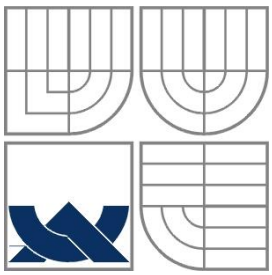
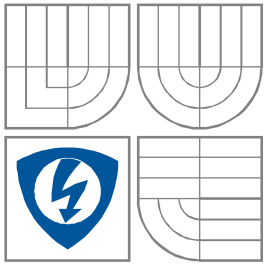


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

VÝVOJ APLIKACÍ PRO ANDROID

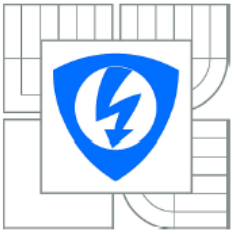
APPLICATIONS DEVELOPMENT FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Jaroslav Kvasnička
AUTHOR

VEDOUCÍ PRÁCE doc. Ing. Tomáš Frýza, Ph. D.
SUPERVISOR

BRNO, 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student:	Jaroslav Kvasnička	***ID:***	146047
Ročník:	3	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Vývoj aplikací pro Android

POKYNY PRO VYPRACOVÁNÍ:

Sestavte možnosti vývoje aplikací pro systém Android (vývojová prostředí, programovací jazyky, toolchain, ...), využitelné v praktické výuce předmětu Mikroprocesorová technika a embedded systémy. Zaměřte se na open source prostředky. Vytvořte dílčí úlohy pro počítačovou výuku; využijte dostupné periférie mobilního zařízení (touchscreen, bezdrátová konektivita, ...). Navrhněte aplikaci komunikační sítě, využívající bezdrátových modulů k přenosu dat/informací a centrálně řízeném z aplikace pro Android.

Realizujte potřebný HW a SW pro komunikační síť tzv. chytré domácnosti. Zaměřte se na praktický aspekt celé aplikace, tj. vhodný výber komponent, optimalizace napájení, zdrojových kódů, apod. Proveďte detailní testování celého systému.

DOPORUČENÁ LITERATURA:

[1] Android Developers [online]. 2014 - [cit. 12. května 2014]. Dostupné na: http://developer.android.com/develop/index.html.

[2] Android [online]. 2014 - [cit. 12. května 2014]. Dostupné na: http://www.android.com/.

Termín zadání:	9.2.2015	***Termín odevzdání:***	28.5.2015

Vedoucí práce: doc. Ing. Tomáš Frýza, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Tomáš Kratochvíl, Ph.D.
Předseda oborové rady

ABSTRAKT

V semestrálním projektu Vývoj aplikací pro Android jsou popsány nástroje, možnosti a dostupné prostředky pro vývoj aplikací na operační systém Android. V první části je představen samotný systém Android a dostupné nástroje pro vývoj aplikací. V druhé části je věnována vývoji ukázkové aplikace, na které jsou vysvětleny postupy při tvorbě samotné aplikace v různých prostředích. Třetí část je věnována návrhu experimentální komunikační sítě pro chytrý dům ovládané ze zařízení se systémem Android. Závěr je věnován zhodnocení vývoje aplikace a komunikační sítě a návrhům pro její možné vylepšení a rozšíření.

KLÍČOVÁ SLOVA

Android, vývoj aplikace, programování, bezdrátové sítě, Arduino

ABSTRACT

The semester project for Android Application Development describes the tools, options and available resources for developing applications on the Android operating system. The work has several parts. The first part is devoted to the presentation of the Android system and available tools for application development. The next section is devoted to the development of sample applications, which explains the procedures for creating the application itself in different environments. The third part is devoted to design an experimental communications network for smart home controlled by a device running Android. The conclusion is devoted to the evaluation of communication networks and proposals for its possible improvements and enhancements.

KEYWORDS

Android, applications, development, programming, wireless network, Arduino

KVASNIČKA, J. *Vývoj aplikací pro Android*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2014. 38 s. Semestrální projekt. Vedoucí práce: doc. Ing. Tomáš Frýza, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svůj semestrální projekt na téma Vývoj aplikací pro Android jsem vypracoval samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto semestrálního projektu 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

Velký dík patří vedoucímu mé práce, docentu Ing. Tomáši Frýzovy, Ph.D. za jeho rady a udání směru, kterým by se měla tato práce ubírat.

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Chytré mobilní telefony se staly z technologické novinky poměrně rychle příjemným doplňkem, ba až dokonce nutností a v současné době mají většinový podíl na trhu s mobilními telefony. Dají se koupit zařízení s různými operačními systémy, jako jsou například Microsoft Windows, Apple iOS, Google Android, Firefox OS, atd. Nejrozšířenější operační systém je ale právě Android. Díky open-source licenci, jež operační systém Android poskytuje, si může prakticky každý naprogramovat vlastní aplikaci.

Cílem této práce je úvod a seznámení s možnostmi vývoje aplikací pro operační systém Android, včetně představení možných vývojových nástrojů a úvodu do problematiky programování aplikací prostřednictvím vývoje ukázkové aplikace ve dvou různých programovacích prostředích. Dále je cílem vytvoření ukázkové komunikační sítě pro chytrou domácnost ovládané pomocí zařízení s operačním systémem Android.

Závěrečná práce je členěna do pěti základních částí. Kapitola 1 obsahuje představení samotného operačního systému Android. Možná vývojová prostředí jsou popsána v kapitole 2. Vývoj samotné ukázkové aplikace je popsán v kapitole 3. Kapitola 4 je věnována návrhu a představení komunikační sítě pro chytrý dům. Poslední kapitola 5 představuje stručné shrnutí celé práce.

1 ANDROID

V následující části je představen operační systém Android a možná vývojová prostředí pro vývoj aplikace.

1.1 Co je Android?

Android je rozsáhlá open source platforma, která vznikla zejména pro mobilní zařízení (chytré telefony, tablety, nositelná elektronika, …). Zahrnuje v sobě operační systém založený na Linuxovém jádře, Android API (software umožňující komunikaci aplikací s Linuxovým jádrem), uživatelské rozhraní a aplikace. Vyvíjí ho konsorcium Open Handset Alliance, jehož cílem je progresivní rozvoj mobilních technologií, které budou mít výrazně nižší náklady na vývoj a distribuci, a zároveň spotřebitelům přinese inovativní, uživatelsky přívětivé prostředí. Při vývoji systému byla brána v úvahu omezení, kterými disponují klasické mobilní zařízení jako výdrž baterie, menší výkonnost a málo dostupné paměti. Zároveň bylo jádro Androidu navrženo pro běh na různém hardwaru. Systém tak může být použit bez ohledu na použitý chipset, velikost či rozlišení obrazovky.

Samotná platforma Android dává k dispozici nejen operační systém s uživatelským prostředím pro koncové uživatele, ale i kompletní řešení nasazení operačního systému (specifikace ovladačů aj.) pro mobilní operátory a výrobce zařízení a v neposlední řadě pro vývojáře aplikací poskytuje efektivní nástroje pro jejich vývoj – Software Development Kit. [1] [6]

1.2 Historie OS

Historie začala založením společnosti Android Inc. V roce 2003 v Palo Alto v Kalifornii. Zakladatelé se jmenovali Andy Rubin, Rich Miner, Nick Sears a Chris White. Společnost začala vyvíjet mobilní aplikace. V srpnu roku 2005 došlo k odkoupení Android Inc. společností Google Inc. a udělal z ní svoji dceřinou společnost. Po odkupu společnosti tým Googlu pod vedením Andyho Rubina vyvinul platformu pro mobilní zařízení, založenou na Linuxovém jádře a v září roku 2007 Google získal několik patentů v oblasti mobilních technologií.

5. listopadu 2007 bylo vytvořeno uskupení Open Handset Alliance. Koncorcium, které zahrnovalo společnosti zabývající se výrobou mobilních telefonů, čipů nebo mobilních aplikací, například Google, HTC, Intel, LG, Motorola, nVidia, Qualcomm, Samsung, Texas Instruments a dalších 25 společností. Dnes konsorcium zahrnuje více než 80 firem. Jejich cílem bylo vytvořit otevřený standart pro mobilní zařízení. V ten samý den také ohlásili svůj první produkt, Android, otevřenou mobilní platformu postavenou na jádře Linux verze 2.6. O týden později byl vydán první Android SDK pro vývojáře pod licencí open-source.

V říjnu roku 2008 byl ve Spojených státech amerických uveden první komerční telefon T-Mobile G1 (HTC Dream) s operačním systémem Android (v České republice

byl uveden v lednu 2009) a zároveň s tím bylo uvolněno SDK 1.0. V roce 2009 vzrostl počet zařízení používající Android na více jak dvacet. Na konci roku 2010 se Android stal vedoucí platformou chytrých telefonů, na počátku roku 2012 měl 59% podíl a v roce 2013 už jednoznačně dominoval trhu s podílem 80 %. [1] [6]

1.3 Historie verzí

Stručný přehled softwarových změn, vylepšení a přidaných funkcí operačního systému Android, které výrobci zařízeni mohou, ale také nemusí využít. Pro přehled byli vybrány 3 verze. První verze Android 1.1 je první komerčně využitá verze v prvním mobilním zařízení s operačním systémem Android. Pro druhou vybranou verzi Android 4.0.1 byla vyvíjena ukázková aplikace a tato verze byla vybrána z důvodu největšího podílu zařízení s touto verzí Androidu na trhu. Poslední je nejnovější verze Android 5.0, který teprve čeká na masové rozšíření.

1.3.1 Android 1.1

Android 1.1 byl oficiálně představen 9. února 2009. O nové verzi 1.1 můžeme mluvit jako o změně symbolické, neboť bylo opraveno a přidáno jen minimum funkcí a celá změna se týkala pouze jednoho zařízení T-Mobile G1. [2] [6]

První Android obsahoval tyto základní funkce a aplikace:

- Webový prohlížeč pro zobrazování, posouvání a přibližování HTML a XHTML stránek s funkcí zobrazovat více stránek ve formě „karet“.
- Fotoaparát bez jakýchkoliv dalších funkcí.
- Android Market – on-line katalog (obchod), obsahující různé druhy aplikací a her.
- Email umožňující přístup k emailovým serverům, které jsou běžně k nasazení na internetu a podporují POP3, IMAP4, SMTP.
- Gmail – email od společnosti Google.
- Google Contacts – kontakty, jež jsou schopny být synchronizovány s aplikací Kontakty.
- Google Calendar – kalendář, jenž je schopen být synchronizován s aplikací Kalendář.
- Google Maps s funkcí Latitude (zjištění polohy GPS souřadnic) a Street View (pohled do ulic ve formě panoramatických záběrů v rozsahu 360° horizontálně a 290° vertikálně)
- Google Sync – software umožňující synchronizaci kontaktů, kalendáře a Gmailu.
- Googel Talk – komunikační služba umožňující zasílání textových zpráv.
- Media Player – přehrávač umožňující správu, import a přehrání audio a video souborů.
- YouTube přehrávač.
- Budík, kalkulačka, galerie obrázků, hlasové vytáčení, atd.

1.3.2 Android 4.0-4.0.4 (Ice Cream Sandwich)

Android Ice Cream Sandwich byl oficiálně představen 19. října 2011. Verze 4.0 je založena na jádře Linux 3.0.1 a jako každá vydaná verze Androidu, i tato přináší jak novinky z pohledu uživatele, tak také z pohledu vývojáře nových aplikací. Tato verze je použita v této práci jako základní verze pro programování. [2] [6]

Seznam základních změn, vylepšení a nových funkcí od verze Android 1.1:

- Rotace aplikací při rotaci zařízení.
- Nahrávání videa.
- Nahrávání videa na portál YouTube a fotografií na Google Picasa
- Optimalizována rychlost hardwaru.
- Podpora pro více velikostí a rozlišení displeje.
- Přepracovaný launcher.
- Podpora přisvětlovací diody.
- Možnost vytvořit Wi-Fi hotspot a sdílet tak internetové připojení pro více zařízení.
- Podpora pro Near Field Communication (NFC) standard
- Podpora více kamer a nových senzorů.
- USB Host.

1.3.3 Android 5.0 (Lollipop)

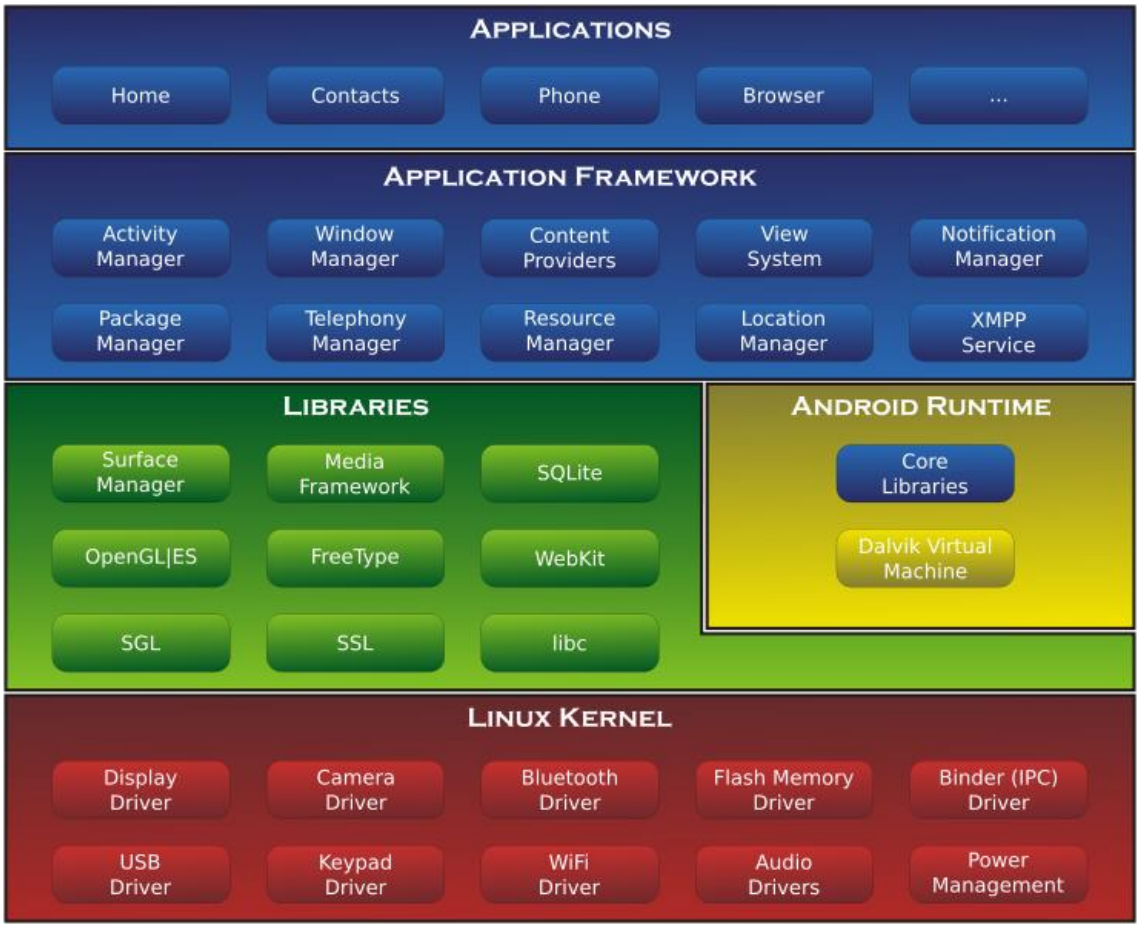
Android Lollipop byl představen 25. června 2014 na konferenci Google I/O 2014. Verze 5.0 je založena na jádře Linux 3.4.0. a přináší jak novinky z pohledu uživatele tak i z pohledu programátora. V době vzniku této práce se jedná o nejnovější verzi operačního systému android. [2] [3] [6]

Seznam základních změn, vylepšení a nových funkcí od verze Android 4.0.4:

- Možnost více uživatelských účtů.
- Podpora pro 64-bitové procesory.
- Audio vstup a výstup přes USB zařízení.
- Funkce Project Volta pro zlepšení výdrže baterie.
- Optimalizace pro telefony s menší pamětí RAM.
- Přidána funkce bezdrátového tisku a bezkontaktních plateb.
- Podpora ART Runtime s AOT kompilací, který je alternativou pro standartní Dalvik s JIT kompilací, přinášející větší výkon a také o 36% delší výdrž.
- Nový vzhled uživatelského rozhraní tzv. „materiál design“.

1.4 Architektura operačního systému Android

Architektura operačního systému Android se skládá z pěti vrstev. Jsou to Linux kernel, Libraries, Android Runtime, Application Framework a Application. Každá vrstva provádí různé operace a vystupuje víceméně samostatně. V praxi však dochází ke spolupráci jednotlivých částí a vrstvy tímto nejsou mezi sebou odděleny. [1] [6]



Obrázek 1.1: Architektura operačního systému Android [21]

1.4.1 Linux kernel

Nejnižší vrstvu architektury představuje Linux Kernel neboli jádro operačního systému. Jejím základní funkcí je implementace abstrakce mezi použitým hardwarem a softwarem ve vyšších vrstvách. Při startu zařízení je jádro zavedeno do operační paměti a je mu předáno řízení, což představuje neustálou kontrolu nad systémem a koordinaci činnosti všech běžících procesů, zabezpečení, správa paměti, síťových připojení, a správa souborů. Na Linux Kernel se také vážou ovladače pro příslušný hardware. [1] [6]

1.4.2 Libraries

Další z vrstev Android architektury jsou Libraries (knihovny). Tyto knihovny jsou napsány C nebo C++ kódu a využívají různé funkce systému. Tyto funkce jsou vývojářům poskytnuty prostřednictvím Android Application Freamwork. [1] [6]

Seznam základních knihoven:

- LibWebCore – knihovna webového prohlížeče, který podporuje i vložené náhledy webových stránek.
- OpenGl – knihovna sloužící k podpoře 3D grafiky založené na OpenGL ES.
- FreeType – knihovna pro bitmapové a vektorové vykreslování písma.
- Libc – standartní C knihovna optimalizovaná pro embedded zařízení.
- SGL – základní knihovna sloužící pro 2D grafický engine.
- SQLite – knihovna obsahující odlehčenou relační databázi, jež je dostupná všem aplikacím.
- SSL – knihovna podporující využití šifrovacího protokolu pro bezpečnou internetovou komunikaci.

