LIST OF FIGURES

Figure No.	Figure Name	Page No.
1.	Fundamental Component of IoT System	11
2.	IoT platform Architecture	13
3.	IOT Data Analytics	16
4.	MQTT Protocol for IoT	18
5.	CoAP Protocol for IoT	19
6.	Supervised Learning	33
7.	Unsupervised Learning	34
8.	Reinforcement Learning	35
9.	Machine Learning Models	36
10.	Global Warming	41
11.	Annual CO2 Emissions	43
12	Manual Air Monitoring through AAQMS	44
13.	Automated Air Monitoring through CAAQMS	45
14.	Arduino Uno	50
15.	ESP8266 WIFI Module	51
16.	16X2 LCD Display	53
17.	MQ135 Gas Sensor	53
18.	Sensitivity Characteristics of MQ135	54
19.	MQ2	55
20.	Wiring Diagram MQ2	56

21.	Sensitivity Characteristics of MQ2	57
22.	DHT11	58
23.	Calibrated Connection	58
24.	Arduino Ide	59
25.	ThingSpeak User Interface	60
26.	Jupyter Notebook User Interface	61
27.	Fritzing User Interface	62
28.	Calibration of MQ-135	63
29.	Datalogger codes	66
30.	VAR Model Codes	68
31.	Fritzing Circuit Diagram and Real Circuit Diagram	69
32.	ThingSpeak Reading	69
33.	Area Plot	70
34.	Correlation Heat Map	70
35.	Median Boxplot	70
36.	VAR Result	71
37.	Training the VAR Model of the selected order(p)	71
38.	Plot of Forecast vs Actuals	72
39.	Accuracy of the predicted values	72
Table No.	Table Name	Page No.
1.	IOT Sensor	15
2.	Pollution Criteria	40
3.	Indexes	71

ABBREVIATIONS

IOT	Internet of Things
PPM	Parts Per Million
IDE	Integrated Development Enviornment
TCP	Transmission Control Protocol
IP	Internet Protocol
LAN	Local Area Network
HTTP	Hypertext Transfer Protocol
DC	Direct Current
IC	Integrated Circuit
AC	Alternating Current
VAR	Vector Autoregression
CAAQMS	Continuous Ambient Air Quality Monitoring System
AAQMS	Ambient Air Quality Monitoring System
MAPE	Mean Absolute Percentage Error
ME	Mean Error
MAE	Mean Absolute Error
MPE	Mean Percentage Error
RMSE	Root Mean Square Error
CORR	Correlation
MINMAX	Minimum and Maximum
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
FPE	Factor Price Equalization
HQIC	Hannan-Quinn Information Criterion
ADF	Augmented Dickey Fuller Test
PCA	Principal Component Analysis
TSNE	t-Stochastic Neighbour Embedding
SVD	Singular Validation Detection
SRAM	Static Random Access Memory
EEPROM	Electrically Erasable Programmable Read Only Memory

TABLE OF CONTENT

Chapter No.	Chapter Name	Page no.
0.	i. Declaration certificate ii. Certificate of approval iii. Acknowledgement iv. Abstract v. List of Figures vi. Abbreviations	2 3 4 5 6-7 8
1.	Introduction	10-25
2.	Motivation	26-28
3.	Literature Review	29-31
4.	Machine Learning	32-38
5.	Air Monitoring System	39-47
6.	Hardware and Software	48-62
7.	Working Principle and Results	63-72
8.	Conclusions	73
9.	Future Work	74
10.	References	75-79

CHAPTER 1 : **INTRODUCTION**

1.1 Internet of Things

Internet of Things (IoT) is the networking of physical objects that contain electronics embedded within their architecture so as to speak and sense interactions amongst one another or with regard to the external environment. Within the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a really few of the specific examples where IoT is strongly established. Kevin Ashton is an innovator and consumer sensor expert who coined the phrase “the Internet of Things” to explain the network connecting objects within the physical world to the web [1].

IoT systems allow users to realize deeper automation, analysis, and integration within a system. They improve the reach of those areas and their accuracy. IoT used different technologies for sensing, networking, and robotics. IoT exploits recent advances in software moreover due to IoT, hardware price decreases. Its new and advanced elements bring major changes within the delivery of products, goods, and services; and therefore, the social, economic, and political impact of these changes.

1.1.1 Working of IoT:

The entire IoT process starts with the devices themselves like smart phones, smart watches, electronic appliances like TV, washer which helps you to speak with the IOT platform.

The basic four fundamental components of an IoT system:

1) Sensors/Devices: Sensors or devices are a key component that helps you to gather live data from the encircling environment. All this data may have various levels of complexities. It might be an easy temperature monitoring sensor, or it's going to be within the variety of the video feed. A device may have various varieties of sensors which performs multiple tasks other than sensing. Example, A transportable device could be a device which has multiple sensors like GPS, camera but your smart phone isn't ready to sense these items.

2) Connectivity: All the collected data is shipped to a cloud infrastructure. The sensors should be connected to the cloud using various mediums of communications. WAN, Bluetooth, Wi-Fi, satellite networks or mobile, etc. are the communication medium.

3) Data Processing: after the data is collected, it Is transferred to the cloud, the collected data is processed by the software. This process will be just checking the temperature, reading on devices like AC or heaters. However, it can sometimes even be very complex like identifying objects, using computer vision on video.

4) User Interface: the data must be available to the end-user in how it might be achieved by triggering alarms on their phones or sending them notification through email or text message. Sometimes the user needs an interface so that it can check data on their IoT system. As an example, the user includes a camera installed in his home. He wants to access recording and everyone feeds with the assistance of an internet server**.**

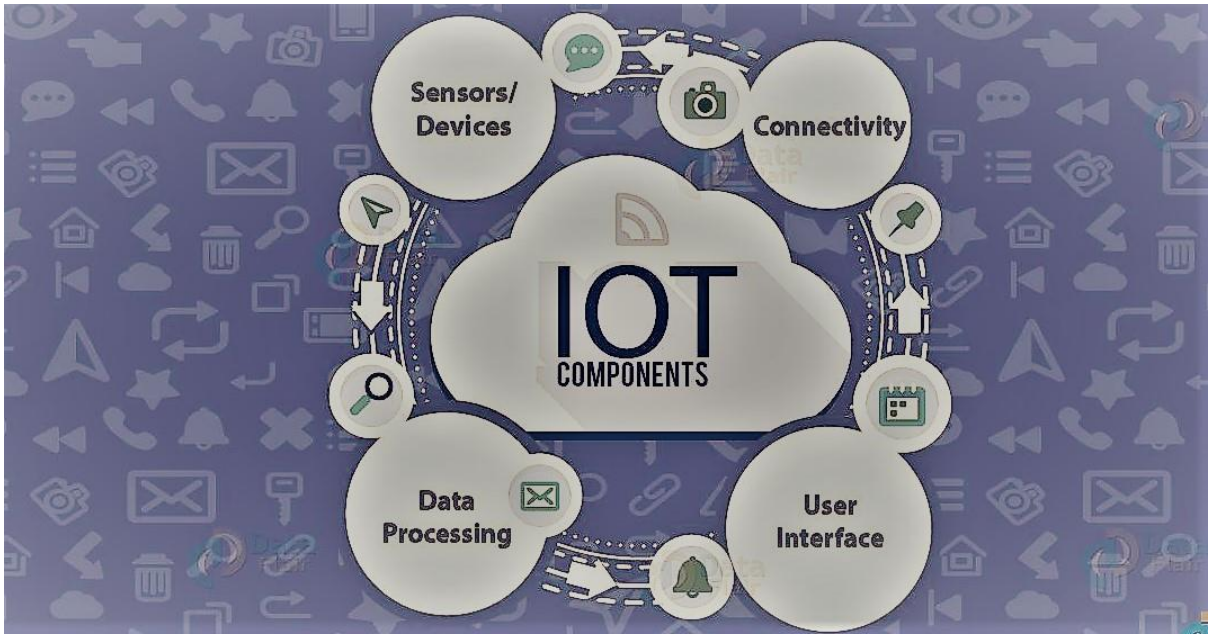


Fig1. Fundamental Component of IoT System.

1.1.2 Architecture of IoT:

There is not any single architecture of IoT on which Everyone Agreed. Different architectures are proposed by different researchers.

Three- and Five-Layer Architectures:

The most basic architecture is three-layer architecture. it had been introduced within the early stages of research during this area. it's three layers, namely, the perception, network, and application layers. (i) The perception layer is the physical layer, which has sensors for sensing and gathering information about the environment. It senses some physical parameters or identifies other smart objects within the environment. (ii) The network layer is to blame for connecting to other smart things, network devices, and servers. Its features are used for transmitting and processing sensor data. (iii) the applying layer is to blame for delivering application specific services to the user. It defines various applications during which the web of Things will be deployed, as an example, smart homes, smart cities, and smart health. The three-layer architecture defines the most idea of the web of Things, but it's not sufficient for research on IoT because research often focuses on finer aspects of the web of Things. That's why, we've got more layered architectures proposed within the literature. One is that the five-layer architecture, which additionally includes the processing and business layers [3–6]. perception, transport, processing, application, and business layers are the five layers. The role of the perception and application layers is the same because of the architecture with three layers. We outline the work of the remaining three layers. (i) The transport layer transfers the sensor data from the perception layer to the processing layer and contrariwise through networks like wireless, 3G, LAN, Bluetooth, RFID, and NFC. (ii) The processing layer is additionally called the middleware layer. It stores, analyses, and processes huge amounts of information that comes from the transport layer. It can manage and supply a various set of services to the lower layers. It employs many technologies like databases, cloud computing, and large processing modules.

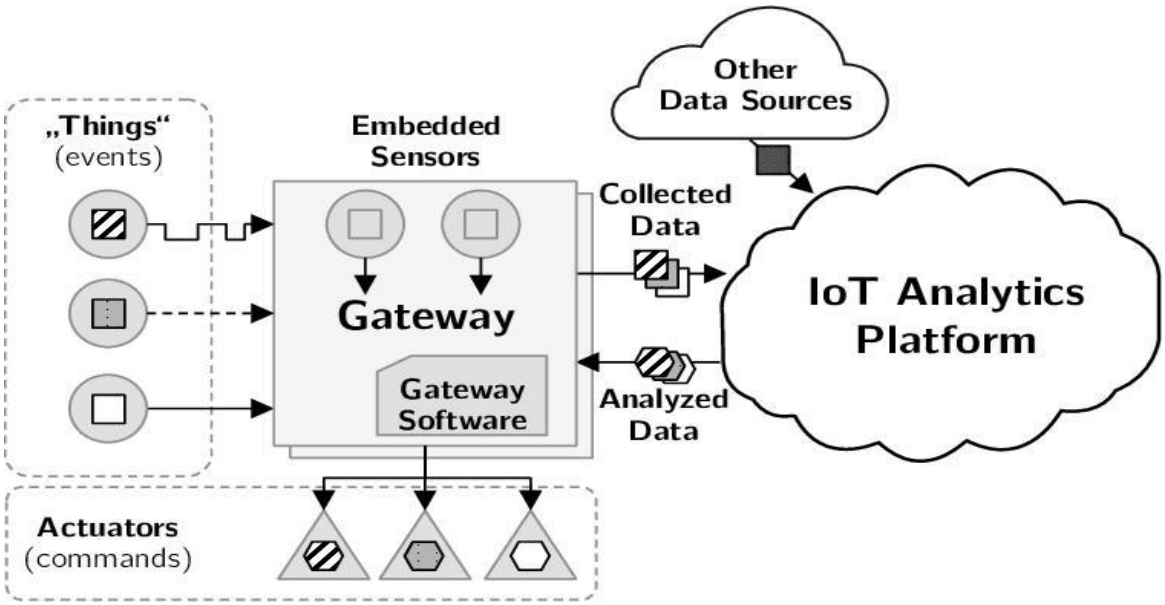


Fig2. IoT platform Architecture.

1.1.3 IoT − Key Features: One of the most important features of IoT is that it includes artificial intelligence, connectivity, sensors, active engagement, and tiny device use. a quick review of those features is given below:

1) AI – IoT- essentially makes virtually anything “smart”, meaning it enhances every aspect of life with the facility of knowledge collection, computer science algorithms, and networks. This will mean something as simple as enhancing your refrigerator and cabinets to detect when milk and your favourite cereal go, and to then place an order along with your preferred grocer.

2) Connectivity – New enabling technologies for networking, and specifically IoT networking, mean networks are not any longer exclusively tied to major providers. Networks can exist on a way smaller and cheaper scale while still being practical. In between its system devices IoT creates these networks.

3) Sensors – IoT will not be given much importance without sensors. They act as defining instruments which transform IoT from a typical passive network of devices into a vigorous system capable of real-world integration

4) Active Engagement – In today’s world the interaction between people happens with passive engagement [3]. IoT initiated a replacement of active content, service engagement or product

5) **Small Devices –** Devices, as predicted, became smaller, cheaper, and more powerful over time. IoT exploits purpose-built small devices to deliver its precision, scalability, and flexibility.

1.1.4 IoT – Advantages:

The advantages of IoT span across every area of lifestyle and business are as follows:

1) **Improved Customer Engagement –** Current analytics suffer from blind-spots and significant flaws in accuracy; and as noted, engagement remains passive. IoT completely transforms this to attain richer and more practical engagement with audiences.
2) **Technology Optimization –** the identical technologies and data which improve the customer experience also improve device use, and aid in additional potent improvements to technology.
3) **Reduced Waste. -**Current analytics give us superficial insight, but IoT provides real-world information resulting in more practical management of resources.
4) **Enhanced Data Collection –**In modern Society there is limitation for data collection moreover there are limitations of design for passive use. IoT breaks it out of these spaces, and places it exactly where humans really need to travel to investigate our world. It allows an accurate picture of everything.

1.1.5 Disadvantages IOT:

1)Security: IoT technology creates an ecosystem of connected devices. Although, while performing this process, the system can offer small authentication control despite sufficient security measures.

2)Privacy: the utilization of IOT, exposes a considerable amount of private data, in extreme detail, without the user's active participation. This creates many privacy issues.

3)Flexibility: there's an enormous concern regarding the flexibleness of an IoT system. it's mainly regarding integrating with another system as there are many diverse systems involved within the process.

4)Complexity: the look of the IOT system is additionally quite complicated. Moreover, the deployment of this and its maintenance are not easy.

5)Compliance: IOT has its own set of rules and regulations. However, due to its complexity, the task of compliance is sort of challenging.

1.1.6 IoT – Sensors:

The most important hardware in IoT could be its sensors. These devices include energy modules, power management modules, RF modules, and sensing modules. RF modules manage communications through their signal processing, Wi-Fi, ZigBee, Bluetooth, radio transceiver, duplexer, and BAW. Sensing of the Sensing module is managed through active and passive management devices. Here may be a list of a number of the measurement devices utilized in IoT:

accelerometers	temperature sensors
magnetometers	proximity sensors
gyroscopes	image sensors
acoustic sensors	light sensors
pressure sensors	gas RFID sensors
humidity sensors	micro flow sensors

Table 1. IoT Sensors.

1.1.7 IoT – Software:

IoT software addresses its main areas of networking and action through embedded systems, platforms, partner systems, and middleware. These individual and master applications are to blame for data collection, device integration, real-time analytics, and application and process extension within the IoT network. They exploit integration with critical business systems (e.g., ordering systems, robotics, scheduling, and more) within the execution of related tasks.

1) Data Collection - This software manages sensing, measurements, light data filtering, light data security, and aggregation of information. It uses certain protocols to help sensors in connecting with real-time, machine-to-machine networks. Then the collected data is distributed according to the settings. It also works in the opposite direction by sending back data over devices. The system transmits all received data to a central hub.
2) Device Integration - Software supporting integration binds (dependent relationships) all system devices to make the body of the IoT system. It

confirms the stable networking between devices and also maintains cooperation between those devices. These applications are the defining software technology of the IoT network because without them, it's not an IoT system. They manage the assorted applications, protocols, and limitations of every device to permit communication.

3) Real-Time Analytics - These applications take data or input from various devices and convert it into viable actions or clear patterns for human analysis. They analyse information supporting various settings and styles so as to perform automation-related tasks or provide the information required by industry.

4) Application and Process Extension - These applications extend the reach of existing systems and software to permit a wider, more practical system. They integrate predefined devices for specific purposes like allowing certain mobile devices or engineering instruments access. It supports improved productivity and it supports more accurate data collection.

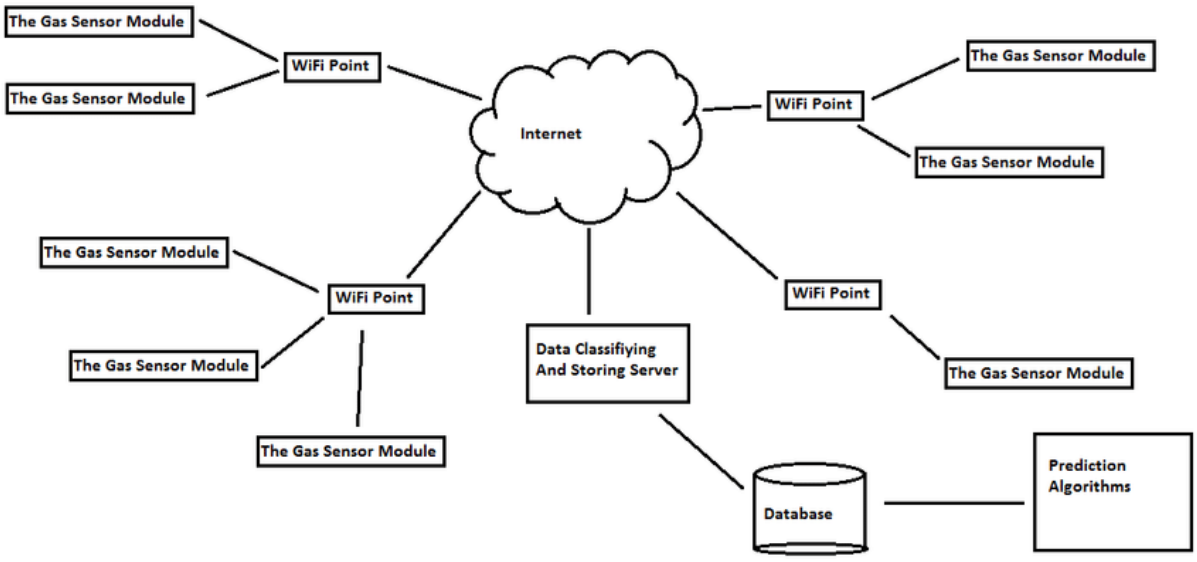


Fig 3. IOT Data Analytics

1.1.8 IoT − Technology and Protocols:

The IoT paradigm has brought with it a decided change during a number of areas starting from hardware design to services offered by software companies. one amongst the aspects most suffering from this is often that of Communication Protocols.

A communication protocol may be thought of as a language that's utilized by two or more machines to speak to every other. It's a collection of rules that are followed by the 2 devices so as to form a sense out of the messages that they pass to every other. Communication Protocols are extremely essential in distributed systems, where different parts of the identical process are administered at over one location, significantly distant from one another. The systems polishing off the processes is also heterogeneous in nature, warranting a standard set of instructions for both systems to speak.

The baby steps of the IoT boom are rooted within the spread of Cyber-Physical systems. The notion of physical devices connected to the net and spending data to and receiving data from it's the spine of a sensible implementation of an IoT solution. This added a brand-new layer of complexity to the prevailing definition of Communication Protocols.

The IoT revolution holds plenty of promise, the implications of which are only viable if there's effective machine-to-machine (M2M) communication, and to aim for real-time M2M communication over the net [4]. The concept of a tool being connected to the net was only considered to be the merchandise of human interaction until this time, and not a result of an autonomous decision. Therefore, protocols considering communication with the net were always a trade-off between unreliable and slow.

1.1.8.1 UDP, or User Datagram Protocol, may be a faster way of communication between two computers by reducing on acknowledgement procedures. it's more of a hearth and forget alternative to the stringent processes utilized by TCP to make sure that the complete message reaches the destination in proper order. As a result, UDP is a smaller amount reliable than TCP/IP but is far faster too. An awfully simplistic solution for rapid prototyping of M2M projects was considered to be UDP thanks to an awfully low overhead in terms of headers also as delay and a straightforward founded routine.

1.1.8.2 MQTT or MQ Telemetry Transport may be a lightweight connectivity protocol geared for IoT applications. it's supports the TCP/IP stack which uses the publish/subscribe method for transportation of information. it's open-ended and supports a high level of scaling, which makes it a perfect platform for development of Internet of Things (IoT) solutions.

