**Lab 1: Wall Follower**

Report written by  
Yicheng Song (ID: 260763294)  
and Tian Han Jiang (ID: 260795887)

As part of Group 51

For ECSE 211: Design Principles and Methods

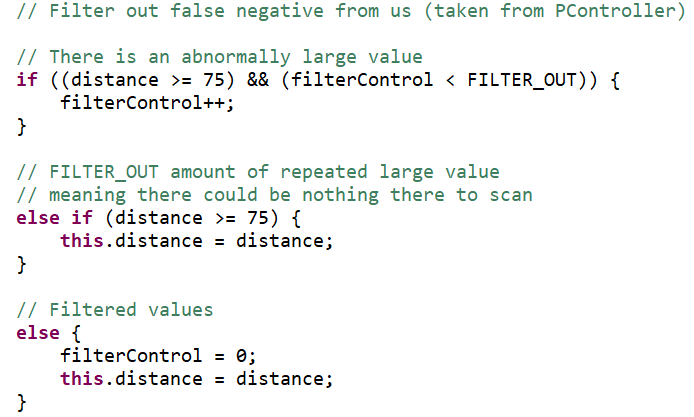
Due January 24th at 11:59PM

McGill University

## Design evaluation

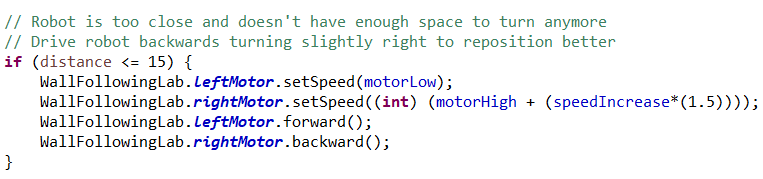
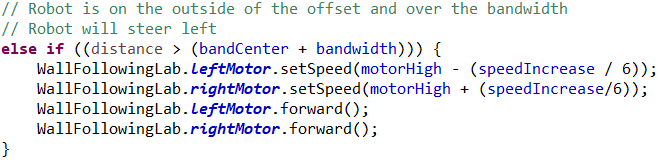
The robot as well as its controller are specifically designed for implementing the wall-following system.

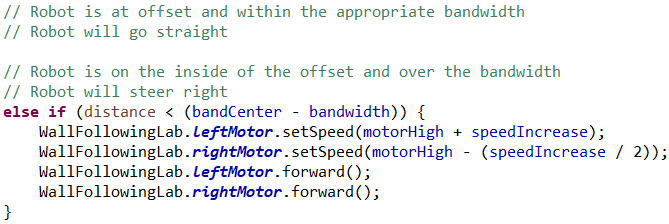
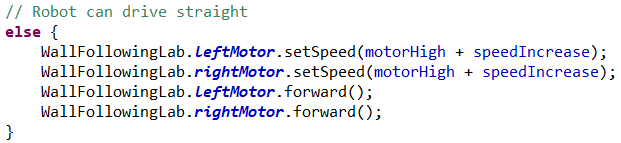
The robot is driven by two motors on both sides of the robot beneath the EV3 Brick. There is multi-directional ball bearing at the tail of the robot which helps with stabilizing the craft and helps support the weight of it. By placing the ultrasonic sensor at the front with a 45-degree angle from the left, the robot can measure its distance to the wall, which data will be supplied as reference to implement the speed-changing process to one of or both motors to tell the robot how to move and keep it from the wall at an approximately constant distance. Both motors together with the ultrasonic sensor is connected to the EV3 brick with given cables short and medium length cables.

The main part of the software implementation are the controller classes. For both controllers, a filter for the ultrasonic sensor is implemented to test whether the data can be used. If a distance considered too large is measure more than 20 times in a row, then there is confirmation that there are no obstacles in front of the robot. Else, it is considered a false negative error.

**Figure 1.** The filter feature

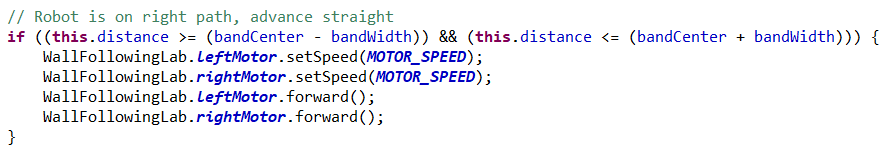
For the Bang-Bang controller, if the robot is too close to the wall, it will drive itself backwards and turn slightly right to reposition better. For those data that is within the threshold (bandwidth to the bandcenter), the robot will go straight. When the robot is on the inside of the offset and over the bandwidth, it will steer right with left wheel speed up and right wheel speed down and vice versa for the opposite situation. Figure 7 shows the steps for Bang-Bang controller through a flowchart.



