A Project report on

Development of Real Time Bus Tracking and Accident Detection System

Submitted in partial fulfillment of the

requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

by

HEMALATHA K	**(204G1A0441)**
DEEPTHI REDDY S	**(214G5A0403)**
KAVYA T	**(204G1A0454)**
DIVIJA C	**(204G1A0425)**

Under the Guidance of

Mr. Y. RAJA KULLAYI REDDY, M.Tech.,
Assistant Professor

ELECTRONICS AND COMMUNICATION ENGINEERING

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY
(Autonomous)
(Affiliated to JNTUA, Approved by AICTE, New Delhi, Accredited by NAAC with 'A' grade &Accredited by NBA (B. TECH ECE, EEE & CSE))
Rotarypuram Village, B K Samudram Mandal, Ananthapuramu - 515701

2023-2024

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY

(Autonomous)

(Affiliated to JNTUA, Approved by AICTE, New Delhi, Accredited by NAAC with 'A' grade &Accredited byNBA (B.TECHECE, EEE&CSE))

Rotarypuram Village, B K Samudram Mandal, Ananthapuramu - 515701

Certificate

This is to certify that the project report entitled Development of Real Time Bus Tracking and Accident Detection System is the bonafide work carried out by Hemalatha K bearing Roll Number 204G1A0441, Deepthi Reddy S bearing Roll Number 214G5A0403, Kavya T bearing Roll Number 204G1A0454 and Divija C bearing Roll Number 204G1A0425 in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering during the academic year 2023-2024.

Project Guide	**Head of theDepartment**
Mr.Y. Raja Kullayi Reddy,M.Tech.,	Dr.M.L.RaviChandra,M.Tech., Ph.D.,
Assistant Professor,	Professor&HOD,
Dept.of ECE	Dept.of ECE

Date:

Place: Anantapur **EXTERNAL EXAMINER**

DECLARATION

We hereby declare that the project report entitled "Development of Real Time Bus Tracking and Accident Detection System" submitted for the award of the degree of Bachelor of Technology in our original work and the project report has not formed the basis for the award of any degree, diploma, associateship, or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma**.**

Place: Rotarypuram

Date:

PROJECT ASSOCIATES

HEMALATHA K	204G1A0441
DEEPTHI REDDY S	214G5A0403
KAVYA T	204G1A0454
DIVIJA C	204G1A0425

INSTITUTION VISION AND MISSION

VISION:

To become a premier Educational Institution in India offering the best teaching and learning environment for our students that will enable them to become complete individuals with professional competency, human touch, ethical values, service motto, and a strong sense of responsibility towards the environment and society at large.

MISSION:

M1: Continually enhance the quality of physical infrastructure and human resources to evolve into a center of excellence in engineering education**.**

M2: Provide comprehensive learning experiences that are conducive for the students to acquire professional competencies, ethical values, life-long learning abilities, and understanding of the technology, environment, and society.

M3: Strengthen industry-institute interactions to enable the students to work on realistic problems and acquire the ability to face the ever-changing requirements of the industry.

M4: Continually enhance the quality of the relationship between students and faculty which is a key to the development of an exciting and rewarding learning environment in the college.

DEPARTMENT VISION AND MISSION

VISION:

To become a department of excellence in Electronics and Communication and allied areas of engineering by empowering rural students with the latest technological updates and human values

MISSION:

DM1: Continually improve the teaching learning and associated processes to prepare the students with problem solving skills.

DM2: Provide comprehensive learning experiences to imbibe industry based technical knowledge and encourage students to pursue higher studies with awareness on ethical values.

DM3: Nurture a strong research eco-system that facilitates quality research by faculty and students.

ACKNOWLEDGMENT

The satisfaction and euphoria accompanying the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we have now the opportunity to express our gratitude to all of them.

It is with immense pleasure that we would like to express our indebted gratitude to our Guide **Mr. Y. Raja Kullayi Reddy,M. Tech.,Assistant Professor, Electronics and Communication Engineering Department**, who has guided us a lot and encouraged us in every step of the project work. We thank him for the stimulating guidance, constant encouragement, and constructive criticism which have made it possible to bring out this project work.

We are very thankful to **Dr. M. L. Ravi Chandra, Professor & HOD, Department of Electronics & Communication Engineering,** for his kind support and for providing the necessary facilities to carry out the work.

We wish to convey our special thanks to **Dr. G. Balakrishna, Principal** of **Srinivasa Ramanujan Institute of Technology** for giving the required information in doing our project work. Not to forget, we thank all other Teaching, and non-teaching staff and our friends who had directly or indirectly helped and supported us in completing our project in time.

We also express our sincere thanks to the Management for providing excellent facilities**.**

Finally, we wish to convey our gratitude to our family who fostered all the requirements and facilities that we need.

Project Associates

ABSTRACT

In today's bustling urban landscapes, buses serve as essential arteries of transportation, linking diverse sectors including educational institutions, businesses, and public organizations. However, the escalating challenge of traffic congestion spurred by burgeoning urban populations demands innovative solutions for enhancing both efficiency and safety in bus operations. A groundbreaking response emerges in the form of a real-time bus tracking system, integrating cutting-edge GSM and GPS technologies to navigate the complexities of modern urban mobility. By equipping buses with GSM trackers for real-time location updates and GPS receivers for precise positioning data, passengers gain the ability to monitor bus movements seamlessly, while transit authorities leverage invaluable insights for fleet management and operational optimization. Complementing these capabilities are strategically deployed tilt sensors, providing nuanced insights into bus orientation and movement patterns, alongside fire sensors for swift detection of onboard hazards. Further enhancing system robustness are ESP32MCU for WiFi connectivity and regulators for efficient power management, solidifying the system's reliability and efficacy. Through this integrated approach, the real-time bus tracking system not only addresses the immediate challenges of urban congestion but also heralds a transformative shift towards smarter, safer, and more efficient urban transportation networks.

Keywords: Sensors, Node MCU,GPS, Pushbullet Application, Folium Map,Visual Studio Code.

LIST OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	DECLARATION	**I**
	VISION AND MISSION	**II**
	ACKNOWLEDGEMENT	**III**
	ABSTRACT	**IV**
	LIST OF FIGURES	**V**
	LIST OF ABBREVIATIONS	**VI**
1	**INTRODUCTION**	**1-18**
	1.1 Embedded Systems	1-2
	1.1.1 Characteristics of Embedded Systems	2-4
	1.1.2 Components of Embedded Systems	4-6
	1.1.3 Embedded System Architecture	7-8
	1.1.4 Advantages of ES	9-10
	1.1.5 Applications of ES	10-12
	1.2 Internet of Things	12-14
	1.2.1 Components and Architecture of IoT	14-16
	1.2.2 Applications and Impact of IoT	17-18
2	**LITERATURE SURVEY**	**19-21**
3	**EXISTING SYSTEM**	**22-23**

	3.1 Existing system	22
	3.2 Drawbacks of existing system	22-23
4	**Development of Real Time Bus Tracking and Accident Detection System**	**24-40**
	4.1 Proposed system	24
	4.1.1 Block diagram	25-26
	4.1.2 Working Principle	26-27
	4.2 Hardware Requirements	27-33
	4.2.1 Vibration Sensor	27-28
	4.2.2 Fire Sensor	28-29
	4.2.3 GPS	29-30
	4.2.4 NodeMCU(ESP32)	31-33
	4.3 Software requirements	33-40
	4.3.1 Arduino IDE	33-36
	4.3.2 Adding Libraries	36-37
	4.3.3 Pushbullet Application	37-39
	4.3.4 Visual Studio Code	39
	4.3.5 Folium Map	40
5	**RESULTS & APPLICATIONS**	**41-44**
	5.1 Results	41-43
	5.2 Applications	43-44

CONCLUSION	**45-46**
Conclusion	45
Future scope	45-46
REFERENCES	**47**
APPENDIX	**48-55**

LIST OF FIGURES

FIG NO.	FIGURE NAME	PAGE NO.
1.1	Characteristics of Embedded System	3
1.2	Microcontroller	5
1.3	Classification of Memory	5
1.4	Architecture of Embedded System	8
1.5	Architecture of IoT	14
1.6	IoT Applications	17
4.1	Block diagram of the proposed system	26
4.2.1	Vibration Sensor	28
4.2.2	Fire Sensor	29
4.2.3	GPS	30
4.2.4	NodeMCU(ESP32)	31
4.2.5	Pin Diagram of NodeMCU(ESP32)	32
4.3.1	Downloading & Install the Arduino IDE	35
4.3.2	Connecting ESP32 & PC through USB Cable	35
5.1.1	Uploading Code to ESP32	41
5.1.2	Getting longitudes & latitudes in Serial Monitor	42
5.1.3	Tracking in Folium Map	42
5.1.4	Notifications of Accidents in Pushbullet App	43

LIST OF ABBREVIATIONS

ACRONYM	DESCRIPTION
IoT	Internet of Things
ES	Embedded Systems
GSM	Global System for Mobile Communications
GPS	Global positioning system
LCD	Liquid Crystal Display
CPU	Central Processing Unit
ROM	Read Only Memory
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
EEPROM	Electrically Erasable Programmable ROM
USB	Universal Serial Bus
LED	Light Emitting Diode
RAM	Random Access Memory
IDE	Integrated Development Environment
GPIO	General Purpose Input/Output
SDA	Serial Data Line
I2C	Inter-Integrated Circuit
SCL	Serial Clock line

CHAPTER-1

INTRODUCTION

1.1 Embedded Systems:

Embedded systems represent the backbone of modern technology, silently powering numerous devices and systems that we encounter in our daily lives. At its core, an embedded system is a specialized computing system designed to perform dedicated functions within a larger mechanical or electrical system. Unlike general-purpose computers, which are designed for a wide range of tasks, embedded systems are tailored to specific applications, often operating with real-time constraints and resource limitations.

At its core, an embedded system can be defined as a specialized computing system that performs dedicated functions within a larger mechanical or electrical system. Unlike traditional computers, which are designed for general-purpose computing tasks, embedded systems are tailored to execute specific functions efficiently and reliably. This specialization allows them to excel in scenarios where performance, reliability, and real-time responsiveness are paramount.

Embedded systems are characterized by their compact size, resource constraints, and real-time operation. They typically consist of a combination of hardware components, such as microcontrollers or microprocessors, memory modules, input/output interfaces, sensors, and actuators, tightly integrated with software code that governs their behavior. This symbiotic relationship between hardware and software enables embedded systems to interact with the physical world, sensing inputs, processing data, and generating outputs to control external devices or systems.

The design and development of embedded systems require a multidisciplinary approach, encompassing aspects of electrical engineering, computer science, and software engineering. Engineers tasked with developing embedded systems must navigate many challenges, including optimizing performance within resource constraints, ensuring reliability in harsh operating environments, and meeting stringent real-time requirements.

Despite their often inconspicuous nature, embedded systems play a pivotal role in shaping the way we live, work, and interact with technology. They power the automation systems that drive modern manufacturing processes, enable the sophisticated features of our smartphones and wearable devices, and control the intricate network of sensors and actuators in smart homes and cities. As technology continues to advance, the prevalence and importance of embedded systems are only expected to grow, further cementing their status as the unsung heroes of the digital age.

In this document, we will delve deeper into the world of embedded systems, exploring their components, design principles, applications, and the challenges inherent in their development. By gaining a deeper understanding of embedded systems, we can appreciate the critical role they play in shaping the technological landscape of the future.

Embedded systems play a pivotal role in hydroponic plant monitoring and control systems, where precise environmental conditions are crucial for optimal plant growth. These systems utilize embedded hardware and software to monitor key parameters such as temperature, humidity, pH levels, nutrient concentrations, and light intensity in real time. Sensors collect data from the growing environment, which is then processed by embedded microcontrollers or processors. Based on predefined algorithms and setpoints, the embedded system regulates environmental variables by controlling actuators such as pumps, valves, fans, and lighting systems. This level of automation ensures that plants receive the conditions for healthy growth while minimizing resource consumption and maximizing yield. Moreover, embedded systems enable remote monitoring and control capabilities, allowing growers to access and adjust settings from anywhere via web interfaces or mobile applications, enhancing efficiency and productivity in hydroponic cultivation.

1.1.1. CHARACTERISTICS OF EMBEDDED SYSTEMS:

Embedded systems possess several distinct characteristics that differentiate them from general-purpose computing systems, underpinning their widespread adoption across various industries and applications.

Firstly, embedded systems are characterized by their specialized functionality tailored to specific tasks or applications. Unlike personal computers or smartphones, which are designed for a wide range of general-purpose computing activities, embedded systems are optimized to perform dedicated functions efficiently and reliably. This specialization allows embedded systems to excel in diverse domains, from automotive control systems and medical devices to industrial automation and consumer electronics.

Secondly, embedded systems are typically constrained by limited resources, including processing power, memory, and energy. Due to their compact size and often remote deployment, embedded devices must operate within stringent resource constraints while delivering high performance and reliability. Engineers designing embedded systems must carefully balance functionality with resource utilization, optimizing algorithms and hardware designs to maximize efficiency within these limitations.

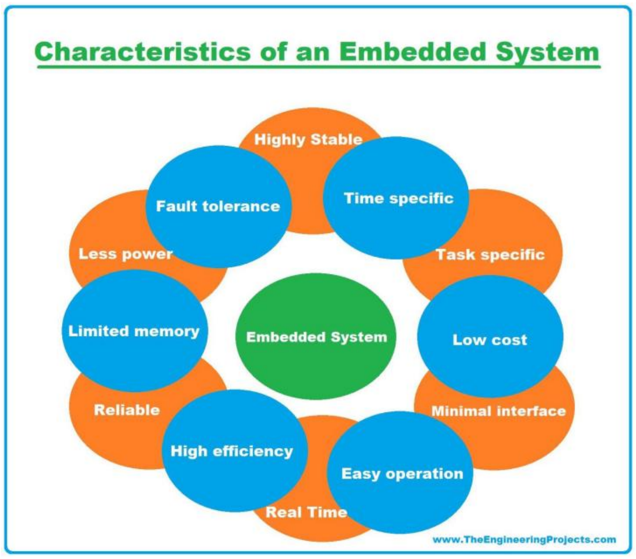


Fig1.1: Characteristics of Embedded Systems

Real-time operation is another defining characteristic of embedded systems, where timely and predictable responses are essential for proper functioning. Many embedded applications require deterministic behavior, where tasks must be completed within specified time constraints to ensure system reliability and safety. Real-time operating systems (RTOS) or specialized scheduling algorithms are commonly employed to manage task execution and prioritize critical operations, enabling embedded systems to meet stringent timing requirements.

Moreover, embedded systems exhibit a high degree of integration, with hardware and software components tightly coupled to perform cohesive functions. Microcontrollers or microprocessors serve as the computational core, interfacing with sensors, actuators, and communication modules to interact with the physical world. This integration enables embedded systems to sense, process, and act upon real-world data in real time, facilitating automation, monitoring, and control in diverse applications.

Lastly, embedded systems often operate in varied and challenging environments, ranging from harsh industrial settings to constrained IoT deployments. Robustness and reliability are paramount, as these systems must withstand temperature fluctuations, humidity, vibration, and other environmental stressors while maintaining uninterrupted operation. Designing embedded systems for reliability involves employing fault-tolerant techniques, redundancy, and comprehensive testing to ensure performance under adverse conditions.

In conclusion, embedded systems exhibit specialized functionality, resource constraints, real-time operation, tight integration, and robustness, making them indispensable in numerous applications where efficiency, reliability, and responsiveness are paramount. Understanding these characteristics is essential for designing and deploying embedded systems effectively across a wide range of domains.

1.1.2. COMPONENTS OF EMBEDDED SYSTEMS:

Embedded systems comprise several essential components that work together to perform dedicated functions efficiently. These components include:

1. MICROCONTROLLER/MICROPROCESSOR:

The microcontroller or microprocessor serves as the brain of the embedded system, executing instructions and coordinating the system's operations. Microcontrollers are highly integrated devices that typically include a CPU, memory, input/output ports, and other peripherals on a single chip. They are favored for embedded applications due to their low cost, low power consumption, and ease of integration. Microprocessors, on the other hand, are more powerful and versatile, often used in applications requiring higher computational performance. The choice between a microcontroller and a microprocessor depends on factors such as processing requirements, cost constraints, and power efficiency. The central processing unit (CPU) of an embedded system, is responsible for executing instructions and controlling the system's operation. Microcontrollers are often preferred for embedded applications due to their integrated peripherals and lower power consumption compared to standalone microprocessors.

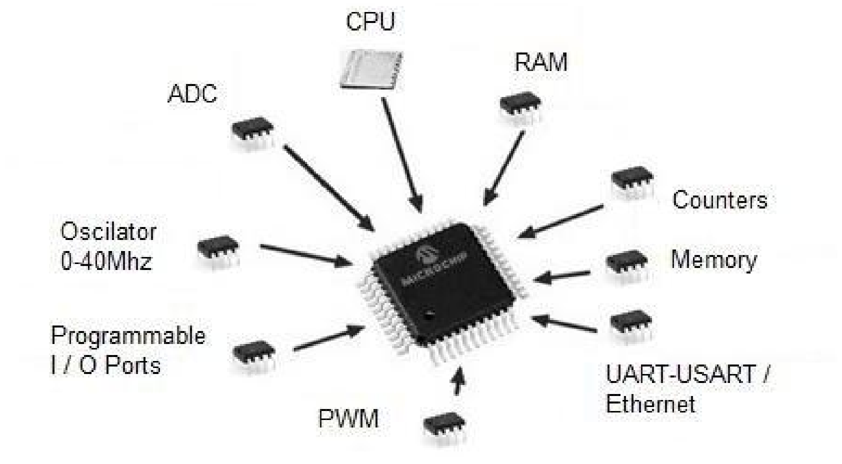


Fig1.2: Microcontroller

2. MEMORY:

Memory plays a critical role in storing program code, data, and configuration settings in embedded systems. Read-only memory (ROM) stores firmware, bootloader, and other essential code that remains unchanged during operation. Random-access memory (RAM) provides temporary storage for data and program variables during execution, facilitating rapid access and manipulation. Electrically Erasable Programmable Read-Only Memory (EEPROM) serves as non-volatile memory for storing configuration parameters, calibration data, and other settings that need to persist across power cycles. The size and type of memory used in an embedded system depend on factors such as program size, data storage requirements, and performance considerations.

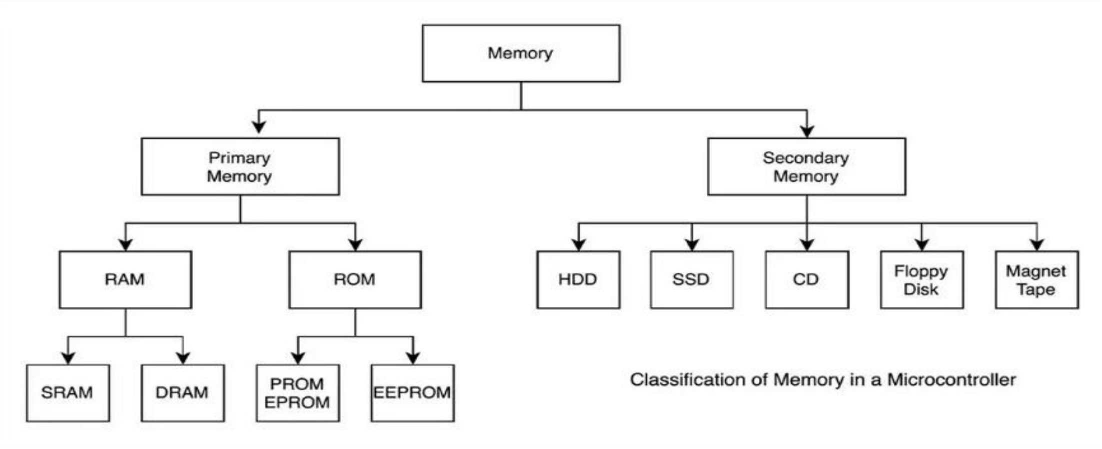


Fig1.3: Classification of Memory

3.INPUT/OUTPUT(I/O) INTERFACES:

Input/output interfaces enable embedded systems to interact with the external environment, facilitating communication with sensors, actuators, and other peripheral devices. General-Purpose Input/Output (GPIO) pins provide versatile digital input and output capabilities, allowing the microcontroller to interface with a wide range of external components. Analog-to-digital converters (ADCs) convert analog signals from sensors into digital values for processing by the microcontroller, while Digital-to-Analog Converters (DACs) perform the opposite function, converting digital signals into analog voltages for driving actuators or generating output signals. These interfaces enable embedded systems to sense inputs, process data, and control external devices, enabling a wide range of applications.

