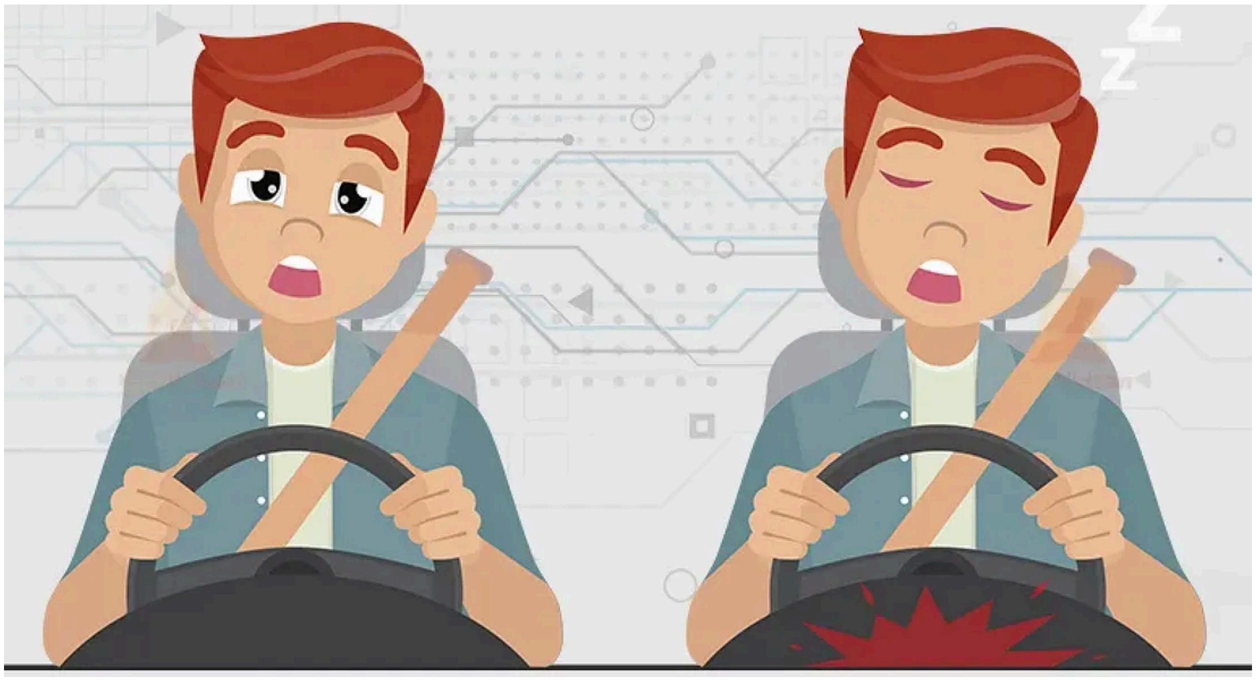


SAFE WHEEL

Electronic Circuits (EC) Lab Project

EE381 | 10th April, 2024

Dhruv (210338)
Ayush Himmatsinghka (210245)
Patil amol sanjiv (210711)
Section B, Table 1

Abstract

Safe Wheel is a drowsiness detection system that utilizes an Arduino microcontroller and sensors like a gyroscope and accelerometer to monitor a vehicle's roll and pitch angles. If these angles exceed set thresholds, indicating potential driver drowsiness, the system triggers alerts such as buzzing sounds and visual warnings on an LCD screen.

In addition, Safe Wheel includes a Bluetooth module that connects to a mobile app, providing real-time feedback on the vehicle's orientation and issuing alerts if drowsiness is detected. By integrating hardware and software components, Safe Wheel aims to enhance driver safety by proactively identifying signs of drowsiness and prompting timely intervention.Through its development and testing, Safe Wheel contributes to ongoing efforts to reduce road accidents caused by driver fatigue.

Introduction:-

In India, drowsy driving is a significant safety concern, contributing to a high number of traffic accidents and fatalities annually. Factors such as long hours of driving, insufficient rest, and monotonous road conditions exacerbate this issue, necessitating effective solutions for detecting and mitigating driver drowsiness.

The Safe Wheel project addresses this challenge by developing a drowsiness detection system tailored for Indian road conditions. This comprehensive approach aims to enhance road safety by empowering drivers to proactively address drowsiness and reduce the risk of accidents.

Existing Solutions for Drowsiness Detection:-

Vehicle-Based Systems:

Many modern vehicles are equipped with drowsiness detection systems that monitor driver behavior and vehicle movements to detect signs of fatigue.These systems typically rely on steering wheel movements, lane deviation, and vehicle speed to assess driver alertness.

Examples include Mercedes-Benz's Attention Assist and Volvo's Driver Alert Control.

Wearable Devices:

Wearable devices such as smartwatches and headsets are also used for drowsiness detection.These devices monitor physiological signals like heart rate, skin conductance, and eye movements to infer the driver's alertness level.

Products like the SmartCap and Optalert use EEG technology to measure brain activity and detect signs of drowsiness.

Camera-Based Systems:

Camera-based systems analyze facial features and eye movements to detect drowsiness in drivers.They track factors like eyelid closure, head position, and blink frequency to assess fatigue levels.

Solutions like Seeing Machines' Guardian System and Bosch's Driver Drowsiness Detection utilize this technology.

Advantages of Safe Wheel:

Real-Time Orientation Monitoring:

- Safe Wheel goes beyond traditional methods by incorporating sensors like gyroscopes and accelerometers to monitor the vehicle's roll and pitch angles in real-time.
- This allows for more accurate detection of drowsiness based on the vehicle's actual orientation, providing a proactive approach to prevent accidents.

Multi-Sensory Alert System:

- Unlike some existing solutions that rely on a single type of sensor or alert mechanism, Safe Wheel integrates multiple sensors and alert modalities.
- By combining auditory alerts (buzzing sounds) with visual warnings (LCD screen), Safe Wheel ensures that drivers receive timely and effective notifications of potential drowsiness.

