

# **“STUDY IN SMART ACCIDENT PREVENTION, DETECTION, AND REPORTING SYSTEM (SAPDRS)”**

## **Major Project**

**Submitted For The Partial Fulfilment Of Diploma In Engineering & Technology**

**(Electronics And Tele-Communication Engineering)**

**Approved by A.I.C.T.E**

**(WEST BENGAL STATE COUNCIL OF TECHNICAL & VOCATIONAL EDUCATION AND SKILL  
DEVELOPMENT)**



**Submitted by:**

**Kunal Das, Arindam Das**

**Nirmal Barman , Chanchal Mandal**

**Bivash Das, Shilajit Paul**

**Raj Barman**

**(Batch: 2022-2025)**



**Under the guidance of: Mr. Lakhan Deo Oraon**

**Government Of West Bengal**

**FALAKATA POLYTECHNIC**

**Department Of Electronics & Telecommunication Engineering**

**Baganbari, P.O- Falakata, Dist: - Alipurduar, West Bengal – 735211**

**Semester - 5<sup>th</sup>**

**Session - July To December, 2024**

**Phone No: - 03563-299677**

**E-mail: - falakatapoly@yahoo.co.in**

**DECLARATION**

We declare that this project report “*Smart Accident Prevention, Detection, and Reporting System*” is the record of authenticate work carried out by us during the period of our final year and has not been submitted to any other institute for the award of any diploma, etc.

-----  
Place:  
Date:

-----  
Place:  
Date:

-----  
Place:  
Date:

-----  
Place:  
Date:

-----  
Place:  
Date:

-----  
Place:  
Date:

-----  
Place:  
Date:



Govt. of West Bengal  
**FALAKATA POLYTECHNIC**  
Baganbari, Falakata, Dist-Alipurduar  
Department: Electronics and Telecommunication Engineering



**Certificate of Approval**

This is to certify that the project report on “Smart Accident Prevention, Detection, and Reporting System (SAPDRS)” carried out by **Kunal Das, Arindam Das, Nirmal Barman, Bivash Das, Chanchal Mandal, Shilajit Paul, Raj Barman** under my guidance and supervision.

In my opinion the report in its present form is in conformity as specified by **FALAKATA POLYTECHNIC** and as per regulation of the **WEST BENGAL STATE COUNCIL OF TECHNICAL & VOCATIONAL EDUCATION & SKILL DEVELOPMENT** to the best of my knowledge. The result presented here are original in nature and worthy of incorporation in project report for the Diploma program in **ELECTRONICS & TELECOMMUNICATION ENGINEERING**.

-----  
Signature of  
Principle/PIC.  
Mr. Mrinal Kanti Byadh  
Principal – in – Charge  
Falakata Polytechnic

-----  
Signature of H.O.D.  
Mr. Arghya Halder  
Department of ETCE  
Falakata Polytechnic

-----  
Signature of Project  
Guide  
Mr. Lakhan Deo  
Oraon  
Lecture in ETCE  
Falakata Polytechnic

## **ACKNOWLEDGMENT**

*This project is a collaborative effort with sincerity and determination with the title "Smart Accident Prevention, Detection, and Reporting System (SAPDRS)". It was a plan with proper guided experiments that started long. Now when it is ready, we would like to acknowledge the people behind the curtain, the project is successful only when such people are brought into limelight. Developing programs successful is only a part of the story and other part completes only when such people are shown in light.*

*We express deep sense of gratitude to the kind help and blessings from **Mr. Mrinal Kranti Byadh**, Principal-in-Charge, F0alakata Polytechnic.*

*We are thankful to **Mr. Arghya Halder**, Head of the Department of Electronics and Telecommunication Engineering lecturer and also our project guide who guided us throughout our project and installed a root of inspiration in us.*

*We are also thankful to Mr. Arghya Haldar, Mr. Lakhan Deo Oraon and Mrs. Chaitali Bhadra, lecturers of the Department for their constant support.*

*We hope such ways of blessing will shower on our lives. Finally, we acknowledge the encouragement and support of our classmates and several others whose help cannot be forgotten.*

*We tried our level best to produce an error free report but if there is any misspelled word any other error, it is purely unintentional.*

### **NAME**

### **SIGNATURE**

- |                    |   |       |
|--------------------|---|-------|
| 1. Kunal Das       | : | ----- |
| 2. Arindam Das     | : | ----- |
| 3. Nirmal Barman   | : | ----- |
| 4. Chanchal Mandal | : | ----- |
| 5. Bivash Das      | : | ----- |
| 6. Shilajit Paul   | : | ----- |
| 7. Raj Barman      | : | ----- |

## **CONTENTS**

1. ABSTRACT-----	1
2. INTRODUCTION-----	2
3. REVIEW OF LITERATURE -----	3
4. THEORETICAL DISCUSSION -----	5
5. EXPERIMENTAL DISCUSSION -----	50
6. EXPENSES-----	87
7. FUTURE SCOPE-----	89
8. CONCLUSION -----	93
9. REFERENCES-----	94

## **LIST OF FIGURES**

FIGURE 1: ESP32 (38 PINS) -----	5
FIGURE 2: ESP 32 CHIP -----	6
FIGURE 3: ESP32 PINOUT -----	9
FIGURE 4: MQ3 MODULE -----	12
FIGURE 5: MQ3 SENSOR-----	13
FIGURE 6: MQ3 ALCOHOL SENSOR WORKING -----	13
FIGURE 7: ICS AND PINOUT OF MQ3 ALCOHOL SENSOR-----	14
FIGURE 8: ULTRASONIC SENSOR (HC-SR04) -----	14
FIGURE 9: WORKING PRINCIPLE OF ULTRASONIC SENSOR (HC-SR04) -----	15
FIGURE 10: ULTRASONIC SENSOR (HC-SR04) PINOUT -----	16
FIGURE 11: GSM MODULE (SIM900A)-----	18
FIGURE 12: GPS MODULE (NEO-6M) -----	21
FIGURE 13: GPS MODULE (NEO-6M) CHIP -----	21
FIGURE 14: GPS MODULE (NEO-6M) CHIP PINOUT -----	22
FIGURE 15: GPS MODULE (NEO-6M) COMPONENTS -----	22
FIGURE 16: GPS MODULE (NEO-6M) PINOUT-----	23
FIGURE 17: MPU6050 -----	25
FIGURE 18:ORIENTATION OF MPU6050-----	26
FIGURE 19: MPU6050 CHIP-----	26
FIGURE 20: MPU6050 PINOUT -----	27
FIGURE 21: 1.3-INCH OLED DISPLAY MODULE -----	28
FIGURE 22: 1.3-INCH OLED DISPLAY MODULE PINOUT -----	29
FIGURE 23: L293D MOTOR DRIVER MODULE -----	31
FIGURE 24: L293D MOTOR DRIVER MODULE PINOUT -----	32
FIGURE 25: TP4056 MODULE-----	34
FIGURE 26: TP4056 MODULE ICS AND COMPONENT-----	34
FIGURE 27: TP4056 MODULE PINOUT-----	35
FIGURE 28: STEP-DOWN (BUCK) VOLTAGE REGULATOR -----	36
FIGURE 29: STEP-DOWN (BUCK) VOLTAGE REGULATOR PINOUT -----	37
FIGURE 30: AMS1117 5V STEP-DOWN POWER SUPPLY MODULE -----	38
FIGURE 31: AMS1117 PINOUT -----	40
FIGURE 32: PIR SENSOR -----	41

FIGURE 33: WORKING PRINCIPLE OF PIR -----	42
FIGURE 34: PINOUT OF PIR-----	42
FIGURE 35: ADJUSTABLE PARAMETERS-----	43
FIGURE 36: 12V DC FAN -----	43
FIGURE 37: IC 7805 -----	44
FIGURE 38: CAPACITOR -----	44
FIGURE 39: BUZZER -----	45
FIGURE 40: LED -----	45
FIGURE 41:DC SWITCH -----	46
FIGURE 42: DC MOTOR -----	46
FIGURE 43: BATTERY (18650) -----	46
FIGURE 44: ARDUINO IDE -----	47
FIGURE 45: BLOCK DIAGRAM-----	50

**LIST OF TABLE**

TABLE 1: ESP32 PIN DESCRIPTION----- 8

TABLE 2: SYSTEM PROGRAM CODE -----61



## **ABSTRACT**

The "Smart Accident Prevention, Detection, and Reporting System (SAPDRS)" project, which is presented here, meets this urgent need of intelligent systems to improve road safety by preventing and detecting and reporting on vehicle accidents. This system has sensors, microcontrollers, and modules for communication as a full-fledged safety solution. The prevention wing will deal with real-time monitoring of speed, proximity between vehicles, and the alertness of the driver. Advanced sensors, such as ultrasonic and infrared sensors, continually collect data to monitor for unsafe conditions and alert the driver. In an accident, it is the detection module that relies on impact sensors and accelerometers to determine if collisions occurred and which onboard microcontroller determines the severity of impact.

The moment an accident is sensed, the reporting module starts transmitting SMS automatically towards emergency services and predefined contacts with information, which would prove to be very crucial, such as GPS location and details about the accident. The system provides real-time communication using GSM/GPRS technology and GPS for location tracking, which is precise to ensure timely help and quicker response times. This multi-layered approach minimizes accidents, responds quickly, and might even reduce deaths through improved emergency assistance. This project has contributed considerably to road safety through the integration of Internet of Things technologies, sensor fusion, and wireless communication to offer a proactive and reactive safety solution for modern transportation.

## INTRODUCTION

Accidents remain a leading cause of injury and fatalities worldwide. Often, the critical factor determining the survival and recovery of accident victims is how quickly emergency services can be alerted and dispatched to the scene. Unfortunately, many accidents go unreported, or are reported too late, which delays medical assistance and can lead to severe consequences. The lack of prompt emergency response can be worsened by the public's reaction to such situations; while some individuals may offer help, others may fail to act or prioritize taking photos or videos over immediate assistance.

Our friend's experience in 2021 highlighted this issue profoundly. He was involved in a traffic accident when a speeding truck lost control, striking the back of his bike. Although he escaped severe injury, his foot was critically hurt. Some bystanders came to offer help, yet none called an ambulance or helped him get to the hospital. This delay in receiving medical assistance underscored a significant problem: while accidents are common, timely intervention is rare, and lives are often lost due to a lack of rapid response. This realization inspired him to create an automatic accident prevention, detection, and reporting system, aiming to save lives by enabling quick, automated responses to accidents.

The primary goal of this project is to develop a system that can automatically prevent, detect, and report accidents, ensuring a swift response that can save lives. The objectives are to use real-time data from sensors to identify potential hazards or erratic driving behaviour, helping prevent accidents. It also aims to identify the occurrence of accidents instantly by analysing various signals from sensors. Additionally, the system will automatically send alerts with real-time data, including location, to emergency services, family members, or relevant authorities, thereby expediting medical assistance and intervention.

Our friend's personal experience was a driving force behind the creation of this system. Being in an accident made him aware of how helpless one can feel, waiting for assistance that may not come in time. Seeing people take pictures instead of calling for help also left a lasting impression. This project is his effort to make sure that people in similar situations receive timely support and medical attention, reducing preventable losses and suffering.

## REVIEW OF LITERATURE

Car safety has evolved significantly since the invention of the automobile, driven by the growing demand for safer transportation and the recognition of the societal costs of traffic accidents. The journey of automotive safety reflects a transformative process from simple machines with little regard for occupant safety to sophisticated systems designed to prevent accidents and minimize injuries. This literature review explores the evolution of car safety, focusing on historical data and key milestones in the development of safety features and technologies.

The development of car safety began with the introduction of safety cages in the 1930s, designed to protect occupants by absorbing crash energy. The 1950s marked a pivotal moment with the introduction of seat belts, becoming mandatory in 1966 and significantly reducing fatalities. The 1970s and 1980s saw the rise of passive safety features like airbags and improved crumple zones, which evolved to offer better protection in collisions. The 1990s and early 2000s shifted focus to active safety systems, including anti-lock braking systems (ABS), traction control systems (TCS), and electronic stability control (ESC), aimed at preventing accidents by enhancing vehicle stability and control.

The late 20th and early 21st centuries also saw the introduction of advanced driver assistance systems (ADAS). These systems, such as lane departure warning, adaptive cruise control, and collision warning, are designed to assist the driver in avoiding accidents by providing warnings or intervening when necessary. The advent of autonomous braking and blind spot detection further elevated the capabilities of ADAS, leading to a safer driving experience. These systems are now seen as essential components of modern vehicles, and their implementation has significantly reduced the incidence of accidents caused by driver error.

The role of government regulation in shaping car safety cannot be overstated. Laws and regulations regarding vehicle safety have been instrumental in forcing manufacturers to adopt new technologies and features. The creation of safety standards by organizations such as the National Highway Traffic Safety Administration (NHTSA) in the U.S. and similar bodies worldwide has pushed car manufacturers to prioritize safety. These regulations have led to the introduction of standards for crash tests, the establishment of safety ratings for vehicles, and the implementation of mandatory features such as airbags, seat belts, and electronic stability control. Crash testing, for example, has become an essential part of the automotive industry, with organizations like the Insurance Institute for Highway Safety (IIHS) conducting rigorous

crash tests to evaluate the safety of vehicles. These tests have had a significant influence on vehicle design, driving improvements in structural integrity, safety features, and overall vehicle performance in accidents.

In recent years, the integration of connected and smart technologies has brought another layer of safety to the automotive world. Vehicle-to-vehicle (V2V) communication, vehicle-to-infrastructure (V2I) systems, and automated driving technologies are some of the most promising innovations in modern car safety. These systems use sensors, cameras, and communication networks to enable vehicles to communicate with each other and with traffic infrastructure, improving awareness and helping prevent accidents. The goal of these systems is to create an environment where vehicles are aware of their surroundings and can make autonomous decisions to avoid collisions. While fully autonomous cars are not yet widespread, many of these technologies are already being incorporated into consumer vehicles, improving safety in the process.

As the industry moves towards the future, artificial intelligence (AI) and machine learning (ML) are expected to play a pivotal role in enhancing car safety. AI has the potential to revolutionize the way cars respond to dangerous situations, allowing vehicles to make real-time decisions based on vast amounts of data gathered from sensors, cameras, and other sources. The development of autonomous vehicles, though still in its infancy, promises to eliminate human error—the leading cause of most traffic accidents. With autonomous driving, vehicles will be equipped with the capability to detect obstacles, predict potential hazards, and navigate safely without direct human intervention. This shift represents the future of automotive safety, where cars themselves will become the first line of defence against accidents.

The historical development of car safety is a testament to the relentless efforts of the automotive industry, governments, and researchers to create safer vehicles. From the basic safety cage of the early years to the sophisticated active and passive systems of today, car safety has evolved significantly, driven by both technological advancements and societal demands for safer transportation. The future of car safety looks even more promising, with autonomous vehicles and connected technologies poised to take the next step in reducing traffic accidents and saving lives. As automotive technology continues to advance, it is clear that the focus on safety will remain at the forefront of innovation.

## THEORETICAL DISCUSSION

The "Accident Prevention, Detection, and Reporting System" aims to leverage modern technology to minimize road accidents, detect accidents when they occur, and ensure rapid reporting to appropriate emergency services for timely intervention. This project integrates advanced sensors, communication modules, and software algorithms to achieve these objectives, contributing to safer transportation systems. The details of the components and technologies involved are outlined below:

### ❖ Components and Technologies Involved –

#### ⇒ Hardware Used –

##### 1. Microcontrollers –

The ESP32 is a low-cost, low-power microcontroller with built-in Wi-Fi and Bluetooth capabilities, making it a highly versatile solution for a wide range of applications. Developed by Espressif Systems, it is widely used in IoT devices,



*Figure 1: ESP32 (38 pins)*

wearables, home automation systems, and more. We use ESP32 as main microcontroller to control and process the sensor data.

Here's a detailed overview of the hardware features of the ESP32:

##### I. Processor and Architecture –

- ❑ **Dual-Core Processor:** The ESP32 features two Xtensa LX6 CPUs, which can operate at clock speeds up to 240 MHz.
- ❑ **32-bit Architecture:** It has a 32-bit architecture, making it capable of handling complex computing tasks.

- ❑ **Ultra-Low-Power Co-Processor:** The ESP32 integrates an Ultra-Low Power (ULP) co-processor, which allows it to perform low-power tasks while the main CPU is in deep sleep mode.

## II. Memory –

- ❑ **RAM:** The ESP32 typically comes with 520 KB of SRAM.
- ❑ **Flash Memory:** Different models of the ESP32 come with varying amounts of flash memory (typically between 4 MB and 16 MB) for storing data and code.

## III. Wireless Connectivity –

- ❑ **Wi-Fi:** The ESP32 has an integrated 2.4 GHz Wi-Fi 802.11 b/g/n module that supports WPA/WPA2 security protocols. It can operate as a station, soft access point, or both simultaneously.
- ❑ **Bluetooth:** It supports Bluetooth v4.2 and BLE (Bluetooth Low Energy) for lower-power communications and is compatible with classic Bluetooth devices.

## IV. GPIO (General-Purpose Input/Output) –

- ❑ **Number of GPIO Pins:** The ESP32 has up to 34 GPIO pins, which can be used for input, output, and various other functions such as PWM, ADC, and more.
- ❑ **Configurable:** These GPIO pins can be reconfigured to act as UART, I2C, SPI, and PWM outputs, among other functionalities.



Figure 2: ESP 32 Chip

## V. Analog Capabilities –

- ❑ **ADC (Analog-to-Digital Converter):** The ESP32 includes up to 18 ADC channels, with a 12-bit resolution. This allows it to measure analogue voltages with high precision.
- ❑ **DAC (Digital-to-Analog Converter):** It also features two 8-bit DAC channels, allowing digital signals to be converted to analogue output.

**VI. Timers and PWMs –**

- ❑ The ESP32 has hardware timers and supports PWM (Pulse Width Modulation) output, making it suitable for tasks like motor control and LED dimming.

**VII. Communication Protocols –**

- ❑ **I2C:** The ESP32 supports the I2C protocol, which allows communication with various sensors and peripherals.
- ❑ **SPI:** It also supports SPI (Serial Peripheral Interface), useful for connecting to displays, memory modules, etc.
- ❑ **UART:** Multiple UART interfaces are available for serial communication.
- ❑ **CAN Bus:** Some variants also support CAN bus for automotive and industrial applications.

**VIII. Power Management and Consumption –**

- ❑ **Operating Voltages:** The ESP32 typically operates at 3.3V but can tolerate input voltages up to 3.6V.
- ❑ **Power Modes:** It features several power-saving modes, including deep sleep and light sleep, making it ideal for battery-powered applications.

