ECSE_211 Winter 2020
Lab 1: Wall Following

Group 54
Joanna Koo (260865846)
Muhammad Mustafa Javed (260808710)
Due January 23rd, 2020

Table of Contents

Section 1: Design Evaluation

1.1 Design workflow

In the design procedure, we first built the LEGO Mindstorms EV3 robot, then implemented the software logic. We built the robot first, because we wanted to take account of the physical properties of the robot in our software design. This allowed us to take considerations of the orientation of ultrasonic sensor and the size of the robot, into writing the code. It also allowed us to test the robot after implementing each method. Furthermore, it was easier to adjust the parameters in the software rather than to alter the design of the robot to achieve the required motion.

1.2 Hardware design overview

The robot is designed to be simple and agile, enabling it to make quick turns around the corners and respond rapidly to the data from the ultrasonic sensor when a wall is detected too close to the robot. We decided to locate the two motors directly underneath the EV3 brick in order to achieve a small and compact body, as we have previously observed that a wider robot was not able to make as quick and sharp turns as the smaller one. This also increased the height of the sensor to prevent any false positive due to the echoes from the floor. The ultrasonic sensor was placed at a 45° angle at front-left corner of the EV3 brick, allowing the detection of a wall placed not only on the side but also placed in front, thus facilitating concave and convex turns. The final robot design can be found in *Figures 1, 2* and *3*.

Figure 1. Bird's eye view of the robot.

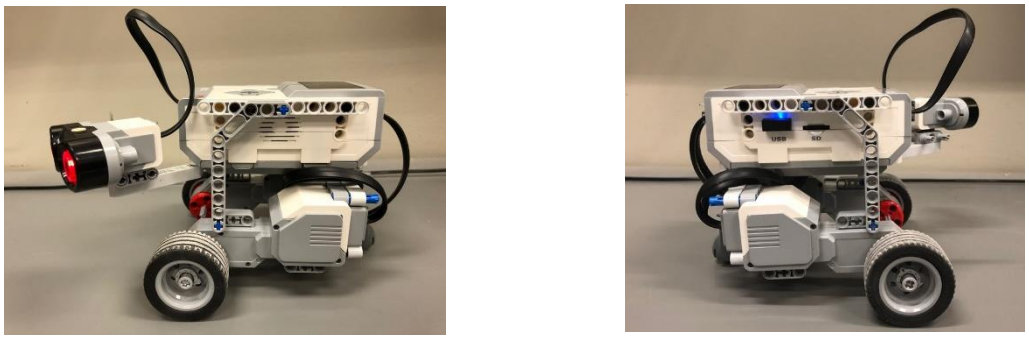

Figure 2. Left and right side profiles of the robot.

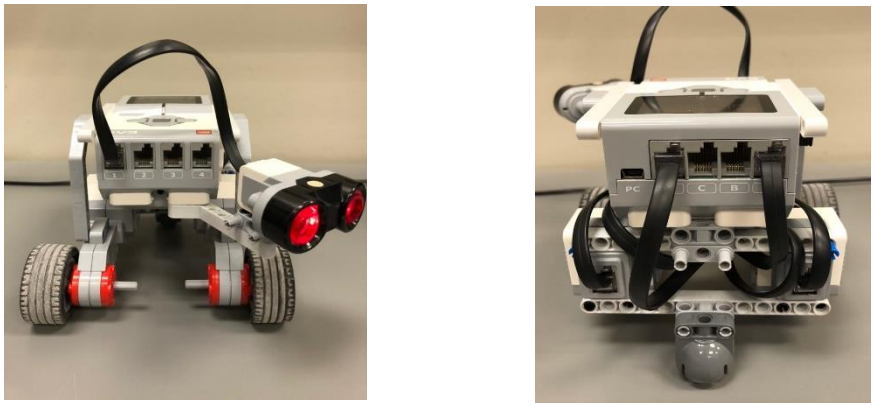

Figure 3. Front and back profiles of the robot.

1.3 Software overview

The robot was required to complete a series of laps around the blocks while maintaining a constant distance from the wall. The software was designed to get an input from the user using the EV3 buttons and initiate the motion in either Bang-Bang Controller method or P-type method.
The program was formed on four classes. The main method in the Main class is responsible for getting inputs from the user, starting a thread for LCD control and, depending on the input, creating an object of the Ultrasonic Controller class in either Bang-Bang mode or P-type mode. The Ultrasonic class extends the Thread class. This allows the object of Ultrasonic class to run concurrently with the Main class. The main method then waits for the user to press any button before ending the process and returning to the operating system. *Figure 4* shows the flowchart for main class.

