TEAM:- CRACK THE HACK
MEMBERS:-
AYUSH SINGH
JIGYASU PANT
SAHAJ DARGAN

Our Solutions to the given Problem Statement

(1) if the cruising speed is more than 90kmph the system alerts the driver by sending a message “Reduce the speed to around 90 km/h”.

(2) If acceleration exceeds 1.4705 m/s2 system displays “Reduce the acceleration to around 1.4705 m/s^2”.

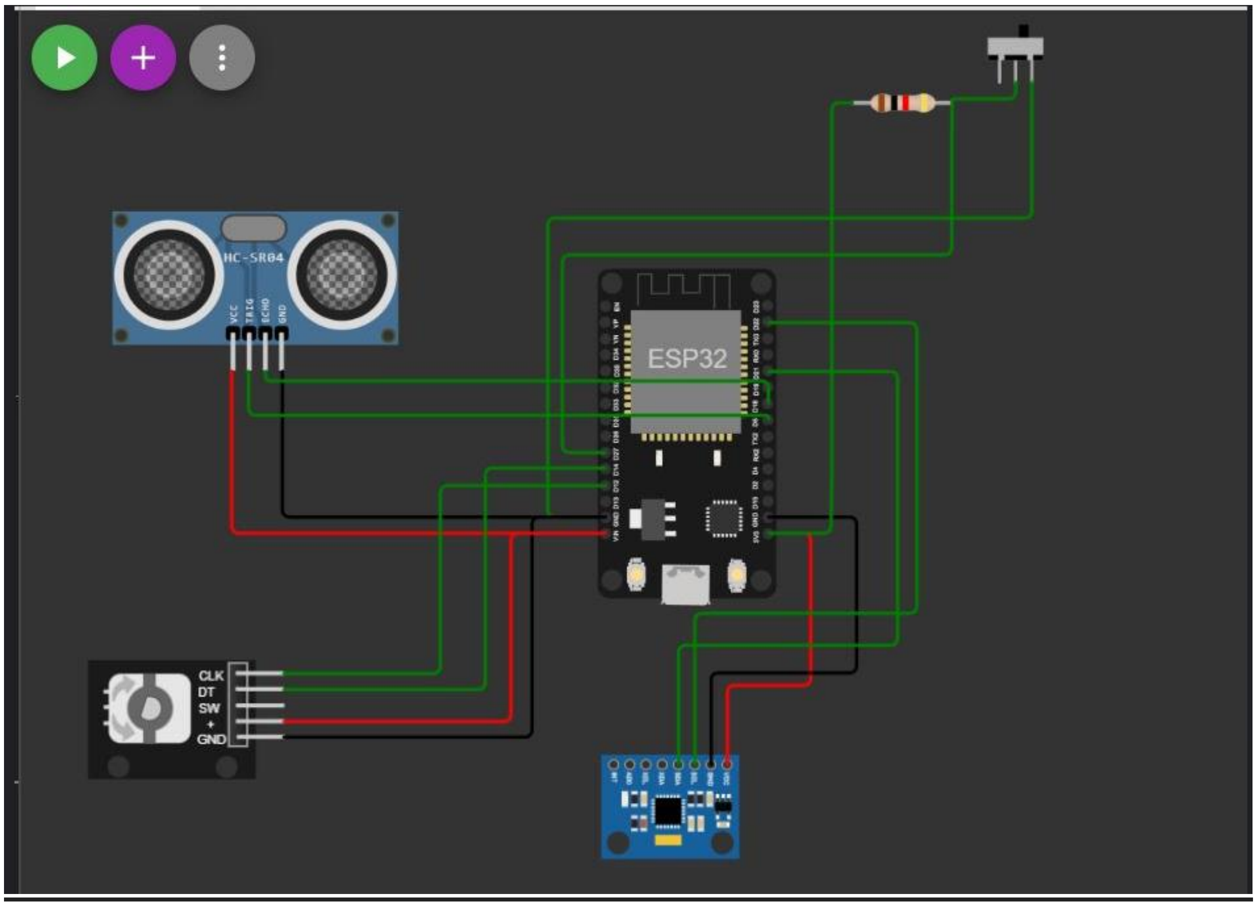
(3) For switching off the vehicle's engine if it remains on until half a minute, we use a switch connected to ESP32.

