Music Player

ECE 303 Section B

Design Project

Jonathan Kosir

Sebastian Monnin

Thomas Lawson

Kalene Kelly

Project Description

The music player project was designed to accomplish two tasks that involved playing audio files. The first task was to create a circuit that resembled a single button digital audio player that was capable of reading digital files from a micro SD card and play them over an audio jack. This circuit would be controlled by a single button that, when pressed, would play or pause the current song being played. If the button was held down for a period of one second, the next song would be read. In order for the arduino unit to read specific audio files, they had to be converted into a 8-bit WAV file which were then transferred onto the SD card. The second task of this project was to create a program and circuit that would play audio files accessed by typing into a browsers URL bar an assigned IP address given to the Arduino. These songs, when clicked on, will be output by the PWM in the Arduino and can be heard by plugging in any device to the 2.5mm headphone jack in the circuit.

Detailed Description

Desired Results

The main priority goals of this project were to be able to make a single circuit that would read digital audio files from a formatted micro SD card and be able to play and control the audio files by a button as well as from a website. When we pressed the button we wanted the ability to pause and play the songs as well as skip to the next song. With the portion of the audio player that was connected to the internet we wanted to use a website to display the songs on the SD card and then control what audio file that would be played by the Arduino's by a simple click to choose your song of choice. Smaller objectives of this project were to implement the option to go to the previous by the use of a button press, and being able to play, pause and skip songs with the web-controlled player.

Simple Audio Player

The simple music player is a circuit controlled by an Arduino Uno microprocessor that utilizes a micro SD card by reading digital audio files that were saved into the Arduino's memory library. The circuit is able to play these digital files through the use of a digital to analog conversion (DAC). In order to convert the digital outputs of the Arduino unit from 0's and 1's into a readable analog voltage, we used the Arduino's PWM, pulse width modulation. Pulse width modulation uses a high clock frequency in a square wave format that spends half of its time at the digital outputs of 0 and 1 per clock cycle. This allows us to modulate a waveform output given by the arduino microprocessor by averaging the digital output values into an analogue voltage that is proportional to the clock cycles of the waveform. The final result of the averaged voltage format allows us to use analogue voltage approximation to play the different songs that are stored in the digital files.

With the average WAV file containing 16-bits, the Arduino only has an 8-bit bus and the operating frequency to run a 19 kHz sample rate within its audio library. This provides us with a lot of limitations regarding the size of audio files and the amount that can be stored within the library. Because most WAV files are 16-bits in size, a program was used in order to convert audio files into and 8-bit 19 kHz WAV file called "InstallWavConverter.reg". This program allowed us to select any audio file and convert it into a format that would be compatible with the Arduino microprocessor by right clicking on the desired file and selecting the "convert to APC/Arduino format" option. These files were then copied into the main root folder of a FAT32 formatted SD card. In order for the code to read the SD card, the card has to be modified into the correct format. This was done by the use of a program called "tmrpcm", this program reformates the given SD card in order for the Arduino unit to fully recognize it.

The circuit we built used an Ethernet Shield that contained a micro SD card slot. This circuit amplified the analogue voltage output given by the Arduino into a viable source that could be used by the 3.5mm stereo socket. The button in the circuit sends an input voltage back to the Arduino, if it is briefly pushed, the microprocessor stop reading from the SD card and provides no voltage signal to the circuit, if the button is pressed again, the digital audio file will pick up where it left off and the signal will continue. If the button is held for a period longer than one second, the next audio file will be read instead of the previous; this demonstrates the circuit's ability to skip desired songs in the arduino library. On the circuit, there is a LED that lights up once an analogue voltage is sent through the circuit, if there is no digital file being read by the processor, then no voltage will be sent through the circuit and the LED will not light up.

