SIXT33N Final Report

Shadaj Laddad, Thomas Lu

OVERVIEW

In this project, we combined hardware and software to create a voice controlled robot car, learning the procedure on how one might engineer a complex project such as this. On the hardware side, we learned how to apply RC circuits to create low-pass, high-pass, and band-pass filters that allowed us to get clean microphone inputs for voice detection and developed circuits to control motors with PWM signals from the Launchpad microcontroller.

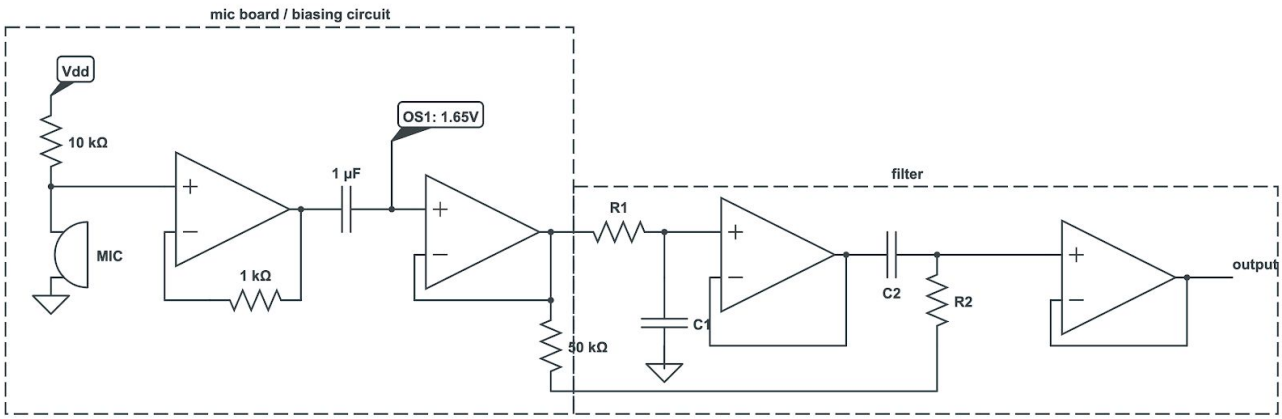
On the software side, we applied the control theory concepts taught in class. Using closed loop control with a system model derived from experiment data (with least-squares), we were able to develop controls that could accurately drive the robot forward and also make turns. In addition, we used the PCA/SVD concepts from class to develop voice recognition capabilities for the robot

COMPONENTS

Front End Circuit

The purpose of SIXT33N's front end circuit was to set up the microphone that allows our car to intake voice commands. Because the front end circuit takes 5 volts, a voltage regulator was used to convert the 9 volt battery into a our desired voltage. The usable range for the microphone's output was set to be between 0 and 3.3 volts, since the Launchpad cannot take negative voltages. In order to achieve this range, a DC offset of 1.65 volts placed the mic signal at the center of this range, and a level shift to 1.65 volts rather than ground was implemented to prevent the DC offset from being amplified along with the signal when sent through the non-inverting amplifier. Then, the mic board's potentiometer was tuned to bring the signal to a desirable amplitude. Ultimately, the microphone's gain is expressed as (k * Vmic + 1.65), Afterwards, the output voltage was filtered with a band-pass filter that removes frequencies outside of about 250 Hz to 2500 Hz, which is the range that contains most of human speech. The band pass filter was created by chaining a low pass filter, which has a transfer function of about 1 / (jw / 2500 + 1), with a high pass filter, which has a transfer function of 250 *jw / (250 * jw + 1).

Front End Circuit Schematic



PCA

In the PCA portion of the project, auditory data was processed in order to allow SIXT33N to recognize voice commands. Our car used the commands “Sahai,” “Loop,” “Ping Pong,” and “Garage Door.” Commands that had indistinct sounds and were similar to other commands did not work as were more likely to be picked up as an incorrect command. Commands such as “Cackle” or “Problem Sets” did not work well with our other phrases in our specific pronunciation, for example. Varying commands by syllables or by the length of the phrase makes the commands more distinguishable. In order to pre-process the data, we chose a threshold volume at which to start the signal, the length of time before the threshold volume is crossed to start processing, and the end length of the signal to process. Performing PCA turned our time-domain signal into vectors that contained most of the variation for our chosen commands. After performing PCA with both 2 and 3 dimensions, we found that the 3rd dimension did not add much variation, so 2 dimensions was sufficient for our purposes.

Controls - Open Loop

With open loop controls, parameters to tune the speed of SIXT33N were discovered to allow the car to reach the velocity V* with both wheels, allowing it to drive straight. Velocity is set to equal d[k + 1] - d[k] for this system for an arbitrary time step. Because the velocity of the car is dependent on the PWM provided to it as an input u, the velocity can be modeled linearly as θ * u - b. The input u can then be discovered through u = (v + b) / θ. Through linear least squares, values for θ and b could be found by varying the input PWM and measuring the resulting velocity.

Controls - Closed Loop

With closed loop, our system is able to take its own output as feedback. In closed loop, a variable δ was defined as $d_L - d_R$. δ is then used as a variable in the velocity equations for each wheel. Then, $u_L = (V^* + b_L - K_L\delta) / \theta_L$ and $u_R = (V^* + b_R + K_R\delta) / \theta_R$. Thus, if the two wheels move at different velocities, they eventually level out. Closed loop controls are necessary because otherwise, a small disturbance may propagate and make SIXT33N move far off-course.

Controls - Controller Values

For our SIXT33N, both of our controller K values were set to 0.65, which was experimentally discovered to be a good controller value for our car. Controller values that weren't large enough caused SIXT33N to require a long time to adjust. Values that were too large caused the error δ to persist in the form of oscillations. Mathematically, in order to be stable, the absolute value of $1 - K_L - K_R$ must be less than 1.

Controls - Turning

Originally, SIXT33N was set to control $\delta = d_L - d_R$ so that both sides of the car would rotate by the same amount. When turning at a fixed radius, we need the difference between the two sides to grow over time, as we expect one side to be spinning at a higher velocity than the other. In order to implement turning controls, we then need to calculate δ based on the robot parameters, expected turning radius, and the current time step (since the expected difference grows over time). Using some simple geometry to calculate how fast the difference will grow when turning at a radius r, we can calculate that we want $\delta[k] = (V^* \times l)/r \times k$. This intuitively makes sense, since if the robot is wider the difference grows faster, and if the radius is smaller the difference grows faster. To make the robot turn, all we need to do is use our existing closed loop code with δ replaced by the new definition.

Closed Loop Block Diagram