Dynamic Maze Solving using D*-Lite and Dead-End Exclusion Algorithm

Department of Electronics and Computer Engineering

Institute of Engineering, Thapathali Campus

2021/03/09

Prepared by:

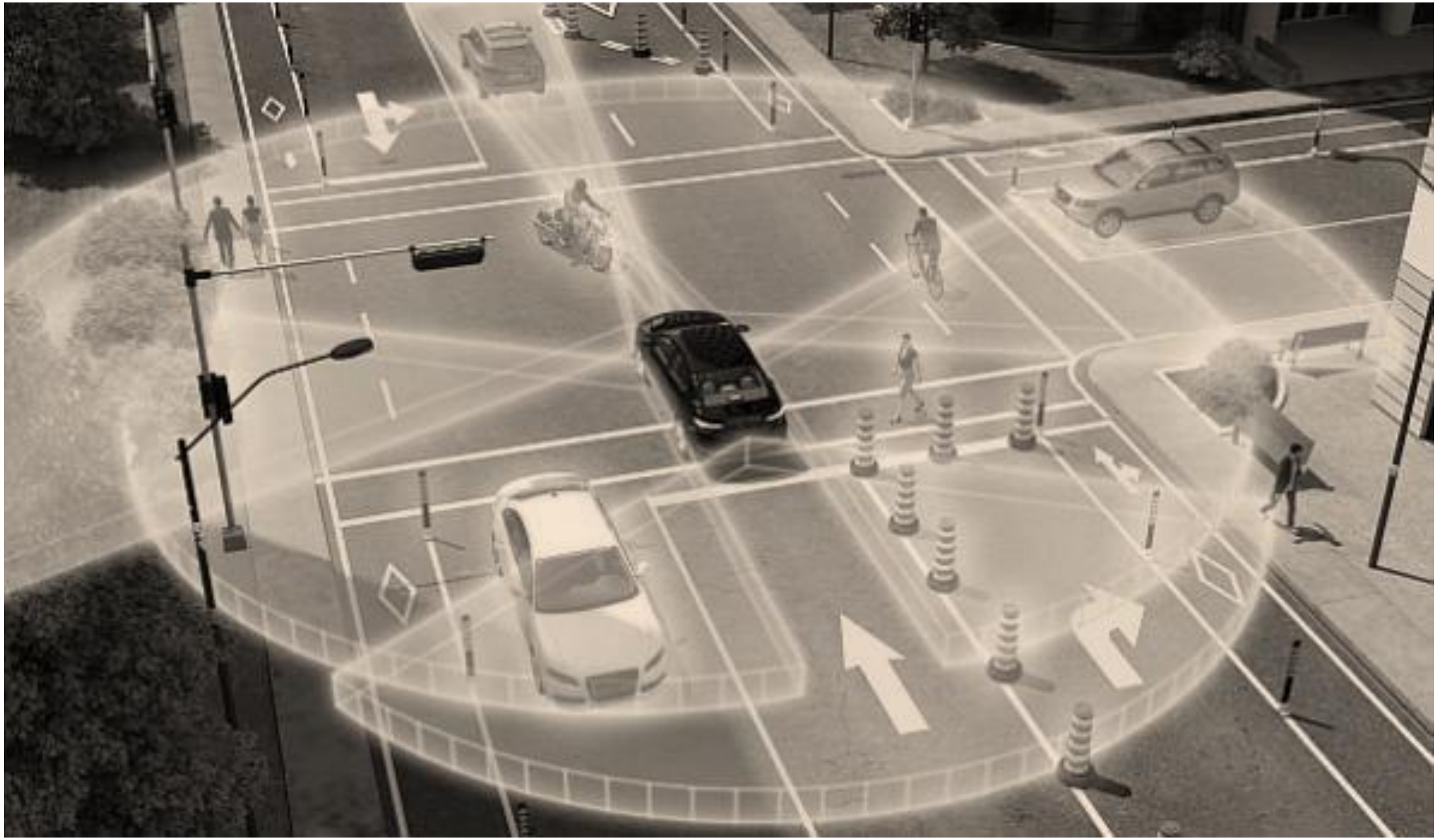
- **Anish Dahal (THA074BEX004)**
- **Dipak Gurung (THA074BEX009)**
- **Prajwol Pakka (THA074BEX022)**
- **Sujal Subedi (THA074BEX043)**

Supervised by:
Er. Dinesh Baniya Kshatri
Lecturer

Presentation Outline

- Motivation
- Project Objectives
- Scope of Project
- Project Applications
- Requirement Analysis
- Methodology
- Result
- Discussion and Analysis
- Future Enhancements
- Conclusion
- Project schedule
- References

Motivation



Project Objectives

- To solve a dynamic maze with real-time obstacles
- To use Dead-End Exclusion algorithm for maze mapping
- To use D*-Lite algorithm for goal seeking

Scope of Project

- Project Capabilities
  - Navigating the maze while avoiding the dead ends
  - Finding the shortest path to the goal from start
  - Choosing the next shortest path after encountering an obstacle

- Project Limitations
  - Events of removing obstacles are not accounted
  - Robot stops moving if there is no possible path to the goal

Project Applications

- Disaster Response
  - Provide essential supplies to people trapped during disasters
- Speleology
  - Exploring caves where humans could get lost
- Warehouse Robots
  - Ordering of goods in shelves within a warehouse
- Cleaning Robots
  - Tidies rooms using the floorplan as a map
- Delivery Robots
  - Semi-autonomous or fully-autonomous distribution of customer orders

Requirement Analysis

- Webots Simulator (R2020B revision1)

  - Sensor Library: Distance sensor, IMU sensor

  - Joint library: Hinge joint, Ball joint

  - Actuator Library: Encoder, Rotational motor

  - C++ Language is used to design the controller

Methodology
(Working Mechanism of System)

- Robot perceives the environment via the sensors

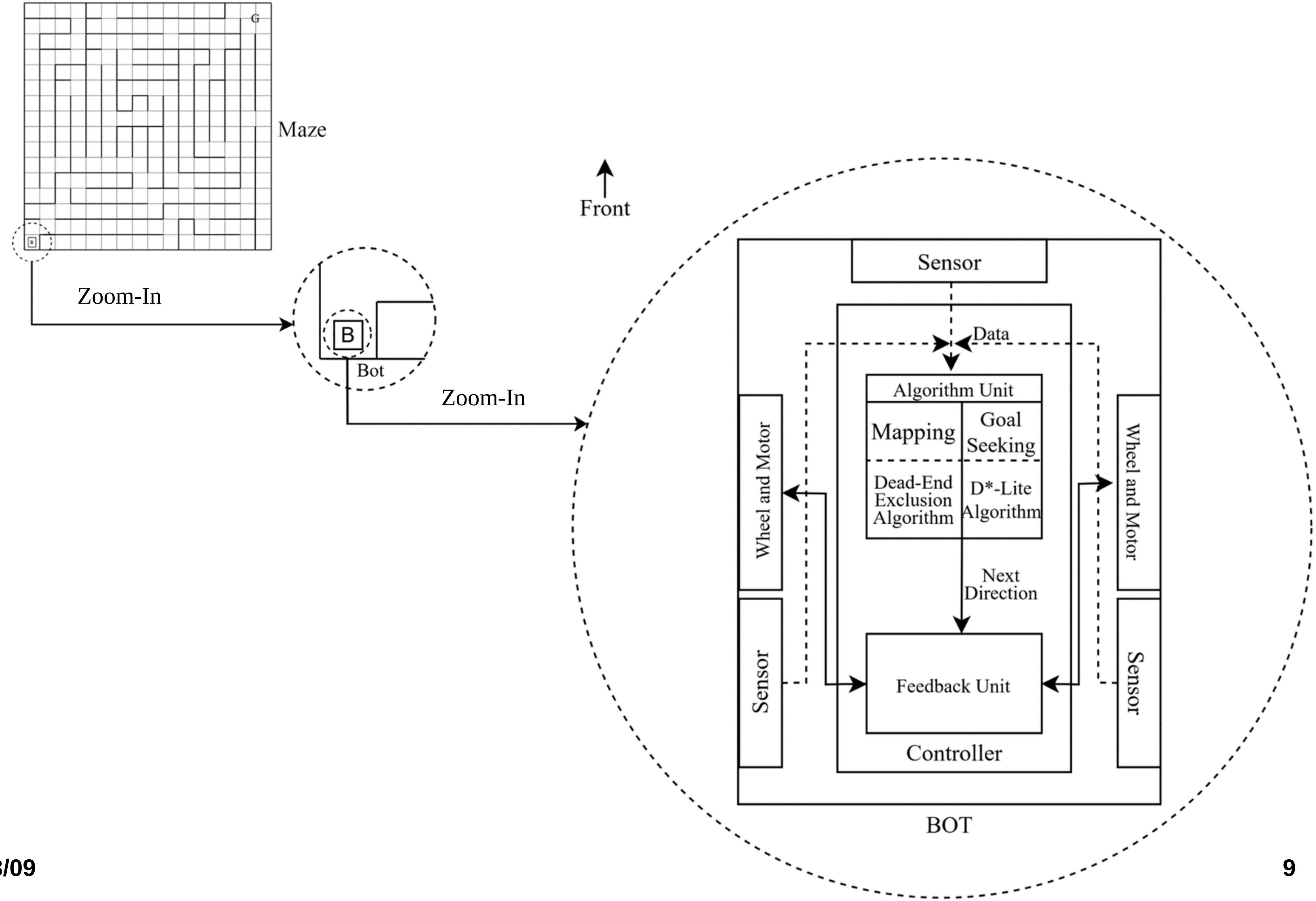
- Controller maintains overall process of mapping and goal-seeking

- Robot performs movement according to output of algorithm

- Feedback controller takes response from the encoders

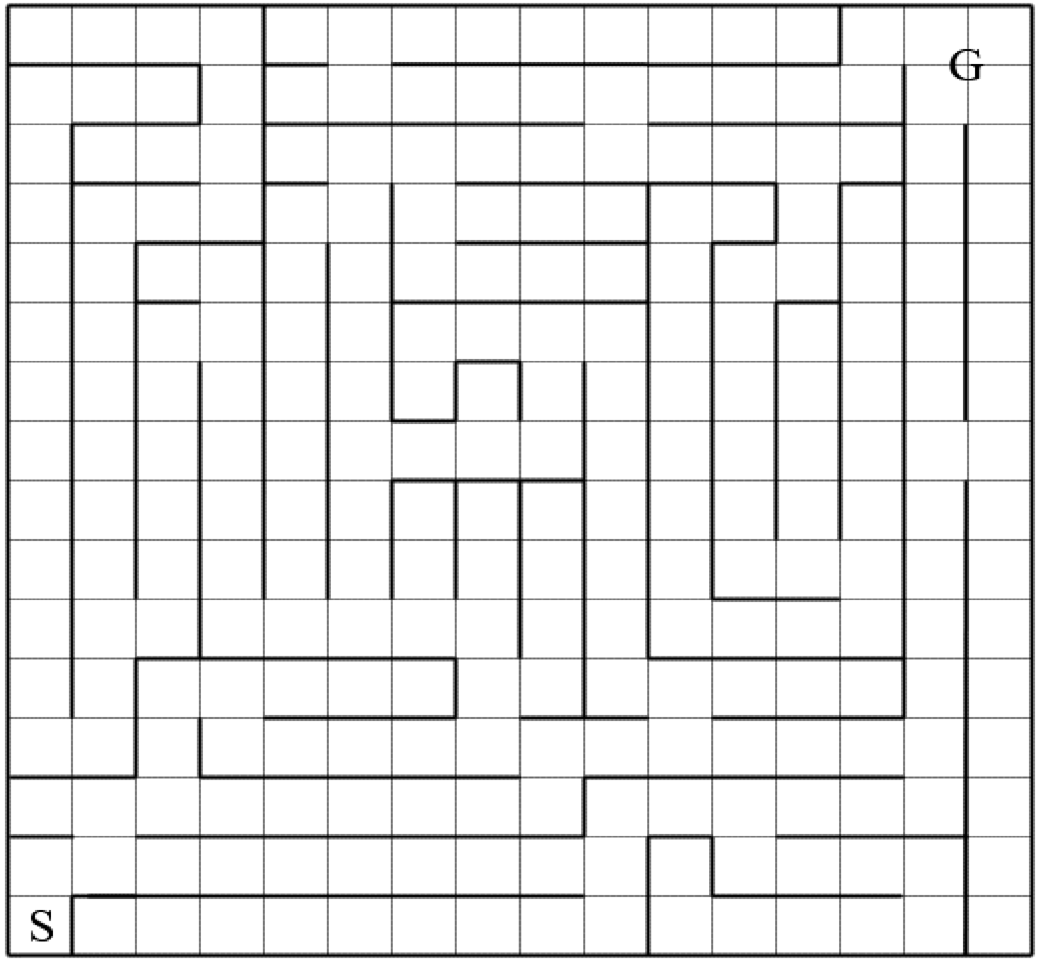
- Robot stops moving if:
  - Destination is reached, or
  - If there is no obstacle free path to goal

Methodology
(System Block Diagram)



