Project Report

On

"SMART ENVIRONMENTAL MONITORING SYSTEM"

By

Piyush Singh

In partial fulfillment of requirements for the award of degree in Embedded System (IOT) (2023)

Under the Project Guidance of

Mrs. Riti Chatterjee (Guide, Webel Fujisoft Vara)

Mr. Sovik Chakraborty (Reviewer,Webel Fujisoft Vara)
&
Mr. Souvik Bhattacharji (Reviewer,Webel Fujisoft Vara)

DEPARTMENT OF INTERNET OF THINGS (IOT)
WEBEL FUJISOFT VARA COE 4.0
(Education Institution in West Bengal)
Jodthbhum, Newtown West Bengal – 743502

Project Review Certificate

This is to certify that the work recorded in this project report entitled **“ Real-time Air Quality, Temperature, and Humidity Tracking via LoRa Sx1278 and Cloud Integration.”** has been carried out by **Piyush Singh** of Internet Of Things Department of Webel Fujisoft vara in partial fulfilment of the requirements for the award of Post-Graduation in Embedded System. This report has been duly reviewed by the undersigned and recommended for final submission for Major Project Viva Examination.

Mrs. Riti Chatterjee (Senior Faculty)
Department of IOT
Webel Fujisoft Vara COE 4.0
Jodthbhum, Newtown West Bengal – 743502

Certificate of Acceptance

This is to certify that the below mentioned student(s) of Internet of Things Department of Webel FujiSoft Vara COE 4.0 has/have worked under the supervision of **Mr. Sovik Chakraborty (Reviewer,Webel Fuji Soft Vara)** of Department of Internet Of Things from **5th June 2023 to 26th July 2023** on the project entitled **“Real-time Air Quality, Temperature, and Humidity Tracking via LoRa Sx1278 and Cloud Integration.”** The project is hereby accepted by the Department of Internet Of Things, Webel FujiSoft Vara in partial fulfilment of the requirements for the award of Post-Graduation in Embedded System.

S.No.	Name of Student(s)	Project Venue
1	Piyush Singh	Webel FujiSoft Vara

Mr. Debashis Mazumdar
Director
Webel Fujisoft Vara CoE 4.0
Jodthbhum, Newtown West Bengal – 743502

Declaration

We the undersigned, hereby declare that the work recorded in this project report entitled **"Real-time Air Quality, Temperature, and Humidity Tracking via LoRa Sx1278 and Cloud Integration."** in partial fulfilment for the requirements of award of PG (Embedded System) from "**Webel Fujisoft Vara CoE 4.0."** is a faithful and bonafide project work carried out at **"Webel Fujisoft Vara, Kolkata"** under the supervision and guidance of **Mrs. Riti Chatterjee (Guide, Webel Fujisoft Vara), Mr. Sovik Chakraborty (Reviewer , Webel Fujisoft Vara) & Mr. Souvik Bhattacharji (Reviewer, Webel Fujisoft Vara)** of Department of Internet Of Things..

The results of this investigation reported in this project have so far not been reported for any other Degree or any other technical forum.

The assistance and help received during the investigation have been duly acknowledged

Piyush Singh

Student (IOT)

Webel FujiSoft Vara

Newtown Kolkata West Bengal – 743502

Acknowledgement

We wish to express our sincere thanks to Guide **Mrs. Riti Chatterjee, Senior Faculty of Webel Fujisoft Vara** for providing us an opportunity to carry out project work on **"Real-time Air Quality, Temperature, and Humidity Tracking via LoRa Sx1278 and Cloud Integration."** and their unlisted encouragement and guidance carrying out this project work.

We sincerely thank **Mr. Sovik Chakraborty (Assistant Professor I) & Mr. Souvik Bhattacharji (Assistant Professor II),** of Department of Internet Of Things., Webel Fujisoft Vara CoE for her guidance and encouragement in carrying out this project till the completion.

We would like to express our sincere thanks to **Mr. Debashis Mazumdar, Director, Webel Fujisoft Vara CoE 4.0** for allowing us to carry out our project from and valuable support and guidance during the project period.
We would like to express our humble gratitude to **Mr. Anirban Choudhuri (Professor), Mr. Sanjay Lal (AVP-Academics), Mr. Abir Banerjee (Assistant Professor), and Mrs. Debjani Dey(Associate Manager)** for their unlisted encouragement and their timely support and guidance till the completion of the project work.

Lastly, we wish to avail this opportunity, express a sense of gratitude and love to all our teachers, staff of the department of Webel Fujisoft Vara CoE 4.0, friends and fellow for their support and help.

Piyush Singh
Student (IOT)
Webel FujiSoft Vara
Newtown Kolkata West Bengal – 743502

Document Control Sheet

1	Report No	IOT/Project/Internal /PG/2023
2	Title of the Report	Smart Environmental Monitoring System
3	Type of Report	Technical
4	Author	Piyush Singh
5	Organizing Unit	Webel Fujisoft Vara CoE 4.0
6	Language of the Document	English
7	Abstract	In this project titled, Smart Environmental Monitoring System: Real-time Air Quality, Temperature, and Humidity Tracking via LoRa Sx1278 and Cloud Integration.
8	Security Classification	General
9	Distribution Statement	General

List of Contents

List of Figures

List of Tables

ABSTRACT

The "Air Quality and Temperature Humidity Monitoring" project utilizes SX1278 (LoRa) communication, Arduino microcontroller, DHT11 sensor for temperature and humidity measurement, and MQ7 sensor for monitoring air quality. The objective of this project is to create a wireless system capable of collecting environmental data and transmitting it to the cloud for remote monitoring and analysis.

The system consists of two main components: the sensor node and the cloud server. The sensor node is equipped with an Arduino board, DHT11 sensor, MQ7 gas sensor, and an SX1278 LoRa module for long-range wireless communication. The DHT11 sensor is used to measure ambient temperature and humidity, while the MQ7 gas sensor is employed to detect carbon monoxide (CO) levels in the air. The Arduino board processes the sensor data, converts it into a suitable format, and transmits it using the SX1278 LoRa module. LoRa (Long Range) technology enables communication over long distances with low power consumption, making it ideal for this application. On the cloud side server receives the data sent by the sensor nodes. This cloud platform stores the received data, performs data analysis, and provides visualizations for real-time monitoring and historical analysis of air quality and environmental conditions. On receiver side data received from cloud and display on the screen.

1.INTRODUCTION

1.1 General overview

The Air Quality and Temperature Humidity Monitoring System is a comprehensive project aimed at designing and implementing a robust solution for real-time monitoring of air quality, temperature, and humidity. The system utilizes cutting-edge technologies, including the SX1278 LoRa module, Arduino microcontroller, DHT11 sensor, and MQ-7 gas sensor, to collect, process, and transmit environmental data to a cloud-based platform. The SX1278 LoRa module plays a pivotal role in the system, providing a reliable and long-range communication channel between the sensor nodes and the cloud platform. LoRa technology is well-suited for environmental monitoring projects due to its low power consumption, extended range, and ability to penetrate obstacles, making it ideal for transmitting data from remote locations. The heart of the system lies in the Arduino microcontroller, which serves as the central processing unit. The Arduino board interfaces with the DHT11 and MQ-7 sensors, collecting real-time data on temperature, humidity, and CO levels. The DHT11 sensor measures ambient temperature and humidity with a high degree of accuracy, while the MQ-7 gas sensor detects carbon monoxide levels, a hazardous gas found in combustion processes.

Air pollution, caused by various human activities and industrial processes, has far-reaching consequences on human health, climate, and ecosystems. The adverse effects of air pollution on respiratory and cardiovascular health, as well as its contribution to global warming and climate change, have led to a growing concern among scientists, policymakers, and the public. Monitoring air quality, temperature, and humidity is crucial in assessing the impact of pollution and taking timely actions to mitigate its effects.

1.2 Literature survey

SL. No	Author(s)	Title	Findings
1	Mia Rosmiati, Fitri Susanti, Moch. Fahru Rizal .	"IoT-Based Environmental Monitoring and Tracking System for Smart Cities Using LoRaWAN and Cloud Computing" .	The research highlights the importance of LoRaWAN technology for long-range communication and cloud integration for real-time data processing and analysis.
2	Waheb A. Jabbar	“LoRaWAN-Based IoT System Implementation for Long-Range Outdoor Air Quality Monitoring”	This research explores the integration of IoT and LoRa technology for real-time environmental monitoring.
3	Mohammad Fadhli, Asriyadi Asriyadi, Andi Ramadhan,Gita Affrylia	"Environmental Data Acquisition and Processing in Smart Cities Using LoRaWAN“.	While the research covers multiple environmental parameters, it emphasizes the significance of air quality, temperature, and humidity monitoring in shaping smart city initiatives. .
4	D.Monica Satyavathi, B.Vandana Mala, Ch.Veera Vamsi,Ch.Neeraj	"An Air Pollution Monitoring System Based on LoRa and MQTT.”	The research presents a case study of an air quality monitoring system deployed in an urban area. The study highlights the advantages of LoRaWAN technology in providing real-time environmental data for urban planning and pollution control.

Table 1.1 : Literature Survey

1.3.Problem definition

- Real-Time Monitoring: Creating a solution capable of collecting and processing data from multiple environmental sensors in real-time is essential. The system should provide accurate and up-to-date information on air quality and weather conditions.
- Long-Range Communication: In remote or large-scale monitoring scenarios, the data transmission infrastructure must overcome communication challenges. Implementing long-range communication technology is critical to ensure seamless data transmission to a centralized platform.
- Data Management and Analysis: The collected environmental data needs to be efficiently stored, organized, and analyzed. Users should have easy access to historical data, and the system must provide data visualization tools for meaningful interpretation.
- User-Friendly Interface: To maximize the system's impact, a user-friendly interface is essential. The monitoring system should be accessible to individuals with varying levels of technical expertise, enabling easy navigation and interpretation of data.
- Scalability and Reliability: As the monitoring system may expand to monitor multiple locations, it must be designed for scalability and reliability. The system should handle a growing number of sensors and users without compromising performance.

1.4. Software Requirements Specifications

1.4.1 Software Requirements

- Arduino IDE
- Install the Arduino LoRa library for SX1278 communication.
- Install the DHT sensor library for the DHT11 sensor.
- TagoIO Cloud platform.

1.4.2 Hardware Requirements

- 2xArduino Board: Arduino Uno
- 2xSX1278 LoRa Module
- DHT11
- MQ7

Functional Requirements:

Functional Requirements for Air Quality and Temperature Humidity Monitoring System using Arduino and SX1278:

- Initialize Sensors and LoRa Module: Initialize the DHT11 and MQ7 sensors, as well as the SX1278 LoRa module, to prepare them for data collection and communication.
- Read Sensor Data: Continuously read data from the DHT11 sensor to measure ambient temperature and humidity. Periodically read data from the MQ7 sensor to detect carbon monoxide (CO) levels in the air.
- Data Processing and Calibration: Process the raw analog sensor readings and convert them into meaningful digital values for temperature, humidity, and CO levels. Perform calibration and error-checking to ensure accurate and reliable environmental data.
- Wireless Data Transmission: Use the SX1278 LoRa module to transmit the processed environmental data wirelessly to the cloud platform. Implement LoRa communication protocols to ensure reliable and long-range data transmission.
- Cloud Connectivity: Establish a connection with the cloud-based platform to send and receive data. Implement HTTP communication protocols for data transmission.
- Real-Time Monitoring: Enable real-time monitoring of air quality, temperature, and humidity on the cloud platform. Display the current sensor readings in LCD.

Non - Functional Requirements

- Performance: The system should have low latency for data transmission and processing to provide real-time monitoring capabilities. It should be able to handle a significant number of sensor readings and user requests without compromising performance.
- Reliability: The system should be highly reliable, with minimal downtime and disruptions in data transmission and analysis. It should have mechanisms to recover from communication failures or sensor malfunctions.
- Scalability: The system should be designed to scale effortlessly as the number of sensors and users increases. It should be capable of handling a growing volume of data without sacrificing performance.
- Compatibility: The system should be compatible with various devices, such as smartphones, tablets, and desktops, for remote access.
- Usability: Users should be able to access and interpret environmental data without requiring technical expertise.
- Portability: Users should be able to set up the system in various situations because it should be portable and simple to deploy.
- Power Efficiency: Sensor readings and LoRa transmission should be optimized to reduce power consumption.
- Accessibility: The system should take accessibility standards into account to make sure people with impairments can use it.

