**Cyfrowe Przetwarzanie Sygnału**

Projekt: Analogowy pomiar natężenia dźwięku

Sprawozdanie

sierż. pchor. Marcin JAKUBCZYK

WEL20EP1S4

# Wprowadzenie

Celem projektu jest zrealizowanie analogowego pomiaru natężenia dźwięku za pomocą mikrofonu pojemnościowego oraz platformy Raspberry Pi 4. Przetworzenie sygnału zostało dokonane za pomocą języka programowania „Python”.

Projekt zakłada użycie niedrogiego mikrofonu, który umożliwi nagranie ścieżki dźwiękowej. Raspberry Pi pozwoli na zapis oraz pozyskanie widma sygnału. Nagrany dźwięk może zostać odtworzony. Program zapewnia wyświetlenie sygnału w dziedzinie częstotliwości (FFT) oraz czasowej.

Projekt przedstawia szerokie możliwości Raspberry Pi oraz proste rozwiązania, które można wykorzystać w procesie cyfrowego przetwarzania sygnału. Do pisania programu wykorzystano edytor VS Code z zainstalowanym pakietem Python oraz wymaganymi bibliotekami.

# Realizacja

## 2.1 Użyte komponenty

**Mikrofon Defender MIC-111**

Tabela 1 Parametry mikrofonu

|  |  |
| --- | --- |
| Rodzaj | Pojemnościowy |
| Złącze | 3,5 mm |
| Pasmo przenoszenia | 20 Hz – 13 kHz |
| Impedancja | 2200 Ω |
| Czułość | 56 dBA |



Rysunek 1 Defender MIC-111

**Raspberry Pi 4**

Raspberry Pi 4 model B z czterordzeniowym SoC (Broadcom BCM2711, Cortex-A72) pracującym z częstotliwością 1,5GHz, 2GB pamięci RAM LPDDR4, układ graficzny Broadcom VideoCore VI.



Rysunek 2 Raspberry Pi 4

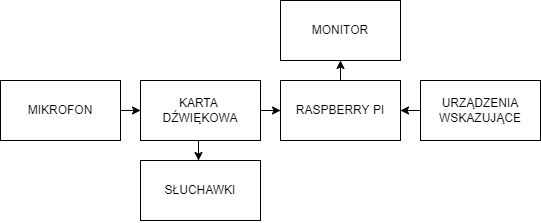
**Karta dźwiękowa 3D Surround**

Karta dźwiękowa Media-tech MT5101 zapewnia wejście mikrofonowe oraz wyjście słuchawkowe. Raspberry Pi 4 posiada tylko wyjście słuchawkowe 3,5 mm. Interfejs USB 2.0, Plug & Play, zasilanie 5V z USB.



Rysunek 3 Karta dźwiękowa USB

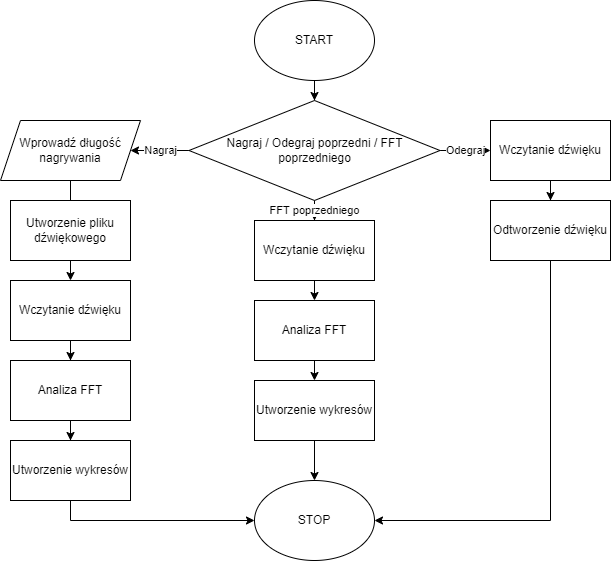
## 2.2 Projekt techniczny



Rysunek 4 Schemat połączenia

Wykonany program w pierwszym etapie nagrywa ścieżkę dźwiękową za pomocą mikrofonu połączonego z karta dźwiękowa i podłączonych do platformy Raspberry Pi. Długość tej ścieżki jest definiowana przez użytkownika. Ścieżka dźwiękowa zapisywana jest na karcie SD w Rpi. Następnie wartości odczytanej amplitudy dźwięku w każdym czasie zapisywane są w tablicy. Na podstawie nagranego fragmentu rysowane są wykresy. Pierwszy prezentuje sygnał w dziedzinie czasu, a drugi w dziedzinie częstotliwości. Program pozwala na pominięcie nagrywania dźwięku, stąd możemy podejrzeć przebiegi poprzedniego dźwięku. Również możemy odtworzyć poprzednie nagrania. Algorytm działania programu przedstawiony jest na rysunku poniżej. Opcjonalnie możemy także zapisać wybrane dane do pliku o rozszerzeniu „.csv”.

