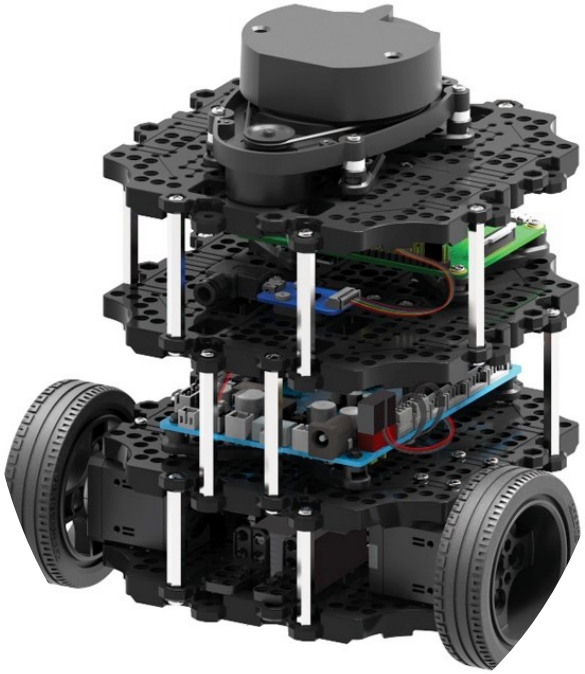


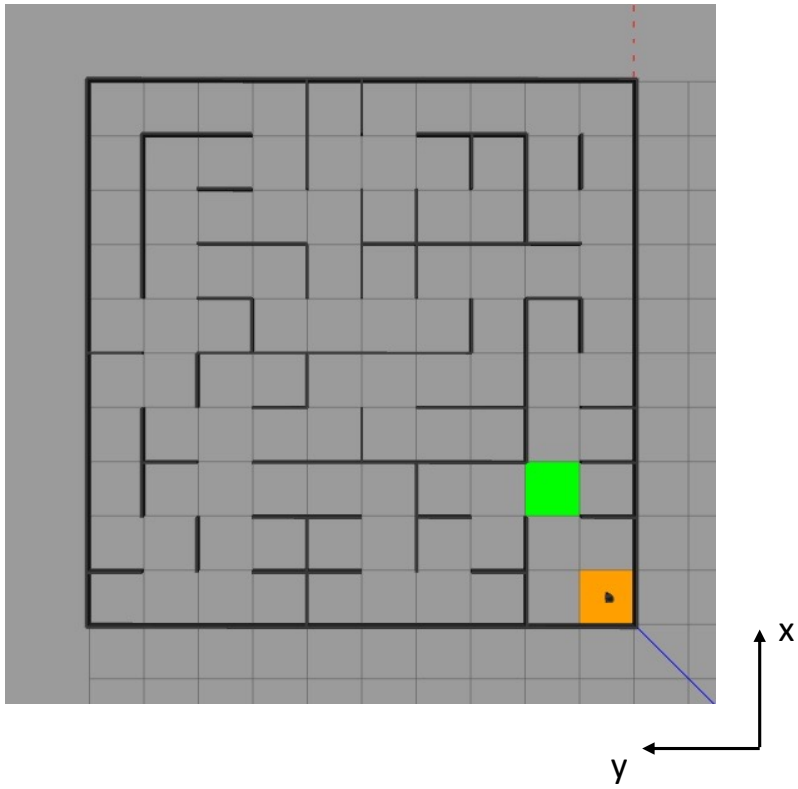
EE3305 / ME3243 Robotic System Design

Description of Project 2
Path Planning and Navigation in ROS
Platform: Turtlebot3 Burger

