Name: Yicheng Song Name: Tian Han Jiang Course code: ECSE211

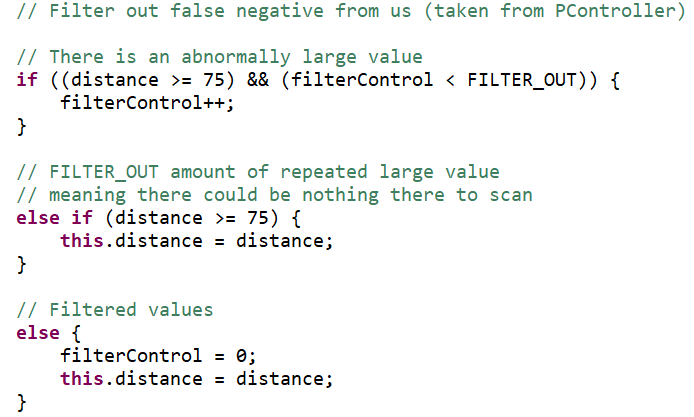
ID: 260763294 ID: 260795887 Group:51

# Report for wall-following robot

**Design evaluation**

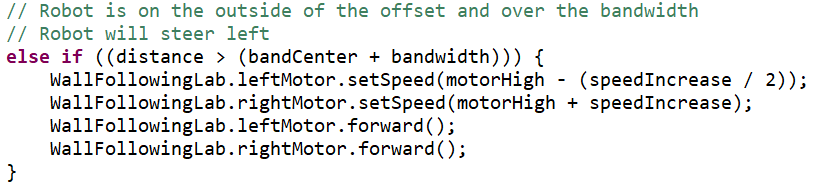
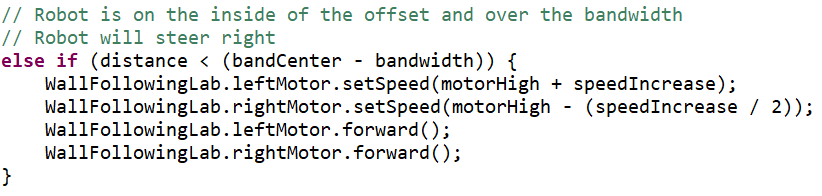
The robot as well as its controller are specifically designed for implementing the wall-following system.

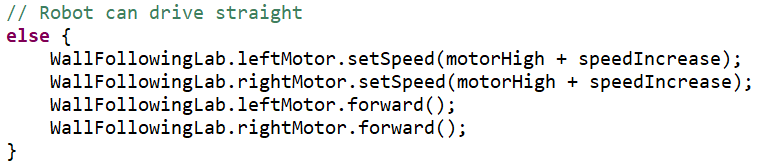
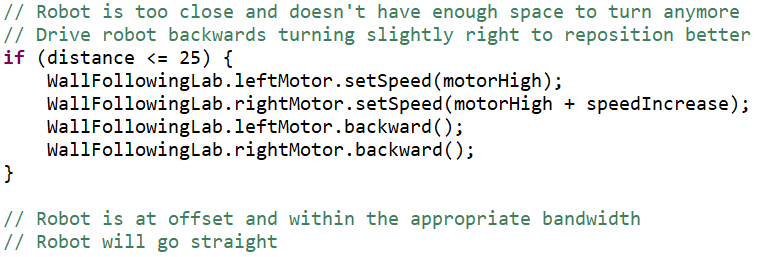
The robot is driven by two motors on two sides beneath the EV3 Brick. There’s a direction-control wheels at the tail of the robot which can also help with balancing it. By placing the ultrasonic sensor at the front with a degree, the robot can measure its distance to the wall, which data will be supplied as reference to implement the speed-changing process to one of or both motors in order to tell the robot how to move and keep it from the wall at a constant distance approximately. Both motors together with the ultrasonic sensor is connected to the EV3 brick with given cables.

The main part for software work is the controller class. For the bang-bang controller, there’s a filter method that works as a data-filter for ultrasonic sensor to test whether or not the data can be used. If a distance is measured which is over the bandwith for 20 times then it will be defined as nothing to scan.

