EMBEDDED SYSTEMS AND IOT PROJECT

Collision Detection System

LAB SLOT – L23 + L24

FACULTY NAME – DR. BHARANI DHARAN

TEAM MEMBERS

SAGNIK CHANDRA – 21BIT0251

NILAY – 21BIT0219

ROHIT KUMAR – 21BIT0317

CHAITANYA GOENKA – 21BIT0071

AKSHIT BAHL – 21BIT0012

ATHARVA BILLORE – 21BIT0307

CONTENT PAGE:

ACKNOWLEDGEMENT:

We would like to express my sincere gratitude to Prof. N. Bharani Dharan for his invaluable guidance and support throughout my project on "Collision Detection System" completed under the course of "Embedded System and IoT". His expertise in the field, insightful inputs, and continuous encouragement have been instrumental in shaping the project's direction and enhancing its outcomes. I am truly thankful for his mentorship, which has significantly contributed to my understanding and proficiency in the subject matter.

Also, we are thankful to all the faculty staff in the VIT practical laboratories.

AIM :

To create a real-time solution utilizing Arduino and suitable sensors for the purpose of detecting and recognizing accidents or unexpected impacts, enabling timely notifications to relevant individual and facilitating immediate assistance and intervention.

Components :

1. **Breadboard Small:**

   A breadboard is a commonly used prototyping tool in electronics. It is a device used to build and test electronic circuits without the need for soldering. Breadboards provide a platform for temporarily connecting electronic components such as resistors, capacitors, integrated circuits (ICs), and other electronic elements.

   The typical breadboard consists of a plastic board with a grid of holes or sockets, where components and wires can be inserted. The holes or sockets are typically arranged in parallel rows, and the rows are electrically connected horizontally. This allows for easy connection of components and the creation of circuits by simply inserting the component leads or wires into the appropriate holes or sockets.

   Breadboards also have metal clips or spring clips underneath the holes or sockets, which provide electrical connections vertically. These clips are typically arranged in groups of five, with each group forming a column. The clips in a column are electrically connected, allowing components or wires inserted in the same column to be connected.

2. **Gps Module:**

   A GPS (Global Positioning System) module is a device that receives signals from a network of satellites in space to determine its own location on Earth. It is commonly used in various applications such as navigation systems, vehicle tracking, mapping, and outdoor sports.

   A GPS module typically consists of a GPS receiver, an antenna, and supporting circuitry. The antenna receives signals from multiple GPS satellites orbiting the Earth. These signals contain information about the satellites' locations and the precise time the signals were transmitted.

   The GPS receiver in the module processes the received signals and calculates the module's position based on the time it took for the signals to reach the module from different satellites. By triangulating the signals from at least four satellites, the GPS module can determine its latitude, longitude, and altitude.

3. **Buzzer**

   A buzzer is a common component used in IoT (Internet of Things) devices for providing audible feedback or alerts. It is an electromechanical device that produces sound when an electric current is passed through it.

   In an IoT context, a buzzer can be integrated into a device to convey important information or notifications to the user. Here are a few examples of how buzzers are used in IoT applications:

   Alarms and Security Systems: Buzzer alerts can be used in IoT-based security systems to notify users of potential threats or breaches. For instance, if a smart door/window sensor detects unauthorized entry, it can trigger the buzzer to sound an alarm.

   Smart Home Devices: IoT devices in smart homes can utilize buzzers to indicate various events. For example, a smart thermostat can sound a buzzer to notify the user when a specific temperature threshold is reached or when an HVAC system is turned on/off.

   Industrial Monitoring: In industrial IoT applications, buzzers can be employed as audio alarms to alert operators or maintenance personnel about critical events or equipment malfunctions. For instance, if a sensor detects a hazardous condition or a machine failure, it can activate the buzzer to draw immediate attention.

4. **Vibration sensor:**

   A vibration sensor, also known as an accelerometer or vibration detector, is a device that detects and measures vibrations or changes in acceleration. It is commonly used in various applications, including IoT devices, to monitor and analyze vibration levels for different purposes.

   The basic principle of a vibration sensor is to measure the acceleration of an object or structure in one or more directions. It typically consists of a mass or proof mass attached to a spring or some other mechanism that converts mechanical motion into an electrical signal. When subjected to vibration or acceleration, the mass moves, causing a change in the electrical signal.