A/Prof. Prahlad Vadakkepat
Dr. Andi Sudjana Putra

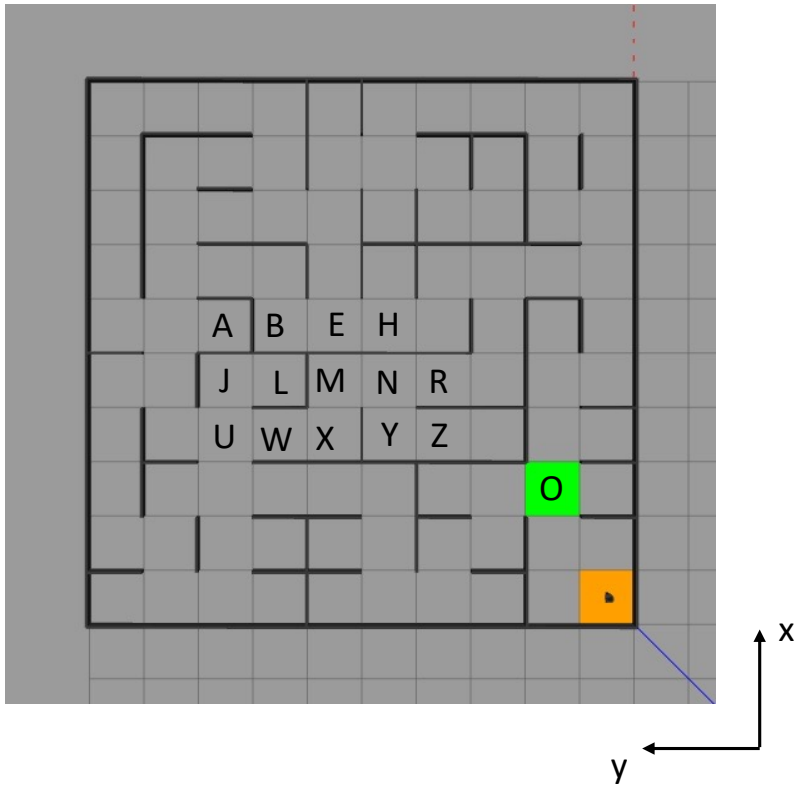
Project 2

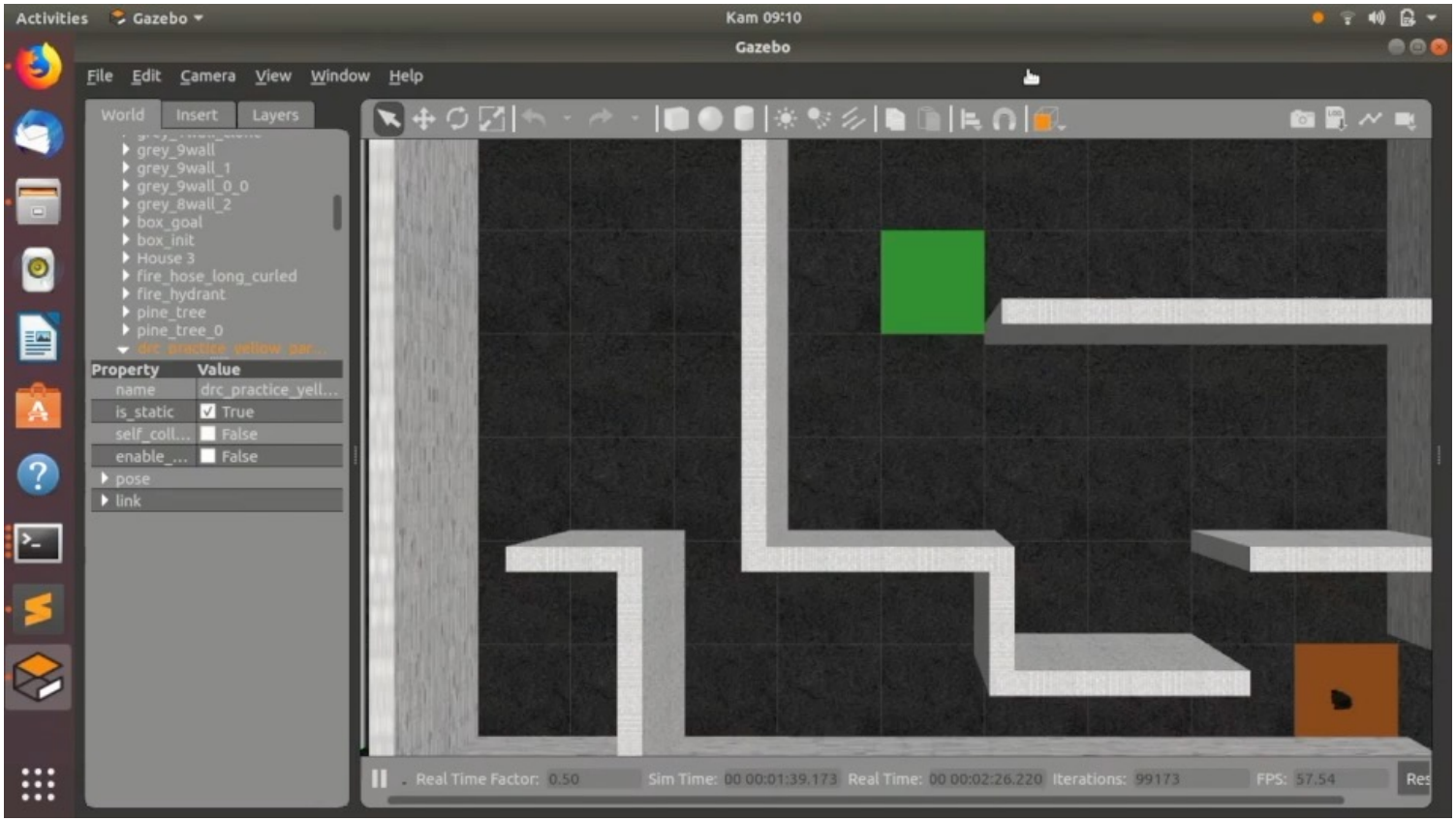
- You will be simulating a Turtlebot3 maneuvering a maze.
- Turtlebot3 will start from an initial cell (orange) and will move to a destination cell (green).
- Information about Turtlebot3: https://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/ **(how to use this?)**



Project 2

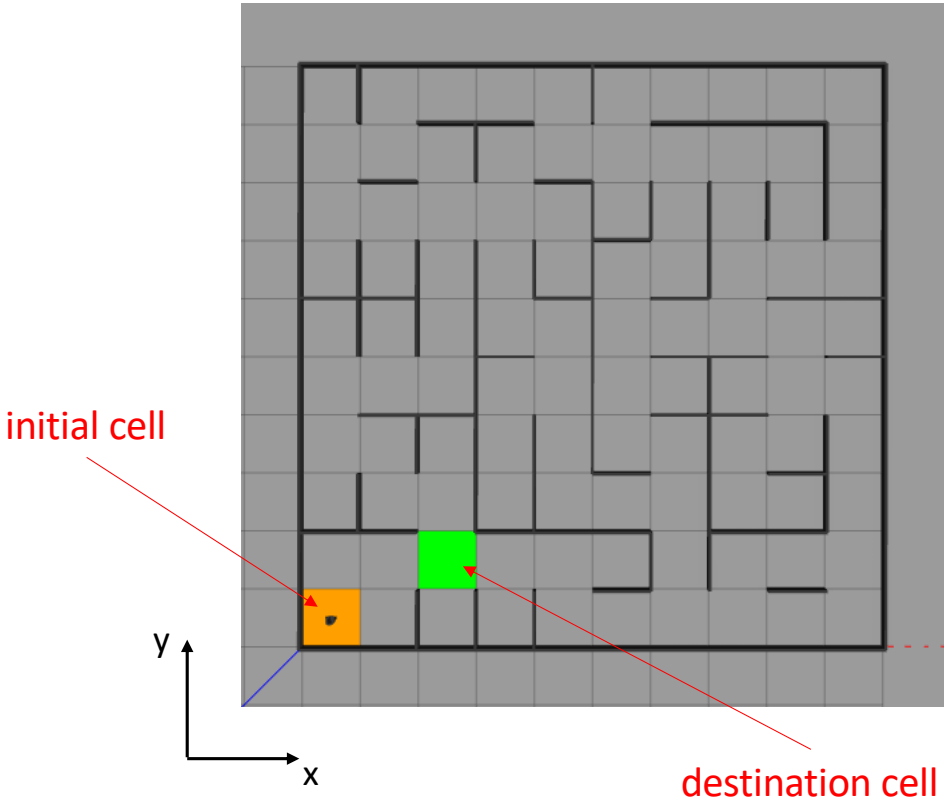
- You will be simulating a Turtlebot3 maneuvering a maze.
- Turtlebot3 will start from an initial cell (orange) and will move to a destination cell (green).
- Information about Turtlebot3: https://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/ **(how to use this?)**




