TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A MINOR PROJECT PROPOSAL

ON

EDR-BASED INTELLIGENT VEHICLE ACCIDENT PREDICTION WITH MULTISENSOR FUSION AND EDGE SUMMARIZATION

Submitted By:

Prenisha Upreti	(THA079BEI029)
Roshan Kr. Gupta	(THA079BEI034)
Suman Phuyal	(THA079BEI042)
Yugal Nyoupane	(THA079BEI048)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

December 2025

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A MINOR PROJECT PROPOSAL

ON

EDR-BASED INTELLIGENT VEHICLE ACCIDENT PREDICTION WITH MULTISENSOR FUSION AND EDGE SUMMARIZATION

Submitted By:

Prenisha Upreti	(THA079BEI029)
Roshan Kr. Gupta	(THA079BEI034)
Suman Phuyal	(THA079BEI042)
Yugal Nyoupane	(THA079BEI048)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

Under the Supervision of

Er. Umesh Kanta Ghimire

December 2025

ACKNOWLEDGEMENT

We would like to thank the Institute of Engineering, Tribhuvan University for providing us the opportunity to implement our knowledge and experience and showcase it into a project this semester.

We would like to express our gratitude to our department **Department of Electronics and Computer Engineering** for providing the opportunity and guiding us through every highs and lows. The department has been guiding and structuring the events to follow up for the minor project in ever so standard way giving us a clear roadmap to do our project.

We would also like to thank your supervisor; **Umesh Kanta Ghimire** for guiding, motivating and listening to the thoughts and creativity that run through our brains.

Prenisha Upreti (THA079BEI029)
Roshan Kr. Gupta (THA079BEI034)
Suman Phuyal (THA079BEI042)
Yugal Nyoupane (THA079BEI048)

ABSTRACT

"EDR (Event Data Recorder) Based Intelligent Vehicle Accident Prediction with Multisensor Fusion and Edge Summarization" highlights the prevention mechanism and alert mechanism in the case of an accident. According to WHO "Approximately 1.19 million people die each year as a result of road traffic crashes." The casualties on the road are often misunderstood and blamed on drivers but the causes are often different. The real cause is never known. The miscommunication and misinformation of vehicle status, in pre-event and post-event of accident often lead into bad accident prevention. This project focuses primarily on prevention and post-event accident event data recorder and alerting via SOS in case of an accident. Multiple sensors like Flame Sensor, Gyroscope, GPS, Crash sensors are connected and embedded with a MultiSensor Fusion algorithm which gives accurate and comprehensive inferences than using some individual sensors with complex algorithms. In a microcontroller, by using the Multifusion algorithm, we integrate a waterfall model for the analytical viewpoint of accident. We use real time data. The real time data is event summarized for multiple instances and is fed into the classifier (Random Forest) for either alerting in case of accident or warning in case of accident prevention. This project aims at solving the manual driven vehicle problem to alert the driver and also aims on being expandable for self-driven cars because accidents can happen to anyone,anything or anyway.

Keywords—*EDR, Multisensor Fusion, Edge Summarization, Sensor, Random Forest*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACC	Accelerometer Sensor
ADC	Analog-to-Digital Converter
AT	Attention (AT Commands)
BLE	Bluetooth Low Energy
CS	Crash Sensor
EDR	Event Data Recorder
ESP-IDF	Espressif IoT Development Framework
ESP32-S3	Espressif 32-bit System on Chip (S3 Variant)
FS	Flame Sensor
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GY	Gyroscope Sensor
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
ML	Machine Learning
MS	Magnetometer Sensor
PWM	Pulse Width Modulation
SIM	Subscriber Identity Module
SOS	Save Our Souls
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver–Transmitter

1. INTRODUCTION

"EDR (Event Data Recorder) Based Intelligent Vehicle Accident Prediction with Multisensor Fusion and Edge Summarization" is a project that focuses on pre-event and post-event analysis and predict the accident and log it. The MultiSensor Fusion gives more accurate data and we can use that data to feed into a real time frame. The edge processing eliminates the unnecessary flagged data that we do not need while analyzing and alerting the system. The traditional project lacked the fusion of two technologies in a single EDR, which just doesn't make this a logger rather make it an intelligent predicting logger further explained in block diagram of methodology.

1.1. Background

Conventional vehicle EDR lacked the intelligent decision making capabilities. The project highlights an intelligent accident intelligent Vehicle Accident Prediction with Multisensor Fusion and Edge Summarization. The accuracy and timely emergency response results in a safety driven concern for the vehicles. The alert system that is classified accordingly makes the use case of this project more practical and more reliable compared to the traditional EDR. Multisensor fusion and edge summarization can improve overall reliability which is more practical in real-world deployment.

1.2. Motivation

This project came into idea when there was a deadly bus accident in Trishuli where at least 54 people went missing after two passenger buses were swept away by a mudslide into the rain-swollen Trishuli River near Simaltal. Nepal police, Armed Police Force and Nepal Army personnel worked tirelessly to find the bus in the swelling river and continuous rainfall but the effort went into vain after the search had no result. Had the rescuers or finders had better technology, the last data recorded, the search would have been more predictive. The timely SOS would have been sent early. This is just an example of a larger problem that occurs in day to day life in a world moving so fast. There also are many instances where the drivers aren't alerted, the state at which they are driving compared to a probabilistic scenario can be dangerous, if we find those pattern and pre-alert the driver , the accident case can be

reduced and possibly save life of thousands travelling. This is why we chose "EDR-Based Intelligent Vehicle Accident Prediction with Multisensor Fusion and Edge Summarization"

1.3. Problem Statement

Existing systems for alerting or logging rely on single sensors or manual reporting, which makes them prone to false alarming and missed detection. The existing system depends on cloud based processing which in case of critical situations is very unrealistic and illogical because the delay in the immediate response required case often lead to major casualties. Safety, risk management and reliability are our main concerns overall. The lack of multisensor Fusion, no edge ML, poor logging are often not considered till date, even if those projects are never brought into a single module that can solve a larger problem scenario. By trying to integrate all the aforementioned problem's solutions into a single module we have tried to solve the existing problem.

1.4. Objectives

The objectives of this project are as follows:

1. To detect vehicle accidents or predict possible accident scenarios using Multi-Sensor Fusion and a Machine Learning model (Random Forest).
2. To log possible accidental parameters or critical spikes in sensor data during an accident or a potential accident scenario for further analysis.

1.5. Scope and Limitations

1.5.1. Scope

The intelligence, detection and logging mechanism are the key features included in our project. We have used technologies like Multisensor Fusion, Edge summarization and Machine Learning model implemented on an ESP32 microcontroller using the TinyML approach. The system performs real-time data acquisition, local processing, alert generation and logging of critical sensor parameters during accident and pre-accident scenarios. The scope of this work is limited to small-scale prototype implementation and controlled testing conditions. The system does not include full-scale vehicle integration, cloud-based analytics, or legal emergency response systems.