1.4.3 Android Runtime

V této vrstvě jsou také obsaženy základní knihovny programovacího jazyka Java. Jejich obsah se blíží platformě Java SE (Standart Edition). Hlavní rozdíl je v nepřítomnosti knihoven pro uživatelské rozhraní AWT (Abstract Window Toolkit) a Swing, které byly nahrazeny knihovnami pro tvorbu uživatelského rozhraní pro Android a byly také přidány knihovny Apache pro práci se sítí. [1] [6]

Do verze 4.3 tato vrstva obsahuje virtuální stroj DVM (Dalvik Virtual Machine), který byl vyvíjen od roku 2005 speciálně pro Android, týmem společnosti Google pod vedením Dana Bornsteina. Novější verze od francouzsko-švýcarské firmy Myriad Group zvaná Dalvik Turbo je výrazně rychlejší, úspornější a zachovává si kompatibilitu s původním Dalvikem.

Překlad aplikace napsané v jazyce Java pro verzi Androidu starší, než verze 4.4 probíhá zkompilování zdrojového Java kódu do Java byte kódu pomocí stejného kompilátoru, jež je používán v případě překladu Java aplikací. Poté se Java byte kód překompiluje pomocí Dalvik kompilátoru a výsledný Dalvik byte kód je spuštěn na DVM. Každá spuštěná Android aplikace běží ve svém vlastním procesu s vlastní instancí DVM. [4] [6]

Od verze 4.4 se požívá ART (Android RunRime) dopředná kompilace AOT (Ahead-of-time). Důvodem je úspora energie a výrazné zrychlení aplikací. AOT kompilace funguje tak, že se aplikace napsaná v jazyce Java napřed přeloží pomocí kompilátoru do Javy byte kódu a poté se přeloží do Dalvik byte kódu. Při instalaci do

systému Android se pouze jednou a napořád zkompiluje do nativního strojového kódu procesoru zařízení. Výsledkem je až dvakrát větší výkon a také o 36% delší výdrž. Zpomalení se týká jen instalace aplikace, při které se provádí její finální optimalizace. Pro zachování kompatibility aplikací třetích stran je zatím přítomen původní JIT (Just-in-time) kompilátor. [5] [6]

1.4.4 Android Application Freamework

Pro vývojáře se jedná o nejdůležitější vrstvu, která poskytuje přístup k nejrůznějším službám, které mohou vývojáři využívat přímo v aplikacích. Tyto služby mohou poskytnout přístup k datům z jiných aplikací, prvkům uživatelského rozhraní, notifikační liště, aplikacím běžící na pozadí, nastavení alarmu, spravovat hardware zařízení atd. [1] [6]

Seznam základních služeb:

- View Systém – Umožňuje použít pro tvorbu uživatelského rozhraní prvky jako textová pole, tlačítka, seznamy, checkboxy, přepínače a jiné.
- Notification Manager – Umožňuje všem aplikacím zobrazit vlastní upozornění ve stavovém řádku.
- Content Providers – Umožňuje přistup a práci s obsahem jiných aplikací jako jsou například kontakty, kalendář a jiné.
- Resource Manager – Poskytuje přístup k „nekódovým“ zdrojům, jako jsou řetězce, grafika, přidané soubory.
- Activity Manager – Řídí životní cyklus aplikací, jejich start, průběh, ukončení a poskytuje orientaci v zásobníku s aplikacemi.
- Package Manager – Obsahuje informace o aplikacích nahraných do operačního systému.

1.4.5 Application

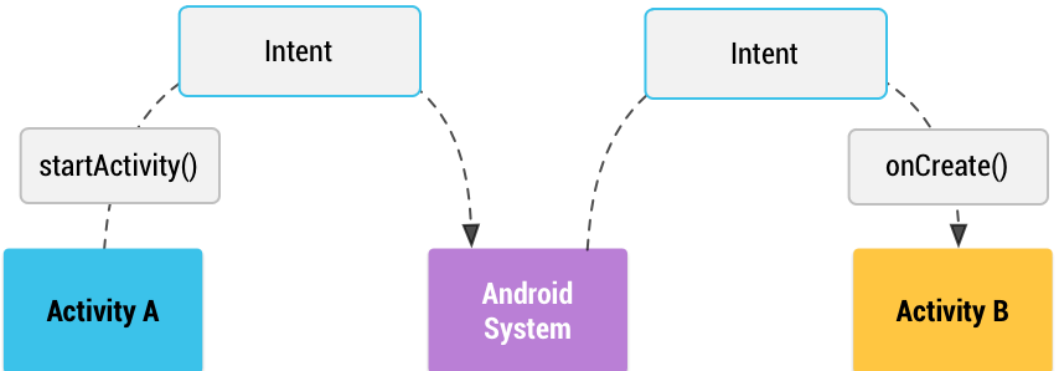
Poslední a nejvyšší vrstvu systému tvoří již samotné aplikace, které jsou využívány uživateli. Jedná se o aplikace již předinstalované nebo dodatečně stažené. [1] [6]

1.5 Základní komponenty aplikace pro Android

Aplikace pro Android se skládá z několika základních komponent, které jsou na sebe volně vázané. Jejich samotná vazba se definuje v pomocí Android Manifestu [7].

1.5.1 Intent

Intent umožňuje posílání zpráv mezi jednotlivými komponentami s požadavkem na provedení určité akce z jiné komponenty aplikace. Intent je zde pro usnadnění komunikace a kooperace jednotlivých komponent. [6] [8]



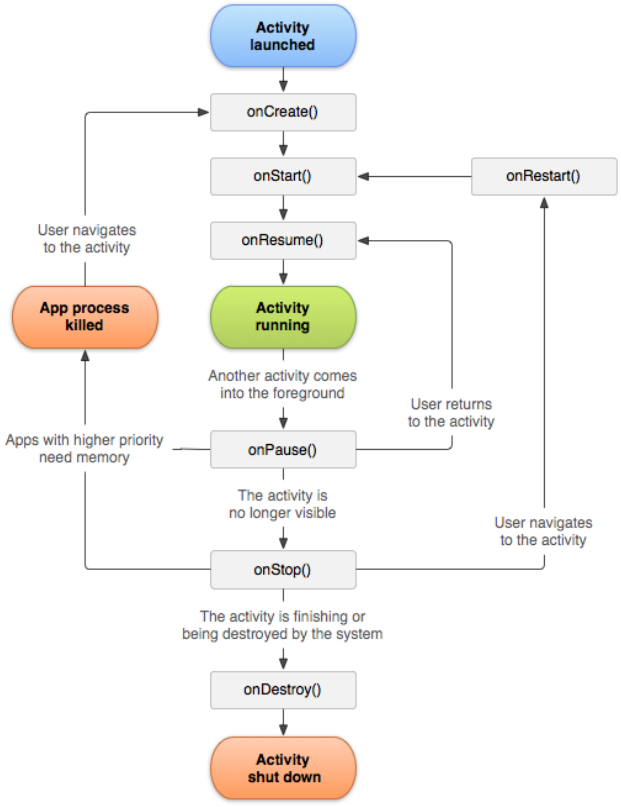
Obrázek 1.2: Znázornění vytvoření Activity pomocí Intent [22]

1.5.2 Content Providers

Content Providers pracuje s daty (ukládání, načítání) řídí přístup k těmto datům pro všechny aplikace. Představují jediný způsob, jak sdílet data v rámci celé aplikace, protože zde neexistuje žádné běžné úložiště, do kterého by měli přístup všechny balíčky operačního systému Android. Operační systém Android je dodáván s Centent Providers pro běžné typy dat, jako jsou například audio, video, obrázky, kontaktní informace, atd. [6] [9]

1.5.3 Activity

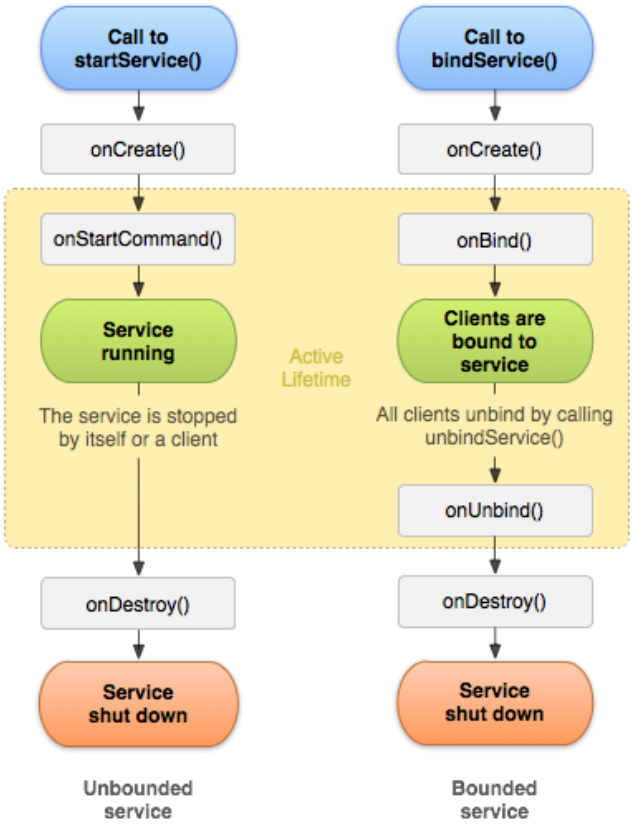
Activity v aplikaci představují základní vizuální komponentu obsahující grafické uživatelské rozhraní pro interakci s uživatelem. Activity představuje jednu obrazovku aplikace. Aplikace obsahuje více Activity, které jsou mezi sebou volně vázány a mohou si předávat informace. Uživatel je schopen mezi nimi volně přepínat. Zahájení activity je poměrně náročná záležitost. Musí se vytvořit nový proces, alokovat paměť pro objekty uživatelských rozhraní, které se rozloží do layoutu obrazovky a připravenou obrazovku vyvolat zobrazení. Aby nedocházelo ke zbytečnému plýtvání výpočetních prostředků je zde Activity Manager, který zodpovídá za vytváření, rušení a celkovou správu životního cyklu Activity [10] [6].



Obrázek 1.3: Životní cyklus Activity popisující její existenci [23]

1.5.4 Service

Service je komponenta aplikace, která provádí dlouhotrvající operace na pozadí, a která neposkytuje uživateli graficko-uživatelské rozhraní a není na něj vázána. Service může být spuštěn komponentou aplikace a může být stále aktivní na pozadí, i když uživatel přejde do jiné aplikace. Navíc lze k Service vázat komponentu aplikace, komunikovat s ním a provádět i mezi procesorovou komunikaci. Například se může Service starat o síťový přenos, přehrávání hudby, komunikovat s Content Providers a to vše na pozadí [11] [6].



Obrázek 1.4: Životní cyklus Service [24]

1.5.5 Android Manifest

Android Manifest je XML soubor s názvem AndroidManifest.xml, který musí být obsažen v kořenovém adresáři aplikace. Tento soubor sděluje systému základní informace o aplikaci, jako jsou název, verze, jednotlivé komponenty, požadovaný systémová práva a další požadavky pro samotný chod aplikace [6] [12].

2 VÝVOJOVÉ NÁSTROJE A PROSTŘEDÍ

Aplikace pro Android jsou napsány v programovacím jazyce Java. Proto lze pro vytvoření aplikace použít klasický textový editor nebo prakticky jakékoliv vývojové prostředí používané pro programovací jazyk Java a poté do něj přidat Android SDK (Software Development Kit). V této práci jsou použita vývojová prostředí Android Studio a App Invertor 2 [6].

2.1 Java Development Kit (JDK)

Java Development Kit (JDK) je soubor základních nástrojů a knihoven pro vývoj aplikací a apletů pro platformu Java. Základní součástí JDK je Java Runtime Enviroment, jež slouží pro spuštění aplikací i vývojových nástrojů, atd. Jelikož se aplikace pro Android programují v jazyce Java je nutné JDK nainstalovat. [13] [6]

Poslední verzi JDK lze stáhnout na adrese:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

2.2 Vývojové prostředí Android Studio

Android Studio je nové vývojové prostředí založené na IntelliJ a vyvíjeno přímo společností Google. Jeho největší výhodou je, že byl od základu vyvíjen pro vývoj aplikací pro Android, a proto obsahuje některé vylepšení, které jinde nevyskytují. Jedná se například o podporu vývoje více variant a multigeneračního APK, který umožní vyvíjet jednu aplikaci jak pro Android Phone, tak i pro Android TV a Android Wear zároveň. Tudíž se nemusí programovat aplikace pro každou platformu zvlášť. Další velkou výhodou je označení Android Studia za oficiální vývojové prostředí, což sebou nese dobrou podporu, časté aktualizace a časem se zvětšující množství informací pro vývoj aplikací. V době vývoje se ale nacházelo pouze v beta verzi a plná verze vyšla až 9. 12. 2014. [14].

Poslední verzi Android Studia lze stáhnout na adrese:

http://developer.android.com/sdk/index.html

2.3 Vývojové prostředí App Inventor 2

App Inventor 2 je open-source webovou službou, vyvinutou společností Google a poprvé spuštěnou 12. 7. 2010. V druhé polovině roku 2011 společnost Google uvolnila zdrojové kódy, ukončila službu na svém serveru a poskytla finanční prostředky na vytvoření MIT Center of Mobile Learning. Verze App Invertor 2 od MIT byla spuštěna v březnu 2012. 6. prosince 2013 byla pak vypuštěna verze App Inventor 2.

App Inventor 2 je grafické programovací prostředí, ve kterém se dají vyvářet aplikace i bez znalosti programovacích jazyků. Protože se jedná o kompletně webovou službu, nemusí se nic stahovat ani instalovat. Veškerá kompilace se provádí na serveru, tudíž není potřeba ani výkonný počítač. Pro započnutí vytváření aplikací se stačí jen přihlásit. [15] [16]

Přihlášení a vývoj aplikace:

http://ai2.appinventor.mit.edu/

2.4 Ostatní vývojové nástroje a prostředky

Další slibný vývojový nástroj je i Eclipse. Jedná se o open source vývojovou platformu určenou pro programování v jazyce Java. Výhodou byla přímá podpora od Google, a proto se Eclipse dalo stáhnout už s implementovaným Android SDK a Android Developer Tool (ADT), což velice usnadňovalo práci při instalaci. Tato výhoda ale uvedením Android Studio verze 1.0 skončila. Eclipse se dá pro vývoj aplikací pro android stále využít, ale implementace Android SDK a ADT musí proběhnout zvlášť [17].

Jako další zajímavá možnost je požití Scripting Layer for Android (SL4A). Jedná se o knihovnu, která dovoluje vytvářet a spouštět skripty napsané v mnoha skriptovacích jazycích jako jsou například Python, Perl, Ruby, Lua, BeanShell, JavaScript, TCL a Rexx. Tyto skripty mají přístup k mnoha API přístupných i normálním aplikacím. [18] [19]

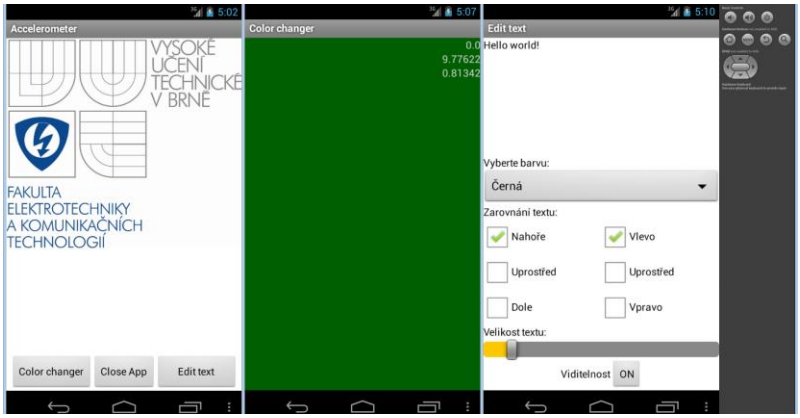
V neposlední řadě je možnost vyvíjet aplikace i v Microsoft Visual Studio, který nově nabízí také vývoj multiplatformních aplikací použitím HTML, CSS a JavaScript založený na Apache Cordova, což je soubor API, který umožňuje vývojářům stejný přístup k funkcím jako kamera nebo akcelerometr ať už se jedná o Android. iOS či Windows Phone. [20]

3 VÝVOJ APLIKACE

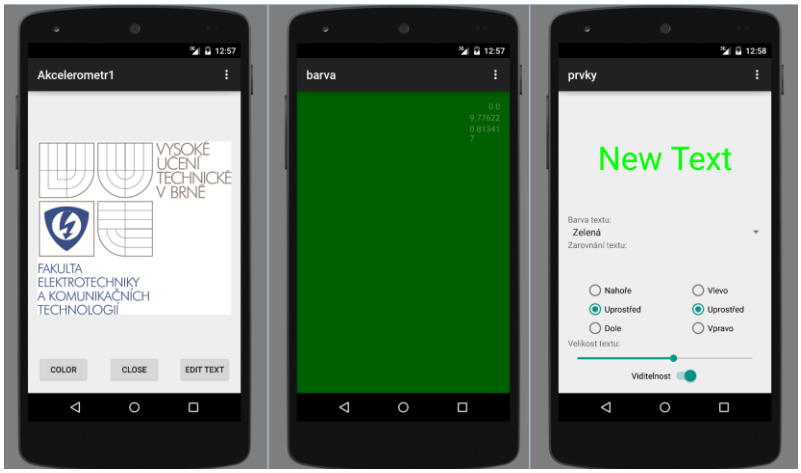
V této kapitole je popsán příklad vývoje jednoduché aplikace. Vývoj bude ukázán ve dvou vývojových prostředích. Jako periférie mobilního zařízení byli zvoleny dotyková vrstva pro ovládání aplikace, akcelerometr pro detekci polohy zařízení a displej zařízení pro zobrazení informací. Místo několika dílčích aplikací byla vytvořena jedna obsáhlejší, z důvodu lepší ukázky provázanosti jednotlivých prvků a částí, ze kterých je složena každá aplikace. Aplikace se skládá ze tří základních částí, kterých není problém ji rozdělit na jednotlivé dílčí úlohy.

