URBAN GUARD-SMART IOT SOLUTIONS FOR SAFER CITIES

A PROJECT REPORT

Submitted by

GOVARDHAN E L	111621104044
ANIRUDH M	111621104012
ARJUN S	111621104017
BHAVANI SHANKAR T	111621104163

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

R.M.K. COLLEGE OF ENGINEERING AND TECHNOLOGY

PUDUVOYAL

ANNA UNIVERSITY : CHENNAI 600 025

MARCH 2025

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**URBAN GUARD-SMART IOT SOLUTION**" is the bonafide work of "**GOVARDHAN E L - (111621104044), ANIRUDH M - (111621104012), ARJUN S - (111621104017), BHAVANI SHANKAR T - (11621104163)**" who carried out the project work under my supervision

SIGNATURE	**SIGNATURE**
Dr. N. Gangatharan	**Dr.N.G.Praveena**
HEAD OF THE DEPARTMENT	SUPERVISOR
	PROFESSOR
Electronics and Communication Engineering	Electronics and Communication Engineering
R.M.K College of Engineering and Technology	R.M.K College of Engineering and Technology

Submitted for the Viva Voce examination held on ________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Support on demand, encouragement at the needed moment and guidance in the right direction are indispensable for the success of our project. We have received these in excess from all corners from various people. We are glad to submit our gratitude to them.

We thank **Shri.R.S.Munirathinam**, our beloved Chairman and **Shri.R.M.Kishore**, our Vice Chairman for extending a generous hand in providing the best resources to the college.

Dr.N.Suresh Kumar, the esteemed Head of our Institution has been a source of motivation to all the staff and students of our college. We are so much thankful to him. Our sincere thanks to **Dr.K. Sivaram**, Dean (Academic) for giving support to complete the project.

Our sincere thanks to **Dr.Gangatharan**, the Head of the Department for his continuous support and motivation throughout our project.

We extend our profound gratitude to **Ms.S.Sesha Vidhya**, our Project Coordinator and our Supervisor **Dr.N.G.Praveena** for her guidance. She has indeed been open all the time to help us throughout the course of the project.We thank her for giving us full support to complete this project successfully.

Last, but not the least, we take this opportunity to thank all the faculty members and supporting staff of the Department of Electronics and Communication Engineering.

Regards to our family, classmates and friends who offered an unflinching moral support for completion of this project.

ABSTRACT

Urbanization brings challenges such as flooding, electrical hazards, inefficient street lighting, and air pollution, requiring innovative solutions for improved city management. This project presents an integrated smart city system utilizing IoT technology with Arduino and ESP32 microcontrollers. The system includes a multi-tier flood detection mechanism that provides early warnings to officials and evacuation alerts to the public via SMS. To enhance electrical safety, a detection system automatically identifies power line faults in floodwater and shuts off the power supply to prevent electrocution hazards. The smart street lighting solution optimizes energy consumption through automated lighting control based on environmental conditions and includes a fault detection system for efficient maintenance. Sensor data is processed and transmitted to the ThingSpeak IoT cloud platform, enabling real-time monitoring and data-driven decision-making for city officials. This project serves as a proof-of-concept for smart city solutions, demonstrating how IoT technology can enhance urban resilience, safety, and sustainability.Additionally, the air quality monitoring system continuously measures pollutant levels and transmits real-time data to a cloud platform, aiding in air pollution control measures and ensuring healthier urban environments.This comprehensive IoT-based smart city solution aims to improve urban resilience, reduce risks associated with environmental hazards, and contribute to a safer, smarter, and more efficient urban ecosystem.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

GSM	GLOBAL SYSTEM FOR MOBILE COMMUNICATION
IDE	INTEGRATED DEVELOPMENT ENIVIRONMENT
PDB	POWER DISTRIBUTION
IOT	INTERNET OF THINGS
ESP32	ESPRESSIF SYSTEMS 32-BIT MICROCONTROLLER
LDR	LIGHT DEPENDENT RESISTOR
DHT11	DIGITAL HUMIDITY AND TEMPERATURE SENSOR
MQ153	AIR QUALITY SENSOR MODEL MQ135
SMS	SHORT MESSAGE SERVICE
API	APPLICATION PROGRAMMING INTERFACE
LED	LIGHT EMITTING DIODE
PCB	PRINTED CIRCUIT BOARD
PWM	PLUSE WIDTH MODULATION
LCD	LIQUID CRYSTAL DISPLAY
GPS	GLOBAL POSITIONING SYSTEM
ADC	ANALOG TODIGITAL CONVERTER
SPI	SERIAL PERIPHERAL INTERFACE
I2C	INTER-INTERGRATED CIRCUIT

CHAPTER 1
INTRODUCTION

Rapid urbanization and population growth have placed immense pressure on city infrastructure, leading to challenges such as flooding, electrical hazards, inefficient street lighting, and deteriorating air quality. Traditional urban management systems often operate in isolation, resulting in delayed responses, increased risks, and inefficiencies in addressing these critical issues. Flooding, for instance, not only causes significant damage to property but also poses severe safety risks, including electrocution due to submerged power lines. Poorly maintained street lighting contributes to accidents and security concerns, while rising pollution levels adversely affect public health and environmental sustainability.To address these challenges, this project proposes an integrated IoT-based smart city solution that leverages real-time monitoring and automated response mechanisms.

- The system incorporates flood detection with multi-tier alerts, electrical hazard prevention through automated power shutdown, smart street lighting for energy efficiency and fault detection, and air quality monitoring to provide valuable environmental data.
- By utilizing microcontrollers, sensors, and cloud-based data management, this solution enables city officials to make informed decisions, improve disaster preparedness, and enhance urban living standards.
- The implementation of such a comprehensive approach demonstrates how technology can be harnessed to create safer, more efficient, and sustainable smart cities.
- Modern cities face numerous challenges due to rapid urbanization, climate change, and increasing environmental concerns. Issues such as flooding, electrical hazards, inefficient street lighting, and declining air quality not only disrupt daily life but also pose significant threats to public safety and infrastructure.

1.1 Problem Statement

1. Lack of a real-time flood detection system leads to delayed response and increased damage in urban areas.
2. Unmonitored electrical installations pose risks of short circuits, electrocution, and fire hazards, especially during floods.
3. Absence of continuous air quality monitoring results in unawareness of harmful pollution levels, impacting public health
4. Traditional street lighting systems lack automated fault detection, leading to prolonged outages and unsafe road conditions Without real-time fault monitoring, maintenance teams must rely on manual inspections, increasing operational costs and labor.

1.2 Existing System

The existing urban management systems face significant limitations due to their fragmented and isolated operational structures. Different departments typically handle various city infrastructure components independently, resulting in inefficient communication, delayed responses, and inadequate resource allocation during emergencies.

- Traditional flood detection methods primarily rely on manual monitoring, which is prone to human error and delays, reducing the effectiveness of early warning systems. The dissemination of flood alerts to the public is often slow, inconsistent, and ineffective, leading to increased vulnerability and potential loss of life during critical situations.
- In current electrical safety mechanisms during floods are mostly reactive rather than preventive. Authorities generally respond only after accidents occur, lacking proactive measures to automatically detect fallen electrical wires submerged in floodwaters and shut down power promptly.

- Existing street lighting systems operate on fixed schedules or basic sensors without intelligent fault detection capabilities. This results in unnecessary energy consumption, higher operational costs, and delayed maintenance responses when faults occur.
- Similarly, air quality monitoring systems in many cities are limited by low spatial resolution, high operational costs, and infrequent data updates. These systems typically provide generalized data that does not accurately reflect localized pollution levels or real-time environmental conditions relevant to public health.
- Furthermore, most urban management solutions lack integration with IoT platforms for centralized data collection and analysis. The absence of an integrated approach prevents comprehensive analysis across multiple urban domains and limits the ability of city administrators to make informed decisions promptly.
- Overall, existing systems fail to provide a unified platform for addressing multiple urban challenges cohesively

1.3 Proposed System

- The proposed system employs water level sensors to detect flood risks in real-time.
- Electrical hazard prevention is implemented using current and voltage sensors, detecting power fluctuations and potential short circuits in urban electrical grids.
- Air quality monitoring is achieved through gas and particulate matter sensors, providing real-time air pollution data for public safety and government response.
- GSM modules send alerts via SMS notifications to authorities and citizens, ensuring people near flood-prone areas receive early warnings.

- The entire system is powered by a Li-ion battery, making it independent of AC power and ensuring continuous operation during blackouts.
- LCD displays provide real-time monitoring for local disaster management centers, allowing for quick decision-making.
- With this proposed system, flood detection, electrical hazard prevention, and air quality monitoring are automated, providing a reliable, power-efficient, and internet-independent solution for smart city safety.

BLOCK DIAGRAM

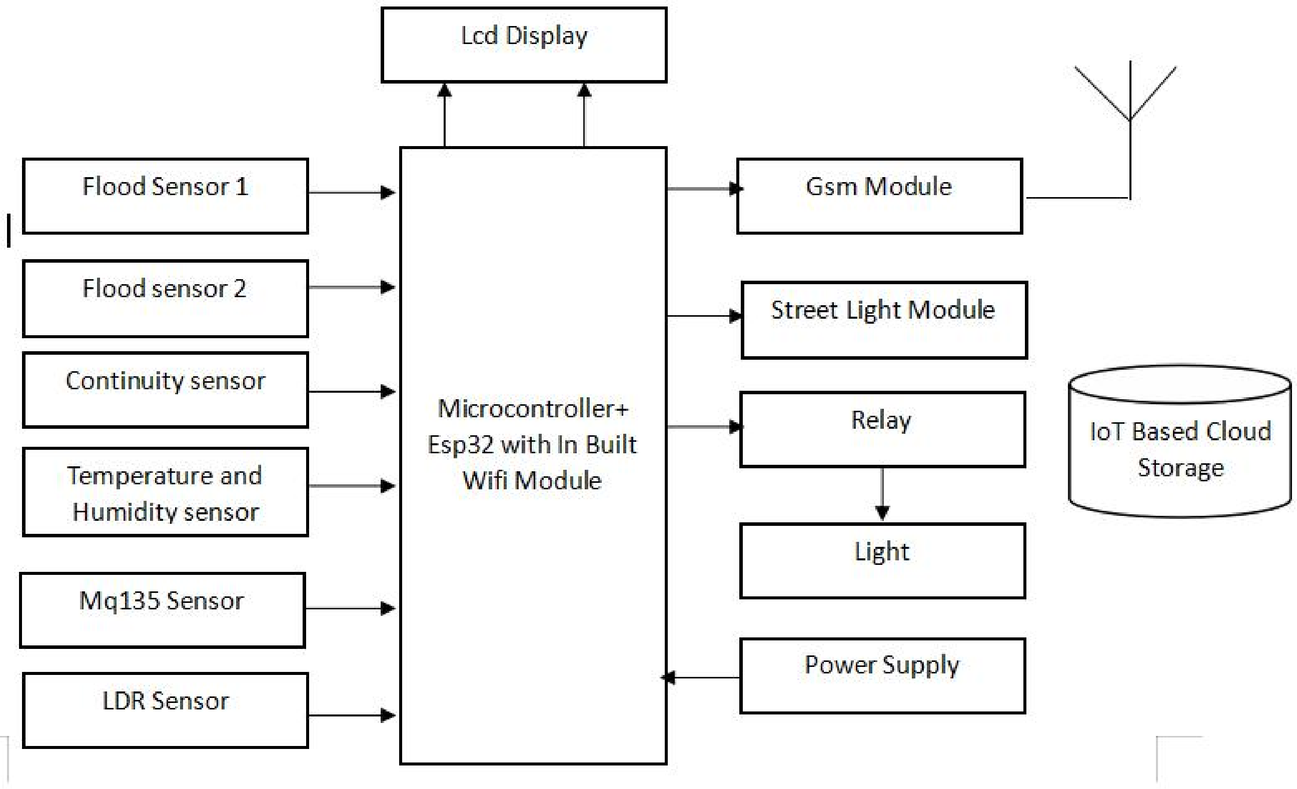


Fig.1.1 Block Diagram of the Proposed System

CHAPTER 2

LITERATURE SURVEY:

2.1 INTRODUCTION:

In the ever-evolving landscape of smart city technologies, our literature survey serves as a guiding beacon, navigating through diverse research contributions in IoT-based flood monitoring, air quality assessment, smart street lighting, and electrical hazard prevention.

This exploration delves into the methodologies, technological advancements, and real-world implementations of IoT-driven solutions in urban infrastructure.

Through an in-depth review of academic literature, we aim to identify key trends, challenges, and opportunities that shape the future of smart city development.