**IX. Hardware Accelerators and Security Features –**

- ❑ **Cryptographic Hardware Acceleration:** The ESP32 includes hardware accelerators for AES (Advanced Encryption Standard), RSA, SHA (Secure Hash Algorithm), and other cryptographic functions, which enhance data security and processing speeds.
- ❑ **Secure Boot and Flash Encryption:** It supports secure boot and flash encryption, which ensure the integrity and confidentiality of the code running on the device.

**X. Peripheral Interfaces –**

- ❑ **Touch Sensors:** The ESP32 has a built-in capacitive touch sensor interface.
- ❑ **SD Card Interface:** It supports interfacing with SD cards for data storage.
- ❑ **Ethernet MAC:** Some models also include an Ethernet MAC interface for wired network connectivity.

**XI. Clock and Reset Mechanisms –**

- ❑ **RTC (Real-Time Clock):** The ESP32 includes a built-in RTC module for timekeeping in low-power modes.

- ❑ **Reset and Brownout Detector:** It has a reset controller and a brownout detector to ensure reliable operation.

## XII. Integrated Antenna and PCB Layout Features –

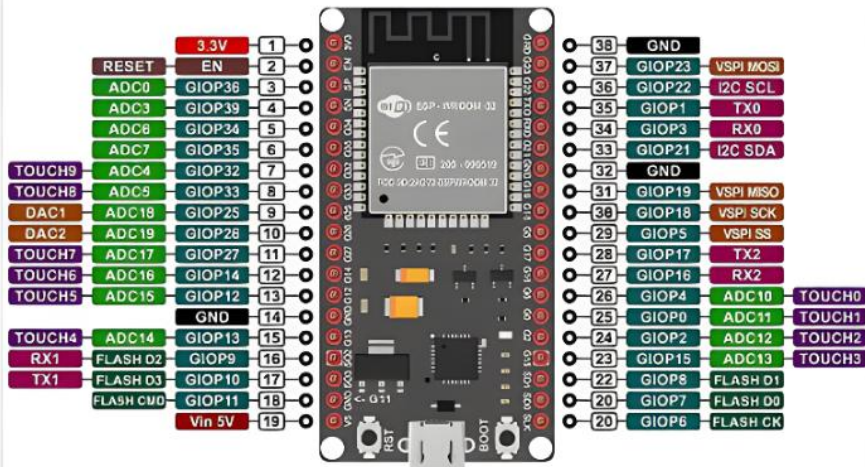
- ❑ **PCB Antenna:** Many ESP32 modules come with a PCB antenna or a U.FL connector for an external antenna, ensuring flexible connectivity options.
- ❑ **RF Shielding:** Many variants feature RF shielding to reduce noise and ensure stable wireless operation.

## XIII. Pin Descriptions –

*Table 1: ESP32 Pin Description*

Pin No.	Symbol	Label/Function	Description
1	3V3	Power Supply (3.3V)	Provides 3.3V power to the ESP32.
2	EN	Enable	Resets the ESP32 if pulled LOW.
3	GND	Ground	Ground connection for the board.
4	VIN	Input Voltage (5V)	External input voltage supply (5V recommended).
5	GPIO36	ADC0	Analog-to-Digital Converter (ADC) channel 0.
6	GPIO39	ADC3	ADC channel 3.
7	GPIO34	ADC6	ADC channel 6.
8	GPIO35	ADC7	ADC channel 7.
9	GPIO32	ADC4	ADC channel 4.
10	GPIO33	ADC5	ADC channel 5.
11	GPIO25	DAC1	Digital-to-Analog Converter (DAC) output 1.
12	GPIO26	DAC2	DAC output 2.
13	GPIO27	ADC17	ADC channel 17.
14	GPIO14	Touch Sensor	Capacitive touch input 4.
15	GPIO12	Touch Sensor	Capacitive touch input 5.



16	GPIO13	Touch Sensor	Capacitive touch input 6.
 <p>Figure 3: ESP32 Pinout</p>			
17	GPIO15	Touch Sensor	Capacitive touch input 3.
18	GPIO2	General Purpose I/O	Common GPIO.
19	GPIO0	Boot Mode Selection	Selects boot mode (Flash or UART).
20	GPIO4	ADC10	ADC channel 10.
21	GPIO16	RX2	UART2 receive pin.
22	GPIO17	TX2	UART2 transmit pin.
23	GPIO5	VSPI (SCK)	SPI Clock signal for VSPI.
24	GPIO18	VSPI (MISO)	SPI Master-In-Slave-Out for VSPI.
25	GPIO19	VSPI (MOSI)	SPI Master-Out-Slave-In for VSPI.
26	GPIO23	VSPI (CS)	SPI Chip Select for VSPI.
27	GPIO22	I2C SCL	I2C Clock Line.
28	GPIO21	I2C SDA	I2C Data Line.
29	TX0	UART0 TX	Default UART Transmit pin.
30	RX0	UART0 RX	Default UART Receive pin.

31	GPIO3	UART RX0	UART RX used as a GPIO.
32	GPIO1	UART TX0	UART TX used as a GPIO.
33	GPIO9	Flash IO2	Used for SPI flash memory.
34	GPIO10	Flash IO3	Used for SPI flash memory.
35	GPIO11	Flash CMD	Flash Command line.
36	GPIO6	Flash CLK	Flash Clock line.
37	GPIO7	Flash IO0	SPI Flash I/O pin.
38	GPIO8	Flash IO1	SPI Flash I/O pin.

#### XIV. Advantages of ESP32 –

- 1) **Integrated Wi-Fi and Bluetooth:** The ESP32 features built-in Wi-Fi and Bluetooth (classic and BLE) capabilities, making it highly suitable for wireless connectivity applications without needing additional modules.
- 2) **High Performance:** The ESP32 is a dual-core or single-core (depending on the variant) microcontroller with a clock speed of up to 240 MHz, providing considerable processing power for a wide range of applications.
- 3) **Low Power Consumption:** It has multiple power-saving modes (deep sleep, light sleep, etc.), making it ideal for battery-operated IoT devices that need to optimize energy usage.
- 4) **Rich Peripheral Support:** The ESP32 supports a variety of peripherals, including ADC (Analog-to-Digital Converter), DAC (Digital-to-Analog Converter), PWM (Pulse Width Modulation), I2C, SPI, UART, CAN bus, touch sensors, and more.
- 5) **Large Memory Capacity:** It features more RAM and flash memory compared to many other microcontrollers, enabling more complex applications and larger codebases.

- 6) **Support for RTOS (Real-Time Operating System):** The ESP32 supports FreeRTOS, making it suitable for real-time and complex multitasking applications.
- 7) **Open-Source Development Environment:** It can be programmed using the Arduino IDE, Platform IO, Espressif's ESP-IDF, Micro Python, and other environments, providing flexibility for various developer preferences.
- 8) **Cost-Effective:** Despite its extensive features, the ESP32 is relatively low-cost, making it an excellent option for hobbyists and professionals alike.
- 9) **Wide Community Support:** The ESP32 benefits from an active community, with ample resources, libraries, forums, and tutorials available online to assist in development.

#### **XV. Limitations of ESP32 –**

- 1) **Complexity for Beginners:** With its numerous features and peripherals, the ESP32 may have a steeper learning curve for beginners compared to simpler microcontrollers like the Arduino Uno.
- 2) **Power Consumption:** Although the ESP32 has low-power modes, its typical active power consumption is higher compared to simpler microcontrollers, which might not be suitable for ultra-low-power applications.
- 3) **Limited I/O Pins:** While it has many GPIO pins, some are shared with onboard functionalities (like the internal flash memory), limiting their usage in specific scenarios. Careful planning is required to maximize their availability.
- 4) **Library and Compatibility Issues:** While many libraries are available, some are designed for specific platforms (like Arduino) and may not function optimally or may require adaptation when used with ESP32.
- 5) **ADC Non-Linearity:** The ESP32's built-in ADC is known to have non-linear characteristics and may require calibration for accurate measurements. This can be a challenge in applications requiring precise analogue inputs.

- 6) **Limited Real-Time Capabilities Compared to Dedicated RTOS Microcontrollers:** Though it supports RTOS, it may not be as performant in strict real-time applications as some specialized real-time microcontrollers.
- 7) **Wi-Fi Connection Issues:** In some cases, the ESP32 can experience issues with Wi-Fi connectivity stability, especially in environments with heavy RF interference.
- 8) **Overhead Due to Dual-Core Architecture:** For simpler applications, the overhead and complexity introduced by dual-core processing may not always be beneficial and could lead to higher power usage without the added performance being fully utilized.

## 2. Alcohol Sensor ( MQ3 ) –

The MQ-3 alcohol sensor is a widely used gas sensor module that is

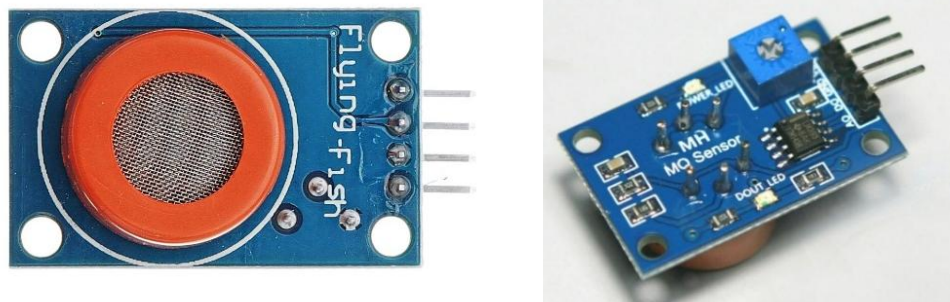


Figure 4: MQ3 Module

specifically designed to detect alcohol vapours in the air. It is part of the MQ gas sensor family and is particularly sensitive to alcohol, making it popular for use in Breathalyzer applications, alcohol detection systems, and other similar projects. Here is a detailed breakdown of its features, working principle, specifications, and typical applications:

### I. Features of MQ-3 Alcohol Sensor –

- ❑ **High Sensitivity to Alcohol:** The sensor is very sensitive to alcohol vapours, making it suitable for alcohol concentration detection.
- ❑ **Fast Response Time:** It provides relatively quick readings, making it efficient for real-time applications.
- ❑ **Wide Detection Range:** MQ-3 is capable of detecting alcohol concentrations from 0.04 mg/L to 4 mg/L in the air.

- ❑ **Analog Output:** It provides an analogue voltage output that corresponds to the concentration of alcohol in the air.
- ❑ **Long Lifespan and Stability:** The sensor is designed to provide stable readings over a long period.

## II. Specifications –

- ❑ **Operating Voltage:** Typically, 5V DC
- ❑ **Output Signal:** Analog voltage
- ❑ **Detection Range:** 0.04 mg/L to 4 mg/L
- ❑ **Preheating Time:** 20 seconds (to stabilize the sensor)
- ❑ **Heater Consumption:** Approximately 0.5W



Figure 5: MQ3 Sensor

## III. Working Principle –

The MQ-3 alcohol sensor works on the principle of **metal-oxide-semiconductor (MOS) gas sensing**. Here's how it works in detail:

- ❑ **Heater Element:** The sensor contains a small heating element that heats the internal sensing material, usually made of a tin dioxide ( $\text{SnO}_2$ ) semiconductor layer. The heating allows the sensor to become sensitive to the gas around it by increasing the activity of the sensor material.

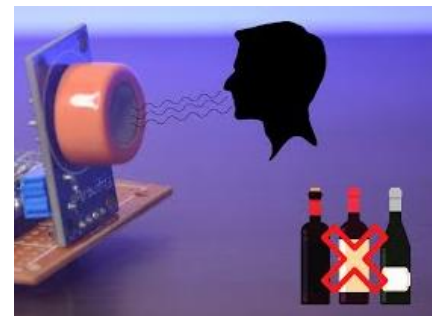


Figure 6: MQ3 Alcohol Sensor Working

- ❑ **Alcohol Absorption:** When the sensor is exposed to alcohol vapours, the alcohol molecules come in contact with the heated sensing element. This causes a change in the conductivity of the sensing element.
- ❑ **Change in Resistance:** Alcohol reduces the sensor's resistance. The level of change is directly related to the concentration of alcohol present in the air.
- ❑ **Signal Output:** This change in resistance is measured as an analogue output voltage, which can then be processed by microcontrollers like an Arduino or ESP32 to determine the alcohol concentration.

## IV. Usage –

- ❑ **Circuit Connections:**

↳ **VCC Pin:** Connects to the power supply (usually 5V).

✍ **GND Pin:** Connects to the ground.

✍ **Analog Output (AOUT):** Outputs a voltage that varies with alcohol concentration.

✍ **Digital Output (DOUT):** The sensor may also have a digital output pin that can be used with a comparator to trigger an alarm at a set threshold.

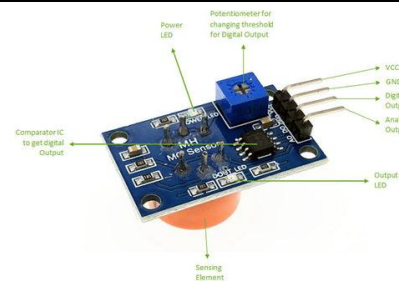


Figure 7: ICs and Pinout of MQ3 Alcohol Sensor

❑ **Calibration:** The analogue output needs to be calibrated for precise measurements. This involves setting a reference point for known alcohol concentrations.

#### V. Advantages –

- 1) **Affordable and Readily Available:** It is relatively low-cost and easy to procure.
- 2) **Easy to Interface:** Simple interfacing with microcontrollers due to its analogue and digital output.

#### VI. Limitations –

- 1) **Preheating Required:** The sensor requires a short preheat period for accurate readings.
- 2) **Cross-Sensitivity:** It can also detect other gases to a lesser extent, such as benzene, methane, etc., which might introduce inaccuracies.

The MQ-3 sensor is a versatile and widely-used alcohol sensor, suitable for both hobbyist and professional projects that require alcohol detection.

### 3. Ultrasonic Sensor (HC-SR04) –

The **HC-SR04 Ultrasonic Sensor** is a versatile and widely used sensor for proximity and distance measurement. It operates by emitting ultrasonic sound waves and measuring the time it takes for the echoes to return after hitting an object. This enables precise



Figure 8: Ultrasonic Sensor (HC-SR04)

calculation of the distance between the sensor and the object.

## I. Principle of Operation –

### 1) Ultrasonic Waves:

- a) The sensor operates at a frequency of **40 kHz**, which is above human hearing range.
- b) A sound wave is emitted from the **transmitter**, and the sensor waits to receive the reflected wave using the **receiver**.

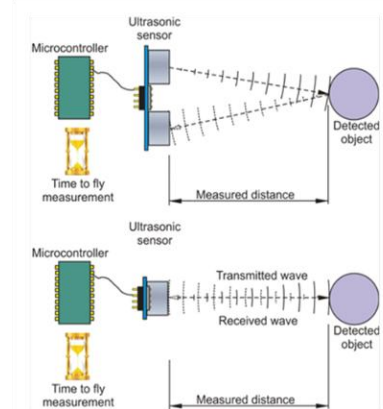


Figure 9: Working Principle Of Ultrasonic Sensor (HC-SR04)

### 2) Time-of-Flight Measurement:

- a) The sensor measures the time interval between the emission of the sound wave and the detection of its echo.
- b) Using the known speed of sound in air (approximately **343 m/s** under standard conditions), the distance  $d$  can be calculated using the formula:

$$d = \frac{v \times t}{2}$$

where:

- ↳  $V$  : speed of sound in air,
- ↳  $t$  : time elapsed between sending and receiving the signal,
- ↳ Division by 2 accounts for the round trip of the wave.

### 3) Measurement Process:

- a) Send a HIGH signal to the TRIG pin for at least 10  $\mu$ s.
- b) The sensor emits an eight-cycle ultrasonic burst at 40 kHz.
- c) The ECHO pin goes HIGH until the reflected wave returns.
- d) Measure the duration  $t$  of the ECHO signal and calculate the distance.

### 4) Mitigation Techniques:

- a) Implement error-handling algorithms.

- b) Use shielding or directed sensing to limit unwanted reflections.
- c) Employ temperature compensation to adjust for environmental changes.

## II. HC-SR04 Sensor Specifications –

The HC-SR04 is a compact, cost-effective sensor with the following features:

- 1) **Power Supply:** 5V DC
- 2) **Current Consumption:**
  - ↳ Quiescent: <2 mA
  - ↳ Operating: ~15 mA
- 3) **Operating Range:** 2 cm to 400 cm
- 4) **Resolution:** 0.3 cm
- 5) **Effectual Angle:** <15°
- 6) **Working Temperature:** -20°C to 70°C

The small size and weight (approximately 10g) make it ideal for embedded systems and applications in robotics and safety devices.

## III. Hardware and Pin Configuration

The sensor includes four pins:

- 1) **VCC:** Supplies +5V DC.
- 2) **GND:** Ground connection.
- 3) **TRIG:** Trigger input pin, where a HIGH signal initiates measurement.
- 4) **ECHO:** Output pin, which goes HIGH for the duration the sound wave is traveling.

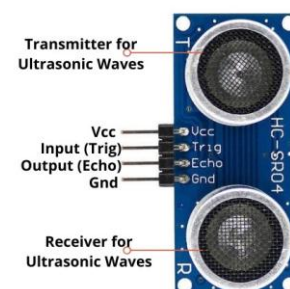


Figure 10: Ultrasonic Sensor (HC-SR04) Pinout

## IV. Sources of Error and Accuracy Challenges

Although the HC-SR04 is reliable, certain factors can lead to inaccuracies:



- 1) **Environmental Conditions:** Variations in temperature, humidity, and pressure affect the speed of sound.
- 2) **Object Surface and Angle:** The sensor may fail to detect irregular or non-perpendicular surfaces as the sound wave may not return.
- 3) **Interference:** Noise from other ultrasonic sensors or environmental sources can cause false readings.
- 4) **Echo Clutter:** In confined spaces, multiple reflections may result in incorrect measurements.

#### V. Merits

- a) **Affordable** and cost-effective.
- b) **Easy to use** with microcontrollers.
- c) **High accuracy** (0.3 cm resolution).
- d) Wide **measurement range** (2–400 cm).
- e) **Low power consumption** and compact size.
- f) Works in **non-contact** and challenging conditions (e.g., smoke, dust).

#### VI. Demerits –

- a) **Environmental sensitivity** (affected by temperature, humidity, and pressure).
- b) Limited **detection angle** ( $<15^\circ$ ).
- c) Struggles with **irregular or soft surfaces**.
- d) **Slower response** for high-speed applications.
- e) Susceptible to **interference** and echo clutter.
- f) Not suitable for **long-range** applications ( $>4$  meters).

#### 4. GSM Module (SIM900A) –

The **SIM900A** is a GSM/GPRS module designed to facilitate seamless communication in embedded applications. It supports quad-band GSM frequencies, including **900/1800 MHz**, making it suitable for operations in regions using these frequency bands.



*Figure 11: GSM Module (SIM900A)*

##### I. Features –

##### 1) Compact Size:

- ✦ Dimensions: 24mm × 24mm × 3mm.
- ✦ Easy to integrate into space-constrained designs.

