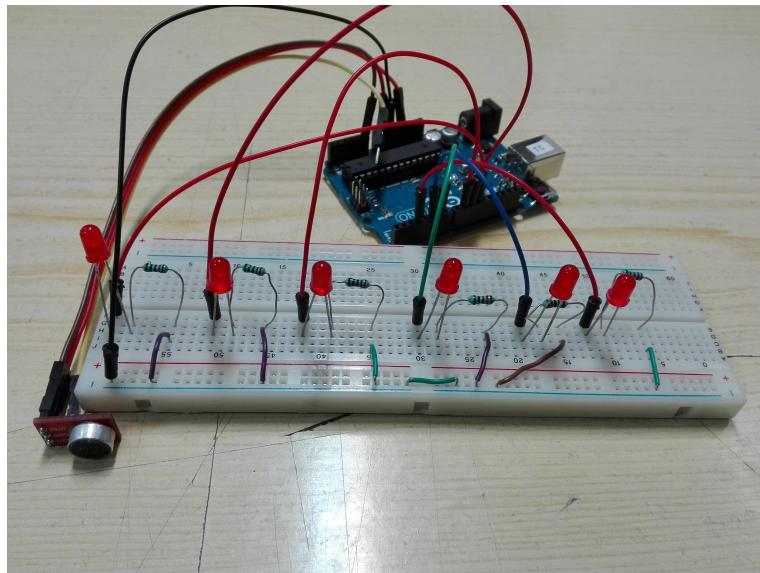


Paweł Budzyński, 208248
Daniel Kończal, 230041

Projekt - stroik gitarowy

Wrocław, 1 lutego 2017



Rysunek 1. Wykorzystane komponenty - obwód rzeczywisty

1. Wstęp

Pierwotnym pomysłem odnośnie tematu realizowanego w ramach projektu z kursu Metrologia i akwizycja danych było stworzenie taśmy świetlnej stworzonej z migających w rytm muzyki diod. Jednak po kilkudniowym namyśle i uświadomieniu sobie o tym jak bardzo zapaleni z nas gitarzyści nasz pomysł przerodził się w stworzenie stroika gitarowego. Idea działania tego czujnika była dość prosta - montaż mikrofonu, który odczyta częstotliwość, określenie marginesu błędu, przyrównanie do wzorca. Mimo tego nieskomplikowanego planu, stworzenie gitarowego stroika okazało się dla nas sporym wyzwaniem. Szczęśliwie udało nam się osiągnąć upragniony efekt, a w poniższym opisie przedstawimy w jaki sposób tego dokonaliśmy.

2. Wykorzystane komponenty i schemat obwodu

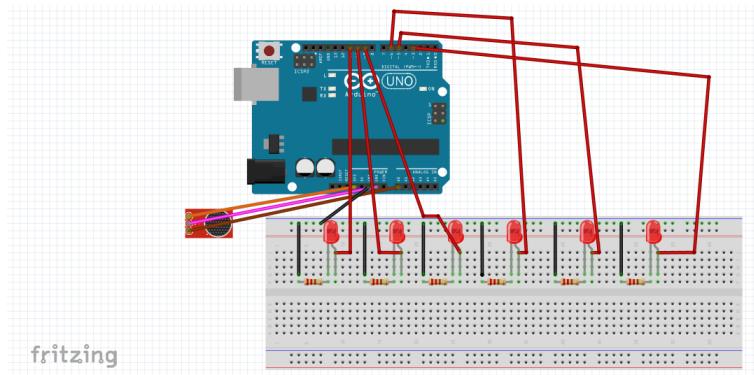
Do zbudowania naszego modelu wykorzystaliśmy:

- 1 mikrokontroler Arduino Uno,
- 1 płytę stykową,
- 1 mikrofon Sparkfun Bob-09964,
- 6 oporników o rezystancji 220Ω ,
- 6 diod LED koloru czerwonego,
- 16 przewodów.

Poniżej prezentujemy wizualizację obwodu wykonaną we Fritzingu, a także model rzeczywisty zawierający wszystkie wymienione komponenty.

