CHAPTER 1
INTRODUCTION

1.1 History

The history of IoT-based flood monitoring and alerting systems has evolved significantly alongside advancements in technology. Initially, flood monitoring relied on traditional methods like manual weather stations, river gauges, and satellite data, which provided limited real-time insights. In the 1990s, IoT technology emerged, leveraging sensors, wireless communication, and cloud computing to enable continuous, remote monitoring of environmental conditions.

By the early 2000s, IoT sensors were integrated into flood management systems, offering real-time data on water levels, rainfall, and river flow. These systems provided more accurate monitoring and faster responses compared to previous methods. In the 2010s, IoT-based solutions gained momentum, particularly in smart cities, where networks of sensors, drones, and mobile applications helped provide timely alerts and improve decision-making.

AI and machine learning further revolutionized flood forecasting and response systems by predicting flood events based on historical and real-time data. IoT systems could now automatically trigger alerts to emergency services and residents when flood risks were detected. Additionally, IoT integration with weather stations, satellite data, and climate change models provided a more comprehensive approach to flood risk management.

Today, IoT-based flood monitoring systems combine big data analytics, real-time alerts, and automated responses. With the rise of cloud computing, predictive analytics, and enhanced communication technologies, these systems are now more reliable, accurate, and interconnected. The focus is on building resilience against climate change and natural disasters, empowering communities with better flood prevention, forecasting, and emergency response capabilities

1.2 Project overview

The **IoT-Based Flood Monitoring and Alerting System** uses IoT technology to monitor flood-prone areas in real time. By deploying sensors to track water levels, rainfall, and soil moisture, the system collects and transmits data wirelessly to a central platform. AI analyzes the data to predict flood risks and trigger automated alerts to authorities and residents via mobile apps or SMS. This system enables early detection, timely flood warnings, and efficient management, reducing potential damage and improving safety during floods. It offers a scalable, real-time solution for flood monitoring and disaster response.

The **IoT-Based Flood Monitoring and Alerting System** uses IoT technology to provide real-time monitoring of water levels, rainfall, and other flood indicators, offering early warnings to reduce flood impact on communities.

Objective

The project aims to:

- Continuously monitor flood-prone areas.
- Provide timely alerts for flood risks.
- Enhance flood management through automated warnings and data analytics.

System Components

1. **Sensors**: Water level, rainfall, and soil moisture sensors to collect real-time data.
2. **Communication Network**: Wireless technologies (Wi-Fi, Lora WAN) transmit data to central servers.
3. **Data Analytics**: Cloud-based AI tools analyse data to predict flood events.
4. **Alerting System**: Automated alerts via mobile apps, SMS, or sirens to notify authorities and residents.
5. **Decision Support Tools**: Provide data for flood management and emergency planning.

Working Mechanism

- **Data Collection**: Sensors gather real-time environmental data.
- **Data Analysis**: AI algorithms predict flood risks.
- **Alert Generation**: Automated alerts notify stakeholders of potential floods.
- **Flood Management**: Authorities can act based on real-time data for evacuation or resource deployment.

Benefits

1. **Early Detection**: Timely intervention to save lives and property.
2. **Real-Time Monitoring**: Continuous data improves decision-making.
3. **Automation**: Reduces response times through automated alerts.
4. **Scalability**: Can be expanded to cover large areas.
5. **Smart City Integration**: Enhances urban flood management.

Challenges

1. **Power Supply**: Remote areas may need solar-powered sensors.
2. **Data Accuracy**: Sensor calibration and maintenance are crucial.
3. **Connectivity**: Reliable communication in remote areas is needed.
4. **Cost**: High initial setup costs but long-term benefits.

Future Trends

- **Integration with Weather Forecasting**: Improved flood predictions.
- **Global Connectivity**: Enhanced data transmission through 5G and satellites.
- **Blockchain**: For securing flood data integrity.

CHAPTER 2
LITERATURE SURVEY

Literature Survey: IOT based flood monitoring and Alerting system

Floods are one of the most devastating natural disasters worldwide. Traditional flood monitoring systems often lack real-time monitoring, predictive analytics, and automated responses. The integration of **Internet of Things (IoT)** technology in flood monitoring and alerting systems has improved early detection, monitoring, and response. IoT-based systems use a network of sensors, communication technologies, and data analytics to predict floods, alert authorities, and minimize damage.

1. IoT Sensors in Flood Monitoring

IoT sensors are key components in real-time flood monitoring systems. These sensors measure variables like water levels, rainfall, and soil moisture. According to **Liu et al. (2020)**, water level sensors (e.g., ultrasonic sensors) are commonly used for real-time flood monitoring. Other studies by **Mastorakis et al. (2018)** emphasize the use of rain gauges and soil moisture sensors for more accurate flood risk assessment.

Advantages:

- Low-cost and scalable
- Continuous data collection
- Accurate environmental monitoring

Challenges:

- Sensor calibration and maintenance
- Environmental factors affecting sensor readings (e.g., temperature, debris)

2. Communication Technologies

The effectiveness of IoT-based flood monitoring relies on the ability to transmit data in real-time. Various communication technologies such as **LoRaWAN**, **ZigBee**, **Wi-Fi**, and **GSM** are used for data transmission. According to **Zhang et al. (2018)**, **LoRaWAN** is particularly suitable due to its long-range capabilities and low power consumption, making it ideal for remote flood-prone areas.

