Kurt Medley
Seminar: Robotics
June 5th, 2012

ROBOTIC ADAPTABILITY

ABSTRACT: The LeJOS NXT is a Java project that incorporates the functionality of the Java language within the NXT brick. The NXT brick controls the Lego Mindstorms robots and all their possible configurations. This is a continuation of the previous lab report. This lab report features the results and conclusion of the robotics lab. Details about the goals of the group, the implementation of programmatic features, and the conclusive results observed by the group.

INTRODUCTION: Previously, we defined the LeJOS NXT as a state machine, which is a machine partitioned by programmed behaviors. Behaviors like "MoveForward", "Stop", "BackUp", etc. can be categorized as specific states of the robot at any given time. The task of this conclusive lab was to conceptualize a way in which the robot could "learn" from its mistakes (i.e. running into a wall) by programming it to define its own states. We had success programming a fixed path for the robot to follow, given the proper sensors. The robot could run into a wall, backup, and continue on a path predefined for it. Using the LeJOS class libraries within the Java SE API, we were going devise a way in which to use sensors to store (or remember) previous decisions, thereby allowing the robot to avoid obstacles in the future.

BACKGROUND: The developers of the LeJOS NXJ have been able to hide the complex details associated with the sensors and actuators on the robot. This encapsulation frees the programmers creative potential as hardware addresses of components are not needed. The open source nature of the Mindstorms robot has been a significant advantage for the aspiring programmer. The community development of the Lego Mindstorms project has allowed programmers to explore robotics at an affordable price. LeJOS has made commands understandable using class libraries and programming practices reinforced by LeJOS NXJ's nearly parallel usage of said libraries.

METHODOLOGY: Having already become familiar with the NXT brick and how to program specifically with LeJOS NXT, our group produces excerpts of code with the intention of recording varying distances by which the robot could future-reference in its navigation out of the computer lab. Here is an excerpt:

```
public class DoorRunner {
    DifferentialPilot pilot;
    TouchSensor bump = new TouchSensor(SensorPort.S1);
    UltrasonicSensor sonic = new UltrasonicSensor(SensorPort.S2);
    private int[] distance = new int[25];
```

Here we've instantiated a DoorRunner class, creating instances of the sensors we need, TouchSensor, DifferentialPilot, and UltrasonicSensor. A pre-defined array of integers was created for the robot to back reference distances for future usage.

```
public void go() {
      pilot.forward();
      while (pilot.isMoving()) {
            if (bump.isPressed()) {
            pilot.stop(); } }
```

The method **go()** used the DifferentialPilot to move forward until the TouchSensor (bump) was pressed.

```
      if (!pilot.isMoving()) {
            pilot.travel(-30, true);
                  sonic.ping();
                  Sound.pause(30);
                  distance[0] = sonic.getDistance();
```

Here the code takes the bump into account. The robot is programmed to backup 30 centimeters, use the UltraSonic sensor to ping the distance and record it into the distance array. And it continues, without recording distances (time restraint), in this same fashion, mapping the route initially set for it.

```
                  pilot.rotate(90);
                  pilot.travel(30, true);
                  pilot.rotate(-90);
                  pilot.travel(30);
                  pilot.rotate(-90);
                  pilot.forward(); }
      while (pilot.isMoving())  {
            if (bump.isPressed()) { pilot.stop(); } }
      if (!pilot.isMoving()) {
            pilot.travel(-30, true);
            pilot.rotate(90);
            pilot.travel(40, true);
            }

      System.out.println(" " +
            pilot.getMovement().getDistanceTraveled());
      Button.waitForAnyPress(); } }
```

And the main() method acts as the driver, creating the TravelTest object named "robot" (clever), with its starting delimiters.

```
public static void main(String[] args) {
      TravelTest robot = new TravelTest();
      robot.pilot = new DifferentialPilot(2.25f, 5.5f,
      Motor.A, Motor.B);
      robot.go();
```

RESULTS: Again, the mechanical engineering was a limitation. Our robot did as instructed. Although it wasn't able to record all the points to follow a path through the lab doors, it did use the TouchSensor to backup, record the distance (30 centimeters) from the wall, and use this information to prevent a second collision into the wall.

CONCLUSION: This hands on lab was incredibly important for my understanding of robotics. The physical application allowed me to realize the challenges facing the robotic engineers and the software behind them. The trial and error given this experiment showed that the environmental and mechanical limitations are a significant factor in the progression of robotic science.

BIBLIOGRAPHY:

http://lejos.sourceforge.net/

The LeJOS homepage.

http://lejos.sourceforge.net/nxt/nxj/tutorial/index.htm

The tutorial used.

http://www.cs.cmu.edu/afs/cs/academic/class/15494-s12/Lectures.html

Turetzky's website on Cognitive Robotics

http://lejos.sourceforge.net/nxt/pc/api/

The LeJOS Class Libraries