4. PERIPHERALS:

Peripherals are additional hardware components integrated into embedded systems to provide specific functionality beyond the basic microcontroller capabilities. Serial communication interfaces such as UART, SPI, and I2C facilitate communication with external devices, sensors, or other embedded systems. Timers and counters generate precise timing intervals, and PWM signals, or capture external events, essential for tasks such as generating accurate clock signals, measuring time intervals, or controlling motor speed. Interrupt controllers manage interrupts from external sources, allowing the microcontroller to respond promptly to time-critical events without wasting CPU cycles polling for status changes. These peripherals enhance the capabilities of embedded systems, enabling them to perform complex tasks efficiently and reliably.

5. POWER SUPPLY MANAGEMENT:

Power supply and management circuits ensure that embedded systems receive a stable and efficient power supply to operate reliably. Power management circuits regulate voltage levels, manage power consumption, and provide features such as sleep modes to optimize energy usage and extend battery life in portable devices. Protection mechanisms such as overvoltage protection, overcurrent protection, and reverse polarity protection safeguard the system against damage from power surges or faults. These components ensure that the embedded system operates within safe operating limits, maximizing reliability and longevity.

1.1.3. EMBEDDED SYSTEM ARCHITECTURE:

Embedded system architecture refers to the structural design and organization of hardware and software components within an embedded system, defining how they interact to perform specific functions efficiently and reliably. This architecture typically comprises hardware components, including microcontrollers or microprocessors, memory modules, input/output interfaces, peripherals, and communication modules, as well as software components, such as firmware, operating systems, and application software. The architecture of an embedded system is influenced by factors such as performance requirements, resource constraints, real-time operation, and scalability.

At the core of embedded system architecture lies the choice between using a microcontroller or a microprocessor. Microcontrollers integrate a CPU, memory, and various peripherals on a single chip, providing a cost-effective and power-efficient solution for embedded applications with modest computational requirements. Microprocessors, on the other hand, offer higher processing power and

flexibility but may require additional external components for interfacing with peripherals and managing system operations. The selection between these options depends on factors such as processing requirements, power constraints, cost considerations, and the desired level of integration.

Memory architecture plays a crucial role in determining the performance and functionality of embedded systems. Read-only memory (ROM) stores firmware, bootloader, and other essential code that remains unchanged during operation,ensuring that the system can boot reliably and execute critical tasks. Random-access memory (RAM) provides temporary storage for data and program variables during execution, facilitating rapid access and manipulation. Electrically Erasable Programmable Read-Only Memory (EEPROM) serves as non-volatile memory for storing configuration parameters, calibration data, and other settings that need to persist across power cycles. The size and type of memory used in an embedded system depend on factors such as program size, data storage requirements, and performance considerations.

Input/output (I/O) interfaces enable embedded systems to interact with the external environment, facilitating communication with sensors, actuators, and other peripheral devices. General-Purpose Input/Output (GPIO) pins provide versatile digital input and output capabilities, allowing the microcontroller to interface with a wide range of external components. Analog-to-digital converters (ADCs) convert analog signals from sensors into digital values for processing by the microcontroller, while Digital-to-Analog Converters (DACs) perform the opposite function, converting digital signals into analog voltages for driving actuators or generating output signals. These interfaces enable embedded systems to sense inputs, process data, and control external devices, enabling a wide range of applications.

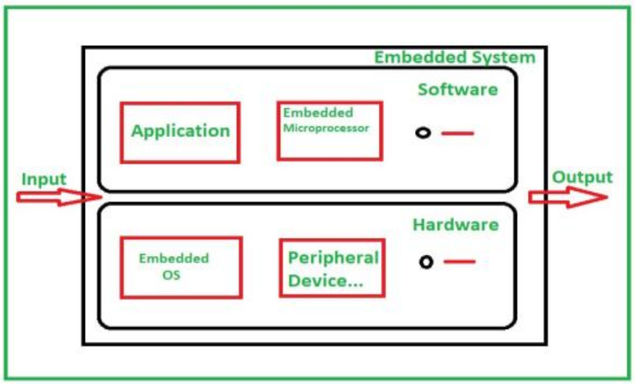


Fig1.4: Architecture of Embedded System

The architecture of an embedded system also includes communication modules for exchanging data with other devices or networks. Wired interfaces such as Ethernet, USB, or CAN bus provide reliable and high-speed communication for applications requiring fast data transfer rates and deterministic behavior. Wireless technologies like Wi-Fi, Bluetooth, Zigbee, or LoRa enable connectivity in IoT applications, allowing embedded systems to communicate over long distances and interact with remote devices or cloud services. The selection of communication modules depends on factors such as data rate requirements, range, power consumption, and compatibility with existing infrastructure.

In conclusion, embedded system architecture encompasses the design and organization of hardware and software components within an embedded system, defining how they interact to perform specific functions efficiently and reliably. By understanding the architectural principles and components of embedded systems, engineers can design and develop systems tailored to meet the requirements of diverse applications across industries.

1.1.4. ADVANTAGES OF EMBEDDED SYSTEMS:

Embedded systems offer a plethora of advantages across various domains, contributing to their widespread adoption and utilization. Here are some key advantages of embedded systems:

1. **Specialized Functionality:** Embedded systems are designed to perform dedicated functions efficiently and reliably. By focusing on specific tasks, they can achieve high performance and optimize resource utilization, making them ideal for applications where tailored functionality is paramount.

2. **Compact Size and Low Cost:** Embedded systems often integrate multiple hardware components onto a single chip, resulting in compact and cost-effective solutions. This makes them suitable for applications with space constraints or cost-sensitive requirements, such as consumer electronics, automotive systems, and IoT devices.

3. **Low Power Consumption:** Many embedded systems are designed to operate on low power, making them suitable for battery-powered or energy-efficient applications. By optimizing hardware and software components for power efficiency, embedded systems can prolong battery life, reduce energy consumption, and minimize environmental impact.

4. **Real-Time Operation:** Embedded systems are capable of executing tasks with precise timing requirements, ensuring timely responses to external events or inputs. This real-time responsiveness is essential for applications such as industrial automation, robotics, medical devices, and automotive safety systems, where delays or inaccuracies could have serious consequences.

5. **High Reliability and Stability:** Embedded systems are often deployed in mission-critical environments where reliability and stability are paramount. By using hardware and software components specifically designed for robustness, embedded systems can operate continuously under harsh conditions without failure, enhancing system uptime and minimizing downtime.

6. **Ease of Integration and Customization:** Embedded systems offer flexibility in terms of hardware and software integration, allowing developers to tailor solutions to meet specific requirements. Modular design approaches enable seamless integration of additional functionality or peripherals, while software frameworks and development tools facilitate customization and rapid prototyping.

7. **Scalability and Versatility:** Embedded systems can be scaled to accommodate a wide range of applications and performance requirements. Whether deploying a simple sensor node in an IoT network or a complex control system in industrial automation, embedded systems offer scalability and versatility to adapt to evolving needs and technological advancements.

8. **Remote Monitoring and Control:** With the advent of IoT technologies, embedded systems enable remote monitoring and control capabilities, allowing users to access and manage devices from anywhere with an internet connection. This remote accessibility enhances convenience, efficiency, and productivity in various applications, including smart homes, industrial monitoring, and healthcare systems.

9. **Security:** Embedded systems can implement security measures to protect against unauthorized access, data breaches, and tampering. Techniques such as encryption, authentication, access control, and secure boot ensure the integrity and confidentiality of data, safeguarding sensitive information in embedded applications.

10. **Innovation and Advancement:** Embedded systems drive innovation and advancement across industries by enabling the development of new technologies and applications. From smart devices and wearables to autonomous vehicles and smart cities, embedded systems continue to push the boundaries of what is possible, shaping the future of technology and society.

Overall, embedded systems offer a myriad of advantages, including specialized functionality, compact size, low power consumption, real-time operation, reliability, scalability, and security, making them indispensable in countless applications across industries.

1.1.5. APPLICATIONS OF EMBEDDED SYSTEMS:

Embedded systems (ES) find applications across a diverse range of industries and domains due to their ability to perform specific tasks efficiently and reliably. Here are some notable applications of embedded systems:

1. Consumer Electronics: Embedded systems power various consumer electronics devices, including smartphones, tablets, smart TVs, digital cameras, gaming consoles, and home appliances such as washing machines, refrigerators, and microwave ovens. These systems provide advanced features, user interfaces, and connectivity options while ensuring seamless operation and energy efficiency.

2. Automotive Industry: Embedded systems play a critical role in modern vehicles, controlling various functions such as engine management, transmission control, anti-lock braking systems (ABS), airbag deployment, entertainment systems, navigation, and advanced driver assistance systems (ADAS). They enable safer, more efficient, and comfortable driving experiences while also supporting vehicle connectivity and autonomous driving features.

3. Industrial Automation: Embedded systems are extensively used in industrial automation for monitoring and controlling manufacturing processes, machinery, robots, and other equipment. They facilitate tasks such as process control, data acquisition, motion control, programmable logic control (PLC), supervisory control and data acquisition (SCADA), and predictive maintenance, improving productivity, efficiency, and reliability in industrial settings.

4. Healthcare and Medical Devices: Embedded systems are integral to medical devices and healthcare systems, supporting functions such as patient monitoring, diagnostic imaging, infusion pumps, implantable devices, electronic health records (EHR), telemedicine, and medical robotics. They enable accurate diagnosis, treatment, and management of health conditions while ensuring patient safety and data privacy.

5. IoT (Internet of Things): Embedded systems form the backbone of the IoT ecosystem, connecting physical objects and devices to the internet and enabling data exchange, monitoring, and control. IoT applications span various domains, including smart homes, smart cities, agriculture, environmental monitoring, asset tracking, energy management, and industrial IoT (IIoT), revolutionizing how we interact with and manage the world around us.

6. Aerospace and Defense: Embedded systems are prevalent in aerospace and defense applications, powering avionics systems, unmanned aerial vehicles (UAVs), satellites, missile guidance systems, radar systems, communication systems, and electronic warfare (EW) systems. They provide critical functionalities such as navigation, communication, surveillance, and target tracking, ensuring mission success and national security.

7. Telecommunications: Embedded systems play a vital role in telecommunications infrastructure, including network switches, routers, base stations, access points, and optical transmission systems. They support high-speed data processing, packet switching, signal processing, and network management, enabling reliable and efficient communication services for businesses and consumers.

8. Energy Management and Smart Grids: Embedded systems contribute to energy management and smart grid solutions, optimizing energy generation, distribution, and consumption. They support functions such as smart metering, demand response, renewable energy integration, grid monitoring, and power quality management, promoting energy efficiency, reliability, and sustainability in power systems.

9. Environmental Monitoring: Embedded systems are employed in environmental monitoring applications to collect data on air quality, water quality, weather conditions, and pollution levels. They support tasks such as sensor data acquisition, data logging, real-time analysis, and remote monitoring, facilitating environmental assessment, protection, and management efforts.

10. Retail and Point-of-Sale (POS) Systems: Embedded systems are utilized in retail and POS systems for inventory management, sales processing, payment processing, customer engagement, and analytics. They enable efficient operation of retail businesses, improve customer experiences, and support omnichannel retail strategies.

These applications illustrate the versatility and ubiquity of embedded systems across various sectors, driving innovation, efficiency, and advancements in technology and society.

1.2. INTERNET OF THINGS:

The Internet of Things (IoT) represents a transformative paradigm shift in the way we interact with technology, enabling seamless connectivity and communication between physical objects and the digital world. At its core, IoT encompasses a vast network of interconnected devices, sensors, actuators, and systems, all equipped with embedded technology and capable of exchanging data over the internet. This interconnected ecosystem facilitates the collection, analysis, and utilization of real-time data, enabling intelligent decision-making, automation, and enhanced user experiences across diverse domains.

One of the key characteristics of IoT is its ability to bridge the gap between the physical and digital realms, enabling objects and devices to communicate, collaborate, and operate in concert with one another. By embedding sensors and communication capabilities into everyday objects and environments, IoT systems can monitor, measure, and analyze various parameters and phenomena, from environmental conditions and energy consumption to human behavior and machine performance. This ubiquitous connectivity fosters a new era of smart and interconnected ecosystems, where information flows seamlessly between devices, systems, and users.

Fig1.5: Internet of Things

IoT finds applications across a wide range of industries and domains, revolutionizing sectors such as healthcare, agriculture, transportation, manufacturing, energy, retail, and smart cities. In healthcare, IoT-enabled devices and wearable technologies facilitate remote patient monitoring, personalized healthcare management, and early intervention strategies, improving patient outcomes and reducing healthcare costs. In agriculture, IoT sensors and actuators enable precision farming techniques, optimizing irrigation, fertilization, and crop management practices to enhance yield, quality, and sustainability.

In transportation, IoT systems support connected vehicles, intelligent transportation systems (ITS), and autonomous driving technologies, enhancing safety, efficiency, and mobility in urban and rural environments. In manufacturing, IoT-enabled smart factories and industrial automation systems streamline production processes, improve quality control, and enable predictive maintenance, maximizing productivity and competitiveness. In energy, IoT solutions enable smart grid management, renewable energy integration, and energy efficiency optimization, reducing waste and carbon emissions while ensuring reliable and sustainable power supply.

Moreover, IoT contributes to the development of smart cities and urban environments, where interconnected systems and infrastructure support efficient resource management, public safety, transportation networks, environmental sustainability, and citizen engagement. By leveraging IoT technologies, cities can monitor and manage critical assets and services, optimize traffic flow and parking, reduce energy consumption and pollution, and enhance the overall quality of life for residents and visitors.

However, the proliferation of IoT also raises concerns regarding data privacy, security, interoperability, and ethical considerations. As IoT devices collect and transmit vast amounts of sensitive data, ensuring robust cybersecurity measures, data encryption, access controls, and privacy safeguards is essential to mitigate risks and protect user privacy. Moreover, addressing interoperability challenges and establishing industry standards and protocols are critical for ensuring seamless integration and compatibility between different IoT devices, platforms, and ecosystems.

1.2.1. COMPONENTS AND ARCHITECTURE OF IOT:

The Internet of Things (IoT) architecture comprises several layers, each consisting of various components that work together to enable connectivity, data exchange, and intelligent decision-making. Here's an overview of the components and architecture of IoT:

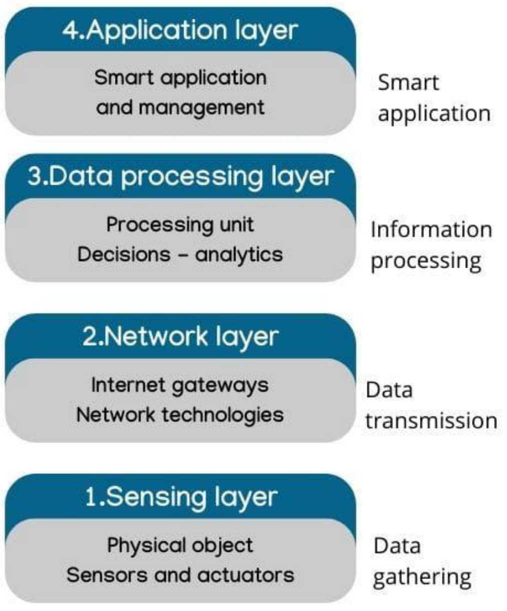

4.Application layer
Smart application and management
Smart application
3.Data processing layer
Processing unit Decisions – analytics
Information processing
2.Network layer
Internet gateways Network technologies
Data transmission
1.Sensing layer
Physical object Sensors and actuators
Data gathering

Fig1.6: Architecture of IoT

1. Perception Layer:

Sensors and Actuators: These are physical devices that capture data from the environment (sensors) or perform actions based on commands received from the IoT system (actuators). Sensors can measure parameters such as temperature, humidity, light, motion, and more, while actuators can control devices like motors, valves, and switches.

2. Network Layer:

Communication Protocols: Various communication protocols are used to facilitate data transmission between IoT devices and the central processing units. These protocols may include Wi-Fi, Bluetooth, Zigbee, Z-Wave, LoRaWAN, MQTT, CoAP, and others, each offering different features such as range, bandwidth, power consumption, and scalability.

Gateways: Gateways act as intermediaries between IoT devices and the cloud or central server. They aggregate data from multiple devices, perform protocol translation, data preprocessing, and filtering tasks, and ensure secure and reliable communication with the backend systems.

3. Middleware Layer:

Data Processing and Analytics: This component processes the raw data collected from sensors, performs data aggregation, filtering, normalization, and analysis, and generates meaningful insights or actionable information. Data processing techniques may include real-time stream processing, batch processing, machine learning, and predictive analytics.

Device Management: Device management functionalities include device provisioning, authentication, configuration, firmware updates, and remote monitoring and control. It ensures that IoT devices are securely onboarded, managed, and maintained throughout their lifecycle.

4. Application Layer:

IoT Applications: These are software applications or services built on top of the IoT infrastructure to address specific use cases and business requirements. IoT applications can range from smart home automation, industrial monitoring and control, asset tracking, environmental monitoring, healthcare management, to smart city solutions.

User Interfaces: User interfaces enable users to interact with IoT systems, visualize data, configure settings, and receive notifications or alerts. User interfaces may include web-based dashboards, mobile applications, command-line interfaces, voice assistants, or augmented reality interfaces, tailored to the needs of different user groups and use cases.

5. Cloud/Backend Layer:

Cloud Infrastructure: Cloud platforms provide scalable and reliable infrastructure for hosting IoT applications, storing and processing data, and enabling advanced analytics and machine learning capabilities. Cloud services such as storage, compute, database, messaging, and identity management are leveraged to build and deploy IoT solutions.

Data Storage and Database Systems: Data collected from IoT devices is stored in databases or data lakes, where it can be accessed, queried, and analyzed by applications and users. Database systems such as relational databases, NoSQL databases, time-series databases, and big data platforms are used to store and manage IoT data effectively.

6. Security Layer:

Authentication and Authorization: Security mechanisms such as authentication and authorization ensure that only authorized users and devices can access IoT resources and data. Techniques such as cryptographic protocols, digital certificates, and access control policies are employed to authenticate users and devices and enforce access rights.

Data Encryption and Privacy: Data encryption techniques are used to protect sensitive information transmitted between IoT devices, gateways, and backend systems, ensuring confidentiality and integrity. Privacy-enhancing technologies such as anonymization, pseudonymization, and data minimization are implemented to safeguard user privacy and comply with regulations.

By understanding the components and architecture of IoT, organizations and developers can design and deploy scalable, secure, and efficient IoT solutions to address a wide range of use cases and unlock the full potential of connected devices and systems.

1.2.2. APPLICATIONS AND IMPACT OF IOT:

Smart Home Automation: Smart home devices connected through IoT technology allow homeowners to remotely control and automate various aspects of their homes, such as lighting, thermostats, security cameras, and appliances. This integration enhances convenience and energy efficiency while offering improved security through remote monitoring and control.

Industrial IoT (IIoT): Industrial IoT applications revolutionize manufacturing and supply chain operations by leveraging sensors, data analytics, and connectivity to optimize processes, monitor equipment health, and enhance productivity. IIoT solutions enable predictive maintenance, real-time monitoring, and seamless integration across factory floors, leading to improved efficiency and reduced downtime.

Smart Cities: IoT technologies facilitate the development of smart cities by integrating digital infrastructure with urban services and utilities. Through applications like intelligent transportation systems, environmental monitoring, and public safety enhancements, smart cities improve efficiency, sustainability, and quality of life for residents while enabling more effective urban planning and resource management.

