 101, 1), 'weather')


temperature['cold'] = fuzz.trimf(temperature.universe, [0, 10, 16])
temperature['mild'] = fuzz.trimf(temperature.universe, [15, 20, 33])
temperature['hot'] = fuzz.trimf(temperature.universe, [32, 36, 40])

rain['dry'] = fuzz.trimf(rain.universe, [2, 20, 30])
rain['lightrain'] = fuzz.trimf(rain.universe, [25, 35, 45])
rain['heavyrain'] = fuzz.trimf(rain.universe, [40, 60, 100])

light['dark'] = fuzz.trimf(light.universe, [0, 20, 30])
light['moderate'] = fuzz.trimf(light.universe, [25, 45, 50])
light['bright'] = fuzz.trimf(light.universe, [48, 80, 100])

weather['sunny'] = fuzz.trimf(weather.universe, [0, 33.75, 67.5]) #0-25 #-5,0
weather['cloudy'] = fuzz.trimf(weather.universe, [66.5, 71.25, 80]) #25-50 #0,
weather['rainy'] = fuzz.trimf(weather.universe, [79, 87.5, 100]) #50-100 #0,-5

#shaded region under graph
#handling.view(sim=conveyor)
temperature.view()
rain.view()
light.view()
weather.view()

# Define the rule base using the 'rule' method
rule1 = ctrl.Rule(temperature['cold'] & rain['dry'] & light['dark'], weather['cloudy'])
rule2 = ctrl.Rule(temperature['cold'] & rain['dry'] & light['moderate'],
weather['sunny'])
rule3 = ctrl.Rule(temperature['cold'] & rain['dry'] & light['bright'], weather['sunny'])
rule4 = ctrl.Rule(temperature['cold'] & rain['lightrain'] & light['dark'],
weather['rainy'])
rule5 = ctrl.Rule(temperature['cold'] & rain['lightrain'] & light['moderate'],
weather['rainy'])
rule6 = ctrl.Rule(temperature['cold'] & rain['lightrain'] & light['bright'],
weather['rainy'])
rule7 = ctrl.Rule(temperature['cold'] & rain['heavyrain'] & light['dark'],
weather['rainy'])
rule8 = ctrl.Rule(temperature['cold'] & rain['heavyrain'] & light['moderate'],
weather['rainy'])
rule9 = ctrl.Rule(temperature['cold'] & rain['heavyrain'] & light['bright'],
weather['rainy'])

rule10 = ctrl.Rule(temperature['mild'] & rain['dry'] & light['dark'], weather['cloudy'])
rule11 = ctrl.Rule(temperature['mild'] & rain['dry'] & light['moderate'],
weather['sunny'])
rule12 = ctrl.Rule(temperature['mild'] & rain['dry'] & light['bright'], weather['sunny'])
rule13 = ctrl.Rule(temperature['mild'] & rain['lightrain'] & light['dark'],
weather['rainy'])
rule14 = ctrl.Rule(temperature['mild'] & rain['lightrain'] & light['moderate'],
weather['rainy'])
rule15 = ctrl.Rule(temperature['mild'] & rain['lightrain'] & light['bright'],
weather['rainy'])
rule16 = ctrl.Rule(temperature['mild'] & rain['heavyrain'] & light['dark'],
weather['rainy'])
rule17 = ctrl.Rule(temperature['mild'] & rain['heavyrain'] & light['moderate'],
weather['rainy'])
rule18 = ctrl.Rule(temperature['mild'] & rain['heavyrain'] & light['bright'],
weather['rainy'])

rule19 = ctrl.Rule(temperature['hot'] & rain['dry'] & light['dark'], weather['cloudy'])
rule20 = ctrl.Rule(temperature['hot'] & rain['dry'] & light['moderate'],
weather['sunny'])
rule21 = ctrl.Rule(temperature['hot'] & rain['dry'] & light['bright'], weather['sunny'])
rule22 = ctrl.Rule(temperature['hot'] & rain['lightrain'] & light['dark'],
weather['rainy'])
rule23 = ctrl.Rule(temperature['hot'] & rain['lightrain'] & light['moderate'],
weather['rainy'])
rule24 = ctrl.Rule(temperature['hot'] & rain['lightrain'] & light['bright'],
weather['rainy'])
rule25 = ctrl.Rule(temperature['hot'] & rain['heavyrain'] & light['dark'],
weather['rainy'])
rule26 = ctrl.Rule(temperature['hot'] & rain['heavyrain'] & light['moderate'],
weather['rainy'])
rule27 = ctrl.Rule(temperature['hot'] & rain['heavyrain'] & light['bright'],
weather['rainy'])


# Create the control system using the rules
dryingRack_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7,
rule8, rule9, rule10,
                              rule11, rule12, rule13, rule14, rule15, rule16, rule17, rule18,
rule19, rule20,
                              rule21, rule22, rule23, rule24, rule25, rule26, rule27])

# Create a control system simulation
dryingRack = ctrl.ControlSystemSimulation(dryingRack_ctrl)

# # Pass inputs to the control system using Antecedent labels with Pythonic API
# # Note: if you like passing many inputs all at once, use .inputs(dict_of_data)
# value tu utk kluakan result normal
dryingRack.input['temperature'] = 27
dryingRack.input['rain'] = 1
dryingRack.input['light'] = 40
```

```
# # Crunch the numbers
dryingRack.compute()

weather.view(sim=dryingRack)

# # Print the result in a human-friendly form
print(dryingRack.output['weather'])


file = open('fuzzydar.picl', 'wb')
pickle.dump(dryingRack,file)
file.close()
```