EINDHOVEN UNIVERSITY OF TECHNOLOGY

4SC020 - Mobile Robot Control

Department of Mechanical Engineering

System design specification

Group 8

Authors Abdullah Coşgun - 1020574
Bram Odrosslij - 1016625
Rinse Hobma - 1022328
Rick Peeters - 1021754
Ivo Mix - 1011609

Course coordinator dr. ir. M.J.G. van de Molengraft
Group tutor dr. ir. C.A. Lopez Martinez

Eindhoven, May 4, 2021

Requirements and specifications

The environment, border and system requirements are defined for each stakeholder. The main stakeholders consist of the hospital staff that serve as a client and the software engineers that design the software. In reality, additional stakeholders such as manufacturing companies and HR departments are also present, these are however excluded as it is out of scope for this project. A visualization of the requirements can be seen below:

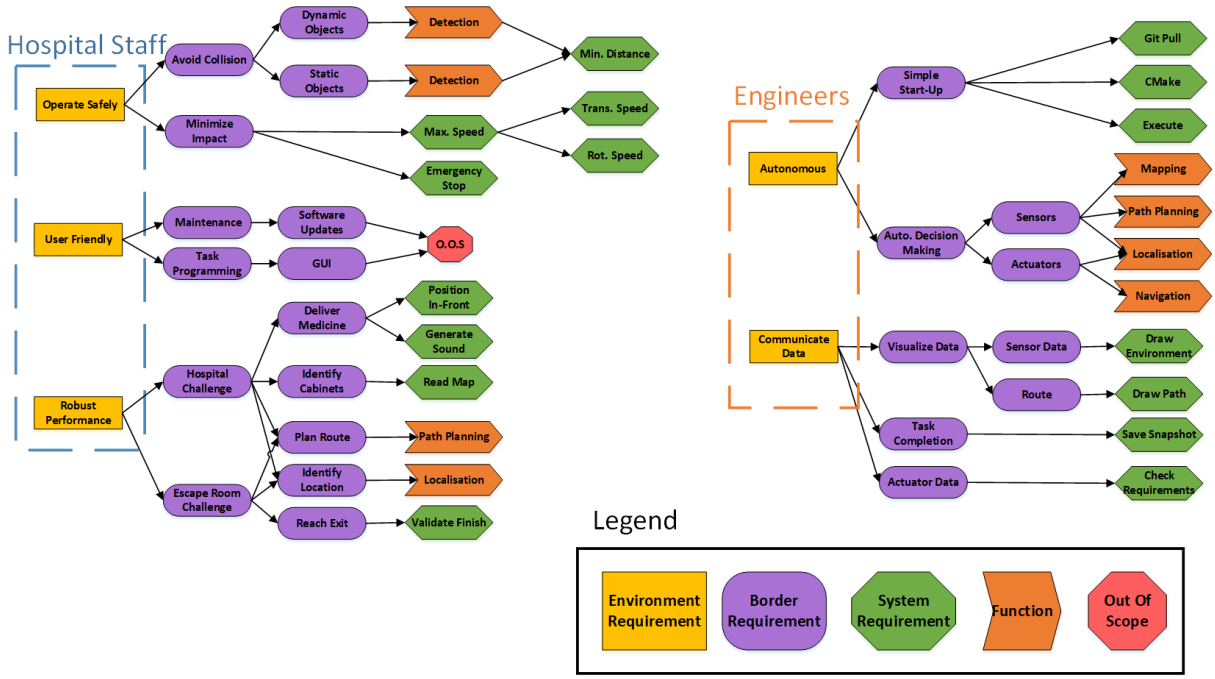

Figure 1: Environment, border and system requirements for the stakeholders.

The main requirements for the hospital staff consist of:

- PICO should **operate safely**: PICO should avoid collision with static/dynamic objects and keep a minimal safe distance to them. Furthermore, if collision were to occur, the impact should be minimal by setting a maximal translational/rotational speed and including an emergency stop. For the safe distance we take a maximum of 0.2 m. The maximum speed is limited in PICO: 0.5 m/s translational, 1.2 rad/s rotational, which is low enough to be considered as safe.
- PICO should be **user friendly**: software maintenance should be easy to handle and task programming must be simple to understand and program for the staff. A Graphical User Interface (GUI) is probably the best solution for this, however, as this is out of the scope for this project, it is not worked out in more detail.
- PICO should have **robust performance**: the given tasks to PICO in the hospital challenge should be executed properly under all circumstances. This includes changing environments/disturbances/human interaction. So PICO should be well aware of its environment and properly execute all functions. This requirement has overlap with the second stakeholder as both parties need a well functioning robot.

The main requirements for the software engineers consist of:

- PICO should be **autonomous**: PICO should complete all tasks without additional user input during operation. This includes localization, mapping, collision detection, path planning and navigation. Furthermore, PICO has to start up simply by following the 'git pull → CMake → exec' procedure.
- PICO should **communicate data**: PICO should communicate the current states to the user and visualize the data (position/route/map). Furthermore, snapshots need to be taken to prove that the tasks are completed. Lastly, actuator data should be communicated such that the requirements on the safety can be well monitored.

