# Simple Voice Recorder

Wanshika W.A.R. (190663R), Wansooriya W.M.H.O. (190664V), Weerasinghe K.N.(190672T), Weerasinghe M.S. (190673X), Wickrama W.M.T.B. (190680P)

EN1093 Laboratory Practice – I

Department of Electronic and Telecommunication Engineering, University of Moratuwa

**Abstract: This report deals with how to design a simple voice recorder capable of simple voice modulation. In this report, we discuss the information about how recordings are done in a recorder, how voice is sampled, quantized and how those encoded bitstreams are saved in an SD card, how multiple voice recordings are saved in an SD card, and how we can selectively playback each recording and how to do minor adjustments during the playback. This paper also includes the implementation of the Atmega 328p Atmel microprocessor, which is used to manage multiple data and commands, how PCM and PWM techniques are used to record and play audios. As a simple voice recorder is a commercial product, this paper focuses on how to oversee both recording and playback in a user-friendly manner by using a user-attractive interface (a menu). At the end of the report, it includes the PCB design and enclosure design of the final product of the laboratory practice – I project (Semester - 02).**

**Key words:** Modulation · Encode · PCM · Record · Playback · User Attractive Interface · PCB · Enclosure

## Introduction

A Voice recorder is equipment that can record sounds in the environment, store them on an SD card and play them using a speaker. It also can add some audio enhancement methods like low pass filter and high pass filter to the audio file.

The project was initially made based on Arduino Uno and then converted into Atmel-based microcontrollers using MicroC. There were also 2 LEDs to indicate the recording process. The final product was shown in a computer-generated PCB model and an enclosure compacted with the structure. The harder part of this project was handling the SD card in which we had to use a library called TMRPCM to do the biddings.

## Method

### 2.1. Modulation Technique

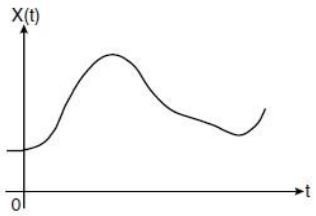
#### 2.1.2. PCM

PCM (pulse code modulation) is a digital modulation technique that is used to convert an analog signal to digital data and represent the amplitude of the signal in binary form approximately at the sampling instant.

### 2.1.3. Basic elements of PCM

- Low pass filter [1]

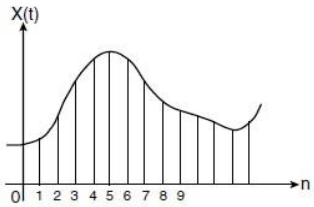
The most audio signal contains information between 0 to 4kHz in the frequency spectrum. But due to the noise, this value may change and adversely affect the input audio. Therefore, we have to include a low pass filter to eliminates high-frequency components that are greater than the highest frequency of the input audio signal, to avoid aliasing of the input audio signal.




[1]

- Sampler [2]

Then the analog signal is converted into the discrete value signal by taking samples periodically. Time interval is decided according to the sampling theorem which is sampling frequency is at least twice the max frequency of the signal, otherwise due to the periodic continuation in frequency domain signal will be overlapped (alias) and cannot be recovered original signal at the output.
In this project, we are dealing with frequency 0 to 4000 Hz for audios. Therefore, we used an 8000Hz sampling frequency to get the samples, and counting is done by timer 1 in the TMRPCM library when getting the samples.

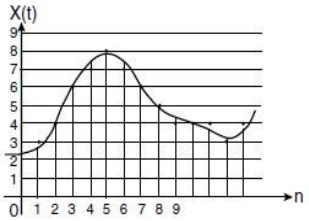



[2]

- Quantizer [3]

At this stage, code determines the digital value of the analog signal samples taken at each sampling point. This digital value is called resolution. Quantitation of this input signal always results in loss of information is called quantization error. And this error is inversely proportional to the range of resolution values.
In this project, we are using an Atmega 328p microcontroller, which gets samples at 8000Hz and quantized the signal by 8-bit resolution which means divide the signal into 256 levels. We used the TMRPCM library to get input. This library converts 10-bit resolution which is microcontroller analog pin resolution into 8-bit resolution by using the "AnlogResolution" specified coding term.

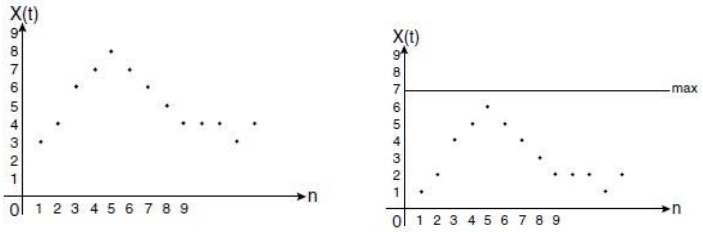
$$\text{AnlogResolution} = \frac{\text{Initial value of the sample}}{1024} \times 256$$




[3]

- Encoder [4]

  Analog signal input is digitalized at the encoder. All the quantized values are designated by binary code. Sampling done in the encoder is the sample-and-hold process. This process minimizes the bandwidth used in the process.




[4]

After those stages, encoded data is saved on the SD card. In this project, we used the TMRPCM library to do all these stages by giving sampling frequency (8000Hz) and the name of the audio by using the specified word "audio.startRecording" and "audio.stopRecording".[5]

```
void record() {
    while (i < 300000) {
      i++;
    }
    i = 0;
    if (recmode == 0) {
      recmode = 1;
      audiofile++;
      switch (audiofile) {
        case 1: audio.startRecording("1.wav", 8000, A0,1);digitalWrite(8, HIGH); recmode1 = 1; lcd.clear();lcd.print("Recording 1.Wav"); break;
        case 2: audio.startRecording("2.wav", 8000, A0,1);digitalWrite(8, HIGH); recmode1 = 1; lcd.clear();lcd.print("Recording 2.Wav"); break;
        case 3: audio.startRecording("3.wav", 8000, A0,1);digitalWrite(8, HIGH); recmode1 = 1; lcd.clear();lcd.print("Recording 3.Wav"); break;
      }
    }
    else {
      recmode = 0;
      switch (audiofile) {
        case 1: audio.stopRecording("1.wav");digitalWrite(8, LOW); lcd.clear(); recmode1 = 0; updateMenu(); break;
        case 2: audio.stopRecording("2.wav");digitalWrite(8, LOW); lcd.clear(); recmode1 = 0; updateMenu(); break;
        case 3: audio.stopRecording("3.wav");digitalWrite(8, LOW); lcd.clear(); recmode1 = 0; updateMenu(); break;
      }
    }
}
```

[5]

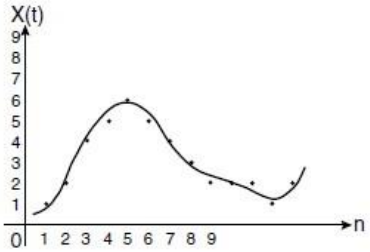
### 2.1.4. Playback technique

After the save of recordings in the SD card we have to use the playback technique to play each of the audio. Rather than using a DAC, we used a microcontroller 9 audio output PWM pin for a 328p microcontroller to get the audio output of the signal.

