AI-ENHANCED WEATHER MONITORING SYSTEM USING ESP8266 AND GOOGLE SHEETS

A PROJECT REPORT

Submitted by

KISHORE G (Reg. No. 22MT029)

K AAHAMAN(Reg. No.22MT025)

This project report is submitted as part of the requirements for the completion of

CORE COURSE PROJECT OF THE CURRICULUM

DEPARTMENT OF MECHATRONICS ENGINEERING

CHENNAI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University, Chennai)

SARATHY NAGAR, KUNDRATHUR – 600 069

TAMIL NADU, INDIA

OCTOBER 2024

CHENNAI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University, Chennai)

SARATHY NAGAR, KUNDRATHUR – 600 069

BONAFIDE CERTIFICATE

Certified that this project report entitled "**AI-Enhanced Weather Monitoring System Using ESP8266 and Google Sheets"** is the bonafide work of **Mr. KISHORE G(Reg. No. :22MT029)** and **Mr. K AAHAMAN (Reg.No. :22MT025)** who carried out the research under our supervision.

Dr. S. CHANDRAVADHANA	**Dr. S. CHANDRAVADHANA**
Head of the Department	SUPERVISOR
Professor,	Professor,
Department of Mechatronics Engineering,	Department of Mechatronics Engineering,
Chennai Institute of Technology,	Chennai Institute of Technology,
Kundrathur – 600 069.	Kundrathur – 600 069.

Certified that the student have attend of viva voce during during the exam held on ……………………….

INTERNAL EXAMINAR **EXTERNAL EXAMINAR**

ABSTRACT

This project focuses on the development of a real-time Weather Monitoring System using the ESP8266 microcontroller and various environmental sensors, integrated with cloud-based storage and AI-based predictive analysis. The system continuously monitors key weather parameters such as temperature, humidity, atmospheric pressure, rainfall, and light intensity. The ESP8266 processes data from sensors, transmitting it wirelessly to Google Sheets for real-time storage and visualization using HTTP protocols. Additionally, a Google Apps Script was developed to automate data handling.

For predictive analysis, AI techniques were employed, specifically Long Short-Term Memory (LSTM) models trained on historical sensor data. The AI model forecasts short-term weather conditions, enhancing the system's capability to predict future environmental changes. A web interface was developed to visualize both real-time sensor data and AI predictions, allowing users to monitor and analyze current conditions and trends effectively.

The project demonstrates the potential of combining IoT and AI technologies for applications in sectors like agriculture, disaster management, and environmental monitoring. Testing showed reliable data collection, accurate predictions, and effective real-time monitoring, making this system a scalable and adaptable solution for future use.

ACKNOWLEDGEMENT

It is our first and foremost duty to express our heartily gratitude and allegiance to our beloved chairman **Sri. P. SRIRAM** for providing us with a good working environment, facilities and opportunity to carry out the research work with full freedom in and out of the campus.

We would like to express our deepest gratitude to our Principal **Dr. A. RAMESH, Ph.D.,** for being our inspiration throughout the entire period of course.

It greatly appreciate and convey my heartfelt thanks to **Dr. S. CHANDRAVADHANA M.E, Ph.D.,** Professor & Head – Mechatronics Engieering, for their continuous support and valuable suggestions throughout the research period.

It gives us immense pleasure to express my deep sense of gratitude to our research supervisor, **Dr. S. CHANDRAVADHANA M.E, Ph.D.,** Assistant Professor, Faculty of Mechatronics Engineering for their valuable guidance, constant inspiration, encouragement, and motivation.

We extend our gratitude and sincere thanks to all the faculty and supporting staff of **Department of Mechatronics Engineering** for their valuable suggestions and support during the course of our research work.

We thank the **ALMIGHTY** to have provided such great personalities in our life and chance to join their research group.

KISHORE G

K AAHAMAN

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Real-time environmental monitoring is now essential for industries such as agriculture, smart cities, and disaster management. Traditionally, weather data was collected manually or through standalone stations with limited real-time access and no predictive capabilities. The integration of **Internet of Things (IoT)** and **Artificial Intelligence (AI)** has transformed weather monitoring, making it more efficient and scalable. IoT systems use wireless sensors to collect data like temperature, humidity, and rainfall, enabling real-time transmission and remote access via the internet.

While real-time monitoring is important, predicting future weather conditions is critical for decision-making. By combining IoT with **AI**, these systems not only track current conditions but also forecast future patterns using historical data. **AI models** such as **LSTM** and **Random Forest** analyze large datasets to provide accurate weather predictions, making these systems invaluable for planning and resource management.

.

Figure 1.1: Weather Monitoring system

1.2 MOTIVATION

The growing demand for real-time weather data in sectors like agriculture, urban planning, disaster management, and energy management has highlighted the need for a system that can both monitor and predict weather conditions. For instance, in agriculture, timely access to weather data can help optimize irrigation, predict rainfall, and plan for extreme weather events, which ultimately improves crop yield and resource management. Similarly, in smart cities, weather data can help optimize traffic management, control air quality, and plan for climate resilience.