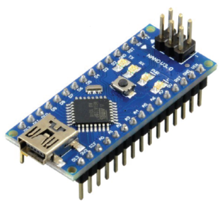
Mobile Application Integration:

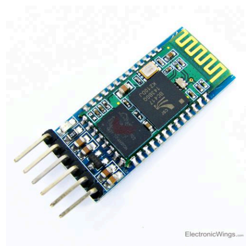
- Safe Wheel enhances driver awareness by integrating with a mobile application that provides real-time feedback on the vehicle's orientation.
- The app enables drivers to stay informed about their alertness level and receive instant alerts if drowsiness is detected, even when they are not actively monitoring the vehicle's dashboard.

Customizable Thresholds and Alerts:

- Safe Wheel allows users to customize drowsiness detection thresholds and alert settings based on their individual preferences and driving habits.
- This flexibility ensures that the system can adapt to varying levels of driver fatigue and provide personalized alerts tailored to each user's needs.

Components used :-

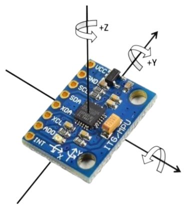

Arduino nano



Bluetooth module (HC-05)

Buzzer (DC)

LCD screen



Gyroscope accelerometer (MPU6050)

Other components: LEDs, resistors, rheostat, connecting wires, bread board

Week to week planning :-

Week 1: Project Setup and Planning

- Procure necessary components: Arduino microcontroller, sensors (gyroscope, accelerometer), LCD screen, buzzer, Bluetooth module.
- Set up a development environment for Arduino programming and mobile app development.
- Research technical specifications and requirements for Safe Wheel.
- Design circuit diagram and layout for hardware components.
- Create a detailed project plan with tasks, milestones, and deadlines.
- Assign roles and responsibilities to team members.
- Conduct kickoff meetings to discuss goals, requirements, and timelines.

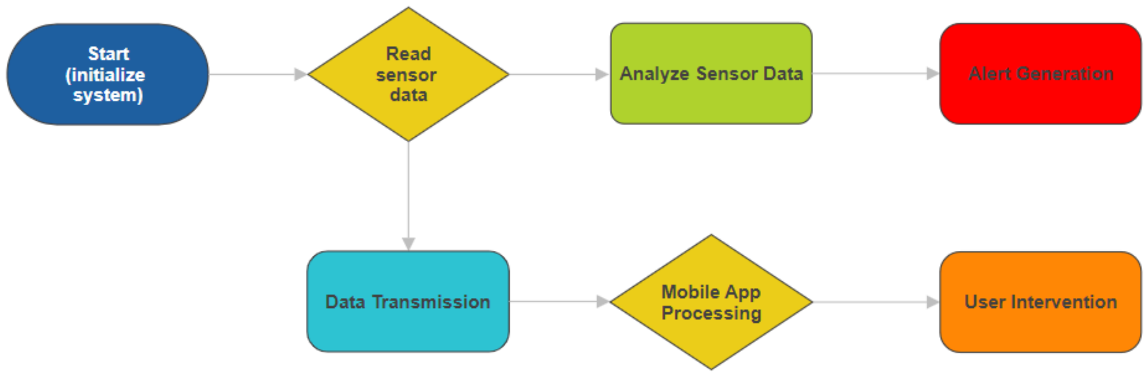
Week 2: Hardware Development and Testing

- Assemble hardware components based on circuit diagrams.
- Write and test code for sensor interfacing (gyroscope, accelerometer).
- Integrate buzzer and LCD screen for alerts.
- Test hardware setup for accuracy and reliability.
- Document hardware development process.
- Address any issues or challenges encountered.

Week 3: Software Development and Integration

- Develop mobile app interface for Safe Wheel.
- Implement Bluetooth communication between Arduino and mobile app.
- Test integration between hardware and software.
- Fine-tune drowsiness detection algorithms.
- Prepare project documentation.
- Conduct comprehensive review of project.
- Prepare for project presentation or demonstration

Flow Chart :-



Start(Initialize System):

- Power on the Arduino microcontroller.
- Initialize sensors (gyroscope, accelerometer) and peripherals.

Read Sensor Data:

- Read roll and pitch angles from the gyroscope and accelerometer.
- Analyze Sensor Data
- Compare roll and pitch angles with predefined thresholds.
- Determine if drowsiness is detected based on angle deviation.

Alert Generation:

- If drowsiness is detected:
- Trigger visual alert on LCD screen.
- Activate buzzer for auditory alert.

Data Transmission:

- Send sensor data to mobile app via Bluetooth module.

Mobile App Processing

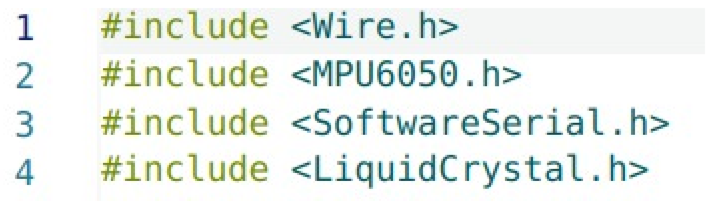
- Receive sensor data from Arduino.
- Analyze data for drowsiness detection.
- If drowsiness is detected:Display warning message on mobile app. Generate audible alerts on mobile device.

User Intervention:

- Driver acknowledges drowsiness alert.
- Take appropriate action (rest, pull over).

