Anika Rede and Hua Chan Zhuo
EE16B Lab
Prof Anant Sahai
12 Dec 2019

Voice-Controlled Car Lab Report

Front-End Circuit

In building the front-end circuit, we first needed to reduce the voltage (to avoid burning electronic components) from 9V to 5V and 3.3V using voltage regulators in order to power the mic-board and the breadboard rail respectively. These voltage regulators' functions in the whole front-end circuit can be seen in **Figure 1** and their internal structures in **Figure 2**. We then built the microphone biasing circuit (**Figure 3**) to provide signals to OS1 and OS2 on the mic-board. The OS1 pin set the amplitude of the signal (DC offset) and the OS2 pin set the level shift. To keep the mic-board signal reasonable, OS1 was determined to be 1.65V, halfway between the 3.3V rail and ground, since we need a usable voltage. We achieved this using a simple voltage divider (two equal resistors connecting 3.3V to ground) with a buffer at the end since the circuit will need to feed into OS1 and OS2. The mic-board contains a non-inverting amplifier with the potentiometer to amplify the microphone's signal using OS2 as a reference voltage (V_{ref}) (**Figure 4**). In order to not have the output voltage (V_{out}) be amplified excessively by the DC offset, the V_{ref} must be centered at 1.65V so that the voltage differences ($V_{in} - V_{ref}$ and $V_{out} - V_{ref}$) remain equal. **Equation 1** shows why these differences affect the gain, so we must keep V_{ref} relative to OS1. Another buffer is added opposite of OS1 and that output voltage is sent to OS2. The resulting biasing circuit is shown in **Figure 5**. These buffers improve accuracy and correct for any uncontrollable inconsistencies within the resistors themselves. Since the human vocal range varies from around 250Hz to 2.5kHz, we created a high-pass filter using a capacitor (0.1μF) and resistor (5.1kΩ) that had a cutoff frequency of 312Hz. The input was the mic-board output to remove any noise below the human vocal range read into the Launchpad. The filter used a V_{ref} of OS2 to keep the signal centered at 1.65V and make sure the DC offset remained in effect. A buffer was added after the high-pass filter with 3.3V and ground references as a preventative measure to avoid shorting the Launchpad. After the front-end circuit, we built the left and right motor drivers for the wheels.

For all the buffers, the gains are 1 in the biasing circuit and high-pass filter. For the non-inverting amplifier within the mic-board itself, Equation 1 shows the gain as a ratio of a V^*_{out} and V^*_{in} where V_{out} and V_{in} are relative to V_{ref}. The cutoff frequency for the high-pass filter was determined using **Equation 2a**. The approximate frequency response of the high-pass filter at the cutoff frequency is shown in **Equation 2b** so when we directly provided V_{in} and the cutoff

frequency, we measured the frequency response and an appropriate result according to Equation 2b showed that we built the circuit correctly.

For our project, we noticed slightly off signals from the mic-board, not completely smooth signals (*i.e.* small steps along the whole signal) coming from the oscilloscope when reading the output of the mic-board. We kept that in mind for future parts of the lab, though in the end this did not play a significant factor in any bugs.