##### 2) GSM/GPRS Compatibility:

- ✦ Supports GSM voice calls, SMS messaging, and GPRS data transfer.
- ✦ Compliant with 3GPP TS 27.007, 27.005, and SIM application toolkit.

##### 3) Low Power Consumption:

- ✦ Operates efficiently with low energy requirements, ideal for battery-powered systems.

##### 4) Integrated TCP/IP Stack:

- ✦ Built-in stack allows internet communication without the need for external microcontrollers to handle networking protocols.

##### 5) Interfaces:

- ✦ UART with adjustable baud rate (up to 115200bps).
- ✦ SIM card interface (1.8V/3V SIM card support).
- ✦ GPIOs, ADC, and keypad interface for enhanced versatility.

#### 6) Antenna Support:

- ↳ External antenna connection through an SMA connector ensures strong signal reception.

#### 7) Network Support:

- ↳ Robust against network issues with features like call-ready detection and network registration status.

### II. Functional Diagram

The module's functional blocks include:

- ↳ **Power Supply:** Requires 3.4V to 4.5V (optimal 4.0V).
- ↳ **RF Section:** Handles GSM signal transmission and reception.
- ↳ **Baseband Section:** Processes digital signals for GSM/GPRS.
- ↳ **SIM Card Interface:** Communicates with the SIM card.

### III. Hardware Connection

To use the SIM900A module:

#### 1. Power Supply:

- ↳ Ensure a stable power source (regulated 4V supply).
- ↳ Connect VCC and GND appropriately.

#### 2. UART Communication:

- ↳ Connect the RXD and TXD pins to the microcontroller.
- ↳ Use a 3.3V or 5V level shifter if necessary.

#### 3. SIM Card:

- ↳ Insert a valid GSM SIM card into the slot.
- ↳ Ensure proper network availability.

#### 4. Antenna:

- ↳ Attach the external antenna for improved signal reception.

### 5. Testing Commands:

↳ Use AT commands like AT (basic module check), AT+CMGS (send SMS), and AT+CGATT (GPRS attachment) for testing.

### IV. Advantages of SIM900A

- 1) **Cost-Effective:** Affordable for a wide range of applications.
- 2) **Ease of Use:** Simplified AT commands make it beginner-friendly.
- 3) **Reliable Performance:** Designed for stable and consistent communication.

### V. Pin Name & Description –

- 1) **VCC:** Power supply input pin. Accepts 3.4V to 4.5V (recommended 4.0V).
- 2) **GND:** Ground pin. Connect to the ground of the power supply.
- 3) **RXD:** UART Receive pin. Used to receive data from the microcontroller (logic level: 3.3V).
- 4) **TXD:** UART Transmit pin. Used to send data to the microcontroller (logic level: 3.3V).
- 5) **DTR:** Data Terminal Ready. Can be used to put the module into sleep mode when idle.
- 6) **RI:** Ring Indicator. Provides an interrupt signal when a call or SMS is received.
- 7) **DCD:** Data Carrier Detect. Indicates whether a data connection is active.
- 8) **CTS:** Clear To Send. Used in UART communication for hardware flow control (logic level: 3.3V).
- 9) **RTS:** Ready To Send. Used in UART communication for hardware flow control (logic level: 3.3V).
- 10) **SIM\_VDD:** Provides power supply to the SIM card (1.8V/3V).
- 11) **SIM\_DATA:** Bidirectional data line for SIM card communication.
- 12) **SIM\_CLK:** Clock signal for the SIM card.
- 13) **SIM\_RST:** Reset signal for the SIM card.
- 14) **AUX\_ANT:** Optional auxiliary antenna connection.
- 15) **ANT:** Main antenna pin for GSM signal reception.

- 16) ADC:** Analog-to-digital converter input. Can measure analogue signals (e.g., voltage).
- 17) SPK\_P/SPK\_N:** Speaker positive and negative terminals for audio output (used in voice call applications).
- 18) MIC\_P/MIC\_N:** Microphone positive and negative terminals for audio input (used in voice call applications).
- 19) RESET:** Hardware reset pin. A low pulse on this pin resets the module.
- 20) STATUS:** Indicates the working status of the module. Can be used to check if the module is powered and operational.

This explanation provides a comprehensive understanding of the SIM900A GSM module, highlighting its functionality, setup, and advantage.

## 5. GPS Module (NEO-6M) –

The **Neo-6M** is a GPS receiver module designed by u-blox, widely used for obtaining location, speed, and time data. It is compact, low-power, and high-performance, making it ideal for embedded systems, robotics, and IoT projects.



Figure 12: GPS Module (NEO-6M)

### I. Features

#### ❑ High Sensitivity:

- ↳ Supports up to **-161 dBm sensitivity**, enabling it to work under challenging conditions, such as urban canyons or indoors.

#### ❑ Positioning:

- ↳ Provides highly accurate location data (up to **2.5 meters CEP** in open sky conditions).

#### ❑ Power Efficiency:

- ↳ Designed with low power consumption, supporting modes like **Power Save Mode (PSM)** for increased efficiency.



Figure 13: GPS Module (NEO-6M) Chip

### ❑ Satellite Compatibility:

- ↳ Supports GPS, SBAS (WAAS, EGNOS, MSAS, and GAGAN), and QZSS systems for improved reliability.

### ❑ Serial Communication:

- ↳ Communicates using UART (Universal Asynchronous Receiver/Transmitter) protocol at default baud rates (usually **9600 bps**, but configurable).

### ❑ Cold, Warm, and Hot Starts:

- ↳ Cold start: <27 seconds.
- ↳ Warm start: <25 seconds.
- ↳ Hot start: <1 second.

### ❑ Data Output:

- ↳ Provides NMEA (National Marine Electronics Association) sentences for position, velocity, and time.

### ❑ Clock:

- ↳ Comes with a built-in RTC (Real-Time Clock) for accurate timing even during standby mode.

13	GND	GND	12
14	MOSI/CFG_COM0	RF_IN	11
15	MISO/CFG_COM1	GND	10
16	CFG_GPS0/SCK	VCC_RF	9
17	Reserved	Reserved	8
<b>NEO-6</b>			
<b>Top View</b>			
18	SDA2	VDDUSB	7
19	SCL2	USB_DP	6
20	TxD1	USB_DM	5
21	RxD1	EXTINT0	4
22	V_BCKP	TIMEPULSE	3
23	VCC	SS_N	2
24	GND	Reserved	1

uBlox NEO-6 Pinout

Figure 14: GPS Module (NEO-6M) Chip Pinout

## II. Technical Specifications

### ❑ Power Supply:

- ↳ Operates at **3V-5V DC**.

### ❑ Current Consumption:

- ↳ Typically around **45mA** during active operation.

### ❑ Dimensions:

- ↳ Compact module with dimensions of around **16mm x 12mm** (without antenna).

### ❑ Antenna:

- ↳ Includes an external active ceramic antenna (patch antenna) for better signal reception.

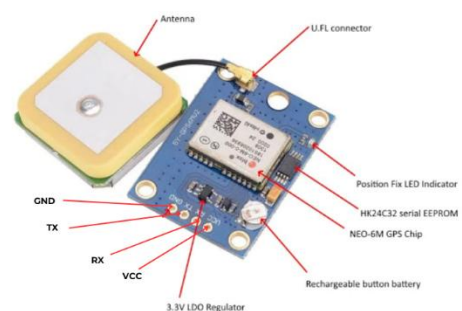


Figure 15: GPS Module (NEO-6M) components

#### ❑ **Connector:**

- ↳ Includes a **4-pin header** for easy interfacing with microcontrollers and development boards like Arduino, Raspberry Pi, or ESP8266.

#### ❑ **Memory:**

- ↳ Features a built-in **EEPROM** to save configuration settings.

### III. Pin Descriptions

- ❑ **VCC:** Power input (3.3V or 5V DC).

- ❑ **GND:** Ground.

- ❑ **TX:** Transmits serial data to the microcontroller.

- ❑ **RX:** Receives serial data from the microcontroller.



Figure 16: GPS Module (NEO-6M) Pinout

### IV. Advantages

#### ❑ **High Sensitivity:**

- ↳ Excellent signal reception (up to **-161 dBm**) ensures performance even in challenging environments like urban areas or partially obstructed regions.

#### ❑ **Compact Size:**

- ↳ Small footprint makes it ideal for space-constrained applications like drones, IoT devices, and portable trackers.

#### ❑ **Low Power Consumption:**

- ↳ Consumes around **45 mA**, making it suitable for battery-powered applications.

#### ❑ **Reliable GPS Performance:**

- ↳ Supports various satellite-based augmentation systems (SBAS) for enhanced accuracy (WAAS, EGNOS, MSAS, GAGAN).

#### ❑ **Ease of Integration:**

- ↳ Simple UART interface, making it compatible with microcontrollers like Arduino, Raspberry Pi, and others.

#### ❑ **External Antenna Support:**

- ↳ Includes an active ceramic antenna that improves signal reception and positioning accuracy.

❑ **EEPROM Storage:**

- ↳ Saves configuration data, reducing the need to reconfigure settings on power loss.

❑ **Wide Range of Applications:**

- ↳ Useful in tracking, geofencing, navigation, and time synchronization.

❑ **Real-Time Data:**

- ↳ Outputs NMEA sentences, providing detailed information like location, velocity, and time.

❑ **Multiple Modes:**

- ↳ Supports power-saving modes to optimize energy consumption based on application needs.

## V. Disadvantages

❑ **Dependency on Open Sky:**

- ↳ Performance drops significantly in obstructed environments (e.g., dense forests, underground, or inside buildings).

❑ **Signal Interference:**

- ↳ Prone to interference from nearby electronics or metal enclosures, which can reduce accuracy or disrupt the signal.

❑ **Limited Accuracy:**

- ↳ While adequate for general applications, its accuracy (around **2.5 meters CEP**) is insufficient for precision-critical tasks.

❑ **Slow Cold Start:**

- ↳ Initial satellite acquisition (cold start) can take up to **27 seconds**, especially in poor signal conditions.

❑ **No Built-In IMU:**

- ↳ Unlike advanced GPS modules, it lacks an inertial measurement unit (IMU) for dead-reckoning or handling signal dropouts.

❑ **Antenna Placement:**

- ↳ External antenna requires careful placement for optimal performance, increasing design complexity.

❑ **Power Requirement:**

- ↳ Although low-power, continuous operation at **3.3V-5V and 45mA** may be a challenge for long-term battery-powered devices.



❑ **Basic Functionality:**

- ✦ No additional features like onboard processing, geofencing alerts, or integrated GSM/GPRS for communication.

❑ **Configuration Complexity:**

- ✦ Custom configurations (e.g., baud rates, update rates) require familiarity with u-blox tools and protocols, which may be challenging for beginners.

❑ **No Built-In Backup Power:**

- ✦ It relies on external power for the real-time clock (RTC) and memory, which means settings can be lost without a backup battery.

## 6. MPU6050 –

The **MPU6050** is a 6-DOF (Degrees of Freedom) IMU sensor, manufactured by InvenSense. It combines a 3-axis gyroscope and a 3-axis accelerometer, with a built-in **Digital Motion Processor (DMP)** for onboard motion processing. It communicates with external devices via the **I2C (Inter-Integrated Circuit)** protocol.



Figure 17: MPU6050

### I. Features –

❑ **Integrated 6-DOF Motion Sensing:**

- ✦ Combines 3-axis gyroscope and 3-axis accelerometer.

❑ **Digital Motion Processor (DMP):**

- ✦ Offloads computation by processing motion algorithms internally, reducing the burden on external microcontrollers.

❑ **High Precision:**

- ✦ Provides accurate angular velocity (gyroscope) and linear acceleration (accelerometer) measurements.

❑ **Wide Measurement Range:**

- ✦ Accelerometer:  $\pm 2g$ ,  $\pm 4g$ ,  $\pm 8g$ ,  $\pm 16g$ .
- ✦ Gyroscope:  $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$ ,  $\pm 2000$  °/s.

### ❑ Communication Interface:

- ↳ Supports I2C protocol (up to 400 kHz).

### ❑ Auxiliary Sensor Support:

- ↳ Provides additional input pins for connecting external sensors, such as magnetometers.

### ❑ Low Power Consumption:

- ↳ Configurable power modes to minimize energy usage.

### ❑ Interrupt Support:

- ↳ Configurable interrupt pin for event-based triggers, such as motion detection or data-ready signals.

### ❑ Temperature Sensor:

- ↳ Built-in temperature sensor for environmental monitoring or calibration.

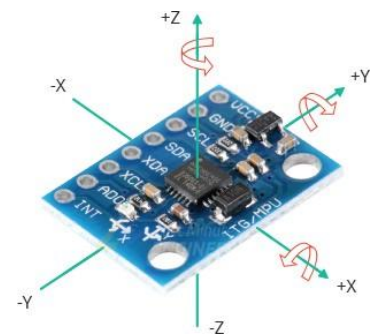


Figure 18: Orientation of MPU6050

## II. Specifications –

### ❑ Power Supply:

- ↳ Operating Voltage: **2.3V to 3.46V** (Typically powered at 3.3V).

### ❑ Current Consumption:

- ↳ Accelerometer: **500  $\mu$ A**.
- ↳ Gyroscope: **3.6 mA**.

### ❑ Sensitivity:

- ↳ Accelerometer: 16384 LSB/g ( $\pm 2g$  setting).
- ↳ Gyroscope: 131 LSB/ $^{\circ}$ /s ( $\pm 250$   $^{\circ}$ /s setting).

### ❑ Output:

- ↳ Provides 16-bit digital output for accelerometer, gyroscope, and temperature data.

### ❑ Operating Temperature:

- ↳  $-40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$ .

### ❑ Dimensions:

- ↳ Typically, **20mm x 16mm**, including breakout board.

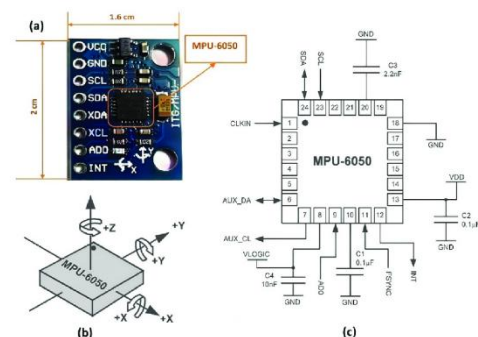


Figure 19: MPU6050 Chip

### III. Pin Description –

❑ **VCC:**

↪ Power input (3.3V typical).

❑ **GND:**

↪ Ground.

❑ **SCL (Serial Clock):**

↪ I2C clock input.

❑ **SDA (Serial Data):**

↪ I2C data line.

❑ **XDA/XCL:**

↪ Auxiliary I2C pins for connecting external sensors.

❑ **ADO (I2C Address Select):**

↪ Used to set the I2C address of the module (0x68 or 0x69).

❑ **INT (Interrupt):**

↪ Outputs interrupt signals for events such as data ready or motion detection.

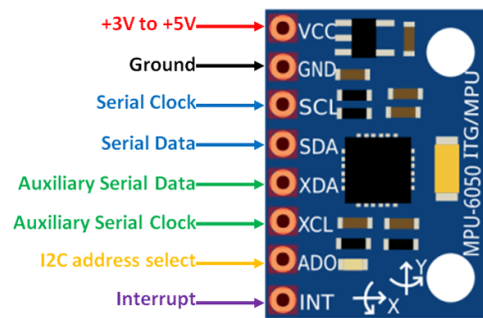


Figure 20: MPU6050 Pinout

### IV. Advantages –

❑ **Compact Design:**

↪ Combines gyroscope and accelerometer on a single chip.

❑ **Ease of Integration:**

↪ I2C communication ensures simple interfacing with microcontrollers.

❑ **Real-Time Processing:**

↪ DMP allows motion data to be processed internally for real-time applications.

❑ **Cost-Effective:**

↪ Affordable compared to other IMU modules with similar features.

❑ **Versatility:**

↪ Suitable for a wide range of applications, including robotics, drones, and wearables.

### V. Disadvantages –

❑ **No Magnetometer:**

↪ Lacks a 3-axis magnetometer, which would provide a complete 9-DOF solution.

❑ **Limited Power Supply Range:**

- ✎ Operates at lower voltages, requiring voltage level conversion for 5V systems.

❑ **Calibration Required:**

- ✎ Requires manual calibration for optimal accuracy, as environmental factors can cause drift.

❑ **Susceptible to Noise:**

- ✎ Requires filtering (e.g., complementary or Kalman filter) for stable output.

## 7. 1.3-inch OLED Display Module –

The **1.3-inch OLED Display Module** uses **Organic Light Emitting Diode (OLED)** technology, offering bright, high-contrast visuals without the need for a backlight. It typically operates using the **SSD1306** or **SH1106** driver IC, supports I2C communication, and is suitable for various projects like IoT dashboards, small devices, and DIY electronics.



Figure 21: 1.3-inch OLED Display Module

### I. Features –

❑ **OLED Technology:**

- ✎ Each pixel emits light, ensuring vibrant displays with no need for a backlight.
- ✎ Provides better visibility in low-light and bright environments.

❑ **Compact Size:**

- ✎ 1.3-inch diagonal display with a resolution of **128x64 pixels**.

❑ **Driver IC:**

- ✎ Generally, uses **SSD1306** or **SH1106** driver for display management.

❑ **Communication Interface:**

- ✎ Uses **I2C (Inter-Integrated Circuit)** communication for simple interfacing with microcontrollers.

❑ **Low Power Consumption:**

- ↳ Efficient operation, consuming minimal power in comparison to LCDs.

❑ **Wide Viewing Angle:**

- ↳ Up to **160°**, ensuring visibility from various perspectives.

❑ **Monochrome Display:**

- ↳ Typically, available in colours like white, yellow, or blue.

❑ **Operating Voltage:**

- ↳ Works at **3.3V or 5V**, making it compatible with most development boards like Arduino, Raspberry Pi, and ESP series.

## II. Specifications –

❑ **Resolution:**

- ↳ 128x64 pixels.

❑ **Screen Dimensions:**

- ↳ Active Area: **29.42mm x 14.7mm** (approx.).
- ↳ Module Size: **35mm x 35mm** (varies by breakout board).

❑ **Interface:**

- ↳ I2C communication protocol.
- ↳ Default I2C address: **0x3C** or **0x3D**, configurable via solder jumpers.

❑ **Voltage:**

- ↳ Power Supply: **3.3V - 5V DC**.
- ↳ Logic Level: Compatible with both 3.3V and 5V systems.

❑ **Operating Temperature:**

- ↳ Typically **-40°C to 85°C**.

❑ **Power Consumption:**

- ↳ Extremely low; typically, less than **10mA**.

❑ **Viewing Angle:**

- ↳ Approximately **160°**.

❑ **Driver IC:**

- ↳ SSD1306 or SH1106.

