# Lab 07

Karol Działowski

Nagrano słowo mama z wydłużonym pierwszym "a". Wycięto samogłoskę za pomocą programu audacity.

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import librosa
import librosa.display
from scipy.io import wavfile
from scipy import signal

import warnings
warnings.filterwarnings('ignore')
```

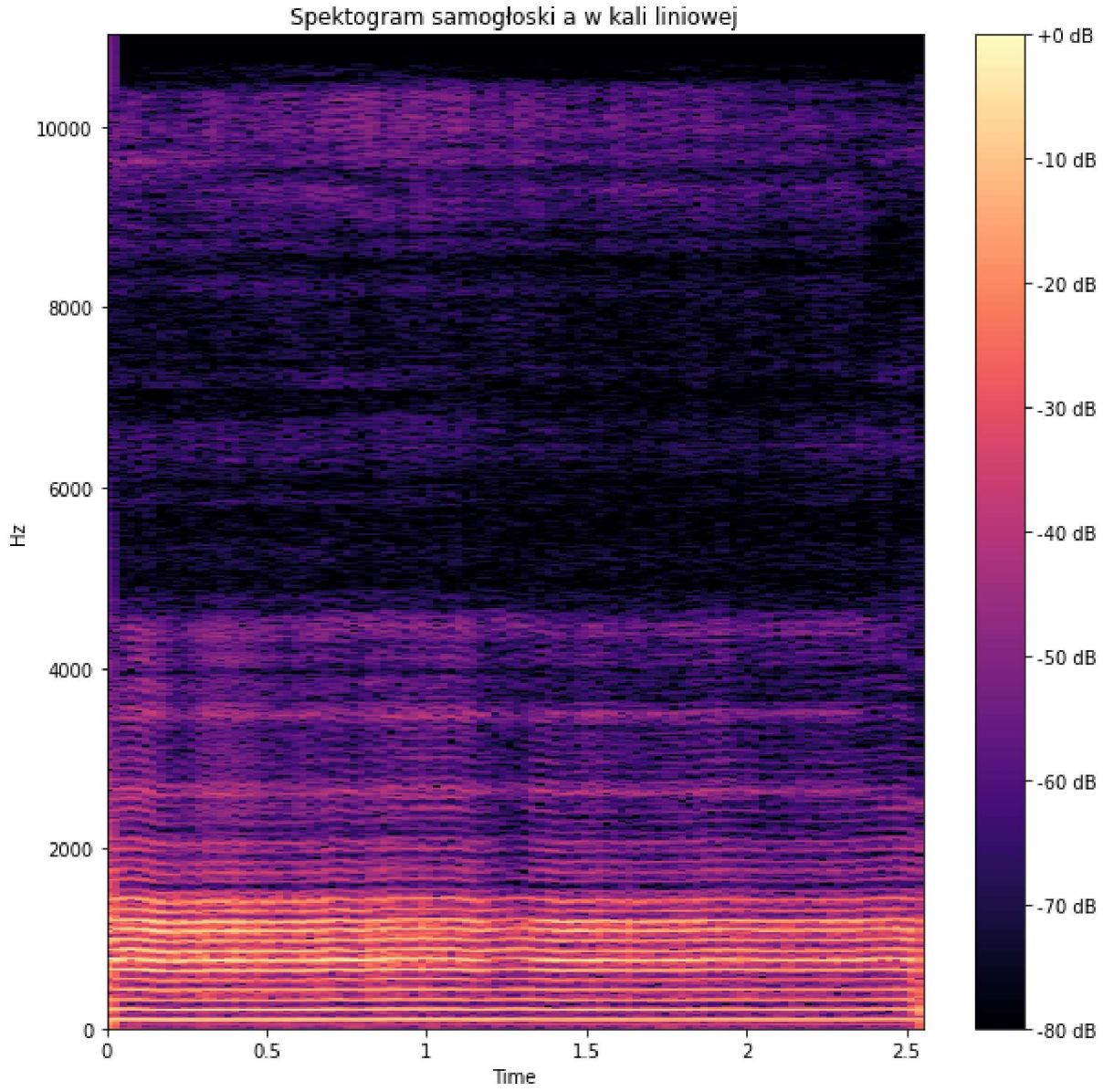
## Wczytanie pliku dźwiękowego

In [3]:
```
y, sr = librosa.load("./data/a1.wav", mono=True)
#y = sound.astype('float32')
```

## Spektogram sygnału

In [4]:
