Department of Electronics and Telecommunication Engineering
Faculty of Engineering
University of Moratuwa

EN1093 - Laboratory Practice - I

Group Project: Simple Voice Recorder

Group members	Index Number
1. Dassanayaka D. M. S. S.	190115P
2. De Silva W. A. D. K.	190128H
3. Dilshan J. V. A. P.	190144D
4. Diron A. G.	190149X

Due Date of Submission
16 / 08 /2021

Abstract

The main goal of this project was to record and play back an audio with enhancements using a simple voice recorder. ADC, DAC, and audio processing techniques together with 8-bit PCM were used to achieve this requirement. User Interface, input block, output block and the processing block are the four main parts of the project. In the final evaluation a voice signal was recorded and played back successfully in the proteus simulation.

CONTENT

1. Introduction……………………………………………………………………………...1
2. Functionality & method………………………………………………………...............1
2.1. Components Used………………………………...1
2.1.1. ATmega328P………………………………………………………………...1
2.1.2. DAC0832 DIY module kits…………………………………………………1
2.1.3. Sound sensor module………………………………………………………..2
2.1.4. 7805A regulator……………………………………………………………...2
2.1.5. SD card module……………………………………………………………2
2.1.6. 1602 LCD Module with I2C Module (Blue)…………………………………2
3. Hardware development…………………………………………………………...............2
3.1. Prototype Description………………………………………………………….2
3.2. Schematics and PCB layout……………………………………………………3
4. Software development……………………………………………………………………...3
4.1. PCM Theory……………………………………………………………………3
4.2. Application of PCM Theories…………………………………………………..3
4.2.1. Recording Audio……………………………………………………………..3
4.2.2. Play Back Audio…………………………………………………………….,3
4.3. AVR Platform……………………………………………………………………,.3
4.4. Audio enhancement techniques description, Matlab simulation…………………3
4.4.1. Bass and Treble effect………………………………………………………..3
4.4.2. Echo effect…………………………………………………………………...3
5. Enclosure design……………………………………………………………………….....4
6. Result……………………………………………………………………………………..4
7. Discussion………………………………………………………………………………...5
7.1. Components……………………………………………………………………….5
7.2. Problems Faced in Sampling, Storage, Playback…………………………………5
7.2.1. Sampling……………………………………………………………………..5
7.2.2. Storage……………………………………………………………………….5
7.2.3. Playback……………………………………………………………………...5
7.3. PCB Designing…………………………………………………………………….6
7.4. Enclosure Design………………………………………………………………….6
8. Acknowledgment………………………………………………………………………...6
9. References……………………………………………………………………………….6
10. Appendices7
10.1. Simulation Circuit………………………………………………………………..7
10.2. Schematic Diagrams……………………………………………………………...7
10.3. PCB Layouts……………………………………………………………………...8
10.4. Flow Chart……………………………………………………………………….9
10.5. Arduino Code……………………………………………………………………10
10.6. Matlab Code……………………………………………………………………..11
10.7. Enclosure Design………………………………………………………………..12

1. INTRODUCTION

The Simple Voice Recorder can be used to record, store, and playback audio as required. An ATmega328P microcontroller was used to process the data and control the functions of the voice recorder. Apart from that, an SD card module, an amplifier module, a DAC 0832LCN IC, an LCD panel and push buttons are the main components of the project used for the tasks of storing audio files, amplifying output audio, interactions with user and digital to analog conversion respectively.

2. FUNCTIONALITY & METHOD

2.1 Components Used

- ATmega328P
- DAC0832 DIY module kits
- Sound sensor module
- 7805A regulator
- SD card module
- 1602 LCD Display 16×2 (Blue)
- I^{2}C Module for 1602 1604 2004 LCD
- PAM8403 Module
- 8 Ω speaker
- 16 MHz crystal oscillator
- Capacitors - 22 pF,0.1 µF, 0.22 µF
- Resistors - 10 kΩ, 470 Ω
- Push buttons
- 2 GB microSD card

2.1.1 ATmega328P

The purpose of the project requires a simple, low-power, and low-cost microcontroller. It is more practical to use an ATmega328P chip as per the requirements of the project. A 16 MHz external crystal oscillator was connected with the microcontroller to generate the clock signal needed to mount the data. The ATmega chip is the most implemented chip on the Arduino development platform.

Figure 1: ATmega328P

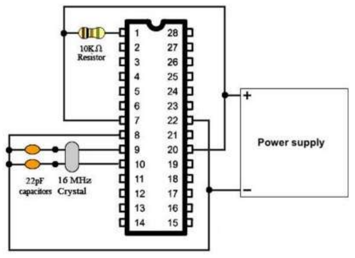


Figure 2: ATmega328P connected with crystal oscillator

2.1.2 DAC0832 DIY module kits

This module includes 2 ICs.

- DAC0832LCN
- LM324N

Module size: $4\ cm \times 4.9\ cm$

Pins on the right:

- OUT2: bipolar output port
- OUT1: unipolar output port

Pins on the left:

- CS: Chip Select, received a single-chip IO, active low.
- WR: write, received microcontroller IO, active low.
- D0-D7: 8-bit parallel data port, received a single-chip IO.
- GND: GND Module

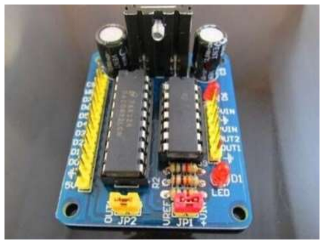


Figure 3: DAC0832 DIY Kits

2.1.3 Sound sensor module

No of pins: 3 (VCC, GND, OUT)
Dimension: $36.5\,mm \times 17\,mm \times 10\,mm$
Weight: 4 g (approx.)
Package contains: 1 × LM393 Sound Detection Sensor Module.

Figure 4: Sound sensor module

2.1.4 7805A regulator

The circuit is powered by a 9V battery. The 7805A regulator is used to reduce the voltage to 5V.