1.5 Proposed Solution Strategy

- Sensor Integration: Integrate the DHT11 sensor for temperature and humidity measurement and the MQ7 gas sensor for carbon monoxide (CO) detection with the Arduino microcontroller. Ensure proper wiring and interfacing for accurate data acquisition.
- LoRa Communication Setup: Incorporate the SX1278 LoRa module for wireless data transmission. Configure the LoRa module to establish long-range communication with the cloud platform. Develop LoRa communication protocols for data packetization and transmission efficiency.
- Cloud Connectivity: Establish a connection between the LoRa-enabled Arduino and the cloud-based platform. Utilize suitable communication protocols such as MQTT or HTTP for secure data transmission.
- Testing and Optimization: Thoroughly test the system for reliability, accuracy, and performance in different environmental conditions. Continuously optimize the system for efficiency and effectiveness based on user feedback and performance metrics.

1.6 Preliminary User's Manual

- Connect the DHT11 sensor and MQ7 gas sensor to the Arduino board using jumper wires. Ensure the connections are secure and correct.
- Connect the SX1278 LoRa module to the Arduino board using jumper wires. Attach the antenna to the LoRa module for better communication range.
- Upload the sketch to the Arduino board using the Arduino IDE.
- Obtain the necessary credentials and access keys to connect the Arduino with the cloud platform securely.
- Once the hardware and software setups are complete, you can start monitoring air quality, temperature, and humidity:

  Access the cloud-based platform through a web. Navigate to the dashboard or monitoring section to view real-time environmental data. Monitor the system regularly to ensure it is running correctly and receive timely alerts if environmental conditions deviate from safe levels.

  Troubleshooting: In case of any issues or errors, refer to the troubleshooting section in the documentation. If the problem persists, contact user support for assistance.

Handle the sensors and components with care to prevent damage.

Place the system in a suitable location away from direct sunlight, water, or extreme temperatures. In case of any unusual readings or alerts, take appropriate precautions and consult relevant authorities if necessary.

To make sure the Arduino and attached components are operating properly, check the power supply.

1.7 Organization of the report

Chapter 1 : Introduction

- Chapter 1.1 : ‘General Overview of the problem’ gives Introduction about the basic idea about the approach on how we solve the problems and the basic problem that we faced.
- Chapter 1.2 : ‘Literature survey’ gives the research that we conducted to know how to solve and understand the problem and validate our approach of solving the same.
- Chapter 1.3 : ‘Problem statement’ talk about the problem that we are trying to solve and approaches that we are taking.
- o Chapter 1.4 : ‘Analysis of the problem’ gives the detailed description of the difficulties associated and encountered .
- Chapter 1.5 : ‘Software Requirement Specification’ elaborates the system , functional and non-functional requirements associated with the project.
- Chapter 1.6 : ‘Proposed Solution Strategy’ gives the explanation on solution for the problems associated with the project.
- Chapter 1.7 : ‘Preliminary user’s manual’ provides the glimpse of how the deployed software works and how it is to be used by the users.

Chapter 2 : ‘Design Strategy’ contains the block diagram, , proposed architecture diagram ,process flow diagram and flowchart.

Chapter 3 : ‘Testing’ contains the various test cases performed for different scenarios to check whether the deployed software matches the requirements of the users.

Chapter 4 : ‘Implementation Details’ explains the incorporated algorithms and various modules .

Chapter 5: ‘Results and Discussions’ gives a comparative view on the obtained results in the deployed application.

Chapter 6: ‘Summary and Conclusion’

- Chapter 6.1 : ‘Summary of achievements’ gives a description on the finally achieved goals within the project.

- Chapter 6.2 : ‘Difficulties encountered’ provides a description on the various difficulties that were encountered while accomplishing the project
- Chapter 6.3 : ‘Limitation of the project’ highlights the project’s functionality restrictions.
- Chapter 6.4: ‘Future Scope of Work’ shows the works associated with the proposed project which can be accomplished in future to improve the project’s functionality.
- Chapter 6.5: Application for Business purpose

Chapter 7: ‘Gantt Chart’ illustrate the timeline of tasks and activities.

Chapter 8: ‘References’ depicts the sources of information that were used in the project.

2.Design

2.1 Block Diagram

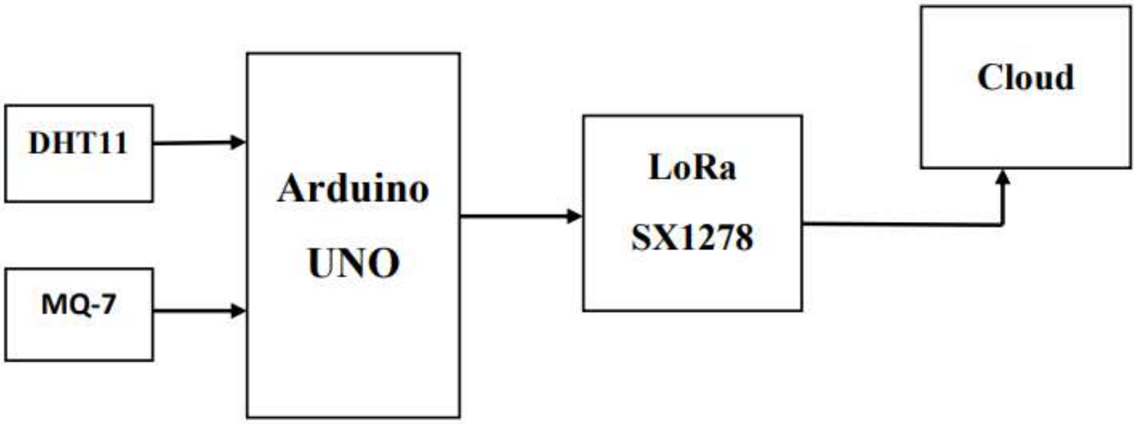


Fig 2.1: Block Diagram of Transmitter Side

The transmitter side, also known as the sensor node, is responsible for collecting environmental data such as air quality, temperature, and humidity using DHT11 and MQ-7 sensors. It processes and packages this data for wireless transmission to the receiver or cloud-based platform. The microcontroller, often an Arduino board, acts as the brain of the sensor node. It receives data from the DHT11 and MQ-7 sensors and performs data processing and calibration. The microcontroller processes the raw data obtained from the sensors to convert it into meaningful measurements for temperature, humidity, and CO gas concentration. Additionally, calibration algorithms may be implemented to ensure accurate and reliable data readings. The transmitter side uses the SX1278 LoRa module for wireless communication. The LoRa module is connected to the microcontroller and serves as the communication gateway.

2.1 Block Diagram

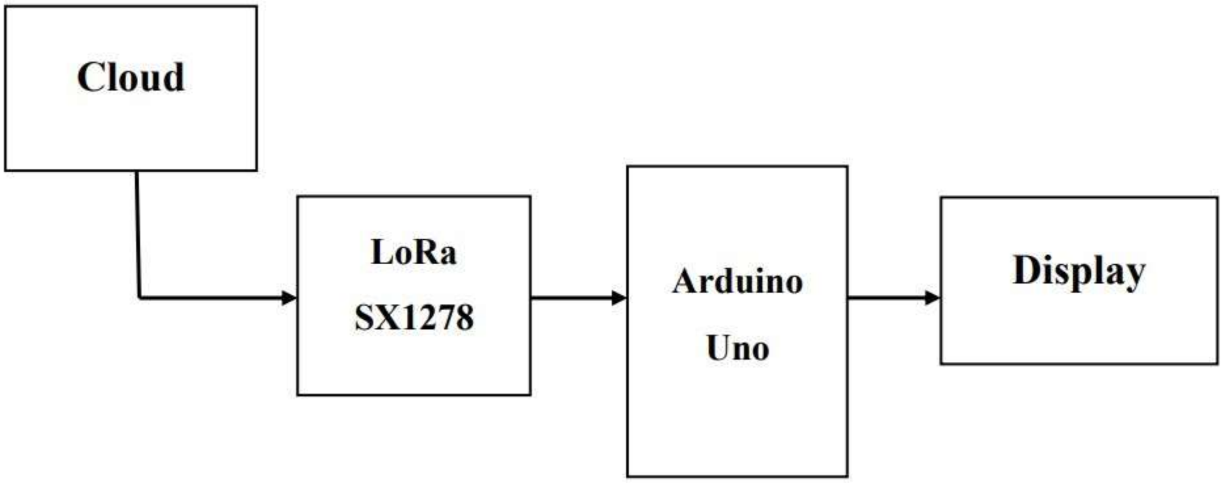


Fig 2.2: Block Diagram od Receiver Side

The receiver side, also known as the cloud-based platform, is responsible for receiving, processing, storing, and analyzing the data transmitted from the sensor node (transmitter side). It serves as the central hub for data management, visualization, and user interaction. The receiver side operates on a cloud server or cloud-based infrastructure. The cloud server hosts the software applications and databases needed to manage and process the incoming data from the Transmitter side. The receiver side also includes an SX1278 LoRa module, similar to the transmitter side. This LoRa module is connected to an Arduino within the cloud server, acting as the receiver for the transmitted data. The LoRa module on the receiver side receives the data transmitted by the sensor node and decodes it into the original data format (temperature, humidity, and CO gas concentration).

The receiver side of the Air Quality and Temperature Humidity Monitoring System is responsible for receiving, processing, and managing the environmental data transmitted from the sensor node. It provides users with a comprehensive platform to monitor, analyze, and respond to environmental conditions in real-time, ensuring safety, and contributing to environmental conservation efforts.

2.2 Flowchart

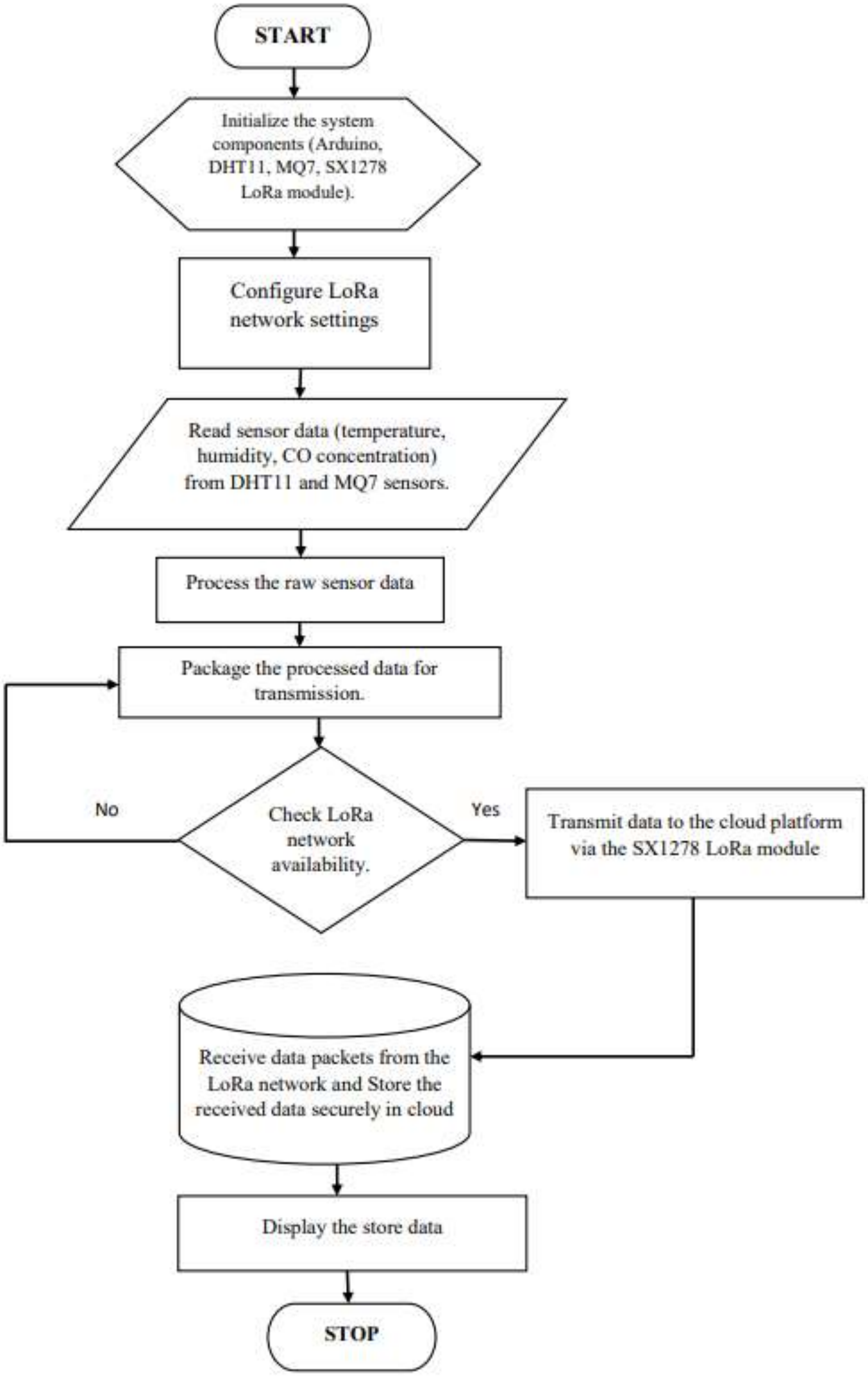

START
Initialize the system components (Arduino, DHT11, MQ7, SX1278 LoRa module).
Configure LoRa network settings
Read sensor data (temperature, humidity, CO concentration) from DHT11 and MQ7 sensors.
Process the raw sensor data
Package the processed data for transmission.
No
Check LoRa network availability.
Yes
Transmit data to the cloud platform via the SX1278 LoRa module
Receive data packets from the LoRa network and Store the received data securely in cloud
Display the store data
STOP

