## 1.1. Nástroj AppDesigner

Na predchádzajúcej prednáške sme k tvorbe grafického používateľského prostredia používali nástroj GUIDE. Tento nástroj bol v MATLABe implementovaný približne pred dvadsiatimi rokmi. Zavedenie nástroja pre generovanie GUI bolo prelomové, ale tento nástroj zastaral a približne od verzie R2018 nie je podporovaný a je možné, že v budúcnosti bude z Matlabu úplne odstránený. Zastaraný nástroj GUIDE je od verzie R2016 nahradený nástrojom AppDesigner, ktorý umožňuje vytvárať moderné GUI s výrazne vyšším počtom prvkov a možnosti. Spôsob programovania v tomto nástroji je odlišný ako tomu bolo v nástroji GUIDE. V AppDesigneri nepracujeme so štruktúrou handles. Spôsob práce v tomto nástroji názorne ukážeme na nasledujúcom príklade.

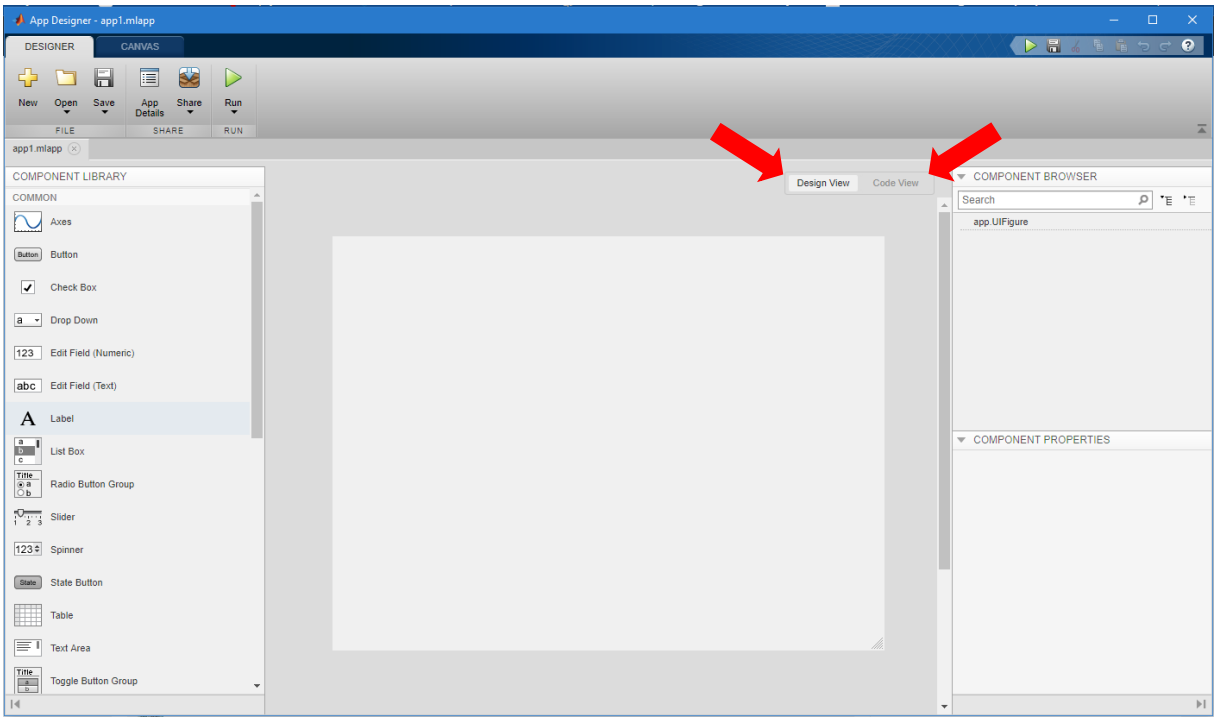
**Príklad:** Pomocou nástroja AppDesigner vytvorte aplikáciu pre načítanie nameraných údajov z .csv súboru. Program na začiatku umožňuje iba načítanie súborov, všetky ďalšie funkcionality programu nesmú byť používateľovi prístupné. Načítané údaje nasledovne zobrazte do grafu, a do príslušných polí pre zobrazovanie statického textu vypíšte štatistické parametre (stredná hodnota, disperzia, smerodajná odchýlka, modus a medián). V prípade, že používateľ načíta viacero dátových súborov program umožní pomocou listboxu používateľovi určiť, ktoré dáta chce zobraziť. Program ďalej bude umožňovať filtráciu načítaných dát pomocou jednorozmerného číslicového DP spriemerňovacieho filtra, ktorého dĺžku impulznej odpovede bude používateľ nastavovať pomocou „Slidera“.