Challenges:

- Ensuring reliable connectivity in remote or urban flood-prone regions
- Interference and delays during extreme weather conditions

3. Data Analytics and Prediction Models

IoT-based systems rely on data analytics to predict floods accurately. Machine learning algorithms such as **Artificial Neural Networks (ANN)**, **Support Vector Machines (SVM)**, and **Random Forests (RF)** have been applied to forecast flood events based on historical and real-time sensor data. Studies by **Jadhav et al. (2019)** and **Ali et al. (2020)** demonstrate that combining sensor data with machine learning algorithms significantly improves flood prediction accuracy.

Advantages:

- Improved prediction of flood events
- Reduces human error in flood forecasting

Challenges:

- Data accuracy and consistency
- Complexity of flood prediction models

4. Alerting Mechanisms

An essential component of IoT-based flood monitoring is the ability to send timely alerts to authorities, residents, and emergency responders. **SMS**, **email**, and **mobile apps** are commonly used to deliver real-time alerts. According to **Thakur et al. (2017)**, early warning systems enabled by IoT help reduce human casualties and property damage by triggering timely evacuations and precautionary measures.

Challenges:

- False alarms or delayed alerts in some cases
- Public reliance on technology for evacuations and flood response

5. Applications of IoT-Based Flood Monitoring

Several applications of IoT-based flood monitoring systems have been explored in literature:

- **Early Warning Systems**: Used to monitor water bodies, detect rising water levels, and forecast floods in river basins or dam reservoirs (**Cheng et al., 2019**).

- **Urban Flood Management**: In smart cities, IoT systems integrate with urban infrastructure to predict and manage floods in real-time (**Sathya et al., 2021**).
- **Agricultural Flood Monitoring**: IoT-based flood monitoring systems are also applied in agriculture to monitor soil moisture and prevent crop damage from floods (**Mastorakis et al., 2018**).

6. Challenges in IoT-Based Flood Monitoring

Despite the significant advantages, several challenges remain in the widespread adoption of IoT-based flood monitoring systems:

- **Power Supply**: Remote areas may face challenges in providing a reliable power supply to sensors, with solutions like solar-powered sensors being commonly used (**Mourad et al., 2020**).
- **Data Accuracy**: Ensuring data reliability is critical for flood prediction. Sensors need regular calibration and maintenance to avoid malfunctioning.
- **Cost and Scalability**: While initial costs are high, the long-term benefits of IoT-based systems justify the investment. However, deploying these systems at a large scale in flood-prone areas can be costly (**Yuan et al., 2018**).

7. Future Trends

The future of IoT-based flood monitoring systems shows promise through advancements in:

- **Big Data and Cloud Computing**: Integration with big data platforms will improve data storage, processing, and analysis for real-time decision-making (**Zhang et al., 2021**).
- **Blockchain**: Securing flood-related data using **blockchain technology** ensures transparency, data integrity, and traceability in disaster management (**Huang et al., 2020**).
- **Drones and UAVs**: Drones are increasingly being used to gather flood data, especially in hard-to-reach areas. They offer real-time aerial surveillance and flood damage assessments (**Bhagat et al., 2020**).

CHAPTER 3
PROPOSED WORK

3.1 Attendance module

3.1.1 Introduction

Flooding is a major natural disaster that can cause significant loss of life and property. Traditional flood monitoring systems, such as manual gauges and weather stations, often lack real-time data collection, prediction capabilities, and automated alerting. The **IoT-based Flood Monitoring and Alerting System** aims to address these limitations by leveraging IoT sensors, communication networks, and data analytics to provide real-time flood monitoring and early alerts, ultimately enhancing disaster management.

