# Smart Accident Detection and Window Safety System

25-26J-282

Thanaweera Arachchige Paduma Kasun Shameera.

IT22606624

B.Sc. (Hons) Degree in Information Technology Specialized Information Technology

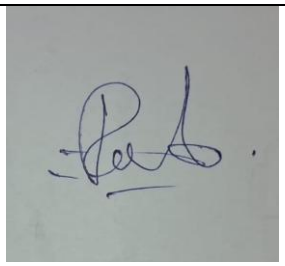
Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

August 2025

## DECLARATION

I declare that this is my own work and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

| **Name** | **Student ID** | **Signature** |
|---|---|---|
| Thanaweera Arachchige Paduma Kasun Shameera | IT22606624 |  |

Signature of the Supervisor (supervisor name)

Date

…… ……

## ABSTRACT

Student safety during school bus transportation is an important concern because children are vulnerable to accidents, unsafe movements, and risky behaviour inside the bus. In many school buses, accident detection and student safety monitoring are still mainly dependent on manual observation by the driver or supervisor. This approach can be unreliable, especially when the driver is focused on road conditions. To address this issue, this research presents a smart accident detection and machine learning based window safety module for the SISURAKSHA school bus safety system.

The proposed module consists of two main components. The first component is a smart accident and emergency detection system developed using an ESP32 microcontroller and multiple sensors, including MPU9250, 801S vibration sensor, piezoelectric sensor, force sensitive resistors, MQ2 gas sensor, fire sensor, raindrop sensor, GPS module, and SIM800L GSM module. The system detects accident-related and emergency conditions such as rollover, sudden speed reduction, abnormal vibration, shock impact, high pressure, fire, smoke, and water-related hazards. To reduce false accident alerts caused by potholes, rough roads, or non-critical sudden braking, a 60-second driver verification mechanism is included using a push button cancellation method. If the alert is not cancelled within the given period, the system confirms the incident and sends an emergency SMS alert with the bus location.

The second component focuses on window safety using a machine learning based camera monitoring approach. Two cameras are positioned to monitor bus window areas and identify unsafe student behaviour, such as placing hands, head, or other body parts outside the window. When such unsafe behaviour is detected, the system generates real-time audio warnings through the speaker to alert students immediately and can also notify the driver dashboard. The proposed module improves school bus safety by combining sensor-based accident detection, emergency alert communication, false alert reduction, and intelligent window safety monitoring.



**Keywords - SISURAKSHA, School Bus Safety, Accident Detection, ESP32, Machine Learning, Window Safety.**

## ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my research supervisor for the continuous guidance, encouragement, and valuable feedback provided throughout the development of this research project. The support and advice received during each stage of the project helped me to improve the quality of the research and complete my individual component successfully.

I would also like to extend my appreciation to the academic staff of the Department of Information Technology, Sri Lanka Institute of Information Technology, for providing the required knowledge, resources, and academic guidance throughout the research period. Their support was valuable in understanding the research process, system design, implementation, and documentation.

My sincere thanks go to my group members of the SISURAKSHA research project for their cooperation, teamwork, and contribution throughout the project. Their ideas, discussions, and support helped in developing the overall school bus safety system effectively. I am especially thankful for the collaboration that allowed my individual component, the smart accident detection and machine learning based window safety module, to be integrated into the overall SISURAKSHA system.

I would also like to thank my friends and colleagues who supported me by sharing feedback, suggestions, and motivation during the development and testing stages of this project. Their encouragement helped me to overcome challenges faced during hardware integration, sensor testing, and system implementation.

Finally, I am deeply grateful to my family for their patience, understanding, and continuous encouragement throughout this research project. Their support gave me the strength and motivation to complete this dissertation successfully. Without the guidance, support, and encouragement of all the above individuals, the successful completion of this research project would not have been possible.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| Abbreviation | Full Term |
|---|---|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| BMP | Barometric Pressure Sensor |
| CNN | Convolutional Neural Network |
| DFPlayer | Digital File Player Mini Audio Module |
| ESP32 | Espressif 32-bit Microcontroller |
| FSR | Force Sensitive Resistor |
| GPIO | General Purpose Input Output |
| GPS | Global Positioning System |
| GSM | Global System for Mobile Communication |
| IDE | Integrated Development Environment |
| IoT | Internet of Things |
| ML | Machine Learning |
| MQ2 | Gas and Smoke Sensor Module |
| SMS | Short Message Service |
| YOLO | You Only Look Once |

# LIST OF APPENDICES

# 1 INTRODUCTION

## 1.1 General Introduction

Student transportation is an essential part of the modern education system, especially for school children who depend on school buses as their daily mode of travel. A school bus is expected to provide a safe, reliable, and controlled transportation environment for students. However, children may face several safety risks during transportation, including road accidents, sudden braking, vehicle collisions, rollover situations, fire hazards, flood or water-related emergencies, and unsafe behaviour inside the bus. Since school children are young and may not always understand the consequences of unsafe actions, continuous monitoring and quick emergency response are highly important.

In many school transportation environments, safety monitoring is still mainly dependent on manual supervision by the driver, conductor, teacher, or bus assistant. Although manual supervision is useful, it has several limitations. The driver must primarily focus on road conditions, traffic signals, pedestrians, and surrounding vehicles. Therefore, it is difficult for the driver to continuously monitor every student and detect every safety risk inside the bus while driving. Similarly, if an accident or emergency occurs, there can be a delay in identifying the incident and informing parents or emergency authorities. This delay can increase the risk to students and reduce the effectiveness of emergency response.

School bus accidents and emergency situations can occur due to different reasons such as vehicle collisions, sudden impacts, unstable road conditions, mechanical failures, poor weather conditions, fire, smoke, and flooding. Apart from external accident risks, unsafe student behaviour inside the bus can also lead to injuries. One common issue is that students may place their hands, heads, or other body parts outside the bus window while the vehicle is moving. This behaviour can be extremely dangerous because roadside objects, passing vehicles, trees, poles, walls, or sudden movements of the bus may cause serious injuries. Therefore, school bus safety should not only focus on accident detection but also on preventing unsafe behaviour before an injury occurs.

The SISURAKSHA project is proposed as a smart school bus safety system designed to improve the protection of school children during transportation. The system combines embedded hardware, sensor-based monitoring, mobile application support, backend services, database integration, and machine learning based detection techniques. The main purpose of the project is to identify dangerous situations early and generate timely alerts to the driver, parents, and relevant emergency contacts. By using a smart monitoring approach, the system reduces dependency on manual observation and supports faster response during critical situations.

This individual research component focuses on two main areas of the SISURAKSHA system: smart accident detection and machine learning based window safety monitoring. The smart accident detection component uses multiple sensors connected to an ESP32 microcontroller to detect accident-related and emergency conditions. These conditions include vehicle rollover, sudden speed reduction, abnormal vibration, shock or impact, high pressure, fire, smoke, gas, and water-related hazard exposure. The system uses sensors and modules such as MPU9250, 801S vibration sensor, piezoelectric sensor, force sensitive resistors, MQ2 gas sensor, fire sensor, raindrop sensor, GPS module, SIM800L GSM module, push button, and DFPlayer Mini.

A major problem in accident detection systems is false accident detection. For example, a bus may pass through a pothole, rough road surface, or experience sudden non-critical braking. These situations may create abnormal sensor readings similar to an actual accident. To reduce this issue, the proposed accident detection module includes a 60-second driver verification mechanism. When a possible accident or emergency is detected, the system gives the driver a 60-second period to cancel the alert using a push button if the event is a false detection. If the alert is not cancelled within the given time, the system confirms the event as a real accident or emergency and sends an emergency alert with the GPS location through the GSM module.

The second component of this research is machine learning based window safety monitoring. This module uses two cameras to monitor the window areas of the school bus. A YOLOv8-based machine learning model is trained to detect unsafe student behaviour such as placing hands, head, or other body parts outside the bus window.

Roboflow is used for dataset annotation and classification, while Google Colab is used for model training. When unsafe behaviour is detected, the system generates a real-time audio warning through the speaker, instructing students to keep their hands, head, or body parts inside the bus.

The overall SISURAKSHA system is supported by a mobile and backend architecture. The frontend is developed using React Native, while Node.js is used for backend services. Supabase and MongoDB are used as databases to store system data, sensor readings, alert records, and other relevant information. Arduino IDE is used to program the ESP32 and integrate the hardware components. Through this combination of technologies, the proposed system provides a smart, practical, and scalable approach for school bus safety monitoring.

This research is significant because it addresses two major safety concerns in school transportation: accident and emergency detection and unsafe student behaviour near bus windows. By combining sensor-based monitoring with machine learning based visual detection, the proposed system improves real-time safety awareness and emergency response. The system also contributes to reducing false alerts through driver verification and provides immediate warning feedback to students through audio alerts. Therefore, this research supports the development of a safer, smarter, and more responsive school bus transportation environment.

### 1.2 Background Literature

School bus safety has become an important area of concern due to the vulnerability of children during transportation. School children are less experienced in identifying dangerous situations and may not always follow safety instructions while travelling. In a moving bus, even a small unsafe action can lead to serious consequences. Therefore, modern transportation systems increasingly require smart monitoring solutions that can detect risks automatically and support timely decision-making [1].

Traditional school bus safety practices mainly depend on human observation. Drivers, conductors, or teachers are usually responsible for monitoring students and responding to emergencies. However, manual supervision is not sufficient in many situations. A driver cannot continuously observe both the road and the students inside the bus.

Similarly, a bus assistant may not be able to monitor all students at the same time, especially when the bus is crowded. These limitations show the need for automated safety systems that can continuously monitor vehicle conditions and student behaviour.

Accident detection systems have been widely studied in the field of intelligent transportation and vehicle safety. Many systems use sensors to identify abnormal vehicle behaviour such as sudden acceleration, sudden braking, collision impact, vibration, and rollover. Accelerometers and gyroscopes are commonly used to measure movement, orientation, and tilt of a vehicle. In this research, the MPU9250 sensor is used to monitor acceleration, angular movement, and vehicle orientation. These readings can help identify sudden movement changes, tilt conditions, and rollover-like situations [2].

Vibration and impact detection are also important in accident identification. A vehicle collision or heavy impact usually produces abnormal vibration and shock. Sensors such as vibration sensors and piezoelectric sensors can be used to detect these physical changes. In the proposed system, the 801S vibration sensor and piezoelectric sensor are used to support impact and shock detection. Force sensitive resistors are also used to identify high pressure or force conditions that may occur during a collision or physical impact. By combining data from multiple sensors, the system can improve accident detection reliability.

A common weakness of single-sensor accident detection systems is false detection. For example, a pothole, speed bump, rough road, or sudden brake may generate sensor readings similar to an accident. If the system immediately sends emergency messages based on a single abnormal reading, it may create unnecessary alerts. Therefore, multi-sensor confirmation and verification mechanisms are important for reducing false positives. The proposed system addresses this issue by using multiple sensor inputs and a 60-second driver cancellation period before confirming an accident.

Emergency communication is another important part of accident detection systems. When an accident occurs, informing responsible parties quickly can reduce response time and improve student safety. GPS modules are commonly used to obtain the

location of the vehicle, while GSM modules can be used to send emergency SMS alerts. In this research, the NEO-6M GPS module is used to obtain the bus location, and the SIM800L GSM module is used to send emergency alert messages. This allows the system to communicate accident information even without depending on a mobile application interface.

Apart from road accidents, school buses may also face emergency hazards such as fire, smoke, gas leakage, or water exposure. Fire and smoke inside a bus can spread quickly and create serious risks for children. Similarly, flood or water-related conditions may become dangerous if the bus enters a waterlogged area. For this reason, the proposed system includes MQ2 gas sensor, fire sensor, and raindrop sensor as supporting emergency detection components. These sensors help expand the safety monitoring capability beyond collision detection.

Window safety is another important issue in school bus transportation. Children may sometimes place their hands, heads, or upper body parts outside the window while the bus is moving. This behaviour may occur due to lack of awareness, playfulness, overcrowding, or insufficient supervision. It can cause severe injuries when the bus passes close to other vehicles, poles, trees, walls, or roadside structures. Therefore, detecting unsafe window behaviour in real time is important for preventing injuries.

Computer vision and machine learning techniques have become effective methods for detecting objects, human actions, and unsafe behaviours. Object detection models can identify specific objects or body parts in images or video streams. YOLO, which stands for You Only Look Once, is a popular object detection approach because it can detect objects quickly and accurately in real time [3]. In this research, YOLOv8 is used to identify unsafe window behaviour such as hands, head, or body parts extending outside the bus window.

The window safety module uses two cameras to monitor the window areas of the bus. Image data related to unsafe window behaviour is collected and annotated using Roboflow. Roboflow supports dataset labelling, classification, and preprocessing, which are important steps in preparing a dataset for object detection model training. Google Colab is used to train the YOLOv8 model because it provides cloud-based

computational resources, including GPU support, which helps improve the efficiency of model training.

Real-time warning generation is also important in student safety systems. Detecting unsafe behaviour alone is not enough unless the system provides immediate feedback. In the proposed window safety module, when the machine learning model detects a hand, head, or body part outside the window, the system plays an audio warning through the speaker. This audio warning helps students immediately correct their behaviour without waiting for manual instruction from the driver or supervisor.

The SISURAKSHA system integrates multiple technologies to provide a complete safety monitoring solution. React Native is used to develop the mobile application frontend, Node.js is used for backend API handling, and Supabase and MongoDB are used for database management. The hardware components are programmed using Arduino IDE, while the machine learning model is trained using YOLOv8, Roboflow, and Google Colab. This combination of embedded systems, IoT communication, mobile application development, and machine learning supports the creation of a smart and practical school bus safety system.

### 1.3 Research Gap

Although several vehicle safety, school bus tracking, and accident detection systems have been introduced, many existing solutions still have limitations when applied to school bus environments. Most available school bus systems mainly focus on location tracking, attendance monitoring, route monitoring, or parent notification. These features are useful for transportation management, but they do not provide a complete safety solution for detecting accident-related events, emergency hazards, and unsafe student behaviour inside the bus.

A major limitation in existing accident detection systems is that many of them are designed for general vehicles rather than school buses. General vehicle accident detection systems commonly focus on collision identification, sudden impact detection, or GPS-based location sharing after an accident. However, school buses require additional safety considerations because the passengers are children. Children may behave unpredictably, may not fully understand safety instructions, and may

require immediate warning or guidance during unsafe situations. Therefore, a school bus safety system should not only detect accidents but also monitor student-related safety risks.

Another important gap is the dependency on limited sensor inputs. Some accident detection systems rely mainly on accelerometer data, GPS speed changes, or vibration readings. However, using a single sensor or a limited number of sensor values can reduce the reliability of the system. For example, a pothole, rough road surface, speed bump, or sudden non-critical braking can produce abnormal sensor readings similar to an accident. This can lead to false accident detection. If an emergency message is sent immediately without verification, it may create unnecessary panic among parents and emergency contacts. Therefore, an effective accident detection system should include multi-sensor confirmation and a false alert reduction mechanism.

Existing systems also show limited support for driver-based accident verification. In real school bus conditions, false alerts are possible due to normal road conditions or non-dangerous vehicle movements. However, most systems do not provide a practical method for the driver to cancel a false accident alert before it is sent. This research addresses this gap by introducing a 60-second driver verification mechanism. When a possible accident or emergency is detected, the system gives the driver 60 seconds to cancel the alert using a push button. If the driver does not cancel the alert within the given time, the system confirms the incident and sends an emergency message with the bus location.

Furthermore, many accident detection systems mainly focus on collision or impact detection and do not consider other emergency hazards such as fire, smoke, gas leakage, or water-related danger. In a school bus environment, these hazards can also create serious risks for students. For example, a fire or smoke situation inside a bus requires immediate detection and response. Similarly, water or flood-related conditions can become dangerous depending on the road and weather situation. Therefore, school bus safety monitoring should include both accident detection and emergency hazard detection.

Another major research gap is the lack of automated window safety monitoring in school buses. In real-world school transportation, students may place their hands, heads, or other body parts outside the bus window while the vehicle is moving. This behaviour can cause severe injuries due to passing vehicles, roadside objects, trees, poles, walls, or sudden movements of the bus. Most existing school bus safety systems do not identify this type of unsafe behaviour automatically. Manual supervision is not always reliable because the driver must focus on the road, and a bus assistant may not be able to monitor all students continuously.

Although camera-based surveillance systems are available, many of them only record video and do not provide intelligent real-time detection of unsafe behaviour. A normal camera system can capture incidents, but it may not automatically detect whether a student is placing a hand, head, or body part outside the window. Therefore, there is a need for a machine learning based window safety monitoring system that can identify unsafe window behaviour in real time and generate immediate warnings.

There is also a gap in integrating sensor-based accident detection with machine learning based student behaviour monitoring within a single school bus safety solution. Many systems focus on either vehicle condition monitoring or passenger monitoring separately. However, school bus safety requires both. A complete system should detect accidents and emergencies while also identifying unsafe actions that may cause injuries. The proposed SISURAKSHA component addresses this gap by combining smart accident detection with YOLOv8-based window safety monitoring.

Therefore, the main research gap identified in this study is the lack of an integrated, real-time, school bus focused safety module that combines multi-sensor accident and emergency detection, false alert reduction, GPS/GSM-based alert communication, and machine learning based window safety monitoring. The proposed research attempts to fill this gap by developing a smart accident detection and window safety module for the SISURAKSHA school bus safety system.

