Model-based Design of Cyber-physical Systems 2024 MazeTurtle Project

Deadline: 18:00, December 9, 2024

Robot Model MazeTurtle

Your task is to develop the program for MazeTurtle. MazeTurtle is a robot designed to explore ventilation systems. Your team has been tasked to map out the first floor ventilation system in an abandoned building. The data that needs to be mapped are the walls.

Development Environment

The software must be developed using *Itemis CREATE*. The *TurtleBot3 Burger* will be provided and must not be changed.

You should only use Itemis CREATE for the basic implementation of the MazeTurtle. Python code should only be used after approval from a teaching assistant. However, it is not necessary to use additional Python code to complete the project.

Input and Output

Information about the input and output can be found in the SCT_interface_manual.

Sensor	Data
Lidar	Distance, Intensity
Imu	Roll, Pitch & Yaw, odometry
Motor encoders	Odometry, speed and rotation

Assignment

You are tasked to map out a ventilation system in an abandoned building. A map of the ventilation system is needed, as this was not recorded in the building plans. The building might collapse and is thus deemed unsafe for people. However, the MazeTurtle (TurtleBot3 Burger) can go inside. The architect that designed the building recovered remnants of the building plans with the following notes on the ventilation system:

1. The system was built in a 4x4 grid shape.
2. The system was built out of 48 by 48 cm metal plates with walls between some plates to guide airflow.
3. The entrance of the system is at (0, 0), which is at the north-west corner of the system.

Your assignment is the following:

1. Manually drive the robot to the centre of the first cell, making sure it is parallel to the wall and facing south. Then, use this position to calibrate the robot (set zero).
2. Drive through the maze, without hitting the walls, as this might cause a collapse.
3. Visit all cells of the maze and log them in the provided data structure.
4. Return to the start of the maze to collect the data from the robot.

To solve unexpected problems, it is possible to take manual control of the robot during its journey through the ventilation system. However, this connection will interfere with the ability of the MazeTurtle to log the system and the map will be of lower quality.

Components

Manual Control Component

Manual control uses keyboard input of the terminal that the program will be run in. The following keystrokes + enter will trigger events. These events need to cause the following actions:

Keystroke	SCT Event	Action
W	computer.w_press	Increase speed
A	computer.a_press	Increase rotational speed for turning left
S	computer.s_press	Set speed to zero
D	computer.d_press	Increase rotational speed for turning right
X	computer.x_press	Decrease speed (negative speed drives backwards)
M	computer.m_press	Switch between autonomous and manual control

Switching between autonomous and manual control must always be possible, even during autonomous driving. After returning to autonomous control, the MazeTurtle should recover its ability to traverse the maze.

Driving Component

The robot has to be able to traverse the maze without hitting the walls. Avoiding walls can be done using the **Lidar**. When the robot comes too close to a wall, make sure to move away from it.

The main difficulty with this component happens when the robot turns, or when the robot experiences drift. This might turn the TurtleBot3 towards the walls, so make sure to continuously realign with the wall.

Logic Component

Logic needs to be designed that explores the whole maze. One sure-fire way of exploring all cells in the grid is by always following the left wall. This means following the left wall from start to end, and from end back to start. However, feel free to solve it another way.

Logging Component

While the robot is traversing the maze, all walls should be logged in a grid-based map. This is done using the grid interface, where you determine the row and column the robot is in, and can retrieve or store data based on that row and column, as well as the cardinal orientation of the robot.

Make sure not to access cells outside of the given grid, as this results in errors in the data structure and might crash the program.

Report

Please prepare a short report for your MazeTurtle project. The goal of the report is to allow teaching assistants to understand the implementation. The report should be lightweight (approximately 2000 words), and contain at least the following information:

- Group name
- Name of all group members
- Readable images of the Statechart and each of its components
- A short description of each of the components' implementation

In addition, you can also provide further notes explaining your design decisions. If you have decided to implement a feature in certain way because a teaching assistant said so, mention it here.

Group Work

The groups should consist of 4 students, or 5 if necessary to ensure all students are in a group. You can form your own group as you want, but please consider that it is easier to schedule working sessions if all group members follow the same master studies. If you do not have a group, let us know and we will assign you to one. Once you have grouped up, register your group within Canvas.

Grading Criteria

Criteria	Points
Statechart Implementation	1.5
Statechart model compiles and executes on the TurtleBot. (i.e. Simulation in Gazebo)	0.5
Manual control is implemented.	0.5
Operating the Turtlebot can switch between Autonomous/Manual control.	0.5
Maze Solver implementation (Gazebo simulation)	5
The robot can successfully traverse the maze without hitting any wall. If a wall is hit one or more times, a penalty is given.	1.0 -0.5
The robot successfully visits all cells, and returns to the entrance. The score will be proportional to the percentage of total cells visited.	2.0
The robot successfully maps the maze and a visual representation (map) is presented. The score will be proportional to the percentage of correctly mapped walls.	1.0
After successful mapping the robot can be sent to a point in the maze without the use of LIDAR	1.0
Documentation	2
Clear explanation in the report.	1.0
Statechart model implementation with appropriate use of Statechart language features, such as hierarchy and orthogonal states	1.0
Physical TurtleBot	1.5
The robot visits a part of the maze without hitting the wall excessively	0.5
The robot visits most of the maze without hitting the wall excessively	0.5
The robot visits all of the maze and returns to the entrance without hitting the wall excessively	0.5