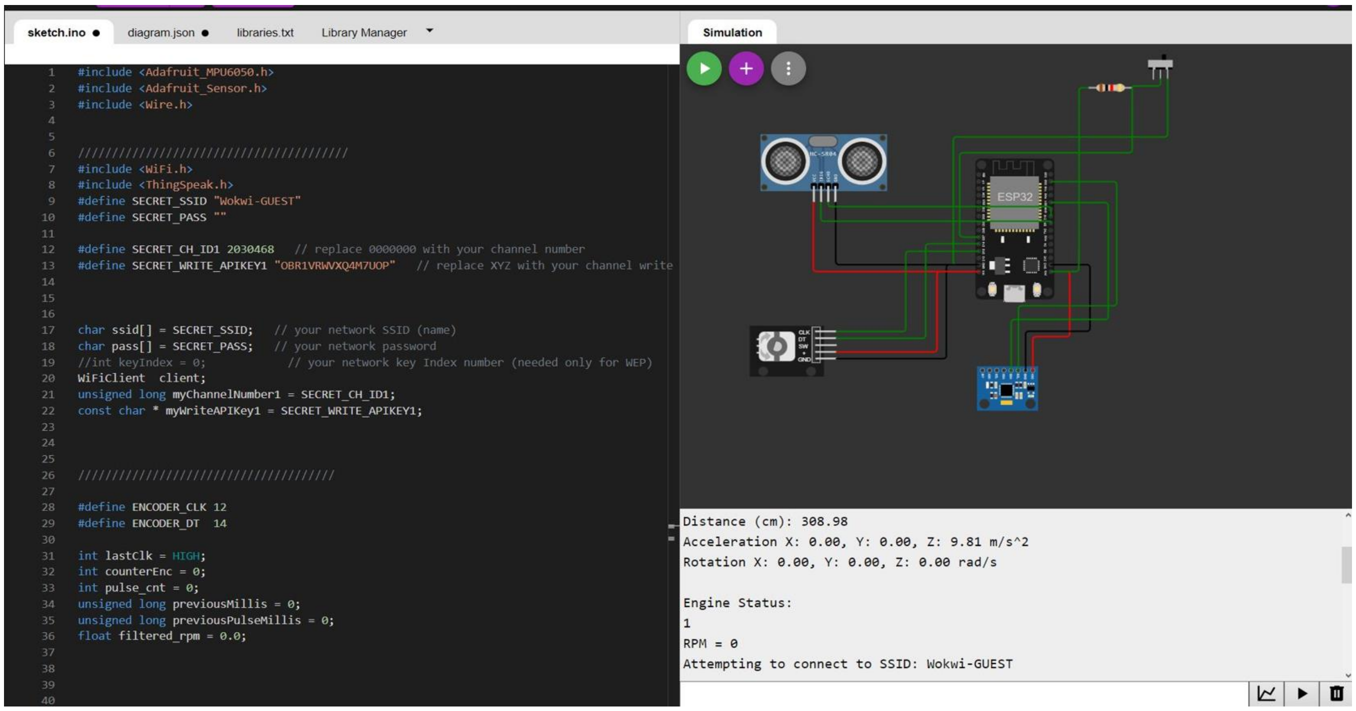
(4) Obstacles in front of the vehicle are detected using an ultrasonic sensor and accordingly the optimal deceleration value is suggested.

(5) We track the location of the vehicle using random GPS locations for the purpose of simulation only. For practical implementation in hardware, we can integrate a GPS module with ESP32 for getting the location.

(6) We have also successfully integrated our Arduino simulation to the cloud storage of Thingspeak using API and Channel ID. On Thingspeak, we have created different graphs of engine status, acceleration, obstacle distance and velocity. The link for the integrated data on Thingspeak is: **https://thingspeak.com/channels/2030468**

