INTRODUCTION

1. INTRODUCTION

Air is one of the essential elements of man's surroundings. The earth's atmosphere contains gases such as Nitrogen, Oxygen, Carbon Monoxide, and traces of some rare elements. Humans need an atmosphere that is free from contaminants. This is very crucial for human life and health. Any change in the natural composition of air may cause grave harm to life forms on earth. Air pollution is the presence of one or more contaminants in the atmosphere such as gases in a quantity that can harm humans, animals, and plant. Air pollutants are measured in Parts per Million (ppm) or ug/m3. Primary pollutants are released directly into the atmosphere. Secondary pollutants are produced when the primary pollutant reacts with other atmospheric chemicals. Air quality affects public health. The effect of air pollution ranges from difficulty in breathing, coughing, aggravation of asthma and emphysema. Polluted air can also impair visibility. Air pollution is accountable for the death of 7 million persons worldwide each year or one in eight premature deaths yearly. Almost 570,000 children under the age of five die every year from respiratory infection linked to indoor/outdoor pollution and second-hand smoke. Children exposed to air pollution have an elevated risk of developing chronic respiratory problems such as asthma. In the monitoring of air pollution, several researchers worldwide have developed models to monitor many of the pollutant gases such as Sulphur Dioxide (SO2), Carbon Monoxide (CO), Carbon Dioxide (CO2), Nitrogen Oxides (NO) etc. This paper focuses on the design and implementation of a smart air pollutant monitoring system. It discusses how the level of pollutants in the air can be monitored using a gas sensor, Arduino microcontroller and a Wi-Fi module. The main objective of this project is to design a smart air pollution monitoring system that can monitor, analyses and log data about air quality to a remote server and keep the data up to date over the dashboard.

1.1. BACKGROUND

Air pollution kills an estimated seven million people worldwide every year. WHO data shows that almost all of the global population (99%) breathe air that exceeds WHO guideline limits containing high levels of pollutants, with low- and middle-income countries suffering from the highest exposures. Smaller-diameter particles (PM2.5 or smaller) are generally more dangerous and ultrafine particles (one micron in diameter or less) can penetrate tissues and organs, posing an even greater risk of systemic health impacts. Air pollution also contributes to global warming. A major component of PM2.5 is black carbon, which can absorb one million times more energy from the sun than carbon dioxide. Experts say reducing pollutants such as black carbon could help slow global warming and improve air quality. Last year, India's capital New Delhi recorded the highest concentration of PM2.5 particles to date, at 14 times the safe limit outlined by the World Health Organization (WHO). New Delhi's chief minister blamed air pollution for the coronavirus spike at the time. Several studies now suggest that persistently high levels of air pollution may have significantly increased deaths from Covid-19. A PM2.5 increase of just 1 microgram per cubic metre corresponded to a 15% increase in Covid-19 deaths in US cities, according to one Harvard University study. To measure the PM2.5 if we have an efficient system the quality of air can be checked periodically and thus can improve the quality of air.

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1. RELATED WORK

1. Arumugham, Anand Jayakumar & Yesyand, Praviss & Prashanth, Venkstesh. (2021). IoT Based Air Pollution Monitoring System.
2. Okokpujie, Kennedy & Noma-Osaghae, Etinosa & Odusami, Modupe & John, Samuel & Oluwatosin, Oluga. (2018). A Smart Air Pollution Monitoring System. International Journal of Civil Engineering and Technology. 9. 799-809.
3. A. J. Alabdullah, B. I. Farhat and S. Chtourou, "Air Quality Arduino Based Monitoring System," 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), 2019, pp. 1-5, doi: 10.1109/CAIS.2019.8769529.
4. Yun-Liang He, Shu-Qin Geng, Xiao-Hong Peng, Li-Gang Hou, Xiang-Kai Gao and Jin-Hui Wang, "Design of outdoor air quality monitoring system based on ZigBee wireless sensor network," 2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), 2016, pp. 368-370, doi: 10.1109/ICSICT.2016.7998923.

2.2. EXISTING SYSTEM

2.2.1 POLLUDRONE

The Polludrone Pro is an advanced Air Quality Monitoring Equipment specially developed for special purpose pollution monitoring in the ambient atmosphere. For instance, applications like Airport Air Quality Monitoring, Tunnel Air Quality Monitoring, Industrial Fenceline Monitoring, etc can use the Polludrone Pro for monitoring purposes. Various use cases like ventilation fan automation in tunnels, air quality impact by frequent flights at airports, industrial impact on the surroundings, etc can analyze the air quality by deploying a sensor network of Polludrone Pro.

Advantages:

- Measures the presence of a large number of particles.
- It is robust under harsh ambient conditions
- Long life battery back-up

Disadvantages:

- Shows the values only at a current time. No storage of previous values.
- Dashboard is inbuilt within the system; therefore, it is difficult to check every time.

2.2.2 ARDUINO-BASED AIR QUALITY MONITORING SYSTEM

The system presents a design for a system that aims to notify the residents of VOC's concentration level in both indoor and outdoor environments. The system is Arduino-based, it will monitor and detect total volatile organic compounds (TVOC) and then inform the user via wireless communication system of its levels to take actions. The sole purpose of the system is to notify the user of the abnormal increase in the above-mentioned parameters whether it is an increase in Temperature, CO2, TVOC, or all of the parameters at once. The system is composed of a 12v solar panel to power up the Arduino. We are going to use a voltage regulator to bring the voltage down to 5V. A multi-function sensor (CCS811) will detect temperature, CO2 and TVOC.

The Arduino will process the readings from the sensor. Then, it will send the data to the user via Bluetooth. The data can be seen on a smartphone, using for example the Bluetooth Terminal HC-05 software.

Overview:

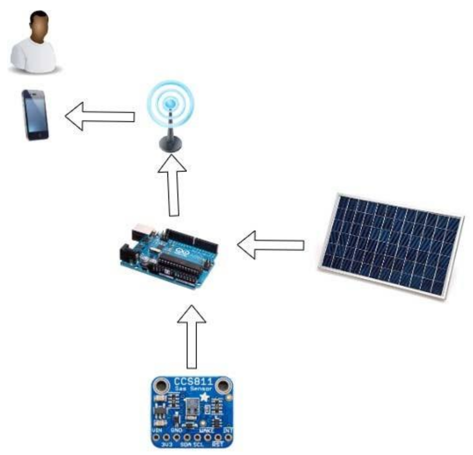


Figure 2.1 - Overview of the Arduino based system

Advantages:

- Helpful in gulf countries where amount of VOC in air would be rather high.

- VOC and CO2 presence can be measured accurately.

Disadvantage:

- Poor dashboard facility
- Detects and valuates only VOC and CO2.

2.2.3 DESIGN OF OUTDOOR AIR QUALITY MONITORING SYSTEM BASED ON ZIGBEE

A wireless network is formed by coordinators, routers, terminal notes, and especially PPB (part per billion) level high precision sensors. Data, sampled and processed by terminal notes, are sent to routers or the coordinator through a serial port, then displayed on a screen of a Pc. The concentration of CO, S02, 03, N02 and NO in the air can be observed online in real time. System architecture, hardware, and software are implemented based on ZigBee wireless sensor network. It has a wide range of uses in the military, industry, agriculture, transportation, medical, home office automation, and other fields.

The design of hardware includes terminal notes, routers and the coordinator. CC2530 chip, a product of T1, is used as the core chip in this design to support building ZigBee network quickly and in low-cost [6]. The chip combines ZigBee protocol named Zstack, which has good performance in industry. The architecture of the chip is divided into three types: CPU and memory module, RF module, and peripherals, clock, power management module. The CC2530 chip integrates a 2.4GHz RF transceiver, an enhanced 8051 MCU of industry-standard, programmable FLASH of maximum 256 KB, 8KB RAM and also offers a broad set of peripherals (including two USART, eight 12-bit ADC and 21 General GPIO.

Circuit structure of terminal nodes:

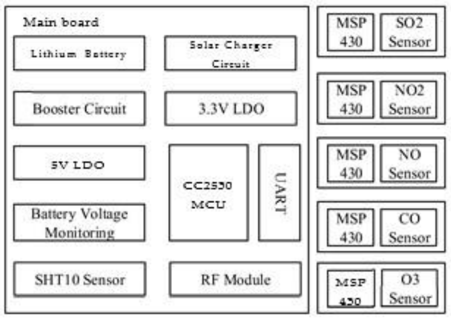


Figure 2.2 – Circuit structure of zigbee based system

Advantages:

- High precision
- Low cost
- Low power consumption
- Different sensors can be connected to network for extended functions and various fields.
- Reduce the complexity and cost of wiring.

Disadvantage:

- Poor dashboard facility.
- Does not detect the value of particulate matter correctly.

2.2.4 DUSTROID

The Dustroid Pro is an Online Dust Monitoring System specially developed for dust levels for critical applications. Dustroid Pro operates on the Laser Scattering Working Principle. Dustroid comes with a heated inlet for dehumidification of air-sample for improved accuracy and stability. The heated inlet reduces dehumidifies the air sample by reducing humidity up to 60%, which makes it an ideal solution for dust monitoring in Humid places. Dustroid Pro measures all the PM parameters like PM1, PM2.5, PM10, PM100, Temperature and Humidity. The Dustroid Pro is technologically equipped to offer higher accuracy with a heated inlet for humidity correction for Particulate monitoring, Auto device firmware updates, an Anti-static inlet to avoid loss of particulate during sampling and remote calibration capabilities.

Figure 2.3 - Dustroid pro

Advantages:

- Measures the presence of a large amount of particles.
- It is robust under harsh ambient conditions.
- Long life battery back-up.

Disadvantages:

- Shows the values only at a current time. No storage of previous values.
- Dashboard is inbuilt within the system; therefore, it is difficult to check every time.

PROBLEM IDENTIFICATION & OBJECTIVES

3. PROBLEM IDENTIFICATION AND OBJECTIVES

3.1. PROBLEM STATEMENT & OBJECTIVES

3.1.1. PROBLEM STATEMENT

Worsening air pollution has been amongst India's most pressing problems in recent years. Air pollution contributes to the premature deaths of 2 million Indians every year. The current scenario shows a need for better and more effective ways of improving air quality across the country, especially in the densest and populated urban spaces.

Even though we have so many air quality monitoring systems available most of the educational institutions does not have one. Many educational institutions are in cities, not too far away from traffic. Air quality monitoring in those areas is essential.

3.1.2. OBJECTIVES

- The primary objective of the project is to build a Green Campus.
- A Green Campus is a place where environmentally friendly practices and education combine to promote sustainable and eco-friendly practices in the campus.
- Monitoring in those areas helps in assessing the level of pollution in relation to the ambient air quality standards.
- PM and AQI values will be calculated and displayed on a dashboard.
- Further a graph will be plotted to compare the quality of air over time.
- Apart from that, the current temperature, atmospheric pressure and humidity will also be displayed on the dashboard.

3.2. SYSTEM STUDY

3.2.1 DRAWBACKS OF EXISTING SYSTEM

- Poor dashboard availability
- Identifying the presence of only few components in certain cases
- Less feasibility in case of polludrone and dustroid

- Most of them does not have a cloud space to store previous values for comparison with existing values so that we can make sure whether the conditions are going good or bad.
- No other additional features other than air quality monitoring

3.2.2 PROPOSED SYSTEM

The system is an advanced real time air quality reporting system supported with Internet Of things (IOT) architecture. The model presented here uses a combination of the Arduino Mega2560 software and hardware along with Gas sensors - MQ135 which help in detecting gases like NO2, CO, CO2, NH3 while measuring their amount decently and based on these values the air quality index (AQI) will be calculated. BME280 is used to measure the temperature and humidity and pressure in the surrounding. PMS7003 is used to get the exact PM values.

