# Protokol k projektu ISS 2024/25

## Rozpoznávanie zvukov automobilových motorov

**Autor**: Roman Nečas

**Login**: xnecasr00

**Dátum**: 7.12.2024

## 1. Úvod a cieľ projektu

Projekt sa zaoberá automatickým rozpoznávaním a klasifikáciou zvukov automobilových motorov. Hlavným cieľom je vyvinúť metodiku, ktorá dokáže priradiť neznáme zvukové nahrávky k referenčným nahrávkam rovnakého automobilu, aj keď motor beží na iných otáčkach.

## 2. Teoretický rozbor

### 2.1 Charakteristika zvuku motora

Zvuk motora je komplexný signál obsahujúci niekoľko kľúčových charakteristík:

- Základná frekvencia závislá od otáčok motora
- Harmonické frekvencie (násobky základnej frekvencie)
- Špecifické spektrálne charakteristiky dané konštrukciou motora

### 2.2 Metódy analýzy

Pre analýzu som zvolil nasledujúce metódy:

1. Časová analýza

- Zobrazenie priebehu signálu
- Výpočet základných štatistických parametrov

2. Frekvenčná analýza

- Fourierova transformácia (FFT)
- Analýza spektra
- Identifikácia základnej frekvencie

3. Normalizovaná spektrálna analýza

- Normalizácia spektra vzhľadom na základnú frekvenciu
- Normalizácia amplitúd

- Analýza harmonických komponentov

Dôvod zvolenia týchto metód bol, aby som sa dostal k charakteristike zvukových nahrávok motorov, v ktorej by boli otáčky motora, teda základná frekvencia "odfiltrované" a mohol som porovnavať zvukové nahrávky nezávisle na otáčkach motora.

# 3. Implementácia

## Knižnice a stiahnutie signálu

In [1]:
```
import os
import re
import glob
import soundfile as sf
from IPython.display import Audio
from IPython.display import display
from scipy import signal
from scipy.io import wavfile
from scipy.fft import fft, ifft, fftfreq
import scipy.io
import matplotlib.pyplot as plt
import numpy as np
```

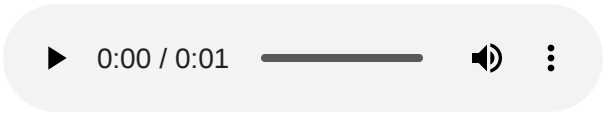
In [2]:
```
login = "xnecasr00"
zip_file = login + ".zip"
assignment_file = "https://www.fit.vut.cz/study/course/ISS/public/proj2024-2
!wget $assignment_file
!unzip -o $zip_file
```

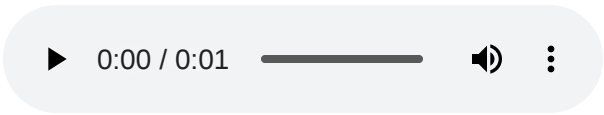
```
--2024-12-07 15:43:24--  https://www.fit.vut.cz/study/course/ISS/public/proj
2024-25/personal/xnecasr00.zip
Resolving www.fit.vut.cz (www.fit.vut.cz)... 147.229.9.65, 2001:67c:1220:80
9::93e5:941
Connecting to www.fit.vut.cz (www.fit.vut.cz)|147.229.9.65|:443... connecte
d.
HTTP request sent, awaiting response... 200 OK
Length: 227217 (222K) [application/zip]
Saving to: ‘xnecasr00.zip’

xnecasr00.zip       100%[===================>] 221.89K   340KB/s    in 0.7s

2024-12-07 15:43:25 (340 KB/s) - ‘xnecasr00.zip’ saved [227217/227217]

Archive:  xnecasr00.zip
   creating: xnecasr00/
  inflating: xnecasr00/BMW_318i_Drive.wav
  inflating: xnecasr00/test_f.wav
  inflating: xnecasr00/Fiat_Panda_Drive.wav
  inflating: xnecasr00/test_t.wav
  inflating: xnecasr00/Audi_A3_Drive.wav
  inflating: xnecasr00/test_s.wav
  inflating: xnecasr00/Peugeot_307_Drive.wav
  inflating: xnecasr00/test_i.wav
```

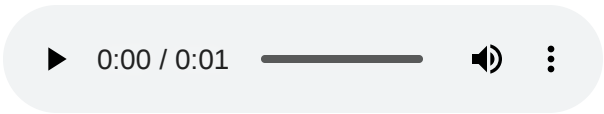
In [3]:

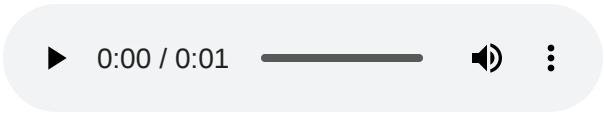
```python
# load the data
# references will be in ref_signals, reference labels in ref_labels, referen
# tests  will be in test_signals, test labels in test_labels, test count in
def get_signals(labs):
    signals = []
    N = len(labs)
    for car in labs:
      filename = login + "/" + car + ".wav"
      s, Fs = sf.read(filename)
      signals.append(s)
    return signals, N, Fs

def play_signals(signals, Fs):
  for signal in signals:
    display(Audio(signal, rate=Fs))

files = glob.glob(login + "/*.wav")
names = [re.sub(login + "/", "", s) for s in files]
labels = [re.sub(".wav", "",s) for s in names]
print ("----- test signals ---------")
r = re.compile("^test_");  test_labels = list(filter(r.match, labels))
print (test_labels); test_signals, N_test, Fs = get_signals(test_labels); pl
print ("----- reference signals ---------")
r = re.compile("(?!^test_)");  ref_labels = list(filter(r.match, labels))
print (ref_labels); ref_signals, N_ref, Fs = get_signals(ref_labels); play_s
```

```
----- test signals ---------
['test_t', 'test_s', 'test_f', 'test_i']
```

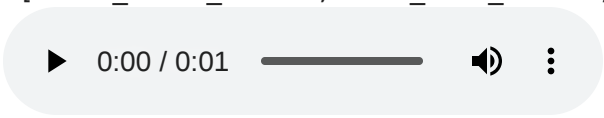


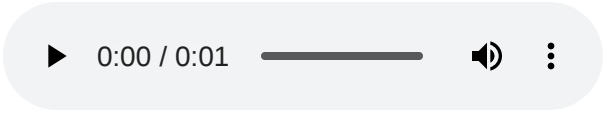


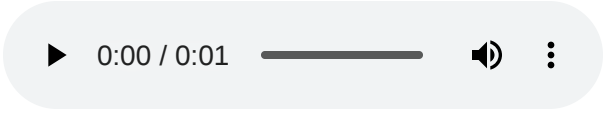


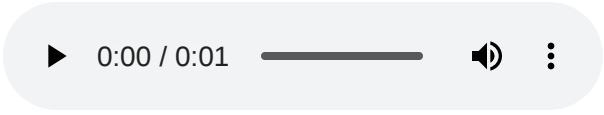



```
----- reference signals ---------
['Fiat_Panda_Drive', 'BMW_318i_Drive', 'Peugeot_307_Drive', 'Audi_A3_Drive']
```










