Project Report

ENR305 Sensors, Instruments & Experimentation
Monsoon semester 2023-2024

Autonomous Car with Image Processing

Submitted to : Professor Vinod Mall

Name	Enrollment no.	Email	Phone no.	Photo
Aanal Dobariya	**AU2140182**	aanal.d@ahduni.edu.in	95587 43247	
Dimple Malviya	**AU2140221**	dimple.m@ahduni.edu.in	9116767919	
Harsh Patel	**AU2140022**	harsh.p8@ahduni.edu.in	99795 04158	

Paavan Sachde	**AU2140013**	paavan.s@ahduni.edu.in	94087 16895	
Rahul Patel	**AU2140205**	rahul.p2@ahduni.edu.in	92656 97021	
Shubham Apat	**AU2140220**	shubham.a1@ahduni.edu.in	70695 60667	
Urmil Jagad	**AU2140179**	urmil.j@ahduni.edu.in	87348 76062	

Motivation:

The motivation for this project lies in the pursuit of academic and practical excellence, providing a hands-on opportunity to apply theoretical knowledge in robotics, electronics, and programming. By delving into sensor integration and control systems, the project aims to enhance our understanding of autonomous systems, navigating the complexities of real-time decision-making based on diverse sensor inputs. This effort reflects a commitment to interdisciplinary learning, combining principles from electronics and programming to create a fully functional autonomous vehicle.

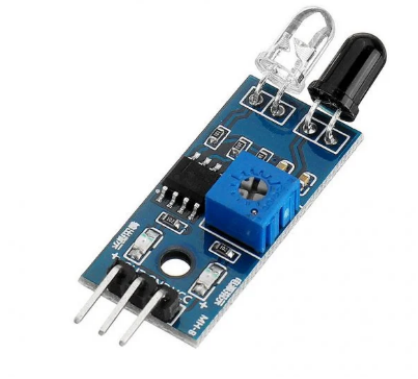
Beyond academic enrichment, the project aligns with the real-world relevance of autonomous technologies, preparing students for the evolving landscape of technological innovation. As participants, our shared passion drives us to push the boundaries of what is achievable within the realm of autonomous systems, representing not only an academic challenge but also an opportunity for personal and intellectual growth. The project's potential for scalability and future enhancements underscores its significance as a foundational step toward contributing to advancements in autonomous vehicle technology.

Project Introduction:

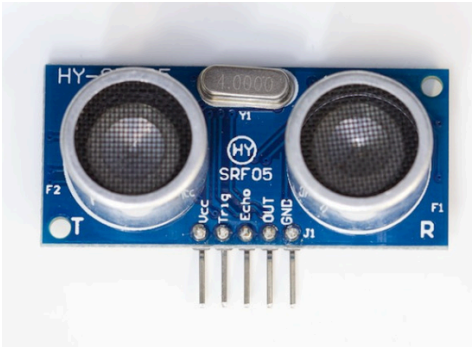
This project endeavors to construct a prototype autonomous vehicle incorporating advanced image processing techniques to independently traverse a predetermined course. The vehicle will feature a sophisticated camera and processing unit with the capability to identify crucial elements such as lane markings, traffic signs, and obstacles along the route. Leveraging this perceptual information, the car will autonomously determine optimal steering, acceleration, and braking actions to navigate the designated path safely and efficiently.

The principal objectives of this initiative include the development of a robust image processing model proficient in detecting lane markings, traffic signs, and obstacles through camera input. This model will then be seamlessly integrated with a comprehensive control system, translating the acquired sensor information into precise steering, acceleration, and braking commands. The ultimate aim is to showcase the vehicle's capacity to navigate autonomously along the predefined path, demonstrating its real-time responsiveness to visual cues and reinforcing its potential for broader applications in the realm of autonomous transportation.