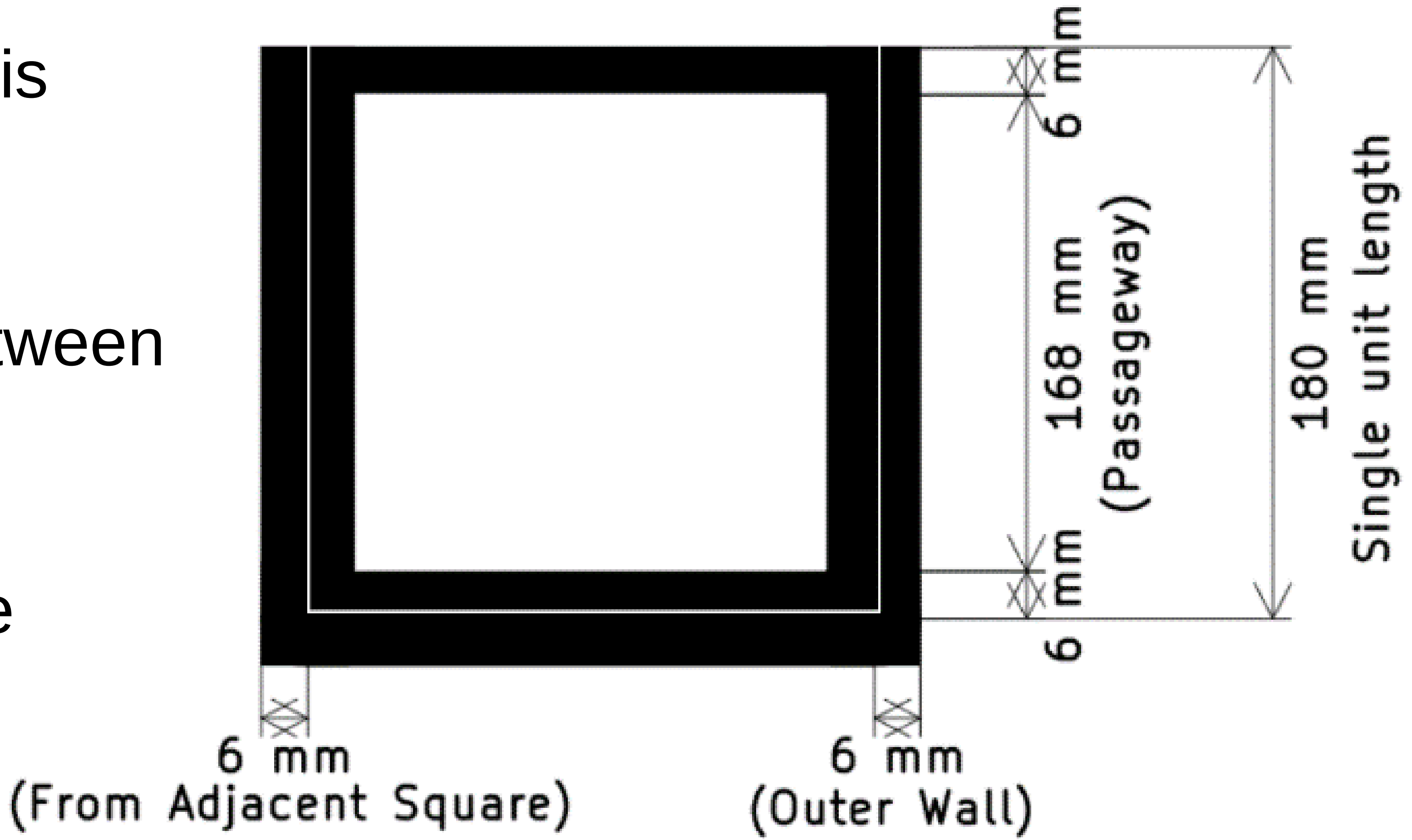
Methodology
(Maze Structure)

- Maze is formed by a 16 × 16 square array
- Start of the maze is located at one of the four corners
- Starting square has walls on three sides
- Goal is a large opening which is composed of 4-unit squares



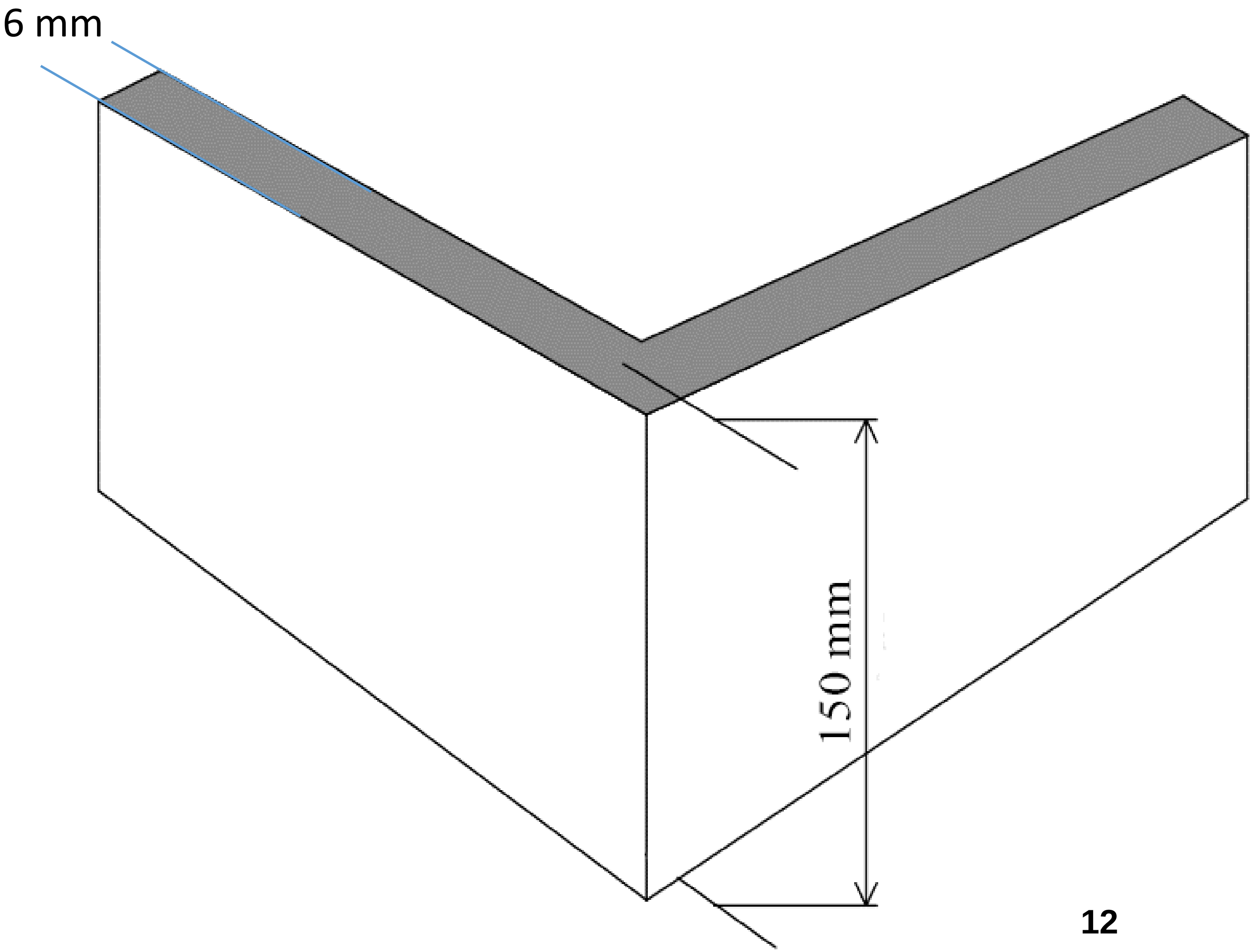
Methodology
(Unit Square of Maze)

- Each cell dimension is 180 mm × 180 mm
- The passageway between the walls is 168 mm
- The inside wall of the maze is 12 mm thick



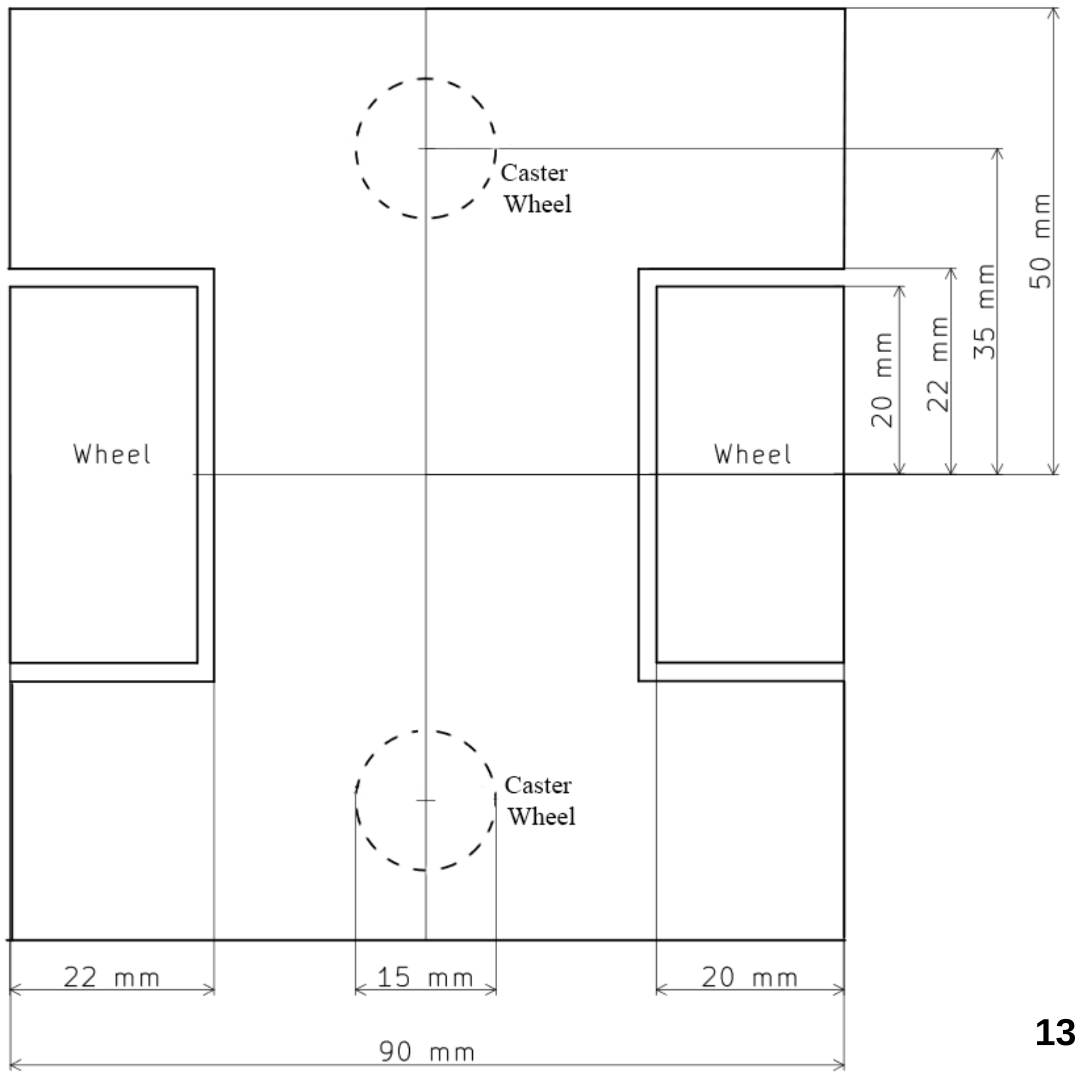
Methodology
(Wall of Maze)

- The walls surrounding the maze is 150 mm high
- The outside wall having width of 6 mm is enclose the entire maze
- The height is constant throughout the maze



Methodology (Top View of Robot)

- The robot size is of 100 mm × 90 mm
- The normal wheels is of 40 mm in diameter and 20 mm wide
- The caster wheels is of diameter of 15 mm

