ECSE 211

Lab 1: Wall Follower

Group 21
Theodore Janson (260868223)
Marwan Khan (260762251)

January 24, 2019

1.Design Evaluation

Hardware

The robot went through two hardware designs during the developmental process. In the first design, the motors were side by side directly under the brick, which made the robot tall and slightly unstable. The design was also slightly asymmetrical. Extending the robot with supports was initially considered, but a second, smarter design was finally opted for. This robot had wheels further apart, allowing for more stability, though the motors remained under the brick. Since the brick needs to be removed to replace its batteries, it was important that it be easily removable – it shouldn't have many support pieces, as they aren't necessary because the robot moves at low speeds. The brick is attached at the front of the robot by a series of L shaped connectors, so that it may easily be removed. These L shaped connectors also prevent front back movement, as well as some degree of lateral movement. Side supports were not used because they took too much time to re-install when the brick was replaced. Another concern was the wires, which could hit the wall when the robot reversed sharply in concave corners. To minimize the extra volume occupied by the wires, they were placed under the brick, so that almost no wiring extended past the brick's volume. The ultrasonic sensor had go be placed at an angle of 45 degrees at the corner frontmost and closest to the wall. When pointed directly to the side, the sensor would fail to detect wall ahead of it, and when placed facing forward, the sensor would fail to detect wall to the side of it, and would consequently turn towards it. An angle of 45 degrees optimized these two concerns, but still had problems – the sensor was detecting wall at an angle non perpendicular to the robot (not the closest distance). Placing the sensor at the back of the robot was also tested, but resulted in slower detection of objects ahead of the robot.

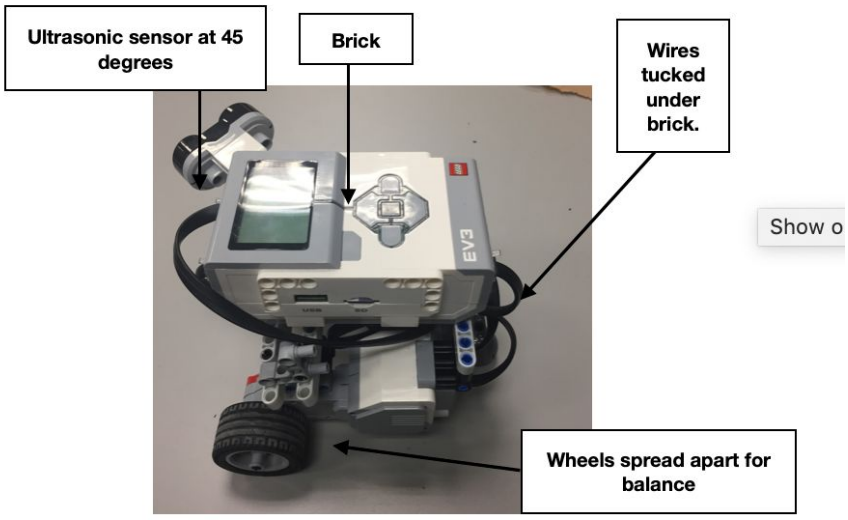

Figure 1. Robot

Software

The software is run through the main WallFollowingLab class, with multiple other threads running concurrently when started in the main method. In the main class, instance variables referring to the offset from the wall, the width of the dead band, and fast and slow speeds of the motors are initialized. Instances of the motor classes are also created. In the main method, instances of the bang-bang and P-type controllers are created. The printer thread, which is used to select and display controller type to user on brick display, and the ultrasonic poller thread, which is used to sample distance to wall every 70 ms are both started.

The bang bang controller works by making corrections of single valued magnitude based on a distance error equal to the difference between the distance to the wall and the band centre, which is the trajectory of correct direction.

distance error = (distance - band centre).

