ECSE 211 Design Principles and Methods

Lab 1 Wall Following

Group 53
Spencer Handfield
260805699
Yi Zhu
260716006

1. Design evaluation

The EV3 wall follower robot consists four major components: two motors, one ultrasonic sensor and the brick. The brick, which functions as the brain of the robot is placed on two parallel, large EV3 motors. It is fixed by four short shafts on the bottom and two vertical pillars on sides. The reasoning behind this particular design was to minimize the width of the robot and thus lower the risk of collision during maneuvering. The ultrasonic sensor is placed on the front left of the brick in a 45-degree angle to guarantee detection on both the front and left side, giving it an accurate assessment, based on the conal projection of the sensor, to see the wall next to it, as well as predict incoming obstacles. In addition, we place it at the bottom, to reduce the false detection over the wall. Through trial and error we observed that this placement often gave more consistent and accurate readings. All motors and the ultrasonic sensor were connected to the brick by Ethernet cables.

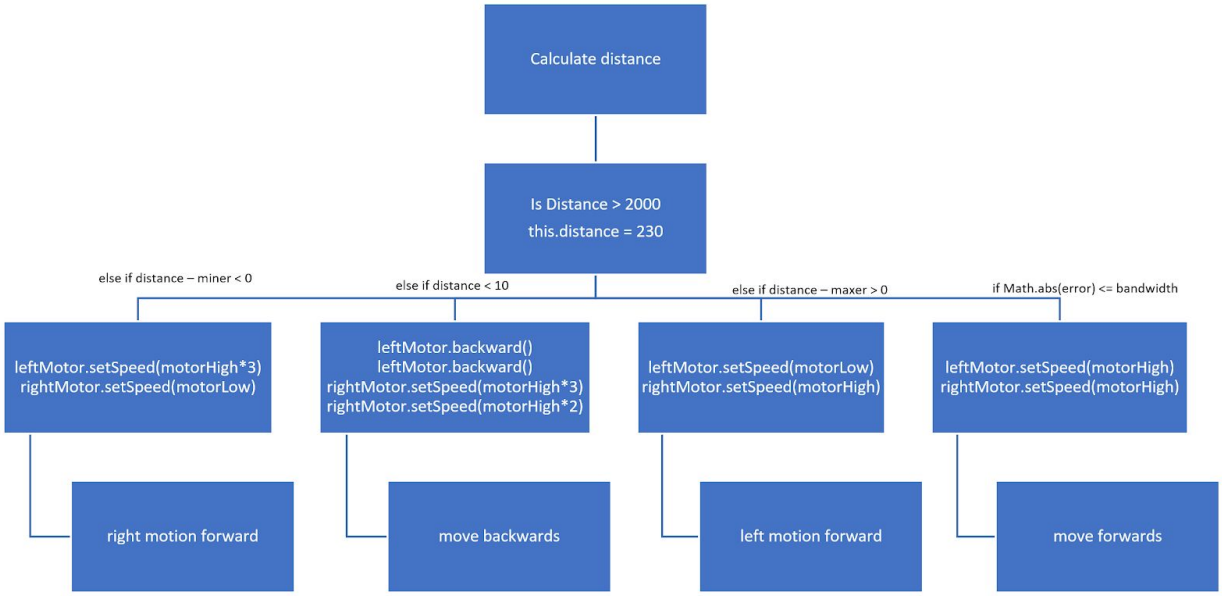
The basic idea for the wall following is using correction to keep the robot in a safe range from the wall. There are two types of operating mechanism implemented in the Wall Following program: Bang-Bang and P type Controller. The Bang-Bang controller works as a binary feedback loop that switches between two states for error correction: on or off. The P-controller on the other hand undertakes error connection based on proportionality, i.e. its correction and relative speed, adjust dynamically to the magnitude of the error that must be corrected. The mathematical relationship of said relationship used in our software is the following:

$$ErrorCorrection = ErrorMultiplier \times |Distance - BandCenter|$$

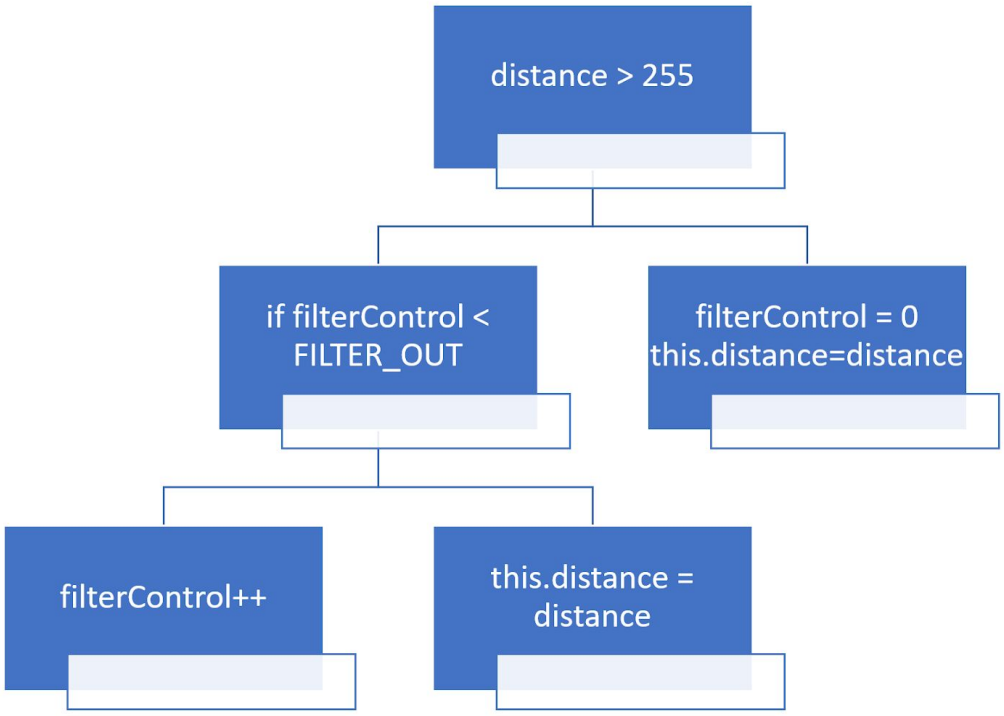
The Software development was simply the prescriptive approach of the aforementioned descriptive explanation. At its core the software is relatively simple using simple polling from the sensor and control to the wheels, a series of if statements check for any possible error case and adjust the speed of the wheels accordingly. Certain extreme cases arose that were added functionality to the robot such as reversing when very near the wall and capping the level of adjustment the P-controller could undertake so as to not create problems in a real world setting. These real world issues were tested upon and used to optimize the code. Numerous runs of the blocks allowed us to effectively calibrate ideal bandcenter for our set of correction values which working in tangent allowed for overwhelmingly successful trials. The only other major component added to the software was due to hardware limitations. The sensor would often falsely read an extremely large value, which when passed to our software, would become so large an integer data type would not be able to handle it and would result in a negative float and thus incorrect movement. Through testing we capped the measurable distance at a degree much higher than the ever observed maximum from the sensor, but still well below the

capacity to falsify entire data types. The overview of the software design can be found below.

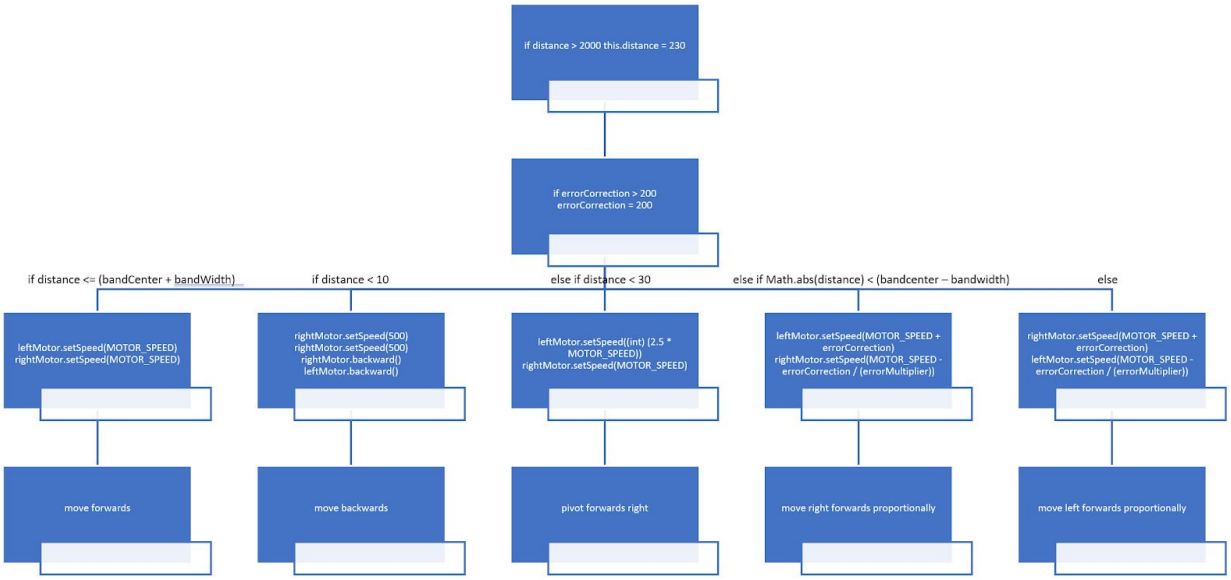
Bang-Bang Controller:



Filter for P Controller:



P Controller:



2.Test data

2.1 Bang- Bang Controller

Test #	Does it complete the lap?	Band Center(cm)	SpeedHigh (deg/sec)	SpeedLow (deg/sec)	Observations
1	No	20	200	100	The robot moved too slow to get away from the wall when near collision
2	No	20	230	150	The robot cannot pass the concave corner
3	No	30	230	150	The robot cannot pass the 180 degree turn
4	Yes	35	230	150	Robot stayed in a

					safe distance from the wall and made precise turn

2.2 P Controller*

Test #	Does it complete the lap without hit?	Band Center(cm)	ErrorMultiplier	Observations
1	No	35	3	The robot made sharp turns and was too slow to get away from the wall
2	No	35	5	The robot made huge 180 turn and collided after turn

*Since we chose 4 as the ErrorMultiplier in our demo, here we picked 3 and 5 for test.

3. Test analysis

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

If the ErrorMultiplier is higher than our demo value, the robot would make a sharp 180 degree turn and would often hit the wall during the turn. If the ErrorMultiplier is lower than the value that we chose, the robot will make relatively large turn and will have risk hitting the wall after the turn due to the suboptimal angle of approach.

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

With the Bang-Bang Controller, it oscillated from -3cm to 3cm from the band center and in P Control it oscillated from -3cm to 7cm from the band center.

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

Yes, especially in P Control. Since the error correction in P Control depends on the distance from the wall, if the robot is not too far from the wall, the speed of correction will be relatively low. During 180 degree turn, the distance from band center will be higher than Bang-Bang Controller, reaching 7cm.

3.4 Describe how this occurs qualitatively for each controller

It happens more often on P Controller than on Bang-Bang Controller. Since the error correction in P Controller depends on the distance from the wall, if the robot is not too far from the wall, the speed of correction will be relatively low. It will take a longer time for the robot to back to the safe range.

4. Observations and conclusions

Since the Bang-Bang Controller only has fixed correction value, the robot maintained relatively stiff performance under this mechanism. It increased the chance of over corrections during the wall following process and incurred more correction actions than P Control. While the correction value of P controller depended on the error distance, large errors would trigger large correction and vice versa, leading to a more measured and appropriate response from the system. Building off this idea, often when entering the error zone the Bang Bang would corrected abruptly and push it out of the opposite error zone where it would again jerk back to repeat the process versus the P-Controller which, once in the desired zone would overarchingly maintain a straight trajectory with minor adjustment. Furthermore, less over corrections or unneeded actions would save more energy for the robot and its battery as well as lower the risk of collision by acting almost preemptively rather than reactively. In conclusion, P Controller would be our first choice.

The main obstacle at the end of our development was that the ultrasonic sensor detected very large distances when there was nothing in front of the robot. A distance of 21474 would be displayed on the LCD screen, an impossible case in the lab and was thus concluded to be a hardware limitation.This situation would occur five times during the average loop. After some research and testing with print commands, we discovered that it was actually a case of data overflow as mentioned in the previous section. This meant that the robot would be receiving, as it turns out, a negative reading when the distance from the wall was actually large and based on our software design, would undertake the wrong correction. We filtered it out by setting the largest value distance that the brick can receive to be 200, this result was tested to prove its effectiveness and relative relevancy. If it received a value larger than 200, the distance would be set to 200, leading it to produce the largest possible correction value, the cap set forth by the assignment.

5. Further improvements

5.1 Three software improvements that could address the ultrasonic sensor errors

1.Filter control for gaps: using a filter control method to distinguish gaps and 180 degree turns would allow the robot to make decisions more accurately as to whether to maintain trajectory (gap) or actually begin turning (corner)
2.Filter control for infinite far distance data and overflow: using filter control to handle extremely high value received by the limited ultrasonic sensor.
3.Emergency backward moving: set an emergency distance at which instead of correcting, the robot backs up to avoid immediate collision while also pushing it into a more appropriate distance where it can make its usual, yet now, more safe adjustments

5.2 Three hardware improvements that could improve the controller performance

1.Multiple ultrasonic sensors: install more ultrasonic sensors on sides and back to avoid collision omnidirectionally.
2.Reduce axle width: reduce axle width to avoid tire collision during 180 degree turn.
3.Reduce robot length: reduce robot length to reduce the risk of collision during moving backwards.

5.3 Other controller types

Using PID Controller(proportional–integral–derivative controller) would improve the performance of robot since it calculates a much more precise correction value allowing for more accurate tracing of the wall. This mechanism would be more effective, at the added cost of minor complexity enhancement.