COLLEGE CODE:3126

COLLEGE NAME: THANGAVELU ENGINEERING COLLEGE

DEPARTMENT:BE.ECE

STUDENT NM-ID: e5c503c5a11af30ad0048a864ee4d662

ROLL NO:312623106012

DATE:16/05/2025

TECHNOLOGY-PROJECT NAME:AI

SUBMITTED BY, D.RAMAKRISHNAN

Your Name and team member names.

RAMAKRISHNAN.D

VISHVA .M

SHANMUGAPRIYAN. K

RAKUL .K

SATHEESHRAJA. R

**Phase** 5: **Project Demonstration** & **Documentation**

**Title**: **Autonomous Vehicles and Robotics**

Abstract:

The project focuses on advancing autonomous vehicles and robotics by integrating real-time sensor data, control algorithms, and robotic mechanisms. It includes the use of LiDAR, cameras, radar, and IMUs to enable environment perception, navigation, obstacle avoidance, and robotic task execution. The design supports scalable applications in vehicles, drones, and industrial robots.

1. Project Demonstration

Overview:

Demonstration highlights autonomous navigation and robotic tasks, showcasing sensor data fusion, control execution, and response capabilities.

Demonstration Details:

Walkthrough: Real-time navigation or robotic task execution with path planning and manipulation.

Sensor Fusion: Live data display from LiDAR, cameras, GPS, and IMU.

Performance Metrics: Obstacle avoidance success rate, path accuracy, and response time.

Safety Measures: Emergency stops, fail-safes, and secure communication.

Outcome:

Showcase reliability and autonomous operation capabilities in dynamic environments.

2. Project Documentation

Overview:

Comprehensive documentation of hardware configuration, sensor integration, control algorithms, and operational workflows.

Documentation Sections:

Architecture: Diagrams of sensors, controllers, actuators, and communication pathways.

Code Documentation: Details of navigation, sensor processing, and robotic control software.

User Guide: Instructions for operating vehicles or robots and interpreting sensor feedback.

Maintenance Guide: Calibration, updates, and diagnostics procedures.

Testing Reports: Performance data, robustness, and adaptability assessments.

Outcome:

Clear guidance for users, developers, and maintainers.

3. Feedback and Final Adjustments

Overview:

Collect feedback from demonstrations and field tests to improve performance and usability.

Steps:

Feedback Collection: From observers and users during demos.

Refinements: Adjust control parameters, sensor calibration, and navigation algorithms.

Final Testing: Confirm improvements through real-world trials.

Outcome:

Optimized and reliable autonomous vehicle and robotics functionality.

4. Final Project Report Submission

Overview:

Summary of all development phases, results, challenges, and future recommendations.

Report Sections:

Executive Summary: Objectives, achievements, and innovations.

Phase Breakdown: Sensor integration, control strategies, and robotic capabilities.

Challenges & Solutions: Issues encountered and resolutions.

Outcomes: Readiness and performance summaries.

Outcome:

Comprehensive report for review and future work.

5. Project Handover and Future Works

Overview:

Project handover with suggestions for scaling and enhancement.

Handover Details:

Next Steps: Multi-agent coordination, advanced sensor integration, or improved controls.

Outcome:

Complete package delivered for ongoing development.

source code :


```python
import numpy as np
import matplotlib.pyplot as plt

# Define vehicle parameters
vehicle_length = 2.5  # meters
vehicle_width = 1.5  # meters
max_steering_angle = 30  # degrees

# Define control inputs
steering_angle = 10  # degrees
throttle = 0.5  # percentage

# Calculate vehicle trajectory
def calculate_trajectory(steering_angle, throttle):
    # Calculate vehicle speed
    speed = throttle * 10  # m/s

    # Calculate vehicle turn radius
    turn_radius = vehicle_length / np.tan(np.radians(steering_angle))

    # Calculate vehicle trajectory
    trajectory = np.zeros((100, 2))
    for i in range(100):
        trajectory[i, 0] = speed * i * np.cos(np.radians(steering_angle * i))
        trajectory[i, 1] = speed * i * np.sin(np.radians(steering_angle * i))

    return trajectory

# Plot vehicle trajectory
trajectory = calculate_trajectory(steering_angle, throttle)
plt.plot(trajectory[:, 0], trajectory[:, 1])
plt.xlabel('X (m)')
plt.ylabel('Y (m)')
plt.title('Vehicle Trajectory')
plt.show()
```

```python
import numpy as np
import matplotlib.pyplot as plt

# Define robot arm parameters
arm_length = 2.0  # meters
arm_angle = 45  # degrees

# Define control inputs
joint_angle = 30  # degrees

# Calculate end effector position
def calculate_end_effector_position(joint_angle):
    # Calculate end effector x position
    x = arm_length * np.cos(np.radians(joint_angle))

    # Calculate end effector y position
    y = arm_length * np.sin(np.radians(joint_angle))

    return x, y

# Plot robot arm
x, y = calculate_end_effector_position(joint_angle)
plt.plot([0, x], [0, y])
plt.xlabel('X (m)')
plt.ylabel('Y (m)')
plt.title('Robot Arm')
plt.show()
```

```python
import numpy as np
import matplotlib.pyplot as plt

# Define sensor data
sensor_data = np.random.rand(100)

# Apply filter to sensor data
def apply_filter(data):
    filtered_data = np.zeros_like(data)
    for i in range(1, len(data)):
```

