ČVUT V PRAZE, FAKULTA STROJNÍ

Ú12110 - Ústav přístrojové a řídící techniky

Snímání signálu pomocí karty Labjack

Python pro vědecké účely

Vyučující
doc. Ivo Bukovský

roč. 5 kr. 11
akademický rok 2013/2014

Jiří Jenč
jiri.jenc@hotmail.com
Jan Mikšík
jan.miksik@gmail.com

Obsah

1. Funkce programu

Program umožňuje vykreslit průběh signálu z jednoho ze 4 analogových vstupů měřicí karty Labjack. Umožňuje nastavit parametry snímání (frekcence a čas), zobrazuje maximum a minimum signálu. Program rovněž umožňuje export do textového dokumentu nebo naopak jeho import. K samotnému spuštění není měřicí karta Labjack vyžadována - v takovém případě program funguje jako vykreslovač grafů imortovaných z txt.

2. Specifikace hardwaru

2.1. Labjack U3-HV

Pro danou úlohu jsme použili Labjack U3-HV.

Použili jsme analogové vstupy AIN0 – AIN3 a GND.

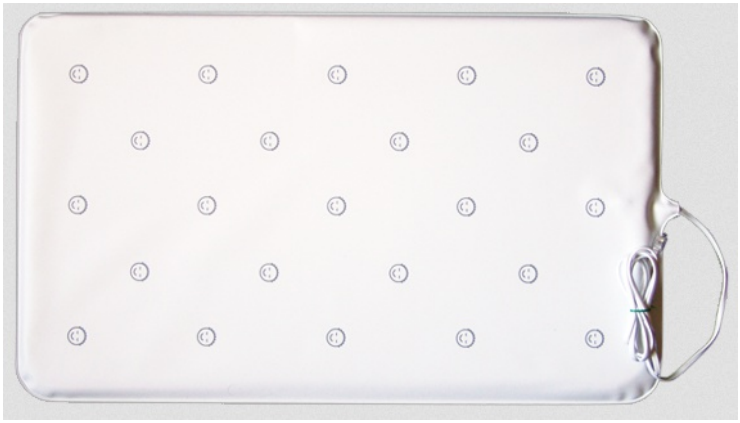
Pro čtení a nastavení frekvence snímání, jsme nepoužili vnitřní timer, ale softwarově vytvořený timer v pythonu.

Obr. 1: Labjack U3-HV

2.2. Monitor dechu Jablotron BM-02

Snímací deska pro monitorování dechu kojence. Jedná se o část systému na předcházení „syndromu náhlého úmrtí kojence“ vlivem zástavy dýchání. V tomto systému je i zařízení pro rozpoznání rizikových stavů.

V naší aplikaci jsme použili pouze snímací desku s obnaženými vodiči místo konektoru. Deska má výstupní vodiče pro signál a zem ty jsme zapojili do GND a druhý do AIN v Labjacku.



Obr. 2: Jablotron BM-02D

3. Specifikace softwaru

3.1. Python 2.7.5

Pro programování jsme použili interpretovaný programovací jazyk Python ve verzi 2.7.5.

Protože jsme, ale používali přídavný hardware a obecně si nemohli vystačit se základními funkcemi pythonu, museli jsme nainsatalovat doplňkové programy, nebo nadstavby pythonu.

3.2. wxPython

wxPython je kombinace Pyhtonu s GUI knihovnou wxWindow, která je multiplatformní. Pro wxPython existuje vývojové prostředí wxGlade, ve kterém jsme vytvořili GUI pomocí nástrojů (viz. dále) a wxGlade poté vygeneroval automaticky kód, který jsme upravili pro naše potřeby.

3.3. LabJack Python

Pro komunikaci s LabJackem slouží LabJackPython, který se musí nainstalovat k instalaci Pythonu. Je jednoduše dostupný ze stránek LabJack.com

LabJack Python podporuje pouze python 2.X, verze 3+ není podporovaná. Podporuje ale všechny platformy (Windows, Linux, macOS).