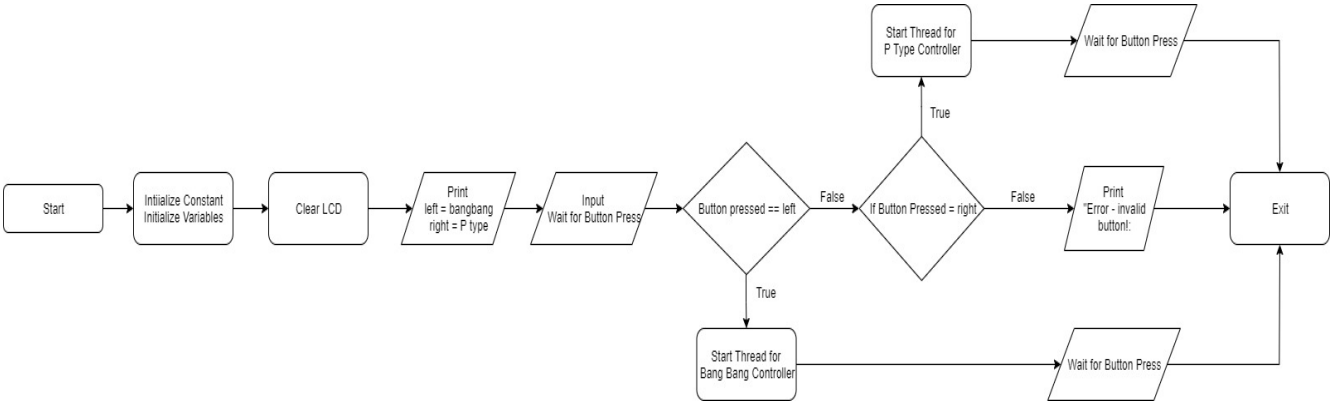

Figure 4. Flowchart for Main class.

Bang-Bang Controller

In the Bang-Bang controller, the sensor reads the distance from the wall and compares it with a pre-set value. If the two values are within a certain threshold, the robot goes straight by setting the speed of the two motors to the same value. If the measured distance is greater than the set distance, the robot is far from the wall. It adjusts by decreasing the speed of the left motor to a fixed value, which causes the robot to drift left. If the measured distance is sensed to be less than the set wall distance, the robot is too close to the wall. It adjusts by decreasing the speed of the right motor to a fixed value, which causes the robot to drift right. The process of Bang-Bang controller is illustrated by the flowchart in *Figure 5*.

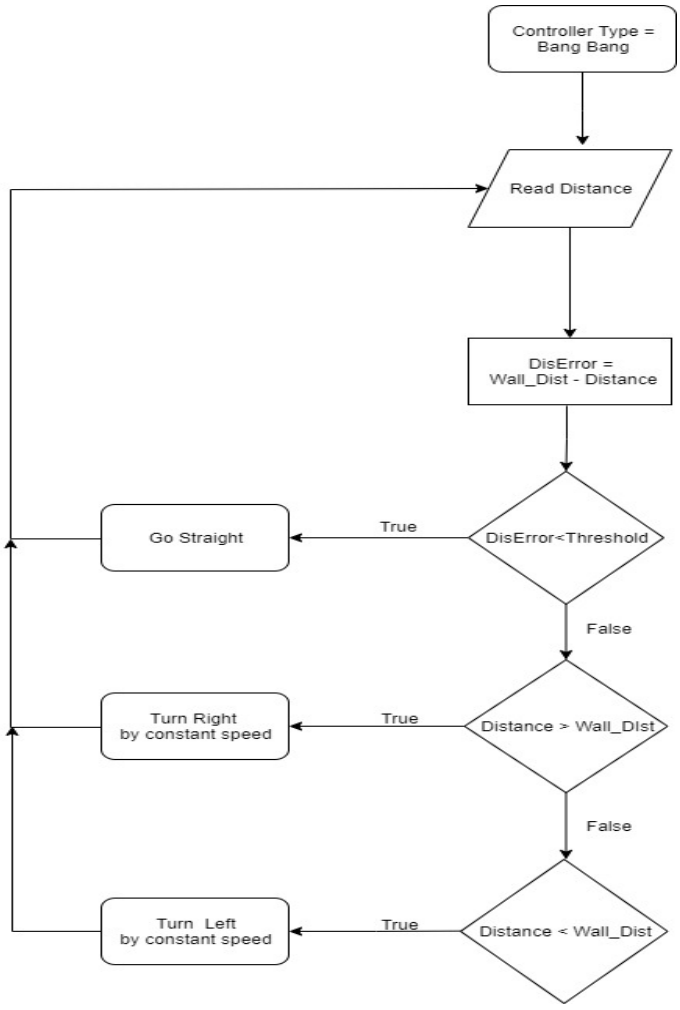

Figure 5: Flowchart for Bang-Bang Controller.

