Technische Universiteit
Eindhoven
University of Technology

Faculty of Mechanical Engineering
4SC020
Quartile 4 - 2019/2020

Mobile Robot Control

Design Document

Baubekova, M. (1426311) - Master Systems&Control
Chatzizacharias, S. (1467751) - Master Systems&Control
Menaria, A (1419684) - Master Systems&Control
Verdonschot, J. (0893516) - Master Systems&Control
Walk, B. (0964797) - Master Systems&Control
Wingelaar, B. (0948655) - Master Systems&Control

May 4th 2020, Eindhoven

Contents

1 Requirements & specifications 2

2 Components 2

3 Functions 3

4 Interfaces 4

1 Requirements & specifications

Requirement	**Specification (corresponding)**
• Robot should exit the room in minimum time	• Robot must finish the trail within 5 minutes
• Robot should remain in motion as much as possible	• Robot cannot remain stationary for more than 30 seconds
• Robot should not bump into walls in the room	• Robot must maintain a distance of 25cm from walls at all times
• Robot should not bump into walls in the hallway	• Robot must remain in the middle of the corridor at all times
• Robot will be able to communicate and explain decisions	• Robot will communicate whenever a change of state occurs & provide reasoning behind it and the new state
• Robot should drive out of escape room autonomously	• Git controlled software must compile the algorithm into 1 executable which drives the robot without any further input
• Robot should be able to move	• Translational and rotational speed limits for robot are $\pm 0.5 m/s$ and $\pm 1.2 rad/s$ respectively
• Robot should be aware of its size	• Robot is 41 cm wide and 35 cm deep according to the Jazz robot datasheet.
• Robot must take its surroundings into account	• Laser range finder on the robot is capable of measuring distances ranging from 0.01m to 10m in a field of view of 4 rad (229.18°). This field of view is equally divided into 1000 points. When robot dimensions and its distance from the walls are known, collision into walls and corners can be prevented.

2 Components

The PICO robot in its environment is divided into components to make it structured. This will also help to organise the (software) Functions in a systematic way. The identified components are listed below.

- perception of sensor data
- control
- path planning
- motion of robot
- monitoring
- User interface
- Finite state machine (FSM)
- Physical environment model
  - walls
  - corridor
  - dynamic objects e.g. humans (not applicable for escape room)
- (virtual) hardware
  - PICO robot body
  - PC with the Ubuntu 16.04 OS
  - sensors
    - Encoders for wheels
    - Laser range finder (LRF)
  - actuators
    - holonomic omni-wheels

3 Functions

In figure 3.1 an overview of the software layout is given. The functions inside the Finite State Machine (FSM) compose the functions of the software design. The FSM will go through the states depending on the findings and the resulting change in booleans and variables. The different functions are shortly explained below.

- *RoomScan*: When the software is first started, the robot will scan the room while slowly rotating. After a scan 360 degrees around, it will be decided if a possible exit is identified. If an exit is found the boolean called ExitFound becomes true and the FSM will go to state *GoToExit*. If no exit is identified, the boolean ExitFound will be false and the next state will be *FindWall*.
- *GoToExit*: The robot will move towards where the exit was identified. Since the data can be inaccurate, this will only be an approximate movement. Once the robot nears the exit and the variable ExitDistance, which is the distance to the approximate location of the exit, is smaller than a certain, to be determined value, the FSM will go to state *ExitAlign*.
- *FindWall*: When no exit is found with the initial scan of the room, the plan is to make the robot follow the walls. To this extend, the robot first has to find the closest wall that it will start to follow and move towards it. Once the variable wallDistance, which is the distance from the robot to the wall, is below a certain, to be determined value, the FSM will go to state *AlignWall*.
- *AlignWall*: In this state the robot will be aligned parallel to the wall. This is done to make the robot face the correct direction to start following the walls. The robot will be aligned to face in the clockwise direction. Once the orientation of the robot is parallel to the wall, the boolean wallParallel will become true and the FSM will go to state *FollowWall*.
- *FollowWall*: The robot will move strictly forward without rotation. If the robot is aligned correctly, the robot will follow the wall. When the data indicates that the robot is no longer moving parallel to the wall, the boolean wallParallel will become false and the FSM will go back to state *AlignWall*. If the laser data indicates that a wall is ahead of the robot, the boolean wallAhead will become true and the FSM will go to state *InCorner*. When the variable LeftWallDist, the distance between the robot and the wall on the left, suddenly becomes larger than a certain, to be determined value, the FSM will go to state *IdentifyExit*.
- *InCorner*: If the robot approaches a corner, it will have to stop moving and rotate approximately 90 degrees. When the rotation is done and there is no longer a wall in front of the robot, the boolean wallAhead will become false and the FSM will go back to state *AlignWall*.
- *IdentifyExit*: At first when the robot enters this state, the boolean ExitFound will be re-initialized to clear it of any previous value to refrain the robot from getting stuck in a loop. When the robot has found a gap in the wall, it will have to be scanned to identify if this is the exit, or if it is not. It could be that the data is incorrect at first and there is actually no gap in the wall, or that there is a small gap in the wall which is not the exit. To this extend the width of the identified exit will be measured. If this is wide enough to be the exit, the boolean ExitFound will become true and the FSM will go to state *ExitAlign*. If the identified exit is not actually an exit, the boolean ExitFound will become false and the FSM will go back to *FollowWall*.
- *ExitAlign*: When the robot has successfully found an exit, it will have to align in front of it to make the movement into the exit. The robot should be clear of the side walls of the exit before moving into the exit. When the robot is aligned, the FSM will go to state *MoveIntoExit*.
- *MoveIntoExit*: The robot will move into the exit hallway. Since the width of the hallway is unknown, the robot can not use a fixed distance to either one of the walls as safety indication. Instead, the walls to either side will be kept to an equal distance of the robot so that the robot is always on the center-line of the hallway. If the measurements indicate that the distance to the walls is not equal, the boolean wallsEqualDist will become false and the FSM goes to state *ReAlign*. Since the length of the hallway is also not known, the robot will keep moving until the ends of the walls are slightly behind the robot. When this happens the boolean wallEnd becomes true and the FSM goes to state *Finished*.
- *ReAlign*: The robot stops moving forward to be able to re-align the robot better to the center-line of the hallway. When the walls on both side are again at an equal distance of the robot, the boolean wallsEqualDist will become true again and the FSM goes back to state *MoveIntoExit*.
- *Finished*: The robot stops moving and the challenge is finished.

