Protokol
ISS-Projekt

Šimon Kadnár(xkadna00)

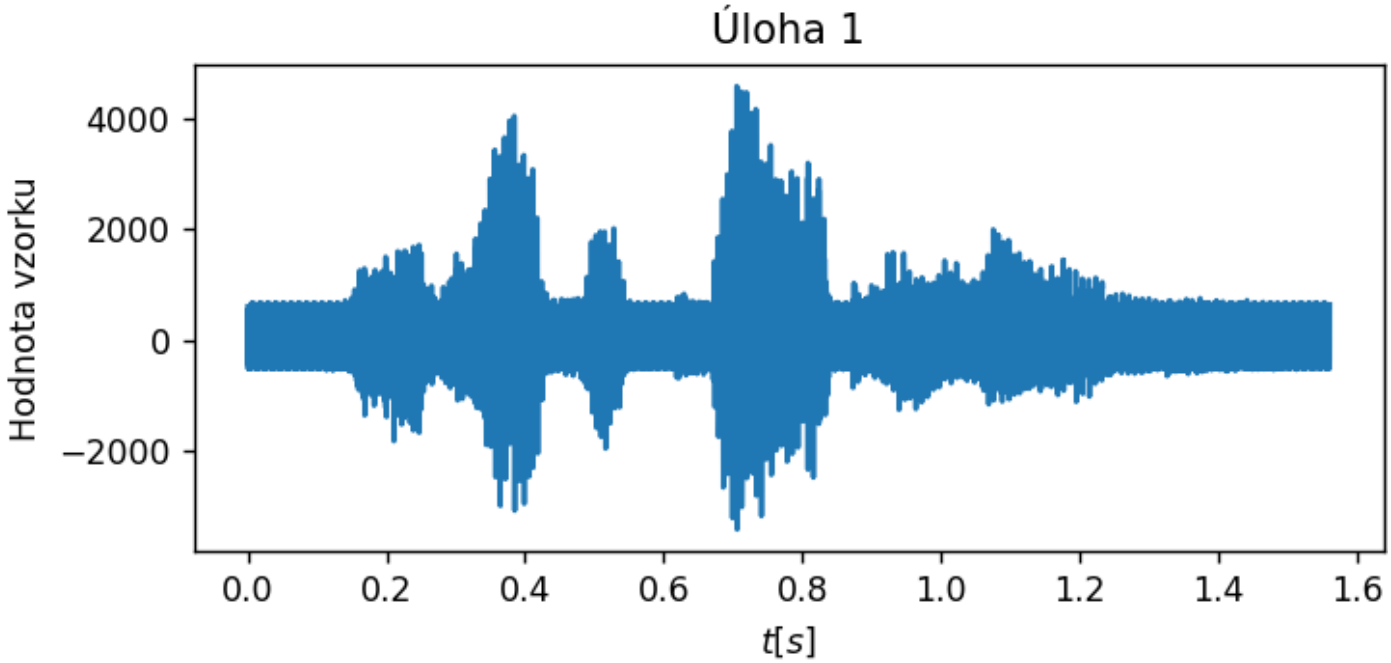
Úloha 1:

počet vzorkov : 24986

dĺžka v sekundách: $\frac{\text{počet vzorkov}}{\text{vzorkovacia frekvencia}}$ = 1.561625s

