SIXT33N Project Report

By: Eric Gan, Shloak Jain, Arjun Mishra

Circuit

Final Design

Power Supply

We powered SIXT33N with two 9V batteries. One of the batteries powered the two motors, while the voltage from the other battery was regulated to 5V via a voltage regulator. This powered the MSP, which output 3.3V to the rails. We added decoupling capacitors between the rails and the grounds to remove large fluctuations in voltage inputs.

Mic Board

The Mic Board circuit is divided into 4 main sections, the Mic Gain, Buffer, Remove Mic Drift, and Variable Gain Amplifier which serve to make incoming signals easier to process by the rest of the front end circuit. In our circuit, the microphone acts as a variable current source. Since we are assuming our op-amps our ideal, no current flows into them which means that all the current flows through the resistor. Therefore the voltage of the positive terminal is represented by $V_{dd} - V_{ss} - i1000\,\Omega$

Microphone Gain

When the microphone detects external sounds, the signal passes through the mic gain, which is represented by a variable current source as the signal has variable amplitude.

Buffer

This is just a unity buffer we utilize to prevent the op-amp and capacitor from affecting the microphone. No gain occurs in this stage.

Remove Microphone Drift

This described as a coupling capacitor which acts as a high pass filter with an extremely low corner frequency. It removes the low frequency drift of the microphone(noise and DC offset).

Variable Gain Amplifier and Biasing Circuit

The Biasing Circuits OS1 and OS2 are used in this variable gain amplifier stage to better analyze our signal. It consists of a DC offset followed by a level shift. OS1 does the DC offset by centering the signal between 0V and 3.3V. OS2 does the level shift, which serves to transition the reference voltage for the non-intervting op-amps to be 1.65V instead of ground. This way the DC offset does

not get amplified. Lastly, the variable gain amplifier is a non inverting op-amp whose gain can be adjusted by changing its corresponding potentiometer.

Filtering

The signal was filtered through a band-pass filter in order to remove noise from aliasing. This was built using a second order low pass and high pass filter. Together the filters serve as a band pass filter by filtering out noises which are not in the range of voice commands we want. We put buffers between the two because without the buffer, the transfer function we want is no longer of a pure band-pass filter. The values and equations we used to create our filter can be found in the next section.

Gain and Frequency Response

Our corner frequencies were calculated using $f = \frac{1}{2\pi}RC$. Using this model, we found that our low pass filter used a resistor of $5100\,\Omega$ and capacitor of $10^{-9}\,\mathrm{F}$ creating a cut off frequency of $3120\,\mathrm{Hz}$. For the high pass filter, we used a resistor of $2000\,\Omega$ and capacitor of $10^{-8}\,\mathrm{F}$ creating a cut off frequency of $795\,\mathrm{Hz}$. To calculate the frequency response and gain, we used the following equations below. Note that $R_1 + R_2 = 50000\,\Omega$. In addition there is also a gain associated with potentiometer 2, which we can control.

$$H(\omega) = \left(\frac{1}{j\omega R_1 C_1 + 1}\right)\left(\frac{j\omega R_2 C_2}{j\omega R_2 C_2 + 1}\right) \tag{1}$$

$$\mathrm{Gain}_{signal} = 1 + \frac{R_1}{R_2} \tag{2}$$

PCA Classification

Commands

The commands we used were tac (straight far), flamingo (left turn), inspire (straight short), and crescendo (right turn). The words that worked well each had a unique syllabic pattern in which words had different numbers and stresses of syllables. Moreover, saying the word consistently with a chosen emphasis helped tremendously. In our case, we made sure to sharply emphasize each syllable of crescendo and not emphasize any syllable of flamingo to accentuate the difference between the two. Earlier, we had the words Maharbiz and Jaijeet, but we ended up removing them because they were too similar to flamingo and inspire and were thus being classified incorrectly.

Processing

There was preprocessing of data needed to make PCA and K-Means work accurately. After recording samples of each of our words, we looked at the time domain signals to define a threshold amplitude and a snippet length to narrow into the chunk of the signal that actually corresponded to saying the word. Through this, we were able to systematically remove outliers from our recordings by seeing where the filtered signal was significantly higher or lower than average. We then took the PCA and used just the first two principal component to do K-Means classification.

