Roger ZHANG ID: 260805143
Group 36

Wall Follower Lab Report

Report presented to Prof. D.A. Lowther and Prof. F. Ferrie

Design Principles and Methods
ECSE 211

Faculty of Engineering
Department of Electrical, Computer and Software Engineering
McGill University
September 21st 2018

1. Design Evaluation

1.1 Description of the design

Hardware design

The robot's hardware design is very simple. It mainly consists of the EV3 brick laying on top of 2 large NXT motors secured on the sides by Lego structures, as can be seen in Figure 3. To prevent the bottom of the motors from rubbing the floor and causing unwanted friction, a small metal ball was added at the front of the robot, as can be seen at the bottom of Figure 2. The large NXT motors were chosen instead of the medium ones because they are better suited to move the robot around, which was the main task of this lab. An ultrasonic sensor was finally added at a 45-degree angle with respect to the brick. This particular angle grants the robot the ability to see a portion of both its front side and left side, allowing it to detect concave corners efficiently.

Software design

First of all, the offset from the wall was set to 30 cm. It was selected because this distance is large enough to allow the ultrasonic sensor to have a big enough field of vision to ignore the gaps between the blocks. However, it is small enough for the robot to recognize the layout of the blocks and follow the wall continuously. The width of the dead band was set to 1 cm so that the robot would correct its trajectory as soon as it deflected from its path.

Bang-Bang controller: The Bang-Bang controller "motor low" and "motor high" speeds were set to 100 and 175 deg/sec respectively. This slow speed was chosen to allow the ultrasonic sensor to give out more readings for a set distance and therefore filter out any faulty data. The Bang-Bang controller has 4 different speeds and wheel behaviours depending on the distance error (the difference between the offset from the wall and distance red by the ultrasonic sensor).

Firstly, if the distance error is below or equal to the dead bandwidth (1 cm), the robot is on the right path and the speed of both motors is set to "motor high" so it goes in a straight line. Secondly, if the distance error is positive and exceeds the dead bandwidth, it means that the robot is too close to the wall and needs to turn right. Therefore, the speed of the motors is set to high on the left and low on the right. A constant of 125 deg/sec was added to the left speed and a constant of 20 deg/sec was subtracted from the right speed to accentuate the turn because there is no risk of crashing into a wall while turning right and it is safer to be further from the wall. Thirdly, when the distance

error is negative, the robot is too far from the wall and it needs to turn left. The speeds of the motors are then set to low for the left one and high for the right one. After testing, both speeds were diminished by 10 and 38 deg/sec respectively so that the turn would be slow. This gives time to the sensor to correct any false negative and prevent a crash. These were the only 3 cases initially, but a fourth one was added after testing. When the distance error is positive and larger than 15 cm, it implies that the robot is very close to the wall and needs to make a sharp turn to the right. Therefore, the right motor is set to high speed, but backwards, so that the robot doesn't advance anymore while proceeding to the turn. The left motor is set to slow speed and a constant of 45 deg/sec is added to both speeds to accentuate the turn.

The P-type controller: The P-type controller has the same 4 speed and motor behaviour as the Bang-Bang controller, but instead of having a "motor high" and "motor low" speed, it adds a proportional value (diff) to a constant speed depending on the case. For example, when the robot is too close to the wall and needs to do a sharp turn, the diff is added to the speeds of both motors while the left one goes forward and the right one goes backward. The base speed was set to 100 deg/sec, which again is a slow speed designed to give more time to the ultrasonic sensor to make more readings. The P-type controller also implements a filter, which filters out false negatives. The ultrasonic sensor reads a maximum distance of 255 cm and returns that value when it sees nothing. Therefore, the filter takes the 10 first maximum distances and discards them to make sure that there is no false negative. It initially discarded 20 max distances before allowing one through, but it was lowered because it was thought to be too slow to detect an actual large distance.

The proportional value modifying the speed of the motors is found by multiplying the distance error by a constant called the P constant. This constant was set to 2 because it was thought to be large enough to accentuate large values and small enough to not over enlarge small values. After testing, the distance error was set to a maximum of 40 cm because values over that would add too much speed to the motors and cause slipping.