2.2 REVIEW OF THE RELATED WORKS:

Title 1: IoT-Based Smart Flood Monitoring and Early Warning System

Authors: Kumar, A.
Year: 2018
Description: This study presents an IoT-based flood monitoring system that integrates real-time water level detection with early warning mechanisms. The proposed system employs water level sensors, GSM modules, and cloud-based analytics to transmit flood alerts to authorities and residents. The research emphasizes the importance of low-latency communication and automated flood prediction to minimize disaster impact. By leveraging IoT-enabled sensor networks, the study demonstrates a scalable and cost-effective approach to flood management in urban areas.

Title 2: Development of an IoT-Based Air Quality Monitoring System for Smart Cities

Authors: Singh, P., & Sharma, R.
Year: 2020
Description: This research introduces an IoT-based air quality monitoring system designed for real-time pollution assessment in smart cities. The system integrates gas sensors (such as MQ135), microcontrollers, and cloud computing to analyze air pollutants and generate real-time alerts. The study highlights the role of IoT in mitigating environmental hazards and optimizing urban planning. The results indicate improved accuracy in air quality forecasting and enhanced decision-making capabilities for municipal authorities.

Title 3: Implementation of Smart Street Lighting Systems Using IoT and AI

Authors: Gupta, R., & Verma, S.
Year: 2019
Description: This paper explores the development of an intelligent street lighting system that utilizes IoT and AI for energy efficiency. The proposed system incorporates motion sensors, adaptive brightness control, and remote monitoring through a centralized dashboard. By leveraging AI-driven analytics, the study demonstrates reduced energy consumption and optimized lighting schedules, contributing to sustainable urban development. The implementation results show significant energy savings and improved operational efficiency in smart city environments.

Title 4: Prevention of Electrical Hazards in Urban Areas Using IoT-Based Monitoring Systems

Authors: Chen, L., & Zhao, Y.
Year: 2021
Description: The study investigates the application of IoT-based monitoring

systems for detecting electrical hazards in urban infrastructures. The proposed system integrates voltage and current sensors, AI-driven predictive maintenance, and cloud-based alert mechanisms to prevent electrical failures. The research highlights the effectiveness of real-time monitoring in reducing electrical fire risks and improving public safety. The study presents case studies demonstrating the successful deployment of such systems in metropolitan areas.

Title 5: IoT-Enabled Smart City Infrastructure for Disaster Management and Hazard Prevention

Authors: Alam, M., & Hasan, R.
Year: 2022
Description: This paper presents an IoT-enabled framework for smart city disaster management, focusing on flood detection, air quality monitoring, and hazard prevention. The system utilizes a network of wireless sensors, cloud computing, and machine learning algorithms to predict and mitigate urban disasters. The study outlines key challenges in large-scale deployment and offers insights into data-driven emergency response strategies. The findings emphasize the role of IoT in building resilient urban infrastructures.

Title 6: Sensor Networks for Smart Cities: Flood and Air Pollution Detection Using IoT

Authors: Lee, H., & Park, J.
Year: 2020
Description: This research examines the integration of sensor networks in smart cities for flood monitoring and air pollution detection. The study employs real-time sensor data analytics to assess environmental conditions and provide early warnings. The results highlight the effectiveness of IoT-driven solutions in mitigating urban environmental risks, with a focus on scalability and cost

efficiency. The paper discusses challenges such as data privacy, sensor accuracy, and network connectivity in large-scale deployments.

Title 7: Integration of Wireless Sensor Networks for Smart City Applications: A Case Study on Flood Monitoring and Street Lighting

Authors: Kumar, S., & Patel, D.
Year: 2021
Description: This study explores the integration of wireless sensor networks (WSNs) for flood monitoring and smart street lighting in urban environments. The system uses IoT-enabled sensors to optimize flood detection and energy-efficient lighting control. The research emphasizes the benefits of real-time monitoring and remote management in smart city applications. The study provides a case analysis showcasing the effectiveness of WSNs in enhancing urban resilience and sustainability.

Title 8: A Review on IoT-Based Smart City Solutions: Flood Detection, Air Quality Monitoring, and Hazard Prevention

Authors: Tan, W., & Zhang, X.
Year: 2019
Description: This comprehensive review examines IoT-based smart city solutions, focusing on flood detection, air quality assessment, and electrical hazard prevention. The paper categorizes existing research based on technological advancements, deployment challenges, and future research directions. The study highlights key IoT frameworks and sensor technologies employed in smart urban infrastructures. The review provides a comparative analysis of different smart city solutions and discusses policy implications for large-scale adoption.

2.3 SUMMARY:

The reviewed literature highlights the significant role of IoT in addressing key smart city challenges, including flood monitoring, air quality assessment, smart street lighting, and electrical hazard prevention. Various studies propose IoT-based frameworks that utilize sensor networks, cloud computing, and AI-driven analytics to enhance urban resilience. The findings underscore the need for real-time monitoring, predictive maintenance, and automated control systems in modern cities.The research on flood monitoring systems demonstrates the effectiveness of IoT-enabled early warning mechanisms in mitigating disaster impacts. Similarly, air quality monitoring solutions leverage sensor data analytics to provide accurate pollution assessments. Smart street lighting systems employing AI-driven optimization techniques showcase energy efficiency improvements, while IoT-based electrical hazard prevention models enhance public safety through predictive analytics. A recurring theme across the studies is the integration of wireless sensor networks and cloud-based platforms for seamless urban management. Despite challenges such as sensor accuracy, data security, and scalability, the literature indicates a promising trajectory for IoT-driven smart city solutions. These insights serve as a foundation for advancing research in urban infrastructure optimization, sustainability, and disaster resilience.

CHAPTER 3

SYSTEM DESCRIPTION

3.1 PROPOSED SYSTEM FOR URBANGUARD: SMART CITY MONITORING AND MANAGEMENT:

The UrbanGuard system is designed to enhance urban safety, disaster management, and environmental monitoring using an integrated IoT-based approach. The system incorporates real-time monitoring, automated alerts, and cloud-based data management to address critical urban challenges such as flooding, electrical hazards, inefficient street lighting, and air pollution.

The proposed system consists of the following key components:

- Water Level Sensors for flood detection and early warning alerts.
- Continuity Sensor for detecting electrical hazards in flooded areas.
- LDR Sensors for automatic street lighting control.
- DHT11 and MQ135 Sensors for temperature, humidity, and air quality monitoring.
- GSM Module for sending SMS alerts to officials and the public.
- ESP32 and Arduino Microcontrollers for data processing and communication.
- ThingSpeak Cloud Platform for real-time data storage and analysis.

3.2 SYSTEM FUNCTIONALITY

- Flood Detection and Alert System: Water level sensors detect rising water levels. The system provides a two-tier warning mechanism, sending early alerts to city officials and evacuation messages to the public when critical levels are reached.

- Electrical Hazard Prevention: The continuity sensor identifies electrical wires submerged in floodwater and automatically shuts down the power supply, preventing electrocution risks and enhancing public safety.
- Smart Street Lighting Automation: LDR sensors detect ambient light levels and automatically turn streetlights on or off, reducing energy consumption. A fault detection system alerts maintenance teams in case of light failures.
- Air Quality and Environmental Monitoring: DHT11 and MQ135 sensors measure temperature, humidity, and air pollution levels. The data is uploaded to the ThingSpeak cloud for real-time monitoring and analysis, helping officials track air quality trends and implement corrective measures.
- Automated Stakeholder Communication: The GSM module ensures timely communication by sending alerts to officials and residents regarding flood risks, power shutdowns, and environmental hazards, improving public safety and awareness.
- Real-Time Data Processing and Visualization: Sensor data is processed by Arduino and ESP32 microcontrollers and stored in the ThingSpeak cloud. This allows city officials to access real-time insights through dashboards for better urban planning and decision-making.

By integrating automated monitoring, alert systems, and real-time data processing, the UrbanGuard system provides a scalable, efficient, and reliable approach to smart city management and disaster prevention.

CHAPTER 4

HARDWARE AND SOFTWARE REQUIREMENTS

The testing and implementation of the UrbanGuard smart city system require careful attention to both hardware and software components to ensure seamless integration and efficient functionality.Initially, the hardware requirements involve setting up and configuring various sensors and microcontrollers. The system includes water level sensors for flood detection, a continuity sensor for electrical hazard prevention, LDR sensors for smart street lighting automation, and DHT11 and MQ135 sensors for air quality monitoring.

These sensors are connected to Arduino and ESP32 microcontrollers, which process the data and communicate with the ThingSpeak IoT cloud. A GSM module is integrated to send real-time SMS alerts to city officials and the public. Each hardware component will be calibrated to ensure accurate data collection and reliable operation.

Once the hardware setup is verified, focus shifts to software development and deployment. The Arduino and ESP32 microcontrollers will be programmed to collect sensor data, process alerts, and automate responses. A ThingSpeak cloud platform will be used for real-time data storage and visualization, allowing city officials to monitor urban conditions remotely.

The flood detection system will be tested for real-time water level monitoring and emergency SMS alerts. The electrical hazard detection system will be assessed for its ability to identify and shut down fallen power lines. The smart street lighting system will be tested for automatic light adjustments and fault detection, while the air quality monitoring system will be evaluated for data accuracy and cloud-based reporting. Upon successful validation, **UrbanGuard** will be fully deployed for continuous operation, with ongoing monitoring and maintenance ensuring efficient urban management, disaster preparedness, and public safety.

4.1 COMPONENTS REQUIRED:

1)ARDUINO MEGA 2560:

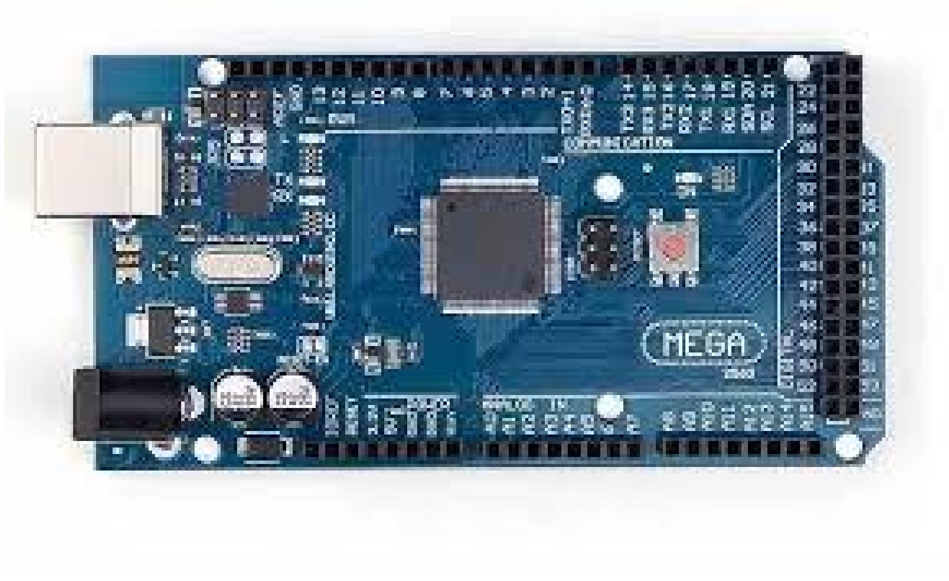


Fig 4.1 Arduino Mega 2560

The Arduino Mega 2560 is a powerful microcontroller board based on the ATmega2560 chip. It is designed for projects that require more I/O pins, memory, and processing power than the standard Arduino Uno. The Mega 2560 is widely used in IoT, robotics, automation, and large-scale embedded systems.

Symbol	Description	Min Voltage	Max Voltage
VINMAX	Maximum input voltage	6V	12V/20V(absolute max)
VUSBMAX	Maximum USB input voltage	-	5.5V
PMAX	Maximum power consumption	-	9W(at 12V input)

Table 4.1 – Power Consumption

The Arduino Mega 2560 can be powered via USB (5V), External power supply (7V – 12V recommended, up to 20V max), Vin pin for direct power

1. I/O & Expansion: The Mega 2560 has 54 digital I/O pins, 15 PWM pins, and 16 analog input pins, making it ideal for large projects with multiple sensors, displays, and actuators.Digital I/O (54 pins): Used for reading sensors, controlling LEDs, motors, and other devices.Analog Input (16 pins): Reads data from analog sensors like temperature, light, and soil moisture sensors.

2. Processing Power & Memory: At the heart of the Arduino Mega 2560 is the ATmega2560 microcontroller, which offers:This increased memory capacity compared to the Arduino Uno (32KB flash, 2KB SRAM, 1KB EEPROM) makes the Mega 2560 suitable for complex projects requiring more program space and data storage.

3. Programming & Software: Programming the Mega 2560 is simple and user-friendly using the Arduino IDE (Integrated Development Environment).Uses C/C++ programming language.Compatible with a vast library ecosystem for sensors, displays, and communication modules.