MQTT hosts a variety of useful features which make it suitable for IoT oriented applications. to place it in simpler terms, imagine a bulletin board where you'll put up notes

or posters. Whenever you have got something interesting that you would like to place up, you design an ad along with your topic written in large, noticeable fonts, and you post it au fait the bulletin board.

MQTT functions

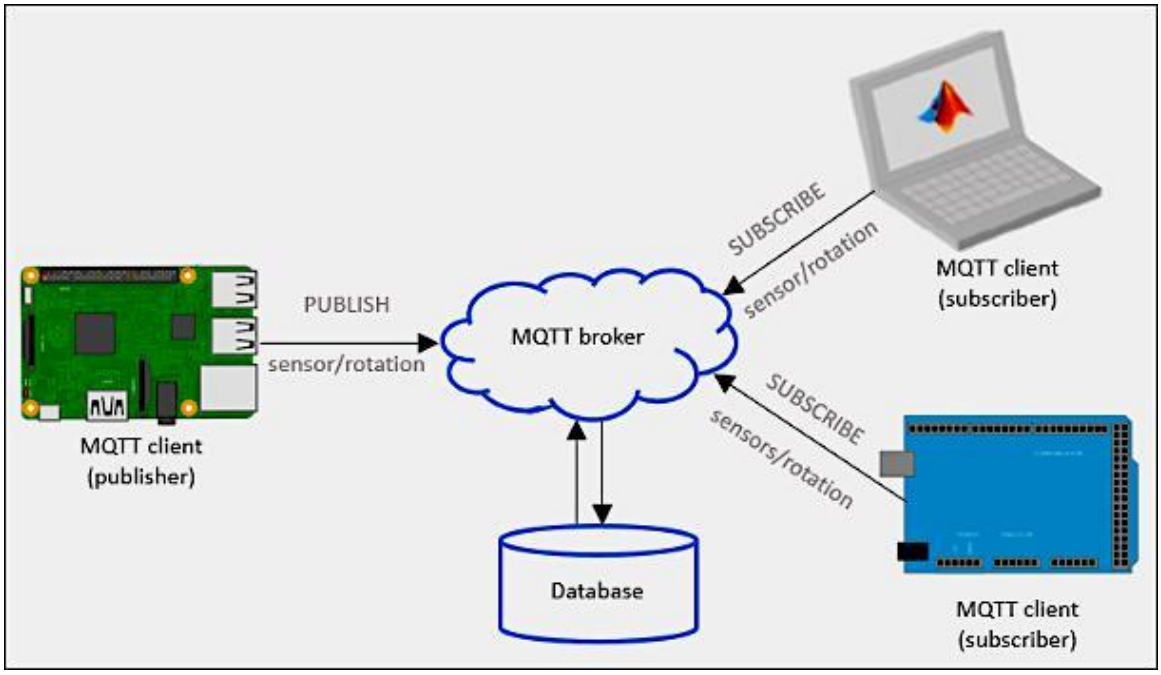


Fig 4. MQTT Protocol for IoT.

MQTT has two broad categories of participating devices. they're called brokers and clients. The clients are those devices which might access or modify data, and brokers are those which host and relay data.

MQTT works on a paradigm called the publish or subscribe method. A client can publish data regarding a particular parameter to the broker under a subject. Another client inquisitive about this subject can purchase this subject and receive regular updates on messages under the subject.

MQTT offers a top quality of service, which from an IoT standpoint is actually the priority attached to the message. a vital message should reach the destination in any case, so it's given a much better QoS, so transmission could also be slow, but delivery is guaranteed. A dynamic data source that prioritizes speed over efficiency, though, is assigned a lower QoS, so it's more of a fire-and-forget affair, like UDP.

MQTT can retain the last good message received under a subject, which it sends to subscribers who subscribe after the chain has been set in motion. This enables

asynchronous connection of subscribers within an existing network of clients and brokers. This also provides a facility to test for redundancy and data loss.

An MQTT client contains a property called the last will and testament. This property allows a client which has disconnected abruptly to send a message to the broker. An SOS call of sorts, which might be used for autonomous regeneration of a wireless sensor network, detection and debugging of stray nodes and outliers furthermore as a routine to make sure a faultless cycle in data flow of a wireless sensor network.

1.1.8.3 CoAP could be a web transfer protocol supporting the remainder model. It's mainly used for lightweight M2M communication thanks to its small header size. it's designed especially for constrained networks and systems within the net of Things paradigm, hence the name, Constrained Application Protocol. CoAP mimics HTTP in terms of user visibility, and from that standpoint, reading sensor values is actually like making an HTTP request. CoAP is taken into account to be a future-proof protocol within the sense that when, because the Gartner hype chart predicts, 50 billion devices are connected to the net, further expansion will necessitate the event of low-cost, low-consumption lightweight devices [5]. CoAP is intended to figure on systems as basic as 10kB RAM systems.

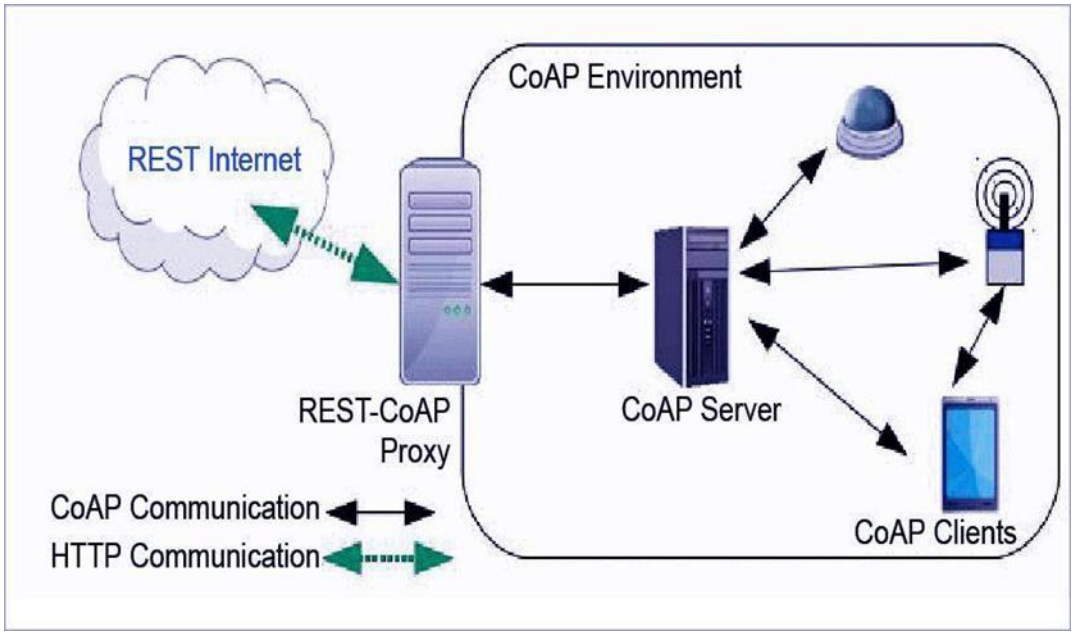


Fig 5. CoAP Protocol for IoT

CoAP is taken into account to be a future-proof protocol within the sense that when, because the Gartner hype chart predicts, 50 billion devices are connected to the net, further expansion will necessitate the event of low-cost, low-consumption lightweight devices [5]. CoAP is intended to figure on systems as basic as 10kB RAM systems.

One of the more interesting features of CoAP is the ability to find nodes within a network. This can be especially useful just in case of designing low power wireless sensor networks which are autonomous and self-healing. Problems regarding scalability will be overcome using CoAP for wireless sensor networks because the discovery property of nodes renders any node discovery routine redundant.

CoAP is made upon the UDP stack, which is the primary difference in comparison with HTTP or MQTT. This makes it faster and more resource optimized instead of resource intensive. However, this also makes it less reliable than HTTP or MQTT, and QoS factors remain static just in case of CoAP. However, the meagre 4-byte header makes it an exquisite option for continuous streaming systems like environmental monitoring sensor networks.

IoT primarily exploits standard protocols and networking technologies. However, the foremost enabling technologies and protocols of IoT are RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and Wi-Fi-Direct. These technologies support the particular networking functionality needed in an IoT system in contrast to a regular uniform network of common systems. IoT primarily exploits standard protocols and networking technologies. However, the foremost enabling technologies and protocols of IoT are RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and WIFI-Direct. These technologies support the particular networking functionality needed in an IoT system in contrast to a regular uniform network of common systems.

NFC and RFID RFID (radio-frequency identification) and NFC (near-field communication) are used for simple, low energy, and versatile options for identity and access tokens, connection bootstrapping, and payments. RFID technology employs 2-way radio transmitter-receivers to spot and track tags related to objects. NFC consists of communication protocols for electronic devices, typically a mobile device and a regular device. Low-Energy Bluetooth This technology supports the low-power, long-use need of IoT function while exploiting a regular technology with native support across systems. Low-Energy Wireless This technology replaces the foremost power-hungry aspect of an IoT system. Though sensors and other elements can power down over long periods, communication links (i.e., wireless link) must remain in listening mode. Low-energy wireless not only reduces consumption, but also extends the lifetime of the device through less use.

Radio Protocols Zig-Bee, Z-Wave, and Thread are radio protocols for creating low-rate private area networks. These technologies are low-power, but they offer high throughput

unlike many similar options. This increases the facility of small local device networks without the everyday costs.

LTE-A LTE-A, or LTE Advanced, delivers a serious upgrade to LTE technology by increasing not only its coverage, but also reducing its latency and raising its throughput. It gives IoT an incredible power through expanding its range, with its most vital applications being vehicle, UAV, and similar communication.

Wi-Fi-Direct eliminates the need for an access point. It allows peer-to-peer connections with the speed of Wi-Fi, but with lower latency. Wi-Fi-Direct eliminates a component of a network that always bogs it down, and it doesn't compromise on speed or throughput.

1. NFC and RFID RFID (radio-frequency identification) and NFC (near-field communication) provide simple, low energy, and versatile options for identity and access tokens, connection bootstrapping, and payments. RFID technology employs two-way radio transmitter-receivers to identify and track tags associated with objects. NFC consists of communication protocols for electronic devices, typically a mobile device and an everyday device.
2. Low-Energy Bluetooth, this technology supports the low-power, long-use need of IoT function while exploiting an everyday technology with native support across systems.
3. Low-Energy Wireless, this technology replaces the foremost power-hungry aspect of an IoT system. Though sensors and other elements can power down over long periods, communication links (i.e., wireless links) must remain in listening mode. Low-energy wireless not only reduces consumption but also extends the lifetime of device through less use.
4. Radio Protocols ZigBee, Thread, and Z-Wave are radio protocols used for creating low-rate private area networks. These technologies are low-power, but they offer high throughput unlike many similar options. This increases the facility of small local device networks without the everyday costs.
5. LTE-A LTE-A, or LTE Advanced, delivers a serious upgrade to LTE technology by increasing not only its coverage, but also reducing its latency and raising its throughput. It gives IoT an incredible power through expanding its range, with its most vital applications being vehicle, UAV, and similar communication.
6. WIFI-Direct WIFI-Direct eliminates the need for an access point. It allows peer-to-peer connections with the speed of Wi-Fi, but with lower latency. WIFI-Direct eliminates a component of a network that always bogs it down, and it doesn't compromise on speed or throughput.

1.1.9 IoT – Security:

Every connected device creates opportunities for attackers. These vulnerabilities are broad, even for one small device. The risks posed include data transfer, device access, malfunctioning devices, and always-on or always-connected devices. The most challenges in security remain the safety limitations related to producing low-cost devices, and also the growing number of devices which creates more opportunities for attacks.

Security Spectrum - The definition of a secured device spans from the foremost simple measures to classy designs. Security should be thought of as a spectrum of vulnerability which changes with time as threats evolve. Security must be assessed, supported user needs and implementation. Users must recognize the impact of security measures because poorly designed security will create more problems than it solves.

Challenges beyond costs and the ubiquity of devices, other security issues plague IoT:

1) **Unpredictable Behaviour** – The sheer volume of deployed devices and their long list of enabling technologies means their behaviour within the sphere are going to be unpredictable. a specific system could even be simple and within administration control, but there are not any guarantees about how it'll interact with others.

2) Device Similarity – IoT devices are almost uniform. They utilize the identical connection technology and components. If one system or device suffers from vulnerability, more have the identical issue.

3) Problematic Deployment – one altogether the foremost goals of IoT remains to position advanced networks and analytics where they previously couldn't go. Unfortunately, this creates the matter of physically securing the devices in these strange or easily accessed places.

4) Long Device Life and Expired Support – one altogether the benefits of IoT devices are longevity, however, that long life also means they'll outlive their device support. Compare this to traditional systems which usually have support and upgrades long after many have stopped using them. Orphaned devices and abandonware lack the identical security hardening of other systems thanks to the evolution of technology over time.

5) No Upgrade Support – Many IoT devices, like many mobile and tiny devices, aren't designed to allow upgrades or any modifications. Others offer inconvenient upgrades, which many householders ignore, or fail to notice.

6) Poor or No Transparency – Many IoT devices fail to supply transparency with respect to their functionality. Users cannot observe or access their processes, and are left to assume how devices behave. they have no control over unwanted functions or data collection; furthermore, when a manufacturer updates the device, it's visiting brings more unwanted functions.

7) No Alerts – Another goal of IoT remains to supply its incredible functionality without being obtrusive. This introduces the matter of user awareness. Users don't monitor the devices or know when something goes wrong. Security breaches can persist over long periods of time without detection.

8) Device Malfunction - IoT introduces a deeper level of automation which could have control over critical systems, and systems impacting life and property. When these systems fail or malfunction, they'll cause substantial damage; as an example, if an IoT furnace system experiences a glitch, it's visiting fail in an unoccupied home and cause frozen pipes and water damage. This forces organizations to form measures against it.

9) Cyber Attacks - IoT devices expose an entire network and anything directly impacted to the danger of attacks. Though those connections deliver powerful integration and productivity, they also create the correct opportunity for mayhem in the form of a hacked stove or fire safety system. the best measures against this address the foremost vulnerable points, and provide custom protections like monitoring and access privileges.

1.1.10 Some of the most effective measures against attacks prove simple:

1) Built-in Security – Individuals and organizations should seek hardened devices, meaning those with security integrated in the hardware and firmware.

2) Encryption – This must be implemented by the manufacturer and through user systems.

3) Risk Analysis – Organizations and individuals must analyse possible threats in designing their systems or choosing them.

4) **Authorization** – Devices, whenever possible, must be subject to privilege policies and access methods.

1.1.11 : IOT Applications :

1) **Wearable** – Wearable technology could be a hallmark of IoT applications and is one of the earliest industries to own deployed IoT at its service [6]. We happen to work out Fitbits, rate monitors and smart watches everywhere nowadays. one in all the lesser-known wearable includes the Guardian glucose monitor. The device is developed to help people affected by diabetes. It detects glucose levels within the body, employing a tiny electrode called glucose sensor placed under the skin and relays the knowledge via oftenest to a monitor.
2) **Smart Home Applications** – When we discuss IoT Applications, Smart Homes are probably the primary thing that we predict. There is also a home Automation System in the system that performs function based on the musical notes.
3) **Health Care** – IoT applications can easily turn reactive medical-based systems into proactive wellness-based systems. The resources that current medical research uses, lacks in the critical real-world information. It mainly uses leftover data, controlled environments, and the volunteers for medical exams. IoT can open different ways to a sea of valuable data through analysis, real-time field data, and testing. The web of Things also improves the present devices in power, precision, and availability. IoT focuses on creating systems instead of just equipment.
4) **Smart Cities** – By now I assume, most of you want to have heard about the term Smart City. The hypothesis of the optimized traffic system discussed earlier is one in all the numerous aspects that constitute a sensible city [7]. The thing about the smart city concept is that it is very specific to a city. The issues faced in different cities are very different . The issues in the city are different from the big apple. Even global issues, like finite clean potable, deteriorating air quality and increasing urban density, occur in numerous

intensities across cities. Hence, they affect each city differently. The Government and engineers can use IoT to research the often-complex factors of planning specific to every city [8]. The utilization of IoT applications can resolve the main problems in areas like water management, waste control, and emergencies.

5) **Agriculture** – Statistics estimate the ever-growing world population to succeed in nearly 10 billion by the year 2050[9]. To feed such an enormous population one has to marry agriculture to technology and procure best results. There are numerous possibilities during this field. one in all of them is the Smart Greenhouse. A greenhouse farming technique enhances the yield of the crops by somehow controlling environmental parameters. However, manual handling leads to production loss, energy loss, and labour cost, making the method less effective. A greenhouse with embedded devices not only makes it easier to be monitored but also, enables us to regulate the climate inside it. Sensors measure different parameters consistent with the plant requirement and send it to the cloud. It then processes the information and applies an impression action.
6) **Industrial Automation** – This is one in all the fields where both faster developments, also because the quality of products, are the critical factors for a better Return on Investment. With IoT Applications, someone could easily even re-engineer products and their packaging to deliver better performance in both cost and customer experience. IoT here can sway game changing with solutions for all the subsequent domains in its arsenal.

CHAPTER 2 : **MOTIVATION**

Mother Earth is dealing with a severe increase in climate change and its consequences that have adverse effects on living organisms. Air pollution is of greater concern to the environmentalists and climate change scientists. Emission of various poisonous gases from industries and vehicles are not only hazardous for the terrestrial organism, but the marine life is also getting adversely affected. World's population is becoming increasingly urban, the cities are under pressure to remain livable. Health problems are arising daily due to poor air quality and are in increase like heart diseases, lung cancer, stroke, and respiratory diseases including asthma. Poor air quality poses a significant risk to the vulnerable section of the society such as children, asthmatic, pregnant women, and the elderly persons. As per WHO statics, millions of premature death cases are reported due to air pollution every year worldwide [28]. In recent years, the air quality of the cities has become one of the major points of concern around the world. Thus, it is necessary to constantly monitor the air quality index of a city to make it smart and livable. Around the world, governments are building smart cities to keep a check on these problems and provide a healthy life for its inhabitants. These cities will utilize WSNs, advanced communication networks, and intelligent systems to solve future challenges and create new services. Arduino microcontroller connected to sensors via an inbuilt Wi-Fi module but is also responsible to send the recorded data to the Thing-Speak, an open source cloud platform on which data can be stored and retrieved via hypertext transfer protocol (HTTP) over the internet [29]. The cities have installed real-time air quality monitoring systems with low-cost IoT

enabled WSN technology is the future for the coming smart cities around the world. Real-time monitoring of the air quality requires the live data transfer between the devices over the internet and it can be visualized using an Android Application which is cost efficient as only one-time installation cost is involved. In the Internet of Things (IoT) based applications. It reduces the mobilization of system hardware at different locations. To achieve efficient IOT accomplishment for an application; the proper sensing and monitoring system is essential. The devices can communicate with each other with the help of Machine to Machine (M2M) communication and the physical devices can be controlled digitally [30]. An IOT based monitoring system can be one possible solution. if any malfunctioning is detected, a message can be passed on immediately to the controlling unit so that further action can be taken and damage can be avoided. If a device can be installed in factories that can constantly monitor the working of the installed machines. Air pollution,

climate change, and its consequences are of a great concern to the environmentalists and climate change conditions. Emission of various poisonous gases from vehicles and industries are not only hazardous for the terrestrial organism, but marine life is getting adversely affected. Due to this, in recent years, the air quality of the cities has become one of the major causes of concern around the world. Thus, it is necessary to constantly monitor the air quality index of a city to make it smart and livable.

Around the world, governments are building the smart cities to keep a check on these problems and provide a healthy life for its inhabitants [31]. The Indian government is in the process to build 100 smart cities by 2050[32]. These cities will utilize advanced communication networks, WSNs, and intelligent systems to solve future challenges and create new services. Real-time monitoring of the air quality requires the live data transfer between the devices over the internet and it can be visualized using an Android Application. It reduces the mobilization of system hardware at different locations, which is cost efficient as only one-time installation cost is involved. The Internet of Things (IoT) based applications.