1.2 Workflow

1. Read the instructions and the requirements of the lab
2. Built a first prototype of the robot without a sensor. This prototype was built using the simplest design possible. Its goal was to drive forward without unwanted changes of direction.
3. Tested the design using a simple code that made the robot advance forward until a button was pressed.
4. Made necessary modifications for the robot to advance properly.
5. Tested the modified design with the simple "advancing forward" code. This trial was successful. An ultrasonic sensor was then added on the left-hand side of the robot, parallel to its core.
6. Tested the design using a very simple wall follower code implementing the "Bang-Bang controller". The Robot was unable to follow the wall due to the angle of the ultrasonic sensor. Made changes and tested the modified design.
7. When the hardware design seemed functional and satisfactory, started making software changes.
8. Repeatedly tested the wall following system implementing the "Bang-Bang controller" while making software and hardware modifications between each test until the wall follower code seemed functional and satisfactory.
9. Implemented and tested the "P-type controller" as part of the wall following system.
10. Repeatedly tested the wall following system implementing the "P-type controller" while making software design modifications between each test until the code seemed functional and satisfactory.
11. Tested the wall following system implementing both the "Bang-Bang controller" and the "P-type controller" on complicated sequences of blocks and made final modifications on both controllers.

1.3 Hardware design

Fig. 1. Aerial view of the robot

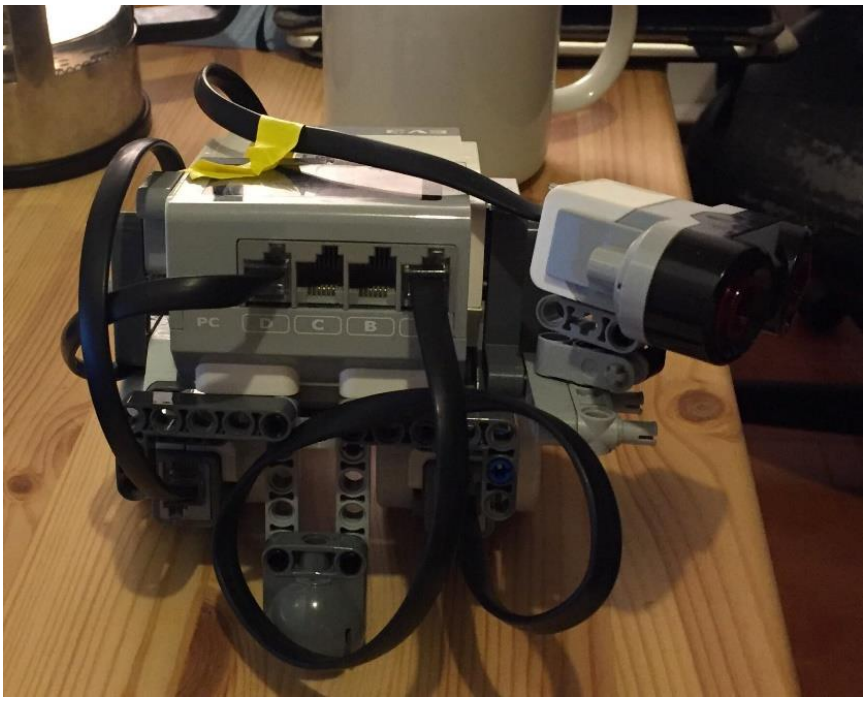

Fig. 2. Front side view of the robot

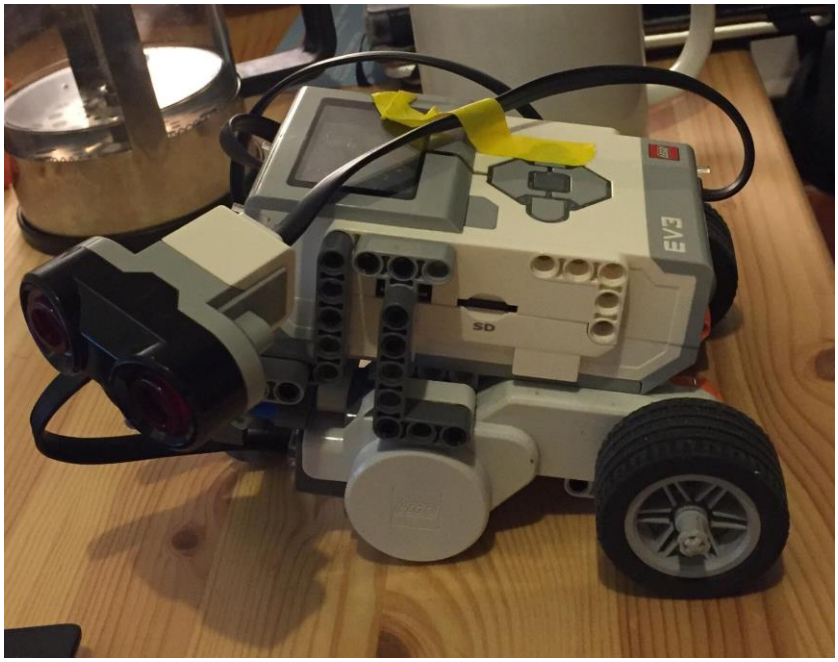

Fig. 3. Left side view of the robot

1.4 Software design

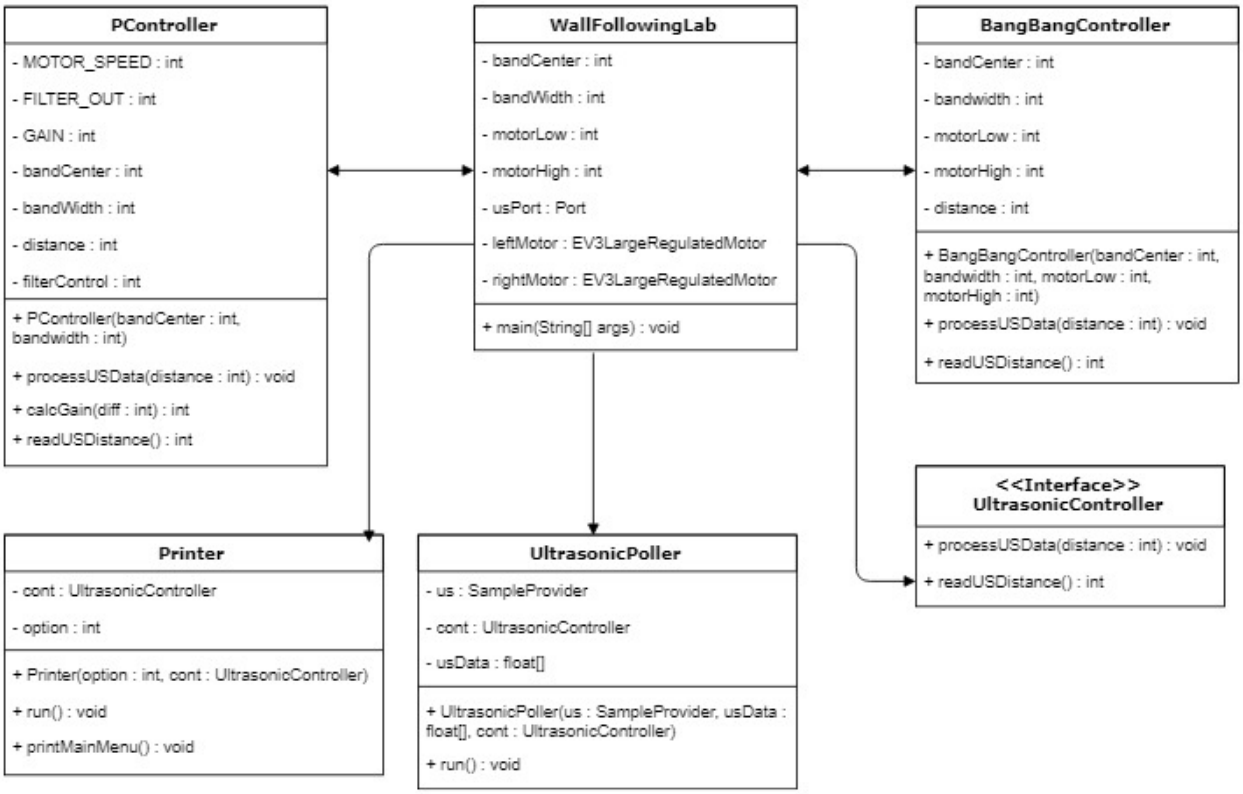

Fig. 4. Class diagram of the software

Test Data

2.1 Testing the P-type controller constant