4. Serial Communication & Connectivity: Unlike the Arduino Uno, which has only one serial port, the Mega 2560 offers four hardware UART serial ports, allowing it to communicate with multiple devices, such as GSM/GPRS modules (SIM800L, SIM900, etc.), GPS modules, Bluetooth and Wi-Fi modulesAdditionally, it supports I2C and SPI protocols for seamless connectivity with displays, sensors, and storage devices.

5. Expansion & Shields: The Arduino Mega 2560 supports a variety of shields (add-on boards) to extend functionality, including Ethernet shield for IoT applications, Motor driver shield for robotics and automation, LCD and touchscreen shields for user interfaces, Wireless communication shields (Wi-Fi, Bluetooth, LoRa, Zigbee) for remote monitoring.

Its open-source nature allows users to modify, customize, and enhance both hardware and software as per project requirements.

2) WATER LEVEL SENSOR:

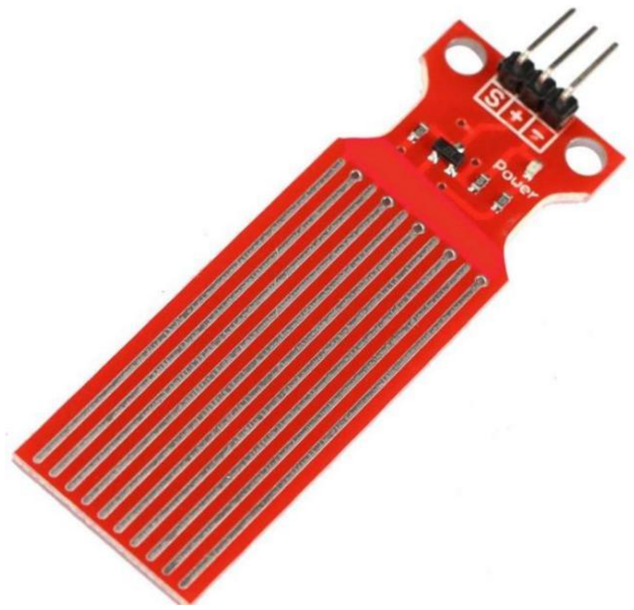


Fig 4.2 Water Level Sensor

A water level sensor is a device used to detect the level of water in a container or reservoir. It is commonly employed in various applications such as water tanks, industrial systems, agricultural irrigation systems, and home appliances like washing machines and dishwashers. Here are some key details about water level sensors

1. Working Principle: Water level sensors operate on different principles depending on the specific type of sensor. Some common working principles include:

- Float Switches: These sensors consist of a buoyant float connected to a switch mechanism. As the water level rises or falls, the float moves accordingly, triggering the switch to open or close.

- Capacitive Sensors: Capacitive water level sensors use changes in capacitance to detect the presence or absence of water. When water comes into contact with the sensor probe, it changes the capacitance.
- Ultrasonic Sensors: Ultrasonic water level sensors emit high-frequency sound waves that bounce off the surface of the water and are reflected back to the sensor. By measuring the time taken for the sound waves to return, the sensor can determine the distance to the water surface and thus the water level.

- Pressure Sensors: Pressure-based water level sensors measure the hydrostatic pressure exerted by the water column above the sensor.

2. Types: Water level sensors come in various types and configurations to suit different applications. Some common types include:

- Vertical Float Switches: These sensors consist of a buoyant float attached to a vertical rod or cable. As the water level changes, the float moves up or down along the rod, activating the switch mechanism.

- Horizontal Float Switches: Similar to vertical float switches, but the float moves horizontally along a guide rod or rail.

- Submersible Sensors: These sensors are designed to be fully submerged in water and are often used in tanks or reservoirs where the water level varies.

- Non-Contact Sensors: Non-contact sensors, such as ultrasonic sensors, do not come into direct contact with the water and are suitable for applications where contact with the water is not desirable or feasible.

3. Installation: Water level sensors are typically installed at predetermined levels within a container or reservoir, depending on the desired level monitoring requirements. Installation methods vary depending on the type of sensor.

4. Output: Water level sensors provide output signals that can be used to indicate the water level or trigger control actions in automated systems. The output signal may be in the form of a digital signal (e.g., open/closed switch), analog voltage or current, or digital communication protocols such as I2C or Modbus.

5. Applications: Water level sensors are used in a wide range of applications, including:

- Monitoring water levels in tanks, reservoirs, and wells for water management and conservation.
- Controlling water pumps and valves to maintain desired water levels.Preventing overflow or dry-running conditions in water systems.
- Monitoring water levels in appliances such as washing machines, dishwashers, and water heaters.
- Industrial applications such as wastewater treatment plants, cooling towers, and chemical processing systems.

Water level sensors play a crucial role in ensuring efficient and reliable operation of water systems and are available in various types and configurations to suit different applications and requirements.

3) MQ135 GAS SENSOR

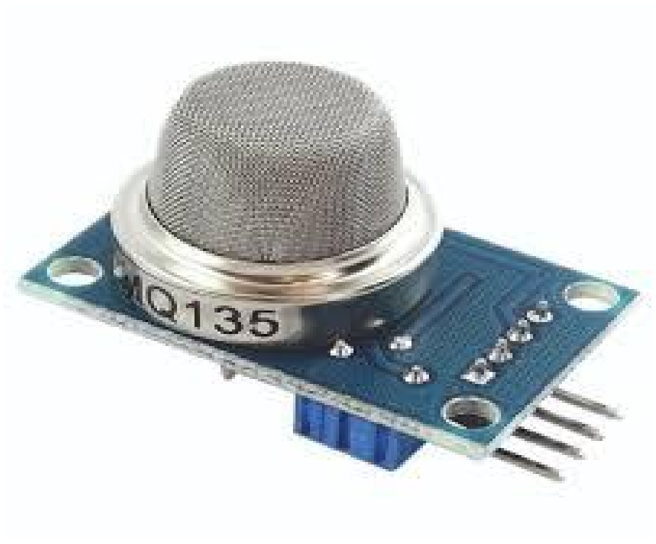


Fig 4.3 MQ135 Gas Sensor

An air quality sensor is a device used to detect and measure the concentration of pollutants or harmful gases in the air. These sensors are commonly used in indoor and outdoor environments to monitor air pollution levels, ensure regulatory compliance, and enhance environmental health and safety. Here are some key details about air quality sensors:

1. Working Principle

Air quality sensors operate using different detection mechanisms depending on the type of pollutant being measured. Some common working principles include:

- Electrochemical Sensors: These sensors measure gas concentrations by generating an electrical signal when a target gas reacts with an electrode. They are commonly used for detecting carbon monoxide (CO), ozone (O3), and nitrogen dioxide (NO2).

- Optical Sensors: Optical sensors, such as infrared (NDIR) sensors, detect gases like carbon dioxide (CO2) by measuring the absorption of infrared light at specific wavelengths.
- Laser Scattering Sensors: These sensors use a laser to detect airborne particulate matter (PM2.5, PM10). When particles pass through the sensor, they scatter the laser light, and the amount of scattering is used to determine the concentration of particles.
- Metal Oxide Semiconductor (MOS) Sensors: MOS sensors detect air pollutants like volatile organic compounds (VOCs) and carbon monoxide by measuring changes in electrical resistance when gases interact with the sensor surface.

2. Types

Air quality sensors come in various types and configurations, depending on the pollutants they detect. Some common types include:

- PM Sensors: Designed to detect particulate matter (PM1.0, PM2.5, PM10) in the air, which is crucial for monitoring dust, smoke, and smog levels.
- Gas Sensors: Used to detect specific gases such as CO, CO2, NO2, SO2, and VOCs.
- Multi-Gas Sensors: Capable of measuring multiple pollutants simultaneously for comprehensive air quality analysis.
- Indoor Air Quality (IAQ) Sensors: Often include a combination of PM, CO2, VOC, and humidity sensors to assess indoor air conditions.
- Outdoor Air Quality Sensors: Rugged sensors designed to withstand environmental conditions and provide real-time pollution monitoring in cities and industrial areas.

3. Installation

Air quality sensors are installed in various locations depending on their purpose:

- Indoor sensors: Placed in homes, offices, schools, and hospitals to monitor indoor air pollution.
- Outdoor sensors: Mounted on poles, buildings, or vehicles to track environmental pollution levels.

4. Output

Air quality sensors provide output signals that can be used for monitoring and control purposes. These outputs can be:

- Digital Output: Data transmitted via I2C, UART, or SPI communication protocols for integration with microcontrollers and IoT systems.
- Analog Output: Voltage or current-based signals representing pollutant concentration levels.

5. Applications

Air quality sensors are widely used in various sectors, including:

- Environmental Monitoring: Tracking pollution levels in urban areas to assess air quality and compliance with regulations.
- Smart Homes & Buildings: Used in HVAC systems to optimize air ventilation based on CO2 and VOC levels.
- Industrial Safety: Detecting toxic gases in workplaces to prevent health hazards.

Air quality sensors play a vital role in protecting human health and the environment.

4) CONTINUITY SENSOR

Fig 4.4 Continuity Sensor

A continuity sensor is a device used to detect electrical continuity in a circuit. It is commonly employed in electrical testing, troubleshooting, and maintenance applications to check whether a circuit is complete or has a break. These sensors are widely used by electricians, engineers, and technicians to ensure proper wiring and connectivity in electrical systems. Here are some key details about continuity sensors:

1. Working Principle

Continuity sensors operate based on the principle of electrical conductivity. When two test points in a circuit are connected, the sensor detects the flow of electrical current and indicates continuity. Some common working mechanisms include:

- Resistance Measurement: The sensor measures electrical resistance between two points. If the resistance is low (near zero), continuity is present; if the resistance is high (infinite), there is a break in the circuit.

- Audible Beep or LED Indication: Many continuity sensors provide an audible beep or light up an LED when continuity is detected, making it easy to use.
- Multimeter-Based Continuity Testing: Some continuity sensors are integrated into digital multimeters, allowing users to switch to a continuity test mode for quick diagnostics.

2. Types

Continuity sensors come in various types and configurations depending on their application and functionality. Some common types include:

- Handheld Continuity Testers: Simple devices with a probe and test lead that indicate continuity with a beep or LED.
- Multimeter-Based Continuity Sensors: Digital multimeters often include a continuity test function for circuit diagnostics.
- Clamp-Type Continuity Testers: These sensors can test continuity in wires without direct contact by detecting the electromagnetic field.
- Automated Continuity Sensors: Used in industrial applications to automatically detect continuity in complex wiring systems.

3. Installation

Continuity sensors do not require permanent installation, as they are typically used as testing tools. The testing process involves:

- Connecting the sensor's probes or leads to both ends of the circuit or wire.
- Observing the indicator (beep, LED, or display) to determine if continuity is present.
- Checking multiple points in complex circuits to locate faults or breaks.

4. Output

Continuity sensors provide different types of output signals to indicate circuit continuity. These outputs include:

- Audible Signal: A buzzer or beep sound when continuity is detected.
- Visual Signal: LED light or display screen showing a continuity indication.
- Digital Readout: Some advanced sensors provide numerical resistance values on an LCD screen.
- Data Logging: Certain industrial-grade continuity testers can store and transmit test results for documentation.

5. Applications

Continuity sensors are widely used in various industries and applications, including:

- Electrical Wiring Testing: Ensuring proper connections in household and industrial electrical systems.
- Circuit Board Troubleshooting: Detecting broken traces or open circuits in PCB manufacturing and repair.
- Automotive Diagnostics: Checking wiring harnesses, fuses, and electrical connections in vehicles.
- Industrial Equipment Maintenance: Verifying the integrity of electrical connections in machinery and automation systems.

Continuity sensors play a crucial role in electrical diagnostics and maintenance, providing a simple yet effective method for detecting circuit integrity. They are available in various forms, from basic handheld testers to advanced automated systems, making them an essential tool for electricians, engineers, and technicians.