Rpi odczytuje wartości napięć pojawiąjących się za membraną mikrofonu, dlatego też aby odczytywać w skali decybelowej natężenie dźwięku należy wykorzystać poniższy wzór:



Rysunek 5 Algorytm działania programu

Kod źródłowy:

import csv

import numpy as np

import scipy.fftpack as fftpk

import scipy.io.wavfile as wavfile

import signal\_envelope as se # Nie działa na Rpi

import sounddevice as sd

import soundfile as sf

from matplotlib import pyplot as plt

samplerate = 44100

filename = 'wav/output1.wav'

def fft():

    # Wywołanie nagrania i zapis

    samplerate, mydata = wavfile.read(filename)

    duration = len(mydata)/samplerate

    print(len(mydata))

    # Realizacja FFT

    fft = abs(fftpk.rfft(mydata))

    freqs = fftpk.rfftfreq(len(fft), (1.0/samplerate))

    # Szukanie y dla x dla wykresu

    max\_y = max(fft)

    max\_x = freqs[fft.argmax()]

    # Oś ox na czas i volt na dB

    time = np.arange(0,duration,1/samplerate)

    mydata = 20\*np.log10(mydata / 5.0)

    # Wykres A(t)[s]

    plt.figure(1)

    plt.subplot(3,1,1)

    plt.plot(time, mydata)

    plt.xlabel('Czas [t]')

    plt.ylabel('Amplituda [dB]')

    plt.title('PRZEBIEG CZASOWY A[t]')

    # Wykres A(f)[Hz]

    plt.subplot(3, 1, 2)

    plt.scatter(max\_x, max\_y, c='r')

    plt.plot(freqs[range(len(fft)//2)], fft[range(len(fft)//2)], '-r')

    plt.legend(["Freq: %2.2f [Hz]" %max\_x], loc='upper center' )

    plt.xscale('log')

    plt.xlabel('Częstotliwość (Hz)')

    plt.ylabel('Amplituda')

    plt.title("WIDMO SYGNAŁU F[f]")

    # Obwiednia do wyznaczenia ch-ki kierunkowej i ch-ka kier

    # Ch-ka tylko dla nagrania "output3.wav" !!!!

    W, \_ = se.read\_wav(filename)

    X\_envelope = se.get\_frontiers(W, 1)

    ##Do plotu dla ch-ki

    os\_y =[]

    absx = np.abs(W[X\_envelope])

    absx = list(absx)

    for x in range(len(absx)):

        if absx[x] > 150:

            os\_y.append(absx[x])

    rozm = np.size(os\_y)

    krok = 360/rozm

    os\_x = []

    krok\_0 = 0

    for i in range(rozm):

        os\_x.append(krok\_0)

        krok\_0+=krok

    plt.subplot(3, 1, 3)

    plt.plot(X\_envelope, np.abs(W[X\_envelope]), '-g')

    plt.title("OBWIEDNIA SYGNAŁU A[t]")

    plt.ylabel('Amplituda [Pa]')

    plt.tight\_layout()

    plt.show()

    ##PLOT dla Ch-ki [ tylko dla nagrania "output3.wav" ]

    plt.figure(2)

    plt.plot(os\_x, os\_y)

    plt.title("CH-ka kierunkowa mikrofonu")

    plt.xlabel('Kąt [st]')

    plt.show()

    # Zapis danych do .csv [ dane do wyboru ]

    with open('exel\data\_file.csv', 'w') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(mydata)

# Sprawdzenie pliku .wav czy jest w folderze

def check():

    try:

        samplerate, mydata = wavfile.read(filename)

    except IOError:

        print("Z pustego i Salomon nie naleje! Nagraj coś :)")

        exit()