Fig 2.3: Flowchart

2.3 Circuit Diagram

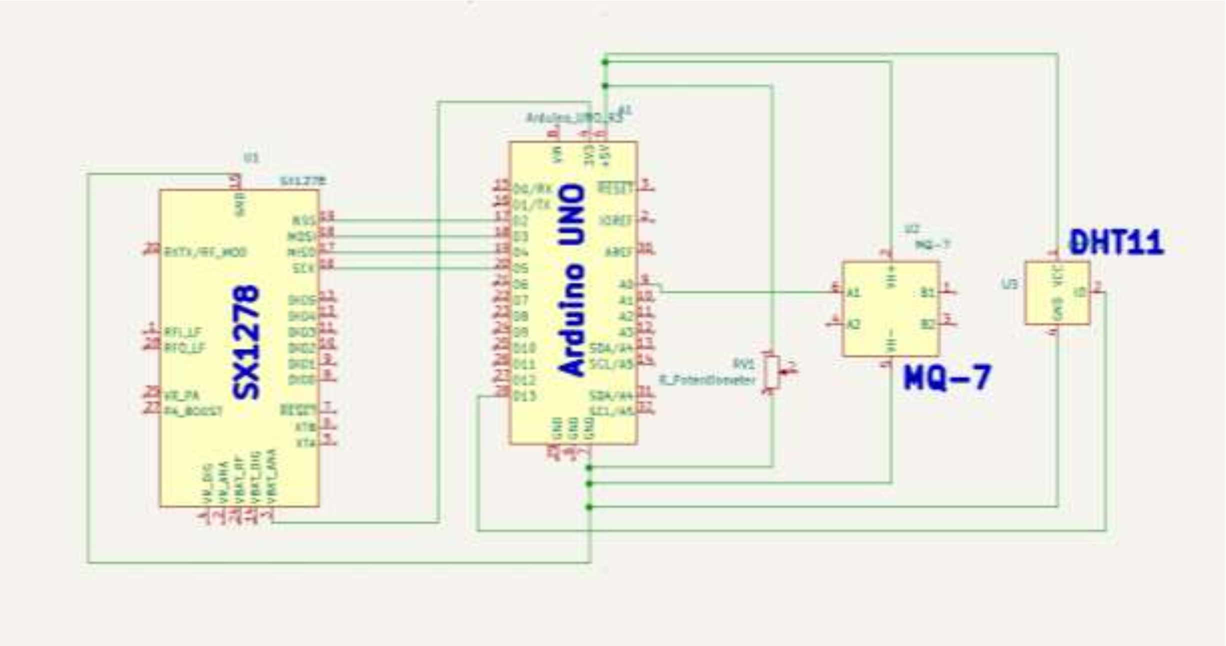


Fig 2.4: Circuit Diagram of Transmitter End

Connect the VCC (+5V) pin of Arduino Uno to the VCC pins of the DHT11, MQ-7 and SX1278 module. This connection provides power to these components. Connect the GND pin of Arduino Uno to the GND pins of the DHT11, MQ-7 and SX1278 module. This connection provides the common ground reference for all the components. Connect a digital pin of the Arduino Uno (e.g., pin D13) to the data pin of the DHT11 sensor. This connection allows the Arduino to read the temperature and humidity data from the DHT11 sensor. Connect an analog pin of the Arduino Uno (e.g., A0) to the analog output pin of the MQ-7 sensor. This connection enables the Arduino to read the analog gas sensor values from the MQ-7 sensor.

2.3 Circuit Diagram

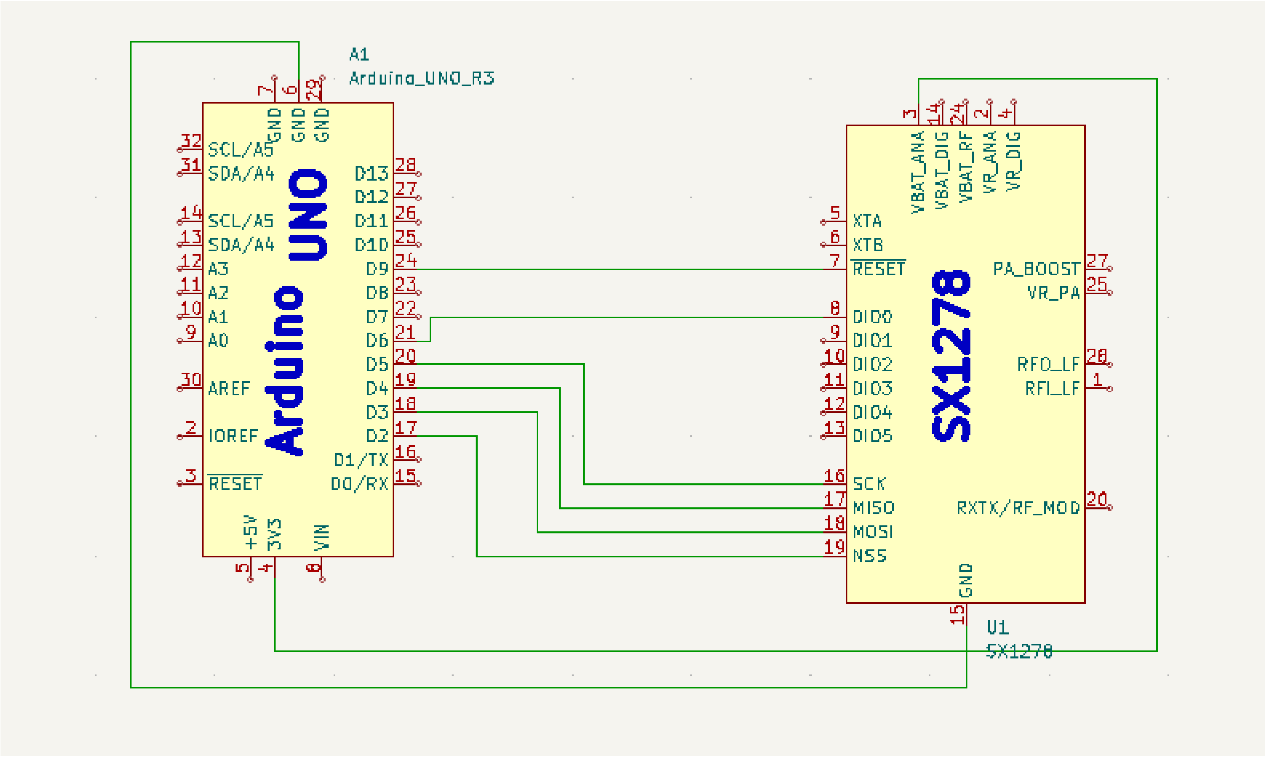


Fig 2.5 Circuit Diagram of Receiver Side

Connect the MOSI pin of the LoRa SX1278 module to digital pin 3 of the Arduino: MOSI (Master Out Slave In) is the output data line from the master device (Arduino) to the slave device (LoRa module). The Arduino uses this line to send data to the LoRa module. By connecting the MOSI pin of the LoRa SX1278 module to digital pin 3 of the Arduino, you are establishing a communication path from the Arduino to the LoRa module for transmitting data. Connect the MISO pin of the LoRa SX1278 module to digital pin 4 of the Arduino: MISO (Master In Slave Out) is the input data line from the slave device (LoRa module) to the master device (Arduino). The LoRa module uses this line to send data back to the Arduino. By connecting the MISO pin of the LoRa SX1278 module to digital pin 4 of the Arduino, you are establishing a communication path from the LoRa module to the Arduino for receiving data. Connect the NSS pin of the LoRa SX1278 module to digital pin 2 of the Arduino: The NSS (Slave Select) pin is used for selecting the LoRa module as the target device for communication. It allows the Arduino to communicate exclusively with the LoRa module. By connecting the NSS pin of the LoRa SX1278 module to digital pin 10 of the Arduino, you are designating pin 10 as the slave select pin for the LoRa module. Connect the SCK pin of the LoRa SX1278 module to digital pin 5 of the Arduino: SCK (Serial Clock) is the clock signal line used for synchronizing data transmission between the master device (Arduino) and the slave device (LoRa module). It ensures that data is transferred reliably and at the correct timing. By connecting the SCK pin of the LoRa SX1278 module to digital pin 5 of the Arduino, you are establishing a clock signal connection between the Arduino and the LoRa module.

Connect the VCC (+5V) pin of Arduino Uno to the VCC pins of the SX1278 module. This connection provides power to these components. Connect the GND pin of Arduino Uno to the GND pins of SX1278 module. This connection provides the common ground reference for all the components

2.4 Process Flow Diagram

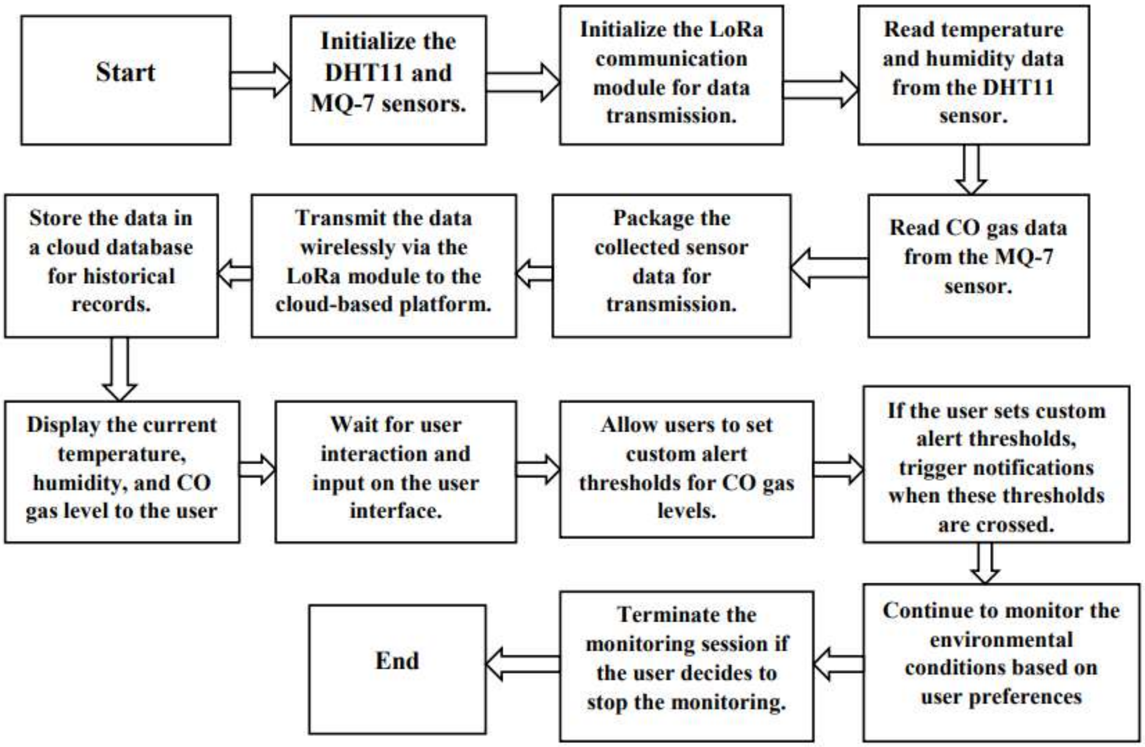

Start
Initialize the DHT11 and MQ-7 sensors.
Initialize the LoRa communication module for data transmission.
Read temperature and humidity data from the DHT11 sensor.
Read CO gas data from the MQ-7 sensor.
Package the collected sensor data for transmission.
Transmit the data wirelessly via the LoRa module to the cloud-based platform.
Store the data in a cloud database for historical records.
Display the current temperature, humidity, and CO gas level to the user
Wait for user interaction and input on the user interface.
Allow users to set custom alert thresholds for CO gas levels.
If the user sets custom alert thresholds, trigger notifications when these thresholds are crossed.
Continue to monitor the environmental conditions based on user preferences
Terminate the monitoring session if the user decides to stop the monitoring.
End

Fig 2.6 Process Flow Diagram

The process flow of the Air Quality and Temperature Humidity Monitoring System begins with the initialization phase, where the DHT11 and MQ-7 sensors, along with the LoRa communication module, are set up for data collection and transmission. During the data collection phase, temperature, humidity, and CO gas data are read from the respective sensors. The collected data is then packaged and transmitted wirelessly to the cloud-based platform. On the cloud platform, the received data is stored in a cloud database for historical records and analyzed to identify trends and patterns. The real-time data is updated on the user interface, allowing users to view the current environmental conditions, set custom alert thresholds, and request historical data. If users set alert thresholds, they are notified through the user interface or email when those thresholds are crossed. The monitoring session continues until the user decides to stop, ensuring continuous monitoring and timely alerts for environmental safety.

