PROJECT REPORT
ON
RTOS-Based Traffic Accident Detection System

Carried Out at

CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING
ELECTRONIC CITY, BANGALORE.

UNDER THE SUPERVISION OF

Mr. Pavan Jadhav

C-DAC Bangalore

Submitted By ,
Tejas Vilas Chaudhari (240850130029)
Anilesh Kumar Singh (240850130009)
Vaishnavi Sanjay Patil (240850130031)

PG DIPLOMA IN BIG DATA ANALYTICS
C-DAC, BANGALORE

Candidate's Declaration

We hereby certify that the work being presented in the report entitled " RTOS-Based Traffic Accident Detection System", in partial fulfillment of the requirements for the award of PG Diploma, and submitted to the C-DAC Bangalore, is an authentic record of our work carried out during the period [Project Duration] under the supervision of Mr. Pavan Jadhav, The matter presented in the report has not been submitted by us for the award of any degree of this or any other Institute/University.

Tejas Vilas Chaudhari (240850130029)

Anilesh Kumar Singh (240850130009)

Vaishnavi Sanjay Patil (240850130031)

Counter Signed by

Name(s) of Supervisor(s)

.....

.....

ACKNOWLEDGMENT

We take this opportunity to express our sincere gratitude to **Mr. Pavan Jadhav**, whose invaluable guidance and support helped us successfully complete this project. We are also thankful to the faculty and staff of C-DAC Bangalore for providing resources and encouragement. Finally, we extend our gratitude to our peers and for their continuous motivation and support.

Tejas Vilas Chaudhari (240850130029)
Anilesh Kumar Singh (240850130009)
Vaishnavi Sanjay Patil (240850130031)

ABSTRACT

The "RTOS-Based Traffic Accident Detection System Using ESP32 WROOM Microcontroller and Vibration Sensor" project aims to develop a robust real-time solution for detecting traffic accidents and promptly alerting emergency services. The system employs an RTOS environment for efficient multitasking and responsive operations. Core components of the system include the ESP32 WROOM microcontroller for handling sensor data, a vibration sensor to detect collision impact, and GPS integration for precise location tracking. Upon detecting an accident, the system evaluates its severity and sends alerts containing GPS coordinates and other critical data to emergency services via the ThingSpeak IoT platform. The use of ThingSpeak allows real-time data visualization and cloud-based management, facilitating seamless communication between the vehicle system and emergency responders. This project addresses the need for enhanced road safety by automating the accident detection and alert process, thereby reducing emergency response time. By utilizing RTOS for task scheduling and real-time decision-making, the system achieves efficient resource management and high performance. This scalable solution offers potential for integration into modern vehicle safety frameworks and smart transportation systems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS & ACRONYMS

- **RTOS:** Real-Time Operating System
- **GPS:** Global Positioning System
- **GSM:** Global System for Mobile Communications
- **MPU:** Microprocessor Unit
- **LCD:** Liquid Crystal Display
- **IDE:** Integrated Development Environment
- **CMOS:** Complementary Metal-Oxide-Semiconductor
- **AVR:** Automatic Voltage Regulator
- **RISC:** Reduced Instruction Set Computing
- **SRAM:** Static Random-Access Memory
- **UART:** Universal Asynchronous Receiver-Transmitter
- **ESP32:** Espressif Systems 32-bit Microcontroller
- **STM:** STMicroelectronics (semiconductor company that manufactures STM32 microcontrollers)
- **USB:** Universal Serial Bus
- **SMS:** Short Message Service
- **API:** Application Programming Interface
- **I2C:** Inter-Integrated Circuit
- **SPI:** Serial Peripheral Interface

Chapter 1

INTRODUCTION

1.1 Background

Road accidents rates are very high nowadays,Accident detection systems play a crucial role in improving road safety by providing timely alerts and information to emergency responders. Traditional systems may have limited real-time data processing capabilities, leading to delays in response time. By leveraging Real-Time Operating Systems (RTOS) and advanced sensors, this project ensures accurate accident detection and efficient data handling for faster emergency response. Thus the systems will make the decision and sends the information to the smartphone, connected to the accelerometer through gsm and gps modules . The Android application in the mobile phone will send text messages to the nearest medical center and friends. Application also shares the exact location of the accident and it can save time.

The proposed system will:

- Continuously monitor vehicle conditions for accident detection using vibration and motion sensors.
- Analyze accident severity using MPU6050 and vibration sensor data.
- Send GPS coordinates and alerts through a GSM module.
- Provide real-time information display on an I2C LCD.
- Enable effective task scheduling and data processing using RTOS.

1.2 Purpose

The purpose of this document is to define the functional, technical, and operational requirements for an RTOS-Based Traffic Accident Detection System. This project aims to develop a system that detects traffic accidents using various sensors and alerts emergency services with essential information, such as GPS location and accident severity.

Avoiding casualties caused by road accidents is the main goal of this documents, with the help of Accelerometer and GPS present in the mobile phones. Based on the data collected from these sensors, which are present in most mobile phones, the location of the accident is sent at

the same time of the accident to the friends and relatives which the user allowed and stored, and also to the rescue and emergency

1.3 Scope

This project aims to implement a real-time accident detection system that:

- Detects collisions and abnormal vehicle behavior using sensors.
- Sends alerts with accident details and GPS coordinates through GSM.
- Displays system status and information on an LCD.
- Utilizes RTOS for task scheduling and optimized system performance.
- Supports scalable integration for additional sensors or modules.
- This design is a system which can detect accidents in significantly less
- time and sends the basic information to the emergency contacts within a few seconds covering
- geographical coordinates. This alert message is sent to the emergency contacts in a short time,
- which will help in saving the valuable lives.

1.4 Objectives:

- User friendly,and real time mobility
- Two mobiles' numbers can be saved. So in case we couldn't reach out one the other will be useful.
- Response time is fast
- Nearby hospital location is sent so it is easier to navigate the patients as soon as possible.
- Text message is sent so internet is not needed.
- Certainly, if the accident happens due to other cases, the used electronic devices will be able to provide the spontaneous message and exact location to police and ambulance in order to recover victims.
- Avoiding casualties caused by road accidents is the main goal of this project, with the help of Accelerometer and GPS present in the mobile phones.

Chapter 2

LITERATURE SURVEY

A system is designed to create an alerts when an accident occurs using Arduino, sensors, GSM, and GPS technology.

The alerts is activated automatically, and the system sends information about the accident, including the vehicle number and other details, to the nearest police station and hospital along with a Google Maps link to the accident scene . The proposed system incorporates an accelerometer to provide information on the rapid acceleration of the vehicle. Once these readings reach threshold values, the Arduino will send a signal to the GPS module to fetch the current location. This information, along with the location of the accident and the speed of the car, will be sent to the appropriate authorities via the GSM module to ensure that help arrives on time . The system consists of a single-board embedded system that is equipped with Global System for Mobile Communication (GSM) and Global Positioning System (GPS), along with a microcontroller. By utilizing GSM and GPS technologies, the system is capable of tracking the vehicle and providing the most up-to-date information about ongoing trips . The system monitors information from the accelerometer and vibration sensor, which allows it to recognize when a severe accident has occurred. Once detected, the system sends an alert message informing about latitude and longitude data, sensed or measured by the GPS module, and sends the information to the police control room, any rescue team, or the car owner through the GSM module. This enables the police to immediately trace the location of the accident and take the necessary actions after receiving the emergency message . The system is designed to make decisions and send information to a smartphone that is connected to the accelerometer through GSM and GPS modules. This can save precious time and enable quick medical attention to be provided to those involved in the accident . The GPS receiver is responsible for receiving location information from satellites in the form of latitude and longitude. The microcontroller processes this information, and the processed information is then sent to the user/owner using a GSM module.. Automatic accident detection systems and alerts can reduce response time by immediately notifying emergency personnel and providing data for analysis . The model integrates an Arduino microcontroller, GPS receiver, and GSM module to detect vehicle accidents and send SMS notifications with location data. An ADXL335 sensor captures vehicle

coordinates and a 16x2 LCDs messages and accident location . IoT can enable automatic notification and response to vehicle accidents by sending accelerometer and GPS sensor signals to the cloud, which triggers an alert message to subscribed individuals. The message includes accident severity and GPS location data. The ambulance can use GPS coordinates for quick response to the scene .

