Project Title
Optional Subtitle

Weiyu Chen
Put Your MU Student IDs Here

Project – **2023**
Degree Major - CSSE

Maynooth International Engineering College
福州大学梅努斯国际工程学院

A project submitted in partial fulfilment of the requirements for the
BSc in Computer Science and Software Engineering

Supervisor: Joseph Timoney

Declaration

We hereby certify that this material, which we now submit for assessment on the program of study as part of **BS** qualification, is *entirely* our own work and has not been taken from the work of others - save and to the extent that such work has been cited and acknowledged within the text of our work.

We hereby acknowledge and accept that this thesis may be distributsd to future first year students, as an example of the standard expected first year projects.

Signed: Chen Weiyu Date: 2023/6/13

P63_Arduino_or_RaspberryPi_based_KTV_Machine

members:
Kang Jiarui
Yang Junxiang
LI Guoqing
Chen Weiyu
Liang Zhewen

Abstract

This project is a machine created with Arduino as the center, which can play songs, display lyrics, and have interactive functions. It can play various songs and display the lyrics of each song being sung on an LED display screen. It also has powerful interactive functions, which can interact with users in various aspects, including visual and auditory interactions. This provides opportunities and platforms for some people who love music and singing. Of course, this also provides opportunities for some introverted individuals, as it will motivate you to speak up bravely.

Introduction

The goal of this project is to synchronize lyrics and songs, and provide positive feedback and interaction for users. So what is needed is not only code development on the program, but also the connection and communication of the Arduino board on the hardware. From a code perspective, how to achieve complete operation and learn new programming languages is a challenge, as well as how to combine with hardware boards and whether there are communication protocols. From a hardware perspective, it is important to choose a suitable Arduino board and install sensors to sense and respond to the surrounding environment, enabling SD cards to read song information and content. We have proposed solutions for different problems. In terms of program development, you can search for code tutorials online and relevant literature for discussion and learning within the group; For hardware issues, we independently developed and designed a circuit structure of one motherboard and two auxiliary boards through online tutorials. The evaluation can be conducted from the following aspects: the quality of the song, the synchronization rate of the lyrics, and the sensitivity to user feedback. This project realizes the synchronization of songs and lyrics, using the uno board and SV5W built-in commands to pause, previous, next, increase or decrease the volume, and specify the song. And use LED screens to interact with users. What's more, we use SD card modules and LRC files to reduce memory footprint on the UNO board.

Technical Background

Arduino is a convenient, flexible, and easy-to-use open-source electronic prototype platform.

Includes hardware (various models of Arduino boards) and software (Arduino IDE).

It is built on the open source simple I/O interface version and has a Processing/Wiring development environment like Java and C languages. It mainly consists of two parts: the hardware part is an Arduino circuit board that can be used for circuit connections; The other is the Arduino IDE, the programming environment on your computer. You just need to write program code in the IDE, upload the program to the Arduino circuit board, and the program will tell the Arduino circuit board what to do

Arduino can perceive the environment through various sensors, and feedback and influence the environment by controlling lights, motors, and other devices. The microcontroller on the board can be programmed using Arduino programming language, compiled into binary files, and burned into the microcontroller. The programming of Arduino is achieved through the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino based projects can only include Arduino, or they can include Arduino and other software running on a PC. They communicate with each other (such as Flash, Processing, MaxMSP) to achieve this.

This project uses three UNO boards, which are connected through the iic communication protocol. One of them serves as the host and the other two serve as slaves.