Activities
Gazebo
Kam 09:10
Gazebo
File Edit Camera View Window Help
World Insert Layers
grey_9wall
grey_9wall_1
grey_9wall_0_0
grey_8wall_2
box_goal
box_init
House 3
fire_hose_long_curled
fire_hydrant
pine_tree
pine_tree_0
Property Value
name drc_practice_yell...
is_static True
self_coll... False
enable_... False
pose
link
Real Time Factor: 0.50
Sim Time: 00 00:01:39.173
Real Time: 00 00:02:26.220
Iterations: 99173
FPS: 57.54

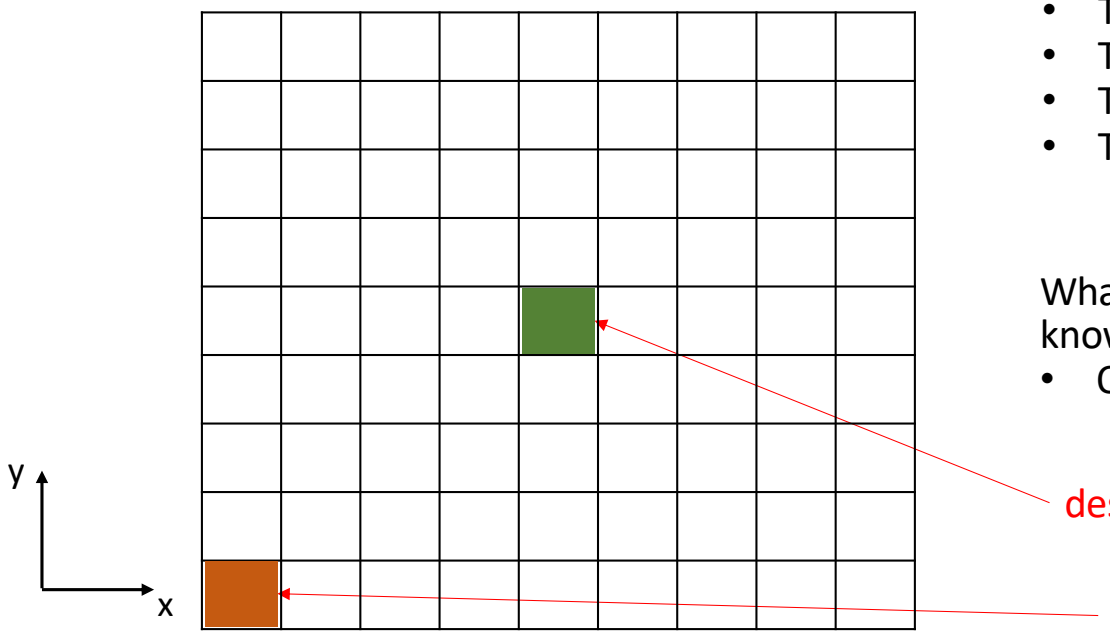
Navigate in a Maze

- Design a **navigation system** for a mobile robot
- Autonomously navigate from start to goal
- Environment is **not fully known**
- Easy task for a human, but complex for a robot, **hence the need to divide into sub-problems**



What is known and not known

The map initially known by turtlebot3 (no obstacles)



At first the robot knows the following information:

- The size of the maze
- The size of each cell
- The initial cell
- The destination cell

What does the robot **not** know?

- Obstacles (e.g., walls)