Figure 5: 7805A Voltage regulator

2.1.5 SD card module

Features:

- Supports Micro SD cards, Micro SDHC card (high speed card)
- Power supply is 4.5 V ~ 5.5 V
- Standard communication interface - SPI (Serial Peripheral Interface)
- 3.3 V voltage regulator circuit board
- 4 M2 screw positioning holes for easy installation

Control Interface: A total of six pins (GND, VCC, MISO, MOSI, SCK, CS)

Figure 6: SD card module

2.1.6 1602 LCD Module with I^{2}C Module (Blue)

A 16×2 LCD module soldered with an I^{2}C connector was used since there were not enough pins available to connect the LCD directly with the ATmega328P chip. By using the I^{2}C interface the number of pins used to control the LCD was limited to two.

I^{2}C connector pins (GND, VCC, SDA, SCL)

Figure 7: LCD Module with I^{2}C Module

3. HARDWARE DEVELOPMENT

3.1 Prototype Description

The sound sensor was used to get the input voice signal. After sampling and quantizing the recorded analog voice, it is encoded using 8-bit PCM techniques and stored in the SD card as a text file(.txt). These encoded samples are then processed using the ATmega328P microcontroller to add effects. The digital to analog conversion is done by DAC0832LCN IC which contains 8 input pins (D0 - D7) and gives the reconstructed analog signal as the output. Later the signal is amplified using PAM8403 module and passed down to the speaker.

3.2 Schematics and PCB layout

Altium Designer (version 21.3.1) was used to design PCB for the project. Altium Designer is a powerful tool for Schematic and PCB layout designing.

In the Schematic drawing instead of connecting all wires, they were labelled such that the schematic will not be complicated and will be easy to understand.

In the PCB layout 10 mils width is used for normal traces and 15 mils width is used for power traces.

4. SOFTWARE DEVELOPMENT

4.1 PCM Theory

In finding the sampling rate *Nyquist's Sampling Theorem* was used.
Human voice contains frequency components in the range 0 – 4 kHz.
According to the theory, the sampling rate should be greater than or equal to twice the maximum frequency component present in the voice.

$$Sampling\ rate\ \geq\ (4\ kHz \times 2 = 8\ kHz)$$

Hence the sampling rate should be 8000 Hz or higher.
In the project 9.2 kHz is used as the sampling frequency.
8-bit PCM is used to encode the samples.

4.2 Application of PCM Theories

4.2.1 Recording Audio

Using the ADC of Arduino Uno, samples were obtained in the required sampling rate with 10-bit generic resolution of the Arduino Uno.
Then to convert the 10-bit value into an 8-bit value, integer division by 4 is applied.
Since 8–bit PCM has a total of 256 levels while 10–bit PCM has total 1024 levels.

$$256 \times 4 = 1024$$

Hence integer division by 4 is applied.
Resulting values are stored in a text file *(.txt).*

4.2.2 Play Back Audio

In playback, the same number of samples as in the recording stage is read in one second which is equal to 9200 samples.
The sample values are fed into an DAC0832LCN IC to obtain the original signal.

4.3 AVR Platform

The LCD navigation and analog to digital conversion were implemented in AVR platform using Microchip Studio. For the navigation of the LCD a separate I^{2}C library and an LCD library are used.

4.4 Audio enhancement techniques description, Matlab simulation

Bass, treble, and echo effects were applied as audio enhancement techniques. None of the inbuilt functions of Matlab were used when adding these effects to ensure that the same techniques can be implemented in Arduino IDE. The effects were clearly distinguishable from the original audio.

4.4.1 Bass and Treble effect

Implementing the bass and treble effects were possible using FIR filtering techniques in Matlab. The spectrum of the recorded sound and its spectrum with bass and treble effects obtained from Matlab is shown below.

4.4.2 Echo effect

According to the project guidelines only one audio enhancement effect was required. The echo effect is an additional audio enhancement technique. The effect was added as a time domain process.

