Group 26 ECSE-211
Rene Gagnon 260801777
Lily Li 260664379

# Lab 1: Wall Follower

## Section 1-Design evaluation:

We figured that best approach to this lab was to keep things as simple as possible, so we used a basic design to build our wall fallower. We only used essential components in our design: an ultrasonic sensor, two motors, an EV3 brick and a couple of LEGO pieces. The goal of this lab was to build a robot that could follow a wall. So, from the start we knew we had to build a robot who was able to go forward, backward, left and right. The robot also needed to be as small as possible. Having a robot with a small width and length allows for better turns in a small environment (concave corner for example). We searched a bit on internet and found a good three-motor chassis design[1] that met these requirements and was built using LEGOs. To evaluate the distance between the robot and the wall, we implemented an ultrasonic sensor. We added a small arm to the front of the robot to hold the ultrasonic sensor. The sensor was oriented at a 45 degrees angle so that the robot could also see ahead of itself.

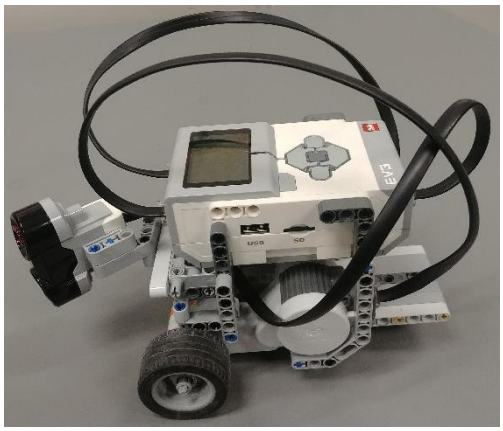


*Figure 1: Left side view of our robot*

*Figure 2: Design on which our robot is based on*

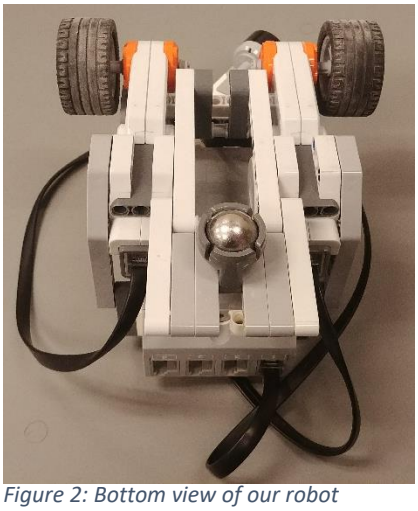


*Figure 2: Bottom view of our robot*

Once the robot was built, we started to work on the controller. An EV3 brick was used as a controller. The brick can execute programs written in Java, so we wrote code and modified already existing classes to program our controller. We wrote two different types of controller for this lab, a Bang-Bang type and a Proportional type (P-type). For both the Bang-Bang controller and the P controller there are three main cases based on the distance of the robot to the wall. First case is that the robot is within the range of the bandwidth, that is the absolute value of the distance and the bandCenter is smaller than the bandwidth. For this case, the robot is going to continue to move forward. The second case is that the robot is too close to the wall. In this case, there are two subcases: first, the robot is too close to the wall that there's not enough room to just turn to the right. In this case, the robot will go backward to move away from the wall. Second, the robot is close to the wall but not yet hit the wall, in this case, the robot simply turns to the right. The last case is that the robot is too far from the wall, then the robot turns to the left to move closer to the wall. The difference between the Bang-Bang controller and the P controller is that using the Bang-Bang controller, the robot is always moving at a fixed speed despite the distance of the robot and the wall, but using the P controller, there is a method calcProp to calculate the speed that the robot should change to according to the distance of the robot and the wall.