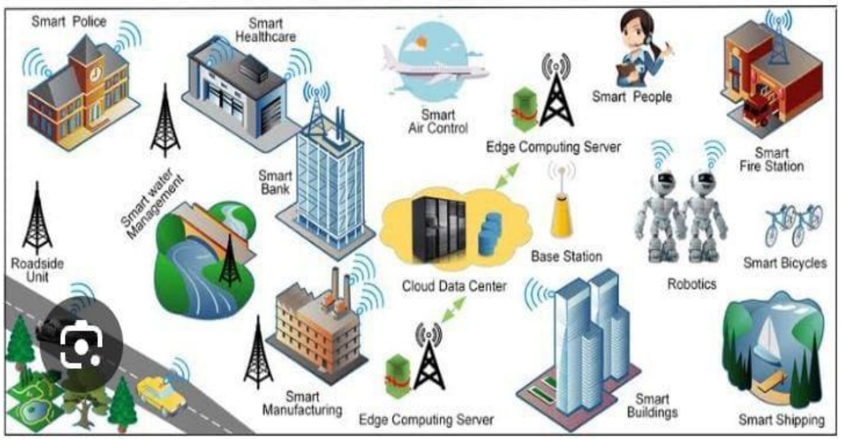


Fig1.7: IoT Applications

Healthcare and Telemedicine: IoT-enabled medical devices and remote monitoring solutions revolutionize healthcare delivery by enabling continuous monitoring of patient's vital signs, medication adherence, and chronic conditions. Telemedicine platforms connect patients with healthcare providers remotely, improving access to care and enabling personalized health management while reducing healthcare costs and hospital readmissions.

Agriculture and Smart Farming: In agriculture, IoT solutions monitor soil conditions, crop health, and weather patterns to optimize farming practices and resource utilization. By providing farmers with real-time data and actionable insights, smart farming techniques improve crop yields, conserve water, and minimize environmental impact, contributing to sustainable agriculture and food security.

Retail and Supply Chain Management: IoT technologies transform retail operations and supply chain management by enabling real-time inventory tracking, supply chain visibility, and personalized customer experiences. With IoT-connected sensors and smart shelves, retailers optimize inventory management, reduce stockouts, and enhance customer satisfaction through personalized marketing and shopping experiences, driving increased sales and operational efficiency.

CHAPTER 2

LITERATURE SURVEY

2.LITERATURE SURVEY

In 2017 submission by Prawat Chaiprapa and Supaporn Kiattisin presented an innovative "Real-Time GPRS Vehicle Tracking System Displayed on a Google-Map-Based Website" This system aimed to revolutionize vehicle tracking by integrating GPS technology and GPRS modules to provide accurate real-time monitoring. By utilizing GPS modules, the system captured precise location data of vehicles, which was then transmitted through GPRS to a web-based platform. On this platform, a dynamic Google Map interface was developed using PHP and AJAX, allowing users to visualize the real-time positions of vehicles. Despite encountering a minor limitation in tracking accuracy, with a discrepancy of approximately 5 meters due to hardware constraints, the system still offered valuable real-time tracking capabilities for vehicle monitoring and management purposes.

The submission marked a significant advancement in vehicle tracking technology, offering a comprehensive solution for real-time monitoring and management. By leveraging the capabilities of GPS and GPRS technologies, the system provided users with accurate and up-to-date information on the location of vehicles. Although constrained by minor tracking inaccuracies, the system's overall functionality remained robust, demonstrating its potential for enhancing efficiency and security in various transportation and logistics applications.[1]

In 2014 submission by P. Zhou and M. Li unveiled an inventive solution titled "How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing" This groundbreaking system leveraged the collective contributions of individuals through their mobile devices to predict bus arrival times accurately. By harnessing participatory sensing, where a large group actively shares data, the system aimed to empower commuters with real-time information, enhancing their ability to plan their journeys effectively and minimize waiting times. Despite the system's innovative approach, it faced inherent challenges, such as the potential unreliability of the collected data and the risk of dwindling participation over time. However, the submission represented a significant leap forward in transportation technology, demonstrating the potential of participatory sensing to revolutionize public transit systems and improve the overall commuting experience for passengers.

The introduction of "How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing" marked a milestone in the field of transportation technology. Authored by P. Zhou and M. Li, this system capitalized on the ubiquity of mobile phones to gather real-time data from a diverse group of contributors. By harnessing the collective wisdom of commuters, the system aimed to provide accurate predictions of bus arrival times, offering passengers greater convenience and efficiency in their travel planning. Despite the challenges associated with maintaining data reliability and sustaining participation levels, the submission exemplified the potential of participatory sensing to transform public transportation systems and pave the way for smarter, more responsive urban mobility solutions.[2]

In May 2016 submission by KuanYew Tan and KokSheik Wong introduced a cost-effective solution titled "Low-Cost Campus Bus Tracker using WiFi Access Points." Leveraging the widespread availability of WiFi internet access points, the system aimed to track campus buses' locations without relying on traditional GPS and GSM modules. While GPS technology has traditionally been instrumental in fleet management, the authors recognized the potential of WLAN infrastructure for object tracking. The system employed two main methods: proximity estimation, which calculates an object's location as it approaches a known point, and scene analysis, an image processing-based technique comparing current scenes to a database of tagged images with location information. These innovative approaches aimed to provide accurate bus tracking capabilities within campus environments, utilizing existing infrastructure to minimize costs and enhance accessibility.

However, despite its innovative approach, the system encountered certain limitations. One significant drawback was the absence of GPS and GSM modules for precise location tracking and data sharing with users' mobile devices. Additionally, establishing WiFi access points across the campus posed challenges, potentially hindering the system's deployment and effectiveness. These limitations underscored the need for further refinement and consideration of alternative solutions to overcome the reliance on GPS and GSM technologies. Despite these challenges, the submission highlighted the potential of leveraging existing infrastructure, such as

WiFi access points, to develop cost-effective solutions for campus bus tracking, with implications for enhancing transportation management systems in similar environments.[3]

In 2016 submission by N. Chadil, A. Russameesawang, and P. Keeratiwintakor utilizes GPS technology and Google Earth for efficient monitoring and management. This open system employs free and open-source software, coupled with readily available commodity hardware, ensuring accessibility and affordability. The GPS tracking device, functioning as an embedded system, communicates location data to the server via the GPRS network. Upon reception, the server, typically a personal computer, stores the information in a database and formats it in a manner compatible with Google Earth or Google Maps for easy search and display. This integration enables real-time visualization of vehicle locations, enhancing tracking and management capabilities.

However, despite its strengths, the system presents a notable drawback concerning communication with users/owners. Following vehicle tracking, there's a requirement to send location updates to users or owners via their cell phones. Regrettably, the system lacks a GSM module for transmitting these messages. This limitation hinders the system's ability to provide comprehensive communication functionalities, potentially impacting user experience and the system's overall utility. Addressing this drawback would involve integrating a GSM module to enable seamless communication between the tracking system and users/owners, enhancing its functionality and usability in real-world scenarios.[4]

CHAPTER 3

EXISTING SYSTEM

3.1. EXISTING SYSTEM:

However, relying on WhatsApp for sharing bus locations also comes with its limitations and disadvantages. Firstly, WhatsApp is primarily designed as a messaging platform rather than a dedicated tracking system. As a result, it may lack certain features and functionalities that are essential for efficient bus tracking, such as real-time updates, accurate location data, and advanced route optimization. This can lead to inconsistencies in the information provided to passengers and may result in frustration or confusion, particularly during peak travel times or in areas with complex transit networks.

Additionally, the use of WhatsApp for sharing bus locations may pose privacy and security concerns for both passengers and operators. Since WhatsApp is a third-party platform, operators have limited control over the privacy settings and security measures implemented within the messaging service. This raises concerns about the protection of sensitive passenger data, such as location information and personal contact details, from unauthorized access or misuse. Moreover, relying on WhatsApp for critical communication may expose operators to risks such as service outages, data breaches, or disruptions in service, which could compromise the reliability and effectiveness of the bus tracking system in providing timely and accurate information to passengers.

3.2. DRAWBACKS OF EXISTING SYSTEM:

Sharing location through WhatsApp, while convenient, has several drawbacks when used as a primary method for bus tracking:

1. Limited Automation: Sharing location via WhatsApp requires manual input from the bus driver or personnel. This manual process can be time-consuming and prone to errors, as it relies on individuals to actively share their location updates.

2.Inaccuracy and Delays: Because sharing location on WhatsApp is manual, the frequency and accuracy of location updates can vary. Bus locations may not be updated in real-time, leading to delays and inaccuracies in tracking the bus's actual position.

3.Dependency on Network Connectivity: Sharing location through WhatsApp relies on internet connectivity. In areas with poor network coverage or during network outages, the ability to share location updates may be compromised, further hindering accurate tracking.

4.Limited Integration: WhatsApp's location-sharing feature is not designed for integration with external systems or applications. As a result, it may be challenging to integrate WhatsApp-based tracking with other transportation management systems or to automate data processing.

5. Scalability: WhatsApp may not be scalable for large-scale bus tracking operations. Managing location updates for multiple buses and coordinating communication with various stakeholders can become cumbersome and inefficient.

6.Reactive Accident Response: In the event of an accident, the response from transit authorities is typically reactive rather than proactive. Emergency services are often notified by bystanders or through conventional communication channels, which can lead to delays in assistance reaching the accident site. This delay in response time can have serious implications for passenger safety and the severity of injuries sustained.

CHAPTER 4
PROPOSED SYSTEM

4.1. PROPOSED SYSTEM

The Real-Time Bus Tracking and Accident Detection System consists of two main components: the Real-Time Bus Tracking system and the Accident Detection system.

Real-TimeBus Tracking System:

1.Utilizes GPS and IoT technologies to monitor the location and movement of buses in real-time.

2.Allows passengers to access bus location information through a mobile application.

3.Enable passengers to plan their journeys more effectively and reduce waiting times at bus stops.

4.Provide transit authorities with data to optimize bus routes, improve scheduling, and enhance operational efficiency.

5.For Continuously tracking we are using Folium Map.

Accident Detection System:

1.Integrated into buses, utilizing sensors such as Vibration sensor and Fire sensor.

2.Monitors the vehicle's behavior and surroundings continuously.

3.Detects irregularities or collisions that may indicate an accident.

4.Automatically sends alerts to emergency services and relevant authorities in the event of an accident to Pushbullet Application .

Key Features:

Integration of GPS and IoT technologies for real-time bus tracking.

1.Accessibility for passengers through mobile applications .

2.Utilization of sensors like Vibration sensor and Fire sensor for accident detection.

3.Automatic alert system for emergency response in case of accidents.

4.Data collection and analysis for optimizing bus routes, scheduling, and operational efficiency.

4.1.1. BLOCK DIAGRAM

The proposed Real-Time Bus Tracking and Accident Detection System is designed to overcome the limitations of existing public transportation systems by leveraging modern technologies for enhanced efficiency, safety, and reliability. At its core, the system integrates several key components to enable real-time tracking of buses and prompt detection of accidents.

The proposed system integrates a variety of hardware and software components to create an advanced monitoring and notification system tailored for bus accidents and fire incidents. At its core, the system utilizes sensors such as vibration and fire sensors to detect potential emergencies on the bus. Upon detecting significant vibrations or signs of fire, these sensors trigger the system to initiate emergency protocols. The Node MCU, functioning as the central processing unit, collects data from the sensors, processes it, and activates appropriate actions based on predefined algorithms and conditions. Additionally, the GPS module continuously tracks the bus's coordinates, providing real-time location data for accurate positioning and mapping of its route.

For notifications and communication, the system relies on the Pushbullet app installed on mobile devices. When an accident or fire incident occurs, the system promptly sends notifications to designated mobile devices via Pushbullet, alerting authorities, emergency responders, and relevant stakeholders about the situation. Meanwhile, the Folium map integration visually displays the bus's real-time location on an interactive map, facilitating continuous tracking and monitoring. This comprehensive approach enables swift response and assistance efforts in the event of emergencies, enhancing safety measures for bus passengers and enabling efficient incident management. Overall, the system offers a robust solution for enhancing safety and security measures in public transportation systems.

Overall, the proposed system offers a comprehensive solution for real-time bus tracking and accident detection, addressing the shortcomings of existing systems. By leveraging advanced technologies and integrating hardware and software components, the system enhances operational efficiency, improves passenger safety, and provides a more reliable and responsive public transportation experience.

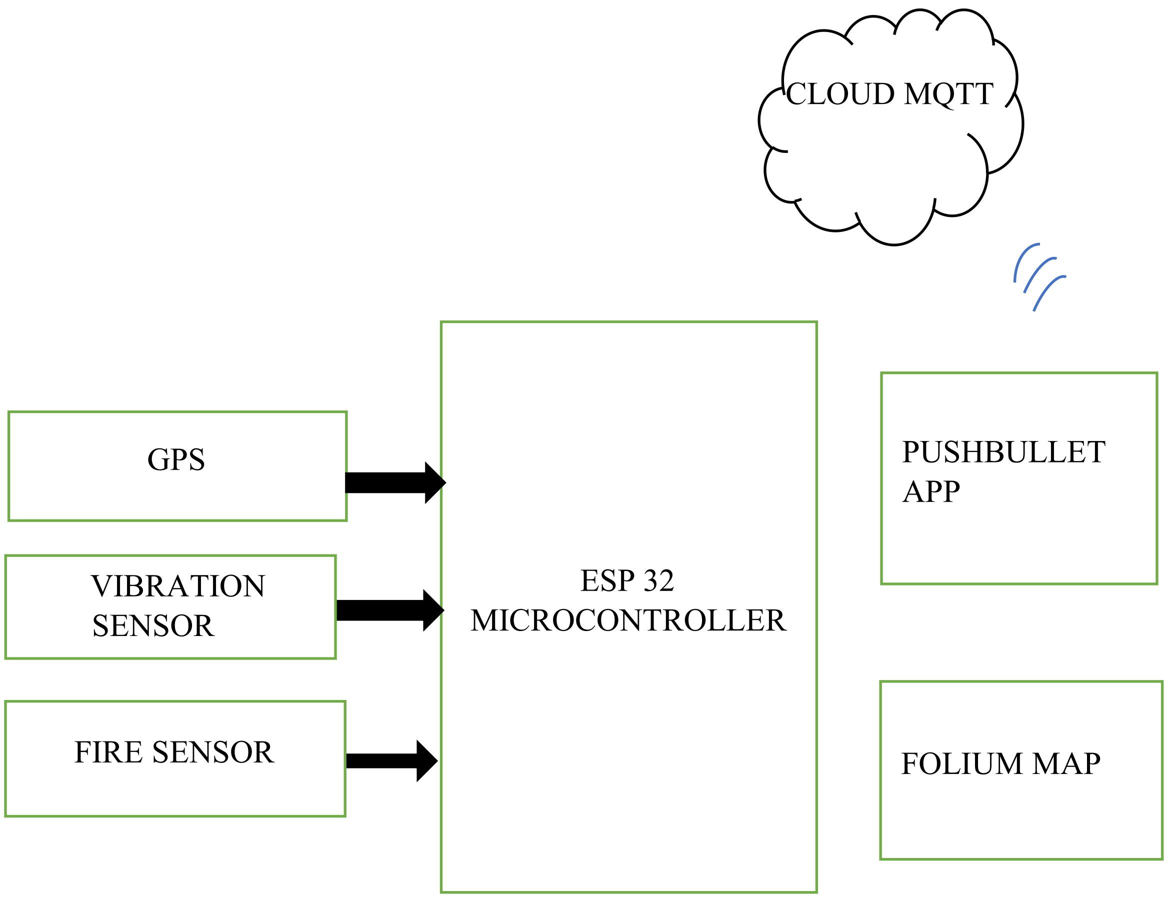


Fig4.1: Block Diagram of Proposed System

4.1.2. WORKING PRINCIPLE

The proposed system operates on a multi-faceted approach to monitor and respond to bus accidents and fire incidents in real-time. At its core, the system relies on a network of sensors strategically placed within the bus. These sensors include vibration sensors to detect sudden impacts or collisions and fire sensors to identify signs of fire or excessive heat. When triggered, these sensors send signals to the Node MCU, serving as the system's central processing unit. The Node MCU processes incoming data from the sensors and evaluates it against predefined algorithms and thresholds. If the data indicates a potential emergency, the Node MCU initiates the appropriate response protocols.

In parallel, the system incorporates GPS technology to continuously track the bus's location in real-time. The GPS module installed on the bus communicates with satellites to determine the precise coordinates of the vehicle. This location data is transmitted to the Node MCU, allowing the system to maintain an accurate record of the bus's movements and where abouts at all times. In the event of an emergency, the GPS data provides crucial information about the bus's location, enabling swift response and assistance from emergency services and relevant authorities.

To facilitate communication and notification, the system leverages the Pushbullet app installed on mobile devices. When an accident or fire incident occurs, the Node MCU triggers the system to send immediate notifications to designated mobile devices via Pushbullet. These notifications alert authorities, emergency responders, and relevant stakeholders about the situation, ensuring prompt action and assistance. Additionally, the system integrates with the Folium map library to visually display the bus's real-time location on an interactive map. This feature enhances situational awareness for responders and stakeholders, enabling them to coordinate their efforts effectively.

Overall, the system's integrated approach enables rapid detection, notification, and response to bus emergencies, enhancing safety and security for passengers and personnel alike.In summary, the working principle of the proposed system involves continuous monitoring of bus conditions, immediate incident detection, and efficient communication to relevant stakeholders through a user-friendly interface. By leveraging advanced technologies and seamless integration, the system enhances public transportation safety, operational efficiency, and passenger satisfaction.

4.2. HARDWARE REQUIREMENTS

4.2.1. VIBRATION SENSOR

Vibration sensors play a pivotal role in modern bus tracking systems, serving as vital components for monitoring and ensuring the safety and operational integrity of vehicles. These sensors are designed to detect mechanical vibrations generated by various sources, including engine operation, road conditions, and passenger movements. By measuring and analyzing these vibrations, vibration sensors provide valuable insights into the health and performance of buses, enabling proactive maintenance and fault detection. For example, abnormal vibrations may indicate issues such as engine malfunctions, wheel imbalance, or worn-out components, allowing fleet managers to address potential problems before they escalate.

In bus tracking systems, vibration sensors serve multiple functions beyond mere vibration detection. They also function as motion sensors, providing real-time data on the movement and activity of buses. This data is essential for tracking the whereabouts of buses, determining their operational status (such as whether they are in motion or stationary), and identifying any unusual activity, such as sudden stops or accelerations. By monitoring vibrations and motion patterns, vibration sensors contribute to the accurate tracking and monitoring of bus fleets, facilitating efficient route planning, schedule adherence, and operational optimization.

Moreover, vibration sensors enhance the security and safety of buses and passengers by detecting unauthorized entry or tampering with the vehicle. In the event of a collision or accident, vibration sensors can detect the impact and trigger emergency response procedures, such as activating airbags or sending alerts to authorities. Additionally, vibration sensors can be integrated with other sensors and technologies in comprehensive bus tracking systems, providing a holistic view of vehicle operations and performance. Overall, vibration sensors are indispensable components of bus tracking systems, enabling proactive maintenance, accurate tracking, and enhanced safety and security measures for passengers and vehicles alike.

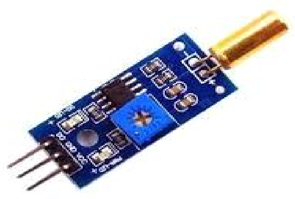


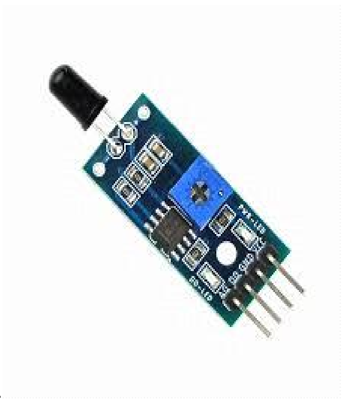
Fig 4.2.1: Vibration Sensor

4.2.2. FIRE SENSOR

Fire sensors are crucial elements in bus tracking systems, playing a vital role in ensuring passenger safety and protecting the vehicle from fire-related incidents. These sensors are designed to promptly detect the presence of fire or excessive heat within the bus cabin. They operate using various detection principles, including smoke detection, heat detection, and flame detection. By continuously monitoring environmental conditions within the bus, fire sensors provide early warnings of potential fire hazards, enabling swift response and evacuation procedures.

In the context of bus tracking, fire sensors serve as essential safety mechanisms. Smoke detectors, for example, can detect the presence of smoke particles in the air, indicating the start of a fire. Heat detectors are designed to monitor temperature changes within the bus, triggering alarms if temperatures rise to levels suggestive of fire or overheating. Flame detectors use infrared or ultraviolet sensors to detect the presence of flames, providing an additional layer of detection for rapid fire response.