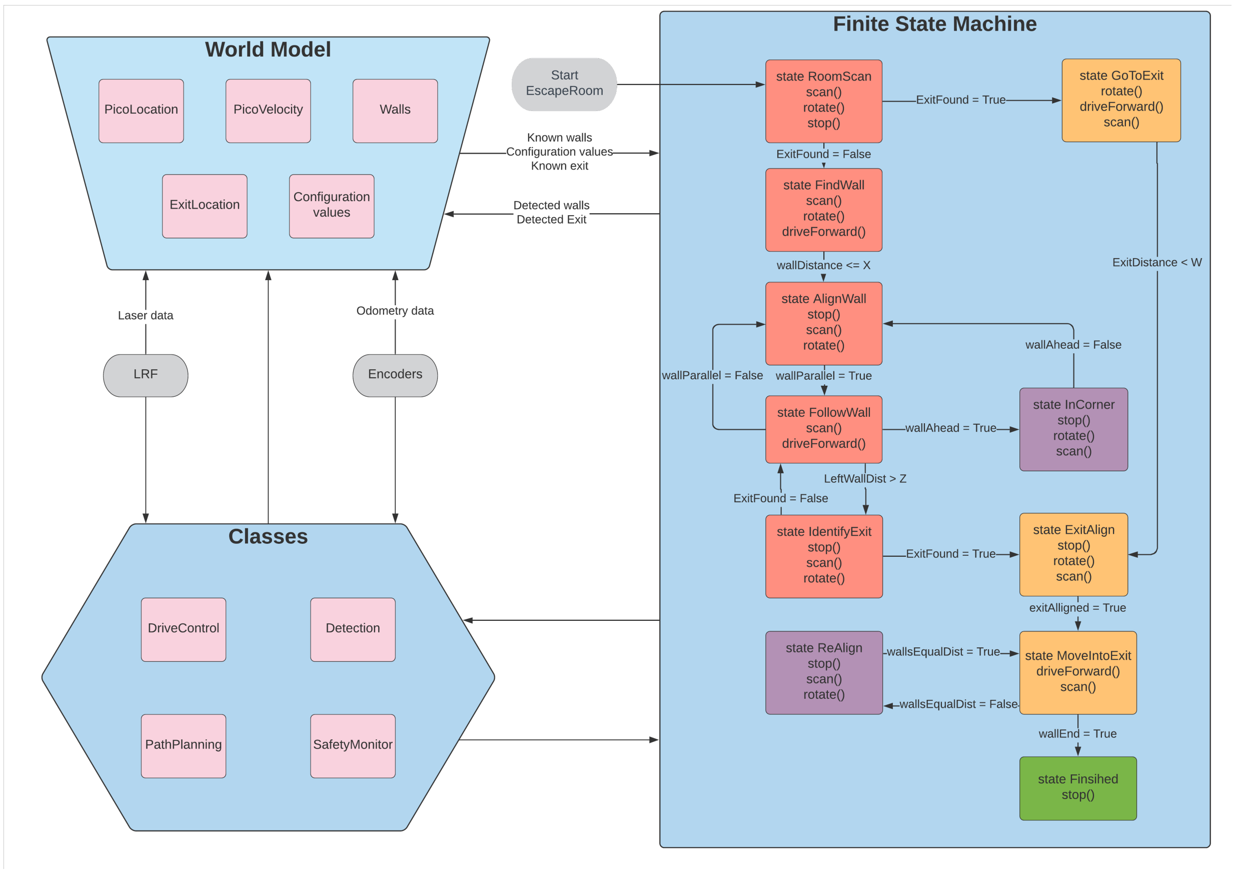

Figure 3.1: Schematic overview of software for escape room challenge. The colors of the blocks in the FSM represent the following: Yellow is for exiting the room, Red is for finding the exit, Purple is safety measures, and Green is finishing the challenge.

4 Interfaces

The interfaces represent the connection and/or communication between the components. The world model is updated through the interface with the perception of sensor data which is influenced by the sensor components of the (virtual) hardware sensors. The world model influences a collective of components (classes in Figure 3.1): path planning, drive control, detection and safety(monitoring), while these components influence the world model itself with the actuator components. All these components communicate with the Finite State Machine (FSM) which is used to solve the escape room problem in a logical way. Finally, the user interacts with the finite state machine by starting the challenge or receiving information on progress.