7:58 88%

Untitled5.ipynb - Colab

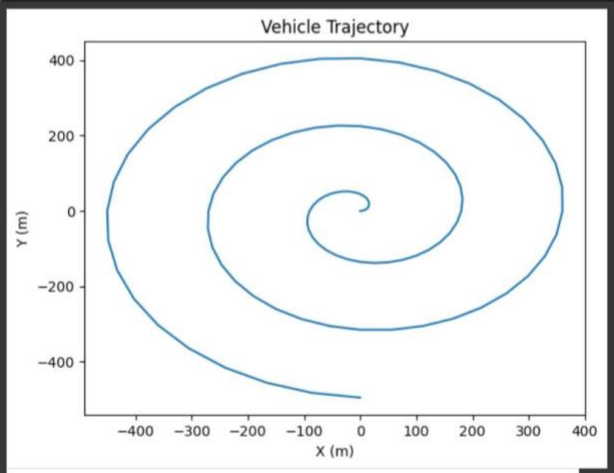
colab.research.google.com

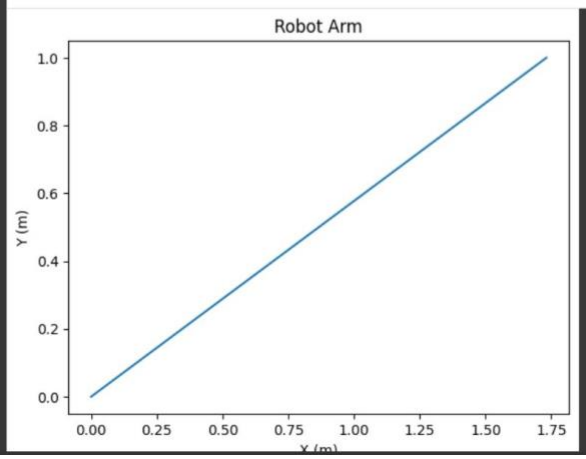
File Edit View Insert Runtime Tools Help

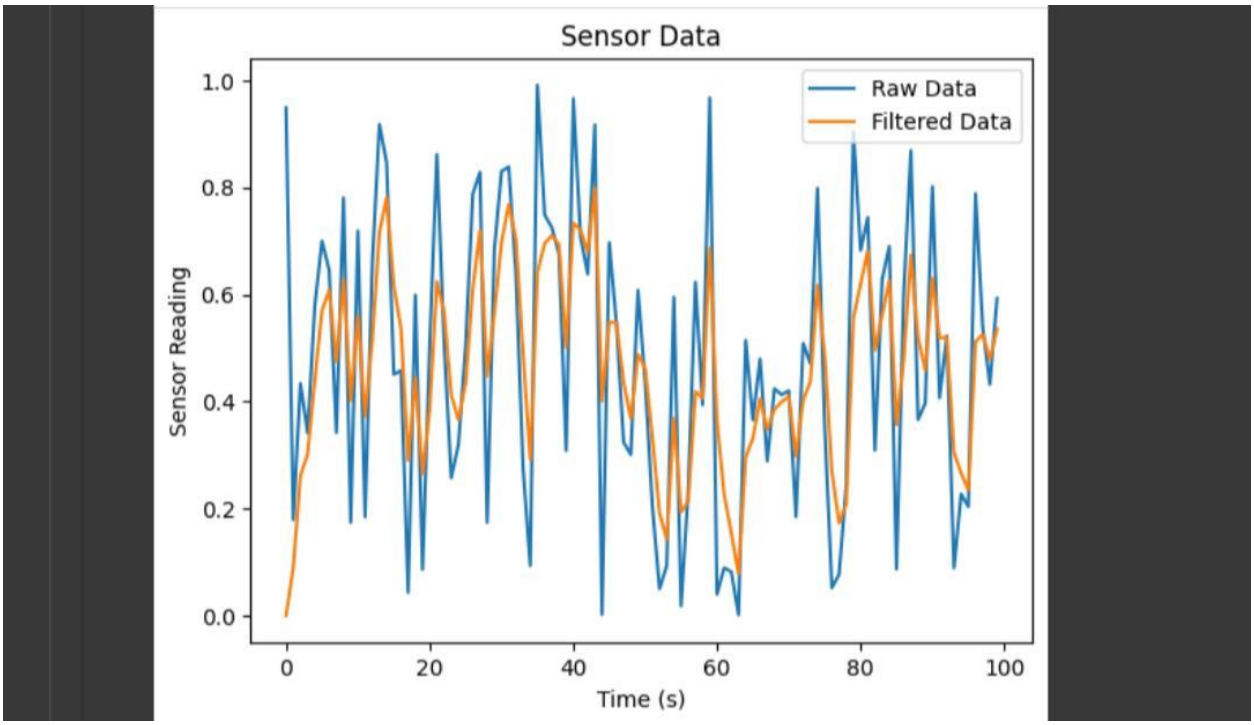
Share

Commands + Code + Text Copy to Drive

```
x, y = calculate_end_effector_position(joint_angle)
plt.plot([0, x], [0, y])
plt.xlabel('X (m)')
plt.ylabel('Y (m)')
plt.title('Robot Arm')
plt.show()



import numpy as np
import matplotlib.pyplot as plt

# Define sensor data
sensor_data = np.random.rand(100)

# Apply filter to sensor data
def apply_filter(data):
    filtered_data = np.zeros_like(data)
    for i in range(1, len(data)):
        filtered_data[i] = 0.5 * data[i] + 0.5 * filtered_data[i-1]
    return filtered_data

# Plot sensor data
filtered_data = apply_filter(sensor_data)
plt.plot(sensor_data, label='Raw Data')
plt.plot(filtered_data, label='Filtered Data')
plt.xlabel('Time (s)')
plt.ylabel('Sensor Reading')
plt.title('Sensor Data')
plt.legend()
plt.show()
```




Sensor Data
Raw Data
Filtered Data
Sensor Reading
Time (s)
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100