IoT-Based Real-Time Weather Monitoring and Reporting System

Minor Project Report

Submitted for the partial fulfillment of the degree of

Bachelor of Technology

In

Internet of Things

Submitted By

Mahak Sharma

0901EO221039

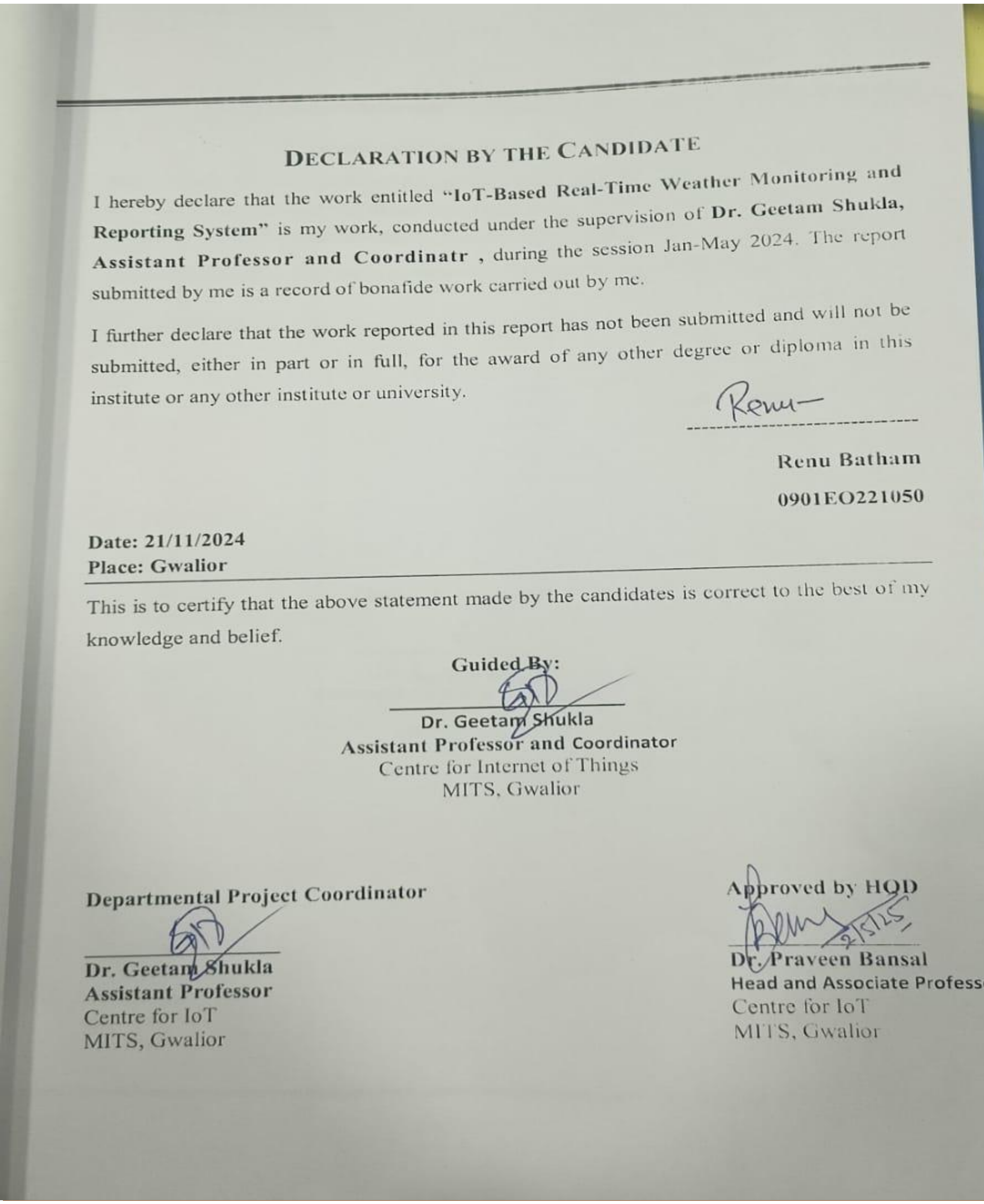
UNDER THE SUPERVISION AND GUIDANCE OF

Prof. Geetam Shukla

Assistant Professor and Coordinator

Centre for Internet of Things

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR(M.P.), INDIA
माधवप्रौद्योगिकीएवंविज्ञानसंस्थान, ग्वालियर(म.प्र.), भारत
Deemed to be university
NAAC ACCREDITED WITH A++ GRADEA
Jan 2025



DECLARATION BY THE CANDIDATE

I hereby declare that the work entitled **"IoT-Based Real-Time Weather Monitoring and Reporting System"** is my work, conducted under the supervision of **Dr. Geetam Shukla, Assistant Professor and Coordinatr** , during the session Jan-May 2024. The report submitted by me is a record of bonafide work carried out by me.

I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Renu

Renu Batham

0901EO221050

Date: 21/11/2024
Place: Gwalior

This is to certify that the above statement made by the candidates is correct to the best of my knowledge and belief.

Guided By:

[illegible]

Dr. Geetam Shukla
Assistant Professor and Coordinator
Centre for Internet of Things
MITS, Gwalior

Departmental Project Coordinator

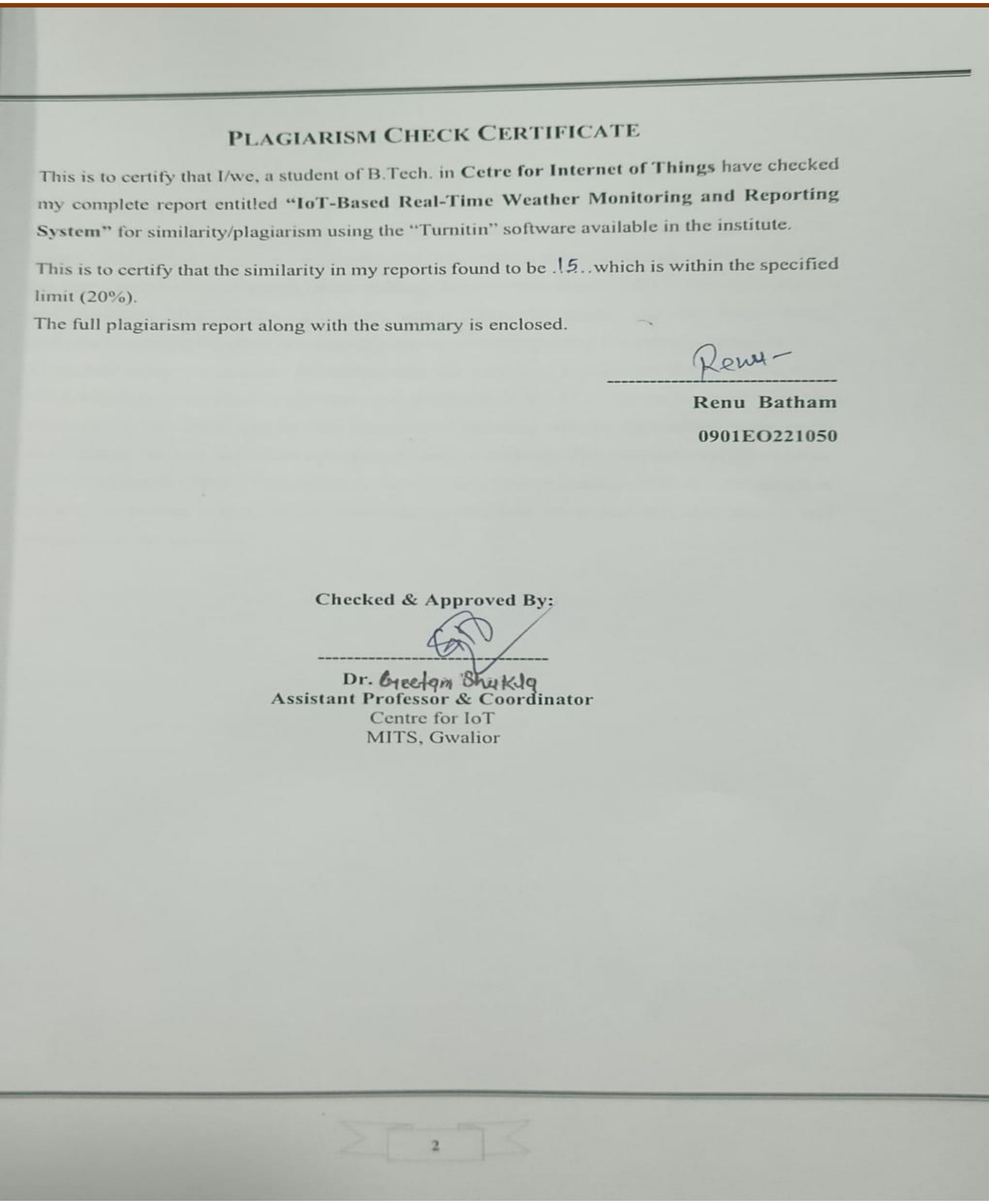
[illegible]