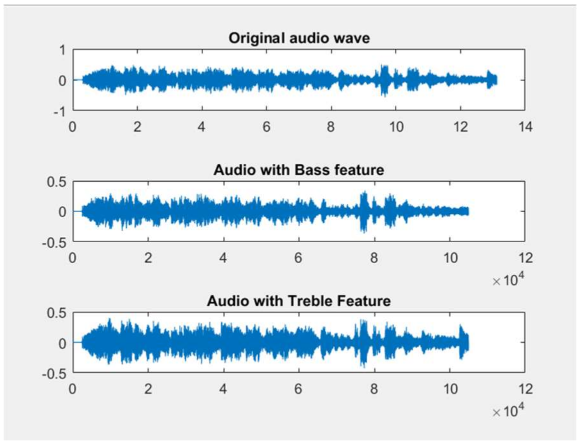


Figure 8: Time domain representation of bass and treble effects

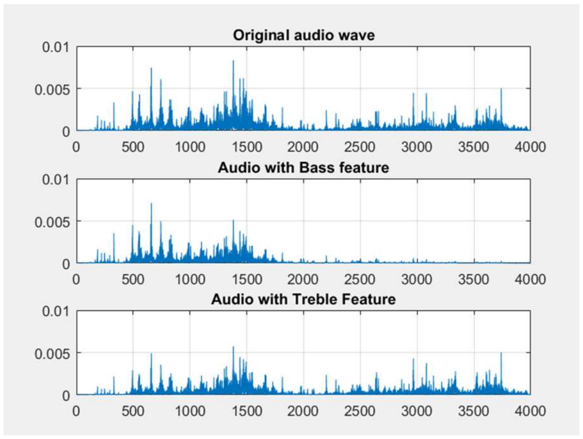


Figure 9: Frequency domain representation of bass and treble effects

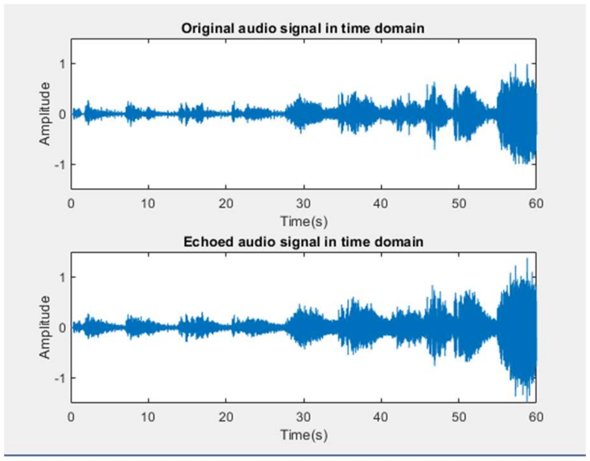


Figure 10: Time domain representation of echo effect

5. ENCLOSURE DESIGN

When designing the enclosure for the voice recorder “SolidWorks®” 3D modeling software was used. The PCB was exported from Altium to the assembly. Blue polished ABS plastic is used as the material for the enclosure. Toggle switches are used for both the power button and the “Record” button. Push buttons are used for “Select”, “Back”, “Up” and “Down”. The LCD display is attached to the front cover of the voice recorder using screws. The PCB and the sound sensor module are mounted onto the PCB placeholder surface using mounting bosses. The vent applied at the bottom of the enclosure, is aligned with the microphone of the sound sensor to ensure proper function of the sensor system as it requires rapid communication with the surrounding environment. Holes at the back cover that are aligned with the speaker allow the sound waves to pass through. The 9 V battery holder is placed on the PCB placeholder surface.

6. RESULT

As shown in the Figure 11 a recorded audio was successfully reconstructed and played back in the Proteus simulation. The navigation system was also implemented successfully.

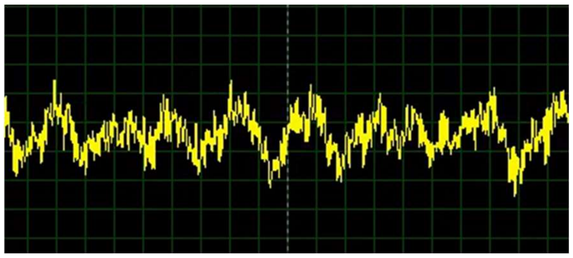


Figure 11: Input waveform

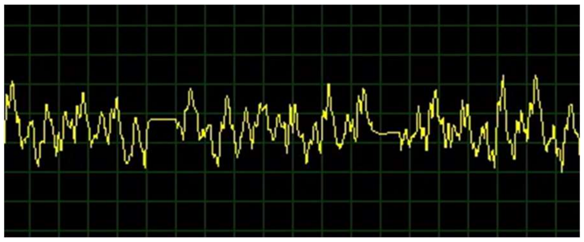


Figure 12: Output waveform

7. DISCUSSION

7.1 Components

Due to the Covid - 19 restrictions, a suitable microphone module could not be found for the project. As an alternative a sound sensor was used.

7.2 Problems Faced in Sampling, Storage, Playback.

7.2.1 Sampling

ADC of the Arduino Uno is used for sampling.

At the beginning, the sampling rate was below 1000 Hz. It seemed to be impossible to increase the sampling rate at first.

After using *mills()* and *micros()* functions of Arduino to measure the time taken for *analogRead()* and *map()* it was observed that default time taken for one cycle of ADC conversion took more time than the time taken for other functions.

As the solution, the prescale of the ADC clock was reduced to 2. Accuracy of ADC was not affected as 10-bit PCM was used as the generic resolution of the ADC.

This contributed greatly to reduce the time for one sampling cycle.

As the *map()* function also took more than 40 microseconds to execute, integer division by 4 was applied.

$$8 - bit\ PCM\ sample = \frac{10 - bit\ PCM\ sample}{4}$$

These major modifications helped to reduce the overall time taken for sampling effectively.

7.2.2 Storage

8–bit PCM values are written to a text file *(.txt).*

The time taken for writing to the file had to be reduced.

All possible functions for writing to a file were tested.

Following are the three major functions used to write a file object.

File.println(SAMPLE VALUE)
File.print(SAMPLE VALUE)
File.write(SAMPLE VALUE)

When time was measured for each function, it was discovered that the *write()* function takes the least time, but the *write()* function behaves in an unpredictable way when an integer value or a string variable is given as the input.

Out of the other two functions, *print()* is used in the code as it takes less time to execute than *println()*.

This way the time taken for one cycle of sampling and storing was reduced to a value less than 125 µs. Take time for one sampling and storing cycle as T.

Hence,

$$sampling\ rate = \frac{1}{T}$$

$$T < 125\,\mu s$$

$$Sampling\ Rate = \frac{1}{T} > 8000\,Hz$$

7.2.3 Playback

Seeking the possibility of a function to read back the values stored in the data file at a rate equal to the sampling rate was the major issue.

After some trials, it was discovered that the *read(buffer, sizeof(buffer))* function can be used to read back the values at a rate that is even greater than the sampling rate used.

Then another problem emerged.

How to add a suitable delay to slow down the above function?

As the solution, the following function was introduced to calculate the required delay.

D = Delay in microseconds

T_1 = Recording time

F = Average sampling rate

T_2 = Average *read()* function execution time for the file we recorded without any delay

$$D = \frac{T_1}{F \times T_1} - \frac{T_2}{F \times T_1}$$

$$D = \frac{1}{F} - \frac{T_2}{F \times T_1}$$

Thus, we overcome the problem.
Finally, audio was recorded and read back at the same sampling rate of 9.2 kHz. Hence a clear audio signal was heard when playing back.

7.3 PCB Designing

There were difficulties in finding components and footprints in Altium. Later libraries were downloaded from the internet to get the schematic and footprints of them. 3D bodies for some of the components could not be found.

Placing of the components and the routing was the most challenging part in the whole PCB designing process. By collaborating with the design of the enclosure the space allocated for the PCB was utilized in an effective way.

It can be further enhanced such that the area of the PCB can be reduced and most of the routing can be done in the top layer.

7.4 Enclosure Design

The enclosure is designed to make it easy for the user to handle the device. The push buttons made of rubber for navigating purposes are placed in a dial configuration to enhance the user experience.

