ECSE 211 Lab 1

Wall Following Lab Report

Group 65

Kathy Yim 260741794

Hanwen Wang 260778557

Design Evaluation

The objective of this lab is to design an EV3 robot that is capable of navigating around walls, gaps, as well as concave and convex corners. The lab mainly consists of two parts – hardware design and software design. For the hardware design, the LEGO Mindstorms EV3 kit is used to construct the robot. For the software design, Java code is implemented then run on the EV3 robot through the leJOS software installed, which provides JRE for the robot.

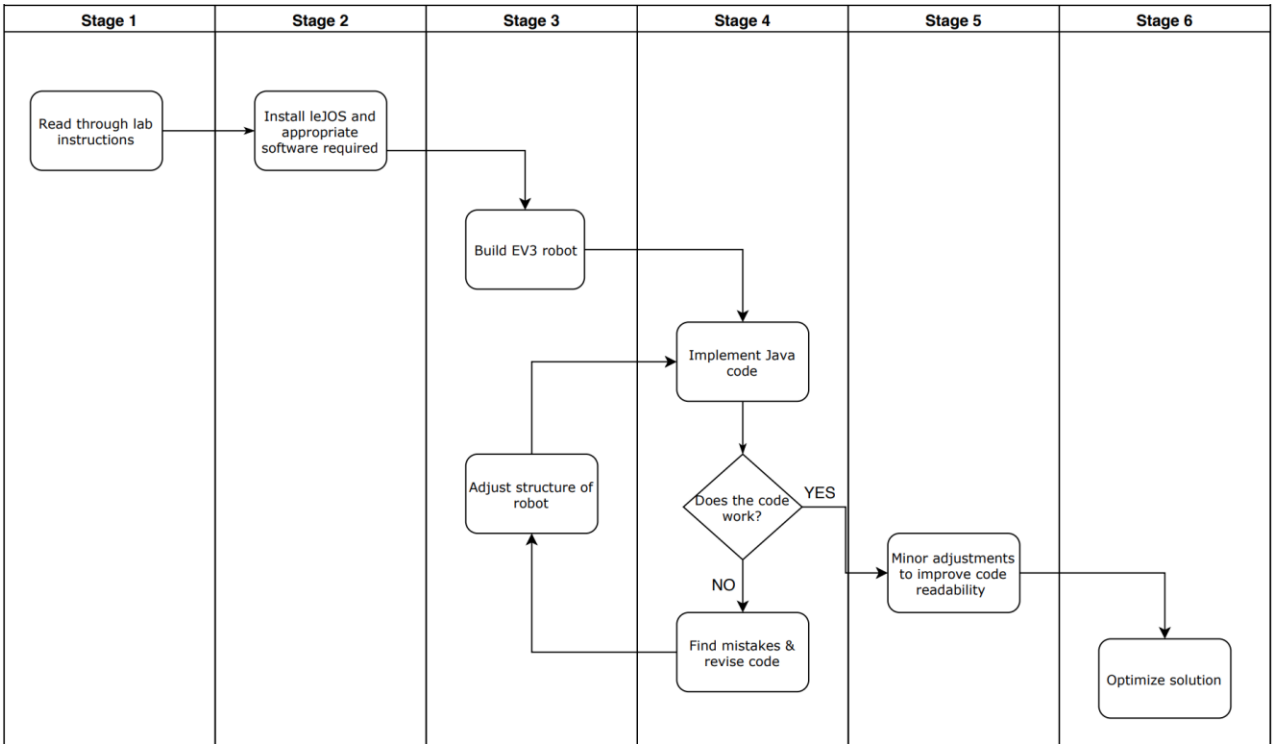

Figure 1: Workflow

Our workflow mainly consists of repeatedly testing our Java code and then revising the code to improve the performance of the EV3 robot. We first tested for movement along straight wall, followed by concave and convex corners, and lastly gaps between walls. If our robot failed any of the above tests, we would modify our code, revisit our code logic, and adjust the structure of the robot if necessary.

The physical design of our robot is shown below in figure 2. For the physical design process, we decided to place the wheels directly below the EV3 brick to aim to reduce the bandwidth. The goal of the front wheel design was to reduce the probability of the EV3 or the sensor touching the wall when the robot rotates an angle of greater or equal to 90 degrees. We placed the ultrasonic sensor horizontally at relatively low height to avoid having the sensor taller than the height of the wall – which could result in significant testing errors if the sensor does not correctly detect the distance and existence of a wall. The main issue with placing the sensor horizontally was that it slightly increased the width of the robot, therefore increasing the possibility of the EV3 touching the wall when turning around a corner.

