Artificial Intelligence Coursework 1

by Hesham Alshawabka
Last Modified: 18/02/25

This report documents the development of the Smart Delivery Robot project, an autonomous delivery system operating within an urban grid environment. The aim is to progressively develop a robot that not only navigates and delivers parcels but also optimizes its route using advanced pathfinding techniques. The project is divided into several tasks that progress from the basic setup of the grid—with validated user input and dynamic visual updates—to a fully functional autonomous agent that makes real-time decisions based on proximity using Manhattan distance. Detailed inline comments in the code provide insight into each function's purpose, ensuring clarity and ease of understanding throughout the development process.

The final implementations are represented by Task 3, which showcases the regular smart robot autonomously delivering parcels, and Advanced Task 2, where the robot employs an A* search algorithm to determine the fastest path while considering environmental constraints. The full source code is available on GitHub at https://github.com/alshawah/Smart-Delivery-Robot, and additional planning diagrams along with a video demonstration are provided as supporting material.

Task 1: Define the Grid Size

In Task 1, the program sets the foundation for the Smart Delivery Robot by creating a grid environment based on user input. The user is prompted to enter a grid size (N x N) where N must be a number between 1 and 6, and the program validates this input to ensure it falls within the allowed range. Once a valid grid size is provided, the code randomly generates a set of delivery points and displays the grid with each cell labeled either as "Clear" or "Delivery." This setup not only prepares the environment for further tasks but also ensures that the user's input is correctly handled from the outset.

- The grid size is stored in the variable N and must be between 1 and 6.
- Input validation is implemented to prevent invalid entries.
- A random set of unique delivery points is generated and displayed on the grid, marking the cells where parcels will be delivered.

This task establishes the basic operational space for the robot, ensuring a robust, user-defined starting point for the rest of the project.

Example output:

```
Enter the grid size (N x N, where N is between 1 and 6):  5

Generated Environment:
(1,1) Clear | (1,2) Clear | (1,3) Clear | (1,4) Clear | (1,5) Clear
(2,1) Clear | (2,2) Clear | (2,3) Clear | (2,4) Clear | (2,5) Clear
(3,1) Clear | (3,2) Delivery | (3,3) Clear | (3,4) Delivery | (3,5) Clear
(4,1) Clear | (4,2) Clear | (4,3) Clear | (4,4) Clear | (4,5) Delivery
(5,1) Clear | (5,2) Clear | (5,3) Clear | (5,4) Clear | (5,5) Clear
```

Task 2: Generate Delivery Points

In Task 2, we build on the environment created in Task 1 by generating unique delivery points and initializing the Smart Delivery Robot. The user is prompted to enter the robot's starting position, which is validated to ensure it falls within the grid boundaries. Once the robot is initialized, it can move in four directions—up, down, left, and right—and deliver parcels when it reaches a cell marked as a delivery point. With each move, the grid is updated in real time to reflect the robot's current position and the remaining delivery points.

- Unique delivery points are generated within the grid, ensuring each point is distinct.
- The robot uses the min() function with Manhattan distance as the metric to order the remaining delivery points and select the closest target, providing a simple yet effective linear search mechanism.
- The grid is dynamically updated after every move and delivery, offering continuous visual feedback.

This task demonstrates the integration of user input, dynamic grid updates, and a basic ordering mechanism that drives the robot's decision-making, setting the stage for fully autonomous navigation in Task 3.e as the metric) to select the closest delivery point. This is a linear search, not a full sort.

Example output:

```
Enter the grid size (N x N, where N is between 1 and 6):  3
Enter the robot's starting position (x y):  2 2

Generated Environment:
(1,1) Clear | (1,2) Clear | (1,3) Delivery
(2,1) Clear | (2,2) Robot | (2,3) Clear
(3,1) Clear | (3,2) Delivery | (3,3) Delivery

Moved right to (2,3)
(1,1) Clear | (1,2) Clear | (1,3) Delivery
(2,1) Clear | (2,2) Delivery | (2,3) Robot
(3,1) Clear | (3,2) Delivery | (3,3) Delivery

Moved down to (3,3)
(1,1) Clear | (1,2) Clear | (1,3) Delivery
(2,1) Clear | (2,2) Delivery | (2,3) Clear
(3,1) Clear | (3,2) Delivery | (3,3) Robot

Delivered at (3,3)
(1,1) Clear | (1,2) Clear | (1,3) Delivery
(2,1) Clear | (2,2) Delivery | (2,3) Clear
(3,1) Clear | (3,2) Delivery | (3,3) Robot

Deliveries remaining: {(3, 2), (1, 3), (2, 2)}
```

