

# Speed Management System Based on GPS

Nalen G. J.; Varshika K.

Electronic and Instrumentation Engineering  
Kumaraguru College of Technology, Coimbatore

**Abstract:- Excessive speeding by drivers is commonly the cause of traffic accidents, which frequently results in property damage or fatalities. There are still a lot of recorded incidents that occur because of irresponsible driving, even with all the efforts taken to protect drivers, such as enforcing traffic laws and attempting to regulate speed in vital zones by using speed limiters. A speed-limiting control system that would detect irresponsible driving, particularly in dangerous areas, is the project's proposed solution to the current issue. Our system uses the Global Positioning device (GPS) to identify crucial zones and automatically modify the vehicle's speed when it surpasses the speed restriction. The car's speed data is automatically gathered and stored on a cloud platform for future reference and real-time tracking. For commercial uses, such as cabs, taxis, and trucks, this proposed approach would be highly beneficial in continually monitoring vehicles on huge mass and in promoting public safety.**

The speed control mechanism may precisely recognize and react to predefined geographic zones by integrating a GPS module. This method offers a dynamic solution by enabling the customization of speed limitations based on the features of each zone. The system's responsiveness is ensured using real-time GPS data, which enables speed restrictions to be changed instantly as vehicles pass through different zones. At the same time the real time speed data of the vehicle is collected in a cloud platform called ThingSpeak. It is an open-source Internet of Things (IoT) analytics platform that allows us to collect, visualize, and analyze real time data from our connected devices. This data would be helpful for real-time tracking of our vehicle in case of any emergencies and accidents. Our proposed system is mainly for commercial purposes and can be mounted in taxis to ensure public safety. This system could also be found very useful for other commercial vehicles including trucks, buses, tipper trucksetc.

## I. INTRODUCTION

Road accidents represent a persistent and pervasive global challenge, exerting profound impacts on public health, economic stability, and societal well-being. These incidents, often resulting from a complex interplay of factors, encompass not only collisions between vehicles but also involve pedestrians, cyclists, and other vulnerable road users. The significance of road accidents extends beyond mere statistics, as they manifest as a major cause of injuries, fatalities, and disabilities, thereby posing a considerable burden on healthcare systems and economies worldwide.

Through the use of GPS technology, this research attempts to automate the management of vehicle speed when it reaches crucial zones, providing an efficient solution to the given problem. Based on the coordinates given by the GPS module, the system will compare the vehicle's speed in the critical zone to the maximum permitted speed and make the necessary adjustments. Several facets of tracking and navigation have been completely transformed by the introduction of GPS technology. In order to ensure the safety of both cars and pedestrians, restricted speeds are essential in some zones, such as residential neighborhoods, construction sites, and school zones. These particular difficulties are the focus of the proposed system.

## II. OBJECTIVE OF THE PROJECT

- To provide advanced driver assistance system for commercial vehicles.
- To enhance safety and compliance with speed limit in certain areas.

## III. METHODOLOGY

This research aims to implement an effective solution to the given problem. The GPS module attempts to collect the location of the moving vehicle. This location data is given in latitude and longitude values. With the help of this data the speed of the device can be calculated for every meter per second movement of the vehicle. The Microcontroller is used to compare the current speed of the vehicle with the predefined vehicles speed for a specific zone. When the speed of the vehicle exceeds the predefined speed, the MCU initiates an action to reduce the speed of the vehicle by passing on the control to the ECU of the vehicle which immediately reduces the vehicles speed according to the speed limit to be maintained for the certain zone. The real time data is stored in a cloud platform which could be monitored continuously, and an alert will be sent in case of crossing the speed limit. It ensures the safety, and this device acts as a precaution for accidents and ensures the safety of the individual. An alert email will be sent to the head office of the authority in case of any accident detected by the vibration sensor along with the exact location and time detected by the GPS module.

#### IV. BLOCK DIAGRAM

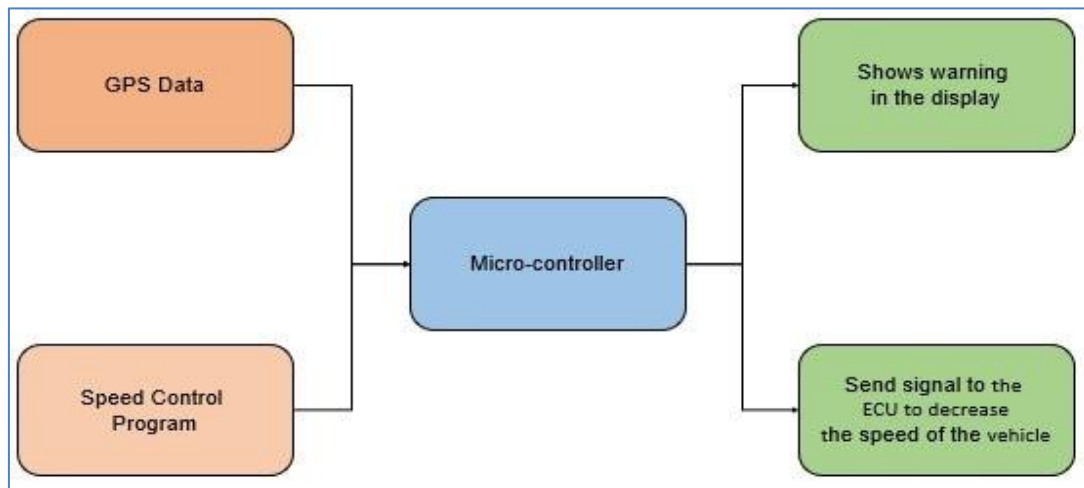


Fig. 1: Block diagram

#### V. COMPONENTS

- ESP8266 Micro-controller
- Neo 7M GPS module
- LCD Display

##### A. ESP8266 Micro-controller

An affordable System-on-a-Chip (SoC) known as the ESP8266 serves as the foundation for the NodeMCU (Node MicroController Unit), which is an open-source software and hardware development environment. The CPU, RAM, networking (Wi-Fi), and even a contemporary operating system and SDK are all present in the Espressif Systems-designed and manufactured ESP8266. It would be a great option for Internet of Things (IoT) projects of all kinds. The ESP8266 has a variety of General-Purpose Input/Output (GPIO) Pins that can be used for various purposes such as interfacing with sensors, actuators and other peripherals. The ESP8266's versatility makes it suitable for countless projects including Smart home devices, IoT sensors, wearables and data logging and monitoring. This model is widely used because of low cost, ease of use and Low power consumption. Fig.2 represents ESP8266 micro controller.