Furthermore, the integration of fire sensors into bus tracking systems helps ensure compliance with safety regulations and standards for public transportation. By incorporating fire sensors, operators demonstrate their commitment to passenger safety and regulatory adherence. In the event of a fire emergency, these sensors can initiate automatic emergency response protocols, such as activating fire suppression systems, sounding alarms, and notifying emergency services. This proactive approach to fire detection and prevention is crucial for minimizing the risk of injury or loss of life and protecting the bus and its occupants from fire-related incidents.



.

Fig 4.2.2. : Fire Sensor

Fire sensors are essential components of bus tracking systems, contributing significantly to passenger safety, vehicle protection, and regulatory compliance. Their ability to detect and alert to fire hazards in real-time enables swift and effective responses to emergencies, enhancing the overall safety and security of bus operations.

4.2.3. GPS

The GPS module installed in each bus is instrumental in providing accurate and real-time location information. By establishing communication with satellites, this module continually updates the bus's position, allowing for precise tracking at any given moment. This real-time data forms the foundation of the Real-Time Bus Tracking System, enabling passengers to access accurate information regarding bus locations and estimated arrival times. Beyond passenger convenience, transit authorities harness this data to optimize routes, refine schedules, and enhance operational efficiency. The reliability and accuracy offered by GPS technology are paramount in monitoring bus movements effectively, ensuring the seamless functioning of the tracking system.

In addition to improving passenger experience, the GPS technology embedded in buses facilitates comprehensive fleet management for transit authorities. With access to precise location data, transit operators can monitor the entire fleet in real-time, enabling proactive decision-making and resource allocation. By analyzing GPS data, transit authorities can identify traffic patterns, optimize bus deployments, and respond swiftly to disruptions or emergencies. This proactive approach enhances service reliability, minimizes delays, and ultimately contributes to a more efficient and responsive public transportation system. Overall, the GPS technology deployed in bus tracking systems plays a vital role in enhancing both passenger satisfaction and operational effectiveness for transit agencies.

Fig 4.2.3 : GPS

Furthermore, the integration of GPS technology into bus tracking systems enhances overall safety and security measures. By precisely tracking bus movements, transit authorities can promptly respond to incidents or emergencies, such as accidents or breakdowns. Real-time location data enables authorities to dispatch assistance quickly to affected buses, ensuring the safety of passengers and minimizing disruptions to service. Additionally, GPS technology aids in theft prevention and vehicle recovery efforts by providing accurate location information in the event of unauthorized bus usage or theft. This proactive approach to safety and security underscores the invaluable role of GPS technology in modern bus tracking systems, ultimately contributing to safer and more reliable public transportation services.

4.2.4. NodeMCU

NodeMCU, an open-source firmware and development kit based on the ESP8266 Wi-Fi module, is commonly used in hydroponic plant monitoring and control systems due to its versatility and connectivity capabilities. NodeMCU interfaces with various sensors such as pH sensors, EC/TDS sensors, temperature sensors, humidity sensors, and dissolved oxygen sensors to collect data related to the hydroponic environment. For example, it can read pH levels, nutrient concentrations, temperature, humidity, and oxygen levels from respective sensors connected to its GPIO pins or via communication protocols like I2C, SPI, or UART.

It processes the sensor data using its embedded microcontroller (usually an ESP8266 or ESP32 chip). It can perform calculations, data filtering, and normalization to prepare the data for analysis and control decisions. The microcontroller may run custom firmware developed using Arduino IDE, Lua scripting, or other compatible programming languages to implement monitoring and control algorithms.

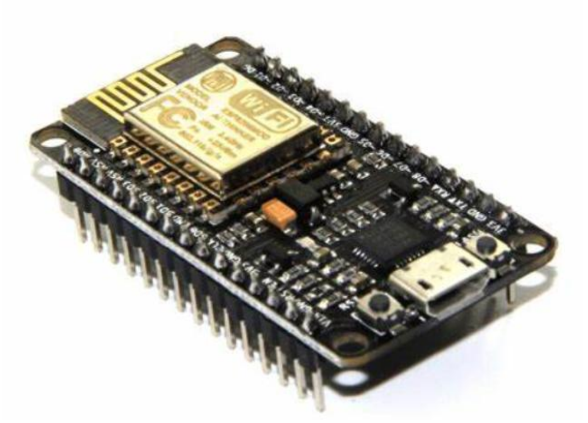


Fig 4.2.4: ESP32

It is equipped with built-in Wi-Fi capabilities, allowing it to connect to local Wi-Fi networks or act as an access point (AP) for remote monitoring and control. It can also establish communication with cloud platforms or local servers through protocols like MQTT (Message Queuing Telemetry Transport), HTTP, or WebSocket, enabling real-time data transmission and remote management. It continuously monitors sensor readings and environmental parameters in the hydroponic system. It checks for deviations from predefined thresholds or setpoints, indicating potential issues such as nutrient imbalances, temperature fluctuations, or humidity variations. Based on the monitored data and control logic programmed into the firmware, NodeMCU can trigger control actions to maintain optimal growing conditions.

For instance, it can activate relays or motor drivers to control water pumps, nutrient dosing pumps, fans, heaters, or lighting systems. NodeMCU can also send notifications, alerts, or data reports to user interfaces such as mobile apps, web dashboards, or email/SMS alerts for real-time monitoring and intervention. NodeMCU can log sensor data to internal memory, external storage (e.g., SD card), or cloud databases for historical analysis, trend identification, and performance optimization. Data analytics algorithms can be implemented to identify patterns, correlations, and anomalies, aiding in decision-making and system improvement. In summary, NodeMCU serves as the central processing and connectivity hub in a hydroponic plant monitoring and controlling system, integrating sensors, processing data, implementing control logic, and facilitating remote monitoring and management for efficient and optimized plant growth.

PIN DIAGRAM OF NODEMCU

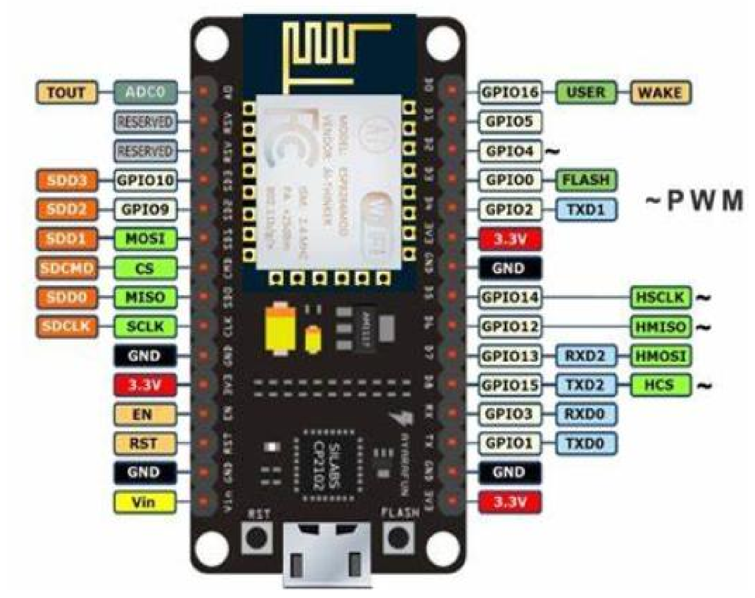


Fig4.2.5: Pin diagram of NodeMCU

The NodeMCU development board typically comes with several pins that serve various functions. Here's a brief overview of the common pins found on a NodeMCU board:

1. Vin: This pin is used to supply power to the NodeMCU board. It can accept a voltage range of 5V to 12V.

2. GND (Ground): These pins are connected to the ground of the circuit and provide a reference voltage for other components.

3. 3V3: This pin provides a regulated 3.3V output, which can be used to power external sensors or modules.

4. TX (Transmit) and RX (Receive): These pins are used for serial communication. TX is used for transmitting data, while RX is used for receiving data.

5. D0 to D8: These pins are digital input/output pins. They can be used for interfacing with digital sensors, controlling LEDs, or other digital devices.

6. A0: This pin is an analog input pin and can be used to read analog signals from sensors such as light sensors or temperature sensors.

7. SCL and SDA: These pins are used for I2C communication. SCL is the clock line, and SDA is the data line. They are used for connecting to I2C-compatible devices such as displays or sensors.

8. SD3, SD2, SD1, and CMD: These pins are used for connecting to an external SD card module for data storage.

9. EN (Enable): This pin is used to enable or disable the NodeMCU module.

10. RST (Reset): This pin is used to reset the NodeMCU module.

4.3. SOFTWARE REQUIREMENTS

4.3.1. ARDUINO IDE

The Arduino Integrated Development Environment (IDE) plays a crucial role in developing, programming, and deploying code for hydroponic plant monitoring and control systems using Arduino-based microcontrollers.

1. Code Development:

Arduino IDE provides a user-friendly interface for writing, editing, and organizing code written in the Arduino programming language (based on C/C++). Developers can create custom firmware to interact with sensors, actuators, and other components in the hydroponic system. This firmware may include functions for data acquisition, processing, control logic, communication protocols, and user interface interactions.

2. Compatibility:

Arduino IDE supports a wide range of Arduino-compatible microcontrollers, including popular models like Arduino Uno, Arduino Nano, Arduino Mega, and ESP8266/ESP32 boards (compatible with NodeMCU). Users can select the appropriate board and port settings within the IDE to compile and upload code to the target microcontroller.

3. Library Integration:

Arduino IDE provides access to a vast library of pre-written code snippets (libraries) that simplify interfacing with sensors, actuators, displays, communication modules, and other peripherals. Users can easily include and utilize these libraries in their projects, saving time and effort in code development.

4. Serial Monitoring:

Arduino IDE includes a Serial Monitor tool that allows real-time communication with the microcontroller via the serial port (USB). This feature is valuable for debugging, testing sensor readings, observing output messages, and monitoring system behavior during runtime.

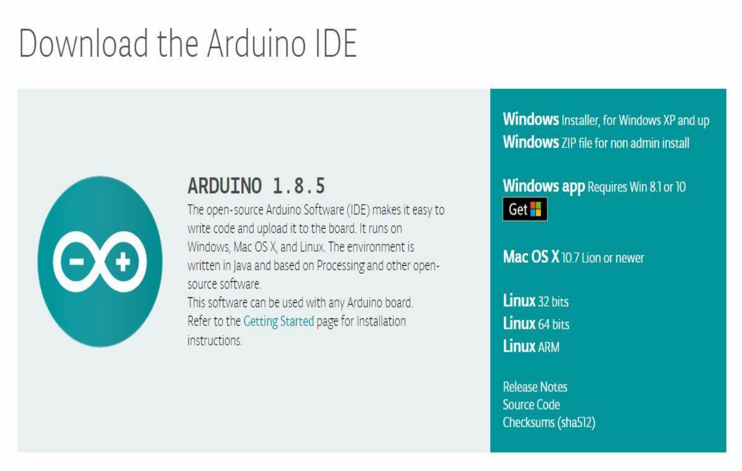
5. Upload and Deployment:

Once the code is written and tested, users can upload it to the target Arduino-compatible microcontroller directly from the Arduino IDE. The IDE handles the compilation process, generates the machine code (hex file), and uploads it to the microcontroller's flash memory for execution. Users can also modify code, recompile, and upload new versions iteratively to implement updates or improvements in the hydroponic system's functionality.

6. Integration with Sensors and Actuators:

Arduino IDE enables seamless integration with various sensors (e.g., pH sensors, EC/TDS sensors, temperature sensors, humidity sensors, dissolved oxygen sensors) and actuators (e.g., pumps, valves, relays, motors) commonly used in hydroponic setups.

STEP 1: DOWNLOAD AND INSTALL THE ARDUINO SOFTWARE IDE:



Download the Arduino IDE

ARDUINO 1.8.5

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

Release Notes
Source Code
Checksums (sha512)

Fig 4.3.1: Downloading and installing the Arduino IDE

STEP 2: GET AN ESP32 ,USB CABLE AND CONNECT THE BOARD

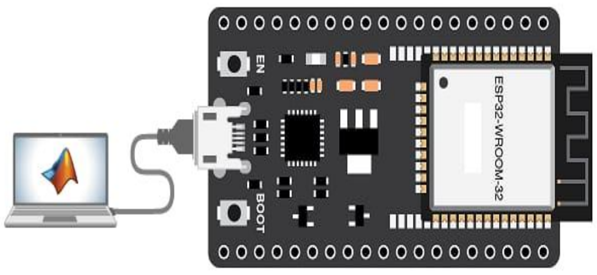
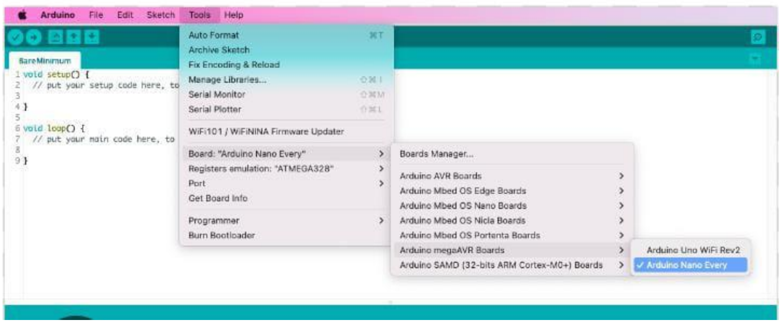


Fig 4.3.2 :Connecting ESP32 and PC through USB Cable

STEP 3: SELECT YOUR BOARD

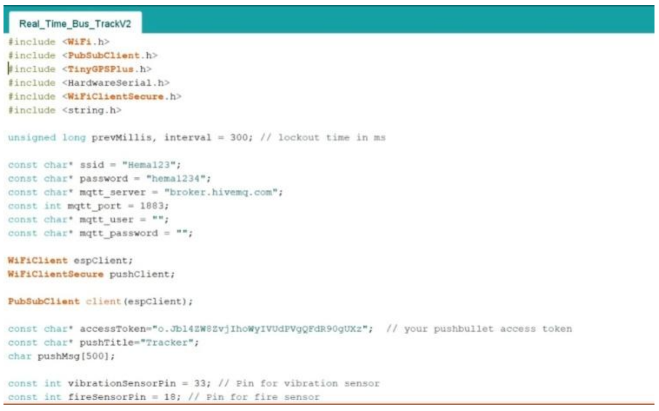


```
void setup() {
  // put your setup code here, to

}

void loop() {
  // put your main code here, to

}
```

Board: "Arduino Nano Every"
Registers emulation: "ATMEGA328"
✓ Arduino Nano Every

Fig 4.3.3 :Select Arduino Board

STEP 4:UPLOAD THE PROGRAM :



```
#include <WiFi.h>
#include <PubSubClient.h>
#include <TinyGPSPlus.h>
#include <HardwareSerial.h>
#include <WiFiClientSecure.h>
#include <string.h>

unsigned long prevMillis, interval = 300; // lockout time in ms

const char* ssid = "Hema123";
const char* password = "hema1234";
const char* mqtt_server = "broker.hivemq.com";
const int mqtt_port = 1883;
const char* mqtt_user = "";
const char* mqtt_password = "";

WiFiClient espClient;
WiFiClientSecure pushClient;

PubSubClient client(espClient);

const char* accessToken="o.Jbl4ZW8ZvjIhoWyIVUdPVgQFdR90gUXz";  // your pushbullet access token
const char* pushTitle="Tracker";
char pushMsg[500];

const int vibrationSensorPin = 33; // Pin for vibration sensor
const int fireSensorPin = 18; // Pin for fire sensor
```

Fig 4.3.4:Upload the Program

STEP 5: RESULT

A few seconds after the upload finishes, yowe can see should see that LED on the board start to blink (in red).

4.3.2. ADDING LIBRARIES:

In a hydroponic plant monitoring and controlling system developed using Arduino-based microcontrollers and the Arduino IDE, adding libraries is a crucial step to simplify interfacing with sensors, actuators, and other peripherals. In this way can add libraries :

1. Library Manager: The Arduino IDE includes a Library Manager tool that allows you to easily search for, install, and manage libraries. To access the Library Manager, open the Arduino IDE, and go to 'Sketch -> Include Library -> Manage Libraries....'.

2. Searching for Libraries:

In the Library Manager window, you can use the search bar to find libraries relevant to your hydroponic system requirements. For example, you can search for libraries related to specific sensors (pH sensors, EC/TDS sensors, temperature sensors, etc.), communication protocols (MQTT, HTTP, etc.), or display modules (LCD, OLED, etc.).

Enter keywords related to the functionality or component you need, and the Library Manager will display relevant libraries.

3. Installing Libraries:

Once you find the desired library in the Library Manager, click on it to view more details such as the library version, author, and a brief description.

To install the library, click the "Install" button next to the library name. The Arduino IDE will download and install the library files to your local Arduino library directory.

4. Including Libraries in Your Sketch:

After installing the required libraries, you can include them in your Arduino sketch (code) to access their functions and features.

To include a library, go to 'Sketch -> Include Library' and select the installed library from the list. Alternatively, you can use '#include <LibraryName.h>' at the beginning of your sketch to include a specific library directly.

5. Using Library Functions:

Once a library is included in your sketch, you can use its predefined functions, constants, and variables to interact with sensors, actuators, displays, or communication modules.Refer to the library's documentation or examples provided by the library author to understand how to use its functions effectively.

6. Compiling and Uploading:

After adding libraries and writing your code, compile the sketch by clicking the checkmark icon in the Arduino IDE toolbar. This step checks for syntax errors and compiles the code into machine code (hex file) suitable for the target microcontroller. If there are no errors, upload the compiled code to the Arduino-compatible microcontroller connected to your computer via USB. Click the right arrow icon in the toolbar to initiate the upload process.

By following these steps, one can effectively add libraries to your hydroponic plant monitoring and controlling system project in the Arduino IDE, making it easier to interface with sensors, actuators, and other components essential for optimal plant growth and management.

4.3.3. Pushbullet Application:

1. Backend System:

- Develop a robust backend system that can track the real-time location of buses using GPS, GSM, or IoT devices installed on each bus.

- Utilize technologies such as MQTT, HTTP APIs, or WebSocket protocols to receive and update bus location data from the buses in real-time.

- Implement a database to store and manage bus routes, schedules, and real-time location data efficiently.

- Ensure that the backend system is scalable, fault-tolerant, and capable of handling a large volume of data and concurrent connections.

2. Integration with Pushbullet:

- Obtain an API key from Pushbullet by signing up for an account on the Pushbullet website.

- Use the Pushbullet API documentation to understand how to authenticate requests and send notifications.

- Develop an integration layer within your backend system that communicates with the Pushbullet API to send notifications to users' devices.

- Implement error handling and retry mechanisms to handle cases where notifications fail to be delivered.

3. User Subscription:

- Create a user interface (web or mobile app) where users can subscribe to receive notifications for specific bus routes, stops, or events.

- Allow users to customize their notification preferences, such as receiving notifications for bus arrivals, delays, or route changes.

- Store user preferences in a database and associate them with unique user identifiers (e.g., user IDs or device tokens).

4. Real-time Updates:

- Implement a real-time data streaming mechanism to continuously update bus location data in your backend system.

- Use algorithms to determine estimated arrival times for buses based on their current locations and historical data.

- Send notifications to subscribed users whenever there are updates or changes in the bus schedules, routes, or locations.

5. Testing and Deployment:

- Conduct thorough testing of the entire system, including unit tests, integration tests, and end-to-end tests, to ensure reliability and correctness.

- Perform load testing to simulate high traffic scenarios and validate system scalability and performance.

- Deploy the system to production using automated deployment pipelines and continuous integration tools.

- Monitor the production environment closely to detect and address any issues that arise in real-time.

4.3.4 Visual Studio Code:

Visual Studio Code (VS Code) is a free, open-source source code editor developed by Microsoft. Launched in 2015, it quickly gained popularity among developers due to its lightweight nature, extensive customization options, and rich features tailored for various programming languages and development workflows.

Key features of Visual Studio Code include:

1. Cross-platform support: VS Code runs on Windows, macOS, and Linux, providing a consistent development experience across different operating systems.

2. Intelligent code editing: VS Code offers features such as syntax highlighting, code completion, and code navigation to boost developer productivity.

3. Built-in terminal: VS Code includes an integrated terminal that allows developers to run command-line tools, scripts, and other tasks without leaving the editor.

4.Extensions: VS Code supports a vast ecosystem of extensions, enabling developers to customize and extend the editor's functionality to suit their specific needs. These extensions cover areas such as language support, debugging, version control, and more.

