Ayden Dauenhauer and Benjamin Banh

Prof. Wood

ELEN 133L

15 May 2025

Lab 6: Real Time Signal Processing with STM32L476-DISCO Development Board

Step 1:

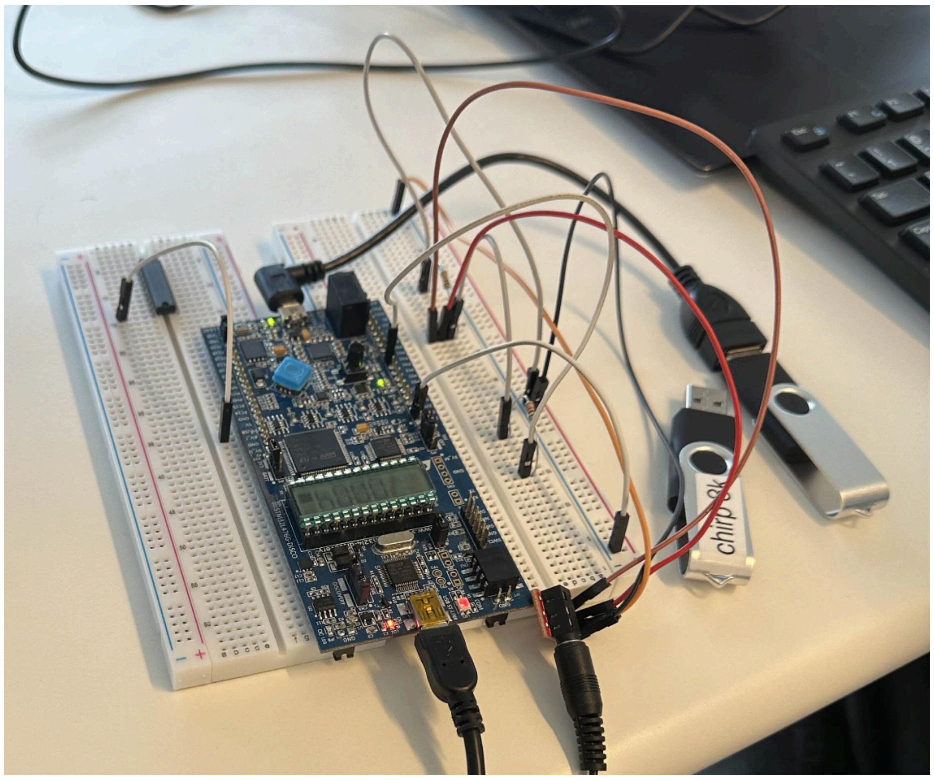

Fig 1: Hardware Set-Up

Step 2:

After setting up the hardware and demo-ing to the TA, we then added new variables to apply filters and sample the audio file from a flash drive. The output is a file read from the flash drive of a frequency sweep from a range of 100Hz to 3900Hz.

Step 3:

The project then continued with the application of togglable filters and adjustable volume according to the joystick built into the STM32 board. The left button would lower the volume, the right button would increase the volume and the up button would enable the filter. The volume control is multiplying or dividing the output with a constant value to show the exponential increase and decrease in volume according to human hearing. The constant of 1.26 was chosen as a 1dB increase in noise level corresponds to a 1.26 increase in sound intensity.

Step 4: High pass filter

In this step, we now implement a simple 2nd order FIR filter through new instructions declared in the filter function. After compiling the code and switching USBs from “freq sweep” to “chirp”, we noticed that the filter acted as a high pass filter which filtered low frequencies and gradually increased in volume as the frequency increases.

Step 5: Low pass filter

The inverse of the previous step occurred here. After filter modifications, the filter acted as a low pass filter attenuating high frequencies starting loud and gradually losing output volume.

Step 6: Band stop filter

The filter was now changed to now incorporate new variables of xold2 and xold3 to store previous sampled values with the output being reliant on both current and previous data. The output of the speaker was attenuated between both the tailends of high and low frequencies, acting as a bandstop filter.

Step 7: Can you hear the effect of the filter on this more complex sound?

No the filter is much harder to notice compared to the simple piano keys.

```
#include "main.h"
#include "coefficients.h"
#include <stdbool.h>

#define XBUFFERLENGTH 512

#define YBUFFERLENGTH 32

extern const float xcoeff[XBUFFERLENGTH];
extern const float ycoeff[YBUFFERLENGTH];

float getSample(void);
void send2OutBuffer(float);

float xnew;
float ynew;
float xold;
float yold;
float xold2;
float xold3;

float outVol;
bool outputEnb;


void leftButtonFunc() {
        outVol = outVol/1.26f;
}


void rightButtonFunc() {
                outVol = outVol * 1.26f;
}

void upButtonFunc() {
        outputEnb = !outputEnb;
}
```

```
//void initFilter(void){
//	/*	Place your initialization code for this module here
//			This code runs once prior to any input, output, or filtering actions
//
//			It is a good place to initialize any data structures you create in this
module  */
//}

//void filter(void) {
//	/*	This is where you place all of your filtering code
//			This function is called 8K times per second prior to outputting any
sound samples to the DAC
//
//			Ordinarily, each time this function is called you would:
//					read one data sample as input
//					perform any filtering or other computation to create an
output value
//					output the calculated value
//	*/
//}

void initFilter(void) {
	xnew = 0.0f;
	ynew = 0.0f;
	outVol = 1.0f;
	outputEnb = true;
	xold = 0.0f;
	yold = 0.0f;
	xold2 = 0.0f;
	xold3 = 0.0f;
}

void filter(void){
	xold= xnew;
	yold = ynew;
	xold3 = xold2;
	xold2 = xold;




	if (outputEnb){
		send2OutBuffer(outVol * ynew);

	}
	else{
```

```
                    send2OutBuffer(xnew);
          }

          xnew = getSample();
          ynew = xnew;
          ynew = 0.5f* xnew + 0.5f* xold3;

}
```