VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV ELEKTROTECHNOLOGIE
DEPARTMENT OF ELECTRICAL AND ELECTRONIC TECHNOLOGY

VÝVOJ MOBILNÍCH APLIKACÍ PRO VZDÁLENÉ OVLÁDÁNÍ PŘÍSTROJOVÉ TECHNIKY
MOBILE APPLICATIONS DEVELOPMENT FOR REMOTE CONTROL OF INSTRUMENTATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Michal Macek**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Martin Frk, Ph.D.**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Elektrotechnická výroba a materiálové inženýrství**
Ústav elektrotechnologie

Student: Bc. Michal Macek ***ID:*** 147441
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Vývoj mobilních aplikací pro vzdálené ovládání přístrojové techniky

POKYNY PRO VYPRACOVÁNÍ:

S využitím dostupné literatury zpracujte rešerši o vývoji a možnostech jednotlivých datových sběrnic využívaných v laboratorní a průmyslové měřicí technice. Analyzujte možnosti jejich využití pro vzdálenou komunikaci a ovládání přístrojové techniky prostřednictvím sítě Internetu. Seznamte se s programovacími jazyky pro tvorbu mobilních aplikací na platformě Android.
V libovolném programovacím jazyce vytvořte pro vybranou přístrojovou techniku sadu univerzálních aplikací respektujících následující požadavky:
- jednotné grafické prostředí umožňující správu a komfortní obsluhu daného přístroje (multimetr, osciloskop, generátor, RLC metr,…),
- možnost ovládání ostatních přístrojů pomocí samostatných SCPI příkazů z příkazového řádku nebo pomocí importovaných sekvencí příkazů,
- modulární uskupení, tj. možnost snadného a dodatečného rozšíření aplikace o další funkce,
- možnost opakovaného měření dané veličiny, včetně základního statistického vyhodnocení a exportu ve vhodném formátu pro další zpracování na počítači.
K příslušným aplikacím vytvořte návod s detailním popisem jednotlivých programátorských kroků a postupem zpracování. Praktickou funkčnost mobilních aplikací demonstrujte na zařízení pracujícím na platformě Android.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucho práce.

Termín zadání: 8.2.2016 ***Termín odevzdání:*** 26.5.2016

Vedoucí práce: Ing. Martin Frk, Ph.D.
Konzultant diplomové práce:

doc. Ing. Petr Bača, Ph.D., *předseda oborové rady*

UPOZORNĚNÍ:
Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT:

Předkládaná práce popisuje rozšiřující se trend využívající mobilní zařízení k ovládání laboratorních přístrojů a zobrazování naměřených dat. Hlavním cílem je vývoj aplikace pro mobilní zařízení, díky které bude možné vzdáleně ovládat a měřit s přístroji v laboratoři. Praktická část je zaměřena na naprogramování mobilní aplikace, která s využitím místní sítě bude schopna připojit se k více laboratorním zařízením, nastavit je pro různé druhy měření a následně zobrazit naměřená data.

ABSTRACT:

The presented work describes the growing trend of using mobile devices to control laboratory instruments and displaying of measured data. The main objective is the application development for mobile devices, which will allow to remotely control and measure with laboratory instruments. The practical part is focused to programming mobile application which will be able by using the local network to connect multiple laboratory instruments, setting them for different types of measurements and then display measured data.

KLÍČOVÁ SLOVA:

Agilent, Keysight, Keithley, Hewlett-Packard, Android, Java, aplikace, čtení dat, měření, měřící přístroj, nastavení, ovládání, programování, komunikace, vzdálené připojení, vzdálené ovládání, vzdálené měření.

KEYWORDS:

Agilent, Keysight, Keithley, Hewlett-Packard, Android, Java, application, data reading, measurement, measuring instrument, settings, control, programming, connection, remote access, remote control, remote measure.

BIBLIOGRAFICKÁ CITACE DÍLA:

MACEK, M. *Vývoj mobilních aplikací pro vzdálené ovládání přístrojové techniky*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 63 s. Vedoucí diplomové práce Ing. Martin Frk, Ph.D..

PROHLÁŠENÍ AUTORA O PŮVODNOSTI DÍLA:

Prohlašuji, že svou diplomovou práci na téma Vývoj mobilních aplikací pro vzdálené ovládání přístrojové techniky jsem vypracoval samostatně pod vedením vedoucího diplomové práce, s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a~jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....
(podpis autora)

PODĚKOVÁNÍ:

Děkuji vedoucímu diplomové práce Ing. Martinu Frkovi, Ph.D. za metodické a cíleně orientované vedení při plnění úkolů realizovaných v návaznosti na diplomovou práci, za konzultace a trpělivost. Dále děkuji ústavu Elektrotechnologie za poskytnutí prostoru k realizaci experimentálních prací, zapůjčení měřících přístrojů a za vstřícnost při řešení nastalých situací..

V Brně dne ....
(podpis autora)

OBSAH

OBSAH **6**
SEZNAM OBRÁZKŮ **8**
ÚVOD **10**
1. OPERAČNÍ SYSTÉM ANDROID **11**
1.1. Obecný popis 11
1.2. Aktuální verze 11
2. KONEKTIVITA LABORATORNÍCH PŘÍSTROJŮ **12**
2.1. CAN bus 12
2.2. RS-232 12
2.3. GPIB 13
2.4. VME standard 13
2.5. VXI standard 14
2.6. USB 14
2.7. LXI 15
3. OVLÁDÁNÍ PŘÍSTROJŮ POMOCÍ SCPI PŘÍKAZŮ **16**
3.1. Příklady syntaxe příkazů pro přístroje Agilent 17
4. APLIKACE LABIC **19**
4.1. Popis funkce aplikace 19
4.1.1. Rozhraní Osciloskopu 22
4.1.2. Rozhraní Generátoru 26
4.1.3. Rozhraní Multimetru 28
4.1.4. Rozhraní RLC metru 31
4.1.5. Rozhraní příkazového řádku 35
5. APLIKACE LABIC - ROZBOR HLAVNÍCH ČÁSTÍ ZDROJOVÉHO KÓDU **37**
5.1.1. Použité knihovny 37
5.1.2. Odesílání nastavení do přístroje 37
5.1.3. Sestavení příkazu pro nastavení přístroje 39
5.1.4. Příjem dat z přístroje 40
5.1.5. Zpracování přijatých dat 41
5.1.6. Funkce krokového měření 43
5.1.7. Zpracování dat pro blok grafu 44
5.1.8. Zobrazení grafu 45
5.1.9. Zpracování dat pro tabulku a její zobrazení 46

5.1.10. Statusy - zobrazování obsahu podle situace v aplikaci a vytváření informačních zpráv pro uživatele 47
5.1.11. Uložení dat do CSV 48
5.1.12. Zobrazování měřených veličin 49
5.1.13. Zobrazování seznamu v dialogu pro výběr funkce 50
6. ZÁVĚR 51
SEZNAM POUŽITÝCH ZDROJŮ 52
SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK 55
SEZNAM PŘÍLOH 56

SEZNAM OBRÁZKŮ

Obr. 3.1: Příklad programátorské nápovědy pro SCPI. 16
Obr. 4.1: Aplikace LABIC - úvodní stránka. 19
Obr. 4.2: Aplikace LABIC - stránka pro nastavení IP adres. 20
Obr. 4.3: Aplikace LABIC - zobrazeni po načtení názvů přístrojů. 20
Obr. 4.4: Aplikace LABIC - zobrazení seznamu po návratu z nastavení IP adres. 21
Obr. 4.5: Aplikace LABIC, rozhraní Osciloskop - úvodní obrazovka. 22
Obr. 4.6: Aplikace LABIC, rozhraní Osciloskop - výběr rozlišení grafu. 23
Obr. 4.7: Aplikace LABIC, rozhraní Osciloskop - nastavení generátoru integrovaného v Osciloskopu. 24
Obr. 4.8: Aplikace LABIC, rozhraní Osciloskop - dialog pro výběr průběhu. 24
Obr. 4.9: Aplikace LABIC, rozhraní Osciloskop - příklad omezení vstupů podle průběhu. 25
Obr. 4.10: Aplikace LABIC, rozhraní Generátor - načítání aktuálního nastavení. 26
Obr. 4.11: Aplikace LABIC, rozhraní Generátor - zobrazení chybového statusu po neúspěšném spojení s přístrojem. 26
Obr. 4.12: Aplikace LABIC, rozhraní Generátor - karta nastavení. 27
Obr. 4.13: Aplikace LABIC, rozhraní Generátor - výběr funkce a zobrazené nastavení. 27
Obr. 4.14: Aplikace LABIC, rozhraní Multimetr - načtení aktuálního nastavení a změření veličiny. 28
Obr. 4.15: Aplikace LABIC, rozhraní Multimetr - opakované měření veličiny. 29
Obr. 4.16: Aplikace LABIC, rozhraní Multimetr - zobrazení naměřených hodnot a informace o uložení CSV souboru. 29
Obr. 4.17: Aplikace LABIC, rozhraní Multimetr - záložka nastavení a výběr měřené veličiny. 30
Obr. 4.18: Aplikace LABIC, rozhraní Multimetr - výběr rozsahu a možnosti nastavení pro měření teploty. 30
Obr. 4.19: Aplikace LABIC, rozhraní RLC metr - načtení a zobrazení aktuální konfigurace. 31
Obr. 4.20: Aplikace LABIC, rozhraní RLC metr - krokové měření a zpřístupnění tlačítek pro vyhodnocení dat. 32
Obr. 4.21: Aplikace LABIC, rozhraní RLC metr - zobrazení naměřených dat v tabulce. 32
Obr. 4.22: Aplikace LABIC, rozhraní RLC metr - zobrazení naměřených dat v grafu. 33
Obr. 4.23: Aplikace LABIC, rozhraní RLC metr - exportování naměřených dat do CSV a informování uživatele 33
Obr. 4.24: Aplikace LABIC, rozhraní RLC metr - možnosti nastavení. 34
Obr. 4.25: Aplikace LABIC, rozhraní Příkazový řádek - hlavní obrazovka. 35
Obr. 4.26: Aplikace LABIC, rozhraní Příkazový řádek - odeslání příkazu a načtení dat. 36
Obr. A.1: Prostředí programu Android Studio. 57
Obr. A.2: SDK Manager - výběrem platforem a nástrojů. 58
Obr. A.3: SDK Manager - odsouhlasení licenčních podmínek. 58
Obr. A.4: Android studio - pojmenování a lokace aplikace. 59
Obr. A.5: Android studio - výběr platformy a minimální verze systému. 60

Obr. A.6: Android studio - výběr aktivity. 60
Obr. A.7: Android studio - pojmenování aktivity. 61
Obr. A.8: Android studio - automatický zdrojový kód Hello World aplikace ve vývojovém prostředí. 61
Obr. A.9: Android studio - spuštění emulátoru. 62
Obr. A.10: Android studio - výběr zařízení. 62
Obr. A.11: Android studio - okno emulátoru. 62

ÚVOD

Rozšíření chytrých mobilních zařízení v posledních letech přineslo obrovskou produkci aplikací na tyto platformy. Aplikace pomáhají uživatelům usnadnit práci i život a každý den takových aplikací přibývá.

Nejrozšířenějším operačním systémem pro mobilní zařízení je momentálně Android. Pro svoji jednoduchost a funkčnost se dostal nejen do mobilních telefonů a tabletů, ale i do palubních počítačů automobilů, spotřební elektroniky, bíle techniky, ovládání domácností a různých dalších zařízení. Výhoda aplikací u tohoto operačního systému je v jejich rozmanitosti a ve funkčnosti mezi různými verzemi systému. Také díky rychlému vývoji elektroniky a s tím i souvisejícím rostoucím výkonem můžeme na mobilních zařízeních již nyní provozovat aplikace téměř stejné jako na stolních počítačích či noteboocích.

Stejně, jako je většina počítačového softwaru postupně přesouvána z disků osobních počítačů na serverové disky v podobě cloudového řešení, můžeme očekávat podobné kroky v prostředí měřící techniky. Již v dnešní době je možné k velkému množství měřící techniky přistupovat přes webové rozhraní a k nemalé části měřící techniky lze přistupovat i z mobilních telefonů. Tento trend v budoucnu pravděpodobně nadále poroste, přinese spoustu nových funkcí a umožní ještě lépe využít potenciál měřících přístrojů.

Využití mobilních platforem k ovládání přístrojové techniky je krokem kupředu. Tento krok otevírá velké možnosti a prostor pro improvizaci a zaplnění určitého místa na trhu s aplikacemi. Aplikace vytvořená na míru přístroje je mocnou zbraní a pomocníkem zároveň.

1. OPERAČNÍ SYSTÉM ANDROID

1.1. Obecný popis

Operační systém Android je open source platforma, kterou vytvořila společnost Google, jedná se tedy o počítačový software. Díky otevřenému zdrojovému kódu je licenčně a technicky snadno dostupný. Uživatel může využívat systém zadarmo a má přístup ke zdrojovým kódům, pokud splní určité podmínky použití softwaru. Operační systém běží na jádru Linux, který zabezpečuje funkčnost systému jako celku, správu paměti, správu procesů, přístup ke službám, ovladačům a různých komponentám. Jednotlivé aplikace k funkcím jádra nepřistupují přímo, ale pomocí Android API. Android byl hlavně vyvíjen jako progresivní operační systém pro PDA, tablety a mobilní telefony.

Systém umožňuje vývojářům vytvářet různorodé aplikace. Díky použití otevřeného jádra Linux jako vlastního virtuálního stroje, dokáže systém optimalizovat paměť a hardwarové prostředky. Díky rostoucí vývojářské komunitě a podpoře výrobců mobilních zařízení se tento systém rozvijí obrovskou rychlostí. Od první verze bylo vydáno několik aktualizací, které opravují chyby a přidávají nové funkce. Od verze 1.5 má každá verze své vlastní jméno podle zákusků (Jelly Bean, KitKat, Lollipop atd.) [1].

1.2. Aktuální verze

Nejnovější operační systému Android je verze 6.0 nazvaný Marshmallow. Dostupný je v nejnovějších mobilních telefonech a lze jej stáhnout i na několik vybraných starších modelů. Přináší nové grafické prostředí a spoustu nových funkcí. Hlavním vylepšením je menší spotřeba baterie, díky automatickému přecházení do režimu spánku a omezením aplikací v tomto režimu pracovat. Vylepšena byla i bezpečnost, kdy je nyní uživatel dotazován zda může aplikace přistupovat ke službám, které se chystá využívat [30].

