Initial idea 4SC020

Group:	2
Ilias Aouaj	0779760
Florian Bastiaens	0818316
Menno Bouma	0784690
Yannick Knops	0755206
Anton Duys	0775033

Introduction

In this document, the initial ideas regarding the layout of the software of the robot are discussed. First a small introduction on the maze challenge is given, then the requirements followed by the software functions are discussed. Next the specifications of the requirements are given and finally several the maze solving algorithms are introduced.

Maze competition

The goal of the maze competition project is to create software that lets the pico robot solve a maze and drive through it fully autonomous as fast as possible. In this maze there will be certain obstacles that the robot has to overcome during the solving of the maze, for example doors, open spaces, dead ends and loops in the maze.

Requirements

Based on the restrictions of the maze competition and the conditions that have to be met, the following requirements are set:

- The maze has to be solved by the robot autonomously within a given timeframe
- The robot has to avoid collisions with the walls and other obstacles
- The robot has to recognize doors
- The robot cannot be idle for a certain time

Functions and Skills

To keep a well structured overview of the functions, they are divided into two categories: the basic hardware functions and the skills. The basic functions are the lowest level and they represent the hardware. The skills are combinations of basic functions to complete more difficult tasks.

The basic functions are:

- Translation using omniwheel actuators
- Rotation using omniwheel actuators
- Scan using the Laser Range Finder (LRF)
- Scan distance driven using odometry

These functions can then be combined to create the following skills:

Drive:
The robot moves using this skill. It uses the basic functions driving straight and turning. It also uses another skill, collision detection.

Collision detection/prevention:
The collision detection skill prevents the robot from hitting walls while navigating through the maze. It provides live feedback control from the LRF by detecting the distance to the walls and correcting the robots movement.

Junction recognition:
This skill is used to recognize junctions and corners in the maze. When the robot is moving through a corridor, the presence of a junction or corner can be determined from the data of the LRF. This data needs to be processed to make the robot aware of the junctions and corners.

Mapping:
This function maps the corners and junctions of the maze as nodes and corridors. The function is essential for preventing loops while solving the maze. This skill depends on the Junction recognition skill.

Localization:
This skill will locate the robot with respect to the maze. This skill will probably use a combination of the LRF and odometry data to determine a location in the map created with the mapping skill.

Recognizing doors:
As previously described, the robot should be able to open doors by beeping in front of them. Therefore the system has to recognize a door to prevent beeping at every obstacle and thus wasting time.

Specifications

Specifications related to the requirements are estimated in this section.

- Two attempts are allowed within a total time of 7 minutes.
- The jury decides if a robot bumps into a wall or obstacle. If so, the attempt is over.
- The robot has to recognize doors. Using a sound signal, the door can be opened.
- If the robot is idle for more then 30 seconds, the attempt is over.

Software overview

The way the software components work together can be seen in Figure 0.1. This is the example of the care robot and it is to a large extent similar to this project. The basic functions are represented by the hardware in robot context.

Maze solver approach

Several approaches are considered for solving the maze, which are explained in this section. At a later stage in the project a final decision on the algorithm can be made.

Right Hand Rule
In order to make the solving process not too complex, a common method can be applied to solve the maze. It is simply following the left wall or the right wall when entering the maze. This only works, however, when the entrance and the exit are both on the outer border of the maze. When this is not the case, the robot can get stuck in a loop using this algorithm. This might be solved by using

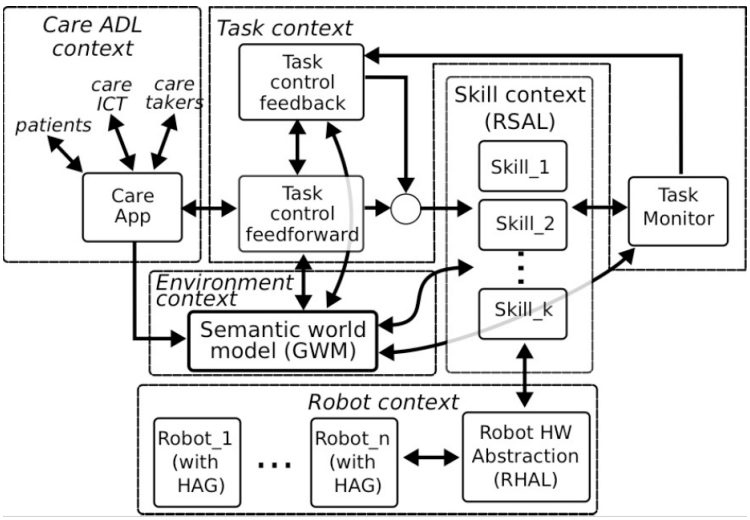

Figure 0.1: *Overview of the software of the care robot*

mapping and changing walls when a loop is detected.

Pledge Algorithm
The Pledge Algorithm is a maze solving algorithm that also deals with disjoint walls. When it is again applied with right hand wall following, everytime the robot finds an object it will take a left turn. It will then keep a counter. For every right turn it increments by one and for every left turn it decrements by one. When the counter returns to 0, the robot will deem the obstacle passed. It will then continue forward until it meets a new obstacle. For this algorithm the only requirement is that the exit is the outer border of the maze.

Trémaux's Algorithm
Finally the Trémaux's Algorithm is considered. At the first junction the robot will take a right turn. While doing this the robot marks the position that it has been. When the robot enters an area that it has already been, the area will be marked a second time. When the robot reaches a junction, it will take the route that is marked the least. If there are several options it will take the rightmost option. Trémaux's Algorithm is a very efficient algorithm to find the exit of a maze and has no requirements on where the entrance and exit are.