Design Principles and Methods Michael Li and Cecilia Jiang

ECSE 211 (Fall 2020) 260869379, 260795889, Group #36

Lab 1: Wall Following

Prof. Frank Ferrie

McGill Faculty of Engineering

Due September 18th, 2019

Section 1. Design Evaluation

The objective of this lab is to build a wall following robot based on Lego Mindstorm programmable bricks. The robot is designed to accomplish the task of navigating around a sequence of blocks. These blocks form a wall with small gaps, or concave corners or convex corners.

Our design consists of two parts. First, the hardware design defines the structure of the robot and the position of the sensor. Then, the software design which consists of Bang-Bang and P-type controllers' implementation. We followed a procedural design approach, and completed the following tasks in order:

- Select appropriate robot design and assemble it pieces by pieces
- Determine angle of the ultrasonic sensor
- Write code for Bang-Bang and P-type controller
- Test and adjust bandCenter and bandWidth
- Test and adjust delta speed in Bang-Bang controller
- Test and adjust the proportional scale in P-type controller
- Improve code for Bang-Bang and P-type controller

1.1 Hardware design

We selected a basic wide but strong robot design shown in fig. 1 below. The main design objective is to ensure a robust robot structure. We use cross shape joints to connect the main brick to wheel so that they won't easily fall apart or experience minor displacement. Two front wheels are assembled to two sides of the brick and we used a single rollable steel ball as the back wheel. In fact, the reason we chose the steel ball instead of the plastic wheel is that the steel ball can rotate in all directions, thus significantly reduce sliding and friction. Besides, the height of the steel ball is perfect. After attaching it to the rear of the brick, the brick remained parallel to ground level.

For the sensor, we find that the ultrasonic sensor only functions well under stable conditions. So we use a bar and a triangle structure that stretches forward to fix the ultrasonic sensor over the main brick. Its position is fixed which is very important in order to get consistent test results. The ultrasonic sensor has a viewing range of approximately 22 degrees. Placing the ultrasonic sensor directly parallel to the "Brick" will result in the robot ignoring blocks at the side. On the other hand, placing the ultrasonic sensor directly orthogonal to the "Brick" results in our robot to ignore obstacles in front of it. To accommodate both situations, we place the sensor at approximately 35 degrees with respect to the horizontal axis.

