**ELEC 291 Section 20C**

Lab 2

L2C

Team 2A

*Student name Student number Contribution percentage*

Andy Ruan 36863141 33.3%

Kevin Wong 32105132 33.3%

Clarence Su 36387132 33.3%

Contribution Summary:

Andy Ruan designed and wired the circuits along with creating the schematic in fritzing. He also produced the frequency constants from C4 to G5 as well as the note and duration arrays for the melodies used in the bonus. He also wrote the Introduction, Motivations and Hardware section of the report.

Kevin Wong wrote code for getting inputs from the users. He wrote code to setup the LED character array and Morse Code character array as well as adding extra feature code to play songs. He wrote the Software sections and Conclusion of the report.

Clarence Su wrote code to translate the input to Morse Code character array and display the string on the 7-segment LED and to play the sound on the Piezo. He also written code to turn on the RGB LEDs depending on duration of music notes for the extra feature.

**Introduction and motivations**

This report outlines the process, challenges, and solutions in which Team 2A used to complete lab 2 of ELEC 291-20C. This will be described in the Lab Description section of this report which is subdivided by the numbered parts of the lab and the type of work. In addition, three appendices will be included which provide a fritzing schematic of the circuit, a copy of the code, and a discussion of the application notes as listed in the lab 2 experiment instructions.

**Lab Description**

Lab 2 comprised of four parts:

1. Implementing a morse code generator
2. Display using the 7-segment LED
3. Add the Piezo buzzer
4. Bonus feature

These four parts resulted in a complete circuit which incorporated the Arduino UNO, an LED, a 7-segment LED, RGB LED, and a Piezo buzzer. The resulting circuit along with the Arduino code allows users to input a speed in words per minute (WPM) and to input a string to be translated into Morse Code. The Arduino will then output a buzzing noise created by the Piezo as well as display the character on the 7-segment LED. The functionality of this circuit was implemented through an Arduino program in which the majority of the code was written in parts 1 and 2.

**Implementing a morse code generator**

**Hardware**

On the hardware side, this part was similar to a part of lab 1 and merely involved connecting an LED to the Arduino for an visual cue that the software worked. Our team considered using the on-board LED, however decided against it as it was much easier to observe the external LED. No problems were encountered while wiring the circuit due to the familiarity of the task.

**Software**

On the software side, a character array was first implemented. Each character was given a Morse Code string that determines the number of dots and dashes it contains. A dot was represented as a 1, a dash was represented as a 3 and nothing was represented as 0. As an example, the character “A” has one dot and one dash so it would be represented as “13000000”. Each of these strings were 8 characters long because the longest Morse Code representation was the error which had 8 dots.

Afterwards, the user input for the speed in words per minute was implemented in order to determine the rate at which the Piezo makes sounds. Next, the team implemented a function to translate each input character into the element (the Morse Code string) in the character array described above.

A problem the team had when implementing the Morse Code generator was

an error with translating strings that contained more than one word.

An unknown character in the string was recognized as an error by the

program. To test, we added Serial.println() in the code to print out the output

of the program and compared it to the input. As a result, we realized that we

forgot to implement code to allow the program to recognize the empty space

between words.

**Display using the 7-segment LED**

**Hardware**

The wiring of the 7-segment LED was similar to that of the RGB LED from lab 1. The only difference was that there were two common terminals, anode in our case, and four cathodes for each common anode. These extra pins lead to problems as we ran out of space to wire the LEDs on the small breadboard without wire cross over. We decided to rebuild the circuit on a larger breadboard which was an inconvenience but not a major obstacle. It was at this step that we also learned that when using the Arduino for serial communications, pins 0 and 1 are dedicated to rx/tx and this caused issues when we attempted to print to the serial monitor and displaying the 7-segment LED until we figured out that we needed to shift the pin assignments up by two.

**Software**

To display characters, numbers and expressions on the LED, the team first created a character array that would present a distinct character. For instance, the letter “A” was represented by a string like: “11101110” which determines which LED segment to turn on.