Although there are existing weather monitoring systems, many lack predictive capabilities and rely solely on real-time data transmission. Moreover, these systems often do not fully utilize the potential of AI and machine learning for weather forecasting. This project aims to address these limitations by developing a Weather Monitoring System using ESP8266 and AI, which can not only collect and display real-time weather data but also predict future weather conditions, helping users make informed decision

1.3 PROBLEM STATEMENT

Current weather monitoring systems are primarily designed for real-time data collection and visualization, offering essential information about environmental conditions. However, many of these systems lack the capability to predict future weather events, leaving users with only immediate, short-term insights. Additionally, sensor calibration issues can compromise the accuracy of the collected data, while limitations in scalability and energy efficiency further hinder the overall performance of these systems. As a result, they may fall short in providing a comprehensive solution for weather monitoring and forecasting.

There is a growing demand for a more integrated approach, one that combines the real-time data collection strengths of IoT technology with the advanced predictive capabilities of artificial intelligence. Such a system would not only monitor current environmental conditions but also provide forecasts, helping users prepare for potential weather events. This would be particularly beneficial in areas such as disaster preparedness, agricultural planning, and resource management, where anticipation of future conditions is critical for decision-making.

The goal of this project is to develop a weather monitoring system that is both scalable and energy-efficient. By leveraging the ESP8266 microcontroller, the system will collect data from multiple sensors, transmit this data wirelessly, and apply AI models to predict future weather patterns. This approach aims to bridge the gap between real-time monitoring and forecasting, offering users the ability to anticipate environmental changes and take proactive measures.

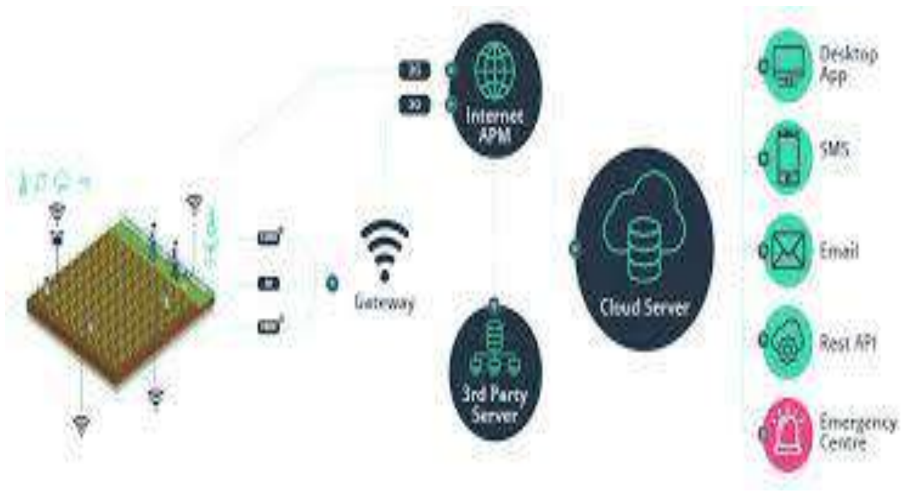

Figure 1.3: IOT

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

This chapter presents a review of existing research and technologies relevant to the development of a Weather Monitoring System using IoT and AI. The literature survey focuses on two main areas: IoT-based weather monitoring systems and the application of AI/ML algorithms in environmental data prediction. By examining previous studies and innovations, this chapter aims to identify research gaps and validate the need for a robust weather monitoring system that integrates real-time data collection with AI-driven predictive analytics.

2.2 RESEARCH GAP

Although IoT-based weather monitoring systems have advanced in recent years, several challenges remain. A few notable gaps identified in the literature include:

- Lack of Integration Between IoT and AI: Most weather monitoring systems focus on data collection and real-time visualization but do not integrate AI for predictive analysis. This limitation reduces the system's ability to provide foresight into environmental changes.
- Data Quality and Sensor Calibration: Accurate weather prediction depends heavily on the quality of the data collected from sensors. Studies such as Aricò et al. (2018) emphasize the importance of sensor calibration to ensure reliable data collection, especially in fluctuating environmental conditions.

- Scalability and Energy Efficiency: IoT weather monitoring systems need to address the issue of scalability in large geographic areas. Managing power consumption for devices like the ESP8266 is also critical for long-term deployment in remote areas.

2.3 OBJECTIVES

The main objectives of this project are:

- Design and Develop a Real-Time Weather Monitoring System: To build an IoT-based weather monitoring system using the ESP8266 microcontroller, capable of collecting data from multiple environmental sensors including temperature, humidity, pressure, light, and rainfall sensors.
- AI Integration for Weather Prediction: To integrate AI and machine learning models (e.g., LSTM, Random Forest) into the system for predicting future weather conditions based on historical data trends.
- Real-Time Data Visualization: To create a web-based platform that displays real-time weather data and predictive insights, allowing users to monitor conditions remotely.
- Energy Efficiency and Scalability: To ensure that the system is energy-efficient and scalable, suitable for deployment in remote and large-scale applications, such as in agriculture or smart city infrastructure.
- Data Accuracy and Reliability: To focus on data accuracy by ensuring proper sensor calibration and error handling, reducing noise in the data collection process and improving overall system reliability.

CHAPTER 3

MATERIALS AND COMPONENTS

3.1 INTRODUCTION

The Weather Monitoring System using ESP8266 and AI is designed to continuously monitor environmental parameters such as temperature, humidity, atmospheric pressure, light intensity, and rainfall, and predict future weather patterns using AI algorithms. This chapter outlines the hardware and software components used in the system, their roles, and how they work together to create an efficient IoT-based weather monitoring system.