3.1 Popis aplikace

Ukázková aplikace se skládá ze tří obrazovek, jak je vidět na obrázku níže. Na první obrazovce je logo VUT a tří tlačítka, z nichž dvě slouží pro přechod na další obrazovky a třetí na ukončení aplikace. Druhá obrazovka obsahuje dynamicky měnící barvu pozadí a zobrazení dat z akcelerometru. Třetí obrazovka obsahuje text „Hello world!“ a základní ovládací prvky pro úpravu barvy, zarovnání, velikosti a viditelnosti textu.



Obrázek 3.1: Ukázková aplikace vytvořená v prostředí App Inventor 2



Obrázek 3.2: Ukázková aplikace vytvořená v prostředí Android Studio

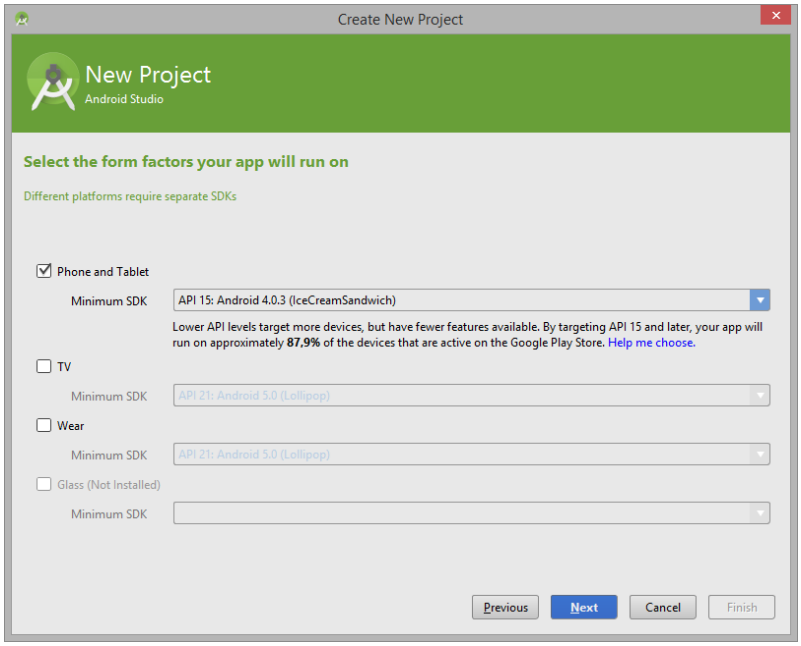
3.2 Vývoj aplikace v prostředí Android Studio

Jako první bude aplikace vyvíjena v prostředí Android Studio a následně v prostředí App Inventor 2 pro srovnání jednotlivých prostředí. Verze prostředí Android Studio, ve kterém byla tato aplikace vytvořena, je 1.0. V této práci je použito přednastavené rozlišení displeje referenčního zařízení Nexus 4. Toto zařízení má rozlišení 768x1280.

3.2.1 Založení nové aplikace v prostředí Android Studio

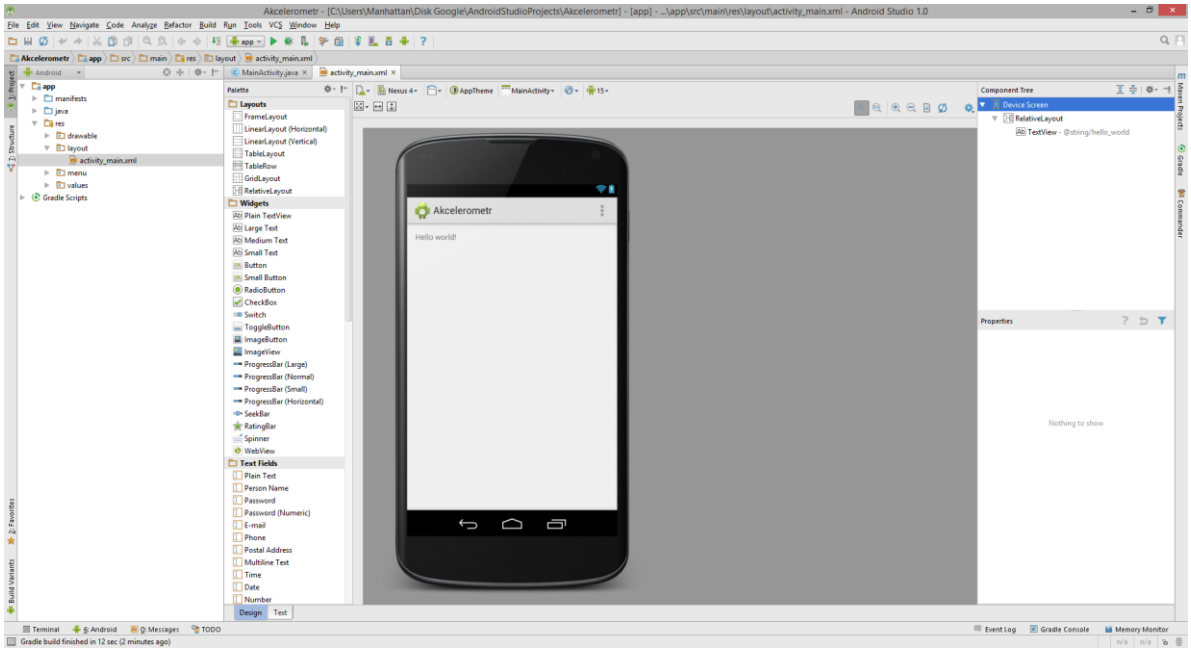
Po otevření vývojového prostředí Android Studio se vytvoří nový projekt pomocí *Start a new Android Studio Project.*

Otevře se dialogové okno pro vytvoření projektu, do něhož se vyplní název projektu, v tomto případě Accelerometer. Na další straně se zvolí zařízení, pro které bude aplikace vytvářena, tedy *Phone and Tablet*, a verzi Androidu na které aplikace poběží. Byla zvolena verze Androidu 4.0.3 z důvodu největšího počtu zařízení s touto verzí.



Obrázek 3.3: Výběr názvu aplikace a cílové verze Androidu

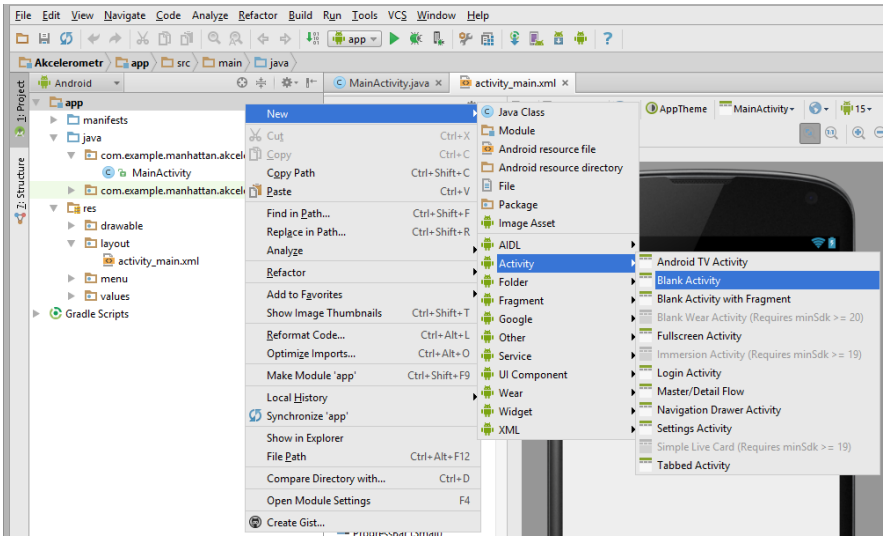
V následujícím okně se volí typ Activity, která se bude primárně vyvíjet. Zvolí se *Blank Activity* a klikne na tlačítko *Next*. V posledním okně se vyplní název hlavní aktivity. Zde byl ponechán původní název *MainActivity*. Vytváření projektu se dokončí kliknutím na tlačítko *Finish*.



Obrázek 3.4: Nová aplikace ve vývojovém prostředí Android Studio

3.2.2 Příprava aplikace v prostředí Android Studio

Pokud se již dopředu ví, kolik bude mít aplikace obrazovek je dobré si je připravit. Klikne se pravým tlačítkem myši na složku *app* v levé části programu a najede na *New > Activity > Blank Activity*.

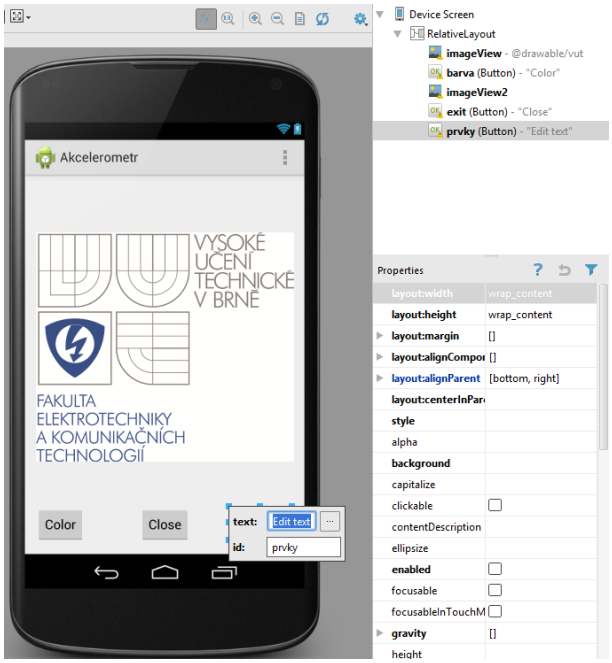


Obrázek 3.5: Přidání Activity

Otevře se dialogové okno, kde se zvolí název Activity. Název *MainActivity2* se změní na *barva* a název grafického rozhraní se změní automatiky na *activity_barva*. Ten se ponechá. Poté se vytvoří další Activity s názvem *prvky*.

3.2.3 Vytvoření grafického rozhraní hlavní obrazovky

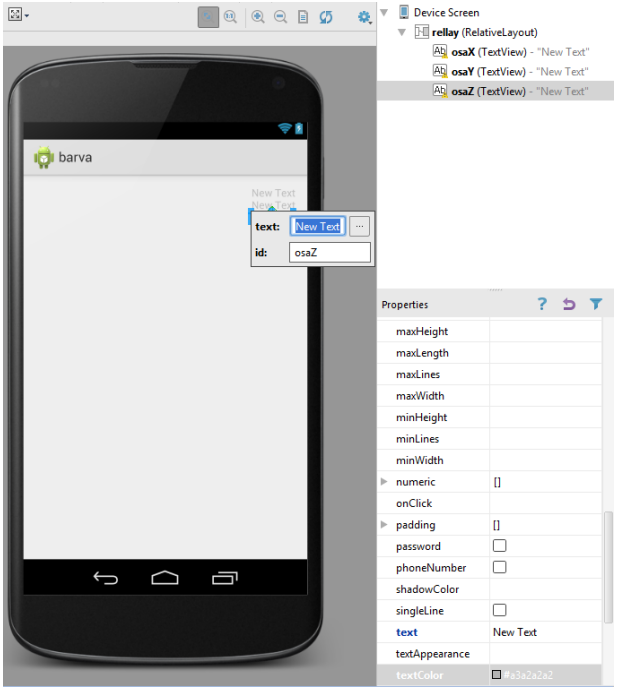
Otevře se soubor *activity_main.xml*, který se otevře v grafickém designéru. Zde se dá pohodlně vytvářet grafické uživatelské rozhraní pro aplikaci. Na pravé straně je zobrazeno okno *Component Tree* s hierarchií vytvořeného grafického prostředí a přehledem použitých prvků. Jako první se smaže prvek *TextView* zobrazující nápis „Hello world!“. Prvek *RelativeLayout* zůstane. Jako další se do zdrojů přidá požitý obrázek. Klikne se pravým tlačítkem myši na složku *drawable* ve složce *res* a poté se vybere *Show in Explorer*. Otevře se složka, ve které se otevře *drawable-xhdpi* a zde se vloží obrázek *vut.png*. Z palety v záložce *Widgets* se přetáhne prvek *ImageView*. Dvakrát na něj klikne levým tlačítkem myši a objeví se okénko, kde jsou řádky *src:* a *id:*. Do *src:* se napíše *@drawable/vut* a na obrazovce se objeví zvolený obrázek. Druhý řádek se nechá beze změny. Poté v se *Properties* nastaví *layout:width* na *fill_parrent*. Jako další se pod obrázek přetáhnou tři tlačítka *Button* ze záložky *Widgets*. Postupně se u všech třech změní popis a ID. To se udělá dvojím kliknutím levého tlačítka a změnou *text:* a *id:*. U prvního tlačítka se nastaví *text:* na *Color* a *id:* na *barva*. U druhého tlačítka bude *text: Close* a *id: close* a u třetího *text: Edit text* a *id: prvky*.



Obrázek 3.6: Rozvržení grafického prostředí hlavní obrazovky

3.2.4 Vytvoření grafického rozhraní obrazovky pro změnu barvy

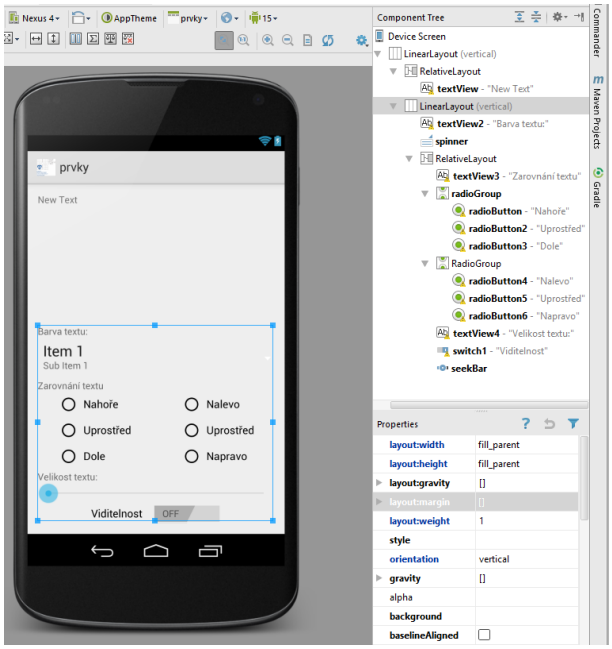
Otevřeme *activity_barva.xml* a jako první se smaže prvek *TextView* zobrazující nápis „Hello world!“. Prvek *RelativeLayout* zůstane. Z palety v záložce *Widgets* se přetáhnou tři prvky *Plain TextView* a po poklepání se postupně změní *id:* u všech tří prvků na *osaX*, *osaY* a *osaZ*. Může se také změnit barva textu, aby byla lépe viditelná na tmavých barvách. V *Properties* se vybere *textColor*, klikne se na tři tečky a v dialogovém okně se vybere *Color*. Zde se dá namíchat jakákoliv barva. Také se změní *id:* pro *RelativeLayout* poklepání na pozadí a změnou *id:* na *rellay“*.



Obrázek 3.7: Rozvržení grafického prostředí obrazovky pro změnu barvy pozadí

3.2.5 Vytvoření grafického rozhraní obrazovky pro úpravu textu

Otevře se *activity_prvky.xml* a smaže prvek *TextView* zobrazující nápis „Hello world!“. Pod paletou se přepne na *Text* a oba *RelativeLayout* se přepíšeou na *LinearLayout* a přidá se řádek *android:orientation = “vertical“*. Tím se zajistí vertikální rozdělení obrazovky. Přepne se zpět a následně se do prvního *LinearLayout* vloží *RelativeLayout,* a do něj *Plain TextView*. U něj se v *Properties* změní *layout:width* a *layout:height* na hodnotu *fill_parrent.* Pod *RelativeLayout* se vloží další *LinearLayout(vertical).* U *RelativeLayout* a druhého *LinearLayout* v *Properties* se nastaví *layout:width* a *layout:height* na *fill_parrent*. Přepne se na *Text* kde se k *RelativeLayout* a druhému *LinearLayout* přidá *android:layout_weight=“1“* zajišťující rozdělení zobrazení 1:1.5. Do druhého *LinearLayout* se poté vloží pod sebe *Plain TextView*, *Spinner* a další *RelaveLayout* u kterého se opět nastaví *layout:width* a *layout:height* na *fill_parrent* . Do druhého *RelativeLayout* se vloží *Plain TextView* a dva *RadioGroup*. Do každého *RadioGroup* se vloží tři *RadioButton* a *RadioGroup* se uspořádá vedle sebe. Pod *RadioGroup* se vloží *Plain TextView*, *SeekBar*, *Switch* a uspořádají se.