Next, the team wrote code to cast the input string’s characters into integers because each character represented an integer according to the ASCII table. Then, the integer was translated to the string that contains only zeros and ones in the character array and passed it into another function that turns a particular LED segment on when the character is “1”.

One problem the team had was an unexpected behavior from the 7-segment LED. The correct amount of LED segments were turned but the position of the on segments mismatched with the character we wanted to display.

Initially, we thought the reason for this error was the logic implemented in

the software. After debugging the code, we were confident the

logic implemented. Therefore, we decided to test the pin assignments by

writing a small test program which turned on the LED segments

correspondingly. We found that the pin assignment for the 7-segment LED

in software was different from the connections on the Arduino and fixed it.

**Add the Piezo buzzer**

**Hardware**

The circuit for the Piezo buzzer is exactly the same as the LED circuit in part 1. This simple task encountered no problems and the buzzer behaved as expected. Most of the time taken up on the hardware side at this point involved rewiring the circuit to an acceptable standard.

**Software**

The team wrote a function that takes each Morse Code string and parses it to determine if a given character is a dot or dash and generates the sound for the dot using the Piezo. The sound duration was as outlined from the Lab Manual. Between dots is a one dot length delay, between dashes is a one dot length delay, between dots and dashes is a 3 dot length delay and between words is a 7 dot length delay.

**Extra Feature: Playing Melodies and using the RGB LED**

The extra feature created was mostly software code. The team translated music notes from several melodies into frequencies that could be used for the Piezo buzzer. The frequencies and the durations of the music notes were added into two separate arrays. Then the team wrote a function that would play the song by iterating through the two arrays to play the music note and duration using Arduino’s tone() function. The LED from part one was switched out for an RGB LED in order to give visual notice of the relative duration of these notes.

**Conclusions**

This lab allowed students to gain more skills with the Arduino. Firstly, students learned to parse string inputs from the user. Secondly, students learned to translate the string into a Morse Code string that would be useful in determining the amounts of dots and dashes in a given character. Lastly, students also learned to use the Piezo to generate sounds based on frequency as well as using their creativity to generate simple melodies.

Our team learned to produce code that is modular and clean by using good comments and by using functions instead of repeating multiple statements. Additionally, the team realized that constant testing of the software was extremely important. The team was able to eliminate a lot of errors by testing continuously instead of testing when every feature was implemented.

**References**

[1] https://en.wikipedia.org/wiki/ASCII

**Appendix I**

Below is a fritzing schematic of the circuit used in Lab 2.