CIRCUIT SCREENSHOTS

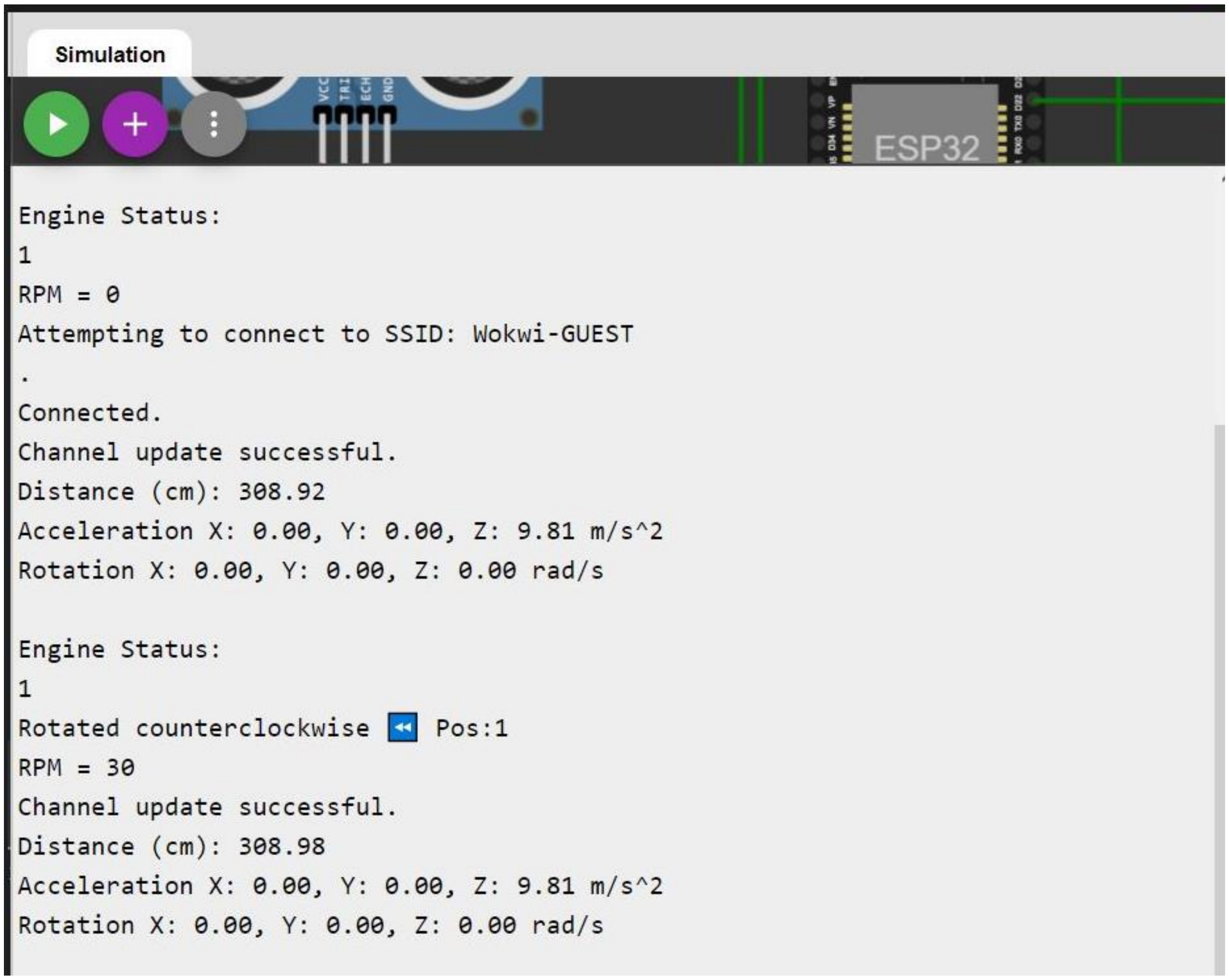




```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>


////////////////////////////////////////
#include <WiFi.h>
#include <ThingSpeak.h>
#define SECRET_SSID "Wokwi-GUEST"
#define SECRET_PASS ""

#define SECRET_CH_ID1 2030468   // replace 0000000 with your channel number
#define SECRET_WRITE_APIKEY1 "OBR1VRWVXQ4M7UOP"   // replace XYZ with your channel write



char ssid[] = SECRET_SSID;   // your network SSID (name)
char pass[] = SECRET_PASS;   // your network password
//int keyIndex = 0;            // your network key Index number (needed only for WEP)
WiFiClient  client;
unsigned long myChannelNumber1 = SECRET_CH_ID1;
const char * myWriteAPIKey1 = SECRET_WRITE_APIKEY1;



//////////////////////////////////////

#define ENCODER_CLK 12
#define ENCODER_DT  14

int lastClk = HIGH;
int counterEnc = 0;
int pulse_cnt = 0;
unsigned long previousMillis = 0;
unsigned long previousPulseMillis = 0;
float filtered_rpm = 0.0;
```