Task 3: Run the Agent

In Task 3, the focus is on automating the Smart Delivery Robot's navigation so that it continuously selects and moves toward the nearest delivery point. Building on the previous tasks, the robot now uses the same ordering mechanism—leveraging the min() function with Manhattan distance as its heuristic—to determine the closest delivery target. As the robot moves one cell at a time in the optimal direction, the grid is updated dynamically, providing real-time feedback of its progress. When the robot reaches a delivery point, it executes the delivery and then re-evaluates the remaining targets, repeating this process until all parcels have been delivered.

- The robot autonomously selects the nearest delivery point using a linear search based on Manhattan distance.
- With every move, the grid is refreshed to show the updated robot position and remaining delivery points.
- The process continues in a loop until there are no more delivery points left, at which point the program informs the user that all deliveries have been completed.

This task effectively demonstrates the integration of real-time decision-making, dynamic visual updates, and autonomous navigation, thereby completing the core objectives of the project.

Example output:

```
Enter the grid size (N x N, where N is between 1 and 6):  5
Enter the robot's starting position (x y):  2 2

Generated Environment:
(1,1) Delivery | (1,2) Clear | (1,3) Clear | (1,4) Clear | (1,5) Clear
(2,1) Delivery | (2,2) Robot | (2,3) Clear | (2,4) Clear | (2,5) Delivery
(3,1) Clear | (3,2) Clear | (3,3) Delivery | (3,4) Delivery | (3,5) Clear
(4,1) Clear | (4,2) Clear | (4,3) Clear | (4,4) Clear | (4,5) Delivery
(5,1) Clear | (5,2) Clear | (5,3) Clear | (5,4) Clear | (5,5) Clear

(1,1) Delivery | (1,2) Clear | (1,3) Clear | (1,4) Clear | (1,5) Clear
(2,1) Delivery | (2,2) Robot | (2,3) Clear | (2,4) Clear | (2,5) Delivery
(3,1) Clear | (3,2) Clear | (3,3) Delivery | (3,4) Delivery | (3,5) Clear
(4,1) Clear | (4,2) Clear | (4,3) Clear | (4,4) Clear | (4,5) Delivery
(5,1) Clear | (5,2) Clear | (5,3) Clear | (5,4) Clear | (5,5) Clear

 ##
.....
.....
 ## keeps searching and delivering packages for many more lines....
 ##
.....
.....

Delivered at (2,5)
(1,1) Clear | (1,2) Clear | (1,3) Clear | (1,4) Clear | (1,5) Clear
(2,1) Clear | (2,2) Clear | (2,3) Clear | (2,4) Clear | (2,5) Robot
(3,1) Clear | (3,2) Clear | (3,3) Clear | (3,4) Clear | (3,5) Clear
(4,1) Clear | (4,2) Clear | (4,3) Clear | (4,4) Clear | (4,5) Clear
(5,1) Clear | (5,2) Clear | (5,3) Clear | (5,4) Clear | (5,5) Clear
```

```
All deliveries completed!
```

Advanced Component Task 1: Enhanced Environment

In this task, we extend the basic grid environment by introducing realistic urban constraints. In addition to generating delivery points, the program now randomly places obstacles, no-entry zones, and one-way streets within the grid. Each cell is clearly labeled to indicate whether it is "Clear," contains a "Delivery," an "Obstacle," a "No-Entry" zone, or a "OneWay" direction. This enhanced environment is designed to simulate real-world urban challenges and test the robot's ability to navigate through more complex and constrained spaces. The task not only adds depth to the simulation but also sets the stage for integrating advanced pathfinding techniques in subsequent tasks.

- The grid now includes additional constraints that provide realistic navigation challenges.
- Randomly generated obstacles, no-entry zones, and one-way streets are clearly marked on the grid.
- This task prepares the robot for the more sophisticated navigation and optimization required in advanced components.

This enhancement ensures that the Smart Delivery Robot must contend with a more dynamic and realistic urban landscape, further demonstrating its adaptability and the robustness of its navigation system.

