Technische Universiteit **Eindhoven** University of Technology

Department of Mechanical Engineering

Embedded Motion Control

Initial Design Report

Group 1

Authors:

A. Aggarwal *(1279491)*
a.aggarwal@student.tue.nl

A.H. Ahmed Ajmal *(1279602)*
a.h.ahmed.ajmal@student.tue.nl

J.J. van Steen *(0914013)*
j.j.v.steen@student.tue.nl

N. Rozkvas *(0924842)*
n.rozkvas@student.tue.nl

W. Verhoeven *(1031018)*
w.verhoeven@student.tue.nl

Responsible teacher:

dr.ir. M.J.G. van de Molengraft
M.J.G.v.d.Molengraft@tue.nl

Eindhoven
May 11, 2018

1 Introduction

In this document, the initial design of the software will be given that describes the steps in how to make PICO complete the escape room challenge. In this challenge, PICO is placed in a rectangular room with one exit, which contains a corridor of over 3 meters to the finish line. PICO should autonomously find the exit and drive past the finish line.

To start off, the requirements and specifications for the escape room challenge and for the hospital room challenge have been described. The requirements for the hospital room challenge have been included so that it is easier to design software that is also still useful for this final challenge. After that, the interfaces of the initial design will be discussed, which gives an explanation of the model of the software that has been chosen, including the plan how to escape. After that, the components are described, which includes both the physical components and the software components, where the latter is a description of the software architecture obtained from the interfaces. Finally, the functions that are going to be used in this software architecture have been described.

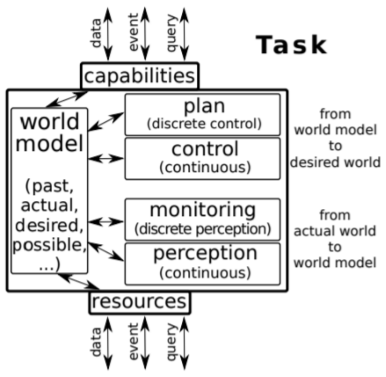
2 Requirements and specifications

The first step to come up with the design of the software is building a list of requirements and attach some specifications that describe how to implement the requirements in the software. A general list of requirements and specifications has been made that should be adhered to for both challenges, as well as a list of requirements that should hold specifically for the escape room challenge or the hospital room challenge. This can be seen in the table below:

	Requirements	**Specifications**
General:		
	PICO has to avoid all walls at any time	PICO should always stay always stay away from walls at least 15 centimeters.
	PICO should complete its task well independent of the initial conditions.	PICO has to scan its surroundings with a frequency of at least 5 Hz while moving to prevent bumping into a wall.
	PICO should operate fully autonomously.	The software should be set up according to the information on the wiki page.
	PICO cannot translate or rotate faster than prescribed.	An uncertainty margin should be present when building a world model to deal with imperfections in the real world
	PICO has to deal with small imperfections in the real world it operates in.	The maximum translational velocity of PICO is 0.5 m/s and the maximum rotational velocity of PICO is 1.2 rad/s.
	The software should be easy to set up.	The final software should not make use of teleoperation.
Escape room challenge:		
	PICO has to cross the finish line as fast as possible with a maximum of 5 minutes.	PICO should drive through the center of the corridor to avoid the wall.
	PICO needs to stop when it crosses the finish line.	PICO needs an algorithm to identify the exit independent of its starting position.
	PICO has to identify the exit of the escape room.	
Hospital room challenge:		
	PICO has to map all different rooms in the hospital.	An algorithm should be created to identify all different rooms in the hospital and store all corner points.
	PICO should identify an object in a room that was not there previously.	There should be an algorithm that interprets data from the world model to understand ans use the high level hint.
	PICO has to be able to drive and park backwards without hitting the walls.	
	PICO will need to interpret a high level hint.	
	The challenge should be finished in 5 minutes.	

3 Interfaces

A high-level overview of the system architecture is created as shown below in the figure and text.



[Task Overview (plan)] Escape Room Challenge

1. Initialize the sensors and actuators.
2. Scan surroundings from given position and orientation.
3. Check for exit
4. IF exit is found: Move to exit and skip to step 7.
   ELSE: Turn 180 degrees and perform step 2 and 3 once again.
5. Check for exit
6. IF exit is found: Move to exit and skip to step 7.
   ELSE: Move into the direction of the farmost point (averaged over multiple points) for a quarter of this distance.
7. Return to step 2.
8. Option 1: Identify the wall of the exit and follow the wall at a given distance.
   Option 2: Identify the wall and orient the robot to face the exit, then drive forward while remaining in the center.
9. Stop once the finish line has been crossed.

Table 1: Description of Interfaces

Capabilities	Omni-directional drive
	Turn
	LRF scans
World Model	Store all data.
	Communicate and mediate between tasks.
Resources	LRF sensor data
	Odometry data
Planning	Generate robot trajectory based on information from Perception & Monitoring interface.
Control	Generate motor input from reference coordinates
Perception and Monitoring	Identify lines
	Identify exit
	Identify furthest point
	Check movement errors

4 Components

4.1 Physical Components

This section details the physical components present on-board the PICO robot.