```
def spectogram(y, title):
    D = librosa.stft(y)  # STFT of y
    S_db = librosa.amplitude_to_db(np.abs(D), ref=np.max)
    fig, ax = plt.subplots(figsize=(10, 10))
    img = librosa.display.specshow(S_db, x_axis='time', y_axis='linear', ax=ax)
    ax.set(title=title)
    fig.colorbar(img, ax=ax, format="%+2.f dB")
    return S_db
```

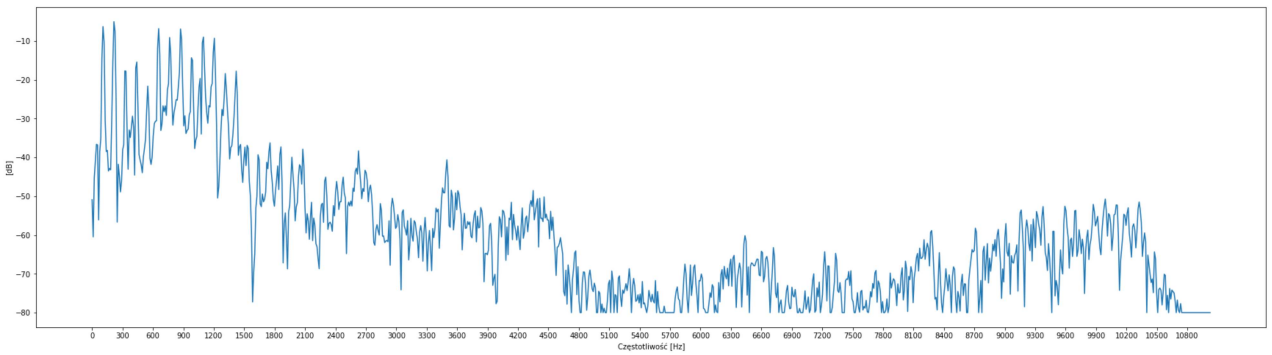
In [5]:
```
spec_a = spectogram(y, "Spektogram samogłoski a w kali liniowej")
```




# Wykres w wycinku czasowym

In [6]:

```
frequencies = librosa.fft_frequencies(sr=22050, n_fft=2048)
fig = plt.figure(figsize=(30, 8))
plt.plot(frequencies, spec_a[:, 20])
plt.xlabel("Częstotliwość [Hz]")
plt.ylabel("[dB]")
_ = plt.xticks(np.arange(0, frequencies[-1], step=300))
```



# Częstotliwość fundamentalna

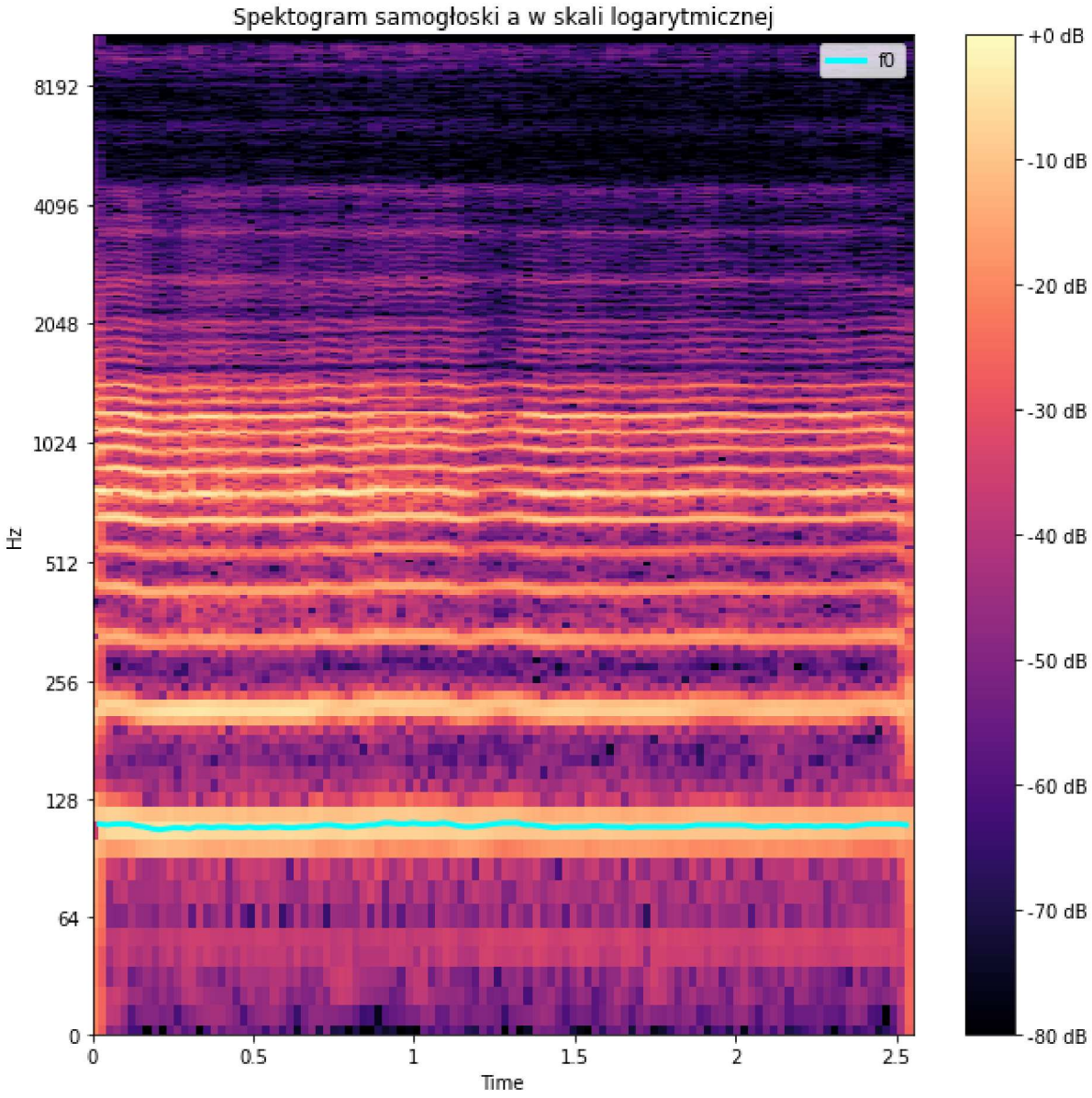
https://librosa.org/doc/main/generated/librosa.pyin.html

In [7]:
```
f0, voiced_flag, voiced_probs = librosa.pyin(y, fmin=librosa.note_to_hz('C2'), fmax=lib
times = librosa.times_like(f0)
```

# Częstotliwość fundamentalna na spektogramie

In [8]:
```
D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
fig, ax = plt.subplots(figsize=(10, 10))
img = librosa.display.specshow(D, x_axis='time', y_axis='log', ax=ax)
ax.set(title='Spektogram samogłoski a w skali logarytmicznej')
fig.colorbar(img, ax=ax, format="%+2.f dB")
ax.plot(times, f0, label='f0', color='cyan', linewidth=3)
ax.legend(loc='upper right')
```

Out[8]:
```
<matplotlib.legend.Legend at 0x1df98ef1400>
```




In [9]:
```
fundamental_frequency = np.nanmean(f0)
print("Częstotliwość fundamentalna na całym sygnale:", fundamental_frequency)
```

