# Donna

Saturday, June 20, 2015

Wheel size: 2.595 \* PI = 8.152 inches (212.08 mm). One odo = 8.152 / 1200 = 0.0068 inches (0.173mm).

Line width is about 0.725 inches or 106 odos.

Code: fc81b4e..3277a91

Put car to right of line

Enter key to start

>> FORWARD

[Odo:0] +++ FOLLOW\_LINE

\*[Odo:0] Position: 3155 L 23 23 23 761 140 23 3 23 R, YAW: 261.94

... ... PID. Position: 3.16, a\_speed: 64, b\_speed: 64

>> FORWARD

\*[Odo:2] Position: 3155 L 23 23 23 761 140 23 3 23 R, YAW: 261.94

[Odo:2] +++ FOLLOW\_LINE

... ... PID. Position: 3.16, a\_speed: 64, b\_speed: 64

>> FORWARD

\*[Odo:10] Position: 3155 L 23 23 23 761 140 23 3 23 R, YAW: 261.94

[Odo:10] +++ FOLLOW\_LINE

... ... PID. Position: 3.16, a\_speed: 64, b\_speed: 64

>> FORWARD

\*[Odo:23] Position: 3155 L 23 23 23 761 140 23 3 23 R, YAW: 261.94

[Odo:24] +++ FOLLOW\_LINE

... ... PID. Position: 3.16, a\_speed: 64, b\_speed: 64

>> FORWARD

So loop is reading about every 12 odometries, or 0.08 inches (2 mm). Then:

\*[Odo:1888] Position: 2804 L 1000 861 658 436 300 258 350 658 R, YAW: 265.37

[Odo:1892] +++ FOLLOW\_LINE

... ... FOUND start of crossing, left: 1, right: 1

\*[Odo:1932] Position: 2804 L 1000 861 658 436 300 258 350 658 R, YAW: 265.37

[Odo:1935] +++ FOLLOW\_LINE\_END

\*[Odo:1963] Position: 2804 L 1000 861 658 436 300 258 350 658 R, YAW: 265.37

[Odo:1967] +++ FOLLOW\_LINE\_END

\*[Odo:1998] Position: 2804 L 1000 861 658 436 300 258 350 658 R, YAW: 265.37

[Odo:2002] +++ FOLLOW\_LINE\_END

\*[Odo:2031] Position: 2804 L 1000 861 658 436 300 258 350 658 R, YAW: 265.37

[Odo:2035] +++ FOLLOW\_LINE\_END

\*[Odo:2066] Position: 2804 L 1000 861 658 436 300 258 350 658 R, YAW: 265.37

[Odo:2072] +++ FOLLOW\_LINE\_END

\*[Odo:2098] Position: 2804 L 1000 861 658 436 300 258 350 658 R, YAW: 265.37

[Odo:2102] +++ FOLLOW\_LINE\_END

\*[Odo:2132] Position: 2804 L 1000 861 658 436 300 258 350 658 R, YAW: 265.37

[Odo:2136] +++ FOUND\_END

\*[Odo:2161] Position: 2804 L 1000 861 658 436 300 258 350 658 R, YAW: 265.37

[Odo:2165] +++ FOUND\_END

\*[Odo:2188] Position: 0 L 0 0 0 8 0 0 0 0 R, YAW: 265.87

[Odo:2189] +++ FOUND\_END

\*[Odo:2204] Position: 0 L 0 0 0 8 0 0 0 0 R, YAW: 265.87

[Odo:2205] +++ FOUND\_END

\*[Odo:2218] Position: 0 L 0 0 0 8 0 0 0 0 R, YAW: 265.87

[Odo:2220] +++ FOUND\_END

\*[Odo:2231] Position: 0 L 0 0 0 8 0 0 0 0 R, YAW: 265.87

[Odo:2232] +++ FOUND\_END

\*[Odo:2239] Position: 0 L 0 0 0 8 0 0 0 0 R, YAW: 265.87

[Odo:2239] +++ FOUND\_END

\*[Odo:2245] Position: 0 L 0 0 0 8 0 0 0 0 R, YAW: 265.87

[Odo:2245] +++ FOUND\_END

\*[Odo:2249] Position: 0 L 0 0 0 8 0 0 0 0 R, YAW: 265.87

[Odo:2250] +++ FOUND\_END

\*[Odo:2252] Position: 0 L 0 0 0 8 0 0 0 0 R, YAW: 265.87

[Odo:2252] +++ FOUND\_END

\*[Odo:2253] Position: 0 L 0 0 0 8 0 0 0 0 R, YAW: 265.87

[Odo:2253] +++ FOUND\_END

Car stopped approximately 0.72 inches beyond end-symbol end (18mm) [105 odos], is further to left. End-symbol is about 2.12 inches (54mm) [318 odos].

At 1892 odos, found line start. At 2136 odos, found line end. Sensor determined line width was 244 odos instead of 318.



6/20/2015 12:09 RUN 2 code: 3277a91..be21ba5

Start with motor to right of line over sensors 2 and 3 (1 is leftmost).

Car corrected so line is between 3 and 4

Enter key to start

>> FORWARD

\*[Odo:0] Position: 1744 L 25 253 738 48 3 25 3 25 R, A: 64, B: 64, YAW: 271.90

[Odo:0] +++ FOLLOW\_LINE

\*[Odo:1] Position: 1744 L 25 253 738 48 3 25 3 25 R, A: 64, B: 64, YAW: 271.90

[Odo:1] +++ FOLLOW\_LINE

----

\*[Odo:1279] Position: 1737 L 0 260 731 0 0 286 145 218 R, A: 64, B: 64, YAW: 273.07

[Odo:1287] +++ FOLLOW\_LINE

... ... FOUND start of crossing, left: 1, right: 1

\*[Odo:1320] Position: 2869 L 218 663 663 570 356 286 145 218 R, A: 64, B: 64, YAW: 273.07

[Odo:1327] +++ FOLLOW\_LINE\_END

\*[Odo:1357] Position: 2869 L 218 663 663 570 356 286 145 218 R, A: 64, B: 64, YAW: 273.07

[Odo:1364] +++ FOLLOW\_LINE\_END

\*[Odo:1393] Position: 2869 L 218 663 663 570 356 286 145 218 R, A: 64, B: 64, YAW: 273.07

[Odo:1399] +++ FOLLOW\_LINE\_END

\*[Odo:1428] Position: 2869 L 218 663 663 570 356 286 145 218 R, A: 64, B: 64, YAW: 273.07

[Odo:1435] +++ FOLLOW\_LINE\_END

\*[Odo:1460] Position: 2869 L 218 663 663 570 356 286 145 218 R, A: 64, B: 64, YAW: 273.07

[Odo:1468] +++ FOLLOW\_LINE\_END

\*[Odo:1496] Position: 2869 L 218 663 663 570 356 286 145 218 R, A: 64, B: 64, YAW: 273.07

[Odo:1504] +++ FOLLOW\_LINE\_END

\*[Odo:1531] Position: 2869 L 218 663 663 570 356 286 145 218 R, A: 0, B: 0, YAW: 273.07

[Odo:1536] +++ FOUND\_END

\*[Odo:1560] Position: 2869 L 218 663 663 570 356 286 145 218 R, A: 0, B: 0, YAW: 273.07

[Odo:1566] +++ FOUND\_END

\*[Odo:1586] Position: 2374 L 30 30 540 323 0 6 0 6 R, A: 0, B: 0, YAW: 272.21

[Odo:1588] +++ FOUND\_END

Starting sensor array: 25 253 738 48 3 25 3 25

Sensor array at line start: 0 260 731 0 0 286 145 218

THERE WAS NO PID CORRECTION. The next sensor reading is:

218 663 663 570 356 286 145 218

which is quite a bit of variation for the line. By near line end, the sensors are (identically)

218 663 663 570 356 286 145 218

After line end, with line continuation, sensors are:

30 30 540 323 0 6 0 6

The line is correctly between sensors 3 and 4.

Crossing line is 0.73in (109 odos). Found start at 1892, found end at 2136 or 244 odos (1.63in).

From line end at 2136 to stop at 2253, took 117 odos to stop (0.78in).

CONCLUSIONS:

* Need to stop faster.
* Sensors have widely different values for a black line. The arbitrary 218 value is probably 200 high. **CHANGING THRESHOLD TO 120**.



2015 06 20 12:41 RUN 3 Code: be21ba5..0fbb27e

Adding faster PID (+/- 20 instead of 10), fast stop mode.

Start between sensors 4 and 5, 20degree angle of backend to the left (yaw right).