2.1 Literature references

"To refer a research paper" -

[1] International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878 (Online), Volume-11 Issue-6, March 2023

[2] RTOS Based Advanced Vehicle Security System for Best Controlling

[3] Bhargava, D. U., Kumar, B., & Rao, A. (2022). AUTOMATIC VEHICLE ACCIDENT DETECTION AND RESCUE SYSTEM USING GSM AND GPS MODULE. JOURNAL OF CRITICAL REVIEWS, 163-168.

Chapter 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Aurdino IDE Requirements

Arduino is based on the ESP32 microcontroller. The Arduino UNO is a widely used microcontroller board that can detect accidents through a vibration sensor and alert emergency services or designated contacts via the GSM module. It is efficient, flexible, and easy to use, making it a popular choice for embedded applications. The Arduino IDE is an open-source development environment used for writing, compiling, and uploading code to Arduino boards. It supports C and C++ with built-in functions for easy hardware control.

Key Features of Arduino IDE

- **Code Editor:** Simple editor with syntax highlighting and auto-formatting.
- **Compiler:** Converts code into machine language for microcontrollers.
- **Libraries:** Pre-written code for sensors, displays, motors, etc.
- **Serial Monitor:** Debugging tool to communicate with the board via UART.
- **Board Manager & Port Selection:** Supports multiple microcontrollers (e.g., AVR, ESP32, STM32).

Installing Arduino IDE

1. **Download:** Arduino Official Website
2. **Install:** Follow the setup instructions for Windows, macOS, or Linux.
3. **Connect Board:** Plug in your Arduino Uno/Nano/Mega/ESP32 via USB.
4. **Select Board & Port:**
    - Go to **Tools > Board** and choose your board.
    - Go to **Tools > Port** and select the correct COM port.
5. **Write & Upload Code:** Open an example sketch (e.g., Blink, thinkspeak) and click Upload**.**

- **Simulink Model (for Simulation and Code Generation):**

    - **Sensor Interface:** Blocks representing the accelerometer, gyroscope, and GPS module. These blocks would simulate sensor readings for testing.

- **Signal Processing:** Blocks to filter and process sensor data. For example, a moving average filter for the accelerometer, and logic to detect impact or rollover events.
- **Accident Detection Logic:** A subsystem that combines the processed sensor data to determine if an accident has occurred. This might involve thresholds for acceleration, angular velocity, or a combination of both.
- **GPS Processing:** Extract location data from the simulated GPS signal.
- **Severity Estimation:** Based on the sensor readings (e.g., magnitude of acceleration change), estimate the severity of the accident (e.g., low, medium, high).
- **ThingSpeak Interface:** Blocks to format data and send it to ThingSpeak using the ThingSpeak API. This would likely involve HTTP requests.
- **RTOS Scheduler:** Configure the Simulink model to generate code compatible with an RTOS. Tasks would be created for sensor reading, data processing, accident detection, and communication.

- **Embedded C Code (Generated from Simulink):**
  - The Simulink model would be used to generate C code that runs on the MCU.
  - This code would implement the RTOS tasks, sensor drivers, signal processing algorithms, accident detection logic, and ThingSpeak communication.
- **ThingSpeak Setup:**
  - Create a ThingSpeak channel to store the data (fields for accelerometer x, y, z; gyroscope x, y, z; latitude, longitude; severity).
  - Configure ThingSpeak to trigger alerts (using MATLAB Analysis or other features) when an accident is detected. Alerts could be sent via SMS, email, or a custom webhook.

RTOS Tasks:

- **Sensor Reading Task:** Reads data from the sensors at a regular interval.
- **Data Processing Task:** Filters and processes the sensor data.
- **Accident Detection Task:** Runs the accident detection logic.
- **Communication Task:** Sends data to ThingSpeak when an accident is detected.

Accident Detection Algorithm (Example):

1. **Impact Detection:** Check if the magnitude of acceleration exceeds a threshold.
2. **Rollover Detection:** Check if the angular velocity exceeds a threshold for a sustained period.
3. **Combined Logic:** Combine impact and rollover detection. For example, an accident might be detected if either impact or rollover is detected, or if both are detected within a short time window.

3.2 ThingSpeak Integration:

- Use the ThingSpeak API to send data from the MCU to the ThingSpeak channel.
- Configure ThingSpeak alerts to notify emergency services.

Key Considerations:

- **Real-time performance:** The system must be able to detect accidents and send alerts quickly.
- **Reliability:** The system must be reliable and robust.
- **Power consumption:** Important for battery-powered devices.
- **Security:** Secure communication between the device and ThingSpeak.
- **Testing:** Thoroughly test the system under various conditions

3.3 Hardware and Software Requirements

➢ **Hardware Requirements**

The Proposed system configuration is as follows:

3.3.1 ESP 32 WROOM/DEVKITC:

ESP32 Hardware Considerations:

- **Power Supply:** The ESP32 can be powered via USB or an external power supply. Consider the power requirements of your sensors as well.
- **Pinouts:** Carefully review the ESP32 pinout diagram to connect your sensors correctly. I2C, SPI, and UART peripherals share pins, so you may need to choose your sensors and connections carefully.
- **Antenna:** The ESP32 has an onboard antenna for Wi-Fi. Ensure that it's not obstructed.

- **External Components:** You'll need to connect the accelerometer/gyroscope, GPS module, and potentially a cellular module to the ESP32. Use appropriate wiring and consider any level shifting that might be required.

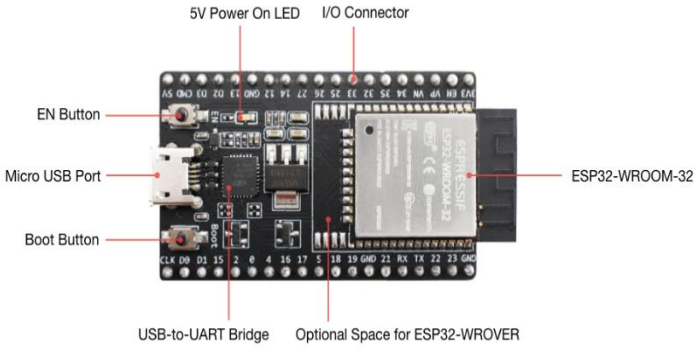

Fig 3.3.1 ESP 32 WROOM/DEVKITC

ESP32 Software Considerations:

- **Development Environment:** You can use the Arduino IDE with the ESP32 core installed, or PlatformIO, which is often preferred for more complex projects. The ESP-IDF (Espressif IoT Development Framework) provides a lower-level, more powerful development environment but is more complex.
- **FreeRTOS Configuration:** Configure FreeRTOS to create the necessary tasks (sensor reading, data processing, accident detection, communication). Pay attention to task priorities and scheduling.
- **Sensor Libraries:** Use appropriate Arduino libraries for your accelerometer/gyroscope (e.g., Adafruit_MPU6050, SparkFunLSM6DS3) and GPS module.
- **ThingSpeak Library:** Use the ThingSpeak library for ESP32 to send data to ThingSpeak over Wi-Fi.
- **Communication:** The ESP32 will communicate with ThingSpeak over Wi-Fi. Ensure that you have a stable Wi-Fi connection. If using a cellular module, you'll need to use a library for that (e.g., SIM900a, SIM800).
- **Memory Management:** The ESP32 has limited memory. Be mindful of memory usage and avoid memory leaks.
- **Debugging:** Use the serial monitor for debugging. You can print sensor readings, status messages, and error codes.