8. ACKNOWLEDGMENT

We would like to express our sincere gratitude to our supervisors Mr. Bhanuka Malith Silva and Mr. Vinu Maddumage for guiding us all along from the beginning till the completion of the project. They were always supporting us and regularly checking our progress since the beginning. Apart from them, we are grateful for other instructors who shared their knowledge and experience with us.

Software Used

- Altium Designer® 2021
- SolidWorks® 2020
- Proteus 8 Professional
- Microchip Studio 7.0
- Arduino IDE

9. REFERENCES

- https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

- https://microcontrollerslab.com/dac0832-8-bit-dac-pinout-example-applications-features/

- Signals and Systems,
  Alan V. Oppenheim , Alan S. Willsky,
  Massachusetts Institute of Technology.

- https://forum.arduino.cc/

10. APPENDICES

10.1 Simulation Circuit

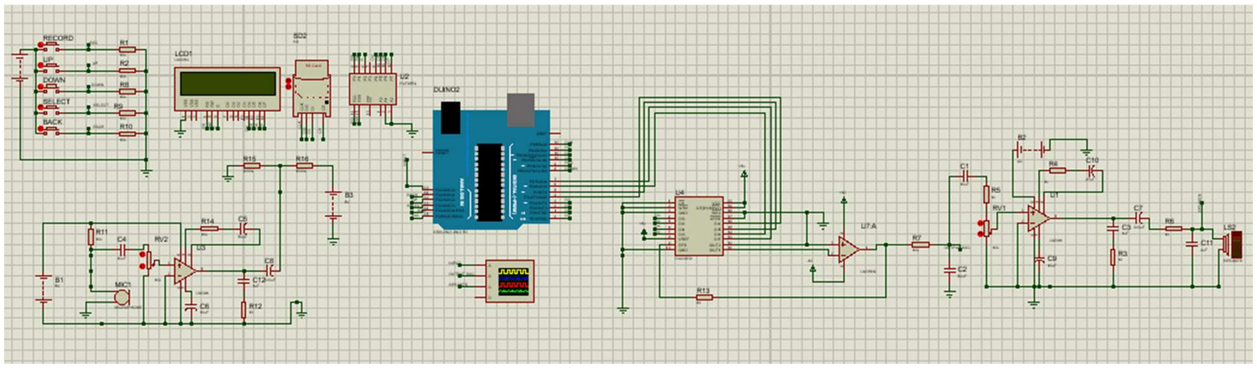


Figure 13: Simulation circuit

10.2 Schematic Diagrams

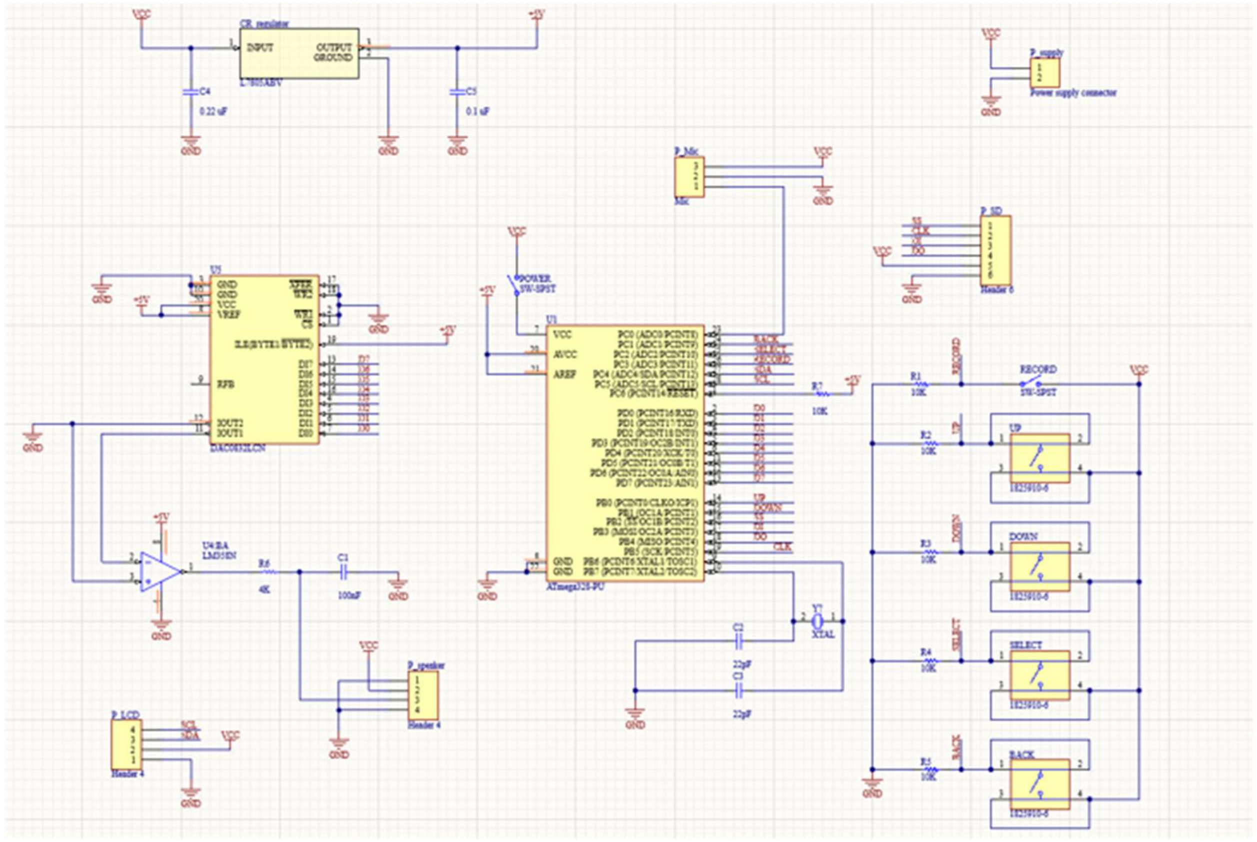


Figure 14: Schematic diagram

10.3 PCB Layouts

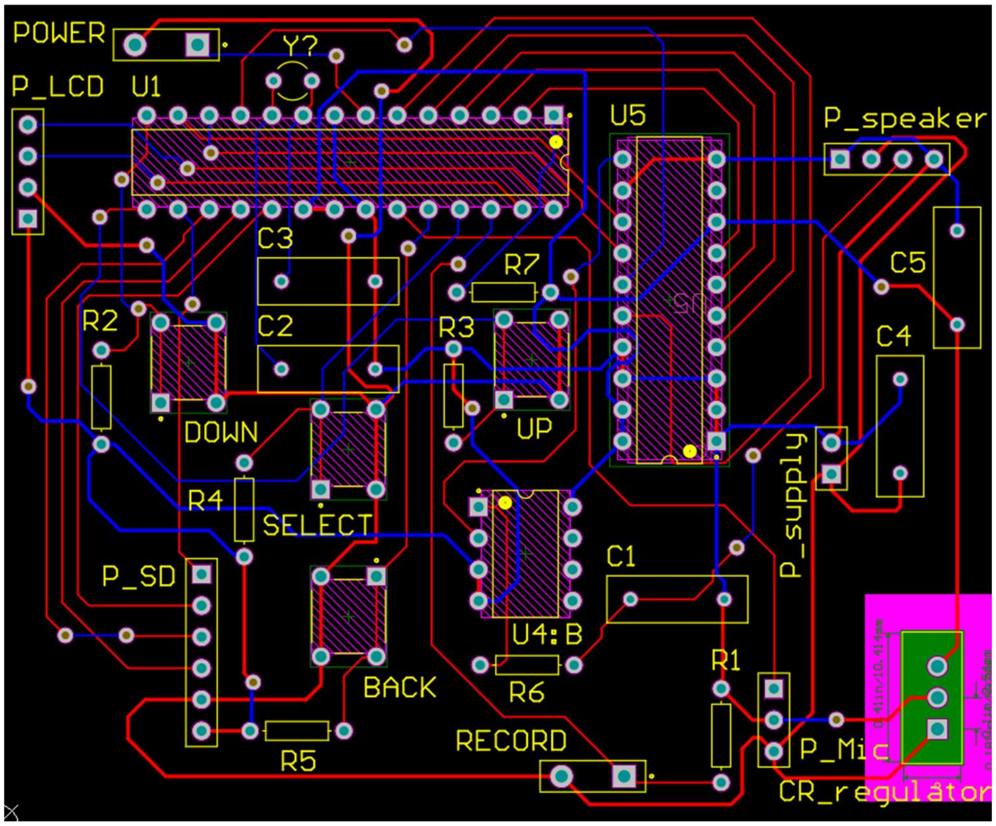


Figure 15: PCB layout

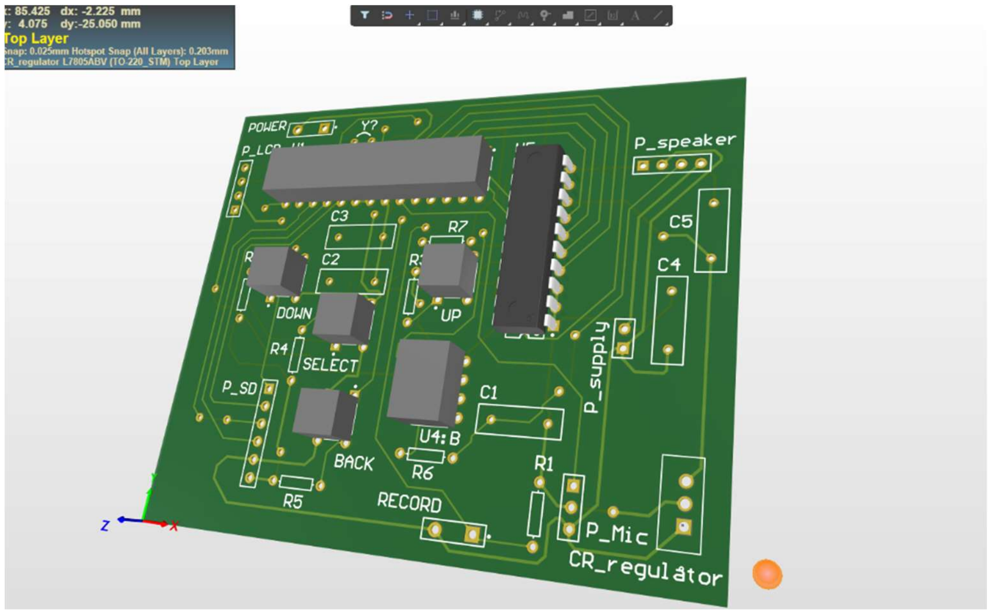


Figure 16: 3D view of PCB

10.4 Flow Chart

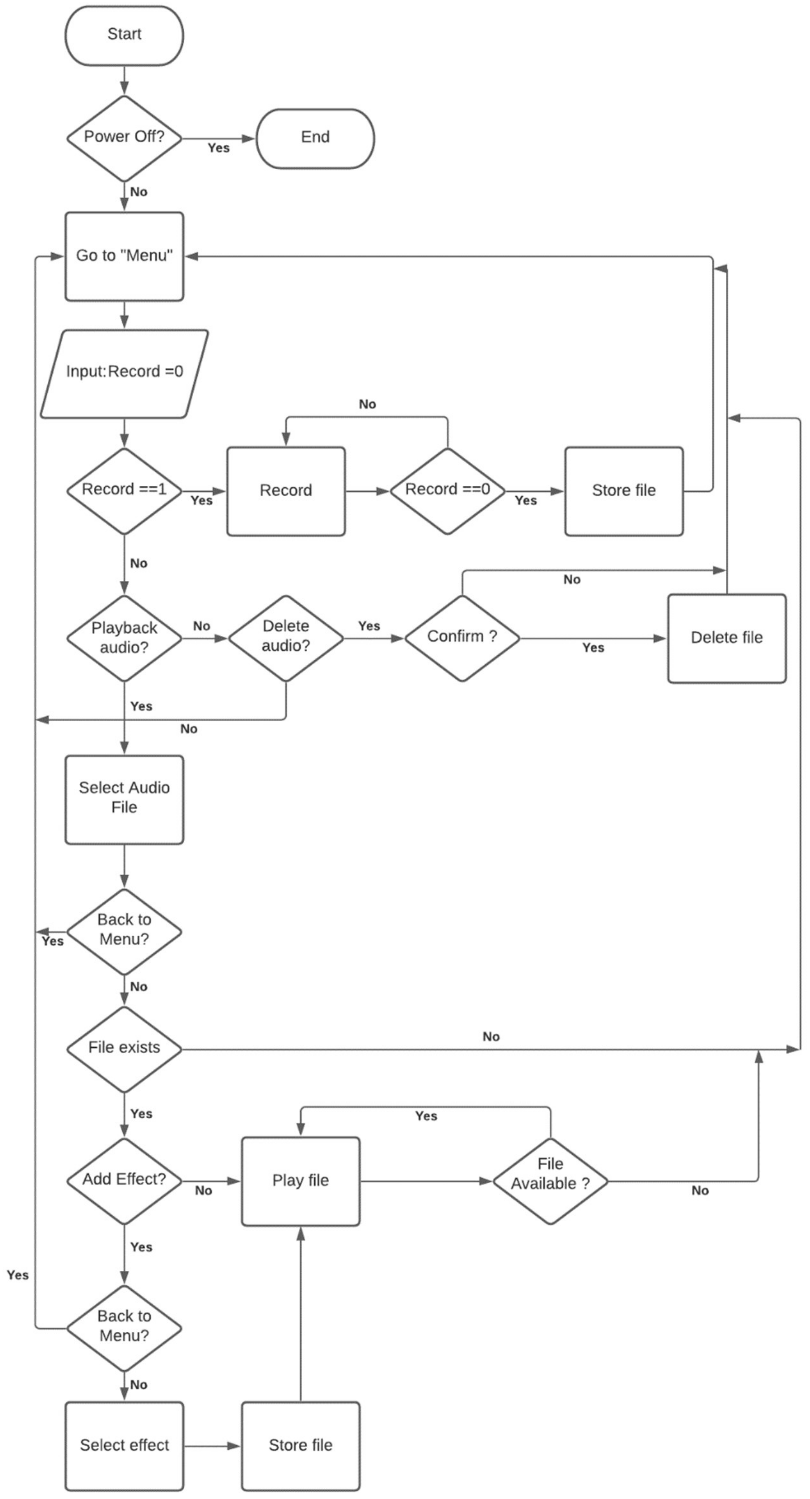
