

katalóg požiadaviek + návrh

Technická dokumentácia

Meranie fyzikálnych veličín pomocou
meracieho prístroja

Dáša Keszeghová, Anna Rebeka Sojka,
Matúš Gál, Jakub Švorc

Obsah

Katalóg požiadaviek	3
1. Úvod	4
1.1. Účel katalógu požiadaviek	4
1.2. Rozsah využitia systému	4
1.3. Slovník pojmov	4
1.4. Odkazy	4
1.5. Prehľad nasledujúcich kapitol	4
2. Všeobecný popis.....	5
2.1. Perspektíva produktu	5
2.2. Funkcie produktu.....	5
2.3. Charakteristika používateľov	5
2.4. Všeobecné obmedzenia	5
2.5. Predpoklady a závislosti	5
3. Špecifikácia požiadaviek	6
3.1. Funkčné požiadavky	6
3.1.1. Komunikácia medzi prístrojom a počítačom	6
3.1.2. Voľba meranej veličiny	6
3.1.3. Nastaviteľná perióda	6
3.1.4. Neustála spätná väzba.....	6
3.1.5. Tvorba tabuľky.....	6
3.1.6. Export tabuľky	7
3.1.7. Tvorba grafu	7
3.1.8. Uloženie a načítanie meraní – domáce úlohy	7
3.1.9. Spúšťanie, priebeh, ukončenie merania a odpojenie prístroja	7
3.1.10. Spúšťanie aplikácie	8
3.1.11. Používateľské rozhranie - rozloženie.....	8
3.2. Kvalitatívne požiadavky	8
3.2.1. Návod na inštaláciu	8
3.2.2. Používateľská príručka.....	8
3.3. Požiadavky na používateľské rozhranie.....	8
3.3.1. Žiadna myš – len klávesnica	8
3.3.2. Jednoduché intuitívne menu.....	9
3.3.3. Orientácia tabuľky (v aplikácii aj v Exceli).....	9
3.3.4. Typ grafu.....	9
3.3.5. Celkové grafické prevedenie	9

3.3.6.	Rozlíšenia obrazovky	9
3.4.	Možné rozšírenie aplikácie	9
3.4.1.	Graf – popis a ovládanie	10
3.4.2.	Prezeranie grafu	10
Technická dokumentácia		11
1.	Špecifikácia vonkajších interfejsov	12
1.1.	Komunikácia s inými zariadeniami	12
1.1.1.	Pripojenie prístroja	12
1.1.2.	Komunikácia prístroja s počítačom	12
1.1.3.	Použité technológie	13
2.	Formáty súborov	13
2.1.	Ukladanie meraní	13
2.2.	Exportovanie meraní	14
3.	Návrh používateľského rozhrania.....	15
4.	Návrh implementácie	17
4.0.	Implementácia vnútorného mechanizmu - popis	17
4.1.	Implementácia grafiky – popis	17
4.2.	Component diagram.....	19
4.3.	Class diagram.....	20
4.4.	Sequence diagram	21
4.5.	Popis jednotlivých súborov	22

Katalóg požiadaviek

1. Úvod

1.1.Účel katalógu požiadaviek

Katalóg požiadaviek bude východiskovým materiálom pre implementáciu. Preto obsahuje podrobný popis používateľských a funkčných požiadaviek, ktoré boli spísané na základe osobného stretnutia a dohody medzi zadávateľom a programátormi.

1.2.Rozsah využitia systému

Vytvoriť softvér, ktorý by umožnil komunikáciu medzi multifunkčným fyzikálnym meracím prístrojom MT 1820 a počítačom tak, aby aj zrakovo postihnutí študenti vedeli spracovať získané hodnoty.

1.3.Slovník pojmov

- *Fyzikálna veličina* - vlastnosť alebo stav fyzikálneho objektu (teplota, elektrický prúd, elektrický odpor, elektrické napätie, ...)
- *Multifunkčný merací prístroj* (napr. MT 1820) - zariadenie určené na meranie hodnoty nejakej fyzikálnej veličiny z výberu, ktorý prístroj ponúka
- *NVDA (NonVisual Desktop Access)* – program, pre zrakovo postihnutých, ktorý číta nahlas text z obrazovky

1.4.Odkazy

- manuál pre používateľa MT 1820

<https://www.manualslib.com/manual/856792/Pro-Skit-Mt-1820.html#manual>

1.5.Prehľad nasledujúcich kapitol

Druhá kapitola dokumentu definuje perspektívy aplikácie a uvádza jej hlavné funkcionality. Popisuje možnosti a preferencie čo sa týka používateľa a taktiež píše o obmedzeniach, predpokladoch a závislostiach aplikácie.

Tretia kapitola sa podrobne venuje požiadavkám zo strany zadávateľa – čo sa týka funkčnej, ale aj vizuálnej časti produktu. Tieto požiadavky boli vymenované a preberané pri osobnom stretnutí, aby aplikácia mohla dosiahnuť najlepší výsledok v praxi – byť užívateľsky pohodlná, užitočná na vyučovaní a bola čo najvhodnejšie navrhnutá práve pre žiakov, ktorí ju budú používať.

2. Všeobecný popis

2.1. Perspektíva produktu

Produkt bude (nielen) grafickou aplikáciou, ktorá bude pracovať s nameranými hodnotami na laboratórnej práci na hodine vyučovania fyziky. Bude hlásiť aktuálne zaznamenanú hodnotu, z nameraných hodnôt tvoriť dvojriadkovú tabuľku (čas-číslo) a tiež bude po meraní poskytovať možnosť vykresliť jednoduchý prehľadný graf závislosti meranej veličiny od času.

2.2. Funkcie produktu

- možnosť zvoliť (na prístroji) fyzikálnu veličinu, ktorú žiak meria (teplota, prúd, odpor, napätie)
- nastaviteľná časová perióda, kedy počítač ukladá nameranú hodnotu
- v reálnom čase aktualizovaná nameraná hodnota
- vykresľovanie tabuľky čas-hodnota
- možnosť exportu nameraných hodnôt vo formáte .xlsx
- možnosť vykresliť čiarový graf po dokončení merania

2.3. Charakteristika používateľov

Aplikácia je navrhnutá len pre jedného používateľa. Zadávatelom je pani učiteľka zo školy pre žiakov so zrakovým postihnutím, preto tento softvér bude špeciálne vytvorený tak, aby čo najviac zjednodušil používanie aplikácie pre týchto žiakov. Intuitívne ovládanie po pamäti vyžaduje, aby všetky funkcie boli vždy dostupné a softvér sa nemenil (žiadne pridávanie / odoberanie funkcií apod.).

2.4. Všeobecné obmedzenia

Pre správne fungovanie programu je nevyhnutné pripojenie zariadenia k počítaču. Pri tvorbe aplikácie sme počítali práve s meracím prístrojom typu MT 1820, ktorý v škole štandardne používajú a považujú za najvhodnejší – keďže ide o spoľahlivý a jednoducho ovládateľný multifunkčný merací prístroj.

2.5. Predpoklady a závislosti

- pripojený fyzikálny merací prístroj typu MT 1820
- nainštalovaný Microsoft Office Excel
- pre zrakovo postihnutého žiaka program NVDA – ktorý nahlas číta z obrazovky

3. Špecifikácia požiadaviek

3.1. Funkčné požiadavky

3.1.1. Komunikácia medzi prístrojom a počítačom

Prístroj bude posielať dáta a počítač ich ukladať pre ďalšiu prácu. Aplikácia si bude automaticky detegovať port, v prípade keď sa nepodarí automaticky detegovať port, aplikácia požiada používateľa o zadanie portu ručne.