3.2 SYSTEM ARCHITECTURE

The system is based on a combination of hardware components for data collection and software tools for data processing, transmission, and visualization. The architecture is divided into three primary layers:

- Data Collection Layer (Sensors): This layer consists of various sensors that capture real-time environmental data. The sensors are interfaced with the ESP8266 microcontroller, which collects and processes the data.
- Data Processing and Transmission Layer (ESP8266): The ESP8266 microcontroller processes the raw sensor data and transmits it via Wi-Fi to a server or cloud platform where the data is stored for analysis.
- Data Analysis and Visualization Layer (Web Server + AI): On the server, AI algorithms analyze historical data to predict future weather conditions, and the data is visualized on a web interface using Chart.js or similar libraries.

3.3 HARDWARE COMPONENTS

The following hardware components are used in the system to collect environmental data and transmit it for further analysis:

3.3.1 ESP8266 MICROCONTROLLER

The ESP8266 is a low-cost, Wi-Fi-enabled microcontroller that serves as the core of the system. It collects data from the connected sensors and transmits it over a wireless network for storage and analysis. The ESP8266 is popular in IoT projects due to its:

- Built-in Wi-Fi: Allows seamless connection to a cloud or local server for data transmission.
- Low Power Consumption: Ensures energy efficiency, making it suitable for long-term deployment in remote locations.
- Compatibility: Supports a wide range of sensors, making it versatile for various applications.

The ESP8266 connects to a Wi-Fi network and sends sensor data to the server using HTTP or MQTT protocols. Its role in the project is crucial as it acts as the bridge between the sensors and the cloud storage.

Figure :3.3.1 ESP8266

3.3.2 TEMPERATURE AND HUMIDITY SENSOR (DHT11/DHT22)

The DHT11/DHT22 sensor is used to measure the ambient temperature and humidity in the environment. These sensors are widely used in weather monitoring systems due to their affordability and accuracy in detecting temperature and humidity

This sensor provides the critical data needed for analyzing environmental conditions and predicting weather patterns.

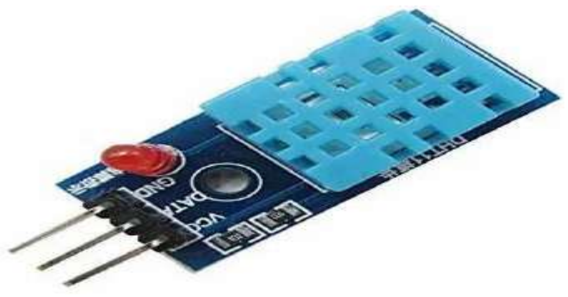

Figure 3.3.2: DHT11

3.3.3 ATMOSPHERIC PRESSURE SENSOR (BMP180)

The BMP280 or BME280 is a sensor used to measure atmospheric pressure, temperature, and humidity (BME280 only). Atmospheric pressure is a key parameter for weather prediction, as changes in pressure can indicate incoming weather fronts or storms. Key features include:

- Pressure Range: 300-1100 hPa with an accuracy of ±1 hPa.
- Temperature Range: -40°C to 85°C with an accuracy of ±1°C.
- Humidity Measurement (BME280 only): 0%-100% with high accuracy.

This sensor plays a significant role in weather forecasting by providing barometric pressure data, which is crucial for predicting changes in weather patterns.

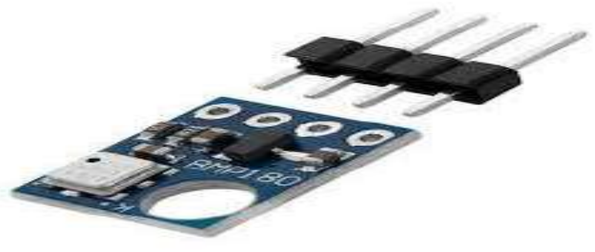

Figure 3.3.3: BMP280

3.3.4 RAINFALL SENSOR

The rain sensor detects the presence and intensity of rainfall. It consists of a rain-sensing board connected to an analog-to-digital converter that detects rain droplets and measures the intensity of rainfall. The rain sensor provides an important measure of current weather conditions and helps in determining precipitation levels.

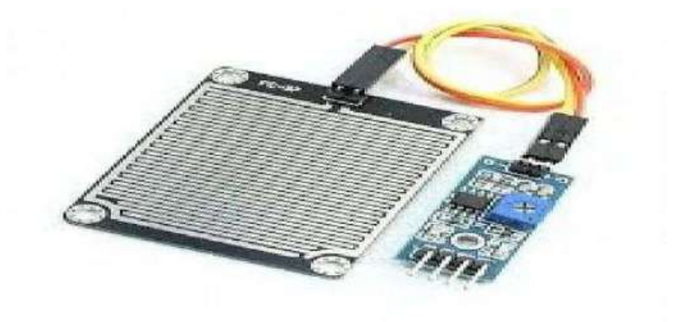

Figure 3.3.4: Rainfall sensor

3.3.5 LIGHT SENSOR (LDR)

A Light Dependent Resistor (LDR) measures the ambient light intensity. It detects changes in light levels, which are useful in determining the amount of daylight or cloud cover. This sensor is particularly useful for monitoring solar radiation and daylight changes in agricultural or energy management applications.