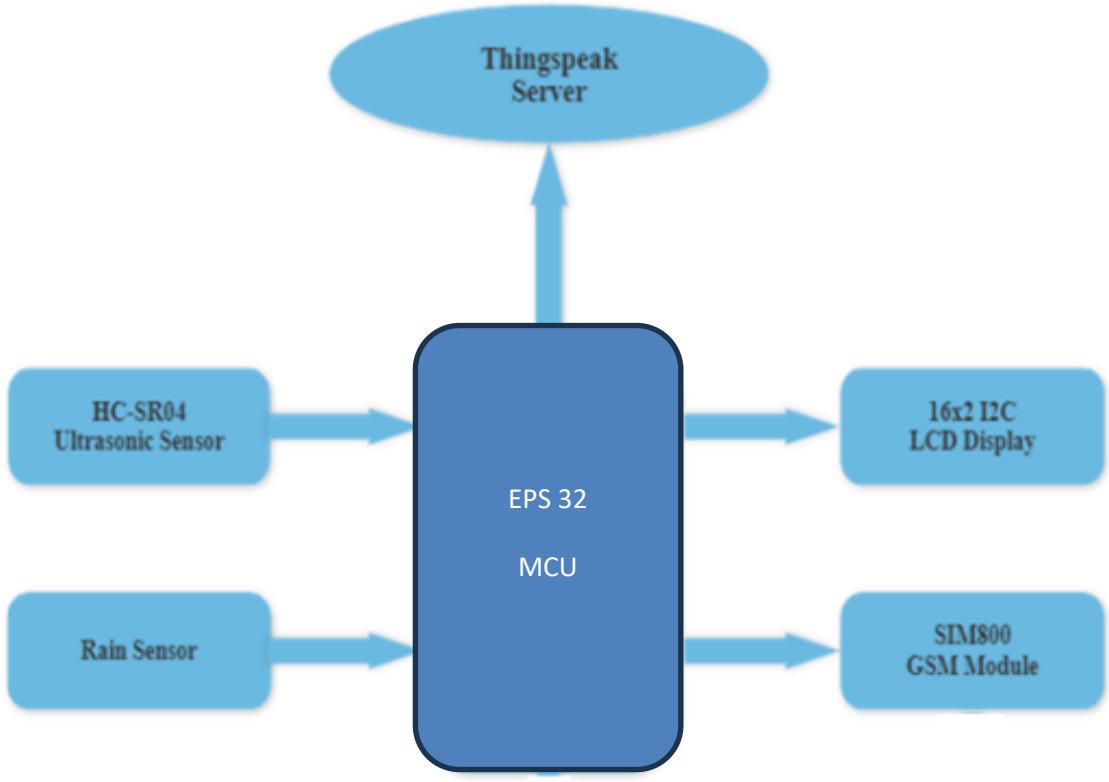

Fig-1: Flood Monitoring and Alerting System

Figure 3.1: Block diagram of attendance module

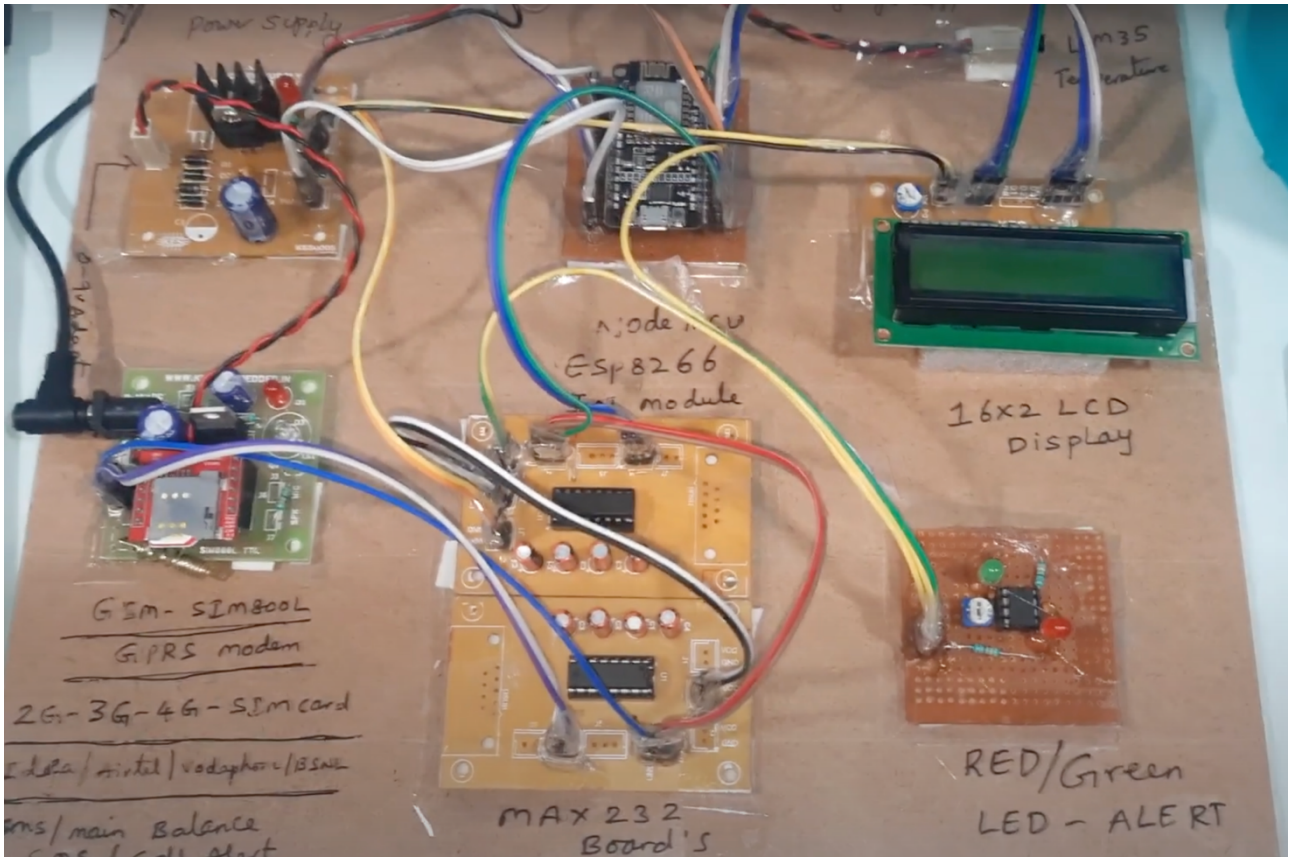

Figure3.2:Attendance module setup

3.1.2 ESP 32 Microcontroller

The **ESP32** is a powerful microcontroller from **Espressif Systems** designed for **IoT applications**. It features a **dual-core processor**, supports **Wi-Fi** and **Bluetooth** (Classic and BLE), and has multiple **GPIO pins** for various input/output functions. With **low power consumption**, **integrated peripherals** like ADC, DAC, PWM, and various communication protocols (SPI, I2C, UART), it is ideal for connected devices. It also offers built-in **security features** and supports **FreeRTOS** for real-time applications. The ESP32 is cost-effective, compact, and widely used in projects such as **smart home automation**, **environmental monitoring**, and **wireless communication**

