Lab 1 Report - Wall Follower

ECSE 211: Design Principles and Methods

Design Evaluation

As it can be seen in figure 1, our workflow consisted essentially of 3 stages: setting up and hardware design, software design, and advanced testing. In this first section, we will cover mostly the first two sections and explain the challenges we faced during those stages.

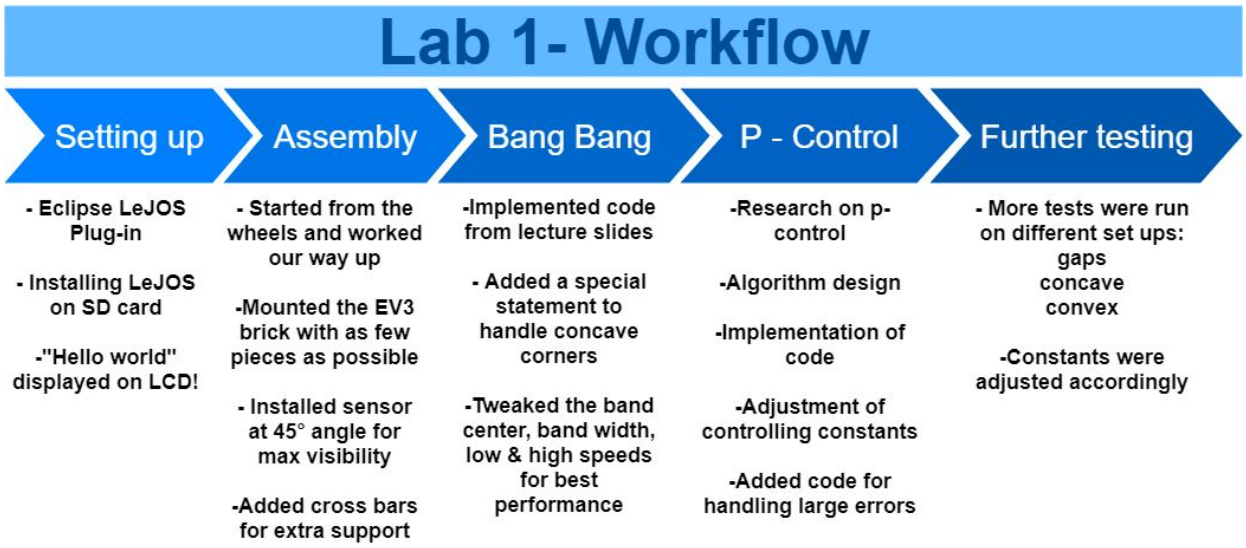

Figure 1: workflow during lab 1

a) Setting up and hardware design:

After performing the required software installations and checking that the EV3 brick is

functional, our first initiative was to assemble the Lego pieces together with some particular constraints in mind: we were aiming for a lightweight and maneuverable robot.

The first design choice that was made consisted of choosing the spacing between the wheels. On one hand, a narrow spacing would come at the price of a decrease in maneuverability. On the other hand, a large spacing allows for a better mobility but requires the use of additional pieces to support the EV3 brick and could lead to a cumbersome robot. Therefore, as it can be seen on figure 2, we chose a spacing of approximately 8 cm that enabled us to use only 2 pieces of Lego to support the EV3 brick. We thus achieved a reasonable trade-off between weight and maneuverability.

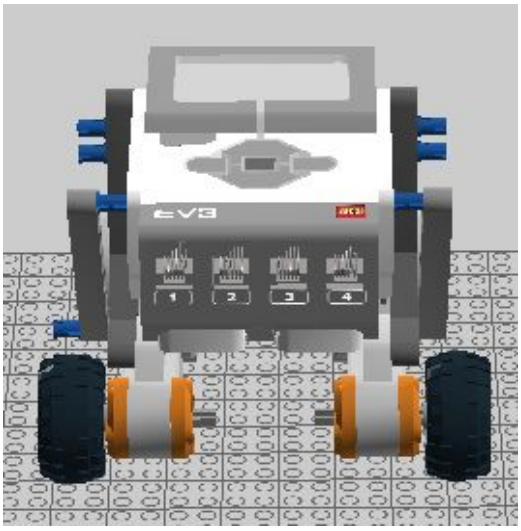

Figure 2: front view of the robot (sensor hidden)