## Analýza signálu v čase

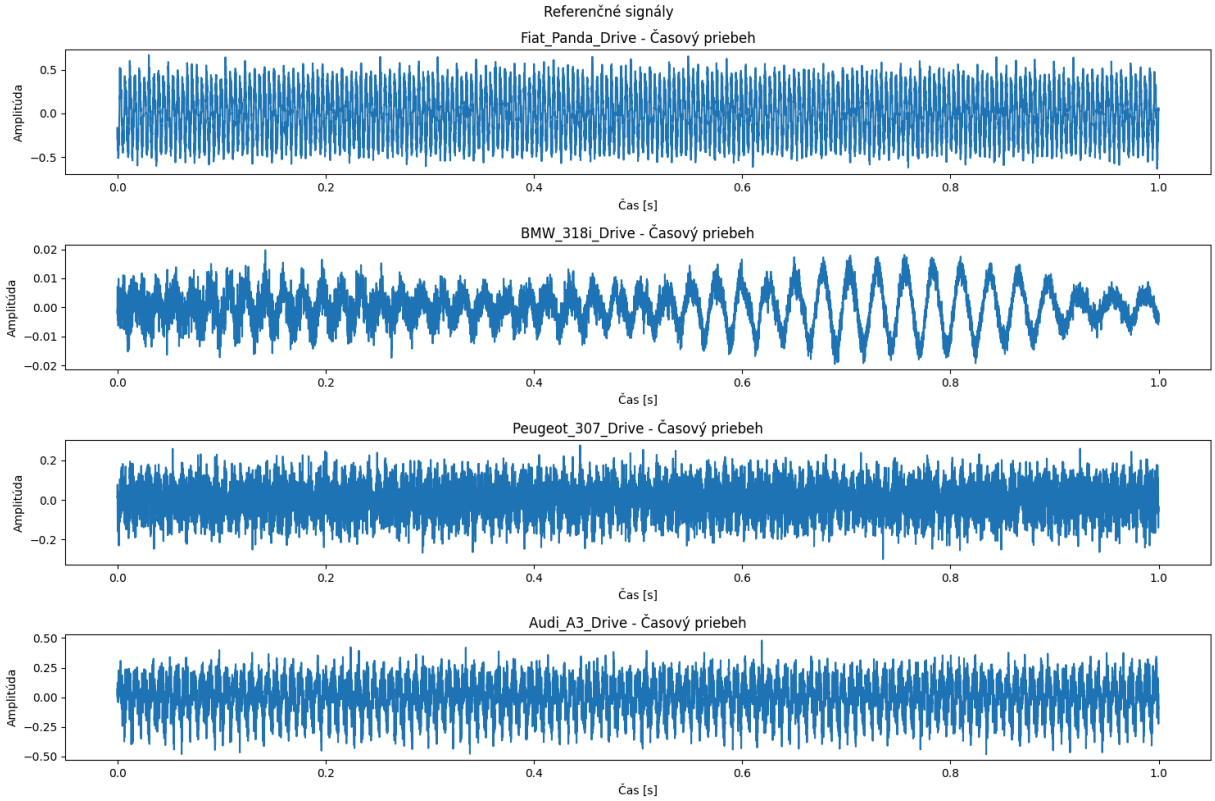
Najskôr si vykreslím časový priebeh signálov, spolu s ich základnými štatistickými parametrami, ktoré mi ale aj tak veľmi nepomohli.

In [4]:

```python
def analyze_signals(signals, Fs, labels, title):
    """
    Analyzuje a vizualizuje základné vlastnosti zvukových signálov

    Parametre:
    signals: list zvukových signálov
    Fs: vzorkovacia frekvencia
    labels: názvy signálov
    title: názov pre graf
    """
    n_signals = len(signals)
    time = np.arange(len(signals[0])) / Fs

    plt.figure(figsize=(15, 10))
```

```
    # Časové priebehy
    for i, (sig, label) in enumerate(zip(signals, labels)):
        plt.subplot(n_signals, 1, i+1)
        plt.plot(time, sig)
        plt.title(f'{label} - Časový priebeh')
        plt.xlabel('Čas [s]')
        plt.ylabel('Amplitúda')

        # Výpis základných štatistík
        print(f"\nŠtatistiky pre {label}:")
        print(f"Maximum: {np.max(sig):.3f}")
        print(f"Minimum: {np.min(sig):.3f}")
        print(f"Stredná hodnota: {np.mean(sig):.3f}")
        print(f"Smerodajná odchýlka: {np.std(sig):.3f}")

    plt.suptitle(title)
    plt.tight_layout()
    plt.show()

# Analýza referenčných signálov
analyze_signals(ref_signals, Fs, ref_labels, "Referenčné signály")

# Analýza testovacích signálov
analyze_signals(test_signals, Fs, test_labels, "Testovacie signály")
```

```
Štatistiky pre Fiat_Panda_Drive:
Maximum: 0.669
Minimum: -0.635
Stredná hodnota: -0.001
Smerodajná odchýlka: 0.291

Štatistiky pre BMW_318i_Drive:
Maximum: 0.020
Minimum: -0.020
Stredná hodnota: 0.000
Smerodajná odchýlka: 0.006

Štatistiky pre Peugeot_307_Drive:
Maximum: 0.277
Minimum: -0.301
Stredná hodnota: -0.000
Smerodajná odchýlka: 0.079

Štatistiky pre Audi_A3_Drive:
Maximum: 0.480
Minimum: -0.482
Stredná hodnota: 0.000
Smerodajná odchýlka: 0.147
```

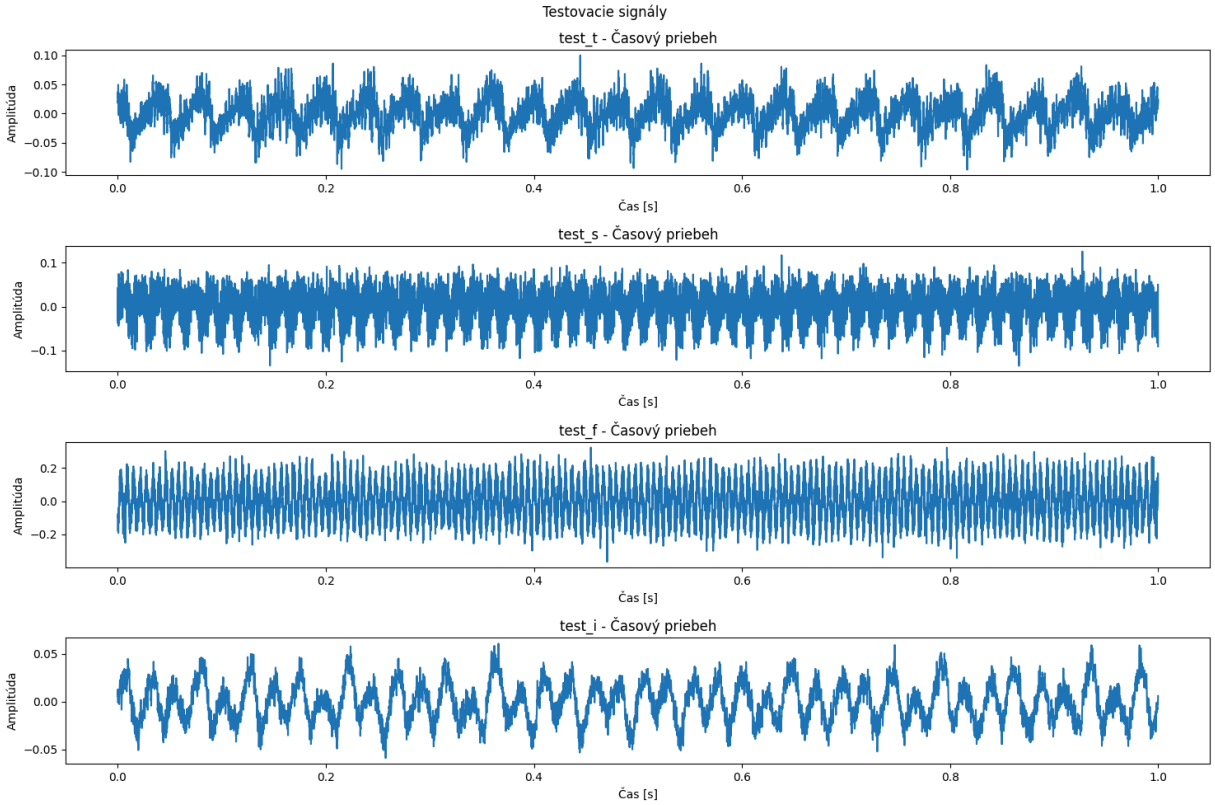



Štatistiky pre test_t:
Maximum: 0.100
Minimum: -0.096
Stredná hodnota: -0.000
Smerodajná odchýlka: 0.027

