Computer Project I – Final Report

1st Asger Song
Aarhus University
Deparment of electrical and computer engineering
Study number:
AU-id:

2nd Andreas Kaag Thomsen
Aarhus University
Department of electrical and computer engineering
Study number: 202105844
AU-id: au691667

***Abstract*—This is our abstract.**

I. Introduction

Overall, the project can be divided into four parts with certain aim and objectives:

1) Understanding the basic concepts of robot programming by working with an Arduino - that is, the *think-see-act cycle*. Reading and analyzing data from Range sensor and RGB sensor.
2) ROS Programming on Raspberry PI. Learning the structure that ROS provides and going through the beginner's tutorial.
3) Programming the robot to avoid obstacles. Then optimize the robot's performance both with respect to linear speed and collision avoidance.
4) Finalizing the code and testing the robot on an obstacle course.

II. Specifications

We have used the following equipment throughout the course.

- Arduino PRO MICRO - 5V/16MHZ
- Ultrasonic Range Finder (LV-MAXSONAR-EZ0)
- RGB Light Sensor ISL29125
- LED's, cables etc.
- Turtlebot3 Burger Robot equipped with i.a. a Raspberry Pi 3 and a 360°LiDAR sensor.

For coding we have used the language C for programming the Arduino on the Arduino software. For programming the Turtlebot we have been using Python and the command prompt with the built in nano-editor.

III. Design and implementation

Design and implementation of the system.

A. Part 1

B. Part 2

C. Part 3

As described, the Turtlebot is equipped with a LiDAR 360°laser sensor, which measures the distance. It continuosly returns an array:

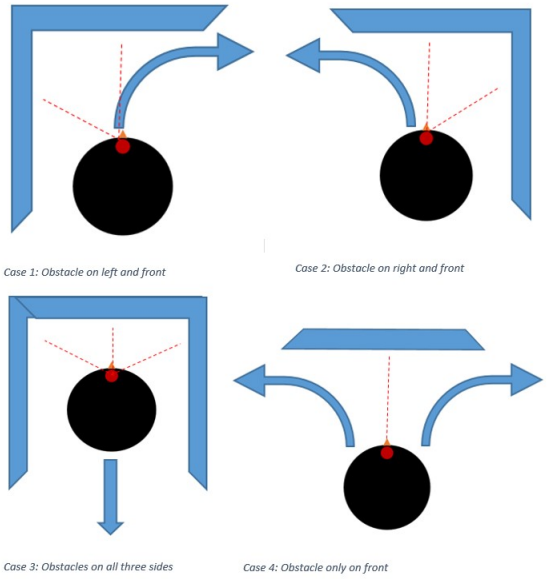
$$dist = [d_0, d_1, \ldots, d_{359}]$$

where d_i is the distance to the nearest object at angle i. We decided to look at a span of 120 degrees, which we divided into three distinct parts:

$$\begin{aligned} left &= [d_{15}, d_{16}, \ldots, d_{60}], \texttt{N=45} \\ front &= [d_0, d_1, \ldots, d_{14}, d_{345}, d_{346}, \ldots, d_{359}], \texttt{N=30} \\ right &= [d_{300}, d_{301}, \ldots, d_{344}], \texttt{N=45} \end{aligned}$$

We decided to do this so the robot more often would *turn* instead of just driving *backwards*, since this supposedly would achieve an overall higher linear speed.

For the different cases of obstacles we have made several cases of `if-else` statements. The cases and the decisions in each case can be seen in the figure below.



D. Part 4

IV. Experiment setup and results

A. Part 1

B. Part 2

C. Part 3

We decided to

D. Part 4

V. Discussion

In this section of the report we will discuss some of the difficulties we encountered during the project. Again, we have divided it into the four main parts of the project described earlier.

A. Part 1

B. Part 2

C. Part 3

1) **Getting wrong sensor measurements**
   We experienced quite a difficulty in getting the right measurements from the sensor. We knew that it would return an array of size 360 with one distance for each angle. However, we couldn't figure out how they were indexed and our robot thus behaved incorrectly and sometimes turned left when it should turn right.
   Solution
   We solved the problem by manually printing out different test angles and then physically measure that angle also. For instance printing `scan_ranges[45]` which gave some change in output distance for some specific angle, which we measured. In turned out that it our code should be reversed.
2)

D. Part 4

VI. Personal contributions

VII. References