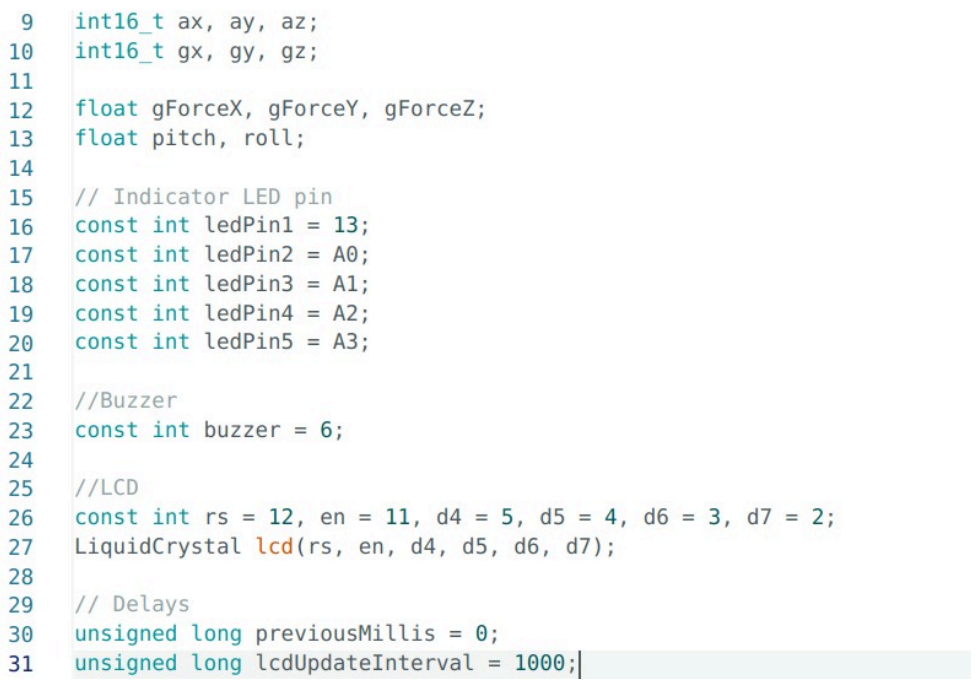
Code:-

1. Libraries



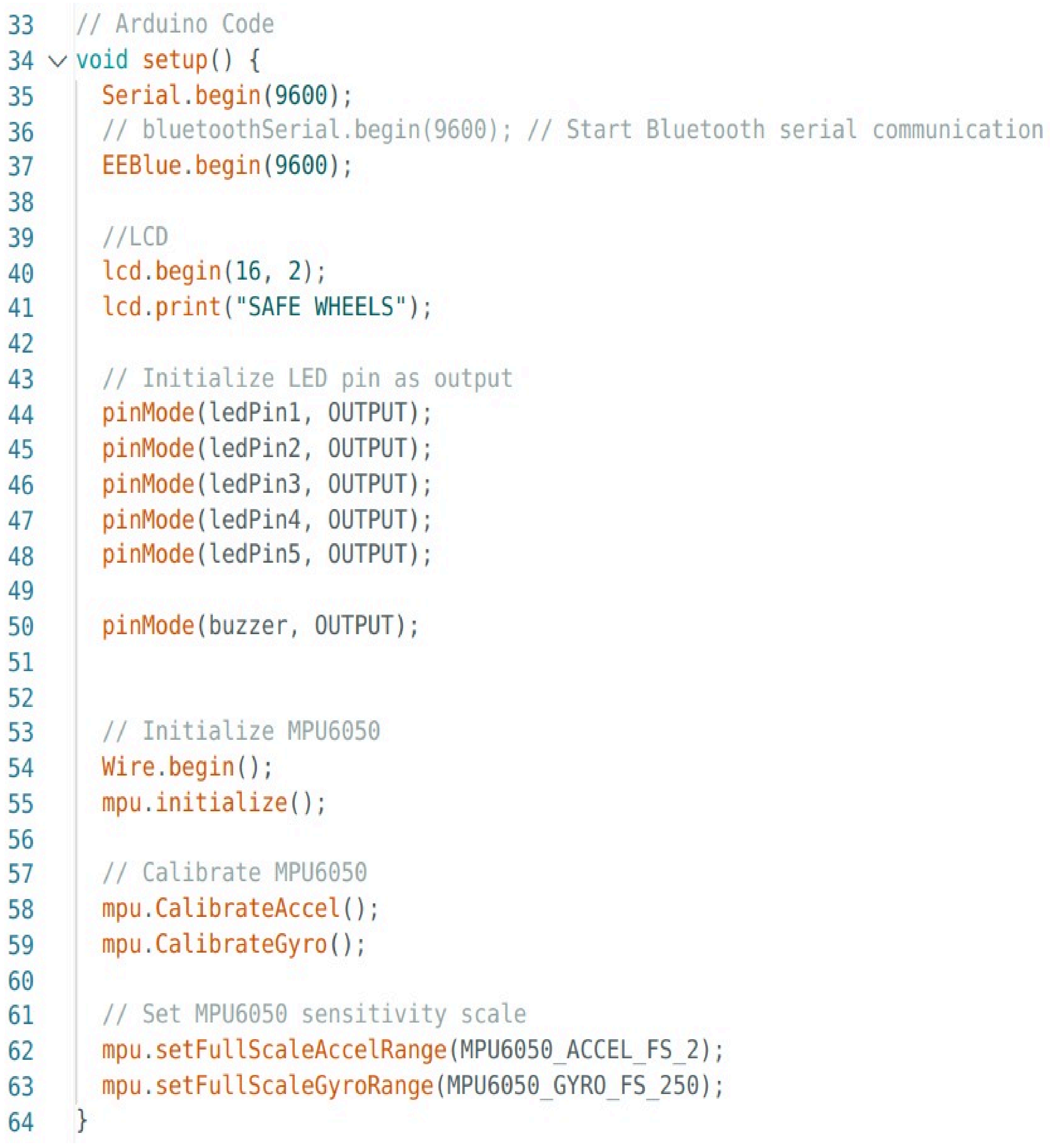
```
#include <Wire.h>
#include <MPU6050.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>
```

