**Section 1-Design evaluation:**

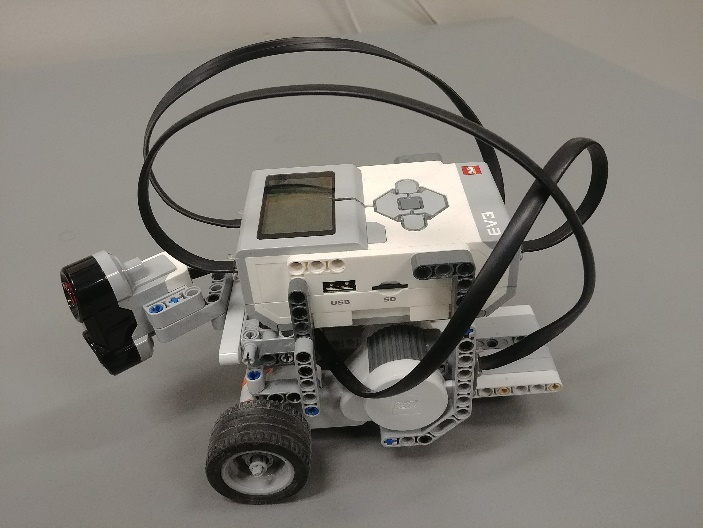
We figured that best approach to this lab was to keep things as simple as possible, so we used a basic design to build our wall fallower. We only used essential components in our design: an ultrasonic sensor, two motors, an EV3 brick and a couple of Lego pieces. We started by building a three-wheel robot. The two front wheels were connected to two separate motors and oversaw the propelling/pushing/pulling of the robot. We also had a free wheel at the back for stability. The EV3 brick, or in other words the brain of the robot, simply rested on top of the robot/chassisThis sensor was at an angle of 45 degrees and was used to measure the distance between the robot and the wall. The motors and the sensor were connected to the EV3 via RJ12 connector cables.



Figure : Design on which our robot is based on

Figure 1: Left side view of our robot

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]](#footnote-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. 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/programed two different type of controller for this lab, a BangBang type and a Proportional type (P-type). 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. 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. Tunning constants

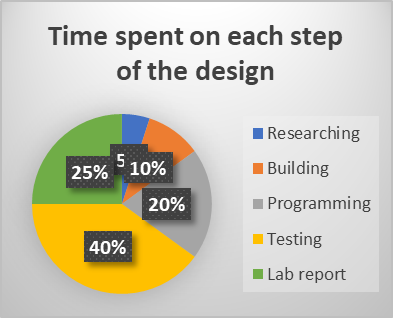


Chart 1: Pie chart of time distribution

**Section 2-Test Data:**

Testing the P-type controller constant: TITLE for the tables??

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Proportional constant** | **Successful laps for a L-shape wall** | **Behavior** | **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 bandwidth. It was very similar to what we observed with our tuned constant.

Bang-Bang controller test:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 |  |  |  |  |  |  |  | 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 jigger/jitter/shake 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/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 path/direction/track. A couple of test and tuning later, we finally had perfect run.

**Section 3-Test Analysis:**

**Section 4-Observation and conclusions:**

**Section 5- Further Improvements:**

Software improvements:



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 addressed by fixing 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.
3. Cd

Other controller type that could be used:

1. Nxtprograms.com, 3-Motor chassis, <http://nxtprograms.com/3-motor_chassis/steps.html>, January 18, 2018 [↑](#footnote-ref-1)