INP\_0 = int(input(" Nagraj - 1 \n FFT poprzedniego nagrania - 2 \n Odegraj poprzednie nagranie - 3 \n"))

if INP\_0 == 1:

    duration = int(input(" Podaj czas nagrywania [s]  "))

    print("     START")

    mydata = sd.rec(int(samplerate \* duration), samplerate=samplerate, channels=1, blocking=True)

    sf.write(filename, mydata, samplerate)

    print("     STOP")

    fft()

elif INP\_0 == 2:

    check()

    fft()

else:

    check()

    samplerate, mydata = sf.read(filename)

    print("     START")

    sd.play(samplerate, mydata)

    sd.wait()

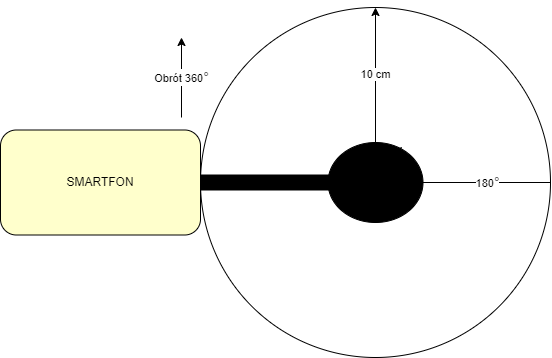
    print("     STOP")

    exit()

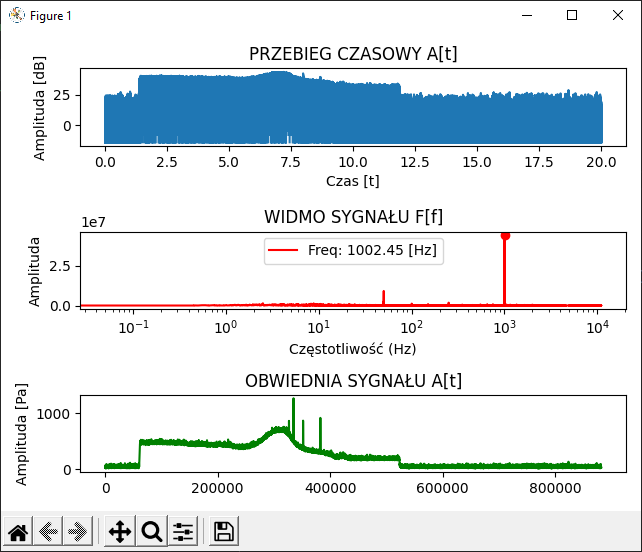
# Badanie Funkcjonalne

## Charakterystyka kierunkowa mikrofonu

W pierwszym etapie zbadano charakterystykę kierunkową mikrofonu. Badanie zrealizowano za pomocą odtwarzanego dźwięku sinusa o częstotliwości 1 kHz. Do zapewniania stałej odległości głośnika telefonu od mikrofonu, telefon zawieszono na sznurku, który podczepiono do głowicy mikrofonu. Odległość telefonu od mikrofonu wynosiła 10 cm. Począwszy od trzonka mikrofonu (0°), poprowadzono telefon po okręgu, zatrzymując się w tym samym miejscu, tym samym robiąc pełen obrót ze stałą prędkością.

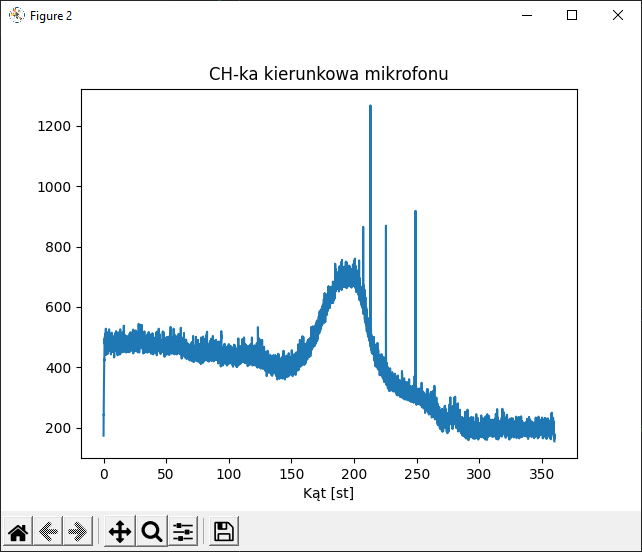


Rysunek 6 Sposób pomiaru ch-ki kierunkowej



Rysunek 7 Uzyskane przebiegi pomiaru charakterystyki

Wydobycie gotowej charakterystyki kierunkowej polegało na wycięciu nagranego sygnału sinusa (output3.wav) oraz zastąpienie wartości czasu na stopnie na osi odciętych (rysunek 8). Natężenie nagranego dźwięku wynosiło 42 dB (± 2 dB).



