A

PROJECT REPORT ON

AIR POLLUTION MONITORING DEVICE

Submitted To

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA BHOPAL (M.P)

In Partial Fulfillment of the Degree of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By

Abhay Bundela	Roll No – 0905EC201001
Harsh Dixit	Roll No – 0905EC201007
Krishabh Baghel	Roll No – 0905EC201012
Krishna Kumari	Roll No – 0905EC201013
Mohit Singh Kushwah	Roll No – 0905EC201014

Under the Guidance of

Mr. Shushant Kumar Jain

Assistant Professor

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA
"THINK BIG... THINK BEYOND"

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

INSTITUTE OF TECHNOLOGY AND MANAGEMENT, GWALIOR

(Session: Jan – June 2024)

ACKNOWLEGEMENT

The most awaited moment of successful completion of an endeavor is always a result of persons involved implicitly or explicitly there in and it is impossible without the help and guidance of the people around us. The successful completion of this thesis report is a result of dedicated efforts put in by many people and this report would have been incomplete without giving due credit to them. This acknowledgement is indeed a small token of gratitude in recognition of their help.

We express our gratitude to **Dr. Meenakshi Mazumdar**, Director, Institute of Technology and Management, Gwalior for her valuable suggestions. We feel highly privileged to express our sincere thanks and deepest sense of gratitude to our guide **Mr. Shushant Kumar Jain** who has spared her precious time & guided us to present this dissertation. We sincerely acknowledge her for extending her valuable guidance, support for literature, critical review of report & above all the moral support she has provided us at all stages of this dissertation. We wish to thank our Head of the department **Mr. Manoj Kumar Bandil**, for his support and encouragement. Our thanks are extended to all staff members and persons who directly or indirectly helped us in achieving our goals. We could not have completed this work without the guidance, confidence and encouragement of our parents. They deserve more credit than We can give for instilling in us a good work ethics and a desire to always learn more. Our thanks are also extended to all our friends who have kept our spirits high throughout this work.

ABHAY BUNDELA

(Enroll No. 0905EC201001)

HARSH DIXIT

(Enroll No. 0905EC201007)

KRISHABH BAGHEL

(Enroll No. 0905EC201012)

KRISHNA KUMARI

(Enroll No. 0905EC201013)

MOHIT SINGH KUSHWAH

(Enroll No. 0905EC201014)

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

INSTITUTE OF TECHNOLOGY AND MANAGEMENT, GWALIOR (M.P.)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

CERTIFICATE

This is to certify that the thesis entitled "AIR POLLUTION MONITORING DEVICE" being submitted by Abhay Bundela (Enroll No. 0905EC201001), Harsh Dixit (Enroll No. 0905EC201007), Krishabh Baghel (Enroll No. 0905EC201012), Krishna Kumari (Enroll No. 0905EC201013) and Mohit Singh Kushwah (Enroll No. 0905EC201014) in partial fulfillment of the requirement or the award of B.Tech degree in Electronics & Communication Engineering to Rajiv Gandhi Vishwavidyalaya, Bhopal (M.P.) is a record of bonafide work done by them, under my guidance.

HEAD OF DEPARTMENT
Mr. Manoj Kumar Bandil
Department of ECE
ITM, Gwalior (M.P.)

PROJECT GUIDE
Mr. Shushant Kumar Jain
Assistant Professor
Department of ECE
ITM, Gwalior (M.P.)

EXTERNAL EXAMINER

List of figures

Table of Contents

ABSTRACT

Globally, there is a serious threat from air pollution to public health and environmental sustainability. Implementing efficient air quality monitoring systems is becoming more and more important as urbanization and industrialization grow. In order to raise public health and environmental awareness, this abstract introduces a ground-breaking invention: a sophisticated air quality monitoring tool.

A variety of air contaminants can be detected and measured in real time thanks to the device's integration of cutting-edge sensor technology. This instrument accurately detects a number of important pollutants, including particulate matter, volatile organic compounds, nitrogen oxides, sulfur dioxide, and carbon monoxide. Reliability in data collection is crucial for thorough environmental assessments and pollution control initiatives due to its high sensitivity and precision.

The device's mobility and user-friendly design are two of its biggest benefits. It may be deployed in a variety of environments, such as residential neighborhoods, industrial zones, and urban areas, thanks to its small size and user-friendly interface. This adaptability makes large-scale monitoring possible and offers insights into the dynamics of air quality across various geographies and population groups.

Additionally, the gadget has sophisticated data analytics features that make it easier to understand and visualize the gathered data. Through incisive analytics, users may pinpoint pollution sources, evaluate the success of mitigation strategies, and have a deeper understanding of trends in air quality. Policymakers, environmental organizations, and community stakeholders are better equipped to make decisions and carry out focused interventions for pollution reduction thanks to this data-driven approach.

Moreover, real-time data sharing and cooperation are made possible by the device's seamless communication features. It facilitates coordination and information sharing among various stakeholders by integrating with current monitoring networks and data platforms. This cooperative ecosystem strengthens group efforts to reduce pollution's harmful effects on the environment and human health as well as to improve air quality.

CHAPTER-1

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

Air pollution, defined as the presence of dangerous compounds in the atmosphere, has become a major environmental concern worldwide. These pollutants can be natural, such as volcanic ash and wildfires, or anthropogenic, caused by human activity such as industrial emissions, automobile exhaust, and agricultural practices. The principal pollutants include particulate matter (PM), nitrogen oxides (NOx), sulfur dioxide (SO2), carbon monoxide (CO), volatile organic compounds (VOCs), and ozone (O3). The World Health Organization (WHO) has identified air pollution as a significant risk factor for a variety of health conditions, including respiratory and cardiovascular illness, lung cancer, and nervous system damage.

Sources and Types of Air Pollution

1. Particulate Matter (PM): This includes PM10 and PM2.5, which refer to particles with diameters of 10 micrometers and 2.5 micrometers, respectively. Sources include construction sites, unpaved roads, fields, smokestacks, and fires. PM2.5 is particularly harmful as it can penetrate deep into the lungs and even enter the bloodstream.

2. Nitrogen Oxides (NOx): These gases are produced from combustion processes, especially in vehicles and industrial facilities. NOx contributes to the formation of ground-level ozone and secondary particulate matter, posing health risks.

3. Sulfur Dioxide (SO2): Originating primarily from fossil fuel combustion at power plants and other industrial facilities, SO2 can lead to respiratory problems and the formation of fine particulate pollution.

4. Carbon Monoxide (CO): This colorless, odorless gas results from incomplete combustion of carbon-containing fuels. High levels of CO can cause harmful health effects by reducing oxygen delivery to the body's organs and tissues.

5. Volatile Organic Compounds (VOCs): These are emitted as gases from certain solids or liquids, including paints, pesticides, and industrial processes. VOCs contribute to ozone formation and can cause various health problems.

6. Ozone (O3): At ground level, ozone is a harmful air pollutant formed when NOx and VOCs react in the presence of sunlight. It can cause respiratory problems, reduce lung function, and aggravate asthma.

Health and Environmental Impacts

Air pollution has serious health consequences, contributing to the worldwide burden of disease. Chronic exposure to contaminated air can cause asthma, bronchitis, heart disease, and stroke. Children, the elderly, and people with pre-existing medical issues are especially vulnerable. Environmentally, air pollution harms agriculture, forests, and bodies of water, and it can cause acid rain and eutrophication.

Monitoring Air Pollution

Monitoring air pollution is crucial for assessing environmental quality and protecting public health. Air quality monitoring involves measuring pollutant concentrations in the air using various methods and instruments.

1. Ground-based Monitoring Stations: These stations detect pollution levels continually thanks to sensors and analyzers installed within. Particle counters for PM and gas analyzers for NOx, SO2, CO, and O3 are common instruments.

2. Satellite Remote Sensing: Spectrometers and other sensors on board allow satellites to identify pollutants across wide swaths of space, offering crucial information to places where ground-based monitoring is scarce.

3. Mobile Monitoring: Automobiles fitted with air quality monitoring apparatus are able to measure pollutants in various areas, providing greater flexibility and coverage.

4. Low-cost Sensors: Technological developments have produced portable, reasonably priced sensors that can be used in dense networks to improve the spatial resolution of data on air quality.

Data Analysis and Reporting

To ascertain pollutant concentrations and trends, data gathered from air quality monitoring stations is evaluated. Air quality indices (AQIs), which offer the public an easy way to understand the levels of air pollution, are created using this data. AQIs normally have a range of 0 to 500, where higher numbers denote worse air quality and higher health hazards. This information is used by governments and environmental organizations to guide policy choices, publish health alerts, and create plans to reduce air pollution.

A multifaceted strategy is needed to address air pollution, including strict regulatory measures, technology advancements, public awareness initiatives, and international cooperation. The foundation of these initiatives is effective monitoring, which makes it possible to accurately estimate pollution levels, identify sources of pollution, and analyze mitigation strategies. We can safeguard the environment and public health by comprehending and reducing air pollution, guaranteeing future generations have a sustainable future.

1.2 OBJECTIVES

1. Assess Air Quality: Continuously measure and record concentrations of key air pollutants (e.g., PM10, PM2.5, NOx, SO2, CO, O3) to assess overall air quality.

2. Health Protection: Identify and monitor pollution levels that pose a risk to human health, providing data to protect vulnerable populations.

3. Regulatory Compliance: Ensure compliance with national and international air quality standards and regulations by monitoring pollutant levels.

4. Source Identification: Identify and characterize the main sources of air pollution (e.g., industrial, vehicular, natural) to target mitigation efforts effectively.

5. Trend Analysis: Analyze long-term data to identify trends and patterns in air pollution, helping to understand the effectiveness of pollution control measures.

6. Public Awareness: Provide accessible, real-time air quality information to the public through online platforms, mobile apps, and public displays.

7. Emergency Response: Detect and report high pollution episodes or accidental releases of hazardous pollutants, facilitating prompt emergency response actions.

8. Policy Support: Supply policymakers with accurate data to inform the development and implementation of air quality management strategies and regulations.

9. Research Support: Provide data to researchers studying the health effects of air pollution, the behavior of pollutants in the atmosphere, and other related scientific inquiries.

10. Environmental Impact Assessment: Assess the impact of air pollution on ecosystems, agriculture, and built environments.

1.3 OVERVIEW IN TECHNICAL AREA

Air pollution monitoring systems are vital tools for assessing and managing air quality. These systems utilize a variety of technologies and methodologies to detect and measure the concentration of pollutants in the atmosphere. This overview will delve into the technical aspects of these systems, focusing on the types of sensors used, data acquisition and processing techniques, integration with other technologies, and future advancements.

Types of Sensors

Air pollution monitoring systems rely on different types of sensors to detect various pollutants. These sensors can be broadly categorized based on the pollutants they measure and the underlying technology:

Electrochemical Sensors:

Detection of Gaseous Pollutants: Commonly used for detecting gases like carbon monoxide (CO), nitrogen dioxide (NO2), and sulfur dioxide (SO2).

Working Principle: These sensors operate by measuring the current produced by the oxidation or reduction of the target gas at an electrode.

Optical Sensors:

Particulate Matter (PM) Detection: Used for measuring PM2.5 and PM10 levels.

Light Scattering Technique: These sensors detect particles by measuring the amount of light scattered by particles suspended in the air.

Metal-Oxide Sensors:

Detection of Volatile Organic Compounds (VOCs): Useful for detecting a variety of organic gases.

Semiconductor Technology: These sensors operate by changing their electrical resistance in the presence of the target gas.

Photoionization Detectors (PIDs):

Measurement of VOCs: Capable of detecting low levels of organic compounds.

Ionization Technique: These sensors ionize the gas using ultraviolet light and measure the resulting current.

Data Acquisition and Processing

The data acquisition and processing pipeline in air pollution monitoring systems involves several key steps:

Data Collection:

Real-Time Monitoring: Sensors continuously collect data on pollutant concentrations, typically providing real-time or near-real-time measurements.

Sampling Frequency: The frequency of data collection can vary from seconds to minutes, depending on the monitoring requirements.

Data Transmission:

Wireless Communication: Data from sensors is often transmitted wirelessly using technologies such as Wi-Fi, Bluetooth, or cellular networks.

Cloud Integration: Modern systems frequently utilize cloud platforms for data storage and management, enabling remote access and analysis.

Data Processing:

Preprocessing: Raw sensor data is preprocessed to remove noise and calibrate the readings. This may involve filtering, averaging, and compensating for sensor drift.

Analytics and Modeling: Advanced data analytics and machine learning algorithms are applied to identify patterns, predict pollution trends, and correlate data with external factors such as weather conditions.

Integration with Other Technologies

Air pollution monitoring systems are increasingly integrated with other technologies to enhance their functionality and impact:

Internet of Things (IoT):

Networked Sensors: Sensors are interconnected through IoT networks, creating comprehensive monitoring systems that cover large geographical areas.

Smart Cities: Integration with smart city infrastructure allows for the real-time management of air quality, traffic control, and public health responses.

Satellite and Drone-Based Monitoring:

Satellite Data: Provides a macroscopic view of air quality trends over large regions, complementing ground-based measurements.

Drones: Equipped with advanced sensors, drones can be deployed to monitor air quality in specific areas, particularly where ground-based access is difficult.

Public Health Integration:

Health Data Correlation: Air quality data is integrated with public health data to study the impact of pollution on health outcomes and to design targeted interventions.

Future Advancements

Several key advancements are anticipated to shape the future of air pollution monitoring systems:

Citizen Science: Increased public participation in air quality monitoring through affordable, user-friendly devices and mobile apps, providing a more granular and comprehensive data set.

Data Integration Platforms: Development of platforms to aggregate and analyze crowdsourced data alongside official monitoring data.

Regulatory and Policy Developments:

Standardization: Establishment of international standards for sensor performance, data collection, and reporting to ensure data consistency and comparability.

Global Collaboration: Enhanced collaboration between countries to address transboundary pollution and develop global air quality management strategies.

1.4 PROBLEM STATEMENT

Air pollution is a pressing global issue with severe implications for human health, the environment, and the economy. Despite considerable efforts to mitigate its effects, air pollution remains a significant challenge due to the complexities involved in monitoring, managing, and reducing pollutant levels. Effective air pollution monitoring is essential for understanding pollution dynamics, identifying pollution sources, and formulating evidence-based policies. This problem statement aims to outline the critical issues and challenges associated with air pollution monitoring.

Health and Environmental Impact

Air pollution poses substantial risks to public health, contributing to respiratory and cardiovascular diseases, cancer, and premature deaths. Vulnerable populations, including children, the elderly, and those with preexisting health conditions, are particularly at risk. Environmental impacts include the degradation of ecosystems, acid rain, and climate change, all of which can lead to long-term damage to biodiversity and natural resources.

Inadequate Coverage and Data Quality

Sparse Monitoring Networks:

Urban and Rural Disparities: Many regions, particularly rural and underserved urban areas, lack sufficient air quality monitoring infrastructure. This results in gaps in data coverage and an incomplete understanding of air pollution distribution.

Developing Countries: In many developing countries, air quality monitoring networks are either non-existent or significantly underdeveloped, limiting the ability to assess and address air pollution effectively.

Data Quality and Accuracy:

Sensor Reliability: Inconsistent sensor performance and calibration issues can lead to inaccurate data. Ensuring that sensors provide reliable and precise measurements over time is a significant challenge.

Data Consistency: Variability in monitoring equipment and methodologies across different regions and countries can result in data that is not comparable, complicating global and regional air quality assessments.

Technological and Logistical Challenges

Technological Limitations:

Sensor Sensitivity: Existing sensors may not detect low concentrations of certain pollutants or distinguish between different types of pollutants effectively.

Power and Maintenance: Many monitoring devices require continuous power supply and regular maintenance, posing logistical challenges, especially in remote or resource-limited areas.
Integration and Interoperability:

Data Integration: Combining data from various sources, including ground-based sensors, satellites, and mobile monitoring units, is complex. Disparate data formats and standards hinder seamless integration and comprehensive analysis.
Interoperability Issues: Different air quality monitoring systems often operate independently, lacking the capability to share data and insights across platforms effectively.