##### ➤ Technical specification

- 32-bit RISC Architecture
- Clock speed - 80MHz
- Flash memory - 4 MB
- RAM - 32 KB
- Operating Voltage - 3.3 V
- Operating Temperature: -40°C to +125°C



Fig. 2: ESP 8266 module

##### B. Neo 7M GPS Module

Fig.3 represents Neo7M GPS Module. This compact electronic device is designed to receive signals from Global Navigation Satellite Systems (GNSS) and determine its precise location on Earth. GPS module contains an antenna that continuously receives radio signals from a network of orbiting GPS satellites. Each satellite transmits precise information about its position and current line. Each signal's arrival time is measured by the GPS module. The speed of light is then used to determine each satellite's distance. Satellite distances are calculated by GPS modules. Using the distances between several known sites (the satellites) to determine a position is known as trilateration. The GPS module can determine its position within two Earthly points using distance data from a minimum of three satellites. The calculated position is expressed in latitude and longitude coordinates:

- Latitude measures the north-south position, relative to the equator (0°).
- Longitude measures the east-west position, relative to the Prime Meridian (0°).

It stands out for its accuracy, versatility, and less complexity, making it a popular choice for various applications. It is widely used in Geotagging, Robotics, Environmental monitoring, and Asset tracking applications.

➤ *Technical Specifications*

- UART communication protocol
- Operating Temperature: -40°C to +85°C
- Working voltage : 3.3v-5v
- Current : 40 MA
- Output frequency: 1hz

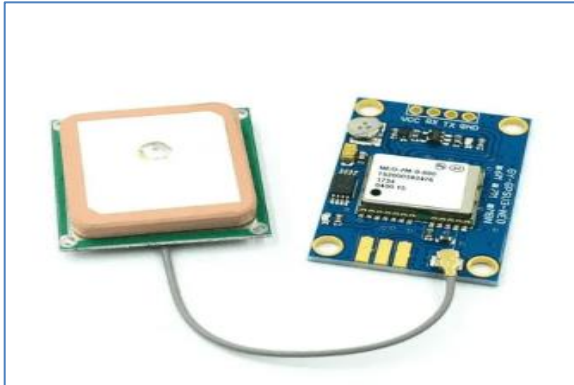


Fig. 3: GPS module

*C. LCD Display*

An electronic gadget that can display text or numbers is called a liquid crystal display. Numerical displays and alphanumeric text displays are the two primary categories of LCD displays. The "crystal" on the display is composed of several shapes. These crystals take the form of "bars" in numeric displays and are merely organized into patterns of "dots" in alphanumeric displays. LCD sizes are varied. The LCD used in our system, model number RG1602A, has a 16x2 character display size, which is a commonly used display size. With its 16-pin layout, it offers multiple features such as power, contrast, control lines, data lines, and backlight. Fig.4 represents LCD display. The 16x2 LCD in the system will show the speed as soon as the car starts, which for example would be displayed as 60 km/h. At some instance when the vehicle enters the speed controlling zone the LCD display will notify the user about the current location. The information will be passed on to proceed the further mechanism in reducing the speed of the vehicle. The 16x2 LCD display will show the speed decrease.

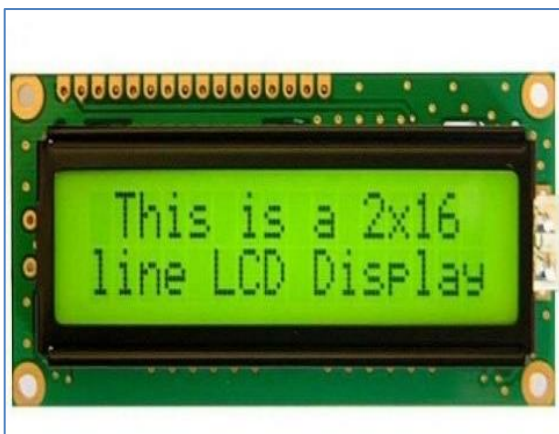


Fig. 4: LCD display

*D. Vibration Sensor*

A vibration sensor is a device that measures the level and characteristics of mechanical vibrations in a system, machine or structure. It converts the physical motion of vibration into an electrical signal providing information about amplitude, frequency and displacement that can be monitored and used for various purposes. There are a wide variety of vibration sensors such as accelerometers, strain gauges and piezoelectric sensors. It also has wide variety of applications including security systems, consumer electronics and biomedical applications. In our project, vibration sensor is used to get the data in case of any vehicle crash or accident. The data collected from the sensor is analyzed for true values and sent to the taxi's head office to indicate that an accident has occurred at the particular location. An email alert is triggered to the Taxi authority office and to the nearby hospital for timely treatment and assistance. Fig.5 represents vibration sensor used.