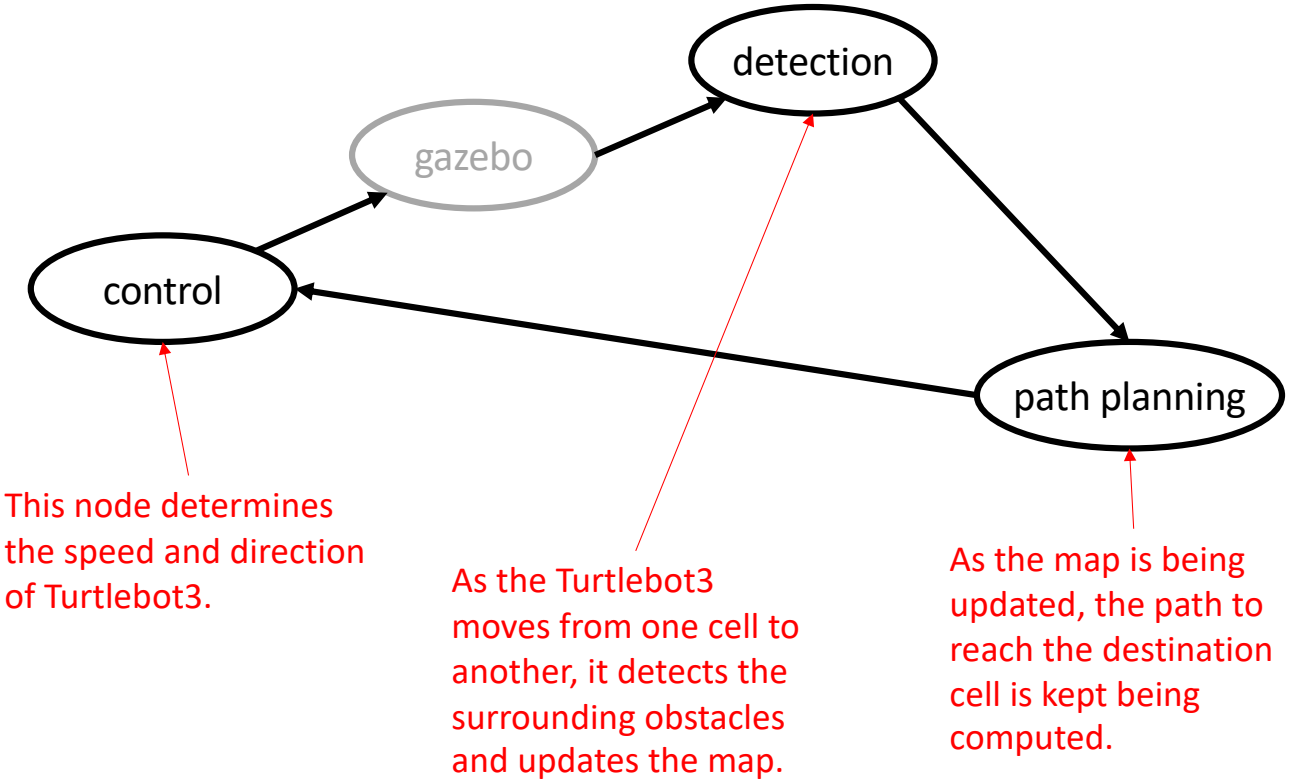
What does the turtlebot need to do?

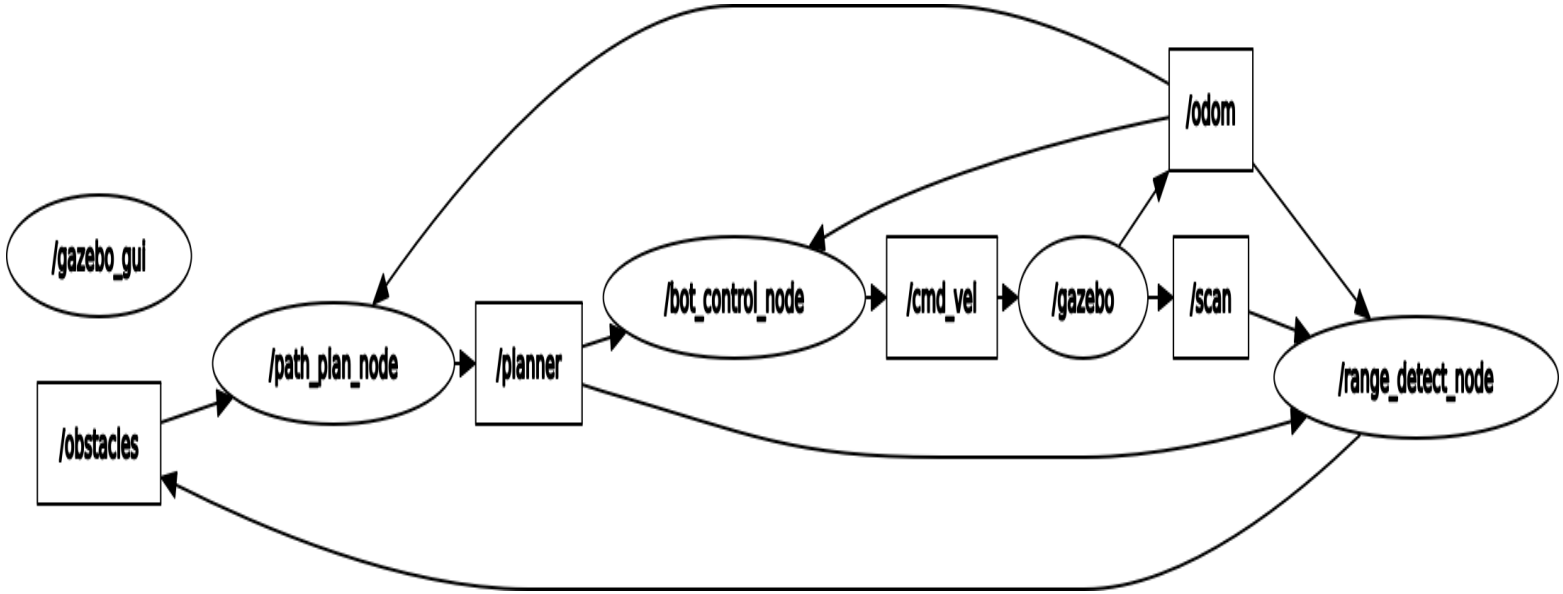
It needs to detects obstacle on its path to the destination cell.

It needs to plan (and re-plan) how to get to the destination cell.

