CSC 714 Project progress report 1

Object Transportation System Using LEGO Mindstorms RCX

Vishnu Dasa Harish Babu (vdasaha@ncsu.edu)
David Boyuka (daboyuka@ncsu.edu)

Project URL: http://www4.ncsu.edu/~vdasaha/rtcs.htm

Progress on tasks

Note: “Common” means the task is for both of us.

Background research tasks:

- **Learn brickOS, run example programs for practice** (Common): **Completed**
  - We successfully learned the basic functioning of brickOS. We were able to run demo programs related to motors and sensors (light and touch) on the intermediate prototype which we have built.
- **Look at brickOS kernel and determine where we will need to make modifications** (Common): **Removed**
  - After further consideration, we believe that the brickOS kernel will be acceptable for our project. It uses a static-priority clock-driven scheduling algorithm, which should give us sufficient real-time control. The scheduling of physical tasks (movement, sensor reading) can be done at the user level.

Algorithm design tasks:

- Come up with some strategies for meeting deadlines more efficiently than an extreme-worst-case bound approach (factoring in current position, etc.) (Common): **In Progress**
  - I (David) did some preliminary mathematical analysis of a basic strategy: driving around in a circular loop, checking each bin along the way. I produced some equations that a bit tricky to use, but correct I think. Next we will try a form of LLF scheduling, and see how that compares.

Physical construction:

- **Setup the tracks and the layout** (Vishnu): **Completed**
- **Build input/output bins** (David): **In Progress**
  - We’re still experimenting with different mechanisms to dump the bins into the train. We think we have a good option now (two arms on the train come down and hit a horizontal bar attached to the input bin, dumping it). We will build it next.
- **Construct the basic delivery vehicle (with pickup/dropoff mechanisms)** (Vishnu): **In Progress**
  - The vehicle itself is complete, as well as the dropoff mechanism. The pickup mechanism is in progress. It must be integrated with the input bin mechanisms.
- **Add bin station sensor to vehicle (for sensing when we arrive)** (David): **In Progress**
  - We tried using a light sensor to detect white patches on the rails, but it wasn’t sensitive enough and was subject to room lighting conditions. We also tried a

touch sensor against objects on the rails, which we think is promising, but we are still tweaking it.

- **Add object pickup sensor (for sensing if objects are present in a bin)** (David): **To Do**
    - We plan to use a light sensor for this, although with our previous experience (see above), we may have to examine some other options (some sort of rotating bar mechanism attached to a rotation sensor, as are used to count people in subways)

Programming Tasks:

- **Code the motion routines (vehicle movement, detecting current position)** (Common): **Completed**
- **Code the loading/unloading routines for moving objects** (Vishnu): **Completed**
- **Code the real-time scheduler to control the above two modules** (David): **To Do**
    - This is pending finalizing the algorithms to use.

Testing tasks:

- **Come up with (physical) test cases, both feasible and infeasible** (David): **To Do**
- **Run the tests (will record video and analyze later, as testing is time-sensitive)** (Common): **To Do**

Open issues

Dealing with sensors: Dealing with light and touch sensors is proving to be more trickier than we thought initially. There are limitations in sensitivity in a couple of the sensors we used for testing. We will need to calibrate them or figure a way to use them in a more clever manner. Also, we determined one our touch sensors was faulty, but we had extras so it wasn't a blocking issue.

Weekly Milestones

Week 1 (11/13 - 11/19):

- Build input/output bins
- Construct delivery vehicle pickup mechanisms
- Add bin station sensor to vehicle

Week 2 (11/20 - 11/26):

- Add object pickup sensor
- Algorithm design
- Code the real-time scheduler

Week 3 (11/27 - 12/3):

- Come up with (physical) test cases, both feasible and infeasible
- Run the tests