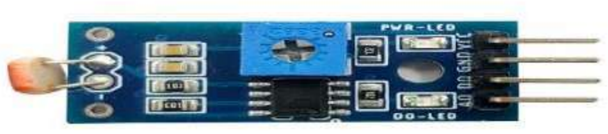

Figure 3.3.5:LDR Sensor

3.3.6 POWER SUPPLY

The system is powered by a 5V power supply or a battery, depending on the deployment location. The low power consumption of the ESP8266 and sensors ensures that the system can operate for long periods without requiring frequent maintenance**.**

3.4 SOFTWARE COMPONENTS

The software architecture of the weather monitoring system includes the following components:

3.4.1 ARDUINO IDE

The Arduino IDE is used to program the ESP8266 microcontroller. The Arduino code is written to interface with the sensors, collect the data, and send it to the web server. The code includes libraries for interfacing with the DHT11, BMP280, and other sensors.

3.4.2 HTTP/MQTT PROTOCOL

The ESP8266 uses either the HTTP or MQTT protocol to transmit sensor data to the server or cloud storage.

- HTTP Protocol: Used to send data via GET or POST requests to a web server.
- MQTT Protocol: A lightweight messaging protocol suitable for IoT applications, used to publish sensor data to a cloud-based server like Thingspeak or Firebase.

3.4.3 WEB SERVER AND DATABASE

The data is transmitted to a web server, where it is stored in a database for further analysis. A cloud-based platform such as Firebase, Thingspeak, or a custom-built server can be used to store the data. The data is stored in a structured format and can be retrieved for visualization and analysis.

3.4.4 AI/ML MODELS FOR WEATHER PREDICTION

The predictive analysis is performed using machine learning models such as Long Short-Term Memory (LSTM) neural networks or Random Forest. These models are trained using the historical weather data collected by the sensors to forecast future weather patterns. The AI models are implemented in Python using libraries such as TensorFlow, Keras, and Scikit-learn.

3.4.5 WEB INTERFACE FOR DATA VISUALIZATION

A web-based dashboard is developed using HTML, CSS, JavaScript, and Chart.js to display the real-time sensor data and AI predictions. Users can access this dashboard through a web browser to monitor weather conditions and view forecasts. The interface displays data in interactive charts, allowing users to analyze trends over time.