5. **Node MCU [ESP-8266]:**

The NodeMCU is an open-source development board based on the ESP8266 microcontroller. It is specifically designed for IoT applications and offers built-in Wi-Fi connectivity, making it an ideal choice for connecting devices to the internet and building IoT projects. Here are some key features and aspects of the NodeMCU (ESP8266):

Microcontroller: The NodeMCU board is powered by the ESP8266 microcontroller, which integrates a 32-bit Tensilica processor, providing sufficient processing power for IoT applications. It operates at a clock frequency of 80 MHz, although some versions of the ESP8266 can be overclocked to 160 MHz.

Wi-Fi Connectivity: One of the main advantages of the NodeMCU is its built-in Wi-Fi capability. It allows the board to connect to wireless networks, access the internet, and communicate with other devices over Wi-Fi. This feature is essential for IoT projects that require internet connectivity or remote control.

Programming Language: The NodeMCU board can be programmed using the Arduino IDE (Integrated Development Environment). It provides a familiar programming environment for developers and allows them to leverage the extensive Arduino ecosystem and libraries. Additionally, the NodeMCU can be programmed using Lua scripting language, which is the default language for the NodeMCU firmware.

GPIO Pins: The NodeMCU board offers a number of General Purpose Input/Output (GPIO) pins, which can be used to interface with various electronic components and sensors. These pins can be used for digital input/output, analog input, or for communication protocols like I2C and SPI.

Flash Memory: The ESP8266 chip on the NodeMCU board comes with onboard flash memory, which can be used for storing the firmware, web pages, and other data. The available flash memory capacity may vary depending on the specific version of the NodeMCU board.

USB Connectivity: The NodeMCU board features a micro USB port for power supply and serial communication. It allows easy connection to a computer for programming and debugging purposes.

Open-Source Community: The NodeMCU is an open-source project, which means the design files, firmware, and software libraries are freely available. This fosters an active and supportive community that continuously develops and shares projects, tutorials, and resources.

The NodeMCU (ESP8266) is widely used for prototyping IoT projects, home automation systems, sensor networks, and other applications that require internet connectivity. Its affordability, ease of use, and extensive community support have made it a popular choice among IoT enthusiasts and developers.

6. **Wire:**
   In the context of IoT and electronics, the terms "male" and "female" are often used to describe connectors or terminals on wires or cables that are used to establish electrical connections between different components. Here's an explanation of male and female wires in IoT:

   Male Connector: A male connector typically has one or more exposed pins or prongs that can be inserted into corresponding female connectors. The pins or prongs are designed to fit into the slots or openings of the female connector, establishing an electrical connection. Male connectors are often considered the "plug" or the source of the connection.

   Female Connector: A female connector has corresponding slots or openings that are designed to receive the pins or prongs of a male connector. When a male connector is inserted into a female connector, the pins or prongs make contact with the corresponding slots, creating an electrical connection. Female connectors are often considered the "socket" or the receiving end of the connection.

   In IoT applications, male and female connectors are commonly used to connect various devices, modules, sensors, or components together. For example:

   Sensor Connections: IoT sensors may have male connectors at the end of their wires, which can be plugged into corresponding female connectors on a microcontroller board or IoT device.

Device Interfacing: Male connectors on IoT devices can be used to connect to female connectors on other devices or interfaces, such as USB ports, Ethernet ports, or GPIO headers.

Expansion Modules: In IoT systems, expansion modules or shields may have female connectors that can receive male connectors from other modules or devices, allowing for easy stacking or connection of multiple components.

Objective:

- **Early accident detection**

  refers to the use of technology and systems to detect and respond to accidents or incidents as quickly as possible, often in their early stages or before they escalate into more severe situations. This concept is particularly relevant in the context of road safety and transportation systems

- **Swift Emergency Response**

  Swift emergency response refers to the prompt and efficient actions taken by emergency services, authorities, and relevant personnel in response to an accident, crisis, or emergency situation. It involves a coordinated effort to provide immediate assistance, mitigate risks, and save lives.

- **Location Identification:**

  Location identification refers to the process of determining and identifying the specific geographic position or coordinates of a person, object, or event. It is a crucial aspect in emergency response, navigation systems, logistics, and various applications in the realm of IoT.

- **Automatic Notification:**

  Automatic notification refers to the process of generating and sending notifications or alerts without manual intervention. It involves the use of technology and systems to detect specific events or conditions and trigger notifications to designated individuals or groups. Automatic notifications play a vital role in various scenarios, including emergency situations, system monitoring, and communication in IoT applications.

- **Post-Accident Assistance**

  Post-accident assistance refers to the support and services provided to individuals and parties involved in an accident to help them cope with the aftermath and navigate the necessary processes.

