TU/e Mechanical Engineering

Quartile 4 - 2016/2017

4SC020

Embedded motion control

Initial design plan

Coordinators:
dr.ir. M.J.G. van de Molengraft
prof.dr. H.P.J. Bruyninckx

Tutor:
W. Houtman MSc.

Group 6

T.A.H. Hoenselaar	0857112
H.H. Ilisu	0852221
L.S. de Jong	0743679
L. Moormann	0861223
S.R.T. Straatman	0777325
J.R. van der Velden	0744957

Technische Universiteit Eindhoven University of Technology

Where innovation starts

Eindhoven, May 10, 2017

Goal

The goal of this project is to design and implement a robotic software system which will let the Pico/Taco robots autonomously solve and navigate through a maze as quick as possible. The software has to be written in C++ programming language and will be tested with two competitions, namely: the corridor challenge and the maze challenge. In this design plan the following five aspects will be described: requirements, specifications, functions, components and interfaces.

Requirements

In order to achieve the goal, the following requirements have to be met:

- The robot has to drive through any maze complying with specifications
- It has to escape the maze within the time limit
- Collisions with the walls must be avoided
- Doors must be opened
- Doors must be recognized in order to achieve the previous requirement
- The software has to be robust for imperfections in the measurement data and disturbances
- The robot must not be idle for a long period of time
- The robot must not ring the bell too often
- The robot must autonomously solve and navigate through the maze
- The Software must be started with only one executable

Specifications

The goal and the requirements will be achieved with the following specifications:

Robot

- The maximum transnational speed of the robot is 0.5 m/s
- The maximum rotational speed equals 1.2 rad/s
- The corridor challenge has to be solved in 5 minutes
- The maze challenge has to be solved in 7 minutes
- Both challenges have a maximum of two trials
- The laser range finder (LRF) has a range of 270°
- The wheel encoders have an unknown accuracy
- The robot must not be idle for more than 30 seconds

Maze

- The corners will be approximately 90°
- The wall distance is 0.5-1.5 meter
- There is only 1 door in the maze

- The door starts opening in 2 seconds
- The door opens if the robot is within 1.3 meter of the door
- The door is open in 5 seconds
- The number of rings must not be larger than the number of potential doors
- The maze may contain loops
- The maze can contain dead ends

Functions

The software must have the following functions in order to meet the requirements and fulfill the goal:

Table 0.1: *Table with functions*

Function	**Description**
Drive forward	The robot must drive forward unless something, for example a wall or a corner, is detected
Drive backward	The robot must drive a little bit backward if it is unable to rotate
Turn left	Make a 90° left turn
Turn right	Make a 90° right turn
Ring bell	The bell must be rang in order to open the door
Localize	The robot has to localize itself in the world model, because the odometry data isn't that accurate
Wait	The robot must wait at a dead end in order to check if it is a door

Components

The following components will be used to reach the goal:

Sensors

- Laser range finder which uses a laser beam to determine the distance to an object
- Wheel encoders (odometry) to estimate the position of the robot relative to a starting location

Actuators

- Holonomic base with omni-wheels
- Bell to open the door
- Pan-tilt unit for head (which will not be used)

Computer

- Intel I7
- Ubuntu 14.04

Interfaces

In the Figure below the following interfaces and their relations are visualized:

- World model
- Task
- Skill
- Motion
- User interface

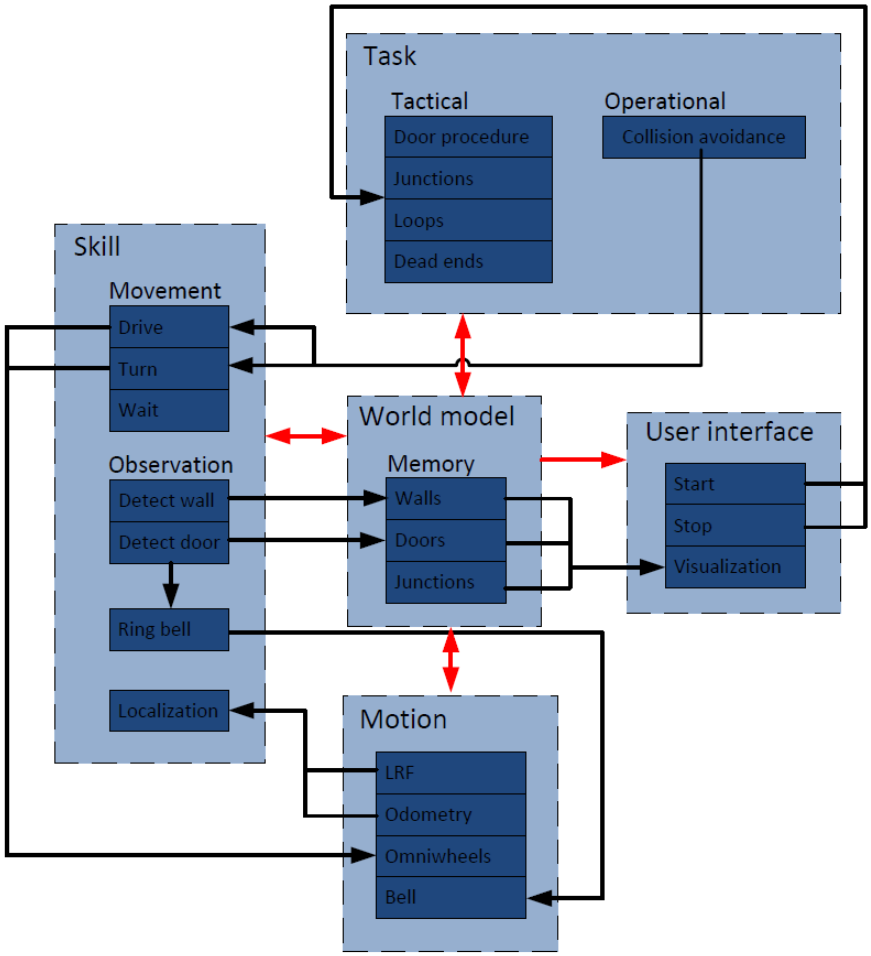

Figure 0.1: *Interfaces*

The main relations between the interfaces are colored red and can be described as follows:
World model → Task: The world model can give information about taken paths to the Task
World model → Skill: The stored observations in the world model are used for movement skills
World model → Motion: The world model can give data to the actuators
World model → User interface: The user interface needs the data from the world model to visualize the world model to the human
Task → World model: The task needs to store information about paths in world model
Skill → World model : The world model is build from observations
Motion → World model : The motion can give sensor data about the position to the world model