Údaje v dátovom súbore sú uložené v nasledovnom formáte:

```
No,    Time,   Value
1,     0.005,  0.00
2,     0.010,  0.02
       ....
```

**Riešenie:**

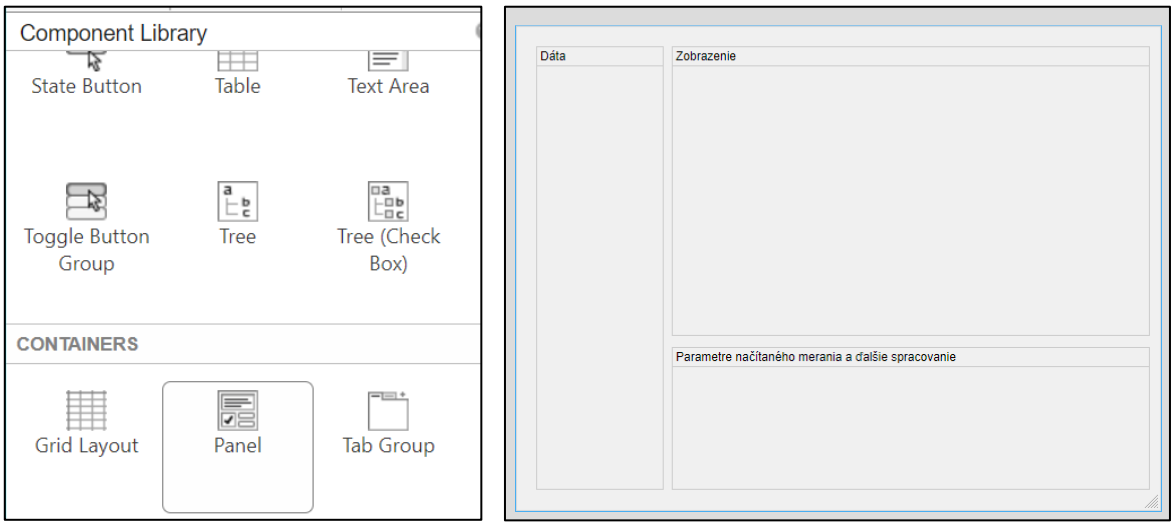
Do príkazového riadku zadáme príkaz ***appdesigner***, zvolíme „Blank App“. Následne sa zobrazí vývojové okno. V prostrednej časti vývojového okna je zobrazený aktuálny návrh GUI, do ktorého je možné umiestňovať ovládacie prvky podobným spôsobom ako v nástroji GUIDE. Na rozdiel od nástroja GUIDE, kód pre obsluhu jednotlivých prvkov sa kód negeneruje do separátneho súboru. Medzi editovaním rozloženia prvkov a kódom sa dá prepínať pomocou prepínačov „Design a Code View“. V ľavej časti okna sa nachádza knižnica obsahujúca ovládacie prvky, ktoré je možné pri tvorbe GUI využiť.




