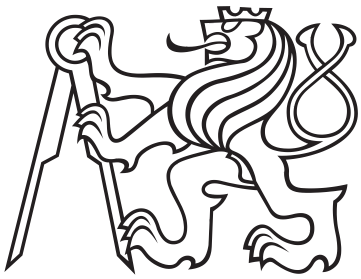
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

Katedra měření

Dekodér Morse

Zpracování signálu v reálném čase

Ondřej Tyle

Vedoucí: Prof. Ing. Pavel Zahradník, CSc.

Obor: Kybernetika a robotika

Květen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tyle** Jméno: **Ondřej** Osobní číslo: **437479**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Dekodér Morse

Název bakalářské práce anglicky:

Morse Decoder

Pokyny pro vypracování:

Na osobním počítači v systému Matlab navrhněte a naprogramujte dekodér Morseovy abecedy. Uvažujte dekódování v reálném čase. Použijte prostředky Matlabu umožňující proudové pracování bloků dat. Uvažujte minimální rychlost dekódovani 30 PARIS.

Seznam doporučené literatury:

[1] www.mathworks.com
[2] https://cs.wikipedia.org/wiki/Morseova_abeceda

Jméno a pracoviště vedoucí(ho) bakalářské práce:

prof. Ing. Pavel Zahradník, CSc., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **09.03.2020** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce:
do konce letního semestru 2020/2021

prof. Ing. Pavel Zahradník, CSc.		prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta

Poděkování

Děkuji svému vedoucímu práce za ochotu se mi věnovat a za jeho připomínky k mé práci. Děkuji také všem, kteří mi jakkoli pomohli k jejímu úspěšnému dokončení.

Prohlášení

Prohlašuji, že jsem zadanou bakalářskou práci zpracoval sám s přispěním vedoucího práce a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé bakalářské práce nebo její části se souhlasem katedry.

V Praze, 22. května 2020

Abstrakt

Tato bakalářská práce se zabývá zpracováváním zvukového signálu v reálném čase. Jejím cílem je vytvořit překladač Morseovy abecedy, který bude vysílaný text vypisovat souběžně se zvukovým výstupem.

Celá práce je psaná v programu MATLAB využívajícím převážně Audio toolbox a DSP System Toolbox.

Řešení problému je rozděleno do dvou částí, a to na přijímání signálu v reálném čase a jeho následné zpracovávání a převedení přijatého signálu na text.

Klíčová slova: Morseova abeceda, zpracování signálu, DSP System Toolbox, Audio toolbox

Vedoucí: Prof. Ing. Pavel Zahradník, CSc.

Abstract

This bachelor thesis deals with audio signal processing in real time. Its goal is to create a Morse code compiler that will print the broadcast text in parallel with the audio output.

The whole work is written in MATLAB using mainly Audio toolbox and DSP System Toolbox.

The solution of the problem is divided into two parts: the reception of the signal in real time and its subsequent processing and conversion of the received signal into text.

Keywords: Morse code, signal processing, DSP System Toolbox, Audio toolbox

Title translation: Morse decoder — Signal processing in real time

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

V této práci se budu zabývat tvorbou programu v prostředí MATLAB, který bude schopný dekódovat signál Morseovy abecedy v reálném čase. Cíle této práce jsou následující:

- Naučit se pracovat se zpracováním signálu v reálném čase.
- Přijmout nezkreslený signál.
- Dekódovat signál s minimální rychlostí 30$PARIS$.

Při práci jsem sestavil následující zapojení dvou počítačů, přes které jsem program testoval. Na prvním počítači jsem spouštěl vysílaný text a na druhém jsem ho přijímal pomocí audio vstupu. Mohl jsem tak regulovat rychlost vysílaného signálu a další parametry. V praxi by šel první počítač nahradit rádiem přijímajícím signál pomocí antény.

Kapitola 2

Příjem signálu v reálném čase

Obrázek 2.1: Zapojení pro příjem signálu.

2.1 Základní nastavení

Pro příjem signálu si nejprve musíme zadefinovat jeho vstupní parametry. V našem případě to jsou tyto:

SampleRate - 8000 - Morse signál je nízkofrekvenční, pracujeme výhradně se vzorkovacím kmitočtem 8kSPS.

NumChannels - 1 - Signál přijímáme pouze pro 1 kanál. Následně ho dále zpracováváme.

Pro zpracování dat v reálném čase využíváme funkce DSP System Toolboxu [1], respektive její funkci dsp.AudioRecorder [2] a funkci AudioDeviceWriter [3] z Audio Toolboxu [4].

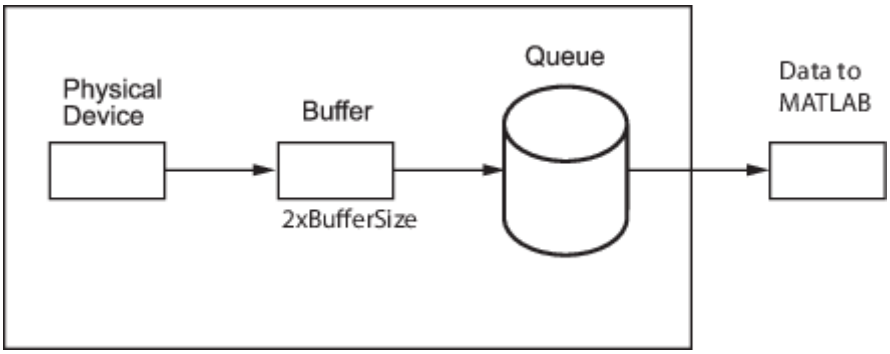
Základní nastavení programu vypadá následovně:

```
File = dsp.AudioRecorder('SampleRate', 8000,'NumChannels',1);
Fs = File.SampleRate;
Out = audioDeviceWriter('SampleRate', Fs);
```

Obrázek 2.2: Nastavení programu pro vstup a výstup signálu.

2.2 Příjem v reálném čase

Pro zpracování dat v reálném čase si nejprve musíme vysvětlit, jak funguje funkce *dsp.AudioRecorder* a jak se s ní pracuje. To si můžeme jednoduše demonstrovat na následujícím obrázku.



Obrázek 2.3: Schéma funkce AudioRecorder.[2]

Zde je vidět, že přijímaný signál začneme přijímat a ukládat do Bufferu. Po jeho naplnění se blok dat přesouvá dále do zásobníku, odkud si ho můžeme odebrat. Signál se přijímá nepřetržitě a my si můžeme odebírat vždy každý blok zvlášť a dále ho zpracovávat. Ve výsledku je to tak, že zpracování netrvá tak dlouho, jako naplnění bloku dat v Bufferu, takže nedochází ke zpomalení.

