## IOT Based Smart Weather Monitoring System

**A Project Report**

***In the partial fulfillment for the award of the degree of***

**B.Tech**

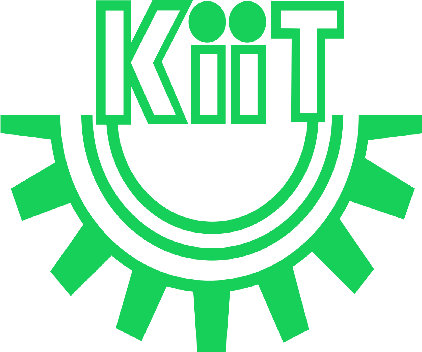
under

# Academy of Skill Development

****

***Submitted by***

**Piyush Raj**

****

# Kalinga Institute of Industrial Technology

**Bhubaneswar, Odisha, India-751024**



**Certificate from the Mentor**

This is to certify that **Piyush Raj** has successfully completed the project titled **IOT Based smart weather Monitoring System** under my supervision during the period from **May to July** which is in partial fulfillment of requirements for the award of the **B.Tech** and submitted to Department **Electronics and Computer Science Engineering** of **Kalinga Institute of Industrial Technology, Bhubaneswar, Odisha, India-751024.**

## Date: 30/06/2025

***Signature of the Mentor***



**Acknowledgement**

I take this opportunity to express my deep gratitude and sincerest thanks to my project mentor, **Debtanu Mondal** for giving the most valuable suggestion, helpful guidance and encouragement in the execution of this project work.

I would like to give a special mention to my colleagues. Last but not the least I am grateful to all the faculty members of **Academy of Skill Development** for their support.

# IOT Based Smart Weather Monitoring System

## ABSTRACT

The **"IoT-Based Smart Weather Monitoring System"** is an advanced and efficient solution designed to monitor real-time environmental conditions using the **ESP32 microcontroller**. This system utilizes various sensors such as the **DHT11/DHT22** for measuring temperature and humidity, the **rain sensor** to detect rainfall, and a **gas sensor (like MQ-135)** to identify harmful gases in the atmosphere. The **ESP32** serves as the central controller with built-in Wi-Fi, making it ideal for seamless cloud connectivity.

Sensor data is collected continuously and transmitted to the **Blynk cloud platform**, allowing users to remotely monitor weather conditions through their smartphones. The Blynk app provides real-time visualization and alerts based on environmental changes, ensuring users stay informed about temperature spikes, humidity variations, rainfall, or air quality issues.

This system offers a scalable, low-cost solution suitable for smart cities, agriculture, industrial environments, and residential areas. By integrating IoT technology, the project promotes **efficient environmental monitoring**, timely alerts, and **better decision- making** based on accurate weather data.

**Table of Contents**

1. **Introduction** 4
2. **Introduction to IoT** 4
3. **Key Components of IoT** 4-5
4. **Components Used in the Project** 6
5. **Component Descriptions** 6
   1. ESP32 Microcontroller 6
   2. DHT11 Temperature and Humidity Sensor 7
   3. Rain Sensor 8
   4. Gas Sensor (MQ-135 or MQ-08) 8
   5. Blynk Cloud Platform 9
6. **Project Overview** 11
7. **How the System Works** 13
8. **Types of Information Provided by the System** 14
9. **Software and Programming Language Used** 14
10. **Project Code** 16
11. **Result and Discussion** 19
12. **Prototype Circuit Diagram** 20
13. **Future Scope** 22
14. **Conclusion** 23
15. **References** 23

## Introduction

### Introduction to IOT (Internet of Things)

The Internet of Things (IOT) refers to the interconnected network of physical devices embedded with sensors, actuators, software, and other technologies that enable them to collect, exchange, and act upon data. IOT has revolutionized the way devices communicate and operate, creating a seamless integration between the digital and physical worlds. This connectivity allows for the automation and optimization of various processes across multiple industries, enhancing efficiency, productivity, and convenience.

## Key Components of IOT

### Sensors

Sensors are fundamental to IOT systems, as they collect data from the physical environment. They measure various parameters such as temperature, humidity, pressure, light, motion, gas concentration, and more. Sensors convert these physical measurements into electronic signals that can be analysed and processed. In an IOT- based smart smoke detection and prevention system, for example, sensor like IR based sensor and MQ-08 (smoke sensor) are used to monitor situation of a place continuously.

### Actuators

Actuators are devices that perform actions or control mechanisms in response to commands received from the IOT system. While sensors collect data, actuators act upon this data. They can control a variety of systems, such as opening valves, starting motors, or adjusting thermostats. Actuators play a critical role in applications where physical action is required, such as smart homes (automatic lighting and temperature control), industrial automation (robot arms), and automotive systems (self-driving car manoeuvres).

### Connectivity

Connectivity is the backbone of IOT, enabling communication between devices and systems. It involves various network technologies, including Wi-Fi, Bluetooth, Zigbee, LoRaWAN, cellular networks (3G, 4G, 5G), and Ethernet, among others. Connectivity allows devices to transmit data to cloud servers or central hubs for processing and analysis. In the IOT based power plant monitoring system, the ESP32 microcontroller uses Wi-Fi to send data from the sensors to the Blynk cloud server for real-time monitoring and notifications.

### Data processing

Data processing is the core of IOT systems, where collected data is analysed to derive meaningful insights and actionable intelligence. This can be done locally on the device

(edge computing) or in centralized data centres (cloud computing). The choice between edge and cloud computing depends on the application requirements, such as latency, bandwidth, and computational power. In our example, data processing involves analysing temperature, air quality, and fire alert data to detect anomalies and trigger alerts or responses accordingly.

### Security

Security is a critical component of IOT, given the vast amount of data transmitted and the potential vulnerabilities in connected devices. IOT security involves ensuring data integrity, confidentiality, and availability through encryption, authentication, and authorization mechanisms. Secure IOT systems protect against threats such as data breaches, unauthorized access, and cyberattacks. Effective security measures are essential to safeguard sensitive data and maintain user trust in IOT applications.

## Applications

IOT has a wide range of applications across various sectors, including:

**Smart Homes and Buildings**: IOT enables automation of lighting, heating, security systems, and appliances, providing enhanced comfort, energy efficiency, and security.