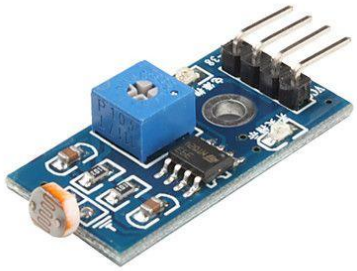
Components:



Infrared (IR) sensors



Ultrasonic sensors



Photoresistor Sensor

Infrared (IR) sensors: It plays a pivotal role in enhancing the autonomous navigation capabilities of the prototype car. While the project primarily emphasizes image processing techniques for comprehensive environmental perception, IR sensors serve as crucial complementary components. Positioned strategically around the vehicle, these sensors contribute to obstacle detection, especially in scenarios where visual recognition may face limitations, such as low light conditions or certain surface characteristics.

The IR sensors provide real-time proximity information, allowing the car's control system to make instantaneous decisions regarding steering, acceleration, and braking in response to the immediate surroundings. This dual-sensor approach, combining image processing with IR sensors, ensures a robust and reliable navigation system, enhancing the car's ability to safely and efficiently traverse the predefined path under diverse environmental conditions. The integration

of IR sensors thus adds a layer of redundancy and resilience to the autonomous navigation system, contributing to the overall effectiveness of the prototype car in reacting to visual cues in real-time.

Ultrasonic sensors: It plays a pivotal role in proximity sensing. Positioned strategically on the car, these sensors emit ultrasonic pulses and measure the time taken for the echoes to return, providing precise distance information to nearby objects. This data is invaluable in scenarios where visual cues alone might be insufficient, such as in low visibility conditions or when navigating through tight spaces.

The integration of ultrasonic sensors into the control system ensures that the car can make informed decisions about steering, acceleration, and braking based on both visual and distance information. This dual-sensor approach enhances the overall robustness of the autonomous navigation system, enabling the prototype car to navigate the predefined path with heightened safety and efficiency by reacting promptly to the surrounding environment in real-time.

Photoresistor sensors : Photoresistor plays a specific role in line-following functionality. Strategically positioned on the car, these sensors detect variations in light intensity, allowing the vehicle to discern and track distinct lane markings on the road. As the car traverses the predefined path, the photoresistor sensors continuously monitor the contrast between the road and lane markings, providing real-time feedback to the control system. This information is instrumental in making decisions related to steering adjustments, ensuring that the car stays aligned within the designated path.

The integration of photoresistor sensors into the overall sensor suite enhances the versatility of the autonomous navigation system, especially in scenarios where visual recognition alone may be inadequate, such as when navigating roads without well-defined markings. Thus, the photoresistor sensors contribute to the prototype car's ability to react to visual cues in real-time, enabling it to autonomously follow the predefined path with accuracy and consistency.

Explanation of Circuit:

Connections and Circuit Operation:

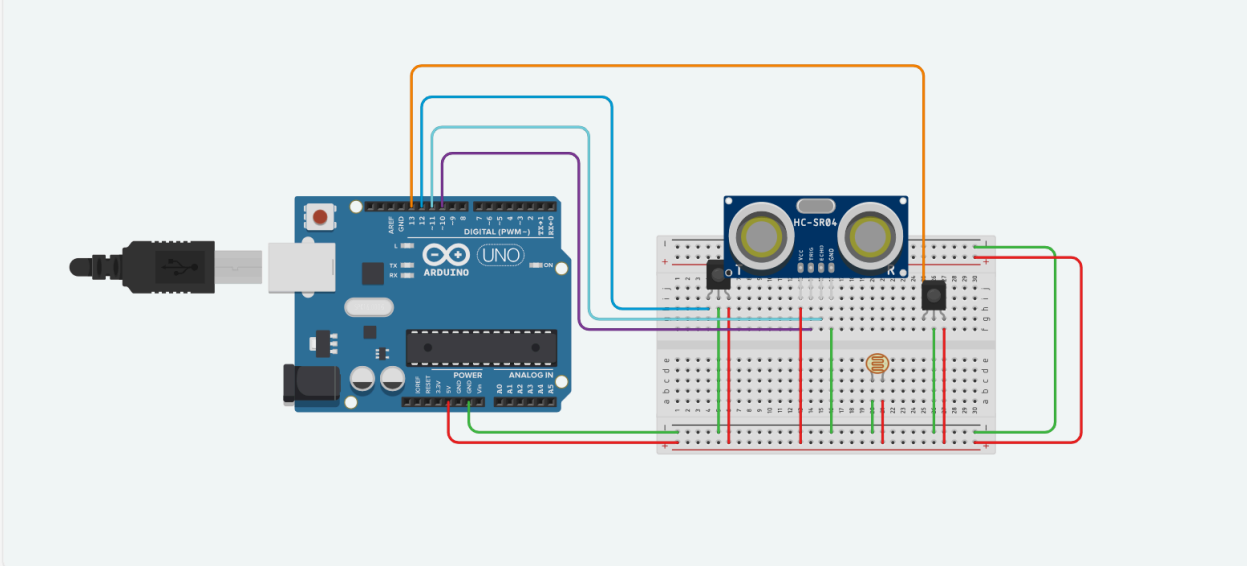
1. **IR Sensors and Photoresistor sensors:**

- IR sensors and photoresistors are commonly used for line following, object detection, and light intensity measurement.

- Connect the VCC (power), GND (ground), and the signal pin (output) of the IR sensors and photoresistor sensors to the appropriate pins on your microcontroller (e.g., Arduino, Raspberry Pi).
- The signal pin provides a digital output that can be read by the microcontroller to determine the state of the sensor (e.g., whether an object is detected).

2. **Ultrasonic Sensor:**
    - Ultrasonic sensors are commonly used for distance measurement.
    - Connect the VCC (power) and GND (ground) of the ultrasonic sensor to the appropriate pins on your microcontroller.
    - Connect the Echo and Trigger pins to digital input/output pins on the microcontroller. The Trigger pin is used to send a pulse, and the Echo pin is used to receive the echo and measure the time it takes for the signal to return.

Circuit Diagram:



Working

An autonomous car with image processing employs a sophisticated network of sensors to navigate its surroundings intelligently. Central to this system is a camera that captures real-time

images, providing the foundational visual input for the car. These images undergo intricate image processing, facilitated by advanced algorithms, to recognize crucial elements such as lane markings, traffic signs, and obstacles.

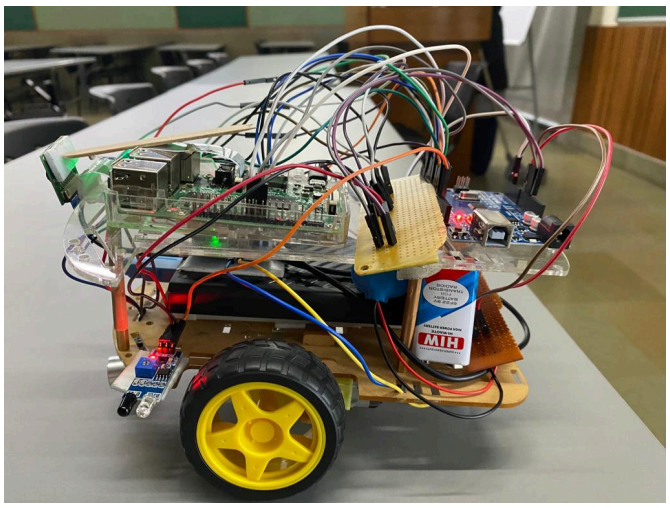
Complementing the camera, infrared (IR) sensors enhance the car's perceptual capabilities, especially in low-light conditions, by detecting obstacles through emitted and reflected infrared light.