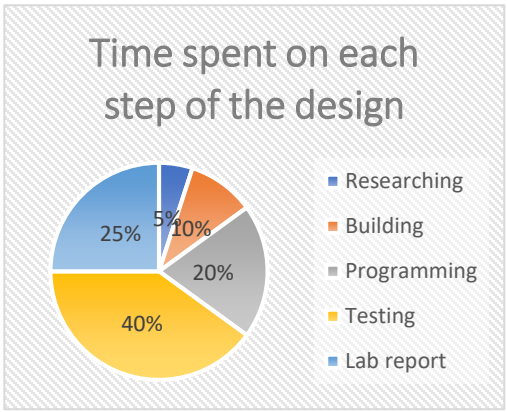
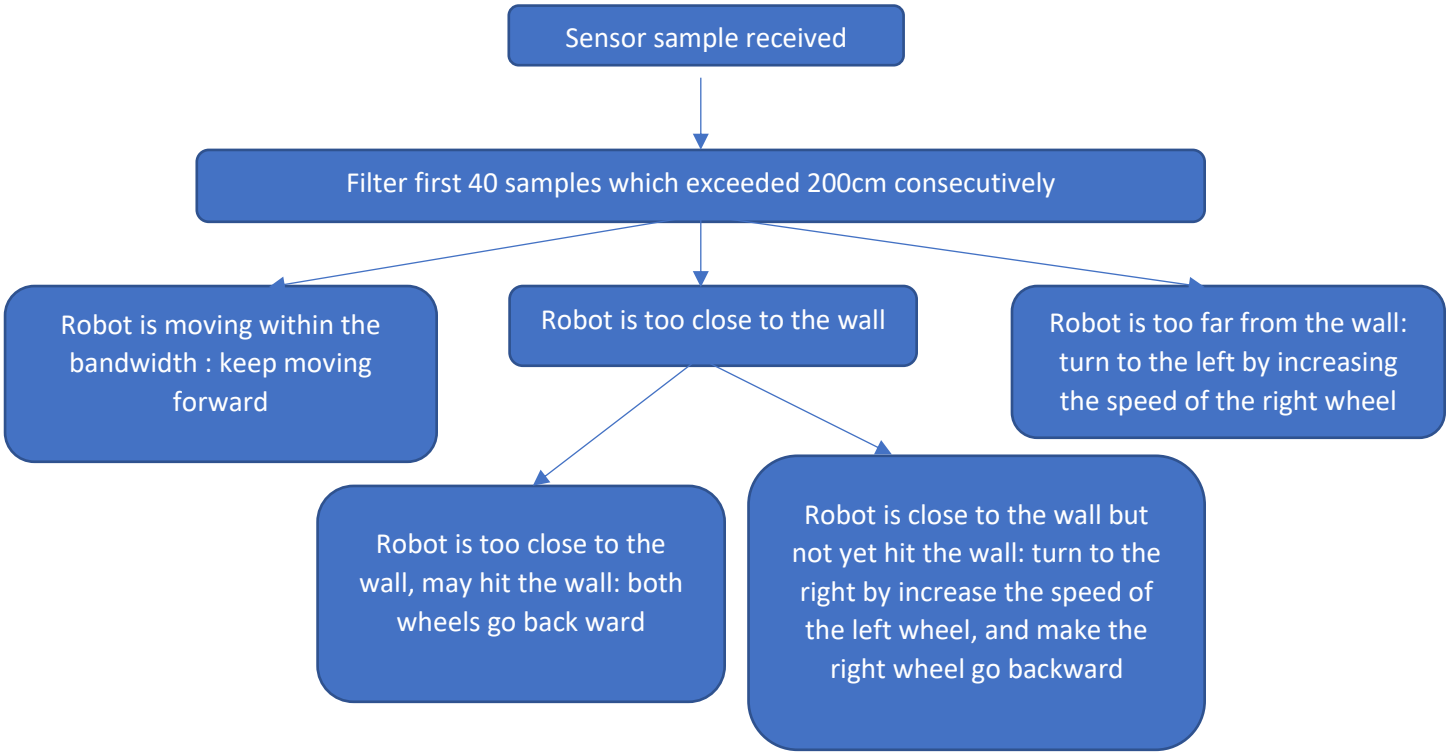



*Chart 1: Pie chart of time distribution*

[1]Nxtprograms.com, 3-Motor chassis, http://nxtprograms.com/3-motor_chassis/steps.html, January 18, 2018

Then we started testing the robot. We ran a multitude of test on our robot. For example, we ran the robot in different configurations of walls and repeat the runs to make sure the robot was consistent. We also gathered data as we tested the robot. This data was used to tune the constant more effectively. The last step was to write a report about the lab. In summary, our workflow was separated in five steps: researching, building, programming, testing and writing a report.

