NAME: SANDHIYA M.V
REG NO: 113323106086
DEPT: ECE
NM ID: aut113323eca48

Phase 3: Implementation of Project

Title: Autonomous Vehicles and Intelligent Robotics System

Objective

The goal of Phase 3 is to implement the core components of the autonomous vehicle and robotics system as planned in Phase 2. This includes perception module development, motion planning and control integration, robotic arm coordination (if applicable), and safety/security protocols.

1. Perception Module Development

Overview

A core component of any autonomous system is the perception module, which interprets sensor data to understand the environment.

Implementation

Sensor Fusion: Data from LiDAR, cameras, and radar will be fused using real-time algorithms.

Object Detection and Classification: AI models (e.g., YOLOv5 or EfficientDet) will be used for identifying vehicles, pedestrians, and obstacles.

Environmental Mapping: SLAM (Simultaneous Localization and Mapping) will be employed to create a real-time map of the surroundings.

Outcome

The system should accurately detect and classify objects and localize the vehicle within a known environment.

2. Motion Planning and Control

Overview

Motion planning is critical for determining how the vehicle or robot navigates through the environment.

Implementation

Path Planning: Algorithms such as A* or RRT will be used to generate optimal paths.

Trajectory Following: PID controllers or Model Predictive Control (MPC) will be implemented for smooth motion.

Obstacle Avoidance: Dynamic obstacle prediction and avoidance will be integrated using real-time data.

Outcome

The autonomous vehicle or robot should be capable of navigating safely, avoiding obstacles, and adjusting paths in real-time.

3. Robotics System Integration (If Applicable)

Overview

If the project includes a robotic system (e.g., delivery robot or robotic arm), Phase 3 will establish basic functionalities.

Implementation

Actuation and Motion Control: Use of servos, motors, and inverse kinematics for robotic arms.

Task Execution: Basic tasks such as picking, placing, or following set routes.

Sensor Feedback: Tactile or visual sensors for improving task precision.

Outcome

The robotic system should complete predefined tasks reliably under normal conditions.

4. Safety and Security Measures

Overview

Ensuring system safety and data security is vital during the early phases of deployment.

Implementation

Fail-safe Mechanisms: Emergency stop protocols and manual override systems.

Data Security: Basic encryption and secure communication between vehicle modules and cloud interfaces.

Regulatory Compliance: Adherence to automotive safety standards like ISO 26262.

Outcome

The system should handle failures gracefully and protect both users and data.

5. Testing and Feedback Collection

Overview

Real-world or simulated testing is crucial for refining the system and ensuring usability and safety.

Implementation

Simulation Environments: Testing in platforms like CARLA or Gazebo for autonomous vehicles.

Pilot Deployments: Limited real-world trials in controlled environments.

Feedback Mechanisms: Collect feedback from engineers, testers, and end-users.

Outcome

This phase should result in a validated perception and control stack, along with improvements based on early testing feedback.

Challenges and Solutions

1. Sensor Noise and Errors

Challenge: Inaccurate data from sensors in varying conditions.

Solution: Implement sensor fusion and redundancy to mitigate errors.

2. Real-time Processing Constraints

Challenge: Delays in decision-making.

Solution: Optimize processing pipeline using edge computing and lightweight models.

3. Safety Verification

Challenge: Proving reliability under edge cases.

Solution: Extensive simulation testing and use of safety standards.

Outcomes of Phase 3

Working perception module (sensor fusion, object detection, SLAM).

Integrated motion planning and control system.

Optional robotic arm or system executing basic tasks.

Basic safety mechanisms and secure communications.

Simulation and/or real-world feedback to inform Phase 4.