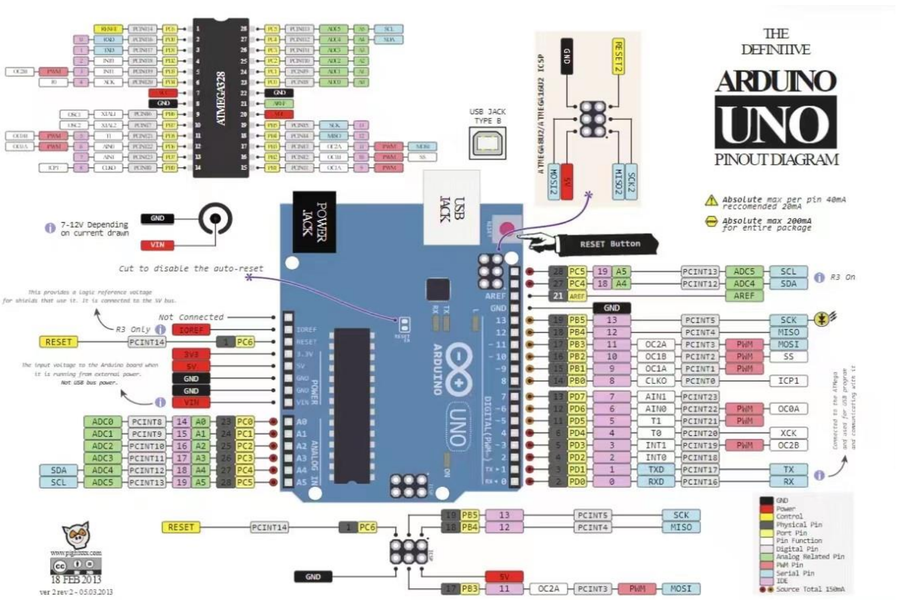
THE DEFINITIVE ARDUINO UNO PINOUT DIAGRAM
ATMEGA328
USB JACK TYPE B
POWER JACK
RESET Button
7-12V Depending on current drawn
Cut to disable the auto-reset
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
18 FEB 2013

Fig. 1. Pin distribution diagram of UNO board

The host is used to control the SV5W module for song playback operations. The built-in commands of SV5W enable playback pause, previous song, next song, volume increase, volume decrease, and specified songs. These functions are controlled separately through six external buttons connected to the six pins of the host.

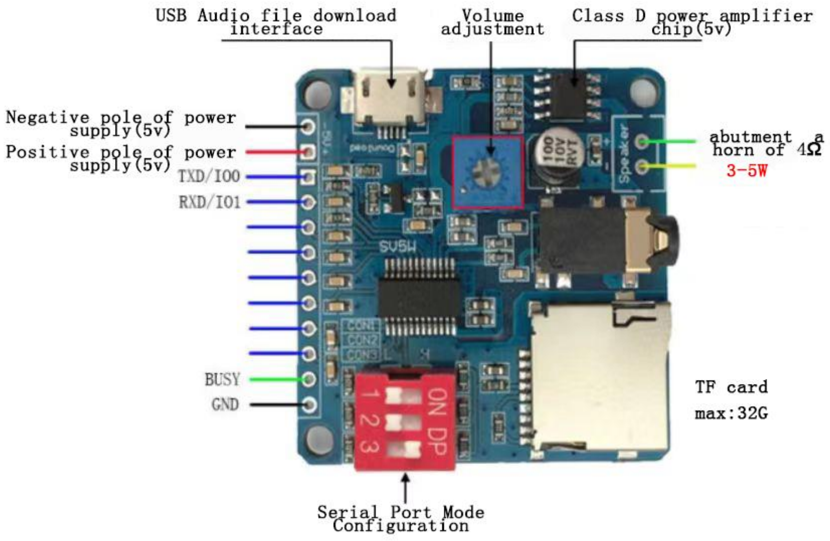
USB Audio file download interface
Volume adjustment
Class D power amplifier chip(5v)
Negative pole of power supply(5v)
Positive pole of power supply(5v)
TXD/IO0
RXD/IO1
BUSY
GND
abutment a horn of 4Ω
3-5W
TF card max:32G
Serial Port Mode Configuration

Fig. 2. Introduction to SV5W

Slave 1 is used to control the LCD 1602 display screen to display lyrics and read lyrics stored in the SD card. 1. Determine whether it is in a paused state by reading the variable of the host's SV5W playback status. If it is, use the interrupt function to make the LCD 1602 display "Song Pause". 2. Use the built-in commands of SV5W to query the current track being played, and then send the track information to Slave 1 through IIC communication. Slave 1 then reads the corresponding lyrics from the SD card and displays them on LCD 1602. Most of the lyrics are relatively long, so it is necessary to scroll and display them on LCD 1602.