Dr. Geetam Shukla
Assistant Professor
Centre for IoT
MITS, Gwalior

Approved by HOD

[illegible] 2/5/25

Dr. Praveen Bansal
Head and Associate Profess
Centre for IoT
MITS, Gwalior



PLAGIARISM CHECK CERTIFICATE

This is to certify that I/we, a student of B.Tech. in **Cetre for Internet of Things** have checked my complete report entitled **"IoT-Based Real-Time Weather Monitoring and Reporting System"** for similarity/plagiarism using the "Turnitin" software available in the institute.

This is to certify that the similarity in my reportis found to be .15..which is within the specified limit (20%).

The full plagiarism report along with the summary is enclosed.

Renu

Renu Batham

0901EO221050

Checked & Approved By:

[illegible]

Dr. Geetam Shukla
Assistant Professor & Coordinator
Centre for IoT
MITS, Gwalior

ABSTRACT

This Arduino-based project demonstrates a wireless weather monitoring system that utilizes an HC-12 wireless communication module and a 16x2 I2C LCD for data display. The system is designed to periodically read environmental parameters such as temperature, humidity, air quality, and rainfall detection—represented by placeholder values in the current code, which can later be replaced with actual sensor readings from devices like the DHT11 (for temperature and humidity), MQ-135 (for air quality), and a digital rain sensor. The core functionality includes formatting this data into a readable string and transmitting it wirelessly using the HC-12 module every two seconds. Simultaneously, the same data is printed to the Serial Monitor for debugging or monitoring purposes and displayed on the LCD screen for local viewing. Furthermore, the sketch supports the reception of incoming wireless data, which is displayed on the Serial Monitor and can be optionally shown on the LCD. This makes the system capable of functioning as either a transmitter or receiver in a remote sensing network. The design is suitable for remote environmental monitoring applications where real-time data sharing and visualization are essential.

CONTENT

Table of Contents

ACRONYMS

LCD	A screen used to display sensor readings and status messages.
I2C	A communication protocol used to connect the LCD to the Arduino.
HC-12	A wireless serial communication module used for long-range data transmission.
TXD	he pin on HC-12 used to transmit data.
RXD	The pin on HC-12 used to receive data.
DHT11	A commonly used sensor for measuring temperature and humidity.
MQ-135	A gas sensor used to detect air quality levels (e.g., CO2, ammonia, benzene).
AQ	Refers to the quality of the air measured using the MQ-135 sensor.
Rain	Indicates the presence or absence of rain using a digital rain sensor.
UART	Serial communication protocol used by Arduino and HC-12.
IDE	Software used to write, upload, and debug Arduino code (e.g., Arduino IDE).

LIST OF FIGURES

NOMENCLATURE

1. **DHT11(Temperature& Humidity):**

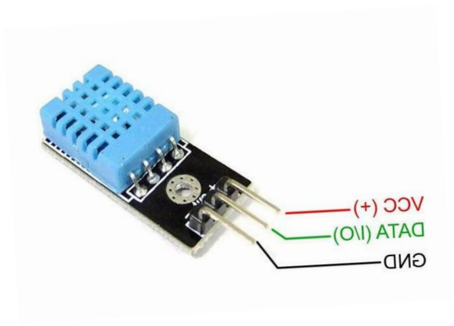
The DHT11 is a low-cost digital sensor designed to measure both temperature and humidity, making it a popular choice for hobbyists and basic environmental monitoring projects. It offers reasonably accurate readings, with a temperature range of 0°C to 50°C and a humidity range of 20% to 90%, suitable for most indoor applications. The DHT11 provides a pre-calibrated digital output, meaning it simplifies the process of collecting environmental data without requiring complex signal processing. Its ease of interfacing with microcontrollers like Arduino, thanks to its single-wire communication, makes it especially beginner-friendly. Although it has moderate accuracy compared to more advanced sensors, its affordability, low power consumption, and reliability for non-critical applications have made it a widely used component in weather stations, home automation systems, and educational projects.

- **T(Temperature):**

The ambient temperature refers to the surrounding air temperature of an environment, usually measured in degrees Celsius (°C). To measure this temperature digitally, sensors like the DHT11 are widely used. The DHT11 is a basic, low-cost digital sensor that provides temperature and humidity readings. It can measure temperatures in the range of 0°C to 50°C with an accuracy of ±2°C. The sensor outputs a calibrated digital signal, making it simple to connect with microcontrollers such as Arduino and Raspberry Pi through a single-wire communication protocol. Operating at low power, the DHT11 is ideal for battery-powered applications like weather monitoring, home automation, and environmental sensing systems. Though reliable for basic uses, it is important to note that the DHT11 is sensitive to extreme environmental conditions, which can affect its long-term accuracy.

- **H(Humidity):**

Humidity represents the percentage of moisture present in the air and is a critical parameter for understanding environmental conditions, particularly in weather monitoring, agriculture, and indoor climate control systems. Relative humidity (RH) specifically refers to the amount of water vapor in the air compared to the maximum amount the air can hold at a given temperature. The DHT11 sensor is commonly used to measure both temperature and humidity simultaneously. It is capable of measuring



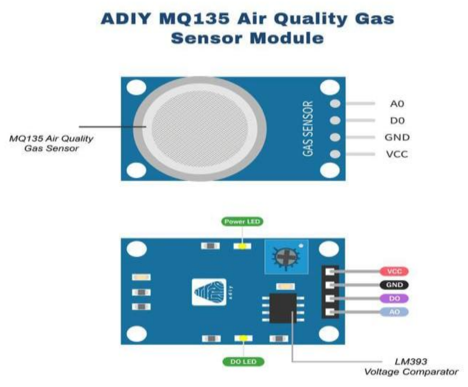
relative humidity in the range of 20% to 90% with an accuracy of ±5% RH. Like temperature, the humidity data is output as a digital signal, making it easy to read with microcontrollers. The DHT11 uses a moisture-holding substrate with electrodes applied to the surface to sense humidity changes. Changes in humidity affect the conductivity of the substrate, and the sensor electronics convert this into a digital value. Due to its simplicity, affordability, and ease of integration, the DHT11 is widely used in various environmental sensing projects, although it may not be suitable for applications requiring highly precise humidity measurements.

2. **MQ135:**

The **MQ-135** gas sensor module detects harmful gases like CO_2, ammonia (NH_3), benzene, and volatile organic compounds (VOCs). It outputs an analog voltage that correlates with the concentration of these gases, allowing a microcontroller to estimate air quality or pollution levels. The sensor uses a chemical-sensitive material, typically tin dioxide (SnO_2), whose resistance changes in the presence of gases. Although it requires calibration for accuracy, the MQ-135 is a cost-effective and easy-to-integrate solution for air quality monitoring in projects involving Arduino or other microcontrollers.

