Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Konfigurace měřícího pracoviště

Plzeň, 2014 Pavel Pour

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 4. 5. 2014

Pavel Pour

Abstract

Configuration of measure workplace

The main purpose of this work is to automate the preparatory work that must be done before one can start measuring physical quantities. The work focuses on computer support for the collection of information that is needed for both the description of the metrological characteristics of measurement workplace and the measured object. This data interrelate in order to deliver a qualified decision about suitability of a particular workplace to measure the given object.

The work focuses on the development of a software application which will enable to facilitate insertion, utilisation and storage of measured data. The aim is to create a specialized and intuitive user interface that will accelerate and streamline the preparation of measurement. The interface should also allow for easy editing and re-use of data.

Abstrakt

Konfigurace měřícího pracoviště

Hlavním záměrem této práce je zautomatizovat přípravné práce, které je nutné provést před tím, než je možné začít s měřením fyzikální veličiny. Zaměřuje se na počítačovou podporu shromažďování informací, které jsou potřebné jak pro popis metrologických charakteristik měřícího pracoviště, tak měřeného objektu. Tato data se vzájemně propojují za účelem kvalifikovaného rozhodnutí o vhodnosti určitého pracoviště k měření daného objektu.

Práce se zaměřuje na vývoj aplikace, která obsluze měřícího pracoviště co nejvíce usnadní vkládání, využití a archivaci dat měřícího zařízení. Cílem je sestrojit specializované a intuitivní uživatelské prostředí, které urychlí a zefektivní přípravu měření a umožní i snadnou pozdější editaci a opětovné využití dat.

Obsah

1 Úvod

Schopnost měřit – kvantifikovat fyzikální vlastnosti jevů – umožnila prudký rozvoj různých věd a průmyslových oborů v posledních stoletích. Spolu s rozvojem průmyslu se zpětně zvyšovaly nároky na měření a vedly k vytvoření nových složitějších měřidel, měřících postupů a předpisů. Stav tohoto průmyslového odvětví je dobře zdokumentován v literatuře. Při tvorbě práce jsem čerpal např. z [JV03], [JV08], [TČG00] nebo [GRS02].

Složitější měřící postupy spolu s novými předpisy, které vyžadují zajistit opakovatelnost měření a archivaci použitých postupů pro jejich dohledatelnost v případě řešení budoucích rozporů, zvedly výrazně administrativní zatížení subjektů, které provádějí různá měření. Dnes je prakticky nemyslitelné provádět měření na profesionální úrovni bez podpory výpočetní techniky. Výpočetní technika může pozitivním způsobem vstupovat do všech fází měření, a to od jeho přípravy až do prezentace výsledků měření.

Za dobu od nástupu PC do podniků a laboratoří do současnosti bylo sice vyvinuto několik systémů pro podporu metrologie, ale oblast měření je natolik rozsáhlá, že každý z těchto systémů pokrývá pouze malou část problematiky. Tyto systémy jsou si většinou podobné v oblasti evidence měřidel v podniku, ale v oblasti vlastního měření je stále velký prostor pro řešení, která by uspokojila konkrétní požadavky různých subjektů.

Tato práce navazuje na moji předchozí bakalářskou práci, ve které jsem se zabýval právě procesem měření a jeho automatizací. Nyní jsem se zaměřil na softwarovou podporu přípravy měřícího pracoviště při měření v elektrotechnice. Při přípravě měřícího pracoviště v elektrotechnice se v měřícím pracovišti používá několik navzájem propojených měřidel, z nichž každé může obsahovat desítky měřících rozsahů s množstvím různých atributů. Proto je příprava pracoviště poměrně administrativně náročná činnost, kterou je výhodné automatizovat pomocí výpočetní techniky.

2 Základy měření

Měřením se zabývá vědní obor, který se nazývá metrologie. Cílem měření je kvantifikovat vybranou fyzikální veličinu za účelem stanovení její velikosti ve stanovených jednotkách. Jednotka měřené veličiny je při měření reprezentována měřícím zařízením, které by mělo být nepřerušeným řetězcem vzájemných porovnání navázáno na mezinárodní standard jednotky dané veličiny.

Zde je tedy možno vidět dvě hlavní oblasti, kterými se metrologie zabývá. První oblast se zabývá právě jednotností měření na mezinárodní a národní úrovni a to tak, že stanoví jakým způsobem je definována jednotka fyzikální veličiny a následně vytvoří přesnou fyzikální realizaci - normál této jednotky.

Pomocí tohoto mezinárodního normálu jsou zpravidla kontrolovány národní normály. S národními normály jsou pak porovnávány např. podnikové normály a s těmito dále pracovní normály, kterými se již přímo kontrolují měřidla používaná pro měření.

Druhou oblastí zájmu metrologie je pak vlastní měření např. při výrobě zboží, výzkumu či sledování klimatu.

Ať již se provádí měření kdekoli, pak základem měření je vždy měřící pracoviště. Měřící pracoviště může být tvořeno pouze jedním měřícím přístrojem a měřeným objektem bez toho, aniž by bylo vyčleněno nějaké zvláštní místo pro provádění měření. Můžeme si představit situaci, kdy zaměstnanec stavební firmy měří délku zdi. Na druhou stranu se měřící pracoviště může skládat z mnoha komplikovaných zařízení, řízených počítačem, umístěných v klimatizované, bezprašné místnosti.

Ať se jedná o první nebo druhý případ, vždy by se měly dodržovat určité stejné zásady, které vycházejí z mezinárodních norem, např. ISO 9001 a měly by být pravidelně kontrolovány managementy podniků či auditovány dozorovými orgány.

Jednou ze zkoumaných oblastí při různých auditech je právě oblast přípravy měření. Kontrolní orgán zjišťuje, jakým způsobem je zajištěna evidence metrologických dat a to jak na straně přípravy měření, tak na straně výstupu z měřících procesů. Z hlediska zajištění správnosti a opakovatelnosti měření je velký důraz kladen na podkladová data používaná jednak pro sestrojení měřícího pracoviště, ale i po provedení měření pro následné vyhodnocení nejistoty měření.

Subjekt, u kterého tvoří měření podstatnou část jeho činnosti se již dnes neobejde bez počítačové podpory měření. Tato podpora může mít například podobu tabulek v sešitu tabulkového procesoru, pokud je druhů měřených objektů poměrně málo. Zde můžeme uvést například laboratoř, která provádí rozbory vody.

Na druhou stranu jsou subjekty, které musí být schopny měřit tisíce různých typů zařízení. Takovým subjektem je i pracoviště, se kterým jsem spolupracoval při vzniku této práce. Toto pracoviště má ve svém výpočetním systému za dobu své existence zavedeno přes tisíc různých typů měřidel elektrických a tisíc mechanických. Toto obrovské množství není možné zvládnout tabulkovým procesorem, a tak na pracovišti postupně začínali hledat a využívat sofistikovanější systémy.

3 Software pro metrologii

Tato práce navazuje na moji předchozí bakalářskou práci, ve které jsem se též zabýval měřícím pracovištěm. Protože jsem původním povoláním elektrotechnik, mám k tomuto oboru blízko. Výběrem tohoto tématu jsem mohl zúročit moje dosavadní zkušenosti v oblasti měření v elektrotechnice, které se počaly ještě v době mé práce v dnes již neexistující laboratoři v ETD Doudlevce v Plzni a spojit je s nabytými poznatky z oblasti informatiky. Již před příchodem PC, jsme se snažili ulehčit si některé činnosti výpočetní technikou, a tak mohu nyní navázat na tehdejší záměr.

V bakalářské práci jsem se zabýval automatizací měřícího procesu. Výsledkem práce je software, který umožňuje dálkově ovládat různá měřící zařízení a provádět měření podle předem specifikovaného scénáře. Jednotlivé scénáře - postupy měření jsou uloženy v databázi, odkud se načítají v případě potřeby. Při této činnosti jsem si uvědomil velikou náročnost přípravy měřících postupů a nutnost kvalitní softwarové podpory pro jejich vytváření. V této práci se tedy zabývám vším, co je třeba zaznamenat, nakonfigurovat a nastavit před vlastním měřením a výsledkem je program, který toto vše v maximální míře ulehčuje a automatizuje.

Před začátkem vývoje software, by se měl provést průzkum trhu a zjistit, jestli již neexistuje podobný software, který by šel použít. Já jsem tento bod vyřešil spoluprací s laboratoří provádějící měření, a to dvěma způsoby.

Za prvé jsem nezačínal zcela od nuly. Laboratoř již používá výpočetní systém, se kterým je spokojená a není záměrem jej nahradit, ale rozšířit. Tím jsem zároveň vyřešil i fázi návrhu vývojové platformy, protože cílem byla co největší kompatibilita se stávajícím systémem.

Za druhé se dá říci, že průzkum trhu za mne provedli pracovníci této laboratoře již v dobách, kdy se rozhodovali jestli nakoupí nějaký specializovaný systém jako hotový produkt, nebo půjdou svojí vlastní cestou.

Tato laboratoř zkoušela dva programy od českého výrobce zařízení pro metrologii, které prodával či prodává jako doplněk ke svým výrobkům.

První verzi programu do laboratoře nakoupili v době, kdy se do podniku zavedly PC. Měl jsem možnost si prohlédnou zažloutlý manuál k tomuto programu P-1106. Byl určen ještě pro operační systém DOS, uživatelské rozhraní používalo znakovou grafiku. Pracovníci laboratoře jej používali hlavně pro kalibrace etalonů, kde se vyplatila poměrně zdlouhavá práce při přípravě jakýchsi řídících skriptů, které byly složeny z řídících příkazů. Většího využití se však tento program v laboratoři nedočkal a údajně i sami dodavatelé programu se zmínili, že nejsou s programem od svého externího dodavatele zcela spokojeni

To byl zřejmě důvod k vývoji zcela nového programu Kalibr, na který již výrobce nasadil vlastní programátory. Program se prodává s určitými změnami dodnes. Pracovníci laboratoře zkoušeli i tento program, který již byl určen pro Windows 95. I když byl daleko lepší než předchozí verze, přesto se neujal, protože program neřešil a ani nemohl řešit různá specifika organizace, jíž je laboratoř součástí. Jedná se především o evidenci a účtování zakázek v silně decentralizovaném prostředí.

Nakonec se v laboratoři rozhodli nasadit jiný systém, jehož vývoj začal pokrytím právě těchto administrativních potřeb a postupně byl slučován s dalšími prvky a činnostmi. Vzhledem k tomu, že laboratoř má postavené své výpočetní prostředí na OS Linux, neuvažovalo se ani o pokusu začlenit program Kalibr do používaného systému.

S programem Kalibr jsem se blíže neseznamoval proto, že rozhodnutí nekoupit tento systém je definitivně rozhodnuto. Také jsem se nechtěl nechat ovlivňovat způsobem, jakým řešili problematiku jeho výrobci a zkusit přijít s jiným pohledem na věc na základě mých zkušeností z elektrotechniky a z vývoje předchozího programu pro kalibrace. Principiální základy obou programů i názvosloví jsou asi do určité míry podobné, protože vychází ze zvyklostí tohoto oboru vyplývajících z norem a různých doporučení České metrologické společnosti (ČMS).

4 Popis měřícího pracoviště

Měřící pracoviště v elektrotechnice slouží k měření jedné nebo několika elektrických veličin. V každém pracovišti se nachází alespoň jeden zdroj veličiny. Zdroj veličiny může být zároveň i měřeným zařízením. Dále by mělo být v pracovišti alespoň jedno měřící zařízení. To je nejjednodušší případ. V praxi se však stává, že výstup zdroje veličiny a vstup zařízení spolu zcela nekorespondují. Může nastat několik případů:

- Veličina výstupu zdroje a vstupu zařízení je stejná, ale rozsah vstupu zařízení nepokrývá rozsah výstupu zdroje.
- Veličina výstupu zdroje je jiná, než veličina vstupu zařízení.
- Kombinace předchozích dvou variant.

V těchto případech se do měřícího pracoviště zařazují další komponenty, ať již ve fyzické formě (např. bočník nebo převodní transformátor), nebo jako softwarový převodník, který je reprezentován pouze matematickým výpočtem (např. převod napětí termoelektrického článku na teplotu ve °C).

4.1 Měřící zařízení

Jak bylo řečeno výše, měřící zařízení jsou základní komponentou měřícího pracoviště. Proto je nutné se s nimi seznámit trochu blíže

Pro účely metrologického popisu je třeba zohlednit údaj o tom, o jaký druh zařízení se jedná a jakým způsobem komunikuje s okolím. To určuje jeho měřící možnosti, přesnost a způsob vyjádření přesnosti. Většinu zařízení a jejich ukazatelů můžeme rozdělit do několika kategorií viz tab. 4.1:

Název	Poznámka
Digitální měřidlo	Zařízení s výstupem na display nebo přes digitální rozhraní
Analogové měřidlo lineární	Ručkové měřidlo s lineární stupnicí
Analogové měřidlo s potlačenou nulou	Lineární stupnice nezačíná od nuly
Analogové měřidlo nelineární	Ručkové měřidlo s nelineární stupnicí
Zdroj veličiny proměnný	Zdroj s regulovatelným výstupem
Zdroj veličiny pevný	Zdroj s jedinou hodnotou výstupu
Převodník analogový	Spojitě pracující převodník
Převodník digitální	Nespojitě pracující převodník
Softwarový převodník	Reprezentován výpočtem

Tabulka 4.1: Kategorie měřících zařízení

Většina dnešních elektrických měřidel je schopna měřit několik různých elektrických veličin. Například běžný univerzální multimetr typu VDM-1 měří veličiny uvedené v tab. 4.2:

Název	Kód
el. napětí stejnosměrné	UDC
el. napětí střídavé	UAC
el. proud stejnosměrný	IDC
el. proud střídavý	IAC
el. odpor	R

Tabulka 4.2: Měřené veličiny VDM-1

Dnešní multimetry jsou však často schopné měřit i el. kapacitu, indukčnost, frekvenci, zesílení tranzistorů, napětí polovodičových přechodů a teplotu jako funkci napětí termoelektrických článků.

4.1.1 Měřící rozsahy

Pro každou veličinu má přístroj několik měřících rozsahů, které se přepínají buď ručně nebo automaticky (u některých multimetrů vyšší třídy) dle velikosti měřené veličiny. Rozsahy měřidla VDM-1 jsou uvedeny v tabulkách 4.3 – 4.7.

Rozsah	Přesnost
200 mV=	± (0,5 % MH + 0,5 % MR)
2 V=	± (0,5 % MH + 0,5 % MR)
20 V=	± (0,5 % MH + 0,5 % MR)
200 V=	± (0,5 % MH + 0,5 % MR)
1000 V=	± (0,5 % MH + 1 % MR)

Tabulka 4.3: Rozsahy stejnosměrného napětí

Rozsah	Přesnost
200 mV~	± (0,5 % MH + 0,5 % MR)
2 V~	± (1 % MH + 0,5 % MR)
20 V~	± (1 % MH + 0,5 % MR)
200 V~	± (1 % MH + 0,5 % MR)
750 V~	± (1 % MH + 1,5 % MR)

Tabulka 4.4: Rozsahy střídavého napětí

Rozsah	Přesnost
200 µA=	± (0,5 % MH + 0,5 % MR)
2 mA=	± (0,5 % MH + 0,5 % MR)
20 mA=	± (0,5 % MH + 0,5 % MR)
200 mA=	± (0,5 % MH + 0,5 % MR)
2 A=	± (1 % MH + 1 % MR)
10 A=	± (1 % MH + 1 % MR)

Tabulka 4.5: Rozsahy stejnosměrného proudu

Rozsah	Přesnost
2mA~	± (1 % MH + 0,5 % MR)
20 mA~	± (1 % MH + 0,5 % MR)
200 mA~	± (1 % MH + 0,5 % MR)
2 A~	± (1 % MH + 1,5 % MR)
10 A~	± (1 % MH + 1,5 % MR)

Tabulka 4.6: Rozsahy střídavého proudu

Rozsah	Přesnost
200 Ω	± (0,5 % MH + 0,5 % MR)
2 kΩ	± (0,5 % MH + 0,5 % MR)
20 kΩ	± (0,5 % MH + 0,5 % MR)
200 kΩ	± (0,5 % MH + 0,5 % MR)
2 MΩ	± (0,5 % MH + 0,5 % MR)
20 MΩ	± (1 % MH + 0,5 % MR)

Tabulka 4.7: Rozsahy elektrického odporu

Pro ukázku byl vybrán jeden z nejjednodušších multimetrů. Solidnější multimetry mohou mít desítky měřících rozsahů a u zkušebních kalibrátorů může počet oscilovat mezi jedním až dvěma sty rozsahů. Pro každý rozsah je třeba zaznamenat větší množství údajů, které popisují možnosti daného rozsahu.

Každý rozsah měřidla je specifikován hlavně druhem měřící veličiny a její velikostí. Velikost veličiny má určité rozpětí, které určuje nejmenší a největší možnou velikost veličiny, při níž měřidlo měří tak, jak výrobce garantuje, a nedojde k jeho poškození.

Nezřídka však rozsah závisí na více veličinách. Jako příklad může posloužit střídavé napětí, jehož měření ovlivňuje i jeho frekvence. Je proto nutné k rozsahu měřidla připojit ještě další veličinu, jako ovlivňující parametr, který musí být při měření též v určitých stanovených mezích.

4.1.2 Přesnost měřidel

To, co odlišuje běžná a kvalitní měřidla, je kromě mechanické konstrukce hlavně jejich přesnost, která má zásadní vliv na přesnost samotného měření. Přesnost měřidla stanovuje většinou jeho výrobce, v opačném případě se musí stanovit kontrolním měřením.

Přesnost měřidla je možné udávat několika různými způsoby viz tab. 4.8:

Název	Označení
% z maximální hodnoty rozsahu měřidla	% MH
počet digitů (digitální měřidla)	DIG
počet dílků stupnice (analogová měřidla s lineární stupnicí)	d
% z délky stupnice (analogová měřidla s nelineární stupnicí)	V
% z měřené hodnoty	% MH
v jednotkách měřené veličiny	jednotky dle SI

Tabulka 4.8: Způsoby uvádění přesnosti měřidel

Ne všechny tyto způsoby se mohou použít u všech druhů měřidel. Většinou se však používají alespoň dva druhy, které se navzájem vhodně doplňují.

Pokud má určitý přístroj místo měření veličiny tuto veličinu převádět na jinou, respektive měnit její velikost, je zapotřebí znát i převodní vztah tohoto převodníku.

4.2 Měřící pracoviště

V předchozím bodě byly podrobně popsány základní stavební prvky pracoviště. Samotné pracoviště není ničím jiným, než vzájemným spojením těchto prvků. Kromě měřících zařízení se tedy v pracovišti nacházejí pouze propojovací prvky. Propojovací prvky by měly vytvářet logickou návaznost toku měřené veličiny v pracovišti, ale neměly by nijak zasahovat do kvality měření.

Pokud by tomu tak bylo, je třeba zohlednit negativní dopad propojovacích prvků například při popisu měřícího zařízení, které bude těmito prvky připojeno. Uvažujme např. měřič odporu, který bude měřit nízké odpory v řádu jednotek Ω. Pokud takový přístroj připojíme přes měřící vodiče, jejichž odpor je 0,01Ω , může tento dodatečný sériový odpor zatížit měření přídavnou chybou až 1%.

Existuje několik způsobů, jak tento problém eliminovat:

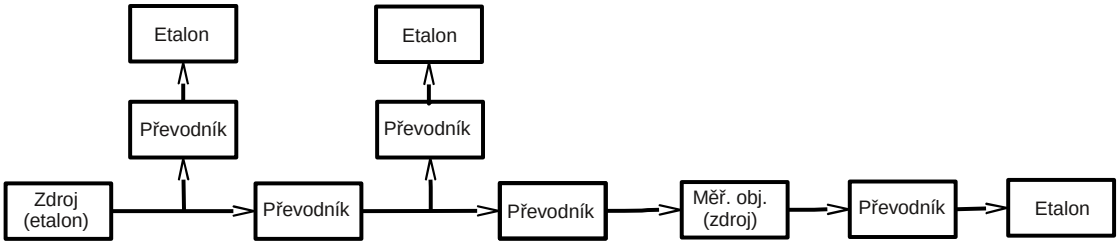
a) Měřidlo měří odpor čtyřmi vodiči.

b) Měřidlo je vybaveno nulováním zbytkového odporu.

c) Měřící vodiče se stanoví jako příslušenství k měřidlu a jejich známý odpor se odečte od měřené hodnoty.

d) Měřící přívody se zakomponují do pracoviště jako samostatné zařízení ve funkci softwarového převodníku, který měřený odpor sníží o odpor přívodních vodičů.

Protože zařízení v pracovišti může být více a jejich vzájemným propojením může vzniknout mnoho kombinací, vznikl již při mé předchozí práci požadavek na usměrnění počtu kombinací a dosažení formy, která by umožnila navrhnout algoritmus pro výpočet podmínek v pracovišti. Vytvořil jsem jakýsi maximální plán pracoviště, se kterým je program schopen pracovat. Nyní jej proto znovu připomenu, protože má přímý vliv na vývoj i v rámci této práce viz obr. 4.1.



Obrázek 4.1: Schéma maximálního pracoviště

Označení „Etalon“ v tomto obrázku může být trochu matoucí, protože pro měření není nutné používat přímo etalony. Myslí se tím libovolné měřící zařízení, které je navázané na etalony a má vliv na přesnost měření. Takovým zařízením totiž může být i přesný zdroj veličiny, který současně nahrazuje samostatný zdroj a např. multimetr.

Tento plán označuje všechny pozice, kde může být umístěno nějaké zařízení. Které pozice budou v praxi skutečně obsazené, záleží na druhu měření. Vlastní popis měřícího pracoviště tedy představuje seznam pozic a identifikaci zařízení, které se na dané pozici nachází.

5 Popis měření

5.1 Postup měření

V předchozích oddílech jsou popsány komponenty použité při měření. Další kategorií informací, které jsou nutné pro provedení měření, je předpis, co se bude s těmito komponentami měřit. Je-li třeba z tohoto hlediska popsat měřený objekt, pak není nutné věnovat se konstrukci tohoto objektu a jeho funkci. Na objekt můžeme nahlížet jako na černou skříňku, kterou popíšeme jejími vstupními a výstupními rozhraními. Pokud je měřené zařízení zdrojem, zajímá nás výstup, v opačném případě vstup.

Je samozřejmé, že pro vlastní měření musí znát obsluha i něco z funkce zařízení, alespoň proto, aby bylo možné zařízení zprovoznit a přepínat do různých módů dle požadavků na měření. Z hlediska vlastního zpracování měření však tyto informace nejsou zásadní a řeší se přečtením manuálu. Přece jen i zde však může software pomoci v tom smyslu, že může dávat při měření obsluze určité pokyny, podle kterých pak obsluha zařízení ovládá.

Na základě teorie černé skříňky je nutné znát následující údaje pro každý měřící bod:

- druh měřené veličiny
- velikost měřené veličiny
- druh veličiny parametru měření
- velikost parametru
- požadovaná přesnost měření

Z hlediska těchto parametrů můžeme měření rozdělit na dvě kategorie:

a) Experimentální. U experimentálních měření bude asi dopředná znalost některých parametrů měření poněkud více zatížena chybami. Experimentální měření jsou poměrně kreativní záležitostí a pro tento druh měření by mohl být určitý software poněkud svazující, i když ani zde není vyloučeno jeho použití.

b) Provozní. Jedná se o běžná měření na předem dostatečně známých objektech, např. při výrobních nebo servisních kontrolách. Při těchto měřeních nejenže je dopředu známo, jakou hodnotu je možné naměřit, je dokonce požadováno, aby naměřená hodnota měla předem danou velikost. Výstupem měření pak není hlavně naměřená hodnota ale spíše informace o tom nakolik se naměřená hodnota odchyluje od původního předpokladu. Na tento druh měření především cílí software vyvíjený v této práci.

Postup měření můžeme chápat jako seznam měřících bodů s popisem hodnot těchto bodů dle výše uvedeného seznamu.

5.2 Popis měření

V předchozích kapitolách byly detailně popsány všechny informace, které je nutné shromáždit před tím, než je možné přistoupit k měření. V tomto posledním kroku se již nevkládají žádná nová data z externích zdrojů (např. manuály, výrobní dokumentace). Poslední část dat pouze spojuje vše v jeden celek.

Z měřícího postupu je známo, na jakých měřících bodech se má měření provést. Z popisu pracoviště a měřících zařízení jsou známy měřící možnosti zařízení. Nyní nezbývá, než-li propojit konkrétní měřící bod s konkrétním pracovištěm, konkrétním měřícím zařízením a jeho konkrétním nastavením – rozsahem. Popis měření je tedy hlavně souhrn identifikátorů těchto prvků, který je jednoznačně identifikuje v místě jejich uložení.

K těmto identifikátorům stačí dodat ještě další dodatečné informace. Používá-li se při měření možnost automatického řízení měření, pak je možné provádět místo jednoho odměru v jednom bodě odměrů několik. Naměřené hodnoty se pak mohou kontrolovat na vznik chyb a zprůměrovat, čímž se zvýší přesnost měření. Dodatečnou informací by tedy mohl být počet odměrů, který se má při měření provést.

Při měření nastane situace, kdy je měřené i měřící zařízení vybaveno displejem, či nějakým číselníkem. Pak je možné se rozhodnout, na kterém z těchto zařízení bude obsluha měření odečítat hodnoty. I tato informace by měla být součástí konfigurace pracoviště.

