|  |
| --- |
| Control of Mobile Robotics  CDA4621.001 S17  Dr. Alfredo Weitzenfeld  Wall Following Task  Tyler Simoni & Esthevan Romeiro  February 27, 2017 |

# Lab Objectives

The objectives for this lab were complete the described tasks listed below by using the IR sensors and RGB color sensor. All of these tasks needed to be completed using closed loop functionality. The task execution can be viewed [here](https://youtu.be/sZhrHHfaY2I) on YouTube. Each task below also is hyperlinked to the appropriate timestamp in the video.

* Wall Distance
* Wall Following
* Color Classification
* Corridor Navigation

# Lab Design Overview

## [Wall Distance](https://youtu.be/sZhrHHfaY2I?t=7)

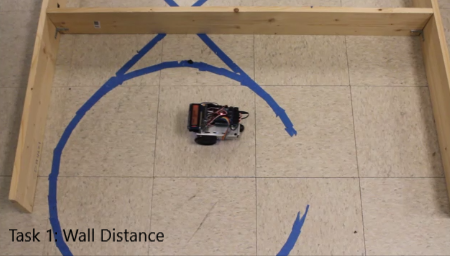
The wall Distance task requires programming of the following five equations in order to detect an object and adjust the robot speeds accordingly.

Figure 2‑1. Wall Distance

1. u(t) = Kp \* e(t)
2. ur(t) = fsat \* (u(t))
3. ur(t) = fsat \* (Kp \* e(t))
4. e(t) = r(t) - y(t)
5. ur(t) = fsat \* (Kp \* (r(t) - y(t)))

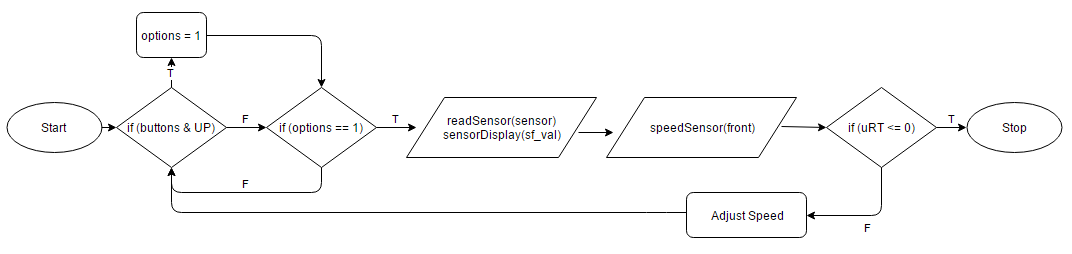
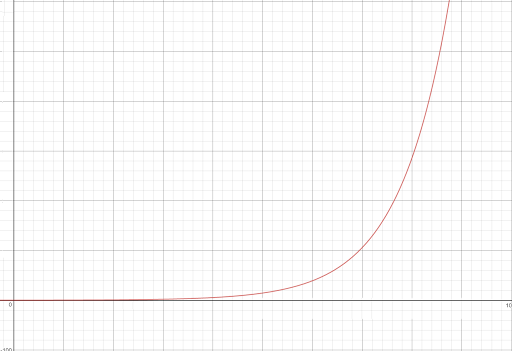
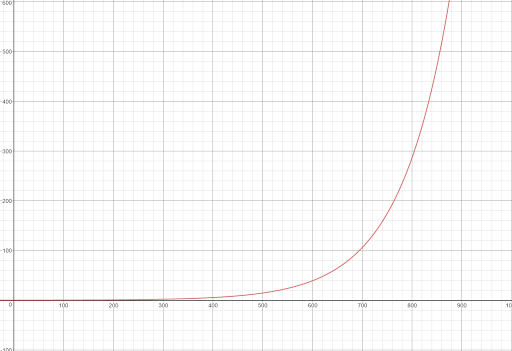
These equations need to be implemented in such a way as to control the velocity – u(t) – to move the robot from an initial distance of 10 inches from the wall to a desired distance of five inches from the wall – r(t) = 5. The robot was to use 5 Kp values (0.5, 1, 2, 5, 20) and apply them to the robots distance code. 