## III. Pin Description –

❑ **VCC:** Power input (3.3V or 5V DC).

❑ **GND:** Ground connection.

❑ **SCL:** Serial Clock Line for I2C communication.

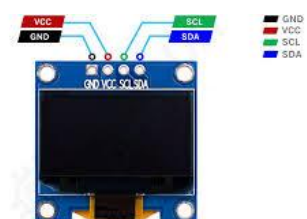


Figure 22: 1.3-inch OLED Display Module Pinout

- ❑ **SDA:** Serial Data Line for I2C communication.

#### IV. Advantages –

- ❑ **High Contrast and Brightness:** Provides clear visuals even in bright light conditions.
- ❑ **Wide Compatibility:** Easily interfaces with microcontrollers (e.g., Arduino, ESP32, STM32) using I2C libraries.
- ❑ **Low Power:** Ideal for battery-powered projects due to its minimal power consumption.
- ❑ **Compact and Lightweight:** Saves space in compact designs like wearable devices or small IoT products.
- ❑ **Wide Viewing Angles:** Makes it easier to read from any position.
- ❑ **High Refresh Rate:** Smooth animations and quick updates, suitable for dynamic visualizations.

#### V. Disadvantages –

- ❑ **Monochrome Only:**
  - ↳ Supports a single colour (e.g., white, blue, or yellow), which limits its use for colour-intensive applications.
- ❑ **Limited Pixel Density:**
  - ↳ The resolution is suitable for small applications but not ideal for detailed images or high-resolution content.
- ❑ **Fragile Display:**
  - ↳ OLED screens are more sensitive to physical damage compared to LCDs.
- ❑ **Image Burn-In:**
  - ↳ Prolonged display of static images can cause burn-in or pixel degradation over time.
- ❑ **Limited I2C Address Range:**
  - ↳ Only two possible I2C addresses (e.g., 0x3C or 0x3D), which may conflict with other I2C devices.

## 8. L293D Motor Driver Module –

The **L293D** is a high-voltage, high-current dual full-bridge motor driver integrated circuit. It is widely used for driving **DC motors**, **stepper motors**, or **actuators** in robotics and embedded systems. The module operates on the principle of an H-bridge, enabling bidirectional control of motors.



Figure 23: L293D Motor Driver Module

The module typically features additional components (like diodes, capacitors, and headers) to simplify motor control tasks.

### I. Features –

#### ❑ Dual H-Bridge Design:

↳ Can independently control two DC motors or a single stepper motor.

#### ❑ Voltage and Current Handling:

↳ Motor Supply Voltage: **4.5V to 46V**.

↳ Logic Voltage: **5V**.

↳ Maximum Output Current: **2A per channel (continuous)** and **4A peak**.

#### ❑ Built-In Protection:

↳ Integrated flyback diodes to protect against back-EMF (electromotive force).

#### ❑ PWM Speed Control:

↳ Supports Pulse Width Modulation (PWM) for speed control.

#### ❑ Thermal Shutdown:

↳ Protection against overheating.

#### ❑ Logic Compatibility:

↳ Compatible with 5V logic levels, suitable for microcontrollers like Arduino, ESP32, Raspberry Pi, etc.

#### ❑ Compact Design:

↳ Simplifies motor driving in embedded systems.

### II. Specifications –

❑ **Motor Supply Voltage ( $V_{ss}$ ):** 4.5V to 46V.

❑ **Logic Supply Voltage ( $V_{cc}$ ):** Typically, 5V.

- ❑ **Output Current:** 2A per channel (continuous), 4A peak.
- ❑ **Power Dissipation:** 25W (with proper heat sink).
- ❑ **Operating Temperature:** -25°C to +130°C.
- ❑ **Driver Configuration:** Two channels (A and B) for independent motor control.
- ❑ **Dimensions:** Varies by module, typically 60mm x 45mm.

### III. Pin Description (Typical Module) –

- ❑ **VCC (5V):** Logic voltage input (5V).
- ❑ **GND:** Ground connection.
- ❑ **12V (Vss):** Motor power supply (4.5V to 46V).
- ❑ **OUT1, OUT2:** Motor A output pins.
- ❑ **OUT3, OUT4:** Motor B output pins.
- ❑ **ENA, ENB:** Enable pins for motors A and B, respectively (high to enable, low to disable).
- ❑ **IN1, IN2:** Input pins for controlling Motor A's direction.
- ❑ **IN3, IN4:** Input pins for controlling Motor B's direction.

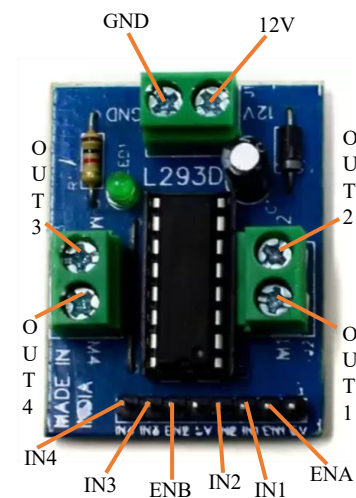


Figure 24: L293D Motor Driver Module Pinout

### IV. Working Principle –

The **L293D module** uses an H-bridge circuit to control the direction and speed of the motors. It works as follows:

#### ❑ Direction Control:

➤ The direction of the motor is determined by the input pins (IN1/IN2 for Motor A, IN3/IN4 for Motor B):

1. Setting IN1 high and IN2 low causes the motor to rotate forward.
2. Setting IN1 low and IN2 high causes the motor to rotate in reverse.

#### ❑ Speed Control:

➤ Speed is controlled using **PWM (Pulse Width Modulation)** signals on the ENA and ENB pins.



1. A higher PWM duty cycle increases motor speed.

❑ **Dual Motor Control:**

↳ The two H-bridges allow independent control of two DC motors, including forward/reverse rotation and speed.

❑ **Stepper Motor Control:**

↳ Both H-bridges can be combined to drive a single stepper motor.

**V. Advantages –**

❑ **Bidirectional Control:**

↳ Easily drives motors in both forward and reverse directions.

❑ **High Voltage Range:**

↳ Can handle a wide range of motor voltages (up to 46V).

❑ **Integrated Flyback Diodes:**

↳ Protects against back-EMF, improving module reliability.

❑ **Independent Motor Control:**

↳ Drives two motors simultaneously with separate control.

❑ **Thermal Protection:**

↳ Protects the IC from overheating.

**VI. Disadvantages –**

❑ **Current Limitation:**

↳ Maximum output current is **2A per channel**, which is insufficient for larger motors.

❑ **Heat Dissipation:**

↳ Generates significant heat during operation, requiring heat sinks or active cooling for higher loads.

❑ **Efficiency:**

↳ Lacks efficiency compared to newer motor drivers (e.g., MOSFET-based drivers like L298N).

❑ **Limited Features:**

↳ Does not support advanced features like current sensing or overcurrent protection.

## 9. TP4056 Module –

The TP4056 module is a compact charging circuit designed for single-cell lithium-ion batteries. It features a TP4056 linear charging IC and integrated protection circuit (DW01 and MOSFETs). This module provides a safe and efficient way to charge Li-Ion or Li-Po batteries from a 5V source, such as a USB port or a DC adapter.



Figure 25: TP4056 Module

### I. Key Components –

#### ❑ TP4056 IC:

- ✦ Core of the module, handles the charging process.
- ✦ Implements Constant Current/Constant Voltage (CC/CV) charging.
- ✦ Has built-in safety features for overvoltage, overcurrent, and thermal regulation.

#### ❑ DW01 Protection IC:

- ✦ Provides battery protection.
- ✦ Prevents overcharge (cut-off at ~4.2V).
- ✦ Prevents over-discharge (cut-off at ~2.4V).
- ✦ Includes short-circuit and overcurrent protection.

#### ❑ Dual MOSFETs:

- ✦ Work with the DW01 IC to disconnect the battery during fault conditions (overcharge/discharge).

#### ❑ Micro-USB Connector:

- ✦ Provides a convenient 5V input for charging.

#### ❑ LED Indicators:

- ✦ **Red LED (CHRG):** Charging in progress.
- ✦ **Blue/Green LED (DONE):** Charging complete



Figure 26: TP4056 Module ICs and Component

### II. Pin Descriptions –

- ❑ **B+:** Connects to the positive terminal of the battery.
- ❑ **B-:** Connects to the negative terminal of the battery.
- ❑ **OUT+:** Connects to the positive terminal of the load (device powered by the battery).

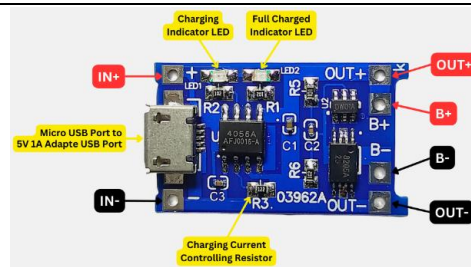


Figure 27: TP4056 Module Pinout

- ❑ **OUT-:** Connects to the negative terminal of the load.
- ❑ **IN+:** Input voltage positive terminal (5V from USB or external source).
- ❑ **IN-:** Input voltage negative terminal (common ground).

### III. Charging Algorithm –

The TP4056 module charges lithium-ion batteries using the **CC/CV** algorithm:

- ❑ **Constant Current (CC):**
  - ✦ When the battery is deeply discharged, the charger supplies a constant current (default: 1A, adjustable via RPROG).
- ❑ **Constant Voltage (CV):**
  - ✦ As the battery approaches 4.2V, the charger shifts to constant voltage mode and reduces the charging current.
- ❑ **Charge Termination:**
  - ✦ Charging stops when the current falls below 10% of the programmed value (default: 100mA for 1A charging).
  - ✦ The module indicates full charge with the blue/green LED.

### IV. Protection Features –

- ❑ **Overcharge Protection:**
  - ✦ Prevents the battery voltage from exceeding 4.2V.
- ❑ **Over-Discharge Protection:**
  - ✦ Disconnects the battery if voltage falls below 2.4V.
- ❑ **Overcurrent Protection:**
  - ✦ Limits current draw during short circuits or excessive loads.
- ❑ **Reverse Polarity Protection:**
  - ✦ Ensures safety if the battery is connected incorrectly.

### V. Adjustable Charging Current –

The charging current is determined by the resistor connected to the **PROG** pin. The default configuration provides 1A charging. The current can be adjusted using the following formula:

$$I_{\text{CHG}} = \frac{1200}{R_{\text{PROG}}} \text{ (Amperes, R in k}\Omega\text{)}$$

### VI. Specifications –

- ❑ **Input Voltage:** 4.5V–5.5V
- ❑ **Charge Voltage:** 4.2V ± 1.5%
- ❑ **Maximum Charge Current:** 1A (default)
- ❑ **Operating Temperature:** -40°C to 85°C
- ❑ **Standby Current:** < 1µA

### VII. Advantages –

- a) Compact and cost-effective design.
- b) Integrated safety features.
- c) Easy to use with plug-and-play functionality.
- d) Adjustable charging current for flexibility.

### VIII. Limitations –

- a) No thermal regulation for high currents; prolonged charging may overheat the module.
- b) Designed for single-cell lithium-ion batteries only.

## 10. Step-down (buck) voltage regulator (12V dc to 3.3V, 5V, 12V dc Converter) –

This power module is a **step-down (buck) voltage regulator** designed to efficiently convert a 12V DC input into multiple regulated DC outputs such as **3.3V, 5V, and 12V**. It uses **LM317 or AMS1117 regulators** or similar ICs for precise voltage regulation.

### I. Key Features –

- ❑ **Input Voltage:**
  - ↳ Nominal: **12V DC**
  - ↳ Range: Typically, **9V–14V DC**



*Figure 28: Step-down (buck) voltage regulator*

### ❑ Outputs:

- ↳ **3.3V DC**
- ↳ **5V DC**
- ↳ **12V DC** (may also act as a pass-through from input if it matches the input voltage).

### ❑ Output Current:

- ↳ Up to **1A** (varies by regulator and thermal dissipation).

### ❑ Efficiency:

- ↳ Varies by output but typically around 80–90% for step-down conversion.

### ❑ Protection Features:

- ↳ Overcurrent protection.
- ↳ Thermal shutdown for regulators.
- ↳ Short-circuit protection.

## II. Pin Description –

- ↳ **GND:** Common ground for input and outputs.
- ↳ **3.3V:** Regulated 3.3V DC output for powering low-power devices such as microcontrollers.
- ↳ **5V:** Regulated 5V DC output suitable for logic-level electronics and peripherals.
- ↳ **12V:** Direct pass-through or regulated output for 12V devices.
- ↳ **DC Jack:** Input voltage source (connect a 12V adapter).



*Figure 29: Step-down (buck) voltage regulator Pinout*

## III. Working Principle –

The module operates based on linear voltage regulation (or switching, depending on the regulator design):

### ❑ Voltage Regulation:

- ↳ Converts the higher input voltage (12V) to lower, stable voltages using buck regulators or linear ICs.
- ↳ Regulated outputs ensure stable operation for sensitive electronics.

#### ❑ Thermal Management:

- ↳ Heat sinks on the regulators manage excess heat produced during voltage drop.
- ↳ The efficiency is higher for smaller voltage differences (e.g., 5V from 12V).

#### IV. Specifications –

- ❑ **Input Voltage:** 9V–14V DC
- ❑ **Output Voltage (3.3V):**  $3.3V \pm 5\%$
- ❑ **Output Voltage (5V):**  $5V \pm 5\%$
- ❑ **Output Voltage (12V):** 12V (pass-through)
- ❑ **Output Current:** Max 1A per channel
- ❑ **Efficiency:** ~80% (varies)
- ❑ **Operating Temperature:** -40°C to 85°C

#### V. Advantages –

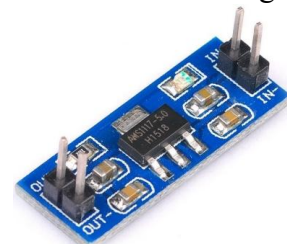
- a) Provides multiple regulated voltages from a single input.
- b) Compact and easy to use with plug-and-play terminals.
- c) Built-in protection for overcurrent and thermal safety.
- d) Suitable for a wide range of applications.

#### VI. Limitations –

- a) Limited output current (typically 1A max per channel).
- b) Linear regulators (if used) can waste power as heat, reducing efficiency.
- c) Input voltage must remain stable; significant deviations may affect output quality.

#### 11. AMS1117 5V Step-Down Power Supply Module –

The AMS1117 5V module is a compact voltage regulator that converts higher input voltages (e.g., 6.5V–12V) to a stable 5V output, commonly used for powering microcontrollers, sensors, and other 5V devices. It employs the **AMS1117-5.0 regulator**, a linear voltage regulator with built-in thermal and short-circuit protection.



*Figure 30: AMS1117 5V Step-Down Power Supply Module*

## I. Specifications –

### ❑ Input Voltage Range:

- ↳ Minimum: **6.5V**
- ↳ Maximum: **12V**

### ❑ Output Voltage:

- ↳ Fixed at **5V ± 1%**.

### ❑ Maximum Output Current:

- ↳ Up to **1A**.

### ❑ Efficiency:

- ↳ Lower than buck converters due to linear regulation (approx. 50–60% depending on input voltage).

### ❑ Dropout Voltage:

- ↳ Typically, **1.1V** at 1A load (input voltage must be at least 6.1V to get a stable 5V output).

### ❑ Protections:

- ↳ Thermal shutdown.
- ↳ Overcurrent protection.
- ↳ Short-circuit protection.

### ❑ Operating Temperature Range:

- ↳ **-40°C to +125°C**.

## II. Features –

### ❑ LED Indicator:

- ↳ A small onboard LED lights up when the module is powered, indicating proper input voltage.

### ❑ Decoupling Capacitors:

- ↳ The module includes capacitors (typically 10μF and 0.1μF) for input and output voltage stabilization, ensuring smooth and ripple-free output.

### ❑ Compact Size:

- ↳ Fits easily into breadboards and small circuits.

### ❑ Pin Headers:

- ↳ Designed for soldering or direct connection to wires or headers.

### III. Pin Description –

- ❑ **VIN:** Input voltage (6.5V–12V DC).
- ❑ **GND:** Ground pin, common for input and output.
- ❑ **VOUT:** Regulated 5V DC output.

### AMS1117 Pinout



Figure 31: AMS1117 Pinout

### IV. Working Principle –

The module uses the **AMS1117-5.0 linear regulator** to step down the input voltage:

#### ❑ Input Stage:

↳ The input voltage (6.5V–12V) is fed to the AMS1117 regulator through the VIN pin.

#### ❑ Voltage Regulation:

↳ The AMS1117 reduces the input voltage to a precise and stable 5V output using its internal reference voltage and feedback mechanism.

#### ❑ Output Filtering:

↳ Capacitors on the module filter the output voltage, removing noise and ripple for smooth operation.

### V. Advantages –

- a) Compact and easy to use.
- b) Stable 5V output suitable for sensitive electronics.
- c) Built-in protections ensure safe operation.
- d) Low-cost solution for 5V power needs.

### VI. Limitations –

- a) **Low Efficiency:** Linear regulators dissipate excess power as heat, making them less efficient compared to switching regulators.
- b) **Heat Generation:** At high input voltages and currents, significant heat is produced, requiring proper ventilation or a heat sink for prolonged use.
- c) **Dropout Voltage:** Requires input voltage to be at least 1.1V higher than 5V to regulate properly.



## 12. PIR Sensor –

A **PIR (Passive Infrared) Sensor**, as shown in the image, is a motion detection sensor designed to detect changes in infrared radiation emitted by warm objects like humans and animals. Below is a detailed explanation of its components, working, and usage:



Figure 32: PIR Sensor

### I. Components of PIR Sensor –

#### a) Dome-Shaped Fresnel Lens:

- ❑ The translucent dome is a **Fresnel lens**, which focuses infrared radiation from a wide area onto the PIR sensor.
- ❑ It enhances the sensitivity and field of view by segmenting the area into multiple zones.

#### b) PIR Sensing Elements:

- ❑ Inside the sensor, there are **two pyroelectric infrared sensors**.
- ❑ These elements detect differences in infrared radiation levels caused by moving objects.

#### c) PCB Board:

- ❑ The PCB contains:
  - ↳ **Three terminals:** VCC (power), OUT (signal output), and GND (ground).
  - ↳ **Amplifier and comparator circuit:** To process the weak signals generated by the sensing elements.
  - ↳ **Adjustable potentiometers:** To configure sensitivity and delay time.

#### d) Electrolytic Capacitors:

- ❑ These capacitors smooth out power supply fluctuations, ensuring stable operation.

### II. Working Principle –

#### ❑ Passive Detection:

- ↳ PIR sensors **do not emit radiation**; they only sense infrared radiation naturally emitted by objects.

### ❑ Detection Process:

↪ When the sensor is powered, it continuously monitors infrared radiation in its detection area.