**Table 1.1: Comparison of Existing Safety Systems and the Proposed SISURAKSHA Module**

| Feature / Safety Requirement | Existing School Bus Tracking Systems | Existing Vehicle Accident Detection Systems | Proposed SISURAKSHA Module |
|---|---|---|---|
| School bus focused design | Partially supported | Not specifically focused on school buses | Specifically designed for school bus safety |
| GPS location tracking | Commonly available | Commonly available | Supported using NEO-6M GPS module |
| Parent notification | Commonly available | Limited or not available | Supported through emergency alert workflow |
| Accident detection | Limited or not available | Available in some systems | Supported using multi-sensor detection |
| Rollover / tilt detection | Rarely available | Sometimes available | Supported using MPU9250 |
| Abnormal vibration detection | Rarely available | Supported in some systems | Supported using 801S vibration sensor |
| Shock / impact detection | Limited | Supported in some systems | Supported using piezoelectric sensor and vibration sensor |
| High pressure / force detection | Not commonly available | Rarely available | Supported using force sensitive resistors |
| Fire detection | Rarely available | Rarely available | Supported using fire sensor |
| Smoke / gas detection | Rarely available | Rarely available | Supported using MQ2 gas sensor |
| Water / flood hazard detection | Not commonly available | Not commonly available | Supported using raindrop/water sensor |
| GSM-based emergency SMS alert | Sometimes available | Commonly available | Supported using SIM800L GSM module |
| False accident alert cancellation | Rarely available | Rarely available | Supported using 60-second driver verification |
| Window safety monitoring | Not available in most systems | Not applicable | Supported using two cameras and YOLOv8 |
| Machine learning based unsafe behaviour detection | Not commonly available | Not applicable | Supported using YOLOv8 model |
| Real-time audio warning | Limited | Limited | Supported through speaker / DFPlayer Mini |
| Integrated accident and student behaviour monitoring | Not commonly available | Not available | Fully supported in the proposed module |

As shown in Table 1.1, existing school bus tracking systems mainly focus on location monitoring and parent notification, while vehicle accident detection systems mainly focus on collision or impact-related events. However, both types of systems have limitations in detecting school bus specific risks such as unsafe student behaviour near windows, fire or smoke hazards, water-related emergency conditions, and false accident alert situations. The proposed SISURAKSHA module provides a more complete safety solution by integrating multi-sensor accident detection, emergency hazard detection, GPS/GSM communication, driver-based false alert cancellation, and machine learning based window safety monitoring.

### 1.4 Research Problem

School bus transportation is widely used by students as a daily mode of travel, but ensuring student safety inside and around the bus remains a major challenge. School children are more vulnerable than adults because they may not fully understand dangerous situations or follow safety instructions consistently. During travel, students can be exposed to accident-related risks such as sudden braking, collision, abnormal vibration, rollover, shock impact, fire, smoke, gas leakage, and water-related hazards. In addition, unsafe student behaviour such as placing hands, head, or other body parts outside the bus window can result in serious injuries.

In many school buses, safety monitoring still depends mainly on manual supervision by the driver, conductor, teacher, or bus assistant. However, this approach is not fully reliable. The driver must focus on road conditions, traffic, pedestrians, and vehicle control, which makes it difficult to continuously observe the students inside the bus. Even when a bus assistant is available, monitoring all students and all window areas at the same time can be difficult, especially when the bus is crowded. Therefore, dangerous situations may not be identified immediately.

Another major problem is the delay in accident detection and emergency communication. If a school bus is involved in an accident or emergency situation, parents and responsible authorities need to receive accurate information as quickly as possible. However, without an automated detection and alert system, incident

reporting may depend on the driver, passengers, or nearby people. This can delay emergency response and may increase the risk to students. Therefore, there is a need for an automated system that can detect accident-related events and send emergency alerts with the bus location.

False accident detection is also a significant problem in sensor-based accident detection systems. A school bus may pass through a pothole, speed bump, rough road surface, or experience sudden non-critical braking. These events may create abnormal sensor readings similar to an actual accident. If the system immediately sends emergency alerts for every abnormal sensor value, it can cause unnecessary panic among parents and emergency contacts. It can also reduce trust in the system. Therefore, the accident detection mechanism must include a method to reduce false alerts while still maintaining quick emergency response.

Furthermore, most safety systems do not adequately address window-related student behaviour. Students may place their hands, head, or body parts outside the bus window while the vehicle is moving. This is a dangerous action because passing vehicles, roadside objects, trees, walls, poles, and sudden vehicle movements can cause severe injuries. Manual monitoring of this behaviour is difficult because the driver cannot continuously observe every window while driving. A normal camera system may record the situation, but it does not automatically detect unsafe behaviour or warn students immediately.

Therefore, the research problem addressed in this study is the lack of an integrated, real-time school bus safety module that can detect accident and emergency conditions, reduce false accident alerts, communicate emergency information with location, and automatically identify unsafe student behaviour near bus windows.

This research focuses on solving the following main problem: How can a smart school bus safety module be developed to detect accident and emergency conditions using multiple sensors, reduce false accident alerts through driver verification, send emergency alerts with GPS location, and detect unsafe window behaviour using machine learning based camera monitoring?

To address this problem, the proposed SISURAKSHA module combines multi-sensor accident and emergency detection with YOLOv8-based window safety monitoring. The accident detection component uses sensors such as MPU9250, 801S vibration sensor, piezoelectric sensor, force sensitive resistors, MQ2 gas sensor, fire sensor, raindrop sensor, GPS module, and SIM800L GSM module. A 60-second driver verification mechanism is included to allow false alerts to be cancelled before emergency messages are sent. The window safety component uses two cameras and a trained YOLOv8 model to detect unsafe behaviour such as hands, head, or body parts outside the window. When such behaviour is detected, the system generates real-time audio warnings to improve student safety.

## 1.5 Research Objectives

### 1.5.1 Main Objectives

The main objective of this study is to develop a smart accident detection and machine learning based window safety module for the SISURAKSHA school bus safety system. This module is designed to improve the safety of school children during transportation by detecting accident-related events, emergency hazard conditions, and unsafe student behaviour near bus windows. The proposed system combines sensor-based accident detection, GPS/GSM-based emergency alert communication, driver-based false alert cancellation, and real-time machine learning based window safety monitoring.

The accident detection component uses an ESP32 microcontroller with multiple sensors to identify abnormal vehicle conditions such as rollover, sudden speed reduction, abnormal vibration, shock impact, high pressure, fire, smoke, gas leakage, and water-related hazards. When a possible accident or emergency condition is detected, the system activates a 60-second driver verification period. During this period, the driver can cancel the alert using a push button if the detected event is a false accident. If the alert is not cancelled within the given time, the system confirms the event as an accident or emergency and sends an alert message with the GPS location through the GSM module.

The window safety component uses two cameras and a YOLOv8-based machine learning model to detect unsafe student behaviour near bus windows. The system identifies situations where students place their hands, heads, or other body parts outside the bus window while the vehicle is moving. When such unsafe behaviour is detected, the system generates a real-time audio warning through the speaker to instruct students to keep their body parts inside the bus. Therefore, the overall objective is to provide a reliable, real-time, and intelligent safety monitoring solution for school buses by integrating embedded systems, IoT communication, and machine learning technologies.

### 1.5.2 Specific Objectives

**Identify and Analyse School Bus Accident and Emergency Safety Requirements**

To identify and analyse the major accident-related and emergency safety risks associated with school bus transportation. This includes studying conditions such as vehicle rollover, sudden speed reduction, abnormal vibration, collision impact, high pressure, fire, smoke, gas leakage, and water-related hazards. The analysis will guide the selection of suitable sensors and system logic required to detect possible accident and emergency situations in real time.

**Develop a Multi-Sensor Based Accident Detection System**

To design and develop a multi-sensor accident detection mechanism using an ESP32 microcontroller and sensors such as MPU9250, 801S vibration sensor, piezoelectric sensor, force sensitive resistors, MQ2 gas sensor, fire sensor, and raindrop sensor. The system will process sensor readings to identify abnormal conditions related to accidents and emergency hazards. By combining multiple sensor inputs, the system aims to improve detection reliability and reduce dependency on a single sensor.

**Implement a False Accident Alert Reduction Mechanism**

To implement a 60-second driver verification mechanism to reduce false accident alerts caused by potholes, rough roads, speed bumps, or non-critical sudden braking. When a possible accident is detected, the system will provide the driver with a limited

time period to cancel the alert using a push button. If the alert is cancelled, the system will return to normal monitoring mode. If it is not cancelled within 60 seconds, the system will confirm the event as a real accident or emergency.

**Develop GPS and GSM Based Emergency Alert Communication**

To integrate GPS and GSM communication for emergency alert generation. The NEO-6M GPS module will be used to retrieve the bus location, while the SIM800L GSM module will be used to send emergency SMS alerts to relevant emergency contacts. This objective supports faster emergency response by providing location-based accident information when a confirmed accident or emergency event occurs.

**Develop a Machine Learning Based Window Safety Detection System**

To develop a camera-based window safety monitoring system using a YOLOv8 machine learning model. Two cameras will be used to monitor the window areas of the school bus and detect unsafe student behaviour such as placing hands, head, or other body parts outside the bus window. Roboflow will be used for dataset annotation and classification, while Google Colab will be used for training the YOLOv8 model.

**Implement Real-Time Audio Warning for Unsafe Window Behaviour**

To provide real-time audio warnings when unsafe student behaviour near bus windows is detected. When the machine learning model identifies a window safety violation, the system will activate the speaker or DFPlayer Mini to play a warning message. This warning will instruct students to keep their hands, head, or body parts inside the bus, helping to prevent injuries before an accident occurs.

**Integrate the Module with the SISURAKSHA System Architecture**

To integrate the accident detection and window safety module with the overall SISURAKSHA system architecture. The frontend will be developed using React Native, the backend will be implemented using Node.js, and Supabase and MongoDB will be used for database management. Arduino IDE will be used for hardware programming and sensor integration. This integration will ensure that the individual module works as part of a complete school bus safety monitoring system.

**Evaluate the Performance of the Proposed Safety Module**

To test and evaluate the performance of the proposed system under prototype-level conditions. The evaluation will consider accident detection accuracy, false alert handling, GPS/GSM alert functionality, emergency hazard detection, YOLOv8-based window safety detection, and real-time audio warning performance. The results will be used to identify the effectiveness, limitations, and future improvement areas of the proposed SISURAKSHA module.

# 2 METHODOLOGY

## 2.1 Methodology

This chapter describes the methodology followed to design, develop, and evaluate the smart accident detection and machine learning based window safety module of the SISURAKSHA school bus safety system. The proposed module was developed to improve student safety during school transportation by identifying accident-related events, emergency hazard conditions, and unsafe student behaviour near bus windows.

The methodology consists of two major parts. The first part focuses on sensor-based accident and emergency detection using an ESP32 microcontroller. This component detects conditions such as rollover, sudden speed reduction, abnormal vibration, shock impact, high pressure, fire, smoke, gas, and water-related hazards. The second part focuses on machine learning based window safety monitoring using two cameras and a YOLOv8 object detection model. This component detects unsafe behaviour such as students placing their hands, heads, or other body parts outside the bus window.

The overall methodology includes requirement gathering, feasibility analysis, problem identification, system design, hardware and software selection, sensor integration, machine learning model development, alert generation, false alert cancellation, and testing. The proposed system was implemented using React Native for the frontend, Node.js for the backend, Supabase and MongoDB for database management, Arduino IDE for ESP32 programming, Roboflow for dataset annotation, Google Colab for model training, and YOLOv8 for machine learning based object detection.

### 2.1.1 Requirements Gathering and Analysing

The requirement gathering process was carried out by analysing the safety problems related to school bus transportation. The main objective of this stage was to identify the risks faced by school children during bus travel and define the functional and non-functional requirements needed to develop a smart safety monitoring module.

Several major safety issues were identified during the requirement analysis. These included accident detection delay, lack of real-time emergency communication, false accident alerts caused by potholes or rough road conditions, fire or smoke hazards,

water-related emergency situations, and unsafe student behaviour near bus windows. Since the driver cannot continuously monitor every student while driving, an automated system was required to assist in identifying dangerous conditions.

The proposed system requirements were divided into accident detection requirements, emergency alert requirements, window safety requirements, and system integration requirements. This categorisation allowed each requirement to be mapped to relevant sensors, software components, machine learning models, and alert mechanisms.

**Table 2.1: Functional Requirements of the Proposed Module**

| Requirement ID | Description |
|---|---|
| FR1 | Detect accident-related events such as rollover, sudden movement, abnormal vibration, shock impact, and high pressure. |
| FR2 | Detect emergency hazards such as fire, smoke, gas, and water exposure. |
| FR3 | Use multiple sensor inputs instead of depending on a single sensor. |
| FR4 | Provide a 60-second driver cancellation mechanism for possible false alerts. |
| FR5 | Retrieve GPS location when a confirmed accident or emergency occurs. |
| FR6 | Send emergency SMS alerts using GSM communication. |
| FR7 | Monitor bus window areas using two cameras. |
| FR8 | Detect unsafe window behaviour using a YOLOv8 machine learning model. |
| FR9 | Provide real-time audio warnings for unsafe window behaviour. |
| FR10 | Integrate the module with the SISURAKSHA frontend, backend, and database architecture. |

### 2.1.2 Feasibility Study

A feasibility study was conducted to identify whether the proposed module could be developed using available hardware, software, and project resources. The feasibility

study focused on technical, economic, operational, and scheduling feasibility. This helped determine whether the solution could be implemented practically within the limitations of an undergraduate research project.

The proposed system is technically feasible because the required technologies and hardware components are available and suitable for prototype-level development. The ESP32 microcontroller supports sensor integration and wireless communication. Sensors such as MPU9250, 801S vibration sensor, piezoelectric sensor, FSR pressure sensors, MQ2 gas sensor, fire sensor, and raindrop sensor can be connected to the ESP32 for accident and emergency detection.

The machine learning based window safety module is also technically feasible because YOLOv8 supports real-time object detection and can be trained to detect specific unsafe behaviours near bus windows. Roboflow can be used for image dataset annotation and preprocessing, while Google Colab provides cloud-based GPU support for model training. React Native, Node.js, Supabase, and MongoDB provide a suitable software architecture for the overall SISURAKSHA system.

The proposed module is economically feasible because the selected hardware components are low-cost and commonly available. ESP32, sensors, GPS, GSM, and audio modules are suitable for prototype development without high development cost. The use of open-source and free development tools such as Arduino IDE, Google Colab, Roboflow free resources, and YOLOv8 reduces the software development cost.

The proposed system is operationally feasible because it is designed to support drivers, parents, and school transportation authorities. The system does not require the driver to continuously monitor sensor values or camera feeds manually. Instead, the system automatically detects possible accidents, emergency hazards, and unsafe window behaviour. The 60-second cancellation mechanism allows the driver to cancel false accident alerts, making the system more practical in real-world road conditions.

**Table 2.2: Feasibility Summary of the Proposed Module**

| Feasibility Area | Justification |
|---|---|

| | |
|---|---|
| Technical Feasibility | Supported by ESP32, sensor modules, YOLOv8, Roboflow, Google Colab, React Native, Node.js, Supabase and MongoDB. |
| Economic Feasibility | Uses low-cost hardware components and free/open-source development platforms where possible. |
| Operational Feasibility | Supports practical school bus operation by automating detection and allowing driver-based false alert cancellation. |
| Scheduling Feasibility | Can be developed through modular tasks such as hardware testing, ML training, alert implementation and system integration. |

### 2.1.3 Problem Statement

School bus transportation is widely used by students as a daily mode of travel, but ensuring student safety inside and around the bus remains a major challenge. School children are more vulnerable than adults because they may not fully understand dangerous situations or follow safety instructions consistently. During travel, students can be exposed to accident-related risks such as sudden braking, collision, abnormal vibration, rollover, shock impact, fire, smoke, gas leakage, and water-related hazards. In addition, unsafe student behaviour such as placing hands, head, or other body parts outside the bus window can result in serious injuries.

In many school buses, safety monitoring still depends mainly on manual supervision by the driver, conductor, teacher, or bus assistant. However, this approach is not fully reliable. The driver must focus on road conditions, traffic, pedestrians, and vehicle control, which makes it difficult to continuously observe the students inside the bus. Even when a bus assistant is available, monitoring all students and all window areas at the same time can be difficult, especially when the bus is crowded. Therefore, dangerous situations may not be identified immediately.

Another major problem is the delay in accident detection and emergency communication. If a school bus is involved in an accident or emergency situation, parents and responsible authorities need to receive accurate information as quickly as possible. However, without an automated detection and alert system, incident

reporting may depend on the driver, passengers, or nearby people. This can delay emergency response and may increase the risk to students. Therefore, there is a need for an automated system that can detect accident-related events and send emergency alerts with the bus location.

False accident detection is also a significant problem in sensor-based accident detection systems. A school bus may pass through a pothole, speed bump, rough road surface, or experience sudden non-critical braking. These events may create abnormal sensor readings similar to an actual accident. If the system immediately sends emergency alerts for every abnormal sensor value, it can cause unnecessary panic among parents and emergency contacts. It can also reduce trust in the system. Therefore, the accident detection mechanism must include a method to reduce false alerts while still maintaining quick emergency response.

Therefore, this research proposes a smart accident detection and machine learning based window safety module for the SISURAKSHA school bus safety system. The system uses multiple sensors to detect accident and emergency conditions and uses a YOLOv8-based machine learning model to detect unsafe student behaviour near windows. A 60-second driver verification mechanism is included to reduce false accident alerts before sending emergency notifications.

### 2.1.4 System Designs

The proposed SISURAKSHA module consists of two main sub-components: the smart accident and emergency detection component and the machine learning based window safety component. The accident detection component uses an ESP32 microcontroller to collect and process sensor data. The window safety component uses two cameras and a trained YOLOv8 model to identify unsafe student behaviour near bus windows.

The overall system follows a modular architecture. Sensor data, machine learning outputs, alert logic, backend communication, and database storage are handled as separate but connected parts of the system. This modular approach improves maintainability and allows each component to be tested individually.