Figure 2‑2. Wall Distance Flowchart

The robot here, started farther than 10 inches just as proof of concept; however, the technique remains the same. The robot starts to move only once the UP button has been pressed. Once that occurs, the robot reads the front sensor and passes the data first to sensorDisplay, which displays the approximate distance on the LCD screen. Then, speedSensor is passed the front sensor data and, using equation 5 from before, generates a proportional gain value. This is then applied to the servo velocity to change the speed as the robot nears the wall. Depending on the Kp value, the robot would either gradually slow down, or come to a quick halt as it neared the goal distance of 5 inches. The smaller the Kp, the more gradual the change in velocity. Below are general graphs representing the time vs distance for each of the Kp values:



## [Wall Following](https://youtu.be/sZhrHHfaY2I?t=19)

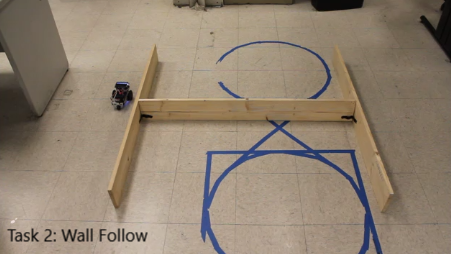
The Wall Following Task requires the robot to move completely around an obstacle while keeping a specified minimum distance of five inches from the obstacle wall. The loop control developed during Task 1 should be implemented in the wall following task. The benefits from the previous task is the ability to correct the robot’s distance proportional to current position versus distance from the obstacle wall. To accomplish the required five inches, the proportional gain (error correction) of the developed function has been modified to satisfy the condition.

Figure 2‑3. Wall Following

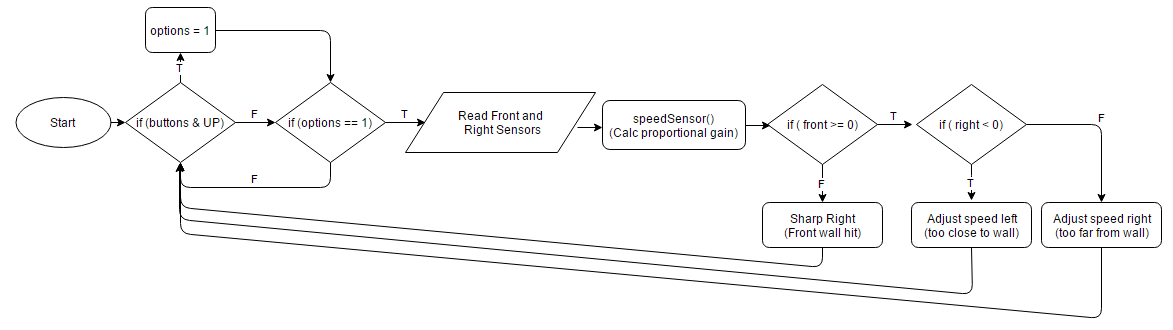


Figure 2‑4. Wall Following Flowchart

The robot performs similarly to task 1. This time though, both the front and right sensors are reading and adjusting the speed of the servos. In order to start the robots path, the UP button needs to be pressed. The robot then proceeds forward with a goal sensor reading of 200, about 5” away. If the robot goes inside this goal, the left wheel slows down proportionally to the distance of the wall. The closer it gets, the slower the left wheel becomes. This causes it to turn left and correct itself towards the goal distance. The same thing happens the robot goes outside the goal distance, the right wheel slows down, causing the same effect in the opposite direction. This same algorithm is used to make the robot turn the outside walls of the ‘H’.

While this is all happening, the robot is also reading the front sensor. The above algorithm all occurs as long as the front sensor is reading outside the goal distance. Once the robot travels inside this distance – for example, if it ran straight into a wall – the robot pivots on its center axis to the left. Once the front sensor returns a reading outside the goal distance, the robot goes back to reading the right sensor.