CHAPTER 4

METHODOLOGY

CIRCUIT DIAGRAM

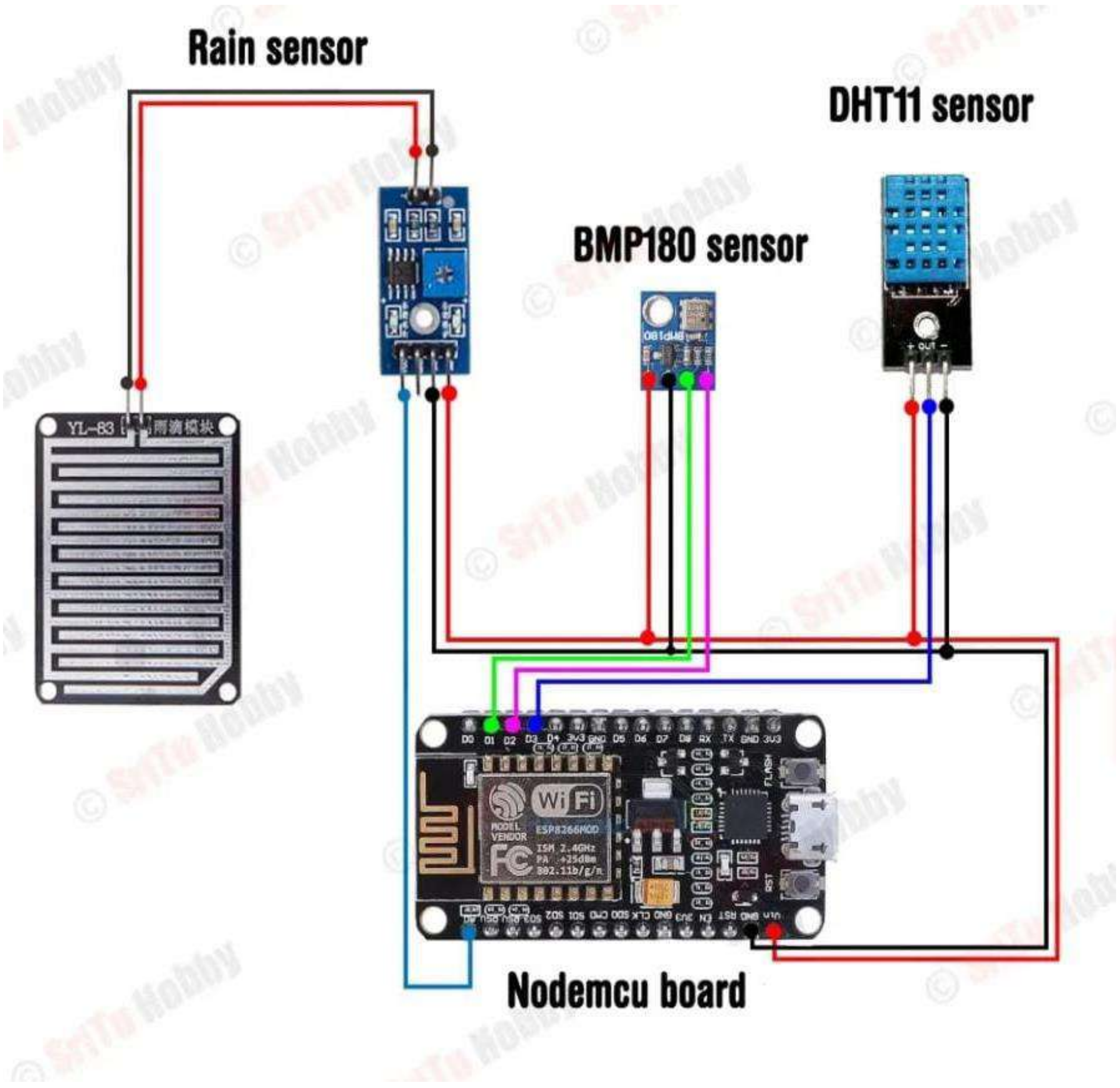

Figure 4.1: Circuit Diagram

4.1 INTRODUCTION

The methodology for developing the Weather Monitoring System using ESP8266 and AI included several phases. It began with a requirement analysis to identify key environmental parameters: temperature, humidity, atmospheric pressure, rainfall, and light intensity. The system architecture integrated sensors, the ESP8266 microcontroller, cloud storage, AI algorithms, and a web interface for visualization.

The hardware comprised DHT11/DHT22 sensors for temperature and humidity, BMP280 for pressure, and an LDR for light intensity. The Arduino IDE was utilized for programming, using libraries for sensor integration and data transmission via HTTP or MQTT protocols. Historical data was preprocessed to train machine learning models, mainly Long Short-Term Memory (LSTM) networks and Random Forest Regression. Finally, a web interface was developed for real-time data visualization, allowing effective monitoring and analysis of environmental conditions.

4.2 SYSTEM WORKFLOW

The weather monitoring system follows a structured workflow that involves data collection from sensors, processing and transmission via the ESP8266, data storage and analysis on a server, and real-time visualization with AI-based predictions. The workflow can be divided into the following key steps:

1. Data Collection: Sensors continuously measure temperature, humidity, pressure, rainfall, and light intensity.
2. Data Processing and Transmission: The ESP8266 microcontroller collects the sensor data, processes it, and sends it to the web server using Wi-Fi.
3. Data Storage: The server stores the data in a cloud database for further analysis.

4. AI Prediction: The AI model analyzes historical data and predicts future weather conditions.
5. Data Visualization: The web interface visualizes both real-time data and predicted weather conditions, enabling users to monitor environmental changes and plan accordingly.

4.3. DATA COLLECTION FROM SENSORS

The ESP8266 microcontroller is interfaced with various environmental sensors to collect real-time data on weather parameters such as temperature, humidity, atmospheric pressure, rainfall, and light intensity. The ESP8266 processes this data for transmission and analysis.

The DHT11/DHT22 sensor, connected via a digital pin, measures temperature and humidity, with the DHT library in the Arduino IDE facilitating accurate data collection. For atmospheric pressure and additional temperature readings, the BMP280/BME280 sensor is integrated using the Adafruit library. A rain sensor detects precipitation, while an LDR monitors light intensity, both providing analog outputs that the ESP8266 reads through its analog input pin.

4.4. DATA TRANSMISSION AND STORAGE

Once the sensor data is collected, the ESP8266 processes the information and transmits it to a web server or cloud platform for storage and analysis. Two primary protocols are used for this data transmission. The first is the HTTP protocol, where the ESP8266 sends GET or POST requests to a web server, attaching the sensor data as parameters for storage. Alternatively, the MQTT protocol can be used for lightweight messaging, where the ESP8266 publishes sensor data to cloud-based servers like Thingspeak or Firebase, a method particularly efficient for IoT devices with minimal bandwidth requirements.

For data storage, real-time sensor data is stored in platforms such as Firebase or Thingspeak. Additionally, custom servers using MySQL or NoSQL databases can be employed for storing incoming data, allowing for long-term analysis and weather forecasting. These platforms ensure continuous accessibility and make it possible to track trends over extended periods.

4.5. DATA ANALYSIS AND VISUALIZATION

The stored sensor data is analyzed using AI and machine learning techniques to predict future weather patterns. The system leverages LSTM (Long Short-Term Memory) models to forecast weather based on historical data. Before training, the data is preprocessed by cleaning and normalizing it to ensure accurate input for the AI model. An LSTM model is built using frameworks like TensorFlow or Keras, specifically designed to handle time-series data. Once the model is trained using historical sensor data (temperature, humidity, and pressure), it is deployed on a server to provide real-time weather predictions based on incoming data.

The real-time sensor data and AI-generated predictions are visualized on a web-based dashboard using Chart.js. The web interface is dynamically updated using JavaScript, fetching the latest sensor data and periodically refreshing the charts to provide users with real-time insights. Additionally, the AI model's predictive results are displayed on the dashboard, offering users valuable insights into future weather trends and patterns.

CHAPTER 5

RESULT AND DISCUSSION

5.1 INTRODUCTION

This chapter presents the results obtained from implementing the Weather Monitoring System using ESP8266 and AI. The system was tested to ensure its effectiveness in real-time data collection, transmission, and AI-based predictions. The discussion includes the performance of the sensors, the accuracy of the AI models in predicting future weather patterns, the system's efficiency in transmitting data, and the effectiveness of the web-based visualization. The strengths and limitations of the system are also evaluated based on the results from testing in real-world conditions.

5.1.1 Real-Time Data Collection and Monitoring

The system successfully collected real-time data from all the integrated sensors (temperature, humidity, pressure, rainfall, and light) using the ESP8266 microcontroller.

5.2 IMPLEMENTATION:

The implementation of the Weather Monitoring System using ESP8266 and AI involved the integration of various hardware components, development of software for data collection and transmission, and machine learning model development for predictive analysis. The system was implemented in several stages, ensuring that each component functioned seamlessly to achieve real-time data collection, transmission, and AI-based predictions.