```
Częstotliwość fundamentalna na całym sygnale: 110.01029285687771
```

# Formanty

Należało wyznaczyć 3 formanty. Powinny być to kolejno około 700, 1000, 2000.

https://www.wikiwand.com/en/Formant

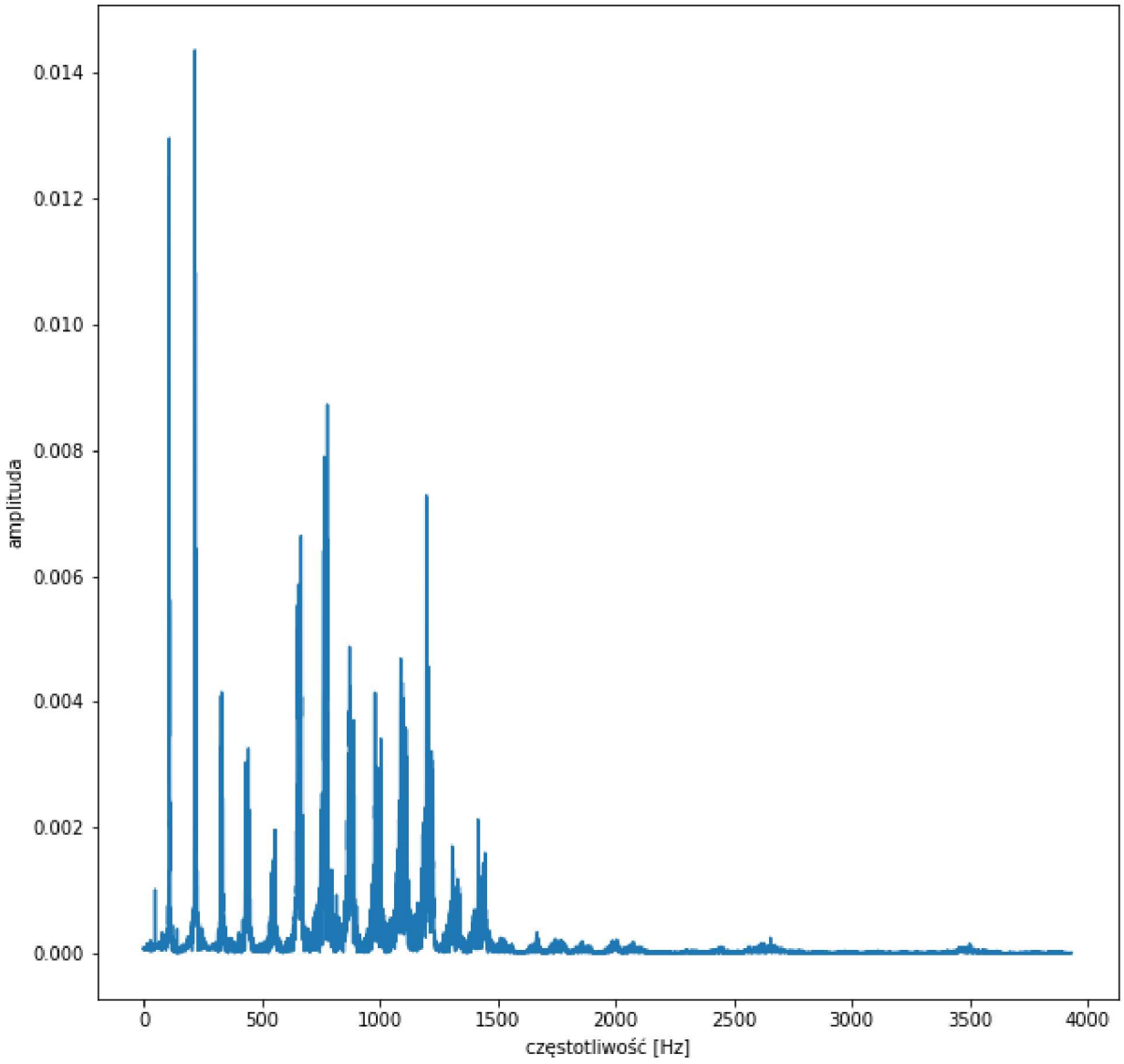
https://stackoverflow.com/questions/25107806/estimate-formants-using-lpc-in-python

https://stackoverflow.com/questions/50428617/substitution-of-lpc-in-python-to-estimate-formants-frequency

## Widmo sygnału

https://librosa.org/doc/0.7.0/generated/librosa.onset.onset_strength.html

In [10]:
```
def plot_widmo(y, fs, title):
    widmo_amp = np.abs(np.fft.rfft(y) / (len(y)//2))
    f = np.fft.rfftfreq(len(y), 1/fs)
    plt.plot(f[:10000], widmo_amp[:10000])
    plt.xlabel('częstotliwość [Hz]')
    plt.ylabel('amplituda')
#     plt.ylim([-300, 0])
    plt.title(title)
    plt.show()