Figure 1: Front-End Circuit Schematic on Breadboard

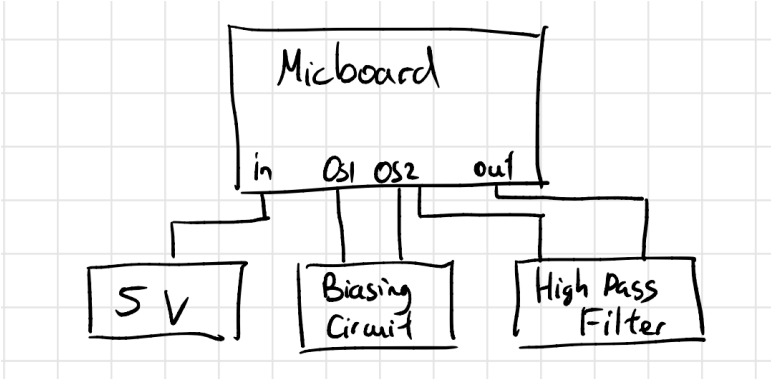


Figure 2: Voltage Regulators Internal Structures

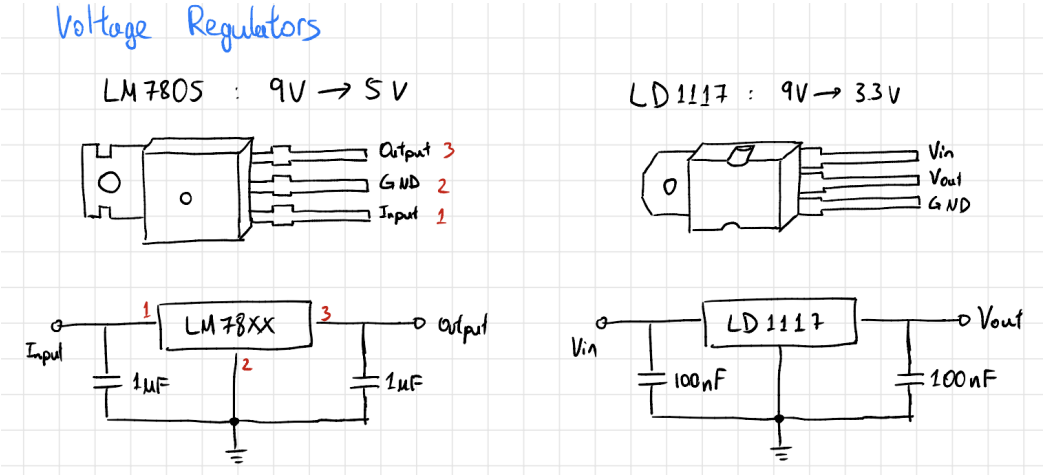


Figure 3: Internal Mic-Board Circuit

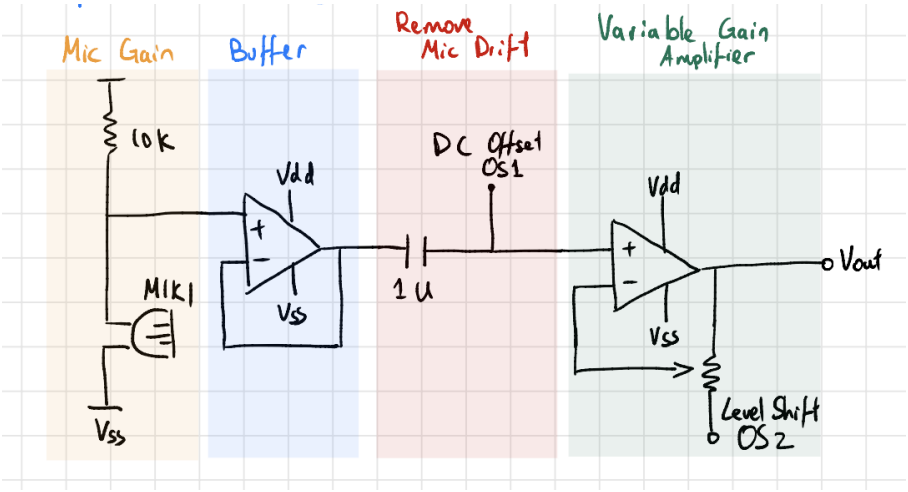


Figure 4: High-Pass Filter

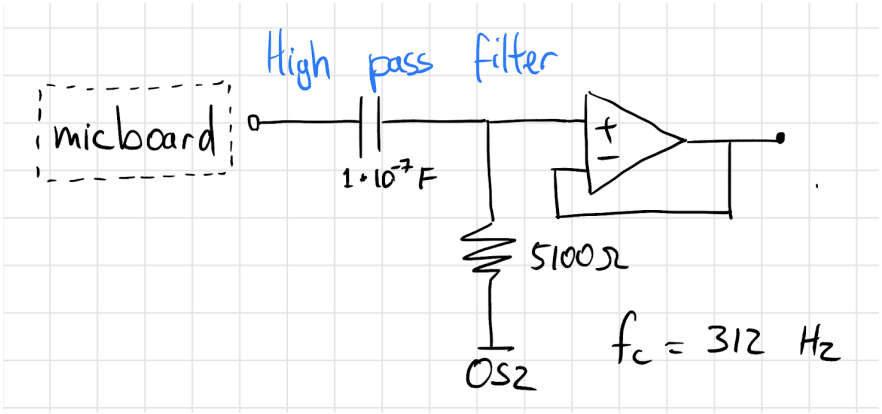
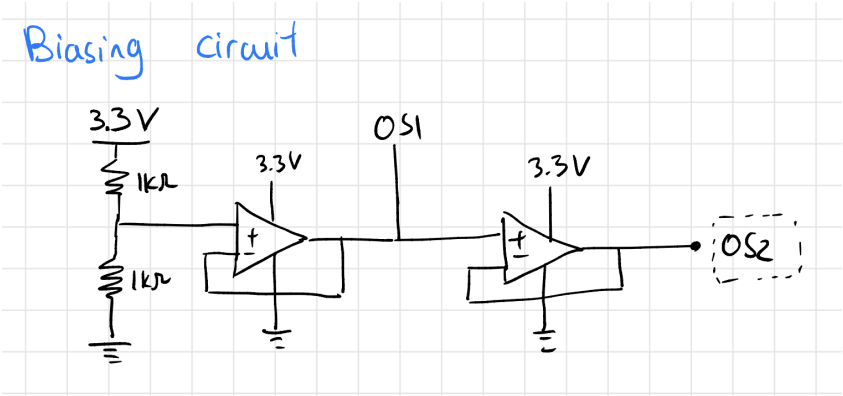
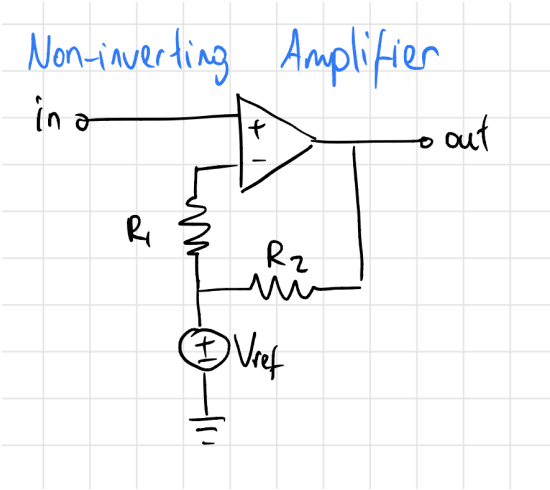


Figure 5: Biasing Circuit



Equation 1: Non-Inverting Amplifier in Mic-Board



(a) By the first golden rule, $V_A = V_{in}$

(b) $\frac{V_{out} - V_{in}}{R_2} = \frac{V_{in} - V_{ref}}{R_1}$ >> substitute $V^*_{in} = V_{in} - V_{ref}$, $V^*_{out} = V_{out} - V_{ref}$

(c) $\frac{V^*_{out} - (V_{in} - V_{ref})}{R_2} = \frac{V^*_{in}}{R_1}$

(d) $\frac{V^*_{out} - V^*_{in}}{R_2} = \frac{V^*_{in}}{R_1}$

(e) $\frac{V^*_{out}}{R_2} = V^*_{in}\left(\frac{1}{R_1} + \frac{1}{R_2}\right)$

(f) $\frac{V^*_{out}}{V^*_{in}} = 1 + \frac{R_2}{R_1}$ >> thus ratio of V^*_{out} and V^*_{in} important

Equation 2: Cutoff Frequency and Frequency Response in High-Pass Filter

(a) $f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi(5100)(1*10^{-7})} \approx 312.068Hz$