For the bang bang controller, the band centre was set to 25, which is close enough to not confuse the sensor with large distances present at gaps, along wall and around turns, but also far enough to allow for conservative maneuvering, as well as accounting for the diagonal wave projection of the sensor. The band width was set to 5 cm, to allow for more stability. The motor speeds were set to 200 and 80 degrees/second for the High and Low speeds. 200 was chosen as the high speed because after some testing, it was the speed at which the robot wasn't moving too aggressively, but could also go around the course in timely fashion. The difference in speeds between low and high is important because it affects the robot's turning ability. A difference of 120 (low being at 80) is used since it wasn't too aggressive nor too passive. The speed of the motors is controlled by four if statements, which are conditional on the robot's distance from the wall. If the distance error is less than or equal to the b\
andwidth, the width of a correct course, the robot is on the correct path, and both motors remain at high speed. If the error is negative (robot between band centre and wall) as well as above a distance of 10 cm from the wall (extreme value), the inside motor's speed is doubled and the outside motor's speed runs lower to get away. Likewise, if the error is positive, then the distance is greater than the bandwidth, and the inside motor's velocity is reduced while the outside one remains constant. Finally, if the robot is in the extreme interval (10 cm and under), the left motor works in reverse at high speed and the right one works forwards at low speed in order to reset the robot's path along the wall. The robot's turning ability is faster when close to the wall than when far to the wall. This was done to make it react fast in dangerous situation, while allowing for a more conservative and gradual turn when far from the wall, in which case a faster turn is

not necessary. In the bang bang controller, we also implemented a filter which uses if conditions, so that the robot ignores gaps. If the sensor polls distances greater than 120 cm for less than 10 polls, then the robot ignores this distance by continuing on proper path (distance is reset to bandwidth). We chose the value of 120 cm because we didn't expect the robot to be at a distance greater than it when following the wall. If the sensor polled such a distance for more than 3 polls, it considered it as a large distance, such as around convex turns. If the distance polled is below 120, the robot simply ignores it. We chose a low number because for larger number of polls, such as when the robot went around convex corners, the robot would take too long to react.

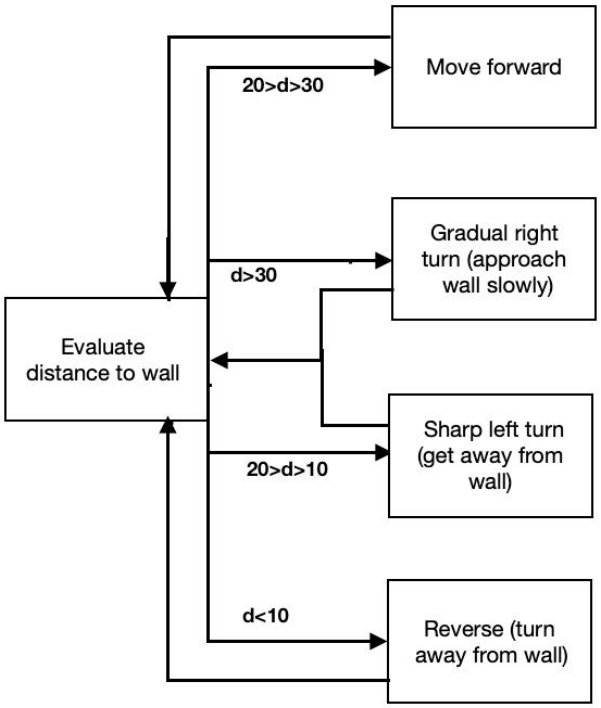

Figure 2: Controller algorithm

The p-controller works similarly to the bang bang, but the motor's speed is proportional to the robot's distance from the wall. The band center was increased to 35 for even more conservative maneuvering, and the band width was decreased to 4 because the motor's reaction when above this threshold is more proportional and stable. The p-controller also works with four if statements, which follow the exact same principles (turn out when close, turn in when far, reverse for extremely close distances). The difference is that the motor's speeds when close or far are increased or decreased with the addition or subtraction of a delta variable. This

variable is calculated by a method which multiples the offset (error) by a constant (3). So, if the distance is 50, offset will be 15 and the delta will be 45. To prevent delta from being too large (when the sensor polls large values), we bounded it too 100. Consequently, the motor speed must remain in the interval (100,300). The p-controller also used a filter. This filter worked the same as the bang bang controller's – if large distances are polled less than 3 times consecutively, they ignored as the offset is reset to the bandwidth.