2. KONEKTIVITA LABORATORNÍCH PŘÍSTROJŮ

Většina dnešní laboratorní měřící techniky již disponuje rozhraním LAN pro připojení přístroje do sítě. Pokud je tak učiněno, přístroj je možné ovládat z počítačů v této síti přes webové rozhraní. V případě, že je síť ve které je připojen přístroj nakonfigurována pro přístup z vnější sítě internet, je možné přístroj ovládat z jakéhokoliv počítače s přístupem k internetu. Veškeré přístroje firem jako Agilent, Keithley, Keysight a Hewlett-Packard obsahují alespoň jedno z následujících komunikačních rozhraní: RS-232, GPIB, USB, LAN, případně i jiná rozhraní. Přes všechna tato rozhraní je možné přístroje ovládat z počítače ke kterému jsou připojeny.

2.1. CAN bus

Sběrnice CAN byla vytvořena pro použití v osobních automobilech na komunikaci mezi řídícími jednotkami a prvky. Firma Bosch ji v 90. letech rozšířila i na jiná technická zařízení. Systém umožňuje v reálném čase poměrně efektivní, decentralizované řízení s velmi vysokou spolehlivostí přenosu dat. Dále systém zajišťuje minimální prodlevu při režijních činnostech, centralizovaný příjem zpráv z více zdrojů, detekci a signalizaci chyb přenosu a další. Využití CAN sběrnice je v dnešní době spíše v automobilech, v průmyslových aplikacích jen zřídka [2].

2.2. RS-232

RS-232 je rozhraní vytvořené pro přenos informací mezi dvěma zařízeními. Toto rozhraní je v dnešní době možné nalézt spíše u starších laboratorních přístrojů, případně u specifických aplikací průmyslové techniky. Osobní počítače tímto rozhraním již delší dobu také nedisponují.

Laboratorní přístroj připojený přes rozhraní RS-232 k počítači je možné ovládat pouze z tohoto počítače za pomocí naprogramovaného softwaru nebo s použitím SCPI příkazů. Dále existují různé možnosti k připojení více přístrojů pomocí RS-232 k jednomu počítači, například rozhraní Agilent E5805A, které obsahuje jeden USB port pro připojení k počítači a pro připojení přístrojů čtyři porty RS-232. Tímto způsobem je tedy možné k jednomu počítači připojit až čtyři měřící přístroje. Další možností je redukce z rozhraní RS-232 na rozhraní USB, která se připojí na kabel mezi přístroj a počítač, nainstalují se příslušné ovladače a přístroj je připraven k použití [3][4].

2.3. GPIB

Jedná se o paralelní, otevřený komunikační standard se 16 bitovou sběrnicí, na kterou může být připojeno až 31 zařízení s délkou kabelu 20 m. Topologie této sítě může být lineární, nebo typu hvězda. Sběrnice je obdobně stará jako RS-232, ale oproti ní nabízí možnost propojení přístrojů do jednoho hostitelského zařízení bez použití dalšího komunikačního zařízení. GPIB používá IEEE.488 standard respektive jeho poslední verzi IEEE.488.2 z roku 1987, která specifikuje minimální požadavky portu v přístrojích, základní příkazy a formát přenášených dat. Dříve byla tato sběrnice známa pod označením HP-IB (Hewlett-Packard Interface Bus), protože vznikla právě ve firmě Hewlett-Packard. Později byla sběrnice přejmenována na GPIB (General-Purpose Interface Bus) [5].

Přístroje připojené přes toto rozhraní je opět možné ovládat pomocí počítače, do kterého je připojen. Další z možností je přístroj připojit do speciální brány, kterou lze zapojit do sítě LAN. Díky tomu je možné přistupovat k přístroji z různých počítačů, které mají přístup do sítě. Ovládání je možné opět pomocí softwaru a příkazů SCPI. I u tohoto rozhraní existuje redukce z GPIB na USB, která se připojí na kabel mezi přístroj a PC.[3][4]

2.4. VME standard

Jedná se o standard počítačové sběrnice, která byla původně vyvinuta pro procesory firmy Motorola a to konkrétně pro řadu 68000. Později se tato sběrnice rozšířila i do jiných aplikací. VME bylo vyvinuto roku 1981 a v průběhu let se objevily nové verze s dalšími novými vlastnostmi a vylepšeními. Například roku 1982 byl vydán nový standard VME64 podporující 64 bitovou komunikaci [7].

Standard ve výsledku obsahuje tři velikosti modulů (karet) ve formátu EuroCard. Na každém modulu je rozdílný počet konektorů, nejmenší modul 3U obsahuje pouze jeden, střední modul 6U obsahuje dva a největší modul 9U obsahuje tři konektory. Využití tohoto standardu je spíše v průmyslových aplikacích, kde je potřebné ovládat velké množství měřících přístrojů a dalšího vybavení [7].

2.5. VXI standard

Sběrnice VXI (VMEbus Extensions for Instrumentation) byla vyvinuta v roce 1987 společnostmi Colorado Data Systems, Hewlett-Packard, Racal Dana a Tektronix. Jejím účelem bylo poskytnout standardizovanou, otevřenou, modulární architekturu pro ovládání průmyslových a laboratorních přístrojů. Verze VXI-1 revize 1 používá plnou 32 bitovou VME architekturu a přidává dva rozměry desky spolu s jedním konektorem. VXI moduly jsou tedy dostupné ve třech velikostech: nejmenší má označení B a obsahuje dva konektory, střední se značí C, také obsahuje dva konektory a největší se značí D a nese 3 konektory. Revize 3 přidala VME s 64 bitovým přenosem dat a dalšími vylepšeními [6][7].

VXI platforma přináší [7]:

- precizní časovou koordinaci mezi přístroji,
- schopnost zvládnout náročnější aplikační požadavky,
- zvýšenou propustnost systému,
- menší rozměry díky vyšší integraci,
- otevřené standardy ,
- standardizovaný software,
- modulární a robustní konstrukci,
- podporu různých dodavatelů.

Připojení sběrnice VXI k počítači může být realizováno pomocí PCI karty zapojené v PC nebo pomocí rozhraní GPIB, MXI2, USB, FireWire, LAN a dalších. K ovládání je použit standardizovaný software pro komunikaci přes tuto platformu. Celkově v systému může být zapojeno až 256 zařízení a mohou mezi ně patřit i další systémy [6][7].

2.6. USB

USB je v dnešní době jedno z nejrozšířenějších rozhraní pro připojení periferií a přístrojů k počítači. Je to univerzální sériová sběrnice nahrazující původní rozhraní jako je RS-232, GPIB a další. Díky své univerzálnosti a standardizaci umožňuje propojení s většinou dnešních přístrojů a počítačů a to hlavně díky technologii Plug & Play, která zajišťuje rozpoznávání připojeného zařízení bez nutnosti instalovat ovladače. Výhodou je také možnost připojení více zařízení k jednomu počítači díky rozbočovačům. Délka USB kabelu může být až 5 metrů, pokud je potřeba delšího kabelu, je nutné použít aktivní člen na každých 5 m pro zachování potřebné úrovně signálu.

Připojit laboratorní přístroj pomocí USB je pravděpodobně jeden z nejjednodušších způsobů jak přístroj ovládat přes počítač. Přístroj stačí připojit a nainstalovat software pro ovládání. Při použití brány Agilent E5813A, která obsahuje 4 USB porty a jeden ethernet port je možné připojit i více přístrojů a napojit je do místní sítě, ze které je bude možné ovládat z jakéhokoliv počítače v této síti. Využití USB je spíše vhodné pro malé měřící systémy [3][4].

2.7. LXI

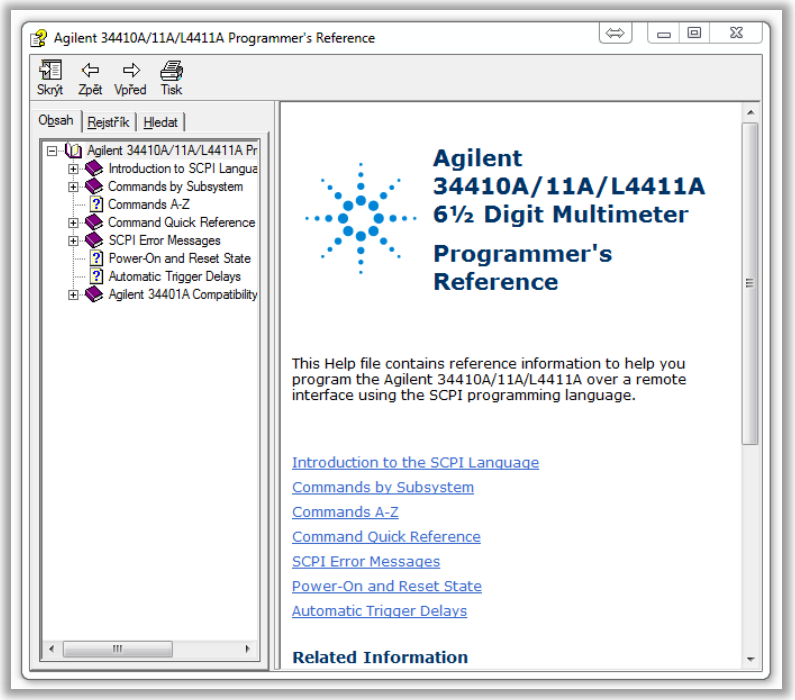
Platforma LXI (LAN eXtensions for Instrumentation) je založena na standardu Ethernet. V roce 2004 začala firma Agilent toto rozhraní integrovat do svých přístrojů. Momentálně se jedná o jeden z nejlepších způsobů připojení měřících přístrojů, protože kombinuje dobré vlastnosti rozhraní GPIB a VXI. Jedná se o podobně jednoduché připojování jako u sběrnice USB, hlavně pokud místní síť podporuje dynamické přidělování IP adres. V tomto případě je nakonfigurování přístroje stejně jednoduché jako v případě USB. Většina přístrojů v nabídce výrobců je již vybavena rozhraním pro tento typ připojení. LXI platforma je vhodna na menší a střední měřící systémy [8][9].

Přístroj je připojen do sítě LAN standardním ethernet kabelem s konektory RJ-45. V případě aktivního DHCP v síti je měřícímu přístroji automaticky přidělena IP adresa a pokud není DHCP aktivní, je třeba síťové rozhraní v přístroji nakonfigurovat podle nastavení sítě. Pokud přístroj v sobě obsahuje firmware s webovým rozhraním, stačí otevřít internetový prohlížeč na počítači připojeném do stejné sítě jako je přístroj a zadat IP adresu měřícího přístroje. Pokud je přístroj správně nastaven v síti, dojde k načtení webového rozhraní pro jeho ovládání. Ovládání pomocí tohoto rozhraní je velice jednoduché, graficky je vytvořeno přesně podle předního panelu přístroje včetně tlačítek a ostatních prvků. Měřené hodnoty jsou zobrazen tak, jak jsou viditelné na displeji přístroje [3][4].

3. OVLÁDÁNÍ PŘÍSTROJŮ POMOCÍ SCPI PŘÍKAZŮ

Většina přístrojů firem Agilent, Keithley, Keysight, Hewlett-Packard vybavených jedním z rozhraní uvedených v kapitole 2, je možné ovládat pomocí příkazů SCPI. Jedná se o standard, který shrnuje příkazy a pravidla pro komunikaci mezi řídící jednotkou a přístrojem v automatizovaném měřícím systému. Je nezávislý na technickém řešení i na protokolu přenosu dat [10][31]. [10]

Protože přístroje obsahují jiné funkce podle jejich zaměření a tím pádem i jiné ovládací příkazy, je kompletní přehled příkazů i s nápovědou, syntaxí a příklady použití dostupný v podobě Windows nápovědy (Obr. 3.1), nebo v podobě programátorského manuálu. Tuto dokumentaci je možné nalézt na optickém médiu dodaném k přístroji, nebo je možné stáhnout ji na stránkách výrobce [31].



Obr. 3.1: Příklad programátorské nápovědy pro SCPI.

3.1. Příklady syntaxe příkazů pro přístroje Agilent

Jako základní příkaz je možné uvést zjištění, kterou verzi jazyka SCPI přístroj podporuje. Pomocí rozhraní dálkového ovládání (například Android aplikace) odešleme příkaz ve formě řetězce textu [31]:

```
SYSTem:VERSion?;   nebo  SYST:VERS?;
```

Přístroj nám odpoví ve formě `RRRR.V`, zde `RRRR` označuje rok a `V` číslo verze pro příslušný rok.

Příklad syntaxe nastavení pro měření teploty u přístroje Agilent 34410A. Do pole typ termistoru se uvádí `FTHermistor` nebo `THERmistor podle použitého termistoru připojeného k přístroji`. Do pole velikost odporu se uvádí hodnota odporu termistoru a to: `2252, 5000` nebo `10000 [31].`

```
CONFigure:TEMPerature <typ termistoru>, <velikost     odporu>;
```

Výsledný zápis může vypadat následovně:

```
CONF:TEMP THER, 5000;
```

Předchozí příkaz musí být použit s příkazem `INITiate.` Tento příkaz provede měření s nastavenými parametry z příkazu `CONF:`

```
INITiate; nebo INIT;
```

Naměřenou hodnotu načteme z přístroje pomocí příkazu:

```
FETCh?; nebo FETC?;
```

Odpověď přístroje je například: `+25.27150000`.

Ukázka syntaxe příkazu, který po odeslání do přístroje vrací odesílateli informace o výrobci přístroje, modelu, sériovém číslu a revizi softwaru nainstalovaného v přístroji:

```
*IDN?
```

Přístroj odpoví ve formě:

```
AGILENT TECHNOLOGIES,<model>,<serial number>,X.XX.XX <NL>
```

Kde první pozice označuje výrobce, následující pozice `<model>` značí model přístroje, například DOS-X 2012A, `<serial number>` značí sériové číslo přístroje a `X.XX.XX` označuje revizi softwaru přístroje [31].

Příklad syntaxe pro nastavení frekvence generátoru u přístroje DSO-X 2012A:

```
:WGEN:FREQuency <frequency>
```

kde <frequency> značí pozici pro číselný údaj v Hz pro frekvenci. Výsledný zápis například pro 1kHz vypadá následovně:

```
:WGEN:FREQuency 1000
```

nebo kratší forma:

```
:WGEN:FREQ 1000
```

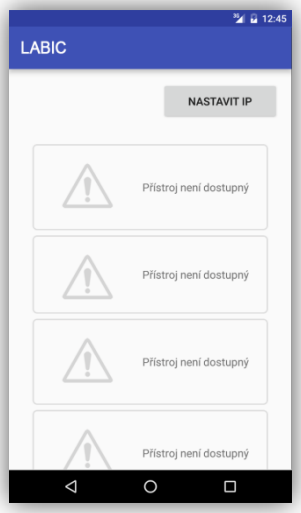
Veškeré příkazy jsou uvedeny v příslušné programátorské příručce každého přístroje, jak je uvedeno v kapitole č.3.