**Healthcare:** IOT devices, such as wearables and remote monitoring systems, track patient health data in real-time, improving healthcare delivery and patient outcomes.

**Industrial IOT (IIOT):** IOT optimizes manufacturing processes, predictive maintenance, and supply chain management, enhancing operational efficiency and reducing downtime.

**Smart Cities:** IOT solutions improve urban infrastructure, including traffic management, waste management, energy distribution, and public safety.

**Agriculture:** IOT technology supports precision farming by monitoring soil conditions, weather, and crop health, enabling more efficient resource use and higher yields. The Internet of Things (IOT) represents a transformative technology that connects the digital and physical worlds through a network of interconnected devices.

## Components used in the Project

Here is a list of the total components used in the "IoT-Based Smart Weather Monitoring System using ESP32":

1. ESP32 Microcontroller: Acts as the central controlling unit of the system, responsible for reading sensor data, processing it, and sending it to the cloud server.
2. DHT11 Sensor: Used for monitoring temperature and humidity in the surrounding environment.
3. Rain Sensor: Detects the presence of rain or water droplets on its surface.
4. MQ-135 Gas Sensor: Used for detecting air quality and the presence of harmful gases in the atmosphere.
5. Blynk Cloud Server: Used for data storage, real-time monitoring, and sending alerts/notifications to the user through the mobile app.
6. Wi-Fi Module (Built-in with ESP32): Provides seamless wireless connectivity between the ESP32 and the Blynk cloud platform.
7. Power Supply (e.g., 5V/9V Battery or Adapter): Powers the ESP32 microcontroller and connected sensors.
8. Connecting Wires and Jumpers: Used to establish circuit connections between sensors and the ESP32.
9. Breadboard: Used for prototyping the system without soldering, allowing quick testing and adjustments.
10. LEDs: Provide visual indication of system power status or sensor activity.
11. Buzzer (Optional): Can be used to provide an audible alert for extreme weather conditions like very high temperature or gas levels.

These components work together to build a functional IoT-based system that can monitor real-time weather conditions and environmental parameters, and send live data to the cloud for remote access and decision-making.

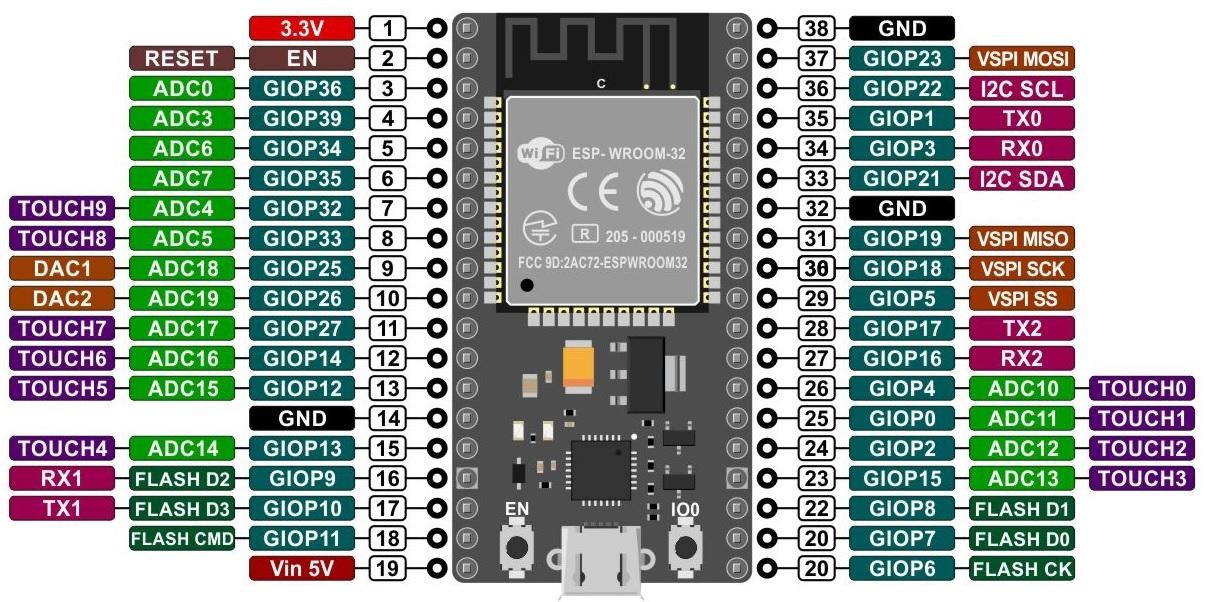
## Components Descriptions

### ESP32 Microcontroller:

* **Integrated Wi-Fi and Bluetooth Connectivity**: The **ESP32** is a powerful microcontroller with built-in **Wi-Fi and Bluetooth**, eliminating the need for external communication modules. This makes it ideal for IoT applications requiring real-time cloud connectivity and wireless sensor data transmission.
* **Dual-Core 32-bit Processor**: It operates on a **dual-core processor** running up to **240 MHz**, offering enhanced speed and multitasking capabilities compared to older boards like ESP8266.
* **Low Power Consumption**: The ESP32 supports **multiple power-saving modes**, making it suitable for battery-operated systems such as remote weather stations in the field.
* **Multiple GPIO Pins and Interfaces**: The board features a wide range of **GPIO**,

**ADC**, **PWM**, **I2C**, **SPI**, and **UART** pins for interfacing with various weather sensors like DHT11, Rain Sensor, and Gas Sensors.

* **Developer-Friendly and Versatile**: It can be easily programmed using the **Arduino IDE**, **PlatformIO**, or **MicroPython**, making it accessible to both beginners and advanced developers.