Obrázek 3.8: Rozložení grafického prostředí pro úpravu textu a jejich hierarchie

3.2.6 Vytvoření hlavní Activity v prostředí Android Studio

Otevře se *MainActivity.java*, kde se bude psát, co by aplikace měla dělat. V případě této aplikace naslouchat stisknutí některého ze tří tlačítek a následně reagovat naprogramovaným způsobem, tj. spuštěním jiné Activity.

Do použitých knihoven se importuje knihovna pomocí *import android.view.View;* a do *public class MainActivity* se implementuje *View.OnClickListener* pomocí *implements*. Po jeho implementaci se v levém okraji pracovního prostoru zobrazí červená žárovka. Klikne se na tuto žárovku a v tabulce s řešením problému se zvolí *Implement Methods*. Po jeho zvolení se objeví okno se zvolením metody, kterou jr potřeba implementovat a zvolí se ta s názvem *onClick()* a pak *OK*. Poté se deklarují tlačítka přidáním *Button talcitkoText; Button tlacitkoBarva;* a *Button tlacitkoExit;* před *protected void onCrate()* čili vytvoření Activity. Při napsání *Button* by se měl do knihoven přidat řádek *import android.widget.Button;*. Pokud ne dopíšeme ho ručně.

```
import android.view.View;
import android.widget.Button;

public class MainActivity extends ActionBarActivity implements
View.OnClickListener {

    Button tlacitkoText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

Jako další se přiřadí definované proměnné *Button* k jednotlivým tlačítkům definovaným v .xml souboru. Deklarace, která proměnná odpovídá tlačítku, se píše do metody *onCreate* příkazem *tlacitkoBarva=(Button)findViewById(R.id.barva);*. Také se hned pod přiřazením tlačítek zapne naslouchání kliknutí na tyto tlačítka pomocí *tlacitkoBarva.setOnClickListener(this);*. Nyní v se v metodě *onClick()* určí co se má stát, pokud se stiskne nějaké tlačítko. O jaké tlačítko se jedná, se zjistí otestováním názvu komponenty, která stisknutí vyvolala. Pro toto testování byl zvolen příkaz *switch(v.getId()) – case R.id.barva*. Kvůli přehlednosti a možnosti modifikace se nebude psát příkaz k otevření nové Activity ale odkáže se na novou metodu, ve které se nová Activity otevře. To se udělá tak že se pod case zapíše název metody, na kterou se odkazuje. V tomto případě *tlacitkoBarvaKlik()*, *tlacitkoExitKlik()* a *tlacitkoTextKlik()*. Poté se vytvoří příslušné metody kdekoliv v *public class MainActivity* pomocí *private void tlacitkoBarvaKlik()*. Jiná Activity se otevře pomocí příkazu *startActivity(new Intent("com.example.philadelphia.akcelerometr1.barva"));*. Část příkazu *com.example.manhattan.akcelerometr* představuje název balíčku, ve kterém se Activity s názvem *prvky* nachází. Vyjímkou je *tlacitkoExitKlik()* ve kterém se volá ukončení programu pomocí *finish();* a *System.exit();*.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tlacitkoBarva = (Button)findViewById(R.id.barva);
        tlacitkoBarva.setOnClickListener(this);
    }
@Override
    public void onClick(View v) {
        switch (v.getId())
        {
            case R.id.barva:
                tlacitkoBarvaKlik();
                break;
            case R.id.close:
                tlacitkoExitKlik();
                break;
        }

    }

    private void tlacitkoBarvaKlik()
    {
startActivity(new Intent("com.example.manhattan.akcelerometr.barva"));
    }

    private void tlacitkoExitKlik()
    {
        finish();
        System.exit(0);
    }
```

3.2.7 Vytvoření Activity barva v prostředí Android Studio

Otevře se *barva.java* a do hlavní třídy *barva* se implementuje *SensorEventListener*. Po jeho implementaci se v levém okraji pracovního prostoru zobrazí červená žárovka. Klikne se na tuto žárovku a v tabulce s řešením problému zvolíme *Implement Methods*. Objeví okno se zvolením metody, která se bude implementovat a zde se klikne na *OK*. Nyní se s použitím *private* deklaruje použité proměnné, které jsou *Sensor akcelerometr, Sensor sensorManager, RelativeLayout relat, TextView osaX, TextView osaY, TextView osaZ, float deltaX = 0, float deltaY = 0* a *float deltaZ = 0.* Poté se k deklarovaným prvkům *relat, osaX, osaY a osaZ* přiřadí prvky z grafického rozhraní pomocí *relat = (RelativeLayout)findViewById(R.id.rellay);* a *osaX = (TextView)findViewById(R.id.osaX);*.

```
public class barva extends ActionBarActivity implements
SensorEventListener {

    private Sensor akcelerometr;
    private SensorManager sensorManager;
    private RelativeLayout relat;
    private TextView osaX, osaY, osaZ;

    private float deltaX = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_barva);

        relat = (RelativeLayout)findViewById(R.id.rellay);
        osaX = (TextView)findViewById(R.id.osaX);

    }
```

Nyní se deklaruje akcelerometr. Jako první se přiřadí k proměnné *sensorManager* přístup k senzorům zařízení pomocí příkazu *sensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);* Poté se otestuje, jestli je akcelerometr k dispozici a pokud ano, bude *sensorManager* naslouchat změnám akcelerometru.

```
sensorManager=(SensorManager)getSystemService(Context.SENSOR_SERVICE);

  if (sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)!=null)
        {
      Akcelerometr =
            sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

      sensorManager.registerListener(this, akcelerometr,
            SensorManager.SENSOR_DELAY_GAME);
        }
        else
        {
        }
```

Jako další se budou zapisovat hodnoty z akcelerometru do proměnných *deltaX, deltaY* a *deltaZ*. V metodě *onSensorChanged()* se přidá řádek *deltaX=event.values[0]*; zajišťující přepis z pole do kterého se zapisují údaje z akcelerometru do proměnných pro pozdější úpravu. Po přepsání se zavolá metoda *zobrazHodnoty();* kterou se poté vytvoří. Tu lze lehce vytvořit tak, že se po napsání volání metody klikne na červenou žárovku, která se zobrazí a zvolí se *Create Method 'zobrazHodnoty'*. V této metodě se přepíší hodnoty z *deltaX, deltaY* a *deltaZ* do textových polí *osaX*, *osaY* a *osaZ* pomocí *osaX.setText(Float.toString(deltaX));*. Data z akcelerometru při naklánění v jednotkách až desítkách, kladných i záporných hodnot a ve formátu *float*. Protože je ale k namíchání RGB barvy potřeba 0 až 255 ve formátu *int* musí se hodnoty z akcelerometru upravit. Zde byla zvolena mocninné funkce pro zvětšení a získání kladných hodnot. Následně se převede *float deltaX, deltaY* a *deltaZ* na nové proměnné *int R, G* a *B*. Teď už jen zbývá změnit aktuální barvu pozadí pomocí proměnných *R, G* a *B* příkazem *relat.setBackgroundColor(Color.argb(255, R, G, B));*.

```
@Override
    public void onSensorChanged(SensorEvent event) {

        deltaX = event.values[0];
        deltaY = event.values[1];
        deltaZ = event.values[2];

        zobrazHodnoty();
    }

    private void zobrazHodnoty() {

        osaX.setText(Float.toString(deltaX));
        osaY.setText(Float.toString(deltaY));
        osaZ.setText(Float.toString(deltaZ));

        deltaX = deltaX*deltaX;
        deltaY = deltaY*deltaY;
        deltaZ = deltaZ*deltaZ;

        int R = (int)deltaX;
        int G = (int)deltaY;
        int B = (int)deltaZ;

        relat.setBackgroundColor(Color.argb(255, R, G, B));


    }
```

3.2.8 Vytvoření Activity prvky v prostředí Android Studio

Otevře se *prvky.java* a do hlavní třídy *public class prvky* se implementuje *AdapterView.OnItemSelectedListener* pomocí *implements* pro naslouchání zvolení prvku v rozevíracím seznamu *Spinner* a po zobrazení červené žárovky se implementuje metoda *onItemSelected()*. Také se implementuje *OnSeekBarChangeListener* pro naslouchání změny polohy *SeekBar* a po zobrazení červené žárovky i metoda *onProgressChanged()*. Pro ukázku vývojových možností se nebude implementovat naslouchače prvků *RadioButton* v hlavní třídě jako předchozí dva, ale bude se odkazovat přímo z *activity_prvky.xml*. To se udělá tak, že se otevře activity_prvky.xml a pod hlavním oknem se přepne na záložku *Text*. U všech *RadioButton* se přidá řádek *android:onClick="onRadioButtonClicked"* pro odkaz na metodu *onRadioButtonClicked* po kliknutí. Naslouchání posledního prvku *Switch* se bude dělat přímo v metodě *onCreate()*, ale před tím se musí deklarovat použité proměnné. Ty se budou deklarovat pomocí *private*. Jsou to tyto prvky *Spinner seznam1*, *TextView textView1*, *SeekBar seekBar*, *Switch prepinac*, *int test1 = 0* a *int test2 = 0*. Také se do pole *Sting[] barva* nadeklaruje, co se bude zobrazovat v rozevíracím seznamu *Spinner*. V tomto případě jsou to barvy černá, červená, žlutá, zelená, tyrkysová a modrá. V deklaraci proměnných chybí RadioButton, protože metoda *onRadioButtonClick()* je volána přímo z *activity_prvky.xml*.

```
public class changetext extends Activity implements
OnItemSelectedListener, OnSeekBarChangeListener{

       private Spinner spinner1;
       private TextView textView1;
       private SeekBar seekBar1;
       private Switch switch1;


       private int test1 = 0;
       private int test2 = 0;

       private String[] barva = { "Černá", "Červená", "Žlutá",
"Zelená", "Tyrkysová", "Modrá" };
protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.changetext);
}
}
```

Poté se k deklarovaným proměnným přiřadí prvky použité v grafickém prostředí v metodě *onCreate()*. Poté z řady *barva* uděláme adaptér pro *Spinner* pomocí *ArrayAdapter<String> adapter_barva = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, barva);* a poté se z adapteru vytvoří seznam *adapter_barva.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);*. Poté se tento seznam přiřadí ke *Spinner* pomocí příkazu *seznam1.setAdapter(adapter_barva);* a zapne se naslouchač na zvolení položky ve *Spinner* pomocí *setOnItemSelectedListener(this)*. Také se zapne naslouchač na změnu *SeekBar* pomocí *setOnSeekBarChangeListener(this)*.

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_prvky);

        seznam1 = (Spinner)findViewById(R.id.spinner);
        textView1 = (TextView)findViewById(R.id.textView);
        seekBar = (SeekBar)findViewById(R.id.seekBar);
        prepinac = (Switch)findViewById(R.id.switch1);

        ArrayAdapter<String>         adapter_barva         =         new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item,
barva);

adapter_barva.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);

        seznam1.setAdapter(adapter_barva);
        seznam1.setOnItemSelectedListener(this);

        seekBar.setOnSeekBarChangeListener(this);

    }
```

Jako další se v metodě *onCreate()* vytvoří funkce pro zneviditelnění textu pomocí *Switch*. Napřed se ale zajistí, aby byl *Switch* při spuštění Activity na pozici *On*. To se udělá pomocí příkazu *prepinac.setCheckd(true)*. Dále se zapne naslouchač přepnutí, do kterého se přímo napíše potřebná funkce zajišťující zneviditelnění. To se udělá otestováním polohy *Switch* a následným příkazem *setVisibility()*.

```
  @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_prvky);

        prepinac.setChecked(true);
        prepinac.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public  void  onCheckedChanged(CompoundButton  buttonView,
boolean isChecked) {
                if(isChecked)
                {
                    textView1.setVisibility(0);
                }else{
                    textView1.setVisibility(4);
                }
            }
        });
    }
```

Nyní se bude pomocí *Spinner* měnit barva zobrazeného textu. To se udělá pomocí testování pozice vybraného prvku v metodě *onItemSelected()* příkazem *switch(position) – case 1: textView1.setTextColor(Color.parseColor("#000000"))*. Hodnota v uvozovkách představuje RGB kód barvy. V tomto případě jsou požity barvy s hodnotami černá = "#000000", červená = "#FF0000", žlutá = "#FFFF00", zelená = "#00FF00", tyrkysová = "#00FFFF" a modrá = "#0000FF".

```
@Override
    public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
            switch (position) {
                case 0:
                  textView1.setTextColor(Color.parseColor("#000000"));
                  break;
            }
    }
```

Teď bude na řadě zarovnání textu. K tomu se využijí všechny *RadioButton* v obou *RadioGroup* a kvůli jejich provázanosti i proměnné *test1* a *test2*. Jelikož se kliknutí řeší odkazováním z *activity_prvky.xml* na konkrétní metodu, musí se také tato metoda vytvořit. Vytvoří se přidáním *public void onRadioButtonClick(View view)* do *public class prvky*. Do této metody se přidá kontrola stisknutí tlačítka *boolean checked = ((RadioButton) view) .isChecked();*. Poté se bude testovat podle ID prvku, na který *RadioButton* bylo kliknuto, a podle toho se nastaví zarovnání textu pomocí *textView1.setGravity (Gravity.TOP).* Po napsání *Gravity.TOP* se nám objeví červená žárovka. Klikne se na ni a zvolí se *Import Class*. Logika pro vzájemné testování obou *RadioGroup* je uvedena jako zdrojový kód níže.

```
public void onRadioButtonClick(View view){
  boolean checked = ((RadioButton)view).isChecked();
  switch (view.getId()){
   case R.id.radioButton:
    if (checked)
      test1=0;
      if (test2==0)
       textView1.setGravity(Gravity.TOP | Gravity.LEFT );
      else if (test2==1)
       textView1.setGravity(Gravity.TOP | Gravity.CENTER_HORIZONTAL );
      else
       textView1.setGravity(Gravity.TOP | Gravity.RIGHT );
                  break;
   case R.id.radioButton6:
    if (checked)
      test2=2;
      if (test1==0)
       textView1.setGravity(Gravity.TOP | Gravity.RIGHT );
      else if (test1==1)
       textView1.setGravity(Gravity.CENTER_VERTICAL | Gravity.RIGHT );
      else
       textView1.setGravity(Gravity.BOTTOM | Gravity.RIGHT );
                  break;
        }
    }
```

Jako poslední se bude měnit velikost textu pomocí *SeekBar*. Využije se toho, že hodnoty z něj jsou v podobě *int*. Tato hodnota v základu odpovídá hodnotám velikostí písma, a proto se může přímo využít. To se udělá přidáním příkazu *textView1.setTextSize(progress)*; přímo do metody *public void onPrograssChanged()*.

```
    public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
        textView1.setTextSize(progress);
    }
```

3.2.9 Úprava AndroidManifest.xml v prostředí Android Studio

Otevře se *AndroidManifest.xml* nacházející se ve složce *app* > *manifest*. Zde se musí doplnit Intent pro jednotlivé Activity, protože je na ně pomocí Intent odkazováno. Protože se používá odkazování na Activity *prvky* a *barva*, bude stačit zavést Intent jen u těchto dvou. To se udělá tak, že mezi *<activity android:label ="@string/title_activity_barva">* a *</activity>* vložíme *<intent-filter><action android:name="com.example.philadelphia.akcelerometr1.barva"/>*, kde *"com. example.philadelphia.akcelerometr1"* představuje název balíčku, ve kterém se tato aktivita nachází a *barva* je název samotné Activity. Dále se deklaruje kategorie Intent pomocí *<category android:name="android.intent.category.DEFAULT" />* a ukončí se *</intent-filter>*. To samé se udělá i pro Activity *prvky*.