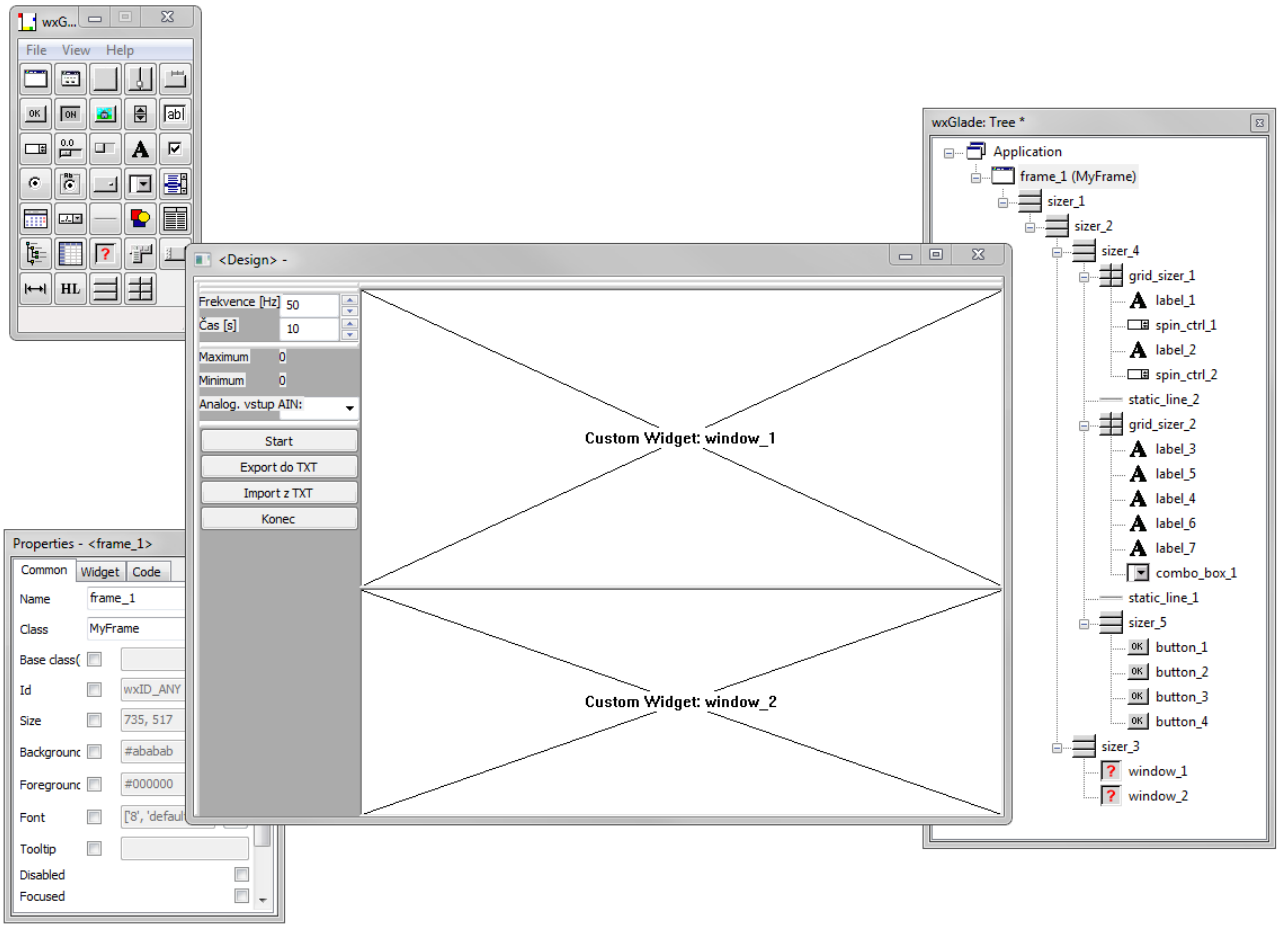
Pro správnou funkci také potřebujeme LabJackDriver, dostupný z: http://labjack.com/support/windows-ud, kde jsme stáhli a nainstalovali příslušný driver.

4. Popis programu

4.1. Práce s wxGlade

V programu si pomocí size-rů rozdělíme okno na požadovaný počet částí, kdy poté do každé části můžeme vložit jeden prvek (textové pole, menu, graf atd.). Program poté sám vygeneruje zdrojový soubor v příslušném jazyku (v našem případě tedy něco.py).