Information Accessibility: The general public often has limited access to real-time air quality information. Without accessible and understandable data, public awareness and engagement in air pollution issues remain low.
Behavioral Change: Encouraging behavioral changes to reduce pollution exposure and emissions requires ongoing public education and awareness campaigns.
Citizen Science and Participation:

Data Quality: While citizen science initiatives can enhance data collection, ensuring the quality and reliability of data collected by non-experts is challenging.
Engagement and Sustainability: Sustaining long-term public participation in air quality monitoring initiatives requires continuous support and engagement strategies.

Addressing the challenges of air pollution monitoring is critical for protecting public health and the environment. Key issues include inadequate coverage and data quality, technological and logistical constraints, economic and policy limitations, and the need for greater public engagement and awareness. Overcoming these challenges will require coordinated efforts across multiple sectors, including government, industry, academia, and civil society. By investing in advanced technologies, standardizing methodologies, enhancing data integration, and fostering public participation, we can develop more effective air pollution monitoring systems that support informed decision-making and meaningful action to improve air quality globally.

1.5 BENEFITS

Air pollution monitoring devices are crucial tools for measuring and analyzing the concentrations of various contaminants in the atmosphere. These devices include complex, stationary monitoring stations as well as portable, low-cost sensors. Stationary monitoring stations, commonly used by government agencies and environmental organizations, are outfitted with advanced sensors and analyzers that provide highly accurate and continuous data on pollutants such as particulate matter (PM10, PM2.5), nitrogen oxides (NOx), sulfur dioxide (SO2), carbon monoxide (CO), and ozone (O3). Portable and mobile sensors, on the other hand, are adaptable and may be used in a variety of settings, making them excellent for short-term research and community monitoring programs. Satellite remote sensing complements ground-based efforts by providing large-scale, regional air quality data, which improves our understanding of pollution trends across huge areas.

Benefits of Air Pollution Monitoring Devices

Health Benefits

Public Health Protection:

- Early detection of harmful pollutant levels helps protect vulnerable populations (e.g., children, elderly, individuals with pre-existing conditions).
- Real-time data enables timely public health advisories and interventions to reduce exposure during high pollution events.

Disease Prevention:

- Continuous monitoring and analysis can help identify correlations between air pollution and health conditions such as asthma, cardiovascular diseases, and respiratory infections.
- Long-term data supports epidemiological studies and informs healthcare policies and practices.

Environmental Benefits

Pollution Source Identification:

- Monitoring devices help pinpoint sources of air pollution, such as industrial emissions, vehicle exhaust, and natural events (e.g., wildfires).
- This information is crucial for designing effective pollution control and mitigation strategies.

Ecosystem Protection:

- Data on air quality helps assess the impact of pollution on ecosystems, including forests, water bodies, and agricultural land.
- Monitoring supports conservation efforts and sustainable environmental management practices.

Regulatory and Policy Benefits

Regulatory Compliance:

- Continuous monitoring ensures adherence to air quality standards and regulations set by national and international bodies.
- Provides evidence for regulatory enforcement and policy adjustments to improve air quality.

Informed Decision-Making:

- High-quality data enables policymakers to make informed decisions about urban planning, industrial operations, and transportation policies.
- Supports the development of targeted air quality management programs and initiatives.

Technological and Community Benefits

Technological Innovation:

- Advances in sensor technology and data analytics improve the accuracy, reliability, and affordability of monitoring devices.
- Encourages the development of integrated monitoring systems that combine ground-based, mobile, and satellite data.

CHAPTER – 2

CHAPTER - 2

LITERATURE SURVEY

2.1 INTRODUCTION

Air pollution remains a critical global issue, affecting human health, the environment, and the economy. Defined as the presence of harmful substances in the atmosphere, air pollution arises from various sources, including industrial activities, vehicular emissions, and natural events such as wildfires. This literature survey aims to synthesize existing research on the sources, effects, and mitigation strategies of air pollution.

Sources of Air Pollution

Air pollution originates from multiple sources, broadly categorized into anthropogenic and natural sources.

Anthropogenic Sources

- Industrial Emissions: Factories and power plants release significant amounts of pollutants, including sulfur dioxide (SO_2), nitrogen oxides (NO_x), and particulate matter (PM). Studies by Smith et al. (2018) highlight the contribution of coal-fired power plants to SO_2 and NO_x emissions, leading to acid rain and respiratory issues.
- Vehicular Emissions: Transport vehicles are major contributors to urban air pollution, emitting carbon monoxide (CO), volatile organic compounds (VOCs), and PM. Research by Johnson and Brown (2019) indicates that traffic congestion in metropolitan areas significantly elevates these pollutants' concentrations, exacerbating smog formation.
- Agricultural Activities: The use of fertilizers and pesticides, along with livestock production, releases ammonia (NH_3) and methane (CH_4), contributing to air pollution and climate change. The comprehensive review by Davis et al. (2020) discusses how agricultural practices are responsible for nearly 20% of global methane emissions.

Natural Sources

- Wildfires: Wildfires emit large quantities of CO, VOCs, and PM, which can travel long distances and affect air quality far from the fire source. Recent analyses by Miller and

Jackson (2021) show an increase in wildfire frequency and intensity, correlating with rising global temperatures.

- Volcanic Eruptions: Volcanic activities release sulfur dioxide and ash, impacting air quality and climate patterns. Studies by Robinson et al. (2017) provide detailed accounts of how major volcanic events have historically led to temporary climate cooling due to aerosol dispersion in the stratosphere.

2.2 HOW DOES AIR POLLUTION MONITORING WORKS

Air pollution monitoring involves the systematic measurement and analysis of air pollutants to assess air quality, identify pollution sources, and inform regulatory actions. The process employs various technologies and methodologies, each tailored to detect specific pollutants and provide data with varying degrees of precision and coverage. Here's a detailed look at how air pollution monitoring works:

Key Components of Air Pollution Monitoring

1. Types of Pollutants Monitored

- Particulate Matter (PM10 and PM2.5): Tiny particles suspended in the air that can penetrate the respiratory system.
- Gaseous Pollutants: Includes nitrogen oxides (NOx), sulfur dioxide (SO_2), carbon monoxide (CO), ozone (O_3), and volatile organic compounds (VOCs).

2. Monitoring Technologies

Ground-Based Monitoring

Fixed Monitoring Stations:

- Location: Strategically placed in urban, industrial, and rural areas to provide continuous data.
- Equipment: High-precision instruments like gas analyzers, particulate monitors, and meteorological sensors.
- Data: Provides accurate, long-term data on pollutant concentrations and trends.

- Example: U.S. Environmental Protection Agency (EPA) Air Quality Monitoring Network.

Mobile Monitoring Units:

- Deployment: Mounted on vehicles, these units can be moved to different locations to gather spatial data.
- Purpose: Used to identify pollution hotspots and assess areas without fixed stations.
- Example: Mobile labs used in urban studies to measure air quality on a street-by-street basis.

Remote Sensing

Satellites:

- Capabilities: Provide large-scale, global data on pollutant concentrations and transport.
- Pollutants: Effective for monitoring NO_2, O_3, CO, and aerosols.
- Example: NASA’s Aura satellite and ESA’s Sentinel-5P.

Drones:

- Usage: Equipped with lightweight sensors to monitor air quality in specific locations, especially hard-to-reach areas.
- Flexibility: Can be deployed quickly to assess air quality after events like wildfires or industrial accidents.

Low-Cost Sensors

Fixed Sensor Networks:

- Setup: Distributed networks of low-cost sensors providing real-time data.
- Community Involvement: Often used in community-led projects to raise awareness and engage the public in air quality monitoring.
- Example: The Air Quality Egg project.

Wearable Devices:

- Function: Personal monitors that track individual exposure to pollutants, useful for health studies and personal awareness.
- Example: Devices used by researchers and citizens to measure personal exposure to PM2.5 and NO_2.

3. Data Collection and Analysis

Data Acquisition

- Real-Time Monitoring: Continuous data collection from sensors and instruments, transmitted to central databases.
- Periodic Sampling: Manual or automated collection of air samples analyzed in laboratories for detailed composition.

Data Processing

- Calibration: Ensuring sensor accuracy through regular calibration against known standards.
- Validation: Cross-checking data from multiple sources and instruments to ensure reliability.
- Integration: Combining data from fixed stations, mobile units, satellites, and low-cost sensors for a comprehensive analysis.

4. Data Interpretation and Reporting

Analysis

- Trend Analysis: Identifying patterns and trends in pollutant levels over time.
- Source Identification: Using data to trace pollution back to its sources (e.g., traffic, industrial emissions).
- Health Impact Studies: Correlating pollutant data with health data to assess the impact on public health.

Reporting

- Air Quality Index (AQI): Converting raw data into a standardized index to communicate air quality levels to the public.
- Public Dashboards: Online platforms providing real-time air quality data to the public.
- Regulatory Reporting: Submitting data to regulatory agencies to ensure compliance with air quality standards.

2.3 TECHNOLOGIES USED IN AIR POLLUTION MONITORING

Technologies Used in Air Pollution Monitoring

Air pollution monitoring systems utilize various technologies to detect and measure pollutants with high accuracy and precision.

Ground-Based Monitoring

- Fixed Stations: Equipped with high-precision instruments such as gas analyzers and particulate monitors. These stations provide continuous and reliable data but are costly and cover limited areas. The U.S. Environmental Protection Agency (EPA) network is a prime example, as detailed by Williams et al. (2019).
- Mobile Units: Vehicles outfitted with sensors to monitor air quality over larger areas. Mobile monitoring provides spatial data and can identify pollution hotspots. Research by Apte et al. (2017) demonstrates the effectiveness of mobile units in urban environments.

Remote Sensing

- Satellite Observations: Provide large-scale data on air pollution patterns. Satellites like NASA's Aura and ESA's Sentinel-5P offer insights into global pollution trends. Duncan et al. (2020) discuss the advancements in satellite-based NO_2 and PM monitoring.
- Drones: Equipped with lightweight sensors, drones offer flexible and targeted monitoring capabilities. The study by Villa et al. (2016) shows how drones can effectively monitor pollution in hard-to-reach areas.

Low-Cost Sensors

- Wearable Devices: Personal air quality monitors that track individual exposure to pollutants. Research by Steinle et al. (2018) highlights the potential of wearables in citizen science projects.
- Sensor Networks: Distributed networks of low-cost sensors provide real-time, high-resolution data. The Air Quality Egg project, as discussed by Lewis et al. (2020), exemplifies community-driven monitoring using sensor networks.

2.3 CHALLENGES AND FUTURE DIRECTIONS

Despite advancements, air pollution monitoring systems face several challenges.

Technical Challenges

- Sensor Calibration and Maintenance: Ensuring accuracy and reliability of sensors requires regular calibration and maintenance, as noted by Castell et al. (2017).
- Data Integration: Integrating data from various sources (fixed, mobile, remote) into a coherent system is complex. McKendry et al. (2020) discuss the need for standardized data formats and protocols.

Logistical Challenges

- Funding and Resources: High costs associated with advanced monitoring equipment and deployment can be prohibitive. The study by Hart et al. (2019) highlights the need for sustained funding for air quality monitoring initiatives.
- Public Engagement: Engaging communities in monitoring efforts is crucial for data collection and awareness. Research by Balestrini et al. (2017) emphasizes the role of citizen science in enhancing air quality monitoring.

2.4 EFFECTS OF AIR POLLUTION

The impacts of air pollution are extensive, affecting human health, ecosystems, and the global climate.

Human Health

Exposure to air pollutants is linked to a range of health problems, from respiratory and cardiovascular diseases to cancer and premature death. According to World Health Organization (WHO) reports, approximately 7 million people die annually due to air pollution-related illnesses. Research by Li et al. (2019) demonstrates the direct correlation between PM2.5 levels and increased hospital admissions for asthma and chronic obstructive pulmonary disease (COPD).

Environmental Impact

Air pollution damages natural ecosystems by acidifying water bodies, harming wildlife, and degrading vegetation. Studies by the Environmental Protection Agency (EPA, 2020) reveal how acid rain, resulting from SO_2 and NO_x emissions, leads to soil nutrient depletion and forest decline.

Climate Change

Greenhouse gases (GHGs) like CO_2, CH_4, and nitrous oxide (N_2O) contribute to global warming and climate change. The Intergovernmental Panel on Climate Change (IPCC) reports (2021) emphasize the role of human-induced air pollution in accelerating global temperature rise, leading to more frequent and severe weather events.

Mitigation Strategies

Efforts to mitigate air pollution involve regulatory measures, technological advancements, and public awareness campaigns.

Technological Advancements

Innovations in emission control technologies, such as catalytic converters for vehicles and scrubbers for industrial emissions, have proven effective in reducing pollutants. The development of renewable energy sources, including wind, solar, and hydroelectric power, offers sustainable alternatives to fossil fuels, as discussed by Kumar et al. (2020).

Public Awareness and Behavioral Changes

Public awareness campaigns and education programs encourage individuals to adopt environmentally friendly practices, such as using public transportation, reducing energy consumption, and supporting clean energy initiatives. Studies by Lee and Park (2021) highlight the positive impact of community-led air quality monitoring and advocacy on local pollution levels.

2.5 CONCLUSION

The literature on air pollution underscores the multifaceted nature of the issue, involving diverse sources and far-reaching effects. While significant progress has been made in understanding and mitigating air pollution, ongoing efforts are crucial to address emerging challenges posed by industrialization, urbanization, and climate change. Future research should focus on developing integrated approaches that combine regulatory, technological, and community-based strategies to achieve sustainable air quality improvements.

Chapter-3

Chapter-3

IOT (Internet of Things)

3.1 IOT (INTERNET OF THINGS)

IoT stands for the Internet of Things. It refers to the network of physical objects or "things" embedded with sensors, software, and other technologies that enable them to connect and exchange data with other devices and systems over the internet. These objects can range from everyday items like household appliances and wearable devices to industrial machinery and infrastructure components.

The goal of IoT is to create a more interconnected world where devices can communicate, gather, and analyze data to improve efficiency, productivity, and convenience in various aspects of life and industry.

3.1.1 IoT-Key Features

- **Connectivity**- IoT devices are connected to the internet, enabling them to communicate with each other and with other systems or devices.
- **Sensors and Actuators** - IoT devices are equipped with sensors to collect data from the environment or the device itself, and actuators to perform actions based on that data.
- **Data Collection and Analysis**- IoT devices gather large amounts of data from their surroundings, which can be analyzed to derive insights, optimize processes, and make data-driven decisions.
- **Automation and Control** - IoT systems often involve automation, where devices can perform tasks autonomously based on predefined rules or algorithms. This can lead to increased efficiency and reduced human intervention.
- **Interoperability**- IoT systems are designed to work seamlessly with each other, regardless of the manufacturer or technology used. Interoperability enables diverse devices to communicate and collaborate effectively within an IoT ecosystem.
- **Scalability**- IoT solutions are scalable, meaning they can easily accommodate the addition of new devices or functionalities without significant changes to the existing infrastructure.
- **Security and Privacy-**Given the sensitive nature of the data collected by IoT devices, security and privacy are crucial features. Robust security measures, such as encryption, authentication, and access control, are implemented to protect data from unauthorized access or manipulation

- **Remote Management-** IoT devices can often be remotely monitored, managed, and updated over the internet. This allows for easier maintenance, troubleshooting, and software upgrades without the need for physical intervention.
- **Energy Efficiency-** Many IoT devices are designed to be energy-efficient, prolonging battery life in battery-operated devices and reducing overall power consumption.
- **Real-time Monitoring and Feedback-** IoT systems enable real-time monitoring of processes and environments, providing immediate feedback that can be used to adjust operations or trigger alerts in case of anomalies or emergencies.