Further the readings will be displayed, and a graph will be plotted on a dashboard. Dashboard is built in influxDB database.

Overview:

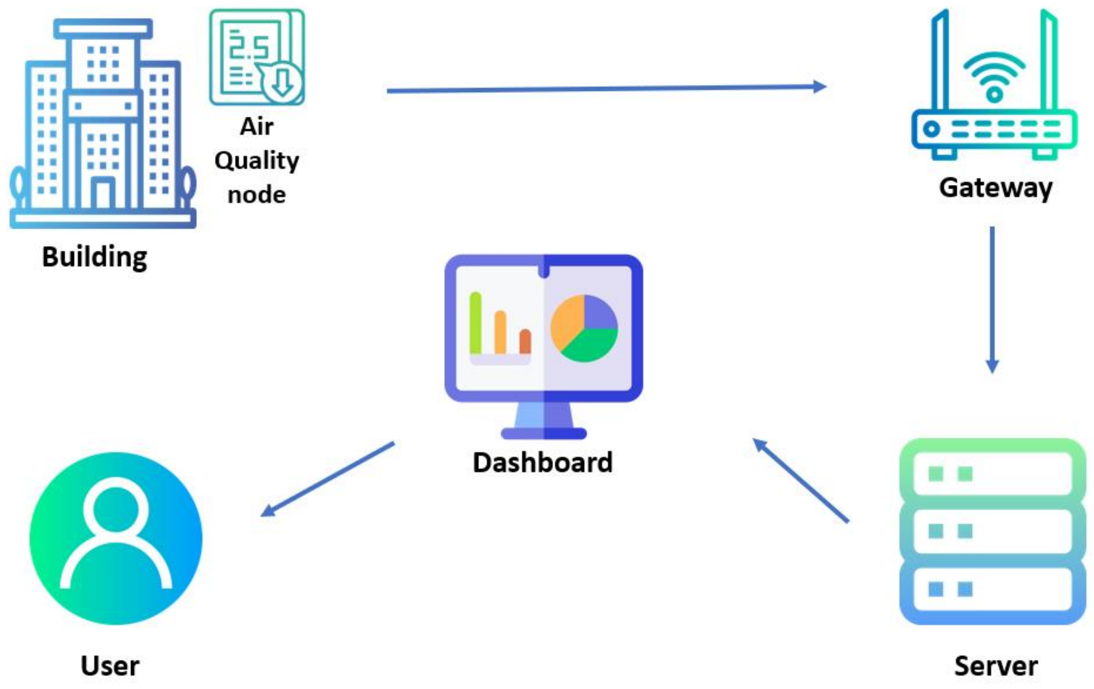


Figure 3.1 - Overview of the project

3.3 FEASIBILITY ANALYSYS

The only concern related to technical feasibility is regarding the gateway for communication. For that the system will be using Lora Wan technology-based gateway. Socially the product will be feasible as it has a visualisation dashboard which helps to view the values from different locations. Also, the system does not radiate any kinds of signals or radiations which pollute the atmosphere. Further it measures the pollutants in atmosphere which is a blessing to the society.

The Economic feasibility includes the cost of the components and cost for installation. The detailed estimation is given in the table below:

Cost Analysis Table:

S No.	**Name of the Component**	**Quantity**	**Cost**
1	The Arduino Mega2560	1	Rs.2,200
2	The Indoor Gateway	1	Rs.23,000
3	BME 280	1	Rs.1,300
4	MQ 135	1	Rs.140
5	PMS 7003	1	Rs.2,260
6	RFM95W	1	Rs.1,220
7	Jump Wire	3 packets	Rs.200
8	Power Supply and Adaptor	1	Rs.600
			Total cost= Rs.30,920

Table 3.1 – Cost Analysis

3.4 SCOPE AND APPLICATIONS

3.4.1 PURPOSE & SCOPE

- The system is used to study if an area surrounding an educational institution has an air pollution problem.
- Monitoring helps in assessing the level of pollution in relation to the ambient air quality standards.
- PM values will be calculated and display on a dashboard.

- Further a graph will be plotted to compare the quality of air over time.

3.4.2 ADVANTAGES

- High quality visualisation dashboard
- PM values of 3 types – PM1, PM2.5 and PM10 will be displayed accurately along with AQI value
- Air quality index of weekends and weekdays can be compared

3.4.3 APPLICATIONS

- It can be used in places like educational institutions, hospitals, hotels, resorts, restaurants, and other firms.
- Professionals like air pollution analysts and climatologist can access the dashboard to perform their analysis and studies.

PROBLEM ANALYSIS & DESIGN

4. PROBLEM ANALYSIS & DESIGN

4.1. HIGH LEVEL DESIGN

- MQ-135 will analyze the quality of air by measuring the concentration of gases like carbon dioxide, ammonia, sulfide, etc.,
- BME280 measures the temperature, pressure, and humidity in the surrounding.
- PMS7003 gets the exact PM values.
- Arduino Mega2560 collects and transport this value to the dashboard via an RF95W module and a gateway.
- Tools in influxDB are used to create the dashboard.
- The gateway is built using Lora Wan technology.

4.1.1. BLOCK DIAGRAM

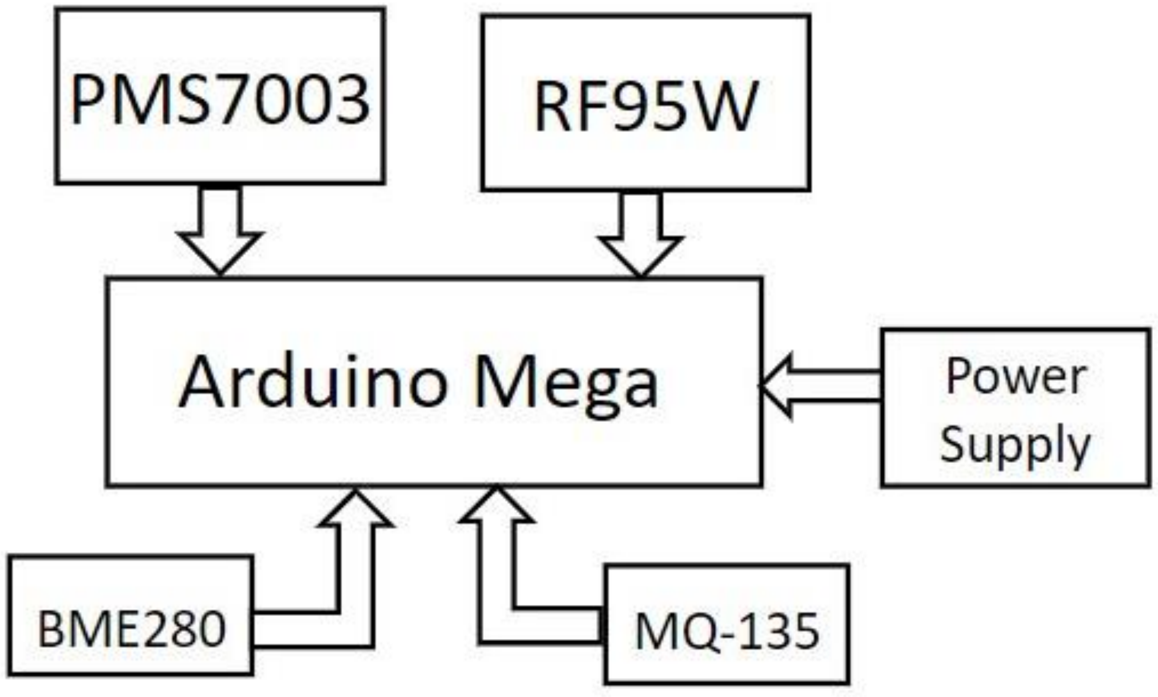


Figure 4.1 Block diagram of Air Quality checking node

4.1.2 CIRCUIT DIAGRAM

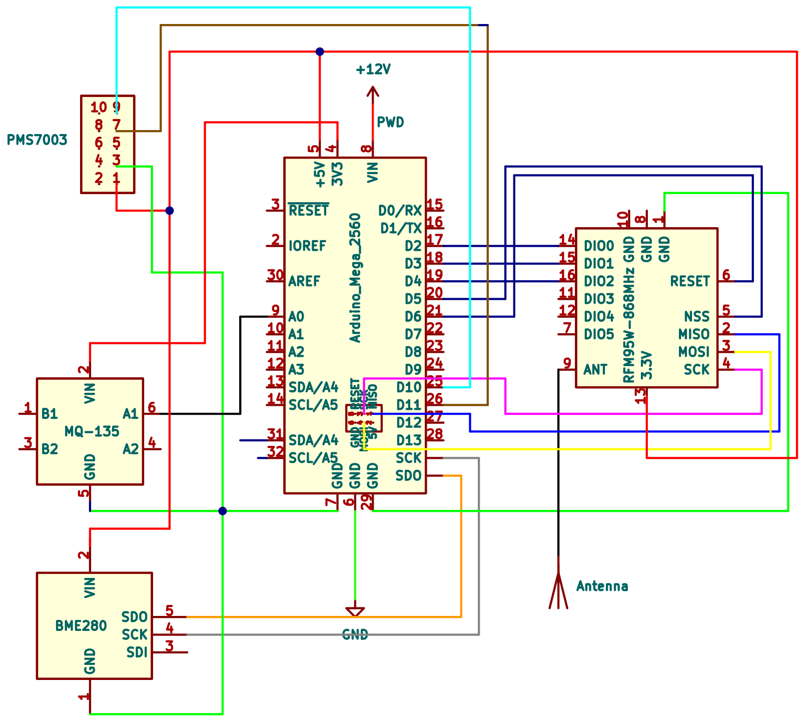


Figure 4.2 Circuit Diagram of Air Quality checking node

4.2 MODULE DESCRIPTION

4.2.1. AIR QUALITY MONITORING

a) ARDUINO MEGA2560

The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila. The Mega 2560 is an update to the Arduino Mega, which it replaces.

Arduino Mega Specifications Table:

MICROCONTROLLER	ATmega2560
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	54 (of which 15 provide PWM output)
ANALOG INPUT PINS	16
DC CURRENT PER I/O PIN	20 Ma
DC CURRENT FOR 3.3V PIN	50 Ma
FLASH MEMORY	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
CLOCK SPEED	16 MHz
LED_BUILTIN	13

Table 4.1 Arduino Mega Specifications

Figure 4.3 Arduino Mega 2560

b) MQ135

The MQ135 is one of the popular gas sensors from the MQ series of sensors that are commonly used in air quality control equipment. It **MQ-135 Gas sensor** can detect gases like Ammonia (NH3), sulfur (S), Benzene (C6H6), CO2, and other harmful gases and smoke and gives the AQI as output based on the concentration of these gases in the atmosphere. Like other MQ series gas sensor, this sensor also has a digital and analog output pin. When the level of these gases goes beyond a threshold limit in the air the digital pin goes high. This threshold value can be set by using the on-board potentiometer. The analog output pin, outputs an analog voltage which can be used to approximate the level of these gases in the atmosphere.