5) RELAY MODULE

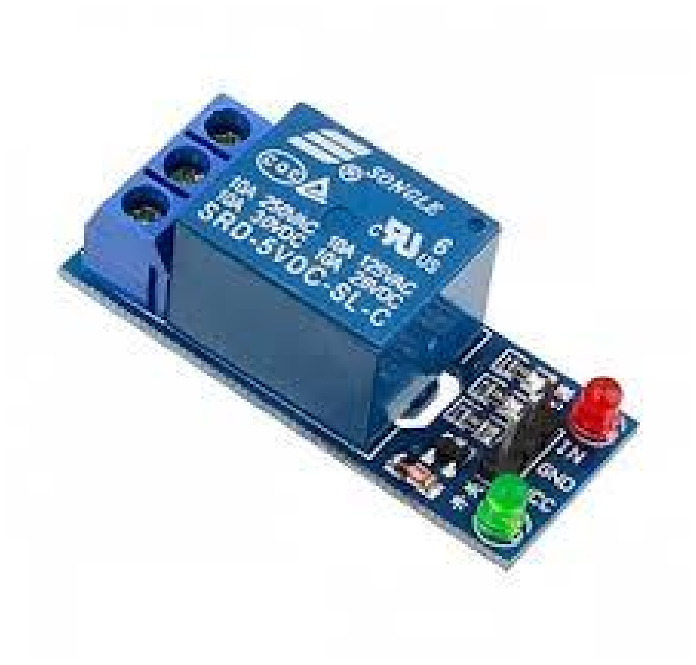


Fig.4.5 Relay Module

A relay module is an electrically operated switch that allows low-power microcontrollers like Arduino, Raspberry Pi, or ESP32 to control high-power devices such as lights, fans, motors, and household appliances. It acts as an interface between the control circuit and the load, providing electrical isolation and enabling automation in various applications. Here are some key details about relay modules:

1. Working Principle

A relay module works by using a small electrical signal to control a larger electrical load. The working mechanism includes:

- Electromagnetic Coil Activation: When a control signal is applied, the coil inside the relay is energized, creating a magnetic field.
- Switching Mechanism: The magnetic field pulls a switch (contact) inside the relay, either opening or closing the circuit to the connected load.

- Electrical Isolation: The low-voltage control side and the high-voltage load side are electrically isolated, ensuring safe operation.

2. Types

Relay modules come in various types based on their configuration, functionality, and number of channels. Common types include:

- Single-Channel Relay Module: Controls a single high-power device using one relay switch.
- Multi-Channel Relay Module: Available in 2, 4, 8, or more channels to control multiple devices independently.
- Solid-State Relay (SSR) Module: Uses semiconductor switching instead of mechanical contacts, offering faster operation and longer lifespan.
- Electromechanical Relay Module: Uses physical moving contacts for switching, suitable for high-power applications.

3. Installation

Relay modules are designed for easy integration with microcontrollers and power circuits. Installation typically involves:

- Connecting the Control Pins: Input pins (IN1, IN2, etc.) are connected to microcontroller GPIO pins.
- Powering the Module: Requires a DC power source, typically 5V or 12V, depending on the relay module.
- Wiring the Load: The high-power device is connected to the Common (COM), Normally Open (NO), or Normally Closed (NC) terminals based on the desired switching operation.

4. Output

Relay modules provide different output configurations based on switching states:

- Normally Open (NO): The circuit is open by default and closes when the relay is activated.
- Normally Closed (NC): The circuit is closed by default and opens when the relay is activated.
- Common (COM): The main terminal used for switching between NO and NC.
- LED Indicator: Many relay modules include LED indicators to show the switching status.

5. Applications

Relay modules are widely used in various industries and home automation projects, including:

- Home Automation: Controlling lights, fans, and electrical appliances remotely.
- Industrial Control Systems: Used in automation and machinery for controlling high-power devices.
- Motor and Pump Control: Switching motors, pumps, and solenoids in irrigation, HVAC, and robotics.

Relay modules play a crucial role in electrical and automation systems by providing safe and efficient switching between low-power control signals and high-power loads. They come in various types and configurations to suit different applications, making them an essential component in IoT, robotics, and industrial automation projects.

6) LCD DISPLAY

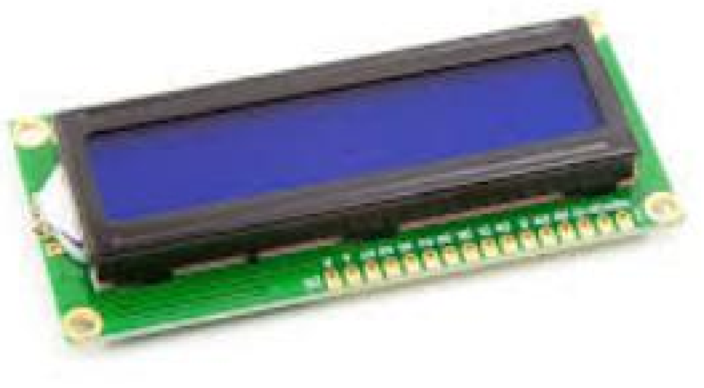


Fig 4.6 LCD Display

An LCD (Liquid Crystal Display) module is a widely used display technology in embedded systems, providing a simple and efficient way to display text, numbers, and basic graphics. LCD displays are commonly used in Arduino, Raspberry Pi, and other microcontroller based projects for user interfaces, data visualization, and system monitoring. Here are some key details about LCD displays:

1. Working Principle

An LCD display works by manipulating liquid crystals to control the passage of light. The working mechanism includes:

- Liquid Crystal Alignment: The display consists of liquid crystal molecules sandwiched between two glass layers.
- Backlight & Polarization: A backlight (LED) provides illumination, and polarizing filters control the light's direction.
- Segment or Pixel Control: Electrical signals change the alignment of liquid crystals, blocking or allowing light to create visible characters or graphics.

2. Types

LCD modules come in various types based on their display capabilities and interface. Common types include:

- Character LCDs: Display only text and numbers in fixed-size character grids (e.g., 16x2, 20x4).
- Graphical LCDs (GLCDs): Support custom graphics and images (e.g., 128x64 pixel displays).
- Segment LCDs: Used in clocks, calculators, and meters to display predefined numbers and symbols.
- TFT & OLED LCDs: Advanced displays that support full-color graphics and higher resolutions.
- I2C LCD Modules: Feature an I2C interface for simplified wiring and reduced pin usage.

3. Installation

LCD modules are designed for easy integration with microcontrollers. Installation typically involves:

- Connecting the Data Pins: Parallel LCDs require multiple digital pins (RS, E, D4–D7), while I2C LCDs need only two (SDA, SCL).
- Powering the Module: Requires 5V or 3.3V power supply depending on the display model.
- Adjusting the Contrast: A potentiometer is often used to set the screen contrast.

- Mounting the Display: The module can be mounted on a PCB or secured in an enclosure with screws or brackets.

4. Output

LCD displays provide visual output in the form of text or graphics. The type of output depends on the display type:

- Character LCDs show alphanumeric data with basic symbols.
- Graphical LCDs can display images, icons, and custom characters.
- Segment LCDs present predefined numerical or symbolic output.

5. Applications

LCD modules are used in a variety of applications, including:

- Embedded Systems: Displaying system status, sensor data, or error messages in microcontroller-based projects.
- Home Appliances: Used in microwave ovens, washing machines, and air conditioners to display settings and operation modes.
- Industrial Automation: Providing real-time monitoring information in control panels and machinery.
- Medical Devices: Used in health monitoring equipment such as heart rate monitors and digital thermometers.
- Consumer Electronics: Found in digital clocks, calculators, remote controls, and handheld gaming devices.

LCD displays offer a reliable and energy-efficient way to provide visual feedback in embedded systems, making them a popular choice for various applications.

7) GSM MODULE

Fig 4.7 GSM Module

A GSM (Global System for Mobile Communications) module is a compact electronic device that enables communication over the GSM network.These modules are designed to work with GSM networks, which are widely used for cellular communication globally. They support various GSM standards, including 2G, 3G, and in some cases, 4G/LTE.

1. Communication: The SIM800L module supports GSM (Global System for Mobile Communications) and GPRS (General Packet Radio Service) communication standards, allowing it to transmit voice, SMS messages, and data over cellular networks. It operates on frequencies in the 2G spectrum, making it compatible with many GSM networks worldwide.

2. Compact Size: One of the main advantages of the SIM800L module is its small size, making it suitable for applications where space is limited. The module typically comes in a compact package with dimensions around 2.5cm x 2.5cm, which makes it easy to integrate into embedded systems.

3. Low Power Consumption: The SIM800L module is designed for low power consumption, which is beneficial for battery-powered applications or projects

where energy efficiency is important. It has various power-saving features to minimize energy usage during idle periods.

4. UART Interface: The module communicates with the host microcontroller (such as Arduino or Raspberry Pi) using a UART (Universal Asynchronous Receiver-Transmitter) interface. This allows the microcontroller to send AT commands to control the module and receive responses for tasks such as sending SMS messages, making phone calls, or establishing GPRS connections.

5.Voltage Requirements: The SIM800L module typically operates at voltages between 3.4V to 4.4V, making it compatible with a wide range of power sources including lithium-ion batteries, DC power supplies, or voltage regulators.

6. External Components: To operate the SIM800L module, additional external components such as a SIM card holder, antenna, and power supply are required. The module usually has onboard circuitry for voltage regulation, SIM card detection, and antenna connection.

7. AT Commands: The SIM800L module is controlled using AT commands, which are standard commands used to communicate with GSM modules. These commands are sent from the microcontroller to the module

Overall, the GSM SIM800L module is a versatile and cost-effective solution for adding cellular connectivity to embedded projects, enabling communication over GSM/GPRS networks in a compact and energy-efficient package.

8) DHT11 Temperature and Humidity Sensor

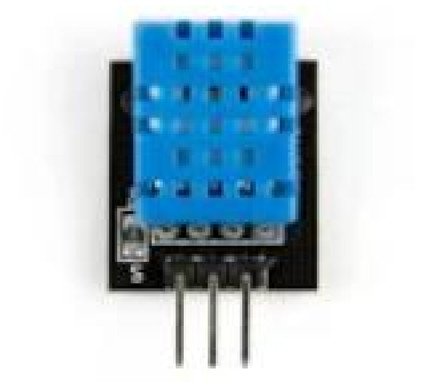


Fig 4.8 DH11 Temperature and Humidity Sensor

The DHT11 is a low-cost temperature and humidity sensor widely used in IoT, home automation, weather monitoring, and industrial applications. It provides digital output and is easy to interface with microcontrollers like the Arduino Mega 2560.

1. Working Principle

The DHT11 sensor consists of a thermistor for measuring temperature and a capacitive humidity sensor for detecting moisture in the air. The sensor processes these values and transmits the data as a digital signal to the microcontroller.

2. Specifications

Parameter	Value
Temperature Range	0°C to 50°C
Humidity Range	20% to 90% RH
Accuracy	±1°C (Temperature), ±5% RH (Humidity)
Operating Voltage	3.3V – 5V
Output	Digital Signal
Sampling Rate	1Hz (1 reading per second)

Table 4.2 - Specification of DH11 Sensor

3. Installation & Connection

The DHT11 sensor has 4 pins:

- VCC → Connect to 5V on Arduino Mega
- GND → Connect to GND
- Data → Connect to a digital pin on the Arduino
- NC (Not Connected)

A pull-up resistor (4.7kΩ – 10kΩ) is required between the VCC and Data pin for stable communication.

4. Output & Communication

- The DHT11 communicates using a single-wire protocol, sending temperature and humidity data to the microcontroller.
- The values can be displayed on an LCD screen, sent to a web server, or used to trigger cooling systems or alarms.

5. Applications

The DHT11 sensor is widely used in:

- Smart cities for monitoring temperature and humidity in air quality systems.
- Home automation to adjust air conditioning or humidifiers.
- Agriculture for greenhouse climate control.
- Industrial use for warehouse and server room temperature monitoring.

The DHT11 is a reliable and energy-efficient sensor, making it ideal for IoT applications requiring basic environmental monitoring.

9) ESP32 MODULE

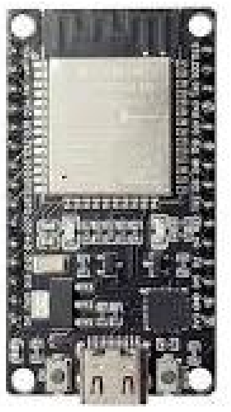


Fig 4.9 ESP32 Module

The ESP32 is a powerful and versatile microcontroller developed by Espressif Systems. It is widely used in IoT (Internet of Things), smart automation, wireless communication, and edge computing due to its built-in Wi-Fi and Bluetooth capabilities.

1. Working Principle

The ESP32 is based on a dual-core Tensilica Xtensa LX6 processor, which allows it to perform multiple tasks efficiently. It integrates Wi-Fi and Bluetooth, enabling it to connect to the internet and communicate with other devices wirelessly.

3. Specifications

Parameter	Value
Processor	Dual-core Xtensa LX6 @ 240MHz
RAM	520KB SRAM + additional PSRAM
Wi-Fi Standard	802.11 b/g/n
Bluetooth	Bluetooth 4.2 (Classic + BLE)
Operating Voltage	3.3V
GPIO Pins	34 (capable of PWM, ADC, I2C, SPI, UART)

Table 4.3-Specifications of ESP32 Module

3. Installation & Connection

- The ESP32 requires 3.3V power (not 5V like some Arduino boards).
- It connects via USB to UART for programming.
- Uses GPIO pins for input/output, I2C, SPI, UART for communication with sensors and other modules.