Software design chart:




## Section 2-Test Data:

Testing the P-type controller constant:

| Test | Proportional constant | Successful laps for a L-shape wall | Problems (if any) |
|---|---|---|---|
| 1 | 2.0 | None | Hit the wall |
| 2 | 6.0 | None | Hit the wall |

In these two tests, we tried to change the value of the proportional constant in the P-Type controller. When tuning the P-Type controller, we found that a constant of 4.0 was working the best. Test 1 was conducted with a lower constant of 2.0 and test 2 with a higher 4.0. Both tests were unsuccessful. In test 1, the robot failed at a concave turn. The robot was turning too wide and hit the wall when facing a concave turn. The wider turns could be associated with a lower constant. For test 2, when the robot faced a convex turn, it made a very sharp turn and hit the wall. This sharper turn could be associated with a higher constant. When observing the behavior of the robot along a straight wall in test 1 and 2, we found that they both followed and very similar trajectory. The trajectory was relatively smooth compared to the BangBang controller and did not oscillate too much along the bandcenter. It was very similar to what we observed with our tuned constant.

Bang-Bang controller test:

| Test | Battery voltage | bandCenter (cm) | bandWidth (cm) | motorLow (deg/sec) | motorHigh (deg/sed) | Comment |
|---|---|---|---|---|---|---|
| 1 | 7.2 | 35 | 3 | 140 | 220 | 2 laps successful, but sharp convex turns are not as smooth as the concave turns |
| 2 | 7.3 | 32 | 3 | 140 | 200 | 2 laps successful with smoother turns at convex corners |
| 3 | 7.4 | 36 | 1 | 140 | 220 | 2 laps successful |

For the Bang-Bang controller, the table above shows three tests we did during testing. One of the problems we found in the first test shown in the table is that even though it is able to run 2 laps successfully, it turns really close to the wall at a convex turn. The reason is that the ultrasonic sensor is placed at a 45 degrees angle in front of the brick so that it is seeing ahead of itself, so the robot is moving into the wall before moving to the end of the wall. To resolve this issue, a filter similar to the one in the P controller was added to make sure there is enough space before the robot makes a turn. This filter is also used to avoid gaps. For Test 2, the bandCenter is reduced to 32 cm, and the motorHigh speed is reduced to 200 deg/sec. With these values, the robot is running smoother at convex turns because of the reduced speed. Since bandCenter and bandwidth are being used for the P controller, we have to find the right values that work for both controllers. The third test is the value we used in the end. A significant change is discovered while reducing the bandwidth, which is that the robot is running a lot smoother than it used to do. The nature of the Bang-Bang controller is that it's almost always going to overcorrect from one extreme to the other, but with a small bandwidth, this nature of Bang-Bang controller is not as obvious as before.

P-type controller test:

| Test | Battery voltage (V) | Constant | Filter Out value | Bandcenter (cm) | Bandwidth (cm) | Motor speed | Max correction (deg/sec) | Successful laps for a L-shape wall | Problems (if any) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 7.0 | 6.0 | 60 | 32 | 3 | 180 | 50 | None | Hit the wall |
| 2 | 7.5 | 5 | 40 | 35 | 3 | 160 | 50 | None | Hit the wall |
| 3 | 7.5 | 4 | 60 | 36 | 1 | 160 | 50 | 3 | Turns a bit wide |

We ran approximately 25 test runs for our P-Type controller. The table above presents the data from 3 of these tests. Test 1 was one of our first test with the P-Type controller. The robot was working fine along straight walls and concave corners. It had a very smooth trajectory and did not oscillate too much. But as soon as it faced a convex corner, it hit the wall. The robot took the corner too sharply and crashed into the wall right after the corner. We made some adjustments in the subsequent runs to address this issue. We ran Test 2 after reducing the constant, decrease the filter out value, increase the bandcenter, and reduce the motor speed. All these adjustments were made to increase the performance and reliability of the robot around convex corners. Test 2 was also a failure. The robot managed to clear one convex corner but failed at the second one. The behavior elsewhere along the wall was still fine, so we concluded that we were in the right direction. A couple of test and tuning later, we finally had perfect run. In the test 3, the robot completed 3 successful laps of our L-shape wall. The only thing that was not quite perfect was again the convex turns. This time they were a bit too wide. We choose to leave this unchanged since it was safer, and we were not evaluated on the quality of our turns.