4. APLIKACE LABIC

Aplikace LABIC (LAboratory Instrument Controller) jejíž hlavním účelem je poskytnout obsluze laboratorních měřících přístrojů jednoduchý a pohodlný nástroj pro jejich vzdálené ovládání. Jedná se o univerzální, uživatelsky přívětivé a dynamické rozhraní, které umožní uživateli ovládat několik vybraných laboratorních přístrojů pomocí mobilního zařízení. Každý z přístrojů má v aplikaci vytvořené vlastní rozhraní pro ovládání jeho hlavních funkcí. V případě, že je potřebné ovládat přístroj, který nemá v aplikaci vytvořené rozhraní, je možné jej ovládat pomocí příkazového řádku použitím SCPI příkazů. Tyto příkazy pro ovládání jsou v laboratorní technice velice rozšířené a proto má aplikace široké uplatnění.

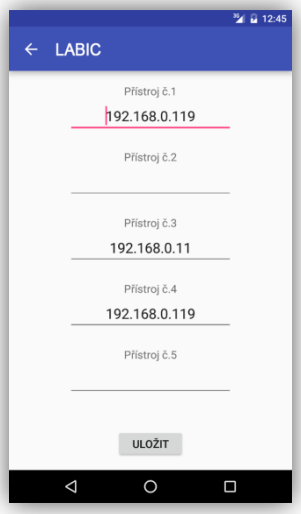
4.1. Popis funkce aplikace

Po instalaci a spuštění aplikace je zobrazena úvodní stránka s několika volnými pozicemi pro přístroje. V momentě spuštění aplikace ještě nejsou přístroje připojeny a tak je zobrazena informace o nedostupnosti s příslušným symbolem (Obr. 4.1). V pravém horním rohu se nachází tlačítko *Nastavit IP*, toto tlačítko uživatele přesune na novou obrazovku, ve které je vyzván k zadání IP adres pro jednotlivé přístroje.



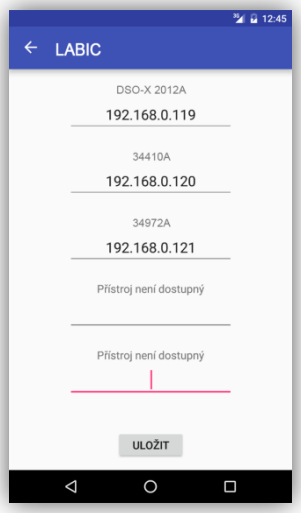
Obr. 4.1: Aplikace LABIC - úvodní stránka.

Na obrazovce pro zadávání IP adres se nachází několik standardních textových polí pro zadání adres jednotlivých přístrojů ve formátu IPv4. Každé jedno pole reprezentuje rozhraní pro přístroj a vkládání textu je softwarově omezeno pouze na číslice a tečku. Není nutné vyplňovat všechna pole, stačí vyplnit tolik polí, kolik je potřebné ovládat přístrojů. Po zadání adres je nutné konfiguraci uložit a to za pomocí tlačítka *Uložit*, které se nachází na konci stránky (Obr. 4.2).



Obr. 4.2: Aplikace LABIC - stránka pro nastavení IP adres.

Toto tlačítko uloží data, spustí komunikaci s přístroji na zadaných IP adresách a zažádá o název zařízení. V momentě kdy dorazí data s názvy přístrojů, jsou nad zadanými IP adresami zobrazeny tyto názvy (Obr. 4.3). Uživatel má nyní možnost data upravit, například pokud zadal špatnou IP adresu a nebo se může vrátit na předchozí obrazovku zobrazující seznam přístrojů.



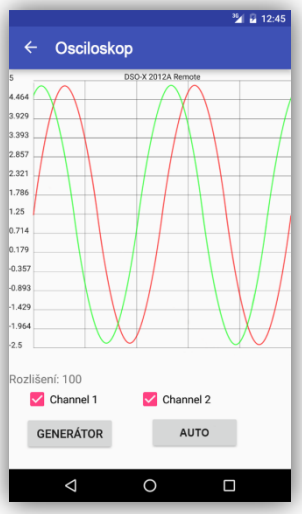
Obr. 4.3: Aplikace LABIC - zobrazeni po načtení názvů přístrojů.

Při návratu uživatele na úvodní obrazovku, je seznam přístrojů naplněn vždy obrázkem a názvem přístroje, které aplikace zjistila podle zadané IP adresy v předchozím zobrazení (Obr. 4.4). Seznam přístrojů je tedy dynamicky nastavován podle dat zadaných na stránce s IP adresami. To znamená, že se přístroje načtou v takovém pořadí, v jakém byly zadány jejich IP adresy. Jednotlivé položky seznamu slouží jako tlačítka ke spuštění rozhraní pro ovládání konkrétního přístroje. Uživatel si tedy vybere který přístroj potřebuje ovládat a stiskem na tuto položku otevře nové okno s ovládáním pro tento přístroj.

Obr. 4.4: Aplikace LABIC - zobrazení seznamu po návratu z nastavení IP adres.

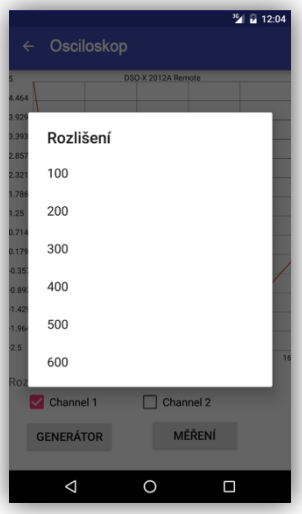
4.1.1. Rozhraní Osciloskopu

Pokud je připojen přístroj Agilent DSO-X 2012A, nebo osciloskop s podobnou syntaxí příkazů, je zobrazen v seznamu přístrojů [11]. Po kliknutí na jeho položku v seznamu se zobrazí nové okno s rozhraním pro ovládání tohoto přístroje (Obr. 4.5). Po zobrazení je automaticky spuštěno načítání dat z přístroje pro první kanál a data jsou zobrazována v grafu. V okně rozhraní se kromě samotného vykresleného průběhu signálu nachází dvě výběrová políčka pro nastavení zobrazení kanálů *Channel 1* a *Channel 2*. Tato výběrová políčka zapínají nebo vypínají zobrazení a načítání dat příslušných kanálů. Dále se zde nacházejí tlačítka *Generátor*, *Auto* a textové tlačítko *Rozlišení*.



Obr. 4.5: Aplikace LABIC, rozhraní Osciloskop - úvodní obrazovka.

Po kliknutí na text *Rozlišení* je zobrazen dialog, kde si uživatel nastaví požadovanou hodnotu z výběru možných rozlišení pro měření (Obr. 4.6). Jedná se o počet bodů načtených ze zobrazeného průběhu signálu na displeji osciloskopu. Malé rozlišení zajistí rychlejší obnovování grafu v aplikaci na úkor méně kvalitního a nepřesného průběhu. Vysoké rozlišení zaručí jemnost a přesnost grafu, ale způsobí pomalejší obnovování dat. Zpomalení obnovování je způsobeno velkým počtem bodů, které se musí odečíst z displeje přístroje a následně předat aplikaci.



Obr. 4.6: Aplikace LABIC, rozhraní Osciloskop - výběr rozlišení grafu.

Vzhledem ke způsobu čtení průběhu z přístroje je důležité použití tlačítka *Auto*, které slouží pro funkci *Auto Scale*. Tato funkce zajistí vhodné zobrazení průběhu na displeji přístroje a tím pádem dojde ke správnému zobrazení průběhu i v aplikaci.

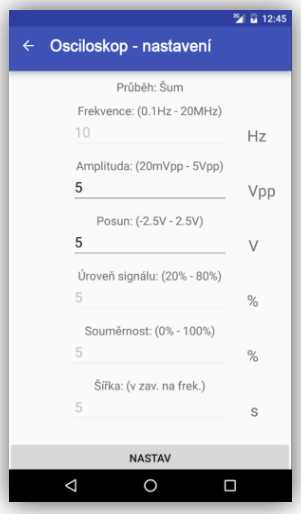
Tlačítko *Generátor* otevře nové okno se šesti vstupními poli pro zadání potřebných parametrů k nastavení požadovaného průběhu generátoru (Obr. 4.7). Tento generátor bývá integrován do většiny osciloskopů příslušné série.

Obr. 4.7: Aplikace LABIC, rozhraní Osciloskop - nastavení generátoru integrovaného v Osciloskopu.

Uživatel vybere požadovaný průběh signálu kliknutím na text *Průběh:*, čímž se otevře nabídka dostupných průběhů (Obr. 4.8). V nabídce se nachází veškeré dostupné průběhy generátoru osciloskopu, jimiž jsou: sinusový průběh, čtvercový, rampa, impuls, DC a šum.

Obr. 4.8: Aplikace LABIC, rozhraní Osciloskop - dialog pro výběr průběhu.

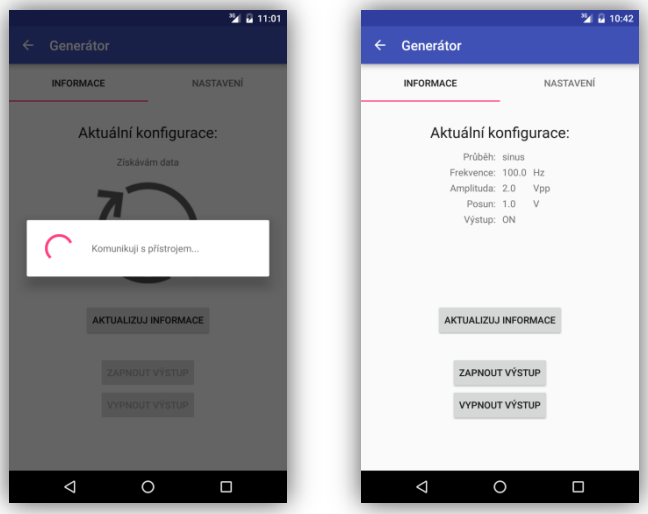
Zvolením průběhu jsou uživateli zobrazena vstupní pole pro parametry podle zvoleného průběhu (Obr. 4.9). Ostatní parametry jsou skryty, aby se předešlo omylům ze strany uživatel v podobě zadávání hodnot do polí jiného průběhu. Nad příslušným polem je vždy uveden název parametru, rozsah a na konci řádku se nachází jednotka zadávaného parametru. Textová pole jsou omezena pouze na zadávání číslic a čárek. V případě zadání hodnot mimo rozsah přístroj data příjme, parametr s chybnou hodnotou nenastaví a ponechá nastavenou původní hodnotu. Stisknutím tlačítka *Nastav*, které se nachází na konci stránky dojde k odeslání dat do přístroje a jeho nastavení.



Obr. 4.9: Aplikace LABIC, rozhraní Osciloskop - příklad omezení vstupů podle průběhu.

4.1.2. Rozhraní Generátoru

Rozhraní umožňuje konfigurovat hlavní funkce kompatibilního generátoru, například Keysight 33220A a generátorů s podobnou syntaxí příkazů [12]. Po zobrazení rozhraní je automaticky spuštěn proces načítání aktuálního natavení přístroje (Obr. 4.10 vlevo). Pokud je spojení s přístrojem navázáno a data jsou přijata aplikací, zobrazí se tato data rozřazena a popsána v informační tabulce. Uživatel má dále možnost aktualizovat tato data pomocí tlačítka *Aktualizovat informace*. (Obr. 4.10 vpravo). Pokud nebylo spojení s přístrojem navázáno, nebo bylo ztraceno, je uživateli místo tabulky s informacemi zobrazen status o problému se spojením (Obr. 4.11). Uživatel opět pomocí tlačítka *Aktualizovat informace* se může pokusit znovu navázat spojení s přístrojem a získat aktuální konfiguraci.



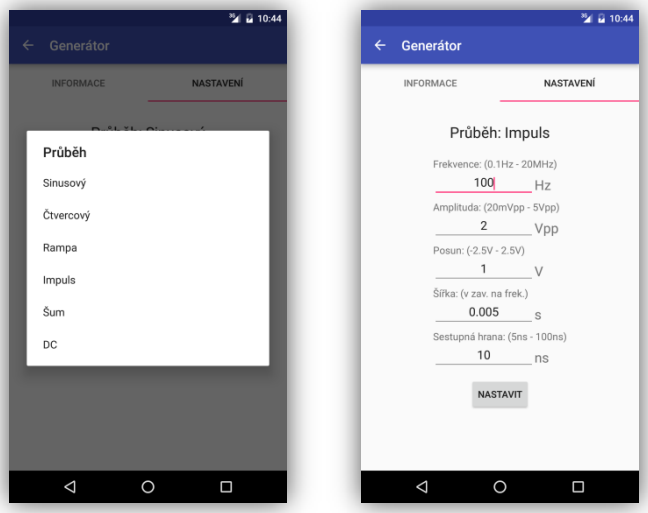
Obr. 4.10: Aplikace LABIC, rozhraní Generátor - načítání aktuálního nastavení.

Obr. 4.11: Aplikace LABIC, rozhraní Generátor - zobrazení chybového statusu po neúspěšném spojení s přístrojem.

Záložka *Nastavení* umožňuje nastavit jednotlivé parametry přístroje jako je funkce, frekvence, amplituda a další. Vzhledem k tomu, že pro každou funkci jsou povoleny rozdílné parametry, zobrazují se vždy jen ty parametry, které patří k dané funkci a ostatní nepotřebné parametry jsou skryty. Touto jednoduchou úpravou je zamezeno dezorientování uživatele a zpřehledněno uživatelské rozhraní. (Obr. 4.12)

Obr. 4.12: Aplikace LABIC, rozhraní Generátor - karta nastavení.

Kliknutím na text *Funkce* je zobrazen dialog s funkcemi generátoru (Obr. 4.13 vlevo), uživatel vybere potřebnou funkci a následně jsou zobrazeny parametry pro zadanou funkci (Obr. 4.13 vpravo). Stisknutím tlačítka *Nastavit* dojde k nastavení generátoru podle zadaných parametrů, zároveň dojde k obnovení informací o aktuální konfiguraci.



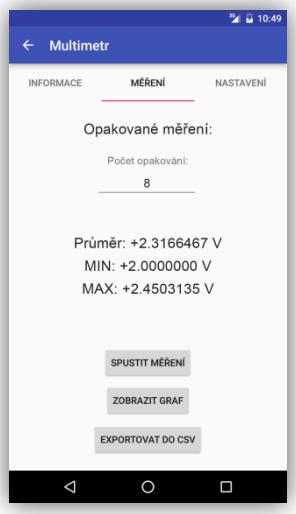
Obr. 4.13: Aplikace LABIC, rozhraní Generátor - výběr funkce a zobrazené nastavení.