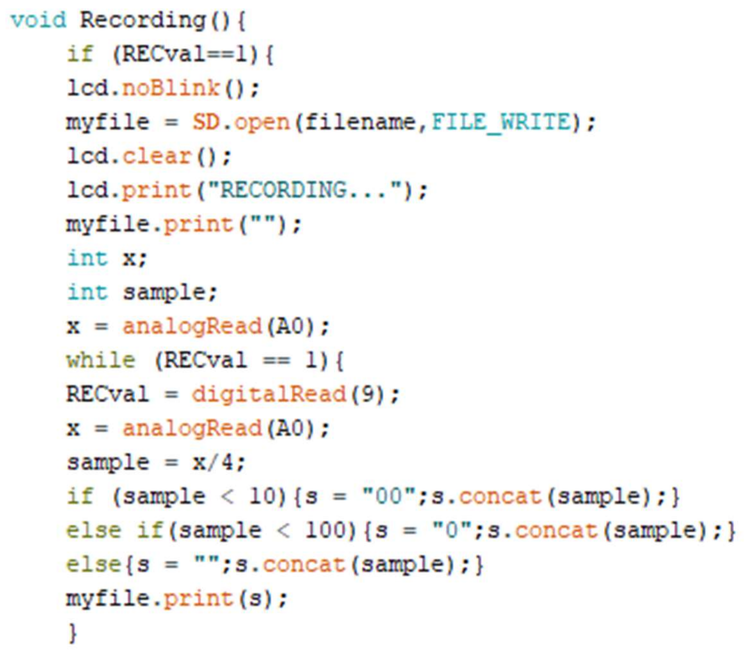


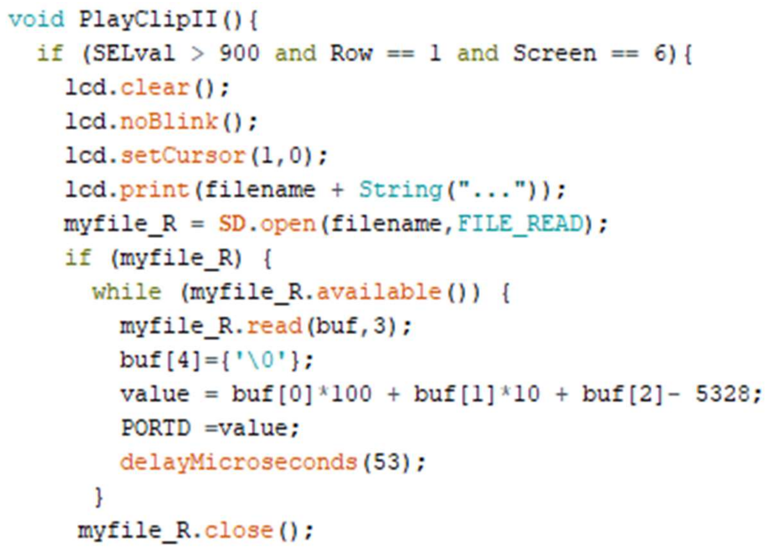
Figure 17: Flow chart

10.5 Arduino Code



```
void Recording(){
    if (RECval==1){
    lcd.noBlink();
    myfile = SD.open(filename,FILE_WRITE);
    lcd.clear();
    lcd.print("RECORDING...");
    myfile.print("");
    int x;
    int sample;
    x = analogRead(A0);
    while (RECval == 1){
    RECval = digitalRead(9);
    x = analogRead(A0);
    sample = x/4;
    if (sample < 10){s = "00";s.concat(sample);}
    else if(sample < 100){s = "0";s.concat(sample);}
    else{s = "";s.concat(sample);}
    myfile.print(s);
    }
```