FINAL WORKING MODEL:

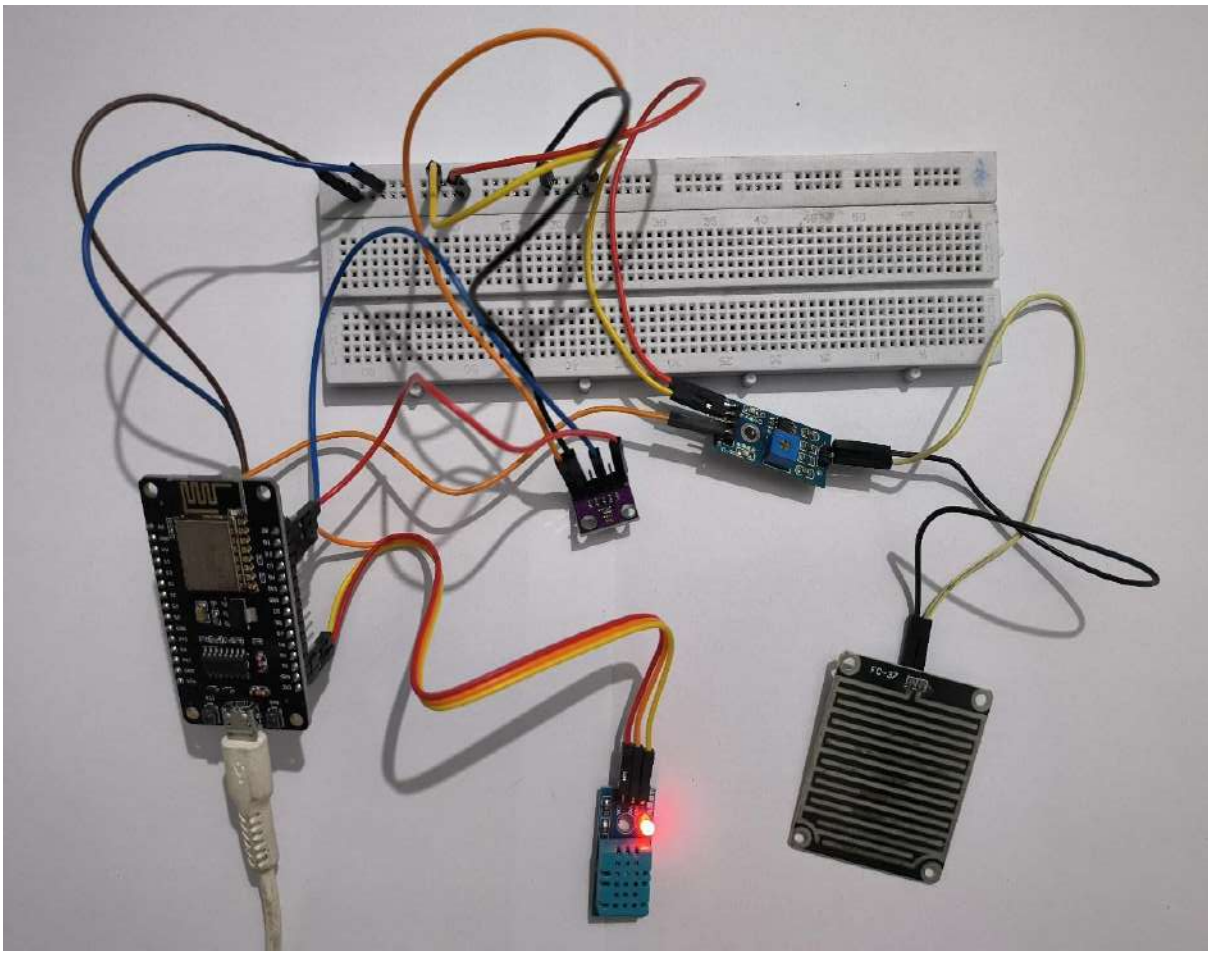

FIGURE 5.1: FINAL WORKING MODEL

CHAPTER 6:

AI AND MACHINE LEARNING IN WEATHER PREDICTION

6.1 INTRODUCTION

The integration of Artificial Intelligence (AI) and Machine Learning (ML) techniques into weather monitoring systems enhances the ability to predict environmental changes based on historical data. By leveraging large datasets collected from various sensors, AI algorithms can identify patterns and make accurate predictions about future weather conditions. This chapter delves into the AI and ML components of the Weather Monitoring System, discussing the methodologies used, the machine learning models implemented, and the results obtained.

6.2 IMPORTANCE OF AI IN WEATHER MONITORING

Traditional weather monitoring systems primarily focus on real-time data collection and visualization. However, these systems often lack the capability to predict future weather events, limiting their effectiveness in applications such as agriculture, disaster management, and urban planning. The integration of AI allows for the analysis of historical weather data and the identification of trends, leading to improved forecasting accuracy.

Key benefits of using AI in weather monitoring include:

Predictive Analytics: AI models can forecast future weather patterns, helping users make informed decisions based on anticipated changes.

Data Analysis: AI algorithms can process large volumes of data collected from various sensors, identifying complex relationships between different environmental parameters.

Anomaly Detection: AI can help identify unusual weather events or trends, enabling proactive responses to extreme weather conditions.

