Department of Electronic & Telecommunication Engineering

University of Moratuwa

Simple Voice Recorder

Group - 21

Rukmal M.A.D.	190531L
Rukshanth S.	190533U
Ruvindi S.M.P.M.	190535D
Sadith W.M.L.	190538N

Supervisor: Mr. Thamindu Naveen

August 2021

Abstract

Our team has developed a simple voice recorder that records the sound and plays back the recorded sound, with additional effects added to it, using the ATmega2560 microcontroller and other inexpensive electronic components. Further, we experimented with the PCB designing, the various aspects of enclosure design, and other aesthetic aspects which would help to make this device a commercially successful device

1. Introduction

A Voice recorder is a device that takes audio input, stores it, and plays recorded audio. Our project was to create a voice recorder with an 8000-sampling rate and 8bit resolution. In addition to that, it must be capable of adding effects to the recorded audio. So, the device mainly consists of a PCB, microphone, SD card, speaker, and an LCD.

When power is supplied the device plays a welcome tone and LCD shows the options "Record", "Gallery" and "About device". Using the up, down, ok, and back buttons we can choose the options. Recording action is indicated by a red led bulb. The gallery consists of the previously recorded files. After selecting a file from the gallery, we can choose whether to play, add effects, or delete the file.

This project is based on an Atmega2560 microcontroller. We used Proteus for the simulation and Microchip studio to code with AVR. The PCB was designed using Altium and the enclosure was designed using SolidWorks.

Our project group consisted of four members. The following will be the details on how we carried out the project, implemented algorithms, schematics, and enclosure designs.

2. Method

a. Recording the sound

The microcontroller (ATmega 328P) which we used for this project includes the inbuilt Analogue to Digital converter (ADC). We used mic module to get inputs to the ADC. It includes pre-amplifier also.

We choose ADC channel zero (PC0 pin of ATmega 328P) to get the input.

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure: 1 ADMUX Register

The main settings of the ADC can be done using ADMUX and ADCSRA registers. First, we have to choose the suitable reference voltage to the ADC. BY setting 6th and 7th bits of ADMUX, AV_{cc} with external capacitor at AREF pin was choose.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

Figure: 2 Reference voltage selection

A capacitor of 100nF and inductor of 10uH are used for this purpose to cancel the noise comes from the reference power lines.

The ADC of the ATmega 328P is a 10 bit one, so the conversion result comes in between 0 and 1023. To change this to 8-bit ADC result, ADLAR bit of ADMUX register changed to 1. It will left shift the converted result to have it as 8-bit one.

MUX3	MUX2	MUX1	MUX0	Input to ADC module
0	0	0	0	Arduino A0
0	0	0	1	Arduino A1
0	0	1	0	Arduino A2
0	0	1	1	Arduino A3
0	1	0	0	Arduino A4
0	1	0	1	Arduino A5

Figure: 3 ADC channel Selection bits

To select the ADC channel to get the input from the microphone, set the bits MUX3:0 to select the ADC0.

ADMUX = 0b01100000;

In ADCSRA register,

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Figure: 4 ADCSRA Register

ADEN bit set to 1 to enable the ADC and ADSC to enable the single conversion.

The ADC circuitry needs to have a clock signal provided to it in the range of 50 kHz to 200 kHz. There is no external ADC clock input on the chip but rather the clock is generated internally from same clock used to run the microcontroller. This CPU clock is too fast (ex: 16MHz) so the chip includes an adjustable "prescaler" to divide the CPU clock down to something acceptable. The prescaler can be set to divide by a choice of divisors (2, 4, 8, 16, 32, 64, or 128). By setting ADPS2:0 bits give the clock signal of 250 kHz to the ADC.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Figure: 5 Prescaler factors

ADCSRA = 0b11000110;

After initializing the basic settings of the ADC, to take the samples at 8000 Hz, we had to call to the ADC in that rate. For that Timer 2 is usedto generate interrupts in CTC mode (clear timer compare).

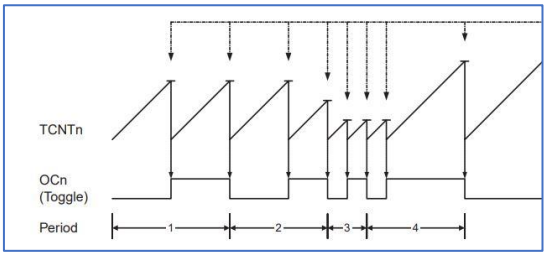

Figure: 6 CTC timer graph

At each time interrupt was triggered, it calls to the ADC to do a single conversion and wait till the conversion is over. After the converted value stored in ADCH register is taken in to variable to write in the file.

RIFF WAVE File Format

Endian	File Offset	Field Name	Field Size	Description
big	0	Chunk ID	4	**"RIFF" Chunk Descriptor**
little	4	Chunk Size	4	The format field here is "WAVE", and it requires
big	8	Format	4	two sub-chunks: "fmt" chunk and "data" chunk.
big	12	SubChunk1 ID	4	
little	16	SubChunk1 Size	4	
little	20	Audio Format	2	
little	22	Channel Number	2	**"fmt" Sub-Chunk**
little	24	Sample Rate	4	This region contains all essential information we
little	28	Byte Rate	4	need to know about a WAVE file, such as sample
little	32	Block Align	2	rate, channel number etc.
little	34	Bits Per-Sample	2	
big	36	SubChunk2 ID	4	**"data" Sub-Chunk**
little	40	SubChunk2 Size	4	The field "SubChunk2 Size" indicates the size of
little	44	Data	SubChunk2 Size	RAW audio data.

Figure: 7 Header bytes of WAV file

To generate the wav file, the header data was initially written to the file.

The data samples taken by the ADC wrote to the file and after all data samples were written, according to the number of samples chunk size and sunchunk2 size positions of header were updated.

b. Play the files

Since we have taken the samples in 8000Hz, to play the audio we had to send those samples to speaker at the same rate. To do it we use timer interrupts to call to a sample sending function at 8000 Hz.

The output samples are given to the external 8-bit DAC to take the analogue value (since ATmega 328P does not include inbuilt DAC). The analogue voltage was given to the speaker through an amplifier to generate the sound.

c. Play with Scaled frequency

To play with Scaled frequency (change the pitch) we used the time frequency duality concept. Since we want to increase the pitch, we have to expand the frequency domain. To do that we try to compress the time domain of the signal by decreasing the time between two playing samples less than they are recorded. For this we should be able to send the samples to speaker at the precise specific rate. To do it, we

decided to use timer interrupts. At every time timer reaches to the given value it triggers an interrupt to send a sample to speaker. By this way we changed the sample play rate and change the pitch accordingly.

From the several timer interrupt modes we choose CTC (clear timer compare) mode for the purpose. Because in that mode, we could change the triggering rate by setting OCRnx value and timer get reset after the compare occurred.

$$f_{OCnx} = \frac{f_{clk\ I/O}}{2\ .\ N.\ (1 + OCRnx)}$$

By setting prescale factor to desired value and f_{ocnx} to the triggering frequency we need, can calculate a value for OCRnx.

When the samples are played in high rate than they are recorded, we were able to achieve high pitch and when they play slowly, we got low pitch.

d. Play with low pass filter

The implementation we used to have a low pass filter is the "Leaky integrator".

Same as before, we use interrupts to give samples in certain rate and using real time sample and the last sample which played calculates a value by following equation.

$$y = (1 - a)y_{prev} + input * a$$

This is working as a low pass filter implementation. We can figure out it more by looking at its bode plot.

e. MathLab Simulation

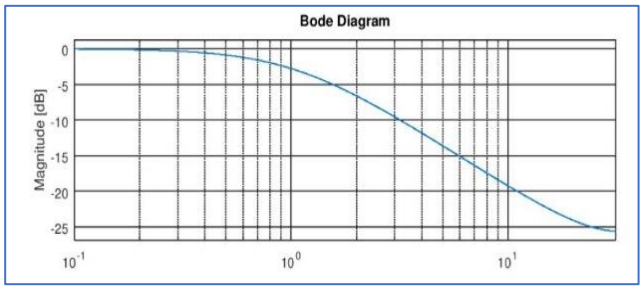

Figure: 8 Bode plot of Leaky Integrator

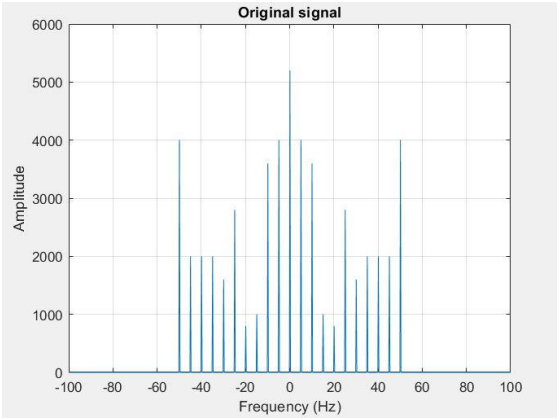

Figure: 9 Original Signal

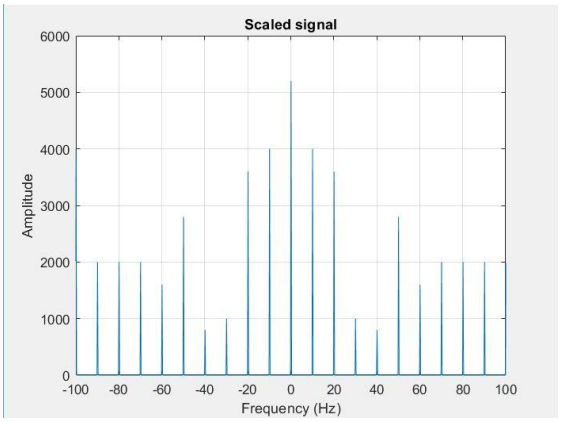

Figure: 10 Scaled Signal

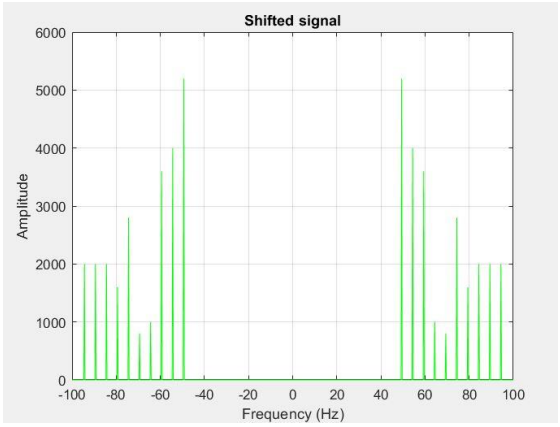

Figure: 11 Shifted Signal

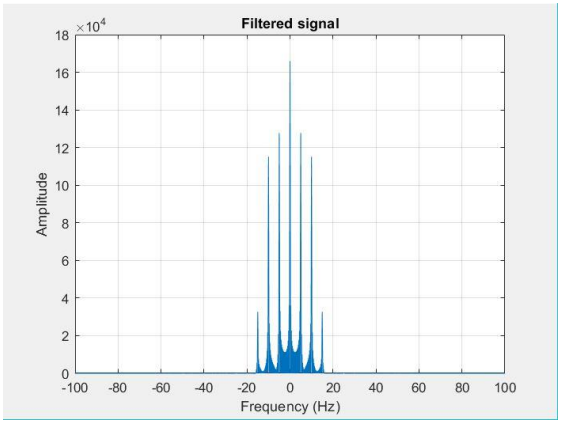

Figure: 12 Filtered Signal

e.1. Audio enhancement techniques used in matlab simulation

- **Filtering**

To implement a low pass filter we use signal processing technique convolution. To take only the low frequency components we had multiply the spectrum of the signal by a square pulse. For that we generated a sinc function and convolved it with original signal in time domain.

When we do this high frequency components are multiplied by zero and low frequency components are multiplied by one giving us only low frequencies as the output.