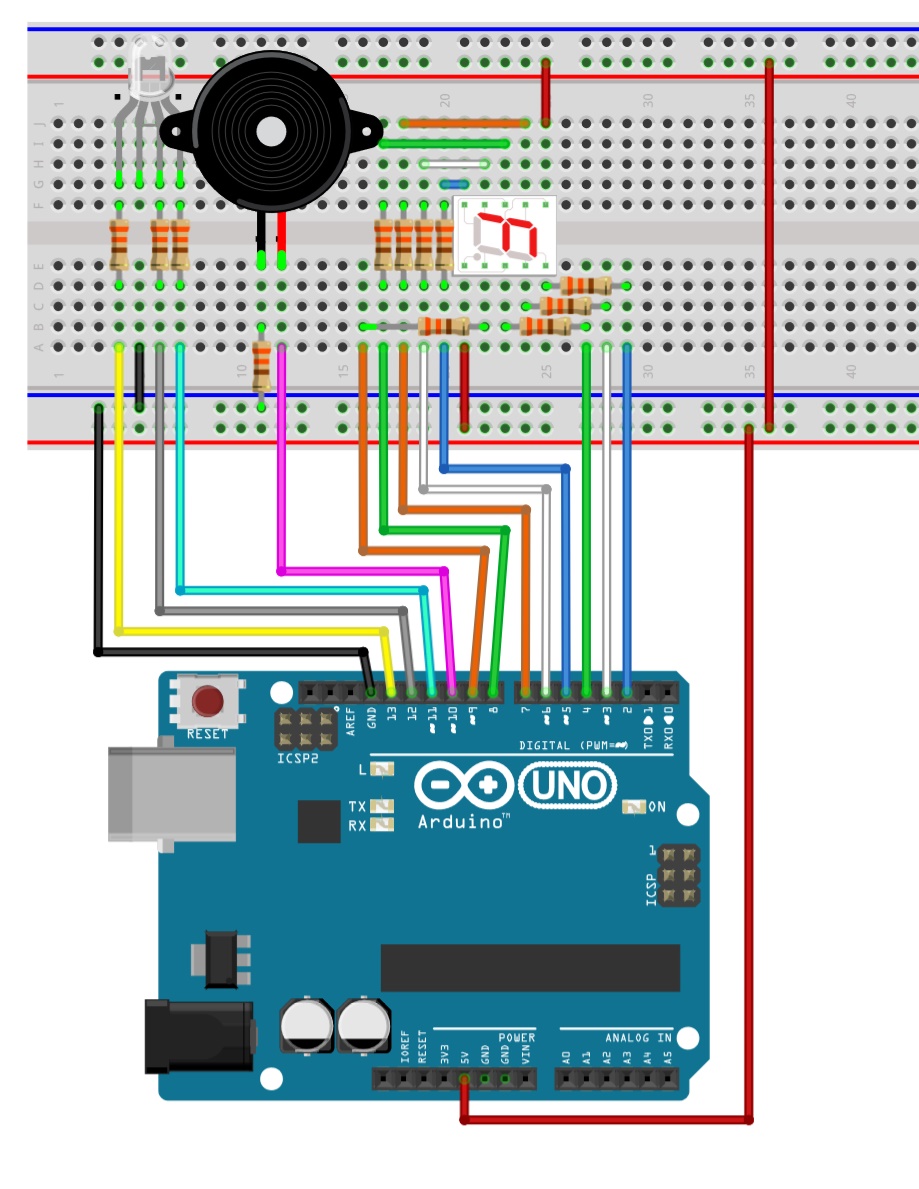


Figure 1: Frtizing schematic of circuit used in lab 2

**Appendix II**

Below is the code written for Lab 2.

//pin assignments  
const int LEDsegmentA = 2;  
const int LEDsegmentB = 3;  
const int LEDsegmentC = 4;  
const int LEDsegmentD = 5;  
const int LEDsegmentE = 6;  
const int LEDsegmentF = 7;  
const int LEDsegmentG = 8;  
const int LEDsegmentDP = 9;  
const int Buzzer = 10;  
const int singleLED = 12;  
const int RedLed = 11;  
const int BlueLed = 13;  
  
  
//user input variable & global flag  
int WPM = 0;  
String inputString; //global flag for inputString  
int choice; //global flag for choosing melody  
boolean inputReady = false; //global flag for input   
const int VOLUME = 50; //volume for the buzzor  
int duration; //length of dot  
char inputCharArray [100]; //used to stored the input  
  
void setup() {  
 //set up the pins for the LED  
 pinMode(LEDsegmentA, OUTPUT);  
 pinMode(LEDsegmentB, OUTPUT);  
 pinMode(LEDsegmentC, OUTPUT);  
 pinMode(LEDsegmentD, OUTPUT);  
 pinMode(LEDsegmentE, OUTPUT);  
 pinMode(LEDsegmentF, OUTPUT);  
 pinMode(LEDsegmentG, OUTPUT);  
 pinMode(LEDsegmentDP, OUTPUT);  
 pinMode(Buzzer, OUTPUT);  
 pinMode(singleLED, OUTPUT);  
 pinMode(RedLed, OUTPUT);  
 pinMode(BlueLed, OUTPUT);  
   