Pro správné fungování si wxGlade importuje některé knihovny, ale některé musíme naimportovat sami (viz. 4.2)



Obr. 3: wxGlade prostředí s otevřeným GUI našeho programu

wxGlade funuje na principu rozdělování okna na rámce (pomocí sizerů).

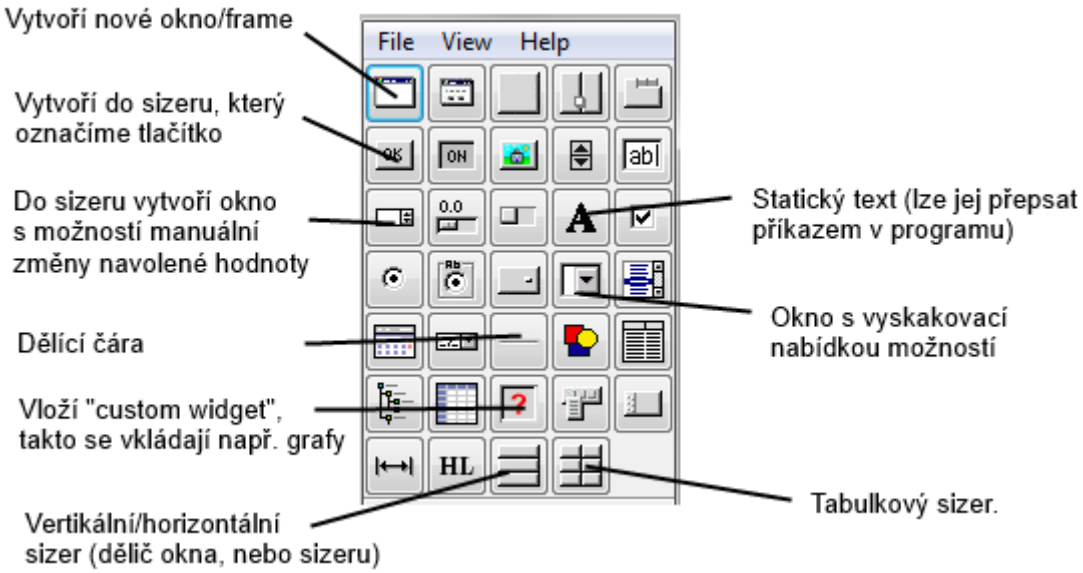
4.1.1. Popis tvorby GUI

Základní GUI jsme rozdělili pomocí horizontálního **sizer_2** na dvě pole.

Do pravého jsme vložili vertikální **sizer_3** o 2 polích, kterému jsme změnili proporci (viz. *Obr. 4*) na 4. Do druhého okna jsme vložili vertikální **sizer_4** o 5 polích, kterému jsme defaultní hodnoty neměnili.

Pomocí proporcí (*proportion*), lze měnit velikost jednotlivých částí GUI. Jedná se ale o proporcionální velikost, která je závislá na velikosti ostatních částí GUI. Tuto hodnotu má každý objekt, který umístíme do GUI.

Popis panelu nástrojů



Graf

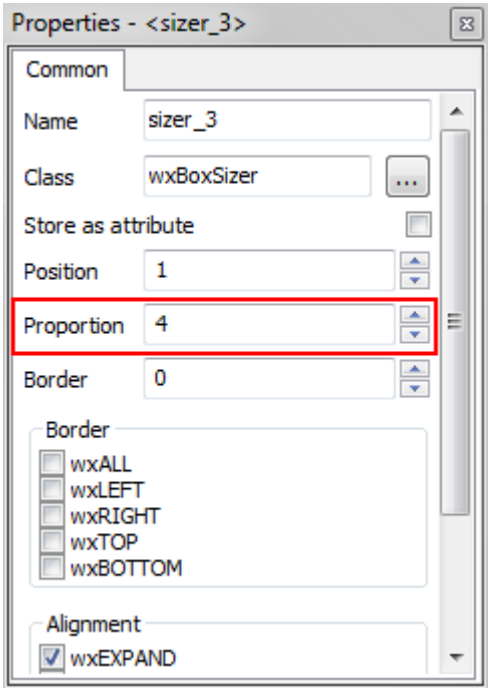
Graf s ovládacími prvky je umístěný v sizer_3.

Graf je umístěn na pozici 0 (horní pozice – označení window_1) a ovládací prvky grafu na pozici 1 (dolní pozice – označení window_2).

