TITLE: AUTONOMOUS VEHICLE AND ROBOTICS

Objective:

- To design and implement an intelligent autonomous vehicle system capable of navigating in real-time using computer vision, sensor data, and machine learning algorithms. The system can also be adapted for robotic platforms in various environments (indoor and outdoor).

Overview:

- Autonomous vehicles and robots rely on sensors, cameras, and algorithms to perceive their environment and make decisions. This project integrates key components such as path planning, obstacle detection, lane detection, and object recognition using Python and ROS (Robot Operating System).

Implementation:

1. Hardware Components:

- Raspberry Pi / NVIDIA Jetson Nano
- Ultrasonic Sensors / LIDAR
- Camera (e.g., Pi Camera)
- Motor Driver & Chassis
- GPS Module (optional)

2. Software Stack:

- Python
- OpenCV for Computer Vision
- TensorFlow or Py Torch for ML models
- ROS for robotics middleware (optional)
- Arduino for motor and sensor control

3. Key Functional Modules:

Lane Detection (Computer Vision):

- Use OpenCV to detect lanes in real-time video feed.

Obstacle Detection (Sensor Fusion):

- Integrate ultrasonic/LIDAR data to avoid collisions.

Object Recognition (ML/DL):

- Train a lightweight CNN to detect traffic signs or pedestrians.

Path Planning & Navigation:

- A* or Dijkstra's algorithm for planning
- PID controller for motion control

Challenges and Solutions:

CHALLENGES:

- Real-time processing
- Noisy sensor data
- Limited hardware resources
- Complex environments

SOLUTIONS:

- Use lightweight models and optimize code with multithreading or CUDA
- Apply Kalman filter or sensor fusion algorithms
- Use jetson nano or cloud-based inference if allowed
- Implement SLAM which stands for Simultaneous localization and mapping

Outcomes:

1. A working prototype of an autonomous robotic car.

1. Capable of navigating a predefined track with real-time obstacle avoidance and lane following.

1. Modular codebase for adaptation to other robotic projects.

1. Data collected for further improvement and ML model training.

LANE DETECTION:

main.py +

```python
import cv2
import numpy as np

def detect_lanes(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    lines = cv2.HoughLinesP(edges, 1, np.pi/180, 50, maxLineGap=50)
    if lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line[0]
            cv2.line(frame, (x1,y1), (x2,y2), (0,255,0), 3)
    return frame
```

OBSTACLE DETECTION:

```python
import RPi.GPIO as GPIO
import time
def get_distance(trigger_pin, echo_pin):
    GPIO.output(trigger_pin, True)
    time.sleep(0.00001)
    GPIO.output(trigger_pin, False)
    start_time = time.time()
    stop_time = time.time()
    while GPIO.input(echo_pin) == 0:
        start_time = time.time()
    while GPIO.input(echo_pin) == 1:
        stop_time = time.time()
distance = (stop_time - start_time) * 17150
    return distance
```

Progress Tracking:

Phase	Status
Hardware setup	Completed
Lane detection	In progress
Obstacle detection	Completed
Path planning	Not started
Final integration	Pending