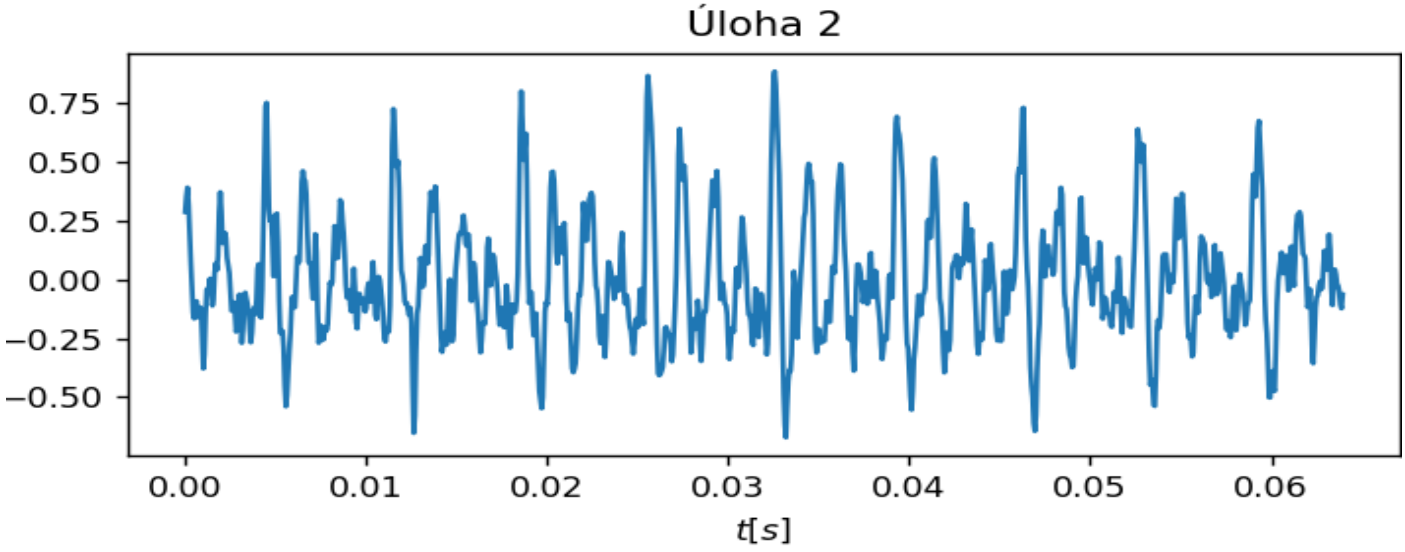
minimálna hodnota vzorku: -3413

maximálna hodnota vzorku: 4574



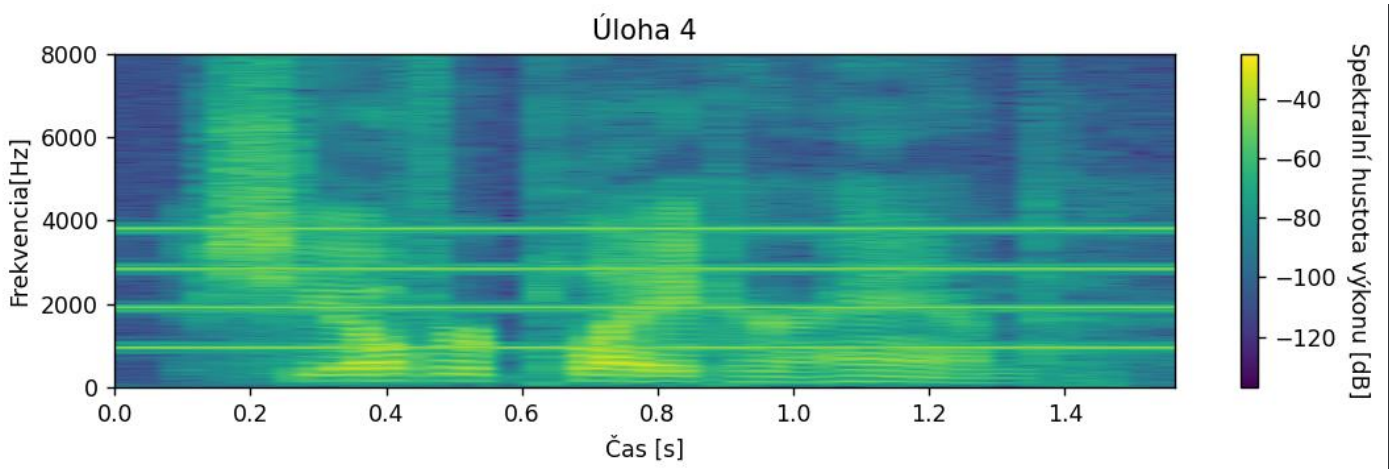
Úloha 2:

Vytvoril som si si funkciu “normaliz” ktorá zahŕňa ustrednenie a aj normalizáciu. Signál som ustrednil a normalizoval pomocou tejto funkcie. Signál som rozdelil na rámce po 1024 vzorkách s prekrytím 512. Následne som si ručne prešiel všetky rámce aby som našiel “pekný” rámec (v mojom prípade to bol rámec č.11).