Proporci window_2 změníme na 0, aby se přizpůsobilo svojí velikosti a graf zabíral celou pravou stranu i na výšku.

Graf se vkládá pomocí „custom widgetu“, který se po vygenerování kódu v pythonu upraví pro zobrazení grafu ve window_1 a ovladačů ve window_2 takto:

```
self.figure = Figure()
        self.axes =
self.figure.add_subplot(111)
        self.window_1 = FigureCanvas(self, -1,
self.figure)
        self.window_2 =
NavigationToolbar2Wx(self.window_1)
```



Obr. 4: Volba proporce u sizer_3

Ovládací panel

Ovládací panel jsme vytvořili pomocí vertikálního **sizer_4**, kde v horním poli je použit **grid_sizer_1** (2x2), uprostřed **grid_sizer_2** (2x3) a nakonec **sizer_5** (4 pole), vzájemně jsou odděleny pomocí **static_line_1** a **2**.

Horizontální proporce jsou 0:0 v případě tabulkových sizerů a veškeré objekty jsou zarovnány do plné šířky příslušného sizeru pomocí ***wxEXPAND***.

Vstupní hodnoty pro frekvenci a čas jsou vytvořeny pomocí **spin_ctrl_1, spin_ctrl_2 a combo_box_1.**

Tlačítka jsou jednoduše tlačítka (**button_#**) a texty jsou statické texty (**static_text_#**)

prvek	změněné vlastnosti (properties)
spin_ctrl_1	proportion = 0 wxEXPAND range = 1,200 value = 50
spin_ctrl_2	proportion = 0 wxEXPAND range = 1,1000 value = 10
combo_box_1	proportion = 0 wxEXPAND wxCB_DROPDOWN (zobrazí vyjíždějící seznam možností) wxCB_READONLY (znemožní přepisování) Choces (Added): 0 1 2 3 Selection = 0
button_#	Label = pozadovany text bez diakritiky wxEXPAND EVT_BUTTON = název funkce při kliknutí
static_text_#	Label = pozadovany text bez diakritiky

4.1.2. Generování kódu

Zdrojový kód se vygeneruje označením „Application“ v okně stromu („Tree“), kde se v okně vlastnostní („Properties“) vybere jazyk, verze a cílový soubor pro uložení (v našem případě python 2.8 a uloží se do *.py)

> Při pojmenovávání jednotlivých prvků ve wxGlade se při použití diakritiky vyskytla chyba kódování. Je možné, že to lze vyřešit pomocí nastavení wxGlade, ale jinak je možné to řešit, až při psaní samotného programu.

4.1.3. Základní orientace ve zdrojovém kódu wxGlade

wxGlade na začátku programu automaticky vygeneroval kódování (CP1250) a podepsal se.

Ostatní kód který je vygenerovaný wxGlade a je určen k určitým změnám je vždy mezi komentáři:

```
# begin wxGlade: MyFrame._něco_
# end wxGlade
```

Mezi těmito komentáři se nachází obsah, který editujeme. Tento obsah budeme v tomto textu identifikovat pomocí odkazů ***MyFrame***

MyFrame.__init__

Zde jsou nastaveny počáteční hodnoty jednotlivých grafických objektů.

V této části nás zajímají hlavně prvky, které mají takovouto stavbu:

```
self.{název objektu}=wx.{druh prvku}({vlastnosti})
```

MyFrame.__set_properties

Zde je pouze nadpis okna a výchozí hodnota v combo_box_1

MyFrame.__do_layout

Zde, je od spodu vytvořen strom hierarchie jednotlivých sizerů a jejich obsahu

```
                sizer_2.Add(sizer_3, 4, wx.EXPAND, 0)
            sizer_1.Add(sizer_2, 1, wx.EXPAND, 0)
        self.SetSizer(sizer_1)
```

Z kousku kódu je vidět, že sizer 1 má v sobě sizer 2, s proporcí 1 vlastností wxEXPAND a ohraničením 0 resp.: „sizer_1.Add(**obsah, proporce, vlastnosti, ohraničení**)“.