V našem případě kód pro příjem vypadá následovně:

Přijímaný signál nejdřív pojmenujeme a nastavíme základní parametry (viz. 1. řádek z obrázku 2.2. Dále vytvoříme cyklus, ve kterém budeme volat funkci *step*(), která vždy vybere blok přijatých dat a dále ho zpracujeme.

```
1 while true
2     x = step(File);
3     step(Out,x);
4     %Zpracovani dat
5 end
6
```

Obrázek 2.4: Cyklus pro zpracování v reálném čase.

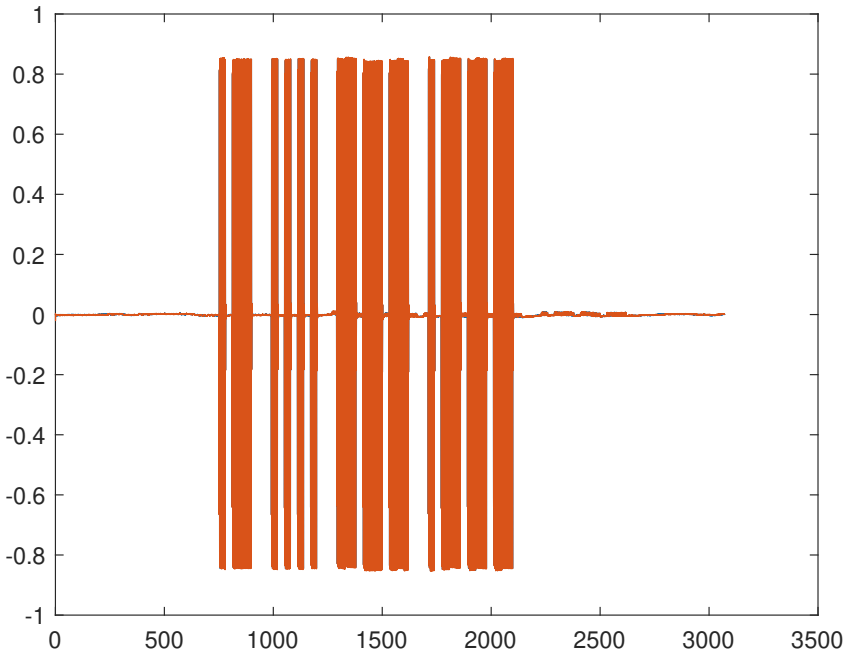
Kapitola 3

Zpracování signálu

3.1 Převod audio signálu do binárního

Po přijetí audio signálu nám počítač přijme data.

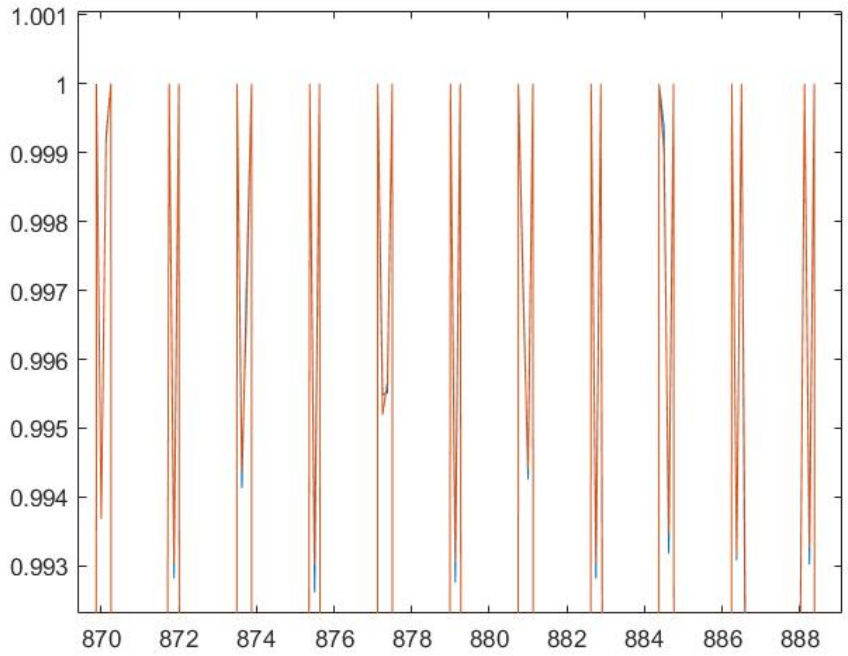
Z obrázku 3.1 je vidět, že vysílaný text je slovo *AHOJ*.
Pro demonstraci je na obrázku 3.1 vidět celý signál a ne jen jeden blok, jak je tomu normálně při běhu programu. Na jednom bloku dat není vidět skoro nic a tudíž nelze demonstrovat postupy, které jsem použil. Vždy, když se zde objeví další takový obrázek, berte to prosím jako demonstraci, která se provádí na každý blok dat zvlášť. Kdybychom zpracované bloky dat poskládali za sebe, jistě bychom dostali úplně stejné obrázky, jako jsou tyto.



Obrázek 3.1: Základní přijatá data.

Při přijímání signálu nastává jedna obtíž, a to rozsah, v jakém lze signál přijmout. Funkce *dsp.AudioRecorder* totiž zvládá přijímat signál jen do určité hlasitosti. Pokud tedy signál bude hlasitější, funkce *dsp.AudioRecorder* ho přijme, ale vše silnější, než je jistá hranice se zdeformuje. Na obrázku 3.2 je vidět příklad, kdy se po přijetí hlasitějšího signálu vršky sinusovky začaly zalamovat dovnitř.