Key ESP32 Considerations for the Project:

- **Real-time Requirements:** The ESP32's dual-core processor and FreeRTOS allow for real-time processing of sensor data. However, you need to carefully design your tasks and scheduling to ensure that you meet the real-time requirements of accident detection.
- **Power Consumption:** If your system is battery-powered, you'll need to consider power consumption. The ESP32 has power-saving modes that you can use to reduce power consumption.
- **Wi-Fi Connectivity:** A reliable Wi-Fi connection is essential for communication with ThingSpeak. Consider the range and stability of your Wi-Fi network.

3.3.2 MPU-6050 (Accelerometer/Gyroscope):

- **Accelerometer:** Measures linear acceleration along three axes (x, y, z). Used for impact detection and, less directly, for contributing to rollover detection. It detects sudden changes in acceleration, which are characteristic of collisions.
- **Gyroscope:** Measures angular velocity (rotation rate) around three axes. Primarily used for rollover detection, as it directly senses rotational motion.

 GPS Module: Provides geographic location (latitude and longitude). Essential for pinpointing the accident location.

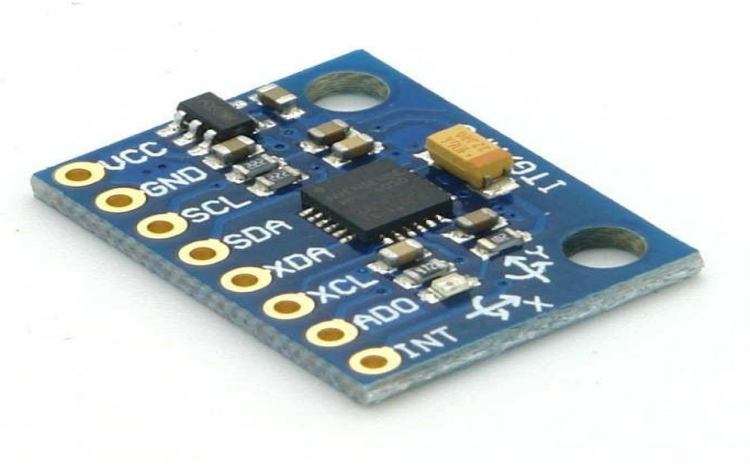

Fig 3.3.2 MPU-6050 (Accelerometer/Gyroscope)

Component Workflow and Function:

➢ **Sensor Data Acquisition:**

- The sensor_reading_task in the RTOS periodically reads data from the MPU-6050 and GPS module.
- **MPU-6050 (Accelerometer):** Measures acceleration due to gravity, vehicle motion, and impacts. The raw acceleration data is noisy.

- **MPU-6050 (Gyroscope):** Measures the rate of rotation of the vehicle. Crucial for detecting rollovers.
- **GPS Module:** Receives signals from GPS satellites and calculates the device's latitude and longitude.

➢ **Data Processing:**

- The data_processing_task receives the raw sensor data.
- **MPU-6050 Data Filtering:** This is *essential*. The accelerometer data is filtered (e.g., using a moving average or Kalman filter) to remove noise and spurious readings. This makes the impact detection more reliable.

3.3.3 Neo 6M (Antenna):

- **Function:** The Neo-6M receives signals from multiple GPS satellites. These signals contain information about the satellite's position and the time the signal was transmitted. The module uses this information to calculate its own position (latitude, longitude, and altitude).
- **Antenna (Crucial):** The antenna is absolutely essential for the Neo-6M to work. It's the component that receives the radio signals from the GPS satellites. Without a properly connected antenna, the Neo-6M will not be able to acquire a GPS fix (i.e., determine its location).
- **Antenna Types:** Neo-6M modules typically come with a ceramic antenna or a connector for an external antenna. External antennas can provide better reception in some situations.
- **Placement:** The antenna should have a clear view of the sky. Obstructions like metal or buildings can significantly weaken the GPS signal. In a vehicle, placement on the dashboard or near a window is often best. Avoid placing the antenna where it might be shielded by metal parts of the car.
- **Connecting the Antenna:** Ensure the antenna is correctly connected to the Neo-6M module.
- **Powering the Antenna:** Some Neo-6M modules require separate power for the antenna. Check the module's datasheet.

  Buck LM2596:

The LM2596 is a DC-DC buck converter that steps down voltage from a higher source (e.g., 12V from a vehicle battery) to a stable 5V or 3.3V to power components like the ESP32, MPU6050, GPS, and GSM modules.

Fig 3.3.3 Neo 6M (Antenna)

3.3.4 Buck LM2596:

DC-DC Buck Converter Step Down Module LM2596 Power Supply is a step-down(buck) switching regulator, capable of driving a 3-A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3 V, 5 V, 12 V, and an adjustable output version.

The LM2596 series operates at a switching frequency of 150kHz, thus allowing smaller sized filter components than what would be required with lower frequency switching regulators.

Fig 3.3.4 Buck LM2596

Hardware Connections:

- **LM2596:**

- **Input:** Connect to the car battery's positive terminal *through a fuse*. Connect the LM2596's ground input to the car battery's negative terminal (ground).
- **Output:** Adjust the output voltage of the LM2596 to 3.3V (or the voltage required by your components). Use a multimeter to verify. Connect the LM2596's positive output to the 3.3V pin of the ESP32, the power pin of the MPU-6050, and the power pin of the GPS module. Connect all grounds together.

3.3.5 Vibration Sensor:

A vibration sensor-based RTOS system for traffic accident detection would utilize a vibration sensor to monitor vehicle movements, triggering an alert to emergency services when a significant impact (accident) is detected, alongside GPS data to pinpoint the location and potentially additional sensors to assess accident severity, all managed by a real-time operating system (RTOS) for timely response.

A vibration sensor can provide additional information about an impact, potentially improving the accuracy and reliability of your accident detection system. It can be especially useful for detecting lower-impact collisions that might not produce a large enough acceleration change to trigger the accelerometer alone.

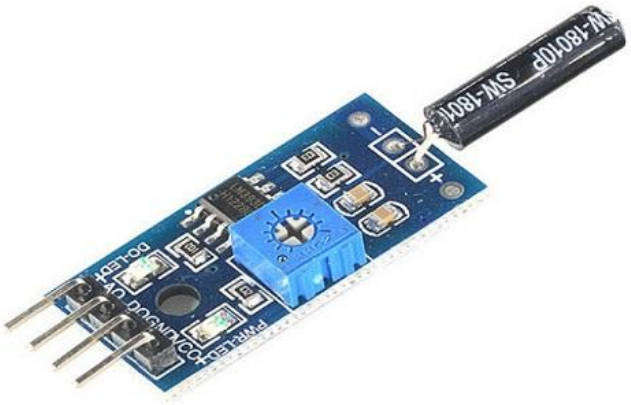

Fig 3.3.5 Vibration Sensor:

3.3.6 GSM 900A:

GSM stands for Global System for Mobile Communication.

A GSM works in 12v input supply. In our project we give 12v from battery directly. There is Vcc, Tx, Rx and Gnd pins are there.There are three led one is power, second one is Network and thirdone is working. It is a 2g unit.A GSM modem or GSM module is a device that uses GSM mobiletelephone technology to provide a wireless data link to a network. GSM modems are used in mobile telephones and other equipment that

communicates with mobile networks. They use SIMs to identify their device to the network.