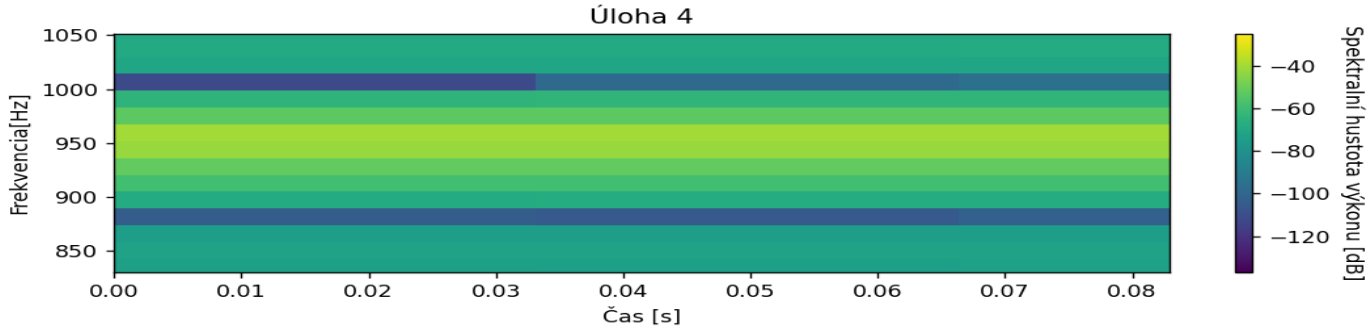
Úloha 4:

Vstupné dáta som opäť ustrednil a normalizoval. Následne som použil funkciu spectrogram ktorá má DFT v sebe obsiahnuté. Nakoniec som použil vzorec $P[k] = 10 \log10 |X[k]|$ a nechal vykresliť graf.



Úloha 5:

Rušivé frekvencie som zistil zo spectogramu. Prvá rušivá komponenta mala frekvenciu 950hz.

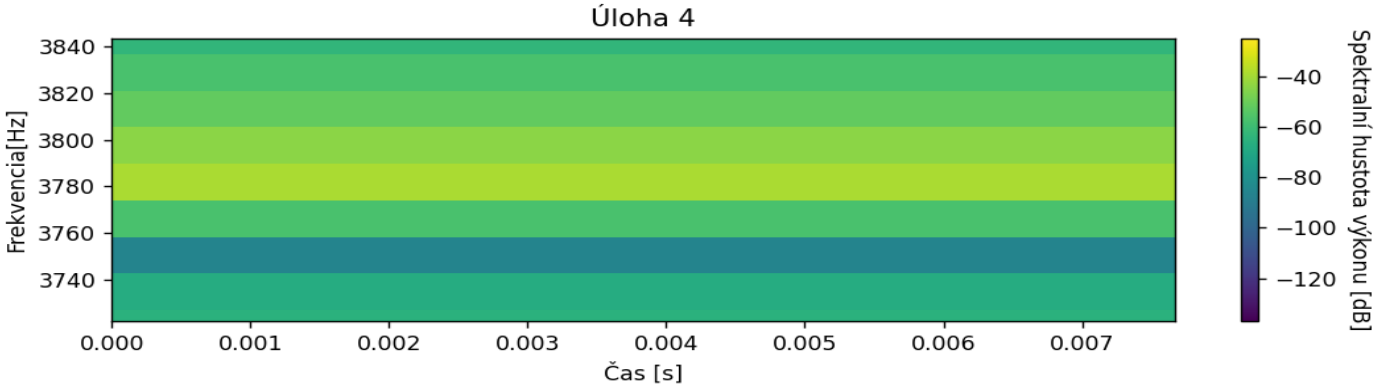


Zvyšné rušivé komponenty mali frekvencie ktoré boli násobkom prvej frekvencie.

Druhá komponenta mala frekvenciu 950*2 = 1900hz.