3.1.2. Voľba meranej veličiny

Žiak bude mať možnosť si nastaviť na prístroji či meria teplotu, prúd, odpor alebo napätie. Aplikácia automaticky zistí nastavenú veličinu z prístroja pri spustení merania a tú si zapamätá.

3.1.3. Nastaviteľná perióda

Žiak bude mať možnosť si zvoliť po akom časovom úseku počítač zaznamená meranú hodnotu. Perióda merania sa bude môcť nastaviť v rozsahu od 1s.

3.1.4. Neustála spätná väzba

Žiak bude môcť v aplikácii sledovať akú hodnotu prístroj nameral v zadanej perióde. Aplikácia žiakovi prečíta zaznamenanú hodnotu vždy po uplynutí časového intervalu. V prípade, že nastavená perióda bude veľmi malá, nebudú sa čítať všetky hodnoty, ale len niektoré, tak aby čítanie hodnôt bolo pre užívateľa zrozumiteľné.

V prípade, že ak žiak nestihne zaregistrovať prečítanú hodnotu, pomocou klávesovej skratky CTRL+R mu prístroj zopakuje naposledy nameranú hodnotu.

3.1.5. Tvorba tabuľky

Už počas merania sa bude v aplikácii tvoriť dvojriadková tabuľka čas-hodnota. Po ukončení merania sa bude dať v tabuľke posúvať pomocou šípiek a nechať si zobrazíť predchádzajúce alebo nasledujúce hodnoty.

3.1.6. Export tabuľky

Žiak bude môcť po meraní exportovať namerané hodnoty do súboru v inom formáte a to konkrétne .xlsx. Z tohto exportovaného súboru sa už dáta načítavať nebudú - na to slúži funkcionality "ukladanie meraní".

3.1.7. Tvorba grafu

Žiak bude môcť po meraní nechať v aplikácii vykresliť čiarový graf závislosti nameraných hodnôt od času. Graf bude mať v aplikácii fixné rozmery, mierka na x-ovej osi bude určená na základe intervalu merania.

3.1.8. Uloženie a načítanie meraní – domáce úlohy

Údaje o meraní budú uložené v súbore s názvom, ktorý zadá žiak pri spustení merania. Žiak bude mať možnosť vrátiť sa k predchádzajúcim meraniam – teda spätne si dať vykresliť graf a zobrazíť namerané hodnoty. Aplikácia si uloží nastavenia jednotlivých meraní.

3.1.9. Spúšťanie, priebeh, ukončenie merania a odpojenie prístroja

3.1.9.1. Spúšťanie a ukončenie merania

Začiatok merania sa bude spúšťať tlačidlom "ŠTART", ktoré bude ovládané klávesnicou. Ak už meranie prebieha, stlačením tohto tlačidla sa meranie ukončí. Keď žiak bude mať dostatočný počet nameraných hodnôt, pomocou tohto tlačidla meranie ukončí.

3.1.9.2. Priebeh merania

Počas merania sa budú hodnoty zaznamenávať, každú zaznamenanú hodnotu aplikácia prečíta a zapíše ju do tabuľky.

3.1.9.3. Odpojenie prístroja

Ak dôjde k odpojeniu prístroja počas merania, tak sa meranie zruší, ale doposiaľ namerané hodnoty zostanú zachované. Po opätovnom zapojení prístroja nebude možné v meraní pokračovať, bude nutné spustiť nové meranie.

3.1.9.4. Zmena meranej veličiny

Ak dôjde k zmene meranej veličiny na prístroji počas merania, tak sa meranie zruší.

3.1.10.Spúšťanie aplikácie

Aplikácia sa bude spúšťať klasicky dvojklikom alebo kombináciou označením ikony a tlačidlom "ENTER", po identifikovaní programom NVDA. Možnosti sa bežne používajú pri práci s operačným systémom Windows.

3.1.11.Používateľské rozhranie - rozloženie

Rozloženie komponentov v okne je navrhnuté tak, aby sa celá plocha dala vizuálne a funkcionálne rozdeliť až na 3 časti (podľa toho čo si používateľ zvolí zobraziť alebo v ktorom stave sa program nachádza):

- Prvá časť bude obsahovať tlačidlá, ktoré je možné v tom momente použiť.
- Druhá časť okna bude miesto vyhradené na tvorbu tabuľky. Tá sa zobrazí v prípade, že práve : prebieha meranie, meranie je ukončené alebo bolo načítané staré meranie.
- Tretia časť je situovaná pod druhou časťou a dá sa zobraziť po ukončení merania alebo načítaní starého merania. Je to plocha obrazovky, kde sa po voľbe tlačidla „vykresliť graf“ zobrazí graf nameraných hodnôt.

3.2.Kvalitatívne požiadavky

3.2.1. Návod na inštaláciu

K programu bude priložený priamy návod pre inštaláciu aplikácie.

3.2.2. Používateľská príručka

Okrem návodu na inštaláciu bude dodaná aj používateľská príručka, ktorá bude stručne popisovať jednotlivé možnosti aplikácie a funkcionality.

3.3.Požiadavky na používateľské rozhranie

3.3.1. Žiadna myš – len klávesnica

Žiaci vôbec nepoužívajú myš, preto musí byť ovládanie pohybu po menu kompletne pomocou klávesnice (ideálne TAB, ENTER a šípky) – sú zvyknutí na klávesnicové skratky.

3.3.2. Jednoduché intuitívne menu

Keďže žiaci sa riadia najmä pamäťou a vlastnou predstavivosťou, menu by malo byť tomuto prispôsobené, teda ikony s jednoznačným popisom, prípadne po zvolení nejakej špecifickej kategórie nastavení sa môže zobrazíť rolovacie menu.

Aplikácia bude na ploche zobrazovať len tie ovládacie prvky, ktoré sa dajú v daný moment použiť.

3.3.3. Orientácia tabuľky (v aplikácii aj v Exceli)

Žiaci sú zvyknutí, že prvý riadok tabuľky obsahuje časový údaj a druhý hodnotu – toto je dobré zachovať.

3.3.4. Typ grafu

Stĺpcový, koláčový graf a iné grafické interpretácie sú pre žiakov iba máťúce. Preto aplikácia bude kresliť jednoduchý čiarový 2D graf, kde na x-ovej osi bude čas a na y-osi budú namerané hodnoty.

3.3.5. Celkové grafické prevedenie

Čím viac farieb a rôznych zbytočných hrán, tým bude pre žiakov orientácia v programe ťažšia. Preto sa bude aplikácia držať takmer úplne čierno-bielej témy. Dôležité hrany budú hrubé a vo vysokom farebnom kontraste voči pozadiu.

3.3.6. Rozlíšenia obrazovky

Aplikácia bude mať fixné rozlíšenie (1080 x 720).

3.4. Možné rozšírenie aplikácie

Softvér bude pripravený aj na pripojenie špecializovaného zariadenia, ktoré mechanickým pohybom ovládača číta grafy pomocou zvukového signálu - to by dodal cvičiaci.

3.4.1. Graf – popis a ovládanie

Aplikácia bude vydávať zvuk príslušnej výšky podľa hodnoty v grafe na danej x-ovej pozícii. Na x-ovej pozícii sa žiak posúva vodorovným posuvným potenciometrom.

3.4.2. Prezeranie grafu

V režime zobrazeného grafu funguje prezeranie grafu cez externé zariadenie automaticky, ak je pripojené, bez potreby akéhokoľvek ovládania v programe.

Technická dokumentácia

1. Špecifikácia vonkajších interfejsov

1.1. Komunikácia s inými zariadeniami

1.1.1. Pripojenie prístroja

Aplikácia komunikuje s multifunkčným meracím prístrojom (MT-1820), ktorý má užívateľ pripojený k počítaču pomocou USB kábla a cez USB port. Aplikácia si bude automaticky detegovať port, v prípade keď sa nepodarí automaticky detegovať port, aplikácia požiada používateľa o zadanie portu ručne.

