1 Requirements

- **Software easy to setup**
  Updated with one easy command, e.g. 'git pull'.
  Software can be compiled using 'cmake' and 'make'.
  One executable to start software.
- **Autonomously solving the corridor challenge**
  Solving the corridor challenge has to be done within 5 minutes.
- **Autonomously solving the maze challenge**
  Solving the maze challenge has to be done within 7 minutes.
- **Avoiding collision**
  Do not bump into walls or doors.
- **Recognize and open a door in the maze**
  There might be multiple dead ends in the maze, one of which is a door. The robot has to be autonomously open the door (by ringing a bell) to solve the maze.
- **Detect corridors, corners, T-junctions and intersections**
  The maze and corridor challenge consists of several, corridors, corners, T-junctions and intersections. Various algorithms created in order to detect these.
- **Navigate through corridors, corners, T-junctions and intersections**
  When corridors, corners, T-junctions and intersections are detected, the robot has to drive trough these. In order to do this, strategies have to be created for each of these.
- **Detecting dead ends**
  The maze can contain dead ends, these have to be distinguished from doors.
- **Detecting maze exit**
  The exit of the maze has to be detected in order to know when the maze is finished.
- **Navigate open spaces**
  The maze can contain open spaced, when these are detected these has to be explored by navigating trough them.

2 Specifications

The specifications are related to the requirements specified above. The specifications are given values,

- The time to reach the end of the maze is 7 minutes.
- The time to accomplish the corridor challenge is 5 minutes
- The robot may not be idle for more than 30 seconds.
- When the robot find a door, it rings a bell and the door will be opened manually.
- Two attempts to solve the maze are allowed. Both attempts have to be finished within 7 minutes total.
- Two attempts are allowed to solve the corridor challenge. Both attempts have to be finished within 5 minutes.
- When the robot bumps into a wall the attempt is over.

- **Maze**
  - All walls are positioned approximately perpendicular to one another.
  - Open spaces might occur (as depicted in Figure 2.1).

  - There may be loops in the maze, which means that some walls may not be connected to other walls.
  - Walls are approximately parallel.

- **Door**
  Doors are specified as described by Figure 2.1

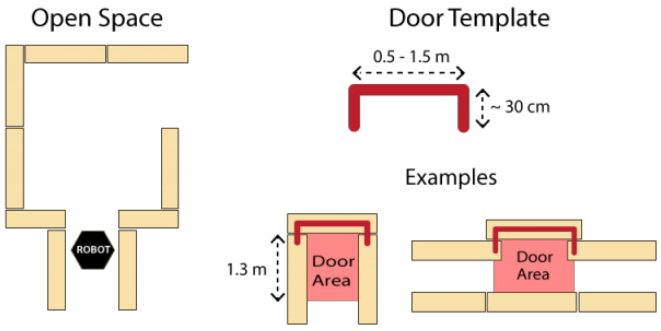

Figure 2.1: *Maze open spaces and door specifications.*

3 Functions

The functions are categorized in three different levels of complexity: Low, Mid and High level. An overview of the functions is given in Figure 3.1.

Low	Mid	High
Input		Mapping
Read data sensors	Detect • Walls • Corridor • Intersections • Open spaces • Dead ends • Doors	Localization Navigate trough maze Positioning for door and ring bell
Output		
Ring bell Move Pico • Straight • Rotate • Sideways	Collision avoidance	

Figure 3.1: *Function overview. The functions are grouped based on complexity as well as input/output type.*

4 Components

The following components are accessible on the pico:

- sensors:
  - Laser Range Finder (LFR)
  - Wheel encoders (odometry)
- actuators:
  - Holonomic base (Omni-wheels)
- Computer
  - Intel i7
  - Ubuntu 14.04

5 Interfaces

The interfaces between functions are shown in Figure 5.1. Here the functions are devided into: Tasks, Skills, Motion and World Model. The world model contains the functions that are used to keep track of Pico's surrounding. Tasks contains the functions on the highest level where the robot decides what it is going to do, like solve the maze. The skill block consists of all functions that make the robot perform lower level tasks such that the high level objective can be achieved. The motion block does the low level operations that make the skill functions work.

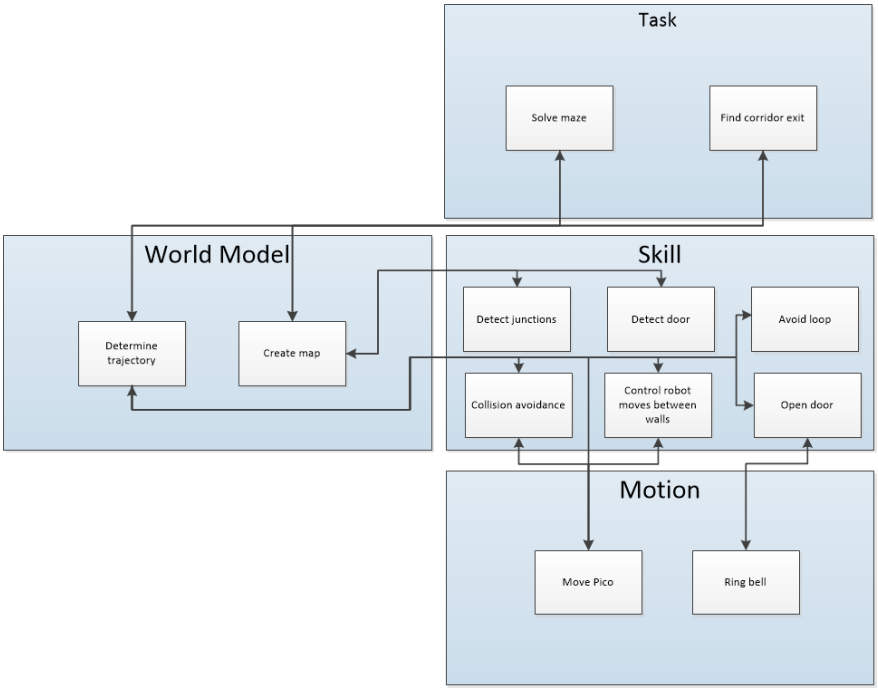

Figure 5.1: *Interface overview of all functions needed by Pico to solve the maze and corridor challenge.*