The P-type controller constant used during the demo was 2. Therefore, a value of 6 for the value above and a value of 1 for the value below were chosen to be tested.

For the high value of 6, the oscillation was very important every time there was a variation of distance. For example, before turning a convex corner, it would follow the wall at a constant speed before starting a sharp turn left at very high speed as soon as the ultrasonic sensor stopped seeing the wall. It would then crash into the wall because of the sharp turn. Also, since the overall speed was increased, the change of directions became larger and made it hard for the robot to maintain the band center which caused a lot of oscillation.

For the low value of 1, there was much less oscillation since it would take a long time for the robot to executes its maneuvers. Furthermore, it has a much easier time staying in the band center since the changes in direction were so small and slow. It would nonetheless crash into walls because its inability to execute turns in time.

2.2 Bang-Bang controller test

The robot completed 1 trial out of 3 implementing the Bang-Bang controller. The oscillation was minimal but still present. It would appear most often while following a wall. The robot tended to slowly turn left while following a straight wall and then suddenly turn right when realizing it got too close. Once the robot started showing that behaviour, it would never turn right enough to return to its band center and would therefore continuously repeat the same behaviour and oscillate. However, when starting a turn close to a convex corner, the robot would tend to execute a large turn because of its slow speed and complete the turn in the band center. Then, the slow and constant speed allowed it to stay in the band center with minimal oscillation.

The robot failed 2 of the trial while turning around a convex corner. After starting a turn far from the wall, the robot would find itself finishing the turn too close to the wall and would fail to read the distance adequately.

2.3 P-type controller test