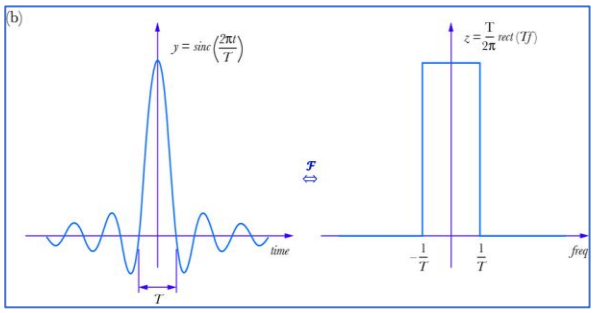

Figure: 13 Dual of sinc function

- **Frequency Scaling**

For this we use time Frequency duality concept. We can stretch the spectrum by compressing the time domain and can compress the spectrum by expanding the time domain.

$$x(t) \leftrightarrow X(\omega)$$

$$x(at) \leftrightarrow \frac{1}{|a|} X\left(\frac{\omega}{a}\right)$$

Figure: 14 Fourier transform of Scaled

When time domain signal scaled with a factor greater than one spectrum stretched and when the factor is less than one spectrum compresses.

When spectrum is stretched, the pitch of audio increases and when spectrum compresses, pitch get reduced.

- **Frequency Shifting**

To shift the frequency components in desired value, we multiply the signal with relevant complex exponential term, to have $X\ (\omega - \omega_0)$

Proteus simulation Oscilloscope Output waveforms

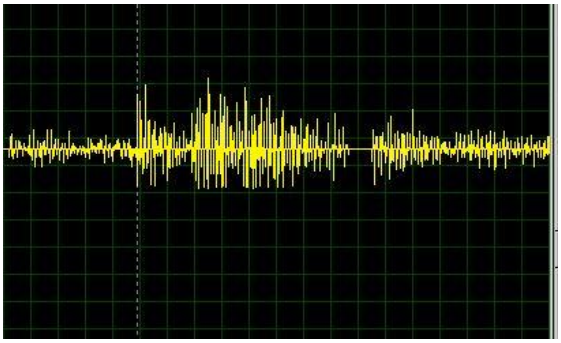

Figure: 15 Original Output without any effect

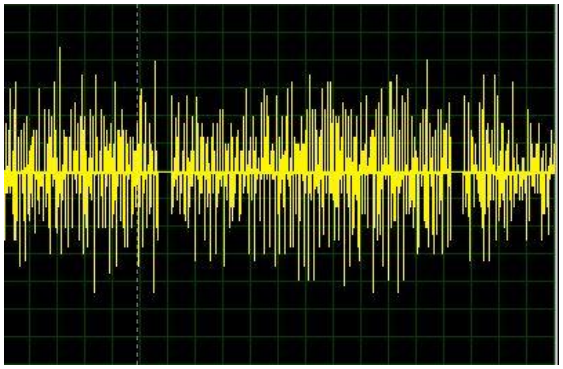

Figure: 16 Output with Filter

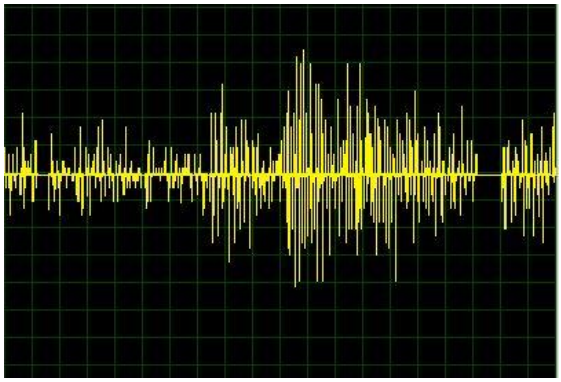

Figure: 17 Frequency Scaled

f. LCD

We used 16x4 LCD as the display. In that LCD we use 4 bit mode to address the pixels because to optimize the pin usage of microcontroller. In the prototype, we used the Arduino LiquidCrystal library to communicate with the LCD. But we implement the LCD addressing code snippet from the basic commands and only able to use it in the proteus simulation.

Since we had to move to ATmega 328P the pins we had to use for LCD decreased more. So we use I2C communication to address the LCD only using SDA and SCL pins. For that Arduino library LiquidCrystalI2C is used.

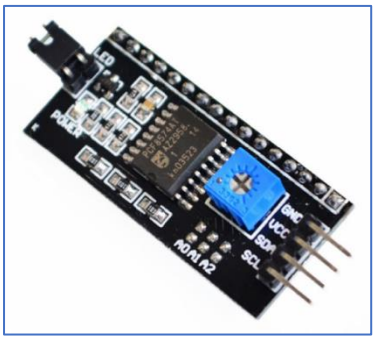

Figure: 18 I2C Module

g. SD Card Module

To store the recorded wav files we used Micro SD card of 8GB. To connect that with the Microcontroller SD card module is used. Since the SD card works in 3.3V and Microcontroller (MCU) works in 5V we had to consider that there should be a logic level converter in the SD card module.

We use SPI communication to address the SD card. For that mainly chip select (SS), Master in Slave out (MISO), Master out Slave in (MOSI) and Clock (SCK) pins are used in the MCU. And to write and read files Arduino SD library is used.

Figure: 19 SD card Module

3. Results

- Record the audio through the microphone to an 8 bit mono ".WAV" file format and stored in the SD card.
- Could not play the selected audio file properly due to the malfunctioning of the amplifier.
- Deleted the selected file from SD card.
- Change the pitch of the recorded audio by frequency Scaling.
- Apply the low pass filter to remove high frequency components.

4. Discussion

a. Choosing the microcontroller

Team made the decision to go with ATmega2560 microcontroller as it had more memory, more pins and more built-in hardware peripherals than the ATmega328.

But it was only available in a surface mount package, so it cannot be inserted into and removed from a socket on the Arduino.

b. Optimizing the Number of Pins Used for Display, Keypad and Buttons

Finding ways to optimize the limited number of pins available in the microcontroller posed a problem. The display would require 7 pins and keypad would require another 7 pins, which would add up to 14 pins for the display and keypad itself.

To reduce this, we considered the following methods.

- We went on with LCD I2C module, which only required 2 data pins to control the LCD.
- Instead of keypad, we got 4 push buttons to implement the whole action in our project.

c. The issue of gathering components

Because of the prevailing situation in the country, the team could not gather all the wanted components. The team continued the project with alternatives.

The DAC 0808 could not be found, the team showed the simulation of the project.

d. Microphone and the Amplifier

KY-038 module is used to get the input voices for this device. The main part of the module is an LM393 comparator. The module was sufficient for the use.

5. List of Components

1. Power Circuit

- ❖ 9V battery holder with a slide switch
  - This used to power up the system using a 9V battery
- ❖ 9V to 5V regulator
  - LM7805M regulator
  - 0.33uF and 0.1uF bypass capacitors to give a stable power supply
  - Green LED and a 100-ohm resistor to indicate power
- ❖ 9V to -5V convertor (For power up the DAC)
  - ICL7660 negative voltage convertor
  - Two 10uF capacitors.
- ❖ 5V to 3.3V regulator (For power up the SD module)
  - AMS1117-3.3 regulator
  - Two 10uF capacitors
  - Two 100uF capacitors

2. Main Circuit

- ❖ ATmega2560 as the controller of the system
- ❖ 4-pin header as UART interface with 5V and GND
- ❖ External Oscillator
  - 16MHz crystal
  - Two 22uF capacitors
- ❖ RESET switch with 10k resistor
- ❖ RED LED with 100-ohm resistor to indicate Recording
- ❖ Four 100nF capacitors as bypass capacitors to give a stable power supply and decrease the ground resistance.
- ❖ 16x2 LCD panel
- ❖ Four push buttons to control the system

3. Audio Input Circuit

- ❖ 6mm condenser microphone
- ❖ Amplifier (For amplify the input audio signal)
  - MMBT2222A-7-F NPN transistor
  - Two 10k resistors
  - One 100k resistor
  - Two 0.1uF capacitors

4. **Audio Output Circuit**
   - ❖ Digital to Analog Convertor
     - DAC0808 device
     - One 5k resistor
     - One 1k resistor
     - One 100nF capacitor
   - ❖ Amplifier (to amplify the output signal)
     - LM386
     - One 1.2k resistor
     - One 10uF capacitor
   - ❖ Speaker
     - 8-ohm speaker
     - 10-ohm resistor
     - 220uF capacitor
     - 0.1uF capacitor

6. **Acknowledgements**

We as a team, sincerely thank our supervisor, Mr. Thamindu Pathirana, for the valuable feedback he gave us at the weekly progress assessment meetings, without which this project would not have been what it is today.

7. **References**

a. http://soundfile.sapp.org/doc/WaveFormat/
b. https://www.microchip.com/en-us/product/ATmega328P
c. https://www.8051projects.net/lcd-interfacing/commands.php
d. Scott Campbell, Arduino LCD Set Up and Programming Guide, https://www.circuitbasics.com/how-to-set-up-an-lcd-display-on-an-arduino/
e. 16x2 Alphanumeric LCD Display Datasheet, https://components101.com/displays/16x2-lcd-pinout-datasheet