Jako poslední dodatek se jeví rozumný požadavek na to, aby ke každému řádku měření mohl být dodán krátký popis, který by instruoval obsluhu měření.

6 Vize projektu

6.1 Současný stav

V současné době je pro zadávání dat používáno grafické rozhraní ve webovém prohlížeči. V prohlížeči se zobrazí html formulář, který obsahuje informace, které jsou uloženy na serveru. Nejdůležitějším prvkem formuláře jsou tabulky, které ve svých řádcích zobrazují informace měření. Na určitých místech tabulky jsou pak vstupní políčka pro vložení naměřených hodnot. V záhlaví formuláře jsou informace o zkoušeném měřidle (typ, výr. číslo atd.), v zápatí jsou vstupní políčka pro poznámky a doplňující informace. Tento formulář je vyhovující a prověřen praxí, proto jsem se snažil jej vzhledově zachovat i v nově vyvíjené aplikaci.

Na serveru je nainstalován tzv. LAMP - operační systém Linux, internetový server Apache, databáze MySQL a engine programovacího jazyka PHP.

Výhodou webového klienta je to, že je multiplatformní - může běžet na různých operačních systémech. V laboratořích, kde se kalibruje, je však všude nainstalován operační systém Linux. Pouze na kancelářském PC, z kterého se provádí fakturace, je OS Windows.

Dále je připravena nativní aplikace pro pracovní stanice, pomocí které lze dálkově ovládat použitá zařízení a provádět automatické měření. Tato aplikace je připravená k nasazení ovšem nepoužívá se z toho důvodu, že neumožňuje za běhu měření upravovat jeho parametry, což je v některých případech podstatná překážka pro její nasazení. Tuto schopnost je třeba zajistit.

6.2 Nevýhody současného řešení

Takto pojatý systém je velmi vhodný pro vývoj a snadnou údržbu aplikace, má však horší výkon. Spojení klienta a serveru je tzv. bezstavové, to znamená, že při každém dotazu klienta musí klient navázat nové spojení se serverem, server musí načíst z databáze všechna potřebná data, která potřebuje ke splnění úkolu a po ukončení dotazu je opět uložit (došlo-li k jejich změně) nebo je zapomenout. Toto se děje po každé akci uživatele. Problematický je rovněž požadavek na ovládání měřících zařízení, která jsou připojena přes sběrnici, z okna webového prohlížeče.

Příprava měření je pomocí HTML formuláře velmi zdlouhavá. Formuláře neumožňují svoji interaktivní konfiguraci. Pokud se uživatel splete při prvotním návrhu tabulky s daty, tabulku již nelze opravit a je nutné vytvořit novou, data ručně přepsat, a původní tabulku smazat. Všechny úpravy formuláře se provádí jeho odesláním na server.

Ruční způsob měření s sebou přináší prodloužení doby měření. U některých složitějších měření se počet jednotlivých odměrů pohybuje kolem stovky. Během této mnohdy jednotvárné práce se postupně může snížit bdělost obsluhy a dojít k chybám.

6.3 Požadovaný stav

Používat nativní program pro pracovní stanice, který by umožňoval interaktivní návrh konfigurace měření v plném rozsahu. Tím zajistit:

- Snadnou přípravu měření v jednotném rozhraní, které se používá i pro provedení měření
- Budoucí propojení konfiguračního rozhraní měření s měřící částí tak, aby mohlo být používáno při provádění měření pro dodatečnou konfiguraci jeho parametrů.

6.4 Požadavky

6.4.1 Karty zařízení

Rozhraní pro vkládání informací o měřících schopnostech měřícího zařízení. Umožnit interaktivním způsobem vložit do systému informace získané z technické dokumentace k zařízení tak, aby byly popsány jeho parametry z hlediska metrologie.

6.4.2 Pracoviště

Rozhraní pro konfiguraci pracoviště na základě plánu pracoviště a seznamu dostupných karet měřících zařízení.

6.4.3 Postupy měření

Interaktivní rozhraní pro vytváření měření jako posloupnosti požadovaných odměrů na měřeném zařízení. Možnost zohlednit parametry měřeného zařízení a požadované parametry měření.

6.4.4 Konfigurace měření

Interaktivní rozhraní pro vytváření vzájemných vazeb mezi postupy měření a předem nakonfigurovanými pracovišti, za účelem sjednocení všech dat potřebných pro provedení měření do jednoho datového souboru, který bude popisovat konkrétní variantu měření.

7 Případy užití

7.1 Role

7.1.1 Metodik

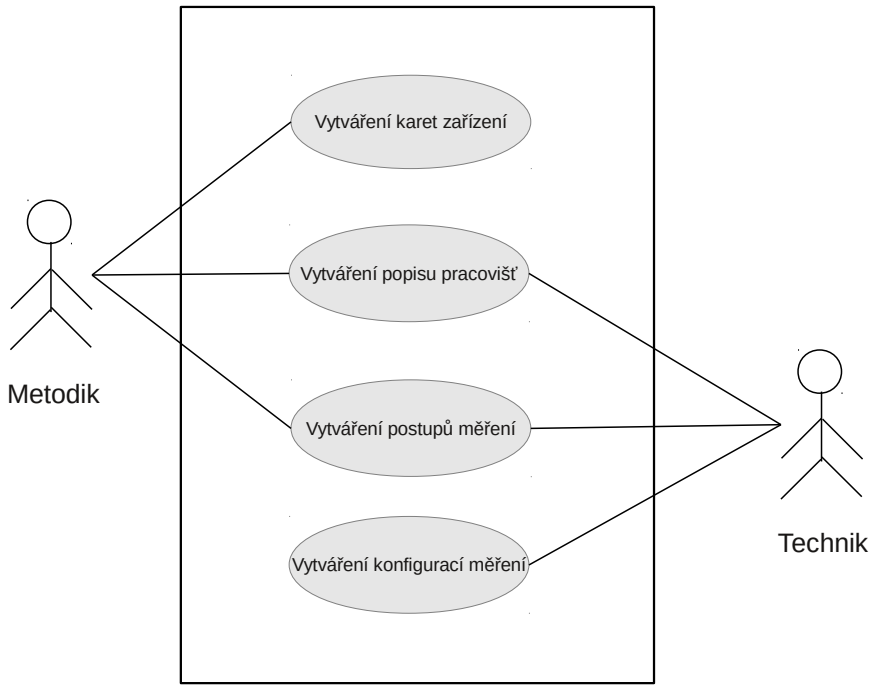
Pracovník zodpovědný za metodické vedení laboratoře. Hlavní těžiště jeho práce spočívá ve znalosti měřícího zařízení podniku a jeho údržby. S každým zakoupeným zařízením se seznámí (manuál) prověří jeho možnosti a parametry, a pak zařízení vloží do systému. Z těchto údajů připravuje pracoviště pro měření, která se v podniku provádějí.

Připravuje postupy pro běžná a standardizovaná měření, kde je dobrá dopředná znalost požadavků na měření

7.1.2 Technik

Provádí měření podle připravených postupů. K měření používá pracoviště vytvořená metodikem, která si samostatně přiřazuje k jednotlivým měřícím bodům na základě umístění měřeného zařízení a pracoviště v rámci jejich prostorového rozmístění.

Pokud provádí nestandardní měření, vytváří samostatně i měřící postupy.



Obrázek 7.1: Případy užití

7.2 Případy užití

Případy užití jsou zobrazeny na obr. 7.1.

7.2.1 Vytváření karet zařízení

Standardní průběh

1) Uživatel si otevře rozhraní pro přípravu karet. Vybere typ zařízení a tlačítkem otevře formulář nové karty. Nebo vybere hotovou kartu a tlačítkem spustí editaci formuláře karty. Nebo vybere rozpracovanou kartu a tlačítkem spustí editaci formuláře.
2) Uživatel pracuje ve formuláři. Formulář je složen z jednotlivých řádků, které popisují vlastnosti jednoho rozsahu zařízení. Řádky jsou podle společných vlastností seskupeny do tabulek. Tabulka je se svými ovládacími prvky sloučena do tzv. bloku. Jednotlivé bloky jsou viditelně oddělené. Uživatel si formulář pomocí tlačítek, která jsou umístěná v záhlavích bloků nakonfiguruje tak, aby mohl vložit potřebné hodnoty.
3) Uživatel stiskne tlačítko, zobrazí se nabídka pro uložení karty. Nabídka umožňuje v textovém poli změnit název formuláře. V nabídce musí vybrat přístup k zařízení a vybrat kartu, která bude současnou kartou přepsána.
4) Po navolení požadovaných hodnot kartu uloží stiskem tlačítka.

Nestandardní průběh

1) Pokud uživatel v bodě 1) nevybere v nabídce položku a stiskne tlačítko pro otevření karty, aplikace nereaguje.
2) Pokud uživatel v bodě 2) zapíše do políčka pro zadání čísla text, program políčko výstražně označí. Pokud uživatel zadá chybný vzorec převodu do příslušného políčka, políčko se výstražně označí. Formulář s výstražně zbarvenými políčky s chybným obsahem nelze uložit jako hotový. Pokud se uživatel pokusí uložit formulář jako hotový a nejsou vyplněna všechna políčka, objeví se upozornění a nevyplněná políčka se výstražně zvýrazní.
3) Pokud uživatel v bodě 3) nevybere v některé z nabídek žádnou položku, zobrazí se místo bodu 4) upozornění, jinak program nereaguje.

7.2.2 Vytváření popisu pracovišť

Standardní průběh

1) Uživatel si otevře rozhraní pro vytváření pracovišť. Vybere buď existující pracoviště, nebo volbu nové pracoviště. Po stisknutí tlačítka se otevře plán pracoviště.
2) Uživatel pracuje na konfiguraci pracoviště. Plán pracoviště je aktivní. Jednotlivé pozice v pracovišti jsou tvořeny tlačítky. Po stisku tlačítka na pozici, je možné na tuto pozici vložit nové zařízení nebo zařízení odstranit a nebo změnit.

3) Uživatel stiskne tlačítko uložit. Zobrazí se nabídka pro uložení karty. Vybere v nabídkách možnost přístupu k pracovišti a pracoviště, které se má přepsat současným pracovištěm. V textovém poli může zadat nový název pracoviště.
4) Po stisku tlačítka se pracoviště uloží.

Nestandardní průběh

1) Pokud v bodě 1) uživatel nevybere položku v nabídce a stiskne políčko pro otevření pracoviště, aplikace nereaguje.
2) Pokud by uživatel chtěl v bodě 2) vložit do pracoviště podruhé tu samou kartu zařízení, zobrazí se varování a vložení neproběhne.
3) Pokud uživatel v bodě 3) nevybere v některé z nabídek žádnou položku, zobrazí se místo bodu 4) upozornění, jinak program nereaguje.

7.2.3 Vytváření postupů měření

Standardní průběh

1) Uživatel otevře rozhraní pro vytváření postupů měření. Vybere buď existující hotový postup nebo rozpracovaný postup a stiskem tlačítka otevře tento postup pro editaci. Nebo vybere v nabídce typ zařízení a stiskem tlačítka iniciuje vytvoření nového postupu.
2) Uživatel pracuje ve formuláři. Formulář je složen z jednotlivých řádků, které popisují vlastnosti jednoho odměru. Řádky jsou podle společných vlastností seskupeny do tabulek. Tabulka je se svými ovládacími prvky sloučena do tzv. bloku. Jednotlivé bloky jsou viditelně oddělené. Uživatel si formulář pomocí tlačítek, která jsou umístěná v záhlavích bloků nakonfiguruje tak, aby mohl vložit potřebné hodnoty.
3) Uživatel stiskne tlačítko, zobrazí se nabídka pro uložení postupu. Nabídka umožňuje v textovém poli změnit název formuláře. V nabídce musí vybrat přístup k zařízení a vybrat postup, který bude současným postupem přepsán.
4) Po stisku tlačítka se postup uloží.

Nestandardní průběh

1) Pokud uživatel v bodě 1) nevybere v nabídce položku a stiskne tlačítko pro otevření postupu, aplikace nereaguje.
2) Pokud uživatel v bodě 2) zapíše do políčka pro zadání čísla text, program políčko výstražně označí. Pokud uživatel zadá chybný vzorec převodu do příslušného políčka, políčko se výstražně označí. Formulář s výstražně zbarvenými políčky s chybným obsahem nelze uložit jako hotový. Pokud se uživatel pokusí uložit formulář jako hotový a nejsou vyplněna všechna políčka, zobrazí se upozornění a nevyplněná políčka se výstražně zvýrazní.
3) Pokud uživatel v bodě 3) nevybere v některé z nabídek žádnou položku, zobrazí se místo bodu 4) upozornění, jinak program nereaguje.

7.2.4 Vytváření konfigurací měření

Standardní průběh

1) Uživatel otevře rozhraní pro vytváření měření. Může pokračovat výběrem rozpracovaného měření. Nebo vybere v jedné nabídce typ zařízení a okamžitě se v jiné nabídce se stromovou organizací zobrazí všechny postupy měření a pod nimi zavěšená všechna měření příslušná k danému měření. V této nabídce může buď vybrat měření a tlačítkem zobrazí formulář tohoto měření, nebo vybere měřící postup a tlačítkem vstoupí do formuláře nového měření.
2) Uživatel pracuje ve formuláři. Data ve formuláři jsou organizována do bloků. V záhlaví bloku jsou nabídky a pole pro vkládání dat, která jsou společná pro celý blok. V záhlaví bloku je volba pro hromadnou změnu výběru pracoviště, která se promítne do všech řádků bloku. V každém řádku je nabídka pracovišť. Po výběru pracoviště se automaticky vypočtou hodnoty v pracovišti. V závislosti na těchto hodnotách se postupně navolí rozsahy na jednotlivých zařízeních v pracovišti. Tyto rozsahy jsou zobrazeny vedle výběru pracoviště. Pokud není při otevírání měření nebo při jeho konfiguraci nalezeno některé zařízení nebo jeho rozsah, vypíše se místo identifikace zařízení či rozsahu chybové hlášení. Automaticky nalezené hodnoty lze ručně změnit.
3) Uživatel stiskne tlačítko uložit. Zobrazí se nabídka pro uložení měření. Vybere v nabídkách možnost přístupu k měření a měření, které se má přepsat současným měřením. V textovém poli může zadat nový název měření.
4) Po stisku tlačítka se měření uloží.

Nestandardní průběh

1) Pokud uživatel v bodě 1) nevybere v nabídce položku a stiskne tlačítko pro otevření měření, aplikace nereaguje.
2) Pokud v bodě 2) nezmění nastavení nabídek z defaultního nastavení na jiné (nevynutí provedení konfigurace pracoviště) a stiskne tlačítko pro uložení, zobrazí se upozornění, jinak aplikace nereaguje.
3) Pokud uživatel v bodě 3) nevybere v některé z nabídek žádnou položku, zobrazí se místo bodu 4) upozornění, jinak program nereaguje.

8 Cílová architektura

Při návrhu cílové architektury bylo hlavním cílem sladit co nejvíce aplikaci s mojí předchozí prací tak, aby byla co nejjednodušší budoucí integrace těchto dvou částí v jeden systém.

Stěžejním rozhodnutím je volba operačního systému. Protože v laboratoři se kterou jsem spolupracoval, používají již přes deset let pouze linuxové stroje, je i tato aplikace cílena na tento operační systém a o podpoře jiného OS se neuvažuje. Volba operačního systému pak implikuje i další komponenty cílové architektury.

8.1 Programovací jazyk

Jako programovací jazyk byl zvolen jazyk C99. Pro vývoj je použita GNU knihovna jazyka C - glibc, která do jazyka přináší další rozšíření pro ulehčení práce. Proto je nutné mít při překladu aplikace v makefile definované makro „D_GNU_SOURCE“.

8.2 Vrstva perzistence

Pro perzistentní vrstvu jsem použil SQL (Structured Query Language) databázi SQLite. Dokumentace k API je přístupná online na [SQL14]. Jedná se o svobodnou knihovnu, která do aplikace přidává embeded SQL server. S touto databází mám již dobré zkušenosti z předchozí práce, kde jsem ji používal na ukládání konfigurace sběrnic při automatickém měření. Tato databáze skýtá dvě hlavní výhody.

- Přes svoji jednoduchost obsahuje i pokročilé možnosti, na které jsme zvyklí u profesionálních databází. Implementuje v sobě skoro celý standard SQL-92 a obsahuje takové vlastnosti jako např. podporu transakcí. Výhoda je v tom, že dotazy odladěné v této databázi lze bez problémů použít i pro přístup do databáze s jiným databázovým strojem.
- Všechna data jedné databáze jsou uložena v jednom databázovém souboru uloženém na disku, který může obsahovat více tabulek. Soubor je ve formátu dbm (Data Base Manager) a je přenositelný mezi různými platformami. Zde je výhoda v tom, že při vývoji je možné mít celou databázi v jednom souboru a není nutné mít neustále k dispozici počítač se spuštěným databázovým strojem. Tím, že jsou soubory nezávislé na architektuře, je lze výhodně používat i pro přenos dat po síti.

SQLite tedy nemá architekturu klient-server jako většina jiných databází a konkurenční přístup k datům řeší na úrovni zamykání celého souboru, které však musí zajistit programátor prostředky operačního systému. SQLite obsahuje kolem dvou set různých funkcí.

Pro překlad kódu s SQLite je třeba do zdrojového souboru vložit hlavičkový soubor: **`#include <sqlite3.h>`** a při překladu přilinkovat knihovnu volbou **`-lsqlite3`**.

8.2.1 Otevření databázového souboru

Otevření databázového souboru se provede pomocí funkce **`sqlite3_open()`**, která má následující prototyp:

```
int sqlite3_open(char *file_name, sqlite3 **db)
```

Význam jednotlivých parametrů je následující:

`file_name` - jméno souboru s databází (může obsahovat i relativní cestu vztaženou k aktuálnímu adresáři)

`db` - adresa ukazatele na strukturu *sqlite3* - poté co je otevřeno spojení s databází, je do této adresy uložen ukazatel na strukturu *sqlite3*, která popisuje otevřené spojení.

8.2.2 Uzavření databázového souboru

```
int sqlite3_close(sqlite3 *db)
```

Jedná se vlastně o destruktor objektu *sqlite3*. Jediným parametrem je právě ukazatel na tuto strukturu.

8.2.3 Vykonání dotazu

Pro jednoduché vykonání dotazu bez jeho předchozí přípravy je možné použít funkci:

```
int sqlite3_exec(
db,
const char *query,
int (*callback)(void *p ,int argc,char**argv,char** names),
void *arg,
char **errmsg)
```

Význam jejích parametrů je následující:

`db` - identifikátor otevřené databáze vrácený funkcí *sqlite3_open()*.

`query` - řetězec obsahující text databázového dotazu.

`callback` - ukazatel na callback funkci, která bude zpracovávat výsledky dotazu.

`arg` - ukazatel na první parametr, který je předán callback funkci.

`errmsg` - adresa ukazatele na řetězec. Do této adresy je uložen ukazatel na řetězec s chybovým hlášením, pokud dojde při zpracování dotazu k chybě.

Funkce *sqlite3_exec* vlastně obaluje 3 funkce: s*qlite3_prepare_v2()*, *sqlite3_step()*, a *sqlite3_finalize()*, které se vnitřně používají pro vykonání dotazu. Oddělené použití těchto funkcí je z hlediska lepšího výkonu databáze nutné v případě, že se používá stejný dotaz pokaždé s jinými daty. Pak se dotaz nejprve přeloží funkcí

sqlite3_prepare_v2(), do přeloženého dotazu se pak pomocí funkce *sqlite3_bind()* vloží požadované hodnoty a dotaz se vykoná pomocí funkce *sqlite3_step()*. Dle mého názoru je vhodnější používat tento způsob i v případě dotazu, který vrací data. Je možné se tak vyhnout nutnosti použití calback funkce.

8.3 Grafické prostředí

Pro realizaci grafického uživatelského rozhraní (GUI) tohoto projektu byla zvolena svobodná grafická knihovna Gimp ToolKit (GTK+) ve verzi 3. Hlavně z důvodu kompatibility s jinými částmi postupně budovaného systému. Je to jedna z mála grafických knihoven, které jsou určeny pro jazyk C. Kromě dokumentace k API, která je distribuována spolu s knihovnou a je též přístupná online na [GTK14], jsou hodnotným zdrojem informací o GTK+ i různé knihy jako např. [AK07], která je volně dostupná v pdf nebo tutoriál [TM06].

Když mluvíme o GTK+, máme většinou na mysli komplexní grafické prostředí. Ve skutečnosti je ale GTK+ pouze jedna z mnoha knihoven, které grafické prostředí tvoří. Protože je ale GTK+ nejvíce vidět (je to ta část, pomocí které programátor vytváří grafické prvky), vžilo se zřejmě označení knihovny pro celý projekt grafického prostředí. Celý projekt je tvořen z několika hlavních a pomocných částí. Seznam těch hlavních je následující:

- **GLib** – rozšíření C knihovny o nové a přenositelné funkčnosti.
- **GObject** – GLib Object System - rozšíření knihovny GLib o plně objektové rozhraní.
- **GIO** - VFS API – abstraktní vysokoúrovňové rozhraní pro virtuální systém souborů umožnující programátorům přenositelný a jednotný přístup k různým zdrojům dat.
- **Pango** - knihovna pro vykreslování písma a zpracování textu v mezinárodním prostředí. Použité kódování je utf-8.
- **Cairo** – 2D renderovací grafická knihovna s podporou různých výstupních zařízení.
- **ATK -** Accessibility Toolkit – zajišťuje interakci s okolním prostředím.
- **GdkPixbuf** – malá knihovna umožňující manipulace s bitmapami, používá se např. na vytváření ikon v programu.
- **GDK** - GIMP Drawing Kit – knihovna, která je vložena jako abstraktní vrstva mezi GTK a grafickým serverem a která zajišťuje přenositelnost mezi různými grafickými systémy (platformami). Provádí vykreslování grafických komponent (widgetů).
- **GTK** - GIMP Toolkit – vlastní knihovna pro vytváření grafického uživatelského rozhraní. Umožňuje vytvářet a manipulovat s grafickými komponentami.

Některé z těchto částí jsou hluboko skryté a programátor s nimi běžně nepřijde do styku. Jiné naopak představují rozhraní pro pro psaní programu. S těmi by se měl programátor řádně seznámit.

8.4 GLib

GLib je multiplatformní knihovna utilit, která významně rozšiřuje funkčnost klasické knihovny C. Původně byla vyvíjena jako část GTK+, ale postupně se osamostatnila a nyní ji lze používat zcela samostatně, pokud programátor potřebuje obohatit program o některé šikovné možnosti, které knihovna obsahuje.

Osobně bych rozdělil přínos knihovny do třech oblastí:

- Obaluje možnosti klasické knihovny C, nikoli za účelem nových funkcí, ale za účelem lepší přenositelnosti kódu. GLib například definuje 8, 16, 32 a popř. i 64-bitový typ integer, u kterého je zaručeno, že bude mít na všech platformách stejnou velikost.
- Obaluje možnosti klasické knihovny C za účelem vylepšení funkčnosti klasických C funkcí. Jako příklad lze uvést nahrazení klasické funkce **`malloc()`** pro alokování paměti funkcemi **`g_new`** a **`g_slice_alloc`**, které sice také „pouze“ alokují paměť, ale dělají to způsobem, který ušetří paměť tím, že zabrání fragmentaci paměti.
- Přidává nové funkčnosti, které v klasické C knihovně nejsou a které jsou programátoři nuceni přidávat do svých programů vlastními silami. Přidává například dynamické řetězce, dynamická pole, spojové seznamy, binární stromy, hashovací tabulky a spoustu dalších abstraktních datových typů.

Z mnoha možností knihovny, bych se zde soustředil na ty prvky, které je nutné používat při práci s GTK+ nebo které jsem použil při vývoji.

8.4.1 Datové typy

Kvůli přenositelnosti byly zavedeny nové datové typy viz tab 8.1:

Typ	Popis	Přípustné hodnoty
`gint8`	8-bitový signed integer	–128 … 127
`guint8`	8-bitový unsigned integer	0 … 255
`gint16`	16-bitový signed integer	–32 768 … 32 767
`guint16`	16-bitový unsigned integer	0 … 65 535
`gint32`	32-bitový signed integer	–2 147 483 648 … 2 147 483 647
`guint32`	32-bitový unsigned integer	0 … 4 294 967 295
`gint64`	64-bitový signed integer	–9 223 372 036 854 775 808 … 9 223 372 036 854 775 807
`guint64`	64-bitový unsigned integer	0 … 18 446 744 073 709 551 615

Tabulka 8.1: Datové typy v glib

Vylepšení klasických datových typů viz tab. 8.2:

Typ	Popis
gpointer	pointer na void (void)
gconstpointer	konstantní pointer na void (const void*)
guchar	stejné jako **unsigned char**
guint	stejné jako **unsigned int**
gushort	stejné jako **unsigned short**
gulong	stejné jako **unsigned long**
gsize	stejné jako **unsigned long**

Tabulka 8.2: Nové názvy pro klasické typy

gconstpointer se používá ve funkčních prototypech funkcí k označení, že data nebudou ve funkci měněna.

8.4.2 Nová makra

Mezi mnoha novými makry jsou i makra **TRUE** a **FALSE**. Tato makra jsou hojně využívána v parametrech funkcí GTK+.