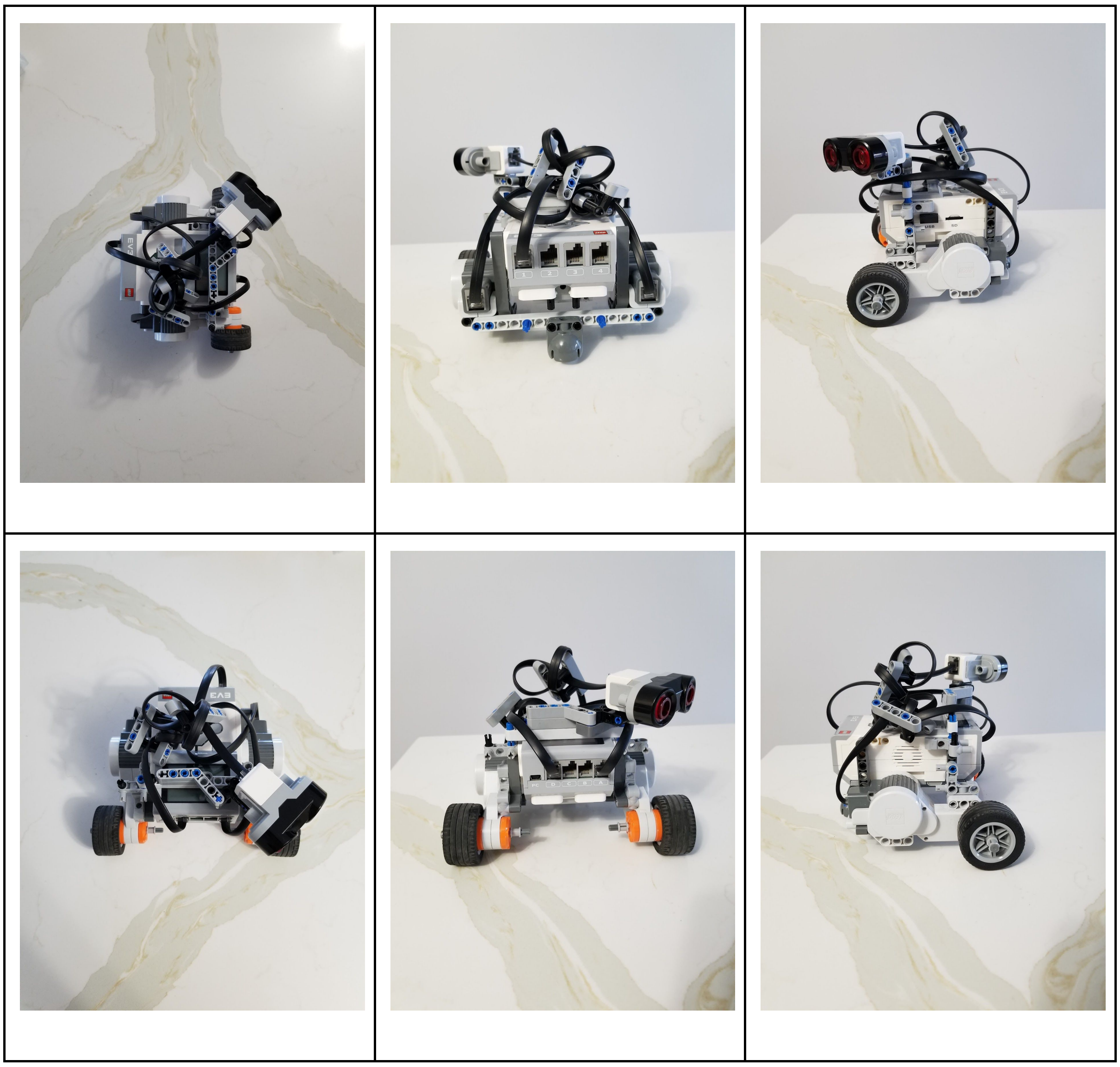


Figure 1: Robot viewed from 6 different angles

1.2 Software Design

The Bang-bang controller and the P-controller both extend the UltrasonicController class. Their difference is the way that they compute delta speed. The Printer class and the UltrasonicPoller class define two runnable threads. Resources class is an enumeration class that lists all the constants needed in controllers.

Filter method: The filter method is defined in the UltrasonicController class. It is used to eliminate randomly generated and infinitely large data. We increased the FILTER_CONTROL from default value from 20 to 30 to allow the robot to filter out the gaps. When a gap is encountered, the ultrasonic sensor returns a relatively big value (i.e. bigger than 255) which is filtered out 30 times before the robot starts adjusting itself.

Bang-bang Controller: Bang-Bang controller can be seen as an on/off switch that exerts the same delta speed onto the wheel whenever the robot drives out of the “safe zone”, which is defined to be bandcenter plus or minus bandwidth (i.e. 32 cm +/- 6 cm). In our design, we divide the dangerous zone further down into two parts In order to minimize oscillation: the one which needs gentle adjustment and the one which needs sharp adjustment. Besides, when we do a right turn, we want to avoid accelerating the left wheel, since it could touch the wall if it is too close to it. So we specified the right turn to operate in the way that the right wheel turns backward and the left wheel turns gently forward. The flow chart in figure 2 visualizes the bang-bang controller’s processData method.

