AUDIO EFFECTS GENERATOR

ECE-53800

DIGITAL SIGNAL PROCESSING PROJECT REPORT

AMIT SINGHA

SUBMITTED TO: Mohamed El-Sharkawy, Professor, ECE, IUPUI

DSP Project Report
Audio Effect Generation

Abstract:
The goal of this project was to generate audio effects. We used audio as input and processed it through our FRDM-K64 board to create different audio effects by altering the input audio. Some of the audio effects successfully implemented in this project include ECHO, REVERB, SLAPBACK, PANNING, and CHORUS. The first three are delay-based audio effects, the fourth is spectral, and the final is modulation-based audio effects. The FRDM-K64F board, the Wolfson Pi Audio Card, Keil MDK-ARM and MATLAB Simulink were used to create these audio effects. These audio effects can be used for a variety of purposes, including music recording and mixing, voice changers, fun sound effects and noises, bass boosters, and equalizers. In the not-too-distant future, we may be able to make the system wireless and based on the internet of things.

Introduction:

Audio Effect: In the realm of audio, "audio effects" refer to hardware and software that alter the audio content of a signal. Audio effects, also known as audio signal processing, are the electronic modifications of audible sounds. Changing the form or characteristics of a sound electronically. Several parameters, such as rate, feedback, and gain, can be changed to achieve the desired results. Their use benefits both live performances and studio recordings and mixing. Audio effects are utilized by producers of musical works for the following five primary reasons: sound design, increasing depth, better fitting a sound in your mix, and so on, Increasing the groove and rhythm, as well as expanding your stereo field

Basic Block Diagram

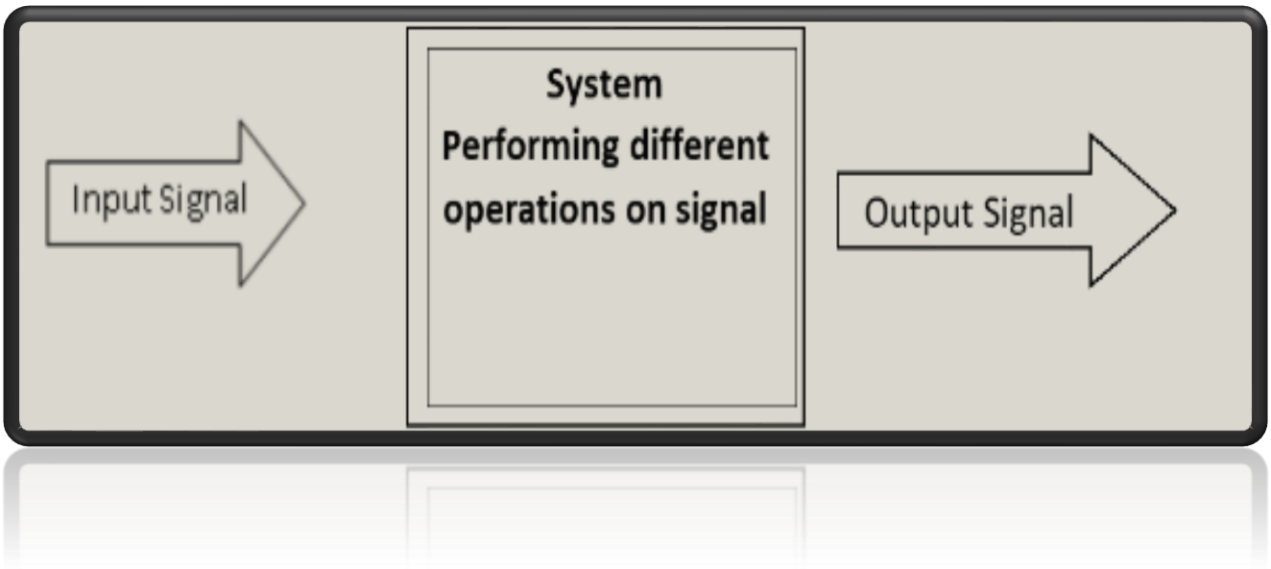


Figure 1: Basic Block Diagram for getting Audio Effect

In this case, audio will be used as input. The FRDM-K64F board will serve as the system, performing various signal processing operations on audio signals, altering and producing various audio effects.

Types of Audio Effects

There are five primary types of Audio effects:

1. Modulation effects—Chorus, Tremolo, Flanger and Phaser
2. Time-based effects—Reverb, Delay and Echo
3. Spectral effects—EQ and Panning
4. Dynamic effects—Compression and Distortion
5. Filters

In our audio effect generator project, we implemented Modulation effects such as Chorus and Tremolo, Time-based effects such as Reverb, Slapback, and Echo, and Spectral effects s as Panning. Below is a brief description of the audio effects that have been implemented.

ECHO:

It is the most well-known audio effect. The echo is a time-based audio effect produced by the Delay audio effect. When the Delay audio effect includes playback heads or "taps" that are heard apart, an echoing effect is simulated at various intervals. Echo effects are audio effects that occur when a signal is delayed in time. It is essentially a reflection of sound that arrives at the listener after the direct sound with a delay.

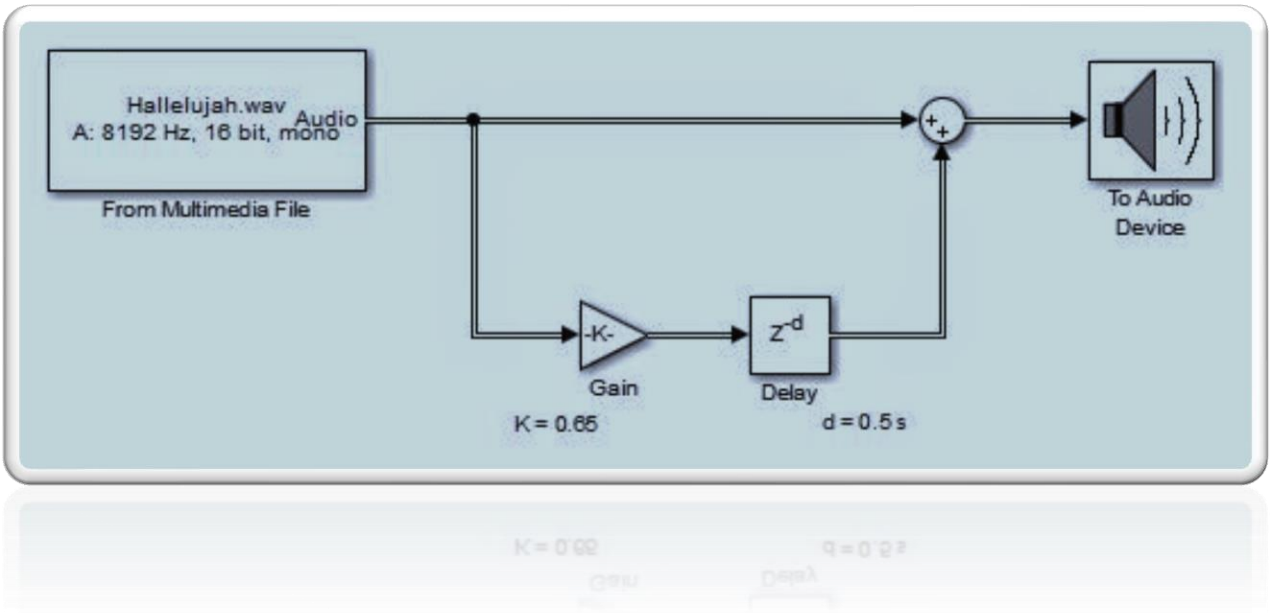


Figure 2: Simulink Representation of ECHO audio effect

Here, Tunable Parameters are Delay and Gain

REVERB:

The spreading of sound waves caused by an impact in a room or other enclosed area is referred to as reverberation. Reverb not only adds depth and width to your mix, but it also gives the listener a sense of the physical location of the sound source and their own position in relation to it. The term "reverberation" is abbreviated to "reverb." It's a common occurrence in our lives, but we rarely think about it. The reverberation found in smaller, more enclosed spaces like tunnels and caves sounds quite different from that found in large indoor spaces like concert halls and cathedrals. Reverb audio effects can be created digitally by using reverb plugins and algorithms to generate multiple echoes that can be tweaked for Delay, volume, and frequency response.

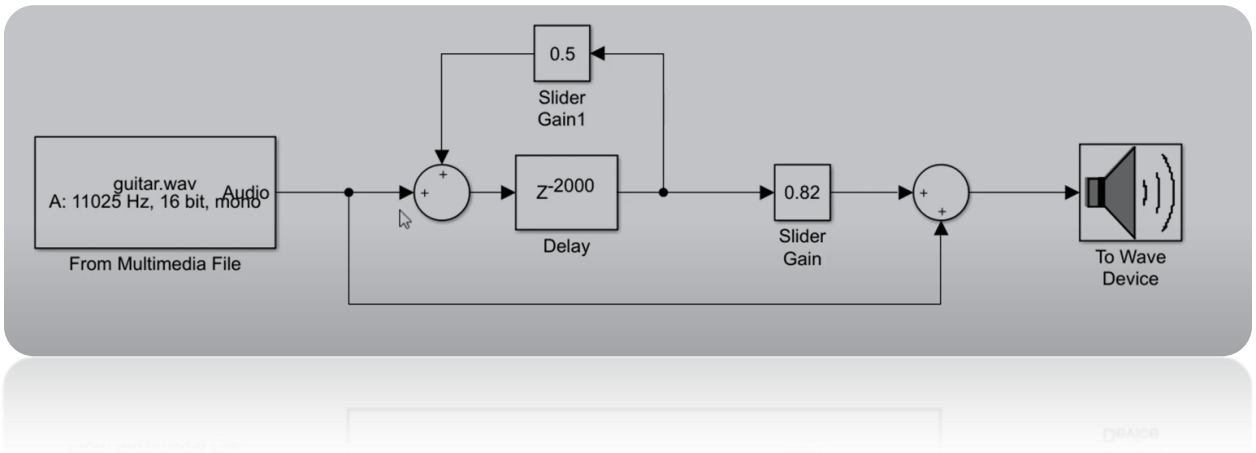


Figure 3: Simulink Representation of REVERB audio effect

SLAPBACK:

The opening sound of Netflix's movie streaming service is an excellent example of the Slapback audio effect. It is also a delay-based audio effect. The key feature of a Slapback is that the delay only occurs once. If you play a note, it will be repeated once. This differs from a standard delay, which will repeat several times before gradually fading away. A Slapback is a clear repeat. A single-repeat echo effect that is commonly heard in 1950s pop and rockabilly styles. Slapback delay is commonly used on electric guitar, but it can also be used on drums, vocals, and other instruments. Delay times are typically in the 40-120 millisecond range.

TREMOLO:

Tremolo is an audio effect that uses modulation. Tremolo changes the volume of an incoming signal by modulating its amplitude. Tremolo is a rhythmic modulation effect that changes the volume of your signal. Tremolo is a popular effect in vintage guitar combo amps, where it is sometimes mistaken for vibrato. The tremolo audio effect modulates the amplitude of an audio signal using a low-frequency oscillator.

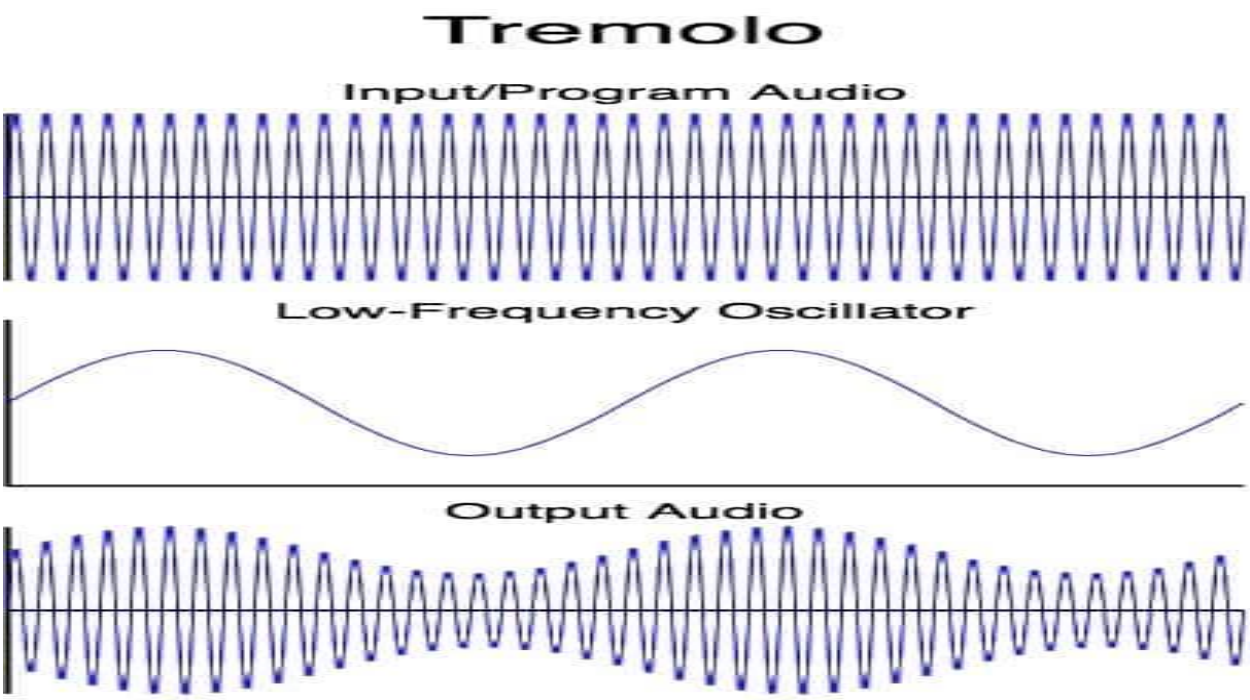


Figure 4: Tremolo Audio Effects

PANNING:

Panning is a Spectral effect. Panning is the process of distributing an audio signal (monaural or stereophonic pairs) into a new stereo or multi-channel sound field determined by a pan control setting. In a stereo clip, for example, you can put more sound in the right channel and less sound in the center channel. Panning produces a variety of spatial effects by allowing more or less of a signal into each speaker.

