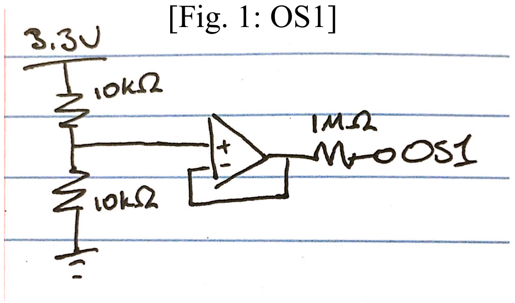
EE 16B Spring 2017
Final Project

SIXT33N Project: Written Report

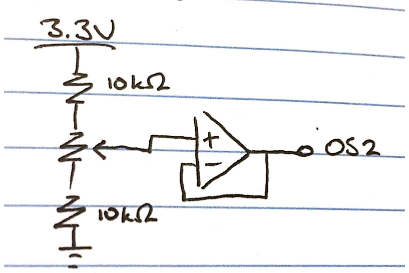
by Edward Doyle: ee16b-aeh
& Jerry Park: ee16b-ami

Circuit:

Final Design:



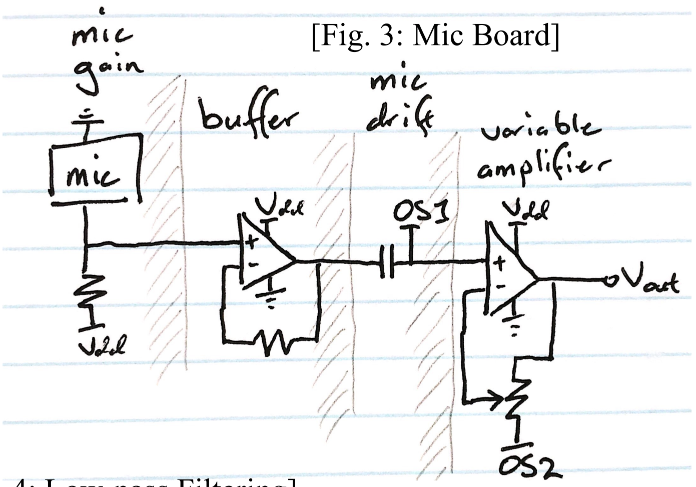
[Fig. 1: OS1]



[Fig. 2: OS2]

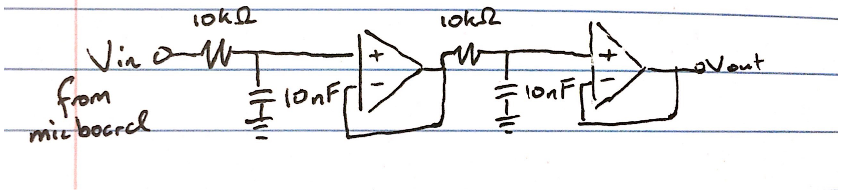
We take process our microphone data in five stages: mic gain, buffer, mic drift, gain, and filtering. First we create voltages OS1 and OS2 to be 1.65V, with OS2 being tunable to correct for any error in the resistors.
In mic gain, the microphone can be thought of as a variable current source so we put the mic in series with the 10 kΩ resistor, then pick off the voltage drop across the resistor.
We then buffer that voltage to prevent disruptions to it. A capacitor in conjunction with OS1 at 1.65V lift up the voltage to take into account the DC offset. Otherwise we'd need negative voltages.

Gain and Frequency Response:



[Fig. 3: Mic Board]

Then we amplify the result, with a variable OS2 to take into account the inaccuracies to our 1.65V offset OS1. The amplification too is variable with the potentiometer. This gives us a signal varying about a center of 1.65V. Then to prevent aliasing due to the launchpad's sampling rate, we combine two buffered low-pass filters for a second-order low-pass filter. The cutoff frequency is $(2\pi RC)^{-1} \approx 1.5$ kHz.

[Fig. 4: Low-pass Filtering]


PCA Classification:

Commands:

Words that have distinct nuances or words that have different number of syllables worked particularly well. For such reason, we used the following commands: Trump, Bernie, Bubble Tea, and AHHHHHHH. When we tried using the word Hot Pot, but it was too similar to Bernie and Bubble Tea.