Next Steps for Phase 4

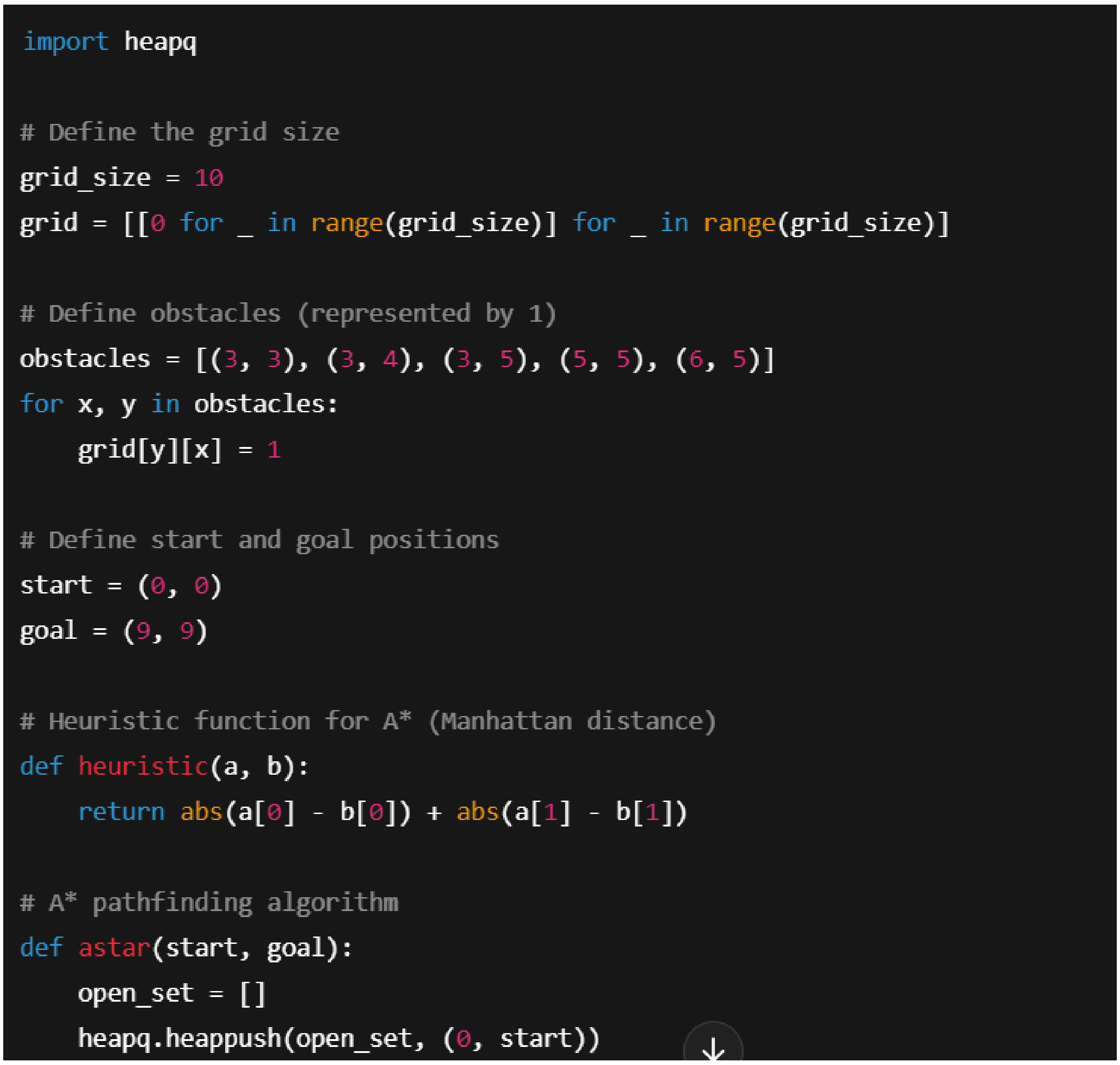
Advanced behavior prediction and collaborative vehicle/robot communication.

Integration of V2X (Vehicle-to-Everything) capabilities.