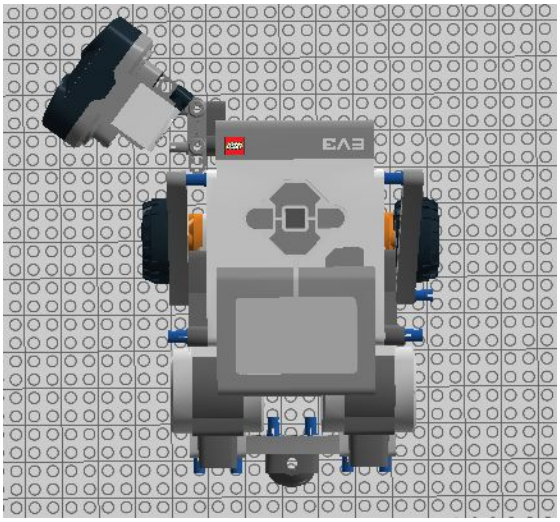

Figure 3: top view of the robot

Once the EV3 brick mounted, we debated on the position of the sensor. As it can be seen in figure 3, we chose to place the sensor at a 45° with the vertical plane in order to be able to detect walls ahead and aside of the robot simultaneously (e.g. in a concave corner) and prevent it from crashing into either of them. Moreover, we intentionally placed the sensor at the robot's front-left corner as we thought that such a position would enable it to detect obstacles in time to have enough space to turn away from the wall.

Finally, as we noticed that the sensor and its fixation pieces were weighing down the front of the robot and could possibly interfered with the clear path of the wheels, a cross piece was added to each side of the vehicle to provide extra support for the EV3 brick (cf. figure 4).

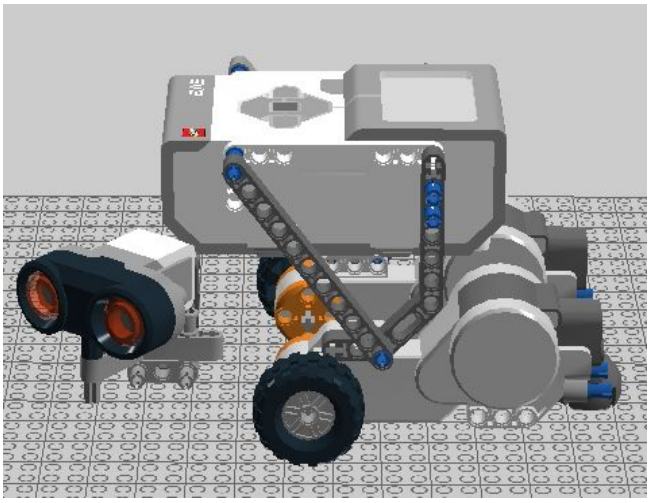

Figure 4: side view of the robot

b) Software design

Once satisfied with the hardware design of our robot, we started implementing the Bang Bang controller. As a first step, we worked together to understand the code provided and the blank parts that we were required to implement. After that, we directly used the code from the lecture slides (with arguments' name fixed): the robot was maintaining the band center distance from the wall and reacting appropriately to convex corners and gaps in the wall. However, when faced with concave corners, the robot did not turn sufficiently fast/early enough to prevent itself from crashing into the wall ahead. Therefore, we added an if block in the code that handled this case: if the displacement error (= band center - US polled distance) is very small but above the threshold (bandwidth) i.e. the robot is very close to the wall, the outer wheel's speed is decreased and the inner wheel's speed is increased in order for the robot to quickly distance itself from the wall ahead of it. The values of the speed change for both wheels were determined by conducting tests and noting down the performance of the robot (still crashes into the wall, turns too fast and spirals out of control, or turns at the appropriate speed to avoid the wall). Please refer to following sections for more information about the testing phase.

After satisfying results from the Bang Bang controller, we moved on to the next controller to implement. This phase started with some online research to fully grasp the concept of proportional control. We understood that, in contrast with the previous controller, p-controlling requires adjusting the speed relatively to the calculated error rather than adjusting it by a constant factor. From that point on, the implementation was fairly straightforward. Whenever the processUSData() function is called, the displacement error is calculated. If the error is above the threshold, we used the $\Delta speed = Kp \times error$ equation to find the adjustment speed. The latter was added to/subtracted from the speed of the appropriate wheels in order to reestablish the fixed distance with the wall. Preliminary tests allowed us to choose the appropriate constants (cf. following sections for more details on the choice of the Kp).

Those tests also urged us to add some code to handle the case where the sensor would get too close to a wall - usually in concave corners - and start returning colossal distances. Since the correction was proportional to the error, the robot would overshoot its adjustment and take unnecessarily large turns that would sometimes take it completely off of its trajectory. Therefore, we added some if statements that would bring down the value of the error if it passes over a certain value. Hence, the correction was smoother and the robot stopped taking unnecessarily large turns.

Test Data

a) Testing the p-controller constant

After running preliminary tests, we determined that a Kp of value 6 yielded the best results. In this section, we run additional tests with values below and above our chosen constant and note down the results that will be discussed in a later section.