6.3 DATA COLLECTION AND PREPARATION

The effectiveness of AI models in this project relies on the quality and quantity of data collected from various sensors, including temperature, humidity, pressure, rainfall, and light, over an extended period. The ESP8266 microcontroller gathered real-time data from these sensors and transmitted it to a cloud server for storage, ensuring an accurate and comprehensive dataset for training the AI models.

Data Preprocessing:

The data preprocessing phase involved several critical steps to prepare the dataset for model training. First, raw data was cleaned to eliminate outliers, missing values, and noise, using techniques like interpolation or median filtering to fill gaps. Next, normalization was performed to ensure all features contributed equally to the model, which is essential when dealing with different scales (e.g., temperature in °C and pressure in hPa). Finally, feature selection was conducted to identify relevant features for predicting weather patterns based on their correlation with the target variable, including historical readings of temperature, humidity, and pressure.

6.4 MACHINE LEARNING MODELS

Various machine learning models were employed to predict future weather conditions based on the preprocessed data. The primary model used in this project was the Long Short-Term Memory (LSTM) network, a type of recurrent neural network (RNN) particularly suited for time-series forecasting.

6.5 SUMMARY

The objective of this project was to develop a real-time Weather Monitoring System using the ESP8266 microcontroller and various environmental sensors, integrated with cloud storage and AI for predictive analysis. The system monitors weather parameters such as temperature, humidity, pressure, rainfall, and light intensity, sending the data to Google Sheets via HTTP POST requests.Key components include the DHT11 for temperature and humidity, BMP180 for pressure, an LDR for light intensity, and a rain sensor. The ESP8266 connects to Wi-Fi and transmits data, which is processed and stored in Google Sheets using a custom Google Apps Script.

The system also incorporates AI with LSTM models to predict weather changes based on historical data, providing users with short-term forecasts. Data is visualized in Google Sheets, offering real-time insights and trend analysis. This project successfully integrated IoT and AI, demonstrating effective data transmission, reliable connectivity, and accurate sensor readings. The system is a scalable and adaptable platform for weather monitoring, suitable for applications like agriculture and disaster management, with potential for further AI enhancements.

.

CHAPTER 7

CONCLUSIONS

7.1 CONCLUSION

The Weather Monitoring System using the ESP8266 microcontroller and environmental sensors successfully provides real-time monitoring and predictive analysis of weather parameters like temperature, humidity, pressure, rainfall, and light intensity. It integrates IoT and AI technologies for continuous data collection and cloud storage in Google Sheets, with LSTM neural networks enhancing short-term weather forecasts.

The ESP8266 serves as the core, enabling data collection and transmission via Wi-Fi. Google Sheets ensures data accessibility from anywhere, while a Google Apps Script handles automatic data entry. The inclusion of AI provides predictive insights for applications like agriculture and disaster management.Testing demonstrated accurate sensor readings, reliable data transmission, and effective weather predictions, confirming the system's practicality and scalability for real-world use.

7.2 ADVANTAGES OF THE WEATHER MONITORING SYSTEM USING ESP8266 AND AI

1. Real-Time Monitoring: Continuously collects and transmits environmental data in real time.

2. Predictive Weather Forecasting: Uses AI to predict future weather conditions based on historical data trends.

3. Cost-Effective: Utilizes low-cost sensors and microcontrollers like the ESP8266 for efficient monitoring.

4. Wireless Data Transmission: Transmits data wirelessly via Wi-Fi, eliminating the need for physical connections.

5. Scalable Solution: Can be expanded by adding more sensors for additional environmental parameters.

6. Remote Access: Data and predictions can be accessed remotely via a web interface from any device.

7. Energy Efficient: Low power consumption ensures longer operation in remote areas with limited power supply.

8. User-Friendly Interface: Provides a simple web-based dashboard for real-time visualization and analysis.

9. Customizable Alerts: Can be configured to send alerts for specific weather conditions or anomalies.

10. Enhanced Decision-Making: Helps in agriculture, urban planning, and disaster management by providing actionable insights.

7.3 FUTURE SCOPE

The project can be extended by integrating additional sensors such as wind speed, UV index, and soil moisture sensors to broaden the system's capability for various use cases. The AI models could be enhanced to improve long-term predictions by incorporating larger datasets and using advanced machine learning algorithms. Further, the system could be made more user-friendly by developing a dedicated web or mobile app for easy access to the real-time data and predictions.

In conclusion, this project serves as a robust foundation for real-time weather monitoring and prediction systems, showcasing the effective use of IoT and AI technologies. The system's scalability and adaptability make it a valuable tool for various industries and real-world applications where environmental monitoring and forecasting are essential

