VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYUŽITÍ SENZORŮ MOBILNÍHO ZAŘÍZENÍ V AUTOMOBILU

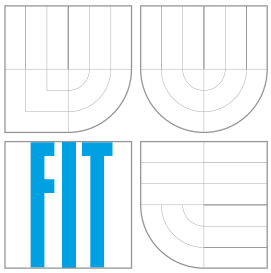
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. LUKÁŠ ARON

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYUŽITÍ SENZORŮ MOBILNÍHO ZAŘÍZENÍ V AUTOMOBILU

USE OF MOBILE DEVICE SENSORS IN A CAR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. LUKÁŠ ARON
AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL ŽÁK
SUPERVISOR

BRNO 2013

Abstrakt

Tato práce se zabývá využitím akcelerometru v mobilním zařízení pro měření vibrací v automobilu. Toto měření mapuje kvalitu silničního povrchu. Ze signálu z akcelerometru a aktuální rychlosti je vyhodnoceno dle prahové funkce, zda se jedná o výmol či jiný typ nerovnosti. Pokud je údaj o dané nerovnosti vyhodnocen tak, že může při vyšší rychlosti poškodit vozidlo, je zaznamenán do interní paměti mobilního zařízení. Druhou částí této práce je vytvoření internetové aplikace, která naměřená data seskupuje, zpracovává na základě metody shlukování a výsledné hodnoty poskytuje zpět uživatelům. Výsledné hodnoty jsou shluky oblastí s výskytem nerovností, tyto shluky jsou zobrazovány na mapu a informují řidiče. Obsahem práce je teorie k senzorům na mobilním zařízení včetně zpracování signálu a implementační řešení mobilní a internetové aplikace.

Abstract

This thesis analyses the use of an accelerometer commonly present in a mobile device for the measuring of vibration in a vehicle. Such measurements are used to map the quality of the road surface. The signal from the accelerometer and the current travelling speed of the vehicle are evaluated according to a threshold function. This function allows for a determination of whether the signal reported by the device signifies a pot-hole or another type of defect on the road surface. If the defect is determined to pose a threat to a vehicle travelling at higher speeds, then this fact is saved into the memory of the device. The second part of this thesis deals with the design of an internet and a mobile application, which is able to collect all measured data processes using cluster analysis and then shares the results with other participating devices. Tha practical results is that a map of road surface defect is thus created, which is then displayed to road users. This thesis deals with theory of mobile device sensors, including signal processing, as well as the design creation of the internet and mobile application.

Klíčová slova

Android, SDK, akcelerometr, gyroskop, zpracování signálu, GPS, silnice, výmol

Keywords

Android, SDK, accelerometer, gyroscope, signal processing, GPS, roads, bump

Citace

Lukáš Aron: Využití senzorů
mobilního zařízení v automobilu, diplomová práce, Brno, FIT VUT v Brně, 2013

Využití senzorů mobilního zařízení v automobilu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Pavla Žáka

.....
Lukáš Aron
20. května 2013

Poděkování

Děkuji mému vedoucímu Ing. Pavlu Žákovi za odborné rady směřující k řešení diplomové práce. Zároveň děkuji mé rodině za podporu při studiu a odhodlání plnit mé touhy a sny.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Při dnešním technologickém pokroku a rozvoji mobilních zařízení, disponujících mnoha senzory, je kladen důraz na rozvoj, výkon, výdrž baterie a mobilitu zařízení. Změna vzhledu a funkční výbavy je obvyklá u každého výrobce, ovšem jsou nastaveny určité představy, lze říci až standardy dnešní doby, které specifikují, co by měl mobilní telefon obsahovat a většina výrobců se je snaží dodržet a přidat mnohem více dalších doplňků. Pokud budou pominuty základní služby jako je volání či zasílání krátkých textových zpráv (SMS) a zasílání multimediálních zpráv (MMS), je trendem moderní doby umožnit přehrát hudbu nebo video v jakémkoliv formátu a v co možná nejvyšší kvalitě, možnost fotografovat ve velkém rozlišení, včetně využití blesku, být neustále v kontaktu s internetem za použití bezdrátových technologií Wi-Fi a BlueTooth nebo v neposlední řadě určit polohu za pomocí GPS modulu. Mezi základní výbavu mobilních zařízení lze zařadit mnoho senzorů, které jsou využívány pro základní věci, jako je například reakce na rotaci zařízení tím, že se přizpůsobí displej otočením do polohy takové, která uživateli vyhovuje. Mnoho senzorů, která jsou standardní výbavou mnoho uživatelů využívá a nejsou si toho vědomi. Každé čidlo neboli senzor lze z programátorského hlediska využít na mnoho aplikací, než na které byl původně zamýšlen.

