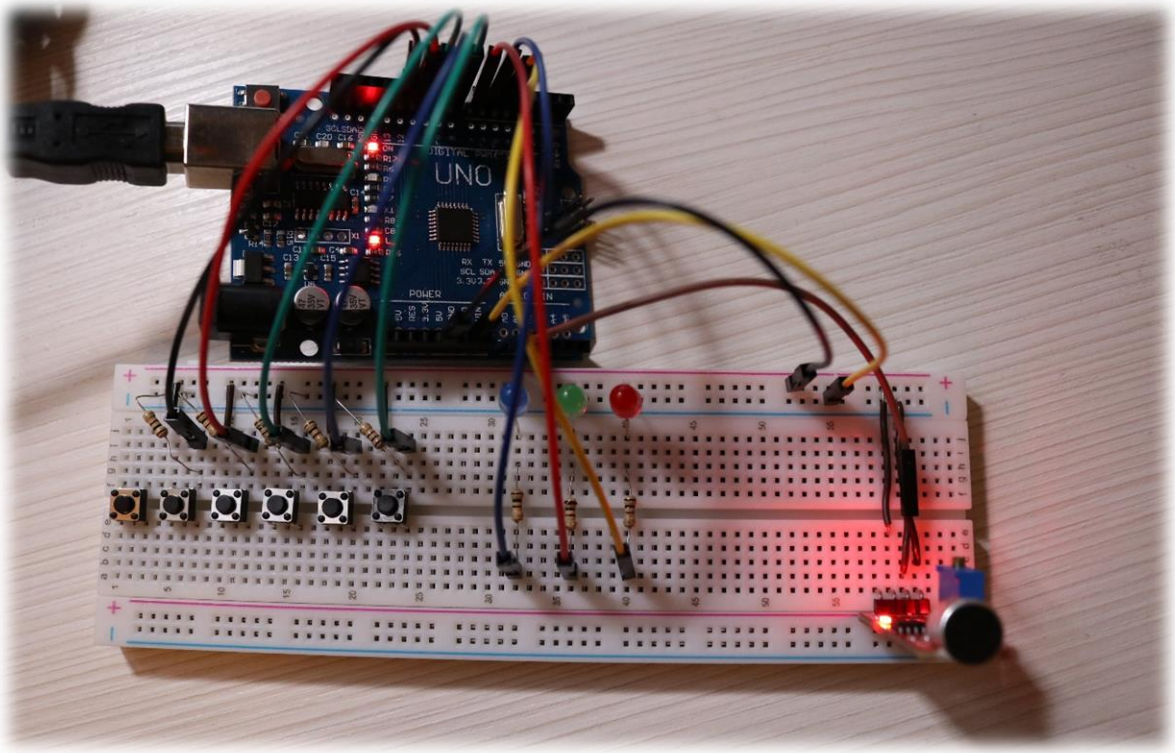
Faculty of Engineering in Foreign Languages

Guitar Tuner

- Microprocessor Architecture Project -



Student: Velicu Ana

Group: 1221A

Introduction

Music is an art that everyone appreciates. For each individual, music represents a unique experience, and some people pursue this kind of art by creating it.

There are a lot of instruments characterized differently. There are keyed instruments such as piano, pipe organ, percussion instrument such as drums, cymbals, maracas, brass instruments, also string instruments, such as the guitar.

The guitar has 6 strings, denoted from top to bottom Em, B, G, D, A, E on open string. Each note can be performed in a different version, being more loud or low, depending on the pitch.

Pitch is indicated by the frequency of that sound. Since the frequency of these notes are known, for us to determine if the guitar is tuned or not, we only need to compare the frequency of the note of a particular string to the actual frequency of the note that the string represents.

The frequencies of the musical notes on the guitar are:

E4 = 329.63 Hz

B3 = 246.94 Hz

G3 = 16.35 Hz

D3 = 146.83 Hz

A2 = 110.00 Hz

E2 = 82.41 Hz

In order to tune a guitar, there will be created a tuner using an Arduino Uno microprocessor.