### Applications in the Project:

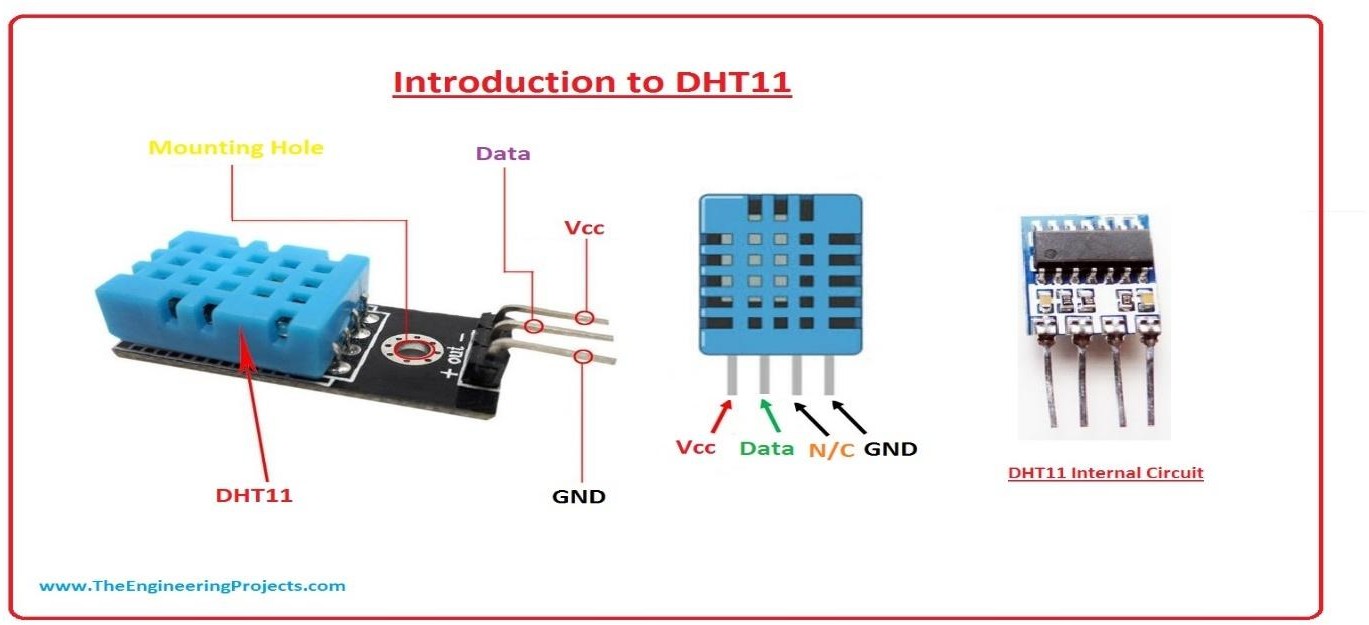
* **Weather Monitoring System**: Acts as the central unit to collect data from sensors (temperature, humidity, gas, rain) and send it to the cloud for real-time monitoring.
* **Remote Environmental Sensing**: Transmits live sensor data to the **Blynk cloud platform**, enabling users to monitor conditions remotely via smartphone.
* **Smart IoT Projects**: Due to its flexibility, ESP32 can be expanded to support additional sensors or automation like triggering alerts or logging data over time.

### DHT11 Temperature and Humidity Sensor Features:

* + **Temperature and Humidity Measurement**: The DHT11 sensor is capable of measuring both **ambient temperature** and **relative humidity**, providing essential environmental data for weather monitoring.
  + **Digital Output**: It provides a **digital signal** output, making it easy to interface with the ESP32 without requiring any additional analog-to-digital conversion.
  + **Reliable and Stable Performance**: Offers **decent accuracy** for low-cost applications, suitable for basic weather stations and environmental sensing.
  + **Low Power Consumption**: The sensor is optimized for **low power usage**, making it ideal for continuous operation in IoT systems.
  + **Compact and Cost-Effective**: Small in size and budget-friendly, it's widely used

in academic and DIY IoT projects.

* **Low Power Consumption:** The sensor consumes minimal power, making it suitable for continuous monitoring applications.



**Figure 2 : DHT 11 Sensor**

### MQ-8 Sensor: Key Features:

* + **Detection Capabilities:**
    - It's sensitive to gases like LPG (liquefied petroleum gas), methane, butane, propane, smoke, and even hydrogen.
    - This makes it versatile for various gas leakage detection applications.

### Operating Principle:

* + - It's a Metal Oxide Semiconductor (MOS) sensor, also known as a chemiresistor.
    - Its sensing element's resistance changes when it comes into contact with target gases.

This resistance variation is then used to determine the gas concentration.

### Applications:

* + - **Gas Leak Detection:** Widely used in home and industrial settings to detect gas leaks, enhancing safety.
    - **Smoke Detection:** Can also detect smoke, making it useful in smoke alarms.
    - **Air Quality Monitoring:** Contributes to air quality monitoring systems.

### Technical Aspects:

* + - Typically operates on a 5V DC power supply.
    - Provides both analog and digital outputs, allowing for flexible interfacing with microcontrollers like Arduino.

10 | P a g e

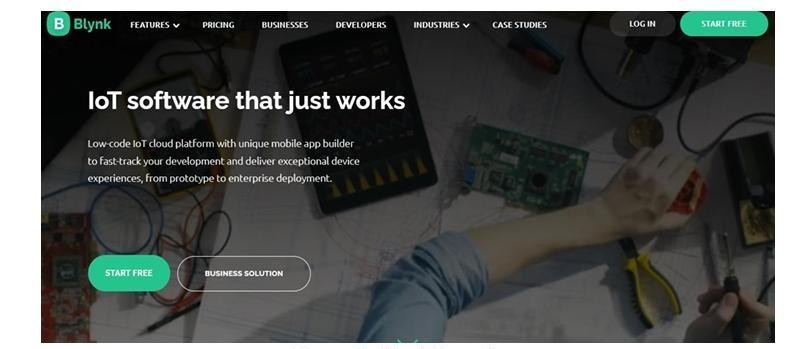
* + - Requires a preheat time to stabilize before taking accurate readings.



## Figure 3 : MQ-8 Sensor

1. **Blynk Cloud Server Features**
   * **Real-Time Data Monitoring:** Blynk provides a cloud platform that allows users to monitor data from their IOT devices in real-time through a smartphone app or web dashboard.
   * **Remote Control:** Users can remotely control connected devices and actuators through the Blynk app, enabling remote management and automation.
   * **Notification Alerts:** Blynk supports real-time notifications, such as push notifications, emails, or SMS, to alert users about critical events or threshold breaches.
   * **Cross-Platform Compatibility:** Blynk works across multiple platforms (iOS, Android, and web), providing a seamless user experience for remote monitoring and control.
   * **Easy Integration and Customization:** The platform offers a user-friendly interface for developers to integrate and customize their IOT applications easily. Applications
   * **Smart Home Automation:** Used to control and monitor home appliances, lights, and security systems remotely.
   * **Industrial IOT:** Enables remote monitoring and management of industrial equipment and processes to optimize performance and prevent downtime.
   * **Smart Agriculture:** Allows farmers to monitor soil conditions, weather data, and crop health remotely and make data-driven decisions.
   * **Health Monitoring:** Facilitates real-time health data monitoring for patients through wearable IOT devices, enabling remote health management.