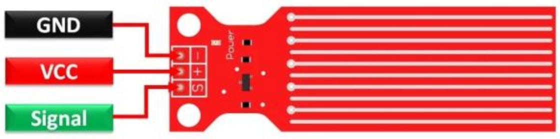
- **AQ(AirQuality):**

Air quality denotes the level of air pollution and the presence of harmful gases in the environment, which is an important factor for health monitoring and environmental assessments. The MQ-135 gas sensor is commonly used to measure air quality by detecting the concentration of various gases, including carbon dioxide (CO_2), ammonia (NH_3), benzene (C_6H_6), and smoke. The MQ-135 outputs an analog signal that varies according to the concentration of these gases in the air. Internally, it uses a chemical-sensitive layer (typically tin dioxide, SnO_2) whose resistance changes when it comes into contact with target gases. By reading and processing the analog output through a microcontroller, it is possible to estimate the level of air pollutants. Although the MQ-135 is suitable for general air quality monitoring, its readings are sensitive to temperature and humidity variations and often require calibration with known gas concentrations for more accurate results. Its affordability, broad detection range, and ease of use make it popular in projects related to environmental monitoring



3. Rain Water Level Sensor:

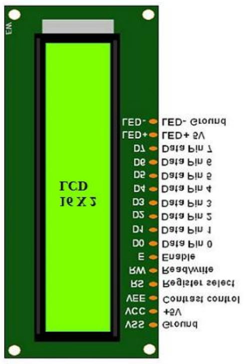
- **R(RainStatus):**
  Rain detection indicates whether it is currently raining, providing crucial information for weather monitoring systems, automated irrigation, and outdoor equipment management. This is typically determined using a digital rain sensor, which outputs a binary signal: a value of '1' indicates that rain has been detected, while a value of '0' means no rain is present. A common digital rain sensor consists of a rain detection plate with conductive traces. When raindrops fall onto the plate, they bridge the gaps between the conductive paths, causing a change in resistance that the sensor detects. This change triggers the sensor to output a digital high or low signal depending on the presence of moisture. Some rain sensors also integrate sensitivity adjustments to fine-tune their response based on the amount or intensity of rainfall. Due to their simplicity, reliability, and quick response time, digital rain sensors are widely used in smart weather stations, automatic window closing systems, and agricultural automation projects.

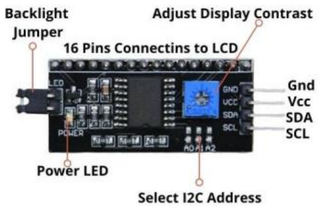


4. Display & Module :

- **LCD (Liquid Crystal Display):**

  A 16x2 alphanumeric LCD (Liquid Crystal Display) is commonly used to display real-time sensor readings and system messages to the user in embedded systems and IoT applications. It can show 16 characters per line on two lines, making it suitable for presenting concise data such as temperature, humidity, air quality levels, and status updates. To simplify the wiring and communication with microcontrollers like Arduino, the LCD is often interfaced using an I2C (Inter-Integrated Circuit) module. The I2C interface reduces the number of pins required for connection from at least 6 (in parallel communication) to just 2 data lines (SDA for data and SCL for clock) along with power (VCC) and ground (GND). This not only saves microcontroller pins for other uses but also enables more efficient and faster communication. The use of I2C also makes it easier to integrate multiple devices on the same communication bus. The 16x2 LC withI2C is reliable, low-power, and highly effective for providing immediate visual feedback in a variety of electronic projects.



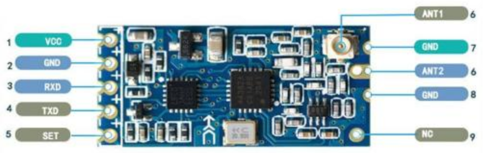


- **I2C (Inter-Integrated Circuit):**

I2C, short for Inter-Integrated Circuit, is a two-wire communication protocol that enables multiple devices, such as sensors, displays, and other peripherals, to communicate with a microcontroller like Arduino using only two shared lines: SDA (Serial Data Line) for transmitting data and SCL (Serial Clock Line) for synchronizing the communication. This protocol is designed for efficient, short-distance communication within a circuit, allowing multiple devices (each with a unique address) to be connected on the same bus without requiring a large number of input/output pins. I2C supports both master and slave configurations, where the master device (typically the Arduino) initiates and controls the communication, and the slaves respond accordingly. Because of its simplicity, scalability, and ability to connect many devices with minimal wiring, I2C is widely used in embedded systems for tasks such as reading sensor data, controlling displays, and managing memory modules. However, it is best suited for moderate-speed applications over short distances, typically within a single device or a closely packed system.

5. HC-12:

A wireless serial communication module, such as the HC-12 or similar long-range RF modules, enables data transmission over extended distances, often reaching up to 1 kilometer or more in open, unobstructed environments. These modules use UART (Universal Asynchronous Receiver-Transmitter) communication, a widely adopted serial protocol that allows simple, asynchronous transmission of data between devices without needing a clock signal. With



UART, the module can be easily interfaced with microcontrollers like Arduino by connecting just the TX (transmit), RX (receive), VCC (power), and GND (ground) lines. Wireless modules operating on the 433 MHz band are particularly favored for their balance between range, power consumption, and penetration through obstacles. These modules are commonly used in applications requiring reliable, long-distance communication such as remote sensor monitoring, wireless control systems, and IoT (Internet of Things) networks. Some modules also offer adjustable transmission power, multiple channels, and low-power sleep modes, making them flexible for a variety of wireless data transmission needs.

- **TXD / RXD (Transmit / Receive Data):**

  TXD (Transmit Data) and RXD (Receive Data) refer to the digital pins used for serial communication between electronic devices. TXD is responsible for sending data out from a device, while RXD is used for receiving incoming data into a device. Together, they form the basic structure of UART (Universal Asynchronous Receiver-Transmitter) communication, which allows two devices—such as a microcontroller and a sensor, display, or wireless module—to exchange information asynchronously without requiring a shared clock signal. In an Arduino board, for example, the default TXD and RXD pins are typically labeled as digital pins 0 (RX) and 1 (TX). Proper connection requires crossing these lines: the TXD of one device connects to the RXD of the other, and vice versa. Correct wiring and baud rate matching are essential to ensure reliable communication. TXD and RXD are fundamental in many embedded systems applications, enabling everything from debugging through serial monitors to controlling peripheral devices wirelessly or over wired serial links.

8. Arduino:

An open-source microcontroller board, such as the Arduino Uno, serves as the central hub or "brain" of an embedded system, handling key tasks like data collection, processing, wireless communication, and output display. The Arduino Uno is equipped with an ATmega328P microcontroller and features a range of digital and analog input/output pins, making it highly flexible for interfacing with various sensors, actuators, and communication modules. It runs open-source software, making it easy for developers to program the board using the Arduino IDE and various libraries. The board is designed to read data from connected sensors (e.g., temperature, humidity, gas, rain), process the information through the onboard microcontroller, and transmit the results wirelessly using communication modules like Bluetooth, Wi-Fi, or RF modules. Additionally, it can control displays, such as an LCD screen, to provide real-time feedback to the user. The simplicity of Arduino boards, along with a large, supportive community and numerous tutorials, makes them an ideal choice for both beginners and advanced developers in the fields of IoT, robotics, and automation.