Project flow

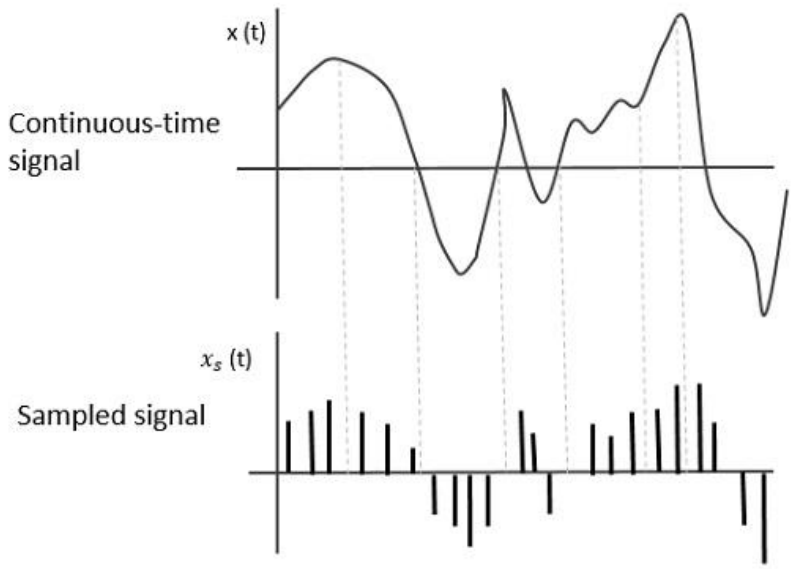
The tuner will convert the sound signal received from the guitar to a frequency, then will compare it to the exact frequency value of the chosen string. When the value correlates, a specific led will be on.

The frequency detection/conversion involves 3 main stages:

- Amplifying
- Offsetting
- Analog to digital conversion

The sound signal being produced will be too weak for the Arduino's ADC to recognize so we need to amplify the signal. After amplification, to keep the signal within the range recognizable by the Arduino's ADC to prevent clipping of the signal, we offset the voltage of the signal. After offsetting, the signal is then passed to the Arduino ADC where it is sampled, and the frequency of that sound is obtained.

The sampling is done by using discrerte – time Forurier Transform. Shortly, it breaks the continuous signal into a sequence of values, which can become a form of Fourier analysis, allowing us to operate with it in the same way, having the same behaviour.



Required components

For this project we need:

- Arduino UNO microcontroller
- Breadboard
- Jump wires
- 6 buttons
- 3 different colour LED
- 3 resistors 100Ω
- 6 resistors 1KΩ
- KY-037 sound detection & sensor microphone

Software resources

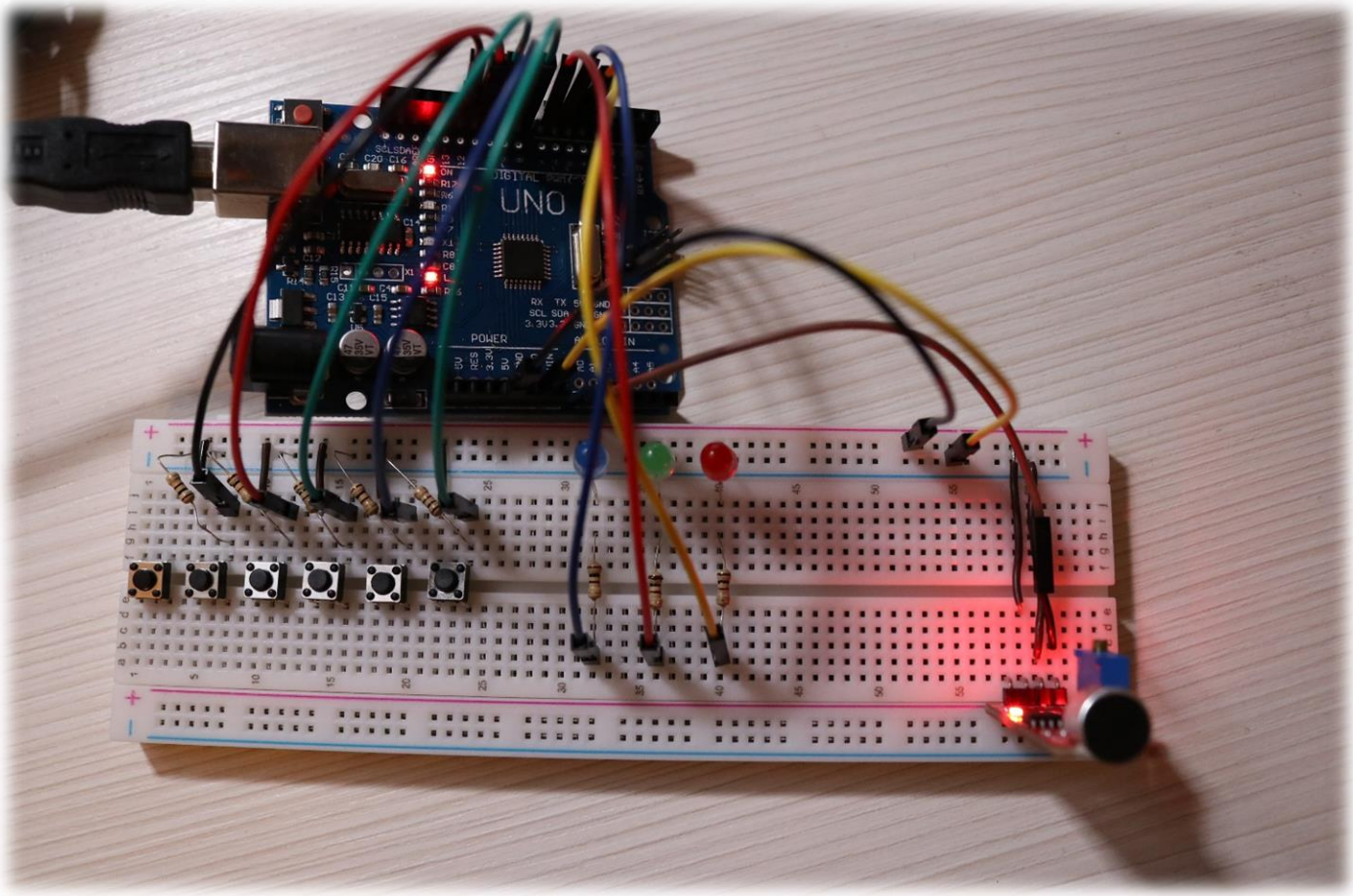
For the project, it is used the Arduino IDE cross-platform application.