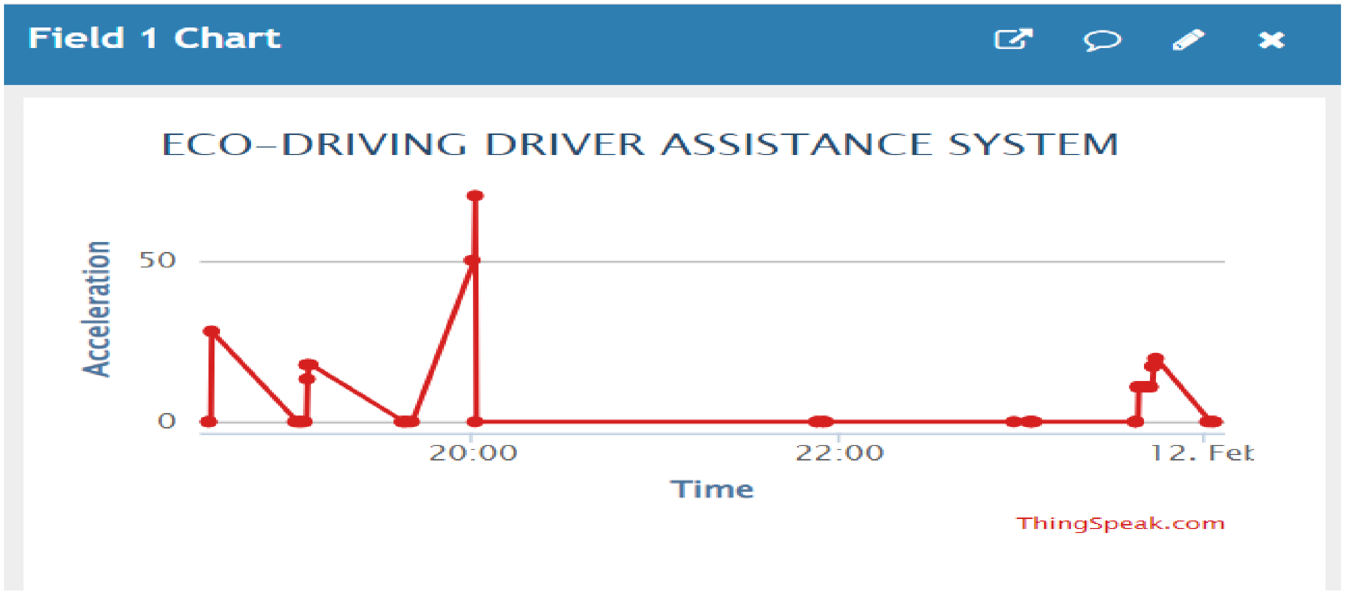
```
Distance (cm): 308.98
Acceleration X: 0.00, Y: 0.00, Z: 9.81 m/s^2
Rotation X: 0.00, Y: 0.00, Z: 0.00 rad/s

Engine Status:
1
RPM = 0
Attempting to connect to SSID: Wokwi-GUEST
```

COMMAND WINDOW:-

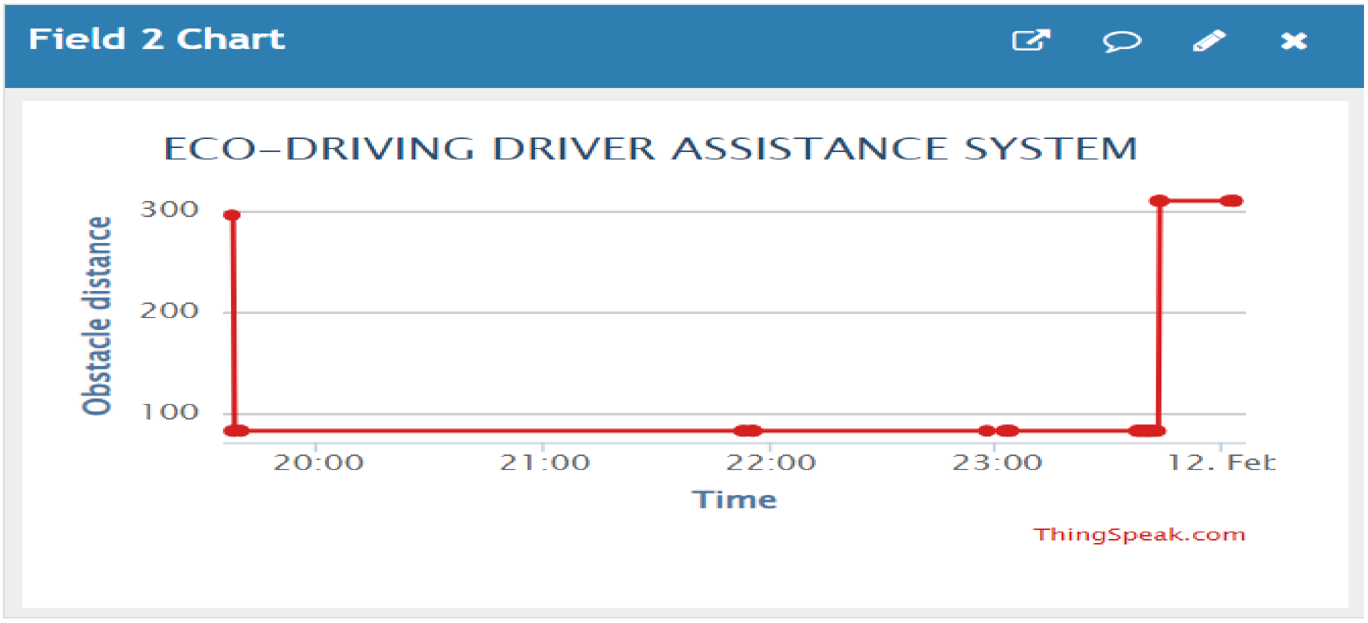


```
Engine Status:
1
RPM = 0
Attempting to connect to SSID: Wokwi-GUEST
.
Connected.
Channel update successful.
Distance (cm): 308.92
Acceleration X: 0.00, Y: 0.00, Z: 9.81 m/s^2
Rotation X: 0.00, Y: 0.00, Z: 0.00 rad/s

Engine Status:
1
Rotated counterclockwise ⏪ Pos:1
RPM = 30
Channel update successful.
Distance (cm): 308.98
Acceleration X: 0.00, Y: 0.00, Z: 9.81 m/s^2
Rotation X: 0.00, Y: 0.00, Z: 0.00 rad/s
```