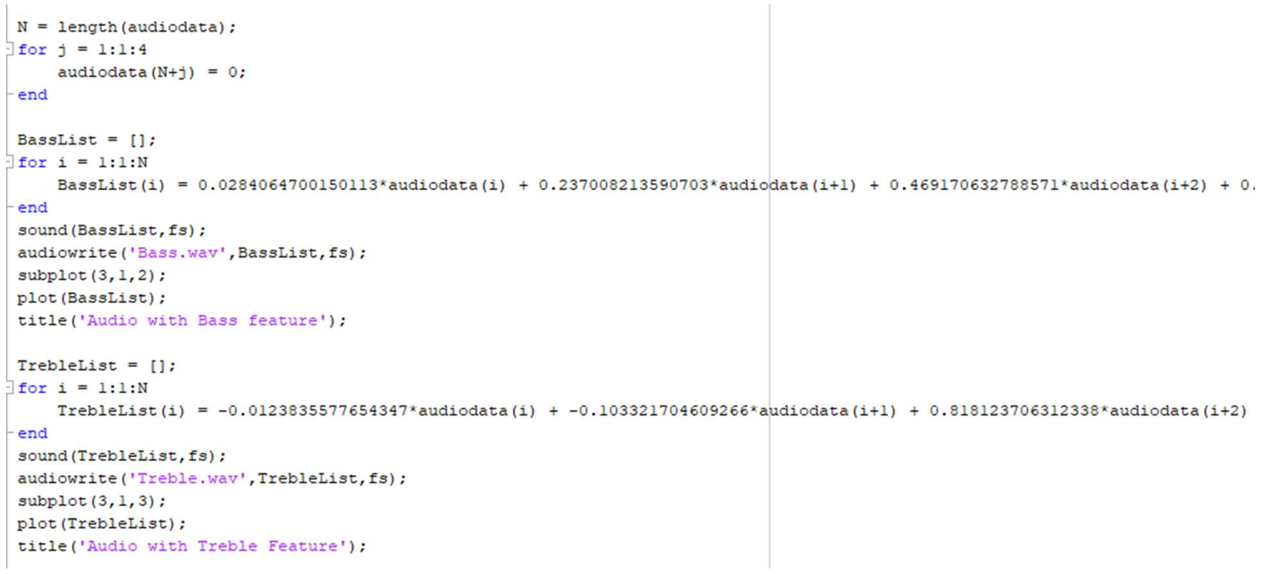
Figure 18: Arduino code for ADC and storing



```
void PlayClipII(){
  if (SELval > 900 and Row == 1 and Screen == 6){
    lcd.clear();
    lcd.noBlink();
    lcd.setCursor(1,0);
    lcd.print(filename + String("..."));
    myfile_R = SD.open(filename,FILE_READ);
    if (myfile_R) {
      while (myfile_R.available()) {
        myfile_R.read(buf,3);
        buf[4]={'\0'};
        value = buf[0]*100 + buf[1]*10 + buf[2]- 5328;
        PORTD =value;
        delayMicroseconds(53);
      }
     myfile_R.close();
```

Figure 19: Arduino code for reconstructing the signal

10.6 Matlab Code

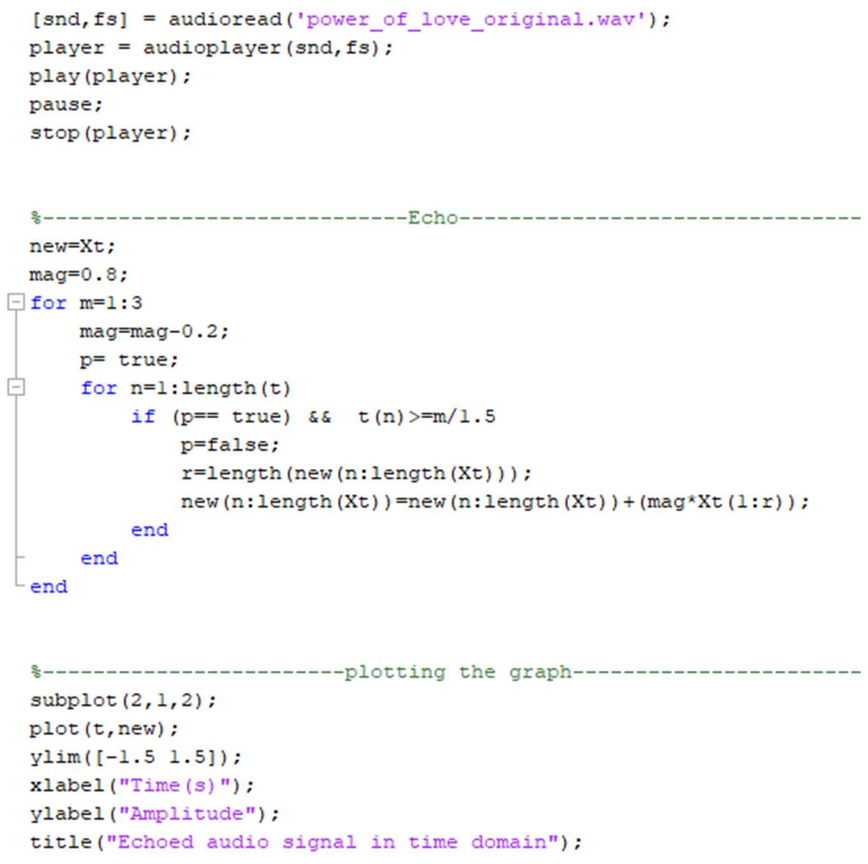


```
N = length(audiodata);
for j = 1:1:4
    audiodata(N+j) = 0;
end

BassList = [];
for i = 1:1:N
    BassList(i) = 0.0284064700150113*audiodata(i) + 0.237008213590703*audiodata(i+1) + 0.469170632788571*audiodata(i+2) + 0.
end
sound(BassList,fs);
audiowrite('Bass.wav',BassList,fs);
subplot(3,1,2);
plot(BassList);
title('Audio with Bass feature');

TrebleList = [];
for i = 1:1:N
    TrebleList(i) = -0.0123835577654347*audiodata(i) + -0.103321704609266*audiodata(i+1) + 0.818123706312338*audiodata(i+2)
end
sound(TrebleList,fs);
audiowrite('Treble.wav',TrebleList,fs);
subplot(3,1,3);
plot(TrebleList);
title('Audio with Treble Feature');
```