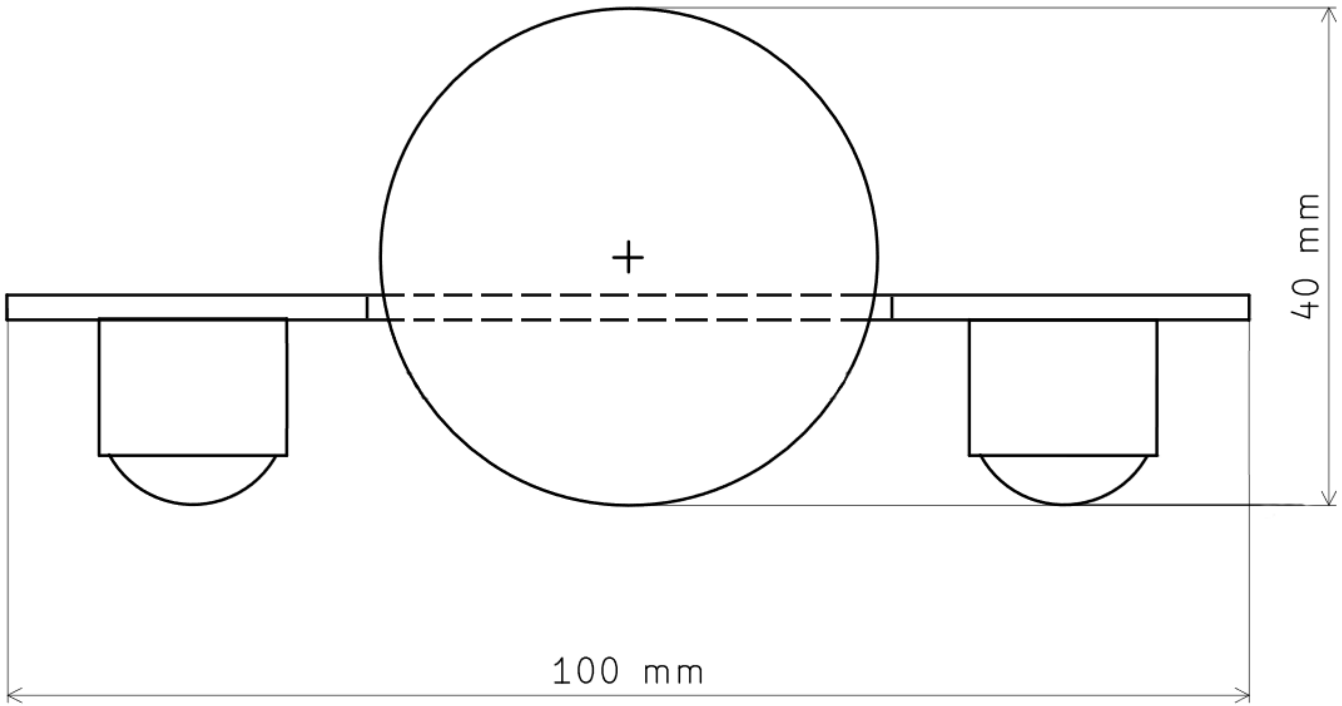


Methodology
(Side View of Robot)

- The total height of bot is 40 mm

- The caster used is15 mm in height

- The left and right views are similar

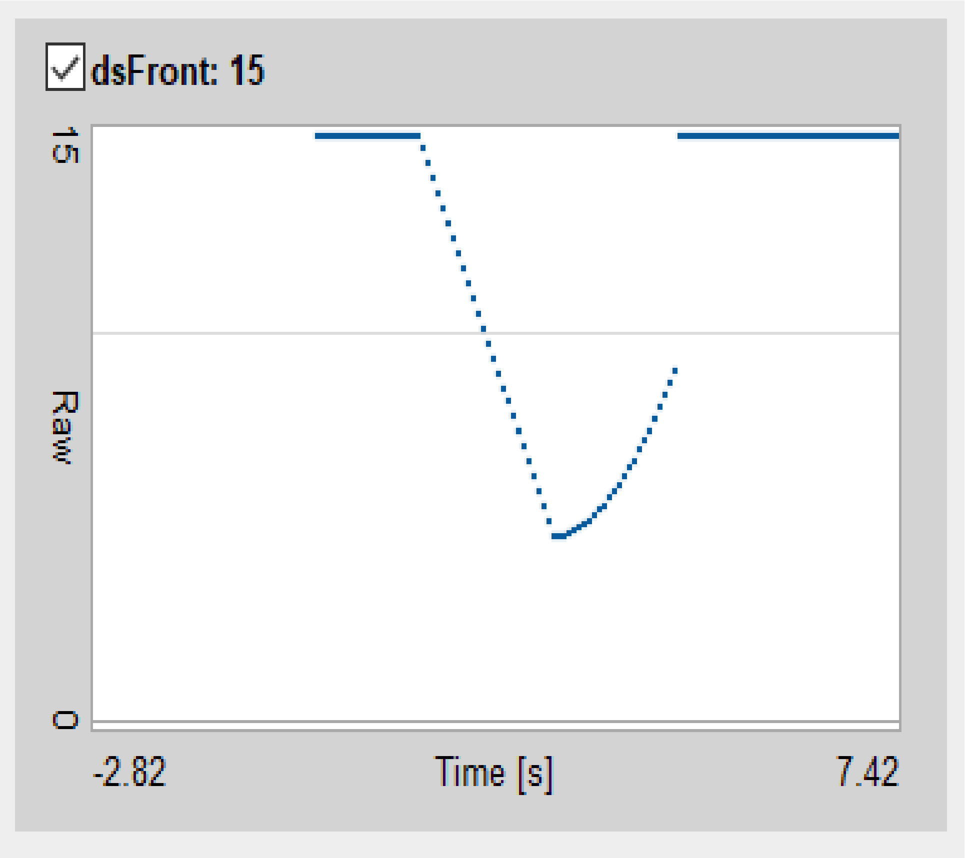


Methodology
(Functions of Robot's Components)

- Distance sensor is used to know the position of wall
- Controller takes sensor data and sends control signals to motor
- Motors rotates wheels and encoder provides distance travelled
- Encoder data is sent to controller to calculate position of robot
- IMU sensor provides the orientation of the robot in a cell

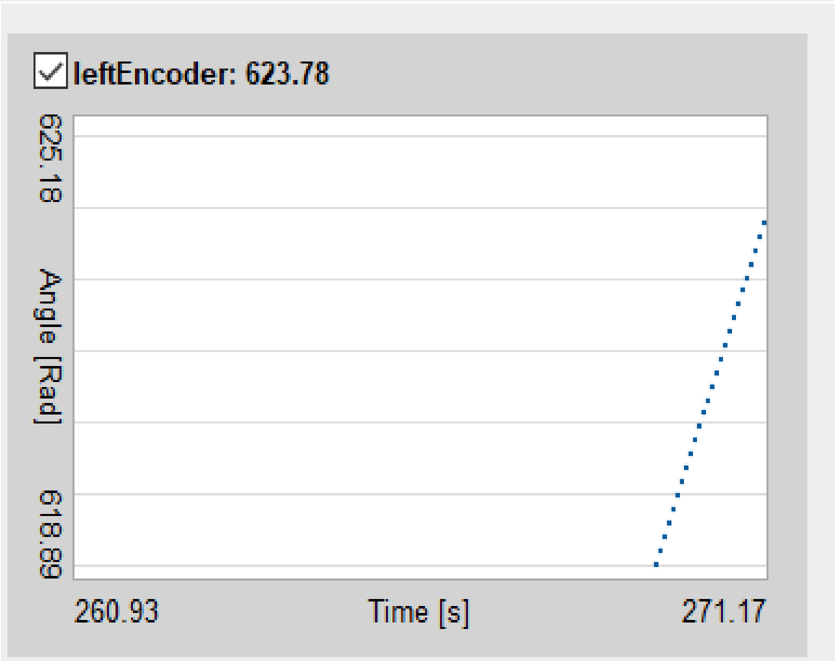
Methodology
(Distance Sensors)

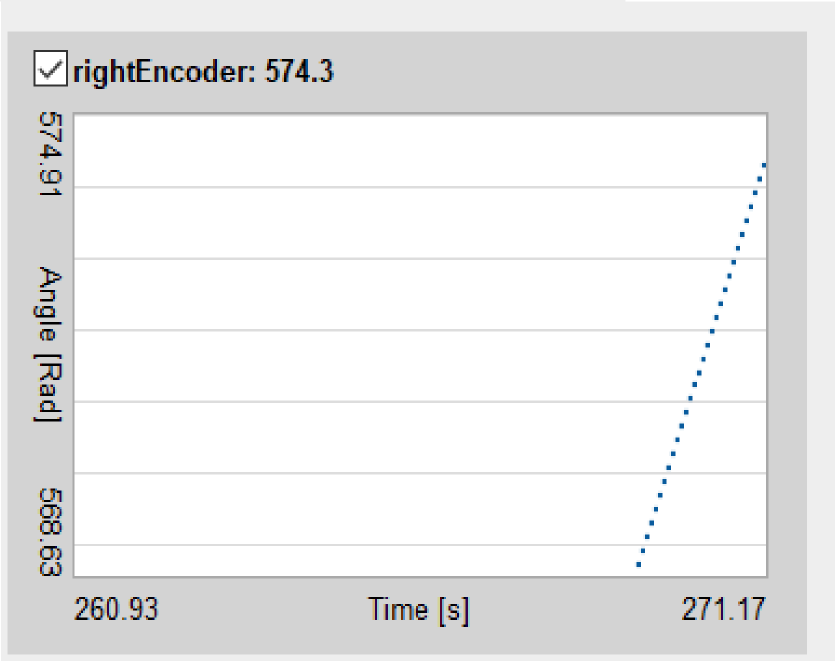
- Distance sensor provides an analog output of 0-15
- The outputs provided by the sensor corresponds to the distance in cm
- Value 15 is outputted when no wall is detected
- When wall or any obstacle is detected, the value is less then 15



Methodology (Encoders) – [1]

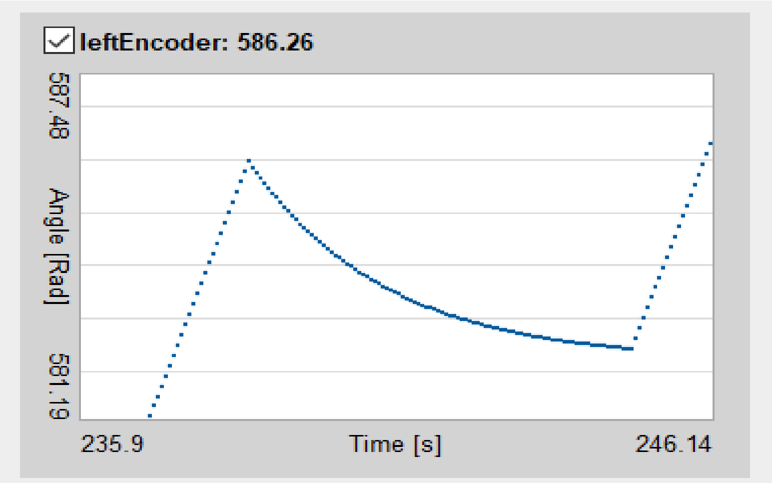
- They provide angular rotation in radians
- Linear distance is calculated by multiplying angular rotation with the radius of the wheel

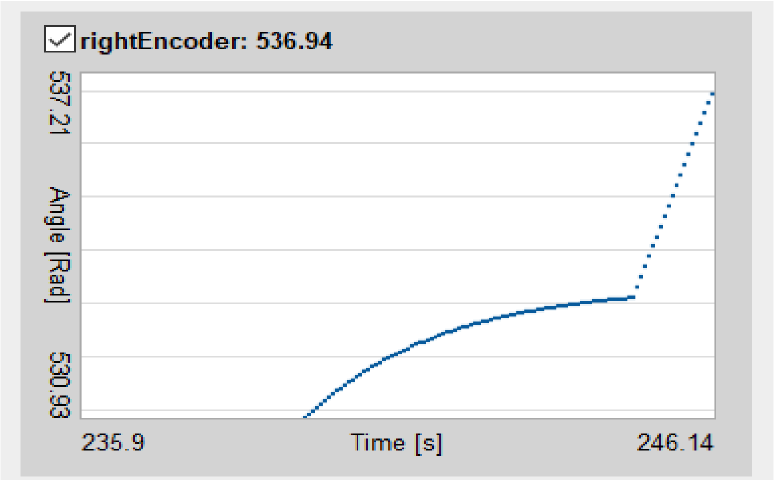




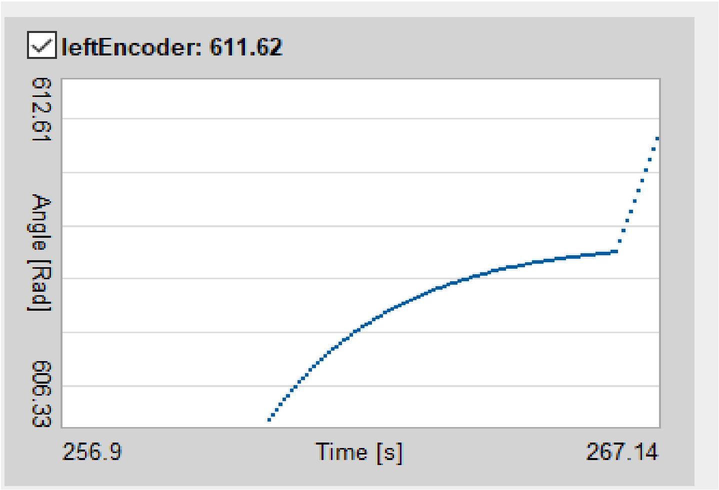
Methodology (Encoders) – [2]