 Serial.begin(9600); // opens serial port, sets data rate to 9600 bps  
 boolean input = false; //no input for the set up yet  
 Serial.flush(); //remove stuff in our buff  
 //ask the user to input the speed  
 Serial.println("input the speed: ");  
 while(!input){  
 if (Serial.available()) {  
 // read the incoming byte:  
 WPM = Serial.parseInt();  
 input = true;  
 }  
 }  
 Serial.println("the speed you input is: ");  
 Serial.println(WPM);  
 duration = 1200/(WPM);  
}  
  
void loop() {  
 //turn off the LED first  
 digitalWrite(BlueLed, HIGH);  
 digitalWrite(RedLed, HIGH);  
 digitalWrite(singleLED, HIGH);  
 turnOffAllSegment();  
 //if there is no input string, ask the user for input  
 if(!inputReady){  
 Serial.println("Please enter string input to generate Morse Code");  
 }  
 while(!inputReady){  
 while(Serial.available()){  
 Serial.println("Your input is: ");  
 inputString = Serial.readString();  
 Serial.println(inputString);  
 inputReady=true;  
 if(inputReady=true){  
 break;  
 }  
 }  
 }  
  
 //parse the input string into char array  
 if(inputReady){  
 //Extra activity recogize the key word Melody and play the song that the user chooses  
 if(inputString == "Melody"){  
 inputReady = false;  
 boolean choiceMade = false;  
 Serial.println("Input an interger to choose one of the following melody to play:");  
 Serial.println("1. Star War Main Theme");  
 Serial.println("2. Star War Imperial March");  
 while(!choiceMade){  
 if (Serial.available()) {  
 // read the incoming byte:  
 choice = Serial.parseInt();  
 choiceMade = true;  
 }  
 }  
 Serial.println("Your input is: ");  
 Serial.println(choice);  
 if(choice==1){  
 playStarWarTheme();  
 }  
 else if(choice==2){  
 playStarWarMarch();  
 }  
 }  
   
 for(int index=0; index<inputString.length(); index++){  
 inputCharArray[index] = inputString.charAt(index);  
 }  
 }  
  
 //translate the the char into morse code and display on the LED as well  
 if(inputReady){  
 for(int index=0; index<inputString.length();index++){  
// LedTranslation(inputCharArray[index]);  
 Serial.println(inputCharArray[index]);//for test  
 morseCodeTranslation(inputCharArray[index]);  
 }  
 inputReady=false;  
 delay(100);  
 }  
   
}  
  
//LED encode for the characters  
char LedSegmentChar[][9] =   
 {  
 "11101110", // A - 0  
 "00111110", // B - 1  
 "10011100", // C - 2  
 "01111010", // D - 3  
 "10011110", // E - 4  
 "10001110", // F - 5  
 "11110110", // G - 6  
 "01101110", // H - 7  
 "00001100", // I - 8  
 "01111000", // J - 9  
 "01101110", // K - 10  
 "00011100", // L - 11  
 "10101000", // M - 12  
 "00101010", // N - 13  
 "11111100", // O - 14  
 "11001110", // P - 15  
 "11100110", // Q - 16  
 "00001010", // R - 17  
 "10110110", // S - 18  
 "00011110", // T - 19  
 "01111100", // U - 20  
 "00111000", // V - 21  
 "01010100", // W - 22  
 "01101110", // X - 23  
 "01110110", // Y - 24  
 "11011010", // Z - 25  
 "11111100", // 0 - 26  
 "01100000", // 1 - 27  
 "11011010", // 2 - 28  
 "11110010", // 3 - 29  
 "01100110", // 4 - 30  
 "10110110", // 5 - 31  
 "10111110", // 6 - 32  
 "11100000", // 7 - 33  
 "11111110", // 8 - 34  
 "11110110", // 9 - 35  
 "00000001", //period - 36  
 };  
  