Štatistiky pre test_s:
Maximum: 0.125
Minimum: -0.135
Stredná hodnota: 0.000
Smerodajná odchýlka: 0.034

Štatistiky pre test_f:
Maximum: 0.322
Minimum: -0.365
Stredná hodnota: -0.000
Smerodajná odchýlka: 0.111

Štatistiky pre test_i:
Maximum: 0.061
Minimum: -0.059
Stredná hodnota: 0.000
Smerodajná odchýlka: 0.020

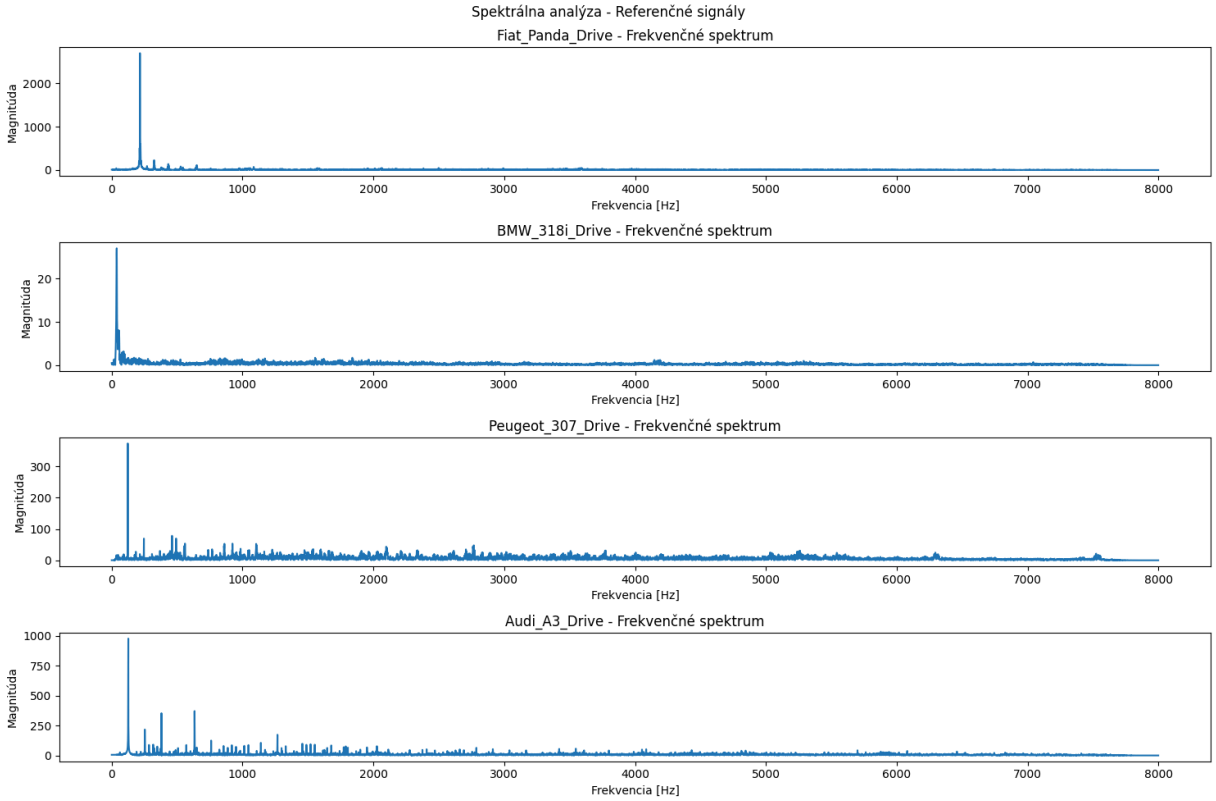


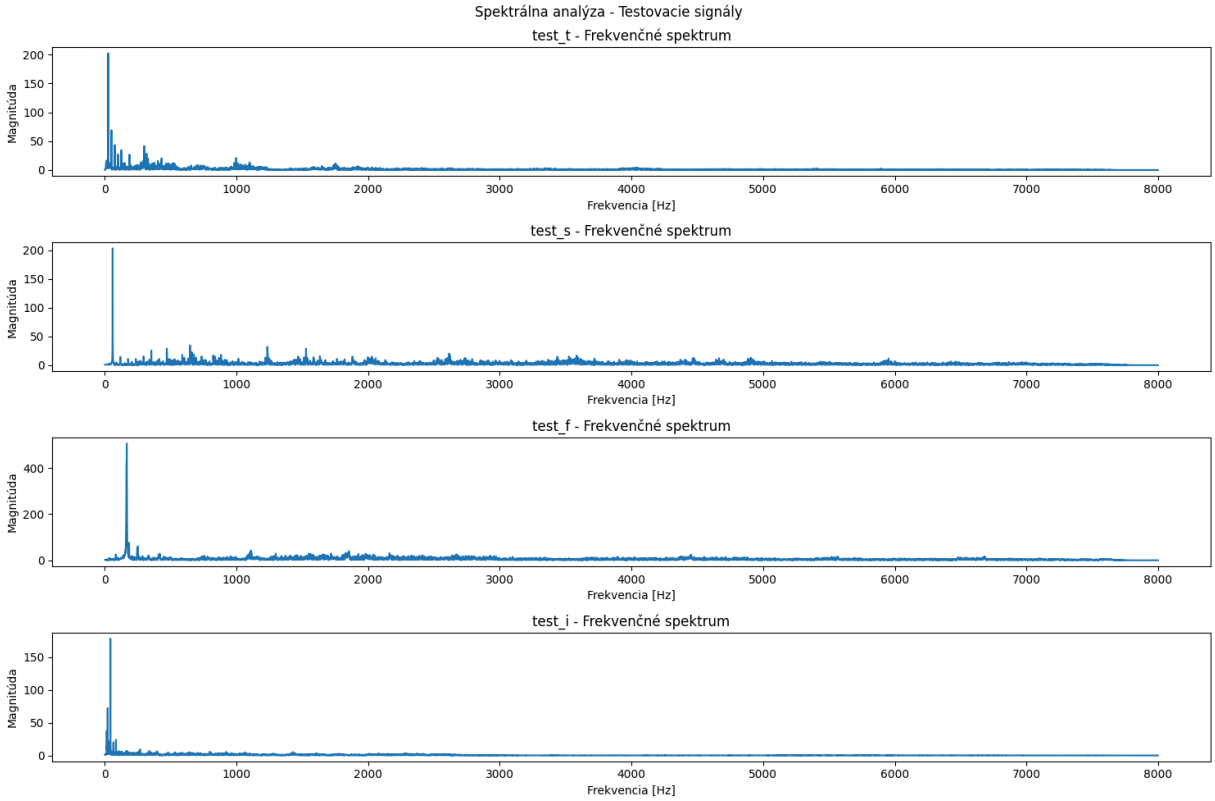

## Spektrálna analýza

Z časového priebehu sa dá pozorovať periodicita signálov. Je pomerne ľahko viditeľná základná frekvencia, teda rýchlost otáčok motora.Pomocou spektrálnej analýzi si teda zobrazíme frekvenčné spektrum.