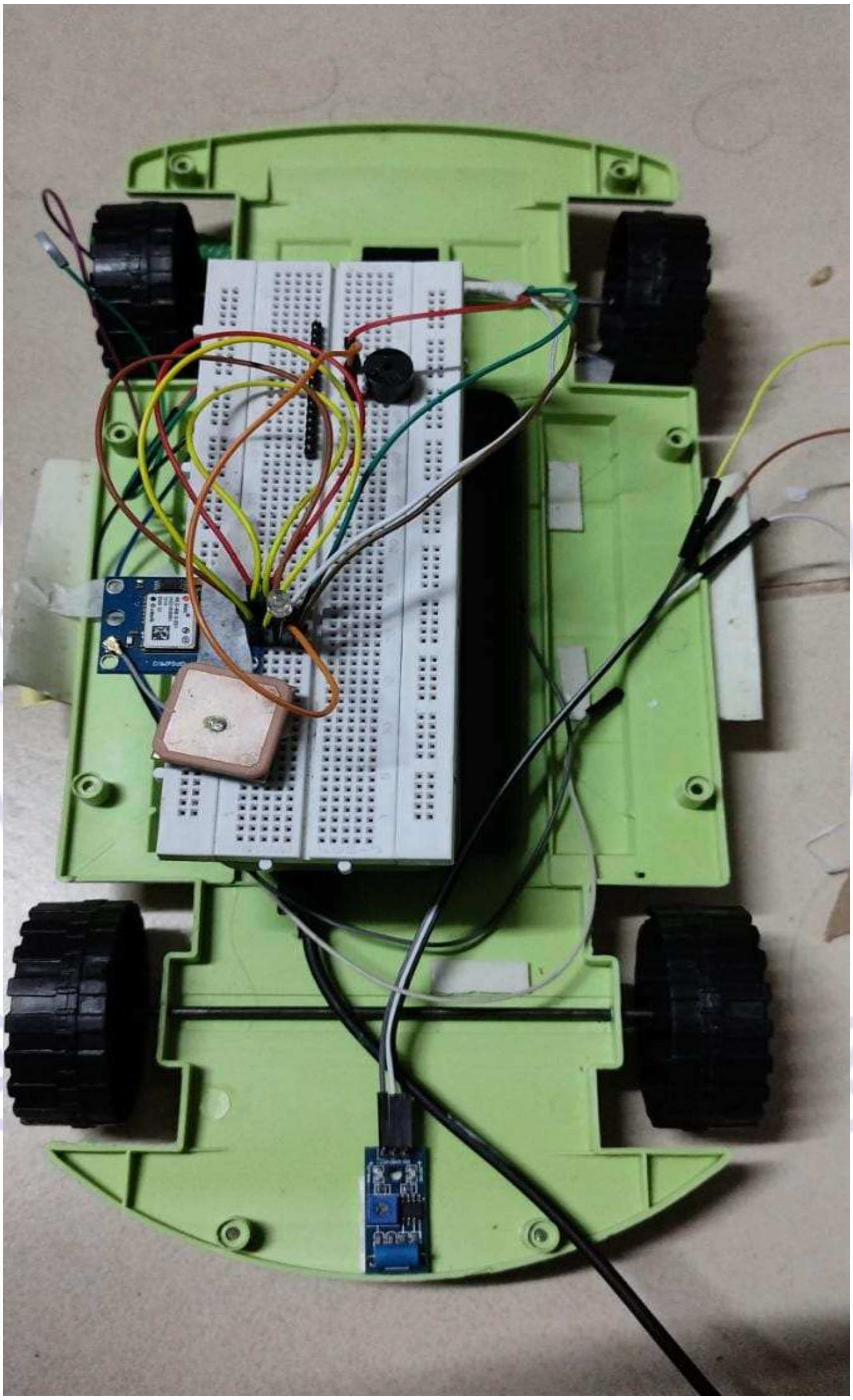
CODE:

```
#include<TinyGPS++.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <UrlEncode.h>
int temp= 0;
int vib_pin = D1;
int flag = 0;
int buzzer = D2;
int counter;
SoftwareSerial GPS_SoftSerial(D4, D3);
TinyGPSPlus gps;
const char* ssid = "Wifi enduku ra";
const char* password = "check123";
String phoneNumber = "+919700662454";
String apiKey = "5371501";
void sendMessage(String message)
{
String url = "http://api.callmebot.com/whatsapp.php?phone=" + phoneNumber
+ "&apikey=" + apiKey + "&text=" +
urlEncode(message);
WiFiClient client;
HTTPClient http;
http.begin(client, url);
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
int httpResponseCode = http.POST(url);
```

```
if (httpResponseCode == 200)
{
Serial.print("Message sent successfully");
}
else
{
Serial.println("Error sending the message");
Serial.print("HTTP response code: ");
Serial.println(httpResponseCode);
}
http.end();
}
void setup()
{
pinMode(vib_pin,INPUT);
pinMode(buzzer,OUTPUT);
Serial.begin(9600);
Serial.println("start");
GPS_SoftSerial.begin(9600);
Serial.println("end");
WiFi.begin(ssid, password);
Serial.print("Connecting to Wifi");
while(WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.print("Connected to WiFi network with IP Address: ");
Serial.println(WiFi.localIP());
}
void loop()
{
int val;
val=digitalRead(vib_pin);
if(digitalRead(vib_pin) == 1)
{
Serial.println("Collision Detected !! Fetching location...");
digitalWrite(buzzer,HIGH);
while(flag == 0)
{
unsigned long start = millis();
do
{
while (GPS_SoftSerial.available())
gps.encode(GPS_SoftSerial.read());
} while (millis() - start < 1000);
double lat_val, lng_val, alt_m_val;
uint8_t hr_val, min_val, sec_val;
bool loc_valid, alt_valid, time_valid;
lat_val = gps.location.lat();
loc_valid = gps.location.isValid();
lng_val = gps.location.lng();
if (!loc_valid)
{
```

