Robotics project work 2021

Siru Tamminen and Martin Kakko

Project description

Our project goal was to make a robot get through a maze by following the wall on its left side. This method of navigating the maze is not the most efficient or fast, but it is a sure way to find a way out, assuming there are no loops in the maze. Besides mazes, this kind of navigation could be used in other places where it is needed to thoroughly search an area for something. For example, the robot could go through a building, and find moisture in the walls. In big buildings this could be useful, since a person would not have to walk through every corner of every room with measuring equipment.

Technical description

We worked with Webots simulator to implement our project. In the simulation we started by building the maze from white wall blocks. The walls were placed so that there would be no loops, so that it was possible to get out of the maze. We looked at many robots in the process of choosing one for our project, but eventually chose to use E-Puck. This decision was made based on the placement of sensors and their precision. The E-Puck has 8 proximity sensors, which were best at sensing the maze walls from a short distance. The algorithm to control the robot was written with python.

Algorithm description

Our algorithm uses four out of eight proximity sensors on the robot. The four sensors in use are the two sensors on the front side of the robot and the two sensors on the left side. Before estimating the robot’s proximity from the wall, the sensor values are converted to be easier to compare. First a threshold value of 100 is subtracted from the sensor value, then the value is divided by 50 and rounded into two decimals. After that, the values can be used to determine which direction the robot should move. Before starting to navigate in the maze the robot will move forward until it finds the maze wall, and after that it starts following it.

The robot checks the sensor values on each timestep. If it detects that the wall is too close to it either on the left side or in front of it, it will slow down the right wheel to adjust its direction. The speed of the right wheel depends on the value received from the front sensor. If the robot detects that there is no wall on the left side anymore, it will slow down the left wheel to turn left, until it finds a wall to follow again.

Process

The original plan of this project was to make the robot turn exactly 90 degrees at every corner. However, we discovered during building the maze that it was not simple to make the walls have exact directions and corners. In retrospect our building strategy was not the best possible to begin with, and the webots environment would have allowed us to build a more precise maze. Since our maze walls were not placed directly, we did not program the robot to make exactly 90 degree turns either. Instead, we developed the robot to follow the wall as well as possible.

After choosing e-puck as our robot the project ran quite smoothly. We already had a general idea of how to make the wall following algorithm, so it was mostly just a matter of refining the code. The final algorithm we developed is decent and the robot does manage to find out of the maze, even though it does it slowly. It could have been made better by working a few more hours adjusting the turning of the robot, but it would probably not make the robot solve the mazes significantly faster.

Self-evaluation

This project offered us an interesting challenge, that we were curious to solve. Personally, we feel we fulfilled the goal of the project, and the finished product is adequate. Some small shortcuts were made in the process, like not using the 90 degree turns, but for two people with no former experience in robotics the result is still good. Our teamwork worked well. The workload was divided equally between us both during all parts of the process. We communicated mainly verbally, which made planning and reviewing the work easy and smooth. Our own grade for this project would be 4 out of 5.