Example output:

```
Enter grid size (N x N, where N is between 1 and 6):  5

Advanced Environment:
(1,1) Delivery | (1,2) Clear | (1,3) Clear | (1,4) OneWay:down | (1,5) Clear
(2,1) Clear | (2,2) Clear | (2,3) OneWay:left | (2,4) No-Entry | (2,5) Obstacle
(3,1) No-Entry | (3,2) OneWay:left | (3,3) Obstacle | (3,4) OneWay:down | (3,5) Clear
(4,1) OneWay:down | (4,2) Clear | (4,3) No-Entry | (4,4) Clear | (4,5) Obstacle
(5,1) Clear | (5,2) No-Entry | (5,3) OneWay:up | (5,4) Clear | (5,5) Clear
```

Advanced Component Task 2: Pathfinding and Optimization

In this task, the focus shifts to refining the robot's navigation by implementing an A* search algorithm for optimal pathfinding. This algorithm calculates the best possible route to a target delivery point by using Manhattan distance as a heuristic, while also considering obstacles, no-entry zones, and one-way street constraints. The robot follows the computed path, updating the grid dynamically after each move, and if the environment changes, the path is recalculated in real time. This task not only improves the efficiency of the delivery process but also demonstrates how advanced AI techniques can be applied to solve complex navigation problems in a dynamic urban setting.

- The A* search algorithm uses a priority queue (via heapq) to order nodes based on their current path cost plus the heuristic, ensuring efficient retrieval of the optimal next step.
- The algorithm successfully navigates through the enhanced environment by avoiding constraints and recalculating the path when necessary.

- The dynamic grid updates during movement provide real-time feedback of the robot's progress, confirming that the optimal path is followed until all deliveries are completed.