4.1.3. Rozhraní Multimetru

Toto rozhraní je určené pro ovládání multimetrů řady Keysight 34410A a podobných, nebo multimetrů se stejnou syntaxí příkazů [13]. Spuštěním rozhraní je aktivován proces komunikace s přístrojem a získání informací o aktuální konfiguraci, podobně jako tomu bylo u rozhraní generátoru. Zároveň s načtením informací o konfiguraci je provedeno měření aktuální veličiny. Změřená hodnota je následně zobrazena pod tabulkou informací. (Obr. 4.14)

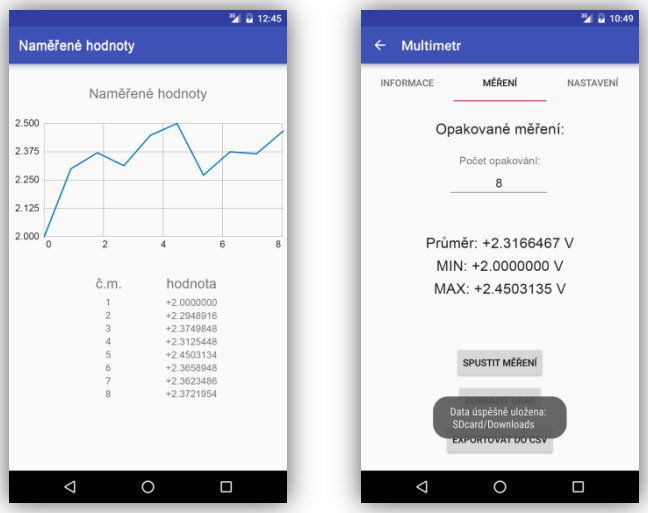
Obr. 4.14: Aplikace LABIC, rozhraní Multimetr - načtení aktuálního nastavení a změření veličiny.

Druhá záložka s názvem *Měření* slouží pro opakované změření veličiny. Uživatel zadá počet opakování, stiskne tlačítko *Změřit* a tím spustí proces měření.Tento proces odečte z přístroje požadovaný počet měření a uloží data. Z dat je následně vypočítána průměrná hodnota, maximální a minimální hodnota. Hodnoty jsou následně zobrazeny přímo na obrazovce nad tlačítky (Obr. 4.15).



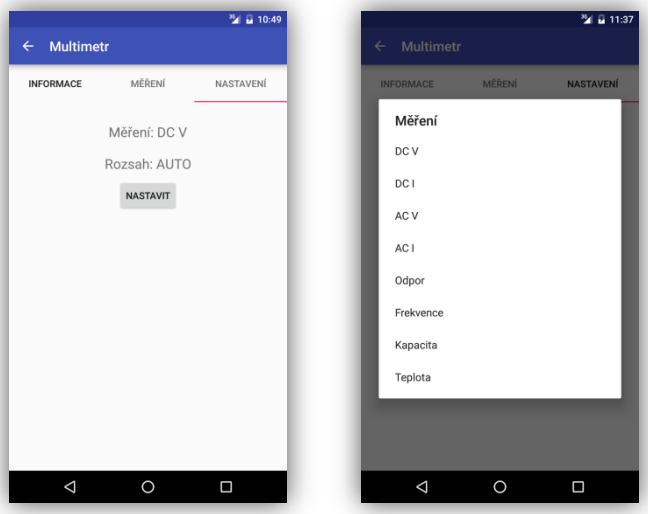
Obr. 4.15: Aplikace LABIC, rozhraní Multimetr - opakované měření veličiny.

Po ukončení měření je zpřístupněno tlačítko *Zobrazit graf* a tlačítko *Export do CSV*. Stisknutím tlačítka *Zobrazit* graf je otevřeno okno s vykresleným grafem a tabulkou hodnot sestavených z naměřených dat (Obr. 4.16 vlevo). Tlačítko *Exportovat do CSV* provede uložení dat do souboru ve formátu CSV v paměti zařízení. Uživatel je následně o dokončení tohoto procesu informován zprávou o uložení (Obr. 4.16 vpravo).



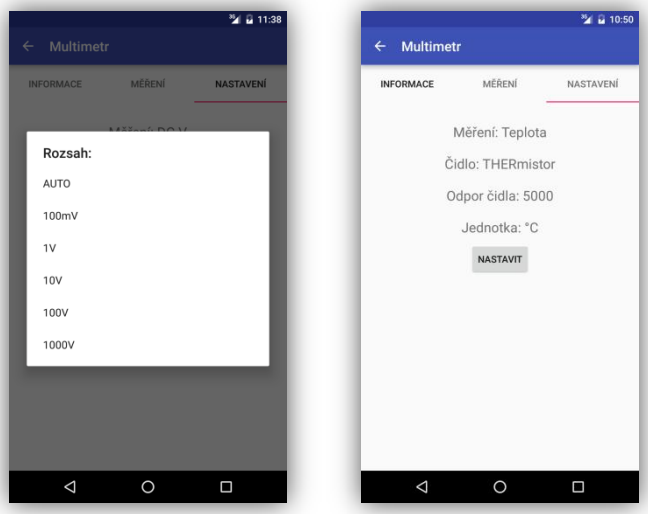
Obr. 4.16: Aplikace LABIC, rozhraní Multimetr - zobrazení naměřených hodnot a informace o uložení CSV souboru.

Ve třetí záložce jménem *Nastavení* (Obr. 4.17 vlevo) uživatel může nastavit aplikaci pro měření veličin, které příslušný multimetr podporuje. První možností je výběr z měření veličin, jako je napětí, proud, teplota, kapacita a další (Obr. 4.17 vpravo).



Obr. 4.17: Aplikace LABIC, rozhraní Multimetr - záložka nastavení a výběr měřené veličiny.

Po výběru měřené veličiny je automaticky vyvoláno další dialogové okno s výběrem rozsahu k dané veličině (Obr. 4.18 vlevo). Pouze v případě měření teploty je místo výběru rozsahu zobrazena volba čidla. Spolu s volbou čidla je uživateli umožněno vybrat odpor čidla a jednotku měření (Obr. 4.18 vpravo). Stisknutím tlačítka *Nastavit* dojde k odeslání příkazů do přístroje a nastavení podle zadaných parametrů.



Obr. 4.18: Aplikace LABIC, rozhraní Multimetr - výběr rozsahu a možnosti nastavení pro měření teploty.

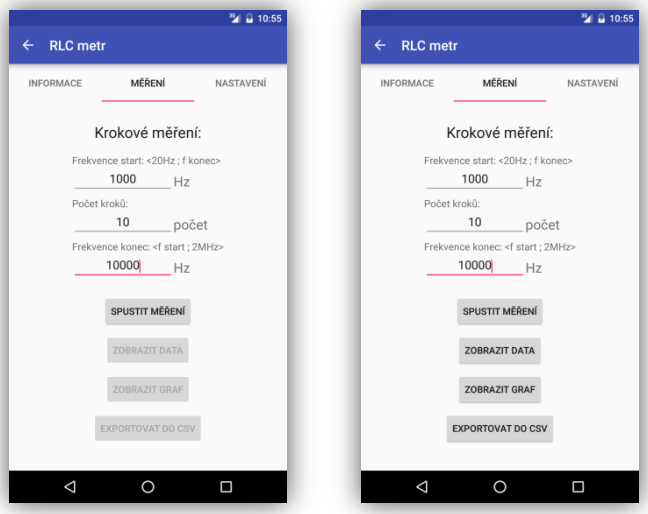
4.1.4. Rozhraní RLC metru

Pro přístroje typu RLC metru je připraveno obdobné rozhraní jako u předchozích přístrojů. Opět je možné rozhraním ovládat přístroje stejné řady jako je Keysight E4980a, nebo RLC přístroje s podobnými SCPI příkazy.[14]

Po spuštění rozhraní je opět zahájena komunikace s přístrojem za účelem zjištění aktuální konfigurace a zobrazení těchto dat do přehledné tabulky. Zároveň se provede jedno měření a naměřená hodnota je zobrazena pod tabulkou s daty. Pokud uživatel potřebuje, může provést další měření pomocí tlačítka *Aktualizovat informace*, které spustí opět celý proces získávání dat a měření. (Obr. 4.19)

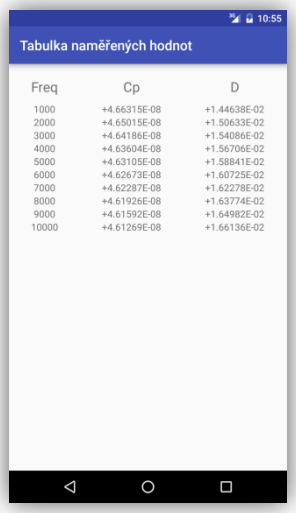
Obr. 4.19: Aplikace LABIC, rozhraní RLC metr - načtení a zobrazení aktuální konfigurace.

V následující záložce s názvem *Měření* se nachází několik prvků pro krokové měření. Uživatel má možnost zadat počáteční frekvenci, konečnou frekvenci a počet kroků mezi těmito frekvencemi (Obr. 4.20 vlevo). Po zadání požadovaných hodnot a stisknutím tlačítka *Spustit měření,* je aplikací rozdělen zadaný interval frekvence na zadaný počet dílků a následně je provedeno měření pro tyto frekvence. Po dokončení měření se objeví informace o úspěšném provedení a jsou zpřístupněna tlačítka pro zpracování dat.(Obr. 4.20 vpravo)



Obr. 4.20: Aplikace LABIC, rozhraní RLC metr - krokové měření a zpřístupnění tlačítek pro vyhodnocení dat.

První z tlačítek, jménem *Zobrazit data* převede změřená data do tabulky a tu zobrazí uživateli v novém okně. (Obr. 4.21)



Obr. 4.21: Aplikace LABIC, rozhraní RLC metr - zobrazení naměřených dat v tabulce.

Druhé tlačítko s názvem *Zobrazit graf* zpracuje naměřená data a zobrazí je ve dvou grafech v novém okně. Vzhledem k tomu, že přístroj měří dvě veličiny, jsou tedy zobrazeny závislosti obou veličin na frekvenci.(Obr. 4.22)

Obr. 4.22: Aplikace LABIC, rozhraní RLC metr - zobrazení naměřených dat v grafu.

Třetí tlačítko jménem *Exportovat do CSV* slouží, jak název napovídá pro exportování změřených dat do souboru formátu CSV, který je uložen do paměti mobilního zařízení, ze které je posléze možné soubor stáhnout do PC a zpracovat například v tabulkovém editoru MS Excel. Po stisknutí tlačítka jsou data uložena do souboru a uživatel je o tom informován informační zprávou (Obr. 4.23).

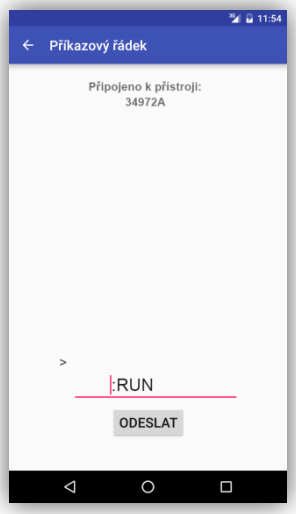
Obr. 4.23: Aplikace LABIC, rozhraní RLC metr - exportování naměřených dat do CSV a informování uživatele

V poslední záložce pojmenované *Nastavení* je uživateli umožněno konfigurovat RLC metr pomocí hlavních parametrů. Kliknutím na text *Funkce* je vyvolána nabídka všech možných měřících metod, které přístroj podporuje.Dále je možné nastavit další parametry jako je frekvence, úroveň, typ měření, délka kabelu a další (Obr. 4.24). Toto nastavení je odesláno stisknutím tlačítka *Nastavit*, které se nachází na konci stránky. Tlačítko odešle nastavení do přístroje a zároveň spustí obnovení dat na kartě *Informace*.

Obr. 4.24: Aplikace LABIC, rozhraní RLC metr - možnosti nastavení.

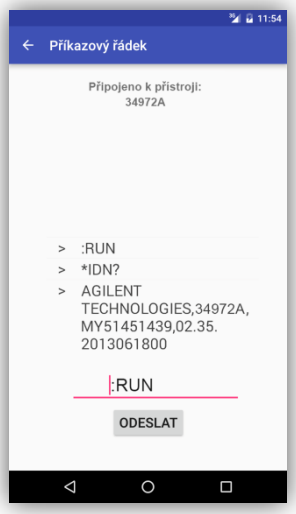
4.1.5. Rozhraní příkazového řádku

Pokud přístroj nemá v aplikaci vytvořené rozhraní a jedná se o přístroj firmy Agilent, Keysight, Keithley nebo se jedná o přístroj, který podporuje stejnou syntaxi SCPI jako tyto přístroje, aplikace jej rozpozná a přidělí přístroji univerzální rozhraní *Prompt*. Díky tomuto rozhraní je možné přístroj ovládat pomocí příkazové řádky. V seznamu na hlavní stránce aplikace je tento přístroj zobrazen s jeho názvem a ikonou příkazové řádky. Stisknutím této položky v seznamu se otevře rozhraní pro ovládání (Obr. 4.25).



Obr. 4.25: Aplikace LABIC, rozhraní Příkazový řádek - hlavní obrazovka.

Po zadání SCPI příkazu a stisknutí tlačítka odeslat dojde k předání příkazu přístroji a příkaz je následně zapsán do seznamu komunikace s přístrojem. Tento seznam je automaticky doplňován odeslanými příkazy a v případě příkazu pro získání dat jsou tato data do seznamu také doplněna (Obr. 4.26). Pokud uživatel potřebuje odeslat více příkazů najednou, stačí tyto příkazy zkopírovat do příkazové řádky a odeslat.



Obr. 4.26: Aplikace LABIC, rozhraní Příkazový řádek - odeslání příkazu a načtení dat.

5. APLIKACE LABIC - ROZBOR HLAVNÍCH ČÁSTÍ ZDROJOVÉHO KÓDU

Kompletní řešení aplikace je naprogramováno v jazyce Java za použití softwaru Android Studio s integrovaným emulátorem systému Android. Aplikace byla navržena tak, aby poskytla uživateli jednoduché a přehledné rozhraní pro ovládání přístrojů a zároveň kladla co nejmenší nároky na výkon mobilního zařízení na kterém je spuštěna. Struktura kódu byla také optimalizována k jednoduchosti a čitelnosti pro případné další rozšiřování aplikace o nová rozhraní přístrojů.

