ECE167 Sensing and Sensor Technology

Lab0

James Huang

Checkoff TA: Ishan -1/13/25 ~1:40PM

commit d5102f98b4b90153936347a79f45a2df33ff058f

Table of contents:

1. Project Introduction 1
2. System Block Diagram 2
3. Hardware Components 3
4. Software Components 4
5. Testing and Result 5-6
6. Conclusion 7

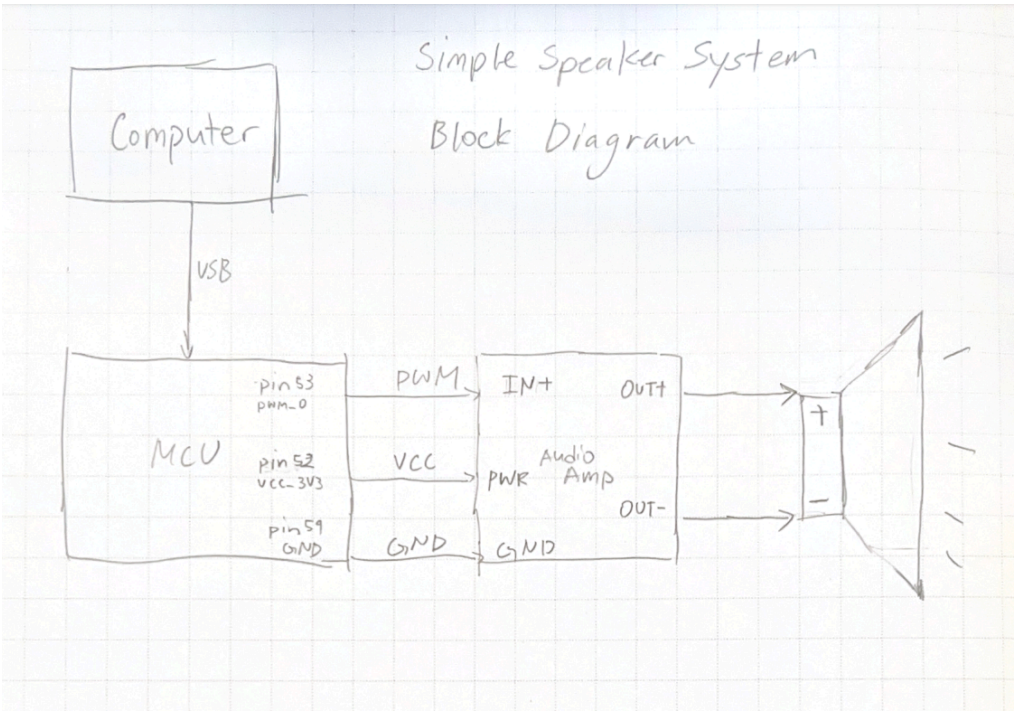
1. Project Introduction

Objectives of this project included setting up the coding environment and simple speaker system. To set up PlatformIO we first download VSCode and add the PlatformIO extension. This allows us to upload code to our MCU. An additional driver is needed from STMicroelectronics in order for the code to be uploaded correctly (https://www.st.com/en/development-tools/stsw-link009.html). Then a simple Hello World program will test our coding environment and MCU. This results in a spam of "Hello World" through serial. The code snippets to set up our Project Configuration File and Hello World program is given in the Lab0 documentation.

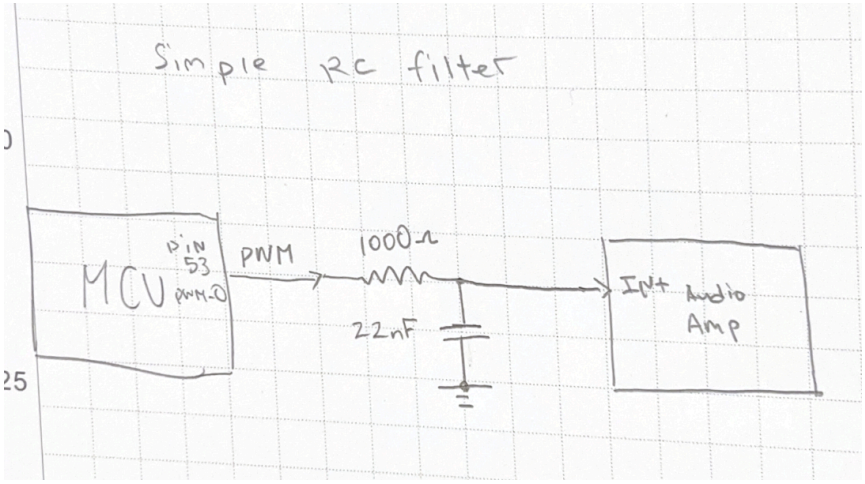
Then a simple speaker system using a SparkFun Mono Audio Amp Breakout - TPA2005D1, and a two pin speaker will be created. With the buttons and potentiometer on our microcontroller, we will be able to change the tone playing through the speakers. An additional RC low pass filter is implemented to smooth out the PWM signal. Along with a software filter to stabilize the flickering raw ADC value. These will improve the sound quality on the speaker.

2. System Block Diagram

The speaker system will be wired in the following block diagram:



To add some hardware filtering, we will implement a simple RC low pass filter.



3. Hardware Components

Microcontroller generates the PWM signal that will be fed into the audio amplifier that will create different tones depending on the signal. `PWM_0` corresponds to pin 53, is the output of the PWM signal that will be connected to the audio amplifier. Pin 52 is a 3.3V VCC that will supply power to the audio amplifier, along with ground pin 59. Lab0 requires a simple two pin speaker which has an + and - pin.

