Department of Mechanical Engineering

4SC020 - Embedded motion control

Design Document

Quartile 4: 2020 - 2021

Supervisors:

Manuel Muñoz Sánchez
René van de Molengraft

Students:

S. Bakker (1002248)	s.bakker@student.tue.nl
B. Kempers (1398164)	b.kempers@student.tue.nl
P. Munns (1522795)	p.h.munns@student.tue.nl
L.A. Oei (1522779)	l.a.oei@student.tue.nl
S.J.J. van der Palen (1501054)	s.j.j.v.d.palen@student.tue.nl
W.W.M.MacAulay (1632817)	w.w.m.macaulay@student.tue.nl

Eindhoven, May 3, 2021

Contents

1 Requirements and specifications

The requirements are derived in accordance with the stakeholder: The client. The requirements are divided into environment requirements (yellow), border requirements (purple) and system requirements (green). These requirements have been translated into product specifications. The end product (software) should meet these specifications in order to comply with the requirements. The requirements and specifications are depicted in Figure 1.1.

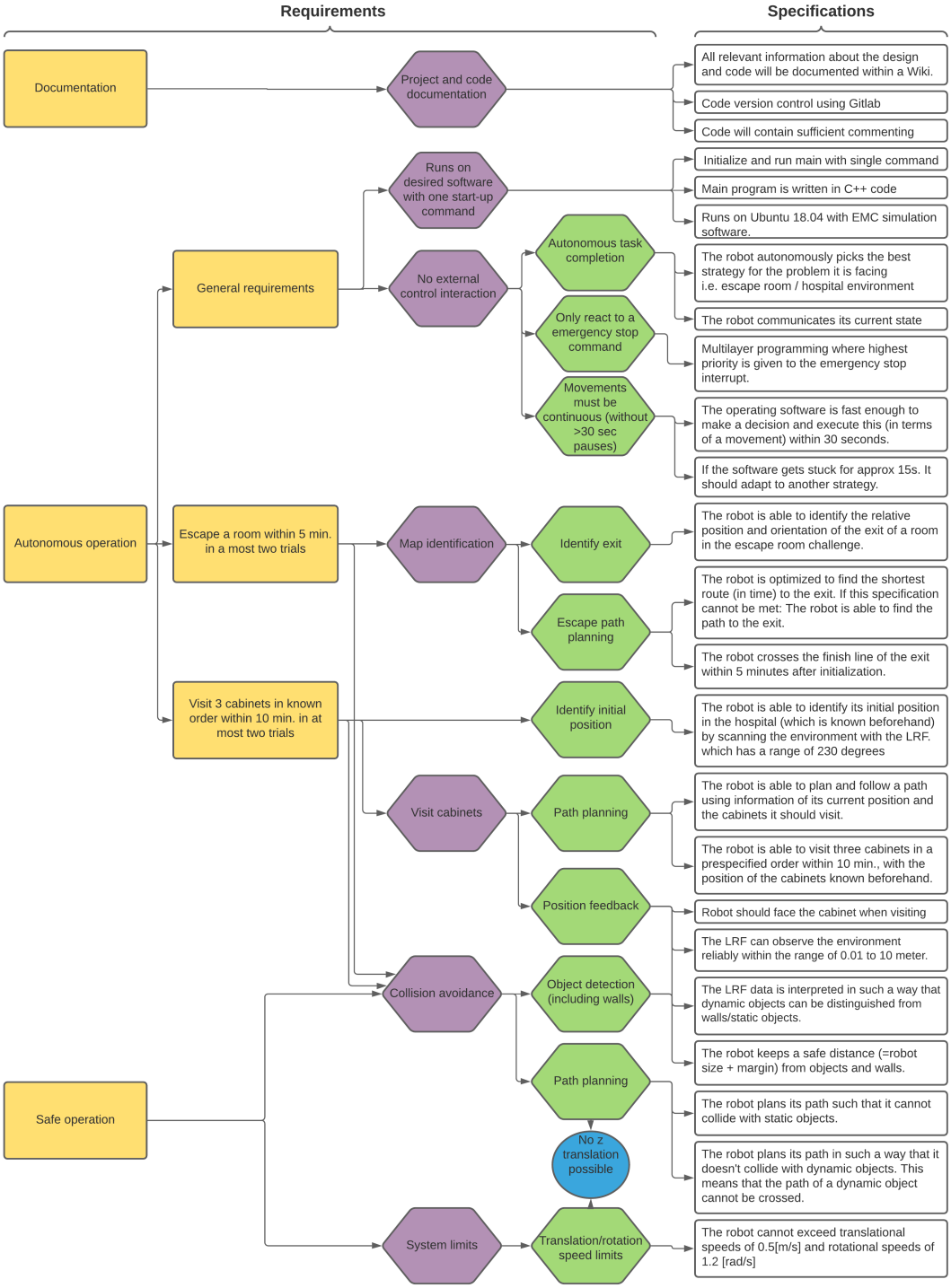

Figure 1.1: Requirements in order of increasing detail interconnected with the specifications.

2 Functions

The functions are designed using a finite state-space machine (FSM), as shown in Figure 2.1. This FSM is kept general to be usable for both escape room and hospital challenges.

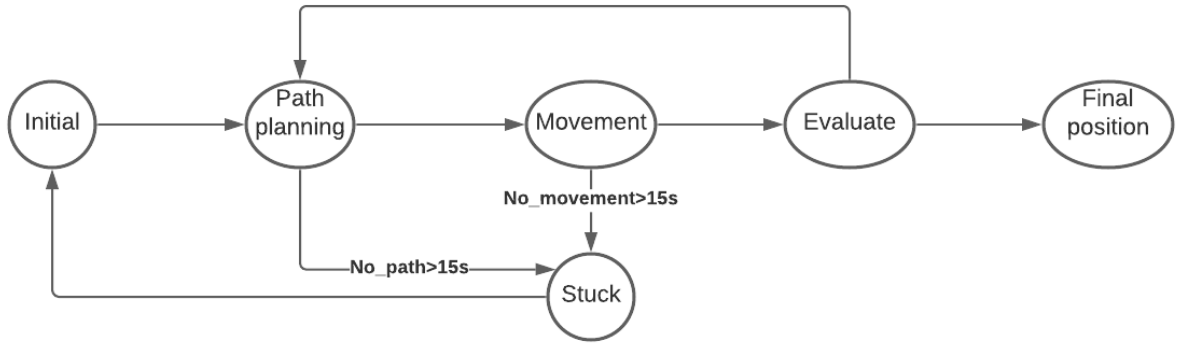

Figure 2.1: Finite state-space machine for a general autonomous movement of PICO

Each state is described below with the functions that the state utilizes. The functions are labelled from A to N and can be found in Table 2.1.

- **Initial State**: In this state, the environment is scanned to get an overview of the position of the robot. A map is made of the environment using the functions mentioned in Mapping & Perception. Items = A, D, E, F, G, H.