Tento problém lze vyřešit připojením kabelu pro příjem dat do vstupu AUDIO IN místo vstupu pro mikrofon. Na mém notebooku ale tento vstup není. Musel jsem tedy hlídat, s jakou hlasitostí signál vysílám z druhého počítače.*



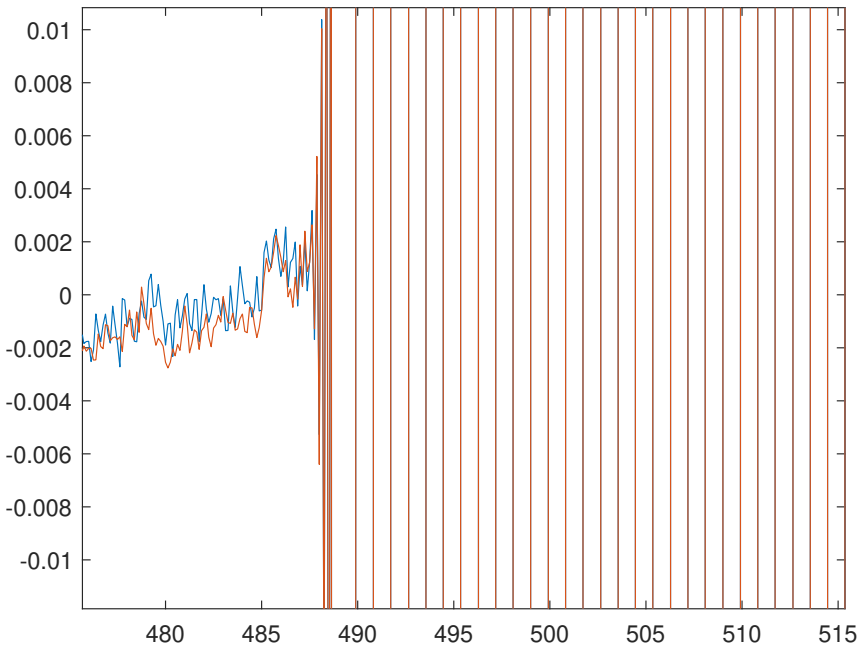
Obrázek 3.2: Přijatý zdeformovaný signál.

Zde bylo potřeba definovat si rozsah, v jakém je možné signál dekódovat. V mém případě jsem rozsah definoval od 10% do 90% rozsahu.

---

* Na chod programu by tento problém neměl mít vliv. Je to jen jistá obtíž, se kterou musíme počítat.

Když si signál přiblížíme (obrázek 3.3), je vidět, že šum se projevuje jen velice málo, jelikož používáme signál přímo z audio vstupu, nikoli pomocí mikrofonu. Odchylka od klidového stavu (tedy od 0) je zde pouze v maximálním rozmezí ± 0.01.

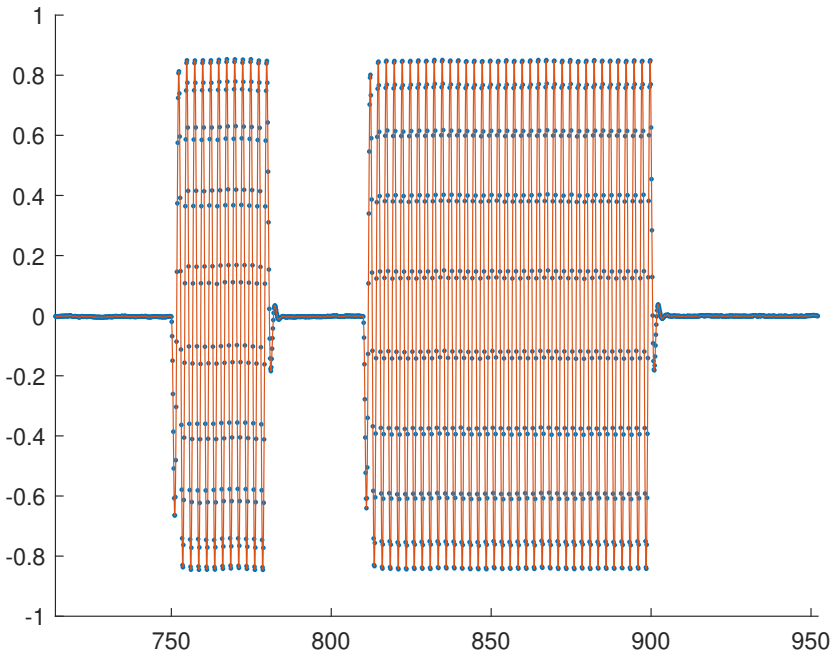


Obrázek 3.3: Základní přiblížená přijatá data.

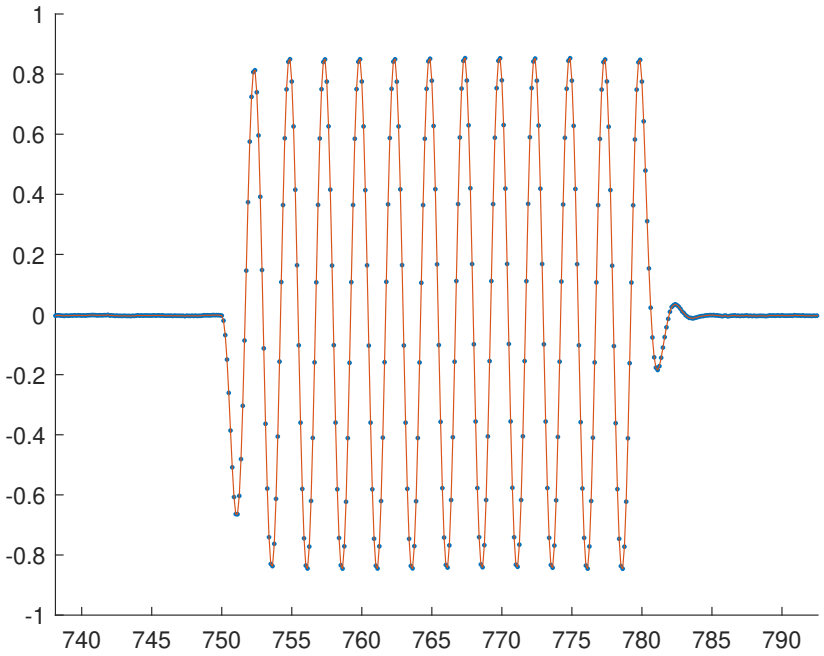
Z přijatých dat potřebujeme vyčíst, kdy je přítomnost tónu aktivní a kdy ne. Chceme tedy ve výsledku docílit toho, že budeme mít časovou osu a na ní buďto 1 nebo 0. Náš signál tedy nejprve do tohoto tvaru musíme dostat. To provedeme následujícími kroky.

3.1.1 Převedení do kladné osy

Pro lepší demonstraci si ukážeme obrázek 3.4, na kterém jsou vidět přijaté hodnoty, propojené křivkou. Na tomto obrázku je vidět pouze písmeno *a* a na obrázku 3.5 je vidět přiblížená první tečka pro přehlednost.