The MQ135 air quality sensor module operates at 5V and consumes around 150mA. It requires some pre-heating before it could actually give accurate results. The pinouts and important components on an MQ135 Module is marked below

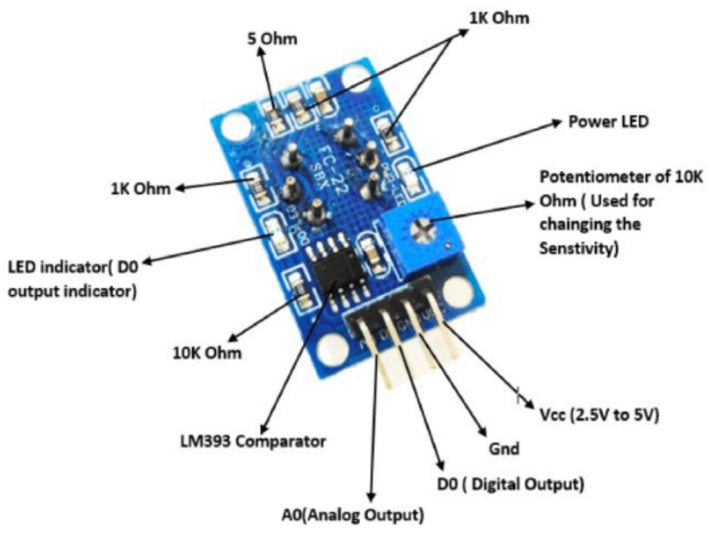


Figure 4.4 MQ135 Pinout

Note that all MQ sensors must be powered up for a pre-heat duration for the sensor to warm up before it can start working. This pre-heat time is normally between 30 seconds to a couple of minutes. When you power up the module the power LED will turn on, leave the module in this state till the pre-heat duration is completed.

Technical Specifications of MQ135 Gas Sensor

- Operating Voltage: 2.5V to 5.0V

- Power consumption: 150mA
- Detect/Measure: NH3, Nox, CO2, Alcohol, Benzene, Smoke
- Typical operating Voltage: 5V
- Digital Output: 0V to 5V (TTL Logic) at 5V Vcc
- Analog Output: 0-5V at 5V Vcc

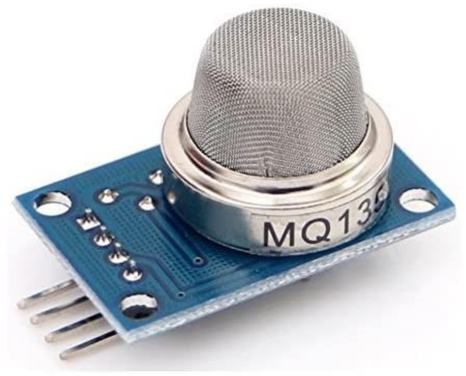


Figure 4.5 MQ135

c) BME280

The BME280 sensor module reads barometric pressure, temperature, and humidity. Because pressure changes with altitude, you can also estimate altitude. There are several versions of this sensor module. The BME280 sensor uses I2C or SPI communication protocol to exchange data with a microcontroller. Writing the code to get the sensor readings is also very straightforward thanks to the BME280_Adafruit library. You just need to use the readTemperature(), readHumidity() and readPressure() methods. You can also estimate altitude using the readAltitude() method.

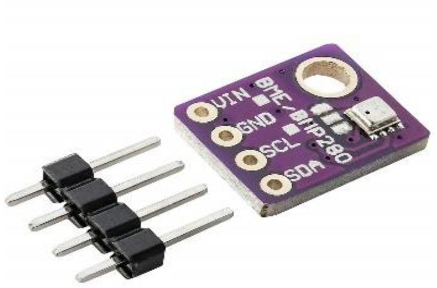


Figure 4.6 BME280

d) PMS7003

PMS7003 is a kind of digital and universal particle concentration sensor, which can be used to obtain the number of suspended particles in the air, i.e., the concentration of particulate matter, and output them in the form of digital interface. It mainly measures the values for PM1, PM2.5 and PM10.

PM1are ultrafine particles with an aerodynamic diameter less than 1 micrometers. Ultra-fine dust is the most damaging variant of fine particles because the particles penetrate directly through the lungs into the bloodstream and are thus spread to the organs.

PM2. 5 refers to particles that have diameter less than 2.5 micrometres (more than 100 times thinner than a human hair) and remain suspended for longer. These particles are formed because of burning fuel and chemical reactions that take place in the atmosphere. It is an air pollutant that is a concern for people's health when levels in air are high. PM2.5 are tiny particles in the air that reduce visibility and cause the air to appear hazy when levels are elevated.

PM10 describes inhalable particles, with diameters that are generally 10 micrometers and smaller. Exposure to high concentrations of PM10 can result in a number of health impacts ranging from coughing and wheezing to asthma attacks and bronchitis to high blood pressure, heart attack, strokes and premature death.

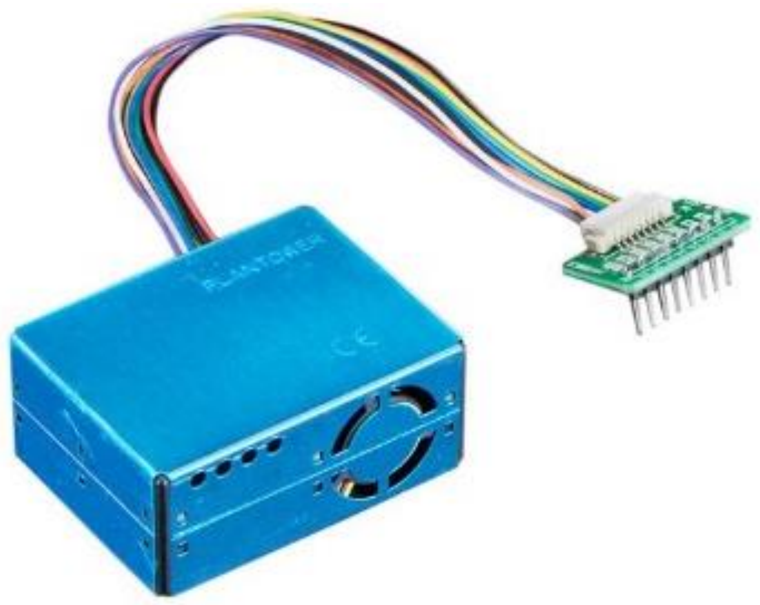


Figure 4.7 PMS7003

4.2.2 DATA TRANSFERRING

a) RFM95W

The RFM95W transceivers feature the LoRaTM long range modem that provides ultra-long range spread spectrum communication and high interference immunity whilst minimising current consumption.

Using Hope RF’s patented LoRaTM modulation technique RFM95W can achieve a sensitivity of over -148dBm using a low-cost crystal and bill of materials. The high sensitivity combined with the integrated +20 dBm power amplifier yields industry leading link budget making it optimal for any application requiring range or robustness. LoRaTM also provides significant advantages in both blocking and selectivity over conventional modulation techniques, solving the traditional design compromise between range, interference immunity and energy consumption.

Specifications:

- LoRaTM Modem.
- 168 dB maximum link budget.
- +20 dBm - 100 mW constant RF output vs. V supply.
- +14 dBm high efficiency PA.
- Programmable bits rate up to 300 kbps.
- High sensitivity: down to -148 dBm.
- Bullet-proof front end: IIP3 = -12.5 dBm.
- Excellent blocking immunity.
- Low RX current of 10.3 mA, 200 nA register retention.
- Fully integrated synthesizer with a resolution of 61 Hz.
- FSK, GFSK, MSK, GMSK, LoRaTM and OOK modulation.
- Built-in bit synchronizer for clock recovery.
- Preamble detection.
- 127 dB Dynamic Range RSSI.
- Automatic RF Sense and CAD with ultra-fast AFC.
- Packet engine up to 256 bytes with CRC.

- Built-in temperature sensor and low battery indicator.
- Modue Size : 16*16mm

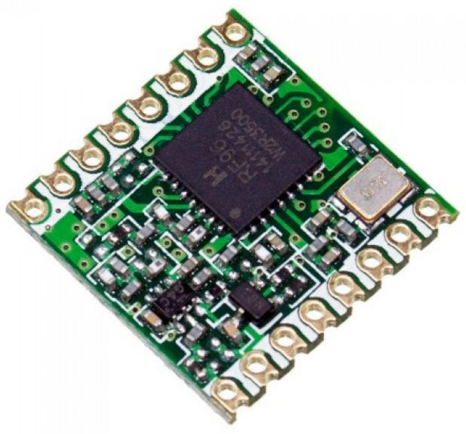


Figure 4.8 RFM95W

b) Dragino Indoor LoRaWAN Gateway

The Dragino is an **open source LoRaWAN Gateway**. It lets you bridge LoRa wireless network to an IP network via WiFi or Ethernet. The LoRa wireless allows users to send data and reach extremely long ranges at low data-rates.

The Dragino uses Semtech packet forwarder and fully compatible with LoRaWAN protocol. It includes a **SX1308 LoRa concentrator**, which provide 10 programmable parallel demodulation paths.

Dragino has pre-configured standard LoRaWAN frequency bands to use for different countries.User can also customized the frequency bands to use in their own LoRa network.

- Open Source OpenWrt system
- Managed by Web GUI, SSH via LAN or WiFi
- Emulates 49x LoRa demodulators
- LoRaWAN Gateway
- 10 programmable parallel demodulation paths
- Remote.it remote management.

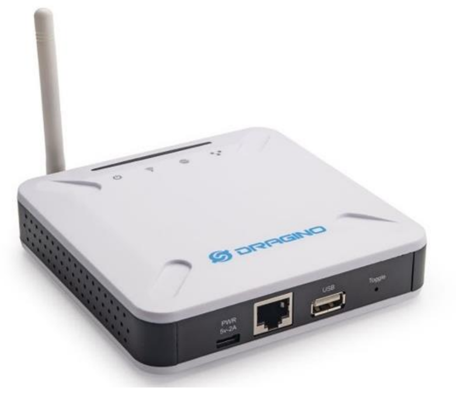


Figure 4.9 Indoor Gateway

LoRa

LoRa is a wireless modulation technique derived from **Chirp Spread Spectrum (CSS)** technology. It encodes information on radio waves using chirp pulses - similar to the way dolphins and bats communicate! LoRa modulated transmission is robust against disturbances and can be received across great distances. LoRa is ideal for applications that transmit small chunks of data with low bit rates. Data can be transmitted at a longer range compared to technologies like WiFi, Bluetooth or ZigBee. These features make LoRa well suited for sensors and actuators that operate in low power mode.

LoRa can be operated on the license free sub-gigahertz bands, for example, 915 MHz, 868 MHz, and 433 MHz. It also can be operated on 2.4 GHz to achieve higher data rates compared to sub-gigahertz bands, at the cost of range. These frequencies fall into ISM bands that are reserved internationally for industrial, scientific, and medical purposes.

What is LoRaWAN?

LoRaWAN is a Media Access Control (MAC) layer protocol built on top of LoRa modulation. It is a software layer which defines how devices use the LoRa hardware, for example when they transmit, and the format of messages.The LoRaWAN protocol is developed and maintained by the LoRa Alliance. The first LoRaWAN specification was released in January 2015. The table below shows the version history of the LoRaWAN specifications. At the time of this writing the latest specifications are 1.0.4 (in 1.0 series) and 1.1 (1.1 series).LoRaWAN is suitable for transmitting small size payloads (like sensor data) over long distances. LoRa modulation provides a significantly greater communication range with low bandwidths than other competing wireless data transmission technologies.

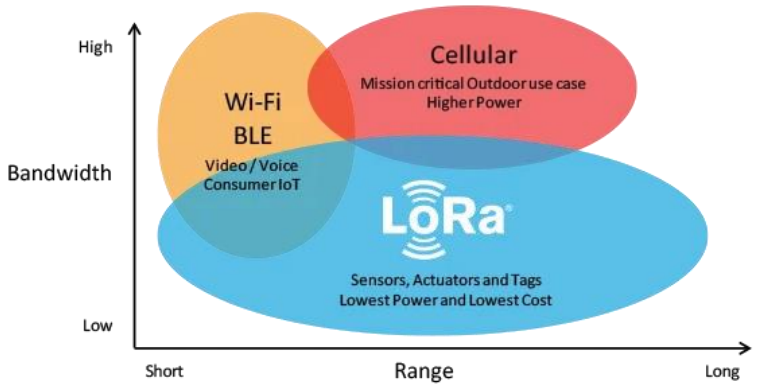

High
Bandwidth
Low
Wi-Fi
BLE
Video / Voice
Consumer IoT
Cellular
Mission critical Outdoor use case
Higher Power
LoRa
Sensors, Actuators and Tags
Lowest Power and Lowest Cost
Short
Range
Long

Figure 4.10 LoRaWAN Technology

Why LoRaWAN ?

