[[1]](#footnote-1)

Assignment 3 – Song Composer

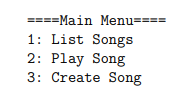
Yvan Gihoza

***Overview*—the third assignment was to give us a practice with designing in Arduino so we can gain experience with microcontroller hardware, serial communication and pulse width modulation. Where we had to design a program that could play, create and store up to four songs.**

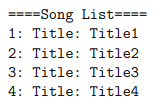
1. INTRODUCTION

T

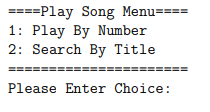
He program was supposed to start with a menu where the user had to choose from 3 options being, listing the saved songs, creating one and playing one.



With the list songs choice the user was prompted to another menu showing all the stored songs.



Play song will bring a menu asking you if you want to insert the number of the song you want to play of you want to type in the whole sting.



And for the last choice it just let the user input the string (notes) of the song he/she wants to store.

1. Application Design

I implemented my program function by function as I found it was the best way to do it, I created the menu program then, other functions which were needed by other functions in order to work for example like play song is dependent on play note so before starting it, I had to make sure that my play note function was working fine.

1. Results

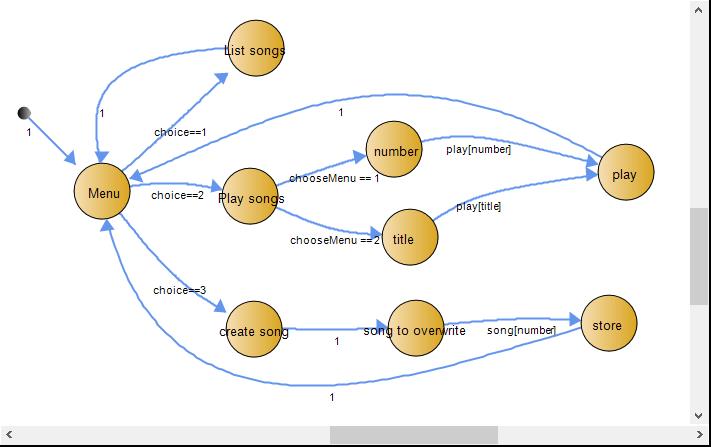
I tested function by function to make sure each was working before implementing a new one. For the third option which was “create song” I was not able to save the title as I believe I was not able to finish the matching function.

.

1. Conclusion

I encountered issues with my board as it broke at some point and I had to wait for one of my classmate to finish so I could use his, other than that my program was performing the basic requirements.

1. State Machine



1. CODE

/\* Yvan Gihoza

\* 010785599

\* Assignment #3

\*/

#define USER\_LINE\_MAX 128

char userLine[USER\_LINE\_MAX];

#define NUMBER\_OF\_SONGS 4

#define MAX\_SONG\_LENGTH 64

#define NOTE\_A 0

#define NOTE\_B 1

#define NOTE\_C 2

#define NOTE\_D 3

#define NOTE\_E 4

#define NOTE\_F 5

#define NOTE\_G 6

#define NOTE\_R 7

#define STR\_LENGTH 100

#define FREQ\_A4\_HZ 440

#define FREQ\_B4\_HZ 493

#define FREQ\_C4\_HZ 261

#define FREQ\_D4\_HZ 293

#define FREQ\_E4\_HZ 329

#define FREQ\_F4\_HZ 349

#define FREQ\_G4\_HZ 392

#define FREQ\_R4\_HZ 0

#define HALFPERIOD\_G4\_US 1276 // delay in microseconds

char songTitle [NUMBER\_OF\_SONGS][STR\_LENGTH]={" Title1"," Title2"," Title3"," Title4"};

// "B2 A2 G2 R0 R0 R0"

uint8\_t song1[NUMBER\_OF\_SONGS][MAX\_SONG\_LENGTH] =

{{(NOTE\_B<<5)+2 ,(NOTE\_A<<5) +2,(NOTE\_G<<5) +2 },{NOTE\_R<<5},{NOTE\_R<<5},{NOTE\_R<<5}};

uint8\_t song[NUMBER\_OF\_SONGS][MAX\_SONG\_LENGTH] =

{{(NOTE\_B<<5)+1 ,(NOTE\_A<<5) +1,(NOTE\_G<<5) +1 },{(NOTE\_R<<5)+8},{(NOTE\_R<<5)+0},{NOTE\_R<<5}};

uint8\_t song2[NUMBER\_OF\_SONGS][MAX\_SONG\_LENGTH] =

{{(NOTE\_B<<5)+2 ,(NOTE\_A<<5) +2,(NOTE\_G<<5) +2 },{NOTE\_R<<5},{NOTE\_R<<5},{NOTE\_R<<5}};

uint8\_t song3[NUMBER\_OF\_SONGS][MAX\_SONG\_LENGTH] =

{{(NOTE\_B<<5)+2 ,(NOTE\_A<<5) +2,(NOTE\_G<<5) +2 },{NOTE\_R<<5},{NOTE\_R<<5},{NOTE\_R<<5}};