The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures.

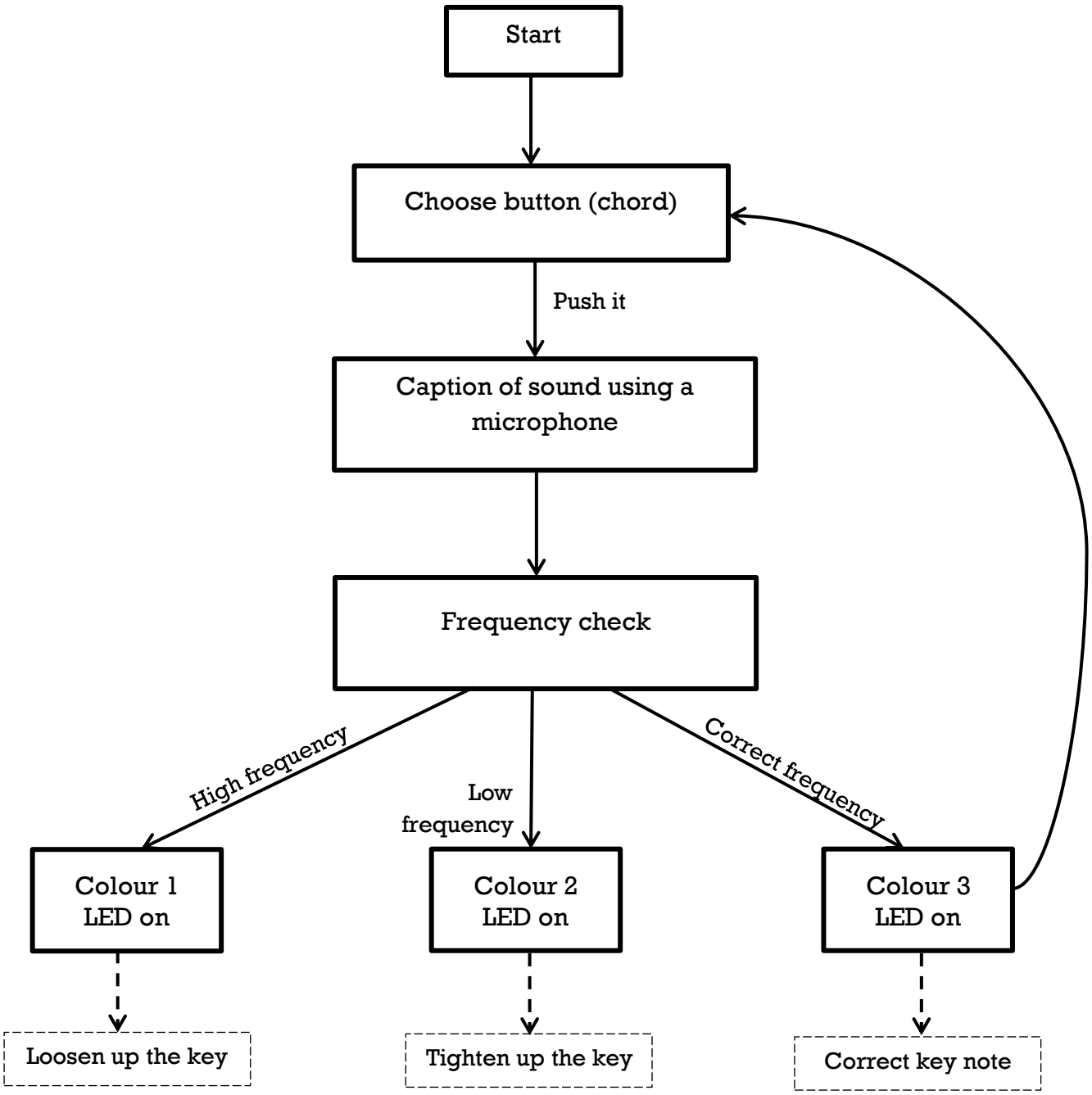
User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution.