4. Output & Communication

- The ESP32 supports Wi-Fi to send and receive data over the internet.
- It can be used as a web server, MQTT client, or for cloud-based data logging.
- Bluetooth allows communication with smartphones, smart home devices, and BLE-based sensors.

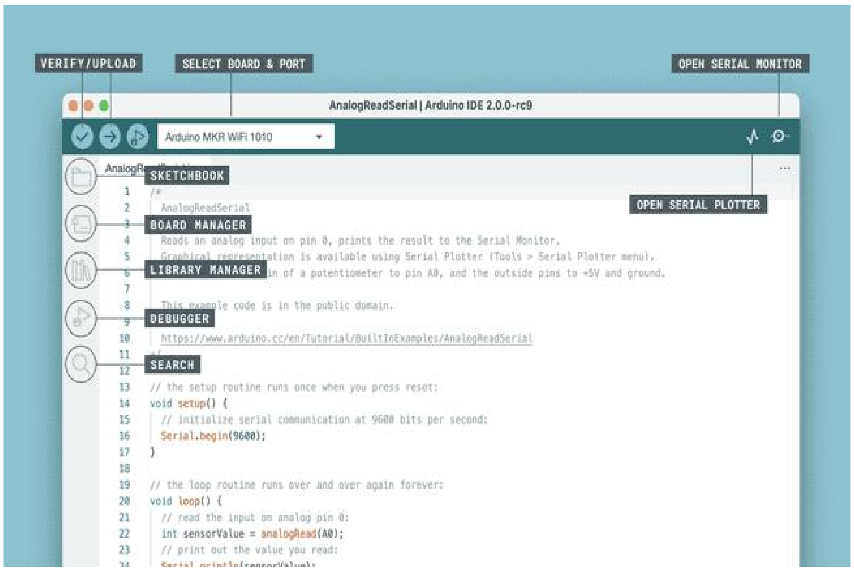
5. Applications

The ESP32 is widely used in:

- IoT smart city applications such as flood detection, smart lighting, and air quality monitoring.
- Home automation for smart switches, voice assistants, and security systems.
- Wearable technology for health tracking and remote monitoring.

The ESP32 is a low-cost, high-performance microcontroller ideal for advanced IoT applications that require wireless communication, real-time processing, and energy efficiency.

4.2 SOFTWARE TOOL:



```
/*
  AnalogReadSerial

  Reads an analog input on pin 0, prints the result to the Serial Monitor.
  Graphical representation is available using Serial Plotter (Tools > Serial Plotter menu).
                  in of a potentiometer to pin A0, and the outside pins to +5V and ground.

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/AnalogReadSerial
*/

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
```

Fig 4.10 Arduino IDE

4.2.1 ARDUINO IDE VERSION 2.3.4:

The Arduino Integrated Development Environment (IDE) is the software environment used to write, compile, and upload code to Arduino boards. Here are some key details about the Arduino IDE:

1. Multi-Platform Support: The Arduino IDE is compatible with major operating systems such as Windows, macOS, and Linux, allowing developers to seamlessly work across different platforms.
2. Intuitive User Interface: Designed for ease of use, the IDE features a clean and simple interface, including a code editor, serial monitor, and built-in library manager, making it accessible for both beginners and experts.
3. Powerful Code Editor: Users can write Arduino programs (sketches) using a simplified version of C/C++. Features like syntax highlighting, auto-completion, and code folding enhance the coding experience and reduce errors.

4. Library Management: The built-in library manager provides access to numerous pre-written libraries, allowing users to integrate various sensors, communication modules, and hardware components effortlessly.
5. Sketch Compilation & Uploading: The IDE translates Arduino sketches into machine code and uploads them directly to the microcontroller via USB. Users can select the appropriate board and communication port for seamless deployment. Serial Communication & Debugging: The integrated serial monitor enables real-time communication with the Arduino board, facilitating debugging, sensor data monitoring, and interaction with the microcontroller.
6. Preloaded Example Programs: A collection of ready-to-use examples demonstrates various functionalities, including sensor interfacing, PWM control, serial communication, and IoT applications, helping users get started quickly.
7. Expandable & Customizable: The IDE supports third-party plugins, custom libraries, and additional board definitions, making it adaptable to various development needs. Users can extend its capabilities to support additional hardware and software tools.
8. Open-Source Ecosystem: As an open-source platform, the Arduino IDE is freely available under the GNU General Public License (GPL). This encourages collaboration, innovation, and continuous improvement by the global developer community.

The Arduino IDE offers a feature-rich yet straightforward development environment for building and deploying Arduino-based projects. Its flexibility, ease of use, and strong community support make it a preferred choice for students, hobbyists, educators, and professionals in the field of embedded systems and IoT.

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION:

In this project, The implementation of the proposed smart city system involves integrating multiple IoT-based solutions using Arduino and ESP32 microcontrollers to address urban challenges such as flood detection, electrical hazard prevention, smart street lighting, and air quality monitoring. Each subsystem is designed to function independently while contributing to the overall system through real-time data sharing and centralized management via the ThingSpeak IoT cloud platform. The hardware setup includes water level sensors for flood detection, a continuity sensor for electrical hazard prevention, LDR sensors for smart street lighting, and DHT11 and MQ135 sensors for air quality monitoring. The Arduino microcontroller processes sensor data and controls alert mechanisms, while the ESP32 microcontroller transmits the collected data to the ThingSpeak cloud for storage and analysis.

5.1.1 Smart City Setup:

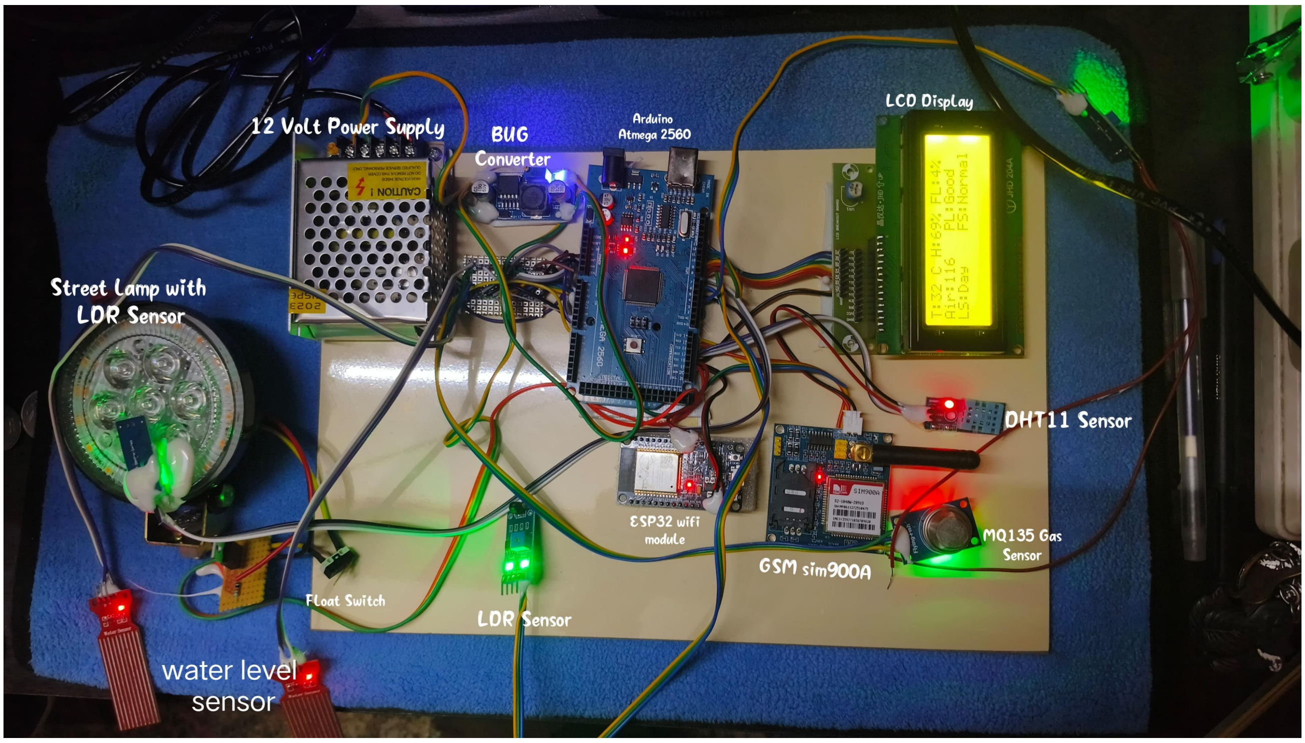


Fig 5.1 Smart City Setup

The **Smart City Setup** integrates multiple modules and sensors to ensure real-time monitoring and automated responses for urban safety and efficiency. The setup consists of:

- The two-tier flood detection mechanism was implemented by positioning water level sensors at different heights. The first sensor alerts officials via notifications when initial flooding is detected, while the second triggers SMS alerts to residents using a GSM module in critical situations.
- For electrical safety, the continuity sensor detects live wires submerged in water and automatically shuts down power in affected areas.
- The smart street lighting system was implemented with LDR sensors to control lights based on ambient light levels. A fault detection mechanism was added to identify malfunctioning lights and their locations.
- Environmental monitoring was achieved by integrating DHT11 and MQ135 sensors to measure temperature, humidity, and air quality. All subsystems were tested individually before being integrated into a unified platform.

5.1.2 Flood DetectionTesting:

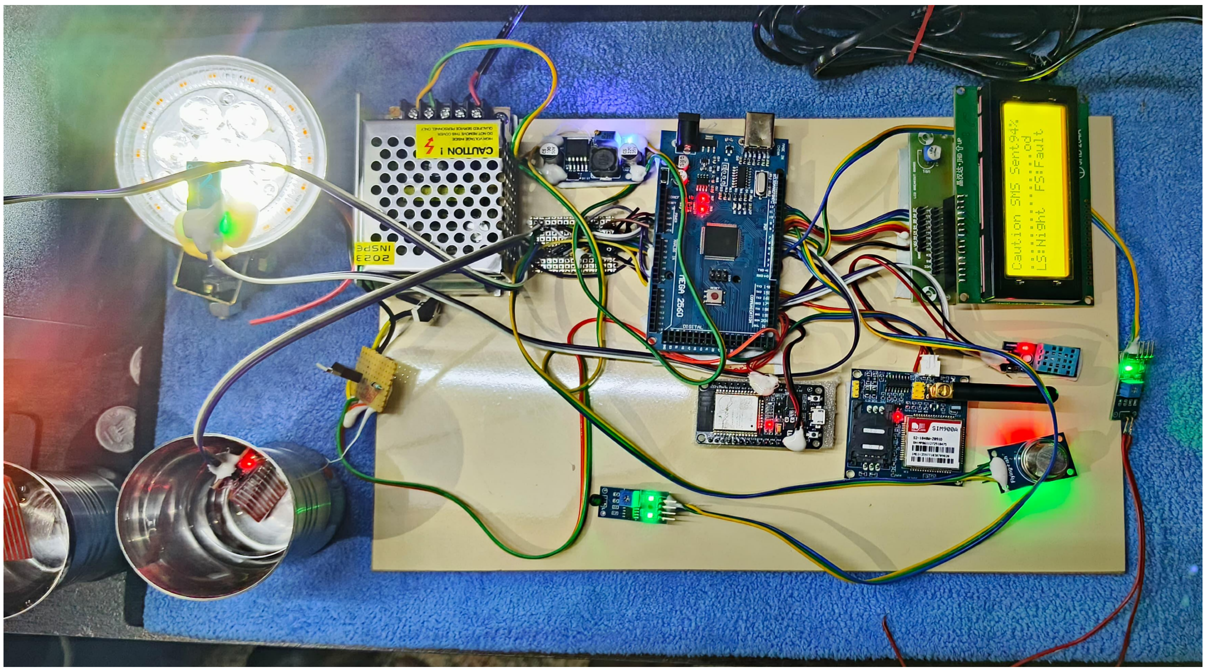


Fig 5.2 Flood Detection Testing

The water sensor is placed in varying water levels to test its accuracy and responsiveness. The ESP32 detects the water presence and transmits the data to the receiver. Upon detecting a flood condition, the GSM module sends an SMS alert to the authorities. The system is monitored to ensure consistent and reliable performance.

