

The Design and Implementation of The Disqo Dawg

Lars Ailo Bongo,
Anders Åberg,
Are Rikardsen

D-340, Fall 2001
University of Tromsø

Background

This project was a mandatory assignment in D-340, Fall 2001, at the University of Tromsø. Below is a link to the assignment description:

http://www.cs.uit.no/studier/kurs/d340/info/2001h/Projects/mandatory_assignment_chep_sheep_robots.htm

Introduction

In this paper we will present how we designed and implemented a program for the dog, and a program for the owner. Both programs are multithreaded. The dogs code is written in C and runs on legOS. The notebooks code is written in C using pthreads. We will describe how we have used mutexes, semaphores, monitors, barriers and message passing for synchronization and communication. We will also describe our simple but effective algorithm to find the sheep and push it back to our base.

The Lego Robot

Design goals:

- High speed.
- When driving forward (searching), should detect obstacles.
- When driving backward (destroying), should have high possibility of hitting sheep.
- Not easily tangled into other robots.

RCX Software

We decided to make the dog independent of the owner. Since our find-sheep, push-sheep algorithm is so simple the dog will itself compute its long-term goals and execute them.

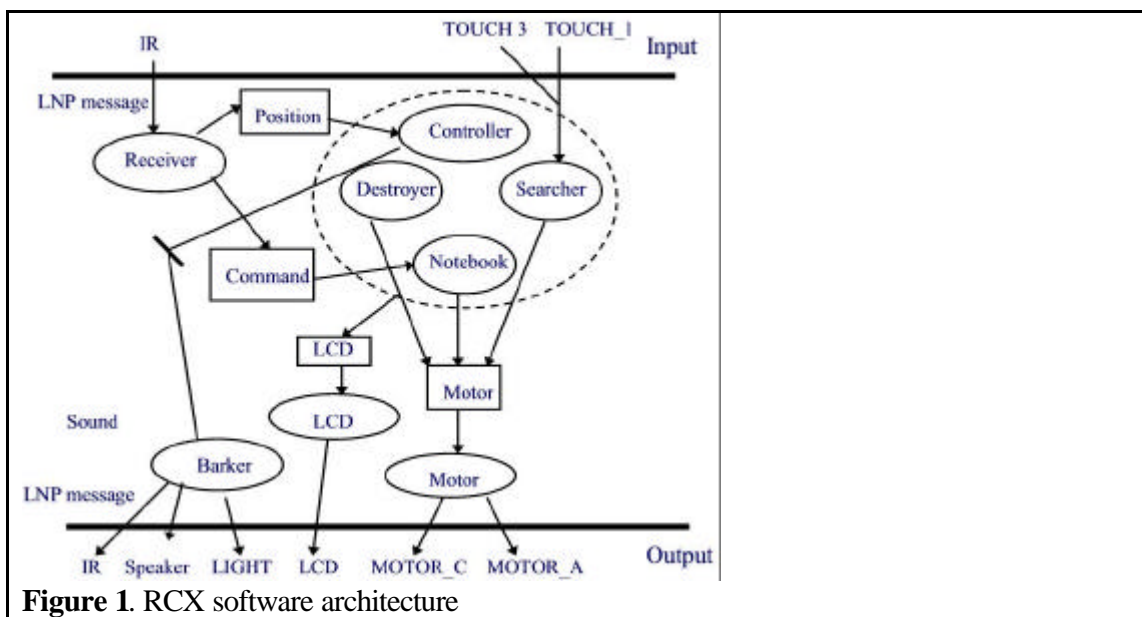


Figure 1. RCX software architecture

In figure 1 the circles represents threads and the rectangles represent channels. Above the upper line are the input devices we have used, and below the lower line are the output devices. The arrows going into a channel represent sends, and the arrows going out of a channel represent receives. All synchronization and communication between the Controller and the I/O threads is by message passing. The dotted circle around Controller, Searcher¹, Destroyer and Notebook represents their tight coupling. The Line between Controller and Barker is a shared counter barrier.

Channels and Condition Variables

The channels are implemented using bounded buffer monitors. To implement a monitor we had to implement condition variables using semaphores. We have also implemented mutexes using semaphores. The channel support two different sends; one blocks when the buffer is full, the other overwrites the oldest item if the buffer is full. We need the non-blocking, send since we have real-time threads that should not be blocked for a long period of time (e.g. Controller), and since we often are interested only in the last sent message (e.g. position information). There is only one receive, that blocks when the buffer is empty. All message passing is asynchronous.