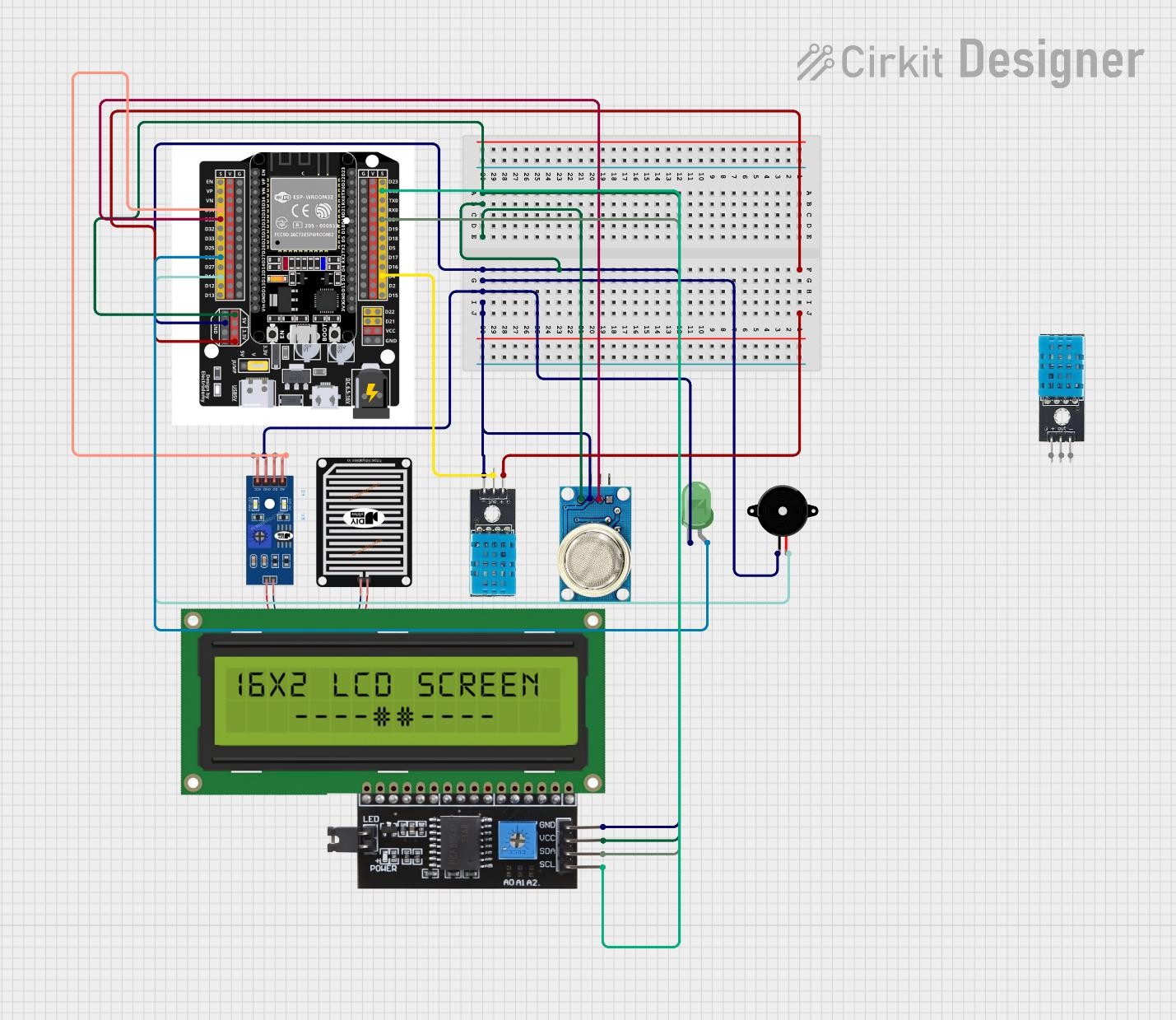
10 | P a g e



## Figure 4 : Blynk Console

The **"IoT-Based Smart Weather Monitoring System"** leverages several key components to deliver a reliable and real-time environmental monitoring solution. The **ESP32 microcontroller** acts as the system's brain, interfacing with sensors such as the **DHT11** (for temperature and humidity), **rain sensor**, and **MQ-135 gas sensor** to collect comprehensive weather and air quality data.

This data is transmitted to the **Blynk cloud server**, which enables **real-time monitoring**, **remote access**, and **instant alerts** through a mobile application. Each component is vital in ensuring the system operates accurately, efficiently, and continuously—providing users with up-to-date information on environmental conditions and supporting better decision-making in applications like smart cities, agriculture, and home automation.



## Circuit Diagram

**Project Overview**

The IoT-Based Smart Weather Monitoring System is designed to continuously monitor environmental parameters such as temperature, humidity, rainfall, and air quality in real-time. This system provides valuable data for weather prediction, air quality assessment, and smart automation in homes, agriculture, and smart cities.

