



DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to Visvesvaraya Technological University (VTU), Belagavi,
Approved by AICTE and UGC, Accredited by NAAC with 'A' grade & ISO 9001 – 2015 Certified Institution)
Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560 111, India



DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

(Accredited by NBA Tier 1: 2022-2025)

Project Report on

INTEGRATED BUS TRACKING AND SAFETY MANAGEMENT SYSTEM

Submitted in partial fulfillment for the award of the degree of

Bachelor of Engineering in Electronics & Communication Engineering

Submitted by

RAGHAVENDRA YARAGATTI	1DS21EC159
VAISHNAVI	1DS21EC184
SHRINIVAS V A	1DS21EC196
SHUBHAVANI P DOLLI	1DS21EC202

Under the Guidance of

Dr. H V Manjunath

Professor

Department of Electronics and Communication Engineering

DSCE, Bengaluru

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANASANGAMA, BELAGAVI-590018, KARNATAKA, INDIA 2024

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to Visvesvaraya Technological University (VTU), Belagavi,
Approved by AICTE and UGC, Accredited by NAAC with 'A' grade & ISO 9001 – 2015 Certified Institution)
Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560 111, India

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

(Accredited by NBA Tier 1: 2022-2025)



CERTIFICATE

Certified that the project report entitled “**Integrated bus tracking and Safety Management System**” carried out by **Raghavendra Yaragatti (1DS21EC159), VAISHNAVI(1DS21EC184), SHRINIVAS V A(1DS21EC196), SHUBHAVANI P DOLLI(1DS21EC202)**

are bonafide students of DAYANANDA SAGAR COLLEGE OF ENGINEERING, an autonomous institution affiliated to VTU, Belagavi in partial fulfillment for the award of Degree of Bachelor of Electronics & Communication Engineering during the year 2024-2025. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements with respect to the work prescribed for the said Degree.

Signature of the guide

Dr. H V Manjunath
Professor
Dept. of ECE, DSCE
Bengaluru

Signature of the HOD

Dr. Shobha. K. R
Professor & Head Dept.
of ECE, DSCE, Bengaluru

Signature of the Principal

Dr. B G Prasad
Principal DSCE, Bengaluru

Name of the Examiners

Signature with date

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to Visvesvaraya Technological University (VTU), Belagavi,
Approved by AICTE and UGC, Accredited by NAAC with 'A' grade & ISO 9001 – 2015 Certified Institution)
Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560 111, India

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

(Accredited by NBA Tier 1: 2022-2025)



DECLARATION

We Raghavendra Yaragatti (1DS21EC159), VAISHNAVI(1DS21EC184), Shrinivas V A (1DS21EC196) and Shubhavani P Dolli (1DS21EC202), respectively, hereby declare that the project work entitled “**Integrated Bus tracking and safety management system**” has been independently done by us under the guidance of ‘**Dr. H V Manjunath**’, Professor, ECE department and submitted in partial fulfillment of the requirement for the award of the degree of **Bachelor of Electronics & Communication Engineering** at **Dayananda Sagar College of Engineering**, an autonomous institution affiliated to VTU, Belagavi during the academic year 2024-2025.

We further declare that we have not submitted this report either in part or in full to any other university for the award of any degree.

RAGHAVENDRA YARAGATTI	1DS21EC159
VAISHNAVI	1DS21EC184
SHRINIVAS V A	1DS21EC196
SHUBHAVANI P DOLLI	1DS21EC202

PLACE: BENGALURU

DATE: 17/11/24

ACKNOWLEDGEMENT

The satisfaction and euphoria accompanying the successful completion of any task would be incomplete without the mention of people who made it possible and under constant guidance and encouragement the task was completed. We sincerely thank the **Management of Dayananda Sagar College of Engineering, Bengaluru.**

We express our sincere regards and thanks to **Dr. B G Prasad, Principal, Dayananda Sagar College of Engineering, Bengaluru.** His constant encouragement guidance and valuable support have been an immense help in realizing this project.

We express our sincere regards and thanks to **Dr. Shobha.K.R, Professor & Head, Department of Electronics & Communication Engineering, Dayananda Sagar College of Engineering, Bengaluru.** Her incessant encouragement guidance and valuable technical support have been an immense help in realizing this project. Her guidance gave us the environment to enhance our knowledge, and skills and to reach the pinnacle with sheer determination, dedication, and hard work.

We would like to express profound gratitude to our Project guide **Dr. H V Manjunath, Professor, Department of Electronics & Communication Engineering, Dayananda Sagar College of Engineering, Bengaluru** who has encouraged us throughout the project. His moral support enabled us to complete our project work successfully.

We express our sincere thanks to Project Coordinators **Dr.S.Thenmozhi, Dr.Suma M.R, Dr.Manasa, Dr.VinayN.A, Mrs.SrividhyaL, and Mrs.Bindhu H.M,** of the **Department of Electronics and Communication Engineering** for their continues support and guidance. We thank all teaching and non-teaching staff of the Department of Electronics and Communication Engineering for their kind and constant support throughout the academic Journey.

RAGHAVENDRA YARGATTI	1DS21EC159
VAISHNAVI	1DS21EC184
SHRINIVAS V A	1DS21EC196
SHUBHAVANI P DOLLI	1DS21EC202

ABSTRACT

This project presents the design and implementation of a comprehensive Bus Tracking and Monitoring System utilizing an Arduino microcontroller. The system is equipped with multiple sensors to ensure the safety and efficiency of bus operations. Emission levels are monitored to ensure compliance with environmental standards, while engine temperature is measured to prevent overheating.

Vehicle stability is detected to alert users of potential hazards, and tire pressure is monitored to prevent accidents caused by underinflation. Real-time location tracking is enabled for effective route management. The system also detects accidents or significant impacts, triggering immediate alerts. All data is transmitted to the Blynk IoT platform using a Wi-Fi module, where it is visualized in real-time. In case of an accident, the system sends instant alerts to designated users, enhancing response times and ensuring safety. This system provides a robust solution for fleet management, enabling continuous monitoring of critical parameters and improving overall bus operation safety and efficiency.

The Bus Tracking and Monitoring System is a solution designed to enhance the efficiency, safety, and convenience of public transportation systems by enabling real-time tracking and monitoring of buses. This system leverages GPS technology, cloud computing, and mobile applications to provide accurate, real-time location updates for buses. With a user-friendly interface, the system allows passengers to track buses on their route, view estimated arrival times, and receive notifications of delays or route changes. Additionally, the monitoring feature aids transportation authorities in overseeing bus operations, optimizing routes, ensuring adherence to schedules, and improving response times for unexpected events. By reducing waiting times and enhancing overall service reliability, this system aims to improve passenger satisfaction and streamline bus fleet management, leading to more sustainable and efficient urban transportation.

ABSTRACT	iii
ACKNOWLEDGMENT	iv
LIST OF TABLES	vii
LIST OF FIGURES	viii
LIST OF ABBREVIATIONS AND SYMBOLS	ix

1. INTRODUCTION	1
1.1 Overview	
1.2 Problem Statement	
1.3 Objectives	
1.4 Motivation	
2. LITERATURE SURVEY	22
3. PROBLEM ANALYSIS & DESIGN.....	40
3.1 Block diagram and working principle of proposed system	
3.2 Circuit diagram of Proposed System	
3.3 Hardware Design	
3.4 Software Design	
4. IMPLEMENTATION	56
4.1 Overview of System Implementation	
4.2 Algorithm or Flow-Chart/Data Flow Diagram	
4.3 Hardware Implementation	
4.4 Software Implementation	
5. RESULTS.....	72
6. APPLICATION, ADVANTAGES& LIMITATION	75
6.1 Application	
6.2 Advantages	
6.3 Limitation	
7. CONCLUSION AND FUTURE SCOPE.....	81
7.1 Conclusion	

7.2 Future Scope

REFERENCES

PUBLICATION DETAILS

PLAGIARISM REPORT

LIST OF TABLES

Fig. No.	Fig. Caption	Page No.
1	Arduino Specifications	15
2	NEO-6MV2 GPS Module Pin Configuration	20
3	ADXL345 Module Pin Configuration	24
4	Specifications of MQ-7 Gas Sensor	29
5	Specifications of Hx710B Air Pressure Sensor	31

LIST OF FIGURES

Fig. No.	Fig. Caption	Page No.
1	Block Diagram of the proposed system	13
2	Circuit Diagram of Proposed System	14
3	Arduino Uno	15
4	ESP8266 Wi-Fi Module	17
5	GPS Module	19
6	Accelerometer	22
7	Vibration Sensor	24
8	16 X 2 LCD Display	26
9	MQ-7 Gas Sensor	28
10	Hx710B Air Pressure Sensor	30
11	Software Installation of Arduino IDE	34
12	Setup of Arduino IDE	39
13	Installation Process of Blynk IOT Application	49
14	Flow Chart of Proposed Methodology	61
15	Hardware Implementation of Proposed Methodology	62
16	Results displayed in LCD, Blynk-IOT web and E-mail	72

LIST OF ABBREVIATIONS

Sl. No.	Abbreviation	Full Form
1.	IoT	Internet of Things
2.	LCD	Liquid Crystal Display
3.	Wi-Fi	Wireless Fidelity
4.	Node-MCU	Microcontroller Unit
5.	PWM	Pulse Width Modulation
6.	USB	Universal Serial Bus
7.	ICSP	In-Circuit Serial Programming
8.	ESP8266	A Wi-Fi microchip with full TCP/IP stack and MCU capability
9.	LED	Light Emitting Diode
10.	PCB	Printed Circuit Board
11.	COM	Common (Pole) on the relay1
12.	IDE	Integrated Development Environment
13.	GUI	Graphical User Interface
14.	API	Application Programming Interface
15.	ANSI	American National Standards Institute
16.	AVR	Alf and Vegard's RISC processor

1. INTRODUCTION

The transportation industry has witnessed a paradigm shift with the advent of the Internet of Things (IoT), which has enabled real-time monitoring and management of vehicles. Urbanization has significantly increased the demand for reliable, efficient, and safe public transportation systems. With the rise in the number of vehicles on the road, fleet management and passenger safety have become critical challenges for transportation authorities. In this context, a Bus Tracking and Monitoring System emerges as a comprehensive solution to address these challenges. This project leverages IoT and embedded systems to develop an intelligent Bus Tracking and Monitoring System that integrates various sensors for safety, efficiency, and environmental compliance. By continuously monitoring critical parameters such as emissions, engine temperature, tire pressure, and vehicle stability, the system ensures proactive management of bus fleets. Additionally, real-time tracking using GPS enables effective route management and enhances operational transparency.

The Bus Tracking and Monitoring System is a cutting-edge IoT-enabled platform that combines technology and innovation to revolutionize public transportation management. Built around an Arduino microcontroller, the system integrates a suite of sensors to monitor crucial parameters such as emissions, engine temperature, tire pressure, vehicle stability, and real-time location. This advanced monitoring ensures operational efficiency and safety by enabling early detection of anomalies like overheating, underinflated tires, or hazardous driving conditions. Moreover, the system is equipped with accident detection capabilities, providing instant alerts through the Blynk IoT platform to facilitate rapid response in emergencies. By offering seamless data visualization, fleet managers can oversee operations in real time, optimize route planning, and implement preventive maintenance strategies. The integration of IoT technology with transportation systems underscores a transformative approach to creating smarter, safer, and greener public transit solutions.

This Bus Tracking and Monitoring System also emphasizes sustainability and technological adaptability for future urban needs. Its real-time emission monitoring capability ensures compliance with strict environmental regulations, contributing to cleaner air and healthier urban environments. By streamlining fleet operations and reducing vehicle downtime through proactive maintenance, it helps minimize operational costs while maximizing efficiency.

1.1 Overview

The Bus Tracking and Monitoring System is an innovative IoT-based solution aimed at enhancing the safety, efficiency, and reliability of public transportation systems. As urban areas expand and the demand for sustainable and safe transportation grows, the integration of advanced technology in fleet management has become imperative. This system addresses critical challenges in modern public transportation, including real-time monitoring, safety assurance, environmental compliance, and effective fleet management. At the core of the system is an Arduino microcontroller, which acts as a central hub for integrating various sensors and communication modules. These sensors continuously monitor essential parameters such as engine temperature, tire pressure, vehicle stability, and emission levels, ensuring that buses operate efficiently and safely. The system also incorporates GPS tracking to provide accurate real-time location data, facilitating effective route management and operational oversight. In case of accidents or significant impacts, vibration sensors trigger immediate alerts, enabling rapid emergency response and minimizing potential harm to passengers.

One of the standout features of the system is its seamless integration with the Blynk IoT platform. This cloud-based platform enables real-time data visualization and remote monitoring through a user-friendly interface accessible via mobile devices or computers. Fleet managers can use this interface to monitor the operational health of buses, receive alerts for anomalies, and make informed decisions to optimize fleet performance. The Blynk platform also supports notifications for critical events, such as high emissions, engine overheating, or stability issues, ensuring that operators can take timely corrective actions.

The system's emphasis on environmental compliance is a key aspect of its design. By continuously monitoring emissions, it ensures that vehicles meet environmental standards, helping cities achieve their air quality goals. This functionality is particularly significant in urban areas with stringent regulations on vehicular emissions. The system's ability to detect and prevent emission-related issues proactively not only reduces environmental impact but also supports the transition toward greener public transportation. Another critical component of the system is its focus on safety. The integration of stability detection sensors ensures that hazardous driving conditions, such as sharp turns or uneven roads, are identified and addressed promptly. Tire pressure monitoring enhances both safety and fuel efficiency by

alerting drivers to deviations from optimal pressure levels. The vibration sensor provides an additional layer of security by detecting collisions or impacts, allowing for swift responses in emergencies.

The hardware configuration of the system, including the Arduino Uno microcontroller, ESP8266 Wi-Fi module, GPS module, and various sensors, is designed to ensure robustness, accuracy, and reliability. The Arduino Uno serves as a powerful and cost-effective platform for sensor integration and data processing, while the ESP8266 module enables secure and efficient data transmission to the cloud. This modular and scalable architecture allows for future upgrades, making the system adaptable to evolving technological and regulatory requirements.

In addition to its technical capabilities, the system offers significant operational benefits. Real-time monitoring and data visualization reduce the risk of vehicle breakdowns by enabling proactive maintenance. By identifying potential issues such as engine overheating or underinflated tires before they escalate, the system minimizes downtime and repair costs, ensuring that buses remain in optimal condition for service. The real-time GPS tracking functionality also enhances route management, enabling better scheduling and reducing delays, thereby improving passenger satisfaction.

The Bus Tracking and Monitoring System is a transformative solution for modernizing public transportation. Its integration of IoT technology addresses critical challenges in fleet management, safety, and environmental sustainability, making it a valuable tool for urban transit authorities and private operators alike. By providing a comprehensive and user-friendly platform for monitoring and managing bus operations, the system represents a significant step toward creating smarter, safer, and more efficient public transportation networks

1.2 Problem Statement

The rapid growth of urban areas has led to an increase in the demand for public transportation, particularly buses, as they provide an affordable and efficient mode of commuting. However, with this growth comes significant challenges in managing bus fleets effectively. The complexity of modern urban transportation systems requires enhanced tools and technologies to ensure safety, efficiency, and environmental compliance. Unfortunately, traditional systems fall short in providing the real-time, integrated monitoring necessary to address these challenges comprehensively.

Current fleet management systems often rely on fragmented solutions that monitor specific parameters such as GPS location or emission levels but fail to integrate critical operational data into a single platform. This lack of integration leads to inefficiencies in data collection and analysis, making it difficult for fleet managers to obtain a holistic view of vehicle performance. The inability to simultaneously monitor vital aspects such as emissions, engine temperature, vehicle stability, tire pressure, and accident detection results in delayed responses to potential issues. This not only compromises the safety of passengers and drivers but also impacts the operational efficiency and reliability of the entire fleet.

One of the most pressing issues is the inability to monitor vehicle emissions in real-time. Urban areas often face strict environmental regulations to combat air pollution, and non-compliance can lead to hefty fines and penalties. Traditional systems are not equipped to provide continuous monitoring of emissions, making it difficult for operators to ensure that vehicles remain within permissible limits. As a result, unchecked emissions contribute significantly to urban air pollution, undermining efforts to create cleaner and more sustainable cities. Another critical challenge is the lack of proactive safety measures in existing systems. Parameters such as engine temperature, vehicle stability, and tire pressure play a crucial role in ensuring safe operations, but they are often overlooked in conventional fleet management. For example, an overheating engine may lead to costly mechanical failures, while unstable movements or underinflated tires can result in accidents, endangering the lives of passengers and drivers.

Accident detection and emergency response are also areas where traditional systems fall short. Most existing setups do not have mechanisms to detect collisions or significant impacts in real-time. This leads to delays in notifying emergency services and responding to accidents, exacerbating the damage and potentially increasing casualties. The lack of an automated alert system for such scenarios further

hampers the ability of fleet managers to address emergencies promptly. In addition to safety and environmental concerns, operational inefficiencies plague traditional systems. Without integrated solutions, fleet managers struggle to optimize routes, schedule maintenance, and manage resources effectively. This often results in higher fuel consumption, increased downtime due to unexpected breakdowns, and reduced overall fleet reliability. Passengers experience delays and inconsistencies in service, eroding trust in public transportation as a reliable mode of travel.

To address the multifaceted challenges, there is an urgent need for a unified, IoT-based bus tracking and monitoring system. Such a system would integrate real-time monitoring of critical parameters, including emissions, engine temperature, vehicle stability, tire pressure, and accident detection, into a single platform. By leveraging IoT technology, this system would provide fleet managers with instant alerts and actionable insights, enabling them to respond promptly to potential issues and maintain optimal vehicle performance.

Real-time data visualization and centralized monitoring would allow fleet operators to track the health and performance of each vehicle in their fleet comprehensively. Automated alerts for anomalies such as high emissions, engine overheating, or unstable movements would ensure timely interventions, reducing the risk of accidents and breakdowns. Moreover, continuous emission monitoring would help operators comply with environmental regulations, contributing to cleaner urban environments. The implementation of such a system would not only enhance safety and efficiency but also reduce operational costs by enabling preventive maintenance and optimizing resource allocation. Passengers would benefit from improved service reliability and safety, restoring trust in public transportation as a dependable mode of travel. Additionally, the integration of IoT technology would position transportation systems as a key component of smart city initiatives, paving the way for more sustainable and connected urban mobility solutions.

1.3 Objectives

1. Real-Time Emission Monitoring

The growing need for environmental sustainability in urban areas requires transportation systems to adhere to stringent emission standards. Real-time emission monitoring ensures that buses operate within permissible pollutant levels, reducing their contribution to air pollution. By integrating gas sensors into the monitoring system, this objective is achieved by continuously measuring pollutant concentrations such as carbon monoxide (CO) and nitrogen oxides (NO_x) emitted by the bus. Operators can use this data to identify and address any irregularities promptly, minimizing environmental impact and avoiding regulatory penalties. This capability supports cities' goals of cleaner air and aligns with global initiatives to combat climate change.

2. Engine Temperature Control

The engine is the heart of any vehicle, and maintaining its temperature within optimal limits is crucial for ensuring reliability and longevity. Real-time monitoring of engine temperature helps detect potential overheating early, preventing severe mechanical failures that could result in costly repairs and service interruptions. Temperature sensors integrated with the system continuously provide updates on engine conditions. If the temperature exceeds safe thresholds, alerts are triggered to prompt immediate corrective action, such as reducing load or performing necessary maintenance. This proactive approach minimizes breakdowns and extends the operational lifespan of the fleet.

3. Vehicle Stability Detection

Ensuring the stability of a bus during operation is critical to passenger safety and comfort. Factors such as sharp turns, uneven roads, or sudden stops can compromise vehicle stability and lead to accidents. By employing accelerometers and gyroscopes, the system detects unusual tilts or abrupt movements that may indicate instability. Real-time alerts notify drivers and fleet managers of hazardous conditions, allowing them to implement corrective measures promptly. This feature not only prevents accidents but also enhances the overall travel experience for passengers, fostering trust in the safety of public transportation.

4. Tire Pressure Monitoring

Tire pressure significantly impacts vehicle safety, fuel efficiency, and operational costs. Underinflated tires increase rolling resistance, leading to higher fuel consumption and premature tire wear. Moreover, low tire pressure raises the risk of blowouts, potentially causing accidents. The system integrates tire pressure sensors to monitor the air pressure in real time and notify operators of any deviations from optimal levels. This enables timely corrective actions, such as inflating the tires to the required pressure, ensuring safer and more cost-efficient operations. Additionally, maintaining proper tire pressure contributes to environmental sustainability by improving fuel economy.

5. Accident Detection and Alerting

Collisions and significant impacts during bus operations can pose serious risks to passenger safety and result in extensive damage to vehicles. Accident detection and alerting systems utilize vibration sensors to identify strong impacts indicative of a collision. Once detected, the system automatically sends notifications to relevant authorities, fleet managers, and emergency services via the Blynk IoT platform. This immediate alert mechanism facilitates swift emergency response, minimizing the potential harm to passengers and reducing the time needed to address the situation. This capability underscores the system's commitment to enhancing safety and operational readiness.

6. Real-Time Location Tracking

Efficient route management and timely service delivery depend on the ability to track bus locations accurately. The integration of GPS modules allows fleet managers to monitor the real-time location of each bus, enabling better coordination and scheduling. Passengers can also benefit from this feature through real-time updates on bus arrival times, enhancing convenience and reliability. This functionality aids in optimizing routes, reducing fuel consumption, and ensuring that buses adhere to their schedules. By providing a transparent and accessible transportation network, real-time location tracking enhances overall service quality.

7. Data Integration and Visualization

Modern fleet management requires a holistic view of operational data to enable informed decision-making. The system integrates data from various sensors, including those for emissions, engine temperature, tire pressure, and stability, and presents it on a unified dashboard using the Blynk IoT

platform. This visualization simplifies the monitoring process by allowing fleet managers to view and analyze data from multiple parameters in a single interface. Alerts and historical data are also accessible, enabling trend analysis and predictive maintenance. The comprehensive and user-friendly dashboard facilitates proactive management and quick responses to any operational issues.

8. Enhanced Fleet Management

A unified system that consolidates real-time monitoring, data visualization, and alerting mechanisms empowers fleet managers to oversee operations more effectively. This objective focuses on optimizing resource allocation, minimizing downtime through preventive maintenance, and enhancing safety protocols. By addressing potential issues before they escalate, the system reduces operational costs and improves the reliability of bus services. Additionally, the insights derived from integrated data support strategic planning, such as route optimization and maintenance scheduling, ensuring that fleet operations remain efficient and passenger-centric. Enhanced fleet management contributes to the long-term sustainability and scalability of the public transportation system.

1.4 Motivation

The increasing complexity and demands of urban transportation systems serve as a key motivator for developing advanced solutions to address the challenges faced by public transportation fleets. With cities expanding and populations growing, the need for efficient, safe, and environmentally sustainable transportation has never been more critical. Traditional systems struggle to cope with the requirements of real-time monitoring and proactive management, often leading to inefficiencies, safety concerns, and non-compliance with environmental regulations.

The motivation for this project stems from the pressing need to integrate modern IoT technologies into public transportation systems to revolutionize fleet management. Ensuring passenger safety, reducing operational costs, and minimizing environmental impact are core drivers. By leveraging advancements in sensors, microcontrollers, and cloud-based platforms, this project aims to bridge the gap between outdated fleet management practices and the demands of smart, connected cities. Moreover, the project seeks to address the critical areas of concern in transportation, such as real-time emission monitoring to combat urban air pollution, accident detection for rapid emergency response, and tire pressure monitoring to prevent accidents. These features contribute to making public transportation more reliable and appealing, encouraging greater adoption by commuters and reducing reliance on private vehicles. Ultimately, the motivation lies in creating a solution that enhances operational efficiency, ensures passenger safety, and supports the vision of sustainable urban mobility.

Another motivating factor is the growing expectation of safety and reliability from public transportation users. Passengers trust buses to provide safe and efficient transit, but incidents such as vehicle breakdowns or accidents undermine this trust. Current systems often fail to address critical safety parameters, such as vehicle stability or accident detection, in a timely manner. By leveraging IoT technologies to monitor these factors in real-time, this project aims to foster greater passenger confidence and satisfaction. Ensuring seamless operations and safety through advanced monitoring systems not only enhances the commuter experience but also aligns with the vision of building a connected and resilient urban infrastructure.

2. LITERATURE SURVEY

1. IoT-Based Bus Fleet Monitoring and Management System

Wang et al. (2021) explored a robust IoT-based solution to enhance the monitoring and management of bus fleets, focusing on leveraging real-time data from multiple integrated sensors. This system collects critical performance metrics, including location data via GPS, engine health, and fuel consumption. The integration of this data into a centralized platform enables operators to optimize routes and improve resource allocation, significantly enhancing operational efficiency. By minimizing delays and optimizing fuel usage, the system reduces operational costs while maintaining high service quality. Wang et al. emphasize how this solution addresses challenges in urban transportation, such as overcrowded routes and inconsistent schedules, while also providing real-time fleet performance insights to managers. This comprehensive approach makes their system a critical advancement in smart transportation and public transit management.