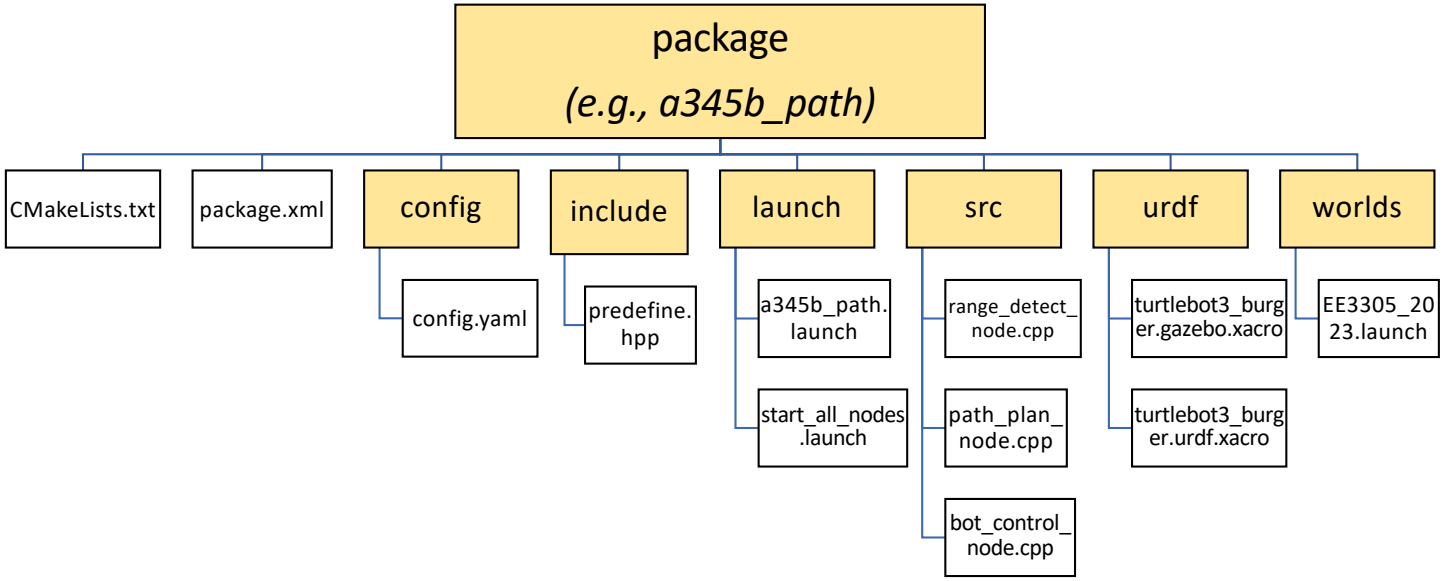
It needs to move to the destination cell along the the path.

Three Nodes (without Topics)



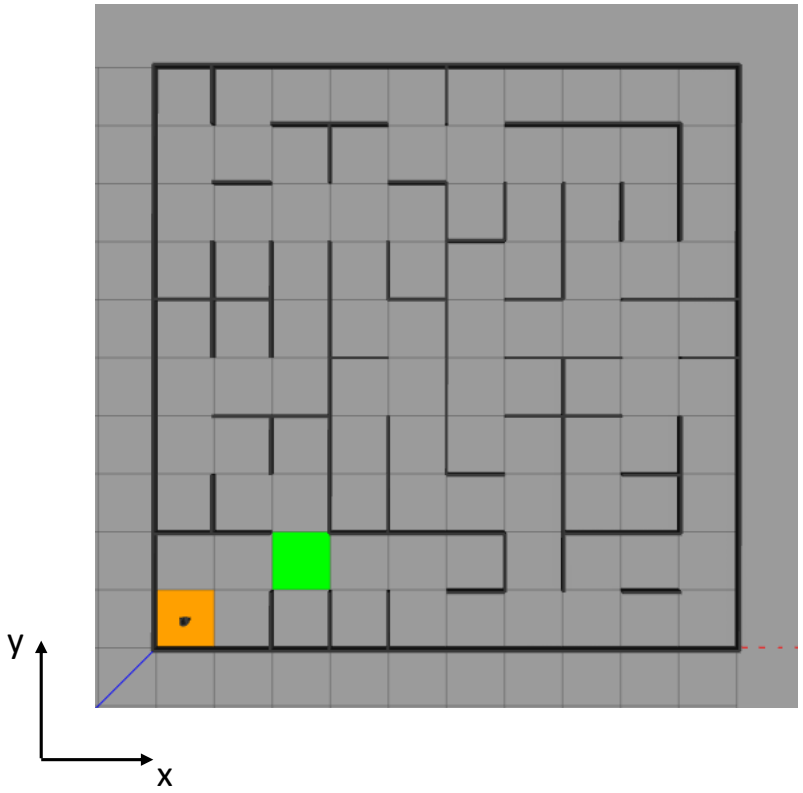

/gazebo_gui
/obstacles
/path_plan_node
/planner
/bot_control_node
/cmd_vel
/gazebo
/scan
/odom
/range_detect_node

File structure (example: a345b_path)