3. Testing

One test case is a document that details the input action or event and an anticipated response in order to assess whether the application feature is appropriately functioning. The expected output, test case identifier, test case name, and objective name should all be included in a test case. Because creating test cases entails thoroughly considering how the application will operate, it is crucial to do so as early in the development life cycle as is practical. Test cases can be used to identify issues with requirements or application design.

Test case specifications

The following are the test case specifications included:

Test Case	Description	Test Steps	Expected Result
Sensor Data Accuracy	Ensure sensor data readings are accurate and within acceptable tolerances.	Place the system in a controlled environment with known temperature, humidity, and CO levels. Record the sensor readings from the DHT11 and MQ7 sensors. Compare the recorded values with the known values to check for accuracy.	The sensor readings closely match the known values within an acceptable range.
LoRa Data Transmission	Verify successful data transmission over the LoRa network.	Configure the system with valid LoRa network settings. Initiate data transmission from the Arduino to the cloud platform via the SX1278 LoRa module. Verify that the cloud platform receives the transmitted data.	The cloud platform receives the transmitted data without data loss or corruption.

Test Case	**Description**	**Test Steps**	**Expected Result**
Cloud Data Storage	Confirm accurate and secure data storage on the cloud platform.	Send multiple sets of sensor data to the cloud platform. Access the cloud platform and verify that the data is stored correctly. Attempt to access the data using unauthorized means to test data security.	The cloud platform stores all received data accurately and securely. Unauthorized access attempts are blocked.
Real-Time Data Visualization	Ensure real-time display of air quality data on the cloud platform	Transmit live sensor data to the cloud platform. Observe the cloud platform's user interface to check if the data is updated in real-time.	The cloud platform's user interface displays real-time air quality data without significant delay.
Scalability	Determine system's scalability to handle multiple monitoring stations.	Deploy multiple instances of the monitoring system in different locations. Ensure each system can independently collect and transmit data without interference. Verify the cloud platform can handle and present data from all monitoring stations effectively.	The system can handle multiple monitoring stations independently and the cloud platform manages data effectively.
System Reliability	Assess system reliability and stability during prolonged operation.	Run the system continuously for an extended period (e.g., 24 hours or more). Monitor the system for any crashes, data transmission failures, or abnormal behavior.	The system remains stable and reliable without unexpected crashes or malfunctions.

Table 3.1:Testing 1

4. Implementation Details

4.1 Technology Background

- **Arduino board**

Popular microcontroller board called the Arduino Uno is based on the ATmega328P microcontroller. One of the Arduino family's most popular boards, it is made for novices and amateurs to quickly prototype and develop their electronic creations.

Here are some key features and specifications of the Arduino Uno:

1.Microprocessor: The ATmega328P microprocessor, which powers the Arduino Uno, operates at 16 MHz, and includes 32 KB of flash memory for storing your programme code. For data storage, it additionally features 1KB of EEPROM and 2KB of SRAM.

2.Digital and Analog I/O: The board has 14 digital input/output pins, 6 of which can be used as PWM (Pulse Width Modulation) outputs, and 8 analogue input/output pins. Sensors, LEDs, motors, and other electronic components can all be connected to and controlled by these pins. For reading analogue signals, it also includes 6 analogue input pins.

3.Interfaces: The Uno has a USB connection that makes it simple to connect it to a computer for serial communication and programming. Additionally, it contains a DC power jack, an ICSP (In-Circuit Serial Programming) header, and a reset button for resuming programming. 4. Power supply: The board can be powered either by an external power source or by a USB connection. A built-in voltage regulator produces a steady 5V output to power the microcontroller and other components, and the recommended input voltage range is 7–12V. 5.Programming: The C/C++-based Arduino programming language was used to create programmes for the Arduino Uno. A user-friendly interface is offered by the Arduino IDE (Integrated Development Environment) for authoring, compiling, and uploading code to the board.

The Arduino Uno is renowned for its ease of use, adaptability, and broad support from the community. It is a great option for both new and seasoned builders due to its big user base and extensive online library of open-source code and training.

Features of the Arduino UNO:

Microcontroller: ATmega328

Operating Voltage: 5V

Input Voltage (recommended): 7-12V

Input Voltage (limits): 6-20V

Digital I/O Pins: 14 (of which 6 provide PWM output)

Analog Input Pins: 6

DC Current per I/O Pin: 40 mA

DC Current for 3.3V Pin: 50 mA

Flash Memory: 32 KB of which 0.5 KB used by bootloader

SRAM: 2 KB (ATmega328)

EEPROM: 1 KB (ATmega328)

Clock Speed: 16 MHz

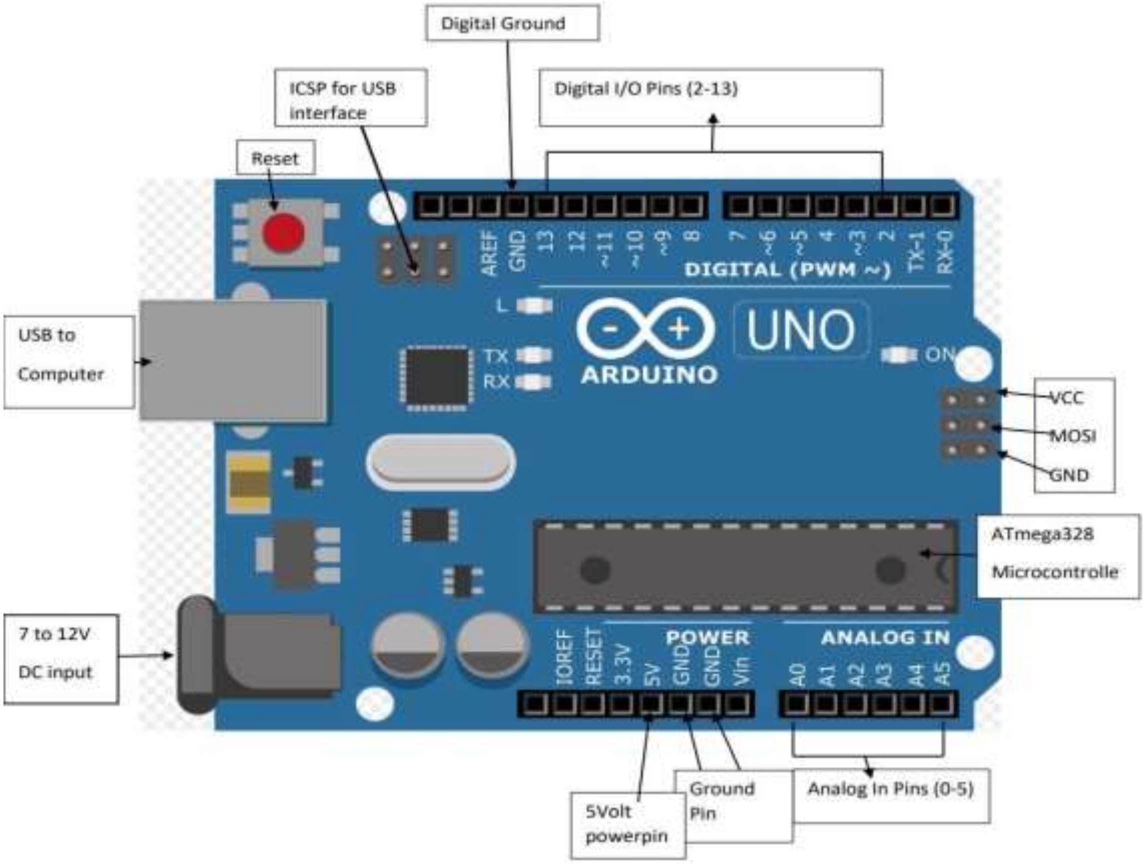


Fig 4.1:Arduino Uno

- **Arduino Hardware part:**

Hardware made with Arduino is open-source. The hardware reference designs are offered on the Arduino website and published under a Creative Commons Attribution Share-Alike 2.5 licence. There are additional layout and production files available for various hardware versions.

The developers have asked that the name Arduino to be reserved for the official product and not used for derived works without permission, even though the hardware and software blueprints are publicly available under copyleft licences. The project is open to integrate contributions from others into the official product, according to the official policy paper on the usage of the Arduino name. Many Arduino-compatible devices that were commercially produced had names that ended in "-Duino" to avoid utilizing the project name.

An Atmel 8-bit AVR microcontroller (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560) with variable amounts of flash memory, pins, and features is the main component of most Arduino boards. In 2012, the 32-bit Arduino Due based on the Atmel SAM3X8E was released. Female headers or single- or double-row pins are used on the boards to provide connections for programming and circuit integration. These could link up with shield-style auxiliary modules. An I2C serial bus may allow for the independent addressability of many and possibly layered shields. A 5 V linear regulator and a 16 MHz crystal oscillator or ceramic resonator are typically included on boards. Due to unique formfactor limitations, some designs, like the Lilypad, operate at 8 MHz without an inbuilt voltage regulator.

The boot loader that comes pre-installed on Arduino microcontrollers makes it easier to upload programmes to the on-chip flash memory. The optiboot bootloader is the Arduino UNO's standard bootloader. Programme code is loaded onto boards using a serial link to another computer. A level shifter circuit can convert between RS-232 logic levels and transistor-transistor logic (TTL) level signals on some serial Arduino boards. A USB-toserial adapter chip like the FTDI FT232 is used to implement the Universal Serial Bus (USB), which is the current programming protocol for Arduino boards. Some boards, like later-model Uno boards, replace the FTDI chip with a separate AVR chip that contains USBto-serial firmware and can be updated via its own ICSP interface.

The majority of the microcontroller's I/O pins are accessible on the Arduino board for usage by other circuits.

14 digital I/O pins, six of which may generate pulse-width modulated signals, and six analogue inputs, which can also be used as six digital inputs, are provided by the Diecimila, Duemilanove, and contemporary Uno.

pins for digital I/O. These pins are connected to female 0.1-inch (2.54 mm) headers on the top of the circuit board.

There are also a number of plug-in application shields that are bought commercially. The male header pins on the underside of the Bare Bones Board and Boarduino boards, as well as the Arduino Nano, may be compatible with solderless breadboards.

- **Arduino Software part:**

IDE

Java was used to create the cross-platform Arduino integrated development environment (IDE), which is available for Windows, macOS, and Linux. It came from the IDE for the programming languages Wiring and Processing. It has a code editor with tools for text copying and pasting, text replacement, automated indenting, brace matching, and syntax highlighting. It also offers straightforward one-click compiling and uploading tools for Arduino programmes. A hierarchy of operating menus, a message area, a text terminal, a toolbar with buttons for standard functions, and more are also included. The GNU General Public Licence, version 2, governs the publication of the IDE's source code.

The Arduino IDE has specific code organization guidelines to support the languages C and C++. A software library from the Wiring project, which offers numerous standard input and output operations, is provided by the Arduino IDE. For the sketch to start and the main programme loop, user-written code only needs two fundamental functions, which are combined with a programme stub main () to create an executable cyclic executive programme using the GNU toolchain, which is also distributed with the IDE. The executable code is transformed via the Arduino IDE's use of avrdude into a text file with hexadecimal encoding, which is then loaded into the Arduino board by a loader programme in the firmware.

- **Sketch**

A program written with the Arduino IDE is called a sketch.

Sketches are saved on the development computer as text files with the file extension. ino.

Arduino Software (IDE) pre-1.0 saved sketches with the extension. pde.