Threads

Receiver sends lnp packets copied to a shared buffer by the packet handler. Since the packet handler cannot block we could not use our usual synchronization primitives. Instead we used our knowledge about the legOS scheduler. A description can be found in the code.

The LCD thread receives strings and writes them to the display. It works like the printer spooler in Unix.

The motor thread receives motor commands from the motor channel and executes them. A motor command can be simple as “go forward” or more complex like “turn 90 degrees to left²”. The motor channel can only contain one motor command, since we are only interested in the most recent command. Controller will ensure that two threads don’t send motor commands interleaved.

Controller, Searcher and Destroyer are used to drive the robot. Controller is a scheduler that schedules Searcher and Destroyer. These threads are schedules using shared variables and condition variables. They communicate by using shared variables.

Notebook is a thread that allows the notebook to take control of the robot. It receives command packets from the notebook. The owner uses these packets to remotely control the dog.

¹ Searcher and Destroyer are named after Iggy Pop’s Search and Destroy. We had programmed the dog to sing I Wanna Be Your Dog when driving around, but we didn’t have enough CPU cycles to spare.

² We never use such complex motor commands.

Barker barks. Controller schedules it using a shared counter barrier. This must be done since Controller receives position packets that tell when Barker can bark.

Controller, Searcher and Destroyer Algorithms

The algorithm has two overall goals:

1. Get into a position where the sheep can be pushed into home base. This is Searchers task. This area is the red area in figure 2, and is called Destroyer area.
2. Push the sheep to home base. This is Destroyers task.

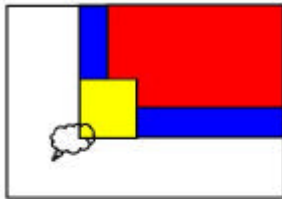


Figure 2

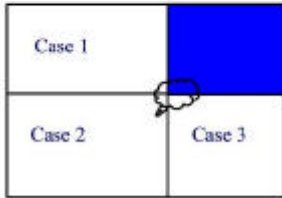


Figure 3

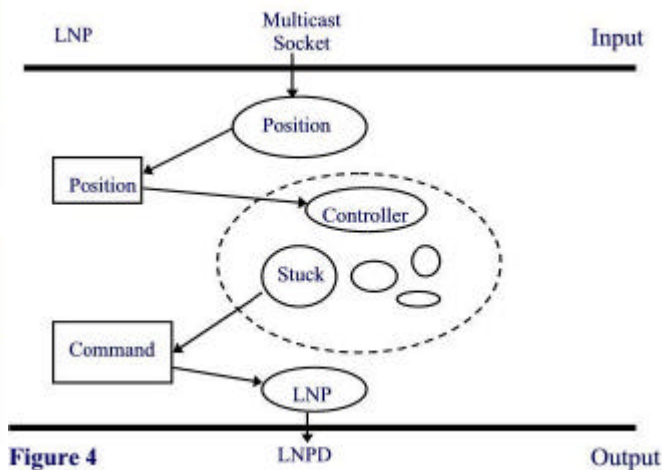


Figure 4

Searcher can be in one of the 4 areas in figure 3. In case 1 it drives east. In case 2 it drives northeast. In case 3 it drives north. This will lead the robot into Destroyer area. When the fourth case (blue area) occurs, the dog enters Destroyer area. Controller will schedule Searcher out and Destroyer in. Lets assume that dog is inside Destroyer area (red) in figure 2. If Destroyer is inside the red area it drives southwest. If it is outside the red area and inside the blue area, it drives back into the red area. If it is outside the blue area, Searcher will take over. This will lead the robot to the lower left corner of the rectangle. When the robot is inside the yellow area in figure 2, we increase Destroyer area. Finally the sheep will be in the lower left corner of Destroyer area, and so will the dog.

Notebook Software

Figure 4 shows the notebook software architecture. We can see that Position receives position information from the Control Tower, and LNP sends command packets to the RCX using lnpd.

Controller keeps a history of the latest N position packets. Currently only dogs and sheep's history are recorded, and only dogs history is used. Controller uses the history to find situations that the dog cannot handle. We have only implemented one such situation; when the dog is stuck. If this happens Controller forks off a Stuck thread that will take control over the robot and get it out of the situation. It does this by sending command packets to RCX's Notebook thread as explained above.