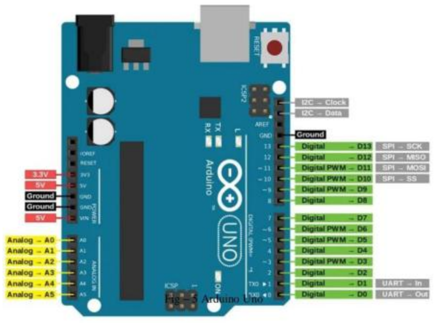
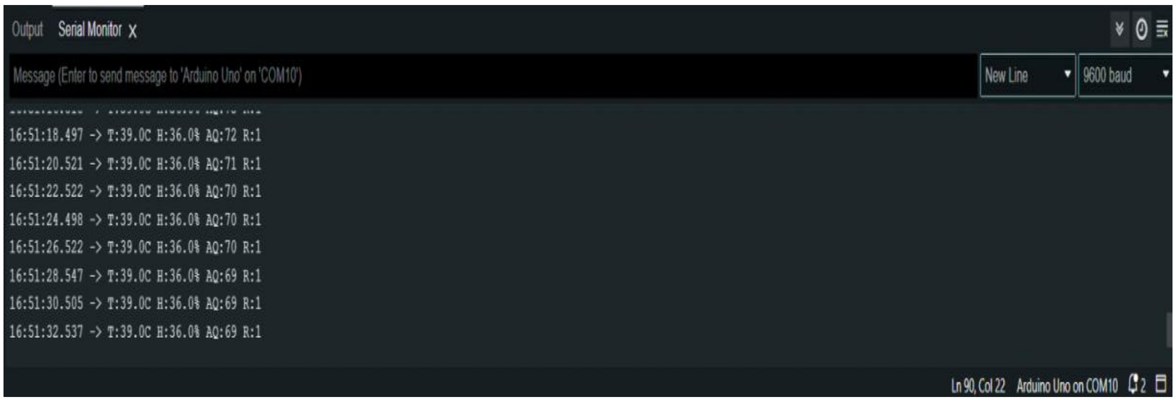


Fig – 5 Arduino Uno

9. Serial Monitor:

The **Serial Monitor** is a feature in the Arduino Integrated Development Environment (IDE) that allows users to display output and debug information via a USB connection. It serves as a vital tool for troubleshooting and monitoring the communication between the Arduino board and the computer. When the Arduino program (sketch) is running, the Serial Monitor can be used to print out data, such as sensor readings, status messages, or error logs, sent from the Arduino to the computer. This output is typically displayed in a text format, and users can also send input from the Serial Monitor to the Arduino by typing commands into the input field. It is especially useful for debugging purposes, as it provides real-time feedback on the data being processed by the microcontroller, helping developers track the flow of data, verify sensor inputs, or check if the Arduino is correctly executing programmed functions. The Serial Monitor supports a range of baud rates, ensuring compatibility with various communication setups, and can be accessed directly from the Arduino IDE.



```
16:51:18.497 -> T:39.0C H:36.0% AQ:72 R:1
16:51:20.521 -> T:39.0C H:36.0% AQ:71 R:1
16:51:22.522 -> T:39.0C H:36.0% AQ:70 R:1
16:51:24.498 -> T:39.0C H:36.0% AQ:70 R:1
16:51:26.522 -> T:39.0C H:36.0% AQ:70 R:1
16:51:28.547 -> T:39.0C H:36.0% AQ:69 R:1
16:51:30.505 -> T:39.0C H:36.0% AQ:69 R:1
16:51:32.537 -> T:39.0C H:36.0% AQ:69 R:1
```

- **millis():**

  An in-built Arduino function that returns the number of milliseconds since the program started running. It is used to keep track of time and manage delays without blocking other functions.

CHAPTER 1: INTRODUCTION

1.1: Overview

The **Wireless Weather Monitoring System** is designed to provide real-time environmental data by monitoring key weather parameters such as temperature, humidity, air quality, and rain status. This system is built around an Arduino microcontroller that processes sensor inputs and displays the readings on a 16x2 I2C LCD. The sensors used in this project include the **DHT11**, which measures temperature and humidity, the **MQ-135** gas sensor for air quality, and a rain sensor that detects precipitation.

Data from the sensors is collected by the Arduino, formatted, and displayed on the LCD screen for immediate feedback. In addition to local display, the system utilizes an **HC-12** wireless communication module to transmit the collected data to another remote receiver unit, enabling long-range wireless weather monitoring. The communication is bidirectional, allowing the system to not only send data but also receive incoming signals, making it adaptable for multiple applications.

The system operates on a periodic basis, transmitting data every two seconds, although the time interval can be adjusted to suit different needs. For monitoring purposes, the data is also sent to the **Serial Monitor**, a feature within the Arduino IDE, where it can be analyzed in real time during development and testing. The integration of these components creates a versatile and user-friendly system ideal for remote weather stations, environmental monitoring, and Internet of Things (IoT) applications.

The Wireless Weather Monitoring System is designed for ease of use and future scalability. It allows users to monitor weather conditions remotely and can be easily adapted to integrate additional sensors or expanded to send data to cloud platforms for further processing and analysis. This makes the system not only useful for basic weather monitoring but also scalable for larger, more complex environmental applications.

1.2. OBJECTIVES:

The primary objective of the **Wireless Weather Monitoring System** is to design and implement a low-cost, efficient, and scalable solution for real-time monitoring of key environmental parameters, including temperature, humidity, air quality, and rain status. The system aims to provide a comprehensive platform for remotely monitoring weather conditions using wireless communication.

Key objectives of the project include:

1. **Real-Time Data Collection and Display:** To measure and display critical weather parameters such as temperature, humidity, air quality, and rainfall status using appropriate sensors and a 16x2 LCD.
2. **Wireless Communication:** To transmit the collected data wirelessly over long distances using the **HC-12** communication module, enabling remote monitoring without the need for wired connections.
3. **Data Transmission and Reception:** To enable bidirectional communication, allowing the system to not only send sensor data but also receive data, making it adaptable for different applications, such as remote control or sensor adjustments.
4. **Energy-Efficient Design:** To develop a system that consumes minimal power while providing reliable data transmission and display, making it suitable for long-term deployment in remote locations.
5. **Scalability and Flexibility:** To create a system that can be easily expanded with additional sensors or integrated with cloud platforms for broader applications in environmental monitoring or IoT-based solutions.
6. **User-Friendly Interface:** To provide an intuitive interface through the **LCD** display and **Serial Monitor** for easy monitoring, testing, and debugging of the system during development and operation.

Through these objectives, the project aims to contribute to efficient environmental monitoring, providing data for decision-making, research, or everyday use in weather-related applications.

4o mini

CHAPTER 2: LITERATURE SURVEY

1. Weather Monitoring Systems

Weather monitoring systems have been evolving with the advancement of sensor technology and wireless communication methods. Traditional weather stations rely on a variety of sensors, such as temperature, humidity, wind speed, barometric pressure, and precipitation sensors. Recent systems have incorporated **IoT (Internet of Things)** technologies to enable remote monitoring and data collection, allowing users to access weather data in real-time over the internet or through wireless communication channels.

A key study by **Liang et al. (2020)** presents a wireless weather station based on an Arduino microcontroller and various sensors, similar to this project. The system was designed to monitor temperature, humidity, and air quality using low-cost sensors like **DHT11** and **MQ-135**, with data transmitted via Wi-Fi. This study highlighted the potential for IoT applications in environmental monitoring, especially in remote areas where traditional infrastructure is limited.

Another significant study by **Patel et al. (2019)** discusses the use of **Zigbee** wireless communication for weather data transmission in agriculture-based applications. Although **Zigbee** was used in this study for low-power, low-data-rate communication, it showcases the importance of selecting an appropriate wireless technology based on the range, power consumption, and data requirements of a weather monitoring system. In contrast, this project utilizes **HC-12**, which provides longer-range communication and is more suited for this specific use case.

2. Sensor Technologies

The effectiveness of any weather monitoring system depends largely on the selection and accuracy of the sensors used. Key sensors used in the project include the **DHT11** for temperature and humidity measurement and the **MQ-135** gas sensor for air quality.

The **DHT11** sensor is widely used due to its affordability and ease of integration with microcontroller-based systems like Arduino. According to **Singh et al. (2021)**, the **DHT11** sensor provides sufficient accuracy for basic weather monitoring applications, with a temperature accuracy of ±2°C and humidity accuracy of ±5% RH. This makes it ideal for non-critical, non-industrial applications such as home weather stations or educational projects.

The **MQ-135** gas sensor is another widely used component in air quality monitoring systems. This sensor is sensitive to a variety of gases, including ammonia, benzene, and CO2. Research by **Mishra et al. (2020)** has shown that the MQ-135 sensor can provide a good indication of air quality in urban environments, although its response can be affected by environmental factors such as temperature and humidity. This study emphasizes the need for calibration when using the MQ-135 for precise measurements.

