Project Report

On

"Robotic Control System Based on Object Detection & Accidental Alert System"

Bachelor of Engineering

Electronics & Telecommunication Engineering

Sant Gadge Baba Amravati University, Amravati.

Submitted by

Arjun S. Mishra

Pranav A. Kale

Prathmesh K. Joshi

Ritesh W. Hande

Guided by

Prof. A. S. Utane

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING,

PROF. RAM MEGHE INSTITUTE OF TECHNOLOGY & RESEARCH,

BADNERA-AMRAVATI

2023-2024

CERTIFICATE

This is to certify that the project report entitled

"Robotic Control System Based on Object Detection & Accidental Alert System"

has been successfully completed by

Arjun S. Mishra

Pranav A. Kale

Prathmesh K. Joshi

Ritesh W. Hande

in satisfactory manner as a partial fulfilment of
Degree of Bachelor of Engineering in
(Electronics & Telecommunication Engineering)
Sant Gadge baba Amravati University, Amravati
during the academic year 2023-2024

Prof. A. S. Utane
Guide
Department of EXTC Engineering.
PRM Institute of Technology & Research,
Badnera

Dr. S. V. Pattalwar
Head,
Department of EXTC Engineering.
PRM Institute of Technology & Research,
Badnera

External Examiner

Vidarbha Youth Welfare Society's

Prof. Ram Meghe Institute of Technology & Research, Badnera-Amravati.

Department of Electronics & Telecommunication Engineering.

B.E. (Electronics and Telecommunication)

Certificate of Originality

This is to certify that, I am responsible for the work submitted in this project. The original work is my own except as specified in the references & acknowledgement. The original work contained herein have not been undertaken or done by unspecified sources or persons.

Arjun S. Mishra

Pranav A. Kale

Prathmesh K. Joshi

Ritesh W. Hande

ACKNOWLEDGEMENT

It is my moral duty & responsibility to be loyal & grateful to those who have shown me the path by throwing their knowledged rays during the Project.

It is worth mentioning here that as a guide ***Prof. A. S. Utane*** *has encouraged me time to time during the Project. He/she all the while was my co-rider. He/she is the reinforcement of my dissertation. Let me be honest to pay him/her utmost regards for his guidance to which my project proved to be a successful one.*

Words are insufficient to show my thankfulness to HOD ***Dr. S. V. Pattalwar*** *who at every point showed me the telescopic way in respect of my project.*

I am grateful to all the ***authors*** *of books & papers which have been referred for this Project. Last but not the least, I am thankful to* ***all teaching, non-teaching staff and everyone*** *who directly or indirectly helped me for the completion of this project.*

Arjun S. Mishra

Pranav A. Kale

Prathmesh K, Joshi

Ritesh W. Hande

ABSTRACT

The integration of robotic control systems with advanced object detection and crash alert mechanisms represents a significant advancement in the realm of robotics and automation. This project focuses on developing a sophisticated robotic control system that leverages cutting-edge object detection algorithms to enhance the robot's environmental awareness and safety. Utilizing computer vision techniques, the system enables the robot to identify and localize various objects in its vicinity, facilitating intelligent navigation and interaction in dynamic environments.

In addition to object detection, the proposed system incorporates a robust Accidental Detection designed to preemptively identify potential collision risks. By continuously monitoring the robot's surroundings and analyzing real-time data, the Accidental Detection can promptly detect obstacles or hazards, triggering immediate corrective actions to avoid collisions and ensure the robot's safe operation.

The control architecture integrates these advanced functionalities to enable seamless coordination between object detection, crash alerting, and robotic navigation. Through a combination of sensor fusion, machine learning, and adaptive control strategies, the system achieves enhanced situational awareness and adaptive decision-making capabilities, allowing the robot to navigate complex environments with increased efficiency and safety.

Furthermore, the project explores the implementation of a user-friendly interface to facilitate easy configuration and customization of the robotic control parameters, enabling operators to adapt the system to specific application requirements effectively. The proposed robotic control system offers promising potential for a wide range of applications, including industrial automation, autonomous vehicles, and service robots, paving the way for safer and more intelligent robotic systems in various domains.

CONTENTS

Chapter 1

INTRODUCTION

In recent years, the integration of robotics and automation technologies has revolutionized various industries, ranging from manufacturing and logistics to healthcare and entertainment. As robotic systems become increasingly prevalent in our daily lives, there is a growing demand for advanced control mechanisms that can enhance their capabilities, safety, and efficiency. One of the critical challenges in developing intelligent robotic systems lies in enabling them to interact seamlessly with their environment, adapt to dynamic changes, and avoid potential hazards autonomously.

Object detection and collision avoidance are fundamental aspects that significantly influence the performance and safety of robotic systems operating in complex and unstructured environments. Traditional robotic control methods often rely on predefined paths or manual intervention, limiting their adaptability and responsiveness to unexpected situations. With the advent of sophisticated sensor technologies and machine learning algorithms, there is an opportunity to develop more intelligent and proactive robotic control systems capable of real-time object recognition and collision prediction.

The primary objective of this project is to design and implement a robotic control system that leverages advanced object detection techniques and crash alert mechanisms to enhance the robot's navigation and safety capabilities. By integrating state-of-the-art computer vision algorithms, the system aims to provide the robot with the ability to identify and localize various objects, such as obstacles, pedestrians, and other vehicles, in its surrounding environment. This enhanced environmental awareness will enable the robot to make informed decisions and navigate complex terrains more effectively, reducing the risk of collisions and enhancing operational efficiency.

By continuously monitoring the robot's surroundings and analyzing real-time sensor data, the Accidental Detection will provide an additional layer of safety, ensuring the robot's safe operation in dynamic and unpredictable environments.

Chapter 2

LITERATURE REVIEW

1. Al-Bahri Mahmood et.al (2021) in their Research "Object Recognition for Organizing the Movement of Self-Driving Car" they stated that Today there is a revolution in the automotive industry. Cars are becoming self-driving with advanced sensors, cameras, and recognition algorithms. Algorithms and scenarios are evolving and improving every day, but it is too early for self-driving cars to go out on public roads. The work of recognition algorithms is far from ideal. For the correct and synchronized operation of all elements of an unmanned vehicle, a person needs to transfer all his intellectual experience to the computer systems of the vehicle. The developers are working to ensure that the car can see and understand what is happening around it. Such cars are already driving on the roads in test mode with a pilot, receiving a huge amount of information and learning

2. Parag Kapre et.al (2022) in their Research “Voice Controlled Car Assistant System and Automatic Breaking System” they stated that the purpose of this project is to build a Voice Controlled Car Assistant System and Automatic Breaking System. A Voice Controlled Car is an advanced robotic vehicle that can be operated by the power of voice commands. It is based on an Arduino microcontroller, motor drivers, and a Bluetooth module. The Arduino hardware is an open-source micro-controller kit used to build digital devices. In our project, we will design the hardware of the Voice Controlled Robotic Car first, then use our previous knowledge of programming to code the entire work. The code will then be simulated on IDE software, and then interfaced with the hardware. An android device with a Bluetooth application is used to control the control unit in coordination with the Bluetooth device, and a Bluetooth module is used to capture and read the voice commands.

   We choose this project because automation has become a significant part of our lives and also has a broad range of applications in the engineering field. Automation plays a vital role in the development of new technology.

3. K.M. Tousif et.al (2022) in their Research “Bluetooth Control Car with Arduino” they Stated that they make the use of Bluetooth technology to control our machine car. We do not call this a robot as this device does not have any sensors. Thereby, senseless robots are machines. The project aims are to design a Bluetooth control Arduino car and write a program into the Arduino microprocessor. Arduino car contains an Arduino microcontroller with basic mobility features. In this project, we make use of Bluetooth technology to control our machine car. After doing this only we can say that we have been able to create as per our goal described. The device can be controlled by any smart device with android. The major reason for using a Bluetooth-based tech is that we can change the remote anytime-mobile phones, tablets, and laptops and physical barriers like walls or doors do not affect the car controls.

4. Daxton Givan et.al (2022) in their Research “Application of Lane Navigation and Object Detection in a Deep-Learning Self-driving Car” they stated that This project develops lane navigation and object detection functions using an RC car and a basic CNN Deep learning algorithm, which shows a great potential for object detection. In addition, it discusses what effects the different dataset size and the presence of obstacles have on the performance through the various case studies. The self-driving car is one of the technologies with the highest potential to become commercialized in the near future, but currently a great technological barrier prevents the existence of fully self-driving cars, allowing for partial self-driving and limited assistance to the driver.

   When considering this technological trend of self-driving cars, even if this project cannot overcome all of the technical limitations and barriers, it is very significant from a pedagogical aspect by expanding the scope of internship and offering the opportunity to create a practical application.

5. Shridhar Devamane et.al (2022) in their Research " Deep Learning Based Approaches for Vehicle Make and Model Recognition" they stated that Object detection is largely used in the area of computer vision and is critical for variety of applications. During the development of half a century, object detection methods have been continuously developed, and generated numerous approaches which obtained promising achievements. At present, the approach of object detection has been largely evolved into two categories which are traditional machine learning methods utilizing varied computer vision techniques and deep learning methods. In spite of this evolution, accurate implementation of Vehicle Make and Model Recognition (VMMR) is exacting owing to alike (kindred) appearance of different models of vehicles. Therefore this paper presents machine as well as deep learning techniques along with transfer learning models for car detection where the classification is generally at the extent of Make, Model and Year. In this paper, firstly the existing techniques centered on traditional machine learning are introduced and summarized.