Controls

Open Loop Model

The open loop model fed PWM, $u[t]$, to the motors based solely on empirically measured motor parameters, θ and β, as well as a target velocity, v^*. The open loop controls can be modeled by the following equations below.

$$v[k] = d[k+1] - d[k] = \theta u[k] - \beta \tag{3}$$

$$u[k] = \frac{v^* + \beta}{\theta} \tag{4}$$

$$v^*[k+1] = d[k+1] - d[k] \tag{5}$$

Closed Loop Model

On the other hand, the closed loop model took into account a delta term, which measured the difference between the distance traveled by the left and right wheels and used that to accordingly adjust the PWM. The closed loop model was necessary because there was error between the model and the actual behavior, attributed to things like motor inconsistencies, wheel tread, weight imbalance, and incorrect parameters. With closed loop control, the car used feedback to dynamically correct for these errors and go straight, whereas with open loop control it strayed off with increasing error at each timestep. The closed loop controls can be modeled by the following equations below.

$$\delta[k] = d_L[k] - d_r[k] \tag{6}$$

$$u_L[t] = \frac{v^* + \beta_L}{\theta_L} - k_L \frac{\delta[k]}{\theta_L} \tag{7}$$

$$u_R[t] = \frac{v^* + \beta_R}{\theta_R} + k_R \frac{\delta[k]}{\theta_R} \tag{8}$$

Choosing Controller Values

The delta terms eigenvalue was calculated to be $\lambda = 1 - k_L - k_R$. Since we were in a discrete time setting and wanted a stable model, this eigenvalue needed to be in the range (-1, 1). With these constraints in place, the final k values were chosen through empirical testing and were found to be $k_L = 0.9$ and $k_R = 0.7$.

Turning

To allow turning, the equation for delta was recalculated using the variables l as the distance between the center of the wheels, and r as the turn radius. The closed loop control operated around this new delta, instead of 0 when driving straight, which led to one of the motors being powered more than the other and thus allowing the car to turn. The equation to calculate the delta values for turning can be modeled by the equation below.

$$\delta[k] = \frac{v^* l}{r} k \tag{9}$$

General Comments

Holistically, this project taught us much about the integration not only on a macroscopic level between the software and hardware aspects of engineering, but also on a microscopic level between the various modules of the course. Building SIXT33N from scratch improved upon much of our circuit design and debugging skills and also taught us much about the application of software to both classification and signal processing. In hindsight, design choices played a large part in the difficulty of the project, as many of our errors came from poor and messy builds that we eventually had to clean up. Hardware failures such as broken MSP pins or other parts also proved to be difficult to debug. Lastly, we learned about the importance of solid data collection especially in large integrative projects as even the smallest errors grow exponentially in size.

.