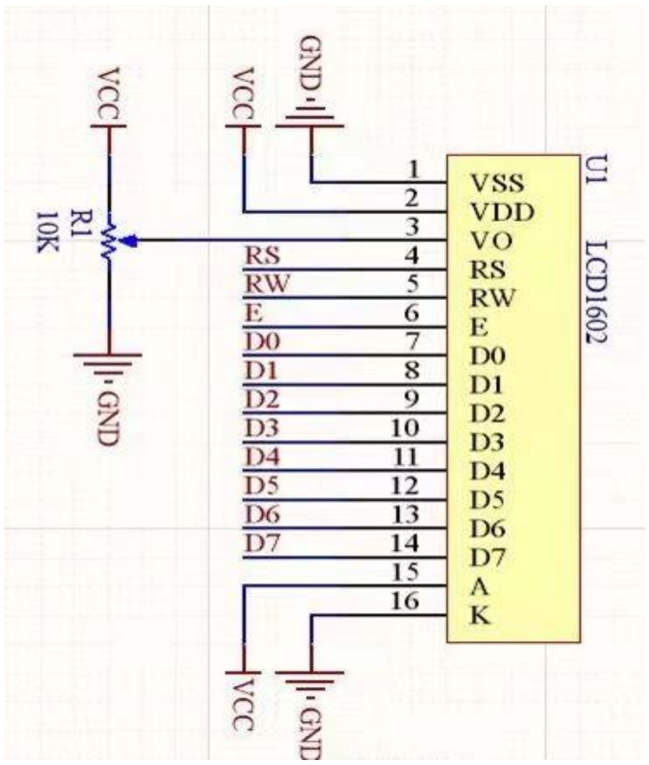

Fig. 3. Circuit schematic diagram of LCD1602

The second slave is used to control the OLED display screen and sound sensor (KV-037). Sound feedback: The sound sensor is used to read the singer's volume in decibels and display it on the OLED. Once the threshold is exceeded, it will display "low voice" on the OLED. On the OLED, new and old songs (old/new) and song tracks are also displayed, both of which are obtained from the host through the iic protocol.

ISD1820 can provide recording and playback functions, only requiring power supply.

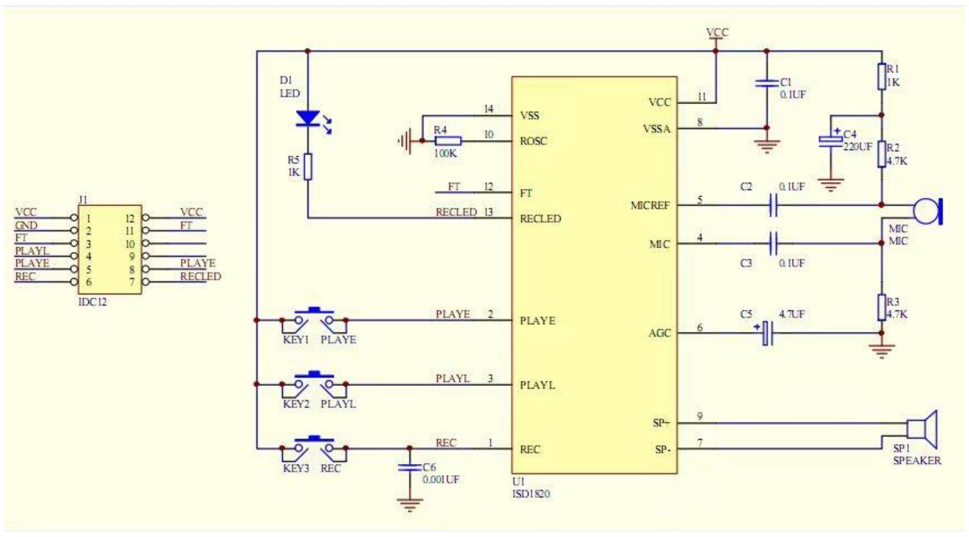

Fig. 4. Circuit schematic diagram of ISD1820

The lyrics in the SD card should have a time prefix before them, which is used to calculate the time when the current lyrics are displayed on the LCD 1602.

The Problem

- 1.Playing songs and displaying lyrics. Some Arduino project samples write the context of lyrics and the tones of songs directly to the source code, but it takes up a lot of memory and is a hassle for both the user and developers. We should find a more subtle way to implement these functions. See [1].

- 2.Have a system where the user can choose which songs to categorize. Help users find the song they want conveniently.