5.1.3 Electrical Hazard Prevention Testing

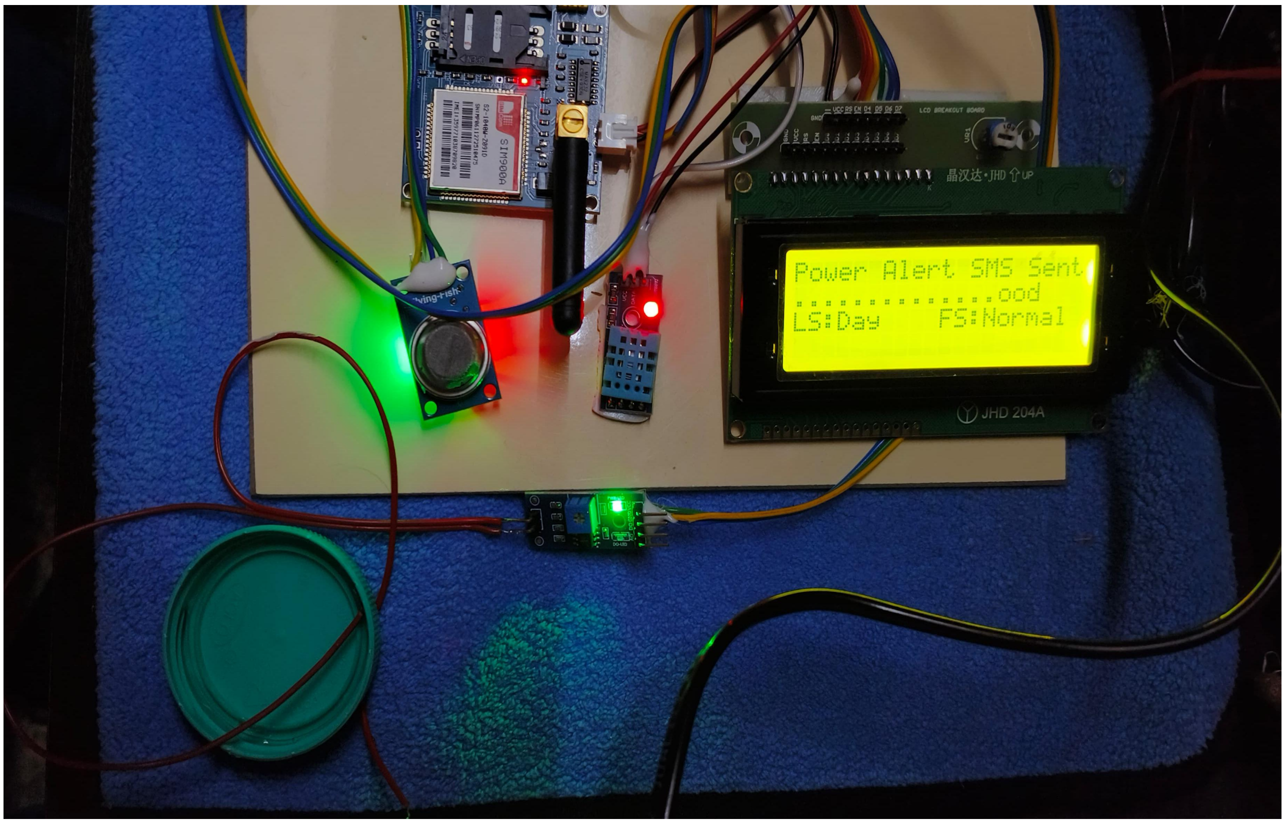


Fig 5.3 Electric Hazard Prevention Testing

The continuity sensor is tested by creating open and closed circuit conditions to simulate faults. The system detects breaks or continuity loss and displays the fault status on the LCD. Simultaneously, the GSM module sends an alert message when a hazard is detected. Multiple tests are conducted to confirm reliable fault detection.

5.1.4 Street Light Monitoring Testing

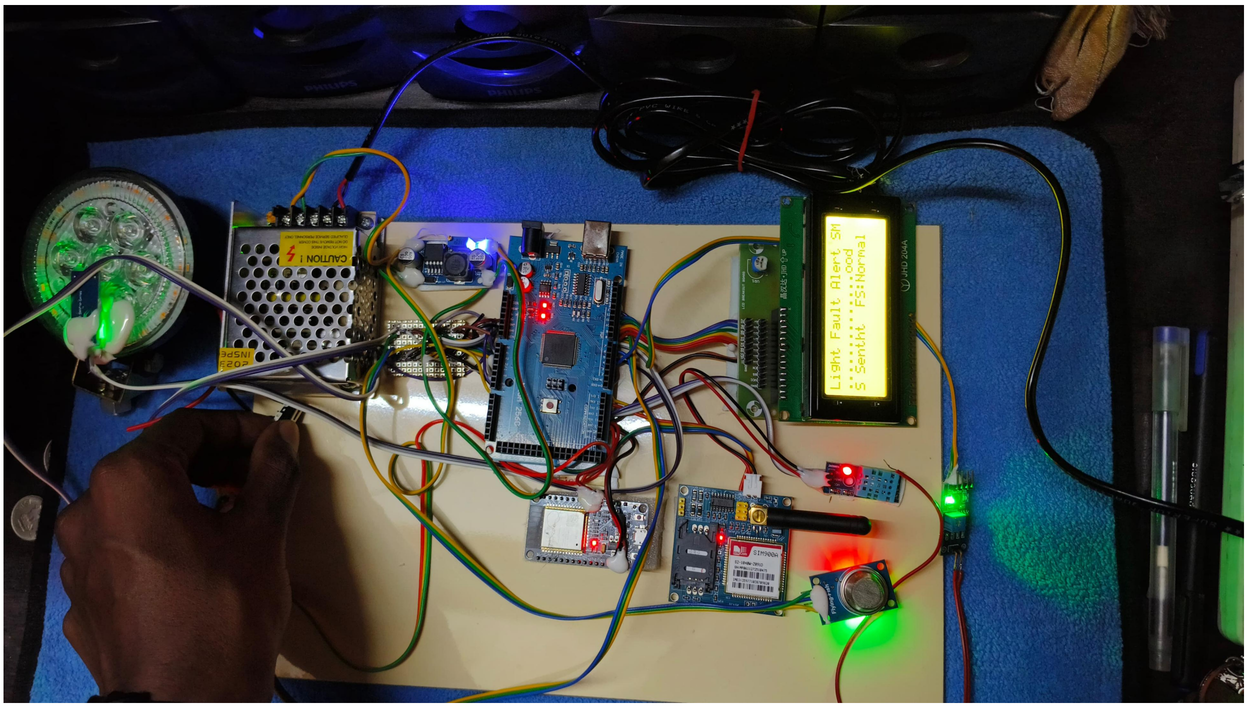


Fig 5.4 Street Light Monitoring Testing

The LDR sensor is tested by exposing it to different light intensities. During low light, the street light is automatically activated, and it turns off in brighter conditions. The status is displayed on the LCD screen.When float switch is Applied to the 2nd LDR the light has made faultThe message will be sent to the officals

5.1.5 Air Quality Monitoring Testing:

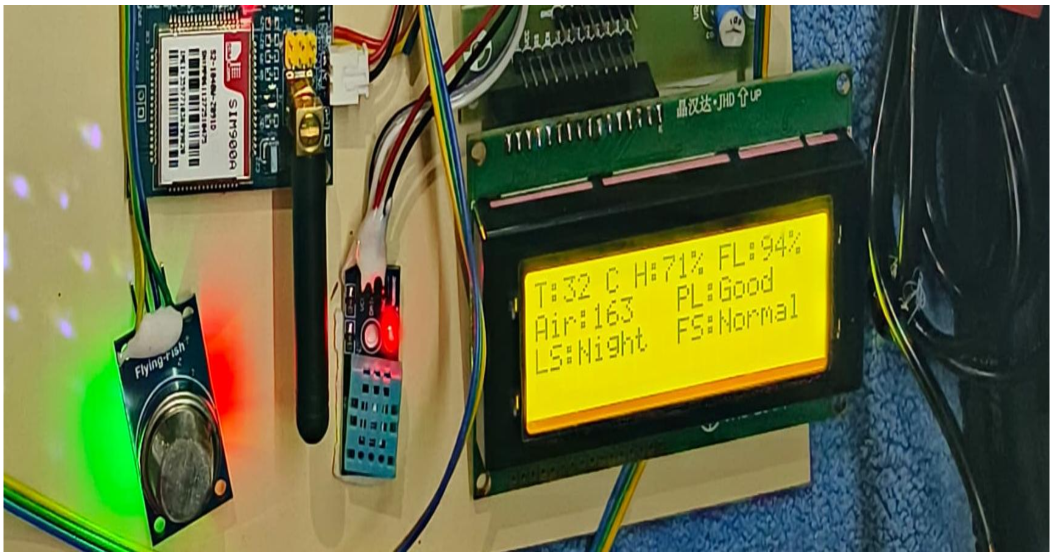


Fig 5.5 Air Quality Monitoring Testing

The LDR sensor is tested by exposing it to different light intensities. During low light, the street light is automatically activated, and it turns off in brighter conditions. The status is displayed on the LCD screen. The system is tested repeatedly to ensure proper light sensitivity and switching accuracy.

5.2 PROGRAM CODE:

5.2.1 ARDUINO MEGA CODE

The Arduino Mega 2560 is a powerful microcontroller board suitable for handling multiple sensors and modules, making it an ideal choice for smart city applications. Below is an overview of different components used in the project, along with their functionalities.

1. Water Level Sensor

   - The water level sensor detects water height to prevent flooding.The sensor is connected to an analog pin (A0) for continuous monitoring
   - Read the analog value using analogRead().Convert the raw data to percentage levels using the map() function.Print the water level percentage to the serial monitor for debugging.
   - If the water level exceeds a threshold, trigger a servo motor to open/close the dam gate and send an alert.

2. Electrical Hazard Detection (Continuity Sensor)

   - This sensor detects water leakage near electrical poles and shuts down power to prevent electrocution risks.The Arduino Mega processes the sensor data and deactivates the power relay when water presence is detected.
   - The sensor is connected to an analog pin (A1) for input readings.If water is detected, activate a relay module to cut off power.An alert is sent to authorities via GSM communication.

3. Air Quality Monitoring (MQ135 Sensor & DHT11)

- The MQ135 sensor measures air pollutants to monitor environmental conditions.The DHT11 sensor collects temperature and humidity data.
- Use the MQ135.h and DHT.h libraries to collect air quality and weather data.Send real-time sensor data to a cloud platform via ESP32 for analysis.Display readings on an LCD screen for local monitoring.

4. LCD Display Integration

- The 16x2 LCD provides real-time feedback on flood levels, air quality, and electrical hazards.
- Use the LiquidCrystal library to interface the LCD.Display sensor readings such as water levels, temperature, and air quality index.Periodically refresh the screen for updated information.

5. GSM Communication for Alerts

- A SIM800L GSM module is used to send alerts when hazards are detected.
- Use the SoftwareSerial.h library for GSM communication.
- If water levels or air quality exceed safe limits, send an SMS alert to authorities.Ensure proper signal strength by using a 5V power supply.