plt.figure(figsize=(10, 10))
plot_widmo(y, sr, "")
```




## Podejście z LPC - niepoprawnie wyznaczone formanty

Próba wyznaczenia formantów z nagranego sygnału za pomocą LPC

https://stackoverflow.com/questions/61519826/how-to-decide-filter-order-in-linear-prediction-coefficients-lpc-while-calcu

In [11]:
```
import numpy as np
import librosa

A = librosa.core.lpc(y, 12)
rts = np.roots(A)
rts = rts[np.imag(rts) >= 0]
angz = np.arctan2(np.imag(rts), np.real(rts))
frqs = angz * (sr*2) / (2 *  np.pi)
frqs.sort()
```

In [12]:
```
frqs
```

Out[12]:
```
array([    0.    ,   808.0363,  2193.5984,  7950.949 , 14560.737 ,
       19611.732 , 22050.    ], dtype=float32)
```

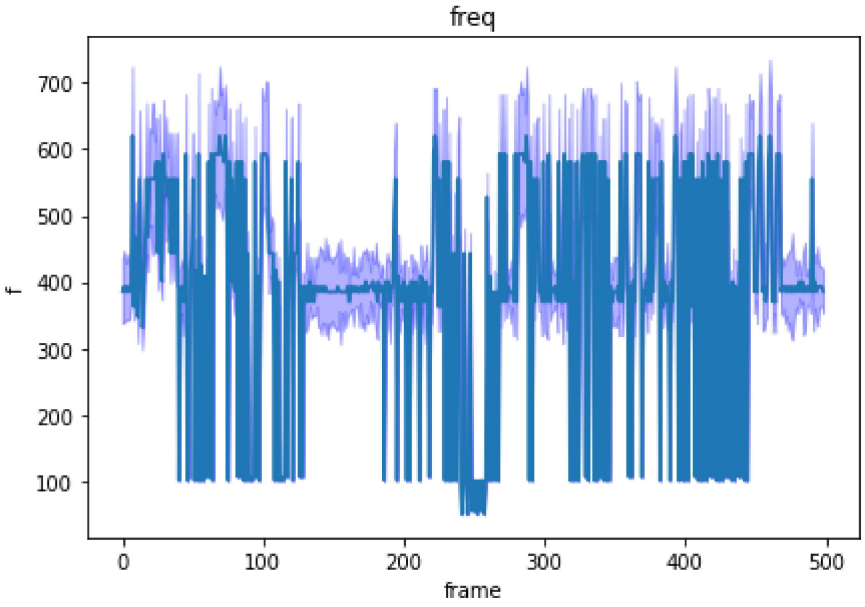
## Podejście z formantfeatures - niepoprawnie wyznaczone formanty

Próba wyznaczenia formantów przy użyciu biblioteki formantfeatures.

In [13]:
```
import numpy as np
import formantfeatures as ff
import matplotlib.pyplot as plt


test_wav = "data/a1.wav" #A sample from RAVDESS

window_length = 0.025   #Keep it such that its easier to differentiate syllables and re
window_step = 0.010
emphasize_ratio = 0.65
f0_min = 30
f0_max = 4000
max_frames = 500
max_formants = 3

formants_features, frame_count, signal_length, trimmed_length = ff.Extract_wav_file_for

print("formants_features max_frames:", formants_features.shape[0], " features count:",

for formant in range(max_formants):
    print("Formant", formant, "Mean freq:", np.mean(formants_features[0:frame_count, (f
    print("Formant", formant, "Mean power:", np.mean(formants_features[0:frame_count, (
    print("Formant", formant, "Mean width:", np.mean(formants_features[0:frame_count, (
    print("Formant", formant, "Mean dissonance:", np.mean(formants_features[0:frame_cou


x_axis_i = [*range(0, frame_count, 1)]

colors = ['b', 'r', 'g']

for formant in range(0, 1):
    formant_decay_rate = 0.5**(formant)

    log_scaled_freq = formants_features[0:frame_count, formant*4]
```

```
    Hz_freq = np.exp(log_scaled_freq / (200*formant_decay_rate))