- 3. Using the hardware serial port of the uno board will cause the instructions sent and received by the SV5W module to go unresponded.

- 4. The communication protocol of the three uno boards is spi or iic.

- 5. Because the built-in commands of SV5W use the soft serial port, the specified commands cannot be sent directly in the serial port monitor, but can only be written in the program, resulting in the failure to specify control.

- 6. LCD1602 is responsible for displaying the lyrics, SV5W is responsible for playing the song, and the lyrics stop playing when the song pauses.

- 7. A sentence that is too long will cause LCD1602 not to display all the lyrics.

The Solution

- 1. The use of SV5W voice module can solve the problem of poor sound quality caused by writing the tone directly to the memory, greatly expanding the time and number of tracks for single playback. And SV5W module has a wealth of built-in hexadecimal instructions, you can expand more functions. The use of sd card can provide a large memory, you can expand the number of lyrics, and the use of sd card can use lrc files, so that you can accurately control the time of each sentence on the LCD1602, without using repeated delay () method, and without manually calculating the time difference between two lyrics, the program will automatically calculate. You can find the code in Appendix A.

- 2. Use the SV5W module's built-in instruction to specify songs to replace the classification of new and old songs and set the 11th song as the dividing line between new and old songs.

- 3. Uno has only 0 (RX) and 1 (TX) a set of hardware serial ports, this set of serial ports is often used for communication with the computer, will cause conflict, so you need to use software to simulate the serial port to communicate with the SV5W module serial port, to ensure that the specified normal corresponding and accepted.

- 4. Through reasonable planning, only one uno board is used for sending, and the other two uno boards are responsible for receiving, which simplifies the procedure and uses the iic protocol uniformly.

- 5. Through the external switch to call the corresponding instructions, to achieve the specified function of sending instructions. However, pay attention to the use of delay () to prevent shake and avoid instruction conflicts. And the switch needs to pull the resistor to avoid the problem of level instability.

- 6. Because slave1's lyric display program is running, the code to pause the lyrics cannot be run. Therefore, run the code that suspends the lyrics through the interrupt function, because the interrupt function has a high priority, trigger the interrupt condition to suspend the original running code first, run the interrupt function first, and then continue to execute the original remaining code after the interrupt function is executed. This solves the problem of synchronous suspension of lyrics and songs, and the lyrics can still be displayed synchronously with the song even after the pause.

- 7. The scrollDisplayLeft() method can be used to implement scrolling, and delay () can be used to determine the scrolling speed based on the length of the lyrics.

Evaluation

Music classification, playback and pause, volume adjustment, lyrics display and pause (Sync with play and pause of lyrics), and microphone volume display can all operate normally. In addition, we have added a button to switch directly from playing a new song to playing an old one and can achieve a ten second recording and recording playback function (which can be played directly at the same time as the song). But slave2 has a problem with the upload port, which causes the display of the song sequence "sq" on the OLED sometimes to be problematic.

Conclusion

This project is written with ARDUINO, which is based on the development board, and the development board needs a USB connection to the computer to complete the transportation.

This project is to make a KTV player centred around the arduino. It should play the song and display the lyrics, possibly using more than just a LCD screen. It should also offer other features to the user that will help them to interact more with the singing.

This project uses three UNO boards, which are connected through the iic communication protocol, one of which serves as the host and the other two as the slave. The host is used to control the SV5W module for the operation of playing songs, through the built-in instructions of SV5W to achieve playback pause, the previous song, the next song, volume increase, volume decrease, specify songs, and through the six pins of the host uno six buttons control these functions respectively.

The No. 1 slave is used to control the lcd1602 display to display the lyrics and read the lyrics stored in the sd card. The No. 1 slave is then displayed on the lcd1602 by reading the corresponding lyrics in the sd card. The No. 2 slave is used to control the oled display and sound sensor, and the ISD1820 can provide the function of recording and recording playback, which only needs to be powered. The lyrics in the SD card should have a time prefix before them, which is used to calculate the time that lcd1602 displays the current lyrics

Future Work: work is being carried out in several areas to further enhance the functional usefulness of the project. In the future, we can add some LED lights to decorate around the device to emit colorful light, to set off the atmosphere of KTV, so that users can be immersive. Not only that, we can increase the use of the development board to achieve the song search function, so that you can quickly find the songs you need, but only the songs that have been downloaded. Not only that, we can set up a stabilizing device to stabilize the connection and stability between each development board, which can be convenient to use.