PWM (pulse width modulation) is also a modulation technique we used to convert saved binary digital signal to analog. When we select to play in the menu microcontroller starts the playback if we selected pause its stops playback or it is stopped after all recorded data is played in each audio.

In the PWM technique, it is done by the cycle method. On cycle have counted up and counting down to its maximum(255 mostly in our case) and minimum level (0 mostly). When the counter matches the saved value in a given sample point, the PWM counter will be switched on. It is switched off when the counter falls below this value.[6] Likewise, the output 9 th pin reconstructs the voice signal by linking and smoothen the signal which is almost the same as the recorded audio.[7]


[6]

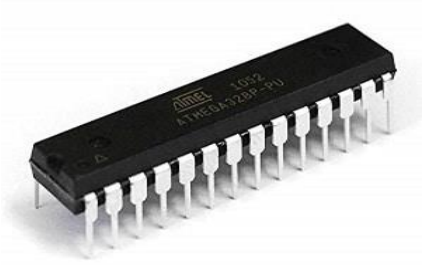



[7]

## 2.2. Components used

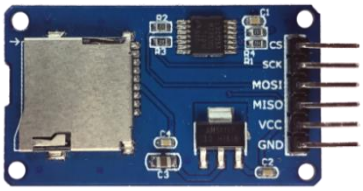
### 2.2.1. Main modules

1. Microcontroller
   Atmega 328p AVR Microcontroller



- Atmega 328p is AVR RISC architecture based 8-bit microcontroller. This is used for Arduino uno board and therefore this is the most popular microcontroller in AVR family.
- We used this micro controller to handle command and data in our voice recorder.

2. SD card module




- Supports Micro SDHC card(high speed card) and Micro SD cards.
- Level conversion circuit board that interfaces level for 5V or 3.3V.
- Needs 4.5V ~ 5V or 3.3 V regulated voltage for power up the circuit.
- Use SPI communication interface for communication with micro controller.
- Can easily installed by using 4 M2 screw positions.

- We used SD card to save recording as a wav file and call them when the user wants to hear audios.

3. Microphone module


- This module used Electret condenser microphone with LM393 as the main chip.
- Need power supply of DC 3.3-5V.
- 0.5 m induction distance helps to detect sound within 0.5m range.
- Signal analog output Indication.
- Single channel to output signal
- The output of the effective signal is in a low level.

- We used microphone module to get input to the Microcontroller as a digital value by doing PCM techniques.

4. Amplifier module

We used PAM8406 Digital Amplifier module With Volume Potentiometer 5Wx2 Stereo.


- 5V recommended voltage need but can handle with 2.5V – 5.5V.
- 2-8 Ohms Speaker load resistance.
- Amplifier has 0.2" centers pin for output and input connections.
- Size: (105 x 24 x 23) mm (H x L x W).

- We used this as an output signal amplifier and also as a low pass filter to convert digital value to analog signal and to remove unwanted noise frequencies.

5. Power supply


- This is an adjustable buck LM2596 switching regulator and this module capable for driving a 3A with excellent load and line regulation. We can adjust output voltage by adjusting the potentiometer. This inbuilt potentiometer has 25 turns and therefore, we can easily adjust the voltage exactly what we need.

- We used this module to get 5V power supply to the circuit.

6. LCD Display with I2C connection


- $I_2C$-LCD display, only using two I/O-Pins.
- It includes $I_2C$ PCF8574T-Driver.
- Pins – VCC, GND, SCL, SDA
- Standard 16 Chars, 2 Lines LCD
- Potentiometer for Contrast
- Size - 36mm x 80mm
- Height (approximately) : 20mm (with the backpack)

- We connect this module to the microcontroller by using SCL and SDA ports.(A4 and A5)
- There are two operating mode.
    - Command mode – giving commands for operate.
    - Data mode – giving data to display in lcd module.

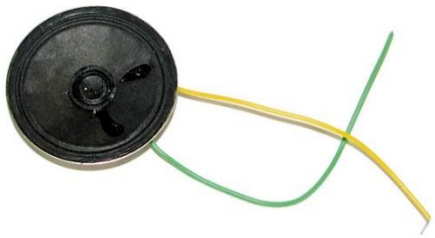
- There are 2 data transfer modes.

    - 8-bit mode – data is taken by 8-bit length data stream.
    - 4- bit mode – data is taken by 4-bit length data stream. This is used for saving space for circuit.( we used our project )

- We used 4-bit data transfer module to do Read, Write commands and data transfer actions.

### 2.2.2. Other components

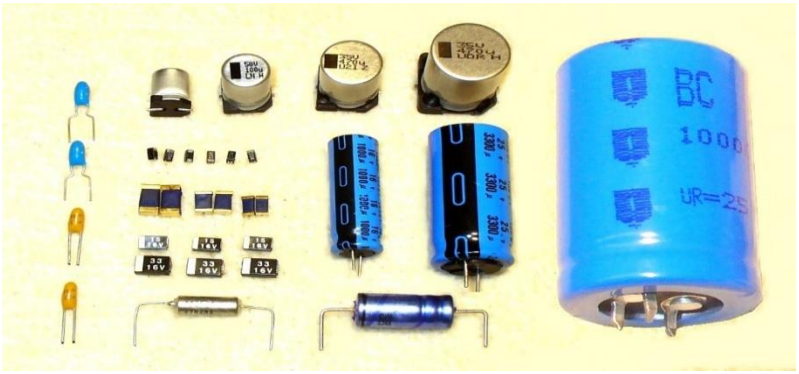
- Speaker – To get the output analog signal.



- Push buttons – To select up, down, select, back, record and effect options
  To handle the menu

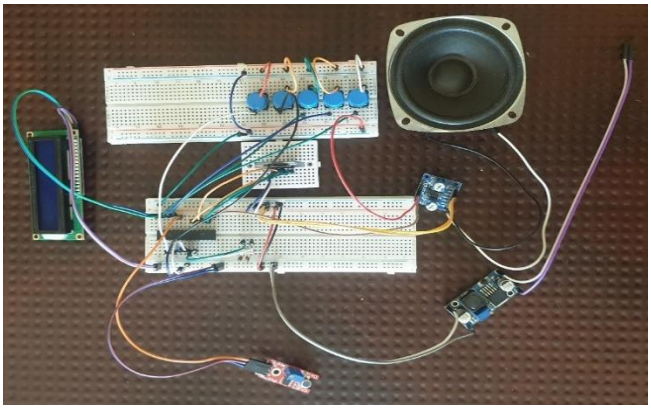

- 16MHz crystal oscillator – we used oscillator as an external timer for the micro-controller rather than using internal oscillator.