Enter key to start

>> FORWARD

\*[Odo:0] Position: 3475 L 3 3 3 483 438 28 3 28 R, A: 64, B: 64, YAW: 269.84

[Odo:0] +++ FOLLOW\_LINE

----

\*[Odo:955] Position: 1102 L 51 965 173 0 6 6 6 6 R, A: 64, B: 64, YAW: 278.51

[Odo:960] +++ FOLLOW\_LINE

\*[Odo:992] Position: 90 L 583 58 0 0 0 0 0 0 R, A: 64, B: 64, YAW: 278.34

[Odo:995] +++ FOLLOW\_LINE

... ... FOUND start of crossing, left: 1, right: 0

\*[Odo:1033] Position: 90 L 583 58 0 0 0 0 0 0 R, A: 64, B: 64, YAW: 278.34

[Odo:1036] +++ FOLLOW\_LINE\_END

\*[Odo:1066] Position: 90 L 583 58 0 0 0 0 0 0 R, A: 64, B: 64, YAW: 278.34

[Odo:1070] +++ FOLLOW\_LINE\_END

\*[Odo:1096] Position: 90 L 583 58 0 0 0 0 0 0 R, A: 64, B: 64, YAW: 278.34

[Odo:1100] +++ FOLLOW\_LINE\_END

\*[Odo:1129] Position: 90 L 583 58 0 0 0 0 0 0 R, A: 64, B: 64, YAW: 278.34

[Odo:1133] +++ FOLLOW\_LINE\_END

\*[Odo:1165] Position: 90 L 583 58 0 0 0 0 0 0 R, A: 64, B: 64, YAW: 278.34

[Odo:1168] +++ FOLLOW\_LINE\_END

\*[Odo:1198] Position: 90 L 583 58 0 0 0 0 0 0 R, A: 64, B: 64, YAW: 278.34

[Odo:1202] +++ FOLLOW\_LINE\_END

\*[Odo:1232] Position: 90 L 583 58 0 0 0 0 0 0 R, A: 0, B: 0, YAW: 278.34

[Odo:1235] +++ FOUND\_END

\*[Odo:1260] Position: 90 L 583 58 0 0 0 0 0 0 R, A: 0, B: 0, YAW: 278.34

[Odo:1263] +++ FOUND\_END

\*[Odo:1283] Position: 5356 L 220 0 0 85 403 718 835 1000 R, A: 0, B: 0, YAW: 278.92

[Odo:1287] +++ FOUND\_END

\*[Odo:1304] Position: 5356 L 220 0 0 85 403 718 835 1000 R, A: 0, B: 0, YAW: 278.92

[Odo:1306] +++ FOUND\_END

\*[Odo:1321] Position: 5356 L 220 0 0 85 403 718 835 1000 R, A: 0, B: 0, YAW: 278.92

[Odo:1323] +++ FOUND\_END

\*[Odo:1333] Position: 5356 L 220 0 0 85 403 718 835 1000 R, A: 0, B: 0, YAW: 278.92

[Odo:1336] +++ FOUND\_END

\*[Odo:1343] Position: 5356 L 220 0 0 85 403 718 835 1000 R, A: 0, B: 0, YAW: 278.92

[Odo:1344] +++ FOUND\_END

\*[Odo:1349] Position: 5356 L 220 0 0 85 403 718 835 1000 R, A: 0, B: 0, YAW: 278.92

[Odo:1350] +++ FOUND\_END

\*[Odo:1353] Position: 5356 L 220 0 0 85 403 718 835 1000 R, A: 0, B: 0, YAW: 278.92

[Odo:1353] +++ FOUND\_END

Starting sensors: 3 3 3 483 438 28 3 28

Sensors at line start: 51 965 173 0 6 6 6 6

ROBOT IS PAST LEFT SENSOR, STILL AT 20 DEGREE ANGLE.

NOTE PID HAS NOT HAPPENED

Sensors at line end: 583 58 0 0 0 0 0 0

Stopped because odometry went more than 200.

From motor stop to full stop: 1353-1235 = 118 or 0.79 inches.



2015 06 20 13:23 RUN4 Code: 0fbb27e..747ba2a, 747ba2a..b97c859, b97c859..3225650

Adding call to motor.Forward at end of FOLLOW\_LINE strategy. Static variables. Stop loggin when at STOP.

Change FIND\_LINE\_END strategy to timeout after odo exceeds 700 instead of 200.

Motor between sensors 2 and 3, yaw right 20 degrees.

Car veered way left then right.



2015 06 20 13:53 RUN 4 Code: 3225650..261e8cd

return pid to +/- 10 variation. Fix bad position of call to Motor::Forward

veered way too left. Stopped after loosing line.

Odoend – start = 325 = 2.178 In

It can take about 0.1 sec to travers the .725 inch line (106 odos), which at 2500 usec max sample time for IR, gives max 41 samples per line traversal.



## Sunday, June 21, 2015, 21:15 Code: 99f0084..0b01750

This may be boring to you, but I finally got my first experiment to work, and I thought I’d share what I learned from the log data spewed out while the robot ran. This gives me a lot of data about the limits of my robot, velocities, common performance and things to look out for.

4961.75 :: --- FOLLOW\_LINE correcting with left turn [Odo:931] State: FOLLOW\_LINE, Position: 2.87   L 0 0 105 673 0 0 0 0 R  , YAW: 270.00, s: 0, e: 0, len: 0

4982.02 :: --- FOLLOW\_LINE on course [Odo:939] State: FOLLOW\_LINE, Position: 3.28   L 168 168 168 1000 394 168 89 168 R  , YAW: 272.24, s: 0, e: 0, len: 0

5008.57 :: --- FOLLOW\_LINE on course [Odo:951] State: FOLLOW\_LINE, Position: 3.24   L 178 178 178 1000 405 100 100 178 R  , YAW: 267.76, s: 0, e: 0, len: 0

5035.23 :: --- FOLLOW\_LINE on course [Odo:965] State: FOLLOW\_LINE, Position: 3.26   L 168 84 168 1000 263 84 84 168 R  , YAW: 269.55, s: 0, e: 0, len: 0

5061.39 :: --- FOLLOW\_LINE on course [Odo:980] State: FOLLOW\_LINE, Position: 3.09   L 168 168 310 1000 236 89 89 168 R  , YAW: 269.11, s: 0, e: 0, len: 0

5087.70 :: +++ FOLLOW\_LINE FOUND LINE START [Odo:1005] State: FIND\_LINE\_END, Position: 2.74  \*L 373 373 373 1000 152 152 152 152 R  , YAW: 267.32, s: 1001, e: 1001, len: 0

5239.12 :: +++ FIND\_LINE\_END FOUND LINE END BY IR [Odo:1135] State: FOUND\_END, Position: 2.50   L 168 0 168 1000 26 0 0 26 R  , YAW: 269.53, s: 1001, e: 1134, len: 133

5260.81 :: +++ FOUND\_END [Odo:1159] State: FOUND\_END, Position: 2.90   L 0 0 110 1000 42 0 0 0 R  , YAW: 268.67, s: 1001, e: 1134, len: 133

==== ==== STATS

loopCount: 80, avg loop duration: 17644.75

usec at start: 3869600, at end: 5281180, duration: 1411580

lineSensor sumLineSensorRead: 159756, avg read duration: 1996.95

sensorStick sumSensorStickRead: 132456,  avg read dration: 1655.70

5324.50 :: Final dump [Odo:1229] State: STOP, Position: 2.90   L 0 0 110 1000 42 0 0 0 R  , YAW: 268.67, s: 1001, e: 1134, len: 133

5370.69 :: +++ STOP [Odo:1259] State: STOP, Position: 2.85   L 0 31 173 1000 31 0 0 0 R  , YAW: 269.05, s: 1001, e: 1134, len: 133

5406.10 :: +++ STOP [Odo:1275] State: STOP, Position: 2.85   L 0 31 173 1000 31 0 0 0 R  , YAW: 269.33, s: 1001, e: 1134, len: 133

5441.49 :: +++ STOP [Odo:1288] State: STOP, Position: 2.70   L 0 100 184 1000 31 0 0 0 R  , YAW: 268.89, s: 1001, e: 1134, len: 133

5477.07 :: +++ STOP [Odo:1295] State: STOP, Position: 2.71   L 31 100 173 1000 31 0 0 0 R  , YAW: 269.55, s: 1001, e: 1134, len: 133

5512.70 :: +++ STOP [Odo:1297] State: STOP, Position: 2.67   L 31 100 242 1000 31 0 0 0 R  , YAW: 270.00, s: 1001, e: 1134, len: 133

This is near the end of my experiment, which was to follow a line until an intersection was found (start of line) and then continue on until the other side of the intersecting line was found and then stop..

The first column is my millisecond counter (i.e., the myMicros() / 1000 value, minus the base time when the program started). Followed by the old strategy state name, then the new strategy state, then the current line position (3.5 is centered on the line, < 3.5 is left of center — yaw left, > 3.5 is right of center — yaw right), then “L” followed by the 8 IR sensor values then “R”. If the lefmost or rightmost sensors detect a line, the “L” becomes “**\***L” or the “R” becomes “R**\***” (i.e. an asterisk is added). Followed by heading (yaw) in degrees, then the odometer reading when the near edge of a line is detected (s) and the odometer when far edge of a line is detected (e) and the length (width) in “odos” of the detected line.