↪ **Stable IR Levels:** If the infrared radiation in the field of view remains constant (e.g., a stationary object), the sensor outputs no signal.

↪ **Changes in IR Levels:** When a moving object, such as a person, enters the field of view, it causes a change in IR levels between the two sensing elements.

↪ **Signal Generation:** This change is processed by the amplifier circuit and produces a digital HIGH signal on the output pin (OUT).

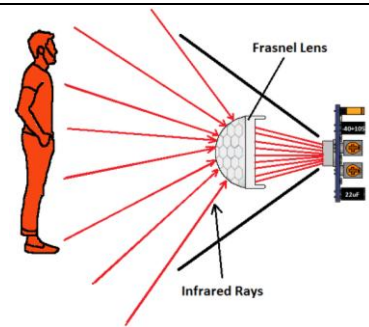


Figure 33: Working Principle Of PIR

### III. Specifications (from typical datasheets) –

- ❑ **Operating Voltage:** 3.3V to 5V
- ❑ **Detection Range:** 3 to 7 meters
- ❑ **Field of View (FOV):** 120° horizontally, 90° vertically
- ❑ **Output Signal:** Digital (HIGH or LOW)
- ❑ **Delay Time:** Adjustable (e.g., 5 sec to 5 min)
- ❑ **Trigger Modes:** Single trigger / Repeat trigger

### IV. Pinout –

❑ **VCC (Power Supply):** Connects to a 3.3V–5V power source.

❑ **OUT (Output):**

↪ Provides a HIGH (1) signal when motion is detected.

↪ Remains LOW (0) otherwise.

❑ **GND (Ground):** Connects to the system's ground.

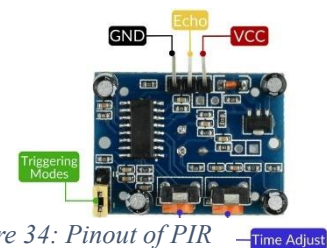


Figure 34: Pinout of PIR

### V. Adjustable Parameters –

❑ **Sensitivity Adjustment (Potentiometer 1):**

↪ Controls the detection range.

↪ Rotating clockwise increases the sensitivity.

### ❑ Delay Time Adjustment (Potentiometer 2):

- ✍ Controls the duration of the output signal after detecting motion.
- ✍ Rotating clockwise increases the delay.

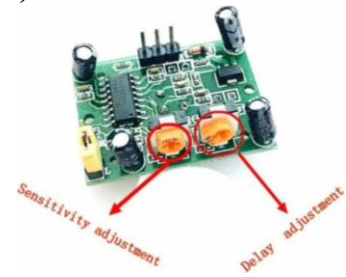


Figure 35: Adjustable Parameters

### VI. Wiring Example –

For connecting the PIR sensor to a microcontroller:

- ❑ **VCC:** Connect to the 5V pin of the microcontroller.
- ❑ **OUT:** Connect to a digital input pin to read the signal.
- ❑ **GND:** Connect to the ground pin of the microcontroller.

### VII. Troubleshooting Tips –

- ❑ Ensure the power supply voltage is within the operating range.
- ❑ Avoid placing the sensor in areas with direct sunlight or heating elements.
- ❑ Properly adjust the sensitivity and delay time to suit your application.

### 13. 12v Dc Fan –

A **12V DC fan** is a type of fan powered by a 12-volt direct current (DC) power source, commonly used for cooling purposes in various applications. These fans are compact, energy-efficient, and versatile, making them suitable for use in electronics, vehicles, and ventilation systems. Below is a typical description:



Figure 36: 12V DC Fan

#### ❑ Features:

- ✍ **Voltage:** Operates on 12 volts DC, making it compatible with a wide range of DC power supplies, including batteries and adapters.
- ✍ **Size:** Available in various sizes (commonly 40mm, 80mm, 120mm, etc.), suitable for different cooling requirements.
- ✍ **Material:** Usually made of durable plastic or metal for longevity and lightweight construction.
- ✍ **Blades:** Equipped with aerodynamic blades to ensure efficient airflow.

↪ **Speed:** Fan speed varies depending on the model, typically measured in revolutions per minute (RPM).

↪ **Power Consumption:** Consumes low power, making it energy-efficient.

↪ **Noise Levels:** Designed to operate quietly, though noise levels vary by model (expressed in dBA).

❑ **Benefits:**

↪ **Energy Efficiency:** Consumes less power compared to AC fans.

↪ **Compact Design:** Easy to install in tight spaces.

↪ **Long Lifespan:** Built for extended operation with reliable performance.

#### 14. IC 7805 (Voltage Regulator) –

❑ **Description:** The 7805 is a linear voltage regulator IC that provides a constant output of **5V** DC for a range of input voltages (typically 7–35V). It is used to power 5V devices and circuits.

❑ **Pins:**

1. **Input (Pin 1):** Accepts input voltage (7V to 35V).
2. **Ground (Pin 2):** Common ground for input and output.
3. **Output (Pin 3):** Provides a regulated 5V DC output.

❑ **Usage:** Protects circuits from voltage fluctuations, ensuring stable operation.

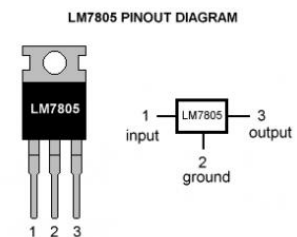


Figure 37: IC 7805

#### 15. 1μF Capacitor –

❑ **Description:** A 1μF capacitor stores and releases charge. It can filter noise, stabilize voltage, or act as a decoupling capacitor in power supply circuits.

❑ **Types:** May be **ceramic**, **electrolytic**, or **tantalum**. The type determines its application and polarity.



Figure 38: Capacitor

### ❑ Pins:

➤ For **polarized capacitors**:

1. **Positive pin**: Connects to a higher voltage side.
2. **Negative pin**: Marked with a stripe and connects to the ground or lower voltage.

➤ **Non-polarized** capacitors have no specific polarity.

❑ **Usage**: Often placed at the input/output of regulators to smooth voltage.

## 16. Buzzer –

❑ **Description**: A buzzer converts electrical signals into sound. It can be **active** (requires DC voltage) or **passive** (requires an AC signal or a control circuit).

### ❑ Pins:

1. **Positive Pin**: Marked with a '+' symbol.
2. **Negative Pin**: Connects to the ground.

❑ **Voltage Range**: Common operating voltage is **3V to 12V**.

❑ **Usage**: Audible alarms, notifications, or signal indicators.

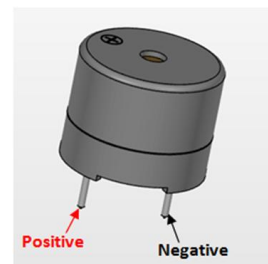


Figure 39: Buzzer

## 17. LED (Light Emitting Diode) –

❑ **Description**: LEDs emit light when current flows through them. They are available in various colours and intensities.

### ❑ Pins:

1. **Anode (Longer pin)**: Connects to the positive side of the power supply.
2. **Cathode (Shorter pin)**: Connects to the ground (or negative).

❑ **Operating Voltage**: Typically **2V to 3.3V**, depending on the LED colour.

❑ **Usage**: Indicators, lighting, and displays.

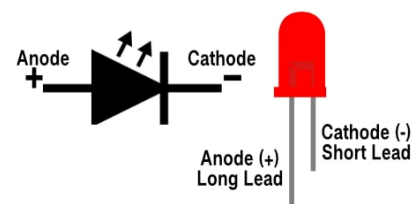


Figure 40: LED

### 18. DC Switch –

❑ **Description:** A switch is a mechanical or electronic device that can open or close an electrical circuit.

❑ **Pins:**

1. **Input Pin:** Connects to the power source.
2. **Output Pin:** Connects to the load.

❑ **Types:** Single Pole Single Throw (SPST) is common in basic DC applications.

❑ **Usage:** Controls the flow of current, turning devices ON or OFF.

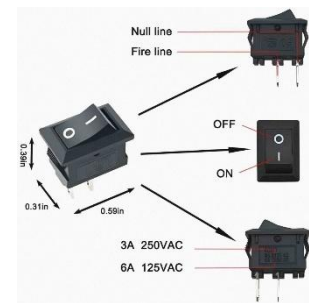


Figure 41: DC Switch

### 19. DC Motor –

❑ **Description:** A DC motor converts electrical energy into mechanical energy. The speed of the motor can be controlled by varying the input voltage.

❑ **Pins:**

1. **Positive Pin (RED):** Connects to the positive terminal of the power supply.
2. **Negative Pin (BLACK):** Connects to the ground or negative terminal.

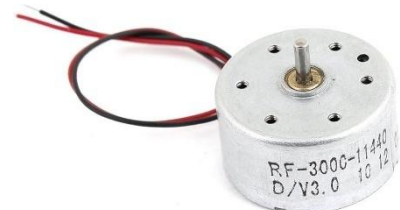


Figure 42: DC Motor

❑ **Voltage Range:** Common motors operate between **3V and 12V**.

❑ **Usage:** Drives mechanical systems in automation, robotics, and vehicles.

### 20. Battery (18650) –

❑ **Description:** The 18650 lithium-ion battery is a rechargeable cell with high energy density and a nominal voltage of **3.7V**. The capacity ranges from **2000mAh to 3500mAh**.

❑ **Pins:**

1. **Positive Terminal:** Marked with a '+' sign.
2. **Negative Terminal:** Marked with a '-' sign.

❑ **Usage:** Powers portable electronics, flashlights, and DIY projects.

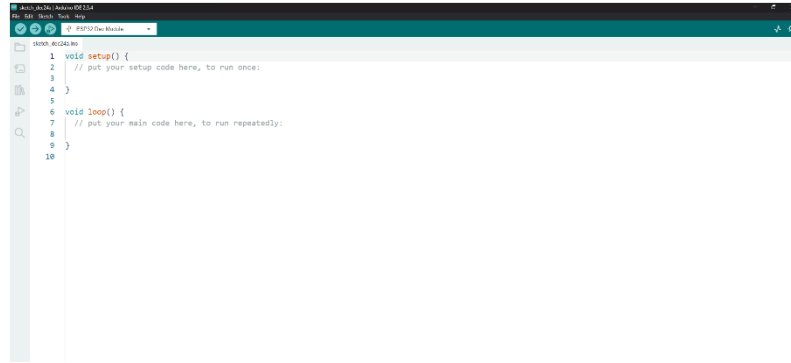


Figure 43: Battery (18650)

## ⇒ Software Used –

### ✚ Arduino IDE:

The **Arduino Integrated Development Environment (IDE)** is an open-source software platform that provides a user-friendly interface for programming Arduino microcontrollers. It's specifically designed to help



*Figure 44: Arduino IDE*

developers, hobbyists, and beginners write, compile, and upload code to Arduino boards with ease.

## ❑ Features of Arduino IDE

### ✚ Cross-Platform Support:

- Available for Windows, macOS, and Linux operating systems.

### ✚ Simple User Interface:

- The IDE is designed with simplicity in mind, featuring an editor for writing code and a console for debugging. The straightforward layout makes it beginner-friendly.

### ✚ Code Editor:

- The editor supports syntax highlighting, automatic indentation, and brace matching to improve readability and reduce errors.

### ✚ Predefined Libraries:

- Comes with a variety of built-in libraries for common tasks like controlling motors, interfacing sensors, or connecting to networks.

#### ✚ **Serial Monitor:**

- The Serial Monitor allows real-time communication between the Arduino board and your computer. It is useful for debugging or sending commands to the board.

#### ✚ **Board Manager:**

- A feature to add or update support for new Arduino boards or compatible hardware directly within the IDE.

#### ✚ **Sketch System:**

- A program written in the Arduino IDE is called a *sketch*. It is based on C/C++ and is stored in .ino files.

#### ✚ **Compiler and Uploader:**

- The IDE uses avr-gcc to compile the code and avr-dude to upload the code to the microcontroller.

#### ✚ **Extensibility:**

- Supports third-party boards and tools via JSON configuration files, enabling use with a variety of microcontrollers beyond standard Arduino boards.

### ❑ **Structure of Arduino IDE**

#### ✚ **Menu Bar:**

- Contains options like File, Edit, Sketch, Tools, and Help for managing projects, code, and settings.

#### ✚ **Code Editor Window:**

- Where users write and edit sketches.

#### ✚ **Output Console:**

- Displays messages such as errors, upload status, and compiler output.

#### ✚ **Toolbar:**

- Quick-access buttons for common tasks like verify (compile), upload, open, save, and start the Serial Monitor.

### ❑ **Advantages of Arduino IDE**

- ✚ **Beginner-Friendly:** Its simplicity allows even non-programmers to start experimenting with microcontrollers.

- ✚ **Open-Source:** Free to download, modify, and distribute.



↳ **Large Community Support:** Thousands of tutorials, examples, and forums are available.

↳ **Library Support:** Simplifies working with various sensors, actuators, and modules.

#### □ **Limitations**

↳ **Basic Text Editor:** Lacks advanced features like autocomplete or refactoring found in modern IDEs.

↳ **Performance:** Can be slow when handling large projects or multiple libraries.

↳ **Limited Debugging Tools:** It doesn't provide advanced debugging capabilities like step-by-step execution.

## EXPERIMENTAL DISCUSSION

### ❖ BLOCK DIAGRAM & FUNCTION

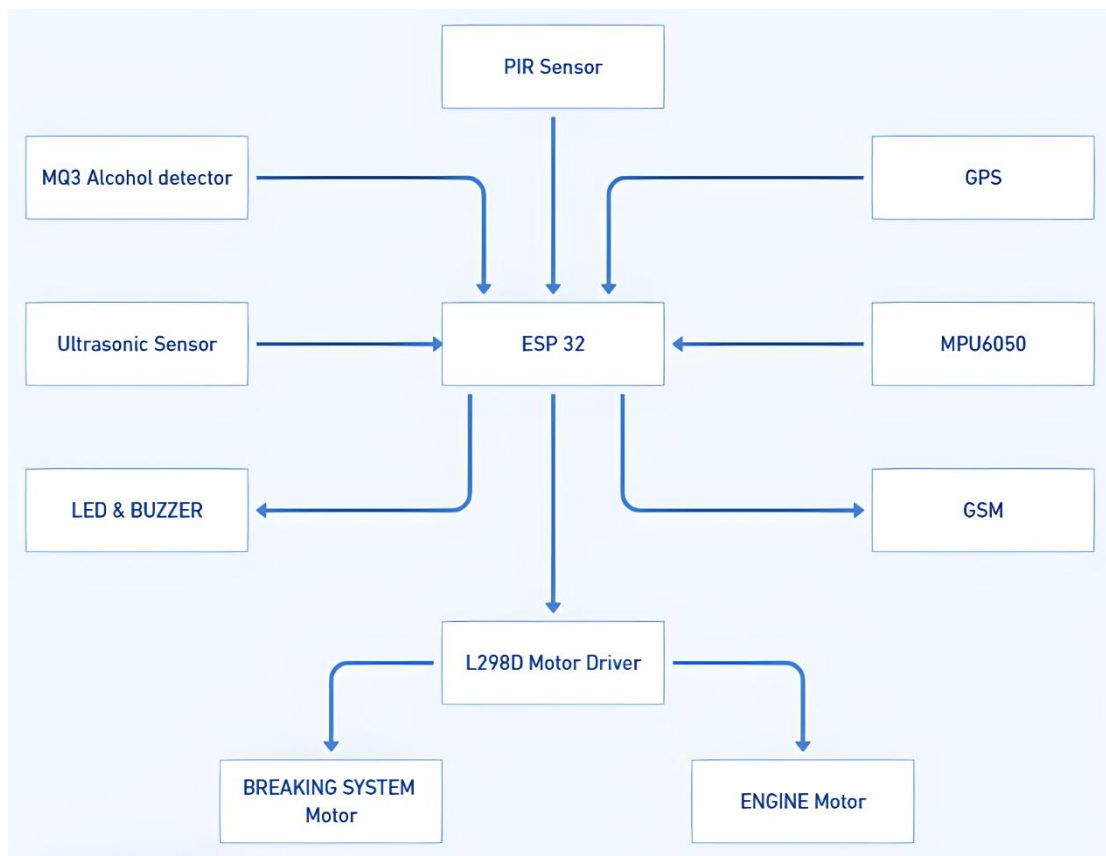


Figure 45: BLOCK DIAGRAM

This project serves as a prototype for a “Smart Accident Prevention, Detection And Reporting System (SAPDRS)” that integrates multiple sensors and modules to improve vehicle safety and efficiency. At the heart of the system is the ESP32 microcontroller, which coordinates the operations of various components to ensure real-time monitoring and decision-making. One of the primary features of the system is the integration of an MQ3 alcohol sensor, which detects alcohol levels in the driver’s breath. If the alcohol concentration exceeds a predetermined threshold, the system triggers an LED and a buzzer to alert the driver and other passengers. This immediate feedback serves as a preventative measure, discouraging impaired driving. Additionally, the system can take further action, such as restricting the engine motor’s operation, to ensure that the vehicle remains stationary, thereby preventing potential accidents caused by drunk driving.

Another critical component of the system is the ultrasonic sensor, which provides real-time obstacle detection. This sensor continuously monitors the vehicle's surroundings, identifying objects or hazards in its path. When an obstacle is detected within a critical

distance, the ultrasonic sensor sends a signal to the ESP32, which processes the data and determines the appropriate response. This could involve alerting the driver through the LED and buzzer, enabling them to take corrective action, or automatically activating the braking system to avoid a collision. The ultrasonic sensor's ability to detect objects accurately and quickly makes it an essential part of the system, particularly for preventing accidents in congested or high-risk areas. By incorporating obstacle detection, the system not only ensures the safety of the driver and passengers but also reduces the risk of harm to pedestrians and other road users.