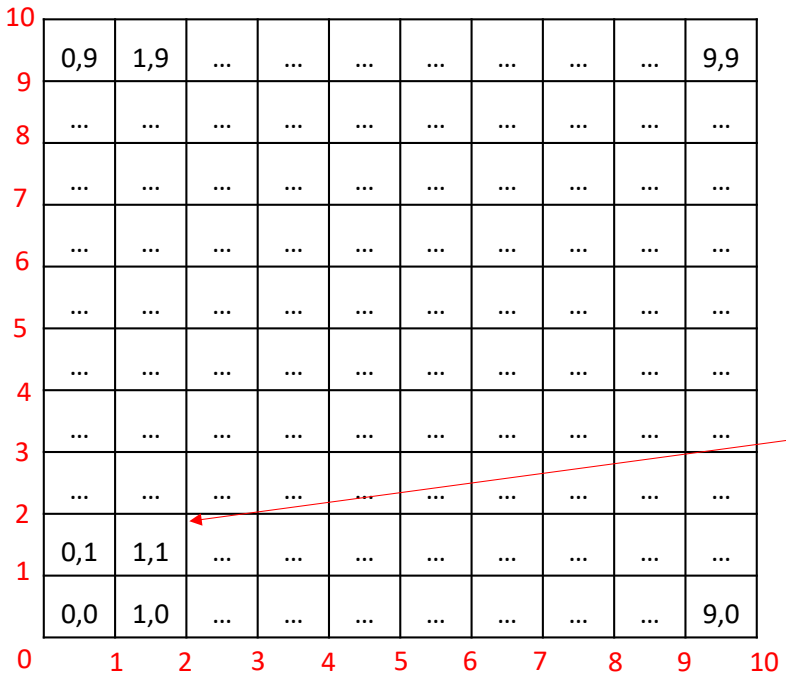


Setup the maze

- Import the maze file (EE3305_2022.world).
  - It defines the wall.
  - It shows the initial and destination cells.
- Launch the maze and the Turtlebot3.



Setup the maze



Adjust the destination cell according to the one assigned to you.

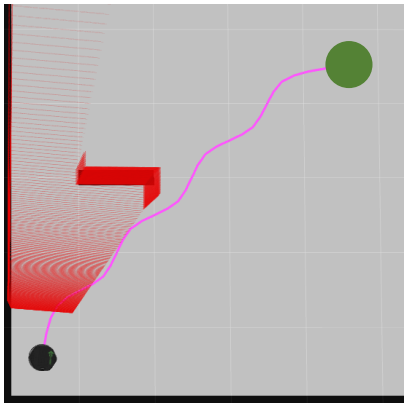
This cell is X=1, Y=1.

In test_world_1.world, the cell is defined referring to the centre of the square, i.e., <1.5 1.5>.

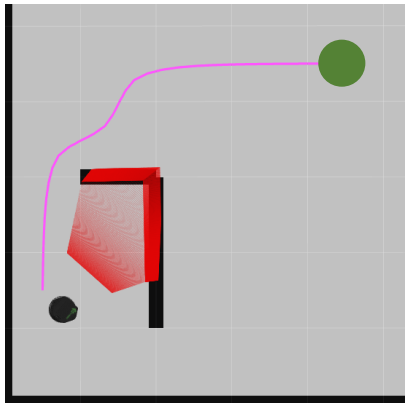
Navigate in a Maze

As the robot is moving, it detects walls in certain location.

(1)

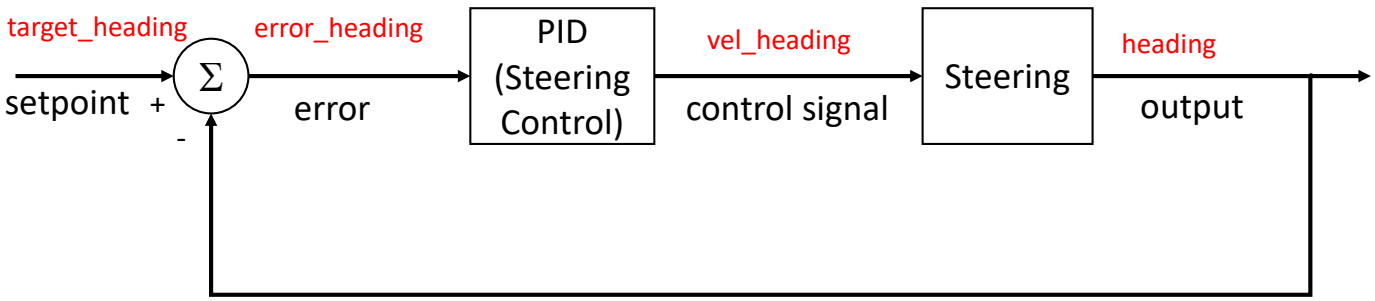


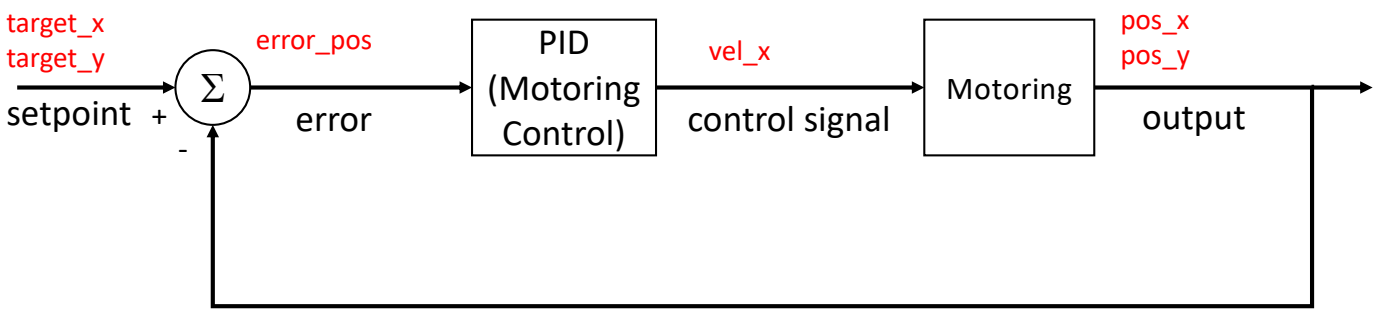
(2)