- **Path Planning**: A path is planned to complete the given task for either the escape room challenge or the hospital challenge. This path includes possible obstacles and the bubble function, which makes sure the robot does not move too close to obstacles. The path is thereby interpreted to useful control commands for PICO to be able to do a movement step. Items = I, M, J.
- **Movement**: In the movement state, PICO executes a movement step according to the planned path. Items = K.
- **Evaluate**: In the evaluation state, a new scan in the path and surroundings of the robot is made. The map of the environment is updated including (dynamic) obstacles. Items = B, D, E, F, G, H.
- **Stuck**: If the robot cannot compute a path or does not move for more than 15 seconds, it is classified as stuck and will move to available free space to try again. Items = C, N.
- **Final Position**: If during evaluation it is encountered that the final position is reached, the transition towards the state 'Final Position' is taken and the robot will stop. Items = L.

Table 2.1: The functions required for making an autonomous PICO.

Item	Function	Description
Sensing		
A	scanEnvironment()	Makes a 140 degree rotation to scan the environment for obstacles and free space for movement. Also, makes a first estimation of PICO's location within the map.
B	senseEnvironment()	Makes a scan in the path direction and nearby surroundings of the robot.
C	senseFreeSpace()	When PICO is stuck at a certain position it searches for free space to move towards.
Mapping and Perception		
D	mapEnvironment()	Maps the environment and updates it with dynamic changes in the environment, such as closed doors or people. It thereby updates the position of the robot with respect to the surroundings in case of drift.
E	lineSegment()	Processes the LFR data to identify lines and makes a distinction between separate lines, e.g. two walls.
F	findCorner()	Detects and locates corners in Cartesian coordinates. Identifies type of corner (inner or outer).