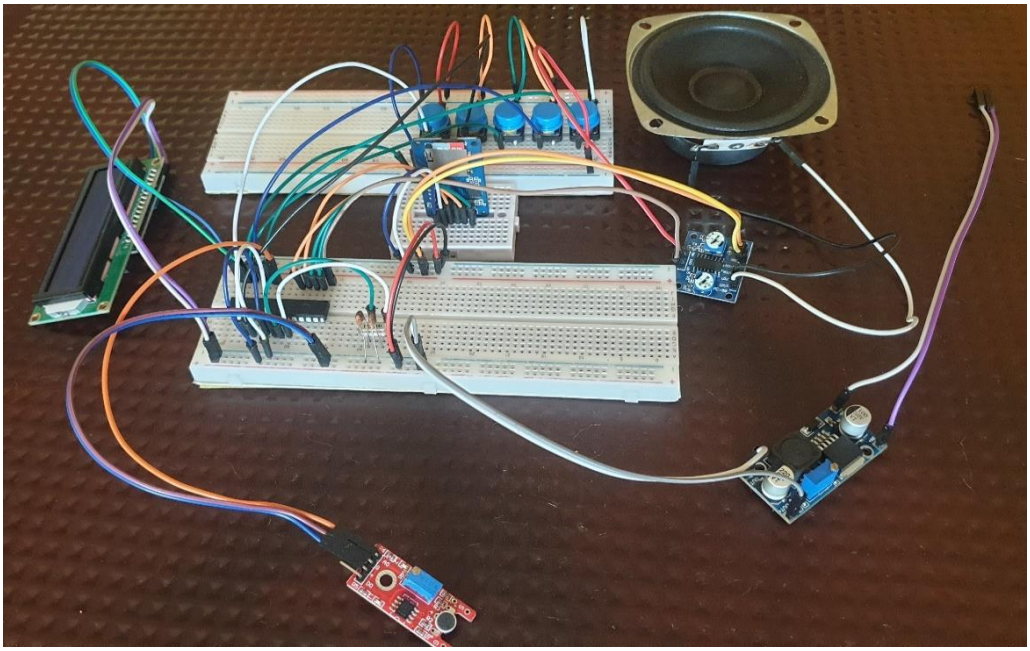
- 2 x 22pf capacitor

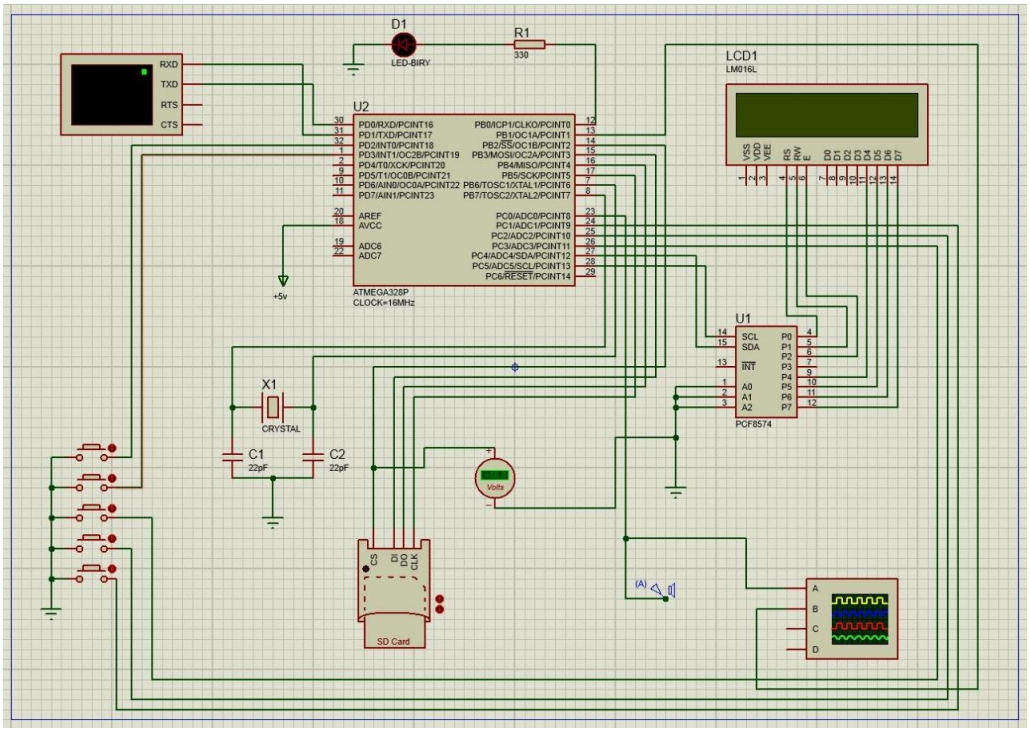



- Resistors

## 2.3. Prototype design results



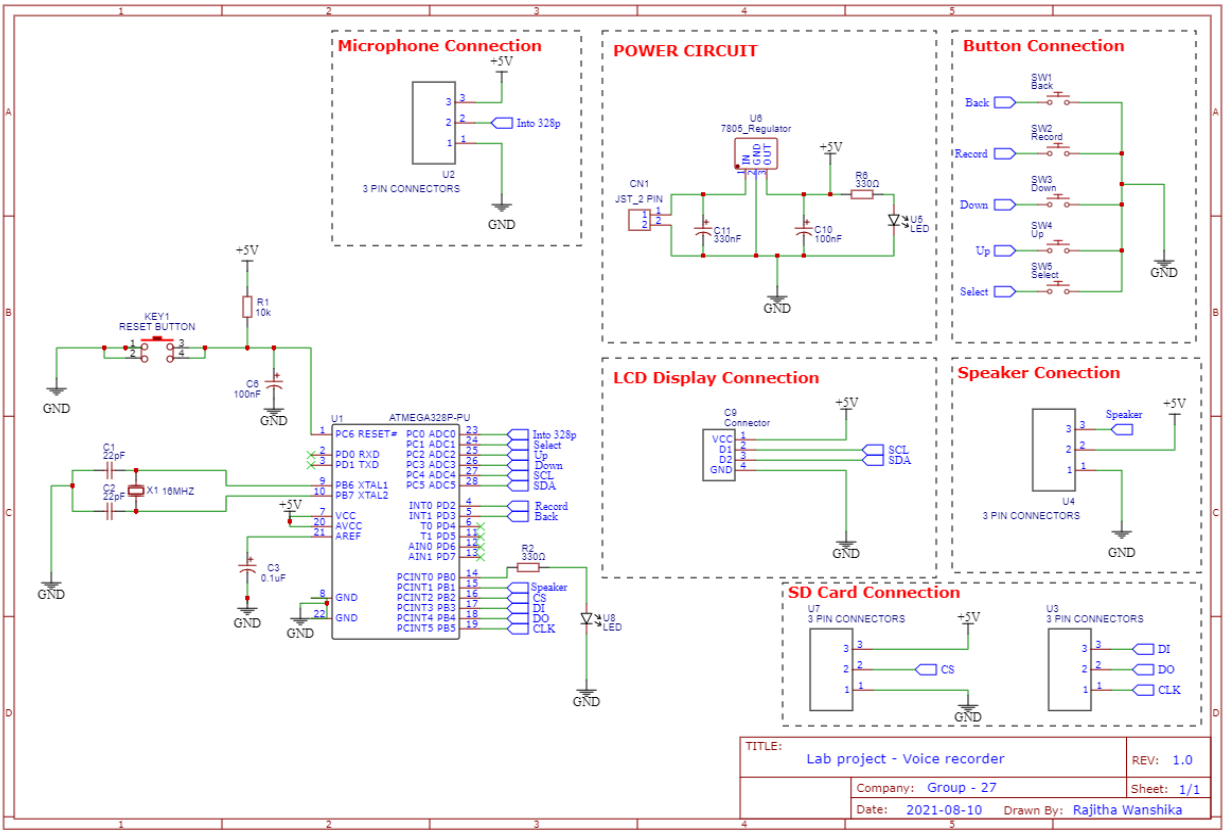





Proteus simulation – This is done before the real simulation to check whether code is working properly.

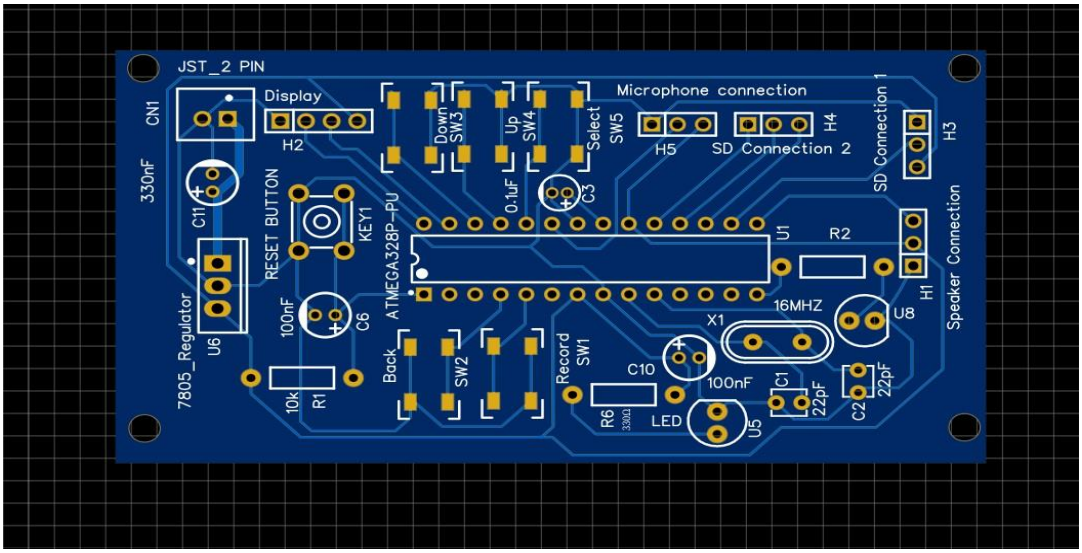
## 2.4. Schematics and PCB layouts

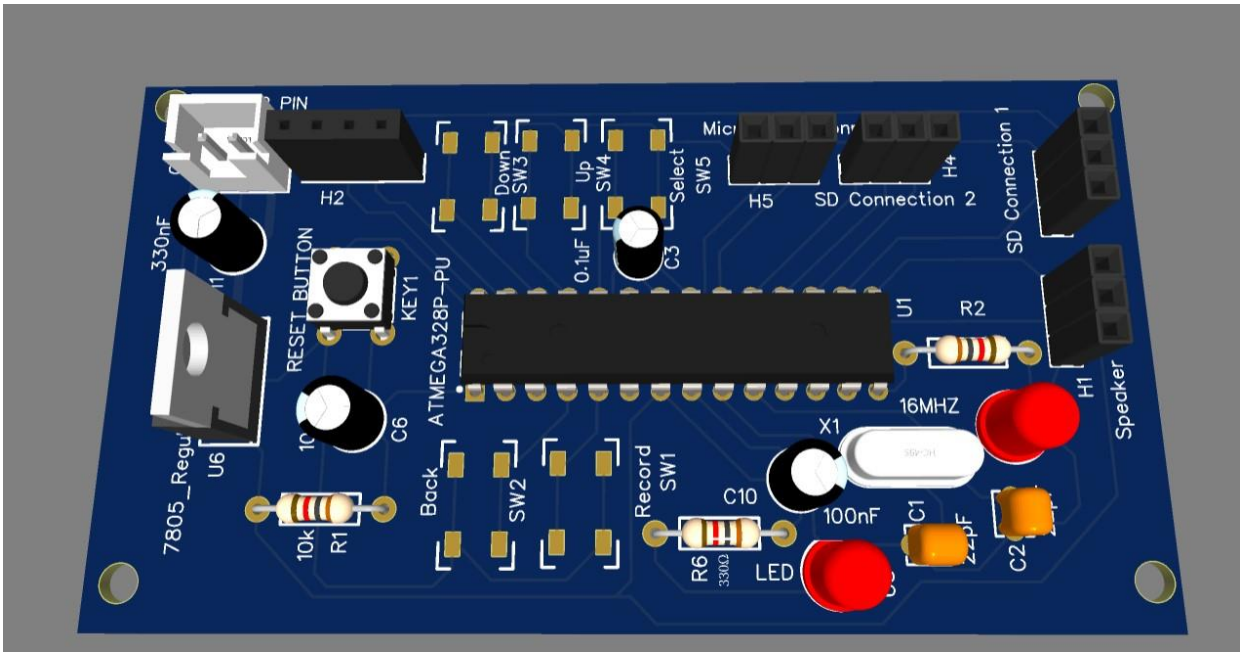
### 2.4.1. Schematics design




### 2.3.2. PCB layouts

PCB is approximately about 8 cm × 6cm in size.






## 2.4. Audio enhancement techniques

The main techniques used in audio enhancing are the high pass and low pass filters. We got to code them in MATLAB and converted them back into Arduino again. The challenge was doing that in the time domain.

Mathematically the technique we used was building the differential equations of the two filters and converting them back into the difference equations as shown below for low pass filters.

The purpose of a low pass filter is to cutoff the frequencies higher than a specified frequency known as the cut off frequency. The CT response of this filter can be found according to the following differential equation,

$y(t)+1\omega cdydt=x(t)$ ,

where $\omega c=2\pi fc\rightarrow$ is the cut-off frequency

By using the approximations

$y[n]=y(nTs)$ ,

$x[n]=x(nTs)$ and

$dydt\approx y(nTs)-y((n-1)Ts)Ts$ where $fs=1Ts\swarrow\rightarrow$ sampling frequency

we were able to generate the difference equation,

$y1=\alpha x1+(1-\alpha)y0$ where $\alpha=2\pi fcfs+2\pi fc$

Likewise for the high pass filter,

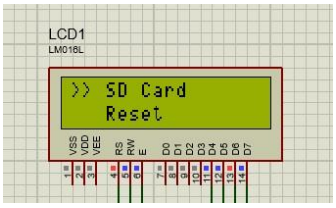
$y1=(1-\alpha)(x1-x0)+(1-\alpha)y0$ with the same $\alpha$.