G_N_ELEMENTS() - vrací počet prvků staticky alokovaných polí

8.4.3 Memory management

Kromě toho, že GLib nabízí programátorovi ekvivalenty všech běžných funkcí pro alokování paměti, které jsou obsažené v klasické knihovně *libc*, přidává navíc ještě makro:

```
struct_type *g_new(struct_type, n_structs)
```

pro alokování paměti pro nové struktury. Makro alokuje paměť pro **n_structs** prvků typu **struct_type** a vrátí ukazatel, který je přetypován na daný typ.

Většina zmíněných funkcí má tu vlastnost, že pokud není možné požadovanou paměť alokovat, GLib okamžitě ukončí běh aplikace. Toto chování se může zdát logické, protože co ještě dělat, pokud zcela došla paměť. Dá se však namítnout, že i v takovéto situaci by bylo možné se pokusit uložit pracně vytvořená data a teprve potom aplikaci ukončit. To se zdá rozumné zvláště v případě, že alokace selhala při požadavku na poměrně velkou části paměti. V takovém případě, je více než pravděpodobné, že záchranná akce, která nebude potřebovat zdaleka tolik paměti uspěje. Pro tyto případy jsou zde tedy ještě varianty výše zmíněných funkcí, které aplikaci jednoduše neukončí. Programátor ovšem musí jako tradičně kontrolovat navrácený ukazatel na hodnotu NULL, aby pracoval s validní pamětí. Jako zástupce těchto funkcí zmíním funkci

```
struct_type *g_try_new(struct_type, n_structs)
```

Paměť alokovaná pomocí funkcí z knihovny Glibc by měla být uvolněna opět pouze za pomoci funkcí z této knihovny, například:

```
void g_free(gpointer memory)
```

8.4.4 Memory Slices

Účelem *memory slices*, je omezit fragmentaci paměti, ke které dochází při alokaci většího množství stejných kousků paměti na heapu. K fragmentaci dochází, protože není možné vždy obdržet kousek paměti, který by odpovídal přesně požadavku programátora. Většinou se přidělené kusy zarovnávají na určité celé jednotky, které jsou větší než požadovaný rozměr. *Slice* alokátor se s tímto problémem vypořádá tak, že alokuje najednou větší množství ucelené paměti, o kterém si vede přesnou evidenci a do tohoto prostoru již pouze směruje ukazatele, které vrací při požadavku na přidělení paměti.

Pro alokování paměti v programu pak může programátor použít dvě různé funkce:

```
g_slice_free1(gsize block_size, gpointer mem_block)
```

Tato funkce se používá při alokování většího množství stejně velkých objektů. Jako parametr požaduje velikost oblasti a vrací ukazatel na požadovanou paměť. Paměť získaná pomocí této funkce by měla být uvolněna pomocí funkce:

Další funkcí, která je vhodná pro alokování jednoho objektu je funkce

```
g_slice_new(type)
```

Tato funkce je vlastně makrem, které je nahrazeno voláním fce **`g_slice_alloc`** s hodnotou **`sizeof(type)`**. Vrací ukazatel, který je přetypován na daný typ:

```
#define g_slice_new(type)  ((type*) g_slice_alloc (sizeof (type))
```

Paměť získaná pomocí této funkce by měla být uvolněna opět pomocí makra.

8.4.5 Spojové seznamy

GLib obsahuje dva typy spojových seznamů: jednosměrně a obousměrně vázané. Jednosměrný seznam se liší od obousměrného hlavně tím, že lze procházet pouze jedním směrem. Jinak jsou oba druhy prakticky stejné, používají i stejné názvy funkcí s jedním malým rozdílem, že jednosměrně vázaný list obsahuje navíc v názvech písmenko „s“. Veškeré další informace jsou proto použitelné pro oba typy.

Spojový seznam (obousměrně vázaný) je tvořen za použití struktur GList:

```
struct GList {
  gpointer data;
  GList *next;
  GList *prev;
};
```

Tyto struktury jsou navzájem provázány odkazy propojujícími sousední prvky seznamu. Prvek data pak slouží jako odkaz na vlastní data uložená do prvku seznamu. Pro práci se seznamy slouží několik základních funkcí:

```
GList *g_list_prepend(GList *list, gpointer data);
GList *g_list_append(GList *list, gpointer data);
```

Tyto funkce slouží k vložení nového prvku na začátek nebo na konec seznamu. Pokud seznam před tím neexistoval, je vytvořen. Z toho důvodu neexistuje žádná funkce typu g_list_new. Funkce vrací nový ukazatel na seznam, který by měl být vždy uložen, protože hodnota ukazatele se při vkládání prvků může měnit. Funkce **`g_list_append`** prochází celý seznam, aby našla koncový prvek a neměla by být používána na rozsáhlé seznamy.

```
GList *g_list_insert(GList *list, gpointer data, gint position);
```

Pomocí této funkce je možné do seznamu vložit data na místo uvedené v parametru **`position`**.

```
GList *g_list_remove(GList *list, gconstpointer data);
GList *g_list_remove_all(GList *list, gconstpointer data);
```

První funkce odstraní ze seznamu prvek, který obsahuje zadaný ukazatel na data. Pokud existují dva prvky se stejným ukazatelem, je odstraněn jen první. Druhá funkce odstraní všechny výskyty prvku. Obě vrátí nový ukazatel na seznam.

```
GList *g_list_remove_link(GList *list, GList *link);
GList *g_list_delete_link(GList *list, GList *link);
```

Tyto dvě funkce slouží též k odstranění prvku ze seznamu, tentokrát je ovšem prvek identifikován pomocí ukazatele na prvek (rozdíl proti předchozí funkci, která vyžadovala ukazatel na data). První funkce prvek odpojí ze seznamu tak, že je s ním možno dále pracovat, druhá funkce jej zcela odstraní.

```
void g_list_free(GList *list);
void g_list_free_full(GList *list, GDestroyNotify free_func);
```

První funkce odstraní všechny prvky seznamu. Pokud prvky obsahují dynamicky alokovaná data, musí je programátor sám uvolnit. Proto existuje i druhá funkce, která má jako parametr funkci, která provede uvolnění paměti prvku.

9 Architektura aplikace

Pro aplikaci jsem použil klasický způsob návrhu architektury Model–View–Controller (MVC). MVC rozděluje aplikaci na tři logické části: Model, View a Controller. Každá z těchto částí má definováno, za co je v rámci aplikace odpovědná.

Jednotlivé vrstvy jsou od sebe v aplikaci co nejvíce oddělené pomocí zapouzdření vrstev uvnitř modulů, kde je jejich kód. To je docíleno tím, že maximální počet funkcí, které patří do daného modulu je definováno jako *static* a pouze nezbytná množina funkcí sloužících pro přístup zvenčí je umístěna v hlavičkovém souboru, aby mohla být přístupná vně modulu.

9.1 Model

Model je datovou základnou aplikace. Skládá se z těchto částí:

- Popis datové perzistentní vrstvy, který tvoří datový popis objektů reálného světa (v tomto případě měřících zařízení, pracovišť, atd.). Popis reálného světa je uskutečněn nadefinovanou strukturou jednotlivých tabulek SQL databáze a jejich vzájemnými vztahy, které jsou realizovány pomocí cizích klíčů tabulek.
- Funkcemi, které slouží pro přístup k perzistentní vrstvě a které tvoří rozhraní mezi perzistentní vrstvou a okolím.
- Formátem dat v paměti aplikace, který slouží pro přenos informací z a do datové vrstvy, a který je do jisté míry otiskem perzistentní vrstvy v paměti.

Všechny části aplikace, které manipulují s datovou vrstvou, mají zvláštní modul pro uložení funkcí datového rozhraní. Modul je realizován souborem, který má na konci svého názvu za posledním podtržítkem koncovku **`_dat.c`**.

9.2 View

Úkolem view (prezentační) vrstvy je prezentovat data obsažená v datové vrstvě, jakož i zachytávat události vzniklé interakcí uživatele s grafickým rozhraním.. Funkce této vrstvy je pokryta použitou grafickou knihovnou.

Grafická knihovna v sobě má též implementovaný vlastní model MVC. Například *combo* boxy nebo stromové pohledy mají striktně oddělenou prezentační a datovou vrstvu. Pokud se změní datová vrstva, pomocí zabudovaného kontroléru se automaticky překreslí grafický pohled na data. Tuto vlastnost knihovny z výhodou používám například při odstranění položky v seznamu. Po odstranění položky není zapotřebí znovu vytvářet celý seznam položek, ale stačí pouze občerstvit data v datovém modelu, který je v terminologii knihovny nazýván *GtkTreeModel* a tvoří univerzální model umožňující uchovávat a procházet data se stromovou strukturou.

Prezentační vrstva je v aplikaci oddělena od zbytku aplikace a tvořena moduly, které mají na konci svého názvu za posledním podtržítkem koncovku **`_gui.c`**. Vzhledem k tomu, že důraz v aplikaci je kladen právě na grafické prostředí a jeho interaktivitu, tvoří grafická vrstva podstatnou část aplikace.

9.3 Controler

Controler zajišťuje interakci mezi datovou a prezentační – view vrstvou a tvoří vlastní chování aplikace. Zahrnuje způsob detekce vzniku události a ovladače, které tvoří vlastní reakci na danou událost. Typy interakce jsou dvojího druhu.

- Interakce vyvolané změnou v datovém modelu. Jak jsem uvedl v předchozím oddíle, tento druh interakce ponechávám zcela v kompetenci použité grafické knihovny.
- Interakce vyvolané změnou v prezentační vrstvě. Zde je použita grafická knihovna jako detektor události. Knihovna umožňuje spojit aktivní grafické prvky se signálem pomocí funkce *g_signal_connect*. Pomocí této funkce se u daného grafického prvku zaregistruje ovladač události, který se aktivuje v případě, že tato událost nastane. Ovladač události je většinou tvořen programátorem vytvořenou funkcí. V určitých případech lze ale zaregistrovat i vhodné vestavěné funkce grafické knihovny.

Do této vrstvy lze zařadit vše, co nepatří do datové a grafické vrstvy. Proto kromě ovladačů událostí je zde i další část kódu, pro kterou se vžilo označení business logika. Hranice mezi tím, co je ještě pouze ovladač a co je business logika, není nijak přesně stanovena. Řídil jsem se tím, že ovladače události jsou zahrnuty do grafické vrstvy, protože většinou potřebují získat z prezentační vrstvy nějaké informace nutné pro jejich činnost a někdy i zpětně provádějí modifikaci grafického rozhraní.

Všechny ostatní funkce, které jsou naprosto nezávislé na prezentační vrstvě, jsem izoloval do dvou modulů. Jeden modul obsahuje obecné utility, které ani nutně nemusí přímo souviset s danou aplikací. Do druhého modulu jsem umístil funkcionalitu automatické konfigurace pracoviště.

10 Grafické rozhraní

Grafické rozhraní kopíruje úvahy rozvedené v předchozích kapitolách. Na základě těchto úvah jsou zapotřebí následující pohledy na zadávaná data:

1. Formulář pro zadání jednotlivých rozsahů měřidel. Formulář bude dělen do tabulek podle druhu měřené veličiny.
2. Celkový pohled na pracoviště – jednoduchý plánek pracoviště zobrazující předpokládané pozice měřidel, v kterém bude možné navolit na jednotlivé pozice určitá měřidla ze seznamu měřidel.
3. Formulář pro zadání postupu měření. Formulář bude dělen do tabulek podle druhu měřené veličiny.
4. Formulář pro vytvoření vazeb mezi měřícími body a pracovišti. Protože pro různé body měření je možné použít jiné měřící pracoviště, je nutné před měřením určit, které pracoviště se použije – tím vznikne kompletní konfigurace měření. Vazeb mezi postupy měření a pracovišti může být více kombinací. To znamená, že jeden měřící postup lze použít ve více konfiguracích.

10.1 Okno aplikace

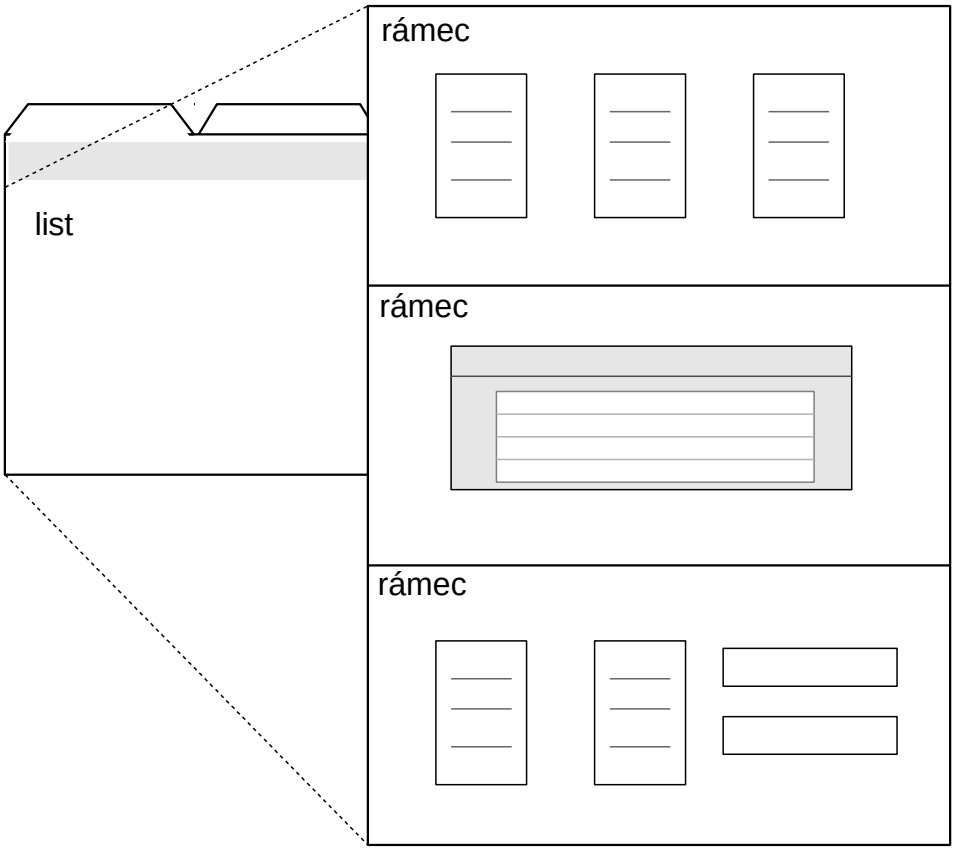
Z hlediska maximální flexibility je žádoucí, aby uživatel mohl provádět více činností najednou bez toho, aby jednu činnost musel přerušit kvůli jiné. Navrhl jsem proto design aplikace jako jeden velký sešit s vloženými listy, přičemž v každém listu je možné provádět jinou činnost.

Prakticky každá činnost nad daty je postupem určitých operací. I v nejjednodušším případě jsou většinou tři:

- Výběr dat, s kterými se bude pracovat.
- Vlastní práce s daty.
- Uložení změn v datech.

Pro první a třetí bod bývají v programech typu textových editorů otevírací okna, která umožňují výběr souboru k otevření a uložení. Protože předpokládám, že v této aplikaci bude zapotřebí složitější výběr, než je okno se seznamem souborů, rozhodl jsem se vstupní a výstupní nabídky umístit přímo do pracovního prostoru. Při zkoumání uživatelské aktivity v současném prostředí internetového prohlížeče, jsem zjistil, že uživatelé často používají tlačítko „zpět“ prohlížeče. Rozhodl jsem se tuto možnost zachovat. Abych se vyhnul zbytečnému opětovnému vytváření jednotlivých obrazovek při přechodu mezi nimi, rozdělil jsem každý list na více virtuálních ploch podobných virtuálním plochám v linuxových grafických prostředích. Na každou tuto virtuální plochu lze umístit jednu obrazovku a tyto pak jednoduše přepínat. Tuto virtuální plochu jsem nazval rámcem.

Celý systém ukazuje následující obrázek 10.1:



Obrázek 10.1: Logické uložení rámců v listu

Pro práci s listy a rámci jsou připraveny funkce z modulu **`gui.c`**.

Pro vložení nového listu se záložkou do sešitu slouží funkce:

```
int Tab_new(char *name)
```

Parametr **`name`** umožňuje při vytvoření nastavit název záložky.

Pro přidání nového rámce do záložky slouží funkce:

```
int Tab_insert_frame(T_FRAME index, GtkWidget *frame,
                GtkAlign halign, GtkAlign valign,
                GtkWidget *toolbar, void *data,
                int (*destroyer)(), int (*backfunct)() )
```

Funkce má následující parametry:

`index` - Pozice pro vložení. Počet pozic je omezen makrem.

`frame` – Grafický obsah rámce.

`halign, valign` – Zarovnání grafického obsahu v oblasti rámce.

`toolbar` – Lokální toolbar k vložení do horního panelu, jinak NULL.

`data` - Ukazatel na data rámce, pokud nejsou tak NULL.

`destroyer` - Ukazatel na destruktor dat v záložce.

`backfunct` - Ukazatel na funkci pro krok zpět.

Rámec se přidává vždy do právě viditelného listu. Konstrukci destruktoru dat jsem se pokusil sjednotit a zformalizovat pomocí maker **`FRAME_DATA_DESTRUCTOR()`** a **`FRAME_DATA_DESTRUCTOR_END`**, která obalují vnitřní logiku fungování destruktoru. Pokud je nebezpečí, že by s obsahem rámce mohla být nechtěně odstraněna neuložená data, je možné do konstruktoru vložit test, který je též formalizován. Test je uvozen makry **`FRAME_DATA_DESTRUCTOR_TEST`** a **`FRAME_DATA_DESTRUCTOR_TEST_END`**. Mezi tyto části lze umístit libovolný kód, který se např. zeptá uživatele, zda chce záložku zavřít a v případě požadavku na přerušení odstranění záložky použije makro **`FRAME_DATA_DESTRUCTOR_SKIP`**.

10.2 Návrh testovacího formuláře

Pro objektivní testování a posouzení možností je nutné vytvořit předběžný grafický návrh formuláře, který umožní ověřit všechny klíčové požadavky kladené na výsledné GUI.

10.2.1 Obecné požadavky na formuláře

Aby byla práce s daty co nejjednodušší a nejpřímější, je třeba nejprve nadefinovat základní požadavky, které by formuláře měly splňovat. Jedná se o obecné požadavky na GUI pro organizování tabulkových dat pro měření, které vychází nejen z mých vlastních poznatků, ale i ze zkušeností osob provádějících měření, s kterými jsem tuto problematiku konzultoval. Jako klíčové bych zdůraznil následující vlastnosti:

- Variabilita. Struktura dat ve formuláři by měla jít lehce měnit. Poměrně často se při měření stává, že je na základě nových zkušeností třeba upravit způsob měření. Pokud měřící formuláře nejsou variabilní, je nutné vytvořit jiné a znovu je pracně vyplňovat. Budu proto požadovat, aby bylo možné jednou vytvořený formulář pomocí jednoduchých úkonů opakovaně překonfigurovat (samozřejmě v rámci určitých pravidel vycházejících z obecných principů měření). Při rekonfiguraci, by měla ve formuláři zůstat všechna data obsažená v polích, která jsou obsažena jak v původní, tak i v nové konfiguraci.
- Trvanlivost. Při vytváření nových měřících postupů je nutno často experimentovat, než je měřící postup náležitě odladěn. Někdy je také problém s technickou dokumentací zařízení, která mají být objektem zkoušení. Dokumentace je mnohdy ztracená nebo neúplná a technická data se musí

dohledávat na internetu. Mnohdy se stane, že se měřící postup několikrát předělává, často se vrací zpět některá z původních konfigurací. Tento problém by částečně mohla vyřešit funkce návratu zpět, která by vracela zpět v čase jednotlivé provedené operace ve struktuře tabulky. Její implementace by však byla poměrně náročná a její použití méně přehledné.

Daleko lepší se mi jeví myšlenka uchovávat stále v GUI všechny vytvořené konfigurace spolu s daty a pouze je skrýt před zraky uživatele. Uživatel by tak mohl tabulku opět vrátit do původní konfigurace a skryté části tabulky by se opět zviditelnily i se svým obsahem. Při uložení tabulek se sice uloží pouze momentálně platná konfigurace, ale skryté prvky GUI neustále existují. Teprve při uzavření formuláře, dojde k jejich odstranění.

- Interaktivita. Změna konfigurace formuláře by se měla projevit ihned.
- Kontrola. Data ve formuláři by měla být průběžně kontrolována na překlepy a jiné odhalitelné chyby.

10.2.2 Konfigurace formuláře

Datový formulář by měl jít konfigurovat pomocí grafických prvků na stránce bez použití hlavní nabídky aplikace, nejlépe pomocí jednoduchého kliknutí myší.

10.2.3 Přidávání nových prvků do formuláře

O přidávání nových prvků do formuláře platí to, co bylo řečeno v předchozím bodě – maximální jednoduchost. Data budou organizována do tabulek dle různých způsobů měření. Tyto tabulky by mělo jít jednoduše vytvořit i odstranit.

Z praxe je zřejmé, že nejčastějším úkonem je vkládání nového řádku do tabulky. Proto je na místě požadavek, aby šlo po naplnění posledního řádku tabulky snadno přidat další řádek bez toho, že by uživatel musel přehmátnout rukou z klávesnice na myš. Nejintuitivnějším řešením bude vložení nového řádku pomocí stisku klávesy Enter na konci předchozího řádku. Tento způsob sám nestačí, protože takto není možné vložit nový řádek před první řádek v tabulce. Použiji proto kombinaci více možností:

- Kontextová nabídka po stisku pravého tlačítka myši nad existujícím řádkem (nad číslem řádku).
  - přidat nový řádek před
  - přidat nový řádek za
  - kopírovat řádek
  - vyjmout řádek
  - vložit řádek před
  - vložit řádek za
- Stisk klávesy Enter po vyplnění existující řádky přidá řádek za.

Protože data se často zadávají po řádcích, je zapotřebí, aby se po stisku tabulátoru kurzor přenesl z právě editované buňky tabulky do buňky napravo od současného umístění.

Při vytváření modelových formulářů měření v tabulkovém procesoru jsem si všiml, že se data v jednotlivých tabulkách formuláře často opakují. Původně jsem proto zvažoval umožnit vytváření nových tabulek pomocí kopírování tabulek již hotových. Po konzultacích jsem od tohoto záměru upustil, neboť jsem byl ujištěn, že toto je cesta, která sice zpočátku práci urychlí, ovšem za cenu vzniku velkého množství chyb, jejichž oprava si vyžádá daleko více času. Chyby vznikají tím, že uživatel opomene ve zkopírované tabulce opravit některé hodnoty, které mají být jiné, než v tabulce původní.

10.2.4 Návrh tabulky formuláře

Formulář by měl obsahovat libovolné množství tabulek, jejichž návrh obsahuje položky dle následujícího obrázku. Též počet řádků bude neomezen.

Číslo bloku	▼B Skrýt blok	↓B Přidat blok	xB Odstranit blok	Výběr ukazatele	Výběr funkce vstup	Výběr funkce	Výběr funkce výstup	Výběr třídy

Ř	Název rozsahu	Rozpětí rozsahu			Vstupní rozpětí			Hodnota rozsahu		Výstupní rozpětí			Počet digitů na max-hodnotu	Délka rozsahu v dílcích	Délka stupnice v [mm]	Skutečná hodnota	Rozpětí parametru			Zadání chyb měřidla

Tabulka 10.1: Logické členění tabulky formuláře

Návrh formuláře dle tab. 10.1 vychází z požadavků na formulář pro zadávání rozsahů měřidel. Tabulka obsahuje proti cílovému stavu zjednodušenou část sloupců zadání chyb měřidla, ale pro testování postačí, protože je na ní možné ověřit všechny funkce, které jsou popsány výše.

Řádky, které jsou uvedené v grafickém návrhu, představují „maximální možnosti“ tabulky a nikdy nebudou zobrazeny všechny současně. Jednotlivé konfigurace tabulky se budou lišit různými kombinacemi zobrazených sloupců, dle výběru druhu měřidla, který bude proveden v liště nad tabulkou.

10.3 Možnosti knihovny GTK+

Grafická knihovna GTK+ obsahuje dva základní nástroje pro zobrazování tabulkových dat. První možností je *GtkTreeView*, druhou je *GtkGrid*. Nezbývá, než pomocí prototypů zjistit, která z variant se bude více hodit pro daný účel.

10.3.1 GTKGrid

GtkGrid patří do skupiny primitiv *Layout Containers*. Nejedná se tedy o widget v pravém slova smyslu, ale pouze o neviditelný kontejner, který může obsahovat a organizovat další widgety. GtkGrid není v GTK+ dlouho. Objevil se až v nedávné verzi 3.2. Dříve jeho funkci zastával widget GtkTable.

Jak již název napovídá, oba tyto widgety slouží k organizování dat do tabulek, a proto se staly mojí první volbou pro využití při vytváření formulářů. GtkTable stále v GTK+ existuje, ale je určen k budoucímu vyřazení. Měl jsem tu možnost s tímto widgetem již pracovat a musím dát za pravdu vývojářům, kteří se tento widget rozhodli nahradit něčím lepším. V GtkGrid byl upraven způsob, jakým se zapisuje umístění jednotlivých prvků v tabulce. Na rozdíl od GtkTable není nutné přepisovat tolik čísel při přeorganizování obsahu tabulky.