The filter method

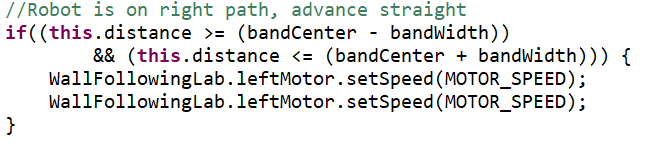
If the robot is too close to the wall, it will backward and turn slightly right to reposition better. For those data that is within the threshold (bandwidth to the bandcenter), the robot will go straight. When the robot is on the inside of the offset and over the bandwidth, it will steer right with left wheel speed up and right wheel speed down, vice versa.





From left-to to right-down: Backward method, Turning left method, Turning right method, Forward method

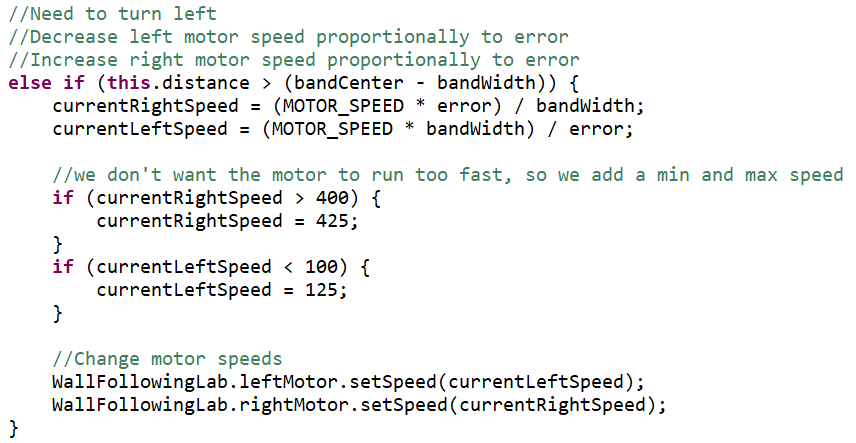
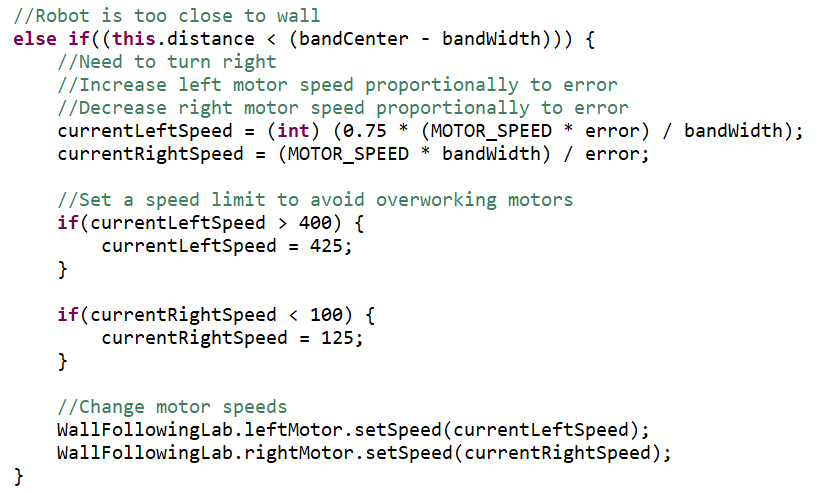
For the p-type controller, the principle is slightly different. The filter method is almost the same. There’s a variable called ‘error’ that works as a proportion. The robot will move forward when on the right path. If it is close to the wall, the method that we take to make it turn right is to increase left motor speed proportionally to error and decrease right motor speed proportionally to error. Similar method also works when the robot is too far from the wall.