Two functions are all that a simple Arduino C/C++ programme needs to do:

After a power-up or reset, a sketch will only ever use setup () once. Variables, input, and output pin modes, as well as additional libraries required by the sketch are initialised using it. loop (): The main programme periodically calls the procedure loop () after calling setup ().

Until the board is turned off or reset, it is in control.

Blink Example

A load resistor and light-emitting diode (LED) are commonly found on Arduino boards, and they are connected between pin 13 and ground, which is a useful feature for many testing and programme operations. An LED is continually blinking in a normal Arduino programme for a beginner.

The internal libraries built into the IDE environment provide the pin Mode (), digital Write (), and delay () functions, all of which are used by this programme. A new Arduino board is often pre-programmed with this code by the maker.

- **LoRa SX1278**

The SX1278 is a specific type of Long Range (LoRa) transceiver module that operates on the LoRaWAN protocol. LoRa is a wireless communication technology known for its long-range capabilities and low power consumption, making it well-suited for Internet of Things (IoT) applications. The SX1278 LoRa module, developed by Semtech Corporation, is one of the popular choices for implementing LoRa-based communication in various IoT projects.

- **Features of Sx1278**

I. Long Range: The SX1278 LoRa module is designed to provide long-range communication, allowing data transmission over several kilometers in open areas. Its ability to penetrate obstacles and walls makes it suitable for applications in urban and rural environments.

II. Low Power Consumption: LoRa technology is known for its low power consumption, making it ideal for battery-powered devices and IoT applications where energy efficiency is crucial. The SX1278 module ensures efficient energy usage, extending the battery life of connected devices.

III. Low Data Rate: LoRa technology operates at relatively low data rates, typically ranging from a few hundred bits per second (bps) to tens of kilobits per second (kbps). This low data rate is ideal for applications where long-range communication and power efficiency take precedence over high data throughput.

IV. Secure Communication: The SX1278 LoRa module offers secure communication using encryption algorithms to protect data transmission from unauthorized access or tampering. This ensures the confidentiality and integrity of the transmitted data.

V. Frequency Bands: The SX1278 module supports different frequency bands, including 433 MHz, 868 MHz, and 915 MHz, depending on regional regulations and requirements. Users need to select the appropriate frequency band based on their location and local regulations.

VI. SPI Interface: The SX1278 LoRa module communicates with the microcontroller or host device through a Serial Peripheral Interface (SPI). This interface enables easy integration with a wide range of microcontrollers and platforms.

VII. Integration with LoRaWAN Protocol: The SX1278 module is often used in conjunction with the LoRaWAN protocol, a networking standard for Low-Power Wide-Area Networks (LPWANs). LoRaWAN provides additional features for managing large-scale IoT deployments and facilitates seamless integration with cloud-based platforms.

- **Application of SX1278: -**

I. Smart agriculture for monitoring soil moisture and environmental conditions.

II. Environmental monitoring systems for air quality, temperature, and humidity measurement.

III. Asset tracking and supply chain management.

IV. Smart city applications for parking, waste management, and street lighting.

V. Industrial automation and remote monitoring applications.

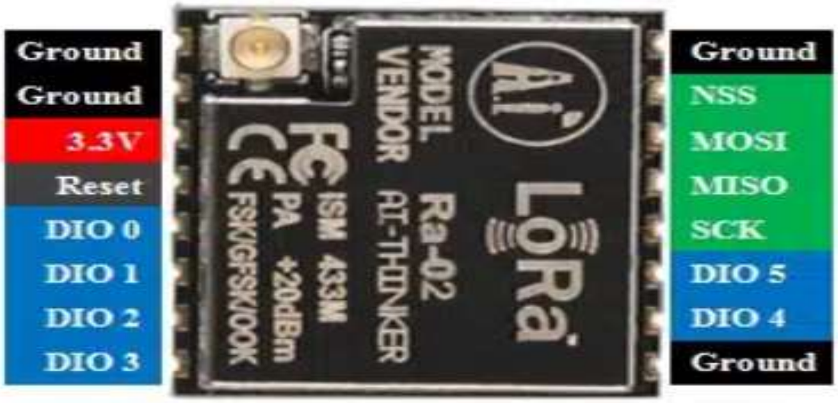


Fig 4.2: LoRa SX1278 Module

- **Temperature and Humidity (DHT11):**

DHT11 Specification:

Operating Voltage: 3.5V to 5.5V

Operating current: 0.3mA (measuring) 60uA (standby)

Output: Serial data

Temperature Range: 0°C to 50°C

Humidity Range: 20% to 90%

Resolution: Temperature and Humidity both are 16-bit

Accuracy: ±1°C and ±1%

- **Overview**

The DHT11 is a popular and low-cost digital temperature and humidity sensor used in various electronics and IoT projects. It is commonly used in applications where monitoring ambient temperature and humidity is essential. The DHT11 sensor is simple to use, making it an ideal choice for beginners and hobbyists. The DHT11 sensor utilizes a humidity-sensitive capacitive element and a thermistor to measure temperature and humidity. When exposed to the surrounding environment, the capacitive element's electrical capacitance changes with humidity variations, while the thermistor's resistance changes with temperature fluctuations. The sensor's internal circuitry converts these changes into digital signals that can be read by a microcontroller.

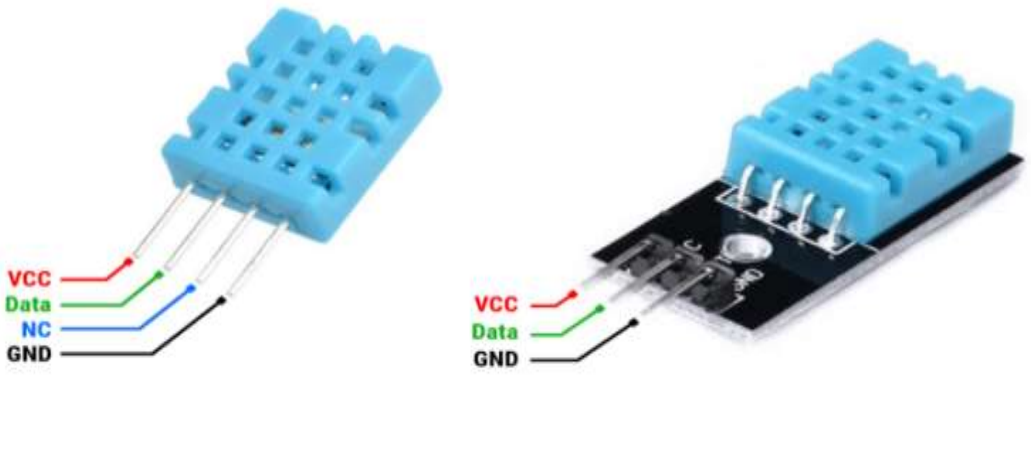


Fig 4.3 Temperature and Humidity Module-DHT11

- **Pin Descriptions:**

The DHT11 sensor has four pins that serve specific functions for communication with external microcontrollers or devices. These pins are labeled on the sensor module and are typically color-coded. Here are the pin descriptions of the DHT11 sensor:

VCC (or +): This pin is the power supply input for the DHT11 sensor. It should be connected to a 3.3V or 5V power source, depending on the sensor's voltage compatibility. Connecting the sensor to a higher voltage than its specified operating voltage may damage the sensor.

Data (or OUT): The data pin is used for both input and output of digital data to and from the DHT11 sensor. It communicates with the external microcontroller or device using a single-wire communication protocol. The data pin sends the temperature and humidity readings from the sensor to the microcontroller, and the microcontroller sends commands to the sensor for data acquisition.

NC (or Not Connected): This pin is not used and remains unconnected. It serves no function and is often left unconnected in the circuit.

GND (or -): The ground pin is connected to the ground (0V) of the power supply to complete the circuit.

- **Gas Sensor (MQ-7)**

MQ-7 Specification:
Gas Detected: Carbon Monoxide (CO)
Operating voltage: DC 5 V.
Vh(high):5.0V±0.1V , Vh(low): 1.5V±0.1V
Heating consumption: About 350mW
Relative humidity: Less than 95%RH
Detecting range : 20ppm-2000ppm carbon monoxide

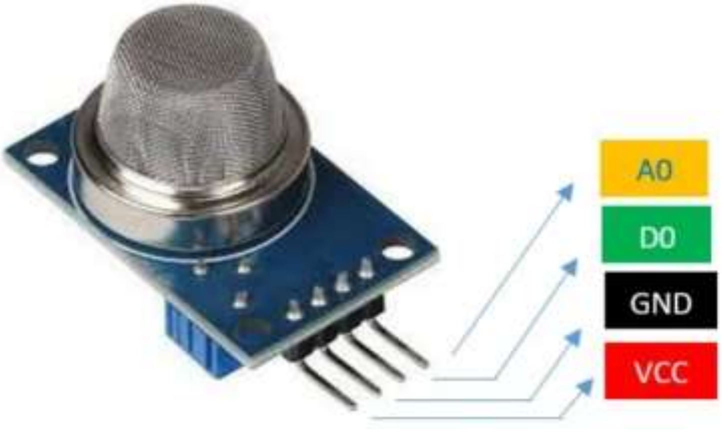


Fig4.4:MQ-7.

- **Pin Descriptions:**

VCC (or +): This pin is the power supply input for the MQ-7 gas sensor. It should be connected to a regulated 5V DC power source to provide power to the sensor's internal components.

GND (or -): The ground pin is connected to the ground (0V) of the power supply to complete the electrical circuit.

AO (Analog Output): The AO pin provides an analog voltage output that varies based on the detected CO gas concentration. This pin can be connected to an analog-to-digital converter (ADC) of a microcontroller to convert the analog voltage into a digital value for CO gas concentration measurement.

DO (Digital Output): The DO pin provides a digital output that switches high or low depending on whether the detected CO gas concentration crosses a predefined threshold. The threshold can be set using an onboard potentiometer. The DO pin is commonly used to trigger an alert or signal when the CO gas concentration exceeds a certain level.

- **Overview:**

This is a MQ7 Gas sensor (CO), suitable for sensing Carbon Monoxide concentrations(PPM) in the air. MQ7 Gas sensor can measure CO concentrations ranging from 20 to 2000 ppm. This sensor has a high sensitivity and fast response time. The MQ-7 sensor requires a warm-up time before it stabilizes and provides accurate readings. This warm-up time is usually specified in the sensor's datasheet and is typically a few minutes.

The sensor’s output is an analog resistance. MQ7 gas sensor has high sensitivity to Carbon Monoxide. The sensor could be used to detect different gases contains CO, it is with low cost and suitable for different application. They are used in gas detecting equipment for carbon monoxide(CO) in family and industry or car.

- **Software Application**

The software application of the Air Quality and Temperature Humidity Monitoring System involves various components that work together to collect, process, and transmit sensor data to a cloud-based platform. The software is responsible for interfacing with the hardware components, data processing, data transmission, and data visualization.

Arduino Firmware:

The Arduino microcontroller serves as the central processing unit of the system. Its firmware is responsible for the following tasks:

- Sensor Data Reading: The firmware reads data from the DHT11 sensor to measure temperature and humidity, as well as from the MQ7 sensor to detect carbon monoxide (CO) levels in the surrounding air.
- Data Processing: After acquiring the sensor data, the firmware processes the raw data to convert it into meaningful units (e.g., Celsius for temperature and percentage for humidity).
- LoRa Communication: The firmware utilizes the SX1278 LoRa module to establish communication with the cloud platform. It packages the processed sensor data into packets suitable for LoRa transmission.

Cloud Platform Interface:

The cloud platform serves as the central hub for receiving, storing, analyzing, and visualizing the transmitted data.

- Data Reception: The cloud application should be designed to receive data packets sent by the Arduino over the LoRa network. It should be capable of handling incoming data from multiple monitoring stations if the system is scalable.
- Data Storage: The received data is stored securely in a database or a storage system for further analysis and historical tracking.
- Data Analysis: The cloud platform can perform data analytics to identify trends, patterns, and anomalies in the air quality data over time. This analysis helps in understanding long-term changes and potential pollution sources.
- Data Visualization: The cloud platform should provide an intuitive and user-friendly interface that allows users to view real-time and historical data through graphs, charts, maps, and other visualizations. This helps in easily interpreting the air quality information.

Integration and API:

For seamless communication between the Arduino and the cloud platform, Application Programming Interfaces (APIs) or other integration methods are utilized. These APIs allow data to be transmitted securely and efficiently from the Arduino to the cloud platform, as well as enable data retrieval and visualization.