MyFrame.<event_handler>

Zde jsou funkce pro interakci s GUI (zmáčknutí tlačítka apod.). Nahrazujeme část kódu print |"Event handler 'funkce' not implemented!"|

4.2. Použité knihovny (importy)

Knihovny které budeme potřebovat pro běh programu:

4.2.1. wxGlade

wxGlade samotný automaticky vygeneroval import dvou knihoven:

```
import wx
import gettext
```

k nim přidáme pro správné vykreslení grafů a nástroje grafu v GUI tyto:

```
import matplotlib
matplotlib.use('WXAgg')
from matplotlib.backends.backend_wxagg import FigureCanvasWxAgg as
FigureCanvas
from matplotlib.backends.backend_wx import NavigationToolbar2Wx
from matplotlib.figure import Figure
```

4.2.2. LabJack

Pro spojení programu s Labjackem

```
import u3
import LabJackPython
```

4.2.3. Vlastní funkce programu

Import funkce pro vyčkání

```
from time import sleep
```

Import funkcí pro graf, které nebyly importovány výše

```
from matplotlib.pyplot import plot, show
```

Import funkcí pro čtení a zapisování do *.txt souborů a tvorbu nulových matic

```
import os
from numpy import savetxt, zeros, loadtxt
```

4.3. Úprava GUI

Po vytvoření základního kódu musíme upravit GUI pro správné zobrazení grafů.

4.3.1. Grafy

Pro správné vykreslení grafů, musíme nahradit původní kód v MyFrame.__init__ :

```
        self.window_1 = window_1(self, wx.ID_ANY)
        self.window_2 = window_2(self, wx.ID_ANY)
```

novým kódem, kdy definujeme obsah window_1 a window_2

```
        self.figure = Figure()
        self.axes = self.figure.add_subplot(111)
        self.window_1 = FigureCanvas(self, -1, self.figure)
        self.window_2 = NavigationToolbar2Wx(self.window_1)
```

4.4. Programové funkce

4.4.1. Kontrola připojení LabJacku

Kontrolu připojení, jsme provedli pomocí příkazu try.

```
try:
    d = u3.U3()
except:
    print u'Labjack není připojen'
```

Funkcí d=u3.U3() se program připojí automaticky k prvnímu nalezenému připojenému LabJacku (problém by mohl být při používání více LabJacků)

4.4.2. Měření

Měření přidáváme do části kódu MyFrame.<event_handlers>:

Funkce začíná za

```
def start_klik(self, event):  # wxGlade: MyFrame.<event_handler>
```

a končí

```
        event.Skip()
```

Přehled proměnných:

xy	matice (n x 2) pro naměřené hodnoty – **globální** proměnná
t	čas zadaný uživatelem
f	frekvence snímání zadaná uživatelem
n	počet vzorků (n=t*f)
AIN	proměná určující číslo analogového vstupu
dt	čas mezi odečítáním hodnot (dt=1/f) v sekundách
a	naměřená hodnota ze vstupu
maximum	maximální a minimální hodnota z naměřených hodnot
minimum	

definování globální proměnné a čtení z GUI:

```
        global xy
        t=self.spin_ctrl_2.GetValue() #cas mereni s
        f=self.spin_ctrl_1.GetValue() #frekvence Hz
        n = t*f #pocet vzorku
        xy=zeros((n,2))
```

Volba analogového vstupu:

```
        AIN=int(self.combo_box_1.GetValue())
```

kdy poté pomocí příkazu

```
a = d.getAIN(AIN)
```

přisupujeme k měřeným datům příslušného vstupu a zapisujeme pak do matice xy v cyklu for, který má velikost podle počtu vzorků.

Nastavení frekvence měření probíhá pouze softwarově, pomocí cyklu a softwarového pozastavení:

```
            sleep(dt)
```

Maximum a minimum musíme pro vykreslení převést na string:

```
        self.label_5.SetLabel(str(maximum))
        self.label_6.SetLabel(str(minimum))
```

Vykreslení naměřených hodnot

```
        self.axes.cla()
        self.axes.plot(xy[:,1],xy[:,0])

        self.axes.set_xlabel(u'Čas [s]')
        self.axes.set_ylabel('Signal [V]')
        self.axes.grid()
        self.window_1.draw()
```

4.4.3. Export naměřených hodnot

V této funkci pouze používáme globální proměnnou xy, kterou si beztak musíme definovat.