References

[1]Song with Songtext(Karaoke) Arduino Project Hub.<https://projecthub.arduino.cc/SuchtFrosch/c12dfeae-f88b-4c47-b4de-67a67d17dc4a >.

[2] Arduino Karaoke Prank Machine. <https://www.gadgetronicx.com/diy-karaoke-prank-machine/>.

[3] Arduino MP3 Player. <http://educ8s.tv/arduino-mp3-player/>.

[4] Lyrics Parse Machine. <https://github.com/CatImmortal/CatLrcParser>.

[5] bug of arduino soft serial port (virtual serial port). < http://t.csdn.cn/IaDbz>.

[6] arduino - I2C communication. < http://t.csdn.cn/DpcPB>.

[7] Common sensor explanation 5 -- Sound sensor - (KY-037). < http://t.csdn.cn/UtS3Y>.

[8] Simple and easy to use ISD1820 voice recording and playback chip. < http://t.csdn.cn/x6LHn>.

[9] Arduino Case - Voice Player Module (DY-SV5W). < http://t.csdn.cn/qNJ4a>.

Appendix A

Source code
This is the code for the first uno board (master):

```
// Using the SV5W module, this uno acts as a host and only sends data to the slave and does not receive information from the slave.
// And through the creation of soft serial port and SV5W module serial communication.
#include<SoftwareSerial.h>//Use a soft serial port
//Reason: uno only 0 (RX) and 1 (TX) a set of hardware serial port, this group of serial port is often used to communicate with the computer,
//if you need to communicate with SV5W module serial port need to use software simulation serial port (soft serial port
#include <Wire.h>//Use iic communication
bool mp3_control = false;//bool variable, the state quantity of whether the voice module is in playback state
SoftwareSerial softSerialsv(10,11);//Soft serial port is defined, 10 is RX and 11 is TX
unsigned char data1=0x00;//Use an unsigned number instead of 0x00.Writing 0X00 directly would cause the write () method to be overloaded,
//requiring type determination
String recipt="";//Used to accept the return value of the SV5W module
int num=0;
void setup()
{
    Wire.begin();//Initialize iic communication with no written address value in parentheses, that is, access as a host
    Serial.begin(9600);//Serial communication with the computer
    softSerialsv.begin(9600);//Soft serial communication with sv5w
    softSerialsv.listen();//Listen softserial port softSerialsv
//Module state control sensor pin (button)
  pinMode(2, INPUT);//Control play and pause, the next 2,3,4,5,6,7 functions will be implemented with flag==true
    pinMode(3, INPUT);//Switch to previous song
    pinMode(4, INPUT);//Switch to the next song
    pinMode(5, INPUT);//Control volume plus
    pinMode(6, INPUT);//Control volume reduction
    pinMode(7, INPUT);//New songs and old songs classification, here uses the specified song instead of classification function
    pinMode(8,OUTPUT);//Initialize pin 8 for sending interrupt trigger mode to slave 1
  digitalWrite(8,LOW);//The default is low, and when high, slave 1 executes the interrupt function
}

void loop()
{
    //Control module plays the song (key sensor high level trigger)
    if((digitalRead(2) == HIGH)&&(mp3_control == false))
    {

        delay(500);//Delay to shake


        mp3_control = true;//Change state mp3 control to true (playback state)

        //Send play command    AA 02 00 AC
        softSerialsv.write(0xAA);
        softSerialsv.write(0x02);
        softSerialsv.write(data1);
        softSerialsv.write(0xAC);
        delay(1000);

        //Send (current directory) sequential play instructions AA 18 01 07 CA
        softSerialsv.write(0xAA);
        softSerialsv.write(0x18);
        softSerialsv.write(0x01);
        softSerialsv.write(0x07);
        softSerialsv.write(0xCA);
        if(num==0){
          writer2_1();
          writer3_1();
          num++;
    }
    inspect_songQequence();
        digitalWrite(8,LOW);
    }
    //Control module pause play (button sensor high level trigger)
    if((digitalRead(2) == HIGH)&&(mp3_control == true))
    {

        delay(500);//Delay to shake
        mp3_control = false;//Change state mp3 control to false (paused state)
  //Send a pause command AA 03 00 AD
        softSerialsv.write(0xAA);
        softSerialsv.write(0x03);