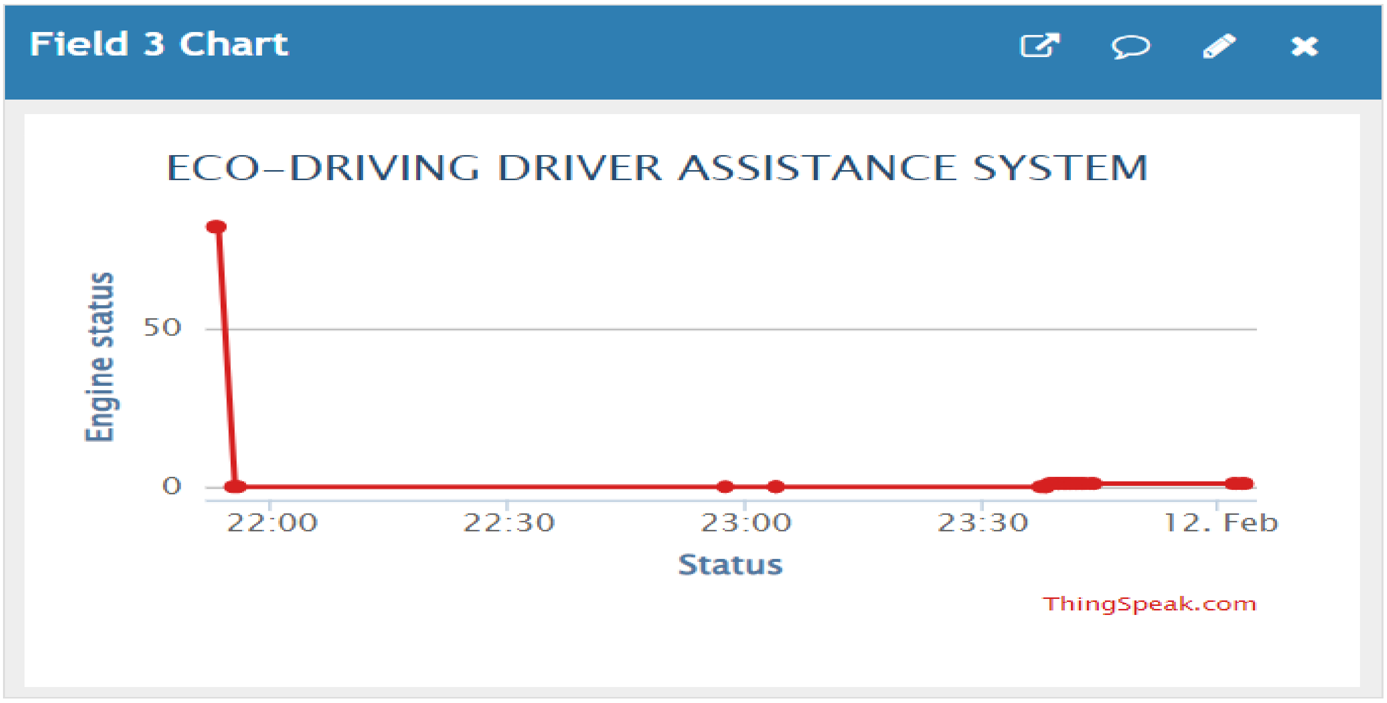
DATA VISUALISATION ON THINGSPEAK



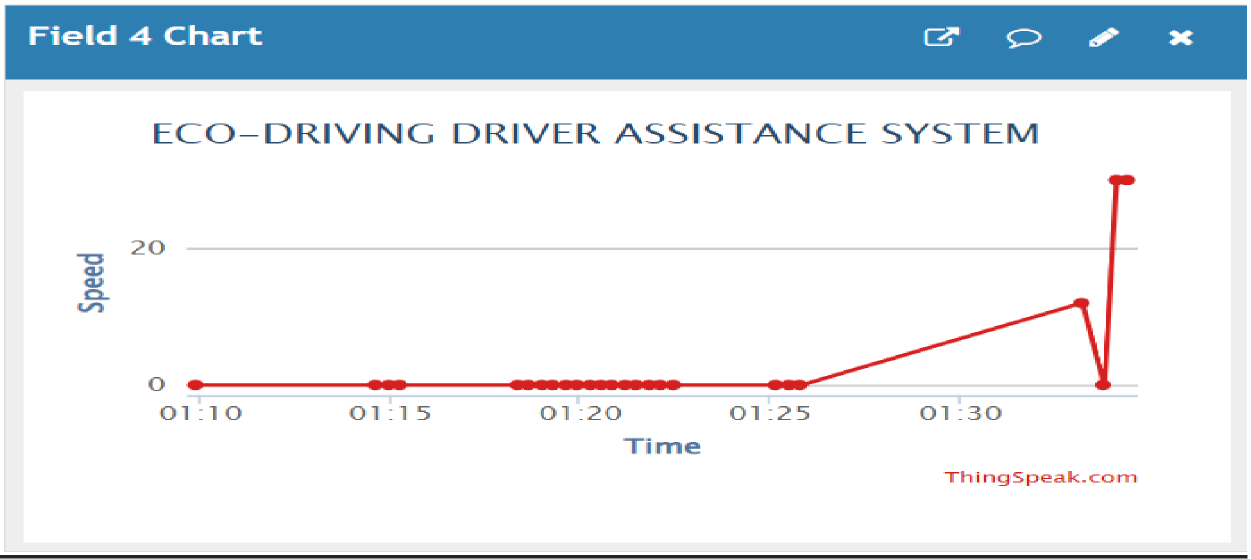
MONITORING OF ACCELERATION



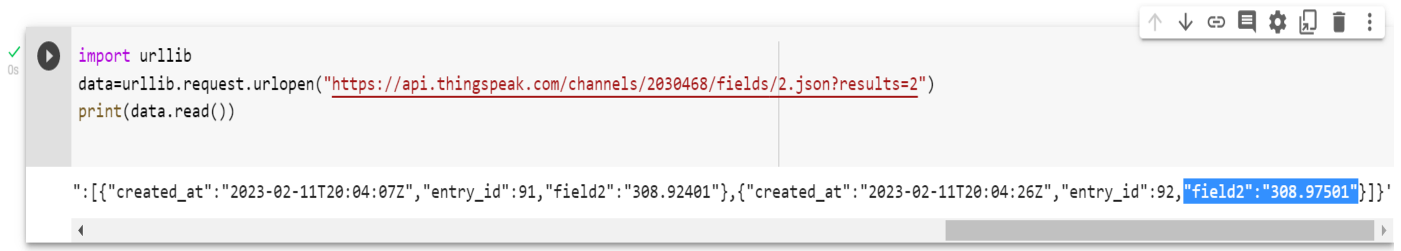
MONITORING OF OBSTACLE DISTANCE