Prístroj sa pripája pomocou COM portov. V prípade, že COM port nie je nainštalovaný aplikácia si doinštaluje potrebný COM port pomocou inštalera (CP210x_VCP_Windows\CP210xVCPInstaller_x64.exe) .

1.1.2. Komunikácia prístroja s počítačom

Na to aby prístroj mohol komunikovať s počítačom a taktiež aj s aplikáciou musí byť v móde REL/RS232 („Relative Value Measurement mode“), tento mód sa dosiahne podržaním tlačidla RS232/REL na prístroji. V prípade, že tento mód nie je zapnutý počítač nevie komunikovať s prístrojom.

Prístroj komunikuje posielaním 14 bitov pričom každý bit má svoj vlastný význam: (význam jednotlivých bitov je možné nájsť na nasledujúcom odkaze - <https://github.com/drahoslavzan/Proskit-MT1820-Probe/blob/master/proskit.cc>)

Význam jednotlivých bitov:

- 00 - znamienko (+/-)
- 01 – 04 hodnota, ktorú prístroj nameral
- 05 – medzera
- 06 – desatinná čiarka (za ktorú cifru z nameranej hodnoty sa má napísať desatinná čiarka)
- 07 – 08 – príznaky
- 09 – 10 – jednotka, v ktorých prístroj meria (fyzikálna veličina)
- 11 – stupnica na dolnej časti displeja prístroja
- 12 – 13 – nový riadok

Jednotky, v ktorých prístroj meria (09-10 bit):

- UNIT_hFE = 0x0010
- UNIT_mV = 0x4080 (napätie vo voltoch)

- UNIT_V = 0x0080 (napätie v milivoltoch)
- UNIT_Ohm = 0x0020 (odpor v Ohmoch)
- UNIT_kOhm = 0x2020 (odpor v kilo-Ohmoch)
- UNIT_MOhm = 0x1020 (odpor v mega-Ohmoch)
- UNIT_DIODE_V = 0x0480
- UNIT_F = 0x0004 (teplota v stupňoch Fahrenheit)
- UNIT_Hz = 0x0008
- UNIT_DUTY_Hz = 0x0200
- UNIT_C = 0x0002 (teplota v stupňoch Celzia)
- UNIT_uA = 0x8040 (elektrický prúd v mikroampéroch)
- UNIT_mA = 0x4040 (elektrický prúd v miliampéroch)
- UNIT_A = 0x0040 (elektrický prúd v ampéroch)

1.1.3. Použité technológie

- Python – Jadro aplikácie bude napísané v tomto jazyku, kvôli veľkej podpore knižníc a modulov na riešenie projektu.
- wxPython – Modul na tvorbu interface-u a interaktívnej časti aplikácie. Nakoľko zabudovaný pythonovský modul tkInter nepodporuje čítanie obrazovky, je tento nástroj najvhodnejší, aj vzhľadom na fakt, že čítač obrazovky NVDA je napísaný aj pythone.
- NVDA – program na čítanie obsahu obrazovky. Interaktívna aplikácia ktorá tvorí zvukový výstup na opísanie označeného prvku obrazovky alebo na opísanie prvku, na ktorý ukazuje myš počítača. Program používajú žiaci, pre ktorých je výsledok projektu určený. Jeho používanie a prístup k nemu je zadarmo.
- JAWS – ďalší čítač obrazovky, ktorý je tiež používaný žiakmi avšak jeho licencia a prístup k nemu je platený.

2. Formáty súborov

Aplikácia bude pracovať s dvomi formátmi súborov:

2.1. Ukladanie meraní

Merania sa budú dať uložiť vo formáte .pickle, tieto súbory sa budú dať spätne otvoriť pomocou aplikácie.

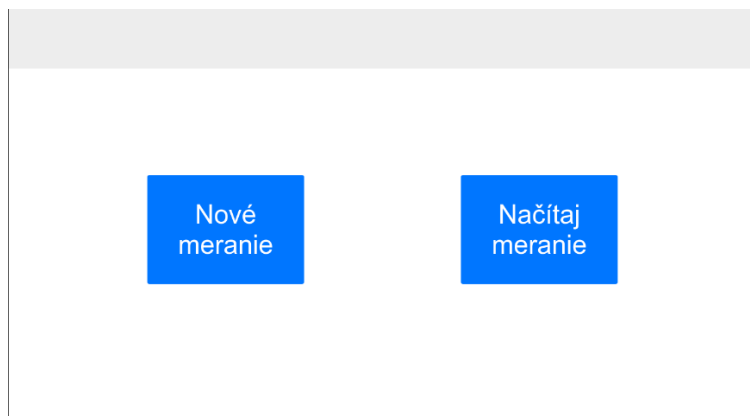
Pickle je modul používaný na serializáciu a deserializáciu pythonovských štruktúr. Pomocou tohto modulu jednoducho skonvertujeme celé meranie na prúd bajtov, ktorý sa dá jednoducho uložiť. Neskôr sa opäť vieme vrátiť do pôvodnej objektovej hierarchie. Na používanie tohto modulu netreba nič inštalovať, stačí použiť import z knižnice – „import pickle“. Na serializáciu budeme používať `pickle.dumps()` a deserializáciu `pickle.loads()`.

2.2. Exportovanie meraní

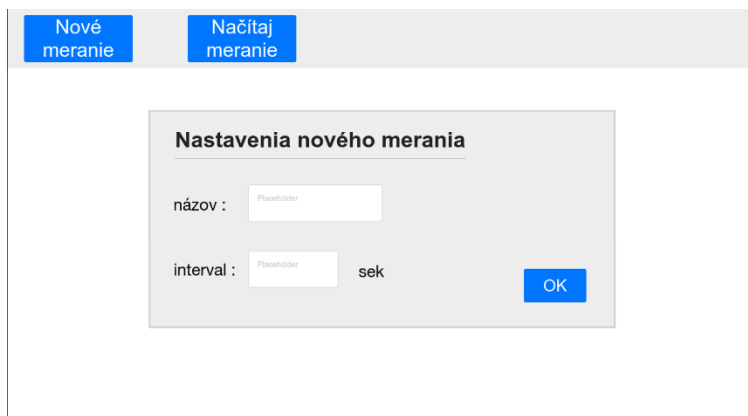
Žiaci si budú môcť exportovať namerané údaje vo forme tabuľky a grafu do Excelu vo formáte .xlsx.

Namerané údaje budú v Exceli zapísane v dvoch riadkoch, pričom prvý bude obsahovať čas a druhý nameranú hodnotu. Graf bude čiarový, na x-ovej osi bude čas a na y-ovej bude nameraná hodnota. Namerané dvojice hodnôt budú vykreslené na grafe a spojené spojovacou čiarou kontrastnej farby.