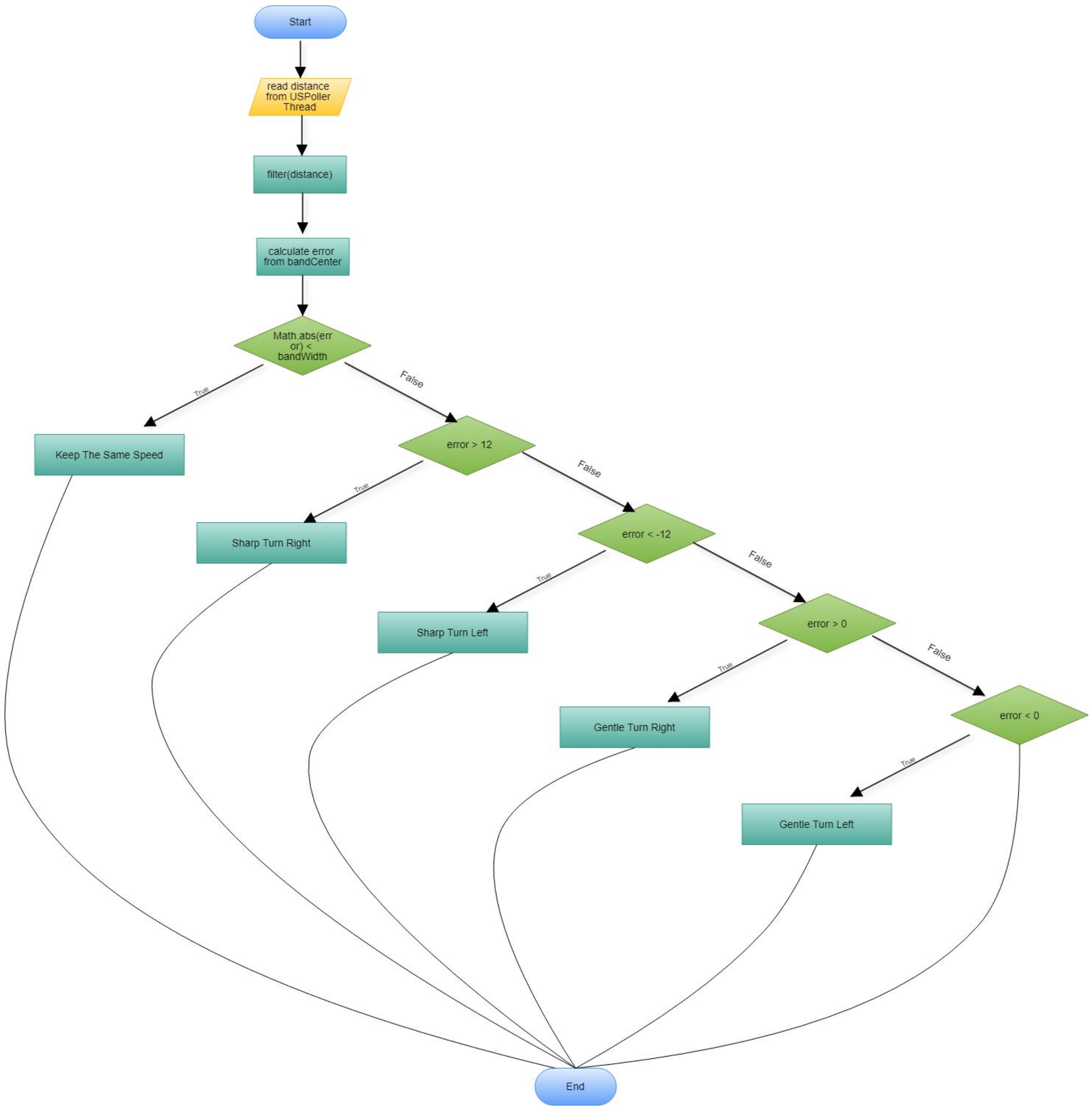


Figure 2: Flow chart illustrating Bang-Bang Controller’s output

P-type Controller: In this type of controller, delta speed exerted on the wheel is proportional to the error corresponding to the difference between the bandcenter and the current distance. The processData method commands that the right motor speed increases by delta speed and left motor speed decreases by delta speed when the error is smaller than zero and its absolute value is bigger than the bandwidth. When the error is positive and bigger than the bandwidth, the right wheel turns backward and the left wheel turns forward. One unique feature of our version of P-controller is that we have an upper bound for the delta speed correction calculated using the calcGain method. Any calculated delta speed that is greater than the threshold will be trimmed to the threshold value. These lines of code help eliminate overturning on sharp turns. It is worth mentioning that we have a lower bound for error as well, because we observed *Math.abs(int number)* fails to function for extremely small negative numbers. The flowchart in figure 3 visualizes the P-Type controller's processData method.