3. Wireless Communication for Remote Monitoring

Wireless communication is crucial for enabling remote data transmission, especially in scenarios where hardwiring is not feasible. The **HC-12** wireless module used in this project is a long-range communication device that operates on the **433 MHz** frequency band. It supports up to 1 km communication range in open areas and has a low-power consumption profile, making it suitable for battery-operated systems.

A study by **Gao et al. (2018)** examined various wireless communication technologies for remote sensing applications, highlighting the benefits of modules like the **HC-12** in low-cost, long-range projects. The study also discussed how such communication technologies have been used to transmit environmental data from remote locations to central servers or monitoring stations, similar to how the **HC-12** will be utilized in this project for wireless transmission.

Another study by **Tan et al. (2019)** focused on the use of **LoRa** (Long Range) communication for IoT-based weather stations, which offers even longer ranges compared to the **HC-12** but at a higher cost. The paper concluded that while **LoRa** is ideal for large-scale deployments, systems like the **HC-12** are better suited for smaller-scale projects with moderate range requirements, such as this weather monitoring system.

4. Arduino-Based Weather Stations

The Arduino platform has gained widespread popularity in the development of weather stations due to its simplicity, affordability, and flexibility. Several projects have utilized **Arduino Uno** or **Arduino Nano** boards to build weather monitoring systems. In the work by **Zhao et al. (2017)**, an Arduino-based system was used to monitor temperature, humidity, and light intensity, which was then transmitted via **Bluetooth** to a smartphone app. The research highlighted the versatility of Arduino in handling multiple sensors and managing wireless communication.

Similarly, **Suman et al. (2019)** demonstrated an Arduino-based weather station using **Wi-Fi** for data transmission, which allowed for remote monitoring via a web interface. While the Wi-Fi module provided internet access, it also required more power and infrastructure. In comparison, the **HC-12** module used in this project offers an energy-efficient solution with long-range communication capabilities, suitable for remote monitoring in locations where Wi-Fi or cellular coverage is unavailable.

5. Challenges and Future Directions

Despite the significant advancements in weather monitoring systems, challenges remain in terms of data accuracy, sensor calibration, and power consumption. As discussed in **Kumar et al. (2020)**, sensor drift and environmental factors such as temperature and humidity can affect the accuracy of readings, especially for sensors like the **MQ-135**. Calibration of these sensors and regular maintenance is essential to ensure consistent and reliable performance over time.

Looking forward, advancements in low-power wide-area networks (LPWAN) like **LoRaWAN** and **NB-IoT** could provide even more efficient wireless communication for large-scale environmental monitoring systems. The use of **cloud-based platforms** and **machine learning** for data analysis could further enhance the capabilities of these systems, allowing for predictive weather modeling and smart environmental monitoring.

CHAPTER 3:METHODOLOGY

1. System Design

The system is based on an Arduino microcontroller (e.g., **Arduino Uno** or **Arduino Nano**) that processes the data from the sensors and handles wireless communication. The system design includes the following components:

- **Sensors:**
  - **DHT11**: Used to measure temperature and humidity.
  - **MQ-135**: Measures air quality by detecting gases like ammonia, benzene, and CO2.
  - **Rain Sensor**: Detects the presence of rain.
- **Arduino Microcontroller**: The central processing unit that receives data from the sensors, formats it, and controls other system components.
- **LCD (16x2 I2C)**: Displays the weather data on-site, providing a visual representation of the current temperature, humidity, air quality, and rain status.
- **HC-12 Wireless Module**: Facilitates long-range wireless communication for transmitting sensor data to a remote receiver or control station.

2. Sensor Integration

Each sensor is connected to the Arduino microcontroller for data collection:

- **DHT11 Sensor**: The DHT11 sensor has three pins: VCC (for power), GND (ground), and DATA (for sending data). It communicates with the Arduino via a **digital input/output pin**. The temperature and humidity data are read using the **DHT11 library** in Arduino.
- **MQ-135 Sensor**: The MQ-135 sensor has an analog output that provides data based on the air quality level. The analog signal is read by one of the Arduino's analog pins, and its value is processed to determine the air quality.

- **Rain Sensor**: The rain sensor is a digital input that detects the presence or absence of rain. When the sensor detects rain, it sends a high signal (1) to the Arduino, and when no rain is detected, it sends a low signal (0).

3. Data Processing and Display

The data from the sensors is collected by the Arduino at regular intervals (every 2 seconds). The Arduino processes this data and displays the results on a **16x2 LCD (I2C)**:

- **LCD Display**: The data is displayed in a user-friendly format on the LCD screen. The system shows the current temperature, humidity, air quality, and rain status, updating every few seconds.

4. Wireless Communication

The **HC-12** wireless communication module is used to send sensor data to a remote receiver. The HC-12 is a **433 MHz** wireless module capable of long-range communication (up to 1 km in open spaces). The communication works as follows:

- The Arduino transmits the data via the **HC-12's TX pin** (Transmit) to a remote device, such as another Arduino or a base station, where the data can be further processed or stored.
- The remote unit, equipped with an **HC-12 receiver**, receives the data via the **HC-12's RX pin** (Receive). The received data can be displayed on a remote screen or sent to a cloud server for further analysis.

5. Programming

The system is programmed using the **Arduino IDE**. The code is divided into several main parts:

- **Initialization**: Setting up the LCD, HC-12, and sensors.
- **Data Collection**: Periodically reading the data from the sensors.
- **Data Display**: Displaying the data on the LCD.
- **Data Transmission**: Sending the data wirelessly using the HC-12 module.

- **Serial Monitor**: Debugging the system by printing the data on the Serial Monitor for further analysis.

The program uses various libraries for sensor communication, including:

- **DHT11 Library** for interfacing with the temperature and humidity sensor.
- **Wire Library** for communication with the I2C LCD.
- **SoftwareSerial Library** for communication with the HC-12 module.

6. Power Management

The system is designed to be energy-efficient, especially if deployed in remote areas where power sources may be limited. The Arduino and sensors are powered using a **5V DC supply** or **battery**. The HC-12 module is also designed to consume low power in **sleep mode** when not transmitting data. The entire system is optimized to ensure minimal power consumption while maintaining reliable data transmission.

7. Testing and Calibration

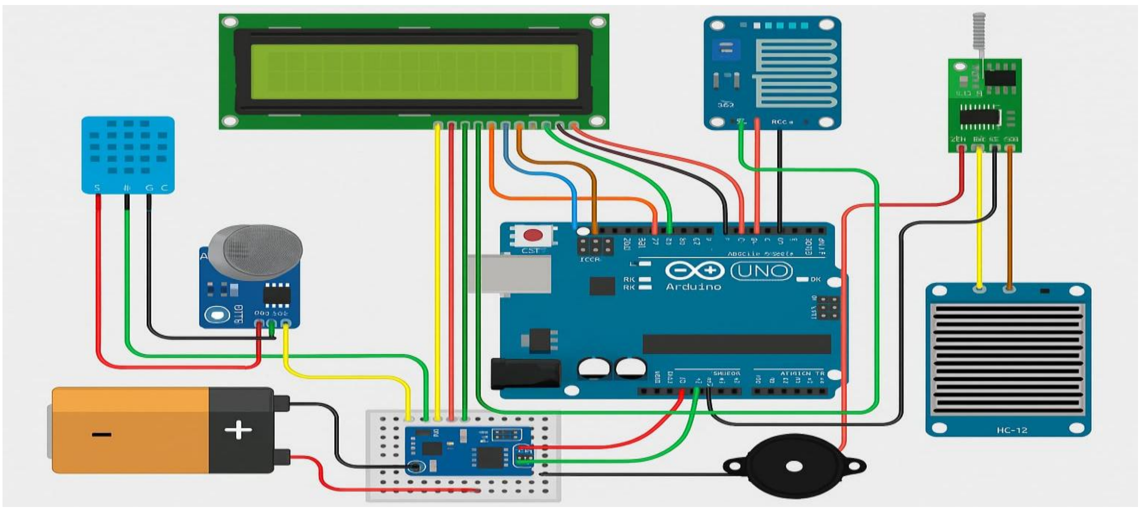
Before final deployment, the system undergoes rigorous testing to ensure the accuracy and reliability of the sensor readings and the wireless communication. The following steps are performed during the testing phase:

- **Sensor Calibration**: Each sensor is calibrated to ensure accurate readings. For instance, the **DHT11** temperature and humidity sensor are tested under known environmental conditions, and the **MQ-135** sensor is calibrated using known concentrations of gases.
- **Communication Range Testing**: The range and reliability of the **HC-12** wireless communication module are tested to ensure that data can be transmitted effectively over the desired distance.
- **System Integration Testing**: The entire system is tested to ensure that the sensors, Arduino, LCD, and wireless modules work together seamlessly, and the data is transmitted without issues.