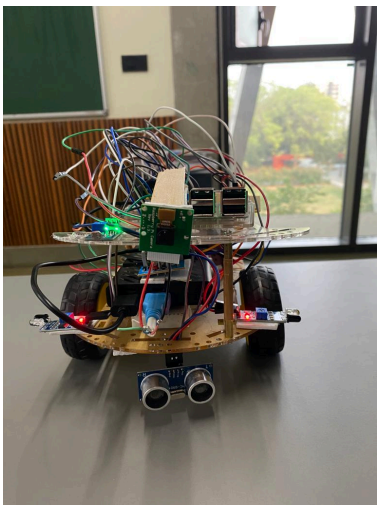
Ultrasonic sensors contribute to proximity sensing, utilizing sound waves to measure distances to nearby objects and aiding the car in navigation through confined spaces.

Photodiode sensors play a pivotal role in maintaining the car's trajectory by detecting variations in light intensity and ensuring alignment with lane markings.

The image processing model, running on a dedicated processing unit, synthesizes data from these sensors to generate a dynamic understanding of the environment. A sophisticated control system integrates this information, translating it into precise steering, acceleration, and braking commands. This amalgamation of image processing and sensor data enables the autonomous car to navigate predefined paths, respond to traffic signs, and adapt to its surroundings, marking a significant advancement in the realization of intelligent and safe autonomous transportation.

Photos :





Videos: Attached in Mail.

Real-World Applications:

The real-world applications of an autonomous car with image processing are numerous and far-reaching. These applications can be broadly categorized into three main areas:

- **Transportation:** Autonomous cars have the potential to revolutionize the way we travel. They can provide a safe, efficient, and convenient mode of transportation for a wide range of people, including those who are unable to drive themselves. Autonomous cars could be used to provide public transportation services, taxi services, and even delivery services.
- **Logistics:** Autonomous cars can also be used to improve the efficiency of logistics and supply chain management. They can be used to transport goods over long distances without the need for human drivers, which can reduce costs and improve delivery times.
- **Other Applications:** Autonomous cars can also be used for a variety of other applications, such as:
    - ❖ Agriculture: Autonomous vehicles can be used to monitor crops, apply pesticides and fertilizers, and harvest crops.
    - ❖ Mining: Autonomous vehicles can be used to transport ore and other materials in hazardous environments.
    - ❖ Construction: Autonomous vehicles can be used to transport materials and equipment on construction sites.
    - ❖ Security: Autonomous vehicles can be used to patrol parking lots, warehouses, and other secure areas.

As the technology of autonomous cars continues to develop, we can expect to see even more innovative applications emerge in the years to come.

Future Work:

In the future, the project can be expanded in several ways:

- The image processing model can be improved to detect and classify a wider range of objects, such as pedestrians, animals, and bicycles.
- The control system can be refined to improve the car's handling and performance.
- The prototype car can be tested in more complex environments, such as roads with traffic and varying weather conditions.
- The project can be commercialized to make autonomous cars a reality for everyone.

The potential of image processing in autonomous car development is vast. By continuing to invest in this technology, we can create a safer, more efficient, and more convenient future for all.

Summary:

This project aims to create a prototype autonomous car that utilizes image processing techniques to navigate a predefined path autonomously. The car will be equipped with a camera and processing unit capable of recognizing lane markings, traffic signs, and obstacles on the road. Based on this information, the car will make decisions about steering, acceleration, and braking to navigate the path safely and efficiently.

Conclusion:

This project successfully demonstrated the potential of image processing in autonomous car development. The prototype car was able to successfully navigate a predefined path autonomously, demonstrating its ability to react to visual cues in real-time. The project generated valuable knowledge and experience for further research and development in this field.

