Group 53 - Wall-Follower
Alessandro Commodari 260636932
Asher Wright 260559393

Lab 1: “Introduction to Embedded Systems” Report

Data Analysis

The bang-bang controller mostly kept the robot at a distance bandCentre. Although it was not always precisely bandCentre away, its average distance away was bandCentre. This is due to it oscillating from one side of the bandwidth to the other. To explain, as soon as it gets outside of the band, it will correct itself by going towards the center of the band. In doing this, it will likely overshoot to the other side. Then it will again try to go back to the center, likely overshooting again. This is the reason for the oscillation. For PController, this oscillation was smaller in amplitude, since it’s adjustment speed is lowered as it approaches the center of the band. This results in a smaller overshoot. A slower adjustment speed meant that it wouldn’t overshoot the band center (as much). Even if the robot reaches the band center and stops oscillating, changes in the wall (like a curve) mean that it has to readjust (and will oscillate again).

Observations and Conclusions

On occasion the ultrasonic sensor would display a maximum measurement in-between normal measurements, while following a solid wall. One reason this could happen is due to the fact that the sensor was angled at 45 degrees and not all signals were pinged back to the ultrasonic sensor (false negative). This error is filterable. It was filtered by ignoring maximum measurements, unless they were read FILTER_OUT times in a row (usually FILTER_OUT = 10). In other words, unless the sensor read a maximum distance at least 10 times in a row, it would ignore max distance measurements. This resulted in these false negatives (that popped up in-between other real measurements) being ignored by the robot.

The ultrasonic sensor did not produce any false positives (consistently/noticeably). Although there were other sources of ultrasonic signals (other teams’ robots), they did not appear to affect the robot. It could be possible that the ultrasonic signal from another robot hit our robot, and result in a false positive, but this was never seen/noticed.

Although not an error in the ultrasonic reading, the sensor took about 3 seconds longer to initialize than the motors. This can be corrected by making the motors stay at idle until the sensor activates. This should be implemented into the final project.

Further Improvements

There were many improvements that were made and that also could be made in the future:

1. **Adding a too-close-to-wall check**. This would be a condition where, if the robot gets too close to a wall, it reverse and rotate. For the bang-bang this was crucial (instead of trying to get away from the wall at a constant rate). For the PController, this was still important, but not crucial, as the PController would try to get away from the wall even faster as it got closer to the wall.

2. A good way to improve the robot is to **better define what it should do in different situations.** One example improvement would be to **act differently depending on what type of turn it was making.** These types could be an outside turn, at the end of a block wall, and an inside turn, like a corner**.** For example, in the PController, making a different proportionality constant for these two cases means that the robot can take inside turns very tight (better corners), while maintaining its distance on an outside turn (not skimming the wall).

3. **Adding a filter** so that the robot can ignore both false negatives and small gaps. If the robot reads a value past a pre-selected "max-distance-threshold", we can choose to ignore it, unless we read it X times in a row. This means that the robot will not react to sudden max values followed by normal values. This is a good way of ignoring false negatives. Additionally, if there are small gaps in the wall, as long as the amount of times the robot measures the gap is fewer than X, it will ignore it.

4. **Adding a second ultrasonic sensor**. Rather than having one sensor at 45 degrees forward from the wall, there could be one pointed directly at the wall, and one 90 degrees from the wall (looking ahead). The first would maintain the robot's distance from the wall, and the second would determine whether or not there was a wall or corner coming up (to avoid

crashing). This additional data would improve the robot’s ability to follow more “difficult” walls (sharper corners).

5. If the design is limited to only using one ultrasonic sensor, it could be improved by **making the sensor rotate on a motor**. This way, the robot could rotate the motor to see what is ahead (to see if it is reaching a corner or not), but it could also get an accurate reading of its distance to the wall. It could either be constantly rotating, and the code could be simultaneously calculating the distance to the wall (using trigonometry), or it could periodically rotate forward to see if a corner was coming

Other controller – differential PController:

A controller type that could potentially yield better results is a controller that is similar to the PController, but uses the **differential distance to the wall** rather than the distance to the wall. The benefits of this can be shown through an example in which the robot is placed far away from the wall, parallel. With the PController, the robot will try to rotate and move forwards towards the wall, hoping to get into the allowed band. However, if it is too far, it will rotate past 90 degrees (past facing the wall), and it will perform a U-turn, missing the wall completely. With the differential algorithm, the robot will not be increasing its turn rate to get closer to the wall, but instead will be trying to minimize (large negative value) the dDistance/dt. This will mean that as the robot spins, as soon as it is perpendicular to the wall, it will head straight for the wall (since dDistance/dt is a maximum). Once it gets closer to the wall, we will lower the wanted dDistance/dt, resulting in it turning forward more and more. Eventually, when it reaches the band, it will want to make dDistance/dt = 0. It will do this by staying parallel with the wall.

There are other benefits to using the differential distance to the wall as well. One is that it can be significantly smoother. The differential distance is calculated by taking into account at least two of the nearest points (D2-D1)/(T2-T1). We can also calculate this by using a linear fit to the last four points! When doing this, if one point is significantly different from the others (but not caught by our filter), it will not affect the movement of the robot as much. This is because, in the NON-differential mode, each point affects the robot in the same way. If the readings are volatile, this results in jerky motion. In the differential mode, each region (four points, for example) affects the robot in the same way, and the individual points do less (as they are considered with their neighbors).