P-type Controller

In the P-type controller, the sensor data is compared with a pre-set wall distance value. If the difference between the two values is within a set bandwidth, the robot goes straight by setting the speed of the two motors to an equal value. If the measured distance is greater than wall distance, the robot turns left by increasing the speed of the right motor and decreasing the speed of the left motor. If the measured distance is less, it drifts left by increasing the speed of the left motor and decreasing the speed of the right motor. Unlike the Bang-Bang controller, the change in the speed of motors is proportional to the distance error between the measured and preset distance. The difference is limited to a maximum value to unbounded increase/decrease in speeds as shown in *Figure 6*. The flowchart in *Figure 7* illustrates this process.

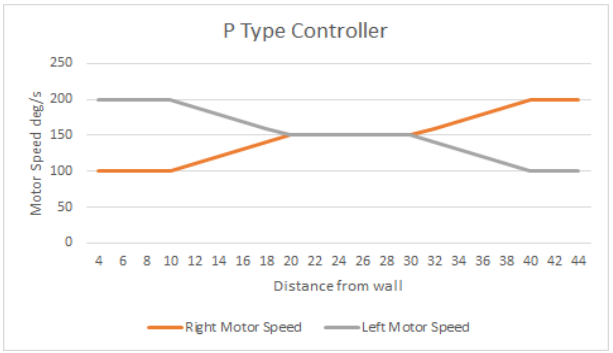

Figure 6: A graph showing the motor speeds as a function of distance measured

Figure 7: Flowchart for P-Type Controller.

Section 2: Test Data

2.1 Test P-type controller constant

The P-type controller constant used in the demo is 5. The two values that are below, 2 and 4, and above, 7 and 9, were tested and the observations are recorded in *Table 1*.

P-type controller constant	Performance
2	The robot was not shaky; however, it often failed to correct the errors with the bandwidth, as the change in motor speed was less responsive to the errors due to the small P-constant. At concave turns, the robot sometimes failed to turn away from the wall. At convex turns, the robot often did not turn sharply enough, and it made wide turns. The desired bandwidth was rarely respected throughout the duration of the test.
4	Similar to the case above, the small P-type constant resulted in failure to make successful turns. As a result, the robot collided twice at convex turns. However, the robot oscillated less than the case above. The bandwidth was respected for most of the time.
7	The robot was relatively stable at the beginning, but soon started to gradually oscillate, especially at the turns. It collided with the wall once at a convex turn. Other turns were also sharp; however, bandwidth was respected for most of the time.
9	The robot was oscillating too much for most of the time. The change of motor speed was too abrupt, and the robot often collided into the wall at both convex and concave turns. The desired bandcenter was rarely respected.

Table 1. Observations for different P-type constants.

2.2 Test Bang-Bang controller

Trial	Success/Fail	Qualitative Observations
1	Success	The robot was stable at the beginning but started to oscillate after the first turn. During most of the turns, it successfully backed up, then re-adjusted the distance when it approached too close to the wall. The robot turned too sharply at some convex corners, but overall, the bandcenter was respected.
2	Success	The robot was stable throughout the first lap. However, it started to oscillate after a sharp convex turn during the second lap. Most of the time, the robot ran too close to the wall.
3	Success	The robot was stable most of the time, during both laps. The bandcenter was sometimes too small (approximately 5 cm) after a convex turn, but soon adjusted the distance.

Table 2. Observations for Bang-Bang Controller tests.

2.3 Test P-type controller

Trial	Success/Fail	Qualitative Observations
1	Success	Even though the robot followed the bandcenter, it was showing a large oscillation pattern in order to make small corrections in distance. For the convex turns, the turn radius was too large and resulted in the robot too far from the wall after completing the turn.
2	Fail	The robot completed its initial lap. During the second lap, the robot completed a concave turn, then collided into the wall. The bandcenter was not respected and decreased over time until it finally collided. The robot was oscillating frequently.
3	Success	The robot was stable when it started running. It maintained a fairly contact distance from the wall (equal to the band center) and the oscillations were greatly reduced. The robot showed sharp turns near convex corners (possibility of collision).

Table 3. Observations for P-type Controller tests.