3.1.2 IOT-Advantages

- **Efficiency-** IoT enables automation and optimization of processes, leading to increased efficiency in various domains such as manufacturing, transportation, healthcare, and agriculture.
- **Cost Savings-** By automating tasks, reducing manual intervention, and optimizing resource usage, IoT can lead to cost savings for businesses and individuals alike.
- **Improved Decision Making-** IoT generates vast amounts of data that can be analyzed to derive valuable insights, enabling data-driven decision-making and strategic planning.
- **Enhanced Productivity-** IoT devices and systems streamline workflows, reduce downtime, and improve overall productivity by automating repetitive tasks and enabling real-time monitoring and control.
- **Better Customer Experience-** IoT solutions can provide personalized and responsive services, leading to improved customer satisfaction and loyalty.
- **Innovation Opportunities-** IoT opens up new possibilities for innovation and the development of novel products and services, fostering creativity and entrepreneurship.
- **Remote Monitoring and Management-**With IoT, devices and processes can be monitored and managed remotely, allowing for greater flexibility and scalability in operations.
- **Safety and Security-** IoT enables enhanced safety and security measures through real-time monitoring, predictive maintenance, and rapid response to emergencies or security threats.
- **Environmental Benefits-** IoT can contribute to environmental sustainability by optimizing resource usage, reducing waste, and enabling more efficient energy management.
- **Improved Healthcare-** IoT applications in healthcare facilitate remote patient monitoring, telemedicine, and personalized treatment plans, leading to better healthcare outcomes and cost savings for patients and healthcare providers.

3.1.3 IOT -Disadvantages

- **Security Concerns-** IoT devices are often vulnerable to cyber attacks due to their interconnected nature and the vast amounts of sensitive data they collect and transmit. Weak security measures can lead to breaches, data theft, and privacy violations.
- **Privacy Risks-**The extensive data collection capabilities of IoT devices raise concerns about privacy infringement. Personal and sensitive information may be collected, stored, and shared without adequate consent or protection, leading to privacy breaches and identity theft.
- **Interoperability Issues-** The lack of standardized protocols and compatibility between different IoT devices and platforms can hinder interoperability and integration, leading to fragmentation and inefficiencies within IoT ecosystems.
- **Complexity and Management Challenges-** Managing and maintaining large-scale IoT deployments can be complex and resource-intensive. Issues such as device provisioning, firmware updates, and lifecycle management may pose challenges for organizations, particularly those with limited technical expertise.
- **Reliability and Stability-** IoT systems may experience reliability issues such as network outages, device failures, or software glitches, leading to disruptions in service and potential safety hazards, especially in critical applications such as healthcare or transportation.

3.1.4 IOT Software

There are various types of software used in IoT systems, each serving different purposes. Here are some common categories of IoT software:

- **Embedded Software-**This type of software runs directly on IoT devices and is responsible for controlling device operations, managing hardware resources, and facilitating communication with other devices or the cloud.
- **IoT Platforms-** IoT platforms provide the infrastructure and tools needed to develop, deploy, and manage IoT solutions. These platforms typically offer features such as device management, data analytics, connectivity management, security, and integration with other systems.
- **Data Analytics and Visualization Tools-** IoT generates vast amounts of data that need to be analyzed to derive actionable insights. Data analytics and visualization tools help process and interpret IoT data, enabling users to monitor performance, detect anomalies, and make data-driven decisions.

3.1.5 IOT -Technology and Protocol

IoT relies on a variety of technologies and protocols to enable communication and data exchange between devices and systems. Here are some of the key technologies and protocols commonly used in IoT:

Wireless Communication Technologies

Wi-Fi- Provides high-speed wireless networking over short to medium distances, suitable for applications where power consumption is not a critical concern.

Bluetooth- Enables short-range communication between devices, often used in wearable devices, smart home appliances, and proximity-based applications.

Zigbee- Low-power, low-data-rate wireless communication protocol ideal for building automation, smart lighting, and sensor networks.

Z-Wave- Proprietary wireless protocol optimized for home automation, offering low power consumption and mesh networking capabilities.

LoRaWAN- Long-range, low-power wide-area network (LPWAN) technology suitable for applications requiring long-range communication with low data rates, such as smart agriculture and asset tracking.

Cellular (4G LTE, 5G)- Cellular networks provide ubiquitous coverage and high-speed data transmission, making them suitable for IoT applications requiring wide-area connectivity and mobility.

Protocols

MQTT (Message Queuing Telemetry Transport)-Lightweight, publish-subscribe messaging protocol designed for IoT applications, offering efficient communication over unreliable networks and low bandwidth.

CoAP (Constrained Application Protocol)- RESTful protocol optimized for constrained devices and low-power networks, suitable for IoT applications where HTTP is impractical.

HTTP/HTTPS- Standard protocols for web communication, often used in IoT applications for device management, data exchange, and integration with web services.

AMQP (Advanced Message Queuing Protocol)- Open-standard messaging protocol for asynchronous communication, commonly used in IoT platforms for reliable message delivery and interoperability.

DDS (Data Distribution Service)- Publish-subscribe middleware protocol for real-time data sharing and communication between devices and systems, suitable for mission-critical IoT applications.

Modbus- Serial communication protocol commonly used in industrial automation and control systems, enabling communication between IoT devices and PLCs (Programmable Logic Controllers).

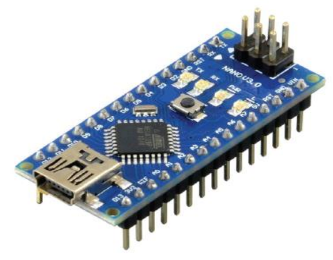
3.1.6 Internet of Things – Common Uses

IoT has applications across all industries and markets.it spans user group from those who want to reduce energy use in their home to large organizations who want to streamline their operations.it proves not just useful, but nearly critical in many industries as technology advances and we move towards the advanced automation imagined in the distant future.

- **Smart Home Automation-** IoT devices enable homeowners to automate and control various aspects of their homes, such as lighting, heating, air conditioning, security cameras, door locks, and appliances, through smartphone apps or voice commands.
- **Industrial Automation and Manufacturing-** IoT solutions optimize industrial processes by monitoring equipment performance, tracking inventory, predicting maintenance needs, and enabling autonomous operation of machinery. This leads to increased productivity, reduced downtime, and cost savings for manufacturers.
- **Smart Energy Management-**IoT devices and sensors monitor energy consumption in buildings, factories, and utilities, enabling efficient energy management, demand response, and optimization of energy usage to reduce costs and environmental impact.
- **Healthcare Monitoring and Telemedicine-** IoT enables remote patient monitoring, wearable health devices, and telemedicine platforms that allow healthcare providers to monitor patient health status, track vital signs, and deliver personalized care outside traditional healthcare settings.
- **Smart Cities and Urban Infrastructure-**IoT technology is used to improve urban infrastructure and services, including traffic management, public transportation, waste management, water and energy distribution, environmental monitoring, and emergency response systems, leading to safer, more efficient cities.

3.2 Arduino NANO

The Arduino Nano is an open-source breadboard-friendly microcontroller board based on the Microchip ATmega328P microcontroller (MCU) and developed by Arduino.cc and initially released in 2008. It offers the same connectivity and specs of the Arduino Uno board in a smaller form factor.



3.2.1 Pin Configuration of A

Figure 1 Arduino Nano

The Arduino Nano is equipped with 30 male I/O headers, in a DIP-30-like configuration, which can be programmed using the Arduino Software integrated development environment (IDE), which is common to all Arduino boards and running both online and offline. The board can be powered through a type-B mini-USB cable or from a 9 V battery

3.2.2 Installation and Coding of Arduino Nano

Step 1: Install Arduino IDE

Arduino IDE is used to write and upload sketch to the Arduino board.

Figure 2 Arduino IDE

Step 2: Required Compo

Arduino Nano

Mini USB

Step 3: Select the Board that is used

Open Arduino IDE > Tools.

Board: "Arduino Nano"

Processor: "Atmega 328P (Old Bootloader)" ===> if an error occurs, select another option.

Port: "COM4" ===> according to the USB port you are using

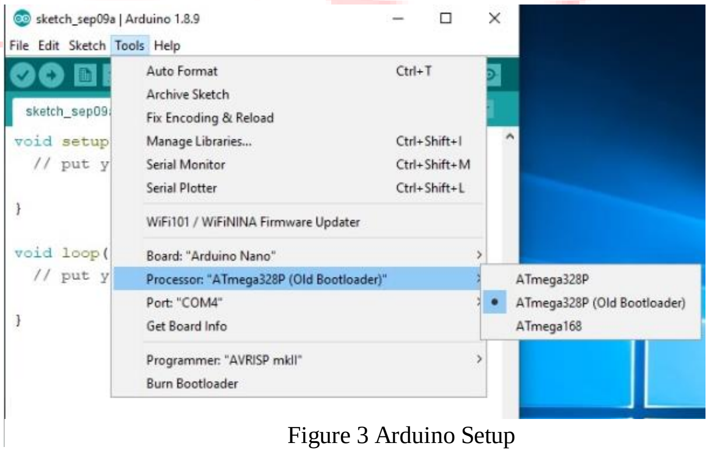

sketch_sep09a | Arduino 1.8.9
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Nano"
Processor: "ATmega328P (Old Bootloader)"
Port: "COM4"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
ATmega328P
ATmega328P (Old Bootloader)
ATmega168

Figure 3 Arduino Setup

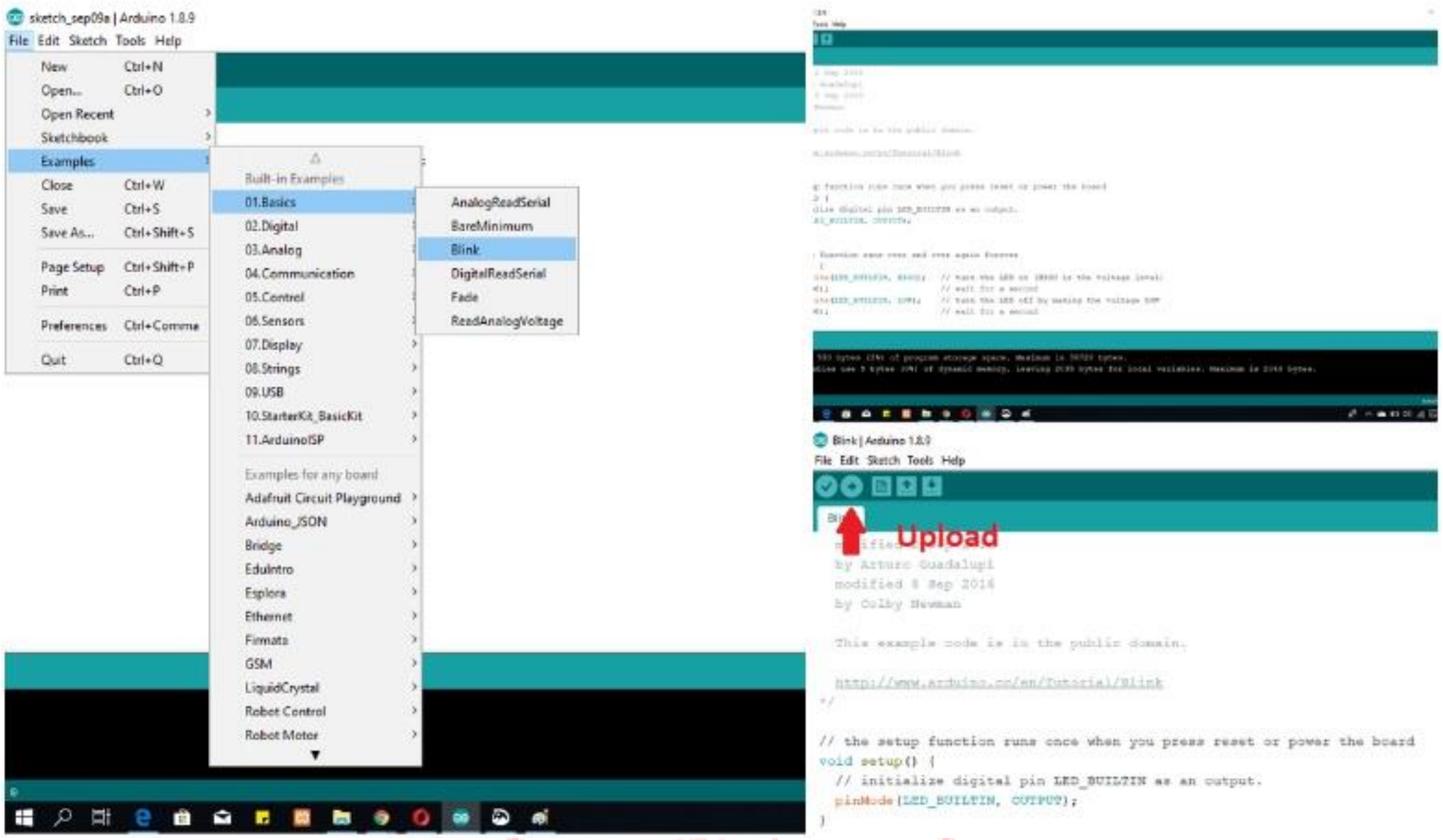
Step 4: Open and Upload Sketch

Open Sketch

Open the LED blink example sketch: File > Examples > 01.Basics > Blink.

Upload Sketch

To upload the program. Click the upload button. Wait for a moment - During the upload process, the RX and TX LEDs will flashing. If the upload is successful, the message "Done uploading" will appear in the status bar and the result is a red LED on Arduino will blink.



```
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
```