2. Real-Time Vehicle Emission Monitoring

Sawant et al. (2021) focused on an IoT-enabled system designed to address the environmental impact of public transportation. This research presents a solution to monitor emissions from vehicles in real time, ensuring compliance with increasingly strict air quality regulations. Using advanced gas sensors, the system detects and records concentrations of harmful pollutants like carbon monoxide (CO), hydrocarbons, and nitrogen oxides (NOx). Sawant et al. highlighted the importance of proactive emission management, allowing operators to take immediate actions, such as engine tuning or route adjustments, to reduce pollutant levels. Their system is crucial for reducing the carbon footprint of urban transportation and supports broader environmental initiatives aimed at mitigating climate change. By ensuring buses remain within regulatory emission limits, the system also helps operators avoid penalties while promoting eco-friendly transportation solutions.

3. Vehicle Accident Detection and Monitoring System

Singh et al. (2021) introduced a novel approach to accident detection using IoT technologies. Their system integrates vibration and impact sensors to identify collisions or significant impacts in real-time. The moment an accident occurs, the system triggers automatic alerts that are sent to relevant authorities, emergency responders, and fleet operators. This rapid notification mechanism ensures quicker response

times, potentially saving lives and minimizing property damage. Singh et al. also discussed the challenges of traditional accident reporting, such as delayed communication and lack of precise location data, which their IoT-based system addresses effectively. By incorporating GPS data with accident detection, the system provides exact incident locations, streamlining rescue and repair efforts. This innovation enhances overall road safety and contributes to the development of a more secure public transportation network.

4. Real-Time Monitoring of Vehicle Tire Pressure

Yang et al. (2020) developed an IoT-based tire pressure monitoring system that underscores the importance of maintaining optimal tire conditions for safety and efficiency. The system continuously measures air pressure within the tires using advanced pressure sensors and communicates deviations from standard levels in real time. Low tire pressure can lead to increased rolling resistance, higher fuel consumption, and the risk of blowouts, while overinflation can reduce tire lifespan and affect vehicle stability. Yang et al.'s research highlights how their system prevents accidents and improves operational efficiency by enabling operators to address tire-related issues proactively. The integration of this technology into fleet management systems not only reduces maintenance costs but also enhances passenger safety and ensures smoother vehicle performance.

5. Vehicle Emission Control System

El-Bendary et al. (2020) introduced a cutting-edge emission control system designed to monitor and manage vehicle emissions in real time. By employing IoT-enabled gas sensors and microcontrollers, the system records pollutant levels and provides actionable feedback to operators. This real-time data allows fleet managers to identify buses that exceed emission thresholds and take immediate corrective actions, such as scheduling maintenance or rerouting the bus to less congested areas. The study emphasizes the significance of such systems in urban environments where air quality regulations are stringent. El-Bendary et al. also discussed the potential for integrating their system with government monitoring platforms, enabling transportation authorities to enforce environmental standards more effectively. Their solution represents a significant step toward sustainable urban mobility, aligning with global efforts to combat air pollution and climate change.

6. Smart Vehicle Tracking System

Shinde et al. (2020) presented a comprehensive IoT-based tracking system that integrates various sensors to provide detailed insights into vehicle performance and location. The system employs GPS modules for real-time tracking and additional sensors to monitor critical parameters such as engine health, fuel efficiency, and driver behavior. By consolidating this data onto a centralized platform, fleet managers gain a holistic view of their operations. Shinde et al. emphasized how this integration improves route optimization, minimizes delays, and enhances overall fleet efficiency. The study also highlights the system's ability to detect and prevent potential issues, such as mechanical failures, before they escalate into costly repairs. This proactive approach not only reduces operational downtime but also ensures a more reliable transportation experience for passengers.

7. Vehicle Stability and Accident Prevention

Verma et al. (2021) focused on developing an IoT-based system to monitor vehicle stability and prevent accidents. Using accelerometers and gyroscopic sensors, the system continuously tracks the bus's movement and identifies unusual patterns, such as sharp tilts or excessive swaying. These indicators of instability could be caused by factors like uneven road surfaces, sudden turns, or high-speed maneuvers. Upon detecting such conditions, the system sends real-time alerts to the driver and fleet managers, allowing immediate corrective action. Verma et al. also explored how integrating this system with accident prevention algorithms could further enhance safety. By predicting potential hazards and enabling proactive responses, the system significantly reduces the likelihood of accidents. Their research highlights the critical role of stability monitoring in ensuring passenger safety and improving driver accountability in public transportation systems.

3. PROBLEM ANALYSIS & DESIGN

3.1 Block diagram and working principle of proposed system

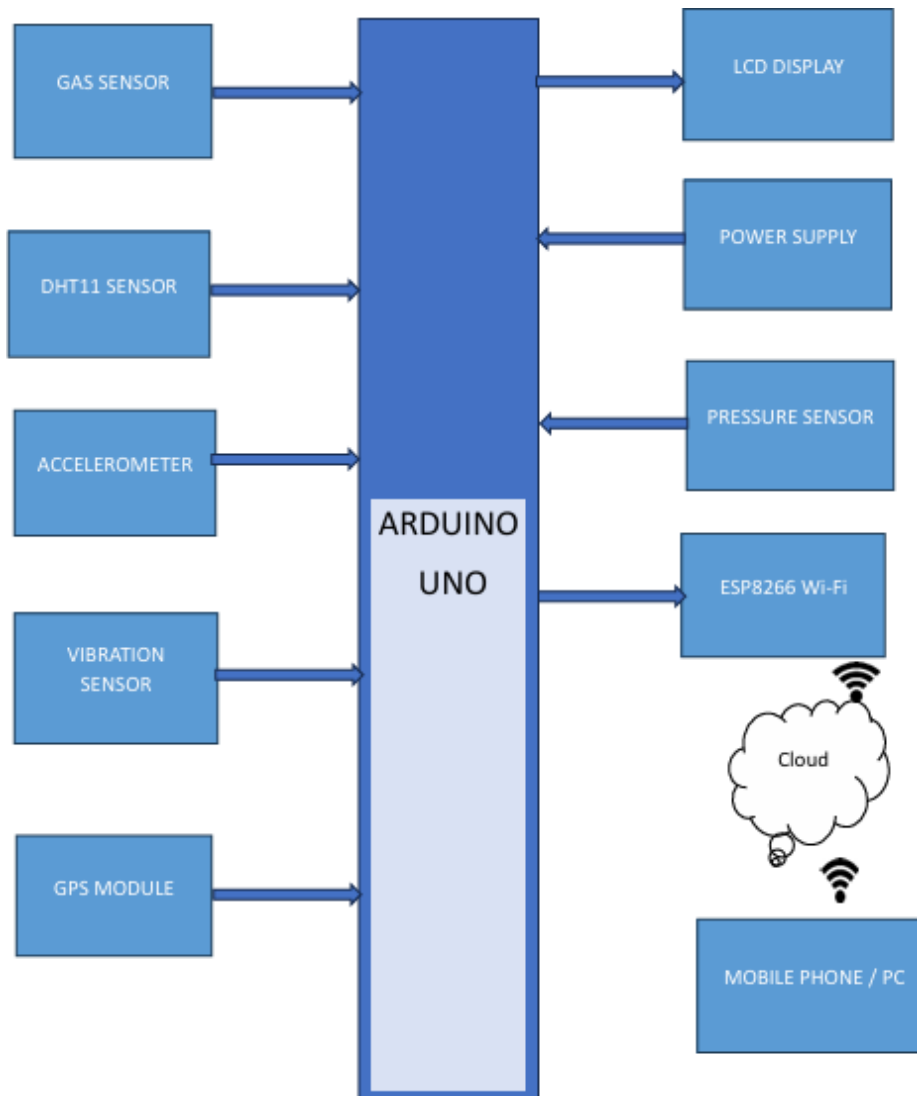


Fig. 1: Block Diagram of Proposed System

3.2 Circuit diagram of Proposed system

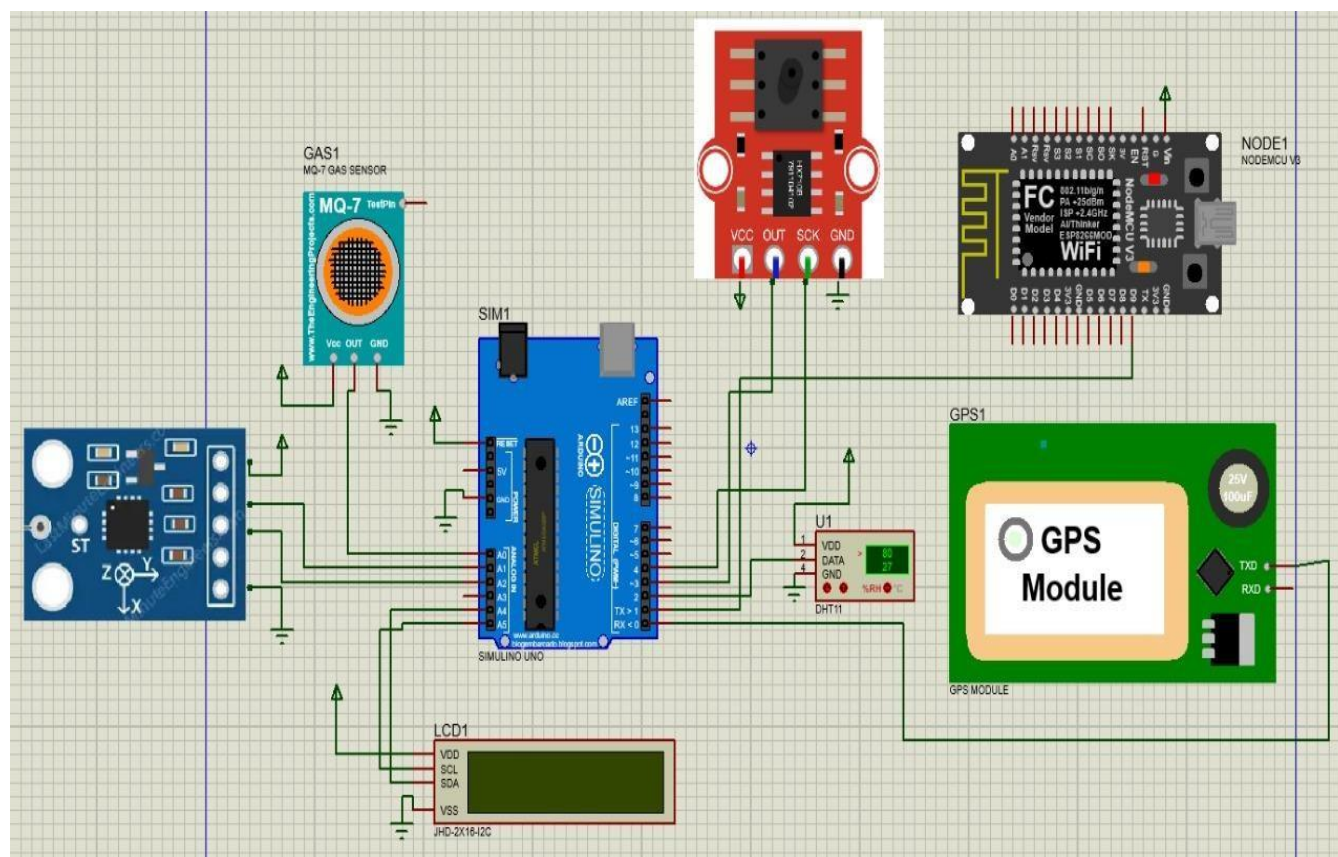


Fig. 2: Circuit Diagram of Proposed System

3.3 Hardware Design

1. Arduino Uno

The Arduino Uno is a widely-used open-source microcontroller board built around the ATmega328P microcontroller. It is widely used in various DIY electronic projects, prototyping, and educational purposes due to its ease of use, affordability, and extensive community support.

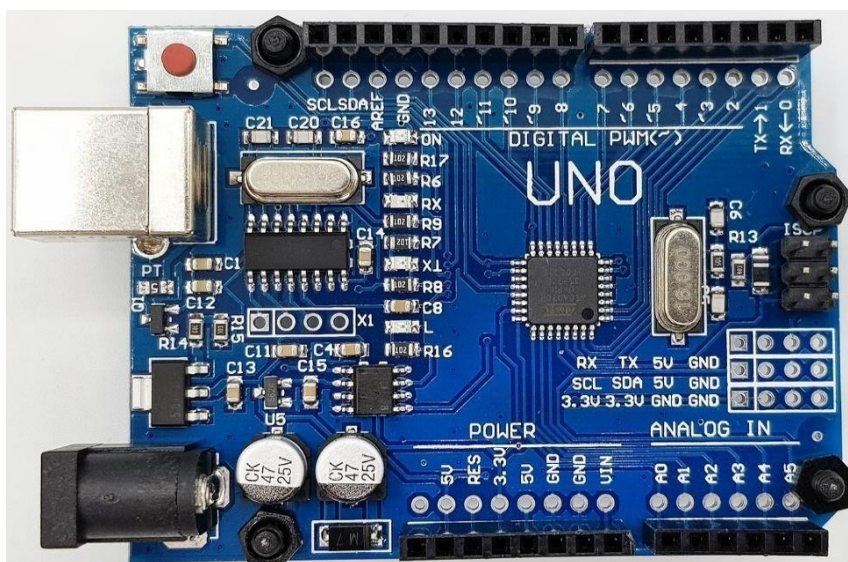


Fig.3. Arduino Uno

Table 1: Arduino Specifications:

Specification	Details
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA

Flash Memory	32 KB (ATmega328P), 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Features

- **Power Supply:** It can be powered via the USB connection or with an external power supply (such as a battery or AC-to-DC adapter). The board automatically selects the appropriate power source.
- **Programming:** The Uno is programmed using the Arduino Software (IDE), which is compatible with Windows, Mac OS X, and Linux. It supports the C/C++ programming language.
- **Resettable Polyfuse:** The board includes a resettable polyfuse that protects the USB ports of your computer from shorts and overcurrent.
- **ICSP Header:** The In-Circuit Serial Programming (ICSP) header allows for programming the microcontroller without using the bootloader.
- **Compatibility:** The Uno is compatible with most Arduino shields, which are modular boards that extend its functionality.

Applications

- **Prototyping:** Widely used for creating prototypes of electronic devices.
- **Educational Projects:** Ideal for teaching electronics and programming.
- **DIY Projects:** Suitable for hobbyists creating interactive projects.
- **Automation and Control:** Used in various automation systems, including home automation, robotics, and IoT applications.

2. WIFI MODULE

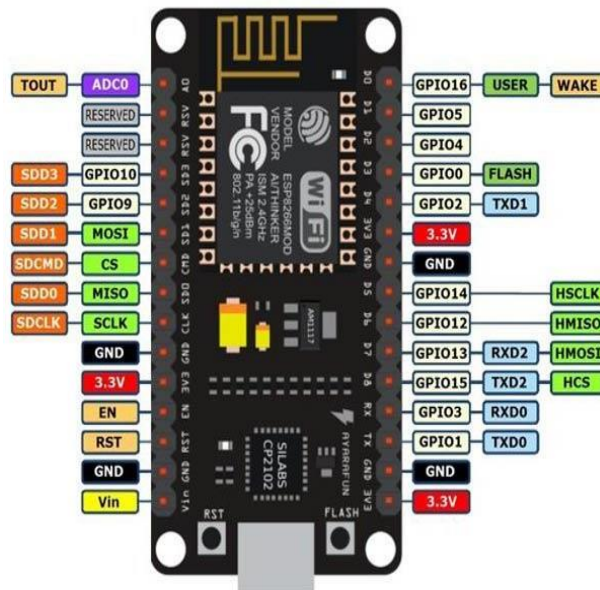


Fig.4. ESP8266 Wi-Fi Module

The ESP8266 is an affordable Wi-Fi microchip with an integrated TCP/IP stack and microcontroller functionality developed by Espressif Systems. Its popularity in IoT projects stems from its cost-effectiveness, user-friendliness, and powerful features.

ESP8266 Specifications:

1. Microcontroller:

- 32-bit RISC CPU: Tensilica L106 running at 80 MHz (can be overclocked to 160 MHz)
- Memory: 32 KB instruction RAM, 80 KB user-data RAM, 16 KB ETS system-data RAM
- External QSPI flash: typically 512 KB to 4 MB

2. Connectivity:

- Wi-Fi: 2.4 GHz IEEE 802.11 b/g/n
- Integrated TR switch, balun, LNA, power amplifier, and matching network
- WEP, WPA/WPA2 security
- Supports AP (Access Point), STA (Station), and AP+STA modes

3. I/O:

- GPIO pins: Up to 17
- ADC: 10-bit ADC (1 channel, up to 1V input)
- Interfaces: SPI, I²C, I²S, UART (HSUART), PWM

4. Power Management:

- Voltage: 3.0V to 3.6V
- Low power consumption:
 - Standby mode: < 1.0 mW (DTIM3)
 - Deep sleep mode: ~10 μ A
 - Light sleep mode: ~0.9 mA
 - Modem-sleep mode: ~15 mA (average, depending on data traffic)

5. Networking:

- TCP/IP protocol stack
- DHCP, DNS, HTTP, HTTPS, SSL/TLS, SMTP
- Supports Smart-Config for simple Wi-Fi configuration

6. Development:

- SDK: Espressif provides an SDK for easy development
- Programming: Can be programmed using the Arduino IDE, NodeMCU, MicroPython, or Espressif's own SDK

3. Power Supply

The board can be powered through three methods: the USB port (providing 5V), the direct current (DC) power connector (ranging from 7V to 12V), or the voltage input pin (also within 7V to 12V). Connecting power to the 3.3V or 5V pins bypasses the voltage regulator and could potentially damage the board.

4. GPS Module



Fig.5. GPS Module

The NEO-6MV2 is the name of the GPS (Global Position System) module that is used in satellites for navigation. All that the module does is find its location on Earth and output the latitude and longitude of that position. It is part of a series of standalone GPS receivers that have the potent u-box 6 position engine. These versatile and reasonably priced receivers have a large selection of connecting options and are compact (16 x 12.2 x 2.4 mm).

NEO-6 modules' small design, power, and memory options make them ideal for battery-operated cell phones with extremely strict financial and spatial constraints. Because of its creative design, the NEO-6MV2 can perform remarkably effectively in even the most challenging navigational conditions.

Applications

- Global Positioning System (GPS) usage
- Mobile phones and tablets
- Guidance and navigation systems
- Unmanned aerial vehicles (UAVs)
- Do-it-yourself (DIY) and hobbyist projects

NEO-6MV2 GPS Module Pin Configuration

The four output pins on the module will each get a description of their respective functions underneath. These four pins are used to power the communication interface and module.

Table 2: NEO-6MV2 GPS Module Pin Configuration

Pin Name	Description
VCC	Positive power pin
RX	UART receive pin
TX	UART transmit pin
GND	Ground

Features

1. **Standalone GPS Receiver:** The module operates independently as a GPS receiver, which means it does not require additional hardware to function for basic GPS operations.
2. **Anti-jamming Technology:** This feature enhances the receiver's ability to maintain signal accuracy and reliability even in environments with interference or signal disruptions.
3. **UART Interface:** The module provides a UART (Universal Asynchronous Receiver-Transmitter) interface for communication with other devices. Additionally, you can use SPI (Serial Peripheral Interface), I²C (Inter-Integrated Circuit), and USB by soldering pins directly to the chip core.
4. **Time-to-First-Fix:**
 - Cold Start: 32 seconds, which is the time required to acquire satellite signals and determine location from scratch.
 - Warm Start: 23 seconds, used when the GPS module has recent satellite data but is not currently active.
 - Hot Start: Less than 1 second, where the module quickly acquires satellite signals due to recent and active data.
5. **Receiver Type:** The module features support for 50 channels and operates on the GPS L1 frequency band. It is also compatible with various Satellite-Based Augmentation Systems (SBAS), including the Wide Area Augmentation System (WAAS), the European Geostationary Navigation Overlay Service (EGNOS), the Multi-functional Satellite Augmentation System (MSAS), and the GPS Aided Geo Augmented Navigation (GAGAN) system.
6. **Maximum Navigation Update Rate:** 5 Hz, which means the module can update its position data up to 5 times per second.
7. **EEPROM with Battery Backup:** Ensures that important data, such as satellite positions and configuration settings, are retained even when the power is off.

Electrical Characteristics:

1. **Default Baud Rate:** 9600 bps (bits per second), which is the standard communication speed for UART data transmission.
2. **Sensitivity:** -160 dBm, indicating the module's ability to detect very weak GPS signals, which enhances its performance in challenging environments.
3. **Supply Voltage:** 3.6V, the operating voltage required to power the module.
4. **Maximum DC Current at Any Output:** 10 mA, the maximum current that the module's output pins can supply without damage.
5. **Operation Limits:**
 - Gravity: Can operate under up to 4g of acceleration.
 - Altitude: Up to 50,000 meters (50 km), making it suitable for high-altitude applications.
 - Velocity: Up to 500 meters per second (approximately 1800 km/h), covering high-speed scenarios such as aircraft or high-speed vehicles.
6. **Operating Temperature Range:** -40°C to 85°C, allowing the module to function reliably across a wide range of environmental conditions.

5. Accelerometer

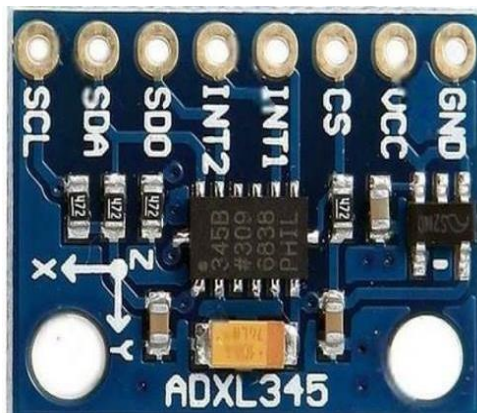


Fig.6. Accelerometer

With I2C and SPI interfaces, the ADXL345 is a compact, low-power, full 3-axis MEMS accelerometer module. The ADXL345 board has an on-board level shifter and 3.3V voltage regulator, which makes interacting with 5V microcontrollers like the Arduino easy.

ADXL345 Module Features:

- DC Supply Voltage range: 3V to 6V
- Integrated low-dropout (LDO) voltage regulator
- Embedded voltage level converter (based on MOSFET technology)
- Compatible with microcontrollers operating at either 3.3V or 5V
- Extremely low power consumption: 40 μ A in active measurement mode and 0.1 μ A in standby mode at 2.5V
- Detection capabilities: Tap and double-tap recognition
- Ability to detect free-fall events
- Supports both SPI and I2C communication protocols
- Acceleration measurement range: $\pm 16g$
- Measurement values for each axis (in g):
- X-axis: -235 to +270
- Y-axis: -240 to +260
- Z-axis: -240 to +270

Applications of ADXL345 Accelerometer

- Affordable, energy-efficient solutions for detecting motion and tilt
- Handheld electronics and smartphones

- Video game consoles and controllers
- Hard drive impact protection
- Camera stabilization systems
- Fitness and wellness monitoring devices

ADXL345 Module Pin Configuration

Table 3: ADXL345 Module Pin Configuration

Pin Name	Pin Configuration
GND	Ground Pin
VCC	Power Supply pin (3V to 6V)
CS	Chip Select Pin
INT1	Interrupt 1 Output
INT2	Interrupt 2 Output
SDO	Serial Data Output
SDA	Serial Data Input & Output
SDL	Serial Communication Clock

6. Vibration Sensor

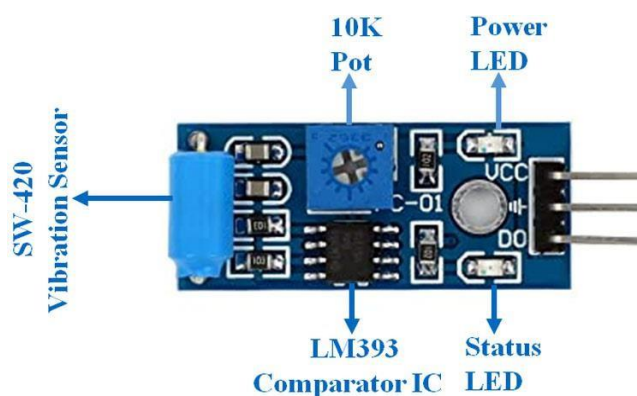


Fig.7. Vibration Sensor

The vibration sensor module based on the **vibration sensor SW-420** and Comparator LM393 is used to detect vibrations. The threshold can adjust using an on-board potentiometer. During no vibration, the sensor provides Logic Low and when the vibration is detected, the sensor provides Logic High.

This Vibration Sensor Module consists of an SW-420 Vibration Sensor, resistors, capacitor, potentiometer, comparator LM393 IC, Power, and status LED in an integrated circuit. It is useful for a variety of shocks triggering, theft alarm, smart car, an earthquake alarm, motorcycle alarm, etc.

Vibration Sensor Module Features & Specifications:

- Operating Voltage: 3.3V to 5V DC
- Operating Current: 15mA
- Using SW-420 normally closed type vibration sensor
- LEDs indicating output and power
- LM393 based design
- Easy to use with Microcontrollers or even with normal Digital/Analog IC
- With bolt holes for easy installation
- Small, cheap and easily available

Applications of Vibration Sensor Module:

- Shocks triggering
- Theft alarm
- Smart car
- Earthquake alarm
- Motorcycle alarm

7. I2 LCD



Fig.8. LCD Display

16X2 LCD Display with IIC I2C interface is an alphanumeric Blue display that can show up to 32 characters on a single screen. You can display more characters by scrolling the texts one by one. We already know that to connect LCD Display directly with the Arduino using 4bit and 8bit modes will utilize many numbers of GPIO Pins of our Arduino or other boards and we would have to end up with less number of pins for other sensors and actuators.