1.5.2. Limitations

The project has a lot of scope, but due to time constraints and limited resources, some major components cannot be implemented in this phase. The addition of a camera module and accident detection through image classification using machine learning cannot be implemented yet. Limited resources also restrict us to using simple models rather than more complex ones at this stage of the project. In the future, with adequate time, computational power, and resources, advanced machine learning models along with camera-based image classification can be integrated to enhance accident detection accuracy.

2. LITERATURE REVIEW

This review explores the evolution of vehicle safety systems from simple mechanical trigger-based mechanisms to advanced IoT-enabled sensor integrations and predictive data-driven frameworks.

2.1. Sensor-Based Detection and Localization

The foundation of modern accident detection systems lies in the reliable utilization of motion sensors. Amin et al. [1] developed a cost-effective and reliable accident detection system using Micro-Electro Mechanical Systems (MEMS)-based Inertial Measurement Units (IMU).

2.1.1. Methodology:

The proposed system monitors sudden deceleration, which serves as a primary indicator of a collision. GPS is integrated with the IMU to provide continuous vehicle tracking. A key feature of the methodology is the implementation of a dead reckoning technique, which estimates the vehicle's last known position using IMU data when GPS signals are unavailable, such as in tunnels or dense urban environments.

2.1.2. Outcomes:

The study demonstrated that low-cost MEMS sensors can accurately detect high-impact events while maintaining reliable location estimation. This ensures that precise coordinates can still be transmitted to emergency response units even in areas with poor satellite coverage.

2.2. IoT Integration and Collision Classification

Advancing beyond basic accident detection, Kumar et al. [1] proposed an Internet of Things (IoT)-based Accident Detection and Classification (ADC) system to enhance situational awareness during emergencies.

2.2.1. Methodology:

The system employs sensor fusion by combining data from smartphone-based sensors, such as accelerometers and gyroscopes, with external vehicle-mounted sensors. Machine learning algorithms, including Naïve Bayes and Hidden Markov Models, are used to process this multisource data to detect collisions and classify accident types, such as rollovers, side impacts, and head-on collisions.

2.2.2. Outcomes:

By identifying the type of accident, the system provides emergency medical services with detailed situational information, enabling improved resource planning. For example, rollover accidents may require specialized rescue equipment, whereas minor collisions may only necessitate traffic control intervention.

2.3. Optimized Data Logging

As vehicle sensor systems generate increasing volumes of data, efficient storage management becomes critical. Yao and Atkins [2] addressed this challenge by introducing a Smart Black Box (SBB) system focused on optimizing the storage of high-bandwidth data such as video streams and high-frequency telemetry.

2.3.1. Methodology:

The SBB treats data logging as an optimization problem using a Mealy machine-based state logic model. A circular buffer continuously overwrites normal driving data, while anomalous or high-value events trigger data locking to preserve critical information.

2.3.2. Outcomes:

This selective data retention approach ensures that essential pre-crash and post-crash evidence is preserved while redundant data is discarded. As a result, storage hardware is utilized efficiently without compromising forensic investigation requirements.

2.4. Severity Prediction Using Machine Learning

Accident severity prediction is equally important for proactive safety planning. Yang et al. [3] employed a Random Forest algorithm to predict the severity of traffic accidents.

2.4.1. Methodology:

The model was trained using key variables including accident location, accident type, road conditions, and vehicle speed. Random Forest operates by constructing multiple decision trees and aggregating their outputs to enhance prediction accuracy and robustness.

2.4.2. Outcomes:

The results demonstrated high accuracy in distinguishing between minor and major injury outcomes. Such predictive insights are valuable for smart city infrastructures to identify high-risk zones and improve road safety designs.

2.5. Proposed System Contribution

Building upon these foundational works, this project proposes an integrated vehicle safety system that combines accident prediction, real-time detection, and robust data logging for forensic analysis. A Random Forest-based model is employed to identify potential accident risks and detect collisions in real time. To enhance data reliability, a linear Kalman filter is used to fuse accelerometer, gyroscope, and GPS data, effectively minimizing sensor noise and measurement errors.

Furthermore, the system incorporates an event summarization technique to address data management challenges. This approach prioritizes the storage of critical incident-related data, reducing overall storage requirements by approximately 50% without compromising the integrity of information required for post-accident investigations.

3. SYSTEM ARCHITECTURE AND METHODOLOGY

3.1. Proposed System Architecture

3.1.1. Proposed System Block Diagram

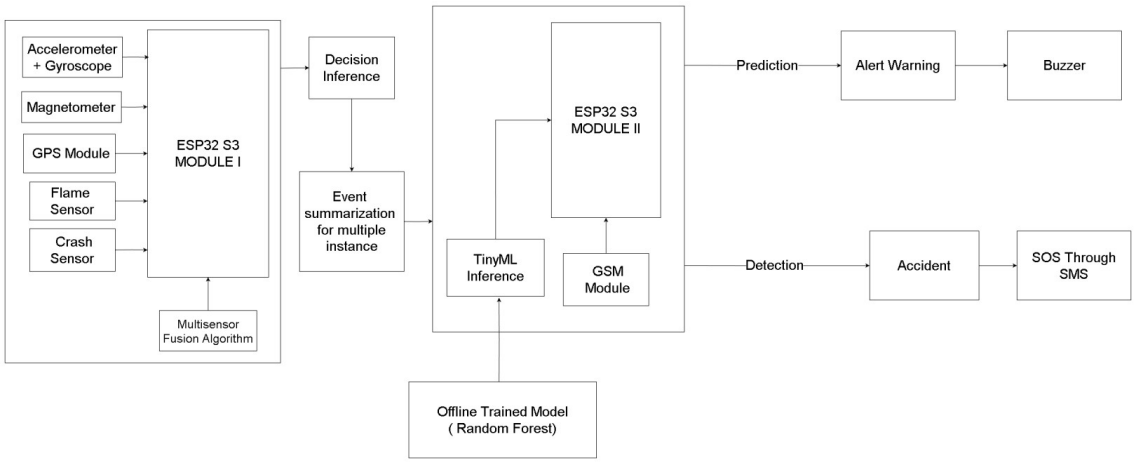


Figure 3.1.: *Block Diagram of Proposed System Architecture*

The block diagram shows us an accident detection and prediction system for vehicles. This system is really good at monitoring vehicles in time using edge computing. The accident detection and prediction system has two parts. Each part of the accident detection and prediction system uses an ESP32 S3 microcontroller. The ESP32 S3 microcontroller is very good at handling the intelligence tasks that the accident detection and prediction system needs. In the beginning Module I is where all the information comes from. It takes in information from sensors that feel movement like the accelerometer, gyroscope and magnetometer. It also gets information from sensors that feel the environment, like the flame and crash sensors. At the time a GPS module keeps track of where the vehicle is. All this information is looked at by the Multisensor Fusion Algorithm. This algorithm helps get rid of any noise and puts all the different pieces of information together. This creates a picture of what is going on with the vehicle at that moment. The Multisensor Fusion Algorithm is very important for the vehicle. It helps the vehicles Module I make

sense of all the information it gets from the motion sensors and environmental sensors. This information is then put together into events. This is better than using lots of pieces of data that do not make sense on their own. The machine learning model can understand events more easily than it can understand these small pieces of data. The machine learning model has a time with events. The main brain of the system is in Module II, where a TinyML Inference engine does its job. This TinyML Inference engine runs a Random Forest model. The Random Forest model was trained on a computer. Then made smaller so it can run on the small hardware of the microcontroller. The system processes data on the device itself which is called the edge. This means the system does not have to wait for the cloud to respond, which is very important when something goes wrong like, during a crash. The TinyML Inference engine and the Random Forest model help the system make decisions quickly.