6. Karuna, Gotlur & Kumar, Ram et.al (2023) in their Research "Motorcycle Crash Detection and Alert System using IoT" they stated that Motorcycle travel is considered the most dangerous mode of transport in the world. Reports suggest that the fatality rate of motorcycles is 212.7 deaths for every million miles travelled on motorcycles. Unlike other forms of travel like cars, buses, etc, motorcycles expose the rider to their surroundings. In cars, the frame protects the driver from hitting the road or falling out of the car. But motorcycles do not have such a possibility. Therefore, the best way to minimize fatalities in accidents is to have an alert system that can alert the emergency services when it detects an imminent crash. This is where the motorcycle crash detection and alert system comes into the picture. It uses the MPU6050 Multi-axes accelerometer to detect when the motorcycle falls to its side. It sends the impact parameters to Firebase cloud and if the values meet the crash criteria, it sends an alert to the emergency contacts as well as to the emergency response services, who can then act according to it.

7. Hallan, Akul & Rawat, Shreya et.al (2020) in their Research “Vehicle Accidental Detection” they stated that The ideation of this system is based on the creation of an automatic crash detection and notification system based on GSM and GPS. The objective of the project taken up is to develop an Accident Alert System based on a GPS, GSM, and Accelerometer/ Vibrational/ Shock Sensors which we further intend to deploy with a Bluetooth Module (Using a lower range Bluetooth range). Our project shall contribute towards the saving of multiple lives which are lost due to accidents, and provide them with quick and better medical support.

CHAPTER 3

METHODOLOGY

3.1 System Block Diagram

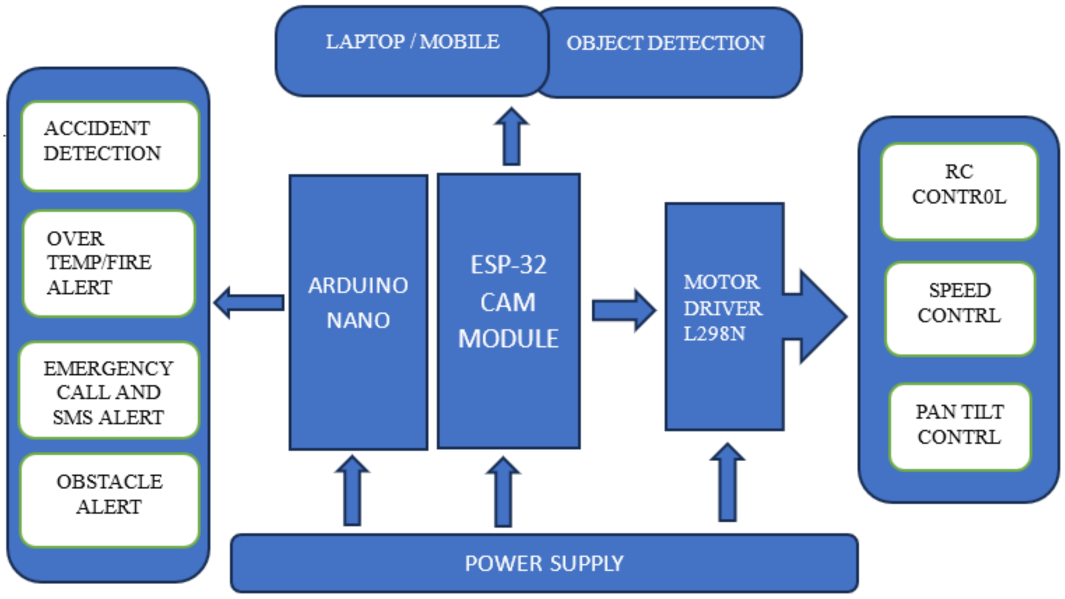


Figure 1 : System Block Diagram

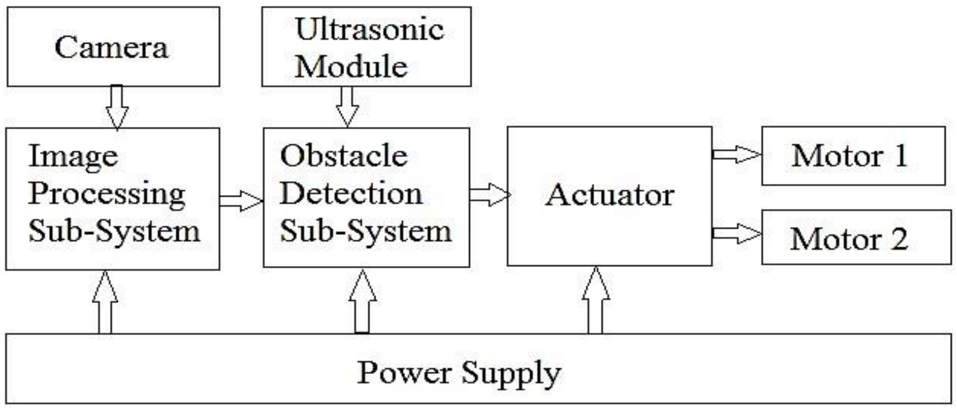


Figure 2 : Accidental Alert System

3.2 Description

The proposed robotic control system integrates advanced object detection and crash alert technologies to enhance the capabilities and safety of robotic platforms in various applications. Utilizing state-of-the-art computer vision algorithms, the system enables the robot to perceive and understand its environment effectively, identifying obstacles, landmarks, and other relevant entities through onboard cameras and sensors.

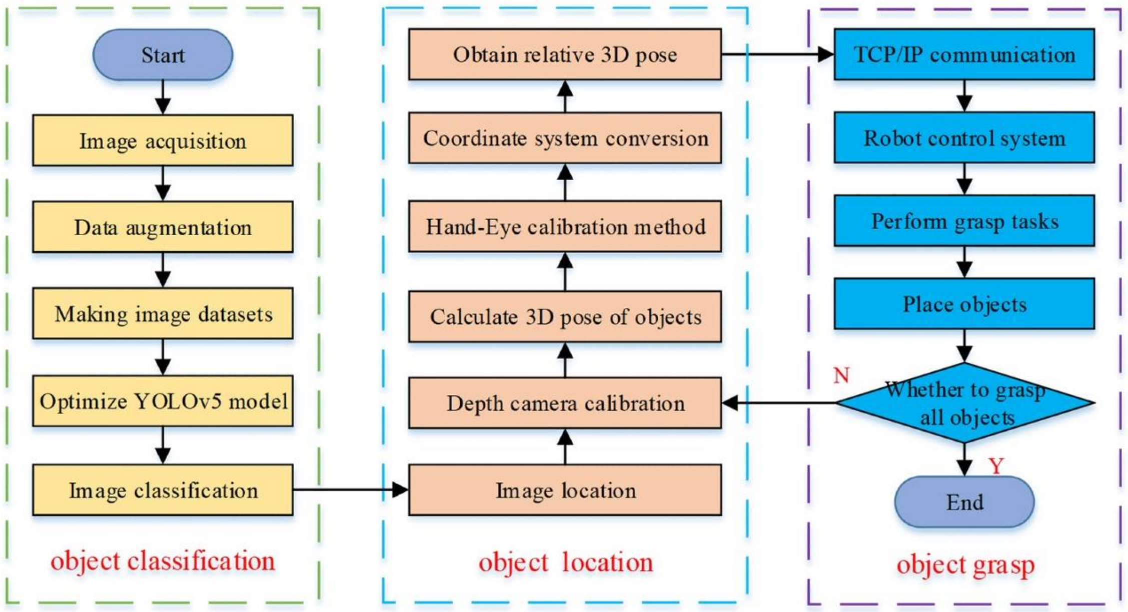
Object detection enhances the robot's situational awareness, enabling intelligent navigation and interaction in dynamic environments. The system's robust crash alert mechanism preemptively identifies potential collision risks, utilizing real-time sensor data and predictive analytics to trigger immediate corrective actions, ensuring safe operation and minimizing accident risks.

Designed with modularity and scalability in mind, the control architecture facilitates seamless integration with diverse robotic platforms and configurations. Leveraging sensor fusion techniques, machine learning algorithms, and adaptive control strategies, the system adapts to different operating conditions and application requirements, allowing operators to customize control parameters and optimize performance.

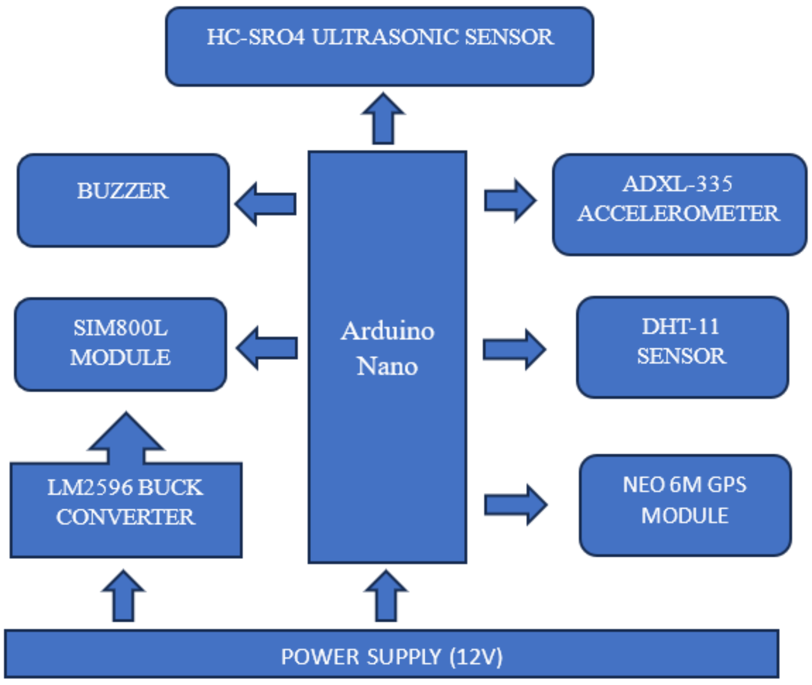
Additionally, the system features an intuitive user interface for easy configuration, monitoring, and control, providing operators with real-time feedback and visualization tools to make informed decisions and ensure smooth operation. Overall, the proposed control system represents a significant advancement in robotic automation, offering a comprehensive solution for intelligent navigation, object detection, and collision avoidance in various industries.