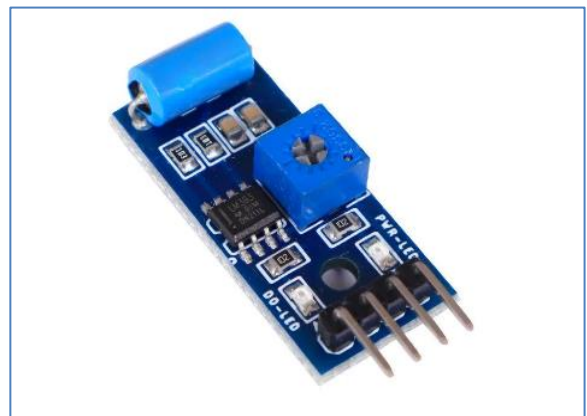


Fig. 5: Vibration Sensor

*E. Thing Speak*

ThingSpeak is an IoT analytics cloud platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. ThingSpeak platform offers a variety of features and is completely user friendly. It collects real time data from our devices. The collected data can be visualised in form of graph and charts. It analyses the collected data and collects insights. We can also control our devices by sending alerts that are commands which alters the vehicles speed when it exceeds the speed in certain zone. It also has a wide range of applications including Agriculture, Environmental monitoring, Smart homes and Asset tracking. Fig.6 represents the working of ThingSpeak IoT cloud platform.