**Obr. 1 Vývojové prostredie nástroja AppDesigner**

### 1.1.1. Dizajn a rozmiestnenie ovládacích prvkov

V ďalšom kroku je potrebné navrhnúť základné rozloženie navrhovanej aplikácie. Z panelu „Component Library“ presunieme do návrhu programu komponenty a pokúšame sa ich nejakým rozumným spôsobom rozmiestniť. Jednotlivé prvky sa snažíme umiestňovať do logických celkov – panelov.



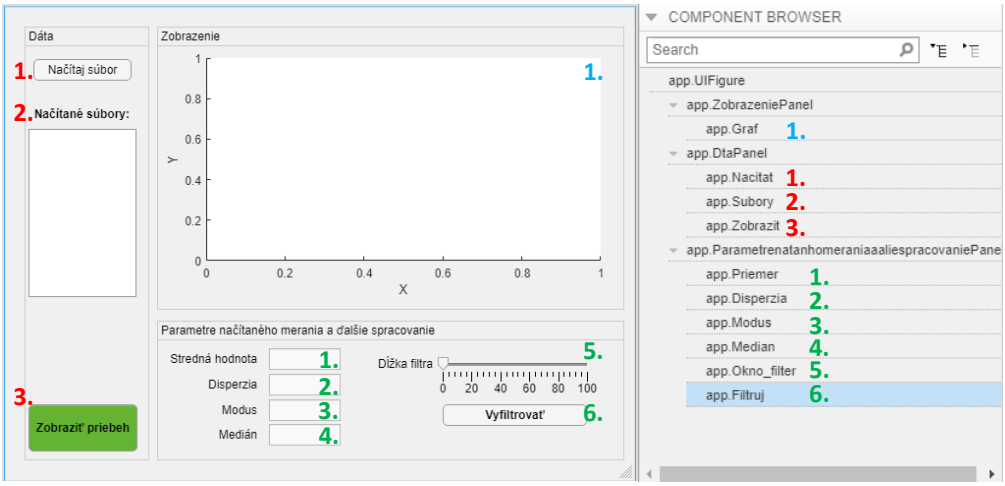

**Obr. 2 Náhľad panela „Component Library“ a rozvrhnutie panelov navrhovaného GUI**

Prvky vyvinutého programu by mohli byť rozmiestnené tak ako je uvedené na Obr. 3. Z panela „Component Browser“ vyberáme nasledovné prvky.

- Button – Načítaj súbor, Zobraziť priebeh a Vyfiltrovať
- Listbox – Načítané súbory
- Axes – Graf

- Edit Field (Text) – Stredná hodnota, Disperzia, Modus, Medián
- Slider – Dĺžka filtra

Funkciou jednotlivých prvkov sa budeme zaoberať v ďalšom texte, ale u väčšiny prvkov je z ich názvov zrejmé k akému účelu budú slúžiť.




**Obr. 3 Náhľad panela „Component Browser“ a rozvrhnutie ovladacích prvkov navrhovaného GUI**

Je potrebné si uvedomiť, že proces návrhu grafického rozhrania je viacmennej kreatívna a niekedy až umelecká činnosť. Výrazne záleží od miery kreativity vývojára. Návrh GUI tiež nespočíva iba v rozmiestnení ovládacích prvkov. Už v tomto kroku je potrebné myslieť na rozumné označovanie prvkov pomocou ich „tag-ov“ tak, aby ich v kóde bolo možné jednoducho použiť! Pomenovať je potrebné hlavne tlačidlá, textové polia a podobne. Naopak panely a grafy zvyčajne nemusíme pomenovať. Prvky pomenúvame v paneli „Component browser“.

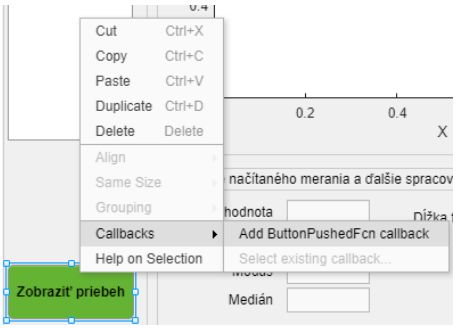
Tiež by sme nemali zabudnúť na to, že textové polia kde sa majú informácie iba zobrazovať by nemali byť používateľom editovateľné. Toto je možné nastaviť pre každé jedno pole v paneli „Edit field properties“.