Rysunek 8 Charakterystyka kierunkowa mikrofonu MIC -111

Z powyższego przebiegu można wnioskować, iż powyższa charakterystyka zbliżona jest do charakterystyki jednokierunkowej kardiodalnej (rysunek 9). Wskazuje na to większa reakcja na dźwięk w otoczeniu 180°. Prawdą jest też, że mikrofon zbiera dźwięk również z innych stron. Jednak należy zaznaczyć, że badania prowadzono w warunkach domowych, gdzie nie zapewniono idealnego, bez odbiciowego środowiska z minimalną ingerencją szumów z otoczenia. Taką charakterystyką cechuje się większość mikrofonów pojemnościowych.

Drugi w kolejności wykres przedstawia widmo sygnału z prążkiem na 1 kHz (rysunek 7). Można zauważyć, że widmo przedstawia również zakłócenia w postaci bardzo niskich prążków w całym paśmie mikrofonu. Są to szumy pochodzące z otoczenia, ale także z drogi jaką musi przejść sygnał przez elementy elektroniczne, aż do samego zapisu cyfrowego.



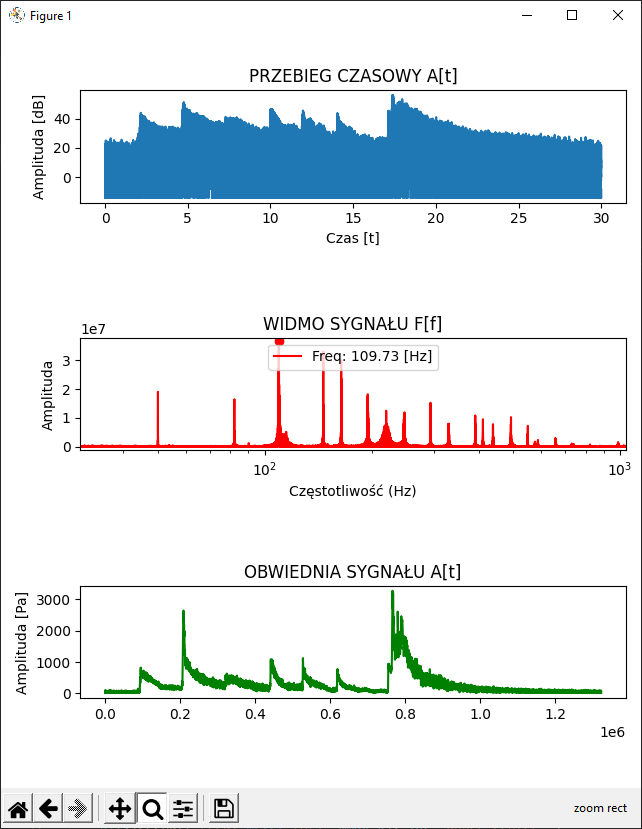
Rysunek 9 Ch-ka kierunkowa kardioidalna

## Badanie dźwięku gitary klasycznej

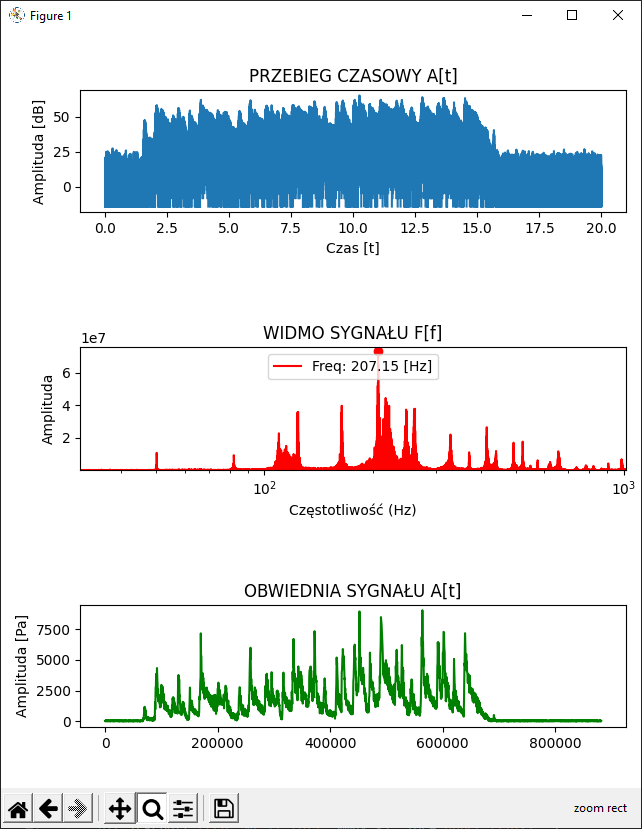
Podczas badań jedną ze ścieżek dźwiękowych była nagrana odpowiednio nastrojona gitara klasyczna. Pierwsze utworzone wykresy przedstawiają pojedynczo zagrane struny (rysunek 10). Natomiast wykresy z drugiej fotografii przedstawiają zagrany fragment utworu z wykorzystaniem chwytów A-mol i E-dur (rysunek 11). Odpowiednio plik fileguitar.wav i fileguitar2.wav.