In [5]:
```python
# Spektrálna analýza
def spectral_analysis(signals, Fs, labels, title):
    """
    Vykoná spektrálnu analýzu signálov

    Parametre:
    signals: list zvukových signálov
    Fs: vzorkovacia frekvencia
    labels: názvy signálov
    title: názov pre graf
    """
    n_signals = len(signals)
    plt.figure(figsize=(15, 10))

    for i, (sig, label) in enumerate(zip(signals, labels)):
        # Výpočet FFT
        freq = np.fft.fftfreq(len(sig), 1/Fs)
        fft_sig = np.fft.fft(sig)

        # Zobrazenie len pozitívnych frekvencií
        positive_freq_idx = freq >= 0

        plt.subplot(n_signals, 1, i+1)
```

```
        plt.plot(freq[positive_freq_idx], np.abs(fft_sig)[positive_freq_idx]
        plt.title(f'{label} - Frekvenčné spektrum')
        plt.xlabel('Frekvencia [Hz]')
        plt.ylabel('Magnitúda')

    plt.suptitle(title)
    plt.tight_layout()
    plt.show()

# Spektrálna analýza referenčných signálov
spectral_analysis(ref_signals, Fs, ref_labels, "Spektrálna analýza - Referer

# Spektrálna analýza testovacích signálov
spectral_analysis(test_signals, Fs, test_labels, "Spektrálna analýza - Testc
```



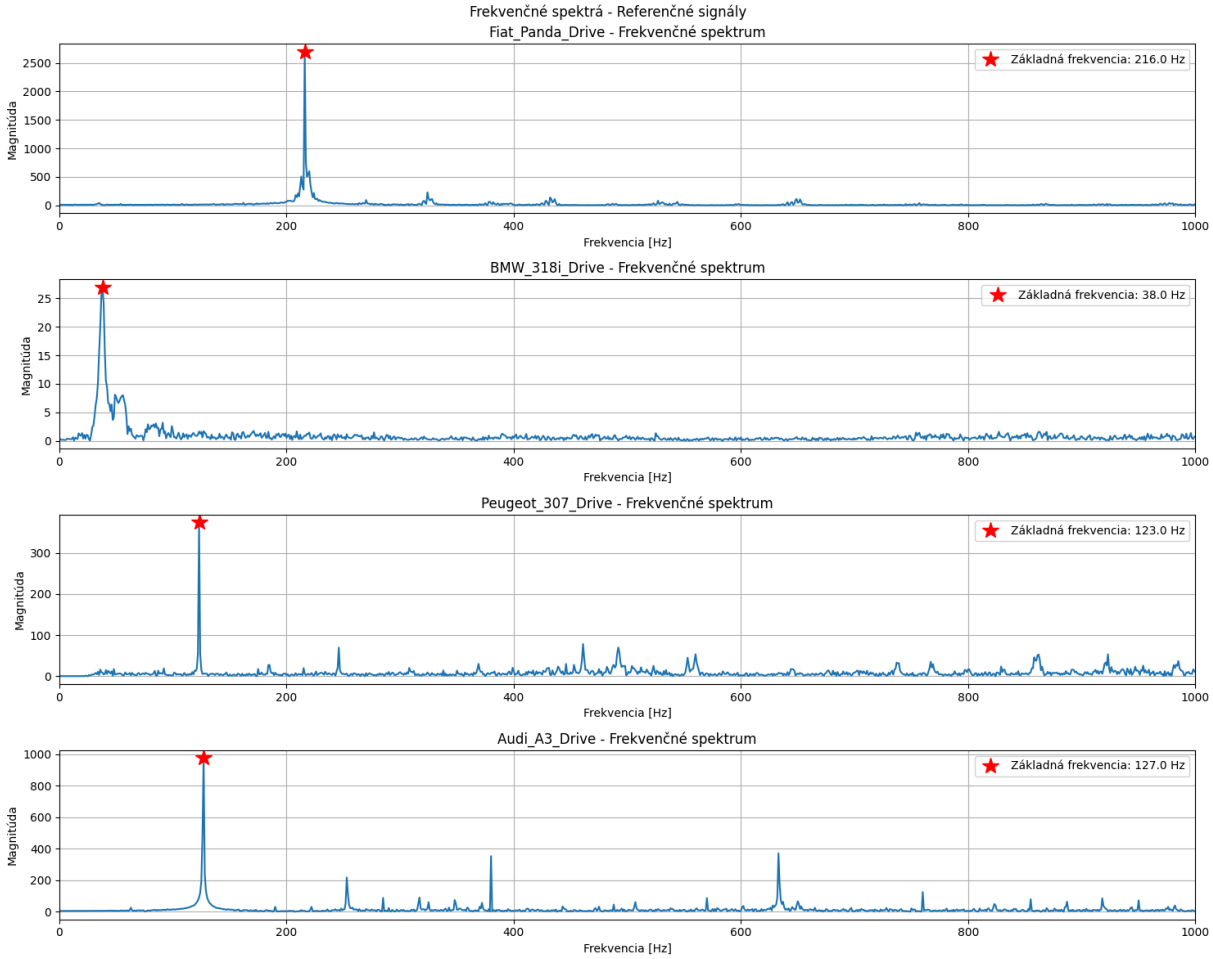


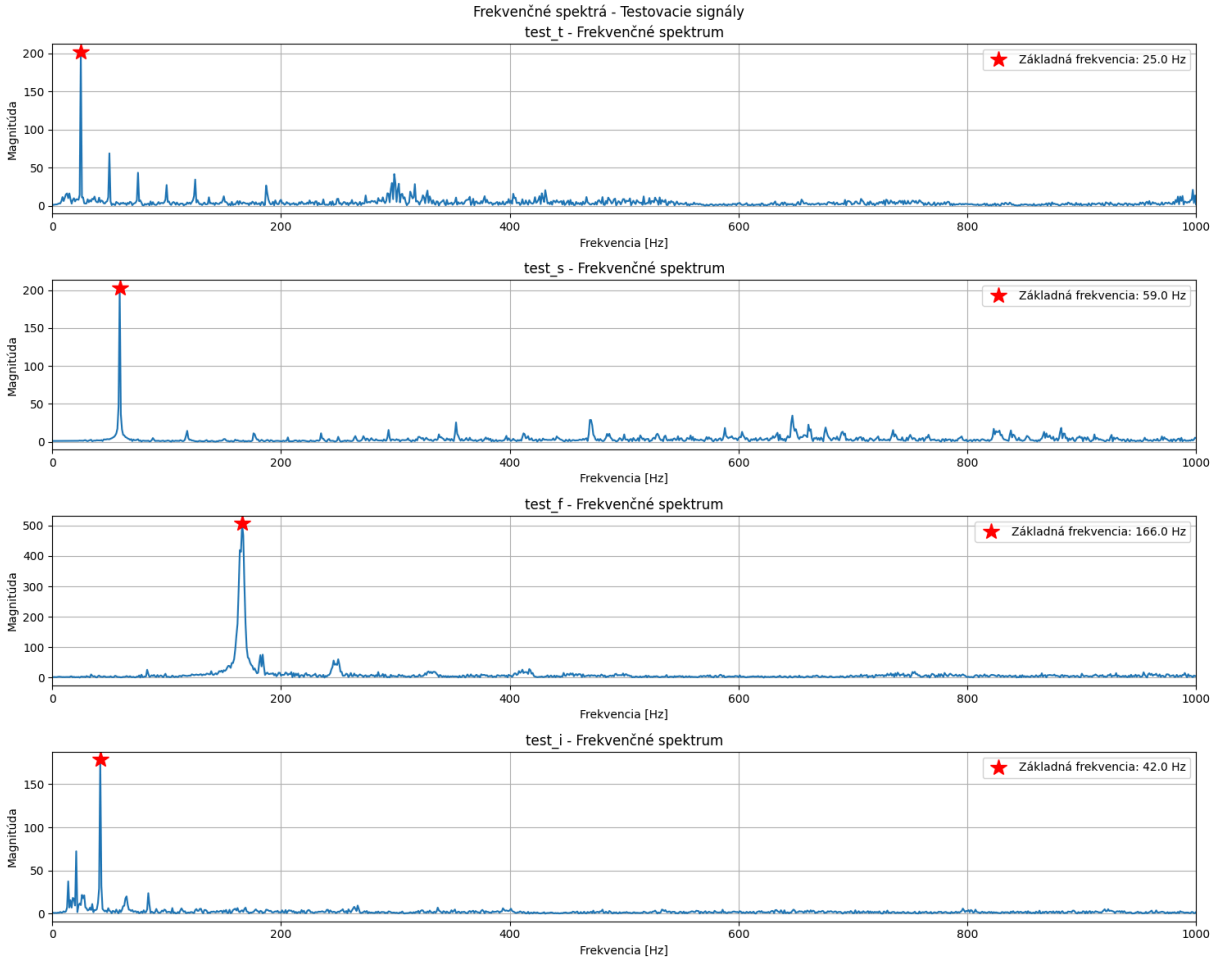

## Hľadanie základných frekvencií

Nájdem a označím si základnú frekvenciu, je dominantná a zo všetkých grafou ju je ľahko vidieť. Vypíšem jej hodnotu. Podľa tejto hodnoty by sa v podstate dali zistiť konkrétne otáčky všetkých motorov.