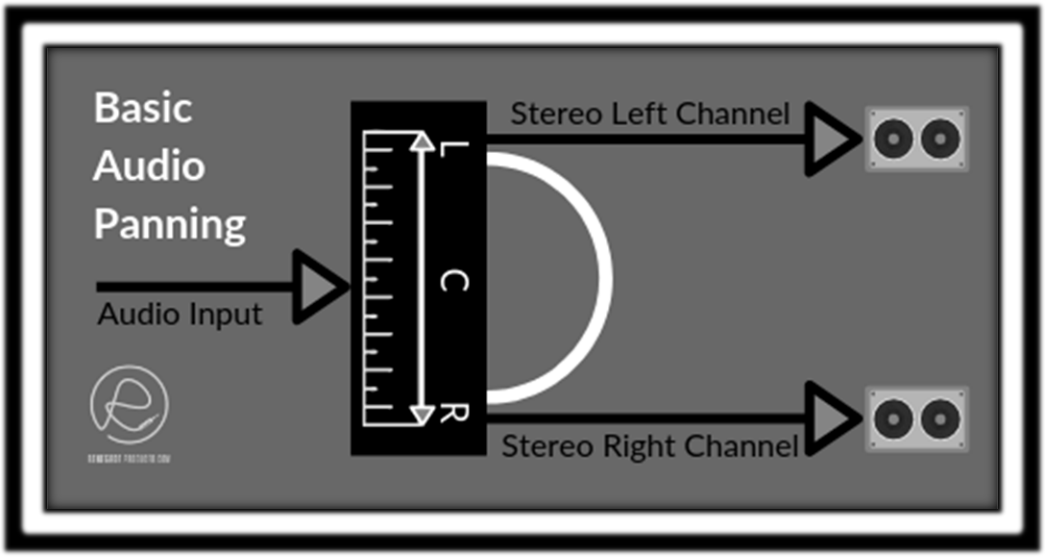


Figure 5: PANNING

CHORUS:

Chorus is produced when sounds that are similar to one another but have slight variations in tuning and timing overlap and are heard as a single unit. It is a natural occurrence that happens when multiple sources produce the same sound. Imagine a real-life choir that has to sing multiple parts simultaneously. They are able to produce a single, recognizable tone when combined. The same effect can be achieved by using the chorus effect!

HARDWARE

This project is implemented using a FREESCALE FRDM-K64F, a Wolfson Pi Audio Card, a PC running Keil MDK-ARM and MATLAB, and connecting cables.

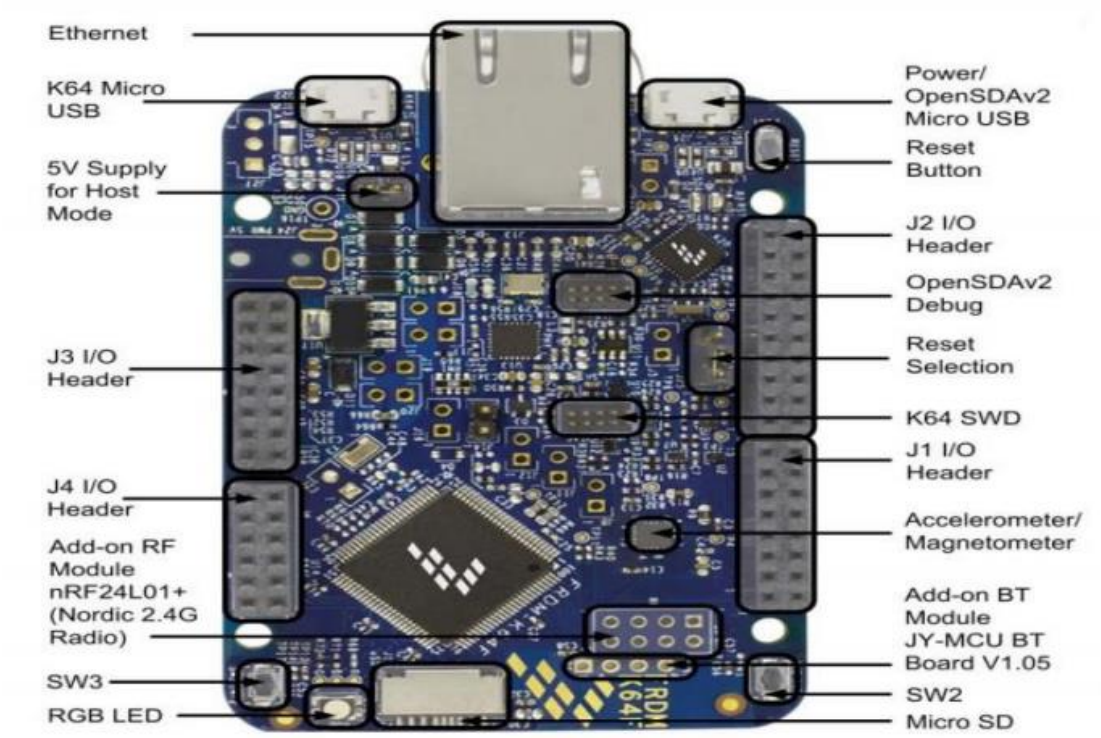


Figure 6: FRDM-K64F components

The Freescale FRDM-K64F is a low-cost development platform featuring a 120 MHz an ARM Cortex-M4 based processor. It connects to a host PC via a USB cable using a CMSIS programming and debugging tool. The Keil MDK-ARM development environment, running on the host PC enables software written in C to be compiled, linked, and downloaded to run on the FRDM-K64F.