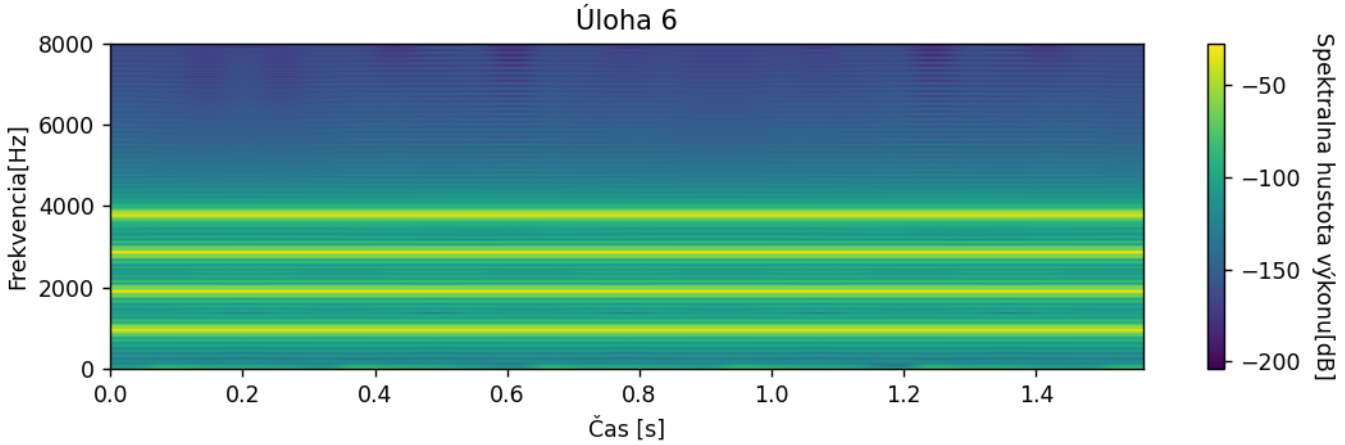
Tretia komponenta mala frekvenciu 950*3 = 2850hz.

Štvrtá komponenta nesedela úplne presne. Na spectrograme je vidieť ,že daná frekvencia nie je úplne násobkom prvej a teda z dôvodu lepších výsledkov som ďalej pracoval s hodnotou 3780hz miesto 3800hz(950*4)



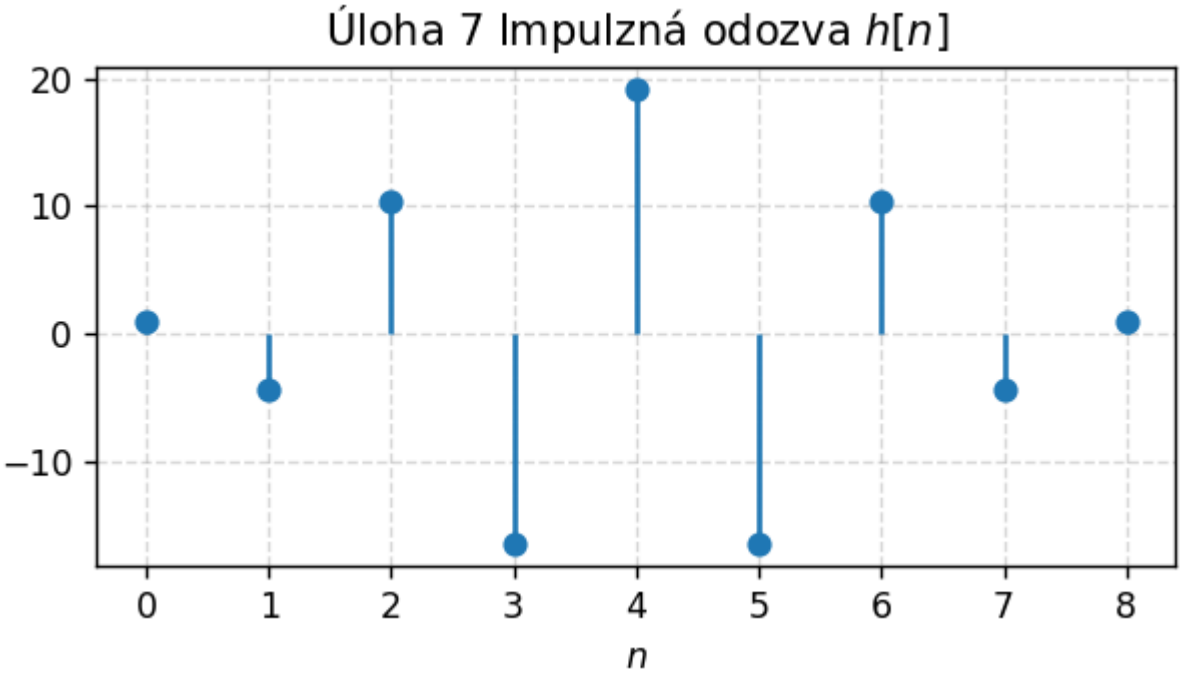
Úloha 6:

Najskôr som si vytvoril pole dát o dĺžke vstupného signálu. Tieto dáta som previedol na 4 signály kosínusu o frekvencií rušivých komponent a následne ich spojil do jedného signálu. Nakoniec som ho ustrednil, znormalizoval a vygeneroval signál rušivých komponent. Zvuk som si následne prehral a odpovedal pípaniu rušivých komponent, potvrdil to aj spectogram.



Úloha 7: Filter v z-rovine

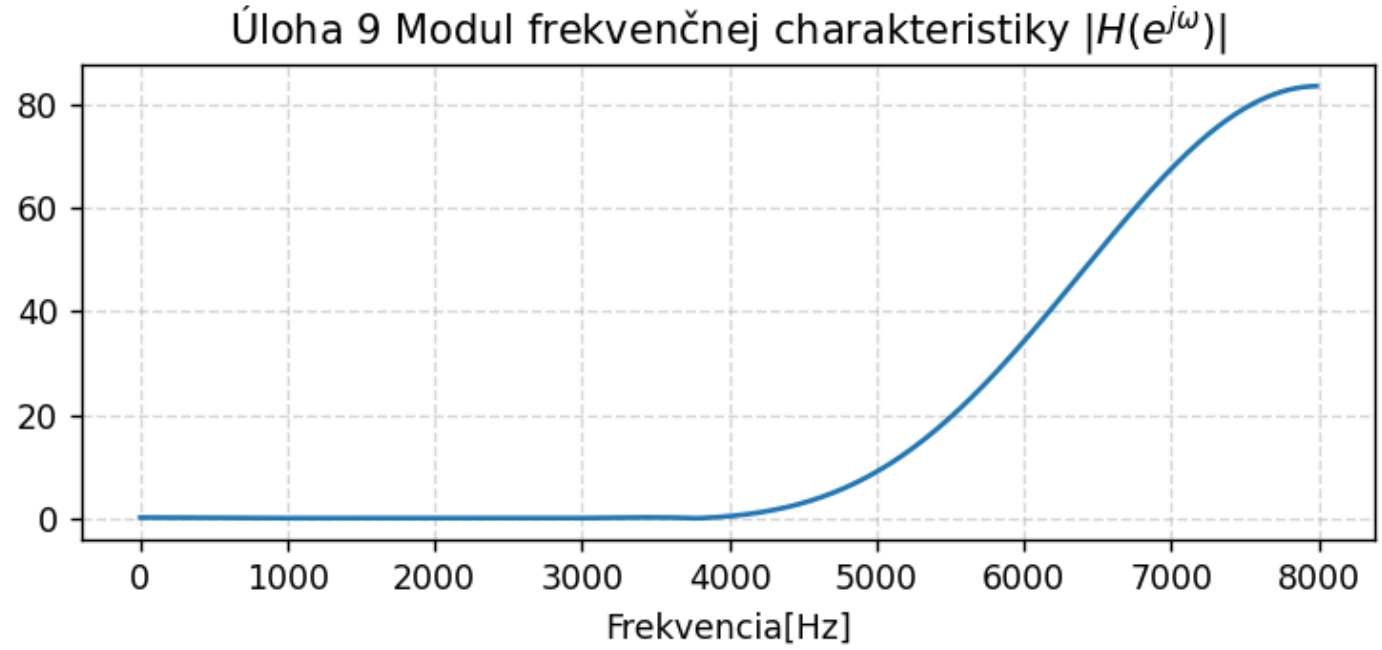
Na začiatku sa vypočíta w_k ktoré sa hneď použije na výpočet n_k ktoré sa ukladá do pola “n”. Pridajú sa k nim komplexne združené nulové body pomocou funkcie np.conj() , vytvorí sa premenná “filter” kde sa uložia tieto body, pridajú sa k nim hodnoty n_k a nulové body sa prevedú na koeficienty filtru pomocou np.poly() . Vyrobený filter však trocha skresľuje signál a to kvôli normalizácií.