    Hz_width  = np.exp(np.log(Hz_freq) - formants_features[0:frame_count, (formant*4)+2

    width_dn = Hz_freq - Hz_width
    width_up = Hz_freq + Hz_width

    plt.plot(x_axis_i, Hz_freq)
    plt.fill_between(x_axis_i, Hz_freq, width_dn, color=colors[formant], alpha=0.30)
    plt.fill_between(x_axis_i, Hz_freq, width_up, color=colors[formant], alpha=0.30)


plt.tight_layout()
plt.xlabel("frame")
plt.ylabel("f")
plt.title("freq")


plt.show()
```

```
Warning! Frame size overflow, Size: 506 Limit: 500
formants_features max_frames: 500  features count: 12 frame_count 499
Formant 0 Mean freq: 1157.5751503006013
Formant 0 Mean power: 647.815631262525
Formant 0 Mean width: 60.51102204408818
Formant 0 Mean dissonance: 21.382765531062123
Formant 1 Mean freq: 541.2665330661323
Formant 1 Mean power: 313.19639278557116
Formant 1 Mean width: 37.89178356713427
Formant 1 Mean dissonance: 5.841683366733467
Formant 2 Mean freq: 210.04809619238478
Formant 2 Mean power: 118.39078156312625
Formant 2 Mean width: 12.889779559118237
Formant 2 Mean dissonance: 1.1983967935871744
```




# Syntezator Klatta

Jako, że nie udało się poprawnie wyznaczyć formantów, do syntezatora klatta wykorzystam częstotliwości dostarczone w instrukcji laboratorium.

TABLE II. Averages of fundamental and formant frequencies and formant amplitudes of vowels by 76 speakers.

| | | i | ɪ | ɛ | æ | ɑ | ɔ | ʊ | u | ʌ | ɝ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Fundamental frequencies (cps) | *M* | 136 | 135 | 130 | 127 | 124 | 129 | 137 | 141 | 130 | 133 |
| | *W* | 235 | 232 | 223 | 210 | 212 | 216 | 232 | 231 | 221 | 218 |
| | *Ch* | 272 | 269 | 260 | 251 | 256 | 263 | 276 | 274 | 261 | 261 |
| Formant frequencies (cps) | | | | | | | | | | | |
| $F_1$ | *M* | 270 | 390 | 530 | 660 | 730 | 570 | 440 | 300 | 640 | 490 |
| | *W* | 310 | 430 | 610 | 860 | 850 | 590 | 470 | 370 | 760 | 500 |
| | *Ch* | 370 | 530 | 690 | 1010 | 1030 | 680 | 560 | 430 | 850 | 560 |
| $F_2$ | *M* | 2290 | 1990 | 1840 | 1720 | 1090 | 840 | 1020 | 870 | 1190 | 1350 |
| | *W* | 2790 | 2480 | 2330 | 2050 | 1220 | 920 | 1160 | 950 | 1400 | 1640 |
| | *Ch* | 3200 | 2730 | 2610 | 2320 | 1370 | 1060 | 1410 | 1170 | 1590 | 1820 |
| $F_3$ | *M* | 3010 | 2550 | 2480 | 2410 | 2440 | 2410 | 2240 | 2240 | 2390 | 1690 |
| | *W* | 3310 | 3070 | 2990 | 2850 | 2810 | 2710 | 2680 | 2670 | 2780 | 1960 |
| | *Ch* | 3730 | 3600 | 3570 | 3320 | 3170 | 3180 | 3310 | 3260 | 3360 | 2160 |
| Formant amplitudes (db) | $L_1$ | −4 | −3 | −2 | −1 | −1 | 0 | −1 | −3 | −1 | −5 |
| | $L_2$ | −24 | −23 | −17 | −12 | −5 | −7 | −12 | −19 | −10 | −15 |
| | $L_3$ | −28 | −27 | −24 | −22 | −28 | −34 | −34 | −43 | −27 | −20 |

# guestdaniel/tdklatt

Na początku wykorzystano implementację rezonatora Klatta dostępną w repozytorium: https://github.com/guestdaniel/tdklatt w celu przetestowania działania syntezatora

In [14]:
```
import tdklatt
```

In [40]:
```
s = tdklatt.klatt_make() # Creates a Klatt synthesizer w/ default settings
s.run()
```

In [41]:
```
s.play()
```

In [42]:
```
s.params
```

Out[42]:
```
{'F0': array([100., 100., 100., ..., 100., 100., 100.]),
 'AV': array([60., 60., 60., ..., 60., 60., 60.]),
 'OQ': None,
 'SQ': None,
 'TL': None,
 'FL': None,
 'DI': None,
 'AVS': array([0., 0., 0., ..., 0., 0., 0.]),
 'AF': array([0., 0., 0., ..., 0., 0., 0.]),
 'AH': array([0., 0., 0., ..., 0., 0., 0.]),
 'FF': [array([500., 500., 500., ..., 500., 500., 500.]),
  array([1500., 1500., 1500., ..., 1500., 1500., 1500.]),
  array([2500., 2500., 2500., ..., 2500., 2500., 2500.]),
  array([3500., 3500., 3500., ..., 3500., 3500., 3500.]),
  array([4500., 4500., 4500., ..., 4500., 4500., 4500.])],
 'BW': [array([50., 50., 50., ..., 50., 50., 50.]),
  array([100., 100., 100., ..., 100., 100., 100.]),
  array([100., 100., 100., ..., 100., 100., 100.]),
  array([200., 200., 200., ..., 200., 200., 200.]),
  array([250., 250., 250., ..., 250., 250., 250.])],
 'FGP': array([0., 0., 0., ..., 0., 0., 0.]),
 'BGP': array([100., 100., 100., ..., 100., 100., 100.]),
 'FGZ': array([1500., 1500., 1500., ..., 1500., 1500., 1500.]),
```

