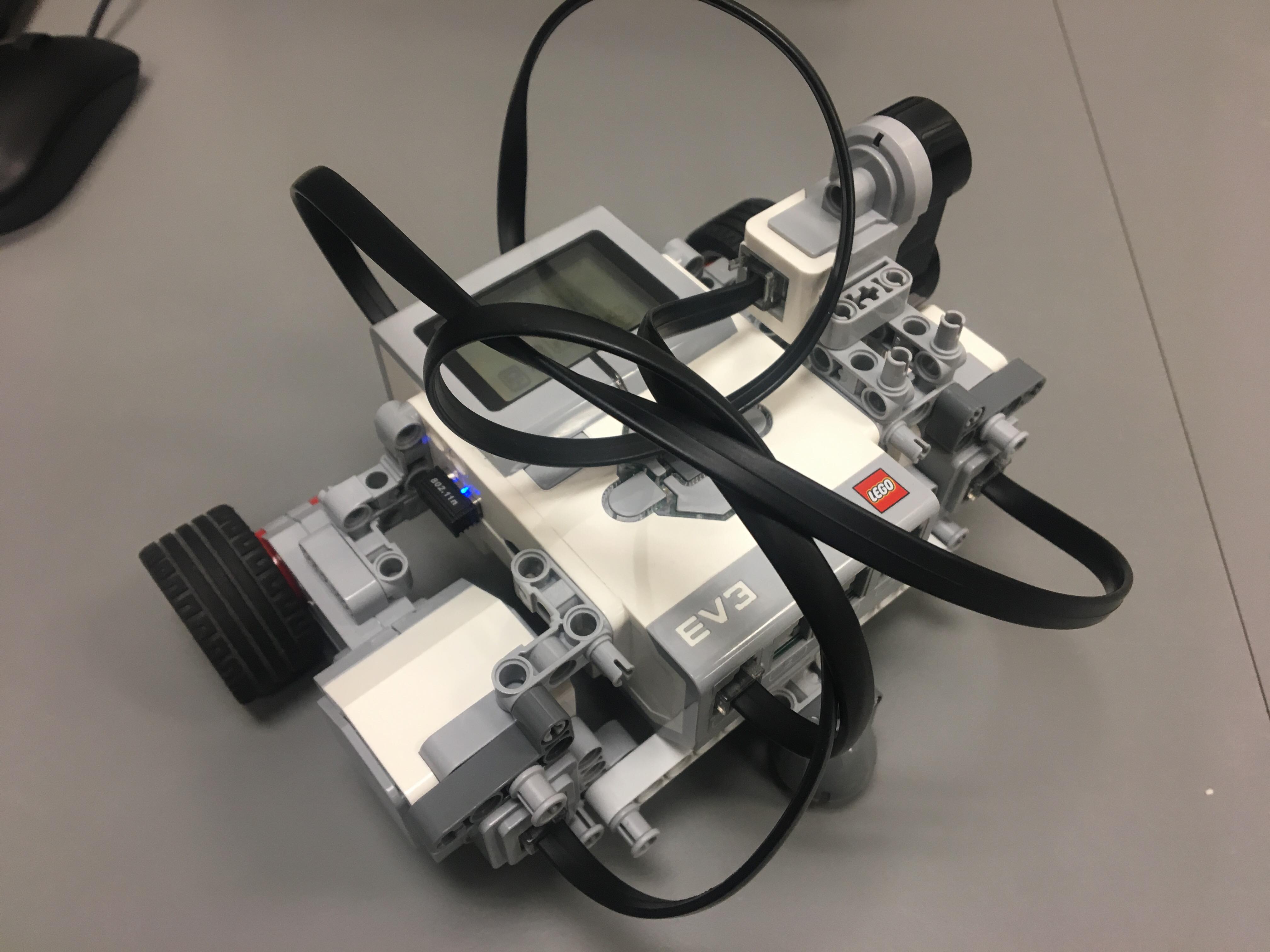
## Overview of Lab #1: Wall Following

*ECSE 211: Design Principles and Methods*

# Macintosh HD:Users:lauriezaccarin:Downloads:50133373_236821503902965_6501941845981921280_n.jpgSection 1: Design Evaluation

