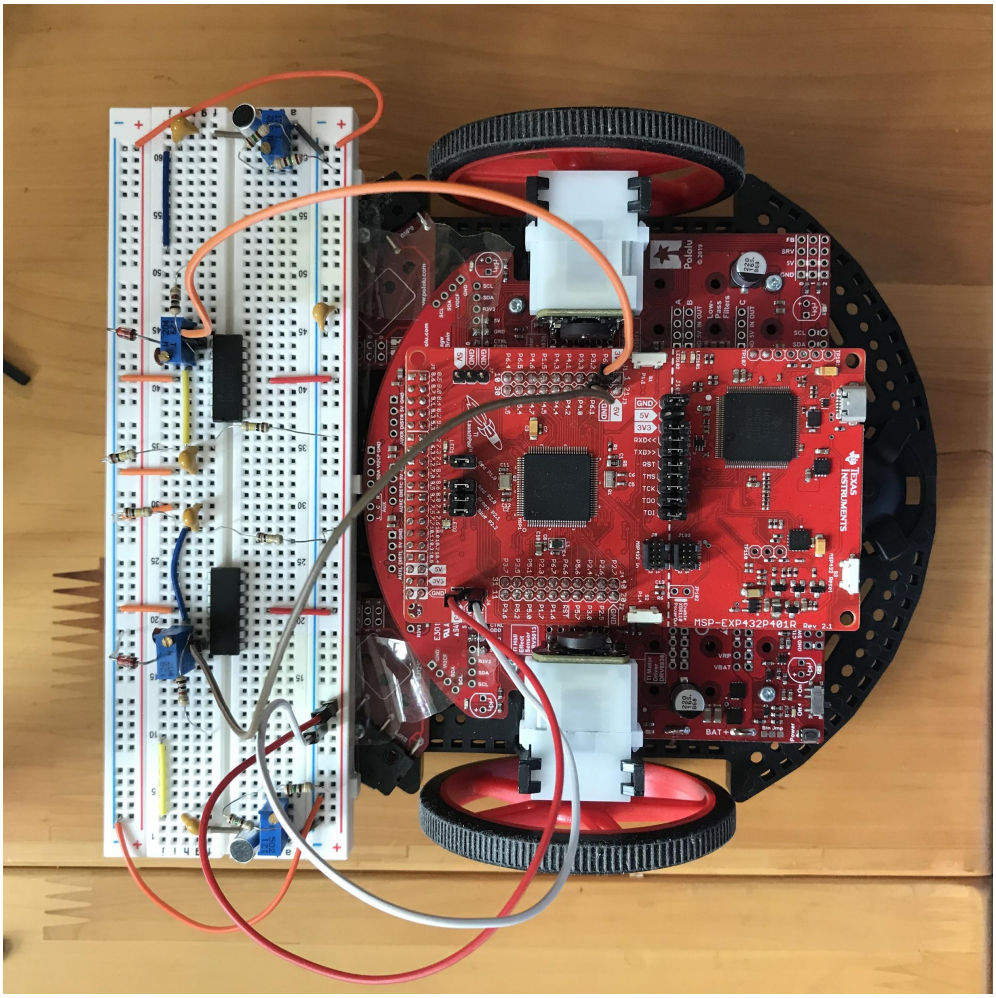
EEC10 Final Project



I. Abstract

In this final project, the robot is equipped with ears and can hear the sound from both left and right. The main objective of this project is to let the robot decide whether it should go forward, left, right, or backward based on the sound intensity it receives. The whole project consists of six parts. The first part, calibrating two microphones to make them have roughly the same magnitude. The second part is design filters to filter out unwanted noise. The third part is to let the robot determine if there is sound and move if there's sound while stop when there isn't. The fourth part is to give it the ability to judge whether to turn left or right based on left and right microphone intensity. And then in the fifth part, the robot now can move to the sound source. And finally the most challenging sixth part, the robot has to turn back if he thinks the sound source is from behind.

II. Introduction

Part1. Re-calibrate the microphone circuits from previous labs.

The goal is to make the magnitude of the two microphones as close as possible. This is critical for the remaining parts of the project because the robot should decide to turn left or right based on the sampling results.

Part2. Design filters to filter out unwanted noise.

This part is also important because from the FTT graph drawn from the raw signal, the noise is pretty visible. So, in this part, the main mission is to build filters for both left and right channels. In this way, the robot can make better decisions on whether to change direction or not.

Part3. Let the robot decide to run or stop.

This is the very first step that we “teach” the robot to run. Basically, the object for this part is to let the robot determine if he “hears” a sound or not. The main challenge is to decide the lower limit level of noise. Then, once decided, if the magnitude is below the limit, the robot stops, otherwise, he runs.

Part4. Let the robot decide to turn left or right.

If the right sound is bigger than the left, the robot turns right, if the left is bigger, he turns left. However, there should be tolerance for the difference between the two channels. In other words, there needs a room for the uncertainty of the microphones.

Part5. Showtime for the robot.

Finally, the robot has “mastered” the skill he needs to move towards a sound source. This part is more challenging because there are more parameters need to be optimized.

Part6. A U-turn for the robot.

This is much more challenging than previous parts because the robot has to know where the sound source is. In the previous part, he might run opposite to the sound source, only knowing to turn left or right. So, this part, the robot should know he is moving away or towards the sound source.