2.Declaration of Variables



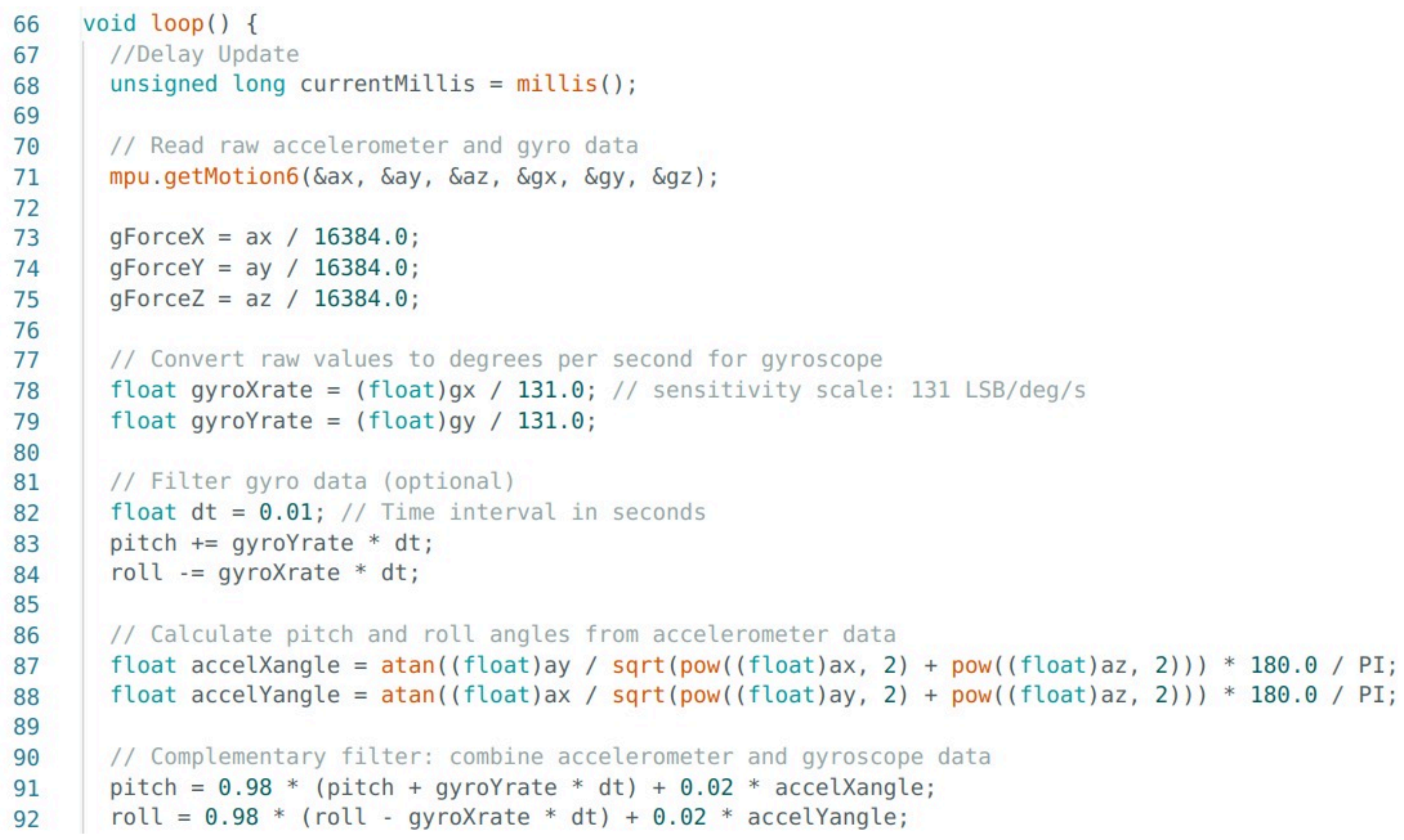
```
int16_t ax, ay, az;
int16_t gx, gy, gz;

float gForceX, gForceY, gForceZ;
float pitch, roll;

// Indicator LED pin
const int ledPin1 = 13;
const int ledPin2 = A0;
const int ledPin3 = A1;
const int ledPin4 = A2;
const int ledPin5 = A3;

//Buzzer
const int buzzer = 6;

//LCD
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// Delays
unsigned long previousMillis = 0;
unsigned long lcdUpdateInterval = 1000;
```

3.Setup



```
// Arduino Code
void setup() {
  Serial.begin(9600);
  // bluetoothSerial.begin(9600); // Start Bluetooth serial communication
  EEBlue.begin(9600);

  //LCD
  lcd.begin(16, 2);
  lcd.print("SAFE WHEELS");

  // Initialize LED pin as output
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
  pinMode(ledPin4, OUTPUT);
  pinMode(ledPin5, OUTPUT);

  pinMode(buzzer, OUTPUT);


  // Initialize MPU6050
  Wire.begin();
  mpu.initialize();

  // Calibrate MPU6050
  mpu.CalibrateAccel();
  mpu.CalibrateGyro();

  // Set MPU6050 sensitivity scale
  mpu.setFullScaleAccelRange(MPU6050_ACCEL_FS_2);
  mpu.setFullScaleGyroRange(MPU6050_GYRO_FS_250);
}
```

4.Roll and Pitch Calculation



```
void loop() {
  //Delay Update
  unsigned long currentMillis = millis();

  // Read raw accelerometer and gyro data
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  gForceX = ax / 16384.0;
  gForceY = ay / 16384.0;
  gForceZ = az / 16384.0;

  // Convert raw values to degrees per second for gyroscope
  float gyroXrate = (float)gx / 131.0; // sensitivity scale: 131 LSB/deg/s
  float gyroYrate = (float)gy / 131.0;

  // Filter gyro data (optional)
  float dt = 0.01; // Time interval in seconds
  pitch += gyroYrate * dt;
  roll -= gyroXrate * dt;

  // Calculate pitch and roll angles from accelerometer data
  float accelXangle = atan((float)ay / sqrt(pow((float)ax, 2) + pow((float)az, 2))) * 180.0 / PI;
  float accelYangle = atan((float)ax / sqrt(pow((float)ay, 2) + pow((float)az, 2))) * 180.0 / PI;

  // Complementary filter: combine accelerometer and gyroscope data
  pitch = 0.98 * (pitch + gyroYrate * dt) + 0.02 * accelXangle;
  roll = 0.98 * (roll - gyroXrate * dt) + 0.02 * accelYangle;
```