Section 3: Test Analysis

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

P-type Constant

For the demo, the P-type constant was set to 5. During the testing, we experimented with higher values of 7 and 9, and with lower values of 2 and 4. For the higher values, the robot made sudden motions, which caused large oscillations and jerky behavior. The bandcenter was maintained; however, it resulted in larger oscillations as the motor speeds were changing by a large value. For lower gains, the robot failed to keep a constant bandcenter and did not turn quickly enough that it eventually ended up colliding with the wall. However, the oscillation was reduced around the bandcenter.

2. How much does your robot oscillate around the band center?

P-type Controller

During the 3 independent trials, we analyzed the motion of the robot. With the gain value set at 5, the robot showed minimal oscillation around the bandcenter as it was factoring in the size of correction required. We observed some oscillations, because the measured distance values were frequently deviating from the threshold. The bandgap was increased that smoothed out the remaining oscillations.

Bang-Bang Controller

With the Bang-Bang controller, the robot oscillated less around the bandcenter. However, compared to the trials done with the P-type controller, the robot reacted slowly whenever the bandwidth was not respected, as the Bang-Bang controller changes speed of each motor only when the robot is not within the threshold. Furthermore, since the Bang-Bang controller's error correction mechanism does not make adjustments proportional to the magnitude of error, its movements were not as smooth.

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

P-type Controller

The robot remained well within the bandwidth and respected the bandcenter when following a straight wall. However, during the concave and convex turns, the robot came too close to the wall. After readjusting the maximum value for the motor speed, the robot ran successfully while respecting the bandwidth. Another filter was added to the algorithm to twist the robot, in case the wall was detected too close to the sensor.

Bang-Bang Controller

The bandwidth was respected for most of the time. However, the robot sometimes ended up overcorrecting the distance when only a slight amount of error was observed; as a result, the robot made extremely sharp turns at convex turns or moved too far away immediately after a convex turn, exceeding the bandwidth at approximately 5 to 10 centimeters.

Section 4: Observations and Conclusions

During the independent trials of both controller types, the Bang-Bang controller had more successful trials than the P-type controller. However, after tuning the parameters, we noticed that the P-type was smoother and had less probability of collision compared to the Bang-Bang controller. Analytically, we found that the P-type is better, because it factors in the size of the error to be corrected. We came to the conclusion that the preferred type of controller was the P-type controller.

The ultrasonic sensor produced both false positive and false negatives. During the convex turn, even when the ultrasonic sensor was pointed to a distance of infinity (for the sensor), it returned a false value, which failed to cause the robot to turn. A possible reason for false positives was the pulses and echoes from other ultrasonic sensors that were operating in the vicinity. At particular instances, the sensor failed to detect an object in its path, especially if the object was too close to the sensor. Two data filters were set up to refine the sensor input. The first filter removed false positive in case of gaps in the wall, by ignoring the first 20 values exceeding 255 centimeters. The second one rectified false negatives by causing the robot to reverse and instantaneously drift whenever the measured distance is less than 5 centimeters.

Section 5: Further Improvements

Software improvement

The following are possible software improvements that can be made to improve ultrasonic sensor errors.

1. A filter that stores the previous value of the distance and compares with the current value can be added. If there is a huge difference in the two values, ignore the current value and take a new reading. This will help to filter out large spikes in sensor data.
2. The reversing of the robot can be improved. By keeping track of the previous valid readings, the correct location of the robot with respect to the wall can be detected. In order to avoid false readings from the sensor, the robot could reverse in the orientation where the correct readings from the sensor was returned, increasing the probability of returning correct values.
3. The sampling rate for the sensor can be increased by decreasing the sleep time for the Ultrasonic thread. Higher sampling rate will allow distance to be measured more quickly and make path corrections accordingly. This will, however, require greater processing power from the microprocessor.

Hardware improvement

The following are possible hardware improvements that can be made to improve the controller performance.

1. The ultrasonic sensor can be mounted on a motor that oscillates and functions like a radar system. This would allow a wider field of view for the sensor and improve detection in both forward and sideways directions.
2. The use of more ultrasonic sensors would increase the robot's range of vision and make the readings more accurate. By installing a sensor facing front and side, the robot will be more capable in detecting the walls.
3. Instead of a two-wheel drive, the robot can be setup to make use of rubber tracks or a four-wheel drive system, which will help to improve the grip over the floor surface. As such the controller will be able to make sharper and more accurate turns without drifting away from the bandcenter.

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

Proportional Integral derivative (PID) controller is one of the controller types that can be used. As a PID controller would apply a correction to the calculated error distance based on proportional, integral and derivative measurements, implementing a PID controller will likely to result in more accurate movements of the robot.