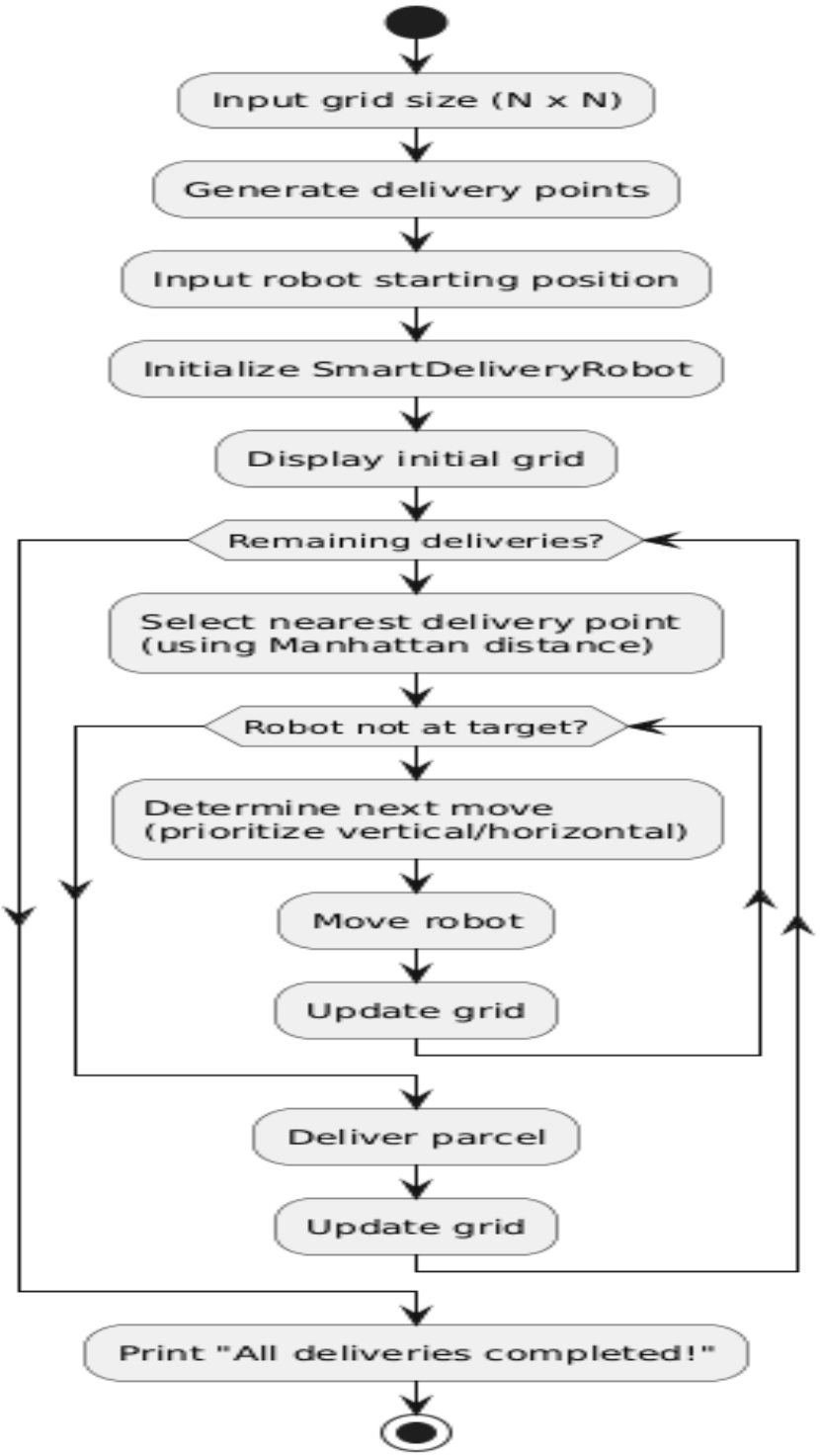
Example output:

```
Enter grid size (N x N, where N is between 1 and 6):  5
Enter robot starting position (x y):  3 2

Advanced Environment:
(1,1) Clear | (1,2) Clear | (1,3) Clear | (1,4) Clear | (1,5) Delivery
(2,1) Delivery | (2,2) Delivery | (2,3) Delivery | (2,4) Obstacle | (2,5) NoEntry
(3,1) Clear | (3,2) NoEntry | (3,3) Delivery | (3,4) Delivery | (3,5) NoEntry
(4,1) OneWay:down | (4,2) Clear | (4,3) Delivery | (4,4) Delivery | (4,5) Delivery
(5,1) NoEntry | (5,2) Clear | (5,3) NoEntry | (5,4) Clear | (5,5) Delivery

(1,1) Clear | (1,2) Clear | (1,3) Clear | (1,4) Clear | (1,5) Delivery
(2,1) Delivery | (2,2) Delivery | (2,3) Delivery | (2,4) Obstacle | (2,5) NoEntry
(3,1) Clear | (3,2) Robot | (3,3) Delivery | (3,4) Delivery | (3,5) NoEntry
(4,1) OneWay:down | (4,2) Clear | (4,3) Delivery | (4,4) Delivery | (4,5) Delivery
(5,1) NoEntry | (5,2) Clear | (5,3) NoEntry | (5,4) Clear | (5,5) Delivery

Path to (3, 3): [(3, 2), (3, 3)]
Moved to (3,3)
(1,1) Clear | (1,2) Clear | (1,3) Clear | (1,4) Clear | (1,5) Delivery
(2,1) Delivery | (2,2) Delivery | (2,3) Delivery | (2,4) Obstacle | (2,5) NoEntry
(3,1) Clear | (3,2) NoEntry | (3,3) Robot | (3,4) Delivery | (3,5) NoEntry
(4,1) OneWay:down | (4,2) Clear | (4,3) Delivery | (4,4) Delivery | (4,5) Delivery
(5,1) NoEntry | (5,2) Clear | (5,3) NoEntry | (5,4) Clear | (5,5) Delivery

Delivered at (3,3)

 ##
.....
.....
 ## keeps searching and delivering packages for many more lines....
 ##
.....
.....

Delivered at (5,5)
(1,1) Clear | (1,2) Clear | (1,3) Clear | (1,4) Clear | (1,5) Clear
(2,1) Clear | (2,2) Clear | (2,3) Clear | (2,4) Obstacle | (2,5) NoEntry
(3,1) Clear | (3,2) NoEntry | (3,3) Clear | (3,4) Clear | (3,5) NoEntry
(4,1) OneWay:down | (4,2) Clear | (4,3) Clear | (4,4) Clear | (4,5) Clear
(5,1) NoEntry | (5,2) Clear | (5,3) NoEntry | (5,4) Clear | (5,5) Robot

All deliveries completed!
```

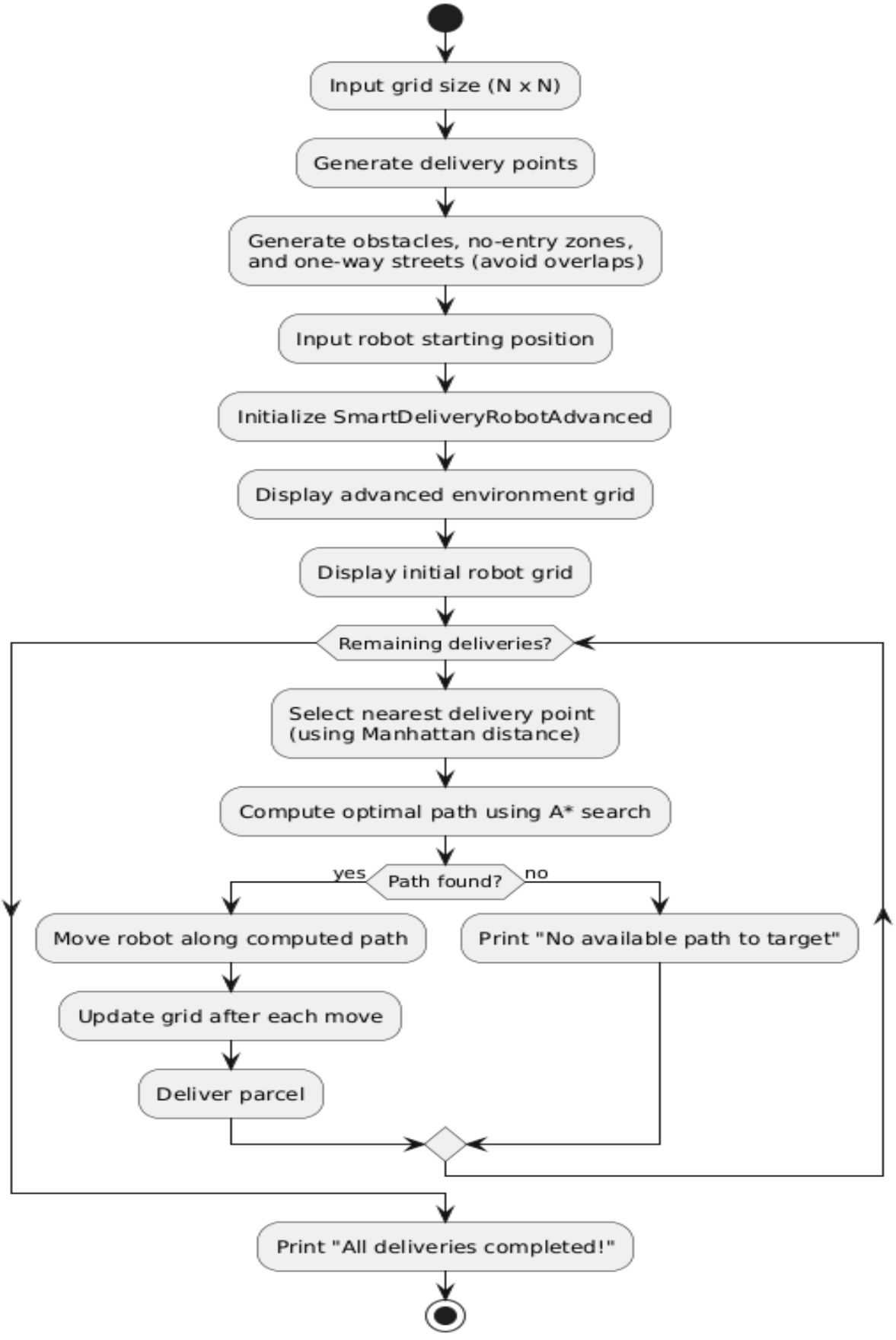
Flowcharts:

Below I will provide two flowcharts for the Delivery Programs that I modeled and used to help me complete the code.

Task 3:

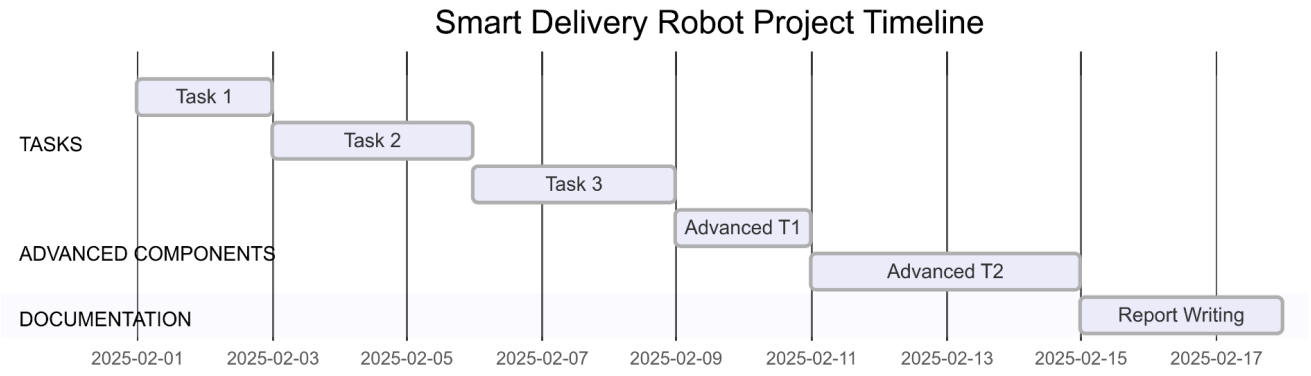


Advanced Task 2:


Input grid size (N x N)
Generate delivery points
Generate obstacles, no-entry zones, and one-way streets (avoid overlaps)
Input robot starting position
Initialize SmartDeliveryRobotAdvanced
Display advanced environment grid
Display initial robot grid
Remaining deliveries?
Select nearest delivery point (using Manhattan distance)
Compute optimal path using A* search
Path found?
yes
no
Move robot along computed path
Print "No available path to target"
Update grid after each move
Deliver parcel
Print "All deliveries completed!"

Gantt Chart:

This is the chart that i used to plan my project:



Ethical Implications:

The deployment of autonomous systems, such as our Smart Delivery Robot, inevitably raises important ethical issues that extend beyond technical performance. As our project demonstrates, while AI can optimize routes and improve delivery efficiency, it also presents challenges in transparency, fairness, and data privacy. For example, the robot's decision-making—though based on clear heuristics like Manhattan distance—is a simplified reflection of the more complex ethical dilemmas encountered in real-world AI applications, where biases can be inadvertently reinforced. According to UNESCO, AI systems are prone to perpetuating stereotypes and inequalities if not carefully managed, underscoring the need for robust safety measures and transparency in how algorithms make decisions.

Moreover, the integration of advanced pathfinding and dynamic routing in our project highlights concerns about data privacy, as continuous processing of location data is required to function effectively. UNESCO emphasizes that ethical frameworks must be in place to safeguard individual rights and ensure that AI technologies are both fair and accountable. This project, while primarily focused on technical implementation, serves as a reminder that as we automate more aspects of our daily lives, we must also carefully consider and address the broader societal and ethical implications of these innovations.

Additionally, AI holds significant promise for educational advancements. For example, during this project I discovered a new method to generate professional flowcharts and Gantt charts using code, which not only improved the visual quality of my diagrams but also saved a considerable amount of time. I used PlantUML to create the flowcharts and Mermaid (a language designed for generating diagrams and visualizations) for the Gantt chart. This approach allows for the seamless display of HTML-based charts through automated tools, eliminating the need for students to produce rough hand-drawn sketches. By integrating these tools into the learning process, students can focus more on problem-solving and critical thinking, rather than spending excessive time on diagram creation.

Bibliography

[1]
Arwen Vaughan, “PlantText UML Editor,” *Planttext.com*, 2013. https://www.planttext.com/
[2]
“Online FlowChart & Diagrams Editor - Mermaid Live Editor,” *mermaid.live*. https://mermaid.live/
[3]
R. Belwariar, “A* Search Algorithm - GeeksforGeeks,” *GeeksforGeeks*, Sep. 07, 2018. https://www.geeksforgeeks.org/a-search-algorithm
[4]
UNESCO, “Artificial Intelligence: Examples of Ethical Dilemmas,” *www.unesco.org*, Apr. 21, 2023. https://www.unesco.org/en/artificial-intelligence/recommendation-ethics/cases
[5]
GeeksforGeeks, “Heuristic Function In AI,” *GeeksforGeeks*, Jun. 16, 2024. https://www.geeksforgeeks.org/heuristic-function-in-ai/