- Ultra-low power - LoRaWAN end devices are optimized to operate in low power mode and can last up to 10 years on a single coin cell battery.
- Long range - LoRaWAN gateways can transmit and receive signals over a distance of over 10 kilometers in rural areas and up to 3 kilometers in dense urban areas.
- Deep indoor penetration - LoRaWAN networks can provide deep indoor coverage, and easily cover multi floor buildings.
- License free spectrum - You don’t have to pay expensive frequency spectrum license fees to deploy a LoRaWAN network.
- Geolocation- A LoRaWAN network can determine the location of end devices using triangulation without the need for GPS. A LoRa end device can be located if at least three gateways pick up its signal.
- High capacity - LoRaWAN Network Servers handle millions of messages from thousands of gateways.
- Public and private deployments - It is easy to deploy public and private LoRaWAN networks using the same hardware (gateways, end devices, antennas) and software (UDP packet forwarders, Basic Station software, LoRaWAN stacks for end devices).

- End-to-end security- LoRaWAN ensures secure communication between the end device and the application server using AES-128 encryption.
- Firmware updates over the air - You can remotely update firmware (applications and the LoRaWAN stack) for a single end device or group of end devices.
- Roaming- LoRaWAN end devices can perform seamless handovers from one network to another.
- Low cost - Minimal infrastructure, low-cost end nodes and open-source software.
- Certification program- The LoRa Alliance certification program certifies end devices and provides end-users with confidence that the devices are reliable and compliant with the LoRaWAN specification.
- Ecosystem- LoRaWAN has a very large ecosystem of device makers, gateway makers, antenna makers, network service providers, and application developers.

Gateway Specifications Table:

OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	54 (of which 15 provide PWM output)
ANALOG INPUT PINS	16
DC CURRENT PER I/O PIN	20 mA
DC CURRENT FOR 3.3V PIN	50 mA
FLASH MEMORY	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
CLOCK SPEED	16 MHz

Table 4.2 Gateway Specifications

c) ChirpStack based Server

The ChirpStack open-source LoRaWAN Network Server stack provides open-source components for LoRaWAN networks. Together they form a ready-to-use solution including a user-friendly web-interface for device management and APIs for integration. The modular architecture makes it possible to integrate within existing infrastructures. All components are licensed under the MIT license and can be used for commercial purposes.

The following components are provided:

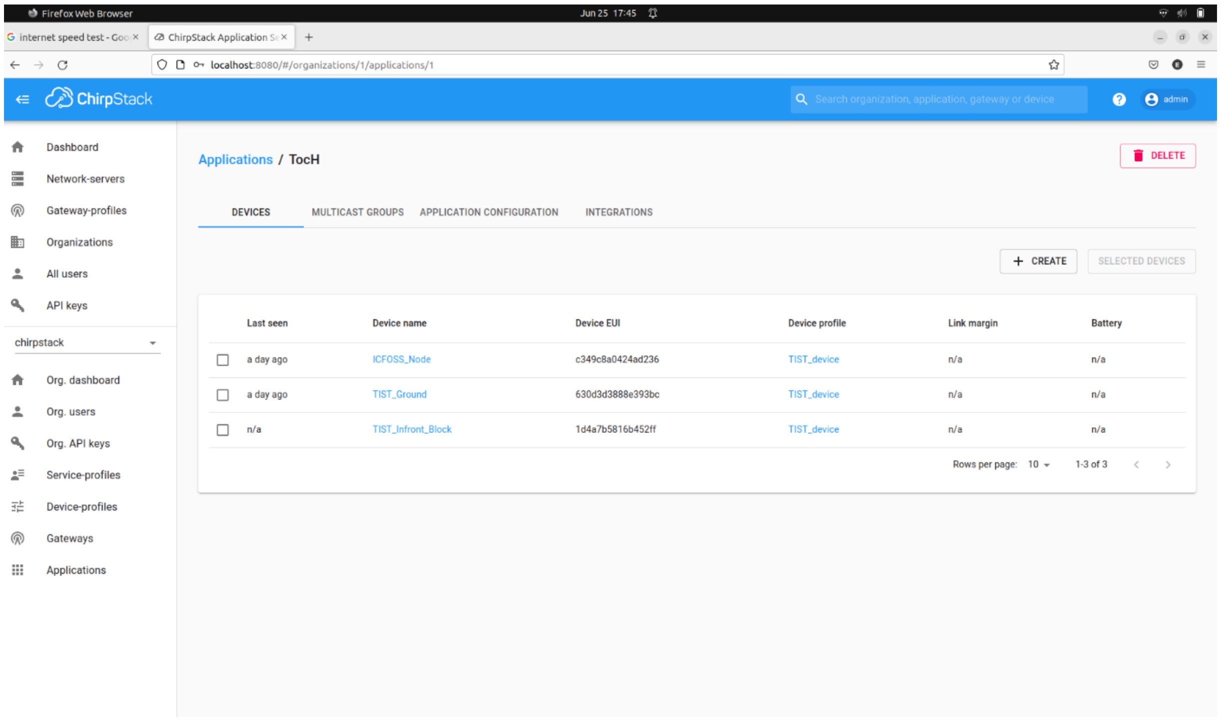
i. ChirpStack Gateway Bridge is a service which converts LoRa® Packet Forwarder protocols into a ChirpStack Network Server common data-format (JSON and Protobuf). This component is part of the ChirpStack open-source LoRaWAN Network Server stack. It handles the communication with LoRaWAN gateways.

ii. ChirpStack Network Server is an open-source LoRaWAN® Network Server implementation. This component is part of the ChirpStack stack. The responsibility of the Network Server component is the de-duplication of received LoRaWAN frames by the LoRa® gateways and for the collected frames handle the:

- Authentication
- LoRaWAN mac-layer (and mac-commands).
- Communication with the ChirpStack Application Server.
- Scheduling of downlink frames.

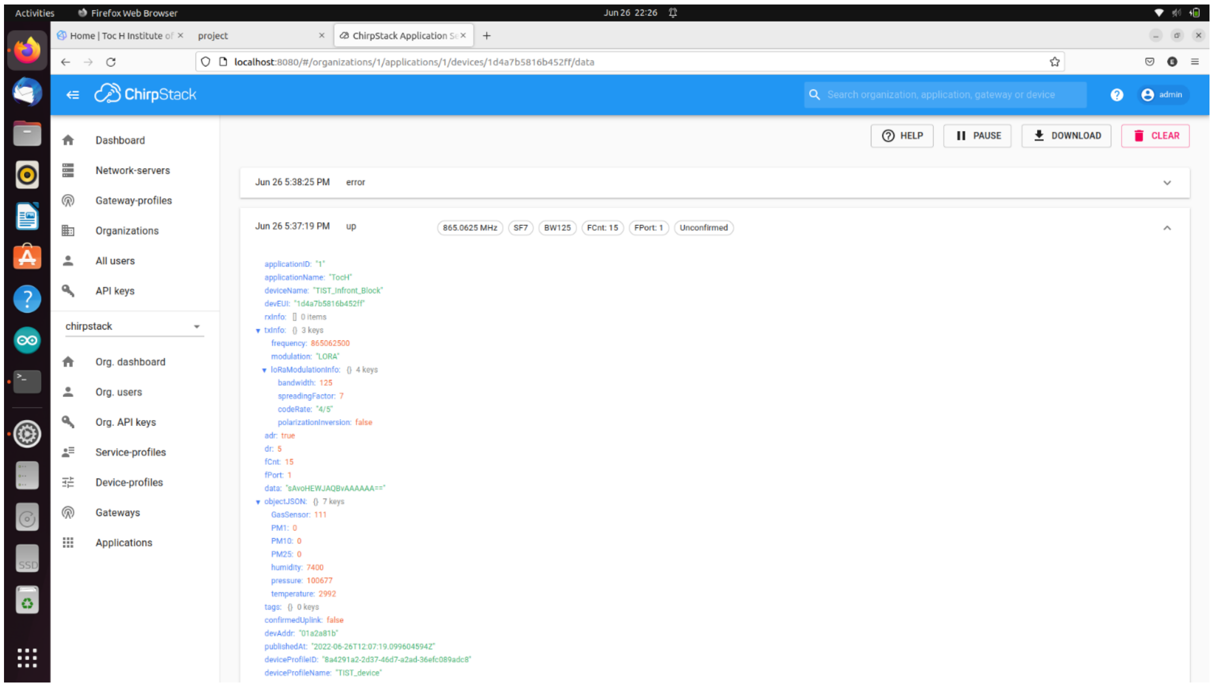
iii. ChirpStack Application Server is an open-source LoRaWAN® Application Server, part of the ChirpStack open-source LoRaWAN Network Server stack. It is responsible for the device "inventory" part of a LoRaWAN infrastructure, handling of join-request and the handling and encryption of application payloads. It offers a web-interface where users, organizations, applications, and devices can be managed. For integration with external services, it offers a grpc and restful API. Device data can be sent and / or received over MQTT, HTTP and be written directly into InfluxDB.

iv. ChirpStack Gateway OS is an open-source Linux based embedded OS which can run on various LoRa® gateway models. The goal is to make it easy to get started with LoRaWAN® and the ChirpStack open-source LoRaWAN Network Server stack with the minimum steps required to setup your gateway(s).



Last seen	Device name	Device EUI	Device profile	Link margin	Battery
a day ago	ICFOSS_Node	c349c8a0424ad236	TIST_device	n/a	n/a
a day ago	TIST_Ground	630d3d3888e393bc	TIST_device	n/a	n/a
n/a	TIST_Infront_Block	1d4a7b5816b452ff	TIST_device	n/a	n/a

Figure 4.11 ChirpStack Server



```
Jun 26 5:38:25 PM    error

Jun 26 5:37:19 PM    up    865.0625 MHz   SF7   BW125   FCnt: 15   FPort: 1   Unconfirmed
```

```
  applicationID: "1"
  applicationName: "TocH"
  deviceName: "TIST_Infront_Block"
  devEUI: "1d4a7b5816b452ff"
  rxInfo: [] 0 items
▼ txInfo: {} 3 keys
    frequency: 865062500
    modulation: "LORA"
  ▼ loRaModulationInfo: {} 4 keys
      bandwidth: 125
      spreadingFactor: 7
      codeRate: "4/5"
      polarizationInversion: false
  adr: true
  dr: 5
  fCnt: 15
  fPort: 1
  data: "sAvoHEWJAQBvAAAAAA=="
▼ objectJSON: {} 7 keys
    GasSensor: 111
    PM1: 0
    PM10: 0
    PM25: 0
    humidity: 7400
    pressure: 100677
    temperature: 2992
  tags: {} 0 keys
  confirmedUplink: false
  devAddr: "01a2a81b"
  publishedAt: "2022-06-26T12:07:19.099604594Z"
  deviceProfileID: "8a4291a2-2d37-46d7-a2ad-36efc089adc8"
  deviceProfileName: "TIST_device"
```

Figure 4.12 Service Uplink

4.2.3 VISUALIZATION

a) Dashboard

The dashboard was built using tools in influxDB database. The InfluxDB user interface (UI) provides tools for building custom dashboards to visualize your data. Dashboard variables allow you to alter specific components of cells' queries without having to edit the queries, making it easy to interact with your dashboard cells and explore your data. The InfluxDB UI provides multiple visualization types to visualize your data in a format that makes the most sense for your use case. We can use the available customization options to customize each visualization. Add annotations to our InfluxDB dashboards to provide useful, contextual information about single points in time. We can add labels if we want to. Labels are a way to add visual metadata to dashboards, tasks, and other items in the InfluxDB UI. It helps in viewing and managing labels in the InfluxDB user interface.