//display the give char array into LED  
void writeLED(char\* a){  
 int pinOffset = 2;  
 for(int index=0; index<8; index++){  
 if(a[index]=='1'){  
 digitalWrite(index+pinOffset, LOW);  
 }  
 else{  
 digitalWrite(index+pinOffset, HIGH);  
 }  
 }  
 }  
  
  
//turn off all the led Segment  
void turnOffAllSegment(){  
 digitalWrite(LEDsegmentA,HIGH);  
 digitalWrite(LEDsegmentB,HIGH);  
 digitalWrite(LEDsegmentC,HIGH);  
 digitalWrite(LEDsegmentD,HIGH);  
 digitalWrite(LEDsegmentE,HIGH);  
 digitalWrite(LEDsegmentF,HIGH);  
 digitalWrite(LEDsegmentG,HIGH);  
 digitalWrite(LEDsegmentDP,HIGH);  
 }  
  
//morse code encode for the character  
char morseCodeArray[][9] =   
{  
 "13000000", //A  
 "31110000", //B  
 "31310000", //C  
 "31100000", //D  
 "10000000", //E  
 "11310000", //F  
 "33100000", //G  
 "11110000", //H  
 "11000000", //I  
 "13330000", //J  
 "31300000", //K  
 "13110000", //L  
 "33000000", //M  
 "31000000", //N  
 "33300000", //O  
 "13310000", //P  
 "33130000", //Q  
 "13100000", //R  
 "11100000", //S  
 "30000000", //T  
 "11300000", //U  
 "11130000", //V  
 "13300000", //W  
 "31130000", //X  
 "31330000", //Y  
 "33110000", //Z  
 "33333000", //0  
 "13333000", //1  
 "11333000", //2  
 "11133000", //3  
 "11113000", //4  
 "11111000", //5  
 "31111000", //6  
 "33111000", //7  
 "33311000", //8  
 "33331000", //9  
 "13131300", //Full Stop (.)  
 "33113300", //Comma (,)  
 "33311100", //Colon (:)  
 "31111300", //Hyphen (-)  
 "11111111" //Error  
};  
  
//translate the char into LED and morseCode  
void morseCodeTranslation(char a){  
 int value = (int) a;  
 if(value >= 65 && value <= 90){  
 value = value -65;  
 writeLED(LedSegmentChar[value]);  
 writeBuzzer(morseCodeArray[value]);  
 turnOffAllSegment();  
 }  
 // a to z  
 else if(value >=97 && value <= 122){  
 value = value -97;  
 writeLED(LedSegmentChar[value]);  
 writeBuzzer(morseCodeArray[value]);  
 turnOffAllSegment();  
 }  
 // 0 to 9  
 else if(value >= 48 && value <= 57){  
 value = value -22;  
 writeLED(LedSegmentChar[value]);  
 writeBuzzer(morseCodeArray[value]);  
 turnOffAllSegment();  
 }  
 else if(value == 46){  
 //period  
 writeLED(LedSegmentChar[36]);  
 writeBuzzer(morseCodeArray[36]);  
 turnOffAllSegment();  
 }  
 else if(value == 44){  
 //comma  
 turnOffAllSegment();  
 writeBuzzer(morseCodeArray[37]);  
 }  
 else if(value == 58){  
 turnOffAllSegment();  
 writeBuzzer(morseCodeArray[38]);  
 }  
 else if(value == 45){  
 turnOffAllSegment();  
 writeBuzzer(morseCodeArray[39]);  
 }  
 else if(value == 32){  
 delay(7\*duration);  
 }  
 else {  
 turnOffAllSegment();  
 writeBuzzer(morseCodeArray[40]);  
 }  
   
 }  
  
//make the buzzor sound according the character input  
void writeBuzzer(char\* a){  
 //iterate through the led segments  
 for(int index=0; index < 8; index++){  
 delay(1\*duration);  
 buzz(a[index]);  
 }  
 delay(3\*duration);//duration between characters  
 }  
  
