ROBOT: Software architectures for robot programming, 2009
Henning Christiansen, 9-mar-2009

Notes on the forthcoming written assignments:

Work on a particular software architecture

1 Introductory remarks

We have now become familiar with leJOS, and most student groups are approaching the point where they know how to establish a Bluetooth connection between the NXT and a laptop. Some students actually succeeded in this at the course March 4; if you need inspiration, look at the uploads on the course's bscw server. You may also contact Donald Axel who has been so kind to help us.

Understanding the basic principles of behavioural robotics, how it is implemented, and how to communicate with the robot is important for the rest of the course.

2 The rest of the course

In the rest of the course we will mainly work on detailed smaller assignments, where each group agrees with the course teacher on a specific task. The purpose with this is not to compete on produced the most fancy, humanlike, sophisticated and fault free robot — the selection of sensors and mechanical actuators that we have available are anyhow too limited for this. These assignments should instead take the form of an investigation of a particular software architecture which you test and experiment with through the construction of a simple robot. We may extend the course curriculum if interesting articles of common interest shows up.

Student groups are expected to make presentation of the intermediate stages of their work. It is highly appreciated if the groups help each other with technical problems, as many difficult, although in principle trivial, obstacles that will arise (as you have noticed already).

It will depend on the choices of topics for the assignments, but we will try to split the work smaller assignments so that you don't have to write a report about everything the last week of the course.

We may choose totally independent assignments for 2–4 students each, but it could be also be interesting if we can define a common project to which all groups could contribute. In the latter case, we may either have each smaller group produce its particular robot for a common purpose, or we create a larger design together and split by software modules. — If it turns up that you have chosen a topic that turns out to be too easy, we can extend it along the way to include more interesting aspects.

Outline of a time schedule

March 11: Continue the fight with the communication parts. Students who have got small programs working that send Bluetooth messages back and forth, should explain how they did to the others. And we may have Donald Axel available to help us also. Secondly, we will take a longer brainstorm with your ideas, and aiming at making a decision of a common task, or otherwise which detailed task that each group should work on.
March 18: Each group will give an overview of the (sub-) task, they want to solve and which technical challenges they expect, and how to solve them; if possible, present the overall structure of the software architecture that you aim at.
March 25 at latest: You teacher gives exact deadlines and requirements for the assignments to be given in.
The following weeks: Working on the assignments. We work together in the same room each Wednesday. Each group may give short statement of how fare they got and which problems they have been facing.
April 11: Each group presents and demonstrates its solution, and if we are working on a common project, we demonstrate the entire cooperating machinery :)

3 Possible topics for the assignments

Here are some possible ideas that you may elaborate on, but you are also more than welcome to search for and suggest other ideas. The proposals are more and less detailed, so you may also take them as patterns that you may apply to other tasks or architectures.

Producing a map of a room
The Lego robots have quite restricting limitations concerning orientation. They have no sense of where they are and no idea of the direction. However, within shorter periods of time, it is possible to get some approximate knowledge about this by keeping track of how far each motor has moved, but after a while, the accumulated inaccuracy (sliding on the floor, etc) has grown to far. So how can we make reliable maps?

Well, here is an idea: You may use the ultrasound sensor as a kind of radar; it may rotate on top of the robot or the robot can rotate as a whole. The sensor can measure if it points towards an object, and it can register the distance to an object, although with some inaccuracy. For each rotation of the "radar" (let us assume it can rotate 120 degrees and then must rotate back and so forth), it can register a vector of readings,

<object, distance, angle relative to robots orientation>

As the robot moves along, it gets a long series of such vectors, and with a bit of programming it should be possible to assemble a 2D map from these vectors. Furthermore, from the part of a map that the robot has learned at a given time, it should be able (sometimes) to recognize where it is at a given moment.

This task may be implemented by a client-server architecture where a laptop takes care of assembling the map from the vectors and of commanding the robot to search in regions of the room where it has not been before. The robot itself should have some sort of behavioural program that makes it possible for it to move around, avoid obstacles etc.

while also being able to take the servers directives into account. Sending messages back and forth is an important part of getting this to work.

With more students in the project, it may also be interesting to have some of you make a graphical interface so that the map under construction is displayed continually with more and more details.

Behavioural programming on your own computer?
The examples and exercise we have worked with until assumes that the behavioural program resides on and is executed onboard the robot. However, we could also reduce the robot to a dumb slave which only executes instructions received from the server and sends sensor data back. It seems possible that you can use Java (with leJOS) on the server and program the robot's dumb program in either leJOS or Lego's graphical language (or try both). What are the advantages and disadvantages of this shift of where control is performed?

Cooperating robots
A very challenging theme is to have robots to cooperate on a task. We may suggest that each robot is running its own behaviours that are designed in such a way that each robot will do what it can to synchronize its actual behaviour with its companion(s). The task may be more or less advanced; here are some ideas:
- Make two fork-lift truck robots that try to locate a special item that is too heavy or too long for one robot to lift; when they have found it, they lift it and move it together (your teacher has no idea how this can be done with the primitive sensors that we have available, but you can probably learn a lot from producing even a solution that does not work).
- A multi-robot version of the room-mapping problem. You may experiment changing how much synchronization is done robot-to-robot, how much by a server, or perhaps by behaviours without explicit synchronization that just happen to sum the right way.

Traffic simulation using robots — is an instance of the cooperating robots, but requires many robots and can benefit from having several groups working on the same project.
- **Motorway driving.** Robots are moving along parallel lanes, they may have different preferences for speed, and they may shift lane for overtaking or for giving room for a faster vehicle. We can imagine robots that resemble different profiles of vehicles/drivers. Equip each robot with behaviours, find a place that you declare is a motorway and set the robots free!
- **Motor race.** It is a bit like the one above, but here we must combine cooperative and competitive behaviours. The goal of each one is to drive faster that the others, e.g., watching for possibilities to overtake the guy in front in case he as too much speed when entering a turn. On the other hand, there are things you must not do even in a motor race!

Simulating social behaviour as an exercise in virtual archaeology — is an instance of the cooperating robots related to the above. You can imagine the robot playing the role of inhabitants of a village, and each has a task to do, e.g., some goes on shopping and others are traders on the market. A robot may great other robots that it meets on its way, etc., and should avoid bumping into other robots. (Due to the limited sensor capabilities, this may be too difficult, but now the idea is given).