RoboCup Rescue

EV3 Workshop

Part 3

Rescue Analysis with RobotC

Program Analysis of Rescue

- RoboCup rescue requires more than Line Following
- Extra challenges include, Obstacles and rescuing the Victim
- When solving these challenges do them as individual programs and then add them to the main program when they work successfully
- Your robot will need to do multiple things at once in order to follow the course AND be on the look out for Obstacles and Chemical spill etc
- Your robot will need to change modes when it detects the Obstacle
- This process is called Task Swapping or Multitasking

Using multiple tasks() to multitask

- We now use a number of tasks() so that the robot is always looking for the challenges set by RoboCup Rescue
- The task main() will have the main switch case control structure which will swap the tasks around as needed
- We then need a separate task() that looks for the Obstacle, another
- To look out for the Chemical Spill etc.
- We also need to detect intersections, chemical spills etc, etc
- Do we do these in tasks OR????

Write small problems as Functions

- In EV3 we created MyBlocks for all the individual processes that were required for the robot to complete a Rescue
- We need one for Line Following and
- One for Getting around the Obstacle
- One for Finding the Rescue Capsule and
- One for picking up the Capsule and putting it down, another
- One for Getting out of the Chemical Spill etc.
- We then call each as needed from with task main() using a Switch control structure.
- Wait! Is there a difference between tasks and functions????

Combining tasks() and Functions

- Tasks can be used to allow the robot to be constantly checking on its environment and change/adapt as necessary
- Functions are individual processes the robot needs to do, once it has decided which task to run due to a change in the environment
- So by using a combination of tasks() and Functions we can solve the RoboCup Rescue challenge

- Lets make a list of everything the robot needs to do

RoboCup Rescue Problems

- Main Control task
- Line Following
  - Follow the line
  - Detect Intersections
  - Navigate Intersections correctly
  - Navigate Gridlock
  - Detect Chemical Spill
- Obstacle
  - Detect the obstacle
  - Navigate around the obstacle
- Chemical Spill
  - Find the can
  - Don’t find the contaminated can
  - Control/Pick up the can
  - Push the can out and release it, or
  - Find the Block
  - Put the can on the Block
  - Find the Exit and then the line
- Possible complications
  - Find the can on an extended spill
  - Find the exit on an extended spill
  - Challenge tile

Which are Tasks?

- Main Control Task
- Line Following
  - Follow the line
  - Detect Intersections
  - Navigate Intersections correctly
  - Navigate Gridlock
- Obstacle
  - Detect the obstacle
  - Navigate around the obstacle

Which are Functions?

- Chemical Spill
  - Detect Chemical Spill
  - Find the can
  - Don’t find the contaminated can
  - Control/Pick up the can
  - Push the can out and release it, or
  - Find the Block put can on Block
  - Find the Exit and then the line
- Possible complications
  - Find the can on an extended spill
  - Find the exit on an extended spill
  - Challenge tile

So How do we Start and Stop tasks?

- In RoboCup Rescue we want to Follow the Line and then change the task when we detect the Obstacle. After getting around the Obstacle, we then want the robot to go back to Line Following. How?
- We need to run optional tasks within the main task containing all the Functions like, Line Following and getting around the Obstacle etc.
- A Parallel task is constantly looking for the Obstacle etc.
- When it Detects the Obstacle, for example, it tells the Main Task to stop what it is doing (usually Line Following) and change to the appropriate task, eg Get around the Obstacle.
- *Note: RobotC has a limit of 10 simultaneously running tasks*

Switch Case – CaseWhere Control Structure

- This example Switch control shows how we can have tasks running simultaneously so that the robot is Line Following the course while looking for the Obstacle.
- We can also see the Function pointers that would contain the Line Following, Obstacle Navigation code
- If these Functions are solved in a completely separate program they can then be copied and pasted into this program. Solve the small problems, then add them to the big problem.

```
1  int taskNo = 1;
2
3  void lineFollow()
4  {
5    //Line Following Code
6  }
7  void obstacleNavigation()
8  {
9    //Navigate around Obstacle Code
10 }
11 task detectObstacle()
12 {
13   while (true)
14   {
15     if (getTouchValue(S2)) //IF touch sensor on port 2 is bumped
16     {
17       taskNo = 2; // Switch to the Obstacle Navigation
18     }
19     else
20     {
21       taskNo = 1; // Keep Line Following
22     }
23   }
24 }
25 task main()
26 {
27   startTask(detectObstacle); //Starts the task detectObstacle
28   // it then constantly looks for
29   while (true)                // the obstacle
30   {
31     switch (taskNo) // has been initalied to taskNo 1
32     {
33     case 1:
34       lineFollow(); // Runs the Line Follow Function
35       break;
36
37     case 2:
38       obstacleNavigation(); // Runs the Obstacle Navigation Function
39       taskNo = 1;    // switches back to the Line Following taskNo 1
40       break;
41
42     default:
43       //Stop robot
44     }
45   }
46 }
```

Pseudocode for Rescue - Example

Main program

Initialise taskNo = 1

Main task()

While programRunning

CaseWhere TaskNo

Case 1:
LineFollowing
Case 2:
ObstacleNavigation
Case 3:
Rescue
EndCase

Tasks

DetectObstacle()
Look for the Obstacle. When found set taskNo to 2 *(ObstacleNavigation)*

DetectChemicalSpill()

Look for Silver indicating the Chemical Spill. When found set taskNo to 3 *(Rescue mode).* After FindTheExit set taskNo back to 1 *(Line Follow mode)*

Sub lineFollow
LineFollowing program
endSub

Sub ObstacleNavigation
ObstacleNavigation program
endSub

Rescue programs

Sub FindTheCan
FindTheCan program
endSub

Sub PickUpCan
PickUpTheCan program
endSub

Sub FindTheExit
FindTheExit program
endSub