- **Hardware-Arduino Uno**

C programming language is employed to program the Arduino Uno. C is a highly robust language suitable for hardware programming. Loops are incorporated, and they are skipped if our voice doesn't meet a specific parameter, allowing the system to progress to the next loop. Eight loops are utilized to ensure smooth operation and maintain the code's flow.

The Arduino code receives a string command from the application and performs actions accordingly. It toggles the assigned pin on or off based on whether the string matches the predefined variables.

4.2 Pseudocode for Transmitted Side

```
Initialize DHT11 sensor on pin 3
Initialize TagoIO communication using Serial interface


Setup:
  Start Serial communication at 9600 baud rate
  Wait until Serial communication is established


  Initialize DHT11 sensor


Main Loop:
  Read temperature from DHT11 sensor and store it in a variable called "temperature"
  Read humidity from DHT11 sensor and store it in a variable called "humidity"
  Read gas value from MQ7 sensor using analogRead() function and store it in a variable called "mq7Value"


  Prepare the data payload in JSON-like format:
    Create a String variable called "payload"
    Add "temperature" value to the payload in JSON format
    Add "humidity" value to the payload in JSON format
    Add "mq7Value" value to the payload in JSON format


  Send the payload data to TagoIO cloud platform using the TagoIO.payload() function with the provided device token


  Wait for 5 seconds (adjust this delay based on your requirements) before the next iteration


End of Main Loop
```

4.3 Pseudocode for Receiver Side:

```
// Include necessary libraries

// Define LoRa and TagoIO configurations
BAND = 868E6
NSS_PIN = 10
RST_PIN = 9
DIO0_PIN = 2
TOKEN = "YOUR_TAGOIO_DEVICE_TOKEN"

// Setup function
function setup():
    // Initialize serial communication
    Serial.begin(9600)
    wait until Serial is ready

    // Configure LoRa pins and initialize LoRa
    LoRa.setPins(NSS_PIN, RST_PIN, DIO0_PIN)
    if LoRa.begin(BAND) is False:
        Print "LoRa initialization failed. Check your connections."
        End the program

    // Initialize TagoIO with the Serial communication
    TagoIO.begin(Serial)

// Loop function
function loop():
    if LoRa has received a packet:
        Initialize an empty string called receivedData

        while LoRa is available to read:
            Append the read character to receivedData
        Print "Received Data: " + receivedData

        // Send data to TagoIO using the specified TOKEN
        TagoIO.payload(TOKEN, receivedData)

    // Add a delay to control how frequently the loop runs
    Delay for 1 seconds
```

5.Results and Discussion

Every connection is done in accordance with the circuit design shown above.

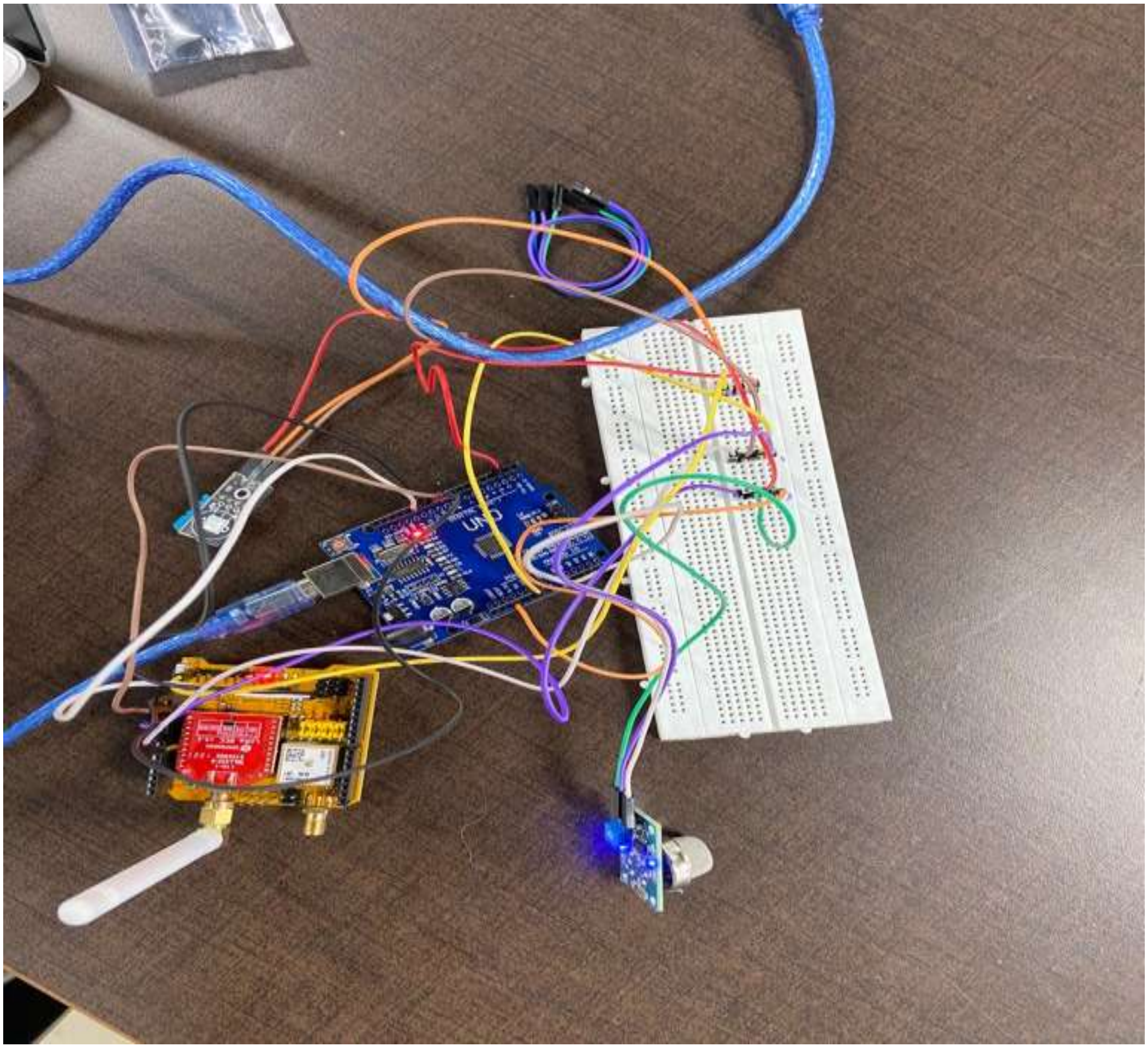
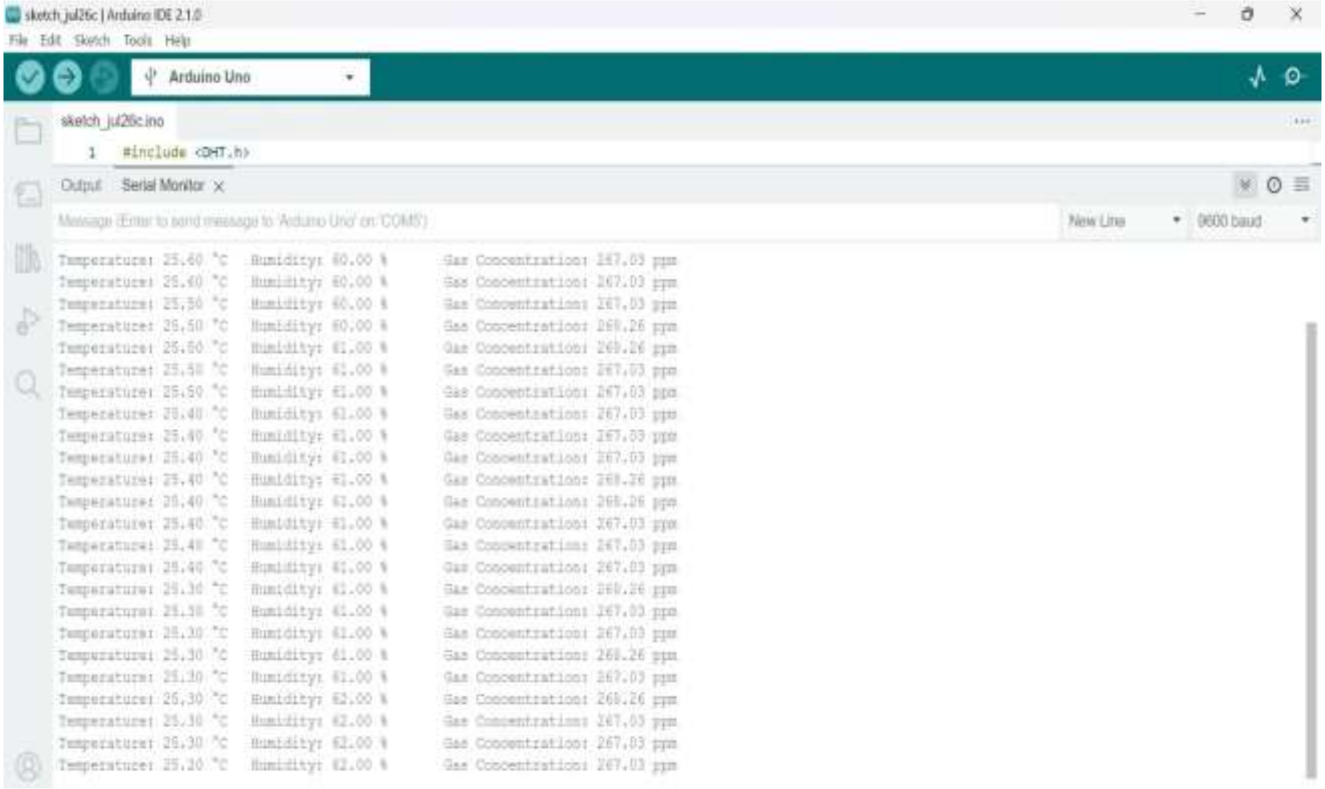


Fig 5.1: Transmitter Side Connection

The processed data, including temperature, humidity, and carbon monoxide levels, is encapsulated into packets suitable for LoRa transmission. These data packets are sent wirelessly via the SX1278 LoRa module, utilizing the LoRaWAN protocol for communication over long distances with low power consumption.

The Arduino board reads the sensor data periodically from the DHT11 and MQ7 sensors. It then processes this data, converting the analog sensor readings into digital format and performing any necessary calibration or data conditioning.


sketch_jul26c | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Arduino Uno
sketch_jul26c.ino
Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM5')
New Line
9600 baud

```
#include <DHT.h>
```

```
Temperature: 25.60 °C   Humidity: 60.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.60 °C   Humidity: 60.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.50 °C   Humidity: 60.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.50 °C   Humidity: 60.00 %      Gas Concentration: 268.26 ppm
Temperature: 25.50 °C   Humidity: 61.00 %      Gas Concentration: 268.26 ppm
Temperature: 25.50 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.50 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %      Gas Concentration: 268.26 ppm
Temperature: 25.40 °C   Humidity: 61.00 %      Gas Concentration: 268.26 ppm
Temperature: 25.40 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.30 °C   Humidity: 61.00 %      Gas Concentration: 268.26 ppm
Temperature: 25.30 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.30 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.30 °C   Humidity: 61.00 %      Gas Concentration: 268.26 ppm
Temperature: 25.30 °C   Humidity: 61.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.30 °C   Humidity: 62.00 %      Gas Concentration: 268.26 ppm
Temperature: 25.30 °C   Humidity: 62.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.30 °C   Humidity: 62.00 %      Gas Concentration: 267.03 ppm
Temperature: 25.20 °C   Humidity: 62.00 %      Gas Concentration: 267.03 ppm
```

Fig 5.2: Transmitter Side Data

The Arduino board reads the sensor data periodically from the DHT11 and MQ7 sensors. It then processes this data, converting the analog sensor readings into digital format and performing any necessary calibration or data conditioning.

