**Software design**

**Team project – Deliverable 1**

**Team number**: <Your group number on Canvas>

**Team members:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Student Nr.** | **E-mail** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

This document has a maximum length of 10 pages.

**April 2013**

# Table of Contents

[Table of Contents 2](#_Toc505179048)

[1. Introduction 3](#_Toc505179049)

[2. Requirements Specification 4](#_Toc505179050)

[2.1 Requirements 4](#_Toc505179051)

[2.2 Use Cases 4](#_Toc505179052)

[3. Implementation remarks 5](#_Toc505179053)

[4. References 5](#_Toc505179054)

# 1. Introduction

**Author(s)**: <name of the team member(s) responsible for this section>

Write a short description of your version of the ROVU system that you will design here. Clearly specify which are the key aspects of your system, such as:

* obstacle avoidance strategies,
* extensions to the base mission provided in the team project guide,
* presence of a central station orchestrating all the robots, if needed,
* decide whether your environment is fully known at the beginning of the mission (so the robots can actually plan their trips), or the environment is unknown (so the robot can encounter unpredicted obstacles), etc.,
* number of involved robots in the missions (maybe your system can be totally independent from the number of robots),
* Types of considered robots, for example you may have a special fast robot only for mapping all the obstacles in the environment, and then “dummy” robots without sensors that take photos only, or you may have always the same type of robots, etc.,

Be creative here!

Don’t forget to mention your references (e.g., to known obstacle avoidance algorithms like this: <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>), if applicable.   
  
Recommended amount of pages: 2-3

Introduction

In this introduction we will give a short description about the implementation of our ROVU system. In ROVU, robots have the mission to cover as much as possible of the area they reside in. With cover we mean that if we divide the area to be monitored in a grid of points, the robots need to make sure that of each point one or more photos are taken. We will define the key aspects of our ROVU system that makes it possible to achieve this goal.

Environment & System

Since environments can change, we decided to make the environment unknown to our ROVU system. With an unknown environment, we need proper obstacle avoidance strategies for our robots and a way to map the environment when obstacles are found. To achieve this, our ROVU system consists of at least one and at most 4 robots and one central station. The mission consists of 2 phases:

1. Map the environment to get a view of the obstacles and possible coverage of the environment.
2. Cover all the possible points in the environment.

Robots

There is only one type of robot and each is equipped with sonar sensors to detect obstacles and a camera to make pictures of grid points it passes. The behavior of the robot will be as follows. A robot will always move straight forward unless an object is detected in front of the robot. When that happens, the robot will turn 90 degrees right and move straight forward again. When a passageway on the left is found, the robot will turn left and move forward again [1]. When a wall is found the robot will turn right again and looks for a passageway on the left. With this algorithm a robot can get stuck in a cycle. For example, exploring the outer walls of the environment. When this happens, the central station will intervene in the behavior of the robot. Beside sensing obstacles and making pictures, a robot is capable of sending and receiving messages of the central station. A robot sends it location every time frame and can get specific orders from the central station which it must obey.

Central station

The central station can be seen as an all-seeing eye watching over the robots. The central station sees the environment as a grid of points that must be covered by one of the robots. Each grid point is assigned a certain status to keep track of the complete state of the environment. These statuses are:

1. Unknown. Zero knowledge of this grid point.
2. Accessible. This means that a sonar sensor of a robot has covered this grid point and did not find any obstacle.
3. Obstacle. This means a grid point is identified as an obstacle.
4. Covered. This is a grid point covered and photographed by a robot

When all unknown grids are identified the central station can give orders to the robots to cover the accessible grid points. An order to a robot can be to ignore his “normal” behavior. This way the robot can get out of a cycle. For example, when its circling around an object. Another order could be directing a robot to a specific grid point that has not been covered yet.

When the only statuses of grid points consist of obstacle or covered, the mission is complete and the central station will order the robots to get back to their starting position.

*Requirements and Implementation remarks*

In section 2 we will elaborate on the requirements specification. In section 3 we will give the remarks of the base implementation of the robotic system.

# 2. Requirements Specification

**Author(s)**: <name of the team member(s) responsible for this section>

This chapter contains the specification and UML representations of all the requirements. The chapter is sectioned in the sections below.

Recommended amount of pages for the whole chapter (including also sections 2.1 and 2.2): 6-8

## 2.1 Requirements

**Functional requirements**

|  |  |  |
| --- | --- | --- |
| **#** | **Short Name** | **Description** |
| F1 | Obstacle avoidance | The rovers shall move freely in the environment and avoid obstacles autonomously. |
| F2 |  |  |
| … |  |  |

**Non-functional requirements**

|  |  |  |
| --- | --- | --- |
| **#** | **Short Name** | **Description & reasoning** |
| NF1 | Obstacle avoidance [Performance] | Each rover shall react to the presence of an obstacle within 50 milliseconds. |
| NF2 | Obstacle avoidance [Safety] | A rover shall always be at least 1 meter from obstacles, other robots, and human beings. |
| … |  |  |

Each non-functional requirement must be tagged with the corresponding category (see slides 24 and 25 of the second lecture for knowing them + refer to Chapter 4.1 of the Sommerville book in Canvas).

## 2.2 Use Cases

Describe AND illustrate your system use case diagrams in this paragraph. Each use case diagram must be represented by:

* a UML use case diagram AND
* a table conforming to the Cockburn template for each use case.

<Figure representing the UML use case diagram>

|  |  |
| --- | --- |
| **Name** | Use case 1 |
| **Short description** |  |
| **Precondition** |  |
| **Postcondition** |  |
| **Error situations** |  |
| **System state in the event of an error** |  |
| **Actors** |  |
| **Trigger** |  |
| **Standard process** |  |
| **Alternative processes** |  |

# 3. Implementation remarks

**Author(s)**: <name of the team member(s) responsible for this section>

In this chapter you will elaborate on what you implemented in your base version of the robotic system.

Recommended amount of pages for this chapter: 1

# 4. References

References here.

[1]<https://www.ibm.com/developerworks/library/j-robots/>