5.Git integration: VS Code comes with built-in Git integration, allowing developers to perform version control operations such as committing changes, viewing diffs, and pushing/pulling code directly from the editor.

6.Debugging support: VS Code provides debugging capabilities for various programming languages and platforms, including breakpoints, variable inspection, and step-by-step execution.

7. Task automation: VS Code supports task automation through its integrated task runner, enabling developers to define and execute custom tasks for building, testing, and deploying applications.

4.3.5 Folium Map:

Incorporating Folium maps into bus tracking systems offers a user-friendly and accessible solution for continuously monitoring bus movements. With Folium maps, users can easily visualize real-time bus locations and routes directly on their mobile devices, enhancing the overall user experience and convenience. By downloading the Folium map application on mobile devices, passengers and transit authorities alike gain instant access to dynamic maps that display live bus positions, stops, and routes.

Folium maps leverage the power of web-based mapping technologies, allowing for seamless integration with GPS data and other tracking information. This integration enables users to track buses in real-time, receive updates on arrival times, and plan their journeys more efficiently. Additionally, Folium maps provide interactivefeatures such as zooming, panning, and street view, enabling users to explore bus routes and surrounding areas with ease.

CHAPTER-5

RESULTS & APPLICATIONS

5.1 RESULTS:

The Real-Time Bus Tracking and Accident Detection System has shown promising results in enhancing the efficiency, safety, and reliability of public bus transportation. Here are some of the key outcomes observed:

Improved Passenger Experience: Passengers now have access to real-time buslocation information, allowing them to plan their journeys more efficiently and reduce waiting times at bus stops. This has led to increased satisfaction among passengers and a more positive perception of public transportation.

Enhanced Operational Efficiency: Transit authorities have been able to optimize bus routes, improve scheduling, and enhance overall operational efficiency based on the data generated by the tracking system. This has resulted in cost savings and improved resource allocation.

Increased Safety: The integration of the Accident Detection System has significantly improved safety measures. By promptly detecting accidents or irregularities, emergency services can be alerted swiftly, potentially reducing response times and saving lives.

Effective Emergency Response: The automatic alert system ensures that relevant authorities are notified immediately in the event of an accident, enabling them to respond promptly and effectively. This has led to faster emergency response times and better coordination in handling incidents.

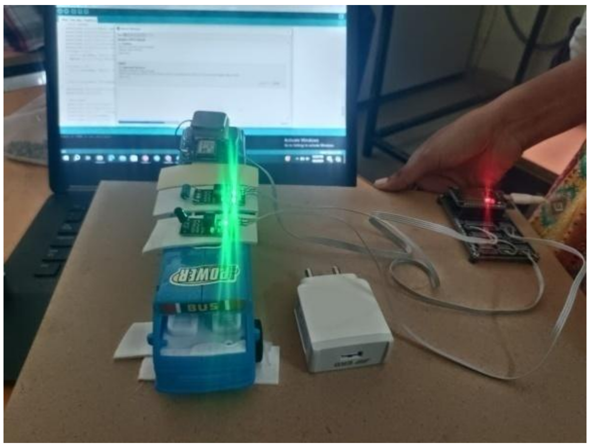
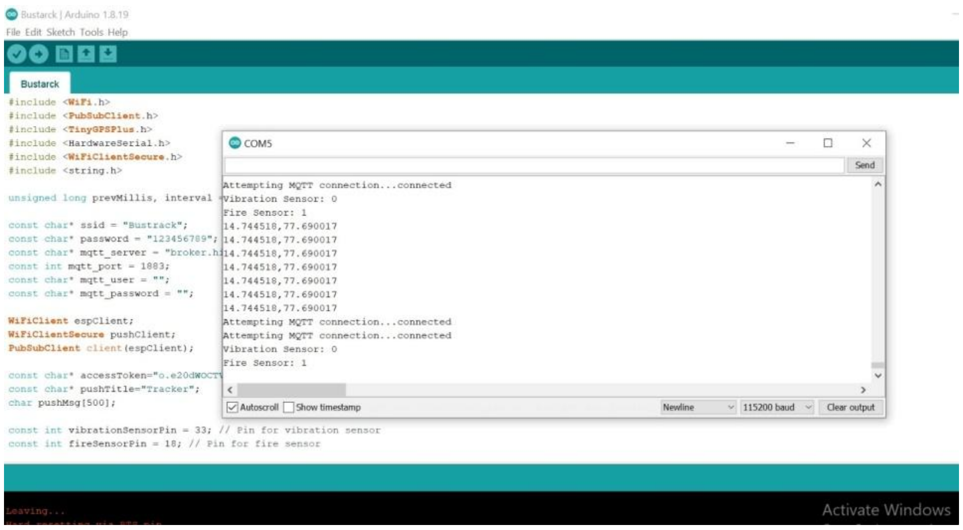


Fig 5.1.1 :Uploading Code to ESP32



```
#include <WiFi.h>
#include <PubSubClient.h>
#include <TinyGPSPlus.h>
#include <HardwareSerial.h>
#include <WiFiClientSecure.h>
#include <string.h>

unsigned long prevMillis, interval

const char* ssid = "Bustrack";
const char* password = "123456789";
const char* mqtt_server = "broker.hi
const int mqtt_port = 1883;
const char* mqtt_user = "";
const char* mqtt_password = "";

WiFiClient espClient;
WiFiClientSecure pushClient;
PubSubClient client(espClient);

const char* accessToken="o.e20dWOCT
const char* pushTitle="Tracker";
char pushMsg[500];

const int vibrationSensorPin = 33; // Pin for vibration sensor
const int fireSensorPin = 18; // Pin for fire sensor
```

```
Attempting MQTT connection...connected
Vibration Sensor: 0
Fire Sensor: 1
14.744518,77.690017
14.744518,77.690017
14.744518,77.690017
14.744518,77.690017
14.744518,77.690017
14.744518,77.690017
14.744518,77.690017
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Vibration Sensor: 0
Fire Sensor: 1
```

[✓ Autoscroll] Show timestamp   Newline   115200 baud   Clear output

```
Leaving...
```

Fig 5.1.2 : Getting longitudes and latitudes in Serial Monitor

Fig 5.1.1 & Fig 5.1.2 illustrates the process of writing code in the Arduino IDE, followed by compiling the code and uploading it to the ESP32 microcontroller. Once the compilation and uploading processes are successfully completed, the longitude and latitude coordinates of the bus are displayed in the Serial Monitor.

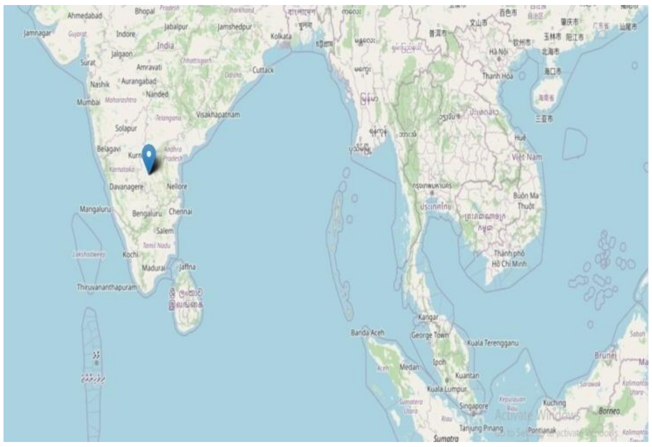
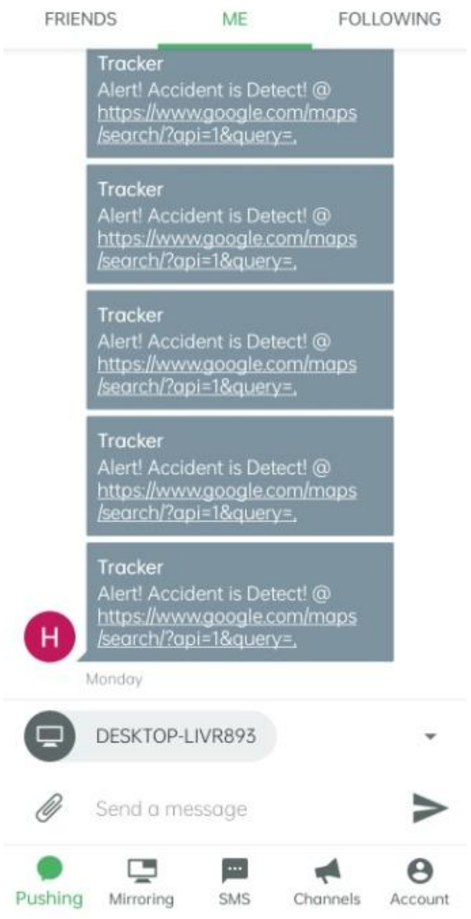


Fig 5.1.3 :Tracking in Folium Map

Figure 5.1.3 illustrates the continuous tracking of the bus on a Folium map. By clicking on the address pointer displayed on the map, users can view the corresponding longitude and latitude coordinates. Notably, the longitude and latitude coordinates displayed on the Folium map and the Serial Monitor are identical.



Tracker
Alert! Accident is Detect! @ https://www.google.com/maps/search/?api=1&query=,

Tracker
Alert! Accident is Detect! @ https://www.google.com/maps/search/?api=1&query=,

Tracker
Alert! Accident is Detect! @ https://www.google.com/maps/search/?api=1&query=,

Tracker
Alert! Accident is Detect! @ https://www.google.com/maps/search/?api=1&query=,

Tracker
Alert! Accident is Detect! @ https://www.google.com/maps/search/?api=1&query=,

Fig 5.1.4 : Notifications of Accidents in Pushbullet App

Figure 5.1.4 illustrates the notification process in the event of fire incidents or accidents. Notifications are sent immediately to the management team to ensure prompt response and action.

5.2 :APPLICATIONS:

1.Performance Analysis: Transit agencies can analyze data collected from bus tracking systems to assess the performance of their services, identify areas for improvement, and make informed decisions about future investments and upgrades.

2. Urban Areas:Bus tracking systems are widely deployed in urban centers with dense populations and extensive public transportation networks. Cities around the world utilize these systems to manage their bus fleets and provide real-time information to passengers.

3.Educational Institutions: Colleges, universities, and large school districts often utilize bus tracking systems to manage their transportation services for students and faculty. These systems help ensure the safety and efficiency of school bus operations.

4. Corporate Campuses: Large corporate campuses and office parks may deploy bus tracking systems to facilitate employee transportation and shuttle services. These systems help employees navigate the campus and reduce reliance on personal vehicles.

5. Rural Areas: While less common, some rural areas may also implement bus tracking systems to provide transit services to residents in remote or underserved communities. These systems help improve access to transportation and connect residents to essential services and amenities and travellers with reliable transportation options.

CONCLUSION AND FUTURE SCOPE

CONCLUSION:

The Real-Time Bus Tracking and Accident Detection System proposed in this project offers a comprehensive solution to address the challenges facing public bus transportation systems. By leveraging GPS, IoT, and sensor technologies, this system enhances the efficiency, safety, and reliability of bus services for both passengers and transit authorities.

Through real-time tracking, passengers gain access to accurate information about bus locations and schedules, empowering them to plan their journeys more effectively and reduce waiting times. Simultaneously, transit authorities benefit from valuable data insights that enable route optimization, scheduling improvements, and overall operational efficiency enhancements.

The integration of an Accident Detection System further enhances safety by promptly identifying and reporting accidents to emergency services and relevant authorities. This swift response capability has the potential to mitigate the severity of accidents and save lives, ensuring the well-being of passengers and drivers alike.

FUTURE SCOPE:

The Real-Time Bus Tracking and Accident Detection System present immense potential for future expansion and refinement. Some avenues for further development include:

Enhanced Predictive Analytics: Implementing advanced data analytics techniques can enable the system to predict potential delays, accidents, or maintenance issues, allowing proactive measures to be taken to mitigate these events.

Integration with Smart City Infrastructure: Integrating the system with broader smart city initiatives can enable seamless coordination with other transportation modes, such as trains, trams, and taxis, to provide passengers with more comprehensive and interconnected mobility solutions.

Augmented Reality Interfaces: Introducing augmented reality interfaces in the passenger application can provide intuitive and immersive navigation guidance, enhancing the user experience and accessibility for diverse passenger demographics.

Autonomous Vehicle Integration: As autonomous vehicle technology evolves, integrating autonomous buses into the system can further improve efficiency, safety, and reliability by eliminating human error and optimizing vehicle operations.

REFERENCES

[1].Wong, Y. D., & Ong, Z. T. (2016). **Real-time GPS based bus tracking and passenger information system**. In 2016 International Conference on Information Networking (ICOIN) (pp. 351-356). IEEE.

[2].Wang, J., Yang, H., & Lu, Y. (2016). **A real-time bus arrival time prediction system based on data mining and mobile services**. IEEE Transactions on Intelligent Transportation Systems, 17(4), 1026-1035.

[3].Zhu, D., Hu, S., & Zhang, C. (2019). **Bus passenger flow prediction method based on the improved SVR model**. Journal of Advanced Transportation, 2019, 1-15.

[4].Sabesan, S., & Rajagopalan, A. N. (2018). **A cloud-based real-time bus monitoring system**. In 2018 2nd International Conference on Intelligent Computing & Control Systems (ICICCS) (pp. 882-886). IEEE.

[5].Ahmed, M. T., & Chowdhury, M. F. (2016). **A novel approach to dynamic bus arrival time prediction using real-time data**. Journal of Advanced Transportation, 50(6), 1660-1675.

[6].Sharma, S., & Garg, R. (2020). **Accident detection and alert system using IoT and GPS**. In 2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC) (pp. 1-5). IEEE.

[7].Fadhil, M. A., Al-Zawi, M. A., & Ameen, W. A. (2016). **Intelligent vehicle accident detection and notification system**. IOP Conference Series: Materials Science and Engineering, 114(1), 012077.

[8].Bansal, M., & Srivastava, S. (2017). **Smart accident detection and reporting system using GPS, GSM and Accelerometer**. In 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence (pp. 612-617). IEEE.

[9].Karpagam, N., & Jayapriya, R. (2020). **IoT-based accident detection and tracking system for vehicles**. In 2020 IEEE International Conference for Innovation in Technology (INOCON) (pp. 1-4). IEEE.

APPENDIX

SOURCE CODE:

Code to be uploaded In Arduino IDE:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <TinyGPSPlus.h>
#include <HardwareSerial.h>
#include <WiFiClientSecure.h>
#include <string.h>

unsigned long prevMillis, interval = 300; // lockout time in ms

const char* ssid = "Bustrack";
const char* password = "123456789";
const char* mqtt_server = "broker.hivemq.com";
const int mqtt_port = 1883;
const char* mqtt_user = "";
const char* mqtt_password = "";

WiFiClient espClient;
WiFiClientSecure pushClient;
PubSubClient client(espClient);

const char* accessToken="o.e20dWOCTVk30TPoesIILXxsjWRcGBXR3";  // your
pushbullet access token
const char* pushTitle="Tracker";
char pushMsg[500];

const int vibrationSensorPin = 33; // Pin for vibration sensor
const int fireSensorPin = 18; // Pin for fire sensor

HardwareSerial ss(2); // Use Serial2 on ESP32
// The TinyGPSPlus object
TinyGPSPlus gps;

const char* lat_topic = "acenaar/lat";
const char* long_topic = "acenaar/long";

String Lat, Lng;

float latitude = 0.0;
float longitude = 0.0;
int count = 1;
// DSTRootCAX3.crt
const char* test_root_ca= \
  "-----BEGIN CERTIFICATE-----\n" \
"MIIDSjCCAjKgAwIBAgIQRK+wgNajJ7qJMDmGLvhAazANBgkqhkiG9w0BAQUFADA/\n" \

"MSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT\n" \

"DkRTVCBSb290IENBIFgzMB4XDTAwMDkzMDIxMTIxOVoXDTIxMDkzMDE0MDExNVow\n" \

"PzEkMCIGA1UEChMbRGlnaXRhbCBTaWduYXR1cmUgVHJ1c3QgQ28uMRcwFQYDVQQD\n" \

"Ew5EU1QgUm9vdCBDQSBYMzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB\n" \

"AN+v6ZdQCINXtMxiZfaQguzH0yxrMMpb7NnDfcdAwRgUi+DoM3ZJKuM/IUmTrE4O\n" \

"rz5Iy2Xu/NMhD2XSKtkyj4zl93ewEnu1lcCJo6m67XMuegwGMoOifooUMM0RoOEq\n" \

"OLl5CjH9UL2AZd+3UWODyOKIYepLYYHsUmu5ouJLGiifSKOeDNoJjj4XLh7dIN9b\n" \

"xiqKqy69cK3FCxolkHRyxXtqqzTWMIn/5WgTe1QLyNau7Fqckh49ZLOMxt+/yUFw\n" \

"7BZy1SbsOFU5Q9D8/RhcQPGX69Wam40dutolucbY38EVAjqr2m7xPi71XAicPNaD\n" \

"aeQQmxkqtilX4+U9m5/wAl0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNV\n" \

"HQ8BAf8EBAMCAQYwHQYDVR0OBBYEFMSnsaR7LHH62+FLkHX/xBVghYkQMA0GCSqG\n" \

"SIb3DQEBBQUAA4IBAQCjGiybFwBcqR7uKGY3Or+Dxz9LwwmglSBd49lZRNI+DT69\n" \

"ikugdB/OEIKcdBodfpga3csTS7MgROSR6cz8faXbauX+5v3gTt23ADq1cEmv8uXr\n" \

"AvHRAosZy5Q6XkjEGB5YGV8eAlrwDPGxrancWYaLbumR9YbK+rlmM6pZW87ipxZz\n" \

"R8srzJmwN0jP41ZL9c8PDHIyh8bwRLtTcm1D9SZImlJnt1ir/md2cXjbDaJWFBM5\n" \
"JDGFoqgCWjBH4d1QB7wCCZAA62RjYJsWvIjJEubSfZGL+T0yjWW06XyxV3b
qxbYo\n" \
 "Ob8VZRzI9neWagqNdwvYkQsEjgfbKbYK7p2CNTUQ\n" \
 "-----END CERTIFICATE-----\n";
const char*  server = "api.pushbullet.com";  // Server URL


void setup_wifi() {
 delay(10);
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);

 WiFi.begin(ssid, password);

 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }

 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
 // Handle incoming messages from the MQTT broker if needed
}

void reconnect() {
 while (!client.connected()) {
   Serial.print("Attempting MQTT connection...");
   if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
     Serial.println("connected");
   } else {
     Serial.print("failed, rc=");
     Serial.print(client.state());
     Serial.println(" try again in 5 seconds");
     delay(5000);
   }
 }
}

void setup() {
 Serial.begin(115200);
 ss.begin(9600, SERIAL_8N1, 16, 17);
 pinMode(vibrationSensorPin, INPUT_PULLUP); // Set vibration sensor pin as input
 pinMode(fireSensorPin, INPUT); // Set fire sensor pin as input
```

```
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
  pushClient.setCACert(test_root_ca);
  Serial.println("\nStarting connection to server...");
  if (!pushClient.connect(server, 443))
  {
    Serial.println("Connection failed!");
  }
  else
  {
    Serial.println("Connected to server!");
  }
}

void loop() {

  while (ss.available() > 0)
    if (gps.encode(ss.read()))
      displayInfo();
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  ////while (ss.available() > 0)
   // if (gps.encode(ss.read()))
     // displayInfo();

   int vibrationValue = digitalRead(vibrationSensorPin); // Read vibration sensor value
   int fireValue = digitalRead(fireSensorPin); // Read fire sensor value
   if (millis() - prevMillis > interval) {
    if (digitalRead(vibrationSensorPin)) { // if LOW
      Serial.println(count);
      prevMillis = millis(); // reset
      count++;
    }
  }
  if(count >= 3){
    count = 0;
    Serial.println("accident Detcedt");
    sendToPushBullet(1);
    delay(5000);
  }
  if(fireValue == 0){
    Serial.println("Fire Detected");
    sendToPushBullet(2);
    delay(5000);
  }
```

