

APÊNDICE D – Segundo algoritmo do Protótipo com adaptação do algoritmo de Kelly

```

/**
 * Centro Universitário de Anápolis - UniEVANGÉLICA
 * Bacharelado em Engenharia de Computação
 * Trabalho de Conclusão de Curso II 2019/2
 *
 * Acadêmicos:
 * Daniel Vaz de Oliveira
 * Jônatas Gabriel da Silva Santos
 *
 * Orientador:
 * William
 *
 * DISPOSITIVO ELETRÔNICO AUTOMATIZADO PARA AFINAR VIOLÃO
 *
 * Referência de código (Algoritmo Identificador de Frequências):
 * https://github.com/akellyirl/Arduino-Guitar-Tuner
 */

#include <Stepper.h>

int steps=100; // Número de passos para o motor
Stepper motor(steps,8,9); // Pinos de controle do motor
int pinL293D = 10; // Pino que liga o circuito que controla o motor (L293D)
int string = 6; // Número da corda do violão
int stringButton = 11; // Botão que troca de corda
float minFrequency; // Frequência mínima da corda selecionada
float maxFrequency; // Frequência máxima da corda selecionada
float stringFrequency; // Frequência ideal da corda selecionada
float stringFrequencyMax; // Frequência de tolerância máxima da corda
selecionada
float stringFrequencyMin; // Frequência de tolerância mínima da corda
selecionada
float frequency;

#define LENGTH 512
int numSamples = 0;
long t0, t;

byte rawData[LENGTH];
int count;

// Sample Frequency in kHz
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;

```

```

void setup(){

    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(7, OUTPUT);

    pinMode(stringButton, INPUT_PULLUP); // Configura pino como entrada e
    habilita pull up interno;
    setString(); // Configura a primeira corda

    pinMode(stringButton, INPUT_PULLUP); // Configura o pino do botão como
    entrada e habilita pull up interno;
    setString(); // Configura a primeira corda

    motor.setSpeed(80); // Configura a velocidade da rotação do motor (RPM)
    pinMode(pinL293D,OUTPUT); // Configura o pino que liga o circuito que
    controla o motor (L293D)
    digitalWrite(pinL293D,LOW); // Mantém desligado o circuito que controla o
    motor (L293D)

    Serial.begin(9600); // Configura a comunicação serial
    Serial.println("Pronto para afinar! @");

    analogReference(EXTERNAL);
    analogRead(A0);

    count = 0;
}

void loop(){

    // Algoritmo de Detecção de Frequência
    if (count < LENGTH) {
        if(numSamples == 0){
            t0 = micros();
        }

        count++;
        rawData[count] = analogRead(A0) >> 2;
        numSamples++;
    }
    else {

        // Calcula a Taxa de Amostragem
        t = micros() - t0;

        Serial.println();
        Serial.print("Tempo médio por amostra: ");
    }
}

```

```

Serial.print((float) t / numSamples);
Serial.println(" µs");
Serial.print("Amostras por segundo: ");
sample_freq = (float) numSamples * 1000000 / t;
Serial.println(sample_freq);

numSamples = 0;

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;
    }
}
// for(i=0; i < len; i++) Serial.println(rawData[i]);
// Frequency identified in Hz
if (thresh >100) {
    freq_per = sample_freq/period;
    Serial.println();
    Serial.println();
    Serial.print("\t\t\t\t\t");
    Serial.print(freq_per);
    Serial.print(" Hz");

    frequency = freq_per;

    if(frequency > minFrequency && frequency < maxFrequency){
        Serial.print(" - @ ");

        if(frequency < stringFrequency - 2 || frequency > stringFrequency
+ 2){