The SparkFun Mono Audio Amp Breakout - TPA2005D1 takes 3.3-5V which can be connected to the microcontroller's power supply along with the ground pin. This audio amp has a set of IN+/- pins that takes in signals, in this case, the PWM signal from the microcontroller goes into the IN+ pin. The set out OUT +/- pins goes to the corresponding +/- pins of our speaker. It also has a set of pins for volume control, but it is not used in this lab.

The RC low pass filter was implemented to smooth out the PWM signal before feeding it into the audio amplifier. This will smooth out the sharp spikes at the edges of the PWM signal and round the sharp edges. To design this filter, we take our max 4000 Hz frequency from the potentiometer and double it to around 8000 Hz as our cutoff frequency. Given the equation $f_c=\frac{1}{2\pi RC}$ we set f_c to 8000 Hz and solve for RC values. Given the resistor and capacitor values in the lab, a 1000Ω resistor and 22nF capacitor is used, which results in a cutoff frequency of 7234 Hz (close enough).

4. Software Components

First task of the lab after setting up the coding environment is to read the ADC value from the potentiometer on the microcontroller. This is done including the `ADC.h` and using the `ADC_Init` function to initialize the potentiometer. Then using the `ADC_Read` function we can get the current ADC reading. This can then be printed and see values change on the serial port in PlatformIO.

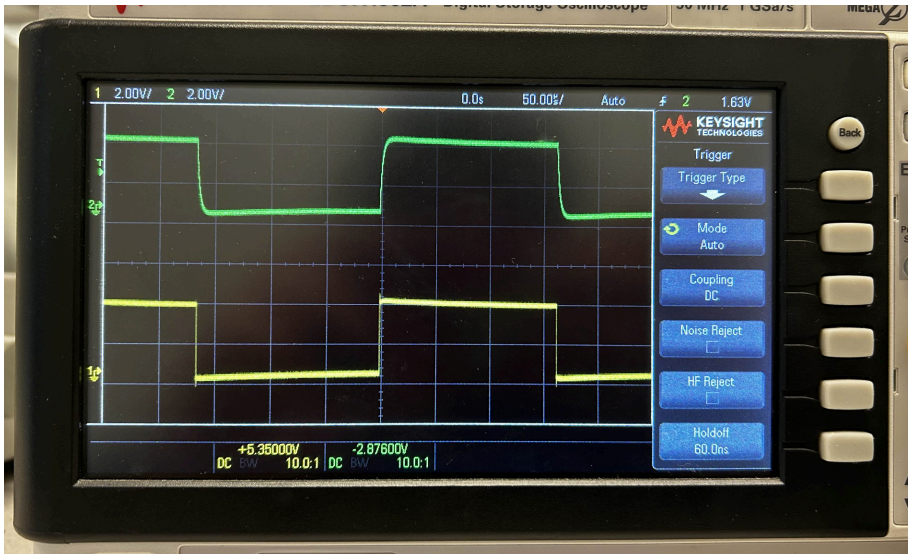
Next to set up PWM we have to include `PWM.h` and initialize using the `PWM_Init` function. First set the duty cycle of the PWM signal to a desired pin using the `PWM_SetDutyCycle` function, this should control the volume of the speaker from values 0-100. Next we set the frequency of the signal to be the current value of the ADC, which can be done using the `PWM_SetFrequency` function. This should result in a change in tone as we turn the potentiometer on our microcontroller.

Lastly, set up the buttons again by including corresponding header and initializing using the init function. To check which button is pressed, we use the `buttons_state` function, which returns the binary value of the buttons, where 1 represents OFF and 0 represents ON. For example, 1111 represents no button is present, and 0111, means the first button is pressed. To check which of the 4 buttons are pressed, we simply check if the function returns either 7, 11, 13, or 14 in decimal, which corresponds to 0111, 1011, 1101, 1110 in binary. Whenever a button is pressed, we can set the duty cycle and the frequency of the signal to whatever we desire.

A software filter is implemented to smooth the flickering potentiometer values when reading the raw ADC values. This can be done with different types of filters, a moving average filter is used due to its easy implementation. A sample size of 10 is used to calculate the average, and by comparing the two outputs, the filtered values are much more stable.

5. Testing and Results

To test the RC low pass filter, an oscilloscope is used to compare the raw PWM signal and the filtered signal. The green signal is the filtered signal, with more rounded and no over spikes at the edges of the signal.



The moving average software filter is used to stabilize the raw ADC value, which flickers really fast. The filter will produce a more stable value for the ADC and produce better sound quality as it controls the frequency of the signal. By printing the raw and filtered values to the serial port, we can see the difference in numbers and how much it fluctuates. As seen below, filtered values are much more stable.

```
filtered pot value: 1770
raw pot value: 1773
filtered pot value: 1770
raw pot value: 1773
filtered pot value: 1770
raw pot value: 1771
filtered pot value: 1770
raw pot value: 1769
filtered pot value: 1770
raw pot value: 1774
filtered pot value: 1770
raw pot value: 1763
filtered pot value: 1770
raw pot value: 1771
filtered pot value: 1771
raw pot value: 1765
filtered pot value: 1770
raw pot value: 1772
```

6. Conclusion

Lab0 was a fun lab to implement, not too difficult. Through this lab, I noticed that the required driver was not included in the Lab0 documentation, which was slightly confusing. IN- pin on the audio amplifier can be connected to ground or left not connected, both ways work. The RC filter smoothed out our rough edged PWM signal, which produced better sound quality. Along with the software filter that reduced the flickering of the ADC value, resulting in better sound quality as well. Duty cycle of the PWM signal controls the volume of the speaker, and the frequency changes the tone. This lab took about 4 hours to set up and complete.