MAP 299/499
Bluetooth communications with C
Professor Henry Walker

BLOCKING V.S NON-BLOCKING MOVEMENTS IN MyroC 2.1

Nicolas Knoebber, Jason Liu

GRINNELL COLLEGE

Introduction

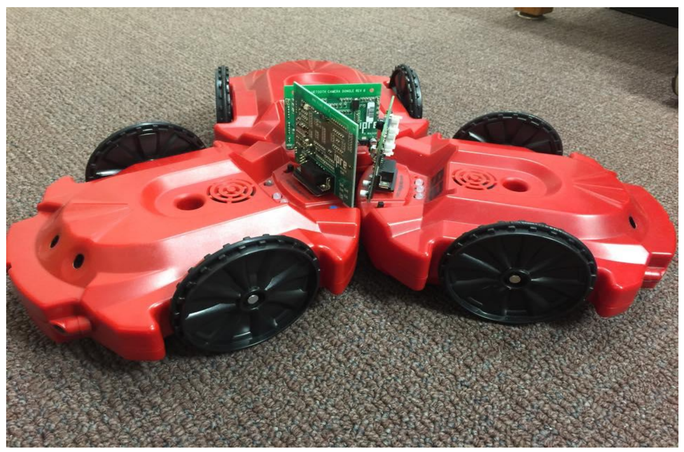
Since 2011, a student-faculty team at Grinnell College has developed and refined a complete course package, called MyroC, comprising a C-based infrastructure, course materials, laboratory exercises, problem sets, projects, etc. for introductory computer science. Using a lab-based pedagogy, the course emphasizes imperative problem solving and utilizes the control of robots as an integrative application theme.

Although this approach has had success, forced sequential processing, called blocking, has been a limitation of MyroC. Our research focuses on non-blocking alternatives for the timed robot-motion commands, and concurrent control for the movement of multiple robots.

The Problem

The original MyroC largely enforced sequential execution on a single robot. Programs connected to a specified robot, and motion commands included a time parameter to block program execution until the command finished. Although blocking provided beginners with a simple execution model, blocking also limited the variety of robot's actions.

- moving forward and beeping simultaneously was impossible with timed functions
- users could not control more than one robot at once



Three robots start together before moving concurrently away

Solution

The recent redesign maintains the simple interface, but blocking information is encoded into the time parameter: A negative time parameter signifies a non-blocking robot command, whereas a non-negative parameter blocks. Before implementing this however, we changed how rMove( ) worked. Previously rMove( ) handled all movement, but had an unintuitive implementation based on translation and rotation.

We replaced rMove( ) with rMotors( ), which takes a left speed and a right speed. With this in place, we could easily implement non-blocking options in all of the timed commands. This allows for more complex programs, without making it harder for beginners to understand.

Sample implementation code

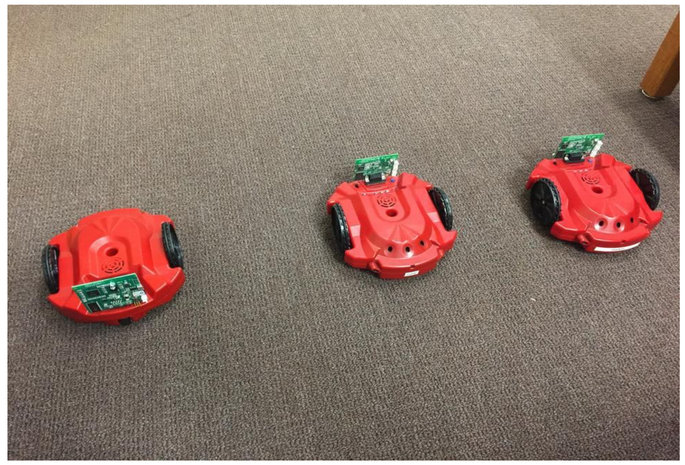
```
/**
* moves Scribbler forward for a specified time and speed
* @param speed the rate at which the robot should move
*       forward
* linear range:
*      -1.0 specifies move backward at full speed
*       0.0  specifies no forward/backward movement
*       1.0  specifies move forward at full speed
* @param time specifies the duration of movement
* if negative:
*    robot continues to move forward until given another
*    motion command or disconnected (non-blocking)
* if nonnegative:
*    robot moves forward for the given duration, in
*      seconds
*
*/

void rForward (double speed, double time)
{
  if(time < 0) //non-blocking condition
    rMotors(speed,speed);
  else
  {
    //convert motion period from milliseconds to seconds
    int utime = (int)(time * 1000000);
    rMotors(speed, speed);
    usleep(utime);
    rMotors(0.0,0.0); //terminate the movement
  }
}
```

Outcomes

Based on the new MyroC, a robot can beep and move at the same time with help of negative time parameter.

As an additional capability, recent refinements allow students at one workstation to control the motions of multiple robots simultaneously. Normally, a workstation sends commands to one robot. Now, a single command allows control to switch between several robots. Combined with blocking and non-blocking commands, students now have opportunities to explore complex programming techniques involving independent and interacting robots.



Three robots start together before doing a solar system simulation

Acknowledgements

We want to thank Vasilisa Bashlovkina and Anita DeWitt for their cooperation in doing this research project. We also want to show our appreciation to Professor Walker for providing instructions and guidances. Finally, we would thank the previous groups that have contributed to MyroC. Their notes, papers and posters paved way to our results.