Nový grid se vytvoří velmi jednoduše pomocí funkce *gtk_grid_new*:

```
GtkWidget *gtk_grid_new(void);
```

Není ani potřeba zadávat počet sloupců a řádků jako dříve. Grid je dostatečně plastický a přizpůsobí se pozdějším změnám. Pokud je widget takto vytvořen, je do něj možné začít vkládat obsah pomocí funkce *gtk_grid_attach*:

```
void gtk_grid_attach(GtkGrid *grid, GtkWidget *child, gint left, gint top,
                                                gint width, gint height);
```

Tato funkce vloží do gridu widget – potomka na pozici, která je daná parametry *left* a *top*. Parametry *width* a *height* slouží k informaci, kolik buněk tabulky do šířky a na výšku vložený widget obsadí. Vynikající je, že widgety není potřeba do gridu vkládat postupně tak, jak jdou za sebou, ale zcela libovolně a grid se vždy svými rozměry automaticky přizpůsobí.

Pokud jsou již všechny řádky a sloupce tabulky obsazené, není možné na tato místa vkládat další widgety, ale je nejprve nutné přidat další řádek resp. sloupec pomocí k tomu určených funkcí:

```
void gtk_grid_insert_row(GtkGrid *grid, gint position);
void gtk_grid_insert_column(GtkGrid *grid, gint position);
```

Pokud se takto vytvoří v tabulce nový prostor, jsou ostatní řádky a sloupce automaticky přečíslovány a jejich obsah je posunut po vložení obsahu do nových prázdných míst.

Takto lze pomocí několika jednoduchých funkcí vytvořit libovolné rozmístění jakýchkoliv widgetů a vytvářet i poměrně komplikované struktury. Pokud však programátor vyžaduje nějakou další funkcionalitu, jako např. záhlaví tabulky, linky v tabulce, barevné buňky či nějakou interaktivitu, musí si vše zařídit sám pomocí ostatních prostředků knihovny.

10.3.2 GTKTreeView

GtkTreeView je widget, který je určen k zobrazování dat, která jsou organizována ve formě seznamu nebo stromu. Uživatel může seznam procházet a vybrat patřičný řádek. Tento widget je často používán pro výběr jedné z několika předem definovaných možností a je také základem pro rozbalovací nabídku – combo box. Data v seznamu je ovšem možné i editovat, a proto by mohl být vhodný i pro použití při konfiguraci měřícího pracoviště. Podobně se GtkTreeView používá např. v aplikaci *MySQL Workbench*, kde slouží pro prohlížení a editaci textových dat v databázových tabulkách.

Koncept GtkTreeView je založen na modelu MVC (Model View Controler). Vrstva modelu je tvořena pomocí rozhraní *GtkTreeModel*. Toto rozhraní má několik implementací. Nejpoužívanější jsou implementace *GtkListStore* a *GtkTreeStore*, které jsou určeny k tomu, aby tvořily datovou vrstvu pro seznamy, respektive pro stromy. Model je propojen s pohledem a jakákoli změna v modelu se hned promítne do pohledu. Jeden model může být použit ve více pohledech a změny se pak promítnou do všech připojených pohledů. Kontrolér pak slouží k obsluze uživatelských akcí nad pohledem.

Vytvoření GtkTreeView probíhá v několika krocích. Nejprve se většinou vytváří model např. pomocí funkce *gtk_list_store_new*:

```
GtkListStore *gtk_list_store_new(gint n_columns, ...);
```

Tato funkce přebírá počet vytvářených sloupců v modelu a seznam jejich typů a vrací ukazatel na nové *store*. Počet sloupců modelu může být vyšší než počet sloupců v pohledu, protože některé sloupce se nemusí zobrazovat, ale mohou sloužit pro nastavení vzhledu sloupců viditelných (např. barvy a typu písma). Příklad volání této funkce může být např. následující:

```
gtk_list_store_new(3, G_TYPE_INT, G_TYPE_STRING, GDK_TYPE_PIXBUF)
```

Nyní je vhodné store naplnit daty, která se mají zobrazit v pohledu, i když data je také možno vkládat do již hotového pohledu. K počátečnímu naplnění store se používá většinou funkce *gtk_list_store_append*:

```
void gtk_list_store_append(GtkListStore *list_store, GtkTreeIter *iter);
```

Tato funkce přidá na konec store další prázdný řádek a do proměnné *iter* uloží iterátor, pomocí kterého lze k tomuto řádku přistupovat pomocí dalších funkcí. Existují také funkce, které umožní vkládat nový řádek kamkoli do existujícího store. Pokud máme do store vložen nový řádek, můžeme jej naplnit daty pomocí funkce *gtk_list_store_set*:

```
void gtk_list_store_set(GtkListStore *list_store, GtkTreeIter *iter, ...);
```

Tato funkce vloží na místo dané iterátorem hodnoty, které jsou předané jako seznam dvojic. První hodnota v dvojici představuje číslo sloupce, kam se data mají uložit a druhá hodnota jsou vlastní data.

Po vytvoření store se pomocí funkce *gtk_tree_view_new_with_model* vytvoří nový *pohled*, ke kterému se zároveň připojí vytvořené store:

```
GtkWidget *gtk_tree_view_new_with_model(GtkTreeModel *model);
```

Vytváření stromové struktury je trochu komplikovanější, protože je nutné pracovat se dvěma iterátory. Druhý iterátor svazuje řádek s jeho rodičem ve stromové struktuře.

Takto vytvořený pohled je ovšem ještě prázdný a nyní je do něj třeba vložit jednotlivé sloupce. Aby se ve sloupcích mohlo něco zobrazit, je nejprve nutné vytvořit tzv. *renderer*, který slouží k vykreslování obsahu sloupce z modelu. Rendererů je několik druhů podle toho, co se bude v daném sloupci vykreslovat. Programátor musí dbát na to, aby typ rendereru odpovídal datovému typu v modelu, jinak výsledek nesplní očekávání. Teprve vložením rendereru do pohledu a nastavením typu renderovaných dat vznikne v pohledu nový sloupec. Pro tento účel se používá několik dalších funkcí:

```
gtk_cell_renderer_text_new();
gint gtk_tree_view_insert_column_with_attributes(GtkTreeView *tree_view,
     gint position, const gchar *title, GtkCellRenderer *cell, ...);
```

Toto je zjednodušený popis GtkTreeView, který mnohdy vystačí, pokud programátor vytváří jednoduché nabídky. I tak je již dosti komplikovaný. Je to daň za určitou univerzálnost, která má umožnit pohled různě modifikovat a programátorům pokročilým v GTK+ pomoci vytvářet svoje vlastní widgety založené na GtkTreeView.

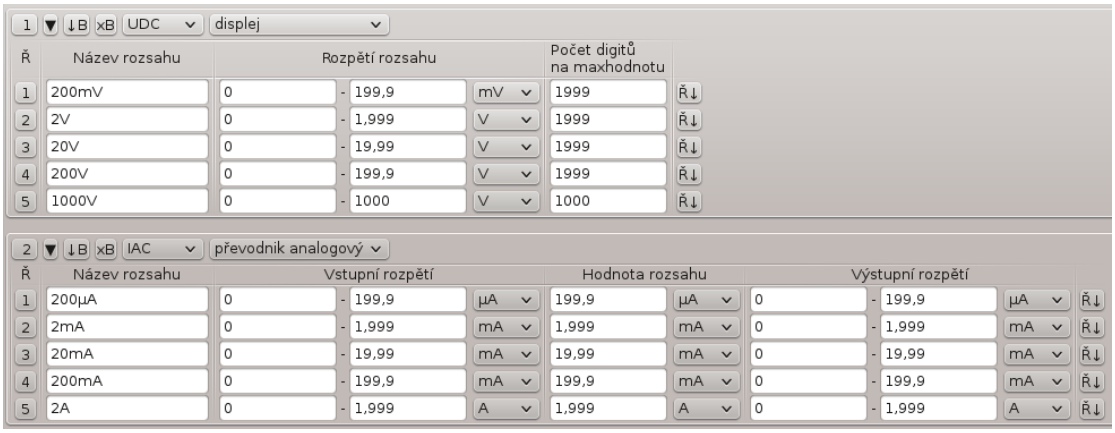
10.4 Prototypy

Abych se mohl rozhodnou, které prostředky z knihovny GTK+ využiji pro tvorbu formulářů, vytvořil jsem pro každý druh prototyp, který vychází z návrhu formuláře, který byl představen v předchozí kapitole. Stanovil jsem si několik kritérií pro výběr nejlepší možnosti:

- **Flexibilita.** I když jsem se snažil dát do návrhu formuláře vše, co by mělo být potřeba pro úspěšné završení další práce na kalibračních formulářích, přece jen nelze v začátku dohlédnout na samý konec práce, a tak je vhodné pomýšlet i na to, jak který způsob omezuje další možné úpravy ve vymyšleném konceptu někdy v budoucnosti.
- **Jednoduchost ovládání.** Ergonomie ovládání jednotlivých widgetů je to, co především zajímá uživatele aplikace a programátor by měl vycházet vstříc.
- **Náročnost na zdroje.** Dá se předpokládat, že aplikace poběží i na slabších strojích, protože při měření se mohou používat různé notebooky nebo starší počítače, které jsou vyřazeny z použití jako kancelářský desktop a slouží jako jednoúčelové technologické PC.
- **Vzhled.** Je to méně důležité kriterium, ale může se stát jazýčkem na vahách při rozhodování, který koncept zvolit.

10.4.1 Prototyp s GTKGrid

GTKGrid jsem se rozhodl vyzkoušet jako první, díky jeho univerzálnosti. Předpokládal jsem, že v tomto konceptu budu schopen zrealizovat vše, co jsem si naplánoval, a pak jej budu moci porovnávat. Výsledek je vidět na následujícím obrázku 10.2:



Obrázek 10.2: Prototyp GUI s GTK Grid

Výsledek není špatný. Myslím, že se podařilo naplnit vizi. Přece jen jsem však narazil na problémy. Zde je výčet vlastností, které charakterizují tento model:

Klady:

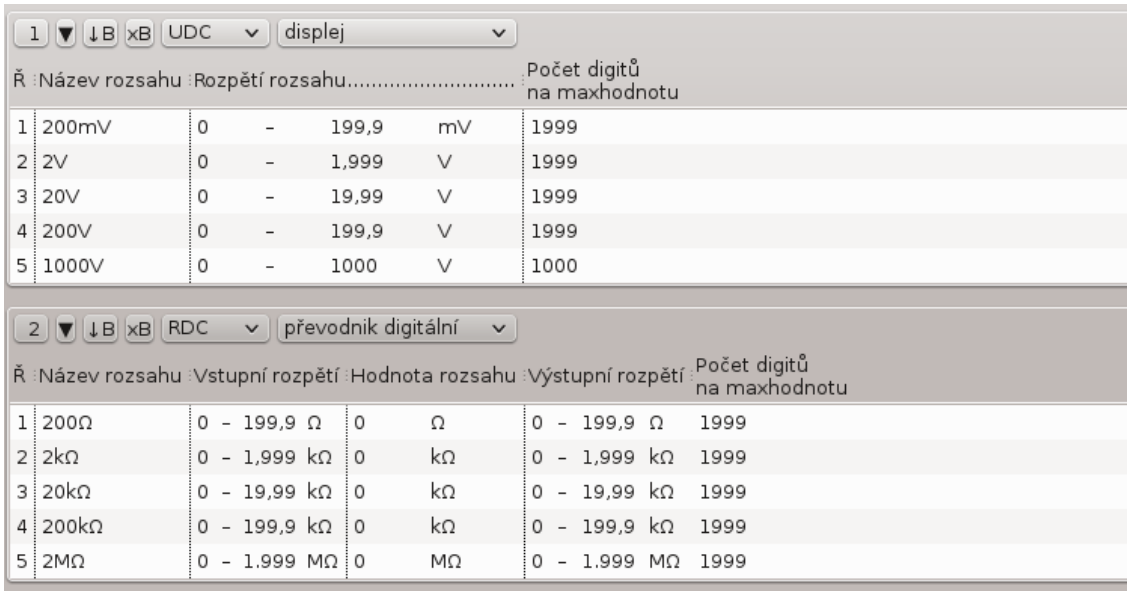
- **Flexibilita.** Jak jsem předpokládal, pomocí samostatných widgetů umístěných do gridu, jsem vytvořil vše, co jsem potřeboval (s ohledem na možnosti grafické knihovny).
- **Jednoduchost ovládání.** Podařilo se naplnit předpokládaný způsob ovládání.

Zápory:

- **Náročnost na zdroje.** Bohužel, tento koncept je poměrně náročný na vykreslování grafických prvků na obrazovku. Tím, že je složen z jednotlivých widgetů vložených do gridu, se při otevření formuláře a při rolování stránky widgety dlouho překreslují a při rolování stránka „zadrhává“.
- **Vzhled.** I když vzhledu nelze zatím nic vytknout, přece jen, při srovnání s GTKTreeView, které bude uvedeno dále, je tento koncept trochu slabší.

10.4.2 Prototyp s GTKTreeView

Jak vypadá formulář s GTKTreeView je vidět na následujícím obrázku 10.3:



Obrázek 10.3: Prototyp GUI s GTK TreWiew

Mé očekávání bylo, že tento koncept by mohl odstranit nedostatky konceptu předchozího. To se potvrdilo, bohužel se ovšem vyskytly problémy jiné. I když GTKTreeView obsahuje vlastní expander pro skrývání obsahu listu, není možné tento expander použít pro mé účely, protože do řádku expanderu je možné vkládat pouze obsah, který odpovídá obsahu vlastního listu. Proto jsem ovládací prvky tabulek použil tak, jak byly navrženy v předchozím konceptu. Zde je seznam dalších vlastností formuláře s GTKTreeView:

Klady:

- **Náročnost na zdroje.** Zde je GTKTreeView excelentní. Je vidět, že se jedná o jeden kompaktní widget, jehož obsah je knihovnou vykreslován pomocí nízkoúrovňových operací renderovací vrstvy. Posuv formuláře je opravdu naprosto plynulý.
- **Vzhled.** Vzhled je subjektivně znatelně lepší než v předchozím případě.

Zápory:

- **Flexibilita.** Jak jsem již popisoval v oddíle o GTKTreeView, každý sloupec musí mít uložen vlastní renderer, který vykresluje obsah políčka. I když je rendererů několik druhů, chybí například možnost vložit do políčka tlačítko. Další problém nastal při vložení rozbalovací nabídky – combo-boxu. Zatímco běžný combo-box z GTK+ knihovny vložený kamkoliv jinam, než do GTKTreeView, zobrazuje hodnoty z jednoho sloupce a při výběru vrací hodnotu

ze sloupce, který je označen jako „ID“ řádku, tak u combo-boxu vloženého do GTKTreeView se vrací vždy hodnota ze sloupce, který je vidět v buňce. Není proto možné svázat některé komplikovanější volby s ID a pracovat s nimi jeho prostřednictvím.

- **Jednoduchost ovládání.** Při kliknutí na jakékoli políčko v listu se prvním klikem nejprve označí daný sloupec a teprve při druhém kliku dojde k přeměně políčka na pole pro editaci textu. Ještě horší je to u combo-boxu. K výběru hodnoty jsou zde zapotřebí čtyři kliknutí. Při editaci také nejde přeskočit na vpravo umístěné políčko pomocí tabelátoru.

10.4.3 Upravený GTKGrid

Po vyzkoušení předchozích dvou konceptů jsem nebyl moc spokojený, protože ani jeden nesplňoval zcela nastavená kritéria. GTKGrid se mi zamlouval svou funkčností, ale měl jsem obavy, jak se bude chovat na slabších počítačích při stovkách řádků.

Proto jsem začal trochu experimentovat a pomocí jednoduchých cyklů jsem zkoušel generovat formuláře s tisíci políčky. Jako hranici jsem si určil 4000 polí, což by mělo odpovídat formuláři s 200 řádky náročnější přístrojové konfigurace.

Zjistil jsem, že formulář se chová znatelně lépe, pokud se na stránku místo widgetu GTKEntry, který slouží pro editaci textu, vloží pouze text GTKLabel. Protože v GTK+ nemůže label zpracovat klik myší, použil jsem tlačítko (GTKButton) s odstraněným reliéfem, aby se lépe renderovalo.

Výsledek byl uspokojivý a tak jsem navrhl další koncept, kdy do formuláře vkládám pouze plochá tlačítka a teprve po kliknutí na tlačítko se buňka přemění na GTKEntry a text lze editovat. Snížila se tím grafická náročnost aplikace a jako vedlejší efekt se zvedla i grafická stránka konceptu, jak je vidět na následujícím obrázku 10.4:

Obrázek 10.4: Prototyp GUI s GTK Grid - upravená verze

Pro pořádek ještě shrnu vlastnosti tohoto konceptu:

- **Flexibilita.** Úpravou konceptu ani v nejmenším neutrpěla a je stále na maximální úrovni danou možnostmi grafické knihovny.
- **Náročnost na zdroje.** Tato vlastnost sice není stále silnou stránkou tohoto konceptu, ale neměla by omezovat uživatele. Navíc je možné, že půjde tuto vlastnost ještě vylepšit při detailnějším rozpracování tohoto konceptu nebo se zlepší s tím, jak začne dozrávat poměrně nově zavedený widget GTKGrid.
- **Jednoduchost ovládání.** Zde došlo k zachování dobrých vlastností konceptu s GTKGrid i přes to, že bylo nutné dodat handler pro změnu tlačítka na textový vstup. Počty kliknutí pro editaci položek jsou poloviční ve srovnání s GTKTreeView.
- **Vzhled.** Subjektivně si myslím, že vizuální stránka se zlepšila a tak se tento koncept stává srovnatelný s GTKTreeView. Zároveň je cesta otevřená i pro další grafické efekty, které by v GTKTreeView nemusely být možné.

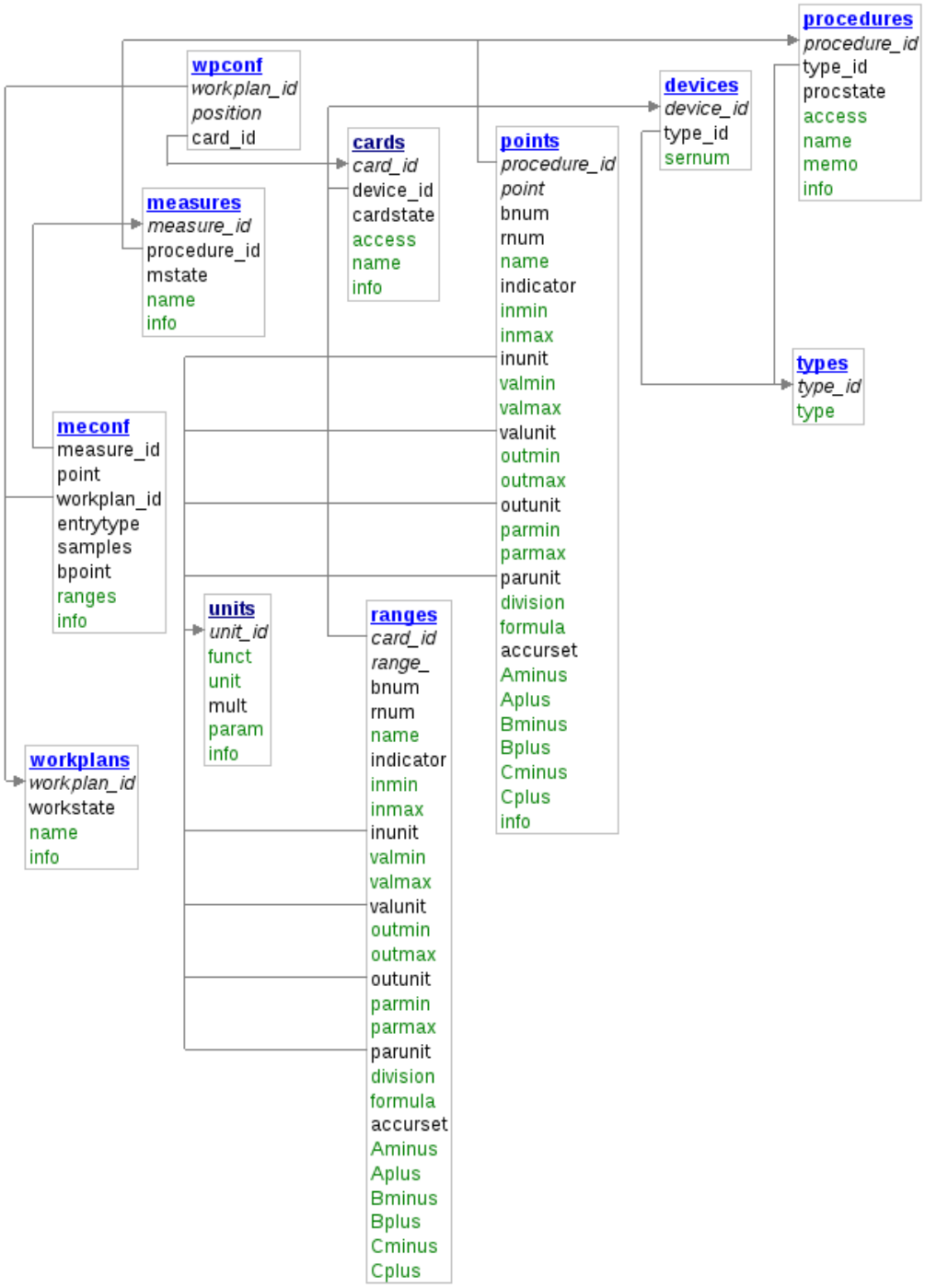
10.5 Závěr

Myslím, že jsem nakonec vybral dobré řešení, i když nesplňuje všechna kritéria zcela na sto procent. Ideálním řešením by mohlo být napsání vlastního, specializovaného widgetu, který by byl podobný prostředí, které se používá pro zobrazování sešitů v tabulkových procesorech. Vzhledem k náročnosti pochopení nízkoúrovňových funkcí grafické knihovny GTK+ by však nároky takového projektu několikrát přesáhly dostupné možnosti.

Věřím, že navržené řešení bude dobrou volbou a uspokojí požadavky uživatelů na funkční a uživatelsky přívětivé grafické uživatelské rozhraní.

11 Datový model

Datový model uchovává všechna shromažďovaná data a modeluje vztahy mezi nimi, jak je vidět na následujícím obrázku 11.1. Obrázek vznikl vizualizací skriptu *confplace.sql*, který je součástí distribuce programu, pomocí nástroje *Adminer*. Pomocí skriptu je vytvořen databázový soubor, s kterým program dále pracuje.



Obrázek 11.1: Schéma datového modelu

11.1 Popis tabulek datového modelu

11.1.1 Měřící jednotky

Tabulka uchovává informace o měřících jednotkách, které jsou používány v aplikaci. Důležité je označení jednotek stejně jako jejich velikost, která je nutná pro přepočet při vzájemném porovnávání různých jednotek stejné veličiny.

```
CREATE TABLE units(
     unit_id INTEGER PRIMARY KEY,
     funct TEXT NOT NULL,             Funkce (veličina)
     unit TEXT NOT NULL,              jednotka veličiny
     mult REAL NOT NULL,              násobitel
     param TEXT NOT NULL,             parametr
     info TEXT NOT NULL               poznámka
);
```

11.1.2 Typy zařízení

Zde jsou uloženy informace o typech (továrních), všech zařízení používaných v systému jak ve funkci měřeného zařízení, tak ve funkci zařízení měřícího.

```
CREATE TABLE types (
     type_id INTEGER PRIMARY KEY,
     type TEXT NOT NULL               typové označení
);
```

11.1.3 Zařízení

Tabulka zařízení přidává k údaji o typu údaj o výrobním čísle zařízení a umožňuje tím jednoznačnou identifikaci jak měřících, tak měřených zařízení.

```
CREATE TABLE devices (
     device_id INTEGER PRIMARY KEY,
     type_id INTEGER NOT NULL,
     sernum TEXT NOT NULL,            výrobní číslo
     FOREIGN KEY(type_id) REFERENCES types(type_id)
);
```

11.1.4 Karty

Karta slouží k popisu měřících schopností měřícího zařízení. První tabulka obsahuje informace o jednotlivých rozsazích zařízení, které je předmětem karty.

```
CREATE TABLE ranges (
     card_id INTEGER NOT NULL,
     range INTEGER NOT NULL,            číslo rozsahu
     bnum INTEGER NOT NULL,             číslo bloku
     rnum INTEGER NOT NULL,             číslo řádku
     name TEXT NOT NULL,                název rozsahu
     indicator INTEGER NOT NULL,        typ rozsahu
     inmin REAL NOT NULL,               vstupní rozpětí
     inmax REAL NOT NULL,
     inunit INTEGER NOT NULL,
     valmin REAL NOT NULL,              rozpětí rozsahu
     valmax REAL NOT NULL,
     valunit INTEGER NOT NULL,
     outmin REAL NOT NULL,              výstupní rozpětí
     outmax REAL NOT NULL,
     outunit INTEGER NOT NULL,
     parmin REAL NOT NULL,              rozpětí parametru
     parmax REAL NOT NULL,
     parunit INTEGER NOT NULL,
     division INTEGER NOT NULL,         dělení ukazatele
     formula TEXT NOT NULL,
     accurset INTEGER NOT NULL,         způsob zadání třídy
     Aminus TEXT NOT NULL,              třídy
     Aplus TEXT NOT NULL,
     Bminus TEXT NOT NULL,
     Bplus TEXT NOT NULL,
     Cminus TEXT NOT NULL,
     Cplus TEXT NOT NULL,
     PRIMARY KEY(card_id, range),
     FOREIGN KEY(card_id) REFERENCES cards(card_id),
     FOREIGN KEY(inunit) REFERENCES units(unit_id),
     FOREIGN KEY(valunit) REFERENCES units(unit_id),
     FOREIGN KEY(outunit) REFERENCES units(unit_id),
     FOREIGN KEY(parunit) REFERENCES units(unit_id)
);
```

Druhá tabulka obsahuje kromě jiného identifikaci karty.

```
CREATE TABLE cards (
     card_id INTEGER PRIMARY KEY,
     device_id INTEGER NOT NULL,
     cardstate INTEGER NOT NULL DEFAULT 0,         pracovní stav karty
     access TEXT NOT NULL,                         přístup k zařízení
     name TEXT NOT NULL,                           název karty
     info TEXT NOT NULL,                           poznámka
     FOREIGN KEY(device_id) REFERENCES devices(device_id)
);
```

11.1.5 Pracoviště

Pracoviště popisuje, na které pozici v plánu pracoviště se nachází jaké zařízení.