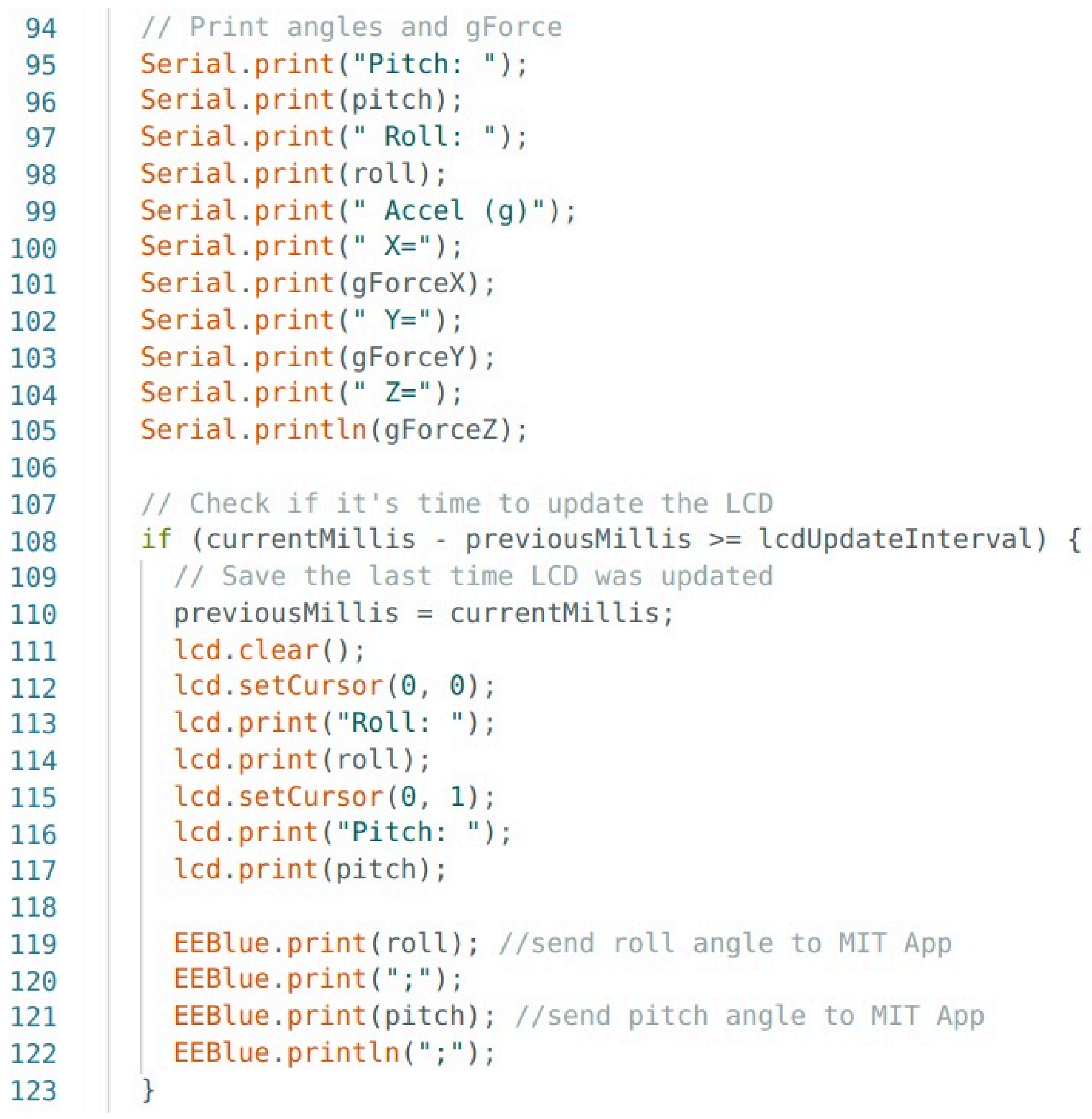
$$Pitch = \theta = arcsin\frac{a_x}{g} = arcsin\frac{a_x}{\sqrt{a_x^2 + a_y^2 + a_z^2}}$$

$$Roll = \phi = arctan\frac{a_y}{a_z}$$

$$g = 9.81\, m/s^2$$

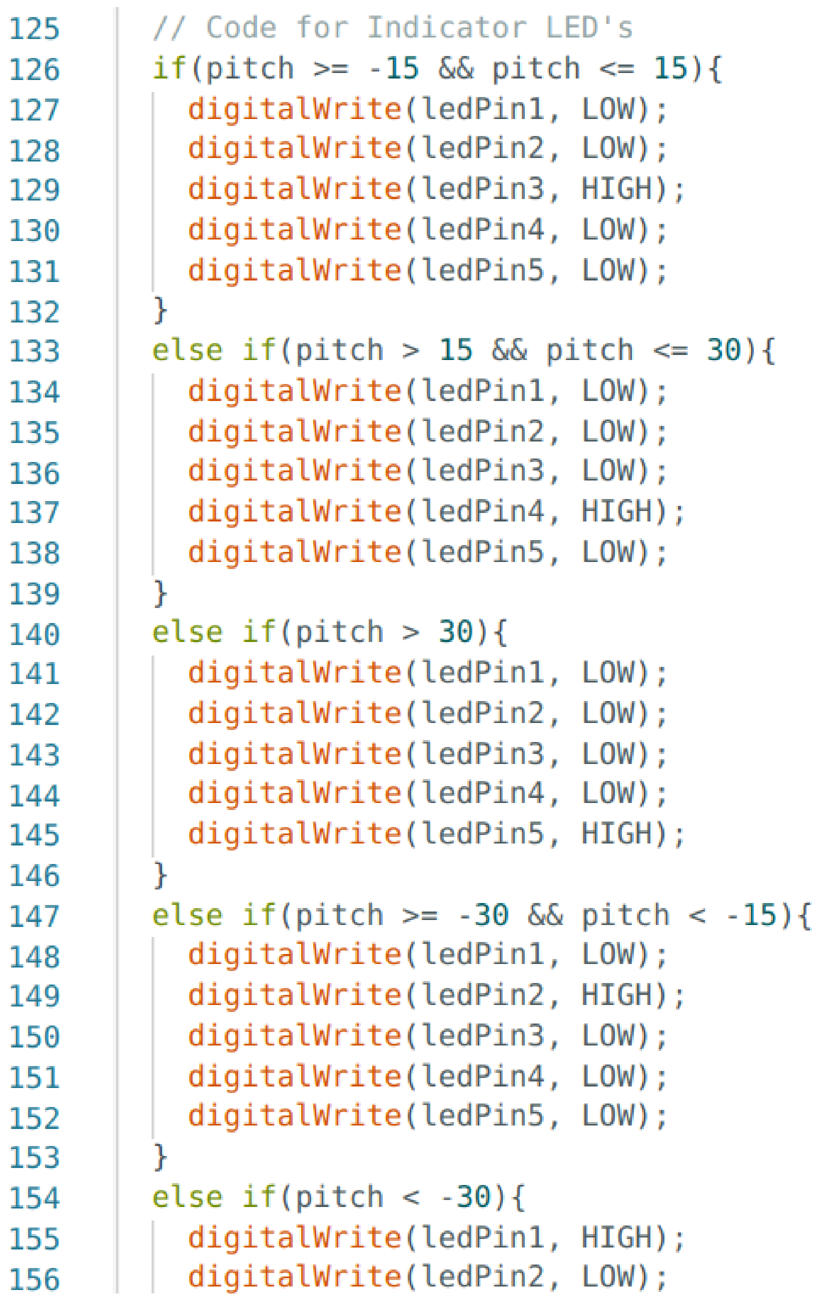
Pitch and Roll angle is calculated from above equation with raw accelerometer data. But above angles are prone to vibration errors so we had to use gyroscope data and complementary filter which take the weighted average of the two sensor readings and returns final roll and pitch angle.