Hardware Connections:

- **GSM 900A Module:**
  - **Power:** Connect the GSM module's power pin (typically 3.3V or 5V - check your module's specifications) to the regulated output of your LM2596 buck converter. Connect the GSM module's ground to the system's ground.
  - **Antenna:** Connect the GSM antenna to the GSM module. *This is essential for the module to function.*
  - **SIM Card:** Insert a SIM card into the GSM module's SIM card slot. Make sure the SIM card has an active data plan or SMS capability.

- **Key Considerations for GSM Usage:**

2. **SIM Card:** Ensure the SIM card is active and has SMS capability.
3. **Antenna:** The antenna is *crucial*. Without it, the GSM module won't work.
4. **Power:** The GSM module can draw significant current, especially when transmitting. Make sure your power supply can handle it.
5. **GSM Library:** Choose a reliable GSM library for your ESP32.

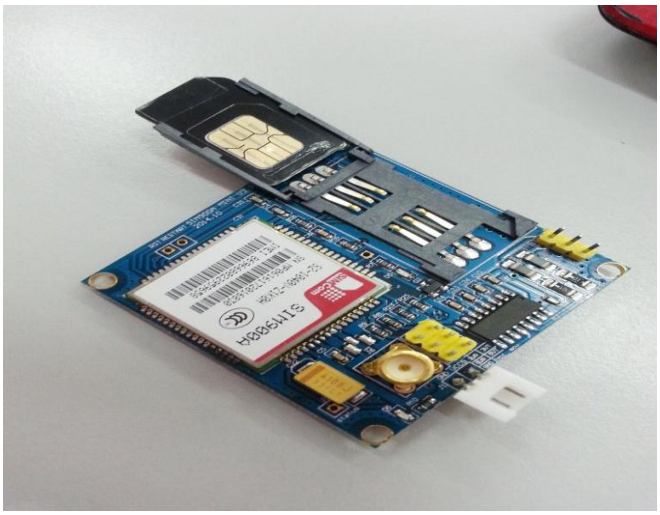

Fig 3.3.6 GSM 900A

3.3.7 Buzzer module:

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.A buzzer provides immediate feedback to the vehicle occupants that an accident has been detected. This can be useful for:

- **Confirmation:** It confirms that the system has detected the accident and is sending an alert.
- **Local Alert:** It can alert other drivers or passersby that an accident has occurred, even if the remote alert fails for some reason.

Fig 3.3.7 Buzzer module

Hardware Connections:

- **Buzzer Module:**
  - **Power:** Connect the buzzer module's power pin (typically 3.3V or 5V) to the regulated output of your LM2596 buck converter. Connect the buzzer's ground pin to the system's ground.
  - **Signal:** Connect the buzzer's signal pin to a digital output pin on the ESP32.

3.4 Software :

The development of the RTOS-Based Traffic Accident Detection System utilized various software tools and technologies. The Arduino IDE was chosen for programming the ESP32 WROOM microcontroller due to its simplicity and extensive library support. The system was programmed primarily in C/C++ to ensure efficient hardware interaction and compatibility with real-time tasks. FreeRTOS was integrated to manage task scheduling and inter-task communication, enabling the execution of concurrent processes such as vibration sensor data acquisition, GPS location retrieval, and emergency notifications. The MQTT protocol facilitated lightweight and efficient communication between the microcontroller and cloud-

based IoT platforms like Blynk or Thingspeak, which provided real-time visualization and monitoring of accident data. This software framework ensured a scalable, real-time system capable of detecting traffic accidents and transmitting crucial information for prompt emergency responses.