MONITORING OF ENGINE STATUS

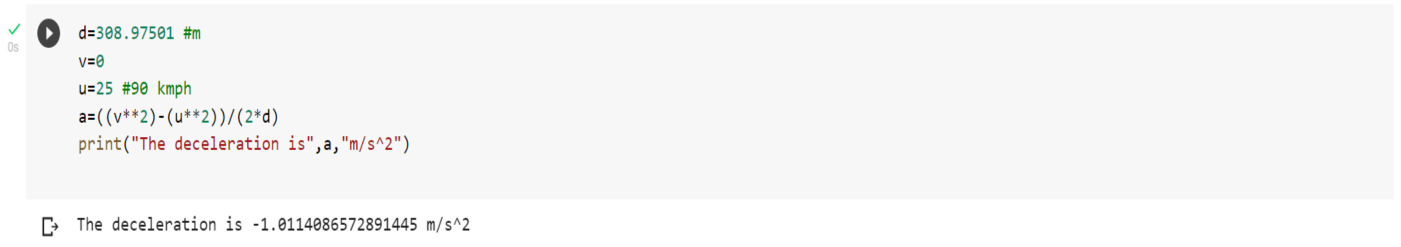


MONITORING OF SPEED



```
import urllib
data=urllib.request.urlopen("https://api.thingspeak.com/channels/2030468/fields/2.json?results=2")
print(data.read())
```

