School of Computer Science and Applications

"Car Obstacle Avoidance"

Minor Project report

Master of Computer Application

Submitted By:

A AYYAPPA SWAMI (R22DE004)
K SREEKANTH (R22DE070)
KALIKIRI PRANEETH (R22DE058)

Under the guidance of
Dr. Deeba K
Assistant Prof. Reva University

Department of CSA

March 2024

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru
www.reva.edu.in

Introduction:

In the realm of automotive technology, obstacle avoidance systems represent a critical advancement towards enhancing road safety. With the rise of autonomous vehicles and the increasing integration of smart sensors and artificial intelligence, the development of effective obstacle avoidance mechanisms has become imperative. These systems are designed to detect and respond to potential hazards on the road, such as other vehicles, pedestrians, or stationary objects, in real-time, thereby mitigating the risk of accidents and collisions. By leveraging a combination of sensors, algorithms, and control mechanisms, car obstacle avoidance systems aim to improve driver and passenger safety while fostering the evolution towards fully autonomous driving.

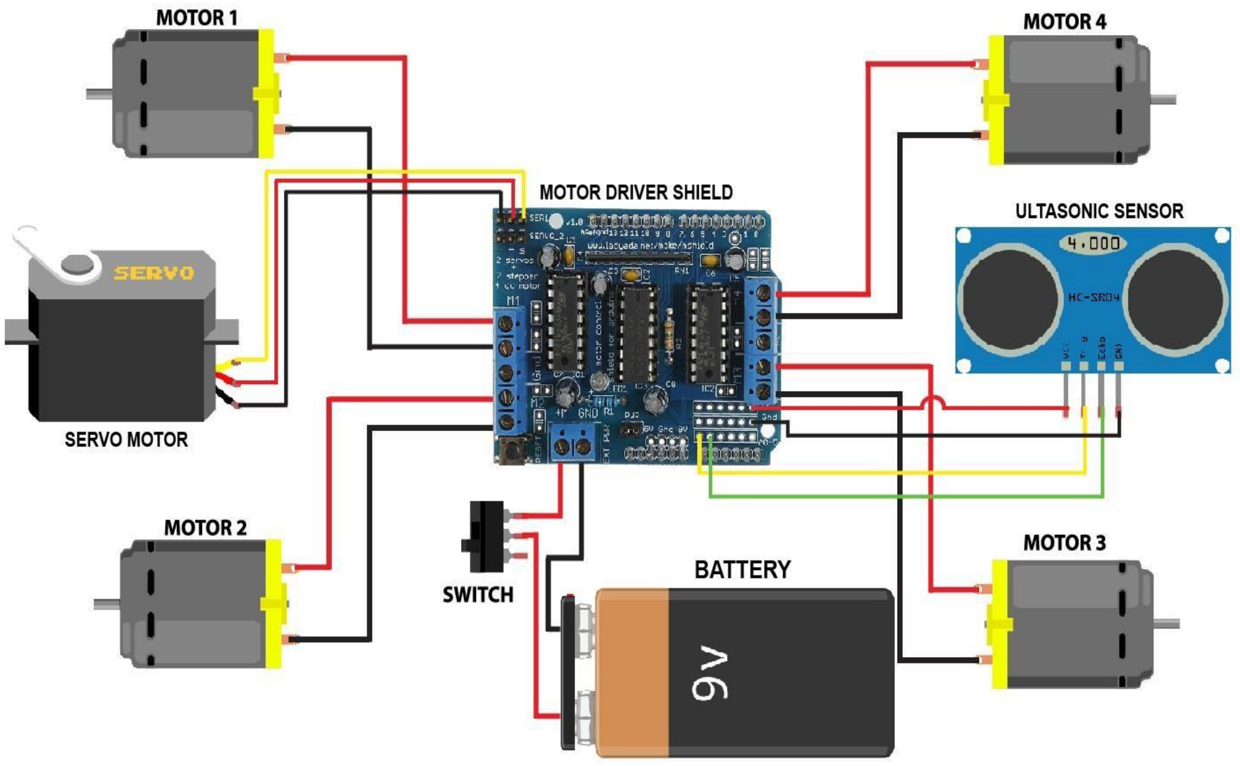
Abstract:

The car obstacle avoidance system operates through a sophisticated integration of components including an Arduino microcontroller, motor driver shield, ultrasonic sensor, and batteries, creating an intelligent and autonomous navigation solution. Mounted on the front of the vehicle, the ultrasonic sensor continuously emits pulses and measures their reflection time to gauge the distance to nearby objects, providing crucial data for obstacle detection. This information is processed in real-time by the Arduino microcontroller, which interprets the sensor data and assesses the proximity of obstacles to the vehicle. Based on this analysis, the microcontroller makes decisions regarding avoidance strategies, determining whether the vehicle should slow down, change direction, or stop to prevent a collision. The motor driver shield interfaces with the Arduino, adjusting the speed and direction of the vehicle's motion accordingly to execute the chosen avoidance man over. Power is supplied by batteries, ensuring the system's autonomy and mobility. With all components working together seamlessly, the car obstacle avoidance system dynamically navigates through complex environments, continuously monitoring its surroundings, processing sensor data, and adjusting motor control to safely avoid obstacles, thereby enhancing safety and efficiency in vehicular transportation.