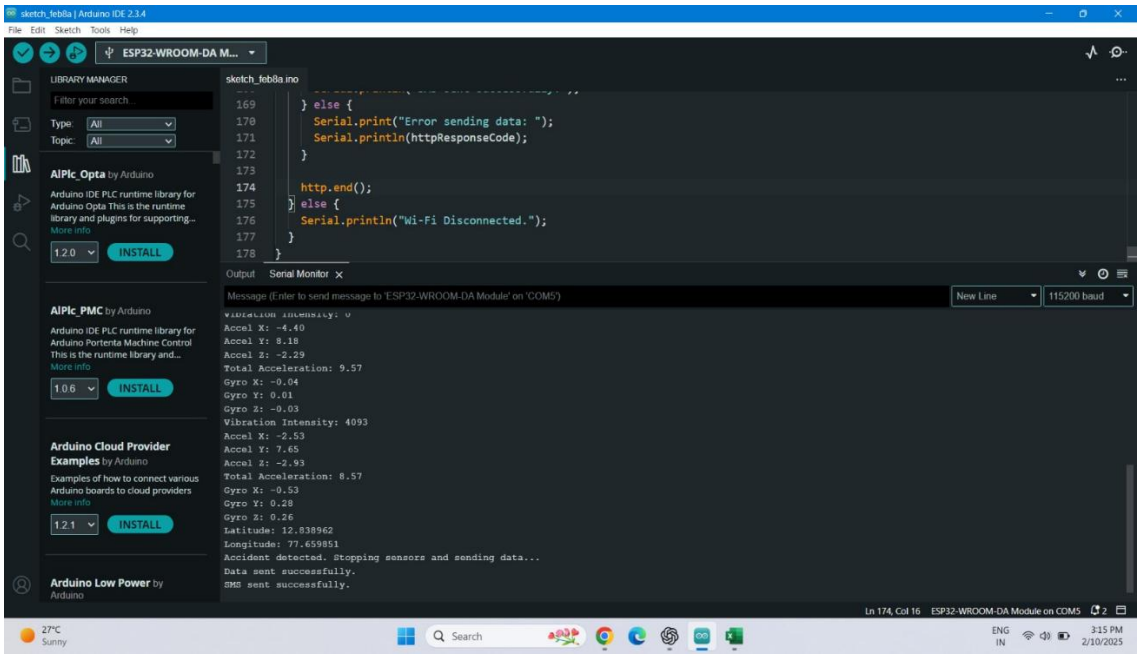

Fig 3.4 Software ide

Chapter 4

SYSTEM ARCHITECTURE

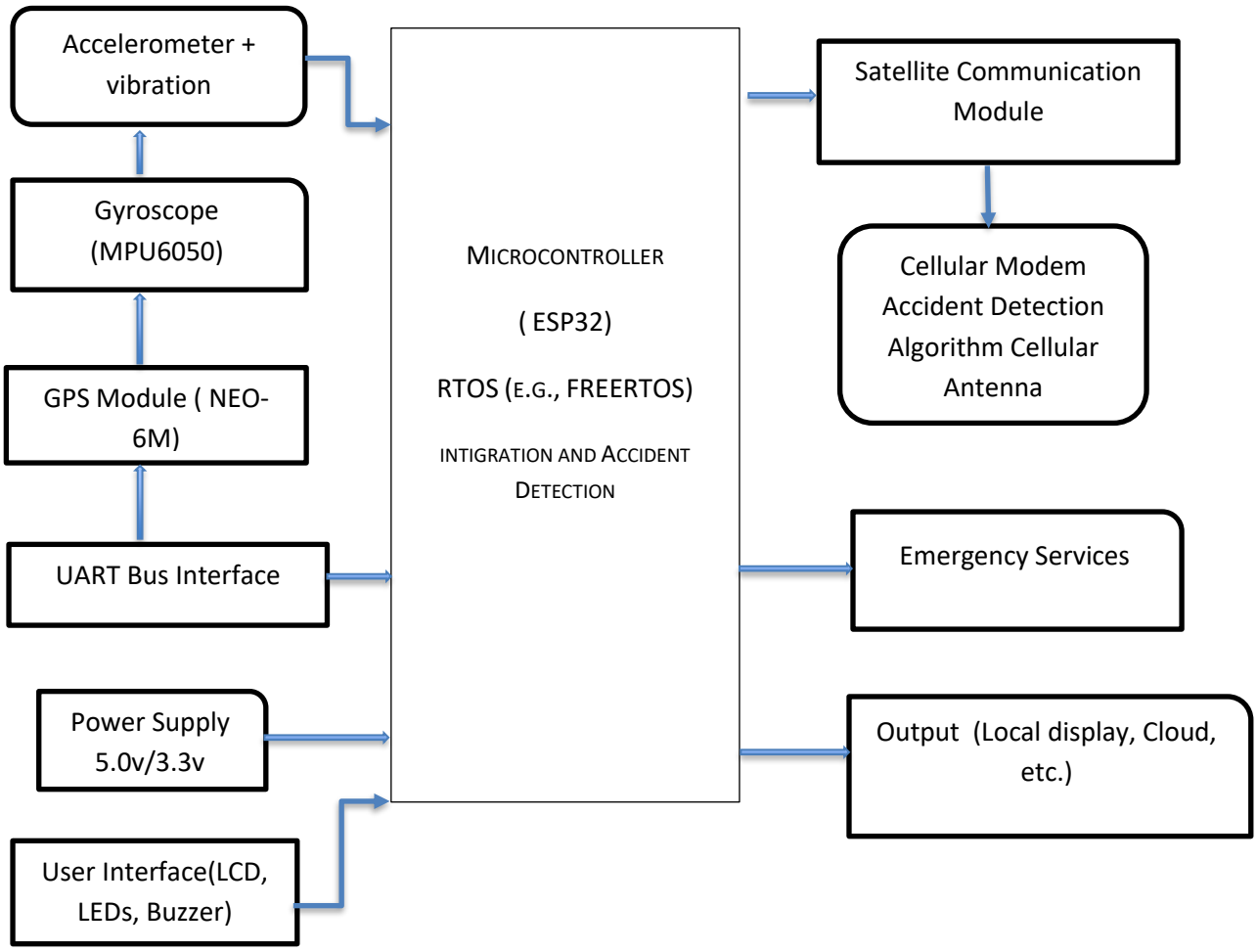

Fig 4.1 System Architecture

4.1 Overview

This initiative aims to enhance road safety and emergency response by promptly relaying the location of a vehicular accident to the nearest police station, hospital, and ambulance, thereby potentially saving lives. The project aims to address the issue of delayed medical attention for accident victims. It utilizes an accelerometer to detect car crashes and GPS and GSM modules to promptly report the incident, potentially expediting the hospitalization of patients and improving their chances of recovery. In the event of a vehicular accident, the accelerometer is designed to detect the impact and send a signal to the processor for subsequent processing. Fig 3 shows the algorithm of how the processor process the data of the accelerometer. To ensure accuracy and reliability, a predetermined threshold value is established within the processor to

continually monitor and compare the sensor or acceleration values. Once the acceleration value surpasses the established threshold, the system will recognize that an accident has occurred. Using GPS technology, the system provides real-time information regarding the accident's location, time, and date, which is then transmitted to ambulance services through GSM communication. The emergency response units receive the accident location in the form of a Google Maps link, enabling them to locate the victim swiftly and potentially save their life. Fig4 showcases a physical model of the accident detection system.

4.2 System Architecture Design

- **Sensor Module:**
    - **GPS Module:** Provides location coordinates.
    - **CAN Bus Interface (Optional but recommended):** For communication with other vehicle systems (speed, brake status, etc.).
- **Microcontroller (MCU) Module:**
    - Powerful MCU (e.g., ARM Cortex-M4/M7) running an RTOS (FreeRTOS, Zephyr).
    - Handles sensor data acquisition, accident detection, communication, and logging.
- **Communication Module:**
    - Cellular Modem (4G/LTE preferred): For sending alerts via SMS or data channels.
    - GPS Antenna.

2. Software Design (RTOS-based):

- **RTOS Kernel:** Manages tasks, scheduling, and inter-task communication.
- **Tasks:**
    - **Sensor Data Acquisition Task (High Priority):** Reads sensor data periodically, performs basic filtering/calibration.

- **Accident Detection Task (Highest Priority):** Analyzes sensor data for accident detection.
- **Emergency Communication Task (High Priority):** Sends alerts to emergency services.
- **GPS Data Processing Task:** Processes raw GPS data to get accurate location.
- **Data Logging Task:** Stores sensor data and system events.
- **System Monitoring Task:** Monitors system health.

4. Alert Message Content:

- **GPS Coordinates (Latitude and Longitude):** Essential for location.
- **Timestamp:** Time of the accident.
- **Accident Severity:** Categorized (High, Medium, Low, or a numerical scale). This can be estimated based on sensor data (e.g., magnitude of acceleration change, airbag deployment).

5. Communication Protocol:

- **Cellular Communication:**
  - **SMS:** A reliable fallback, but limited data capacity.
- **Emergency Services Integration:** Explore if there are specific protocols or APIs used by local emergency services for automated alerts.

6. Severity Level Classification:

- **High Severity:** Airbag deployment, significant deceleration/acceleration changes, rollover detected.
- **Medium Severity:** High acceleration/deceleration without airbag deployment, sudden change in GPS location combined with unusual motion patterns.
- **Low Severity:** Minor impacts, potential for false positives. Further analysis might be needed.

7. Software Design Details:

- **Real-time Performance:** Task prioritization is critical.
- **Error Handling:** Handle sensor failures, communication issues, and other errors gracefully.
- **Power Management:** Minimize power consumption, especially if battery-powered.
- **Security:** Prevent unauthorized access and malicious attacks.

8. Testing and Validation:

- **Unit Testing:** Individual modules.
- **Integration Testing:** Interactions between modules.
- **System Testing:** Complete system in simulated and real-world (controlled) environments.
- **Field Testing:** Extensive testing in various scenarios