Uložení a načtení hodnot opět je přiřazené k příslušným tlačítkům a je mezi:

```
def export_klik(self, event):  # wxGlade: MyFrame.<event_handler>
        event.Skip()
```

Pro otevření dialogového okna pro výběr k exportu (stejně jako pro import) složí část:

```
dlg = wx.FileDialog(
            self, message=u"Uložit jako...",
            defaultDir=self.currentDirectory,
            defaultFile="",
            wildcard=wildcard,
            style=wx.SAVE
            )
        if dlg.ShowModal() == wx.ID_OK:
            path = dlg.GetPath()
            print u'Exportovaný soubor: ',path
        dlg.Destroy()
```

self.currentDirectory odkazuje na složku ve které je umístěn soubor

Při ukládání, vkládáme do prvního řádku popis proměnných (header).

Uložené naměřené hodnoty jsou odděleny tabulátorem, resp. novým řádkem.

```
        try:
            savetxt(path ,xy, header='Signal[V]\t\t Cas[s]')
        except UnboundLocalError:
            print u'Export zrušen'
```

4.4.4. Import naměřených hodnot

V této funkci pouze používáme globální proměnnou xy, kterou si beztak musíme definovat.

Nachází se mezi řádky:

```
def import_klik(self, event):  # wxGlade: MyFrame.<event_handler>
        event.Skip()
```

Okno pro volbu importovaného souboru:

```
dlg = wx.FileDialog(
            self, message="Vyberte soubor",
            defaultDir=self.currentDirectory,
            defaultFile="",
            wildcard=wildcard,
            style=wx.OPEN | wx.MULTIPLE | wx.CHANGE_DIR
            )
        if dlg.ShowModal() == wx.ID_OK:
            paths = dlg.GetPaths()
            paths = paths[0]
            print u'Importovaný soubor: ',paths
        dlf.Destroy()
```

Výstupem z dlf.GetPaths() je vektor 1x1 proto musíme udělat paths=paths[0]

Poté se zkusí vykreslit bez prvního řádku, jinak se import neprovede:

```
 try:
            xy=loadtxt(paths,skiprows=1)
        except UnboundLocalError:
            print u'Import zrušen'
```

skiprows=1 přeskočí první řádek

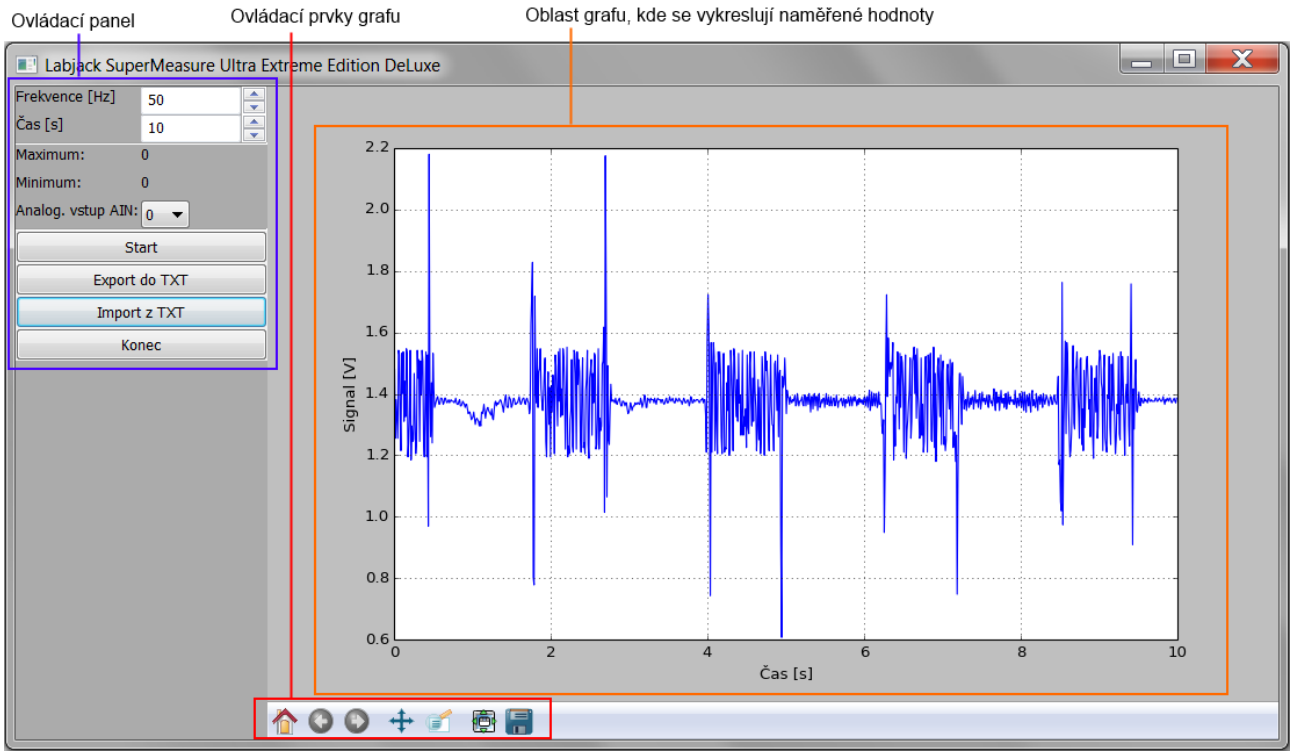
Vykreslení je stejné jako při běžném měření.