5.1.1. Použité knihovny

Do aplikace byly integrovány celkem tři knihovny. První z knihoven je nejdůležitější knihovnou pro chod aplikace a tou je AAIo [15]. Tato knihovna byla vytvořena společností Agilent pro řízení komunikace s přístroji po síti. Další dvě knihovny jsou použity pro zpracování nebo zobrazení dat. Knihovna GraphView ve verzi 4.0.1 slouží pro vytvoření grafů a knihovna OpenCSV ve verzi 3.5 je určená ke zpracování souborů CSV [16][17].

5.1.2. Odesílání nastavení do přístroje

Kód uvedený v této kapitole popisuje vytvoření spojení s přístrojem na zadané IP adrese a následné odeslání příkazu do zařízení. Této komunikace je využito ve všech rozhraních, kde je prováděna komunikace s přístrojem.

Každé vytvoření spojení s přístrojem musí být realizováno v Asynchronním procesu [18]. Tato třída *AsyncTask* zajistí vykonávání veškerého v ní zapsaného kódu na pozadí aplikace. To znamená, že je vykonáván na pozadí vlákna přiděleného hlavní třídě aplikace. Pokud by tak nebylo učiněno a kód byl vykonáván v hlavní třídě aktivity, docházelo by k „zamrzání“ aplikace během komunikace s přístrojem. Důvodem tohoto problému je čas potřebný pro provedení komunikace a případné zpoždění způsobené například vytížením sítě.

Dále je nutné při komunikaci s přístrojem využít pasivního ošetření chyb a to pomocí bloku *try-catch* [19]. Do bloku *try* je umístěna část kódu, která může vyvolat nějakou chybu. Pokud nastane v bloku *try* chyba, jeho vykonávání se přeruší a program přejde do bloku *catch*, kde by měla být chyba odchycena a tím pádem nedojde k pádu celé aplikace. Pokud vše v bloku *try* proběhne bez chyby, je vykonán celý a *catch* se přeskočí.

Zde je uveden úryvek kódu obsluhující spojení s přístrojem podle popisu uvedeného výše. Veškeré příkazy *connection* patří ke knihovně AAIo. Funkce jednotlivých částí je popsána v komentářích kódu.

```
public class AgilentGenAsync extends AsyncTask<String, Integer, String> {
    private IO connection = null;

    @Override
    protected void onPreExecute() {
        //zde umístěný kód je proveden před spuštěním hlavní
        //části doInBackground
    }

    @Override
    protected String doInBackground(String... command) {
        //kód který potřebujeme provést ve vlastním procesu
        StringBuffer response;
        //blok pro kód, který může vyvolat chybu
        try
        {
            //vytvoření instance connection třídy IO
            connection = new IO();
            //otevření spojení s přístrojem na IP adrese a portu
            connection.openWithAddress(ip_adresa, port);
            //odeslání příkazu do přístroje
            connection.print(command);
        }
        catch (AAIoException ex)
        {
            //odchycení případné vyjímky spojení s přístrojem
            return final_response;
        }
        catch (Exception ex)
        {
            //odchycení případných obecných vyjímek
            return final_response;
        }
        finally
        {
            //nakonec uzavřeno spojení
            connection.close();
        }
        return final_response;
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
        //většinou kód který který předává informace
        //o průběhu doInBackground zpět do popředí
    }

    @Override
    protected void onPostExecute(String result) {
        //kód provedený po skončení doInBackground
    }
}
```

5.1.3. Sestavení příkazu pro nastavení přístroje

Následující část kódu prezentuje funkci k sestavení řetězce příkazů pro odeslání do přístroje. Funkce je vykonána po stisknutí tlačítka *Nastavit* v rozhraní generátoru, v ostatních rozhraních je funkce obdobná [20]. Proměnná *prubeh* je nastavena zvolením průběhu signálu v dialogu *Průběh* (Obr. 4.13 vlevo). Na konci funkce je volána metoda, které je předán příkaz pro odeslání do přístroje.

```
/*akce pro tlačítko Nastavit*/
public void btnGenNastavit(View v){
    //zjištění jaký byl vybrán průběh
    switch (prubeh){
        case "SIN":
            //načtení hodnot zadaných uživatelem ze vstupních polí
            frekvence = etxtFrek.getText().toString();
            amplituda = etxtAmp.getText().toString();
            posun = etxtPos.getText().toString();

            //vytvoření příkazu, který bude odeslán přístroji
            //spojení příkazu s hodnotami do jednoho řetězce
            CMD = "FUNC " + prubeh + ";FREQ " + frekvence + ";VOLT " +
amplituda + ";VOLT:OFFS " + posun;
            break;

        case "SQU":
            frekvence = etxtFrek.getText().toString();
            amplituda = etxtAmp.getText().toString();
            posun = etxtPos.getText().toString();
            uroven = etxtUro.getText().toString();

            CMD = "FUNC " + prubeh + ";FREQ " + frekvence + ";VOLT " +
amplituda +
                    ";VOLT:OFFS " + posun + ";\nFUNC:SQU:DCYC " + uroven;
            break;

        case "RAMP":

        //< vynechaný kód >

    }
    //předání dat metodě pro odeslání
    sendData(instrumentip, CMD);
}
```

5.1.4. Příjem dat z přístroje

Proces komunikace s přístrojem při získávání dat do aplikace je velice podobný procesu odesílání nastavení uvedeného v kapitole 5.1.2. Hlavní rozdíl nastává na dvou řádcích kódu, kde místo *connection.print* , který pouze odesílá data do přístroje je použit příkaz *connection.query*, který odešle data a zpracuje odpověď z přístroje. Dále je změna v metodě *onPostExecute*, kde je nyní zpracování a předání dat zpět do popředí. Uvedené změny jsou předvedeny v úryvku kódu níže.

```
public class AgilentGenAsync extends AsyncTask<String, Integer, String>{

         //< vynechaný kód >

         //blok pro kód, který může vyvolat chybu
         try
         {
             //vytvoření instance connection třídy IO
             connection = new IO();
             //otevření spojení s přístrojem na IP adrese a portu
             connection.openWithAddress(ip_adresa, port);
             //odeslání příkazu do přístroje a načtení odpovědi
             connection.query(command, response);
         }
          catch

         //< vynechaný kód >

    @Override
    protected void onPostExecute(String result) {
         //kód provedený po skončení doInBackground
         //předání dat
         setData(result);
    }
}
```

5.1.5. Zpracování přijatých dat

Každé rozhraní přístrojů v aplikaci obsahuje metodu, která obstarává zpracování přijatých dat z přístroje. Tato metoda je u každého rozhraní řešena jiným způsobem z důvodu rozdílného zaměření přístrojů na různé veličiny. Ve výsledku mají tyto metody společný cíl a to rozdělit data, vhodně je zpracovat a nastavit nebo zobrazit na potřebných místech v aplikaci. Pro ukázku bude použita metoda z rozhraní multimetru.

```
//result = "Funkce,Rozsah,Rozlišení,Hodnota/volitelný parametr(např. jednotka)
//result = "FREQ,+5.00000000000E+03,+3.00000000000E+00,-2.50000000000E-05/C
private void setData(String result){
    //rozdělení dat podle mezery
    String data[] = result.split(" ");
    //oddělení volitelného parametru
    String data_values[] = data[1].split("/");
    //rozdělení hodnot podle čárky do pole
    String values[] = data_values[0].split(",");

    //nastavení proměnné od kterého prvku začínají data
    Integer select = 0;
    //podmínka při měření teploty, kde jsou přijaty jen dvě hodnoty
    if (data[0].equals("TEMP")){
        select = 1;
        //uložení jednotky teploty do proměnné
        zobraz_tepl_jednot = data_values[1];
    }

    //vytvoření pole pro převedená data
    String params[] = new String[5];

    //převedení všech hodnot v poli z datového typu string...
    //... na float - akceptuje matematický zápis
    for (int i = select; i < values.length; i++) {
        params[i-select] = Float.toString(Float.parseFloat(values[i]));
    }
    //nastavení textu a skrytí řádku tabulky
    txtRozsahInfo.setText("Rozsah:");
    tableRowPosun.setVisibility(View.GONE);

    //podle zjištěného typu měření jsou nastavovány prvky a zobrazena data
    switch (data[0]){
        //pro napětí
        case "VOLT":
            txtDataPrubeh.setText("VDC");
            txtDataFrek.setText(params[0]);
            txtDataAmpl.setText(params[1]);
            txtZmerenoData.setText(params[2] + " V");
            break;

        //pro proud
        case "CURR":
            txtDataPrubeh.setText("ADC");
            txtDataFrek.setText(params[0]);
            txtDataAmpl.setText(params[1]);
            txtZmerenoData.setText(params[2] + " I");
            break;


//< vynechaný kód >

..... pokračování .....
```

```
..... pokračování .....

        //pro teplotu
        case "TEMP":
            //nastavení prvků a textu
            tableRowPosun.setVisibility(View.VISIBLE);
            txtDataPrubeh.setText("měření teploty");
            txtRozsahInfo.setText("Parametr:");
            txtDataFrek.setText(params[0]);
            txtDataAmpl.setText(params[1]);
            txtDataPosun.setText(values[0]);

            //vložení stupňů před jednotku, pokud se jedná o °C, nebo °F
            if (zobraz_tepl_jednot.equals("C")||
                 zobraz_tepl_jednot.equals("F"))
             {
                txtZmerenoData.setText(params[2] +" °"+ zobraz_tepl_jednot);
            }
            //jinak pro ostatní jednotky jako je K zobrazení bez stupňů
            else {txtZmerenoData.setText(params[2] +" "+ zobraz_tepl_jednot);}
            break;
        //pokud by došlo k chybě s daty, je zobrazena informační zpráva
        default:
            Toast.makeText(getApplicationContext(), "UPS! Nastala chyba při
                nastavování dat", Toast.LENGTH_LONG).show();
    }
}
```

5.1.6. Funkce krokového měření

Tato funkce je využita v rozhraní RLC Metru a zajišťuje opakované měření s uživatelem stanoveným počtem kroků viz kapitola 4.1.4. Uživatelem je zadána počáteční frekvence, koncová frekvence a počet kroků [27]. V bloku komunikace s přístrojem je vytvořena smyčka, která provede potřebný počet nastavení a měření po jednotlivých krocích frekvence, dokud není splněna podmínka. Řetězec se změřenými daty je následně předán ke zpracování. Toto řešení bylo použito vzhledem ke skutečnosti, že přístroji není sám schopen změřit krokově hodnoty při zadání počáteční a konečné frekvence. Proto je nutné vždy po jednom kroku nastavit frekvenci a změřit hodnoty.

```
//funkce tlačítka Spustit měření
public void btnStartZmerit(View v) {
    //načtení hodnot zadaných uživatelem ze vstupních polí
    freq_start = Integer.parseInt(etxtFreqStart.getText().toString());
    freq_step = Integer.parseInt(etxtFreqStep.getText().toString());
    freq_end = Integer.parseInt(etxtFreqEnd.getText().toString());

    //splňují zadané údaje podmínku?
    if ((freq_start < freq_end) && (freq_step > 1) && (freq_step <=
         (freq_end - freq_start))) {

         //velikost jednoho kroku
        stepping = (freq_end - freq_start) / (freq_step - 1);

        //volání metody pro komunikaci s přístrojem
        sendData(instrumentip, CMD);
    }
    else {
        //zobrazení informační zprávy v případě špatného zadání hodnot
        Toast.makeText(getApplicationContext(), "Některá z hodnot není správně
zadána.", Toast.LENGTH_LONG).show();
    }
}
//async pro komunikaci s pristrojem
public class AgilentGenAsync extends AsyncTask<String, Integer, String> {

//< vynechaný kód >

     Integer loop = freq_start;
     //smyčka pro krokování frekvence od-do po n krocích
     while (loop <= freq_end){
         //sestavení příkazu pro nastavení frekvence a změření hodnoty
         command[1] = "FREQ:CW " + loop + "\nFETC:IMP:FORM?";
        //odeslání příkazu a načtení odpovědi
        connection.query(command[1], response);
        //přidání odpovědi do řetězce hodnot
        final_response += ";"+ loop +","+ response.toString();
        //přičtení kroku frekvence k aktuálně změřené frekvenci
        loop += stepping;
     }

//< vynechaný kód >

    @Override
    protected void onPostExecute(String result) {
        setData(result);
    }
}
```

5.1.7. Zpracování dat pro blok grafu

Změřená data z krokového měření je třeba vhodně zpracovat, aby mohla být vložena do grafu a správně zobrazena. Data jsou z přístroje získána v podobě textového řetězce a mohou obsahovat i několik parametrů oddělených separátorem.

Například u krokového měření v rozhraní RLC Metru, jsou data před zpracováním uložena do jednoho řetězce. V tomto řetězci se nachází několik měření pro různou frekvenci a tato měření jsou od sebe oddělena středníkem. Každé z těchto měření obsahuje 3 hodnoty, které jsou odděleny čárkou.

