

1 Overview

In this document a summary of the embedded software design is presented. This software is used to solve the following problems:

- **Corridor challenge:** The robot should drive through a corridor and take the first exit
- **Maze challenge:** The robot should drive through a maze and find the exit

2 Requirements & Specifications

Type	Requirement	Specification
General	<ul style="list-style-type: none">- Be able to 'solve' any given configuration of walls- Do not stand still too long: Detect time that robot is standing still, initiate movement after set time- Do not bump into walls: Detect walls, Define minimum distance- Move autonomously: Detect openings, junctions, crossings, dead ends, open spaces etc. and make optimal decision- Software easy to set up: As defined on general wiki page.- Only one executable is allowed. The software will be updated on the robot before the challenge starts- Stop movement after task achieved	<ul style="list-style-type: none">- The maximal translational velocity of PICO is 0.5 m/s- The maximal rotation velocity of PICO is 1.2 rad/s- Pico should not stay still for more than 30 seconds- Complete task within 2 attempts- The LRF has a width of about 4 rad (from -2 to 2 rad), with a resolution of about 1000 points.
Corridor	<ul style="list-style-type: none">- Finish the corridor challenge fast: Detect opening either on left/right, take turn, stop after finish line	<ul style="list-style-type: none">- Back wheel across finish line within 5 minutes. Terminate afterwards.
Maze	<ul style="list-style-type: none">- Finish the maze challenge fast: Navigate maze, find exit, stop after finish line- Be able to reconstruct maze- Determine difference between dead end and door- Deal with open spaces- Deal with loops- Be able to open doors	<ul style="list-style-type: none">- Back wheel across finish line within 7 minutes. Terminate afterwards.- Decide where to go when at a T-junction or crossing- Ring a bell and wait at a dead end to check for a door

3 Functions

function name	description	
Low-level	<code>initialize</code>	Initialize actuators
	<code>readSensors</code>	Read the odometer and laser data
	<code>turnLeft</code>	Turn 90° left
	<code>turnRight</code>	Turn 90° right
	<code>turnAround</code>	Turn 180°
	<code>stopMovement</code>	Stop omniwheels
	<code>driveForward</code>	Accelerate or decelerate
	<code>driveBackward</code>	Drive backward
	<code>driveLeft</code>	Move left
	<code>driveRight</code>	Move right
	<code>ringbell</code>	Ring the bell of the door
Mid-level	<code>detectWall</code>	Detect a wall (~30 cm)
	<code>detectCorner</code>	Detect a corner (crossing of two walls)
	<code>detectDeadEnd</code>	Detect a dead end
	<code>detectFinish</code>	Detect the finish line
	<code>detectOpenSpace</code>	Detect an open space
	<code>detectOpenWorld</code>	Detect if in the open world (like the maze exit)
	<code>detectTJunction</code>	Detect a T-junction (where three corridors meet)
	<code>detectCrossing</code>	Detect a crossing (where four corridors meet)
	<code>shutDown</code>	Terminate robot, if required
	<code>checkDoor</code>	Send a signal and wait x seconds
<code>chooseCorridor</code>	Choose which corridor to take	
High-level	<code>stayBetweenWalls</code>	Stay in the center of two walls
	<code>createMap</code>	Build map of surroundings
	<code>trackPath</code>	Track the path through the map
	<code>detectLoop</code>	Detect a loop in the maze
	<code>detectStuck</code>	Detect if stuck
	<code>optimalDecision</code>	Decide next move based on given algorithm

4 Components

The PICO robot consists of multiple components which are listed below.
Sensors:

- Laser Range Finder (LRF): Through the LRF on the PICO one can detect the distance to an object. This is accomplished by sending a laser pulse in a narrow beam towards the object and measuring the time taken by the pulse to be reflected off the target and returned to the sender.
- Wheel encoders (odometry): Through the encoder we will obtain the speed of the wheels which can be used to control PICO based on the

provided data.

Actuators:

- Holonomic base (omni-wheels)

PICO Computer:

- Ubuntu 14.04
- Intel I7

5 Interfaces