3.3 Flowchart of Implementation

1. Yolo v5 algorithm



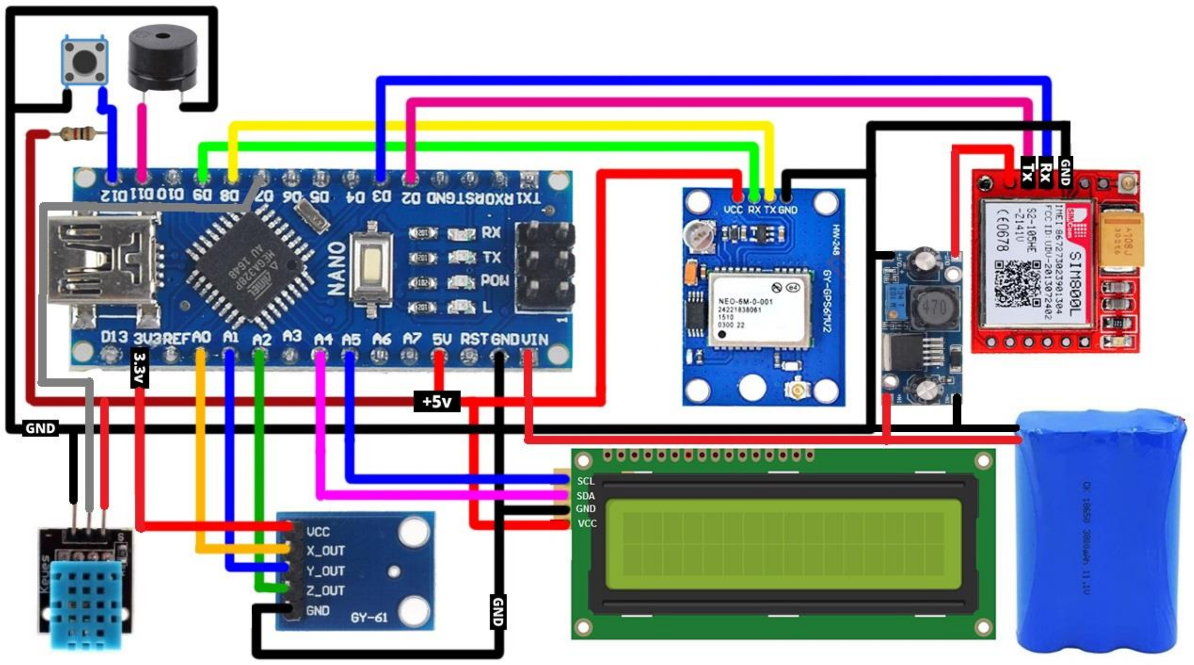
2. Accidental Alert System Flowchart



Chapter 4

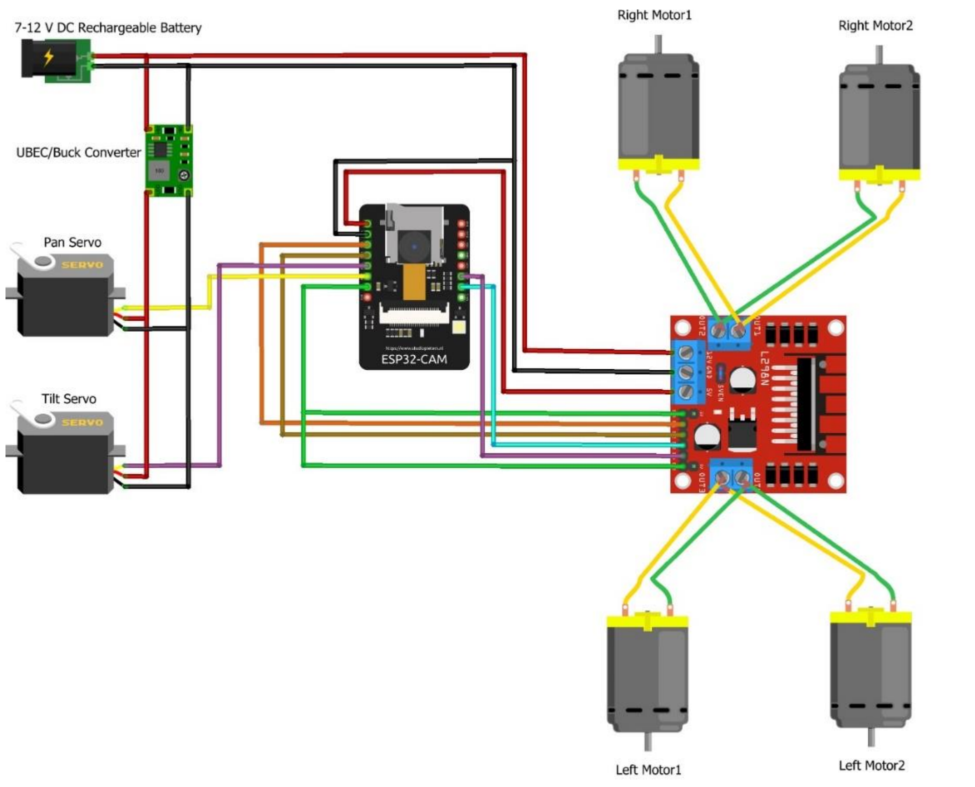
Implementation

4.1 Circuit Diagram of Accidental Detection



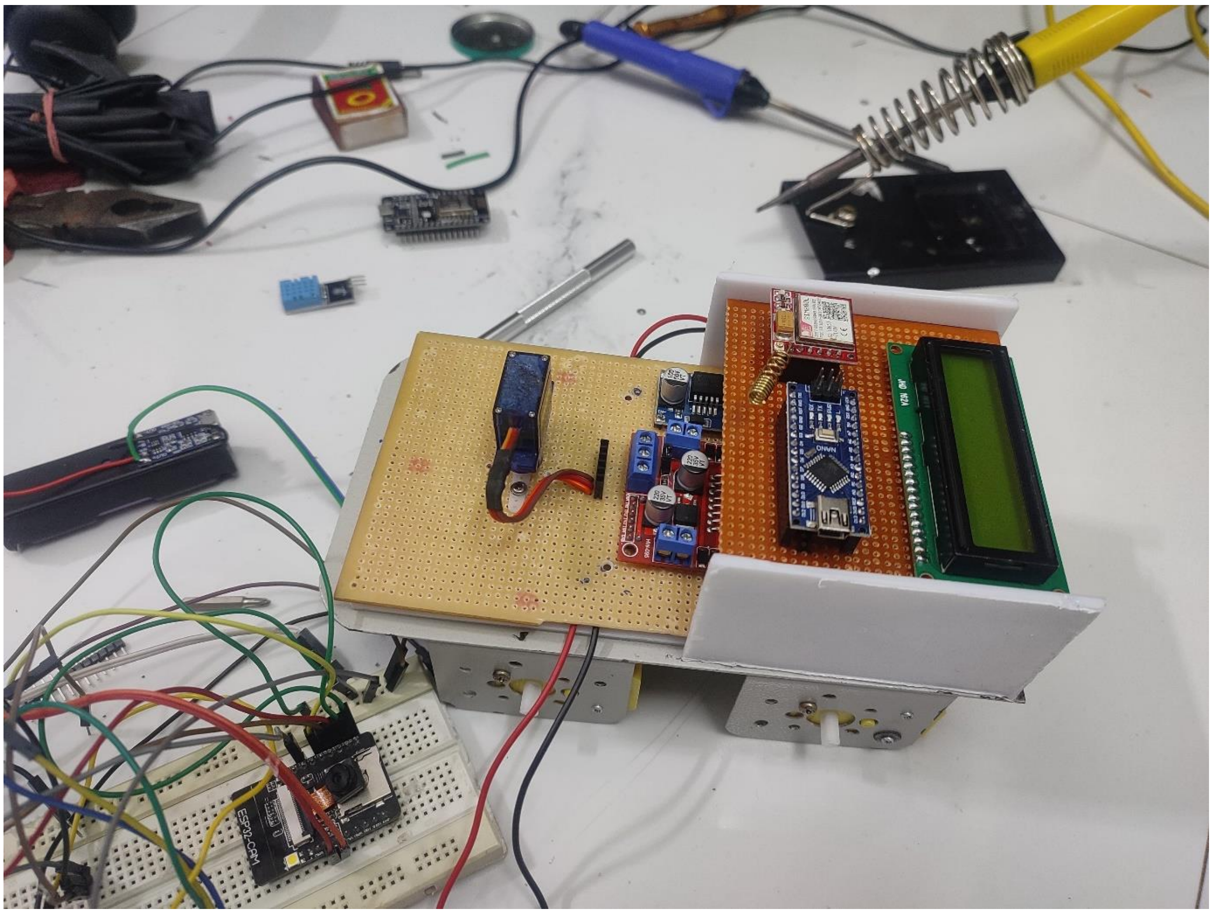
An Accidental Alert System plays a critical role in ensuring the robot's safety. It typically comprises sensors like accelerometers or collision sensors, a microcontroller for data processing, and an alert mechanism such as a buzzer or LED. Integrated into the project, it continuously monitors the robot's surroundings for potential accidents or collisions. Upon detection, it triggers alerts to notify users or autonomously halts the robot's movement to prevent damage or injury.