Dále je potřebné metodě grafu předat tyto hodnoty ve správném datovém typu *double*. Tím pádem je třeba všechna data převést z datového typu *string* na *double*. Ke splnění výše uvedených požadavkům slouží následující kód.

```
public void sortData(String table_data){
    try{
        //ověření zda nejsou data prázdná
        if (!table_data.equals("")) {

            //odstranění nepotřebného prvního středníku z rětězce
            table_data = table_data.substring(1);

            //rozdělení řetězce podle středníků na jednotlivá měření
            String data[] = table_data.split(";");

            //vytvoření dvojrozměrného pole pro rozdělení hodnot
            String data2[][] = new String[data.length][3];

            //rozdělení jednotlivých měření na hodnoty podle středníku
            for (int i = 0; i < data.length; i++) {
                data2[i] = data[i].split(",");
            }

            //inicializace pole pro data grafu
            Double freq[] = new Double[data.length];
            Double param_1[] = new Double[data.length];
            Double param_2[] = new Double[data.length];

            //rozdělení parametry do vlastních polí a převedení na double
            for (int i = 0; i < data.length; i++) {
                freq[i] = Double.valueOf(data2[i][0]);
                param_1[i] = Double.valueOf(data2[i][1]);
                param_2[i] = Double.valueOf(data2[i][2]);
            }

            //volání metody pro vytvoření grafu a předání dat
            vytvorGraf(freq,param_1,param_2);
        }

         //< vynechaný kód >
}
```

5.1.8. Zobrazení grafu

Zobrazení grafu využívá rozhraní RLC Metr a Multimetr. Naměřená data z krokového měření jsou zpracována viz kapitola 5.1.7 a zobrazena v příslušných závislostech v grafu [16].

```
private void vytvorGraf(final Double freq[], Double param_1[], Double
param_2[]){
    //inicializace pole pro objekty bodů pro graf
    DataPoint[] datas = new DataPoint[freq.length];
    DataPoint[] datas_2 = new DataPoint[freq.length];

    //rozložení dat x,y do objektů a pak do polí
    //datas - pole pro první parametr, datas_2 - pole pro druhý
    for (int i=0; i<freq.length; i++){
        datas[i] = new DataPoint(freq[i], param_1[i]);
        datas_2[i] = new DataPoint(freq[i], param_2[i]);
    }

    /**graf 1**/
    GraphView graph = (GraphView) findViewById(R.id.graph);
    LineGraphSeries<DataPoint> series = new LineGraphSeries<DataPoint>(datas);

    //nastavení max. a min. hodnoty osy X podle dat
    graph.getViewport().setMinX(freq[0]);
    graph.getViewport().setMaxX(freq[freq.length-1]);
    //povolení ručního nastavení osyX
    graph.getViewport().setXAxisBoundsManual(true);

    //prohledání pole a získání max. a min. hodnoty na osu Y
    yMin = param_1[0];
    yMax = param_1[0];
    for (int i=1; i<param_1.length; i++){
        if (yMax < param_1[i]){
            yMax = param_1[i];
        }
        if (yMin > param_1[i]){
            yMin = param_1[i];
        }
    }

    //nastavení max. a min. hodnoty osy Y podle nalezených dat
    graph.getViewport().setMinY(yMin);
    graph.getViewport().setMaxY(yMax);
    //povolení ručního nastavení osy Y
    graph.getViewport().setYAxisBoundsManual(true);

    //upravení formátu zobrazení čísel pro osy
    NumberFormat nf = NumberFormat.getInstance();
    //počet čísel ZA desetinnou čárkou
    nf.setMinimumFractionDigits(3);
    //počet čísel PŘED desetinnou čárkou
    nf.setMinimumIntegerDigits(2);

    //nastavení formátu čísel pro osy
    //nastavení jen pro osu y - proto (null, nf) jako (x,y)
    graph.getGridLabelRenderer().setLabelFormatter(new
DefaultLabelFormatter(null, nf));

    //vytvoření grafu z dat
    graph.addSeries(series);
//< vynechaný kód pro graf 2 >
}
```

5.1.9. Zpracování dat pro tabulku a její zobrazení

Obdobným způsobem jakým byla zpracována data pro graf v kapitole 5.1.7, jsou zpracována data i pro tabulku [21]. Zobrazení tabulky je oproti vykreslení grafu podstatně jednodušší a proto z komentářů v kódu jednoduše vyplývá logika. Data pro tabulku jsou přijímána v totožném formátu jako pro graf.

```
public void sortData(String table_data){
    try{
        //ověření zda nejsou data prázdná
        if (!table_data.equals("")) {
            //odstranění nepotřebného prvního středníku z řetězce
            table_data = table_data.substring(1);

            //rozdělení řetězce podle středníků na jednotlivá měření
            String data[] = table_data.split(";");

            //vytvoření dvojrozměrného pole pro rozdělení hodnot
            String data2[][] = new String[data.length][3];

            //rozdělení jednotlivých měření na hodnoty podle středníku
            for (int i = 0; i < data.length; i++) {
                data2[i] = data[i].split(",");
            }

            //vytvoření tabulky
            TableLayout data_table = (TableLayout)
findViewById(R.id.tableMeas);
            //nastavení tabulky
            data_table.setStretchAllColumns(true);
            data_table.bringToFront();

            //smyčka pro vkládání dat do řádků pod sebe
            for (int i = 0; i < data.length; i++) {
                //vytvoření řádku
                TableRow tr = new TableRow(this);

                //vytvoření sloupečku
                TextView c1 = new TextView(this);
                //vycentrování textu ve sloupečku
                c1.setGravity(Gravity.CENTER_HORIZONTAL);
                //vložení hodnoty do buňky rádku prvního sloupečku
                c1.setText(data2[i][0]);

                //stejné jako u sloupečku jedna
                TextView c2 = new TextView(this);
                c2.setGravity(Gravity.CENTER_HORIZONTAL);
                c2.setText(data2[i][1]);

                TextView c3 = new TextView(this);
                c3.setGravity(Gravity.CENTER_HORIZONTAL);
                c3.setText(data2[i][2]);

                //vložení sloupečků do tabulky
                tr.addView(c1);
                tr.addView(c2);
                tr.addView(c3);
                //vložení řádku do tabulky
                data_table.addView(tr);
            }
        }
//< vynechaný kód >
}
```

5.1.10. Statusy - zobrazování obsahu podle situace v aplikaci a vytváření informačních zpráv pro uživatele

Ke zlepšení přehlednosti zdrojového kódu byla vytvořena metoda která má na starosti zobrazování, skrývání prvků a informování uživatele o situaci v aplikaci. Tato metoda je nejčastěji využita při komunikaci aplikace s měřícím přístrojem a zobrazuje například situaci, kdy se nelze připojit k přístroji (Obr. 4.11), nebo kdy byla data úspěšně načtena a zobrazena (Obr. 4.10 vpravo). K použití této metody ji stačí kdekoliv ve zdrojovém kódu zavolat s předáním čísla statusu, který se má aktivovat a případně i zprávou, která se má zobrazit [22].

```
//volání metody kdekoliv dle potřeby pro vytvoření statusu:

//< vynechaný kód >

vytvorStatus(1, "- přístroj není dostupný")

//< vynechaný kód >

private void vytvorStatus(Integer status, String message){
    switch (status){
        //status pro neznámou kritickou chybu
        case 0:{
            //vyvolá informační bublinu se zadaným textem
            Toast.makeText(getApplicationContext(), "UPS! Nastala kritická
chyba " + message, Toast.LENGTH_LONG).show();

            //skryje tabulku s informacemi
            tableInfo.setVisibility(View.GONE);

            //zobrazí layout pro status
            layoutStatus.setVisibility(View.VISIBLE);

            //nastaví zadaný text pro layout statusu
            txtStatus.setText("Spojení s přístrojem nebylo navázáno");

            //nastaví ikonu pro daný status
            imgBtnStatus.setImageResource(R.drawable.reload_sad_smile);

            //deaktivuje tlačítko pro zapnutí, vypnutí výstupu a nastavení
            btnVystupON.setEnabled(false);
            btnVystupOFF.setEnabled(false);
            btnGenNastavit.setEnabled(false);
            break;
        }
        //status pro chybu spojení s přístrojem
        case 1:{
            Toast.makeText(getApplicationContext(), "UPS! Nastala chyba
spojení " + message, Toast.LENGTH_LONG).show();
            tableInfo.setVisibility(View.GONE);
            layoutStatus.setVisibility(View.VISIBLE);
            txtStatus.setText("Spojení s přístrojem nebylo navázáno");
            imgBtnStatus.setImageResource(R.drawable.reload_sad_smile);
            btnVystupON.setEnabled(false);
            btnVystupOFF.setEnabled(false);
            btnGenNastavit.setEnabled(false);
            break;
        }
//< vynechaný kód >
}
```

5.1.11. Uložení dat do CSV

Pro práci se soubory CSV byla použita knihovna OpenCSV, která zajistí čtení, nebo zápis naměřených dat z krokového měření RLC Metru a Multimetru do souboru [17]. V metodě pro tlačítko *Exportovat do CSV* jsou naměřená data zpracována a zapsána do souboru *rlc_mereni.csv*, který je uložen ve složce *Downloads* na SD kartě [23].

```
public void btnCsvExport(View v) {
    //práce se souborem musí být prováděna v bloku try-catch
    try {
        // jsou data prázdná? jde zapisovat do uložiště?
        if (!meas_data.equals("") && isExternalStorageWritable()) {

            //odstranění prvního středníku z řetězce
            String data = meas_data.substring(1);
            //rozdělení řetězce podle středníku do pole
            String data_arr[] = data.split(";");

            //vytvoření cesty pro soubor
            File file = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNL
OADS), "rlc_mereni.csv");
            //vytvoření souboru
            CSVWriter writer = new CSVWriter(new FileWriter(file));

            //cyklus pro zapsání dat do souboru
            for (int i=0; i<data_arr.length; i++) {
                String[] entries = data_arr[i].split(",");
                writer.writeNext(entries);
            }
            //ukončení práce se souborem
            writer.close();
            //informační zpráva o úspěšném dokončení
            Toast.makeText(getApplicationContext(), "Data úspěšně uložena:\n
SDcard/Downloads", Toast.LENGTH_LONG).show();
        }
        //pokud není splněna podmínka, zobrazí informační hlášku
        else Toast.makeText(getApplicationContext(), "UPS! chyba při
zpracování dat", Toast.LENGTH_LONG).show();
    }
//< vynechaný kód >
}
```

Při práci se souborem je nutné ověřit, zda lze zapisovat do uložiště zařízení. To je ošetřeno následující metodou, která je volána při práci se souborem a má logickou návratovou hodnotu. Pokud lze zapisovat je vrácena pravda, pokud nelze nepravda.

```
public boolean isExternalStorageWritable() {
    //zjištění statusu uložiště
    String status = Environment.getExternalStorageState();
    //pokud souhlasí status vrať pravda
    if (Environment.MEDIA_MOUNTED.equals(status)) {
        return true;
    }
    //jinak zobraz informační zprávu a vrať nepravda
Toast.makeText(getApplicationContext(), "UPS! nelze zapisovat do uložiště",
Toast.LENGTH_LONG).show();
    return false;
}
```

5.1.12. Zobrazování měřených veličin

Vzhledem ke skutečnosti, že většina měřících přístrojů je schopna měřit více veličin, je nutné zobrazovat v aplikaci správné veličiny u změřené hodnoty [26]. O tuto záležitost se například pro rozhraní RLC Metru stará níže uvedená metoda, volána z metody *setData*, která zpracovává data. Metodě je předán název (zkratka) momentálního měření veličiny. Tato zkratka je předána v podobě textového řetězce a nabývá velikosti od 2 znaků až po 4 znaky v závislosti na měřené veličině. Navíc tato zkratka v sobě nese vždy označení dvou veličin, například *CP* a *D* nebo *Ls* a *Rs*.

```
private void rozlozVeliciny (String data) {
    //vytvoření pole pro veličiny
    String zmereno_lab[] = new String[2];
    //ošetření, zda je přijata opravdu zkratka veličin
    if (!data.equals("") && (data.length() >= 2) && (data.length() <= 4)) {
        //pokud se jedná o jednu ze čtyř specifických zkratek
        if (data.equals("ZTD") || data.equals("ZTR") || data.equals("YTD") ||
data.equals("YTR")) {
            //první veličina je první znak
            zmereno_lab[0] = data.substring(0, 1);
            //druhá veličina je třetí znak (znak "T" je označení pro tangens)
            zmereno_lab[1] = data.substring(2);
            //před druhou veličinu vložím symbol tangensu
            zmereno_lab[1] = "θ" + zmereno_lab[1];
        //pokud má zkratka dva znaky
        } else if (data.length() == 2) {
            //první veličina je první znak
            zmereno_lab[0] = data.substring(0, 1);
            //druhá veličina je druhý znak
            zmereno_lab[1] = data.substring(1);
        //pro ostatní počet znaků ve zkratce
        } else {
            //převedení na malé písmo
            data = data.toLowerCase();
            //první veličina jsou první dva znaky
            zmereno_lab[0] = data.substring(0, 2);
            //druhá veličina jsou zbývající znaky
            zmereno_lab[1] = data.substring(2);

            //první znak veličiny musí být vždy velkým písmem
            zmereno_lab[0] = zmereno_lab[0].substring(0, 1).toUpperCase() +
zmereno_lab[0].substring(1);
            zmereno_lab[1] = zmereno_lab[1].substring(0, 1).toUpperCase() +
zmereno_lab[1].substring(1);
        }
        //zobrazení veličin
        txtZmerenoLabel1.setText(zmereno_lab[0]);
        txtZmerenoLabel2.setText(zmereno_lab[1]);
    //v případě, že nastane problém se zpracováním veličin
    } else {
        //zobrazení chybové hlášky
        Toast.makeText(getApplicationContext(), "UPS! Nastal poroblém se
zobrazením veličin", Toast.LENGTH_LONG).show();
        txtZmerenoLabel1.setText("??");
        txtZmerenoLabel2.setText("??");
    }
}
```

5.1.13. Zobrazování seznamu v dialogu pro výběr funkce

V následujícím úryvku kódu je předvedeno naplnění dialogu pro zobrazení seznamu funkcí RLC metru. Uživateli je po stisknutí na text funkce zobrazeno dialogové okno se seznamem funkcí, které je RLC metr schopen měřit a uživatel vybere jednu z nich [24]. Následně je podle výběru uživatele zobrazen příslušný název funkce v kartě nastavení a do proměnné *dialog_func* je uložen příkaz s názvem funkce ve formátu SCPI, který je poté v metodě pro odeslání dat předán přístroji. Například pro měření *Cp-D* je uložen do proměnné příkaz *CPD*.

```
//zobrazení dialogu pro výběr funkce RLC metru
public void txtFunc(View view) {
    //vytvoření a naplnění pole těxtovými řetězci pro zobrazení seznamu
    final String[] func_array = {
            "Cp - D", "Cp - Q", "Cp - G", "Cp - Rp", "Cs - D", "Cs - Q",
            "Cs - Rs", "Lp - D", "Lp - Q", "Lp - G", "Lp - Rp", "Ls - D",
            "Ls - Q", "Ls - Rs", "R - X", "Z - θD", "Z - θR", "G - B",
            "Y - θD", "Y - θR"};
    //vytvoření nového dialogu
    AlertDialog.Builder builderFunc = new
AlertDialog.Builder(AgilentRLCActivity.this);
    //nastavení titulky dialogu
    builderFunc.setTitle("Funkce");
    //vložení seznamu funkcí do dialogu a naslouchání co bylo vybráno
    builderFunc.setItems(func_array, new DialogInterface.OnClickListener() {
        //metoda pro indexaci vybrané položky seznamu
        public void onClick(DialogInterface dialog, int which) {
            //vytvoření a naplnění pole s příkazy ve stejném pořadí..
            //..jako pole pro seznam funkcí
            String[] data_func_array = {
                    "CPD", "CPQ", "CPG", "CPRP", "CSD", "CSQ", "CSRS",
                    "LPD", "LPQ", "LPG", "LPRP", "CSD", "LSD", "LSQ",
                    "LSRS", "RX", "ZTD", "ZTR", "GB", "YTD", "YTR"
            };
            //zobrazení vybrané funkce v textu po zavření dialogu
            txtFunc.setText("Funkce: " + func_array[which]);
            //uložení příkazu podle vybrané funkce do proměnné
            dialog_func = data_func_array[which];
        }
    //zobrazení dialogu
    }).show();
}
```

6. ZÁVĚR

Z dostupné literatury byl vytvořen přehled rozhraní pro komunikaci s měřícími přístroji a jejich vzdálené ovládání., Tato rozhraní můžeme v dnešní době nalézt na většině měřících přístrojů. V laboratorních podmínkách je nejvíce rozšířené rozhraní LAN, USB a také starší rozhraní GPIB. Připojení měřících přístrojů je velice jednoduché, hlavně díky standardizaci komunikačních rozhraní. Také jejich vzdálené ovládání je velice jednoduché a to hlavně díky webovému rozhraní nebo možnosti zadávání příkazů SCPI přes příkazovou řádku.