První tabulka obsahuje hlavně identifikaci pracoviště.

```
CREATE TABLE workplans (
     workplan_id INTEGER PRIMARY KEY,
     workstate INTEGER NOT NULL DEFAULT 0,         stav pracoviště
     name TEXT NOT NULL,                           název
     info TEXT NOT NULL                            poznámka
);
```

Druhá tabulka obsahuje umístění zařízení v pracovišti.

```
CREATE TABLE wpconf (
     workplan_id INTEGER NOT NULL,
     position INTEGER NOT NULL,                    číslo pozice
     card_id INTEGER NOT NULL,                     ID karty
     PRIMARY KEY(workplan_id, position),
     FOREIGN KEY(workplan_id) REFERENCES workplans(workplan_id),
     FOREIGN KEY(card_id) REFERENCES cards(card_id)
);
```

11.1.6 Postupy měření

Postupy měření popisují jednotlivé měřící body (odměry).

První tabulka obsahuje hlavně identifikaci postupu:

```
CREATE TABLE procedures (
     procedure_id INTEGER PRIMARY KEY,
     type_id INTEGER NOT NULL,                    typ zařízení
     procstate INTEGER NOT NULL DEFAULT 0,        pracovní stav popisu
     access TEXT NOT NULL,                        přístup k zařízení
     name TEXT NOT NULL,                          název postupu
     memo TEXT NOT NULL,
     info TEXT NOT NULL,                          poznámka
     FOREIGN KEY(type_id) REFERENCES types(type_id)
);
```

Druhá tabulka obsahuje jednotlivé měřící body:

```
CREATE TABLE points (
     procedure_id INTEGER NOT NULL,
     point INTEGER NOT NULL,                 číslo bodu
     bnum INTEGER NOT NULL,                  číslo bloku
     rnum INTEGER NOT NULL,                  číslo řádku
     name TEXT NOT NULL,                     název rozsahu
     indicator INTEGER NOT NULL,             typ rozsahu
     inmin REAL NOT NULL,                    vstupní rozpětí
     inmax REAL NOT NULL,
     inunit INTEGER NOT NULL,
     valmin REAL NOT NULL,                   rozpětí rozsahu
     valmax REAL NOT NULL,
     valunit INTEGER NOT NULL,
     outmin REAL NOT NULL,                   výstupní rozpětí
     outmax REAL NOT NULL,
     outunit INTEGER NOT NULL,
     parmin REAL NOT NULL,                   rozpětí parametru
     parmax REAL NOT NULL,
     parunit INTEGER NOT NULL,
     division INTEGER NOT NULL,              dělení ukazatele
     formula TEXT NOT NULL,
     accurset INTEGER NOT NULL,              způsob zadání třídy
     Aminus TEXT NOT NULL,                   třídy
     Aplus TEXT NOT NULL,
     Bminus TEXT NOT NULL,
     Bplus TEXT NOT NULL,
     Cminus TEXT NOT NULL,
     Cplus TEXT NOT NULL,
     info TEXT NOT NULL,                     poznámka
     PRIMARY KEY(procedure_id, point),
     FOREIGN KEY(procedure_id) REFERENCES procedures(procedure_id),
     FOREIGN KEY(inunit) REFERENCES units(unit_id),
     FOREIGN KEY(valunit) REFERENCES units(unit_id),
     FOREIGN KEY(outunit) REFERENCES units(unit_id),
     FOREIGN KEY(parunit) REFERENCES units(unit_id)
);
```

11.1.7 Měření

Měření spojuje dohromady informace z předchozích tabulek a vytvoří kompletní popis měření.

První tabulka obsahuje hlavně identifikaci měření a propojení na postup měření:

```
CREATE TABLE measures (
     measure_id INTEGER PRIMARY KEY,
     procedure_id INTEGER NOT NULL,                propojený postup
     mstate INTEGER NOT NULL DEFAULT 0,            pracovní stav měření
     name TEXT NOT NULL,                           název
     info TEXT NOT NULL,                           poznámka
     FOREIGN KEY(measure_id) REFERENCES measures(measure_id),
     FOREIGN KEY(procedure_id) REFERENCES procedures(procedure_id)
);
```

Druhá tabulka vytváří vazby mezi body postupu a měřícími zařízeními:

```
CREATE TABLE meconf (
     measure_id INTEGER NOT NULL,
     point INTEGER NOT NULL,                       číslo bodu měření
     workplan_id INTEGER NOT NULL,                 použité pracoviště
     entrytype INTEGER NOT NULL DEFAULT 0,         typ vstupu hodnoty
     samples INTEGER NOT NULL DEFAULT 1,           počet odměrů
     bpoint INTEGER NOT NULL DEFAULT 0,            break point
     ranges TEXT NOT NULL,                         popis rozsahů
     info TEXT NOT NULL,                           poznámka
     FOREIGN KEY(workplan_id) REFERENCES workplans(workplan_id)
);
```

11.2 Integrita dat

Všude, kde je to rozumné, jsem se snažil uchránit integritu dat pomocí cizích klíčů. V některých případech by však lpění na datové integritě mohlo způsobit nepoužitelnost celého systému. Jde hlavně o vztah mezi měřeními, která v sobě koncentrují informace z celého datového modelu a zbytkem modelu.

Dle zkušeností může být typů zařízení v systému několik set až tisíc. Tomu přibližně odpovídá i počet postupů. Ke každému postupu v průměru existují dvě měření. Pokud byl byla zavedena naprostá datová integrita, musel by uživatel, který by chtěl odstranit některý rozsah měřícího zařízení, napřed odstranit tento rozsah ze všech měření, kterých

může být až několik desítek tisíc. Kaskádové odstraňování záznamů na základě cizích klíčů by mohlo problém vyřešit. Místo toho jsem se vydal cestou zjednodušení datového modelu.

V tomto případě je tolerováno porušení datové integrity s tím, že algoritmy, které pracují s daty, musí toto porušení detekovat a zpracovat. K opravě dat v měřeních pak bude docházet postupně, jak se budou používat jednotlivá měření. U často používaných ihned, u jiných v řádu měsíců až let. U některých vůbec, protože se může jednat o měření na zařízeních, která se již přestala používat. Následující tabulka 11.1 popisuje kódování chybových stavů porušení integrity:

	workplan_id	**dev[i].card_id**	**dev[i].range_num**
Nebylo nalezeno pracoviště	0	0	0
Nebyl nalezen řádek měření	-1	0	0
Nebylo nalezeno měřící zařízení	id	-1	0
Nebyl nalezen rozsah zařízení	id	id	-1

Tabulka 11.1: Chybové stavy při zpracování měření

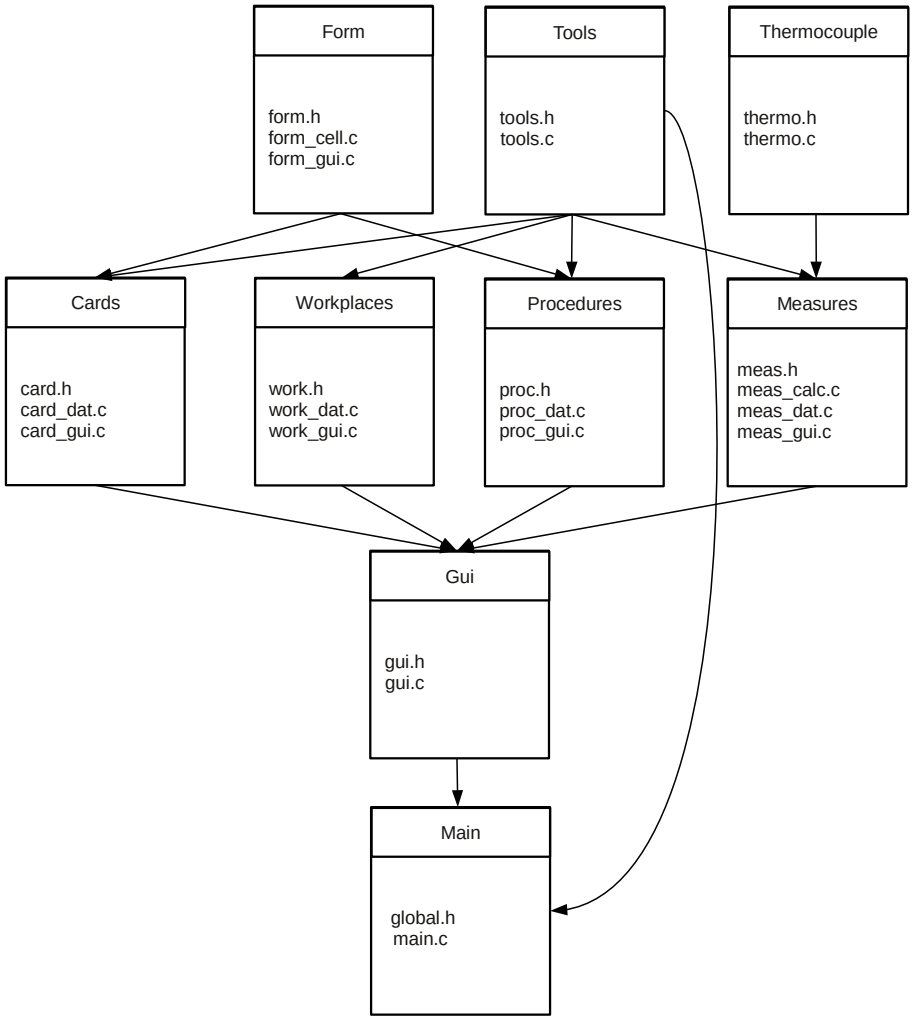
Zatím jsem se rozhodl zachovat integritu u vztahu mezi měřeními a pracovišti. Uživatel by nemůže odstranit pracoviště, které je používáno v některém měření. Ale může toto pracoviště modifikovat a stejně tak může modifikovat karty zařízení, které vlastně tvoří pracoviště. Dle budoucích zkušeností je možné vztah ještě více rozvolnit.

Na základě rozvolnění integrity jsem mohl ušetřit další tabulku, která by svazovala jednotlivé body postupu s rozsahy měřících zařízení a která by poměrně komplikovala dotaz pro získání všech dat měření. Místo toho jsem konfiguraci celého pracoviště pro jeden řádek postupu uložil do jednoho řádku měření jako textový řetězec, který lze velmi jednoduše rozparsovat:

```
INSERT INTO meconf VALUES
(2, 1, 3, 1, 1, 0, '^^^^4,1^^^^', '');
```

11.3 Design

Systém je členěn na jednotlivé moduly. Moduly, které spolu logicky souvisí, mají společný hlavičkový soubor, který umožňuje sdílet informace uvnitř skupiny a zároveň tvoří rozhraní skupiny pro zbytek aplikace. Provázanost jednotlivých modulů ukazuje tabulka 11.2:



Obrázek 11.2: Schéma designu aplikace

12 Implementační detaily

12.1 Konfigurovatelné formuláře

V aplikaci jsou dva druhy konfigurovatelných formulářů: Formulář karet a formulář postupů. V obou druzích formuláře se vyskytuje více sloupců v tabulkách, které mají stejný nebo podobný význam. Na základě tohoto zjištění jsem tyto dva moduly rozdělil do tří částí. Dvě části jsou specifické pro každý druh formuláře. Obsahují funkce pro načítání a ukládání daného typu formuláře. Do poslední části byly vyčleněny funkce, které mají úzký vztah k samotné práci s formulářem:

- Karty měření
  - card_gui.c – grafické nabídky pro otevírání a ukládání karet
  - card_dat.c – datová vrstva pro otevírání a ukládání karet
- Postupy
  - proc_gui.c – grafické nabídky pro otevírání a ukládání postupů
  - proc_dat.c – datová vrstva pro otevírání a ukládání postupů
- Obsluha formuláře
  - form_gui.c – funkce pro práci s formulářem
  - form_cell.c – funkce pro práci s jednotlivými částmi (buňkami) tabulek

12.1.1 Konfigurace tabulek

Funkce pro obsluhu formuláře jsou tedy stejné pro oba typy. Protože se však tabulky pro oba druhy měření liší, je třeba chování formulářů modifikovat. Konfigurace chování jednotlivých sloupců tabulky je uložena pro každý druh zvlášť v datové struktuře **`INDICATOR`**, která popisuje složení tabulky pro různé druhy nastavených indikátorů a v datové struktuře **`COLSET`**, která popisuje, jak mají vypadat jednotlivé sestavy sloupců pro daný druh formuláře. Zkrácený příklad konfigurace ukazují následující výpisy.

Takto lze v případě potřeby jednoduchým způsobem v programu modifikovat chování formulářů, přidávat do formulářů nové sloupce nebo přidat zcela nový druh formuláře bez nutnosti změn v existujícím kódu. Přidá se pouze kód obsluhující nové možnosti a příslušná konfigurace.

```
typedef struct{
     T_SET set;          kód setu
     char *title1;       hlavička 1. varianty
     char *title2;       hlavička 2. varianty
     int variant;        přepínač variant
}COLSET;
COLSET colsets_proc[] =
{
     {C_INOUT, "Vstup", "Hodnota výstupu", 1},
     {C_NOMVAL, "Hodnota rozsahu", "", 0},
     {C_DIVIS, "Digity/Dílky/mm", "", 0},
};
```

```
typedef struct{
     int idx;            index v poli
     int id;             ID indikátoru
     char *text;         název indikátoru
     T_SET colconf;      konfigurace sloupcových sestav
     T_FUNC funconf;     konfigurace funkcí v bloku
}INDICATOR;
INDICATOR indicators_proc[] =
{
     {0, I_NONE,  "Ukazatel", C_NONE, F_NONE},
     {1, I_DIGIT, "displej", C_NOMVAL | C_REALVAL | C_DIVIS, F_NOMVAL},
     {2, I_SOURC, "zdroj veličiny", C_NOMVAL | C_REALVAL, F_NOMVAL},
     {3, I_SOSOL, "zdroj veličiny pevný", C_NOMVAL | C_REALVAL, F_NOMVAL},
     {4, I_ARRAY_STOP, "Chybný indikátor", C_NONE, F_NONE}
};
```

Datová struktura která popisuje formulář, obsahuje ukazatele ukazující přímo na tyto konfigurační datové struktury. Ukazatelé jsou nastaveny při otevírání formuláře podle jeho druhu. Funkce pracující s formulářem pak pomocí ukazatelů přistupují do konfiguračních struktur a podle uložené konfigurace modifikují svou činnost.

12.1.2 Aktivní buňka tabulek

Jak jsem popsal v části věnující se hodnocení prototypů formulářového rozhraní, konečný koncept jsem postavil na formuláři, který má buňky tabulky přecházející mezi aktivním a pasivním režimem. Uchovávání buněk v pasivním stavu šetří systémové prostředky počítače, protože pasivní buňka je méně náročná než aktivní.

Při vlastní implementaci formulářů jsem byl nepříjemně překvapen chováním grafické knihovny (z hlediska knihovny je chování logické), které za určitých podmínek způsobovalo zasekávání buněk v aktivním či pasivním stavu a znemožnilo další práci s formulářem.

Jako základní požadavek na formuláře jsem nastolil možnost ovládat formulář nejen myší ale i klávesnicí. A právě při přechodu mezi oběma druhy ovládání vznikaly uvedené problémy. V případě, že ve vstupní buňce byl zachycen chybný obsah, pokusil se program označit aktivní buňku, ale chybně.

Pro přechod mezi stavy jsou v programu použity dvě funkce:

`Entry_replace()` - Buňku identifikovanou ukazatelem na řádek a číslem sloupce změní z buttonu na entry

`Button_replace()` - Buňku identifikovanou ukazatelem na řádek a číslem sloupce změní z entry na button

Pro oba druhy způsobu ovládání jsem vytvořil dva různé scénáře. Oba scénáře popisují situaci, která nastane při přechodu z jedné buňky (Ba) do jiné buňky (Bb). Jednotlivé zde popisované akce grafické knihovny probíhají ve vlákně grafické knihovny a jsou řazeny sekvenčně:

Scénář myš

Po kliknutí na buňku Bb dostane nejprve signál buňka Bb (příchod fokusu) a ta se stane aktivní. Teprve pak dostane signál buňka Ba (ztráta fokusu), která začne kontrolovat text a zjistí, že je vadný. Problém je v tom, že mezi tím se stala aktivní jiná buňka.

Řešení:

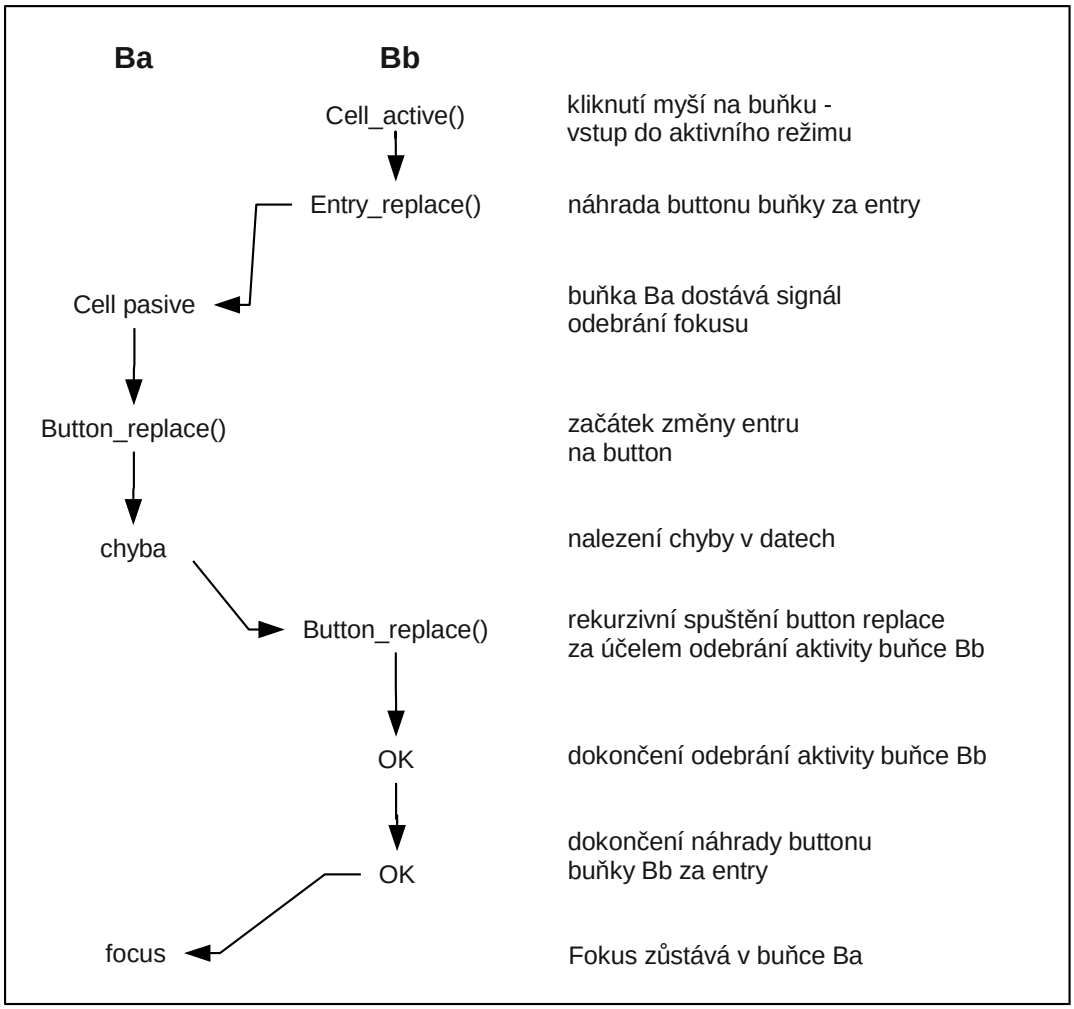
Když Ba kontroluje text, je již aktivní buňka Bb. Proto jsou použity globální proměnné **`growx`** a **`colnumx`**, které slouží jako jednoznačné identifikátory libovolné buňky. Tyto proměnné nastaví Bb, když se stává aktivní. Ba tyto proměnné přečte a zjistí, že je již aktivní jiná buňka.

Kontrola textu probíhá ve funkci, **`Button_replace()`**, která se spouští při zneaktivnění Ba. V této funkci jsou přečteny identifikátory a pokud funkce zjistí, že je již jiná buňka aktivní, spustí sama sebe s těmito identifikátory (buňky Bb), aby zneaktivnila Bb.

Pokud by Bb obsahovala také nevhodný obsah (třeba načtením z databáze), spustil by se tento algoritmus ještě i pro Bb a došlo by k zacyklení. Proto existuje ještě příznak **`ignore`**, který nastaví funkce **`Button_replace()`**, když spouští sebe sama a tento příznak způsobí, že funkce přeskočí u buňky Bb kontrolu chyb.

Poté, co funkce **`Button_replace()`** zneaktivní Bb, nastaví v Ba červené zvýraznění a přesune do Ba fokus.

Sekvenční diagram tohoto scénáře je na následujícím obr. 12.1.



Obrázek 12.1: Diagram zpracování chybného obsahu v buňce při práci s myší

Scénář tabulátor

Tento scénář je poměrně jednodušší. Po stisknutí tabelátoru dostane nejprve signál buňka Ba (ztráta fokusu), která začne kontrolovat text a zjistí, že je vadný. Potom dostane signál buňka Bb, která se stane aktivní.

Řešení:

Když **`Button_replace()`** při kontrole Ba zjistí chybu, nastaví proměnnou **`blockentry`** na 1 a tím způsobí, že funkce **`Cell_button2entry()`**, která má provést následně zaktivnění Bb, tento flag přečte a ukončí se bez akce (aniž by nastavila identifikátory na hodnoty Bb).

`Button_replace()` po nastavení tohoto flagu provede zvýraznění Ba a nastaví do ní kurzor.

Přechod mezi scénáři

I když byly tyto algoritmy nastaveny tak, aby pracovaly každý zvlášť a vzájemně se neovlivňovaly, v praxi nastávají další kombinace, když dochází k přechodům mezi oběma způsoby ovládání přímo v jedné buňce.

Např. když se do buňky klikne, zapíše se nevhodný text a pokračuje se v pohybu tabelátorem. Zde je nutné dodržet další podmínky:

Poté co dojde k přenosu fokusu zpět z Bb na Ba, kvůli opravě dat, musí se proměnné **`growx`** a **`colnumx`** nastavit na hodnoty Ba, aby stav byl stejný, jako když se klikne do Ba.

Dále se musí proměnná **`blockentry`** vynulovat, aby neblokovala příjem fokusu při kliknutí myší na jinou buňku než Ba a tím umožnila opuštění buňky Ba pomocí myši.

12.2 Převodník termoelektrického napětí na °C

Měření teploty založené na termoelektrickém jevu je dnes velmi rozšířené. Termočlánky mají navzdory své nízké ceně poměrně dobrou přesnost měření spojenou s širokým měřícím rozsahem. Nízká cena umožňuje použití termočlánků i v extrémních podmínkách, jako např. měření teploty ve vysokých pecích, kdy během krátké doby měření dojde k roztavení termočlánku.

Termočlánek vznikne pevným spojením dvou vodičů z různých materiálů. Druh materiálů je volen tak, aby bylo výsledné napětí co největší při zachování ostatních sledovaných parametrů termočlánku, jako jsou např. linearita či rozsah měření. Při studiu termočlánků jsem čerpal ze zajímavé knihy [MK05], kde je popsán princip fungování termočlánků:

„Termoelektrické články jsou založeny na Seebeckově jevu, tj, na jevu převodu tepelné energie na elektrickou. Seebeckův jev vzniká tím, že v teplejší části vodiče mají nositelé náboje větší energii.“

Pro teoretický výpočet výsledného napětí se používají tzv. Seebeckovy koeficienty, které závisí na vlastnostech použitých materiálů. Základní výpočet napětí v případě, že teplý a studený konec termočlánku mají nízký rozdíl teplot, je jednoduchý. Ovšem v širším rozsahu teplot se uplatňuje nelinearita a pro výpočty je nutné použít polynomy vyšších řádů.

Protože výpočet je komplikovaný (polynomy mohou být až 13. řádu), jsou tyto výpočty tabelovány v normách. Tyto tabulky se využívají pro převod teploty na termoelektrické napětí.

Ve světě je standardizováno několik druhů termočlánků, které se liší svými parametry a každý druh je vhodný pro jiné aplikace. Pro tyto standardizované typy byla zavedena písmenná identifikační značka a jejich parametry jsou normalizované. U nás se termočlánky zabývá norma ČSN 60584-1. Tuto normu jsem neměl k dispozici a tak jsem jako další zdroj použil webové stránky NIST (National Institute of Standards and Technology) [NIST99].