Figure 4 Arduino Code Upload

CHAPTER – 4

CHAPTER – 4

IMPLEMENTATION AND RESULT

4.1 PROGRAMMING

Source Code For Arduino

```
#include "SoftwareSerial.h"
#include "MHZ19.h"
#include "PMS.h"
#include "MQ131.h"
#include "dht.h"
#include "DS3231.h"


#define led 13
#define tvocPin 7  // VOC sensor activation
#define dht22 5 // DHT22 temperature and humidity sensor

dht DHT; // Creats a DHT object
DS3231  rtc(SDA, SCL); // Initiate the DS3231 Real Time Clock module using the I2C interface
Time  t; // Init a Time-data structure
MHZ19 myMHZ19;    // CO2 Sensor
SoftwareSerial co2Serial(2, 3);  // (RX, TX) MH-Z19 serial
SoftwareSerial pmsSerial(8, 9); // Particulate Matter sensor
PMS pms(pmsSerial);
PMS::DATA data;

unsigned long dataTimer = 0;
unsigned long dataTimer3 = 0;
unsigned long dataTimer4 = 0;
int readDHT, temp, hum;
int CO2;
int o3;
int tvoc;
int pm25;
int hours, minutes;
int previousMinutes = 1;
String timeString;
String receivedData = "Z";
uint8_t tempData[96] = {};
uint8_t humData[96] = {};
uint8_t tvocData[96] = {};
uint8_t co2Data[96] = {};
uint8_t pm25Data[96] = {};
uint8_t o3Data[96] = {};

int8_t last24Hours[12] = {};
```

```
int yAxisValues[4] = {};
int maxV = 0;
int8_t r = 99;

void setup() {
  Serial.begin(9600);
  // Device to serial monitor feedback
  pinMode(6, OUTPUT);
  pinMode(tvocPin, OUTPUT);

  // Warming up sensors
  digitalWrite(6, HIGH);        // Ozone sensor
  digitalWrite(tvocPin, HIGH);  // TVOC sensor
  delay(20 * 1000); // delay 20 seconds
  digitalWrite(6, LOW);
  digitalWrite(tvocPin, LOW);

  // Initialize all sensors
  rtc.begin();
  co2Serial.begin(9600);
  pmsSerial.begin(9600);
  myMHZ19.begin(co2Serial);
  myMHZ19.autoCalibration(false);  // Turn auto calibration ON (OFF autoCalibration(false))
  MQ131.begin(6, A0, LOW_CONCENTRATION, 1000000); //
  MQ131.setTimeToRead(20);
  MQ131.setR0(9000);
}

void loop() {
  readDHT = DHT.read22(dht22); // Reads the data from the sensor
  temp = DHT.temperature; // Gets the values of the temperature
  hum = DHT.humidity; // Gets the values of the humidity

  // Read TVOC for 5 seconds
  digitalWrite(tvocPin, HIGH);
  delay(5000);
  tvoc = analogRead(A1);
  digitalWrite(tvocPin, LOW);

  checkForIncomingData();

  co2Serial.listen();
  dataTimer = millis();
  while (millis() - dataTimer <= 3000) {
    CO2 = myMHZ19.getCO2(); // Request CO2 (as ppm)
  }

  checkForIncomingData();
```

```
pmsSerial.listen();
dataTimer3 = millis();
while (millis() - dataTimer3 <= 1000) {
  pms.readUntil(data);
  pm25 = data.PM_AE_UG_2_5;
}
checkForIncomingData();

// Read MQ131 Ozone sensor
MQ131.sample();
o3 = MQ131.getO3(PPB);

checkForIncomingData();
t = rtc.getTime();
hours = t.hour;
minutes = t.min;
// Store current sensors data
storeData();

// Send the data to the Nextion display
dataTimer4 = millis();
while (millis() - dataTimer4 <= 200) {
  Serial.print("co2V.val=");
  Serial.print(CO2);
  Serial.write(0xff);
  Serial.write(0xff);
  Serial.write(0xff);

  Serial.print("pm25V.val=");
  Serial.print(pm25);
  Serial.write(0xff);
  Serial.write(0xff);
  Serial.write(0xff);

  Serial.print("o3V.val=");
  Serial.print(o3);
  Serial.write(0xff);
  Serial.write(0xff);
  Serial.write(0xff);

  Serial.print("tempV.val=");
  Serial.print(temp);
  Serial.write(0xff);
  Serial.write(0xff);
  Serial.write(0xff);

  Serial.print("humV.val=");
  Serial.print(hum);
  Serial.write(0xff);
  Serial.write(0xff);
```

```
    Serial.write(0xff);

    Serial.print("tvocV.val=");
    Serial.print(tvoc);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
  }
}

void checkForIncomingData() {
  // Check if data is coming from the Nextion
  if (Serial.available() > 0) {
    receivedData = Serial.readString();
    delay(30);
    if (receivedData == "0") {
      r = 0;
    }
    if (receivedData == "1") {
      r = 1;
    }
    if (receivedData == "2") {
      r = 2;
    }
    if (receivedData == "3") {
      r = 3;
    }
    if (receivedData == "4") {
      r = 4;
    }
  }
    if (r == 0 || r == 1 || r == 2 || r == 3 || r == 4) {
    delay(200);
    dataTimer3 = millis();
    while (millis() - dataTimer3 <= 200) {
      Serial.print("pageSwitch.val="); // Activate page 1, or the waveform on the Nextion display
      Serial.print(1);
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
    delay(100);
    getLast24Hours(); // get the last 24 hours and print them on as X-axis values on the waveform
    getYAxisValues(); // get the Y-axis values according to the sensor, it's range and it's max
value. Print the Y-axis values as well as scale the Y-axis of the wavefrom accordingly
    sendDataToWaveform(); // send the stored data of the last 24 hours to the waveform
    r = 99; // reset the "r" to 99(arbitrary number, different than the ones we assign when we
receive data depending on which sensor we have pressed)
  }
}
```

```
void storeData() {
  // Storing current sensor values into arrays
  if ((minutes - previousMinutes) >= 15) {  // store the value each 15 minutes
    memmove(tempData, &tempData[1], sizeof(tempData)); // Slide data down one position
    tempData[sizeof(tempData) - 1] = temp; // store newest value to last position
    memmove(humData, &humData[1], sizeof(humData));
    humData[sizeof(humData) - 1] = hum;
    memmove(tvocData, &tvocData[1], sizeof(tvocData));
        tvocData[sizeof(tvocData) - 1] = map(tvoc, 0, 1000, 0, 255);
    memmove(co2Data, &co2Data[1], sizeof(co2Data));
    co2Data[sizeof(co2Data) - 1] = map(CO2, 0, 3000, 0, 255);
    memmove(pm25Data, &pm25Data[1], sizeof(pm25Data));
    pm25Data[sizeof(pm25Data) - 1] = map(pm25, 0, 1000, 0, 255);
    memmove(o3Data, &o3Data[1], sizeof(o3Data));
    o3Data[sizeof(o3Data) - 1] = map(o3, 0, 1000, 0, 255);
    previousMinutes = minutes;

  }
    else if ((minutes - previousMinutes) == -45) { // when minutes start from 0, next hour
    memmove(tempData, &tempData[1], sizeof(tempData)); // Slide data down one position
    tempData[sizeof(tempData) - 1] = temp; // store newest value to last position
    memmove(humData, &humData[1], sizeof(humData));
    humData[sizeof(humData) - 1] = hum;
    memmove(tvocData, &tvocData[1], sizeof(tvocData));
    tvocData[sizeof(tvocData) - 1] = map(tvoc, 0, 1000, 0, 255);
    memmove(co2Data, &co2Data[1], sizeof(co2Data));
    co2Data[sizeof(co2Data) - 1] = map(CO2, 0, 3000, 0, 255);
    memmove(pm25Data, &pm25Data[1], sizeof(pm25Data));
    pm25Data[sizeof(pm25Data) - 1] = map(pm25, 0, 1000, 0, 255);
    memmove(o3Data, &o3Data[1], sizeof(o3Data));
    o3Data[sizeof(o3Data) - 1] = map(o3, 0, 1000, 0, 255);
    previousMinutes = minutes;
  }
}
void getLast24Hours() {
  for (int i = 11; i >= 0; i--) {
    last24Hours[11] = hours; // get the current hour - according to this hour get 12 more hours,
each 2 hours. For example, current hour = 13, so 11, 9, 7...
    last24Hours[i - 1] = last24Hours[i] - 2;
    if (last24Hours[i - 1] < 0) {
      for (int k = -0; k > -11; k--) {
        if (last24Hours[i - 1] == k) {
          last24Hours[i - 1] = 24 + k;
        }
      }
    }
  }
  // send the hours values to the Nextion display
  for (int i = 0; i < 12; i++) {
    String last24 = ("n") + String(i) + String(".val=") + String(last24Hours[i]); // e.g. for i=0 > "n0.val="
    Serial.print(last24);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
    delay(20);
  }
  // Another write just to make sure it sends all data
  for (int i = 0; i < 12; i++) {
    String last24 = ("n") + String(i) + String(".val=") + String(last24Hours[i]); // e.g. for i=0 > "n0.val="
    Serial.print(last24);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
    delay(20);
  }
}
void getYAxisValues() {
  maxV = 0;
  // PM2.5 Y-axis values
  if (r == 0) {
    // Get the max sensor value from the last 24 hours
    for (int i = 0; i < sizeof(pm25Data); i++) {
      if (maxV < map(pm25Data[i], 0, 255, 0, 1000)) {
        maxV = map(pm25Data[i], 0, 255, 0, 1000);
      }
    }
    // Setting the Y-axis values and scaling the waveform
    if (maxV <= 100) {
      yAxisValues[0] = 25;
      yAxisValues[1] = 50;
      yAxisValues[2] = 75;
      yAxisValues[3] = 100;
      Serial.print("s0.dis="); // this command ".dis" is used for scalign the Y-axis
      Serial.print(78 * 10);  // scale the waveform Y-axis - pm2.5 values are from 0 to 1000 which are represented from 0 to 78% in the Y axis - 78% because the wavefore is 200px which is 78% of 255 which is the default 100% value of the waveform - 78*10 because we show values from 0 to 100, which are 10 times smaller
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
    // if the value is higher than 100, get its max value, and according to it scale the y-axis - For example, if the max value is 235, set the max value of the Y-axis to 300 - 235/100=2+1=3*100=300
    else if (maxV > 100) {
      int l = ((maxV / 100) + 1) * 100;  // get the hundreds value so we can properly scale the Y axis of the waveform
      yAxisValues[0] = l / 4;
      yAxisValues[1] = l / 2;
      yAxisValues[2] = l * 3 / 4;
      yAxisValues[3] = l;
      float ll = 78.0 / (l / 1000.0); // scale value for the Y-axis in % - We multiply by 78 instead
of 100 because our waveform is 200px in height, which is 78% of 255 (255 is max value the
waveform can accept, 1 byte)
      Serial.print("s0.dis=");
      Serial.print(round(ll));
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
  }
  // TVOC Y-axis values
  if (r == 2) {
    // Get the max sensor value from the last 24 hours
    for (int i = 0; i < sizeof(tvocData); i++) {
      if (maxV < map(tvocData[i], 0, 255, 0, 1000)) {
        maxV = map(tvocData[i], 0, 255, 0, 1000);
      }
    }
    // Setting the Y-axis values and scaling the waveform
    if (maxV <= 100) {
      yAxisValues[0] = 25;
      yAxisValues[1] = 50;
      yAxisValues[2] = 75;
      yAxisValues[3] = 100;
      Serial.print("s0.dis=");
      Serial.print(78 * 10);
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
    else if (maxV > 100) {
      int l = ((maxV / 100) + 1) * 100;  // get the hundreds value so we can properly scale the Y
axis of the waveform
      yAxisValues[0] = l / 4;
      yAxisValues[1] = l / 2;
      yAxisValues[2] = l * 3 / 4;
      yAxisValues[3] = l;
      float ll = 78.0 / (l / 1000.0); // scale value for the Y-axis in % - We multiply by 78 instead
of 100 because our waveform is 200px in height, which is 78% of 255 (255 is max value the
waveform can accept, 1 byte)
      Serial.print("s0.dis=");
      Serial.print(round(ll));
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
  }
  // Ozone Y-axis values
  if (r == 3) {
    // Get the max sensor value from the last 24 hours
    for (int i = 0; i < sizeof(o3Data); i++) {
      if (maxV < map(o3Data[i], 0, 255, 0, 1000)) {
        maxV = map(o3Data[i], 0, 255, 0, 1000);
      }
    }
    // Setting the Y-axis values and scaling the waveform
    if (maxV <= 100) {
      yAxisValues[0] = 25;
      yAxisValues[1] = 50;
      yAxisValues[2] = 75;
      yAxisValues[3] = 100;
      Serial.print("s0.dis=");
      Serial.print(78 * 10);
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
    else if (maxV > 100) {
      int l = ((maxV / 100) + 1) * 100;  // get the hundreds value so we can properly scale the Y
axis of the waveform
      yAxisValues[0] = l / 4;
      yAxisValues[1] = l / 2;
      yAxisValues[2] = l * 3 / 4;
      yAxisValues[3] = l;
      float ll = 78.0 / (l / 1000.0); // scale value for the Y-axis in % - We multiply by 78 instead
of 100 because our waveform is 200px in height, which is 78% of 255 (255 is max value the
waveform can accept, 1 byte)
      Serial.print("s0.dis=");
      Serial.print(round(ll));
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
  }
  // CO2 Y-axis values are fixed from 0 to 3000 so we don't need to look for the max value in
the array
  if (r == 1) {
    // Get the max sensor value from the last 24 hours
    for (int i = 0; i < sizeof(co2Data); i++) {
      if (maxV < map(co2Data[i], 0, 255, 0, 3000)) {
        maxV = map(co2Data[i], 0, 255, 0, 3000);
      }
    }
    if (maxV <= 2000) {
      // Setting the Y-axis values and scaling the waveform
      yAxisValues[0] = 500;
      yAxisValues[1] = 1000;
      yAxisValues[2] = 1500;
      yAxisValues[3] = 2000;
      Serial.print("s0.dis=");
      Serial.print(117);  // scale the waveform from 0 - 3000 to 0 - 2000 range
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
    if (maxV > 2000) {
      // Setting the Y-axis values and scaling the waveform
      yAxisValues[0] = 750;
      yAxisValues[1] = 1500;
      yAxisValues[2] = 2250;
      yAxisValues[3] = 3000;
      Serial.print("s0.dis=");
      Serial.print(78);
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
  }
  // Temperature and Humidity Y-axis values - fixed from 0 to 100
  if (r == 4) {
    // Setting the Y-axis values and scaling the waveform
    yAxisValues[0] = 25;
    yAxisValues[1] = 50;
    yAxisValues[2] = 75;
    yAxisValues[3] = 100;
    Serial.print("s0.dis=");
    Serial.print(200); // from 0 to 100 - 255/100 * 78 = ~200
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
  }
  delay(50);
  // Send the  Y-axis values to the Nextion display
  for (int i = 0; i < 4; i++) {
    String yValues = ("y") + String(i) + String(".val=") + String(yAxisValues[i]); // e.g. for i=0
> "y0.val="
    Serial.print(yValues);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
    delay(10);
  }
}

void sendDataToWaveform() {
  int k = 0;
```

```
 while (k != 2) {
   String str = String("addt 1,0,") + String(288); // with this command we tell the nextion
display that we will send an array of data to the waveform
   Serial.print(str);
   delay(100);
   Serial.write(0xFF);
   Serial.write(0xFF);
   Serial.write(0xFF);
   delay(100);
   // Now depending on the selected sensor we want the values stored in the arrays
   // PM2.5
   if (r == 0) {
     for (int t = 0; t < sizeof(pm25Data); t++) {
       int z = 0;
       while (z != 3) {
         Serial.write(pm25Data[t]);
         z++;
       }
     }
   }
   // CO2
   if (r == 1) {
     for (int t = 0; t < sizeof(co2Data); t++) {
       int z = 0;
       while (z != 3) {
         Serial.write(co2Data[t]);
         z++;
       }
     }
   }
   // TVOC
   if (r == 2) {
     for (int t = 0; t < sizeof(tvocData); t++) {
       int z = 0;
       while (z != 3) {
         Serial.write(tvocData[t]);
         z++;
       }
     }
   }
   // Ozone
   if (r == 3) {
     // Temperature values on channel 0
     for (int t = 0; t < sizeof(o3Data); t++) {
       int z = 0;
       while (z != 3) {
         Serial.write(o3Data[t]);
         z++;
       }
     }
```

```
    }
    // Temp and hum
    if (r == 4) {
      for (int t = 0; t < sizeof(humData); t++) {
        int z = 0;
        while (z != 3) {
          Serial.write(humData[t]);
          z++;
        }
      }
      delay(100);
      Serial.write(0xFF);
      Serial.write(0xFF);
      Serial.write(0xFF);
      delay(100);
      // Humidity values on channel 1
      String str = String("addt 1,1,") + String(288);
      Serial.print(str);
      delay(100);
      Serial.write(0xFF);
      Serial.write(0xFF);
      Serial.write(0xFF);
      delay(100);
      for (int t = 0; t < sizeof(tempData); t++) {
        int z = 0;
        while (z != 3) {
          Serial.write(tempData[t]);
          z++;
        }
      }
      delay(100);
      Serial.write(0xFF);
      Serial.write(0xFF);
      Serial.write(0xFF);
    }
    k++;
  }
}
```

4.2 WORKING OF CIRCUIT

Microcontroller Unit (MCU): The core of this circuit is a microcontroller unit (MCU) such as Arduino Uno or ESP32. This MCU will have the following responsibilities: - Powering the Sensors: The MCU will supply regulated voltage to each sensor based on its specific requirements. - Data Acquisition: The MCU will communicate with each sensor using the appropriate protocol. This may involve I2C for PMS7003, MH-Z19, and DS3231, SPI for the display, and analog reads for the MQ-131 and MP503 (based on the model). - Data Processing:

The raw data from the sensors must be converted into meaningful units (e.g., ppm for CO2, µg/m³ for PM). The MCU may utilize calibration factors and specific calculations for each sensor. - Real-Time Clock (RTC): The DS3231 provides real-time functionality, enabling the MCU to timestamp sensor readings for data logging purposes. - Display Control: The MCU will communicate with the SPI display to present the processed sensor data in a user-friendly format (e.g., numerical values, graphs).

Sensor Connections:

- PMS7003: This sensor typically uses I2C communication. The MCU will need to be programmed to send commands to the sensor and receive data packets containing PM concentrations in various size ranges.
- MH-Z19: Similar to the PMS7003, the MH-Z19 also uses I2C communication. The MCU will interact with the sensor to retrieve CO2 concentration readings.
- MQ-131: This sensor operates by varying its resistance based on the ozone concentration in the surrounding air. The MCU will read the analog voltage on the sensor's output pin and convert it to an estimated ozone concentration using a pre-determined calibration curve.
- MP503: Depending on the specific model, the MP503 might use analog or digital output. In the case of analog output, the MCU will read the voltage and convert it to an estimated VOC concentration using a calibration curve. Some models might have a digital output indicating VOC presence/absence.
- DHT22: This digital sensor communicates using a single data pin. The MCU will initiate communication and read the temperature and humidity data transmitted by the DHT22.