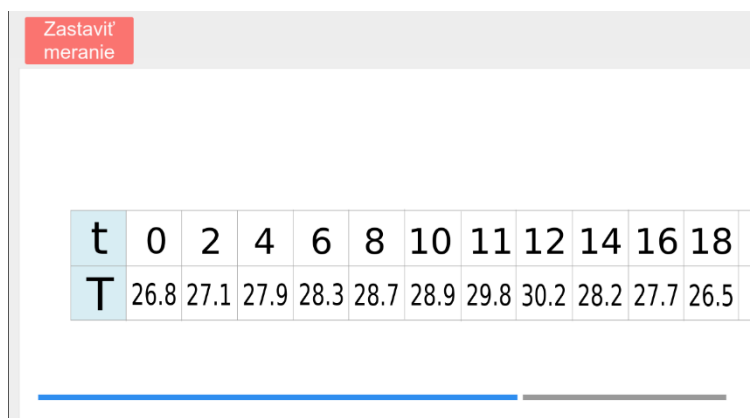
3. Návrh používateľského rozhrania



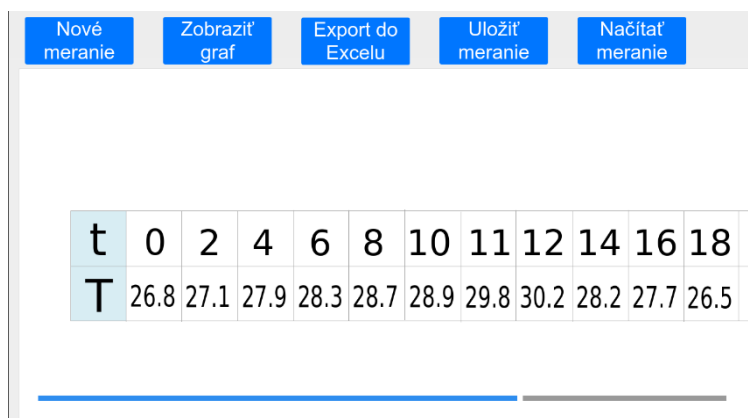
obr. 1 - štart programu



obr. 2 – nové meranie



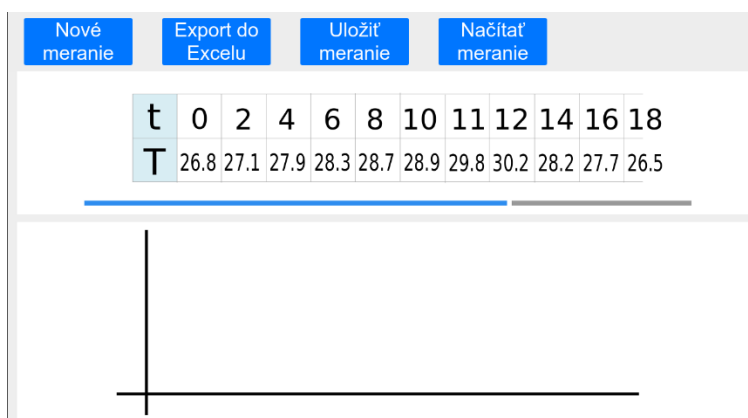
obr. 3 – počas merania



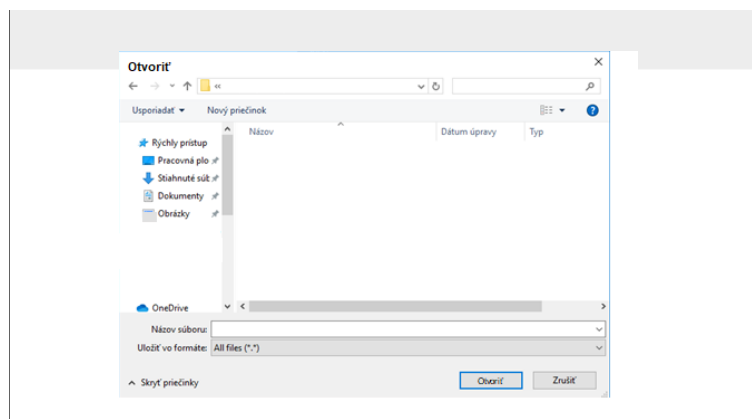
obr. 4 – po meraní / načítané meranie



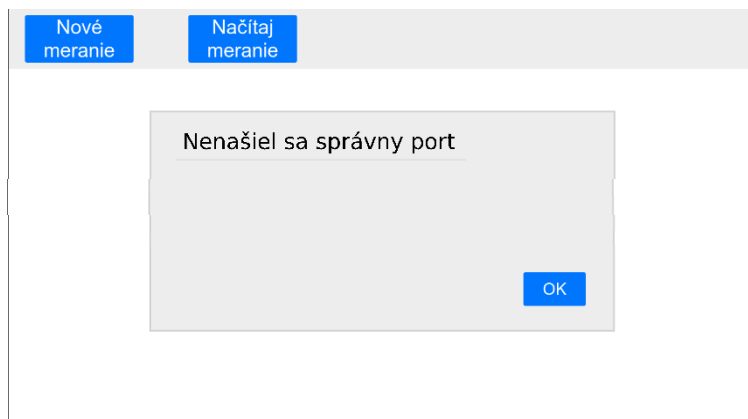
obr. 5 – možnosti exportu



obr. 6 – zobrazenie grafu



obr. 7 – načítanie merania



obr. 9 – chybové hlásenie o porte

4. Návrh implementácie

4.0. Implementácia vnútorného mechanizmu - popis

O správny chod aplikácie sa budú starať nástroje : **Handler**, **Parser** a **Data Manager**.

- **Handler** (naprogramovaný v súbore handler.py) spracúva všetky udalosti prichádzajúce z grafickej plochy, teda čo sa má vykonať : po kliknutí na konkrétne tlačidlo, po stlačení klávesovej skratky apod. Teda udalosť na tlačidle volá Handler a pomocou dohodnutého kódu informuje o aké tlačidlo ide. Rovnako to je aj s udalosťou stlačenia klávesovej skratky. Ďalej má na starosti vyhľadanie portu. V prípade, že sa to nepodarí, podáva hlásenie a pýta správne číslo portu od používateľa.
- **Parser** spracováva a prekladá prichádzajúce dáta z prístroja. Tie posiela priamo Data Manager-u alebo prípadne Handler-u podľa situácie.
- **Data Manager** si pamätá dáta a prijíma pokyny od Handler-a. Namerané hodnoty ukladá, exportuje, ponúka na zobrazenie apod.

4.1. Implementácia grafiky – popis

Grafická časť aplikácie bude rozdelená do ôsmich súborov nasledovne :

- button_panel.py
- start_up.py
- graph_panel.py
- input_panel.py
- messagebox.py
- panel_handler.py
- table_panel.py
- window.py

Každý súbor reprezentuje zobrazenie niektorej časti okna, ktorá je rozdelená na panely. Panely sa zobrazujú v závislosti od stavu okna a kliknutých tlačidiel.

