IT5611 - Embedded Systems and Internet of Things Laboratory

A MINI PROJECT REPORT

Submitted by

Gowtham Rajasekaran 2022506084

B. Tech(6/8)

DEPARTMENT OF INFORMATION TECHNOLOGY

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY

CHENNAI - 600 044

Car Collision Alert System

1. Introduction

Road accidents are a leading cause of death and injury globally. A major reason behind collisions is the lack of real-time awareness of surroundings, especially in tight spaces or reverse movement. Modern vehicles use sensors to monitor proximity and trigger alerts or safety systems. This project presents a prototype of such a collision detection system using affordable components like Arduino, ultrasonic sensors, and motion detection units. It provides both preventive alerts (LEDs and buzzers) and post-collision actions (airbag simulation and Bluetooth alerts).

2. Objective

To design a functional, low-cost prototype that:

- Detects surrounding obstacles and alerts using LEDs and buzzers.

- Switches to enhanced reverse mode for parking assistance.

- Detects collisions via motion sensor and triggers airbag simulation.

- Sends wireless alerts using Bluetooth upon impact.

3. Components Used

• Arduino Uno – Microcontroller board controlling the entire system.

• Ultrasonic Sensor – For detecting object distance.

• MPU6050 – 6-axis motion sensor for detecting impact.

• LEDs (Red, Orange, White) – Visual proximity indicators.

• Buzzers – Audible alerts in reverse mode.

• Switch – Toggles reverse mode.

• HC-05 Bluetooth Module – Sends impact alerts wirelessly.

• Breadboard – For circuit prototyping.

• Jumper Wires – For connections.

• Resistors (10kΩ & 220Ω) – Used for limiting current `and voltage.

4. System Design

The system is built on a breadboard with Arduino as the controller. The ultrasonic sensor is used for object detection, with outputs mapped to LEDs that indicate distance visually:

- White LED: Object far

- Orange LED: Object moderately close

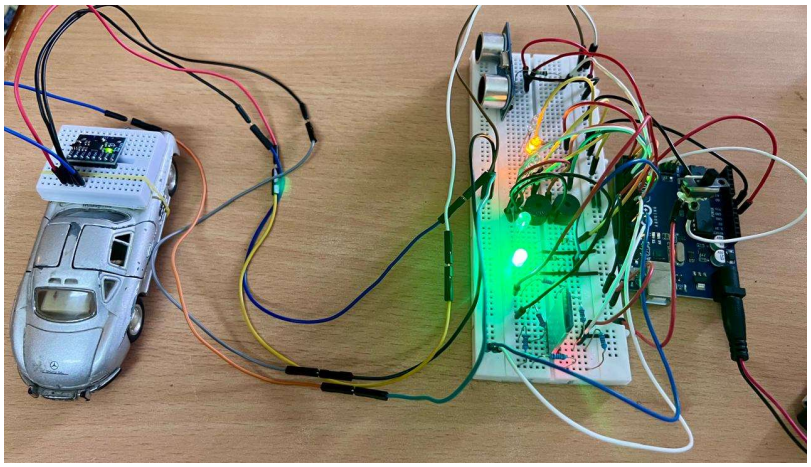
- Red LED: Object very close

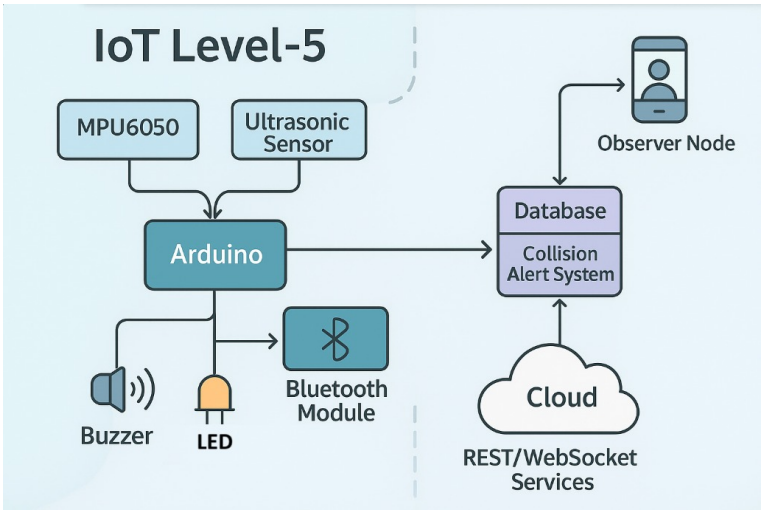
A switch connected to a digital pin on Arduino toggles reverse mode. In this mode, the buzzer activates and increases beeping frequency as proximity decreases, helping in parking.