So, near the end of my experiment you see:

At 4961.75 milliseconds, having traveled 931 odos, the robot is trying to follow the line but the line position has drifted down to 2.87, much less than 3.5 so the robot is trying to do a left turn to move it more over the center of the line. The current heading is 270 degrees and no line has been detected yet.

At 4982.02 milliseconds, having traveled 939 odos and having done a small left turn, we’re back within an acceptable line position (I don’t correct until the line position is less than 3 or greater than 4). Notice that only 19 milliseconds has elapsed between strategy decisions and the robot has travelled only 8 odos (about 0.054 inches).

Skipping until 5087.7 milliseconds, we see the robot has still been drifting left a bit (line position is now 2.74 and the yaw is down to 267.32 degrees) but it found the beginning edge of the line at 1005 odos. Normally, I would do a yaw correction but I don’t while I’m in the middle of detecting the end of a line. Given that I just manually placed the robot near a line on a piece of paper to start, this is close to the expected 1029 odos for the beginning of the line. The robot is tilted left a bit so you see the left set of sensor values are at 373 and the center line is still at 1000 while the right ones are still essentially white (152). Note the log says “**\***L” meaning it thinks a left turn has been detect while the right sensor says “R” without the asterisk as it hasn’t see the line yet. The robot strategy now changes to FIND\_LINE\_END.

At 5239.12 milliseconds, the robot has found the end of the line at odo 1135. You see at the end of the log line that the line start was at 1001 odos and the end is  1134 and the length is 133 odos. The expectation was 106 odos (0.725 inches), so it’s pretty close. The robot state changes to FOUND\_END which is where I would normally make a decision about what to do. For now, the decision is to just stop.

At 5260.81 milliseconds I try to stop to robot. I’m at 1159 odos. The outer loop will now keep pinging the sensors until the odometer quits changing so I can see how far the robot goes when told to stop.

The line beginning with “==== ==== STATS” is the final stats. We see that the strategy loop was executed 80 times, each loop taking about 17.6 milliseconds (the log is showing usecs instead of ms for the duration). Each strategy loop happens after the robot travels about 0.05 inches (from above). It took 1.41 seconds to travel the 7 inches and then to stop beyond the line. I can even compute the velocity of the robot. I went 1229 odos (8.3572 inches) in 1.411 sec or 5.9 inches per second. Divide by the wheel circumference of 8.152 inches we get 0.73 RPM.

Let’s compare that to the speed while in the middle of the loop where a yaw correction is taking place. At 4961 ms we are at odo 931. At 5239 we are at odo 1135. We’ve traveled 204 odos (1.387 in) in 278 ms or 5 inches/second or 0.61 RPM. So doing the yaw correction slowed things down, which is expected because the way I do yaw correction is to stop the robot, do very short right or left turns until the line is nearly centered again, and then go forward again.

Looking further at the stats, I see it took on average 1.996 ms to read the IR sensors, and 1.655 ms to read the heading sensor via I2C.

The dump continues until the odometer doesn’t change. We tried to stop at 1159 odos but didn’t come to a complete halt until 1297 odos, traveling 138 odos or 0.938 inches. Not exactly stopping on a dime. But that’s not a big bad thing. My sensors are out in front of the robot, and my intent is to do a right or left turn in place to find the line before continuing. It would work best if the line moved more towards the center of the robot when I execute my turn. I may even want to wait stopping until I’m more likely to be nearly centered on the line.

TODO

Even though I only found the left part of the crossing line to start, I need to keep track of if the right part is found while looking for the other side of the line.

I need to change my FOUND\_END strategy to then decide on the left-most possible turn (i.e. a left turn, continue or a right turn) and execute that turn — this will be a simple left-hand maze solver strategy.

I need to detect the end of the maze. Actually, that part is build in there as the end-of-line finder will stop when either it finds the other side of the line or it has traveled a known maximum distance.

I need to detect dead ends and execute a 180 degree turn.

Actually I plan on using my heading sensor to execute 90 and 180 degree turns — the plan is to turn maybe 80 or 170 degrees regardless of the line sensor and then use the line sensor to complete the turn.



## Monday, June 22, 2015 Code: 0b01750..1a6d2e1

Added code to sample the left/right endpoint sensors while crawling over the start of line/end of line range. At the end, if more than half the sensors have found a left/right line, we will assume the line goes left and/or right at the intersection.

Added code to sample the center sensor(s) while attempting to stop after having crossed the endpoint of a line. If half the samples show a center line, we will assume there is a continuation ahead after the intersecting line.

Log from sample that goes straight until it finds a right-turning line.

Start

Enter key to start

7093.96 :: --- FOLLOW\_LINE on course [Odo:0] State: FOLLOW\_LINE, Position: 3.40 L 100 100 100 1000 1000 100 26 100 R , YAW: 257.26, s: 0, e: 0, len: 0

7113.40 :: --- FOLLOW\_LINE on course [Odo:0] State: FOLLOW\_LINE, Position: 3.41 L 89 89 89 1000 1000 89 21 89 R , YAW: 256.97, s: 0, e: 0, len: 0

7138.91 :: --- FOLLOW\_LINE on course [Odo:1] State: FOLLOW\_LINE, Position: 3.41 L 89 89 89 1000 1000 89 21 89 R , YAW: 259.74, s: 0, e: 0, len: 0

7164.39 :: --- FOLLOW\_LINE on course [Odo:6] State: FOLLOW\_LINE, Position: 3.41 L 89 89 89 1000 1000 89 21 89 R , YAW: 256.90, s: 0, e: 0, len: 0

7189.90 :: --- FOLLOW\_LINE on course [Odo:12] State: FOLLOW\_LINE, Position: 3.41 L 89 89 89 1000 1000 89 21 89 R , YAW: 255.60, s: 0, e: 0, len: 0

7215.59 :: --- FOLLOW\_LINE on course [Odo:20] State: FOLLOW\_LINE, Position: 3.41 L 89 89 89 1000 1000 89 26 89 R , YAW: 256.55, s: 0, e: 0, len: 0

7241.20 :: --- FOLLOW\_LINE on course [Odo:31] State: FOLLOW\_LINE, Position: 3.41 L 89 89 89 1000 1000 89 21 89 R , YAW: 255.49, s: 0, e: 0, len: 0

7266.86 :: --- FOLLOW\_LINE on course [Odo:49] State: FOLLOW\_LINE, Position: 3.41 L 89 89 89 1000 1000 89 26 89 R , YAW: 256.75, s: 0, e: 0, len: 0

7292.54 :: --- FOLLOW\_LINE on course [Odo:70] State: FOLLOW\_LINE, Position: 3.41 L 89 89 89 1000 1000 89 21 89 R , YAW: 255.73, s: 0, e: 0, len: 0

7318.23 :: --- FOLLOW\_LINE on course [Odo:90] State: FOLLOW\_LINE, Position: 3.54 L 21 89 89 1000 1000 89 21 89 R , YAW: 255.70, s: 0, e: 0, len: 0

7343.87 :: --- FOLLOW\_LINE on course [Odo:112] State: FOLLOW\_LINE, Position: 3.73 L 0 26 26 1000 1000 100 26 100 R , YAW: 256.32, s: 0, e: 0, len: 0

7369.91 :: --- FOLLOW\_LINE on course [Odo:135] State: FOLLOW\_LINE, Position: 3.74 L 0 0 0 1000 1000 105 36 105 R , YAW: 255.78, s: 0, e: 0, len: 0

7395.57 :: --- FOLLOW\_LINE on course [Odo:158] State: FOLLOW\_LINE, Position: 3.73 L 0 0 0 1000 1000 100 26 100 R , YAW: 255.96, s: 0, e: 0, len: 0

7421.27 :: --- FOLLOW\_LINE on course [Odo:184] State: FOLLOW\_LINE, Position: 3.66 L 0 0 0 1000 952 31 31 100 R , YAW: 256.91, s: 0, e: 0, len: 0

7446.52 :: --- FOLLOW\_LINE on course [Odo:210] State: FOLLOW\_LINE, Position: 3.50 L 0 0 0 1000 1000 31 31 31 R , YAW: 256.28, s: 0, e: 0, len: 0

7471.88 :: --- FOLLOW\_LINE on course [Odo:236] State: FOLLOW\_LINE, Position: 3.50 L 21 21 21 1000 1000 21 0 21 R , YAW: 257.68, s: 0, e: 0, len: 0

7497.58 :: --- FOLLOW\_LINE on course [Odo:259] State: FOLLOW\_LINE, Position: 3.38 L 26 100 26 1000 1000 26 0 26 R , YAW: 256.72, s: 0, e: 0, len: 0

7523.40 :: --- FOLLOW\_LINE on course [Odo:281] State: FOLLOW\_LINE, Position: 3.34 L 47 126 47 1000 957 47 0 47 R , YAW: 255.79, s: 0, e: 0, len: 0