Úkolem práce bylo seznámení se s vývojovým prostředím pro mobilní zařízení s operačním systémem Android a osvojení programovacího jazyka pro tuto platformu. Toho bylo docíleno pomocí příruček a průvodců společnosti Google. Bylo nainstalováno, nastaveno a prostudováno vývojové prostředí Android Studio. V tomto prostředí byla kompletně vyvíjena a odladěna celá aplikace o které pojednává tato diplomová práce. Jako programovací jazyk byla zvolena Java a pro její studium byla využita internetová literatura v podobě Google Android Reference, různých programátorských fór, blogů a stránek.

Dle zadání měla být vytvořena aplikace, která by v sobě sdružovala několik rozhraní pro ovládání různých laboratorních přístrojů například osciloskopu, generátoru, RLC metru a podobně, zároveň byla funkční a uživatelsky přívětivá. Tohoto cíle se podařilo dosáhnout a vznikla aplikace LABIC, která nabízí dynamické prostředí pro připojení několika přístrojů s předdefinovaným rozhraním pro jejich ovládání. Aplikace umožňuje ovládání přístroje osciloskop, zobrazení průběhů signálů na obou kanálech a nastavení generátoru signálu integrovaného v přístroji. Dále aplikace podporuje přístroj multimetr, se kterým je schopna měřit veškeré veličiny podporované přístrojem a nastavení jeho vlastností. Do aplikace byl také zařazen generátor a většina jeho nastavení, dále RLC metr s kompletním pokrytím měřících funkcí a krokovým měřením.Veškeré tyto uvedené přístroje je aplikace schopna obsloužit za předpokladu, že přístroje používají stejnou syntaxi SCPI příkazů pro jaké byla aplikace naprogramována. Dále aplikace podporuje připojení jakéhokoliv jiného přístroje a umožňuje jeho řízení pomocí příkazové řádky s podporou SCPI.

SEZNAM POUŽITÝCH ZDROJŮ

[1] UJBÁNYAI, Miroslav. *Programujeme pro Android*. Vyd. 1. Praha: Grada, 2012, 187 s. Průvodce (Grada). ISBN 978-80-247-3995-3.

[2] ŠPONAR, Radek. Vlastnosti a užití průmyslových sběrnic. *Elektrorevue* [online]. 2004, 2004(19) [cit. 2016-03-25]. Dostupné z: http://www.elektrorevue.cz/clanky/04019/index.html#kap2

[3] *Agilent Technologies USB/LAN/GPIB Interfaces: Connectivity Guide*. 2004. Dostupné také z: http://literature.cdn.keysight.com/litweb/pdf/5188-5724.pdf

[4] *Agilent I/O Connectivity: for easy PC-to-instrument connection*. 2012. Dostupné také z: http://www.testequity.com/documents/pdf/I-O_connectivity.pdf

[5] FEDRA, Zbyněk. Automatizované měření nf obvodu pomocí P1DD a sběrnice GPIB. *Elektrorevue* [online]. 2004, 2004(59) [cit. 2016-04-02]. Dostupné z: http://www.elektrorevue.cz/clanky/04059/index.html

[6] Short Tutorial on VXI. *National Instruments* [online]. [cit. 2016-04-20]. Dostupné z: http://www.ni.com/white-paper/2899/en/

[7] *Introduction to VME / VXI / VXS Standards*. 2012. Dostupné také z: http://file.wiener-d.com/documentation/General/WIENER_VME_VXI_VXS_introduction_1.0.pdf

[8] History of LXI. *LXI Consortium* [online]. [cit. 2016-04-20]. Dostupné z: http://www.lxistandard.org/About/History.aspx

[9] Frequently Asked Questions. *LXI Consortium* [online]. [cit. 2016-04-20]. Dostupné z: http://www.lxistandard.org/About/QAndA.aspx#1

[10] Ústav řízení systémů a spolehlivosti. *Programování GPIB* [online]. 2013 [cit. 2016-04-25]. Dostupné z: http://www.rss.tul.cz/ftppub/cms/04_%20GPIB_%20SW.pdf

[11] DSOX2012A Oscilloscope: 100 MHz, 2 Channels. KEYSIGHT. *Agilent Technologies* [online]. 2015 [cit. 2016-05-04]. Dostupné z: http://www.keysight.com/main/techSupport.jspx?cc=CZ&lc=eng&pid=x201831

[12] 33521A Function / Arbitrary Waveform Generator, 30 MHz. KEYSIGHT. *Keysight technologies* [online]. 2015 [cit. 2016-05-04]. Dostupné z: http://www.keysight.com/main/techSupport.jspx?cc=CZ&lc=eng&pid=1871159

[13] 34410A Digital Miltimeter. KEYSIGHT. *Agilent Technologies* [online]. 2015 [cit. 2016-05-04]. Dostupné z: http://www.keysight.com/en/pd-692834-pn-34410A/digital-multimeter-6-digit-high-performance?cc=CZ&lc=eng

[14] E4980A Precision LCR Meter: 20 Hz to 2 MHz. KEYSIGHT. *Keysight technologies* [online]. 2016 [cit. 2016-05-04]. Dostupné z: http://www.keysight.com/main/techSupport.jspx?cc=CZ&lc=eng&nid=-34124.536908436&pid=715495&pageMode=PL

[15] Smart Device Programming Tools and Examples for Instrument Control. *Keysight Technologies* [online]. 2016 [cit. 2016-05-01]. Dostupné z: http://www.keysight.com/main/software.jspx?cc=CZ&lc=eng&ckey=2075645&nid=-35714.384321.02&id=2075645&cmpid=zzfindsmartdevice

[16] GraphView - open source graph plotting library for Android. *Graph View* [online]. 2016 [cit. 2016-05-01]. Dostupné z: http://www.android-graphview.org/download--getting-started.html

[17] OpenCSV. *opencsv* [online]. 2016 [cit. 2016-05-06]. Dostupné z: http://opencsv.sourceforge.net/index.html

[18] AsyncTask. *Android Developers* [online]. [cit. 2016-05-06]. Dostupné z: https://developer.android.com/reference/android/os/AsyncTask.html

[19] How to avoid Force Close Error in Android. *Hardik Trivedi* [online]. [cit. 2016-05-06]. Dostupné z: https://trivedihardik.wordpress.com/2011/08/20/how-to-avoid-force-close-error-in-android/

[20] Button. *Android Developers* [online]. [cit. 2016-05-06]. Dostupné z: https://developer.android.com/reference/android/widget/Button.html

[21] TableLayout. *Android Developers* [online]. [cit. 2016-05-10]. Dostupné z: https://developer.android.com/reference/android/widget/TableLayout.html

[22] Toast. *Android Developers* [online]. [cit. 2016-05-10]. Dostupné z: https://developer.android.com/reference/android/widget/Toast.html

[23] File. *Android Developers* [online]. [cit. 2016-05-10]. Dostupné z: https://developer.android.com/reference/java/io/File.html

[24] AlertDialog. *Android Developers* [online]. [cit. 2016-05-10]. Dostupné z: https://developer.android.com/reference/android/app/AlertDialog.html

[25] DialogInterface.OnClickListener. *Android Developers* [online]. [cit. 2016-05-10]. Dostupné z: https://developer.android.com/reference/android/content/DialogInterface.OnClickListener.html

[26] TextView. *Android Developers* [online]. 2016 [cit. 2016-05-13]. Dostupné z: http://developer.android.com/reference/android/widget/TextView.html

[27] EditText. *Android Developers* [online]. 2016 [cit. 2016-05-13]. Dostupné z: http://developer.android.com/reference/android/widget/EditText.html

[28] ORACLE. *Java SE 7: Features and Enhancements* [online]. 2013 [cit. 2016-04-10]. Dostupné z: http://www.oracle.com/technetwork/java/javase/jdk7-relnotes-418459.html

[29] GOOGLE. *Android Studio* [online]. 2016 [cit. 2016-03-15]. Dostupné z: https://developer.android.com/studio/index.html

[30] Android 6.0 Marshmallow. *Android* [online]. 2016 [cit. 2016-05-03]. Dostupné z: https://www.android.com/versions/marshmallow-6-0/

[31] MACEK, M. *Využití mobilní platformy Android k ovládání přístrojové techniky*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 57 s. Vedoucí bakalářské práce Ing. Martin Frk, Ph.D., FEKT VUT v Brně

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

API Rozhraní pro programování aplikací - Application Programming Interface

PDA Mobilní zařízení - Personal Digital Assistant

LAN Lokální počítačová síť - Local Area Network

RS-232 Sériový port, komunikační rozhraní

GPIB Rozhraní pro měřící přístroje IEEE.488 - General Purpose Interface Bus

SCPI Standardizovaná syntaxe příkazů pro měřící přístroje - Standard Commands for Programmable Instruments

RJ-45 Typ zapojení síťového kabelu

DHCP Automatická konfigurace zařízení v síti - Dynamic Host Configuration Protocol

PCI Rozšiřující sběrnice na základní desce počítače - Peripheral Component Interconnect

MXI2 Systémová sběrnice - Multisystem eXtension Interface

FireWire Standardní sériová sběrnice pro připojení periferií k počítači

IPv4 Čtvrtá verze internet protokolu - Internet Protocol version 4

CSV Souborový formát pro výměnu dat - Comma-Separated Values

SD Formát paměťové karty s pamětí typu flash - Secure Digital

SEZNAM PŘÍLOH

A VÝVOJOVÉ PROSTŘEDÍ 57

A.1 Android Studio 57

A.2 Vytvoření nové aplikace v Android Studio 59

A.3 Obsah DVD 63

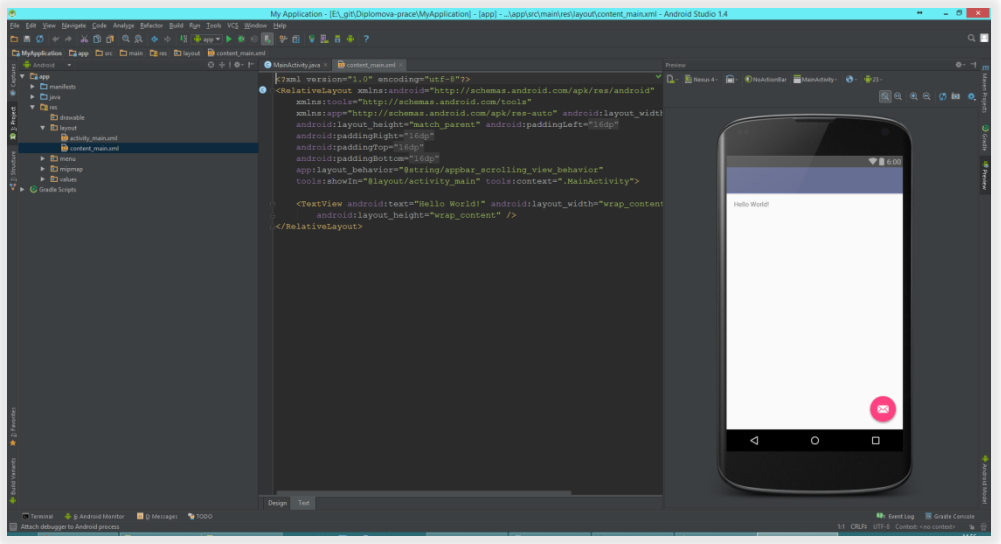
A VÝVOJOVÉ PROSTŘEDÍ

Pro vyvíjení aplikací v jazyce Java je k dispozici velké množství nástrojů a prostředí. Každý programátor má tedy možnost si vybrat software podle svého uvážení a požadavků. Jeden z nejrozšířenějších programů pro vývoj aplikací v jazyce Java se jmenuje Eclipse a přináší opravdu bohaté prostředí funkcí, které programátorovi zjednodušují vývoj. Od tohoto programu se odvíjí spousta jiných vývojových prostředí.

A.1 Android Studio

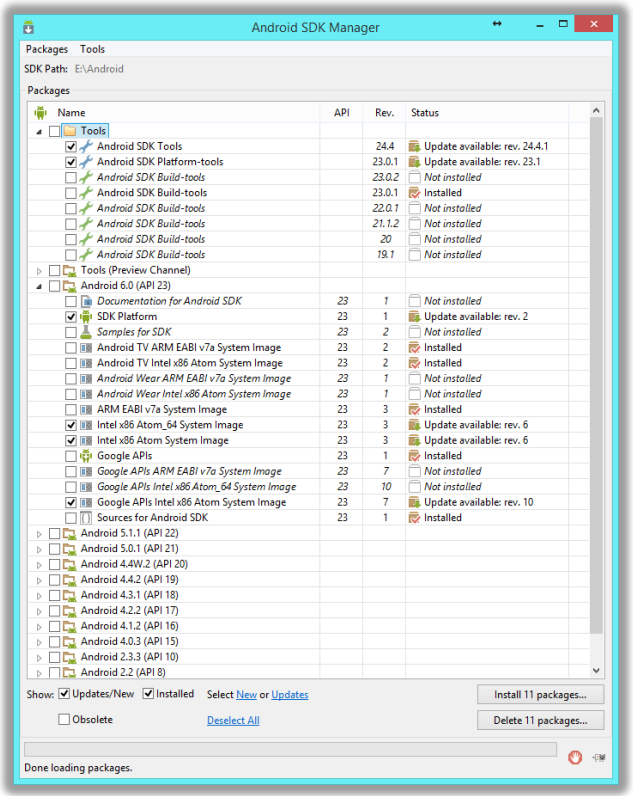
Společnost Google vydala software jménem Android Studio, který vychází z již zmíněného programu Eclipse (Obr. A.1). Android Studio je kompletní balíček pro tvorbu Android aplikací, kde nejsou potřebné žádné dodatečné a podpůrné programy. Pokud vývojář vyžaduje některou z nestandardních funkcí jednoduše si ji stáhne přes rozhraní přímo v programu, kde je funkce následně patřičně zavedena nebo nainstalována. Android studio má integrovaný nástroj pro emulaci zařízení s operačním systémem Android. Díky této emulaci má programátor možnost testovat a ladit svoji aplikaci přímo při programování, což značně ulehčuje práci a zároveň umožňuje vyvíjet aplikace, i když programátor nevlastní Android zařízení.