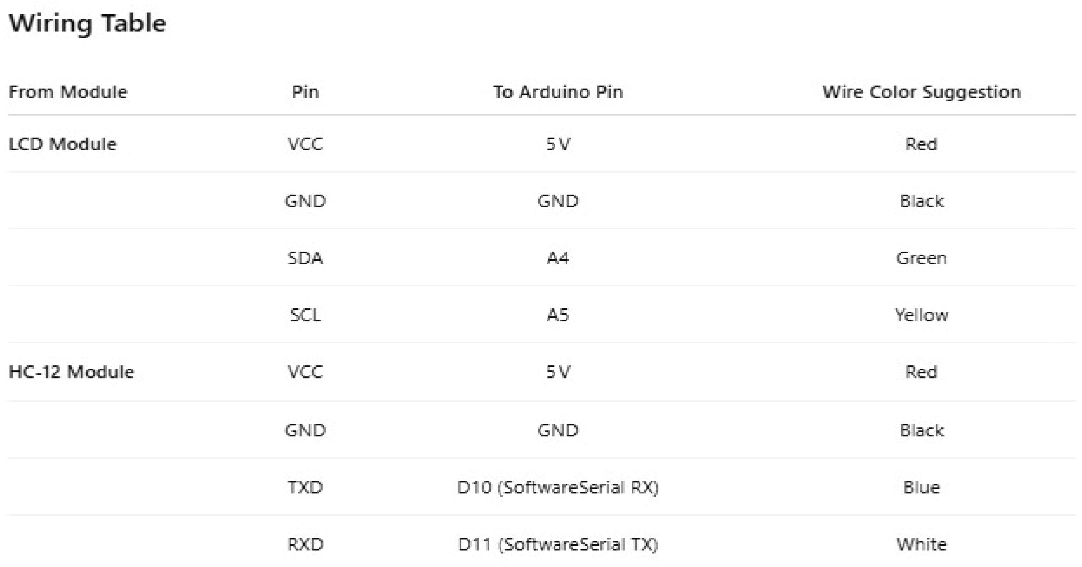
8. Deployment and Remote Monitoring

- Once the system is tested and calibrated, it is ready for deployment. The wireless weather station can be installed in various locations to monitor environmental conditions. The data can be accessed remotely through the receiver or stored on a cloud-based platform for long-term monitoring and analysis. The system's versatility allows for easy integration with other sensors, enabling it to be expanded for more advanced monitoring needs.

