COMPX349 Assignment 3 Report

Joel Howse: 1507593 – Elliot Boreham: [not sure]

# Algorithms

Everything we could handle with events we did. There are event listeners registered for when each of the line sensors changed states. We used the events to update global state, and then call a decision routine which decided how to react to the current state of the system.

The decision routine aimed to keep the robot over the line, that was simplest. The other option would have been to follow an edge of the tape, but this would have caused more complications when trying to detect intersections.

The largest challenge we faced was differentiating between sharp corners, and intersections. In the end we implemented a routine which the robot would execute each time it came off the line. When it off the line we assumed it was in one of three states:

* At an intersection
* On a sharp corner
* At the end of the tape/line

We mainly focused on differentiating between an intersection and the latter. If at a corner or at the end of a line, it was easy to try one way and then if that failed after turning a certain amount just turn the other way until it found a line again. This solved most of the issues we encountered. To detect an intersection when we came off a line, we moved the robot forward a bit and checked if we detected a line again. If we found something then we assumed we were at an intersection, otherwise we would fall back to the corner/end of line logic.

Once an intersection it was easy to perform the intersection specific logic described in the assignment brief. Either keep going straight or backup and turn either left or right (depending on what the state was, simple to alternate by toggling a boolean).

Some notes of the above algorithm:

* The largest issue with this approach is sometimes not detecting the intersection. Because the line sensors are right on the edge of the tape, it is possible for it to not detect double white but instead catch the edge of another part of the intersection, detecting a corner. In our final demonstration we made sure the intersections had no overlaps on the edge to avoid this problem.
* We would disable any other action happening from other threads while we were detecting an intersection so it wouldn’t be interrupted.
* We experimented with the compass but found it inconsistent to work with and having to calibrate it on each start slowed down our ability to iterate. You were able to set the calibration programmatically, but it was rarely consistent.

# Advanced Options

We implemented the sonar sensor to detect objects. It was simple to implement and worked consistently. We ran the check 4 times per second by using the microbit timer, and then had some global state switches we could set to disable driving.

As a bonus we also implemented some logic that made the robot go backwards if an object got too close by re-enabling driving and reversing the definitions for the directions (forward became backwards, left became right and so on) which worked perfectly! Struggled going backwards around sharp corners though.

# Group Roles

Most of the development was done together. Elliot had a lot more experience with the sensors and the task in general - he had built a line following robot in a previous paper – so was able to provide wisdom around how we tackle the problem. Joel was confident and experienced with software and was able to write and organize most of the code, deferring to Elliot when there were issues and confusion. The roles were fair - working to everyone’s strengths - and the group functioned well.