- DS3231: This RTC module often uses I2C communication. The MCU can interact with the DS3231 to set the time and access real-time date and time information for data logging purposes.

SPI Display:

The SPI display connects to the MCU using a dedicated SPI communication protocol. The MCU sends commands and data to the display to control the information displayed, such as sensor readings and timestamps.

Power Supply:

The circuit will require a regulated power supply to provide the necessary voltage to the MCU and each sensor according to their individual specifications. A common approach is to use a USB power bank or a wall adapter with a voltage regulator circuit.

4.3 BLOCK DIAGRAM

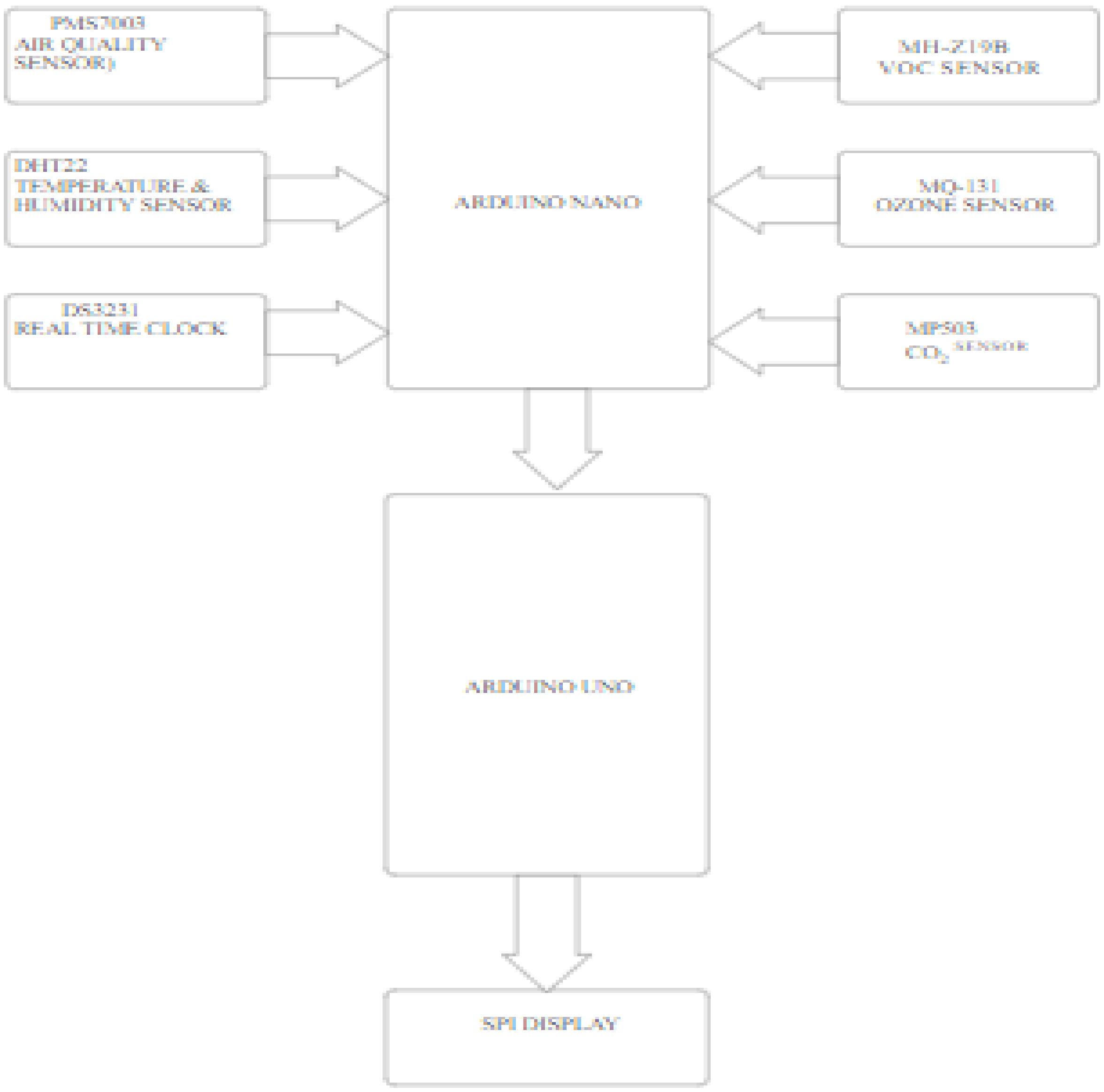


Figure 5 Block Diagram

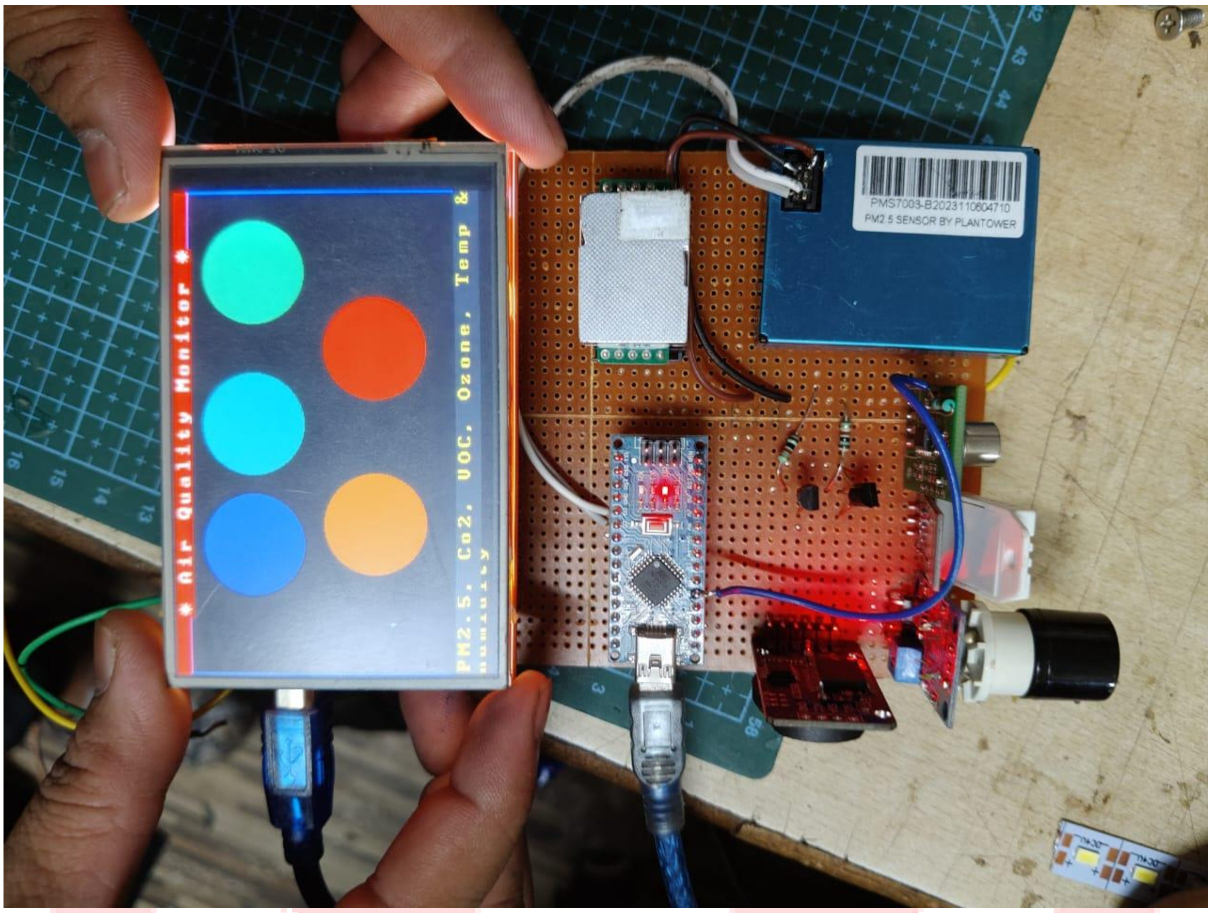

* Air Quality Monitor *
PM2.5, Co2, VOC, Ozone, Temp & Humidity
PMS7003-B2023110604710
PM2.5 SENSOR BY PLANTOWER

Figure 6 Images of Prototypes

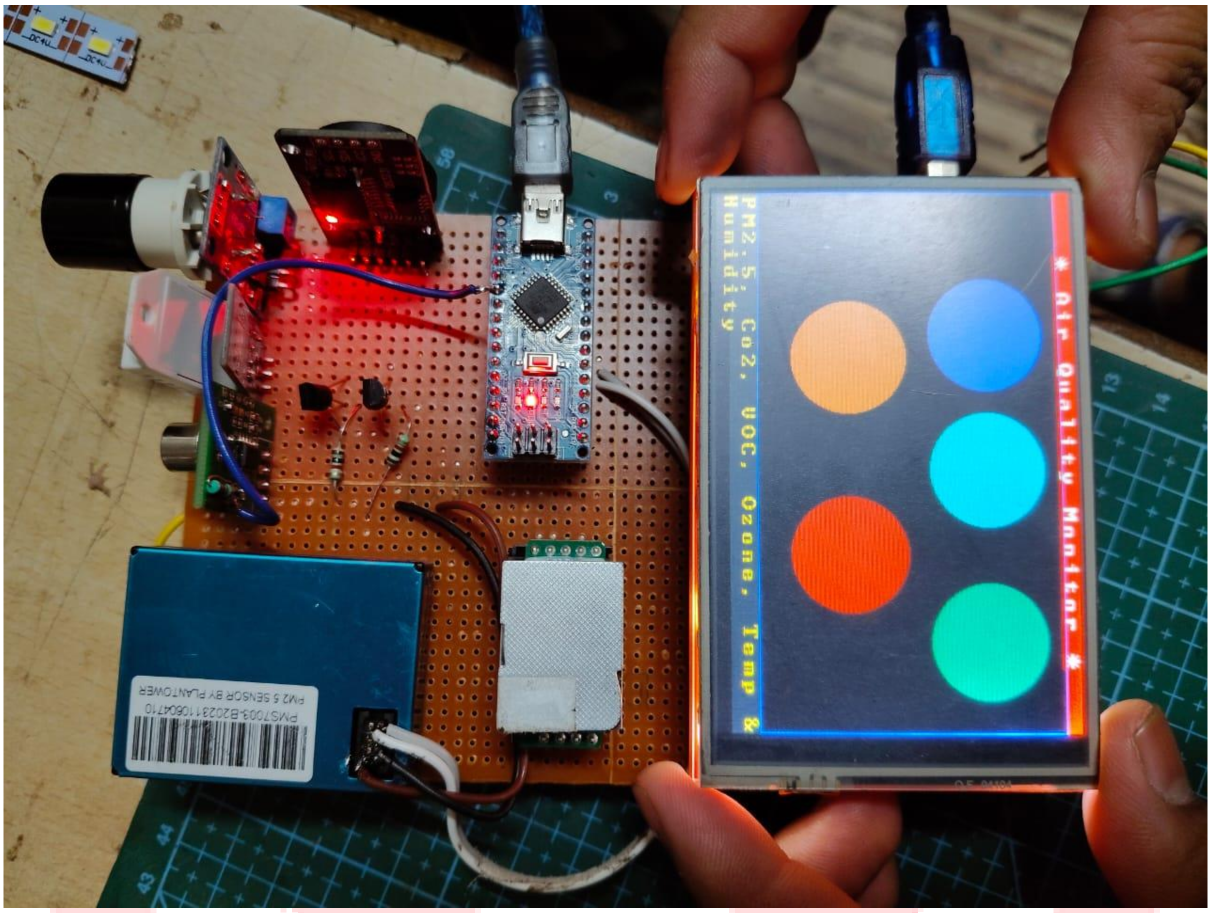


Figure 7 Images of prototypes

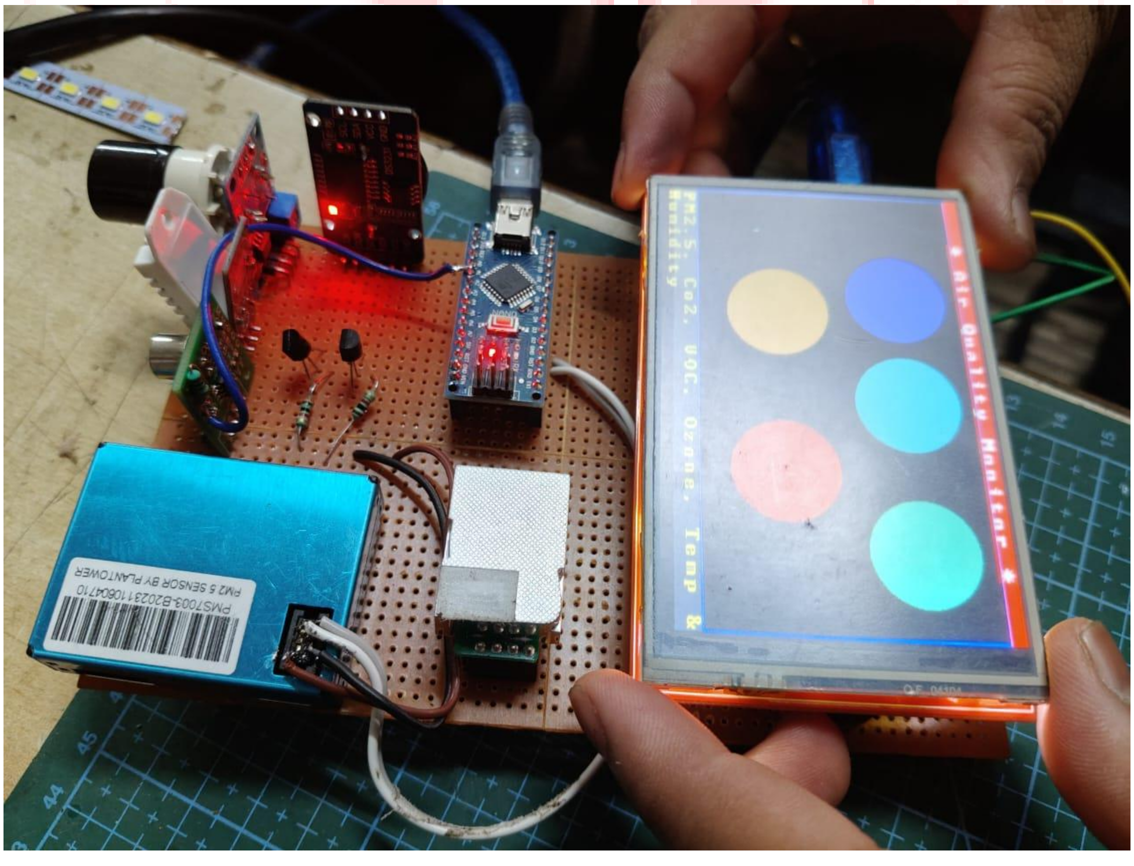


Figure 8 Images of Prototypes

4.4 RESULT

Air pollution is a global concern that affects human health, the environment, and weather patterns. To effectively address this issue, we need a way to measure and track air pollution. Air pollution monitoring systems (APMS) are designed for this purpose.

An APMS can be a network of devices or a single, advanced instrument that detects and measures various pollutants in the air. These systems are essential for:

Understanding Air Quality: APMS continuously monitor pollutant levels, providing valuable data on air quality in specific locations. This data helps identify trends, track pollution sources, and assess the effectiveness of air quality control measures.

Protecting Public Health: APMS can trigger alerts when pollutant concentrations exceed safe limits, prompting individuals, especially vulnerable populations like children and the elderly, to take precautions such as staying indoors or wearing masks.