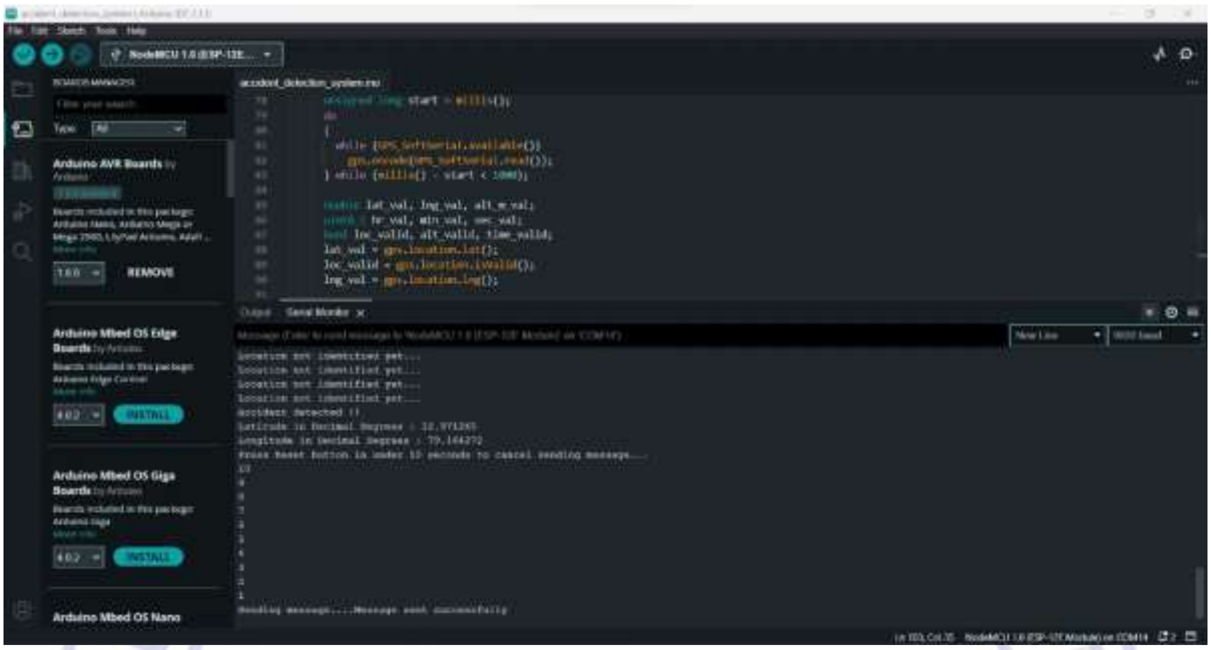
```
Serial.println("Location not identified yet...");
delay(1000);
}
else
{
flag = 1;
Serial.println("Accident detected !!");
Serial.print("Latitude in Decimal Degrees : ");
Serial.println(lat_val, 6);
Serial.print("Longitude in Decimal Degrees : ");
Serial.println(lng_val, 6);
Serial.println("Press Reset button in under 10 seconds to cancel sending
message...");
counter = 10;
digitalWrite(buzzer,HIGH);
while (counter > 0)
{
Serial.println(counter);
delay(1000);
counter = counter-1;
}
digitalWrite(buzzer,LOW);
Serial.print("Sending message....");
sendMessage("Accident Detected !! Location:
https://maps.google.com/?q="+String(lat_val,6)+"," + String(lng_val,6));
}
}
flag = 0;}
```