- Sensors
  - **Laser Ranger Finder (LRF)**: The LRF is a sensor used to determine the distance of objects from the robot. This is accomplished by sending sending laser pulses to nearby objects and measuring the time taken for the laser pulses to be reflected back from the objects.
  - **Wheel Encoders (Odometry)**: The wheel encoders are used to determine the position and velocity of each wheel of the robot.
- Actuators
  - **Holonomic Wheels (Omni-wheels)**: The PICO robot has three wheels that give it omni-directional movement capabilities.
- Computation Hardware
  - **Intel Core i7 Processor (Running Ubuntu 16.04)**

4.2 Software Components

The software of PICO can be split into different components in which the incoming data is converted into the outcoming data. The software components of PICO can be found in the figure below. The block-diagram starts with the incoming raw sensor data. This data needs to be filtered to because of sensor noise, this in done in the filter-block. The filtered data of the LRF sensor can be used for four different purposes, these blocks are:

- Wall detection: Uses the filtered LRF data to detect walls on the left and right side of PICO. Output is the minimum distance to the wall on the left and right side.
- Furthest point detection: LRF data is used to find the exit, in case the exit is not found from the current position of PICO, PICO has to change his position and look again for the exit. Output is a new position from where pico should start looking for the exit.
- Exit detection: LRF data is used to find the exit, in this situation the exit is found. Outputs of the component are the coordinates of the two corners of the exit.
- Finish identification: When PICO is positioned inside the corridor the finishline has to be determined, from the LRF data. Outputs are the coordinates of the finish line.

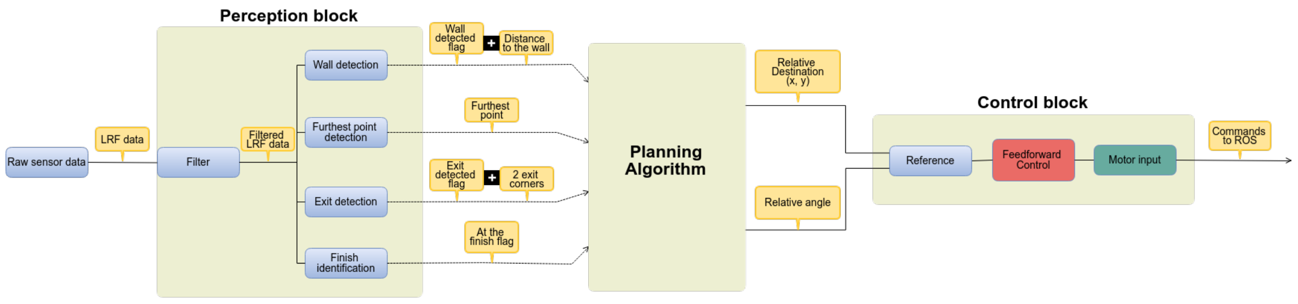


Figure 1: Software Components of PICO.

The planning algoritm uses the output of the perception block to determine the extual location where pico has to navigate to. The planning algorithm uses the coordinates of the exit, finishline or furtest point and tries to avoid the walls. The output of the planning algoritm are coordinates of the position and the turning angle which PICO has to make before navigating in a staight line to that position.

The Control Block is used to translate the destination coordinates and turning angle to in command to sent so PICO.

5 Functions

The table below gives a list of all functions (Low-level is not complete yet) which will be used to let PICO drive to the exit and through the corridor. The functions are sorted in a hierarchy, a description of the hierarchy is given below:

1. High-Level: The high-level functions are the fases in which PICO can be in. When all high-level functions are completed it means PICO has accomplished the task to find the exit and drive through the corridor.
2. Mid-Level: The high-level functions consists of mid-level functions. Depending what high-level function PICO is in a sequence of mid-level functions are looped, until the 'check' function is completed. When the 'check' function has a positive result PICO can move to the next high-level function, which corresponds to an other mid-level function sequence.
3. Low-Level: The mid-level function uses the low-level functions to let PICO use the sensors, actuators and do calculations. The low-level functions can be used in different mid-level functions.

Table 2: Description of Functions

Function Hierarchy	**Function**	**Description**
High-level	`initialize`	Initialize all sensors (and actuators).
	`phase_toExit`	Find and drive to the exit.
	`phase_throughCorridor`	Drive though the corridor.
	`terminateProgram`	Stop and exit the program.
Mid-level	`exit_localization`	Localize the exit.
	`exit_localizationFurthestPoint`	Define a point from which the exit can be localized.
	`exit_determineTrajectory`	Determine the trajectory to come to the localized point.
	`exit_driveTo`	Control input to drive according to the trajectory.
	`exit_check`	Check if PICO is at the location where the exit is.
	`corridor_localization`	Localize the end of the corridor (finish line).
	`corridor_driveThrough`	Drive through the corridor, while scanning both walls.
	`corrodor_check`	Check if PICO is at the location where the finish line is defined.
Low-level	`drive`	Drive at a certain speed.
	`turn`	Turn for a certain angle.
	`lineFit`	Algorithm which uses LRF data to determine walls.
	`scanLRF`	Get data from LRF data.
	`scanOdometry`	Get position from odemetry data (not necessary yet).
	`driveForward`	Drive at a pre defined speed.
	`turnAround`	Turn for a pre defined angle of 180 degrees.