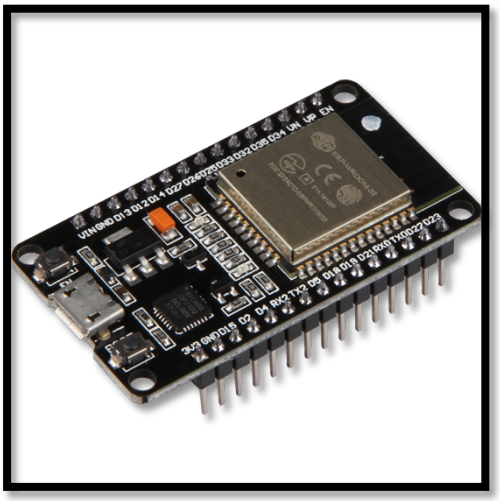

Figure 3.1.2:ESP 32 MCU

3.1.3 Ultra Sonic Sensor

An **ultrasonic sensor** is an electronic device that measures distance by emitting ultrasonic sound waves and detecting the time it takes for the waves to reflect back from an object. It consists of a **transmitter**, which sends out high-frequency sound waves, and a **receiver**, which detects the reflected waves. The distance to the object is calculated based on the time delay between transmission and reception of the waves. Ultrasonic sensors are widely used in applications such as **robotics**, **distance measurement**, **object detection**, and **level sensing** due to their **non-contact measurement** capability, accuracy, low cost, and simplicity. They are particularly useful in **flood monitoring systems** to measure water levels in rivers or dams. However, their performance can be affected by environmental factors such as temperature and humidity, and they typically have a range of up to a few meters

Figure 3.1.3:Ulra Sonic sensor

3.1.4 Working of GMS Module

A **GSM module** is a device that allows communication over a cellular network using a SIM card. It enables devices to send and receive **SMS messages**, make **voice calls**, and access **data** through mobile networks. The module operates by first registering itself with the nearest cell tower after being powered on. Once connected, it can send SMS messages using AT commands, receive messages, and even make or receive voice calls. Additionally, some GSM modules support **GPRS** for data communication, allowing devices to send and receive data over the internet. The GSM module is controlled through a series of **AT commands** that allow the microcontroller to interact with it for various tasks, such as sending alerts, remote control via SMS, and connecting IoT devices to cellular networks. It is widely used in applications like **remote monitoring**, **SMS-based control systems**, and **vehicle tracking**. The GSM module is cost-effective and provides reliable communication over cellular networks, though its data speed and performance are dependent on the network coverage and the type of network available

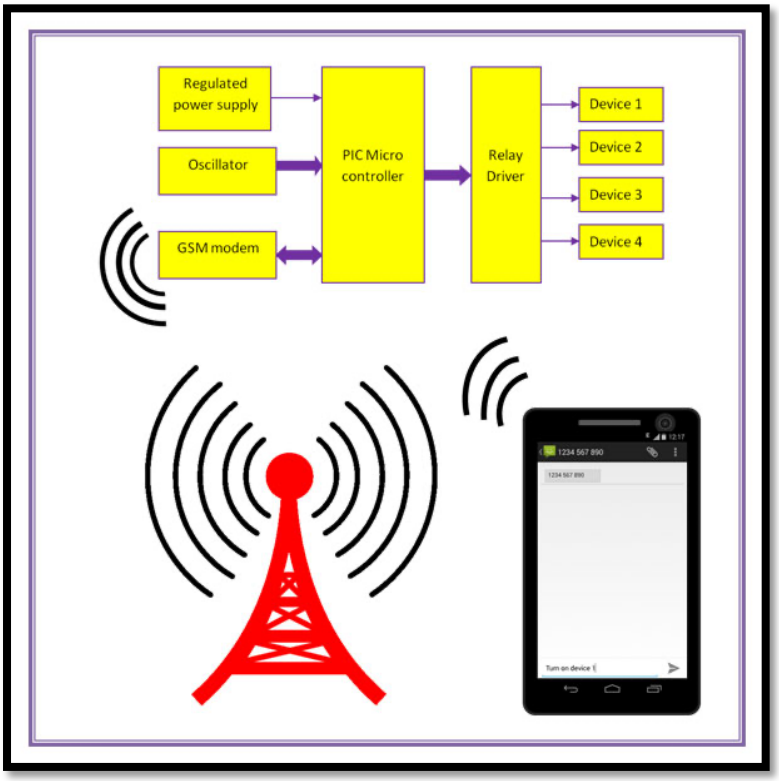

Figure 3.1.4: GSM Module system

3.2 Two-Way Communication

3.2.1 Introduction

In an **IoT-based flood monitoring and alerting system**, **two-way communication** refers to the ability to send data from sensors to a central server or monitoring station, as well as allowing commands or alerts to be sent back to the system or users. This ensures continuous monitoring, real-time updates, and interactive control over flood monitoring processes