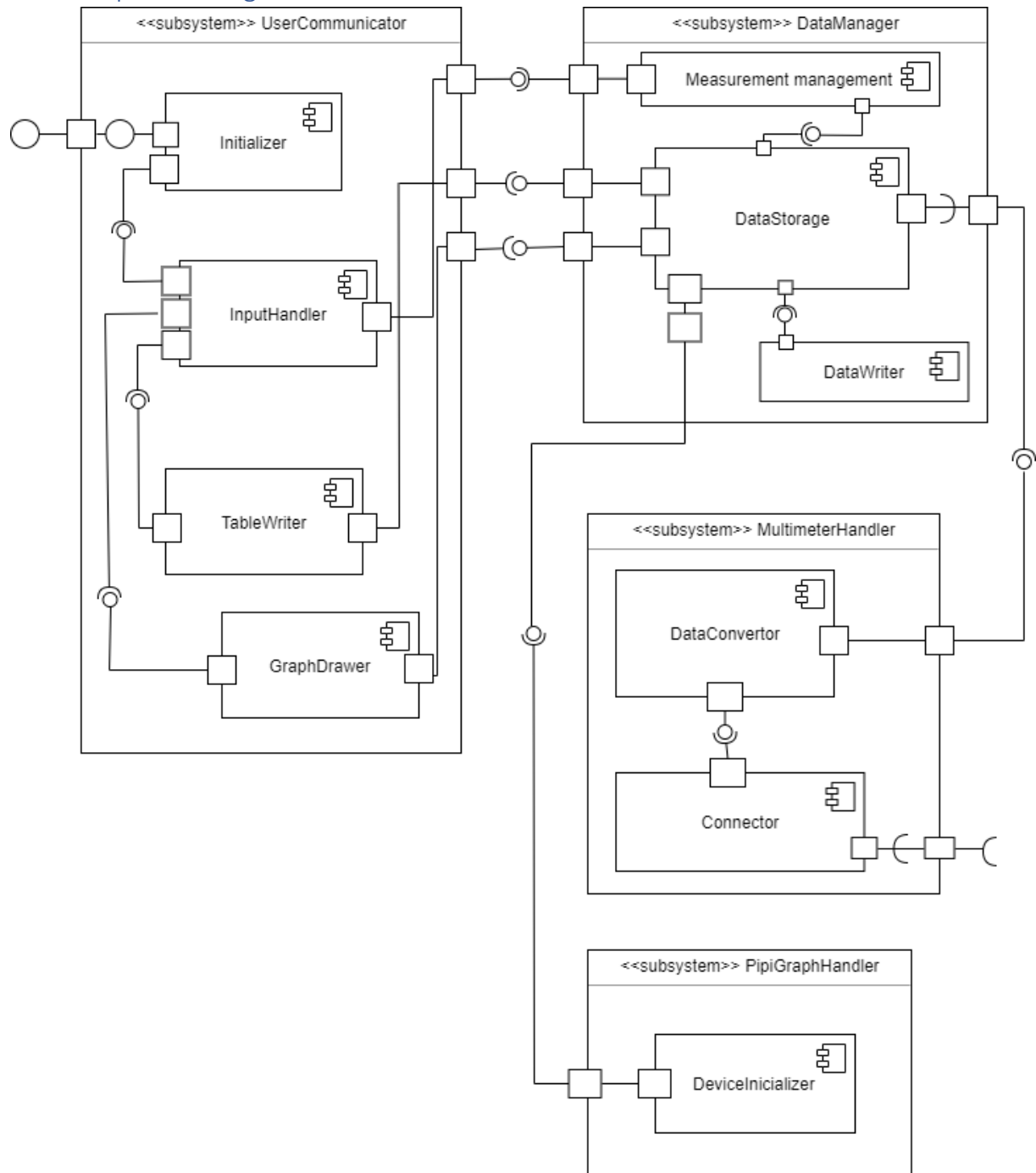
Samotná vykreslovacia plocha – **MainWindow**, využíva funkcie (dedí od) triedy Frame. Rozloženie a vyobrazené prvky v danom frame sa prispôbia akciám užívateľa.

Inicializuje sa jedno okno, ktoré funguje počas celého behu aplikácie. Toto okno sa predáva ako argument, pri výtvarní panelov a ich zobrazovania a iba

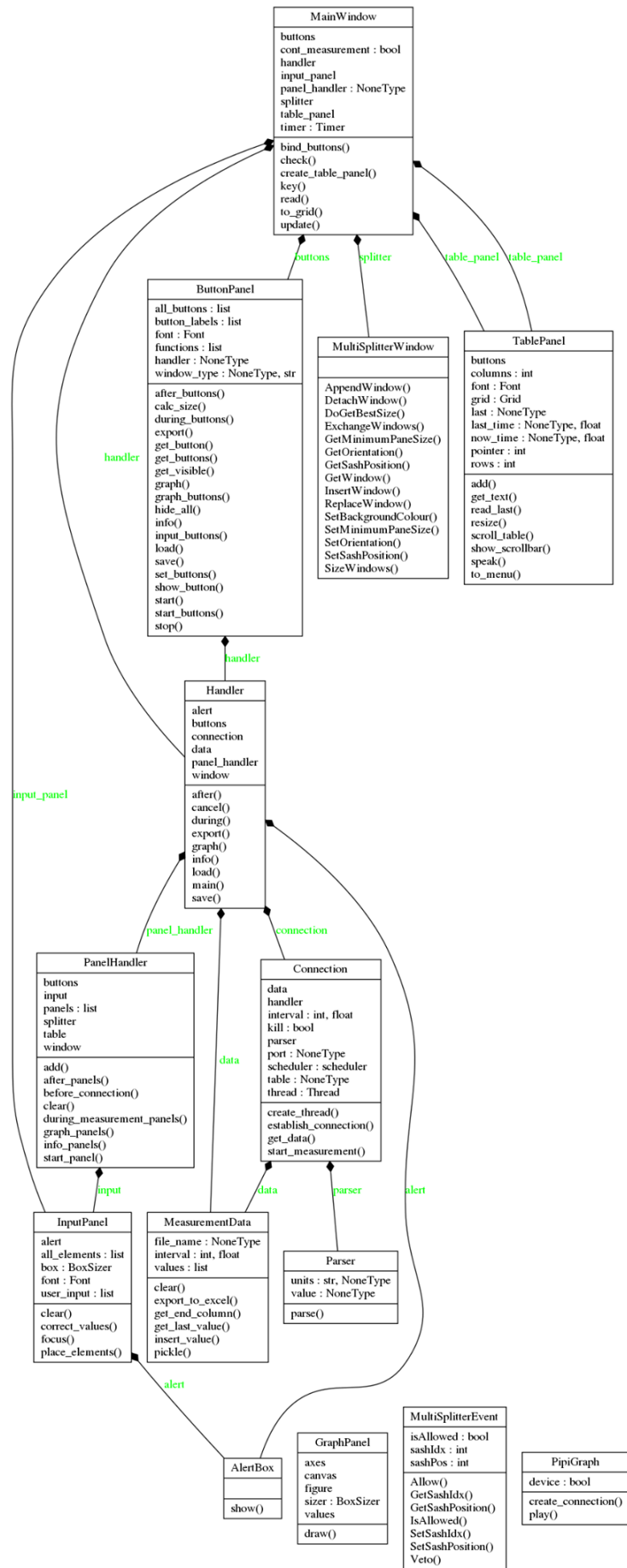
sa prekresľuje obsah tohto okna. Tým pádom sa zabráni duplikovaniu kódu a pre každú obrazovku sa zachová rovnaká funkcionálnosť okna.

Panely sa zobrazujú do Frame-u pomocou triedy typu **Splitter**. Ktorá Frame **horizontálne** rozdelí na viaceré Panely podľa potreby (napr. niekedy sa graf zobrazuje, inokedy nie).