**Obr. 4 Náhľad panela pre editáciu vlastnosti prvku**

### 1.1.2. Programovanie funkcionalít

Tak, ako pri programovaní obyčajných funkcií, resp. skriptov, aj v tomto prípade postupujeme v štýle „krok-po-kroku“. Nesnažíme sa naprogramovať všetko naraz! Najlepšie v danom prípade bude, ak najprv naprogramujeme funkcionalitu tlačidla „Zobraziť priebeh“. Po stlačení tlačidla sa má zobraziť zvolený priebeh do príslušného grafu. Nakoľko ešte nemáme vyriešené načítavanie dát, v danom tlačidle si dočasne vygenerujeme dáta vlastné.

**CallBack pre tlačidlo „Zobraziť priebeh“:** Pravým tlačidlom myši klikneme na tlačidlo z ponuky vyberieme nasledovne:




**Obr. 5 Kontextové okno pre vygenerovanie callback funkcie ovládacieho prvku**

Automaticky sa vývojové prostredie prepne do režimu „Code View“, kde si môžeme všimnúť, že pribudla funkcia ***ZobrazitButtonPushed(app, event)***, ktorá je editovateľná. Riadky kódu, ktoré sú biele môžeme editovať a riadky sivej farby sa editovať nedajú. Vytvoríme jednoduchý kód pre vygenerovanie priebehu signálu sin(x) v intervale od 0 do 2pi s krokom 0.01 a tento signál zobrazíme do grafu. Na rozdiel od skriptu, resp. programovania v prostredí GUIDE, ak používame AppDesigner je vždy potrebné zadefinovať, to v ktorom grafe sa budú údaje zobrazovať. V našom prípade to bude ***app.Graf***. V obsluhe tohto tlačidla tiež môžeme naprogramovať aj funkcionalitu zobrazovania štatistických parametrov signálu, pre lepšie zobrazenie hodnôt môžeme vypočítané parametre zaokrúhliť na 3 desatinné miesta. Všimnime si z Obr. 6, že hodnotu štatistického parametru do textového poľa zobrazujeme tak, že nastavíme atribút ***Value*** *(napríklad:* ***app.Priemer.Value = 10****)*. Toto je podobné ako to bolo v nástroji GUIDE, akurát nie je potrebné použiť funkciu ***set()***.

```
30        function ZobrazitButtonPushed(app, event)
31            t = 0:0.01:2*pi;
32            ft= sin(t);
33            plot(app.Graf,t,ft);
34            xlabel(app.Graf,'Čas [s]');
35            ylabel(app.Graf,'f(t)');
36
37            disperzia = std(ft)^2;
38            stredna_hodnota = mean(ft);
39            modus = mode(ft);
40            med = median(ft);
41            app.Priemer.Value = num2str(round(stredna_hodnota*1000)/1000);
42            app.Disperzia.Value = num2str(round(disperzia*1000)/1000);
43            app.Modus.Value = num2str(round(modus*1000)/1000);
44            app.Median.Value = num2str(round(med*1000)/1000);
45
46        end
```

**Obr. 6 Zdrojový kód pre obsluhu tlačidla „Zobraziť priebeh“**

Po stlačení tlačidla by sa mal zobraziť priebeh signálu do grafu a tiež by sa mali zobraziť štatistické parametre signálu.