Control Structure and Its Variables

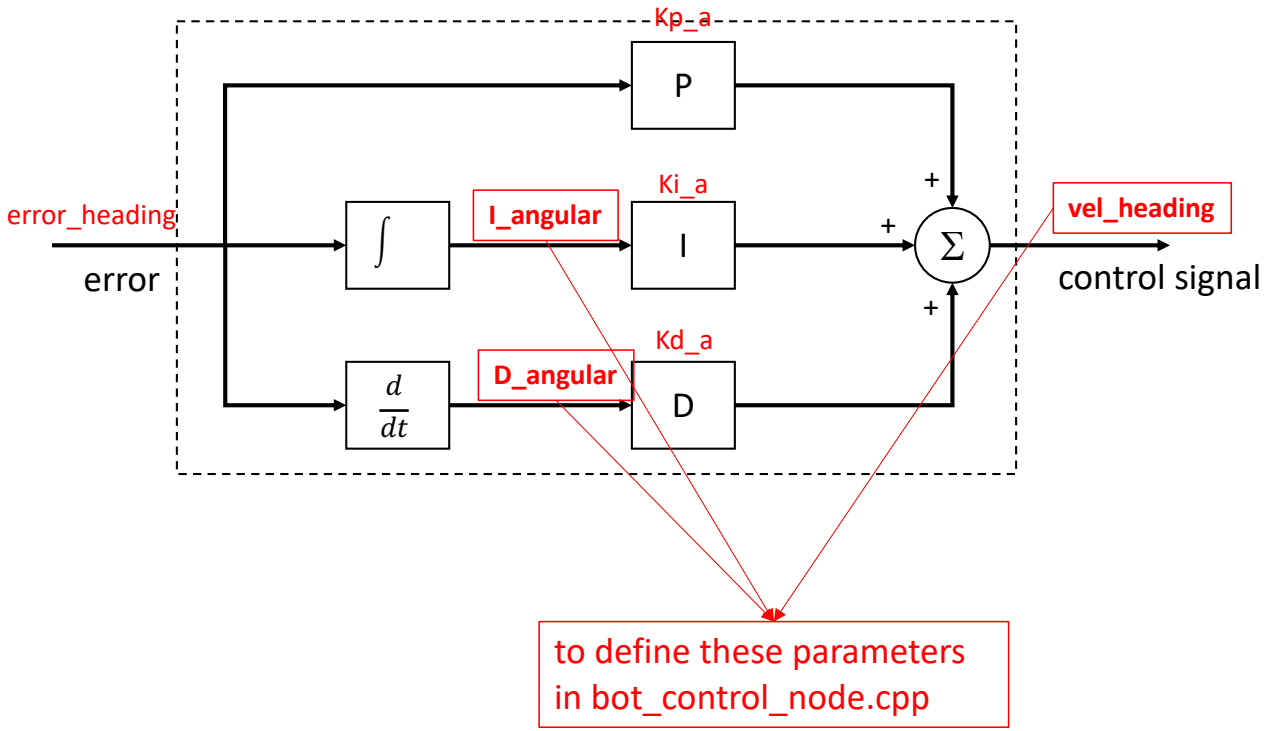
(Project 2)





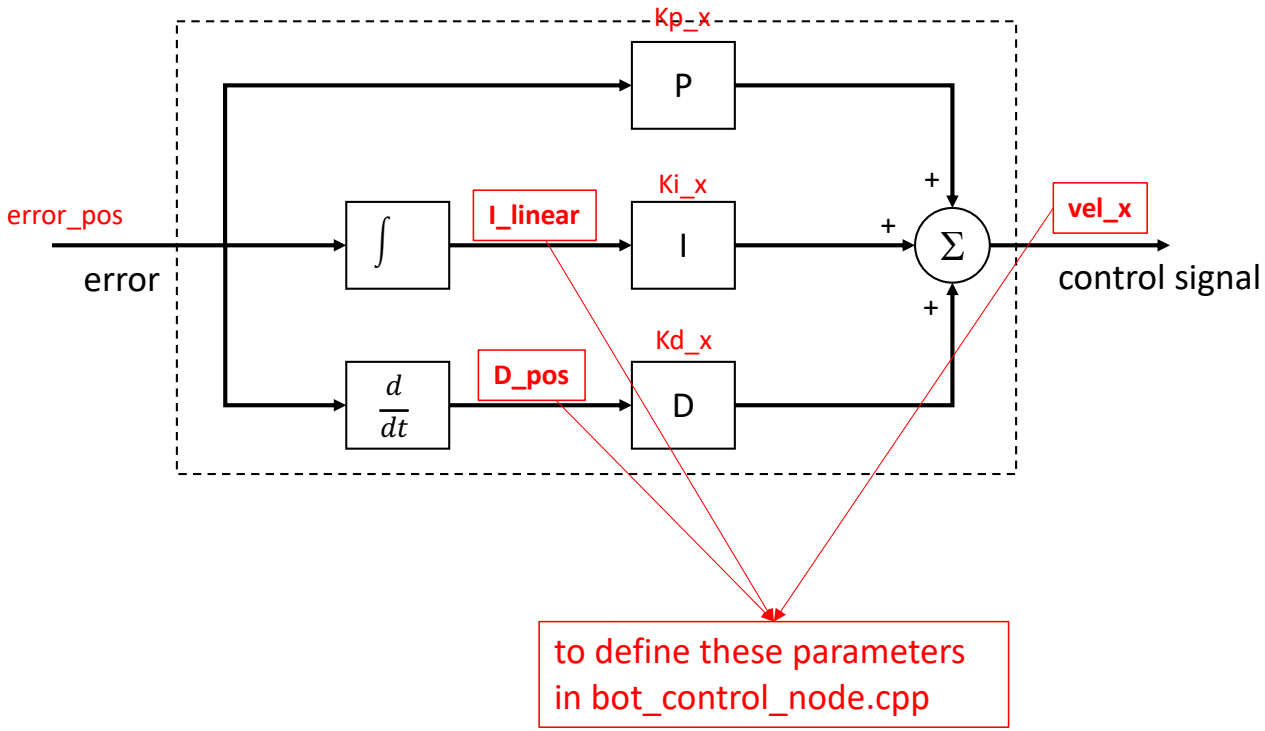
Steering Control

(Project 2)



