# Maze Solving by an Autonomous Robot

Discovering the real-time systems

Project Report

Quentin Chatelais
Horatiu Vultur
Emmanouil Kanellis

Aalborg University
Electronics and IT

Here you can write something about which tools and software you have used for typesetting the document, running simulations and creating figures. If you do not know what to write, either leave this page blank or have a look at the colophon in some of your books.

**Electronics and IT**
Aalborg University
http://www.aau.dk

**AALBORG UNIVERSITY**
STUDENT REPORT

**Title:**
Maze Solving by an Autonomous Robot

**Theme:**
Embedded Systems

**Project Period:**
Fall Semester 2010

**Project Group:**
ESS7

**Participant(s):**
Quentin Chatelais
Horatiu Vultur
Emmanouil Kanellis

**Supervisor(s):**
Arne Skou

**Copies:** 1

**Page Numbers:** 21

**Date of Completion:**
October 28, 2014

**Abstract:**

Here is the abstract

# Contents

# Todo list

# Preface

This document is the report concerning the project done in the fall semester 2014 by our group. The group was composed of three international students who studied Embedded Software Systems on the 7th semester at Aalborg University.

Aalborg University, October 28, 2014

Quentin Chatelais
<qchate14@student.aau.dk>

Horatiu Vultur
<username2@student.aau.dk>

Emmanouil Kanellis
<username3@student.aau.dk>

# Chapter 1

# Introduction

## 1.1 Context

The rapid growth in the technology in our times is greater than ever. More and more autonomous robotic devices are infiltrated in the lives of people making their easier. Great amounts of money are spent annually on the research of building smart robotic vehicles. In many cases, autonomous robots can be the best option for specific missions. Conservative ways of rescuing survivors are time consuming and harmful for the survivors. Hence unmanned autonomous robotic vehicles which can enter the collapsed builds searching for survivors maybe a solution of finding survivors faster and safer. Being equipped with the necessary sensory devices unmanned autonomous robotic vehicles can scan the environment sending precious information to the rescue teams about the location of survivors. In addition, there are also places where use of robots is the only way to achieve a work. Space exhibitions, nuclear plants, chemical factories, or any environment unreachable for humans could be explored by a autonomous robots. So, independent mapping and localization for a robot became one of the main goals in robotics technology. This is a complex problem, and is not totally solved today. The principal difficulties are the accuracy of measurements, and the real-time processing, correlated with minimum processing power, due to non-infinite capabilities in most of the embedded systems. The whole above problem can be separated in three subproblems: the localization and the mapping, known as SLAM[1], and the path planning.

### 1.1.1 Localization

Localization is the main problem for finding a path. The robot has to know the answer of the question : "Where I am?" because from there he can find the path to new position. The localization algorithms depends a lot of the environment in which robots are found and the characteristics of the robot. If the robot is indoors it has to use a different algorithm like it was outside. There are different algorithms that

[1]Simultaneous Localization And Mapping

can be used for the localization like dead reckoning, least mean squares or can be used modern technology like GPS. This problem is in tight relation with mapping problem. These problems can be categories in three ways: where the map and location are known, where the map is known but the location is unknown and the hardest category it would be where the map and the location are unknown. In every algorithm used to localize the current position of the robot there would be some errors in the measurements, that is why it is indicated to use multiple algorithms to detect more precisely the current position.

### 1.1.2 Mapping

Mapping a environment is a complex problem, which have a lot of solutions, each with its advantages and drawbacks, and dealing between the accuracy of the map and the computing power available on the robot. To be more precise, this problem of mapping has two components : how to save the map into memory (map representation), and how to recognize it while moving or not, with which kind of sensors (map learning). So one of the main troubles with that is there is often a few memory and computational power, but the more accurate algorithms are of course the most greedy in CPU and memory. The difficulty is to balanced all this parameters. That can be achieve for example in :

- not coding the best algorithm ever, but accurate enough to map correctly ;
- giving the robot some informations about the environment in order to simplify the problem, so it's not a full autonomous robot anymore, but mapping is easier (for example, if robot know it'll map an environment only made of orthogonal plans, it's easier for both map representation and map ) ;
- limiting the accuracy of registered informations where it's not useful to be precise, and increase resolution on critical points.

Also, the system has to be fault-tolerant, because sensors can be wrong. Combine several sensors and compare their values is a way of manage that. Finally, mapping is also extremely dependant of localization, so each mistake with localization make the mapping goes wrong, and size of errors may grow up with the larger of the map.

### 1.1.3 Path planning

Path planning or motion planning is the navigation process of an autonomous robot in a specific area in which are existing numerous obstacles that needs to be avoided. In our case the autonomous robot must find the finish position with the fastest or shortest possible way. This process can be performed by various algorithms such as Grid-Based Search, Interval-Based Search, Geometric Algorithms, Potential field, Sampling-Based Algorithm.

## 1.2 The project

For this project, the choice is to simulate the problem that has just been presented by putting an autonomous robot with a maze as external environment.