//make the buzzor sound based on the dot or dash  
void buzz(char a){  
 //todo add a delay between characters  
 if(a == '1'){  
 analogWrite(Buzzer, VOLUME);  
 digitalWrite(singleLED, LOW);  
 Serial.println('1'); //for test purpose  
 delay(duration);  
 digitalWrite(singleLED, HIGH);  
 analogWrite(Buzzer, 0);  
 }  
 else if(a == '3'){  
 analogWrite(Buzzer, VOLUME);  
 digitalWrite(singleLED, LOW);  
 Serial.println('3'); //for test purpose  
 delay(3\*duration);  
 digitalWrite(singleLED, HIGH);  
 analogWrite(Buzzer, 0);  
 }  
 else if(a == '0'){  
 Serial.println('0'); //for test purpose  
 }  
 }  
  
/\*  
 \* Extra activity section  
 \*/  
 //note frequency  
int note\_c5 = 523;  
int note\_c5s = 554;  
int note\_c4 = 262;  
int note\_c4s = 277;  
int note\_d4 = 294;  
int note\_d4s = 311;  
int note\_d5 = 587;  
int note\_d5s = 622;  
int note\_e4 = 330;  
int note\_e5 = 659;  
int note\_f4 = 349;  
int note\_f4s = 370;  
int note\_f5 = 698;  
int note\_f5s = 740;  
int note\_g4 = 392;  
int note\_g4s = 415;  
int note\_g5 = 784;  
int note\_a4 = 440;  
int note\_a4s = 466;  
int note\_b4 = 494;  
  
//note of the song  
int noteStarWar[]={note\_d4, note\_d4, note\_d4, note\_g4, note\_d5, note\_c5, note\_b4, note\_a4, note\_g5, note\_d5, note\_c5, note\_b4, note\_a4,  
 note\_g5, note\_d5, note\_c5, note\_b4, note\_c5, note\_a4};  
//duration of each note corresponding to the note array  
int durationStarWar[] = {1, 1, 1, 6, 6, 1, 1, 1, 6, 3, 1, 1, 1, 6, 3, 1, 1, 1, 4};  
  
//play the starWar melody  
void playStarWarTheme(){  
 int pace = 450;  
 for(int index = 0; index<19; index++){  
 if(durationStarWar[index]==1){  
 digitalWrite(RedLed, LOW);  
 }  
 else if(durationStarWar[index]==3){  
 digitalWrite(BlueLed, LOW);  
 }  
 else if(durationStarWar[index]==4){  
 digitalWrite(singleLED, LOW);  
 }  
 else{  
 digitalWrite(BlueLed, LOW);  
 digitalWrite(RedLed, LOW);  
 }  
 //select what character to display on LED basing on the note  
 if(noteStarWar[index]==note\_c4||noteStarWar[index]==note\_c4s||noteStarWar[index]==note\_c5||noteStarWar[index]==note\_c5s){  
 writeLED(LedSegmentChar[2]);  
 }  
 else if(noteStarWar[index]==note\_d4||noteStarWar[index]==note\_d4s||noteStarWar[index]==note\_d5s||noteStarWar[index]==note\_d5){  
 writeLED(LedSegmentChar[3]);  
 }  
 else if(noteStarWar[index]==note\_e4||noteStarWar[index]==note\_e5){  
 writeLED(LedSegmentChar[4]);  
 }  
 else if(noteStarWar[index]==note\_f4||noteStarWar[index]==note\_f4s||noteStarWar[index]==note\_f5s||noteStarWar[index]==note\_f5){  
 writeLED(LedSegmentChar[5]);  
 }  
 else if(noteStarWar[index]==note\_g4||noteStarWar[index]==note\_g4s||noteStarWar[index]==note\_g5){  
 writeLED(LedSegmentChar[6]);  
 }  
 else if(noteStarWar[index]==note\_a4||noteStarWar[index]==note\_a4s){  
 writeLED(LedSegmentChar[0]);  
 }  
 else if(noteStarWar[index]==note\_b4){  
 writeLED(LedSegmentChar[1]);  
 }  
 tone(Buzzer, noteStarWar[index], durationStarWar[index]\*pace);  
 delay(400);  
 digitalWrite(BlueLed, HIGH);  
 digitalWrite(RedLed, HIGH);  
 digitalWrite(singleLED, HIGH);  
 turnOffAllSegment();  
 }  
 }  
  