In [6]:
```
def analyze_fundamental_frequency(signal, fs, freq_range=(20, 500)):
    """
    Analyzuje frekvenčné spektrum a nájde základnú frekvenciu

    Parametre:
    signal: vstupný signál
    fs: vzorkovacia frekvencia
    freq_range: rozsah frekvencií pre hľadanie základnej frekvencie (min, ma

    Návratové hodnoty:
    freqs: všetky frekvencie
    magnitude: magnitudy frekvencií
    fund_freq: základná frekvencia
    """
    # Výpočet FFT
    fft_sig = np.fft.fft(signal)
    freqs = np.fft.fftfreq(len(signal), 1/fs)
    magnitude = np.abs(fft_sig)

    # Obmedzenie na pozitívne frekvencie
    positive_freq_mask = freqs >= 0
    freqs = freqs[positive_freq_mask]
    magnitude = magnitude[positive_freq_mask]
```

```
    # Nájdenie základnej frekvencie v danom rozsahu
    freq_mask = (freqs >= freq_range[0]) & (freqs <= freq_range[1])
    fund_freq = freqs[freq_mask][np.argmax(magnitude[freq_mask])]

    return freqs, magnitude, fund_freq

def plot_frequency_spectra(signals, labels, fs, title):
    """
    Vykreslí frekvenčné spektrá so zvýraznenou základnou frekvenciou
    """
    n_signals = len(signals)
    plt.figure(figsize=(15, 3*n_signals))

    for i, (signal, label) in enumerate(zip(signals, labels)):
        # Analýza frekvenčného spektra
        freqs, magnitude, fund_freq = analyze_fundamental_frequency(signal,

        # Vykreslenie spektra
        plt.subplot(n_signals, 1, i+1)
        plt.plot(freqs, magnitude)

        # Zvýraznenie základnej frekvencie
        fund_idx = np.argmin(np.abs(freqs - fund_freq))
        plt.plot(fund_freq, magnitude[fund_idx], 'r*',
                markersize=15, label=f'Základná frekvencia: {fund_freq:.1f}

        plt.title(f'{label} - Frekvenčné spektrum')
        plt.xlabel('Frekvencia [Hz]')
        plt.ylabel('Magnitúda')
        plt.grid(True)
        plt.legend()

        # Obmedzenie zobrazenia na relevantný rozsah frekvencií
        plt.xlim(0, 1000)  # zobrazíme len prvých 1000 Hz

    plt.suptitle(title)
    plt.tight_layout()
    plt.show()

# Analýza referenčných signálov
plot_frequency_spectra(ref_signals, ref_labels, Fs, "Frekvenčné spektrá - Re

# Analýza testovacích signálov
plot_frequency_spectra(test_signals, test_labels, Fs, "Frekvenčné spektrá -

# Výpis základných frekvencií
print("\nZákladné frekvencie referenčných signálov:")
for signal, label in zip(ref_signals, ref_labels):
    _, _, fund_freq = analyze_fundamental_frequency(signal, Fs)
    print(f"{label}: {fund_freq:.1f} Hz")

print("\nZákladné frekvencie testovacích signálov:")
for signal, label in zip(test_signals, test_labels):
    _, _, fund_freq = analyze_fundamental_frequency(signal, Fs)
    print(f"{label}: {fund_freq:.1f} Hz")
```


Frekvenčné spektrá - Referenčné signály
Fiat_Panda_Drive - Frekvenčné spektrum
Základná frekvencia: 216.0 Hz
Magnitúda
Frekvencia [Hz]
BMW_318i_Drive - Frekvenčné spektrum
Základná frekvencia: 38.0 Hz
Magnitúda
Frekvencia [Hz]
Peugeot_307_Drive - Frekvenčné spektrum
Základná frekvencia: 123.0 Hz
Magnitúda
Frekvencia [Hz]
Audi_A3_Drive - Frekvenčné spektrum
Základná frekvencia: 127.0 Hz
Magnitúda
Frekvencia [Hz]




```
Základné frekvencie referenčných signálov:
Fiat_Panda_Drive: 216.0 Hz
BMW_318i_Drive: 38.0 Hz
Peugeot_307_Drive: 123.0 Hz
Audi_A3_Drive: 127.0 Hz

Základné frekvencie testovacích signálov:
test_t: 25.0 Hz
test_s: 59.0 Hz
test_f: 166.0 Hz
test_i: 42.0 Hz
```

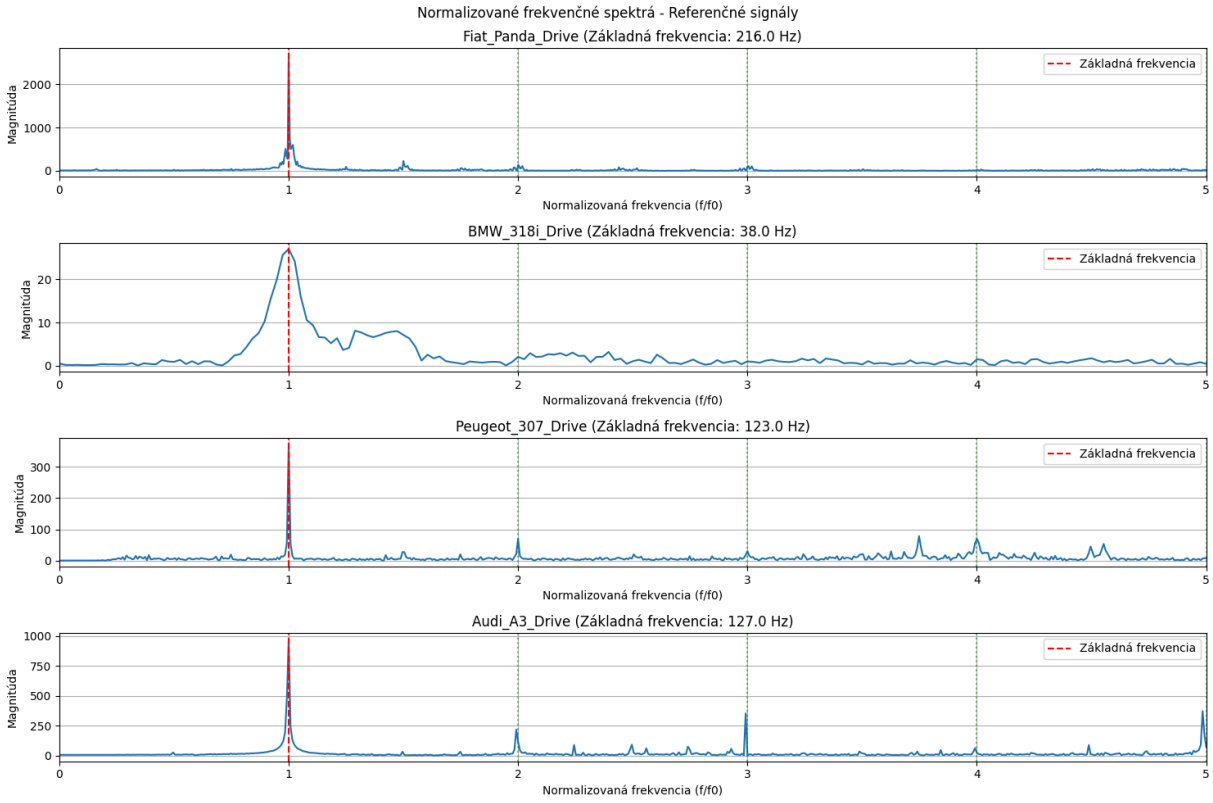
## Normalizácia frekvenčnej osi

Znormalizujem si frekvenčnú os na násobky základnej frekvencie. Frekvenčné spektrum sa bude dať porovnávať medzi nahrávkami.