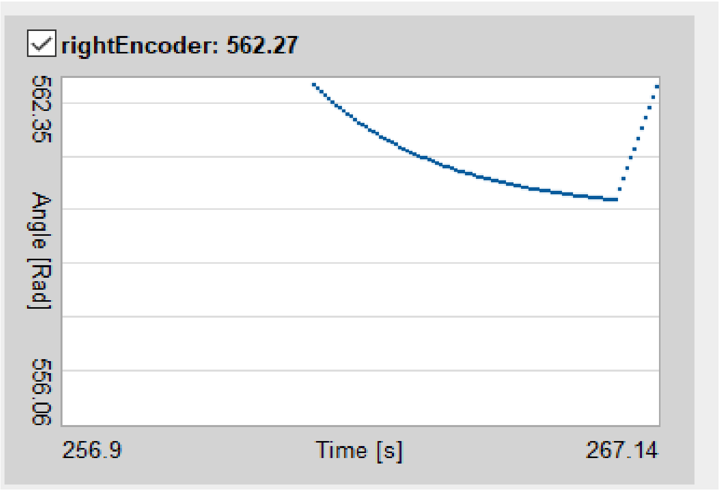
- For anti-clockwise turn, right encoder data will increase and left will decrease





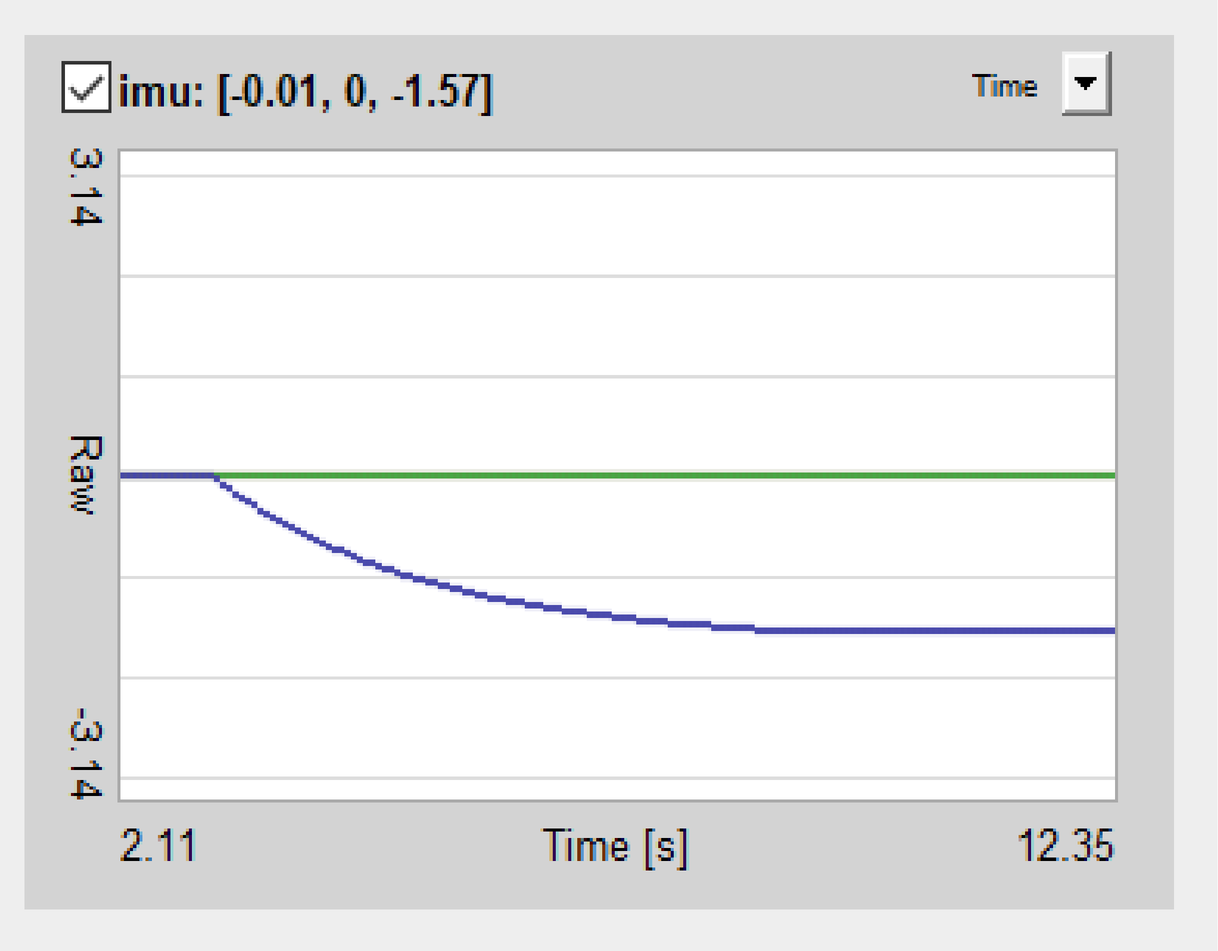
- For clockwise turn, left encoder data will increase and right will decrease





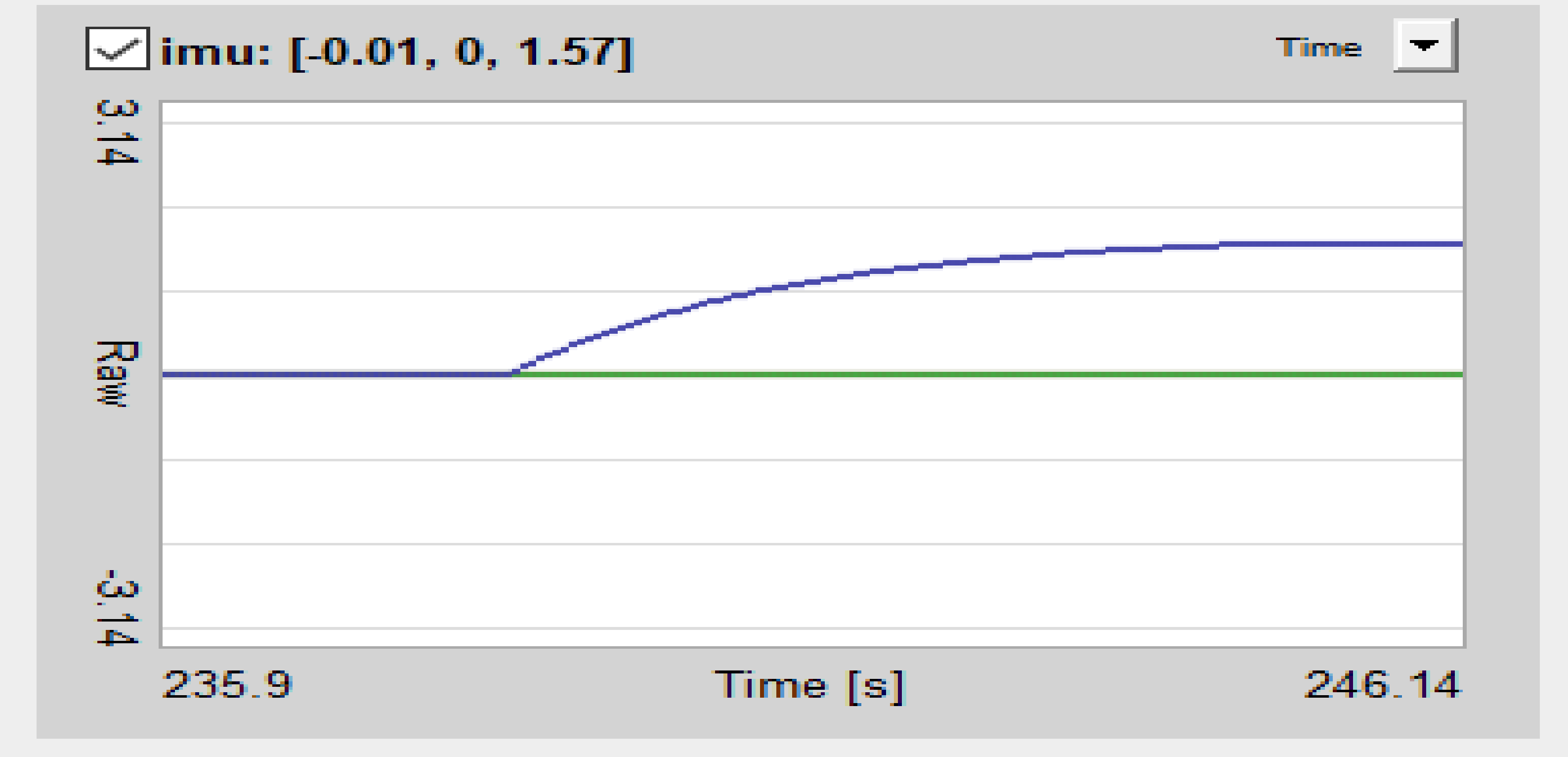
Methodology (IMU Sensor) – [1]

- Used to calculate the orientation of the robot
- Provides the roll, pitch and yaw of the robot
- Only yaw is considered, which is the rotation around Y-axis
- Yaw data changes by -1.57 radian during clockwise turn



Methodology (IMU Sensor) – [2]

- Yaw data changes by +1.57 radian during counter clockwise turn
- Among the three data shown in figure, the third data relates to yaw



Methodology
(Need for PID Tuning)

- Positional error arises while performing 90° or 180° rotation of the robot

- Error is calculated from the left and right distance sensors
  - $Error\ = Left\ Distance\ Sensor\ value - \ Right\ Distance\ sensor\ Value$

- If the error is not corrected, the robot moves off track and collides with a side wall

- To mitigate the above positional error PID tuning is applied

Methodology
(Maze Solving Process)

- The maze solving takes place in two steps
  - Maze Exploration
  - Goal Seeking

- Maze Exploration refers to virtual mapping of the maze

- Maze Exploration is carried out through Dead-End algorithm

- Goal Seeking represents finding optimal path

- Goal Seeking is carried out using D*-Lite algorithm

Methodology
(Description of Dead-End Exclusion Algorithm)

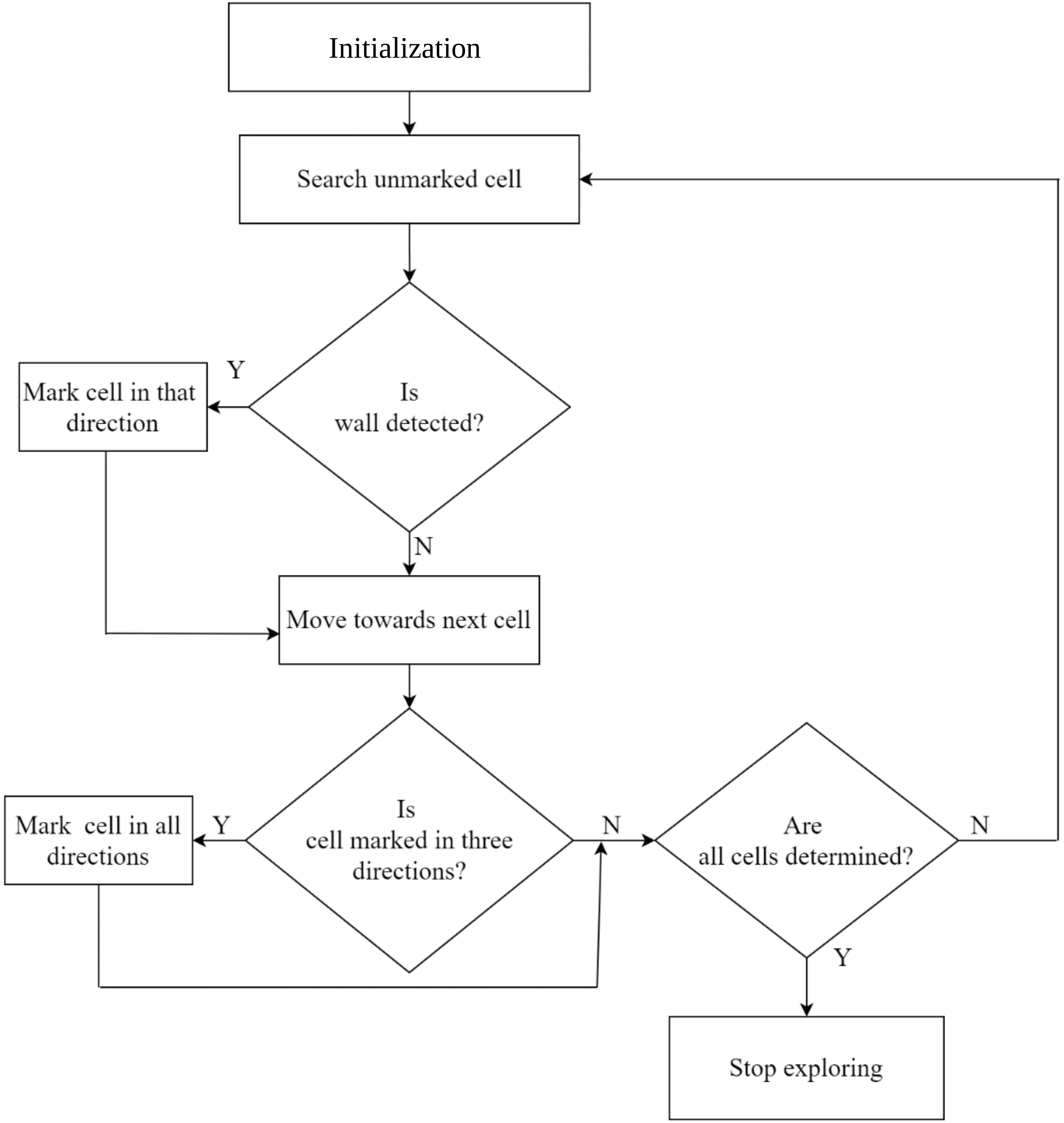
- Cells touching the edges of the maze are marked in the direction of the wall