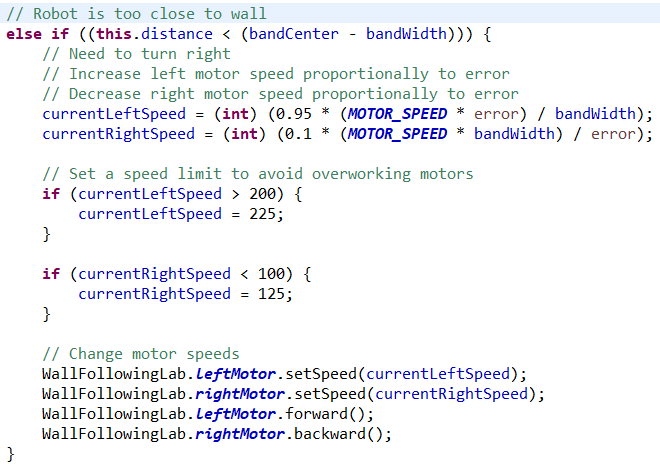
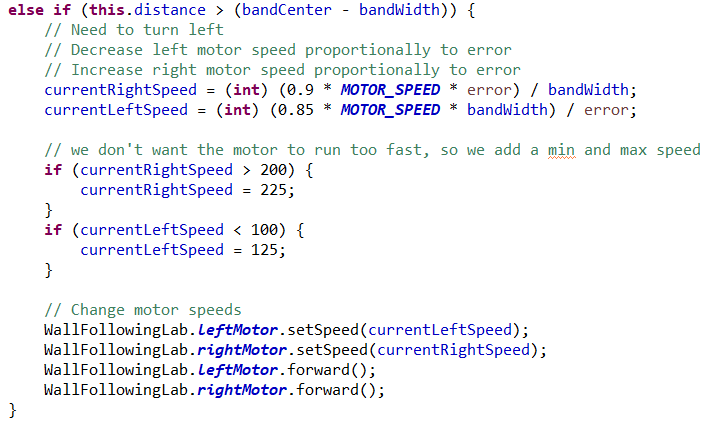


**Figure 2.** From left to right, up to down: Backwards, Turning left, Turning right and Forward implementation.

For the P-type controller, the principle is slightly different. The filter is, for practical purposes, identical to that of the Bang-Bang controller. However, there is a variable called ‘error’ that takes the difference between the band center and the distance measured by the ultrasonic sensor. The robot will move forward when on the right path. If it is close to the wall, the method that we take to make it turn right is to increase left motor speed proportionally to the error and decrease right motor speed proportionally to error. Similar method also works when the robot is too far from the wall.



**Figure 3.** Forward implementation



**Figure 4.** Turning right implementation **Figure 5.** Turning left implementation

Before reaching the final hardware design, we considered changing the position of EV3 brick but failed as there were no optimal support. The direction-control wheel occupies the space between the motors so the brick needs to be suspended above the wheel. Many bricks would be needed to achieve the goal and the supporting block seems a little bit fragile. However, we found that the robot would not be a smart choice for the test due to its large volume. The robot for the demonstration should be much smaller and be able to pass through the corners. As a result, we finally reduced the space between the two motors, put the EV3 brick upon them and change the position of the direction-control wheel at the tail and finally make it suitable for the test.

The process of finding appropriate constants of both controllers was not simple. It took plenty of time to test whether the constant worked. This is further emphasized in the process of finding suitable proportion of p-type controller. For the p-type controller, we first set the speeds of both wheels to be:

However, we noticed that the robot was constantly hitting the wall when it should be turning a full 90 degrees. After changing the bandwidth to 1cm, tests only succeeded for the first corner.