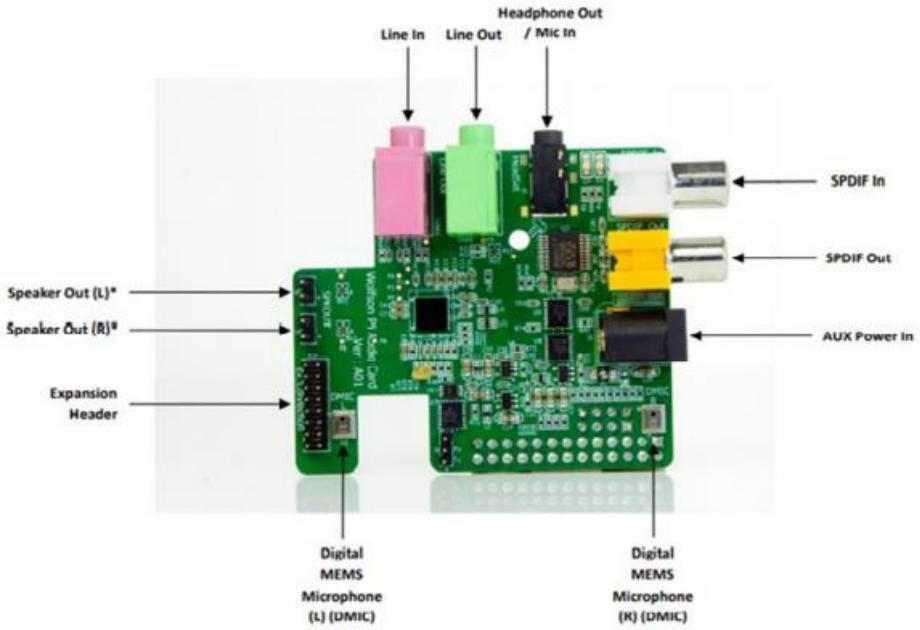
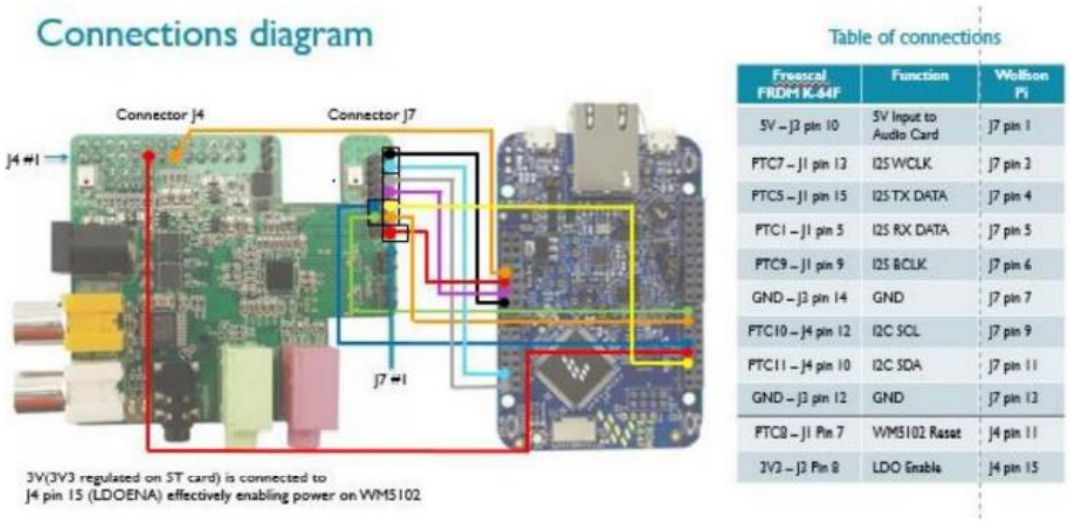


Figure 7: Wolfson Pi Audio Card



Freescal FRDM K-64F	Function	Wolfson Pi
5V – J3 pin 10	5V input to Audio Card	J7 pin 1
PTC7 – J1 pin 13	I2S WCLK	J7 pin 3
PTC5 – J1 pin 15	I2S TX DATA	J7 pin 4
PTC1 – J1 pin 5	I2S RX DATA	J7 pin 5
PTC9 – J1 pin 9	I2S BCLK	J7 pin 6
GND – J3 pin 14	GND	J7 pin 7
PTC10 – J4 pin 12	I2C SCL	J7 pin 9
PTC11 – J4 pin 10	I2C SDA	J7 pin 11
GND – J3 pin 12	GND	J7 pin 13
PTC8 – J1 Pin 7	WM5102 Reset	J4 pin 11
3V3 – J3 Pin 8	LDO Enable	J4 pin 15

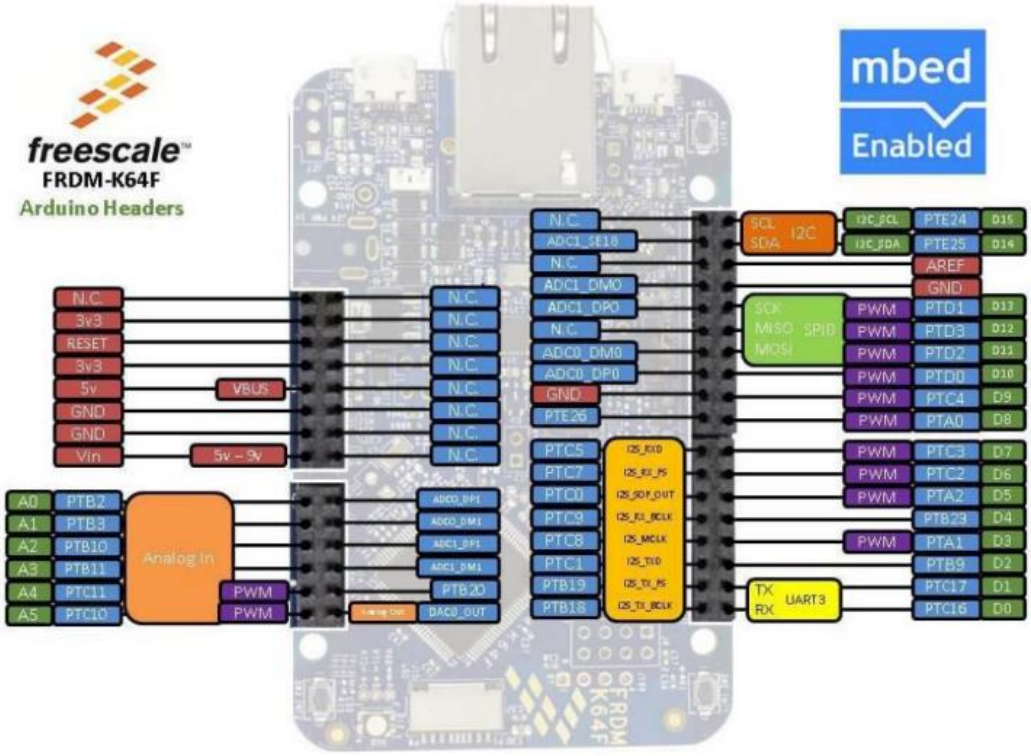
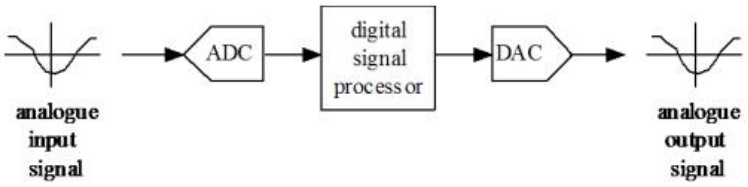


Figure 8: Pin Diagram of FRDM-K64F

DIGITAL SIGNAL PROCESSING SYSTEM

A basic DSP system, suitable for processing audio frequency signals comprises a digital signal processor and analogue interfaces as shown in figure. The FRDM-K64F and Wolfson Pi Audio Card provide just such a system, using the Cortex-M4 floating point processor on the FRDM-K64F and the WM5102 codec on the Wolfson Pi Audio Card. The term codec refers to the coding of analogue waveforms as digital signals and the decoding of digital signals as analogue waveforms. The WM5102 codec performs both the analogue to digital conversion (ADC) and digital to analogue conversion (DAC) functions shown in figure.