The robot completed 2 laps out of 3 implementing the P-type controller. There was less oscillation than during the Bang-Bang controller test. It would mainly occur for 2 cases. Firstly, when the

robot made too sharp of a turn around convex corners and realized it couldn't make it, it would oscillate and readjust itself. Secondly, after turning around convex corners, it would oscillate briefly to readjust itself and find its band center. The lack of oscillation support that the robot was efficient at adjusting its motor speeds and staying in its band center.

The failed trial was caused by gap after a concave corner. Failing to detect the corner before being very close to the wall, the robot's sharp turn right oriented its ultrasonic sensor directly into the gap, making it believe that it was a convex corner it should turn around.

Test Analysis

As recorded in the test data, the Bang-Bang controller caused more oscillation than the P-type controller. It would mainly have a hard time finding back its way to the band center once it had lost it. It would stay in the band center after starting or finishing a turn around a convex corner at the right distance but would start to oscillate as soon as it got too close to the wall. It would then exceed the band width by 10 to 15 cm and then oscillate closer and closer to the wall.

The Bang-Bang controller was especially inefficient around concave corners. While not crashing into them, the robot would get very close to the wall while executing its turn and then wouldn't be able to find its way back to the band center. This would cause frequent oscillation. However, when it was able to maintain a good distance and stay in the band center, oscillation was minimal.

As for the P-type controller, oscillation would mainly occur in short bursts. For example, after turning a corner, the robot would oscillate repetitively for a few seconds while finding its way back to the band center and then continue with minimal oscillation. It would rarely exceed its band width by more than 10 cm. The robot also oscillated briefly while passing a gap but would quickly go back to the band center.