12.2.1 Výpočet termoelektrického napětí z hodnoty ve °C

Nejrozšířenějším typem článku je typ K. Tento termočlánek se dodává i jako příslušenství k různým digitálním měřícím přístrojům, proto jsem si jej zvolil pro realizaci softwarového převodníku. Tento typ termočlánku má pro výpočet termoelektrického napětí nad nulou netypický polynom:

$$E = a_0 + \sum_{i=1}^{n} a_i (t_{90})^i + c_0 \, e^{[c_1 (t_{90} - 126{,}9686)^2]} \qquad (12.1)$$

Kde
E je hodnota termoelektrického napětí v mV
a_0, c_0, a_i a c_i jsou koeficienty udané v normě.

Při hodnotách nižších než 0 má vztah pro výpočet stejný jako ostatní druhy termočlánků:

$$E = \sum_{i=1}^{n} a_i t^i \qquad (12.2)$$

Jak jsem uvedl výše, tyto hodnoty jsou vypočteny v normě a jsou tabelovány v kroku po 1 °C. Tabulku z normy jsem tedy přenesl do programu. Využil jsem toho, že hodnoty jsou uváděny v kroku 1 °C a sestrojil jsem vztah pro přímý přepočet hodnoty ve °C na index do pole, kde jsou hodnoty uloženy. V případech, kdy není hodnota zaokrouhlená na celé stupně, se vezmou dvě nejbližší celé hodnoty a mezi nimi se provede lineární interpolace.

12.2.2 Výpočet hodnoty ve °C z termoelektrického napětí

Pro výpočet hodnot v pracovišti je kromě přepočtu termoelektrického napětí z hodnoty ve °C potřeba i přepočet obrácený to znamená výpočet hodnoty ve °C z termoelektrického napětí. Pro tento zpětný převod nejsou k dispozici tabulky a musí se použít inverzní polynom ve tvaru:

$$t = a_0 + \sum_{i=1}^{n} b_i E^i \qquad (12.3)$$

Kde a_0 a b_i je koeficienty uvedené v normě. Hodnoty b_i pro článek typu K jsou v následující tabulce pro tři různé teplotní rozsahy:

	-200 – 0 °C	0 – 500 °C	500 – 1372 °C
b0	0.0000000E+00	0.000000E+00	-1.318058E+02
b1	2.5173462E+01	2.508355E+01	4.830222E+01
b2	-1.1662878E+00	7.860106E-02	-1.646031E+00
b3	-1.0833638E+00	-2.503131E-01	5.464731E-02
b4	-8.9773540E-01	8.315270E-02	-9.650715E-04
b5	-3.7342377E-01	-1.228034E-02	8.802193E-06
b6	-8.6632643E-02	9.804036E-04	-3.110810E-08
b7	-1.0450598E-02	-4.413030E-05	0.000000E+00
b8	-5.1920577E-04	1.057734E-06	0.000000E+00
b9	0.0000000E+00	-1.052755E-08	0.0000000E+00

Tabulka 12.1: Koeficienty pro výpočet teploty pro článek typu K

Při vlastním výpočtu je třeba nejprve rozhodnout, do kterého rozpětí teplota spadá a poté provést výpočet se správnými koeficienty.

12.2.3 Přesnost převodu

Kontrolu přesnosti převodu ze °C na mV jsem provedl ručním přepočtením hodnoty pomocí tabulky a porovnal s hodnotou vypočtenou programem. Hodnoty se prakticky nelišily.

Přesnost výpočtu z mV na °C jsem provedl dosazením známé hodnoty vypočtené z tabulky a porovnáním výsledků. Naměřené chyby ukazuje následující tabulka:

Napětí [mV]	Teplota [°C]	Chyba [°C]
-5.8910016	-199.9331802	3.3471063001e-04
-3.554003	-100.0037832	3.6830852180e-05
-3.9e-06	-9.817651954e-05	1.8573488544e-02
4.0959959	99.96318648	3.6727040346e-04
8.1379961	199.9719662	1.3968859526e-04
12.2089958	300.0241805	8.0928508510e-05
16.3969958	400.0016117	4.2791199053e-06
20.6439958	499.9803914	3.9018770887e-05
24.9049958	599.9937294	1.0284479190e-05
29.1289958	699.9808953	2.7150306155e-05
33.2749959	800.0074284	9.4103710981e-06
37.325996	900.0126356	1.4150474547e-05
41.2759961	999.9870795	1.2820679731e-05
45.1189962	1100.002491	2.3554137791e-06
48.8379964	1200.021565	1.8053936321e-05
52.4099965	1299.959257	3.1264986326e-05

Tabulka 12.2: Vypočtené hodnoty převodu z mV na °C

Z tabulky je patrné, že největší chyba se nachází v oblasti nuly. Proto jsem pro testování použil hodnoty, které se přibližovaly velmi těsně k nulové hodnotě (1e-4°C). Přesto, však neklesla přesnost pod 1.8e-02, což je dostatečná přesnost pro přepočet pracoviště pro účely konfigurace. Vstupní rozpětí měřidel je totiž 200mV, výjimečně 60mv, a proto má výpočet velikou rezervu. Při hodnotě rovné nule se přepočet přes vzorec nepoužije.

12.3 Přesnost výpočtu pracoviště

Na základě znalosti zpracovávaných dat je třeba rozhodnout o přesnosti jejich zpracování. Při běžném měření v elektrotechnice se nejmenší číselné hodnoty používají při měření elektrické kapacity a vodivosti. Jednotkou elektrické kapacity je jeden F, což je pro praktické použití velmi velká jednotka, a tak se používají jednotky menší. Nejmenší běžně vyráběná velikost kondenzátorů se uvádí v pikofaradech [pF]. Jeden pikofarad má rozměr 1e-12F. Pro měření elektrické vodivost se používá jednotka siemens [S]. Odpor 1TΩ má elektrickou vodivost 1e-12S.

Jako nejvyšší dosažitelnou přesnost jsem stanovil požadavek měření 0,1pF s přesností na 1%. Z toho vyplývá požadavek na uchování čísel v plovoucí čárce s rozlišením na 1e-15.

Toto rozlišení odpovídá hodnotě ε, která je stanovená normou *IEEE Standard for Binary-Point Arithmetic* (ANSI/IEEE 754-1985). Tato norma se zabývá standardy pro reprezentaci reálných čísel v plovoucí řádové čárce v počítači a matematickými operacemi s těmito čísly. Hodnoty je možné najít též v [HS96]. Pro typ float je hodnota stanovena na $\varepsilon = 1{,}19209290e\text{-}07$, což je pro potřeby tohoto programu přesnost nedostatečná. Vyšší rozlišení poskytuje datový typ double, který má hodnotu $\varepsilon = 2{,}2204460492503131e\text{-}16$. Tato přesnost je dostačující.

Vzhledem k možné kumulaci zaokrouhlovacích chyb při posloupnosti matematických operací, zvýším hodnotu ε, která je interně používána v tomto programu jako tolerance porovnávacích operací na trojnásobek hodnoty z normy. Nutno podotknout, že takto nastavená přesnost je více méně extrémní a v běžné praxi se téměř nevyskytuje. Je však nutné mít určitou rezervu.

Další zvýšení přesnosti by šlo dosáhnout přechodem na typ long double, který má v Linuxu hodnotu $\varepsilon = 1{,}08420217248550443400745280087e\text{-}19$. Je ovšem třeba vzít v úvahu, že tato přesnost by zřejmě zůstala z 99,9% nevyužita a cenou by bylo zvýšení paměťových nároků reálných čísel z 64bitů na 128bitů a zároveň zpomalení některých matematických operací.

13 Ověření funkčnosti

Ve skriptu *confplace.sql* je uložena testovací sada dat, která jsou použitelná i jako demo data. Pomocí skriptu jsou vygenerována data do databáze programu. Pomocí těchto dat se může uživatel seznámit z funkcí programu a provádět různé experimenty. Pokud dojde při experimentování k odstranění těchto dat, je možné databázi kdykoli vrátit do původního stavu příkazem **`sqlite3 confplace.db < confplace.sql`**.

Uložená testovací data neodpovídají zcela běžným měřením, spíše jsou zaměřena na otestování hraničních hodnot.

Skript obsahuje hlavně:

- Tabulku všech fyzikálních jednotek, s kterými lze v programu pracovat.
- Tabulku s několika zařízeními, která je možné v programu používat jako měřící zařízení stejně jako zařízené měřené.
- Tabulku s kartami několika různých měřidel
  - D4845 – prototyp digitálního multimetru
  - M-140 – prototyp přesného zdroje
  - Cívka x100 – prototyp převodníku – zesilovače proudu
  - Bočník 10A/60mV – prototyp převodníku mezi nestejnými veličinami
  - Převod K – softwarový převodník pro termočlánky
- Tabulku s nakonfigurovanými pracovišti
  - Přímé měření
  - Pracoviště s cívkou
  - Pracoviště s bočníkem
  - Pracoviště na měření elektronických teploměrů s termočlánkem K
- Tabulku s několika postupy měření
  - Měření analogového měřidla PU500
  - Měření analogového zdroje BS544
  - Měření digitálního teploměru DT500
- Tabulky s několika měřeními

Měření jsou koncipována tak, aby pokryla různé druhy testů. Řádky měření, které testují nějakou funkcionalitu, mají uveden komentář, který uvádí druh testu nebo pokyn, jak test provést. Komentář je možné zobrazit příslušným tlačítkem.

Testování probíhá ve dvou fázích. V první fázi se kontroluje korektní načtení měření. Porovnává se, zda údaje v načteném formuláři odpovídají hodnotám uvedeným v komentáři.

V druhé fázi se v kombo boxu výběru pracovišť nejprve nastaví položka „Výběr pracoviště“ a následně se znovu nastaví položka, která před tím byla na kombu nastavená. Tím se program přinutí, aby znovu provedl výpočet pracoviště. Kontroluje se, zda program správně nalezl rozsahy přístrojů, které jsou uvedeny v komentáři.

14 Závěr

Záměrem této práce bylo zrychlit a zefektivnit činnosti při vytváření konfigurace měření a zároveň umožnit doplnění této funkcionality do programu, který byl vyvinut v rámci mé bakalářské práce.

Požadované cíle se podařilo naplnit vývojem aplikace, která slouží pro interaktivní konfiguraci měření. Při vývoji této aplikace byly použity takové prostředky, které umožní jednoduché začlenění grafického rozhraní vyvinuté aplikace do aplikace původní. Oddělení datové a grafické vrstvy umožňuje jednoduchou výměnu dat a interoperabilitu s dalšími částmi aplikace.

Kvantifikovat úsporu času při použití vytvořené aplikace je dosti náročné, protože příprava pracoviště je poměrně tvůrčí záležitost a závisí na mnoha okolnostech. Nelze použít jednoduchý test, kdy jeden pracovník vytvoří konfiguraci jedním způsobem a následně druhým způsobem a porovnají se naměřené časy provedení. Problém je v tom, že při prvním provedení pracovník nejprve sbírá zkušenosti a teprve poznává dané zařízení. Při druhém opakování však již přirozeně použije zkušenosti z předchozí práce.

Proto jsme v laboratoři, se kterou jsem spolupracoval, navrhli test, kdy jednu přípravu dělají nezávisle dva pracovníci, a to každý jiným způsobem. Zároveň jsem pracovníky sledoval při práci a snažil se zhodnotit rozdíl v ergonomii práce. Jsem si vědom, že při tomto způsobu testování hrají určitou roli různé pracovní zkušenosti a momentální stav invence obou pracovníků. Proto jsme testů provedli několik a pracovníky střídali. Důležité bylo, že se vždy jednalo o nový druh měření, který dosud tato laboratoř neprováděla. Z tohoto důvodu testování trvalo poměrně dlouhou dobu, protože jsme museli čekat, až se objeví příslušná zakázka.

Z celkové doby práce na konfiguraci pracoviště byla vyloučena doba studia manuálů k zařízení. Z naměřených údajů jsem vypočetl průměrné hodnoty a zjistil jsem, že průměrná doba vytváření konfigurace starým způsobem byla 43 minut. Novým způsobem bylo možné vytvořit stejné konfigurace za 34 minut. Což představuje úsporu 21% původního času. Zároveň je zde ještě prostor ke zrychlení, protože jsem u uživatelů registroval při práci s novým rozhraním jisté zaváhání při některých operacích. Lze předpokládat, že až se uživatelé plně ztotožní s novým rozhraním, bude práce ještě efektivnější.

Při testování aplikace jsme objevili další možnosti, které by zefektivnily práci s programem. Za stěžejní funkcionalitu považuji dát uživateli při konfiguraci měření možnost k vizualizaci pracoviště, kde by byla přehledně zobrazena měřená veličina v celém pracovišti. Někdy totiž není možné provést správnou konfiguraci pracoviště, protože uživatel udělá chybu v některém z předchozích stupňů přípravy (karty zařízení, postupy měření ...). Vizualizace pracoviště by umožnila rychle identifikovat místa nekompatibility mezi jednotlivými částmi vstupních dat.

Použité zkratky

ČMS	- Česká metrologická společnost
GTK+	- Gimp ToolKit - Grafická knihovna
GUI	- Graphic User Interface - Grafické uživatelské rozhraní
SQL	- Structured Query Language - Strukturovaný dotazovací jazyk používaný v databázích
NIST	- National Institute of Standards and Technology - agentura U.S. Commerce Departmen
ČSN	- Česká státní norma

Literatura

[JV03] Prof. Ing. Jindřich Vítovec, DrSc: *Měření základních elektrických veličin*. Vyd 2003 Praha: ČMS, 2003. 181 s.

[JV08] Prof. Ing. Jindřich Vítovec, DrSc: *Elektrická měření I – elektrické měřící přístroje*. Vyd 12/2008 Praha: ČMS, 2008. 22 s.

[TČG00] O. Tůmová, V. Čtvrtník, J. Girgr: *Elektrická měření – měřící metody*. 1. vyd. Plzeň: Západočeská univerzita v Plzni, 2000. 178 s. ISBN 80-7082-607-X

[GRS02] E. Gescheidtová, J. Rez, M Steinbauer: *Měření v elektrotechnice*. 1. vyd Brno: Vysoké učení technické v Brně, Nakladatelství VUTIUM, 2002. 182 s. ISBN 80-214-1990-3

[SQL14] *SQLite: Categorical Index Of SQLite Documents*. HWACI. [online]. [cit. 2014-01-02]. Dostupné z: http://sqlite.org/docs.html

[GTK14] *The GTK+ project: API Documentation*. THE GTK+ TEAM. [online]. [cit. 2013-11-02]. Dostupné z: http://www.gtk.org/documentation.php

[AK07] Andrew Krause: *Foundations of GTK+ Development*, Apress 2007 230s. ISBN 1-59059-793-1

[TM06] Tim-Philipp Müller: *GTK+ 2.0 Tree View Tutorial*, pouze v pdf, 26.9.2006, 88s

[MK05] Marcel Kreidl: *Měření teploty. Senzory a měřící obvody*. 1. vyd Praha: BEN - technická literatura, 2005. 230 s. ISBN 80-7300-145-4

[NIST99] *NIST ITS-90 Thermocouple Database: NIST Standard Reference Database 60*. NIST. [online]. Version 2.0. NIST, 21.12.1999, 25.7.2013 [cit. 2014-04-07]. Dostupné z: http://srdata.nist.gov/its90/main/

[HS96] Samuel P. Harbison, Guy l. Steele Jr.: *Referenční příručka jazyka C*, 1. vydání Veletiny: SCIENCE, 1996, 336 s. ISBN 80-901475-50

Přílohy

Instalační příručka

Confplace je program určený pro přípravu měřících pracovišť na linuxových operačních systémech. Umožňuje jednoduchým a tvořivým způsobem organizovat data a sdílet je mezi jednotlivými stupni přípravy měření. Může být využit v elektrotechnice i při měření mechanických veličin. Poté, co je uživatelem zadáno potřebné množství dat, je program schopen provádět samostatně výběr vhodných nastavení měřících rozsahů použitých zařízení.

Instalace programu

Pro chod programu je nutné mít v operačním systému nainstalovány některé nezbytné externí součásti. Jedná se především o tyto komponenty:

Název	Popis
GTK+ 3.10	grafická knihovna a její závislosti
sqlite3	databázová knihovna a klient
libwebkitgtk 3.0	knihovna pro zobrazování HTML stránek

Tabulka 1: Závislosti programu

Konkrétní verze komponent jsou uvedeny v souboru **`readme.txt`**, který je součástí distribuce programu. Další potřebné komponenty, jako knihovna *glibc* a podobně, jsou běžnou součástí linuxových operačních systémů.

Program neobsahuje zvláštní instalátor, a tak se instalace programu provádí pouhým nakopírováním složky s programem na požadované místo. Je třeba dát pozor na to, aby instalace programu byla určena pro použitou architekturu a obsahovala všechny potřebné části.

Ve složce musí být uložen binární soubor **`confplace`**, který představuje vlastní spustitelný program. Dále je ve složce uložen soubor **`confplace.db`**. I tento soubor je velmi důležitý a bez jeho přítomnosti nelze program používat. Tento soubor obsahuje databázová data nutná pro běh programu. Pokud se provádí čistá prvotní instalace, obsahuje tento soubor pouze základní údaje. Nejdůležitější z obsahu databáze jsou data fyzikálních jednotek, se kterými je program schopen pracovat. Bez těchto údajů nelze program spustit. Zbytek obsahu databáze tvoří ukázkové příklady, které lze bez obav v prostředí programu následně smazat.

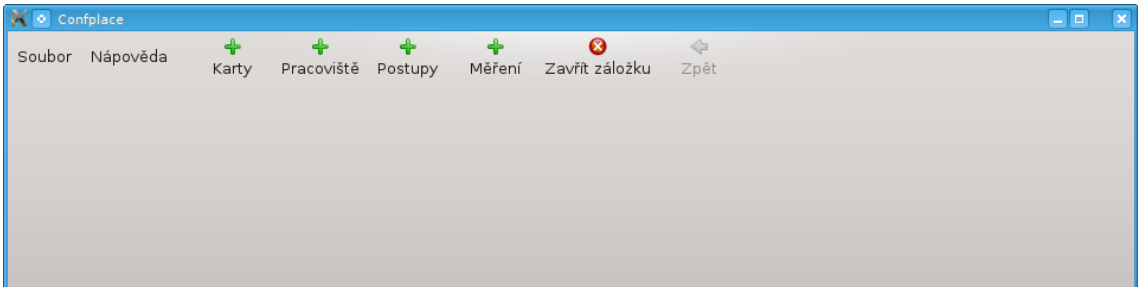
Pokud dojde ke ztrátě tohoto souboru, je možné vytvořit nový pomocí databázového skriptu s názvem **`confplace.sql`**, který je též přiložen. Obnovu souboru je možno provést z příkazové řádky operačního systému spuštěním následujícího příkazu: **`sqlite3 confplace.db < confplace.sql`**. Po spuštění tohoto jednoduchého příkazu se v adresáři vytvoří nový databázový soubor s implicitním obsahem.

Poslední součástí instalační složky je kromě souboru **`readme.txt`**, který byl již zmíněn výše, podadresář **`doc`**, který obsahuje tuto nápovědu.

Uživatelská příručka

Hlavní okno aplikace

Po spuštění programu příkazem **`confplace`** v instalačním adresáři se otevře hlavní okno programu:



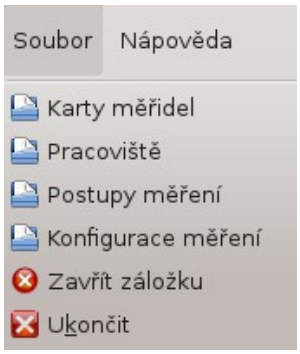
Obrázek 1: Hlavní okno aplikace

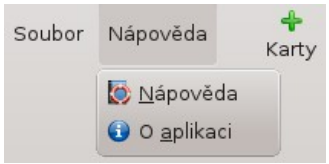
Na obrázku je vidět v horní části panel, který obsahuje ve své levé části menu aplikace a za ním hlavní nástrojovou lištu - toolbar. Zbytek okna tvoří prázdný prostor, který obsahuje sešit, do kterého se vkládají karty se záložkami s obsahem jednotlivých činností, které se v aplikaci dají provádět. Karet je možné otevřít několik od každé činnosti, a i když je jejich maximální počet omezen, je pro práci dostatečný. Protože se pojem "karta" užívá v tomto programu pro označení určitého druhu formuláře, bude v dalším textu používán pro kartu výraz "záložka" nebo "list v sešitu".

Menu

Menu "Soubor" obsahuje v horní části položky pro otevírání záložek s jednotlivými činnostmi. Ve spodní části jsou volby pro zavření viditelné záložky a položka pro uzavření celé aplikace.

Menu "Nápověda" obsahuje na prvním místě položku pro otevření prohlížeče tohoto manuálu a na druhé pozici volbu pro vyvolání okénka s informacemi o verzi programu:





Obrázek 2: Menu aplikace

Nástrojová lišta

Nástrojová lišta obsahuje ikony, které slouží k jednoduchému ovládání programu. Ikony s piktogramem znaménka plus slouží k rychlému otevření listů v sešitu s jednotlivými činnostmi. Další ikona slouží k uzavření listu, který se nachází navrchu. Poslední ikona slouží pro pohyb v rámci jednoho listu a aktivuje se v případě, že list umožňuje provést krok zpět:

Obrázek 3: Nástrojová lišta

Za touto hlavní nástrojovou lištou vpravo je volný prostor, který je využíván jednotlivými činnostmi programu pro umístění jejich lokálních nástrojových lišt.

Příprava měření

Před tím, než se začne s jakýmkoli měřením, je nanejvýš vhodné provést jeho dobrou přípravu. V opačném případe zde hrozí, že naměřené výsledky neodpovídají realitě, ale navíc je možné způsobit závadu na měřeném či měřícím zařízení. V krajním případě je zde i nebezpečí úrazu osob či hmotné škody většího rozsahu. Každý, kdo provádí měření by si měl dobře rozmyslet následující:

- Na jakém zařízení se bude měřit.
- Které parametry / fyzikální veličiny se budou měřit.
- Jakou měřící metodou.
- Jaká měřící či pomocná zařízení budou použita vzhledem k použité měřící metodě.
- Jak budou zařízení navzájem propojena vzhledem k použité měřící metodě.
- Jak budou měřící zařízení nastavena.

Několik předchozích bodů se může rozvinout do velkého množství dat, která je zapotřebí spojit, než se začne s měřením. Protože se mnohá měření vzájemně podobají buď proto, že mají měřená zařízení podobné vlastnosti, nebo jsou používána stejná měřící zařízení (nebo jejich část), je nanejvýš výhodné, udělat nashromážděná data znovu použitelná i pro další měření v budoucnu. To je jedním z hlavních úkolů tohoto programu. Druhým úkolem je tato shromážděná data použít i k automatizaci často prováděných činností. Z hlediska tohoto programu je transformováno šest předchozích bodů do čtyřech konkrétních aktivit. Zde jsou seřazeny tak, jak na sebe vzájemně navazují:

- Příprava měřících zařízení. Uvědomění si parametrů zařízení (studium manuálů), specifikace měřících schopností a limitů dosažitelného měřícího zařízení.
- Propojení více měřících zařízení do jednotlivých pracovišť (zapojení) specializovaných na určité druhy měření.
- Rozpracování měřeného objektu. Uvědomění si, jaké fyzikální veličiny a s jakými očekávanými hodnotami je nutné na zařízení měřit, aby byla splněna očekávání kladená na měření.

- Propojení výše zmíněných tří bodů do jednoho celku tak, aby měřící zařízení propojená do měřících pracovišť, pokryla svými schopnostmi všechny naplánované měřící body.

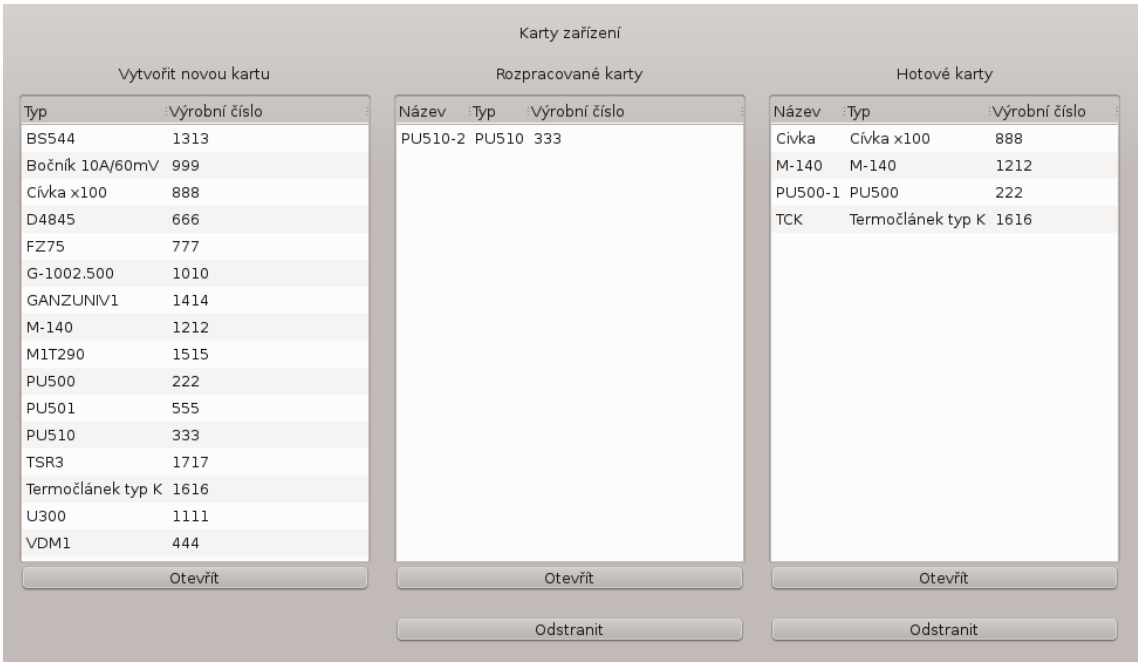
Výše uvedené body jsou zapracovány do programu. Pro každou činnost, která odpovídá jednomu bodu, existuje samostatné prostředí.