4.2 Pan-Tilt Circuit Diagram



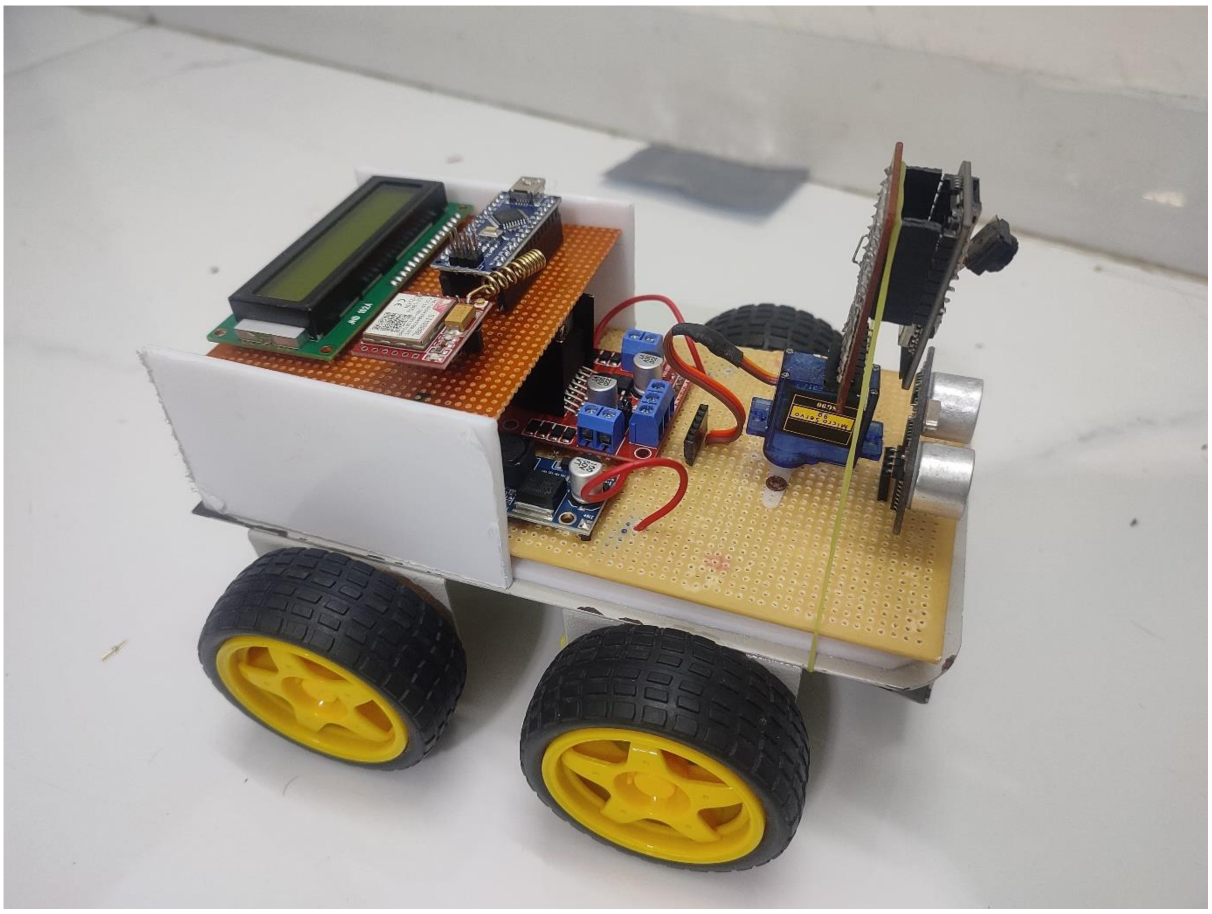
A Pan-Tilt circuit enables the orientation adjustment of sensors or cameras to focus on detected objects. Integrated with object detection sensors, it facilitates real-time tracking of objects within the robot's field of view. Additionally, it collaborates with the accidental alert system, ensuring the robot's safety by swiftly reorienting sensors away from detected obstacles or triggering alerts to prevent collisions.

4.3 Actual Pictures



We have utilized the ESP32 CAM with the L298N motor driver for monitoring and controlling the bot. This allows for controlling its speed, Headlight brightness, and the camera's pan-tilt position via the IP address generated by the ESP32 CAM on a web server, providing a better user interface. Additionally, we have integrated object detection into the web server for enhanced functionality.

To further enhance the bot's capabilities, we have incorporated an Arduino Nano and an ultrasonic sensor. This setup notifies the user with a beep when the bot approaches an obstacle. Furthermore, we have integrated a SIM800L and a 6M GPS module to provide call and SMS alerts in case of accidental behavior by the bot. These alerts are triggered by sensors such as the ADXL335 accelerometer for Accidental detection and the DHT11 sensor for monitoring high temperature and humidity, which helps in detecting fire and other emergencies, ensuring the safety and improved performance of the bot.



Chapter 5

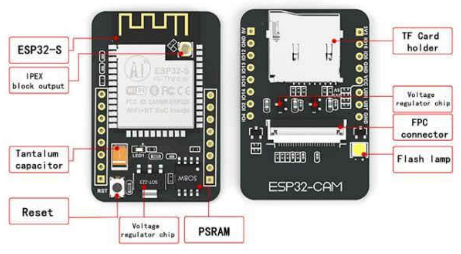
COMPONENT DESCRIPTION

5.1 HARDWARE REQUIREMENTS

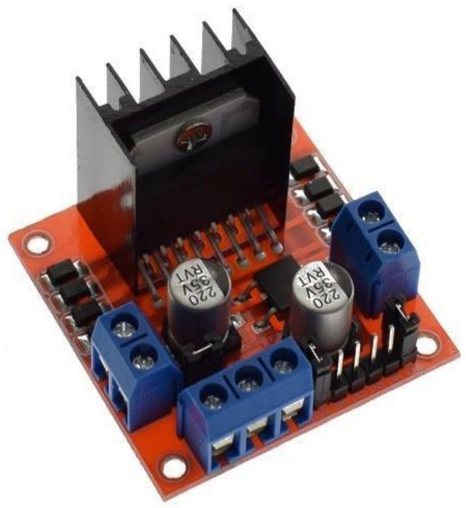
1. **ESP32 CAM**
2. **L298n Motor Driver**
3. **Gear motor**
4. **18650 Battery**
5. **Lm2596 buck converter**
6. **Arduino Nano**
7. **Neo 6m GPS module**
8. **18650 Battery Servo motor**
9. **I2C LCD**
10. **SIM 800l**
11. **L298n Motor Driver**

1. **ESP32 CAM**

The ESP8266 and ESP32 are popular microcontroller platforms developed by Espressif Systems, known for their built-in Wi-Fi capabilities and low-cost, making them ideal for Internet of Things (IoT) projects.



2. **Motor Driver L298:** L298 Motor Driver Module is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. L298 Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.



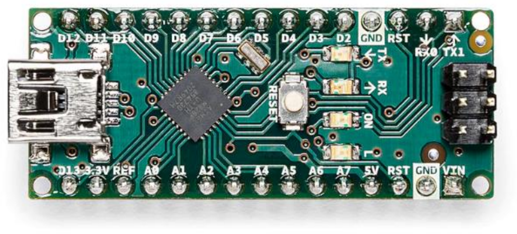
3. **GEAR MOTOR :** A gear motor combines an electric motor with gears, allowing speed adjustment and torque amplification. Widely applied in robotics, automotive, and industrial settings, they offer precise control. Available in DC, AC, and brushless DC types, each suits different needs. DC gear motors excel in robotics and automotive systems, providing dependable performance and control. AC gear motors power household appliances like HVAC systems and industrial conveyors. Brushless DC gear motors offer high efficiency and reliability, ideal for precision applications with low maintenance needs.

4. **Batteries 18650 :** 18650 batteries are cylindrical lithium-ion rechargeable batteries, measuring 18mm in diameter and 65mm in length. They're commonly used in portable electronic devices such as laptops, flashlights, power tools, and electric vehicles. Known for their high energy density and long lifespan, 18650 batteries provide reliable power for extended periods.

5. **LM2596 Dc to Dc Buck Converter :** The LM2596 is a popular type of DC to DC buck converter integrated circuit (IC) manufactured by Texas Instruments. Buck converters are a type of step-down voltage regulator, meaning they lower the input voltage to a lower output voltage. The LM2596 specifically is widely used due to its efficiency, ease of use, and versatility in various electronic applications.

6. **Arduino Nano :** The Arduino Nano is a compact and breadboard-friendly version of the popular Arduino board. It is based on the ATmega328P microcontroller and has similar functionalities to the Arduino UNO but in a smaller form factor. The Nano is equipped with 14 digital I/O pins, 8 analog input pins, and runs at 5V.

   It's an excellent choice for projects where space is limited, such as wearable electronics, small robots, and more. The Nano can be programmed using the Arduino IDE, making it easy to get started for beginners and versatile for experienced users.



7. **Neo 6mm GPS Module :** The Neo-6M GPS module is a popular and affordable GPS receiver module that is commonly used in Arduino and other microcontroller-based projects. It uses the MediaTek MT3333 chipset and can receive signals from GPS, GLONASS, and other GNSS satellites.

8. **Servo Motor :** A servomotor is a closed-loop servomechanism that uses position feedback (either linear or rotational position) to control its motion and final position. The input to its control is a signal (either analog or digital) representing the desired position of the output shaft.

9. **I2C LCD :** An I2C LCD (Inter-Integrated Circuit Liquid Crystal Display) is a type of LCD module that uses the I2C (Inter-Integrated Circuit) protocol for communication. This protocol allows for serial communication between a microcontroller or other host device and the LCD module using only two wires: a data line (SDA) and a clock line (SCL).

5.2 SOFTWARE REQUIREMENTS

1. Arduino IDE

Arduino IDE Software Intro

An official software introduced by Arduino.cc, that is mainly used for writing, compiling and uploading the code in almost all Arduino modules/boards. Arduino IDE is open-source software and is easily available to download & install.