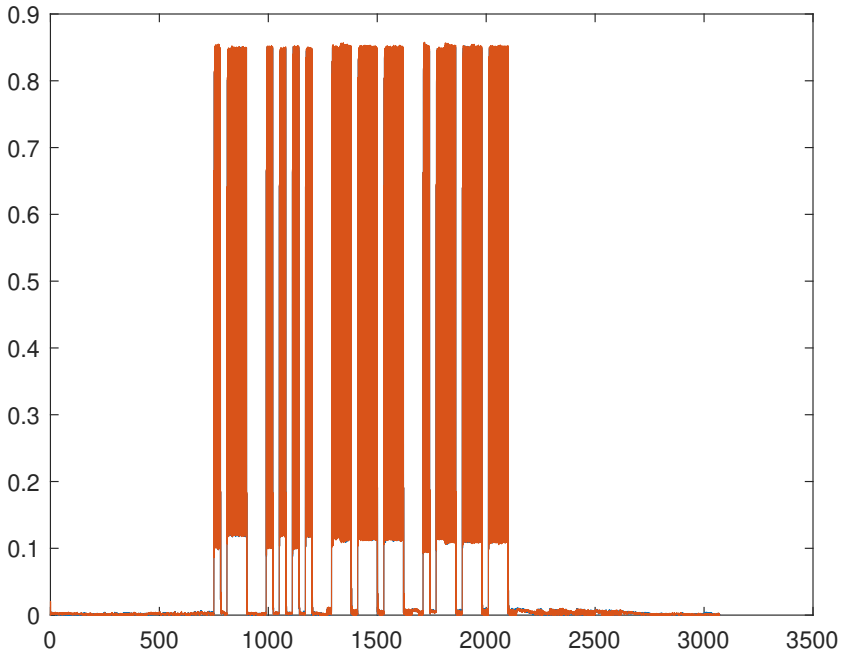


Obrázek 3.4: Písmeno *a* se zobrazenými přijatými hodnotami.



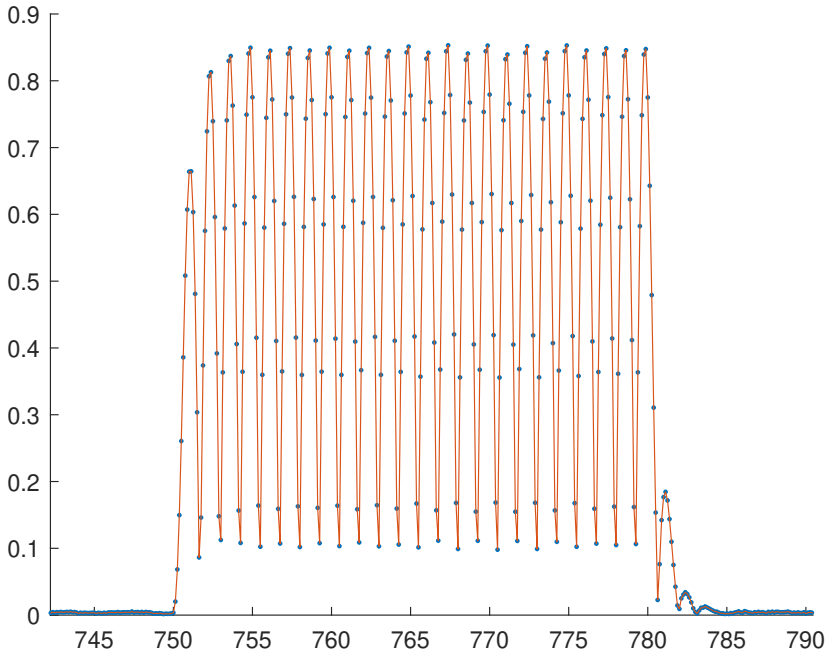
Obrázek 3.5: Tečka se zobrazenými přijatými hodnotami.

Zde je vidět, že přijatý signál je sinusového tvaru. Nás ale zajímá pouze jeho přítomnost. Převedeme ho tedy nejprve do kladné osy aplikováním funkce *abs*(), tedy absolutní hodnoty. Výsledný signál nyní vypadá takto:



Obrázek 3.6: Signál po aplikování absolutní hodnoty.

Zde uvedeme ještě přiblížený obrázek pro a se zobrazenými body. Obrázek osvětluje bílé místo v obrázku 3.6 v rozmezí od 0 do 0.1, které je způsobeno absencí hodnot kolem 0. Po překlopení absolutní hodnotou se tedy dole udělá místo a body zůstávají pouze výše.



Obrázek 3.7: Písmeno *a* po aplikování absolutní hodnoty.

Nyní chceme odstranit sinusové kmitání při daném tónu a nahradit ho čistou 1 nebo 0.

3.1.2 Odstranění sinusového kmitání

Sinusové kmitání lze jednoduše odstranit aplikací filtru. Je zde mnoho možností, jaký filtr zvolit. Testoval jsem plovoucí průměr a také dolní propust. Parametry pro návrh filtru dolní propusti jsem volil následovně:

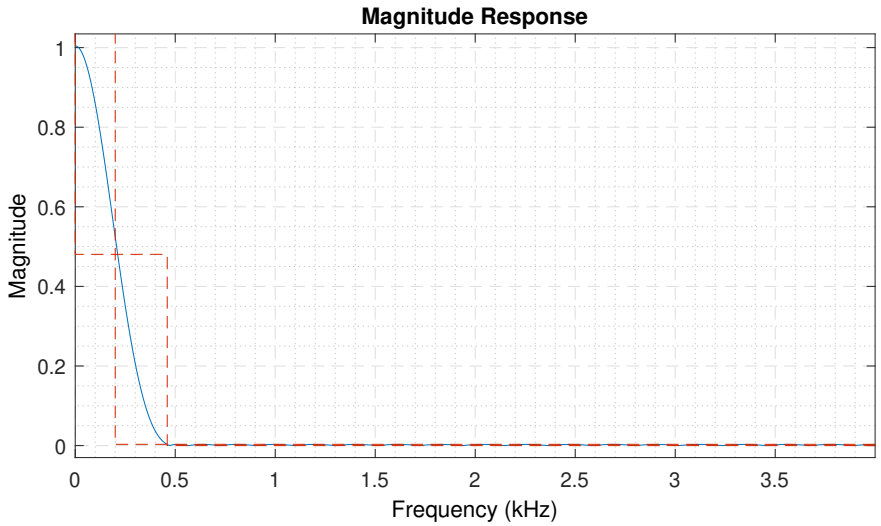
- řád filtru: 42
- maximální útlum v propustném pásmu: 10dB
- minimální útlum v nepropustném pásmu: 50dB
- mez propustného pásma: 200Hz
- mez nepropustného pásma: 460Hz

Kód pro jeho použití je následující:

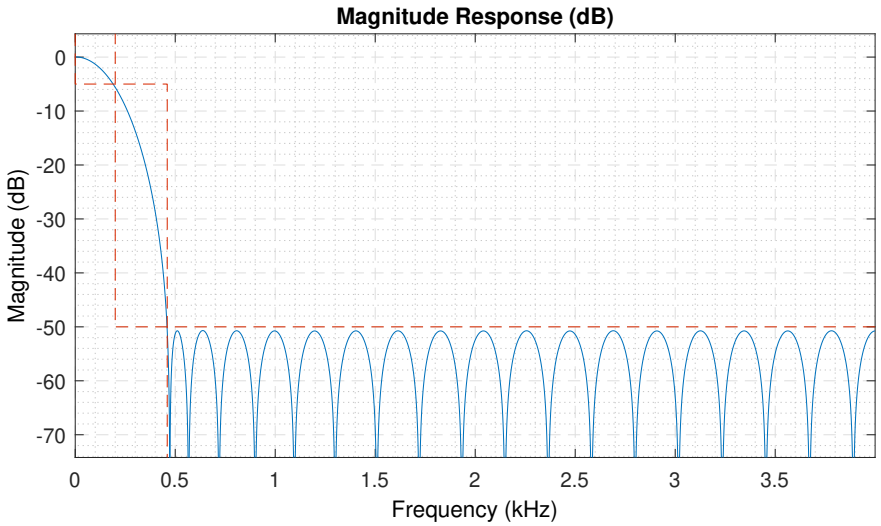
```
d = designfilt('lowpassfir','PassbandFrequency',0.05,...
    'StopbandFrequency',0.115,'PassbandRipple',10,...
    'StopbandAttenuation',50);
x = step(File);
z = filter(d,abs(x));

```

Obrázek 3.8: Aplikace dolní propusti.



Obrázek 3.9: Magnitudová charakteristika filtru(lineární).



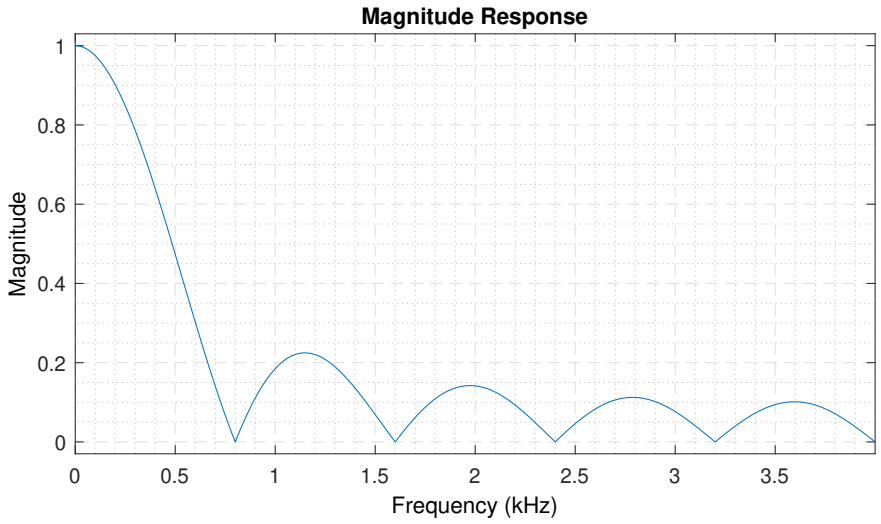
Obrázek 3.10: Magnitudová charakteristika filtru(v dB).

Pro návrh plovoucího průměru jsem volil rozsah průměrování na 10 bodů. Z obrázku 3.7 je vidět, že právě cca 10 bodů se nachází v jedné periodě sinusové funkce.

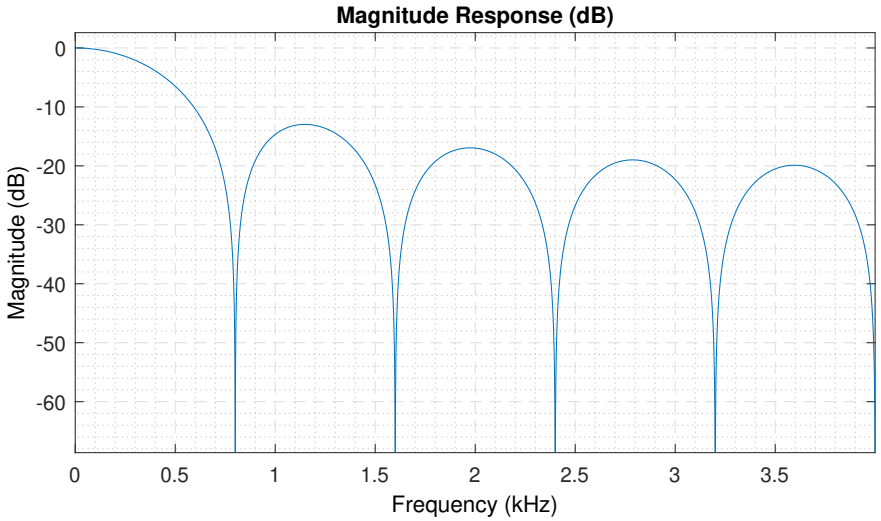
Matlabové použití je následující:

```
1 B = 1/10*ones(10,1);
2 x = step(File);
3 z = filter(B,1,abs(x));
4
```

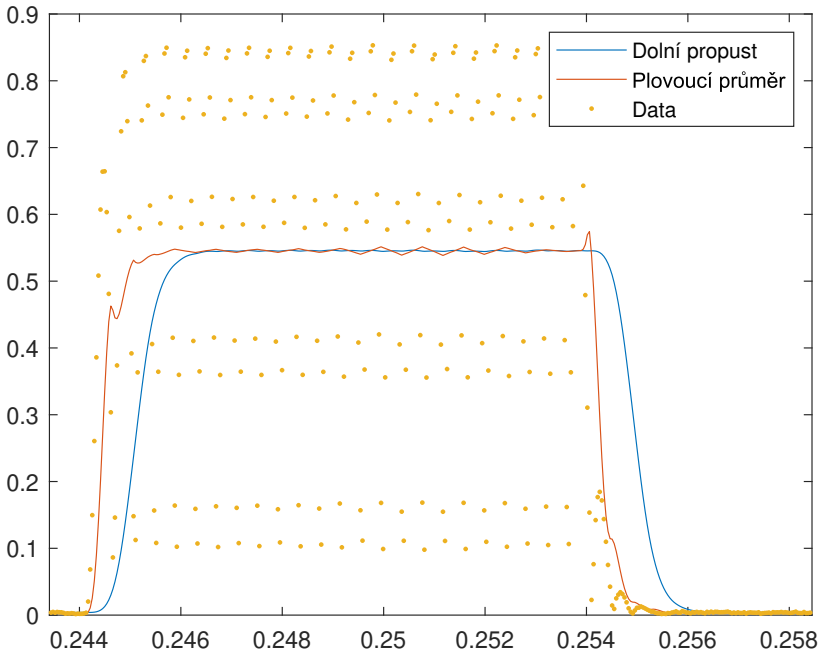
Obrázek 3.11: Aplikace plovoucího průměru.