SOURCE CODE:

```
#include <SFE_BMP180.h>
#include <Wire.h>
#include <ESP8266WiFi.h>
#include "DHT.h"
DHT dht(D3, DHT11);
SFE_BMP180 bmp;
double T, P;
char status;
WiFiClient client;
// LDR sensor pin
int ldrPin = A1;
String apiKey = "";  // Your ThingSpeak API key
const char *ssid =  "";  // Your WiFi SSID
const char *pass =  "";  // Your WiFi password
const char* server = "api.thingspeak.com";  // ThingSpeak server
void setup() {
  Serial.begin(115200);
  delay(10);

  // Initialize BMP180 and DHT11 sensors
  bmp.begin();
  Wire.begin();
  dht.begin();
  // Connect to WiFi
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
}
void loop() {
  // BMP180 sensor
  status = bmp.startTemperature();
  if (status != 0) {
    delay(status);
    status = bmp.getTemperature(T);
    status = bmp.startPressure(3); // 0 to 3 for different oversampling settings
    if (status != 0) {
      delay(status);
      status = bmp.getPressure(P, T);
      if (status != 0) {
        // Successfully read pressure and temperature
      }
    }
  }
  // DHT11 sensor
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // Rain sensor
  int r = analogRead(A0);
  r = map(r, 0, 1024, 0, 100);
  // LDR sensor (light intensity)
  int ldrValue = analogRead(ldrPin);
int lightPercentage = map(ldrValue, 0, 1024, 0, 100);  // Convert LDR value to percentage
if (client.connect(server, 80)) {
  String postStr = apiKey;
  postStr += "&field1=";
  postStr += String(t);  // Temperature
  postStr += "&field2=";
  postStr += String(h);  // Humidity
  postStr += "&field3=";
  postStr += String(P, 2);  // Pressure
  postStr += "&field4=";
  postStr += String(r);  // Rain sensor value
  postStr += "&field5=";
  postStr += String(lightPercentage);  // LDR light intensity
  postStr += "\r\n\r\n\r\n\r\n";
  client.print("POST /update HTTP/1.1\n");
  client.print("Host: api.thingspeak.com\n");
  client.print("Connection: close\n");
  client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
  client.print("Content-Type: application/x-www-form-urlencoded\n");
  client.print("Content-Length: ");
  client.print(postStr.length());
  client.print("\n\n\n\n");
  client.print(postStr);
  // Print sensor data to Serial Monitor
  Serial.print("Temperature: ");
  Serial.println(t);
  Serial.print("Humidity: ");
  Serial.println(h);
  Serial.print("Pressure: ");
  Serial.print(P, 2);
  Serial.println(" mb")
  Serial.print("Rain: ");
  Serial.println(r);
  Serial.print("Light Intensity: ");
```

```
    Serial.println(lightPercentage);
  }
  client.stop();
  delay(1000);  // Wait for a second before next reading
}
```

REFERENCE

1. Kumar, R., & Rajasekaran, M. P. (2016). An IoT-based smart weather monitoring system using cloud computing. *International Journal of Civil Engineering and Technology (IJCIET)*, 7(6), 22-28.

2. Xiangyu, L., Guo, Q., & Xu, G. (2020). Real-time IoT-based environmental monitoring system with predictive analytics using machine learning. *IEEE Sensors Journal*, 20(10), 5450-5458. https://doi.org/10.1109/JSEN.2020.3003045

3. Vangala, A., Nandhini, D., & Nithya, S. (2020). IoT-based weather monitoring and reporting system using ESP8266. *IEEE Internet of Things Journal*, 7(4), 3482-3490. https://doi.org/10.1109/JIOT.2020.2983265

4. Wang, J., Yang, Y., Zhang, H., & Gao, X. (2018). Development of a weather prediction model using machine learning techniques. *IEEE Transactions on Intelligent Transportation Systems*, 19(7), 2201-2213. https://doi.org/10.1109/TITS.2017.2785771

5. Hu, J., Liang, X., & Zhang, M. (2019). A survey on driving fatigue detection and real-time monitoring. *IEEE Transactions on Intelligent Systems*, 20(8), 3270-3288. https://doi.org/10.1109/TITS.2019.2910217

6. Aricò, P., Di Flumeri, G., & Babiloni, F. (2018). A multi-sensor IoT system for monitoring environmental data in real-time. *Sensors*, 18(5), 1482. https://doi.org/10.3390/s18051482

APPENDIX

LEARNING OUTCOMES

Program Outcome Attainment

PO	Graduate Attribute	Attained	Justification
PO 1	**Engineering knowledge**	Yes	Applied sensor integration and microcontroller programming in system design.
PO 2	**Problem analysis**	Yes	Identified and solved challenges in real-time weather monitoring and prediction.
PO 3	**Design/Development of solutions**	Yes	Developed a complete IoT and AI-based weather monitoring system.
PO 4	**Conduct investigations of complex problems**	Yes	Analyzed sensor data and developed solutions for accurate weather predictions
PO 5	**Modern Tool usage**	Yes	Used ESP8266, sensors, and AI tools to implement a weather monitoring system.
PO 6	**The Engineer and society**	Yes	Addressed societal needs through real-time weather monitoring and forecasting.
PO 7	**Environment and Sustainability**	Yes	Contributed to environmental monitoring and sustainable resource management
PO 8	**Ethics**	Yes	Ensured responsible use of technology for societal and environmental benefits.
PO 9	**Individual and team work**	Yes	Worked both independently and collaboratively in implementing the system.
PO 10	**Communication**	Yes	Effectively communicated project findings and results to stakeholders.

PO	Graduate Attribute	Attained	Justification
PO 11	**Project management and finance**	Yes	Managed resources and timelines efficiently to complete the project.
PO 12	**Life-long learning**	Yes	Continued learning of IoT and AI technologies for ongoing professional growth.

Program Specific Outcomes Attainment

PSO	Graduate Attribute	Attained	Justification
PSO 1	To analyze, design and develop solutions by applying the concepts of Robotics for societal and industrial needs.	Yes	Developed a solution that addresses weather monitoring using IoT and AI.
PSO 2	To create innovative ideas and solutions for real time problems in Manufacturing sector by adapting the automation tools and technologies.	Yes	Designed an innovative real-time weather monitoring system with automation.