Processing:

To distinguish between voice commands, PCA was the main part of the processing. After zero-meaning and finding each principal component, we used k-means clustering to classify each input sound to a particular command. To make it more robust, we had a k-means threshold, which did not process the voice command if the distance to the closest centroid was greater than this threshold, and loudness threshold in the code to reduce misclassification and background noise.

Controls:

Open Loop Model:

Through the encoder, we can detect the distance the car moved with each wheel at each time step k noted as $d_{L,R}[k]$. With open loop, we can model the system with the following form where u[k] denotes the PWM user input, but with left and right wheels separately because the differences in hardware (i.e. motor power, friction):

$$d_L[k+1] - d_L[k] = \theta_L u_L[k] - \beta_L$$
$$d_R[k+1] - d_R[k] = \theta_R u_R[k] - \beta_R$$

We first run the car with a range of values for both u's and with the distances we obtain, we can use least squares regression to estimate the value of θ, β. Once we have the four values, we can set the PWM u_L and u_R to values that can set the model velocity to any desired value. To move straight, we can manipulate u_L and u_R so that

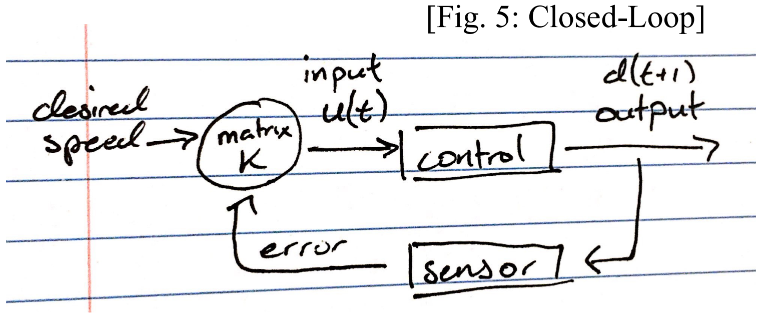
$$d_L[k+1] - d_L[k] = d_R[k+1] - d_R[k]$$

Closed Loop Model:

In order for the car to move straight, we need the closed loop model as the user cannot provide direct input values when driving, so the car has to be programmed to modify its PWM values itself to move straight through a closed feedback loop. This is better than open-loop model, as with open-loop errors in initial measurements cascade, while in closed-loop they are corrected for irregardless.

$d_L[k+1]=d_L[k]+\theta_L(k_1 d_L[k] + k_2\, e[k])-\beta_L$

$d_R[k+1]=d_R[k]+\theta_R(k_1 d_R[k] + k_2\, e[k])-\beta_R$



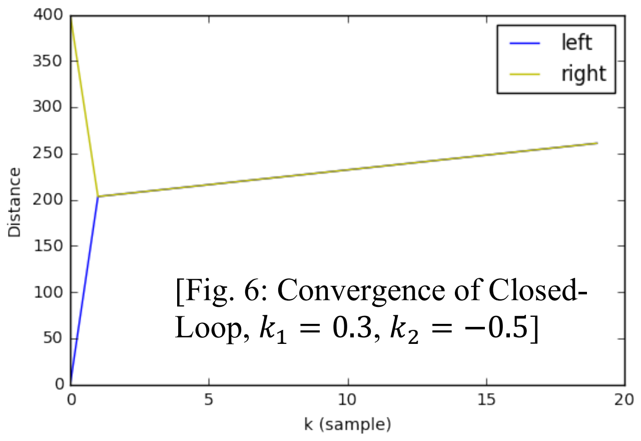
[Fig. 5: Closed-Loop]

Choosing Controller Values:

We chose $k_1 = 0.3$ and $k_2 = -0.5$ because not only does it satisfy the inequality needed for convergence but also these particular values converge without any oscillation. For turning, we directly modified the PWM values and modified them through trial and error. For example, we decreased the left wheel's velocity while keeping the right wheel's velocity higher in order to turn left.

Turning:

We turn by directly modifying the error $d_L[k] - d_R[k]$. By increasing it, closed loop corrects by turning left; decreasing turns right.



[Fig. 6: Convergence of Closed-Loop, $k_1 = 0.3$, $k_2 = -0.5$]

General:

During the course of this project, we learned the importance of taking baby steps, testing after every addition. Filling out a whole block of code or large bits of circuitry at once made it difficult to debug. The o-scope was our savior.