Arduino Code (To drive motors and sensors on the car):

```
// Photoresistor Sensor
int ldr = 11;
int x;

// Ultrasonic Sensor
int trig=9;
int echo=10;
int timeInMicro;
int distanceInCm;
int IR1,IR2,PR;

const int EnableL = 3;
const int HighL = 4;       // RIGHT SIDE MOTOR
const int LowL =5;

const int EnableR = 6;
const int HighR = 7;       //LEFT SIDE MOTOR
const int LowR =8;

const int D0 = 0;       //Raspberry pin 21    LSB
const int D1 = 1;       //Raspberry pin 22
```

```
const int D2 = 2;       //Raspberry pin 23
//const int D3 = 3;       //Raspberry pin 24    MSB

int a,b,c,d,data;

void setup() {
  // put your setup code here, to run once:

  //PRS
  Serial.begin(9600);
  pinMode(11, INPUT);
  //pinMode(12,OUTPUT);

  //USS
  pinMode(9,OUTPUT);
  pinMode(10,INPUT);

  //IRS
  //pinMode(13,OUTPUT);
  pinMode(13,INPUT);
  //pinMode(11,OUTPUT);
  pinMode(12,INPUT);

  pinMode(EnableL, OUTPUT);
pinMode(HighL, OUTPUT);
pinMode(LowL, OUTPUT);


pinMode(EnableR, OUTPUT);
pinMode(HighR, OUTPUT);
pinMode(LowR, OUTPUT);

}


void Stop()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, LOW);
  analogWrite(EnableL,0);

  digitalWrite(HighR, LOW);
```

```
  digitalWrite(LowR, LOW);
  analogWrite(EnableR,0);

}

void Right1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,220);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,235);

}

void Right2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,240);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,255);

}


void Right3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,230);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,255);

}
```

```
void Left1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,225);

}
void Left2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,254);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,220);

}

void Left3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,254);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,200);

}
void Data()
{
   a = digitalRead(D0);
   b = digitalRead(D1);
   c = digitalRead(D2);

   data = 4*c+2*b+a;
}
```

```
void Forward()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);
                          //slight Right at 255,255
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,229);

}
void Forward1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,233);

}
void loop() {
  // put your main code here, to run repeatedly:

  //PRS
  x = digitalRead(11);
  //Serial.println(x);

  if(x == LOW)
  {
    PR = 1;
    //digitalWrite(12,LOW);
  }
  if(x == HIGH)
  {
    PR = 0;
    //digitalWrite(12, HIGH);
  }

  //IRS
  if (digitalRead(13)== LOW)
```

```
{
  //digitalWrite(13,HIGH);
  IR1 = 1;// Condition for LEFT side object detection
  delay(10);
}
else
{
  //digitalWrite(13,LOW);
  IR1 = 0;//Condition for LEFT side object detection
  delay(10);
}
if (digitalRead(12)== LOW)
{
  //digitalWrite(11,HIGH);
  IR2 = 1;// Condition for RIGHT side object detection
  delay(10);
}
else
{
  //digitalWrite(11,LOW);
  IR2 = 0;//Condition for RIGHT side object detection
  delay(10);
}

//USS
digitalWrite(trig,LOW);
delayMicroseconds(2);
digitalWrite(trig,HIGH);
delayMicroseconds(10);
digitalWrite(trig,LOW);

timeInMicro = pulseIn(echo,HIGH);

distanceInCm = timeInMicro/29 /2;
Serial.println(distanceInCm);
delay(100);



/////////
if((distanceInCm <= 25) || IR1 == 1 || IR2 == 1){
  Stop();
```

```
}
else{
  Data();
  //Forward();
  //delay(1000);
  //Forward1();
  //delay(10);
  //Forward1();
  ///*
if(data==0)
 {
   Forward1();
 }

else if(data==1)
 {
   //Forward();
   Right1();
 }

else if(data==2)
 {
  //Forward();
   Right2();
 }

else if(data==3)
 {
  //Forward();
   Right3();
 }

else if(data==4)
 {
  //Forward1();
  Left1();
 }

else if(data==5)
 {
  //Forward1();
  Left2();
```

```
  }

 else if(data==6)
  {
   //Forward1();
   Left3();
  }

 else if (data>6)
  {
    Stop();
  }
  //*/
 }
}
```