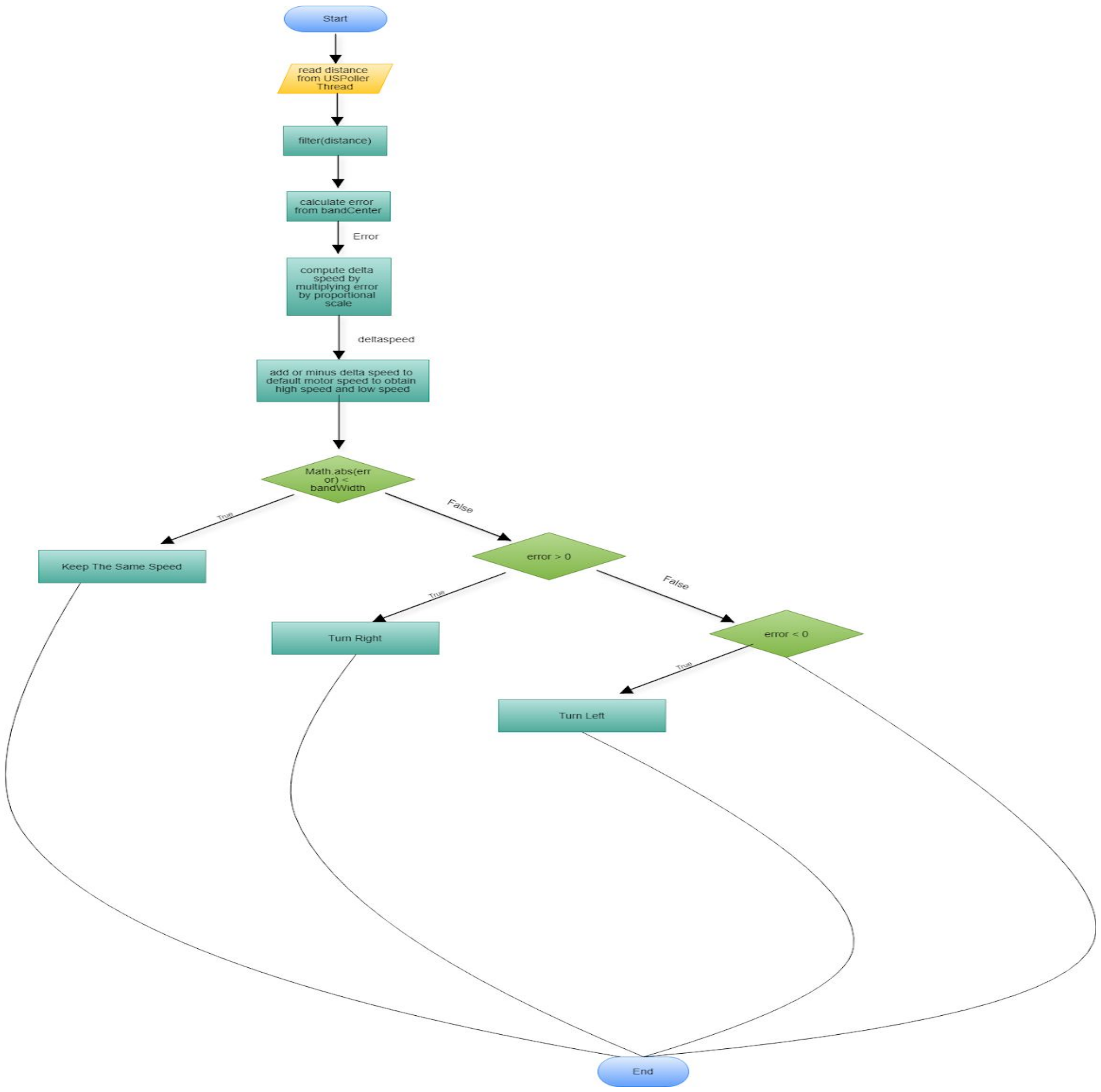
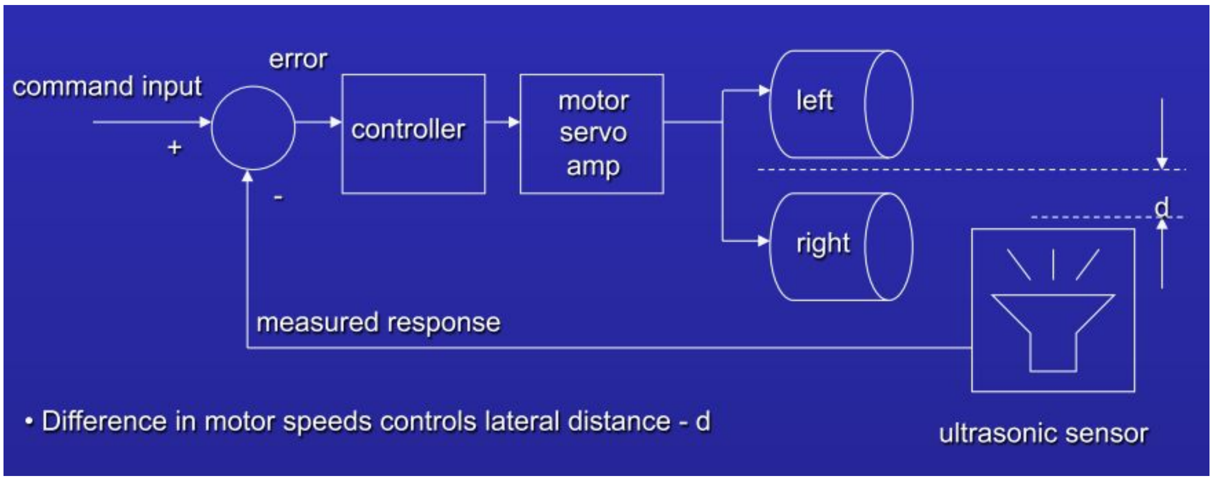

