

4B-Dog.vi
 Fiamma:Users:andy:Professional:UW Design:Art 483, Winter 2007:Robot Exercises:4B-Dog.vi
 Last modified on 12/13/06 at 8:31 PM
 Printed on 12/13/06 at 8:53 PM

The Dog [B]

- Port 1: touch sensor
- Port 2: light sensor
- Port 3: sound sensor
- Port 4: distance sensor

- Port A: motor
- Port B: lamp
- Port C: motor

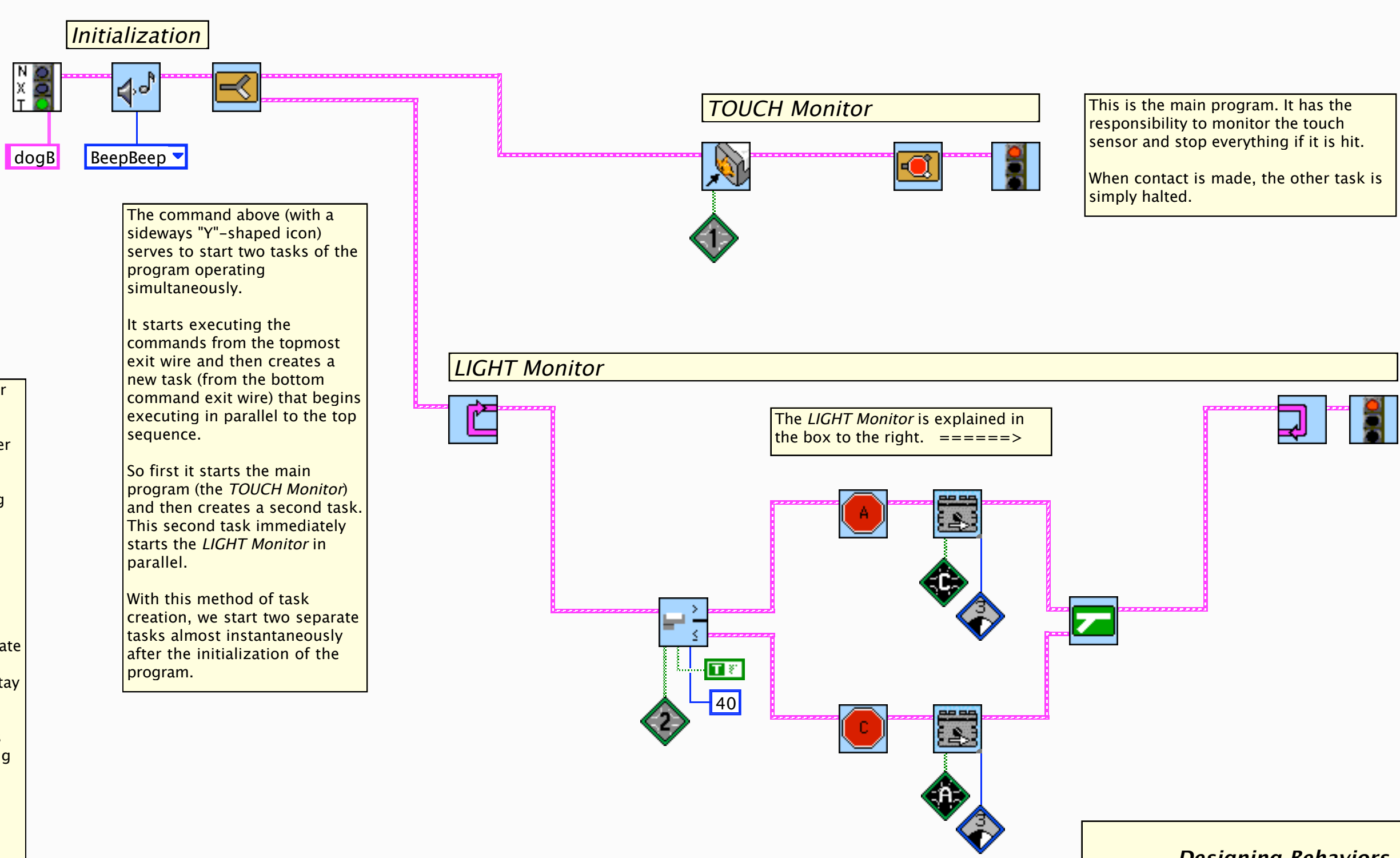
This robot crudely models the behavior of a dog. Being a canine, it is a highly sentient, intelligent, and perceptive creature, of course. Our dog is a hunter and tracker.

This dog's mission is to find something ahead of it in the bush. Since we don't have smell sensors for this robot, instead we'll have him follow a visual trail on the ground—a black line on a light background.

The robot does this by using a light sensor pointed at the ground to evaluate reflected light and determine its intensity, and then steering itself to stay on the black line.

In addition to this activity, the robot is simultaneously doing another: stopping if it makes contact with anything.

In this program, the robot has two separately executing tasks, operating simultaneously to accomplish each of the separate goals. Each task continuously monitors one of the input sensors, and takes appropriate action when something is detected by that sensor.



The command above (with a sideways "Y"-shaped icon) serves to start two tasks of the program operating simultaneously.

It starts executing the commands from the topmost exit wire and then creates a new task (from the bottom command exit wire) that begins executing in parallel to the top sequence.

So first it starts the main program (the *TOUCH Monitor*) and then creates a second task. This second task immediately starts the *LIGHT Monitor* in parallel.

With this method of task creation, we start two separate tasks almost instantaneously after the initialization of the program.

Designing Behaviors
 Fundamentals of Interface Design
 UW Division of Design
 ART 483, Winter 2007
 Davidson/Roesler

LIGHT Monitor

This task has the responsibility to monitor the light sensor and direct the robot to follow a black line. It runs continuously until stopped by the main program.

The following "pseudo-code" summarizes the method in a textual form of logic.

```
-- follow a black line on the ground
loop forever
  -- check the light sensor value
  if (light is bright) then
    -- off the track, turn left
    stop left motor
    run right motor
  else
    -- on the track, turn right
    stop right motor
    run left motor
  endif -- light sensor check
```

The method is to continuously check the light sensor and evaluate the intensity of the reflected light by comparing its value to a threshold level. If the sensor detects a value above the threshold (here we set it to 40% brightness, but this can be changed depending the specific environment), then it is assumed to be white and the robot is off the track. If it is 40% or less, then it is assumed to be black and on the track.

If the robot is on the track, it turns a little to the right and continues forward. If it is off the track, it instead turns a little to the left and proceeds forward also. Since it is doing this continuously, in this way it inches forward (at a given speed, also modifiable according to taste), constantly adjusting its course to the left or right to stay on the black line.

This scheme seems like magic, but it actually works. There is a flaw in the logic which you will undoubtedly discover if you give the robot a complicated track to follow, but it works especially well for a circular track. Improving the scheme to follow a more general track is, as math professors love to say, left as an exercise for the student.