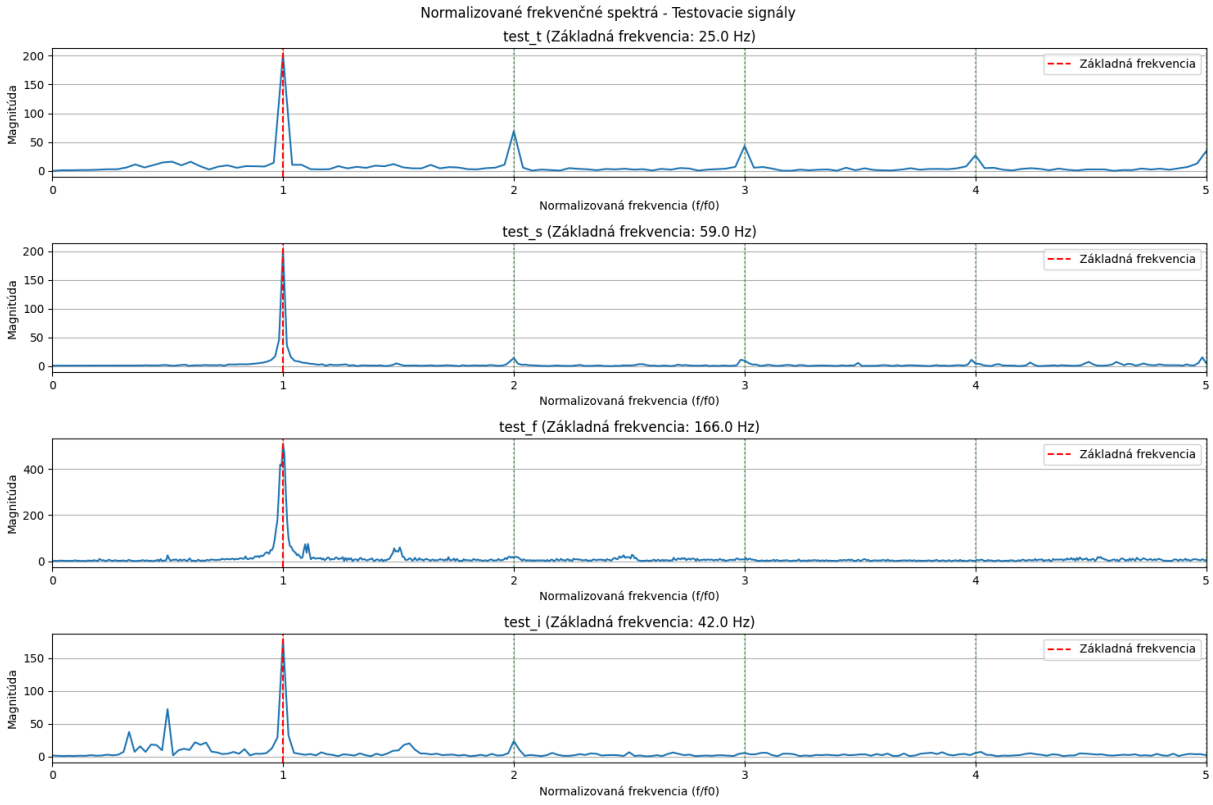
In [7]:
```python
def get_normalized_spectrum(signal, fs):
    """
    Vypočíta a posunie frekvenčné spektrum tak, aby základná frekvencia bola
    """

    # Výpočet FFT
    fft_sig = np.fft.fft(signal)
    freqs = np.fft.fftfreq(len(signal), 1/fs)
    magnitude = np.abs(fft_sig)
```

```
    # Nájdenie základnej frekvencie (v rozsahu 20-500 Hz)
    positive_mask = freqs >= 0
    freq_range_mask = (freqs >= 20) & (freqs <= 500)
    valid_mask = positive_mask & freq_range_mask

    fund_freq = freqs[valid_mask][np.argmax(magnitude[valid_mask])]

    # Vytvorenie normalizovanej frekvenčnej osi
    # Posunieme frekvencie tak, aby základná frekvencia bola na 1
    normalized_freqs = freqs / fund_freq

    return normalized_freqs[positive_mask], magnitude[positive_mask], fund_f

def plot_normalized_spectra(signals, labels, fs, title):
    """
    Vykreslí normalizované frekvenčné spektrá
    """
    plt.figure(figsize=(15, 10))

    for i, (signal, label) in enumerate(zip(signals, labels)):
        # Získanie normalizovaného spektra
        norm_freqs, magnitude, fund_freq = get_normalized_spectrum(signal, f

        plt.subplot(len(signals), 1, i+1)
        plt.plot(norm_freqs, magnitude)
        plt.axvline(x=1, color='r', linestyle='--', label='Základná frekvenc

        # Zvýraznenie harmonických
        for n in range(2, 6):
            plt.axvline(x=n, color='g', linestyle=':', alpha=0.5)

        plt.title(f'{label} (Základná frekvencia: {fund_freq:.1f} Hz)')
        plt.xlabel('Normalizovaná frekvencia (f/f0)')
        plt.ylabel('Magnitúda')
        plt.xlim(0, 5)  # Zobrazíme prvých 5 harmonických
        plt.grid(True)
        plt.legend()

    plt.suptitle(title)
    plt.tight_layout()
    plt.show()

# Analýza referenčných signálov
print("Referenčné signály:")
plot_normalized_spectra(ref_signals, ref_labels, Fs, "Normalizované frekvenč

# Analýza testovacích signálov
print("\nTestovacie signály:")
plot_normalized_spectra(test_signals, test_labels, Fs, "Normalizované frekve
```

```
Referenčné signály:
```




Testovacie signály:




Teraz už z grafov je možné badať podobnosti, prípadne rozdiely medzi jednotlivými signálmi.

# Výsledná analýza spektra a výpočet vzdialeností medzi nahrávkami

Ďalej normalizujem amplitúdu, a vypočítam pomer medzi základnou frekvenciou a jej harmonickými násobkami. Avšak hodnotu frekvencií hramonických násobkov budem brať ako maximálnu hodnotu z malého okolia harmonického násobku aby som obmedzil nepresnosti, napríklad pri nahrávke Audi_A3_Drive sú vrcholy frekvencií v harmonických násobkoch mierne posunuté, čo by mohlo spôsobovať nepresné výsledky. Vypočítam vzdialenosti medzi pomermi, pričom do výpočtu budem brať len prvých 6 harmonických frekvencií okrem základnej, je to z dôvodu toho že aj keď som zobral viac, tak sa už výsledok zásadne nezmenil.