8. Appendices

- **Appendix I - Schematic Diagram**

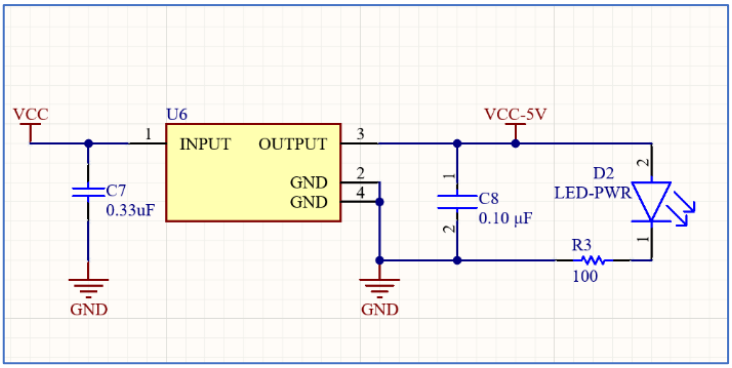

Figure: 20 5V Regulator

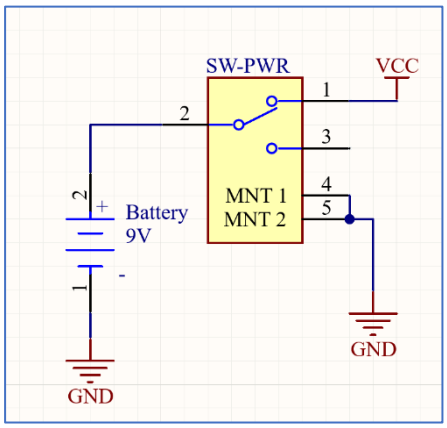

Figure: 21 9V Battery and Slide

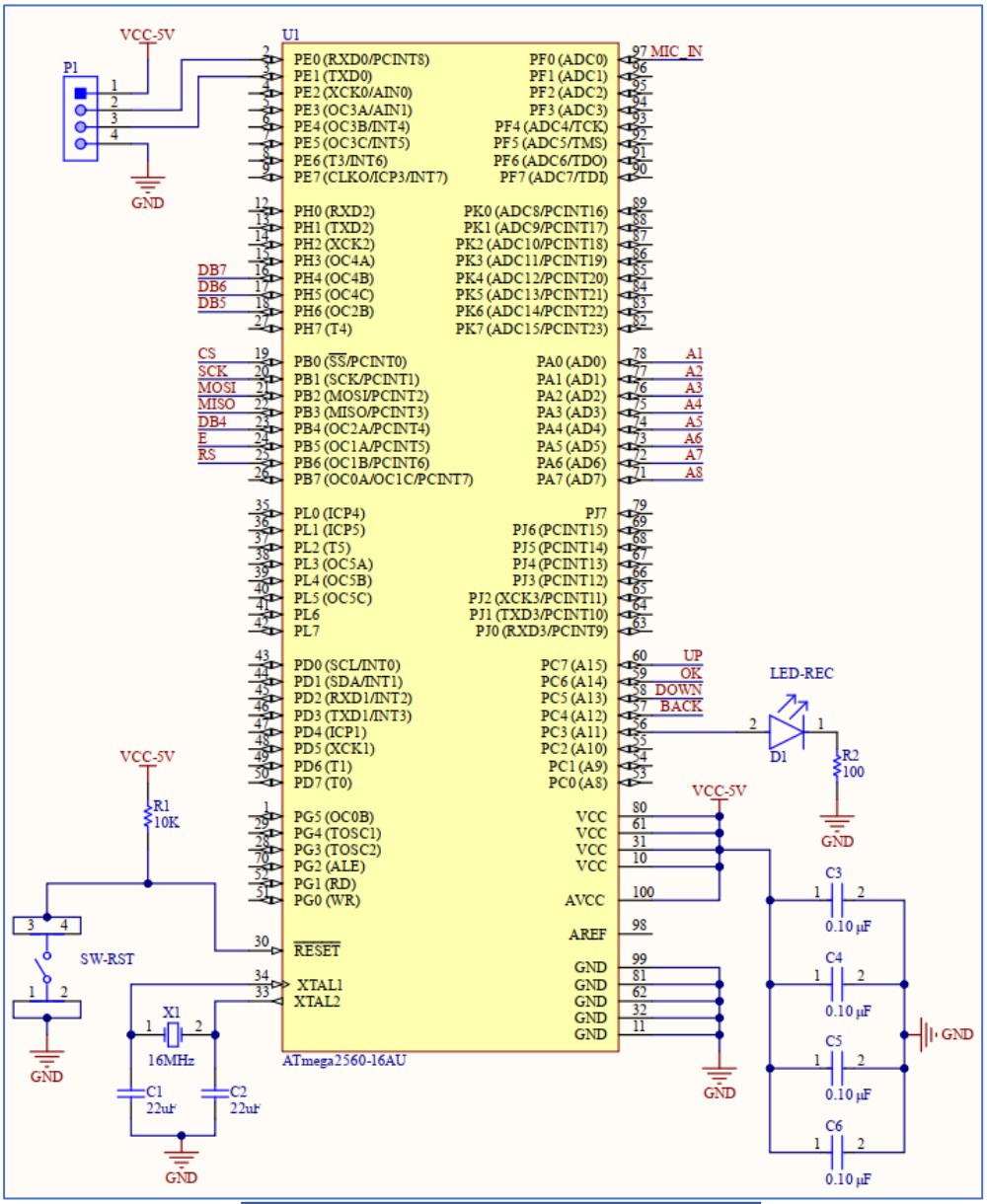

Figure: 22 ATmega2560 -16U

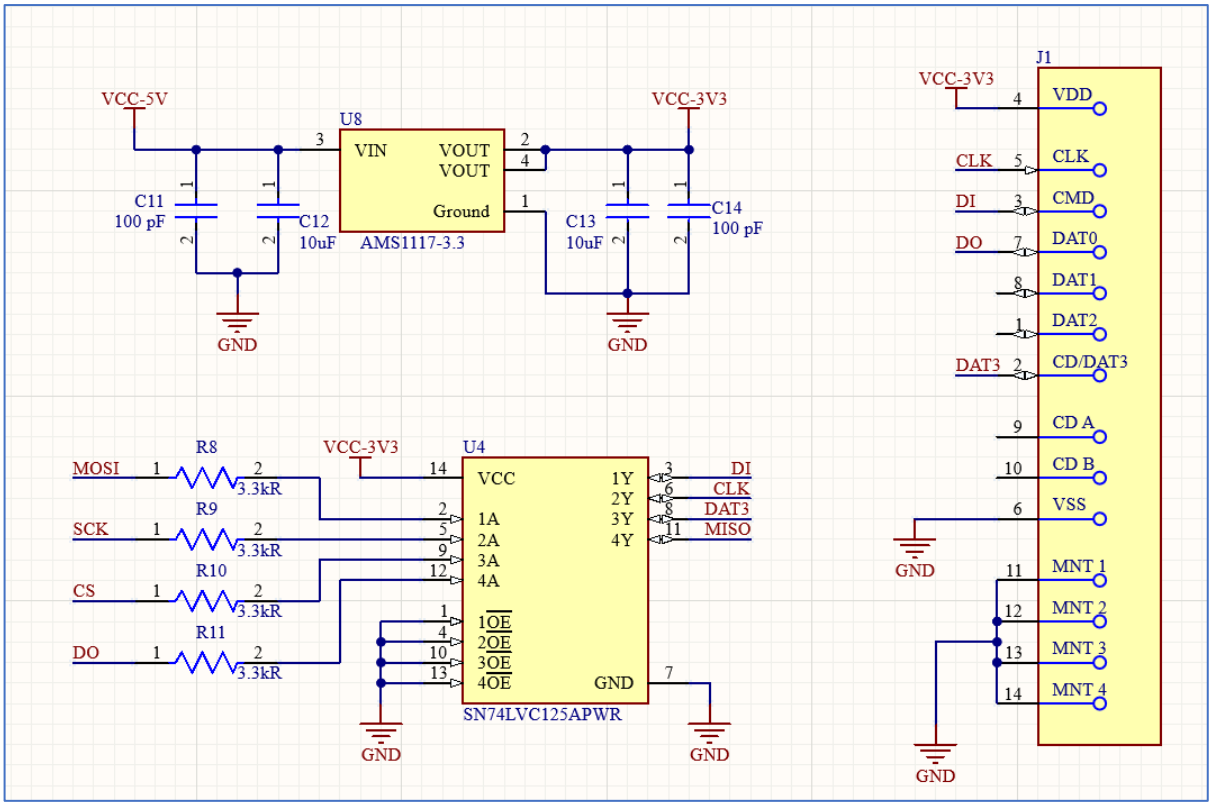

Figure: 23 SD Module

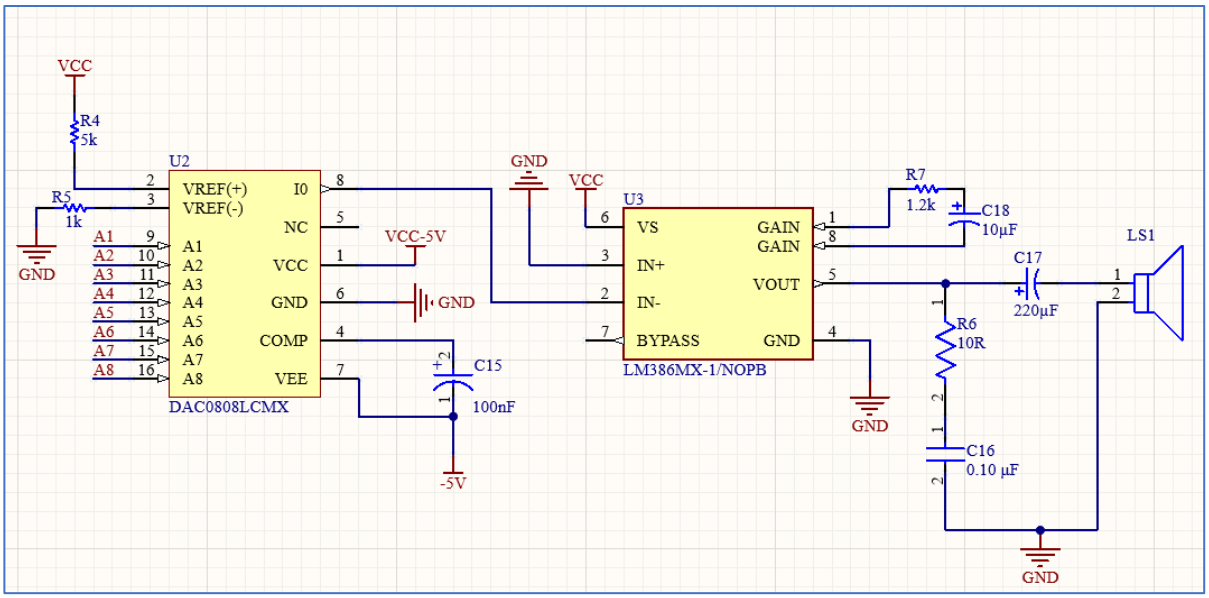

Figure: 24 Audio Output

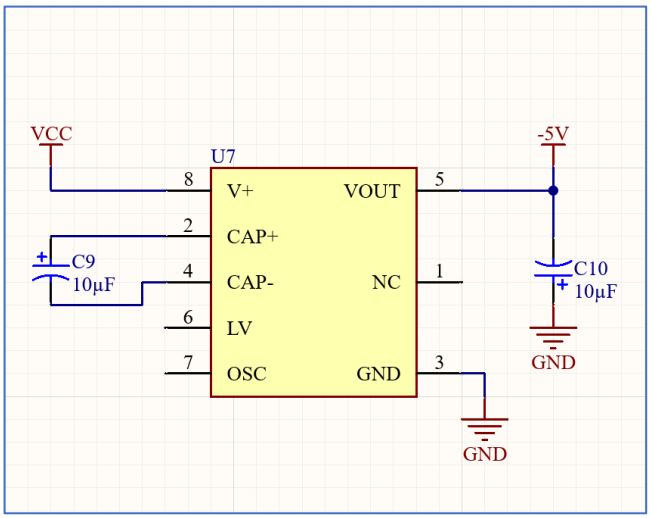

Figure: 25 Negative Voltage Convertor

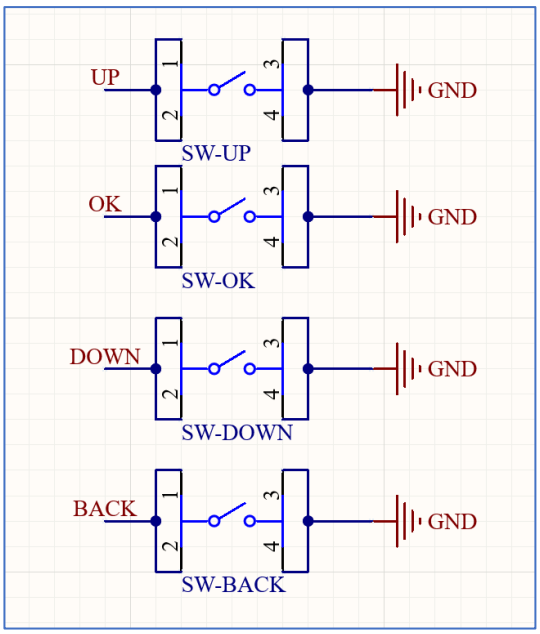

Figure: 26 Push Buttons

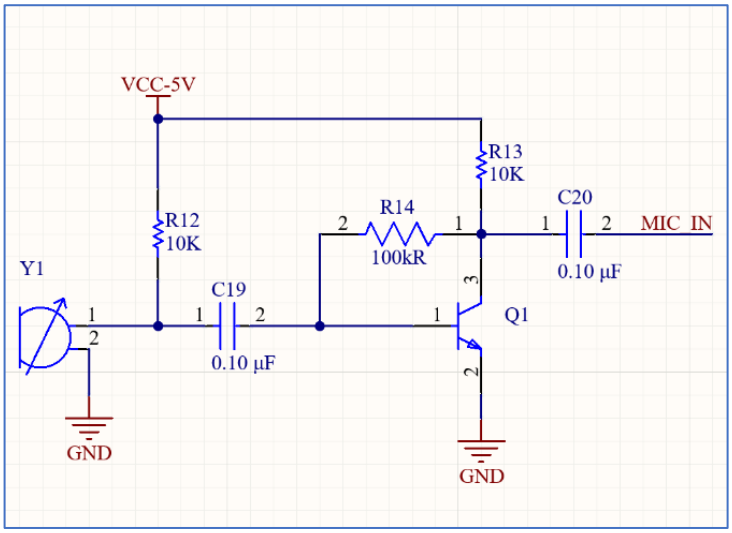

Figure: 27 Audio Input

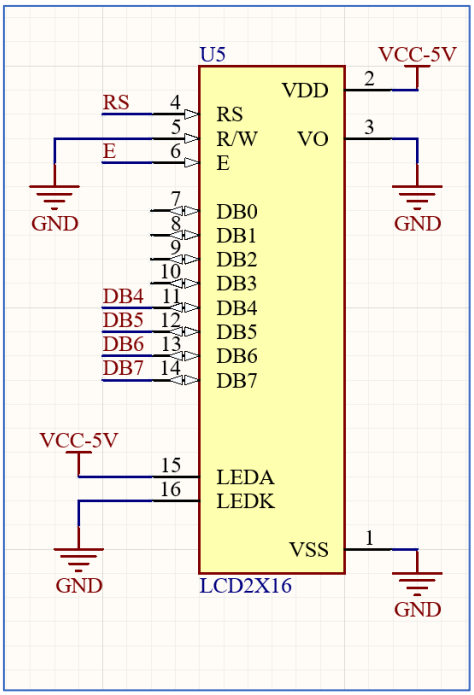

Figure: 28 LCD Module

- **Appendix II – PCB Layout and 3D View**

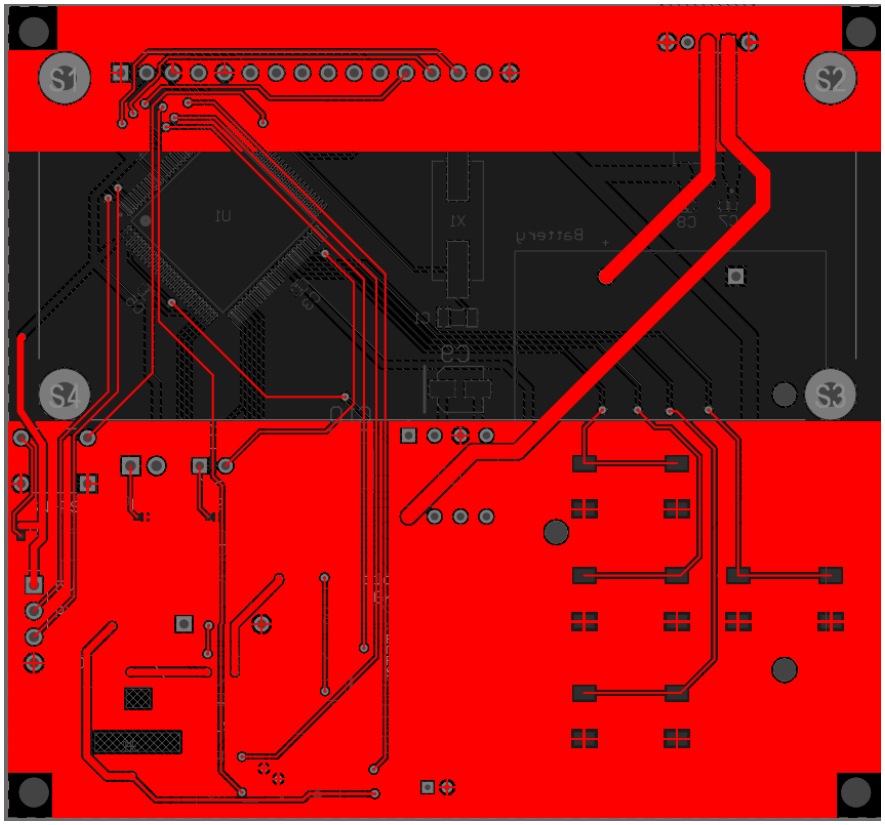

Figure: 29 Tracks on the top

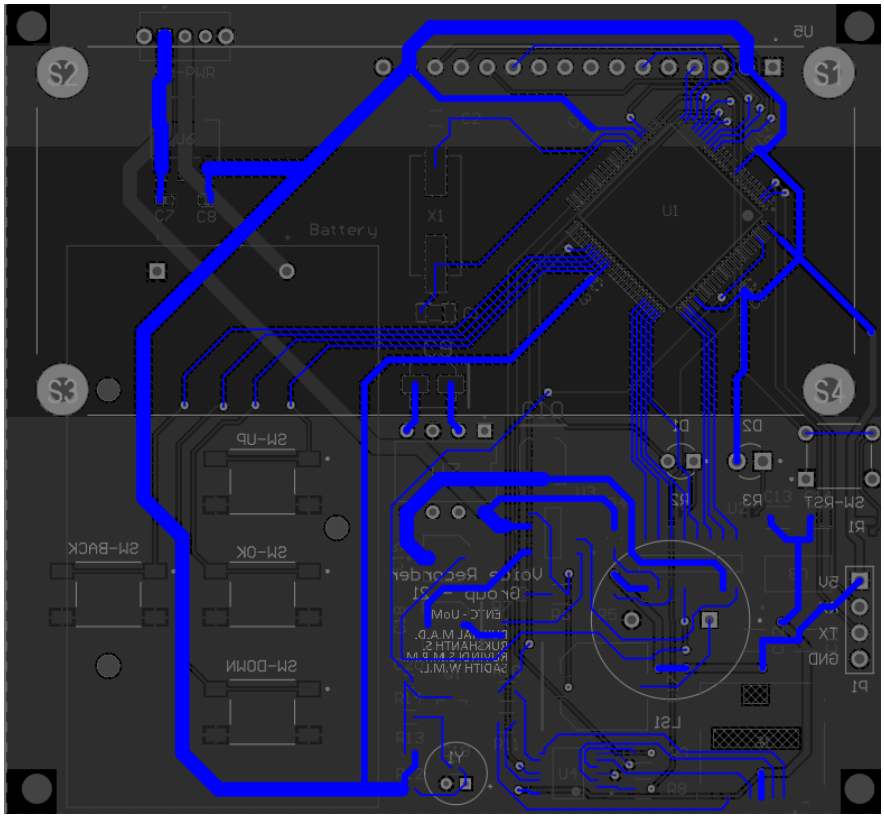

Figure: 30 Tracks on the bottom

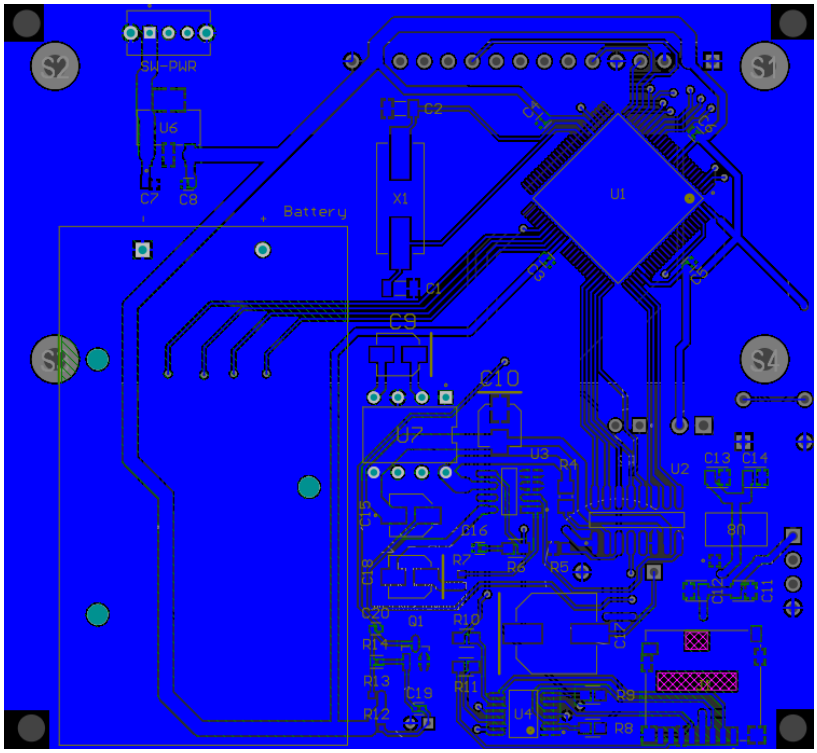

Figure: 31 Tracks and the polygon on the bottom

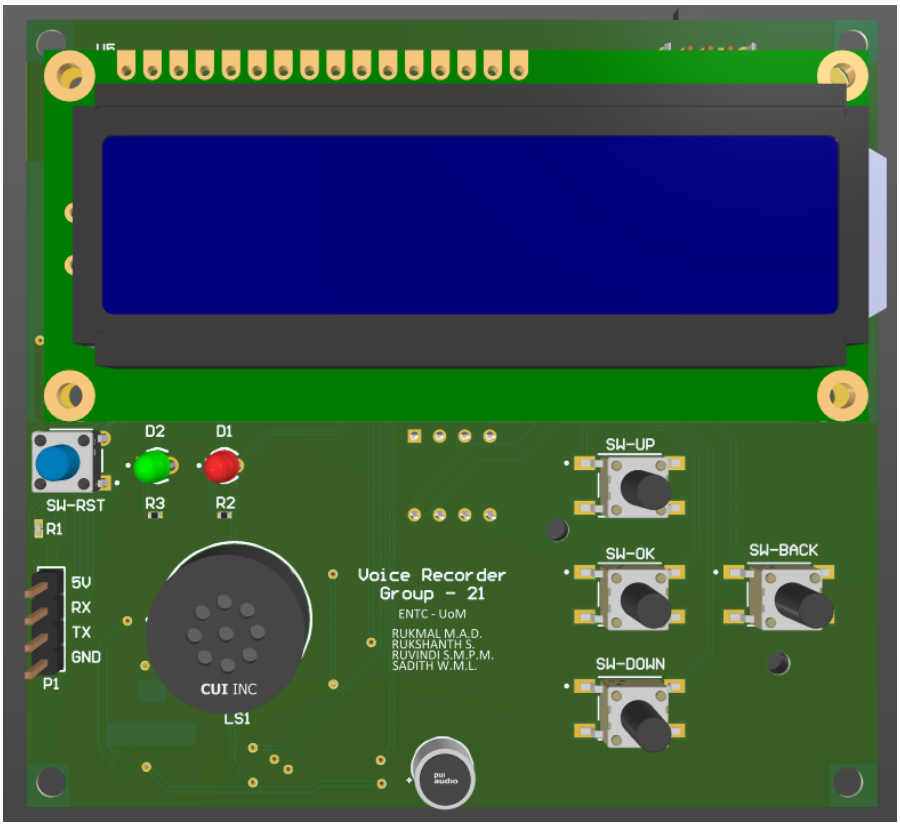

Figure: 32 3D top view

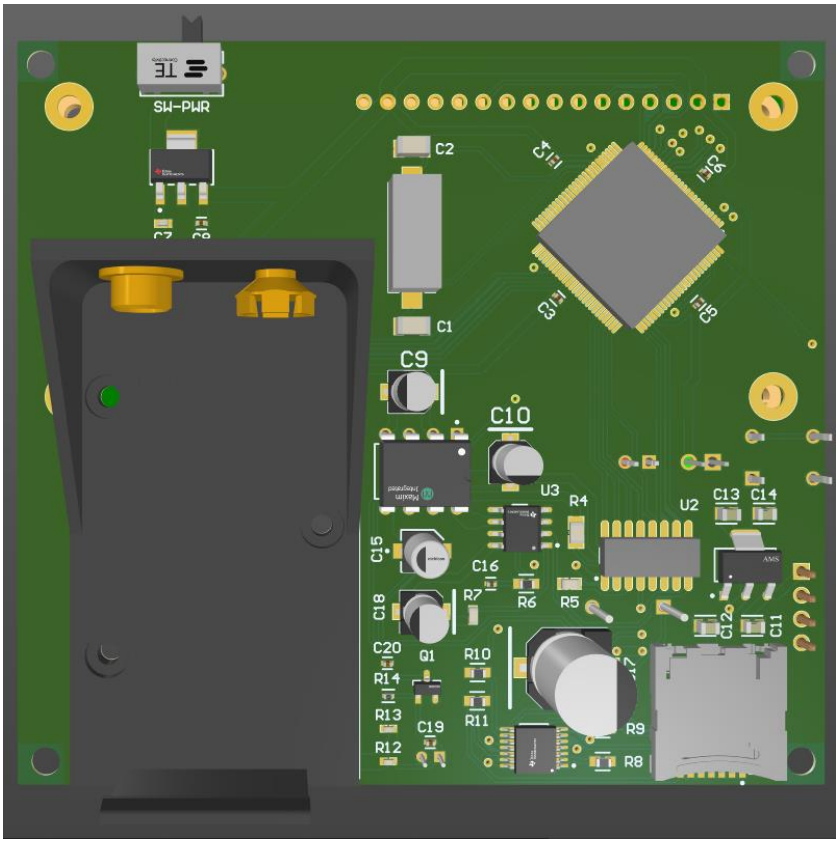

Figure: 33 3D bottom view

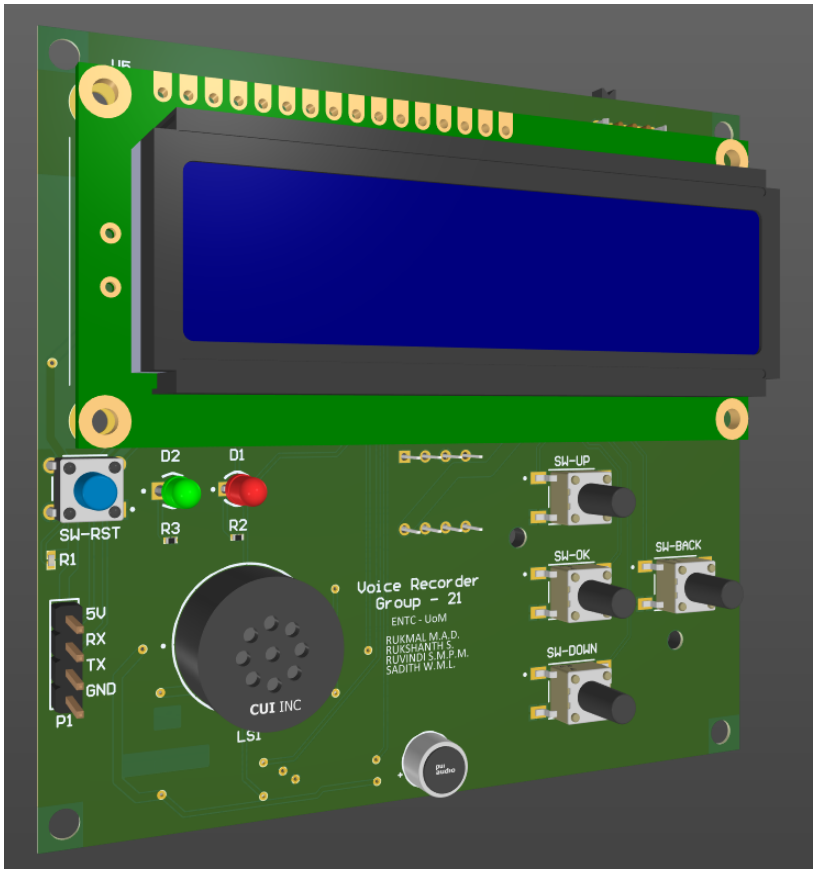

Figure: 34 3D top side view

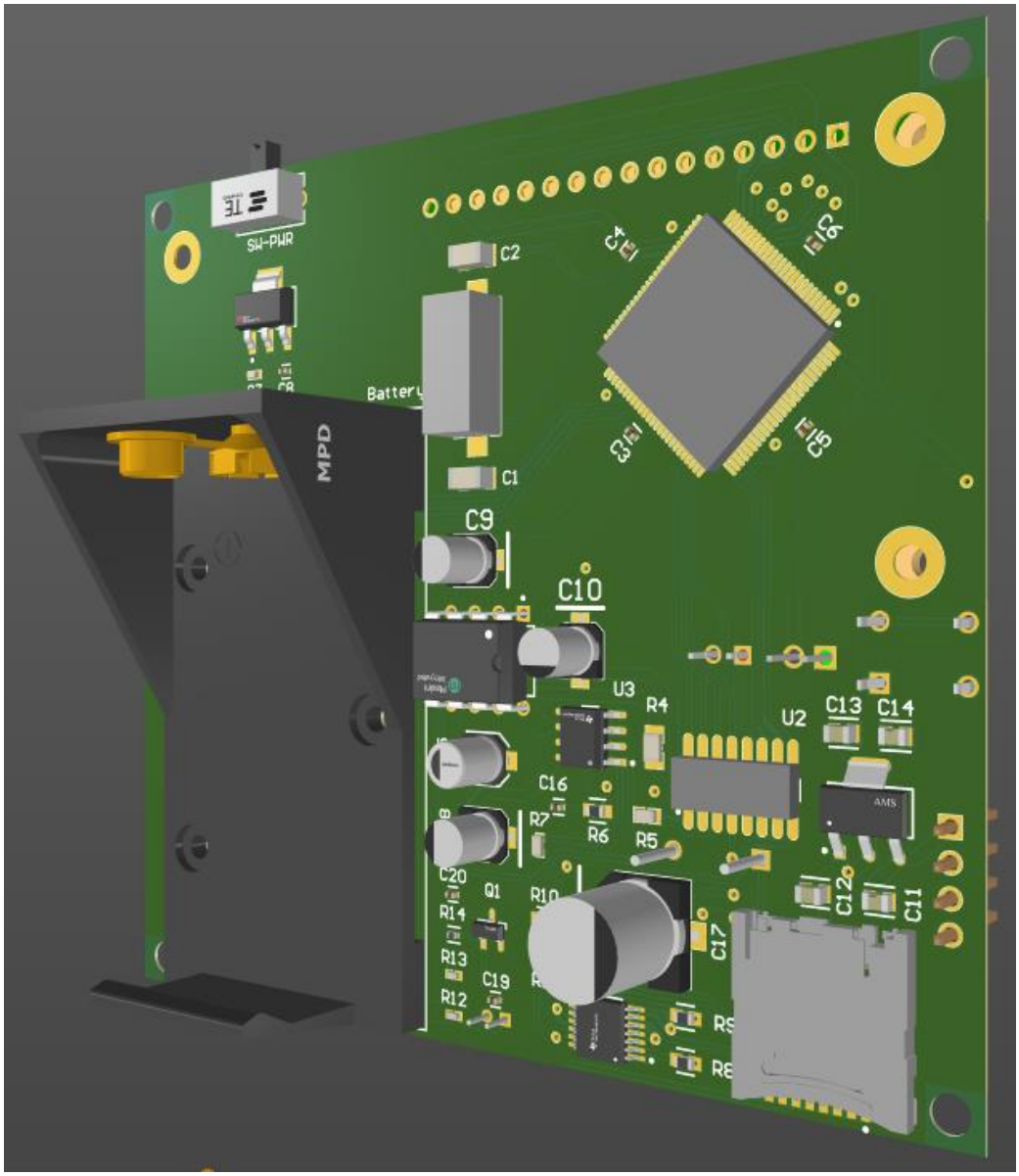

Figure: 35 3D bottom side view

- **Appendix III – Enclosure design**