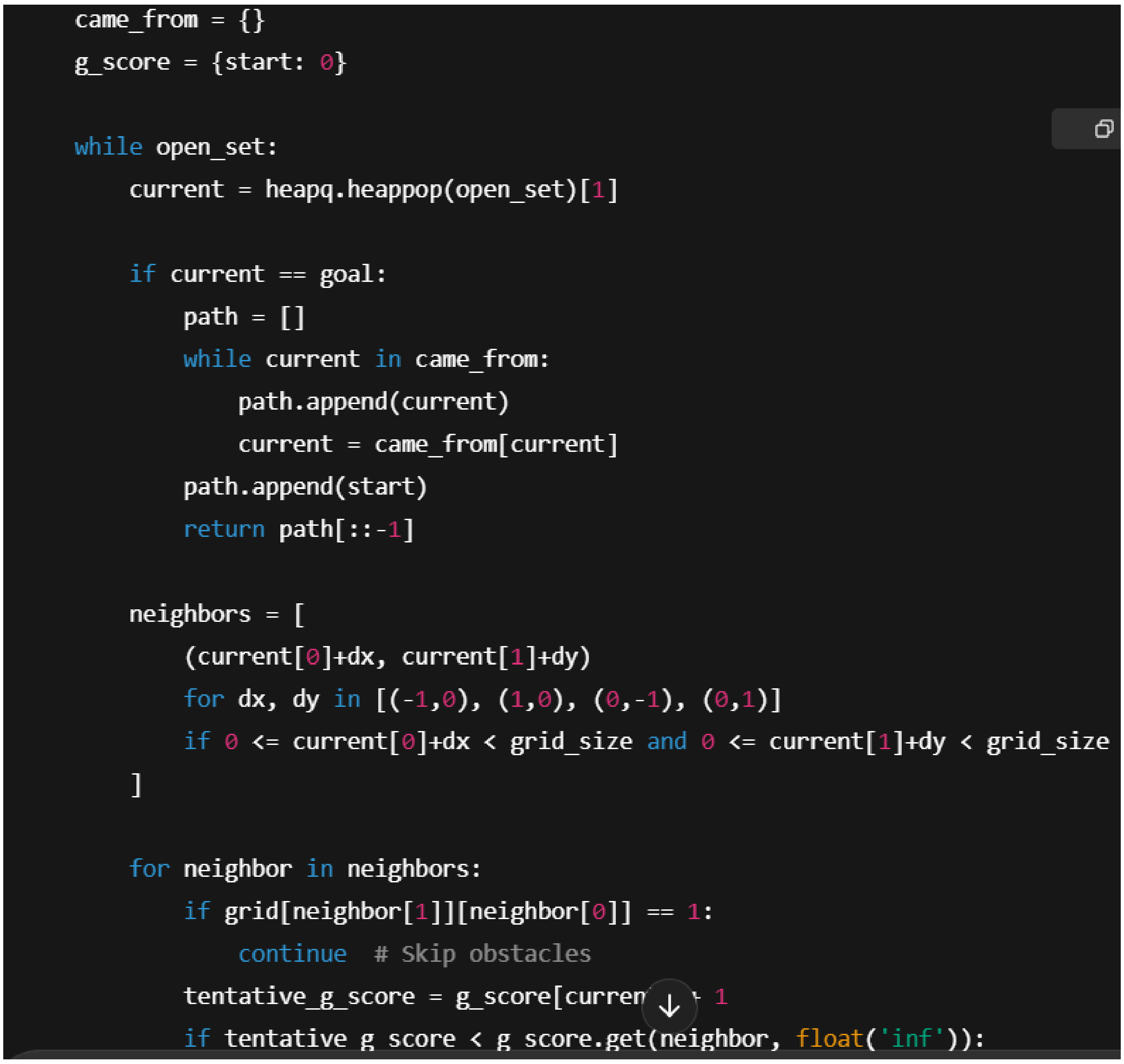
Expanded real-world testing and route scaling.

AI model refinement and hardware optimization.