Tento projekt je primárně zaměřen na tvorbu aplikace využívající senzory na mobilním zařízení. Zařízení obsahuje operační systém Android, který je popsán v kapitole 2. V této kapitole lze nalézt i podrobnosti k vývoji, ladění a následné distribuce aplikace na dané platformě.

Tento projekt si klade za cíl navrhnout a naprogramovat aplikaci pro mobilní zařízení, které se ukotví v kokpitu automobilu. Na základě senzorů v mobilním zařízení bude probíhat, při jízdě v automobilu, mapování silničního povrchu a v případě výskytu nerovností nebo výmolů, bude zaznamenána poloha a naměřená hodnota. Tyto naměřené hodnoty jsou ukládány na mobilním zařízení. Součástí aplikace je navržení internetové aplikace, jejíž hlavním úkolem je shromažďování dat ze všech mobilních zařízení. Data jsou na serveru zpracována metodou shlukování a následně poskytována uživatelům mobilních zařízení. Uživatelé mohou výsledné oblasti s naměřenými nerovnostmi pozorovat na mapě v mobilní aplikaci nebo na internetové stránce projektu. Při vývoji je nutné využít několik senzorů v mobilním zařízení, kde primárně je využit akcelerometr a GPS modul. Základní informace k senzorům lze nalézt v kapitole 3.

Kapitola 4 popisuje základní princip získání dat ze senzorů a vlastnosti signálu. Následuje popis internetových technologií v kapitole 5, které jsou využity při tvorbě internetové aplikace. Návrh a informace týkající se tvorby mobilní aplikace lze nalézt v kapitole 6. Tato kapitola zahrnuje řešení všech problémů, které je nutné vyřešit při tvorbě projektu.

Poslední kapitolou zabývající se implementací je řešení internetové aplikace nebo webové služby 7. V kapitole 8 jsou shrnuty výsledky této práce, ať už z pohledu mobilní aplikace, nebo internetové služby. Poslední kapitolou je závěr 9, kde je celkové hodnocení práce a přínos s případnými možnostmi, jak aplikaci v budoucnu vylepšit nebo návrhy možných změn.

Kapitola 2

Mobilní zařízení

Obsahem kapitoly je základní popis mobilní platformy Android od společnosti Google Inc. Tento operační systém je popsán v závislosti na mobilní zařízení, na která byl původně vyvinut a je i nadále vyvíjen. Následuje popis životního cyklu aplikace, který je specifický právě na mobilních zařízeních s tímto systémem. Nemá smysl rozebírat operační systém příliš do detailů z důvodu častých změn a rychlému vývoji. Přesto je popsána základní architektura systému, která je shodná přes většinu vydaných verzí a jsou v dnešní době nejvíce používané. Aktuálně je primární verze systému Android označena verzí 4.x a jeho nejnovější verze nese kódové označení Jelly Bean (4.2).

Další částí kapitoly je popis vývoje softwaru pro mobilní zařízení na platformě Android, zahrnující základní popis balíčku SDK pro Android od společnosti Google Inc., který je hlavním zdrojem knihoven, nástrojů a dokumentace k dané platformě. V této části lze nalézt popis dostupných nástrojů pro usnadnění vývoje, ladění a následnou distribuci hotové aplikace. V samotném závěru kapitoly je alternativní způsob vývoje mobilních aplikací, které jsou doprovázeny popisem jednotlivých způsobů a přístupů umožňujících tvorbu aplikací na více platforem.

Hlavním rozdílem u alternativní tvorby aplikací je implementační jazyk. Trendem dnešní doby je využít rozhraní Web UI, které je obsaženo ve všech moderních mobilních zařízeních. Na jeho základě je vytvořena aplikace v jazycích HTML, CSS a JavaScriptu.

2.1 Operační systém Android

Android je mobilní operační systém založený na jádře Linux, který je primárně určen na mobilní zařízení s dotykovým displejem, jako jsou chytré telefony (smartphone) a tablety. Systém byl vyvíjen společností Android Inc, kterou Google Inc. finančně podpořil a následně v roce 2005 zakoupil. Operační systém byl představen veřejnosti v roce 2007 společně se založením konsorcia Open Handset Alliance. Konsorcium je zaměřené na hardware, software a telekomunikační společnosti a jejich prosazování svobodných standardů mobilních zařízení. Informace pocházejí z literatury [18].