7549.11 :: --- FOLLOW\_LINE on course [Odo:305] State: FOLLOW\_LINE, Position: 3.14 L 100 100 100 1000 900 31 31 31 R , YAW: 256.51, s: 0, e: 0, len: 0

7575.27 :: --- FOLLOW\_LINE on course [Odo:335] State: FOLLOW\_LINE, Position: 3.10 L 110 110 110 1000 652 110 42 42 R , YAW: 256.86, s: 0, e: 0, len: 0

7701.74 :: --- FOLLOW\_LINE correcting with left turn [Odo:408] State: FOLLOW\_LINE, Position: 2.87 L 168 168 168 1000 557 89 21 21 R , YAW: 256.68, s: 0, e: 0, len: 0

7823.81 :: --- FOLLOW\_LINE correcting with left turn [Odo:415] State: FOLLOW\_LINE, Position: 2.88 L 242 242 173 1000 873 94 26 26 R , YAW: 256.75, s: 0, e: 0, len: 0

7845.78 :: --- FOLLOW\_LINE on course [Odo:415] State: FOLLOW\_LINE, Position: 3.11 L 168 242 89 1000 1000 21 21 89 R , YAW: 257.99, s: 0, e: 0, len: 0

7871.95 :: --- FOLLOW\_LINE on course [Odo:417] State: FOLLOW\_LINE, Position: 3.02 L 242 242 89 1000 1000 21 21 89 R , YAW: 253.42, s: 0, e: 0, len: 0

7898.14 :: --- FOLLOW\_LINE on course [Odo:422] State: FOLLOW\_LINE, Position: 3.10 L 168 247 89 1000 1000 21 21 89 R , YAW: 253.27, s: 0, e: 0, len: 0

7924.32 :: --- FOLLOW\_LINE on course [Odo:432] State: FOLLOW\_LINE, Position: 3.02 L 242 242 89 1000 1000 26 26 89 R , YAW: 253.35, s: 0, e: 0, len: 0

7950.51 :: --- FOLLOW\_LINE on course [Odo:450] State: FOLLOW\_LINE, Position: 3.07 L 178 252 178 1000 1000 31 31 100 R , YAW: 254.22, s: 0, e: 0, len: 0

8077.10 :: --- FOLLOW\_LINE correcting with left turn [Odo:480] State: FOLLOW\_LINE, Position: 2.98 L 247 247 178 1000 1000 21 21 89 R , YAW: 254.22, s: 0, e: 0, len: 0

8099.26 :: --- FOLLOW\_LINE on course [Odo:480] State: FOLLOW\_LINE, Position: 3.23 L 89 242 173 1000 1000 89 26 89 R , YAW: 254.09, s: 0, e: 0, len: 0

8125.46 :: --- FOLLOW\_LINE on course [Odo:486] State: FOLLOW\_LINE, Position: 3.23 L 89 247 168 1000 1000 89 21 89 R , YAW: 253.51, s: 0, e: 0, len: 0

8151.61 :: --- FOLLOW\_LINE on course [Odo:494] State: FOLLOW\_LINE, Position: 3.23 L 89 242 173 1000 1000 89 26 89 R , YAW: 255.23, s: 0, e: 0, len: 0

8177.83 :: --- FOLLOW\_LINE on course [Odo:503] State: FOLLOW\_LINE, Position: 3.23 L 94 247 178 1000 1000 94 21 94 R , YAW: 249.71, s: 0, e: 0, len: 0

8204.01 :: --- FOLLOW\_LINE on course [Odo:517] State: FOLLOW\_LINE, Position: 3.23 L 89 242 173 1000 1000 89 26 89 R , YAW: 251.23, s: 0, e: 0, len: 0

8230.13 :: --- FOLLOW\_LINE on course [Odo:534] State: FOLLOW\_LINE, Position: 3.29 L 89 173 173 1000 1000 89 26 89 R , YAW: 252.35, s: 0, e: 0, len: 0

8256.36 :: --- FOLLOW\_LINE on course [Odo:552] State: FOLLOW\_LINE, Position: 3.28 L 26 94 178 1000 1000 26 0 26 R , YAW: 252.73, s: 0, e: 0, len: 0

8282.19 :: --- FOLLOW\_LINE on course [Odo:570] State: FOLLOW\_LINE, Position: 3.35 L 110 110 184 1000 1000 110 26 110 R , YAW: 252.88, s: 0, e: 0, len: 0

8308.86 :: --- FOLLOW\_LINE on course [Odo:591] State: FOLLOW\_LINE, Position: 3.29 L 100 100 184 1000 1000 26 26 100 R , YAW: 251.24, s: 0, e: 0, len: 0

8335.41 :: --- FOLLOW\_LINE on course [Odo:610] State: FOLLOW\_LINE, Position: 3.54 L 26 94 94 1000 1000 94 26 94 R , YAW: 252.92, s: 0, e: 0, len: 0

8361.24 :: --- FOLLOW\_LINE on course [Odo:635] State: FOLLOW\_LINE, Position: 3.54 L 21 89 89 1000 1000 89 21 89 R , YAW: 252.27, s: 0, e: 0, len: 0

8387.08 :: --- FOLLOW\_LINE on course [Odo:659] State: FOLLOW\_LINE, Position: 3.41 L 89 89 89 1000 1000 89 21 89 R , YAW: 252.22, s: 0, e: 0, len: 0

8412.90 :: --- FOLLOW\_LINE on course [Odo:683] State: FOLLOW\_LINE, Position: 3.39 L 110 110 110 1000 1000 110 26 110 R , YAW: 251.57, s: 0, e: 0, len: 0

8439.63 :: --- FOLLOW\_LINE on course [Odo:707] State: FOLLOW\_LINE, Position: 3.41 L 89 89 89 1000 1000 89 21 89 R , YAW: 250.12, s: 0, e: 0, len: 0

8465.43 :: --- FOLLOW\_LINE on course [Odo:730] State: FOLLOW\_LINE, Position: 3.50 L 89 89 89 1000 1000 89 89 89 R , YAW: 252.19, s: 0, e: 0, len: 0

8491.28 :: --- FOLLOW\_LINE on course [Odo:755] State: FOLLOW\_LINE, Position: 3.55 L 89 89 89 1000 1000 173 89 89 R , YAW: 251.32, s: 0, e: 0, len: 0

8517.28 :: --- FOLLOW\_LINE on course [Odo:781] State: FOLLOW\_LINE, Position: 3.55 L 89 89 89 1000 1000 173 89 89 R , YAW: 252.73, s: 0, e: 0, len: 0

8543.30 :: --- FOLLOW\_LINE on course [Odo:808] State: FOLLOW\_LINE, Position: 3.66 L 89 89 89 1000 1000 178 89 178 R , YAW: 251.41, s: 0, e: 0, len: 0

8569.48 :: --- FOLLOW\_LINE on course [Odo:836] State: FOLLOW\_LINE, Position: 3.66 L 89 89 89 1000 1000 173 89 173 R , YAW: 251.23, s: 0, e: 0, len: 0

8595.65 :: --- FOLLOW\_LINE on course [Odo:866] State: FOLLOW\_LINE, Position: 3.66 L 89 89 21 1000 1000 89 89 168 R , YAW: 251.48, s: 0, e: 0, len: 0

8621.66 :: --- FOLLOW\_LINE on course [Odo:893] State: FOLLOW\_LINE, Position: 3.79 L 110 42 42 1000 1000 110 110 194 R , YAW: 252.20, s: 0, e: 0, len: 0

8648.17 :: --- FOLLOW\_LINE on course [Odo:918] State: FOLLOW\_LINE, Position: 3.79 L 89 89 21 1000 1000 168 89 242 R , YAW: 252.03, s: 0, e: 0, len: 0

8674.34 :: +++ FOLLOW\_LINE **FOUND LINE START** [Odo:948] State: FINE\_LINE\_END, Position: 5.00 L 26 26 0 1000 1000 1000 1000 1000 R\* , YAW: 252.19, s: 943, e: 943, len: 0

8753.44 :: +++ FIND\_LINE\_END **FOUND LINE END BY IR** [Odo:1033] State: FOUND\_END, Position: 4.32 L 0 0 0 689 326 326 184 184 R , YAW: 251.88, s: 943, e: 1030, len: **87**

8978.93 :: +++ FOUND\_END [Odo:1189] State: STOP, Position: 7.00 L 0 0 0 0 0 31 31 105 R , YAW: 251.33, s: 943, e: 1030, len: 87

Left/right segment samples: 16, center segment samples: 58, L: 0, C: 0, **R\*: 14**

==== ==== STATS

loopCount: 124, avg loop duration: 15448.97

usec at start: 7088584, at end: 9004256, duration: 1915672

lineSensor sumLineSensorRead: 244828, avg read duration: 1974.42

sensorStick sumSensorStickRead: 202084, avg read dration: 1629.71

9047.81 :: Final dump [Odo:1195] State: STOP, Position: 7.00 L 0 0 0 0 0 31 31 105 R , YAW: 251.33, s: 943, e: 1030, len: 87

In case any of this gives you useful ideas.