Setup :



MODEL:

Result:



```
unsigned long start = millis();
do
{
  while (GPS_SoftSerial.available())
    gps.encode(GPS_SoftSerial.read());
} while (millis() - start < 1000);

double lat_val, lng_val, alt_m_val;
uint8_t hr_val, min_val, sec_val;
bool loc_valid, alt_valid, time_valid;
lat_val = gps.location.lat();
loc_valid = gps.location.isValid();
lng_val = gps.location.lng();
```

```
Location not identified yet...
Location not identified yet...
Location not identified yet...
Location not identified yet...
Accident detected !!
Latitude in Decimal Degrees : 12.971265
Longitude in Decimal Degrees : 79.164272
Press Reset button in under 10 seconds to cancel sending message...
10
9
8
7
6
5
4
3
2
1
Sending message....Message sent successfully
```

A collision detection system experiment typically involves testing the effectiveness and accuracy of a collision detection algorithm or technology. The purpose is to evaluate the system's ability to detect and respond to potential collisions between objects or vehicles. Here's a general outline of what the experiment might involve and the expected results:

Experimental Setup: The experiment would involve setting up a controlled environment or scenario where collisions can be simulated. This may include using physical objects, virtual simulations, or a combination of both.

Collision Scenarios: Various collision scenarios would be created, representing different real-world situations. For example, head-on collisions, rear-end collisions, or collisions at intersections. These scenarios would be designed to test the system's ability to detect collisions accurately.

Data Collection: Sensors or detectors integrated into the collision detection system would capture relevant data during the experiment. This may include position, velocity, acceleration, and other parameters of the objects or vehicles involved.

System Performance Evaluation: The collected data would be analyzed to evaluate the system's performance. Key metrics for evaluation may include detection accuracy, response time, false positive/negative rates, and the system's ability to handle various collision scenarios.

Comparison with Ground Truth: The experimental results would be compared against a ground truth or known outcomes to assess the accuracy of the collision detection system. This may involve using reference data or independent measurement systems to validate the system's performance.

Optimization and Improvement: Based on the experiment's results, any identified limitations or areas for improvement in the collision detection system would be addressed. This may involve refining algorithms, adjusting sensor placements, or enhancing system parameters to enhance accuracy and performance.

The expected results of a collision detection system experiment would vary depending on the specific technology, algorithm, or system being tested. The experiment aims to provide insights into the system's effectiveness in accurately detecting collisions and responding appropriately to mitigate potential risks.

It's important to note that real-world performance may differ from experimental results, as the experiment is typically conducted in a controlled environment. Factors such as environmental conditions, sensor limitations, and unpredictable human behavior can impact the system's performance in actual deployment scenarios.

Conclusion:

- The precise location of the vehicle can be easily identified.
- An advanced safety system with intelligent technology.
- It reduces the valuable time needed to rescue accident victims.
- Applicable to various types of vehicles such as motorcycles, cars, and trucks

References:

- Citation Sourav Kumar Panwar et al 2020 J. Phys.: Conf. Ser. 1706 012152
- DOI 10.1088/1742-6596/1706/1/012152
- https://circuitdigest.com/microcontroller-projects/arduino-based-accident-alert-system-using-gps-gsm- accelerometer
- An Automated System for Accident Detection Asad Ali and Mohamad Eid Applied Interactive Multimedia (AIM) LaboratoryDivision of Engineering, New York University Abu Dhabi Abu Dhabi, United Arab Emirates {aa2469, mohamad.eid}@nyu.edu.
- Vehicle Accident Detection and Prevention using IoT and Deep Learning Lakshmy S, Renjith Gopan, Meenakshi M L, Adithya V, Mariya R Elizabeth (Department of Electronics and Communication, Mar Baselios

Drive link : https://drive.google.com/drive/folders/1zX7_SUSsxPq_zaay8p9-rZdWajoHUHJ2?usp=sharing