However, the advantage to a horizontal sensor is that it can detect a wall and measure distance slightly faster compared to a vertical sensor where the two lenses are lined up vertically and have the same distance to the wall.

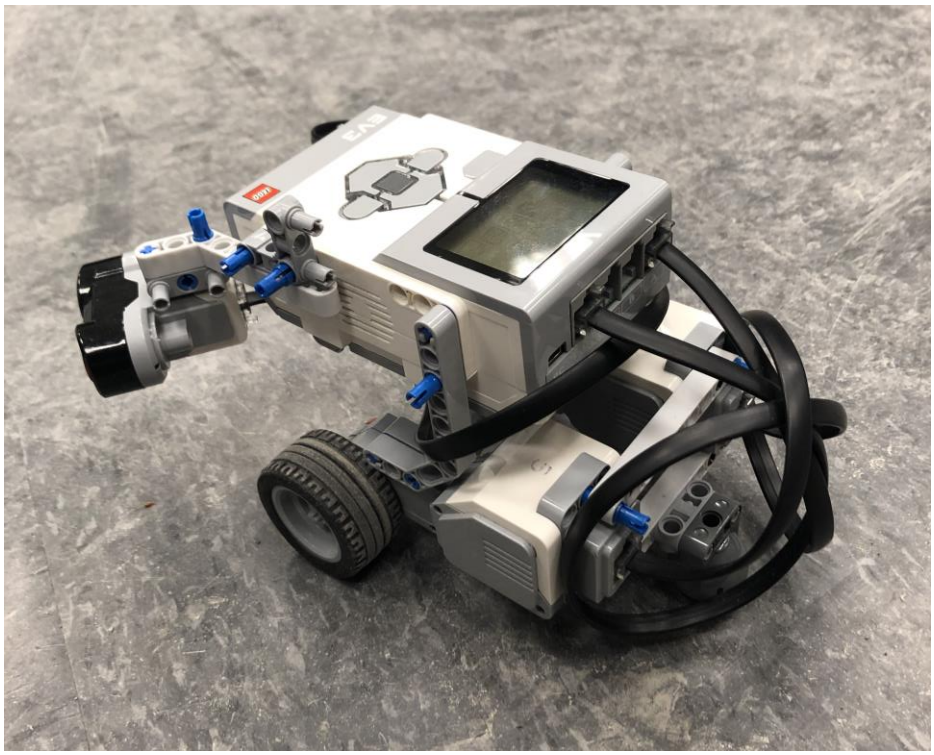

Figure 2. EV3 robot

The software design of the EV3 is divided into two main components – Bang-Bang Controller and P-Controller.

For the Bang-Bang Controller, we adjusted the speed of two motors depending on the direction of movement and degree of turning we needed the EV3 robot to accomplish. For a set range, we allowed the EV3 to continue moving forward. When a certain distance is less than the set range, the EV3 is programmed to turn right. However, when this distance is greater than the set range, the robot is programmed to turn left until it moves back into a range to continue moving forward. It took many trials and testing to find the most suitable values for all aspects of the EV3.