**CallBack pre tlačidlo „Načítaj:**

Rovnako, ako v predchádzajúcom príklade je potrebné najprv vytvoriť callback a v ňom editovať kód. Pre načítanie viacerých súborov môžeme použiť príkaz ***uigetfile***. Načítané názvy súborov rovno môžeme uložiť ako zoznam do listboxu ***app.Subory***. Tento kód funguje pre výber viacerých súborov súčasne, pre výber iba jedného súboru fungovať nebude (viac v podkapitole „Na zamyslenie“).

```
69          function NacitatButtonPushed(app, event)
70              [F,P]=uigetfile('*.csv', 'MultiSelect', 'on');
71              app.Subory.Items=F;
72          end
```

**Obr. 7 Zdrojový kód pre obsluhu tlačidla „Načítaj súbor“**

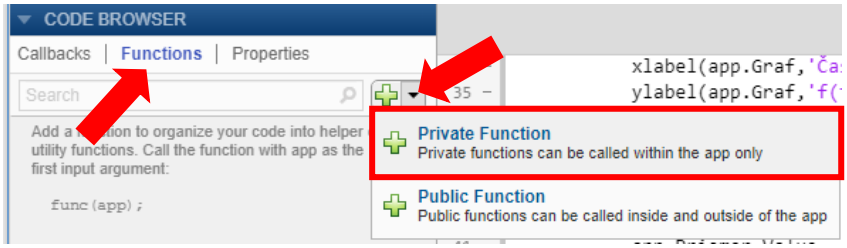
Môžeme si všimnúť nevyužitú premennú ***P***. Do tejto premennej sa uloží celá cesta k súborom, ktoré sme vybrali. Zatiaľ budeme uvažovať, že súbory sú uložené v priečinku ako je uložený program, ale neskôr je program vhodné rozšíriť o možnosť vybrať súbory z ľubovoľného priečinka. Vtedy cestu k súborom budeme potrebovať.

*Pozn. Úplnú cestu k súboru je možné vygenerovať s využitím funkcii strcat() alebo sprintf(), tiež je možné využiť príkaz fullfile(). Úplná cesta potom bude mať nasledovný tvar: „C:\priecinok1\priecinok2\subor.csv*

**Funkcia pre importovanie dát:**

V predchádzajúcom texte sme dáta pre zobrazovanie definovali priamo v zdrojovom kóde tlačidla. Toto bolo len dočasné riešenie pre „oživenie“ tlačidla. Pokiaľ chceme pracovať s dátami, ktoré sú uložené v súboroch, je potrebné napísať vlastnú funkciu pre import. Postupujeme nasledovne:

a.) Vytvoríme privátnu funkciu pre importovanie dát: môžeme ju nazvať napríklad ***importCSV***. Funkcia nemusí mať, žiadne výstupné premenné pretože importované dáta uložíme priamo do štruktúry/objektu app.




**Obr. 8 Postup vloženia privátnej funkcie do kódu**

Samotný import dát zo súboru je možné vykonať pomocou funkcie ***readtable()***. Takto sa dáta zo súboru načítajú priamo do údajového typu tabuľka. Tieto údaje je potom možné previesť na údajový typ double a vykresliť ich. Premenné ***t*** (čas) a funkčné hodnoty - ***ft*** sú uložené v druhom a treťom stĺpci tabuľky. Celá funkcia aj s uložením údajov do premenných, ktorú je možné použiť v rámci celého programu je uvedená na Obr. 10. Pre zavedenie premenných ***app.ft*** a ***app.t*** je potrebné vytvoriť privátne premenné celého programu, podobne ako sme vytvorili privátnu funkciu.