Kp	Lap	Band center	Oscillations	Additional remarks
2	Not completed	Inapplicable (did not finish lap)	Very low, within bandwidth	Ran into the corner wall
10	Not completed	Inapplicable (did not finish lap)	Oscillates a lot	Overshoots the correction speed when turning

b) Bang Bang controller tester

Trial	Wall description	Band center	Oscillations	Lap
1	Gaps in straight walls and large corner wall	Successfully maintained	Moderate and easily within the band width	Completed
2	Gaps in straight walls and narrow corner wall	Successfully maintained	Moderate and at the limits of the band width	Completed
3	Gaps in straight walls and in corner wall	Moderately maintained	Oscillates more than the previous 2 laps but within the band width	Completed

c) P-type controller test

Trial	Wall description	Band center	Oscillations	Lap
1	Gaps in straight walls and large corner wall	Successfully maintained	Low and within band width, smooth turns	Completed
2	Gaps in straight walls and narrow corner wall	Successfully maintained	Low and within band width, smooth turns	Completed
3	Gaps in straight walls and in corner wall	Moderately maintained	Moderate and within band width, rougher turns	Completed

Test Analysis

a) Testing the p-controller constant

As it can be seen in the data collected, when the proportional gain constant is lower than the one used for the demo, the robot does not succeed in completing a lap. Indeed, the robot would not turn in time to avoid the wall ahead of it. We can associate this result to the fact that the proportional gain constant and the resulting $\Delta speed$ are too low to perform the required turn to avoid the wall. The low constant can also explain the lack of oscillations of the robot: the change of speed inflicted on the wheels was not high enough to make the robot significantly change its direction.

Similarly, when the proportional gain constant is higher than the one used for the demo, the robot does not succeed in completing the lap. Indeed, the robot would often spiral out of its planned trajectory after trying to avoid a corner wall or make a turn around a brick. We can associate this result to the fact that the proportional gain constant and the resulting $\Delta speed$ were too high: the robot would accelerate too fast when it realizes that it is close from a corner wall. By doing so, it would go off its trajectory by making an unnecessarily large turn (best case) or by starting to run in circles around itself (worst case). The overly large constant also explains the frequent and agitated oscillation behaviour: corrections were too high and would throw the robot off its trajectory forcing it to have to perform more corrections to get back to the band center distance.

b) Bang Bang controller tester

For simple wall configurations (trials 1 and 2), the robot successfully maintained a fixed distance from the wall and did not exceed the bandwidth when oscillating. However, for the third configuration of the wall (trial 3), the robot experiences more remarkable oscillations and leaves the band center by no more than a couple of centimeters when it faces a gap in the corner wall.

c) P-controller tester

Similarly to the Bang Bang controller tests, the robot successfully maintained a fixed distance from the wall and did not exceed the bandwidth when operating around the first two wall configurations. However, the oscillations were not as frequent and the turns were smoother than when the Bang Bang controller was used. For the third wall configuration, the oscillations were more observable when the robot passed by the gap in the corner wall as more adjustments had to be made. The turn near the corner wall gap was also rougher and the robot would go slightly above the bandwidth. Overall, however, the robot maintained a fixed distance with the wall and stayed reasonably within the bandwidth.

Observations and conclusions

Based on our tests and their results, we would choose the p-controller as it provided a more stable robot with a low oscillation frequency rate and smoother turns around corners and edge bricks.

Regarding the ultrasonic sensor errors, we observed more false negatives than false positives. Indeed, we noticed that, when a sensor is very close to a wall (e.g. turning in a concave corner), the US poller shows high values instead of indicating low values corresponding to the proximity with the wall. The erroneous values were low enough to pass through the filter provided in the starter code but high enough to cause problems for the trajectory of our robot. As mentioned in the software design section, these gigantic false values were handled in the p-controller by scaling them down to a lower value when they are detected.

Further improvements

In order to improve the performance of the controller, here are some suggestions of modifications to the hardware:

- We suggest to move the sensor to the side of the robot and add another sensor to the front. This will allow the controlling unit to accurately keep a fixed distance from the wall and to detect a corner wall coming up ahead of the vehicle in order to appropriately plan a turn.
- If an additional ultrasonic sensor is unavailable, we can use an infrared sensor instead
- If no other additional sensor is available, we could alternatively mount the ultrasonic sensor on a motor that would make it sweep an area of 270° degrees. By sweeping this area, the robot would be able to detect the walls on its side and in front of it and take appropriate action to maintain the band center from the walls.

We can also improve our software in order to address the ultrasonic sensor errors:

- First, we can run additional tests to tweak and improve the constants that we used.
- Second, we suggest to add code to handle better the turns around the bricks and the corners: e.g. for the p-controller, we can create more specific proportional functions for different detected distances. The shorter the distance detected, the greater the coefficient of the proportional function (so that the robot turns faster).
- Finally, we suggest to use another type of controller that would register the systematic error of the sensor and act upon it. Indeed, in ECSE 200 (Electric Circuits 1), we were introduced to the concept of a PID (Proportional Integral Derivative) controller. This controller, like the p-controller, would take note of the error from the desired wall distance but would also register the past deviations of the robot and the rate at which it is deviating from the intended distance. Combining all this information, the controller would perform the necessary actions to maintain the fixed distance.