5.Displaying results on serial monitor and LCD display



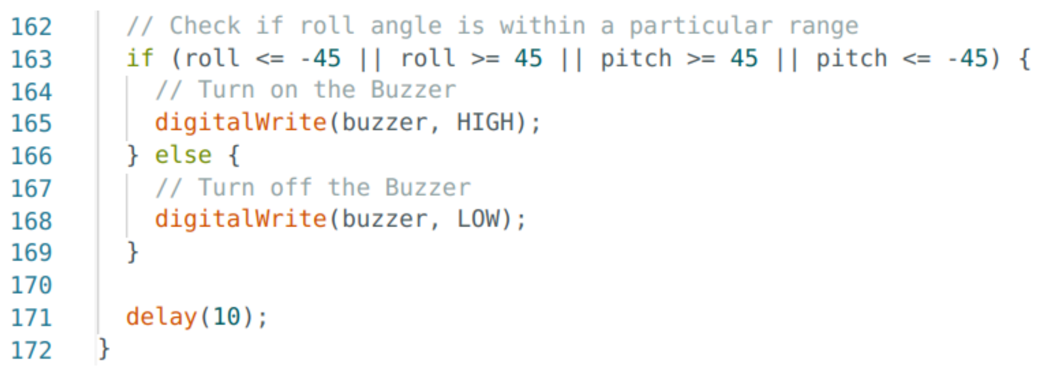
```
// Print angles and gForce
Serial.print("Pitch: ");
Serial.print(pitch);
Serial.print(" Roll: ");
Serial.print(roll);
Serial.print(" Accel (g)");
Serial.print(" X=");
Serial.print(gForceX);
Serial.print(" Y=");
Serial.print(gForceY);
Serial.print(" Z=");
Serial.println(gForceZ);

// Check if it's time to update the LCD
if (currentMillis - previousMillis >= lcdUpdateInterval) {
  // Save the last time LCD was updated
  previousMillis = currentMillis;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Roll: ");
  lcd.print(roll);
  lcd.setCursor(0, 1);
  lcd.print("Pitch: ");
  lcd.print(pitch);

  EEBlue.print(roll); //send roll angle to MIT App
  EEBlue.print(";");
  EEBlue.print(pitch); //send pitch angle to MIT App
  EEBlue.println(";");
}
```

6.Threshold for Indicator Led



```
// Code for Indicator LED's
if(pitch >= -15 && pitch <= 15){
  digitalWrite(ledPin1, LOW);
  digitalWrite(ledPin2, LOW);
  digitalWrite(ledPin3, HIGH);
  digitalWrite(ledPin4, LOW);
  digitalWrite(ledPin5, LOW);
}
else if(pitch > 15 && pitch <= 30){
  digitalWrite(ledPin1, LOW);
  digitalWrite(ledPin2, LOW);
  digitalWrite(ledPin3, LOW);
  digitalWrite(ledPin4, HIGH);
  digitalWrite(ledPin5, LOW);
}
else if(pitch > 30){
  digitalWrite(ledPin1, LOW);
  digitalWrite(ledPin2, LOW);
  digitalWrite(ledPin3, LOW);
  digitalWrite(ledPin4, LOW);
  digitalWrite(ledPin5, HIGH);
}
else if(pitch >= -30 && pitch < -15){
  digitalWrite(ledPin1, LOW);
  digitalWrite(ledPin2, HIGH);
  digitalWrite(ledPin3, LOW);
  digitalWrite(ledPin4, LOW);
  digitalWrite(ledPin5, LOW);
}
else if(pitch < -30){
  digitalWrite(ledPin1, HIGH);
  digitalWrite(ledPin2, LOW);
```

7. Threshold for Sleep Detection



```
  // Check if roll angle is within a particular range
  if (roll <= -45 || roll >= 45 || pitch >= 45 || pitch <= -45) {
    // Turn on the Buzzer
    digitalWrite(buzzer, HIGH);
  } else {
    // Turn off the Buzzer
    digitalWrite(buzzer, LOW);
  }

  delay(10);
}
```

Testing and Analysis: Safe Wheel Components

Arduino Microcontroller:

- Tested functionality of Arduino microcontroller using basic code.
- Verified compatibility with selected sensors and peripherals.
- Analyzed power consumption and processing capabilities under different loads.
- Conducted testing to assess reliability and stability over extended operation period.

Sensors (Gyroscope and Accelerometer):

- Tested accuracy and precision of sensor readings in detecting roll and pitch angles.
- Evaluated noise levels and signal filtering techniques for improved data integrity.