Air pollution is the main concern for all the developing as well as the developed countries. As per the studies after poverty Air pollution is the major problem faced by the world today. Humans are at the verge of treating development and advancements compromising with the life-threatening issues. The rapid increase in the concentration of various harmful gases in our atmosphere causes has adverse effects on humans and ecological community. Air pollution may be linked with lung's impairments, cardiovascular diseases, lung cancer, and community acquired pneumonia. This paper deals with the monitoring of harmful pollutants and simultaneously displaying the sensors data over VGA display. Along with monitoring of gas pollutants, making people aware of the adverse effects of those is required by keeping regular track of these gases. Suspended particulate matter (SPM), carbon monoxide (CO), oxides of nitrogen (NO), oxides of sulfur (SO), lead aerosol, volatile organic compounds (VOC) are the main constituents of air pollutants. This will be led them to take some precautionary measures and steps to fight or overcome these silent Killer pollutants. These not only making our live worse but also decreasing their average life expectancy by 1.8 years. If people along with the initiative taken by the government contribute to fight against this problem, they can be controlled and reduced up to some extent for the better future. The rise in the population in upcoming years mainly in the developing countries will lead to lack of capital for the major concern like air pollution control and it signifies that the conditions will worsen in many parts that will reach

megacity status. As far as sustainability is concerned the poverty, ozone depletion and air pollution are the major problem in the world. In developing countries like India, Air pollution has become the main reason of concern. The pollution due to constructions, industries and traffics along with the several other factors worsening the situation. They are producing a large number of particulate matters with the air. Air quality index of several cities are not event not desirable or suitable for the humans. The AQI level of these cities are above 300 on the scale of 0 to500. This provides the information of pollutants in the region of monitoring stations. These stations are installed by CPCB for air pollution monitoring. But at the same time the number of monitoring stations and the frequency of the sampling of the pollutants is not up to the mark. The monitoring is carried out for 24 hours (8 hourly sampling of particulate matter) which is not providing a real time data to its users. On the contrary side the number of monitoring stations are very limited in comparison to the geographical area of any city or state. This is the place where this portable device will play a vital role in providing overcoming the limitation of CPCB monitoring stations . We can get a real time data at the frequency of 15 samples per minute of the surrounding with the radius of 10 m2. Although this is a standalone device and need no any external heavy power sources it can easily be carried cloud along with the user in manner to get the sensor data of the nearby environment. This will make and provide a mobile monitoring service to users covering large geographical area in comparison to traditional monitoring stations. The prominent reason behind this is to provide an easy access to these data over wide number of users by facilitating cloud storage service. The sensors data can be fetched by different users through the shareable link of the cloud platform. This research is aimed to develop a cloud-based air quality monitoring and heart care system, which can be accessed on mobile application and personal computer via android application and web server. The project model will take the form of standalone device which monitors the air quality nearby and sends the data to the cloud and can easily be accessed by the public. This will also provide the record of data to the remote users so that they can use it accordingly their purpose the collected data will be stored on the cloud.

CHAPTER 3 : **Literature Review**

3.1 Chen Xiaojun et al. [37] introduced the Internet of Things (IOT) into the field of environmental protection, they put forward a kind of real-time air pollution monitoring and forecasting system. By using IOT, that system reduced the hardware cost into 1/10 as before. Besides the functions of conventional air automatic monitoring system, it was also exhibiting the function of forecasting development trend of air pollution within a certain time range by analyzing the data obtained by front-end perception system according to neural network technology.Thissystem used a large number of sensors to ensure monitoring accuracy, reduced monitoring cost and make monitoring data in monitoring area more systematic and perfect. A large number of field data provided by the front-end sensor network makes big data analysis in the background application layer more direct and effective and it is providing a real and effective decision-making basis for emergency response.

3.2 K. S. E. Phala et al. [38] presented an air quality monitoring system (AQMS) based on IEEE/ISO/IEC 21451 standards. In the development of AQMS, they had used the GSM wireless communication module. The developed system was capable of real-time measurement of air polluted gases such as CO2, CO, NO2, and SO2. The machine-to-machine communication of the air quality monitoring station and PC with the sink node was successfully implemented. Various gas sensor technologies were evaluated for the system and ultimately electrochemical and infrared sensors were used. The AQMS uses an array of sensors to take measurements of the ambient air surrounding it and wirelessly transmits the data to the base station. A graphical user interface (GUI), which makes it easy for end user(s) to interact with the system, was developed. Gas concentration values were plotted on the GUI. The defined calibration of the instruments at time interims assures that the desired accuracy is sustained.

3.3 Kan Zheng et al. [39] proposed a new method to implement the air quality monitoring system based on state-of-the-art Internet-of-Things (IoT) techniques. In this system, portable sensors collect the air quality information timely, which is transmitted through a low power wide area network. All air quality data are processed and analyzed in the IoT cloud. The completed air quality monitoring system, including both hardware and software, is developed and deployed successfully in urban environments. Experimental results show that the proposed system is reliable in sensing the air quality, which helps reveal the change patterns of air quality to some extent.

3.4 Marin B. Marinov et al. [40] presented an approach for cost-effective measurement of relevant environmental parameters, based on a scalable sensor array with integrated amperometric and infrared gas sensors. The device had been tested in the city and the measurement was compared with the output data of the local environmental control authority stations. The preliminary results show that this approach can be used as an economical alternative to the professional grade systems.

3.5 Rohini Shete et al. [41] mainly propounded a stand-alone system which is providing a dynamic datasheet about the parameters of the city environment. The system is using a low-cost low power ARM based minicomputer that is Raspberry Pi. It can communicate through Local Area Network (LAN) or the external Wi-Fi module. Commands from users are processed at the Raspberry Pi using Python language. The data can be monitored with other terminal devices like Laptop, Smartphone and Tablet which is endowed with the internet facility. This framework is giving access to real-time information about an urban environment which includes the parameters: temperature, humidity, pressure, CO and harmful air pollutants. The system aids in the sustainable growth of the city and improves the lives of the citizens. The ubiquitous availability of dynamic datasheets on the dashboard and the time to time graphical representation can help in planning the control measures against increasing pollution levels and create awareness among the people.

3.6 MukeshJha et al. [42] presented a novel sensor network design and implementation for advanced urban micro-climate monitoring and environment modeling for future adaptation of urban infrastructure. they presented an appropriate framework for monitoring, modeling and ultimately manipulating the urban microclimate for efficient adaptation of future urban infrastructure. They calibrate the various types of sensors (temperature, humidity, thermal camera, wind sensors) and install multiple sensor nodes throughout the city at different locations to collect and stream data. The node has its own communication module which is capable of communicating over the Internet. Thereby incorporating the ideas of Internet-of-Things. They explored various sensor network and data-processing architectures to support microclimate modeling through assimilation of recorded and analyzed data while minimizing the cost of sensors, their deployment and necessary communication and power infrastructures.

3.7 Stefan Nastic et al. [43] introduced the concept of software-defined IoT units – a novel approach to IoT cloud computing that encapsulates fine-grained IoT resources and IoT capabilities in well-defined APIs in order to provide a unified view on accessing, configuring and operating IoT cloud systems. Our software-defined IoT units are the fundamental building blocks of software-defined IoT cloud systems. They presented our framework for dynamic, on-demand provisioning and deploying such software-defined IoT

cloud systems. By automating provisioning processes and supporting managed configuration models, the framework simplifies provisioning and enables flexible runtime customizations of software-defined IoT cloud systems. They showed how they are used to abstract IoT resources and capabilities in the cloud, by encapsulating them in software-defined API. They presented automated unit composition and managed configuration, the main techniques for provisioning software-defined IoT systems. They demonstrated it advantages on a real-world IoT cloud system for managing electric fleet vehicles.

3.8 Pravin Gupta et al. [44] proposed a prototype for an Environmental Air Pollution Monitoring System for monitoring the concentrations of major air pollutant gases. The system uses low cost air-quality monitoring nodes consisting of low-cost semiconductor gas sensors with Wi-Fi modules. This system measures concentrations of gases such as CO, CO2, SO2 and NO2 using semiconductor sensors. Realization of data gathered by sensors is displayed on Raspberry pi 3 based Web Server. A MEAN stack is developed to display data over websites.

3.9 Chiwewe, Tapiwa M., and Jeoffrey Ditsela [47] collected air quality data from different cities of South Africa. Machine learning technique was applied to the data and prediction models were generated for ground level ozone.

3.10 Yurii Maslyiak et al. [49] proposed in which the architecture of software and hardware systems for environmental monitoring is considered. Its main tasks are determined. The prediction of air pollution by harmful emissions from motor vehicles using mathematical models and visualization of modeling results using this website. Mathematical models are represented in the form of differential equations. They describe dynamics and spatial distribution of harmful motor vehicles emissions.

CHAPTER 4 : **MACHINE LEARNING**

4.1 Machine Learning Definition:

Machine learning may be a sort of AI that allows a system to find out from data instead of through explicit programming. However, machine learning isn't a straightforward process because the algorithms ingest training data, it's then possible to supply more precise models that support that data. A machine learning model is the model which generates the output by training the model with the algorithms one provides. After training, once you provide a model with an input, you'll run an output for instance, a predictive algorithm will create a predictive model. Then, once you provide the predictive model with data, you'll receive a prediction supporting the info that trained the model.

Machine learning, one in every of the highest emerging sciences, has a particularly broad range of applications. However, many books on the topic provide only a theoretical approach, making it difficult for a newcomer to know the topic material.
Machine learning may be a subfield of applied science which has been generated with the study of pattern recognition and computational learning theory in computing. Machine learning explores the development and study of algorithms that may learn from and make predictions on data. Such algorithms operate by building a model from example inputs so as to create data-driven predictions or decisions, instead of following strictly static program instructions. Machine learning is mainly associated with the infrequently overlaps with computational statistics. it's strong ties to mathematical optimization, which deliver methods, theory and application domains to the sphere. Machine learning is utilized in a very range of computing tasks where designing and programming explicit algorithms is infeasible.

4.2 Machine learning Tasks:

Machine learning tasks are divided into many categories, betting on the character of the academic “signal” or “feedback” available to a learning system.

1) **Supervised learning:** The computer is presented with example inputs and their desired outputs, given by a “teacher”, and the goal is to learn a general rule that maps inputs to outputs. Supervised learning is analogous to training a child to walk. You will hold the child’s hand, show him how to take his foot forward, walk yourself for a demonstration and so on, until the child learns to walk on his own. Regression similarly, in the case of

supervised learning; you give concrete known examples to the computer. You say that for a given feature value x1 the output is y1, for x2 it is y2, for x3 it is y3, and so on. Based on this data, you let the computer figure out an empirical relation between x and y. Once the machine trained in this way with sufficient number of data points, now you would ask the machine to predict Y for a given X. Assume that you know the real value of Y for this given X, you will be able to deduce whether the machine's prediction is correct.

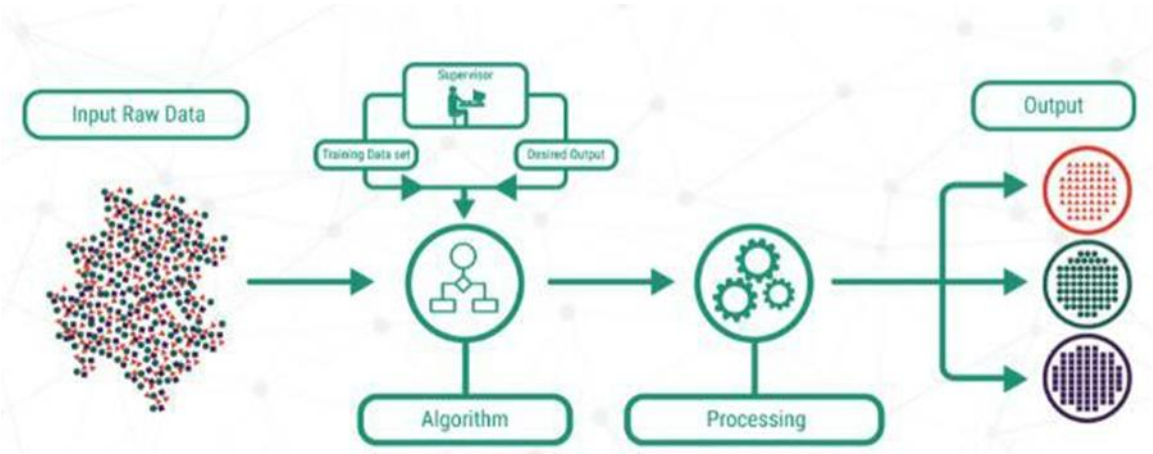


Fig 6. Supervised Learning

Thus, you will test whether the machine has learned by using the known test data. Once you are satisfied that the machine is able to do the predictions with a desired level of accuracy (say 80 to 90%) you can stop further training the machine. Now, you can safely use a machine to do the predictions on unknown data points, or ask the machine to predict Y for a given X for which you do not know the real value of Y. This training comes under regression that we talked about earlier.

Supervised Learning is where the AI really begins its journey. This technique applied successfully in several cases. Supervised Learning starts where AI starts its journey. This method was applied successfully in several cases.

2) **Unsupervised learning:** No labels are given to the learned algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discover hidden patterns in data) or a means towards the end. To arrive at answers to such questions, you can understand that the number of data points that the machine will require to deduce a strategy would be very large. In case of supervised learning, the machine can be trained with even a few thousands of data points. However, in case of unsupervised learning, the number of data points that is reasonably accepted for

learning starts in a few millions. These days, the data is generally abundantly available. The data ideally requires curetting. However, the amount of data that continuously flows in a social area network, in most cases data curation, is an impossible task.

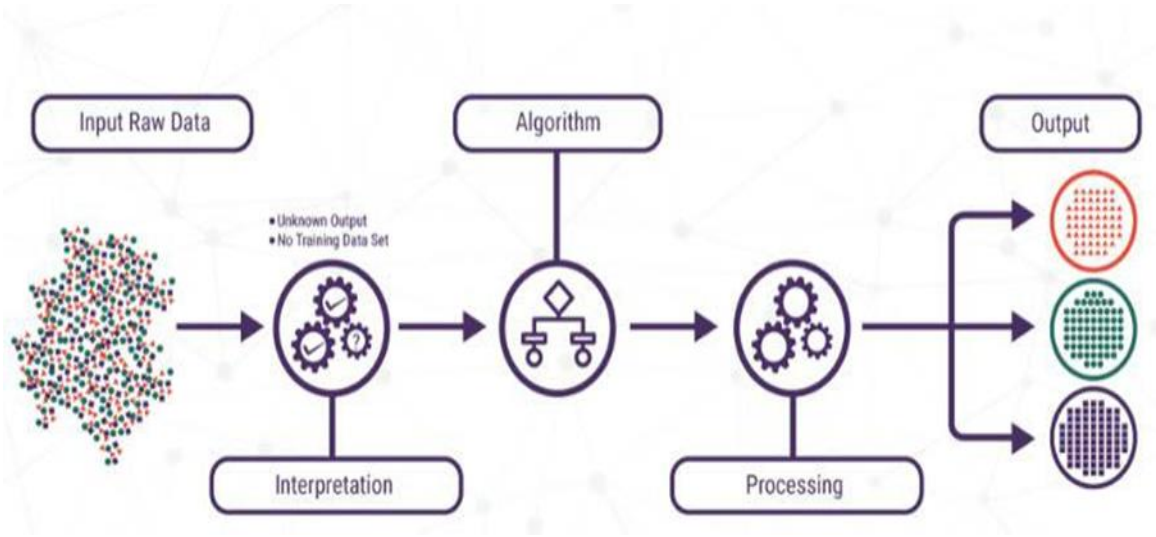


Fig 7. Unsupervised Learning

3) **Reinforcement learning:** A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle), without a teacher explicitly telling it whether it has been close to its goal or not. Another example learning to play a game by playing against an opponent. Consider training a pet dog, we train our pet to bring a ball to us. We throw a ball at a certain distance and ask the dog to fetch it back to us. Every time the dog did this right, we rewarded the dog. Slowly, the dog learns that doing the job rightly gives him a reward and then the dog starts doing the job the right way every time in future. Exactly, this concept is applied in "Reinforcement" type of learning. The technique is initially developed for machines to play games. The machine is given an algorithm to analyse all possible moves at each stage of game. The machine may select one of the moves in random. If the move is right, the machine is rewarded, otherwise it will be penalized. Slowly, the machine will start to differentiate between right and wrong moves and after several iterations would be penalized. Slowly, the machine will start to differentiate between right and wrong moves and after several iterations would learn to solve game puzzles with better accuracy. The accuracy of winning the game is improving as the machine plays more and more games.

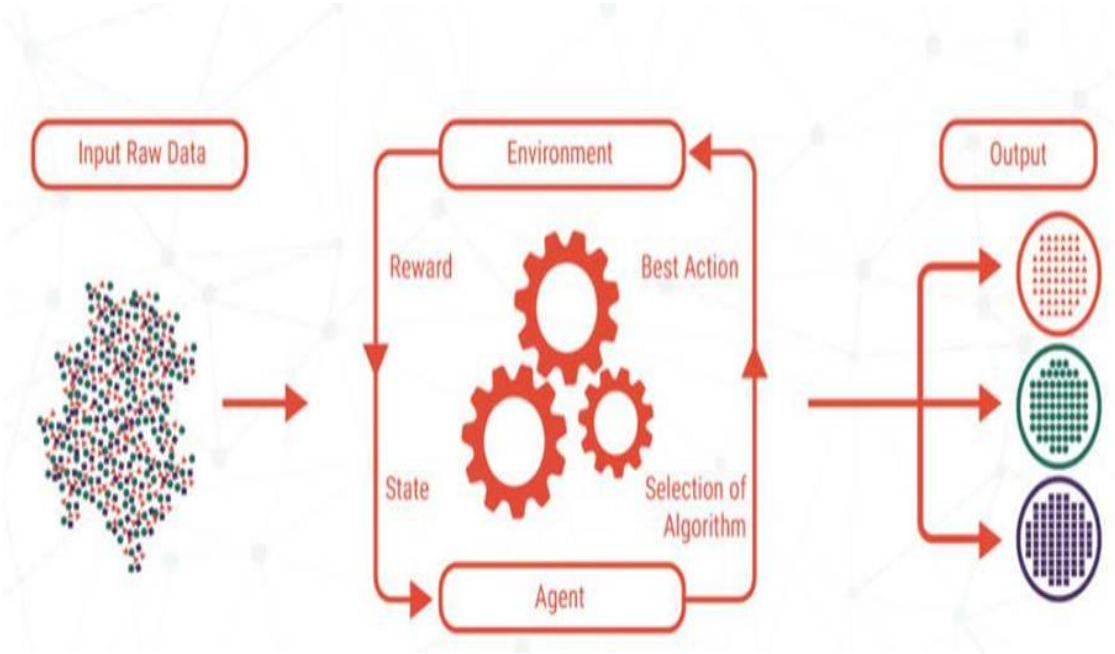


Fig 8. Reinforcement Learning

4) **Deep Learning:** Deep learning will be a model based on Artificial Neural Networks (ANN), more specifically Convolution Neural Networks (CNN) s. There are several architectures used in deep learning such as deep neural networks, deep belief networks, recurrent neural networks, and convolution neural networks. These networks have been successfully applied in solving problems of computer vision, speech recognition, natural language processing, bioinformatics, drug design, medical image analysis, and games. There are several other fields in which deep learning proactively applied. Deep learning requires huge processing power and humongous data, which is generally easily available these days.

5) **Deep Reinforcement**: Learning the Deep Reinforcement Learning (DRL) combines the techniques of both deep and reinforcement learning. The reinforcement learning algorithms like Q-learning had now combined with deep learning create a powerful DRL model. The technique has been a great success in the fields of robotics, video games, finance and healthcare. Many previously unsolved problems are now solved by creating DRL models. There is lots of research going on in this area and this was very actively pursued by the industries.

4.3 Machine learning Models:

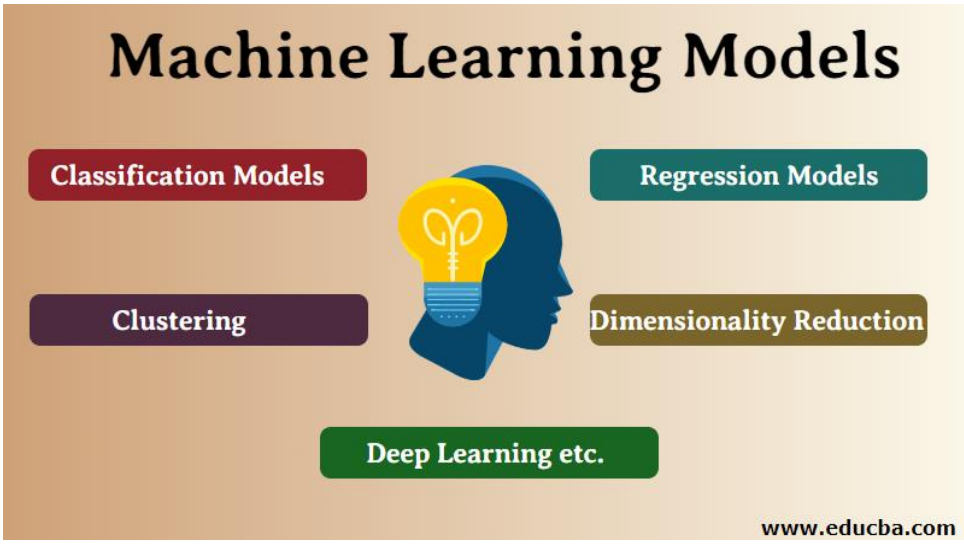


Fig 9. Machine Learning Models

4.3.1 Machine learning Models Definition : A machine learning model is the output of the training process and is defined as the mathematical representation of the real-world process. The machine learning algorithm find the patterns in the training dataset which is used to approximate the target function and is responsible for the mapping of the inputs to the outputs from the available dataset. These machine learning methods depend upon the type of task and are classified as Classification models, Regression models, Clustering, Dimensionality Reductions, Principal Component Analysis, etc.