Audio Player Via Internet

For this part of the project we used an online library called SimpleSDAudio by Lutz Lisseck. Lisseck's library uses assembly code which is different from the C++ derived code that the Arduino normally uses. This allows the Arduino to have more processing speed which enables us to use the SD card alongside the Ethernet, without too many problems. The program uses two 512-byte loop segments, one is being read while the other is being played. Once the 512-byte loop being played is finished the one being read begins to play and a new second 512-byte loop starts to be read. One of the limitations we ran into is both the Micro SD card reader and the Ethernet port operate using the SPI bus, and the Arduino only allows one to access it at a time. Luckily Lisseck's library takes care of that problem for us by designating appropriate times for each one to use the bus. For instance the server is not always running, the Arduino listens for a client call to the server. Once the Arduino hears a client call the Ethernet is given the priority of the bus to display the needed information to the user's browser. As soon as a link is pressed to play a song the Arduino outputs the information through the Ethernet to switch pages and show what song is playing, then the priority of the bus is given back the SD card so the Arduino can read the files to play the song.

One of the main problems with the library for us is that when using the Arduino Uno the primary PWM outputs are assigned to D9 and D10 and the secondary is PWM is D5 and D6. For us to play distinguishable music we must access the primary PWM outputs. With the Ethernet shield we are using when the Ethernet is listening for client calls it locks up outputs 9 and 10. Therefore we are not able to output music through those ports. So to create distinguishable music we must change the wiring.c file that is standard with the Arduino program to make outputs 5 and 6 our new primary PWM outputs. This creates another limitation because of this we can only output 8 bit stereo sound instead of the 16 bit WAV files we were able to output with the button player. So like the button player we had to convert the files to a different format so they are runnable on the Arduino. Luckily the rest of our project comes pretty simple using Lisseck's library. The library, once understood, is easily used to output HTML. With that HTML we were able to create our own webpage for the songs on the Arduino. Just like our button player this library allows us to read music off our SD card to play over our speaker component, though the libarary is limited and the library used in the button player has more song manipulation (ie: play, pause, repeat etc) abilities than the Lissecks's.

Again we were using the Arduino's PWM output to change our digital audio into analog voltage approximation. We fed this through a resistor divider network which acts as a basic first order low pass filter to knock off some of the PWM's clock signal to get rid of some of the static and leave just the audio signal we want. Not only that, it also acts as an amplifier for our speaker component.

Conclusion

Our goals which were highlighted above, at the beginning of the detailed section, were great and would have given us a very pleasing result. In the end we completed our bigger, main, goals and missed a few of our smaller goals. Our project turned out very well and we were very happy with it. Many things turned out much differently than we had expected and if given more time would change some things and add more functionality to it.

Though the goal of our project was to create a two-in-one device we ended up with two separate devices. The small programmable memory on the Arduino Uno caused a problem for us and we had to separate the two ideas into two separate devices. If we would have realized this earlier in the implementation process we might have combined two Arduino units or gotten a bigger Arduino like the Arduino Mega.

Separately both devices turned out great. Both circuits worked and did what we were aiming for except for some small functionality. We hit our main goal, though missed some of our smaller goals. With the button player we hit our main goal of being able to play music off of a micro SD card and we were able to play, pause, and go to the next song via a button press. If given more time we would have liked to have a restart song function as well as a previous song function. Also for the button player we could have gone a different route and instead of making the commands a single button press we could have added one or two more buttons and had different buttons do different things.

For our server player there was many things we would have liked to do but ran out of time to implement. We hit our base goal of displaying the songs that were on the micro SD card on a web page via an assigned IP address and then using a user's click on a link to send a command over a home network connection to the Arduino. We would have loved to add more usability and graphical design to our web page. Functionality wise we would have had the songs play continuously (ie. If a song ends the next song on the list begins to play), we would have loved to have a previous song, next song, pause song, and play song, button on the web page. We also would have liked to make the webpage more visually pleasing.

In the end our overall result was very pleasing and was a very good learning experience. It was our first time having to program for something with a small memory and made us realize and learn different things which we never really thought about. Though we did not meet our smaller goals we met all of our bigger goals, except for making it an all-in-one device, which was limited by the device and not a result of a lack in trying.