Design of MATLAB Simulink Model

The project simulation is carried out using Simulink by MATLAB. We can design models in Simulink®, generate code, and run the executables on NXP FRDM-K64F. More details regarding this is given in MathWorks official documentation.

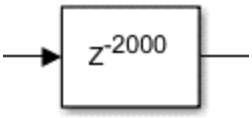
The capabilities include automated build and execution, processor-optimized code for ARM® Cortex-M, block libraries for on-chip and on-board peripherals, real-time scheduling from CMSIS-RTOS RTX, deployment support and real-time parameter tuning, and signal monitoring when tethered to Simulink.

The supported peripherals and key features for the NXP FRDM-K64F board are:

- Deterministic real-time execution at rates reaching 1kHz
- High-performance signal monitoring (20 signals at 100Hz)
- Digital input/outputs
- Analog input/outputs
- PWM outputs
- TCP/IP and UDP blocks
- I2C read/write
- Serial receive/transmit
- Quadrature encoder
- On-board accelerometer, magnetometer, and push button(s)
- SD card logging

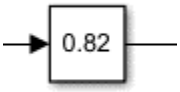
For designing various sound effects, we used the following Simulink Blocks.

Delay



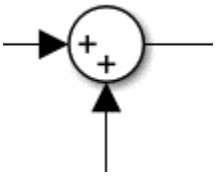
Delay input signal by a specified number of samples.

Slider Gain



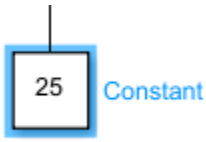
Move the slider to modify the scalar gain.

Sum



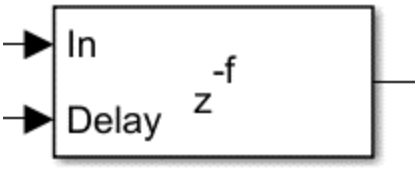
Add or subtract inputs. Specify one of the following:
a) character vector containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension.

Constant block



Output the constant specified by the 'Constant value' parameter. If 'Constant value' is a vector and 'Interpret vector parameters as 1-D' is on, treat the constant value as a 1-D array. Otherwise, output a matrix with the same dimensions as the constant value.

Variable Delay

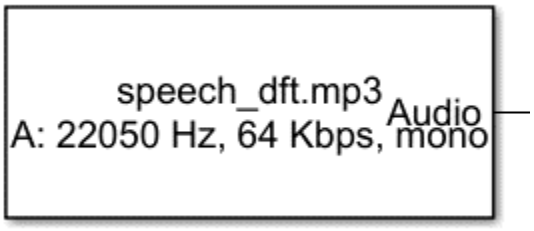


Delay discrete-time input by the time-varying fractional number of sample periods specified by the 'Delay' input. The input delay is clipped to a valid range (Dmin to Dmax) that is determined by the parameter settings. The block provides Linear, FIR, and Farrow interpolation modes. In FIR mode, the filter is designed using the 'firnyquist' function when Bandwidth value is 1, and 'intfilt' function from the Signal Processing Toolbox when Bandwidth is smaller than 1.

Audio sink

Play audio stream from your computer's audio device.

Audio Source

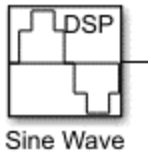


Reads multimedia files containing audio, video, or audio and video data.

For Windows platforms, this block reads compressed or uncompressed multimedia files.

For non- Windows platforms, this block reads uncompressed video and audio AVI files, and video only, compressed or uncompressed files.

Sinewave generator



Output samples of a sinusoid. To generate more than one sinusoid simultaneously, enter a vector of values for the Amplitude, Frequency, and Phase offset parameters.

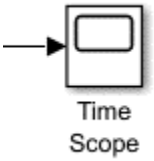
Random source

Output a random signal with uniform or Gaussian (normal) distribution. Set output repeatability to Nonrepeatable (block randomly selects initial seed every time simulation starts), Repeatable (block randomly selects initial seed once and uses it every time simulation starts), or Specify seed (block uses specified initial seed every time simulation starts, producing repeatable output).

Lowpass filter

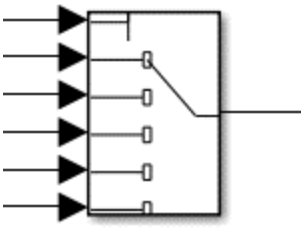
Design a FIR or IIR lowpass filter using desired configuration.