To overcome this problem we use LCD I2C backpack with our LCD. This I2C Backpack uses PCF8574 Remote 8 bit I/O Expander. It translates the data received from the I2C Bus into Parallel data that is needed for the LCD Display.

I2C – Inter-Integrated Circuit:

Inter-integrated Circuit (in short I2C) is a two-wire short distance communication protocol. You can use multiple slave devices in the same two wires with one or more master controllers. You may wonder how does the master identifies which slave does the data to be sent. In I2C the external devices have an I2C

address for different external devices like LCD Backpack, OLED Display, etc. By using the address the data is sent to the specific device connected on the same I2C Bus.

The message is broken into two frames and sent serially via the I2C Bus. The first frame contains the address, once the address matches with any device on I2C bus, that device will send an acknowledge signal to the master. After receiving the acknowledgment from the slave the data bits are sent. By this method an I2C bus works.

16×2 LCD Display:

A LCD Display 16x2 means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in the 5×7 pixel matrix.

16×2 LCD Pinout:

There are totally 16 pins in an LCD Display. You can use directly all the pins in 8-bit mode with Arduino or 12 pins using 4-bit mode. In this tutorial, we use the I2C module for LCD and multiplex it into just 4 pins. This pin details might not be useful while using I2C Method but this is the actual pin details of all the pins in LCD Display.

- Vcc – Power Supply (5v)
- Vdd/GND – Ground
- V0 – Brightness Control using Potentiometer
- RS – Register select. Specify what we are sending Command or Data. Sets to 0 for Command mode like setCursor, LCD Clear, TurnOFF LCD. Set 1 for data mode like sending Data/Characters.
- R/W – Read/Write. Mostly we are writing catalog/characters to the registers.
- E – Enable writing to Registers.
- D0 to D7 – Data pins. Send 4bit/8bit data to display characters. The Arduino library provides 4bit and 8bit mode. The data will be in ASCII format.
- A/LED+ – Anode (Backlight LED)

- K/LED- – Cathode (Backlight LED)

I2C Pinout:

The LCD I2C Backpack only has 4 Pins. They are

- GND – Ground
- VCC – 5V Power Supply
- SDA – Data Line
- SCK – Clock Line

The module has a contrast adjustment pot on the underside of the display. This may require adjusting for the screen to display text correctly.

Features :

- Arduino IIC/I2C interface was developed to reduce the IO port usage on Arduino board
- I2C adapter allows flexibility in connections
- I2C Reduces the overall wirings.
- 16 characters wide, 2 rows
- White text on the Blue background
- Single LED backlight included can be dimmed easily with a resistor or PWM

8. MQ-7 Gas Sensor



Fig.9. MQ-7 Gas Sensor

Features

- High sensitivity to carbon monoxide
- Stable and long life

Application

They are used in gas detecting equipment for carbon monoxide(CO) in family and industry or car.

Specifications:

Table 4A: Standard work condition

Symbol	Parameter name	Technical condition	Remark
Vc	circuit voltage	5V±0.1	Ac or Dc
VH (H)	Heating voltage (high)	5V±0.1	Ac or Dc
VH (L)	Heating voltage (low)	1.4V±0.1	Ac or Dc
RL	Load resistance	Can adjust	
RH	Heating resistance	33Ω±5%	Room temperature
TH (H)	Heating time (high)	60±1 seconds	
TH (L)	Heating time (low)	90±1 seconds	
PH	Heating consumption	About 350mW	

Table 4B. Environment conditions

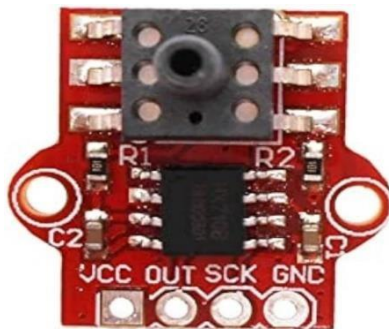
Symbol	Parameters	Technical conditions	Remark
Tao	Using temperature	-20°C-50°C	
Tas	Storage temperature	-20°C-50°C	Advice using scope
RH	Relative humidity	Less than 95%RH	

02	Oxygen concentration	21%(stand condition) The oxygen concentration can affect the sensitivity characteristic	Minimum value is over 2%
----	----------------------	--	--------------------------

Table 4C. Sensitivity characteristic

symbol	Parameters	Technical parameters	Remark
Rs	Surface resistance Of sensitive body	2-20k	In 100ppm Carbon Monoxide
a(300/100ppm)	Concentration slope rate	Less than 0.5	Rs (300ppm)/Rs(100ppm)
Standard working condition	Temperature -20°C±2°C relative humidity 65%±5% RL:10KΩ±5%		
	Vc:5V±0.1V VH:5V±0.1V VH:1.4V±0.1V		
Preheat time	No less than 48 hours	Detecting range: 20ppm-2000ppm carbon monoxide	

9. Hx710B Air Pressure Sensor

**Fig.10. Hx710B Air Pressure Sensor**

HX710B Atmospheric Pressure Sensor Module. This barometric pressure sensor is optimized for altimeters and variometers with an altitude resolution of 10 cm. The sensor module includes a high linearity pressure sensor and an ultra-low power; 24 bit ADC with internal factory calibrated coefficients. It provides a precise digital 24 Bit pressure and temperature value and different operation modes that allow the user to optimize for conversion speed and current consumption.

This HX710B air pressure sensor module uses a high-precision AD sampling chip, adopts a 0-40KPa air pressure sensor, can connect a 2.5mm hose, can detect water level, and other air pressure

Features:

1. Two Selectable Differential Input Channels.
2. On-Chip Active Low Noise PGA.
3. Selectable Gain of 32,64 and 128.
4. On-Chip Power Supply Regulator For Load Cell and ADC Analog Power Supply.
5. On-Chip Power-ON-Reset
6. Simple Digital Control And Serial Interface.

Table 5: Specifications of Hx710B Air Pressure Sensor

Interface Type	Serial
Data Output Rate	10 SPS / 80SPS
Max. Operating Current (mA)	1.5
Operating Temperature (°C)	-40 to 85
IC Package	16 Pin SOP

3.4 Software Design

Arduino IDE

Introduction to Arduino IDE:

The Arduino Integrated Development Environment (IDE) is a powerful and user-friendly platform designed for programming Arduino boards. It provides a comprehensive environment for writing, compiling, and uploading code to Arduino hardware. This software simplifies the process of interacting with microcontrollers, making it accessible to both beginners and advanced users. The Arduino IDE supports various Arduino boards and compatible microcontrollers, enabling a wide range of applications from basic LED blinking to complex robotics projects.

User Interface:

The Arduino IDE features a straightforward user interface that consists of a text editor for writing code, a message area for displaying compilation results and error messages, and a toolbar with essential functions. The editor is equipped with syntax highlighting, auto-indentation, and other features that enhance code readability and ease of writing. The toolbar provides buttons for verifying code, uploading to the board, and accessing various IDE tools and settings.

Programming Language:

The Arduino IDE uses a simplified version of C++ for programming Arduino boards. The language is designed to be approachable for beginners while still powerful enough for advanced users. The IDE includes a set of core libraries that abstract complex functions and simplify the programming process. These libraries cover a wide range of functionalities, including digital and analog I/O, communication protocols, and sensor interfacing.

Code Structure:

Arduino programs, known as sketches, follow a specific structure comprising two main functions: `setup()` and `loop()`. The `setup()` function is executed once at the beginning to initialize settings and configurations, while the `loop()` function runs repeatedly to perform the main tasks of the program. This structure allows for continuous operation and interaction with the hardware, facilitating real-time applications and control.

Libraries and Examples:

The Arduino IDE comes with a rich collection of built-in libraries and example sketches. Libraries provide pre-written code to simplify complex tasks, such as interfacing with sensors, displays, and communication modules. Example sketches serve as starting points for projects and demonstrate how to use various libraries and functions. Users can also install additional libraries from the Arduino Library Manager or third-party sources to extend functionality.

Compilation and Uploading:

Once the code is written, the Arduino IDE compiles it into a binary format that the microcontroller can understand. The compilation process checks for syntax errors and ensures that the code adheres to the requirements of the Arduino platform. After successful compilation, the IDE uploads the binary file to the Arduino board via a USB connection. This process enables the microcontroller to execute the program and interact with connected components.

Debugging and Troubleshooting:

While the Arduino IDE does not have built-in debugging tools, it provides essential features for troubleshooting and debugging. Users can use serial communication to send and receive data between the Arduino board and the IDE, allowing them to monitor variables, display messages, and diagnose issues. The IDE's message area also displays error messages and warnings that can help identify and resolve problems during compilation and uploading.

Community and Support:

The Arduino IDE is supported by a vibrant and active community of users and developers. The Arduino forums, official website, and numerous online resources provide valuable information, tutorials, and support. Users can seek help, share their projects, and collaborate with others to solve problems and improve their skills. The strong community support contributes to the ongoing development and enhancement of the Arduino platform.

Installation of Arduino IDE 2.3.2:

Steps:

1. Go to browser
2. Paste this link on Browser <https://www.arduino.cc/en/software>

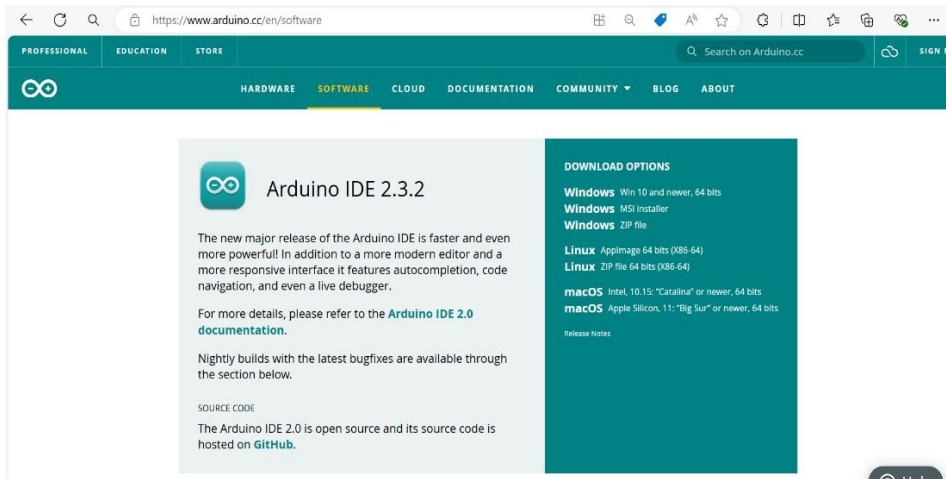


Fig.11.1 Select the software according to your OS

3. Based on your PC OS install the software and Click on Just Download to download free version.
4. After Download open the file

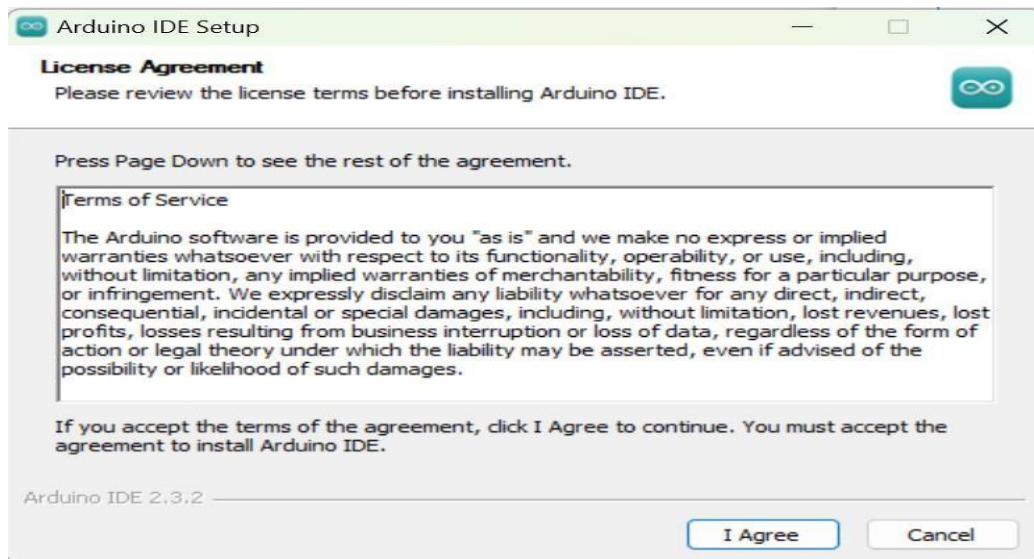


Fig.112 Agree the License agreement

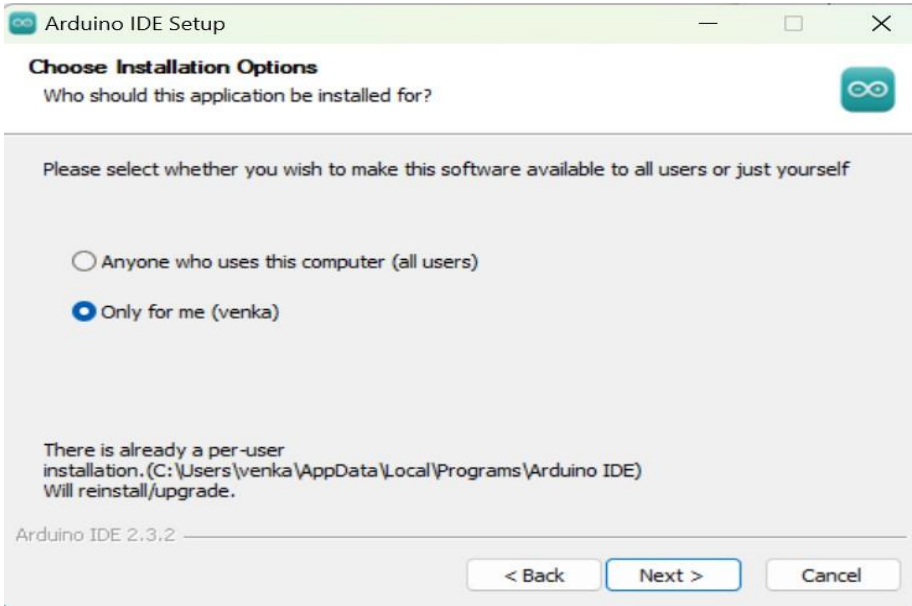


Fig.11.3 Choose Installation option

6. Click on Next

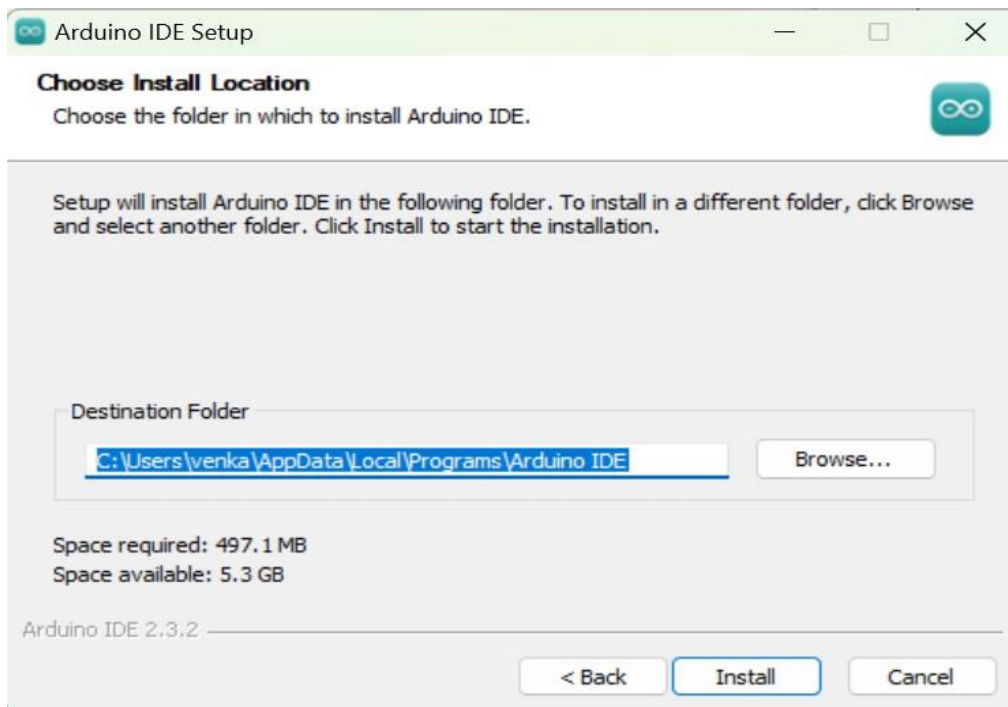


Fig.114 Choose the installation location

7. Click on Install (If you want you can change path)

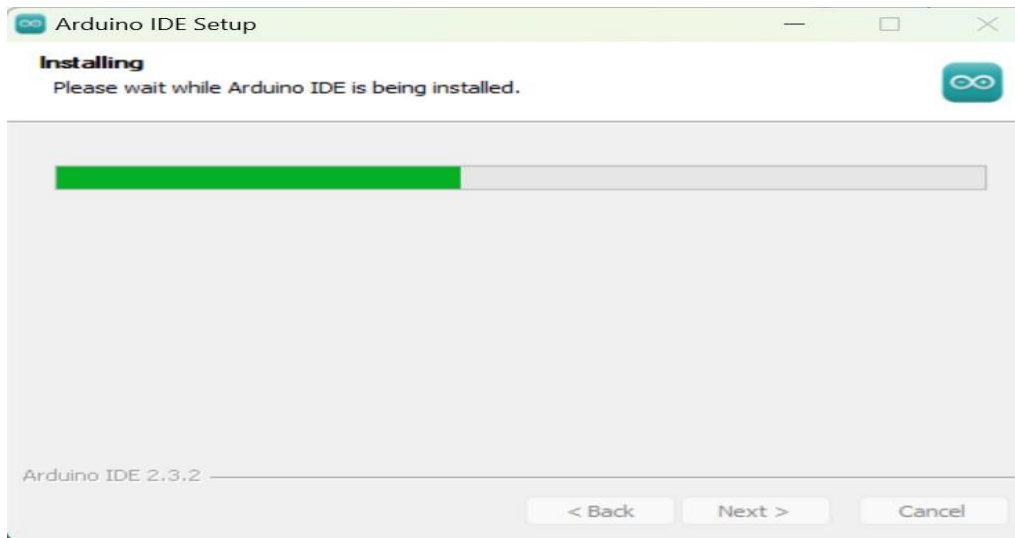


Fig.11.5 Installing the software files

8. We can see Arduino IDE is Installing

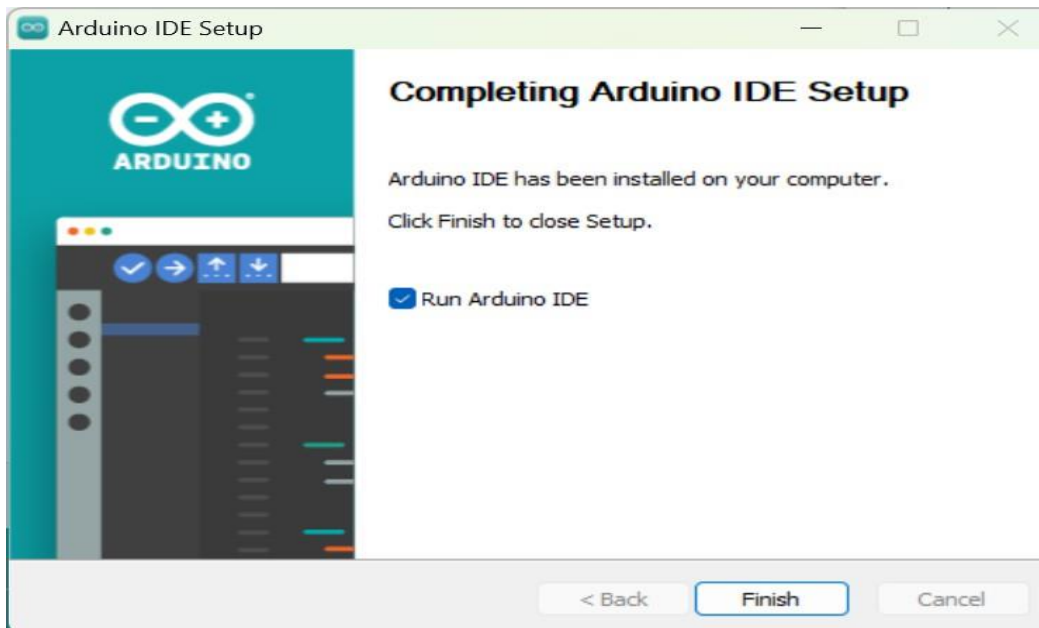


Fig.11.6 Completed Installation

8. Click on Finish

9. Know Arduino IDE is installed successfully

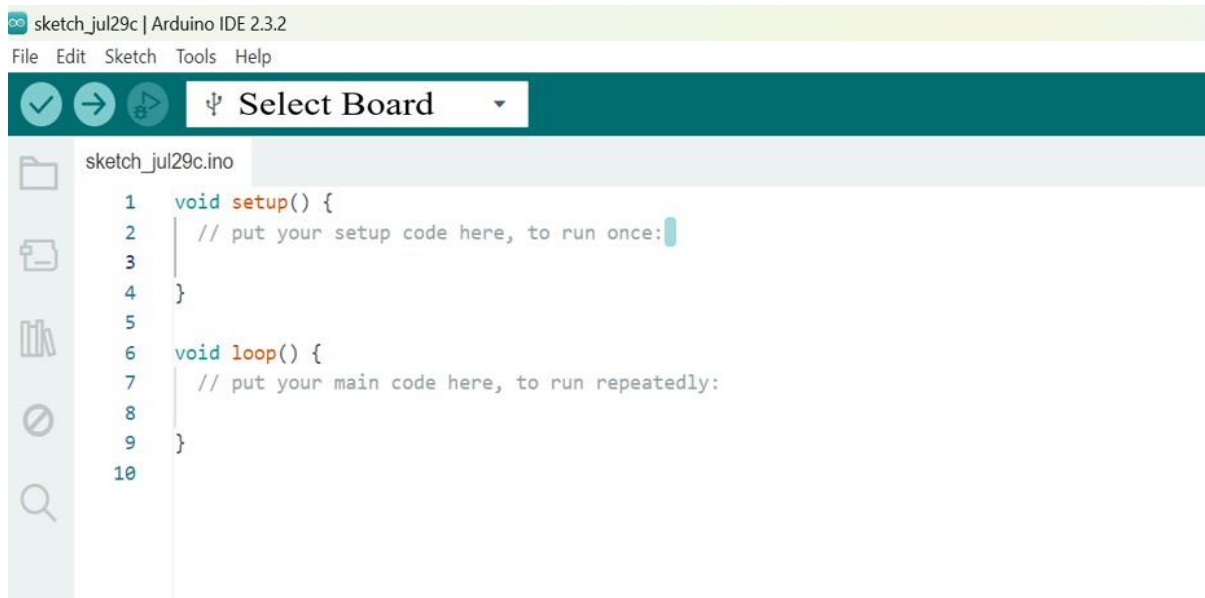


Fig.11.7 Open the Arduino IDE software

10. This page will open, when we open this software initially.

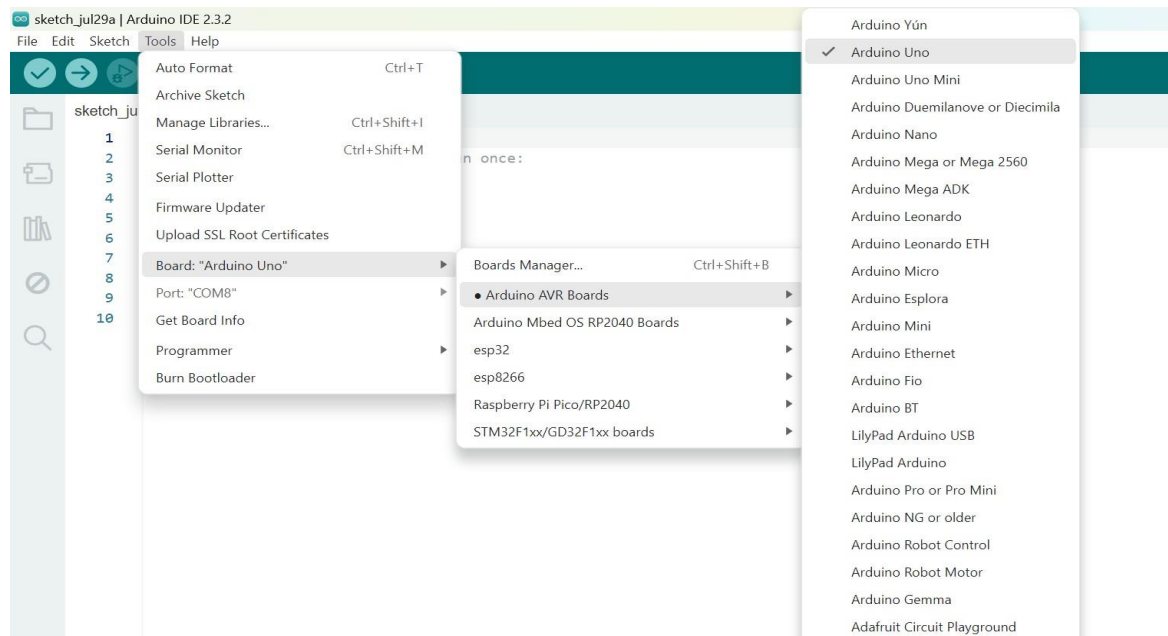


Fig.11.8 Select the board

11. To select the board
12. Goto tools
13. In that Goto board manager
14. In that Goto Arduino AVR Boards
15. Select Arduino Uno