4.3.2 Types Machine learning Models :

1) **Classification :** With respect to machine learning, classification is the task of predicting the type or class of an object within a finite number of options. The output variable for classification is always a categorical variable. For example, predicting an email is spam or not is a standard binary classification task. Now let's note down some important models for classification problems.
    a) K-Nearest neighbours algorithm – simple but computationally exhaustive.
    b) Naive Bayes – Based on Bayes theorem.
    c) Logistic Regression – Linear model for binary classification.
    d) SVM – can be used for binary/multiclass classifications.
    e) Decision Tree – 'If Else' based classifier, more robust to outliers.

f) Ensembles – Combination of multiple machine learning models clubbed together to get better results.

2) **Regression :** In the machine, learning regression is a set of problems where the output variable can take continuous values. For example, predicting the airline price can be considered as a standard regression task. Let's note down some important regression models used in practice.

   a) Linear Regression – Simplest baseline model for regression task, works well only when data is linearly separable and very less or no multicollinearity is present.
   b) Lasso Regression – Linear regression with L2 regularization.
   c) Ridge Regression – Linear regression with L1 regularization.
   d) SVM regression
   e) Decision Tree Regression etc.

3) **Clustering** : In simple words, clustering is the task of grouping similar objects together. It helps to identify similar objects automatically without manual intervention. We can not build effective supervised machine learning models (models that need to be trained with manually curated or labelled data) without homogeneous data. Clustering helps us achieve this in a smarter way. Following are some of the widely used clustering models:

   a) K means – Simple but suffers from high variance.
   b) K means++ – Modified version of K means.
   c) K medoids.
   d) Agglomerative clustering – A hierarchical clustering model.
   e) DBSCAN – Density-based clustering algorithm etc.

4) **Dimensional Reduction :** Dimensionality is the number of predictor variables used to predict the independent variable or target. Often in the real world datasets the number of variables is too high. Too many variables also bring the curse of overfitting to the models. In practice among these large numbers of variables, not all variables contribute equally towards the goal and in a large number of cases, we can actually preserve variances with a lesser number of variables. Let's list out some commonly used models for dimensionality reduction.

a) PCA : It creates lesser numbers of new variables out of a large number of predictors. The new variables are independent of each other but less interpretable.
b) TSNE : Provides lower dimensional embedding of higher dimensional data points.
c) SVD : Singular Value Decomposition is used to decompose the matrix into smaller parts in order to efficient calculation.

5) **Deep Learning :** Deep learning is a subset of machine learning which deals with neural networks. Based on the architecture of neural networks let's list down important deep learning models:
    a) Multi Layer Perception
    b) Convolution Neural Network
    c) Recurrent Neural Network
    d) Boltzmann Machine
    e) Autoencoders etc.

CHAPTER 5 : **AIR MONITORING SYSTEMS**

5.1 Introduction:

Air, being an invisible and dynamic medium, it is difficult to know if the air quality of a region is as per the air quality standards. It is imperative to monitor the air quality frequently to assess the pollution level in accordance with the National Air Quality Standards set under the Air (Prevention and Control of Pollution) Act, 1981. Air monitoring is done to measure the pollution level of an area – indoors or outdoors. The results of which indicate the status of the quality of the air we breathe.

That's why the term 'ambient air quality monitoring' was coined. An Air Monitoring System detects and measures the pollution in the surrounding. It represents the pollution data as concentrations of different pollutants – whether it is dust particles or gases.
Ambient air monitoring helps in evaluating and understanding pollution status and trends. It also helps to know whether pollution control strategies in place are working or not. Pollution monitoring also helps in determining the air purification method required, for instance, dry deposition, dilution, precipitation, chemical treatment, etc.

However, you might have come across two different terms in this aspect – Ambient Air Quality Monitoring System (AAQMS) and Continuous Air Quality Monitoring System (CAAQMS). Though both may sound the same, there's a difference between these monitoring systems, depending upon the data usage, equipment cost, operating cost, system reliability, and ease of operation.

5.2 Need for Air Quality Monitoring:

Clean, dry air consists primarily of nitrogen and oxygen—78 percent and 21 percent respectively, by volume. The remaining 1 percent is a mixture of other gases, mostly argon (0.9 percent), along with trace (very small) amounts of carbon dioxide, methane, hydrogen, helium, and more. Water vapour is also a normal, though quite variable, component of the atmosphere, normally ranging from 0.01 to 4 percent by volume; under very humid conditions the moisture content of air may be as high as 5 percent.

There are six major air pollutants that have been designated by the U.S. Environmental Protection Agency (EPA) as “criteria” pollutants—criteria meaning that the concentrations of these pollutants in the atmosphere are useful as indicators of overall air quality. The sources, acceptable concentrations, and effects of the criteria pollutants are summarized in the table.

pollutant	common sources	maximum acceptable concentration in the atmosphere	environmental risks	human health risks
carbon monoxide (CO)	automobile emissions, fires, industrial processes	35 ppm (1-hour period); 9 ppm (8-hour period)	contributes to smog formation	exacerbates symptoms of heart disease, such as chest pain; may cause vision problems and reduce physical and mental capabilities in healthy people
nitrogen oxides (NO and NO_2)	automobile emissions, electricity generation, industrial processes	0.053 ppm (1-year period)	damage to foliage; contributes to smog formation	inflammation and irritation of breathing passages
sulfur dioxide (SO_2)	electricity generation, fossil-fuel combustion, industrial processes, automobile emissions	0.03 ppm (1-year period); 0.14 ppm (24-hour period)	major cause of haze; contributes to acid rain formation, which subsequently damages foliage, buildings, and monuments; reacts to form particulate matter	breathing difficulties, particularly for people with asthma and heart disease

Table 2. Pollution Criteria

The gaseous criteria air pollutants of primary concern in urban settings include sulphur dioxide, nitrogen dioxide, and carbon monoxide; these are emitted directly into the air from fossil fuels such as fuel oil, gasoline, and natural gas that are burned in power plants, automobiles, and other combustion sources. Ozone (a key component of smog) is also a gaseous pollutant; it forms in the atmosphere via complex chemical reactions occurring between nitrogen dioxide and various volatile organic compounds (e.g., gasoline vapours). Airborne suspensions of extremely small solid or liquid particles called “particulates” (e.g., soot, dust, smokes, fumes, mists), especially those less than 10 micrometres (µm; millionths of a metre) in size, are significant air pollutants because of their very harmful effects on human health. They are emitted by various industrial processes, coal- or oil-burning power plants, residential heating systems, and automobiles. Lead fumes (airborne particulates less

than 0.5 μm in size) are particularly toxic and are an important pollutant of many diesel fuels. Except for lead, criteria pollutants are emitted in industrialized countries at very high rates, typically measured in millions of tons per year. All except ozone are discharged directly into the atmosphere from a wide variety of sources. They are regulated primarily by establishing ambient air quality standards, which are maximum acceptable concentrations of each criteria pollutant in the atmosphere, regardless of its origin.

Also , Global warming is recognized by almost all atmospheric scientists as a significant environmental problem caused by an increase in levels of certain trace gases in the atmosphere since the beginning of the Industrial Revolution in the mid-18th century. These gases, collectively called greenhouse gases, include carbon dioxide, organic chemicals called chlorofluorocarbons (CFCs), methane, nitrous oxide, ozone, and many others. Carbon dioxide, although not the most potent of the greenhouse gases, is the most important because of the huge volumes emitted into the air by combustion of fossil fuels (e.g., gasoline, oil, coal).

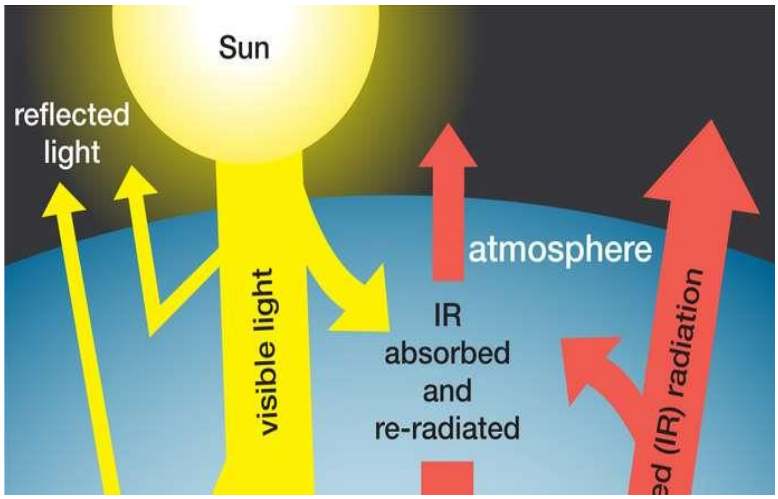


Fig 10. Global Warming

Carbon dioxide is considered a normal component of the atmosphere, and before the Industrial Revolution the average levels of this gas were about 280 parts per million (ppm). By 2020 the levels of carbon dioxide had reached 417 ppm, and they continue to increase at a rate of almost 3 ppm per year. Many scientists think that carbon dioxide should be regulated as a pollutant—a position taken by the EPA in 2009 in a ruling that such regulations could be promulgated. International cooperation and agreements, such as the Paris Agreement of 2015, would be necessary to reduce carbon dioxide emissions worldwide.

Because some air pollutants persist in the atmosphere and are carried long distances by winds, air pollution transcends local, regional, and continental boundaries, and it also may

have an effect on global climate and weather. For example, acid rain has gained worldwide attention since the 1970s as a regional and even continental problem. Acid rain occurs when sulphur dioxide and nitrogen oxides from the burning of fossil fuels combine with water vapour in the atmosphere, forming sulfuric acid and nitric acid mists. The resulting acidic precipitation is damaging to water, forest, and soil resources. It has caused the disappearance of fish from many lakes in the Adirondack Mountains of North America, the widespread death of forests in mountains of Europe, and damage to tree growth in the United States and Canada. Acid rain can also corrode building materials and be hazardous to human health. These problems are not contained by political boundaries. Emissions from the burning of fossil fuels in the middle sections of the United States and Canada are precipitated as acid rain in the eastern regions of those countries, and acid rain in Norway comes largely from industrial areas in Great Britain and continental Europe. The international scope of the problem has led to the signing of international agreements on the limitation of sulphur and nitrogen oxide emissions.

Another global problem caused by air pollution is the ozone depletion in the stratosphere. At ground level (i.e., in the troposphere), ozone is a pollutant, but at altitudes above 12 km (7 miles) it plays a crucial role in absorbing and thereby blocking ultraviolet radiation (UV) from the Sun before it reaches the ground. Exposure to UV radiation has been linked to skin cancer and other health problems. In 1985 it was discovered that a large “ozone hole,” an ozone-depleted region, is present every year between August and November over the continent of Antarctica. The size of this hole is increased by the presence in the atmosphere of chlorofluorocarbons (CFCs); these emanate from aerosol spray cans, refrigerators, industrial solvents, and other sources and are transported to Antarctica by atmospheric circulation. It had already been demonstrated in the mid-1970s that CFCs posed a threat to the global ozonosphere, and in 1978 the use of CFCs as propellants in aerosol cans was banned in the United States. Their use was subsequently restricted in several other countries. In 1987 representatives from more than 45 countries signed the Montreal Protocol, agreeing to place severe limitations on the production of CFCs. The efficacy of this legislation and global effort can be seen in the ozone layer’s continued recovery; in 2019 scientists recorded the smallest ozone hole above Antarctica since 1982.

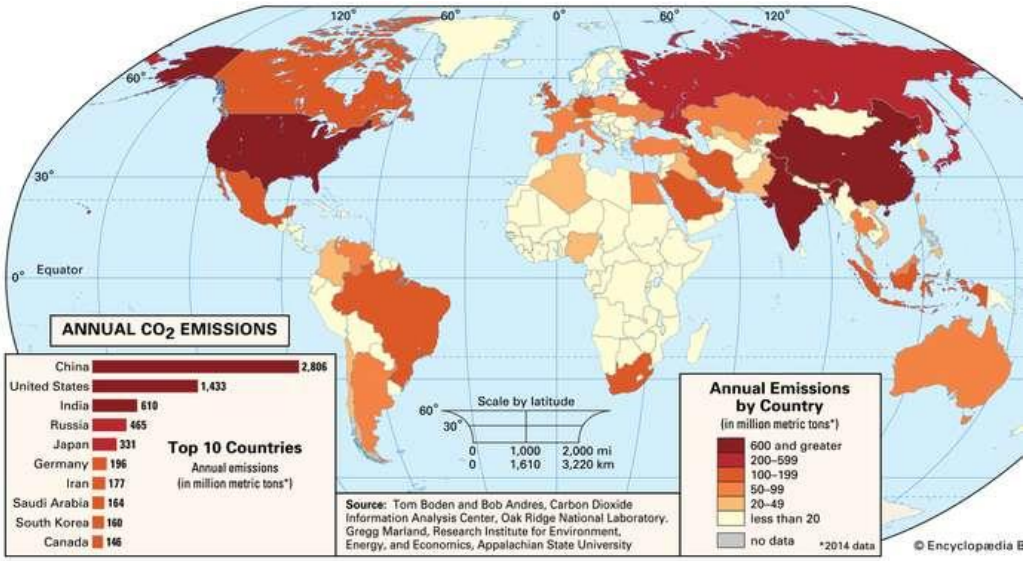


Fig 11. Annual CO2 Emissions

One of the most significant effects of air pollution is on climate change, particularly global warming. As a result of the growing worldwide consumption of fossil fuels, carbon dioxide levels in the atmosphere have increased steadily since 1900, and the rate of increase is accelerating. It has been estimated that if carbon dioxide levels are not reduced, average global air temperatures may rise another 4 °C (7.2 °F) by the end of the 21st century. Such a warming trend might cause melting of the polar ice caps, rising of the sea level, and flooding of the coastal areas of the world. Changes in precipitation patterns caused by global warming might have adverse effects on agriculture and forest ecosystems, and higher temperatures and humidity might increase the incidence of disease in humans and animals in some parts of the world. Implementation of international agreements on reducing greenhouse gases are required to protect global air quality and to mitigate the effects of global warming.

Health risks related to indoor air pollution have become an issue of concern because people generally spend most of their time indoors at home and at work. The problem has been exacerbated by well-meaning efforts to lower air-exchange rates in buildings in order to conserve energy; these efforts unfortunately allow contaminants to accumulate indoors. Indoor air pollutants include various combustion products from stoves, kerosene space heaters, and fireplaces, as well as volatile organic compounds (VOCs) from household products (e.g., paints, cleaning agents, and pesticides). Formaldehyde off-gassing from building products (especially particleboard and plywood) and from dry-cleaned textiles can accumulate in indoor air. Bacteria, viruses, molds, animal dander, dust mites, and pollen are biological contaminants that can cause disease and other health problems, especially if they

build up in and are spread by central heating or cooling systems. Environmental tobacco smoke, also called secondhand smoke, is an indoor air pollutant in many homes, despite widespread knowledge about the harmful effects of smoking. Second hand smoke contains many carcinogenic compounds as well as strong irritants. In some geographic regions, naturally occurring radon, a radioactive gas, can seep from the ground into buildings and accumulate to harmful levels. Exposure to all indoor air pollutants can be reduced by appropriate building construction and maintenance methods, limitations on pollutant sources, and provision of adequate ventilation.

5.3 Different Types of Air Quality Monitoring:

3.3.1 AAQMS : AAQMS is usually known as the Manual Air Quality Monitoring System. Under this system, the device samples the ambient air, and then after a few days of data collection, it is transferred manually to the centre where the data is analysed. The report is generated manually based on the analytics, and then finally, the data is archived to the server.

AAQMS generally includes a High Volume Sampler System or manual sampler which draws a known volume of air through the filter. After sample collection for the defined period, the sampler is taken manually to the laboratory for analysis. The filter is analysed for different pollutants based on its weight before and after sampling. Thus, it’s a process of sample collection, sample transfer to the lab, analysis, data recording, and post-consumption. So, the whole process may take up to 2-7 days to complete to get the pollution information of the location.

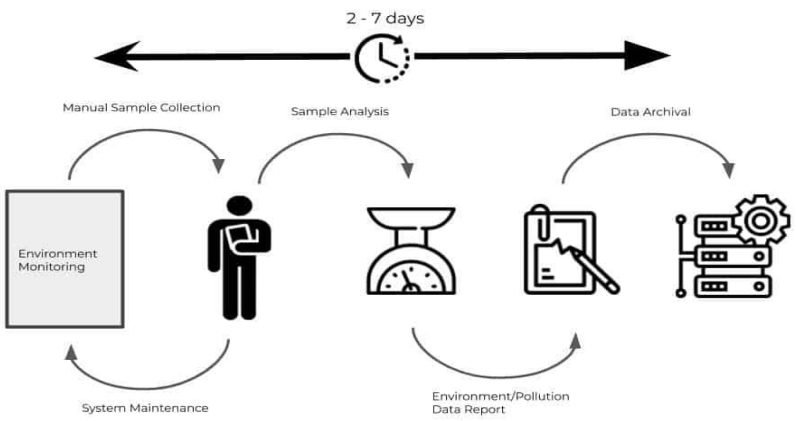


Fig 12. Manual Air Monitoring through AAQMS

3.3.2 CAAQMS : On the other hand, CAAQMS is the advanced version of AAQMS. It uses high-end technology like IoT for automated data collection and its transfer and analytics at the central server. The data transfer to the server is in real-time, and its interval is adjustable between 2-30 min. At the centre, the data is analysed automatically with advanced AI and archived or used accordingly.

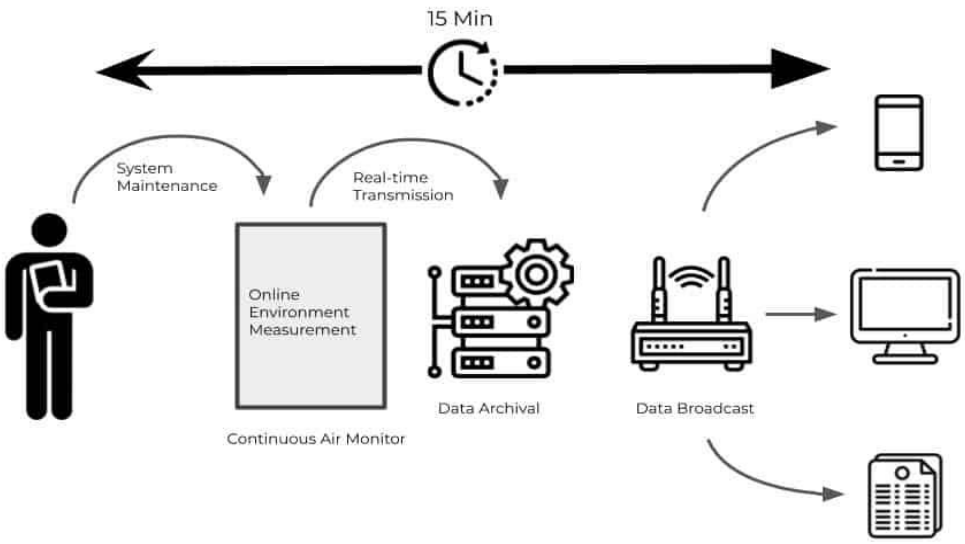


Fig 13. Automated Air Monitoring through CAAQMS

CAAQMS monitors pollutants using different analysers and working principles, thereby, which reduces the chances of manual error, generate data within minutes, and transmit the data. The data generated is disseminated online through a digital display board to the public. But there are two ways of data analysis for a CAAQMS:

a) **Sensor-based Systems** – The air monitoring system collects the sample and has sensors that operate on different working principles to measure different pollutants instantly. For instance, PM sensor works on Laser Scattering method, and Gas sensors work on NDIR, Electrochemical Sensing or Photoionization technique. Temperature, Pressure, and Humidity are measured using a solid-state semi-conductor sensing technique.

b) **Regulatory Monitoring by CPCBs and SPCBs** – The Central Pollution Control Board (CPCB) and State Pollution Control Board (SPCB) collect pollution data from various cities and collectively analyse at the centre. Traditional CAAQMS have dedicated analysers with different working principles for each pollutant. For example, SO2 is measured using fluorescence, NO2 using a chemiluminescent reaction, CO using infrared absorption, PM using filtration and gravimetry, etc.