Figure: 36 Enclosure picture - 1

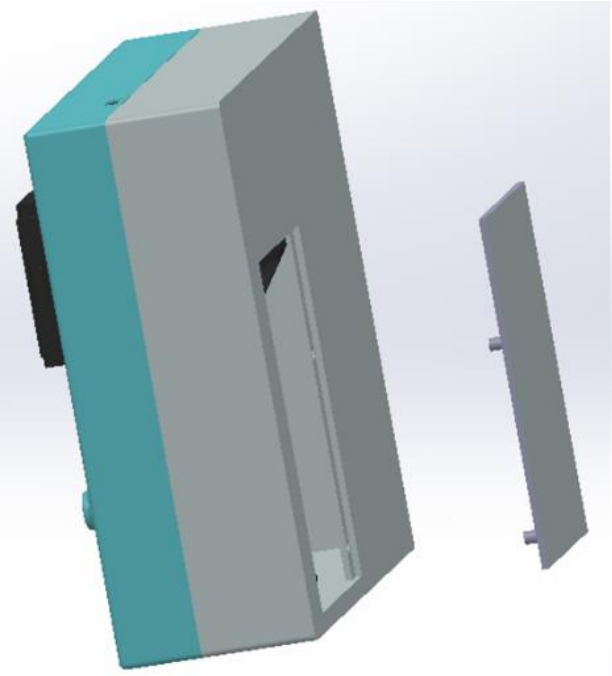

Figure: 37 Enclosure picture - 2

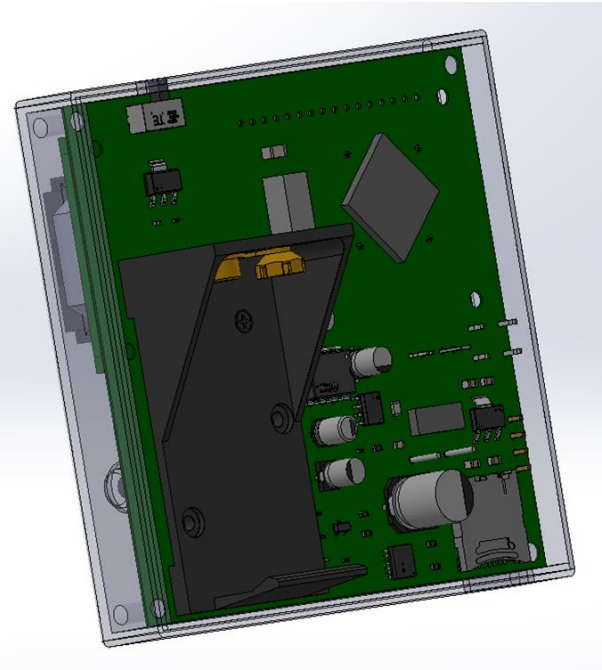

Figure: 38 Enclosure picture - 3

- **Appendix IV – Prototype design**

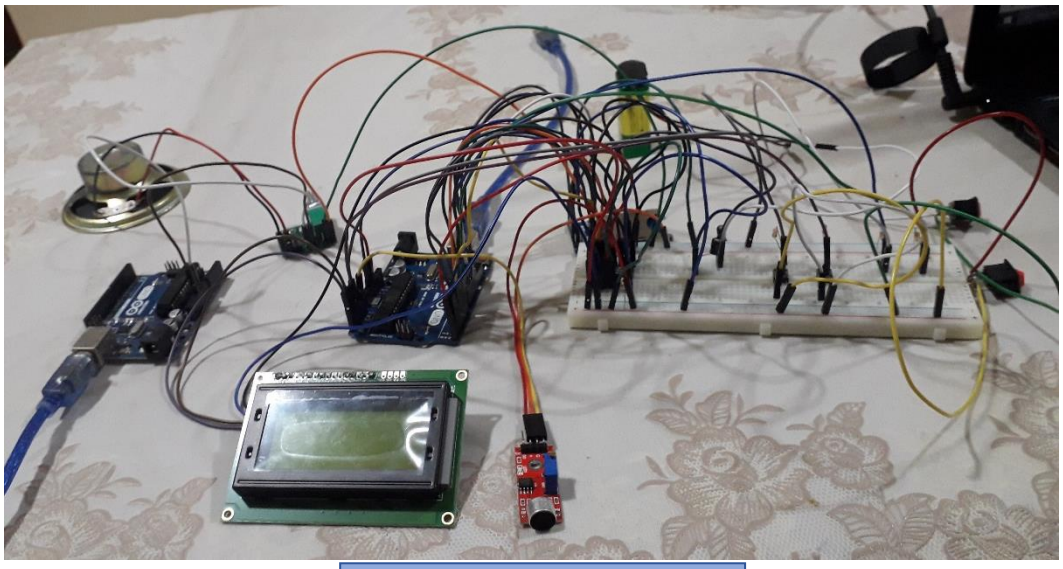

Figure: 39 Prototype picture -1

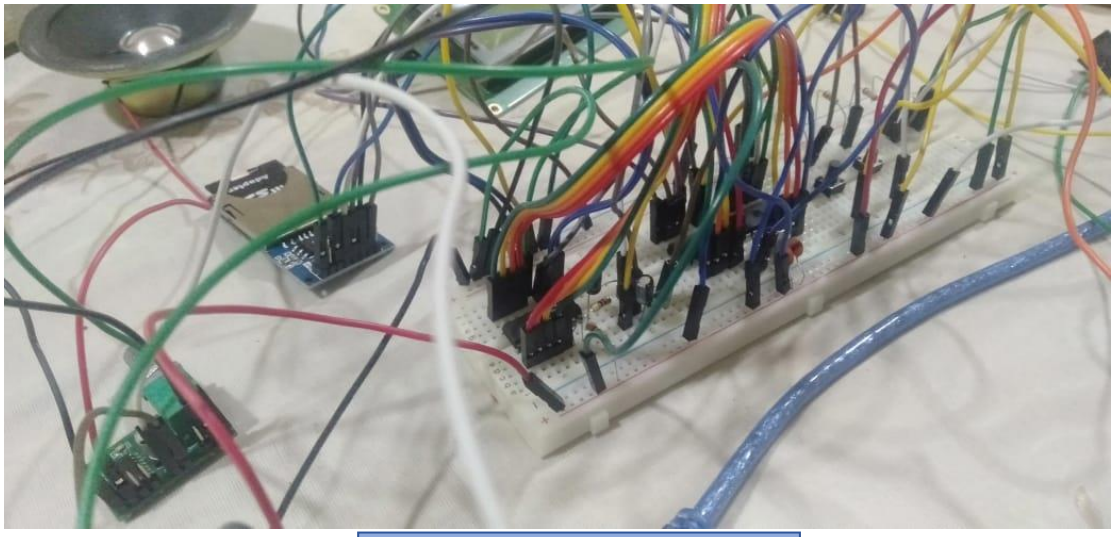

Figure: 40 Prototype picture - 2

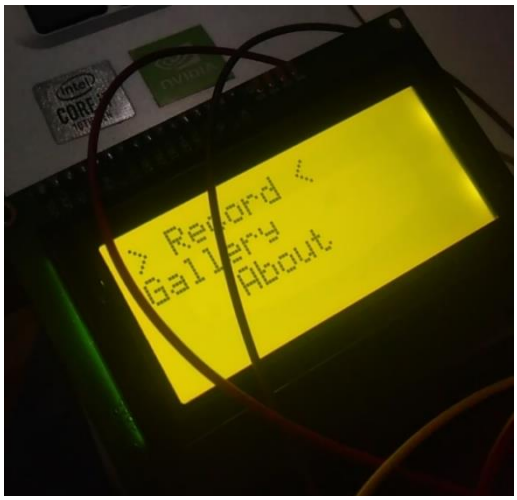

Figure: 41 Prototype picture - 2

- **Appendix V – MATLAB code**

```
close all; %closing all windows


% generate the
signal.....................................................
frequancy = [5 10 15 20 25 30 35 40 45 50 ];
delta0 = 0.001;
time_dom = -2:delta0:2-delta0;
sample_signal = zero_arr(length(time_dom));
for j=1:length(time_dom)
   sample_signal(j) =
1.3+2*sin(2*pi*frequancy(1)*time_dom(j))+1
.8*sin(2*pi*frequancy(2)*time_dom(j))+0.5*s
in(2*pi*frequancy(3)*time_dom(j))+0.4*sin(2
*pi*frequancy(4)*time_dom(j))+1.4*sin(2*pi*
frequancy(5)*time_dom(j))+0.8*sin(2*pi*freq
uancy(6)*time_dom(j))+sin(2*pi*frequancy(7
)*time_dom(j))+sin(2*pi*frequancy(8)*time_
dom(j))+sin(2*pi*frequancy(9)*time_dom(j))
+2*sin(2*pi*frequancy(10)*time_dom(j));
end
figure;
fs = 1000;
f1 = -fs/2:fs/3999:(fs/2);
y1=my_fftshift(abs(my_fft(sample_signal,400
0)));
plot(f1,y1);
xlabel("Frequency (Hz)");