Figure 20: Matlab code for Bass & Treble



```
[snd,fs] = audioread('power_of_love_original.wav');
player = audioplayer(snd,fs);
play(player);
pause;
stop(player);


%------------------------------Echo--------------------------------
new=Xt;
mag=0.8;
for m=1:3
    mag=mag-0.2;
    p= true;
    for n=1:length(t)
        if (p== true) &&  t(n)>=m/1.5
            p=false;
            r=length(new(n:length(Xt)));
            new(n:length(Xt))=new(n:length(Xt))+(mag*Xt(1:r));
        end
    end
end


%------------------------plotting the graph-----------------------
subplot(2,1,2);
plot(t,new);
ylim([-1.5 1.5]);
xlabel("Time(s)");
ylabel("Amplitude");
title("Echoed audio signal in time domain");
```

Figure 21: Matlab code for echo effect

10.7 Enclosure Design

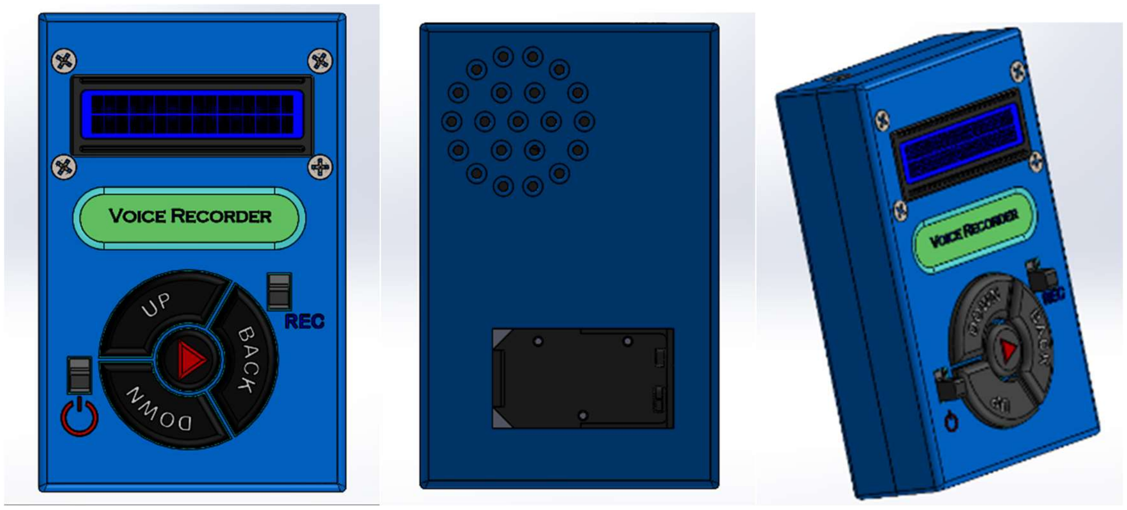


Figure 22: Front, back, side views

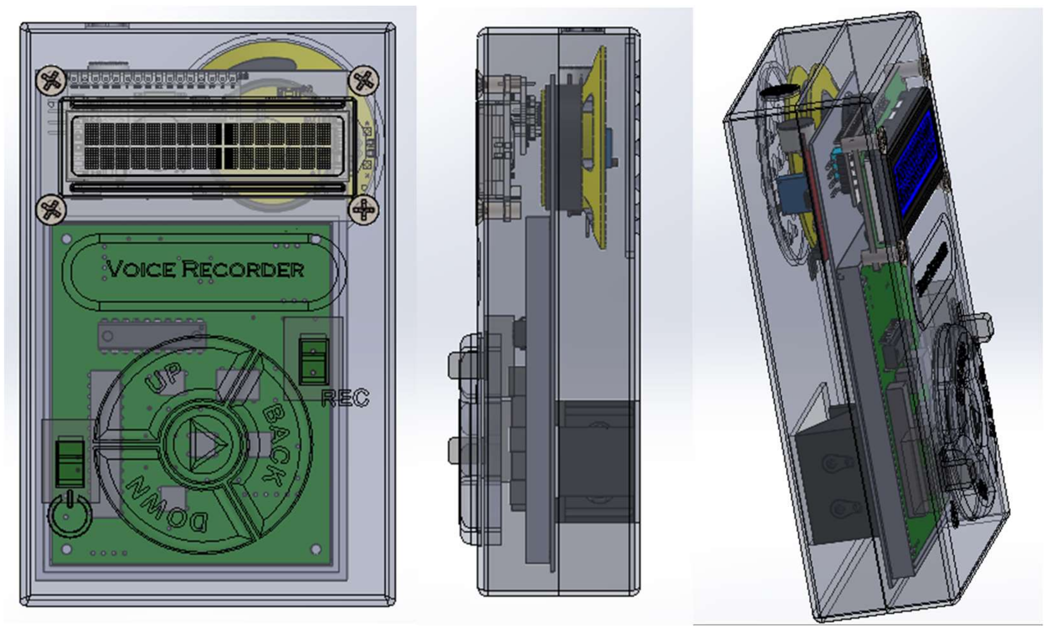


Figure 23: Transparent views