Aktuální verze programu Android Studio je ke stažení na adrese: http://developer.android.com/sdk/index.html



Obr. A.1: Prostředí programu Android Studio.

Velice důležitou součástí Android Studia je SDK Manager, ve kterém je možné vybrat a stáhnout balíček s příslušnou verzí Android, který bude použit při vývoji aplikace. Díky tomu je také možné vytvořenou aplikaci otestovat na různých verzích operačního systému Android. Kromě stažitelných balíčků verzí jsou zde k dispozici různé nástroje, které je možné využít při vývoji aplikace. (Obr. A.2)



Obr. A.2: SDK Manager - výběrem platforem a nástrojů.

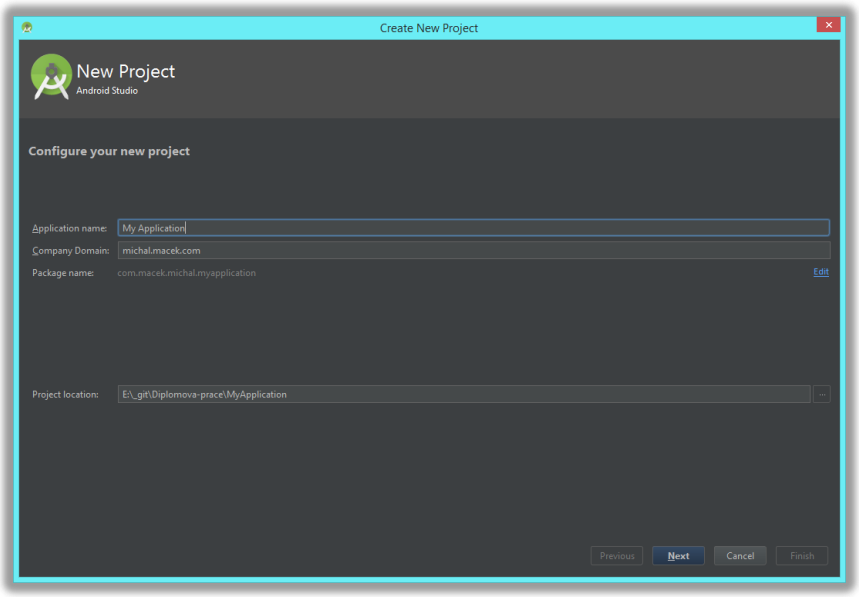
Další neméně důležitou funkcí Android Studia je emulátor Android zařízení. Díky této funkci je možné nastavit a emulovat v podstatě jakékoliv zařízení s operačním systémem Android, na kterém je možné následně otestovat a postupně odlaďovat vyvíjenou aplikaci. Je možné u zařízení nastavit rozlišení obrazovky, verzi operačního systému, velikost paměti RAM a spoustu dalších parametrů, které zaručí co nejpřesnější simulaci. (Obr. A.3).

Obr. A.3: SDK Manager - odsouhlasení licenčních podmínek.

A.2 Vytvoření nové aplikace v Android Studio

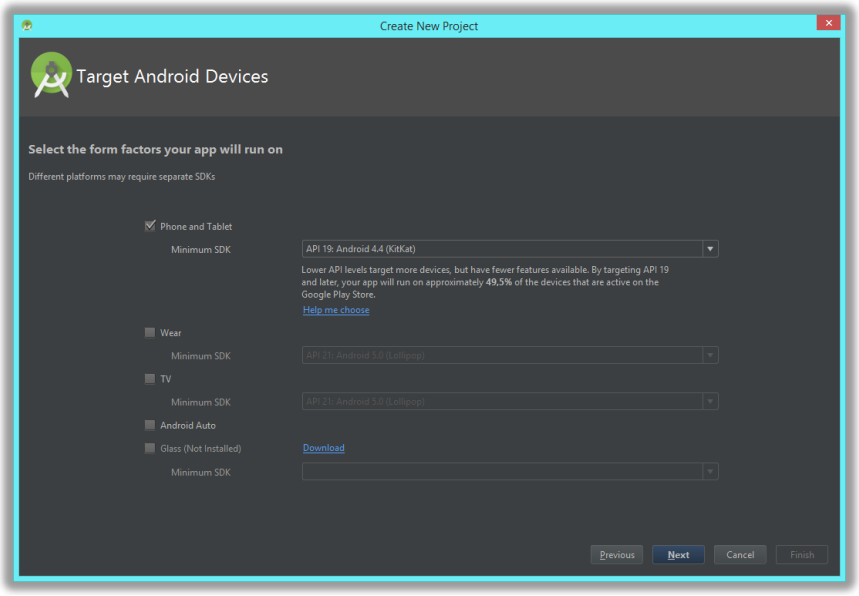
Pro vytvoření nové aplikace stačí stisknout tlačítko *New Project*, které se nachází na uvítací kartě, nebo otevřít menu *File* a zvolit *New Project*. Tím se otevře okno průvodce tvorby aplikace a základního nastavení (Obr. A.4)

Do pole *Application Name* uživatel vloží název aplikace a do pole *Company Domain* doménu projektu. Následně v poli *Project location* vybere cestu kam se má nový projekt uložit.



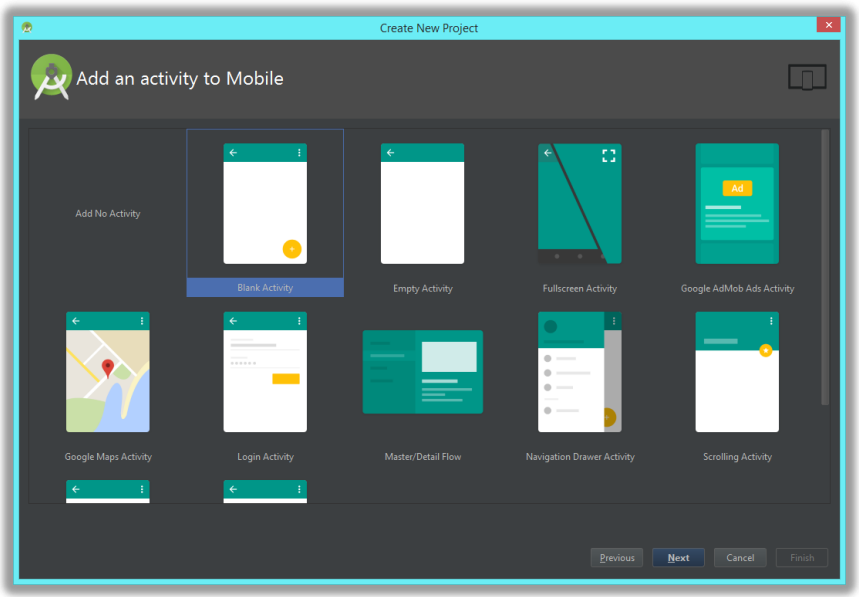
Obr. A.4: Android studio - pojmenování a lokace aplikace.

V dalším okně je uživatel vyzván k výběru pro jakou platformu bude chtít aplikaci vyvíjet. Na výběr je telefon, tablet, hodinky, televize, automobil a nebo Google glass. Po výběru platformy je třeba zvolit, pro jakou nejnižší verzi operačního systému android bude aplikace dostupná (Obr. A.5).



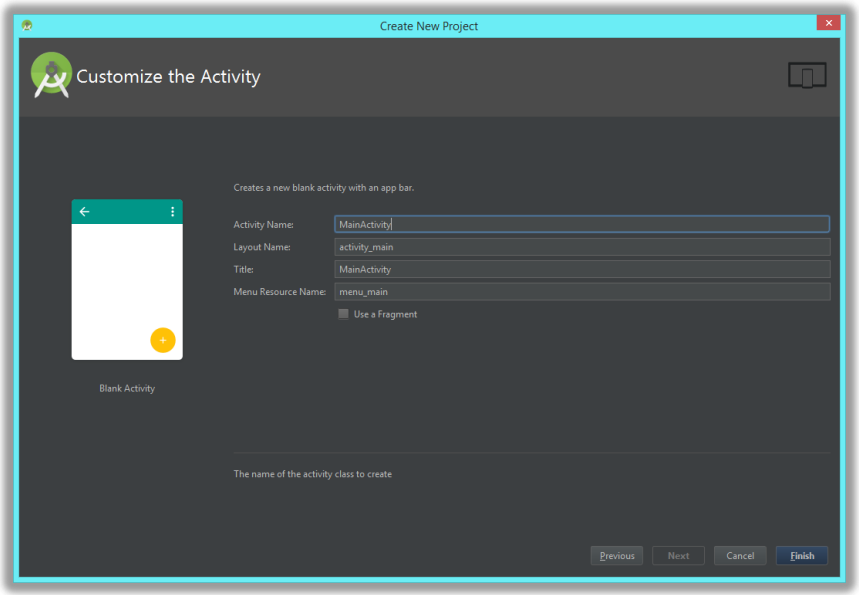
Obr. A.5: Android studio - výběr platformy a minimální verze systému.

Po nastavení platformy a verze následuje volba počáteční aktivity, ze které se bude projekt odvíjet (Obr. A.6). Pro začátek je vhodné zvolit prázdnou aktivitu *Blanc Activity*.



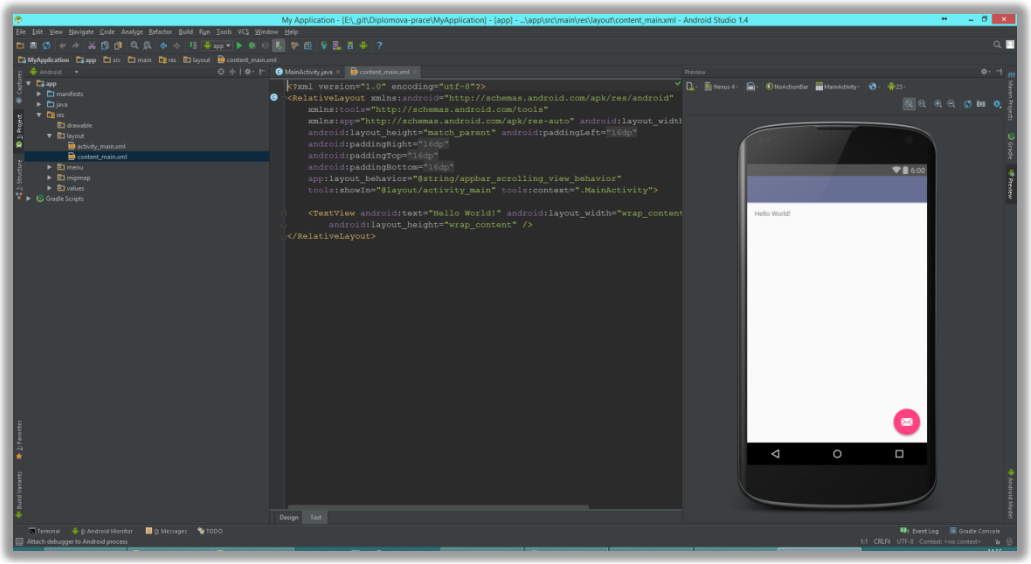
Obr. A.6: Android studio - výběr aktivity.

Zvolenou aktivitu z předchozího kroku je důležité vhodně pojmenovat a při tom dodržet velbloudí notaci (Obr. A.7).



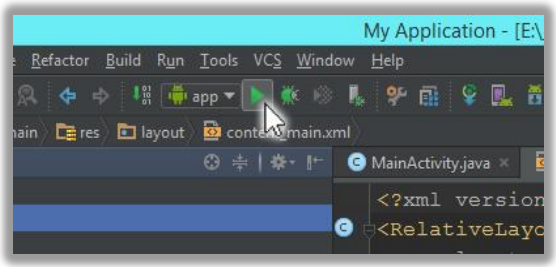
Obr. A.7: Android studio - pojmenování aktivity.

Následně po dokončení průvodce nové aplikace je otevřen zdrojový kód aplikace zobrazující prázdnou aktivitu s textem „Hello World!“ (Obr. A.8).

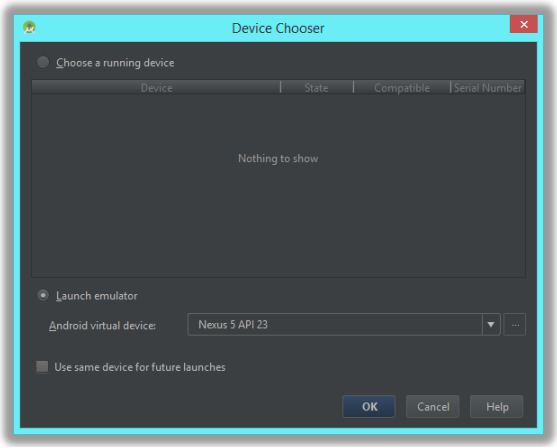


Obr. A.8: Android studio - automatický zdrojový kód Hello World aplikace ve vývojovém prostředí.

Pro spuštění emulátoru s vytvořenou aplikací stačí použít ikonu vyznačenou na obrázku Obr. A.9 a poté vybrat zařízení na kterém bude aplikace testována(Obr. A.10).

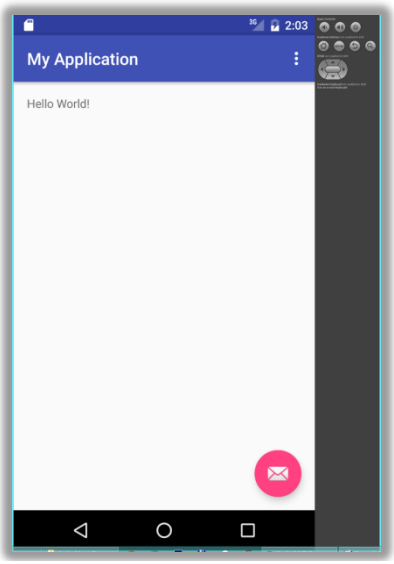


Obr. A.9: Android studio - spuštění emulátoru.



Obr. A.10: Android studio - výběr zařízení.

Spuštěný emulátor zobrazí po načtení zamčený displej a po odemčení displeje se spustí vytvořená (Obr. A.11).



Obr. A.11: Android studio - okno emulátoru.

A.3 Obsah DVD

1. Složka **Android Studio** obsahuje kompletní vývojové prostředí pro vývoj aplikací pro OS Android
2. Složka **Programátorská dokumentace** obsahuje programátorské příručky a návody k laboratorním přístrojům použitým v aplikaci
3. Složka **LABIC source** obsahuje nekompilované zdrojové kódy aplikace LABIC o které pojednává tato práce
4. Složka **LABIC app** obsahuje instalační soubor aplikace LABIC pro OS Android
5. Složka **Diplomová práce** obsahuje kopii této diplomové práce ve formátu pdf a docx