States

A so called finite state machine takes care of states that the system can be in. The sequence of functions that is called depends on this state. Below some proposed states are elaborated upon:

- **Initialize**: This state initializes the robot and makes sure everything is ready to go.
- **Where am I?**: Here, the system checks its surroundings by making a full circle scan. If this is not enough, other manoeuvres are performed to find to a recognizable location on the map. This state can also check whether the system is still in the location where it thinks it is.
- **Plan path**: This state plans a trajectory towards the next objective, either being an exit or cabinet.
- **Follow path**: This state makes sure that the path is followed by means of sensor data.
- **Objective reached**: Once an objective is reached this state is entered. From here we can continue to go towards the next objective or choose to stop the system if all objectives are reached.
- **Shut down**: This state shuts down the system.
- **Perimeter check**: This state is entered through the follow path state if an object is detected. The system then checks the perimeter and whether it is close to an object that is moving or if the system simply drifted from its intended path.

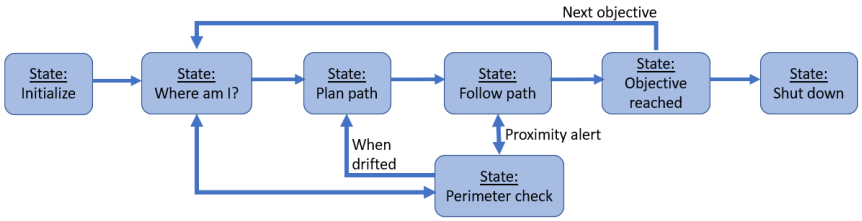

Figure 2: Finite state machine

Components

The components of the system are graphical displayed in Figure 3 and consists of the following components:

- **User**: the user inputs data to the world model and can receive data from the world model in the form of visualization or audio.
- **PICO**: PICO consists of multiple hardware parts (Omni-wheels, Laser range finder (LRF), Sonar, Camera, Audio signal) with each of the parts detecting or outputting information.
- **Finite State Machine (FSM)**: the FSM described the multiple states in which the system can be. The FSM is described in more detail in the previous paragraph.
- **Software architecture**: the software architecture can be described by multiple parts, each with its own functions. The functions that match the components are described in more detail in the next paragraph:
  - World Model: The world model stores data of the surroundings. The other components can use this data and send data back to the world model.
  - Localisation & Mapping: Locates the PICO robot, loads the map and plans the trajectory.
  - Detection: Receives the LRF and the odometry data, processes the data and outputs it to the world model.
  - Path planning: Determines the optimal path to the objective using the data from the world model and sends it to the motion control component.
  - Motion control: Controls the actuators of the PICO robot, which results in movement.

Functions

The complete task of the robot can be divided into several smaller subtasks. These subtasks can be described by functions that need to be executed in order to succeed. These functions can be divided into five main categories: detection, localisation, mapping, path planning and navigation (motion control). The most important functions will be shortly explained below.

Detection:

- **readLaserData()**: Data from the lasers is obtained. This data contains the measured distances, but also information like the maximum range of the sensor and the minimum and maximum angle.
- **readOdometryData()**: This function retrieves the odometry data, which contains both the displacement in x- and y-direction and the rotation since the start.
- **getMinimumDistance()**: The distance towards the most nearby wall is retrieved.
- **wallDetected()**: This will become true if a wall is too nearby and a change of direction should be executed in order to avoid collision.
- **movingObstacle()**: A moving obstacle is detected which potentially will collide with PICO, therefore the current trajectory needs to be adjusted.

Localisation & Mapping:

- **readStaticMap()**: Read the static map into vector format.
- **extractFeatures()**: Extract features such as points/lines/corners.
- **fitData()**: Fits the data to obtain best estimate for robot location.

Path planning:

- **getTarget()**: Obtain the location of the target destination in the world map.
- **planGlobalPath()**: An appropriate global trajectory will be planned to reach every corridor in the most efficient way. The actual path can deviate from this path during operation.
- **planLocalPath()**: A local trajectory will be planned to reach the local segments in the global path. This local path accounts for static/dynamic objects and disturbances.

Navigation:

- **driveRobot()**: control the speed in x-direction, y-direction and rotation.
- **Avoid()**: When the trajectory of a moving object will cross the trajectory of PICO, PICO should change the trajectory to avoid this.
- **Stop()**: When a (moving) obstacle is too close, PICO should stop immediately, to avoid or minimize impact.

Interfaces

The interfaces are the connections and communication between the different components of the system. For example, the interface between the sensor data and the world model should allow the flow of information from the sensors to the world model block. The mapping block for example needs to have an interface which allows to read and write information of the world model in order to compare its perception to the existing map.

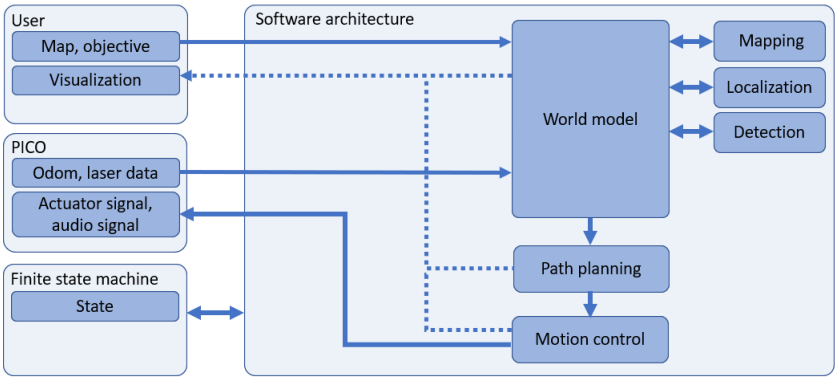

Figure 3: Interfaces

The figure shows a general layout of how the components in the system are connected. The components could differ from the eventual system.