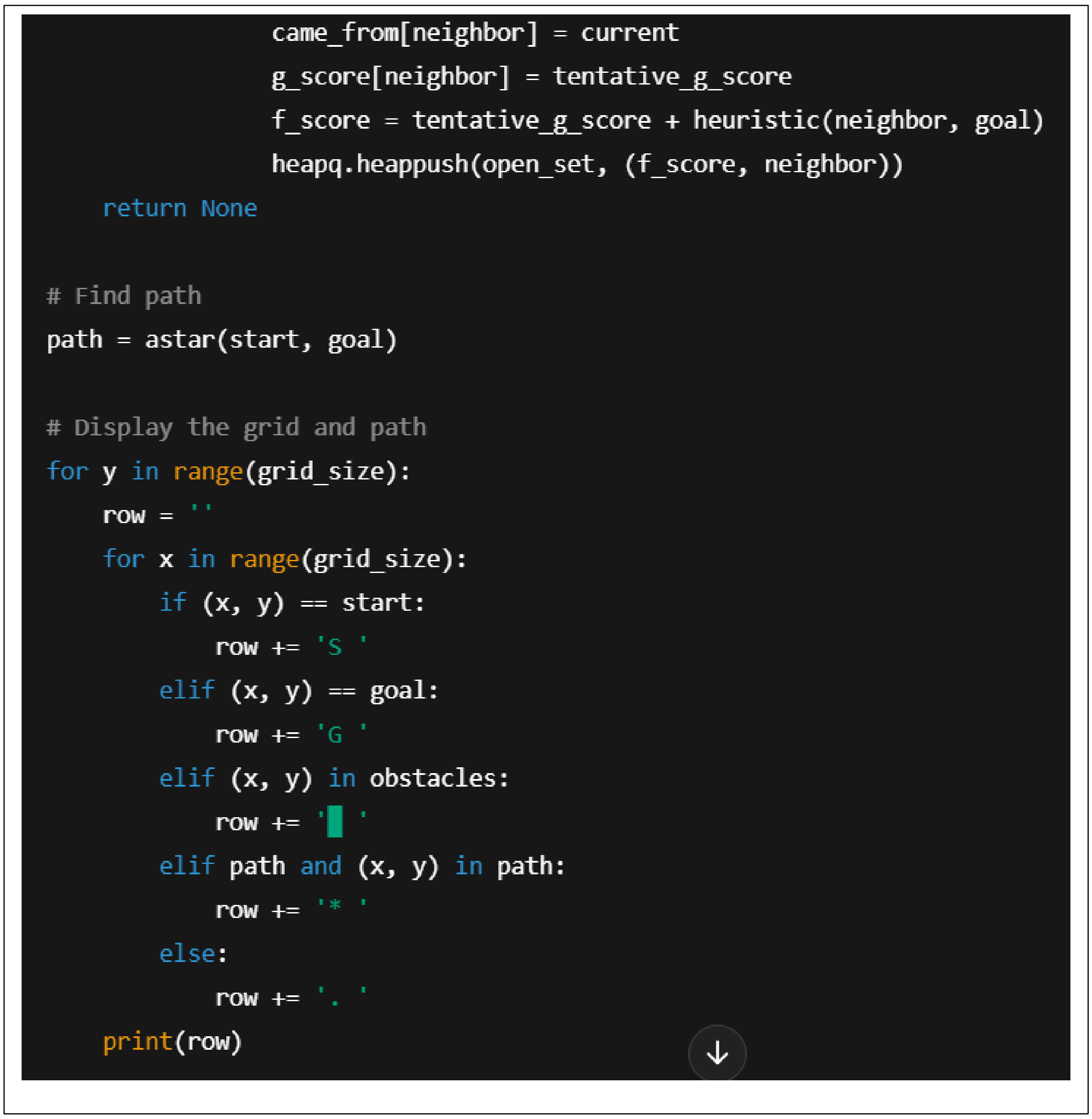
CODE:



```python
import heapq

# Define the grid size
grid_size = 10
grid = [[0 for _ in range(grid_size)] for _ in range(grid_size)]

# Define obstacles (represented by 1)
obstacles = [(3, 3), (3, 4), (3, 5), (5, 5), (6, 5)]
for x, y in obstacles:
    grid[y][x] = 1

# Define start and goal positions
start = (0, 0)
goal = (9, 9)

# Heuristic function for A* (Manhattan distance)
def heuristic(a, b):
    return abs(a[0] - b[0]) + abs(a[1] - b[1])

# A* pathfinding algorithm
def astar(start, goal):
    open_set = []
    heapq.heappush(open_set, (0, start))
```