```
<activity
       android:name=".MainActivity"
      android:label="@string/app_name" >
      <intent-filter>
         <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.LAUNCHER" />
       </intent-filter>
</activity>
<activity
      android:name=".barva"
      android:label="@string/title_activity_barva" >
      <intent-filter>
<action android:name="com.example.philadelphia.akcelerometr1.barva" />
      <category android:name="android.intent.category.DEFAULT" />
      </intent-filter>
</activity>
<activity
      android:name=".prvky"
      android:label="@string/title_activity_prvky" >
      <intent-filter>
<action android:name="com.example.philadelphia.akcelerometr1.prvky" />
           <category android:name="android.intent.category.DEFAULT" />
      </intent-filter>
</activity>
```

Nyní je aplikace hotová a může se otestovat v emulátoru kliknutím na zelenou šipku.

3.3 Vývoj aplikace v prostředí App Inventor 2

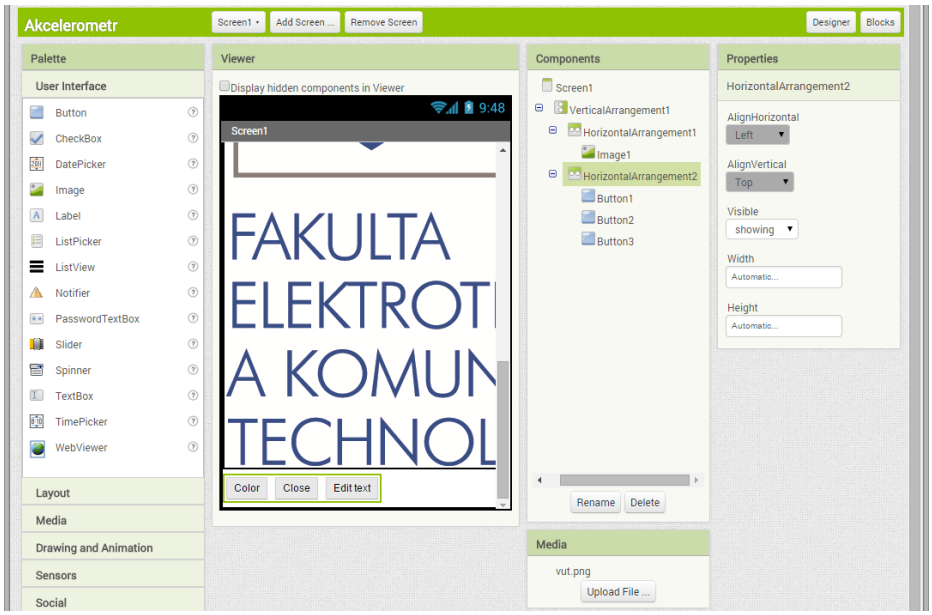
Jako druhé vývojové prostředí bylo zvoleno prostředí App Inventor 2 pro ukázku různých možností vývoje aplikace.

3.3.1 Založení nové aplikace v prostředí App Inventor 2

Po přihlášení do App Invertor 2 stačí kliknout na *Start new project*. Vyskočí okénko s názvem nového projektu, kde se nazve *Akcelerometr* a klikne na *OK*. Poté se otevře návrh grafického rozhraní.

3.3.2 Vytvoření grafického rozhraní hlavní obrazovky

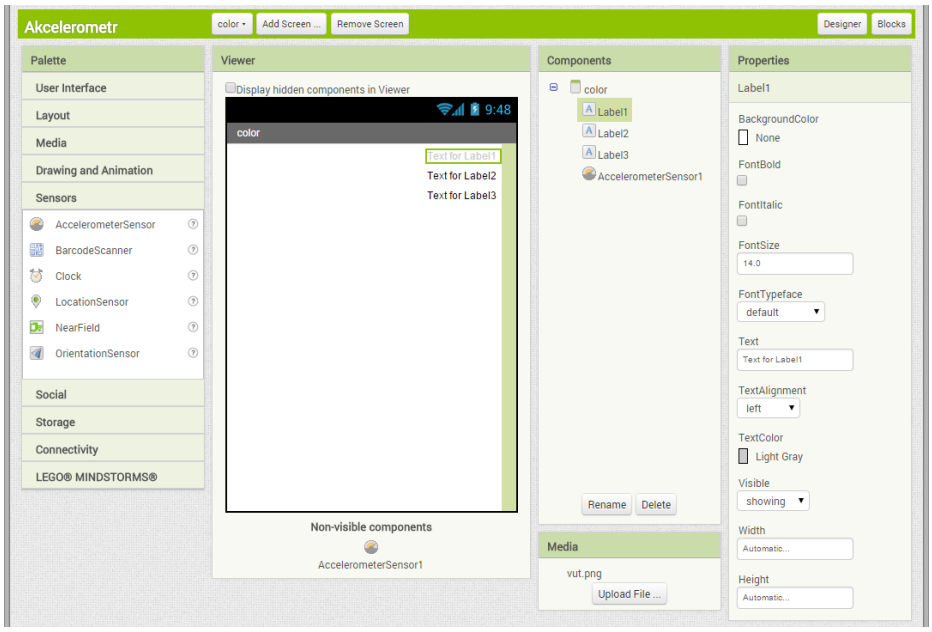
Jako hlavní obrazovku se ponechá *Screen1*. Do ní se z palety přetáhne *VerticalArrangement* ze záložky *Layout*. Do *VerticalArrangement* se přidá *HorizontalArrangement* a do něj *Image* ze záložky *User Interface*. Pod první *HorizontalArrangement* je pak vložen další *HorizontalArrangement* a do něj vedle sebe tři *Button* ze záložky *User Interface*. Jako další se do projektu nahraje obrázek, který se bude zobrazovat na hlavní stránce. Klikne se na *Upload File ...* nacházející se v *Media* hned pod *Components*, vyskočí okno se zvolením umístění obrázku, vybere se obrázek *vut.png* a klikne na *OK*. Klikne se na komponentu *Image1*. V *Properties* se klikne na *Picture* vybere *vut.png* a potvrdí. Poté se v *Properties* nastaví *Width* na *Fill parrent*. Klikne se na *Button1* a v *Properties* se změní v položce *Text* na *Color*. U *Button2* na *Close* a u *Button3* na *Edit text*. Konečný vzhled obrazovky je zobrazen na obrázku níže.



Obrázek 3.9: Rozložení hlavní obrazovky v prostředí App Incentor 2

3.3.3 Vytvoření grafického rozhraní obrazovky pro změnu barvy

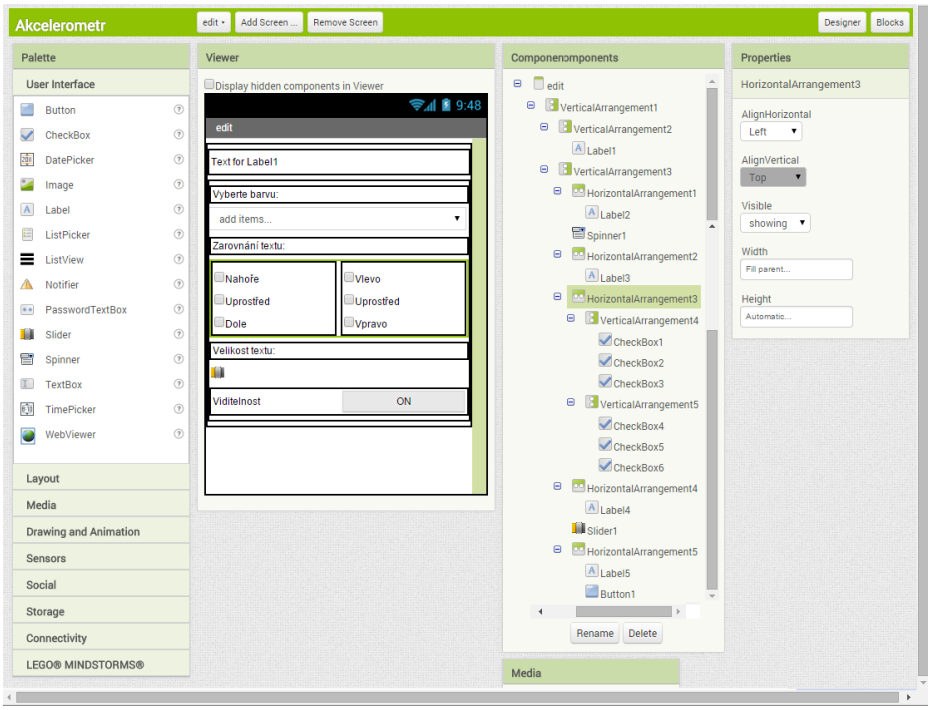
Klikne se na *Add Screen…*, a po zobrazení dialogového okna s názvem se nazve *color* a klikne na *OK*. Do grafického rozhraní se přetáhnou tři *Label* ze záložky *User Interface* a *AccelerometerSensor* ze záložky *Sensors*. Zarovnání textu vpravo se dosáhne tak, že se v *Components* klikne na *color* a v *Properties* se nastaví *AlignHorizontal* na *Right*. Poté se klikne postupně na *Label1*, *Label2*, *Label3* a v *Properies* změní *TextColor* na *Light Gray*. Správně vytvořená obrazovka je na obrázku níže



Obrázek 3.10: Rozložení obrazovky pro změnu barvy v prostředí App Incentor 2

3.3.4 Vytvoření grafického rozhraní obrazovky pro úpravu textu

Klikne se na *Add Screen…*, a po zobrazení dialogového okna s názvem se nazve *edit* a klikne se na *OK*. Do grafického rozhraní se přetáhne *VerticalArrangement*, do něj další *VerticalArrangement* a do něj *Label*. Do prvního *VerticalArrangement* se přidáme třetí *VerticalArrangement* hned pod druhý *VerticalArrangement.* Do třetího *VerticalArrangement* se vloží *HorizontalArrangement*. Do něj se vloží *Label* a změní se jeho text v *Properties* na *Vyberte barvu:*. Pod *HorizontalArrangement* se vloží *Spinner* ze záložky *User Interface*. Klikne se na *Spinner1* a v *Properties* v *ElemetsFromString* se napíší postupně všechny použité barvy oddělené čárkou. V této aplikaci to jsou *Černá, Červená, Žlutá, Zelená, Tyrkysová* a *Modrá*. Poté se do třetího *VerticalArrangement* hned pod první *HorizontalArrangemet* vloží druhý *HorizontalArrangemet* a do něj *Label*, který se přepíše na *Zarovnání textu:*. Pod druhý *HorizontalArrangemet* se vloží třetí *HorizontalArrangemet.* Do něj se vloží vedle sebe dva *VerticalArrangement* a to každého *VerticalArrangement* pod sebe tři *CheckBox* ze záložky *User Inteface*. U všech *CheckBox* v *Properties* se postupně nastaví *Text* na *Nahoře, Uprostřed, Dole, Vlevo, Uprostřed* a *Vpravo*. Pod dva vedle sebe umístěné *VerticalArrangement* se vloží další *HorizontalArrangement*, do něj se vloží *Label* a změní se jeho text na *Velikost textu:*. Pod poslední *HorizontalArrangement* se vloží *Slider* ze záložky *User Interface*, pod něj další *HorizontalArrangement* a do něj vedle sebe *Label*, jehož text se změní na *Viditelnost* a *Button*, jehož text se změní na *ON*. U všech prvků, které se nacházejí v této obrazovce v *Properties* se změní *Width* na *Fill parrent*.

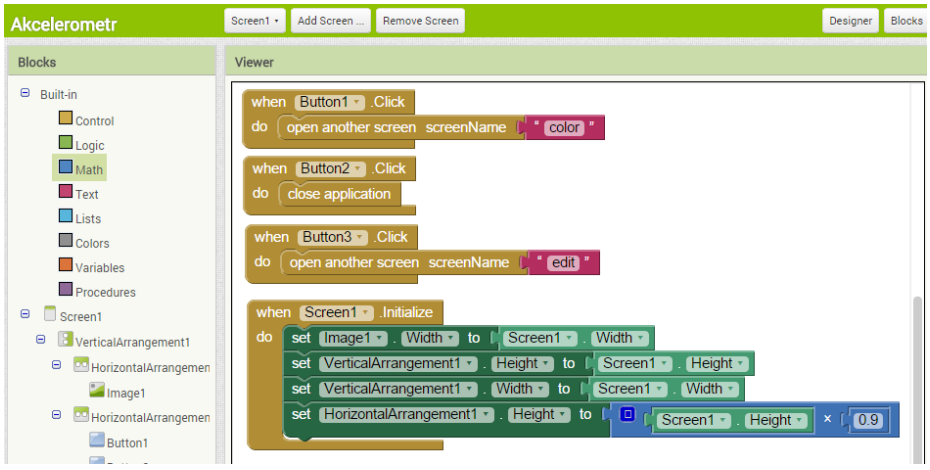


Obrázek 3.11: Rozložení obrazovky pro úpravu textu v prostředí App Incentor 2

3.3.5 Vytvoření Activity hlavní obrazovky v App Inventor 2

Stisknutím rozevíracího seznamu vytvořených obrazovek vedle tlačítka *Add Screen…* se otevře hlavní obrazovka s názvem *Screen1*. Poté se v pravém horním rohu klikne na *Blocks*, čímž se okno přepne do samotného blokového programování aplikace. Jako první se naprogramuje, co se stane po stisknutí jednotlivých tlačítek. V levé části v *Blocks* se klikne na *Button1* do pracovního okna *Viewer* se přetáhne blok *when Button1.Click*. To samé se udělá i s *Button2* a *Button3*. Poté se v *Blocks* klikne na *Control* a do *Viewer* se přetáhne blok *open another screen screenName*. Tento blok se pak přetáhne do *when Button1.Click* hned za *do*. Poté se z *Text* přetáhne blok, kde jsou prázdné uvozovky a připojí se za open *another screen screenName*. Do tohoto bloku se napíše název obrazovky, která se bude po stisknutí tohoto tlačítka otvírat. V tomto případě *color*. Klikne se na *open another screen screenName*, stiskne se Ctrl+C a poté Ctrl+V. Objeví se nám druhý *open another screen screenName* i se vším co do něj bylo připojeno. Přetáhne se do *when Button3.Click* a změní se název obrazovky na *edit*. Poté se z Control přetáhne blok *close application* a připojí se do *when Button2.Click*.

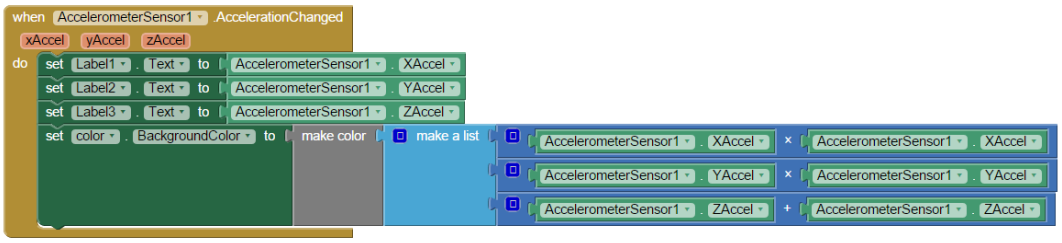
Funkčně už by tato obrazovka fungovala, ale ještě se upraví zobrazení, aby se také správně zobrazovala a dobře vypadala. Klikne se na *Screen1* a přetáhne se blok *when Screen1.Initialize*. Poté se do něj přetáhne *set Image1.Width to* nacházející se v *Image1*. Klikne se na Screen1. Z něj se přetáhne *Screen1.Width* a připojí k *set Image1.Width to*. Tím se zajistí roztažení obrázku přes celou šířku obrazovky. Klikne se na *VerticalArrangement1* a přetáhne z něj *set VerticalArrangement1.Height to* a za něj připojíme *Screen1.Height*. Klikne se na *set VerticalArrangement1.Height to*, zkopíruje se a vloží pod původní. Poté se u tohoto bloku klikne na *Height* a u rozevíracího seznamu na *Width*. Těmato dvěma bloky jsme zajistily roztažení přes celou obrazovku. Klikne se na *HorizontalArrangement1*, přetáhne se *set HorizontalArangement1.Height to* a připojí opět do *when Screen1.Initialize*. Klikne se na *Math* a přetáhne se blok s násobením, který se připojí k poslednímu přidanému bloku. Do prvního okénka v bloku násobení se přetáhne *Screen1.Height* a do druhého blok s číslem z *Math*, které se přepíše na *0.9*. Tím se zajistí, že se tlačítka budou zobrazovat ve spodních 10% obrazovky.



Obrázek 3.12: Zapojení bloků hlavní obrazovky v App Inventor 2

3.3.6 Vytvoření Activity obrazovky pro změnu barvy v App Inventor 2