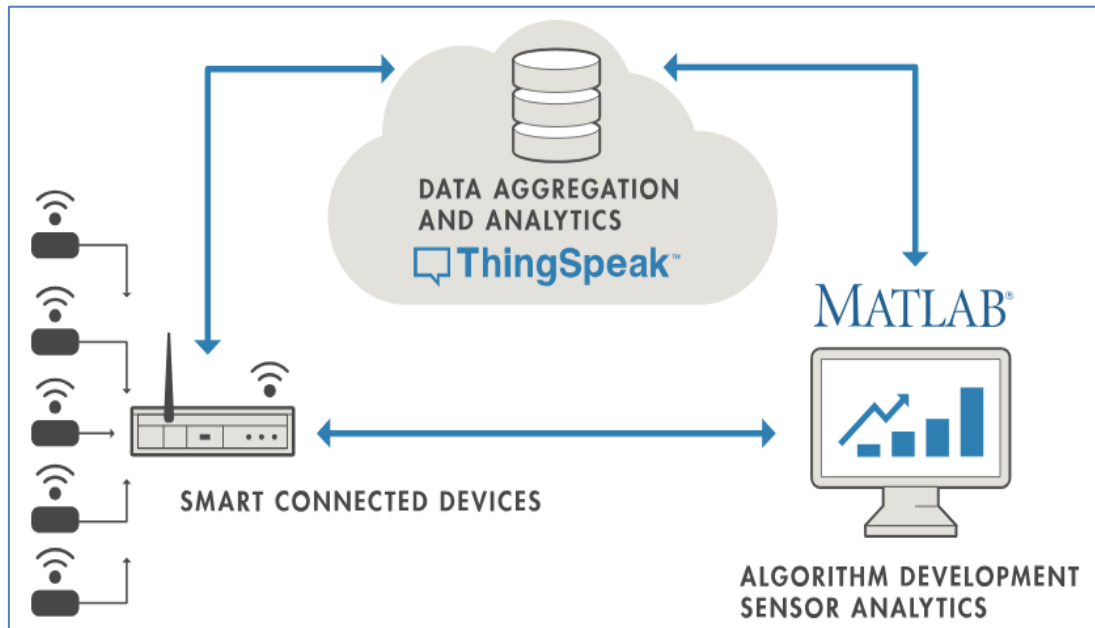


Fig. 6: Thing Speak block diagram

### VI. PROPOSED SYSTEM

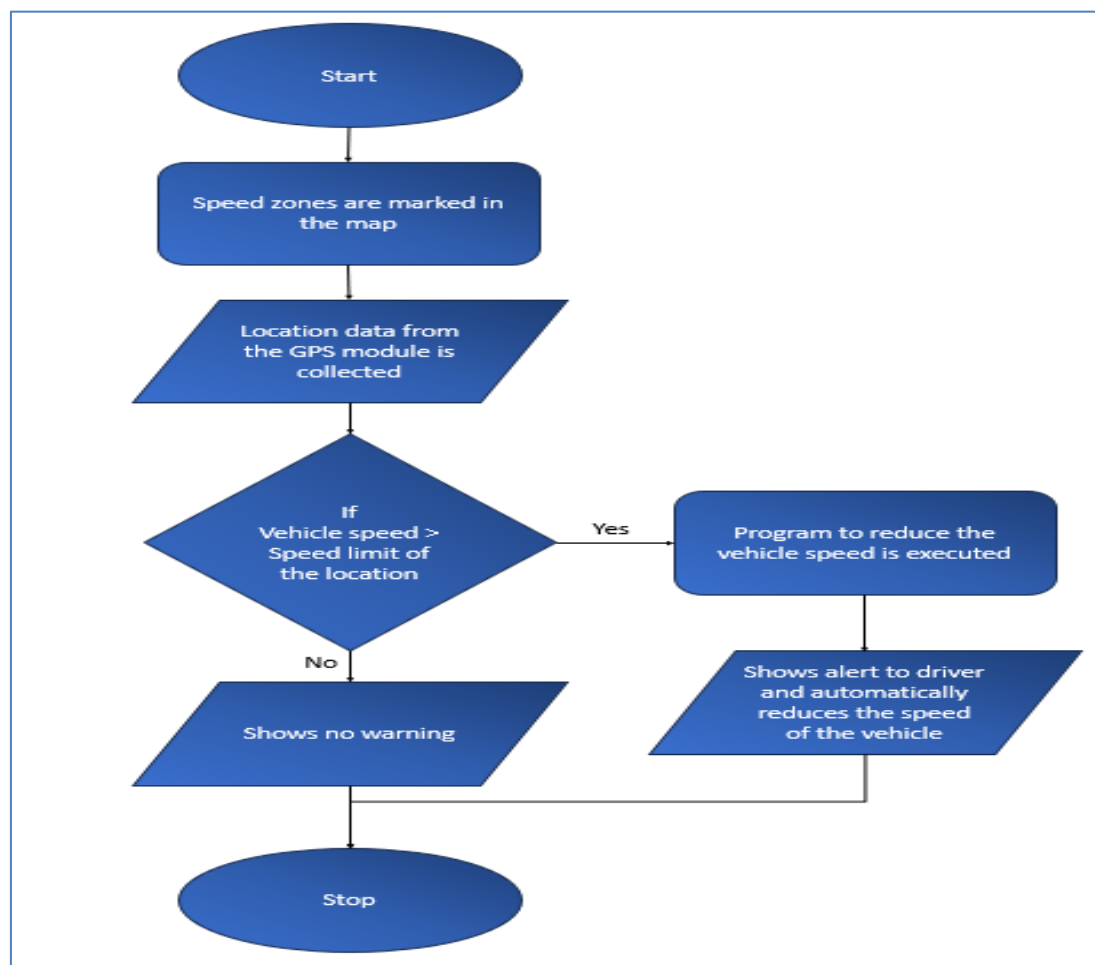
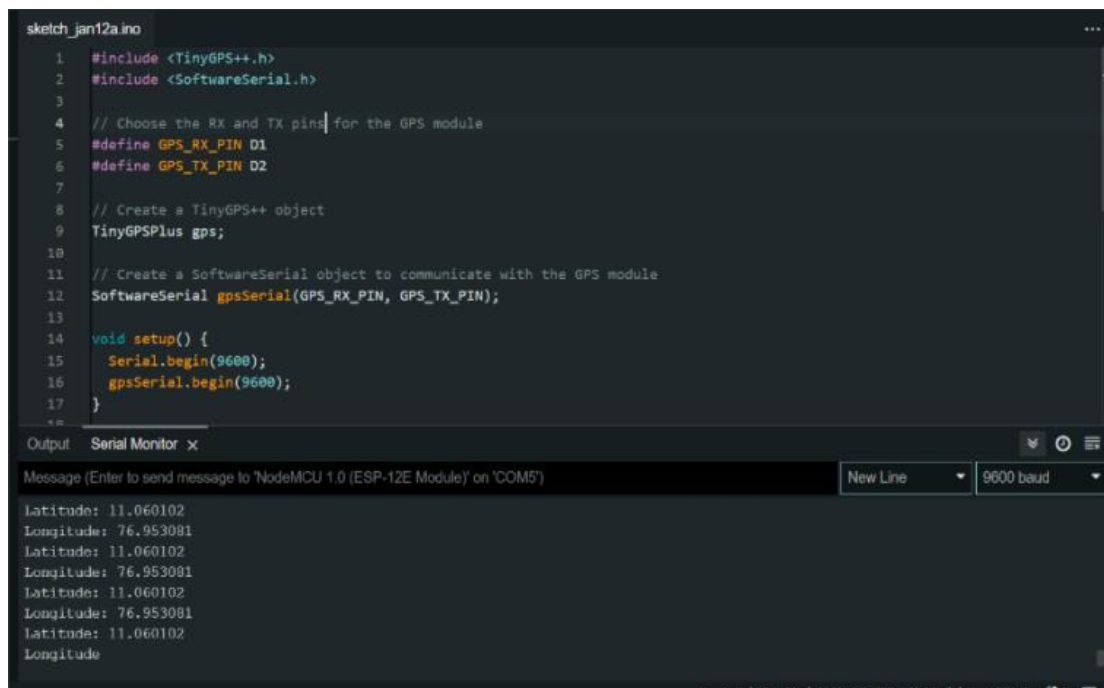


Fig. 7: Flowchart

## VII. TESTING

We practically implemented our project idea to testing. Firstly, we connected the GPS module with the NodeMCU ESP8266 micro controller to get the exact location or the zone where the vehicle lies in. GPS module contains an antenna that continuously receives radio signals from a network of orbiting GPS satellites. The exact position of the vehicle is expressed in latitude and longitude coordinates. The result is shown as an output where the location is received from the GPS module. Fig.8 represents the test output of the GPS. Based on the location coordinates the zones are divided, in which the zones where the speed to be controlled is considered. We tested this by

keeping the location of the vehicle inside the specified location i.e., school zone. Now the micro controller compares the current location and target location, and it is identified as school zone. Now the speed of the vehicle is monitored. If the speed of the vehicle is under the predefined speed limit there is no change in speed of the vehicle, but if the speed exceeds the predefined speed limit the LCD displays the warning to the driver and the micro controller sends the signal to reduce the speed of the vehicle. Fig.9 represents the output of the code. Also, the speed data of the vehicle is collected and sent to a cloud platform called ThingSpeak where the speed of the vehicle could be monitored in real time. The speed is normalized once the vehicle crosses the speed limit zone.



```

sketch_jan12a.ino
1 #include <TinyGPS++.h>
2 #include <SoftwareSerial.h>
3
4 // Choose the RX and TX pins for the GPS module
5 #define GPS_RX_PIN D1
6 #define GPS_TX_PIN D2
7
8 // Create a TinyGPS++ object
9 TinyGPSPlus gps;
10
11 // Create a SoftwareSerial object to communicate with the GPS module
12 SoftwareSerial gpsSerial(GPS_RX_PIN, GPS_TX_PIN);
13
14 void setup() {
15   Serial.begin(9600);
16   gpsSerial.begin(9600);
17 }

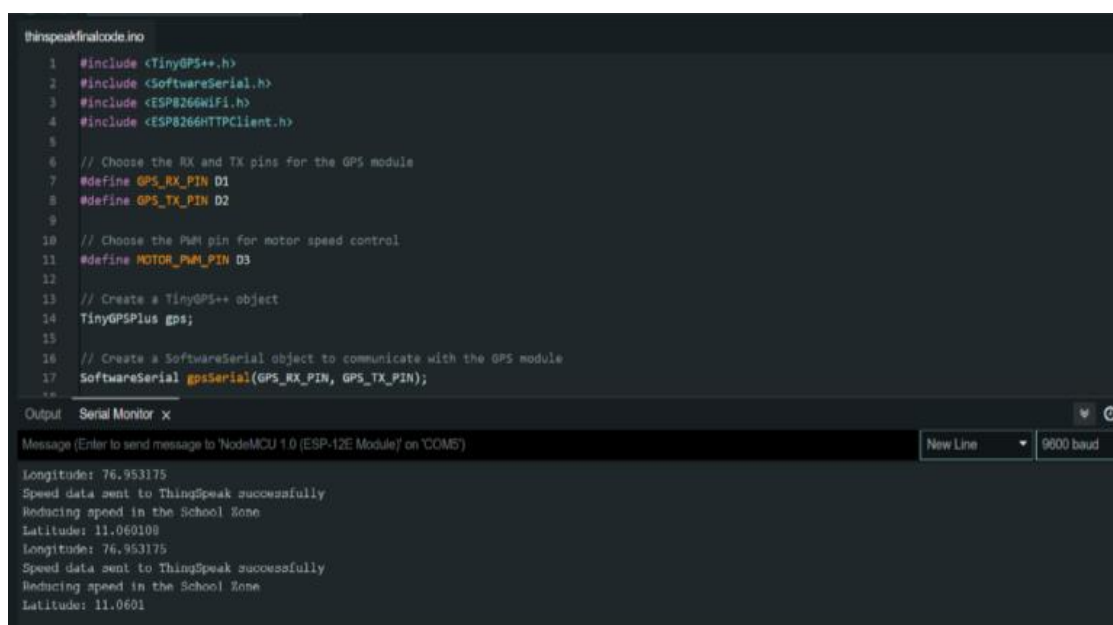
```