Start
read distance from USPoller Thread
filter(distance)
calculate error from bandCenter
Error
compute delta speed by multiplying error by proportional scale
deltaspeed
add or minus delta speed to default motor speed to obtain high speed and low speed
Math.abs(error) < bandWidth
True
False
Keep The Same Speed
error > 0
True
False
Turn Right
error < 0
True
Turn Left
End

Figure 3: Flow chart illustrating P-type Controller’s output



The diagram above illustrates the feedback loop. At each sampling interval, we first collect data from the US sensor. Then the measured response is processed by the selected controller. The controller calculates delta speed and sends instructions to motors. The robot moves in a different direction and ultrasonic sensor's reading changes. This is a completed loop that repeats itself until any button on LeJos brick is pressed.

Section 2: Test Data

Testing the P-type controller constant

In the case where the P-type controller's constant is set to 15 (higher than the one used in the demo i.e. 4), its performance was bad and inconsistent. The robot went through multiple long oscillation periods as every time it is getting around convex and concave corners. The robot is moving in a counterclockwise manner with respect to the obstacles. Every time it turns left to get around a concave corner, it overturns. Consequently, it gets too close to the wall after each concave corner. Afterward, it tries to readjust its position by executing a sharp right turn, but this operation moves it too far from the wall. Consequently, its path is remarkably unstable and characterized by various unexpected oscillations especially during sharp turns. To conclude, the robot moves between the two extremities from being either too close or too far from the wall while it is turning and is not able to maintain a constant distance from the wall. In fact, every small flick in sensor data destabilizes the robot and causes it to oscillate.

In the case where the P-type controller's constant is set to 2 (smaller than the one used in the demo i.e. 4), its performance was decent, but there is still room for improvement. The robot was able to maintain a consistent bandcenter from the wall once it reaches the set offset distance. In addition, it is capable of stabilizing itself after various oscillations. It experienced short oscillation periods after sharp turns but was remarkably stable overall. However, when turning convex and concave corners, it turns are occasionally not pronounced enough to imitate the shape of the path (i.e. it goes forward almost like a straight line for short corners). Moreover, it is lacking responsiveness as it sometimes turns too late. When the robot is too close from the wall, it moves away by turning 180 degrees on itself instead of driving slowly away from the obstacles. Nonetheless, it is capable of getting back to the set offset distance and continue straight without any oscillation.