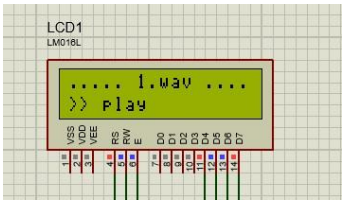
## 2.5. Menu handling

Our product has two menu options.

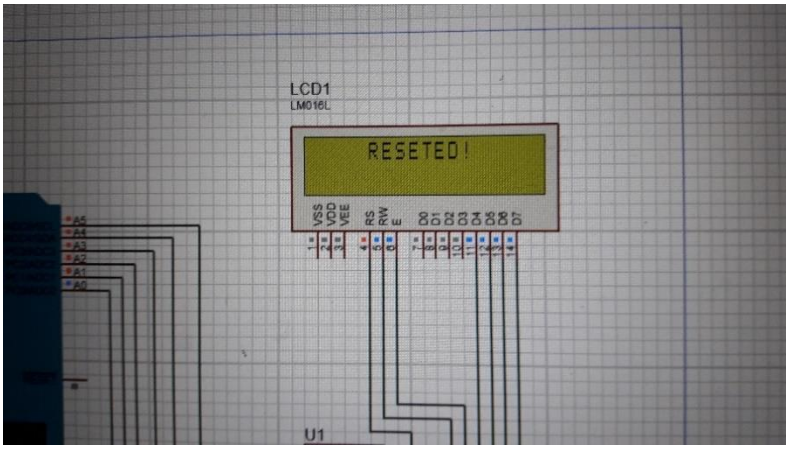
I. Play selectively each recorded audio – we can play selectively 1.wav or 2. Wav or 3. Wav at any given time by going through the menu.


II. Reset the SD card – if we select reset, we create a code that it can delete all the recording in the SD and clean the SD card.[8]



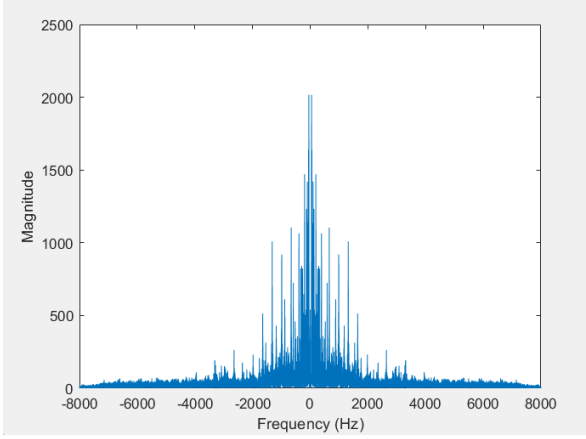




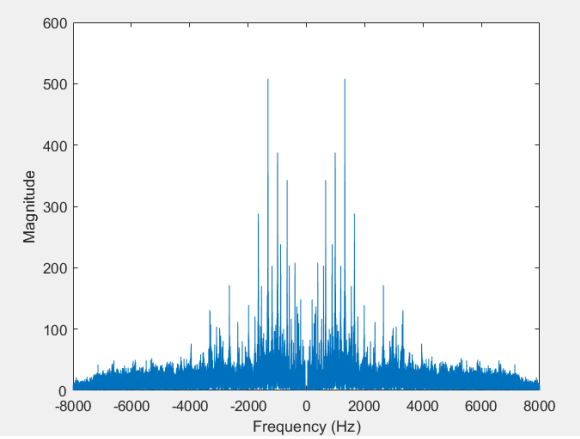

[8]

## 2.6. Simulated inputs and outputs

- The MATLAB code for the high pass filter and the results are as follows.



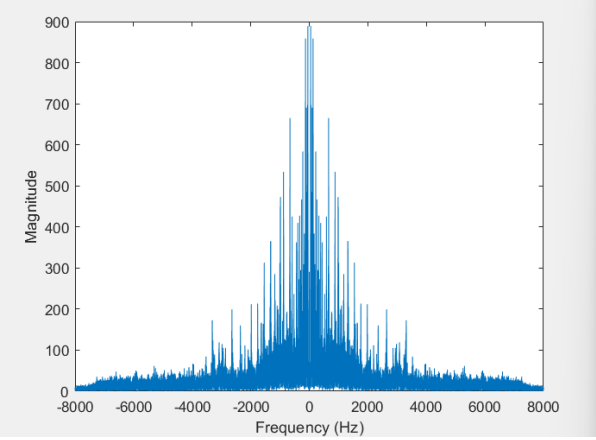

Signal in frequency domain before filtering



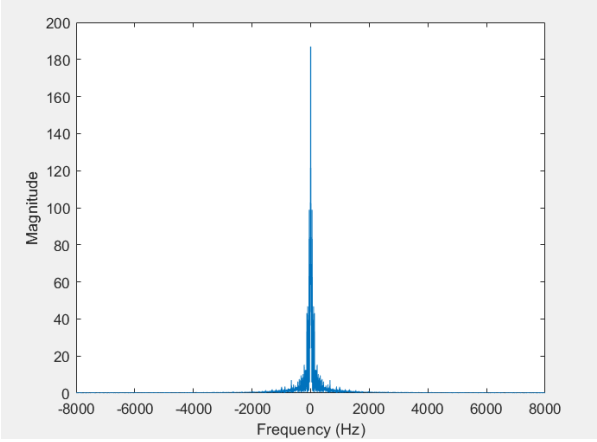

Signal in frequency domain after filtering

```
load handel.mat

filename = 'handel.flac';
audiowrite(filename,y,Fs);
samples = [20*Fs,34.5*Fs];
clear y Fs
[y,Fs] = audioread('test.wav',samples);
t=linspace(0,1/Fs,length(y));

y2=[0];
Fc=1000;
x0=0;
alpha=(2*pi*Fc)/(Fs+2*pi*Fs);
for t1=2:length(t)
    s=(1-alpha)*(y(t1)-x0+y2(end));
    y2=[y2 s];
    x0=y(t1);
end

figure(1)
Y=fft(y);
n = length(y);
fshift = (-n/2:n/2-1)*(Fs/n);
yshift = fftshift(Y);
plot(fshift,abs(yshift))
xlabel('Frequency (Hz)')
ylabel('Magnitude')

figure(2)
Y2=fft(y2);
n2 = length(y2);
fshift = (-n2/2:n2/2-1)*(Fs/n2);
yshift = fftshift(Y2);
plot(fshift,abs(yshift))
xlabel('Frequency (Hz)')
ylabel('Magnitude')

sound(y2,Fs)
```