The MPU6050 constantly monitors vibration. When a strong impact is detected:

- A signal is sent via serial (COM7) to a connected PC running Flask, which visually simulates airbag deployment on a webpage.

- Simultaneously, the HC-05 Bluetooth module sends a broadcast message to paired nearby devices indicating airbag activation.





5. Working Principle

- Proximity Detection: The ultrasonic sensor continuously sends out signals and measures the return time to calculate object distance. Based on this, appropriate LEDs light up. In reverse mode, the buzzer's tone increases as the car gets closer to an obstacle.

- Reverse Mode Activation: The switch acts as a manual gear simulator. When flipped, it alters the program logic to prioritize rear proximity sensing and enables the buzzer.

- Collision Detection: The MPU6050 measures motion data in all directions. When vibration exceeds a predefined threshold:

  - The Arduino sends data to the PC serially.

  - The Flask app responds by displaying an airbag deployment visualization on the web.

  - Bluetooth alerts are sent instantly via HC-05.

6. Source Code

code.ino

```
#include <Wire.h>
#include <SoftwareSerial.h>
#define TRIG_PIN 9
#define ECHO_PIN 10
#define GREEN_GOOD 13
#define RED_FAIL 2
#define RED_LED 3
#define ORANGE_LED 4
#define WHITE_LED 5
#define BUZZER 6
#define SWITCH_PIN 7

const int MPU_ADDR = 0x68;
int16_t ax, ay, az;
float prevAcceleration = 0;
const float impactThreshold = 2.5;

SoftwareSerial BT(11,12);
void setup() {
  Serial.begin(9600);
  BT.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(GREEN_GOOD, OUTPUT);
  pinMode(RED_FAIL, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(ORANGE_LED, OUTPUT);
  pinMode(WHITE_LED, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(SWITCH_PIN, INPUT);
  Wire.begin();
  Wire.setClock(400000);
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  Serial.println("MPU6050 initialized!");
  digitalWrite(GREEN_GOOD, HIGH);
}

long getDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);


 digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  return duration * 0.034 / 2;
}

void controlLEDs(long distance) {
```

```
  if(distance < 15) {
    digitalWrite(RED_LED, HIGH);
    digitalWrite(ORANGE_LED, LOW);
    digitalWrite(WHITE_LED, LOW);
  }
  else if(distance >= 15 && distance < 25) {
    digitalWrite(RED_LED, LOW);
    digitalWrite(ORANGE_LED, HIGH);
    digitalWrite(WHITE_LED, LOW);
  }
  else if(distance <= 35) {
    digitalWrite(RED_LED, LOW);
    digitalWrite(ORANGE_LED, LOW);
    digitalWrite(WHITE_LED, HIGH);
  }
  else{
    digitalWrite(RED_LED, LOW);
    digitalWrite(ORANGE_LED, LOW);
    digitalWrite(WHITE_LED, LOW);
  }
}

void controlBuzzer(long distance) {
  if (distance < 25) {
    int delayTime = map(distance, 0, 30, 50, 1000); // Faster beeps as object gets closer
    tone(BUZZER, 1000);


delay(delayTime);
    noTone(BUZZER);
    delay(delayTime);
  } else {
    noTone(BUZZER);
  }
}

void checkCrash() {
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_ADDR, 6, true);
  ax = Wire.read() << 8 | Wire.read();
  ay = Wire.read() << 8 | Wire.read();
  az = Wire.read() << 8 | Wire.read();

  float ax_g = ax / 16384.0;
  float ay_g = ay / 16384.0;
  float az_g = az / 16384.0;
  float acceleration = sqrt(ax_g * ax_g + ay_g * ay_g + az_g * az_g);
  float impactForce = abs(acceleration - prevAcceleration);

  Serial.print("Impact Force: ");
  Serial.println(impactForce);

  if (impactForce > impactThreshold) {
    digitalWrite(GREEN_GOOD, LOW);
    digitalWrite(RED_FAIL, HIGH);
    digitalWrite(RED_LED, LOW);
    digitalWrite(ORANGE_LED, LOW);
```

```
digitalWrite(WHITE_LED, LOW);
      BT.println(" 🚨 Code00X: Release Air Bag");
      Serial.println(" 🚗💥 Crash Detected! System Disabled.");

      while (true) {
        digitalWrite(RED_FAIL, HIGH);
        digitalWrite(GREEN_GOOD, LOW);
        tone(BUZZER, 1000);
      }
    }
    prevAcceleration = acceleration;
}

void loop() {
    checkCrash();

    long distance = getDistance();
    int mode = digitalRead(SWITCH_PIN);
    controlLEDs(distance);

    if (mode == 1) {
      controlBuzzer(distance);
    } else {
      noTone(BUZZER);
    }
    delay(100);
}
```