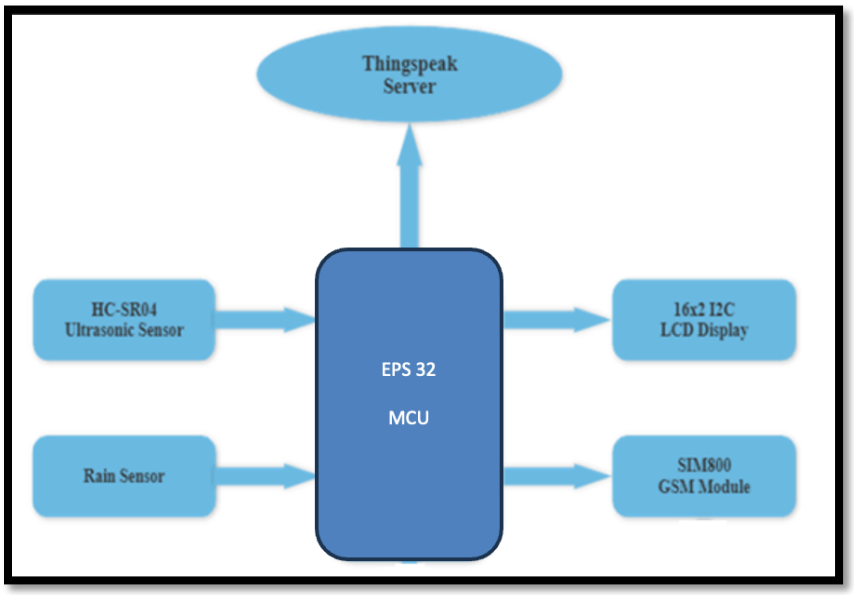

Figure 3.2: Block diagram of two-way communication.

In sensors such as water level and rainfall sensors continuously collect environmental data. This data is processed by a microcontroller (like **ESP32**), which communicates with a **GSM module** to send SMS alerts when flood risks are detected. The system stores data in a **cloud server**, which can also generate alerts based on specific conditions. These alerts are sent to users via a **mobile app**, where they can receive flood warnings and interact with the system. The **two-way communication** feature allows users to respond to alerts, request further data, or send commands to adjust the system settings, such as triggering flood control mechanisms or changing thresholds. This interactive setup enhances the system's effectiveness in managing flood risks by providing real-time updates and remote control capabilities.

3.2.2 LCD I2C

An **LCD I2C** is a type of display that uses the **I2C communication protocol** to interface with microcontrollers, enabling the display of alphanumeric characters such as sensor readings or status messages. The I2C protocol allows communication using just two data lines, **SDA (Serial Data Line)** and **SCL (Serial Clock Line)**, making it more efficient than standard parallel LCD displays, which require more pins. The LCD typically comes in sizes like 16x2 or 20x4, providing multiple lines of text. It also often includes a backlight for visibility in low-light conditions. This type of display is commonly used in **embedded systems** and **IoT projects**, where limited I/O pins are available, such as in microcontrollers like **Arduino** or **ESP32**. The **LCD I2C** is compact, easy to wire, and supported by libraries that simplify integration, making it ideal for applications like **sensor monitoring** and **real-time data display**. Its use of I2C communication reduces wiring complexity and pin usage, offering a space-efficient and cost-effective solution for various projects

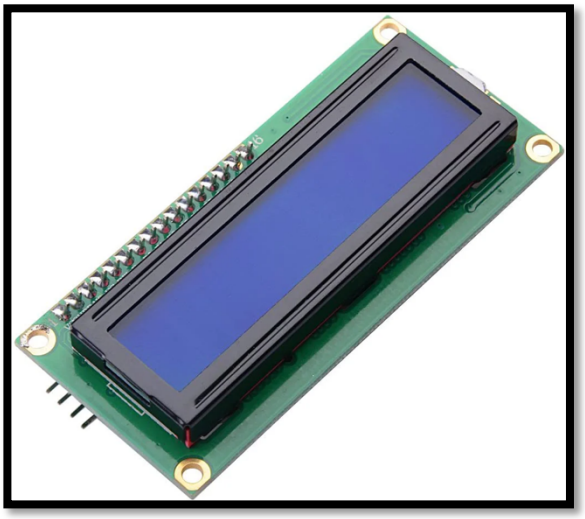

Figure 3.3: LCD I2C

3.3 Additional Features

Additional features of an IoT-based flood monitoring and alerting system :

1. **Real-Time Data Visualization**: Display data through web dashboards or mobile apps for easy monitoring.
2. **GPS Integration**: Provides location-based monitoring and flood mapping.
3. **Automated Flood Control**: Activates flood control measures (e.g., pumps or barriers) based on thresholds.
4. **Weather Forecast Integration**: Uses weather data for early flood warnings and proactive actions.
5. **Energy Efficiency**: Utilizes low-power sensors or solar panels for extended operation.
6. **Multiple Alert Methods**: Sends alerts via SMS, email, push notifications, and phone calls.
7. **Data Logging & Reporting**: Logs sensor data and generates reports for analysis.
8. **SMS-Based Remote Control**: Allows users to send commands to control or adjust the system.
9. **Integration with Social Media/Authorities**: Shares alerts and updates with local authorities or on social media.
10. **Fault Detection & Self-Diagnosis**: Detects faulty components and alerts users for maintenance.