When changing the P-type constant, there are important speed variations, especially when increasing it. The larger the constant, the more abrupt and important the changes of direction are, and the more difficult it is for the robot to find and stay in the band center. Therefore, larger constants seem to cause more oscillation. When lowering the constant, the speed diminishes which causes slower manoeuvres and less oscillation. However, the slow speed makes the robot very inefficient at making turns. For example, a low constant such as 1 causes the robot to be incapable of clearing a concave corner and consequently incapable of completing a lap.

Observations and Conclusions

Based on my observations and various test, it can be determined that for the task of navigating a series of blocks, the most efficient controller is the P-type controller. Since the speed difference is proportional, this controller is efficient in many more situations then the Bang-Bang controller. For example, when the robot is too close to the wall, the Bang-Bang controller implements the same behaviour no matter the distance error. On the other hand, the P-type controller will execute an abrupt manoeuvre and a big variation of angle if the robot is very close to the wall but a small manoeuvre if the robot is only a few cm off the band center. Therefore, the P-type controller shows less oscillation and has an easier time maintaining the band center. It is thus more efficient and better suited to complete the task of this lab.

For both controllers, the ultrasonic sensor frequently produced false negatives. This phenomenon was accentuated by the fact that the sensor would read its maximum distance (255 cm) whenever it was unable to give back a value. False negatives were rarer when the robot stayed in the band center but would occur every 3-4 seconds when it encountered complications such as gaps or corners. A filter was implemented for the P-type controller which ignored the first 10 readings of maximum distances. Another option used in both controllers was to diminish the speed, which allowed more correct readings per cm.

Further Improvements

5.1 Software improvements to improve the ultrasonic sensor errors

One possible software improvement to address the sensor errors such as the false negatives would be to stop the motors for a few seconds when a maximum distance occurs. This would give time to the sensor to collect more readings to support or discard its original reading. This improvement would implement some sort of timer (to stop the motor for a set number of seconds) and an array of readings. The software would then iterate through the array and identify the distance that re-occurs the most often as the truthful one.

Another possible software improvement would be to filter all the distances red by the sensor. For example, every distance needs to be repeated 2 times before letting it through. This would single out and eliminate a lot of errors due to false positives, false negatives and noise.

As a third software improvement, it would be possible to modify the speed of the wheels in an inversely proportional manner in respect to the amount of variation there is in the readings. For

example, while following a wall, there would be minimal variation of distance therefore the robot could maintain a fast speed. On the other, while turning a corner for example, there would be a lot of variation and therefore the robot could slow down to give the sensor time to collect more readings.

5.2 Hardware improvements to improve the controller performance

The fact that the robot was only able to see from his upper left corner was an issue. It made it difficult for it to detect obstacles directly in front of it and properly gauge the distance of a corner while turning around it. This problem could be resolved by adding a second ultrasonic on the upper right corner. This would improve its field of vision and therefore improve the robot's ability to detect obstacles. Putting the one sensor on the front parallel to the robot and another on the side perpendicular to it could be another design option.

A second hardware improvement could be to add 2 additional motors with 2 additional wheels. These wheels would improve the maneuverability of the robot and reduce the slipping. The robot would be able to make quicker and more precise turns, which in turns reduces oscillation and makes it easier for the robot to stay in the band center.

Thirdly, a final hardware improvement could be to put the sensor closer to the ground. During the demo, the sensor was the highest component of the robot. Its altitude possibly caused some of the signals to go over the wall, which increases the probability of the sensor returning a null value. Therefore, putting the sensor closer to the ground would increase the probability of signals returning and providing a correct distance.

5.3 Alternative controller types

Instead of a Bang-Bang or a P-type controller, a proportional-integral-derivative (PID) controller could also be used. Similarly, to the P-type, it considers the distance error and applies a proportional correction. However, if that is not enough, the PID controller will also use an integral and a derivative term to correct the error. The integral "increases action in relation not only to the error but also the time for which it has persisted.[1]" Thus, if the error persists, the derivative will help the

[1] "PID Controller." Wikipedia, Wikimedia Foundation, 27 Aug. 2018, en.wikipedia.org/wiki/PID_controller#Control_loop_example.

proportional value correct the error. As for the derivative, it looks at the rate of change of error, trying to bring it to 0.