ylabel("Amplitude");
title("Original signal");
grid();
xlim([-100,100]);


%filtering.....................................................
.....


fil_fre = 15;
delta1 = 0.001;
t=-2:delta1:(2-delta1);
w = 2*pi*fil_fre/3;
sinc_signal = sinc(w.*t);
filtered =
my_convolve(sample_signal,sinc_signal);
fs = 1000;
f2 = -fs/2:fs/7998:(fs/2);
y2=my_fftshift(abs(my_fft(filtered,7999)));
figure;
plot(f2,y2);
xlabel("Frequency (Hz)");
ylabel("Amplitude");
title("Filtered signal");
grid();
xlim([-100,100]);



%frequancy
scale.....................................................
scaled_signal = zero_arr(length(time_dom));
factor = 2;
for j=1:length(time_dom)
   scaled_signal(j) =
1.3+2*sin(factor*2*pi*frequancy(1)*time_do
m(j))+1.8*sin(factor*2*pi*frequancy(2)*time
_dom(j))+0.5*sin(factor*2*pi*frequancy(3)*ti
me_dom(j))+0.4*sin(factor*2*pi*frequancy(4)
*time_dom(j))+1.4*sin(factor*2*pi*frequancy
(5)*time_dom(j))+0.8*sin(factor*2*pi*frequa
ncy(6)*time_dom(j))+sin(factor*2*pi*frequan
cy(7)*time_dom(j))+sin(factor*2*pi*frequanc
y(8)*time_dom(j))+sin(factor*2*pi*frequancy
(9)*time_dom(j))+2*sin(factor*2*pi*frequanc
y(10)*time_dom(j));
end
```

```
figure;
fs = 1000;
f3 = -fs/2:fs/3999:(fs/2);
y3=my_fftshift(abs(my_fft(scaled_signal,4000
)));
plot(f3,y3);
xlabel("Frequency (Hz)");