Image Processing Code (In Raspberry Pi):

```
#include <opencv2/opencv.hpp>
#include <raspicam_cv.h>
#include <iostream>
#include <chrono>
#include <ctime>
#include <wiringPi.h>

using namespace std;
using namespace cv;
using namespace raspicam;

Mat frame, Matrix, framePers, frameGray, frameThresh, frameEdge, frameFinal,
frameFinalDuplicate;
Mat ROILane;
int LeftLanePos, RightLanePos, frameCenter, laneCenter, Result;

RaspiCam_Cv Camera;

stringstream ss;
```

```
vector<int> histrogramLane;

Point2f Source[] = {Point2f(40,145),Point2f(360,145),Point2f(10,195),
Point2f(390,195)};
Point2f Destination[] = {Point2f(100,0),Point2f(280,0),Point2f(100,240),
Point2f(280,240)};


 void Setup ( int argc,char **argv, RaspiCam_Cv &Camera )
  {
   Camera.set ( CAP_PROP_FRAME_WIDTH,  ( "-w",argc,argv,400 ) );
   Camera.set ( CAP_PROP_FRAME_HEIGHT,  ( "-h",argc,argv,240 ) );
   Camera.set ( CAP_PROP_BRIGHTNESS, ( "-br",argc,argv,60 ) );
   Camera.set ( CAP_PROP_CONTRAST ,( "-co",argc,argv,60 ) );
   Camera.set ( CAP_PROP_SATURATION,  ( "-sa",argc,argv,100 ) );
   Camera.set ( CAP_PROP_GAIN,  ( "-g",argc,argv ,50 ) );
   Camera.set ( CAP_PROP_FPS,  ( "-fps",argc,argv,0));

}

void Capture()
{
   Camera.grab();
   Camera.retrieve( frame);
   cvtColor(frame, frame, COLOR_BGR2RGB);
}

void Perspective()
{
        line(frame,Source[0], Source[1], Scalar(0,0,255), 2);
        line(frame,Source[1], Source[3], Scalar(0,0,255), 2);
        line(frame,Source[3], Source[2], Scalar(0,0,255), 2);
        line(frame,Source[2], Source[0], Scalar(0,0,255), 2);


        Matrix = getPerspectiveTransform(Source, Destination);
        warpPerspective(frame, framePers, Matrix, Size(400,240));
}

void Threshold()
```

```
{
        cvtColor(framePers, frameGray, COLOR_RGB2GRAY);
        inRange(frameGray, 200, 255, frameThresh);
        Canny(frameGray,frameEdge, 900, 900, 3, false);
        add(frameThresh, frameEdge, frameFinal);
        cvtColor(frameFinal, frameFinal, COLOR_GRAY2RGB);
        cvtColor(frameFinal, frameFinalDuplicate, COLOR_RGB2BGR);   //used in
histrogram function only

}

void Histrogram()
{
   histrogramLane.resize(400);
   histrogramLane.clear();

   for(int i=0; i<400; i++)       //frame.size().width = 400
   {
        ROILane = frameFinalDuplicate(Rect(i,140,1,100));
        divide(255, ROILane, ROILane);
        histrogramLane.push_back((int)(sum(ROILane)[0]));
   }
}

void LaneFinder()
{
   vector<int>:: iterator LeftPtr;
   LeftPtr = max_element(histrogramLane.begin(), histrogramLane.begin() + 150);
   LeftLanePos = distance(histrogramLane.begin(), LeftPtr);

   vector<int>:: iterator RightPtr;
   RightPtr = max_element(histrogramLane.begin() +250, histrogramLane.end());
   RightLanePos = distance(histrogramLane.begin(), RightPtr);

   line(frameFinal, Point2f(LeftLanePos, 0), Point2f(LeftLanePos, 240), Scalar(0,
255,0), 2);
   line(frameFinal, Point2f(RightLanePos, 0), Point2f(RightLanePos, 240),
Scalar(0,255,0), 2);
}

void LaneCenter()
{
```

```