Alert System (Visual and Auditory):

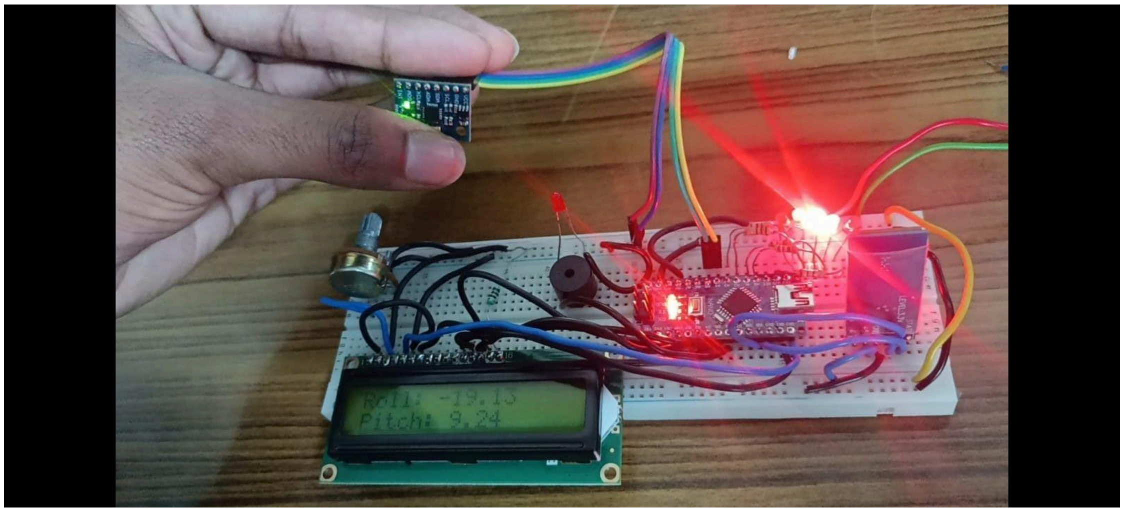
- Tested visual alert system using LCD screen for displaying warnings to the driver.
- Analyzed visibility and readability of alerts under different lighting conditions.
- Evaluated sound volume and frequency for optimal alertness without causing distraction or discomfort to the driver.

Bluetooth Connectivity and Mobile Application:

- Tested Bluetooth module for reliable connectivity with mobile devices.
- Analyzed data transmission rates and latency for real-time feedback.
- Evaluated user interface of mobile application for intuitive navigation and ease of use.

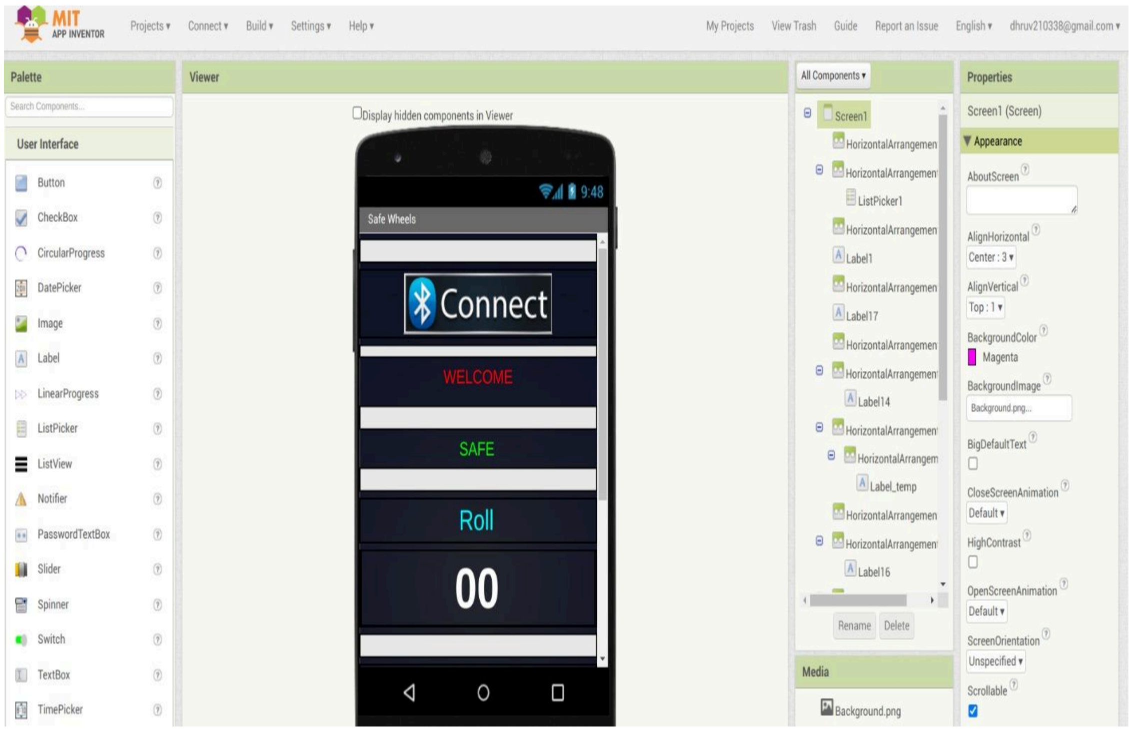
Outcome:-

Overall Circuit:



App - Designer block

.



AboutScreen

AlignHorizontal: Center : 3

AlignVertical: Top : 1

BackgroundColor: Magenta

BackgroundImage: Background.png...