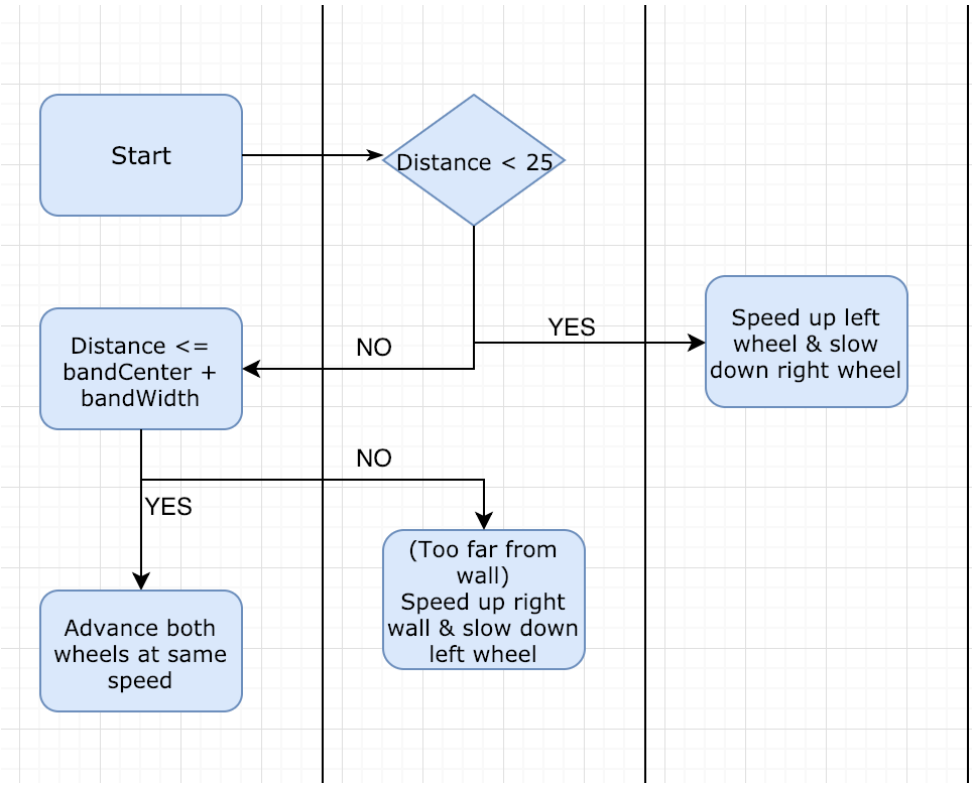

Figure 3. Bang-Bang Controller workflow

When the EV3 encounters a concave corner where it needs to make a right turn, we slowed down the right wheel and sped up the left wheel to allow for a large degree turn in a short period of time. When the EV3 encounters a convex corner where it needs to turn left, we decided to keep the right wheel at the same speed while slowing down the left wheel. This is to give the robot time to re-adjust its position but also to prevent the robot from touching the corner while making an outer-corner turn. The same logic was applied for concave left turns and convex right turns.

For the P-Controller, we used similar logic as we did for the Bang-Bang Controller except the speed was set proportional to the distance. The constants defined had to be tweaked often and tested multiple times for both controllers. Since the speed is proportional to some distance, we had to slow down one wheel and speed up the other simultaneously so the robot can make smoother turns

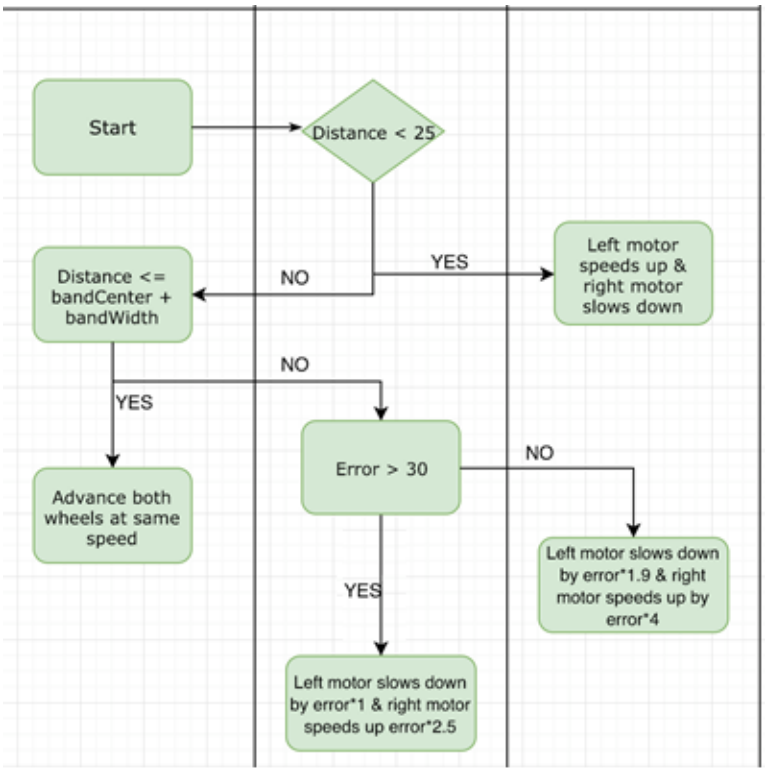

Figure 4. P-controller workflow

As seen in figure 4, our p-controller workflow is very similar to that of the Bang-Bang controller, with the difference being that the motors are slowed by the 'error' multiplied by some constant due to the proportionality factor.

Test Data

Trial	1	2	3
Band Center	35	35	35
Observations	- Easily turns around concave corners without crashing - Keeps a safe distance when turning convex corners - Frequent oscillations for enhanced wall detection - Occasionally does not detect gaps and crashes	- Occasionally fails concave corner turns and crashes - Usually stays within band center after oscillations	- Sometimes makes wide turns on a narrow path - Would miss the corner and make a circular path - Small oscillations but tends to remain a safe distance from the wall

Table 1: Bang-Bang Controller Tests

Trial	1	2	3
Band Center	35	35	35
Observations	-Smooth turns around concave corners - Some oscillations but still stays on path	- Accurate at detecting gaps & does not turn into them - Still some oscillations but no crashes	-Issues with convex turns and U-shape paths

Table 2: P-Controller Tests

Trial	1	2
Constant for both conditions	0.5, 0.05	3, 1
Observations	- Easily crashes on concave corners - Frequent oscillations due to slower convergence to band center	- Speed of both wheels are too high for robot to follow path without crashing - Abrupt responses and movement to any wall detection

Table 3: P-Controller Constant Tests

Test Analysis

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

Based on the data collected through different trials, a different p-type constant from the one used in the demo can cause the trajectory of the robot's movements to change drastically. Referring back to table 3, it can be concluded that a smaller p-type constant decreases the speed of the robot which often times causes it to crash as it does not move quickly enough to avoid a wall. Whereas larger constant causes the robot to oscillate much more frequently and even over-turn when it detects a corner.