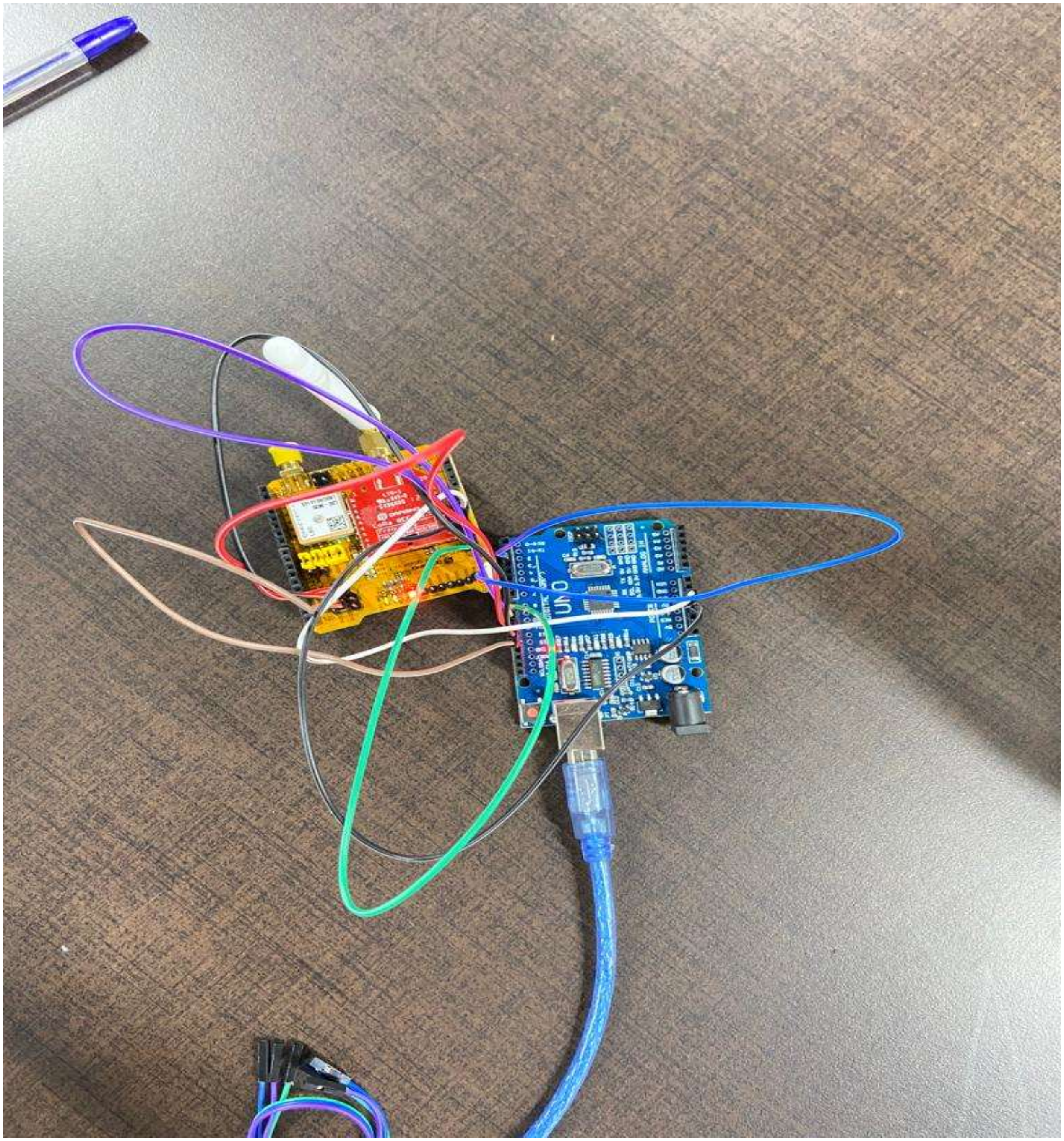
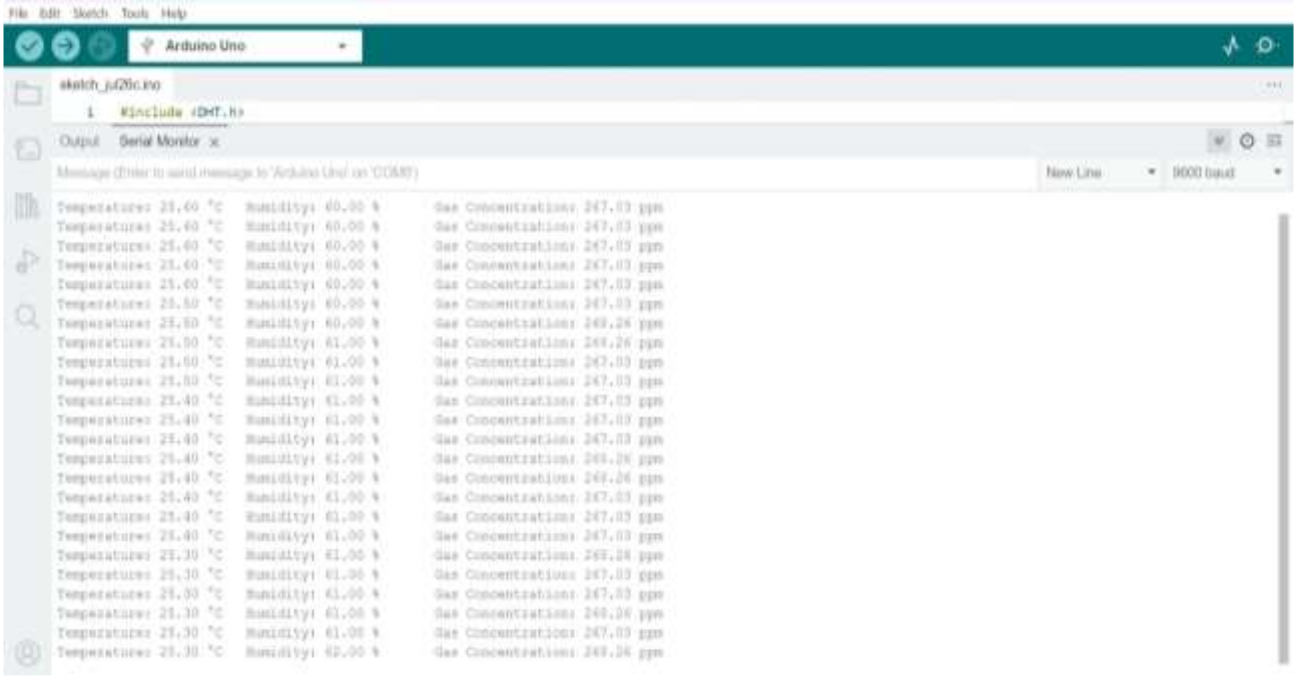


Fig 5.3: Receiver Side Connection



File Edit Sketch Tools Help
Arduino Uno
sketch_jul26c.ino
Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM3')
New Line
9600 baud

```
#include <DHT.h>
```

```
Temperature: 25.60 °C   Humidity: 60.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.60 °C   Humidity: 60.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.60 °C   Humidity: 60.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.60 °C   Humidity: 60.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.60 °C   Humidity: 60.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.50 °C   Humidity: 60.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.50 °C   Humidity: 60.00 %       Gas Concentration: 268.26 ppm
Temperature: 25.50 °C   Humidity: 61.00 %       Gas Concentration: 268.26 ppm
Temperature: 25.50 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.50 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %       Gas Concentration: 268.26 ppm
Temperature: 25.40 °C   Humidity: 61.00 %       Gas Concentration: 268.26 ppm
Temperature: 25.40 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.40 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.30 °C   Humidity: 61.00 %       Gas Concentration: 268.26 ppm
Temperature: 25.30 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.30 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.30 °C   Humidity: 61.00 %       Gas Concentration: 268.26 ppm
Temperature: 25.30 °C   Humidity: 61.00 %       Gas Concentration: 267.03 ppm
Temperature: 25.30 °C   Humidity: 62.00 %       Gas Concentration: 268.26 ppm
```

Fig 5.4: Receiver side Data

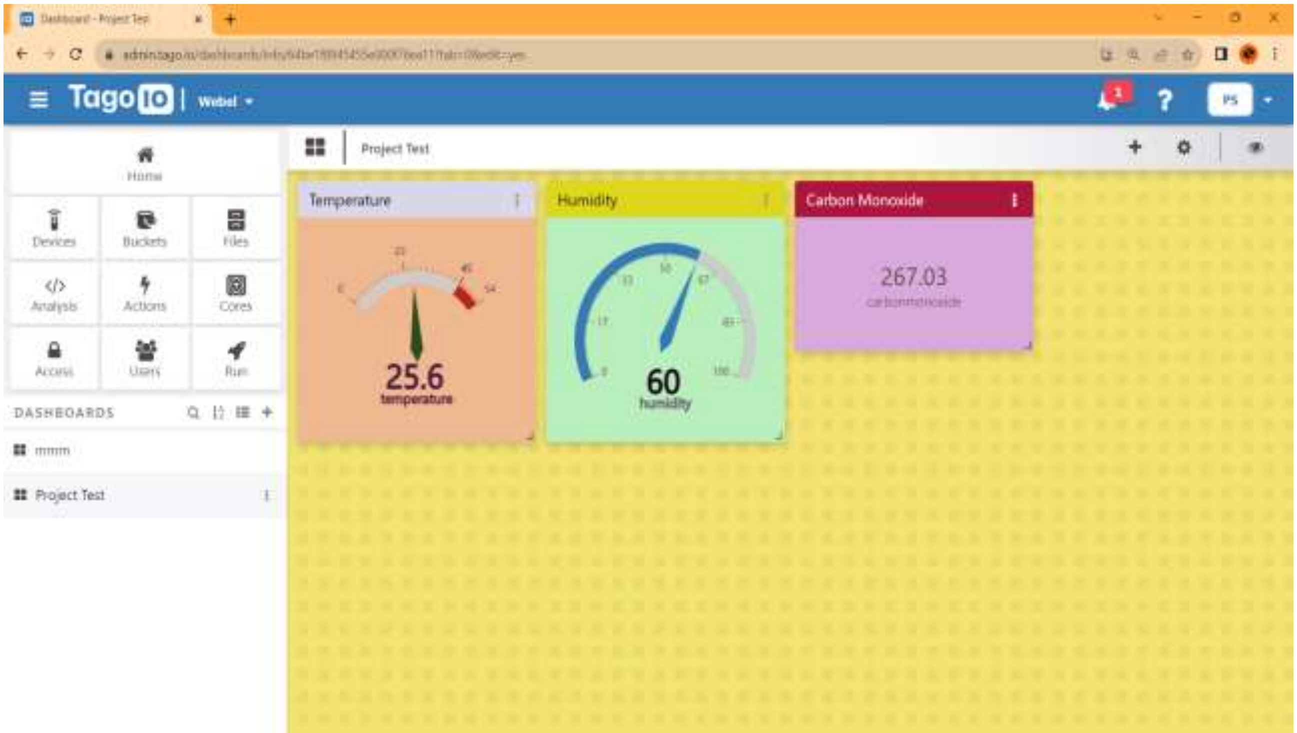
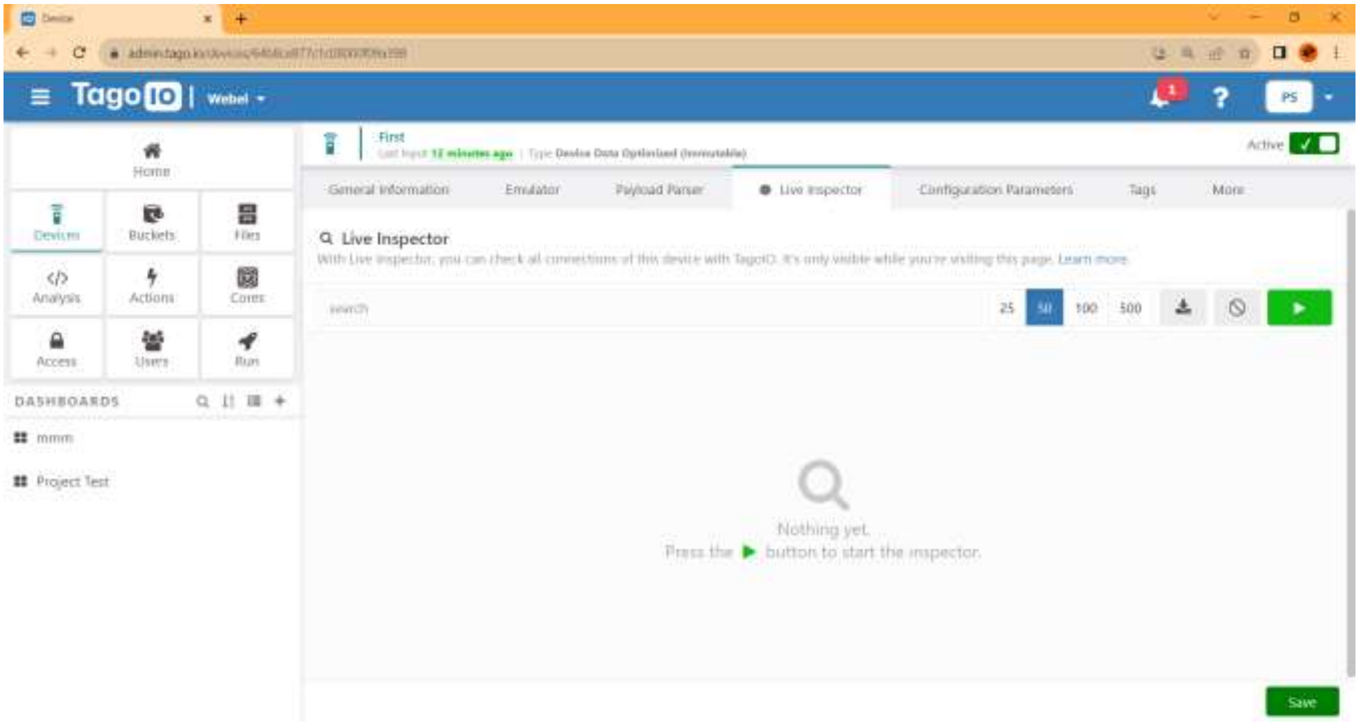


Fig 5.5: Dash Board of Cloud .