**Obr. 9 Postup vloženia privátnej premennej do kódu**

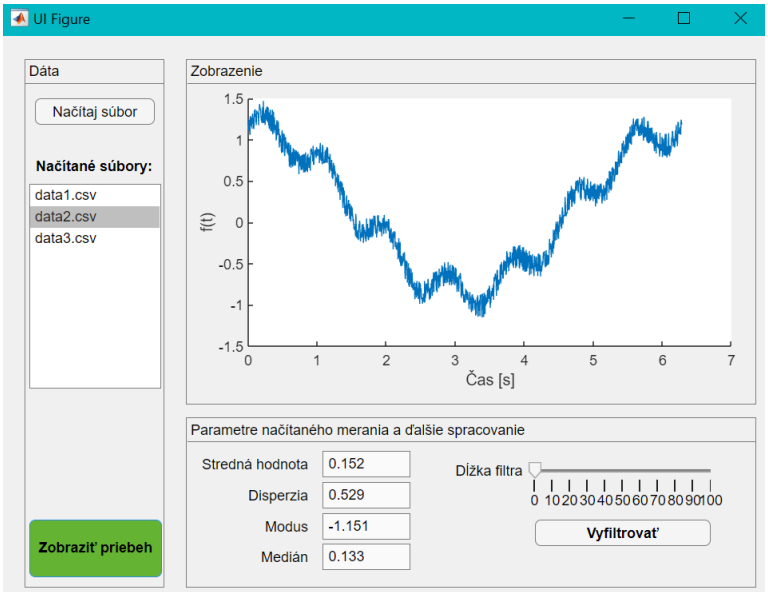
Upravené kódy pre ***properties***, funkciu ***importCSV*** a tlačidlo „Zobraziť“ sú nasledovné:

```
4   properties (Access = public) ...
30
31  properties (Access = private)
32      t;
33      ft;
34      ft_filt;
35  end
36
37  methods (Access = private)
38
39      function importCSV(app)
40          A = readtable(app.Subory.Value);
41          app.t = table2array( A(:,2) );
42          app.ft= table2array( A(:,3) );
43      end
44  end
```

```
47  methods (Access = private)
48
49      % Button pushed function: Zobrazit
50      function ZobrazitButtonPushed(app, event)
51          app.importCSV;
52          plot(app.Graf,app.t,app.ft);
53          xlabel(app.Graf,'Čas [s]');
54          ylabel(app.Graf,'f(t)');
55          disperzia = std(app.ft)^2;
56          stredna_hodnota = mean(app.ft);
57          modus = mode(app.ft);
58          med = median(app.ft);
59          app.Priemer.Value = num2str(round(stredna_hodnota*1000)/1000);
60          app.Disperzia.Value = num2str(round(disperzia*1000)/1000);
61          app.Modus.Value = num2str(round(modus*1000)/1000);
62          app.Median.Value = num2str(round(med*1000)/1000);
63
64      end
65
66      % Button pushed function: Nacitat
67      function NacitatButtonPushed(app, event)
68          [F,P]=uigetfile('*.csv', 'MultiSelect', 'on');
69          app.Subory.Items=F;
70      end
```

**Obr. 10 Kompletný zdrojový kód pre načítanie údajov z CSV súborov a ich následné zobrazenie**

V tomto momente by program mal byť aspoň čiastočne funkčný. Mal by umožňovať načítať viacero súborov, zobraziť ich názvy, po vybratí jedného zo súborov a stlačení tlačidla „Zobraziť“ by mal vykresliť zvolený priebeh.

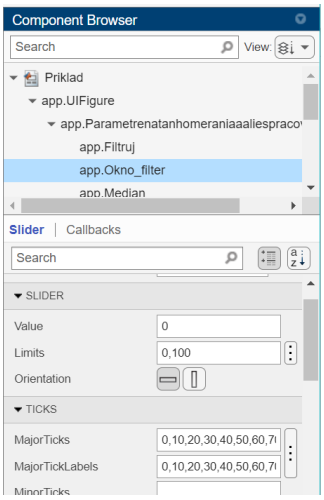



**Obr. 11 Náhľad čiastočne funkčného programu**

## Filtrácia dát:

Poslednou funkcionalitou, ktorú je potrebne naprogramovať je vyhladzovanie dát pomocou spriemerňovacieho filtra. Spriemerňovací filter sa tiež nazýva aj pohyblivý priemer. Filter funguje tak, že sa vypočíta konvolúcia medzi vektorom dát a impulznou odpoveďou filtra, ktorá má dĺžku N prvkov a všetky prvky majú rovnakú hodnotu 1/N. Na samotnú filtráciu môžeme použiť funkciu ***conv()***.