Left: Forward method

Left-down: Turning right method

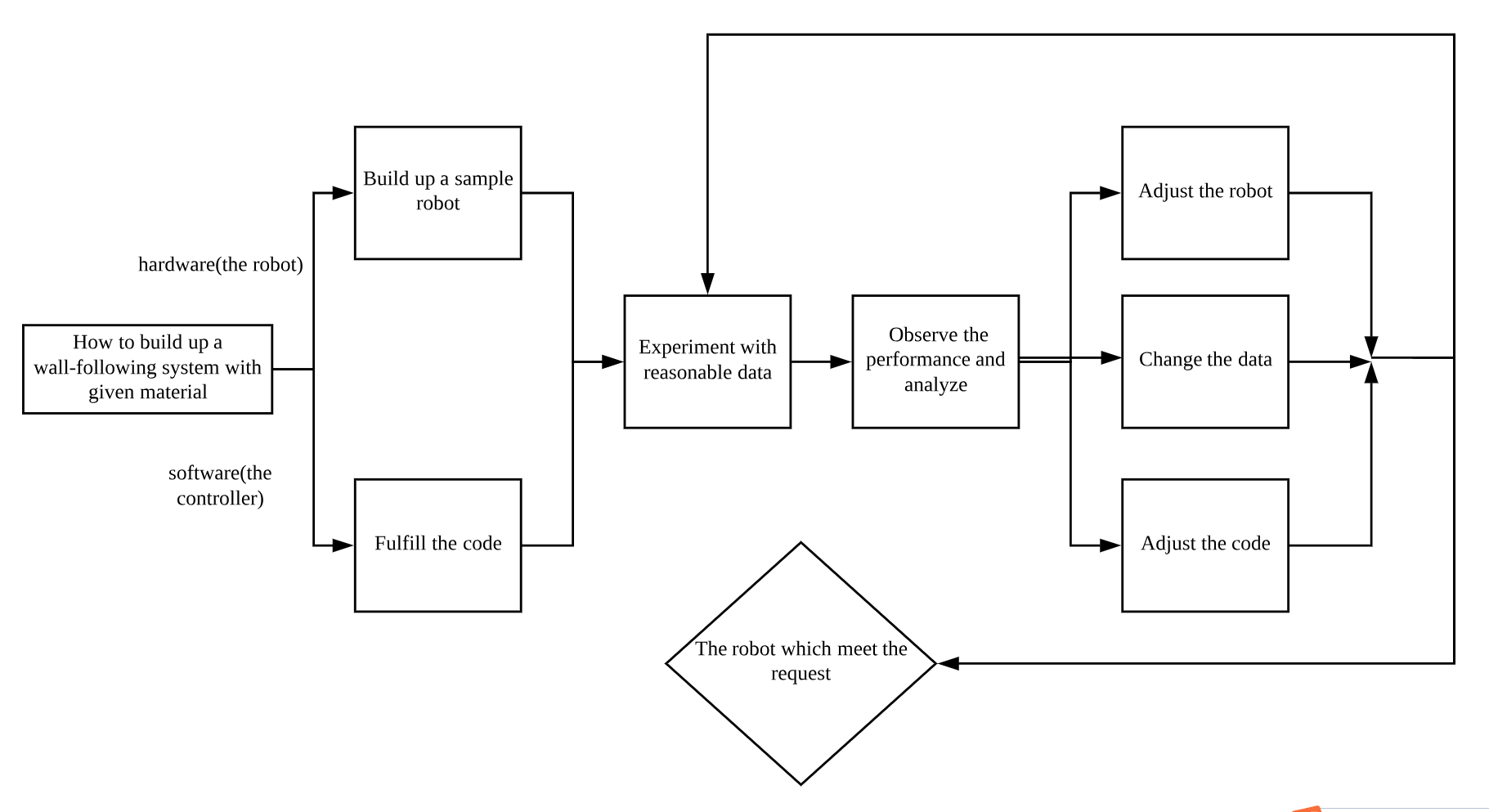
Down: Turning left method



Before reaching the final hardware design, we considered about changing the position of EV3 brick but failed. The direction-control wheel occupies the space between motors so the brick needs to be suspended above the wheel. It may spend to many blocks to achieve the goal and the supporting block seems a little bit fragile. However, we found that the robot is not a smart choice for the test due to its large volume. The robot for demo should be much smaller and is able to pass through the corner. As a result, we finally reduce the space between the two motors, put the EV3 brick upon them and change the position of the direction-control wheel at the tail and finally make it suitable for the test.

The process of finding appropriate constants of both controllers are not quite simple. It takes plenty of time to test whether or not does the constant works. This extremely reflects from the process of finding suitable proportion of p-type controller. We first take error as 2 and simply multiply it to the initial speed and that is a terrible. The robot constantly hitting the wall at where it should turn. Then we change the error to 1 but the test only success for the first corner, so we try to change the way of changing speed. We give a coefficient which equals 0.75 to the left motor so that it could react not too fast in avoid of hitting the wall. Actually, this trial works and finally passed the test. When it comes to bang-bang controller, things become not so difficult to deal with. The bandwidth can be simply estimated and be tested by trial and error.