3. Opis projektu

Jak wspomnialiśmy we wstępie idea projektu jest dość prosta. Opis krok po kroku umieścimy poniżej wraz z kodem programistycznym, jednak teraz



Rysunek 2. Wizualizacja obwodu

kilka słów o tym co tutaj właściwie się dzieje. Na początku odpowiemy na pytanie dlaczego wybraliśmy takie diody i takie oporniki. Otóż specyfikacja diody czerwonej jest nam dość dobrze znana, a więc nie mieliśmy problemu z dobraniem odpowiedniego opornika oraz umieszczenia jej w obwodzie. Wszystkie użyte przez nas diody mają jednakowe parametry. Każda dioda jest podłączona do innego pinu co jest równoważne z połączeniem równoległym diod, a więc każda dioda musi mieć dopasowany opornik. Wartość opornika wyliczyliśmy w sprawozdaniu nr 2 (Stosunek różnicy napięcia całkowitego i spadku napięcia na diodzie do natężenia). Ponadto korzystaliśmy już z podobnych schematów na zajeciach więc wydaje nam się uzasadnione użycie oporników o wartości 220Ω każdy.

Specyfikację oraz sposób podłączenie mikrofonu Sparkfun odnaleźliśmy w internecie. Jak się okazało nie jest wymagany do jego podłączenia żaden opornik. Mikrofon posiada 3 wyjścia - VCC podłączone do pinu $3.3V$, AUD podłączone do pinu analogowego A0 oraz GND - uziemienie. Uzyskane przez nas próbki sygnału miały postać sinusoidalnego sygnału ciągłego. Aby "wyciągnąć" z niego wartości częstotliwości próbkowanych sygnałów zastosowaliśmy dyskretną transformację Fouriera, którą poznaliśmy na wykładzie. Dzięki odpowiedniemu dobraniu szybkości przesyłu (115200 zamiast standardowego 9600), zastosowaniu dyskretnej transformacji Fouriera oraz dobraniu odpowiednich stałych udało nam się zamienić próbkowane dane na wartości częstotliwości. Dzięki temu mogliśmy przystąpić do głównego testu - strojenia gitary. Po sprawdzeniu częstotliwości odpowiadających kolejnym strunom gitary akustycznej ustaliliśmy tolerowane przedziały częstotliwości, w których traktujemy strunę za nastrojoną. Gdy struna jest nastrojona zapala się przyporządkowana do niej dioda i świeci przez 2 sekundy. W tym czasie odczyt częstotliwości jest wstrzymany. Po 2 sekundach możemy znów wrócić do odczytu częstotliwości drgającej struny.

W naszym projekcie skorzystaliśmy nie tylko z programu Arduino, ale również z języka programistycznego python. Dzięki pythonowi mogliśmy w prosty sposób obliczyć dyskretną transformację Fouriera korzystając z biblioteki numpy. W skrypcie Arduino zostaje wykonany pomiar sygnału, następnie uruchamiając skrypt w pythonie korzystamy z danych przekazywanych do monitora portu szeregowego i dokonujemy na nich przekształceń uzyskując

częstotliwość. Gdy otrzymana częstotliwość należy do któregoś z przedziałów nastrojonej struny, python przesyła informację do monitora portu szeregowego o zapaleniu odpowiedniej diody. W skrypcie Arduino znajdują się warunki, które sprawdzają czy w monitorze portu szeregowego występują informacje na temat zapalenia diody. Po znalezieniu takowej informacji, pobieranie sygnału zostaje wstrzymane na 2s, w tym czasie zapala się odpowiednia dioda LED.

4. Użycie transformaty Fouriera

Odczytany przez nas sygnał nie jest ciągły, próbki pobierane są przez Arduino z pewną częstotliwością, przez co niezbędne jest użycie dyskretnej transformaty Fouriera.

$$\mathcal{F}\{x\}_k = X_k = \sum_{n=0}^{N-1} x_n e^{-i \frac{2\pi k n}{N}}.$$

Znając z wykładu ogólną zasadę działania transformaty przystąpmy do implementacji algorytmu.

Na początek zastanówmy się, jaka powinna być częstotliwość próbkowania. Wskazówką dla nas będzie twierdzenie o próbkowaniu Kotielnikowa-Shannona. Mówi ono, że dla sygnałów składających się z mieszanki fal sinusoidalnych o częstotliwości nie większej niż f , częstotliwość próbkowania powinna być nie mniejsza niż $2f$.