We solved this problem instead by choosing values below:

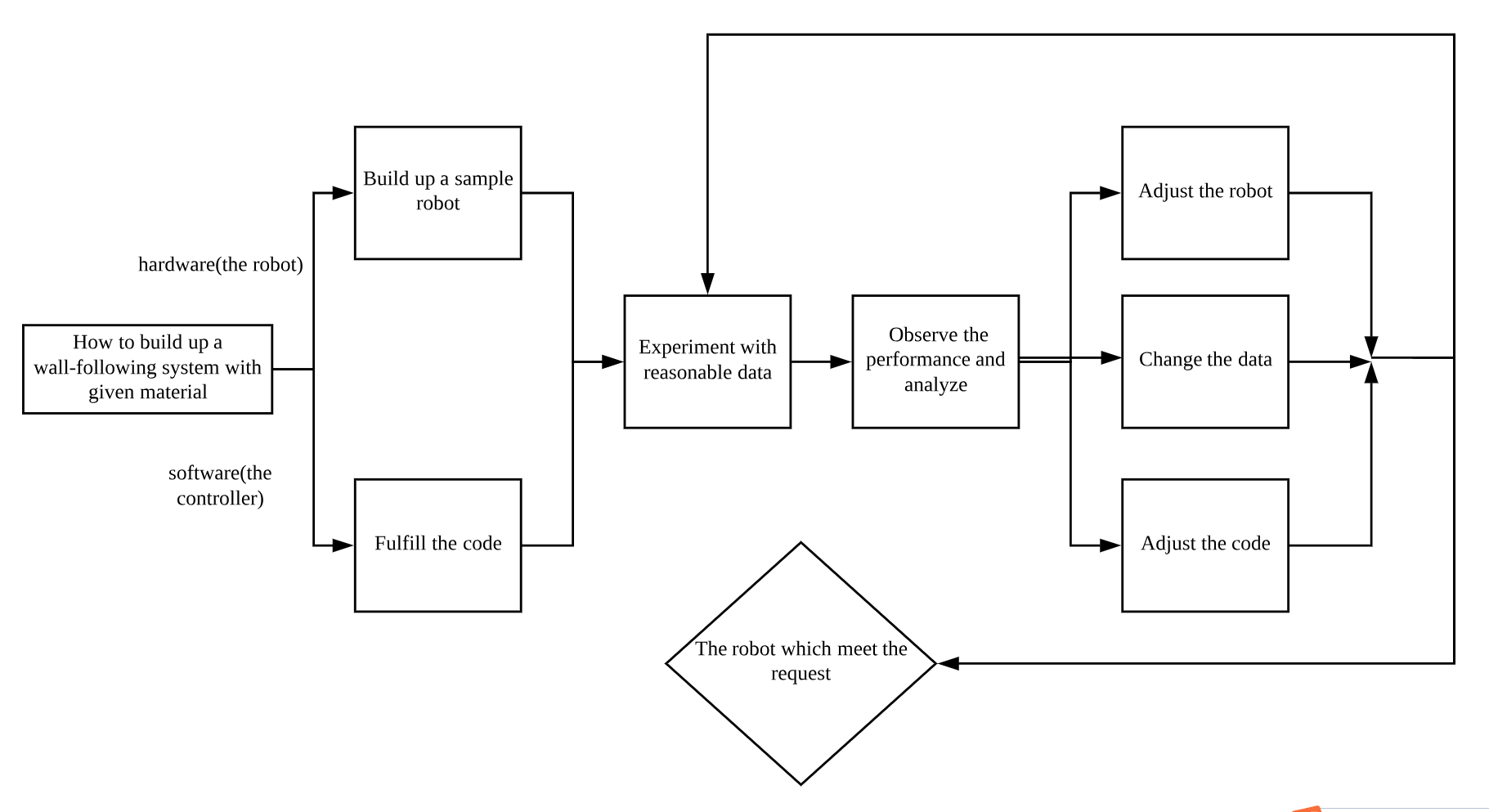
For when the robot needs to turn left:

For when the robot needs to turn right:

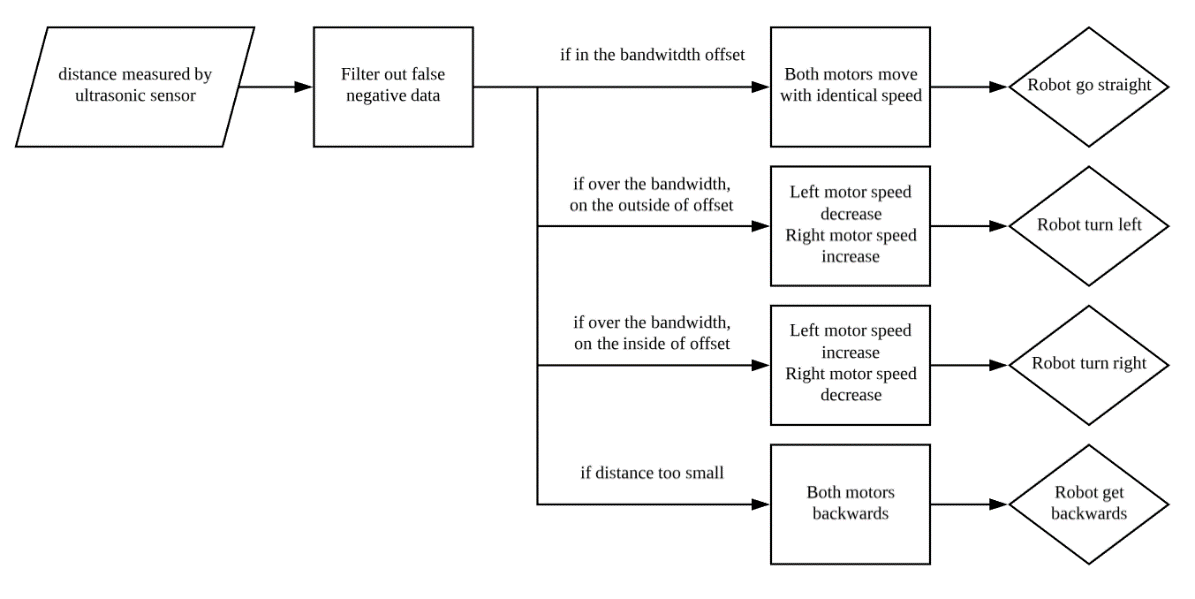
This allowed to robot to tackle left turns at a much slower and wider angle and still allow quick sharp right turns for tight spaces since the sensor is biased towards the left at a 45-degree angle and cannot sense a right turn until the robot is very close to the wall.

In summary, the design of both hardware and software parts are considered in detail. It takes us time and patience to deal with the problems we faced.

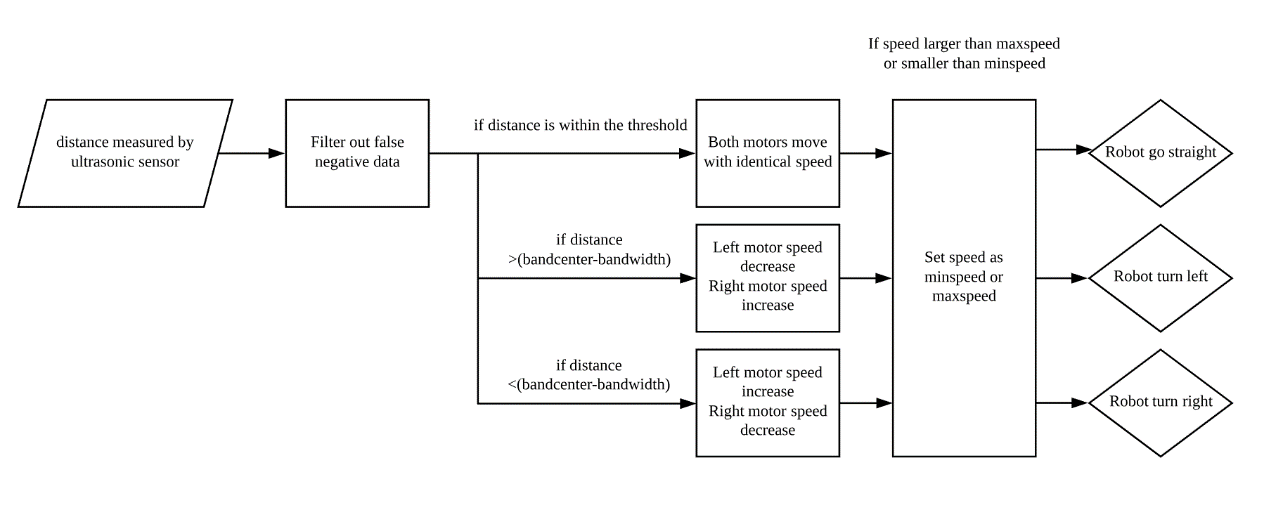
Below is a workflow that spans a week:

**Figure 6.** Workflow

Most of the software development was done within the second half of this lab with testing taking a full day.