4.3 Flow Diagram

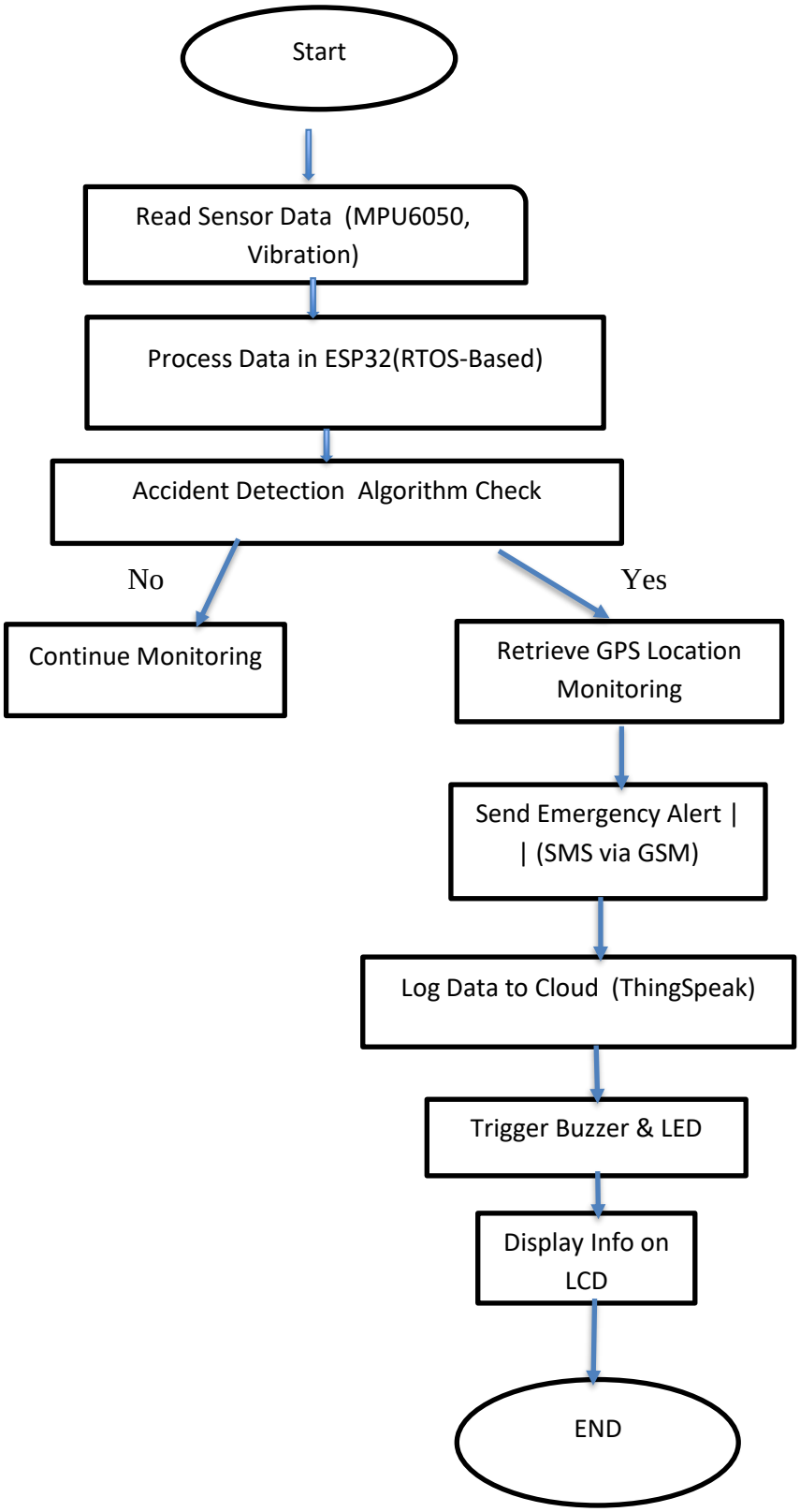

*Fig*4.3 Flow Diagram

Chapter 5

SYSTEM DESIGN

5.1 Introduction

This project harnesses the capabilities of the ESP32 WROOM microcontroller, vibration sensors, and the MPU6050 sensor within an RTOS environment to manage tasks and ensure high-performance operations. Upon detecting an accident, the system captures GPS coordinates and sends SMS alerts via GSM to emergency services while simultaneously logging data to the ThingSpeak cloud platform. By integrating RTOS-based task scheduling, sensor-driven accident detection, and IoT capabilities, the project provides a scalable framework suitable for modern vehicle safety systems. The system's ability to promptly detect accidents and transmit critical information reduces emergency response times, potentially saving lives..

5.2 System Components

The system consists of several hardware and software components that work together to detect accidents, gather location data, and send emergency alerts.

1.Microcontroller Unit (MCU)

- **ESP32-WROOM**
    - Runs **RTOS for real-time task management**.
    - Handles sensor data processing, decision-making, and communication.

2.Sensors for Accident Detection

- **MPU6050 (Accelerometer + Gyroscope)**
    - Detects sudden acceleration changes and vehicle rollover.
    - Measures impact force and tilt angles.
- **Vibration Sensor**
    - Detects strong vibrations or shocks that indicate a collision.
    - Works with a **customizable analog threshold**.

3.GPS Module for Location Tracking

- **NEO-6M GPS Module**

- Provides **real-time latitude and longitude** of the vehicle.
- Helps in **emergency response coordination**.

4 .Communication Modules

- **GSM Module (SIM800/SIM900)**
  - Sends **SMS alerts** to emergency services with GPS coordinates.
  - Can make **emergency calls** if needed.
- **Wi-Fi (ESP32 Built-in)**
  - Connects to the **ThingSpeak cloud** for data logging.

5.Output and Alert Mechanisms

- **LCD/OLED Display**
  - Shows **real-time status**, accident details, and location.
- **Buzzer and LED Indicators**
  - Provides **audible and visual alerts** when an accident is detected.

6.Power Management

- **LM2596 Buck Converter**
  - Provides stable **3.3V/5V** power to all components.
- **Battery Backup (Li-ion / Lead Acid)**
  - Ensures operation during power loss in accidents.

7. Cloud & Data Logging (Optional)

- **ThingSpeak / Firebase**
  - Stores **accident history for analysis**.
  - Provides **remote access to accident records**.

8. Software & Development Tools

- **RTOS (FreeRTOS)** → Manages multiple tasks (sensor reading, GPS, GSM, cloud logging).
- **STM32CubeIDE / Arduino IDE** → Used for coding and flashing ESP32.
- **MATLAB & ThingSpeak** → Data visualization and analysis.

5.3 Data Flow in the System

The system follows a structured data flow from sensor input to emergency alerts and cloud storage. Below is a step-by-step breakdown:

- **Step 1: Sensor Data Acquisition**
  - MPU6050 (Accelerometer & Gyroscope) reads real-time motion and tilt data.
  - Vibration Sensor detects sudden shocks or collisions.
  - The ESP32 microcontroller continuously collects data from both sensors using I2C & GPIO interfaces.
- **Step 2: Data Processing in ESP32 (RTOS Environment)**
  - The RTOS-based system assigns dedicated tasks for:
    - Sensor Reading Task → Collects real-time accelerometer, gyroscope, and vibration data.
    - Accident Detection Task → Compares sensor values with predefined threshold limits.
  - If sensor values exceed the threshold, the system triggers the accident detection sequence.
- **Step 3: GPS Data Retrieval**
  - If an accident is detected, ESP32 requests real-time location data from the GPS Module (NEO-6M) via UART.
  - GPS coordinates (Latitude & Longitude) are retrieved.
- **Step 4: Emergency Alert Transmission**
  - The ESP32 sends an SMS alert via the GSM Module (SIM800) to emergency services.
  - The SMS includes GPS coordinates for quick response.

- (Optional) Emergency Call Feature can also be activated.

🔹 **Step 5: Cloud Data Logging**

- The accident data (sensor readings, time, GPS location) is sent to the cloud (ThingSpeak/Firebase) via Wi-Fi (ESP32 Built-in).
- This enables historical tracking & visualization for further analysis.

🔹 **Step 6: Local Alerts & Display**

- Buzzer & LED Indicators activate to alert nearby people.
- Accident details are displayed on the LCD.

🔹 **Step 7: System Reset & Monitoring Loop**

- The system waits for a reset command or automatically restarts monitoring after a predefined time.

5.4 System Design Principles

Designing an RTOS-Based Traffic Accident Detection System requires following key system design principles to ensure real-time performance, reliability, and scalability. Below are the core principles:

1.Real-Time Operation (Deterministic Behavior)

- Principle: The system must respond within a predictable time frame.
- Implementation:
    - Uses RTOS task scheduling (e.g., FreeRTOS) to handle multiple operations in parallel.
    - Assigns high-priority tasks to accident detection and emergency alerting.
    - Implements interrupts for sensor data processing (MPU6050 & Vibration Sensor).

2.Modularity & Task Separation

- Principle: The system should be divided into independent modules for better maintenance and scalability.
- Implementation:
    - Sensor Handling Task → Reads MPU6050 & vibration sensor data.
    - Accident Detection Task → Runs an algorithm to detect impact.
    - GPS Task → Retrieves real-time location.
    - Communication Task → Sends SMS via GSM & logs data to the cloud.
    - User Interface Task → Displays information on LCD.

3.Fault Tolerance & Redundancy

- Principle: The system must handle failures and ensure continuous operation.
- Implementation:
    - Watchdog Timers (WDT) to reset the system in case of a crash.
    - Redundant Data Check: If one sensor fails, the system verifies data from other sensors.
    - Automatic Reset after sending an emergency alert to resume normal operation.

4.Energy Efficiency

- Principle: The system should optimize power consumption to increase battery life.
- Implementation:
    - Uses low-power modes of ESP32 to reduce energy usage when idle.
    - Event-driven processing (sensors wake up MCU only when needed).
    - Buck Converter (LM2596) efficiently manages power supply.