ylabel("Amplitude");
title("Scaled signal");
grid();
xlim([-100,100]);

%frequency shift.....
fshift=50;
val = exp(1i*2*pi*fshift*time_dom);
shifted = val.*sample_signal;
figure;
fs = 1000;
f4 = -fs/2:fs/3999:(fs/2);
y4=abs(my_fft(shifted,4000));
p= y4((fshift)*4:(fshift+50)*4);
sz = zero_arr((450-fshift)*4);
mz = zero_arr((fshift-1)*8);
p1 = [ 0 0 0 sz flip(p) mz p sz 0 0 0];
plot(f4,p1,"g");
xlabel("Frequency (Hz)");
ylabel("Amplitude");
title("Shifted signal");
grid();
xlim([-100,100]);
```

```
%FFt
function.....

function z = my_fft(x,num_fft)
N = length(x);
z = zeros(1,num_fft);
sum=0;
for k = 1:num_fft
   for jj = 1:N
      sum = sum + x(jj)*exp(-2i*pi*(jj-1)*(k-
1)/num_fft);
   end
   z(k) = sum;
   sum = 0;
end
return
end

%my_fft_shift
function.....

function s = my_fftshift(vec)
l = length(vec);
if(mod(l,2)==0)
   s = [vec(l/2+1:l),vec(1:l/2)];
end
if(mod(l,2)~=0)
   s = [vec((l+1)/2+1:l),vec(1:(l+1)/2)];
end
return
end


%my_convolve
function.....

function Y= my_convolve(x, h)
```

```
m = length(h);

n = length(x);

Ny = m+n-1;

Y = zero_arr(Ny);

for i = 1:n

   for k = 1:m

      Y(i+k-1) = Y(i+k-1)+h(k)*x(i);

   end

end

return

end


%zeroes function
implementation.....

function z = zero_arr(n)

z=[0,0];

for g = 3:n

   z (g) = 0;

end

end
```