Previously, my robot:

\* Started on a line and looked at the Pololu IR sensors to get the “line position value”. A value of 3500 (in my log, I divide by 1000 so the value shown would be 3.5) is considered to be dead center on the line.

\* I arbitrarily decided that if the line position value became less than 3000, then the robot is veering to the right (the left sensors are lighting up) so I stop the robot, make a small left turn correction and check to see the new line position value. I keep making corrections until the line position value is greater than 3000 again. Similarly, if the value is more than 4000, I stop and make right turn corrections.

\* When the leftmost or rightmost of the 8 sensors detected a line, the robot entered into a mode where it assumed it had found the leading edge of the line and now it wanted to find the trailing edge. So it marked the leading edge position (using the quadrature wheel position sensor) and continued forward (with no yaw corrections) until both the left and right sensors read zero again (or until a maximum distance was travelled so the robot didn’t just go on forever). It was searching to find the trailing edge of the line.

\* When the trailing edge of the line was found, the robot stopped.

Tonight, so far, I’ve enhanced that code.

\* When the trailing edge of a line is found, in addition to the above I begin separately sampling the leftmost and rightmost sensors. I count how many black readings there are for those two sensors, and the total number of samples that were made. I keep this up, as above, until both the left and right sensors see white again (i.e., I’ve found the trailing edge of the line). If more than half the samples showed a black line, I assume there is a left or right intersection. For example, if, starting at the leading edge, I make 16 samples until I see the trailing edge, and I find that 14 of the samples found a line on the left and 5 found a line on the right, I assume the right samples are noise but there is a line to the left.

\* Once the trailing edge is found, I begin sampling the middle sensor(s) (If there are an odd number of them, I just use the center one, if there are an even number, I use the center two — there are eight sensor always but my code is a bit more general). I tell the robot to stop starting at the trailing edge and keep sampling the center sensor(s) until I find the odometer no longer changes (i.e., the robot has finally stopped). It takes the robot currently over an inch of travel before it finally stops. Like above, I keep track of how many center samples were made and how many of those samples showed a black line. If at least half of them found a black line, I assume that the original line continues beyond the crossing line.

That’s how I detect a left, right or continuation, or any combination of them, when a crossing line is found.

Upcoming:

\* My line sensor is a bit unreliable. For some reasons, it sometimes reports it found a black spot when there clearly isn’t one. I will probably change my code that reads the IR sensors so that it always keeps the last two readings and reports the average of those two readings. That might help filter some of the sensor error problems.

\* I’m about to add a Map class. When ever I find an intersecting line, it will report how far it traveled from the start or last intersection, in what heading, and if there are left, center and/or right options at the intersection. I’ll build a map using a tree for that that class. For example, if the maze looks like:

     +===

     |

=====+=====+

     |     |

     |     |

     |

     +==

Where the maze starts on the left, the map would end up being:

length: 5

heading: E

left: —————>  length: 1

              heading: N

              left: x

              center: x

              right: ——> length: 3 (since l/c/r are ‘x’, this is a dead end)

                         heading: E

                         left: x

                         center: x

                         right: x

center: ——> length: 5

            heading: E

            left: x

            center: x

            right: ——> length: 2

                       heading: S

                       left: x

                       center: x

                       right:

right: -> length: 3

          heading: S

          left: ——> length: 2

                    heading: E

                    left: x

                    center: x

                    right:

          center: x

          right:



## Tuesday, June 13, 2015 13:10 Code: cc738ac..804beb4

Added code to detect left/center/right options and execute the leftmost option. Log:

Start

Enter key to start

1923.58 :: --- FOLLOW\_LINE correcting with right turn [Odo:0] State: FOLLOW\_LINE, Position: 4.61 L 80 80 8 42 234 1000 157 80 R , YAW: 267.46, s: 0, e: 0, len: 0

1945.66 :: --- FOLLOW\_LINE correcting with right turn [Odo:0] State: FOLLOW\_LINE, Position: 4.56 L 88 88 8 45 211 1000 128 88 R , YAW: 267.75, s: 0, e: 0, len: 0

1973.82 :: --- FOLLOW\_LINE correcting with right turn [Odo:1] State: FOLLOW\_LINE, Position: 4.59 L 82 82 11 45 217 1000 134 82 R , YAW: 266.99, s: 0, e: 0, len: 0

2002.22 :: --- FOLLOW\_LINE correcting with right turn [Odo:6] State: FOLLOW\_LINE, Position: 4.59 L 82 82 11 45 205 1000 128 82 R , YAW: 267.78, s: 0, e: 0, len: 0

2030.68 :: --- FOLLOW\_LINE correcting with right turn [Odo:13] State: FOLLOW\_LINE, Position: 4.60 L 88 88 8 45 211 1000 165 88 R , YAW: 267.55, s: 0, e: 0, len: 0

2059.01 :: --- FOLLOW\_LINE correcting with right turn [Odo:24] State: FOLLOW\_LINE, Position: 4.78 L 62 62 25 25 182 1000 182 105 R , YAW: 267.52, s: 0, e: 0, len: 0

2087.76 :: --- FOLLOW\_LINE correcting with right turn [Odo:37] State: FOLLOW\_LINE, Position: 4.60 L 88 88 14 51 171 1000 208 88 R , YAW: 267.55, s: 0, e: 0, len: 0

2116.31 :: --- FOLLOW\_LINE correcting with right turn [Odo:51] State: FOLLOW\_LINE, Position: 4.81 L 62 62 22 22 145 1000 185 108 R , YAW: 267.53, s: 0, e: 0, len: 0

2145.03 :: --- FOLLOW\_LINE correcting with right turn [Odo:68] State: FOLLOW\_LINE, Position: 4.65 L 88 88 14 51 131 1000 254 88 R , YAW: 267.80, s: 0, e: 0, len: 0

2173.57 :: --- FOLLOW\_LINE correcting with right turn [Odo:91] State: FOLLOW\_LINE, Position: 5.26 L 48 48 11 48 91 1000 297 91 R , YAW: 267.81, s: 0, e: 0, len: 0

2201.99 :: --- FOLLOW\_LINE correcting with right turn [Odo:113] State: FOLLOW\_LINE, Position: 4.93 L 57 57 20 20 57 1000 351 57 R , YAW: 267.25, s: 0, e: 0, len: 0

2230.53 :: --- FOLLOW\_LINE correcting with right turn [Odo:139] State: FOLLOW\_LINE, Position: 4.96 L 57 57 20 20 57 1000 391 57 R , YAW: 266.96, s: 0, e: 0, len: 0

2259.16 :: --- FOLLOW\_LINE correcting with right turn [Odo:165] State: FOLLOW\_LINE, Position: 5.32 L 45 45 8 8 91 1000 417 91 R , YAW: 268.08, s: 0, e: 0, len: 0

2287.33 :: --- FOLLOW\_LINE correcting with right turn [Odo:187] State: FOLLOW\_LINE, Position: 5.00 L 51 51 8 51 94 1000 474 94 R , YAW: 266.97, s: 0, e: 0, len: 0

2315.73 :: --- FOLLOW\_LINE correcting with right turn [Odo:212] State: FOLLOW\_LINE, Position: 5.37 L 45 45 8 45 85 1000 551 85 R , YAW: 268.36, s: 0, e: 0, len: 0

2344.19 :: --- FOLLOW\_LINE correcting with right turn [Odo:240] State: FOLLOW\_LINE, Position: 5.44 L 45 45 8 45 88 1000 660 128 R , YAW: 267.26, s: 0, e: 0, len: 0

2372.65 :: --- FOLLOW\_LINE correcting with right turn [Odo:269] State: FOLLOW\_LINE, Position: 5.45 L 45 45 11 45 91 1000 794 91 R , YAW: 265.35, s: 0, e: 0, len: 0

2401.22 :: --- FOLLOW\_LINE correcting with right turn [Odo:296] State: FOLLOW\_LINE, Position: 5.30 L 45 91 8 8 45 1000 757 91 R , YAW: 267.82, s: 0, e: 0, len: 0

2429.49 :: --- FOLLOW\_LINE correcting with right turn [Odo:324] State: FOLLOW\_LINE, Position: 5.27 L 60 60 22 22 60 922 980 102 R , YAW: 268.90, s: 0, e: 0, len: 0

2457.99 :: --- FOLLOW\_LINE correcting with right turn [Odo:354] State: FOLLOW\_LINE, Position: 5.60 L 45 45 8 8 45 971 1000 128 R , YAW: 266.80, s: 0, e: 0, len: 0