III. Description

Part 1.

The calibrating of microphones is very straightforward. By apply a sinewave sound, it is easy to adjust the gain for each microphone. After changing the Rx for both circuits, the two microphones' outputs are roughly matched.

Part 2.

This part, two filters, a high pass filter and a low pass filter, are needed to form a bandpass filter. When there are no filters applied, according to FTT Magnitude Graph, the noise below 300Hz is quite discernible, so, setting the high pass value to 300Hz is necessary. And according to the Wikipedia page, 300Hz high pass filter has alpha value 0.84. For low pass value, setting it to 3Khz is a good idea, the alpha value is 0.6534. In the code, this value is named "lowP." After that, just applying the previous lab's knowledge will get the filtered value.

Part 3.

The logic for part 3 is simple. If the robot receives a sound, he will move forward, otherwise, if there is no sound, the robot stops. The magnitude of sound that the robot receives is stored in max1[] and max2[]. To determine if there is sound, it is better to use the max value that the microphone receives. According to the code provided, the 1000 samples are divided into 10 groups with size of 100. Then, the highest value in each group will be chosen and stored in max1[] and max2[]. After that, those two arrays have stored 10 data, and we get the average of the 10 data by adding them and divide the sum with 10. Those values are now avgmax1 and avgmax2. After that, the baseline of hearing a sound should be set for there is still some other noise, which is provided by giving offsets (lines 259 and 276). Finally, to determine whether there is sound, there should be a threshold. If the avgmax is above the threshold, the sound is heard, if it is below the limit, there is no sound. The chosen value for this threshold is 0.015, after applying different levels of sound and testing the magnitude of the input.

Part 4.

This part is like the previous part, there should be a threshold to determine if one side's volume is larger. To better interpret the results, the robot better uses the root-mean-square value, according to the hint provided by lab instructions. Getting the RMS value is not hard, we sum the squared value of each sample then divided the sum by sample size, and take square root of it. In the code is denoted by rms1 and rms2. After getting rms1 and rms2, which represent the sound magnitude received by the left and right speakers, the difference between them can be calculated. The method adopted is the relative RMS value, rmsRelative, which is got by getting the absolute value of (rms1 - rms2) and dividing it by the average value of (rms1 + rms2) (line 306). Then it is time to determine the threshold value. The value chosen in this case is 5%, which is 0.05. If the difference is larger than 5%, the robot needs to turn, and it will turn according to the actual rms value. If rms1 > rms2, which means the left magnitude is larger than the right one, the robot turns left. Same when rms1 < rms2. Also, the logic of controlling go forward or stop is changed here and is easy to see. If the difference is less than 5%, which means the robot does not need to make turns, all he needs to do is moving forward or stop, and the logic in part 3 is used here. If the rmsRelative is less than 5% and avgmax is greater than 0.015, the robot moves, otherwise it stops.

Part 5.

This part needs to combine part 3 and part 4 to let the robot find his way to the speaker. As described in part 4, combining all the logic together, the robot is good to go. However, there is some problem with the original code provided, which is motor speed. Since the analog circuit designed is not so reliable, the robot actually makes the wrong decisions. The problem is that the robot goes too fast that he missed the speaker and go elsewhere, so the mission for this part is to find appropriate values for motor speeds. After many trials, slowing down the forward speed and reduced the time motor runs (lines 367-368), and the speed and time for moving to

the right or left are good options. In this way, there is enough room for the robot to correct himself when making mistakes and finally reached the speaker.

Part 6.

This part is the most challenging part of this project and perhaps the most frustrating part. No matter how hard I tried, the microphone cannot find if the robot is moving away from the speaker. The logic is that, since the robot does not have eyes, the just goes where he wants. But as he moves forward and finds the magnitude of sound is becoming smaller, he should realize that he is moving in the wrong direction. The codes are from line 329 to line 347. The idea is to record the three most recent values of avgmax, and store them into arrays small1[] and small2[]. After the array has three elements, a comparison is made between the three elements. If [2] < [1] < [0] for left (small1[]) or right side (small2[]), which means the magnitude of sound is decreasing after every move of each channel, the robot should turn 180 degrees. However, for some reason, it is impossible for the microphone to pick up the decrease in volume. After testing the robot on a table and apply three levels of sound to it, in descending order, it turns 180 degrees. But on the ground, when the robot is moving further and further from the speaker, the microphone just does not give the wanted magnitude. Also, many efforts have been made such as increasing the distance for each move of the robot so that it can move even further from the speaker, change rms values to avgmax values, apply proper sinewave to assure the uniform sound loudness, but they just do not work. After spending 10 hours or more on it, I concluded that the analog circuit and the microphone are not capable of doing such challenge. In all, this part is challenging in a way that is out of my expectation and also my capability.

IV. Resources and Acknowledgements

EEC10_S20_Final_Project.pdf

https://en.wikipedia.org/wiki/High-pass_filter

https://en.wikipedia.org/wiki/Low-pass_filter

https://canvas.ucdavis.edu/courses/438487/external_tools/5280

I would like to thank the lab instructions provided by instructors, as wells as Timothy Ambrose's introduction video for the final project. Also, I would like to extend my sincere thanks to Xinyang Cao for giving me guidance to check off the lab. In addition, I greatly appreciate Brandon Truong for helping me solve the problems I met during the project.