int sizeMarch = 38;  
//note array for imperial march  
int noteMarch[] = {note\_g4, note\_g4, note\_g4, note\_d4s, note\_a4s, note\_g4, note\_d4s, note\_a4s, note\_g4,   
 note\_d5, note\_d5, note\_d5, note\_d5s, note\_a4s, note\_f4s, note\_d4, note\_a4s, note\_g4,   
 note\_g5, note\_g4, note\_g4, note\_g5, note\_f5s, note\_f5, note\_e5, note\_g4s, note\_c5s, note\_b4,  
 note\_a4s, note\_d4s, note\_f4s, note\_e4, note\_f4, note\_a4s, note\_g4, note\_b4, note\_d5  
 };  
//duration of the note corresponding to the note array  
int durationMarch[] = {4, 4, 4, 3, 1, 4, 3, 1, 8,  
 4, 4, 4, 3, 1, 4, 3, 1, 8,  
 4, 3, 1, 4, 3, 1, 4, 4, 4, 3, 1,  
 4, 4, 4, 3, 1, 4, 3, 1, 6  
 };  
void playStarWarMarch(){  
 int pace = 255;  
 for(int index = 0; index<sizeMarch; index++){  
 if(durationMarch[index]==1){  
 digitalWrite(RedLed, LOW);  
 }  
 else if(durationMarch[index]==3){  
 digitalWrite(BlueLed, LOW);  
 }  
 else if(durationMarch[index]==4){  
 digitalWrite(singleLED, LOW);  
 }  
 else if(durationMarch[index]==6){  
 digitalWrite(BlueLed, LOW);  
 digitalWrite(RedLed, LOW);  
 }  
 else{  
 digitalWrite(BlueLed, LOW);  
 digitalWrite(RedLed, LOW);  
 digitalWrite(singleLED, LOW);  
   
 }  
  
 //select what character to display on LED basing on the note  
 if((noteStarWar[index]==note\_c4)||(noteStarWar[index]==note\_c4s)||(noteStarWar[index]==note\_c5)||(noteStarWar[index]==note\_c5s)){  
 writeLED(LedSegmentChar[2]);  
 }  
 else if((noteStarWar[index]==note\_d4)||(noteStarWar[index]==note\_d4s)||(noteStarWar[index]==note\_d5s)||(noteStarWar[index]==note\_d5)){  
 writeLED(LedSegmentChar[3]);  
 }  
 else if((noteStarWar[index]==note\_e4)||(noteStarWar[index]==note\_e5)){  
 writeLED(LedSegmentChar[4]);  
 }  
 else if((noteStarWar[index]==note\_f4)||(noteStarWar[index]==note\_f4s)||(noteStarWar[index]==note\_f5s)||(noteStarWar[index]==note\_f5)){  
 writeLED(LedSegmentChar[5]);  
 }  
 else if((noteStarWar[index]==note\_g4)||(noteStarWar[index]==note\_g4s)||(noteStarWar[index]==note\_g5)){  
 writeLED(LedSegmentChar[6]);  
 }  
 else if((noteStarWar[index]==note\_a4)||(noteStarWar[index]==note\_a4s)){  
 writeLED(LedSegmentChar[0]);  
 }  
 else if(noteStarWar[index]==note\_b4){  
 writeLED(LedSegmentChar[1]);  
 }  
   
 tone(Buzzer, noteMarch[index], durationMarch[index]\*pace);  
 delay(450);  
 digitalWrite(BlueLed, HIGH);  
 digitalWrite(RedLed, HIGH);  
 digitalWrite(singleLED, HIGH);  
 turnOffAllSegment();  
 }  
 }