2486.38 :: --- FOLLOW\_LINE correcting with right turn [Odo:385] State: FOLLOW\_LINE, Position: 5.63 L 45 45 8 8 45 877 1000 128 R , YAW: 267.95, s: 0, e: 0, len: 0

2514.85 :: --- FOLLOW\_LINE correcting with right turn [Odo:417] State: FOLLOW\_LINE, Position: 5.32 L 51 51 14 51 51 734 1000 134 R , YAW: 267.32, s: 0, e: 0, len: 0

2543.54 :: --- FOLLOW\_LINE correcting with right turn [Odo:450] State: FOLLOW\_LINE, Position: 5.36 L 62 62 25 25 62 694 1000 182 R , YAW: 267.94, s: 0, e: 0, len: 0

2572.30 :: --- FOLLOW\_LINE correcting with right turn [Odo:485] State: FOLLOW\_LINE, Position: 5.49 L 91 45 8 45 8 545 1000 165 R , YAW: 267.64, s: 0, e: 0, len: 0

2600.69 :: --- FOLLOW\_LINE correcting with right turn [Odo:516] State: FOLLOW\_LINE, Position: 5.39 L 62 62 20 62 20 440 1000 194 R , YAW: 267.18, s: 0, e: 0, len: 0

2629.39 :: --- FOLLOW\_LINE correcting with right turn [Odo:550] State: FOLLOW\_LINE, Position: 5.37 L 97 51 14 51 14 388 1000 228 R , YAW: 267.06, s: 0, e: 0, len: 0

2658.09 :: --- FOLLOW\_LINE correcting with right turn [Odo:583] State: FOLLOW\_LINE, Position: 5.54 L 62 62 25 25 25 305 1000 228 R , YAW: 266.53, s: 0, e: 0, len: 0

2686.83 :: --- FOLLOW\_LINE correcting with right turn [Odo:617] State: FOLLOW\_LINE, Position: 5.66 L 51 51 14 14 14 265 1000 265 R , YAW: 265.82, s: 0, e: 0, len: 0

2715.59 :: --- FOLLOW\_LINE correcting with right turn [Odo:652] State: FOLLOW\_LINE, Position: 5.73 L 51 51 14 14 14 171 1000 305 R , YAW: 265.03, s: 0, e: 0, len: 0

2744.29 :: +++ FOLLOW\_LINE FOUND LINE START [Odo:687] State: FIND\_LINE\_END, Position: 6.18 L 48 48 11 48 11 142 1000 425 R\* , YAW: 266.47, s: 683, e: 683, len: 0

3009.92 :: +++ FIND\_LINE\_END FOUND LINE END BY IR [Odo:1009] State: FOUND\_END, Position: 6.54 L 40 40 2 2 2 40 208 248 R , YAW: 267.03, s: 683, e: 1008, len: 325

3036.29 :: +++ PRE\_ROLL [Odo:1042] State: PRE\_ROLL, Position: 7.00 L 54 54 8 8 8 54 8 54 R , YAW: 267.30, s: 683, e: 1008, len: 325

goal odo: 1302

3057.73 :: +++ PRE\_ROLL [Odo:1068] State: PRE\_ROLL, Position: 7.00 L 68 68 17 17 17 17 17 68 R , YAW: 267.04, s: 683, e: 1008, len: 325

goal odo: 1302

3084.24 :: +++ PRE\_ROLL [Odo:1099] State: PRE\_ROLL, Position: 7.00 L 65 65 28 28 28 65 28 65 R , YAW: 267.73, s: 683, e: 1008, len: 325

goal odo: 1302

3110.80 :: +++ PRE\_ROLL [Odo:1130] State: PRE\_ROLL, Position: 7.00 L 60 60 22 22 22 60 22 100 R , YAW: 266.34, s: 683, e: 1008, len: 325

goal odo: 1302

3137.46 :: +++ PRE\_ROLL [Odo:1166] State: PRE\_ROLL, Position: 7.00 L 54 54 14 14 14 54 14 94 R , YAW: 266.11, s: 683, e: 1008, len: 325

goal odo: 1302

3164.01 :: +++ PRE\_ROLL [Odo:1198] State: PRE\_ROLL, Position: 7.00 L 60 60 20 20 20 60 20 100 R , YAW: 266.74, s: 683, e: 1008, len: 325

goal odo: 1302

3190.65 :: +++ PRE\_ROLL [Odo:1233] State: PRE\_ROLL, Position: 7.00 L 60 60 20 20 20 20 20 60 R , YAW: 267.57, s: 683, e: 1008, len: 325

goal odo: 1302

3217.19 :: +++ PRE\_ROLL [Odo:1267] State: PRE\_ROLL, Position: 7.00 L 20 60 20 20 20 20 20 60 R , YAW: 267.60, s: 683, e: 1008, len: 325

goal odo: 1302

3243.78 :: +++ PRE\_ROLL [Odo:1300] State: PRE\_ROLL, Position: 7.00 L 60 60 20 20 20 20 20 60 R , YAW: 267.32, s: 683, e: 1008, len: 325

goal odo: 1302

3270.26 :: +++ STOPPING WAITING FOR FULL STOP [Odo:1339] State: STOPPING, Position: 7.00 L 65 65 11 11 11 11 11 65 R , YAW: 268.66, s: 683, e: 1008, len: 325

3297.77 :: +++ STOPPING WAITING FOR FULL STOP [Odo:1372] State: STOPPING, Position: 7.00 L 57 57 17 17 17 17 17 57 R , YAW: 267.87, s: 683, e: 1008, len: 325

3325.28 :: +++ STOPPING WAITING FOR FULL STOP [Odo:1401] State: STOPPING, Position: 7.00 L 68 68 17 17 17 17 17 68 R , YAW: 266.81, s: 683, e: 1008, len: 325

3352.68 :: +++ STOPPING WAITING FOR FULL STOP [Odo:1424] State: STOPPING, Position: 7.00 L 14 54 14 14 14 14 14 54 R , YAW: 266.81, s: 683, e: 1008, len: 325

3380.41 :: +++ STOPPING WAITING FOR FULL STOP [Odo:1443] State: STOPPING, Position: 7.00 L 8 54 8 8 8 54 8 54 R , YAW: 267.34, s: 683, e: 1008, len: 325

3407.10 :: +++ STOPPING WAITING FOR FULL STOP [Odo:1458] State: STOPPING, Position: 7.00 L 11 11 11 11 11 57 11 57 R , YAW: 267.32, s: 683, e: 1008, len: 325

3434.65 :: +++ STOPPING WAITING FOR FULL STOP [Odo:1471] State: STOPPING, Position: 7.00 L 17 17 17 17 17 57 17 57 R , YAW: 266.81, s: 683, e: 1008, len: 325

3462.16 :: +++ STOPPING WAITING FOR FULL STOP [Odo:1480] State: STOPPING, Position: 7.00 L 11 11 11 11 11 57 11 57 R , YAW: 267.35, s: 683, e: 1008, len: 325

3489.70 :: +++ STOPPING WAITING FOR FULL STOP [Odo:1486] State: STOPPING, Position: 7.00 L 14 14 14 14 14 54 14 54 R , YAW: 266.27, s: 683, e: 1008, len: 325

3517.23 :: +++ STOPPING FOUND\_END [Odo:1489] State: STOPPING, Position: 7.00 L 20 20 20 20 20 20 20 62 R , YAW: 266.01, s: 683, e: 1008, len: 325

Left/right segment samples: 55, center segment samples: 1, L: 0, C\*: 0, R\*: 54

3556.33 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1490] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 17 17 17 17 17 57 17 57 R , YAW: 267.07, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3592.24 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1490] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 11 11 0 11 11 51 11 51 R , YAW: 267.60, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3627.94 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1496] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 11 11 0 11 11 51 11 51 R , YAW: 261.68, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3663.64 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1511] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 14 14 14 14 14 14 14 57 R , YAW: 268.13, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3699.53 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1526] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 14 54 14 14 14 54 14 54 R , YAW: 266.08, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3735.41 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1545] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 17 17 0 17 17 17 17 57 R , YAW: 270.53, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3771.11 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1563] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 14 54 14 14 14 54 14 54 R , YAW: 263.98, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3807.07 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1585] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 54 54 14 14 14 54 14 100 R , YAW: 274.16, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3843.03 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1613] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 51 51 11 11 11 51 11 51 R , YAW: 269.23, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3878.88 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1644] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 48 48 11 11 11 48 11 94 R , YAW: 274.76, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3914.73 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1674] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 62 62 11 11 11 11 11 62 R , YAW: 279.67, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3950.62 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1710] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 48 48 11 11 11 48 11 97 R , YAW: 273.98, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