Úloha 9:

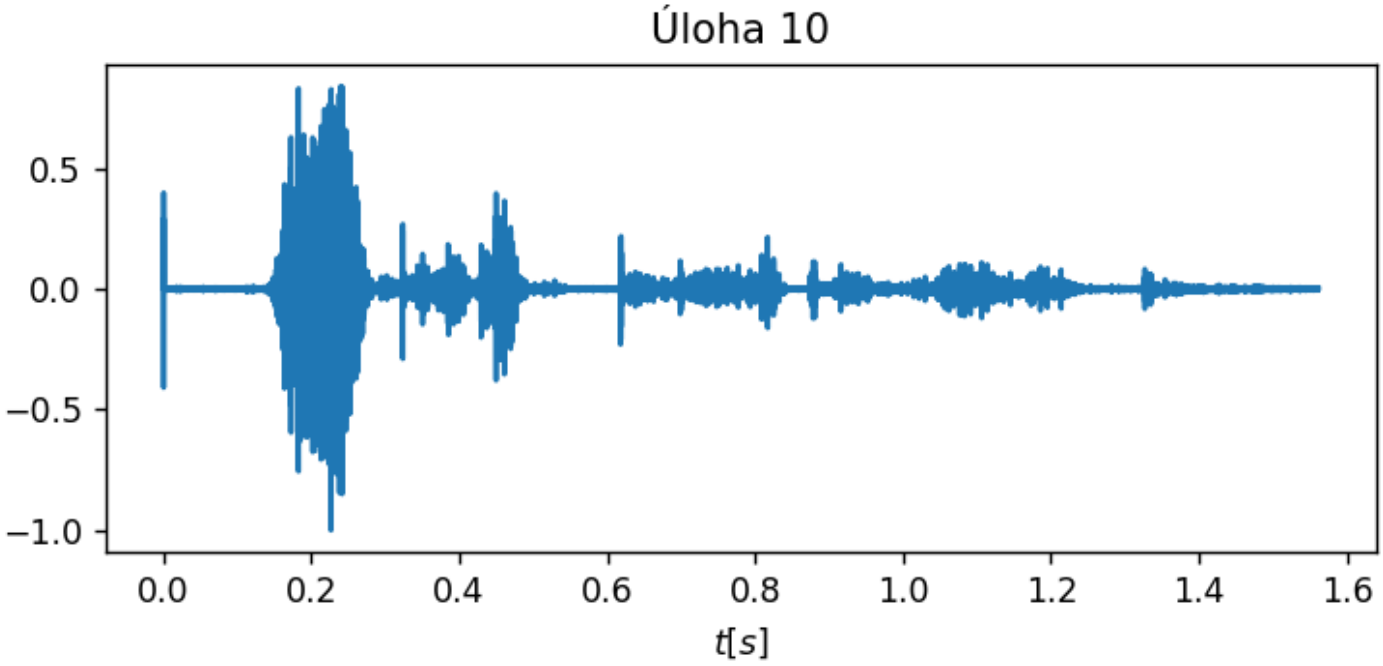
Frekvenčnú charakteristiku som vypočítal pomocou funkcie freqz().

Filter potláča nižšie frekvencie čo odpovedá frekvenciám rušivých komponent.



Úloha 10:

Signál sa najskôr ustrední a znormalizuje. Následne požijem vyrobený filter, dáta sa opäť znormalizujú a výsledný signál sa vygeneruje. Nahrávka po filtrácií je však skreslená. Filter zosilnil vysoké frekvencie.



Zdroje :

a) Dokumentácia numpy
b) Jupyter notebook (Katka Žmolíková)