3.2. Working Principle

The proposed system uses a combination of multisensor fusion and TinyMachine Learning (ML)-based event classification (Random Forest) to monitor vehicles in pre-event and post-event analysis in real time, responding to potential accidents and generating alerts when necessary. The system is built around two microcontrollers, with one responsible for multisensor fusion and the other for verifying accident parameters via the ML model for prediction or alert.

The Espressif 32-bit System on Chip (S3 Variant) (ESP32-S3) Module I acquires continuous data from multiple sensors including Accelerometer Sensor (ACC), Gyroscope Sensor (GY), Magnetometer Sensor (MS), Global Positioning System (GPS), Flame Sensor (FS), and Crash Sensor (CS). These sensors collectively provide comprehensive information regarding the motion, orientation, location, and critical environment conditions of a vehicle. Significant deviations in sensor data are forwarded to the second module using decision inference.

A multisensor fusion algorithm based on the Kalman Filter is employed, integrating readings from all sensors to produce robust and reliable state estimation while minimizing noise and transient errors. Features of potential abnormal events are extracted from raw sensor readings, including acceleration magnitude, orientation angles, sudden speed changes, and activation of crash or flame sensors, forming the input for ML-based classification.

The ESP32-S3 Module II performs the ML inference to classify events as normal, warning, or accident. For normal events, the system continues monitoring without intervention. For warning events, the system triggers a local alert such as a buzzer. When an accident is detected, the system activates the Global System for Mobile Communications (GSM) module to send a Save Our Souls (SOS) message to predefined emergency contacts. A brief observation window ensures persistent abnormal conditions are confirmed before sending

SOS signals, reducing false alarms.

Overall, the system integrates real-time sensor monitoring, robust data fusion, and lightweight ML inference to provide a reliable, fast, and energy-efficient solution for intelligent vehicle accident detection. Preventive warnings and emergency communication improve vehicle safety while maintaining low computational requirements suitable for microcontroller-based embedded platforms.

3.3. Kalman Filter Algorithm

The Kalman Filter is an optimal recursive estimation algorithm that simplifies the computational process of data fusion by introducing assumptions regarding system dynamics, state evolution, and the statistical properties of noise and estimation errors [4]. It provides an efficient solution to the linear quadratic estimation problem for systems affected by stochastic disturbances.

The Kalman Filter formulates estimates of the internal state of a linear dynamic system by processing a sequence of noisy measurements. The system is assumed to be disturbed by zero-mean Gaussian white noise, and the filter recursively updates the state estimates as new measurements become available [5]. Its predictor–corrector structure can be divided into two distinct phases:

- Time Update (Prediction)
- Measurement Update (Correction)

3.3.1. Assumptions

The Kalman Filter operates under the following assumptions [4]:

1. Linear system dynamics:

$$\mathbf{s}_t = \mathbf{A}\mathbf{s}_{t-1} + \mathbf{B}\mathbf{u}_t + \mathbf{w}_t \tag{3.1}$$

2. Linear measurement model:

$$\mathbf{z}_t = \mathbf{H}\mathbf{s}_t + \mathbf{v}_t \tag{3.2}$$

3. Process and measurement noises are mutually independent.

3.3.2. Time Update (Prediction Phase)

State Prediction

$$\hat{\mathbf{s}}_{t|t-1} = \mathbf{A}\hat{\mathbf{s}}_{t-1|t-1} + \mathbf{B}\mathbf{u}_t \tag{3.3}$$

Covariance Prediction

$$\mathbf{P}_{t|t-1} = \mathbf{A}\mathbf{P}_{t-1|t-1}\mathbf{A}^T + \mathbf{Q} \tag{3.4}$$

3.3.3. Measurement Update (Correction Phase)

Kalman Gain

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}^T \left(\mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{3.5}$$

Innovation (Measurement Residual)

$$\mathbf{y}_t = \mathbf{z}_t - \mathbf{H}\hat{\mathbf{s}}_{t|t-1} \tag{3.6}$$

State Update

$$\hat{\mathbf{s}}_{t|t} = \hat{\mathbf{s}}_{t|t-1} + \mathbf{K}_t\mathbf{y}_t \tag{3.7}$$

Covariance Update

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\,\mathbf{P}_{t|t-1} \tag{3.8}$$

3.4. Random Forest Machine Learning Model

A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(x, \Theta_k), k = 1, \dots\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x [6].

The common element in all of these procedures is that for the kth tree, a random vector Θ_k is generated, independent of the past random vectors $\Theta_1, \dots, \Theta_{k-1}$ but with the same distribution; and a tree is grown using the training set and Θ_k, resulting in a classifier $h(x, \Theta_k)$ where x is an input vector. For instance, in bagging the random vector Θ is generated as the counts in N boxes resulting from N darts thrown at random at the boxes, where N is the number of examples in the training set. In random split selection, Θ consists of a number of independent random integers between 1 and K. The nature and dimensionality of Θ depends on its use in tree construction. After a large number of trees is generated, they vote for the most popular class. These procedures are called random forests [7].

3.4.1. Random Forest Convergence

Given an ensemble of classifiers $h_1(x), h_2(x), \ldots, h_K(x)$, and with the training set drawn at random from the distribution of the random vector (Y, X), define the margin function as:

$$\text{mg}(X, Y) = \text{avg}_k I(h_k(X) = Y) - \max_{j \neq Y} \text{avg}_k I(h_k(X) = j), \tag{3.9}$$

where $I(\cdot)$ is the indicator function. The margin measures the extent to which the average number of votes at (X, Y) for the correct class exceeds the average vote for any other class. The larger the margin, the more confidence in the classification. The generalization error is given by:

$$PE^* = P_{X,Y}(\text{mg}(X, Y) < 0), \tag{3.10}$$

where the subscripts X, Y indicate that the probability is over the X, Y space.