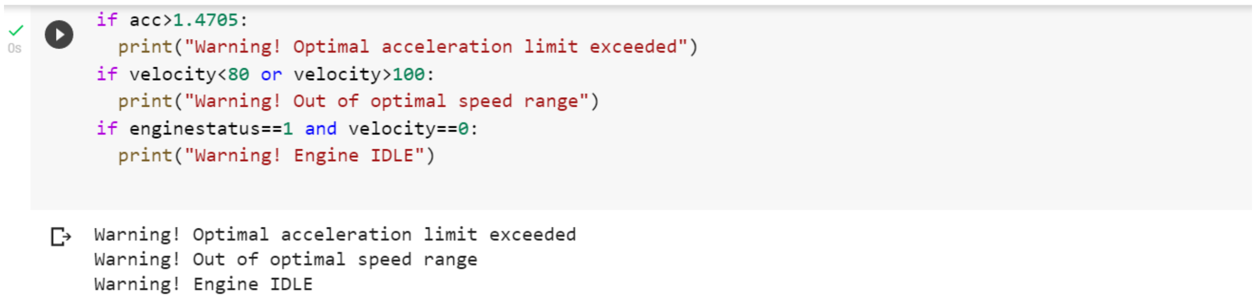
```
":[{"created_at":"2023-02-11T20:04:07Z","entry_id":91,"field2":"308.92401"},{"created_at":"2023-02-11T20:04:26Z","entry_id":92,"field2":"308.97501"}]}'
```



```
d=308.97501 #m
v=0
u=25 #90 kmph
a=((v**2)-(u**2))/(2*d)
print("The deceleration is",a,"m/s^2")
```

```
The deceleration is -1.0114086572891445 m/s^2
```

OPTIMAL DECELERATION CALCULATED



```
if acc>1.4705:
  print("Warning! Optimal acceleration limit exceeded")
if velocity<80 or velocity>100:
  print("Warning! Out of optimal speed range")
if enginestatus==1 and velocity==0:
  print("Warning! Engine IDLE")
```

```
Warning! Optimal acceleration limit exceeded
Warning! Out of optimal speed range
Warning! Engine IDLE
```

ALERTS GIVEN WHENEVER REQUIRED

PROJECT DESCRIPTION

The objective of the project is to devise a system wherein the driver can be assisted using multiple ways such that he or she adheres to the eco-driving principles. The driver is made to follow certain rules such that the total energy consumed is minimal. In order to accomplish this task, we begin with connecting the microcontroller to the required electronic components. The electronic components include an ultrasonic sensor for obstacle distance calculation, switch indicating the switching off and on of the engine, encoder for calculation of deceleration and speed and accelerometer sensor for calculating acceleration.

We have implemented a driver assistance system that continuously monitors the parameters of a vehicle and provides alerts to the user if any of the defined principles are violated.
In the system, the cruising speed is set to 90km/hr and represents the cruising speed for the vehicle. If the actual speed is more than the limit it alerts the driver to slow down.

The acceleration limit is set to 1.4705m/s^2 if the driver exceeds this limit alert is sent and the driver is intimidated to limit his/her acceleration. To prevent the idling of the engine a time limit is set for 30 sec if it is exceeded it recommends the driver switch off the engine for fuel saving. Our system detects obstacles in the proximity of 50m by using ultrasonic sensors and alerts the users for optimal deceleration.

WOKWI CODE

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>


////////////////////////////////////////
#include <WiFi.h>
#include <ThingSpeak.h>
#define SECRET_SSID "Wokwi-GUEST"
#define SECRET_PASS ""

#define SECRET_CH_ID1 2030468   // replace 0000000 with your channel number
#define SECRET_WRITE_APIKEY1 "OBR1VRWVXQ4M7UOP"   // replace XYZ with your
channel write API Key



char ssid[] = SECRET_SSID;   // your network SSID (name)
char pass[] = SECRET_PASS;   // your network password
//int keyIndex = 0;            // your network key Index number (needed only for WEP)
WiFiClient  client;
unsigned long myChannelNumber1 = SECRET_CH_ID1;
const char * myWriteAPIKey1 = SECRET_WRITE_APIKEY1;



//////////////////////////////////////

#define ENCODER_CLK 12
#define ENCODER_DT  14

int lastClk = HIGH;
int counterEnc = 0;
int pulse_cnt = 0;
```

```
unsigned long previousMillis = 0;
unsigned long previousPulseMillis = 0;
float filtered_rpm = 0.0;





//////////////////////////////////////
Adafruit_MPU6050 mpu;


const int trigPin = 5;
const int echoPin = 18;

int buttonpin=27;
int buttonRead;



//define sound speed in cm/uS
#define SOUND_SPEED 0.034

long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
////////////////////////////////////////////////