Live Inspector

With Live Inspector, you can check all connections of this device with TagoIO. It's only visible while you're visiting this page. Learn more

Nothing yet.

Press the ▶ button to start the inspector.

Fig 5.6: Here Live Data will Store

6.Summary and Conclusions

6.1 Summary of achievements

Successful Sensor Integration: The project achieved successful integration of the DHT11 and MQ7 sensors with the Arduino board. The DHT11 accurately measures temperature and humidity, while the MQ7 reliably detects carbon monoxide levels in the air.

LoRa Communication Implementation: The SX1278 LoRa module was effectively employed for wireless communication between the sensor node and gateway node. LoRa's long-range capabilities allowed for data transmission over considerable distances, making it suitable for outdoor applications.

Real-time Data Transmission: Data collected from the sensor node was transmitted in real-time to the cloud server via the gateway node. This allowed for continuous monitoring and analysis of air quality, temperature, and humidity from remote locations.

Cloud Data Storage and Visualization: The cloud server was set up to receive, store, and display the incoming sensor data. Data visualization tools were implemented to provide easy-to-understand graphical representations, facilitating data analysis and interpretation.

Environmental Insights: The continuous monitoring of air quality and environmental conditions provided valuable insights into air pollution levels, temperature variations, and humidity changes. These insights could aid in making informed decisions for environmental management and public health.

Practical Implementation: The project demonstrated a practical and functional solution for real-world air quality and environmental monitoring. It serves as a foundation for potential applications in smart cities, agriculture, and industrial settings.

6.2 Main Difficulties Encountered and How they were Tackled

During the course of the project, several challenges and difficulties were encountered. Here are the main difficulties faced and how they were tackled:

Sensor Calibration and Accuracy:

- Difficulty: Ensuring accurate readings from the DHT11 and MQ7 sensors was challenging, as they require proper calibration to provide reliable data.
- Solution: Calibration procedures for each sensor were thoroughly researched and implemented. Calibration coefficients were applied to the raw sensor data to improve accuracy. Additionally, periodic calibration checks were performed to maintain data integrity.

LoRa Communication Range and Interference:

- Difficulty: LoRa communication can be affected by physical obstructions and interference, limiting the effective range of data transmission.
- Solution: To tackle this, the placement of the gateway node was optimized for optimal coverage and signal strength. The use of external antennas for both the sensor and gateway nodes helped improve communication range. The project also implemented error-checking and data retransmission mechanisms to ensure data integrity.

Cloud Data Transmission and Security:

- Difficulty: Establishing a secure connection between the gateway node and the cloud server for data transmission required careful consideration of data privacy and security.
- Solution: Secure data transmission protocols, such as SSL/TLS, were used to encrypt the data during transmission. API keys or authentication tokens were employed to ensure only authorized devices could access the cloud server. Regular security audits were conducted to identify and address potential vulnerabilities.

Data Visualization and Cloud Integration:

- Difficulty: Integrating the cloud server with data visualization tools to provide real-time monitoring and analysis presented some challenges.

- Solution: The project utilized cloud platforms that offered built-in data visualization services, simplifying the integration process. Custom scripts and dashboards were created to present the data in a user-friendly and intuitive manner.

6.3 Limitation of the project

- Limited Sensor Range and Coverage: The LoRa communication technology used in the project has a limited range, typically a few kilometers in urban areas. This may lead to restricted coverage in large-scale monitoring applications, requiring the deployment of multiple gateway nodes to overcome this limitation.
- Data Latency: Due to the nature of LoRa communication and potential network congestion, there might be some latency in data transmission. In certain time-critical applications, such as emergency response systems, real-time data updates may be delayed, affecting the responsiveness of the monitoring system.
- Limited Gas Detection Capability: The project employs the MQ7 sensor for detecting carbon monoxide (CO) levels. However, for comprehensive air quality monitoring, other pollutants like ozone, nitrogen dioxide, particulate matter, etc., should also be considered. Integrating additional gas sensors would enhance the system's capabilities but might increase complexity and cost.
- Cloud Connectivity Dependency: The system's effectiveness relies on a stable internet connection for data transmission to the cloud server. In areas with poor connectivity or during network outages, data may be lost or delayed, affecting the system's real-time monitoring capabilities.
- Security Concerns: Transmitting data over the cloud introduces security and privacy risks. Ensuring robust data encryption, authentication mechanisms, and proper access controls are essential to safeguard sensitive environmental data from unauthorized access.
- Environmental Impact: The project aims to monitor environmental conditions, but the deployment of electronic devices and their potential impact on the environment through waste generation and energy consumption should be carefully considered.

- Complex Data Analysis: While the project collects valuable data, the analysis and interpretation of large datasets might pose challenges, especially if advanced data analytics and machine learning techniques are required to derive meaningful insights.

6.4 Future Scope of Work

- Smart Grid Implementation: Incorporating a smart grid system would allow the project to be powered by renewable energy sources, such as solar or wind, reducing dependency on traditional battery power and enhancing sustainability.
- Machine Learning for Data Analysis: Utilizing machine learning algorithms to analyze data can provide advanced insights and predictions about environmental patterns and potential pollution trends. ML models can identify anomalies, predict air quality changes, and enable data-driven decision-making for environmental management.
- Integration with IoT Ecosystems: Integrating the environmental monitoring system with other IoT devices and smart city initiatives can create a more interconnected and intelligent urban environment. This could include collaboration with weather stations, traffic management systems, and public health services.
- Community Engagement: Encouraging community engagement by sharing environmental data with the public can raise awareness about air quality and environmental issues. Citizen participation in data collection and analysis can lead to more comprehensive monitoring and promote environmental advocacy.
- Multisensor Integration: To provide comprehensive environmental monitoring, additional sensors can be integrated into the system. This may include sensors for detecting other pollutants such as ozone, nitrogen dioxide, particulate matter, volatile organic compounds (VOCs), etc. Multisensor integration will offer a more holistic view of air quality and environmental conditions.

6.5 Application for Business purpose

- Environmental Monitoring: The project can be deployed in various locations to monitor air quality, temperature, and humidity in real-time. This data can be used by environmental agencies, research institutions, and governments to assess air pollution levels, identify pollution sources, and make informed decisions to improve air quality.

- Public Health: Monitoring air quality is crucial for public health, as poor air quality can lead to respiratory issues and other health problems. The data collected can be used to issue health advisories and alerts, especially during periods of high pollution or extreme weather conditions.

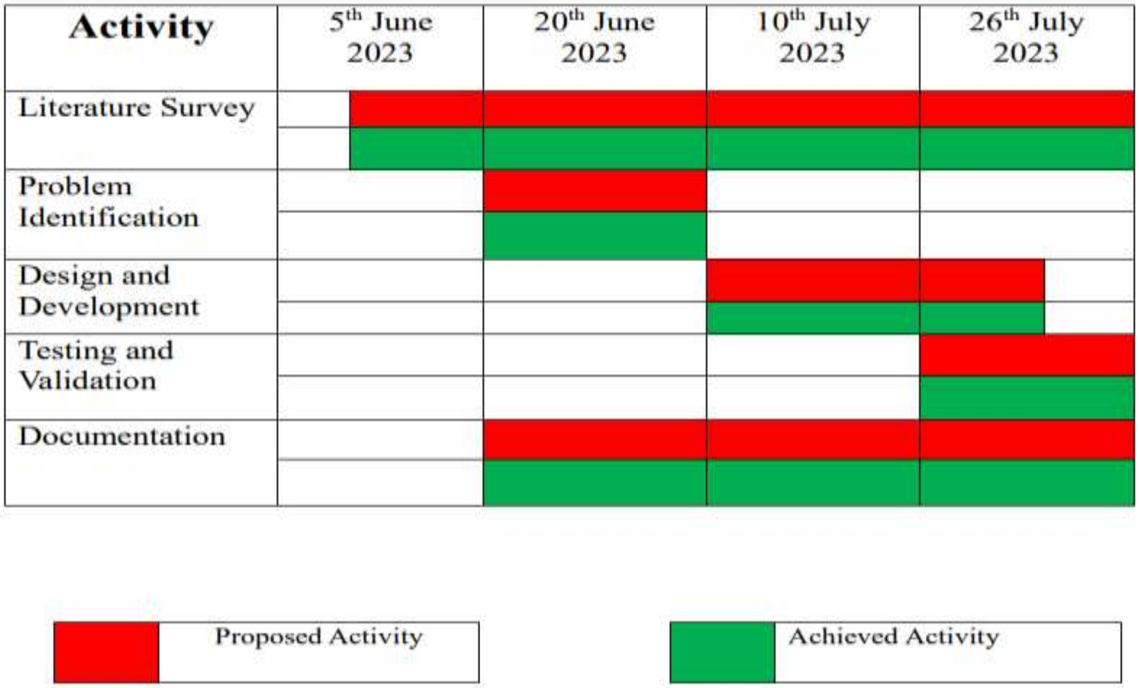
- Industrial Emissions Monitoring*: Industries that emit pollutants such as factories, refineries, and power plants can use the system to monitor their emissions in real-time. This allows them to ensure compliance with environmental regulations, identify potential sources of pollution, and implement timely corrective measures to reduce emissions.

- Transportation and Logistics*: Logistics companies can use the system to monitor environmental conditions during the transportation of sensitive goods, such as pharmaceuticals or perishable items. Real-time tracking of temperature and humidity helps maintain product quality and compliance with transportation regulations.

- Agriculture and Farming*: Agricultural businesses can use the system to monitor environmental conditions in farmlands, greenhouses, and livestock areas. Monitoring temperature and humidity levels can optimize crop growth, prevent livestock stress, and ensure proper climate control in greenhouse environments.

- Community Engagement: By sharing the data with the public, communities can become more aware of their environment and actively participate in improving air quality and living conditions.

7.Gantt Chart



Activity	5th June 2023	20th June 2023	10th July 2023	26th July 2023
Literature Survey	🟥	🟥	🟥	🟥
	🟩	🟩	🟩	🟩
Problem Identification		🟥		
		🟩		
Design and Development			🟥	🟥
			🟩	🟩
Testing and Validation				🟥
				🟩
Documentation		🟥	🟥	🟥
		🟩	🟩	🟩

🟥 Proposed Activity

🟩 Achieved Activity

Fig 7.1 : Gantt Chart

8.References

[1] J. Jo, B. Jo, J. Kim, S. Kim, W. Han
Development of an iot-based indoor air quality monitoring platform
Journal of Sensors, 2020 (2020)

[2] G. Zhao, G. Huang, H. He, Q. Wang
Innovative Spatial-Temporal Network Modeling and Analysis Method of Air Quality
IEEE Access, 7 (2019), pp. 26241-26254

[3] H. Gupta, D. Bhardwaj, H. Agrawal, V. A. 2019 IEEE International Conference on Sustainable Energy Technologies (ICSET), pp. 173–177, 2019

[4] S. Devarakonda, P. Sevusu, H. Liu, R. Liu, L. Iftode, B. Nath
Real-time air quality monitoring through mobile sensing in metropolitan areas
Proceedings of the 2nd ACM SIGKDD international workshop on urban computing, ACM (2013), p. 15

[5] R. Kiruthika, A. Umamakeswari
Low cost pollution control and air quality monitoring system using Raspberry Pi for Internet of Things
2017 International Conference on Energy, Communication, Data Analytics and Soft Computing
(ICECDS), IEEE (2017), pp. 2319-2326

[6] A.K. Saha, S. Sircar, P. Chatterjee, S. Dutta, A. Mitra, A. Chatterjee, S.P. Chattopadhyay, H.N. Saha
A raspberry Pi controlled cloud based air and sound pollution monitoring system with temperature and humidity sensing
2018 IEEE 8th Annual Computing and Communication Workshop and Conference
(CCWC), IEEE (2018), pp. 607-611