3.3.3 Epidemic due to the current manual method : Since AAQMS is a manual monitor, it has certain limitations, and its data accuracy might be dubious. Hence, maintenance of instruments and evaluation of ambient air quality monitoring stations is crucial to ensure data quality, and for analytical quality control and following monitoring and calibration protocols.

Manual monitoring through AAQMS using different measuring techniques and instruments results in data variation in the same location. Studies show that due to incorrect flow measurement and calibration, the average data error ranges from 10-26% for PM monitoring. Inconsistent power supply and voltage fluctuations affect monitoring as well. For gaseous pollutants, duration of sampling, sample dilution, and temperature controls are essential to ensure proper testing. Moreover, the manual monitoring poses constraints for some of the gases, like, monitoring of benzene and ozone is very difficult using the manual method.

3.3.4 The difference that will change our lives : Imagine a time where citizens would not even know the air quality of their state or country. Now, with the continuous air quality monitoring systems and AQI dissemination to the public through display boards, people are more aware and cautious about the health impact of air pollution. Citizens and authority can initiate action to reduce air pollution with the environment data. Introduction of continuous air quality monitor has eased the environment monitoring through its various features.

a) **Scalable Real-time Monitoring** – Manual AAQMS is bulky and huge and therefore, its mass installation is not possible. On the other hand, sensor-based CAAQMS is compact and lightweight to allow for deployment en masse. This feature makes hyperspatial data possible to gather data from all corners of the region or city.
b) **Data Acquisition** – Manual monitoring results in delayed data collection, transmission, and availability. Whereas, real-time monitoring enables the environmental assessment as soon as the sample is collected. This bridges the time-lapse between data acquisition and analytics for speeding up the whole monitoring process.
c) **Measurement Features** – Currently, the existing AAQMS only monitor limited parameters, like PM, harmful gases like SO2, NO2, CO, CO2, etc. The advanced CAAQMS covers a broader range of parameters including PM, SOx, NOx, CO,

CO2, TVOCs, and weather parameters like temperature, humidity, pressure, wind speed & direction, light intensity, UV radiations, noise, rainfall, and floods.

d) **Decision-Making** – In AAQMS, data analysis may take up to several days, which delays the decision-making process. Faster data analytics through CAAQMS allows for on-time pollution monitoring for taking immediate measures to avoid critical situations in the future. It is an essential tool for better compliance enforcement through credible pollution monitoring and reporting practices.

e) **Analytics** – In AAQMS, data is analysed manually and may account for human error. Sensor-based CAAQMS incorporates automated data analytics supported by AI technology. It minimizes manual intervention to enhance data accuracy for strengthening the pollution control regime.

f) **Public Awareness** – One of the main purposes of ambient air quality monitoring is creating public awareness of environmental conditions. Delayed data acquisition through manual air monitors cannot provide real-time environmental data to the public. That's why automatic monitors are crucial for providing timely data. With real-time data analytics, the data can be served to the common people instantly through integrated Public displays, mobile app, web widget, alerts, etc.

While there is a strong reason for regulatory air quality monitoring to expand, its excessive cost and size create a problem. Thus, cost-effective sensor-based air quality monitoring system presents a feasible option for deploying air monitors for the general public. This also helps in democratizing the environment data even in the remote areas.

Automatic CAAQMS has made air quality monitoring simpler and provides more accurate data. A dense network of cost-effective sensor-based air quality monitors is the need of the hour. Advanced technologies in the area of air quality monitoring is becoming essential for the faster and real-time problem addressed on pollution mitigation plans.

CHAPTER 6 : **HARDWARES AND SOFTWARES**

6.1 Hardware Specifications

6.1.1 Microcontroller : It is an integrated circuit which functions as a small computer. A microcontroller contains a processor which has memory and programmable output/input peripherals. Its memory is in the form of ferroelectric RAM [11]. These are generally designed for embedded systems. Microcontroller is used for providing the sufficient platform for interfacing the sensors to display and communicate the data to the cloud. Basically, it performs as a minicomputer which follows the command provided to it. The command used for microcontroller is in the form of code.

6.1.2.1 Arduino Uno : The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.

The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes preprogrammed with a bootloader that

allows uploading new code to it without the use of an external hardware programmer.

While the Uno communicates using the original STK500 protocol,[1] it differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.[7]

6.1.2.2 Arduino Uno Technical Specifications :

a) Operating Voltage: 5 Volts
b) Input Voltage: 7 to 20 Volts
c) Digital I/O Pins: 14 (of which 6 can provide PWM output)
d) UART: 1
e) I2C: 1
f) SPPI: 1
g) Analog Input Pins: 6
h) DC Current per I/O Pin: 20 mA
i) DC Current for 3.3V Pin: 50 mA
j) Flash Memory: 32 KB of which 0.5 KB used by bootloader
k) SRAM: 2 KB
l) EEPROM: 1 KB
m) Clock Speed: 16 MHz
n) Length: 68.6 mm
o) Width: 53.4 mm
p) Weight: 25 g

6.1.2.3 Arduino Uno General Pin Functions :

a) **LED**: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it is off.
b) **VIN**: The input voltage to the Arduino/Genuino board when it is using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
c) **5V**: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V).

Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.

d) **3V3**: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

e) GND: Ground pins.

f) I**OREF**: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source, or enable voltage translators on the outputs to work with the 5V or 3.3V.

g) **Reset**: Typically used to add a reset button to shields that block the one on the board.

6.1.2.4 Arduino Uno Special Pin Functions :

a) **Serial / UART**: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.

b) **External interrupts**: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

c) **PWM (pulse-width modulation)**: pins 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the analogWrite() function.

d) **SPI (Serial Peripheral Interface)**: pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPI communication using the SPI library.

e) **TWI (two-wire interface) / I²C**: pin SDA (A4) and pin SCL (A5). Support TWI communication using the Wire library.

f) **AREF (analog reference)**: Reference voltage for the analog inputs.

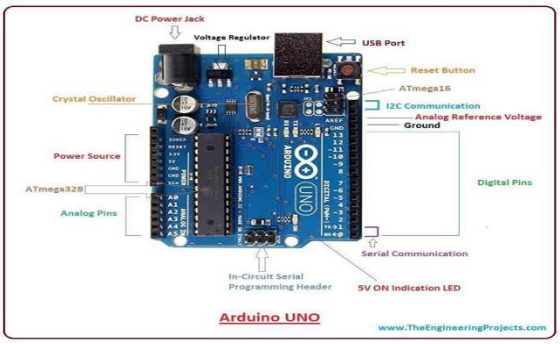


Fig 14. Arduino Uno

6.1.3.1 ESP8266 WIFI Module : The ESP8266 is a low-cost Wi-Fi microchip, with a full TCP/IP stack and microcontroller capability. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. The ESP8285 is an ESP8266 with 1 MiB of built-in flash, allowing the building of single-chip devices capable of connecting to Wi-Fi.

6.1.3.2 ESP8266 WIFI Module Features :

a) Processor: L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz
b) Memory:
   i. 32 KiB instruction RAM
   ii. 32 KiB instruction cache RAM
   iii. 80 KiB user-data RAM
   iv. 16 KiB ETS system-data RAM
c) External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
d) 17 GPIO pins
e) UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
f) 10-bit ADC (successive approximation ADC)

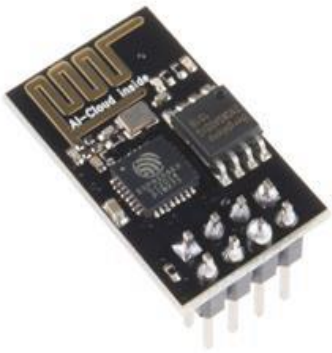


Fig 15. ESP8266 WIFI Module

6.1.4.1 16X2 LCD Display : LCD modules are very commonly used in most embedded projects, the reason being its cheap price, availability and programmer friendly. Most of us would have come across these displays in our day to day life, either at PCO's or calculators. The appearance and the pinouts have already been visualized above now let us get a bit technical.

16×2 LCD is named so because; it has 16 Columns and 2 Rows. There are a lot of combinations available like, 8×1, 8×2, 10×2, 16×1, etc. but the most used one is the 16×2 LCD. So, it will have (16×2=32) 32 characters in total and each character will be made of 5×8 Pixel Dots.

Each character has (5×8=40) 40 Pixels and for 32 Characters we will have (32×40) 1280 Pixels. Further, the LCD should also be instructed about the Position of the Pixels. Hence it will be a hectic task to handle everything with the help of MCU, hence an Interface IC like HD44780is used, which is mounted on the backside of the LCD Module itself. The function of this IC is to get the Commands and Data from the MCU and process them to display meaningful information onto our LCD Screen. You can learn how to interface an LCD using the above mentioned links. If you are an advanced programmer and would like to create your own library for interfacing your Microcontroller with this LCD module then you have to understand the HD44780 IC is working and commands which can be found its datasheet.

6.1.4.2 16X2 LCD Display Features :

a) Operating Voltage is 4.7V to 5.3V
b) Current consumption is 1mA without backlight.
c) Alphanumeric LCD display module, meaning can display alphabets and numbers
d) Consists of two rows and each row can print 16 characters.
e) Each character is build by a 5×8 pixel box
f) Can work on both 8-bit and 4-bit mode
g) It can also display any custom generated characters

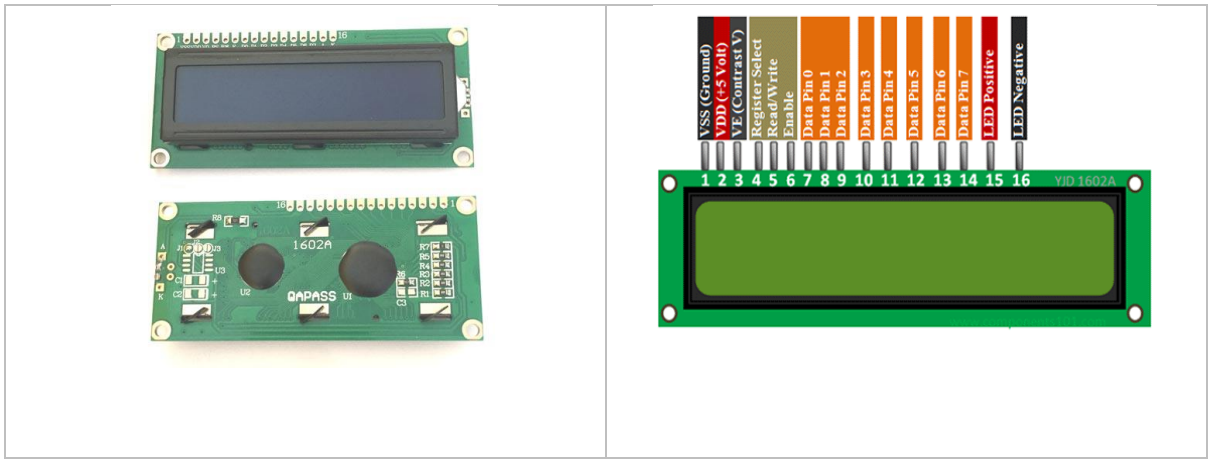


Fig 16. 16X2 LCD Display

6.1.5 Sensors :

6.1.5.1 MQ135 Gas Sensor : The MQ-135 gas sensor is available as a prewired module or as just the gas sensor part alone. If you're trying to detect the presence of a gas (not measuring PPM) only then you can buy a cheap module since it comes with onboard electronics to provide a digital output signal.

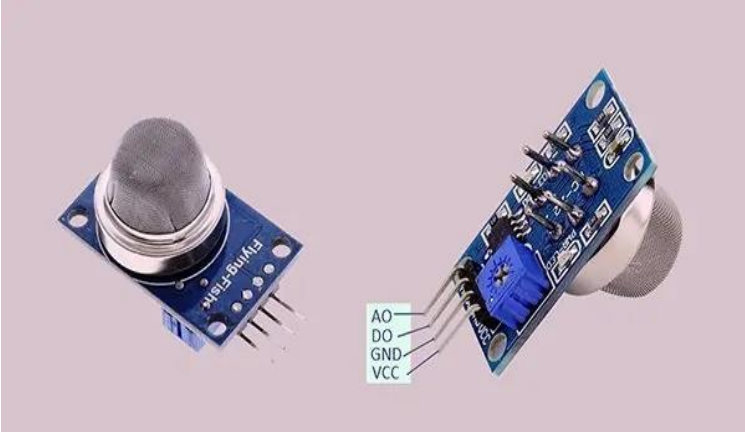


Fig 17. MQ135 Gas Sensor

The MQ-135 gas sensor module has four pins (standard male headers) as shown above. The air quality information is readable via the AO pin. Since AO is a varying voltage, this pin should be connected to one analog input of a microcontroller. The DO pin is an open-collector digital output (but with a pull-up resistor onboard) that becomes low when the detection level exceeds a predefined level. The detection threshold can be tailored through the small trimpot soldered on the bottom of the module.

Well, first of all power up the module alone with a regulated 5VDC power supply. The heater of the MQ-135 sensor requires 5V and have 33Ω±5% resistance, so your power supply must render a minimum 200mA of current

for the sensor part. Remember that at first the MQ-135 gas sensor have to be kept on continuously for its preheating time (over 24 hour) before you can actually play with it. Thereafter, introduce the MQ-135 sensor to the gas you want to detect and slowly adjust the trimpot until DO gets low.

Now every time your sensor gets introduced to that gas at predefined concentration, DO will go low (0V) else will remain high (5V). Note that the green LED (OUT_LED) should also light up when gas is observed. Also note that when the sensor is powered up for gas detection, it needs around 60-120 seconds to settle, because the heater will need that much time to heat the sensor up. Remember, the MQ-135 gas sensor is suitable for detecting (or measuring) of NH3, NOx, alcohol, Benzene, smoke, CO2, etc.

Fig 18. depicts the typical sensitivity characteristics of MQ135 gas sensor for several gases. That means the given plot tells us the concentration of a gas in part per million (ppm) according to the resistance ratio of the sensor (Rs/Ro). Here, Rs is the resistance of the sensor that changes depending on the concentration of gas, whereas Ro is the resistance of the sensor at a known concentration without the presence of other gases, or in the fresh air.

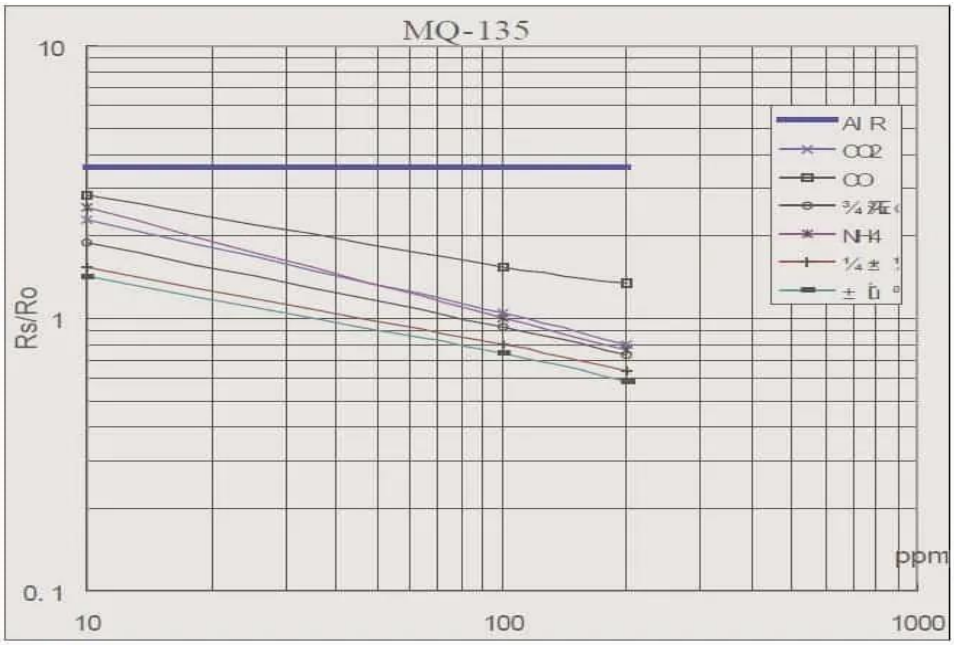


Fig 18. Sensitivity Characteristics of MQ135

In this sensitivity characteristic curve (borrowed from the datasheet), Ro is the sensor resistance at 100ppm of NH3 in clean air, and Rs is the sensor resistance at various concentrations of gases.

If you want to do anything serious with MQ-135 gas sensor, then you should learn to analyse this plot pretty well. However that needs some skill and patience as the scale is log-log i.e. in a linear scale, the behaviour of the gas concentration with respect to the resistance ratio is exponential (not linear). For determining Rs/Ro, you can use the fact that the Rs/Ro ratio is a constant in clean air. Also, by definition, only the Rs changes when a gas is present while Ro remains constant. Therefore, if you can determine the value of Rs in clean air, you can also determine Ro and ultimately Rs/Ro.

MQ-135 gas sensor calibration is done at first by finding the value of Ro in fresh air, and then using that value to find Rs through the formula:

Rs = (Vcc/VRL-1)* RL

that means, Rs = (5V/(sensorValue * (5.0/1023.0))-1)*RL. [i]

6.1.5.2 MQ2 Gas Sensor : The MQ-2 Gas sensor can detect or measure gasses like LPG, Alcohol, Propane, Hydrogen, CO and even methane. The module version of this sensor comes with a Digital Pin which makes this sensor to operate even without a microcontroller and that comes in handy when you are only trying to detect one particular gas. When it comes to measuring the gas in ppm the analog pin has to be used, the analog pin also TTL driven and works on 5V and hence can be used with most common microcontrollers.

Fig 19. MQ2

So if you are looking for a sensor to detect or measure gasses like LPG, Alcohol, Propane, Hydrogen, CO and even methane with or without a microcontroller then this sensor might be the right choice for you.

Using an MQ sensor it detects a gas is very easy. You can either use the digital pin or the analog pin to accomplish this. Simply power the module with 5V and you should notice the power LED on the module to glow and when no gas it detected the output LED will remain turned off meaning the digital output pin will be 0V. Remember that these sensors have to be kept on for pre-heating time (mentioned in features above) before you can actually work with it. Now, introduce the sensor to the gas you want to detect and you should see the output LED to go high along with the digital pin, if not use the potentiometer until the output gets high. Now every time your sensor gets introduced to this gas at this particular concentration the digital pin will go high (5V) else will remain low (0V).

You can also use the analog pin to achieve the same thing. Read the analog values (0-5V) using a microcontroller, this value will be directly proportional to the concentration of the gas to which the sensor detects. You can experiment with these values and check how the sensor reacts to different concentration of gas and develop your program accordingly.

for some accuracy with your readings then measuring the PPM would be the best way to go with it. It can also help you to distinguish one gas from another. So, to measure PPM you can directly use a module. A basic wiring for the sensor from datasheet is shown below.

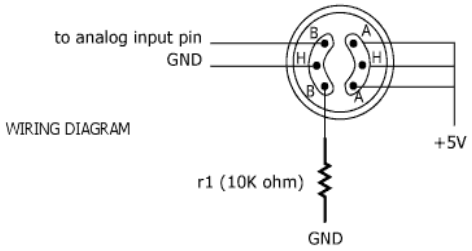


Fig 20. Wiring Diagram MQ2

The procedure to measure PPM using MQ sensor is the same but few constant values will vary based on the type of MQ sensor used. Basically, we need to look into the (Rs/Ro) VS PPM graph given in the datasheet (also shown below).

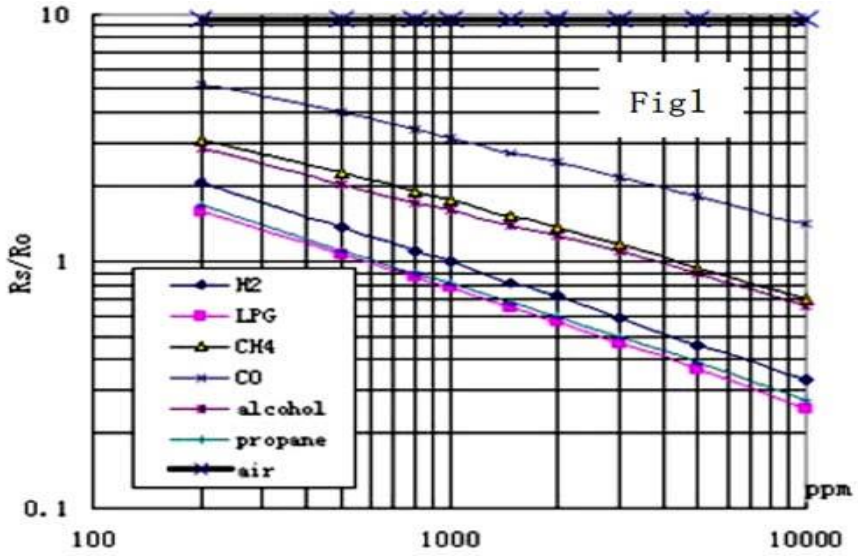


Fig 21. Sensitivity Characteristics of MQ2

The value of Ro is the value of resistance in fresh air and the value of Rs is the value of resistance in Gas concentration. First, you should calibrate the sensor by finding the values of Ro in fresh air and then use that value to find Rs using the formula:

Resistance of Sensor(Rs) : Rs=(Vc/VRL-1)*RL. [ii]

Once we calculate Rs and Ro we can find the ratio and then use the graph shown above we can calculate the equivalent value of PPM for that particular gas.