- Arduino IDE is an open-source software, designed by Arduino.cc and mainly used for writing, compiling & uploading code to almost all Arduino Modules.
- It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
- It is available for all operating systems Le., MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role in debugging, editing and compiling the code.
- A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.
- Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.
- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
- The IDE environment mainly contains two basic parts: Editor and Compiler where the former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.

- This environment supports both c and c++ language.

- As you go to the preference section and check the complication section, the Output Pane will show the code compilation as you click the upload button.

- And at the end of the compilation, it will show you the hex file it has generated for the recent sketch that will send to the Arduino Board for the Specific task you aim to achieve.

- **Edit -** Used for copying and pasting the code with further modifications for font

- **Sketch -** For compilation and programming

- **Tools -** Mainly used for testing projects. The Programmer section in the panel is used for burning a bootloader to the new microcontroller.

- **Help -** In case you are feeling sceptical about software, complete help is available from getting started to troubleshoot.

- The checkmark appearing in the circular button is used to verify the code. Click this once you have written your code.

- The arrow key will upload and transfer the required code to the Arduino board • The dotted paper is used for creating a new file.

- The upward arrow is reserved for opening an existing Arduino project.

- The downward arrow is used to save the current running code.

- The button appearing on the top right corner is a Serial Monitor - A separate pop-up window that acts as an independent terminal and plays a vital role in sending and receiving the Serial Data. You can also go to the Tools panel and select Serial Monitor or pressing Ctrl + Shift + M all at once will open it instantly The Serial Monitor will actually help to debug the written Sketches where you can get a hold of how your program is operating Your Arduino Module should be connected to your computer by USB cable in order to activate the Serial Monitor.

- You need to select the baud rate of the Arduino Board you are using right now. For my Arduino Uno Baud Rate is 9600, as you write the following code and click the Serial Monitor, the output will show as the image below

Chapter 6

Experimental Results & Output

We have utilized the ESP32 CAM with the L298N motor driver for monitoring and controlling the bot. This allows for controlling its speed, Headlight brightness, and the camera's pan-tilt position via the IP address generated by the ESP32 CAM on a web server, providing a better user interface. Additionally, we have integrated object detection into the web server for enhanced functionality.

To further enhance the bot's capabilities, we have incorporated an Arduino Nano and an ultrasonic sensor. This setup notifies the user with a beep when the bot approaches an obstacle. Furthermore, we have integrated a SIM800L and a 6M GPS module to provide call and SMS alerts in case of accidental behavior by the bot. These alerts are triggered by sensors such as the ADXL335 accelerometer for Accidental detection and the DHT11 sensor for monitoring high temperature and humidity, which helps in detecting fire and other emergencies, ensuring the safety and improved performance of the bot.

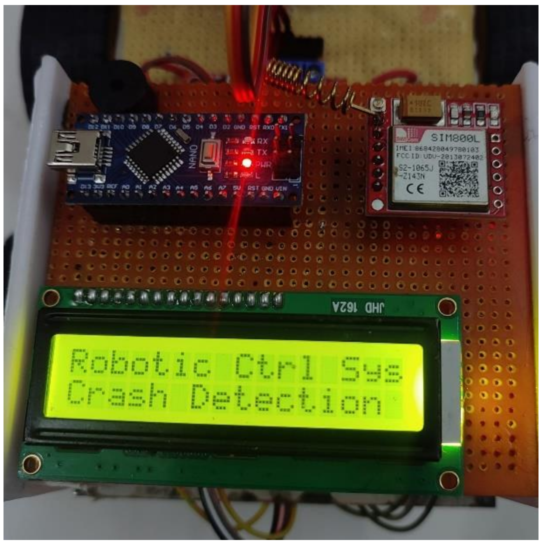


Figure 3: Accidental Detection

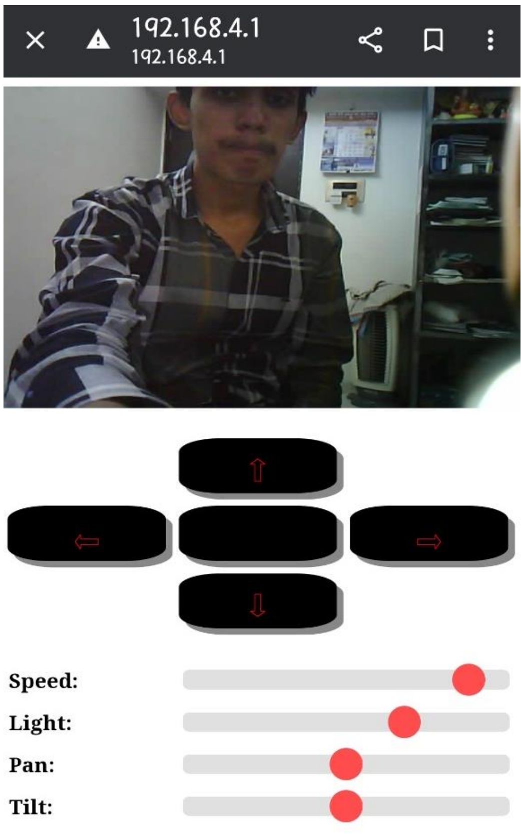


Figure 4: Pan-Tilt Web Page

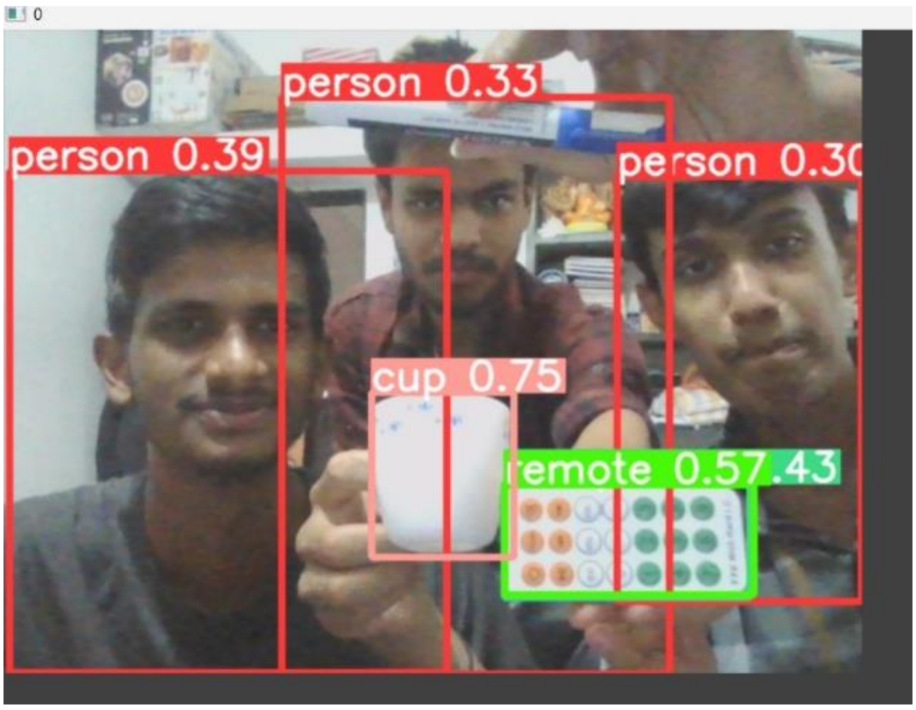


Figure 5: Object Detection

Chapter 7

APPLICATION

Robotic control systems based on object recognition and distance estimation have a wide range of applications, including:

1. **Manufacturing**: Robotic control systems can be used to automate manufacturing tasks such as assembly, welding, and painting. Object recognition and distance estimation can be used to ensure that robots accurately locate and manipulate objects.

2. **Logistics:** Robotic control systems can be used to automate logistics tasks such as picking and packing, palletizing, and transportation. Object recognition and distance estimation can be used to ensure that robots accurately identify and handle objects.

3. **Accident Detection and Notification:** The system can detect sudden changes in vehicle speed, tilt, or impact using sensors. Upon detecting an accident, it can automatically send an alert to emergency contacts or local authorities via GSM.

4. **Vehicle Tracking:** GPS integration allows real-time tracking of the vehicle's location. This feature can be useful for fleet management, stolen vehicle recovery, and monitoring vehicle routes.

5. **Agriculture**: Robotic control systems can be used to automate agricultural tasks such as harvesting, weeding, and spraying. Object recognition and distance estimation can be used to ensure that robots accurately identify and interact with crops and pests.

6. **Search and rescue**: Robotic control systems can be used to search for people trapped in collapsed buildings or other disaster zones. Object recognition and distance estimation can be used to help robots navigate difficult terrain and identify survivors.

7. **Emergency Assistance:** Drivers can manualy trigger emergency alerts using a panic button or through a mobile app connected to the system. This can be helpful in situations where immediate assistance is required, such as medical emergencies or breakdowns.

Chapter 8

ADVANTAGES AND DISADVANTAGE

Advantages:

1. **Accuracy**: Robotic control systems based on object recognition and distance estimation can be very accurate, as they can use sensors to precisely measure the position and orientation of objects.

2. **Flexibility**: Robotic control systems based on object recognition and distance estimation can be flexible and adaptable, as they can be programmed to recognize and interact with a variety of different objects.

3. **Safety:** Robotic control systems based on object recognition and distance estimation can be safer than traditional robotic systems, as they can be used to automate tasks that are dangerous or repetitive.

4. **Real-time Alerts:** Provides immediate alerts to emergency contacts or authorities during accidents or emergencies.

5. **Location Tracking:** Allows real-time tracking of the vehicle's location, aiding in stolen vehicle recovery and route monitoring.

Disadvantages*:*

1. **Cost**: Robotic control systems based on object recognition and distance estimation can be expensive, as they require sophisticated sensors and software.

2. **Complexity**: Robotic control systems based on object recognition and distance estimation can be complex to develop and maintain.

3. **Reliability:** Robotic control systems based on object recognition and distance estimation may not be reliable in all environments, such as those with low lighting or heavy dust.