Tabela 2 Częstotliwości strun odpowiednio nastrojonej gitary

|  |  |  |
| --- | --- | --- |
| Numer struny | Nuta | Częstotliwość [Hz] |
| 1 | E4 | 329,63 |
| 2 | B3 | 246,94 |
| 3 | G3 | 196,00 |
| 4 | D3 | 146,83 |
| 5 | A2 | 110,00 |
| 6 | E2 | 82,41 |



Rysunek 10 Przetworzona ścieżka dźwiękowa nagranej gitary fileguitar.wav



Rysunek 11 Przetworzona ścieżka dźwiękowa nagranej gitary fileguitar2.wav

Widmo sygnału obu ścieżek dźwiękowych są bardzo podobne. Liczba prążków na wykresach równa jest 15. Nie obserwujemy tylko 6 prążków, gdyż sam sposób wydobycia dźwięku nie jest idealny. Każde dotknięcie struny dłonią czy kostką wydobywa dźwięk. Przesunięcia po gryfie, nieodpowiednia zmiana chwytu lub uderzenia o pudło rezonansowe także wywołują niepożądane dźwięki. Jednak na wykresie można wyznaczyć prążki charakteryzujące poszczególne struny gitary. Natężenie gitary wyniosło około 65 dB. Dla porównania stopnia natężenia dźwięku poniżej dołączono przykładowe wartości natężenia odpowiadające różnym źródłom dźwięku.

* 10 dB – szelest liści przy łagodnym wietrze
* 20 dB – szept
* 30 dB – bardzo spokojna ulica bez ruchu
* 40 dB – szmery w domu
* 50 dB – szum w biurach
* 60 dB – odkurzacz
* 70 dB – wnętrze głośnej restauracji, darcie papieru
* 80 dB – głośna muzyka w pomieszczeniach, trąbienie
* 90 dB – ruch uliczny
* 100 dB – motocykl bez tłumika
* 110 dB – piła łańcuchowa
* 120 dB - maksymalny dopuszczalny poziom natężenia dźwięku fajerwerków
* 130 dB – wirnik helikoptera w odległości 5 metrów, granica powyżej której może dojść do trwałego uszkodzenia słuchu
* 140 dB – start myśliwca
* 150 dB - wystrzał z karabinu
* 160 dB – eksplozja bomby
* 190 dB – start rakiety kosmicznej
* 220 dB – eksplozja bomby atomowej

# Wnioski

Rezultatem powyższej pracy jest program napisany w języku Python, uruchomiony na platformie Raspberry Pi. Program umożliwia analizę widmową sygnału niskiej częstotliwości. Napisany program, pobiera sygnał akustyczny z pliku dźwiękowego, może również nagrać nowy dźwięk, jak i go odtworzyć.

W badaniach nie zostały spełnione odpowiednie warunki akustyczne np. wygłuszenie otoczenia, dlatego obserwujemy niepożądane prążki na wykresach widma.

W projekcie napotkano problem z uruchomieniem biblioteki „signal\_envelope” na mini-komputerze Rpi. Stąd część badań (ch-ka kierunkowa mikrofonu) została zrealizowana przy pomocy systemu Windows 10 i PC. Bez powyższej biblioteki program działa odpowiednio na Rpi.

Do przedstawionego programu można wprowadzić szereg usprawnień i dodatkowych funkcji, a przede wszystkim opracować graficzny interfejs użytkownika. Jedną z propozycji może być rzeczywiste rysowanie wykresów, dzięki którym można nastroić instrumenty lub określić dopuszczalny poziom natężenia nasłuchiwanego dźwięku.

# Załączniki

/CPS\_project

* fft.py

/CPS\_project/wav

* fileguitar.wav
* fileguitar2.wav
* outpu1
* outpu2
* outpu3
* outpu4