Obrázek 3.12: Magnitudová charakteristika plovoucího průměru(lineární).



Obrázek 3.13: Magnitudová charakteristika plovoucího průměru(dB).

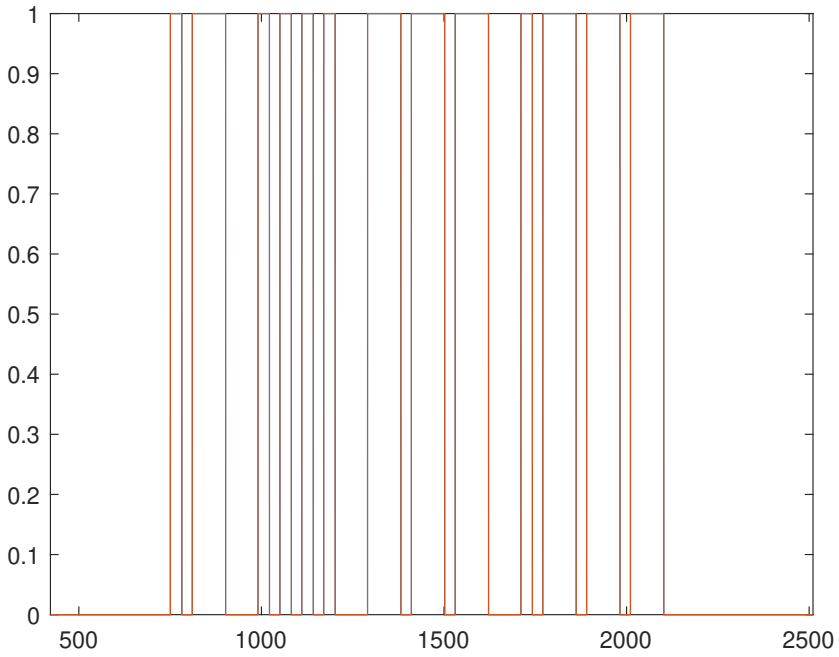


Obrázek 3.14: Srovnání dolní propusti a plovoucího průměru.

Testováním obou variant jsem zjistil, že plovoucí průměr je přibližně čtyřikrát rychlejší než dolní propust. I když je jeho graf "ošklivější", na chodu programu to nic nemění. Používám tedy jej.

3.1.3 Konečná úprava

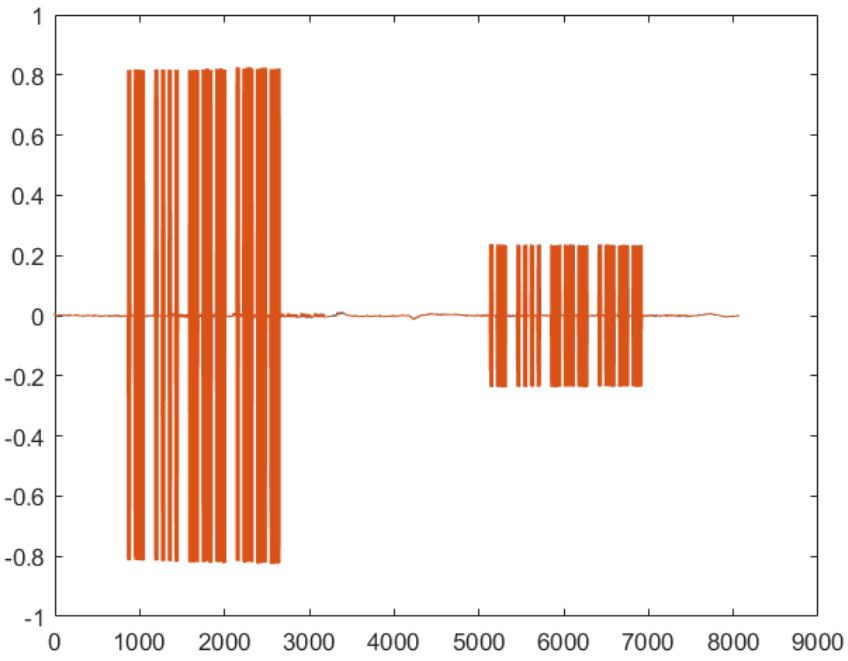
Výsledný signál převedeme na binární tak, že použijeme podmínku >, kterou rozdělíme signál na hodnoty větší než daná hodnota, která v mém případě je 0.05. Výsledný signál v časové ose vypadá následovně:



Obrázek 3.15: Signál převedený do binárního.

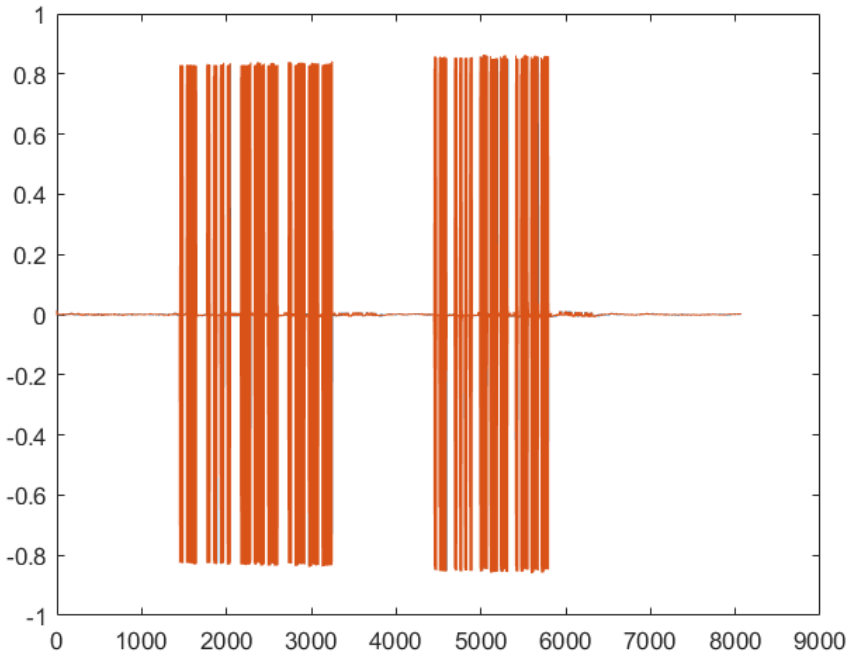
3.1.4 Rozsah a rychlost signálu

Zde bych jen uvedl, že již zmíněný rozsah přijímaného signálu jsem stanovil od 5% do 90% rozsahu. Přikládám demonstrativní obrázek pro porovnání.