Core Concepts:

* Sensor Integration:

The system integrates multiple sensors to collect accurate weather data:

* + DHT11 Sensor – Measures temperature and humidity.
  + Rain Sensor – Detects rainfall or water presence.
  + MQ-135 Gas Sensor – Monitors air quality by detecting harmful gases and pollution levels.
* IoT Connectivity:
  + All sensor data is wirelessly transmitted using Wi-Fi via the ESP32 microcontroller, which has built-in wireless capability.
  + The ESP32 processes sensor readings and sends the data to the Blynk cloud platform for remote access.
* Real-Time Monitoring and Alerting:
  + Users can view real-time data on a mobile application using Blynk.
  + Optional alerts can be set for specific thresholds such as:
    - High temperature
    - Poor air quality
    - Rainfall detection
  + The system can also trigger buzzers or LED indicators for local alerts.
* Data Logging and Analysis:
  + The system logs weather data in the cloud, allowing:
    - Historical data tracking
    - Graphical visualization
    - Improved decision-making for environmental or agricultural activities

Key Components:

* + Sensors: DHT11, Rain Sensor, MQ-135
  + Microcontroller: ESP32 (for data processing and wireless connectivity)
  + Communication Module: Built-in Wi-Fi on ESP32
  + Display and Indicators: LCD screen, LEDs, Buzzer
  + Cloud Platform: Blynk (for real-time monitoring and data storage)
  + Mobile App Interface: For accessing live sensor data and alerts

Applications:

* + Smart homes and buildings
  + Agriculture and irrigation management
  + Air pollution monitoring in urban areas
  + Environmental data logging in remote locations
  + Smart city infrastructure
  + Weather research and forecasting

### Benefits:

* + **Real-time environmental data monitoring**
  + **Improved accuracy** in weather-based decision-making
  + **Remote access** to temperature, humidity, air quality, and rainfall data via smartphone
  + **Early detection** of pollution or hazardous air conditions
  + **Automation-ready** for smart homes or agricultural systems
  + **Low-cost, scalable** solution for smart city and rural applications

In essence, the **IoT-based Smart Weather Monitoring System** utilizes sensor integration and wireless communication to provide a proactive and efficient weather data collection and alerting platform.

## How the System Works:

### Sensor Data Acquisition

* + Sensors such as **DHT11**, **Rain Sensor**, and **MQ-135** continuously monitor the environment.
  + The sensors detect key weather parameters such as **temperature**, **humidity**, **rain presence**, and **air quality**.
  + These readings are sent to the **ESP32 microcontroller**.

### Data Processing

* + The ESP32 processes the incoming data and checks whether any values exceed predefined thresholds (e.g., high temperature, poor air quality).
  + It may calculate averages, trends, or perform basic filtering before forwarding the data.

### Alerts and Actions

* + If any parameter exceeds safe or preset limits:
    - A **buzzer** can be activated for audio alerts.
    - **LEDs** (Red/Green) indicate the system status visually.
    - Simultaneously, data is sent to the **Blynk cloud server**.

### Remote Monitoring and Control

* + Users can monitor **live sensor values** and **system alerts** through the **Blynk mobile app**, even if they are not physically present.
  + Depending on the setup, users can:
    - Receive **notifications** (e.g., if rain is detected or air quality worsens).
    - **Visualize sensor data** through graphs and widgets.

### Automation and Future Control

* + With relays and actuators integrated, future versions of the system could:
    - **Trigger ventilation fans** if air quality is poor
    - **Close/open windows** when rain is detected
    - **Control irrigation** based on humidity and temperature

The "IoT-Based Smart Weather Monitoring System" delivers an intelligent, real-time, and remotely accessible weather station solution. By using sensors like DHT11, MQ- 135, and Rain Sensor, in combination with the ESP32 microcontroller and Blynk cloud, the system demonstrates how IoT can revolutionize environmental monitoring, smart agriculture, and climate-responsive infrastructure.

## Types of Information Provided by the System:

The IoT-Based Smart Weather Monitoring System provides several types of information:

1. Temperature and Humidity Alerts = Immediate updates on ambient temperature and humidity levels using the DHT11 sensor.
2. Rain Detection Alerts = Real-time alerts about the presence of rainfall detected by the rain sensor, allowing users to respond quickly to changing weather.
3. Air Quality Notifications = Alerts related to air pollution levels using the MQ-135 gas sensor, helping in early detection of unhealthy environmental conditions.
4. Historical Data and Trends = The Blynk cloud server stores historical sensor data, allowing users to analyze trends over time. This data can be used for predicting weather behavior, planning irrigation in agriculture, or assessing pollution levels.

### Software and Programming Language Used

The software used to develop the program for the "IoT-Based Smart Weather Monitoring System" is the Arduino IDE (Integrated Development Environment). The programming language used to write the program is the Arduino Programming Language, which is a simplified version of C/C++ tailored for programming Arduino- compatible microcontrollers, such as the ESP32 used in this project.

### Arduino IDE

*Features*

* + User-Friendly Interface: The Arduino IDE offers a simple and intuitive user interface that makes it easy for beginners and professionals alike to write, compile, and upload code to microcontrollers.
  + Built-in Libraries: The IDE includes numerous built-in libraries that simplify the integration of various sensors and modules, such as the DHT11 sensor, MQ-135 gas sensor, and Blynk communication library used in this project.
  + Serial Monitor: The IDE provides a serial monitor that allows users to communicate with the microcontroller in real-time, which is useful for debugging and monitoring sensor output directly from the ESP32.
  + Cross-Platform Support: The Arduino IDE is available for Windows, macOS, and Linux, making it accessible to users across different systems.
  + Open Source: The Arduino IDE is open-source software, meaning it is free to use and continuously improved by a large community of developers.

Applications in the Project

* + Code Development: The Arduino IDE is used to write the code that controls the ESP32 microcontroller. The code is written to interface with the DHT11 sensor for temperature and humidity monitoring, the MQ-135 sensor for air quality monitoring, and the rain sensor for detecting rainfall.
  + Code Compilation and Uploading: The IDE compiles the written code and uploads it to the ESP32 microcontroller via USB, enabling the ESP32 to execute the program.
  + Debugging: The serial monitor feature is used for debugging the program and checking the output of the sensors in real-time, allowing for adjustments and optimization of the code.

### Arduino Programming Language

*Features*

* + Simplified Syntax: The Arduino programming language uses a simplified syntax based on C/C++, making it easier for beginners to learn and write code for microcontrollers.
  + High-Level Abstractions: It provides high-level functions like digitalWrite(), digitalRead(), analogRead(), and analogWrite() to make interfacing with hardware components much easier.
  + Wide Community Support: The Arduino programming language has a large community of users and developers, offering extensive resources, tutorials, and example codes to support project development.
  + Flexible and Scalable: The language is flexible enough to handle simple tasks such as LED blinking to complex IoT systems for environmental monitoring.