Resources

Yates, Darren. "Arduino Project 5: Digital Audio Player." *Arduino Project 5: Digital Audio Player*. APC Mag, 20 May 2013. Web. 05 Dec. 2013.

http://apcmag.com/arduino-project-5-digital-audio-player.htm

Yates, Darron. "Arduino Project 6: Web-controlled Music Player." *Arduino Project 6: Web-controlled Music Player*. APC Mag, 8 July 2013. Web. 05 Dec. 2013.

http://apcmag.com/arduino-project-6-web-controlled-music-player.htm

- Button Player
  - Used information to understand how audio output as well as the SD card reader works.
  - Used Circuit diagram to help create our circuit
  - Used their WAV file converter apcmag.com/arduino.htm
  - Used their code

- Server Music Player
  - Used Circuit diagram to help create our circuit
  - Used information to understand how establish a connection between the Arduino and sever and the use of the "simpleSDAudio" library
  - Used their AHS Stereo file converter dl.techlife.net/apps/netplay.zip.
  - Learned about the "simpleSDAudio" library and used one of the libraries examples as a template SimpleSDAudio'

*/

Team Member	Hours/week	% of Contribution
Jon Kosir	**2.5**	**25**
Sebastian Monnin	**2.5**	**25**
Thomas Lawson	**2.5**	**25**
Kalene Kelly	**2.5**	**25**