Karty zařízení

Úvodní nabídka karet

Karty zařízení slouží k popisu měřících schopností zařízení. Karta může být vytvořena pouze pro konkrétní zařízení, které je identifikováno pomocí svého továrního typu a výrobního čísla. Je to z toho důvodu, že i dvě na první pohled zcela stejná zařízení, mohou mít různé parametry, například díky stárnutí. I zařízení, které některými svými parametry (pracovním rozsahem, přesností) zcela neodpovídá technické dokumentaci výrobce, může být výhodně používáno na měření s nižšími požadavky na schopnosti zařízení. Stačí jen zjištěné slabší parametry opravit v kartě zařízení a zařízení pak lze bez problémů v programu dále používat k činnostem, na které tyto parametry stačí.

Po kliknutí na ikonu "Karty" nebo po použití volby "Soubor->Karty měřidel", se otevře nabídka vytvoření a editace karet:



Obrázek 4: Úvodní nabídka karet

Úvodní obrazovka karet obsahuje celkem tři nabídky:

První nabídka vlevo umožňuje vytvořit novou - prázdnou kartu. Nejprve je nutné vybrat konkrétní zařízení (dle typu a výrobního čísla), pro které bude karta vytvářena. Stiskem tlačítka "Otevřít", pod nabídkou, se otevře pracovní formulář.

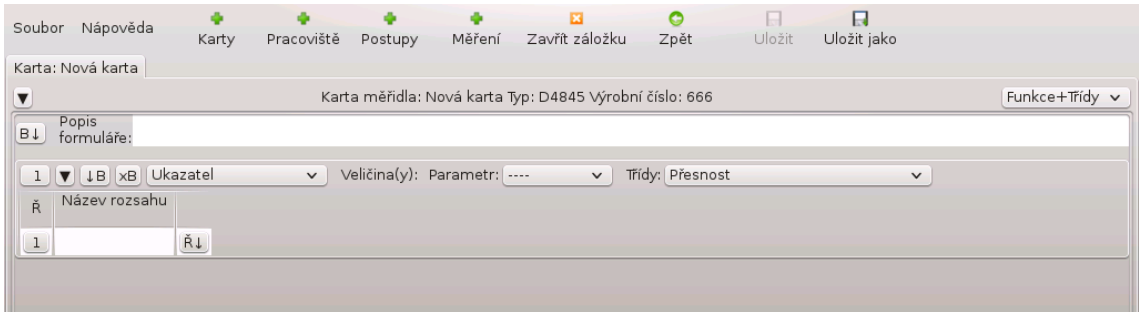
Prostřední nabídka obsahuje seznam rozpracovaných karet. Ze zkušeností je zřejmé, že ne vždy se podaří sestavit celou kartu naráz. Někdy je nutné práci přerušit z časových důvodů, jindy pro nedostatek informací. Pro tyto případy jsou zde zobrazeny rozpracované karty zvlášť, kvůli jejich lepšímu následnému vyhledávání.

Poslední nabídka slouží k otevírání hotových karet za účelem jejich editace. Tato nabídka, stejně jako nabídka předchozí, též umožňuje i trvalé odstranění karty z databáze. Po stisku tlačítka "Odstranit" je ještě uživatel dotázán, zda si skutečně přeje smazat danou kartu v databázi. Pokud odpoví kladně, je karta nevratně odstraněna.

Po výběru jedné z možností se otevře nová záložka v sešitu s formulářem karty, který slouží ke vkládání hodnot a jejich editaci. Pokud použije uživatel k otevření formuláře nabídku vlevo, zobrazí se nová karta obsahující pouze minimální základ formuláře tak, jak je vidět na dalším obrázku:

Formulář karty

Spolu s formulářem se na horní liště objeví i panel tohoto formuláře, který obsahuje dvě ikony pro uložení formuláře.



Obrázek 5: Formulář karty

Přesto, že minimální formulář neobsahuje žádná data, je zde vidět množství důležitých ovládacích prvků, které vnášejí do formuláře požadovanou interaktivitu. V následujícím seznamu jsou seřazeny směrem shora dolů:

Statické záhlaví

Statické záhlaví je vždy viditelné a obsahuje ovládací prvky a informace, které by měly být uživateli vždy po ruce.

- Záložka s nápisem "Karta: Nová karta". První slovo "Karta" oznamuje uživateli, jaký druh formuláře je v záložce otevřen a usnadňuje orientaci v programu, kde je otevřeno více záložek. Za dvojtečkou je pak uveden název karty zadaný uživatelem. Tento název zadává uživatel při konečném uložení rozpracované karty a slouží k její identifikaci. Do doby uložení karty pod novým názvem, je karta pojmenována "Nová karta".

- Pod záložkou se nalézá expander označený piktogramem "▼". Tento expander je nadřazen ostatním expandérům, které jsou umístěny dále ve formuláři a které slouží ke skrývání/expandování jednotlivých funkčních bloků formuláře tak, aby se zvýšila jeho přehlednost. Tímto konkrétním expandérem je pak možné skrýt/expandovat všechny bloky najednou.
- Vpravo od expanderu se nalézá text s přesnou identifikací zařízení, které je v kartě popsáno.
- Vpravo od identifikace se nalézá přepínač pohledu zobrazení dat. Protože v některých případech mohou být řádky formuláře velmi dlouhé a nevejdou se celé na monitor, může si uživatel tímto přepínačem některé části informací v řádku skrýt. Řádek je tématicky rozdělen na tři části.
    - Záhlaví řádku - tlačítko s číslem řádku a pole pro jeho název. Záhlaví je vždy viditelné.
    - Údaje o měřících schopnostech, které se týkají fyzikálních veličin a jejich rozpětí. Lze volitelně skrýt.
    - Údaje o přesnosti měření. Lze volitelně skrýt.

Rolovací část formuláře

Tato část je celá uvnitř oblasti ohraničené automatickými posuvníky, které umožňují v případě potřeby rolovat obsah.

- Tlačítko s piktogramem "B↓" slouží ke vložení nového bloku řádků na začátek formuláře.
- Popis formuláře je textové okno, do kterého je možné zapisovat poznámky k tomuto formuláři. Většinou se zde mohou zapisovat informace k zapamatování, proč byly do formuláře zapsané určité hodnoty apod.
- První blok s řádky.

Blok formuláře

Funkcí bloku je sdružovat a hromadně modifikovat jednotlivé řádky karty, které k sobě nějakým způsobem patří. Všechny řádky v jednom bloku mají stejný ukazatel a stejné měřené veličiny (viz dále). Každý blok má své záhlaví, které slouží k interaktivní modifikaci tabulky, která je v bloku umístěná.

- Tlačítko s číslem bloku. Kromě toho, že tlačítko jednoznačně identifikuje blok ve formuláři, má ještě další funkci. Po jeho stisknutí levým tlačítkem myši je možné celý blok přenést na jiné místo ve formuláři. Po najetí myší na tlačítko jiného bloku je možné blok "upustit" na nové místo. Ostatní bloky budou posunuty a všechny bloky budou přečíslovány dle svých nových pozic.
- Vpravo následuje expandér bloku, jehož funkce byla popsána výše. Tlačítko s piktogramem "B↓" slouží ke vložení nového bloku řádků za současný blok.
- Tlačítko s piktogramem "xB" slouží k nenávratnému odstranění bloku z formuláře. Ještě před tímto krokem je uživatel dotázán, zda odstranění skutečně požaduje.

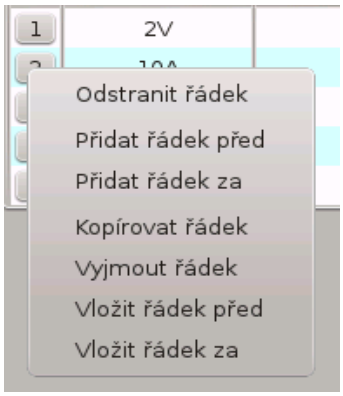
- Roletka s nápisem "Ukazatel" je jednou ze stěžejních voleb. Slovo ukazatel je zástupné pro širší okruh pojmů a rozumí se jím způsob interakce zařízení s obsluhou. Typickým ukazatelem je například displej nebo stupnice měřícího přístroje. Zařízení však může být v pracovišti i zdrojem nebo převodníkem, který "jen" převádí jednu fyzikální veličinu na jinou, aniž by cokoliv ukazoval. Po výběru některého z předvolených ukazatelů se překonfiguruje tabulka tak, že jsou do ní vloženy sloupce, které jsou potřebné pro popis měřících možností zařízení s tímto druhem ukazatele.
- Nápis "Veličina(y):" Označuje oblast, kde se objeví nabídka nebo nabídky pro výběr veličin v závislosti na nastavení roletky "Ukazatel". Nabídka je jedna pro veličinu nastavenou na zařízení v případě, že rozsah zařízení měří veličinu, nebo jsou tři v případě, že zařízení plní funkci převodníku. Nastavuje se pak vstupní, nastavená a výstupní veličina.
- Nabídka "Parametr" slouží k výběru parametrické veličiny. Parametrická veličina zpřesňuje podmínky měření. Např. je-li měřenou veličinou střídavé napětí, pak parametrickou veličinou by měla být frekvence tohoto napětí, protože frekvence měřeného napětí může mít podstatný vliv na přesnost měření, zvláště například u ručkových měřidel. Po výběru parametrické veličiny se do tabulky v bloku přidají sloupce pro zadání rozsahu parametru, pro který platí údaje v daném řádku.
- Nabídka "Přesnost". Zde se provádí výběr způsobu, kterým bude zadávána přesnost (třída přesnosti) zařízení na konkrétním řádku. Stejně jako v předchozích případech, i zde se po výběru zadání vloží do tabulky patřičné sloupce pro zadání hodnot.

Řádek formuláře

Řádek formuláře obsahuje vlastní textová pole pro zadávání číselných hodnot popisujících schopnosti měřicího zařízení. Každý řádek obsahuje minimálně následující prvky:

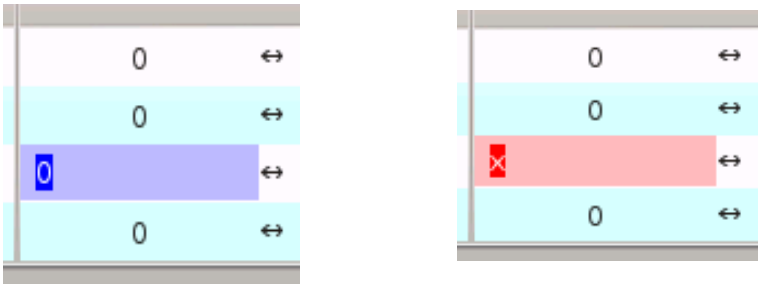
- Tlačítko s číslem řádku, které má celkem tři funkce:
  - Identifikuje řádek v rámci bloku.
  - Pomocí myši je možné za toto tlačítko přenést řádek na jiné místo v bloku.
  - Po kliknutí pravým tlačítkem myši na toto tlačítko se vyvolá kontextová nabídka řádku.
- Textové pole "Název rozsahu". Jak již název napovídá, slouží jako označení rozsahu pro operátora zařízení. Název by měl korespondovat s tím, jaké je označení rozsahu měření na samotném přístroji.
- Tlačítko s piktogramem "Ř↓" slouží ke vložení nového řádku za současný řádek.

Pro rychlejší práci s údaji je možné řádky bloku, mimo jiné, i kopírovat a vkládat pomocí kontextového menu. Všechny možnosti tohoto menu ukazuje následující obrázek:



Obrázek 6: Kontextová nabídka řádku

Jednotlivé buňky tabulky bloku jsou aktivní. To znamená, že po kliknutí do buňky, se tato změní na textový vstup, do kterého je možné zadávat text. Aktivní buňka je také barevně zvýrazněna. Při odejmutí zaměření buňce se buňka opět změní na prostý text. Při přechodu buňky z aktivního stavu do neaktivního se u všech číselných polí provádí kontrola, zda je zadaný text možné převést na desetinné číslo a v případě chyby se buňka zvýrazní výstražnou barvou. Na obrázcích níže je vidět vlevo aktivní buňka a vpravo buňka s chybou:



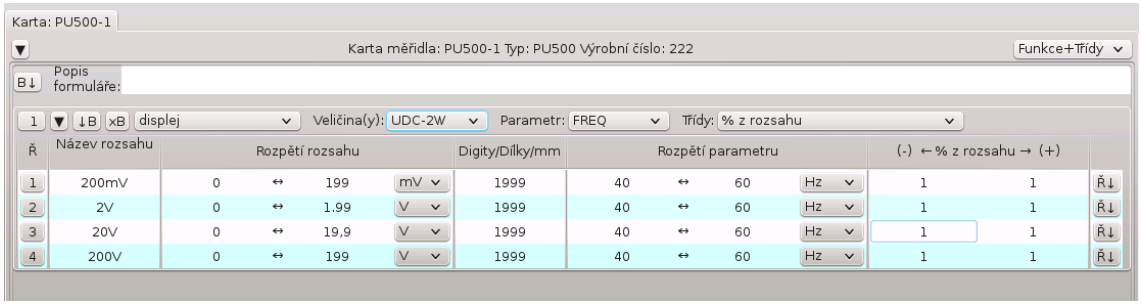
Obrázek 7: Aktivní buňka tabulky

Existují dvě textová pole se speciálním režimem. Kontrola obsahu buňky se neprovádí u textového pole pro zadání názvu řádku. Zde může být zapsáno cokoliv. Další zvláštní buňkou je pole pro zadání převodu převodníku. V této buňce se provádí kontrola, zda text odpovídá validnímu výrazu pro převod hodnoty ze vstupu na výstup převodníku. Pro všechna políčka platí, že se před konečným uložením karty kontroluje, zda políčko nezůstalo prázdné a pokud ano, formulář nelze uložit jako hotový. Práci je však možné kdykoli přerušit, aniž by byl formulář korektně dokončen a změny provedené ve formuláři uložit (formulář uložit jako rozpracovaný) stiskem ikony "Uložit" na horním panelu.

Do pole převodu převodníku lze vkládat pouze takový výraz, který je schopen program následně zpracovat. Formát tohoto výrazu je následující: "IN [+-*/K] VAL". Ve výrazu se zatím mohou vyskytovat následující prvky:

- Vstupní hodnota převodníku zastoupená řetězcem: "IN".
- Hodnota nastavená na převodníku zastoupená řetězcem: "VAL".
- Operátor, který je aplikován na hodnoty IN a VAL.
- Symbol sčítání: "+".
- Symbol odčítání: "-"
- Symbol násobení: "*".
- Symbol dělení: "/"
- Symbol převodu termoelektrického napětí termočlánku typu K na teplotu ve °C: "K"

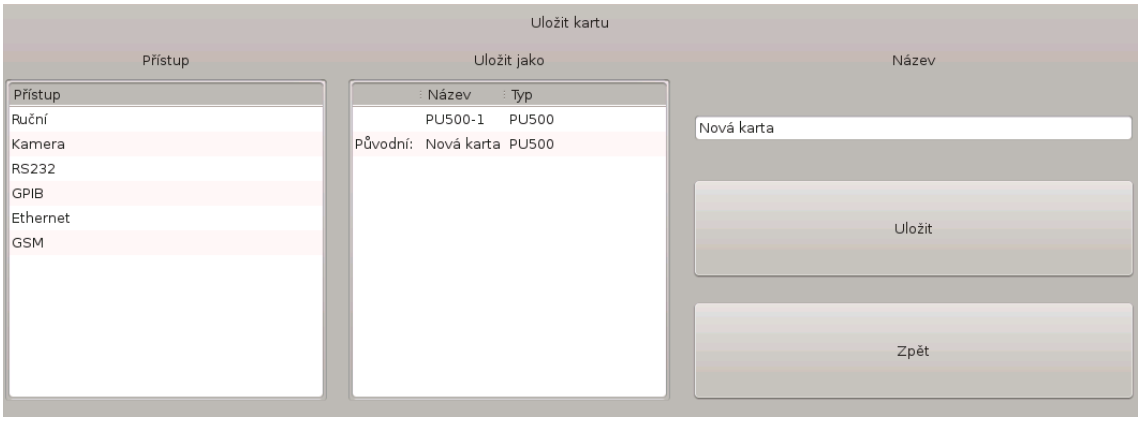
Následující obrázek ukazuje, jak může vypadat nakonfigurovaný formulář:



Obrázek 8: Nakonfigurovaný formulář

Uložení karty

Řádně vyplněnou kartu lze uložit jako hotovou stiskem ikony "Uložit jako" na horním panelu. Po stisku ikony se objeví následující nabídka:



Obrázek 9: Nabídka uložení karty

V nabídce je nutné vybrat v levém poli způsob, jakým se bude k zařízení přistupovat. V prostředním poli je seznam karet stejného měřidla. Jak je vidět, může jich být více a uživatel se může rozhodnout, kterou z nich přepíše právě ukládaným obsahem. V pravém poli je zapotřebí změnit defaultní název "Nová karta" na smysluplné označení ukládané karty. Po vyplnění a stisku tlačítka "Uložit" se ještě program zeptá, zda chce uživatel opravdu přepsat vybranou kartu a v případě, že odpoví kladně, je karta uložena a zobrazí se opět úvodní nabídka karet.

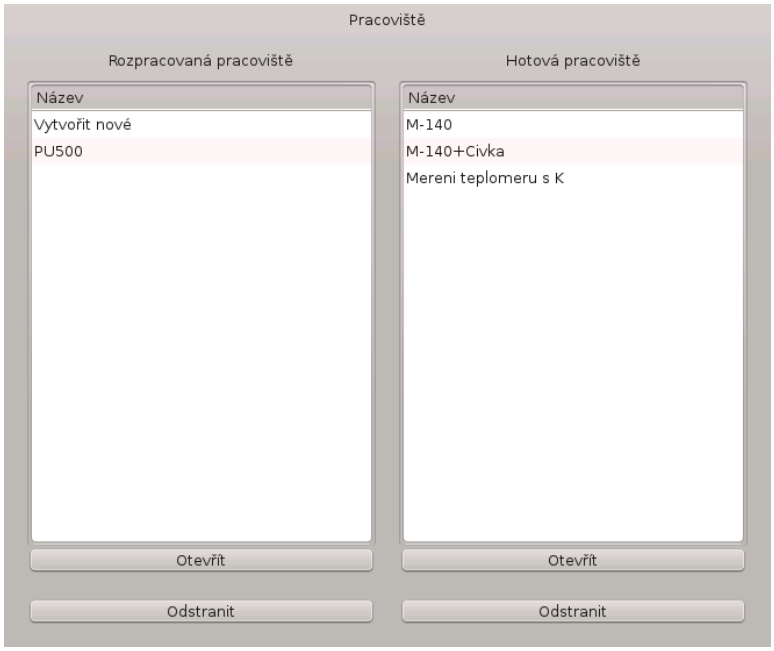
14.1.1 Pracoviště

Pokud si uživatel vytvoří karty pro svoje měřící zařízení, aby je mohl používat, musí je zařadit do nějakého pracoviště, protože pouze celá pracoviště lze propojit s měřenými body. Pojem pracoviště vnáší do organizace měření informaci o tom:

- Jaká zařízení budou při měření použita.
- Jak mezi sebou budou propojena.

Propojením těchto dvou kategorií vznikne přesný popis pracoviště, který lze používat univerzálně na určitou kategorii měřících úloh. Sestavení pracoviště je v grafickém prostředí jednoduchý úkon. Poté, co uživatel vybere volbu "Pracoviště" buď v menu aplikace, nebo na horním panelu, otevře se úvodní nabídka pracovišť:

Úvodní nabídka pracovišť



Obrázek 10: Úvodní nabídka otevření pracoviště

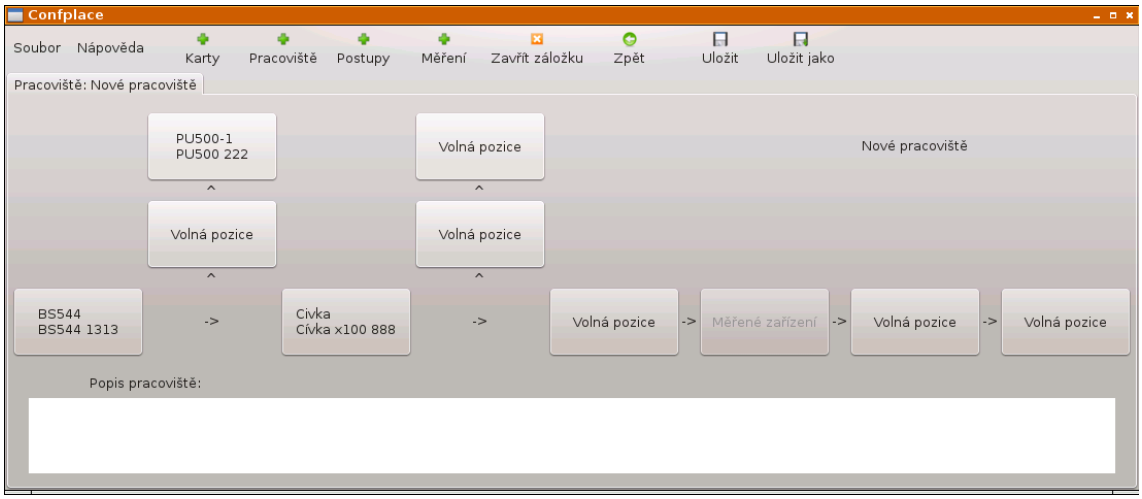
V levém poli lze vybrat rozpracovaná pracoviště k dokončení a v pravém poli lze vybrat hotová pracoviště k editaci.

V obou polích lze pracoviště odstranit po stisknutí tlačítka "Odstranit". Nejprve je ještě uživatel dotázán, zda si skutečně přeje pracoviště odstranit a v případě, že odpoví kladně, dojde k nenávratnému odstranění pracoviště. Odstranit nejde pracoviště, které je používáno v některém měření. V tomto případě se místo odstranění objeví varovné hlášení.

Po výběru některého pracoviště v seznamu stisku tlačítka "Otevřít", se otevře nová obrazovka s plánkem pracoviště.

Plán pracoviště

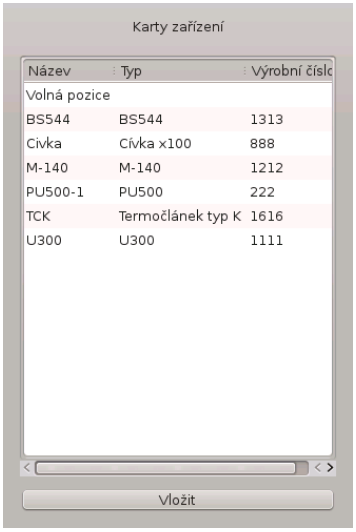
Plán pracoviště představuje souhrn všech pozic, s kterými dovede program pracovat. Na všechny pozice, kromě pozice měřeného zařízení, je možné vložit jakékoli měřící zařízení, které má vytvořenou kartu. Je ovšem třeba zdůraznit, že některé kombinace rozložení nemusí dávat smysl. Předpokládá se, že uživatel programu je odborník a program tedy nijak nezasahuje do jeho potřeb. Jediná věc, která je zapovězená, je dvojnásobné vložení toho samého zařízení do pracoviště.



Obrázek 11: Prostředí pro konfiguraci pracoviště

Zcela nalevo se v pracovišti nachází zdroj veličiny a šipky mezi tlačítky karet pak představují tok veličiny. V každém pracovišti by měl být zdroj veličiny, pokud jím není samo měřené zařízení. Zde na ukázkovém obrázku máme jako zdroj (stejnosměrného proudu) zařízení BS544. Protože jeho výstupní proud není dostatečný pro měření zařízení, je zesílen cívkou. Měřeným zařízeným bude zřejmě nějaký malý klešťový ampérmetr a cívka bude nasazena na jeho měřící kleště. Protože je třeba znát přesnou hodnotu proudu, která je přiváděna do měřeného ampérmetru, je v pracovišti přítomen ještě kontrolní multimetr PU500, kterým se měří proud jdoucí ze zdroje.