Obrázek 3.16: Rozsah přijímaného signálu (první je 80% a druhý 20% rozsahu).

Rychlostí, kterou program zvládne dekódovat, myslím rychlost odesílaného signálu, tedy jak dlouho trvá tečka a čárka.
Ze zadání mám řečeno, že program by měl dekódovat signál vysílaný minimální rychlostí $30PARIS$*. Testoval jsem tedy hranice, kdy již program není schopen signál dekódovat. Při hodnotách kolem $50PARIS$ již program nerozpozná tečku a čárku. Je zde již málo hodnot pro porovnání. Pro srovnání přikládám obrázek, kde první slovo je vysíláno rychlostí $30PARIS$ a druhé rychlostí $40PARIS$. Minimální rychlost je $3PARIS$.



Obrázek 3.17: Porovnání rychlostí přijímaného signálu.

---

*[5] Pro určení rychlosti vysílání se bere jako reference pětipísmenné slovo "PARIS"(celkem 10 teček, 4 čárky, 4 mezery). Celková doba jeho odvysílání včetně mezery za slovem je tedy 50 základních jednotek (délek teček). Pokud tedy například hovoříme o tempu vysílání 12 slov za minutu (WPM, words per minute), odpovídá to právě rychlosti průměrně jednoho písmene (znaku) za sekundu (12*5 = 60 znaků za minutu), délka tečky je 60/(12*50) = 0,1s.

3.1.5 Celý kód pro zpracování v reálném čase

```
File = dsp.AudioRecorder('SampleRate', 8000,'NumChannels',1);
Fs = File.SampleRate;
Out = audioDeviceWriter('SampleRate', Fs);
%-------------------------------------------------------------
c = [0 0 0;0 0 0];                %deklarace promennych a poli
temp = 0;
out = "";
B = 1/10*ones(10,1);                          %plovouci prumer
%-------------------------------------------------------------
while true           %program bezi dokud neni manualne ukoncen
    x = step(File);          %vyber bloku dat ze zasobniku
    step(Out,x);                  %vypis dat do reproduktoru
    z = filter(B,1,abs(x));       %aplikace plovouciho prumeru
    z(1:40) = [];                   %redukce prechodoveho jevu
    y = z > 0.05 ;                     %prevedeni na binarni
    %-------------------------------------------------------
     if isempty(c)                       %korekce nuloveho pole
        c = [0 0 0];
    end
    for i = 1:size(y)    %secteni bodu se stejnou hodnotou
        if y(i) == temp
            c(end,2) = c(end,2) + 1;
        else
            c(end,1) = temp;
            temp = abs(temp-1);
            if c(end,2) ~= 0
                c = [c  ;0 0 0];
            end
        end
    end
    [c,unit] = IO2morse2(c);      %prevod do Morseovy abecedy
    code = c(:,3);       %vyjmuti dat pro vysledny preklad

    idx=find(c(:,3)==124);          %hledani mezery mezi slovy
    if ~isempty(idx)               %ulozeni slova a mazani dat
        out = out + demorse(char(code(1:idx)'));
        c(1:idx,:) = [];
    end
    code = c(:,3);       %vyjmuti dat pro vysledny preklad
    text = out + demorse(char(code'))          %preklad Morse
end

```

Obrázek 3.18: Kód pro zpracování signálu v reálném čase.

3.2 Převod binárního signálu do Morseovy abecedy

3.2.1 Převod signálu na matici

Přijatý signál jednoduchým for cyklem převedeme na matici, která bude obsahovat v prvním sloupci 1 nebo 0 a ve druhém sloupci počet, kolikrát se dané číslo vyskytlo, dokud se nezměnilo na to druhé. Například u slova *AHOJ* to bude vypadat následovně:

0	0	⋮	⋮	⋮	⋮
0	10264	1	331	0	939
1	337	0	301	1	338
0	300	1	337	0	300
1	974	0	940	1	972
0	939	1	973	0	301
1	337	0	300	1	971
0	301	1	972	0	301
1	337	0	300	1	971
0	301	1	972	0	11157
⋮	⋮	⋮	⋮	0	0

Tabulka 3.1: Hodnoty výskytu jednotlivých čísel.

Z tabulky je vidět, že první a poslední hodnota u 0 (zanedbáme-li úplně první a poslední nulovou hodnotu) je mnohonásobně větší, než u ostatních čísel. To je způsobeno čekáním před začátkem vysílání a po jeho ukončení. Další hodnoty vidíme relativně stejné. Některé se pohybují kolem $300 - 330$ a jiné kolem $940 - 970$. To nám ukazuje, kde je čárka a kde tečka. abychom ale dokázali signál dekódovat, je potřeba zachovávat univerzální délky tečky,

čárky a všech pauz. Tyto délky jsou vypsány v následující tabulce.

tečka	1 x *unit*
čárka	3 x *unit*
pauza mezi symboly v písmenu	1 x *unit*
pauza mezi jednotlivými písmeny	3 x *unit*
pauza mezi slovy	7 x *unit*

Tabulka 3.2: Pravidla pro časování v Morseově abecedě.

Kde *unit* se určuje podle rychlosti vysílání.

3.2.2 Hledání *unit*

Při překladu signálu v reálném čase se tabulka s hodnotami pro 1 a 0 doplňuje během vysílání. Je z ní zapotřebí určit délku tečky, abychom následně mohli správně přeložit vysílaný signál. *unit* hledáme tak, že si vždy z tabulky vybereme nejmenší hodnotu pro 1 a 0 a uděláme u nich průměr. Tato hodnota se stále mění a vylepšuje se v průběhu překladu. Může se tedy stát, že na začátku překladu bude přeložený text chybný v důsledku absence dat pro správné určení *unit*. Například u textu, který bude začínat na písmeno T se nejprve ukáže E a až po odeslání nějakého dalšího znaku obsahujícího tečku se přeložený text opraví.

```
1  function unit = findunit(d)
2  I = d((d(:,1)==1),:);
3  O = d((d(:,1)==0),:);
4  if O(1,2) == 0
5     O(1,:) = [];
6  end
7  u1 = min(I(:,2));
8  u2 = min(O(:,2));
9  unit = (u1+u2)/2;
10 end
11
```

Obrázek 3.19: Funkce pro hledání *unit*

Kde d je tabulka hodnot pro 1 a 0 viz Tabulka 3.1.