4. **Safety concerns**: There are some safety concerns associated with robotic control systems based on object recognition and distance estimation, such as the risk of collisions with humans or other objects.

5. **False Alarms:** Without proper calibration, the system may trigger false alarms due to sensor inaccuracies, leading to unnecessary alerts.

6. **Security Concerns:** GSM-based systems can be susceptible to hacking or unauthorized access if not adequately secured.

Chapter 9

CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION

The proposed robotic control system leverages advanced object detection and crash alert technologies to enhance the navigation, safety, and efficiency of robotic platforms across diverse applications. Integrating state-of-the-art computer vision algorithms, the system enables robots to perceive and understand their environment, identify obstacles, and interact intelligently in dynamic settings.

With a robust crash alert mechanism, the system preemptively detects potential collision risks, triggering immediate corrective actions to ensure safe operation and minimize accident risks. Designed for modularity, scalability, and adaptability, the control architecture facilitates seamless integration with different robotic platforms, allowing operators to customize control parameters and optimize performance.

Additionally, an intuitive user interface provides operators with real-time feedback and visualization tools for easy configuration and monitoring, ensuring smooth and efficient robotic operation.

9.2 FUTURE SCOPE

The future scope of this robotic control system encompasses several promising avenues for further development and implementation. One potential direction is the integration of more advanced artificial intelligence (AI) techniques, such as reinforcement learning, to enhance the system's adaptive decision-making capabilities and autonomy. This could enable the robot to learn from its interactions and experiences, continuously improving its navigation and collision avoidance strategies.

Furthermore, the incorporation of additional sensor modalities, such as LiDAR and radar, could further enhance the system's environmental perception capabilities, providing more comprehensive and accurate spatial awareness. This could enable the robot to navigate complex terrains and environments with increased precision and reliability.

Another exciting prospect is the application of the control system in collaborative robotic systems, where multiple robots work together in coordinated tasks. Implementing communication and coordination algorithms could facilitate collaborative navigation, object manipulation, and task execution, opening up new opportunities for automation in industries like manufacturing, logistics, and healthcare.

Lastly, the continuous advancements in hardware technologies, sensor capabilities, and AI algorithms offer potential avenues for improving the system's performance, efficiency, and safety, paving the way for the development of next-generation robotic systems capable of operating in increasingly challenging and dynamic environments.

REFERENCES

[1] S., Mohanapriya & P., Natesan & P., Indhumathi & P., Mohanapriya & R., Monisha. (2021). "Object and Lane detection for autonomous vehicle using YOLO V3 algorithm" *AIP Conference Proceedings. 2387.140009.10.1063/5.0068836.*

[2] A., Shaikh & Kureshi, Dr. (2020). "Object Detection and Tracking using YOLO v3 Framework for Increased Resolution Video" *International Journal of Innovative Technology and Exploring Engineering. 9. 118- 125.10.35940/ijitee.E3038.049620.*

[3] Panthati, Jagadeesh. (2022). "Traffic Object Detection and Distance Estimation Using YOLOv3." *10.4271/2022-28-0120.*

[4] Givan, Daxton & Hanshaw, Noah & Choi, Heejun. (2022). "Application of Lane Navigation and Object Detection in a Deep-Learning Self-driving Car." *10.1007/978-3-030-98012-2_63.*

[5] Lim, Yuxiang & Tiang, Sew & Lim, Wei Hong & Wong, Chin Hong & Mastaneh, Mokayef & Chong, K. & Sun, Bo. (2024). "Object Detection in Autonomous Vehicles: A Performance Analysis." *10.1007/978-981-99-8498-5_21.*

[6] Wang, Rui & Wang, Ziyue & Xu, Zhengwei & Wang, Chi & Liu, Qiang & Zhang, Yuxin. (2021). "A Real-time Object Detector for Autonomous Vehicles Based on YOLOv4. Computational Intelligence and Neuroscience." *2021. 10.1155/2021/9218137.*

[7] Monteadora Renzo (2022). "Ros Based Self-Driving Rc Car with Tensorrt Integration In Reducing Behavioral Cloning Latency And Object Detection For Collosion Prevention." *10.13140/RG.2.2.15364.60803..*

[8] Hallan, Akul & Rawat, Shreya & Kumar, Sandeep & Ul-Haq, Ahsan & Gandhar, Abhishek. (2020). "Vehicle Accidental Alert System." *24-30.*

[9] Karuna, Gotlur & Kumar, Ram & Sai, Vadlapatla & Abhishek, Jula & Shashikanth, Mood & Kashyap, Burra.(2023). "Motorcycle Accidental Detection and Alert System using IoT." *E3S Web of Conferences. 391. 10.1051/e3sconf/202339101145.*