BigDefaultText: ☐

CloseScreenAnimation: Default

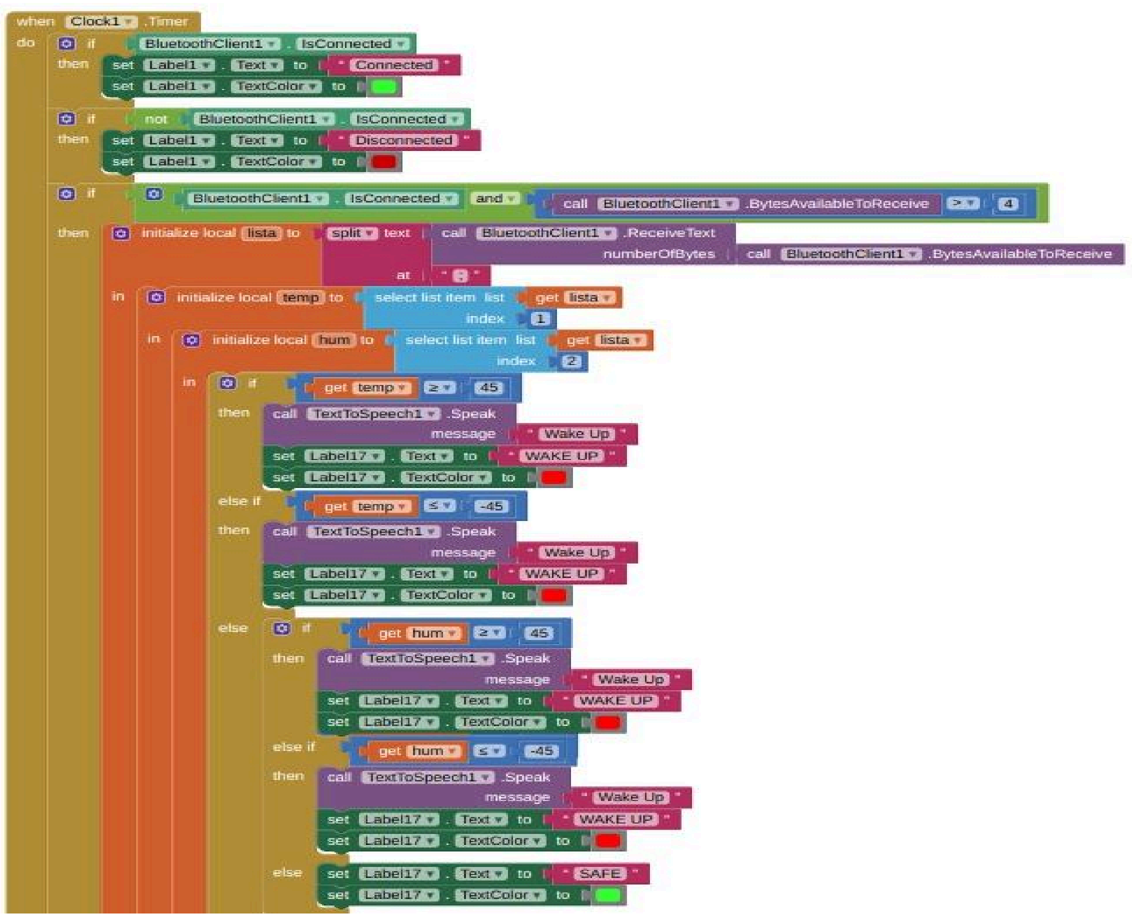
HighContrast: ☐

OpenScreenAnimation: Default

ScreenOrientation: Unspecified

Scrollable: ✓

App-Block code



```
when Clock1 .Timer
do  if BluetoothClient1 . IsConnected
    then set Label1 . Text to " Connected "
         set Label1 . TextColor to 🟩
    if not BluetoothClient1 . IsConnected
    then set Label1 . Text to " Disconnected "
         set Label1 . TextColor to 🟥
    if BluetoothClient1 . IsConnected and call BluetoothClient1 .BytesAvailableToReceive > 4
    then initialize local lista to split text call BluetoothClient1 .ReceiveText
                                                   numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
                                        at " : "
         in initialize local temp to select list item list get lista
                                                      index 1
            in initialize local hum to select list item list get lista
                                                        index 2
               in if get temp ≥ 45
                  then call TextToSpeech1 .Speak
                                          message " Wake Up "
                       set Label17 . Text to " WAKE UP "
                       set Label17 . TextColor to 🟥
                  else if get temp ≤ -45
                  then call TextToSpeech1 .Speak
                                          message " Wake Up "
                       set Label17 . Text to " WAKE UP "
                       set Label17 . TextColor to 🟥
                  else if get hum ≥ 45
                       then call TextToSpeech1 .Speak
                                               message " Wake Up "
                            set Label17 . Text to " WAKE UP "
                            set Label17 . TextColor to 🟥
                       else if get hum ≤ -45
                       then call TextToSpeech1 .Speak
                                               message " Wake Up "
                            set Label17 . Text to " WAKE UP "
                            set Label17 . TextColor to 🟥
                       else set Label17 . Text to " SAFE "
                            set Label17 . TextColor to 🟩
```

Project Video:

https://drive.google.com/drive/u/0/folders/1OQ8o3xJQLd7XG--4OhxZUNPFBqMKChrr

Conclusion :

In conclusion, the development of a drowsiness detection system utilizing the MPU6050 sensor marks a significant step towards enhancing driver safety and mitigating the risks associated with drowsy driving. By monitoring the pitch and roll angles of the driver's head in real-time, the system can promptly identify signs of drowsiness and alert the driver, thereby preventing potential accidents and safeguarding lives on the road.

While the current implementation demonstrates the feasibility of utilizing inertial sensors for drowsiness detection, there are ample opportunities for further improvement and refinement.

Future Scope :

Enhancing Driver Drowsiness Detection

While the current implementation utilizes the MPU6050 sensor to monitor pitch and roll angles of the driver's head for detecting drowsiness, there are several avenues for further improvement to enhance accuracy and reliability.

1. Sensor Fusion with Additional Sensors:

Integrating additional sensors such as a magnetometer and GPS can provide complementary data to improve the robustness of drowsiness detection. The magnetometer can offer absolute orientation information, aiding in compensating for drift in gyroscope data over time. Additionally, GPS data can provide contextual information such as vehicle speed and location, which can be correlated with driver behavior to better discern signs of drowsiness.

2. Kalman Filtering:

Implementing Kalman filtering techniques can effectively combine data from multiple sensors while accounting for noise and uncertainties, resulting in smoother and more accurate estimations of the driver's head orientation. By dynamically adjusting the weighting of sensor inputs based on their reliability, Kalman filters can mitigate errors and improve the overall performance of the drowsiness detection system.