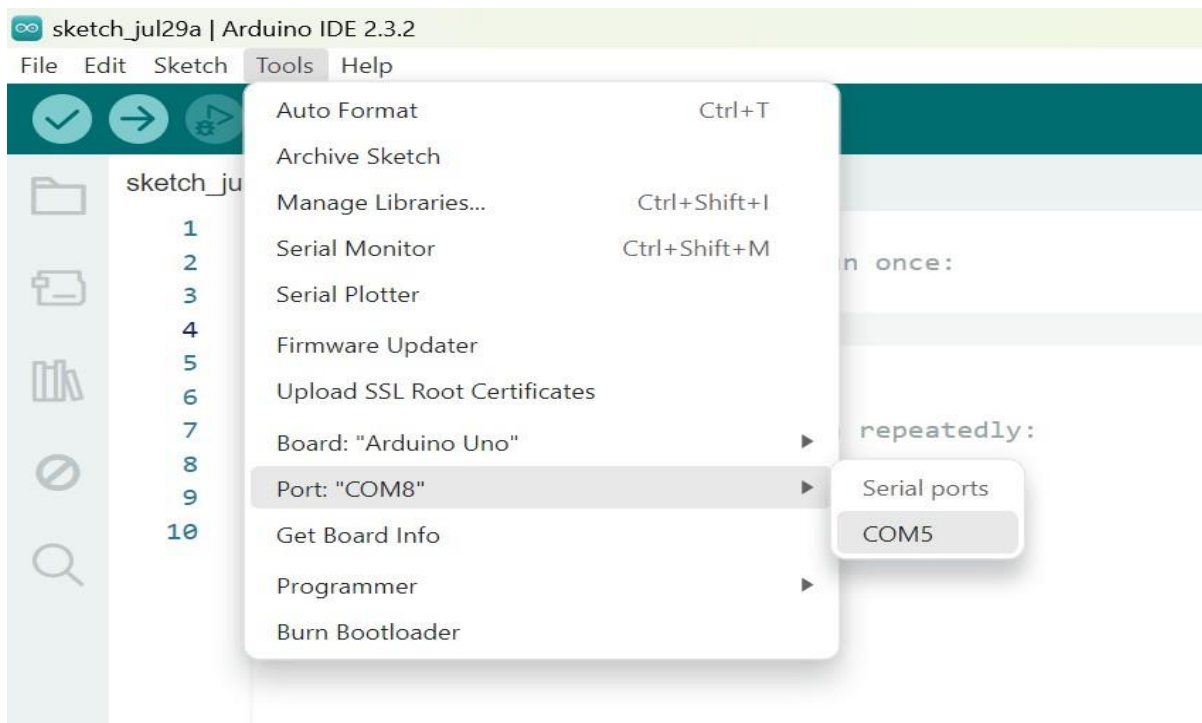


Fig.11.9 Select the port

16. To select port
17. Connect Arduino Uno to PC using Arduino connecting cable
18. Goto tools
19. In that Goto port
20. Select the port in which Arduino Uno is connected

Setting Up the Arduino IDE to Program ESP8266:

To program the NodeMCU using Arduino IDE first we need to install the NodeMCU ESP8266 library.

Steps to install library:

1. Click on File.
2. Go to Preferences and Click on it.
3. Go to settings.
4. Select the sketchbook location as per your convenience.
5. Select editor language as System default.
6. Select editor font size as 12.
7. In Additional board manager URL's copy the below link
http://arduino.esp8266.com/stable/package_esp8266com_index.json
8. Click on OK

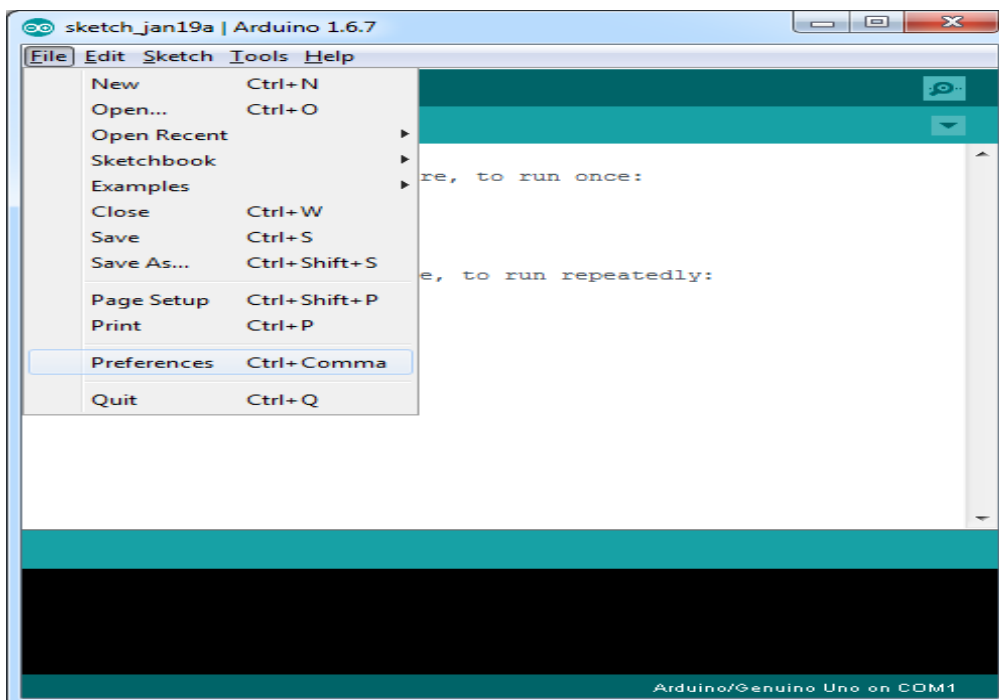


Fig.12.1 Open preferences

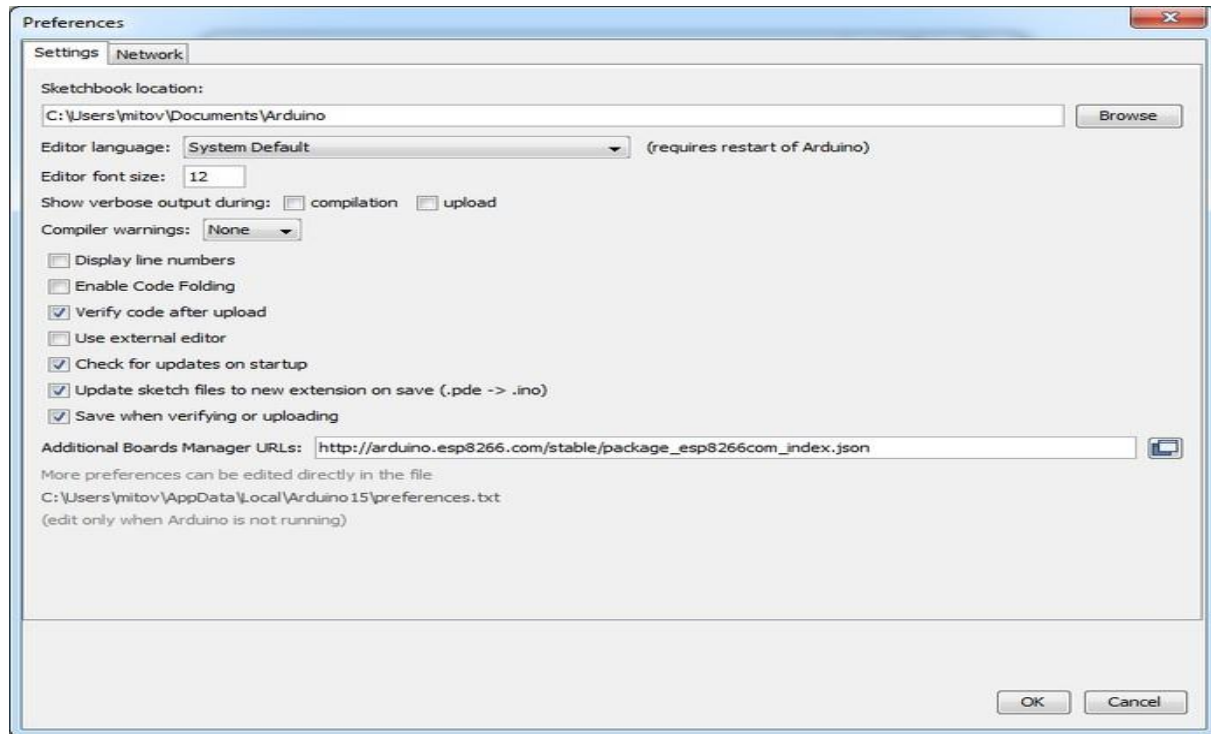


Fig.122 Open Additional Boards Manager URLs

Steps to Install the ESP8266 Board Libraries and Tools:

1. Click on Tools in Arduino IDE.
2. In that go to Board: Arduino/Genuino Uno.
3. In that go to Board Manger Click on it.
4. In Board manager search bar type ESP.
5. Then select esp8266 by ESP8266 community.
6. Click on Install.
7. After installation Click on close.

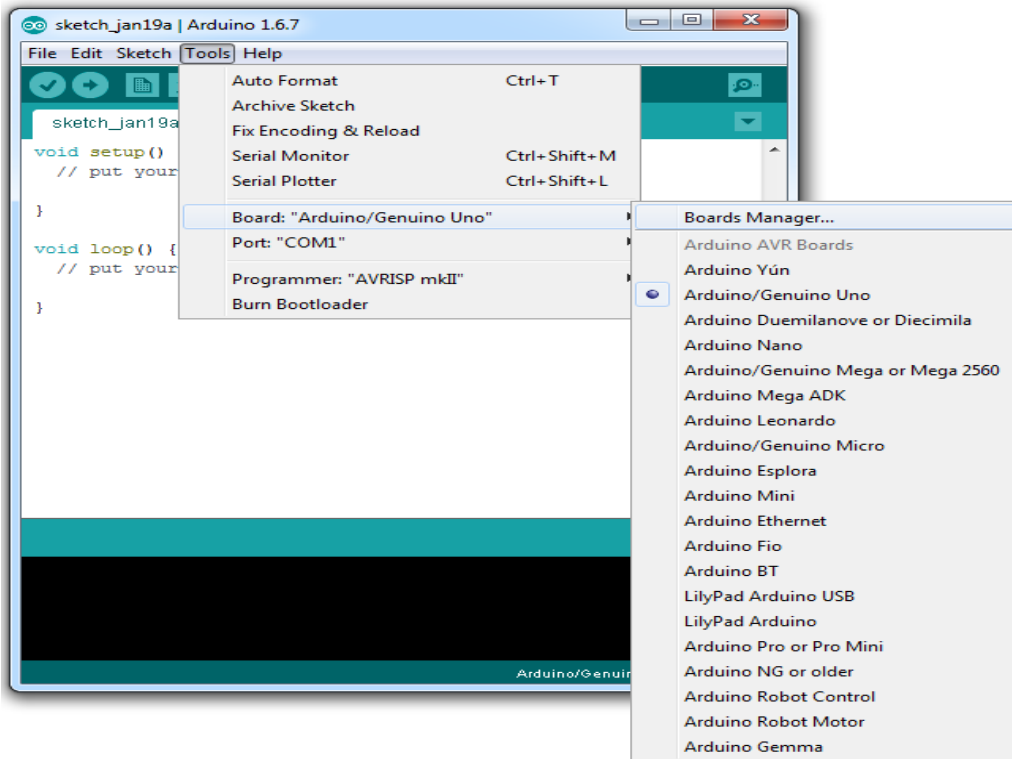


Fig.12.3 Open Boards Manager

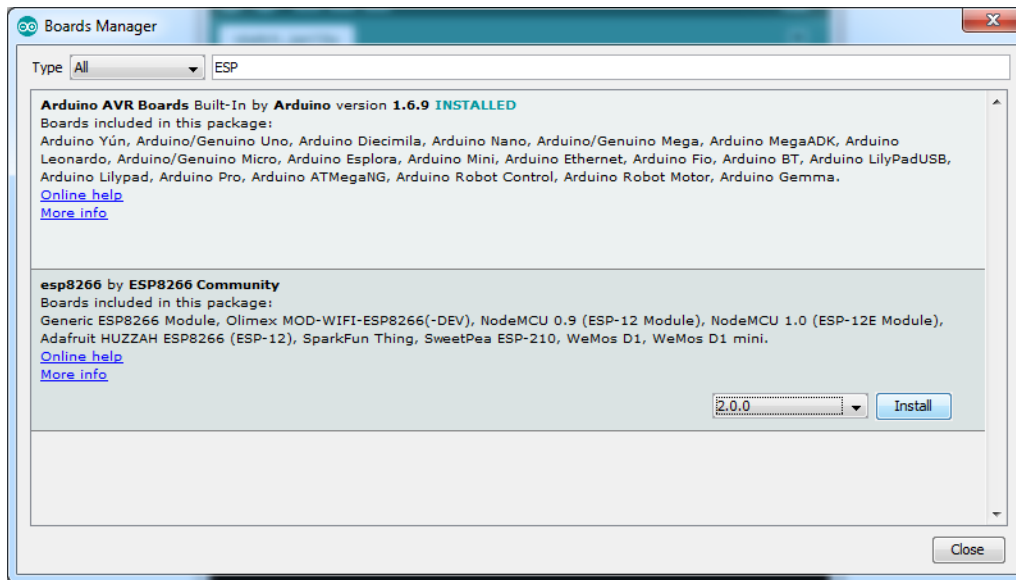


Fig.12.4 Type ESP

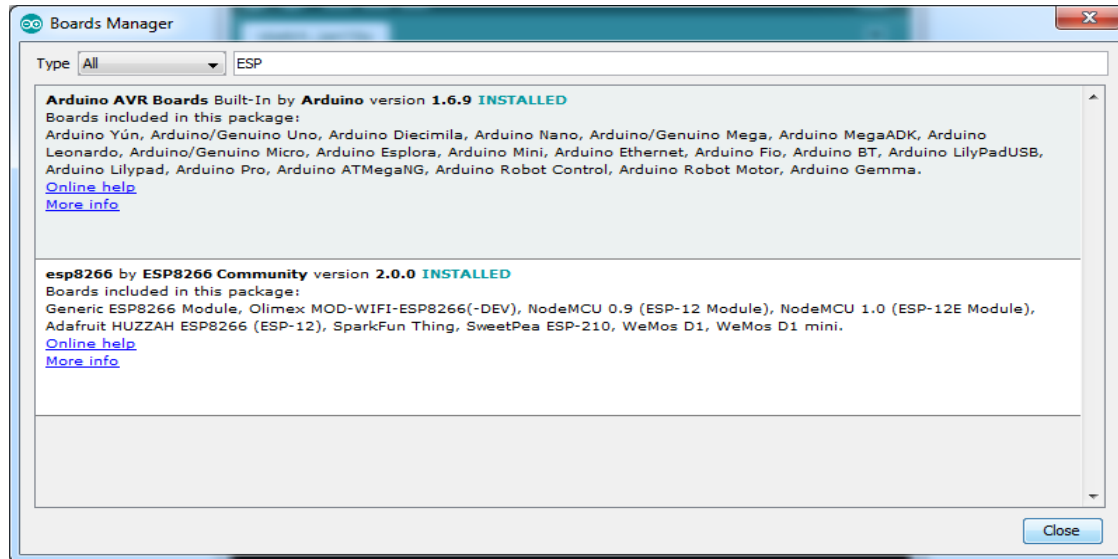


Fig.12.5 Click on install esp8266 file

Steps to Test the ESP8266 With Arduino Project:

1.Steps to select the microcontroller:

1. Click on Tools
2. In that go to Board.
3. In that select the microcontroller board as NodeMCU 0.9(ESP-12 Module) or NodeMCU 1.0(ESP-12E Module).

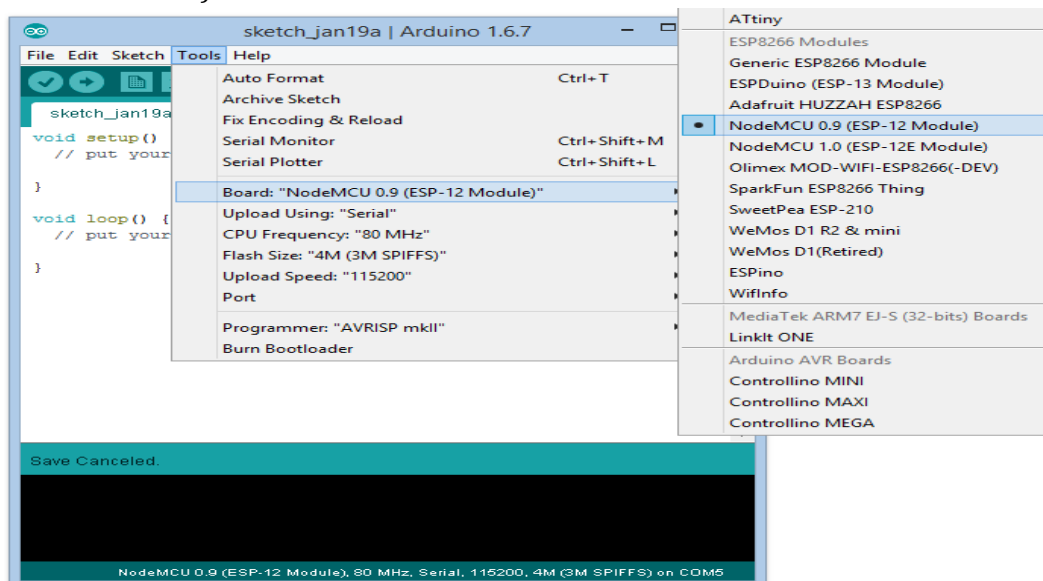


Fig.12.6 Select the Board as Node MCU 0.9 or 1.0

2.Steps to select the Port (to upload):

1. Click on tools.
2. Go to port, select the port in which you have connected between microcontroller to PC.
3. Write the program or code on Arduino IDE.

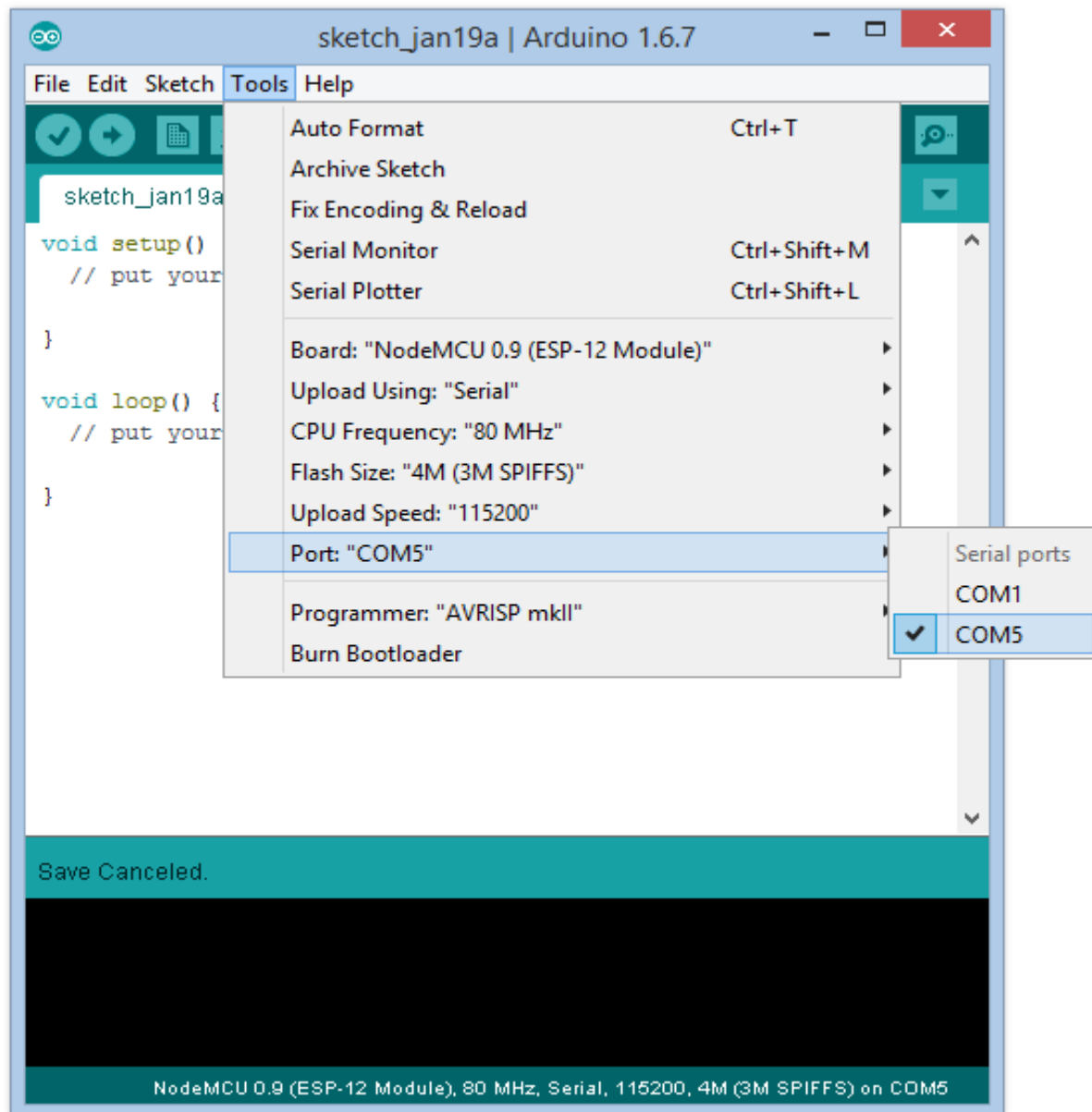


Fig.12.7 Select the port

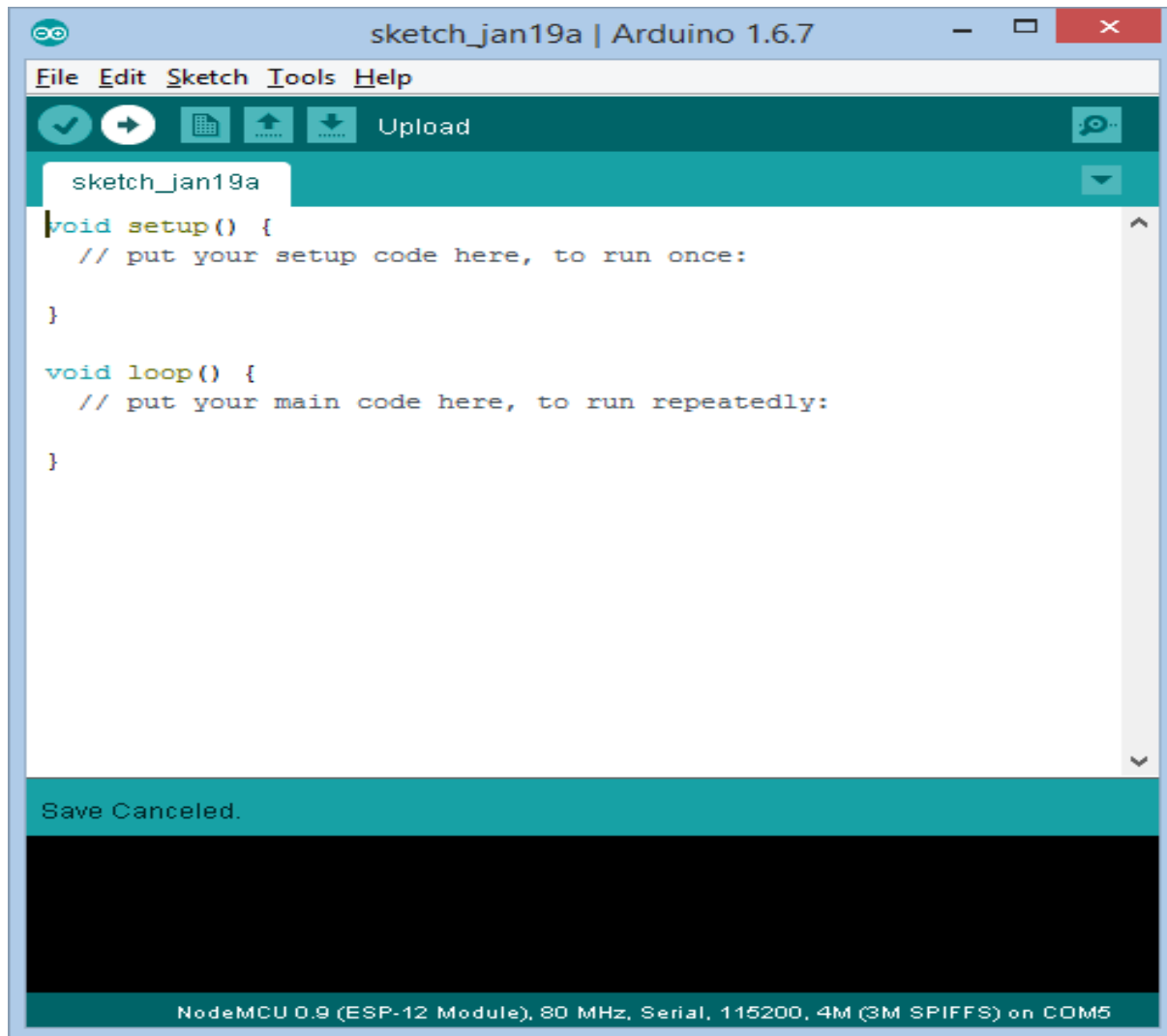


Fig.12.8 Open the Arduino IDE software

EMBEDDED C

Embedded C is a specialized programming language extension of the C programming language, designed for use in embedded systems. These systems are typically resource-constrained, with limited memory, processing power, and input/output capabilities. Embedded C provides the necessary tools and capabilities to manage these constraints effectively. Here's a detailed explanation:

Introduction to Embedded C

Embedded C is a set of language extensions for the C programming language to address the needs of embedded system developers. It is widely used in the development of firmware for microcontrollers and other small devices. The primary goal of Embedded C is to allow developers to write code that can interact directly with the hardware while maintaining the efficiency and flexibility of the C language.