// some initial song with notes

const char menuMain[] = "\*\*\*Main Menu\*\*\*\n1: List Song\n2: Play Song\n3: Create Song\n";

const char menuPlay[] = "Play Menu\nSearch By Title\nNumber\n";

char my\_buffer[50] = "";

//

void stripEOL(char string[], int n)

{

char \*token;

int i;

for(i = 0; i < n; i++)

{

if(string[i] == '\n' || string[i] == '\r')

string[i] = '\0';

}

}

//serial code

void ReadString()

{

// put your main code here, to run repeatedly:

int i = 0;

int keep\_reading = 1;

while(keep\_reading){

while(Serial.available() == 0);

my\_buffer[i] = Serial.read();

if(my\_buffer[i] == '\n'){

keep\_reading = 0;

}

if(i == 49){

keep\_reading = 0;

}

i+=1;

}

stripEOL(my\_buffer,50);

}

//menu function

uint8\_t DisplayMenu(const char menu[])

{

// Make a duplicate

char \* copy = strdup(menu);

char \* tokenMenu = strtok(copy, "\n");

while( tokenMenu != NULL)

{

Serial.println(tokenMenu);

tokenMenu = strtok(NULL, "\n");

}

Serial.println("======================");

Serial.println("Please Enter Choice:");

ReadString();

int chooseMenuType = my\_buffer[0];

switch(chooseMenuType)

{

//listing the songs

case '1':

{

ListSongs(songTitle);

ReadString();

break;

}

//playing the songs

case '2':

{

Serial.println();

Serial.println("====Play Song Menu====");

Serial.println("1: Play By Number ");

Serial.println("2: Search By Title");

Serial.println("======================");

Serial.println("Please Enter Choice:");

Serial.println();

Serial.println();

ReadString();

int choice = my\_buffer[0];

if(choice == '1')

{

ListSongs(songTitle);

ReadString();

int songToPlay = my\_buffer[0];

PlaySong(song[songToPlay]);

}

if(choice == '2')

{

Serial.println("Please Enter the name of the song");

ReadString();

int songToPlay = my\_buffer[0];

PlaySong(song[songToPlay]);

}

break;

}

//creating song

case '3':

{

Serial.println("which son would you like to overwrite (1-4): ");

ReadString();

int songToReplace = my\_buffer[0];

Serial.println("Enter the song: ");

ReadString();

String title = my\_buffer;

StoreSong(song[songToReplace], title.c\_str());

Serial.println("Song was created: ");

Serial.println();

ListSongs(songTitle);

ReadString();

break;

}

}

}

//List Songs Function

void ListSongs(char songTitle[NUMBER\_OF\_SONGS][STR\_LENGTH])

{

int i = 0;

Serial.println();

Serial.println("====Song List====");

for(; i < NUMBER\_OF\_SONGS; i++)

{

Serial.print(i+1);

Serial.print(":");

Serial.print("Title: ");

Serial.println(songTitle[i]);

}

Serial.println();

Serial.println();

}

//Play Song Function

void PlaySong(uint8\_t song[])

{

//uint8\_t duration = 0;

uint8\_t store[50];

uint8\_t character;

uint8\_t duration;

int i=0;

for(; i < MAX\_SONG\_LENGTH; i++)

{

character = UnpackNoteLetterASCII(store[i]);

duration = UnpackNoteDuration(store[i]);

//while(character != 'R' && duration != 0)

if((character != 'R' || character != 'r') && duration != 0)

PlayNote(character, duration);

if(character == 'R' || character == 'r')

continue;

else

return;

}

}

//Pack function

uint8\_t PackNote(char letterASCII, uint8\_t duration)

{

uint8\_t packNote = 0 ;

if(letterASCII == 'a' || letterASCII == 'A')

packNote = (NOTE\_D << 5) | (duration);

else if(letterASCII == 'b' || letterASCII == 'B')

packNote = (NOTE\_D << 5) | (duration);

else if(letterASCII == 'c' || letterASCII == 'C')

packNote = (NOTE\_D << 5) | (duration);

else if(letterASCII == 'd' || letterASCII == 'D')

packNote = (NOTE\_D << 5) | (duration);

else if(letterASCII == 'e' || letterASCII == 'E')

packNote = (NOTE\_D << 5) | (duration);

else if(letterASCII == 'f' || letterASCII == 'F')

packNote = (NOTE\_D << 5) | (duration);

else if(letterASCII == 'g' || letterASCII == 'G')

packNote = (NOTE\_D << 5) | (duration);

else if(letterASCII == 'r' || letterASCII == 'R')

packNote = (NOTE\_D << 5) | (duration);

else

packNote = 0xf | duration;

return packNote;