- **Appendix VI – AVR code**

```
#define F_CPU 16000000UL

//importing related libraries
#include <avr/io.h>
#include <util/delay.h>
#include <Arduino.h>
#include <SPI.h>
#include <SD.h>

//LCD
#define LCD_DATA PORTL //PORTL selected as
LCD DATA port
#define ctrl DDRL //PORTL selected as command
port

#define en PL1 //enable command port
#define rs PL0 //resistor select command port

//function declarations

void openfile(String filename);
void start_play_withfilter(String filename);
ISR(TIMER0_COMPA_vect );
void pitch_shift();
void start_play_pitch(String filename, byte factor);
ISR(TIMER1_COMPA_vect );
//////////////
void create_wav_format(String F_name);
void ADC_Setup();
void ADC_run_timer();
ISR(TIMER2_COMPA_vect );
void stop_ADC();
void do_conversion();
//////////////
void LCD_cmd(unsigned char cmd);
void init_LCD(void);
void LCD_write(unsigned char data);
void LCD_clear(void);
void LCD_Print (char *str);
void LCD_Printpos (char row, char pos, char *str);
//////File Man
void playFile(String fileName);
void stopPlay();
void startRecord(const char *fileName);
void stopRecord(const char *fileName);
void deleteFile(String fileName);
void readFile(String fileName);
void gallery();
int getCount(File dir);
void fileList(File dir);
void createFile(String fileName);
void writeLine(String fileName, String text);
void updateMenu();
void recordAction();
String generateName();
void mainAction();
void updateOptions( );
void opAction(String f_name);
void action1();
void action2();
void action3();


File data_file;
File file;

// initializing main parameters
int menu = 1;
int subMenu = 0;
int fileCount = 0;

//ADC variables
bool stat=0; //state changing variable
uint16_t ADC_out; //ADC result

//Header data
//sampling rate 8000
// bits per sample 8
char Header[44] = {/*RIFF
Section*/0x52,0x49,0x46,0x46,0x00,0x00,0x00,0x00,
0x57,0x41,0x56,0x45,/*Format
Section*/0x66,0x6D,0x74,0x20,0x10,0x00,0x00,0x00
,0x01,0x00,0x01,0x00,0x40,0x1F,0x00,0x00,0x40,0x
1F,0x00,0x00,0x01,0x00,0x08,0x00,/*Data
```

```
Header*/0x64,0x61,0x74,0x61,0x00,0x00,0x00,0x00
};
boolean is_rec;

#define cs_pin 53

//pitch_shift
byte pitch_factor=0;
short ocr1a_val;

//low pass filter
volatile int preval=0;

boolean which_effect;// select effect

int main() {
  DDRA = 0xFF; //set PORTA as output
  DDRF = DDRF & ~(1<<0); //set A0 as input
  DIDR0 = 0b00000001;



  //PUSH Buttons
  DDRC = DDRC & ~(1<<6); //OK button
  DDRC = DDRC & ~(1<<4); //BACK button
  DDRC = DDRC & ~(1<<7); //UP button
  DDRC = DDRC & ~(1<<5); //DOWN button
  PORTC = 0xF0; //enable pull up resistors
  if(!SD.begin(cs_pin)){//Initializing SD card
    while(1);//Try again
  }
  //start LCD
  init_LCD();
  //welcome message
  LCD_cmd(0xC2);
  LCD_Print(" WELCOME!!...");
  _delay_ms(300);
  LCD_clear();

  updateMenu();

  fileCount = getCount(SD.open("/")); //count the file
in SD

  //start_play_pitch("sith.wav",1);
  //start_play_withfilter("sith.wav");

  while(1) {
            if (~PINC & (1<<5))//check for DB press
            {
                        menu++;
                        updateMenu();
                        _delay_ms(100);
                        while (~PINC & (1<<5));
            }
            if (~PINC & (1<<7)) //check for UB press
            {
                        menu--;
                        updateMenu();
                        _delay_ms(100);
                        while (~PINC & (1<<7));
            }
            if (~PINC & (1<<6))  //check for SB press
            {
                        mainAction();
                        updateMenu();
                        _delay_ms(100);
                        while (~PINC & (1<<6));
            }

   }



}

//FileManager code----------------------------------------
--------------------

//playing a WAV file
void playFile(String fileName)
{
            start_play_pitch(fileName,0);
}

//stop playing a WAV file
void stopPlay()
{
            TIMSK0=0; //set counter to 0
            TIMSK1=0;
            PORTA=0; //set outputs to LOW
}

//record and save WAV file
void startRecord(const char *fileName)
{

            is_rec=true; //Turn ON ADC
            create_wav_format(fileName);

}

void stopRecord(const char *fileName)
{

            is_rec = false; //stop ADC
}

//delete a file
void deleteFile(String fileName)
{

            //checking for the file..
            if (SD.exists(fileName))
            {
                        SD.remove(fileName);

            }

}

//read a file
void readFile(String fileName)
{
```

```
		data_file = SD.open(fileName);
		if (data_file)
		{
			while (data_file.available())
			{
			}
			data_file.close();

		}
}

//Gallery
void gallery()
{
		File dir = SD.open("/");
		String fileNames[fileCount];
		dir.rewindDirectory();
		int index = 0;
		//getting the file name in to the fileNames
array
		while (true)
		{
			File entry =  dir.openNextFile();
			if (!entry)
			break;
			fileNames[index] = entry.name();
			entry.close();
			index++;
		}

			LCD_cmd(0xC0);//row 2
			int n1 = fileNames[0].length();
//convert string to char
			char zero[n1+1];
			strcpy(zero,fileNames[0].c_str());
			LCD_Print(zero);
			LCD_cmd(0x90); //row 3
			int n2 = fileNames[1].length();
			char one[n2+1];
			strcpy(one,fileNames[1].c_str());
			LCD_Print(one);
			LCD_cmd(0xD0); //row 4
			int n3 = fileNames[2].length();
			char two[n3+1];
			strcpy(two,fileNames[2].c_str());
			LCD_Print(two);

}

//calculate the file count
int getCount(File dir)
{
		int count = 0;

		while (true)
		{
			File entry =  dir.openNextFile();
			if (!entry)
			break;
			entry.close();
			count++;
		}
		return count;
}

// print files
void fileList(File dir)
{
		fileCount = getCount(dir); // get the file
count and update the global variable
		String fileNames[fileCount];
		dir.rewindDirectory();
		int index = 0;
		while (true)
		{
			File entry =  dir.openNextFile();
			if (!entry)
			break;
			fileNames[index] = entry.name();
			entry.close();
			index++;
		}
		String tempFile = "temp.txt";
		if (SD.exists(tempFile)) // for removing
irremovable temp files to avoid overwriting
		{SD.remove(tempFile);
		}
		createFile(tempFile);
		writeLine(tempFile,String(index));
		for (int i=0; i<index; i++)
		{
			writeLine(tempFile,fileNames[i]);
		}
}

//creating a file
void createFile(String fileName)
{
		data_file = SD.open(fileName,
FILE_WRITE);
		data_file.close();
}

//writing in a file
void writeLine(String fileName, String text)
{
		data_file = SD.open(fileName,
FILE_WRITE);
		if (data_file)
		{
			data_file.close();

		}
}

//Display code for LCD----------------------------------
----------
void updateMenu() {
		switch (menu) {
			case 0:
			menu = 3;
			LCD_clear();
			LCD_cmd(0x80);
			LCD_Print("Record");
			LCD_cmd(0xC0);
			LCD_Print("Gallery");
```

```
                LCD_cmd(0x90);
                LCD_Print(">About Device<");
                break;
                case 1:
                LCD_clear();
                LCD_cmd(0x80);
                LCD_Print(">Record<");
                LCD_cmd(0xC0);
                LCD_Print("Gallery");
                LCD_cmd(0x90);
                LCD_Print("About Device");

                break;
                case 2:
                LCD_clear();
                LCD_cmd(0x80);
                LCD_Print("Record");
                LCD_cmd(0xC0);
                LCD_Print(">Gallery<");
                LCD_cmd(0x90);
                LCD_Print("About Device");
                break;
                case 3:
                LCD_clear();
                LCD_cmd(0x80);
                LCD_Print("Record");
                LCD_cmd(0xC0);
                LCD_Print("Gallery");
                LCD_cmd(0x90);
                LCD_Print(">About Device<");
                break;
                case 4:
                menu = 1;
                LCD_clear();
                LCD_cmd(0x80);
                LCD_Print(">Record<");
                LCD_cmd(0xC0);
                LCD_Print("Gallery");
                LCD_cmd(0x90);
                LCD_Print("About Device");
                break;
        }
}

void recordAction()
{
        String tempString = generateName();
        int len = tempString.length() + 1;
        char recName[len];
        tempString.toCharArray(recName,len);
        LCD_clear();
        LCD_cmd(0x80);
        LCD_Print("  RECORDING... ");
        LCD_cmd(0xC0);
        LCD_Print(recName);
        LCD_cmd(0x90);
        LCD_Print("   Press OK to");
        LCD_cmd(0xD0);
        LCD_Print("      STOP ");

        startRecord(recName);  //start recording
        _delay_ms(200); // to stop jumping in to stop
recording when pressed OK to record from the record
sub menu
        stopRecord(recName);   //stop recording
        LCD_clear();
        LCD_cmd(0x80);
        LCD_Print("  RECORDING... ");
        LCD_cmd(0xC0);
        LCD_Print("    STOPPED!!   ");
        LCD_cmd(0x90);
        LCD_Print(" Press BACK for ");
        LCD_cmd(0xD0);
        LCD_Print("   Main menu   ");
        while(PINC & (1<<4)); //waiting for back
key press
}

// generate a suitable name
String generateName()
{
        String tempName;
        File dir = SD.open("/");
        String nameList[fileCount];
        int index = 0; // index for assigning strings to
the array
        //getting the file name in to the NameList
array
        while (true)
        {
                File entry =  dir.openNextFile();
                if (!entry)
                break;
                nameList[index] = entry.name(); //
assigning the names into the array
                entry.close();
                index++;
        }
        //searching for generate a name that doesn't
exist in SD
        int num = 1;
        bool found_name;
        while (true)
        {
                found_name = false;
                tempName = "REC_" +
String(num) + ".wav";
                for (int i=0; i<fileCount; i++)
                {

        if(nameList[i].equalsIgnoreCase(tempName)
)
                        {
                                found_name =
true;
                                break;
                        }

                }
                num++;
                if(!found_name)
                return tempName;
        }
}
```

```
void mainAction()
{
	switch (menu)
	{
		case 1:
		action1();
		break;
		case 2:
		action2();
		break;
		case 3:
		action3();
		break;
	}
}

//
int opMenu = 1;
void updateOptions( )
{
	switch (opMenu)
	{
		case 0:
		opMenu = 3;
		LCD_clear();
		LCD_cmd(0x80);
		LCD_Print(" -- OPTIONS -- ");
		LCD_cmd(0xC0);
		LCD_Print("Play");
		LCD_cmd(0x90);
		LCD_Print("Use_effect");
		LCD_cmd(0xD0);
		LCD_Print("> Delete");
		break;
		case 1:
		LCD_clear();
		LCD_cmd(0x80);
		LCD_Print(" -- OPTIONS -- ");
		LCD_cmd(0xC0);
		LCD_Print("> Play");
		LCD_cmd(0x90);
		LCD_Print("Use_effect");
		LCD_cmd(0xD0);
		LCD_Print("Delete");
		break;
		case 2:
		LCD_clear();
		LCD_cmd(0x80);
		LCD_Print(" -- OPTIONS -- ");
		LCD_cmd(0xC0);
		LCD_Print("Play");
		LCD_cmd(0x90);
		LCD_Print("> Use_effect");
		LCD_cmd(0xD0);
		LCD_Print("Delete");
		break;
		case 3:
		LCD_clear();
		LCD_cmd(0x80);
		LCD_Print(" -- OPTIONS -- ");
		LCD_cmd(0xC0);
		LCD_Print("Play");
		LCD_cmd(0x90);
		LCD_Print("Use_effect");
		LCD_cmd(0xD0);
		LCD_Print("> Delete");
		break;
		case 4:
		opMenu = 1;
		LCD_clear();
		LCD_cmd(0x80);
		LCD_Print(" -- OPTIONS -- ");
		LCD_cmd(0xC0);
		LCD_Print("> Play");
		LCD_cmd(0x90);
		LCD_Print("Use_effect");
		LCD_cmd(0xD0);
		LCD_Print("Delete");
		break;
	}
}

void opAction(String f_name)
{
	switch (opMenu)
	{
		case 1:
		{
		LCD_clear();
		LCD_cmd(0x80);
		LCD_Print("   Playing... ");
		LCD_cmd(0xC0);
		int n4 = f_name.length();
		char f_n_n4[n4+1];
		strcpy(f_n_n4,f_name.c_str());
		LCD_Print(f_n_n4);
		//playing..
		playFile(f_name); // call to play
function
		LCD_cmd(0x90);
		LCD_Print("  Press OK to");
		LCD_cmd(0xD0);
		LCD_Print("      STOP ");
		while(PINC & (1<<4)) // waiting
for back button
		{
			if ((~PINC & (1<<6)))
//OK button
			{
				//stop playing....
				stopPlay();
				LCD_clear();

	LCD_cmd(0x80);
				LCD_Print("
Stopped! ");

	LCD_cmd(0xC0);
				int n5 =
f_name.length();
				char
f_n_n5[n5+1];

	strcpy(f_n_n5,f_name.c_str());
```

```
		LCD_Print(f_n_n5);

		LCD_cmd(0x90);
					LCD_Print("
Press BACK for ");

		LCD_cmd(0xD0);
					LCD_Print("
Options   ");
					//while (PINC &
(1<<4));
				}

			}

			while (~PINC & (1<<4)); // to
avoid pressing the back button multiple times
			break;
			}
			case 2: //----------------------
Effects
			LCD_clear();
			LCD_cmd(0x80);
			LCD_Print(" -- Effects -- ");
			LCD_cmd(0x90);
			LCD_Print("UP-
PITCH_CHANGE");
			LCD_cmd(0xD0);
			LCD_Print("DOWN-
LP_FILTER");
			while(1){
				if(~PINC & (1<<7)){
					which_effect=1;
					break;
				}
				if(~PINC & (1<<5)){
					which_effect=0;
					break;
				}
			}

			if(which_effect){

		start_play_pitch(f_name,1);//playing with
effects..
				}else{

		start_play_withfilter(f_name);//play with
low pass filter
			}
			LCD_clear();
			LCD_cmd(0x80);
			LCD_Print("    Playing");
			LCD_cmd(0xC0);
			LCD_Print("      with");
			LCD_cmd(0x90);
			LCD_Print("    Effects...");
			LCD_cmd(0xD0);
			LCD_Print("TO STOP PRESS
OK");
			while(PINC & (1<<4)) // waiting
for back button
			{

				if (~PINC & (1<<6))
//OK button
				{
					stopPlay();
//stop playing....
					LCD_clear();

		LCD_cmd(0xC0);
					LCD_Print("
PRESS BACK");

		LCD_cmd(0x90);
					LCD_Print("
FOR OPTIONS");
				}

			}
			while (~PINC & (1<<4)); // to
avoid pressing the back button multiple times
			break;

			case 3:
			{
			LCD_clear();
			LCD_cmd(0x80);
			LCD_Print("  Deleted! ");
			LCD_cmd(0xC0);
			int n6 = f_name.length();
			char f_n_n6[n6+1];
			strcpy(f_n_n6,f_name.c_str());
			LCD_Print(f_n_n6);
			deleteFile(f_name);//deleting..
			LCD_cmd(0x90);
			LCD_Print(" Press BACK for ");
			LCD_cmd(0xD0);
			LCD_Print("   Main menu   ");
			while(PINC & (1<<4));// waiting
for back button
			_delay_ms(300);
			break;
			}
		}
}

void action1()
{
		LCD_clear();
		LCD_cmd(0x80);
		LCD_Print("  > Record <  ");
		LCD_cmd(0xC0);
		LCD_Print(" Press OK to ");
		LCD_cmd(0x90);
		LCD_Print("    start    ");
		_delay_ms(300); // stop jump in to recording
when pressed OK to enter to the record sub menu
		while(1)
		{
			if (~PINC & (1<<6))
			{
```

```
                        recordAction();
                        return;
                }
                else if (~PINC & (1<<4)) //what