```
'BGZ': array([6000., 6000., 6000., ..., 6000., 6000., 6000.]),
'BGS': array([200., 200., 200., ..., 200., 200., 200.]),
'FNP': array([250., 250., 250., ..., 250., 250., 250.]),
'BNP': array([100., 100., 100., ..., 100., 100., 100.]),
'FNZ': array([250., 250., 250., ..., 250., 250., 250.]),
'BNZ': array([100., 100., 100., ..., 100., 100., 100.]),
'FTP': None,
'BTP': None,
'FTZ': None,
'BTZ': None,
'A2F': None,
'A3F': None,
'A4F': None,
'A5F': None,
'A6F': None,
'B2F': None,
'B3F': None,
'B4F': None,
'B5F': None,
'B6F': None,
'A1V': None,
'A2V': None,
'A3V': None,
'A4V': None,
'ATV': None,
'A1': array([0., 0., 0., ..., 0., 0., 0.]),
'A2': array([0., 0., 0., ..., 0., 0., 0.]),
'A3': array([0., 0., 0., ..., 0., 0., 0.]),
'A4': array([0., 0., 0., ..., 0., 0., 0.]),
'A5': array([0., 0., 0., ..., 0., 0., 0.]),
'AN': array([0., 0., 0., ..., 0., 0., 0.]),
'ANV': None,
'SW': array([0., 0., 0., ..., 0., 0., 0.]),
'INV_SAMP': None,
'N_SAMP': 10000,
'FS': 10000,
'DT': 0.0001,
'VER': 'KLSYN80',
'A6': array([0., 0., 0., ..., 0., 0., 0.]),
'DUR': 1,
'N_FORM': 5}
```

## Samogłoska "a"

In [72]:
```
s.params["F0"] = np.ones(s.params["N_SAMP"])*110 # Change F0 to steady-state 200 Hz
formants = [np.ones(s.params["N_SAMP"])*730, np.ones(s.params["N_SAMP"])*1090, np.ones(
s.params["FF"] = formants
```

In [73]:
```
s.run()
s.play()
```

In [74]:
```
wavfile.write(f"./data/a_synth.wav", 16000, y)
```

## Samogłoska "i"

In [75]:
```
s.params["F0"] = np.ones(s.params["N_SAMP"])*110 # Change F0 to steady-state 200 Hz