Output Serial Monitor x

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM5')

Latitude: 11.060102  
Longitude: 76.953081  
Latitude: 11.060102  
Longitude: 76.953081  
Latitude: 11.060102  
Longitude: 76.953081  
Latitude: 11.060102  
Longitude

Fig. 8: Arduino Code Output



```

thingspeakfinalcode.ino
1 #include <TinyGPS++.h>
2 #include <SoftwareSerial.h>
3 #include <ESP8266WiFi.h>
4 #include <ESP8266HTTPClient.h>
5
6 // Choose the RX and TX pins for the GPS module
7 #define GPS_RX_PIN D1
8 #define GPS_TX_PIN D2
9
10 // Choose the PWM pin for motor speed control
11 #define MOTOR_PWM_PIN D3
12
13 // Create a TinyGPS++ object
14 TinyGPSPlus gps;
15
16 // Create a SoftwareSerial object to communicate with the GPS module
17 SoftwareSerial gpsSerial(GPS_RX_PIN, GPS_TX_PIN);

```

Output Serial Monitor x

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM5')

Longitude: 76.953175  
Speed data sent to ThingSpeak successfully  
Reducing speed in the School Zone  
Latitude: 11.060108  
Longitude: 76.953175  
Speed data sent to ThingSpeak successfully  
Reducing speed in the School Zone  
Latitude: 11.0601

Fig. 9: Arduino Code Output

**VIII. THING SPEAK CLOUD DATA**

The data from the ESP8266 module is sent to ThingSpeak IoT cloud platform. Data such as speed of the vehicle, data from the vibration sensor, latitude and longitude of the vehicle is collected in different fields in the ThingSpeak channel. The speed of the vehicle is plotted in

the graph which can be used to analyse the vehicle speed data for the future purposes and this speed is also displayed in the real time graphics for better understanding. The data is updated every 10 seconds whenever the vehicle is switched ON. Fig 10, 11 and 12 represents the data collected in the ThingSpeak cloud platform.