**Figure 7.** Bang-Bang controller flowchart



**Figure 8.** P-type controller flowchart

## Test data

**Testing the P-type controller constant**

***1. Choose 2 values above and below your P-type controller constant used in the demo.***

***2. Run the robot using the P-type controller.***

***3. Note its performance, i.e. band center and oscillation behavior, for the 2 cases.***

**First test values**

For when the robot needs to turn left:

For when the robot needs to turn right:

**Second test values**

For when the robot needs to turn left:

For when the robot needs to turn right:

are chosen for the P-type controller constants. With the first p-type constants, the robot doesn’t turn quickly enough to stay within the offset and as such, spends a good portion of the distance of the next straight wall surface correcting its trajectory. With the second p type constants, the robot is prone to crashing into the wall as it oversteers itself. Band center wise, the first set of constants always results in being way too far from the band center after turns as the robot is travelling too fast compared to its turn speed. The opposite is true for the second test data as it oversteers and crashes into the wall as it turns too much and as such, isn’t consistent with its positioning. Oscillations for both cases were frequent yet for the first case, were less severe as it turned less quickly compared to the second case.

**Figure 9.** Test wall placements and initial robot position

**Bang-Bang controller test (3 independent trials)**

***1. Place the robot at the starting corner of a wall.***

***2. Ensure the wall contains convex corners, concave corners, and gaps.***

***3. Run the robot using the Bang-Bang controller.***

***4. Check if it completes a lap without touching the wall.***

***5. Note its performance, i.e. band center and oscillation behavior for each trial.***

**Table 1.** Bang-Bang controller test results

|  |  |  |  |
| --- | --- | --- | --- |
| Trials | Band center | Oscillation | Lap completion |
| Trial 1 | Can be over band center while turning a full 90-degrees. Can also be too close to wall right after a turn | Frequent small oscillations | Yes |
| Trial 2 | Can be over band center while turning a full 90-degrees. Can also be too close to wall right after a turn | Frequent small oscillations | Yes |
| Trial 3 | Can be over band center while turning a full 90-degrees. Can also be too close to wall right after a turn | Frequent small oscillations | yes |

Overall, test results are very consistent. The robot was able to lap around the configured path until turned off after 10+ laps.

Refer to Figure 9 for test configuration.

**P-type controller test (3 independent trials)**

***1. Place the robot at the starting corner of a wall.***

***2. Ensure the wall contains convex corners, concave corners, and gaps.***

***3. Run the robot using the P-type controller.***

***4. Check if it completes a lap without touching the wall.***

***5. Note its performance, i.e. band center and oscillation behavior for each trial.***

**Table 2.** P-type controller test results

|  |  |  |  |
| --- | --- | --- | --- |
| Trials | Band center | Oscillation | Lap completion |
| Trial 1 | Really close to wall after a 180-degree turn. Back to configured band center after 10 cm | Constant small oscillations | Yes |
| Trial 2 | Really close to wall after a 180-degree turn. Back to configured band center after 10 cm | Constant small oscillations | Yes |
| Trial 3 | Really close to wall after a 180-degree turn. Back to configured band center after 10 cm | Constant small oscillations | Yes |

Overall, test results are very consistent. The robot was able to lap around the configured path until turned off after 10+ laps.

Refer to Figure 9 for test configuration.

## Test analysis

***What happens when your P-type constant is different from the one used in the demo?***

The P-type constant affects the rate of turn when challenging turns. Low constants (For left turns: higher right wheel constant, lower left wheel constant, vice versa for right turns) results in a wide turning radius while high constants (For right turns: lower right wheel constant, higher left wheel constant, vice versa for left turns) results in an oversteer and after a few turns, a crash into the wall.

With p-type constants of:

For when the robot needs to turn left:

For when the robot needs to turn right:

The robot can still occasionally oversteer as shown in band center results in table 2. However, this effect is minimized and often, simply graze the left wheel on a corner of the wall before being back on track.

***How much does your robot oscillate around the band center?***

Since we modified the bandwidth from 3cm to 2cm, there is an increase in oscillations. However, these oscillations are smaller. Most of the time, for both controllers, both wheels are not at the same speed as it constantly corrects itself to be within the band center. It is important to note that oscillations are less frequent but bigger in Bang-Bang controller but more frequent and smaller in P-type controller.