# Part I

# Project specifications

# Chapter 2

# Maze solving : problem definition

The basic definition of the problem is the following : "An autonomous robot has to solve a maze in which it has been placed". The robot is put at a random position in the maze, and have to find its way to another position known as the end, and recognizable by the robot. Several parameters have to be specified before the beginning, in order to define the project as precisely as possible. This parameters (or rules) are described below :

1. The maze is a large rectangle, where there are severals walls.
   (a) There are walls all around the maze (it is closed) ;
   (b) The walls are straights ;
   (c) the walls are perpendicular to each other ;
   (d) The maze can be divided in squares (also called cells) of the same dimensions, which do not contain inner walls. In other words, walls have to be positioned only on the sides of this squares.
2. The robot is independant.
   (a) The robot initially knows nothing about the shape of the maze ;
   (b) The robot can know the dimensions of the maze ;
   (c) The robot can assume that he moves through a maze with the above defined. parameters.
3. The robot have to reach several goals :
   (a) Find the end position ;
   (b) Find the shortest path between the cell where it started and the end position.

The definition of this parameters was done before anithing else, and is the starting point of the analysis of the problem. This rules which are the skeleton of the project, and without them it is impossible to go forward. This rules has been defined clearly for everyone to agree on important points of the project, like the things the robot should do, and also contains some restrictions compare to the general problem of the behaviour of an autonomous robot in an unknown environment, as introduced at the beginning of this document. the reason that led to these restrictions is that it is better to have first a closest goal to reach, instead of directly beginning with a too hard and large problem. It is small step by small step that a project can be managed without run out of steam or getting lost. Althrough, this project was managed step by step, and these steps was decided since the beginning.

**The first step** consists in deciding the way the project will be managed, and what is the best environment to work on it together. The main questions are : what kind of robot will be used ? With wich language will the sofware be written ? How the work will be split between the group members ?

**The second step** is the theorical analysis. In this part, the main activity is to read documents on the work which already exists on the subject, find the right algorithms, and then go deeply in their understanding. This step could have be the first one, but the decision was made to first look at the environment and the tools we will use, in order to have more precise ideas of what we have to read, and be able to contrust a sketch of coding in our minds while reading all the papers. Furthermore, the first step helped to knows which part of the problem each member of the group is the most interested in, and then better separate the research work.

**The third step** is the software analysis, a step to decide how the software embedded in the robot will be organized. Here it is an embedded system, so tasks have to be defined with their general behaviour, and the way they will intercommunicate.

**The fourth step** is the implementation. First doing small tests on the hardware, in order to be more confident with our working environement, the robot, its sensors and acuators. Then it is finally time to write the first version of the implementation with the algorithms and the organization defined in the previous steps. Tests has to be permormed oftenly independantly on the different parts of the software, before to merge all the working parts to make a complete program. It is a very critical step, because it is when all the previous decisions are tested in real conditions, and sometimes things can not work as expected, and some difficulties can be endure. So here the team work is more important than ever, and it's why regular meetings have to be schedule, to review the work done, and decide together how to go through the difficulties. It is also possible to go back to the previous steps and come back to the implementation as much as needed.

**The (optional) fifth step** can be to generalize the problem, for example by using a maze where walls are not all straights, or in a real room with several kinds of objects (different sizes and shapes) as walls, or also by not specify the dimensions of the environment to map. In brief, this step can be done if all the previous steps had been done and the deadline is not reach, and consists in remove the restrictions one by one to have a more generic and smarter robot.

# Chapter 3

# Working environment

## 3.1 The Lego robot

At the beginning, the robot has two wheels and one sensor. This architecture can be modified later in the project after several tests to better solve the maze. In order to realize this project, we will use as main controller a NXT Micro Computer brick. This brick will communicate with other devices from LEGO to map the environment and to find the shortest path from the start point to end point. The other components could be distance sensors to find where is the wall, the color sensor to detect if it arrived in the right position which is the end, the motors to move the robot and to rotate the distance sensor for a better understanding of the environment.

## 3.2 The maze

## 3.3 NxtOsek and C

After we decide on which components we use to create the robot we had also to decide what software platforms we have to use. To program this device we will use nxtOSEK, which is a Real-Time Operating System for Lego Mindstorms programmable NXT controller. To edit the source code we can use the software gedit, which is default installed in the Ubuntu operating system and as a compiler we use gnu-arm compiler.

# Chapter 4

# Group Work

How we worked in group, planned and manage meetings, and divided work in different tasks.

## 4.1 A repository on Github

# Part II

# Problem Analysis

# Chapter 5

# The robot behaviour through the maze

# Chapter 6

# A real time system

## 6.1 Definition of tasks

## 6.2 Scheduling

# Chapter 7

# Response time analysis

## 7.1 Measurement time

## 7.2 Tasks Duration

# Chapter 8

# Needed algorithms

## 8.1 Localization

## 8.2 Mapping

## 8.3 Movement

## 8.4 Path planning

# Part III

# Implementation

# Chapter 9

# Different modules for different tasks

# Chapter 10

# Shared Variables

# Chapter 11

# Tasks Implementation

# Part IV

# Conclusion

Examples of bibliography references : [Madsen, 2010], [Oetiker, 2010] and [Mittelbach, 2005].

# Bibliography

Madsen, L. (2010). Introduktion til LaTeX. http://www.imf.au.dk/system/latex/bog/.

Mittelbach, F. (2005). *The LATEX companion*. Addison-Wesley, 2. ed. edition.

Oetiker, T. (2010). The not so short a introduction to LaTeX2e. http://tobi.oetiker.ch/lshort/lshort.pdf.

# Appendix A

# Appendix A name

Here is the first appendix