Stisknutím rozevíracího seznamu vytvořených obrazovek vedle tlačítka *Add Screen…* se otevře hlavní obrazovka s názvem *color*. Klikne se na *AccelerometerSensor1* přetáhne se do *Viewer* blok *when AccelerometerSensor1.AccelerometerChanged*. Z *Label1* se přetáhne *set Label1.Text to* a připojí se do prvního bloku. Klikne se na *AccelerometerSensor1*, vybere se blok *AccelerometerSensor1.XAccel* a připojí k *set Label1.Text to*. Klikne se na blok *set Label1.Text to* a dvakrát se zkopíruje i s *AccelerometerSensor1.XAccel* a připojí se pod původní blok. U kopií se klikne postupně na *Label1*, který se změní na *Label2* u první a *Label3* u druhé a na *XAccel*, která se změní na *YAccel* a *ZAccel*. Poté se klikne na *color* a přetáhne se blok *set color.BackgroundColor to* pod *set Label3.Text to*. Dále se klikne na *Colors* a přetáhne se blok *make color*, který se připojí k *set color.BackgroundColor to*. Čísla, která jsou zapojena do bloku *make a list* se smažou a místo nich se připojí *násobení* z *Math*. V každém bloku násobení se bude násobit stejná osa se stejnou osou. Takže do prvního násobení se přetáhnou dva *AccelerometerSensor1.XAccel*, do druhého *AccelerometerSensor1.YAccel* a do třetího *AccelerometerSensor1.ZAccel*. Tím je tato Activity hotova.

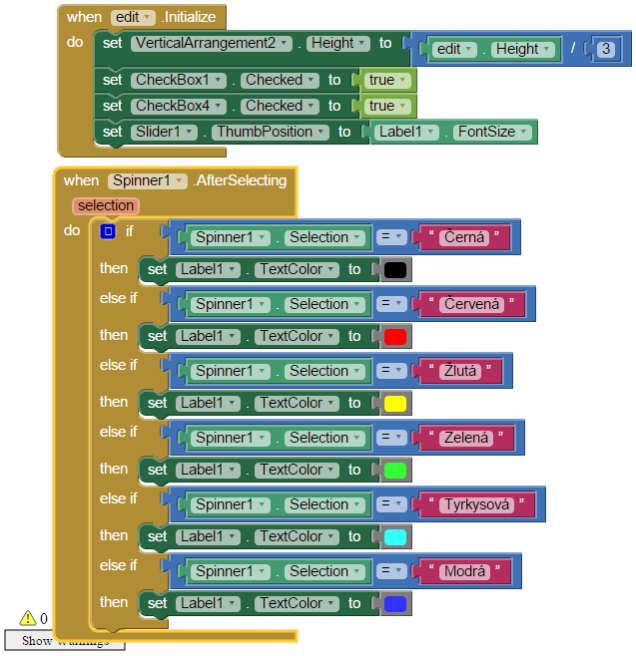


Obrázek 3.13: Zapojení bloků obrazovky pro změnu barvy pozadí v App Inventor 2

3.3.7 Vytvoření Activity obrazovky pro změnu textu v App Inventor 2

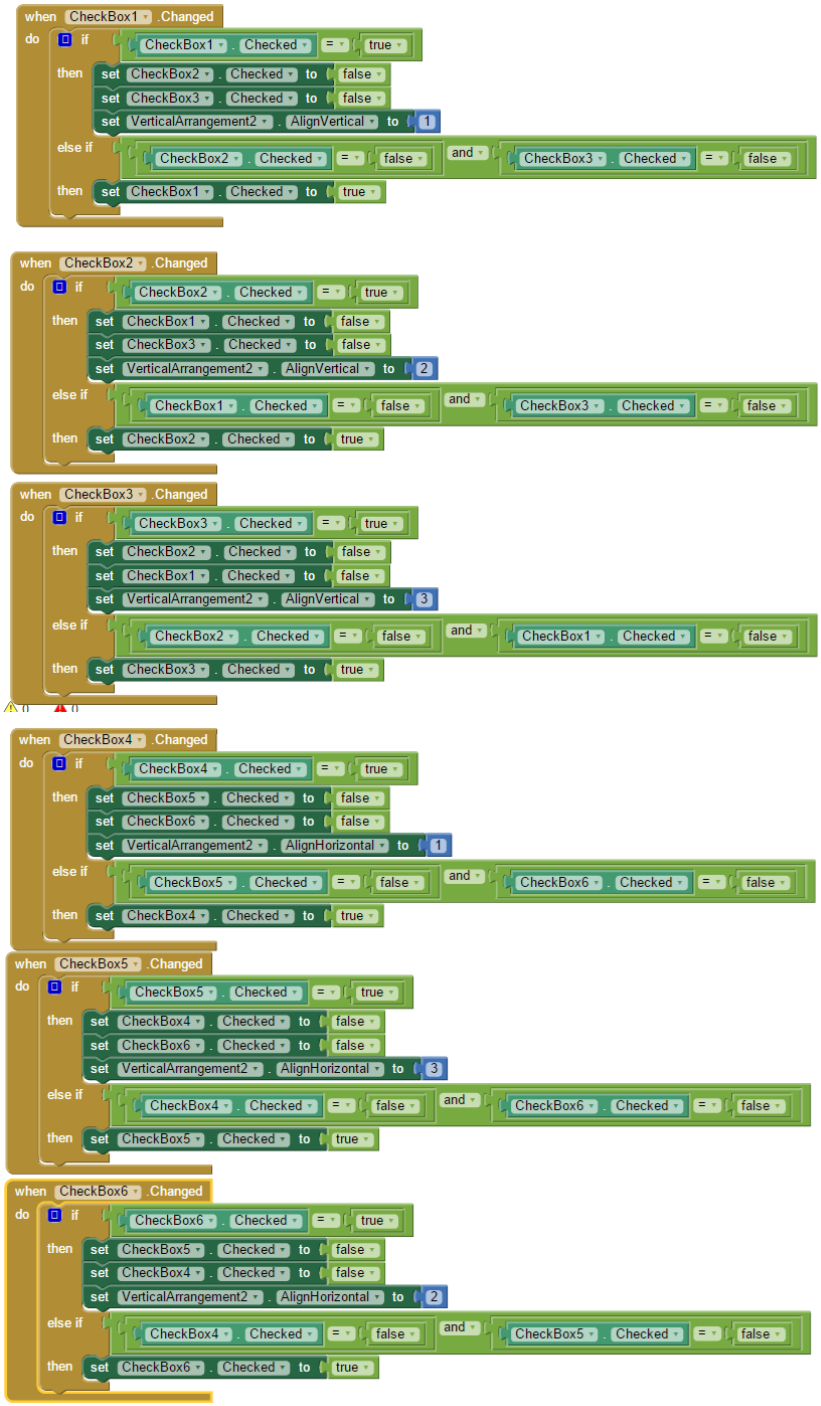
Stisknutím rozevíracího seznamu vytvořených obrazovek vedle tlačítka *Add Screen…* se otevře hlavní obrazovka s názvem *edit*. Zde se jako první začne rozvržením grafického prostředí. Klikne se na *edit* a přetáhne se blok *when edit .Initialize*. Do něj se přetáhne *set VerticalArrangement2.Height to* z *VerticalArramgement2*. K němu se připojí blok *dělení* z *Math*, do prvního okénka se dá *edit.Height* nacházející se v *edit* a číslo z *Math*, které se přepíše na *3*. Poté se do *when edit.Initialize* z *Checkbox1* a *CheckBox4* přetáhne *set CheckBox1.Checked to* a *set CheckBox4.Checked to*. Za tyto dva bloky se připojí blok *true* z *Logic*. Z *Slider1* se přetáhne *set Slider1.ThumbPosition to* a za něj se připojí blok *Label1.FontSize* z *Label1*. Tím je základní rozvržení hotové.

Nyní změna barvy textu. Klikne se na Spinner1 a přetáhne se blok *when Spinner1.AfterSelecting*. Do něj se z *Control* přetáhne blok *if then*. V tomto bloku se klikne na modrý čtvereček. Objeví se okénko, ve kterém se bude přetahovat *else if* z levé strany do *if* na pravé straně. Přetáhne se jich o jednu méně, než kolik je barev, čili pět. K prvnímu *if* se přetáhne blok *rovná se* z *Math*. Do prvního okénka se přetáhne blok *Spinner1.Selection* ze *Spinner1*, do druhého se přetáhnou *uvozovky* z *Text* a do uvozovek se napíše název barvy úplně stejně, jak se psal při návrhu grafického rozhraní. Do *then* nacházející se pod *if* se přetáhne blok *set Label1.TextColor to* z *Label1* a do něj se zapojí blok se čtverečkem *barvy* z *Colors*. Takto se to opakuje i u ostatních barev.



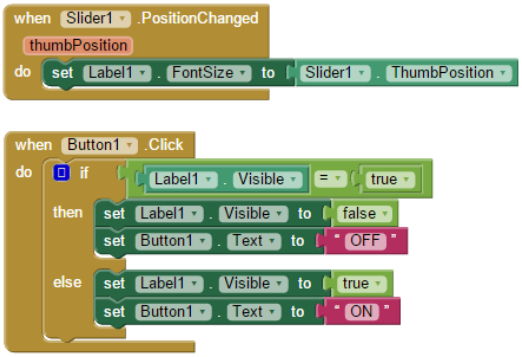
Obrázek 3.14: Bloky pro změnu barvy textu v App inventor 2

Teď k zarovnání textu. Klikne se na CheckBox1 a přetáhne se *when CheckBox1.Changed*. Do tohoto bloku se přetáhne *if then* z *Control* a pomocí modrého čtverečku se rozšíří o jedno *else if*. V prvním *if* se bude testovat, jestli došlo k zaškrtnutí CheckBox1 a pokud ano ostatní dva bloky se odškrtnou. To se udělá tak, že se do *if* přetáhne *rovná se* z *Logic*, do jež prvního okénka se přetáhne *CheckBox1.Checked* z *Checkbox1* a do druhého *true* z *Logic*. Do *then* pod *if* se poté přetáhne *set CheckBox2.Checked to* z *CheckBox2* a *set CheckBox3.Checked to* z *CheckBox3*. Za tyto dva bloky se připojí *false* z *Logic* a pod tyto dva bloky se připojí *set VerticalArrangement.AlignVertical to* a do něj *číslo* z *Math*, které se přepíše na *1*. tím se docílí zarovnání vlevo. Nyní se udělá to, aby šel *CheckBox1* odškrtnou pouze stisknutí některého z dalších dvou *CheckBox*. Do *else if* se přetáhne *and* a do každého okénka v *and* se přetáhne *rovná se* z *Logic*. Do prvního okénka se u prvního *rovná se* přetáhne *CheckBox2.Checked* z *CheckBox2* a do druhého *false* z *Logic*. U druhého *rovná se* se do prvního okénka přetáhne *CheckBox3.Checked* z *CheckBox3* a do druhého *false* z *Logic*. Do *then* se pod *else if* přetáhne *set CheckBox1.Checked to* z *CheckBox1* a za něj se připojí *true* z *Logic*. Stejnou logikou se ošetří i ostatní *CheckBox* s tím, že zarovnání uprostřed je 2 a zarovnání napravo 3. U horizontálního zarovnání se použije místo bloku *set VerticalArrangement.AlignVertical to* blok *set VerticalArrangement.AlignHorizontal to* s tím, že zarovnání nahoru je 1, uprostřed 3 a dolů 2.



Obrázek 3.15: Zarovnání textu pomocí CheckBox v App Inventor 2

Následuje velikost textu a viditelnost textu. Přetáhne se *when Slider1.positionChanged* z *Slider1*, do něj se přetáhne blok *set Label1.FontSize to* z *Label1* a za něj se připojí *Slider1.ThumbPosition* ze *Slider1*. Klikne se na *Button1* a přetáhne se *when Button1.Click*. Do něj se vloží blok *if else* z *Control*, který se pomocí modrého čtverečku rozšíří o *else*. Za *if* se připojí *rovná se* z *Logic* do jehož prvního okénka se přetáhne *Label1.Visible* z *Label1* a do druhého *true* z *Logic*. K *then* se připojí *set Label1.Visible to* z *Label1* a *set Button1.Text to* z *Button*. K *set Label1.Visible to* se připojí *false* z *Logic* a za *set Button1.Text to* uvozovky z *Text*, které se přepíše na *OFF*. Bloky *set Label1.Visible to* a *set Button1.Text to* se zkopírují i s připojenými bloky, připojí se k *else*, klikne se na *false* a změní se na *true*. Text v uvozovkách se přepíše na *ON*.



Obrázek 3.16: Velikost a viditelnost textu v App Inventor 2

Nyní je aplikace hotová a po kliknutí na Build > App(provide QR code for .apk) se aplikace sestaví a na obrazovce se objeví QR kód s odkazem na stažení této aplikace.

4 NÁVRH KOMUNIKAČNÍ SÍTĚ

Tato část je věnována návrhu aplikace bezdrátové komunikační sítě za použití zařízení s operačním systémem Android jako řídící centrály celé sítě. Jako ukázkový příklad bezdrátové komunikace byl zvolen návrh chytrého domu. Chytrý dům byl zvolen z důvodu postupného nástupu chytrých zařízení a z důvodu rozrůstajícího se internet věcí. Z důvodu praktického aspektu budou v této práci brány zařízení, které jsou běžně dostupné, a které omezí potřebné zásahy do již existující infrastruktury obydlí.

4.1 Komunikační síť

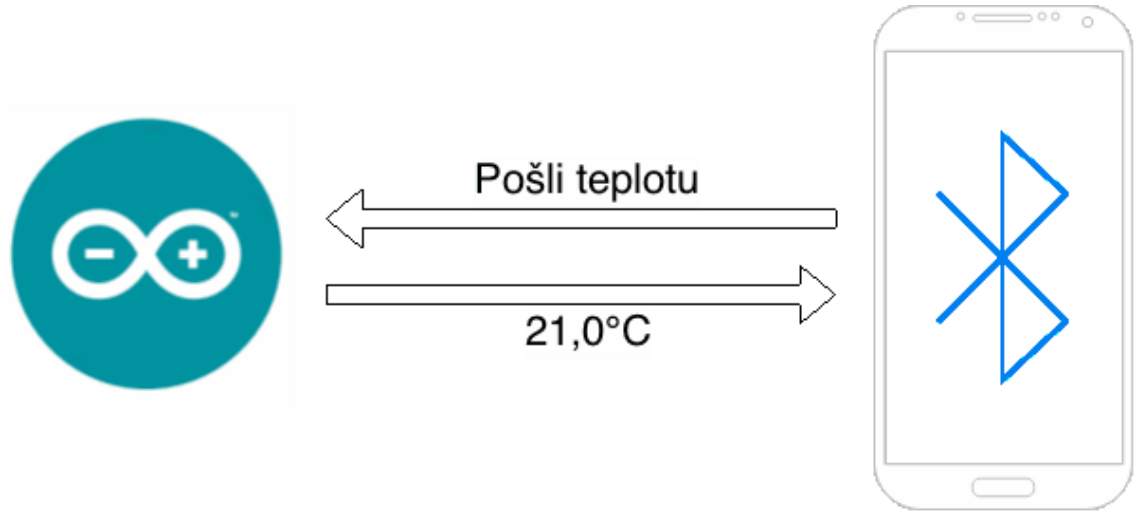
V hypotetickém obydlí bude komunikační síť složena z několika druhů modulů. Nejjednodušší částí systému jsou pasivní moduly, které provádí pouze určitou činnost na základě přijaté zprávy. Například odesílání naměřených dat ze senzorů do centrály. O něco složitější jsou aktivní moduly, které mohou své chování v průběhu chodu systému měnit a obstarat základní automatizované nastavení ovládacího prvku. Jelikož se při návrhu počítá s integrací systému do již existující infrastruktury bez nutnosti jakýchkoliv, případně s minimálními úpravami budou některé moduly napájeny pouze z baterie, a proto je při výběru komponent a komunikace kladen důraz na spotřebu.

4.1.1 Výběr komunikačního standardu

Zařízení s operačním systémem bývají prakticky všechny vybaveny stejným druhem zařízení umožňující bezdrátovou komunikace, mezi které patří například Bluetooth, Wi-fi, NFC, a některé i infraport. Jako nejlepší řešení se zdá být komunikace prostřednictvím Bluetooth, které má pro chystané použití dostatečnou přenosovou rychlost a malou spotřebu. Při použití Wi-Fi může být sice větší přenosová rychlost, ale má také větší spotřebu. Další bezdrátová komunikace NFC je použitelná pouze do několika málo desítek centimetrů a použití infraportu je podmíněno přímou viditelností. [25]

4.1.2 Příklad pasivního modulu

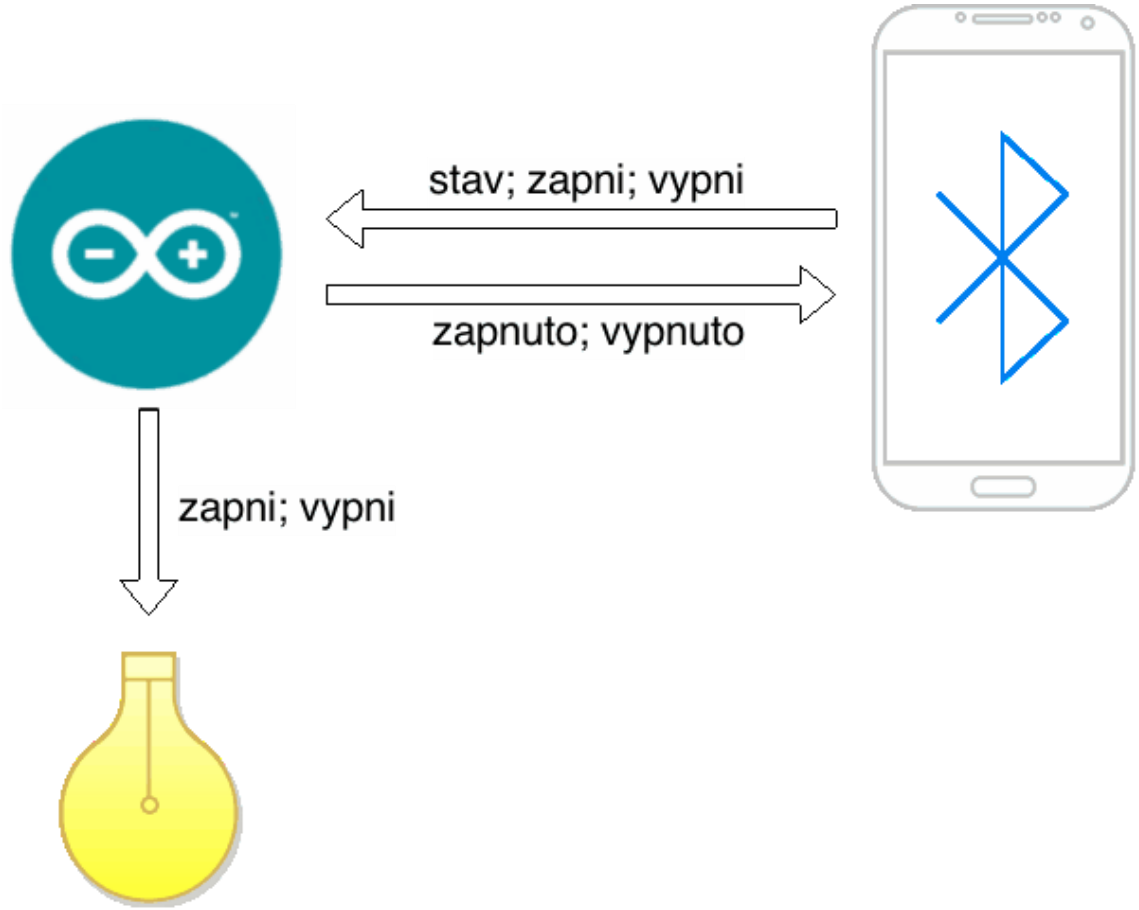
Jedna z věcí, které by chytrý dům měl zvládat je zajistit optimální vytápění. Proto byl jako jeden z ukázkových pasivních modulů vytvořen modul pro monitorování teploty jako ukázka pasivního modulu shromažďující data a odesílající je do centrály. Jako další pasivní modul byl vytvořen modul pro automatické ovládání žaluzií jako ukázka ovládání spotřebičů z centrály. Modul bude sloužit k samotnému ovládání žaluzií a bude řízen pokyny z centrály