        softSerialsv.write(data1);
        softSerialsv.write(0xAD);

          digitalWrite(8,HIGH);

    }
//Control module plays the previous song (button sensor high level trigger)
    if((digitalRead(3) == HIGH)&&(mp3_control == true))
    {

        delay(500);//Delay to shake

        //Send the last command AA 05 00 AF
        softSerialsv.write(0xAA);
        softSerialsv.write(0x05);
        softSerialsv.write(data1);
        softSerialsv.write(0xAF);
        inspect_songQequence();

    }
```

```
//Control module plays the next song (button sensor high level trigger)
   if((digitalRead(4) == HIGH)&&(mp3_control == true))
 {

    delay(500); //Delay to shake

    //Send the next command AA 06 00 B0
    softSerialsv.write(0xAA);
    softSerialsv.write(0x06);
    softSerialsv.write(data1);
    softSerialsv.write(0xB0);
    inspect_songQequence();

 }
 //Control module volume plus (button sensor high level trigger)
   if((digitalRead(5) == HIGH)&&(mp3_control == true))
 {

    delay(500); //Delay to shake

    //Send the volume plus command AA 14 00 BE
    softSerialsv.write(0xAA);
    softSerialsv.write(0x14);
    softSerialsv.write(data1);
    softSerialsv.write(0xBE);
 }
 //Control module volume reduction (button sensor high level trigger)
 if((digitalRead(6) == HIGH)&&(mp3_control == true))
 {

    delay(500);//Delay to shake

   //Send the volume decrement command AA 15 00 BF
    softSerialsv.write(0xAA);
    softSerialsv.write(0x15);
    softSerialsv.write(data1);
    softSerialsv.write(0xBF);
 }//Control module new and old song classification (button sensor high level trigger),
 // here uses the specified song to replace the classification function, the last three are to be changed
 if((digitalRead(7) == HIGH))
 {

        delay(500);//Delay to shake

        //Send the command to play the specified song AA 07 02 00 0B BE
        softSerialsv.write(0xAA);
        softSerialsv.write(0x07);
        softSerialsv.write(0x02);
        softSerialsv.write(data1);
        softSerialsv.write(0x0B);
        softSerialsv.write(0xBE);
        inspect_songQequence();
    }

}
// Query the current track, suitable for 0-99 tracks
void inspect_songQequence(){
    softSerialsv.write(0xAA);
    softSerialsv.write(0x0D);
    softSerialsv.write(data1);
    softSerialsv.write(0xB7);
    //The serial port hexadecimal conversion decimal
    int i,j;
    String sq="";
    while (softSerialsv.available()) {
    int in = (char)softSerialsv.read();
    recipt+=in;
    recipt+=',';
    delay(2);
    }
    Serial.println(recipt);
    if(recipt.length()==17) sq=(String)recipt.charAt(11);// 1-9 songs,
    //these lines are converted with (String) because when you add two character variables, the result is an integer,
    //and when you add characters, you're actually adding their ASCII valu
    else sq=(String)recipt.charAt(11)+(String)recipt.charAt(12);//10 to 20 songs
    Serial.println(sq);
    writer2(sq);//Pass the song order to slave1
    writer3(sq);//Pass the song sequence to slave2
    recipt="";
}
// Send message to slave 2 (start playing)
void writer2_1(){
    Wire.beginTransmission(2);//Start transferring data
```

```

  Wire.write('1');//Send a signal to start displaying subtitles

  Wire.endTransmission(); //Ending the transfer
}
void writer3_1(){
  Wire.beginTransmission(3);//Start transferring data

  Wire.write('1');//Send a signal to start displaying subtitles

  Wire.endTransmission(); //Ending the transfer
}
// Send a message to Slave 2 (song sequence)
void writer2(String sq){
  Wire.beginTransmission(2);//Start transferring data
  if(sq.length()==1) Wire.write(sq.charAt(0));//Song number difference
  else {
    Wire.write(sq.charAt(0));
    Wire.write(sq.charAt(1));
  }

  Wire.endTransmission(); //Ending the transfer
 }
 //Send a message to Slave 3 (song sequence)
void writer3(String sq){
  Wire.beginTransmission(3); //Start transferring data
  if(sq.length()==1) Wire.write(sq.charAt(0));//Song number difference
  else {
    Wire.write(sq.charAt(0));
    Wire.write(sq.charAt(1));
  }
  Wire.endTransmission(); //Ending the transfer

 }
```