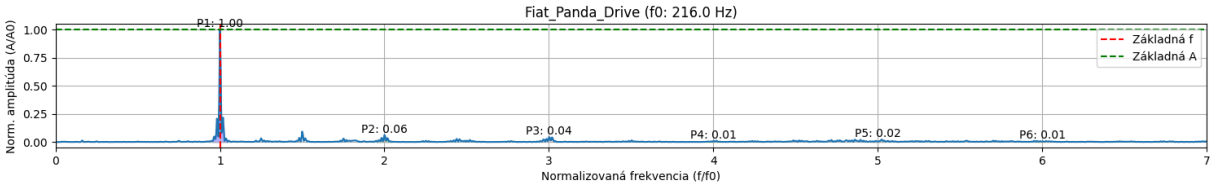
In [10]:
```
def get_normalized_spectrum_with_peak_ratios(signal, fs, window_width=0.05):
    """
    Vykonáva spektrálnu analýzu zvukových signálov založenú na pomeroch
    maximálnych amplitúd v okolí harmonických frekvencií.

    Parametre:
    signal: Časový priebeh zvukového signálu
    fs: Vzorkovacia frekvencia
    window_width: Šírka okna okolo harmonických pre analýzu (ako zlomok harm

    Vracia:
    norm_freqs: Normalizované frekvencie (relatívne k základnej)
    magnitude: Normalizované magnitudové spektrum
    fund_freq: Základná frekvencia
    peak_ratios: Normalizované pomery maximálnych amplitúd harmonických komp
    highlight_masks: Boolovské masky pre vizualizáciu
    """
    # Aplikujeme Hanningovo okno pre zníženie spektrálneho presakovania
    window = np.hanning(len(signal))
    signal = signal * window

    # Vypočítame FFT a získame frekvenčnú os
    fft_sig = np.fft.fft(signal)
    freqs = np.fft.fftfreq(len(signal), 1/fs)
    magnitude = np.abs(fft_sig)

    # Vytvoríme masky pre platné frekvenčné rozsahy
    positive_mask = freqs >= 0
    # Zameriame sa na typický rozsah frekvencií motora (20-500 Hz)
    freq_range_mask = (freqs >= 20) & (freqs <= 500)
    valid_mask = positive_mask & freq_range_mask

    # Odhadneme úroveň šumu pre lepšiu detekciu špičiek
    noise_floor = np.median(magnitude[valid_mask])
    signal_threshold = noise_floor * 3

    # Nájdeme významné špičky nad úrovňou šumu
    peaks = magnitude[valid_mask] > signal_threshold
    potential_freqs = freqs[valid_mask][peaks]
    potential_mags = magnitude[valid_mask][peaks]

    # Identifikujeme základnú frekvenciu z najsilnejšej špičky
    if len(potential_mags) > 0:
        fund_freq_idx = np.argmax(potential_mags)
```

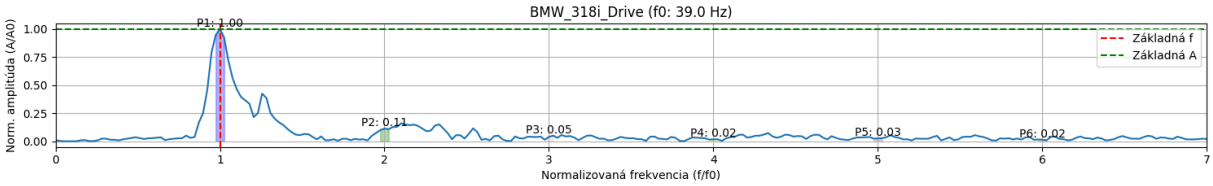
```
        fund_freq = potential_freqs[fund_freq_idx]
        fund_magnitude = potential_mags[fund_freq_idx]
    else:
        max_idx = np.argmax(magnitude[valid_mask])
        fund_freq = freqs[valid_mask][max_idx]
        fund_magnitude = magnitude[valid_mask][max_idx]

    # Normalizujeme amplitúdy vzhľadom na amplitúdu základnej frekvencie
    magnitude = magnitude / fund_magnitude

    # Normalizujeme frekvencie relatívne k základnej
    norm_freqs = freqs[positive_mask] / fund_freq
    magnitude = magnitude[positive_mask]

    # Analyzujeme harmonické komponenty pomocou maximálnych amplitúd
    peak_ratios = []
    n_harmonics = 6
    highlight_masks = []

    for n in range(1, n_harmonics + 1):
        # Vytvoríme úzke okno okolo každej harmonickej frekvencie
        window_mask = (norm_freqs >= n - window_width) & \
                      (norm_freqs <= n + window_width)
        highlight_masks.append(window_mask)

        # Použijeme maximálnu hodnotu v okne
        if np.any(window_mask):
            peak_value = np.max(magnitude[window_mask])
            peak_ratios.append(peak_value)
        else:
            peak_ratios.append(noise_floor / fund_magnitude)  # Normalizovar

    return norm_freqs, magnitude, fund_freq, np.array(peak_ratios), highligh

def plot_normalized_spectra(signals, labels, fs, title, window_width=0.05):
    """
    Vytvára vizualizáciu normalizovaných spektier so zvýraznenými harmonický
    """
    plt.figure(figsize=(15, 10))

    for i, (signal, label) in enumerate(zip(signals, labels)):
        norm_freqs, magnitude, fund_freq, peak_ratios, highlight_masks = \
            get_normalized_spectrum_with_peak_ratios(signal, fs, window_widt

        plt.subplot(len(signals), 1, i+1)
        plt.plot(norm_freqs, magnitude)

        # Zvýrazníme oblasti harmonických frekvencií a pridáme označenie pom
        colors = ['blue', 'green', 'pink', 'yellow', 'gray', 'orange']
        for n, (mask, ratio, color) in enumerate(zip(highlight_masks, peak_r
            plt.fill_between(norm_freqs[mask], magnitude[mask], alpha=0.3, c
            center_freq = n + 1
            max_mag = np.max(magnitude[mask]) if np.any(mask) else 0
            plt.text(center_freq, max_mag, f'P{n+1}: {ratio:.2f}',
                     horizontalalignment='center', verticalalignment='bottom'
```

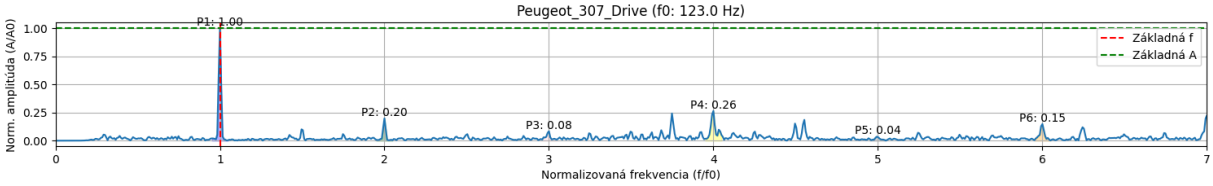
```
        plt.axvline(x=1, color='r', linestyle='--', label='Základná f')
        plt.axhline(y=1, color='g', linestyle='--', label='Základná A')

        plt.title(f'{label} (f0: {fund_freq:.1f} Hz)')
        plt.xlabel('Normalizovaná frekvencia (f/f0)')
        plt.ylabel('Norm. amplitúda (A/A0)')
        plt.xlim(0, 7)
        plt.grid(True)
        plt.legend()

    plt.suptitle(f'{title}\nPomery maximálnych amplitúd (±{window_width * 10
    plt.tight_layout()
    plt.show()

def match_test_to_reference(ref_signals, test_signals, ref_labels, test_labe
    """
    Priraďuje testovacie signály k referenčným na základe porovnania
    pomerov maximálnych amplitúd harmonických frekvencií.
    Pre každý referenčný signál nájde testovací signál s najmenšou vzdialenc
    """
    # Získame pomery špičiek pre všetky signály
    ref_ratios = []
    test_ratios = []

    for signal in ref_signals:
        _, _, _, ratios, _ = get_normalized_spectrum_with_peak_ratios(signal
        ref_ratios.append(ratios)

    for signal in test_signals:
        _, _, _, ratios, _ = get_normalized_spectrum_with_peak_ratios(signal
        test_ratios.append(ratios)

    ref_ratios = np.array(ref_ratios)
    test_ratios = np.array(test_ratios)

    # Vypočítame euklidovskú vzdialenosť medzi pomermi
    all_distances = np.zeros((len(test_signals), len(ref_signals)))
    for i, test_ratio in enumerate(test_ratios):
        for j, ref_ratio in enumerate(ref_ratios):
            distance = np.sqrt(np.sum((test_ratio - ref_ratio) ** 2))
            all_distances[i, j] = distance

    # Výpis matice vzdialeností
    print("\nVzdialenosti medzi pomermi maximálnych amplitúd:")
    print("-" * 50)
    print(f"{'':15}", end="")
    for ref_label in ref_labels:
        print(f"{ref_label:15} ", end="")
    print()

    for i, test_label in enumerate(test_labels):
        print(f"{test_label:15}", end="")
        for j in range(len(ref_labels)):
            print(f"{all_distances[i,j]:15.3f}", end="")
        print()
```