Obrázek 4.1: Komunikace s pasivním modulem pro snímání teploty

4.1.3 Příklad aktivního modulu

Jako příklad aktivního modulu byl vytvořen dvojmodul pro ovládání osvětlení. Tento modul zvládá nejen provádět určitou činnost na základě pokynu z centrály, ale i změnu běhu modulu na základě vnějšího podmětu bez nutnosti zásahu centrály. V tomto případě zapnutí světla ihned po sepnutí spínače bez nutnosti čekání na připojení centrály a vyhodnocení stavu. Informace o stavu světla se poté odešle do centrály hned po jejím připojení. Základem dvojmodul bude opět Arduino. První modul obsahuje spínač, Bluetooth modul pro komunikaci s centrálou a další bezdrátový modul pro komunikaci a řízení druhého modulu. Druhý modul je kromě Arduina složen i z bezdrátového modulu pro řízení z prvního modulu a z relé ovládajícího světla na výstupu.



Obrázek 4.2: Komunikace s aktivním modulem pro zapínání osvětlení

4.1.4 Princip činnosti bezdrátové sítě

Pro přenos informačních dat mezi jednotlivými prvky vyla zvolen systém vycházející z hvězdicové topologie, díky jejím nesporným výhodám. Hvězdicová topologie je snadno rozšiřitelná, snadno se dají nalézt závady, a pokud zkolabuje jeden prvek tak mohou ostatní bez problémů pokračovat v činnosti dále. Celá síť funguje tak, že se centrála připojí vždy k jednomu danému konkrétnímu bodu v síti, například k modulu pro snímání teploty, od kterého získá data. Po získání dat se tyto data vyhodnotí a poté se Android od tohoto bodu odpojí a připojí se na další bod pro získání a vyhodnocení dalších dat. Tato metoda přináší jednu nespornou výhodu. Protože moduly nejsou připojeny pořád, a protože stačí, aby byly aktivní pouze o něco málo delší dobu, než je potřebná k odeslání dat do centrály, může být zařízení po celou zbylou dobu v režimu spánku, což přináší úsporu energie.

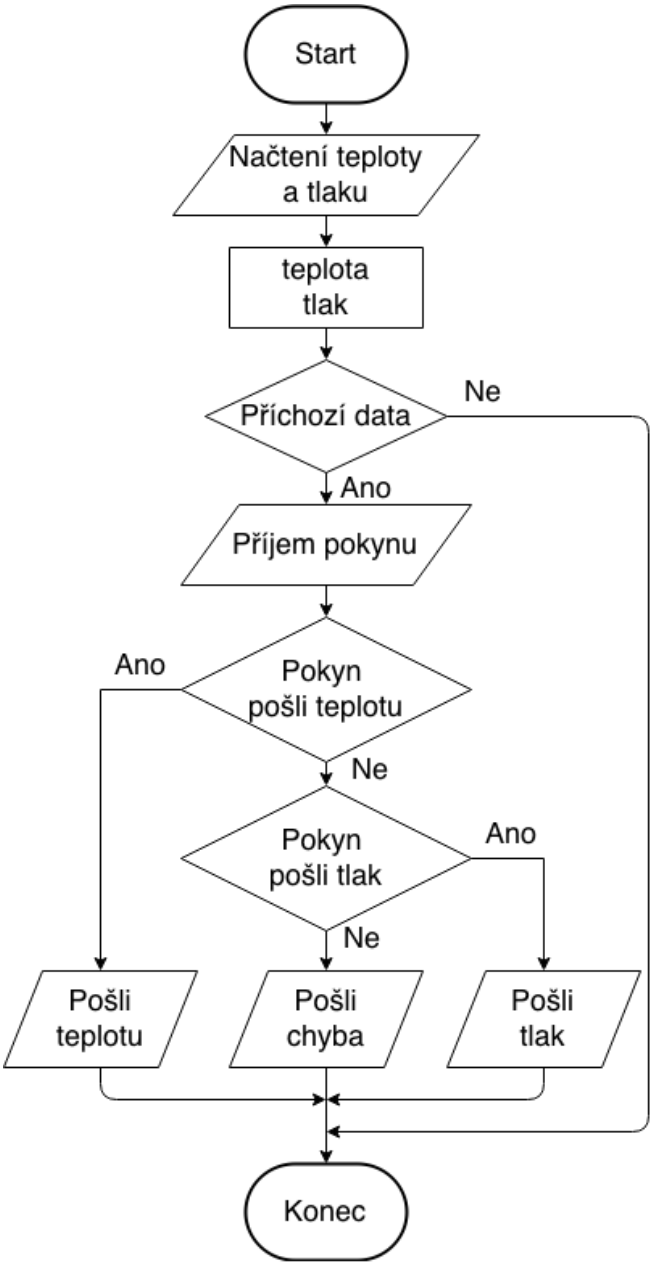
Obrázek 4.3: Komunikace v rámci bezdrátové sítě

4.2 Realizace modulů

Jako základní programovatelné obvody modulů byly vybrány programovatelné obvody Arduino pro svou rozšířenost, cenu, možnosti využití a možnost uspání v případě nevyužití. Programovatelné obvody jsou podle potřeby doplněny o bezdrátové moduly a potřebné senzory. Jako poslední prvek bude komunikační síť obsahovat centrálu, která bude obsluhovat celý systém. V tomto případě se jedná o mobilní zařízení s operačním systémem Android.

4.2.1 Modul pro snímání teploty a tlaku

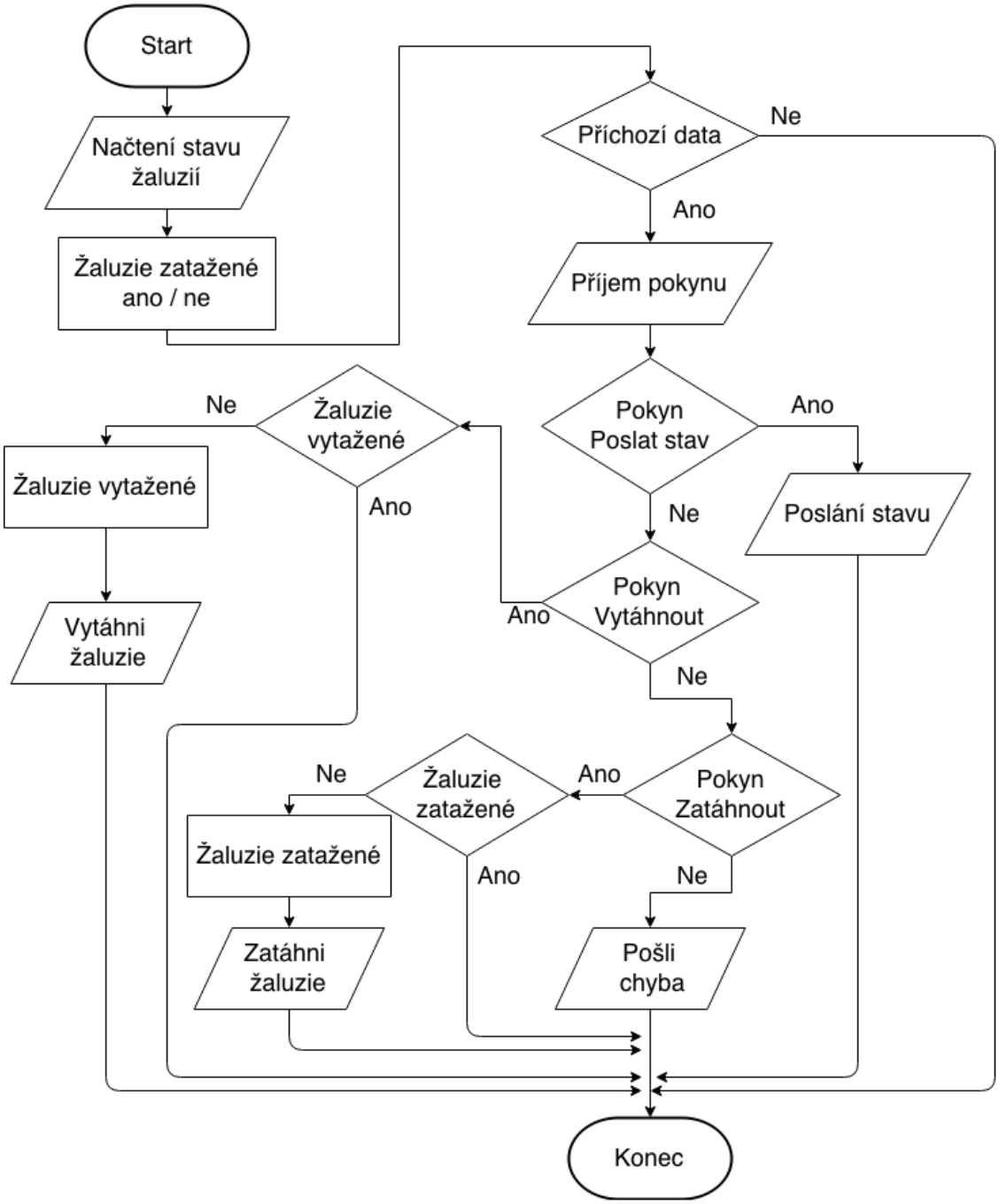
Modul pro snímání teploty a tlaku je pasivní modul, který po zaslání pokynu z centrály odešle zpět aktuální teplotu nebo atmosférický tlak. Jako základ je požit programovatelný obvod Arduino, u kterého je využita jeden I2C port pro komunikaci s teplotním čidlem a jeden UART port pro komunikaci přes Bluetooth. Jako teplotní senzor byl zvolen senzor BMP 180 od firmy Bosch. Při zapnutí se jako první načte teplota. Poté se střídá kontrolování, zda nepřišel příkaz a aktualizace teploty a tlaku, aby při dotazu mohly být ihned poslány informace. Podle povelu se následně zjistí buď aktuální teplota, nebo čas a odešle se do centrály, jak je popsáno v následujícím diagramu.



Obrázek 4.4: Vývojový diagram modulu pro snímání teploty a tlaku

4.2.2 Modul ovládání žaluzií

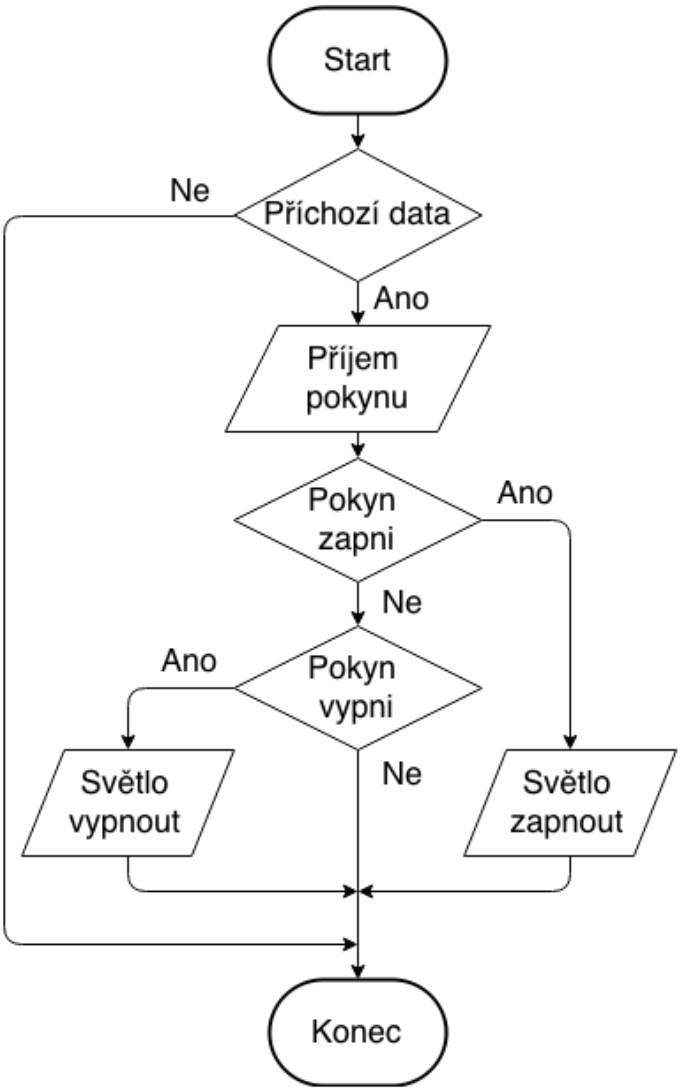
Modul pro ovládání žaluzií je pasivní modul, který přijímá pokyny z centrály a podle těchto pokynů zatáhne nebo odtáhne žaluzie. Jako základ je požitý programovatelný obvod Arduino, u kterého jsou využity čtyři digitální výstupy pro řízení krokového motorku a jeden digitální výstup pro senzor zatažení. Dále je využitý jeden UART port pro komunikaci s Bluetooth modulem a zařízením Android. Při zapnutí se jako první zkontroluje, zda jdou žaluzie zatažené nebo vytažené. Poté modul kontroluje, zda přišel příslušný povel. Na základě tohoto povelu modul buď pošle zpět informaci o stavu žaluzií, nebo je vytáhne či zatáhne, pokud to není v konfliktu s jejich předchozím stavem. Tedy aby se žaluzie nezačali znovu vytahovat, pokud jsou již vytažené a nedošlo tak k jejich poškození. Logika modulu je uvedena na následujícím vývojovém diagramu.



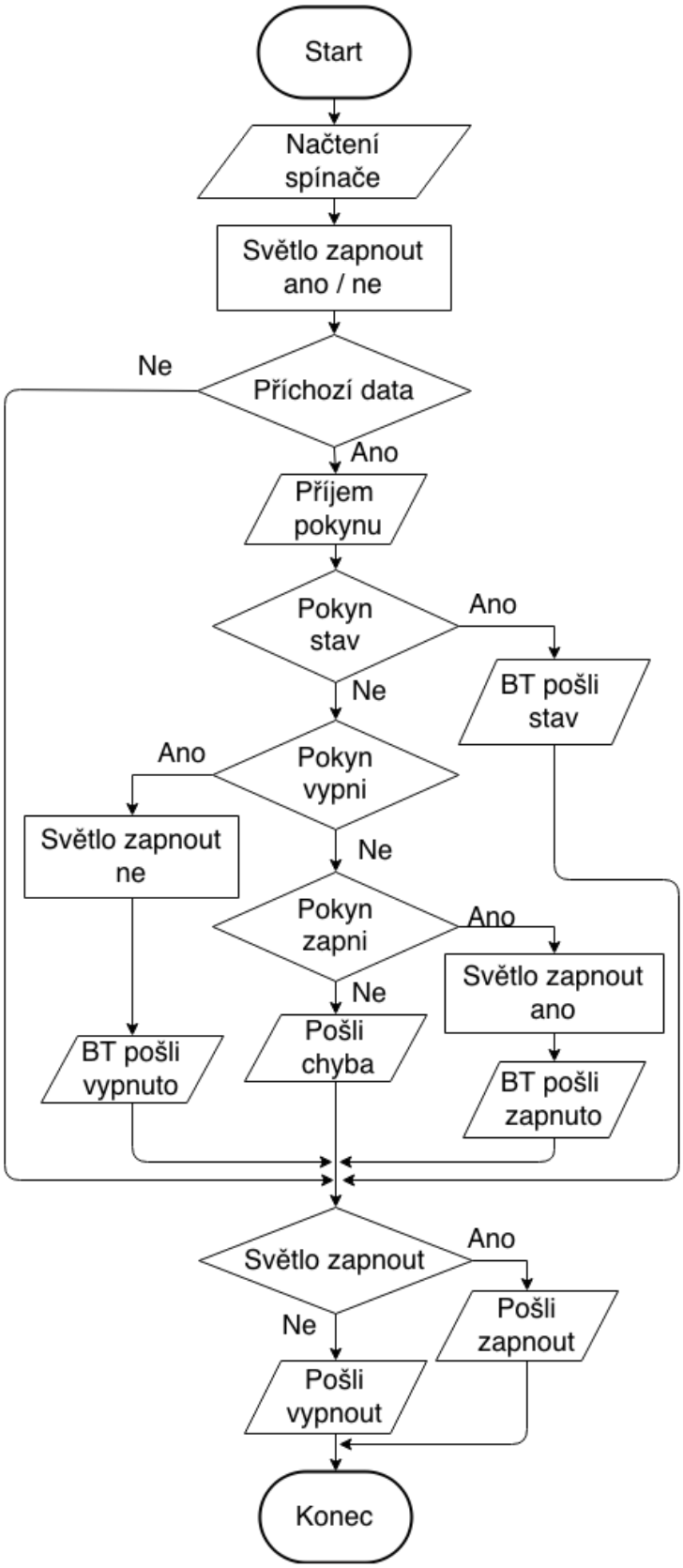
Obrázek 4.5: Vývojový diagram modulu pro ovládání žaluzií

4.2.3 Modul pro ovládání světla

Modul pro ovládání světla je aktivní modul, který nejen přijímá pokyny z centrály, ale sám dokáže vyslat pokyny pro rozsvícení světla. Tento modul je složen ze dvou částí. První část je složena z Arduina, přijímacího modulu a relé. U Arduina je vyžitý jeden digitální vstup pro připojení přijímacího modulu a jeden digitální výstup pro spínání světla. Po zapnutí modul zkontroluje, zda nepřišel pokyn. Poté tento pokyn vyhodnotí a na jeho základě sepne nebo rozepne ovládací relé. Druhá ze dvou částí je složena z Arduina, spínače, vysílacího modulu a Bluetooth. U Arduina je vyučit jeden digitální vstup pro vypínač, jeden digitální výstup pro posílání instrukcí přes bezdrátový modul a jeden UART port pro komunikaci přes Bluetooth. Při zapnutí jako první modul zkontroluje, zda byl nebo nebyl sepnut spínač. Poté se kontroluje, jestli nepřišel pokyn a jako poslední se po vyhodnocení stisknutí spínače i poslaného pokynu odešle informace o zhasnutí či vypnutí do druhé části modulu. Logika modulu je popsána na následujícím vývojovém digramu.



Obrázek 4.6: Vývojový diagram pro první část modulu pro zapínání světla



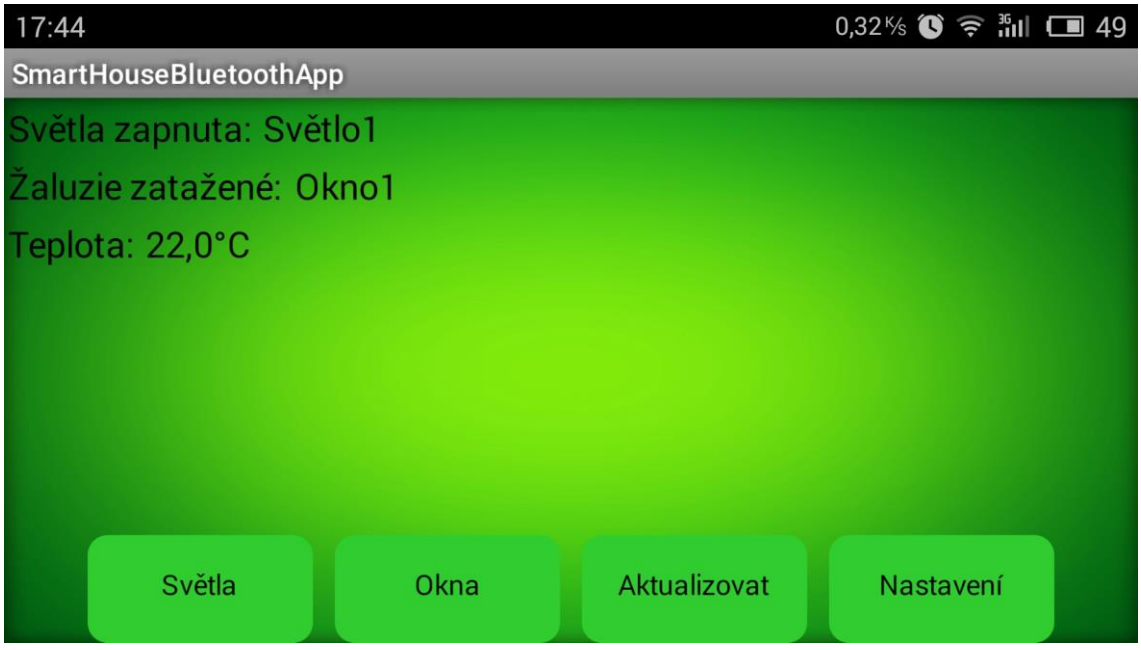
Obrázek 4.7: Vývojový diagram druhé části modulu pro ovládání světla

4.3 Realizace centrály pro ovládání sítě

Celá síť chytrého domu je ovládána z centrály, která představuje zařízení s operačním systémem Android a příslušnou aplikaci. Aplikace pro ovládání byla vytvořena ve vývojovém prostředí App Inventor 2. Toto vývojové prostředí má sice nevýhodu v podobě vlastního kompilátoru, který poskytuje horší optimalizaci, aplikace má větší spotřebu a občas způsobí i pád aplikace, ale také má ohromné výhody díky obsahu všech potřebných komponentů, které jsou zároveň i lehce použitelné. Také má výhodu v podobě jednoduchého a rychlého vytváření a neustálého předělávání vytvářené experimentální aplikace. Vytvořená aplikace byla nazvána SmartHouseBluetoothApp.

4.3.1 Popis Aplikace

Aplikace pro ovládání chytrého domu se skládá ze čtyř obrazovek. Hlavní obrazovka obsahuje základní informace o stavu domu. Jsou to informace o zapnutých světlech, zatažených žaluziích a teplotě. Dále hlavní obrazovka čtyři tlačítka jak je vidět níže. První tlačítko slouží k otevření obrazovky *Lights*, kde je možné ovládat jednotlivá světla. Druhé tlačítko otevře obrazovku pro ovládání oken. Třetí tlačítko slouží pro aktualizaci základních informací a čtvrté tlačítko slouží zobrazení obrazovky s nastavením.

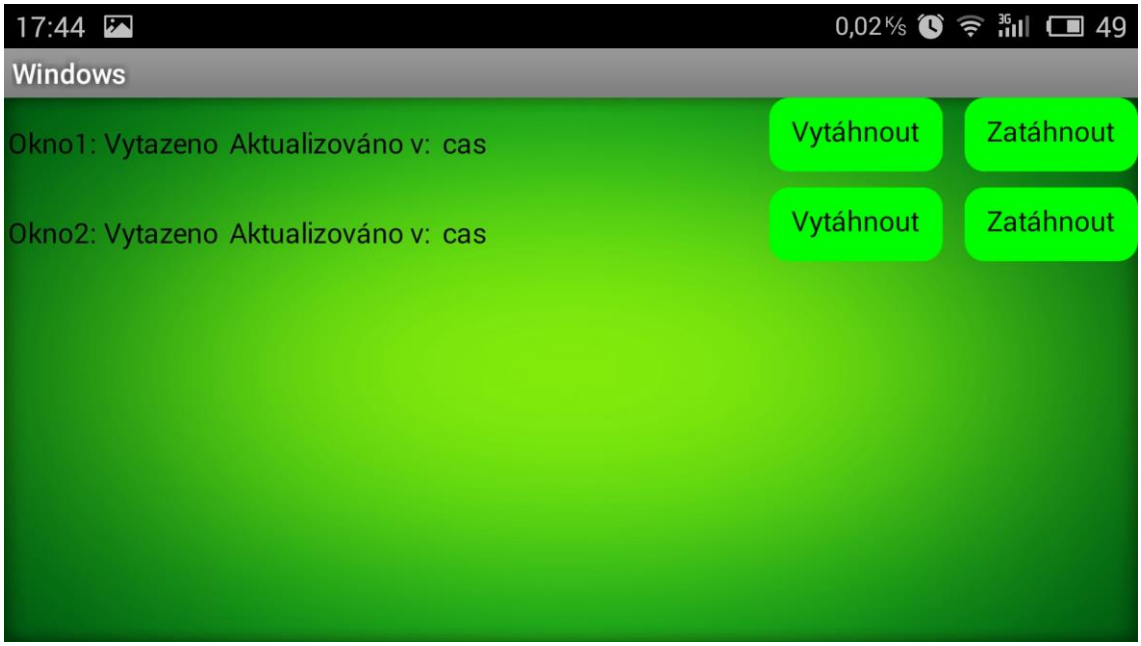


Obrázek 4.8: SmartHouseBluetoothApp hlavní obrazovka

Po kliknutí na tlačítko *Světla* se zobrazí obrazovka *Lights* pro ovládání světel. Při každé inicializaci obrazovky vyskočí vyskakovací okno, které se ptá, zda je potřeba aktualizovat stav světel. Pokud je kliknuto na *Ano* aplikace aktualizuje stav u všech světel, které má v databázi a poté vjede na ovládací obrazovku. Pokud zvolíme *Ne,* vjede aplikace přímo na ovládací obrazovku. Zde jsou zobrazeny informace o stavu světla, času poslední akce a dvě ovládací tlačítka pro zapnutí a vypnutí. Jeden řádek vždy odpovídá jednotlivému světlu, jak je vidět na obrázku níže.

Obrázek 4.9: SmartHouseBluetoothApp obrazovka pro ovládání světel

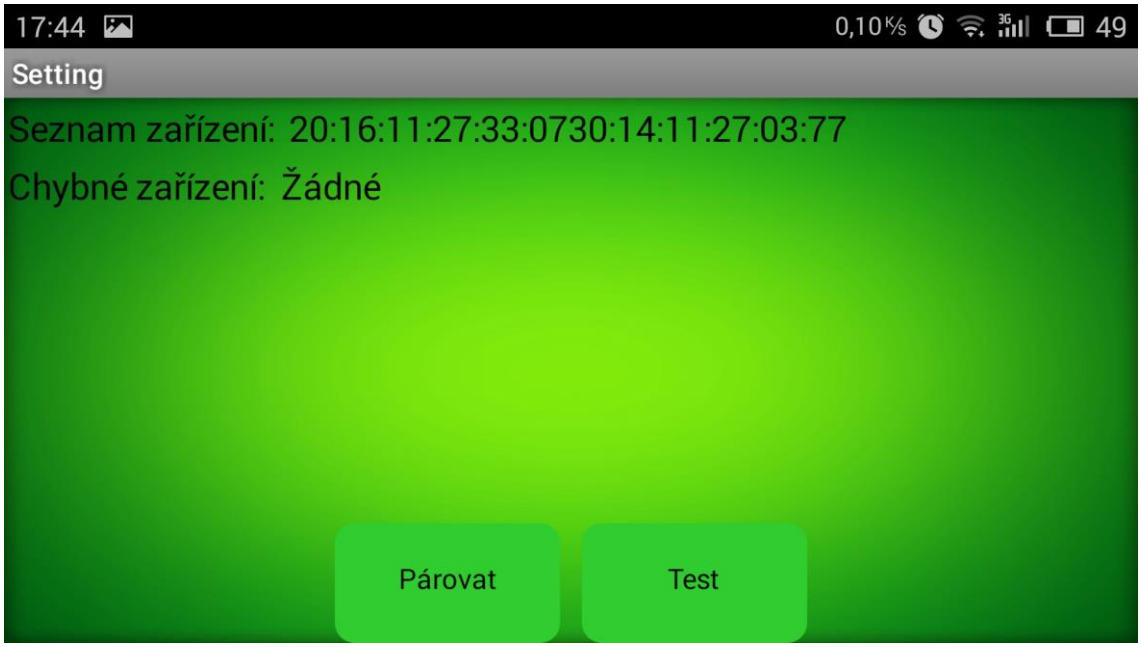
Po kliknutí na tlačítko *Okna* se otevře obrazovka *Windows* pro ovládání zatemnění oken. Při každé inicializaci obrazovky vyskočí vyskakovací okno, které se ptá, zda je potřeba aktualizovat stav zatažení oken. Jde o stejný princip jako u předchozí obrazovky *Lights*. Také stejně jako u předchozí obrazovky jsou zde zobrazeny informace o stavu zatemnění oken, času poslední akce a dvě ovládací tlačítka pro vytáhnutí a zatáhnutí žaluzií. Jeden řádek vždy odpovídá jednotlivému oknu, jak je vidět na obrázku níže.



Obrázek 4.10: SmartHouseBluetoothApp obrazovka pro ovládání zatemnění oken

