ECSE 211 Lab 1 Report Group 33

Authors: Aleksas Murauskas, 260718389 and Talaal Mazhar Shafi, 260865281

1. Design Evaluation

This portion of our report will discuss the hardware and software design of our Wall-Following Robot. After downloading and setting up LeJos on our laptops and our bricks, we first worked on our hardware's design. Our design's key components included the EV3 brick, and ultrasonic sensor, two motors to power two forward wheels, and a rear rollerball. The motors were placed directly under the EV3 brick to allow the robot to make tighter turns, with the rollerball centered in the rear of the device. The sensor was kept close to the robot and at a 45 degree offset from the vehicle's forward face, this was to ensure the sensor could properly interpret corners in the wall, both concave and convex. We then began building our first prototype. Upon building the actual robot, we discovered the sensor was too close to the brick for the the wires to connect to the US sensor's rear socket. To fix this issue we created a stabilized arm visible in Figure 2 to allow the wires to properly connect to the sensor to the EV3 brick. When Trials began, we built an arch above the EV3 brick to keep the wires vertically suspended, rather than fall to the sides or rear of the device to avoid unintentional collisons. The Device displayed in Figures 1, 2, and 3 was our final design for this lab.

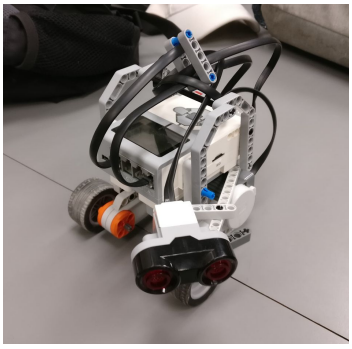

Figure 1

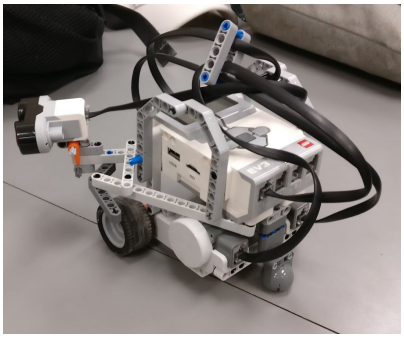

Figure 2

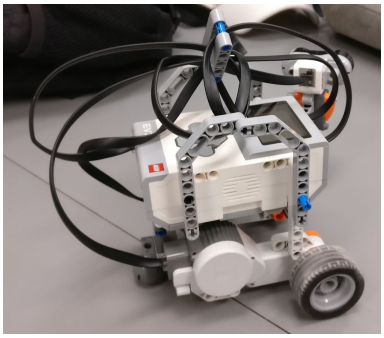

Figure 3.

On the software front, our program starts when the WallFollowingLab program is executed on the EV3 device. Afterward, a message appears requesting the user to enter their desired controller type, left for BangBang and right for P-Type. Both follow a standardized feedback loop shown in simplified form in figure 4. Both would take an input from the US

sensor, and first a filter method [the given rudimentary filter] would determine if the distance was valid, if not the distance would be ignored by the rest of the method. Next the program would calculate error using the following equation. $Error = BandCenter - Distance$
The value of this error would determine whether the robot would go straight, left or right. The key difference between the two controllers was how they accomplished their turns. Bang Bang used hard coded values to determine how fast the wheels should turn when too close or far, while P-Type used a proportional correction with the following equation.

$$CorrectionValue = |error * 5| \qquad if\ (CorrectionValue > 200)\ \{CorrectionValue = 200\}$$

We found the constant 5 through tweaking and tests of trial and error. A constant too large would result in a device too jerky, while a constant too small would result in The cap on correction value was introduced after 180 degree turns would result in a turn too large, resulting in a collision before the sensor would recognize it was too close to the wall. The cap prevented this from happening in subsequent tests. We used a BandCenter of 25 and a Bandwidth of 3 for both controller types. We felt these two values gave us the most consistent lap around objects. However our robot was still having difficulty navigating concave turns. We then introduced another if statement, if the robot was less than 18 or 20 [for BangBang and P-Type repsectively] the robot would spin in place, with one motor reversing direction. The distances were found through trial and error tweaking, but had to be outside the bandwidth.

