CPE 470/670 Autonomous Mobile Robots

Ball Sorter Robot Contest
Team 6

Omid Tutakhil,
Sandeep Mathew,
Jin Jiang

Hardware and Software

The hardware design was based on the one provided in the instruction manual provided with the NxT Kit. The team modified the ball kicking arm to one suitable for picking up balls and throwing it appropriately. In order to make the robot move fast on the edges 2 plastic tires were added on sides of the robot, this ensured that there is less friction when moving along the edges.

For the software the team used a sequential design initially and made sure that basic steps worked properly. After the basic steps were done, the following enhancements were made

(a) Use of feedback from the compass to make the robot move straight
(b) Addition of timeouts to ensure that robot gets itself unstuck while turning
(c) A default action to be performed when the robot is completely stuck .

One factor that was not considered in the initial design is the 2 minute time frame provided for the contest, the focus was to collect as many balls as possible without giving consideration for the 2 minute time frame provided. The main sequential algorithm is as follows

<u>Main Sequential Algorithm</u>

1) Move towards the edge and turn towards ball
2) Move till you find a dark spot
3) Move little backwards
4) Move the kicker down
5) Move forward for an arbitrary amount of time
6) check if you have a blue ball
    if yes
        pick ball up
        move backwards till you are off dark
        turn around 0
        move till you hit the other black spot
        drop the ball there
        turn around and repeat from step 1
7) check if you have a red ball
    if yes , then
        move the kicker upwards
        move till you hit the wall
        drop the red ball
        move back till you are off dark

turn east
move forward
turn south
and repeat from step 1 again

8) you do not have a red ball or a blue ball :(
move back till you are off dark
turn east
move forward
turn south
and repeat from step 1 again

9) If We have sweeped for long time , the start over by moving to end where we began

Feedback Algorithm used to make the robot move straight

1) Get the initial compass reading (absolute)
2) Get the relative reading from the compass by converting the next absolute reading into a relative reading corresponding to initial reading
3) Based on the relative reading adjust the motor powers appropriately
4) After a timeout , update the initial compass reading

Unstuck algorithm for turning

1) the max time required for a complete turn in the given terrain was initially experimentally calculated
2) If robot tries to turn left using the left wheel (moving backward) and time expired then use the right wheel (moving forward) to turn to desired direction
3) Step 2 applies to right wheel too...
4) Repeat step 2 and 3 till we reach the compass direction tolerance range

Default unstuck algorithm

1) If the robot is in a given state for a long time , it probably means it is stuck , if the robot is stuck then
2) check if the robot arm has a ball , if it has a ball , then deal with ball appropriately.
3) If there is no ball in the robot arm then move backward , then turn towards the direction facing the balls (notice that turn step has timeout and unstuck behaviour built into it) and rejuvenate itself :)

Problems and Solutions

This project had a few problems. The major problem was the compass sensor. The compass sensor has a very major flaw in that the slightest change in elevation would cause a drastic change in the reading and could change the direction that the robot was traveling in. To counteract the problem the team decided to minimize the use of the compass sensor to just telling the robot if it was moving, and if it was facing north, south, east or west. The other problem that we encountered was calibrate the robot to go straight. At first the team tried to make the robot go straight by using the compass sensor but because of the flaw in the sensor this was proven to be impossible, the solution to this was to just calibrate the motors to go straight as possible.

Another problem was getting the robot unstuck. Because of how the robot was built adding touch sensors and sonar sensors was not possible. So in order to fix this problem timeouts were used to determine if the robot was stuck. Timeouts were also used to see if the robot was in possession of a ball, how long to wait for a ball to be found before trying again in a different location and how long to move before turning.

Another issue that the robot had was when it got close to the edge of the table it had the chance of turning into the wall and becoming stuck. To combat this issue two solutions were put in place, one hardware and one software. The hardware solution was to but a circular lego piece on an arm in either side of the robot and when the robot got close to the wall the arm would keep the robot from turing into the wall. The software solution was take a compass reading to check if it was going the correct direction. If the compass reading showed it was going in the incorrect direction it would stop the robot and backup and turn until it faced the correct direction then resume its state.

Unsolved Problems

The only unsolved problem was an issue with the ball falling off of the ball scope before it reached the goal. The reason that the problem was not fixed because it was not encountered during testing. The solution to this problem is easy as all one needs to do is add a state that will check if a ball is still in the scoop.