In summary, the design of both hardware and software parts are considered in detail. It takes us time and patience to deal with the problems we faced.



**Test data**

Testing the P-type controller constant

***1. Choose 2 values above and below your P-type controller constant used in the demo. 2. Run the robot using the P-type controller.***

***3. Note its performance, i.e. band center and oscillation behavior, for the 2 cases.***

Bang-Bang controller test (3 independent trials)

***1. Place the robot at the starting corner of a wall.***

***2. Ensure the wall contains convex corners, concave corners, and gaps.***

***3. Run the robot using the Bang-Bang controller.***

***4. Check if it completes a lap without touching the wall.***

***5. Note its performance, i.e. band center and oscillation behavior for each trial.***

P-type controller test (3 independent trials)

***1. Place the robot at the starting corner of a wall.***

***2. Ensure the wall contains convex corners, concave corners, and gaps.***

***3. Run the robot using the P-type controller.***

***4. Check if it completes a lap without touching the wall.***

***5. Note its performance, i.e. band center and oscillation behavior for each trial.***

**Test analysis**

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

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

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

***Describe how this occurs qualitatively for each controller.***

**Observations and Conclusions**

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

We prefer to take bang-bang controller. At very beginning we should claim that both of the controllers work and it is hard to tell which one is better. The bang-bang controller is chosen just because it is uncomplicated to implement. Although the oscillations appear more frequently than p-type controller, especially when the robot has just turned at the corner, the robot is always on its right path and can adjust to its right path as fast as possible if there’s a deviation. As a result it is more flexible compared to p-type controller in many cases.

***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?***

The ultrasonic sensor hardly produces false positives but sometimes there are false negatives. During our tests there wasn’t any false positives (or, if any, quite few). However, there were some false negatives exist. The frequency is low but when the robot is turning, it seems a little high. We thought the false are due to the degree of the ultrasonic sensor. If the sensor face towards the wall, then the robot reacts too fast so that the it may hit the wall and the sensor fails to detect. Similar conditions appear when there’s a gap that is too large or the degree of the sensor is small. This kind of false can almost be filtered by the filter method in the controller class, and give an angle (for instance, 45 degree) to the sensor also works.

**Further improvements**

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

-A backward method can be implemented to bang-bang controller so that if the robot is too close to the wall due to the sensor errors, it can go back to the previous position and re-measure the distance in order to act as it should be.

-We can improve the filter method in some ways. For instance, the data to be taken to justify whether there is something to detect can be larger (larger FILTER\_OUT). It may take more time for a robot to complete one lap, but this kind of change is one of the most effective ones

-A stop method can be used. When there’s a data that is with large difference to the previous one (for example, is 30% smaller or larger), then the robot should stop and keep the sensor working to measure a number of data (for example, measure the distance 10 times) and calculate their variance. If the variance is smaller then a given number (for example, 1) then the data measured by sensor at the first time is correct so that the robot can take action to react. Otherwise the data that is quite different from the previous one is an error and should not be considered.

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

-Lower the height of ultrasonic sensor so that it could measure the distance more accurately. Or we can use some other kind of sensors that are more precise.

-The position where the EV3 lies could be changed in order to lower the center of mass , and the wheels can be changed with tracks so that the robot could be more stable and has lower possibility to slip on the ground. For example, the EV3 brick could lie down at the center of those wheels. If not doing so, the robot may have the risk to rollover when implementing bang-bang controller.

-The distance between the two motors can be reduced in order to decrease the radius of rotation, which will lead to a more frequent speed-changing process which make the robot to move faster in both bang-bang controller and p-type controller. This means the robot could move along the wall more smoothly and keep its distance from the wall in a smaller threshold at the same time.

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

We can use the derivative controller, which take the distance measured by the ultrasonic sensor and calculate its derivation of time **t** to get the how fast the robot is approaching or getting away from the wall. Then the speed could be taken as the data to change the speed of both motors (for example, when the robot is closing to the wall, find its speed and add up to the speed of left motor while reduce the speed of the right motor by the same degree).