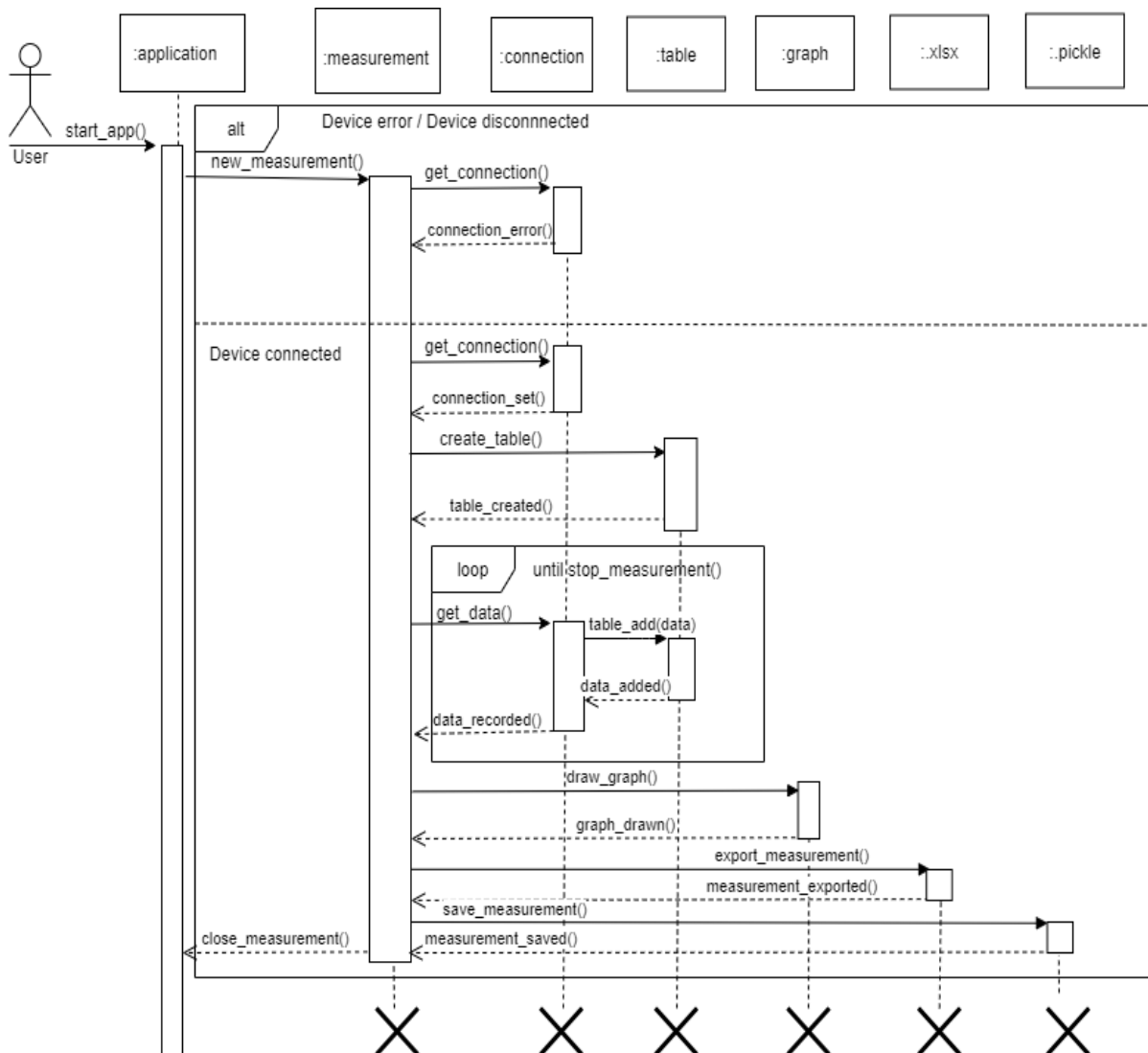
4.2. Component diagram



4.3. Class diagram



4.4. Sequence diagram



4.5. Popis jednotlivých súborov

- **`__init__.py`**

Tento súbor je inicializačný súbor. Slúži ako „entry point“ pri kompilovaní do spúšťateľného exe súboru. Vytvára sa v ňom hlavné okno *window* a inicializuje sa v ňom *handler*, ktorý má na starosti prácu s dátami z prístroja a ich následné spracovávanie.

- **`button_panel.py`**

V súbore sa vytvárajú všetky tlačidlá potrebné počas celého behu aplikácie, ktoré sa neskôr priradia oknu. Teda inicializujú sa iba raz a neskôr sa len prispôsobuje ich pozícia a viditeľnosť. Celú túto funkcionality zastrešuje trieda *ButtonPanel*. Vďaka nej sa vyhneme zbytočnej duplicite kódu a tiež roztrúsenosti funkcií, ktoré spracúvajú udalosti tlačidiel. Takto je to možné nájsť všetko na jednom mieste, pohodlne ku nim pristupovať a medzi sebou ich porovnávať.

Obsahuje rôzne funkcie, ktoré nám zabezpečujú zobrazovanie a skrývanie tlačidiel, ich komunikáciu s handlerom a iné pomocné funkcie, ktoré selektujú konkrétne tlačidlá :

- `__init__(self, parent=None)` – vytvára panel buttonov a triedne premenné
- `set_buttons(self)` – vytvára buttony a nastavuje ich atribúty, všetky *Button* sú zatiaľ nastavené ako skryté
- `get_button(self, text)` – vráti button s labelom, ktorý je rovnaký ako vstupný atribút text
- `show_button(self, text)` – nastaví ako viditeľný button s labelom, ktorý je rovnaký ako vstupný atribút text
- `get_buttons(self)` – vráti zoznam všetkých buttonov (viditeľných aj skrytých)
- `calc_size(element)` – vráti požadovanú veľkosť buttonov vzhľadom na plochu okna
- `calc_size(element)` – vráti požadovanú veľkosť buttonov vzhľadom na plochu okna
- `load(self, event)` – po kliknutí na button „Načítať meranie“ sa v spolupráci s handlerom načíta meranie z pickle súboru
- `info(self, event)` – po kliknutí na button „Nové meranie“ sa spustí *handler.info()*
- `start(self, event)` – po kliknutí na button „OK“ sa spustí *handler.during()*
- `stop(self, event)` – po kliknutí na button „Zastaviť meranie“ sa spustí *handler.cancel()*
- `save(self, event)` – po kliknutí na button „Uložiť meranie“ sa spustí *handler.save()*

- `export(self, event)` – po kliknutí na button „Export do Excelu“ sa spustí `handler.export()`
- `graph(self, event)` – po kliknutí na button „Zobraziť graf“ sa spustí `handler.graph()`
- `get_visible(self)` – vráti zoznam všetkých viditeľných buttonov
- `hide_all(self)` – všetky buttony nastaví ako skryté
- `input_buttons(self)` – nastaví buttony pre stav „Nastavenie nového merania“
- `start_buttons(self)` – nastaví buttony pre stav „Nové meranie“
- `during_buttons(self)` – nastaví buttony pre stav „Počas merania“
- `graph_buttons(self)` – nastaví buttony pre stav „Zobrazený graf“
- `after_buttons(self)` – nastaví buttony pre stav „Po meraní“

- ***connection.py***

Obsahuje triedu *Connection*, ktorá vytvára spojenie s prístrojom, číta údaje z prístroja a spravuje vlákno, v ktorom beží samotná komunikácia prístroja, teda meranie. Dáta z prístroja posielajú Parseru, ktorý bity premení do žiakom známej a zrozumiteľnej podoby. Tieto údaje uskladňuje trieda *MeasurementData* a zaznamenané údaje sú pridávané do tabuľky (*TablePanel*).

Preto sú potrebné metódy :

- `establish_connection(self)` – pokúša sa o spojenie s prístrojom, ak sa to nepodarí vráti *False* inak *True*
- `start_measurement(self)` – vykoná nastavenia pre stav „Počas merania“
- `get_data(self)` – číta a ukladá a pomocou parseru spracúva prichádzajúce hodnoty z prístroja
- `create_thread(self)` – vytvorí vlastný thread pre meranie

- ***data_parser.py***

V tomto súbore trieda *Parser* zastrešuje premenu bitov na hodnotu viditeľnú aj na meracom prístroji. Z prúdu bitov je vyčítaná nameraná hodnota a veličina. Keďže jednotky merania sa môžu počas merania meniť, nameraná hodnota je vždy prepočítaná na základné jednotky. Význam jednotlivých bitov takéhoto bitového prúdu je popísaný v časti 1.1.2.

Táto trieda namerané hodnoty vráti triede *Connection*, ktorá ich následne pošle objektu, ktorý obsluhuje tabuľku a doplní ich tam.

Má len jednu metódu (okrem init-u) `parse(self, bytestream)` : ktorá podľa spomínanej časti 1.1.2. spracuje string dát.