Vložení zařízení do pracoviště je jednoduché. Stačí stisknout tlačítko na příslušné pozici a místo plánku pracoviště se objeví nabídka s dostupnými kartami:

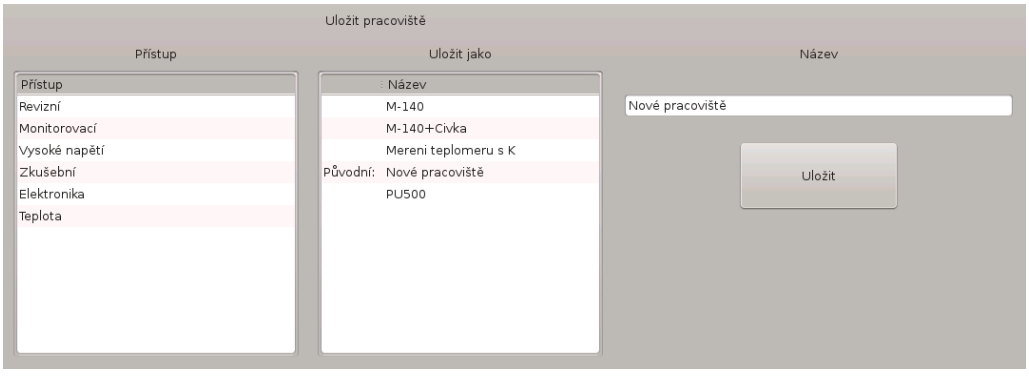


Obrázek 12: Nabídka zařízení

Vložení zařízení se provede stiskem tlačítka "Vložit". Zařízení je možné z pracoviště odstranit tak, že se stiskne příslušné tlačítko zařízení v pracovišti a následně se v nabídce vybere možnost "Volná pozice", která se nachází zcela nahoře a tato se vloží.

Uložení pracoviště

Poté, co je pracoviště nakonfigurováno, je možné jej uložit pomocí nabídky pro uložení pracovišť. V levém poli je nutné vybrat zařazení pracoviště, v prostředním vybrat pracoviště, které bude při uložení přepsáno a v pravém zadat nový název pracoviště. Po stisku tlačítka "Uložit" se ještě program zeptá, zda chce uživatel opravdu přepsat vybrané pracoviště. V případě, že odpoví kladně, je pracoviště uloženo a zobrazí se opět úvodní nabídka.



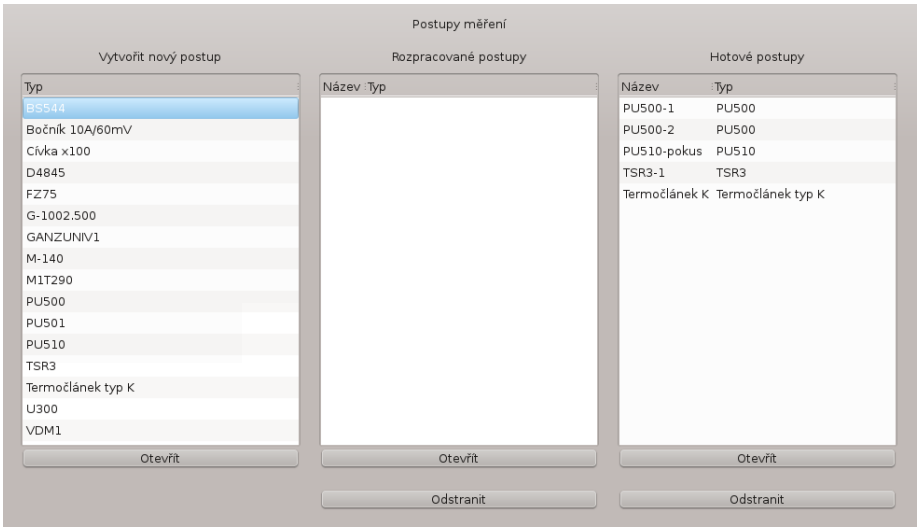
Obrázek 13: Nabídka uložení pracoviště

Postupy měření

Postup měření je předpisem, podle kterého se provádí jednotlivé měřící operace. Celé měření se skládá z jednotlivých odměrů, které se sdružují do skupin podle měřené veličiny. O každém jednotlivém odměru je třeba znát mnoho informací. Pro organizaci postupů měření je používán podobný interaktivní formulář, jako v případě karet zařízení, ale s určitými rozdíly.

Úvodní nabídka postupů měření

Po otevření záložky postupů se zobrazí úvodní nabídka všech postupů:



Obrázek 14: Úvodní nabídka postupů měření

Na první pohled tato nabídka připomíná nabídku výběru karet. Zatímco u karet ale platí, že každá karta musí být sdružena s konkrétním kusem zařízení, u postupů tomu tak není. Měřící postup je sdružen s konkrétním typem zařízení. Je to logické. Máme-li zkušebnu, která provádí měření, vlastní určitý omezený počet měřících zařízení, ale měřených objektů může být velmi mnoho. Proto je důležité rozdělit měřená zařízení do určitých typů - celků, kde každý kus tohoto typu se kontroluje stejným způsobem.

Nejjednodušší je tato kategorizace u měřidel, která mají typ přidělený výrobcem a výrobce dodržuje předpoklad, že všechna zařízení stejného typu mají stejné parametry. Jako příklad lze uvést elektrické měřící přístroje. Např. PU500, VDM1, M1T390 jsou zcela konkrétní typy měřidel.

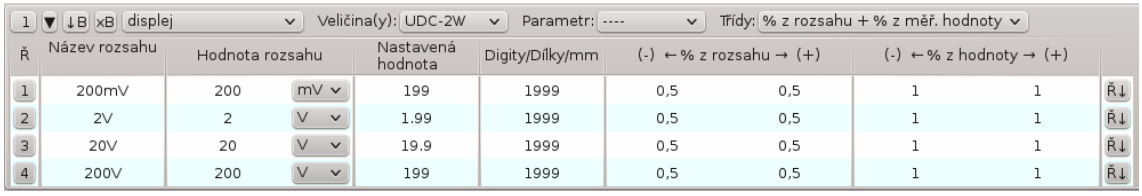
Horší situace je například u posuvných měřítek, na kterých běžně výrobci typ neuvádí. Zde je nutné provést kategorizaci vlastní a rozdělit všechna posuvná měřítka podle jejich délky, dělení stupnice a přesnosti, protože každá z různých kombinací těchto parametrů implikuje jiný měřící postup. Situace může být ještě komplikovanější

u technologických celků, které se vzájemně liší mnoha parametry, z nichž některé se vlastně dodatečně zjišťují v průběhu měření. I z tohoto důvodu jsou formuláře v programu tak flexibilní, aby umožnily snadnou dodatečnou změnu postupu měření.

V levém poli nabídky může uživatel vybrat typ zařízení, pro které se bude vytvářet nový postup, v prostředním může otevřít postup rozpracovaný a v pravém vybrat k editaci některý z hotových postupů. Nabídky též umožňují i trvalé odstranění postupu z databáze. Po stisku tlačítka "Odstranit" je ještě uživatel dotázán, zda si skutečně přeje smazat daný postup v databázi. Po stisku tlačítka "Otevřít" se zobrazí obrazovka s formulářem:

Formulář postupu měření

Na následujícím obrázku je ukázka tabulky formuláře postupu. Ovládací prvky jsou stejné jako ve formuláři karet, který byl představen na začátku. Rozdíly jsou ve struktuře dat:



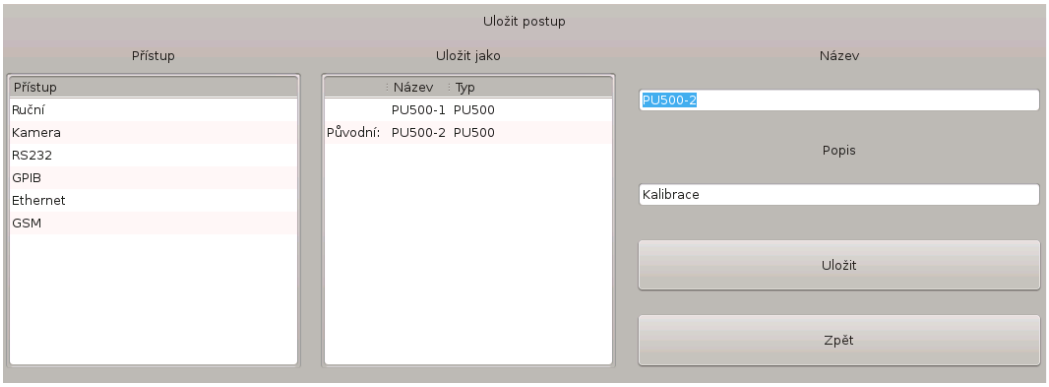
Obrázek 15: Tabulka formuláře postupu měření

Zatímco u karet se zadávalo rozpětí hodnot, ve kterých je přístroj schopen pracovat. Zde se zadávají konkrétní hodnoty, při kterých má být měření provedeno. Jak je na obrázku vidět, původní sloupce "Rozpětí rozsahu" se nyní transformovaly do sloupce "Hodnota rozsahu", zatímco navíc přibyl sloupec "Nastavená hodnota". Tento sloupec představuje jmenovitou hodnotu, která by měla být během měření na daném zařízení (totéž platí o hodnotě parametru měření). Jestli tato hodnota bude skutečně taková, závisí na konkrétních podmínkách měření a rozdíl těchto hodnot může představovat naměřenou chybu.

Uložení postupu

Stejně jako v případě karet, je postup kdykoli možné uložit jako rozpracovaný pomocí ikony "Uložit" na horním panelu aplikace. Při tomto uložení se nekontroluje validita postupu.

Také uložení postupu jako hotového se neliší od uložení karty. Při konečném uložení je kontrolováno správné vyplnění všech položek a v případě chyby je uživatel upozorněn a proces ukládání je přerušen. V případě validního formuláře se zobrazí nabídka uložení postupu. Opět je třeba vybrat v levém poli přístup k měřenému zařízení, v prostředním poli postup k přepsání a v pravém upravit nový název postupu:



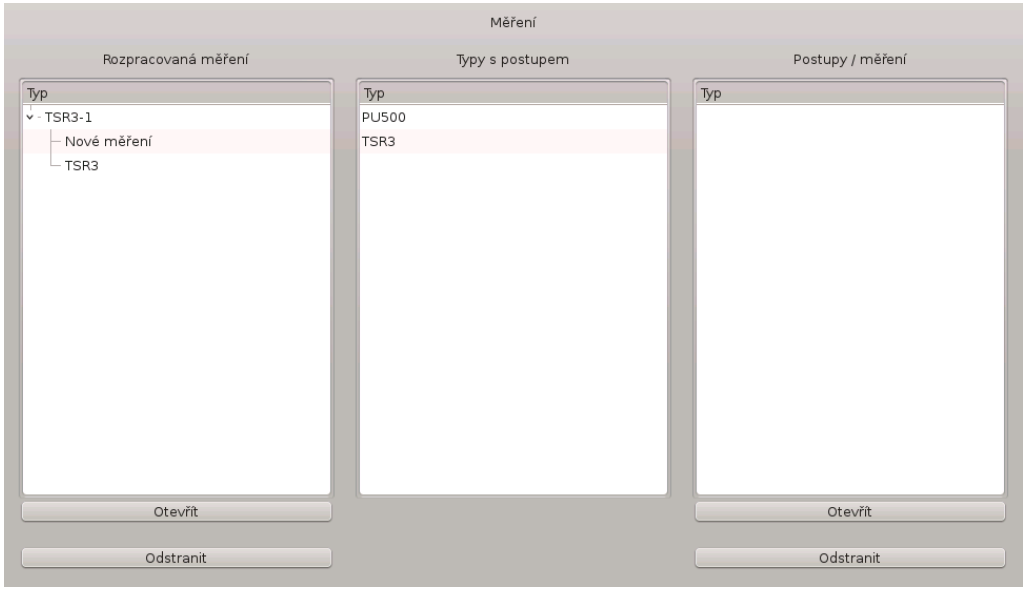
Obrázek 16: Nabídka uložení postupu měření

Konfigurace měření

V okamžiku, kdy uživatel provedl potřebné kroky popsané výše v tomto manuálu, má připravena všechna data k tomu, aby mohl úspěšně dokončit celý proces přípravy měření. Poslední krok má za úkol propojit všechny informace zadané v předchozích kapitolách do jednoho funkčního celku. Protože předchozích dat může být opravdu mnoho, je tato činnost automatizována, aby se odstranilo zdlouhavé postupné procházení mnoha nabídek s rozsahy zařízení.

Úvodní nabídka měření

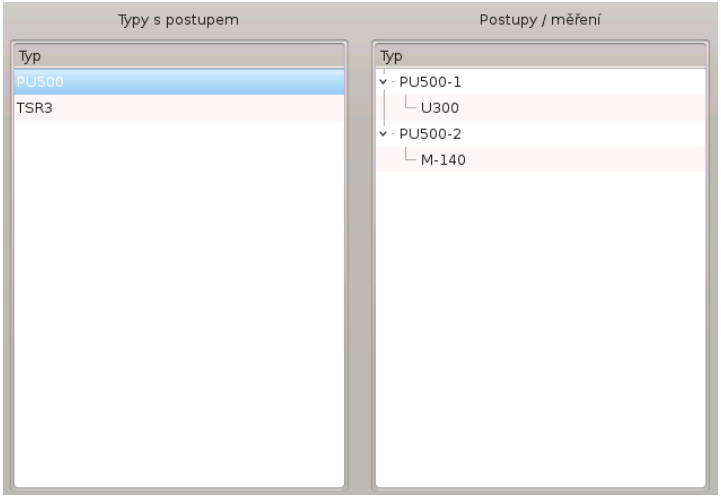
Úvodní nabídka měření na následujícím obrázku má opět tři různá pole, stejně jako karty a postupy, ovšem samotné nabídky se trochu liší. Měření nemůže existovat samo o sobě, ale je vždy svázáno s nějakým měřícím postupem, který doplňuje:



Obrázek 17: Úvodní nabídka měření

V levém poli je tradičně výběr rozpracovaných měření. Nabídka představuje stromovou strukturu. Jak je vidět na obrázku, jedná se vlastně o seznam postupů, pod kterými jsou zavěšeny k nim příslušná měření. K jednomu postupu může existovat neomezené množství měření, která se od sebe mohou lišit například použitými pracovišti.

V prostřední nabídce je vidět seznam typů, pro které existuje nějaký postup a pravá nabídka je v tomto případě prázdná. Má to svůj důvod, který vyvěrá z předpokládaného množství měření. Měřící pracoviště může za několik let svého provozu vytvořit stovky až tisíce různých postupů a ke každému z těchto postupů i několik měření, stalo by se hledání v těchto měřeních velmi nepřehledné. Proto jsou v prostřední nabídce zobrazeny pouze typy zařízení. Poté, co uživatel vybere v nabídce kliknutím nějaký typ, objeví se teprve v pravé nabídce všechny postupy příslušné k tomuto typu a pod nimi zavěšené ve stromové struktuře se objeví měření, jak je vidět na následujícím obrázku:



Obrázek 18: Detail interaktivní nabídky měření

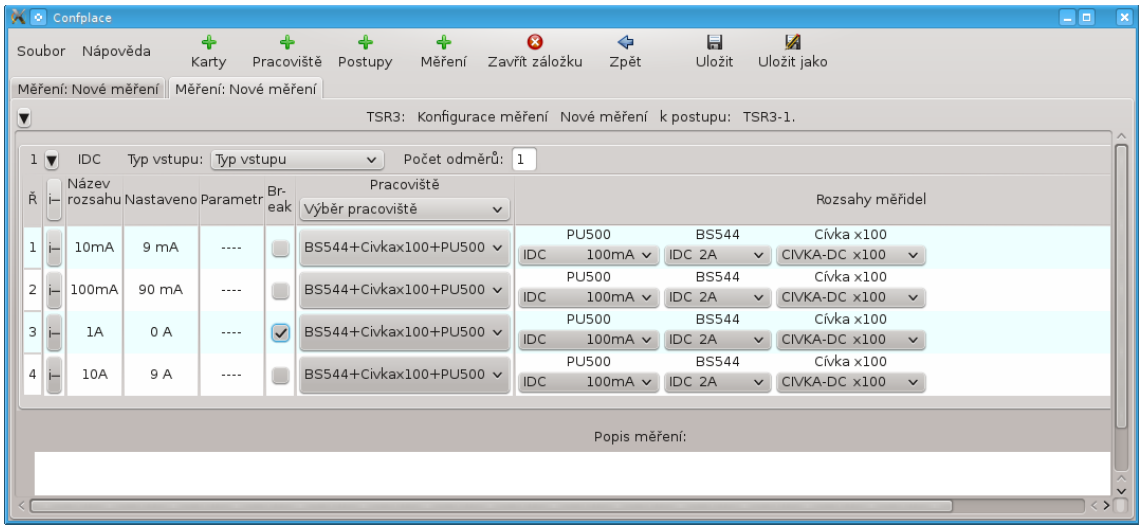
Uživatel má nyní na výběr dvě možnosti. Buď klikne myší na název postupu a stiskem tlačítka "Otevřít", otevře nové měření pro tento postup nebo klikne na název měření a po stisku tlačítka"Otevřít" otevře toto měření pro editaci.

Formulář měření

Formulář měření je na následujícím obrázku. Pro lepší pochopení jeho popisu budiž řečeno, že pracoviště, které je v ukázce formuláře použito, je to samé, které je prezentováno v části tohoto manuálu, který se týká pracovišť. Statické záhlaví formuláře zůstává stejné jako u karet či postupů.

V záhlaví bloku, zcela nahoře vpravo od expandéru bloku, je zobrazena veličina, která je v tomto bloku měřená a je společná pro všechny řádky. Vpravo od ní je výběr typu vstupu. Zde je nutné vybrat jednu ze dvou možností - zda se při měření bude odečítat hodnota z měřeného zařízení nebo z měřícího zařízení. Ještě více vpravo je kolonka pro zadání počtu odměrů. Tato hodnota se využije při automatickém měření pro stanovení nejistoty typu A.

V záhlaví tabulky se vlevo nachází tlačítko s piktogramem "i-". Písmeno "i" v piktogramu znamená slovo "Informace" a tímto tlačítkem se zapíná zobrazení doprovodných informací ke každému řádku, tedy v případě, že takové informace existují. Pokud ano, je o tom uživatel informován piktogramem "i+". Piktogram "i-" tedy znamená, že není co zobrazit. Vpravo od tlačítka, za hlavičkami sloupců popisujících měření, je umístěna nabídka pro hromadný výběr pracoviště. Záměrem je urychlit práci. V praxi bylo zjištěno, že většina bloků používá stejné pracoviště pro všechny řádky měření. Pokud uživatel provede výběr pracoviště v této nabídce, pak se tato informace automaticky nastaví i na všech ostatních nabídkách řádků tabulky. Pokud by snad alespoň jeden řádek používal jiné pracoviště, nic uživateli nebrání v tom, aby poté provedl individuální přenastavení tohoto řádku na jinou hodnotu.



Obrázek 19: Formulář konfigurace měření

V řádku je za jeho číslem opět umístěno tlačítko pro informační pole. Tentokrát s jinou funkcí. Toto tlačítko umožňuje přidat k řádku kalibrace informaci, která se posléze zobrazí obsluze zařízení při vlastním měření. Piktogram "i-" na tlačítku znamená, že k řádku zatím žádná poznámka k měření neexistuje (nebyla přidána v minulosti). Po prvním stisku tlačítka s piktogramem "i-"se nad řádkem objeví textové pole pro zápis poznámky a piktogram se změní na "i+". Takto již zůstane stále. Teprve pokud by byl text z poznámky odstraněn a měření bylo uloženo, při dalším načtení formuláře bude mít tlačítko opět původní piktogram "i-".

Za tlačítkem informačním se nachází tři pole s popisem rozsahu zařízení a další aktivní prvek, kterým je zaškrtávací políčko "Breakpoint". Tato políčka se používají při automatickém měření. Pokud je políčko zatrženo, pak se při automatickém měření před tímto řádkem toto automatické měření zastaví. Je to velmi důležité, pokud je třeba během automatické kalibrace provést operaci, která nejde zautomatizovat. Například změnit parametry nějakého zařízení, které nelze dálkově ovládat nebo přepojit kabely a podobně.

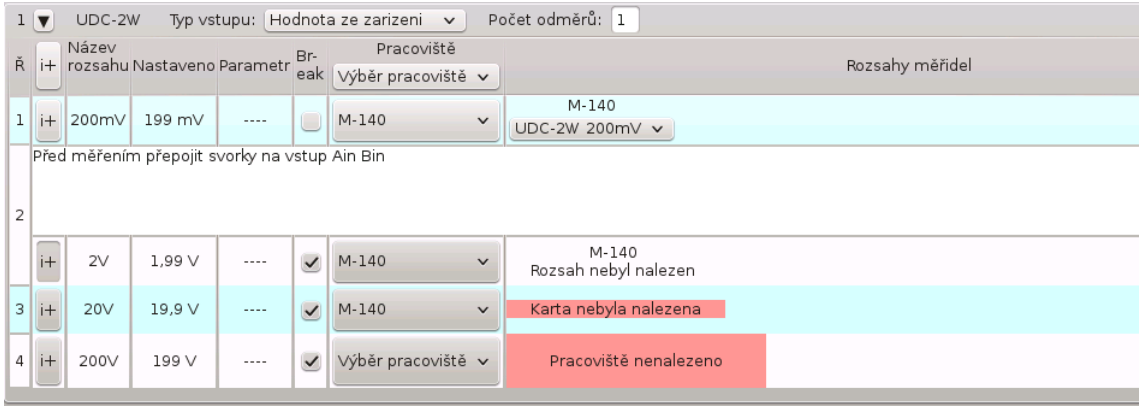
Dále je nabídka pro výběr pracoviště, o které byla řeč již výše. Po výběru pracoviště se automaticky provede jeho výpočet a vpravo od výběru se vypíší všechna zařízení, která byla nalezena v pracovišti spolu se svými rozsahy. Tyto rozsahy jsou nalezeny též automaticky podle hodnot veličiny v pracovišti. Při hledání vhodných rozsahů se vybere rozsah, který první optimálně splňuje podmínky.

Může se stát, že takových rozsahů je více a že v kartě není dostatek údajů pro výběr toho nejlepšího. V tomto případě si uživatel po kontrole nalezených rozsahů může v nabídce rozsahů daného zařízení vybrat jiný. Jako příklad může posloužit přesný zdroj veličiny (napětí proudu, frekvence ..), který má dva rozsahy shodné veličiny a parametru, z nichž jeden je přesnější, ale je méně výkonný, než ten druhý, který má větší výstupní výkon za cenu menší přesnosti. Protože takovéto speciální informace se do karty měřidel neukládají, nemůže se program správně rozhodnout a je nutná ruční

intervence uživatele. Uživatel již většinou o této anomálii dopředu ví a počítá s ní. Platí zde pravidlo, že pokud jsou v kartě takovéto podobné rozsahy, z nichž některý se používá častěji, pak by měl být v kartě uveden jako první a program ho vždy preferuje. To znamená, že se minimalizuje počet špatných rozhodnutí, která je třeba ručně korigovat.

V nabídce pracovišť je i speciální položka "Bez pracoviště". Tato volba se zadá v případě, kdy některý měřící bod nemá dostupné pracoviště, které by šlo použít, ale měření v tomto bodě se bude provádět a údaje se budou zapisovat do měřícího protokolu. Tato volba programu říká, že nejde o chybu, uživatel nezapomněl vybrat pracoviště a měřicí bod se má zachovat. Naměřená hodnota se při měření například může získat od subjektu, který provádí subdodávku měření některých parametrů vlastní aparaturou.

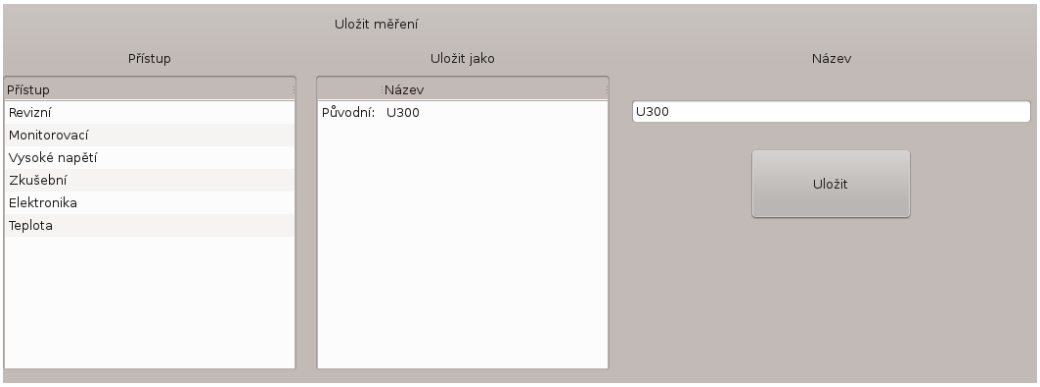
Pokud uživatel vybere pracoviště, jehož všechna měřidla nepokrývají svými parametry druh a velikost veličiny v pracovišti, pak se ve výpisu měřidel a rozsahů vypíše tato informace: "Rozsah nebyl nalezen". Také se může stát, že během životnosti konfigurace měření dojde ke změně některého pracoviště nebo ke změně v kartě zařízení, které je součástí pracoviště. Pokud program toto rozpozná, vypíše tuto informaci v červeném poli jako chybový stav. Uživatel programu pak může vybrat jiné pracoviště, které splňuje požadavky nebo nejprve opravit konfiguraci pracoviště a posléze se opět vrátit ke konfiguraci měření. Následující obrázek ukazuje některé chybové stavy. Na obrázku je vidět i použití tlačítka pro zapnutí textového pole poznámky, jak bylo uvedeno výše.



Obrázek 20: Chybové stavy po načtení měření

Uložení měření

Pro uložení měření je k dispozici již známá obrazovka, kde se v levém poli vybere přístup k měření, v prostředním poli měření k přepsání a v pravém poli je možné změnit název měření:



Obrázek 21: Nabídka uložení měření