G	findEntrance()	Finds the centre location of a gap of a size within certain thresholds using sensCorner() and lineSegment().
H	findObstacle()	Identifies dynamic and static obstacles.
Path Planning		
I	planPath()	Plans a feasible path using interconnected waypoints from PICO's current location to the next objective. If a obstacle is in line with the planned path, it will recalculate an optimal path.
J	executePath()	Iterates through each waypoint and converts them from global/local Cartesian coordinates to control actions for the move() function.
Motion Control		
K	move()	Responsible for the position controller and converts position and rotation to velocity commands for the actuators.
L	stop()	Immediately sets all velocity commands to the actuators to zero.
M	bubbleRobot()	A virtual safety dome around PICO that makes it move away from nearby obstacles. If PICO senses that an obstacle is within its bubble, a counter control action will be applied in the movement state to move away from the obstruction.
N	moveToFreeSpace()	If PICO gets stuck, it moves to nearby free space to start its localization again.

3 Components and Interfaces

For the mobile robot control competition a PICO autonomous robot is used to carry out the project tasks. The PICO robot is split into two main features of hardware and software components, as shown in Figure 3.1.
The hardware components are made up of two sensors. Firstly, a laser range finder (LRF) is used by the PICO to scan surroundings and give vital distance data to help aid control the robot to fulfil its tasks. Secondly, wheel encoders are used to give data of PICO's translations and rotations. In addition, actuators consisting of a holonomic base (omni-wheels) rotate and translate the robot around the map.
Furthermore, for the challenges the PICO robot will be placed in a simulated physical environment consisting of walls, rooms, exits, dynamic objects(people) and static objects. However, for this project only a simulation of the PICO robot will be used so some physical components that are not stated in this document will be ignored.
The diagram of Figure 3.1 further shows how the components exchange information, data and how they are connected within the full system, i.e. the interfaces. The interfaces are vital to giving a good overview of the system and provide better planning for ensuring the specifications and requirements are met.

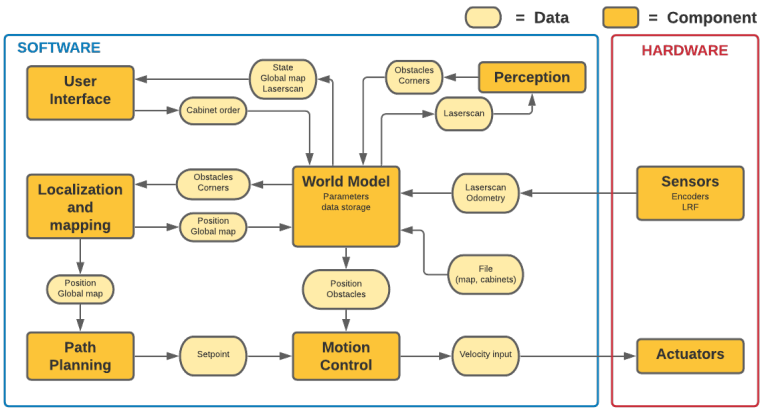

Figure 3.1: The interfaces between the different software and hardware components.

The components with interfaces of the system are discussed as follows:

- The world model is the central data storage for the system and provides many interfaces with components. It helps to provide a map of all the features and all external data.
- The perception software provides updated feedback of the surroundings as the PICO robot moves through the environment. The data of the surrounding objects and current position of the PICO are processed and updated to the world model.
- The localization and mapping of objects and obstacles allows the PICO to path plan and gives added data to the world model so the PICO robot can execute tasks.
- The path planning provides the strategy of the PICO robot for the project. By receiving information of the position of objects and obstacles via the localization and mapping of the external environment the path planning can give data to the motion control to execute the movement of the PICO.
- The motion control component of the robot helps the PICO move by sending data to the actuators provided by the world model and relating to the path planning strategy in order to execute the project requirements.
- The user interface can give visualisation to the user and can update the current status of parameters and data.
- The sensors provide data of the PICO's current surroundings to the world model