Time scope



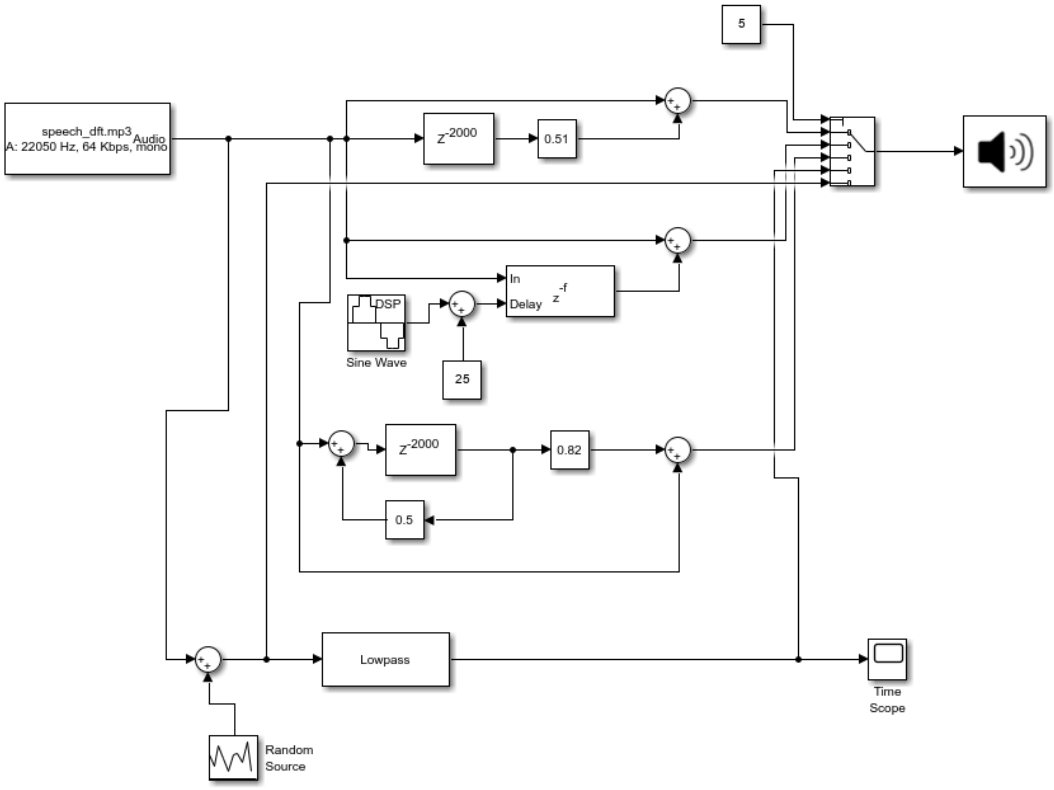
To view the output signal

Multiport Switch



Pass through the input signals corresponding to the truncated value of the first input. The inputs are numbered top to bottom (or left to right). The first input port is the control port. The other input ports are data ports.

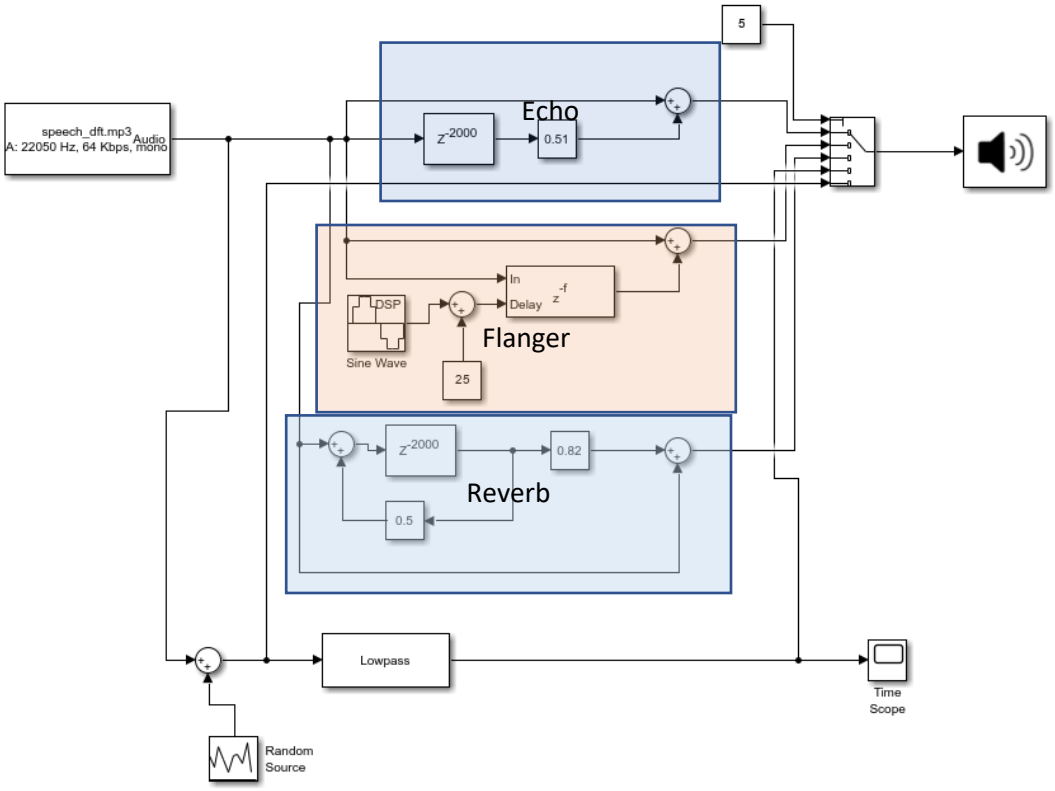
Complete Model



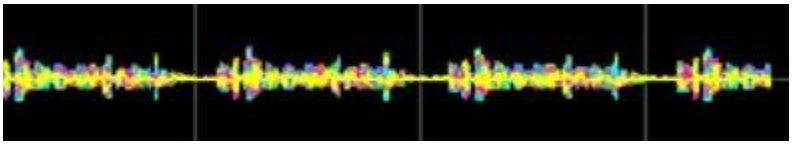
The Simulink model given above represents the initial design of our project and during the implementation in Keil we have changed some of the effects. Delay based effects like Echo, Flanger, Reverb are shown here (from top to bottom order). The low pass filter and noise addition set up is for demonstration of denoising using FIR filter.

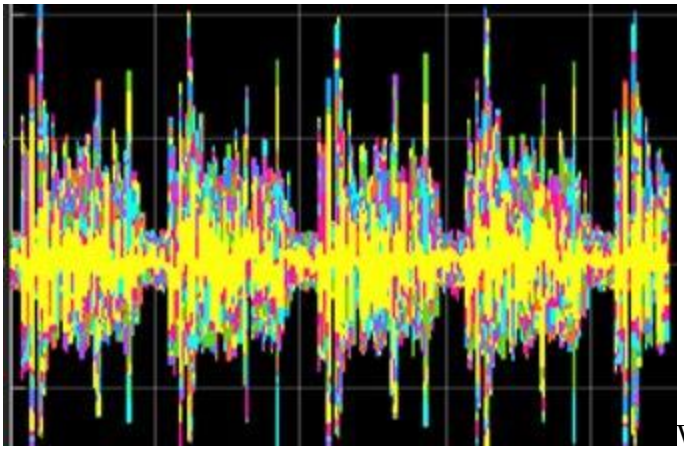
The various part s of this model are used for separate effects They are indicated in the below diagram using separate colored boxes.

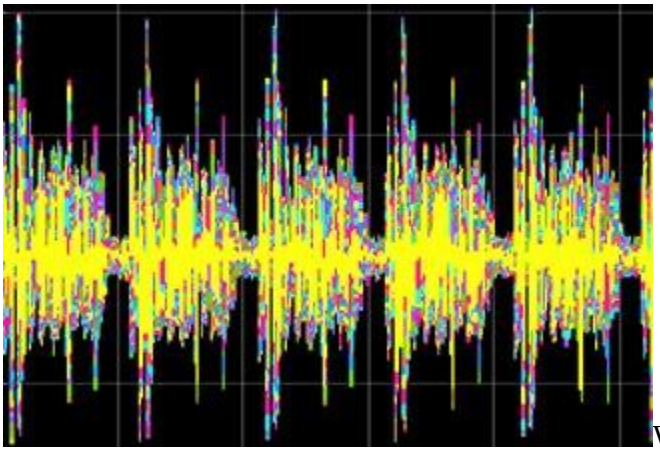
Selection of output signal is according to the switch position; we can change this by using the constant block right above the multiport switch. Selected switch value will provide corresponding effect in the speaker output.