```
 // Print sensor readings to serial monitor
 Serial.print("Vibration Sensor: ");
 Serial.println(vibrationValue);
 Serial.print("Fire Sensor: ");
 Serial.println(fireValue);
 char message[100];
 if (gps.location.isValid()) {
   sprintf(message, "%0.6f, %0.6f", gps.location.lat(), gps.location.lng());
   client.publish(lat_topic, message);
   delay(1000);
 }
 delay(300);
}
void displayInfo()
{
 if (gps.location.isValid())
 {
   Lat = String(gps.location.lat(), 6);
   Lng = String(gps.location.lng(), 6);
   Serial.print(gps.location.lat(), 6);
   Serial.print(F(","));
   Serial.print(gps.location.lng(), 6);
 }
 else
 {
   Serial.print(F("INVALID"));
 }
 Serial.println();
}

void sendToPushBullet(int line){
 Serial.println("\nStarting connection to server...");
 if (!pushClient.connect(server, 443))
 {
   Serial.println("Connection failed!");
 }
 else
 {
   Serial.println("Connected to server!");

   //HTTP REQUEST
   pushClient.println("POST /v2/pushes HTTP/1.0");

   //HTTP HEADER
   pushClient.println("Host: api.pushbullet.com");
   pushClient.println("Content-Type: application/json");
   pushClient.println("Connection: close");
   pushClient.print("Access-Token:");
   pushClient.println(accessToken);
   pushClient.print("Content-Length: ");
```

```
  if(line == 1){
    sprintf(pushMsg, "Alert! Accident is Detect! @ https://www.google.com/maps/search/?api=1&query=%s,%s", Lat, Lng);
    Serial.println("Sent Message Accident");
  }
  if(line == 2){
    sprintf(pushMsg, "Alert! Fire is Detect! @ https://www.google.com/maps/search/?api=1&query=%s,%s", Lat, Lng);
  }

  //sprintf(pushMsg, "%s\nhttps://www.google.com/maps/search/?api=1&query=%s,%s", "Vehicle is stopped!", Lat, Lon);

  pushClient.println(41+strlen(pushTitle)+strlen(pushMsg));
  pushClient.println();

  //HTTP BODY
  pushClient.print("{\"type\": \"note\", \"title\": \"");
  pushClient.print(pushTitle);
  pushClient.print("\", \"body\": \"");
  pushClient.print(pushMsg);
  pushClient.println("\"}");

  while (pushClient.connected()) {
    String line = pushClient.readStringUntil('\n');
    if (line == "\r") {
      Serial.println("headers received");
      break;
    }
  }
  // if there are incoming bytes available
  // from the server, read them and print them:
  while (pushClient.available())
  {
    char c = pushClient.read();
    Serial.write(c);
  }

  pushClient.stop();
 }
}
```

Code to be uploaded In Visual Studio Code:

```
import time

import folium

import paho.mqtt.client as mqtt

# Initialize Folium map

folium_map = folium.Map(location=[15.834536, 78.029366], zoom_start=10)

# Save the map to HTML file

folium_map.save('map.html')

def on_connect(client, userdata, flags, rc):

    print("Connected to MQTT broker with result code "+str(rc))

    client.subscribe("acenaar/lat")

    client.subscribe("acenaar/lng")  # Corrected topic name

def on_message(client, userdata, msg):

    global latitude, longitude

    print("Message received:", msg.topic, msg.payload)

    message = msg.payload.decode('utf-8').split(",")

    latitude = float(message[0])

    longitude = float(message[1])

    update_map(latitude, longitude)

def update_map(latitude, longitude):

    if latitude is not None and longitude is not None:

        # Add a marker for the new coordinates

        folium.Marker([latitude, longitude], popup=f'Lat: {latitude}, Long: {longitude}').add_to(folium_map)

        print("Map is updated")

        # Save the updated map
        folium_map.save('map.html')
if __name__ == "__main__":
    latitude = None
    longitude = None
    # Initialize MQTT client
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect("broker.hivemq.com", 1883, 60)
    # Run MQTT client loop in a separate thread
    client.loop_start()
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        print("Exiting...")
        client.disconnect()
```

Development of Real Time Bus Tracking and Accident Detection System

Y.Raja Kullayi Reddy,M.Tech,
Assistant Professor
ElectronicsandCommunication Engineering
SrinivasaRamanujanInstituteof Engineering
Anantapur,India.
rajakullayi.ece@srit.ac.in

Hemalatha K
Electronics and Communication EngineeringSrinivasa Ramanujan Instituteof Technology
Anantapur,India.
204g1a0441@srit.ac.in

Deepthi Reddy S
Electronics and Communication EngineeringSrinivasa Ramanujan Instituteof Technology
Anantapur,India.
214g5a0403@srit.ac.in

Kavya T
ElectronicsandCommunication Engineering
SrinivasaRamanujanInstituteof Technology
Anantapur,India.
204g1a0454@srit.ac.in

Divija C
Electronics and Communication Engineering
SrinivasaRamanujanInstitute of Technology
Anantapur,India.
204g1a0425@srit.ac.in

***Abstract*— In today's bustling urban landscapes, buses serve as essential arteries of transportation, linking diverse sectors including educational institutions, businesses, and public organizations. However, the escalating challenge of traffic congestion spurred by burgeoning urban populations demands innovative solutions for enhancing both efficiency and safety in bus operations. A groundbreaking response emerges in the form of a real-time bus tracking system, integrating cutting-edge GSM and GPS technologies to navigate the complexities of modern urban mobility. By equipping buses with GSM trackers for real-time location updates and GPS receivers for precise positioning data, passengers gain the ability to monitor bus movements seamlessly, while transit authorities leverage invaluable insights for fleet management and operational optimization. Complementing these capabilities are strategically deployed tilt sensors, providing nuanced insights into bus orientation and movement patterns, alongside fire sensors for swift detection of onboard hazards. Further enhancing system robustness are ESP32MCU for WiFi connectivity and regulators for efficient power management, solidifying the system's reliability and efficacy.Through this integrated approach, the real-time bus tracking system not only addresses the immediate challenges of urban congestion but also heralds a transformative shift towards smarter, safer, and more efficient urban transportation networks.**

Keywords—GSM,GPS,Fire Sensor,route optimization,Vibration Sensor, emergency response, data analysis, traffic patterns.

I.Introduction

In today's fast-paced world, efficient transportation systems are essential for the smooth functioning of various organizations, ranging from educational institutions to corporate firms. However, the rapid growth in population and vehicle numbers has resulted in escalating traffic congestion, posing significant challenges to effective transportation management. To address these issues, the development of a real-time bus tracking system utilizing GSM technology has emerged as a promising solution. This project aims to revolutionize the way buses are managed and operated by providing real-time updates on their locations, enhancing efficiency, and improving passenger safety.

With buses serving as a primary mode of transportation for many sectors, including schools, universities, businesses, and government organizations, the need for a reliable tracking system has never been greater. By incorporating GSM trackers onto buses, this system enables administrators and passengers to monitor their current locations accurately. Moreover, the integration of tilt sensors offers additional insights into bus movements by detecting changes in orientation, enhancing the system's functionality beyond mere location tracking. Furthermore, the inclusion of fire sensors adds a layer of safety by promptly detecting any signs of fire or excessive heat onboard, ensuring the well-being of passengers and personnel.

Moreover, the adoption of ESP32 MCU for WiFi connectivity and regulators fortifies the system's capabilities, facilitating seamless communication and judicious power management. Through the amalgamation of these cutting-edge technologies, the envisioned real-time bus tracking system aspires not only to mitigate traffic congestion through route optimization but also to prioritize the safety and comfort of passengers. This project heralds a transformative stride in the modernization of urban transportation infrastructure, charting a course towards smarter, more efficient, and safer bus operations amidst the burgeoning urban landscape.

II. LITERATURE SURVEY

2.1. The paper titled "A Real-Time GPRS Vehicle Tracking System Displayed on a Google-Map-Based Website" authored by Prawat Chaiprapa, Supaporn Kiattisin, and Adisorn Leelasantitham presents a literature survey on the development of a real-time vehicle tracking system utilizing GPS technology and GPRS modules. The system enables the tracking of vehicles by receiving location data and transmitting it to a website for display on a Google Map interface. The proposed system utilizes PHP and AJAX programming languages to develop the Google Map API, facilitating the construction of the map interface on the website. This paper provides insights into the technical aspects ofcorporate firms. However, the rapid growth in population and vehicle numbers has resulted in escalating traffic congestion, posing significant challenges to effective transportation management. To address these issues, the development of a real-time bus tracking system utilizing GSM technology has emerged as a promising solution. This project aims to revolutionize the way buses are managed and operated by providing real-time updates on their locations, enhancing efficiency, and improving passenger safety.

With buses serving as a primary mode of transportation for many sectors, including schools, universities, businesses, and government organizations, the need for a reliable tracking system has never been .

By incorporating GSM trackers onto buses, this system enables administrators and passengers to monitor their current locations accurately. Moreover, the integration of tilt sensors offers additional insights into bus movements by detecting changes in orientation, enhancing the system's functionality beyond mere location tracking. Furthermore, the inclusion of fire sensors adds a layer of safety by promptly detecting any signs of fire or excessive heat onboard, ensuring the well-being of passengers and personnel

.

Moreover, the adoption of ESP32 MCU for WiFi connectivity and regulators fortifies the system's capabilities, facilitating seamless communication and judicious power implementing a real-time vehicle tracking system, focusing on the integration of GPS and GPRS technologies with web-based mapping platforms for efficient monitoring and management of vehicle fleets.

2.2. The paper titled "Low-Cost Campus Bus Tracker using WiFi Access Points" authored by KuanYew Tan and KokSheik Wong in May 2016 provides a comprehensive literature review on a low-cost approach to implementing a campus bus tracking system utilizing WiFi access points. In contrast to traditional GPS-based tracking systems, this paper explores the use of WiFi infrastructure for location tracking, taking advantage of the widespread availability of WiFi internet access points. The proposed system utilizes two main techniques for tracking: proximity estimation, which predicts the location of an object as it approaches known WiFi access points, and scene analysis, an image processing-based method that compares current scene images with a database of tagged scene images to determine location. By leveraging existing WiFi infrastructure and innovative tracking techniques, this paper offers insights into developing cost-effective and efficient bus tracking solutions tailored for campus environments.While our project focuses on implementing a real-time bus tracking system using GSM technology and incorporates tilt sensors and fire sensors for enhanced safety, the literature survey by Tan and Wong explores a low-cost approach to bus tracking using WiFi access points. Unlike our project, which relies on GPS modules for accurate location tracking and GPRS for data transmission to a website, their study investigates the feasibility of utilizing existing WiFi infrastructure for proximity-based tracking and scene. While both projects aim to improve bus

III EXISTING SYSTEM

However, relying on WhatsApp for sharing bus locations also comes with its limitations and disadvantages. Firstly, WhatsApp is primarily designed as a messaging platform rather than a dedicated tracking system. As a result, it may lack certain features and functionalities that are essential for efficient bus tracking, such as real-time updates, accurate location data, and advanced route optimization. This can lead to inconsistencies in the information provided to passengers and may result in frustration or confusion, particularly during peak travel times or in areas with complex transit networks.

Additionally, the use of WhatsApp for sharing bus locations may pose privacy and security concerns for both passengers and operators. Since WhatsApp is a third-party platform, operators have limited control over the privacy settings and security measures implemented within the messaging service. This raises concerns about the protection of sensitive passenger data, such as location information and personal contact details, from unauthorized access or misuse. Moreover, relying on WhatsApp for critical communication may expose operators to risks such as service outages, data breaches, or disruptions in service, which could compromise the reliability and effectiveness of the bus tracking system in providing timely and accurate information to passengers.

IV PROPOSED WORK

The Real-Time Bus Tracking and Accident Detection System consists of two main components: the Real-Time Bus Tracking system and the Accident Detection system.

Real-Time Bus Tracking System:

1.Utilizes GPS and IoT technologies to monitor the location and movement of buses in real-time.

2.Allows passengers to access bus location information through a mobile application.

3.Enable passengers to plan their journeys more effectively and reduce waiting times at bus stops.

4.Provide transit authorities with data to optimize bus routes, improve scheduling, and enhance operational efficiency.

5.For Continuously tracking we are using Folium Map.

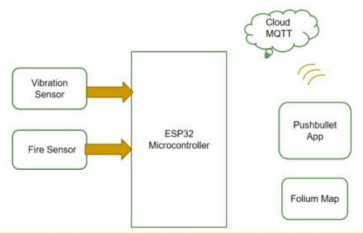
Accident Detection System:

1.Integrated into buses, utilizing sensors such as Vibration sensor and Fire sensor.

2.Monitors the vehicle's behavior and surroundings continuously.

3.Detects irregularities or collisions that may indicate an accident.

4.Automatically sends alerts to emergency services and relevant authorities in the event of an accident to Pushbullet Application



.

Fig.Block diagram of Proposed System.

III. SYSTEM DESIGN

The system integrates WiFi access points for location tracking, tilt sensors for monitoring bus orientation, and fire sensors for early detection of onboard incidents, ensuring enhanced safety and efficiency in campus bus operations.

3.1. SENSORS

Vibration sensor, also known as a vibration detector or accelerometer, is a device designed to detect and measure vibrations or oscillations in an object or system. These sensors can detect various types of vibrations, including mechanical vibrations caused by machinery or vehicles, seismic vibrations caused by earthquakes, and even human-induced vibrations. Vibration sensors typically consist of a sensing element, such as a piezoelectric crystal or microelectromechanical system (MEMS), which generates an electrical signal in response to mechanical movement or acceleration.

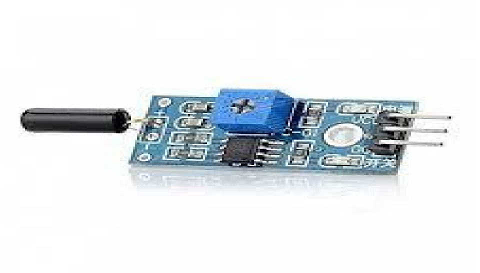


Fig1: Vibration Sensor

Fire sensors, also referred to as heat detectors or smoke detectors, are devices designed to detect the presence of fire or excessive heat in their vicinity. They function by monitoring environmental changes, such as temperature fluctuations or the presence of smoke particles, and triggering an alarm or alert system when these indicators surpass predefined thresholds. Fire sensors are crucial components of fire detection and prevention systems, providing early warning of potential fire hazards and enabling swift evacuation procedures and firefighting responses. By promptly detecting fires or heat anomalies onboard vehicles or within buildings, fire sensors play a vital role in safeguarding lives and property, reducing the risk of injuries and mitigating the impact of fire-related incidents.

Fig2: Fire Sensor

3.2. MICROCONTROLLER

The ESP32 MCU (Microcontroller Unit) is a powerful and versatile microcontroller developed by Espressif Systems. It is part of the ESP32 series of chips, which integrate Wi-Fi and Bluetooth connectivity along with a dual-core processor, making it ideal for a wide range of IoT (Internet of Things) applications. The ESP32 MCU offers a rich set of features, including GPIO (General Purpose Input/Output) pins, ADC (Analog-to-Digital Converter), DAC (Digital-to-Analog Converter), SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), UART (Universal Asynchronous Receiver-Transmitter), and PWM (Pulse Width Modulation) capabilities. Its low power consumption, high processing power, and extensive connectivity options make it suitable for various applications, including wireless communication, sensor data acquisition, home automation, and industrial control systems. Additionally, the ESP32 MCU is supported by a robust ecosystem of development tools, libraries, and community resources, making it accessible and easy to use for developers and hobbyist alike.

Fig3:ESP32 Module

3.3 GPS

GPS, short for Global Positioning System, is a satellite-based navigation system that provides location and time information anywhere on Earth where there is an unobstructed line of sight to four or more GPS satellites. Developed by the United States Department of Defense, GPS consists of a network of at least 24 satellites orbiting the Earth, broadcasting precise timing signals. GPS receivers, such as those found in smartphones, cars, and specialized tracking devices, use these signals to calculate their exact position, velocity, and time. GPS technology has a wide range of applications, including navigation for vehicles, aircraft, and ships, location-based services, mapping, surveying, and outdoor recreation. It has revolutionized the way people navigate and interact with the world around them, enabling precise positioning information to be accessible to anyone, anywhere, at any time.

Fig:GPS

3.4 Pushbullet App

1. **Backend System:**

- Develop a robust backend system that can track the real-time location of buses using GPS, GSM, or IoT devices installed on each bus.
- Utilize technologies such as MQTT, HTTP APIs, or WebSocket protocols to receive and update bus location data from the buses in real-time
- Implement a database to store and manage bus routes, schedules, and real-time location data efficiently

- Ensure that the backend system is scalable, fault-tolerant, and capable of handling a large volume of data and concurrent connections.

2. Integration with Pushbullet:

- Obtain an API key from Pushbullet by signing up for an account on the Pushbullet website.
- Use the Pushbullet API documentation to understand how to authenticate requests and send notifications.

3.5 Visual Studio Code

Visual Studio Code (VS Code) is a free, open-source source code editor developed by Microsoft. Launched in 2015, it quickly gained popularity among developers due to its lightweight nature, extensive customization options, and rich features tailored for various programming languages and development workflows.

Key features of Visual Studio Code include:

1. Cross-platform support: VS Code runs on Windows, macOS, and Linux, providing a consistent development experience across different operating systems.

2. Intelligent code editing: VS Code offers features such as syntax highlighting, code completion, and code navigation to boost developer productivity.

3.6 Folium Map

Incorporating Folium maps into bus tracking systems offers a user-friendly and accessible solution for continuously monitoring bus movements. With Folium maps, users can easily visualize real-time bus locations and routes directly on their mobile devices, enhancing the overall user experience and convenience. By downloading the Folium map application on mobile devices, passengers and transit authorities alike gain instant access to dynamic maps that display live bus positions, stops, and routes.

Folium maps leverage the power of web-based mapping technologies, allowing for seamless integration with GPS data and other tracking information. This integration enables users to track buses in real-time, receive updates on arrival times, and plan their journeys more efficiently. Additionally, Folium maps provide interactivefeatures such as zooming, panning, and street view, enabling users to explore bus routes and surrounding areas with ease.

IV.SYSTEM IMPLEMENTATION

The implementation of the bus tracking system involves several key steps to ensure the seamless operation of the system and the integration of various hardware and software components. Firstly, the hardware components, including GSM modules, GPS receivers, and sensors, are installed on each bus according to the system's specifications. These components are carefully integrated into the bus's existing infrastructure to minimize disruption to its operations.

Next, the software components, including the central server software and client applications, are developed and deployed. The central server software is responsible for receiving, processing, and storing location data from the buses, while the client applications provide users with access to real-time bus location information. Both components are rigorously tested to ensure their reliability, security, and scalability.

Once the hardware and software components are in place, the system undergoes thorough testing to verify its functionality and performance. This testing phase includes simulating various scenarios to assess the system's response under different conditions, such as poor network connectivity or sensor malfunctions. Any issues or bugs identified during testing are addressed promptly, and the system is refined iteratively until it meets the desired specifications.

Finally, the bus tracking system is deployed across the entire fleet of buses, and training is provided to operators and administrators on how to use the system effectively. Continuous monitoring and maintenance are carried out to ensure the system's ongoing reliability and performance, with updates and enhancements implemented as needed to address evolving requirements and technological advancements.

Overall, the implementation of the bus tracking system involves a coordinated effort to integrate hardware and software components, rigorously test the system's functionality, and deploy it across the entire bus fleet. Through careful planning and execution, the system enhances the efficiency, reliability, and safety of public transportation operations, ultimately improving the overall passenger experience.

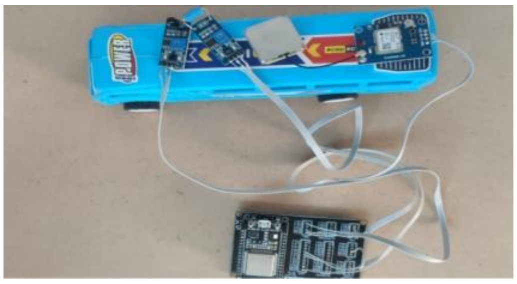


Fig 1: General Architecture

V. RESULTS AND DISCUSSIONS

RESULTS

The Real-Time Bus Tracking and Accident Detection System has shown promising results in enhancing the efficiency, safety, and reliability of public bus transportation. Here are some of the key outcomes observed:

Improved Passenger Experience: Passengers now have access to real-time buslocation information, allowing them to plan their journeys more efficiently and reduce waiting times at bus stops. This has led to increased satisfaction among passengers and a more positive perception of public transportation.

Enhanced Operational Efficiency: Transit authorities have been able to optimize bus routes, improve scheduling, and enhance overall operational efficiency based on the data generated by the tracking system. This has resulted in cost savings and improved resource allocation.

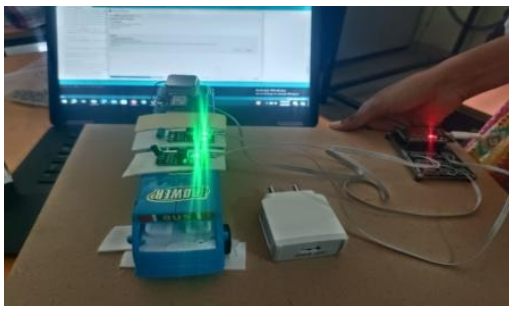
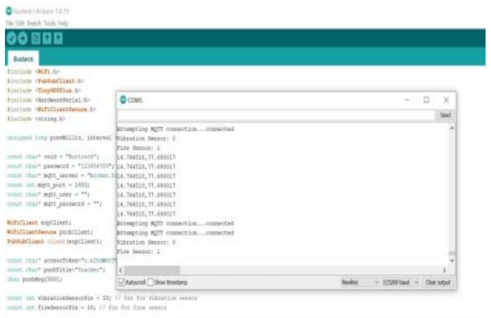


Fig 1:Uploading Code to ESP32