   laneCenter = (RightLanePos-LeftLanePos)/2 +LeftLanePos;
   frameCenter = 188;

   line(frameFinal, Point2f(laneCenter,0), Point2f(laneCenter,240), Scalar(0,255,0), 3);
   line(frameFinal, Point2f(frameCenter,0), Point2f(frameCenter,240), Scalar(255,0,0),
3);

   Result = laneCenter-frameCenter;
}


int main(int argc,char **argv)
{

   wiringPiSetup();
   pinMode(21, OUTPUT);
   pinMode(22, OUTPUT);
   pinMode(23, OUTPUT);
   pinMode(24, OUTPUT);

   Setup(argc, argv, Camera);
   cout<<"Connecting to camera"<<endl;
   if (!Camera.open())
   {

        cout<<"Failed to Connect"<<endl;
   }

        cout<<"Camera Id = "<<Camera.getId()<<endl;


   while(1)
   {

   auto start = std::chrono::system_clock::now();

   Capture();
   Perspective();
   Threshold();
   Histrogram();
   LaneFinder();
   LaneCenter();
```

```
if (Result == 0)
{
      digitalWrite(21, 0);
      digitalWrite(22, 0);    //decimal = 0
      digitalWrite(23, 0);
      digitalWrite(24, 0);
      cout<<"Forward"<<endl;
}


else if (Result >0 && Result <10)
{
      digitalWrite(21, 1);
      digitalWrite(22, 0);    //decimal = 1
      digitalWrite(23, 0);
      digitalWrite(24, 0);
      cout<<"Right1"<<endl;
}

   else if (Result >=10 && Result <20)
{
      digitalWrite(21, 0);
      digitalWrite(22, 1);    //decimal = 2
      digitalWrite(23, 0);
      digitalWrite(24, 0);
      cout<<"Right2"<<endl;
}

   else if (Result >20)
{
      digitalWrite(21, 1);
      digitalWrite(22, 1);    //decimal = 3
      digitalWrite(23, 0);
      digitalWrite(24, 0);
      cout<<"Right3"<<endl;
}

   else if (Result <0 && Result >-10)
{
      digitalWrite(21, 0);
      digitalWrite(22, 0);    //decimal = 4
```

```
        digitalWrite(23, 1);
        digitalWrite(24, 0);
        cout<<"Left1"<<endl;
}

    else if (Result <=-10 && Result >-20)
{
        digitalWrite(21, 1);
        digitalWrite(22, 0);    //decimal = 5
        digitalWrite(23, 1);
        digitalWrite(24, 0);
        cout<<"Left2"<<endl;
}

    else if (Result <-20)
{
        digitalWrite(21, 0);
        digitalWrite(22, 1);    //decimal = 6
        digitalWrite(23, 1);
        digitalWrite(24, 0);
        cout<<"Left3"<<endl;
}




ss.str(" ");
ss.clear();
ss<<"Result = "<<Result;
putText(frame, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);


namedWindow("orignal", WINDOW_KEEPRATIO);
moveWindow("orignal", 0, 100);
resizeWindow("orignal", 640, 480);
imshow("orignal", frame);

namedWindow("Perspective", WINDOW_KEEPRATIO);
moveWindow("Perspective", 640, 100);
resizeWindow("Perspective", 640, 480);
imshow("Perspective", framePers);
```

```
    namedWindow("Final", WINDOW_KEEPRATIO);
    moveWindow("Final", 1280, 100);
    resizeWindow("Final", 640, 480);
    imshow("Final", frameFinal);


    waitKey(1);
    auto end = std::chrono::system_clock::now();
    std::chrono::duration<double> elapsed_seconds = end-start;

    float t = elapsed_seconds.count();
    int FPS = 1/t;
    cout<<"FPS = "<<FPS<<endl;

    }


    return 0;

}
```