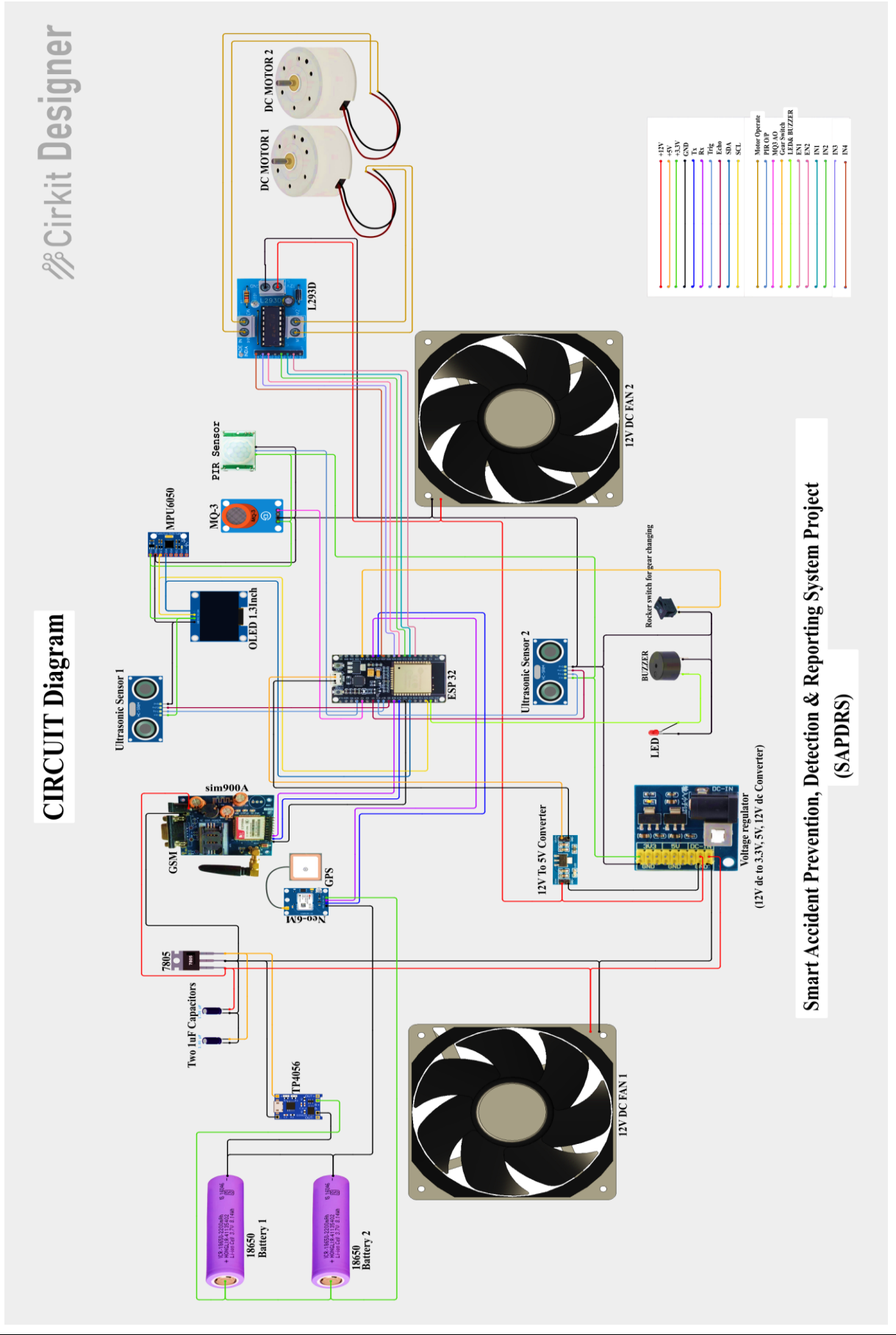
The PIR (Passive Infrared) sensor adds another dimension to the system by promoting energy efficiency and reducing fuel consumption. The PIR sensor is designed to detect the presence or absence of occupants in the vehicle. If no movement is detected for a predefined duration (indicating the vehicle is unoccupied), the system can automatically turn off auxiliary systems such as the air conditioning or idle engine. This feature prevents unnecessary fuel consumption when the vehicle is stationary and unoccupied, contributing to overall energy savings and environmental benefits. When the PIR sensor detects the presence of an occupant, the systems are reactivated seamlessly to ensure convenience and comfort.

The GPS and GSM modules add another layer of functionality to the system, enabling real-time location tracking and communication. The GPS module continuously provides the vehicle's coordinates, which are crucial in emergencies. In the event of an accident or other critical situations detected by the system, the GSM module is used to send an alert message containing the vehicle's location to predesignated contacts, such as family members, emergency services, or roadside assistance. This ensures that help can be dispatched promptly, minimizing delays in providing medical attention or technical support. By combining GPS and GSM technologies, the system enhances situational awareness and improves the efficiency of emergency response, making it particularly valuable in scenarios where immediate assistance is required.

The L293D motor driver plays a vital role in the prototype, managing the braking and engine systems. Connected to the ESP32, the L293D driver allows precise control over the vehicle's mechanical operations. For instance, in response to unsafe conditions detected by the MQ3 alcohol sensor, ultrasonic sensor, or PIR sensor, the motor driver can activate the braking system to bring the vehicle to a stop or restrict the engine motor's functionality to prevent it from starting. This integration ensures that the system not only monitors and alerts but also takes direct action to mitigate risks. While the MPU6050 module is included

in the design, its accelerometer functionality is not utilized in this version of the project. Overall, the system demonstrates an effective combination of technologies to prevent accidents, protect passengers, reduce fuel consumption, and provide timely responses in emergencies.

❖ CIRCUIT DIAGRAM



Smart Accident Prevention, Detection & Reporting System Project (SAPDRS)

## ❖ CIRCUIT CONNECTION EXPLANATION

### 1. Power Supply:

- ❑ The 18650 lithium-ion batteries have their positive terminal connected to the GPS module VCC and their negative terminal connected to the GPS module GND.
- ❑ The TP4056 module is connected to the battery terminals of the 18650 batteries, and its output terminals are connected to the GPS module.

### 2. Microcontroller (ESP32):

- ❑ For the ultrasonic sensors, the Sensor 1 trigger is connected to GPIO 17 and its echo is connected to GPIO 5, while the Sensor 2 trigger is connected to GPIO 16 and its echo is connected to GPIO 4.
- ❑ The MPU6050 sensor has its VCC connected to 3.3V, GND connected to GND, SCL connected to GPIO 22, and SDA connected to GPIO 21.
- ❑ The MQ-3 sensor has its VCC connected to 5V, GND connected to GND, and A0 connected to GPIO 34 (ADC).
- ❑ The PIR sensor has its VCC connected to 5V, GND connected to GND, and OUT connected to GPIO 2.

### 3. Communication Modules:

- ❑ The GSM module has its VCC connected to 5V, GND connected to GND, TX connected to GPIO 19, and RX connected to GPIO 18.
- ❑ The GPS module has its VCC connected to the 18650 batteries, GND connected to the 18650 batteries, TX connected to GPIO 14, and RX connected to GPIO 12.

### 4. Actuators:

- ❑ For the DC motors:
  - ↳ Motor 1 has EN1 connected to GPIO 35, IN1 connected to GPIO 32, and IN2 connected to GPIO 33.
  - ↳ Motor 2 has EN2 connected to GPIO 25, IN3 connected to GPIO 26, and IN4 connected to GPIO 27.
- ❑ The 12V DC fans have their VCC connected to the 12V supply and GND connected to GND.

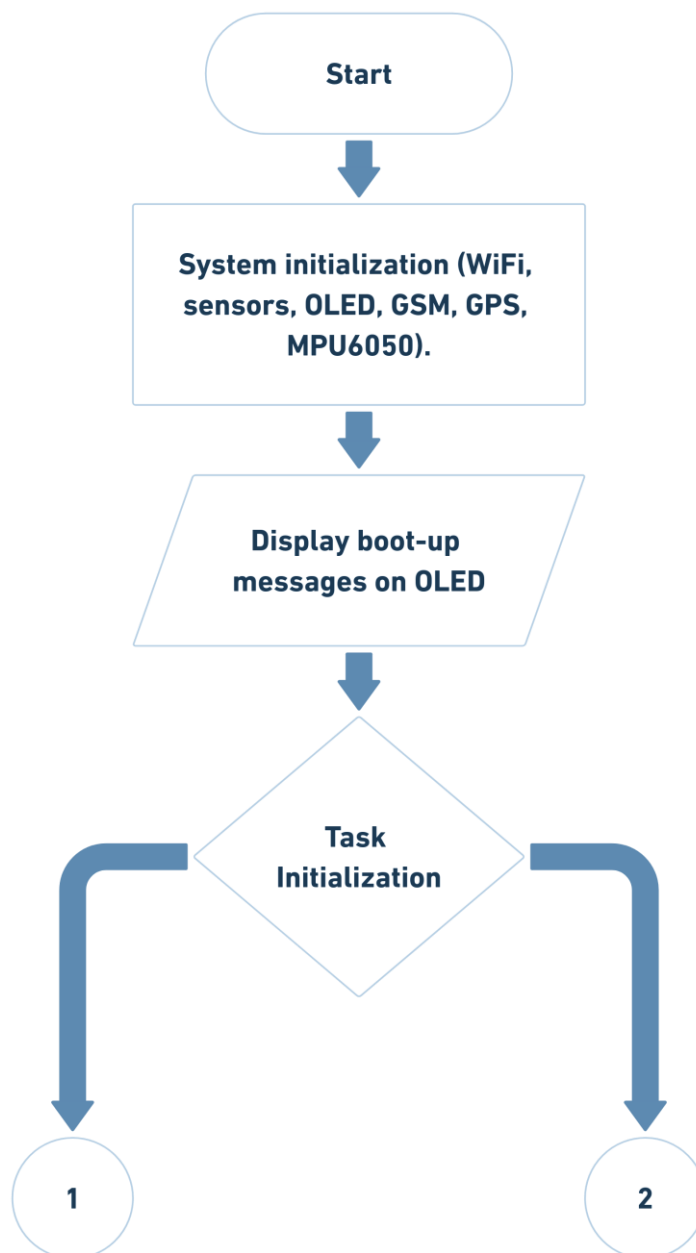
**5. Other Components:**

- ❑ The OLED display has its VCC connected to 3.3V, GND connected to GND, SCL connected to GPIO 22, and SDA connected to GPIO 21.
- ❑ The LED has its anode connected to GPIO 23 and cathode connected to GND.
- ❑ The buzzer has its positive terminal connected to GPIO 23 and negative terminal connected to GND.
- ❑ The rocker button is connected to GPIO 13 and GND for gear changing.

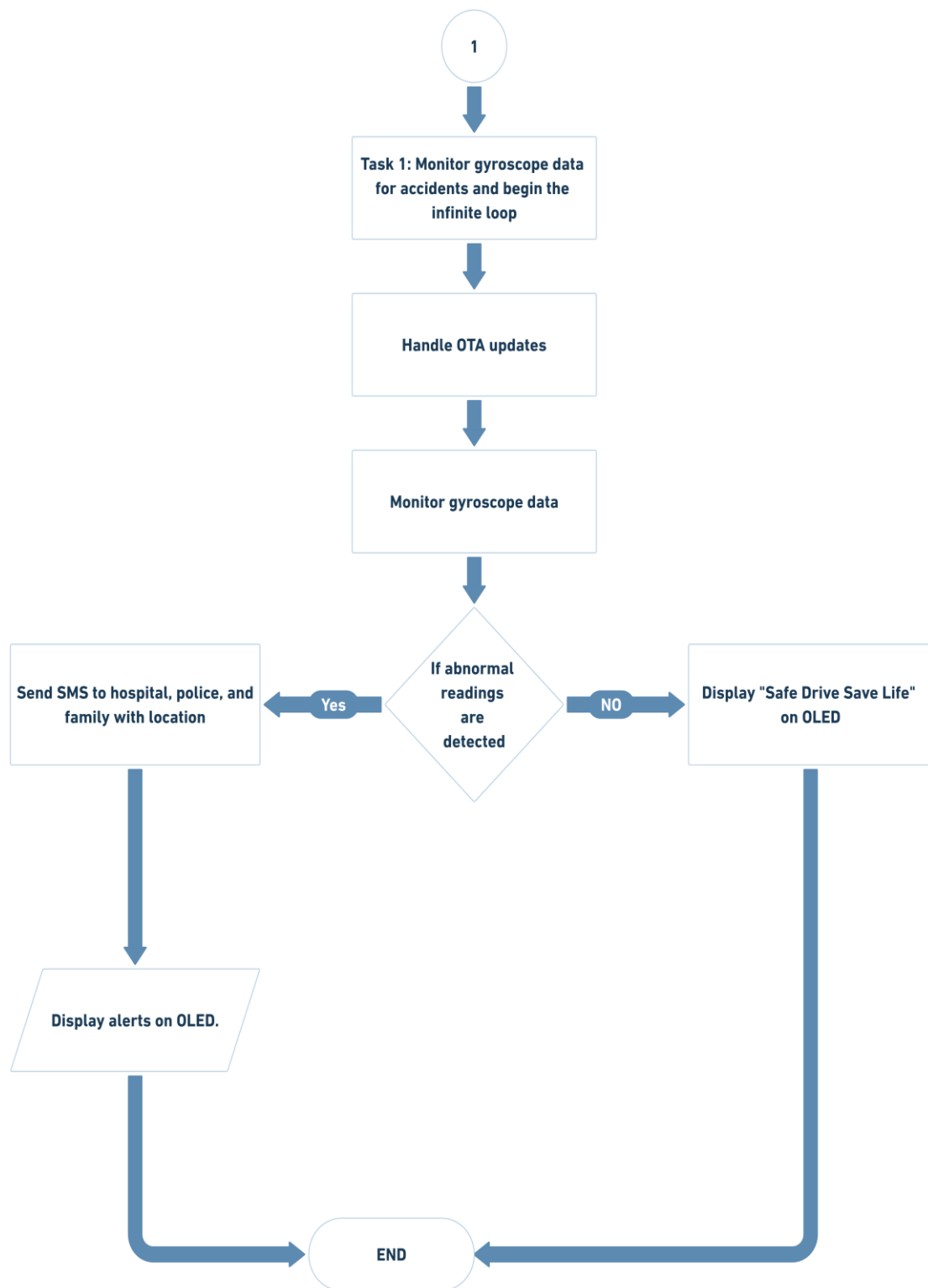
**6. Miscellaneous:**

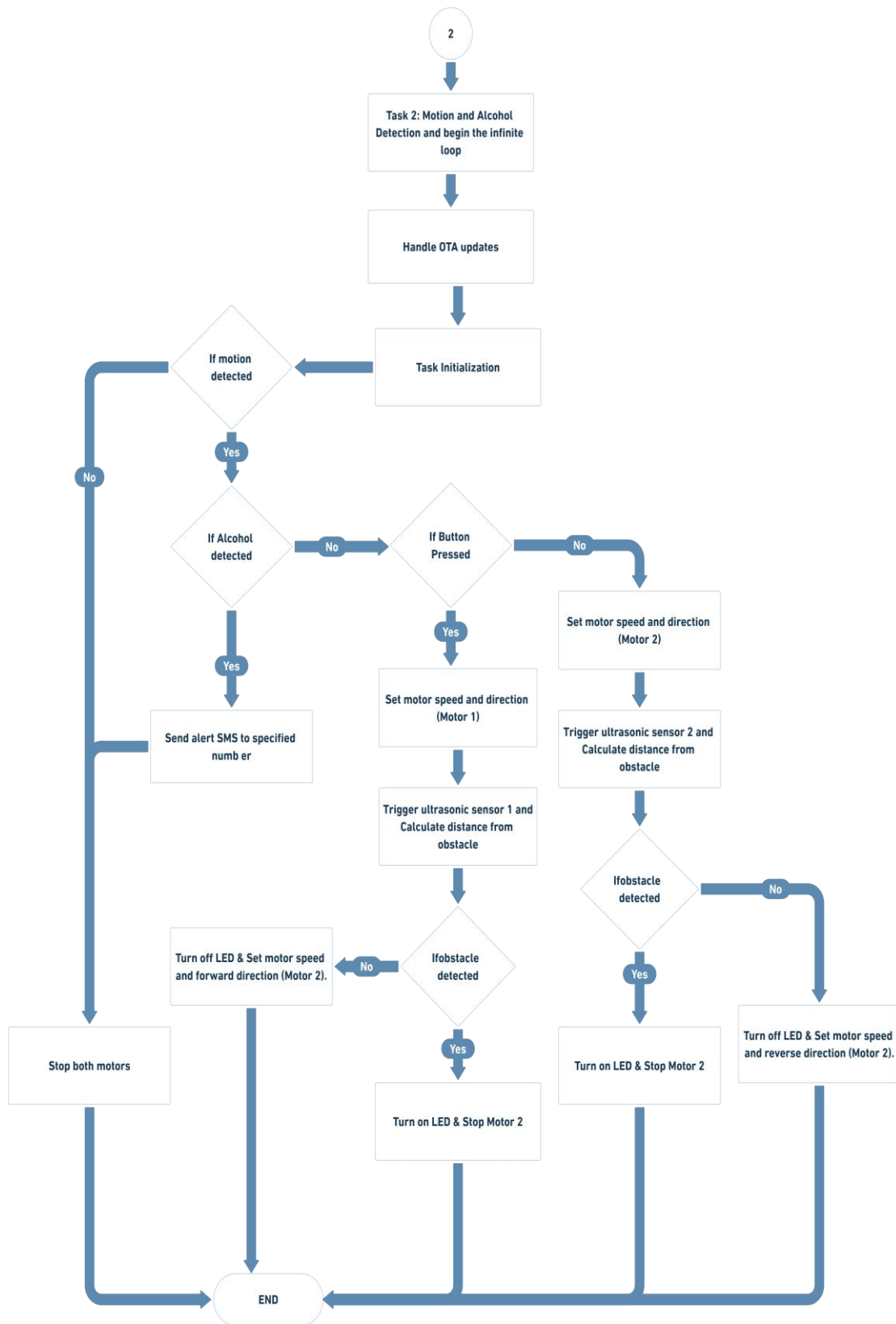
- ❑ The 7805 voltage regulator has its input from the TP4056 module and its output to the TP4056 input.

## ❖ FLOWCHART









## ❖ ALGORITHM

### 1. Start:

- ❑ Initialize IMU, sensors, GSM module, GPS module, OLED display, and Wi-Fi/Bluetooth connectivity.
- ❑ Display boot-up messages on the OLED.

### 2. Task Initialization:

- ❑ Branch into Task 1 and Task 2.

### 3. Task 1: Monitor Gyroscope Data:

- ❑ Handle continuous GPS updates.
- ❑ Monitor gyroscope data from the MPU6050 sensor.
  - If an abnormal reading (indicative of an accident) is detected:
    - Fetch GPS coordinates.
    - Send SMS alerts with GPS coordinates and a Google Maps link to predefined contacts (hospitals, police, family members).
    - Display alerts on the OLED.
  - If no abnormal reading is detected, continue monitoring gyroscope data.

### 4. Task 2: Motion and Alcohol Detection:

- ❑ Handle continuous GPS updates.
- ❑ Initialize sensors and modules.
- ❑ If motion is detected by the PIR sensor:
  - If alcohol is detected by the MQ-3 sensor:
    - Send SMS alert about alcohol detection.
    - Stop both motors.
  - If no alcohol is detected:
    - Check if the button is pressed:
      - If pressed:
        - Set motor speed and direction for Motor 2.
        - Trigger Ultrasonic Sensor 2.
        - If an obstacle is detected:
          - Turn on LED B.
          - Stop Motor 2.
          - Reverse Motor 2.

- If no obstacle is detected, continue monitoring.
- If not pressed:
  - Set motor speed and direction for Motor 1.
  - Trigger Ultrasonic Sensor 1.
  - If an obstacle is detected:
    - Turn on LED A.
    - Stop Motor 1.
    - Reverse Motor 1.
  - If no obstacle is detected, continue monitoring.
- ▣ If no motion is detected, continue monitoring for motion.