Environmental Monitoring: APMS data is crucial for environmental research and policy development aimed at protecting ecosystems and addressing climate change.

There are two main categories of APMS:

Continuous Monitoring Systems: These systems collect real-time data on pollutant levels using various sensors to detect pollutants like particulate matter (PM2.5 and PM10), ozone (O3), carbon monoxide (CO), sulfur dioxide (SO2), nitrogen dioxide (NO2), and volatile organic compounds (VOCs). The data is transmitted wirelessly to a central server for processing, analysis, and dissemination, enabling immediate response to pollution spikes.

Passive Monitoring Systems: These systems collect air samples over a specific period for analysis in a laboratory to determine pollutant concentrations. While less expensive and easier to deploy, they do not provide real-time data like continuous systems.

CHAPTER – 5

CHAPTER – 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

Air pollution monitoring devices play a crucial role in addressing the global challenge of air quality deterioration. Equipped with advanced sensors and real-time data processing capabilities, these devices provide vital information about the levels and sources of various pollutants in the atmosphere. By offering precise and timely data, they enable governments, environmental agencies, and researchers to make informed decisions and implement effective strategies for pollution control.

The integration of air pollution monitoring devices into urban planning and public health initiatives has already shown significant benefits. They help identify pollution hotspots, track trends over time, and assess the effectiveness of regulatory measures. Additionally, these devices empower communities by providing accessible information about local air quality, encouraging public awareness and engagement in environmental conservation efforts.

Technological advancements continue to enhance the functionality and accessibility of these devices, making them more affordable and user-friendly. The development of portable and wearable monitors further democratizes air quality monitoring, allowing individuals to take proactive measures to protect their health.

In conclusion, air pollution monitoring devices are indispensable tools in the fight against air pollution. They not only facilitate the collection of critical data, but also drive policy-making and community actions aimed at reducing pollution and mitigating its impacts on health and the environment. As technology continues to evolve, these devices will become even more integral to our efforts to create cleaner, healthier, and more sustainable living environments.

5.2 FUTURE WORK

As the urgency to address air pollution intensifies, future work in air pollution monitoring will be pivotal in developing comprehensive strategies to mitigate this global challenge. Several key areas of research and development will shape the future landscape of air pollution monitoring, driving improvements in technology, methodology, and policy integration.

1. Development of Next-Generation Sensors

In the future, our focus will be on developing advanced air pollution sensors that are more sensitive, accurate, and capable of detecting a wider range of pollutants at lower concentrations. This will involve:

1. Nanotechnology and Advanced Materials: Research into nanomaterials and other advanced materials will be conducted to produce sensors with enhanced detection capabilities. These materials can help develop sensors that are not only more sensitive but also more durable and cost-effective.

2. Multi-Pollutant Detection: We will work on developing sensors capable of detecting multiple pollutants simultaneously to provide a more comprehensive picture of air quality. Future sensors will aim to measure particulate matter (PM), nitrogen oxides (NOx), sulfur dioxide (SO2), carbon monoxide (CO), volatile organic compounds (VOCs), and more, all within a single device.

3. Integration with Internet of Things (IoT): The IoT will play a significant role in future air pollution monitoring efforts, focusing on smart networks and data integration.

4. Advanced Data Analytics and Machine Learning: We will develop machine learning algorithms to analyze air quality data, predict pollution trends, and identify sources of pollution. Additionally, big data techniques will be employed to process and analyze the vast datasets generated by extensive monitoring networks.

5. Satellite and Drone Technology: Satellite monitoring will be enhanced to provide higher resolution and more frequent data on air quality at a regional and global scale. Additionally,

drones equipped with advanced sensors will be developed to monitor air quality in inaccessible or hazardous areas.5. Citizen Science and Public Engagement

Future efforts will focus on engaging the public in air pollution monitoring through citizen science initiatives:

- Affordable Devices: Designing low-cost, user-friendly air quality monitors that individuals can use in their homes and communities. This democratizes data collection and raises public awareness about local air quality issues.
- Crowdsourced Data Platforms: Creating platforms for the collection, analysis, and sharing of crowdsourced air quality data. These platforms will enhance community engagement and provide valuable data to supplement official monitoring networks.

The future of air pollution monitoring will involve critical areas of research and development such as linking air quality data with health outcomes, conducting large-scale epidemiological studies, and developing robust regulatory and policy frameworks. This will require integrating air pollution data with health records and collaborating with healthcare providers and public health organizations. Additionally, it will be essential to establish international standards for air quality monitoring and data reporting, integrate air quality data into policy-making processes at different levels, and address disparities in air quality monitoring and management, particularly in underserved communities. Inclusive public engagement will also be important in the policy-making process to ensure that the needs and concerns of all populations are addressed. The advancement of technology and public awareness will make air pollution monitoring an even more powerful tool in our efforts to protect public health and the environment.

Source Code For Arduino

```
#include "SoftwareSerial.h"
#include "MHZ19.h"   // https://github.com/WifWaf/MH-Z19
#include "PMS.h"     //https://github.com/fu-hsi/pms
#include "MQ131.h"   // https://github.com/ostaquet/Arduino-MQ131-driver
#include "dht.h"     // https://github.com/RobTillaart/DHTlib
#include "DS3231.h"  // http://www.rinkydinkelectronics.com/library.php?id=73

/*
 * Pleas note that the code can be a bit complex. For each sensor module I'm
using libraries which can be found in the links above. So in order to
understand how everything works,
 * you must take a look at each library how it works by reading its
instructions.
 * For calibrating the sensors also use the libraries documentation
 * Also note that the code is not very well optimized.
 */

#define led 13
#define tvocPin 7  // VOC sensor activation
#define dht22 5 // DHT22 temperature and humidity sensor

dht DHT; // Creats a DHT object
DS3231  rtc(SDA, SCL); // Initiate the DS3231 Real Time Clock module using
the I2C interface
Time  t; // Init a Time-data structure
MHZ19 myMHZ19;    // CO2 Sensor
SoftwareSerial co2Serial(2, 3);  // (RX, TX) MH-Z19 serial
SoftwareSerial pmsSerial(8, 9); // Particulate Matter sensor
PMS pms(pmsSerial);
PMS::DATA data;

unsigned long dataTimer = 0;
unsigned long dataTimer3 = 0;
unsigned long dataTimer4 = 0;
int readDHT, temp, hum;
int CO2;
int o3;
int tvoc;
int pm25;
int hours, minutes;
int previousMinutes = 1;
String timeString;
String receivedData = "Z";
// We store the last 24 hours sensor values  in arrays - store value each 15
minutes so for 24 hours we need 96 bytes.
// We must use bytes and can't increse the storing to let's say 5 mins
because the Arduino Pro Mini has a limited dynamic memory
uint8_t tempData[96] = {};
uint8_t humData[96] = {};
uint8_t tvocData[96] = {};
uint8_t co2Data[96] = {};
uint8_t pm25Data[96] = {};
```

```
uint8_t o3Data[96] = {};

int8_t last24Hours[12] = {};
int yAxisValues[4] = {};
int maxV = 0;
int8_t r = 99;

void setup() {
  Serial.begin(9600);
  // Device to serial monitor feedback
  pinMode(6, OUTPUT);
  pinMode(tvocPin, OUTPUT);

  // Warming up sensors
  digitalWrite(6, HIGH);        // Ozone sensor
  digitalWrite(tvocPin, HIGH);  // TVOC sensor
  delay(20 * 1000); // delay 20 seconds
  digitalWrite(6, LOW);
  digitalWrite(tvocPin, LOW);

  // Initialize all sensors
  rtc.begin();
  co2Serial.begin(9600);
  pmsSerial.begin(9600);
  myMHZ19.begin(co2Serial);
  myMHZ19.autoCalibration(false);  // Turn auto calibration ON (OFF
autoCalibration(false))
  MQ131.begin(6, A0, LOW_CONCENTRATION, 1000000); //
  MQ131.setTimeToRead(20); // Set how many seconds we will read from the
Ozone sensor. It blocks flow
  MQ131.setR0(9000); // We get this value using the calirabrate() function
from the Library calibration example
}

void loop() {
  // Read temperature and humidity from DHT22 sensor
  readDHT = DHT.read22(dht22); // Reads the data from the sensor
  temp = DHT.temperature; // Gets the values of the temperature
  hum = DHT.humidity; // Gets the values of the humidity

  // Read TVOC for 5 seconds
  digitalWrite(tvocPin, HIGH);
  delay(5000); // Blocking the programm - It would be best if the sensor
heater is active all the time, we would get the most accurate values that
way. The thing is that the sensers heats up quaite a lot and messes with the
tempereture values. If better air circulation is provided to the case, that's
the way to go.
  tvoc = analogRead(A1); // Please note that we are only reading raw data
from this sensor, not ppm or ppb values. Just analog values from 0 to 1024.
Higher values means there is a presence of VOC
  digitalWrite(tvocPin, LOW);

  // Check for incoming data from the display - check whether we have licked
a partucular sensor to for reading the last 24 hours
  checkForIncomingData();
  // Read MHZ19 - CO2 sensor for 3 seconds - if we don't use a blocking
method with the while loop we won't get values from the sensor.
  co2Serial.listen();
  dataTimer = millis();
  while (millis() - dataTimer <= 3000) {
    CO2 = myMHZ19.getCO2(); // Request CO2 (as ppm)
  }

  // we check for incoming data after each operation, because operation is
blocking the program
  checkForIncomingData();

  // Read Particulate Matter sensor for 2 seconds
  pmsSerial.listen();
  dataTimer3 = millis();
  while (millis() - dataTimer3 <= 1000) {
    pms.readUntil(data);
    pm25 = data.PM_AE_UG_2_5;
  }
  checkForIncomingData();

  // Read MQ131 Ozone sensor
  MQ131.sample();
  /* This is also a blocking function which is using delay.
     When callibrating the Ozone sensor you will notice that the reading
time, in order to get stable readings is usually high, like more then a
minute.
     But if we use it here just like that everyting will be freezed for that
time. However, you can set the sampling time lower, but the output might not
be accurate in such a case.
     Also,you set the ozone sensor heater to be active all the time in order
to get more accurate results, but this may cause significant heat from the
sensor so you would need a case fan in order to accurate values from the temp
sensor.
     I suggest trying the library examples for calibrating and see what will
get you test best results. I would also suggest not including this sensor in
this project unless you really want it.
  */
  o3 = MQ131.getO3(PPB);

  checkForIncomingData();

  // Get the time from the DS3231 Real Time Clock module - For setting the
time use the library example
  t = rtc.getTime();
  hours = t.hour;
  minutes = t.min;
  // Store current sensors data
  storeData();

  // Send the data to the Nextion display
  dataTimer4 = millis();
  while (millis() - dataTimer4 <= 200) {
    Serial.print("co2V.val=");
    Serial.print(CO2);
    // each command ends with these three unique write commands in order the
data to be send to the Nextion display
```

```
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);

    Serial.print("pm25V.val=");
    Serial.print(pm25);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);

    Serial.print("o3V.val=");
    Serial.print(o3);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);

    Serial.print("tempV.val=");
    Serial.print(temp);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);

    Serial.print("humV.val=");
    Serial.print(hum);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);

    Serial.print("tvocV.val=");
    Serial.print(tvoc);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
  }
}

void checkForIncomingData() {
  // Check if data is coming from the Nextion
  if (Serial.available() > 0) {
    receivedData = Serial.readString();
    delay(30);
    if (receivedData == "0") {
      r = 0;
    }
    if (receivedData == "1") {
      r = 1;
    }
    if (receivedData == "2") {
      r = 2;
    }
    if (receivedData == "3") {
      r = 3;
    }
    if (receivedData == "4") {
      r = 4;
    }
  }
  // if we have received any data, send data to the Nextion display to change
to page 1, or the waveform
  if (r == 0 || r == 1 || r == 2 || r == 3 || r == 4) {
    delay(200);
    dataTimer3 = millis();
    while (millis() - dataTimer3 <= 200) {
      Serial.print("pageSwitch.val="); // Activate page 1, or the waveform on
the Nextion display
      Serial.print(1);
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
    delay(100);
    getLast24Hours(); // get the last 24 hours and print them on as X-axis
values on the waveform
    getYAxisValues(); // get the Y-axis values according to the sensor, it's
range and it's max value. Print the Y-axis values as well as scale the Y-axis
of the wavefrom accordingly
    sendDataToWaveform(); // send the stored data of the last 24 hours to the
waveform
    r = 99; // reset the "r" to 99(arbitrary number, different than the ones
we assign when we receive data depending on which sensor we have pressed)
  }
}
void storeData() {
  // Storing current sensor values into arrays
  if ((minutes - previousMinutes) >= 15) {  // store the value each 15
minutes
    memmove(tempData, &tempData[1], sizeof(tempData)); // Slide data down one
position
    tempData[sizeof(tempData) - 1] = temp; // store newest value to last
position
    memmove(humData, &humData[1], sizeof(humData));
    humData[sizeof(humData) - 1] = hum;
    memmove(tvocData, &tvocData[1], sizeof(tvocData));
    // we use bytes for storing the data, as we said the Arduino Pro mini
doesn't have enough memory, so we must convert the values from 0 to 1000 to 0
to 255 which is one byte
    tvocData[sizeof(tvocData) - 1] = map(tvoc, 0, 1000, 0, 255);
    memmove(co2Data, &co2Data[1], sizeof(co2Data));
    co2Data[sizeof(co2Data) - 1] = map(CO2, 0, 3000, 0, 255);
    memmove(pm25Data, &pm25Data[1], sizeof(pm25Data));
    pm25Data[sizeof(pm25Data) - 1] = map(pm25, 0, 1000, 0, 255);
    memmove(o3Data, &o3Data[1], sizeof(o3Data));
    o3Data[sizeof(o3Data) - 1] = map(o3, 0, 1000, 0, 255);
    previousMinutes = minutes;

  }
  // So these if statemets check whether have passed 15 mins since the last
time we stored a value - you can change this to any minutes you want, but you
need to do that on both if statemets, for example "10" in the first if
statement, and "-50" in the second if statement
  else if ((minutes - previousMinutes) == -45) { // when minutes start from
0, next hour
    memmove(tempData, &tempData[1], sizeof(tempData)); // Slide data down one
position
```

```
    tempData[sizeof(tempData) - 1] = temp; // store newest value to last
position
    memmove(humData, &humData[1], sizeof(humData));
    humData[sizeof(humData) - 1] = hum;
    memmove(tvocData, &tvocData[1], sizeof(tvocData));
    tvocData[sizeof(tvocData) - 1] = map(tvoc, 0, 1000, 0, 255);
    memmove(co2Data, &co2Data[1], sizeof(co2Data));
    co2Data[sizeof(co2Data) - 1] = map(CO2, 0, 3000, 0, 255);
    memmove(pm25Data, &pm25Data[1], sizeof(pm25Data));
    pm25Data[sizeof(pm25Data) - 1] = map(pm25, 0, 1000, 0, 255);
    memmove(o3Data, &o3Data[1], sizeof(o3Data));
    o3Data[sizeof(o3Data) - 1] = map(o3, 0, 1000, 0, 255);
    previousMinutes = minutes;
  }
}

void getLast24Hours() {
  for (int i = 11; i >= 0; i--) {
    last24Hours[11] = hours; // get the current hour - according to this hour
get 12 more hours, each 2 hours. For example, current hour = 13, so 11, 9,
7...
    last24Hours[i - 1] = last24Hours[i] - 2;
    if (last24Hours[i - 1] < 0) {
      for (int k = -0; k > -11; k--) {
        if (last24Hours[i - 1] == k) {
          last24Hours[i - 1] = 24 + k;
        }
      }
    }
  }
  // send the hours values to the Nextion display
  for (int i = 0; i < 12; i++) {
    String last24 = ("n") + String(i) + String(".val=") +
String(last24Hours[i]); // e.g. for i=0 > "n0.val="
    Serial.print(last24);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
    delay(20);
  }
  // Another write just to make sure it sends all data
  for (int i = 0; i < 12; i++) {
    String last24 = ("n") + String(i) + String(".val=") +
String(last24Hours[i]); // e.g. for i=0 > "n0.val="
    Serial.print(last24);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
    delay(20);
  }
}
// With the following custom function we set the Y axis value for each sensor
individually, as each sensor has different maximum value for the Y axis
void getYAxisValues() {
  maxV = 0;
  // PM2.5 Y-axis values
  if (r == 0) {
    // Get the max sensor value from the last 24 hours
    for (int i = 0; i < sizeof(pm25Data); i++) {
      if (maxV < map(pm25Data[i], 0, 255, 0, 1000)) {
        maxV = map(pm25Data[i], 0, 255, 0, 1000);
      }
    }
    // Setting the Y-axis values and scaling the waveform
    if (maxV <= 100) {
      yAxisValues[0] = 25;
      yAxisValues[1] = 50;
      yAxisValues[2] = 75;
      yAxisValues[3] = 100;
      Serial.print("s0.dis="); // this command ".dis" is used for scalign the
Y-axis
      Serial.print(78 * 10);  // scale the waveform Y-axis - pm2.5 values are
from 0 to 1000 which are represented from 0 to 78% in the Y axis - 78%
because the wavefore is 200px which is 78% of 255 which is the default 100%
value of the waveform - 78*10 because we show values from 0 to 100, which are
10 times smaller
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
    // if the value is higher than 100, get its max value, and according to
it scale the y-axis - For example, if the max value is 235, set the max value
of the Y-axis to 300 - 235/100=2+1=3*100=300
    else if (maxV > 100) {
      int l = ((maxV / 100) + 1) * 100;  // get the hundreds value so we can
properly scale the Y axis of the waveform
      yAxisValues[0] = l / 4;
      yAxisValues[1] = l / 2;
      yAxisValues[2] = l * 3 / 4;
      yAxisValues[3] = l;
      float ll = 78.0 / (l / 1000.0); // scale value for the Y-axis in % - We
multiply by 78 instead of 100 because our waveform is 200px in height, which
is 78% of 255 (255 is max value the waveform can accept, 1 byte)
      Serial.print("s0.dis=");
      Serial.print(round(ll));
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
  }
  // TVOC Y-axis values
  if (r == 2) {
    // Get the max sensor value from the last 24 hours
    for (int i = 0; i < sizeof(tvocData); i++) {
      if (maxV < map(tvocData[i], 0, 255, 0, 1000)) {
        maxV = map(tvocData[i], 0, 255, 0, 1000);
      }
    }
    // Setting the Y-axis values and scaling the waveform
    if (maxV <= 100) {
      yAxisValues[0] = 25;
      yAxisValues[1] = 50;
      yAxisValues[2] = 75;
      yAxisValues[3] = 100;
      Serial.print("s0.dis=");
      Serial.print(78 * 10);
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
    else if (maxV > 100) {
      int l = ((maxV / 100) + 1) * 100;  // get the hundreds value so we can
properly scale the Y axis of the waveform
      yAxisValues[0] = l / 4;
      yAxisValues[1] = l / 2;
      yAxisValues[2] = l * 3 / 4;
      yAxisValues[3] = l;
      float ll = 78.0 / (l / 1000.0); // scale value for the Y-axis in % - We
multiply by 78 instead of 100 because our waveform is 200px in height, which
is 78% of 255 (255 is max value the waveform can accept, 1 byte)
      Serial.print("s0.dis=");
      Serial.print(round(ll));
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
  }
  // Ozone Y-axis values
  if (r == 3) {
    // Get the max sensor value from the last 24 hours
    for (int i = 0; i < sizeof(o3Data); i++) {
      if (maxV < map(o3Data[i], 0, 255, 0, 1000)) {
        maxV = map(o3Data[i], 0, 255, 0, 1000);
      }
    }
    // Setting the Y-axis values and scaling the waveform
    if (maxV <= 100) {
      yAxisValues[0] = 25;
      yAxisValues[1] = 50;
      yAxisValues[2] = 75;
      yAxisValues[3] = 100;
      Serial.print("s0.dis=");
      Serial.print(78 * 10);
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
    else if (maxV > 100) {
      int l = ((maxV / 100) + 1) * 100;  // get the hundreds value so we can
properly scale the Y axis of the waveform
      yAxisValues[0] = l / 4;
      yAxisValues[1] = l / 2;
      yAxisValues[2] = l * 3 / 4;
      yAxisValues[3] = l;
      float ll = 78.0 / (l / 1000.0); // scale value for the Y-axis in % - We
multiply by 78 instead of 100 because our waveform is 200px in height, which
is 78% of 255 (255 is max value the waveform can accept, 1 byte)
      Serial.print("s0.dis=");
      Serial.print(round(ll));
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
  }
  // CO2 Y-axis values are fixed from 0 to 3000 so we don't need to look for
the max value in the array
  if (r == 1) {
    // Get the max sensor value from the last 24 hours
    for (int i = 0; i < sizeof(co2Data); i++) {
      if (maxV < map(co2Data[i], 0, 255, 0, 3000)) {
        maxV = map(co2Data[i], 0, 255, 0, 3000);
      }
    }
    if (maxV <= 2000) {
      // Setting the Y-axis values and scaling the waveform
      yAxisValues[0] = 500;
      yAxisValues[1] = 1000;
      yAxisValues[2] = 1500;
      yAxisValues[3] = 2000;
      Serial.print("s0.dis=");
      Serial.print(117);  // scale the waveform from 0 - 3000 to 0 - 2000
range
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
    if (maxV > 2000) {
      // Setting the Y-axis values and scaling the waveform
      yAxisValues[0] = 750;
      yAxisValues[1] = 1500;
      yAxisValues[2] = 2250;
      yAxisValues[3] = 3000;
      Serial.print("s0.dis=");
      Serial.print(78);
      Serial.write(0xff);
      Serial.write(0xff);
      Serial.write(0xff);
    }
  }
  // Temperature and Humidity Y-axis values - fixed from 0 to 100
  if (r == 4) {
    // Setting the Y-axis values and scaling the waveform
    yAxisValues[0] = 25;
    yAxisValues[1] = 50;
    yAxisValues[2] = 75;
    yAxisValues[3] = 100;
    Serial.print("s0.dis=");
    Serial.print(200); // from 0 to 100 - 255/100 * 78 = ~200
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
  }
  delay(50);
  // Send the  Y-axis values to the Nextion display
  for (int i = 0; i < 4; i++) {
    String yValues = ("y") + String(i) + String(".val=") +
String(yAxisValues[i]); // e.g. for i=0 > "y0.val="
    Serial.print(yValues);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
    delay(10);
  }
}

void sendDataToWaveform() {
  int k = 0;
  while (k != 2) {
    String str = String("addt 1,0,") + String(288); // with this command we
tell the nextion display that we will send an array of data to the waveform
    Serial.print(str);
    delay(100);
    Serial.write(0xFF);
    Serial.write(0xFF);
    Serial.write(0xFF);
    delay(100);
    // Now depending on the selected sensor we want the values stored in the
arrays
    // PM2.5
    if (r == 0) {
      for (int t = 0; t < sizeof(pm25Data); t++) {
        int z = 0;
        while (z != 3) {
          Serial.write(pm25Data[t]);
          z++;
        }
      }
    }
    // CO2
    if (r == 1) {
      for (int t = 0; t < sizeof(co2Data); t++) {
        int z = 0;
        while (z != 3) {
          Serial.write(co2Data[t]);
          z++;
        }
      }
    }
    // TVOC
    if (r == 2) {
      for (int t = 0; t < sizeof(tvocData); t++) {
        int z = 0;
        while (z != 3) {
          Serial.write(tvocData[t]);
          z++;
        }
      }
    }
    // Ozone
    if (r == 3) {
      // Temperature values on channel 0
      for (int t = 0; t < sizeof(o3Data); t++) {
        int z = 0;
        while (z != 3) {
          Serial.write(o3Data[t]);
          z++;
        }
      }
    }
    // Temp and hum
    if (r == 4) {
      for (int t = 0; t < sizeof(humData); t++) {
        int z = 0;
        while (z != 3) {
          Serial.write(humData[t]);
          z++;
        }
      }
      delay(100);
      Serial.write(0xFF);
      Serial.write(0xFF);
      Serial.write(0xFF);
      delay(100);
      // Humidity values on channel 1
      String str = String("addt 1,1,") + String(288);
      Serial.print(str);
      delay(100);
      Serial.write(0xFF);
      Serial.write(0xFF);
      Serial.write(0xFF);
      delay(100);
      for (int t = 0; t < sizeof(tempData); t++) {
        int z = 0;
        while (z != 3) {
          Serial.write(tempData[t]);
          z++;
        }
      }
      delay(100);
      Serial.write(0xFF);
      Serial.write(0xFF);
      Serial.write(0xFF);
    }
    k++;
  }
}
```