In random forests, $h_k(X) = h(X, \Theta_k)$. For a large number of trees, it follows from the Strong Law of Large Numbers and the tree structure that:

Theorem 1. As the number of trees increases, for almost all sequences $\Theta_1, \ldots, PE^*$ converges to

$$P_{X,Y}\Big(P_\Theta(h(X, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(X, \Theta) = j) < 0\Big). \tag{3.11}$$

This result explains why random forests do not overfit as more trees are added, but produce a limiting value of the generalization error [6].

3.4.2. Key Features of Random Forests

- Its accuracy is as good as AdaBoost and sometimes better [6].
- It is relatively robust to outliers and noise [6].
- It is faster than bagging or boosting [6].
- Provides useful internal estimates of error, strength, correlation, and variable importance [6].
- Simple and easily parallelized [7].

3.5. Flowchart of Proposed System Architecture

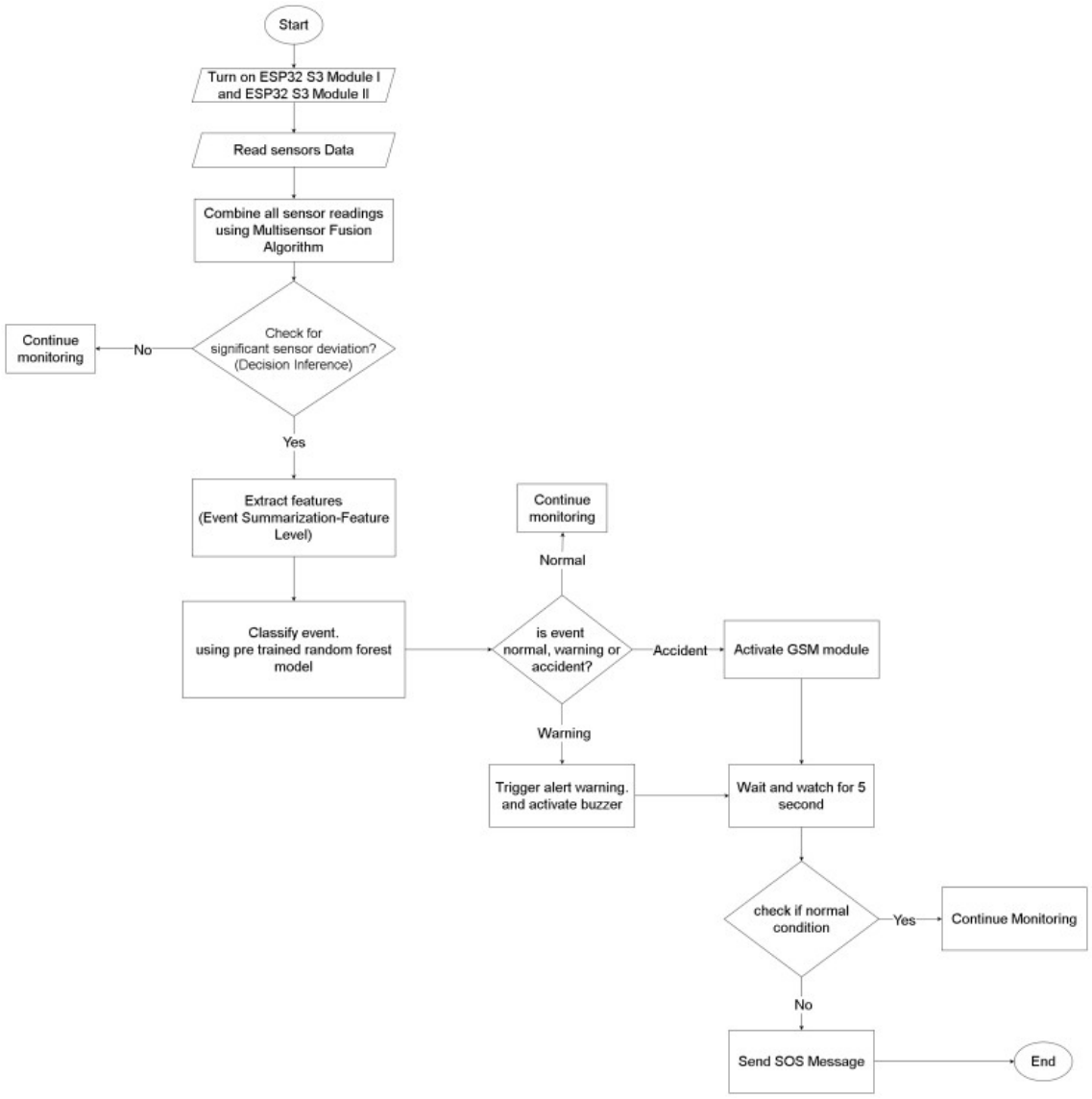


Figure 3.2.: *Flowchart of Proposed System Architecture*

The system utilizes two ESP32 S3 modules to continuously monitor environment data, combining readings through a Multisensor Fusion Algorithm using the ESP32 S3 Module I to identify potential emergencies. When a significant sensor deviation is detected, the system extracts specific data features and employs a pre-trained Random Forest machine learning model done in the ESP32 S2 Module II to categorize the event as Normal, Warning, or Accident. While "Normal" events are ignored, a "Warning" triggers a buzzer and an "Accident" classification activates the GSM module. To prevent false alarms, the system implements a 5-second verification delay; if the situation does not return to normal within this window, it automatically sends an SOS message before ending the cycle.

4. IMPLEMENTATION DETAILS

4.1. Hardware Implementation

4.1.1. Accelerometer and Gyroscope Sensors

An ACC and GY together form an Inertial Measurement Unit (IMU) used to measure linear acceleration and angular velocity of the vehicle. These sensors continuously monitor dynamic motions such as acceleration, braking, skidding, tilting, sharp turns, rollover, and collision forces.

The accelerometer and gyroscope together measure both linear and rotational motion of the vehicle, enabling accurate detection of sudden impacts and abnormal movements. They provide continuous, low-noise motion data that is well suited for real-time processing while maintaining low power consumption and minimal computational overhead. Due to their compact size, cost-effectiveness, and ease of integration with microcontrollers, these sensors are widely used in embedded safety systems. Furthermore, the combined motion data from the ACC and GY is highly compatible with both rule-based and ML-based accident detection systems. This integrated sensing capability allows effective feature extraction for identifying unsafe driving behavior, loss of vehicle stability, and crash events, making the IMU a key component in accident prediction and vehicle safety applications.
The MPU6050 is a 6-axis MEMS IMU that integrates a 3-axis ACC and a 3-axis GY along with an internal Digital Motion Processor (DMP). It outputs motion data digitally via the I^2C interface. By combining acceleration and rotational data, it effectively detects sudden motion changes and crash occurrences.