Figures

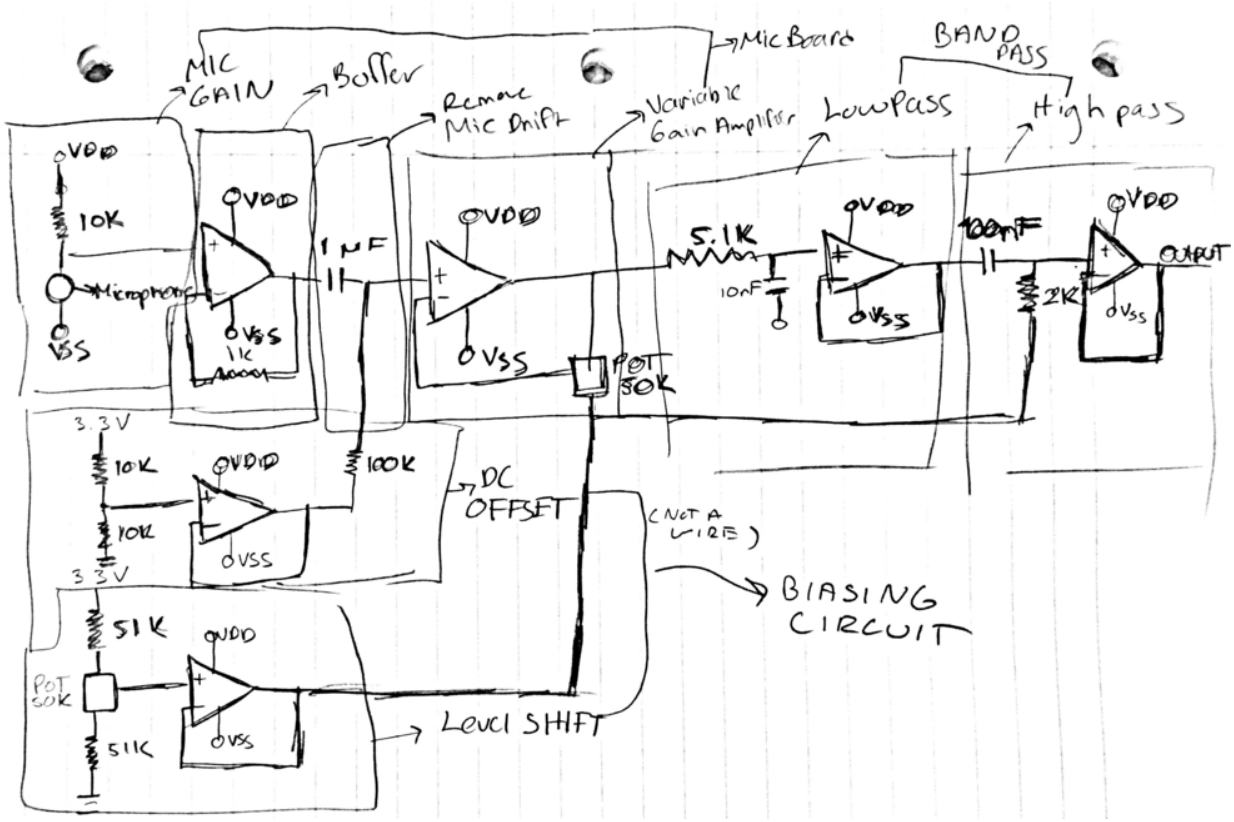

Figure 1: Full Circuit Diagram

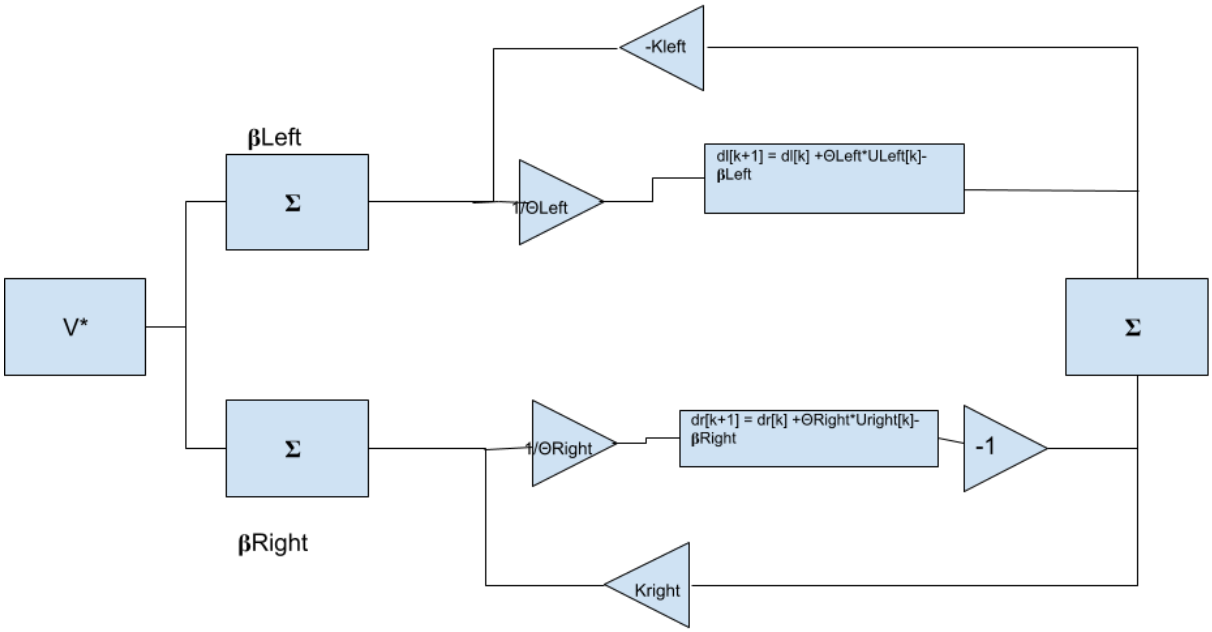

Figure 2: Closed Loop Control Scheme Block Diagram