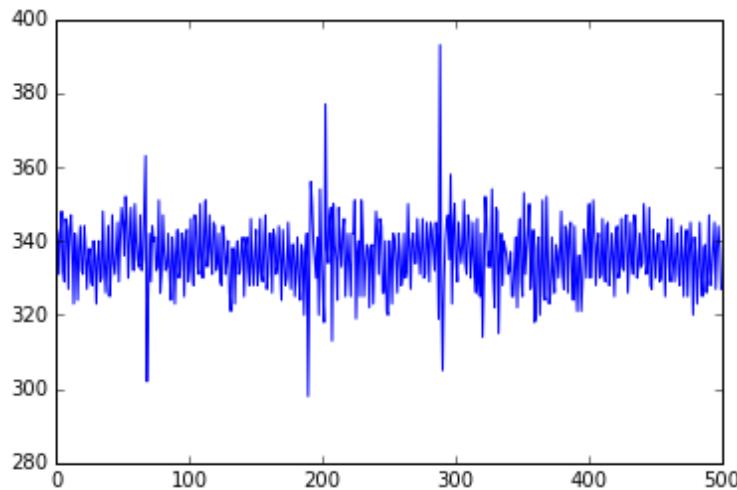
Mierzony przez nas sygnał to dźwięki strun gitarowych mieszczących się w przedziale $80 - 340\text{Hz}$, zatem dostateczna częstotliwość powinno być 640Hz . Częstotliwość uzyskaną przez nas w trakcie pomiarów odczytamy mierząc czas jaki zajęły pomiary.

$$f_p = \frac{1}{T_p}.$$

A więc średnia częstotliwość próbkowania to 2.58kHz co jest wartością wystarczającą do prawidłowego odtworzenia sygnału.

Wiedząc, że transformata zeruje wartości wtedy, gdy $\frac{f}{f_p} = \frac{l}{N}$ dla pewnego $l \in \mathbb{Z}$, zastanówmy się nad ilością próbek. Mając na uwadze tąże wykonywanie pomiarów, dobraliśmy stałą $N = 500 \approx 0.2f_p$.

5. Kod programu Arduino



Rysunek 3. Wykres zebranych danych

```

1 /* Program do obsługi strojka gitarowego Arduino */
2
3 // Deklaracja zmiennych
4 unsigned int knock;
5 float t;
6 const int n = 500; // Liczba probek
7 int cmd = 0;
8
9 void setup()
10 {
11     // Konfiguracja portu szeregowego
12     Serial.begin(115200);
13     // Ustawienie pinow kontrolujacych diody
14     pinMode(3, OUTPUT);
15     pinMode(5, OUTPUT);
16     pinMode(6, OUTPUT);
17     pinMode(9, OUTPUT);
18     pinMode(10, OUTPUT);
19     pinMode(11, OUTPUT);
20     // Arduino czeka na sygnal rozpoczęcia pomiarow
21     Serial.println("Gotowy?");
22     while(Serial.available() == 0);
23     while(Serial.available() > 0) {
24         Serial.read();
25     }
26 }
27
28 void loop()
29 {
30     Serial.print(cmd);
31     t = millis(); // Czas start
32     for(int i=0; i<n; i++){
33         knock = analogRead(0); // Odczyt z mikrofonu
34         Serial.println(knock); // Wyslanie danych do komputera
35     }
36     t = millis() - t; // Obliczenie czasu pomiaru
37 }
```