By integrating Arduino Mega with multiple sensors and communication modules, this system offers real-time monitoring and automated safety measures for smart cities. The scalability and efficiency of this design make it a cost-effective and reliable solution for urban hazard prevention.

```
#include <LiquidCrystal.h>
#include <DHT.h>
// Define the pins for the sensors
const int mq135Pin = A8;
const int dhtPin = 4;
const int waterSensor1Pin = A0;
const int waterSensor2Pin = A1;
const int ledPin = 13;        // Pin for onboard LED
// Location coordinates
const double incidentLatitude = 13.359275818903855;
const double incidentLongitude = 80.14081621102996;
const double evacuationLatitude = 13.120906252110908;
const double evacuationLongitude = 80.10190809383144;

// LCD Setup
LiquidCrystal lcd(8, 9, 10, 11, 12, 13);
DHT dht(dhtPin, DHT11);
void setup() {
  Serial.begin(9600);       // For debugging
  Serial1.begin(9600);      // For communication with ESP32 (changed to Serial1)
  Serial3.begin(9600);      // For GSM module communication
  lcd.begin(20, 4);         // Initialize the LCD immediately
  lcd.setCursor(0, 0);
  lcd.print("Initializing...");
  delay(100);
  dht.begin();              // Initialize the DHT sensor
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH); // Turn on LED initially
  delay(20000);             // Wait for GSM module to register
  // Clear the LCD after initialization
  lcd.clear();
  // Check GSM module connection
  Serial.println("Checking GSM module connection...");
  Serial3.println("AT");
  delay(1000);
  while (Serial3.available()) {
    Serial.print(Serial3.readString());
  }
  // Set SMS mode to text
  Serial.println("Setting SMS mode to text...");
  Serial3.println("AT+CMGF=1");
}

void loop() {
    // Read temperature and humidity from DHT11
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();

    if (isnan(temperature) || isnan(humidity)) {
        Serial.println("Failed to read from DHT sensor!");
        lcd.setCursor(0, 0);
        lcd.print("DHT Error");
        delay(1000);
        return;
    }
    // Read air quality and water levels
    int airQuality = analogRead(mq135Pin);
    int waterLevel1 = analogRead(waterSensor1Pin);
    int waterLevel2 = analogRead(waterSensor2Pin);

    // Compare water levels and send SMS alerts if necessary
    if (waterLevel1 > 500) { // Threshold for Water Sensor 1
        sendSMS("Officer", "Water level is increasing at:
```

```
https://www.google.com/maps/search/?api=1&query=" + String(incidentLatitude) + "," + String(incidentLongitude));
      lcd.setCursor(0,3);
      lcd.print("W1 High: Msg Sent");
      delay(500);
   }
   if (waterLevel1 > 500 && waterLevel2 > 500) { // Threshold for both Water Sensors
      sendSMS("Public", "Evacuate immediately from: https://www.google.com/maps/search/?api=1&query=" + String(evacuationLatitude) + "," + String(evacuationLongitude));
      lcd.setCursor(0,3);
      lcd.print("Evacuation Msg Sent");
      delay(500);
   }
   // Display data on the LCD
   lcd.setCursor(0,0);
   lcd.print("T: ");
   lcd.print(temperature);
   lcd.setCursor(10,0);
   lcd.print("H: ");
   lcd.print(humidity);
   lcd.setCursor(0,1);
   lcd.print("Air: ");
   lcd.print(airQuality);
   lcd.setCursor(0,2);
   lcd.print("W1: ");
   lcd.print(waterLevel1);
   lcd.setCursor(10,2);
```

```
  lcd.print("W2: ");
  lcd.print(waterLevel2);
  // Send data to ESP32 via Serial1
  String dataToSend = String(temperature) + "," + String(humidity) + "," +
String(airQuality) + "," +
                    String(waterLevel1) + "," + String(waterLevel2) + "\n";
  Serial1.print(dataToSend); // Send data to ESP32 via Serial1
  Serial.println("Data sent to ESP32: " + dataToSend); // Print sent data to serial
monitor
  delay(1000); // Wait for a second before sending again
}
void sendSMS(String recipient, String message) {
  Serial.println("Sending SMS...");
  String phoneNumber;
  if (recipient == "Officer") {
    phoneNumber = "09080192655"; // Replace with officer's number
  } else if (recipient == "Public") {
    phoneNumber = "+919080192655"; // Replace with public's number or a
broadcast number
  }
  Serial3.println("AT+CMGS=\"" + phoneNumber + "\"");
  delay(1000);
  while (Serial3.available()) {
    Serial.print(Serial3.readString());
  }
  Serial3.println(message);
Serial3.println((char)26);
delay (100);
}
```

5.2.2 ESP32 CODE

The ESP32 microcontroller is a powerful and versatile platform for IoT applications, featuring built-in Wi-Fi and Bluetooth capabilities. It is well-suited for real-time monitoring, data collection, and cloud integration in smart city projects. This section details the implementation of ESP32 in the project.

1. ESP32 Microcontroller: The ESP32 is used as the core processing unit for collecting sensor data and transmitting it over the internet using Wi-Fi or GSM connectivity. It supports multiple sensor interfaces, including analog, digital, I2C, and SPI, making it ideal for IoT applications.

2. Sensor Integration: The system includes various sensors such as Water level sensors for flood detection, Continuity sensors for electrical hazard prevention, LDR sensors for smart street lighting control, DHT11 & MQ135 sensors for air quality monitoring, These sensors are connected to the ESP32's GPIO pins to collect real-time environmental data.

3. Data Transmission and Processing: The ESP32 transmits collected data to the cloud using Wi-Fi or a GSM module, depending on the network availability.Data is sent to ThingSpeak or another cloud storage platform for real-time monitoring and analytics.

4. Power Management: The ESP32 operates efficiently with a regulated power supply to ensure stable performance.Sleep modes and power optimization techniques are implemented to extend battery life in low-power applications.

5. Real-Time Alerts and Notifications: In case of potential hazards (e.g., rising water levels or electrical faults), the ESP32 triggers automated alerts via SMS (GSM module) or email notifications through cloud services.A warning system is activated to inform local authorities and residents about the emergency.

6. ESP32 Programming and Libraries: The Arduino IDE is used for programming the ESP32, with libraries such as WiFi.h for internet connectivity, DHT.h for temperature and humidity readings, MQ135.h for air quality sensing,ThingSpeak.h for cloud data integration, SoftwareSerial.h for GSM communication.

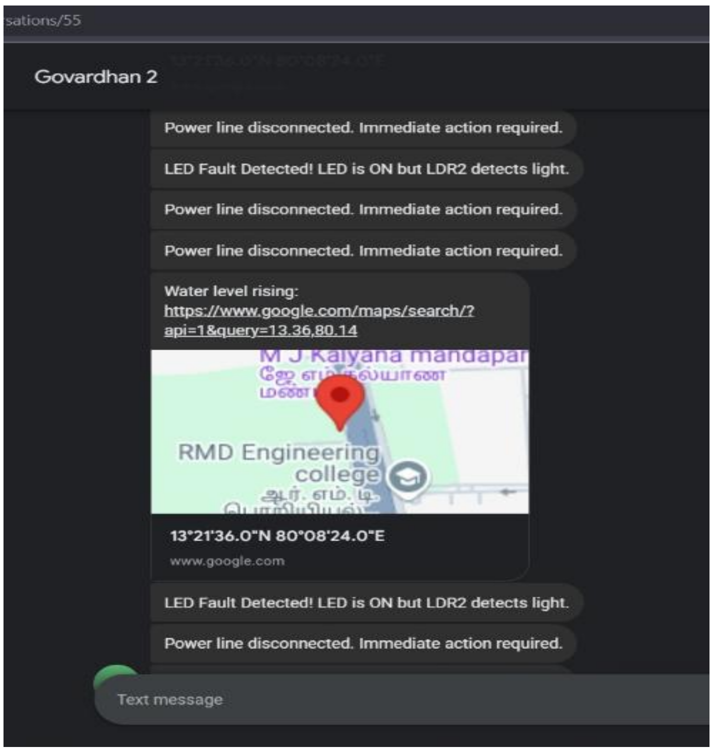
By utilizing the ESP32 in this project, we enable a cost-effective, real-time monitoring system that enhances urban safety and efficiency.

```
#include <WiFi.h>
#include <HTTPClient.h>
// Wi-Fi credentials
const char* ssid = "Project";          // Replace with your Wi-Fi SSID
const char* password = "1234567890";   // Replace with your Wi-Fi password
// ThingSpeak credentials
const char* thingSpeakAPIKey = "4ZM3QI25EMIOO5PA"; // Replace with your
ThingSpeak API key
void setup() {
  Serial.begin(9600);        // Initialize Serial Monitor for debugging
  Serial2.begin(9600, SERIAL_8N1, 16, 17); // Initialize Serial2 for
communication with Arduino Mega (GPIO16 as RX2, GPIO17 as TX2)
  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("-> Connecting to WiFi...");
  }
  Serial.println("-> Connected to WiFi");
}
void loop() {
  if (Serial2.available()) {
```

```
  String data = Serial2.readStringUntil('\n'); // Read data from Arduino Mega
  if (data.length() > 0) {
   Serial.println("Received data: " + data);
   // Parse the received data
   int commaIndex1 = data.indexOf(',');
   int commaIndex2 = data.indexOf(',', commaIndex1 + 1);
   int commaIndex3 = data.indexOf(',', commaIndex2 + 1);
   int commaIndex4 = data.indexOf(',', commaIndex3 + 1);
   int commaIndex5 = data.indexOf(',', commaIndex4 + 1);
   if (commaIndex1 != -1 && commaIndex2 != -1 && commaIndex3 != -1 &&
commaIndex4 != -1 && commaIndex5 != -1) {
    float temperature = data.substring(0, commaIndex1).toFloat();
    float humidity = data.substring(commaIndex1 + 1, commaIndex2).toFloat();
    int airQuality = data.substring(commaIndex2 + 1, commaIndex3).toInt();
    int waterLevel1 = data.substring(commaIndex3 + 1, commaIndex4).toInt();
    int waterLevel2 = data.substring(commaIndex4 + 1, commaIndex5).toInt();
    int soilMoistureState = data.substring(commaIndex5 + 1).toInt();
    // Print parsed values to serial monitor
    Serial.println("Parsed Data:");
    Serial.println("Temperature: " + String(temperature));
    Serial.println("Humidity: " + String(humidity));
    Serial.println("Air Quality: " + String(airQuality));
    Serial.println("Water Level W1: " + String(waterLevel1));
    Serial.println("Water Level W2: " + String(waterLevel2));
    Serial.println("Soil Moisture State: " + String(soilMoistureState));
    // Upload parsed data to ThingSpeak
    uploadToThingSpeak(temperature, humidity, airQuality, waterLevel1,
waterLevel2, soilMoistureState);
   } else {
        Serial.println("Error parsing received data.");
      }
    }
    delay(1000); // Wait before checking again
  }
}
void uploadToThingSpeak(float temperature, float humidity, int airQuality, int
waterLevel1, int waterLevel2, int soilMoistureState) {
  HTTPClient http;
  // Construct the URL for ThingSpeak API
  String url = "http://api.thingspeak.com/update?api_key=" +
String(thingSpeakAPIKey) +
               "&field1=" + String(temperature) +
               "&field2=" + String(humidity) +
               "&field3=" + String(airQuality) +
               "&field4=" + String(waterLevel1) +
               "&field5=" + String(waterLevel2) +
               "&field6=" + String(soilMoistureState);
  http.begin(url); // Initialize HTTP client with the URL
  int httpResponseCode = http.GET(); // Send GET request
  if (httpResponseCode > 0) {
    Serial.println("-> Data uploaded to ThingSpeak successfully.");
    Serial.println("HTTP Response Code: " + String(httpResponseCode));
  } else {
    Serial.println("-> Error uploading data to ThingSpeak.");
    Serial.println("HTTP Response Code: " + String(httpResponseCode));
  }
  http.end(); // Close HTTP connection
```

CHAPTER 6
RESULTS

The proposed system is fully automated, with the readings being recorded in a controlled environment, which may vary slightly in real-world conditions. The following images display the outputs generated by the system, including mobile alerts, data displayed on the serial monitor, and the activation of streetlights and hazard detection systems.


sations/55

Govardhan 2

Power line disconnected. Immediate action required.

LED Fault Detected! LED is ON but LDR2 detects light.

Power line disconnected. Immediate action required.

Power line disconnected. Immediate action required.

Water level rising:
https://www.google.com/maps/search/?api=1&query=13.36,80.14

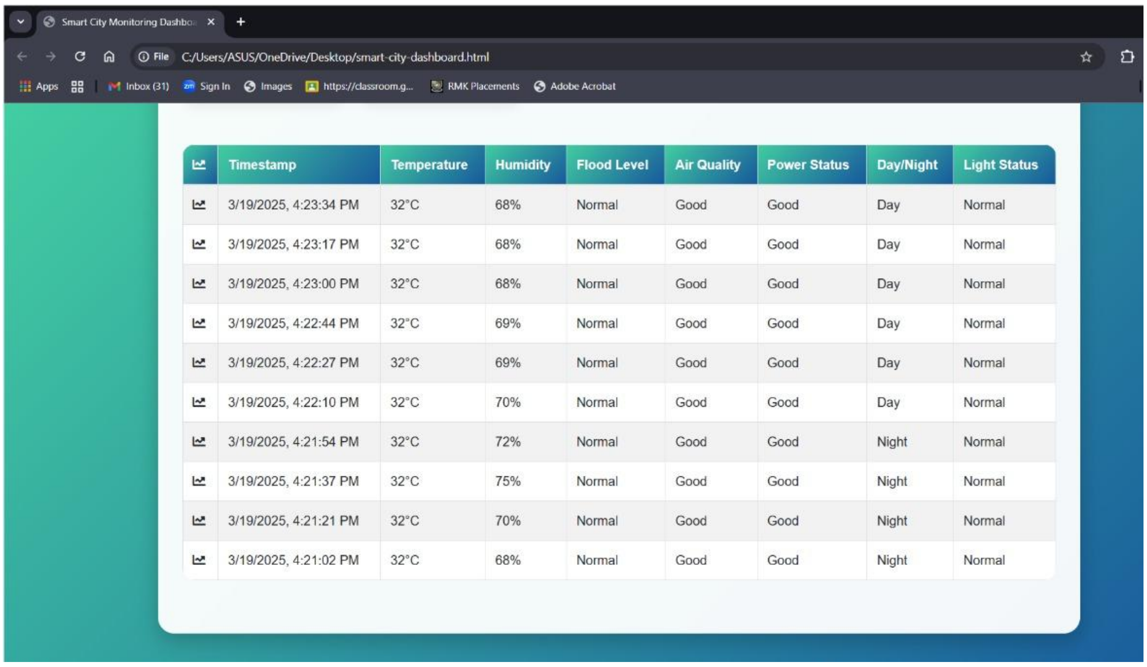
13°21'36.0"N 80°08'24.0"E
www.google.com

LED Fault Detected! LED is ON but LDR2 detects light.