- The robot searches and moves towards an unmarked cell

- If a wall is detected, the cell is marked towards the wall

- Close the cell if it is marked in three directions and treat it as a virtual wall

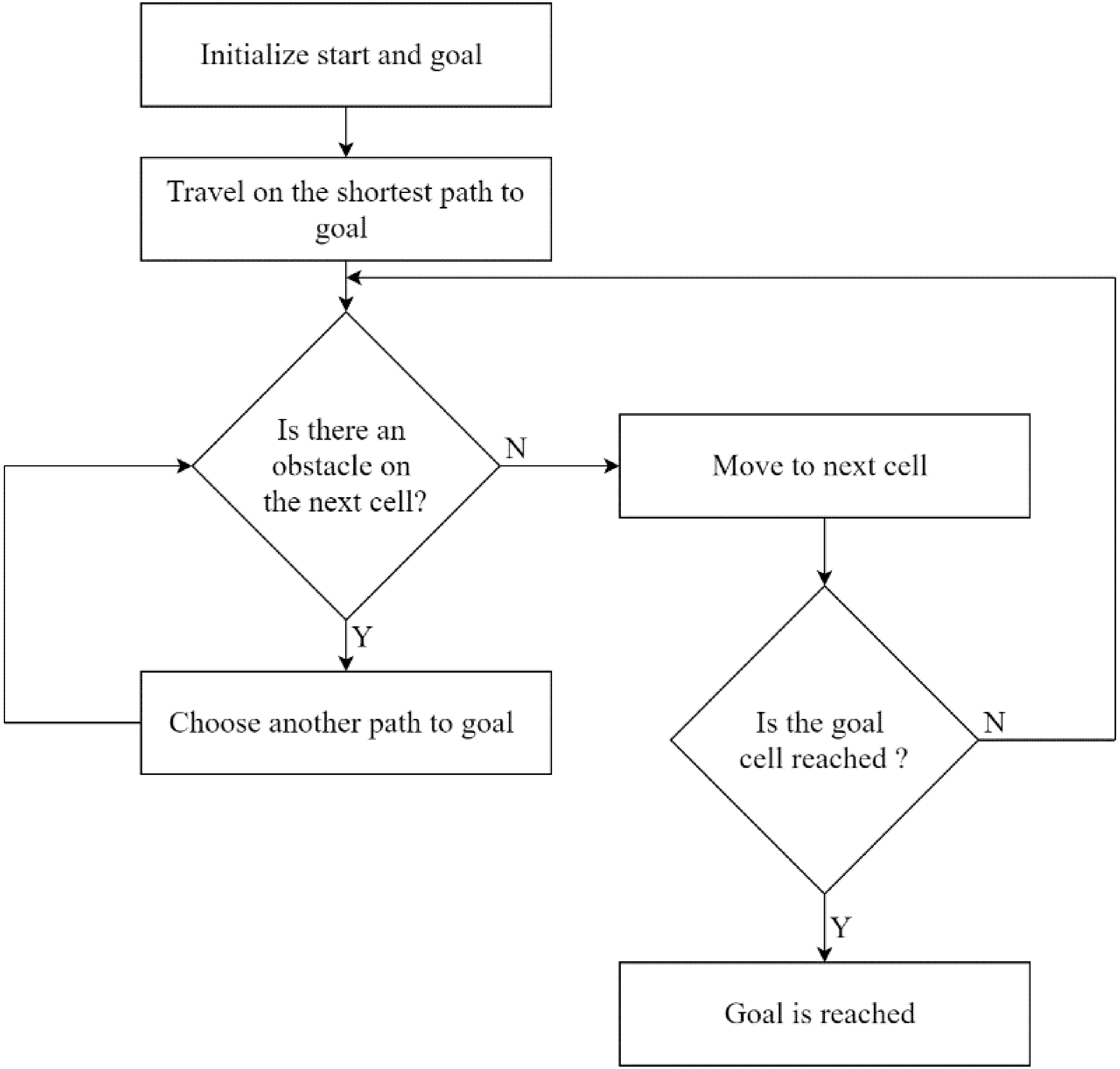
Methodology
(Dead-End Exclusion Flowchart)



Methodology
(Description of D*-Lite Algorithm)

- Initializes start and goal
- Travel on the shortest path to the goal
- Choose another path when obstacle is detected
- Move on the next selected path
- Robot stops moving when goal is reached

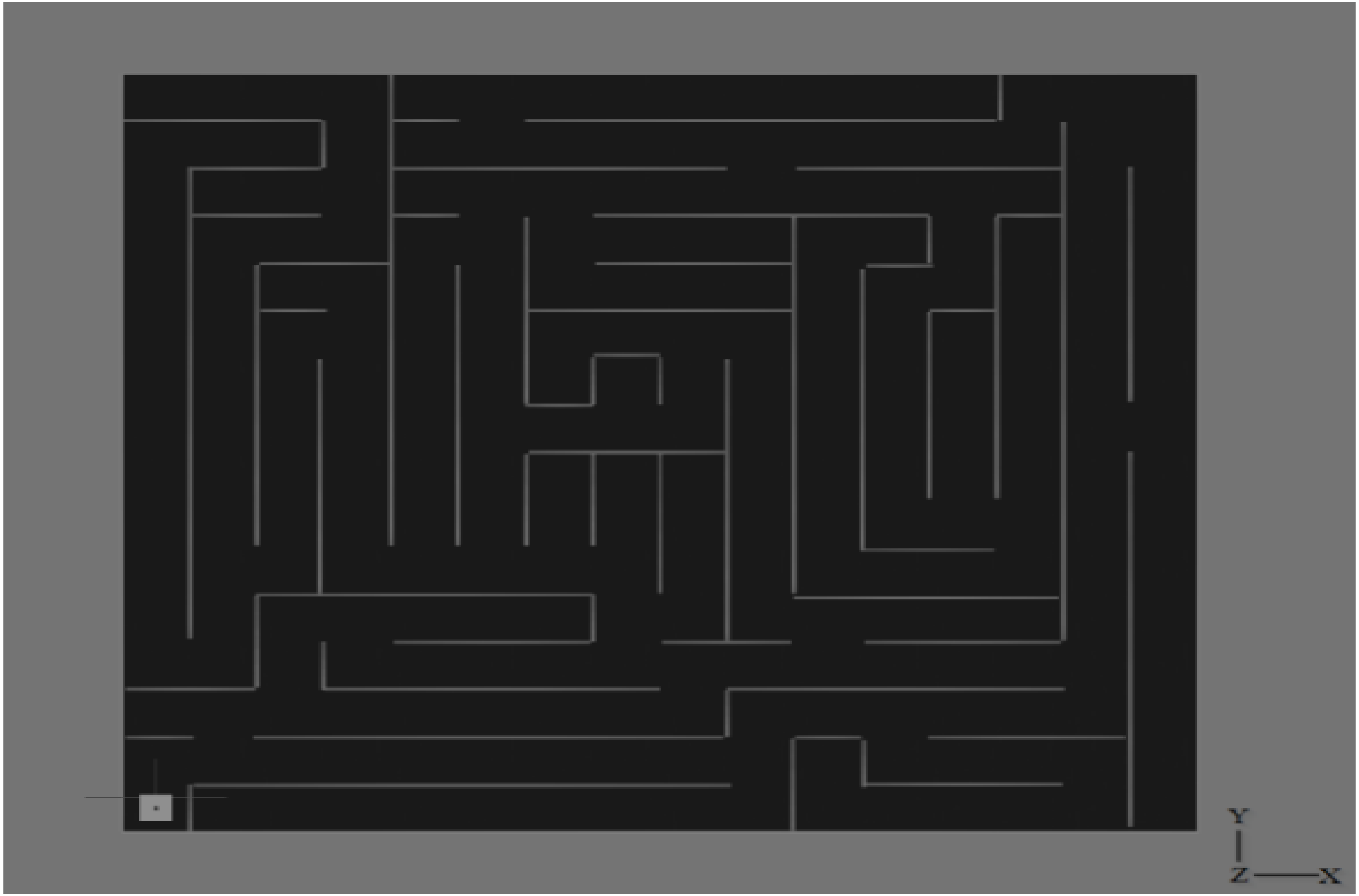
Methodology (D*-Lite Flowchart)



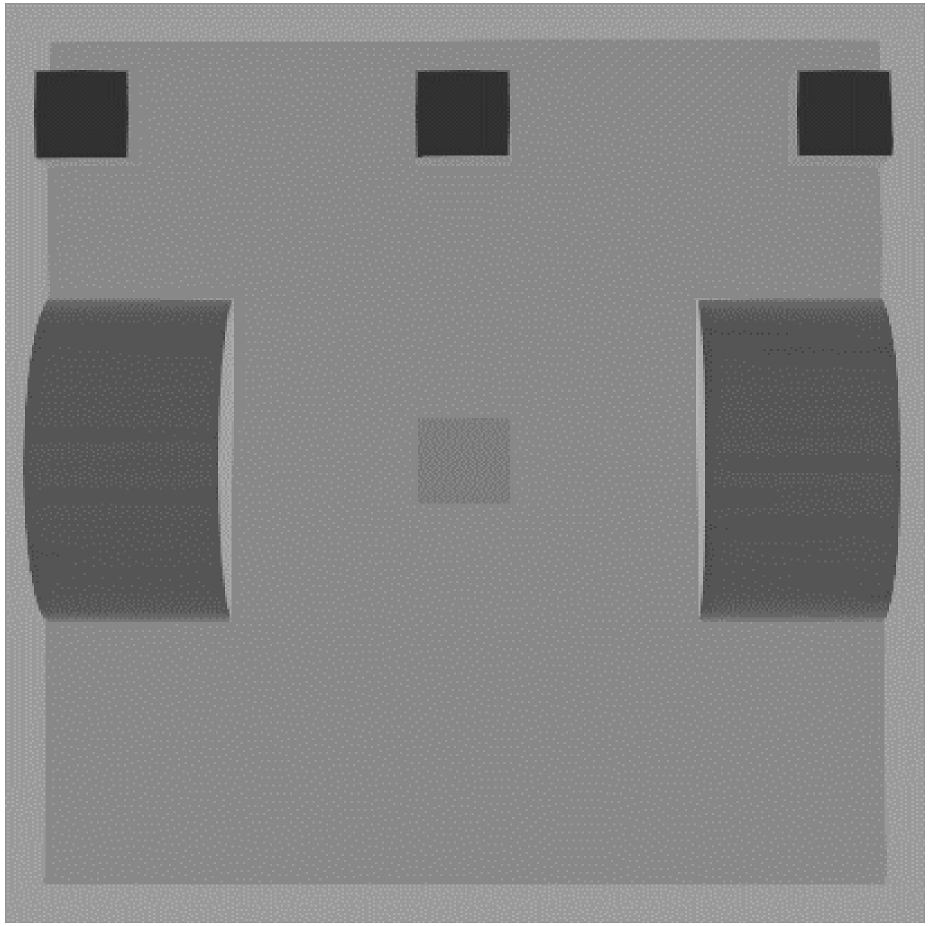
Methodology (Data Structures)

- 1-Dimensional Array
  - Used to store the information of the cells
  - Each cell is represented by a unique number called the “Cell ID”
- Priority Queue
  - Used as open list in D*-Lite algorithm
  - Cells of the maze are stored with the help of the key values
- Movement Queue
  - The paths to the goal are stored
  - Start cell is stored at first ,then the successor cells are stored

Result
(Top View of 3D Maze)



Result
(Simulated 3D View of Robot)



Top view of the robot

Side view of the robot

Result
(PID Tuning Procedure)

- The general equation for feedback in a PID controller is given by

$$\beta = k_p \times e + k_d . \times (pE - e) + k_i \times tE$$

k_p= Proportionality gain	e = Present error
k_d= Derivative gain	pE = Past Error
k_i= Integral gain	tE = Total Error

- k_p = 0.05, k_d = 0.0001 and k_i= 0.00001 are obtained from PID tuning

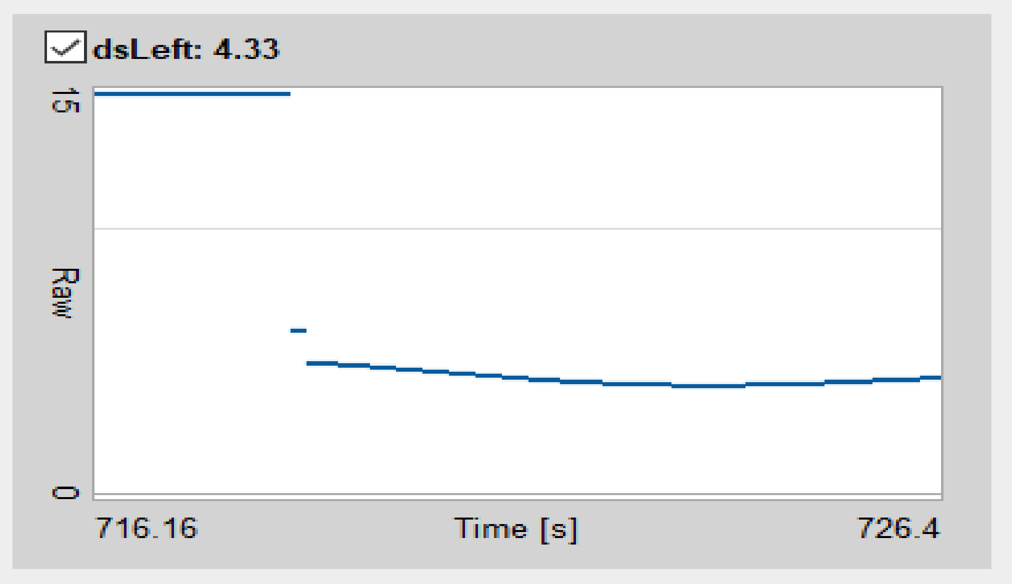
Result (Effect from PID Parameters)