Appendix B

This is the code for the second uno board (slave1):

```
//This uno is used as slave 1 to control the LED1602
#include <LiquidCrystal.h>//LED1602 display required library
#include <Wire.h>
#include <SD.h>//SD card library required
#include <SPI.h>
LiquidCrystal lcd(9,8,4,5,6,7);//Define LED1602
const int chipSelect = 10;
File myFile;
int num=1;//Stands for song order
bool start=false;
void setup() {
    lcd.begin(16,2);//Initialize the width and height of the LED1602 display
    Wire.begin(2);//Initialize the iic communication, as slave 1, the iic communication address is 2
    Wire.onReceive(receiveEvent);//This registers an event on the slave side that is fired when the slave receives data from the host
    Serial.begin(9600); //Serial communication with the computer
    attachInterrupt(0,pause,RISING);//Initialize the interrupt pin, the interrupt number is 0, the actual pin is D2,
    // and the interrupt trigger mode is RISING (falling edge trigger, that is, high level changes to low level).

}

void loop() {
    if(start) sd(num);
}
void receiveEvent(int howMany) {//An event that is fired when slave 1 receives data.
//This event takes an int (the number of bytes read from the host) and returns no value
    start=true;
    String s="";
    while(Wire.available()){
        char m=Wire.read();
        s+=(char)m;
        delay(10);
    }
    num=s.toInt();//Use the toInt() method to convert a String to an int
    sd(num);
```

```
}

void pause(){//Interrupt function
    lcd.clear();
    lcd.begin(16,2);
    lcd.setCursor(0,0);
    lcd.print("Song Pause");
    while(digitalRead(2) == HIGH){}
    lcd.clear();
}
void sd(int num){//Used to get lyrics from sd card
    int count=0;//A footer representing a row
    int start=0;//The first time node is considered separately
    String b="";//For accessing lyrics
    String time1="";//Time node 1
    String time2="";//Time node 2
    int time=0;//The difference between time node 1 and time node 2,
    //which is the duration of a lyric sentence appearing in LCD
    while (!Serial);
    Serial.print("Initializing SD card...");
    pinMode(chipSelect, OUTPUT);
    if (!SD.begin(chipSelect)) {
        Serial.println("initialization failed.");
        return;
    }
    Serial.println("initialization done.");
    myFile = SD.open("song"+(String)num+".txt");
    if (myFile) {
        Serial.println("song"+(String)num+".txt");
        while(myFile.available()){
            char a=myFile.read();
            if(start==0){
                if(a==(char)'[') count=0;
                if(0<count&count<9)    time1+=a;
                else if(count==9)    start=1;
            }
            if(count>9&&a!=(char)'[') b+=a;
            if(a==(char)'[') {
                count=0;
                count++;
                continue;
            }
            if(0<count&count<9)    time2+=a;
            time =timecount(time2)-timecount(time1);
            if(count==9&&start==1){
            lcd.clear();//Clear the screen and position your cursor to the upper-left corner of the screen
            lcd.begin(16,2);
            lcd.print(b);
            Serial.print(b);
            for(int i=0;i<13;i++){//Scrolling lyrics
                lcd.scrollDisplayLeft();
                delay(time/13);
            }
            time1=time2;
            time2="";
            b="";
            }
            count++;
            }

        myFile.close();
        Serial.println("done.");
    }
    else {
        Serial.println("error opening test.txt");
    }
}
int timecount(String time){//Used to convert time
    int times=0;
    times+=(int)time.charAt(1)*60000;
    int i=(int)time.charAt(3)*10+(int)time.charAt(4);
    times+=i*1000;
    int j=(int)time.charAt(6)*10+(int)time.charAt(7);
    times+=i*10;
    return times;
}
```