Power line disconnected. Immediate action required.

Fig 6.1 GSM Module Alert Notification

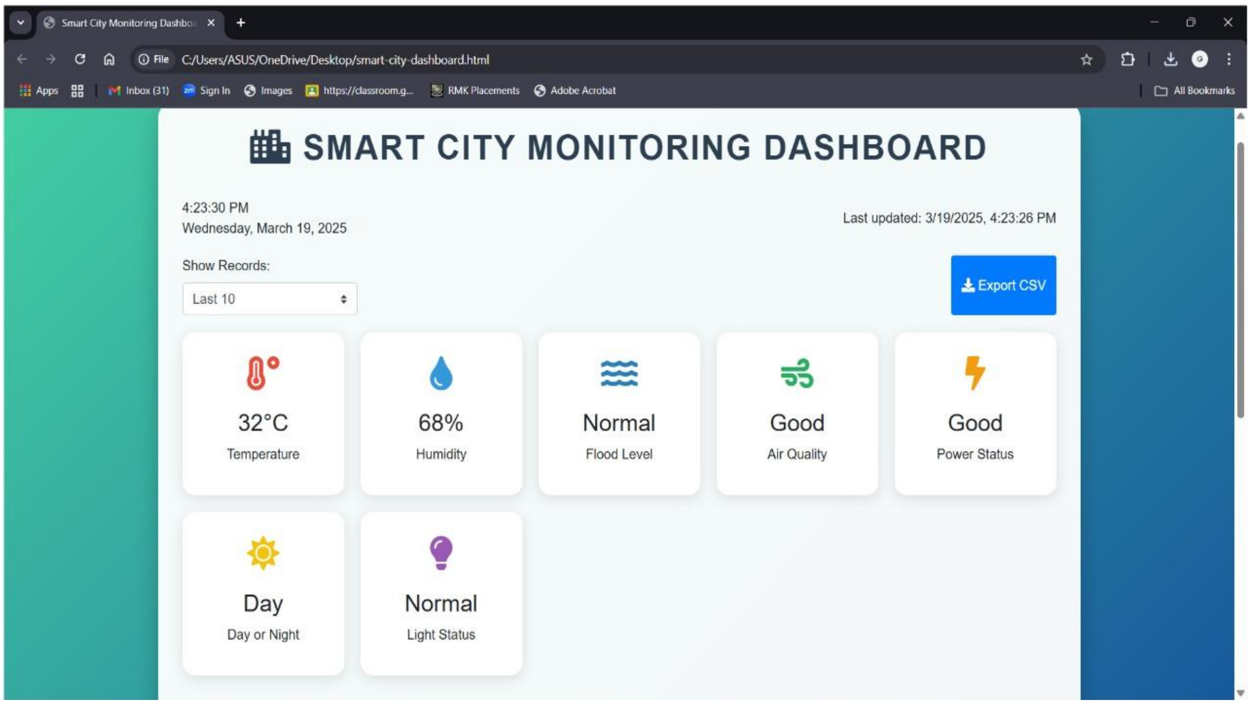
The implemented system successfully performs real-time monitoring using IoT technology. The collected data from various sensors, including flood detection, air quality monitoring, electrical hazard detection, and streetlight control, is transmitted to a web-based platform. The system displays live data readings on a webpage, providing remote access for continuous monitoring. This real-time visualization allows users to track environmental parameters and receive instant alerts, enhancing the efficiency and reliability of the smart city setup.


Smart City Monitoring Dashbo
File C:/Users/ASUS/OneDrive/Desktop/smart-city-dashboard.html
Apps
Inbox (31)
Sign In
Images
https://classroom.g...
RMK Placements
Adobe Acrobat

📈	Timestamp	Temperature	Humidity	Flood Level	Air Quality	Power Status	Day/Night	Light Status
📈	3/19/2025, 4:23:34 PM	32°C	68%	Normal	Good	Good	Day	Normal
📈	3/19/2025, 4:23:17 PM	32°C	68%	Normal	Good	Good	Day	Normal
📈	3/19/2025, 4:23:00 PM	32°C	68%	Normal	Good	Good	Day	Normal
📈	3/19/2025, 4:22:44 PM	32°C	69%	Normal	Good	Good	Day	Normal
📈	3/19/2025, 4:22:27 PM	32°C	69%	Normal	Good	Good	Day	Normal
📈	3/19/2025, 4:22:10 PM	32°C	70%	Normal	Good	Good	Day	Normal
📈	3/19/2025, 4:21:54 PM	32°C	72%	Normal	Good	Good	Night	Normal
📈	3/19/2025, 4:21:37 PM	32°C	75%	Normal	Good	Good	Night	Normal
📈	3/19/2025, 4:21:21 PM	32°C	70%	Normal	Good	Good	Night	Normal
📈	3/19/2025, 4:21:02 PM	32°C	68%	Normal	Good	Good	Night	Normal

Fig 6.2 Real Time Monitoring Using Iot

The implemented system effectively displays real-time flood status on the IoT dashboard. Before detecting a flood, the system shows the "Flood Level: Normal" status, indicating safe water levels.



SMART CITY MONITORING DASHBOARD

4:23:30 PM
Wednesday, March 19, 2025

Last updated: 3/19/2025, 4:23:26 PM

Show Records: Last 10

32°C
Temperature

68%
Humidity

Normal
Flood Level

Good
Air Quality

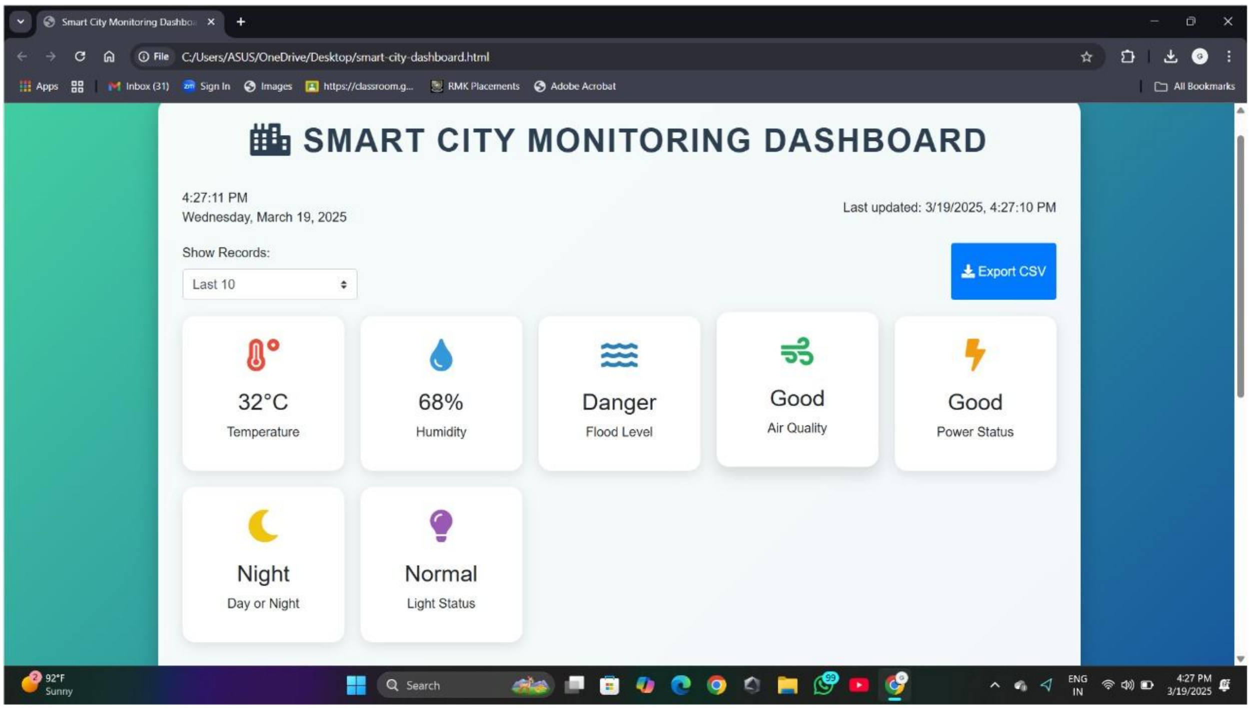
Good
Power Status

Day
Day or Night

Normal
Light Status

Fig 6.3 Before Flood Detection in City

However, when the water level crosses the predefined threshold, the system immediately updates the status to "Flood Level: Danger", signaling a potential flooding event. This real-time status change is reflected on the webpage, allowing for quick identification of hazardous conditions and enabling timely preventive measures.



SMART CITY MONITORING DASHBOARD

4:27:11 PM
Wednesday, March 19, 2025

Last updated: 3/19/2025, 4:27:10 PM

Show Records: Last 10

Export CSV

32°C Temperature	68% Humidity	Danger Flood Level	Good Air Quality	Good Power Status
Night Day or Night	Normal Light Status			

Fig 6.4 After The Flood Detection in City

CHAPTER 7
CONCLUSION

7.CONCLUSION:

In conclusion, The proposed smart reservoir management system integrates IoT-driven solutions to enhance flood control, public safety, and efficient water management.

- **Enhanced Flood Control**: The system efficiently monitors water inflow and outflow, preventing sudden surges and minimizing flood risks in urban and residential areas.
- **Real-Time Hazard Detection**: By integrating continuity sensors, the system identifies electrical hazards in flood-prone regions and automatically shuts down power to prevent accidents.
- **Early Warning System**: The GSM module provides timely alerts to authorities and residents, allowing them to take necessary precautions during emergencies.
- **Smart Street Lighting Management**: The automated lighting system optimizes energy consumption and detects faults in real-time, ensuring uninterrupted functionality.
- **Air Quality Monitoring**: The system continuously tracks environmental parameters like temperature, humidity, and pollutant levels, aiding in proactive urban planning and public health measures.

By integrating these smart technologies, the proposed system enhances urban safety, optimizes resource management, and strengthens disaster preparedness in smart cities.

7.1 FUTURE SCOPE:

1. Expansion to Additional Urban Challenges:
   - Future enhancements will include the integration of IoT solutions for traffic management, waste disposal monitoring, and public transportation optimization to create a more interconnected smart city ecosystem.
2. Integration of AI and Machine Learning:
   - Advanced AI-driven analytics can be implemented to predict flood risks, electrical hazards, and air quality fluctuations, enabling proactive decision-making and automated responses.
3. Enhanced Data Security and Privacy:
   - Strengthening encryption methods and implementing strict access control mechanisms will ensure the protection of sensitive urban data against cyber threats.
4. Scalable and Modular Architecture:
   - A modular design will allow easy integration of new components and features, ensuring adaptability as urban landscapes grow and evolve.
5. User-Friendly Dashboard Interface:
   - Developing an interactive dashboard for city officials and residents will improve accessibility to real-time data, enhancing engagement and informed decision-making.
6. Sustainability and Energy Optimization:
   - Incorporating renewable energy sources such as solar-powered IoT nodes can further enhance the system’s sustainability and reduce operational costs.

REFERENCES

1. Alam, M., & Hasan, R. (2022). IoT-Enabled Smart City Infrastructure for Disaster Management and Hazard Prevention. Proceedings of the International Conference on Future Technologies, pp. 312-320.
2. Chen, L. & Wong, P. (2022). 'Wireless Sensor Network-Based Smart Flood Monitoring System with GSM Integration', Proceedings of the International Conference on IoT and Smart Cities, Singapore, pp. 143-152.
3. Gupta, P., Sharma, A., & Singh, R. (2020). 'IoT-Based Flood Detection and Early Warning System', International Journal of Advanced Research in Electronics and Communication Engineering, Vol. 9, No. 3, pp. 102-108.
4. Kumar, A.(2018). IoT-Based Smart Flood Monitoring and Early Warning System. International Journal of Smart Sensor Technologies, Vol. 15(2), pp. 78-85.
5. Lee, H., & Park, J. (2020). Sensor Networks for Smart Cities: Flood and Air Pollution Detection Using IoT. Journal of Advanced Sensor Technologies, Vol. 8, pp. 199-210.
6. Patel, J. & Desai, A. (2021). 'Integration of IoT and GSM for Electrical Hazard Prevention in Smart Cities', Smart Infrastructure Journal, Vol. 6, No. 4, pp. 87-94.
7. Singh, P., & Sharma, R. (2020). Development of an IoT-Based Air Quality Monitoring System for Smart Cities. Journal of Environmental Monitoring, Vol. 23, pp. 112-120.
8. Tan, W., & Zhang, X. (2019). A Review on IoT-Based Smart City Solutions: Flood Detection, Air Quality Monitoring, and Hazard Prevention. International Journal of Urban Technologies, Vol. 12(3), pp. 120-132.