3.2.3 Výsledný převod do Morseovy abecedy

S nalezenou hodnotou pro *unit* již můžeme signál přeložit do Morseovy abecedy. To uděláme jednoduchým for cyklem, při kterém postupně projdeme matici s hodnotami a přiřadíme k nim znak z Morseovy abecedy. Vše ukládáme do dalšího sloupce v matici pro přehlednost a možnou kontrolu. Celá funkce pro převod je vidět níže.

```
1  function [out,unit]=IO2morse2(d)
2  unit = findunit(d);
3  carka = unit*3;
4  tecka = unit;
5  vmezera = unit*3-unit/8;
6  slovo = unit*7-unit/8;
7  for i = 1:size(d,1)
8      if d(i,1) == 1
9          if d(i,2) > carka
10             d(i,3) = '-';
11         elseif d(i,2) > tecka
12                d(i,3) = '.';
13         end
14     else
15         if d(i,2) > slovo
16             d(i,3) = '|';
17         elseif d(i,2) > vmezera
18             d(i,3) = '/';
19         end
20     end
21 end
22 out = d;
23 end
24
```

Obrázek 3.20: Funkce pro převod do Morseovy abecedy

Kde d je tabulka hodnot pro 1 a 0 viz Tabulka 3.1 a *unit* vypočítaná hodnota pomocí funkce $findunit$ viz kapitola 3.2.2.

V 5. a 6. řádku si můžete povšimnout korekce pro mezery, která koriguje správné vypisování znaků. Hodnota $unit/8$ je získána praktickými experimenty.

Po překladu signálu si z matice dále přeložený text uložíme do speciální pro-

měnné, kterou budeme následně převádět do češtiny. Proměnná připravená na překlad vypadá následovně:

124	46	45	47	46	46	46	46	47	...
...	45	45	45	47	46	45	45	45	124

Tabulka 3.3: Vysílaný text(AHOJ) v Morseově abecedě v kódu ASCII.

Zde je vidět, že se text zobrazuje ne pomocí znaků Morseovy abecedy, ale pomocí jejich hodnot v kódu ASCII.

3.3 Převod textu z Morseovy abecedy do češtiny

Převod z Morseovy abecedy probíhá jednoduchým přiřazením jednotlivých znaků abecedy k jednotlivým písmenům. Učiníme tak vnořeným for cyklem. Funkce *demorse* popisující přepis viz níže.

```
function text = demorse(text)

text= text(find(~isspace(text)));        %mazani mezer v textu

morse={'.-','-...','-.-.','-..','.','..-.','--.','....','..',
'.---','-.-','.-..','--','-.','---','.--.','--.-','.-.','...',
'-','..-','...-','.--','-..-','-.--','--..'};
letter={'A','B','C','D','E','F','G','H','I','J','K','L','M',
'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};
chars = strsplit(text,'|')';          %rozdeleni textu podle |
for i = 1 : size(chars,1)
    out{i} = split(chars{i},'/');
end
code = "";
for i = 1:length(out)                          %preklad slov
    for j = 1:size(out{i})
        for k = 1:length(letter)
            if strcmpi(out{i}(j),morse(k))==1
                code= code + letter(k);
            end
        end
    end
    code= code + " ";
end
text = code;
end

```

Obrázek 3.21: Funkce pro převod Morseovy abecedy do češtiny.

Kde *text* je soubor s daty z tabulky 3.3 převedený z ASCII na symboly Morseovy abecedy.

Výsledný přepsaný text vypisujeme v přímém čase do terminálového okna.

Kapitola 4

Závěr

V této práci jsou popsány funkce Audio Toolboxu a DSP System Toolboxu umožňující zpracování signálu v reálném čase. Jsou zde vypsány základní parametry pro příjem signálu a také výsledný program umožňující zpracování v reálném čase. Je zde ukázáno jak přijatý signál upravit, odfiltrovat a dále zpracovat a připravit na dekódování. Jsou zde také ukázány funkce pro samotné dekódování a následné vypisování včetně funkce na hledání *unit*. Všechny cíle práce tak byly splněny.

Literatura

[1] *DSP System Toolbox* The MathWorks, Inc. MathWorks - Makers of MATLAB and Simulink [online]. Copyright © 1994 [cit. 29.04.2020]. Dostupné z: https://es.mathworks.com/products/dsp-system.html

[2] *dsp.AudioRecorder* The MathWorks, Inc. MathWorks - Makers of MATLAB and Simulink [online]. Copyright © 1994 [cit. 21.04.2020]. Dostupné z: https://www.mathworks.com/help/dsp/ref/dsp.audiorecorder-system-object.html

[3] *AudioDeviceWriter* The MathWorks, Inc. MathWorks - Makers of MATLAB and Simulink [online]. Copyright © 1994 [cit. 21.04.2020]. Dostupné z: z: https://es.mathworks.com/help/audio/ref/audiodevicewriter-system-object.html

[4] *AudioToolbox* The MathWorks, Inc. MathWorks - Makers of MATLAB and Simulink [online]. Copyright © 1994 [cit. 21.04.2020]. Dostupné z: Dostupné z: https://es.mathworks.com/help/audio/index.html

[5] *Morseova abeceda* Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Morseova_abeceda

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tyle** Jméno: **Ondřej** Osobní číslo: **437479**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Dekodér Morse

Název bakalářské práce anglicky:

Morse Decoder

Pokyny pro vypracování:

Na osobním počítači v systému Matlab navrhněte a naprogramujte dekodér Morseovy abecedy. Uvažujte dekódování v reálném čase. Použijte prostředky Matlabu umožňující proudové pracování bloků dat. Uvažujte minimální rychlost dekódovani 30 PARIS.

Seznam doporučené literatury:

[1] www.mathworks.com
[2] https://cs.wikipedia.org/wiki/Morseova_abeceda

Jméno a pracoviště vedoucí(ho) bakalářské práce:

prof. Ing. Pavel Zahradník, CSc., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **09.03.2020** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce:
do konce letního semestru 2020/2021

prof. Ing. Pavel Zahradník, CSc.		prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta