	<pre>pocet sekund ime_len = data.size / fs hodnota pro zapis wavfile souboru ata_max = data.max() vypis informaci rint(f"Pocet vzorku: {data.size}") rint(f"Delka signalu: {time_len}") rint(f"Max: {data.max()}") rint(f"Min: {data.min()}") vykresleni grafu ime = np.arange(data.size) / fs lt.figure(figsize=(12, 6)) lt.plot(time_data)</pre>
	<pre>lt.figure(figsize=(12, 6)) lt.plot(time, data) pojemnovani os lt.gca().set_xlabel('t [s]') lt.gca().set_title('Zvukový signál') lt.tight_layout() cet vzorku: 71066 lka signalu: 4.441625 x: 2363 n: -2316 Zvukový signál</pre>
,	ruhý úkol druhém úkolu se ustředňuje signál, normuje se a rozděluje se do rámců. Střední hodnota se vypočítá jako aritmet iměr jednotlivých hodnot. Jednotlivé rámce se vytváří pomocí jednoduchého cyklu, během kterého se vymezí notlivé rámce (frames) do pole a vkládají se do seznmau rámců. ustredneni signalu ean = sum(data) / data.size ata = data - mean normalizace
	<pre>tvorba ramcu of_frames = [] = 0 hile True: # posledni ramec doplni nulami if i + 1024 > data.size: last_frame = data[i:data.size-1] + np.zeros(data.size-i-1) lof_frames.append(np.array(last_frame)) break lof_frames.append(np.array(data[i:i+1024])) i += 512 vybrany ramec rame = lof_frames[20] vykresleni grafu lt.figure(figsize=(12, 6)) lt.plot(np.arrange(1024) / fs, lof_frames[20])</pre>
-	etí úkol tí úloha spočívá v pouažítí DFT (diskrétní Fourierova transformace). Nejprve se pomocí příkazu scipy.linalg.df tvoří matice o velikosti 1024x1024, který se poté vynásobí s vektorem rámce. Výsledné hodnoty se hodí do absoludnoty. rom scipy.linalg import dft = len(frame) vypocet pomoci matice a ulození pole do promenne
	<pre>vypocet dft pomoci iterace for k in range(N): base = np.exp([-2j * np.pi / N * k * n</pre>
I	Dektrogram strou tohoto úkolu je příkaz scipy.signal.spectrogram, který vrací všechny důležitá data k zobrazení spektrognálu. rom scipy.signal import spectrogram spektrogram za pouziti ramcu o 1024 vzorcich s presahy 512, t, sgr = spectrogram(data, fs, nperseg=1024, noverlap=512)
	<pre>uprava jednotlivych koeficientu DFT na logaritmickou stupnici gr_log = 10 * np.log10(sgr+1e-20) vykresleni grafu lt.figure(figsize=(12, 6)) lt.pcolormesh(t, f, sgr_log) pojemnovani os lt.gca().set_title('Spektrogram originálního signálu') lt.gca().set_xlabel('Čas [s]') lt.gca().set_ylabel('Frekvence [Hz]') bar = plt.colorbar() bar.set_label('Spektralní hustota výkonu [dB]', rotation=270, labelpad=15) lt.tight_layout()</pre>
	Spektrogram originálního signálu 60 80 100 120 120 120 120
I	rčení rušivých signálů kvence rušivých signálů můžeme určit z grafu Segment signálu , nebo ze spektrogramu. Já to určil pomocí kniholakce scipy. signal. find_peaks , který hledá ostré hrany a vrací indexy hodnot v seznamu, kde se hrany nacház
97	<pre>seznam jednotlivych frekvenci rusivych signalu os_freq = [] seznam indexu rusivych frekvenci v ramci pouziti v dalsich ukolu os_indices = [] pect = abs(matrix @ lof_frames[0]) indexy rusivych frekvenci v signalu hleda ostré hrany v segmentech po 10 eaks, _ = find_peaks(spect[0:len(spect)//2], distance=10) ulozeni hodnot s hodnotami ostrych hran vetsich nez 1 os_indices = list(filter(lambda x: spect[x] > 1., peaks)) ulozeni nalezenych frekvenci cos_freq = [(i+1) * 671.875 for i in range(4)] os_freq = [i * fs / 1024 for i in cos_indices]</pre>
1	presny vypis rusivych frekvenci rint(f"Rušivé frekvence v Hz: {cos_freq}") šivé frekvence v Hz: [671.875, 1343.75, 2015.625, 2703.125] enerování signálu ní vygenerujeme signál, který nám rušil původní signál. Sečteme 4 kosíny o frekvencích uložených v cos_freq a prišeme do souboru 4cos.wav. hlasitost plume = 1 vytvoreni pole hodnot x-sove osy pof_samples = np.arange(data.size) / fs
	<pre>vytvoreni souctu 4 kosin utput_cos = sum([volume * np.cos(2 * np.pi * f * nof_samples)</pre>
	<pre>lt.gca().set_xlabel('Čas [s]') lt.gca().set_ylabel('Frekvence [Hz]') bar = plt.colorbar() bar.set_label('Spektralni hustota výkonu [dB]', rotation=270, labelpad=15) lt.tight_layout() generovani souboru avfile.write(get_audio_path('4cos.wav'), fs, output_cos.astype(np.int16)) Spektrogram rušivého signálu - 50 - 25</pre>
	5000 - 50
	sticí filtr adání jsou 3 možnosti, jak vytvořit filtry: výrova filtru v z-rovině návrh filtru ze spektra návrh 4 pásmových zádrží usil jsem vytvořit všechny 3 filtry. Někde jsem byl úspěšnější, někde méně. vroba filtru v z rovině
,	iprve jsem si vypočítal nulové body sedící na jednotkové kružnici v komplexní rovině. Toho docílíme dosazením do orce. $n_k = e^{j\omega_k}$ $n_k = e^{j2\pi\frac{f_k}{F_s}}$ sé přidáme jejich komplexně sdružené hodnoty a převedeme na koeficienty. Implusní odezvu vypočítáme pomocí nvoluce jednotkového impulsu a našeho filtru. $ \text{rom scipy.signal import } \text{lfilter} $ $ \text{vypocet jednotlivych omega_k ze vzorce 2 * pi * f_k / Fs} $
	<pre>wypotet jeanutivy(n omeya_k 2e v2v pi * 1 - 1 - k / rs mega_k = list(map(lambda f: 2 * np.pi * f / fs, cos_freq)) vypocet n_k ze vzorce n_k = e^(j*omega_k) _k = [np.exp(lj * omega) for omega in omega_k] pridani komplexne sdruzenych nulovych bodu a hledani koeficientu filtru _k = n_k + [np.conj(n) for n in n_k] , a = np.poly(n_k), [1, *np.zeros(len(n_k))] jednotkovy impuls a impulsni odezva of_imp = 9 mp = [1, *np.zeros(nof_imp-1)] = lfilter(b, a, imp) vypis keoficientu rint(f'b: {b}') rint(f'a: {a}') lt.figure(figsize=(10, 5)) lt.stem(np.arange(len(h)), h) lt.stem(np.arange(len(h)), h) lt.stem(lp.arange(len(h)), h) lt.ttight_layout()</pre>
	[1.
	uly a póly ly a póly vypočítáme pomocí příkazu scipy. signal. ft2zpk a body zobrazíme na jednotkové kružnici.
02 [03	<pre>rom scipy.signal import tf2zpk zjisteni nul a polu tez mozno pouzit z = np.roots(b) , p, _ = tf2zpk(b, a) lt.figure(figsize=(7, 7)) jednotkova kruznice ng = np.linspace(0, 2*np.pi, 100) lt.plot(np.cos(ang), np.sin(ang)) nuly, poly lt.scatter(np.real(z), np.imag(z), marker='o', facecolors='none',</pre>
	<pre>lt.grid(alpha=0.5, linestyle='') lt.legend(loc='upper left') lt.tight_layout() 100</pre>
	0.00 -0.25 -0.50 -1.00 -1.00 -0.75 -0.50 -0.25 0.00 0.25 0.50 0.75 1.00
ı	ekvenční charakteristika kvenční charakteristiku vygenerujeme pomocí příkazu scipy.signal.freqz . Ten nám vygeneruje magnitudu a nument $H(e^{j\omega})$. rom scipy.signal import freqz $frekvencni\ charakteristika \ ,\ H=freqz(b,\ a)$
	<pre>vykresleni grafu ig, ax = plt.subplots(1, 2, figsize=(12, 4)) x[0].plot(w / 2 / np.pi * fs, np.abs(H)) x[0].set_xlabel('Frekvence [Hz]') x[0].set_title('Modul frekvenční charakteristiky \$ H(e^{j\omega}) \$') x[1].plot(w / 2 / np.pi * fs, np.angle(H)) x[1].set_xlabel('Frekvence [Hz]') x[1].set_title('Argument frekvenční charakteristiky' \</pre>
	Modul frekvenční charakteristiky H(e ^{jω}) Argument frekvenční charakteristikyarg H(e ^{jω}) Argument frekvenční charakteristikyarg H(e ^{jω})
	trace ní již máme všechno k tomu, abychom dokázali originální signál vyfiltrovat. Použijeme na to příkaz ipy.signal.lfilter, který provede konvoluci impulsní odezvy na originální signál. Nasledně provedu normaliza filtrovaného signálu pomocí dělením maximem signálu v absolutní hodnotě a vynásobením int16_max hodnotou p sílení zvuku. Zvuk -1 až 1 totiž nejde ve formátu int16 slyšet.
06	<pre>provedeni filtrace iltered_z = lfilter(b, a, data) vykresleni spektrogramu , t, sfgr = spectrogram(filtered_z, fs, nperseg=1024, noverlap=512) fgr_log = 10 * np.log10(sfgr+le-20) lt.figure(figsize=(15, 5)) lt.pcolormesh(t, f, sfgr_log) pojemovani os lt.gca().set_title('Spektrogram vyfiltrovaného signálu') lt.gca().set_xlabel('Čas [s]') lt.gca().set_ylabel('Frekvence [Hz]')</pre>
	<pre>bar = plt.colorbar() bar.set_label('Spektralni hustota výkonu [dB]', rotation=270, labelpad=15) normalizace signalu iltered_max = filtered_z.max() iltered_z = filtered_z / filtered_max zvetseni hlasitosti iltered_z = np.iinfo(np.int16).max * filtered_z lt.tight_layout() vytvoreni vyfiltrovaneho souboru avfile.write(get_audio_path(f'clean_z.wav'), fs, filtered_z.astype(np.int16))</pre>
	Spektrogram vyfiltrovaného signálu Spektrogram vyfiltrovaného sig
; ;	vsledek filtrace spektrogramu vyfiltrovaného signálu můžeme pozorovat, že rušivé frekvence se nám podařili odstranit. Probléme filtrace velice zkreslila původní signál. V porovnání s originálním signálem filtr "zeslabil" frekvence pod 3 kHz (tmarva) a "zesílil" frekvence nad 3 kHz (světlá barva). světlení najdeme v modulu frekvenční charakteristiky. Z grafu frekvenční charakteristiky můžeme vyčíst, že se jed
	rní propust. To znamená, že nízké frekvence jsou utlumovány (do ± 3.2 kHz). Body nad 3.2 kHz nabývají hodnotam což znamená, že původní signál zesiluje. Ávrh filtru ze spektra ní zkusíme vytvořit druhý filtr a to ze spektra. Nejprve si vygenerujeme frekvenční charakteristiku našeho filtru, kt mezí 1024 vzorků bude obsahovat všude hodnotu 1 . Na indexech frekvencí rušivého signálu nastavíme hodnoty ržeme buďto použít náš seznam cos_indices , nebo indexy spočítat matematicky: $index_k = \left\lfloor 1024 \cdot \frac{f_k}{Fs} \right\rfloor$
07	<pre>vypln frekvencni charakteristiky jednickami krome indexu s rusivymi frekvencemi = list(np.ones(513)) or i in cos_indices: H[i] = 0 pripojeni druhe poloviny frekvencni charakteristiky = H + list(np.flip(H[1:512], 0)) provedena inverzni FFT a posun prvniho indedu doprostred navrat jen realne casti et_val = np.real(np.fft.fftshift(np.fft.ifft(H))) , a = ret_val, [1, *np.zeros(len(ret_val)-1)] generovani jednotkoveho impulsu</pre>
	<pre>a zobrazeni impulsni odezvy of_imp = 1024 mp = [1, *np.zeros(nof_imp-1)]</pre>
	o zóo 4óo 6óo 8óo 1óo uly a póly ejně jako u výroby filtru v z-rovině, proces výpočtu je tu zde stejný. zjisteni nul a polu tez mozno pouzit z = np.roots(b)
	<pre>p, p, = tf2zpk(b, a) lt.figure(figsize=(7, 7)) jednotkova kruznice ng = np.linspace(0, 2*np.pi, 100) lt.plot(np.cos(ang), np.sin(ang)) nuly, poly lt.scatter(np.real(z), np.imag(z), marker='o', facecolors='none',</pre>
	1.00
	-0.75 -1.00 -1.00 -0.75 -0.50 -0.25 0.00 0.25 0.50 0.75 1.00 Realná složka R{z}
	0.92 0.90 0.90
	0.90 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	0.90 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