Bang-Bang controller test

Test 1:

At the beginning of the lap, the bandcenter was stable until the robot encounters a convex corner where it had to turn right. It turns right sharply and started experiencing oscillations. Afterward, it turns left at a concave corner. The left turn was executed too roughly and the robot was getting to close from the wall. Consequently, a sharp right turn was triggered and the robot turns almost 180 degrees on itself before turning left once again to stay on the path. In addition, when the robot is too close from the wall, it experienced violent vibrations by turning away and towards the wall constantly. The robot is very inconsistent when it is too close to the obstacles and it seemed to have trouble getting back to the set offset distance where the machine is relatively stable. Finally, the robot successfully completed a lap without touching the wall.

Test 2:

For the second trial, the band center was consistent for most of the lap. Its behaviour is very stable and is able to make small adjustments to preserve its distance from the wall. However, the robot behaves strangely at concave corners. On the first occurrence, the robot started by moving away from the wall by sharply turning right before turning left on the concave corner. On the second occurrence, the robot turns left too early and detects the wall at a very close distance. Hence, a sharp right turn is triggered and the robots turns almost 180 degrees to move away before turning left again. Consequently, almost each time it is executing a left “U” turn, it experiences intense oscillations. Finally, the robot successfully completed a lap without touching the wall.

Test 3:

On the third trial, the robot still makes multiple adjustments while getting around concave corners. Therefore, there is a heavy oscillation period during left sharp turns. However, it is very stable when going straight. When it gets too close from the wall, the robot makes a 180 degrees right turn to move away from it. In addition, on convex corners, it tends to make a 180 degrees right turn to move away from the wall to stabilize itself at the bandcenter. Finally, the robot successfully completed a lap without touching the wall.

P-type controller test

Test 1:

The robot is very stable during most of the itinerary, it makes small adjustments in order to maintain the band center distance (32 cm). The robot alternates well between going forward and turning. However, its right turn is still not perfect. Instead of making a 90 degrees right turn and stay parallel to the wall, the robot makes a 135 degrees right turn on itself every time it detects a convex corner. For concave corners, it makes small to medium left turns while preserving the set offset distance from the wall. The robot gets around concave corners flawlessly. The robot imitates the shape of the path almost perfectly and maintains its equilibrium state with small adjustments. Finally, the robot successfully completed a lap without touching the wall.

Test 2:

The robot moves parallel to the wall and maintains a bandcenter of 32 cm most of the time. However, its performance on left turns is mediocre for this trial. Each left turn is accompanied by a sharp right turn that moves the robot away from the wall, because it was getting too close of the obstacle. Then, the left turn resumes since the robot is now detecting an infinitely big distance after the sharp right turn. Although this is an issue, the robot is capable of completing any circuit without any problems. Finally, it maintains a constant bandcenter from the wall when going straight.

Test 3:

The behaviour of the robot is similar to the previous trials. It imitates well the shape of the path and maintains a constant bandcenter to the wall by making adjustments when necessary. Every correction is proportional to the magnitude of the error since the robot is capable of identifying to what extent it needs to turn in order to preserve the set offset distance from the wall. Important oscillations occur on convex corners, but the robot is able to get back to the equilibrium state quickly. Finally, the robot successfully completed a lap without touching the wall.

Section 3: Test Analysis

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

When the P-type constant is bigger than the one used in the demo, the robot is going through oscillation periods on concave and convex corners, because of excessive adjustments. Oscillations occur during each sharp turn, because the robot turns either too close or too far from the wall, therefore it needs to readjust itself. On concave corners, its left turn is too sharp. Consequently, the robot is getting to close from the wall. Hence, an excessive sharp right turn is triggered to move away from the wall. On convex corners, the robot makes a 180 degrees turn to move away from the wall which is unnecessary. Afterwards, it makes a sharp left turn to get back to the set offset distance where the robot is relatively stable. Therefore, its path is incredibly unstable around concave and convex corners, but rather stable when going straight. In fact, it switches between the two extremities (too close or too far from the wall) while making sharp turns.

When the P-type constant is smaller than the one used in the demo, its performance is stable but not optimal. The robot was able to maintain its bandcenter when going straight and return to the equilibrium state on various minimal oscillations. It experienced short oscillation periods on sharp turns but was not as intense compared to when the constant is very big. Nonetheless, it does not turn sharply enough to imitate the shape of the path when getting around convex and concave corners (i.e. it goes forward almost as a straight line for short corners). In addition, it is lacking responsiveness as it occasionally turned a bit too late. When the robot is too close from the wall, it moves away from the wall by turning 180 degrees on itself instead of driving slowly away from the wall. However, after distancing itself from the wall, it maintains a consistent band center from the obstacles without any oscillation.