should do if press back
                return;
        }

}

// Gallery Menu
void action2()
{
        const char row_pos[4] =
{0x80,0xC0,0x90,0xD0}; //Array with LCD row ID
        File dir = SD.open("/");
        fileCount = getCount(dir); // get the file
count and update the global variable
        String fileNames[fileCount];
        dir.rewindDirectory();
        int index = 0; // index for assigning strings to
the array
        //getting the file name in to the fileNames
array
        while (true)
        {
                File entry =  dir.openNextFile();
                if (!entry)
                break;
                fileNames[index] = entry.name();
// assigning the names into the array
                entry.close();
                index++;
        }
        //update the gallery menu
        int pointerLine = 1;
        int x = 0; //variable for scroll the file list
        int displayIndex;
        int upCount = 0;
        int downCount = 2;

        LCD_clear();
        LCD_cmd(0x80);
        LCD_Print(" --- Gallery --- ");
        /*
        int n1 = fileNames[0].length(); // code
snippet to convert string to char
        char zero[n1+1];
        strcpy(zero,fileNames[0].c_str());
        */
        for (int i=0;i<3;i++)
        {
                LCD_cmd(row_pos[i+1]);
                if(pointerLine==i+1){
                int v1 = fileNames[i].length();
                char ch_v1[v1+3];
                strcpy(ch_v1,(">>" +
fileNames[i]).c_str());
                LCD_Print(ch_v1);
                }
                else{
                        int v11 =
fileNames[i].length();
                        char ch_v11[v11+1];

        strcpy(ch_v11,(fileNames[i]).c_str());
                        LCD_Print(ch_v11);
                }
        }

        while(PINC & (1<<4)) // waiting for back
button
        {

                // going down on the list
                if (~PINC & (1<<5))
                {

                        if (upCount < 2)
                        upCount++;
                        if (pointerLine >=
fileCount) // to connect the bottom of the menu to the
top
                        {
                                pointerLine = 0;
                                x = 0;
                        }
                        pointerLine++;
                        if (pointerLine > 3 &&
downCount == 0)
                        x++;

                        if (downCount > 0)
                        downCount--;
                        //update the gallery menu
                        LCD_clear();
                        LCD_cmd(0x80);
                        LCD_Print(" --- Gallery -
-- ");
                        displayIndex = 1;  // only
used for identify the printing line
                        for (int i=x;i<3+x;i++)
                        {

        LCD_cmd(row_pos[displayIndex]);
                                if (pointerLine
== i+1){
                                        int v2
= fileNames[i].length();
                                        char
ch_v2[v2+3];

        strcpy(ch_v2,(">>" + fileNames[i]).c_str());

        LCD_Print(ch_v2);
                                }
                                else{
                                        int v22
= fileNames[i].length();
                                        char
ch_v22[v22+1];

        strcpy(ch_v22,(fileNames[i]).c_str());

        LCD_Print(ch_v22);
                                }
```

```
                displayIndex++;
            }

            while (~PINC & (1<<5));
// to avoid auto pressing the button
        }

        // going up on the list
        if (~PINC & (1<<7))
        {


            if (downCount < 2)
            downCount++;
            if (upCount == 0) //&&
pointerLine < fileCount - 3
            x--;
            if (upCount > 0)
            upCount--;
            if (pointerLine == 1) // to
connect the bottom of the menu to the top
            {
                pointerLine =
fileCount+1;
                x = fileCount -
3;
                upCount += 2;
            }
            pointerLine--;
            //update the gallery menu
            LCD_clear();
            LCD_cmd(0x80);
            LCD_Print(" --- Gallery -
-- ");
            displayIndex = 1;  // only
used for identify the printing line
            for (int i=x;i<3+x;i++)
            {

    LCD_cmd(row_pos[displayIndex]);
                if (pointerLine
== i+1){
                    int v3
= fileNames[i].length();
                    char
ch_v3[v3+3];

    strcpy(ch_v3,(">>" + fileNames[i]).c_str());

    LCD_Print(ch_v3);
                }
                else{
                    int v33
= fileNames[i].length();
                    char
ch_v33[v33+1];

    strcpy(ch_v33,(fileNames[i]).c_str());

    LCD_Print(ch_v33);
                }
                displayIndex++;
            }
                while (~PINC & (1<<7));
// to avoid auto pressing the button
            }

            // OK button for Gallery
            _delay_ms(300);
            if (~PINC & (1<<6))
            {
                updateOptions();
                while (~PINC & (1<<6));
// to avoid pressing the back button multiple times
                // gallery control
                while(PINC & (1<<4)) //
waiting for back button
                {
                    if (~PINC &
(1<<5))
                    {

    opMenu++;

    updateOptions();

    _delay_ms(100);
                        while
(~PINC & (1<<5));
                    }
                    if (~PINC &
(1<<7))
                    {

    opMenu--;

    updateOptions();

    delay(100);
                        while
(~PINC & (1<<7));
                    }
                    //Select button
for Options Menu
                    if (~PINC &
(1<<6))
                    {

                        while
(~PINC & (1<<6));  // to avoid pressing the back
button multiple times

    opAction(fileNames[pointerLine-1]);  // pass
the selected file name as the variable
                        if
(opMenu == 3)

    {break;

    } // stop updating options menu

    updateOptions();

                    }
                }
                if (opMenu == 3)
```

```
                        {Serial.println("Going to
Main menu after deleting");break;} //to reupdate the
gallery after deleting a file

                        while (~PINC & (1<<4));
// to avoid pressing the back button multiple times
                        Serial.println("Back from
options");
                        opMenu = 1; //reset the
options menu
                        //update the gallery menu
after pressing back
                        LCD_clear();
                        LCD_cmd(0x80);
                        LCD_Print(" --- Gallery -
-- ");
                        displayIndex = 1;  // only
used for identify the printing line
                        for (int i=x;i<3+x;i++)
                        {

        LCD_cmd(row_pos[displayIndex]);
                                if (pointerLine
== i+1){
                                        int v4
= fileNames[i].length();
                                        char
ch_v4[v4+3];

        strcpy(ch_v4,(">>" + fileNames[i]).c_str());

        LCD_Print(ch_v4);
                                }
                                else{
                                        int v44
= fileNames[i].length();
                                        char
ch_v44[v44+1];

        strcpy(ch_v44,(fileNames[i]).c_str());

        LCD_Print(ch_v44);
                                }
                                displayIndex++;
                        }
                        while (~PINC & (1<<6));
// to avoid auto pressing the button
                }

                //End of waiting block..
        }
}


void action3()
{

        LCD_clear();
        LCD_cmd(0x80);
        LCD_Print("> About Device <");
        LCD_cmd(0xC0);
        LCD_Print(" Voice Recorder");
        LCD_cmd(0x90);
        LCD_Print("      v1.0      ");
        LCD_cmd(0xD0);
        LCD_Print(" We are Group 21");

        while(PINC & (1<<4));

}

//effects code----------------------------------------------
--------------
void openfile(String filename){
  data_file = SD.open(filename,FILE_READ);
  data_file.seek(60); //pass the header
}

void start_play_withfilter(String filename){
 openfile(filename);
 noInterrupts();// disable interrupts
 TCCR0B = (1<<WGM02); // CTC Mode
 TCCR0B = (1 << CS01); // Set preScaler to divide by
8
 TIMSK0 = (1 << OCIE0A); // Call ISR when TCNT0
= OCRA0
 OCR0A = 150; // set sample play rate to 8000Hz_124
 interrupts(); // Enable interrupts to generate
waveform!

}

ISR(TIMER0_COMPA_vect) { // Called when
TCNT0 == OCR0A
  if(data_file.available()){
    int v = (int)data_file.read();
    PORTA = 0.3*preval + 0.3*(v);//write data to
PORTA using leaky integrator
    preval = v;
  }else{
  PORTA=0;
  data_file.close();
  TIMSK0=0;
  }
    TCNT0 = 0;//set counter again to zero
}

void pitch_shift(){ // function for pitch shifting
  switch(pitch_factor){
    case 4 : ocr1a_val=250; break;
    case 3 : ocr1a_val=200; break;
    case 0 : ocr1a_val=170; break;
    case 1 : ocr1a_val=100; break;
    case 2 : ocr1a_val=90; break;
  }
}

void start_play_pitch(String filename,byte factor){
 pitch_factor=factor;
 openfile(filename);
 pitch_shift();//take the pitch factor
 noInterrupts();// disable interrupts
 TCCR1B = (1<<WGM12); // CTC Mode
 TCCR1B = (1 << CS11); // Set preScaler to divide by
8
```

```
 TIMSK1 = (1 << OCIE1A); // Call ISR when TCNT1
= OCRA1
 OCR1A = ocr1a_val; // set sample play rate to
8000Hz_124
 interrupts(); // Enable interrupts to generate
waveform!

}

ISR(TIMER1_COMPA_vect) { // Called when
TCNT1 == OCR1A
  if(data_file.available()){
    PORTA= data_file.read();//write data to PORTA
  }else{
  PORTA=0;
  data_file.close();
  TIMSK1=0;
  }
    TCNT1 = 0;//set counter again to zero
}

//Code related to LCD----------------------------------------
--------------
void init_LCD(void){
        DDRL = 0b11111111;
        _delay_ms(20);
        LCD_cmd(0x02); //initializing 4 bit mode of
20x4 LCD
        LCD_cmd(0x28);
        LCD_cmd(0x0c);
        LCD_cmd(0x06); // make increment in
cursor
        LCD_cmd(0x01); // clear LCD
        _delay_ms(2);
        //initialization complete
}

//function to send commands
void LCD_cmd(unsigned char cmd){
        LCD_DATA = (LCD_DATA & 0x0F) |
(cmd & 0xF0); //last 4 bits
        LCD_DATA &= ~ (1<<rs); //set for
commands
        LCD_DATA |= (1<<en);
        _delay_us(1);
        LCD_DATA &= ~ (1<<en);
        _delay_us(200);
        LCD_DATA = (LCD_DATA & 0x0F) |
(cmd << 4); // first 4 bits of the cmd
        LCD_DATA |= (1<<en);
        _delay_us(1);
        LCD_DATA &= ~ (1<<en);
        _delay_ms(2);
        return;

}

//function to send data
void LCD_write(unsigned char data){
        LCD_DATA = (LCD_DATA & 0x0F) |
(data & 0xF0); //last 4 bits
        LCD_DATA |= (1<<rs); //set for data
        LCD_DATA |= (1<<en);
        _delay_us(1);
        LCD_DATA &= ~ (1<<en);
        _delay_us(200);
        LCD_DATA = (LCD_DATA & 0x0F) |
(data << 4); // first 4 bits of the cmd
        LCD_DATA |= (1<<en);
        _delay_us(1);
        LCD_DATA &= ~ (1<<en);
        _delay_ms(2);
        return;
}

void LCD_clear(void){
        LCD_cmd (0x01);         //Clear LCD
        _delay_ms(2);                   //Wait
to clean LCD
        LCD_cmd (0x80);         //Move to
Position Line 1, Position 1

}

//print the string on display
void LCD_Print (char *str)
{
        int i;
        for(i=0; str[i]!=0; i++)// loop to print the
string
        {
                LCD_DATA = (LCD_DATA &
0x0F) | (str[i] & 0xF0); //last 4 bits
                LCD_DATA |= (1<<rs);
                LCD_DATA|= (1<<en);
                _delay_us(1);
                LCD_DATA &= ~ (1<<en);
                _delay_us(200);
                LCD_DATA = (LCD_DATA &
0x0F) | (str[i] << 4); //first 4 bits
                LCD_DATA |= (1<<en);
                _delay_us(1);
                LCD_DATA &= ~ (1<<en);
                _delay_ms(2);
        }
}

//Write on a specific location
void LCD_Printpos (char row, char pos, char *str)
{
        if (row == 0 && pos<16)
        LCD_cmd((pos & 0x0F)|0x80);
        else if (row == 1 && pos<16)
        LCD_cmd((pos & 0x0F)|0xC0);
        else if (row==2 && pos<16)
        LCD_cmd((pos&0x0F)|0x94);
        else if(row==3 && pos<16)
        LCD_cmd((pos & 0x0F)|0xD4);
        LCD_Print(str);

}


//for the recording and saving-------------------------------
---------------------------
```

```
//creating_WAV_file_format

void create_wav_format(String F_name)
{
	uint32_t Samples=0; //initialize sample
count
	file = SD.open(F_name,FILE_WRITE);
//opening the file in write mode

	//First write the header
	for(uint8_t pos=0; pos<44; pos++){
		file.write(Header[pos]);
	}
	//record from ADC
	ADC_Setup();
	ADC_run_timer();
	_delay_ms(200);//to stop double press
	while((PINC & (1<<6))){ //&
digitalRead(selectButton)
		_delay_us(1);
		if(stat){
			Samples += 1;
			do_conversion();
			stat=0;
			file.write(ADC_out);
		}
	}
	stop_ADC();

	file.close();
	//completing the header
	file = SD.open(F_name,O_RDWR);
	file.seek(40);// go to data part size position

	for(uint8_t pos=0; pos<4; pos++){
		file.write( ( Samples>>(8*pos) ) &
0xFF);
	}

	file.seek(4); //wave file size position

	for(uint8_t pos=0; pos<4; pos++){
		file.write( ( (Samples+44)
>>(8*pos) ) & 0xFF);
	}

	file.close();
	Samples =0;
}

void ADC_Setup(void){
	noInterrupts();
	ADMUX = 0b01100000; //REF--
>AVCC,Left adjusted, A0
	ADCSRA = 0b10000110; //Enable,
PreScale-->64-->250kHz

	interrupts();

}

void ADC_run_timer(void){
	noInterrupts();// disable interrupts
	TCCR2B = (1<<WGM21); // CTC Mode
	TCCR2B = (1 << CS21); // Set PreScaler to
divide by 8
	TIMSK2 = (1 << OCIE2A); // Call ISR
when TCNT2 = OCRA2
	OCR2A = 240; // set sample taking rate to
8000Hz
	interrupts(); // Enable interrupts
}

ISR(TIMER2_COMPA_vect){
	stat=1;//set ADC to get the sample
	TCNT2 = 0;
}

void stop_ADC(void){
	noInterrupts();// disable interrupts
	TIMSK2 = 0x00;
	ADCSRA = 0x00;
}

void do_conversion(void){
	noInterrupts();
	ADCSRA = ADCSRA | (1<<ADSC); //set
single conversion
	//polling
	while(ADCSRA & (1 << ADSC)); //wait till
the conversion is done
	interrupts();
	ADC_out = ADCH; //output read from data
register

	ADCSRA |= 1 << ADIF; //clearing the ADC
Interrupt Flag
}
```