## ❖ PROGRAM CODE

Table 2: System Program CODE

```

#include <WiFi.h> // Include the WiFi library

#include <ArduinoOTA.h> // Include the OTA (Over-the-Air) updates library

#include <U8g2lib.h> // Include the U8g2 library for OLED display

#include <SPI.h> // Include the SPI library for communication

#include <Adafruit_MPU6050.h> // Include the Adafruit MPU6050 library for the
accelerometer and gyroscope

#include <Adafruit_Sensor.h> // Include the Adafruit Sensor library for sensor abstractions

#include <HardwareSerial.h> // Include the HardwareSerial library for serial
communication

#include <TinyGPS++.h> // Include the TinyGPS++ library for GPS data

#include <Wire.h> // Include the Wire library for I2C communication

#include <freertos/FreeRTOS.h> // Include the FreeRTOS library for multitasking

#include <freertos/task.h> // Include the FreeRTOS task library

const char* ssid = "INFINITY"; // Define the SSID of the WiFi network

const char* password = "Kd8637333961!!"; // Define the password of the WiFi network

#define EN1 35 // Define the EN1 pin

#define IN1 32 // Define the IN1 pin

#define IN2 33 // Define the IN2 pin

```



[illegible][illegible][illegible][illegible][illegible][illegible][illegible][illegible]

```
0x00, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00,
```

0x00, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xfe, 0x00, 0x00, 0x00, 0x03, 0xfc, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xfe, 0x00, 0x00, 0x00, 0x07, 0xf8, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x1f, 0xff, 0xff, 0x0f, 0xe0, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x1f, 0xff, 0xff, 0x1f, 0xc0, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3e, 0x1f, 0xff, 0xfe, 0x3f, 0xc0, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x1f, 0xff, 0xfc, 0x7f, 0x80, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x1f, 0xff, 0xf8, 0x7f, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x1f, 0x83, 0xf0, 0xfe, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x1f, 0x87, 0xf1, 0xfc, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x1f, 0x8f, 0xe3, 0xfc, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x9f, 0xc7, 0xf8, 0x00, 0x00, 0x00, 0x00,  
0x00,



0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbf, 0x87, 0xf0, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbf, 0x0f, 0xe0, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbf, 0x1f, 0xc0, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbe, 0x3f, 0xc0, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbc, 0x7f, 0x80, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xb8, 0x7f, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xb0, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xa1, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x83, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x87, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x87, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x8f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x9f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbf, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbf, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbf, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xbc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xb8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xb0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xb0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00,

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00
```

```
};
```

```
float latitude; // Define a variable to store the latitude
```

```
float longitude; // Define a variable to store the longitude
```

```
const int led = 23; // Define the LED pin (white)
```

```
const int p = 2; // Define the PIR sensor pin (green)

const int button = 13; // Define the button pin

const int t1 = 17; // Define the ultrasonic sensor 1 trigger pin

const int e1 = 5; // Define the ultrasonic sensor 1 echo pin

const int t2 = 16; // Define the ultrasonic sensor 2 trigger pin

const int e2 = 4; // Define the ultrasonic sensor 2 echo pin

long duration; // Define a variable to store the duration of ultrasonic pulse

float distance; // Define a variable to store the calculated distance

TaskHandle_t Task1; // Define a task handle for Task1

TaskHandle_t Task2; // Define a task handle for Task2

void Task1Code(void* parameter); // Declare Task1Code function

void Task2Code(void* parameter); // Declare Task2Code function

void sendSMS(const char* phoneNumber); // Declare sendSMS function

void alsms(const char* phoneNumber); // Declare alsms function

int accidentCount = 0; // Define a variable to count the number of accidents

void setup() {

    Serial.begin(9600); // Initialize serial communication at 9600 baud rate

    pinMode(button, INPUT_PULLUP); // Set button pin as input with internal pull-up resistor

    pinMode(p, INPUT); // Set PIR sensor pin as input

    pinMode(al, INPUT); // Set alcohol sensor pin as input
```

```
pinMode(e1, INPUT); // Set ultrasonic sensor 1 echo pin as input

pinMode(t1, OUTPUT); // Set ultrasonic sensor 1 trigger pin as output

pinMode(e2, INPUT); // Set ultrasonic sensor 2 echo pin as input

pinMode(t2, OUTPUT); // Set ultrasonic sensor 2 trigger pin as output

pinMode(led, OUTPUT); // Set LED pin as output

pinMode(IN1, OUTPUT); // Set motor driver input pin 1 as output

pinMode(IN2, OUTPUT); // Set motor driver input pin 2 as output

pinMode(IN3, OUTPUT); // Set motor driver input pin 3 as output

pinMode(IN4, OUTPUT); // Set motor driver input pin 4 as output

pinMode(EN1, OUTPUT); // Set motor driver enable pin 1 as output

pinMode(EN2, OUTPUT); // Set motor driver enable pin 2 as output

oled.begin(); // Initialize the OLED display

oled.clear(); // Clear the OLED display

oled.setFont(u8g2_font_ncenB08_tr); // Set the font for the OLED display

oled.firstPage(); // Prepare the first page for drawing

do {

    oled.drawBitmap(0, 0, 128 / 8, 64, logo); // Draw the logo on the OLED display

} while (oled.nextPage()); // Continue to the next page if there is one

delay(10000); // Wait for 10 seconds

oled.clearBuffer(); // Clear the display buffer
```

```
oled.setFont(u8g2_font_ncenB18_tf); // Set the font for the OLED display

oled.setCursor(5, 20); // Set cursor position

oled.println("Welcome"); // Print "Welcome" on the OLED display

oled.setCursor(42, 60); // Set cursor position

oled.println("Sir"); // Print "Sir" on the OLED display

oled.sendBuffer(); // Send the buffer to the OLED display

delay(2000); // Wait for 2 seconds

SerialGSM.begin(9600, SERIAL_8N1, 19, 18); // Initialize serial communication for GSM
at 9600 baud rate

SerialGPS.begin(9600, SERIAL_8N1, 14, 12); // Initialize serial communication for GPS
at 9600 baud rate

delay(100); // Wait for 100 milliseconds

if (!mpu.begin()) { // Check if the MPU6050 sensor initialization failed

    Serial.println("Failed to find MPU6050 chip"); // Print error message to serial monitor

    while (1) delay(10); // Wait indefinitely

}

mpu.setAccelerometerRange(MPU6050_RANGE_8_G); // Set the accelerometer range

mpu.setGyroRange(MPU6050_RANGE_500_DEG); // Set the gyroscope range

mpu.setFilterBandwidth(MPU6050_BAND_21_HZ); // Set the filter bandwidth

Serial.println("Sensor initialized."); // Print initialization message to serial monitor
```

```
oled.clearBuffer(); // Clear the display buffer

oled.setFont(u8g2_font_ncenB14_tf); // Set the font for the OLED display

oled.setCursor(20, 15); // Set cursor position

oled.println("Sensor"); // Print "Sensor" on the OLED display

oled.setCursor(5, 50); // Set cursor position

oled.println("Initialized"); // Print "Initialized" on the OLED display

oled.sendBuffer(); // Send the buffer to the OLED display

delay(3000); // Wait for 3 seconds

Serial.println("Booting"); // Print booting message to serial monitor

oled.clearBuffer(); // Clear the display buffer

oled.setFont(u8g2_font_ncenB18_tf); // Set the font for the OLED display

oled.setCursor(5, 40); // Set cursor position

oled.println("Booting..."); // Print "Booting..." on the OLED display

oled.sendBuffer(); // Send the buffer to the OLED display

delay(2000); // Wait for 2 seconds

WiFi.begin(ssid, password); // Connect to WiFi network

while (WiFi.status() != WL_CONNECTED) { // Wait until WiFi is connected

    delay(1000); // Wait for 1 second

    Serial.println("Connecting to WiFi..."); // Print connecting message to serial monitor

}
```

```
Serial.println("Connected to WiFi"); // Print connected message to serial monitor

oled.clearBuffer(); // Clear the display buffer

oled.setFont(u8g2_font_ncenB12_tf); // Set the font for the OLED display

oled.setCursor(0, 20); // Set cursor position

oled.println("Connected"); // Print "Connected" on the OLED display

oled.setCursor(40, 40); // Set cursor position

oled.println("To"); // Print "To" on the OLED display

oled.setCursor(40, 60); // Set cursor position

oled.println("WIFI.."); // Print "WIFI.." on the OLED display

oled.sendBuffer(); // Send the buffer to the OLED display

delay(1000); // Wait for 1 second

ArduinoOTA.onStart([]() { // Define the OTA onStart callback

    String type; // Create a string to hold the update type

    if (ArduinoOTA.getCommand() == U_FLASH) { // Check if the command is to update
the sketch

        type = "sketch"; // Set type to "sketch"

    }

    else { // Otherwise

        type = "filesystem"; // Set type to "filesystem"

    }

}
```



```

Serial.println("Start updating " + type); // Print start updating message to serial monitor

});

ArduinoOTA.onEnd([]() { // Define the OTA onEnd callback

    Serial.println("\nEnd"); // Print end message to serial monitor

});

ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) { // Define the OTA
onProgress callback

    Serial.printf("Progress: %u%%\r", (progress / (total / 100))); // Print progress message to
serial monitor

});

ArduinoOTA.onError([](ota_error_t error) { // Define the OTA onError callback

    Serial.printf("Error[%u]: ", error); // Print error message to serial monitor

    if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed"); // Print auth failed
message if authentication error

    else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed"); // Print begin
failed message if beginning error

    else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed"); // Print
connect failed message if connection error

    else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed"); // Print
receive failed message if receiving error

    else if (error == OTA_END_ERROR) Serial.println("End Failed"); // Print end failed
message if ending error

```

```
});

ArduinoOTA.begin(); // Begin OTA updates

oled.clearBuffer(); // Clear the display buffer

oled.setFont(u8g2_font_ncenB18_tf); // Set the font for the OLED display

oled.setCursor(2, 20); // Set cursor position

oled.println("OTA"); // Print "OTA" on the OLED display

oled.setCursor(2, 60); // Set cursor position

oled.println("Ready..."); // Print "Ready..." on the OLED display

oled.sendBuffer(); // Send the buffer to the OLED display

delay(2000); // Wait for 2 seconds

Serial.println("OTA ready"); // Print OTA ready message to serial monitor

Serial.print("IP address: "); // Print IP address label to serial monitor

Serial.println(WiFi.localIP()); // Print local IP address to serial monitor

xTaskCreatePinnedToCore(Task1Code, "Task1", 10000, NULL, 1, &Task1, 0); // Create
Task1 on core 0

xTaskCreatePinnedToCore(Task2Code, "Task2", 10000, NULL, 1, &Task2, 1); // Create
Task2 on core 1

}

void loop() {}

void Task1Code(void* parameter) {
```

```

while (1) { // Infinite loop

    ArduinoOTA.handle(); // Handle OTA updates

    while (SerialGPS.available() > 0) { // Check if data is available from the GPS

        gps.encode(SerialGPS.read()); // Encode the GPS data

        if (gps.location.isUpdated()) { // Check if the GPS location is updated

            latitude = gps.location.lat(); // Update the latitude

            longitude = gps.location.lng(); // Update the longitude

        }

    }

    sensors_event_t a, g, temp; // Create sensor event variables

    mpu.getEvent(&a, &g, &temp); // Get sensor events for accelerometer, gyroscope, and
temperature

    if (abs(g.gyro.x) > 2 || abs(g.gyro.y) > 2 || abs(g.gyro.z) > 2) { // Check if the gyroscope
readings exceed threshold

        accidentCount++; // Increment accident count if threshold is exceeded

    }

    else {

        accidentCount = 0; // Reset accident count if threshold is not exceeded

    }

    if (accidentCount >= 2) { // Check if accident count is greater than or equal to 2

```

```
// Confirm after 3 consecutive detections

oled.clearBuffer(); // Clear OLED display buffer

oled.setCursor(2, 10); // Set cursor position on OLED display

oled.setFont(u8g2_font_ncenB10_tf); // Set font for OLED display

oled.println("Accident Alert!!!"); // Print accident alert message

oled.setCursor(30, 30); // Set cursor position on OLED display

oled.println("Send SMS"); // Print send SMS message

oled.setCursor(22, 50); // Set cursor position on OLED display

oled.println("TO Hospital"); // Print message to hospital

oled.sendBuffer(); // Send buffer to OLED display

sendSMS("+919064564189"); // Send SMS to hospital

delay(6000); // Delay for 6 seconds

oled.clearBuffer(); // Clear OLED display buffer

oled.setCursor(2, 10); // Set cursor position on OLED display

oled.println("Accident Alert!!!"); // Print accident alert message

oled.setCursor(23, 30); // Set cursor position on OLED display

oled.println("Send SMS"); // Print send SMS message

oled.setCursor(20, 50); // Set cursor position on OLED display

oled.println("To Police"); // Print message to police

oled.sendBuffer(); // Send buffer to OLED display
```

```
sendSMS("+918918673369"); // Send SMS to police

delay(6000); // Delay for 6 seconds

oled.clearBuffer(); // Clear OLED display buffer

oled.setCursor(2, 10); // Set cursor position on OLED display

oled.println("Accident Alert!!!"); // Print accident alert message

oled.setCursor(20, 30); // Set cursor position on OLED display

oled.println("Send SMS To"); // Print send SMS message

oled.setCursor(2, 50); // Set cursor position on OLED display

oled.println("Family Member"); // Print message to family member

oled.sendBuffer(); // Send buffer to OLED display

sendSMS("+919382395550"); // Send SMS to family member

delay(6000); // Delay for 6 seconds

}

else {

    oled.clearBuffer(); // Clear OLED display buffer

    oled.setFont(u8g2_font_ncenB12_tf); // Set font for OLED display

    oled.setCursor(5, 20); // Set cursor position on OLED display

    oled.println("Safe Drive"); // Print safe drive message

    oled.setCursor(40, 60); // Set cursor position on OLED display

    oled.println("Save Life"); // Print save life message
```

```
oled.sendBuffer(); // Send buffer to OLED display

delay(10); // Delay for 10 milliseconds

}

delay(100); // Delay for 100 milliseconds

}

}

void Task2Code(void* parameter) {

while (1) { // Infinite loop

    ArduinoOTA.handle(); // Handle OTA updates

    int pir = digitalRead(p); // Read the PIR sensor value

    int alc = analogRead(al); // Read the alcohol sensor value

    int b = digitalRead(button); // Read the button value

    int v = alc / 10; // Convert alcohol sensor value to a smaller range

    Serial.println(pir); // Print the PIR sensor value to serial monitor

    Serial.println(v); // Print the converted alcohol sensor value to serial monitor

    // oled.clearBuffer(); // Clear the OLED display buffer (commented out)

    // oled.setFont(u8g2_font_ncenB12_tf); // Set the font for the OLED display (commented
out)

    // oled.setCursor(40, 40); // Set cursor position on OLED display (commented out)

    // oled.println(pir); // Print PIR sensor value on the OLED display (commented out)
```

```
// oled.sendBuffer(); // Send buffer to OLED display (commented out)

// delay(100); // Wait for 100 milliseconds (commented out)

if (pir == 1) { // Check if PIR sensor is triggered

    if (v <= 340) { // Check if alcohol sensor value is within safe range

        if (b == 1) { // Check if button is pressed

            analogWrite(EN1, 100); // Set motor speed

            digitalWrite(IN1, HIGH); // Set motor direction

            digitalWrite(IN2, LOW); // Set motor direction

            delay(100); // Wait for 100 milliseconds

            digitalWrite(t1, LOW); // Set ultrasonic sensor trigger low

            delayMicroseconds(5); // Wait for 5 microseconds

            digitalWrite(t1, HIGH); // Set ultrasonic sensor trigger high

            delayMicroseconds(20); // Wait for 20 microseconds

            digitalWrite(t1, LOW); // Set ultrasonic sensor trigger low

            duration = pulseIn(e1, HIGH); // Read pulse duration from ultrasonic sensor

            distance = (duration / 2) * 0.0343; // Calculate distance from pulse duration

            delay(100); // Wait for 100 milliseconds

            // Serial.println(distance); // Print distance to serial monitor (commented out)

            delay(100); // Wait for 100 milliseconds

            if (distance <= 5) { // Check if distance is less than or equal to 5 cm
```

```
digitalWrite(led, HIGH); // Turn on LED

delay(100); // Wait for 100 milliseconds

digitalWrite(IN3, LOW); // Stop motor

digitalWrite(IN4, LOW); // Stop motor

delay(100); // Wait for 100 milliseconds

}

else {

    digitalWrite(led, LOW); // Turn off LED

    delay(100); // Wait for 100 milliseconds

    analogWrite(EN2, 75); // Set motor speed

    digitalWrite(IN3, LOW); // Set motor direction

    digitalWrite(IN4, HIGH); // Set motor direction

    delay(100); // Wait for 100 milliseconds

}

}

else { // If button is not pressed

    analogWrite(EN1, 100); // Set motor speed

    digitalWrite(IN1, HIGH); // Set motor direction

    digitalWrite(IN2, LOW); // Set motor direction

    delay(100); // Wait for 100 milliseconds
```



```
digitalWrite(t2, LOW); // Set ultrasonic sensor trigger low

delayMicroseconds(5); // Wait for 5 microseconds

digitalWrite(t2, HIGH); // Set ultrasonic sensor trigger high

delayMicroseconds(20); // Wait for 20 microseconds

digitalWrite(t2, LOW); // Set ultrasonic sensor trigger low

duration = pulseIn(e2, HIGH); // Read pulse duration from ultrasonic sensor

distance = (duration / 2) * 0.0343; // Calculate distance from pulse duration

delay(100); // Wait for 100 milliseconds

// Serial.println(distance); // Print distance to serial monitor (commented out)

delay(100); // Wait for 100 milliseconds

if (distance <= 5) { // Check if distance is less than or equal to 5 cm

    digitalWrite(led, HIGH); // Turn on LED

    delay(100); // Wait for 100 milliseconds

    digitalWrite(IN3, LOW); // Stop motor

    digitalWrite(IN4, LOW); // Stop motor

    delay(100); // Wait for 100 milliseconds

}

else {

    digitalWrite(led, LOW); // Turn off LED

    analogWrite(EN2, 75); // Set motor speed
```

```
digitalWrite(IN4, LOW); // Set motor direction

digitalWrite(IN3, HIGH); // Set motor direction

delay(100); // Wait for 100 milliseconds

}

}

}

else { // If alcohol sensor value is not within safe range

    // sendSMS1("+918637333961","The Driver is drunk."); // Send SMS indicating driver
    is drunk (commented out)

    alsms("+918637333961"); // Send alcohol sensor SMS

    delay(6000); // Wait for 6 seconds

    // analogWrite(EN1, 150); // Set motor speed (commented out)