Appendix: Program CODE:-

```
// Pan Tilt Program
#include "esp_camera.h"
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <iostream>
#include <sstream>
#include <ESP32Servo.h>
#define PAN_PIN 14
#define TILT_PIN 15
Servo panServo;
Servo tiltServo;
struct MOTOR_PINS
{
  int pinEn;
  int pinIN1;
  int pinIN2;
};
std::vector<MOTOR_PINS> motorPins =
{
  {2, 12, 13}, //RIGHT_MOTOR Pins (EnA, IN1, IN2)
  {2, 1, 3},  //LEFT_MOTOR  Pins (EnB, IN3, IN4)
};
#define LIGHT_PIN 4
#define UP 1
#define DOWN 2
#define LEFT 3
#define RIGHT 4
#define STOP 0
#define RIGHT_MOTOR 0
#define LEFT_MOTOR 1
#define FORWARD 1
#define BACKWARD -1
const int PWMFreq = 1000; /* 1 KHz */
const int PWMResolution = 8;
const int PWMSpeedChannel = 2;
const int PWMLightChannel = 3;
//Camera related constants
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
```

```
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      21
#define Y4_GPIO_NUM      19
#define Y3_GPIO_NUM      18
#define Y2_GPIO_NUM       5
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22
const char* ssid     = "MyWiFiCar";
const char* password = "123456789";

AsyncWebServer server(80);
AsyncWebSocket wsCamera("/Camera");
AsyncWebSocket wsCarInput("/CarInput");
uint32_t cameraClientId = 0;

const char* htmlHomePage PROGMEM = R"HTMLHOMEPAGE(
<!DOCTYPE html>
<html>
  <head>
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
    <style>
    .arrows {
      font-size:30px;
      color:red;
    }
    td.button {
      background-color:black;
      border-radius:25%;
      box-shadow: 5px 5px #888888;
    }
    td.button:active {
      transform: translate(5px,5px);
      box-shadow: none;
    }

    .noselect {
      -webkit-touch-callout: none; /* iOS Safari */
        -webkit-user-select: none; /* Safari */
         -khtml-user-select: none; /* Konqueror HTML */
           -moz-user-select: none; /* Firefox */
            -ms-user-select: none; /* Internet Explorer/Edge */
                user-select: none; /* Non-prefixed version, currently
                                      supported by Chrome and Opera */
    }

    .slidecontainer {
      width: 100%;
    }

    .slider {
      -webkit-appearance: none;
      width: 100%;
      height: 15px;
      border-radius: 5px;
      background: #d3d3d3;
      outline: none;
      opacity: 0.7;
      -webkit-transition: .2s;
      transition: opacity .2s;
    }

    .slider:hover {
      opacity: 1;
    }

    .slider::-webkit-slider-thumb {
      -webkit-appearance: none;
      appearance: none;
      width: 25px;
      height: 25px;
      border-radius: 50%;
      background: red;
      cursor: pointer;
    }

    .slider::-moz-range-thumb {
      width: 25px;
      height: 25px;
      border-radius: 50%;
      background: red;
      cursor: pointer;
    }

    </style>

  </head>
  <body class="noselect" align="center" style="background-color:white">
    <table id="mainTable" style="width:400px;margin:auto;table-layout:fixed" CELLSPACING=10>
      <tr>
        <img id="cameraImage" src="" style="width:400px;height:250px"></td>
      </tr>
      <tr>
        <td></td>
        <td class="button" ontouchstart='sendButtonInput("MoveCar","1")' ontouchend='sendButtonInput("MoveCar","0")'><span
class="arrows" >&#8679;</span></td>
        <td></td>
      </tr>
      <tr>
        <td class="button" ontouchstart='sendButtonInput("MoveCar","3")' ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8678;</span></td>
        <td class="button"></td>
        <td class="button" ontouchstart='sendButtonInput("MoveCar","4")' ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8680;</span></td>
      </tr>
      <tr>
        <td></td>
        <td class="button" ontouchstart='sendButtonInput("MoveCar","2")' ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8681;</span></td>
        <td></td>
      </tr>
      <tr/><tr/>
      <tr>
        <td style="text-align:left"><b>Speed:</b></td>
        <td colspan=2>
         <div class="slidecontainer">
            <input type="range" min="0" max="255" value="150" class="slider" id="Speed" oninput='sendButtonInput("Speed",value)'>
          </div>
        </td>
      </tr>
      <tr>
        <td style="text-align:left"><b>Light:</b></td>
        <td colspan=2>
          <div class="slidecontainer">
            <input type="range" min="0" max="255" value="0" class="slider" id="Light" oninput='sendButtonInput("Light",value)'>
          </div>
        </td>
      </tr>
      <tr>
        <td style="text-align:left"><b>Pan:</b></td>
        <td colspan=2>
         <div class="slidecontainer">
            <input type="range" min="0" max="180" value="90" class="slider" id="Pan" oninput='sendButtonInput("Pan",value)'>
          </div>
        </td>
      </tr>
      <tr>
        <td style="text-align:left"><b>Tilt:</b></td>
        <td colspan=2>
          <div class="slidecontainer">
            <input type="range" min="0" max="180" value="90" class="slider" id="Tilt" oninput='sendButtonInput("Tilt",value)'>
          </div>
        </td>
      </tr>
    </table>
```

```
<script>
  var webSocketCameraUrl = "ws:\/\/" + window.location.hostname + "/Camera";
  var webSocketCarInputUrl = "ws:\/\/" + window.location.hostname + "/CarInput";
  var websocketCamera;
  var websocketCarInput;

  function initCameraWebSocket()
  {
    websocketCamera = new WebSocket(webSocketCameraUrl);
    websocketCamera.binaryType = 'blob';
    websocketCamera.onopen    = function(event){};
    websocketCamera.onclose   = function(event){setTimeout(initCameraWebSocket, 2000);};
    websocketCamera.onmessage = function(event)
    {
      var imageId = document.getElementById("cameraImage");
      imageId.src = URL.createObjectURL(event.data);
    };
  }

  function initCarInputWebSocket()
  {
    websocketCarInput = new WebSocket(webSocketCarInputUrl);
    websocketCarInput.onopen    = function(event)
    {
      sendButtonInput("Speed", document.getElementById("Speed").value);
      sendButtonInput("Light", document.getElementById("Light").value);
      sendButtonInput("Pan", document.getElementById("Pan").value);
      sendButtonInput("Tilt", document.getElementById("Tilt").value);
    };
    websocketCarInput.onclose   = function(event){setTimeout(initCarInputWebSocket, 2000);};
    websocketCarInput.onmessage = function(event){};
  }

  function initWebSocket()
  {
    initCameraWebSocket ();
    initCarInputWebSocket();
  }

  function sendButtonInput(key, value)
  {
    var data = key + "," + value;
    websocketCarInput.send(data);
  }

  window.onload = initWebSocket;
  document.getElementById("mainTable").addEventListener("touchend", function(event){
    event.preventDefault()
  });
```

```
    </script>
  </body>
</html>
)HTMLHOMEPAGE";


void rotateMotor(int motorNumber, int motorDirection)
{
  if (motorDirection == FORWARD)
  {
    digitalWrite(motorPins[motorNumber].pinIN1, HIGH);
    digitalWrite(motorPins[motorNumber].pinIN2, LOW);
  }
  else if (motorDirection == BACKWARD)
  {
    digitalWrite(motorPins[motorNumber].pinIN1, LOW);
    digitalWrite(motorPins[motorNumber].pinIN2, HIGH);
  }
  else
  {
    digitalWrite(motorPins[motorNumber].pinIN1, LOW);
    digitalWrite(motorPins[motorNumber].pinIN2, LOW);
  }
}

void moveCar(int inputValue)
{
  Serial.printf("Got value as %d\n", inputValue);
  switch(inputValue)
  {

    case UP:
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR, FORWARD);
      break;

    case DOWN:
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, BACKWARD);
      break;

    case LEFT:
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR, BACKWARD);
      break;

    case RIGHT:
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, FORWARD);
    break;

   case STOP:
    rotateMotor(RIGHT_MOTOR, STOP);
    rotateMotor(LEFT_MOTOR, STOP);
    break;

   default:
    rotateMotor(RIGHT_MOTOR, STOP);
    rotateMotor(LEFT_MOTOR, STOP);
    break;
  }
}

void handleRoot(AsyncWebServerRequest *request)
{
 request->send_P(200, "text/html", htmlHomePage);
}

void handleNotFound(AsyncWebServerRequest *request)
{
   request->send(404, "text/plain", "File Not Found");
}

void onCarInputWebSocketEvent(AsyncWebSocket *server,
                      AsyncWebSocketClient *client,
                      AwsEventType type,
                      void *arg,
                      uint8_t *data,
                      size_t len)
{
 switch (type)
  {
   case WS_EVT_CONNECT:
    Serial.printf("WebSocket client #%u connected from %s\n", client->id(), client->remoteIP().toString().c_str());
    break;
   case WS_EVT_DISCONNECT:
    Serial.printf("WebSocket client #%u disconnected\n", client->id());
    moveCar(0);
    ledcWrite(PWMLightChannel, 0);
    panServo.write(90);
    tiltServo.write(90);
    break;
   case WS_EVT_DATA:
    AwsFrameInfo *info;
    info = (AwsFrameInfo*)arg;
    if (info->final && info->index == 0 && info->len == len && info->opcode == WS_TEXT)
    {
      std::string myData = "";
```

```
        myData.assign((char *)data, len);
        std::istringstream ss(myData);
        std::string key, value;
        std::getline(ss, key, ',');
        std::getline(ss, value, ',');
        Serial.printf("Key [%s] Value[%s]\n", key.c_str(), value.c_str());
        int valueInt = atoi(value.c_str());
        if (key == "MoveCar")
        {
          moveCar(valueInt);
        }
        else if (key == "Speed")
        {
          ledcWrite(PWMSpeedChannel, valueInt);
        }
        else if (key == "Light")
        {
          ledcWrite(PWMLightChannel, valueInt);
        }
        else if (key == "Pan")
        {
          panServo.write(valueInt);
        }
        else if (key == "Tilt")
        {
          tiltServo.write(valueInt);
        }
      }
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      break;
    default:
      break;
  }
}

void onCameraWebSocketEvent(AsyncWebSocket *server,
                      AsyncWebSocketClient *client,
                      AwsEventType type,
                      void *arg,
                      uint8_t *data,
                      size_t len)
{
  switch (type)
  {
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u connected from %s\n", client->id(), client->remoteIP().toString().c_str());
      cameraClientId = client->id();
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u disconnected\n", client->id());
      cameraClientId = 0;
      break;
    case WS_EVT_DATA:
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      break;
    default:
      break;
  }
}

void setupCamera()
{
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_4;
  config.ledc_timer = LEDC_TIMER_2;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 10;
  config.fb_count = 1;

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK)
  {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  if (psramFound())
  {
    heap_caps_malloc_extmem_enable(20000);
    Serial.printf("PSRAM initialized. malloc to take memory from psram above this size");
  }
}

void sendCameraPicture()
{
  if (cameraClientId == 0)
  {
    return;
  }
  unsigned long  startTime1 = millis();
  //capture a frame
  camera_fb_t * fb = esp_camera_fb_get();
  if (!fb)
  {
      Serial.println("Frame buffer could not be acquired");
      return;
  }

  unsigned long  startTime2 = millis();
  wsCamera.binary(cameraClientId, fb->buf, fb->len);
  esp_camera_fb_return(fb);

  //Wait for message to be delivered
  while (true)
  {
    AsyncWebSocketClient * clientPointer = wsCamera.client(cameraClientId);
    if (!clientPointer || !(clientPointer->queueIsFull()))
    {
      break;
    }
    delay(1);
  }

  unsigned long  startTime3 = millis();
  Serial.printf("Time taken Total: %d|%d|%d\n",startTime3 - startTime1, startTime2 - startTime1, startTime3-startTime2 );
}

void setUpPinModes()
{
  panServo.attach(PAN_PIN);
  tiltServo.attach(TILT_PIN);

  //Set up PWM
  ledcSetup(PWMSpeedChannel, PWMFreq, PWMResolution);
  ledcSetup(PWMLightChannel, PWMFreq, PWMResolution);

  for (int i = 0; i < motorPins.size(); i++)
  {
