Name: Sai Dhaksha. S
Reg.no: 113323106083
Dept: ECE
NM.Id: aut113323ecb39

INDEX

COLLEGE CODE: 1133

COLLEGE NAME: VELAMMAL INSTITUTE 0F TECHNOLOGY

DEPARTMENT: ELECTRONICS AND COMMUNICATION ENGINEERING

STUDENT NM-ID: aut113323ecb39

ROLL NO: 113323106083

DATE: 07-05-2025

TECHNOLOGY-PROJECT NAME: AUTONOMOUS VEHICLES AND ROBOTICS

SUBMITTED BY,

Sai Dhaksha. S

Sagufta Begum

Shalabha G Anjana

Sandhiya. MV

Punugoti Deekshita

Autonomous Vehicles and Robotics

1. Project Demonstration
2. Project Documentation
3. Feedback and Final Adjustments
4. Final Project Report Submission
5. Project Handover and Future Works

1. Project Demonstration

Overview:
The Autonomous Vehicles and Robotics system will be demonstrated to stakeholders, showcasing its real-time control systems, autonomous navigation, obstacle detection, and robotic interactions.

Demonstration Details:
• System Walkthrough: A live walkthrough from sensor input to autonomous responses.
• Navigation Accuracy: Demonstration of route planning and object avoidance.
• Sensor Integration: Real-time data from LiDAR, cameras, and ultrasonic sensors.
• Performance Metrics: Latency, accuracy, and environment adaptability.
• Security: Secure communication protocols in robotic coordination.

Outcome:
Showcasing real-world adaptability, decision-making accuracy, and safe operation in varied conditions.

2. Project Documentation

Overview:
Complete technical documentation covering system architecture, autonomous algorithms, robotic modules, and control logic.

Documentation Sections:
• System Architecture: Diagrams and modular design.
• Code Documentation: Navigation, control logic, and sensor processing.
• User Guide: Operation and interface guidelines.
• Admin Guide: Maintenance and troubleshooting.
• Testing Reports: Performance, environment handling, and system robustness.

Outcome:
Enables future development, enhancements, and deployment readiness.

3. Feedback and Final Adjustments

Overview:
Collect feedback from supervisors and test observers.

Steps:
• Feedback Collection: Structured surveys and observation.
• Refinement: Fine-tuning algorithms and mechanical responses.
• Final Testing: Re-validation under controlled scenarios.

Outcome:
System optimized for deployment and real-world scenarios.

4. Final Project Report Submission

Overview:
Summary of development, integration, and testing phases.

Report Sections:
• Executive Summary
• Phase Breakdown: Sensor fusion, motion control, AI-based pathfinding.
• Challenges & Solutions: Sensor noise, mechanical constraints.
• Outcomes: Demonstrated success in semi-autonomous navigation.

Outcome:
Complete documentation supporting scalability and research continuation.

5. Project Handover and Future Works

Overview:
Preparation for continuation or deployment by future teams.

Handover Details:
• Next Steps: Advanced autonomy, multi-agent robotics, cloud integration.

Outcome:
Clear path for innovation and enhancements.

CODE:

```
import random
import time
import heapq
import hashlib

# -------------------- SENSOR SIMULATION --------------------
class Sensor:
    def read(self):
        raise NotImplementedError

class LiDARSensor(Sensor):
    def read(self):
        return random.uniform(0.5, 10.0)  # distance in meters

class UltrasonicSensor(Sensor):
    def read(self):
        return random.uniform(0.2, 5.0)

class CameraSensor(Sensor):
    def read(self):
        return random.choice(['Clear', 'Obstacle'])

# ------------------ OBSTACLE DETECTION ----------------------
def detect_obstacle(lidar_val, ultrasonic_val, camera_val):
    return lidar_val < 1.0 or ultrasonic_val < 0.5 or camera_val == 'Obstacle'

# ------------------ SIMPLE PATH PLANNING --------------------
def heuristic(a, b):
    return abs(a[0] - b[0]) + abs(a[1] - b[1])

def a_star(grid, start, goal):
    neighbors = [(0,1), (1,0), (0,-1), (-1,0)]
    close_set = set()
    came_from = {}
    gscore = {start:0}
    fscore = {start:heuristic(start, goal)}
```

```python
oheap = []

heapq.heappush(oheap, (fscore[start], start))

while oheap:
    _, current = heapq.heappop(oheap)
    if current == goal:
        data = []
        while current in came_from:
            data.append(current)
            current = came_from[current]
        return data[::-1]

    close_set.add(current)
    for i, j in neighbors:
        neighbor = current[0] + i, current[1] + j
        tentative_g_score = gscore[current] + 1
        if 0 <= neighbor[0] < len(grid):
            if 0 <= neighbor[1] < len(grid[0]):
                if grid[neighbor[0]][neighbor[1]] == 1:
                    continue
            else:
                continue
        else:
            continue

        if neighbor in close_set and tentative_g_score >= gscore.get(neighbor, 0):
            continue

        if tentative_g_score < gscore.get(neighbor, float('inf')) or neighbor not in [i[1] for i in oheap]:
            came_from[neighbor] = current
            gscore[neighbor] = tentative_g_score
            fscore[neighbor] = tentative_g_score + heuristic(neighbor, goal)
            heapq.heappush(oheap, (fscore[neighbor], neighbor))
return []
```

```python
# ------------------ AUTONOMOUS NAVIGATION --------------------
def navigate(path):
    for step in path:
        print(f"Moving to {step}")
        time.sleep(0.5)
    print("Reached Destination Successfully")

# ------------------ SECURITY PROTOCOL MOCK -------------------
def secure_communication(data, key='robotics_secure_key'):
    secure_hash = hashlib.sha256((data + key).encode()).hexdigest()
    return f"Secured Message: {secure_hash}"

# ------------------ MAIN EXECUTION --------------------------
def main():
    print("Initializing Sensors...")
    lidar = LiDARSensor()
    ultrasonic = UltrasonicSensor()
    camera = CameraSensor()

    print("Reading Sensors...")
    lidar_val = lidar.read()
    ultrasonic_val = ultrasonic.read()
    camera_val = camera.read()

    print(f"LiDAR: {lidar_val:.2f}m, Ultrasonic: {ultrasonic_val:.2f}m, Camera: {camera_val}")

    if detect_obstacle(lidar_val, ultrasonic_val, camera_val):
        print("⚠ Obstacle Detected! Recalculating Path...")
    else:
        print("✅ Path Clear. Proceeding...")

    # Grid map: 0 = free, 1 = obstacle
```

```python
    grid = [
        [0, 0, 0, 1, 0],
        [0, 1, 0, 1, 0],
        [0, 1, 0, 0, 0],
        [0, 0, 0, 1, 0],
        [1, 1, 0, 0, 0]
    ]
    start = (0, 0)
    goal = (4, 4)

    path = a_star(grid, start, goal)
    if path:
        navigate(path)
    else:
        print("❌ No viable path found!")

    print(secure_communication("Robot ready for next mission"))

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Initializing Sensors...
Reading Sensors...
LiDAR: 9.40m, Ultrasonic: 1.40m, Camera: Obstacle
⚠ Obstacle Detected! Recalculating Path...
Moving to (0, 1)
Moving to (0, 2)
Moving to (1, 2)
Moving to (2, 2)
Moving to (2, 3)
Moving to (2, 4)
Moving to (3, 4)
Moving to (4, 4)
Reached Destination Successfully
Secured Message: 7a11af0e2aae6e3b07350e075e5aea1641da6c634f3b5e7587714d
    f908d5a32f
```