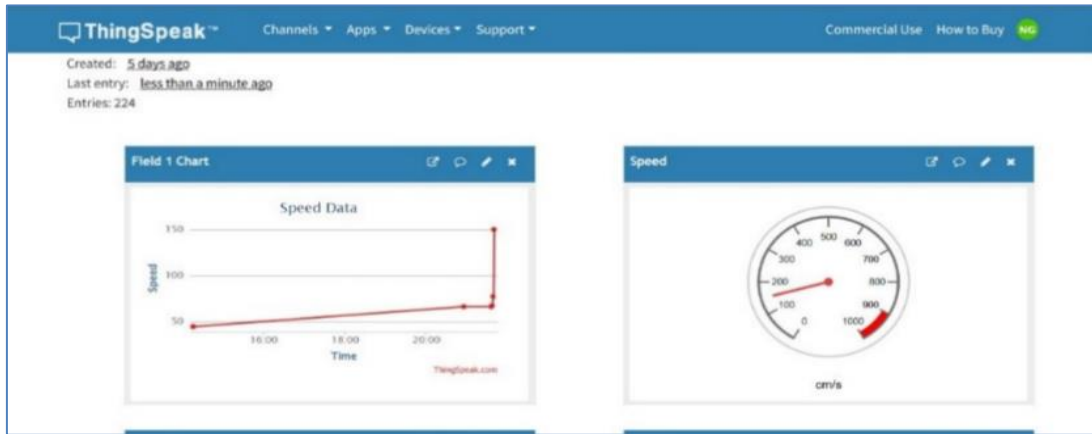


Fig. 10: Thing Speak Speed Graph

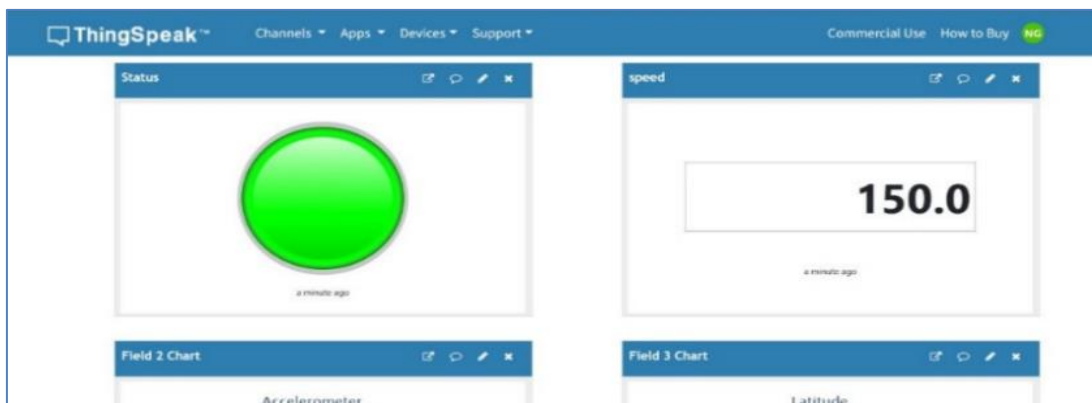


Fig. 11: Vehicle Status Data in Thing Speak

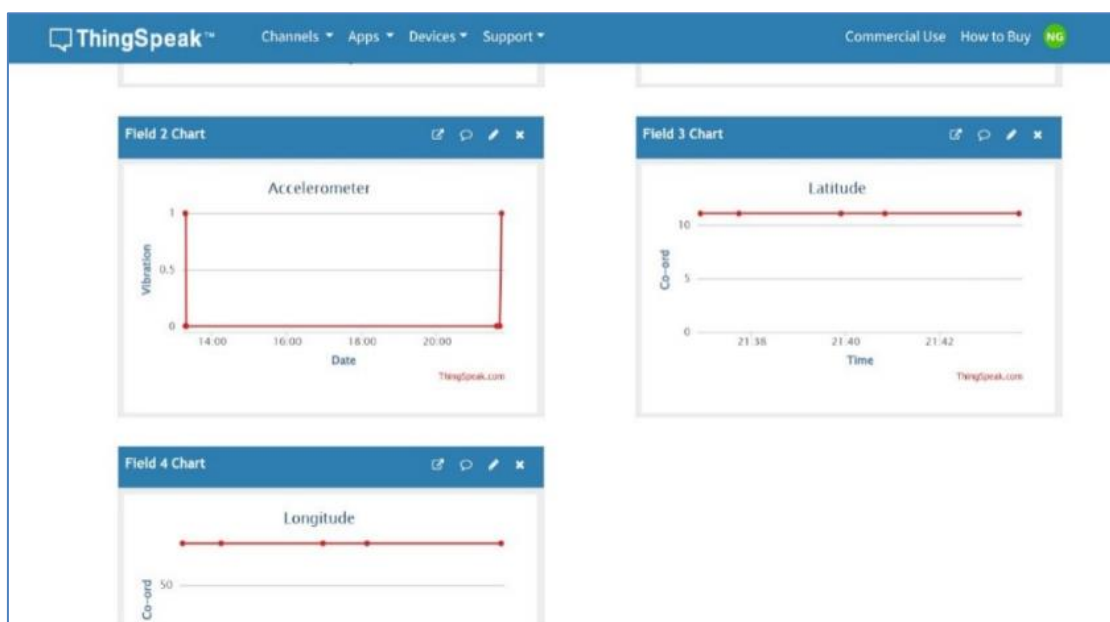


Fig. 12: Latitude, Longitude Graph

## IX. ACCIDENT ALERT NOTIFICATION

When an accident occurs to the vehicle, vibration sensor detects the accident and sends this data to ThingSpeak. Fig.14 represents the output of the vibration sensor. Using MATLAB Analysis, the data from the vibration sensor is analysed. The MATLAB analysis is programmed in such a way that the data from the vibration sensor is monitored continuously and whenever an

accident is detected an email alert is automatically sent to the head office in case of emergency along with the exact location coordinates and time where the accident has occurred. As an additional application Hospitals in the area are notified when a passenger needs emergency rescue. The hospital confirms the accident and authorizes it after verifying it at the designated site. It would help to offer timely assistance and care to the taxi driver as well as the passenger. Fig.13 represents the email alert received.

### A. MATLAB code

```

1 channelID = 2402663;
2
3 % Provide the ThingSpeak alerts API key. All alerts API keys start with TAK.
4 alertApiKey = 'TAKqp8beG3vbnDbhInh';
5
6 % Set the address for the HTTP call
7 alertUrl='https://api.thingspeak.com/alerts/send';
8
9 % webwrite uses weboptions to add required headers. Alerts needs a ThingSpeak-Alerts-API-Key
10 options = weboptions("HeaderFields", ["ThingSpeak-Alerts-API-Key", alertApiKey]);
11
12 % Set the email subject.
13 alertSubject = sprintf("Accident alert");
14
15 % Read the recent data.
16 speed = thingSpeakRead(channelID,'Fields',1);
17 lat = thingSpeakRead(channelID,'Fields',3);
18 lng = thingSpeakRead(channelID,'Fields',4);
19 % Check to make sure the data was read correctly from the channel.
20
21 alertBody = [sprintf('Speed: %.2f\n', speed),sprintf('Latitude: %.6f\n', lat),
22             sprintf('Longitude: %.6f\n', lng)];
23
24 % Catch errors so the MATLAB code does not disable a TimeControl if it fails
25 try
26     webwrite(alertUrl , "body", alertBody, "subject", alertSubject, options);
27 catch someException
28     fprintf("Failed to send alert: %s\n", someException.message);
29 end