- **graph_panel.py**

V súbore je trieda *GraphPanel*, ktorá vykresľuje z nameraných hodnôt čiarový graf. Namerané hodnoty sa predávajú pri inicializácii triedy *GraphPanel*. Graf je umiestnený pod tabuľkou.

Nájdeme tu 2 metódy :

- `__init__(self, parent, values)` – vytvorí panel, ktorý bude mať ako rodiča druhý argument „parent“ (multisplitter okna) a uloží si hodnoty *values* (čo je dvojrozmerný zoznam hodnôt z prístroja v tvare čas-hodnota)
- `draw(self)` – nakreslí graf a zaznačí doň hodnoty

- **handler.py**

Trieda *Handler* v tomto súbore obsluhuje dopyty od používateľa - klávesové udalosti, stláčanie tlačidiel a klikanie myšou. Okrem toho, má na starosti aj volanie metód, ktoré prispôsobujú obsah okna. Reaguje aj na pripojenie zariadenia, či je pripojené, či sa s ním dá komunikovať a či sa na danom porte nachádza.

Spravuje aj súbory – ich ukladanie, načítavanie a export údajov do excelu.

Taktiež využíva triedu *AlertBox*, pomocou ktorej informuje používateľa o prípadných chybách a vykonaných úkonoch.

Jeho metódy :

- `__init__(self, main_window)` – inicializuje svoje premenné, medzi nimi je *main_window*, čo je okno, vytvorené v súbore `__init__.py`
- `info(self)` – zavolá metódy `window.end()`, `buttons.input_buttons()` a `panel_handler.info_panels()`
- `after(self)` – zavolá metódy `buttons.after_buttons()` a `panel_handler.after_panels()`
- `graph(self)` – zavolá metódy `buttons.graph_buttons()` a `panel_handler.graph_panels()`
- `save(self)` – zavolá metódu `data.pickle()`, ktorú ak sa podarí správne vykonať (vráti *True*), tak vyskočí hlásenie s textom „Meranie bolo uložené!“, inak sa tiež zobrazí hlásenie ale s textom „Meranie sa nepodarilo uložiť!“
- `load(self, *args)` – premaže zobrazené a zobrazí stav „Po meraní“ s načítanými hodnotami z **args*, ktoré ešte spracuje metódou `measurement_data.unpickle(args[0])`
- `def export(self)` – zavolá metódu `data.export_to_excel()`, ktorú ak sa podarí správne vykonať (vráti *True*), tak vyskočí hlásenie s textom

„Údaje boli exportované do excelu!“, inak sa tiež zobrazí hlásenie ale s textom „Údaje sa nepodarilo exportovať!“

- *during(self) – stará sa o správny beh merania a spúšťa metódy pre :
nadviazanie spojenia s prístrojom, prekresľovanie okna, čítanie hodnôt
a ich ukladanie v nastavenom intervale podľa voľby používateľa;
v prípade, že sa nepodari nadviazať spojenie s prístrojom vyvolá
chybové hlásenie s textom „Nepodarilo sa vytvoriť spojenie s
prístrojom!“ a znova obnoví stav „Nastavenie nového merania“*
- *cancel(self, error=False) – v prípade, že nejde o chybu (teda error je
False), tak sa predpokladá, že používateľ klikol na tlačidlo „Zastaviť
meranie“, táto metóda na základe toho predpokladu zavolá vlastnú
metódu self.after(); ak by išlo o chybu, tak sa zariadenie muselo odpojiť
alebo prestať komunikovať s počítačom – vtedy sa vyvolá chybové
hlásenie s textom „Porucha meracieho prístroja“*
- *main(self) – zavolá metódy, buttons.start_buttons() a
panel_handler.start_panels()*

- **input_panel.py**

Reprezentuje úvodnú obrazovku. Trieda *InputPanel* vytvára 2 prázdne Textfieldy, do ktorých užívateľ zadá meno súboru a interval, v ktorom sa majú hodnoty zaznamenávať. Následne po potvrdení údajov používateľom kontroluje dané vstupy – teda či nie je číslo záporné a zadané korektne, či je zadaný názov súboru. V prípade zlých vstupov zobrazí okno so správou o chybe.

A to pomocou :

- *__init__(self, parent) – vytvorí panel, ktorý bude mať ako rodiča druhý
argument „parent“ (multisplitter okna) a vytvorí svoje premenné*
- *place_elements(self) – umiestňuje elementy (labely, textfieldy) na
požadované miesto na ploche*
- *correct_values(self) – kontroluje správnosť vložených údajov podľa
dohodnutých podmienok, obe polia sú povinné, takže v prípade, že
niečo nebude zadané alebo nebude zadané správne (napr. interval
merania ≤ 0) vyskočí o tom hlásenie*
- *clear(self) – vymaže všetky elementy z okna*
- *focus(self) – nastaví focus na prvý textfield (pre pohyb tabulátorom)*

- **measurement_data.py**

Obsahuje triedu *MeasurementData*, ktorá spravuje namerané údaje, stará sa o exportovanie nameraných údajov vo forme tabuľky a grafu do Excelu a ukladá

meranie do formátu .pickle, ktorý je používaný pri načítavaní už vykonaných meraní v aplikácii.

Preto viaceré metódy :

- `__init__(self)` – inicializuje svoje premenné, medzi nimi aj `self.values`, kde si bude pamätať všetky dvojice čas-hodnota
- `insert_value(self, value)` – v prípade, že hodnoty prejdú logickou kontrolou správnosti, budú zapamätané do `self.values`
- `get_last_value(self)` – vráti poslednú nameranú dvojicu čas-hodnota
- `export_to_excel(self)` – uloží hodnoty zo `self.values` do excelu pomocou knižnice `xlsxwriter`; v prípade, že sa to podarí vráti `True`, inak `False`
- `pickle(self)` – uloží hodnoty zo `self.values` do súboru pomocou modulu `pickle`; v prípade, že sa to podarí vráti `True`, inak `False`
- `clear(self)` – premaže všetky uložené hodnoty a iné premenné
- `unpickle(path)` – cieľový súbor (z `path`) pomocou modulu `pickle` prečíta a uloží do `self.values`

- ***messagebox.py***

Obsahuje triedu `AlertBox`, ktorá má za úlohu vytvoriť vyskakovanie okno s upozornením. Toto dialógové okno je využívané pri chybových hláškach alebo pri úspešne vykonaných udalostiach ako napr. exportovanie meranie.

Obsahuje len jednu metódu pre vyvolanie hlásenia : `show(text)` – kde *argument text* bude výsledným textom hlásenia

- ***panel_handler.py***

Obsahuje triedu `PanelHandler`, ktorá obsluhuje výmenu panelov, keď je potrebné zmeniť rozloženie okna napríklad pri kliknutí na niektoré tlačidlo alebo pri použití niektorej klávesovej skratky.

Celá aplikácia beží v jednom frame, na základe stavu, v ktorom sa aplikácia nachádza (napr. počas merania, po meraní, s vykreslením grafom, ...) sú panely pridávané alebo odoberané z hlavného framu.