```
#include <WiFi.h>
#include <PubSubClient.h>
#include <TinyGPSPlus.h>
#include <HardwareSerial.h>
#include <WiFiClientSecure.h>
#include <string.h>

unsigned long prevMillis, interval

const char* ssid = "Bustrack";
const char* password = "123456789";
const char* mqtt_server = "broker.
const int mqtt_port = 1883;
const char* mqtt_user = "";
const char* mqtt_password = "";

WiFiClient espClient;
WiFiClientSecure pushClient;
PubSubClient client(espClient);

const char* accessToken="o.e20dW
const char* pushTitle="Tracker";
char pushMsg[500];

const int vibrationSensorPin = 33; // Pin for vibration sensor
const int fireSensorPin = 18; // Pin for fire sensor
```

```
Attempting MQTT connection...connected
Vibration Sensor: 0
Fire Sensor: 1
14.744518,77.690017
14.744518,77.690017
14.744518,77.690017
14.744518,77.690017
14.744518,77.690017
14.744518,77.690017
14.744518,77.690017
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Vibration Sensor: 0
Fire Sensor: 1
```

Fig 2:Getting longitudes and latitudes in Serial Monitor

Fig 1. & Fig 2 illustrates the process of writing code in the Arduino IDE, followed by compiling the code and uploading it to the ESP32 microcontroller. Once the compilation and uploading processes are successfully completed, the longitude and latitude coordinates of the bus are displayed in the Serial Monitor.

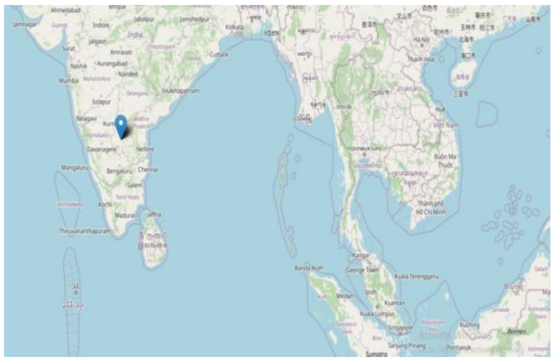


Figure 3 illustrates the continuous tracking of the bus on a Folium map. By clicking on the address pointer displayed on the map, users can view the corresponding longitude and latitude coordinates. Notably, the longitude and latitude coordinates displayed on the Folium map and the Serial Monitor are identical.

CONCLUSION

In conclusion, the development of a bus tracking and accident detection system marks a significant stride towards addressing the evolving challenges in urban transportation. By leveraging advanced technologies like GSM tracking, GPS positioning, and a myriad of sensors, this system offers comprehensive real-time monitoring of bus fleets, enhancing operational efficiency and passenger safety. The integration of ESP32 MCU for WiFi connectivity and regulators ensures seamless communication and efficient power management, further optimizing system performance. Beyond its technical prowess, this project underscores a commitment to innovation and sustainability in urban mobility. With its ability to mitigate traffic congestion, improve route planning, and enhance emergency response capabilities, the bus tracking and accident detection system represents a pivotal advancement in shaping smarter, more resilient cities for the future.

REFERENCES

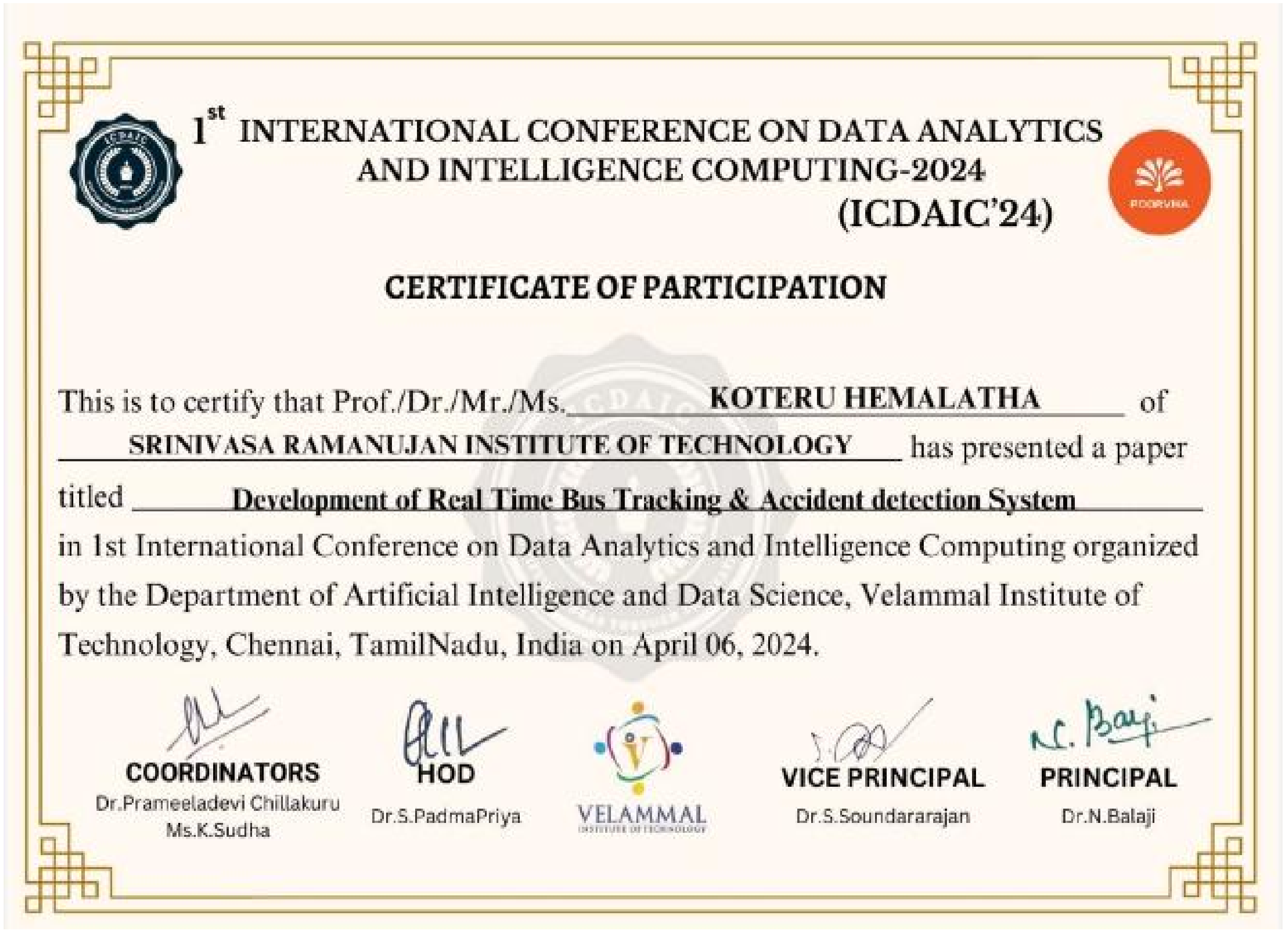
[1] Prawat Chaiprapa, Supaporn Kiattisin and Adisorn Leelasantitham, "A Real-Time GPRS Vehicle Tracking System Displayed on a Google-Map-Based Website", 2011.

[2] P. Zhou, Y. Zheng, and M. Li, "How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing", 2014.

[3] KuanYew Tan, and KokSheik Wong, "ILow-Cost Campus Bus Tracker using WiFi Access Points", 2016.

[4] N. Chadil, A. Russameesawang, and P. Keeratiwintakor, "Real-Time Tracking Management System Using GPS and Google Earth", 2018.

INTERNATIONAL CONFERENCE CERTIFICATES



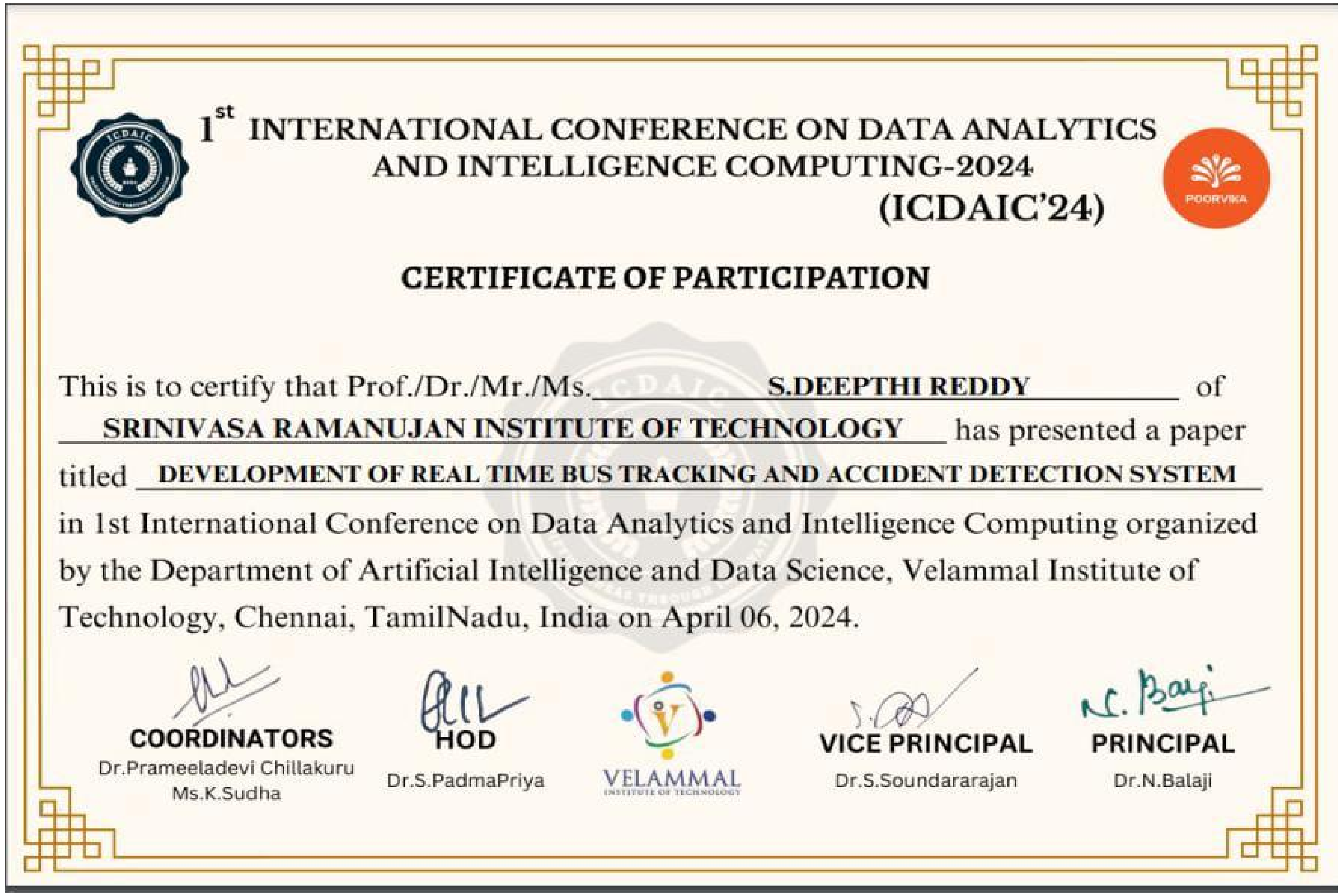
1st INTERNATIONAL CONFERENCE ON DATA ANALYTICS AND INTELLIGENCE COMPUTING-2024 (ICDAIC'24)

CERTIFICATE OF PARTICIPATION

This is to certify that Prof./Dr./Mr./Ms. **KOTERU HEMALATHA** of **SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY** has presented a paper titled **Development of Real Time Bus Tracking & Accident detection System** in 1st International Conference on Data Analytics and Intelligence Computing organized by the Department of Artificial Intelligence and Data Science, Velammal Institute of Technology, Chennai, TamilNadu, India on April 06, 2024.

[illegible]	[illegible]	[illegible]	N.C. Balaji
COORDINATORS	**HOD**	**VICE PRINCIPAL**	**PRINCIPAL**
Dr.Prameeladevi Chillakuru Ms.K.Sudha	Dr.S.PadmaPriya	Dr.S.Soundararajan	Dr.N.Balaji

VELAMMAL
INSTITUTE OF TECHNOLOGY



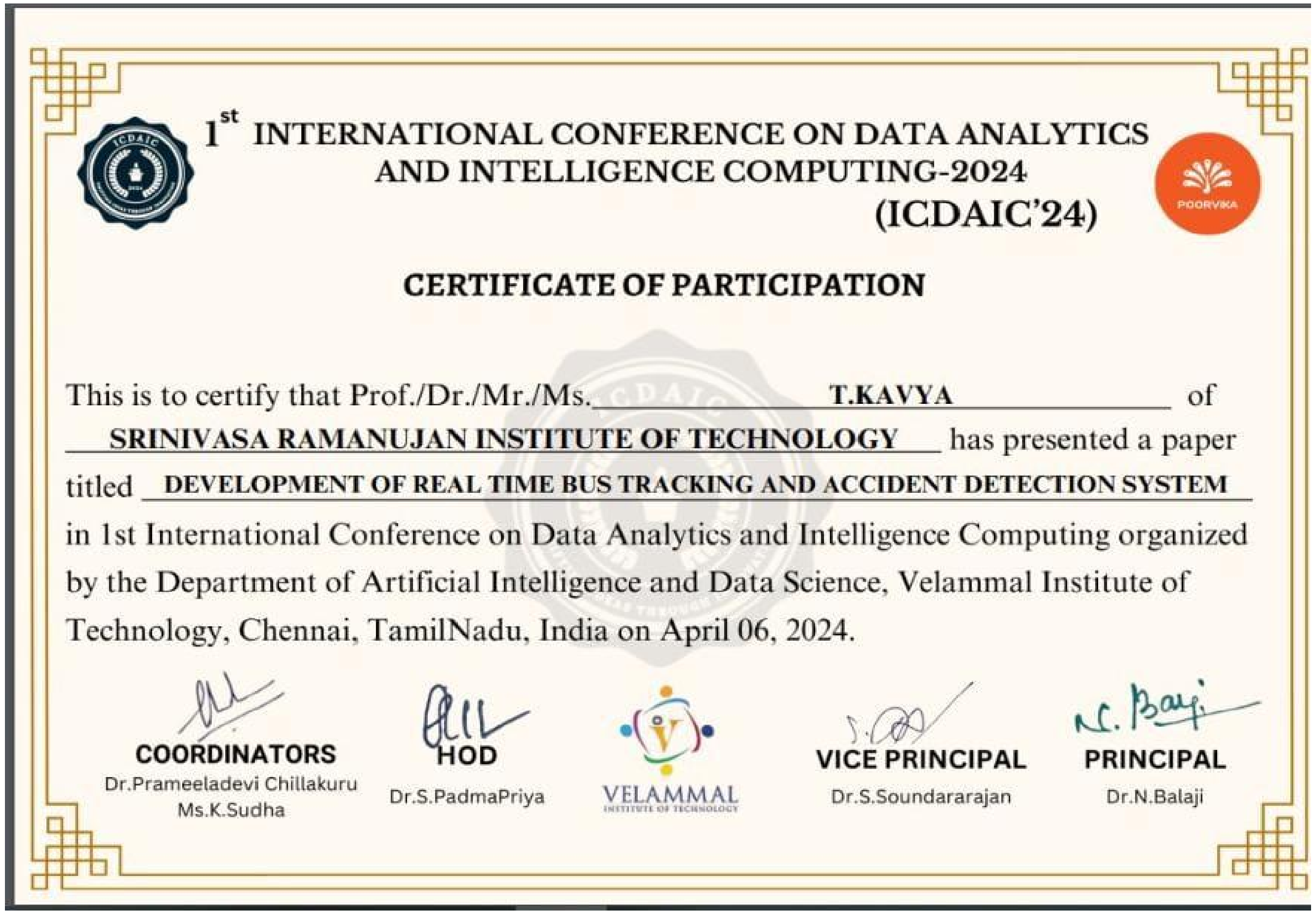
1st INTERNATIONAL CONFERENCE ON DATA ANALYTICS AND INTELLIGENCE COMPUTING-2024
(ICDAIC'24)

POORVIKA

CERTIFICATE OF PARTICIPATION