```

### B. Email alert

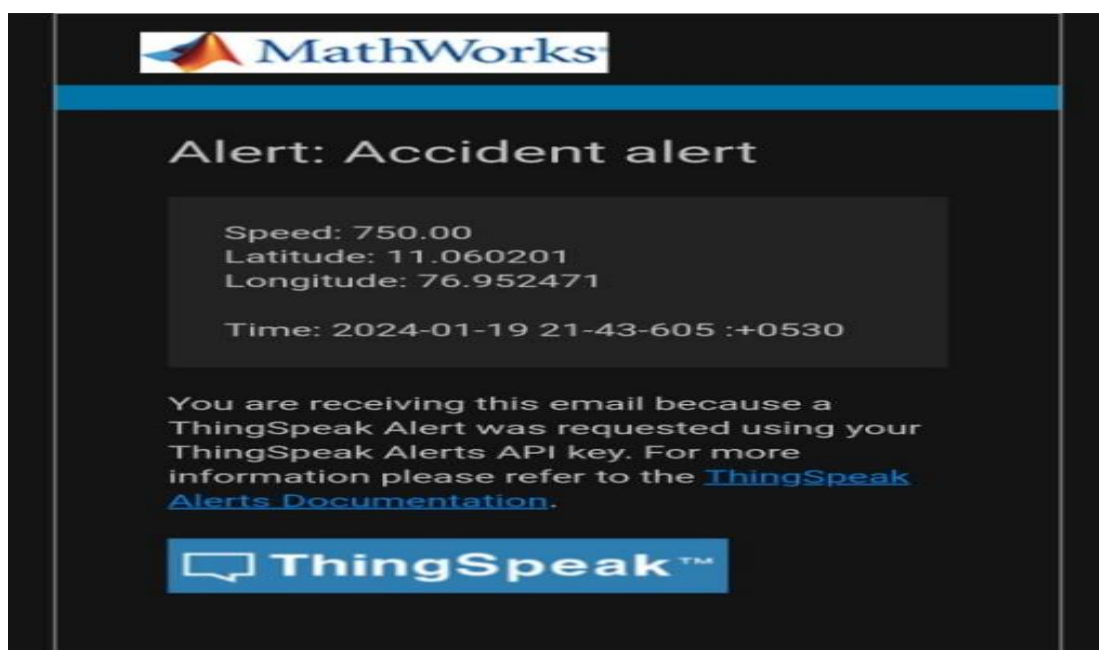


Fig. 13: Alert mail

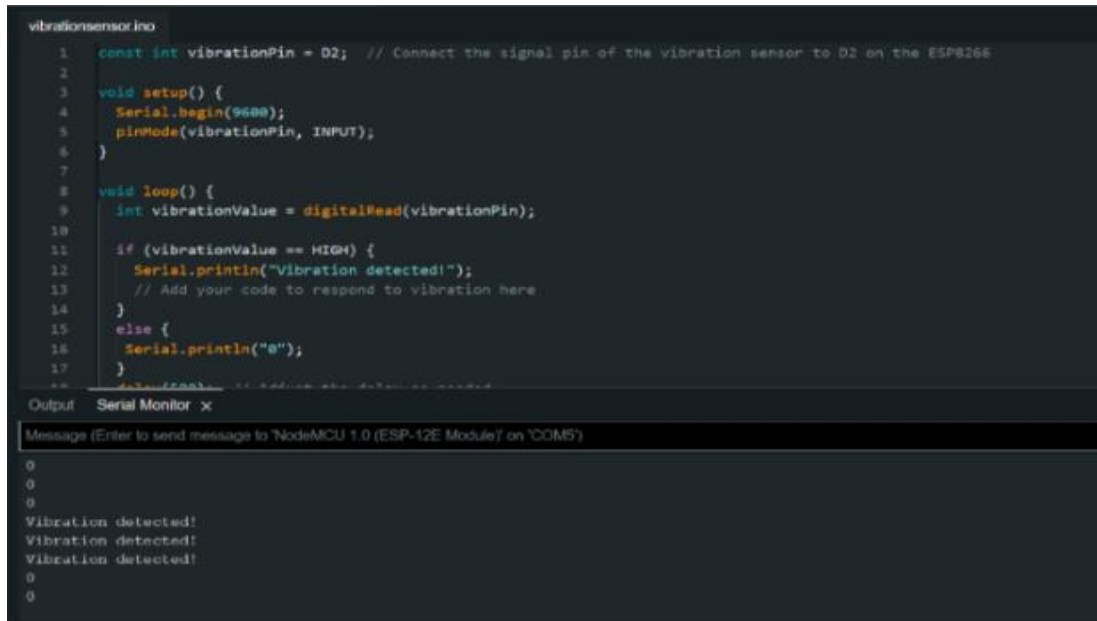


Fig. 14: Vibration Sensor Detects Accident

**X. SOFTWARE IMPLEMENTATION**

```

#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
// Choose the RX and TX pins for the GPS module
#define GPS_RX_PIN D1
#define GPS_TX_PIN D2
#define VIBRATION_SENSOR_PIN D3
#define MOTOR_PWM_PIN D4
// Create a TinyGPS++ object
TinyGPSPlus gps;

// Create a SoftwareSerial object to communicate with
the GPS module
SoftwareSerial      gpsSerial(GPS_RX_PIN,
GPS_TX_PIN);

// Define a struct to represent a location with a name
and coordinates
struct Location {
    const char* name;
    double latitude;
    double longitude;
};

// Set the locations with names and coordinates
Location locations[] = {
    {"School Zone ", 11.0600, 76.9528}, // School
    {"Hospital Zone", 12.3456, 78.9012} // Hospital
// Add more locations as needed
};

// Set the maximum speed allowed in the specific zone
(in meters per second)
float maxAllowedSpeed = 10.0; // Adjust as needed

// ThingSpeak settings