How much does your robot oscillate around the band center?

The band center of the robot is 32 cm and its band width, 6 cm. This allows the robot to not make any corrections when the measured distance is within [26, 38] cm. This margin of 12 cm is relatively big and allows the robot to stay stable when it is around the bandcenter. Therefore, the band width is big enough to eliminate any unnecessary oscillations and small enough to trigger adjustments when the error is too bigger. Hence, the robot is relatively stable around the band center.

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

As the robot approaches a convex or concave corner, the absolute value of the error (i.e. the difference between the bandcenter and the measured distance) exceeds the bandwidth. For convex corners, the error exceeds the bandwidth by 5 to 25 cm (i.e. the bandwidth is 6 cm and the calculated error varies between 11 cm to 26 cm). The closer the robot is from the wall, the greater this error will be, therefore the more it would exceed the bandwidth. For concave corners, the error is negative, hence the absolute error is considered in this case. When the robot is executing a sharp left turn, the error exceeds the bandwidth by a value in the range of [1,∞[, because at some point the ultrasonic sensor is detecting nothing so it outputs an infinitely big distance. In this case, the absolute value of the error is infinitely big which exceeds the bandwidth. Finally, when the robot is going in a straight line, it occasionally exceeds the bandwidth by a value from [1,5] which triggers a small adjustment from both types of controller.

Describe how this occurs qualitatively for each controller.

For Bang-Bang controller, the robot follows the wall until it encounters a concave corner. Then, it continues on a straight line until the sensor detects nothing (i.e. an infinitely big distance) which triggers a sharp left turn since the calculated error is infinitely small (i.e. |32 cm - 21478999 cm| > 6 cm, the absolute value of the error is bigger than bandwidth). However, the sharp left turn is often excessive and the robot will detect that it is too close from the wall (i.e. error = 32 cm - 10 cm = 22 cm) before the left turn is completed. In other words, while getting around the concave corner, it receives an error that is greater than the bandwidth, hence the robot makes a 180 degrees right turn to move away from the wall. When it is encountering a convex corner, it detects a very small distance (i.e. bandcenter - distance detected = 32 cm - 17 cm = 15 cm), hence the error is greater than the bandwidth. Consequently, it will trigger a sharp right turn. In addition, when the robot is going in a straight line, it occasionally detects false negatives which triggers a left turn, since the sensor detects an infinitely big distance. Consequently, the robot moves closer to the wall until it detects that it is too close and makes a right turn to move away. These false negatives cause oscillations in the Bang-Bang controller.

Similarly to the Bang-Bang controller, the robot running on the P-type controller exceeds the bandwidth when it encounters concave and convex corners. For concave corners, the robot continues on a straight line until the sensor detects nothing (i.e. an infinitely big distance) which triggers a left turn since the calculated error is infinitely small (i.e. |32 cm - 21478999 cm| > 6 cm, the absolute value of the error is bigger than bandwidth). However, the robot varies the amplitude of its rotation to the extent of the corner. In other words, it makes multiples small to medium left turns instead of making a sharp left turn when it gets around concave corners. Often, it starts by making a small left turn followed by a big one, then it adjusts itself to stay close to the bandcenter. For convex corners, the robot detects a very small distance from the wall, hence a big error which exceeds the bandwidth. Consequently, the P-type controller triggers a 135 degrees right turn away from the wall. Afterward, the robot adjusts itself to maintain a distance equivalent to the bandcenter from the wall.

4. Observation and conclusion

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

The P-type is a better controller compared to the Bang-bang controller. Although both of them have stable performance when the robot moves along a straight line, the P-type controller allows the robot to drive around convex or concave corners more smoothly since the robot performs a sequence of minor to intermediate turns as explained in the test analysis section. In addition, the P-type controller outputs corrections that are proportional to the magnitude of the error. On the other side, Bang-bang type can only performs sharp turns and this behavioral pattern is especially problematic when doing a U-turn around a concave corner.