Basic Features

Embedded C retains the syntax and structure of standard C but includes additional features to handle hardware-specific tasks. These features include direct access to hardware via memory-mapped registers, bit manipulation, and the ability to disable and enable interrupts. These capabilities make Embedded C an ideal choice for programming microcontrollers and other embedded devices.

Hardware Interaction

One of the main advantages of Embedded C is its ability to interact directly with hardware. This interaction is often achieved through special functions or by manipulating hardware registers directly. For example, to turn on an LED connected to a specific pin of a microcontroller, an Embedded C program can write to the corresponding port register.

Memory Management

Embedded systems often have very limited memory, so efficient memory management is crucial. Embedded C provides mechanisms to manage memory effectively, such as static memory allocation, which can help avoid the overhead and fragmentation associated with dynamic memory allocation. This is particularly important in real-time systems where predictable performance is required.

Real-Time Constraints

Many embedded systems operate under real-time constraints, meaning they must respond to events within a strict time frame. Embedded C supports real-time programming through the use of interrupts and timers. Developers can write interrupt service routines (ISRs) that execute in response to specific hardware events, ensuring timely processing of inputs.

Efficiency

Efficiency is a key concern in embedded systems, where processing power and memory are limited. Embedded C allows for low-level optimizations, such as inline assembly, to take full advantage of the hardware capabilities. Developers can write highly efficient code that runs faster and uses less memory, which is essential for battery-powered devices.

Development Tools

The development of Embedded C programs typically involves specialized tools, such as integrated development environments (IDEs), compilers, and debuggers tailored for embedded systems. These tools often include simulators and in-circuit emulators (ICEs) that help developers test and debug their code on real hardware.

Safety and Reliability

Embedded systems often perform critical functions, so safety and reliability are paramount. Embedded C includes features to enhance reliability, such as volatile keywords to prevent compiler optimizations from removing necessary hardware interactions. Additionally, developers use rigorous testing and validation techniques to ensure the reliability of embedded software.

Industry Applications

Embedded C is used in a wide range of industry applications, from consumer electronics to automotive systems, industrial automation, and medical devices. Its ability to provide low-level hardware access, efficient memory management, and real-time capabilities makes it suitable for developing robust and efficient embedded software for various applications.

BLYNK APP

Introduction to Blynk IoT Application

Blynk is a versatile Internet of Things (IoT) platform designed to facilitate the creation and management of connected projects. It provides a user-friendly interface and a range of tools to enable developers and hobbyists to quickly build and deploy IoT solutions. With Blynk, users can design custom mobile applications to control and monitor their IoT devices from anywhere in the world. The platform is compatible with a wide range of hardware, including popular microcontrollers like Arduino, ESP8266, and Raspberry Pi, making it accessible for various applications.

Core Features

The core features of the Blynk IoT platform include a customizable mobile app, a cloud server, and an extensive library of pre-built widgets. The mobile app allows users to create a virtual dashboard with widgets such as buttons, sliders, and gauges that interact with their hardware. The cloud server ensures seamless communication between the mobile app and the IoT devices, handling data transmission and control commands. Blynk also offers a robust library of libraries and example projects to help users get started quickly and integrate their devices with the platform.

Hardware Integration

Blynk supports a wide range of hardware platforms, making it highly versatile for different IoT projects. Integration with hardware is facilitated through Blynk's device-specific libraries, which simplify the process of connecting devices to the Blynk cloud. For instance, users can connect an Arduino board to the Blynk platform using a compatible Ethernet or Wi-Fi shield, while ESP8266 and ESP32 modules can connect directly via Wi-Fi. This hardware flexibility allows users to build various applications, from home automation to industrial monitoring.

User Interface Design

One of Blynk's standout features is its intuitive drag-and-drop interface for designing mobile app dashboards. Users can create and customize their app's layout by simply dragging widgets onto the screen and configuring their properties. This visual design approach eliminates the need for complex programming, making it accessible for non-technical users. Additionally, the mobile app provides real-

time feedback and control over connected devices, allowing users to monitor and manage their IoT projects efficiently.

Cloud Connectivity

Blynk's cloud server plays a crucial role in facilitating communication between the mobile app and IoT devices. The server ensures that data and commands are transmitted reliably and securely over the internet. Users can access their IoT devices from anywhere, thanks to the cloud-based architecture. The platform also supports data storage and retrieval, enabling users to track historical data and analyze trends over time. This cloud connectivity feature enhances the flexibility and scalability of IoT applications.

Security Measures

Security is a critical aspect of IoT applications, and Blynk incorporates several measures to protect user data and device interactions. The platform uses encryption for data transmission between the mobile app, cloud server, and IoT devices. Additionally, Blynk provides options for user authentication and access control, ensuring that only authorized individuals can interact with the devices. By implementing these security features, Blynk helps safeguard user information and prevent unauthorized access to IoT systems.

Applications and Use Cases

Blynk's IoT platform is used in a diverse range of applications, from smart home automation to industrial monitoring and environmental sensing. Users can automate household devices, monitor energy consumption, control security systems, and even track weather conditions using Blynk. The platform's versatility and ease of use make it a popular choice for both personal and professional IoT projects. Whether for hobbyists exploring new ideas or businesses developing commercial solutions, Blynk offers the tools and flexibility to bring IoT concepts to life.

Installation of Blynk IoT Application:

Steps:

1. Goto Browser
2. Paste this link on browser <https://blynk.io>
3. Click ON Sign UP

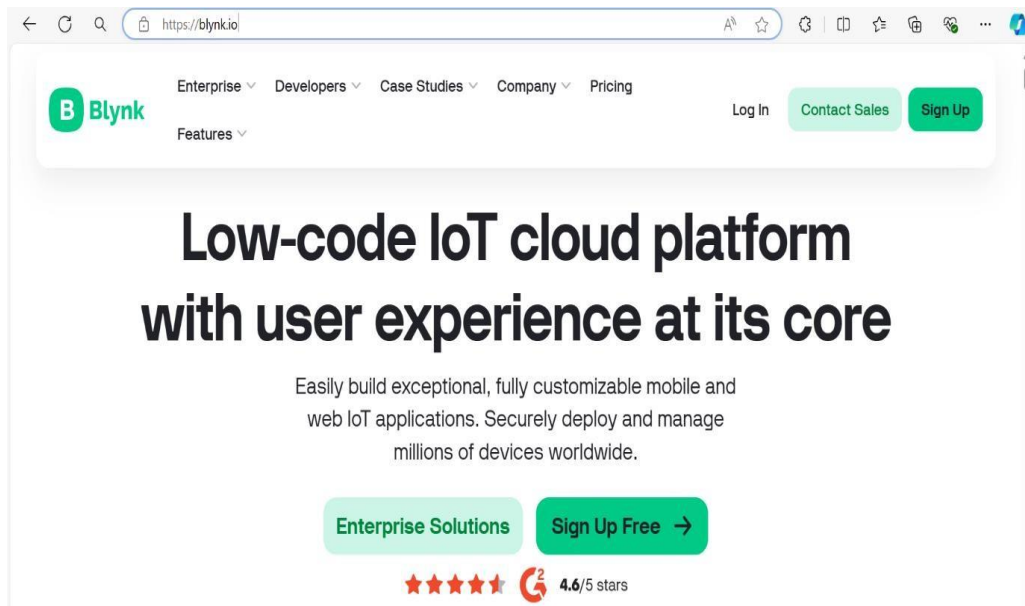


Fig.13.1. Blynk IoT

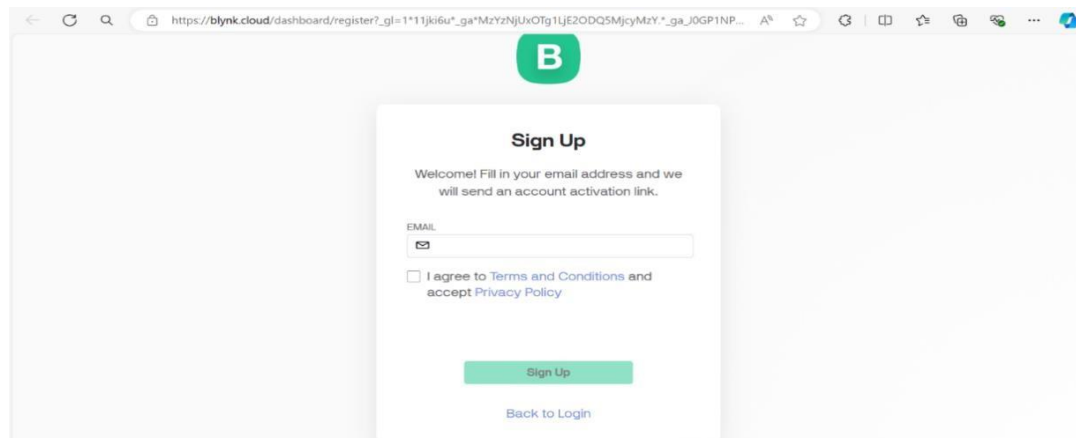


Fig.13.2 Sign UP page

4. Sign in with your email address

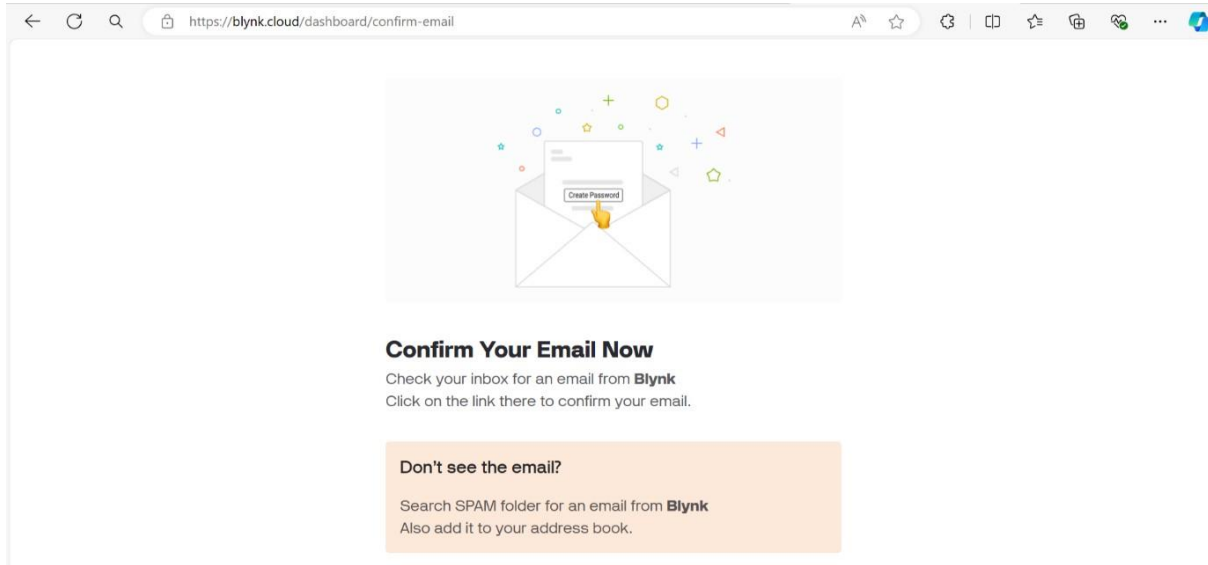


Fig.13.3 Confirm email

5. Confirm Your Email

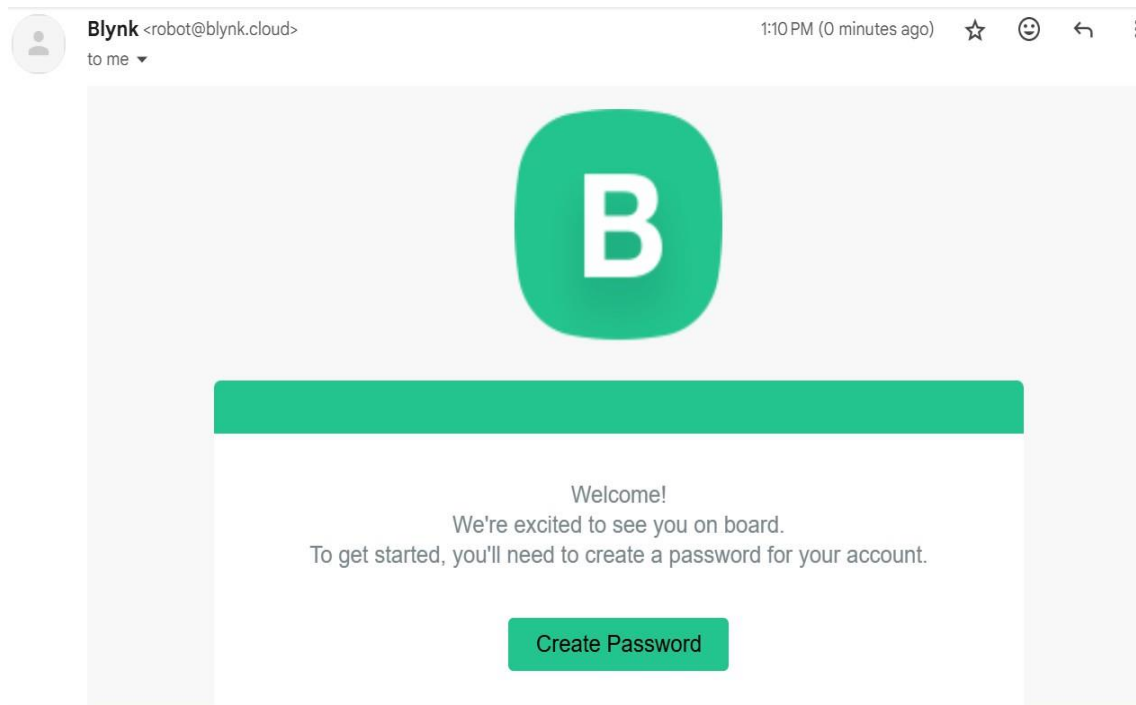
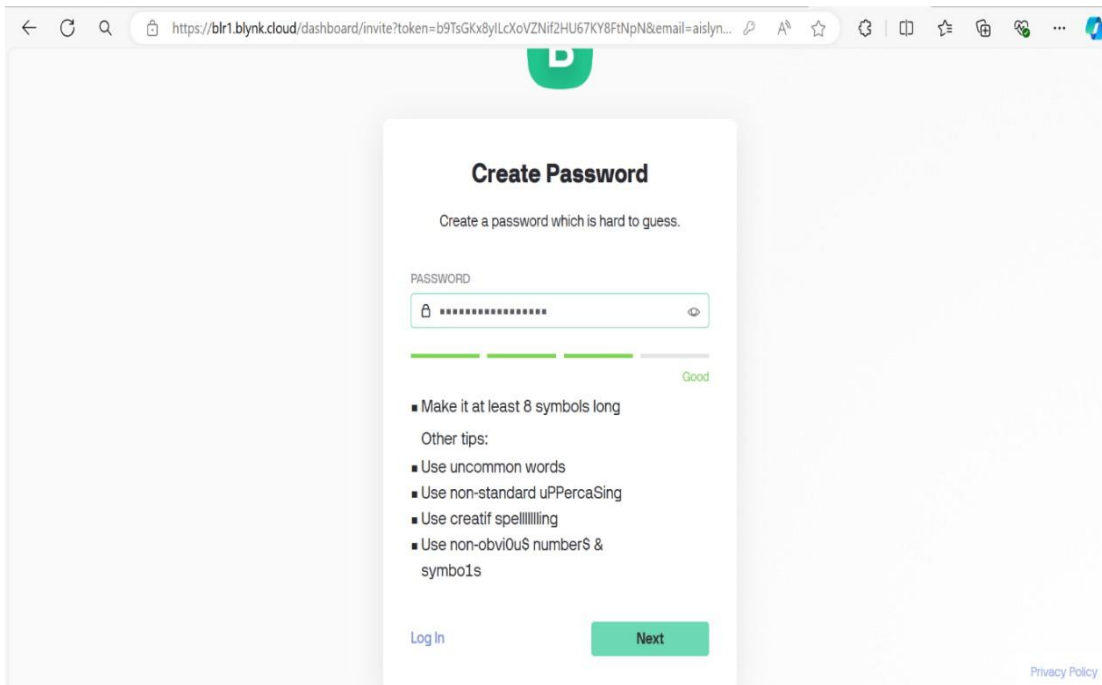



Fig.13.4 Click on Create Password

6. Create password



← ↻ 🔍 https://blr1.blynk.cloud/dashboard/invite?token=b9TsGKx8ylLcXoVZNif2HU67KY8FtNpN8&email=aislyn...



Create Password

Create a password which is hard to guess.

PASSWORD

🔒 •••••••••••••••• 🔍

Good

- Make it at least 8 symbols long

Other tips:

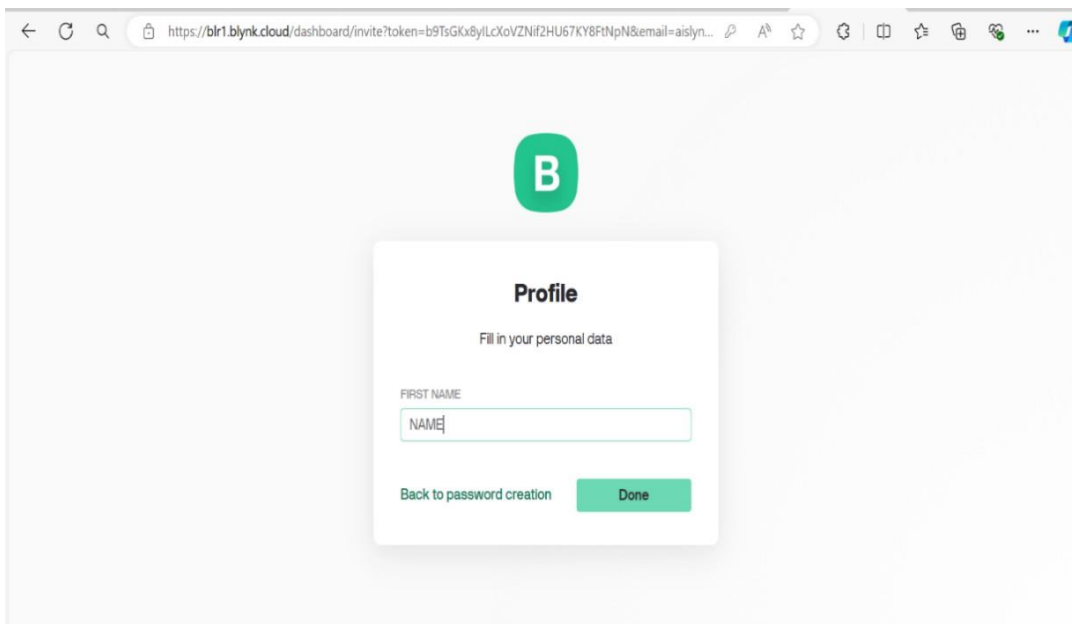
- Use uncommon words
- Use non-standard uPPerCaSing
- Use creatif spelllllllling
- Use non-obvi0u\$ number\$ & symbo1s

[Log In](#) [Next](#)


[Privacy Policy](#)

Fig.13.5 Create Strong password

7. Provide Your Name



← ↻ 🔍 https://blr1.blynk.cloud/dashboard/invite?token=b9TsGKx8ylLcXoVZNif2HU67KY8FtNpN8&email=aislyn...



Profile

Fill in your personal data.

FIRST NAME

NAME

[Back to password creation](#) [Done](#)