Resistor

The resistor is a passive electrical component that creates resistance in the flow of electric current. In almost all electrical networks and electronic circuits they can be found. The resistance is measured in ohms (Ω). An ohm is the resistance that occurs when a current of one ampere (A) passes through a resistor with a one volt (V) drop across its terminals.

Resistors are used for many purposes. A few examples include limiting electric current, voltage division, heat generation, matching and loading circuits, gain control, and setting time constants. They are commercially available with resistance values over a range of more than nine orders of magnitude. They can be used as electric brakes to dissipate kinetic energy from trains, or be smaller than a square millimeter for electronics.

Resistors can be divided by functional type as well as resistance material. The following breakdown for the types can be made:

Fixed Resistors- the value is fixed &cannot be varied.

Variable Resistor-the value can be varied by an adjuster knob.

Resistor Color Code

An electronic color code is a code that is used to specify the ratings of certain electrical components, such as the resistance in Ohms of a resistor. Electronic color codes are also used to rate capacitors, inductors, diodes, and other electronic components, but are most typically used for resistors. Only resistors are addressed by this calculator.

How the color coding works:

The color coding for resistors is an international standard that is defined in IEC 60062. The resistor color code shown in the table below involves various colors that represent significant figures, multiplier, tolerance, reliability, and temperature coefficient. Which of these the color refers to is dependent on the position of the color band on the resistor. In a typical four-band resistor, there is a spacing between the third and the fourth band to indicate how the resistor should be read (from left to right, with the lone band after the spacing being the right-most band). In the explanation below, a four-band resistor (the one specifically shown below) will be used. Other possible resistor variations will be described after.

Resistor 4 band significant figure component

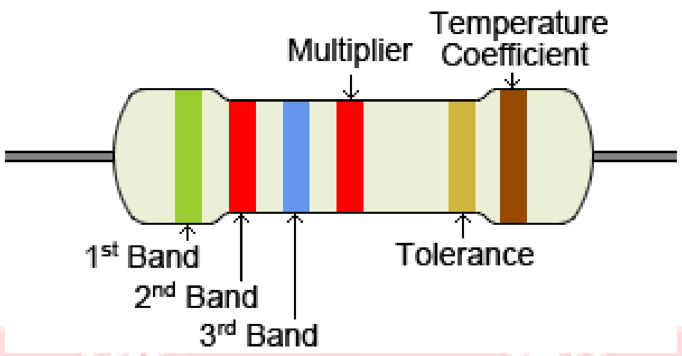
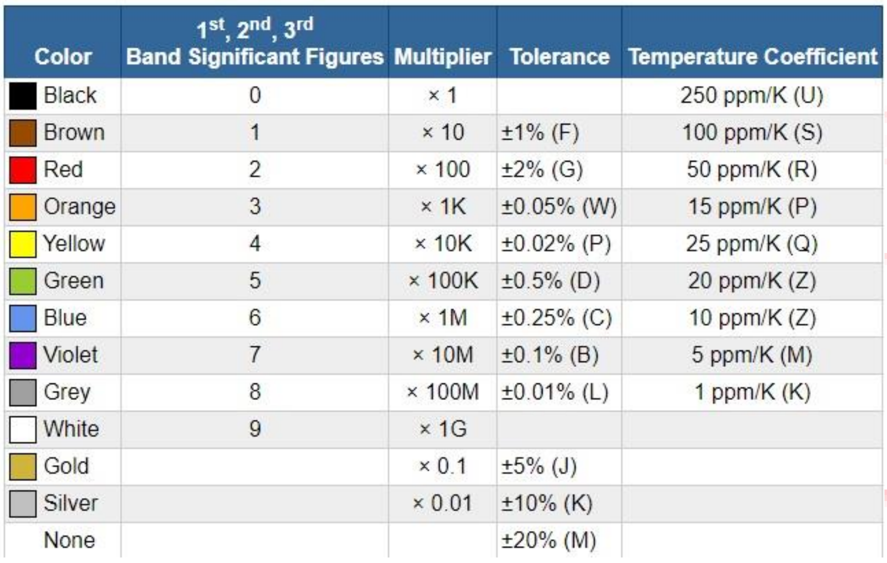


Figure 9 Resistor colour Band

On the most precise of resistors, a 6th band may be present. The first three bands would be the significant figure bands, the 4th the multiplier, the 5th the tolerance, and the 6th could be either reliability or temperature coefficient. There are also other possible variations, but these are some of the more common configurations.

Table 1 Colour Coding



Color	1st, 2nd, 3rd Band Significant Figures	Multiplier	Tolerance	Temperature Coefficient
⬛ Black	0	× 1		250 ppm/K (U)
🟫 Brown	1	× 10	±1% (F)	100 ppm/K (S)
🟥 Red	2	× 100	±2% (G)	50 ppm/K (R)
🟧 Orange	3	× 1K	±0.05% (W)	15 ppm/K (P)
🟨 Yellow	4	× 10K	±0.02% (P)	25 ppm/K (Q)
🟩 Green	5	× 100K	±0.5% (D)	20 ppm/K (Z)
🟦 Blue	6	× 1M	±0.25% (C)	10 ppm/K (Z)
🟪 Violet	7	× 10M	±0.1% (B)	5 ppm/K (M)
▨ Grey	8	× 100M	±0.01% (L)	1 ppm/K (K)
⬜ White	9	× 1G		
▨ Gold		× 0.1	±5% (J)	
▨ Silver		× 0.01	±10% (K)	
None			±20% (M)	

Transistor

Transistors are fundamental components in modern electronics, serving as the building blocks of digital circuits and amplifiers. They're semiconductor devices used to amplify or switch electronic signals and electrical power. Invented in the late 1940s, transistors revolutionized the field of electronics, replacing bulky and less efficient vacuum tubes.

Transistors come in various types, including bipolar junction transistors (BJTs) and field-effect transistors (FETs). BJTs have three regions of operation: the emitter, the base, and the collector. They can amplify current and are commonly used in analog circuits. FETs, on the other hand, control current with an electric field instead of through direct contact, making them ideal for digital circuits.

Transistors have enabled the miniaturization of electronic devices, leading to the development of smaller, faster, and more efficient computers, smartphones, and other electronic gadgets. They're essential components in integrated circuits (ICs), where millions of transistors can be packed onto a single chip, allowing for complex functionality in a tiny space.

Figure 10 Transistor

Capacitor

A capacitor is an electrical component that stores and releases electrical energy. It consists of two conductive plates separated by an insulating material called a dielectric. When a voltage is applied across the plates, electric charge accumulates on them, creating an electric field between them. This process stores energy in the capacitor.

Capacitors come in various types and sizes, with different capacitance values and voltage ratings to suit different applications. They're used in circuits for a variety of purposes:

- Energy Storage- Capacitors store electrical energy and release it when needed. They can provide a quick burst of energy in devices like camera flashes or in power supply circuits to smooth out voltage fluctuations.
- Filtering- Capacitors are used in conjunction with resistors and inductors to filter out unwanted signals or noise from electrical circuits. They're often found in power supply circuits to reduce ripple voltage.
- Timing- Capacitors are used in timing circuits to control the frequency of oscillations. They determine the time constant in circuits like RC (resistor-capacitor) oscillators.
- Coupling and Decoupling- Capacitors are used to couple or decouple different parts of a circuit. They allow AC signals to pass while blocking DC signals, or they can bypass AC signals to ground, reducing noise in the circuit.