```

```

const char *ssid = "YourWiFiSSID";
const char *password = "YourWiFiPassword";
const char *thingSpeakApiKey =
"0ABUCI59LXX7EX2S";
const char *thingSpeakAddress =
"api.thingspeak.com";
// Function prototypes
bool isInZone(double currentLatitude, double
currentLongitude, Location targetLocation);
void controlSpeedInZone(float currentSpeed, const
char* locationName);
void sendToThingSpeak(float speed, int
vibrationValue, double currentLatitude, double
currentLongitude);

void setup() {
    Serial.begin(9600);
    gpsSerial.begin(9600);
    pinMode(MOTOR_PWM_PIN, OUTPUT);
    pinMode(VIBRATION_SENSOR_PIN, INPUT);
    // Connect to WiFi
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi...");
    }
    Serial.println("Connected to WiFi");
}

void loop() {
    // Read GPS data
    while (gpsSerial.available() > 0) {
        if (gps.encode(gpsSerial.read())) {
            // Check if GPS data is valid
            if (gps.location.isValid()) {
                // Print the latitude, longitude, and altitude
                Serial.print("Latitude: ");
                Serial.println(gps.location.lat(), 6);
            }
        }
    }
}

```



```

Serial.print("Longitude: ");
Serial.println(gps.location.lng(), 6);
int vibrationValue =
digitalRead(VIBRATION_SENSOR_PIN);
if (sensorValue == HIGH)
{
Serial.println("Vibration detected!");
} else
{
Serial.println("No vibration");
}

// Check if the vehicle is in any of the specific
zones
for (int i = 0; i < sizeof(locations) /
sizeof(locations[0]); i++) {
if (isInZone(gps.location.lat(),
gps.location.lng(), locations[i])) {
float currentSpeedCmPerSec =
gps.speed.isValid() ? (gps.speed.mps() * 100) : 0.0;
handleVibration();
// Send speed data to ThingSpeak
sendToThingSpeak(currentSpeedCmPerSec,
vibrationValue, gps.location.lat(), gps.location.lng());

// Control the vehicle speed in the specific
zone
controlSpeedInZone(currentSpeedCmPerSec,
locations[i].name);
break; // No need to check other zones if the
current location is in one
}
} else {
// Print "No GPS Fix" if the GPS data is not valid
Serial.println("No GPS Fix");
}
}
}

bool isInZone(double currentLatitude, double
currentLongitude, Location targetLocation) {
// You can customize the criteria for being in the
zone
// For simplicity, we use a small radius around the
target coordinates
double radius = 0.001; // Adjust as needed
return (fabs(currentLatitude - targetLocation.latitude)
< radius) &&
(fabs(currentLongitude -
targetLocation.longitude) < radius);
}

void controlSpeedInZone(float currentSpeed, const
char* locationName) {
if (currentSpeed > maxAllowedSpeed) {
// Implement your speed control logic here
// For example, reduce motor speed or activate
brakes

```

```

int motorSpeed = map(currentSpeed, 0,
maxAllowedSpeed, 0, 255); // Map speed to motor PWM
range
analogWrite(MOTOR_PWM_PIN, motorSpeed);
Serial.print("Reducing speed in the ");
Serial.println(locationName);
} else {
// If speed is within the allowed range, stop the
motor
analogWrite(MOTOR_PWM_PIN, 0);
}

void sendToThingSpeak(float speed, int vibration,
double currentLatitude, double currentLongitude) {
if (WiFi.status() == WL_CONNECTED) {
HTTPClient http;
WiFiClient client; // Declare a WiFiClient variable

// ThingSpeak API endpoint
String url = "http://" + String(thingSpeakAddress) +
"/update?api_key=" + String(thingSpeakApiKey) +
"&field1=" + String(speed) + "&field2=" +
String(vibrationValue) + "&field3=" +
String(currentLatitude) + "&field4=" +
String(currentLongitude);

// Use WiFiClient variable to begin HTTP request
http.begin(client, url);

// Make the request
int httpCode = http.GET();

// Check the response
if (httpCode == HTTP_CODE_OK) {
Serial.println("Speed data sent to ThingSpeak
successfully");
} else {
Serial.println("Failed to send speed data to
ThingSpeak");
Serial.println(httpCode);
}

http.end();
} else {
Serial.println("Not connected to WiFi");
}
}

```

## XI. CONCLUSION

The discussed system can be designed to control the speed of vehicle in specific zones to avoid the accidents in the low-speed areas. By comparing the initial speed data in specific zone with vehicles speed to avoid accidents and over speeding of vehicles. The proposed system is mainly for commercial purposes such as for taxis and other public means of transport where prior safety is completely necessary. So hence the vehicle speed controller automatically helps in lowering the speed and therefore this

system has heavy impacts to reduce the lives of people by accidents.

### REFERENCES

- [1]. You-Ren Chen, Keng-Pin Chen, and Pao-Ann Hsiung presented the “Traffic Light Optimization and Control System” published in 19th International Conference on Intelligent Transportation System (ITSC) IEEE2016.
- [2]. Himesh Gupta and Aditya Pundir presented the “RF Module Based Speed Check and Seat Belt Detection System” published in Second International Conference on Computational Intelligence & Communication Technology IEEE2016.
- [3]. Christoph Kandler and Tim Koenings presented the “Stability Investigation of an Idle Speed Control Loop for a Hybrid Electric Vehicle”. This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination. IEEE2015.
- [4]. Martin Treiber and Arne Kesting presented the “Automatic and efficient driving strategies. While approaching a traffic light” published in 17<sup>th</sup> International Conference on Intelligent Transportation Systems (ITSC) IEEE2014.
- [5]. Aamir Sarwar Jahan, and Imdadul Hoq, presented the “GPS Enabled Speed Control Embedded System Speed Limiting Device with Display and Engine Control Interface” published in 2013.
- [6]. K.Govindaraju, S.Boopathi, F.Parvez Ahmed, S.Thulasi Ram and M.Jagadeeshraja presented the “Embedded Based Vehicle Speed Control System Using Wireless Technology” in International Journal Of Innovative Research In Electrical, Electronics, Instrumentation And Control Engineering in 2014.
- [7]. Vennela Priyadarshni, P.Gopi Krishna and K.Sreenivasa Ravi presented the “GPS and GSM Enabled Embedded Vehicle Speed Limiting Device” in Indian Journal of Science and Technology in 2016.