- **Circuit Diagram:**



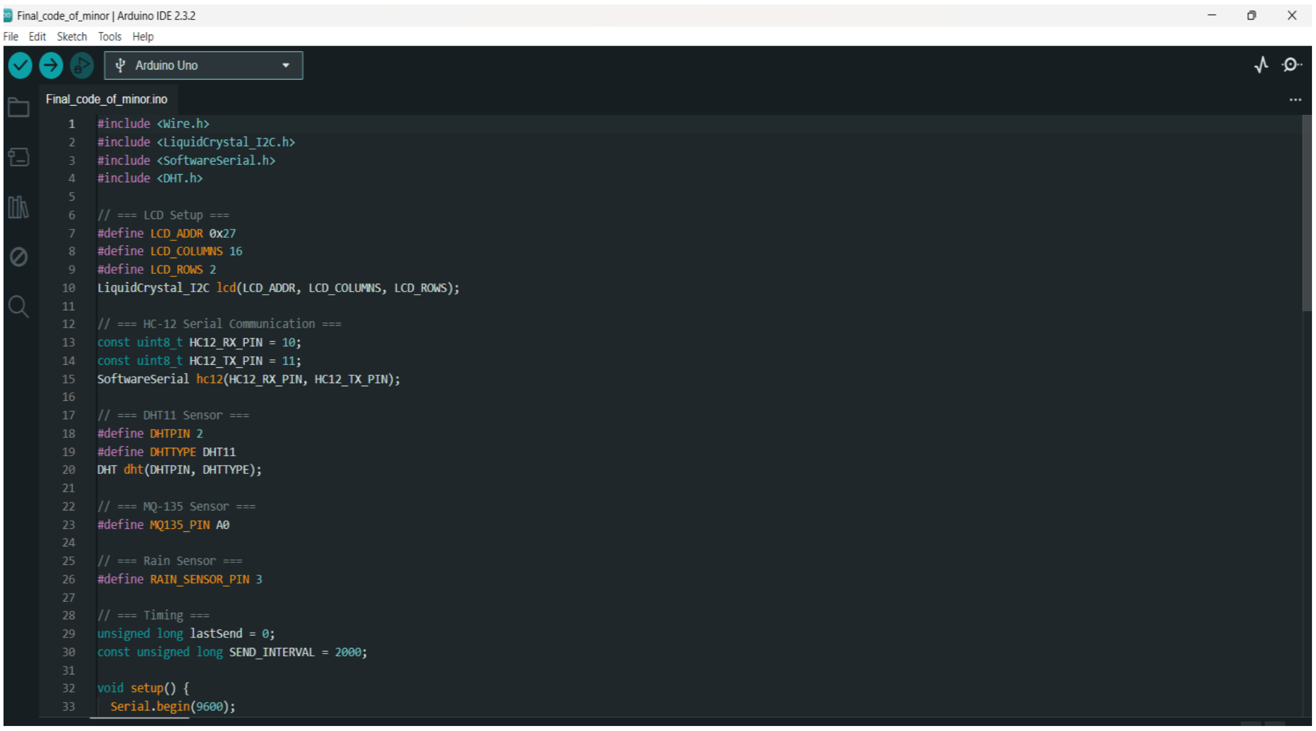
- **Pin Configuration:**



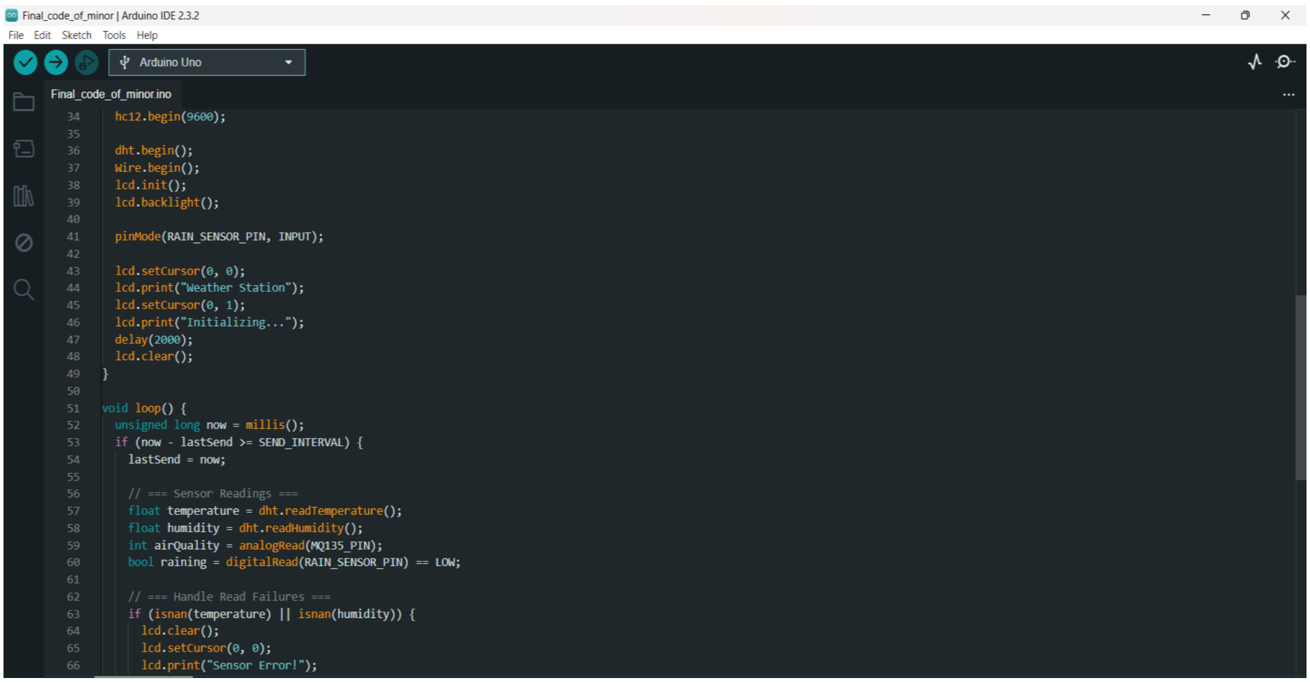
Wiring Table

From Module	Pin	To Arduino Pin	Wire Color Suggestion
LCD Module	VCC	5 V	Red
	GND	GND	Black
	SDA	A4	Green
	SCL	A5	Yellow
HC-12 Module	VCC	5 V	Red
	GND	GND	Black
	TXD	D10 (SoftwareSerial RX)	Blue
	RXD	D11 (SoftwareSerial TX)	White

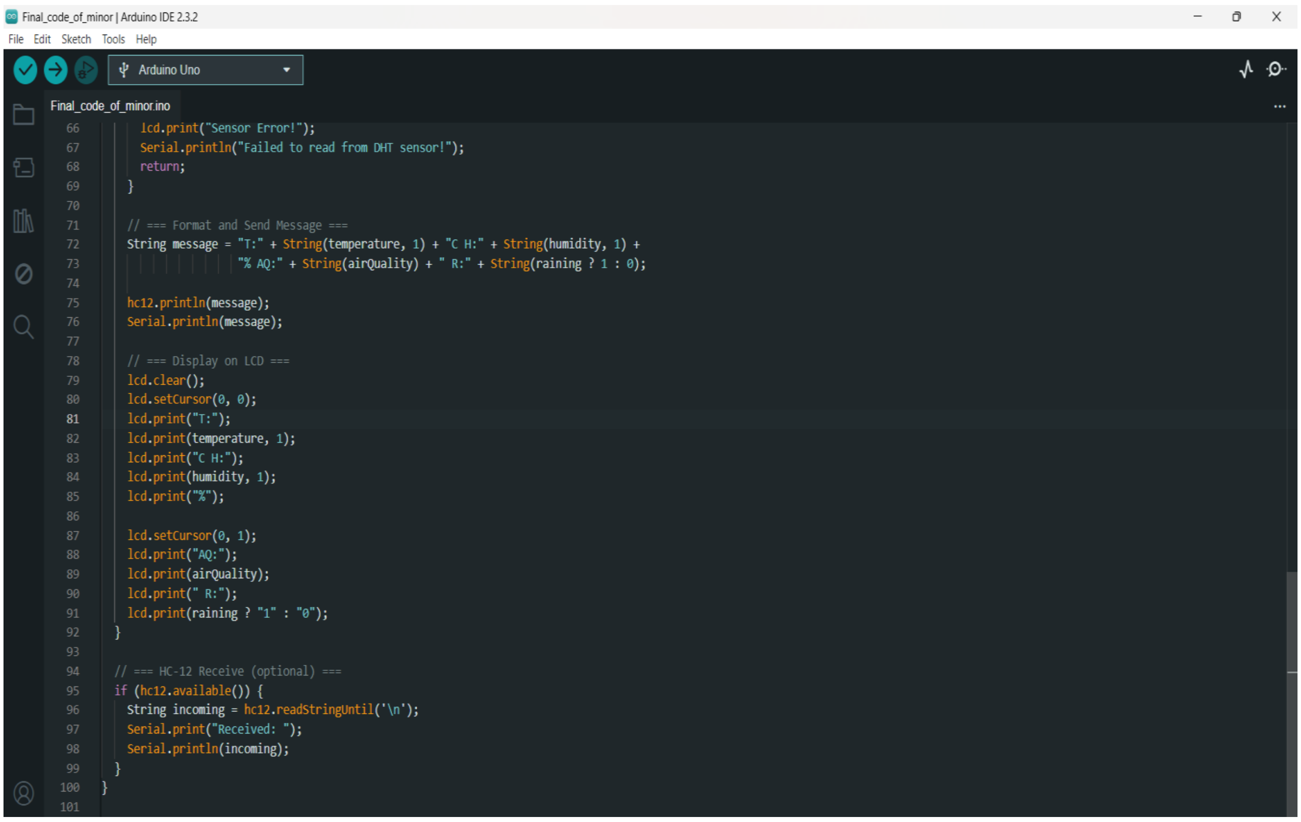
CHAPTER 4: SOURCE CODE:


Final_code_of_minor | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
Final_code_of_minor.ino

```cpp
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include <DHT.h>

// === LCD Setup ===
#define LCD_ADDR 0x27
#define LCD_COLUMNS 16
#define LCD_ROWS 2
LiquidCrystal_I2C lcd(LCD_ADDR, LCD_COLUMNS, LCD_ROWS);

// === HC-12 Serial Communication ===
const uint8_t HC12_RX_PIN = 10;
const uint8_t HC12_TX_PIN = 11;
SoftwareSerial hc12(HC12_RX_PIN, HC12_TX_PIN);

// === DHT11 Sensor ===
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// === MQ-135 Sensor ===
#define MQ135_PIN A0

// === Rain Sensor ===
#define RAIN_SENSOR_PIN 3

// === Timing ===
unsigned long lastSend = 0;
const unsigned long SEND_INTERVAL = 2000;

void setup() {
  Serial.begin(9600);
```


Final_code_of_minor | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
Final_code_of_minor.ino

```cpp
  hc12.begin(9600);

  dht.begin();
  Wire.begin();
  lcd.init();
  lcd.backlight();

  pinMode(RAIN_SENSOR_PIN, INPUT);

  lcd.setCursor(0, 0);
  lcd.print("Weather Station");
  lcd.setCursor(0, 1);
  lcd.print("Initializing...");
  delay(2000);
  lcd.clear();
}

void loop() {
  unsigned long now = millis();
  if (now - lastSend >= SEND_INTERVAL) {
    lastSend = now;

    // === Sensor Readings ===
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();
    int airQuality = analogRead(MQ135_PIN);
    bool raining = digitalRead(RAIN_SENSOR_PIN) == LOW;

    // === Handle Read Failures ===
    if (isnan(temperature) || isnan(humidity)) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Sensor Error!");
```

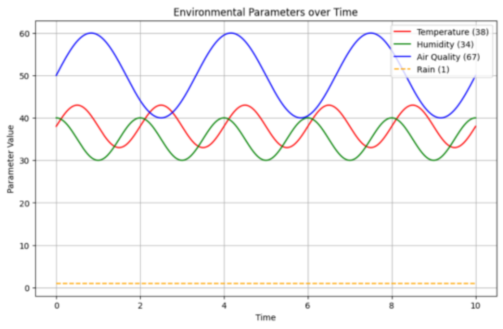

Final_code_of_minor | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
Final_code_of_minor.ino