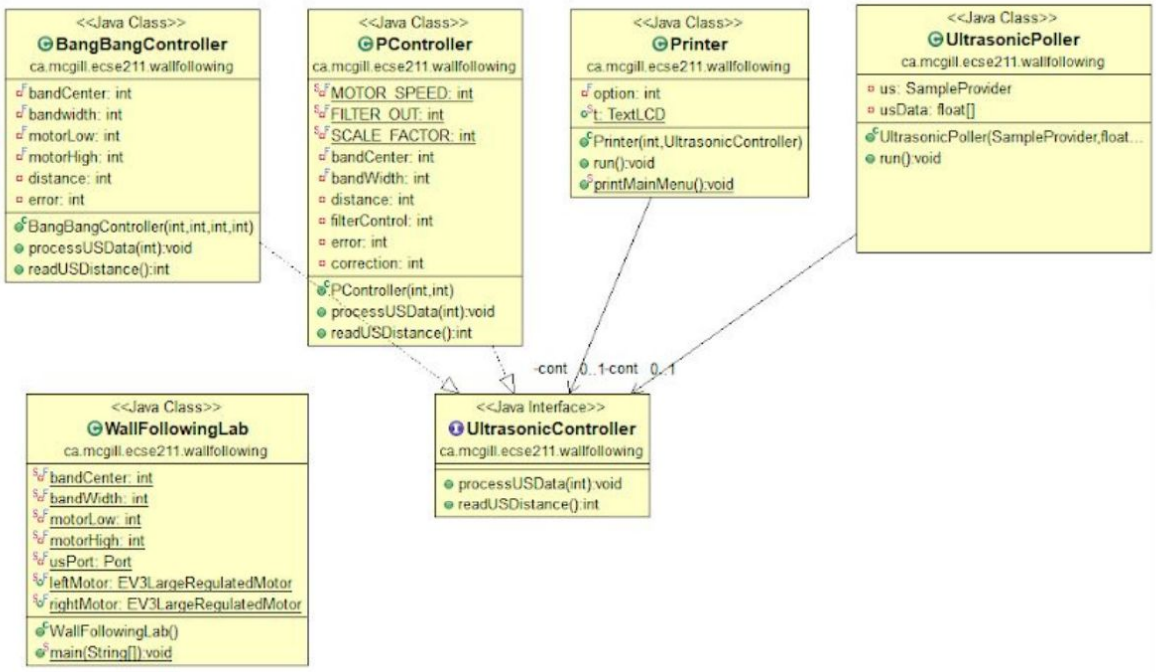

Figure: Class diagram

2.Test Data

Testing the P-type Controller constant:

In this test, we used a band center of 25cm and a band width of 5cm.

The scale factor used for the Demo was 3 so we chose 5 and 1 as our test values:

● Scale factor of 5:

The robot was a little bit faster and more responsive; However, it was over correction to the right when it should make a left turn in a corner, this is due to time it take for the filter to process the data.

● Scale factor 1:

In this case, the robot was a bit more stable, but it wasn't fixing it's trajectory enough which made it go into gaps or the wall from time to time.

Bang-Bang Controller test:

The robot made it through a full lap with all three trials, without touching the wall or going into The gap. It would oscillating around the band center most of the time but was fairly stable, however it would go to close to the blocks around concave turns or sometimes take very wide turns. It would comfortably deal with convex turns by instantly rotating back to the right path, when it got too close to the blocks.

P-type Controller test:

Our P-type controller also made it through all the trials, completing a full lap. It was oscillating a bit less than with the Bang-Bang Controller but would sometimes get too close around convex turns. The filter worked decently to avoid the gaps, it took us some time to achieve this by configuring the FILTER_OUT value and the filtering distance.