6.1.5.3 DHT11 : The DHT11 is a commonly used Temperature and humidity sensor. The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers.

The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. So if you are looking to measure in this range then this sensor might be the right choice for you.

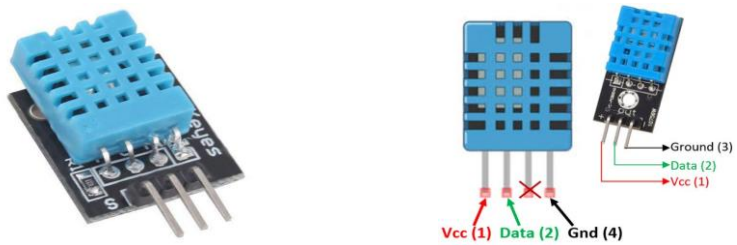


Fig 22. DHT11

The DHT11 Sensor is factory calibrated and outputs serial data and hence it is highly easy to set it up. The connection diagram for this sensor is shown below.

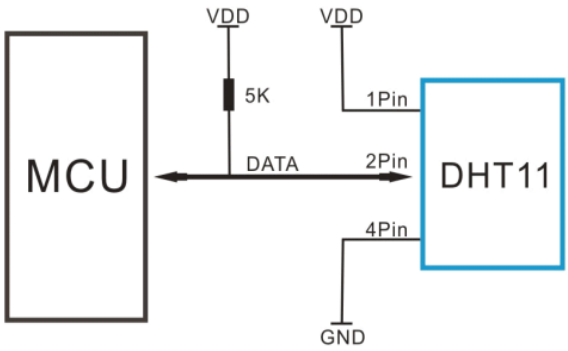


Fig 23. Calibrated Connection.

As you can see the data pin is connected to an I/O pin of the MCU and a 5K pull-up resistor is used. This data pin outputs the value of both temperature and humidity as serial data. If you are trying to interface DHT11 with Arduino then there are ready-made libraries for it which will give you a quick start.

6.2 Software's :

6.2.1 Arduino Ide : The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards.

The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures.

User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.[8] By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.

Arduino IDE is a derivative of the Processing IDE, however as of version 2.0, the Processing IDE will be replaced with the Visual Studio Code-based Eclipse Theia IDE framework.

With the rising popularity of Arduino as a software platform, other vendors started to implement custom open source compilers and tools (cores) that can build and upload sketches to other microcontrollers that are not supported by Arduino's official line of microcontrollers.

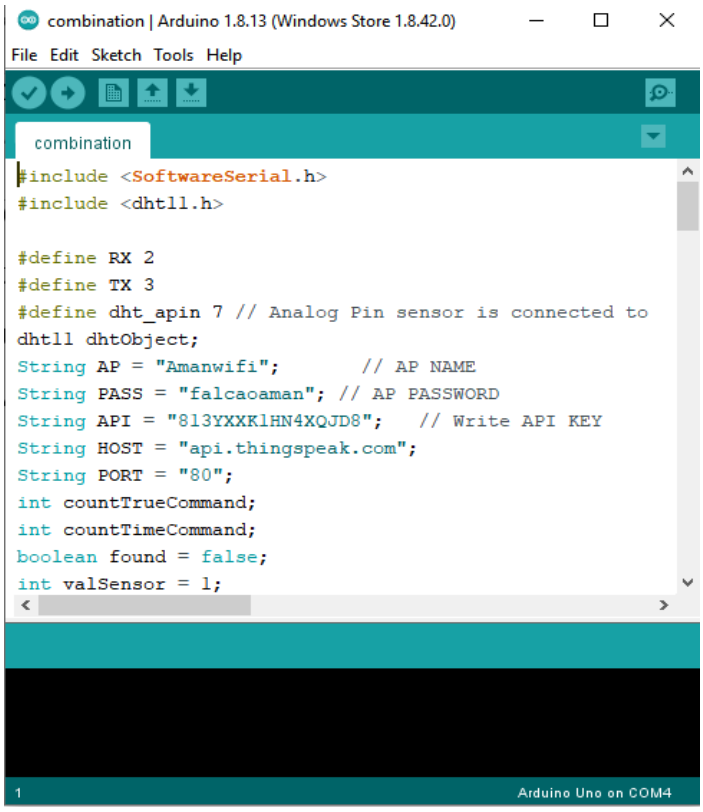


Fig 24. Arduino Ide

6.2.2 ThingSpeak : ThingSpeak is an IOT analytics platform service that allows to aggregate, visualize and analyse live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by the IOT devices to ThingSpeak server. With the ability to execute MATLAB code in ThingSpeak one can perform online analysis and processing of the data as it comes in.

Fig 25. ThingSpeak User Interface

6.2.3 Jupyter Notebook : Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell. To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user. Jupyter Notebook can connect to many kernels to allow

programming in different languages. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), it was 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to The Atlantic, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

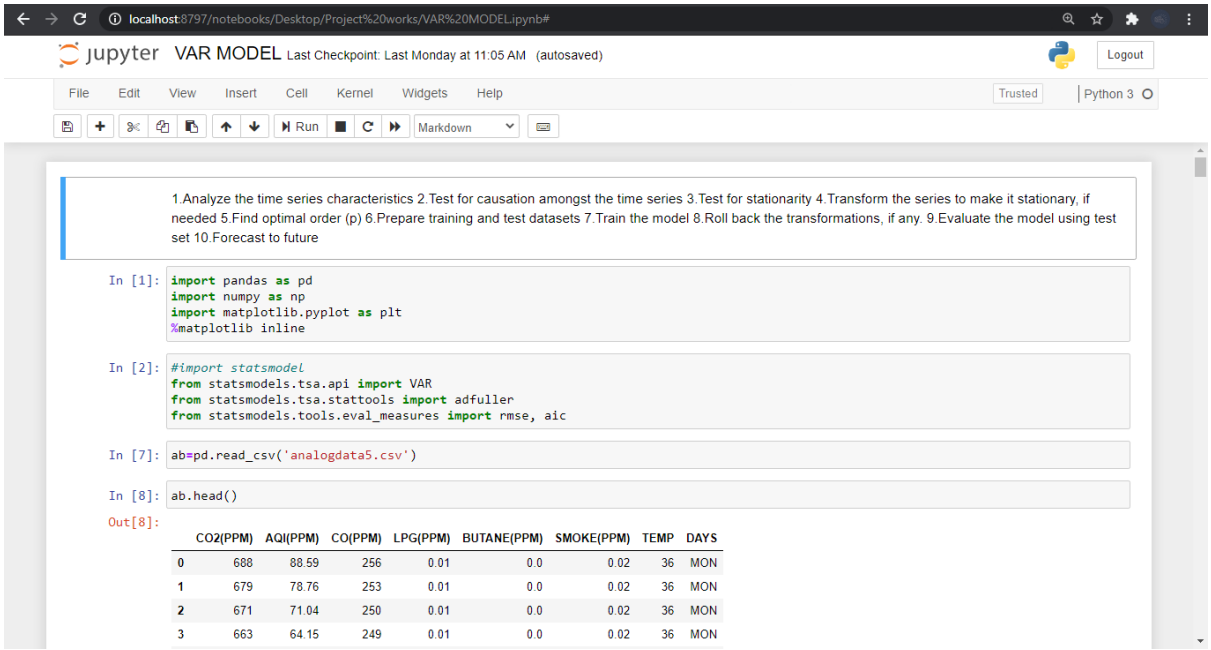


Fig 26. Jupyter Notebook User Interface

6.2.4 Fritzing : Fritzing is an open-source initiative to develop amateur or hobby CAD software for the design of electronics hardware, to support designers and artists ready to move from experimenting with a prototype to building a more permanent circuit.

The software is created in the spirit of the Processing programming language and the Arduino microcontroller and allows a designer, artist, researcher, or hobbyist to document their Arduino-based prototype and create a PCB layout for

manufacturing. The associated website helps users share and discuss drafts and experiences as well as to reduce manufacturing costs.

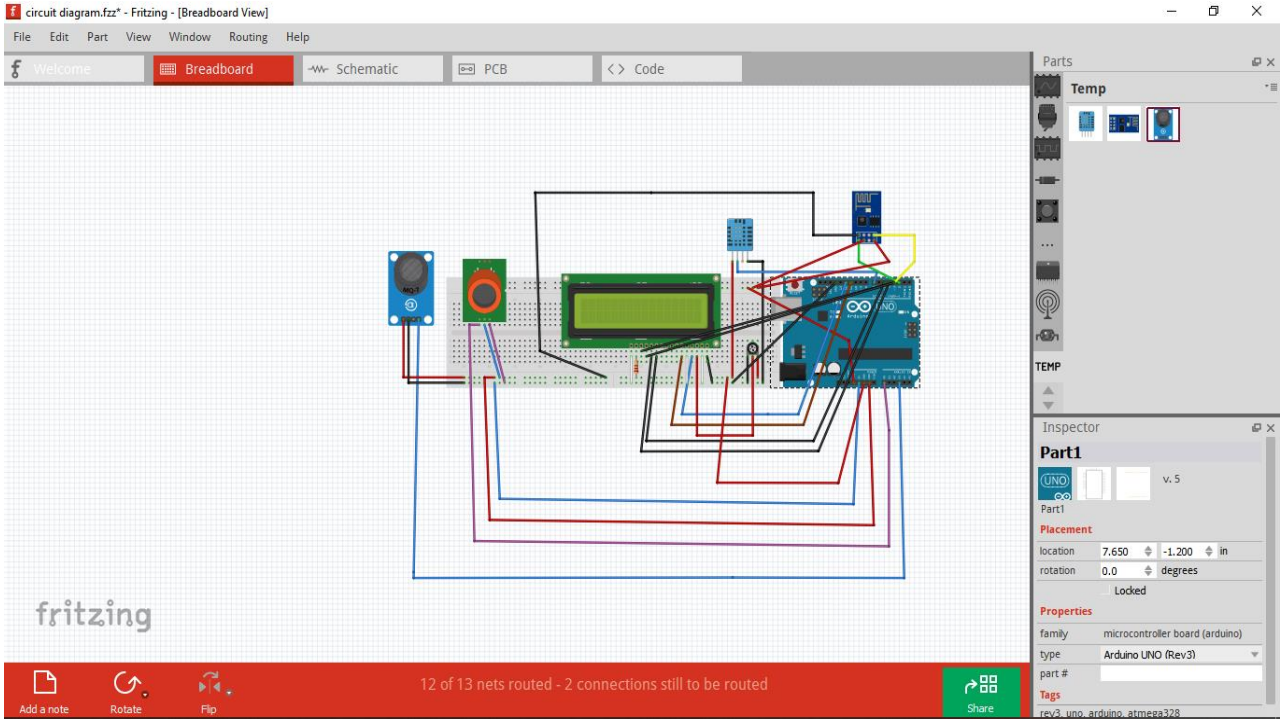


Fig 27. Fritzing User Interface

CHAPTER 7 : **WORKING PRINCIPLE AND RESULTS**

7.1 Working Principle : The device developed in this project can be installed near any Wi-Fi hotspot in a populated urban area. As the device is powered, the Arduino board loads the required libraries, flashes some initial messages on the LCD screen and start sensing data from the MQ-135 sensor, MQ-2 sensor and DHT11. The sensitivity curve of the sensor for different combustible gases is already mentioned in Section 4.2.3. The sensor is calibrated according to the formula [i] so that its analog output voltage is proportional to the concentration of polluting gases in PPM. The analog voltage sensed at the pin A0 of the Arduino is converted to a digital value by using the in-built ADC channel of the Arduino.

Upload

calibrated_mq135

```
int gas_sensor = A0; //Sensor pin
float m = -0.3376; //Slope
float b = 0.7165; //Y-Intercept
float R0 = 4.25; //Sensor Resistance in fresh air from previous code

void setup() {
  Serial.begin(9600); //Baud rate

  pinMode(gas_sensor, INPUT); //Set gas sensor as input
}

void loop() {

  float sensor_volt; //Define variable for sensor voltage
  float RS_gas; //Define variable for sensor resistance
  float ratio; //Define variable for ratio
  int sensorValue = analogRead(gas_sensor); //Read analog values of sensor
  Serial.print("SENSOR RAW VALUE = ");
  Serial.println(sensorValue);
  sensor_volt = sensorValue*(5.0/1023.0); //Convert analog values to voltage
  Serial.print("Sensor value in volts = ");
  Serial.println(sensor_volt);
  RS_gas = ((5.0*10.0)/sensor_volt)-10.0; //Get value of RS in a gas
  Serial.print("Rs value = ");
  Serial.println(RS_gas);
  ratio = RS_gas/R0;  // Get ratio RS_gas/RS_air

  Serial.print("Ratio = ");
  Serial.println(ratio);
  float ppm_log = (log10(ratio)-b)/m; //Get ppm value in linear scale according to the the ratio value
  float ppm = pow(10, ppm_log); //Convert ppm value to log scale
  Serial.print("Our desired PPM = ");
```

MQ135_1

```
void setup() {
  Serial.begin(9600); //Baud rate
  pinMode(A0,INPUT);
}

void loop() {
  float sensor_volt; //Define variable for sensor voltage
  float RS_air; //Define variable for sensor resistance
  float R0; //Define variable for R0
  float sensorValue=0.0; //Define variable for analog readings
  Serial.print("Sensor Reading = ");
  Serial.println(analogRead(A0));

  for(int x = 0 ; x < 500 ; x++) //Start for loop
  {
    sensorValue = sensorValue + analogRead(A0); //Add analog values of sensor 500 times
  }
  sensorValue = sensorValue/500.0; //Take average of readings
  Serial.print("Average = ");
  Serial.println(sensorValue);
  sensor_volt = sensorValue*(5.0/1023.0); //Convert average to voltage
  RS_air = ((5.0*10.0)/sensor_volt)-10.0; //Calculate RS in fresh air
  R0 = RS_air/3.7; //Calculate R0

  Serial.print("R0 = "); //Display "R0"
  Serial.println(R0); //Display value of R0
  delay(1000); //Wait 1 second

}
```

Fig 28. Calibration of MQ-135

The Arduino board has 10-bit ADC channels, so the digitized value ranges from 0 to 1023. The digitized value can be assumed proportional to the concentration of gases in PPM. The read value is first displayed on LCD screen and passed to the ESP8266 module wrapped in proper string through virtual serial function. The Wi-Fi module is configured to connect with the ThingSpeak IOT platform. ThingSpeak is an IOT analytics platform service that allows to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by the IOT devices to ThingSpeak server. With the ability to execute MATLAB code in ThingSpeak one can perform online analysis and processing of the data as it comes in.

The Wi-Fi module can be connected with the ThingSpeak server by sending AT commands from the module. The module first test the AT startup by sending the following command –

AT

The command is passed by the controller to the Wi-Fi module using software serial function. In response to the command 'AT', the platform must respond with 'OK' if the cloud service is running. Then, the AT command to view the version information is passed as follow –

AT + GMR

In response to this command, the IOT platform must respond by sending back the version information, sdk version and the time bin is compiled. Next, the AT command to set the connection to Wi-Fi mode is sent as follow –

AT + CWMODE = 3

By setting the parameter in CWMODE to 3, the Wi-Fi connection is configured to SoftAP as well as station mode. This AT command can in fact take three parameters as follow –

1 – set Wi-Fi connection to station mode
2 – set Wi-Fi connection to SoftAP mode
3 – set Wi-Fi connection to SoftAP + station mode

In response to this command, the IOT platform must send back the string indication the Wi-Fi connection mode set. Now the AT command to reset the module is sent as follow –
AT + RST
In response to this command, the Wi-Fi module must restart and send back a response of 'OK'. After resetting the module, AT command to setup multiple connections is enabled by sending the following command –

AT + CIPMUX=1

This AT command can take two parameters – 0 for setting single connection and 1 for setting multiple connections. Next, the command to connect with the Access Point (AP) is passed

which takes two parameters where first parameter is the SSID of the registered cloud service on ThingSpeak and the other parameter is the password to login the cloud service.

AT+CWJAP=”Amanwifi”,”egP@$$w0rd?
Now, the AT command to get local IP address is passed as follow –

AT + CIFSR

In response to this command, the local IP address of the Wi-Fi connection is sent back by the module. Now, the module is ready to establish TCP IP connection with the ThingSpeak server. The controller reads the sensor data and store it in a string variable. The TCP IP connection is established by sending the following AT command –

AT + CIPSTART = 4, “TCP”, “184.106.153.149”, 80

The AT + CIPSTART command can be used to establish a TCP connection, register an UDP port or establish an SSL connection. In the above command, it is used to establish a TCP IP connection. For establishing a TCP-IP connection, the command takes four parameters where first parameter is link ID which can be a number between 0 to 4, second parameter is connection type which can be TCP or UDP, third parameter is remote IP address or IP address of the cloud service to connect with and last parameter is detection time interval for checking if the connection is live. If the last parameter is set to 0, the TCP keep-alive feature is disabled otherwise a time interval in seconds range from 1 to 7200 can be passed as parameter. In response to this command, the server must respond with ‘OK’ if connection is successfully established otherwise it should respond with message ‘ERROR’.

Now when the connection with the server is successfully established and the controller has read the sensor value, it can send the data to the cloud using the following command –

AT + CIPSEND = 4

This command takes four parameters, where first parameter is the link ID which can be a number between 0 to 4, second parameter is data length which can be maximum 2048 bytes long, third parameter is remote IP in case of an UDP connection and remote port number in case of UDP connection. The third and fourth parameter are optional and used only in case of UDP connection with the server. Since, the TCP IP connection is established, these parameters

are not used. The command is followed by a string containing the URL having the field names and values passed through the HTTP GET method. In this project, a string containing the URL having API Key and the sensor value as the field and value is passed. The passed field and its value are logged on the cloud server. It is important to pass the API key in this URL as one of the field-value in order to connect with the registered cloud service. The Air quality measured by sensor can now be monitored and recorded through the thingspeak IOT plat form through the Wi-Fi module.

Now Coming to the datalogger , the data that was collected from the sensors were directly downloaded as the csv file using Python codes using Serial Library .

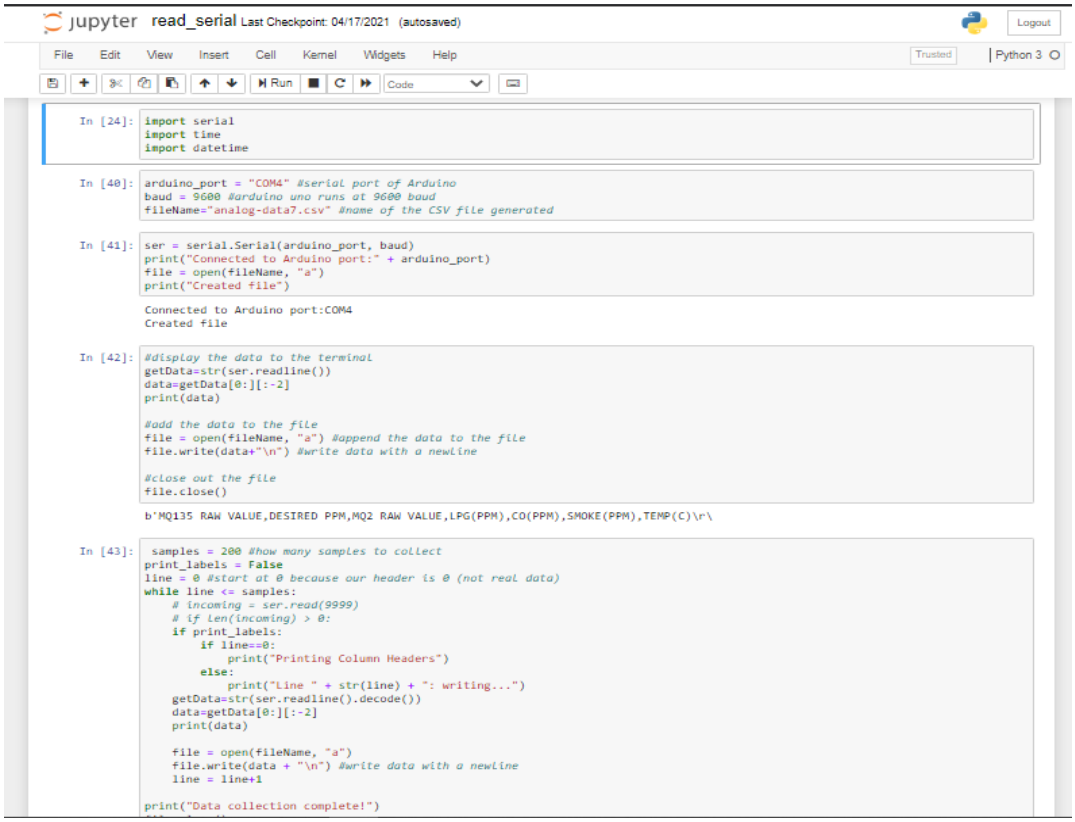


Fig 29. Datalogger codes.