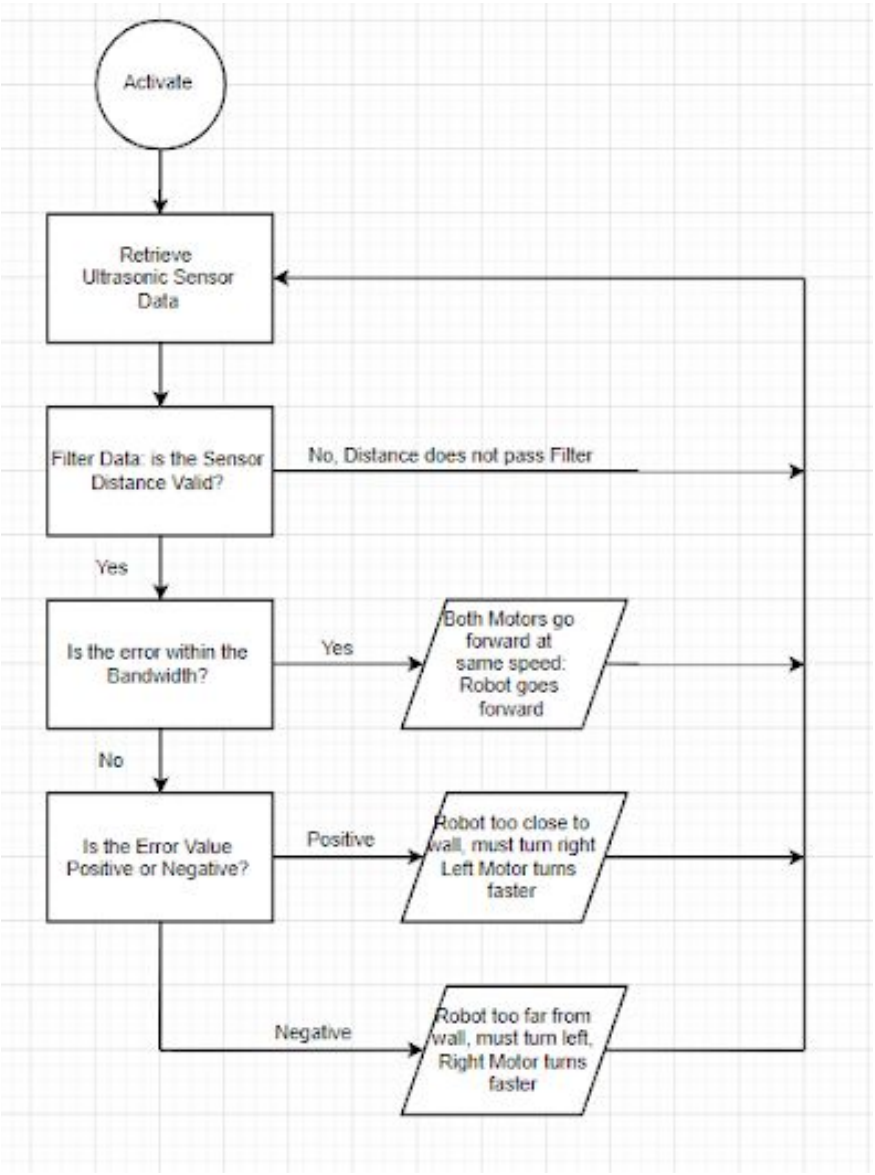
Activate
Retrieve Ultrasonic Sensor Data
Filter Data: is the Sensor Distance Valid?
No, Distance does not pass Filter
Yes
Is the error within the Bandwidth?
Yes
Both Motors go forward at same speed: Robot goes forward
No
Is the Error Value Positive or Negative?
Positive
Robot too close to wall, must turn right Left Motor turns faster
Negative
Robot too far from wall, must turn left, Right Motor turns faster

Figure 4.

2. Test Data- Tables including tests measurements

P Controller Constant Test

Trials	Speed Constants and Variables	Band Constants and Variables	Observations
1	AdjustVal = Math.abs(error*10) MOTOR_SPEED = 200 No speed cap	BandCenter = 25cm BandWidth = 3cm	The robot initially followed the line well, but corrections were too large, leading it to continuously oscillate around the bandcenter over a large band. The robot was able to follow the wall until a convex turn where the robot overcorrected into the wall.
2	AdjustVal = Math.abs(error*1) MOTOR_SPEED = 200 No speed cap	BandCenter = 25cm BandWidth = 3cm	The robot followed the Band quite well, making only very small adjustments. This value was too small to properly turn. Concave turns would result in a collision with the wall, as the robot would not turn fast enough to avoid contact.