#### 2.1.4.1 Overall System Diagram

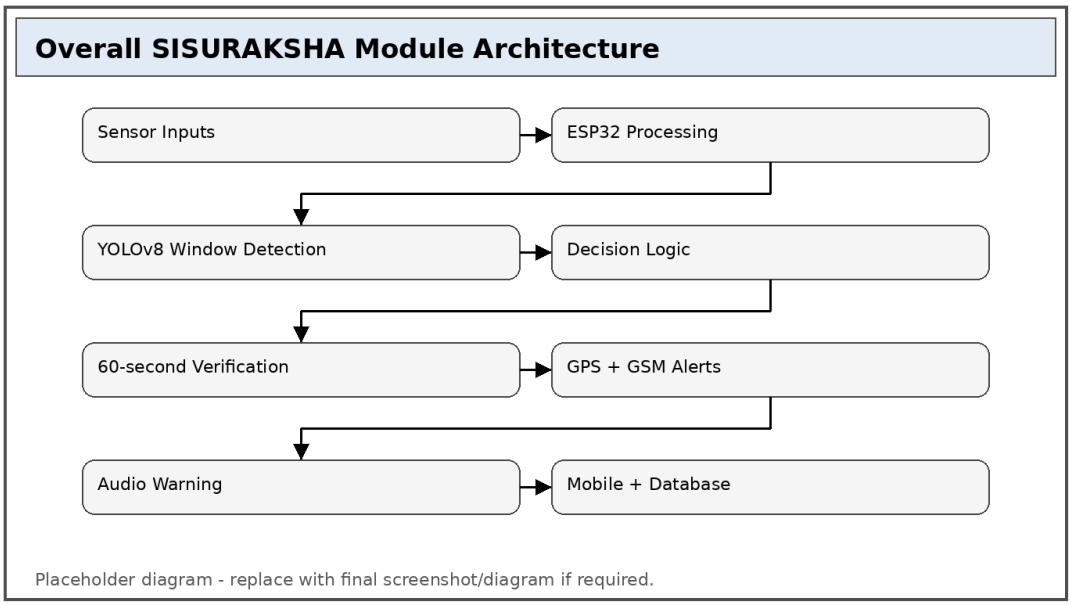



**Figure 2.1: Overall System Diagram of the SISURAKSHA Smart Accident Detection and Window Safety Module**

Figure 2.1 shows the overall architecture of the proposed SISURAKSHA module. The accident detection sensor module collects physical and environmental data using sensors connected to the ESP32. The window safety module uses two cameras and a YOLOv8 model to detect unsafe student behaviour. The alert decision unit processes both sensor-based and camera-based outputs. If an accident is detected, the system activates a 60-second driver verification period. If the alert is not cancelled, the system sends an emergency SMS with GPS location. If unsafe window behaviour is detected, the system provides an audio warning through the speaker.

### 2.1.4.2 Accident Detection Module Design

The accident detection module is designed to identify abnormal vehicle conditions and emergency hazards using multiple sensors. The ESP32 microcontroller acts as the main control unit. The MPU9250 sensor detects acceleration, tilt, and orientation changes. The 801S vibration sensor and piezoelectric sensor detect abnormal vibration and shock impact. Force sensitive resistors detect high pressure or force. MQ2 gas sensor, fire sensor, and raindrop sensor detect smoke, gas, fire, and water-related hazards.

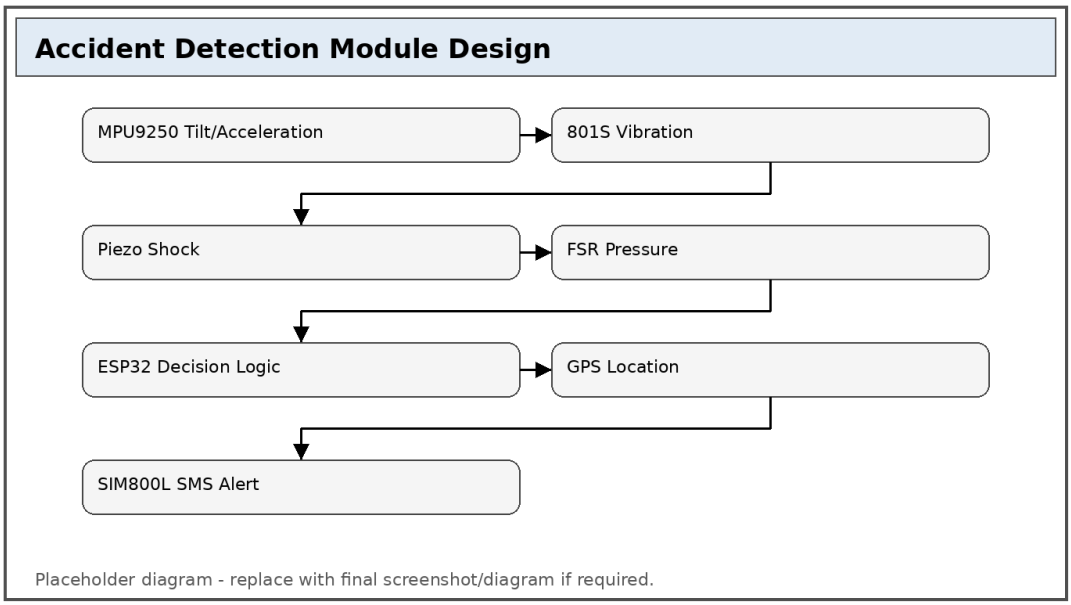



**Figure 2.2: Accident Detection Module Design**

The accident detection module does not depend on a single sensor. Instead, it uses a multi-sensor verification approach to improve reliability. For example, a rollover condition can be identified using MPU9250 tilt readings, while a collision-like condition can be confirmed using vibration or shock sensor readings. This approach helps reduce false accident detection caused by normal road disturbances.

### 2.1.4.3 Emergency Hazard Detection Module Design

The emergency hazard detection module identifies fire, smoke, gas, and water-related hazardous conditions. These hazards may not always be caused by a road accident, but they can create serious risk for students inside the bus. The fire sensor detects flame-related risk, while the MQ2 sensor detects smoke or gas. The raindrop sensor is used to identify water exposure or possible flood-related conditions.

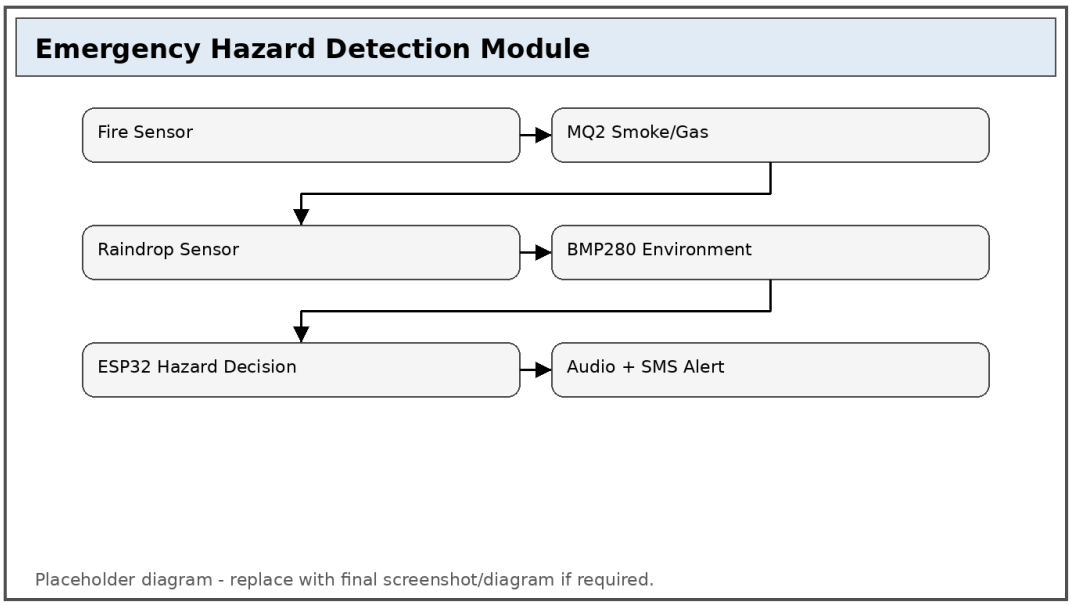



**Figure 2.3: Emergency Hazard Detection Module Design**

These hazard sensors are connected to the ESP32 and checked continuously. If a hazardous condition is detected, the system can generate an alert even if the bus is not involved in a collision. This design improves the overall safety coverage of the system by considering emergency scenarios beyond conventional accident detection.

### 2.1.4.4 Window Safety Module Design

The window safety module is developed to detect unsafe student behaviour near bus windows. Two cameras are used to monitor the window areas of the school bus. A YOLOv8 object detection model is trained to identify unsafe actions such as hands, head, or body parts extending outside the bus window. The detected output can be used to activate an audio warning and notify the driver dashboard.

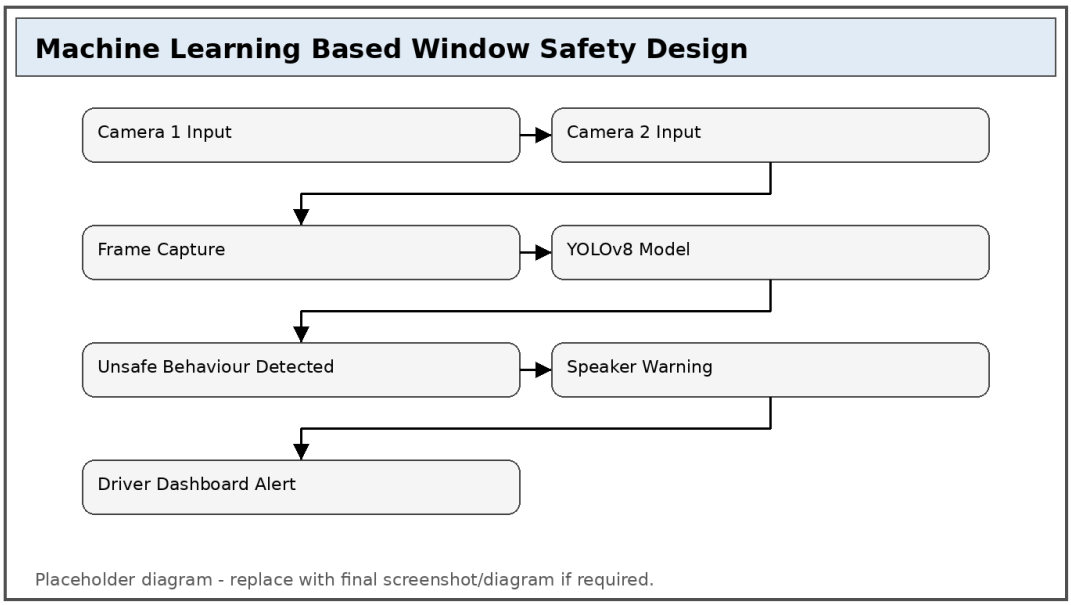



**Figure 2.4: Window Safety Machine Learning Module Design**

When the YOLOv8 model detects a hand, head, or body part outside the window, the system triggers an audio warning. This warning instructs the student to keep their body parts inside the bus. The purpose of this module is to prevent injuries before they occur by correcting unsafe behaviour in real time.

### 2.1.4.5 Accident Alert Workflow

The accident alert workflow includes possible accident detection, driver verification, false alert cancellation, and emergency alert sending. This workflow is important because false accident detection can occur due to potholes, rough roads, speed bumps, or sudden non-critical braking.

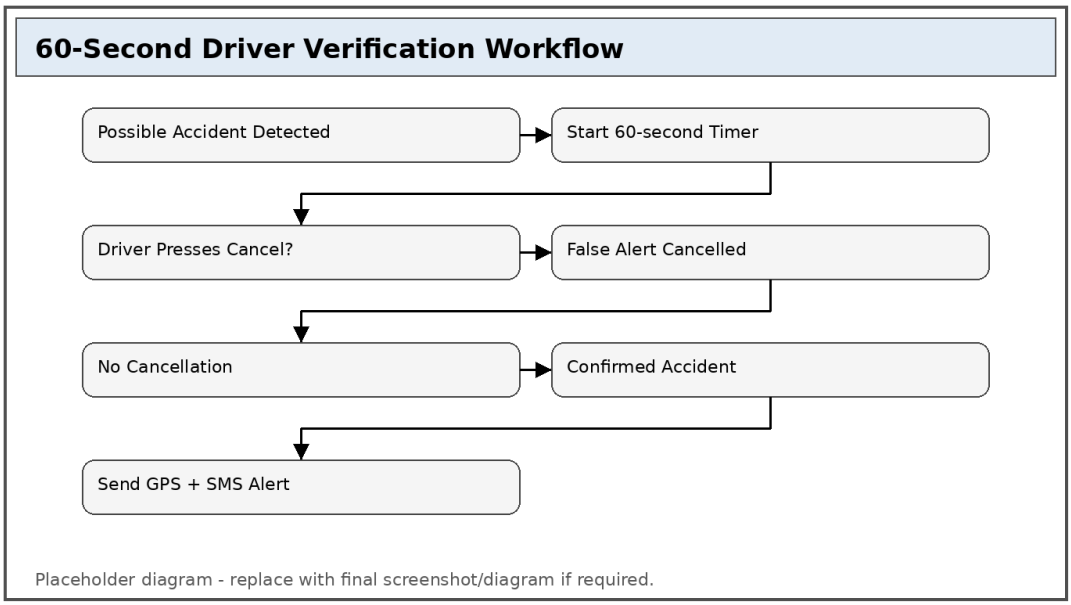



**Figure 2.5: Accident Detection and 60-Second Driver Verification Workflow**

This workflow reduces unnecessary emergency messages. When a possible accident is detected, the system gives the driver 60 seconds to cancel the alert. If the driver cancels the alert, the system records the event as a false detection and returns to monitoring mode. If the driver does not cancel within 60 seconds, the system confirms the accident or emergency and sends an alert message with GPS location.

### 2.1.5 Sensor Selection and Data Acquisition

The sensor selection process was based on the accident and emergency conditions that needed to be detected. Each sensor was selected for a specific purpose. The ESP32 was selected as the main controller because it supports multiple sensor connections and is suitable for IoT-based embedded systems. Sensor data is collected continuously by the ESP32 and compared with threshold values or decision rules.

**Table 2.3: Sensor and Module Selection for Accident and Emergency Detection**

| **Component / Sensor** | **Purpose in the System** |
|---|---|
| ESP32 | Main microcontroller for sensor processing and control |
| MPU9250 | Detects acceleration, tilt, orientation and rollover conditions |
| 801S Vibration Sensor | Detects abnormal vibration and impact |

| | |
|---|---|
| Piezoelectric Sensor | Detects shock or collision-like impact |
| Force Sensitive Resistor | Detects high pressure or force conditions |
| NEO-6M GPS Module | Retrieves bus location during confirmed accident |
| SIM800L GSM Module | Sends emergency SMS alerts |
| Push Button | Allows driver to cancel false accident alerts |
| DFPlayer Mini / Speaker | Provides audio warnings |
| MQ2 Gas Sensor | Detects smoke or gas-related hazards |
| Fire Sensor | Detects fire or flame-related hazards |
| Raindrop Sensor | Detects water or flood-related exposure |
| LM2596 Buck Converter | Provides stable voltage regulation for modules |
| BMP280 | Supports environmental monitoring |

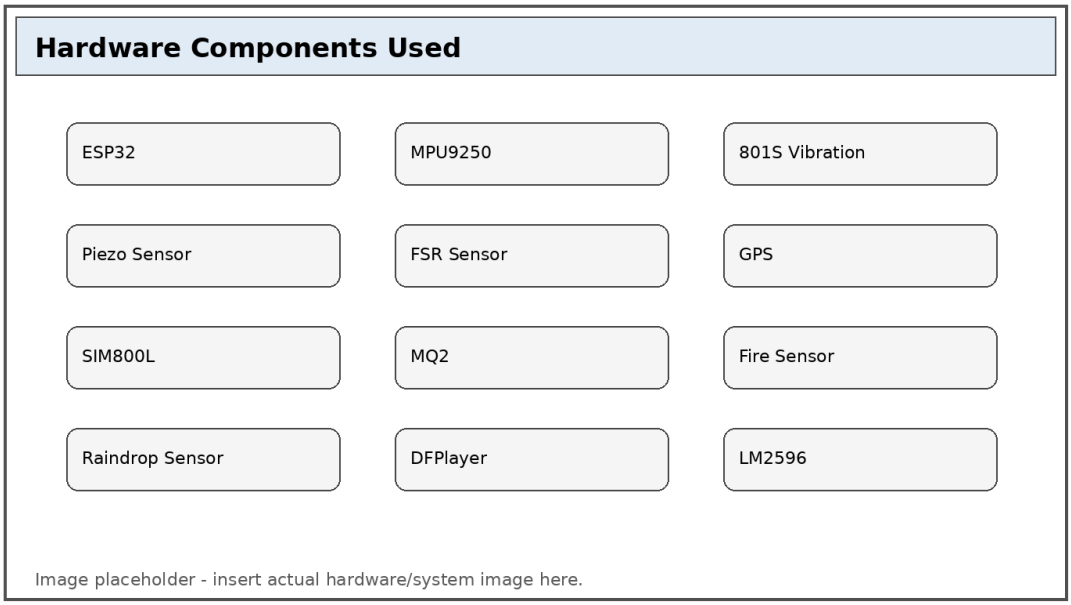



**Figure 2.7: Hardware Components Used in the Proposed Module**

Figure 2.7 is a placeholder for the hardware component collage. In the final submission, this figure should be replaced with clear images of the actual sensors and modules used in the prototype. The figure should include ESP32, MPU9250, 801S vibration sensor, piezoelectric sensor, FSR sensors, NEO-6M GPS, SIM800L GSM, MQ2 gas sensor, fire sensor, raindrop sensor, DFPlayer Mini, speaker and LM2596 buck converter.

### 2.1.6 Accident Detection Logic

The accident detection logic is developed using a multi-sensor decision-making approach. Instead of depending on a single sensor value, the system considers multiple sensor conditions to identify possible accident situations. This helps reduce false positives and improves the reliability of the detection process.