Operační systém Android je vyvíjen jako open source pod dvěma typy licencování. Linux jako jádro systému je vyvíjeno pod GNU General Public License, ale samotná platforma Android podléhá licenci Apache Software Lincense. Tyto licence umožňuje software volně upravovat, distribuovat výrobcům mobilních zařízení, operátorům nebo programátorům. Android má rozsáhlou komunitu vývojářů, kteří rozšiřují standardní funkcionalitu systému tvorbou vlastních aplikací, ať už komerčních či volně ke stažení. Aplikace jsou psány pri-

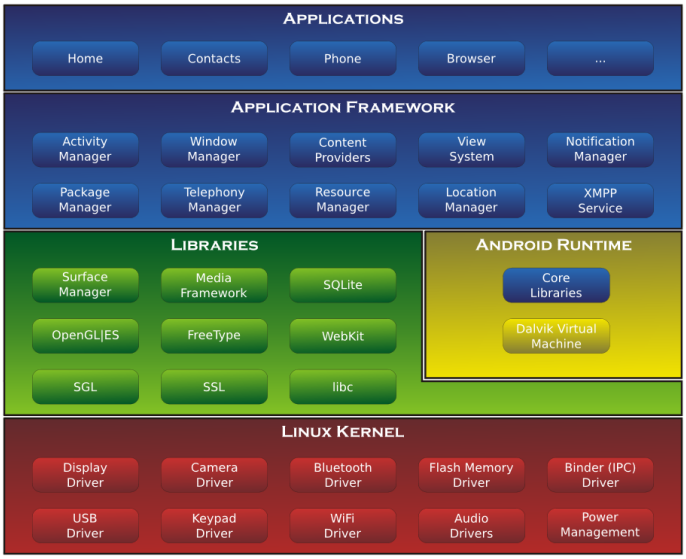
márně v upravené verzi programovacího jazyka Java. Aplikace rozšiřující systém Android lze distribuovat a stahovat z tzv. marketů, kde mezi nejpoužívanější a nejvyhledávanější patří Google Play, který je podporován společností Google Inc., jak je již z názvu patrné. Tento market aktuálně (v roce 2012) obsahuje zhruba 700 000 aplikací, které jsou rozčleněny do několika základních kategorií.

Tyto skutečnosti umožnily Androidu k dosažení nejvíce používané mobilní platformě na světě. Android používá celosvětově zhruba 75% všech mobilních zařízení. Jelikož se jedná o velice populární systém je postupně nasazován i do jiných zařízení než jen mobilních telefonů nebo tabletů. Své uplatnění nachází v televizních přijímačích, rekordérech, set-top boxech, čtečkách elektronických knih a v neposlední řadě i ve fotoaparátech.

Architekturu systému Android popisuje obrázek 2.1, který je přejat z literatury [18]. Na schématu lze vidět rozdělení do pěti základních skupin. První a zároveň spodní vrstva je `Linux kernel`, která definuje základ (jádro) operačního systému, které obsahuje základní ovladače, správu paměti a napájení. Další vrstvou je `Libraries` zahrnující základní knihovny pro jazyk C, jádro internetového prohlížeče, databázi SQLite, framework pro práci s médii, knihovnu pro práci s grafikou OpenGL ES, síťové knihovny a knihovny nutné pro běh a případný vývoj aplikací. Na stejné úrovni lze najít `Android runtime` část popisující základní knihovny jazyka Java a virtuální stroj nutný pro spuštění kompilovaných souborů jazyka.

Jedná se o framework pro vývoj aplikací, který zapouzdřuje nižší vrstvy architektury platformy Android do jednoduchého API. Toto API je základní přístupný bod pro všechny vývojáře, kteří tvoří aplikace pro tuto platformu v jazyce Java. Tento způsob vývoje se nazývá nativní, tedy přirozený, kde programovací jazyk je přímo doporučován, spravován a podporován výrobcem platformy včetně dodání podpůrných nástrojů.

Jelikož je Android z většiny vybudován na jazyce Java (mimo Linuxové jádro) je nutné zajistit přítomnost virtuálního stroje. Pro efektivní využití mobilní platformy, nízké spotřeby elektrické energie a odstranění některých částí klasického virtuálního stroje, které nejsou v mobilních zařízeních často používány, byl vyvinut virtuální stroj Dalvik. Informace vycházejí z knih [16], [18].