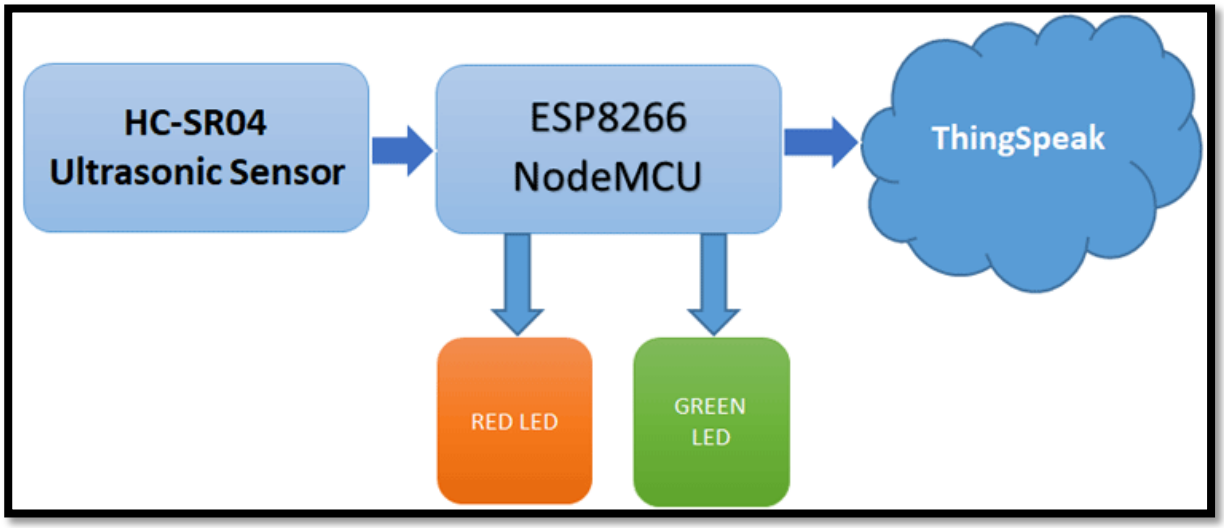

Figure 3.4 : Block diagram of IOT Based Flood Monitoring and Alerting System.

3.4 Final Project Outlook

Final Project Outlook: IoT-Based Flood Monitoring and Alerting System

The **IoT-Based Flood Monitoring and Alerting System** aims to provide real-time monitoring, alerts, and control in flood-prone areas. The system integrates various sensors like water level sensors, rain gauges, and moisture detectors to continuously collect environmental data. This data is processed by a **microcontroller** (ESP32/Arduino) and communicated through a **GSM module** for SMS-based alerts. Additionally, the system includes a **cloud server** for storing and processing data, enabling access to alerts via mobile apps or web interfaces.

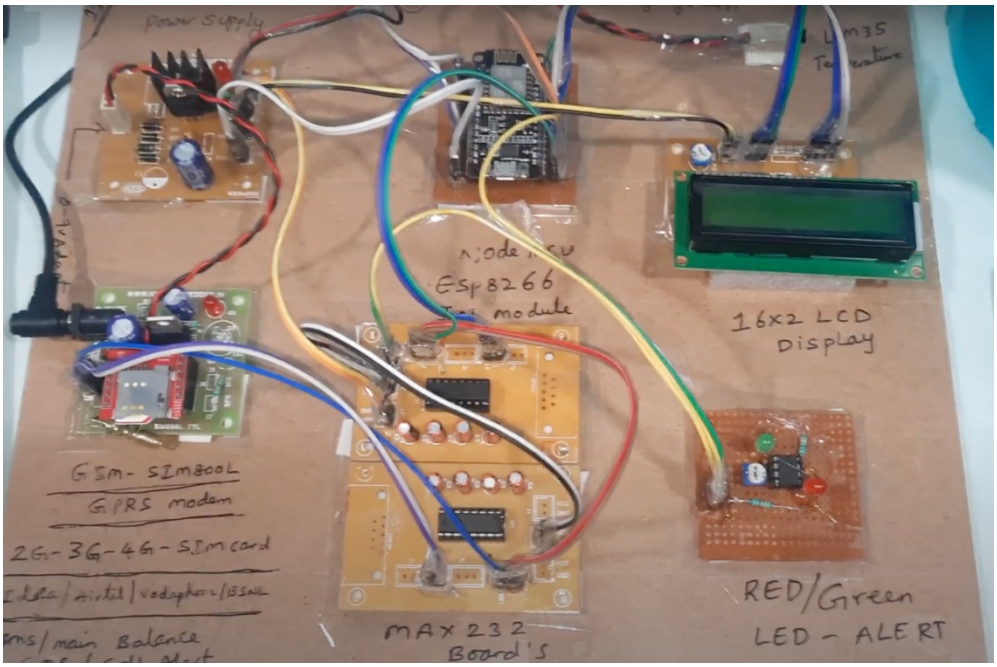

Figure 3.4:Final project outlook

The system supports **two-way communication**, allowing users to receive alerts and send commands for adjustments or further information. It also integrates additional features such as **real-time data visualization**, **weather forecast integration**, **automated flood control**, and **remote control via SMS**. These features ensure the system can provide timely warnings, take automated actions, and allow for remote management.

The project is designed to be **energy-efficient**, with **low-power sensors** and options for **solar power** in remote locations. **Data logging** and **report generation** capabilities make it easy to analyze flood trends and system performance over time.

Ultimately, this system aims to **improve flood preparedness and response** by providing early warnings, real-time monitoring, and seamless communication between users, authorities, and the system.

CHAPTER 4
RESULT ANALYSIS

The **result analysis** of the IoT-based flood monitoring system focuses on key areas: ensuring **sensor accuracy** for reliable data, evaluating the **timeliness of alerts** through SMS, and testing **two-way communication** for user interaction. It also examines the efficiency of **data storage** and access via cloud, the effectiveness of **automated flood control** actions, and **power efficiency** for remote areas. The **user experience** is assessed for ease of use, while the system's ability to **integrate with authorities** and provide **reliable alerts** is also tested. Overall, the system's performance is optimized for accurate, responsive flood monitoring and emergency management.