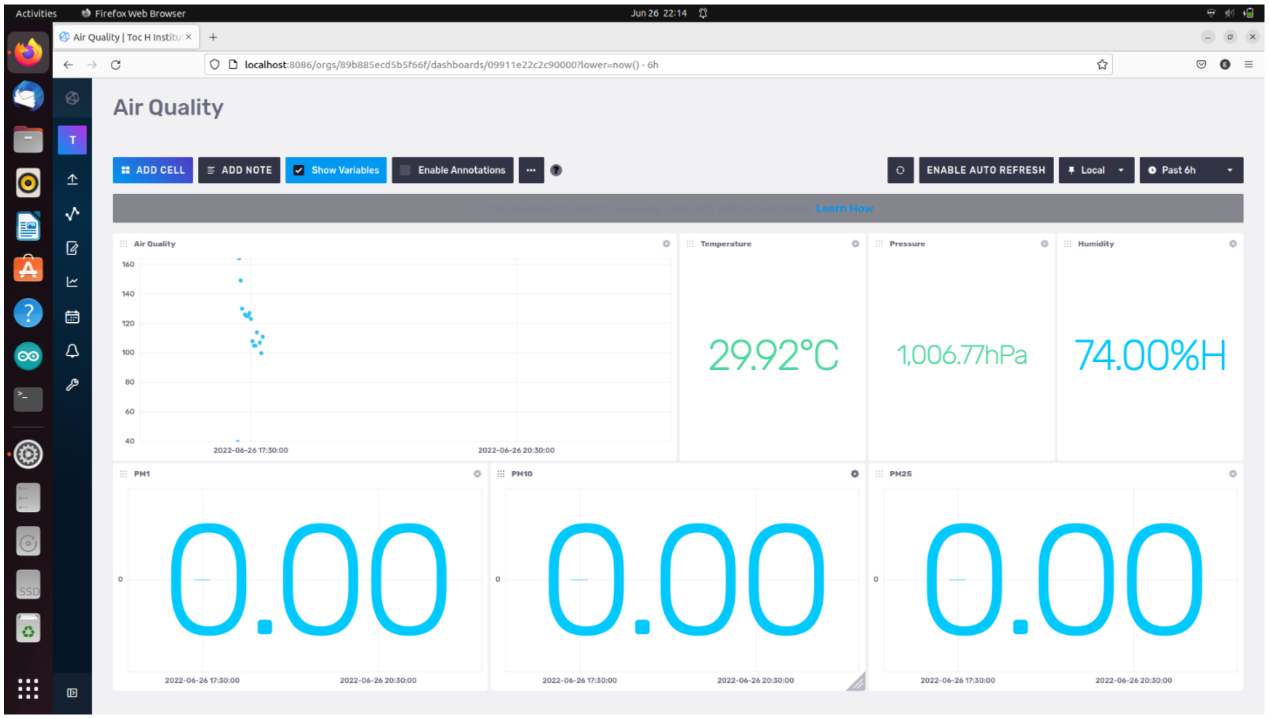


Figure 4.13 – Dashboard

b) Map

Another visualization tool provided is a map which consists of marking on positions where a node is placed. The Flask framework is used to deploy the Map. Flask is a micro web framework written in Python. It is classified as a microframework because it does not require tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions

Pros and cons of Flask and Django

1) Simplicity of development

2) Library support

Advantages of Flask-based systems

- higher flexibility
- higher compatibility with latest technologies
- high scalability for simple web applications
- technical experimentation
- customization
- slightly higher framework performance
- easier to use for simple cases
- smaller size of the code base

Disadvantages of Flask-based systems

- more potential for security risks
- slower MVP development in most cases
- more complex tech stack
- higher maintenance costs for more complex systems
- more complicated maintenance for larger implementation

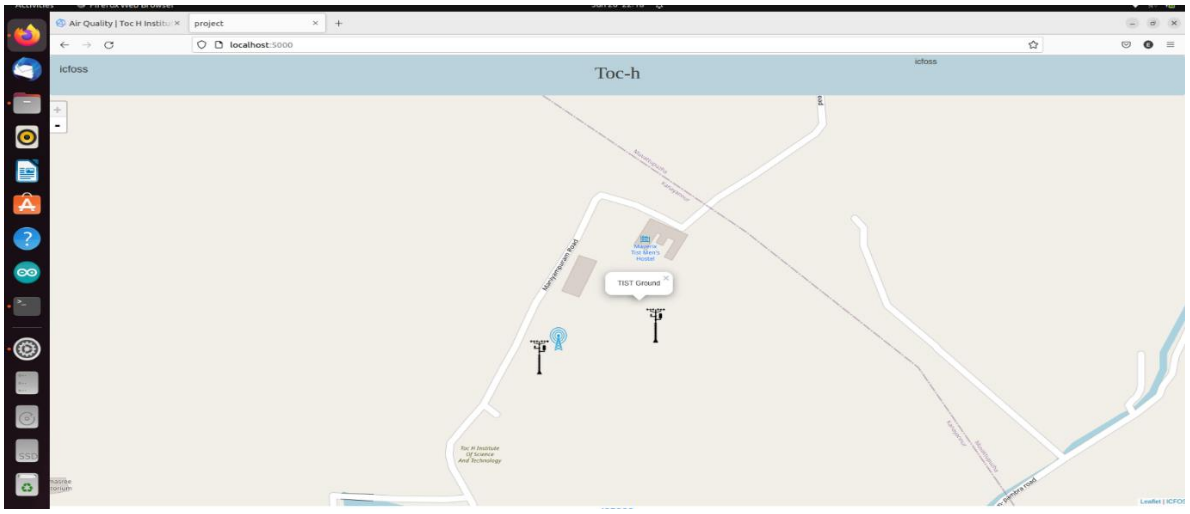


Figure 4.14 – Map

4.3 DATAFLOW DIAGRAM

4.3.1 FLOWCHART OF OVERALL OPERATION

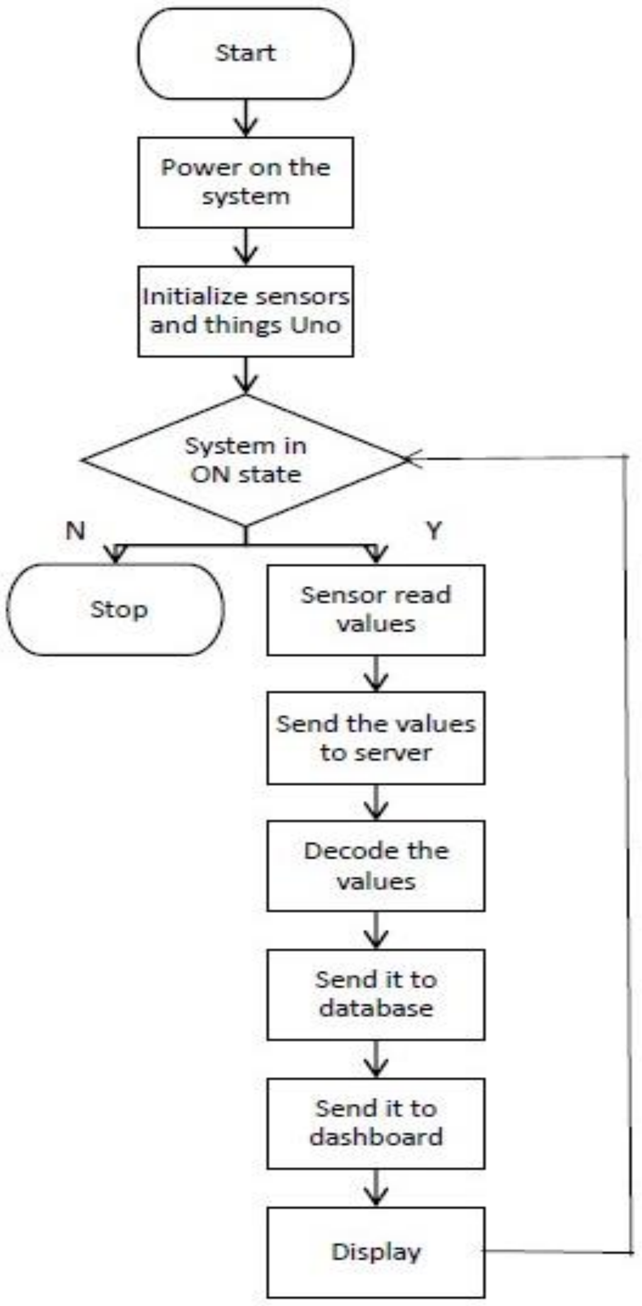


Figure 4.15 Flowchart

4.4 DATABASE DESIGN

The database was built on top of InfluxDB. It is a multi-tenanted time series database, UI and dashboarding tools, background processing and monitoring agent. It structures our queries and separate common logic into functions and libraries that are easily shared and help speed development. We can also enrich our time series data with other SQL data stores or with cloud-based data stores. InfluxDB has no external dependencies and provides an SQL-like language, listening on port 8086.

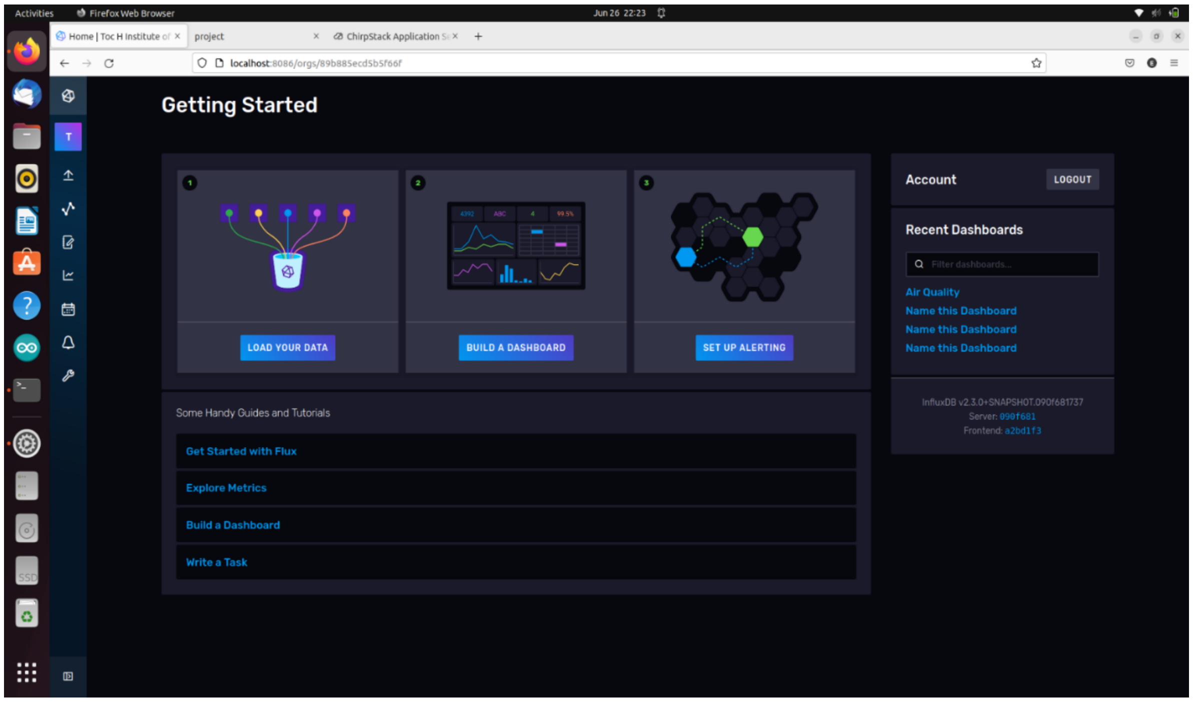


Figure 4.16 Database

Advantages of influxDB:

- InfluxDB has a set of APIs and tools to help you get started quickly with more capability and less code.
- Run and grow large data workloads at high volumes globally.
- Series cardinality and high throughput to continuously ingest and transform hundreds of millions of time series per second.