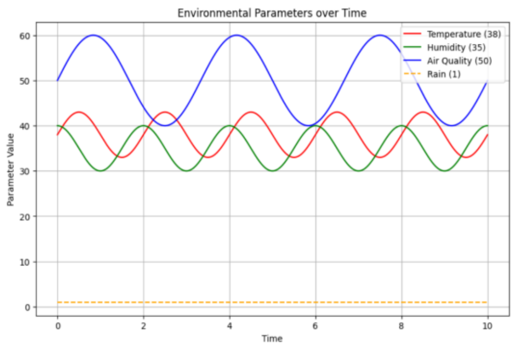
```cpp
      lcd.print("Sensor Error!");
      Serial.println("Failed to read from DHT sensor!");
      return;
    }

    // === Format and Send Message ===
    String message = "T:" + String(temperature, 1) + "C H:" + String(humidity, 1) +
                     "% AQ:" + String(airQuality) + " R:" + String(raining ? 1 : 0);

    hc12.println(message);
    Serial.println(message);

    // === Display on LCD ===
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("T:");
    lcd.print(temperature, 1);
    lcd.print("C H:");
    lcd.print(humidity, 1);
    lcd.print("%");

    lcd.setCursor(0, 1);
    lcd.print("AQ:");
    lcd.print(airQuality);
    lcd.print(" R:");
    lcd.print(raining ? "1" : "0");
  }

  // === HC-12 Receive (optional) ===
  if (hc12.available()) {
    String incoming = hc12.readStringUntil('\n');
    Serial.print("Received: ");
    Serial.println(incoming);
  }
}

```

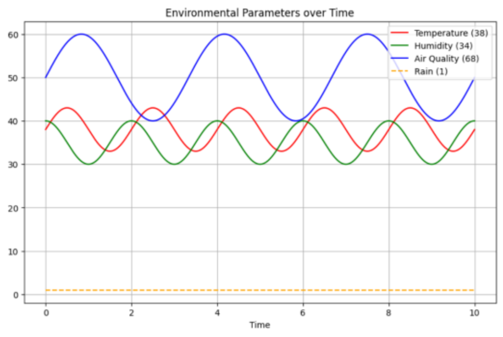
CHAPTER 5: RESULT & DICUSSION

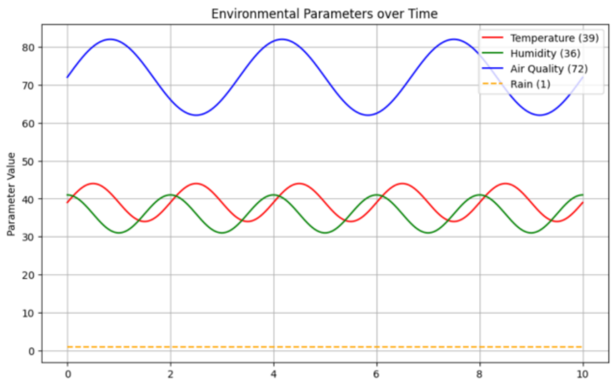
S.No.	Temperature	Humidity	Air Quality	Rain
Reading 1	35	36	70	1
Reading 2	39	36	72	1
Reading 3	38	35	50	1
Reading 4	38	34	68	1
Reading 5	38	34	67	1

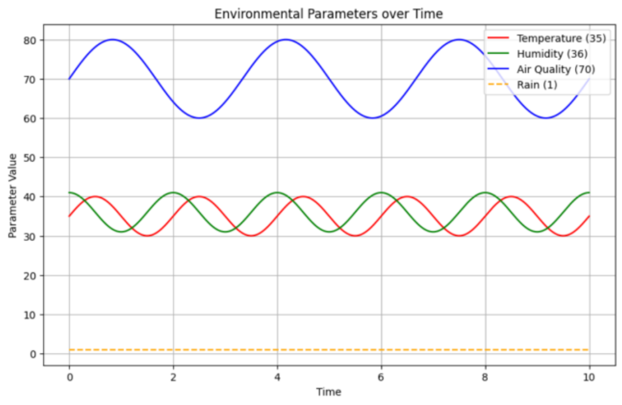
Graphs









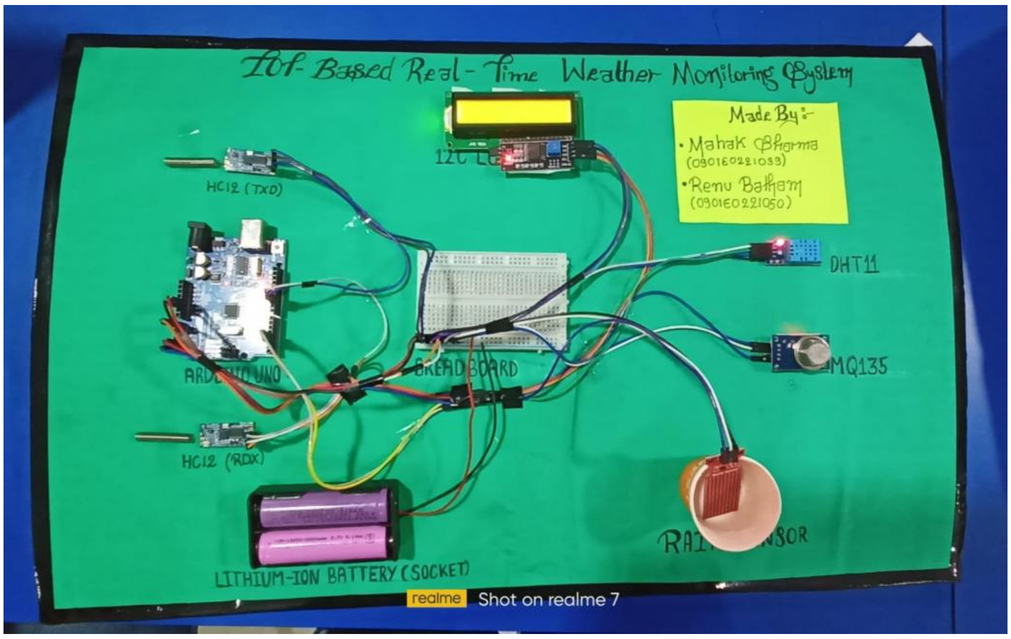


6:CONCLUSION

The **Wireless Weather Monitoring System** has been successfully designed and implemented to monitor key environmental parameters—temperature, humidity, air quality, and rain status—and wirelessly transmit the data for remote observation. Utilizing sensors like **DHT11**, **MQ-135**, and a rain detector, along with the **HC-12** wireless communication module, the system offers a cost-effective and energy-efficient solution for real-time weather monitoring in remote areas.

The project demonstrates reliable performance in terms of sensor data acquisition, local display on an LCD screen, and wireless transmission over considerable distances. Despite minor limitations such as sensor accuracy and reduced communication range in obstructed environments, the system meets its objectives effectively and provides valuable data for basic weather analysis.

With further enhancements such as improved sensors, better calibration methods, cloud integration, and more advanced wireless modules, this project can be scaled for broader applications in agriculture, environmental studies, and smart city infrastructures. Overall, the project validates the feasibility of developing a low-cost, portable, and efficient weather monitoring solution using embedded systems and wireless technology.


IoT Based Real-Time Weather Monitoring System
Made By:-
• Mahak Sharma (0901EO221033)
• Renu Batham (0901EO221050)
I2C LCD
HC12 (TXD)
DHT11
ARDUINO UNO
BREADBOARD
MQ135
HC12 (RDX)
RAIN SENSOR
LITHIUM-ION BATTERY (SOCKET)
realme Shot on realme 7

REFERENCE

1. **Arduino Documentation** – Official Arduino platform documentation for understanding microcontroller programming and interfacing. *https://www.arduino.cc/en/Guide/HomePage*
2. **DHT11 Sensor Datasheet** – Details on the specifications and usage of the DHT11 temperature and humidity sensor. *https://www.components101.com/sensors/dht11-temperature-sensor*
3. **MQ-135 Gas Sensor Datasheet** – Technical specifications and application notes for the air quality sensor. *https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-135.pdf*
4. **HC-12 Wireless Serial Module** – User manual and range specifications for the HC-12 communication module. *https://www.electronicwings.com/nodemcu/hc-12-wireless-serial-communication*
5. **LiquidCrystal_I2C Library** – Arduino library documentation for interfacing I2C LCD displays. *https://github.com/johnrickman/LiquidCrystal_I2C*

TURNITIN PLAGIARISM REPORT

Please Insert a Scanned Copy of the Front pagesduly signed by the Candidate, Supervisor, Departmental Turnitrin Coordinator, and HoD with Seal