4.1 IOT based flood monitoring and Alerting system

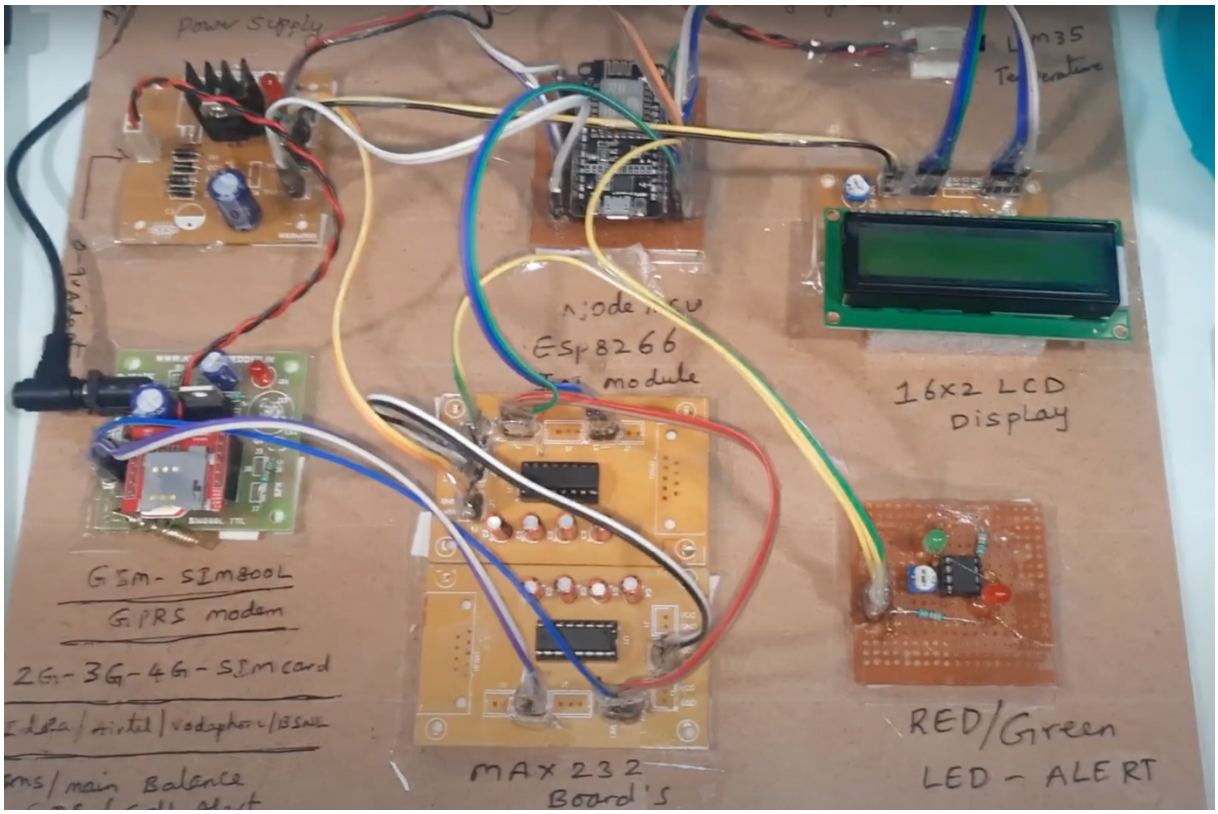

Figure 4.1: IOT based flood monitoring and alerting system.

4.2 Video conference.

Two-way communication is a form of transmission in which both parties involved transmit information. Two-way communication has also been referred to as interpersonal communication. One-way communication is when a message flows from sender to receiver only, thus providing no feedback. Some examples of one-way communication include: radio or television programs or even listening to policy statements from top executives. Two-way communication is especially significant in that it enables feedback to improve a situation.

4.3 Digital library & E-notes.

The **IoT-Based Flood Monitoring and Alerting System** can integrate with **Digital Libraries** and **E-notes** to enhance flood management and education. A **Digital Library** would store resources like research papers, manuals, and guidelines on flood monitoring and mitigation, providing easy access to vital information. **E-notes** can document real-time flood data, alerts, and safety measures, which can be shared with local authorities and communities. This integration improves **knowledge sharing**, **communication**, and **disaster preparedness**, offering a more efficient and collaborative approach to flood management and response.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

The **IoT-based Flood Monitoring and Alerting System** is a highly effective tool for flood prevention and management. By utilizing various sensors (such as water level and rainfall sensors), the system can continuously monitor environmental conditions in real time. It processes this data through a microcontroller and triggers timely alerts via SMS, mobile apps, or cloud platforms, allowing local authorities and citizens to take preventive actions. The system's **two-way communication** allows users to interact with the system, making it more dynamic and responsive. The integration of **automated flood control measures**, **weather forecasting**, and **real-time data visualization** further enhances its effectiveness, ensuring better preparedness and faster responses during flood events. This system not only reduces flood-related risks but also empowers communities to take proactive measures, thus saving lives and property.