**Table 2.4: Accident and Emergency Conditions with Sensor Mapping**

| Detected Condition | Main Sensor / Module | Supporting Sensor / Module |
|---|---|---|
| Rollover / Tilt | MPU9250 | Vibration sensor |
| Sudden speed reduction | GPS / MPU9250 | Vibration sensor |
| Abnormal vibration | 801S vibration sensor | MPU9250 |
| Shock / Impact | Piezoelectric sensor | 801S vibration sensor |
| High pressure / force | Force Sensitive Resistor | Piezoelectric sensor |
| Fire | Fire sensor | MQ2 sensor |
| Smoke / Gas | MQ2 sensor | Fire sensor |
| Water / Flood exposure | Raindrop sensor | Manual verification |
| Confirmed accident alert | ESP32 decision logic | GPS + GSM |

The system classifies detected events into different levels. Minor abnormal events are treated as warnings. Strong sensor readings are treated as possible accidents. When the driver does not cancel the alert within 60 seconds, the event is treated as a confirmed accident or emergency. The system can be improved further by adding adaptive threshold learning using real road data collected during testing.

### 2.1.7 60-Second Driver Verification and False Alert Cancellation

False accident detection is a common issue in sensor-based accident detection systems. Road conditions such as potholes, bumps, rough roads, or sudden braking can produce abnormal sensor readings. To reduce false alerts, the proposed system includes a 60-second driver verification mechanism.

When the system detects a possible accident or emergency, it starts a countdown timer. During this 60-second period, the driver can press the push button to cancel the alert. If the button is pressed, the system identifies the event as a false detection and stops

the emergency alert process. If the button is not pressed within 60 seconds, the system confirms the event and sends an emergency SMS with the GPS location. This mechanism improves the practical reliability of the system by balancing fast emergency response with false alert reduction.

### 2.1.8 Machine Learning Based Window Safety Detection

The window safety detection system uses a camera-based machine learning approach. Two cameras are used to monitor the window areas of the school bus. The system is trained to detect unsafe student behaviours such as placing hands, head, or body parts outside the bus window. The YOLOv8 object detection model is used for this task because it is suitable for real-time object detection.

The dataset is prepared by collecting image samples of safe and unsafe window behaviour. Roboflow is used to annotate and classify the dataset. The annotated dataset is then exported and trained using Google Colab. The trained YOLOv8 model detects unsafe behaviour from camera input. If unsafe behaviour is detected, the system triggers an audio warning through the speaker.

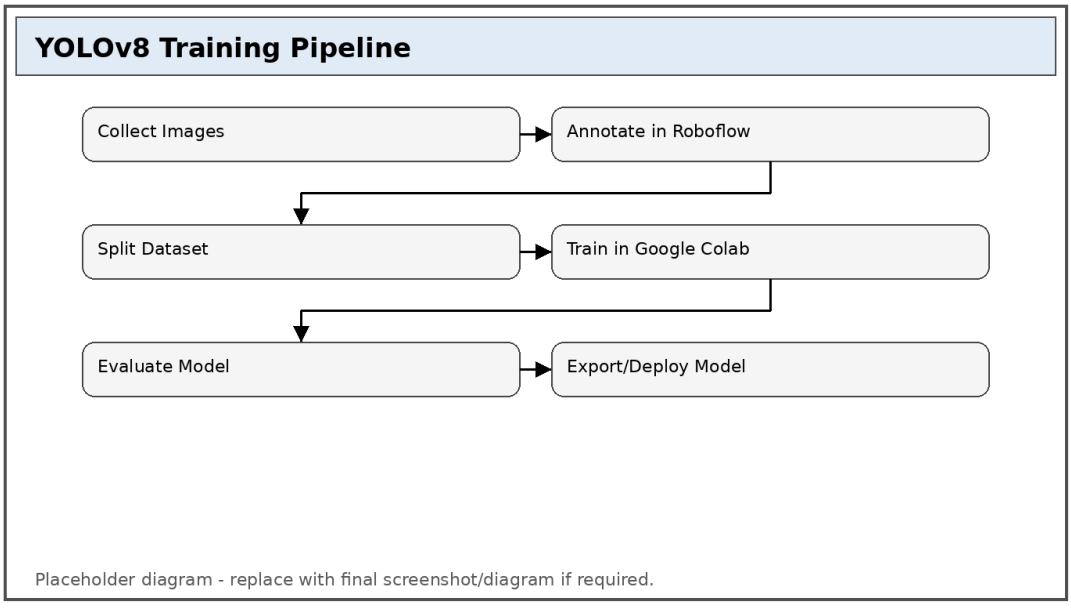



**Figure 2.6: YOLOv8 Window Safety Detection Training Pipeline**

Figure 2.6 illustrates the training pipeline followed for the machine learning based window safety module. The process starts with collecting image data, uploading images to Roboflow, annotating unsafe behaviours, splitting the dataset, training the

YOLOv8 model in Google Colab, evaluating the model and deploying it for prototype-level detection.

### 2.1.9 Choosing the Correct Technologies and Frameworks

Different technologies were selected to develop the SISURAKSHA system based on the requirements of each component. The system required mobile application development, backend communication, database storage, embedded programming, sensor integration, machine learning model training, and dataset annotation.

**Table 2.5: Technologies and Frameworks Used in the Proposed System**

| Area | Technology / Tool | Purpose |
|---|---|---|
| Frontend | React Native | Mobile application interface |
| Backend | Node.js | API handling and server-side communication |
| Database | Supabase | Structured data storage and real-time support |
| Database | MongoDB | Flexible storage for sensor logs and alert records |
| Hardware Programming | Arduino IDE | Programming ESP32 and sensors |
| Machine Learning Model | YOLOv8 | Detecting unsafe window behaviour |
| Dataset Annotation | Roboflow | Dataset labelling, classification and preprocessing |
| Model Training | Google Colab | Training YOLOv8 using cloud resources |
| Microcontroller | ESP32 | Main embedded control unit |
| Emergency Communication | SIM800L GSM | Sending SMS alerts |
| Location Tracking | NEO-6M GPS | Retrieving bus location |

React Native was selected for frontend development because it supports mobile application development and provides a user-friendly interface. Node.js was selected for backend development because it supports API-based communication and real-time system interaction. Supabase and MongoDB were selected to store user data, alert

records, sensor logs, and system information. Arduino IDE was selected for programming the ESP32 and integrating hardware sensors. YOLOv8 was selected for machine learning based window safety detection because it supports real-time object detection. Roboflow was used to prepare and annotate the dataset, while Google Colab was used to train the model using cloud-based GPU resources.

### 2.2 Commercialization Aspects of the Product

The SISURAKSHA school bus safety system has potential for commercialization because it addresses a real-world safety problem in student transportation. Schools, private school bus operators, parents, and transport service providers can benefit from a smart safety monitoring solution that improves accident detection, emergency communication, and student behaviour monitoring.

The proposed module can be commercialized as part of a complete school bus safety platform. The accident detection system can be installed in school buses as a hardware unit, while the mobile application can be used by parents, drivers, and school authorities to receive alerts and monitor safety-related information. The window safety module can be offered as an advanced feature for buses where student behaviour monitoring is required.

A basic version of the product can include GPS tracking, accident detection, and SMS emergency alerts. A premium version can include machine learning based window safety detection, real-time audio warnings, cloud-based data storage, alert history, analytics dashboards, and parent notification features. The system can also be expanded in the future with additional features such as live camera monitoring, school administration dashboards, predictive maintenance, and WhatsApp-based notifications through cloud API integration.

Since the system uses affordable sensors, ESP32, open-source tools, and scalable software technologies, it can be developed at a relatively low cost compared to advanced commercial vehicle monitoring systems. This makes SISURAKSHA suitable for local school transportation environments and future expansion into broader student transport safety solutions.

# 3 TESTING AND IMPLEMENTATION

## 3.1 Implementation

This chapter presents the implementation and testing of the smart accident detection and machine learning based window safety module. The implementation was carried out in stages to ensure that each hardware, software, and machine learning component could be tested individually before full integration. This approach reduced debugging complexity and improved the reliability of the complete prototype.

### 3.1.1 Hardware Implementation

The hardware implementation of the smart accident and emergency detection module was developed using an ESP32 microcontroller as the main processing unit. The ESP32 was selected because it supports multiple sensor connections, digital and analog input reading, serial communication, and IoT-based system integration. All accident detection, emergency hazard detection, false alert cancellation, and audio warning related components were connected to the ESP32 and programmed using Arduino IDE.

The hardware setup consists of several sensors and modules that are used to identify accident-related and emergency conditions in the school bus. The MPU9250 sensor is used to detect acceleration, angular movement, tilt, and rollover-like situations. The 801S vibration sensor and piezoelectric sensor are used to identify abnormal vibration, shock, and impact. Force Sensitive Resistor sensors are used to detect high pressure or force conditions. The MQ2 gas sensor, fire sensor, and raindrop sensor are used to detect smoke, gas, fire, and water-related hazards. The NEO-6M GPS module is used to obtain the bus location when a confirmed accident or emergency occurs, while the SIM800L GSM module is used to send emergency SMS alerts.

A push button was included in the hardware design to support the 60-second false alert cancellation mechanism. When the system detects a possible accident or emergency, the driver can press the push button within 60 seconds if the event is a false detection. If the button is not pressed within the given time, the system confirms the event and proceeds with the emergency alert process. This mechanism helps reduce unnecessary emergency messages caused by potholes, rough roads, or sudden non-critical braking.

The DFPlayer Mini module with a speaker is used to provide audio warnings. In the accident detection component, the speaker can notify the driver when a possible accident or emergency condition is detected. In the window safety component, the speaker is used to warn students when the machine learning model detects unsafe behaviour near the bus window, such as placing hands, head, or body parts outside the bus. The LM2596 Buck Converter module is used to regulate voltage and provide a stable power supply to the hardware components.

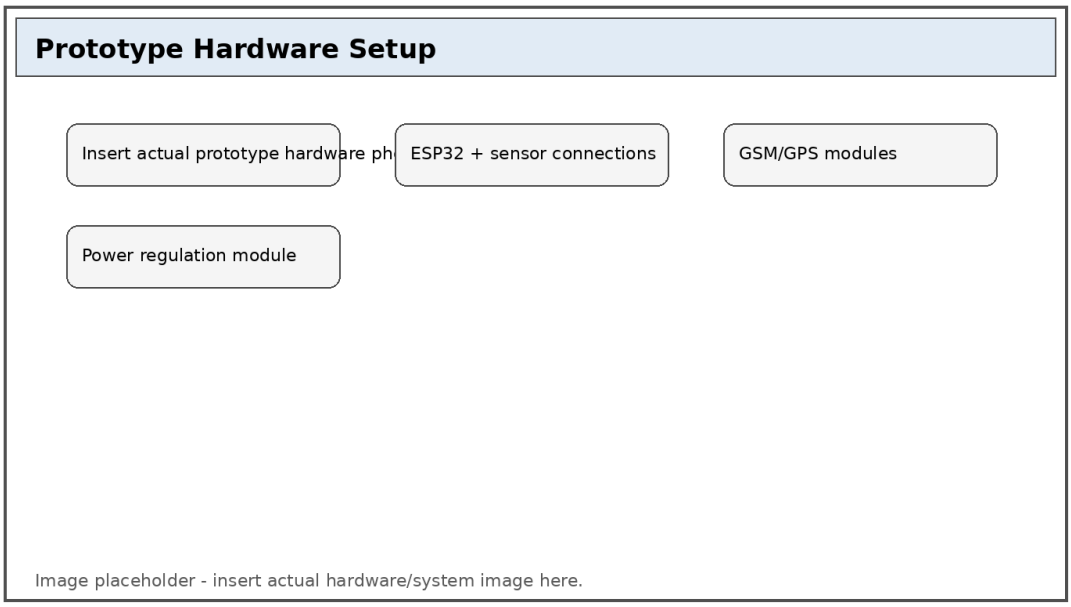



**Figure 3.1: Prototype Hardware Setup of the Accident Detection Module**

Figure 3.1 shows a placeholder for the prototype hardware setup of the accident detection module. In the final report, this placeholder should be replaced with an actual photograph of the connected hardware prototype. The image should clearly show the ESP32 microcontroller, sensor modules, GSM/GPS modules, power regulation circuit, push button, and audio output components.

**Table 3.1: Hardware Components Used in the Accident Detection Module**

| Component | Purpose |
|---|---|
| ESP32 | Main microcontroller for sensor processing and control |
| MPU9250 | Detects acceleration, tilt, orientation and rollover conditions |

| | |
|---|---|
| 801S Vibration Sensor | Detects abnormal vibration and impact |
| Piezoelectric Sensor | Detects shock or collision-like impact |
| Force Sensitive Resistor | Detects high pressure or force conditions |
| NEO-6M GPS Module | Retrieves bus location during confirmed accident |
| SIM800L GSM Module | Sends emergency SMS alerts |
| Push Button | Allows driver to cancel false accident alerts |
| DFPlayer Mini / Speaker | Provides audio warnings |
| MQ2 Gas Sensor | Detects smoke or gas-related hazards |
| Fire Sensor | Detects fire or flame-related hazards |
| Raindrop Sensor | Detects water or flood-related exposure |
| LM2596 Buck Converter | Provides stable voltage regulation for modules |
| BMP280 | Supports environmental monitoring |

### 3.1.2 ESP32 and Sensor Integration

The ESP32 was programmed using Arduino IDE. Each sensor was first connected and tested individually to confirm whether stable values could be read from the microcontroller. After individual validation, the sensors were integrated into the main accident detection program. The sensor readings were printed to the serial monitor during initial testing to observe value ranges under normal and abnormal conditions.

The MPU9250 sensor was used to obtain acceleration and gyroscope readings. These readings help identify sudden changes in movement and rollover-like tilt conditions. The 801S vibration sensor was used to detect strong vibrations. The piezoelectric sensor was used as an additional shock detection input. Force sensitive resistors were used to identify high pressure or physical force. The fire sensor, MQ2 sensor, and raindrop sensor were used as emergency hazard inputs.

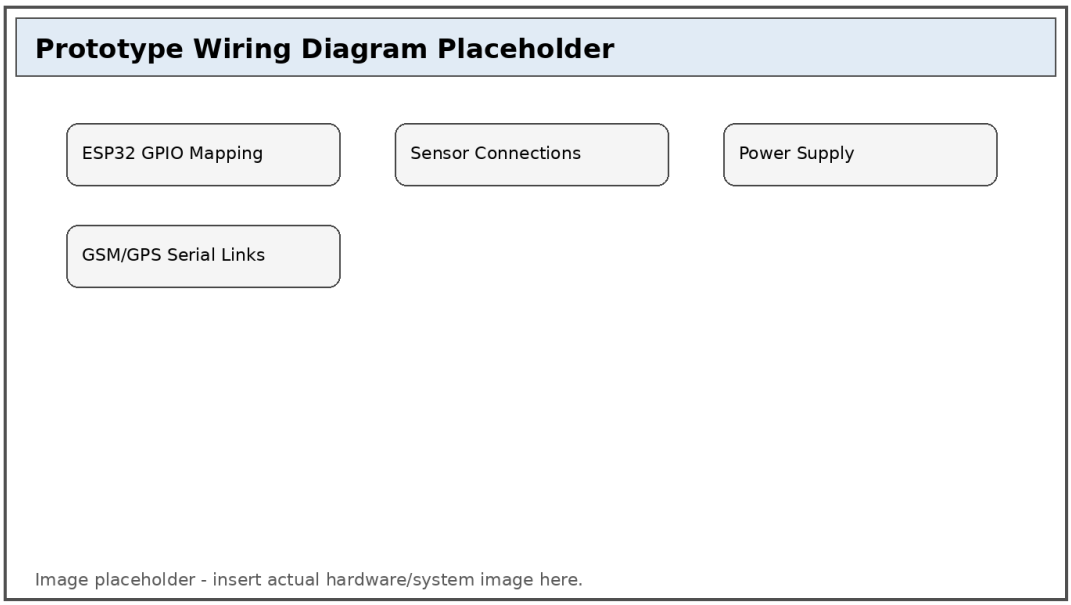



**Figure 3.2: Prototype Wiring Diagram of the Hardware Module**

**Table 3.2: GPIO and Connection Mapping Placeholder**

| Module | Connection Type | ESP32 Connection | Function |
|---|---|---|---|
| MPU9250 | I2C | SDA/SCL pins | Acceleration, gyroscope and tilt readings |
| 801S Vibration Sensor | Digital/Analog | ESP32 input pin | Vibration detection |
| Piezoelectric Sensor | Analog | ESP32 analog pin | Shock detection |
| FSR Sensors | Analog | ESP32 analog pins | Pressure/force detection |
| SIM800L GSM | Serial UART | TX/RX pins | SMS communication |
| NEO-6M GPS | Serial UART | TX/RX pins | Location data |
| Push Button | Digital | ESP32 input pin | False alert cancellation |
| DFPlayer Mini | Serial/Digital | ESP32 output/serial pins | Audio warning playback |
| Fire Sensor | Digital/Analog | ESP32 input pin | Fire detection |
| MQ2 Gas Sensor | Analog/Digital | ESP32 input pin | Smoke/gas detection |
| Raindrop Sensor | Analog/Digital | ESP32 input pin | Water exposure detection |

### 3.1.3 Accident and Emergency Detection Implementation

The accident detection implementation follows a threshold-based decision logic. Each sensor value is compared with a predefined threshold. If the value exceeds the threshold, the system marks the event as a possible abnormal condition. Since one

sensor reading alone may not be sufficient to confirm an accident, the system checks whether more than one related sensor is triggered. For example, abnormal MPU9250 readings combined with high vibration readings indicate a stronger accident possibility than vibration alone.

Emergency hazard detection was implemented using the MQ2 gas sensor, fire sensor, and raindrop sensor. If fire or smoke is detected, the system treats the event as a critical emergency condition. Water exposure is treated as a warning or hazard condition depending on the intensity and system logic. These hazard detections are important because accidents are not the only risk inside school buses.