(b) $V_{out} = \frac{V_{in}}{\sqrt{2}}$

System ID

In System ID, we installed the encoders and began testing the determined linear-system model of the left and right wheels according to **Equation 3a** and **3b**. System ID included open-loop data collection to find the aforementioned θ and β values for each wheel. Using least-squares linear regression, we determined θ and β based on the above equations with the A matrix composed of (u[n] - 1) values on each wheel and **y** vector of v[n] values. Thus, least squares returns $\mathbf{x} = (A^TA)^{-1}A^T\mathbf{y}$ when **x** is composed of θ and β (**Equation 4**). To get the car to drive straight for controls, the next part of the project, both wheels must move at the same velocity. To determine the motor's velocity ranges, the operating point of velocity (v*) which both wheels could operate at was set at the midpoint of the overlapping range of velocity from this open-loop data. However, our car had trouble with the fine data set we had and the range was not sufficient for the next labs. A GSI suggested we use the coarse data and a larger range for PWM and see whether that worked well for the rest of the lab, and this solution worked out fine for us. Thus we made the range of PWM 120 to 180 with a step of 10 and got a velocity range of 33.5 to 73.6. We found the midpoint and determined v* *i.e.* the operating point as 53.55.

Equation 3: Linear Model of Wheels

(a) $v_L[n] = \frac{d_L[n+1] - d_L[n]}{n+1-n} = d_L[n+1] - d_L[n] = \theta_L u_L[n] - \beta_L$

(b) $v_R[n] = \frac{d_R[n+1] - d_R[n]}{n+1-n} = d_R[n+1] - d_R[n] = \theta_R u_R[n] - \beta_R$

where v[n] = velocity at time-step n, d[n] = distance at time-step n,
u[n] = input to system at time-step n, θ and β determined empirically

Equation 4: Theta, Beta, and Operating-Point Velocity Values

$\theta_L = 0.5468$ $\theta_R = 0.632$

$\beta_L = 21.18$ $\beta_R = 52.77$

Velocity range = [33.5, 73.6]

v* = midpoint of velocity range = (73.6 + 33.5) / 2 = 53.55

Controls

Closed-Loop

The closed-loop model is required to correct for any issues with right and left wheels/motors, uneven load on the car, or possible circuit loading. In open-loop dynamics, our car made a few random turns before going straight, and it is necessary to find delta values of differences between left and right wheels to try and optimize the car's motion (minimizing delta). By equating the open-loop and closed-loop models, we can use the θ, β, and v^* values we derived in open-loop to make the car drive straight accordingly by estimating k-values. As seen in **Equation 5**, $\delta[n]$ values play an important factor in the user inputs of $u_L[n]$ and $u_R[n]$. In **Equation 5f**, we find the steady state of δ by finding $\delta[n+1]$ in terms of $\delta[n]$. The coefficient $(1 - k_L - k_R)$ evolves the state variable each time step and represents our one-dimensional closed-loop matrix of A+BK.

Equation 5: Closed-Loop Model

(a) $\delta[n] = d_L[n] - d_R[n]$

(b) $v_L[n] = \frac{d_L[n+1] - d_L[n]}{n+1-n} = d_L[n+1] - d_L[n] = v^* - k_L\delta[n]$

(c) $v_R[n] = \frac{d_R[n+1] - d_R[n]}{n+1-n} = d_R[n+1] - d_R[n] = v^* - k_R\delta[n]$

(d) $u_L[n] = \frac{v^* + \beta_L}{\theta_L} - k_L\frac{\delta[n]}{\theta_L}$ << set equations 5b = 3a, find $u_L[n]$

(e) $u_R[n] = \frac{v^* + \beta_R}{\theta_R} + k_R\frac{\delta[n]}{\theta_R}$ << set equations 5c = 3b, find $u_R[n]$

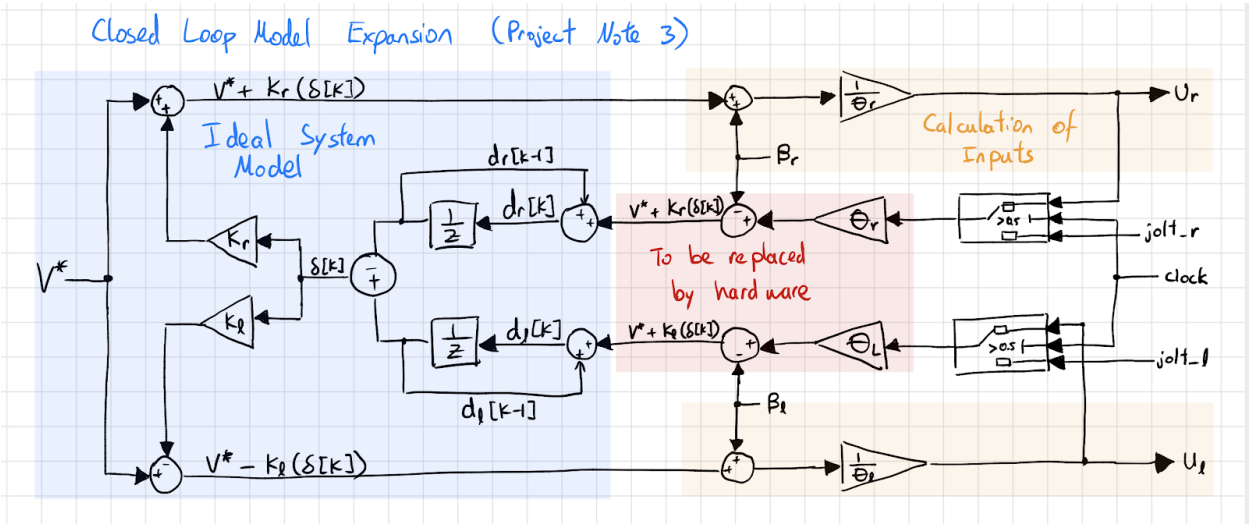
(f) $\delta[n+1] = d_L[n+1] - d_R[n+1]$

(i) $\delta[n+1] = v^* - k_L\delta[n] + d_L[n] - (v^* + k_R\delta[n] + d_R[n])$

(ii) $\delta[n+1] = v^* - k_L\delta[n] + d_L[n] - v^* - k_R\delta[n] - d_R[n]$

(iii) $\delta[n+1] = d_L[n] + d_R[n] - k_L\delta[n] - k_R\delta[n]$

(iv) $\delta[n+1] = \delta[n](1 - k_L - k_R)$



Controller Values

We chose k values initially as 0.5 and 0.5 in view of Equation 5f shown above. The coefficient of $(1 - k_L - k_R)$ to 0 to set the next time-step as 0 would have been ideal. However, we changed the k-values according to how our car turned irregularly. Our car first jerked right, then turned largely to the left, and finally continued straight. Since the car took a long time to converge to a straight line, we knew k values had to be adjusted *up*, and through trial-and-error we figured out the best k_L as 0.75 and k_R as 0.6. Given Equation 5f, we can treat $\delta[k]$ as the state variable and make sure that the system is stable by having eigenvalue $\left|(1 - k_L - k_R)\right| < 1$, so that $\delta[k]$ goes to zero eventually. In our case $\left|(1 - 0.75 - 0.6)\right| = 0.35 < 1$, which satisfies stability. After this adjustment of k-values, the car drove in a relatively straight line.

Turning

In order to make turns, we realized that we need to use θ values and arc length in representing $\delta[k]$. Thus, we realized that **Equation 6** would show a right turn and its negative would be a left turn. Although the car was not greatly affected by mechanical errors, we decided to implement a straight correction function to keep accuracy of moving straight as high as we could. This function simply used the same Equation 6 but used a turn radius called "straight radius" which equalled infinity (slope of a vertical line).

<u>Equation 6: Turning Right Equation</u>

(a) $\delta[k] = d_L[k] - d_R[k] = (k * \frac{v^*}{m})(\frac{r + len/2}{r}) - (k * \frac{v^*}{m})(\frac{r - len/2}{r})$

(b) $\delta[k] = d_L[k] - d_R[k] = (k * \frac{v^*}{m})\frac{len}{r}$

where len = car's width, r = turn radius, and m = 5

Principal Component Analysis

We used the words “cat” for far, “green bean” for left, “sad bears” for short, and “Charlie” (the name of our car) for right. We chose these words keeping in mind a difference in syllable count and uniqueness of sounds. We had trouble with “sad bears” and “green bean” since both had a pause in-between and similar intonation, causing lower accuracy on these words. We also made the data sets with each of us having two words each. The differences between female and male vocal ranges may have factored into the accuracy issues, but we fixed any underlying issues by adjusting the length of audio snippets (from 80 to 100) and threshold to to find the start of the spoken command (from 0.5 to 0.6). These adjustments made the accuracy reach 100% for each word in the datasets. We first preprocessed the data by dividing signal into samples, finding the mean of each sample, subtracting each sample by their mean, and summing the absolute value of each sample. Then we stacked the training sets, zero-meaned the data, took SVD of the demeaned data set, and extracted the largest principal components. We used 3 principal components from the largest sigma values in the Σ matrix from SVD. We projected the demeaned data set onto this new basis of three PCA vectors, determined the centroid locations, and created a classifier function to project the data points onto these centroids and find argmin to find the closest centroid.

Future Improvements

To improve this lab, we would have liked a little more information on how to decide on vocal commands to use for the car. For example, near the end of the PCA lab, we found out that we could use Excel to isolate less helpful recordings from our data (would be super helpful to know at the beginning of the lab). However, given that there must be a way to check those signals from each word against each other *prior* to the lengthy PCA process to possibly see if the signals are different enough. By the end of the lab, we just fudged around the length and threshold values to pass >80% accuracy on the commands, but we probably should have gone back and used different commands. However, going through the whole PCA classification process seemed too arduous if the commands could be manipulated to work semi-perfectly.

Another suggestion would be to make a lab report due each week but in a shorter form. That may add a little onus on each week but going back and recalling the whole lab at the end of the semester may be a little less useful than going in-depth into the material *while* learning it as a means of reviewing the concepts again. Outside of these two suggestions, one cool improvement would be to make the car literally go backwards since it’s capable of the other basic movements. Finally, we struggled a little with the castor wheel on our car for a while before a GSI just suggested to tape it to the underside of the car, so clarification on that in the beginning of building the car would be helpful even if just to a few people.

General

We learned quite a lot during this voice-activated car project. From the beginning, we realized that even small factors can play a large effect on the car's performance. We did not originally tape the castor wheel to the car, and that caused us some issues because we disagreed on whether it significantly affected the car's functionality. Consulting with a GSI, he suggested to just tape it. Otherwise, the solid wheels were slightly off from straight, but that was corrected for with k and delta values. Around building the encoders, we finished the lab so early compared to everyone else that we didn't know there was a lab bug where we needed voltage divider between the encoder and Launchpad, so though we thought our car had run into major issues, we made that simple fix and continued on with the project. In the last lab, we had to change some k values to make straight far and 90 degree left turn consistent with commands. We did not run into huge difficulties or setbacks except the last lab when our car suddenly stopped and started frequently instead of running with the code. The mind-blowingly simple solution of simplifying our Energia code made everything work correctly, so even small coding differences could cause EE troubles!

This car project really helped us apply class material while we were learning them, especially with PCA, centroid clustering, and classification. The visualization graphs with 3D and 2D representations for 3 principal components taken from ~44-dimension vectors solidified the concepts we had learned in a physical experience. This project also helped visualize OMP and least-squares in relation to PCA. We were able to create linear models from wheels and manipulate them using delta values for each time-step. Adjusting k-values to match each wheel's speed to the other and making methods for turning also hit home key physics concepts that are integral to any kind of electric car project. In general, we also learned necessary EE skills for properly debugging circuits using the oscilloscope, multimeter, and function generator all on our own very gradually over the course of the semester.