## Section 3-Test Analysis:

When the P-type constant was different than the one used in the demo, we observed more or less sharp turns depending if the constant was higher or lower. For the trajectories, we were not able to associate a clear behavior (underdamped, critically damped, overdamped) with these constants, since they were both very similar to the trajectory associated with the constant used in the demo. The trajectory observed with the demo constant was a bit underdamped (it oscillates a bit around the bandcenter). Also, the wall configurations we used never featured long straight walls, which could have been better to observe behaviors.

The oscillation around the bandcenter depends on the bandwidth. A smaller bandwidth results in less oscillation. The robot follows more closely the bandcenter. On the other hand, a bigger bandwidth results in more oscillation, because the robot corrects its trajectory less often. This result in more deviation from the bandcenter.

When the correction applied to the trajectory was to big, we exceeded the bandwidth. The faster the robot moved the more it exceeded the bandwidth. For example, if our robot was too close to the wall, with high speed the robot would over correct its trajectory and go to far from the wall.

For the BangBang controller, the robot tended to exceed the bandwith more. For the P-type controller, the robot exceeded it less.

## Section 4-Observation and conclusions:

Based on our observations and analysis, the P-type controller is the controller we would prefer to use. In our testing it was the easiest to tune, was the most reliable, seemed to adapt better to different wall configuration and had the smoothest behavior.

Our ultrasonic sensor had regularly problems when measuring distances. First, sometimes when starting the code, the sensor would measure a distance of 0 cm even when there was nothing in front of it. These false positive would often go away after the code ran for a couple seconds. The sensor also measured false negatives from times to times, but we would filter these values out and they were never processed by the controller. We could filter these out because they were only measured for a brief amount of time. So, when a value was very high, the filter would wait 40 samples or 2.8 seconds before sending the value to be process. A false negative, being very large and only there for a brief period, was then filtered out and ignored.

## Section 5- Further Improvements:

Software improvements:

1. Increase the sensor polling rate so that data is received faster. To achieve that, the Thread.sleep() method in the UltrasonicPoller class could be decreased or eliminated so that the sampling rate is higher, so that the robot is able to react fast to turns and gaps.
2. A problem we saw during testing is that since the ultrasonic sensor is facing at a 45-degree angle, it sees ahead of itself therefore leads to a sharp turn at convex turns. A rotating ultrasonic sensor can be implemente to see both ahead and on the side.
3. For the P controller, a constant value is used to calculate the correction of the speed, this could be improved by using another function without the constant value.

Hardware improvements:

1. The arm, on the front of the robot, holding the ultrasonic sensor was bending. This was an issue since the captor was facing a bit downward and could measure the distance to the ground instead of the wall. This could have been fixed by placing the sensor higher off the ground and directly on the robot chassis. This would allow for a better orientation of the sensor and more stability.

2. The robot, although not very big, still measured 14 cm wide and 16 cm long. With a bit more, research and more time spend on designing the robot chassis, we can surely decrease these dimensions and have a robot with a smaller footprint. The smaller size would allow for more maneuverability and would decrease the chances of the robot hitting the wall.
3. Add another ultrasonic sensor to the back of the robot so that the robot does not hit the wall from the back while backing up from the wall in front of it.

An alternative to the P-type or BangBang controller could be a modified version the P-Type controller. In the normal P-Type controller the relation between the distance seen by the robot and the correction is linear (we use a proportional constant). But we could try use other type of mathematical relationship such as exponential, inverse, quadratic, etc … and implement another algorithm to determine the relation between the distance and the correction.