Requirements for the 2007 Robot Project - Critical Systems Development

Rune Winther

April 13, 2007

1 Introduction

This document presents the requirement of the 2007-project in the "critical systems development" course. Briefly described, the project consists of programming a robot to solve a number of tasks as defined by our robot laboratory, in such a way that the robot is roobust with regard to single point of failures. In principle, this means that the software that is developed should be such that failure in any single component should not prevent the robot from solving its tasks. In practice, the required fault tolerance will be less strict.

The background for the tasks the robot will face is the Robocup competetion, arranged annually in Copenhagen. Since the Robocup tasks only change minimally from one year to another, we have built a sligthtly minimized version of last years Robocup challenge in our own robot laboratory. The group that has the best performing robot will be allowed to represent Østfold University College at the 2007 Robocup competetion. Deadline and actual criteria for being chosen will be announced later.

The most important aim of this project is to teach some of the practicalities of developing fault tolerant software, and to supplement the more theoretical sides of the course. In order to be able to develop the robot's software successfully, you will need to know which components might fail and how they might fail, i.e. you will need to perform some level of risk analysis. To make your robot fault tolerant, you will need to find ways of programming the robot so that it can complete its tasks, even in the case of component failures.

2 The robot

The robot's main parts are its *chassis* and its *electronics*.

2.1 Robot Chassis

The robot's chassis has been purchased from Lynxmotion, USA, and is of the type 4WD3. Figure1 shows the basic robot. The robot is manouvered through the use of 4 motors, one for each wheel. In the case of our robot, the motors on each side are wired together so that the robot is driven in "tank-mode". This means that the speed of the wheels on the left and right sides are controlled independently. To make the robot turn, you just have to set different speeds on the left and right sides.



Figure 1: The basic 4WD3 chassis with wheels and motors.

The motors can be driven equally fast forward and backwards, which also means that the robot can "turn on a dime", i.e. changing its heading without changing its position.

2.2 Electronics

The robot's electronics is made up from both "off-the-shelf" and self-made components. The following is a list of the robot's electronic components:

- Eyebot controller, a Motorola 68332 CPU complemented with a number of IO-options.
- Three IR distance measuring sensors. One facing forward and one on each side at 90 degrees to the robot's front.
- Forward facing camera.

- Two shaft encoders, one each side measuring the rotation of the wheels.
- Lightsensor array, positioned under the front of the robot. This array is used to follow lines.
- Motorcontroller for controlling the speed of the motors on each side.
- Servo with arm to be used for grabing a golfball.
- Pressure sensor that detects contact with an object in front of the arm controlled by the servo. Attached to this arm is also a magnet on springloaded string.

Information on how to program the robot and utilize it's sensors will be provided elsewhere.

3 The challenge

The challenge essentially consists of two parts:

- To program the robot so that it solves all defined tasks.
- To program the robot so that it is robust with regard various component failures.

3.1 The defined tasks

As stated in Chapter 1, the tasks are based on the Robocup 2006 competetion. For the sake of this course a few of the 2006 tasks will be ignored.

The layout of the tasks is illustrated in Figure 3.1, where the numbers refer to the different tasks. All the tasks are described in Table 3.1 below.

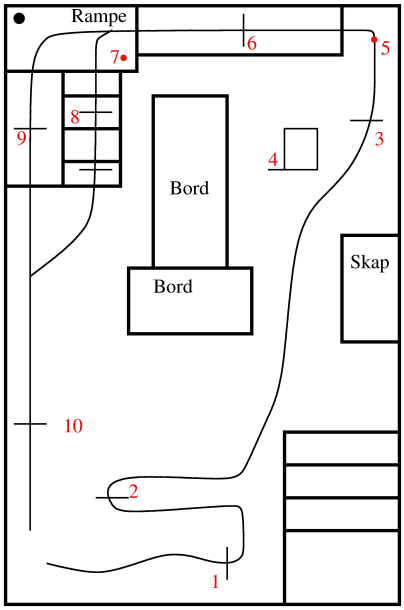


Figure 2: Schematic layout of the tasks.

Task #	**Description**	**In project?**
1	Passing the gate within a time limit of 30 seconds	Yes
2	Drive through the gate.	Yes
3	Drive through gate.	Yes
4	Open the tunnel and drive through.	Optional
5	Find the golfball and put it in the hole on top of the ramp.	Optional
6	Drive up the ramp, through the gate.	Yes
7	Find the second golf ball and put it in the hole on the ramp.	Optional
8	Drive down the stairs	Choose 8, 9 or both
9	For extra points, also drive through the gate on the slope down from the ramp.	Choose 8, 9 or both.
10	Drive through the gate.	Yes

Table 1: A brief description of the various tasks.

3.2 Required robustness

Since this project's theme is the development of critical systems, which obviously need to be robust, certain robustness requirements must be met by

the developed software. These are:

- Excluding the shaft encoders, no singel sensor failure shall prevent the robot reaching the finishline. This means that the robot should be able to handle failure of any one of the light sensors, distance sensors and camera.
- A critical[1] failure to any of its actuators[2] should be identified and the robot stopped.

Failures might occur while the robot is driving, i.e. after the robot has started.

4 Expected deliverables

There are essentially four different deliverables that are required in this project:

- A risk analysis of the robots essential hardware, i.e. sensors and actuators.
- A detailed design, including a specific description of what has been done to achieve robustness.
- The code that makes the robot solve all its tasks.
- An explicit argument giving a convincing explanation of why the developed code fulfills its requirements[3].

[1] In the meaning "impossible to complete any more tasks"
[2] The robot's actuators comprise the light emitting diodes, the motors and the servo.
[3] Functional as well as non-functional