    pinMode(motorPins[i].pinEn, OUTPUT);
    pinMode(motorPins[i].pinIN1, OUTPUT);
    pinMode(motorPins[i].pinIN2, OUTPUT);
    /* Attach the PWM Channel to the motor enb Pin */
    ledcAttachPin(motorPins[i].pinEn, PWMSpeedChannel);
  }
  moveCar(STOP);

  pinMode(LIGHT_PIN, OUTPUT);
  ledcAttachPin(LIGHT_PIN, PWMLightChannel);
}


void setup(void)
{
  setUpPinModes();
  //Serial.begin(115200);

  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  server.on("/", HTTP_GET, handleRoot);
  server.onNotFound(handleNotFound);
  wsCamera.onEvent(onCameraWebSocketEvent);
  server.addHandler(&wsCamera);
  wsCarInput.onEvent(onCarInputWebSocketEvent);
  server.addHandler(&wsCarInput);
  server.begin();
  Serial.println("HTTP server started");
  setupCamera();
}
void loop()
{
  wsCamera.cleanupClients();
  wsCarInput.cleanupClients();
  sendCameraPicture();
  Serial.printf("SPIRam Total heap %d, SPIRam Free Heap %d\n", ESP.getPsramSize(), ESP.getFreePsram());
}
//Accidental Alert System
#include<LiquidCrystal_I2C.h>
#include <AltSoftSerial.h>
#include <TinyGPS++.h>
```

```
#include <SoftwareSerial.h>
#include <math.h>
#include <DHT.h>
#define DHTPIN 7 // Pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);
#include<Wire.h>
//must add i2c lcd address use i2c-scanner.ino file
LiquidCrystal_I2C lcd(0x27, 16, 2);
//--------------------------------------------------------------
//emergency phone number with country code
const String EMERGENCY_PHONE = "+919172752999";
//--------------------------------------------------------------
//GSM Module RX pin to Arduino 3
//GSM Module TX pin to Arduino 2
#define rxPin 2
#define txPin 3
SoftwareSerial sim800(rxPin,txPin);
//--------------------------------------------------------------
//GPS Module RX pin to Arduino 9
//GPS Module TX pin to Arduino 8
AltSoftSerial neogps;
TinyGPSPlus gps;
//--------------------------------------------------------------
String sms_status,sender_number,received_date,msg;
String latitude, longitude;
//--------------------------------------------------------------
#define BUZZER_PIN 11
#define BUZZER 11
#define BUTTON 12
//--------------------------------------------------------------
#define xPin A1
#define yPin A2
#define zPin A3
//--------------------------------------------------------------
byte updateflag;
int xaxis = 0, yaxis = 0, zaxis = 0;
int deltx = 0, delty = 0, deltz = 0;
int vibration = 1, devibrate = 37;
int magnitude = 0;
int sensitivity = 8 ;
double angle;
boolean impact_detected = false;
//Used to run impact routine every 2mS.
unsigned long time1;
unsigned long impact_time;
unsigned long alert_delay = 20000; //30 seconds
//--------------------------------------------------------------
void setup()
```

```
{
 Serial.begin(9600);
  dht.begin();
  pinMode(BUZZER_PIN, OUTPUT);
  //--------------------------------------------------------------
  //Serial.println("Arduino serial initialize");
  Serial.begin(9600);
  //--------------------------------------------------------------
  //Serial.println("SIM800L serial initialize");
  sim800.begin(9600);
  //--------------------------------------------------------------
  //Serial.println("NEO6M serial initialize");
  neogps.begin(9600);
  //--------------------------------------------------------------
  pinMode(BUZZER, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);
  //--------------------------------------------------------------
  //initialize lcd screen
  lcd.begin();
  // turn on the backlight
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Robotic Ctrl Sys");
  lcd.setCursor(0, 1);
   lcd.print("Crash Detection");
  delay(5000);
  lcd.clear();
  //--------------------------------------------------------------
  sms_status = "";
  sender_number="";
  received_date="";
  msg="";
  //--------------------------------------------------------------
  sim800.println("AT"); //Check GSM Module
  delay(1000);
  //SendAT("AT", "OK", 2000); //Check GSM Module
  sim800.println("ATE1"); //Echo ON
  delay(1000);
  //SendAT("ATE1", "OK", 2000); //Echo ON
  sim800.println("AT+CPIN?"); //Check SIM ready
  delay(1000);
  //SendAT("AT+CPIN?", "READY", 2000); //Check SIM ready
  sim800.println("AT+CMGF=1"); //SMS text mode
  delay(1000);
  //SendAT("AT+CMGF=1", "OK", 2000); //SMS text mode
  sim800.println("AT+CNMI=1,1,0,0,0"); /// Decides how newly arrived SMS should be handled
  delay(1000);
  //SendAT("AT+CNMI=1,1,0,0,0", "OK", 2000); //set sms received format
  //AT +CNMI = 2,1,0,0,0 - AT +CNMI = 2,2,0,0,0 (both are same)
  //--------------------------------------------------------------
  time1 = micros();
  //Serial.print("time1 = "); Serial.println(time1);
  //--------------------------------------------------------------
  //read calibrated values. otherwise false impact will trigger
  //when you reset your Arduino. (By pressing reset button)
  xaxis = analogRead(xPin);
  yaxis = analogRead(yPin);
  zaxis = analogRead(zPin);
  //--------------------------------------------------------------
}
void loop()
{
 delay(2000); // Wait a few seconds between measurements.

  float temperature = dht.readTemperature(); // Read temperature as Celsius
  if (isnan(temperature)) {
    Serial.println("Failed to read temperature from DHT sensor!");
    return;
  }
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
  if (temperature > 39) { // Set your temperature threshold here
    digitalWrite(BUZZER_PIN, HIGH); // Turn on the buzzer
    delay(2000); // Buzzer on time
    digitalWrite(BUZZER_PIN, LOW); // Turn off the buzzer
    delay(1000); // Wait before next check
  }
  //--------------------------------------------------------------
  //call impact routine every 2mS
  if (micros() - time1 > 1999) Impact();
  //--------------------------------------------------------------
  if(updateflag > 0)
  {
    updateflag=0;
    Serial.println("Impact detected!!");
    Serial.print("Magnitude:"); Serial.println(magnitude);
    getGps();
    digitalWrite(BUZZER, HIGH);
    impact_detected = true;
    impact_time = millis();
    lcd.clear();
    lcd.setCursor(0,0); //col=0 row=0
    lcd.print("Crash Detected");
    lcd.setCursor(0,1); //col=0 row=1
    lcd.print("Magnitude:"+String(magnitude));
  }
  //--------------------------------------------------------------
  if(impact_detected == true)
  {
    if(millis() - impact_time >= alert_delay) {
      digitalWrite(BUZZER, LOW);
      makeCall();
      delay(1000);
      sendAlert();
      impact_detected = false;
      impact_time = 0;
    }
  }
  if(digitalRead(BUTTON) == LOW){
    delay(200);
    digitalWrite(BUZZER, LOW);
    impact_detected = false;
    impact_time = 0;
  }
  //--------------------------------------------------------------
  while(sim800.available()){
    parseData(sim800.readString());
  }
  //--------------------------------------------------------------
  while(Serial.available())  {
    sim800.println(Serial.readString());
  }
  }
void Impact()
{
  //--------------------------------------------------------------
  time1 = micros(); // resets time value
  //--------------------------------------------------------------
  int oldx = xaxis; //store previous axis readings for comparison
  int oldy = yaxis;
  int oldz = zaxis;
  xaxis = analogRead(xPin);
  yaxis = analogRead(yPin);
  zaxis = analogRead(zPin);
  //--------------------------------------------------------------
  //loop counter prevents false triggering. Vibration resets if there is an impact. Don't detect new changes until that "time" has passed.
  vibration--;
  //Serial.print("Vibration = "); Serial.println(vibration);
  if(vibration < 0) vibration = 0;
  //Serial.println("Vibration Reset!");
 if(vibration > 0) return;
  //--------------------------------------------------------------
  deltx = xaxis - oldx;
  delty = yaxis - oldy;
  deltz = zaxis - oldz;
  //Magnitude to calculate force of impact.
  magnitude = sqrt(sq(deltx) + sq(delty) + sq(deltz));
  if (magnitude >= sensitivity) //impact detected
  {
    updateflag=1;
    // reset anti-vibration counter
    vibration = devibrate;
  }
  else
  {
    //if (magnitude > 15)
      //Serial.println(magnitude);
    //reset magnitude of impact to 0
    magnitude=0;
  }
}
void parseData(String buff){
  Serial.println(buff);
  unsigned int len, index;
  //--------------------------------------------------------------
  //Remove sent "AT Command" from the response string.
  index = buff.indexOf("\r");
  buff.remove(0, index+2);
  buff.trim();
if(buff != "OK"){
    //--------------------------------------------------------------
    index = buff.indexOf(":");
    String cmd = buff.substring(0, index);
    cmd.trim();
    buff.remove(0, index+2);
    //Serial.println(buff);
    //--------------------------------------------------------------
    if(cmd == "+CMTI"){
      //get newly arrived memory location and store it in temp
      //temp = 4
      index = buff.indexOf(",");
      String temp = buff.substring(index+1, buff.length());
      temp = "AT+CMGR=" + temp + "\r";
      //AT+CMGR=4 i.e. get message stored at memory location 4
      sim800.println(temp);
    }
    //--------------------------------------------------------------
    else if(cmd == "+CMGR"){
      //extractSms(buff);
      //Serial.println(buff.indexOf(EMERGENCY_PHONE));
      if(buff.indexOf(EMERGENCY_PHONE) > 1){
        buff.toLowerCase();
        //Serial.println(buff.indexOf("get gps"));
        if(buff.indexOf("get gps") > 1){
          getGps();
          String sms_data;
          sms_data = "GPS Location Data\r";
          sms_data += "http://maps.google.com/maps?q=loc:";
          sms_data += latitude + "," + longitude;
          sendSms(sms_data);
        }
      }
    }
    //------------------------------------------------------------
  }
  else{
  //The result of AT Command is "OK"
  }
 }
void getGps()
{
  // Can take up to 60 seconds
  boolean newData = false;
  for (unsigned long start = millis(); millis() - start < 2000;){
    while (neogps.available()){
      if (gps.encode(neogps.read())){
        newData = true;
        break;
      }
    }
  }
  if (newData) //If newData is true
  {
    latitude = String(gps.location.lat(), 6);
    longitude = String(gps.location.lng(), 6);
    newData = false;
  }
  else {
    Serial.println("No GPS data is available");
    latitude = "";
    longitude = "";
  }
  Serial.print("Latitude= "); Serial.println(latitude);
  Serial.print("Longitude= "); Serial.println(longitude);
}
void sendAlert()
{
  String sms_data;
  sms_data = "Accident Alert!!\r";
  sms_data += "http://maps.google.com/maps?q=loc:";
  sms_data += latitude + "," + longitude;
  sendSms(sms_data);
}
void makeCall()
```

```
{
  Serial.println("calling....");
  sim800.println("ATD"+EMERGENCY_PHONE+";");
  delay(20000); //20 sec delay
  sim800.println("ATH");
  delay(1000); //1 sec delay
}
void sendSms(String text)
{
  //return;
  sim800.print("AT+CMGF=1\r");
  delay(1000);
  sim800.print("AT+CMGS=\""+EMERGENCY_PHONE+"\"\r");
  delay(1000);
  sim800.print(text);
  delay(100);
  sim800.write(0x1A); //ascii code for ctrl-26 //sim800.println((char)26); //ascii code for ctrl-26
  delay(1000);
  Serial.println("SMS Sent Successfully.");
}
boolean SendAT(String at_command, String expected_answer, unsigned int timeout){
    uint8_t x=0;
    boolean answer=0;
    String response;
    unsigned long previous;
    //Clean the input buffer
    while( sim800.available() > 0) sim800.read();
   sim800.println(at_command);
    x = 0;
    previous = millis();
    //this loop waits for the answer with time out
    do{
    //if there are data in the UART input buffer, reads it and checks for the asnwer
        if(sim800.available() != 0){
            response += sim800.read();
            x++;
            // check if the desired answer (OK) is in the response of the module
            if(response.indexOf(expected_answer) > 0){
                answer = 1;
                break;
            }
        }
    }while((answer == 0) && ((millis() - previous) < timeout));

  Serial.println(response);
  return answer;
}
```