```
// FILE NAME: Music Box Frequency Detection KDK Design

// AUTHOR: Kyle Trevis

// VERSION: 1.0.5

// PURPOSE: Detecting Fundemental Frequency of .wav file audio

// DATE: 4-15-2020

//////////////////////////////////////////////////////////////////REFERENCES//////////
////////////////////////////////////////////////

// SD card managemnt code based on "write()" function from arduino references. Orignal
by Massimo Banzi, modified by Scott Fitzgerald //

// Autocorrelation Algorithm and Frequency detection based on "Reliable Frequency
Measurement Using Autocorrelation" by akellyiri     //

// Knowledge of SD and file managment and pins derived from Arduino Reference website
//

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


// INLCUDE H FILES


#include <SD.h>

#include <SPI.h>


  int b0 = A2;

  int b1 = A4;

  int b2 = A5;


  bool v0 = 0;

  bool v1 = 0;

  bool v2 = 0;



void setup() {

  Serial.begin(9600);


  pinMode(b0, INPUT);

  pinMode(b1, INPUT);

  pinMode(b2, INPUT);
```

```
}

void loop() {

  v0 = digitalRead(b0);
  delay(500);
  v1 = digitalRead(b1);
  delay(500);
  v2 = digitalRead(b2);
  delay(500);

  if(v0==HIGH){
    File aud_file = file_access("TRACK1.WAV");
    freq_det_play(aud_file);
  }
  delay(1000);

   if(v1==HIGH){
    File aud_file = file_access("TRACK2.WAV");
    freq_det_play(aud_file);
  }
  delay(1000);

   if(v2==HIGH){
    File aud_file = file_access("TRACK3.WAV");
    freq_det_play(aud_file);
  }
  delay(1000);

}

File file_access(char filename) {
  if (!SD.begin(4)) {
      return;
```

```
  }

  if(!SD.exists("rec_audio.wav")){                      //Checking if recorded audio
exists
    return;
  }
  File rAudio = SD.open(filename);              //Reading file from SD card

  return rAudio;
}

void freq_det_play(File rAudio){

  if(!rAudio.available()){                                    //Checking if audio file
is empty
    return;
  }

  int len = sizeof(rAudio);                                   // get length of file in
bytes

  char audata[len];                                       //create a char array to store
the sampled values in, allowing us to operate on them

  rAudio.read(audata, len);

  const float sample_freq = 16000;

  int i,j,k,z;

  int tone_values[len/320];
  memset(tone_values,0,(len/320));

  long sum, old_sum;

  int threshold = 0;
```

```
 float fund_freq = 0;

 byte pd_state = 0;



 sum = 0;
 pd_state = 0;
 int period = 0;

 //Actual Autocorellation Algorithm
for(k=0; k<(len/320); k++){
 for(i = (k*320); i <((k+1)*320); i++){
   old_sum = sum;
   sum = 0;

   for(j = (k*320); j < (((k+1)*320)-i); j++){
     sum += ((audata[j]-128)*(audata[j+i]-128)/256);
   }

   //Peak Detection State Machine

   if(pd_state ==2 && (sum - old_sum) <= 0 )
   {
     period = (i-(k*320));
     pd_state = 3;
   }
   if (pd_state == 1 && (sum > threshold) && (sum-old_sum) > 0){
     pd_state = 2;
   }
   if(i == (k*320)){
     threshold = (sum * 0.5);
     pd_state = 1;
   }
```

```
  }
  if(pd_state == 3){
    fund_freq = (sample_freq/period);
    tone_values[k] = fund_freq;
  }
  if(pd_state != 3){
    tone_values[k] = 0;
  }
 }
 for(z=0;z<(len/320);z++){
    tone(3,tone_values[z],20);
    delay(20);
 }
}
```