3986.49 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1746] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 48 48 8 8 8 48 8 48 R , YAW: 284.22, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4021.71 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1783] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 51 51 11 11 11 11 11 51 R , YAW: 280.08, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4057.53 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1820] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 85 45 8 45 8 45 8 85 R , YAW: 289.57, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4092.92 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1855] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 91 45 8 45 45 45 8 91 R , YAW: 283.86, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4128.42 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1892] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 111 65 25 25 25 25 25 65 R , YAW: 294.17, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4164.46 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1935] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 65 65 25 25 25 25 25 65 R , YAW: 298.10, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4200.37 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:1974] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 60 60 8 8 8 8 8 60 R , YAW: 299.58, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4235.40 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:2014] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 48 48 8 8 8 8 8 48 R , YAW: 293.26, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4270.21 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:2054] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 48 48 8 48 8 8 8 88 R , YAW: 299.72, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4305.56 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:2096] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 54 54 11 11 11 11 11 54 R , YAW: 306.39, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4341.44 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:2137] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 65 65 25 25 25 25 0 65 R , YAW: 309.81, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4377.12 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:2177] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 60 60 20 20 20 20 20 60 R , YAW: 306.83, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4412.98 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:2220] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 48 48 11 11 11 48 11 100 R , YAW: 315.29, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4449.08 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:2265] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 54 54 17 17 17 54 17 54 R , YAW: 315.44, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4484.92 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:2310] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 65 65 0 25 25 25 25 65 R , YAW: 316.48, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4520.63 :: +++ RIGHT\_TURN\_MIN\_YAW [Odo:2353] State: RIGHT\_TURN\_MIN\_YAW, Position: 7.00 L 40 40 0 40 0 40 0 80 R , YAW: 322.62, s: 683, e: 1008, len: 325

turn\_start\_heading\_: 266.01, goal heading: 321.00

4555.94 :: +++ RIGHT\_TURN [Odo:2393] State: RIGHT\_TURN, Position: 7.00 L 68 68 28 28 28 28 28 68 R , YAW: 320.62, s: 683, e: 1008, len: 325

4580.48 :: +++ RIGHT\_TURN [Odo:2423] State: RIGHT\_TURN, Position: 7.00 L 48 48 8 8 8 48 8 48 R , YAW: 327.48, s: 683, e: 1008, len: 325

4604.27 :: +++ RIGHT\_TURN [Odo:2454] State: RIGHT\_TURN, Position: 7.00 L 8 8 0 8 8 8 8 57 R , YAW: 329.81, s: 683, e: 1008, len: 325

4627.57 :: +++ RIGHT\_TURN [Odo:2483] State: RIGHT\_TURN, Position: 7.00 L 62 22 22 22 22 22 22 62 R , YAW: 327.44, s: 683, e: 1008, len: 325

4652.02 :: +++ RIGHT\_TURN [Odo:2512] State: RIGHT\_TURN, Position: 7.00 L 51 51 11 11 11 11 11 51 R , YAW: 332.60, s: 683, e: 1008, len: 325

4676.49 :: +++ RIGHT\_TURN [Odo:2543] State: RIGHT\_TURN, Position: 7.00 L 11 11 0 11 11 11 11 62 R , YAW: 334.66, s: 683, e: 1008, len: 325

4700.82 :: +++ RIGHT\_TURN [Odo:2571] State: RIGHT\_TURN, Position: 7.00 L 11 54 11 11 11 11 11 54 R , YAW: 335.19, s: 683, e: 1008, len: 325

4725.34 :: +++ RIGHT\_TURN [Odo:2595] State: RIGHT\_TURN, Position: 7.00 L 11 11 11 11 11 11 11 62 R , YAW: 336.75, s: 683, e: 1008, len: 325

4749.77 :: +++ RIGHT\_TURN [Odo:2624] State: RIGHT\_TURN, Position: 7.00 L 51 51 11 11 11 11 11 51 R , YAW: 338.44, s: 683, e: 1008, len: 325

4774.35 :: +++ RIGHT\_TURN [Odo:2652] State: RIGHT\_TURN, Position: 7.00 L 48 48 8 8 8 48 8 48 R , YAW: 339.68, s: 683, e: 1008, len: 325

4798.04 :: +++ RIGHT\_TURN [Odo:2679] State: RIGHT\_TURN, Position: 7.00 L 65 65 8 8 8 8 8 65 R , YAW: 341.45, s: 683, e: 1008, len: 325

4821.68 :: +++ RIGHT\_TURN [Odo:2703] State: RIGHT\_TURN, Position: 7.00 L 54 54 14 14 14 14 14 54 R , YAW: 341.20, s: 683, e: 1008, len: 325

4846.20 :: +++ RIGHT\_TURN [Odo:2732] State: RIGHT\_TURN, Position: 7.00 L 8 8 8 8 8 8 8 51 R , YAW: 344.47, s: 683, e: 1008, len: 325

4869.54 :: +++ RIGHT\_TURN [Odo:2759] State: RIGHT\_TURN, Position: 7.00 L 8 48 8 8 8 8 8 48 R , YAW: 345.79, s: 683, e: 1008, len: 325

4893.01 :: +++ RIGHT\_TURN [Odo:2785] State: RIGHT\_TURN, Position: 7.00 L 48 48 8 8 8 48 8 48 R , YAW: 344.11, s: 683, e: 1008, len: 325

4916.71 :: +++ RIGHT\_TURN [Odo:2811] State: RIGHT\_TURN, Position: 7.00 L 62 62 0 22 22 22 0 62 R , YAW: 348.38, s: 683, e: 1008, len: 325

4940.97 :: +++ RIGHT\_TURN [Odo:2836] State: RIGHT\_TURN, Position: 7.00 L 65 65 8 8 8 8 8 65 R , YAW: 348.83, s: 683, e: 1008, len: 325

4964.46 :: +++ RIGHT\_TURN [Odo:2863] State: RIGHT\_TURN, Position: 7.00 L 62 62 22 22 22 22 22 108 R , YAW: 348.61, s: 683, e: 1008, len: 325

4989.08 :: +++ RIGHT\_TURN [Odo:2892] State: RIGHT\_TURN, Position: 5.15 L 62 62 22 22 22 22 22 311 R , YAW: 351.92, s: 683, e: 1008, len: 325

==== ==== STATS

loopCount: 151, avg loop duration: 20476.29

usec at start: 1917392, at end: 5009312, duration: 3091920

lineSensor sumLineSensorRead: 417168, avg read duration: 2762.70

sensorStick sumSensorStickRead: 246388, avg read dration: 1631.71

5052.85 :: Final dump [Odo:2974] State: STOP, Position: 5.15 L 62 62 22 22 22 22 22 311 R , YAW: 351.92, s: 683, e: 1008, len: 325

== == MAP @2382

x: 0, y: 0, turn: C, heading: 0.00, logical heading: E

length: 0, odometer: 0, loop detected: f, has\_left\_turn: f, has\_right\_turn: f, has\_continuation: T

left: 0, center: 0, right: 2456

== == MAP @2456

x: 0, y: -1490, turn: R, heading: 266.01, logical heading: S

length: 1490, odometer: 1490, loop detected: f, has\_left\_turn: f, has\_right\_turn: T, has\_continuation: T

left: 0, center: 0, right: 0

5173.15 :: +++ STOP [Odo:3111] State: STOP, Position: 5.04 L 57 57 0 0 20 1000 391 97 R , YAW: 353.21, s: 683, e: 1008, len: 325

5209.52 :: +++ STOP [Odo:3153] State: STOP, Position: 4.62 L 51 51 14 14 405 1000 51 91 R , YAW: 2.05, s: 683, e: 1008, len: 325

5245.92 :: +++ STOP [Odo:3182] State: STOP, Position: 4.50 L 40 40 2 40 902 505 2 80 R , YAW: 5.13, s: 683, e: 1008, len: 325

5281.78 :: +++ STOP [Odo:3201] State: STOP, Position: 4.28 L 40 40 2 80 1000 234 2 80 R , YAW: 6.58, s: 683, e: 1008, len: 325

5317.70 :: +++ STOP [Odo:3211] State: STOP, Position: 4.17 L 40 40 2 157 1000 157 2 80 R , YAW: 7.33, s: 683, e: 1008, len: 325

5353.89 :: +++ STOP [Odo:3215] State: STOP, Position: 4.11 L 42 42 5 200 1000 117 5 80 R , YAW: 7.78, s: 683, e: 1008, len: 325



18:14 Status. Code: ed415a2..bf4b119

The robot first solved a primitive maze. Here is the e-mail I send to Ralf and Grey about the Strategy code.

For you edification, here’s a description of my Strategy state machine. It starts out in the FOLLOW\_LINE state, requires that the robot IR sensors be on the starting line and the robot be more or less be pointed in the starting direction. There are a lot of states in my Strategy state machine as I try to overcome the impreciseness of everything.