The 60-second verification timer was implemented after the system detected a possible accident or emergency. During this time, the push button input is monitored. If the driver presses the button, the alert is cancelled. If not, the system confirms the accident and sends emergency information. This implementation improves reliability because it prevents immediate false alerts from being sent for potholes and rough roads.

### 3.1.4 GPS and GSM Alert Implementation

The GPS and GSM modules were integrated to support location-based emergency communication. The NEO-6M GPS module retrieves latitude and longitude values. The SIM800L GSM module sends an SMS message to a predefined emergency contact. The emergency message includes the accident status and the current GPS location. This allows emergency contacts to identify the location of the school bus quickly.

During implementation, serial communication was used to read GPS data and communicate with the GSM module. The GSM module requires a stable power supply; therefore, the LM2596 buck converter was used to provide regulated power. SMS functionality was tested by sending sample messages before integrating the accident confirmation logic.

### 3.1.5 Machine Learning Based Window Safety Implementation

The window safety implementation used a YOLOv8 object detection model to identify unsafe student behaviour near bus windows. The dataset was prepared using images

representing different conditions such as safe sitting, hands outside the window, head outside the window, and body parts extending outside the window. These images were uploaded to Roboflow for annotation and dataset preparation.

Roboflow was used to label the relevant objects in each image. The annotation classes may include hand outside window, head outside window, and body part outside window. After annotation, the dataset was split into training, validation, and testing subsets. The dataset was then exported in a YOLOv8-compatible format and trained using Google Colab.

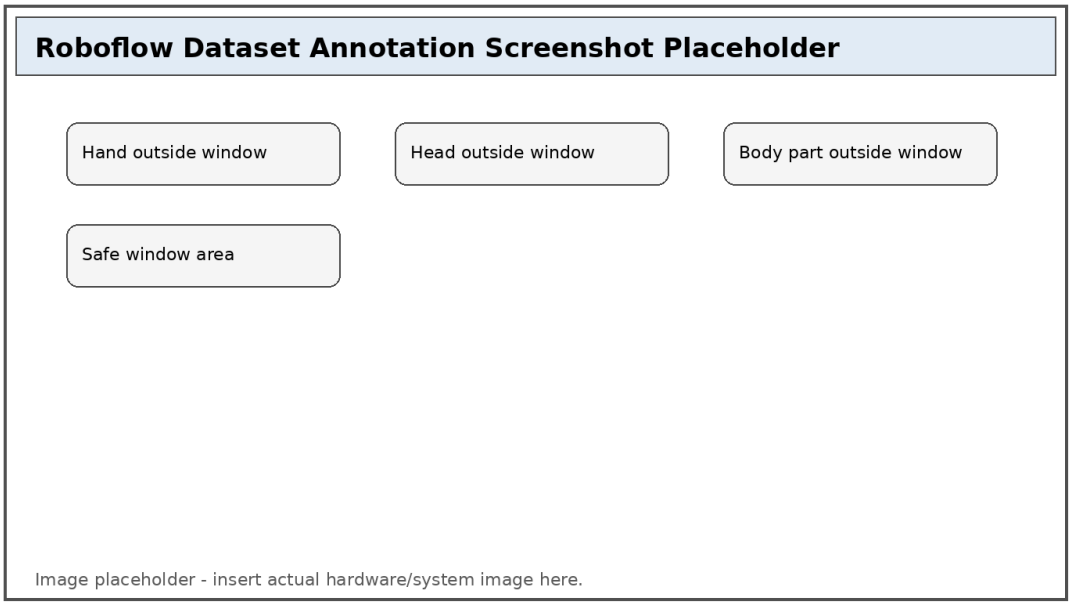



**Figure 3.3: Roboflow Dataset Annotation Screenshot Placeholder**

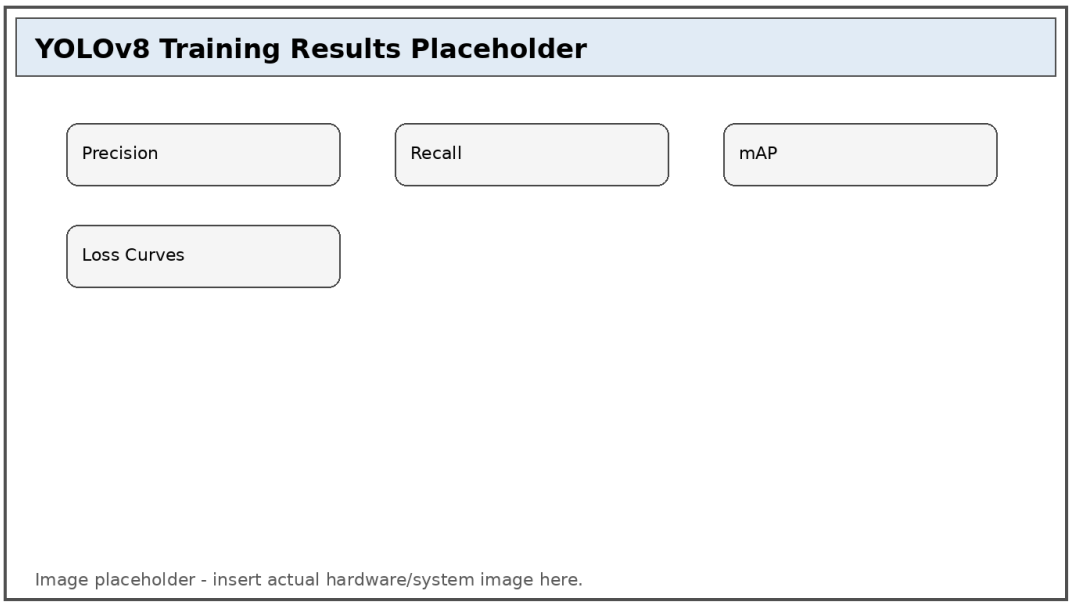



**Figure 3.4: YOLOv8 Training Results Placeholder**

The trained YOLOv8 model was used to process frames from the two cameras. If the model detected a student placing a hand, head, or body part outside the bus window, the system triggered an audio warning. The warning informs students to keep their body parts inside the bus. In the final system, the detected event can also be sent to the driver dashboard through the backend.

### 3.1.6 Audio Warning Implementation

The audio warning mechanism was implemented using a DFPlayer Mini module and speaker. Pre-recorded audio clips can be stored on a memory card and played when a safety event is detected. For window safety, the warning message can instruct students to keep their hands, head, or body parts inside the bus. For accident detection, the warning can notify the driver that a possible accident has been detected and that the 60-second verification period has started.

The audio warning system is important because it provides immediate feedback. In the case of unsafe window behaviour, immediate feedback can prevent injury before it occurs. In the case of accident detection, audio feedback informs the driver that the system has detected an abnormal condition and allows the driver to cancel the alert if it is false.

### 3.1.7 Frontend, Backend and Database Integration

The mobile application frontend was developed using React Native. The frontend can display safety alerts, accident history, bus status, and relevant notification information. Node.js was selected as the backend technology to handle API communication between the mobile application, database, and system modules. Supabase and MongoDB were used to manage structured and flexible data storage requirements.

Supabase can be used for authentication, structured alert data, and real-time updates, while MongoDB can be used to store sensor logs, detection records, and event history. This architecture allows the system to be extended in the future with dashboards, analytics, parent notifications, and school administration views.

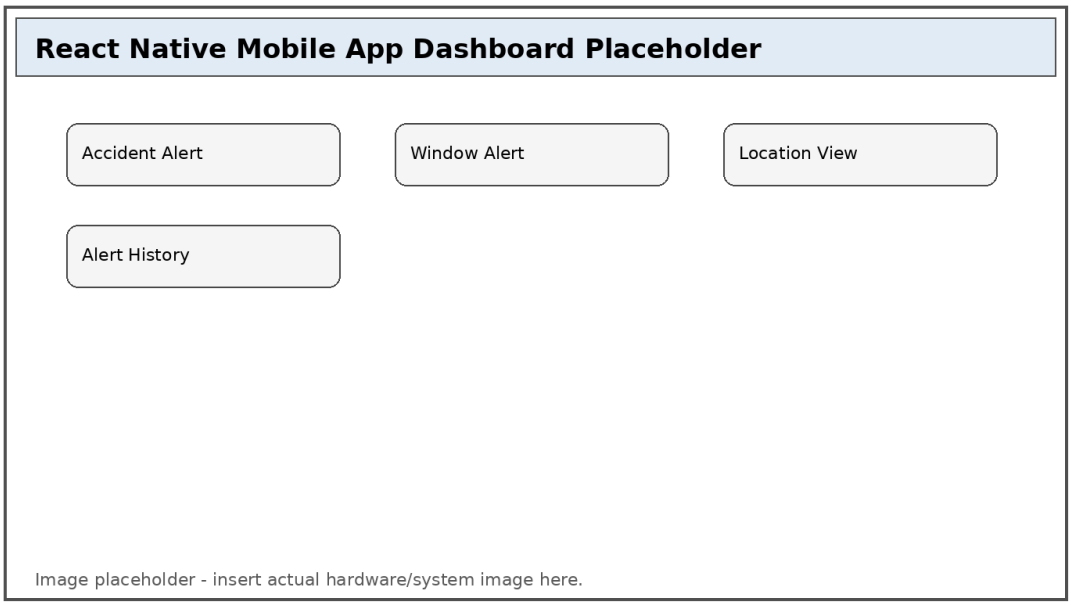



**Figure 3.5: React Native Mobile Application Dashboard Placeholder**

## 3.2 Testing

Testing was carried out to evaluate whether the proposed module performs according to the defined requirements. Since this is a prototype-level research project, testing was conducted using simulated conditions and controlled demonstrations. Each hardware sensor was tested separately before integrated testing. The machine learning model was tested using image/video samples representing safe and unsafe window behaviour.

### 3.2.1 Test Plan and Test Strategy

The test strategy included unit testing, integration testing, and system testing. Unit testing focused on individual sensors and modules. Integration testing focused on combining sensors with ESP32 logic and combining YOLOv8 detection with audio warnings. System testing focused on the complete workflow from detection to alert generation. The main purpose was to verify the response of the system under normal, abnormal, and false alert conditions.

**Table 3.3: Accident Detection Test Cases**

| Test Case ID | Test Scenario | Expected Input / Condition | Expected Output |
|---|---|---|---|
| TC-AD-01 | Simulate rollover / tilt condition | MPU9250 exceeds tilt threshold | Possible accident detected |
| TC-AD-02 | Apply strong vibration to sensor | 801S vibration threshold exceeded | Abnormal vibration detected |
| TC-AD-03 | Apply shock to piezoelectric sensor | Piezoelectric value exceeds threshold | Shock / impact detected |
| TC-AD-04 | Apply pressure to FSR | FSR value increases above threshold | High pressure condition detected |
| TC-AD-05 | Trigger possible accident and press cancel button | Button pressed within 60 seconds | False alert cancelled |
| TC-AD-06 | Trigger possible accident without cancellation | No button press within 60 seconds | Confirmed accident and SMS alert sent |
| TC-AD-07 | Activate fire sensor | Fire threshold triggered | Fire hazard detected |
| TC-AD-08 | Expose MQ2 to smoke/gas test condition | MQ2 value exceeds threshold | Smoke/gas hazard detected |
| TC-AD-09 | Expose raindrop sensor to water | Water sensor value changes | Water hazard warning generated |

**Table 3.4: Window Safety Detection Test Cases**

| Test Case ID | Test Scenario | Expected Input / Condition | Expected Output |
|---|---|---|---|
| TC-WS-01 | Student hand inside the bus window | Safe image/frame | No warning generated |
| TC-WS-02 | Student hand outside the window | Hand outside window detected | Audio warning activated |

| | | | |
|---|---|---|---|
| TC-WS-03 | Student head outside the window | Head outside window detected | Audio warning activated |
| TC-WS-04 | Body part outside the window | Body outside window detected | Audio warning activated |
| TC-WS-05 | Multiple students near window | Unsafe object detected if any body part crosses window boundary | Warning generated only for unsafe behaviour |
| TC-WS-06 | Low light image | Camera frame quality reduced | Model detection evaluated |
| TC-WS-07 | Camera 1 detection | Unsafe behaviour on first monitored side | Audio warning and driver alert generated |
| TC-WS-08 | Camera 2 detection | Unsafe behaviour on second monitored side | Audio warning and driver alert generated |

The test cases in Table 3.3 and Table 3.4 are designed to evaluate both major components of the individual module. The accident detection test cases check hardware sensor responses, false alert cancellation, and SMS alert workflow. The window safety test cases check whether the machine learning model detects unsafe student behaviour correctly and activates the audio warning system.

# 4 RESULTS AND DISCUSSION

## 4.1 Results

The results section presents the expected and prototype-level outcomes of the smart accident detection and machine learning based window safety module. Since the system was developed as a prototype, the results are based on controlled testing scenarios, sensor value observations, simulated accident conditions, and sample camera-based window safety detections. The final numerical values should be updated after completing full hardware testing and machine learning evaluation.

The accident detection component was expected to identify abnormal physical conditions such as tilt, vibration, shock, pressure, fire, smoke, and water exposure. During prototype testing, each sensor should be evaluated separately to determine baseline values under normal conditions and abnormal values under test conditions. These observations help define threshold values for accident and emergency detection.

The 60-second driver verification mechanism is expected to reduce false accident alerts. If a false accident is detected due to road disturbances, the driver can press the push button and cancel the alert. If the driver does not cancel the alert, the system confirms the event and sends a GPS/GSM-based emergency message. This result supports the practical use of the system under real school bus conditions.

The machine learning based window safety component is expected to identify unsafe behaviour such as hands, head, or body parts outside the bus window. When such behaviour is detected, the speaker should immediately play an audio warning. The results of this module should be evaluated using detection accuracy, confidence level, response time, and false positive/false negative observations.

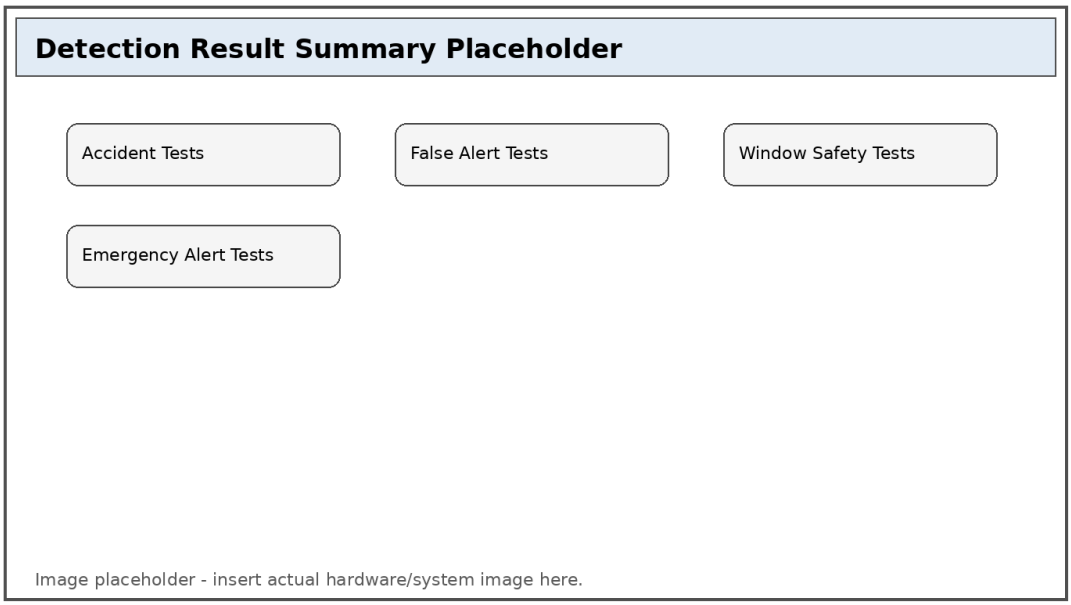



**Figure 4.1: Detection Result Summary Placeholder**

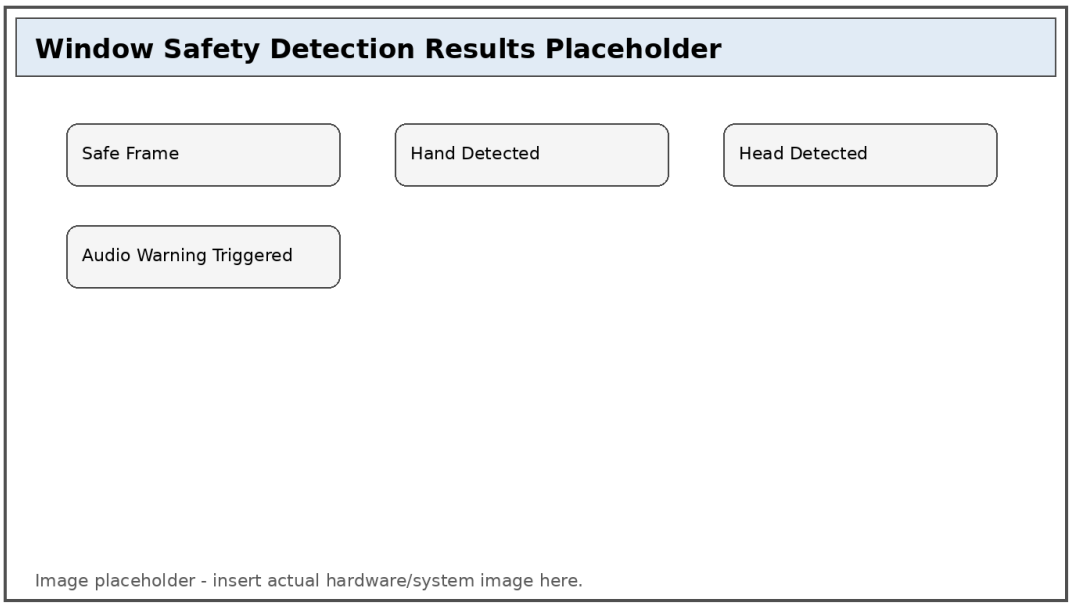



**Figure 4.2: Window Safety Detection Results Placeholder**

**Table 4.1: Prototype Testing Result Summary**

| Test Area | Test Condition | Result | Observation |
|---|---|---|---|
| Accident detection using MPU9250 | Rollover / tilt condition | [Insert result] | [Insert observation] |
| Vibration detection | Abnormal vibration condition | [Insert result] | [Insert observation] |

| | | | |
|---|---|---|---|
| Shock detection | Piezoelectric impact | [Insert result] | [Insert observation] |
| Fire/smoke detection | Fire and gas hazard | [Insert result] | [Insert observation] |
| Water hazard detection | Water exposure | [Insert result] | [Insert observation] |
| False alert cancellation | Driver presses button within 60 seconds | Alert cancelled | Mechanism supports false positive reduction |
| SMS alert generation | Confirmed accident after countdown | SMS sent with location | [Insert GPS sample] |
| Window safety detection | Hand/head/body outside window | Audio warning activated | [Insert YOLO confidence] |

### 4.2 Research Findings

The main finding of this research is that a school bus safety system should not rely only on location tracking or manual supervision. School bus safety requires a broader approach that can identify vehicle-related accident events, emergency hazards, and unsafe student behaviour. The proposed SISURAKSHA module addresses this by combining embedded sensors with machine learning based camera monitoring.