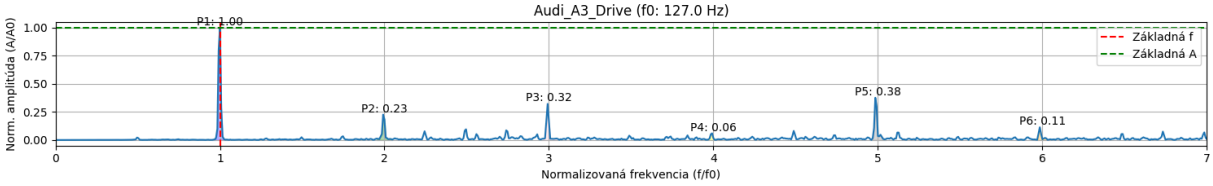
```
    # Pre každý referenčný signál nájdeme najbližší testovací signál
    final_matches = {}

    for ref_idx, ref_label in enumerate(ref_labels):
        # Nájdeme najbližší testovací signál pre tento referenčný signál
        distances_to_ref = all_distances[:, ref_idx]
        closest_test_idx = np.argmin(distances_to_ref)
        test_label = test_labels[closest_test_idx]
        min_distance = distances_to_ref[closest_test_idx]

        final_matches[test_label] = (ref_label, min_distance)

    # Výpis výsledkov
    print("\nNájdené zhody:")
    print("-" * 50)
    for test_label, (ref_label, distance) in final_matches.items():
        print(f"{test_label} -> {ref_label} (vzdialenosť: {distance:.3f})")

    # Nájdeme nepriradené testovacie signály
    matched_test_labels = set(final_matches.keys())
    unmatched_test_labels = set(test_labels) - matched_test_labels

    if unmatched_test_labels:
        print("\nNepriradené testovacie signály:")
        for test_label in unmatched_test_labels:
            print(f"- {test_label}")

    return final_matches

def analyze_car_sounds(ref_signals, test_signals, ref_labels, test_labels, f
    """
    Vykonáva kompletnú analýzu zvukov automobilových motorov.
    """
    plot_normalized_spectra(ref_signals, ref_labels, fs,
                            "Analýza referenčných signálov")

    plot_normalized_spectra(test_signals, test_labels, fs,
                            "Analýza testovacích signálov")

    matches = match_test_to_reference(
        ref_signals, test_signals, ref_labels, test_labels, fs)

    return matches
# Spustíme analýzu
matches = analyze_car_sounds(ref_signals, test_signals, ref_labels, test_lab
```

Analýza referenčných signálov
Pomery maximálnych amplitúd (±5.0% okno)

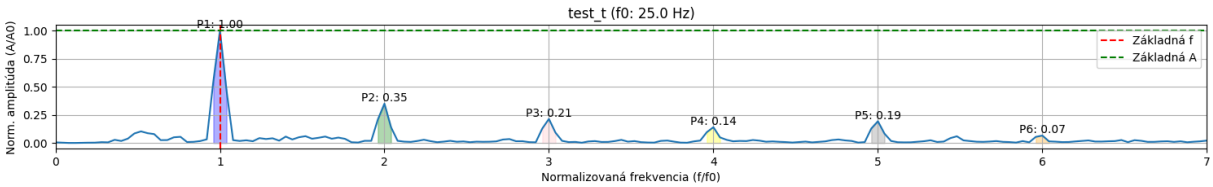


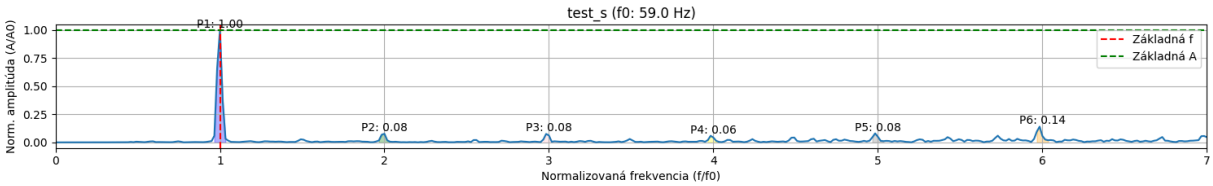



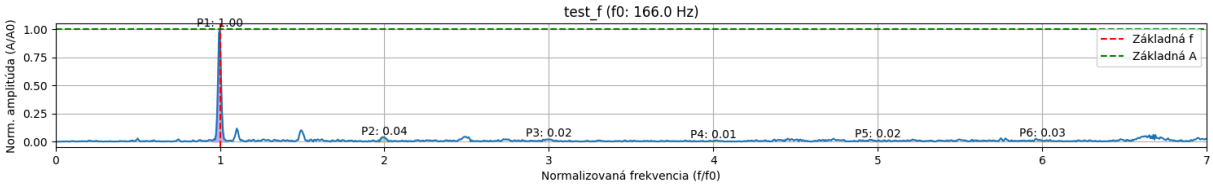



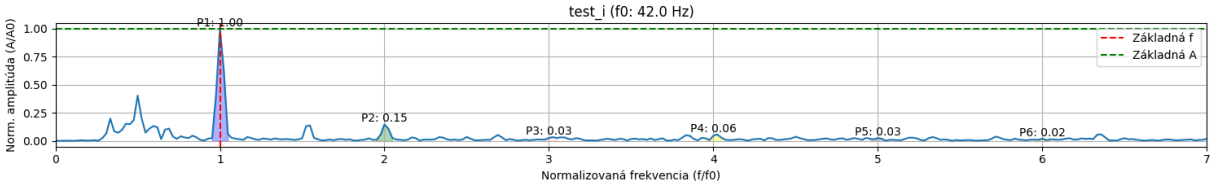




Analýza testovacích signálov
Pomery maximálnych amplitúd (±5.0% okno)



```
Vzdialenosti medzi pomermi maximálnych amplitúd:
--------------------------------------------------
               Fiat_Panda_Drive BMW_318i_Drive  Peugeot_307_Drive Audi_A3_Dr
ive
test_t                    0.400          0.356          0.295          0.263
test_s                    0.153          0.146          0.244          0.414
test_f                    0.036          0.080          0.329          0.513
test_i                    0.098          0.054          0.253          0.468

Nájdené zhody:
--------------------------------------------------
test_f -> Fiat_Panda_Drive (vzdialenosť: 0.036)
test_i -> BMW_318i_Drive (vzdialenosť: 0.054)
test_s -> Peugeot_307_Drive (vzdialenosť: 0.244)
test_t -> Audi_A3_Drive (vzdialenosť: 0.263)
```

## 4. Výsledky

Ku každému autu som dokázal násť neznámu nahrávku s najmenšou vzdialenosťou. Pri Peugeote sú výsledky najmenej jednoznačné. Ale všetky nahrávky zrejme patria k daným autám. Ďalej môžeme vidieť, že Fiat aj BMW maju veľmi malú vzdialenosť k svojim priradeným nahrávkam, pričom Peugeot a Audi majú výrazne vyššiu túto vzdialenosť. To môže byť spôsobené výraznejším rozdielom medzi otáčkami motorov oboch párov nahrávok Peugeotu a Audi, prípadne rôznym šumom v pozadí. Hoci za ideálnych okolností by mali byť tieto vplivy odfiltrované.

## 5. Záver

Implementovaný systém by mal demonštrovať schopnosť rozpoznávať a priraďovať zvuky automobilových motorov na základe ich spektrálnych charakteristík, konkrétne porovnávaním pomerov základnej frekvencie k jej harmonickým frekvenciám.