Appendix C

This is the code for the third uno board (slave2)：

```
// This uno serves as slave 2 for controlling OLED display and sound sensor (KV-037).
#include <Wire.h>
#include <Adafruit_GFX.h>//Parent library required for the display
#include <Adafruit_SSD1306.h>//Sublibraries required for the display
Adafruit_SSD1306 oled(128, 64, &Wire, -1);//Define the display
const int soundPin = A3;//As an analog interface for sound decibel acquisition
int num=1;//Stands for song order

void setup() {
  analogWrite(5,100);//Through pin 5, adjust the lcd1602 backlight.
  //Because pin 2 is not enough, pin 5 of machine 3 is used instead
  oled.begin(SSD1306_SWITCHCAPVCC,0X3C);//Initialize the display,0X3C is the address of the iic communication,
  //which is obtained by calling iic_Scanner
  Wire.begin(3);//As slave 2, the iic communication address is 3
  Wire.onReceive(receiveEvent);//This registers an event on the slave side that is fired when the slave receives data from the host
  Serial.begin(9600); //Initiate serial communication with the computer
}
void loop() {
  oled.clearDisplay();//Clear the oled cache
  oled.setTextSize(2);//Set the font size to 1
  oled.setTextColor(1);//Sets the font color, default 1 (white)
  oled.setCursor(0,0);//Sets the cursor position
  oled.print("sq:");//Shows the song order in English
  //There are 20 songs, and the corresponding number of songs is 1-20,
  //with 10 new songs and 10 old songs each, 1-10 for new songs and 11-20 for old songs
  oled.setCursor(35,0);//Show song order (1-20)
  oled.print(num);
  if(num<=10){//Identify old and new songs. Less than or equal to 10 is new
    oled.setCursor(90,0);
    oled.print("new");//Show new songs
    }
    else if(num>10){//Judge the old and new songs, more than 10 are old songs
       oled.setCursor(90,0);
       oled.print("old");//Display old songs
    }
    int value =analogRead(soundPin);//Detect the microphone decibel size
    Serial.println(value);//Display microphone volume
    delay(500);
    if(value<125){
    oled.setCursor(0,25);
    oled.print("low voice");
    }1*Micro SD card module
 6*10kΩ resistors
 1*sound sensor (KV-037)
 1*4Ω, 3W spea
    oled.setCursor(0,50);
    oled.print("db:");//Feedback sounds too small in English

    oled.setCursor(35,50);
    oled.print(value);//Display volume in English
    oled.display();
    delay(2000);
 }
```

Appendix D

Project consumables

3* UNO boards
2* bread boards
6*vertical micro switches
1*SV5W voice module
2*TF cards with 1GB memory
A number of single rows of needles
A number of public to public dupont lines
A number of public to mother dupont lines
A number of mother to mother dupont lines
A number of jumpers
1 solder wire

1*ISD1820 recording voice module
1*LCD1602
ker
1*8Ω, 0.5W speaker
1*OLED
2*PH2.0 terminal wires
1*soldering iron

Appendix E

Software libraries
<SoftwareSerial.h>//Use a soft serial port
<Wire.h>// Communicate using iic
<LiquidCrystal.h>// Libraries required for LED1602 display
<SD.h>//Libraries required for the SD module
<SPI.h>//Communicate using spi
<Adafruit_GFX.h>// parent library for oled display
<Adafruit_SSD1306.h>// Sublibraries for oled displays

Appendix F

Wiring instructions
1.KV-037: The sound sensor (KV-037) uses pin A0 connected to pin A3 of slave2

2.LCD1602:LCD1602 adopts 4-bit data line connection method, and uses D4 ~ D7 to transmit data. D4 to D7 of LCD1602 correspond to D4 to D7 of slave1, RS pin is connected to D9 of slave1, RW is connected to GND, E is connected to D8 of slave1, and V0 is connected to D5 of slave2

3.SD module: The MISO, MOSI, SCK, and CS of the SD module are connected to the ICSP pin on slave1

4.ISD1820: The ISD1820 recording voice module is only connected to VCC and GND

5.SV5W: The TX pin of the SV5W voice module is connected to pin D10 of the master, RX is connected to pin D11 of the master, GND and BUSY are connected to GND, and the serial port mode is configured as con1-0, con2-0, and con3- 1

6.The devices using IIC and SPI communication are connected to the corresponding IIC and SPI pins