Does the ultrasonic sensor produce false positives or false negatives? How frequent are they? Are they filtered?

Frequent false negatives occur. Most of them are filtered since we have increased the FILTER_CONTROL constant. 2147589989489584

False positives are less frequent but are not filtered. They occur when the robot drives too close to the wall.

Section 5: Further Improvements

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

1. Improving the run function in the UltrasonicPoller class: Adding a bound of [0,255] to the distance variable. Therefore, distance can only be a value within [0,255]. In addition, we want to store previously fetched distance and setting the average of those values to the distance variable. This way, the distance would be more representative of its current position from the wall. Hence, the impact of outlier values is decreased. Moreover, the sampling rate could be reduced in order to decrease the number of false negatives or false positives detected. Sampling fewer times can give the “Brick” more resources for correcting the position of the robot.
2. Implement a Timer class: A timer class could be implemented with a “timeOut” function that runs the corrections. Hence, the sensor can be started before the wheels of the motors start to turn. Consequently, the robot won’t run into a wall if we are placing it in front of a wall. The timer class would avoid starting the wheels before the sensor which can be dangerous. In addition, the Timer

class would have a method that handles the timer exceptions. A new thread needs to be generated to run an instance of the Timer class

3. Implement a PID controller: Instead of using a Bang-Bang Controller or P-Type Controller, a PID controller could be implemented to bring more stability to the control system. A PID controller increases the stability of the control system due to its Derivative Controller and increases the control system's response by decreasing the steady-state error thanks to the Integral Controller. In fact, the PID Controller gives a better performance overall which would allow the robot to be more stable, undergo fewer oscillations and have a faster response time. As noted during the lab, when the robot is too close to the wall, the US sensor is not working properly. Therefore, PID allows to preserve its bandcenter distance by inducing more stability. In addition, the time it would require to get back to the equilibrium state would be faster as the PID controller improves the control system's response time.

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

1. Mount the sensor at a 45° angle: Fix the sensor lower on the robot with an angle of 45 degrees with respect to the robot's wheelbase. This way the robot will detect the signals more consistently and improve overall coverage. By putting the sensor at an angle of 45 degrees, it covers equally both the wall parallel to the robot and the one in front of it. This angle is the ideal angle for this wall following task if we only had one sensor. In addition, fixing it lower on the robot will decrease the number of false negatives detected. This modification will make the robot more responsive as it will detect any obstacles sooner.
2. Distribute weight evenly and put the wheels closer: By bringing the wheels closer, i.e. under the "Brick", the robot will have better traction and weight distribution since the entire weight are focused on the wheels. This modification avoids slippage, therefore the robot is more responsive to the controller.
3. Adding a second sensor: Add a second US sensor (i.e. one facing the wall and one facing forward). This approach resolves the limited range of the ultrasonic sensor. At every sample, we collect data from both sensors and determine which one to use by selecting the smaller one. With this approach, the robot detects obstacles earlier which makes it more responsive and less prone to failure.

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

A Proportional Integral controller, Proportional Derivative Controller and the Proportional Integral Controller could be used instead of the Bang-Bang and P-type controller.

The Proportional Derivative Controller is a combination of the Proportional Controller and the Derivative Controller. In fact, it adds the output from the Proportional Controller to the output from the Derivative Controller to form its output. The Proportional Derivative controller greatly improves the stability of the control system without affecting the steady-state error.

The Proportional Integral Controller produces an output that corresponds to a combination of outputs from the Proportional Controller and the Integral Controller. The Integral Controller is effective in increasing the control system response by decreasing the steady-state error. However, Integral Controllers decrease the stability of the system. To overcome this disadvantage, the Proportional Integral Controller is used. This type of controller has the same advantages of Integral Controllers without affecting the stability of the system. Therefore, PI Controllers are used to increase control systems' response time and decrease steady-state error.

Finally, the Proportional Integral Derivative (PID) Controller is one of the most widely used controllers. Its output consist of a combination of the outputs from the proportional, integral and derivative controllers. Therefore, this type of controller includes all the advantages of the Derivative and Integral controllers overcoming their disadvantages such as steady-state error problem for Derivative Controller and stability problem for Integral Controller. Hence, the PID Controller improves the stability and the response time of the control system.