Jeho metódy :

- `__init__(self, window, splitter)` – vytvorí triedne premenné, medzi nimi si zapamätá okno z argumentu `window` (okno z `__init__.py`), `splitter` (teda `multisplitter` okna) a `buttony` daného okna

- `info_panels(self)` – zavolá svoju metódu `self.clear(True)`, vytvorí `InputPanel` (**`input_panel.py`**) a volá vlastnú metódu `self.add()`, ktorou zobrazí a pridá `InputPanel` a buttony do `multisplittera`
- `start_panel(self)` – volá vlastnú metódu `self.add()`, ktorou zobrazí a pridá buttony do `multisplittera`
- `during_measurement_panels(self)` – volá vlastnú metódu `self.add()`, ktorou zobrazí a pridá buttony a tabuľku do `multisplittera`
- `after_panels(self)` – zavolá svoju metódu `self.clear()`, volá vlastnú metódu `self.add()`, ktorou zobrazí a pridá buttony a tabuľku do `multisplittera`, povolí scrollovanie tabuľky pomocou `table.show_scrollbar()` a zavolá `table.set_after_measurement()`
- `graph_panels(self)` – zavolá svoju metódu `self.clear()`, vytvorí `GraphPanel` (**`graph_panel.py`**) a volá vlastnú metódu `self.add()`, ktorou zobrazí a pridá buttony, tabuľku a graf do `multisplittera`
- `clear(self, second_measurement=False)` – vymaže všetky panely z `multisplittera` a vyčistí okno
- `add(self, *args)` – pridá všetky panely z druhého argumentu do `multisplittera`
- `before_connection(self)` – skryje prvý panel (buttonov) a premaže plochu volaním vlastnej funkcie `self.clear()`

- **`pipigraph.py`**

Obsahuje triedu `PipiGraph`, ktorá pracuje s prístrojom „ π graf“. Trieda `PipiGraph` je inicializovaná s nameranými hodnotami z triedy `MeasurementData` a vytvára spojenie s prístrojom. Po úspešnom nadviazaní spojenia s prístrojom vydáva zvuku vo forme pípania.

Vydávanie zvuku funguje nasledovne: Posuvným potenciometrom sa používateľ virtuálne posúva po čiarovom grafe z nameraných hodnôt. Na základe pozície posuvného potenciometra sa určí aká x-ová súradnica grafu predstavuje túto hodnotu a následne trieda `PipiGraph` vydá zvuk zodpovedajúci y-ovej súradnici príslušnej pre danú x-ovú súradnicu.

Výška tónu sa mení na základe hodnoty aké je reprezentovaná v grafe. Čím vyššia hodnota na vstupe príde, tým vyšší bude tón.

Popis jednotlivých metód :

- `__init__(self, values)` – uloží výsledok o pokuse nadviazať spojenie s prístrojom volaním vlastnej metódy `self.create_connection()`, vytvorí triedne premenné a medzi nimi si uloží aj vstupné values
- `create_connection(self)` – nadväzuje resp. pokúša sa nadviazať spojenie s π graf-om ; ak sa to podarí vráti `True`, inak `False`

- `get_time(self, value)` – dá k príslušnej hodnote 0-1023 vhodný čas na základe hodnôt/časov z merania
- `get_value(self)` – využitím matematiky vypočíta k x-ovému bodu (z `get_time`) príslušnú y-ovú súradnicu
- `read_values(self)` – komunikuje s π graf-om a vydáva zvuk pomocou volania vlastnej metódy `self.play()`
- `scale(self, value)` – prispôsobuje namerané hodnoty ľudskému sluchu
- `play(self)` – pomocou modulu `winsound` vydáva zvuk

- ***splitter.py***

Tento súbor zobrazuje zmodifikovanú originálnu verziu `wx.MultiSplitterWindow`. V súbore sa nachádza trieda `MultiSplitterWindow`. Táto trieda zaistuje rozmiestňovanie panelov v okne. Taktiež sa stará a obsluhuje vymieňanie panelov, ich zobrazovanie a skrývanie podľa toho, v akom stave merania sa používateľ nachádza a podľa toho, aké akcie používateľ robí.

- ***table_panel.py***

Obsahuje triedu `TablePanel`, ktorá vytvára dvojriadkovú tabuľku a umožňuje ovládanie tabuľky. Zaobstaráva pridávanie hodnôt do tabuľky, posúvanie (scrollovanie) tabuľky, zmenu veľkosti tabuľky. Taktiež, má metódu, ktorá umožňuje čítanie hodnôt v tabuľke, táto metóda potrebuje k fungovaniu internet. Nakoľko NVDA screen reader nevie čítať tabuľku, tak čítanie je zabezpečené pomocou tejto metódy.

Podrobnejší popis jednotlivých metód :

- `__init__(self, parent)` – vytvorí panel, ktorý bude mať ako rodiča druhý argument „parent“ (`multisplitter` okna), vytvorí svoje premenné a nastaví tabuľke vhodné rozmery
- `to_menu(self, event)` – v prípade, že používateľ stlačí tlačidlo TAB, tak bude jeho focus prenasťavený na prvé viditeľné tlačidlo
- `resize(self, num)` – zväčší tabuľku o „num“ okien
- `add(self, time, value, load=False)` – pridá dvojicu hodnôt čas-dátum do tabuľky; v prípade, že nejde o načítanie celej tabuľky naraz, je potrebné tabuľku správne zväčšiť
- `scroll_table(self)` – zakáže scrollovanie v tabuľke
- `get_text(self, event)` – získa string hodnoty políčka, kde je focus a zavolá na ňu svoju metódu `self.speak()`
- `speak(self, value, repeat=False)` – prečíta danú hodnotu za pomoci modulu `pygame`

- `read_last(self, event)` – zavolá `self.speak()` na poslednú nameranú hodnotu
- `show_scrollbar(self)` – zobrazí horizontálny scrollbar
- `set_after_measurement(self)` – nastaví vlastnú premennú `self.after_measurement` na `True` (dôležité inde na kontrolu stavu)

- **window.py**

Trieda `MainWindow` v tomto súbore predstavuje základné hlavné okno, do ktoré sa zobrazujú panely a všetky ovládacie prvky. Toto okno existuje počas celého behu aplikácie, je iba jedno a toto okno sa ďalej predáva ako argument na zobrazovanie panelov pomocou `splitter-a`. Stará sa aj o vytvorenie klávesových skratiek a ich bindovanie na príslušné funkcie.

A jeho metódy :

- `__init__(self)` – vytvorí hlavné okno, vytvorí viacero triednych premenných a medzi nimi aj `multisplitter (self.splitter)`, `self.handler`, `ButtonPanel (self.buttons)`, zatiaľ skrytý `TablePanel(self.table_panel)`, zatiaľ skrytý `InputPanel(self.input_panel)` a zavolá svoju funkciu `self.bind_buttons()`
- `bind_buttons(self)` – všetkým buttonom priradí ich funkcie (ktoré sú definované v **button_panel.py**) a ich klávesové skratky podľa dohody
- `end(self)` – ukončuje thredy
- `read(self, event)` – z klávesovej skratky : volá prečítanie poslednej hodnoty, teda `table_panel.read_last(event)`
- `beep(self)` – (ak ešte neexistuje) vytvorí a uloží ako `self.ppg` novú inštanciu triedy `PipiGraph`, nastaví ho ako aktívny a pustí ho pomocou jeho metódy `self.ppg.read_values()`
- `pipi(self, event)` – spustí thread pre pipigraf v prípade, že je program v stave „Zobrazený graf“ (a už taký thread nebeží)
- `update(self)` – po krátkom čase zavolá svoju funkciu `self.check()`
- `check(self, event)` – kontroluje či beží meranie : ak áno, spustí timer a po dvoch 2 sekundách zavolá sama seba (čím to znova skontroluje); ak meranie skončilo, tak zavolá metódu `handler.cancel(True)`
- `create_table_panel(self)` – vytvorí `TablePanel`, ktorý skryje a priradí ho premennej `self.panel_handler.table`
- `to_grid(self, event)` – metóda sa volá v prípade, že sa používateľ snaží dostať TABOM alebo šípkami z posledného buttonu na tabuľku, ak práve beží meranie, tak to nie je povolené a focus sa mu znova nastaví na prvý (a jediný) button „Zataviť meranie“, v každom inom prípade, sa mu nastaví focus na prvú nameranú hodnotu (nie čas, ale hodnotu)