5.Security & Data Integrity

- Principle: Ensuring data privacy and secure communication is essential.
- Implementation:
    - Uses encryption (AES) for cloud data transmission.
    - Stores backup accident logs in non-volatile memory (EEPROM).
    - Error-checking mechanisms for GSM/GPS data validation.

6.Scalability & IoT Integration

- Principle: The design should allow for future upgrades and expansion.
- Implementation:
    - Supports Wi-Fi & MQTT for IoT-based expansion.
    - Can be integrated with vehicle monitoring systems.
    - Cloud logging with ThingSpeak/MATLAB enables AI-based accident analysis.

7.User-Centric Interface

- Principle: The system should provide clear and user-friendly feedback.
- Implementation:
    - LCD/OLED Display to show accident status and GPS location.
    - Buzzer & LED Indicators to alert nearby people.
    - Mobile App (Future Scope) to track vehicle status remotely.

Chapter 6

IMPLEMENTATION

A smart accident detection and alert system is a system that detects accidents and alerts emergency services by utilising advanced technology such as sensors, GPS, and wireless communication. The implementation of such a system takes multiple stages and necessitates precise planning and execution.

The first step in implementing a smart accident detection and alert system is identifying the system's core components. Sensors to detect accidents, a communication system to provide alerts to emergency services, and a control unit to operate the system are common components. Accelerometers, gyroscopes, and GPS sensors can be employed in the system to detect sudden shifts in acceleration or rotation, as well as the vehicle's location.

After identifying the system's core components, the next step is to design the hardware and software components. This includes selecting the proper sensors and communication equipment, as well as build-ing the algorithms that will analyse sensor data and trigger an alert if an accident occurs. The system's control unit is in charge of coordinating communication between sensors and emergency services, as well as providing an interface for users to engage with the system.

After the hardware and software components are set up, the system must be built and tested. This includes assembling the hardware, programming the software, and testing the system under various cir-cumstances to ensure that it works properly. To ensure that the system can detect accidents and provide notifications to emergency services, it should be tested in both simulated and real-world circumstances.Once built and tested, the system can be installed in vehicles or other locations where accidents are likely to occur. Mounting the sensors in the appropriate locations, connecting the communication devices, and setting the control unit are typical installation steps. After installation, the system should be tested again to ensure that it is working properly.

Finally, to ensure that the system continues to perform properly throughout time, it should be moni-tored and maintained. This includes monitoring the operation of sensors and

communication devices, as well as updating software and hardware as needed. Regular maintenance and upgrades help ensure that the system remains to be reliable and effective in detecting and alerting emergency services to accidents.

Maintenance requirements The frequency and complexity of maintenance operations necessary to ensure the system's efficient operation.Cost The expenses related to installing, running, and maintaining the system.

In conclusion, there are a number of steps involved in the implementation of a smart accident detec-tion and alert system, such as identifying the system's essential components, designing the hardware and software components, building and testing the system, installing the system, and monitoring and maintaining the system over time. A smart accident detection and alert system can be an effective tool for increasing traffic safety and minimising the effects of accidents with careful planning and execution.

6.1 Pin Connection Table: ESP32 WROOM-32 RTOS-Based Traffic Accident Detection System Project

Component	**ESP32 Pin**	**Pin Description**
MPU6050 (I2C)	GPIO 21 (SDA)	Serial Data Line
	GPIO 22 (SCL)	Serial Clock Line
GPS Module	GPIO 16 (RX)	GPS Data Received (UART1)
	GPIO 17 (TX)	GPS Data Transmitted (UART1)
GSM Module	GPIO 26 (TX)	GSM Data Transmitted (UART2)
	GPIO 27 (RX)	GSM Data Received (UART2)
Vibration Sensor	GPIO 13 (DO)	Digital Output Signal
	GPIO 32 (AO)	Analog Output Signal
Wi-Fi Module (Inbuilt)	-	Wi-Fi Communication
Power Supply	3.3V, GND	Power for Modules