5.2 Future Scope

\The **future scope** of the IoT-based flood monitoring and alerting system is vast and can include:

1. **Advanced Analytics and AI Integration**: Incorporating **machine learning** and **artificial intelligence** to predict floods more accurately and generate automated decision-making based on historical data and weather patterns.
2. **Wider Integration with Smart Cities**: Expanding the system to be part of the **smart city infrastructure**, allowing seamless data sharing with other systems such as traffic, water supply, and emergency services for a more coordinated response.
3. **Expansion of Sensor Types**: Integrating more types of sensors such as **air quality monitors**, **soil moisture sensors**, and **river-flow sensors** to provide a more comprehensive flood prediction system.
4. **Improved User Interface**: Enhancing the **mobile app** or **web interface** for better user experience, adding features like **voice commands**, **interactive maps**, and **real-time video feeds** from flood-prone areas.
5. **Global Application and Remote Areas**: Expanding the system for use in global flood-prone regions, especially in remote and underdeveloped areas, using **solar-powered devices** for sustainable operation.
6. **Blockchain for Data Security**: Implementing **blockchain technology** to ensure secure and transparent data sharing, especially for local authorities and agencies involved in flood management.

REFERENCES

1. **Vasudevan, S., & Vishnu, P. R. (2019)** - Explores IoT applications for flood monitoring and control.
2. Wikipedia, "IOT based flood monitoring and alerting system ," [Online]. Available: https://en.wikipedia.org/wiki/Flood_warning
3. OpenAI, ChatGPT, ChatGPT Model for Natural Language Processing Assistance, 2025. [Online]. Available: https://openai.com/chatgpt
4. Google

APPENDIX

PROGRAM CODE

```
#include <WiFi.h>
#include <ThingSpeak.h>
#include <Wire.h>
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
SoftwareSerial SIM900(16,17);
String SMS;
// WiFi credentials
const char* ssid = "spi";
const char* password = "12345678";

// ThingSpeak settings
unsigned long myChannelNumber = 2722948;
const char* myWriteAPIKey = "CU0CW593FTDV31GR";

// GPIO Pin assignments
const int trigPin = 18;
const int echoPin = 19;
const int redLedPin = 2;
const int greenLedPin = 4;
#define rainPin 14                              // Pin for rain sensor digital output
#define DHTPIN 25          // Pin where the DHT11 data pin is connected
#define DHTTYPE DHT11       // DHT 11
DHT dht(DHTPIN, DHTTYPE);    // Initialize DHT sensor


// GSM Module configuration
//SoftwareSerial gsmSerial(16, 17);  // TX, RX

// LCD setup (I2C address 0x27)
LiquidCrystal_I2C lcd(0x27, 16, 2);

// ThingSpeak client
WiFiClient client;

void setup() {
  // Start Serial communication
  Serial.begin(9600);
   SIM900.begin(9600);
```

```
  // Initialize LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Starting...");

  // Setup pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(redLedPin, OUTPUT);
  pinMode(greenLedPin, OUTPUT);
  pinMode(rainPin, INPUT);
  dht.begin(); // Initialize DHT sensor

  // Connect to Wi-Fi
  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected to WiFi");

  // Connect to ThingSpeak
  ThingSpeak.begin(client);

  // Initialize GSM Module
  //gsmSerial.begin(9600);
  delay(1000);
}

void loop() {
  float distance = measureDistance();
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Display data on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Distance: ");
  lcd.print(distance);
  lcd.print(" cm");
  lcd.setCursor(0, 1);
  lcd.print("T:");
```

```
  lcd.print(temperature);
  lcd.print("C H:");
  lcd.print(humidity);
  lcd.print("%");

  // Check if DHT sensor reads failed
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
  }

  // Distance detection with LED and GSM alert
  if (distance < 20) {
    digitalWrite(redLedPin, HIGH);
    digitalWrite(greenLedPin, LOW);

    // Send alert via GSM if distance is less than threshold
     SMS = "Alert:Flood Detected ";//////////////////////////////SMS message
   get_msg(SMS);
  } else {
    digitalWrite(redLedPin, LOW);
    digitalWrite(greenLedPin, HIGH);
  }

  // Rain detection with GSM alert
  int rainStatus = digitalRead(rainPin);
  if (rainStatus == LOW) {  // Assuming LOW means rain detected
    Serial.println("Rain detected!");
    SMS = "Alert:Rain Detected ";//////////////////////////////SMS message
   get_msg(SMS);
    delay(60000);  // Delay to avoid repeated alerts
  }

  // Send data to ThingSpeak
  sendDataToThingSpeak(distance, humidity, temperature);

  // Wait 20 seconds before the next reading
  delay(20000);
}

float measureDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
```

```
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  float distance = (duration * 0.034) / 2;
  return distance;
}

void sendDataToThingSpeak(float distance, float humidity, float temperature) {
  ThingSpeak.setField(1, distance);
  ThingSpeak.setField(2, humidity);
  ThingSpeak.setField(3, temperature);
  int responseCode = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  if (responseCode == 200) {
    Serial.println("Data sent to ThingSpeak");
  } else {
    Serial.print("Failed to send data, code: ");
    Serial.println(responseCode);
  }
}
int get_msg(String message){
  SIM900.print("AT+CMGF=1\r");     // AT command to set SIM900 to SMS mode
  delay(100);
  SIM900.print("AT+CNMI=2,2,0,0,0\r");       // Set module to send SMS data to serial out
upon receipt
  delay(100);

  SIM900.println("AT+CMGF=1"); // Replace x with mobile number
  delay(1000);
  SIM900.println("AT+CMGS= \"+919380676549\"\r"); // Replace * with mobile number  sim
number - 8861273413
  delay(1000);
  SIM900.println(message);// The SMS text you want to send
  delay(100);
  SIM900.println((char)26);// ASCII code of CTRL+Z

}
```