Motoring Control

(Project 2)



Parameters to setup

(Project 2)

- Obtained from public variables in BotControl.hpp
- Defined in config.yaml

Kp_x:
Ki_x:
Kd_x:
Kp_a:
Ki_a:
Kd_a:

to tune these parameters

dt: 0.1 (assume this is the computation rate)
goal_x:
goal_y:

Location of Destination Cell

In EE3305_2023.world

- <pose>1.5 1.5 0 0 0 0</pose>

In predefine.hpp

- GOAL_X=1
- GOAL_Y=1

In config.yaml

- goal_x
- goal_y

Files

(refer to the 3 nodes)

Detection

- range_detect_node.cpp

Path planning

- path_plan_node.cpp

Control

- bot_control_node.cpp

CMakeLists.txt

- find_package
- catkin_package
- include
- add_executable
  - range_detect_node
  - src/range_detect_node.cpp
- target_link_libraries
  - range_detect_node

roscpp, sensor_msgs, nav_msgs, geometry_msgs, std_msgs

form one detection node

CMakeLists.txt

- add_executable
  - path_plan_node
  - src/path_plan_node.cpp
- target_link_libraries
  - path_plan_node
- add_executable
  - bot_control_node
  - src/bot_control_node.cpp
- target_link_libraries
  - bot_control_node

form one path planning node

form one PID control node

package.xml

- <buildtool_depend>catkin</buildtool_depend>
- <build_depend>…</build_depend>
- <build_export_depend>…</build_export_depend>
- <exec_depend>…</exec_depend>

roscpp, sensor_msgs,
nav_msgs, geometry_msgs,
std_msgs

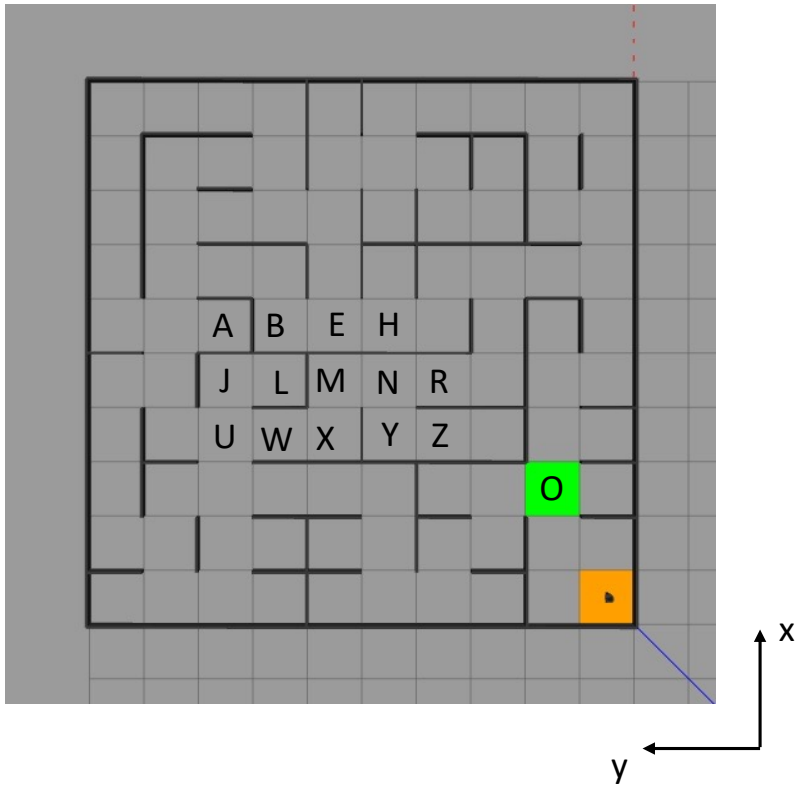
To launch the simulation

- Build the package
- Source the workspace
- Launch the environment (one terminal)
  - # In ~/*workspace/*
  - $ roslaunch *package launch_file.launch*
- Launch the simulation (another terminal)
  - # In ~/*workspace*
  - $ roslaunch *package launch_file.launch*
  - (different launch file from above)
- If at any time permission is denied to run a file:
  - $ chmod +x *file_name*

Report

1. Initial conditions. Show the initial and assigned destination cells in the maze.
2. Path planning.
   a) Discuss how the path planning works, in particular the lines titled ”Poll the queue” and ”Find the path if the goal is found and break the loop”,
   b) Identify: (1) the path planned at the start cell, (2) the path planned at cell O, and (3) the eventual path. Discuss why they are the same or different.
3. Navigation. Discuss how the PID control is implemented, including:
   a) The code that you have added,
   b) The control gains.
4. Performance enhancement. Discuss what you planned to improve, how you did it and the results.
5. ROS structure. Include the graph of ROS nodes and topics of the simulation. Identify and describe the 3 main nodes and topics pertaining to those nodes.

The Maze



Explore ROS (command $ rqt_graph)

- Select “Nodes/Topics (active)”.
- Describe the nodes and topics related to the 3 main nodes of the simulation.
  - Identify the nodes and topics.
  - Show the relevant nodes and topics.

Thank you.