4 weeks

Appendix

CODE FOR WEBSERVER PLAYER

```
/*
  SimpleSDAudio webserver example

  Select files to play via webbrowser.

  Uses standard Ethernet-Shield with SD card or Ethernet-Arduino.
  Audio output on default pins: 9 for standard, 44 for mega Arduinos.

  Don't use stereo on non-mega Arduinos as a pin collision between
  Eth-CS and second audio output will occur (both pin 10)!

  created  01 Jul 2012 by Lutz Lisseck,
  with help from Ladyada SD webserver example.

  Visit our hackerspace website for more information:
  http://www.hackerspace-ffm.de/wiki/index.php?title=SimpleSDAudio
 */
```

```
/*

This Example was modifieded and changed to fit our needs in our Final Project for ECE 303

Jonathan K

Sebatian M

Thomas L

Kelly K

*/

#include <SPI.h>

#include <Ethernet.h>

#include <SimpleSDAudio.h>


// Enter a MAC address and IP address for your controller below.

// The IP address will be dependent on your local network:

//Mac address really not needed

byte mac[] = {

  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEB };

  //ip address we assign

IPAddress ip(192,168,1,130);


// Initialize the Ethernet server library

// with the IP address and port you want to use

// (port 80 is default for HTTP):

EthernetServer server(80);




// We have also a favicon...

const uint8_t favData[] PROGMEM  = {

		0x00, 0x00, 0x01, 0x00, 0x01, 0x00, 0x10, 0x10, 0x10, 0x00, 0x00, 0x00,

		0x00, 0x00, 0x28, 0x01, 0x00, 0x00, 0x16, 0x00, 0x00, 0x00, 0x28, 0x00,

		0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x01, 0x00,

		0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,

		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
		0x00, 0x00, 0xFF, 0xFE, 0xFF, 0x00, 0xFD, 0xC6, 0x8D, 0x00, 0xB2, 0x4F,
		0x00, 0x00, 0xB1, 0x5C, 0x16, 0x00, 0xB0, 0x51, 0x08, 0x00, 0xFF, 0xF9,
		0xFF, 0x00, 0xB4, 0x51, 0x0D, 0x00, 0xB6, 0x4A, 0x00, 0x00, 0xF8, 0xFD,
		0xFF, 0x00, 0xFB, 0xFF, 0xFC, 0x00, 0xE0, 0x68, 0x04, 0x00, 0xAC, 0x57,
		0x0D, 0x00, 0xA6, 0x5B, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC,
		0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC,
		0xCC, 0xCC, 0xCC, 0xCA, 0x5C, 0xCC, 0xCC, 0xCC, 0xCC, 0x5C, 0xCC, 0xCC,
		0x51, 0xCC, 0xCC, 0xC2, 0xC5, 0x5C, 0xBC, 0x1C, 0xC5, 0xCC, 0xCC, 0xCC,
		0x55, 0x5C, 0xCC, 0x55, 0xCA, 0x5C, 0xC1, 0x55, 0x55, 0x5C, 0xC7, 0x45,
		0xCA, 0x5C, 0xC1, 0x55, 0x55, 0x5C, 0x55, 0xC5, 0xAA, 0x5C, 0xC1, 0x55,
		0x55, 0x5C, 0x55, 0xC5, 0xAA, 0x5C, 0xC1, 0x55, 0x58, 0x5C, 0x3C, 0xC5,
		0xCA, 0x5C, 0xCC, 0xCC, 0x59, 0x0C, 0x6C, 0x55, 0xCA, 0x5C, 0xCC, 0xCC,
		0xC5, 0x5C, 0xCC, 0x1C, 0xC5, 0xCC, 0xCC, 0xCC, 0xCC, 0x5C, 0xCC, 0xCC,
		0x51, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCA, 0x5C, 0xCC, 0xCC, 0xCC,
		0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC,
		0xCC, 0xCC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
		0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

// helper function to determine free ram at runtime
int freeRam () {
  extern int __heap_start, *__brkval;
  int v;
  return (int) &v - (__brkval == 0 ? (int) &__heap_start : (int) __brkval);
}
//setup
void setup() {
  pinMode(8,INPUT_PULLUP);
  digitalWrite(8,HIGH);
```

```
 // Open serial communications and wait for port to open:
  Serial.begin(9600);
   while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }
 SdPlay.init(SSDA_MODE_STEREO | SSDA_MODE_HALFRATE);

  // start the Ethernet connection and the server:
  Ethernet.begin(mac,ip);
  server.begin();
  //print serial for debugging
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}

EthernetClient *pclient;

//asigns the Songs a link to be clicked on
void dir_callback(char *buf) {
  if(pclient) {
    pclient->print("<li><a href=\"");
    pclient->print(buf);
    pclient->println("\">");
    pclient->print(buf);
    pclient->println("</a></li>");
  }
}

// How big our line buffer should be. 100 is plenty!
#define BUFSIZ 100
boolean playflag = false;

void loop() {
  char clientline[BUFSIZ];
```

```
int index = 0;

if(playflag) { SdPlay.play(); playflag = false; }

SdPlay.worker();

// listen for incoming clientsy

EthernetClient client = server.available();

if (client) {

  Serial.println("new client");

  // an http request ends with a blank line

  boolean currentLineIsBlank = true;

  pclient = &client;

  index = 0;

  while (client.connected()) {

    if (client.available()) {

      char c = client.read();


      // If it isn't a new line, add the character to the buffer

      if (c != '\n' && c != '\r') {

        clientline[index] = c;

        index++;

        // are we too big for the buffer? start tossing out data

        if (index >= BUFSIZ)

          index = BUFSIZ -1;


        // continue to read more data!

        continue;

      }


      // got a \n or \r new line, which means the string is done

      clientline[index] = 0;


      // Print it out to the serial

      Serial.println(clientline);


      // Look for substring such as a request to get the root file
      if (strstr(clientline, "GET / ") != 0) {

        // send a standard http response header

        client.println(F("HTTP/1.1 200 OK"));

        client.println(F("Content-Type: text/html"));

        client.println();



        // Prints text to webpage in HTML Format

        client.println(F("<h1 style='text-align: center;'>Simple Server Music Player</h1><p style='text-align: center;'>Jonathan Kosir</p><p
style='text-align: center;'>Kelly Kalene</p><p style='text-align: center;'>Sebastian Monnin</p><p style='text-align: center;'>Thomas
Lawson</p><h1 style='text-align: center;'</h1>"));

        client.println(F("<h4 style='text-align: center;'>Pick file to play:<p>"));

        //The call to retrieve the song list from the SD card

        SdPlay.dir(&dir_callback);

        client.println(F("</p>"));

        client.println(F("<img alt='' src='http://seasexperience.eas.muohio.edu/wp-content/uploads/2013/05/Screen-Shot-2013-01-22-at-11.58.08-
AM.png' style='text-align: center; width: 180px; height: 129px;'/></h4>"));



      } else if (strstr(clientline, "GET /") != 0) {

        // this time no space after the /, so a sub-file!

        char *filename;



        filename = clientline + 5; // look after the "GET /" (5 chars)

        // a little trick, look for the " HTTP/1.1" string and

        // turn the first character of the substring into a 0 to clear it out.

        (strstr(clientline, " HTTP"))[0] = 0;



        // print the file we want

        Serial.println(filename);



        if(strstr(filename,"favicon.ico")) {

          // Tx favicon from data block

          client.println(F("HTTP/1.1 200 OK"));

          client.println(F("Content-Type: image/x-icon"));

          client.println();

          for(int i=0;i<sizeof(favData);i++) {
            client.write(pgm_read_byte_near(favData+i));

          }

        } else {


          if (! SdPlay.setFile(filename)) {

            client.println(F("HTTP/1.1 404 Not Found"));

            client.println(F("Content-Type: text/html"));

            client.println();

            client.println(F("<h2>File Not Found!</h2>"));

            break;

          }


          Serial.println(F("Opened!"));


          client.println(F("HTTP/1.1 200 OK"));

          client.println(F("Content-Type: text/html"));

          client.println();

          //Prints text to the webpage via HTML

          client.print(F("<h2>Now playing...  "));

          //get the file name of the song to print

          client.print(filename);

          client.print("</h2>");

          client.println(F("<h1 style='text-align: center;'>Simple Server Music Player</h1><p style='text-align: center;'>Jonathan Kosir</p><p
style='text-align: center;'>Kelly Kalene</p><p style='text-align: center;'>Sebastian Monnin</p><p style='text-align: center;'>Thomas
Lawson</p><h1 style='text-align: center;'</h1>"));

          client.println(F("<h4 style='text-align: center;'>Pick file to play:<p>"));

          //Print the list of songs on the sd card

          SdPlay.dir(&dir_callback);

          client.println(F("</p>"));

          client.println(F("<img alt='' src='http://seasexperience.eas.muohio.edu/wp-content/uploads/2013/05/Screen-Shot-2013-01-22-at-11.58.08-
AM.png' style='text-align: center; width: 180px; height: 129px;'/></h4>"));

          playflag = true;

        }


      } else {
        // If the file doesnt exist send a 404 error to browser
        client.println(F("HTTP/1.1 404 Not Found"));
        client.println(F("Content-Type: text/html"));
        client.println();
        client.println(F("<h2>File Not Found!</h2>"));
      }
      break;
    }

  }
  // give the web browser time to receive the data
  delay(1);
  // close the connection:
  client.stop();
  pclient = NULL;
  Serial.println("client disonnected");
}
}
```