- The MATLAB code for the low pass filter and the results are as follows,



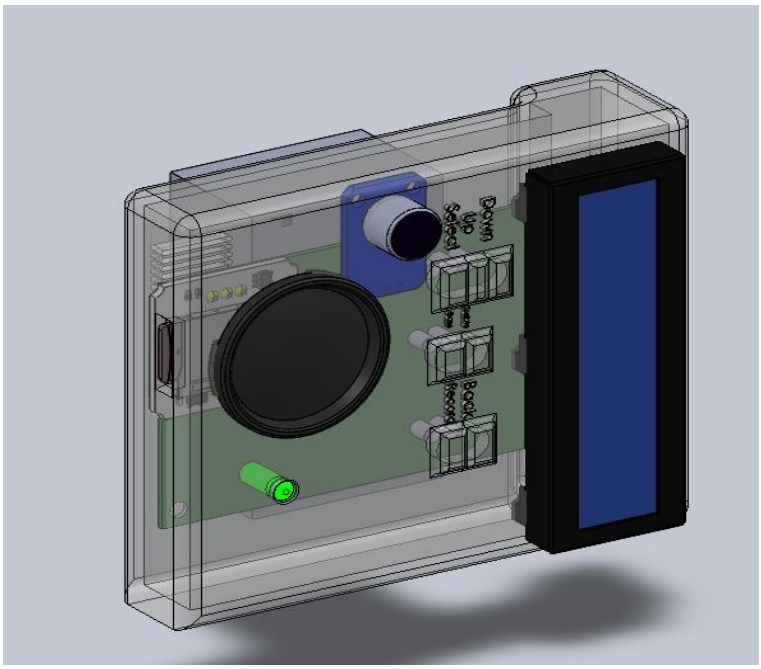

Signal in frequency domain before filtering

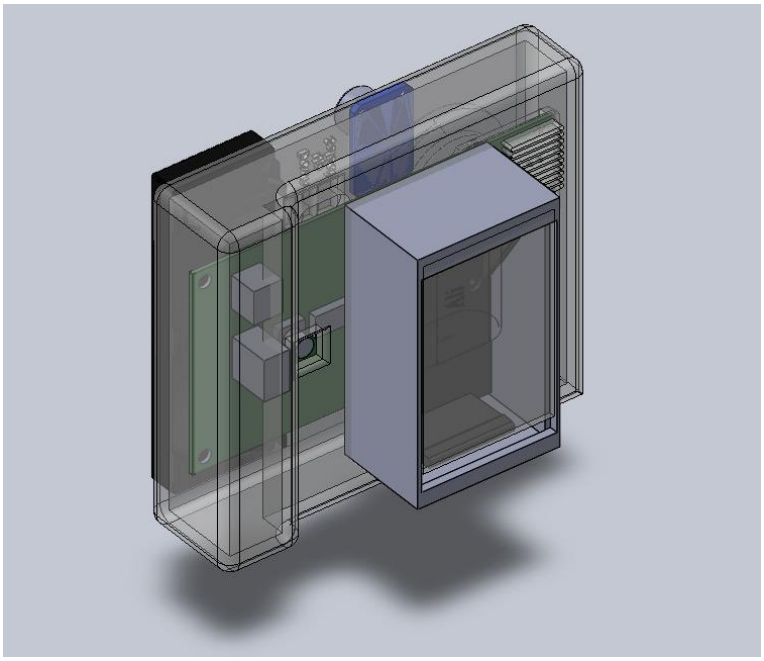



Signal in frequency domain after filtering

```
load handel.mat

filename = 'handel.flac';
audiowrite(filename,y,Fs);
samples = [26*Fs,34*Fs];
clear y Fs
[y,Fs] = audioread('test.wav',samples);
t=linspace(0,1/Fs,length(y));

y2=[y(1)];
Fc=40;
alpha=(2*pi*Fc)/(Fs+2*pi*Fs);
for t1=1:length(t)
    s=alpha*y(t1)+(1-alpha)*y2(end);
    y2=[y2 s];
end

figure(1)
Y=fft(y);
n = length(y);
fshift = (-n/2:n/2-1)*(Fs/n);
yshift = fftshift(Y);
plot(fshift,abs(yshift))
xlabel('Frequency (Hz)')
ylabel('Magnitude')

figure(2)
Y2=fft(y2);
n2 = length(y2);
fshift = (-n2/2:n2/2-1)*(Fs/n2);
yshift = fftshift(Y2);
plot(fshift,abs(yshift))
xlabel('Frequency (Hz)')
ylabel('Magnitude')

sound(y2,Fs)
```

## 2.7. Solidswork design (Diagrams of each part and of full assembly)

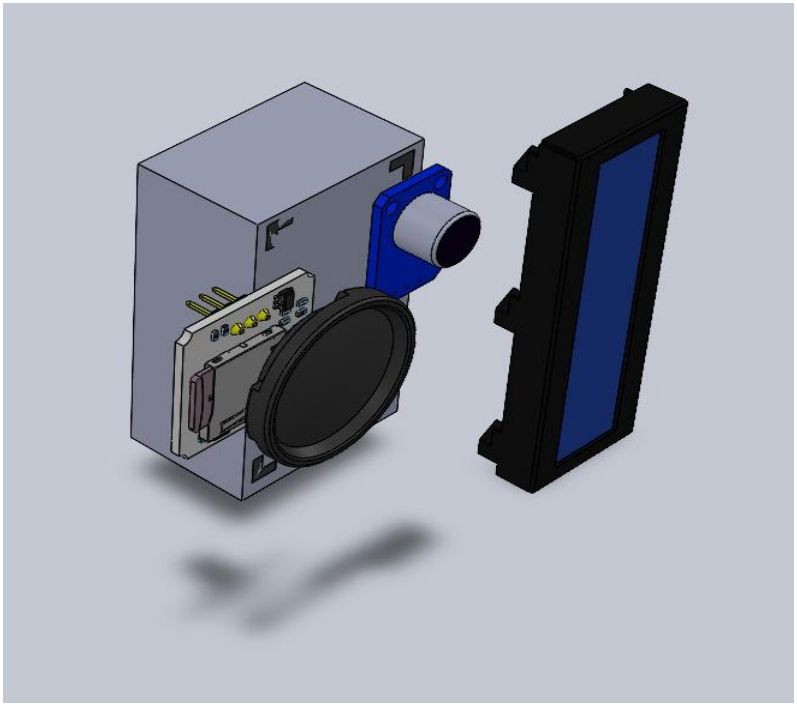


Enclosure design layout

- Enclosure size is approximately – 91.96mm × 78.74 mm × 24.92 mm( L × W × T)
  Thickness will increase up to 41.68 mm because of the size of battery pack.


External view of the enclosure



Internal part diagram

## Results

- Our project can handle up to three voice recordings, it can save them, and user can easily navigate through menu and can play any record at any time.