This is to certify that Prof./Dr./Mr./Ms. **S.DEEPTHI REDDY** of **SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY** has presented a paper titled **DEVELOPMENT OF REAL TIME BUS TRACKING AND ACCIDENT DETECTION SYSTEM** in 1st International Conference on Data Analytics and Intelligence Computing organized by the Department of Artificial Intelligence and Data Science, Velammal Institute of Technology, Chennai, TamilNadu, India on April 06, 2024.

[illegible]	[illegible]		[illegible]	[illegible]
COORDINATORS	**HOD**	VELAMMAL INSTITUTE OF TECHNOLOGY	**VICE PRINCIPAL**	**PRINCIPAL**
Dr.Prameeladevi Chillakuru Ms.K.Sudha	Dr.S.PadmaPriya		Dr.S.Soundararajan	Dr.N.Balaji



1st INTERNATIONAL CONFERENCE ON DATA ANALYTICS AND INTELLIGENCE COMPUTING-2024 (ICDAIC'24)

CERTIFICATE OF PARTICIPATION

This is to certify that Prof./Dr./Mr./Ms. **T.KAVYA** of **SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY** has presented a paper titled **DEVELOPMENT OF REAL TIME BUS TRACKING AND ACCIDENT DETECTION SYSTEM** in 1st International Conference on Data Analytics and Intelligence Computing organized by the Department of Artificial Intelligence and Data Science, Velammal Institute of Technology, Chennai, TamilNadu, India on April 06, 2024.

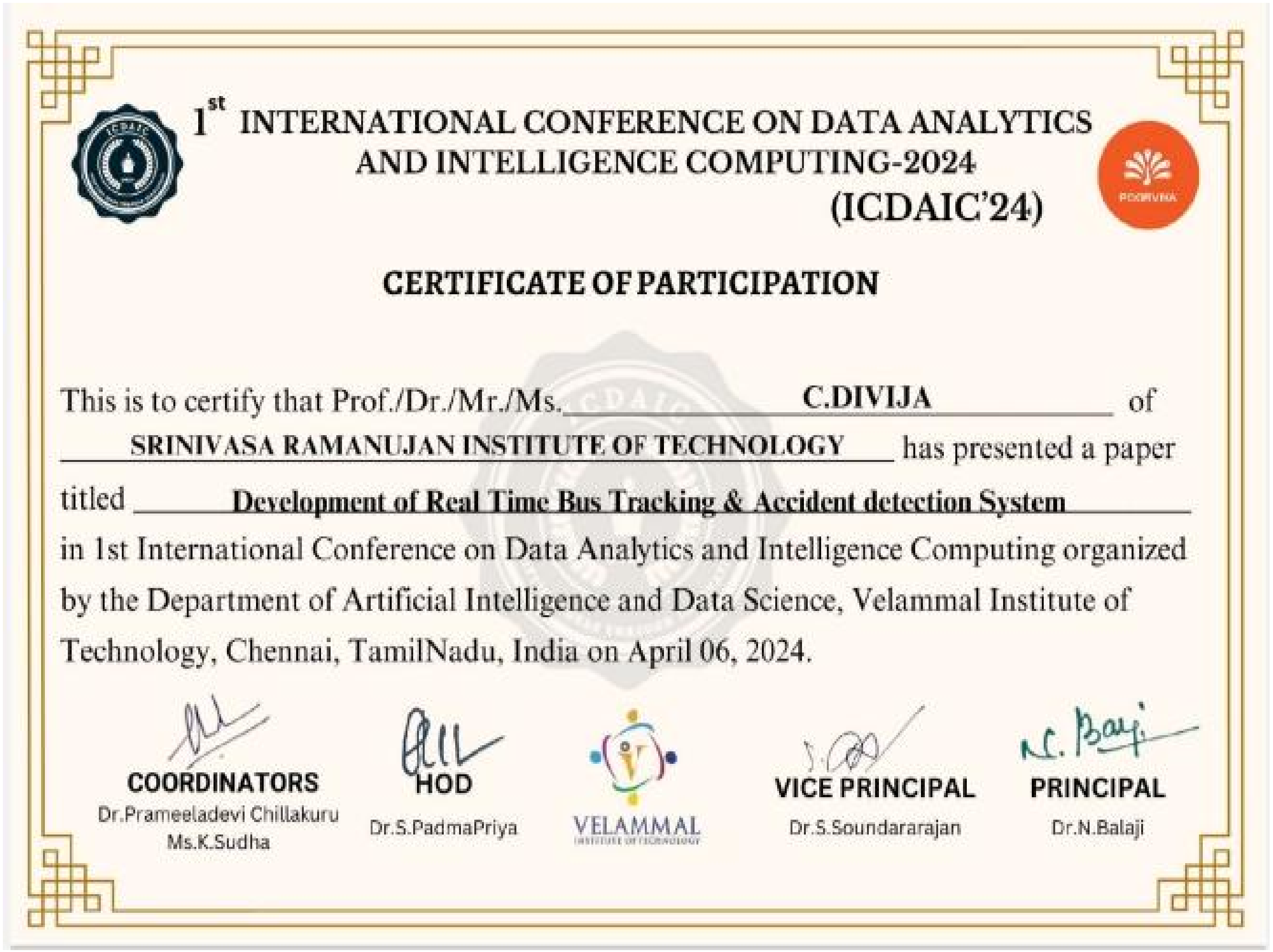
[illegible]
COORDINATORS
Dr.Prameeladevi Chillakuru
Ms.K.Sudha

[illegible]
HOD
Dr.S.PadmaPriya

VELAMMAL
INSTITUTE OF TECHNOLOGY

[illegible]
VICE PRINCIPAL
Dr.S.Soundararajan

[illegible]
PRINCIPAL
Dr.N.Balaji



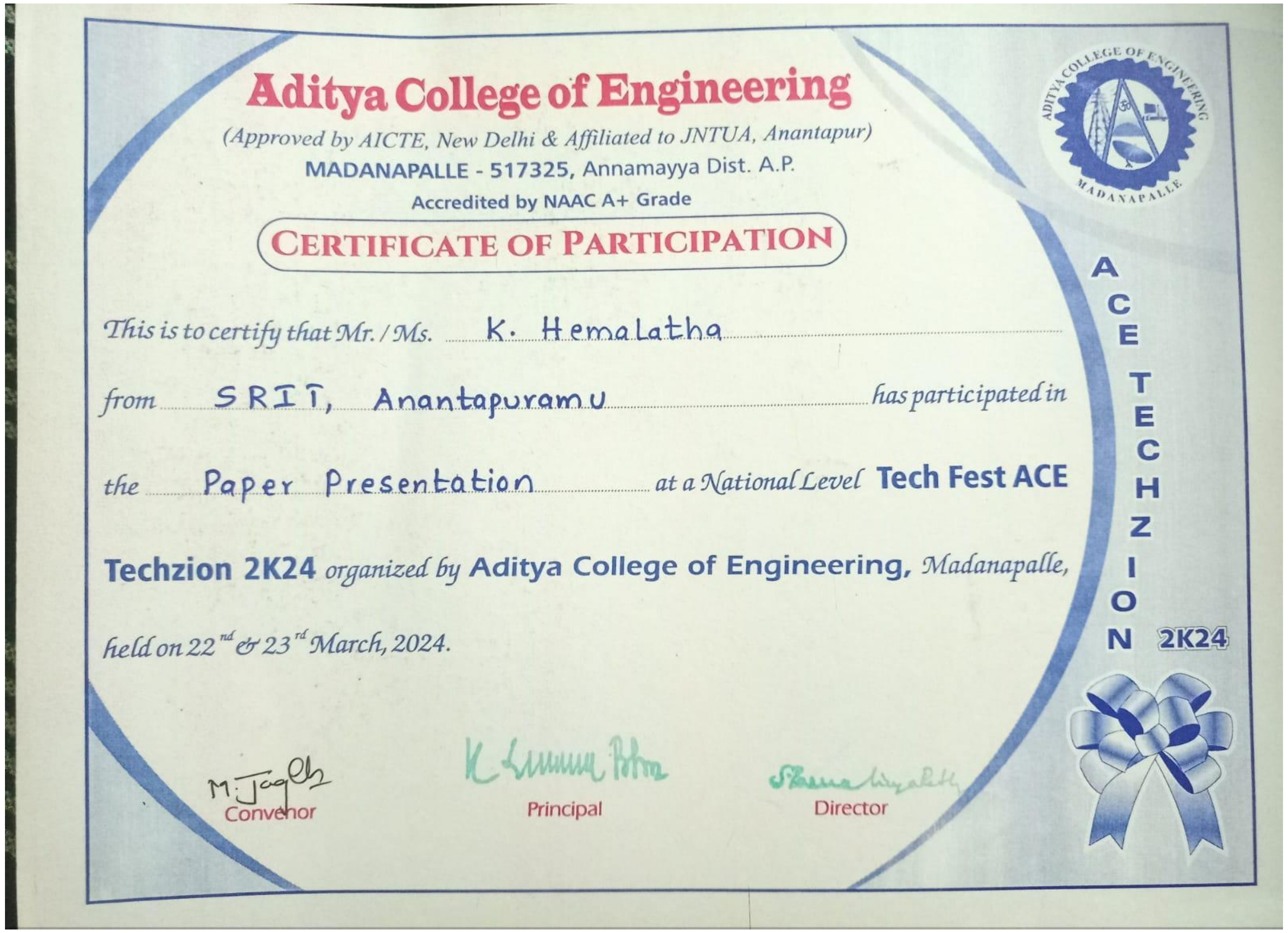
1st INTERNATIONAL CONFERENCE ON DATA ANALYTICS AND INTELLIGENCE COMPUTING-2024 (ICDAIC'24)

CERTIFICATE OF PARTICIPATION

This is to certify that Prof./Dr./Mr./Ms. **C.DIVIJA** of **SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY** has presented a paper titled **Development of Real Time Bus Tracking & Accident detection System** in 1st International Conference on Data Analytics and Intelligence Computing organized by the Department of Artificial Intelligence and Data Science, Velammal Institute of Technology, Chennai, TamilNadu, India on April 06, 2024.

COORDINATORS	HOD	VICE PRINCIPAL	PRINCIPAL
Dr.Prameeladevi Chillakuru Ms.K.Sudha	Dr.S.PadmaPriya	Dr.S.Soundararajan	Dr.N.Balaji

VELAMMAL



Aditya College of Engineering

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapur)
MADANAPALLE - 517325, Annamayya Dist. A.P.
Accredited by NAAC A+ Grade

CERTIFICATE OF PARTICIPATION

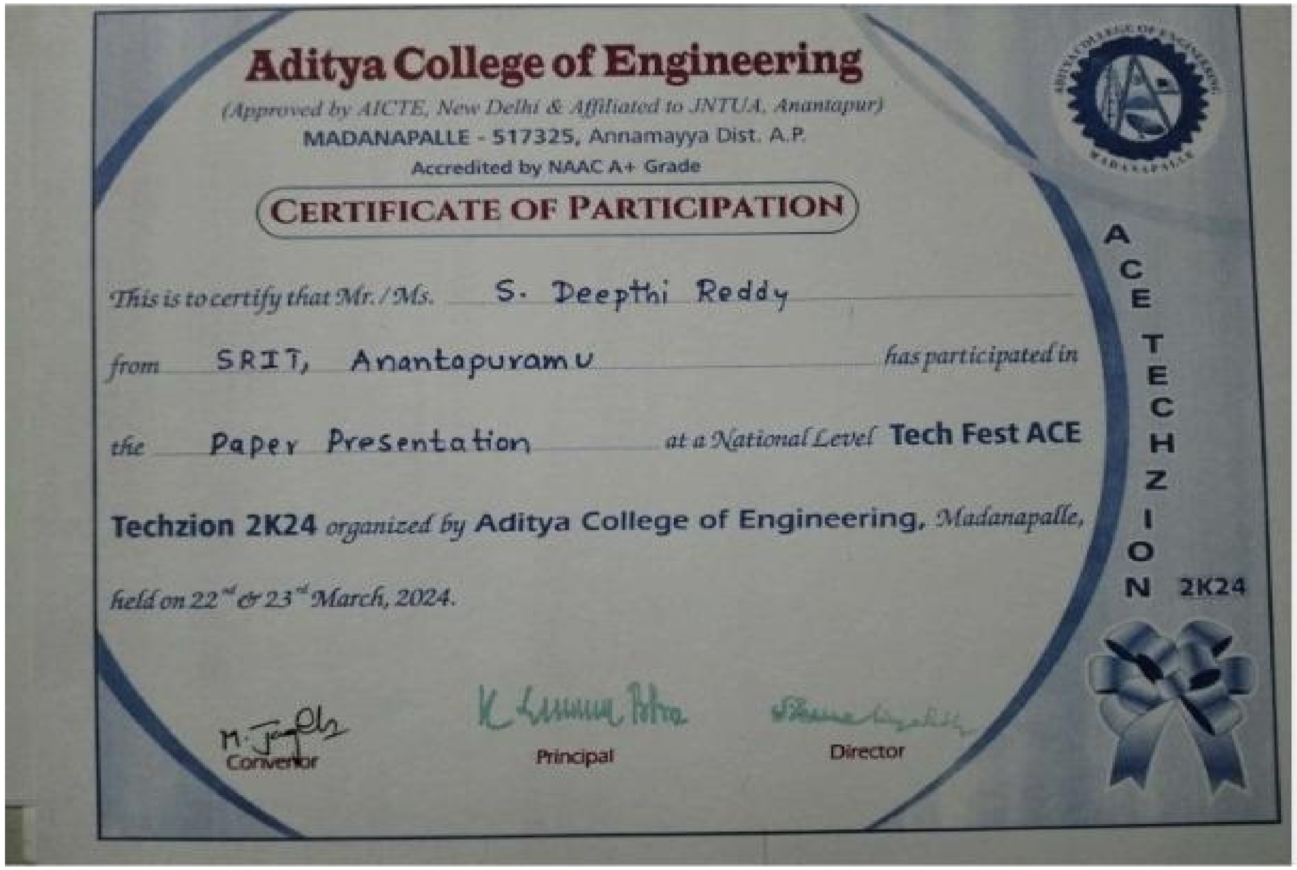
ACE TECHZION 2K24

This is to certify that Mr. / Ms. K. Hemalatha *from* SRIT, Anantapuramu *has participated in the* Paper Presentation *at a National Level* **Tech Fest ACE Techzion 2K24** *organized by* **Aditya College of Engineering,** *Madanapalle, held on 22nd & 23rd March, 2024.*

M.Jagdh
Convenor

[illegible]
Principal

[illegible]
Director



Aditya College of Engineering

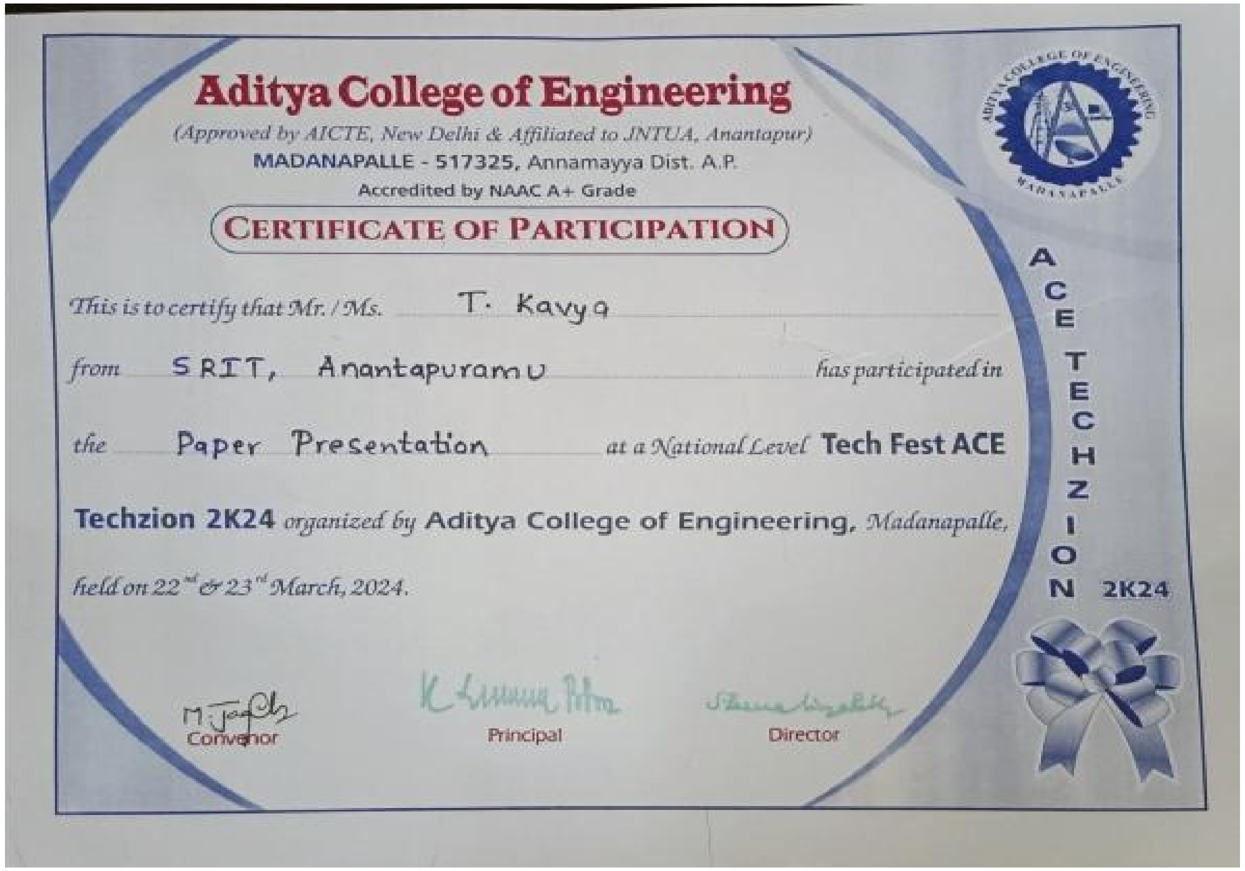
(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapur)
MADANAPALLE - 517325, Annamayya Dist. A.P.
Accredited by NAAC A+ Grade

CERTIFICATE OF PARTICIPATION

This is to certify that Mr. / Ms. S. Deepthi Reddy *from* SRIT, Anantapuramu *has participated in the* Paper Presentation *at a National Level* **Tech Fest ACE Techzion 2K24** *organized by* Aditya College of Engineering, *Madanapalle, held on 22nd & 23rd March, 2024.*

ACE TECHZION 2K24

[illegible] Convenor | [illegible] Principal | [illegible] Director



Aditya College of Engineering

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapur)
MADANAPALLE - 517325, Annamayya Dist. A.P.
Accredited by NAAC A+ Grade

CERTIFICATE OF PARTICIPATION

This is to certify that Mr. / Ms. T. Kavya

from SRIT, Anantapuramu *has participated in*

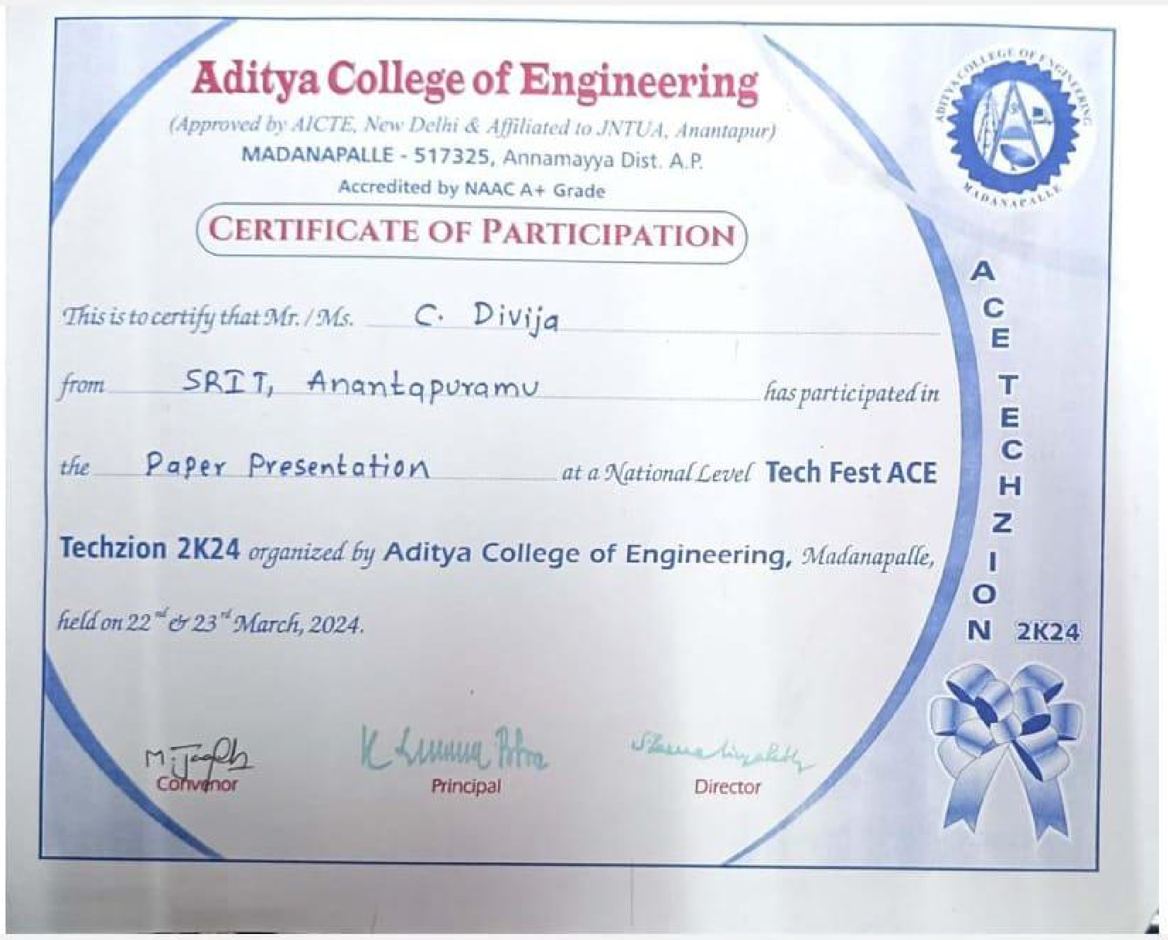
the Paper Presentation *at a National Level* **Tech Fest ACE Techzion 2K24** *organized by* **Aditya College of Engineering**, *Madanapalle, held on 22nd & 23rd March, 2024.*

ACE TECHZION 2K24

[illegible] Convenor

[illegible] Principal

[illegible] Director



Aditya College of Engineering

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapur)
MADANAPALLE - 517325, Annamayya Dist. A.P.
Accredited by NAAC A+ Grade

CERTIFICATE OF PARTICIPATION

ACE TECHZION 2K24

This is to certify that Mr. / Ms. C. Divija *from* SRIT, Anantapuramu *has participated in the* Paper Presentation *at a National Level* **Tech Fest ACE Techzion 2K24** *organized by* **Aditya College of Engineering,** *Madanapalle, held on 22nd & 23rd March, 2024.*

[illegible] Convenor

[illegible] Principal

[illegible] Director