***Did it ever exceed the bandwidth? If so, by how much?***

Whenever the robot finishes tackling a 90-degree turn, for both controllers, the robot takes about 10 cm of distance to revert to its offset from the wall as it occasionally gets too close to it, especially after 180 degree turns as the brick’s width occupies a large distance. Usually, it exceeds the bandwidth by no more than 5 cm.

***Describe how this occurs qualitatively for each controller.***

For the p-type controller, right turns as sharper than left turns as the constants for such turns are different to account for the positioning of the sensor. However, the width of wall blocks was not accounted for when doing a full 180 degree turn. As such, when the robot finishes the turn, it is closer to the wall than the bandwidth allows it to be.

For the Bang-Bang controller, this is mitigated by the reverse repositioning functionality coded in but as the sensor is positioned towards the left at 45-degrees. In the middle of a 180-degree turn, the sensor cannot measure the distance between itself and the width of the wall block it’s trying to round.

## Observations and Conclusions

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

We prefer to work with the bang-bang controller. At the very beginning, we should claim that both of the controllers work and it is hard to tell which one is better. The bang-bang controller is chosen just because it is uncomplicated to implement and easier to adapt for future uses. Although the oscillations are bigger than for the p-type controller, especially when the robot has just turned at the corner, the robot is always on its right path and can adjust to its right path as fast as possible if there’s a deviation. As a result, it is more flexible compared to p-type controller in many cases. Furthermore, the Bang-Bang type requires far less computation and is a solid choice for future labs where more complex tasks needs to be completed in parallel with

***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?***

The ultrasonic sensor hardly produces false positives but sometimes, there are false negatives. During our tests there wasn’t any false positives (or, if any, quite few). However, there were some false negatives. The frequency is low but when the robot is turning, the chances of one are higher. We thought the false negatives are due to the angle of the ultrasonic sensor. If the sensor face towards the wall, then the robot reacts too slow so that the it may hit the wall and the sensor fails to detect. Similar conditions appear when there’s a gap that is too large or the degree of the sensor is small. This kind of false negative error can almost be filtered out from the data in the controller class, and given an angle (for instance, 45 degree), most false detection errors are avoided. The robot does filter false negative results due the filter feature for gaps up to 15 cm.

## Further improvements

***What software improvements could you make to address the ultrasonic sensor errors? Give 3 examples.***

-A backward method can be and has been implemented to the bang-bang controller so that if the robot is too close to the wall due to the sensor errors, it can go back to the previous position and re-measure the distance in order to act as it should be.

-We can improve the filtering of data in some ways. For instance, the data to be taken to justify whether there is something to detect can be larger (larger FILTER\_OUT). It may however fail to turn if the FILTER\_OUT variable is too big.

-A stop method can be used. When there’s a data that is with large difference to the previous one (for example, is 30% smaller or larger), then the robot should stop and keep the sensor working to measure a number of data (for example, measure the distance 10 times) and calculate their variance. If the variance is smaller then a given number (for example, 1) then the data measured by sensor at the first time is correct so that the robot can take action to react. Otherwise the data that is quite different from the previous one is an error and should not be considered.

***What hardware improvements could you make to improve the controller performance? Give 3 examples.***

-Lower the height of the ultrasonic sensor so that it could measure the distance more accurately. Or we can use some other kind of sensors or a combination of different types of sensors to be more precise.

-The position where the EV3 lies could be changed in order to lower the center of gravity, and the wheels can be changed with tracks so that the robot could be more stable and has lower possibility to slip on the ground. For example, the EV3 brick could lie down at the center of those wheels. If not doing so, the robot may have the risk to rollover when implementing bang-bang controller.

-The distance between the two motors can be reduced in order to decrease the radius of rotation, which will lead to a more frequent speed-changing process which make the robot to move faster in both bang-bang controller and p-type controller. This means the robot could move along the wall more smoothly and keep its distance from the wall in a smaller threshold at the same time.

***What other controller types could be used in place of the Bang-Bang or P-type?***

We can use the derivative controller, which take the distance measured by the ultrasonic sensor and calculate its derivation of time **t** to get the how fast the robot is approaching or getting away from the wall. Then the speed could be taken as the data to change the speed of both motors (for example, when the robot is closing to the wall, find its speed and add up to the speed of left motor while reduce the speed of the right motor by the same degree).