Finally, we trained our model in Jupyter Notebook using VAR Model Formula. Autoregression models, the time series is modeled as a linear combination of it's own lags. That is, the past values of the series are used to forecast the current and future.

A typical AR(p) model equation looks something like this:

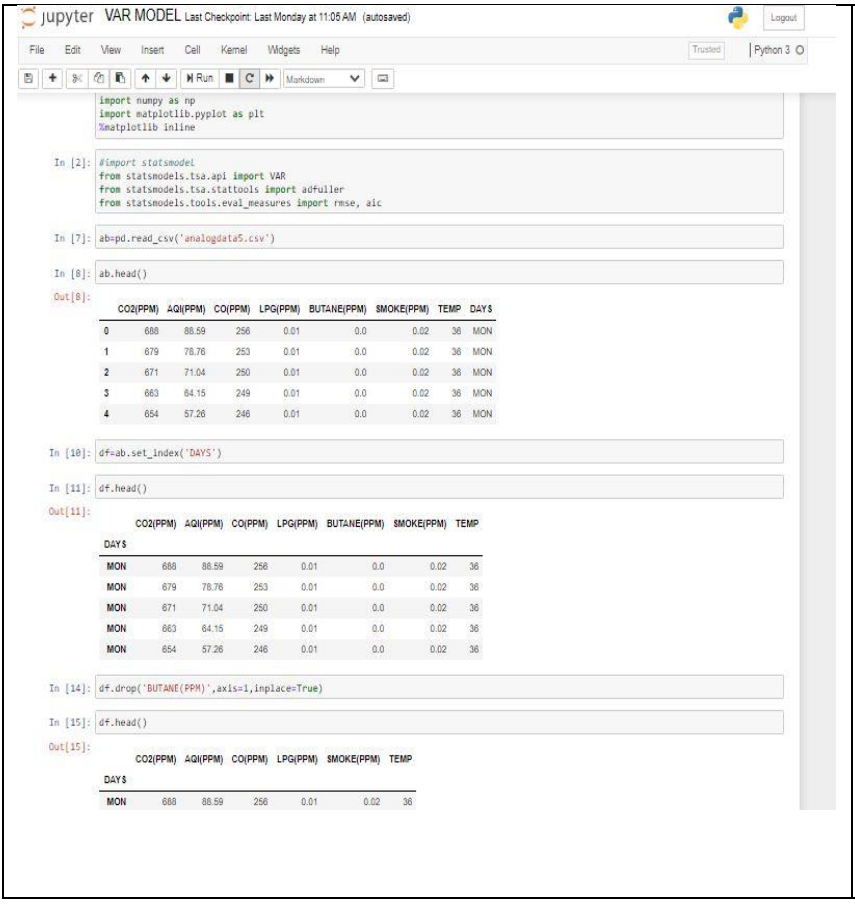
$$y_t = \alpha + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_p y_{t-p} + \epsilon_t$$

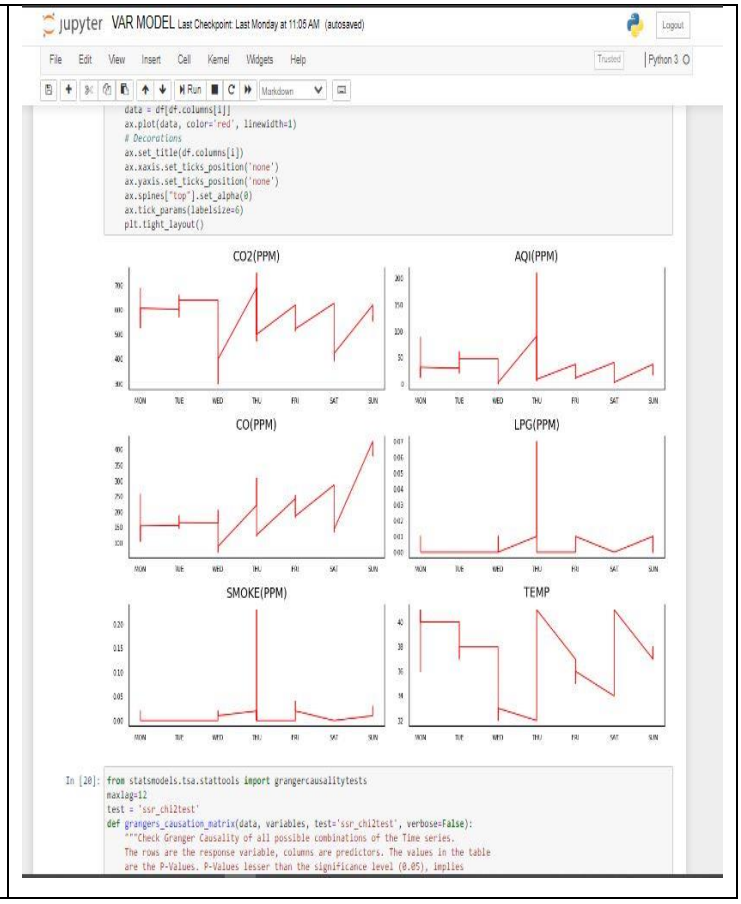
where α is the intercept, a constant and β1, β2 till βp are the coefficients of the lags of Y till order p.

Order ‘p’ means, up to p-lags of Y is used and they are the predictors in the equation. The ε_{t} is the error, which is considered as white noise.

Steps to build a VAR Model :

a) Analyze the time series characteristics
b) Test for causation amongst the time series
c) Test for stationarity
d) Transform the series to make it stationary, if needed
e) Find optimal order (p)
f) Prepare training and test datasets
g) Train the model
h) Roll back the transformations, if any.
i) Evaluate the model using test set
j) Forecast to future





```
    zero, that is, the X does not cause Y can be rejected.

    data      : pandas dataframe containing the time series variables
    variables : list containing names of the time series variables.
    """
    df = pd.DataFrame(np.zeros((len(variables), len(variables))), columns=variables, index=variables)
    for c in df.columns:
        for r in df.index:
            test_result = grangercausalitytests(data[[r, c]], maxlag=maxlag, verbose=False)
            p_values = [round(test_result[i+1][0][[test][1],4) for i in range(maxlag)]
            if verbose: print(f'Y = {r}, X = {c}, P Values = {p_values}')
            min_p_value = np.min(p_values)
            df.loc[r, c] = min_p_value
    df.columns = [var + '_x' for var in variables]
    df.index = [var + '_y' for var in variables]
    return df

grangers_causation_matrix(df, variables = df.columns)
```

```
C:\Users\HP\anaconda3\lib\site-packages\statsmodels\base\model.py:1832: ValueWarning: covariance of constraints does not have f
ull rank. The number of constraints is 9, but rank is 2
  'rank is %d' % (J, J_), ValueWarning)
C:\Users\HP\anaconda3\lib\site-packages\statsmodels\base\model.py:1832: ValueWarning: covariance of constraints does not have f
ull rank. The number of constraints is 12, but rank is 1
  'rank is %d' % (J, J_), ValueWarning)
```

Out[20]:

	CO2(PPM)_x	AQI(PPM)_x	CO(PPM)_x	LPG(PPM)_x	SMOKE(PPM)_x	TEMP_x
CO2(PPM)_y	1.0000	0.0006	0.0008	0.1004	0.000	0.2782
AQI(PPM)_y	0.0000	1.0000	0.0000	0.0000	0.000	0.1572
CO(PPM)_y	0.3565	0.0650	1.0000	0.1862	0.000	0.1291
LPG(PPM)_y	0.0329	0.0000	0.0000	1.0000	0.000	0.0716
SMOKE(PPM)_y	0.0640	0.0000	0.0001	0.0000	1.000	0.0016
TEMP_y	0.0757	0.1744	0.0000	0.1198	0.091	1.0000

In [21]:
```
from statsmodels.tsa.vector_ar.vecm import coint_johansen

def cointegration_test(df, alpha=0.05):
    """Perform Johanson's Cointegration Test and Report Summary"""
    out = coint_johansen(df,-1,5)
    d = {'0.90':0, '0.95':1, '0.99':2}
    traces = out.lr1
    cvts = out.cvt[:, d[str(1-alpha)]]
    def adjust(val, length= 6): return str(val).ljust(length)

    # Summary
    print('Name   ::  Test Stat > C(95%)    =>   Signif  \n', '--'*20)
    for col, trace, cvt in zip(df.columns, traces, cvts):
        print(adjust(col), ':: ', adjust(round(trace,2), 9), ">", adjust(cvt, 8), ' =>  ' , trace > cvt)
```

```
(4, 6)
```

In [23]:
```
def adfuller_test(series, signif=0.05, name='', verbose=False):
    """Perform ADFuller to test for Stationarity of given series and print report"""
    r = adfuller(series, autolag='AIC')
    output = {'test_statistic':round(r[0], 4), 'pvalue':round(r[1], 4), 'n_lags':round(r[2], 4), 'n_obs':r[3]}
    p_value = output['pvalue']
    def adjust(val, length= 6): return str(val).ljust(length)

    # Print Summary
    print(f'    Augmented Dickey-Fuller Test on "{name}"', "\n   ", '-'*47)
    print(f' Null Hypothesis: Data has unit root. Non-Stationary.')
    print(f' Significance Level    = {signif}')
    print(f' Test Statistic        = {output["test_statistic"]}')
    print(f' No. Lags Chosen       = {output["n_lags"]}')

    for key,val in r[4].items():
        print(f' Critical value {adjust(key)} = {round(val, 3)}')

    if p_value <= signif:
        print(f" => P-Value = {p_value}. Rejecting Null Hypothesis.")
        print(f" => Series is Stationary.")
    else:
        print(f" => P-Value = {p_value}. Weak evidence to reject the Null Hypothesis.")
        print(f" => Series is Non-Stationary.")
```

In [24]:
```
# ADF Test on each column
for name, column in df_train.iteritems():
    adfuller_test(column, name=column.name)
    print('\n')
```

```
    Augmented Dickey-Fuller Test on "CO2(PPM)"
   -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    = 0.05
 Test Statistic        = -3.1337
 No. Lags Chosen       = 1
 Critical value 1%     = -3.432
 Critical value 5%     = -2.862
 Critical value 10%    = -2.567
 => P-Value = 0.0242. Rejecting Null Hypothesis.
 => Series is Stationary.


    Augmented Dickey-Fuller Test on "AQI(PPM)"
   -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    = 0.05
 Test Statistic        = -4.871
 No. Lags Chosen       = 9
 Critical value 1%     = -3.432
 Critical value 5%     = -2.862
```

```
    Augmented Dickey-Fuller Test on "CO(PPM)"
   -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    = 0.05
 Test Statistic        = -61.3673
 No. Lags Chosen       = 0
 Critical value 1%     = -3.432
 Critical value 5%     = -2.862
 Critical value 10%    = -2.567
 => P-Value = 0.0. Rejecting Null Hypothesis.
 => Series is Stationary.


    Augmented Dickey-Fuller Test on "LPG(PPM)"
   -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    = 0.05
 Test Statistic        = -16.6972
 No. Lags Chosen       = 22
 Critical value 1%     = -3.432
 Critical value 5%     = -2.862
 Critical value 10%    = -2.567
 => P-Value = 0.0. Rejecting Null Hypothesis.
 => Series is Stationary.


    Augmented Dickey-Fuller Test on "SMOKE(PPM)"
   -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    = 0.05
 Test Statistic        = -16.8525
 No. Lags Chosen       = 24
 Critical value 1%     = -3.432
 Critical value 5%     = -2.862
 Critical value 10%    = -2.567
 => P-Value = 0.0. Rejecting Null Hypothesis.
 => Series is Stationary.


    Augmented Dickey-Fuller Test on "TEMP"
   -----------------------------------------------
 Null Hypothesis: Data has unit root. Non-Stationary.
 Significance Level    = 0.05
 Test Statistic        = -22.7435
 No. Lags Chosen       = 7
 Critical value 1%     = -3.432
 Critical value 5%     = -2.862
 Critical value 10%    = -2.567
 => P-Value = 0.0. Rejecting Null Hypothesis.
 => Series is Stationary.
```

```
FPE :  3.757192914849944e-10
HQIC:  -21.5162592598744
```

In [29]:
```
model_fitted = model.fit(3)
model_fitted.summary()
```

```
Log likelihood:          9113.08    FPE:                 4.28339e-10
AIC:                    -21.5711    Det(Omega_mle):      4.16216e-10
--------------------------------------------------------------------------------
Results for equation CO2(PPM)
==================================================================================
                     coefficient       std. error           t-stat          prob
----------------------------------------------------------------------------------
const                  -0.034327         0.141955           -0.242         0.809
L1.CO2(PPM)             0.041153         0.036916            1.115         0.265
L1.AQI(PPM)            -0.496988         0.108391           -4.585         0.000
L1.CO(PPM)              0.101756         0.037225            2.734         0.006
L1.LPG(PPM)          -594.185216       201.400777           -2.950         0.003
L1.SMOKE(PPM)         620.633456        79.454524            7.811         0.000
L1.TEMP                 0.977328         0.776420            1.259         0.208
L2.CO2(PPM)            -0.008381         0.037214           -0.225         0.822
L2.AQI(PPM)            -0.091488         0.110107           -0.831         0.406
L2.CO(PPM)              0.019770         0.037499            0.527         0.598
L2.LPG(PPM)          -190.918247       206.571467           -0.924         0.355
L2.SMOKE(PPM)         236.786559        91.950764            2.575         0.010
L2.TEMP                -0.259702         0.806088           -0.322         0.747
```

In [42]:
```
from statsmodels.stats.stattools import durbin_watson
out = durbin_watson(model_fitted.resid)

for col, val in zip(df.columns, out):
    print(col, ':', round(val, 2))
```

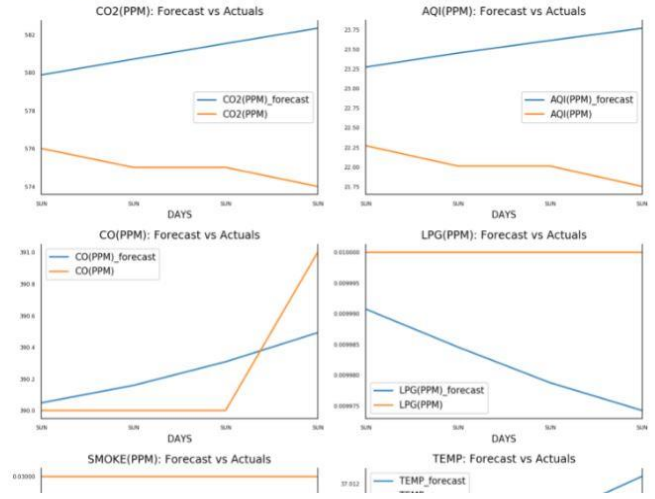
```
CO2(PPM) : 2.0
AQI(PPM) : 2.0
CO(PPM) : 2.0
LPG(PPM) : 2.02
SMOKE(PPM) : 2.02
TEMP : 1.99
```

In [43]:
```
# Get the lag order
lag_order = model_fitted.k_ar
print(lag_order)  #> 4

# Input data for forecasting
forecast_input = df_differenced.values[-lag_order:]
forecast_input
```

```
3
```

Out[43]:
```
array([[0.  , 0.  , 0.  , 0.  , 0.  , 0.  ],
       [0.  , 0.  , 0.  , 0.  , 0.  , 0.  ],
```

```
    df_results[col+'_forecast'].plot(legend=True, ax=ax).autoscale(axis='x',tight=True)
    df_test[col][-nobs:].plot(legend=True, ax=ax);
    ax.set_title(col + ": Forecast vs Actuals")
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.spines["top"].set_alpha(0)
    ax.tick_params(labelsize=6)

plt.tight_layout();
```




```
    mape = np.mean(np.abs(forecast - actual)/np.abs(actual))  # MAPE
    me = np.mean(forecast - actual)             # ME
    mae = np.mean(np.abs(forecast - actual))    # MAE
    mpe = np.mean((forecast - actual)/actual)   # MPE
    rmse = np.mean((forecast - actual)**2)**.5  # RMSE
    corr = np.corrcoef(forecast, actual)[0,1]   # corr
    mins = np.amin(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    maxs = np.amax(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    minmax = 1 - np.mean(mins/maxs)             # minmax
    return({ 'mape':mape, 'me':me, 'mae': mae,
            'mpe': mpe,'rmse':rmse, 'corr':corr})

print('Forecast Accuracy of: CO2(PPM)')
accuracy_prod = forecast_accuracy(df_results['CO2(PPM)_forecast'].values, df_test['CO2(PPM)'])
for k, v in accuracy_prod.items():
    print(k, ': ', round(v,4))

print('\nForecast Accuracy of: AQI(PPM)')
accuracy_prod = forecast_accuracy(df_results['AQI(PPM)_forecast'].values, df_test['AQI(PPM)'])
for k, v in accuracy_prod.items():
    print(k, ': ', round(v,4))

print('\nForecast Accuracy of: CO(PPM)')
accuracy_prod = forecast_accuracy(df_results['CO(PPM)_forecast'].values, df_test['CO(PPM)'])
for k, v in accuracy_prod.items():
    print(k, ': ', round(v,4))

print('\nForecast Accuracy of: LPG(PPM)')
accuracy_prod = forecast_accuracy(df_results['LPG(PPM)_forecast'].values, df_test['LPG(PPM)'])
for k, v in accuracy_prod.items():
    print(k, ': ', round(v,4))

print('\nForecast Accuracy of: SMOKE(PPM)')
accuracy_prod = forecast_accuracy(df_results['SMOKE(PPM)_forecast'].values, df_test['SMOKE(PPM)'])
for k, v in accuracy_prod.items():
    print(k, ': ', round(v,4))

print('\nForecast Accuracy of: TEMP')
accuracy_prod = forecast_accuracy(df_results['TEMP_forecast'].values, df_test['TEMP'])
for k, v in accuracy_prod.items():
    print(k, ': ', round(v,4))
```

```
Forecast Accuracy of: CO2(PPM)
mape :  0.0106
me :  6.1084
mae :  6.1084
mpe :  0.0106
rmse :  6.3151
corr :  -0.9483
```

Fig 30. VAR Model Codes

7.2 Circuit Diagram:

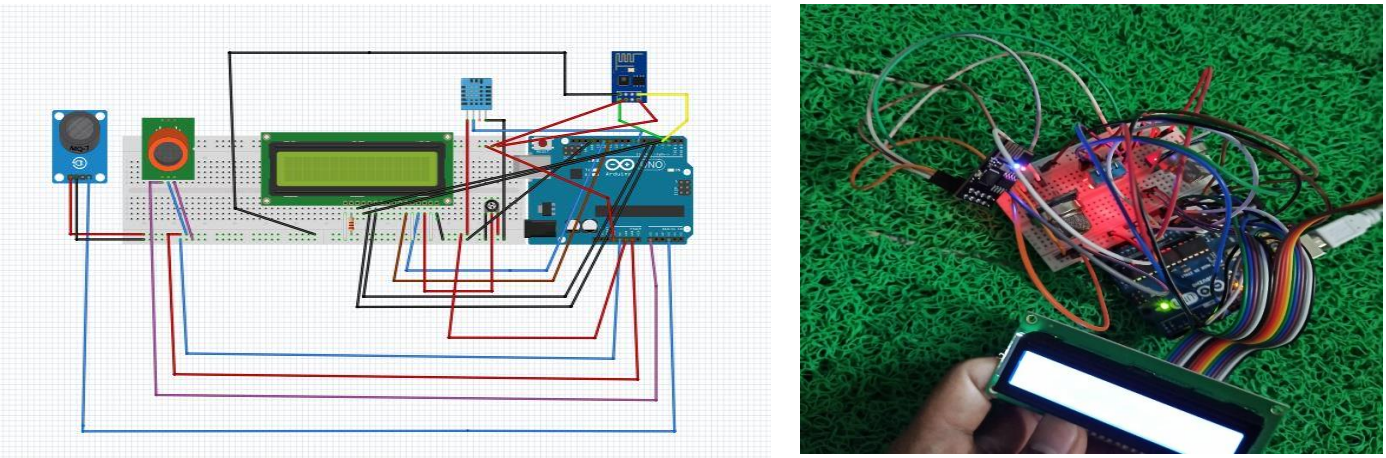


Fig 31. Fritzing Circuit Diagram and Real Circuit Diagram

7.3 Results:

7.3.1 ThingSpeak Readings :

Fig 32. ThingSpeak Reading

The reading was taken directly from sensor and the data was uploaded to ThingSpeak through ESP8266 and Arduino. With the help of this the data were downloaded and used for further analysis in Python.