Tiež je potrebné upraviť parametre slideru, ktorým sa nastavuje dĺžka filtra – musí to byť celé číslo. V „Component browser“ editujeme príslušný prvok a vymažeme „MinorTicks“ a v kóde načítanú hodnotu slidera zaokrúhlime pomocou funkcie ***round()***.




**Obr. 12 Nastavenie vlastnosti slideru**

V kóde môžeme pridať novú privátnu premennú programu ***ft_filt***, tiež pridáme callback pre tlačidlo „Vyfiltrovať“. Zdrojové kódy sú zobrazené na Obr. 13. Vidíme, že v callbacku tlačidla pre filtráciu je načítaná aktuálna hodnota slidera, ktorá je zaokrúhlená na celé číslo. Následne je vygenerovaná postupnosť členov impulznej odpovede filtra. Filtrácia je vykonaná pomocou funkcie ***conv()***. Je známe, že konvolúcia predlžuje postupnosť o N-1 prvkov. Preto je potrebné konvolúciu vykonať s parametrom ***'same'***, čím sa zabezpečí to, že vyfiltrovaný vektor ***ft_filt*** bude mať rovnakú dĺžku ako pôvodný vektor ***ft***. Nakoniec je možné zobraziť filtrovaný priebeh.

```
classdef Priklad < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public) ...

    properties (Access = private)
        t;
        ft;
        ft_filt;
    end
```

```
        % Button pushed function: Filtruj
        function FiltrujButtonPushed(app, event)

            L = round(app.Okno_filter.Value);
            b = ones(L,1)/L;


            app.ft_filt = conv(app.ft,b,'same');
            plot(app.Graf,app.t,app.ft_filt);
        end
    end
```

**Obr. 13 Zdrojové kódy pre privátne premenné a obsluhu tlačidla „Vyfiltrovať“**

Po týchto úpravách kódu, je možné program považovať za hotový. Netreba však zabúdať, že je to hrubá verzia programu, ktorá obsahuje veľa chýb, ktoré je možné odstrániť len ďalším ladením.

### 1.1.3. Na zamyslenie (Ladenie a dokončenie programu)

- Čo sa stane ak používateľ vyberie iba jeden CSV súbor? Ako to napraviť?
- Čo sa stane ak sa stlačí tlačidlo „Zobraziť priebeh“ bez toho, aby boli načítané dáta? Ako to napraviť?
- Čo sa stane ak sa stlačí tlačidlo „Vyfiltruj“ bez toho aby boli načítané dáta? Ako to napraviť?
- Čo sa stane ak sa pokúsime načítať súbory z iného priečinka ako z priečinka kde sa nachádza program?