I^2C Interface

In today's technological world, many applications such as auto mobiles, laptops, embedded devices and other peripherals need to connect multiple devices. We also need to make them able to communicate with each other and this requirement gives rise to the need for any medium or channel which can act as a bridge between these peripherals to share data or information. There are many communication protocols for this purpose. One of them is Inter-Integrated circuit (I2C or IIC) protocol. I2C has two-wires named SDA (serial data line) and SCL (serial clock line), the bidirectional serial bus that provides a simple and

efficient communication between devices. It is multi-master and multi-slave protocol but single-master and the multi-slave combination are mostly used. Master is a device which initiates transactions and generates a clock signal. The slave is a device which is being addressed by the master. [8]

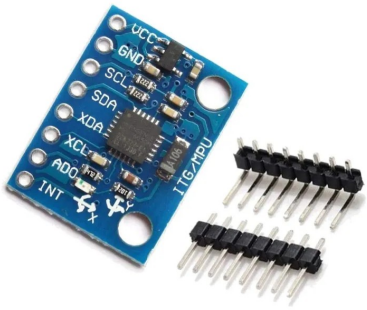


Figure 4.1.: *MPU6050 Accelerometer and Gyroscope Module*

4.1.2. Magnetometer Sensor

A MS measures the strength and direction of a magnetic field, typically the Earth's magnetic field, to determine vehicle orientation and heading.

The magnetometer provides an absolute heading reference by measuring the Earth's magnetic field, which is essential for determining the vehicle's orientation. It improves the long-term stability of motion estimation by compensating for drift errors that accumulate in accelerometer and gyroscope data. The sensor operates with low power consumption while delivering high-resolution digital output, making it suitable for continuous real-time applications. Additionally, magnetometers are cost-effective and highly compatible with ML-based systems, as the heading and magnetic field features can be effectively utilized for orientation-aware accident prediction and vehicle safety analysis.
The HMC5883L is a 3-axis MEMS MS that uses magnetoresistive sensing elements to measure magnetic field components along the X, Y, and Z axes and provides digital output through the I^2C interface.

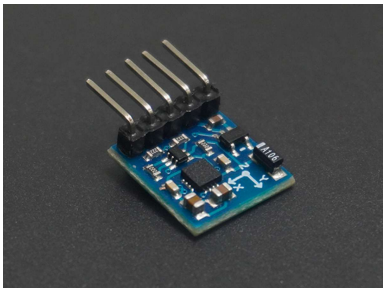


Figure 4.2.: *HMC5883L Magnetometer Module*

4.1.3. GPS Sensor

A GPS sensor determines the absolute geographical position of a vehicle by receiving signals from multiple satellites. It provides latitude, longitude, speed, altitude, and timestamp information essential for vehicle tracking and accident localization.
The GPS module provides accurate real-time latitude and longitude information, enabling precise tracking of the vehicle's geographical location. It also measures vehicle speed and heading, which are essential parameters for understanding motion patterns and driving behavior. In the event of an accident, the GPS module enables exact location reporting, allowing emergency services to respond quickly and efficiently. The module operates with low power consumption, making it suitable for continuous tracking applications. Additionally, the stable and reliable data output from the GPS module is well suited for ML-based processing, supporting location-aware accident prediction and post-accident analysis.

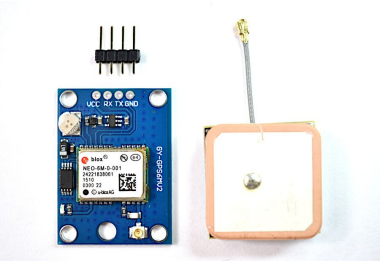


Figure 4.3.: *GPS NEO-6M Module*

4.1.4. Flame Sensor

A FS detects the presence of fire by sensing infrared radiation emitted during combustion. It is critical for identifying fire hazards following vehicle collisions.

The flame sensor enables early detection of fire by identifying infrared radiation emitted from flames, which is critical for enhancing vehicle safety during post-crash or fault conditions. It offers a fast response time, allowing the system to react promptly to fire-related hazards. The sensor features a simple interface that can be easily integrated with microcontrollers, while maintaining low power consumption and cost. Additionally, the output from the flame sensor is suitable for both rule-based and ML-based decision systems, supporting reliable fire detection and emergency response in accident prediction and vehicle safety applications.

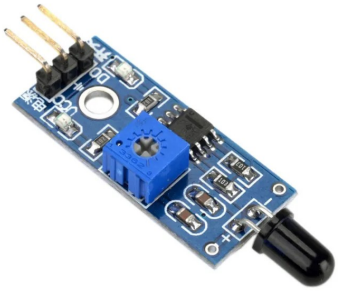


Figure 4.4.: *Flame Sensor Module*

4.1.5. Crash Sensor

A CS, also known as an impact sensor, detects sudden mechanical shocks or collision forces acting on the vehicle.
The crash sensor enables immediate detection of collision events by sensing sudden impact forces acting on the vehicle. It provides high reliability in impact sensing, ensuring accurate identification of crash conditions with minimal false triggers. The sensor requires very low processing overhead, making it suitable for real-time embedded systems. Its fast response time allows rapid activation of emergency mechanisms such as alerts and SOS signaling. Additionally, the crash sensor is cost-effective and durable, and it provides strong support for decision inference when combined with multisensor fusion and ML-based accident detection frameworks.

Figure 4.5.: *Crash Sensor Module*

4.1.6. GSM Module

A GSM module enables embedded systems to send and receive data over cellular networks using a Subscriber Identity Module (SIM). It allows communication with predefined contacts such as emergency services and family members.

The GSM module communicates with the controller via Universal Asynchronous Receiver–Transmitter (UART) using Attention (AT Commands) (AT) commands. Once a SIM is inserted, the module registers with the nearest cellular base station. During emergency events such as crashes or fire detection, the module automatically transmits SMS alerts to predefined mobile numbers.

The GSM module enables long-distance communication without dependency on internet connectivity, making it suitable for vehicle safety applications in diverse environments. It supports real-time transmission of emergency alerts through SMS, ensuring timely notification to predefined contacts or emergency services. The module operates reliably in rural and remote areas where internet access may be limited. Its simple interface allows easy integration with microcontrollers, while maintaining low power consumption. Additionally, GSM modules are cost-effective and widely supported, making them a practical choice for reliable emergency communication in accident detection and response systems.

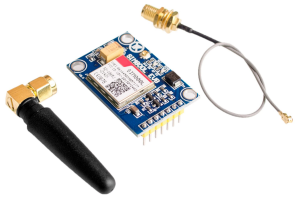


Figure 4.6.: *GSM SIM800L Module*

4.1.7. ESP32-S3 Microcontroller

The ESP32-S3 is a high-performance, low-power microcontroller developed by Espressif Systems for advanced embedded and IoT applications. It integrates wireless connectivity and AI acceleration features, making it suitable for real-time sensing and edge intelligence.

The ESP32-S3 is based on a dual-core Xtensa® 32-bit LX7 processor operating up to 240 MHz. It includes built-in Wi-Fi and Bluetooth Low Energy (BLE), along with rich peripheral support such as Serial Peripheral Interface (SPI), I^2C, UART, Analog-to-Digital Converter (ADC), Pulse Width Modulation (PWM), and General Purpose Input/Output (GPIO)s. Support for external flash and PSRAM enables efficient handling of sensor data and ML models.

The ESP32-S3 offers high processing capability, making it suitable for handling real-time sensor data acquisition and decision-making tasks. It is optimized for ML inference,

enabling efficient execution of lightweight machine learning models directly on the device. The microcontroller supports simultaneous interfacing with multiple sensors, which is essential for multisensor fusion in vehicle safety systems. Additionally, the ESP32-S3 operates with low power consumption, making it well suited for continuous monitoring applications in embedded and edge-based accident detection systems.

The ESP32-S3 is specifically designed for edge ML applications. Its LX7 processor supports vector instructions that accelerate mathematical operations used in ML inference. Frameworks such as TensorFlow Lite for Microcontrollers and ESP-DSP enable deployment of trained models directly on the device. In this project, the ESP32-S3 performs real-time sensor fusion, feature extraction, and decision inference locally, reducing latency and eliminating cloud dependency. This makes it ideal for safety-critical applications such as crash and fire detection.

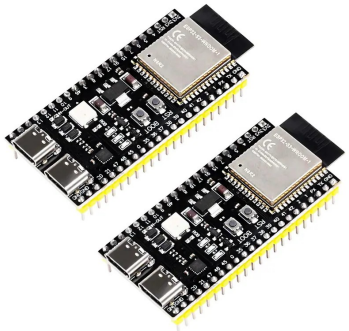


Figure 4.7.: *ESP32-S3 Module*

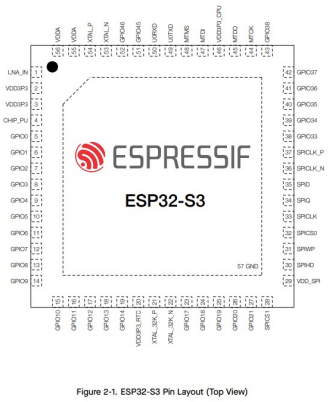


Figure 4.8.: *ESP32-S3 Configuration*

The ESP32-S3 is a highly integrated system-on-chip (SoC) designed for embedded and IoT applications, offering a versatile pin configuration to support a wide range of peripherals and interfaces. The device operates on a 3.3 V power supply, provided through multiple

power pins such as VDD3P3, VDD3P3_CPU, and VDD_SPI, with a common ground reference to ensure stable operation. The CHIP_PU pin controls the enable and reset functionality of the chip, allowing controlled startup and power management. Clocking is achieved through the XTAL_P and XTAL_N pins, which connect to an external high-frequency crystal oscillator, while optional low-power timing support is available through the XTAL_32K_P and XTAL_32K_N pins.

The ESP32-S3 provides a large number of general-purpose input/output (GPIO) pins that can be flexibly configured for digital input and output, analog-to-digital conversion, pulse-width modulation, and various communication protocols including UART, SPI, and I^2C through an internal GPIO matrix. Native USB 2.0 full-speed support is integrated into the chip and is enabled through the USB D+ and USB D− pins, allowing direct USB communication without the need for an external USB-to-UART interface. External non-volatile memory and optional PSRAM are interfaced through dedicated SPI pins, which are typically reserved for memory operations. Debugging and in-circuit testing are supported through JTAG interface pins, while wireless communication is facilitated via the `LNA_IN` radio-frequency input connected to the antenna circuitry.

4.2. Software Implementation

The software implementation involves programming the ESP32-S3 microcontroller using the Arduino Integrated Development Environment (IDE) and Espressif IoT Development Framework (ESP-IDF). The code integrates sensor data acquisition, preprocessing, feature extraction, and decision-making algorithms for accident and fire detection. The ML model is trained offline using Python-based libraries such as NumPy and Pandas, and then converted to a format compatible with the ESP32-S3 for deployment. The system continuously monitors sensor inputs, processes the data in real-time, and triggers alerts via the GSM module when hazardous events are detected.

4.2.1. Arduino IDE

Arduino IDE is a widely used software platform for writing, compiling, and uploading programs to microcontroller boards such as Arduino and NodeMCU. It provides a simple and intuitive environment that allows developers to efficiently develop embedded applications. The IDE includes a built-in code editor with syntax highlighting, a compiler, and tools for uploading firmware to hardware devices. It also supports a large number of libraries, enabling easy integration of sensors, communication modules, and peripheral devices. Arduino IDE is compatible with multiple operating systems, making it accessible and convenient for developers.

4.2.2. TensorFlow

TensorFlow is an open-source software framework developed by Google for numerical computation and data processing. It provides a flexible platform for handling large datasets, performing mathematical operations, and building computational models. TensorFlow offers a rich set of libraries and tools that help developers process and analyze data efficiently. In this project, it is used as a supportive software tool for data handling and model development during the system design and evaluation phase.

4.2.3. VS code IDE

VS Code IDE is a powerful and versatile code editor developed by Microsoft. It provides a rich set of features for writing, debugging, and managing code across various programming languages. VS Code supports extensions that enhance its functionality, including support for Python development, which is essential for data analysis and ML model development in this project. The IDE offers integrated terminal access, version control integration, and customizable settings, making it a preferred choice for developers working on complex software projects.

4.2.4. Jupyter Notebook

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It is widely used for data analysis, scientific computing, and machine learning tasks. In this project, Jupyter Notebook serves as a supportive tool for exploratory data analysis, feature engineering, and model training. It provides an interactive environment where developers can write and execute Python code, visualize data using libraries like Matplotlib and Seaborn, and document their findings in a structured format.

4.3. Datasets

The proposed system will be using a sensor-based dataset to train a TinyML model relying on the accelerometer, gyroscope, GPS data. The dataset we use will not be the data under crash condition as these kinds of data are difficult to find on internet and difficult to collect ourselves. The better option would be to use the dataset under normal driving conditions.

The data was found on the website “Kaggle”. This data contains the motion sensor-based data collected during various driving conditions including normal vehicle movement sharp turns, road bumps which are essential for the accurate prediction of accident. [9]

As given on the website, the devices were fitted on the vehicle as shown in the figure below. Data of many different components including accelerometer, gyroscope, GPS, camera was collected. We plan to use only the data of accelerometers, gyroscope and GPS. [10] The data were produced in three different vehicles, with three different drivers, in three different environments in which there are three different surface types, in addition to variations in conservation state and presence of obstacles and anomalies, such as speed 20 bumps and potholes. The data was collected from three different types of roads i.e. Asphalt Road, Dirt Road and Cobblestone Road. The dataset contains the data of good roads, bad roads, and regular roads. Speed bumps in Cobblestone and Asphalt roads are also included in the dataset. It is also inclusive of driving events, such as lane change, braking, skidding, aquaplaning, turning right or left etc.

# AccX Acceleration on X axis.	# AccY Acceleration on Y axis.	# AccZ Acceleration on Z axis.	# GyroX Rotation on X axis.	# GyroY Rotation on Y axis.	# GyroZ Rotation on Z axis.	Class Driving behavior class.	# Timestamp Unit of time provided by android sensors.
-4.85 5.86	-7.62 4.31	-6.11 5.56	-0.51 1.49	-1.31 1.71	-0.98 0.72	SLOW 41% NORMAL 32% Other (814) 26%	819k 821k
0.75819385	-0.2177906	0.45726347	0.0	0.0	0.0	AGGRESSIVE	818922
0.6675599	-0.03861046	0.23141575	-0.054367086	-0.0077121737	0.22525656	AGGRESSIVE	818923
2.7244492	-7.5841208	2.390926	0.023823744	0.01366811	-0.038026363	AGGRESSIVE	818923
2.33095	-7.6217537	2.5290236	0.05681047	-0.18058704	-0.05207626	AGGRESSIVE	818924
2.8472152	-6.7556214	2.22464	-0.03176499	-0.03520111	0.035277467	AGGRESSIVE	818924
0.44222856	-1.6468334	0.06529236	-0.01038471	-0.024205536	0.0010690142	AGGRESSIVE	818925
0.2383908	0.029413342	0.33282852	0.0012217305	-0.015042556	0.023060163	AGGRESSIVE	818925
0.0434494	0.22891134	0.35466862	0.070249505	0.050320026	0.07742717	AGGRESSIVE	818926
-0.095363855	-0.027441323	-0.039779663	0.040927973	-0.010155635	0.1366811	AGGRESSIVE	818926
0.27051997	-0.07373035	0.0030345917	0.0036651916	-0.0077121737	0.2576324	AGGRESSIVE	818927
0.5129715	0.1584896	0.2572117	0.040927973	-0.005879578	0.33215797	AGGRESSIVE	818927
0.3643882	0.23400217	-0.8731127	0.09840806	0.052763484	0.41218132	AGGRESSIVE	818928
-0.9127637	-0.3456328	0.83376026	0.0580322	-0.02603813	0.3669773	AGGRESSIVE	818929
0.030257344	0.041704893	0.77029896	0.06291912	-0.022983804	0.19837849	AGGRESSIVE	818929
-0.31680942	2.2246227	1.0167103	-0.04337143	0.07047858	0.09636399	AGGRESSIVE	818930
-0.5736394	1.895479	0.25301552	0.002443461	0.019165898	4.5814895E-4	AGGRESSIVE	818930
-0.28935122	-0.04953766	-1.1415176	-0.04153884	-0.048640147	-0.06429356	AGGRESSIVE	818931
0.009707689	-2.3025599	-1.537632	-0.03176499	-0.10667235	-0.06734789	AGGRESSIVE	818931

Figure 4.9.: *Datasets*

5. EXPECTED OUTCOMES

5.1. Accident Prediction and Crash Detection

The system is expected to analyze multisensory fused data including ACC, GY, heading direction, fire detection (FS), and crash impact signals (CS) to compute a real-time accident risk score. Based on this risk score, the system will predict potential accident scenarios and detect crash occurrences under impact conditions using ML-based inference.

5.2. Event Data Recorder (EDR)

The Event Data Recorder (EDR) module is expected to continuously record and store structured sensor and event data in real time. When an accident occurs, the EDR preserves critical event information, including sensor readings and system decisions. This recorded data provides reliable evidence for post-accident analysis and investigation.

5.3. Alert and SOS Signaling Mechanism

The system is expected to trigger real-time alerts or warning signals, such as buzzer activation, to notify the driver upon detecting high accident risk scenarios. In the event of a confirmed accident, the system generates SOS signals and transmits them to nearby emergency helplines through the GSM module.

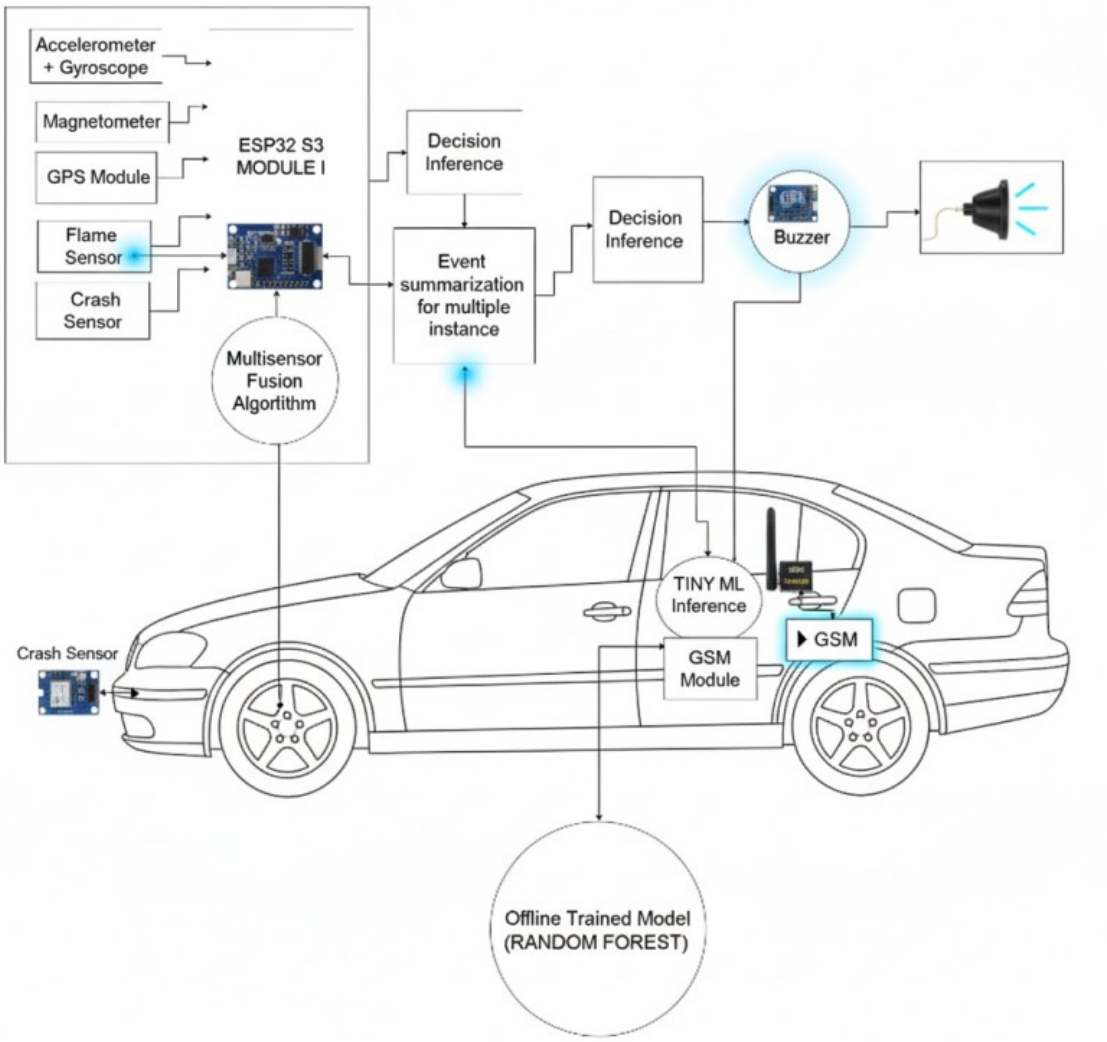


Figure 5.1.: *Expected Outcomes of the Proposed Vehicle Safety System*

A. PROJECT BUDGET

The total budget required for the successful completion of the project is estimated to be between **NPR 7000 - NPR 11000**. This budget covers various aspects of the project, including hardware purchases, component materials, software services, and miscellaneous expenses.

A.1. Bill of Materials (BOM)

This section details all components that will be integrated into the final product/system. These are the materials that constitute the deliverable hardware (if any).

Table A.1.: *Project Budget*

SN	Components	Qty	Unit Cost (NPR)	Total (NPR)
1	ESP32 S3	2	1500	3000
2	GSM800L	1	1000	1000
3	GPS	1	800	800
4	MPU6050	1	700	700
5	HMC5883L	1	450	450
6	Flame Sensor	1	320	320
7	Crash Sensor	1	100	100
8	SIM Card	1	100	100
9	Bread Board	1	450	450
10	Resistor	1	150	150
11	Buzzer	1	100	100
12	Wire	1	200	200
13	Power Supply	1	400	400
			Total	**7770**

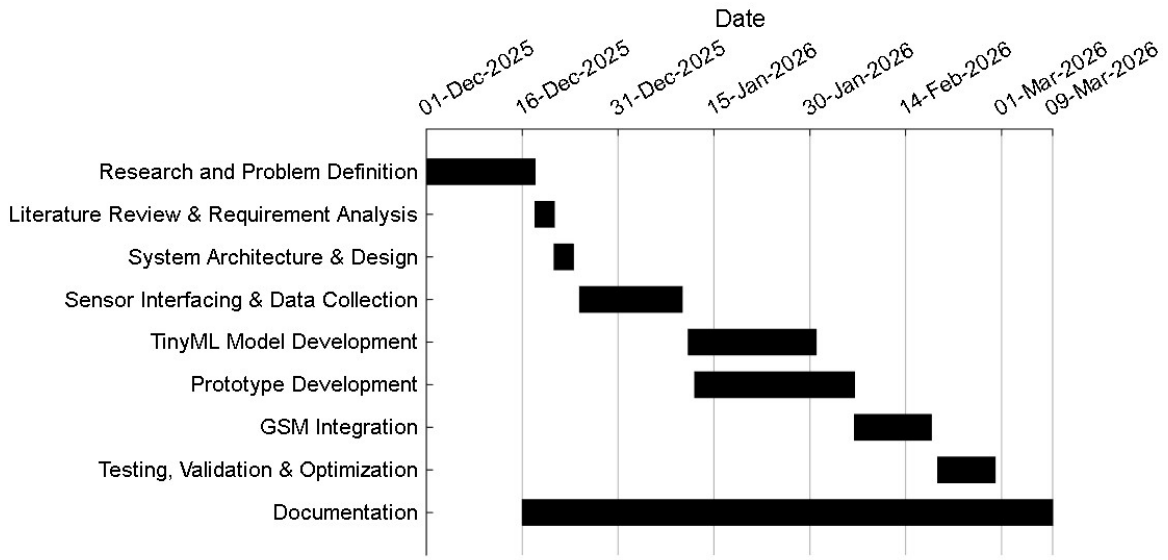
B. PROJECT TIMELINE

The project will be executed over a period of approximately 6 months, divided into several key phases. Each phase includes specific tasks and milestones to ensure steady progress toward project completion.

B.1. Gantt Chart

This is the estimated timeline for the project, detailing the key phases and milestones. The project is structured into several stages, each with specific tasks and deliverables.

Table B.1.: *Project Gantt Schedule*



C. FEASIBILITY

C.1. Technical Feasibility

The proposed system is implemented using the ESP32-S3 microcontroller integrated with multiple sensors, including ACC, GY, MS, FS, CS, an ultrasonic sensor, along with GPS and GSM modules. The software development is carried out using open-source platforms such as the Arduino IDE and ESP-IDF, while Python-based libraries including NumPy and Pandas are used for sensor data analysis and ML model development. Training and validation datasets are obtained from open-source repositories such as Kaggle and supplemented with real-time sensor data, ensuring reliable model performance and real-world applicability.

C.2. Economic Feasibility

All hardware components used in the project are readily available and cost-effective, making the overall system economically viable for large-scale deployment. The use of free and open-source software tools eliminates licensing costs, further reducing the total project expenditure. This cost-efficient design supports practical implementation in real-world vehicle safety applications.

C.3. Operational Feasibility

The system operates using a SIM card and a GSM-enabled mobile network, allowing reliable communication without internet dependency. The GSM module enables the ESP32-S3 to transmit alerts, SOS messages, and GPS-based location information to predefined contacts or emergency services during accident or fire detection events. This ensures continuous and effective system operation in both urban and remote environments.

REFERENCES

[1] N. Kumar, D. Acharya, and D. Lohani, "An iot-based vehicle accident detection and classification system using sensor fusion," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 869–880, 2021.

[2] Y. Yao and E. Atkins, "The smart black box: A value-driven high-bandwidth automotive event data recorder," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1484–1496, 2021.

[3] J. Yang, S. Han, and Y. Chen, "Prediction of traffic accident severity based on random forest," *Journal of Advanced Transportation*, vol. 2023, p. Article ID 7641472, 2023.

[4] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[5] C. Montella, "The kalman filter and related algorithms: A literature review," University Research Report, Tech. Rep., 2011.

[6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] T.Rajendran. (2025) Vehicle accident detection using random forest. Accessed: 2025-12-25. [Online]. Available: https://www.scribd.com/document/793483984/20

[8] V. K. Pandey, S. Kumar, V. Kumar, and P. Goel, "A review paper on i2c communication protocol," *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 4, no. 2, 2018, paper ID: V4I2-1286. [Online]. Available: https://www.ijariit.com/manuscripts/v4i2/V4I2-1286.pdf

[9] OutOfSkills. (2025) Driving behavior dataset. Accessed: 2025-12-25. [Online]. Available: https://www.kaggle.com/datasets/outofskills/driving-behavior

[10] G. R. Menon. (2025) Passive vehicular sensor eda. Accessed: 2025-12-25. [Online]. Available: https://www.kaggle.com/code/gautamrmenon/passive-vehicular-sensor-eda/notebook