- Batch and streaming to ingest and join data from millions of sources in both batch and stream.
- Flexible storage to manage retention for high fidelity and down sampled data.
- A highly available service that runs on 3 clouds, 4 continents, 20 regions.

4.5 ALGORITHMS

4.5.1 ALGORITHM OF OVERALL OPERATION OF THE SYSTEM

Step 1: Start

Step 2: Power up the system, i.e., the nodes and gateway.

Step 3: Initialize the sensors and Arduino Mega.

Step 4: If system is powered ON repeat the following steps:

Step 4.1: Sensors read the values and send the output to Arduino Mega microcontroller.

Step 4.2: Send these values to the server via gateway.

Step 4.3: Decode the values.

Step 4.4: Send the values to the database and store them.

Step 4.5: Send the values to the dashboard and display the current value, minimum value, and maximum value of AQI and PM values and plot graphs for them too.

Step 5: Stop

IMPLEMENTATION AND PRESENTATION OF DATA

5. IMPLEMENTATION AND PRESENTATION OF DATA

5.1. SYSTEM REQUIREMENTS

- LoRa WAN based indoor gateway
- Arduino Mega 2560
- BME280
- PMS7003
- MQ135
- InfluxDB
- Arduino IDE

5.2. IMPLEMENTATION DETAILS

The entire coding for all the components was done on the Arduino IDE platform. Each sensor was separately programmed with the microcontroller and its output was verified. The Arduino Mega was connected to the USB port of the laptop to run and upload the program. Once we assured that all the components are working individually, we then connected them together and wrote the program for the entire Arduino Mega in which each individual program of the sensors we converted into separate functions and combined under a single program as modules. This program was compiled, executed, and uploaded into the Arduino Mega.

The Node sensed, measured and sent the values to the gateway, built on LoRa Wan technology. From there the data was sent to the server, which was implemented using ChirpStack. The ChirpStack open-source LoRaWAN Network Server stack provides open-source components for LoRaWAN networks. Together they form a ready-to-use solution including a user-friendly web-interface for device management and APIs for integration. The modular architecture makes it possible to integrate within existing infrastructures. All components are licensed under the MIT license and can be used for commercial purposes. The data sent by the gateway will be in a decoded form, so the necessary programs were written to decode the data values. After decoding these values were send to the database created using influxDB. From the database the dashboard accesses the data.

The dashboard was implemented using InfluxDB tools. In order to create panels, we had inbuilt options. After creating the panels, each panel was customized according to the needs using SQL queries. Based on the values of AQI, PM1, PM2.5 and PM10, four different graphs were plotted with respect to time. Also the maximum, minimum and current values of PMs were displayed along with their graphs. Further the dashboard was again updated. Currently it can also show the time at which the AQI was maximum within a specified time range.

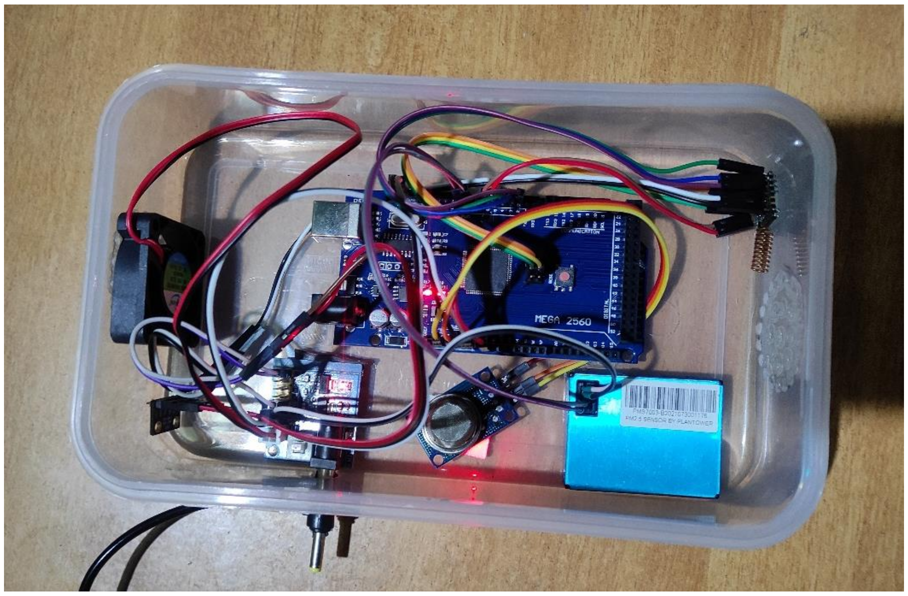


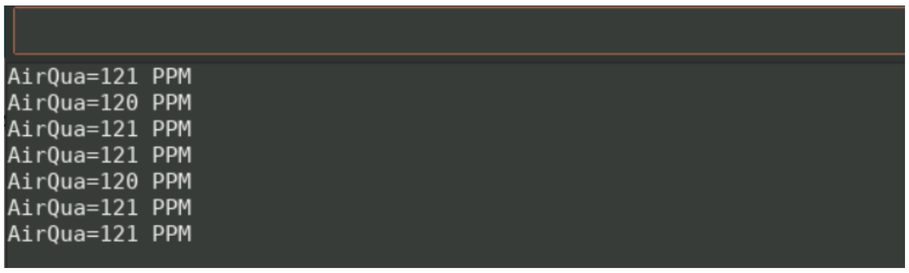
Figure 5.1 – Node implemented

5.3. TESTING

5.3.1 UNIT TESTING

In unit testing module will be tested against a basic list of tests. This verifies that each module satisfies the minimum requirements for it to be accepted into the project. Firstly, the Arduino Mega was checked by plugging it into the computer via USB port. The red LED on Arduino Mega started glowing showing that the microcontroller was working. All the sensors were connected to Arduino Mega separately and each of their outputs were analyzed.

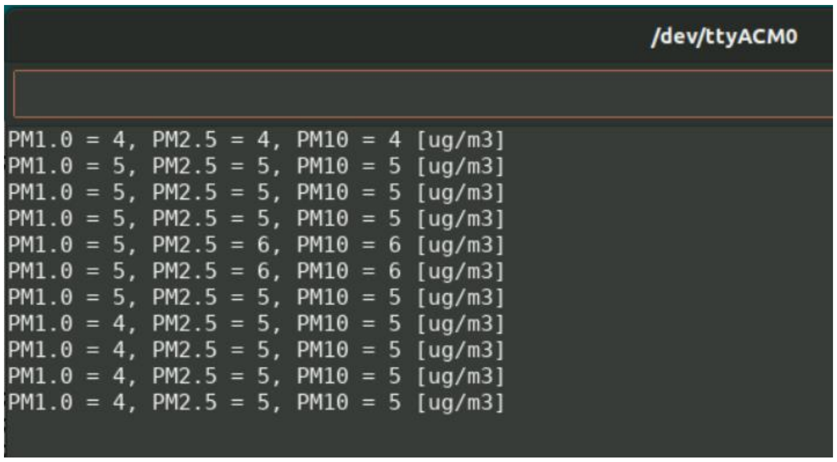
a) MQ135



```
AirQua=121 PPM
AirQua=120 PPM
AirQua=121 PPM
AirQua=121 PPM
AirQua=120 PPM
AirQua=121 PPM
AirQua=121 PPM
```

Figure 5.2 MQ135 Output

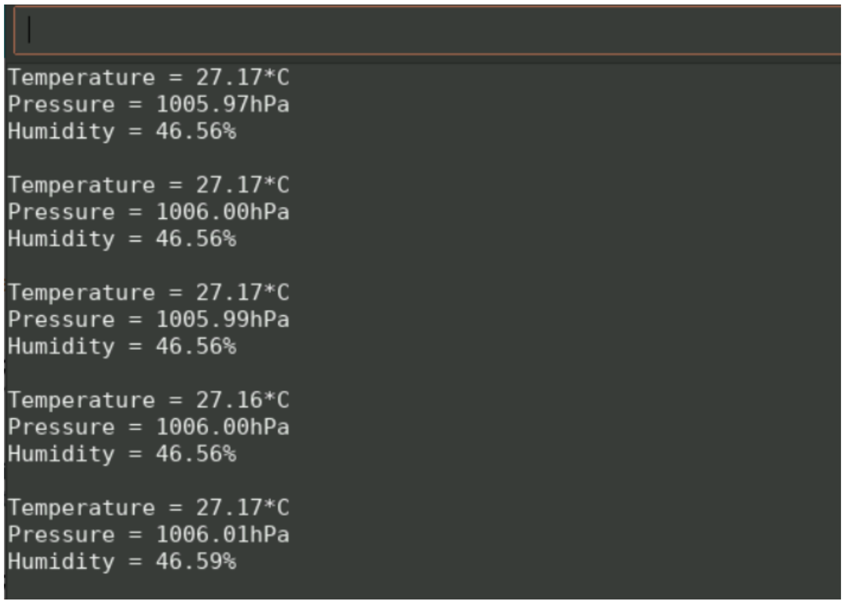
b) PMS7003



```
/dev/ttyACM0

PM1.0 = 4, PM2.5 = 4, PM10 = 4 [ug/m3]
PM1.0 = 5, PM2.5 = 5, PM10 = 5 [ug/m3]
PM1.0 = 5, PM2.5 = 5, PM10 = 5 [ug/m3]
PM1.0 = 5, PM2.5 = 5, PM10 = 5 [ug/m3]
PM1.0 = 5, PM2.5 = 6, PM10 = 6 [ug/m3]
PM1.0 = 5, PM2.5 = 6, PM10 = 6 [ug/m3]
PM1.0 = 5, PM2.5 = 5, PM10 = 5 [ug/m3]
PM1.0 = 4, PM2.5 = 5, PM10 = 5 [ug/m3]
PM1.0 = 4, PM2.5 = 5, PM10 = 5 [ug/m3]
PM1.0 = 4, PM2.5 = 5, PM10 = 5 [ug/m3]
PM1.0 = 4, PM2.5 = 5, PM10 = 5 [ug/m3]
```

Figure 5.3 PMS7003 Output

c) BME280



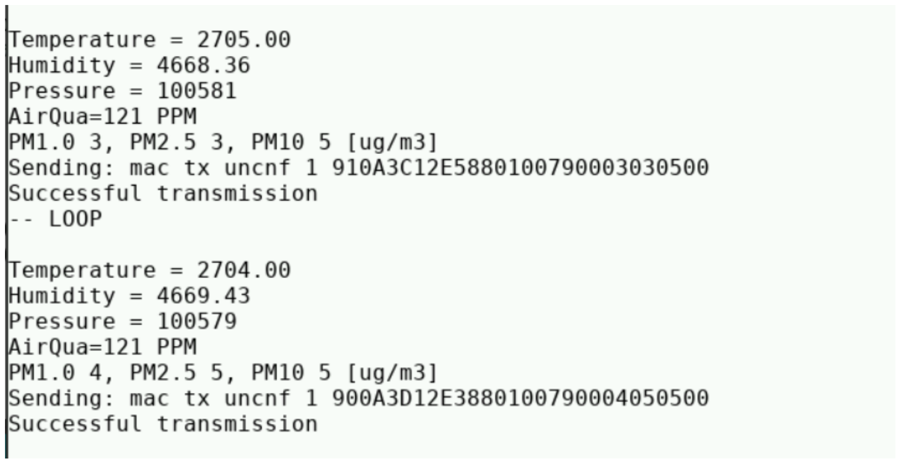
```
Temperature = 27.17*C
Pressure = 1005.97hPa
Humidity = 46.56%

Temperature = 27.17*C
Pressure = 1006.00hPa
Humidity = 46.56%

Temperature = 27.17*C
Pressure = 1005.99hPa
Humidity = 46.56%

Temperature = 27.16*C
Pressure = 1006.00hPa
Humidity = 46.56%

Temperature = 27.17*C
Pressure = 1006.01hPa
Humidity = 46.59%
```

Figure 5.4 BME280 Output

5.3.2 INTEGRATION TESTING

Integration testing is the type of software testing in which the different units, modules or components of a software application are tested as a combined entity. Here all the three sensors are combined together in the Arduino Mega board and their combined output is generated.