CODE FOR BUTTON PLAYER

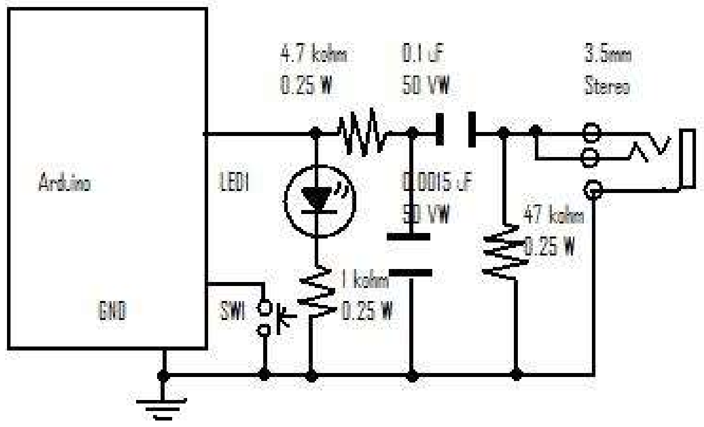
```
/*This was taken from apMag and was used for our finalproject for ECE 303
comments added by us to get an understanding of the code
made few very minor changes to get the code to work for us
Jonathn k
Kelly K
Sebastian M
Thomas L
*/

// --------------------------------------------------------------------------------
// DO NOT USE CLASS-10 CARDS on this project - they're too fast to operate using SPI
// --------------------------------------------------------------------------------
```