3.Test Analysis

● From the results obtained in section 2, the stability of our robot decreased when we changed the scale factor of the P-Controller, and this made it crash into the wall occasionally.

● for straight paths, our robot was staying around our band center, However, on right turns, it was going too close to the wall before correcting its trajectory.

● It was exceeding our band center by approximately 4 cm on convex turns because we made them the less sharp than concave turns, but started correcting itself after that.

● This issue occurred on both controllers for the same reason, but the P-Controller was correcting itself faster under those circumstances compared to the Bang-Bang.

4.Observations and Conclusions

1. Based on your analysis, which controller would you use and why?

It is better to use P-type. The robot had less oscillation due to the fact that the motor responses are proportional to distance from the band center. The robot remained on bandwidth more of the time by about 30%. Although the robot came closer to the wall during convex turns using the p-type, the reversing was very reliable and the robot did not hit the wall during the trials. The problem with the bang bang is that since the sensor it is placed at a 45, distances polled are exaggerated values of the actual distance. When the robot turns away from straight line by 45 deg, robot sensor looks forwards, which leads to a misreading, so that it turns back straight

away. P type improves on this by having more proportional changes in robot direction. Yes, ultrasonic.

1. Does the ultrasonic sensor produce false positives (detection of non-existent objects) and/or false negatives (failure to detect objects)?How frequent were they?Were they filtered?

Yes, but not for extended periods of time. The sensor polled false negatives and positives, but for no more than 1 or 2 polls in a row. They weren't frequent enough to pose an issue as the sensor would get a proper reading soon thereafter. False negatives and positives seemed to be more common when the battery dropped below 7.2V. There was no filter for detecting false positives or negatives because they weren't very frequent.

5. Further improvements

Software improvements to sensor

1. The first improvement we would make to the sensor would be to account for the random incorrect readings that the sensor sometimes picks up, as well as the issue that because the sensor is placed at a 45 degree angle, it doesn't detect the wall directly beside it nor directly ahead of it. The solution would be to use a servo motor to rotate the sensor 90 degrees between North and East for a clockwise path. Doing so could make the robot get more information about its surroundings and could therefore stay on the path more accurately. This could be done by rotating the sensor by 10 degrees for every 2-3 polls (similarly to how to filter works.)
2. Another solution to the random reading problem would be to take the average of a 3-5 values before making a decision on whether or not to approach or get away from the wall. This could also be done with a filter which would return a distance value calculated from an average of distance when the method is called on in conditional statements.
3. Filtering out false negatives (failure to detect an object) and positives (object not actually there) could be done more effectively using a combination of the two solutions above. If the sensor were to detect large values for 3-5 polls, the motors could be programmed to stop, the sensor could then be used to look around using a motor, and the average of the readings polled could then be used to make a decision.

Hardware improvements

1. Using one sensor looking forward and a second looking to the side would allow for the robot to get a more accurate reading of the distance to the wall, as well as be able to detect object ahead of it. The code would be a little more complexes because the robot would need to make a decision based on two pieces of information.
2. A brush could be placed ahead of the tires to clean the very dusty floor ahead of it, similarly to a zamboni. The dust reduced the friction of the tires, which gave the robot less traction.
3. If a higher quality sensor were used, there would be less errors in the readings, allowing for a more accurate following of the band center.

Other controller types

1. We could use a proportional-derivative controller (PD). The advantage of such a controller is that it is a more accurate and stable controller, so the robot would undergo less oscillation. The advantage of this controller is the derivative factor: this would limit the overshoot of the correction by slowing down the motors as the robot re-approaches the band center. This would reduce oscillation considerably.

   Source:
   "Chapter 9.3 - Proportional-Derivative Control.” Engineering360, GlobalSpec, 2004, https://www.globalspec.com/reference/51986/160210/chapter-9-3-proportional-derivative-control