Combined Output on Arduino IDE:



```
Temperature = 2705.00
Humidity = 4668.36
Pressure = 100581
AirQua=121 PPM
PM1.0 3, PM2.5 3, PM10 5 [ug/m3]
Sending: mac tx uncnf 1 910A3C12E5880100790003030500
Successful transmission
-- LOOP

Temperature = 2704.00
Humidity = 4669.43
Pressure = 100579
AirQua=121 PPM
PM1.0 4, PM2.5 5, PM10 5 [ug/m3]
Sending: mac tx uncnf 1 900A3D12E3880100790004050500
Successful transmission
```

Figure 5.5 Integration testing

5.3.3 SYSTEM TESTING

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested. Here the values received on the InfluxDB dashboard are tested against the value generated in integration testing and also the value with other existing systems.

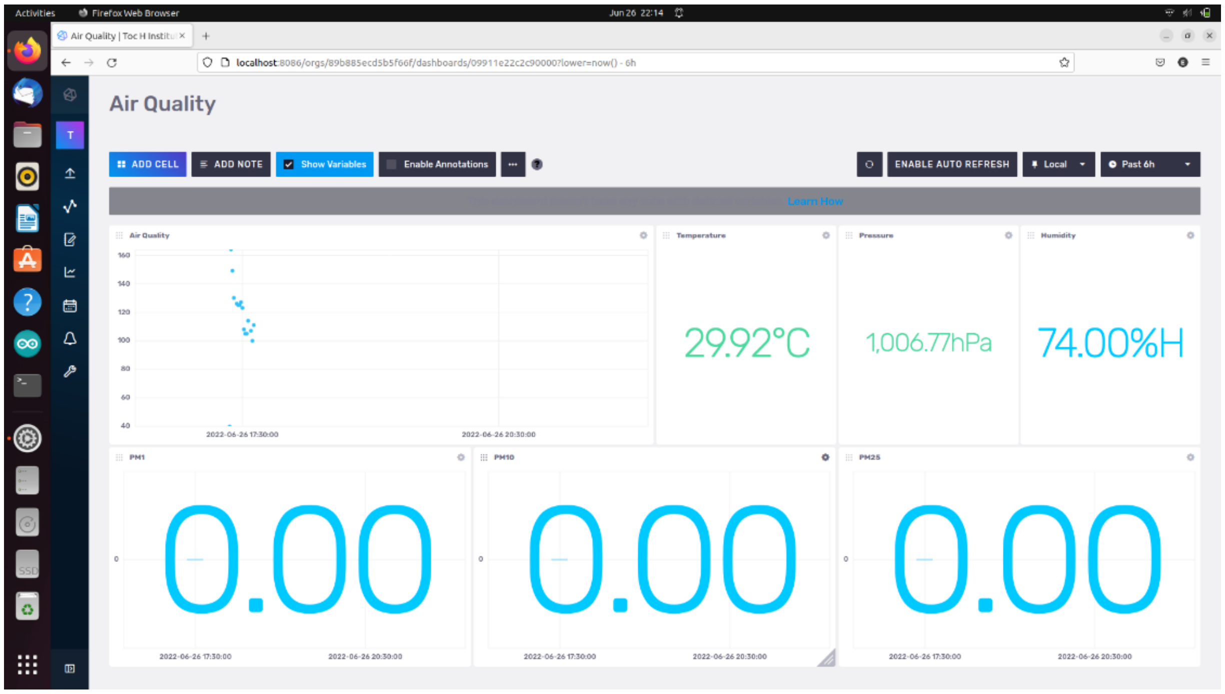


Figure 5.6 System testing

5.4. EXPERIMENTAL RESULTS & DISCUSSION

The system was successfully executed. The sensors sensed and displayed the value which was like the values compared to the existing systems. The values shown in the dashboard were same as the ones displayed in the server and terminal of Arduino IDE. That implies, there was no disturbance of any kind in the transfer of data. The AQI and PM values were plotted with respect to time and the maximum, minimum and current AQI and PM values were displayed beneath the graph. Values of the non-functional units like temperature, pressure and humidity were also displayed on the dashboard precisely. Even a minute change in values over time were monitored.

The dashboard was designed in such a way that the values will be monitored in every 5 seconds. Graph will be plotted based on this concept. Also, we can set the time range of monitoring between 5 minutes to 7 days. The time at which maximum AQI value was shown in a particular time range will be displayed along with the AQI value separately.

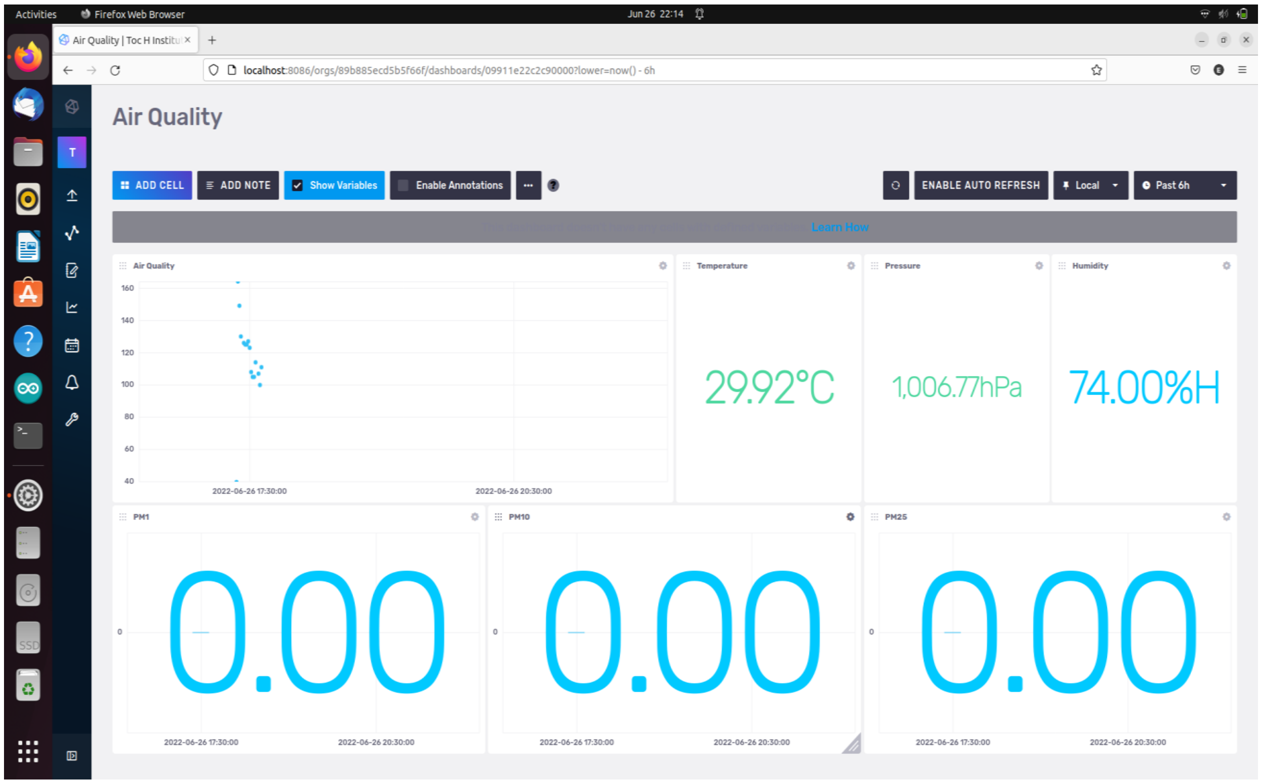


Figure 5.7 - Dashboard implemented

CONCLUSION

6. CONCLUSION

Air quality monitoring is challenging to enact as it requires the effective integration of multiple environmental data sources, which often originate from different environmental networks and institutions. Air quality monitoring system is basically used to investigate air quality and the effects of air pollution. Interpretation of ambient air monitoring data often involves a consideration of the spatial and temporal representativeness[9] of the data gathered, and the health effects associated with exposure to the monitored levels. The system monitors and calculate the level of air pollution and plot graphs to compare it with previous day’s data. Since the system is low in cost and easy to operate it can be set up in almost all places like education institutions, hotels, factories, hospitals, and malls.

FUTURE SCOPE

7. FUTURE SCOPE

The architecture proposed in the system will help in analysing the air quality for the particular surrounding in the real time and it will be helpful for colleges, schools and other buildings to get the accurate data of air quality in their surroundings. The values of particulate matters like PM1, PM2.5 and PM10 will be plotted against date on dashboard which makes it more clear to understand the exact nature of the air around. Especially in India which is ranked high in the list with maximum air pollution this system will be truly beneficial. AQI and PM values on weekdays and weekends in places like educational institutions can be compared which might show a decrease in values on weekends due to the isolation of the campus.

Further if we are using an outdoor gateway which covers around 10 kms in diameter, we can implement a map of an area within that range on the dashboard, which shows the location where each node is placed via a marker on that spot. When the cursor is moved onto that location the values on that location can be viewed. Also, the professionals like air quality analysts, climatologists etc., can use the dashboard in their analysis and assessments.

REFERENCES

8. REFERENCES

1. R. K. Jha, "Air Quality Sensing and Reporting System Using IoT," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020, pp. 790-793, doi: 10.1109/ICIRCA48905.2020.9182796.
2. N. Mishra, N. Gupta and A. Rana, "Air Quality Monitoring and IoT- Past and Future," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2020, pp. 703-706, doi: 10.1109/ICRITO48877.2020.9197927.
3. S. Faiazuddin, M. V. Lakshmaiah, K. T. Alam and M. Ravikiran, "IoT based Indoor Air Quality Monitoring system using Raspberry Pi4," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2020, pp. 714-719, doi: 10.1109/ICECA49313.2020.9297442.
4. D. Zhang and S. S. Woo, "Real Time Localized Air Quality Monitoring and Prediction Through Mobile and Fixed IoT Sensing Network," in IEEE Access, vol. 8, pp. 89584-89594, 2020, doi: 10.1109/ACCESS.2020.2993547.
5. V. Choudhary, J. H. Teh, V. Beltran and H. B. Lim, "AirQ: A Smart IoT Platform for Air Quality Monitoring," 2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC), 2020, pp. 1-2, doi: 10.1109/CCNC46108.2020.9045550.
6. H. Cho and Y. Baek, "Design and Implementation of a Smart Air Quality Monitoring and Purifying System for the School Environment," 2022 IEEE International Conference on Consumer Electronics (ICCE), 2022, pp. 1-4, doi: 10.1109/ICCE53296.2022.9730505.
7. S. Esfahani, P. Rollins, J. P. Specht, M. Cole and J. W. Gardner, "Smart City Battery Operated IoT Based Indoor Air Quality Monitoring System," 2020 IEEE SENSORS, 2020, pp. 1-4, doi: 10.1109/SENSORS47125.2020.9278913.