Serial.begin(115200);
  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
  case MPU6050_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case MPU6050_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case MPU6050_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case MPU6050_RANGE_16_G:
    Serial.println("+-16G");
    break;
  }
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  Serial.print("Gyro range set to: ");
  switch (mpu.getGyroRange()) {
  case MPU6050_RANGE_250_DEG:
    Serial.println("+- 250 deg/s");
    break;
  case MPU6050_RANGE_500_DEG:
    Serial.println("+- 500 deg/s");
    break;
  case MPU6050_RANGE_1000_DEG:
    Serial.println("+- 1000 deg/s");
    break;
  case MPU6050_RANGE_2000_DEG:
    Serial.println("+- 2000 deg/s");
    break;
  }

  mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
  Serial.print("Filter bandwidth set to: ");
  switch (mpu.getFilterBandwidth()) {
  case MPU6050_BAND_260_HZ:
    Serial.println("260 Hz");
    break;
  case MPU6050_BAND_184_HZ:
    Serial.println("184 Hz");
    break;
  case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
  case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
  case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
  case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
  case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
  }

  Serial.println("");
  delay(100);

//////////////////////////////////////////////////////////
pinMode(buttonpin, INPUT);





/////////////////////////////////////////////////
WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);  // Initialize ThingSpeak

/////////////////////////////////////////////////

  Serial.begin(115200);
  pinMode(ENCODER_CLK, INPUT);
  pinMode(ENCODER_DT, INPUT);
}


/////////////////////////////////////////////////
```

```
void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_SPEED/2;

  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  delay(1000);

////////////////////////////////////////////////

  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);


Serial.print("Acceleration X: ");
  Serial.print(a.acceleration.x);
  Serial.print(", Y: ");
  Serial.print(a.acceleration.y);
  Serial.print(", Z: ");
  Serial.print(a.acceleration.z);
  Serial.println(" m/s^2");

  Serial.print("Rotation X: ");
  Serial.print(g.gyro.x);
  Serial.print(", Y: ");
  Serial.print(g.gyro.y);
  Serial.print(", Z: ");
  Serial.print(g.gyro.z);
  Serial.println(" rad/s");

  Serial.println("");
  delay(500);
```

```
//////////////////////////////////////////////
buttonRead=digitalRead(buttonpin);
Serial.println("Engine Status:");
  Serial.println(!buttonRead);

//////////////////////////////////////////////

  unsigned long currentMillis = millis();

  int newClk = digitalRead(ENCODER_CLK);
  if (newClk != lastClk) {
    // There was a change on the CLK pin
    lastClk = newClk;
    int dtValue = digitalRead(ENCODER_DT);
    // Rotating clockwise causes the CLK pin to go low first, and then the DT pin goes low too.
    if (newClk == LOW && dtValue == HIGH) {
      counterEnc--;
      Serial.print("Rotated clockwise ⏩ Pos:");
      Serial.println(counterEnc);
    }
    // Rotating counterclockwise causes the DT pin to go low first, and then the CLK pin go low.
    if (newClk == LOW && dtValue == LOW) {
      counterEnc++;
      Serial.print("Rotated counterclockwise ⏪ Pos:");
      Serial.println(counterEnc);
    }
    pulse_cnt++;
  }
int RPM = pulse_cnt * 30;
  // Calculate speed by counting the pulses during a second
  if (currentMillis-previousMillis >= 1000) {
    // After counting the pulses we have to calculate the speed(revolutions per minute),
    // by knowing that KY-040 Rotary Encoder module has 20 steps per revolution.
    // We derive the equation: RPM =(counts_second / 20)* 60s = counts_second * 3
    int RPM = pulse_cnt * 30;
    previousMillis = currentMillis;
    pulse_cnt = 0;
    Serial.print("RPM = ");
    Serial.println(RPM);
  }
```

```
/////////////////////////////////////////////////
if(WiFi.status() != WL_CONNECTED)
  {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(SECRET_SSID);
    while(WiFi.status() != WL_CONNECTED)
    {
      WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this  line if using open or
WEP network
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected.");
  }


ThingSpeak.setField(1, a.acceleration.x);
ThingSpeak.setField(2, distanceCm);
ThingSpeak.setField(3, !buttonRead);
ThingSpeak.setField(4, RPM);

int x = ThingSpeak.writeFields(myChannelNumber1, myWriteAPIKey1);
if(x == 200)
  {
    Serial.println("Channel update successful.");
  }
  else
  {
    Serial.println("Problem updating channel. HTTP error code " + String(x));
  }

 delay(15000); // Wait to update the channel again


/////////////////////////////////////////////////

}
```