## [Color Classification](https://youtu.be/sZhrHHfaY2I?t=71)

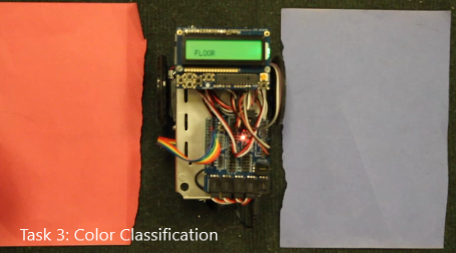
The Color Classification task requires the robot to distinguish between specific colors. The different colors are read through the RGB color sensor located below the robot. The readings of the RGB values from this module do not represent real world RGB values for the colors used; however, the ratio between each Red Green and Blue values maintain similarities seen in actual RGB values. Once each color is read, the robot should display the name of the color that has been read on its LCD screen. The ability to distinguish between read colors will be a key feature in the following task.

Figure 2‑5. Color Classification

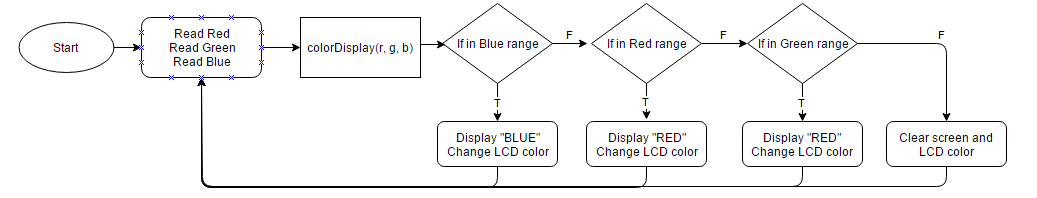


Figure 2‑6. Color Classification flowchart

The task was quite simple in comparison with the others. The robot reads in each of the three colors from the RGB color sensor and then puts them into the colorDisplay function. This takes in the three red, green, and blue values and checks them against ranges. The ranges were found by putting the robot on the color and getting the min and max values read for each of the color’s three RGB values. The ranges were expanded a little bit to make them more accurate at displaying the correct color. The ranges can be seen below in the code for the colorDisplay function:

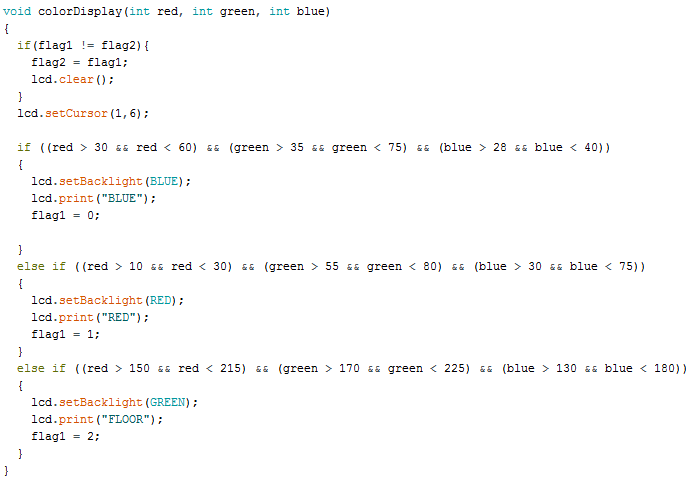


Figure 2‑7. colorDisplay function

## [Corridor Navigation](https://youtu.be/sZhrHHfaY2I?t=88)

Figure 2‑8. Corridor Navigation

The Corridor Navigation task requires the robot to navigate between corridors keeping a reasonable distance from them while locating that it is back on its original starting position. Recognition of starting and ending position will be done via color classification.

This task requires the combination of logic from the Color Classification task and Wall Following task. The robot is required to navigate without hitting any obstacles and stopping once it has recognized that it is arrived back at the origin. This final task is the culmination of all the tasks in one task.