Table no 6.1 Pin confirmation

6.2 Result:

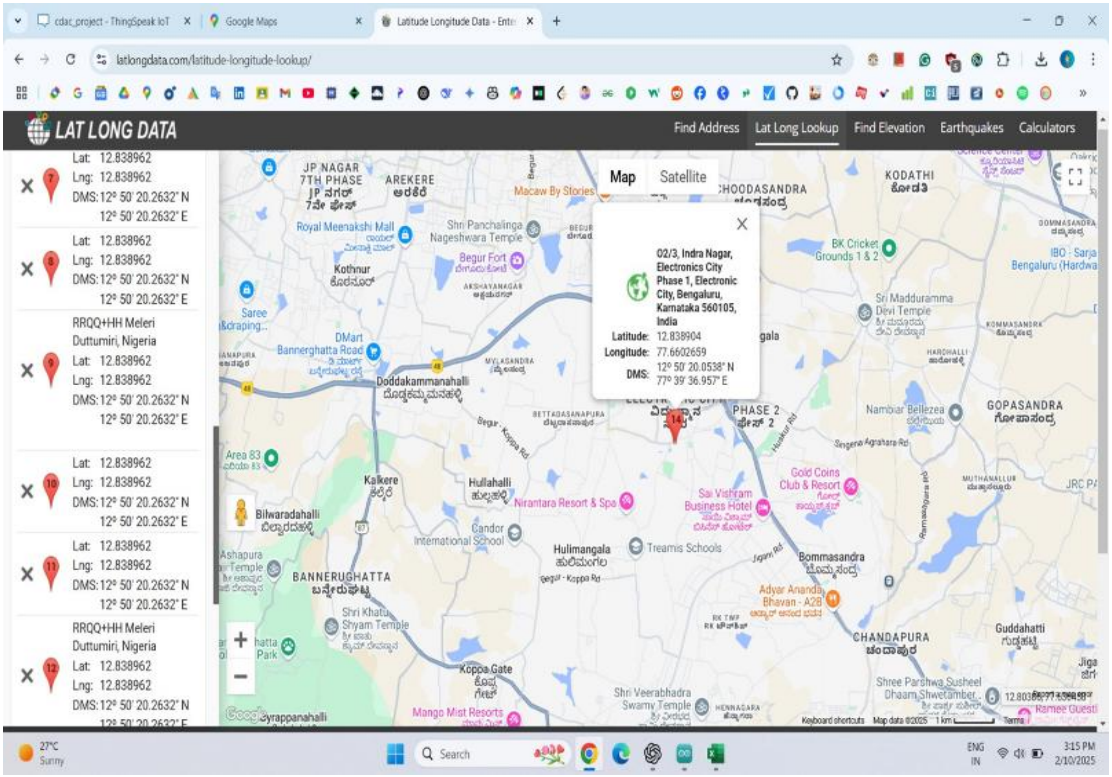

Fig 6.1 result on google map

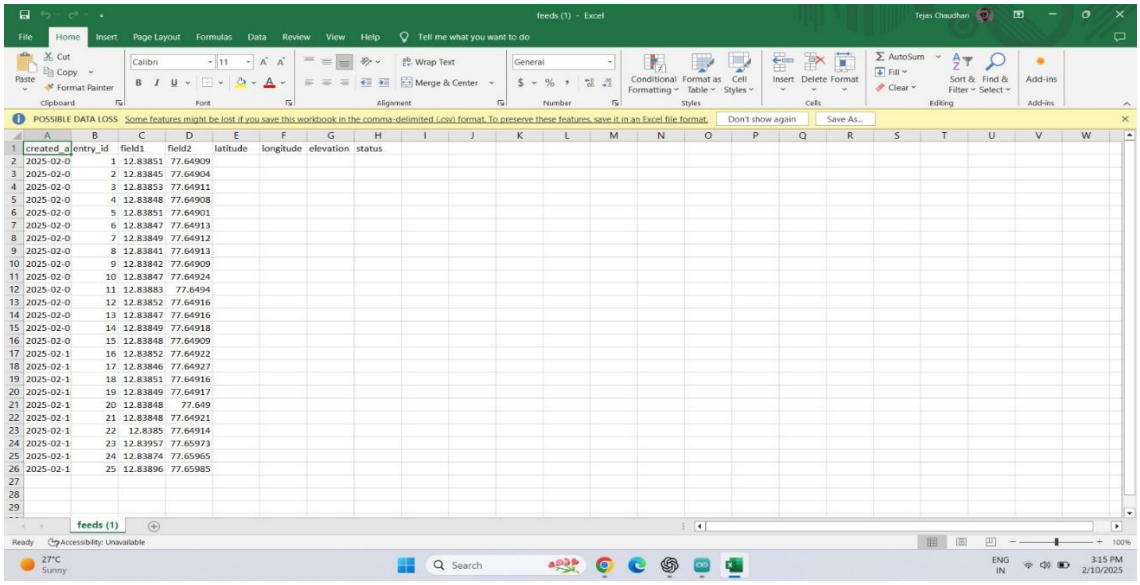

Fig 6.2 Field record data sheet

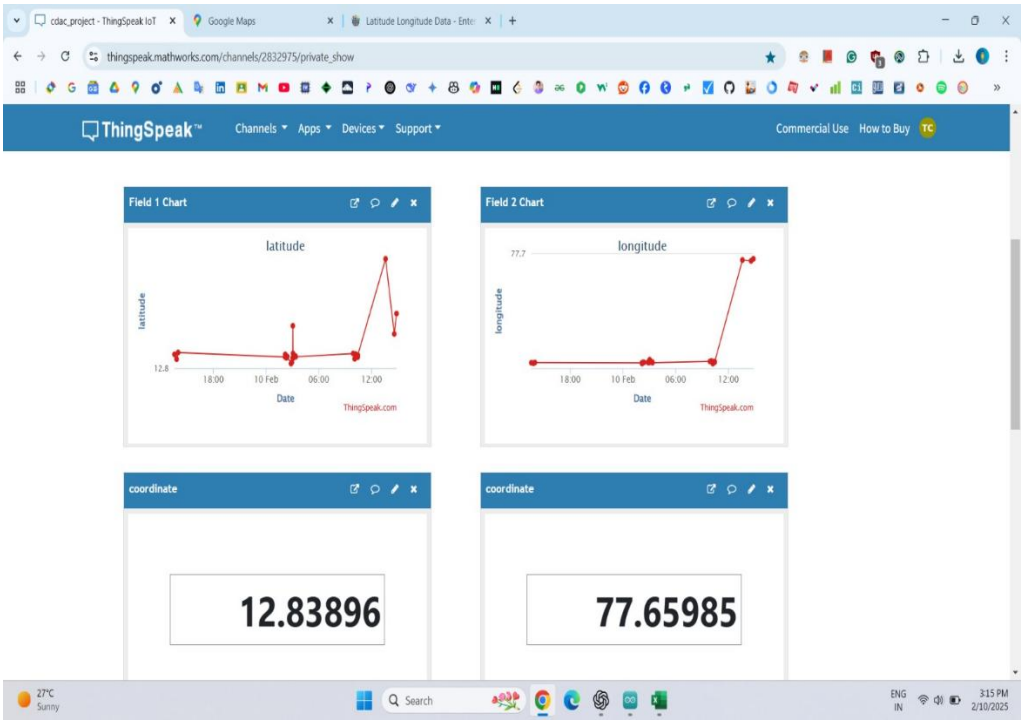

Fig 6.3 Thingspeak Api cloud

6.3 Workflow Diagram

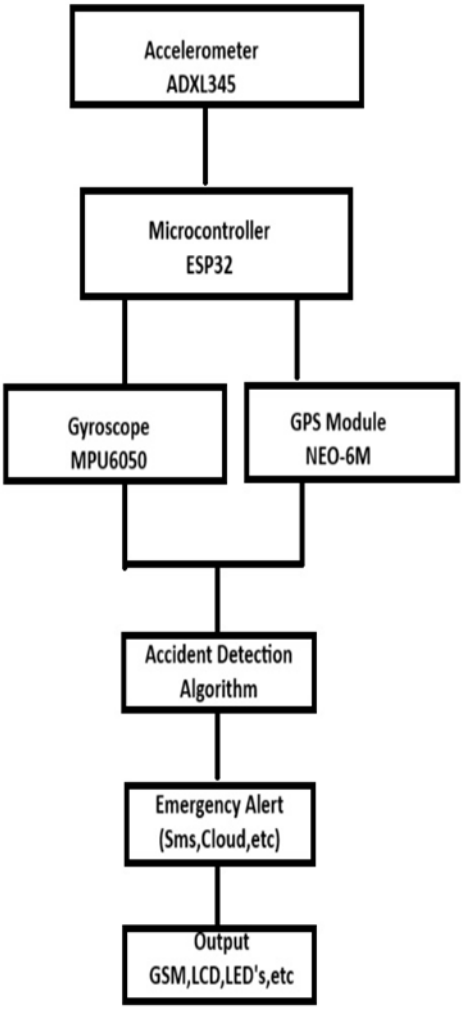

Fig 6.4 Workflow Diagram

Chapter 7

CONCLUSION

7.1 CONCLUSION

The RTOS-Based Traffic Accident Detection System provides a reliable and scalable solution for detecting traffic accidents and alerting emergency services. By integrating advanced sensors, real-time data communication, and RTOS capabilities, the system ensures rapid detection and response to potential road hazards, enhancing road safety.

7.2 FUTURE SCOPE

The accident detection and alert system could benefit from a number of potential upgrades and enhance-ments as technology evolves. Here are a few advancements that can be made to the system in the future:

- The accuracy and dependability of accident detection system is one area for development. In order

to reduce the number of false positives and ensure that emergency services are only notified in the

event of a true emergency, the system must be able to distinguish accidents from other events that

might set off false alarms by using advanced algorithms and machine learning techniques.

- Real-time audio and video recording capabilities could be added to the system as another potential

upgrade. This could provide supportive proof for insurance claims and legal actions, as well as

improving the accuracy of accident reconstructions.

• Integrating this system with advanced driving assistance systems (ADAS) is another prospective

improvement. ADAS systems can give drivers instantaneous information about their surround-

ings, including other vehicles, pedestrians, and potential hazards. Integrating ADAS with this

system may make it possible to identify accidents more quickly and correctly while also giving

emergency services more comprehensive information about the accident.

• Accident detection system might potentially be integrated with techy accessories like smart watch-

es or fitness trackers. Additional information like driver's heart rate and blood pressure readings

could assist emergency responders in determining the severity of the collision and in providing

appropriate treatment.

• Finally, the system might be integrated with autonomous vehicles to trigger emergency responses

such as automatically stopping the vehicle, potentially improving safety.

Chapter 8

REFERENCES

8.1 Documentations

[1] https://www.researchgate.net/publication/372092332_IoT Based_Smart_Accident_Detection_and_Alert_System

[2] International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878 (Online), Volume-11 Issue-6, March 2023

[3] https://ieeexplore.ieee.org/document/9703970

8.2 Websites

[1] https://www.ijrte.org/wp-content/uploads/papers/v11i6/F75060311623.pdf

[2] https://ijcrt.org/papers/IJCRT2207676.pdf

[3]https://www.researchgate.net/publication/309526963_Vehicle_collision_detection_and_lane_assist_system_using_RTOS

8.3 Journal Articles

[1] International Journal of Research Publication and Reviews Journal homepage: www.ijrpr.com ISSN 2582-7421