The state machine is executed as part of the outer loop and just executes as fast as possible with no delays, though one or two of the actions I take in turning add a bit of delay to make the turning work better. At the beginning of the Strtegy I always perform a couple of calculations:

* Set leftTurnFound if the leftmost IR Sensor (hereafter called just IR) has a value that exceeds kLINE\_DETECTED\_THRESHOLD which is 400.
* Set rightTurnFound if the rightmost IR has a value that exceeds kLINE\_DETECTED\_THRESHOLD.
* Set deadEndFound to (this formula says that I must travel an inch without the middle IR sensors seeing the line):
  + If the middle IR(s) have a value that exceeds kLINE\_DETECTED\_THRESHOLD, deadEndFound is false.
  + Otherwise, if the robot has traveled at least kMIN\_NO\_CENTER\_IS\_DEAD\_END odos (currently the number of odos in an inch) in a row without seeing a line under the middle IRs, deadEndFound is true. The IR sensors have enough noise and glitchiness that you don’t want to trust a single reading.
* Set position to the IR “position value", which is 0 to 7000, divided by 1000. I just like dealing with the floating point version.

So, here’s what each state does. At the end of the state (indicated by ‘break', the loop starts again and jumps into the code for the appropriate (sometimes new) state.

FOLLOW\_LINE

* If deadEndFound
  + The robot needs to turn around. Turning is very imprecise, as is all movement, so I do it in two parts. I use the magnetic compass to capture the current heading and begin a left turn of 145 degrees, instead of 180 all at once. When the compass heading says I’ve done that much of the turn, handled by the upcoming DEAD\_END\_MIN\_YAW state, I then revert to the mode where I begin looking for the line and try to stop turning when the line is somewhat centered. The reason for the 145 degrees is that by experiment, I found that when I turn off the motors, they don’t stop instantly and that gives me a reasonable first try that still leaves me room to turn some more to find the line. So, just set state to DEAD\_END\_MIN\_YAW and break;
* Switching to personal mode here, using “I” instead of “the robot”. If I see the line over the center IR \_AND\_ either the leftmost or rightmost sensor detects a line, then I believe I have found the beginning of an intersecting line. I don’t want to trust that as the sole indicator that I’ve found a line—I want to keep moving until I find the other side of the intersecting line, and then see if I consistently found the left and/or right intersecting line. [Note here: the IR sensor itself is subject to wonky behavior at times, so I should really take the average of 2 or more samples, but each sample adds about 2ms of time when using the digital IR sensors, and with a moving robot, it quickly becomes possible to totally miss something if you are doing something else when you should have been sensing. I have a strategy to fix this, but I’ll do it later]. So, if I found a possible intersecting, I want to go into the state where I crawl until I find the other side of the line. Set state to FIND\_LINE\_END and break.
* I’m not at a dead end, I’m not over a line, so I want to just continue following the line. Here I use the “position” value of the IR which will give a value of 3.5 if I’m centered on the line, < 3.5 if I’m drifting to the right (i.e., the left IR are lighting up) and >3.5 if I’m drifting left (i.e., the right IR are lighting up). This is where you would normally do a PID calculation to adjust the left/right motor speeds. I’ll do that later. For now, I just make a small left or right turn, if needed, and continue on to the loop again. If I’m still off, I’ll make another correction, etc. With the PID algorithm, It’s possible that I never correct enough in time. So far, experimentally, my corrections seem to work well enough for now. So, make a left or right correction, if necessary and just continue in the same state. break.

FIND\_LINE\_END

I’ve found the beginning edge of an intersecting line, I want to find the trailing edge.

Call ImageLine(). This function is called through every loop when you are either crossing a line, or are within a short distance on the other side. ImageLine() uses the state value to decide to look for one of two things: if I haven’t seen the trailing edge of the line yet, count how many times I see the line with the leftmost IR and the rightmost IR, and count how many times I made that test, if I have see the trailing edge of the line, count how many times I see the line with the center IR and how many times I made that test. What is going on here is I want to know if the intersecting line goes to the left, to the right, or if the current line continues, or any combination of them. The robot may be slightly askew on the line, so simply taking the first reading isn’t reliable. I know that with the speed the robot is going, and the published width of the line, and how often the strategy loop executes, I can get a doze or so samples as I cross an intersecting line. That’s fine for left/right detection, but I also want to know if the current line continues ahead as well. I don’t want to stop to robot just as I discover the trailing edge as I want to robot to stop when it’s close to centered on the intersecting line so I can spin in place. So, as I cross the line, I keep sensing just the center IR to figure out if the current line continues. For each direction (left, continue, right), if the number of times I sense the line is more than half the number of samples I took for that sensor, I believe there is a possible line in that direction.

So, I sample the left and right IR while crossing the line in this state. Once I discover that neither the left nor right IRs see the line anymore, I believe I’ve found the trailing edge. In that case, capture the odo distance of the trailing edge (I already captured the begining of the line odo). Now I have a width of the line—if it’s not reasonable, I plan on ignoring the line and continuing on (not in the code yet).  As part of the error handling, if I don’t find the line end within about 5 inches of the start, I stop the robot in an error state. So, having found that neither the left or right IR sense a line, set the state to FOUND\_END and break;

NOTE: You should think about all the ways the robot can fail and try to decide strange state and do something about it. For instance, when I’m trying to turn 90 degrees to the left to continue on an intersecting line, If I come within a small number of degrees of a complete circle, I give up trying to find the line and stop. A smart robot will have a fall back position for every error possibility. I don’t. Not yet. Always assume that what you expect to happen won’t. Motors fail, controllers fail, sensors give false readings, motors spin without the expected movement, etc.

FOUND\_END

I’ve found the trailing edge of the line so here I capture the real odo of the trailing edge. If the line with  is at least kODO\_WIDTH\_OF\_STOP\_SYMBOL odos, I believe I found the stop symbol and set set the state to SOLVED and break. Otherwise I set the state to PRE\_ROLL and break.

PRE\_ROLL

I’ve found the trailing edge but I don’t want to stop immediately. I want to stop at a place where if I turn left or right in place (spin in place), I will be close to the latitude where I can continue on in a straight line along the intersecting line. I also need to discover if the current line continues on past the intersection. I did some crude experimentation and found that I need to keep going another kODOS\_TO\_OVERSHOOT\_LINE odos to reach that point so that after I’ve turned off the motors, I’m ready to spin in place. I captured the odo where I found the trailing edge of the line and now I stay in this state until I’ve gone at least kODOS\_TO\_OVERSHOOT\_LINE odos more. After than I set state to STOPPING and break;

STOPPING

I’ve reach a point where I want to stop moving forward. I could just execute a turn immediately, and that may be good, but for now I come to a complete stop first. Each time I go through the strategy loop, I keep track of  the odo value at the end of the previous loop. At the beginning of this state, I compare the current odo value to that previous value. If they are not the same, I haven’t come to a complete stop yet and I just stay in this state. Once the robot has completely stopped, I then look at my sense of whether I found a left, right or continuation line. For now, I’m just doing a left-handed tree search, so I build some mapping info (not needed yet) and then take the first possible new direction in the order of left, continue or right by setting the state to LEFT\_TURN\_YAW, FOLLOW\_LINE or RIGHT\_TURN\_YAW. Remember that I won’t find a dead end here as that was detected in the FOLLOW\_LINE state above.

LEFT\_TURN\_YAW or RIGHT\_TURN\_YAW

I’ve come to an intersecting line and positioned the robot so that a spin in place should get me close to being over the center of the intersecting line. Spinning in place is pretty imprecise, so I do it in two steps. I may have the sensors on a continuation line and need to turn left or right and then start sampling the IR until I find a line. I’m already on a line, so trying to sense the new line will fail immediately. So I want to turn about some amount blindly and then start looking for the line. By experiment, I found that if I turn 45 degrees of the planned 90 degrees in the desired direction and then begin looking for the line, it works out pretty well. So in the STOPPING state, I captured the original heading. In these states, I compute the “goal yaw” as 45 degrees from that in the desired direction. As long as I’m not beyond the goal yaw, I make a slight turn in the correct direction and stay in this state. Once I’ve reached the goal yaw, I set the state to either LEFT\_TURN or RIGHT\_TURN and break.

LEFT\_TURN or RIGHT\_TURN

I’ve spun around until I’ve nearly made a 90 degree turn. Now I make a quick calculation of the heading to see if I’ve spun too far. If so, I stop in an error state. Otherwise I keep turning in the desired direction until the line begins to appear in the IR in the expected direction. Once I see the line appear, I have found that I can just go forward again and the normal correction (PID) will take care of things. So, once I see the line in the IR I change the state to FOLLOW\_LINE and break.

DEAD\_END\_MIN\_YAW

This is like doing a left or right turn except I need to make a 180 degree turn instead of a 90 degree turn. The only difference is the blind yaw amount I make before I begin looking for the line. I blindly turn kTURN\_AROUND\_DELTA\_YAW degrees to the left and then set the state to LEFT\_TURN to complete the turn and continue on.

STOP

This state is for when something wonky has happened. It stops the motors and the outer loop stops doing interesting things.

SOLVED

This is the big ticket. It’s like STOP except it prints a different message.