1 Overview

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

- **Coridor 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

Table 1: Requirements and specifications

Type	Requirement	Specification
General	- Achieve task - Do not bump into walls - Move autonomously - Software easy to set up - Only one executable is allowed. The software will be updated on the robot before the challenge starts - Do not stand still too long - Stop movement after task achieved - Be able to 'solve' any given configuration of walls	- The maximal traslational 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	- Finish the corridor challenge fast	- Back wheel across finish line within 5 minutes
Maze	- Finish the maze challenge fast - Be able to reconstruct maze - Determine difference between dead end and door - Deal with open spaces - Deal with loops - Be able to open doors	- Back wheel across finish line within 7 minutes

3 Functions

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

4 Components

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