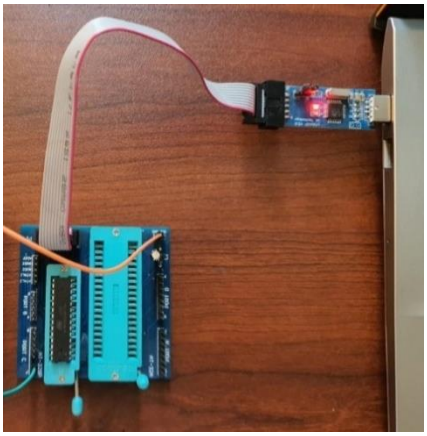
- When we were making the prototype, we had a heating issue in the amplifier module. The reason was the current drawn by the speaker was high. To avoid this, we used a phone charger as a separate power supply. But the voltage was still high. So, we used a DC-DC buck converter to step down the voltage.

- First, we planned to add a keypad to the circuit. But there were not enough pins for that because of other modules like the SD module and LCD. To avoid that we used the I2C communication method to reduce no of connectors of the LCD display. So, we could connect it using only 2 connectors.

- In the beginning, we had memory problems in the microcontroller because our code was too large. So, we had to remove the keypad and optimize the code. We added 5 push buttons instead of a keypad.

- When we are moving from Arduino to AVR C we had to face many difficulties with libraries. It was exceedingly difficult to convert our Arduino code to AVR C. So, we were informed to continue with Arduino.


- At first, we used an Arduino Uno board for testing our prototype. When we use the microcontroller separately, we had some difficulties with uploading the code to microcontroller. For that we used a development board and a USBasp.



- Because we used the microcontroller separately, we had to add an external oscillator to the circuit. But in the first test run circuit responses were awfully slow. So, we had to find information about fuse bits of a MC and change them according to our oscillator.

## Discussion

- We able to know how to design a new product with all pcb, enclosure design and with full code.
- We able to how to handle multiple bugs in the code how to fix them.
- With a lot of failures, we can be able to build a very user-friendly voice recorder with record, play ,pause, stop, selectively pay audio options with audio enhancements and with a SD module.
- We learn how deliver a certain task at a given time period.
- We learn how to use Altium, Solidswork and proteus softwares for design purpose.

## Acknowledgements

First and foremost, we would like to express our sincere gratitude to our group coordinator Mr. Bhanuka Malith Silva, assistant lecturer, Electronic and Telecommunication department, University of Moratuwa for keeping contact with us throughout the entire span of the project taking countless meetings with us and guiding us in each single step of our project.
And we would like to thank Mrs. Dileeka Dias, Senior assistant lecturer, Electronic and Telecommunication department, University of Moratuwa along with all the junior lecturers Mr. Sahan Liyanarachchi, Kithmin Wickramasinghe, Abarajithan Gnaneswaran, Achintha Wijesinghe, Ashwin Silva, Thamindu P, Hiran for conducting sessions based on specific stages of the project and for the knowledge given through the Introduction to Telecommunications module.
And we would also like to remember Mr. Ajith Pasquel for conducting sessions on SOLIDWORKS essentials and Altium under the module electronic product design and manufacture.

# Appendices

## 5.1. Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SD.h>
#include <SPI.h>
#include <TMRpcm.h>

LiquidCrystal_I2C lcd(0x20, 16, 2);

#define SD_ChipSelectPin 10
Sd2Card card;
TMRpcm audio;
int audiofile = 0;
unsigned long i = 0;
bool recmode = 0;

int upButton = A2;
int downButton = A3;
int selectButton = A1;
int backButton = 3;
int recbutton = 2;

int menu = 1;
int menu1 = 1;
int menusd = 0;
int menuplay = 0;
int playpause =0;
int recmode1 = 0;

void setup(){
  lcd.begin(16,2);
  Serial.begin(9600);

  pinMode(A0, INPUT);
  pinMode(8, OUTPUT);
  SD.begin(SD_ChipSelectPin);
  audio.CSPin = SD_ChipSelectPin;

  audio.speakerPin = 9;
  pinMode(9,OUTPUT);

  pinMode(upButton, INPUT_PULLUP);
  pinMode(downButton, INPUT_PULLUP);
  pinMode(selectButton, INPUT_PULLUP);
  pinMode(backButton, INPUT_PULLUP);
  pinMode(recbutton, INPUT_PULLUP);

  lcd.clear();
  lcd.setCursor(1,0);
  lcd.print("Voice Recorder");
  lcd.setCursor(4,1);
  lcd.print("Group 27");
```

```
  delay(3000);
  lcd.clear();

  updateMenu();
 }

 void loop() {
  if (!digitalRead(downButton) and !menusd and !menuplay and !recmode1){
    menu++;
    if (menu>2) {
        menu = 1;
     }
    updateMenu();
    delay(350);
    while (!digitalRead(downButton));
  }

  if (!digitalRead(upButton) and !menusd and !menuplay and !recmode1){
    menu--;
    if (menu<1) {
         menu = 2;
        }
    updateMenu();
    delay(350);
    while (!digitalRead(upButton));
  }

  if (!digitalRead(downButton) and menusd and !menuplay and !recmode1){
    menu1++;
    updatesd();
    delay(350);
    while (!digitalRead(downButton));
  }

  if (!digitalRead(upButton) and menusd and !menuplay and !recmode1){
    menu1--;
    updatesd();
    delay(350);
    while (!digitalRead(upButton));
  }

  if (!digitalRead(selectButton) and !menusd and !menuplay and menu==1 and !recmode1){
    menusd=1;
    //menu1=audiofile;
    updatesd();
    delay(350);
    while (!digitalRead(selectButton));
  }

  if (!digitalRead(selectButton) and !menusd and !menuplay and menu==2 and !recmode1){
    Reset();
    delay(350);
    while (!digitalRead(selectButton));
  }

  if (!digitalRead(selectButton) and menusd and audiofile and !menuplay and menu==1 and !recmode1){
```

```
    menuplay=1;
    playmenu();
    playpause =1;
    delay(350);
    while (!digitalRead(selectButton));
   }

   if (!digitalRead(selectButton) and menusd and menuplay and playpause and menu==1 and !recmode1){
    audio.pause();
    playpause=-1;
    lcd.clear();
    lcd.print("  Audio paused  ");
    delay(350);
    while (!digitalRead(selectButton));
   }

    if (!digitalRead(selectButton) and menusd and audiofile and menuplay and menu==1 and playpause==-1 and
!recmode1){
    playmenu();
    int playpause =1;
    delay(350);
    while (!digitalRead(selectButton));
   }

   if (!digitalRead(backButton) and menusd and menuplay and menu==1 and !recmode1){
    menuplay=0;
    if (playpause ==1 or !audio.isPlaying()) {
     audio.disable();
    }
    if (playpause ==-1 or audio.isPlaying()) {
     audio.pause();
     audio.disable();
    }
    updatesd();
    delay(350);
    while (!digitalRead(backButton));
   }

   if (!digitalRead(backButton) and menusd and !menuplay and !recmode1){
    menusd=0;
    updateMenu();
    delay(350);
    while (!digitalRead(backButton));
   }

   if (!digitalRead(recbutton))   {
      record();
      delay(500);
    }

}

void updateMenu() {
     if (menu ==1)  {
      lcd.clear();
      lcd.print(">> SD Card");
```