Bang-Bang Controller Test

Trials	Speed Constants and Variables	Band Constants and Variables	Observations
1	Motor Low= 160 Motor High = 320	BandCenter = 25cm BandWidth = 3cm	The robot oscillated around the band far too jerkily and rarely ever rolled just forward. It often overcorrected to return to the band, and this overcorrection caused the robot to collide attempting to pass a convex turn at the end of the wall.
2	Motor Low= 120 Motor High = 240	BandCenter = 20cm BandWidth = 3cm	This band was much closer to the wall, so the robot followed the new distance. However, at this new distance close concave turns were taken too slowly, resulting in a collision.
3	Motor Low= 120 Motor High = 240	BandCenter = 25cm BandWidth = 3cm	The robot oscillated around the band distance somewhat jerkily but was able to navigate the wall for a full loop.

P-Type Controller Test

Trials	Speed Constants and Variables	Band Constants and Variables	Observations
1	AdjustVal = Math.abs(error*5) MOTOR_SPEED = 200 AdjustVal cap at 200	BandCenter = 25cm BandWidth = 10cm	The wide band allowed the robot to roll forward more often than the other trials. However, the robot would approach the wall too closely before turning, resulting in an
2	AdjustVal = Math.abs(error*5) MOTOR_SPEED = 200 AdjustVal cap at 100	BandCenter = 25cm BandWidth = 3cm	The robot followed the band rather well, making only small adjustments. However the sensor scraped the wall during a concave turn since it wasn't able to turn fast enough.
3	AdjustVal = Math.abs(error*5) MOTOR_SPEED = 200 AdjustVal cap at 200	BandCenter = 25cm BandWidth = 3cm	The robot was able to complete a full lap, with one close call passing at the end of the wall before turning.

3. Test Analysis

Our P Constant tests revealed that a constant too large would result in very large corrections, sometimes resulting in over-corrections, whereas a smaller P-type constants results in a robot incapable of turning fast enough to avoid collision with the wall. Our Bang-Bang Robot oscillates around the bandcenter very jerkily as it's motor speeds are hard coded, while our P-type much more elegantly and closely followed the band. Bang-Bang controller would occasionally exceed the bandwidth by 3 to 4 cm due to the occasional overcorrection. P-Type would do this less often, and it's say from the band was smaller at about 2 to 3 cm. The Bang-Bang controller sways from the band more often because it's code does not take into account how far from the band it is, only using simple values to adjust the path. P-type by comparison is resulting in the robot making a proportional response, therefore it is less likely to overcorrect.

4. Observations and Conclusions

While the bang-bang controller was easier and took less time to code, what we noticed during our trials was that it was the more likely controller type to fail had our hard coded value not been precise. This was proven during our demo too, when during a turn our robot under corrected due to us not setting a strong enough value, and drove out of bounds. The p-type controller, however, took more time to program but produced better results. During our trials we noticed that it was less likely to fail because its correction is proportional to the error, unlike the bang-bang controller which had constant values of shifting the motors. It did take slightly longer to complete the course due to there being more calculations, but the change in speed was not significant enough to warrant any changes. This controller type too failed when demoed as it over corrected during a concave turn. Despite this, after our analysis, we conclude that the p-type controller is the better option.

With regards to the sensor, we noticed that there were times when the sensor would get stuck at a certain value and ignore its surroundings, proceeding to continue going straight. This was always fixed by waving our hand in front of the sensor, allowing it to recalibrate and correctly maneuver the course designed.

5. Further Improvements

Software improvements

- In addition to walls, allow for the detection of moving objects
- Create a more robust filter to interpret data with increased precision
- Increase the frequency of ultrasonic tests for a more time updated reading of distance by sensor

Hardware improvements

- Add an extra sensor to ensure the detection of objects

- Use different motors to allow the wheels to move and allow more efficient turning
- Use tires that are less resistant to the friction of the dirt on the ground of the lab room

For a controller type we could use a PID (proportional-integral-derivative) controller, which is a more sophisticated model of our P-controller. This controller takes external factors into account such as a friction, and works to smooth the rate at which the changes are made when applying its correction, which would lead to a robot which moves around the course more fluidly. This is also the control type is universally used in applications with automatic control.