Components:

1. **Arduino Uno Board:** The heart of the project, Arduino Uno provides the necessary computing power and interfaces to connect with sensors, actuators, and communication modules. Serving as the core controller, Arduino Uno facilitates data processing, sensor interfacing, and communication with external devices or networks.
2. **Motor Driver Shiels:** Motor driver shield is a compact electronic module designed to control the movement of motors in various applications, such as robotics, automation, and hobby projects. Typically used with microcontrollers like Arduino, Raspberry Pi, or other embedded systems, the motor driver shield simplifies the process of interfacing with motors by providing an integrated solution.
3. **Ultrasonic Sensor:** An ultrasonic sensor is mounted inside the smart dustbin to measure the fill level of waste. This sensor emits ultrasonic waves and calculates the distance to the nearest object, enabling accurate detection of waste volume.
4. **Micro Servo 9g:** The incorporation of a Micro Servo 9g in a Smart Dustbin IoT project adds an additional layer of functionality by enabling automated lid control. This feature enhances user convenience and hygiene while optimizing waste management processes. By integrating the Micro Servo 9g with Arduino Uno and other components, the Smart Dustbin becomes a sophisticated IoT device capable of real-time monitoring and intelligent lid operation.
5. **Power Supply:** Typically powered by a USB connection or battery, ensuring continuous operation.

Circuit Diagram :



Source Code:

```
//94% of storage used … If you run out, you can't create, edit, and
upload files. Get 100 GB of storage for ₹130.00 ₹0 for 1 month. #include
<Servo.h>
#include <NewPing.h>

#define SERVO_PIN 3
#define ULTRASONIC_SENSOR_TRIG 11
#define ULTRASONIC_SENSOR_ECHO 12
#define MAX_REGULAR_MOTOR_SPEED 75
#define MAX_MOTOR_ADJUST_SPEED 150
#define DISTANCE_TO_CHECK 30

//Right motor int
enableRightMotor=5; int
rightMotorPin1=7; int
rightMotorPin2=8;

//Left motor int
enableLeftMotor=6; int
leftMotorPin1=9; int
leftMotorPin2=10;

NewPing mySensor(ULTRASONIC_SENSOR_TRIG, ULTRASONIC_SENSOR_ECHO, 400);
Servo myServo; void setup()
{
  // put your setup code here, to run once:
  pinMode(enableRightMotor,OUTPUT);
pinMode(rightMotorPin1,OUTPUT);   pinMode(rightMotorPin2,OUTPUT);

pinMode(enableLeftMotor,OUTPUT);
pinMode(leftMotorPin1,OUTPUT);
pinMode(leftMotorPin2,OUTPUT);

myServo.attach(SERVO_PIN);
myServo.write(90);
rotateMotor(0,0);
}
void loop()
{
   int distance =
mySensor.ping_cm();
```

```
  //If distance is within 30 cm then adjust motor direction as below
if (distance > 0 && distance < DISTANCE_TO_CHECK)   {
    //Stop motors
rotateMotor(0, 0);
delay(500);

    //Reverse motors     rotateMotor(-MAX_MOTOR_ADJUST_SPEED, -
MAX_MOTOR_ADJUST_SPEED);             delay(200);

    //Stop motors
rotateMotor(0, 0);
delay(500);

    //Rotate servo to left
myServo.write(180);     delay(500);

    //Read left side distance using ultrasonic sensor
int distanceLeft = mySensor.ping_cm();
    //Rotate servo to right
myServo.write(0);         delay(500);

    //Read right side distance using ultrasonic sensor
int distanceRight = mySensor.ping_cm();
    //Bring servo to center
myServo.write(90);      delay(500);
         if (distanceLeft
== 0 )
    {       rotateMotor(MAX_MOTOR_ADJUST_SPEED, -
MAX_MOTOR_ADJUST_SPEED);       delay(200);
    }     else if (distanceRight
== 0 )
    {       rotateMotor(-MAX_MOTOR_ADJUST_SPEED,
MAX_MOTOR_ADJUST_SPEED);       delay(200);
    }
    else if (distanceLeft >= distanceRight)
    {                       rotateMotor(MAX_MOTOR_ADJUST_SPEED,    -
MAX_MOTOR_ADJUST_SPEED);       delay(200);
    }     else     {       rotateMotor(-MAX_MOTOR_ADJUST_SPEED,
MAX_MOTOR_ADJUST_SPEED);       delay(200);
    }
rotateMotor(0, 0);
delay(200);
  }   else   {     rotateMotor(MAX_REGULAR_MOTOR_SPEED,
MAX_REGULAR_MOTOR_SPEED);
  }
}
```

```
  void rotateMotor(int rightMotorSpeed, int
leftMotorSpeed)
{   if (rightMotorSpeed <
0)
  {
digitalWrite(rightMotorPin1,LOW);
digitalWrite(rightMotorPin2,HIGH);
  }   else if (rightMotorSpeed
>= 0)
  {
digitalWrite(rightMotorPin1,HIGH);
digitalWrite(rightMotorPin2,LOW);
  }    if (leftMotorSpeed
< 0)
  {
digitalWrite(leftMotorPin1,LOW);
digitalWrite(leftMotorPin2,HIGH);
  }   else if (leftMotorSpeed
>= 0)
  {
digitalWrite(leftMotorPin1,HIGH);
digitalWrite(leftMotorPin2,LOW);
  }    analogWrite(enableRightMotor,
abs(rightMotorSpeed));   analogWrite(enableLeftMotor,
abs(leftMotorSpeed));
```

Conclusion:

In conclusion, the development of a robust car obstacle avoidance system represents a significant advancement in automotive safety and autonomy. By leveraging state of-the-art sensors, algorithms, and control mechanisms, these systems can detect and respond to potential hazards on the road in real-time, thereby reducing the risk of accidents and collisions. However, the successful implementation of such systems requires careful consideration of system requirements, including sensor integration, processing power, algorithms, and reliability. Moving forward, continued research and development efforts in this field are essential to further enhance the effectiveness and reliability of obstacle avoidance systems, ultimately paving the way for safer and more efficient transportation systems.