Třetí tlačítko *Aktualizovat* slouží pro aktualizaci informací na hlavní obrazovce. Jsou to informace o tom, která světla jsou momentálně zapnutá, která okna zatažená a také jaká je aktuální teplota.

Po stisknutí čtvrtého tlačítka s názvem *Nastavení* se otevře obrazovka *Setting* sloužící pro základní nastavení. Tato obrazovka obsahuje základní informace o modulech, jako je seznam všech modulů v databázi či zobrazení nefunkčního modulu a dvě tlačítka. První tlačítko *Párovat* slouží pro najetí do nastavení telefonu pro párování telefonu a Bluetooth modulu, které musí být napřed spárovány. Druhé tlačítko *Test* slouží pro testování připojení jednotlivých modulů. Po jeho stisknutí se centrála postupně připojí ke všem modulům obsažených v databázi a vyzkouší komunikaci. Moduly, ke kterým se nepodaří připojit nebo nepošlou zpět odpověď, se vyhodnotí jako chybné zařízení a vypíše se pod seznam všech modulů.



Obrázek 4.11: SmartHouseBluetoothApp obrazovka pro nastavení

5 ZÁVĚR

Jako první cíl bylo seznámení s operačním systémem Android, coby relativně novou platformou, a se základní strukturou aplikace. Při seznámení s operačním systém Android bylo dobré si uvědomit, že věci, které již považujeme v chytrých mobilních telefonech za samozřejmost, zde nebyly vždy a že každá nová vydaná verze přináší i nové možnosti vývoje aplikací. Také při seznámení se strukturou aplikace bylo dobré zjistit, že aplikace ve skutečnosti nefunguje jako jeden celek, ale jako jednotlivé byť mezi sebou provázané komponenty.

Druhým stanoveným cílem byl výběr vývojových prostředí. Jelikož je operační systém Android open-source, jsou hlavní vývojové prostředky volně dostupné. Jako hlavní vývojové prostředí bylo napřed zvoleno Eclipse, jež označil i Google za hlavní vývojové prostředí pro operační systém Android. Během vytváření této práce ale Google 9. prosince 2014 po roce a půl vývoje vydal vlastní vývojové prostředí Android Studio a Eclipse upravené od Google se již nedá stáhnout. Proto musela být tato práce přepracována pro vývojové prostředí Android Studio. Nevýhoda Android Studia je, že je v rané verzi 1.0, a i když je vše funkční tak to ještě není dokončené. Například možnost nainstalovat zvlášť vývojové prostředí a zvlášť emulátor, což se později ukázalo chybou, protože při nainstalováním každé části na jiný disk se stala simulace aplikace nefunkční. Jako doplňkové vývojové prostředí bylo vybráno grafické vývojové prostředí App Inventor 2 pro ukázku různých možností vývoje.

Další cíl této práce bylo vyvinout ukázkovou aplikaci pro seznámení s problematikou vývoje aplikací. Aplikace obsahuje základní prvky grafického rozhraní jako například tlačítka, seznam, posuvnou lištu, přepínač, text a také ukázku práce se senzory, konkrétně s akcelerometrem. Zde hodně pomáhala dobře zpracovaná oficiální dokumentace.

V neposlední řadě bylo za úkol vytvořit ukázkovou komunikačních sítí pro chytrý dům, a která je řízena zařízením s operačním systémem Android. Pro chytrý dům bylo uvažováno s několika komunikačními protokoly, ale nakonec bylo zvoleno Bluetooth díky své nízké spotřebě, přímému spojení, velkému výběru a nízké ceny modulů. Síť obsahuje celkem čtyři prvky, z nichž jeden je zařízení s Androidem jako centrála pro a ostatní jsou moduly zajišťující snímání a řízení. Pro Android byla vytvořena ovládací aplikace ve vývojovém prostředí App Inventor 2. Během vytváření byla síť i postupně testována. Nejlepších výsledků bylo dosaženo při použití stejné verze Bluetooth 2.0 u modulu i zařízení Android. Tato síť má velký potenciál k budoucímu rozvoji a vylepšení, protože i velké firmy jako Google začínají vyvíjet zařízení pro internet věcí jako například chytrý termostat Nest. Také Bluetooth je v nejnovějších verzích přizpůsobování pro internet věcí jak spotřebou, tak i množstvím připojených zařízení.

Přínosem této práce je především úvod do programování aplikací pro operační systém Android i těm, kteří nemají s programováním zkušenosti a také i vzorová aplikace, jež demonstruje princip vývoje. Dále také ukázka komunikační sítě chytrého domu, která ukazuje, možností ovládání domu i se základními znalostmi programování.

LITERATURA

[1] Portál Wikipedia.org. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://cs.wikipedia.org/wiki/Android_(opera%C4%8Dn%C3%AD_syst%C3%A9m)

[2] Portál Wikipedia.org. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://en.wikipedia.org/wiki/Android_version_history

[3] Portál Wikipedia.org. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://en.wikipedia.org/wiki/Android_Lollipop

[4] Portál Wikipedia.org. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://en.wikipedia.org/wiki/Just-in-time_compilation

[5] Portál Source.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: https://source.android.com/devices/tech/dalvik/art.html

[6] VÁVRŮ, Jiří a Miroslav UJBÁNYAI. *Programujeme pro Android*. 2. rozš. vyd. Praha: Grada, 2013, 250 s. Průvodce. ISBN 978-80-247-4863-4.

[7] Portál Developer.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://developer.android.com/guide/components/index.html

[8] Portál Developer.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://developer.android.com/guide/components/intents-filters.html

[9] Portál Developer.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://developer.android.com/guide/topics/providers/content-providers.html

[10] Portál Developer.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://developer.android.com/reference/android/app/Activity.html

[11] Portál Developer.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://developer.android.com/guide/components/services.html

[12] Portál Developer.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://developer.android.com/guide/topics/manifest/manifest-intro.html

[13] Portál Wikipedia.org. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://cs.wikipedia.org/wiki/JDK

[14] Portál Developer.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://developer.android.com/sdk/installing/studio.html

[15] Portál Wikipedia.org. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://en.wikipedia.org/wiki/App_Inventor_for_Android

[16] WOLBER, David, Hal ABELSON, Ellen SPERTUS a Liz LOONEY. *App inventor*. 1. vyd. Brno: Computer Press, 2014, 368 s. ISBN 978-80-251-4195-3.

[17] Portál Wikipedia.org. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://cs.wikipedia.org/wiki/Eclipse_(v%C3%BDvojov%C3%A9_prost%C5%99ed%C3%AD)

[18] Portál Wikipedia.org. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://en.wikipedia.org/wiki/Scripting_Layer_for_Android

[19] Portál Google.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: https://code.google.com/p/android-scripting/

[20] Portál Microsoft.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://msdn.microsoft.com/cs-cz/library/dn771552.aspx

[21] Portál Wikipedia.org. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://upload.wikimedia.org/wikipedia/commons/a/af/Android-System-Architecture.svg

[22] Portál Developer.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://developer.android.com/images/components/intent-filters@2x.png

[23] Portál Developer.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://developer.android.com/images/activity_lifecycle.png

[24] Portál Developer.android.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://developer.android.com/images/service_lifecycle.png

[25] Bluetooth. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-05-21]. Dostupné z: http://cs.wikipedia.org/wiki/Bluetooth

Významné zdroje použité pro programování:

[26] Portál Javacodegeeks.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://examples.javacodegeeks.com/android/core/hardware/sensor/android-accelerometer-example/

[27] Portál Appsrox.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://www.appsrox.com/android/tutorials/accelerometer-golf/

[28] Portál Compiletimeerror.com. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://www.compiletimeerror.com/2013/09/android-spinnersdropdownlist-tutorial.html#.VILtdjGG8wM

[29] Using the BMP085. Adafruit [online]. 2012 [cit. 2015-05-21]. Dostupné z: https://learn.adafruit.com/bmp085/using-the-bmp085-api-v2