### Applications in the Project

* + Sensor Data Collection: The Arduino code is written to collect data from the DHT11 temperature and humidity sensor, the MQ-135 gas sensor, and the rain sensor. Functions and libraries specific to each sensor are used to read data accurately.
  + Data Processing and Analysis: The program processes the collected sensor data to analyze the environmental conditions where the system is implemented. It checks whether the data is within predefined thresholds and responds accordingly.
  + Data Transmission to Cloud: The Arduino code includes functions to establish a Wi- Fi connection using the ESP32 and transmit the processed sensor data to the Blynk cloud server for remote monitoring and notifications.
  + Event Handling and Alerts: The code is written to handle different weather events, such as rainfall or air pollution, and it triggers actions such as activating a buzzer or sending a mobile notification to the user via the Blynk platform.

The Arduino IDE and the Arduino programming language play a crucial role in the development of the "IoT-Based Smart Weather Monitoring System". These tools provide an accessible and efficient environment for writing, debugging, and deploying code to the ESP32 microcontroller, enabling real-time monitoring and control of environmental parameters. The combination of these software tools ensures seamless integration of sensors and reliable communication with the Blynk cloud platform for effective IoT-based monitoring.

## Project Code

To create the "IoT-Based Smart Weather Monitoring System" with Blynk interfacing, we'll write a complete Arduino code that integrates the DHT11 sensor for monitoring temperature and humidity, the MQ-135 gas sensor for air quality measurement, and the rain sensor for detecting rainfall, using the ESP32 microcontroller. This code will also connect to the Blynk cloud platform to send real-time data and receive alerts.

### Prerequisites

1. Arduino IDE installed on your computer.
2. ESP32 Board Package installed in the Arduino IDE.
   * Add this to Preferences: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-

pages/package\_esp32\_index.json

1. Blynk Library installed in the Arduino IDE.
   * Install via: Sketch > Include Library > Manage Libraries > Search “Blynk” and install.
2. Blynk App set up on your mobile device with a project created that includes widgets for:

* Temperature
* Humidity
* Air Quality
* Rain Detection
* Notifications (optional)

### Required Libraries

* WiFi.h – for Wi-Fi connectivity with ESP32
* BlynkSimpleEsp32.h – for communication with the Blynk cloud
* DHT.h – for interfacing the DHT11 sensor

Note: The rain sensor and MQ-135 do not require any specific libraries. Their data will be read using analogRead() (for MQ-135) and digitalRead() or analogRead() (for rain sensor depending on model).

**CODE**

#define BLYNK\_TEMPLATE\_ID "TMPL37diIJvAO"

#define BLYNK\_TEMPLATE\_NAME "Smart Weather Monitoring System" #define BLYNK\_AUTH\_TOKEN "ps6xU0w293ccSUMKwWUJWAEtwB0A2ihj"

#include <WiFi.h>

#include <BlynkSimpleEsp32.h> #include "DHT.h"

#include <Wire.h>

#include <LiquidCrystal\_I2C.h"

// Replace with your actual credentials char auth[] = BLYNK\_AUTH\_TOKEN;

char ssid[] = "Aagaye Bheek Mangne"; char pass[] = "arrestedinbeauty";

// Sensor and output pins #define DHTPIN 4

#define DHTTYPE DHT11 #define MQ8PIN 35

#define RAINPIN 34

#define BUZZER 14

#define LED 26

DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal\_I2C lcd(0x27, 16, 2); // Confirmed I2C address 0x27 bool alertSent = false;

void setup() { Serial.begin(115200); Blynk.begin(auth, ssid, pass);

Wire.begin(21, 22); // Set I2C pins for ESP32 lcd.init(); // Initialize the LCD lcd.backlight(); // Turn on LCD backlight

dht.begin();

pinMode(BUZZER, OUTPUT); pinMode(LED, OUTPUT);

lcd.setCursor(0, 0); lcd.print("Weather Monitor");

delay(2000); lcd.clear();

}

void loop() { Blynk.run();

static unsigned long lastUpdate = 0; unsigned long now = millis();

if (now - lastUpdate >= 2000) { lastUpdate = now;

float temp = dht.readTemperature(); float hum = dht.readHumidity(); int gasValue = analogRead(MQ8PIN);

int rainValue = analogRead(RAINPIN);

if (!isnan(temp) && !isnan(hum)) { Blynk.virtualWrite(V1, (double)temp); Blynk.virtualWrite(V2, (double)hum); Blynk.virtualWrite(V3, gasValue); Blynk.virtualWrite(V4, rainValue);

}

lcd.setCursor(0, 0);

lcd.print("T:"); lcd.print(temp, 1);

lcd.print(" H:"); lcd.print(hum, 1); lcd.setCursor(0, 1); lcd.print("G:"); lcd.print(gasValue);

lcd.print(" R:"); lcd.print(rainValue);

if (gasValue > 2500 && !alertSent) { digitalWrite(BUZZER, HIGH); digitalWrite(LED, HIGH);

Blynk.logEvent("gas\_alert", "High gas level detected!"); alertSent = true;

}

else if (gasValue <= 2500) { digitalWrite(BUZZER, LOW); digitalWrite(LED, LOW); alertSent = false;

}

}}

**Result and Discussion**

The results demonstrate the effectiveness of the **IoT-Based Smart Weather Monitoring System** in accurately capturing and reporting real-time environmental data. The system successfully monitors **temperature, humidity, rainfall**, and **air quality**, providing continuous updates through the **Blynk cloud platform**.

The integration of **ESP32 with Blynk** enables seamless **remote access and monitoring** via smartphone, ensuring that users are informed about changing weather conditions even when they are away from the system's physical location. The system also effectively triggers **local alerts** using LEDs or buzzers when critical thresholds are crossed.