app.py

```
from flask import Flask, render_template
from flask_socketio import SocketIO
app = Flask(__name__)
socketio = SocketIO(
    app,
    cors_allowed_origins="*",
    async_mode="threading"  # Ensure compatibility with default Python threads
)

@app.route('/')
def index():
    return render_template('index.html')

@socketio.on('release_airbag')
def handle_release_airbag():
    print("Received 'release_airbag' signal, triggering video.")
    socketio.emit('play_video')

if __name__ == '__main__':
    socketio.run(app, debug=True)
```

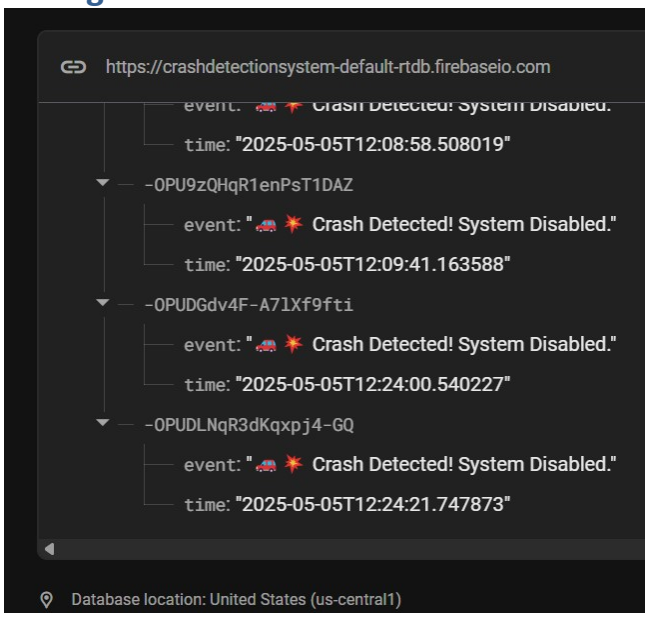
lisenter.py

```
import serial
import time
import socketio
import firebase_admin
from firebase_admin import credentials, db
from datetime import datetime

try:
   ser = serial.Serial('COM7', 9600)  # For Windows
   print("Successfully connected to COM7")
except serial.SerialException as e:
   print(f"Failed to connect to COM7: {e}")
   exit(1)  # Exit if COM7 connection fails

# Initialize Firebase
cred = credentials.Certificate("firebase-key.json")  # Your service account key file
firebase_admin.initialize_app(cred, {
   'databaseURL': 'https://crashdetectionsystem-default-rtdb.firebaseio.com'
})

# Initialize SocketIO client
sio = socketio.Client()

@sio.event
def connect():
   print("Successfully connected to Flask server")

def listen_for_trigger():
   while True:
      if ser.in_waiting > 0:
         data = ser.readline().decode('utf-8').strip()
         print(f"Received: {data}")

         if data == "⚠️ Crash Detected! System Disabled.":
            print("Trigger received, notifying frontend.")
            sio.emit('release_airbag')  # Emit WebSocket event
            # Log to Firebase
            timestamp = datetime.now().isoformat()
            db.reference('crashes').push({
               'time': timestamp,
               'event': data
            })
      time.sleep(0.1)

# Connect after event handlers are set
sio.connect('http://localhost:5000')

if __name__ == '__main__':
   # Start listening after connection
   listen_for_trigger()
```

7. Output in Web

AIR BAG BURSTS OPEN

8. Storage in Cloud



9. Advantages

- Low-Cost Implementation: Uses affordable components.
- Dual-Mode Detection: Both obstacle prevention and collision response.
- Visual & Audio Feedback: Enhances driver awareness.
- Wireless Alert Capability: Bluetooth messaging enhances post-impact communication.
- Expandable: Can be upgraded with more sensors for full 360° coverage.

10. Conclusion

This collision detection system offers an effective and low-cost prototype for real-time vehicle safety. It integrates obstacle sensing and impact response using basic electronics and microcontroller logic. While it uses a single ultrasonic sensor in this version, the concept can easily scale to multiple sensors for full environmental awareness. The addition of motion-triggered alerts and wireless communication adds practical value for safety systems in modern vehicles.