Capacitors come in various types, each designed for specific applications and with unique characteristics. Here are some common types of capacitors:

- Ceramic Capacitors- These capacitors use a ceramic material as the dielectric. They are widely used due to their small size, low cost, and stability over a wide range of temperatures and frequencies.
- Electrolytic Capacitors- These capacitors use an electrolyte (usually a liquid or gel) as the dielectric. They offer high capacitance values but are polarized, meaning they must be connected in a specific orientation in a circuit.
- Film Capacitors- These capacitors use a thin plastic film as the dielectric. They offer good stability, high insulation resistance, and low dielectric losses.

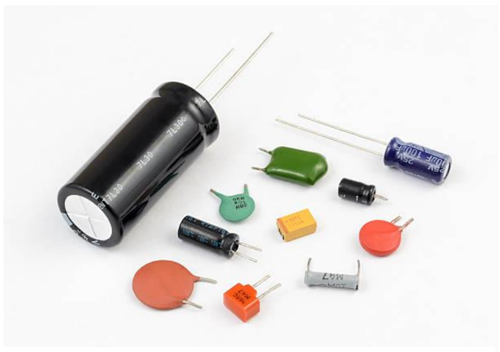


Figure 11 Capacitor

PCB Board

A PCB (Printed Circuit Board) is a foundational component in modern electronics, serving as a platform for assembling electronic circuits. Typically made of fiberglass or epoxy laminate, it features a thin layer of copper foil etched into pathways called traces. These traces connect electronic components such as resistors, capacitors, and integrated circuits, forming a functional circuit. PCBs offer several advantages, including compactness, reliability, and ease of mass production. They come in various types, such as single-sided, double-sided, and multi-layered, to accommodate different circuit complexities. PCBs play a crucial role in virtually all electronic devices, from smartphones and computers to household appliances and industrial machinery.

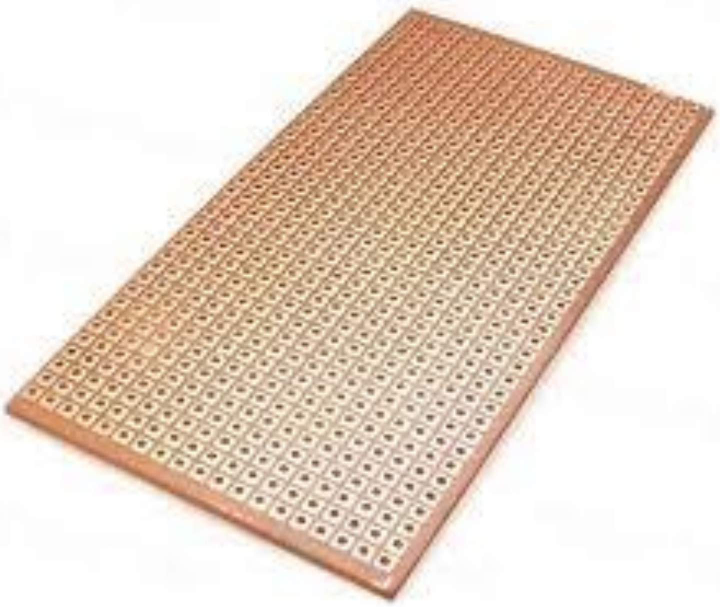


Figure 12 Dot PCB Board

PMS7003 Sensor (Air Quality Sensor)

The PMS7003 is a type of particulate matter (PM) sensor manufactured by Plantower. It's designed to detect and measure airborne particulate matter with a diameter of 2.5 micrometers or less (PM2.5) and particulate matter with a diameter of 10 micrometers or less (PM10). These particles can come from various sources such as smoke, dust, pollen, and vehicle emissions and can have detrimental effects on human health when inhaled.

The PMS7003 sensor utilizes laser scattering technology to measure particle concentrations in the air. It consists of a laser diode, a photoelectric sensor, and a microprocessor to calculate particle concentrations based on the scattering of light by the particles.

These sensors are commonly used in air quality monitoring systems, indoor air purifiers, environmental monitoring devices, projects aimed at measuring and analyzing air pollution levels. They provide real-time data on particulate matter levels, allowing for better understanding and management of air quality in indoor and outdoor environments.

Figure 13 PMS7003 Sensor

MH-Z19 Sensor (CO_2 Sensor)

The MH-Z19 is a compact infrared carbon dioxide (CO2) sensor module. Manufactured by Winsen, it's designed for measuring CO2 concentrations in the atmosphere. This sensor uses non-dispersive infrared (NDIR) technology to accurately detect CO2 levels.

NDIR sensors work by measuring the absorption of infrared light by CO2 molecules. The MH-Z19 sensor contains an infrared light source, a gas chamber, and a detector. When infrared light passes through the gas chamber containing the air sample, the CO2 molecules absorb some of the infrared light at specific wavelengths. The amount of light absorbed is proportional to the concentration of CO2 in the air. By measuring the intensity of the light after it passes through the gas chamber, the sensor can determine the CO2 concentration.

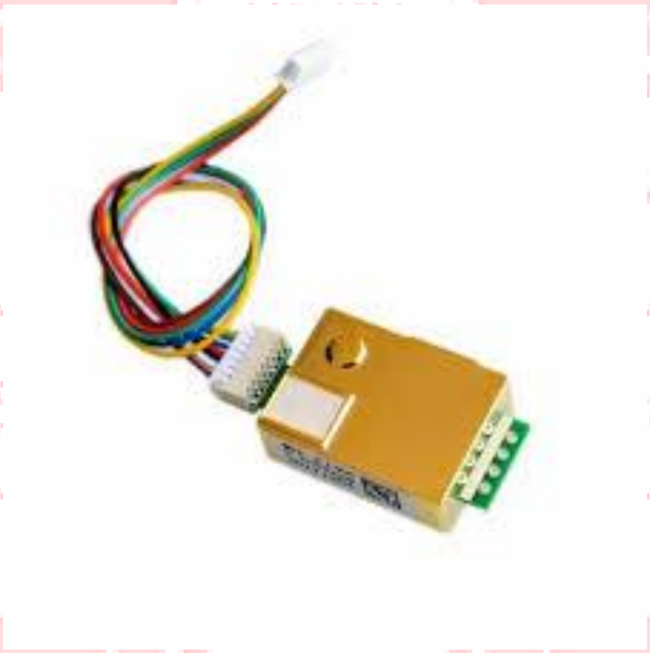


Figure 14 MH-Z19

MQ-131 Sensor (Ozone Sensor)

The MQ131 is a gas sensor module designed to detect ozone (O_3) in the air. Manufactured by Winsen, it's part of the MQ series of gas sensors widely used in various environmental monitoring applications.

The MQ131 sensor operates on the principle of chemiresistive technology. It contains a sensing element made of a tin dioxide (SnO_2) semiconductor material that changes its resistance when exposed to ozone gas. The resistance change is proportional to the concentration of ozone in the air.

The sensor module includes a built-in heater to control the operating temperature of the sensing element. This heating element is necessary for the sensor to function properly and ensure accurate measurements.

The MQ131 sensor outputs an analog voltage signal that varies based on the ozone concentration detected. By measuring this voltage output, typically through an analog-to-digital converter (ADC) connected to a microcontroller or data acquisition system, one can determine the ozone concentration in the surrounding environment.

Applications of the MQ131 sensor include ozone monitoring in indoor and outdoor air quality monitoring systems, ozone generators, environmental monitoring devices, and industrial safety equipment where ozone detection is critical.

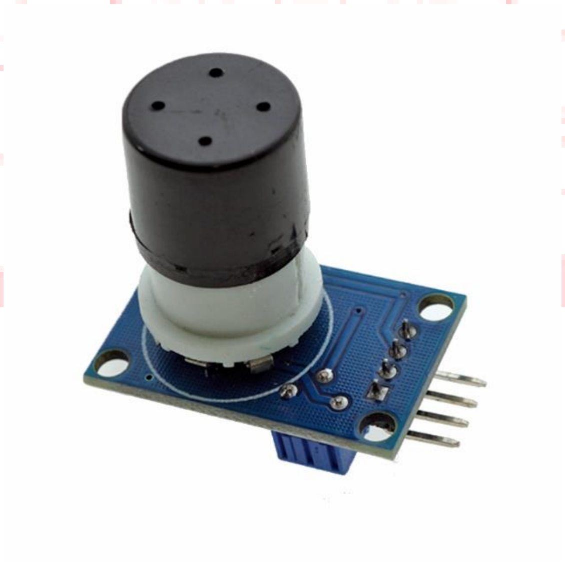


Figure 15 MQ-131 SENSOR

MP503 (VOC Sensor)

The MP503 is a gas sensor module primarily designed to detect the presence of formaldehyde (HCHO) in the air. Formaldehyde is a colorless, strong-smelling chemical often used in building materials and household products, and exposure to high levels can cause health issues.

The MP503 sensor operates based on the principle of semiconductor gas sensing. It contains a sensitive layer composed of tin dioxide (SnO_2) nanoparticles, which undergo changes in conductivity when exposed to formaldehyde gas. The resistance change in the sensor's sensitive layer is proportional to the concentration of formaldehyde in the air.

Figure 16 MP503

DHT 22 (Digital Humidity &Temperature Sensor)

The DHT22 is a digital temperature and humidity sensor module that provides accurate readings of temperature and relative humidity. It's part of the DHT series of sensors produced by Aosong Electronics.

The DHT22 sensor consists of a capacitive humidity sensing element and a thermistor for temperature measurement, integrated with signal processing circuitry. It communicates with external devices, such as microcontrollers, through a single-wire digital interface, making it easy to interface with various electronic platforms.

Key features of the DHT22 sensor include:

- High Accuracy- The DHT22 sensor offers high accuracy in both temperature and humidity measurements, typically with ±0.5°C accuracy for temperature and ±2-5% accuracy for relative humidity.
- Wide Operating Range- It can operate within a wide temperature range, typically from -40°C to 80°C, making it suitable for various environmental conditions.

Applications of the DHT22 sensor include weather stations, environmental monitoring systems, HVAC (Heating, Ventilation, and Air Conditioning) systems, agricultural monitoring, and home automation projects where accurate temperature and humidity measurements are required.

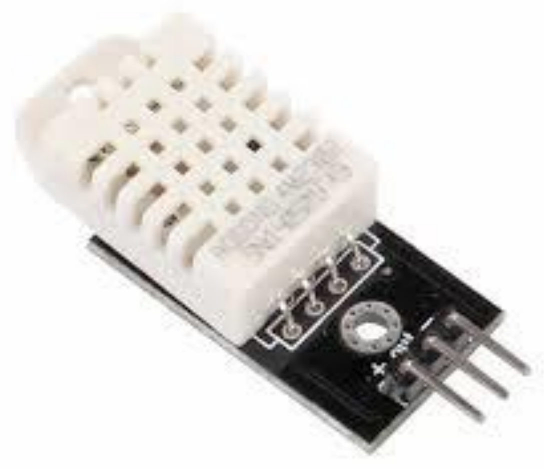


Figure 17 DHT22

DS3231 (Real Time Clock)

The DS3231 is a highly accurate real-time clock (RTC) module manufactured by Maxim Integrated. It provides precise timekeeping functions for electronic devices and systems.

Key features of the DS3231 RTC module include:

- High Accuracy-The DS3231 offers excellent timekeeping accuracy, typically within a few seconds per month, even in extreme temperature conditions.
- Integrated Temperature Compensation- It includes an integrated temperature-compensated crystal oscillator (TCXO) that maintains accurate timekeeping over a wide temperature range without the need for external calibration.
- Battery Backup-The DS3231 has a built-in backup battery input that allows it to maintain timekeeping functions during power outages or when the main power source is disconnected.

The DS3231 RTC module is commonly used in a wide range of applications, including:

- Clocks and watches

- Data loggers and recorders

- Temperature monitoring systems

- Industrial automation

- Embedded systems and IoT (Internet of Things) devices

- Consumer electronics

- Automotive applications

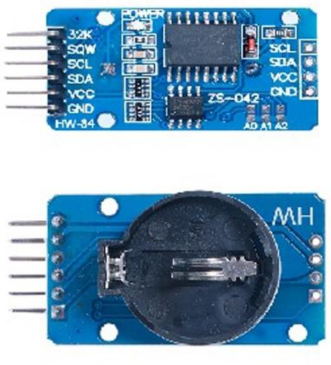


Figure 18 DS3231

SPI DISPLAY

An SPI touch display is a type of touchscreen display that utilizes the Serial Peripheral Interface (SPI) communication protocol for data transfer between the microcontroller or host device and the display controller.

It works:

- Display Panel- The display panel includes the screen where graphics and text are displayed, and it's often a TFT (Thin-Film Transistor) LCD (Liquid Crystal Display) or OLED (Organic Light Emitting Diode) panel.
- Touch Sensor- The touch sensor layer, usually capacitive or resistive, allows users to interact with the display by touching the screen.

- Controller IC- The display controller IC manages the display and touch functionalities. It communicates with the microcontroller or host device using the SPI protocol.
- SPI Interface- SPI is a synchronous serial communication protocol that allows devices to exchange data over short distances. In an SPI touch display, the microcontroller or host device sends commands and data to the display controller via SPI, and the controller responds accordingly. This enables the microcontroller to control what is displayed on the screen and to receive touch input from users.
- Driver Software- The microcontroller or host device typically runs driver software to interface with the display controller and handle display updates, touch events, and other interactions.

SPI touch displays are commonly used in various applications, including:

- Portable electronic devices like smartphones and tablets

- Industrial control panels

- Medical devices

- Consumer electronics

- Automotive infotainment systems

- Home automation interfaces

Figure 19 SPI Display

Chapter-6

Chapter-6

Biblography

6.1 References

Kim Oanh N.T. et al.'s (2018) "A Review of Air Pollution Monitoring Techniques" offers a thorough analysis of a variety of monitoring approaches, including both established practices and recently developed technologies.
Mukhtar A. et al.'s "Air Pollution Monitoring and Control: Recent Advances and Future Challenges" (2020) explores the latest developments in air pollution monitoring and control technology while outlining obstacles and potential paths forward.

The use of satellite data for air quality monitoring is the subject of "Satellite-Based Air Quality Monitoring: From Source to Impact" by Bechle M.J. et al. (2015), which offers insights into global-scale monitoring initiatives.
Research on satellite-based monitoring methods for assessing air quality is frequently published in the journal "Remote Sensing of Environment".

The use of inexpensive sensors for air quality monitoring is covered in "Low-Cost Sensors for the Measurement of Atmospheric Composition: Overview of Topic and Future Applications" by Lewis A.C. et al. (2016), along with their possible uses and drawbacks.
The creation and implementation of inexpensive air quality monitoring stations for urban settings are covered in the 2015 paper "Development of Low-Cost Portable Air Quality Monitoring Stations for Assessing Urban Air Quality" by Kumar P. et al.

According to Yigitcanlar T. et al. (2018), "Smart Cities and Smart Sensors: Assessing the Data Privacy and Security Challenges" delves into the integration of smart sensors for air quality monitoring into urban contexts, emphasizing privacy and security problems.
An overview of emerging technologies for air quality monitoring in the context of smart cities can be found in "Air Quality Monitoring in Smart Cities: A Review of Emerging Technologies and Deployment Challenges" by Sharma A. et al. (2019).

Srivastava A. et al.'s "Data-Driven Techniques for Monitoring and Predicting Air Quality" (2019) explores data-driven methods for air quality monitoring and prediction, such as artificial intelligence and machine learning.
Recent developments in air quality forecasting models and methodologies are reviewed in "Air Quality Forecasting: Recent Advances, Challenges, and Opportunities" by Zhang Y. et al. (2019).

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

INSTITUTE OF TECHNOLOGY AND MANAGEMENT, GWALIOR (M.P.)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

CERTIFICATE OF APPROVAL

The forgoing project entitled “Air Pollution Monitoring Device” is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a pre-requisite to the degree for which it is submitted. It is understood that by this approval, the undersigned do not necessarily endorse any conclusion or opinion therein, but approve the project for the purpose for which it was submitted.

EXAMINER **HOD**

DIRECTOR