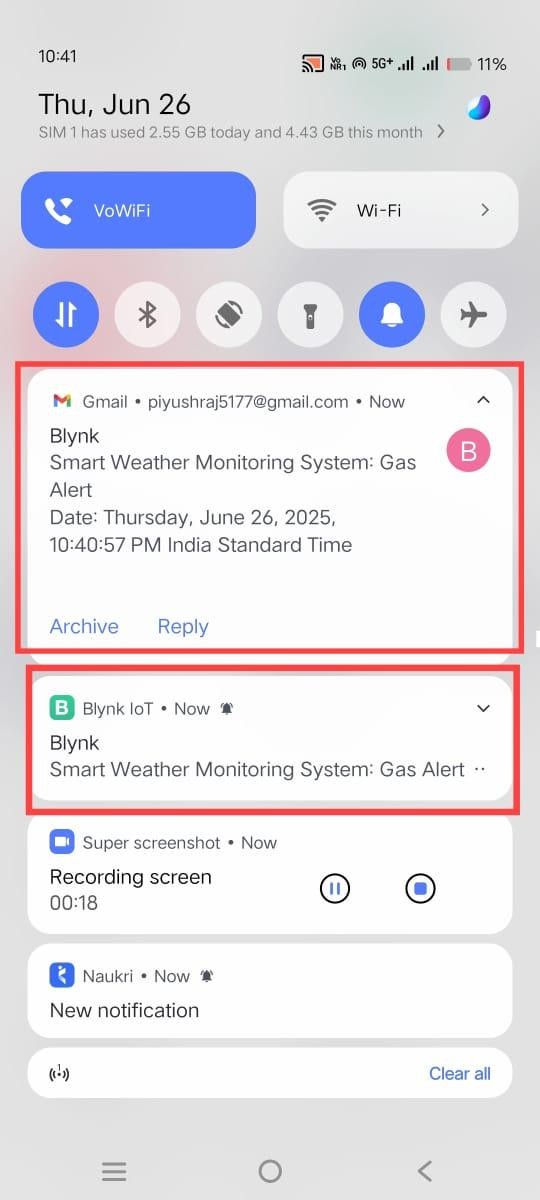
This real-time visibility into environmental parameters not only supports **decision- making** in applications like agriculture, home automation, and pollution control, but also lays a foundation for building smarter, responsive systems in the future. The project confirms that combining IoT sensors with cloud communication platforms like Blynk results in a **reliable, scalable, and user-friendly solution** for smart weather monitoring.

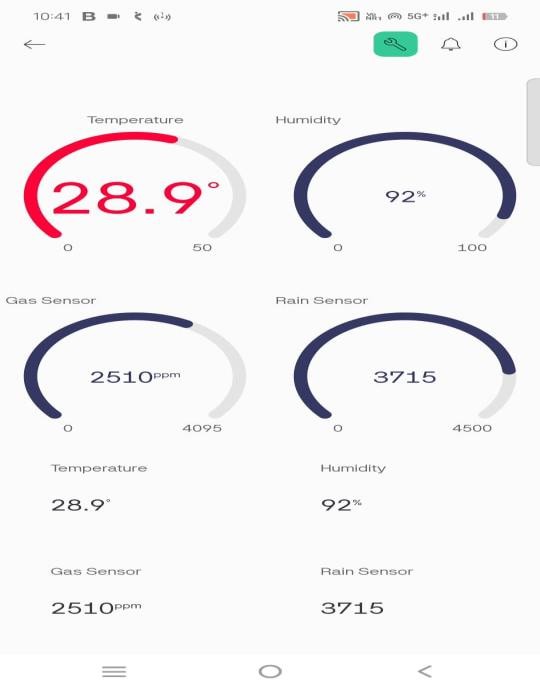
The system's **response time** is crucial for ensuring timely environmental awareness and accurate data-driven decisions. The **2–3 second response time** achieved in this project is considered satisfactory for real-time weather monitoring. However, further optimization can be explored to reduce latency, especially in applications requiring faster data refresh rates.

The **scalability** of the system makes it suitable for a wide range of environments, including **smart homes**, **agricultural fields**, **urban air quality zones**, and **remote weather stations**. The use of the **ESP32 microcontroller** along with the **Blynk cloud platform** offers a cost-effective, reliable, and efficient solution for environmental data collection and reporting.

The **IoT-Based Smart Weather Monitoring System**, utilizing **ESP32**, **DHT11**, **MQ-135**, **rain sensor**, and **Blynk**, provides a dependable and user-friendly platform for continuous weather and air quality observation. Its ability to monitor parameters, alert users in real-time, and support cloud-based historical data visualization highlights its practical usefulness. Future developments may focus on integrating **more precise sensors**, **solar-powered modules**, and **advanced data analytics** to further enhance performance, sustainability, and scalability.

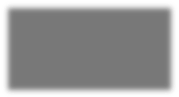
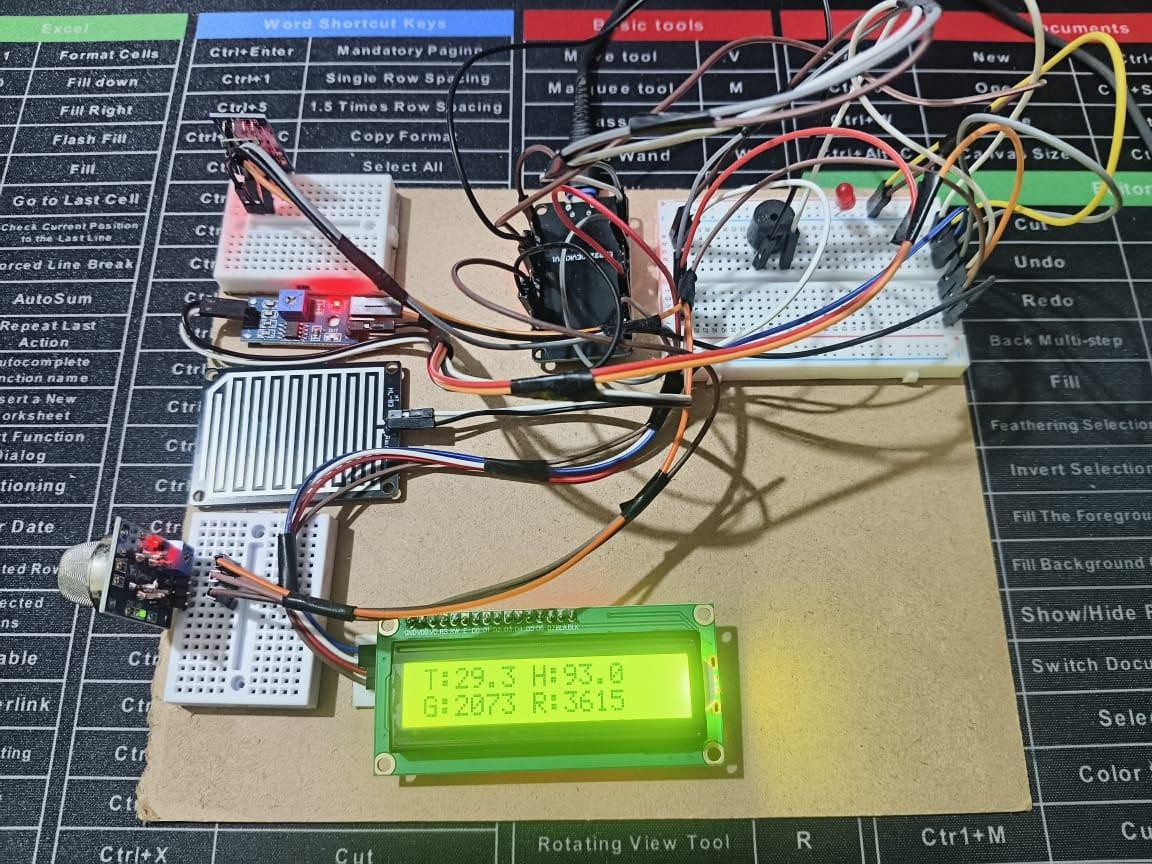
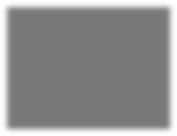
## PROTYPE CIRCUIT DIAGRAM