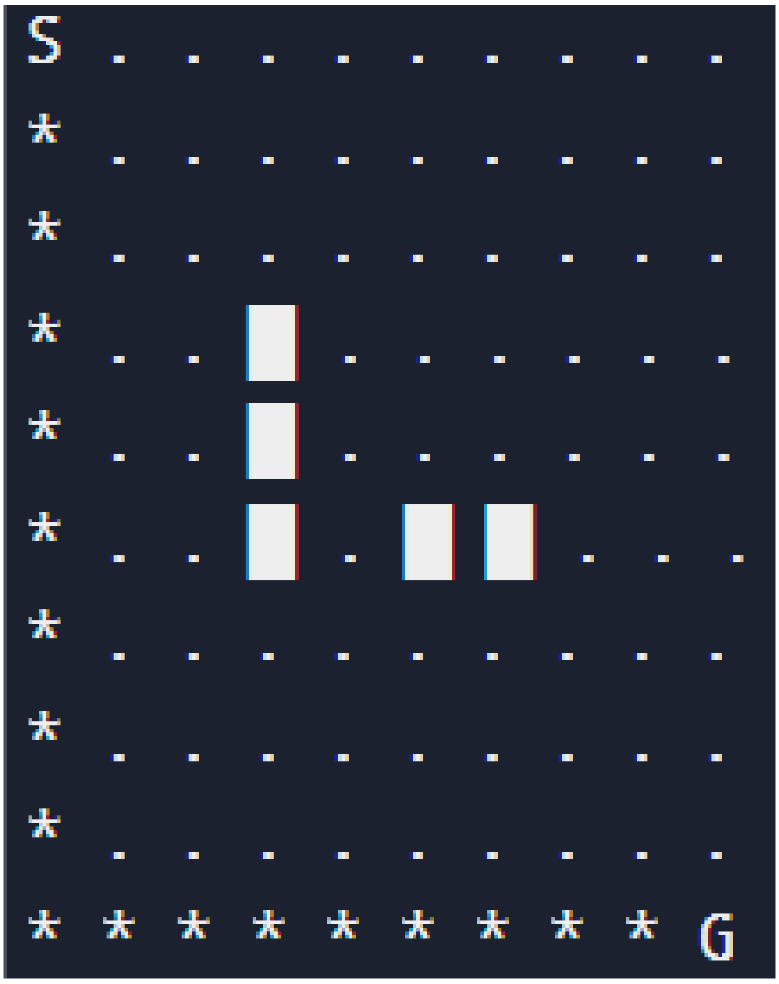


```
    came_from = {}
    g_score = {start: 0}

    while open_set:
        current = heapq.heappop(open_set)[1]

        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start)
            return path[::-1]

        neighbors = [
            (current[0]+dx, current[1]+dy)
            for dx, dy in [(-1,0), (1,0), (0,-1), (0,1)]
            if 0 <= current[0]+dx < grid_size and 0 <= current[1]+dy < grid_size
        ]

        for neighbor in neighbors:
            if grid[neighbor[1]][neighbor[0]] == 1:
                continue  # Skip obstacles
            tentative_g_score = g_score[curren[illegible]+ 1
            if tentative_g_score < g_score.get(neighbor, float('inf')):
```



```python
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score = tentative_g_score + heuristic(neighbor, goal)
                heapq.heappush(open_set, (f_score, neighbor))
    return None

# Find path
path = astar(start, goal)

# Display the grid and path
for y in range(grid_size):
    row = ''
    for x in range(grid_size):
        if (x, y) == start:
            row += 'S '
        elif (x, y) == goal:
            row += 'G '
        elif (x, y) in obstacles:
            row += '█ '
        elif path and (x, y) in path:
            row += '* '
        else:
            row += '. '
    print(row)
```

OUTPUT:



```
S . . . . . . . . .
* . . . . . . . . .
* . . . . . . . . .
* . . █ . . . . . .
* . . █ . . . . . .
* . . █ . █ █ . . .
* . . . . . . . . .
* . . . . . . . . .
* . . . . . . . . .
* * * * * * * * * G
```