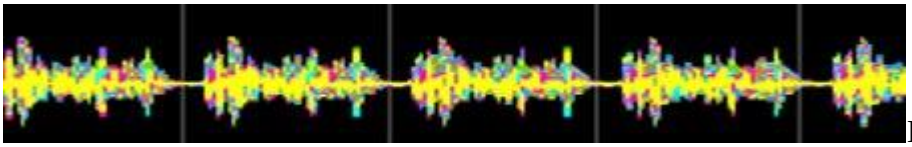


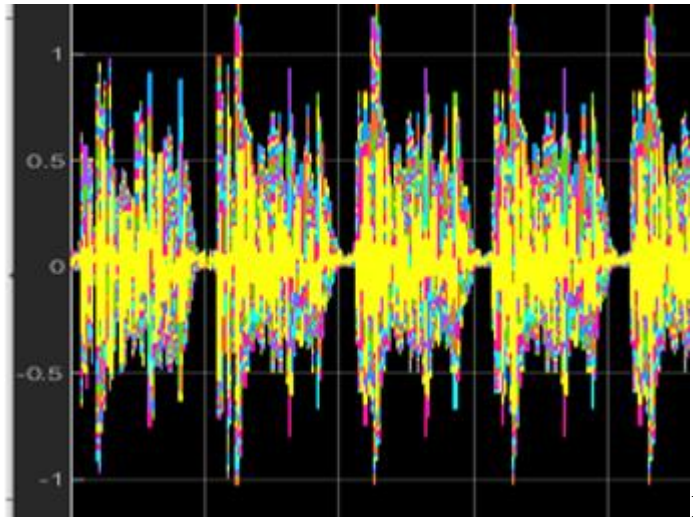
The output waveforms from the above model:

 Original wave form

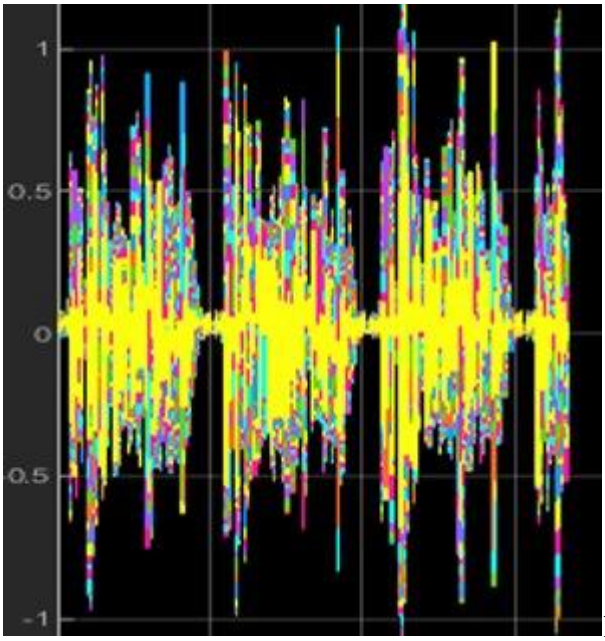
 Without Noise


With Noise


Echo



Reverb



Flanger

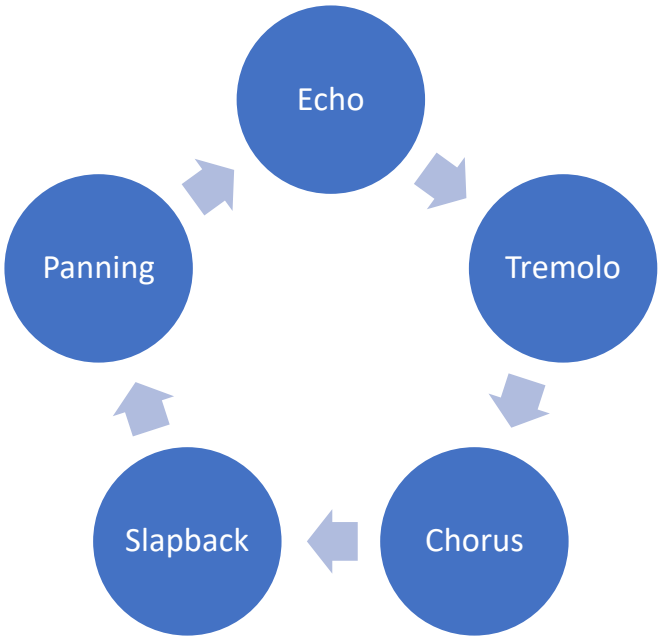
Specifications:
FRDM K64F
Wolfson pi

Software:
Keil MDK-ARM, MATLAB, MATLAB Audio Tool Box, WinScope, Sweep Generator.

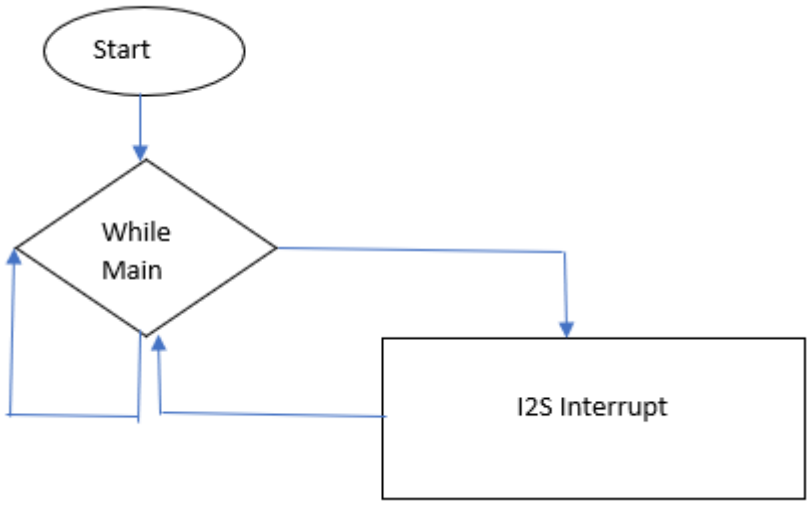
System:

Implementation Methodology:

For Implementation of this system State Machine model is used to switch between different audio effects. System will always be in one of the states and will switch to the next state on press of a button.



Initialized microcontroller peripherals, audio interrupt handler with selecting the input audio device either mic or line in. From the available two switches on board, SW1 is configured as input and used for switching the audio effects on pressing.
In the main while loop check for switch press is done. During execution of main loop if I2S interrupt occurs then microcontroller serves the interrupt and goes back to execution of main routine. I2S handler will receive the audio signal, process it and then send to the output using I2S protocol.



Debouncing of switch:

Often false detections of push buttons are detected usually due to mechanical and physical, this transition may be read as multiple presses or can be detected when it is not actually pressed, for debouncing here, checking is done twice in a short period of time to ensure button is really pressed.