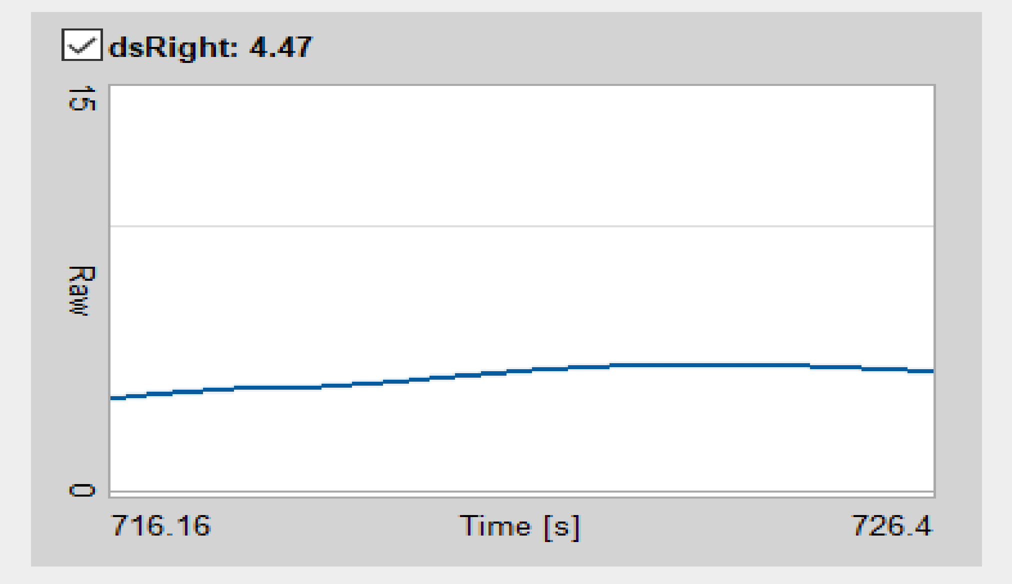
- k_d and k_i cannot be ignored in the feedback even though they are small
  - slightest change in velocity causes a drastic change in orientation of the robot

Parameter	Rise Time	Overshoot	Settling Time	Stability
$K_p \uparrow$	Decrease	Increase	Small change	Degrade
$K_d \uparrow$	Decrease	Decrease	Decrease	Improve
$K_i \uparrow$	Decrease	Increase	Increase	Degrade

Result
(Error Mitigation by PID controller) – [1]

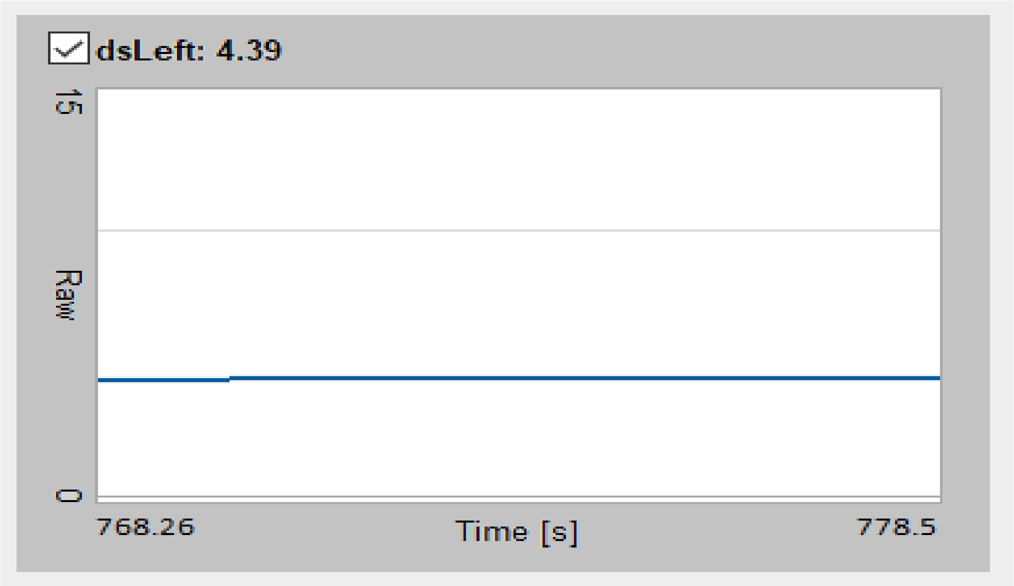
- Error is taken from the difference of left and right distance sensors
- Represents the proximity of the robot to a wall
- The error is maximum while moving forward after a turn

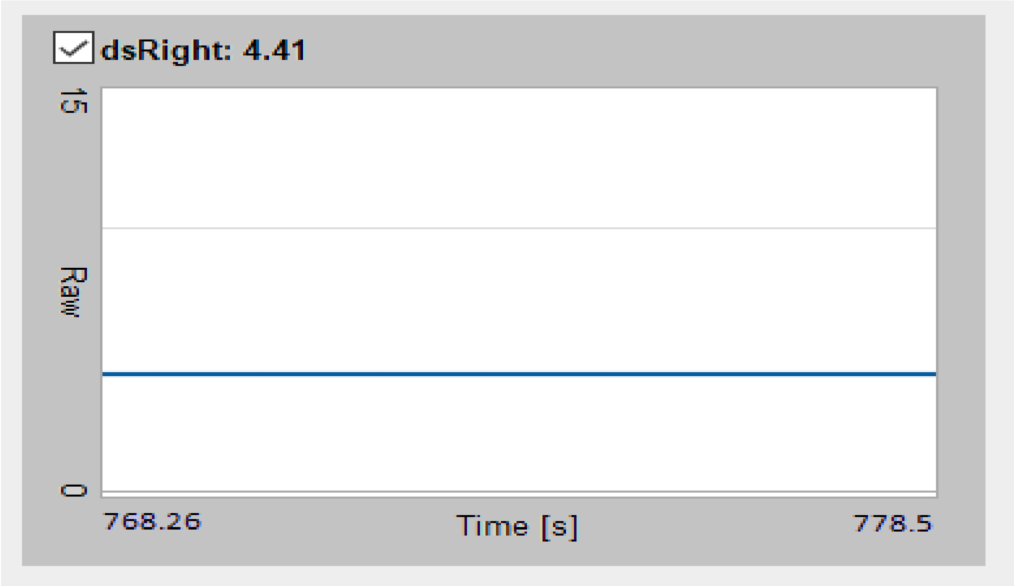




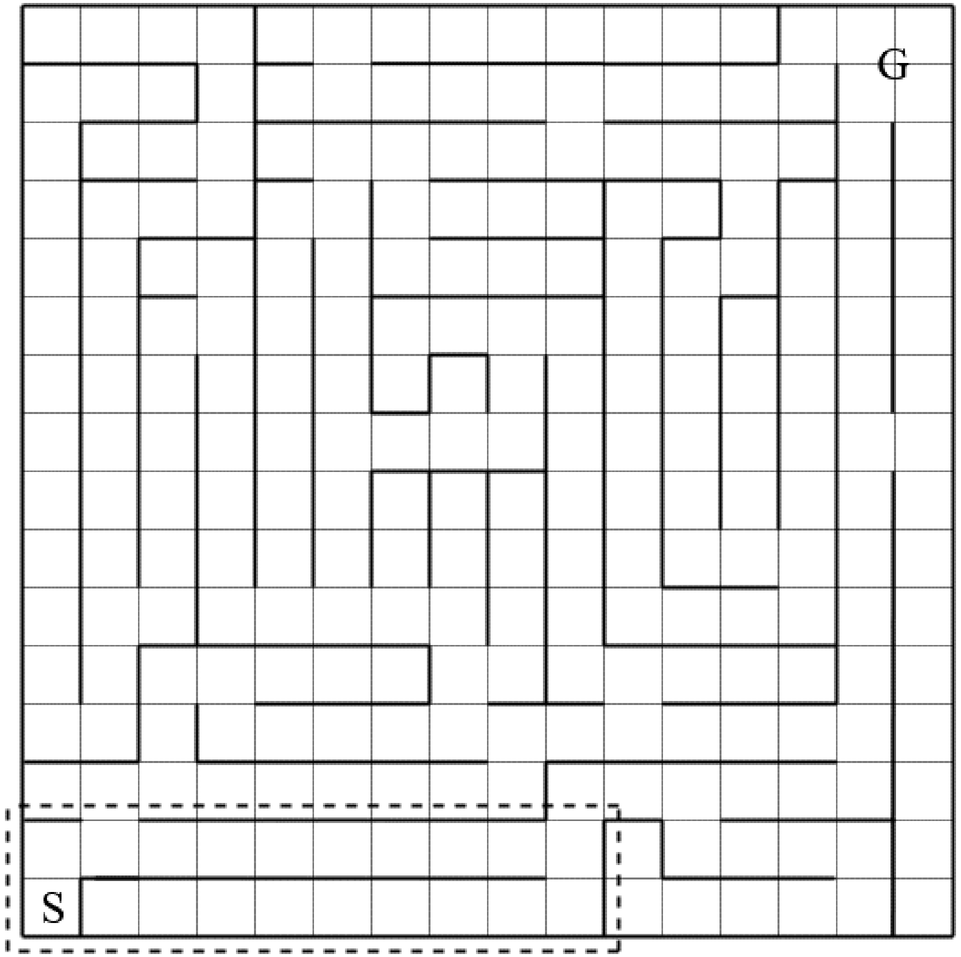
Result
(Error Mitigation by PID controller) – [2]

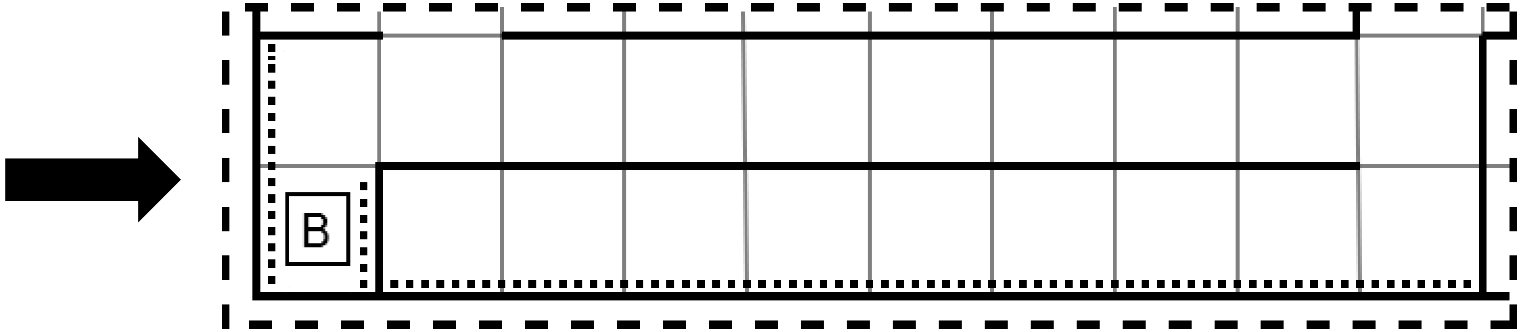
- Feedback mitigates the error and keeps the robot in center of track
- The error is small however, it is not exactly zero





Analysis and Discussion (Dead-End Exclusion Process) – [1]

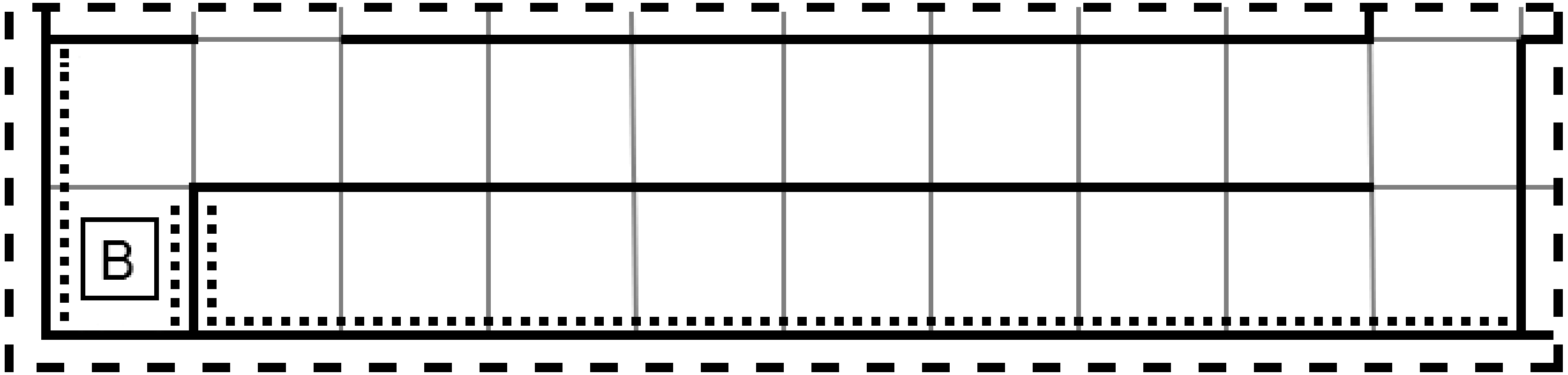




Analysis and Discussion (Dead-End Exclusion Process) – [2]