4.4.5. Ukončení programu

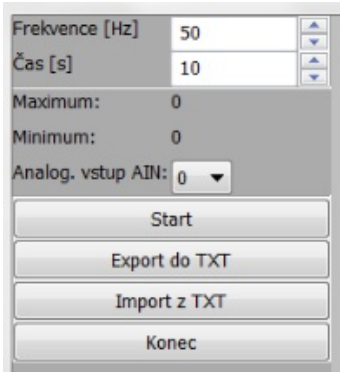
Ukončení na tlačítku s funkcí konec_klik. Musí se vypnout také spojení s LabJackem, jinak python hlásí chybu.

```
        self.Close(True)
        LabJackPython.Close()
```

5. Popis GUI



5.1. Ovládací panel



Ovládací panel se skládá ze tří částí.

V první, která je nahoře, je nastavení vlastností měření. Kdy nastavujeme frekvenci snímání a dobu měření.

V druhé části jsou zobrazena maxima a minima naměřených hodnot a pod nimi volba analogového vstupu LabJacku, z kterého získáváme signál. Vstupy je možné volit v rozsahu AIN0 až AIN3.

Ve třetí části jsou 4 ovládací tlačítka:

- Start

Začne zaznamenávat signál z nastaveného analogového vstupu. Naměřené hodnoty se poté zobrazí v poli grafu.

- Export do TXT

Exportuje hodnoty z grafu do textového souboru ve formě dvou sloupců a přidá jako první řádek popis sloupců.

- Import z TXT

Importuje hodnoty z textového souboru a vykreslí v grafu. Takto načtené hodnoty neovlivní zobrazené maximum a minimum.

- Konec

Zavře program

6. Závěr

Při tvorbě programu jsme se seznámili měřicí kartou Labjak a tvorbou grafických uživatelských rozhraní v modulech wxPython a wxGlade. Samotný program není složitý, nejvíce času zabralo jeho odladění. Samotné programování v Pythonu (a tudíž i předmět 'Python pro vědecké výpočty' považujeme za velice přínosné, jelikož jde o bezplatnou alternativu k programu Matlab.

Vytvořený program vykresluje naměřené hodnoty jednoho z analogových vstupů LabJacku. Může tyto hodnoty exportovat do txt a opět importovat z txt a vykreslit.

Program je to jednoduchý, proto jsme věnovali část této práce popisu práce s wxGlade.

Program trpí jednou chybou, kdy je nastaveno časování odměřování hodnot softwarově, což vytváří zátěž procesoru a při vyšších frekvencích celkový čas měření neodpovídá reálnému času měření.

7. Zdroje

[1] Stránky předmětu :

http://users.fs.cvut.cz/ivo.bukovsky/PVVR/index.htm

[2] LabJackPython :

http://labjack.com/support/labjackpython

[3] Základní příkazy LabJackPythonu:
http://labjack.com/support/labjackpython/low-level

[4] Vlastnosti grafu :

http://matplotlib.org/api/figure_api.html

[5] Dialogová okna :

http://www.blog.pythonlibrary.org/2010/06/26/the-dialogs-of-wxpython-part-1-of-2/