Short Audio Sample will be captured in a buffer and then will be added with the required delay and amplitude back to original signal, Bytes of audio on left and right channel are separated with

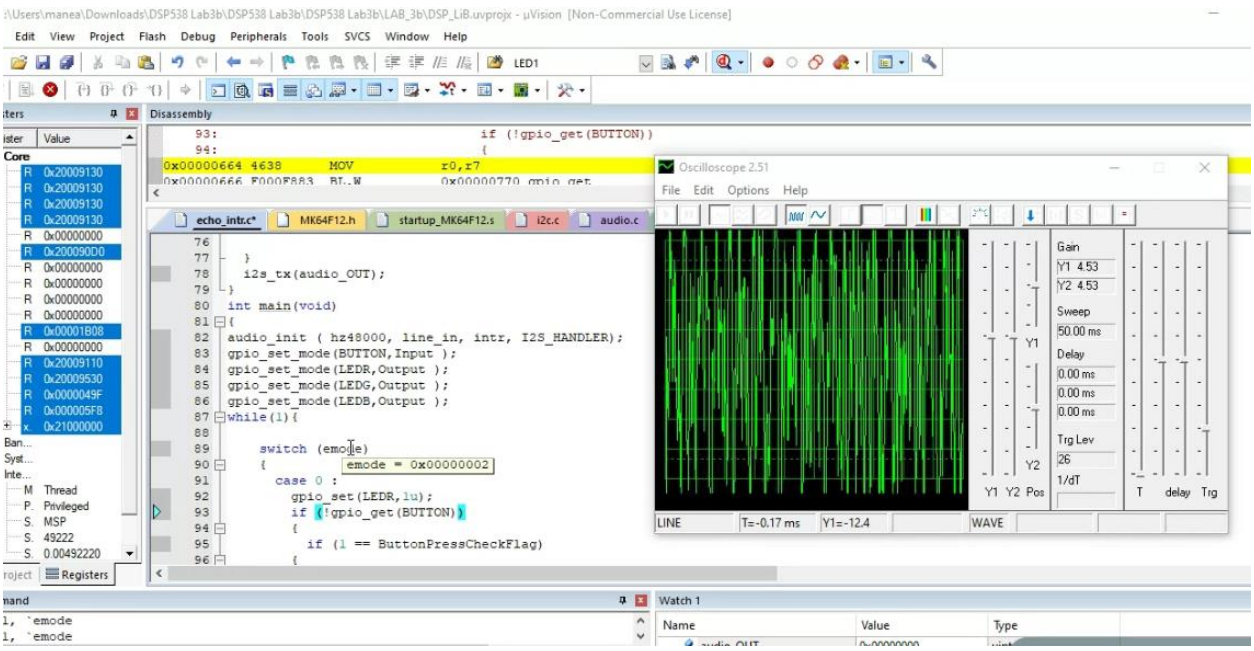
required shifting of bytes, masking and sent to output.

```
   delayed_sample = buffer[buf_ptr];
   audio_out_chL = delayed_sample + audio_chL;
   buffer[buf_ptr] = audio_chL + delayed_sample*GAIN;
   buf_ptr = (buf_ptr+1)%DELAY_BUF_SIZE;
   audio_OUT = ((audio_chR<<16 & 0xFFFF0000)) + (audio_out_chL& 0x0000FFFF);
 break;
 case 2: // Tremelo
 delayed_sample = buffer[buf_ptr];
 audio_out_chL = delayed_sample*sine_table[sine_ptr];
 audio_chR =  delayed_sample*sine_table[sine_ptr];
 buffer[buf_ptr] = audio_chL ;
 buf_ptr = (buf_ptr+1)%DELAY_BUF_SIZE;
 //--audio_out_chL = audio_chL*vargain;
 //--vargain= vargain+2;
 sine_ptr = (sine_ptr+1) % LOOP_SIZE;
 audio_OUT = ((audio_chR<<16 & 0xFFFF0000)) + (audio_out_chL& 0x0000FFFF);
 //if (vargain>=10)
 //{ vargain =0.1;}
 break;
 case 3: //slapback
 delayed_sample = buffer2[buf_ptr2];
 audio_out_chL = delayed_sample + audio_chL;
 buffer2[buf_ptr2] = audio_chL + delayed_sample*0.4;
 buf_ptr2 = (buf_ptr2+1)%DELAY_BUF_CH1;
 audio_OUT = ((audio_chR<<16 & 0xFFFF0000)) + (audio_out_chL& 0x0000FFFF);
 break;
```

```
case 5: //Pan Audio
if(pflag==0u) {
delayed_sample = buffer[buf_ptr];
audio_out_chL = 0.1*delayed_sample;
audio_out_chR = 1.5*delayed_sample;
buffer[buf_ptr] = audio_chL;
buf_ptr = (buf_ptr+1)%DELAY_BUF_SIZE;
audio_OUT = ((audio_out_chR<<16 & 0xFFFF0000)) + (audio_out_chL& 0x00000000);
pflag =1u;
}
else if(pflag == 1u){
delayed_sample = buffer[buf_ptr];
audio_out_chL = delayed_sample;
buffer[buf_ptr] = audio_chL;
buf_ptr = (buf_ptr+1)%DELAY_BUF_SIZE;
audio_OUT = ((audio_out_chR<<16 & 0xFFFF0000)) + (audio_out_chL& 0x0000FFFF);
pflag = 2u;
}
else if(pflag ==2u){
 //Pan Audio 2
delayed_sample = buffer[buf_ptr];
audio_out_chL = 1.5*delayed_sample;
audio_out_chR = 0.1*delayed_sample;
buffer[buf_ptr] = audio_chL;
buf_ptr = (buf_ptr+1)%DELAY_BUF_SIZE;
audio_OUT = ((audio_out_chR<<16 & 0x00000000)) + (audio_out_chL& 0x0000FFFF);
pflag =0u;
```

Tremolo effect is achieved by multiplying incoming audio with a triangular wave to get the effect. Panning Effect is achieved by reducing the amplitude of one channel and increasing the amplitude of other channel this results in panning effect which gives sense of object moving from one point to other.

Output:
Output audio effect wave form on WinScope.



Conclusion:
The said design was implemented using FRDM and Wolfson pi board with concepts of signal processing, multiplication of signals, addition of signals, change of channels and manipulation of its amplitude to achieve desired audio effects like Chorus, Echo, Tremolo, Slap back and panning. Audio effects were generated for Real Time Audio input from mic and any audio data played from any device.

Future works: In the future, we plan to make the system wireless and IoT-based, as well as add lighting to it.

1. **References**:
   ECE538 : DSP-LAB 1 , 5, 6.
2. Introduction to Signal Processing, Sophocles J. Orfanidis
3. Digital Signal Processing Principles, Algorithms and Applications Third Edition, John G. Proakis and Dimitris G. Manolakis
4. https://www.productionmusiclive.com/blogs/news/audio-effects-explained-2021
5. https://blog.landr.com/audio-effects-plugins-guide/