- The figure shows the initialization process

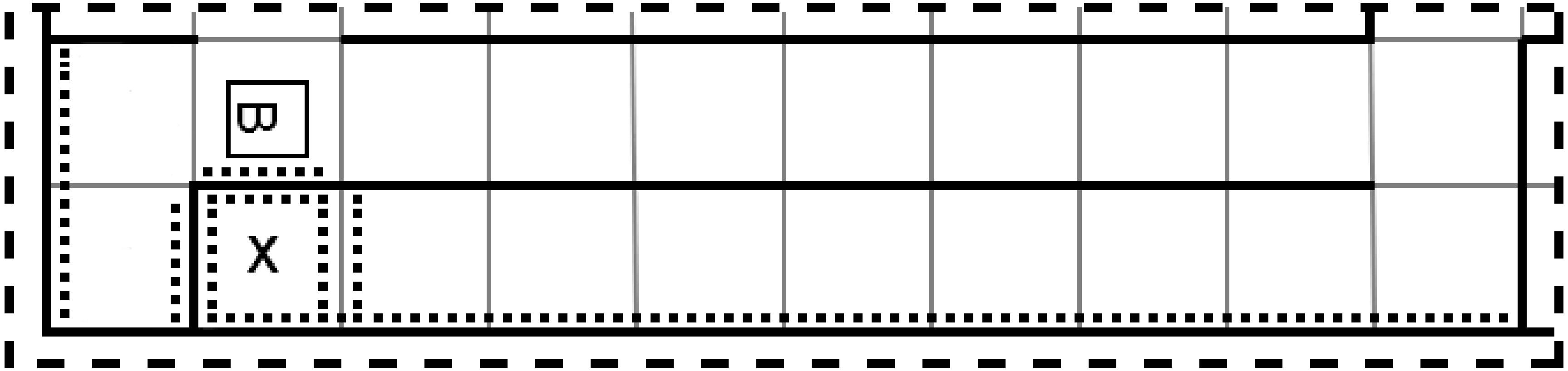
- Robot takes the sensor value and updates its current and neighboring cells



Analysis and Discussion
(Dead-End Exclusion Process) – [3]

- The robot moves forward, then updates current and neighboring cells

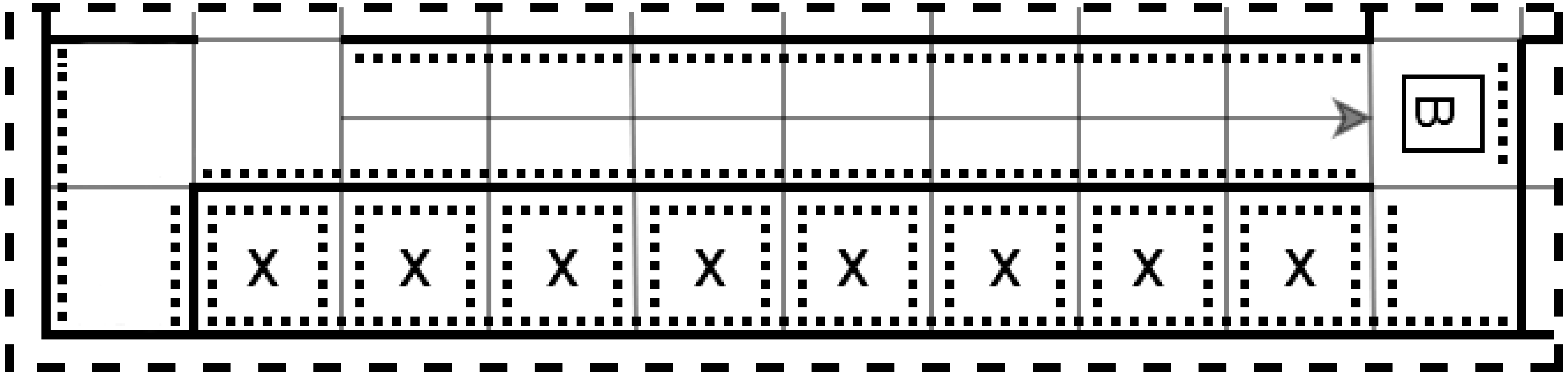
- The neighboring cell is marked in three directions and so it is closed



Analysis and Discussion (Dead-End Exclusion Process) – [4]

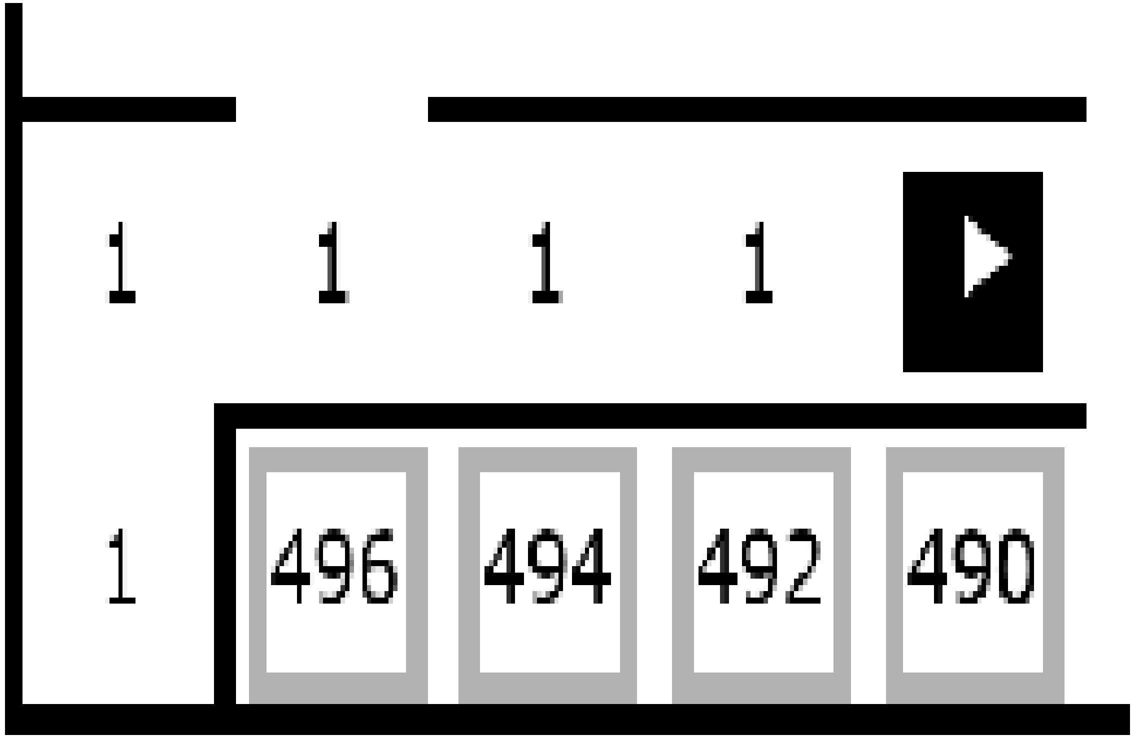
- While the bot moves in the second row, it marks the adjacent cells in the first row

- The first row satisfies the closing conditions and so the cells are closed without visiting them



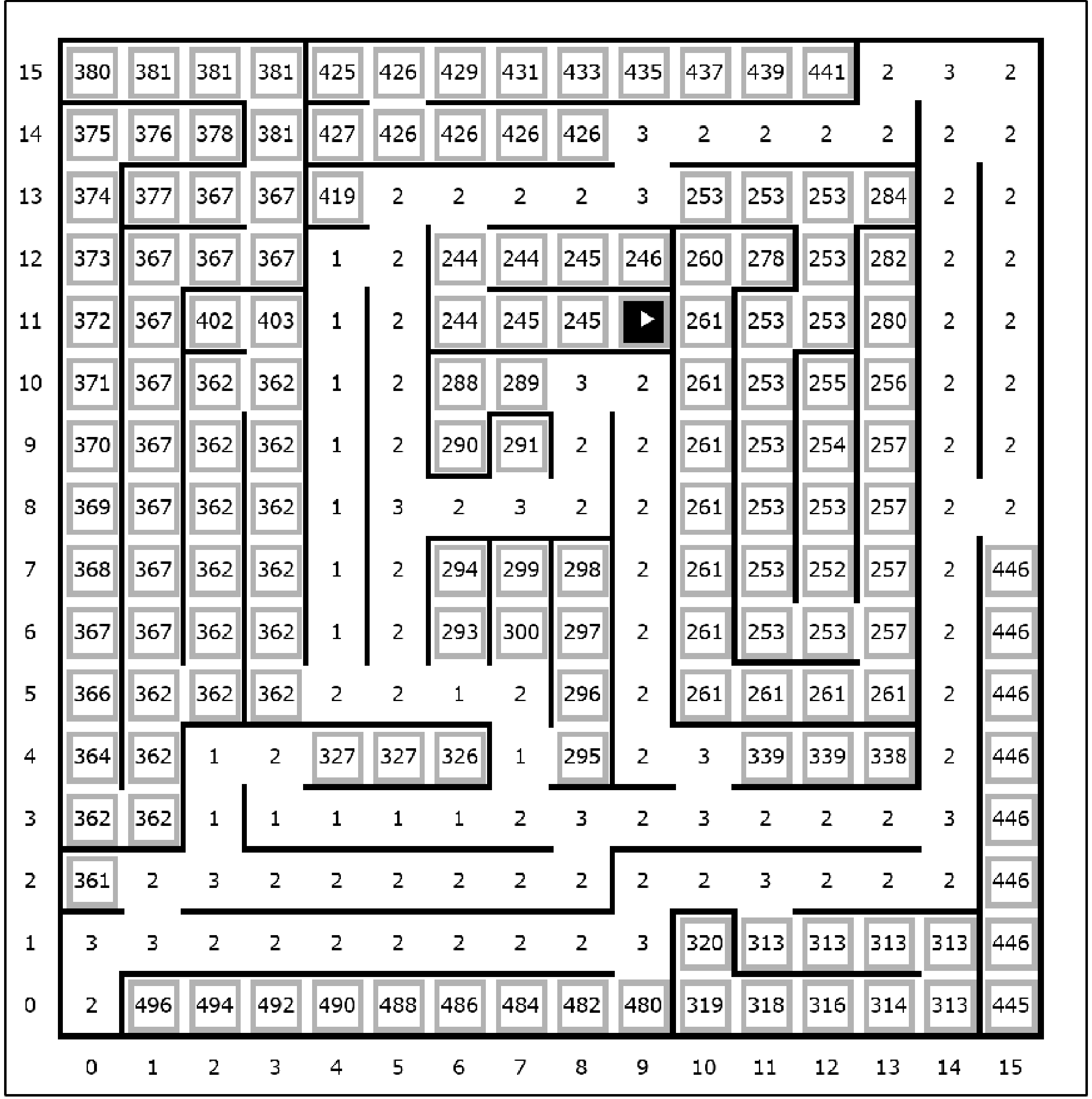
Analysis and Discussion (Dead-End Exclusion Process) – [5]

- Number in unclosed cells represent the visit count
- Closed cells are given a varying cost
- The cost of closed cell is provided by subtracting the number of unique cells visited from 500



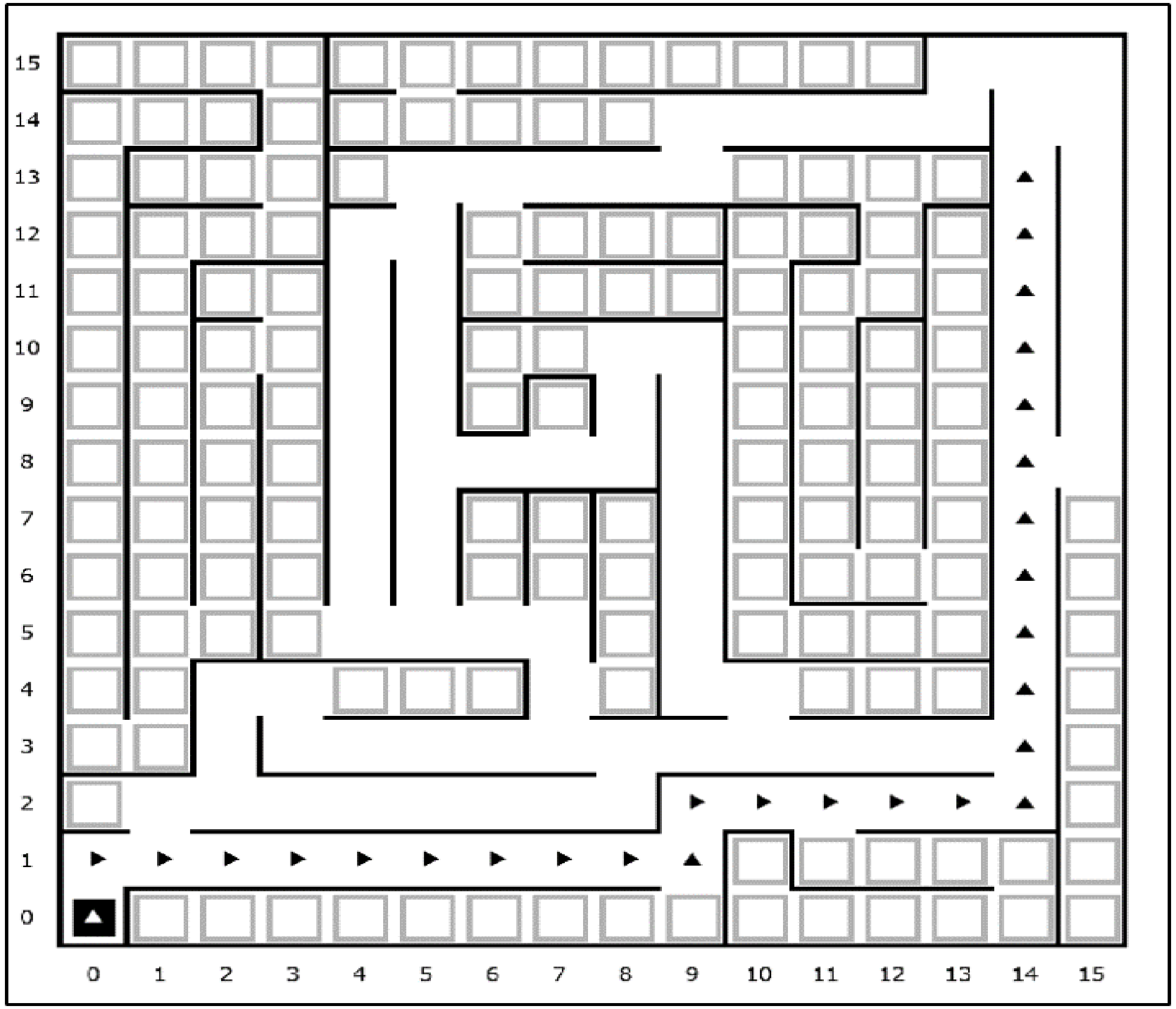
Analysis and Discussion (Dead-End Exclusion Process) – [6]

- Cells with box lead towards dead-end
- Path which leads to goal are formed from the box-less cells
- Numbers represents the cost value of the cell
- Cells with lower cost are given higher priority by the robot while moving
- The information of the whole map is stored in memory



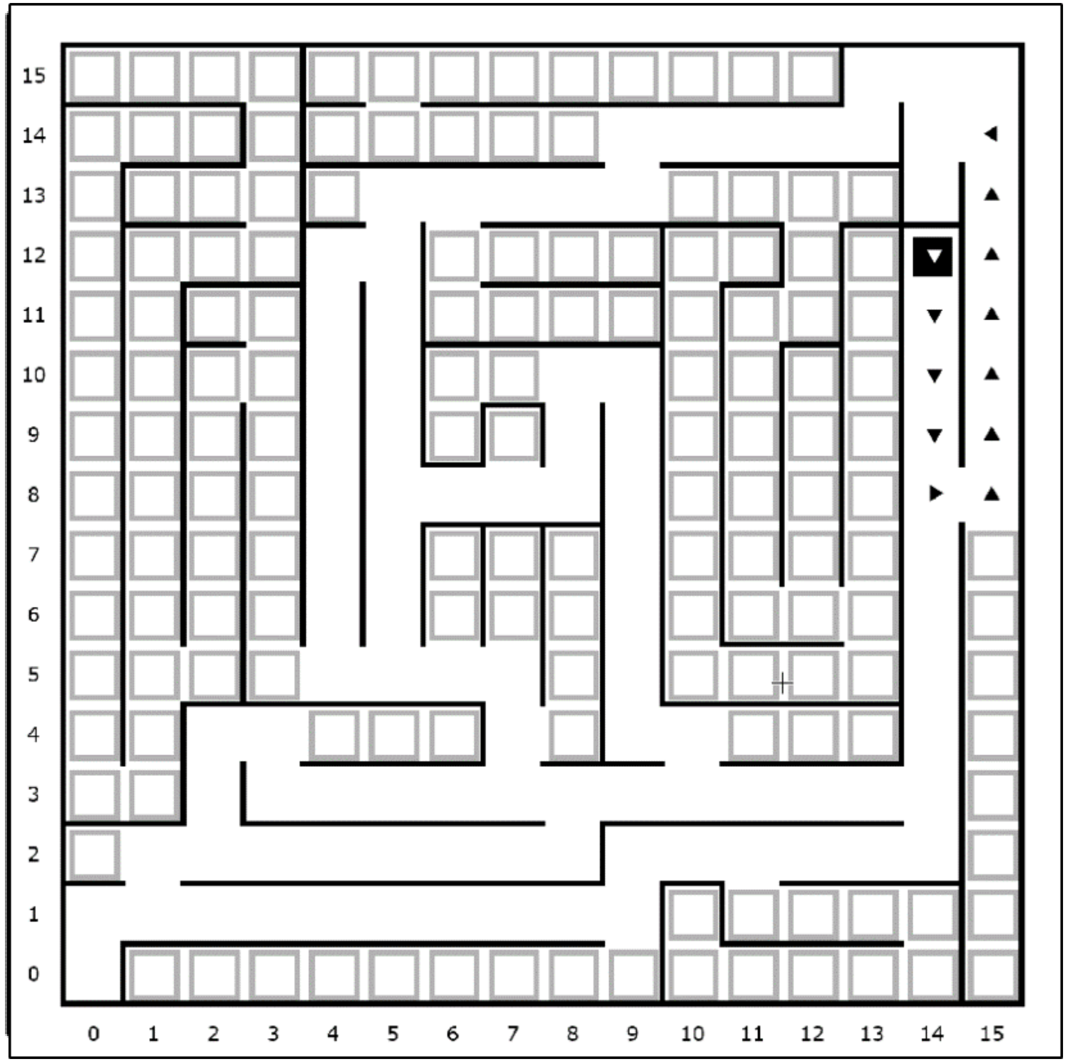
Analysis and Discussion
(D*-Lite Process) – [1]

- Bottom left corner is starting position
- Top right corner is goal position
- In a maze without obstacles, the robot follows the shortest path to the goal



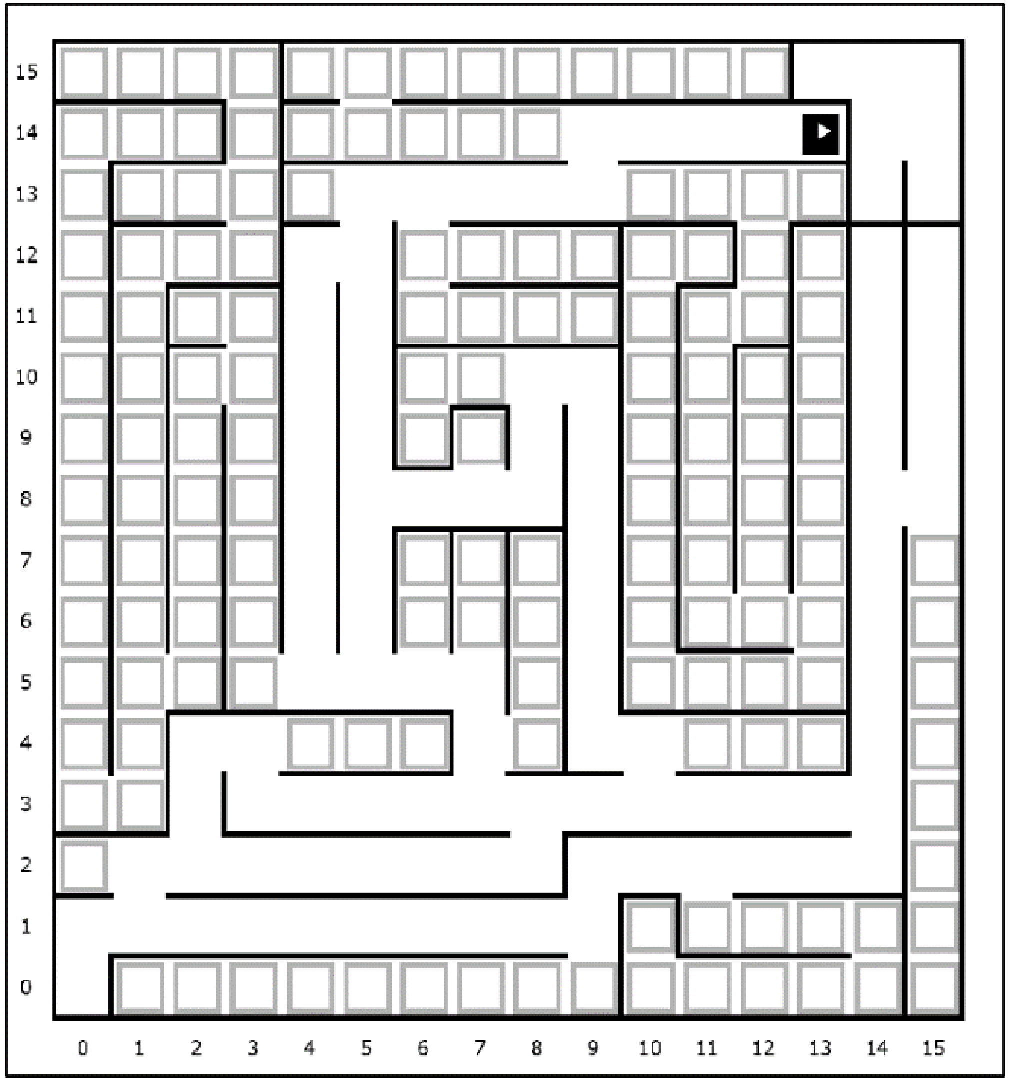
Analysis and Discussion
(D*-Lite Process) – [2]

- Maze with obstacles and alternative routes:
  - Robot discovers the obstacles as a wall in its path in between the cells (14,12) and (14,13)
  - Bot calculates the next shortest path that leads to the goal



Analysis and Discussion (D*-Lite Process) – [3]

- Maze with obstacles and no alternative route:
  - Robot discovers the obstacles in its path
  - The robot keeps moving until remaining obstacles are found in its path
  - After all obstacles in its path have been exhausted, the robot stops



Future Enhancements

- For more efficient movement, the chassis could be made circular
- Robot could be made to move diagonally to access 8 cells at a time
- By using an aerial camera, the removal as well as addition of real-time obstacles can be relayed to the robot
- For goal-seeking, sampling-based algorithms like Rapidly Exploring Random Trees (RRT) and RRT* can be used

Conclusion

- In the designing phase, the robot and maze were simulated in a virtual environment
- Dead-End exclusion algorithm was used for maze exploration
- Goal seeking was executed by using D*-Lite algorithm
- Obstacles that were added to the maze were detected and avoided in real-time
- All of the project objectives were successfully completed

Project Schedule (Gantt Chart)

Project Start Date	Sep,2020	Project End Date	Feb,2021

Task	Progress (%)	Sep	Oct	Nov	Dec	Jan	Feb
Brain Storming and Topic Discussion	100						
Documentation	100						
Follow Webots Tutorial	100						
Learn Maze-Solving Robots in Webots	100						
Learn Dead-End Exclusion Algorithm	100						
Learn D*-Lite Algorithm	100						
Maze 3D Structure Design	100						
Robot Structure Design	100						
Program Robot Control	100						
Program Dead End Exclusion Algorithm	100						
Test and Debug Dead End Exclusion Algorithm	100						
Program D*-Lite Algorithm	100						
Test and Debug D*-Lite Algorithm	100						

References – [1]

X. Li, X. Jia, X. Xu, J. Xiao and H. Li, "An improved algorithm of the exploring process in Micromouse Competition," in School of Automation Science and Electrical Engineering, Beijing, 2010.

A. T. Le and T. T. Le, "Search-Based Planning and Replanning in Robotics and Autonomous Systems," in *Advanced Path Planning for Mobile Entities*, IntechOpen, 2017, pp. 63-89.

A. Stentz, "Optimal and Efficient Path Planning for Partially-Known Environments," in *IEEE International Conference on Robotics and Automation*, San Diego, 1994.

References – [2]

S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research,* vol. 30, no. 846, pp. 846-894, 2011.

K. R. Kozlowski, "RRT-path – A Guided Rapidly Exploring Random Tree," in *Robot Motion and Control*, London, Springer-Verlag London, 2009, pp. 307-316.