**Figure : Blynk App and Mobile Notification**

**Figure :** Blynk APP Mobile Dashboard

**Figure : Project Original Image PROTOTYPE MODEL**



**Figure : Blynk Iot Web Dashboard**

## Future Scope of the IoT-Based Smart Weather Monitoring System

The future of IoT-based smart weather monitoring systems is extremely promising, with advancements focused on improving accuracy, real-time analytics, environmental adaptability, and decision-making support. Here's a breakdown of the key areas of future scope:

1. Enhanced Sensor Technology:
   * Development of high-precision sensors for monitoring a broader range of environmental parameters like wind speed, barometric pressure, and UV index.
   * Integration of multifunctional sensors that combine temperature, humidity, gas detection, and particulate matter monitoring into a single unit.
   * Use of nanotechnology-based sensors for ultra-low-power and miniature weather stations.
2. Artificial Intelligence (AI) and Machine Learning (ML) Integration:
   * AI-based algorithms for weather pattern recognition, enabling predictive analytics for rainfall, pollution spikes, and heatwaves.
   * ML models that learn environmental trends over time to suggest optimal actions (e.g., for smart irrigation or pollution control).
   * Anomaly detection for identifying abnormal changes in temperature or humidity that could indicate equipment failure or sudden climatic shifts.
3. Improved Connectivity and Interoperability:
   * Adoption of next-generation IoT communication protocols (like LoRaWAN, NB- IoT) for wider range and lower power consumption in remote areas.
   * Seamless integration with smart city infrastructure, allowing weather data to support traffic control, public health systems, and disaster readiness.
   * Cloud platforms with multi-device compatibility for managing large-scale weather monitoring networks from a centralized dashboard.
4. Advanced Environmental Response Systems:
   * Automated environmental control systems that adjust fans, shades, irrigation, or ventilation based on real-time weather conditions.
   * Solar-powered and weatherproof enclosures to allow 24/7 operation in outdoor or harsh environments.
   * Use of drones or autonomous vehicles equipped with sensors to gather weather data from inaccessible locations.
5. Real-Time Monitoring and Alerting Enhancements:
   * Faster and more intuitive data visualization dashboards using mobile and web applications.
   * Integration with emergency alert systems for sending real-time notifications about extreme weather to authorities or local populations.
   * Voice assistant integration (Alexa, Google Assistant) for voice-based access to live environmental data.
6. Greater Focus on Preventative and Sustainability Measures:
   * Predictive insights that help prevent crop loss due to drought, frost, or unseasonal rainfall.
   * Monitoring environmental factors to control disease outbreaks, particularly those linked to air quality or humidity.
   * Support for climate research and sustainability planning through long-term data logging and AI-driven environmental modeling.

## Conclusion

The future scope of IoT-based smart weather monitoring systems reflects a transition toward more intelligent, connected, and proactive solutions. These systems are no longer just tools for observing environmental conditions but are becoming central components in smart agriculture, smart cities, and sustainable living.

By incorporating advanced sensors, AI-based analytics, real-time cloud access, and automated responses, the next generation of weather systems will offer deeper insights, faster reaction times, and greater utility across various sectors. As IoT technology continues to evolve, these systems will play a critical role in climate adaptation, disaster prevention, environmental awareness, and sustainable development, ensuring a smarter and safer future for all.

## References

### Arduino Documentation

Arduino Team. (n.d.). *Arduino Language Reference*. Retrieved from <https://www.arduino.cc/reference/en/>

This reference provides a comprehensive guide to the Arduino programming language, syntax, functions, and libraries, which are crucial for programming the **ESP32 microcontroller** used in this project.

### Blynk Documentation

Blynk IoT Platform. (n.d.). *Blynk Documentation*. Retrieved from <https://docs.blynk.io/> The Blynk documentation offers detailed information on setting up the Blynk platform, interfacing with hardware like the **ESP32**, and using the Blynk app for real-time **IoT monitoring and control**.

### DHT Sensor Library Documentation

Adafruit Industries. (n.d.). *DHT Sensor Library Documentation*. Retrieved from <https://github.com/adafruit/DHT-sensor-library>

This resource was used for understanding and integrating the **DHT11 temperature and humidity sensor** with the ESP32 through Arduino IDE.

### MQ-135 Gas Sensor Datasheet

Winsen Electronics. (n.d.). *MQ-135 Gas Sensor Datasheet*. Retrieved from [https://www.winsen-sensor.com](https://www.winsen-sensor.com/)

The official datasheet provides technical details required for using the **MQ-135 sensor**

for air quality monitoring.

### Rain Sensor Module Guide

Circuit Digest. (n.d.). *Rain Sensor Module Interfacing with Arduino*. Retrieved from https://circuitdigest.com

This guide helped understand the working and interfacing of the **Rain Detection Module** with microcontrollers.

Internet Engineering Task Force (IETF). (2021). IOT Security: State of the Art and Challenges. Retrieved from https://datatracker.ietf.org/doc/html/draft-ietf-iot-security- best practices-01

This document provides guidelines and best practices for securing IOT systems against cyber threats, which is relevant for ensuring the security of the IOT Based Smart Smoke and Fire Detection and Prevention system.