Another finding is that multi-sensor verification is important for accident detection. A single vibration reading or accelerometer reading may not always indicate a real accident. Road disturbances such as potholes and speed bumps can create abnormal values. Therefore, the use of multiple sensors such as MPU9250, 801S vibration sensor, piezoelectric sensor, and FSR sensors improves the reliability of accident identification.

The 60-second driver verification mechanism is an important practical contribution. It provides a balance between fast emergency response and false alert reduction. This is especially useful in real-world school bus environments where rough roads and sudden braking may occur frequently.

The machine learning based window safety module demonstrates that computer vision can be used to identify unsafe student behaviour that is difficult to monitor manually. YOLOv8 is suitable for this task because it supports real-time object detection and can be trained with custom annotated datasets.

The integration of audio warnings is also important because it provides immediate corrective feedback to students. Instead of only notifying the driver, the system can directly warn students when unsafe behaviour is detected. This can reduce injury risk before an accident occurs.

### 4.3 Discussion

The proposed module provides a practical safety solution for school bus transportation by integrating hardware-based accident detection and machine learning based window safety monitoring. Compared with conventional school bus tracking systems, SISURAKSHA focuses not only on where the bus is located but also on what is happening inside and around the bus. This makes the system more relevant to student safety.

The sensor-based accident detection component is useful for detecting sudden physical changes in the bus. However, sensor readings can vary depending on road conditions, mounting position, sensor quality, and threshold values. Therefore, calibration is important before real-world deployment. The thresholds used in prototype testing should be refined using real driving data collected from different road conditions.

The false alert cancellation mechanism helps improve user trust. If parents or emergency contacts receive frequent false messages, they may ignore future alerts. By allowing the driver to cancel false alerts within 60 seconds, the system reduces unnecessary notifications. However, this mechanism also depends on the driver's ability to respond within the given time period. Future versions can include automatic severity classification to reduce driver dependency.

The window safety module depends on camera position, lighting conditions, image quality, and dataset quality. If the dataset is not diverse enough, the model may fail to detect some real-world behaviours. Therefore, the dataset should include different students, clothing types, window angles, lighting conditions, and bus interior conditions. Roboflow and Google Colab support the development process, but real-world performance should be validated with additional test data.

The system architecture using React Native, Node.js, Supabase, and MongoDB supports future expansion. Alerts, logs, images, and sensor data can be stored and

analysed for safety improvements. However, privacy considerations are important when using cameras inside school buses. Any camera-based monitoring system should follow ethical guidelines, data protection practices, and school authority approval.

# 5 SUMMARY OF EACH STUDENT'S CONTRIBUTION

This section summarises the contribution of the student to the SISURAKSHA research project. The overall SISURAKSHA project is a group-based school bus safety system, while this individual report focuses on the smart accident detection and machine learning based window safety module.

**Table 5.1: Summary of Individual Contribution**

| Student | Component | Contribution |
|---|---|---|
| Thanaweera Arachchige Paduma Kasun Shameera | Smart accident detection and machine learning based window safety module | Designed the accident detection workflow, selected sensors, integrated ESP32-based accident/emergency detection logic, proposed 60-second false alert cancellation, developed YOLOv8-based window safety detection workflow, and documented testing and implementation. |
| Group Member 2 | [Insert component name] | [Insert contribution] |
| Group Member 3 | [Insert component name] | [Insert contribution] |
| Group Member 4 | [Insert component name] | [Insert contribution] |

The individual contribution presented in this report focuses on improving school bus safety by detecting accident-related events and preventing unsafe window behaviour. The accident detection component includes sensor selection, hardware logic, emergency alert workflow, and false alert reduction. The window safety component includes camera-based monitoring, YOLOv8 model training workflow, Roboflow dataset preparation, and audio warning generation. These components are integrated with the overall SISURAKSHA architecture using mobile, backend, and database technologies.

# 6 CONCLUSIONS

This research presented a smart accident detection and machine learning based window safety module for the SISURAKSHA school bus safety system. The purpose of the study was to improve student safety during school bus transportation by detecting accident-related events, identifying emergency hazards, reducing false accident alerts, and monitoring unsafe student behaviour near bus windows.

The accident detection component was designed using an ESP32 microcontroller and multiple sensors, including MPU9250, 801S vibration sensor, piezoelectric sensor, force sensitive resistors, MQ2 gas sensor, fire sensor, raindrop sensor, GPS module, and SIM800L GSM module. The system detects conditions such as rollover, sudden speed reduction, abnormal vibration, shock impact, high pressure, fire, smoke, gas, and water exposure. By using multiple sensors, the system improves reliability compared to single-sensor detection approaches.

A key contribution of the system is the 60-second driver verification mechanism. This mechanism allows the driver to cancel false accident alerts caused by potholes, rough roads, or sudden non-critical braking. If the driver does not cancel the alert within 60 seconds, the system confirms the incident and sends an emergency SMS with the GPS location. This improves practical usability by balancing false alert reduction with emergency response.

The window safety component uses two cameras and a YOLOv8-based machine learning model to detect unsafe student behaviour such as placing hands, head, or body parts outside the bus window. When unsafe behaviour is detected, an audio warning is generated through the speaker. This allows the system to correct unsafe behaviour immediately and reduce the risk of injuries caused by passing vehicles, roadside objects, or sudden bus movement.

The proposed module was designed using modern technologies including React Native, Node.js, Supabase, MongoDB, Arduino IDE, Roboflow, Google Colab, and YOLOv8. This technology stack supports the development of a scalable, practical, and modular school bus safety solution. The research demonstrates how embedded

systems, IoT communication, and machine learning can be combined to address real-world transportation safety problems.

Although the proposed system provides a strong prototype-level solution, several limitations remain. The accident detection thresholds require further tuning using real road data. The YOLOv8 window safety model requires a diverse dataset for improved real-world performance. GSM-based SMS communication may depend on network availability, and camera-based monitoring must consider privacy and ethical requirements. Future improvements can include cloud-based dashboards, WhatsApp or mobile push notifications, automatic severity classification, real-time video analytics, improved sensor calibration, and large-scale testing in actual school bus environments.

In conclusion, the proposed SISURAKSHA smart accident detection and window safety module contributes to the development of a safer and more intelligent school bus transportation environment. By integrating accident detection, emergency hazard monitoring, false alert cancellation, GPS/GSM alerting, and machine learning based window safety detection, the system provides a comprehensive approach to protecting school children during daily transportation.

# REFERENCE LIST

[1] National Highway Traffic Safety Administration, School Bus Safety, U.S. Department of Transportation, accessed 2026.

[2] TDK InvenSense, MPU-9250 Product Specification, TDK Corporation, accessed 2026.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[4] Ultralytics, YOLOv8 Documentation, Ultralytics, accessed 2026.

[5] Espressif Systems, ESP32 Technical Reference Manual, Espressif Systems, accessed 2026.

[6] SIMCom, SIM800L Hardware Design Guide, SIMCom Wireless Solutions, accessed 2026.

[7] u-blox, NEO-6 GPS Modules Data Sheet, u-blox AG, accessed 2026.

[8] Roboflow, Roboflow Documentation for Dataset Annotation and Computer Vision Workflows, accessed 2026.

[9] Google, Google Colaboratory Documentation, accessed 2026.

[10] Arduino, Arduino IDE Documentation, accessed 2026.

[11] Meta, React Native Documentation, accessed 2026.

[12] OpenJS Foundation, Node.js Documentation, accessed 2026.

[13] Supabase, Supabase Documentation, accessed 2026.

[14] MongoDB, MongoDB Manual, accessed 2026.

[15] D. J. Fagnant and K. Kockelman, Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations, Transportation Research Part A: Policy and Practice, vol. 77, pp. 167-181, 2015.

[16] S. S. Manvi and S. Tangade, A survey on authentication schemes in VANETs for secured communication, Vehicular Communications, vol. 9, pp. 19-30, 2017.

[17] H. Abdi and L. J. Williams, Principal component analysis, Wiley Interdisciplinary Reviews: Computational Statistics, vol. 2, no. 4, pp. 433-459, 2010.

[18] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2nd ed., 2022.

# APPENDICES

## Appendix A: Sample Arduino IDE Code Placeholders

This appendix should include the final Arduino IDE code used for ESP32 sensor reading, accident detection logic, false alert cancellation, GPS reading, GSM SMS sending, and DFPlayer Mini audio warning control. The code should be inserted after final testing.

```
// SISURAKSHA - Security Sensor Dashboard
// IT22610102 | AsyncWebServer + SSE — production grade
// Zero blocking, zero freezing, auto-recovery
// Sensors: Flame(34) | Rain(35) | MQ2(32) |
//          Pressure(33) | Vibration(26) | MPU6050
// Outputs: Buzzer(25) | LCD I2C(0x27)


#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <WiFi.h>
#include <AsyncTCP.h>           // install: AsyncTCP by ESP32Async
#include <ESPAsyncWebServer.h>  // install: ESPAsyncWebServer by ESP32Async
#include <LiquidCrystal_I2C.h> // install: LiquidCrystal I2C by Frank de Brabander
```

```
// -- WiFi Credentials
const char* WIFI_SSID     = "SLT_FIBER_PYt5z";
const char* WIFI_PASSWORD = "07726Padumapks";

// -- Pin Definitions
#define FLAME_PIN    34
#define RAIN_PIN     35
#define MQ2_PIN      32
#define PRESSURE_PIN 33
#define VIB_PIN      26
#define BUZZER_PIN   25
#define SDA_PIN      18
#define SCL_PIN      19

// -- Thresholds
#define FLAME_THRESHOLD    1000
#define RAIN_THRESHOLD     2500
#define MQ2_THRESHOLD      2000
#define PRESSURE_THRESHOLD 500

// -- Objects
Adafruit_MPU6050  mpu;
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
AsyncWebServer    server(80);
AsyncEventSource  events("/events");  // SSE endpoint

// -- Mutex for shared data
SemaphoreHandle_t dataMutex;

// -- Sensor Data
struct SensorData {
   int   flameRaw   = 4095;
   int   rainRaw    = 4095;
   int   gasRaw     = 0;
   int   pressRaw   = 0;
   int   vibRaw     = 0;
   bool  flameAlert = false;
   bool  rainAlert  = false;
   bool  gasAlert   = false;
   bool  pressAlert = false;
   bool  vibAlert   = false;
   float tempC      = 0.0;
   bool  mpuOK      = false;
} sd;
```

```
// HTML — EventSource dashboard with auto-reconnect

const char HTML_PAGE[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html>
<head>
  <title>SISURAKSHA</title>
  <meta name='viewport' content='width=device-width,initial-scale=1'>
  <style>
    *{box-sizing:border-box}
    body{font-family:Arial,sans-serif;background:#1a1a2e;color:#eee;margin:0;padding:15px}
    h1{color:#e94560;text-align:center;margin-bottom:5px}
    h3{color:#fff;background:#0f3460;padding:8px 12px;border-radius:6px;margin:0}
    .card{background:#16213e;border-radius:10px;padding:15px;margin:12px 0}
    .v{font-size:1.2em;font-weight:bold;color:#00b4d8}
    .alert{color:#ff4444;font-weight:bold;font-size:1.3em}
    .ok{color:#44ff44;font-weight:bold}
    .warn{color:#ffaa00;font-weight:bold}
    .lbl{color:#aaa;font-size:.88em}
    table{width:100%;border-collapse:collapse;margin-top:8px}
    td{padding:7px 4px;border-bottom:1px solid #0f3460}
```

```
    #bar{text-align:center;padding:6px;border-radius:6px;margin:8px 0;font-weight:bold}
    .live{background:#1a3a1a;color:#44ff44}
    .dead{background:#3a1a1a;color:#ff6666}
    #info{text-align:center;color:#00b4d8;font-size:.82em;margin:4px 0}
    #meta{text-align:center;color:#444;font-size:.75em;margin-top:12px}
    .bar-wrap{background:#0f3460;border-radius:6px;height:14px;width:100%;margin-top:4px}
    .bar-fill{height:14px;border-radius:6px;transition:width .3s}
  </style>
</head>
<body>
  <h1>SISURAKSHA</h1>
  <p style='text-align:center;color:#666;margin:0 0 8px'>Security Sensor Dashboard</p>
  <div id='bar' class='dead'>Connecting...</div>
  <p id='info'></p>

  <div class='card'>
    <h3>Flame Sensor</h3>
    <table>
      <tr><td class='lbl'>Status</td>    <td id='fs'><span class='ok'>Clear</span></td></tr>
      <tr><td class='lbl'>Raw Value</td> <td class='v' id='fr'>--</td></tr>
```

```
    <tr><td class='lbl'>Level</td>
      <td><div class='bar-wrap'><div class='bar-fill' id='fb'
style='width:0%;background:#e94560'></div></div></td></tr>
  </table>
 </div>

 <div class='card'>
  <h3>Rain Sensor (HW-028)</h3>
  <table>
    <tr><td class='lbl'>Status</td>  <td id='rs'><span
class='ok'>Dry</span></td></tr>
    <tr><td class='lbl'>Raw Value</td> <td class='v' id='rr'>--</td></tr>
    <tr><td class='lbl'>Level</td>
      <td><div class='bar-wrap'><div class='bar-fill' id='rb'
style='width:0%;background:#00b4d8'></div></div></td></tr>
  </table>
 </div>

 <div class='card'>
  <h3>MQ2 Gas Sensor</h3>
  <table>
    <tr><td class='lbl'>Status</td>  <td id='gs'><span
class='ok'>Clear</span></td></tr>
    <tr><td class='lbl'>Raw Value</td> <td class='v' id='gr'>--</td></tr>
```

```
    <tr><td class='lbl'>Level</td>
      <td><div class='bar-wrap'><div class='bar-fill' id='gb'
style='width:0%;background:#ffaa00'></div></div></td></tr>
  </table>
 </div>

 <div class='card'>
  <h3>Pressure Sensor (RFP602)</h3>
  <table>
    <tr><td class='lbl'>Status</td>  <td id='ps'><span class='ok'>No
Contact</span></td></tr>
    <tr><td class='lbl'>Raw Value</td> <td class='v' id='pr'>--</td></tr>
    <tr><td class='lbl'>Level</td>
      <td><div class='bar-wrap'><div class='bar-fill' id='pb'
style='width:0%;background:#a855f7'></div></div></td></tr>
  </table>
 </div>

 <div class='card'>
  <h3>Vibration Sensor</h3>
  <table>
    <tr><td class='lbl'>Status</td> <td id='vs'><span
class='ok'>Stable</span></td></tr>
  </table>
```

```
  </div>

  <div class='card'>
    <h3>MPU-6050 (Temperature)</h3>
    <table>
      <tr><td class='lbl'>Sensor</td> <td id='ms'><span
class='warn'>Init...</span></td></tr>
      <tr><td class='lbl'>Temp</td>   <td class='v' id='tp'>--</td></tr>
    </table>
  </div>

  <p id='meta'>SISURAKSHA Security Node | 25-26J-282</p>

<script>
  var es, t0 = Date.now(), frames = 0;

  function connect() {
    if (es) es.close();
    es = new EventSource('/events');

    es.onopen = function() {
      document.getElementById('bar').className   = 'live';
      document.getElementById('bar').textContent = 'Live - Streaming';
```

```
};

es.onerror = function() {
  document.getElementById('bar').className   = 'dead';
  document.getElementById('bar').textContent = 'Reconnecting...';
};

es.addEventListener('s', function(e) {
  try {
    var d = JSON.parse(e.data);

    // Flame
    document.getElementById('fs').innerHTML =
      d.fa ? "<span class='alert'>FIRE DETECTED</span>"
           : "<span class='ok'>Clear</span>";
    document.getElementById('fr').textContent = d.fr;
    document.getElementById('fb').style.width =
      Math.min(100, Math.round((4095 - d.fr) / 4095 * 100)) + '%';

    // Rain
    document.getElementById('rs').innerHTML =
      d.ra ? "<span class='warn'>RAIN DETECTED</span>"
           : "<span class='ok'>Dry</span>";
```

```
document.getElementById('rr').textContent = d.rr;
document.getElementById('rb').style.width =
  Math.min(100, Math.round((4095 - d.rr) / 4095 * 100)) + '%';

// Gas
document.getElementById('gs').innerHTML =
  d.ga ? "<span class='alert'>GAS DETECTED</span>"
       : "<span class='ok'>Clear</span>";
document.getElementById('gr').textContent = d.gr;
document.getElementById('gb').style.width =
  Math.min(100, Math.round(d.gr / 4095 * 100)) + '%';

// Pressure
document.getElementById('ps').innerHTML =
  d.pa ? "<span class='warn'>PRESSURE DETECTED</span>"
       : "<span class='ok'>No Contact</span>";
document.getElementById('pr').textContent = d.pr;
document.getElementById('pb').style.width =
  Math.min(100, Math.round(d.pr / 4095 * 100)) + '%';

// Vibration
document.getElementById('vs').innerHTML =
  d.va ? "<span class='alert'>VIBRATION DETECTED</span>"
```

```
        : "<span class='ok'>Stable</span>";

      // MPU Temp
      document.getElementById('ms').innerHTML =
        d.mo ? "<span class='ok'>OK</span>"
             : "<span class='alert'>FAILED</span>";
      document.getElementById('tp').textContent = d.tp.toFixed(1) + ' C';

      // FPS counter
      frames++;
      var now = Date.now();
      if (now - t0 >= 1000) {
        var fps = (frames * 1000 / (now - t0)).toFixed(1);
        document.getElementById('info').textContent =
          fps + ' updates/sec | ' + new Date().toLocaleTimeString();
        frames = 0; t0 = now;
      }
    } catch(err) {}
  });
}

connect();
</script>
```

```
</body>

</html>

)rawliteral";



// Compact JSON builder


String buildJSON() {

  String j = "{";

  j += "\"fr\":"  + String(sd.flameRaw)            + ",";

  j += "\"fa\":"  + String(sd.flameAlert  ? 1 : 0)  + ",";

  j += "\"rr\":"  + String(sd.rainRaw)             + ",";

  j += "\"ra\":"  + String(sd.rainAlert   ? 1 : 0)  + ",";

  j += "\"gr\":"  + String(sd.gasRaw)              + ",";

  j += "\"ga\":"  + String(sd.gasAlert    ? 1 : 0)  + ",";

  j += "\"pr\":"  + String(sd.pressRaw)            + ",";

  j += "\"pa\":"  + String(sd.pressAlert  ? 1 : 0)  + ",";

  j += "\"va\":"  + String(sd.vibAlert    ? 1 : 0)  + ",";

  j += "\"tp\":"  + String(sd.tempC, 1)            + ",";

  j += "\"mo\":"  + String(sd.mpuOK       ? 1 : 0);

  j += "}";

  return j;

}
```

```
// CORE 0 — Sensor Reading Task
// I2C (MPU6050 + LCD) MUST run on same core as WiFi
// to prevent I2C/WiFi bus contention freezing

void sensorTask(void* param) {
  lcd.init();
  lcd.backlight();

  for (;;) {
    // Analog sensors
    int fv = analogRead(FLAME_PIN);
    int rv = analogRead(RAIN_PIN);
    int gv = analogRead(MQ2_PIN);
    int pv = analogRead(PRESSURE_PIN);
    int vv = digitalRead(VIB_PIN);

    // MPU6050 — temp only
    sensors_event_t a, g, t;
    mpu.getEvent(&a, &g, &t);

    if (xSemaphoreTake(dataMutex, pdMS_TO_TICKS(10))) {
```

```
    sd.flameRaw  = fv;
    sd.rainRaw   = rv;
    sd.gasRaw    = gv;
    sd.pressRaw  = pv;
    sd.vibRaw    = vv;
    sd.flameAlert = (fv < FLAME_THRESHOLD);
    sd.rainAlert  = (rv < RAIN_THRESHOLD);
    sd.gasAlert   = (gv > MQ2_THRESHOLD);
    sd.pressAlert = (pv > PRESSURE_THRESHOLD);
    sd.vibAlert   = (vv == HIGH);
    sd.tempC      = t.temperature;
    xSemaphoreGive(dataMutex);
}

// Buzzer — fire alert only
digitalWrite(BUZZER_PIN, sd.flameAlert ? HIGH : LOW);

// LCD update every 1 second
static uint32_t lastLCD = 0;
if (millis() - lastLCD > 1000) {
    lcd.clear();
    lcd.setCursor(0, 0);
    if     (sd.flameAlert) lcd.print("!!! FIRE !!!");
```

```
      else if (sd.gasAlert)   lcd.print("GAS DETECTED");
      else if (sd.vibAlert)   lcd.print("VIB DETECTED");
      else                    lcd.print("SISURAKSHA: OK");
      lcd.setCursor(0, 1);
      lcd.print("T:"); lcd.print(sd.tempC, 1);
      lcd.print("C ");
      lcd.print(sd.rainAlert ? "RAINY" : "DRY");
      lastLCD = millis();
    }

    vTaskDelay(pdMS_TO_TICKS(50));
  }
}


// CORE 1 — SSE Broadcast Task
// Pushes data every 100ms = 10 updates/sec

void broadcastTask(void* param) {
  for (;;) {
    vTaskDelay(pdMS_TO_TICKS(100));

    if (events.count() == 0) continue;
```

```
      String json;
      if (xSemaphoreTake(dataMutex, pdMS_TO_TICKS(10))) {
        json = buildJSON();
        xSemaphoreGive(dataMutex);
      }

      if (json.length() > 0) {
        events.send(json.c_str(), "s", millis());
      }
    }
}

void setup() {
  Serial.begin(115200);
  delay(500);

  dataMutex = xSemaphoreCreateMutex();

  // Pins
  pinMode(FLAME_PIN,    INPUT);
  pinMode(RAIN_PIN,     INPUT);
  pinMode(MQ2_PIN,      INPUT);
```

```
pinMode(PRESSURE_PIN, INPUT);
pinMode(VIB_PIN,      INPUT);
pinMode(BUZZER_PIN,   OUTPUT);
digitalWrite(BUZZER_PIN, LOW);

// MPU6050
Wire.begin(SDA_PIN, SCL_PIN);
if (mpu.begin(0x68)) {
   sd.mpuOK = true;
   mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
   mpu.setGyroRange(MPU6050_RANGE_500_DEG);
   mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
   Serial.println("[MPU] OK");
} else {
   Serial.println("[MPU] FAILED — check wiring!");
}

// WiFi
WiFi.mode(WIFI_STA);
WiFi.setAutoReconnect(true);
WiFi.persistent(true);
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