Figure #2: Improved design

Figure #1: Initial design

The initial design of our robot can be seen on the left. The two wheels attached to the motor and placed outwards on either side of the EV3. This wide-set wheel placement hindered the robot’s capacity of making sharper and faster turns, so the configuration was later reimagined with the wheels places instead under the EV3. As expected, this facilitated the robot to make sharper turns. Another design element that has improved upon was the placement of the ultrasonic sensor. Initially, it was placed at the front of the robot to the left. Since the robot was conceived to make left turns for the purposes of this lab, the sensor was always nearest to the obstacles and therefore was more likely to make contact with the wall. It was therefore moved to the back of the robot, at a 45**°** angle.

# Figure #3: Bang-Bang controller logic

# Figure #4: P-controller logic

# Section 2: Test Data

Testing the P-type controller constant

The P-type controller constant used for our demo was of 4. Although this was a smaller value than the value that our TA suggested (5-6), it made more sense for our robot to use this value of 4 since we had a smaller axle track compared to many others.

Trial n**°**1 (Constant increased to 6): The robot turns too quickly as the motor speed increases dramatically. This causes the robot to get to close to the obstacles. The robot therefore overcorrects and turns approximately 180**°** away from the wall. The band center is too small from the adjustment, and there is more oscillation due to the error from the overcorrection.

Trial n**°**2 (Constant decreased to 2): The robot turns too slowly, and as a result, when it approaches the wall it turns away to slowly to avoid contact.  Although the contact was avoiding by having the car move backwards instead of making the turn, it was more likely to collide with the wall as it was moving backwards. The turning is not fast enough in proportion to cruising speed.

Bang-Bang controller test (3 independent trials)

Trial n**°**1: The robot correctly follows the wall in a relatively straight manner, but when it reaches the end of the wall such that the robot is required to go around to reach the exterior side, the turning radius is too large. As a result, although the turn is smooth without oscillation, the robot gets too distant from the wall and thus must adjust its position by going to the left. This correction to the left causes the robot to approach the next concave corner at the wrong angle (it faces the wall instead of being parallel to it). The result indicated that the speed of right wheel should be increased so that the car makes a sharper turn. This change is made before proceeding to trial n**°**2.

Trial n**°**2: The robot successfully completed the first concave corner. Soon after, it registered a gap in the wall as a convex corner and turned into the wall that began after the gap. This indicated the need to improve the filtering so that the robot can ignore the narrow slit. Before trial 3, we also increased the band-center to decrease the risk of collision when robot begins a large left turn when it is not supposed to.

Trial n**°**3: The robot followed the wall correctly, although there was visible amount of oscillation, primarily due to correction of the robot’s trajectory towards the left. We also noted that our robot tends to move towards right, since the floor was too dusty, so presumably each of the tires experienced different levels of friction. The dustiness also causes slipping of the wheels from the lack of traction. None the less, it successfully turned two corners before turning too sharply at the end of the first wall.

P-type controller test (3 independent trials)

Trial n**°**1: The robot started the lap well, moving in a straight line. However, at the first convex turn it turned to soon, and therefore overcorrected dramatically.

Trial n°2: The robot successfully handled a gap within the wall and completed the first convex turn. However, it approached the concave turn with the wrong angle and did not sufficiently correct its trajectory to clear the turn.

Trial n**°**3: The robot handled straight lines, gaps, convex and concave turns, but the oscillation of the robot was noticeable as the speed was quite slow. Its progress was not as smooth as it can be with the p-control.

# Section 3: Test Analysis

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

When we change the p-type constant from the most effective and successful value for our code, the robot does not perform as well. When the value is too high, the robot tends to dramatically overcorrect small errors and consequently collides into the wall at very high speed or moves away from the wall at very high speed. In other words, there was a tendency for the robot to go off track and lose its correct direction significantly enough to be unable to correct back by itself.

On the other hand, when the constant is set at a too low value, the robot is too slow to turn in relation to its cruising speed. The correction is too slow for the speed at which the robot approaches the obstacles, so it fails to avoid collision.