Fig.13.6 Enter first name

8. Create New template

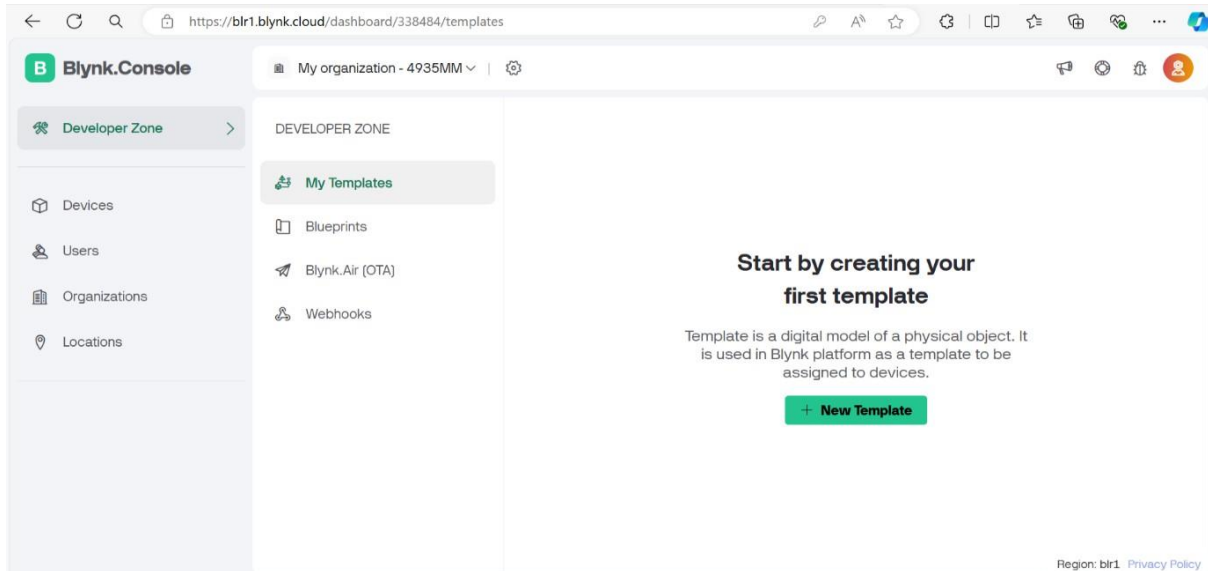


Fig.13.7 Enter first name

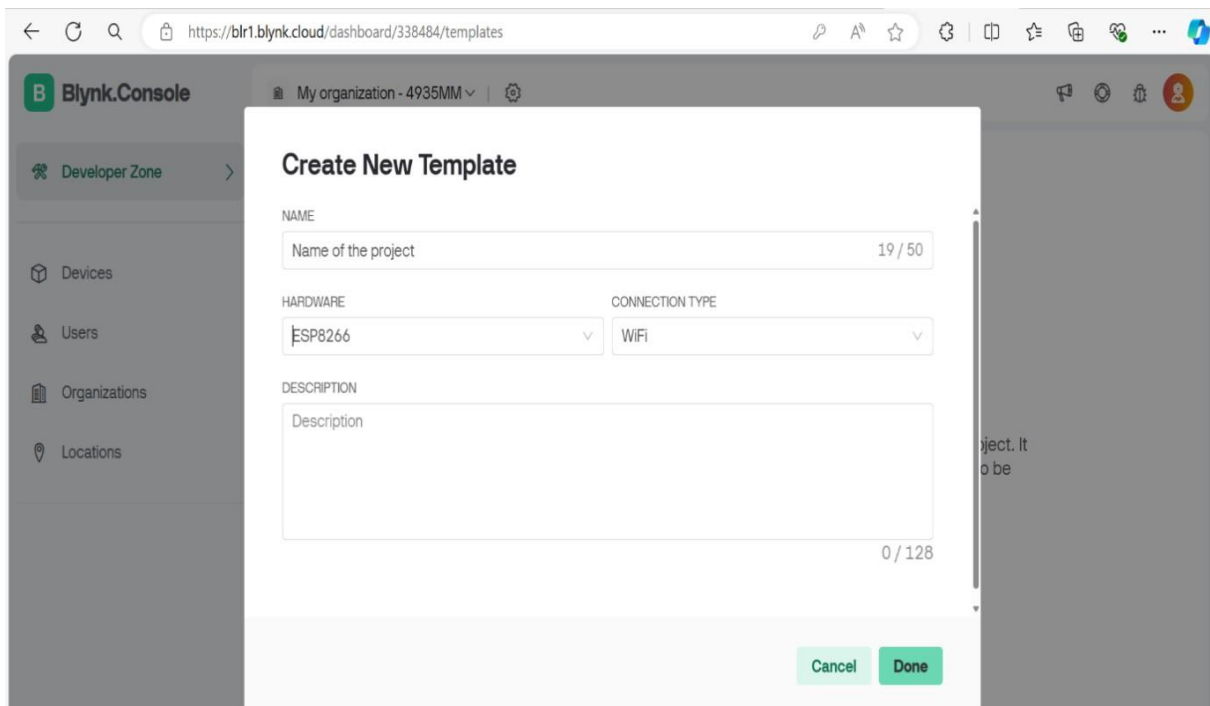


Fig.13.8 Enter the project name and select hardware

9. Goto Events and Notifications

10. Create new event

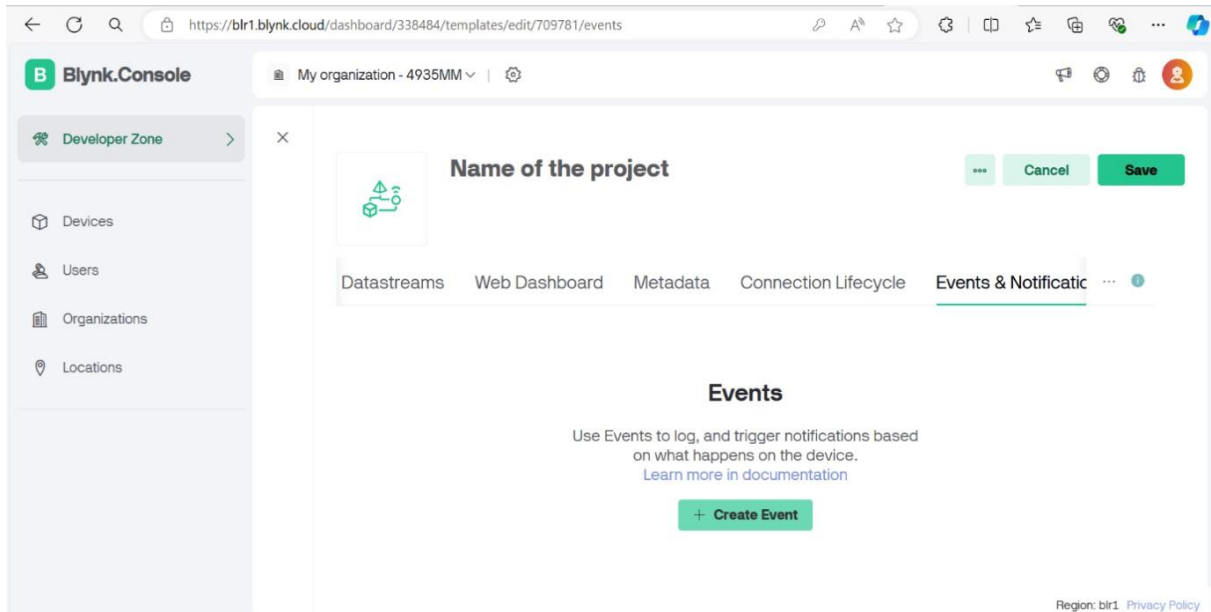


Fig.13.9 Create event

11. Add New Event

12. Provide event name

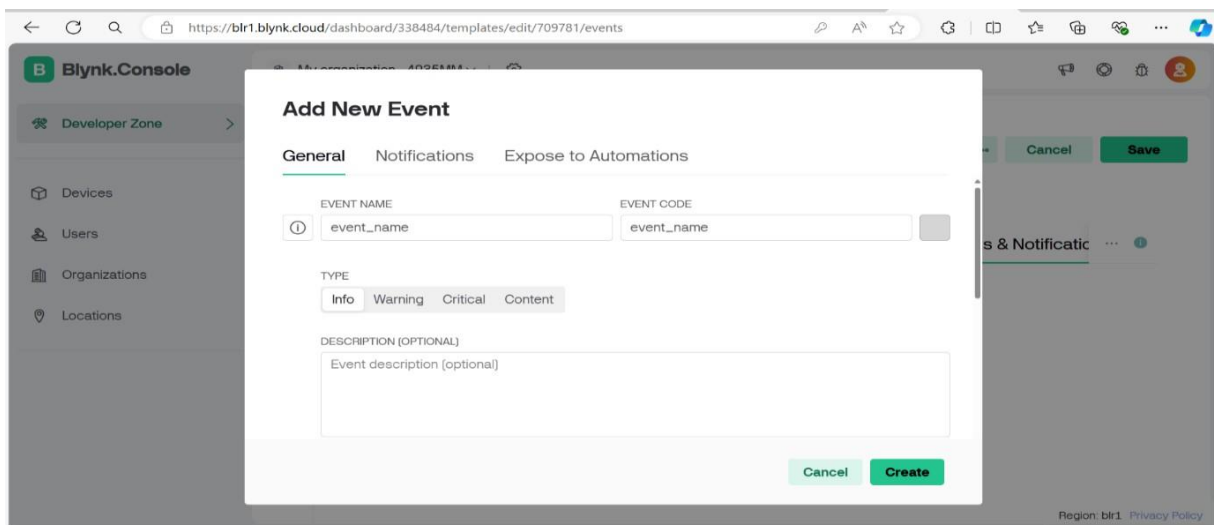


Fig.13.10 Enter the Event name

13. Allow show event in notifications section of mobile app

14. Allow send event to time line

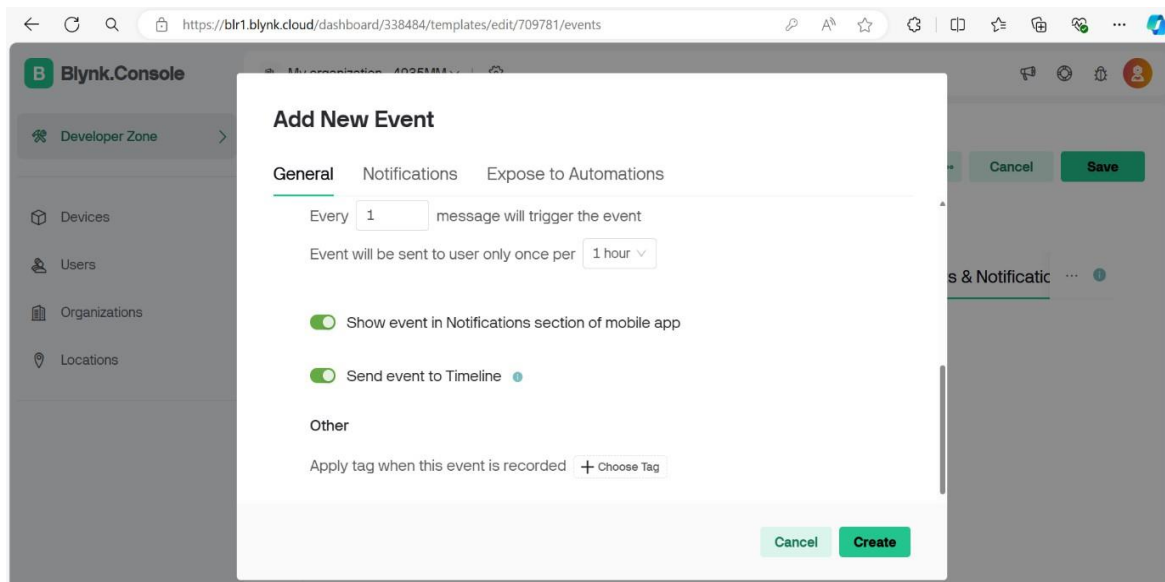


Fig.13.11 Click Enable

15. Go to notifications

16. Allow enable notifications

17. E-MAIL TO Device owner

18. PUSH NOTIFICATIONS TO Device owner

19. Allow enable notification management

20. save

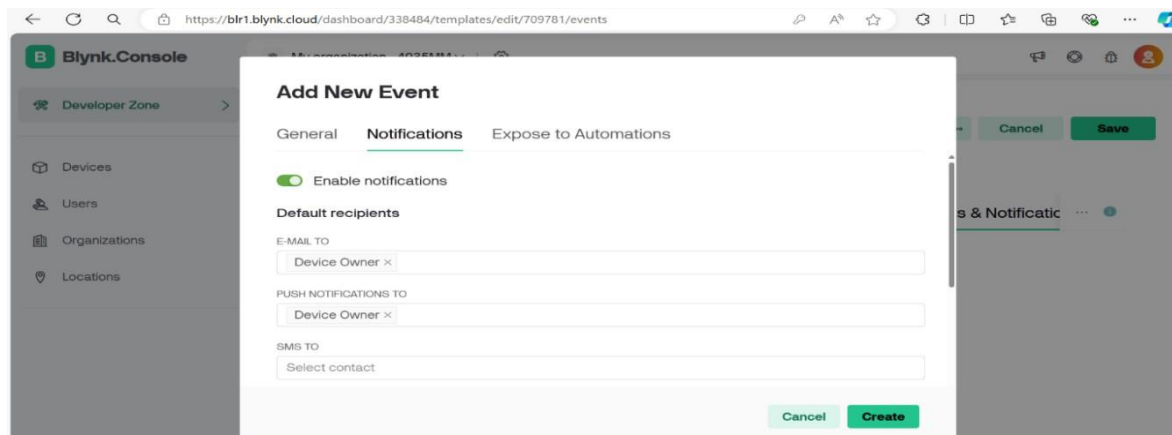


Fig.13.12 Select as Device Owner

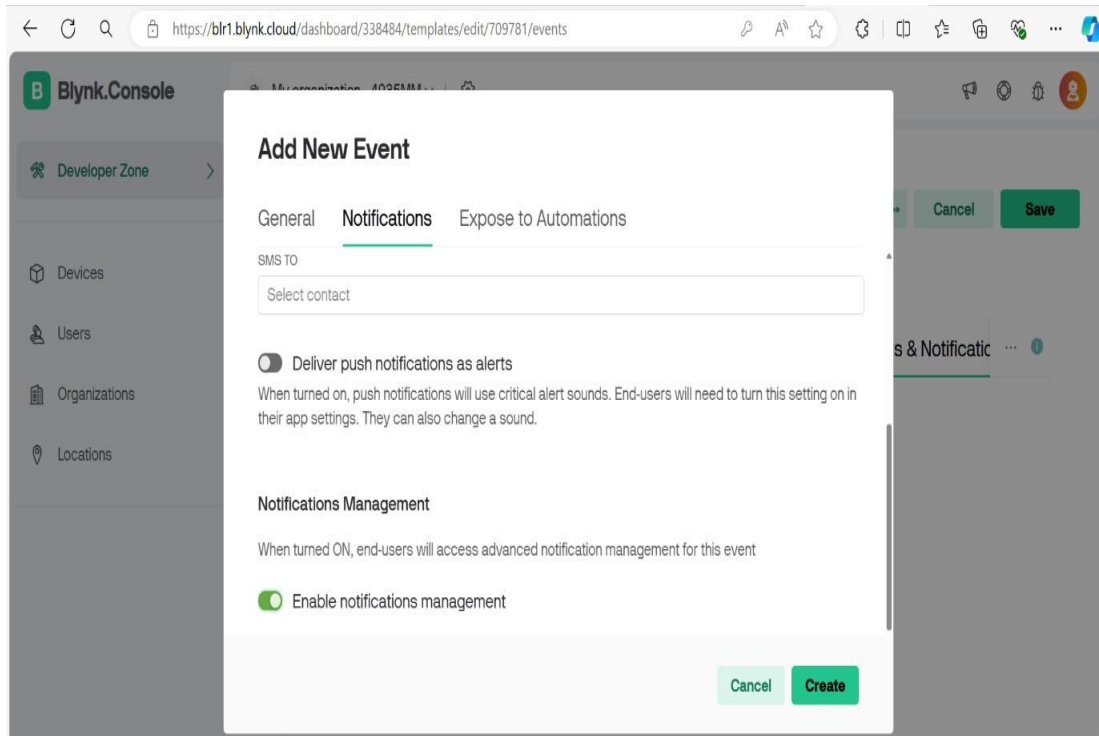


Fig.13.13 Enable Notification management

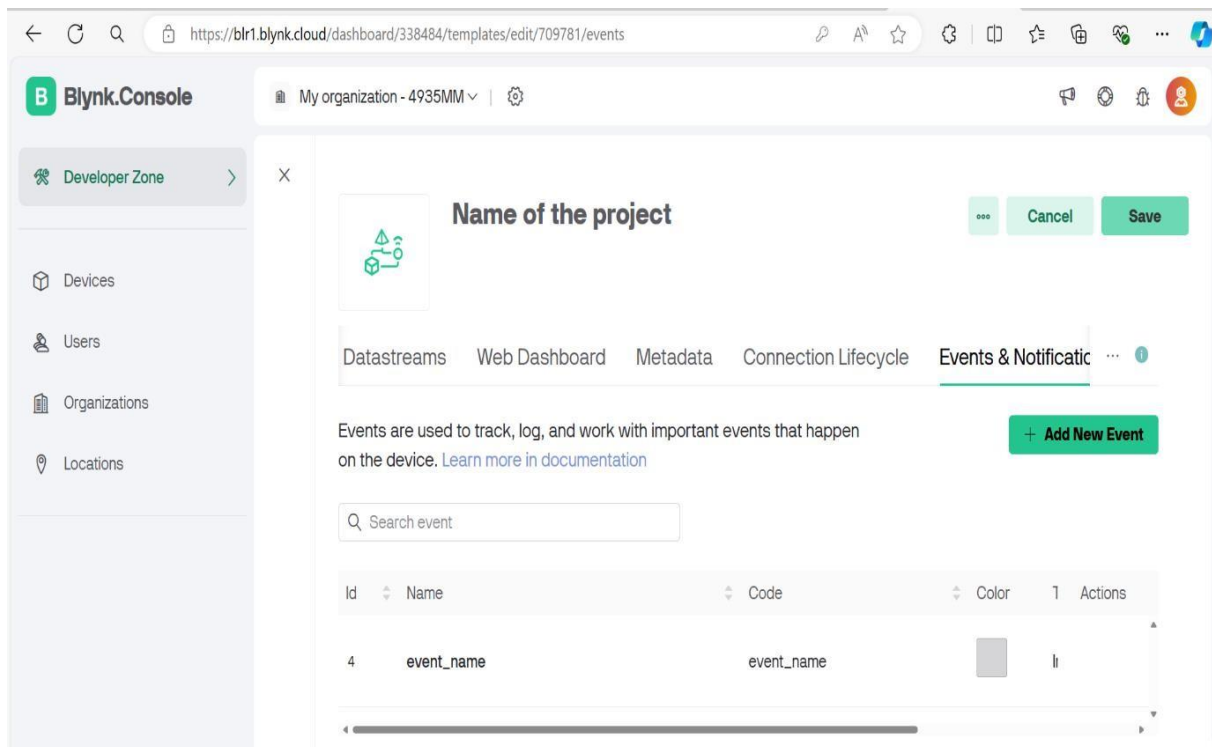


Fig.13.14 Save the event

4. IMPLEMENTATION

4.1 Overview of System Implementation

The implementation of the proposed Bus Tracking and Monitoring System involves a systematic integration of hardware and software components, ensuring seamless functionality and real-time data transmission. The system's design centers around the Arduino Uno microcontroller, which processes data collected from various sensors and modules. The ESP8266 Wi-Fi module serves as a communication bridge between the hardware and the cloud-based Blynk IoT platform, enabling real-time monitoring and visualization of critical parameters.

▪ Hardware Integration

The hardware integration process involves assembling and connecting various components to create a functional and reliable Bus Tracking and Monitoring System. At the core of the setup is the Arduino Uno microcontroller, which acts as the central hub, interfacing with multiple sensors and modules. Sensors such as the gas sensor, temperature sensor (DHT11), tire pressure sensor, vibration sensor, and accelerometer are used to monitor critical parameters like emissions, engine temperature, tire pressure, accident impacts, and vehicle stability. A GPS module provides real-time location tracking, while the ESP8266 Wi-Fi module facilitates communication with the cloud-based Blynk IoT platform.

Each sensor is connected to the appropriate analog or digital pins on the Arduino Uno, ensuring accurate data collection. The GPS and Wi-Fi modules communicate with the Arduino via UART, enabling seamless data transmission. The system is powered through a stable power supply, with voltage regulators used as necessary to protect sensitive components. All hardware components are tested individually to ensure proper functionality before integration into the system.

Once assembled, the hardware is housed in a durable enclosure to protect against vibrations, dust, and physical damage, making it suitable for deployment in the challenging conditions of a moving bus. The integrated hardware ensures reliable data collection and processing, serving as the foundation for the system's real-time monitoring and alert capabilities.

▪ Software Development

Software development for the Bus Tracking and Monitoring System focuses on creating the logic and communication protocols necessary for real-time data collection, processing, and visualization. The Arduino IDE is used to program the Arduino Uno microcontroller, which acts as the system's central

processing unit. The programming involves writing code to handle input from sensors, process the data, and detect anomalies based on predefined thresholds. For instance, the system is programmed to identify high emission levels, overheating, tire pressure deviations, or unstable movements and trigger alerts accordingly.

The software also integrates the ESP8266 Wi-Fi module for cloud communication. This involves configuring the module with the appropriate network credentials and ensuring seamless data transmission to the Blynk IoT platform. The Blynk platform serves as a cloud-based interface for real-time monitoring and management. Widgets are designed on the Blynk dashboard to visualize parameters such as emissions, engine temperature, tire pressure, vehicle stability, and GPS location. Additionally, notification settings are configured to alert users via email, SMS, or app notifications in the event of anomalies or emergencies.

To enhance functionality, the software incorporates features for logging historical data on the Blynk platform. This enables trend analysis and predictive maintenance, allowing fleet managers to optimize operations. During development, the software is thoroughly tested in simulation environments to ensure accuracy, reliability, and responsiveness. Debugging tools within the Arduino IDE help identify and resolve issues, ensuring smooth communication between the hardware and cloud systems. By integrating sensor management, data processing, anomaly detection, and cloud communication, the software enables the system to operate autonomously while providing real-time insights and alerts to fleet operators. This software-driven approach ensures the system is scalable, user-friendly, and capable of meeting the dynamic needs of modern public transportation systems.

- **Cloud Integration**

Cloud integration is a critical component of the Bus Tracking and Monitoring System, enabling real-time data transmission, visualization, and management through the **Blynk IoT platform**. This integration bridges the gap between the hardware deployed on the bus and the end-users, such as fleet managers and emergency responders, providing a centralized, accessible interface for monitoring and control.

The **ESP8266 Wi-Fi module** is configured to establish a secure connection between the hardware system and the cloud platform. It transmits processed data from sensors, such as emission levels, engine temperature, tire pressure, vehicle stability, and GPS location, to the Blynk server. The communication is achieved using standard IoT protocols like HTTP or MQTT, ensuring reliable data transmission even in real-time conditions.

On the cloud side, the Blynk IoT platform hosts a customizable dashboard that visualizes incoming data using widgets like gauges, graphs, maps, and alerts. For example:

Real-time GPS location is displayed on a live map, enabling effective route management.

Sensor readings are visualized through graphs, allowing fleet managers to monitor trends such as tire pressure fluctuations or engine temperature variations. In addition to real-time monitoring, the Blynk platform supports notification services, sending alerts via email, SMS, or push notifications to relevant stakeholders when anomalies, such as a tire blowout or high emissions, are detected. This immediate notification system ensures quick responses to critical events.

The platform also offers data storage and analytics capabilities, enabling historical data to be logged and analysed for trend identification, performance evaluation, and predictive maintenance. For instance, recurring patterns in engine temperature could signal the need for preventive repairs, reducing downtime and costs. Cloud integration also ensures scalability—multiple buses can be connected to the same Blynk dashboard, making it possible to monitor an entire fleet from a single interface. The flexibility of the platform allows customization to accommodate additional sensors or features, ensuring the system remains future-proof and adaptable to evolving requirements.

▪ **Testing and Calibration**

Once the hardware and software components are integrated, the Bus Tracking and Monitoring System undergoes a comprehensive testing and calibration process to ensure it operates accurately and reliably. The first phase involves rigorous testing in a controlled environment, where each sensor is carefully calibrated to produce precise readings. For instance, the gas sensor is exposed to controlled levels of pollutants to adjust its sensitivity and ensure accurate detection of emission levels, while the tire pressure sensor is tested against manual gauges to verify its reliability. Similarly, the accelerometer and vibration sensors are subjected to simulated conditions, such as sudden tilts or impacts, to confirm their responsiveness to vehicle stability issues or accidents. The communication between the hardware and the cloud platform is also validated during this phase. The ESP8266 Wi-Fi module is tested for consistent data transmission, ensuring that sensor data is accurately reflected on the Blynk IoT platform in real time.

To evaluate the alert mechanisms, simulated scenarios are created. For example, artificially high emission levels or a drop in tire pressure are triggered to confirm that the system detects these anomalies and sends immediate alerts to fleet managers or drivers via the cloud platform. Following successful results in the controlled environment, the system is deployed for real-world testing to evaluate its performance

under actual operating conditions. During this phase, the system is installed on a bus and monitored as it operates in dynamic environments, including varying road conditions, temperatures, and network connectivity. Real-time data collection, processing, and alerting are observed to ensure the system responds effectively to real-world challenges. This iterative process of testing, calibration, and real-world validation guarantees that the system is robust, reliable, and ready for deployment in practical scenarios.

▪ **Deployment and Operation**

Following rigorous testing and calibration, the Bus Tracking and Monitoring System is deployed in buses for real-time operation. During deployment, the hardware is securely installed within the vehicle, ensuring that sensors and modules are properly mounted and shielded from physical damage or environmental conditions like vibrations, dust, or temperature variations. The enclosure is designed to protect the system while allowing easy access for maintenance. The software is fine-tuned for continuous data collection and transmission, ensuring minimal latency in real-time monitoring.

In operation, the system functions autonomously, collecting data from integrated sensors that monitor parameters such as emissions, engine temperature, tire pressure, vehicle stability, and GPS location. This data is processed by the Arduino Uno microcontroller and transmitted via the ESP8266 Wi-Fi module to the cloud-based Blynk IoT platform. Fleet managers access the Blynk dashboard through their devices, where they can visualize real-time data using intuitive widgets such as graphs, gauges, and maps. This dashboard allows them to monitor the operational status of each bus, identify anomalies, and take preemptive actions, such as scheduling maintenance or adjusting routes. Simultaneously, drivers receive alerts directly through on-board indicators, mobile devices, or integrated display systems. For example, if the system detects underinflated tires or overheating, drivers are notified instantly, enabling immediate corrective actions. Notifications are also sent to relevant stakeholders via email or push messages for critical events like accidents or excessive emissions. The system supports decision-making processes by providing actionable insights, improving fleet efficiency, and ensuring compliance with safety and environmental regulations.

The deployment ensures that the system operates seamlessly under dynamic conditions, providing a robust solution for managing public transportation. By automating data collection, monitoring, and alerting, the system minimizes human intervention, reduces operational costs, and enhances passenger safety and environmental sustainability.

4.2 ALGORITHM and FLOW-CHART Diagram

▪ ALGORITHM

1. **Start:** The system begins its operation by initializing the hardware components. This step ensures that all sensors, microcontrollers, and communication modules are powered up and ready for data collection.
2. **Initialize Hardware:** The microcontroller (e.g., Arduino) initializes connected sensors, including GPS, tire pressure sensors, accelerometer, and emission detectors, ensuring they are functional and ready for data acquisition.
3. **Loop Begins:** The system enters a continuous loop to monitor the bus's real-time status.
4. **Read Sensors:** All connected sensors collect data about the bus's operational parameters, such as emissions, tire pressure, engine temperature, and vibrations. This data is essential for detecting anomalies or triggering alerts.
5. **GPS Check:** The system checks whether the GPS module is operational and providing valid location data.
 - **If GPS data is valid:** The process continues to the next step.
 - **If GPS data is invalid:** An error is flagged, and the system reports a "GPS Connection Error" before stopping the loop for troubleshooting.
6. **Send Data to Serial:** The collected sensor data is sent to the microcontroller's serial port for logging or debugging. This step is useful for maintenance and ensuring data accuracy during testing.
7. **Update LCD Display:** Real-time status updates are displayed on an onboard LCD screen. This provides drivers or technicians with immediate feedback on critical parameters.
8. **Send Data to Blynk:** The processed data is transmitted to the Blynk IoT platform over the internet. This allows fleet managers to monitor the bus's status remotely through a dashboard.
9. **Check Critical:** The system evaluates the data to determine whether any parameter is critical (e.g., tire pressure too low, high emission levels, or engine overheating).
 - **If Critical:** The system logs the critical event and sends an alert along with the bus's location to relevant authorities or operators for immediate action.
 - **If Not Critical:** The loop ends, and the system continues monitoring in real-time.
9. **Stop:** If an error occurs or the operation is terminated, the system halts, requiring **intervention** for troubleshooting or restart.