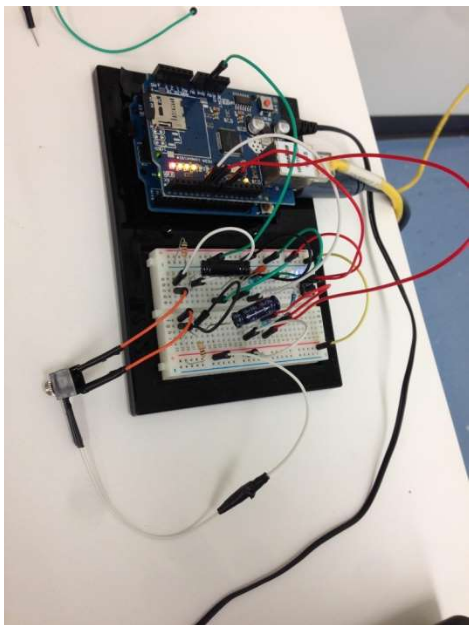
```
#include <SD.h>

#include <TMRpcm.h>

TMRpcm tmrpcm;


File root;

File entry;


// ---------------------------------------------------------------------------------

// set chipSelect to '10' if using the $2 SD card module or '4' if using the

// Ethernet shield's microSD card instead.

const int chipSelect = 4;

// ---------------------------------------------------------------------------------


const int oldCard = SPI_HALF_SPEED;

const int newCard = SPI_QUARTER_SPEED;


// ---------------------------------------------------------------------------------

// set cardType to 'oldCard' if using an old SD card (more than a few years old) or

// to 'newCard' if using a newly-purchase Class-4 card.

int cardType = oldCard;

// ---------------------------------------------------------------------------------


int wasPlaying = 0;

int inSwitch = 7;

int finished = 0;

int start = 0;

int pauseOn = 0;

unsigned long timeDiff = 0;

unsigned long timePress = 0;


//Setup
```

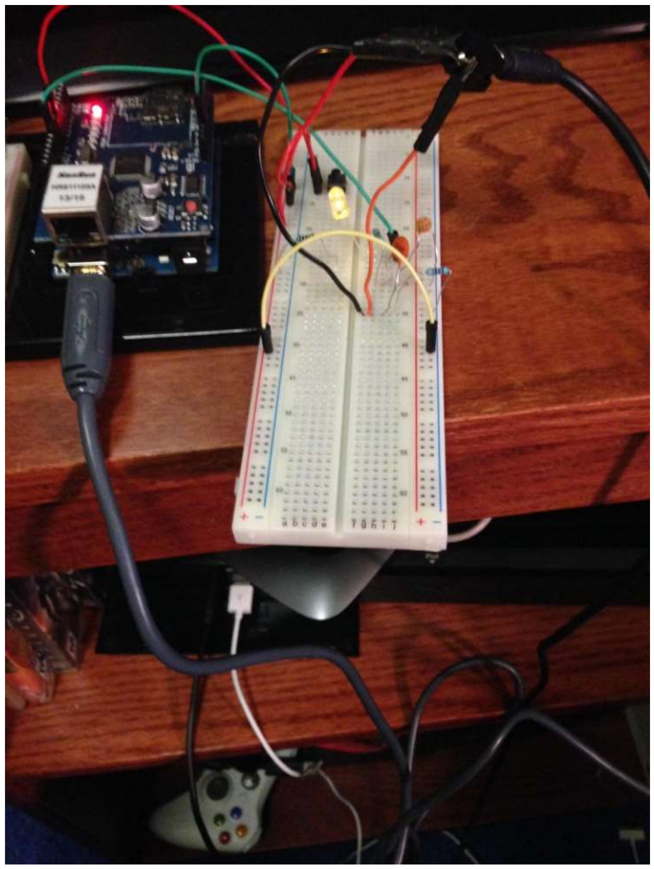
```
void setup() {
  Serial.begin(9600);
  Serial.print("\nInitializing SD card...");
  pinMode(chipSelect, OUTPUT);
  if (!SD.begin(chipSelect,cardType)) {
    Serial.println("failed!");
    return;
  }
  Serial.println("done.");

  tmrpcm.speakerPin = 9;

  pinMode(inSwitch,INPUT_PULLUP);
  digitalWrite(inSwitch,HIGH);

  root = SD.open("/");
}

void loop(void) {
  //Checks for if the song is over if it is plays next
  if(!tmrpcm.isPlaying() && wasPlaying == 1) {
    tmrpcm.stopPlayback();
    playNext();
  }

  //Checks the button every 100ms (Gets rid of rapid button reads -works like a wait statement)
  if (millis() - timeDiff > 50) { // check switch every 100ms
     timeDiff = millis(); // get current millisecond count

      //If Switch was pressed
      if(digitalRead(inSwitch) == LOW) {
```