}

//Unpack function

uint8\_t UnpackNoteLetterASCII(uint8\_t packedNote)

{

return packedNote>>5;

}

//

uint8\_t UnpackNoteDuration(uint8\_t packedNote)

{

return(packedNote & 31);

}

//letter

bool checkChar(char letter)

{

if(letter == 'a' || letter == 'A' ||letter == 'b' || letter == 'B' ||letter == 'c' || letter == 'C' ||letter == 'd' || letter == 'D' ||letter == 'e' || letter == 'E' ||letter == 'f' || letter == 'F' ||letter == 'g' || letter == 'G' ||letter == 'r' || letter == 'R' )

return true;

}

//number

bool checkNum(char x)

{

if(x >= '0' && x <= '9')

return true;

}

//Store song function

void StoreSong(uint8\_t song[], const char songString[])

{

uint8\_t packNote;

char character;

uint8\_t duration;

int cFlag=0;

int dFlag = 0;

int i = 0;

for(; i < strlen(songString); i++)

{

if(cFlag && dFlag)

{

int n = 0;

packNote = PackNote(character, duration);

cFlag = 0;

dFlag = 0;

song[n] = packNote;

n++;

}

//check for the char if one set a flag

if(checkChar(songString[i]))

{

character = songString[i];

cFlag = 1;

}

//check for numbers

if(checkNum(songString[i]))

{

if(checkNum(songString[i+1]))

{

char x[] = {songString[i]};

char y[] = {songString[i+1]};

strcat(x, y);

duration = atoi(x);

//dFlag = 1;

}

else

{

duration = songString[i];

//dFlag = 1;

}

dFlag = 1;

}

}

}

//Play Note Functions

void PlayNote(uint8\_t letterASCII, uint8\_t quarters)

{

int period ;

int halfPeriodUS;

int numIterations;

//A

if(letterASCII == 0)

{

period = (1000000/FREQ\_A4\_HZ);

halfPeriodUS = period/2;

numIterations = (FREQ\_A4\_HZ/4)\*quarters;

}

//B

else if(letterASCII == 1)

{

period = (1000000/FREQ\_B4\_HZ);

halfPeriodUS = period/2;

numIterations = (FREQ\_B4\_HZ/4)\*quarters;

}

//C

else if(letterASCII == 2)

{

period = (1000000/FREQ\_C4\_HZ);

halfPeriodUS = period/2;

numIterations = (FREQ\_C4\_HZ/4)\*quarters;

}

//D

else if(letterASCII == 3)

{

period = (1000000/FREQ\_D4\_HZ);

halfPeriodUS = period/2;

numIterations = (FREQ\_D4\_HZ/4)\*quarters;

}

//E

else if(letterASCII == 4)

{

period = (1000000/FREQ\_E4\_HZ);

halfPeriodUS = period/2;

numIterations = (FREQ\_E4\_HZ/4)\*quarters;

}

//F

else if(letterASCII == 5)

{

period = (1000000/FREQ\_F4\_HZ);

halfPeriodUS = period/2;

numIterations = (FREQ\_F4\_HZ/4)\*quarters;

}

//G

else if(letterASCII == 6)

{

period = (1000000/FREQ\_G4\_HZ);

halfPeriodUS = period/2;

numIterations = (FREQ\_G4\_HZ/4)\*quarters;

}

//R

else if(letterASCII == 7)

{

//period = (1000000/1000);

halfPeriodUS = 500;

numIterations = (1000/4)\*quarters;

}

int i;

for(i = 0; i < numIterations; i++)

{

if(letterASCII!= 7)

analogWrite(5, 8);

delayMicroseconds(halfPeriodUS);

analogWrite(5, 0);

delayMicroseconds(halfPeriodUS);

}

}

//Matching Function

int MatchScore(const char countQueryString[], const char templates[])

{

/\*int i;

for(i = 0; i < sizeof(countQueryString); i++)

{

if(strcmp(countQueryString, templates))

i++;

}

return i;\*/

int i = 0;

char \* countQuery = strdup(countQueryString);

char \* tmplt = strdup(templates);

char \* tokenCountQuery = strtok(countQuery, " ");

while( tokenCountQuery != NULL)

{

if(strcmp(countQuery, tmplt) == 0)

i++;

}

return i;

}

//////////////////////////////////////////////////////////////////////

void setup() {

// put your setup code here, to run once:

Serial.begin(115200);

while(!Serial);

}

void loop() {

DisplayMenu(menuMain);

}

1. [↑](#footnote-ref-1)