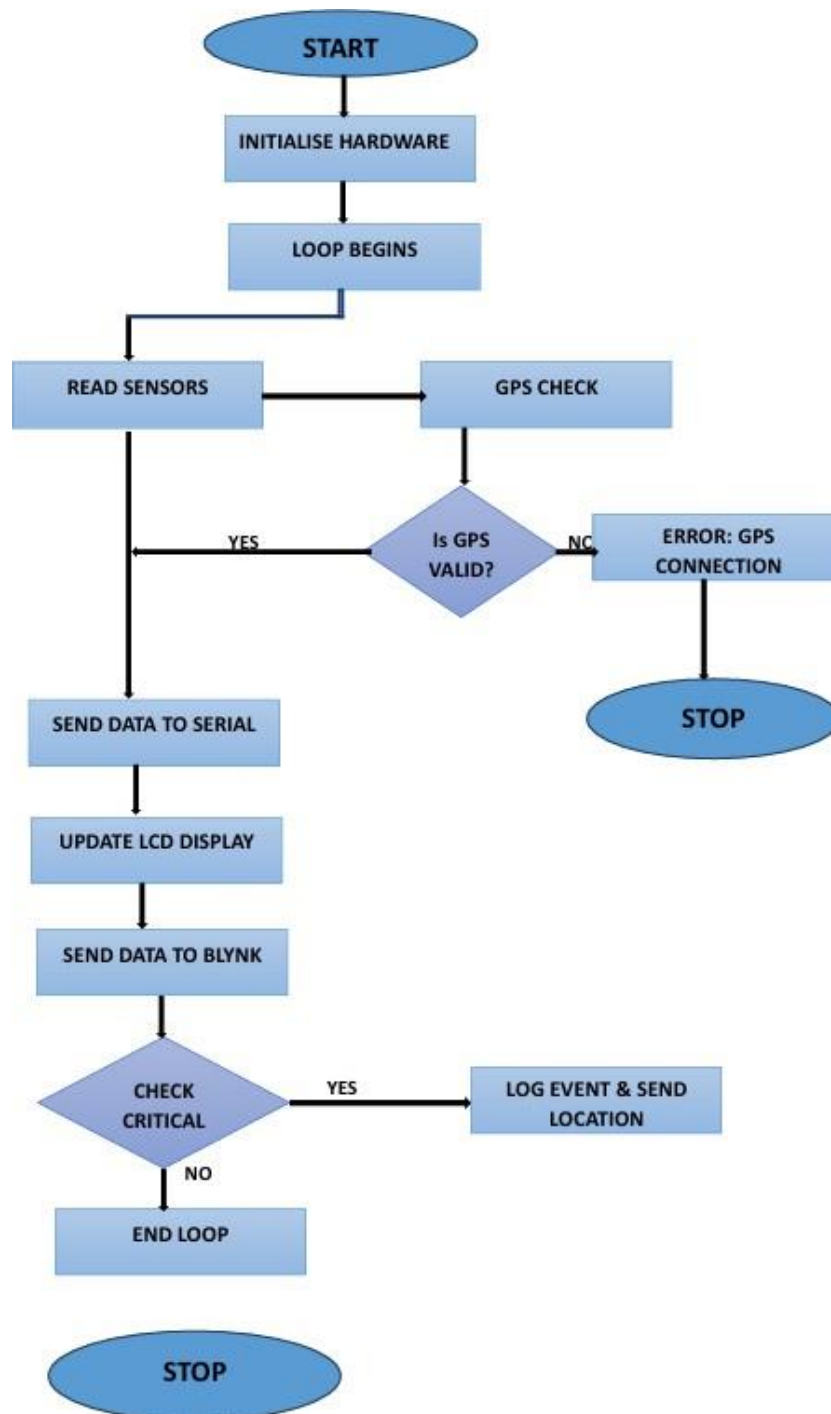
FLOW-CHART

Fig.14. FLOW CHART of Proposed Methodology

4.3 Hardware Implementation

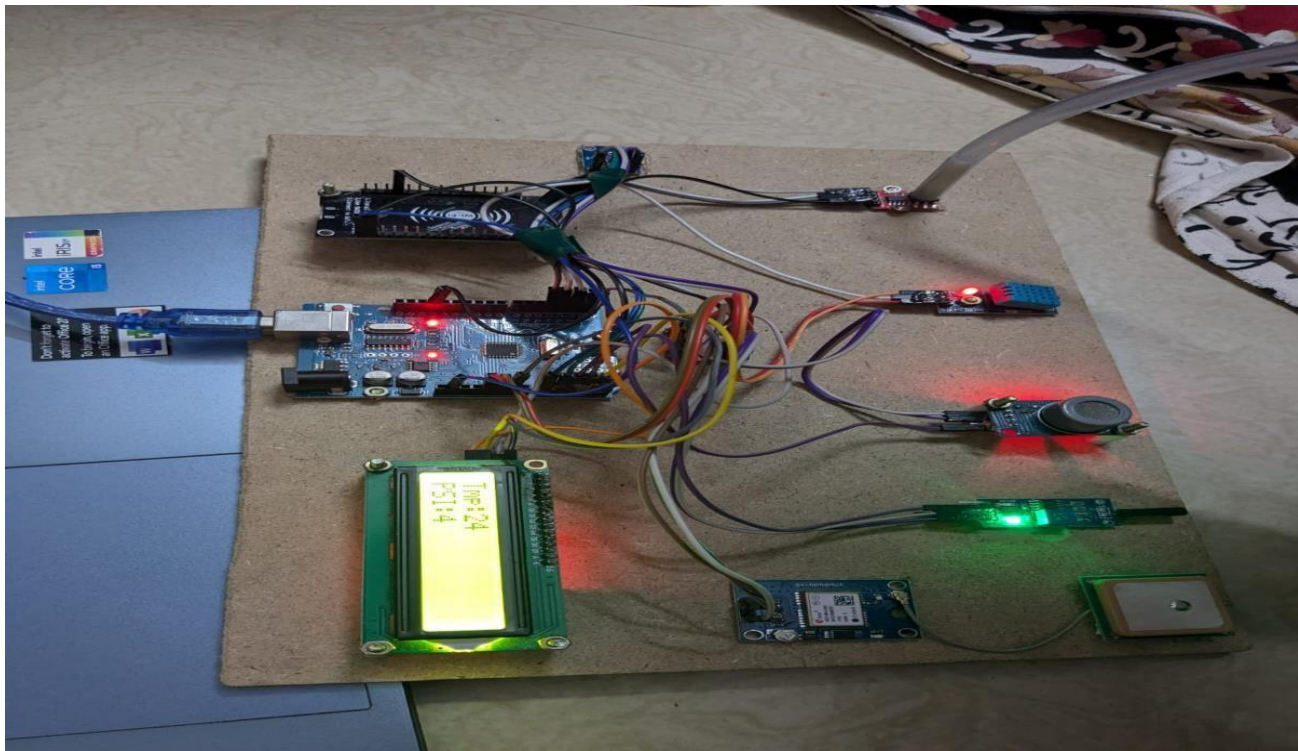
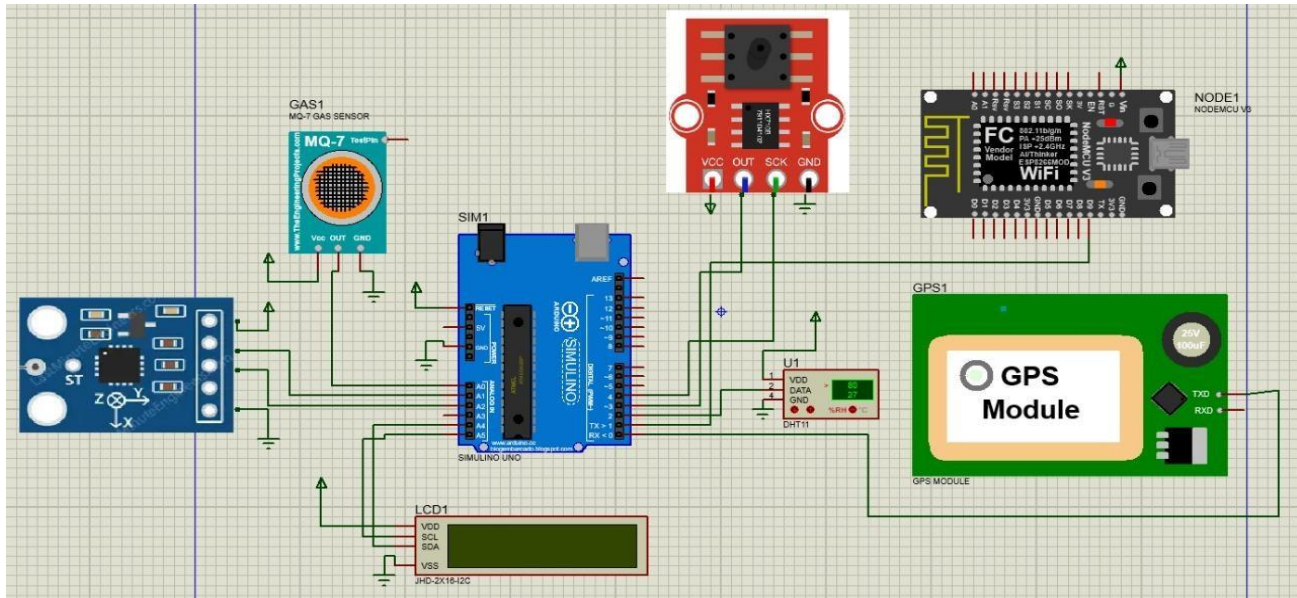


Fig 15. Hardware Implementation of Proposed Methodology

4.4 Software Implementation

▪ Code for Arduino Board

```
#include <SoftwareSerial.h>
#include <Q2HX711.h>
#include <TinyGPS++.h>
#include <LiquidCrystal_I2C.h>
#include "DHT.h"

#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
TinyGPSPlus gps;
LiquidCrystal_I2C lcd(0x27, 16, 2);

int Smoke_Sensor = A0;
int Smoke_Sensor_result;

int x = A1;
int y = A2;
int x_r;
int y_r;

int vibration = A3;
int vibration_r;

int t;

double latitude;
double longitude;
String latitude_data;
String longitude_data;
```



```
const byte MPS_OUT_pin = 3; // OUT data pin
const byte MPS_SCK_pin = 4; // clock data pin
int avg_size = 10; // #pts to average over

int psi;
Q2HX711 MPS20N0040D(MPS_OUT_pin, MPS_SCK_pin); // start comm with the HX710B

void setup()
{
  Serial.begin(9600);
  dht.begin();
  lcd.init();
  lcd.clear();
  lcd.backlight();
  pinMode(vibration,INPUT);
}

void loop()
{
  t = dht.readTemperature();

  vibration_r=digitalRead(vibration);

  Smoke_Sensor_result = analogRead(Smoke_Sensor);
  x_r = analogRead(x);
  y_r = analogRead(y);

  while (Serial.available() > 0)
    if (gps.encode(Serial.read()))
    {
```

```
if (gps.location.isValid())
{
    double latitude = (gps.location.lat());
    double longitude = (gps.location.lng());
    latitude_data= (String(latitude, 6));
    longitude_data= (String(longitude, 6));
    Serial.println(latitude_data);
    Serial.println(longitude_data);
}
}
if (millis() > 5000 && gps.charsProcessed() < 10)
{
    Serial.println(F("GPS Connection Error!!"));
    while (true);
}

float avg_val = 0.0;
for (int ii=0;ii<avg_size;ii++)
{
    avg_val += MPS20N0040D.read();
    delay(50);
}
avg_val /= avg_size;
psi= (avg_val*5.0/1024/1000)-45;
delay(100);
if(psi<0)
{
    psi=0;
}

Serial.print(t);
```

```
Serial.print(",");
Serial.print(vibration_r);
Serial.print(",");
Serial.print(x_r);
Serial.print(",");
Serial.print(y_r);
Serial.print(",");
Serial.print(Smoke_Sensor_result);
Serial.print(",");
Serial.print(psi);
Serial.print(",");
Serial.print(latitude_data);
Serial.print(",");
Serial.println(longitude_data);

lcd.setCursor(0,0);
lcd.print("Tmp:"+String(t));
lcd.setCursor(0,1);
lcd.print("PSI:"+String(psi));
delay(1000);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("LT:"+String(latitude_data));
lcd.setCursor(0,1);
lcd.print("LN:"+String(longitude_data));
delay(1000);
lcd.clear();
}
```

▪ Code for Wi-Fi Module

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#define BLYNK_PRINT Serial

#define BLYNK_TEMPLATE_ID "TMPL3rdWtMKQy"
#define BLYNK_TEMPLATE_NAME "bus tracking"
#define BLYNK_AUTH_TOKEN "ZdJlbZ4BDuPz9ACcwqX7V8HCArbh89Q0"

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Pixel 6a";
char pass[] = "raghavendra";

String data;
String data1;
String data2;
String data3;
String data4;
String data5;
String data6;
String data7;
String data8;

int t;
int vibration_r;
int x_r;
int y_r;
int Smoke_Sensor_result;
int psi;
double latitude;
double longitude;
```

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

  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
}

void loop()
{
  if (Serial.available() > 0)
  {
    data = Serial.readStringUntil('\n');
    Serial.println(data);
    if (data != "\n")
    {
      data1 = getValue(data, ',', 0);
      data2 = getValue(data, ',', 1);
      data3 = getValue(data, ',', 2);
      data4 = getValue(data, ',', 3);
      data5 = getValue(data, ',', 4);
      data6 = getValue(data, ',', 5);
      data7 = getValue(data, ',', 6);
      data8 = getValue(data, ',', 7);

      t=data1.toInt();
      vibration_r=data2.toInt();
      x_r=data3.toInt();
      Smoke_Sensor_result=data4.toInt();
      y_r=data5.toInt();
      psi=data6.toInt();
```

```
latitude=data7.toInt();
longitude=data8.toInt();

Blynk.virtualWrite(V0,t);
Blynk.virtualWrite(V1,psi);
Blynk.virtualWrite(V2,Smoke_Sensor_result);
Serial.print(t);
Serial.print(",");
Serial.print(vibration_r);
Serial.print(",");
Serial.print(x_r);
Serial.print(",");
Serial.print(Smoke_Sensor_result);
Serial.print(",");
Serial.print(x_r);
Serial.print(",");
Serial.print(psi);
Serial.print(",");
Serial.print(latitude);
Serial.print(",");
Serial.println(longitude);

if (psi!=0)//psi<30
{
  Blynk.logEvent("tyer_pressure", "LOW Tyer pressure detected");
  delay(100);
  Serial.println("tyer");
  Blynk.logEvent("google_map", "http://maps.google.com/maps?q=" + String(latitude) + "," +
    String(longitude));
  delay(100);
}
```

```
if (Smoke_Sensor_result>200)
{
    Blynk.logEvent("smoke_detection", "Smoke detected");
    delay(100);
    Blynk.logEvent("google_map", "http://maps.google.com/maps?q=" + String(latitude) + "," +
        String(longitude));
    delay(100);
}
```

```
if ((vibration_r==0)||((x_r>500)||((y_r>400))
{
    Blynk.logEvent("accident_detected", "Accident detected");
    delay(100);
    Blynk.logEvent("google_map", "http://maps.google.com/maps?q=" + String(latitude) + "," +
        String(longitude));
    delay(100);
}
```

```
}
}
```

```
Blynk.run();
data = " ";
}
```

```
String getValue(String data, char seperator, int index)
{
    int found = 0;
    int strIndex[] = {0, -1};
    int maxIndex = data.length() - 1;
```

```
for (int i = 0; i <= maxIndex && found <= index; i++)  
{  
    if (data.charAt(i) == seperator || i == maxIndex)  
    {  
        found++;  
        strIndex[0] = strIndex[1] + 1;  
        strIndex[1] = (i == maxIndex) ? i + 1 : i;  
    }  
}  
return found > index ? data.substring(strIndex[0], strIndex[1]) : "";  
  
}
```


5. RESULTS AND ANALYSIS

The Bus Tracking and Monitoring System is an IoT-based solution designed to enhance bus operations and fleet management. It improves safety by detecting vehicle stability issues, tire pressure anomalies, and accidents in real time, enabling quick responses to potential hazards. The system also minimizes environmental impact by monitoring and ensuring compliance with emission regulations.

With GPS tracking and data visualization capabilities, fleet managers can optimize routes and resources while maintaining efficient operations. Early detection of mechanical issues through continuous monitoring supports preventive maintenance, reducing downtime and repair costs. Integration with the Blynk IoT platform offers an intuitive interface for managing and visualizing data, streamlining decision-making processes, and enhancing overall efficiency. This comprehensive approach ensures safer, more sustainable, and efficient public transportation.

Additionally, the system's ability to provide real-time insights and alerts empowers operators to proactively address issues, reducing the likelihood of breakdowns and ensuring uninterrupted service. By integrating multiple monitoring features into a single platform, the system simplifies fleet management, making it more effective and reliable. The use of the Blynk IoT platform enhances accessibility, allowing operators to monitor bus performance remotely and make data-driven decisions. This holistic approach not only improves passenger safety and satisfaction but also aligns with environmental sustainability goals, making the system a vital tool for modernizing public transportation networks.

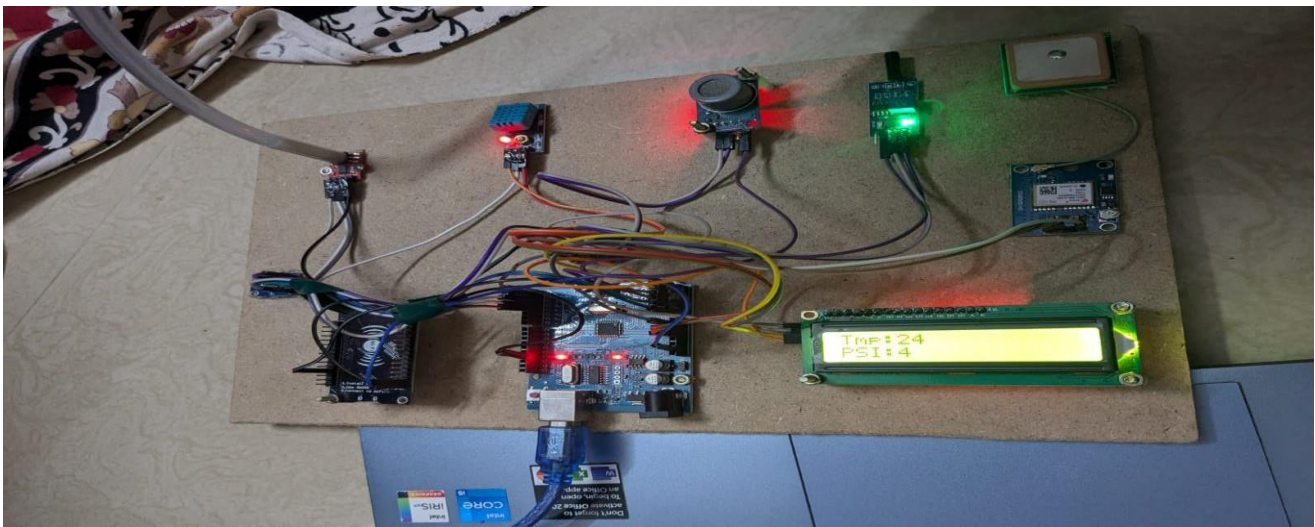


Fig. 16.a) The Temperature, Pressure and Location Values displayed on LCD Display

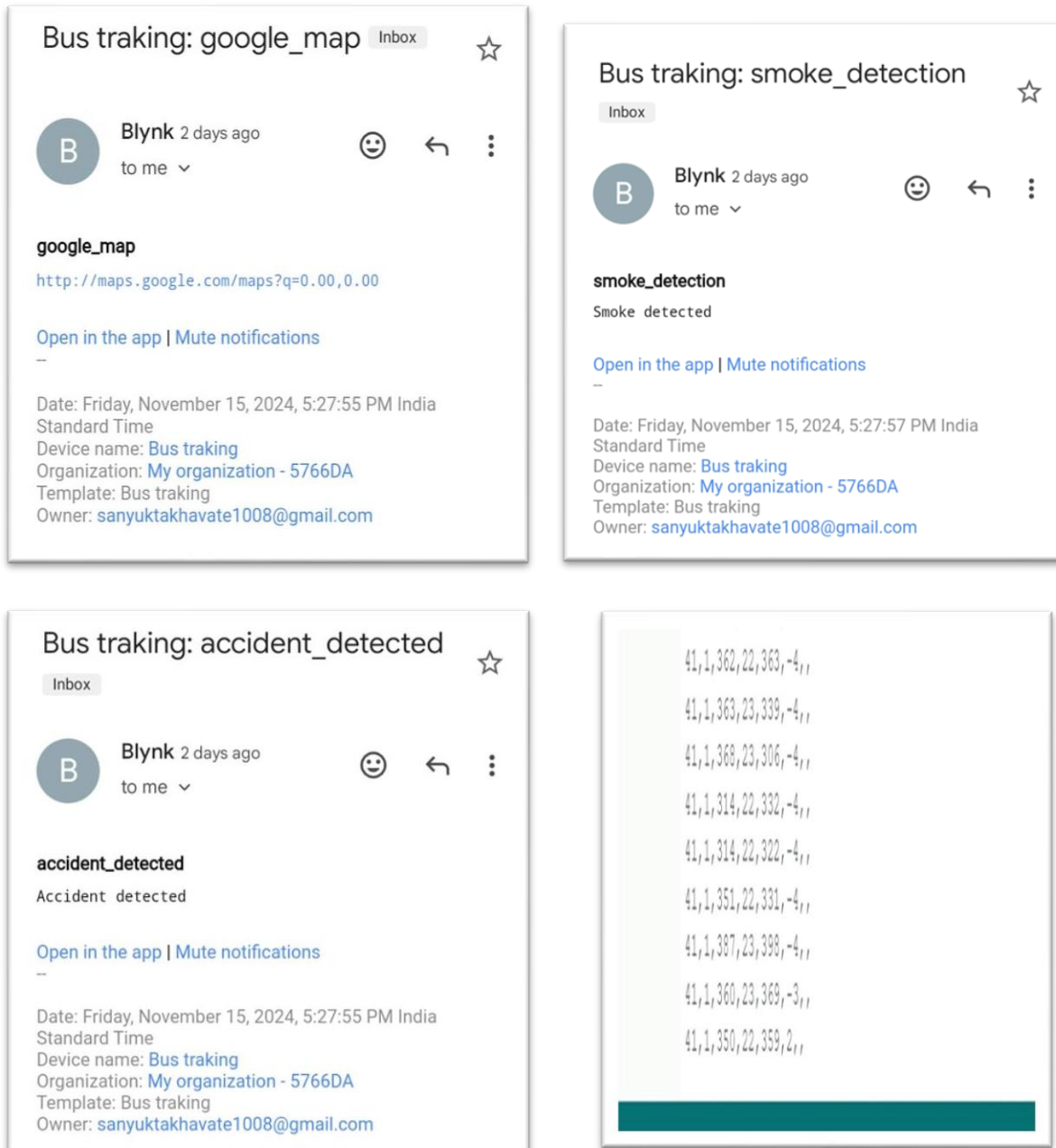


Fig. 16.b)The Notifications Sent to Mail of Owner

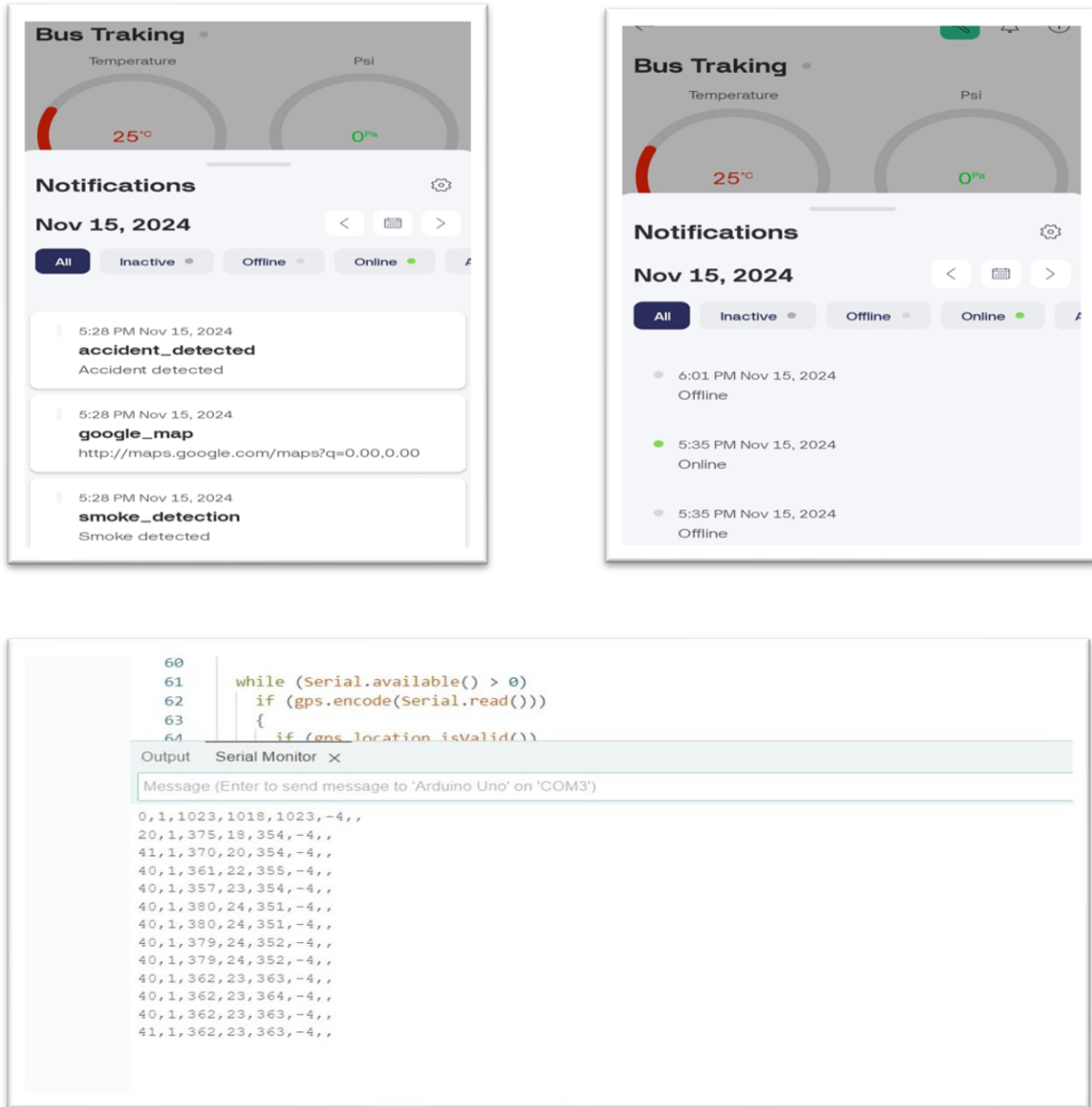


Fig.16.c) The output displayed in Serial Monitor and Notifications sent to Blynk-IOT web or app

6. APPLICATION, ADVANTAGES& LIMITATION

6.1 Application

The IoT-based Bus Tracking and Monitoring System serves multiple applications, addressing critical aspects of transportation such as efficiency, safety, and environmental sustainability. Its role in smart city integration makes it a pivotal component of modern urban planning and management.