APPENDIX

APPENDIX I – CODE SNIPPETS

A. MAIN PROGRAM

Description : This is the code snippet for the air quality checking node.

```
#include <Adafruit_Sensor.h>
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include <Adafruit_BME280.h>
#include <PMserial.h>

#define DEBUG
// LoRaWAN NwkSKey, network session key
// This is the default Semtech key, which is used by the
early prototype TTN
// network.
static const PROGMEM u1_t NWKSKEY[16] = {0x49, 0x3C, 0xEB,
0x39, 0x81, 0x91, 0xF4, 0xDF, 0x3B, 0x1C, 0x06, 0xC8, 0xC9,
0x5D, 0x96, 0x60};

// LoRaWAN AppSKey, application session key
// This is the default Semtech key, which is used by the
early prototype TTN
// network.
static const u1_t PROGMEM APPSKEY[16] = {0x0E, 0x0D, 0x7A,
0xD8, 0x48, 0x6B, 0xA3, 0x4A, 0x8C, 0x76, 0x4A, 0xAA, 0x84,
0x75, 0xE9, 0x0F};

// LoRaWAN end-device address (DevAddr)
static const u4_t DEVADDR = 0x01a2a81b ; // <-- Change this
address for every node!

// These callbacks are only used in over-the-air
activation, so they are
// left empty here (we cannot leave them out completely
unless
// DISABLE_JOIN is set in config.h, otherwise the linker
will complain).
```

```
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }
struct data {
  short int atmTemp;
  short int atmHum;
  uint32_t atmPressure;
  short int gasSensor;
  char PM1;
  char PM25;
  char PM10;
  char err;
}mydata;

Adafruit_BME280 bme;
int sensorValue;
SerialPM pms(PMSx003, 10, 11);  // PMSx003, RX, TX
static osjob_t sendjob;
bool next = false;
// Schedule TX every this many seconds (might become longer
due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 60;

// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 6,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 5,
    .dio = {2, 3, 4},
};

void onEvent (ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            Serial.println(F("EV_SCAN_TIMEOUT"));
            break;
        case EV_BEACON_FOUND:
            Serial.println(F("EV_BEACON_FOUND"));
            break;
        case EV_BEACON_MISSED:
            Serial.println(F("EV_BEACON_MISSED"));
            break;
        case EV_BEACON_TRACKED:
            Serial.println(F("EV_BEACON_TRACKED"));
            break;
        case EV_JOINING:
            Serial.println(F("EV_JOINING"));
            break;
        case EV_JOINED:
            Serial.println(F("EV_JOINED"));
            break;
        case EV_RFU1:
            Serial.println(F("EV_RFU1"));
            break;
        case EV_JOIN_FAILED:
            Serial.println(F("EV_JOIN_FAILED"));
            break;
        case EV_REJOIN_FAILED:
            Serial.println(F("EV_REJOIN_FAILED"));
            break;
        case EV_TXCOMPLETE:
            Serial.println(F("EV_TXCOMPLETE (includes
waiting for RX windows)"));
            if (LMIC.txrxFlags & TXRX_ACK)
              Serial.println(F("Received ack"));
            if (LMIC.dataLen) {
              Serial.println(F("Received "));
              Serial.println(LMIC.dataLen);
              Serial.println(F(" bytes of payload"));
            }
            // Schedule next transmission
            os_setTimedCallback(&sendjob,
os_getTime()+sec2osticks(TX_INTERVAL), do_send);
            break;
        case EV_LOST_TSYNC:
            Serial.println(F("EV_LOST_TSYNC"));
            break;
        case EV_RESET:
            Serial.println(F("EV_RESET"));
            break;
        case EV_RXCOMPLETE:
            // data received in ping slot
            Serial.println(F("EV_RXCOMPLETE"));
            break;
        case EV_LINK_DEAD:
            Serial.println(F("EV_LINK_DEAD"));
            break;
        case EV_LINK_ALIVE:
            Serial.println(F("EV_LINK_ALIVE"));
            break;
         default:
            Serial.println(F("Unknown event"));
            break;
    }
}

void do_send(osjob_t* j)
{
    readBME280();
    readMQ135();
    readPMS7003();

    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Prepare upstream data transmission at the next
possible time.
        LMIC_setTxData2(1, (unsigned char *)&mydata,
sizeof(mydata) - 1, 0);
        Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}

void setup() {
    Serial.begin(115200);
    pms.init();
    Serial.println(F("Starting"));

    #ifdef VCC_ENABLE
    // For Pinoccio Scout boards
    pinMode(VCC_ENABLE, OUTPUT);
    digitalWrite(VCC_ENABLE, HIGH);
    delay(1000);
    #endif

    // LMIC init
    os_init();
    // Reset the MAC state. Session and pending data
transfers will be discarded.
    LMIC_reset();

    // Set static session parameters. Instead of
dynamically establishing a session
    // by joining the network, precomputed session
parameters are be provided.
    #ifdef PROGMEM
    // On AVR, these values are stored in flash and only
copied to RAM
    // once. Copy them to a temporary buffer here,
LMIC_setSession will
    // copy them into a buffer of its own again.
    uint8_t appskey[sizeof(APPSKEY)];
    uint8_t nwkskey[sizeof(NWKSKEY)];
    memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
    memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
    LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
    #else
    // If not running an AVR with PROGMEM, just use the
arrays directly
    LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
    #endif

    #if defined(CFG_eu868)
    // Set up the channels used by the Things Network,
which corresponds
```

```
    // to the defaults of most gateways. Without this, only
three base
    // channels from the LoRaWAN specification are used,
which certainly
    // works, so it is good for debugging, but can overload
those
    // frequencies, so be sure to configure the full
frequency range of
    // your network here (unless your network
autoconfigures them).
    // Setting up channels should happen after
LMIC_setSession, as that
    // configures the minimal channel set.
    // NA-US channels 0-71 are configured automatically
          LMIC_setupChannel(0, 865062500,
DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);      // g-band
          LMIC_setupChannel(1, 865402500,
DR_RANGE_MAP(DR_SF12, DR_SF7B), BAND_CENTI);      // g-band
          LMIC_setupChannel(2, 865985000,
DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);      // g-band
    // TTN defines an additional channel at 869.525Mhz
using SF9 for class B
    // devices' ping slots. LMIC does not have an easy way
to define set this
    // frequency and support for class B is spotty and
untested, so this
    // frequency is not configured here.
    #elif defined(CFG_us915)
    // NA-US channels 0-71 are configured automatically
    // but only one group of 8 should (a subband) should be
active
    // TTN recommends the second sub band, 1 in a zero
based count.
    // https://github.com/TheThingsNetwork/gateway-
conf/blob/master/US-global_conf.json
    LMIC_selectSubBand(1);
    #endif

    // Disable link check validation
    LMIC_setLinkCheckMode(0);
    // TTN uses SF9 for its RX2 window.
    LMIC.dn2Dr = DR_SF9;

    // Set data rate and transmit power for uplink (note:
txpow seems to be ignored by the library)
    LMIC_setDrTxpow(DR_SF7,14);

    // Start job
    do_send(&sendjob);
}



void loop() {
    extern volatile unsigned long timer0_overflow_count;

    if (next == false)
    {
        os_runloop_once();
    }
    else
    {

        #ifdef DEBUG
        Serial.print(F("Enter sleeping forever "));
        Serial.flush(); // give the serial print chance to
complete
        #endif
//          LowPower.powerDown(SLEEP_FOREVER, ADC_OFF,
BOD_OFF);
        cli();
        timer0_overflow_count += 8 * 64 *
clockCyclesPerMicrosecond();
        sei();

        #ifdef DEBUG
        Serial.println(F("Sleep complete"));
        #endif
    }
```

```
}




void readBME280()
{

  bme.begin(0x76);
///
 uint32_t Pres = (bme.readPressure() / 100.0F)*100;
 float temperature=bme.readTemperature()*100;
 float Humidity=bme.readHumidity()*100;

  delay(1000);
  mydata.atmTemp=temperature;
  mydata.atmHum=Humidity;
  mydata.atmPressure = Pres;

//  payload[1] = temperature;
//  payload[2] = Humidity;
  Serial.println(mydata.atmTemp);
  Serial.println(mydata.atmHum);
  Serial.println(mydata.atmPressure);
 //Serial.println(payload[3]);
}


void readMQ135()
{
    int sensorValue;
    sensorValue = analogRead(0);       // read analog input
pin 0
    Serial.print("AirQua=");


    mydata.gasSensor=sensorValue;
    Serial.println(mydata.gasSensor);
    delay(1000);                                    // wait
100ms for next reading
```

```
}


void readPMS7003()
{

 pms.read();                   // read the PM sensor
  Serial.print(F("PM1.0
"));Serial.println(pms.pm01);//Serial.print(F(", "));
  Serial.print(F("PM2.5
"));Serial.println(pms.pm25);//Serial.print(F(", "));
  Serial.print(F("PM10 "))
;Serial.println(pms.pm10);//Serial.println(F(" [ug/m3]"));
  mydata.PM1=pms.pm01;
  mydata.PM25=pms.pm25;
  mydata.PM10=pms.pm10;
  delay(1000);
}
```

B. INFLUXDB QUERIES

Description : These are the queries written in InfluxDB dashboard to make panels and customize it.

Query For MQ135

Description : Receives the Air Quality Index value and plot a graph on these values with respect to time and displays the current, maximum and minimum AQI values.

```
SELECT "value" FROM
"device_frmpayload_data_Air_Quality_Index" WHERE
("device_name" = 'TIST_Infront_of_Block') AND time >= now() -
5m
```

Query For BME280 - TEMPERATURE

Description : Receives the Temperature and display it on the dashboard.

```
SELECT "value"  / 100 FROM
"device_frmpayload_data_Temperature" WHERE ("device_name" =
'TIST_Infront_of_Block') AND time >= now() - 5m
```

Query For BME280 - HUMIDITY

Description : Receives the Humidity value and display it on the dashboard.

```
SELECT "value"  / 100 FROM "device_frmpayload_data_Humidity"
WHERE ("device_name" = 'TIST_Infront_of_Block') AND time >=
now() - 5m
```

Query For BME280 – ATMOSPHERIC PRESSURE

Description : Receives the Atmospheric Pressure and display it on the dashboard.

```
SELECT "value"  / 100 FROM "device_frmpayload_data_Pressure"
WHERE ("device_name" = 'TIST_Infront_of_Block') AND time >=
now() - 5m
```

Query For PMS7003 – PM1

Description : Receives the PM1 value and plot a graph on these values with respect to time and displays the current, maximum and minimum PM1 values.

```
SELECT "value" FROM "device_frmpayload_data_PM1" WHERE
("device_name" = 'TIST_Infront_of_Block') AND time >= now() -
5m
```

Query For PMS7003 – PM2.5

Description : Receives the PM2.5 value and plot a graph on these values with respect to time and displays the current, maximum and minimum PM2.5 values.

```
SELECT "value" FROM "device_frmpayload_data_PM25" WHERE
("device_name" = 'TIST_Infront_of_Block') AND time >= now() -
5m
```

Query For PMS7003 – PM10

Description : Receives the PM10 value and plot a graph on these values with respect to time and displays the current, maximum and minimum PM10 values.

```
SELECT "value" FROM "device_frmpayload_data_PM10" WHERE
("device_name" = 'TIST_Infront_of_Block') AND time >= now() -
5m
```

QUERY TO FIND MAXIMUM AQI VALUE

```
SELECT max("value") FROM
"device_frmpayload_data_Air_Quality_Index" WHERE
("device_name" = 'TIST_Infront_of_Block') AND time >= now() -
5m GROUP BY time(24h)
```