```

```

        steps = 400;
    } else if(frequency < stringFrequency - 3 || frequency >
stringFrequency + 3){
        steps = 1000;
    } else if(frequency < stringFrequency - 4 || frequency >
stringFrequency + 4){
        steps = 5000;
    }else{
        steps = 95;
    }

    if(frequency < stringFrequencyMin){
        Serial.print(" | ");
        Serial.print(stringFrequency);
        Serial.print(" (");
        Serial.print(stringFrequencyMin);
        Serial.print(" ~ ");
        Serial.print(stringFrequencyMax);
        Serial.print(") Apertando +");
        Serial.println(steps);
        stepRight();
    } else if(frequency > stringFrequencyMax) {
        Serial.print(" | ");
        Serial.print(stringFrequency);
        Serial.print(" (");
        Serial.print(stringFrequencyMin);
        Serial.print(" ~ ");
        Serial.print(stringFrequencyMax);
        Serial.print(") Afrouxando +");
        Serial.println(steps);
        stepLeft();
    } else {
        Serial.print(" | ");
        Serial.print(stringFrequency);
        Serial.print(" (");
        Serial.print(stringFrequencyMin);
        Serial.print(" ~ ");
        Serial.print(stringFrequencyMax);
        Serial.println("                AFINADO!!!");
        delay(2000);
    }

    } else {
        Serial.println(" - FORA DO INTERVALO DA CORDA");
    }
}
count = 0;
}

// Troca a corda se o botão for pressionado
if(digitalRead(stringButton) == HIGH){
    setString();
    delay(300);
}

```

```

}

// Para o motor e desliga o circuito de controle
void stopStepper(){
  digitalWrite(pinL293D,LOW);
  //PORTB &= B11111011;
  motor.step(0);
}

// Roda o motor no sentido anti-horário
void stepLeft(){
  digitalWrite(pinL293D,HIGH);
  //PORTB |= B00001000;
  motor.step(-steps);
  stopStepper();
}

// Roda o motor no sentido horário
void stepRight(){
  digitalWrite(pinL293D,HIGH);
  //PORTB |= B00001000;
  motor.step(steps);
  stopStepper();
}

// Troca para a próxima corda
void setString(){
  switch(string){
    case 1:
      string = 2;
      PORTD = B00001000;
      // TODO Configurar parâmetros da corda A 110 Hz
      minFrequency = 95.25;
      maxFrequency = 127.14;
      stringFrequency = 110;
      stringFrequencyMax = 110.55;
      stringFrequencyMin = 109.45;
      sample_freq = 8360;
      break;
    case 2:
      string = 3;
      PORTD = B00010000;
      // TODO Configurar parâmetros da corda D 146.832 Hz
      minFrequency = 127.14;
      maxFrequency = 169.171;
      stringFrequency = 146.83;
      stringFrequencyMax = 147.38;
      stringFrequencyMin = 146.28;
      sample_freq = 8576;
      break;
    case 3:
      string = 4;

```

```
    PORTD = B00100000;
    // TODO Configurar parâmetros da corda G 195.998 Hz
    minFrequency = 169.171;
    maxFrequency = 220;
    stringFrequency = 196;
    stringFrequencyMax = 196.55;
    stringFrequencyMin = 195.45;
    sample_freq = 8305;
    break;
case 4:
    string = 5;
    PORTD = B01000000;
    // TODO Configurar parâmetros da corda B 246.942 Hz
    minFrequency = 220;
    maxFrequency = 285.425;
    stringFrequency = 246.94;
    stringFrequencyMax = 247.49;
    stringFrequencyMin = 246.39;
    sample_freq = 8836;
    break;
case 5:
    string = 6;
    PORTD = B10000000;
    // TODO Configurar parâmetros da corda E 329.628 Hz
    minFrequency = 285.425;
    maxFrequency = 378.995;
    stringFrequency = 329.63;
    stringFrequencyMax = 330.18;
    stringFrequencyMin = 329.08;
    sample_freq = 8470;
    break;
case 6:
    string = 1;
    PORTD = B00000100;
    // TODO Configurar parâmetros da corda E 82.407 Hz
    minFrequency = 71.36;
    maxFrequency = 95.25;
    stringFrequency = 82.41;
    stringFrequencyMax = 82.96;
    stringFrequencyMin = 81.86;
    sample_freq = 8323;
    break;
}
}
```