formants = [np.ones(s.params["N_SAMP"])*270, np.ones(s.params["N_SAMP"])*2300, np.ones(
s.params["FF"] = formants
```

In [76]:
```
s.run()
s.play()
```

In [77]:
```
wavfile.write(f"./data/i_synth.wav", 16000, y)
```

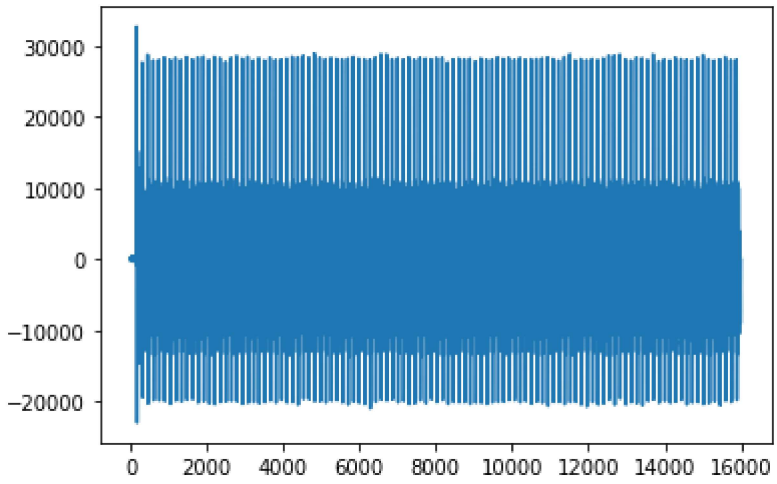
## Samogłoska "o"

In [78]:
```
s.params["F0"] = np.ones(s.params["N_SAMP"])*110 # Change F0 to steady-state 200 Hz
formants = [np.ones(s.params["N_SAMP"])*500, np.ones(s.params["N_SAMP"])*800, np.ones(s
s.params["FF"] = formants
```

In [82]:
```
s.run()
s.play()
```

In [80]:
```
y = s._get_int16at16K()
plt.plot(y)
```

Out[80]: [<matplotlib.lines.Line2D at 0x1df98f7ab50>]

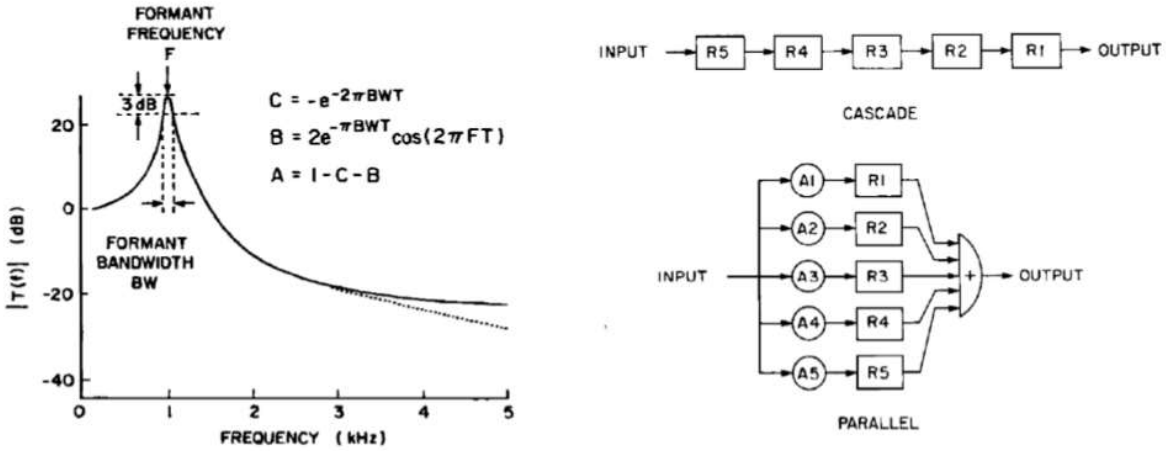



In [81]:
```
wavfile.write(f"./data/o_synth.wav", 16000, y)
```

## Wnioski

Generowane samogłoski na podstawie podanych częstotliwości fundamentalnych i formantów rzeczywiście generowane są w sposób zrozumiały. Z łatwością można powiedzieć, że nie jest to dźwięk naturalny, bo brzmi "metalicznie".

# Własna implementacja syntezatora Klatta




Rys. 1. Cyfrowy rezonator Klatta

Cyfrowy rezonator Klatta opisany jest przez równanie różnicowe drugiego rzędu posiadający odpowiedź częstotliwościową opisaną zależnością:

$$H(z) = \frac{A}{1 - Bz^{-1} - Cz^{-2}}$$

gdzie:

$C = -e^{-2\pi \cdot BW \cdot T}$

$B = 2e^{-\pi \cdot BW \cdot T} \cos(2\pi \cdot F \cdot T)$

$A = 1 - B - C$

Praktyczna realizacja sprowadza się do realizacji następującego równania różnicowego:

$$y(n) = Ax(n) + By(n-1) + Cy(n-2)$$

Po analizie kodu gotowej implementacji syntezatora Klatta postanowiłem pominąć ten podpunkt.