1. Fleet Management

The system enables real-time tracking and monitoring of multiple buses in a fleet, providing critical data such as location, speed, and route adherence. Fleet managers can use this information to optimize routes, reduce idle time, and improve overall operational efficiency. It also allows for better scheduling and allocation of resources, ensuring timely arrivals and minimizing downtime.

2. Environmental Monitoring

By integrating emission sensors, the system continuously monitors pollutants like CO₂ and NO_x, ensuring buses comply with environmental regulations. This data can help authorities track and control emissions, contributing to reduced air pollution in urban areas. The insights gathered can also guide policymakers in formulating strategies to minimize the environmental impact of public transportation.

3. Passenger Safety

The system enhances safety by monitoring critical parameters such as vehicle stability, tire pressure, and engine temperature. Any deviation from safe thresholds triggers alerts to drivers and fleet managers, helping prevent accidents. Features like real-time GPS tracking also reassure passengers, as their journeys can be monitored for added security.

4. Accident Response

In the event of a collision or accident, the system uses accelerometers and vibration sensors to detect the impact and automatically send alerts with the vehicle's location to emergency contacts or authorities. This feature significantly reduces response times, improving the chances of saving lives and mitigating further damage.

5. Public Transportation Planning

Data collected by the system, such as passenger load, route efficiency, and peak travel times, can help urban planners optimize public transportation networks. By analyzing trends, authorities can identify high-demand routes, improve scheduling, and allocate buses more effectively, enhancing the overall public transportation experience.

6. Preventive Maintenance

The system tracks vehicle performance data, such as engine temperature and tire pressure, to predict potential maintenance issues. Alerts for minor issues allow timely repairs before they escalate into significant problems, reducing breakdowns and extending the lifespan of vehicles. This predictive approach helps lower maintenance costs and ensures smoother operations.

7. Smart City Integration

The system aligns with the broader goals of smart cities by contributing to sustainable and efficient transportation infrastructure. Data from the system can be integrated into centralized smart city platforms, allowing for better traffic management, reduced congestion, and improved air quality monitoring. This fosters a more connected and intelligent urban environment, benefiting both commuters and city administrators.

6.2 Advantages

1. Enhanced Fleet Efficiency

By providing real-time location tracking and performance data, the system allows fleet managers to optimize routes, reduce idle times, and improve schedule adherence. This leads to more efficient operations, lower fuel consumption, and reduced operational costs.

2. Improved Passenger Safety

Monitoring critical parameters such as vehicle stability, tire pressure, and engine temperature ensures early detection of potential issues, reducing the likelihood of accidents. Emergency alerts and location tracking further enhance safety by enabling faster response times during emergencies.

3. Environmental Sustainability

Continuous monitoring of emissions helps ensure compliance with environmental regulations, reducing the carbon footprint of public transportation. The system promotes sustainable practices by encouraging the use of cleaner, more efficient transportation methods.

4. Reduced Maintenance Costs

Predictive maintenance through real-time data collection helps identify and address minor issues before they become major problems. This reduces unexpected breakdowns, minimizes downtime, and extends the lifespan of vehicle components, leading to cost savings.

5. Better Decision-Making

The integration of real-time data on vehicle performance and passenger load provides actionable insights for fleet managers. This data-driven approach supports better decision-making, such as optimizing bus allocation, route planning, and scheduling based on demand patterns.

6. Quick Emergency Response

The system's ability to detect accidents and send immediate alerts to authorities with GPS location reduces response times. This can potentially save lives and minimize damage, providing a critical safety feature for public transportation.

7. Enhanced User Experience

Passengers benefit from real-time updates on bus locations, estimated arrival times, and service status, which enhances their travel experience. Transparent communication fosters trust in public transportation systems.

8. Support for Smart City Initiatives

The system aligns with the goals of smart cities by contributing to efficient traffic management, reduced congestion, and better resource utilization. The data collected can be integrated into centralized platforms, aiding in comprehensive urban planning.

9. Scalability and Flexibility

The system is scalable, allowing for the addition of more vehicles and features as needed. Its modular design enables integration with other IoT applications, making it adaptable to evolving technological and transportation needs.

10. Increased Operational Transparency

By providing a centralized dashboard for monitoring all buses in real-time, the system improves operational transparency. Fleet managers and stakeholders can access accurate and timely information, ensuring accountability and trust in the system.

6.3 Limitation

1. High Initial Cost

The implementation of an IoT-based system requires a significant initial investment in hardware, sensors, software, and cloud infrastructure. For some transportation operators, the cost might be a barrier to adoption.

2. Dependence on Internet Connectivity

The system heavily relies on stable internet connectivity for real-time data transmission between the buses and the cloud. In areas with poor network coverage, system performance may degrade, leading to delayed updates and reduced efficiency.

3. Maintenance and Upkeep

Regular maintenance of sensors, devices, and the overall system is essential to ensure its accuracy and reliability. This includes recalibration of sensors and addressing wear and tear, which can increase operational costs over time.

4. Data Privacy and Security Concerns

Handling large volumes of real-time data, including GPS locations and passenger information, raises concerns about data privacy and security. Cyberattacks or unauthorized access to the system could compromise sensitive information or disrupt operations.

5. Complexity in Integration

Integrating the IoT-based system with existing fleet management platforms or smart city frameworks can be complex and time-consuming. Compatibility issues with legacy systems may arise, requiring additional resources for customization.

6. Sensor Limitations

The accuracy and performance of the system are highly dependent on the quality and range of the sensors used. Inexpensive or low-quality sensors may fail to deliver accurate readings, leading to false alerts or missed detections.

7. Environmental Impact on Sensors

External environmental factors such as extreme weather conditions, vibrations, or dust can affect the performance and longevity of sensors, leading to potential inaccuracies or system downtime.

8. Limited Scalability in Remote Areas

While the system is scalable in urban settings, its deployment in rural or remote areas may face challenges due to lack of infrastructure, such as reliable power supply and communication networks.

9. Training Requirements

Operating and maintaining the system requires technical expertise. Fleet managers, drivers, and technicians need to be trained to use the system effectively, which could increase the overall deployment time and cost.

10. Risk of Over-Reliance

Over-reliance on the system for decision-making without proper human oversight could lead to problems if the system encounters errors, malfunctions, or false data. A balanced approach combining automation with human intervention is necessary.

7. CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The Bus Tracking and Monitoring System is a cutting-edge solution leveraging IoT technology and advanced sensors to revolutionize the management of public transportation fleets. By incorporating real-time monitoring of critical parameters such as emissions, engine temperature, vehicle stability, tire pressure, and accident detection, the system ensures comprehensive oversight of bus operations. This proactive management enables timely interventions, preventing potential hazards and reducing the risk of costly breakdowns.

The integration of GPS tracking with data visualization on the Blynk IoT platform significantly enhances operational efficiency. Fleet managers can optimize route planning, monitor vehicle locations in real-time, and address issues promptly through a user-friendly dashboard. The system also supports preventive maintenance, reducing downtime and extending the lifespan of buses, thereby minimizing operational costs. In addition to improving safety and efficiency, the system contributes to environmental sustainability by monitoring emissions and ensuring compliance with regulatory standards. This feature aligns with modern urban goals of reducing pollution and promoting eco-friendly public transportation.

The system also emphasizes environmental sustainability, a pressing concern in densely populated cities. By monitoring and controlling emissions in real-time, it not only ensures compliance with environmental regulations but also actively contributes to reducing air pollution. This feature aligns with global efforts to mitigate climate change and improve urban air quality, making public transportation a more sustainable alternative to private vehicles. Moreover, the accident detection and notification system is a critical component that enhances passenger safety. Using vibration sensors, the system detects collisions or significant impacts and immediately alerts the relevant authorities or fleet operators. This rapid response capability ensures that assistance can be dispatched promptly, minimizing harm and enabling quick recovery from incidents.

In conclusion, the Bus Tracking and Monitoring System addresses multiple facets of modern fleet management, from safety and efficiency to sustainability and cost-effectiveness. Its integrated approach not only improves operational reliability but also enhances the overall quality of public transportation services, making it an indispensable tool for smart city initiatives. By reducing downtime, improving resource allocation, and ensuring environmental compliance, the system sets a benchmark for the future of urban mobility.

7.2 Future Scope

The Bus Tracking and Monitoring System has immense potential for future enhancements and broader applications. As IoT technology continues to evolve, the system can be upgraded to include more advanced features and integrations, making it an even more robust solution for smart transportation.

The Bus Tracking and Monitoring System offers significant potential for future advancements, making it an essential component of modern transportation systems. One promising area is the integration of advanced analytics and artificial intelligence (AI). By analyzing historical and real-time data, the system can enable predictive maintenance, identifying potential mechanical failures before they occur and allowing operators to schedule repairs efficiently. Additionally, AI-powered analytics can assess driving behavior, promoting safer and more fuel-efficient practices. The integration of machine learning can also optimize route planning, analyzing traffic patterns, weather conditions, and passenger demand to suggest more efficient routes.

Expanding sensor capabilities is another critical aspect of the system's future. Advanced emission control can be achieved by incorporating sensors that detect a broader range of pollutants, ensuring stricter environmental compliance. Passenger-focused additions, such as smart cameras or infrared sensors, can automate passenger counting, helping to manage bus capacities and adjust services dynamically. Weather monitoring sensors can further enhance safety by providing alerts about adverse conditions. Passenger interaction can be significantly improved by integrating the system with mobile applications. These applications could provide real-time information on bus locations, estimated arrival times, and seat availability, enhancing the overall commuter experience. Additionally, features like panic buttons for passengers can improve safety by enabling immediate communication with authorities in emergencies.

The system has the potential to play a pivotal role in smart city initiatives. It can integrate with broader transportation networks, such as traffic light systems, to prioritize buses and reduce delays. Moreover,

data collected from the system can inform city planners, helping optimize bus routes, manage peak traffic, and improve urban infrastructure. Renewable energy sources, such as solar power, can also be incorporated into the system, powering IoT devices sustainably and supporting electric or hybrid buses by monitoring battery health and energy consumption.

In the long term, the system can evolve to support autonomous vehicle technologies. As self-driving buses become a reality, the system can enable real-time communication with smart infrastructure and other vehicles to ensure safe and efficient operations. The integration of Geographic Information System (GIS) technologies can provide detailed geographical insights, such as identifying accident-prone zones or frequently problematic routes, further improving safety and efficiency.

By adopting these advancements, the Bus Tracking and Monitoring System will continue to address the growing challenges of urban transportation. Its evolution will not only enhance operational efficiency and passenger safety but also contribute to sustainability and the successful implementation of smart city initiatives.

8. References

- [1]. J. Park, S. J. Jung, and H. K. Kim, "Bus Tracking and Monitoring System Based on IoT," *IEEE Access*, vol. 8, pp. 14683-14691, 2020. doi: 10.1109/ACCESS.2020.2965904.
- [2]. A. Shinde, M. M. Naik, P. Gadkari, and S. Deshmukh, "IoT-Based Smart Vehicle Monitoring System Using Sensors," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4153-4161, 2020. doi: 10.1109/JIOT.2020.2971817.
- [3]. A. Roy, S. Kar, and P. K. Mandal, "IoT-Based Intelligent Vehicle Tracking and Monitoring System Using Raspberry Pi," in *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 2021, pp. 1-6. doi: 10.1109/UPCON52273.2021.9667614.
- [4]. Y. Wang, L. Zhao, and Y. Zhang, "A Comprehensive IoT-based Bus Fleet Monitoring and Management System," in *2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP)*, 2021, pp. 45-49. doi: 10.1109/ICSIP52628.2021.9688990.
- [5]. M. El-Bendary, M. F. A. El-Soudani, and A. M. El-Shafee, "IoT-Based Vehicle Emission Monitoring and Control System," in *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, 2020, pp. 109-113. doi: 10.1109/ITCE48509.2020.9047749.
- [6]. T. B. Sawant, B. S. More, and V. K. Joshi, "IoT Enabled Real-Time Vehicle Emission Monitoring System," *IEEE Sensors Letters*, vol. 5, no. 1, pp. 1-4, 2021. doi: 10.1109/LSENS.2020.3047352.
- [7]. P. Singh, M. Sood, and A. Kumar, "IoT-Enabled Vehicle Accident Detection and Monitoring System," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2315-2322, 2021. doi: 10.1109/JIOT.2020.3026944.
- [8]. S. Yang, L. Zhang, Y. Hu, and X. Zeng, "Real-Time Monitoring System for Vehicle Tire Pressure Based on IoT," *IEEE Access*, vol. 8, pp. 183987-183997, 2020. doi: 10.1109/ACCESS.2020.3029521.
- [9]. N. Y. I. Putri, S. Kurniawan, and F. Riandini, "Design of an IoT-Based Vehicle Tracking System Using GPS and GSM Modules," in *2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)*, 2021, pp. 77-82. doi: 10.1109/IoTaIS53135.2021.9618913.

- [10]. A. K. Verma, V. K. Tomar, and M. Singh, "IoT-Based Vehicle Safety System for Accident Prevention and Notification," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5648-5656, 2021. doi: 10.1109/TII.2021.3066635.
- [11]. M. I. Mazumder, S. Parvez, and M. A. Rahman, "IoT-Based Smart Vehicle Emission Monitoring System," in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2021, pp. 211-215. doi: 10.1109/IEMCON53756.2021.9623212.
- [12]. A. Sharma, N. K. Verma, and P. P. Roy, "IoT-Enabled Smart Vehicle Monitoring System for Fleet Management," *IEEE Sensors Journal*, vol. 21, no. 6, pp. 7538-7546, 2021. doi: 10.1109/JSEN.2021.3050226.



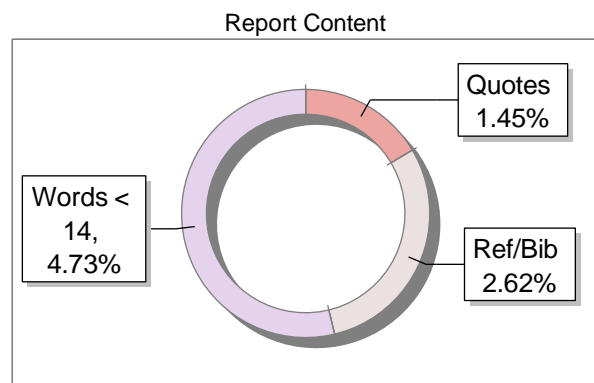
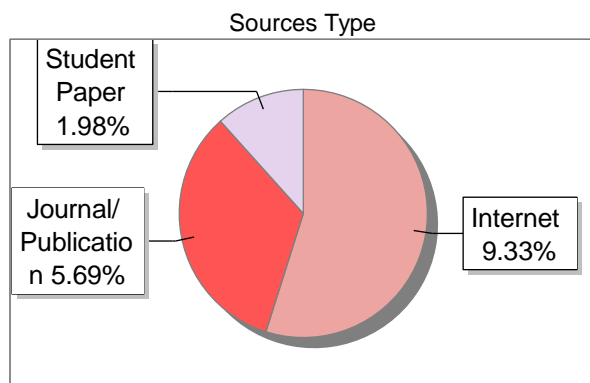
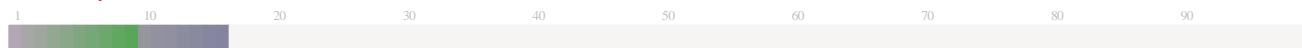
The Report is Generated by DrillBit Plagiarism Detection Software

Submission Information

Author Name	HVM
Title	IOT5
Paper/Submission ID	2704729
Submitted by	hod-ece@dayanandasagar.edu
Submission Date	2024-12-05 11:25:38
Total Pages, Total Words	94, 15948
Document type	Project Work

Result Information

Similarity **17 %**



Exclude Information

Quotes	Not Excluded
References/Bibliography	Not Excluded
Source: Excluded < 14 Words	Not Excluded
Excluded Source	0 %
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File





DrillBit Similarity Report

17

SIMILARITY %

125

MATCHED SOURCES

B

GRADE

A-Satisfactory (0-10%)

B-Upgrade (11-40%)

C-Poor (41-60%)

D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	www.tpiconference.in	1	Publication
2	goldenseahoian.hoabinhland.vn	1	Internet Data
3	Submitted to Visvesvaraya Technological University, Belagavi	1	Student Paper
4	www.slideshare.net	1	Internet Data
5	REPOSITORY - Submitted to Exam section VTU on 2024-07-31 16-08 962766	1	Student Paper
6	dochero.tips	<1	Internet Data
7	drttit.gvet.edu.in	<1	Publication
8	www.gofleet.com	<1	Internet Data
9	assets.sjbit.edu.in	<1	Publication
10	www.matellio.com	<1	Internet Data
11	www.scribd.com	<1	Internet Data
12	jamiahamdard.edu	<1	Publication
13	Submitted to Visvesvaraya Technological University, Belagavi	<1	Student Paper
14	ghparafusosferramentas.com.br	<1	Internet Data

15	REPOSITORY - Submitted to Exam section VTU on 2024-07-31 17-17 898230	<1	Student Paper
16	www.leewayhertz.com	<1	Internet Data
17	www.srielectronics.com	<1	Internet Data
18	duc.edu.iq	<1	Publication
19	www.smec.ac.in	<1	Publication
20	IEEE 2018 18th International Symposium on Communications and Inform, by Duangsuwan, Sarun - 2018	<1	Publication
21	fastercapital.com	<1	Internet Data
22	Thesis Submitted to Shodhganga Repository	<1	Publication
23	en.wikipedia.org	<1	Internet Data
24	kunkune.co.uk	<1	Internet Data
25	Submitted to Visvesvaraya Technological University, Belagavi	<1	Student Paper
26	www.nituk.ac.in	<1	Publication
27	www.pce.ac.in	<1	Publication
28	randomnerdtutorials.com	<1	Internet Data
29	www.scribd.com	<1	Internet Data
30	repo.darmajaya.ac.id	<1	Publication
31	springeropen.com	<1	Internet Data
32	www.tcetmumbai.in	<1	Publication
33	blog.airlinehyd.com	<1	Internet Data

34	ijream.org	<1	Publication
35	erj.ersjournals.com	<1	Internet Data
36	jai.front-sci.com	<1	Publication
37	www3.epa.gov	<1	Internet Data
38	docplayer.net	<1	Internet Data
39	dsbs.edu.in	<1	Publication
40	www.sit.ac.in	<1	Publication
41	springeropen.com	<1	Internet Data
42	repositorioslatinoamericanos	<1	Publication
43	www.hrpub.org	<1	Publication
44	CirclePIN A Novel Authentication Mechanism for Smartwatches to Prevent Unauthor by Guerar-2020	<1	Publication
45	Thesis Submitted to Shodhganga Repository	<1	Publication
46	www.doaj.org	<1	Publication
47	www.linkedin.com	<1	Internet Data
48	docplayer.net	<1	Internet Data
49	johnleonard.com	<1	Internet Data
50	www.dx.doi.org	<1	Publication
51	IEEE 216 IEEE Symposium on Technologies for Homeland Security (HST) by	<1	Publication
52	documents1.worldbank.org	<1	Publication

53	inba.info	<1	Internet Data
54	www.hseblog.com	<1	Internet Data
55	www.mdpi.com	<1	Internet Data
56	auamh.realestateplanet.info	<1	Internet Data
57	Data Tools Improve Nutrient Monitoring, by Danielsen, Karlin - 2018	<1	Publication
58	fastercapital.com	<1	Internet Data
59	repository.up.ac.za	<1	Publication
60	The Formation of Employee Satisfaction with Airline Information Systems by Au-2012	<1	Publication
61	www.linkedin.com	<1	Internet Data
62	abnews-wire.blogspot.com	<1	Internet Data
63	utilitiesone.com	<1	Internet Data
64	www.frontiersin.org	<1	Internet Data
65	www.marketsandmarkets.com	<1	Internet Data
66	bimcafe.in	<1	Internet Data
67	blog.hubspot.com	<1	Internet Data
68	docplayer.net	<1	Internet Data
69	ijcea.com	<1	Publication
70	justdoelectronics.com	<1	Internet Data
71	Measuring the Resiliency of the Manhattan Points of Entry in the Face by Omer-2011	<1	Publication

72	pdfcookie.com	<1	Internet Data
73	religiondocbox.com	<1	Internet Data
74	sites.ndtv.com	<1	Internet Data
75	www.azosensors.com	<1	Internet Data
76	www.facttwin.com	<1	Internet Data
77	www.financialexpress.com	<1	Internet Data
78	www.freepatentsonline.com	<1	Internet Data
79	www.freepatentsonline.com	<1	Internet Data
80	www.inderscience.com	<1	Internet Data
81	www.readbag.com	<1	Internet Data
82	IEEE 2011 IEEE International Conference on Robotics and Biomimetics by	<1	Publication
83	American Institute of Aeronautics and Astronautics 49th AIAAASMEAS	<1	Publication
84	arduino-research-papers.blogspot.com	<1	Internet Data
85	asbmr.onlinelibrary.wiley.com	<1	Internet Data
86	azslide.com	<1	Internet Data
87	A CANNY GENARAL REMOTE CONTROL Frame Work Home Appratuses -Submitted to JNTUH,Telangana By 156C1D6813	<1	Student Paper
88	bibliomed.org	<1	Internet Data
89	bmcreview.org	<1	Internet Data

90	Comparison of ANN Controller and PID Controller for Industrial Wa- www.ijcaonline.org	<1	Publication
91	digitalcommons.cwu.edu	<1	Internet Data
92	dochero.tips	<1	Internet Data
93	docplayer.net	<1	Internet Data
94	docplayer.net	<1	Internet Data
95	docplayer.net	<1	Internet Data
96	dovepress.com	<1	Internet Data
97	ejbiophysics.org	<1	Internet Data
98	en.wikipedia.org	<1	Internet Data
99	fastercapital.com	<1	Internet Data
100	Fracture control and structural integrity (FraCSI) education and research by Saxena-2018	<1	Publication
101	Interactive Screen Video StreamingBased Pervasive Mobilby Zhan Ma 2017- ieeeexplore.org	<1	Publication
102	link.springer.com	<1	Internet Data
103	moam.info	<1	Internet Data
104	pdfcookie.com	<1	Internet Data
105	pmc.ncbi.nlm.nih.gov	<1	Internet Data
106	qdoc.tips	<1	Internet Data
107	scidoc.org	<1	Internet Data

108	springeropen.com	<1	Internet Data
109	Submitted to Visvesvaraya Technological University, Belagavi	<1	Student Paper
110	Thesis submitted to shodhganga - shodhganga.inflibnet.ac.in	<1	Publication
111	Thesis Submitted to Shodhganga Repository	<1	Publication
112	The Use of an e-Learning System for Agricultural Extension A Case Study of the by Park-2007	<1	Publication
113	tunnellingjournal.com	<1	Internet Data
114	www.gofleet.com	<1	Internet Data
115	www.iiste.org	<1	Internet Data
116	www.ijert.org	<1	Internet Data
117	www.ijstr.org	<1	Internet Data
118	www.linkedin.com	<1	Internet Data
119	www.linkedin.com	<1	Internet Data
120	www.linkedin.com	<1	Internet Data
121	www.scribd.com	<1	Internet Data
122	www.wagnerreese.com	<1	Internet Data
123	xn--flge-1ra.app	<1	Internet Data
124	xyonline.net	<1	Publication
125	American Institute of Aeronautics and Astronautics 24th Joint Propul	<1	Publication