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

At the cruising speed used for our demo, the robot does not oscillate too heavily around the band center. The oscillation is noticeable but is not so large that it hinders the goal of the robot to complete the course. The robot oscillates at most by about 20 degrees, and it was frequent to observe it after making a turn at the corner.

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

Yes, the robot would at first sometimes exceed the bandwidth. The error in the distance from robot to wall was at most 10 cm, but usually stayed at within 5 cm.

*Describe how this occurs qualitatively for each controller.*

Since we had a lower cruising speed for both controllers, we could not observe much significant difference between their behaviour. That being said, the oscillation behaviour was more frequently observed using the p-type controller, although it had a smaller tendency to exceed the bandwidth. For the bang-bang controller, the oscillation behaviour was similar to that of the p-type, but it had a slightly higher tendency to exceed the bandwidth, especially when the floor was more slippery. Indeed, the normal turning speed would sometimes fall short in this occasion.

# Section 4: Observations and Conclusions

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

The Bang-Bang controller was more effective for this project as the sensor provided is too poor for the precise adjustments of p-control. Indeed, the false measurements read by the poor sensor would cause the wheels to over or under correct the trajectory of the robot when using the p-control. On the other hand, for the bang-bang control, so long as the speed is within a reasonable range, a wrong measurement will not cause such extreme speeds, and the error is more easily correctible. This is because the speed is not proportional to the distance; a big error will not cause a large over correction in the bang-bang control. Often during the performances, the filter would let large false values through as they were just under the error threshold. These values would very negatively interfere with the p-control. However, it must be mentioned that it was notable that both controllers worked effectively well at the slower cruising speed.

*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?*

One of the biggest problems that was encountered during the conception of the code was the inconsistency of the ultrasonic sensor. Once the distance between the sensor and the object located was less than 30 centimeters, the distance returned by the sensor would start to fluctuate drastically between very large and small values, such as 4 to 70 centimeters. This means that both false positives and false negatives were read by the sensor. Since these random values could not be predicted or explained, the best solution was to impose a set value of 30 centimeters when the distance between the robot and the obstacle dropped below 30 centimeters. This solution effectively filtered out all false positive and negative readings and permitted the robot to follow its course correctly.

# Section 5: Further Improvements

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

A possible improvement would be to add a filter that catches very low error values as well as the pre-existing high values errors. This is difficult as larger stray values are easy to identify and filter out, but all the distance measurements taken are small by nature, as the robot is always relatively close to the obstacles, rendering the distinction of false small values difficult. Another improvement would consist of fine-tuning the speed of the motor by controlling them more precisely. In the bang-bang control the robot often has to back up in order to avoid obstacles. Finer tuning would remove this extra step, as the robot would never get so close to the wall. Finally, another improvement would be related to the turning of convex corners. When the robot approaches a convex corner, it registers suddenly very large distances indicating an approaching corner, due to the 45° angle of the sensor. The robot would then start to turn prematurely, requiring us to install a thread for it to sleep an arbitrary amount of time before making the turn in order to successfully avoid the wall. However, by calculating the theoretical optimal speed of the robot, the amount of time could be set to a specific value instead of an arbitrary one. The robot might perform more stably if these values were all calculated.

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

The first hardware improvement to bring to the robot would be to adjust the angle of sensor. Had the sensor been more extruded and facing more to the right (towards the obstacles), it would have been able to detect obstacles sooner. A second improvement would be to solidify the wheels of the robot. The design of the robots with the wheels under the main frame caused the robot to be slightly shaky and not quite stable. This certainly caused higher oscillation during the performance. Finally, a big issue encountered was the lack of adherence of the wheels to the floor. By simply adding rubber bands to the circumference of the wheels, the traction could have been drastically increased and there would be less slipping of the wheels.

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

A PID (Proportional Integral Derivative) controller is another common type of controller that could have been used for this assignment. The “P” (proportional) portion applies a correction that is proportional to the error. The “I” (integral) portion has the function of adjusting the applied correction according the error and the quantity of time it was recorded. Finally, the “D” (derivative) portion reduces the amount of overshooting that occurs during the correction.