How much does your robot oscillate around the band center?

Our robot oscillates around the band center a fair amount of times to a maximum of 8cm. It does not oscillate as much as some other robots do because oscillations occur more frequently with an increase in speed and our robot does not move at a very high speed. It tends to stay on the right side of the band center to be safe.

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

Yes, since our bandwidth is 2cm the EV3 often moves in and out of the bandwidth and around the band center at no more than 8cm on each side.

Describe how this occurs qualitatively for each controller

In the Bang-Bang controller, the robot's sudden changes in direction allow it to move back into the bandwidth faster. For the p-controller however, the robot tends to gradually steer back into bandwidth range due to a proportional response. Generally, the robot would move outside of the bandwidth because it does not know its path beforehand and is therefore forced to react suddenly when the sensor detects a wall.

Observations and Conclusions

Based on our analysis of the two systems, the p-controller is considered to be optimal for the EV3 as it causes fewer oscillations compared to the Bang-Bang controller. Although the Bang-Bang controller tends to attain its desired output quicker, the robot will often overshoot the angle it is supposed to be at and then make multiple adjustments to its orientation before continuing to move forward. However, the increased control of the motor position in the p-controller comes with the cost of the system being over-damped.

Theoretically, the p-controller is more desirable because its speed is proportional to distance although, during testing, we found the oscillations of both the p-controller and Bang-Bang controller to be insignificant during runs. However, the distinguishing factor occurs when the robot makes convex left turns operating on Bang-Bang controller - where oscillations are much more drastic compared to left turns on p-controller. This is because of the fact that the robot is reading significantly different values and needs times to adjust its position.

When the ultrasonic sensor comes too close to a wall, it has difficult recognizing the obstacle and this produces a false negative. This did not happen very often with our robot since we had a band center of 27 so it was not moving very close to the wall. The p-controller filters out up to 18 values that are larger than 255. In the case that the sensor receives a false positive signal that it thinks it received or if the sensor sends out a signal but does not receive it, the robot remains moving in a straight path. However if after 17 values the robot still receives values greater than 255, then that value is taken into consideration.

Further Improvements

HARDWARE IMPROVEMENTS

- Addition of a rotating sensor that revolves and reads in values and once it reaches a set degree, it outputs a value to inform us how far from the wall the robot is
- Addition of a touch sensor to prevent collision by reorienting the robot in the right direction after detecting obstacles
- Increasing the number of wheels to two on each side for the robot to have a better grip of the ground and make better turns

SOFTWARE IMPROVEMENTS

- Implementing an algorithm to compare filtered out values and verify that the values are indeed real values so the robot can react faster after detecting obstacles
- Implementation of an algorithm that can recognize unwanted values when its oscillating too much; skips methods to turn left/right and moves in a straight path
- Implementation of an algorithm that combines the p-controller and Bang-Bang controller; a state-driven p-controller that has proportionality functions but also a fully backwards state to avoid crashing into walls

OTHER CONTROLLERS

Proportional-Integral Control

Proportional-integral (PI) controllers are slightly more complex than P-controllers because they have two tuning parameters to adjust. However, the integral action lets PI controllers eliminate offset, which is a major weakness of the P-controller. In PI controllers, the value of the controller output is fed into the system as the manipulated variable input.

The PI algorithm is described as the following equation:

$$CO = CO_{bias} + Kc \cdot e(t) + \frac{Kc}{Ti} \int e(t)dt$$

Where Kc and *T*i are both tuning parameters, CO is the controller output signal, and e(t) is the current controller error.

Adaptive Control

The adaptive control system contains a controller that can detect changes in the characteristic of the process and gather information. Over time, it adjusts the controller parameters and utilizes this information to optimize its performance. It senses and adapts to the feedback loop it is inside, and develops into a better controller over time. Compared to the p-controller or the Bang-Bang controller where the parameters are constant, the adaptive controller does not rely on design conducted prior to the closed loop operation.

Sources: https://controlguru.com/integral-action-and-pi-control/

https://www.britannica.com/technology/adaptive-control