7.3.2 Data Analysis Using Python :

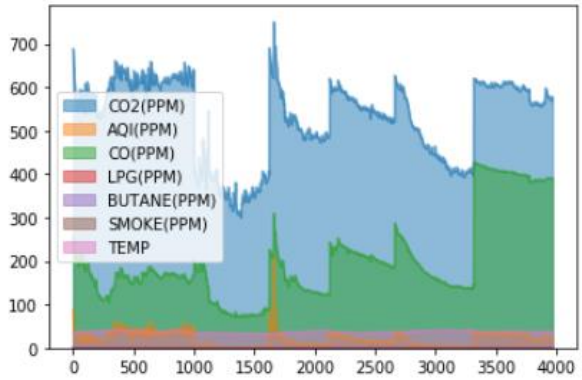


Fig 33. Area Plot.
This plot shows the data for every data point over a period of time. We can also see that PPM levels and temperature were high on some days and dipped on some as the sensors were kept on different places and on different days. The highs and dips for CO2, CO and AQI coincided which is a major point to note.

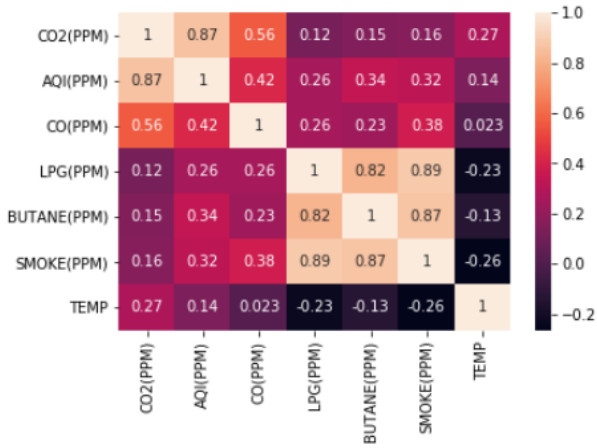


Fig 34. Correlation Heat Map.
We can easily identify the correlation of the parameters in the dataset. It is easy to see that the AQI is independent on these factors to various extent. A high negative as well as a moderate correlation is present. From here we can identify that AQI is inversely to the rest of the value. It is to be noted that negative relations mean that as one value decreases other increases.

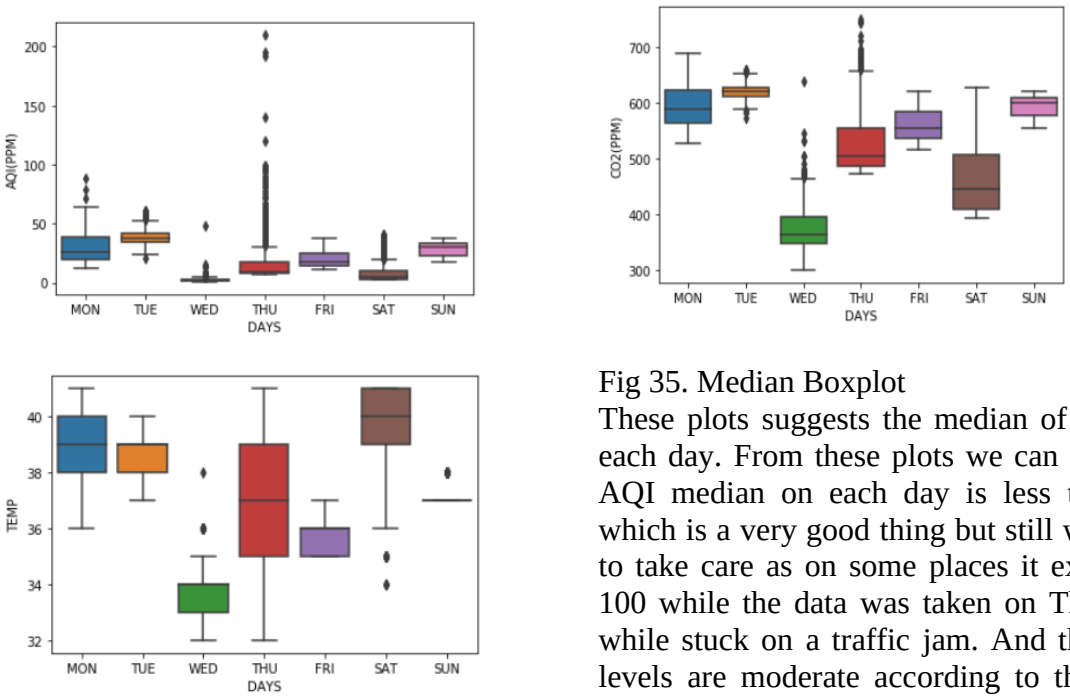


Fig 35. Median Boxplot
These plots suggests the median of data in each day. From these plots we can see that AQI median on each day is less then 50 which is a very good thing but still we need to take care as on some places it exceeded 100 while the data was taken on Thursday while stuck on a traffic jam. And the CO2 levels are moderate according to the table given below.

Air Quality Index	Who Needs to be Concerned?	What Should I Do?
Good 0-50	It's a great day to be active outside.	
Moderate 51-100	Some people who may be unusually sensitive to particle pollution.	**Unusually sensitive people:** *Consider reducing* prolonged or heavy exertion. Watch for symptoms such as coughing or shortness of breath. These are signs to take it easier. **Everyone else:** It's a good day to be active outside.
Unhealthy for Sensitive Groups 101-150	Sensitive groups include **people with heart or lung disease, older adults, children and teenagers.**	**Sensitive groups:** *Reduce* prolonged or heavy exertion. It's OK to be active outside, but take more breaks and do less intense activities. Watch for symptoms such as coughing or shortness of breath. **People with asthma** should follow their asthma action plans and keep quick relief medicine handy. **If you have heart disease:** Symptoms such as palpitations, shortness of breath, or unusual fatigue may indicate a serious problem. If you have any of these, contact your heath care provider.
Unhealthy 151 to 200	Everyone	**Sensitive groups:** *Avoid* prolonged or heavy exertion. Move activities indoors or reschedule to a time when the air quality is better. **Everyone else:** *Reduce* prolonged or heavy exertion. Take more breaks during all outdoor activities.
Very Unhealthy 201-300	Everyone	**Sensitive groups:** *Avoid all* physical activity outdoors. Move activities indoors or reschedule to a time when air quality is better. **Everyone else:** *Avoid* prolonged or heavy exertion. Consider moving activities indoors or rescheduling to a time when air quality is better.
Hazardous 301-500	Everyone	**Everyone**: *Avoid all* physical activity outdoors. **Sensitive groups:** Remain indoors and keep activity levels low. Follow tips for keeping particle levels low indoors.

CO_2 [ppm]	Air Quality
2100	BAD Heavily contaminated indoor air Ventilation required
2000	
1900	
1800	
1700	
1600	
1500	MEDIOCRE Contaminated indoor air Ventilation recommended
1400	
1300	
1200	
1100	
1000	FAIR
900	
800	GOOD
700	
600	EXCELLENT
500	
400	

Table 3. Indexes.

7.3.3 VAR Results :

```
   Augmented Dickey-Fuller Test on "CO2(PPM)"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -65.4296
No. Lags Chosen       = 0
Critical value 1%     = -3.432
Critical value 5%     = -2.862
Critical value 10%    = -2.567
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.

   Augmented Dickey-Fuller Test on "AQI(PPM)"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -15.5782
No. Lags Chosen       = 31
Critical value 1%     = -3.432
Critical value 5%     = -2.862
Critical value 10%    = -2.567
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.

   Augmented Dickey-Fuller Test on "CO(PPM)"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -61.3673
No. Lags Chosen       = 0
Critical value 1%     = -3.432
Critical value 5%     = -2.862
Critical value 10%    = -2.567
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.
```

```
   Augmented Dickey-Fuller Test on "LPG(PPM)"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -16.6972
No. Lags Chosen       = 22
Critical value 1%     = -3.432
Critical value 5%     = -2.862
Critical value 10%    = -2.567
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.

   Augmented Dickey-Fuller Test on "SMOKE(PPM)"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -16.8525
No. Lags Chosen       = 24
Critical value 1%     = -3.432
Critical value 5%     = -2.862
Critical value 10%    = -2.567
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.

   Augmented Dickey-Fuller Test on "TEMP"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -22.7435
No. Lags Chosen       = 7
Critical value 1%     = -3.432
Critical value 5%     = -2.862
Critical value 10%    = -2.567
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.
```

Fig 36. Since the VAR model requires the time series we want to forecast to be stationary, it is customary to check all the time series in the system for stationarity.
Just to refresh, a stationary time series is one whose characteristics like mean and variance does not change over time. This is done through Augmented Dickey-Fuller Test (ADF Test).

```
In [29]: model_fitted = model.fit(3)
         model_fitted.summary()

Out[29]:   Summary of Regression Results
           ==================================
           Model:                         VAR
           Method:                        OLS
           Date:           Mon, 26, Apr, 2021
           Time:                     11:30:53
           --------------------------------------------------------------------
           No. of Equations:         6.00000    BIC:               -21.3902
           Nobs:                     3961.00    HQIC:              -21.5070
           Log likelihood:           9113.08    FPE:            4.28339e-10
           AIC:                     -21.5711    Det(Omega_mle):    4.16216e-10
           --------------------------------------------------------------------
           Results for equation CO2(PPM)
           ==============================================================================
                            coefficient       std. error          t-stat            prob
           ------------------------------------------------------------------------------
           const              -0.034327        0.141955          -0.242           0.809
           L1.CO2(PPM)         0.041153        0.036916           1.115           0.265
           L1.AQI(PPM)        -0.496988        0.108391          -4.585           0.000
           L1.CO(PPM)          0.101756        0.037225           2.734           0.006
           L1.LPG(PPM)       -594.185216      201.490777          -2.950           0.003
```

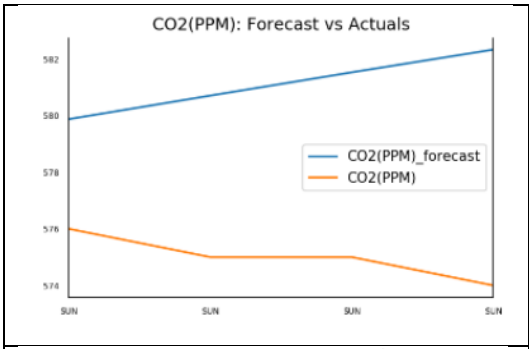
```
In [29]: model_fitted = model.fit(3)
         model_fitted.summary()

         L3.SMOKE(PPM)     -17.815378       91.931750          -0.194           0.846
         L3.TEMP            -0.414029        0.777026          -0.533           0.594
         ==============================================================================

         Results for equation AQI(PPM)
         ==============================================================================
                          coefficient       std. error          t-stat            prob
         ------------------------------------------------------------------------------
         const              -0.020287        0.050071          -0.405           0.685
         L1.CO2(PPM)         0.075327        0.013021           5.785           0.000
         L1.AQI(PPM)        -0.450297        0.038232         -11.778           0.000
         L1.CO(PPM)          0.051503        0.013130           3.922           0.000
         L1.LPG(PPM)      -776.525113       71.038932         -10.931           0.000
         L1.SMOKE(PPM)     795.899496       28.025535          28.399           0.000
         L1.TEMP             0.539363        0.273862           1.969           0.049
         L2.CO2(PPM)        -0.010736        0.013126          -0.818           0.413
         L2.AQI(PPM)        -0.024984        0.038837          -0.643           0.520
         L2.CO(PPM)          0.007463        0.013227           0.564           0.573
         L2.LPG(PPM)       -93.288042       72.862760          -1.280           0.200
         L2.SMOKE(PPM)     115.794672       32.433262           3.570           0.000
         L2.TEMP            -0.183414        0.284327          -0.645           0.519
```

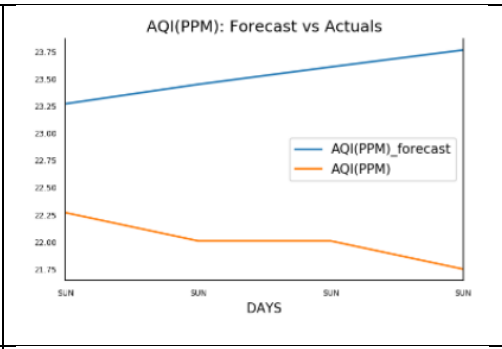
```
model_fitted = model.fit(3)
model_fitted.summary()

L3.SMOKE(PPM)     -33.129661       32.426555          -1.022           0.307
L3.TEMP            -0.291312        0.274076          -1.063           0.288
==============================================================================

Results for equation CO(PPM)
==============================================================================
                 coefficient       std. error          t-stat            prob
------------------------------------------------------------------------------
const              0.035604        0.092860           0.383           0.701
L1.CO2(PPM)        0.083911        0.024148           3.475           0.001
L1.AQI(PPM)       -0.263942        0.070904          -3.723           0.000
L1.CO(PPM)        -0.005437        0.024351          -0.223           0.823
L1.LPG(PPM)     -382.523653      131.745600          -2.904           0.004
L1.SMOKE(PPM)    390.473135       51.974894           7.513           0.000
L1.TEMP            0.045014        0.507892           0.089           0.929
L2.CO2(PPM)        0.025830        0.024343           1.061           0.289
L2.AQI(PPM)       -0.001583        0.072026          -0.022           0.982
L2.CO(PPM)        -0.048153        0.024530          -1.963           0.050
L2.LPG(PPM)       85.718825       135.127989           0.634           0.526
L2.SMOKE(PPM)     24.483227        60.149265           0.407           0.684
L2.TEMP           -1.086118        0.527300          -2.060           0.039
```

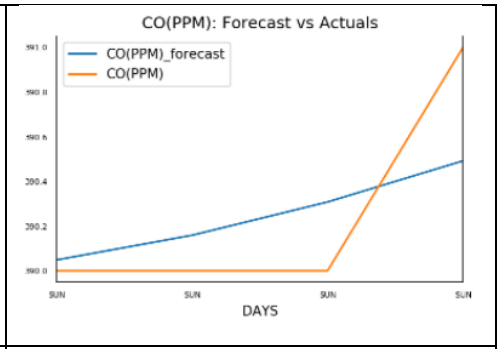
```
model_fitted = model.fit(3)
model_fitted.summary()

L3.SMOKE(PPM)     -0.007756        0.029571          -0.262           0.793
L3.TEMP           -0.000043        0.000250          -0.170           0.865
==============================================================================

Results for equation TEMP
==============================================================================
                 coefficient       std. error          t-stat            prob
------------------------------------------------------------------------------
const              0.000858        0.003391           0.253           0.800
L1.CO2(PPM)        0.002251        0.000882           2.552           0.011
L1.AQI(PPM)       -0.001250        0.002589          -0.483           0.629
L1.CO(PPM)        -0.006785        0.000889          -7.630           0.000
L1.LPG(PPM)       -1.108781        4.811168          -0.230           0.818
L1.SMOKE(PPM)      2.138963        1.898052           1.127           0.260
L1.TEMP           -0.310013        0.018548         -16.715           0.000
L2.CO2(PPM)        0.001523        0.000889           1.714           0.087
L2.AQI(PPM)       -0.002512        0.002630          -0.955           0.340
L2.CO(PPM)        -0.003147        0.000896          -3.513           0.000
L2.LPG(PPM)        3.670204        4.934689           0.744           0.457
L2.SMOKE(PPM)      2.837984        2.196569           1.292           0.196
L2.TEMP           -0.130958        0.019256          -6.801           0.000
```

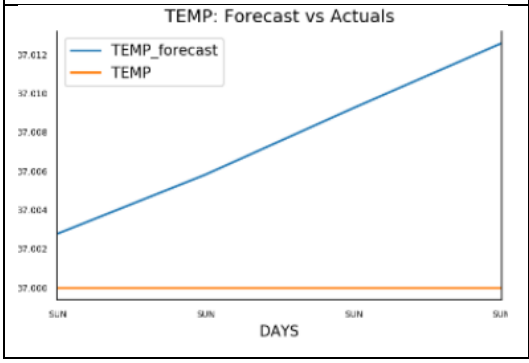
```
model_fitted = model.fit(3)
model_fitted.summary()

L2.LPG(PPM)        3.670204        4.934689           0.744           0.457
L2.SMOKE(PPM)      2.837984        2.196569           1.292           0.196
L2.TEMP           -0.130958        0.019256          -6.801           0.000
L3.CO2(PPM)       -0.000805        0.000860          -0.936           0.349
L3.AQI(PPM)        0.005738        0.002483           2.311           0.021
L3.CO(PPM)        -0.000588        0.000886          -0.664           0.507
L3.LPG(PPM)       -0.289549        4.892379          -0.059           0.953
L3.SMOKE(PPM)     -4.362134        2.196114          -1.986           0.047
L3.TEMP           -0.050675        0.018562          -2.730           0.006
==============================================================================

Correlation matrix of residuals
             CO2(PPM)  AQI(PPM)   CO(PPM)  LPG(PPM)  SMOKE(PPM)      TEMP
CO2(PPM)     1.000000  0.785029  0.614026  0.157415    0.116646 -0.060648
AQI(PPM)     0.785029  1.000000  0.490515  0.519484    0.524658 -0.052798
CO(PPM)      0.614026  0.490515  1.000000  0.254122    0.205237 -0.438271
LPG(PPM)     0.157415  0.519484  0.254122  1.000000    0.715019 -0.085157
SMOKE(PPM)   0.116646  0.524658  0.205237  0.715019    1.000000 -0.045948
TEMP        -0.060648 -0.052798 -0.438271 -0.085157   -0.045948  1.000000
```

Fig 37. Training the VAR Model of the selected order(p)









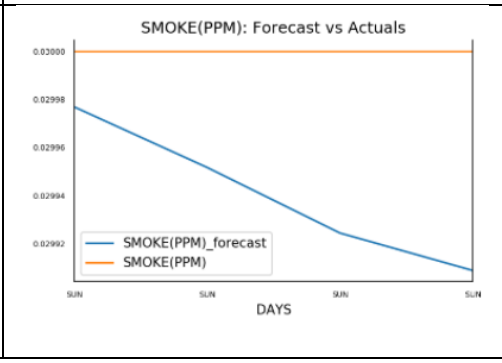


Fig 38. Plot of Forecast vs Actuals
We can see here that difference between the predicted values and the actuals are very low and in case of temperature and smoke the difference is in points which is a very good sign.

```
Forecast Accuracy of: CO2(PPM)
mape :  0.0106
me :  6.1084
mae :  6.1084
mpe :  0.0106
rmse :  6.3151
corr :  -0.9483

Forecast Accuracy of: AQI(PPM)
mape :  0.0691
me :  1.5168
mae :  1.5168
mpe :  0.0691
rmse :  1.5597
corr :  -0.9509

Forecast Accuracy of: CO(PPM)
mape :  0.0007
me :  0.0016
mae :  0.2558
mpe :  0.0
rmse :  0.3085
corr :  0.833
```

```
Forecast Accuracy of: LPG(PPM)
mape :  0.0018
me :  -0.0
mae :  0.0
mpe :  -0.0018
rmse :  0.0
corr :  nan

Forecast Accuracy of: SMOKE(PPM)
mape :  0.002
me :  -0.0001
mae :  0.0001
mpe :  -0.002
rmse :  0.0001
corr :  nan

Forecast Accuracy of: TEMP
mape :  0.0002
me :  0.0076
mae :  0.0076
mpe :  0.0002
rmse :  0.0084
corr :  nan
```

Fig 39. Accuracy of the predicted values.
As we can see the mape (Mean Absolute Percentage Error) of each predicted value is less than 10% which suggests that the predicted values are quite accurate.