Cody Constine, Christopher John, Dan Frost, Nolawee Mengist
October 20, 2016
Lab 3.3 write up

Lab 3.3

```
switch(code) {
    case 0: {
      inverseKin(pathOrder[p1]);
      code = 1;
    }
    break;
    case 1: {
      sparki.motorRotate(MOTOR_LEFT, DIR_CCW, percL);
      sparki.motorRotate(MOTOR_RIGHT, DIR_CW, percR);
      code = 2;
    }
    break;
    case 2: {
      int x, y = getCords(p1);
      double x2 = x/10;
      double y2 = y/10;
      if(abs(xi-x2)>.5 && abs(yi-y2)){
        p1++;
        code = 0;
      }
    }
    break;
    default: {
        sparki.moveStop();
    }
```

State 1: Calculate Inverse Kinematics of current target
State 2: Start the motors with the speed given above
State 3: Wait till within .5 of next point and change the current target to the next in the path, and start the machine over

Every loop: Calculate spaki's current positions

(1)

We used 5 cm as how close our sparki need to start the next point. This is close enough to the point that he won't start moving through a blocked point, but also it is not late enough that he won't pass over the current point.

(2)

If Sparki sees an obstacle, he will calculate the square that is in the grid, mark that square as an obstacle, and recalculate the Dijkstra's algorithm, and then start the state machine over. We have included our code snippet above of the state machine we