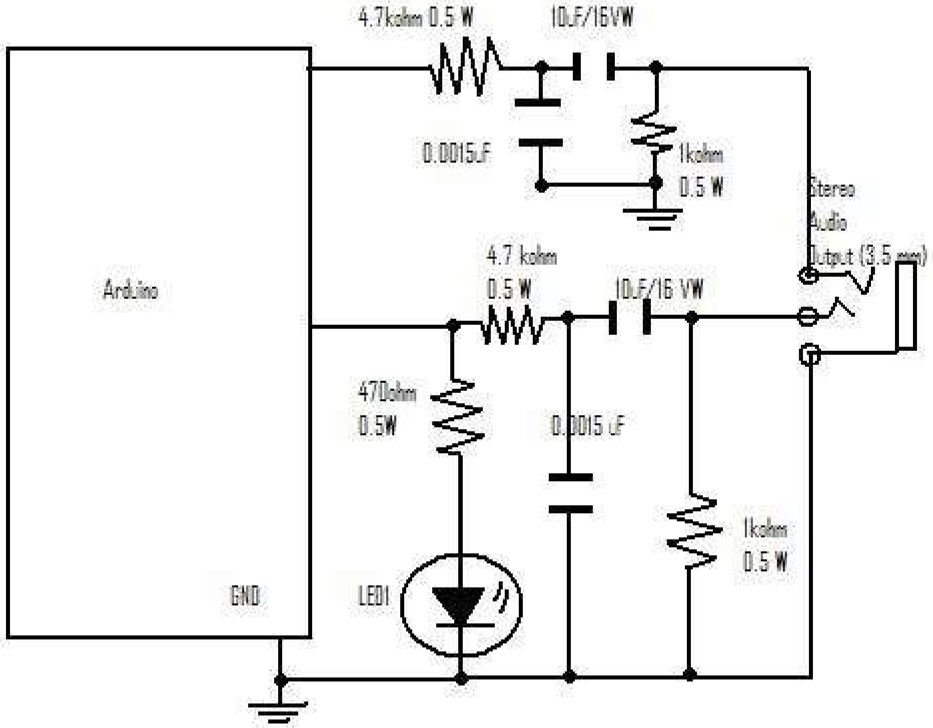
```
//if the first song on the sd has not been started begin play of the first song
if(start==0) {
  start=1;
  playNext();
  delay(200);

} else {

  //Counts how long button is pressed in for
  timePress = millis();
  while(digitalRead(inSwitch)==LOW) {
    delay(50);
  }

  //iff it was a quick press and music is playing pause music
  //if it was a quick press and music is paused play music
  if (millis() - timePress < 1000 && start == 1) {
    tmrpcm.pause();
    if (pauseOn == 0) {
      pauseOn = 1;
    } else {
      pauseOn = 0;
    }
  }

  //if it was a long press play next song
  //unless it was the last song then it restarts from the beginning
  if (millis() - timePress > 1000 && start == 1) {
     if (pauseOn == 1) {pauseOn = 0; tmrpcm.pause();}
     tmrpcm.stopPlayback();
     timePress = millis();
     if (finished == 0) {
```

```
              playNext();
            } else {
              finished = 0;
              Serial.println("Restarting.");
              root.rewindDirectory();
              playNext();
            }
          }
        }
      }
  }
}

//Method to play our next song
void playNext() {

  entry = root.openNextFile();
  if (entry) {

    entry.close();
    tmrpcm.play(entry.name());
    wasPlaying = 1;
  } else {
    if (wasPlaying == 1) {
      Serial.println("Completed playback.");
      wasPlaying = 0;
      finished = 1;
      start = 0;
      root.rewindDirectory();
    }
  }
}
```


4.7 kohm
0.25 W
0.1 uF
50 VW
3.5mm
Stereo
Arduino
LED1
0.0015 uF
50 VW
47 kohm
0.25 W
1 kohm
0.25 W
SW1
GND






4.7kohm 0.5 W
10uF/16VW
0.0015uF
1kohm
0.5 W
Stereo
Audio
Output (3.5 mm)
Arduino
4.7 kohm
0.5 W
10uF/16 VW
470ohm
0.5W
0.0015 uF
1kohm
0.5 W
GND
LED1