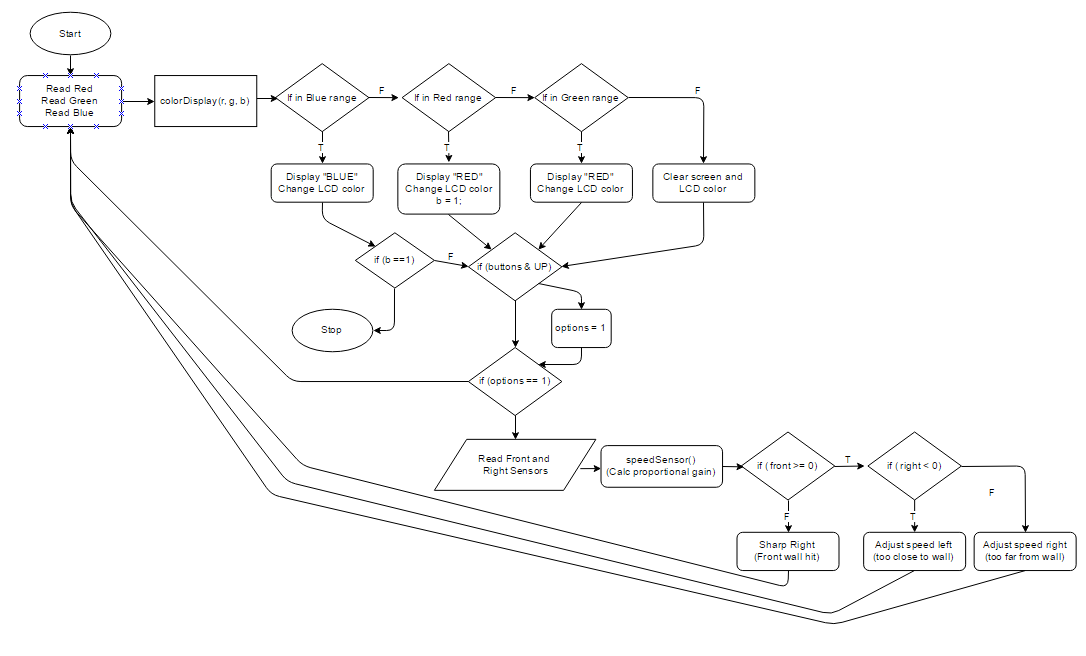


Figure 2‑9. Corridor Navigation Flowchart

For this task, the robot starts on the blue patch, displaying BLUE on the LCD display. Once it leaves the blue patch and goes over the floor, the LCD displays FLOOR and turns green. The robot navigates similarly to task 2, only this time using all three sensors. Instead of a “goal” distance as in task 2, the robot takes the average of the right and left reading, thus making the “goal” the center between the two boards. Once the robot crosses the red patch, it changes the value of b. This variable is a flag to tell the robot to stop the next time it hits the blue patch. The robot continues to navigate, avoiding all walls. Once it detects the blue patch, the robot detaches the two servos, and stops moving.

# Conclusions

All in all, this lab proved to be much more difficult than anticipated. The big issue for this lab was mostly caused from battery discharge. The robots were using many more components than last time, and thusly, drawing quite a lot more voltage. Not only did this create an issue with losing servo power, but it also caused a lot of electronic noise between the IR sensors and the color sensors. Batteries had to be changed much more often for this lab.

Task 1 had no real issues and was fairly straight forward. Task 2 proved to be the most difficult due to IR sensor readings. Due to the thin boards, when taking the outside turns, the robot would sometimes miss the edge of the board and drive in a tight circle instead of following the next straight away. After tweaking down the speeds, the robot was eventually able to navigate around the obstacle. Task 3 was also quite easy and the only real “difficulty” came from getting the ranges to cooperate with displaying the correct colors. Task 4 was a close second in difficulty. Originally, the robot was not detecting color with the IR sensors reading. This was found to be caused by electronic noise and the ranges had to be re-calibrated. After some more tweaking, the robot was successfully able to navigate the maze.