The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.

Hardware Implementation



Organizational Chart



Application code

```
int buttonarray[] = {12, 11, 10, 9, 8}; // [E2, A2, D3, G3, B3, E4]

float freqarray[] = {82.41, 110.00, 146.83, 196.00, 246.94, 329.63};//sll in Hz


int lowerLed = 7;

int higherLed = 6;

int justRight = 5;


#define LENGTH 512


byte rawData[LENGTH];

int count = 0;


const float sample_freq = 8919;

int len = sizeof(rawData);

int i,k;

long sum, sum_old;

int thresh = 0;

float freq_per = 0;

byte pd_state = 0;

int value;


void setup(){

 for (int i=0; i<5; i++)

 {

  pinMode(buttonarray[i], INPUT);

 }

 pinMode(lowerLed, OUTPUT);

 pinMode(higherLed, OUTPUT);

 pinMode(justRight, OUTPUT);


 Serial.begin(9600);

}

void loop(){


  if (count < LENGTH)

 {

  count++;

  rawData[count] = analogRead(A0)>>1;

 }

 else {

  sum = 0;

  pd_state = 0;

  int period = 0;

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

  {

   // Autocorrelation

   sum_old = sum;

   sum = 0;

   for(k=0; k < len-i; k++) sum += (rawData[k]-128)*(rawData[k+i]-128)/256;
```

```
    // Serial.println(sum);

    // Peak Detect State Machine
    if (pd_state == 2 && (sum-sum_old) <=0)
    {
      period = i;
      pd_state = 3;
    }
    if (pd_state == 1 && (sum > thresh) &&
(sum-sum_old) > 0) pd_state = 2;
    if (!i) {
      thresh = sum * 0.5;
      pd_state = 1;
    }
   }
   if (thresh >100) {
    freq_per = sample_freq/period;
    Serial.println(freq_per);

      for ( int s=0; s<5 ; s++)
    {
      value = digitalRead(buttonarray[s]);
      Serial.println(value);

      digitalWrite(lowerLed, LOW);
      digitalWrite(justRight, LOW);
      digitalWrite(higherLed, LOW);

if (value == HIGH)
{
 //Serial.println("I'm HEREEEEEE");
 Serial.println("");

 if (freq_per - freqarray[s] < 0)
 {
  Serial.println("strange");
  Serial.println(freq_per);
  Serial.println(freqarray[s]);
  digitalWrite(lowerLed, HIGH);
  digitalWrite(justRight, LOW);
  digitalWrite(higherLed, LOW);
 }
 else if(freq_per - freqarray[s] > 10)
 {
  Serial.println("lasaa");
  Serial.println(freq_per);
  Serial.println(freqarray[s]);
  digitalWrite(higherLed, HIGH);
  digitalWrite(lowerLed, LOW);
  digitalWrite(justRight, LOW);
 }
 else
 {
  Serial.println("bines");
  Serial.println(freq_per);
```

```
      Serial.println(freqarray[s]);
      digitalWrite(justRight, HIGH);
      digitalWrite(higherLed, LOW);
      digitalWrite(lowerLed, LOW);
    }
   }
  }
}
   count = 0;
}
}
```

Credits

1. https://circuitdigest.com/microcontroller-projects/arduino-uno-guitar-tuner
2. https://create.arduino.cc/projecthub/electropeak/how-to-use-ky-037-sound-detection-sensor-with-arduino-a757a7
3. https://www.youtube.com/watch?v=ErtPBvEnQZc
4. https://youtu.be/k9zQjEaKtfk
5. https://youtu.be/CbovaHqvdsM