```
      lcd.setCursor(0, 1);
      lcd.print("   Reset");
      }
    if (menu ==2)    {
      lcd.clear();
      lcd.print("   SD Card");
      lcd.setCursor(0, 1);
      lcd.print(">> Reset");
     }

 }

void updatesd() {
  if (audiofile==0){
    lcd.clear();
    lcd.print("No Recordings");
  }
  if (menu1 == 0 and audiofile != 0) {
      menu1=1;
      lcd.clear();
      lcd.print("..... 1.wav ....");
      lcd.setCursor(0, 1);
      lcd.print(">> play");
  }
  if (menu1 == 1 and menu1<=audiofile) {
      lcd.clear();
      lcd.print("..... 1.wav ....");
      lcd.setCursor(0, 1);
      lcd.print(">> play");
  }
  if (menu1 == 2 and menu1<=audiofile) {
      lcd.clear();
      lcd.print("..... 2.wav ....");
      lcd.setCursor(0, 1);
      lcd.print(">> play");
  }
  if (menu1 == 3 and menu1<=audiofile ){
      lcd.clear();
      lcd.print("..... 3.wav ....");
      lcd.setCursor(0, 1);
      lcd.print(">> play");
  }
  if (menu1 == 4 and audiofile==3 ){
      menu1 = 3;
      lcd.clear();
      lcd.print("..... 3.wav ....");
      lcd.setCursor(0, 1);
      lcd.print(">> play");
  }
}

void playmenu() {
  if (menu1 == 1 and menu1<=audiofile) {
      lcd.clear();
      lcd.print(" 1.wav playing ");
      lcd.setCursor(0, 1);
```

```
      lcd.print(">> back");
      audio.setVolume(6);
      audio.quality(1);
      audio.play("1.wav");
   }
   if (menu1 == 2 and menu1<=audiofile ){
      lcd.clear();
      lcd.print(" 2.wav playing ");
      lcd.setCursor(0, 1);
      lcd.print(">> back");
      audio.setVolume(6);
      audio.quality(1);
      audio.play("2.wav");
   }
   if (menu1 == 3 and menu1<=audiofile ){
      lcd.clear();
      lcd.print(" 3.wav playing ");
      lcd.setCursor(0, 1);
      lcd.print(">> back");
      audio.setVolume(6);
      audio.quality(1);
      audio.play("3.wav");
   }
}


void Reset() {
  for (int i = -1; i <= audiofile; i++) {
    if (audiofile == 0){
      lcd.clear();
      lcd.print(" sd is already free");
      delay(500);
    }
    if (audiofile == 1){
      SD.remove("1.wav");
    }
   if (audiofile == 2){
      SD.remove("1.wav");SD.remove("2.wav");
    }
    if (audiofile == 3){
      SD.remove("1.wav");SD.remove("2.wav");SD.remove("3.wav");
    }
  }
  audiofile=0;
  lcd.clear();
  lcd.print("    RESETED!    ");
  delay(1000);
  updateMenu();
  delay(350);

}

void record() {
    while (i < 300000) {
      i++;
    }
```

```
    i = 0;
    if (recmode == 0) {
      recmode = 1;
      audiofile++;
      switch (audiofile) {
        case 1: audio.startRecording("1.wav", 8000, A0,1);digitalWrite(8, HIGH); recmode1 = 1;
lcd.clear();lcd.print("Recording 1.Wav"); break;
        case 2: audio.startRecording("2.wav", 8000, A0,1);digitalWrite(8, HIGH); recmode1 = 1;
lcd.clear();lcd.print("Recording 2.Wav"); break;
        case 3: audio.startRecording("3.wav", 8000, A0,1);digitalWrite(8, HIGH); recmode1 = 1;
lcd.clear();lcd.print("Recording 3.Wav"); break;
      }
    }
    else {
      recmode = 0;
      switch (audiofile) {
        case 1: audio.stopRecording("1.wav");digitalWrite(8, LOW); lcd.clear(); recmode1 = 0; updateMenu();
break;
        case 2: audio.stopRecording("2.wav");digitalWrite(8, LOW); lcd.clear(); recmode1 = 0; updateMenu();
break;
        case 3: audio.stopRecording("3.wav");digitalWrite(8, LOW); lcd.clear(); recmode1 = 0; updateMenu();
break;
      }
    }
}
```

## 5.2. "TMRPCM" library

TMRPCM is an Arduino based library used to play wav file formats in a Arduino codes. With that its used Arduino SD library to communicate data and commands with the SD card such as binary values of a wav file. TMRPCM used to access wav file data, that can play by using PWM pins(9) from a speaker by connecting with the Arduino UNO board.

We can connect the speaker directly into output digital PWM pin 9,10 for 328 devices such as UNO, NANO etc. The output is not a smoothy signal, it is a choppy PWM digital signal. To get a better sound experience we can use a digital to analog converter.

**Main features**

- Direct playback can be done by getting wav file data from SD card.
- It can play 8-bit resolution signal with a sample rate of 8000Hz to 32000Hz, Mono type.
- Allows only code in main loop to run when audio playback occur.- Async playback.
- "TIMER1", single time operator for mega and uno boards or the "TIMER3,4 or 5" only for mega board.
- Dual speakers or Complimentary output.
- By double Oversampling.(By sampling theorem)
- TMRPCM supports to almost every Arduino boards and Atmel micro-controllers such as uno, mega, nano etc.

## References/bibliography

### References

[1] Rajkumar2506, "Interfacing LCD With Arduino Using Only 3 Pins," instructable circuit, 30 november 2016. [Online]. Available: https://www.instructables.com/Interfacing-LCD-With-Arduino-Using-Only-3-Pins/.

[2] arduino engineer, "arduino engineer," Digital Audio Recorder, 29 July 2016. [Online]. Available: https://duino4projects.com/arduino-projects-digital-audio-recorder/.

[3] U. a. Butt, "arduino-audio-player," EngineersGarage, 24 March 2021. [Online]. Available: https://www.engineersgarage.com/arduino-audio-player/.

[4] D. 8.-b. D. t. A. Converter, "dac0808-8-bit-digital-to-analog-converter," microcontrollerslab, 2013-2021. [Online]. Available: https://microcontrollerslab.com/dac0808-8-bit-digital-to-analog-converter/.

[5] Microchip-Atmel, "atmel," Microchip, 2020. [Online]. Available: https://start.atmel.com/#dashboard.

[6] wikipiedia, "TMRh20/TMRpcm," Tmrpcm, [Online]. Available: https://github.com/TMRh20/TMRpcm/wiki.

[7] Arduino, "Arduino/Tutorial/LibraryExamples," Arduino, 2021(c). [Online]. Available: https://www.arduino.cc/en/Tutorial/LibraryExamples.