```
Serial.print("Connecting");
int attempts = 0;
while (WiFi.status() != WL_CONNECTED && attempts < 30) {
   delay(500);
   Serial.print(".");
   attempts++;
}

if (WiFi.status() != WL_CONNECTED) {
   Serial.println("\n[ERROR] WiFi failed - restarting");
   delay(2000);
   ESP.restart();
}

Serial.println();
Serial.println("================================");
Serial.println(" WiFi Connected!");
Serial.print  (" Dashboard: http://");
Serial.println(WiFi.localIP());
Serial.print  (" JSON:      http://");
Serial.print  (WiFi.localIP());
Serial.println("/data");
Serial.println("================================");
```

```
// AsyncWebServer routes
server.on("/", HTTP_GET, [](AsyncWebServerRequest* req) {
  req->send_P(200, "text/html", HTML_PAGE);
});

server.on("/data", HTTP_GET, [](AsyncWebServerRequest* req) {
  String json;
  if (xSemaphoreTake(dataMutex, pdMS_TO_TICKS(10))) {
    json = buildJSON();
    xSemaphoreGive(dataMutex);
  }
  AsyncWebServerResponse* res =
    req->beginResponse(200, "application/json", json);
  res->addHeader("Access-Control-Allow-Origin", "*");
  req->send(res);
});

events.onConnect([](AsyncEventSourceClient* client) {
  Serial.printf("[SSE] Client connected, id=%u\n", client->lastId());
  String json;
  if (xSemaphoreTake(dataMutex, pdMS_TO_TICKS(10))) {
    json = buildJSON();
```

```
        xSemaphoreGive(dataMutex);
      }
      client->send(json.c_str(), "s", millis(), 1000);
    });
    server.addHandler(&events);

    server.begin();
    Serial.println("AsyncWebServer started");

    // All tasks on Core 0 — eliminates I2C/WiFi contention
    xTaskCreatePinnedToCore(sensorTask,    "Sensors",   8192, NULL, 2, NULL, 0);
    xTaskCreatePinnedToCore(broadcastTask, "Broadcast", 4096, NULL, 1, NULL,
0);

    Serial.println("Ready! All systems running.");
}

void loop() {
    // WiFi watchdog — auto reconnect if dropped
    if (WiFi.status() != WL_CONNECTED) {
      Serial.println("[WiFi] Dropped — reconnecting...");
      WiFi.disconnect();
      WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

```
      int attempts = 0;
      while (WiFi.status() != WL_CONNECTED && attempts < 20) {
        vTaskDelay(pdMS_TO_TICKS(500));
        attempts++;
      }
      if (WiFi.status() == WL_CONNECTED) {
        Serial.print("[WiFi] Reconnected: ");
        Serial.println(WiFi.localIP());
      }
    }
    vTaskDelay(pdMS_TO_TICKS(5000));  // check every 5 seconds
}
```

**Appendix B: Windows Safty.py**

```
import cv2
import threading
import torch
import requests
import time
import os
import sys
from pathlib import Path
from ultralytics import YOLO
```

```
from datetime import datetime

import argparse



def _attach_repo_root():
    this_file = Path(_file_).resolve()
    for parent in this_file.parents:
        if (parent / "shared_network_config.py").exists():
            root_path = str(parent)
            if root_path not in sys.path:
                sys.path.append(root_path)
            return



_attach_repo_root()

from shared_network_config import build_phone_video_url, load_network_config

NETWORK_CONFIG = load_network_config()

# ---------------------------------------------------------------------
# SOUND ALERT — plays alert.wav (non-blocking) on each detection
```

```
# ---------------------------------------------------------------------------
_SOUND_PATH = os.path.normpath(
    os.path.join(os.path.dirname(os.path.abspath(_file_)),
                 '..', 'driver_app', 'assets', 'sounds', 'alert.wav')
)
_last_sound_time: dict = {}
SOUND_COOLDOWN = 3  # seconds between sounds for the same detection class


def play_alert_sound(detection_class: str = "default") -> None:
    """Play alert.wav in a background thread (non-blocking). Per-class cooldown."""
    now = time.time()
    if now - _last_sound_time.get(detection_class, 0) < SOUND_COOLDOWN:
        return
    _last_sound_time[detection_class] = now


    def _play():
        try:
            if sys.platform == "win32":
                import winsound
                winsound.PlaySound(
                    _SOUND_PATH,
                    winsound.SND_FILENAME | winsound.SND_ASYNC |
winsound.SND_NOWAIT
```

```
            )
        else:
            import subprocess
            player = "afplay" if sys.platform == "darwin" else "aplay"
            subprocess.Popen([player, _SOUND_PATH],
                             stdout=subprocess.DEVNULL,
                             stderr=subprocess.DEVNULL)
    except Exception as e:
        print(f"[Sound] Could not play alert: {e}")

threading.Thread(target=_play, daemon=True).start()
# ---------------------------------------------------------------------

# --- DEFAULT CONFIGURATION ---
DEFAULT_SERVER_URL =
NETWORK_CONFIG["WINDOW_SAFETY_SERVER_URL"]
DEFAULT_DRIVER_ID = NETWORK_CONFIG["DRIVER_ID"]
DEFAULT_PHONE_IP = NETWORK_CONFIG["PHONE_IP"]

# Initialize parser
parser = argparse.ArgumentParser(description="Window Safety Monitoring
System")
parser.add_argument("--driver_id", type=str, default=DEFAULT_DRIVER_ID,
help="Driver UUID")
```

```
parser.add_argument("--server_url", type=str, default=DEFAULT_SERVER_URL,
help="Backend API URL for window safety")

parser.add_argument("--phone_ip", type=str, default=DEFAULT_PHONE_IP,
help="IP address of the phone camera (e.g. 192.168.1.103:8080)")


args = parser.parse_args()


SERVER_URL = args.server_url

DRIVER_ID = args.driver_id

PHONE_IP = args.phone_ip

VIDEO_URL = build_phone_video_url(PHONE_IP)


# --- GPU ACCELERATION ---

DEVICE = 0 if torch.cuda.is_available() else 'cpu'

USE_HALF = torch.cuda.is_available()  # FP16 only on GPU

print(f"Using device: {'GPU (CUDA)' if torch.cuda.is_available() else 'CPU'}")


# --- ALERT TRACKING ---

last_alert_time = {}

ALERT_COOLDOWN = 5  # Seconds between alerts for same detection type


# --- SERVER COMMUNICATION ---

def send_heartbeat():
```

```
    """Send heartbeat every 5 seconds to server"""
    while True:
        try:
            response = requests.post(
                f"{SERVER_URL}/heartbeat",
                json={"driver_id": DRIVER_ID},
                timeout=3
            )
            if response.status_code == 200:
                data = response.json()
                print(f"[{datetime.now().strftime('%H:%M:%S')}] Heartbeat sent - System {'enabled' if data.get('system_enabled') else 'disabled'}")
            else:
                print(f"[{datetime.now().strftime('%H:%M:%S')}] Heartbeat failed: {response.status_code}")
        except requests.exceptions.RequestException as e:
            print(f"[{datetime.now().strftime('%H:%M:%S')}] Heartbeat error: {str(e)[:50]}")
        time.sleep(5)


def send_alert(alert_type, severity, message, confidence=None):
    """Send alert to server with cooldown"""
    global last_alert_time
    current_time = time.time()
```

```
# Check cooldown
alert_key = f"{alert_type}_{severity}"
if alert_key in last_alert_time:
    if current_time - last_alert_time[alert_key] < ALERT_COOLDOWN:
        return False

last_alert_time[alert_key] = current_time

try:
    payload = {
        "driver_id": DRIVER_ID,
        "alert_type": alert_type,
        "severity": severity,
        "status": severity,
        "message": message
    }
    if confidence:
        payload["confidence"] = confidence

    response = requests.post(
        f"{SERVER_URL}/alerts",
        json=payload,
```

```
            timeout=3
        )
        if response.status_code == 201:
            print(f"[{datetime.now().strftime('%H:%M:%S')}] Alert sent: {severity} - {message}")
            return True
        else:
            print(f"[{datetime.now().strftime('%H:%M:%S')}] Alert failed: {response.status_code}")
            return False
    except requests.exceptions.RequestException as e:
        print(f"[{datetime.now().strftime('%H:%M:%S')}] Alert error: {str(e)[:50]}")
        return False

# --- THREADED CAMERA CLASS (reduces latency) ---
class FastCamera:
    def _init_(self, url):
        self.cap = cv2.VideoCapture(url, cv2.CAP_FFMPEG)
        self.cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)
        self.ret, self.frame = self.cap.read()
        self.stopped = False
        threading.Thread(target=self.update, daemon=True).start()
```

```
    def update(self):
        while not self.stopped:
            self.ret, self.frame = self.cap.read()

    def get_frame(self):
        return self.frame

    def release(self):
        self.stopped = True
        self.cap.release()

# --- START HEARTBEAT THREAD ---
heartbeat_thread = threading.Thread(target=send_heartbeat, daemon=True)
heartbeat_thread.start()
print(" Heartbeat thread started")

# --- INITIALIZE MODEL ---
model = YOLO('kasun_model.pt')
model.fuse()  # Fuse layers for faster inference

# --- CONNECT TO CAMERA ---
print(f"Connecting to IP Camera: {PHONE_IP}...")
cam = FastCamera(VIDEO_URL)
```

```
print(f" Connected to IP Camera: {PHONE_IP}")
print(f" Model classes: {model.names}")
print(f" Server URL: {SERVER_URL}")
print("-" * 50)

# --- MAIN DETECTION LOOP ---
while True:
    frame = cam.get_frame()
    if frame is None:
        continue

    # Resize frame for faster processing
    frame = cv2.resize(frame, (640, 480))

    # Run YOLOv8 detection - OPTIMIZED for low latency
    results = model.predict(
        frame,
        imgsz=320,         # Smaller = much faster
        conf=0.4,          # Confidence threshold
        device=DEVICE,     # GPU if available
        half=USE_HALF,     # FP16 for speed
        verbose=False      # No console spam
    )
```

```
    # Check if any objects were detected
    for r in results:
        if len(r.boxes) > 0:
            # Iterate through each detection
            for box in r.boxes:
                cls_id = int(box.cls[0])
                confidence = float(box.conf[0])
                class_name = model.names[cls_id]

                # Determine severity and alert_type based on detection
                # head = DANGER, hand = WARNING, body = WARNING
                if "head" in class_name.lower():
                    alert_type = "head_detected"
                    severity = "DANGER"
                    message = f"Head detected outside window! Confidence: {confidence:.2%}"
                elif "hand" in class_name.lower():
                    alert_type = "hand_detected"
                    severity = "WARNING"
                    message = f"Hand detected outside window! Confidence: {confidence:.2%}"
                elif "body" in class_name.lower():
```

```
                alert_type = "body_detected"
                severity = "WARNING"
                message = f"Body detected outside window! Confidence:
{confidence:.2%}"
            else:
                alert_type = "unknown_detected"
                severity = "WARNING"
                message = f"{class_name} detected! Confidence: {confidence:.2%}"

            # Send alert to server with specific detection type
            send_alert(
                alert_type=alert_type,
                severity=severity,
                message=message,
                confidence=confidence
            )

        # Draw warning text on frame
        cv2.putText(frame, "!! SAFETY VIOLATION !!", (50, 50),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 3)

    # Show the frame with YOLO boxes
    annotated_frame = results[0].plot()
```

```
    cv2.imshow("Window Safety System", annotated_frame)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        print("\n Shutting down...")
        break

cam.release()
cv2.destroyAllWindows()
print(" Window Safety System stopped")
```

### Appendix C: Sample Alert Message Format

Sample emergency SMS message format:

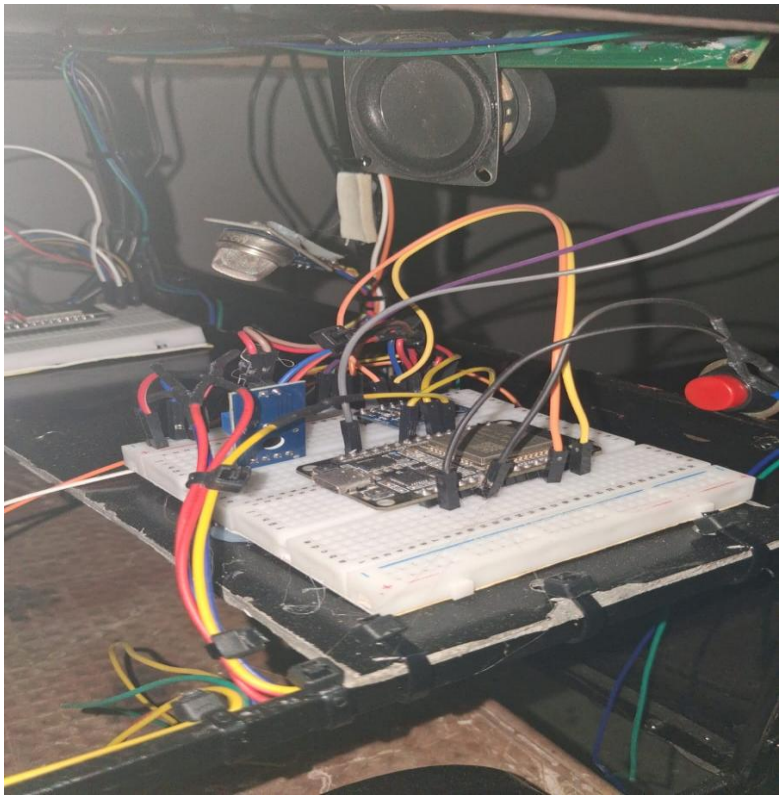
### Appendix D: Dataset Annotation Classes

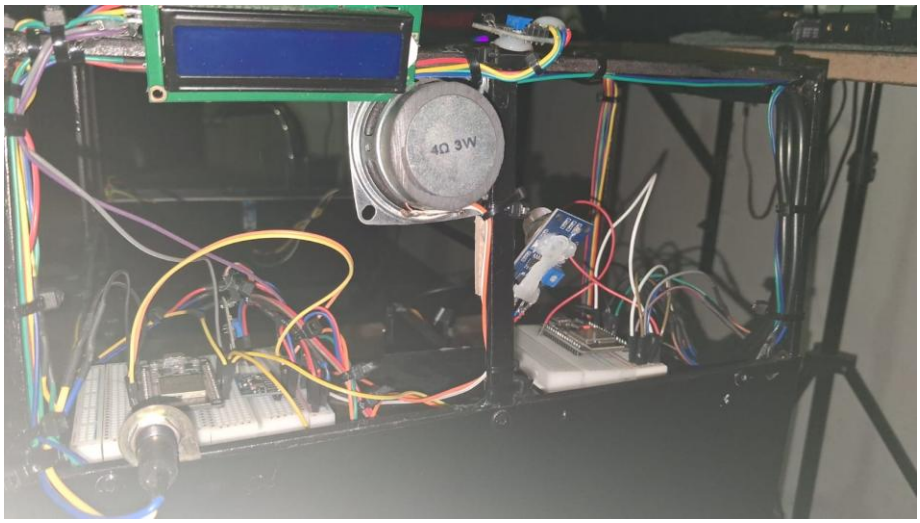
**Appendix Table C.1: Suggested Dataset Annotation Classes**

| Class Name | Description |
|---|---|
| Full hand | Student hand placed outside the bus window |
| head | Student head placed outside the bus window |
| Full Body | Any unsafe body part extended outside window |

### Appendix E: Additional Prototype Photographs

This appendix should include additional hardware prototype images, sensor close-up images, Roboflow screenshots, Google Colab training screenshots, and mobile application screenshots.





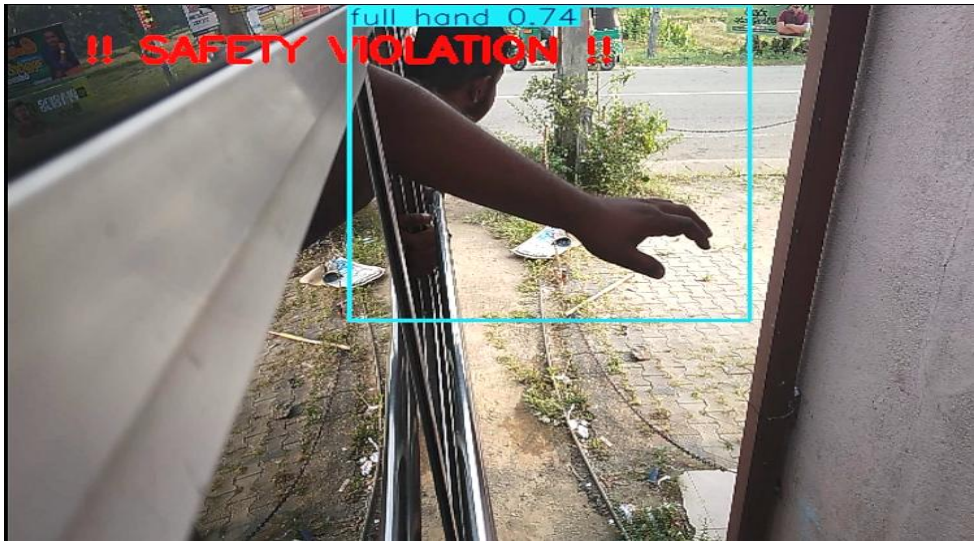

**Appendix F:  Testing Model Training**


full_hand 0.84
!! SAFETY VIOLATION !!


full_hand 0.45
full_hand 0.81
!! SAFETY VI


full_hand 0.83
!! SAFETY VIOLATION !!


full_hand 0.41
!! SAFETY VIOLATION !!


full_hand 0.74
!! SAFETY VIOLATION !!

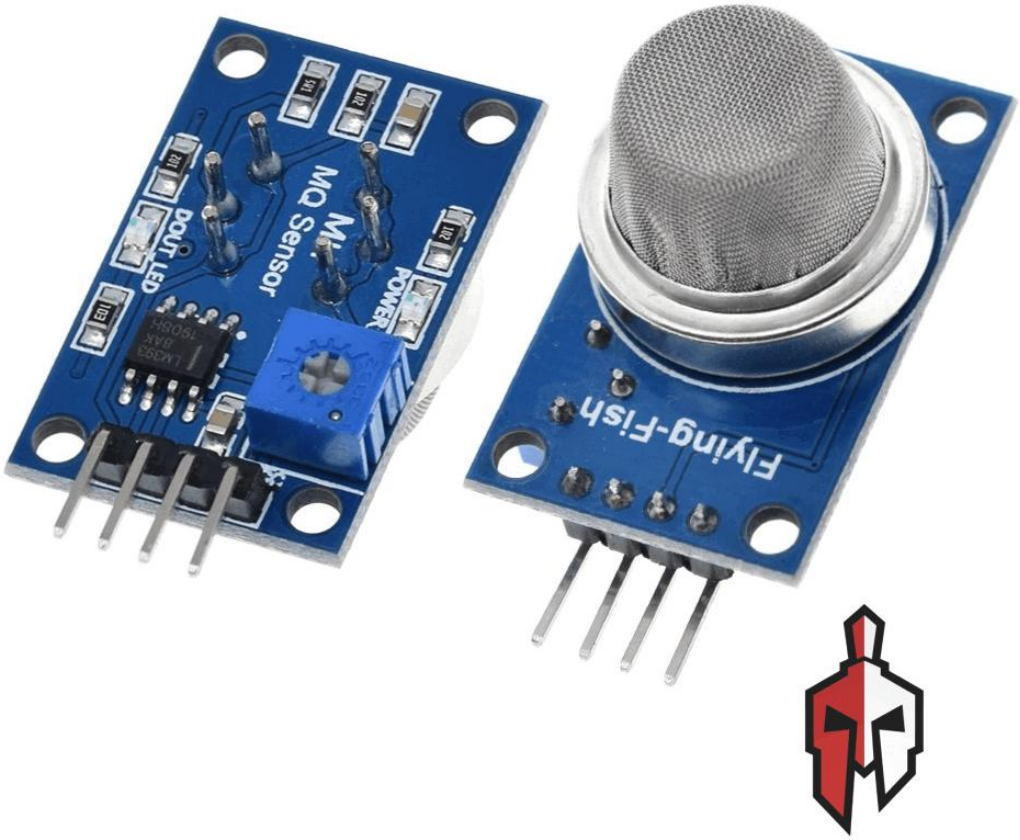

full_hand 0.50
head 0.74
!! SAFETY VIOLATION !!

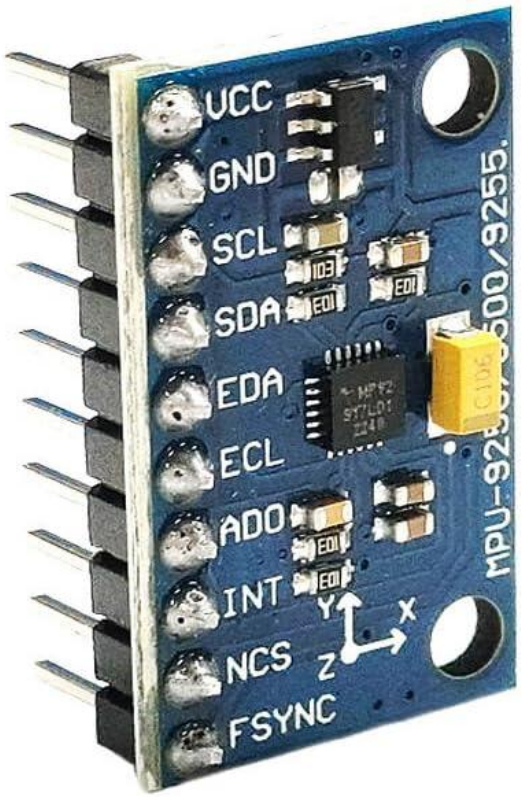

head 0.54
!! SAFETY VIOLATION !!


**Appendix G: Image of sensers**

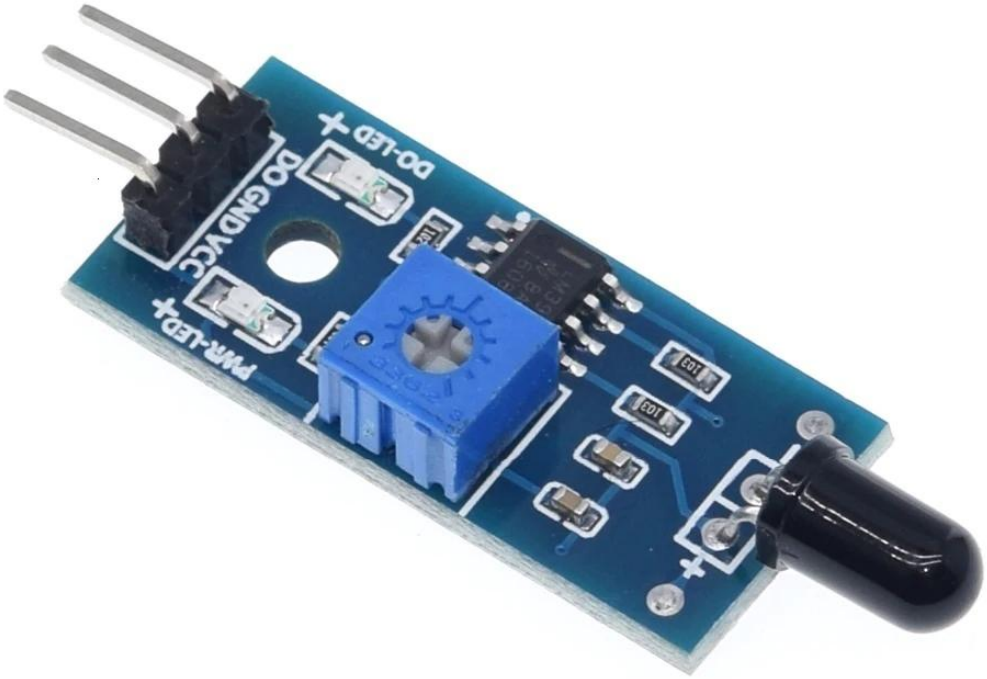

SIM800L

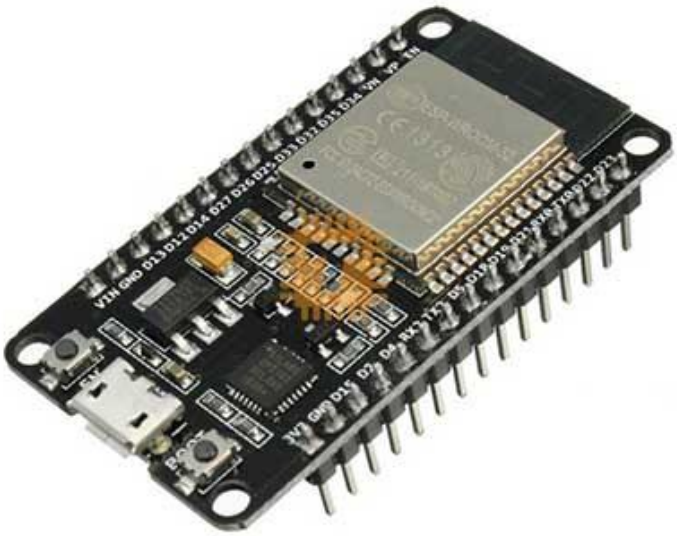

FC-37

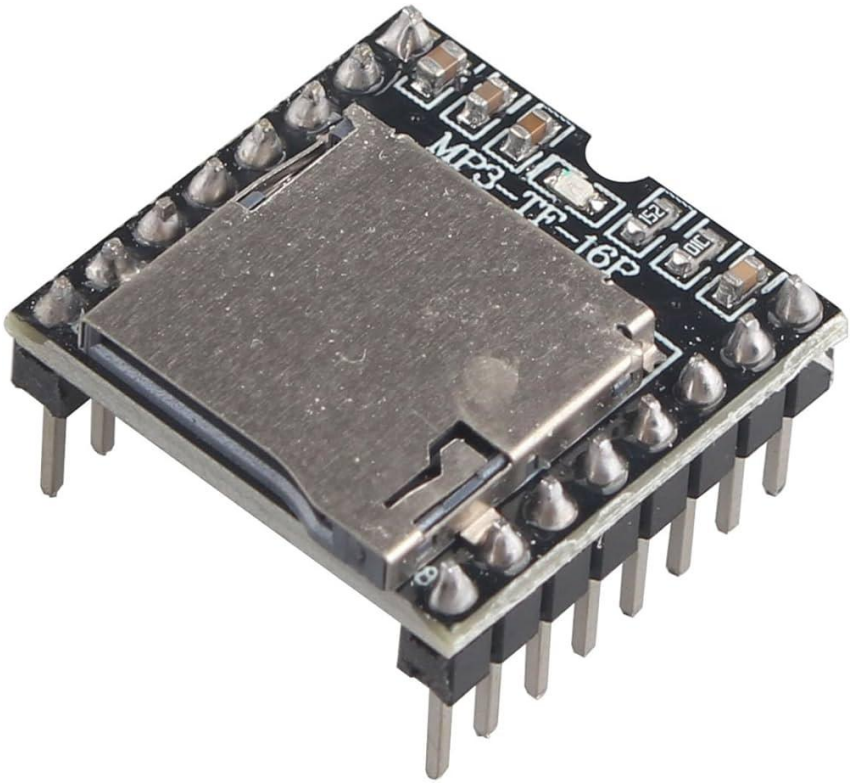

MQ Sensor
POWER
Flying-Fish

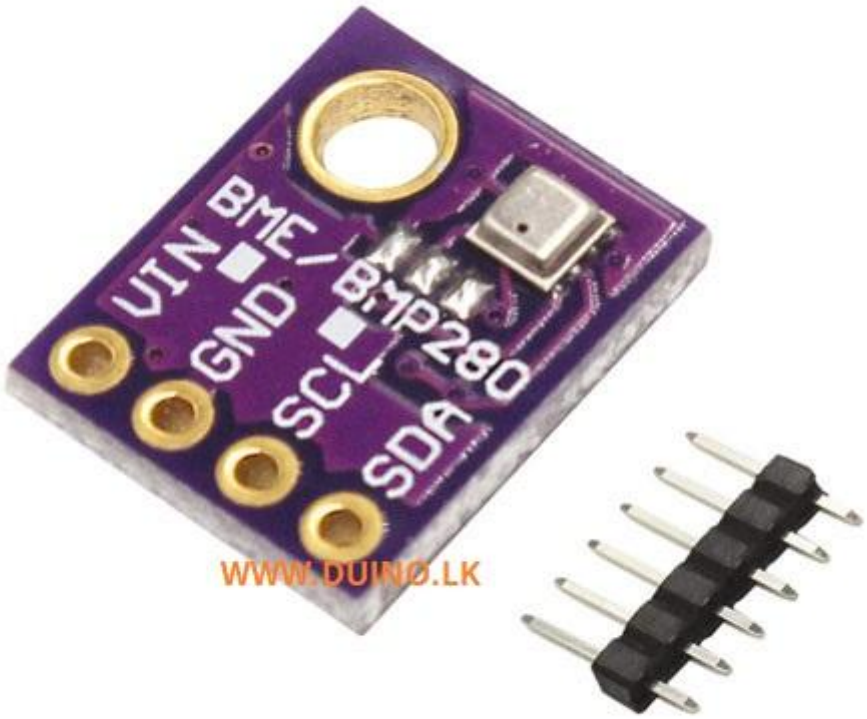

VCC
GND
SCL
SDA
EDA
ECL
ADO
INT
NCS
FSYNC
Y
X
Z


IN+
IN-
OUT+
OUT-
100
470


DO-LED
DO GND VCC
PWR-LED




MP3-TF-16P


VIN
GND
SCL
SDA
BME/
BMP280
WWW.DUINO.LK