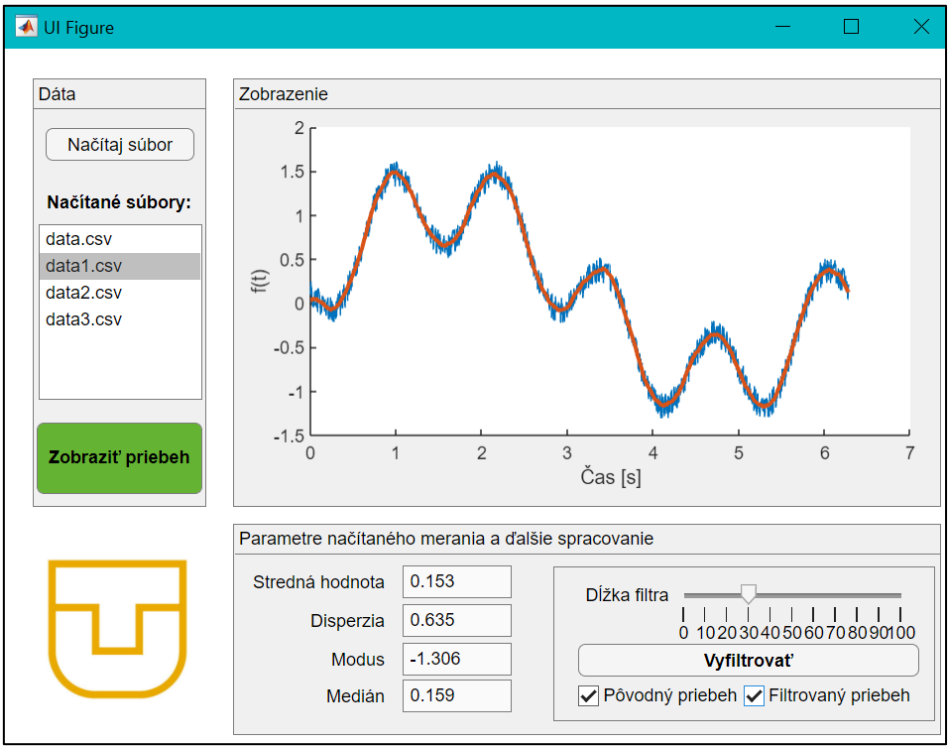
Predchádzajúce otázky smerujú k takzvanej „Blbuvzdornosti“. Pri vývoji programu je potrebné vždy myslieť na rôzne neštandardné situácie, ktoré môžu nastať. Je potrebné počítať s tým, že používateľ nebude študovať dokumentáciu programu, namiesto toho bude chaoticky klikať a skúšať rôzne možnosti.

### 1.1.4. Doplňujúce úlohy

**Úloha 1.:** Rozšírte program o možnosť vybrať, aký graf sa bude zobrazovať (pomocou checkboxov): Graf sa bude meniť bez toho, aby bolo opätovne použité tlačidlo „Zobraziť priebeh“.

- V jednom grafe bude pôvodný priebeh aj filtrovaný priebeh.
- V grafe bude iba pôvodný priebeh.
- V grafe bude iba filtrovaný priebeh.

**Úloha 2.:** Pridajte logo Technickej univerzity.




**Obr. 14 Náhľad finálneho programu s funkcionalitou pre voľbu zobrazeného priebehu a s logom Technickej univerzity v Košiciach**

**Príloha:** Skript pre vygenerovanie štyroch testovacích CSV súborov.

```
% Generovanie CSV súborov, ktoré obsahujú priebehy multiharmonických
% signálov s pridaným šumom
% Predmet: UDM
%
% Technická univerzita v Košiciach
% Fakulta elektrotechniky a informatiky
% Katedra technológií v elektronike
% (c) Ondrej Kováč 2023
%-------------------------------------------------------------------------%
clear; clc;
t = 0:0.005:2*pi;
no = 1:length(t);
n = 0.3*rand([1 length(t)]);

s1 = sin(t)+0.2*sin(7*t)+n;
TABLE1 = table(no',t',s1', 'VariableNames',{'No','Time','Value'});
writetable(TABLE1,'data1.csv')

s2 = sin(2*t)+0.2*sin(7*t)+n;
TABLE1 = table(no',t',s2', 'VariableNames',{'No','Time','Value'});
writetable(TABLE1,'data2.csv')

s3 = cos(2*t)+0.2*sin(7*t)+n;
TABLE1 = table(no',t',s3', 'VariableNames',{'No','Time','Value'});
writetable(TABLE1,'data3.csv')

s4 = cos(t)+0.2*sin(5*t)+n;
TABLE1 = table(no',t',s4', 'VariableNames',{'No','Time','Value'});
writetable(TABLE1,'data4.csv')
```