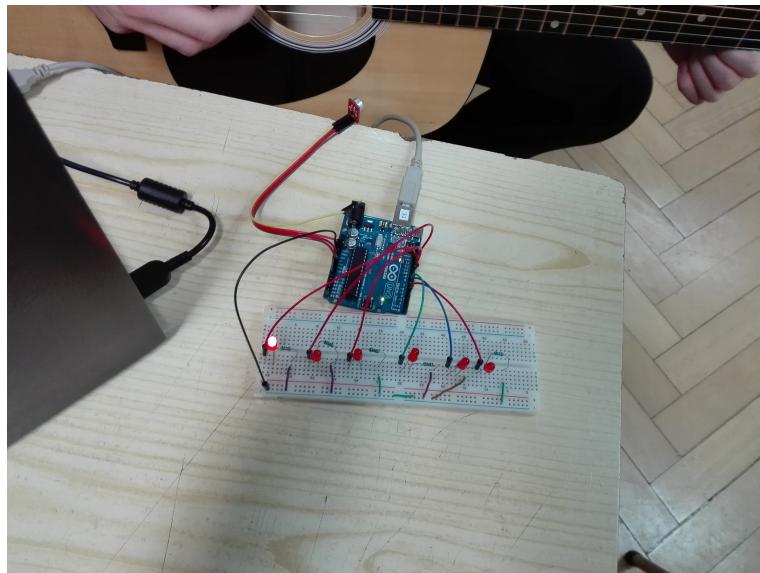
```

38 Serial.println("Czas:");
39 Serial.println(t); // Czas pomiaru trafia do komputera
40
41
42 // Blok kontrolujacy diody
43 while (Serial.available() > 0)
44 {
45     cmd = Serial.read();
46     if (cmd == 51){
47         digitalWrite(3, HIGH);
48         delay(2000);
49         digitalWrite(3, LOW);
50     }
51     else if (cmd == 53){
52         digitalWrite(5, HIGH);
53         delay(2000);
54         digitalWrite(5, LOW);
55     }
56     else if (cmd == 54){
57         digitalWrite(6, HIGH);
58         delay(2000);
59         digitalWrite(6, LOW);
60     }
61     else if (cmd == 57){
62         digitalWrite(9, HIGH);
63         delay(2000);
64         digitalWrite(9, LOW);
65     }
66     else if (cmd == 48){
67         digitalWrite(10, HIGH);
68         delay(2000);
69         digitalWrite(10, LOW);
70     }
71     else if (cmd == 49){
72         digitalWrite(11, HIGH);
73         delay(2000);
74         digitalWrite(11, LOW);
75     }
76     break;
77 }
78
79
80
81 }

```

6. Kod programu python

```
1 # -*- coding: utf-8 -*-
2
3 import matplotlib.pyplot as plt
4 import numpy as np
5 import serial
6
7 # Nazwianie polaczenia z Arduino poprzez port szeregowy
8 port = "COM8"
9 sio = serial.Serial(port, 115200)
10 sio.close()
11 sio.open()
12 sio.readline()
13 # Wyslanie sygnalu rozpoczęcia transmisji danych
14 sio.write("Tak!".encode('ascii'))
15
16 # Deklaracja zmiennych
17 N = 0
18 n = np.arange(N)
19 freq = None
20 t = 0
21
22
23 # Petla interpretacji danych
24 while True:
25     dane = []
26     t = 0
27     N = 0
28
29     while not t:
30         line = sio.readline()
31         try:
32             # Odczyt wysylanych przez mikrokontroler
33             # danych z mikrofonu
34             temp = float(line)
35             dane.append(temp)
36             N = N + 1
37         except ValueError:
38             # Odczyt czasu trwania pomiarow
39             if 'Czas' in str(line[:4]):
40                 t = sio.readline()
41                 t = int(t[:3])
42
43
44     dane = np.array(dane)
45     n = np.arange(N)
46
47     F = (n/(t*0.001)) # Przedzial czestotliwosci
48     fft = np.fft.fft(dane) # Szybka transformata F. danych
49     mod = np.abs(fft)
50
51     l = np.argmax(mod[1:(N+1)/2]) # Pobranie maksymalnego argumentu
52     freq = F[l]
53
54     print(freq, 'Hz')
55     # Blok kontrolujacy zapalanie diod
```



Rysunek 4. Działający model

```
56     if 325.0 < freq < 327.0:  
57         sio.write("3".encode('ascii'))  
58     elif 244.0 < freq < 246.0:  
59         sio.write("5".encode('ascii'))  
60     elif 384.5 < freq < 386.5:  
61         sio.write("6".encode('ascii'))  
62     elif 286.0 < freq < 288.0:  
63         sio.write("9".encode('ascii'))  
64     elif 107.0 < freq < 109.0:  
65         sio.write("0".encode('ascii'))  
66     elif 160.0 < freq < 162.0:  
67         sio.write("1".encode('ascii'))
```

7. Podsumowanie

Mimo wielu trudów udało nam się dobrnąć z naszym modelem do końca. Niestety nie jest on tak dokładny jak przewidywaliśmy, mimo dobrego odczytu częstotliwości nie byliśmy w stanie usunąć szumów, ani ustabilizować tej częstotliwości stąd czasami dość spora rozbieżność z rzeczywistymi wskazaniami. Jednakże projekt uważamy za sukces, ponieważ da się gitarę nastroić dzięki naszemu modelowi. Nie jest to może najdokładniejszy sposób, ale z pewnością jest to możliwe.