    // digitalWrite(IN1, LOW); // Stop motor (commented out)

    // digitalWrite(IN2, LOW); // Stop motor (commented out)

    // delay(100); // Wait for 100 milliseconds (commented out)


    digitalWrite(IN3, LOW); // Stop motor

    digitalWrite(IN4, LOW); // Stop motor

    analogWrite(EN1, 100); // Set motor speed

    digitalWrite(IN1, LOW); // Stop motor
```

```

    digitalWrite(IN2, LOW); // Stop motor

    delay(100); // Wait for 100 milliseconds

}

}

else { // If PIR sensor is not triggered

    // sendSMS1("+918637333961","The Driver is drunk."); // Send SMS indicating driver
    is drunk (commented out)

    digitalWrite(IN3, LOW); // Stop motor

    digitalWrite(IN4, LOW); // Stop motor

    delay(100); // Wait for 100 milliseconds

    analogWrite(EN1, 100); // Set motor speed

    digitalWrite(IN1, LOW); // Stop motor

    digitalWrite(IN2, LOW); // Stop motor

    delay(100); // Wait for 100 milliseconds

}

delay(100); // Wait for 100 milliseconds

}

}

void sendSMS(const char* phoneNumber) {

```

```
String locationLink = "Accident detected!!! -> https://maps.google.com/?q="; // Create the
message with accident alert and link to location
```

```
locationLink += String(latitude, 6) + "," + String(longitude, 6); // Append the latitude and
longitude to the location link
```

```
SerialGSM.println("AT+CMGF=1"); // Set GSM module to text mode
```

```
delay(500); // Wait for 500 milliseconds
```

```
SerialGSM.print("AT+CMGS=\""); // Start the AT command to send SMS
```

```
SerialGSM.print(phoneNumber); // Insert the phone number to send SMS
```

```
SerialGSM.println(""); // End the AT command to send SMS
```

```
delay(500); // Wait for 500 milliseconds
```

```
SerialGSM.print(locationLink); // Send the location link as the SMS message
```

```
delay(500); // Wait for 500 milliseconds
```

```
SerialGSM.write(26); // Send the ASCII code for CTRL+Z to indicate the end of the
message
```

```
delay(500); // Wait for 500 milliseconds
```

```
Serial.print("Sent SMS to: "); // Print message to serial monitor
```

```
Serial.println(phoneNumber); // Print phone number to serial monitor
```

```
Serial.print("Message: "); // Print message to serial monitor
```

```
Serial.println(locationLink); // Print location link to serial monitor
```

```
}
```

```
void alsms(const char* phoneNumber) {
```

```
String locationLink = "The Driver is drunk."; // Create the message indicating the driver is
drunk

locationLink += "Location -> https://maps.google.com/?q="; // Add the link to location

locationLink += String(latitude, 6) + "," + String(longitude, 6); // Append the latitude and
longitude to the location link

SerialGSM.println("AT+CMGF=1"); // Set GSM module to text mode

delay(500); // Wait for 500 milliseconds

SerialGSM.print("AT+CMGS=\""); // Start the AT command to send SMS

SerialGSM.print(phoneNumber); // Insert the phone number to send SMS

SerialGSM.println(""); // End the AT command to send SMS

delay(500); // Wait for 500 milliseconds

SerialGSM.print(locationLink); // Send the location link as the SMS message

delay(500); // Wait for 500 milliseconds

SerialGSM.write(26); // Send the ASCII code for CTRL+Z to indicate the end of the
message

delay(500); // Wait for 500 milliseconds

Serial.print("Sent SMS to: "); // Print message to serial monitor

Serial.println(phoneNumber); // Print phone number to serial monitor

Serial.print("Message: "); // Print message to serial monitor

Serial.println(locationLink); // Print location link to serial monitor

}
```

## ❖ CODE EXPLANATION

The code in the provided Table outlines a robust accident detection and alert system that integrates multiple sensors and communication modules using an ESP32 microcontroller. It employs advanced features like FreeRTOS for multitasking and OTA (Over-The-Air) updates for remote firmware updates. The key components include an MPU6050 accelerometer and gyroscope, GPS module, GSM module, PIR motion sensor, alcohol sensor, ultrasonic sensors, an OLED display, and motor control circuitry.

In the setup function, the system initializes all hardware components, establishes a Wi-Fi connection, configures the MPU6050 sensor, and prepares the system for OTA updates. The OLED display provides real-time status updates during initialization and operation. Two FreeRTOS tasks are created: one for monitoring gyroscopic and GPS data for accident detection, and another for handling additional sensors like the PIR motion detector, alcohol sensor, and ultrasonic sensors.

Task 1 monitors the MPU6050 gyroscope for sudden movements indicative of an accident. If an accident is detected, the system fetches GPS coordinates and sends SMS alerts to predefined contacts such as hospitals, police, and family members. These alerts include a Google Maps link to the accident location, providing precise coordinates for rescue services.

Task 2 manages real-time monitoring of PIR and alcohol sensors to ensure safe vehicle operation. Additionally, it integrates ultrasonic sensors to prevent collisions by measuring the distance to obstacles in front of or behind the vehicle. If the distance is below a safety threshold, the system takes preventive actions such as stopping the motor and activating a warning LED. The task also uses the OLED display to provide feedback, such as safety messages or alerts when unsafe conditions are detected.

The sendSMS and alsms functions handle transmitting accident and safety alerts via the GSM module. They generate text messages with location links and send them to the specified phone numbers. By combining real-time data monitoring, collision prevention, and communication capabilities, this system enhances safety and responsiveness in critical scenarios.

## EXPENSES

Costing					
System Circuit Modules, Connectors & Wires Cost					
SL NO.	Component Name	QTY.	Rate	Amount	Delivery Charge
1	ESP32 (38 Pin)	1	₹ 339.00	₹ 339.00	₹ 125.00
2	PIR	1	₹ 62.00	₹ 62.00	₹ 125.00
3	MQ3	1	₹ 100.00	₹ 100.00	₹ 59.00
4	MPU6050	1	₹ 100.00	₹ 100.00	₹ 63.00
5	ULTRASONIC SENSOR	2	₹ 125.00	₹ 250.00	₹ 125.00
6	1.3inch OLED Display	1	₹ 220.00	₹ 220.00	
7	L293D	1	₹ 119.00	₹ 119.00	
8	GPS (NEO 6M)	1	₹ 198.00	₹ 198.00	
9	GSM (SIM900A)	1	₹ 600.00	₹ 600.00	
10	TP4056	1	₹ 16.00	₹ 16.00	
11	IC7805	1	₹ 10.00	₹ 10.00	
12	1uF Capacitor	2	₹ 2.00	₹ 4.00	
13	Buzzer	1	₹ 14.00	₹ 14.00	
14	LED	1	₹ 1.30	₹ 1.30	
15	DC switch	1	₹ 10.00	₹ 10.00	
16	DC Mottor	2	₹ 50.00	₹ 100.00	
17	12V DC FAN	2	₹ 75.00	₹ 150.00	
18	BETTERY(18560)	2	₹ 70.00	₹ 140.00	
19	AMS1117 5V Step-Down Power Supply Module	1	₹ 19.00	₹ 19.00	
20	Step-down (buck) voltage regulator (12V dc to 3.3V, 5V, 12V dc Converter)	1	₹ 39.00	₹ 39.00	
21	Jumper Wire	100	₹ 1.00	₹ 100.00	
22	40x1 Male Berg Strip	2	₹ 7.00	₹ 14.00	
23	40x1 Female Berg Strip	1	₹ 8.00	₹ 8.00	
24	18650 SMD/SMT Dual Battery Holder	2	₹ 65.00	₹ 130.00	
25	PCB Board	1	₹ 15.00	₹ 15.00	
26	DB9 Female Right Angle Connector	1	₹ 14.00	₹ 14.00	
27	DB9 Male Straight Connector	1	₹ 14.00	₹ 14.00	
SUB TOTAL				₹	2,786.30
TOTAL DELIVERY CHARGE				₹	497.00
GRAND TOTAL				₹	3,283.30

Decoration Cost				
SL NO.	Component Name	QTY.	Rate	Amount
1	PVC (8X10)	1	₹ 80.00	₹ 80.00
2	PVC (2X6)	1	₹ 60.00	₹ 60.00
3	PVC (4X4)	1	₹ 40.00	₹ 40.00
4	Glue Stick	6	₹ 15.00	₹ 90.00
5	Paper	2	₹ 5.00	₹ 10.00
6	Glue	2	₹ 5.00	₹ 10.00
7	PVC Pipe (12inch)	1	₹ 20.00	₹ 20.00
8	2x2 Plywood Board	1	₹ 60.00	₹ 60.00
9	Screw & Nuts	10	₹ 2.00	₹ 20.00
<b>TOTAL Price</b>				₹ 390.00



## FUTURE SCOPE

The Smart Accident Prevention, Detection, and Reporting System (SAPDRS) has immense potential for integration into real-world vehicles, ensuring a safer driving experience for millions. The design, as depicted in the circuit layout, provides the foundation for a comprehensive and reliable system. When integrated with a car, the SAPDRS would work seamlessly with the car's existing electronics, including its control unit, power management systems, and sensor arrays. By utilizing ultrasonic sensors for obstacle detection and GSM-GPS modules for accurate location reporting, the SAPDRS can act as the central nervous system of accident prevention. It would communicate with the car's engine control module to ensure preventive measures such as automatic braking or speed reduction when detecting imminent collisions. The DC motors and fans illustrated in the design can be adapted to control real-world functions, such as airflow systems for engine temperature management or the actuation of safety mechanisms like airbags. The embedded software within the ESP32 microcontroller would act as the brain, interpreting sensor inputs and executing commands in real-time.

Furthermore, the modular nature of the SAPDRS makes it scalable for modern vehicles, including electric and hybrid cars. The integration of sensors such as the MPU6050 accelerometer and gyroscope ensures accurate real-time data about vehicle orientation and dynamics, enabling precise accident detection. Meanwhile, the GSM and GPS modules play a critical role in reporting accidents to emergency services and providing live updates to authorities or family members. The transition from this prototype to a car-ready design would involve developing automotive-grade components that are rugged, heat-resistant, and weatherproof. In addition, the power system shown here could be replaced by the car's battery management system, ensuring a constant and uninterrupted power supply. When fully implemented, SAPDRS would align with advanced driver-assistance systems (ADAS) to form a cohesive safety ecosystem, creating smarter, more responsive vehicles.

The SAPDRS integration with real cars opens doors to its compatibility with autonomous driving technologies. Autonomous vehicles already employ a range of cameras and LiDAR systems to analyse their surroundings, and SAPDRS could act as an additional safety layer. For instance, its ultrasonic sensors could provide redundancy in obstacle detection, minimizing the chances of sensor blind spots. The ability to detect and prevent accidents before they occur will enhance public trust in smart car technologies. Additionally, SAPDRS could be

linked with vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication networks, sharing accident or hazard information with nearby vehicles and traffic management systems. This makes SAPDRS a pivotal innovation, not just for individual vehicle safety but also for the larger goal of intelligent transportation systems (ITS).

To meet the rapidly evolving demands of automotive technology, SAPDRS can be upgraded with advanced components, significantly enhancing its functionality and reliability. The current design employs ultrasonic sensors for obstacle detection, which, while effective, can be supplemented or replaced with LiDAR (Light Detection and Ranging) sensors for more precise and long-range detection of objects. LiDAR systems generate high-resolution 3D maps of the vehicle's surroundings, enabling advanced decision-making in both urban and highway driving scenarios. Similarly, replacing the MPU6050 accelerometer with more sophisticated inertial measurement units (IMUs) could improve the system's ability to detect sharp movements and vibrations indicative of accidents. Infrared cameras or thermal sensors could also be integrated to detect pedestrians or animals in low-visibility conditions, further enhancing the system's accident-prevention capabilities.

Actuators play a critical role in executing commands from the control unit. While the current design uses basic DC motors, these could be upgraded to servo motors for more precise control in applications such as steering corrections or deploying retractable safety mechanisms. The 12V DC fans depicted in the design could also be replaced with electronically controlled actuators for better energy efficiency and adaptability. In terms of reporting, the system could transition from GSM-based communication to 5G-enabled modules, ensuring faster and more reliable transmission of accident data to emergency responders. Additionally, a cloud-based interface could be introduced to store accident data, which could later be analysed for understanding accident trends and improving future designs. Integrating advanced processors like NVIDIA Jetson Nano or Raspberry Pi Compute Modules alongside ESP32 would enable machine learning capabilities, allowing the system to learn and improve over time based on the data it gathers.

The power supply system can also be enhanced to make SAPDRS more adaptable to real-world car environments. Instead of relying on 18650 batteries and regulators, the system could draw power directly from the car's alternator or regenerative braking system. Moreover, introducing energy harvesting mechanisms, such as solar panels or piezoelectric systems, could make SAPDRS energy-efficient and sustainable. The inclusion of CAN (Controller Area

Network) bus compatibility would ensure seamless communication between SAPDRS and the car's onboard systems. Finally, the physical design of SAPDRS could be miniaturized, using printed circuit boards (PCBs) with advanced surface-mount technology (SMT) to fit neatly into compact car environments. These upgrades will not only make SAPDRS more effective but also ensure its compatibility with the cutting-edge technologies found in modern vehicles.

The future vision for SAPDRS lies in its potential to evolve into a fully autonomous safety system that can revolutionize road safety and emergency response frameworks. In the coming years, SAPDRS could be integrated into connected car ecosystems, enabling vehicles to communicate with each other, road infrastructure, and centralized traffic management systems. This could lead to the creation of accident-free corridors, where SAPDRS-equipped vehicles work in harmony to prevent collisions through predictive algorithms and shared data. The adoption of AI and machine learning technologies could further enhance the system's capabilities, allowing it to predict risky driving behaviours and alert drivers in advance. Additionally, SAPDRS could become a part of international safety standards, paving the way for its mandatory implementation in all vehicles as a regulatory requirement.

Looking further ahead, SAPDRS could be integrated with futuristic transportation modes, such as flying cars or hyperloop systems, ensuring the safety of these next-generation technologies. It could also play a crucial role in smart cities, where it would act as an intermediary between vehicles and citywide emergency response systems. For example, upon detecting an accident, SAPDRS could immediately alert nearby medical drones or autonomous ambulances, ensuring rapid assistance. Its potential to gather and analyse data from accidents could also contribute to urban planning, helping authorities identify accident-prone areas and implement targeted safety measures. SAPDRS could also be integrated into wearable technology, such as smart helmets for bikers, to provide an added layer of personal safety.

As SAPDRS evolves, the development of alternative sensors and actuators will play a critical role in enhancing its capabilities. Quantum sensors could provide unparalleled precision in detecting changes in the vehicle's environment, while bio-sensors embedded in seats or steering wheels could monitor the driver's health and alert them in case of fatigue or sudden medical issues. Actuators powered by artificial muscles could replace traditional mechanical systems, offering faster response times and greater reliability. Moreover, SAPDRS could explore integration with renewable energy sources, such as hydrogen fuel cells, to ensure sustainability. In summary, the SAPDRS project holds immense promise for shaping the future

of transportation safety, offering a blend of innovation, efficiency, and reliability that could save countless lives and redefine the way we approach vehicle safety.

## CONCLUSION

The "Smart Accident Prevention, Detection, and Reporting System (SAPDRS)" is a pioneering advancement in road safety technology. Utilizing cutting-edge sensors, microcontrollers, and communication technologies, SAPDRS provides a comprehensive solution for preventing accidents, detecting them swiftly, and ensuring effective reporting. The system integrates an MPU6050 accelerometer and gyroscope, GPS and GSM modules, PIR motion sensors, alcohol sensors, ultrasonic sensors, and an OLED display, all managed by an ESP32 microcontroller.

The preventive module of SAPDRS enhances driving safety through real-time monitoring and alerts. Ultrasonic sensors measure distances to obstacles, and the system can take preventive actions like stopping the vehicle if a collision risk is detected. The detection module ensures that any accident is promptly identified using the MPU6050 sensor, which monitors vehicle movements. Upon detecting a collision, the system retrieves GPS coordinates and sends SMS alerts with location details to predefined contacts, including emergency services.

Real-time monitoring of the driver's condition is achieved through the PIR motion detector and alcohol sensor. If alcohol is detected, the system can disable the vehicle's engine to prevent unsafe operation. The reporting functionality facilitates immediate communication with emergency services, providing precise accident location details for quick response. The OLED display provides feedback to the driver, showing safety messages and system status updates.

SAPDRS showcases the potential of embedded systems and IoT in enhancing transportation safety. By combining real-time data monitoring, collision prevention, and efficient communication, SAPDRS reduces accident risks and ensures rapid response, paving the way for a safer transportation ecosystem.

## REFERENCES

### 1. Datasheet:

- **ESP32:**
  - ⇒ [https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf)
- **MQ3 Gas Sensor:**
  - ⇒ <https://www.sparkfun.com/datasheets/Sensors/MQ-3.pdf>
- **Ultrasonic Sensor (HC-SR04):**
  - ⇒ <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>
- **PIR Sensor:**
  - ⇒ <https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf>
- **MPU6050 Accelerometer and Gyroscope:**
  - ⇒ <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>
- **L293D Motor Driver:**
  - ⇒ <https://www.ti.com/lit/ds/symlink/l293.pdf>

### 2. ESP32 Official Documentation:

- ⇒ <https://www.espressif.com/en/support/documents/technical-documents>

### 3. Arduino IDE: <https://www.arduino.cc/en/software>

### 4. Interfacing:

- **MQ3 Gas Sensor:**
  - ⇒ <https://microcontrollerslab.com/mq3-alcohol-sensor-arduino-tutorial/>
- **Ultrasonic Sensor (HC-SR04):**
  - ⇒ <https://randomnerdtutorials.com/esp32-hc-sr04-ultrasonic-arduino/>
- **PIR Sensor:**
  - ⇒ <https://www.electronicwings.com/esp32/pir-sensor-interfacing-with-esp32>
- **GPS Module (e.g., NEO-6M):**
  - ⇒ <https://randomnerdtutorials.com/esp32-neo-6m-gps-module-arduino/>
- **GSM Module (e.g., SIM900A):**
  - ⇒ <https://www.electronicwings.com/sensors-modules/sim900a-gsmgprs-module>

- **MPU6050 Accelerometer and Gyroscope:**

- ⇒ <https://www.electronicwings.com/esp32/mpu6050-gyroscope-interfacing-with-esp32>

- **L293D Motor Driver:**

- ⇒ <https://www.electronicwings.com/esp32/dc-motor-interfacing-with-esp32>

**5. AI –**

**ChatGPT –** <https://chatgpt.com/>

**Copilot –** <https://copilot.microsoft.com/chats/ssMY6TaGH82rx8z5ZiLtr>