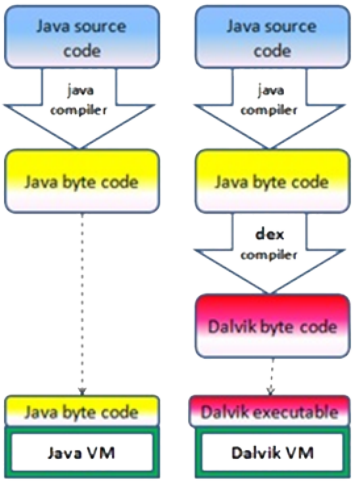


Obrázek 2.1: schéma architektury systému Android, přejato z [18]

V porovnání s klasickým operačním systémem pro stolní počítač nebo notebook je Android rozšířen o možnosti získat polohu, orientaci zařízení v prostoru, přijímat rádiový signál a spolupracovat s celou řadou senzorů, které jsou v dnešní době integrovány do mobilních zařízení. Mezi nejvíce používané senzory lze zařadit například akcelerometr, gyroskop či magnetometr (kompas). Tyto senzory lze využít například pro detekování otáčení mobilního zařízení a tím přizpůsobit zobrazení obsahu v závislosti na poloze zařízení. Detailní popis senzorů použitých pro implementaci tohoto projektu lze nalézt v kapitole 3.

Důležitým krokem společnosti Google, při vývoji operačního systému Android byl fakt, že se jedná o mobilní platformu. Tato platforma má odlišné nároky na paměť a rychlost. Dalo by se konstatovat, že je v těchto parametrech o osm až deset let za klasickými počítači či notebooky. Zároveň mají omezený přísun energie. Při vývoji operačního systému Android byl brán zřetel na tyto nároky. Výsledek lze pozorovat množstvím knihoven a balíčků na platformu Android, které jsou plné nových vlastností a optimalizací.

Tyto kroky vedly Google k optimalizaci JVM (Java Virtual Machine) - virtuální stroj, přes který se aplikace spouštějí. Na platformě Android dostal tento virtuální stroj název Dalvik VM, který nese své jméno po městě na Islandu. Dalvik VM zpracovává vygenerované třídní (class) soubory Javy a vytváří z nich jeden nebo více Dalvik spustitelných souborů (s koncovkou .dex). Cílem Dalvik VM je vždy najít cestu k optimalizaci. Především se jedná o optimalizaci parametrů místa, výkonu a zachování co nejdelší výdrže baterie. Porovnání Dalvik VM s klasickám Java VM je ilustrováno obrázkem 2.2. V levé části je zobrazen postup překladu zdrojových souborů jazyka Java do byte kódu a následného spuštění přes Java virtuální stroj. V pravé části je stejný proces na platformě Android, kde je přidán navíc proces překladu do Dalvik byte kódu a následné spuštění v Dalvik virtuálním stroji. Další informace o Dalvik VM lze nalézt v literatuře [13], ze které tento text čerpá.



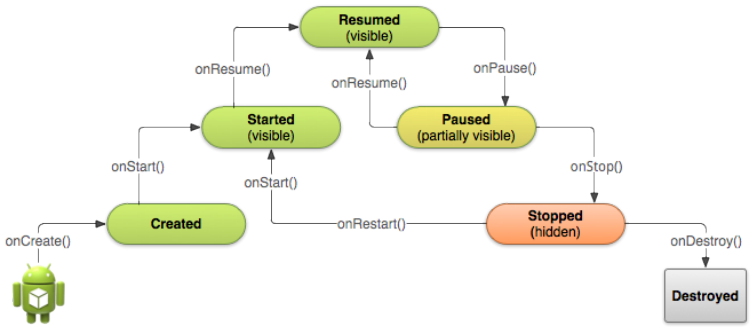
Obrázek 2.2: porovnání Java VM s Dalvik VM, přejato z [13]

Životní cyklus

Na operačním systému Android se lze setkat se specifickým životním cyklem aplikace či služby, kde první je pro aplikace s grafickým uživatelským rozhraním a druhý pro aplikace běžící na pozadí, pojmenované jako služby. Aplikace s grafickým uživatelským rozhraním

je na platformě Android rozdělena na aktivity. Aktivita je vždy jedna obrazovka aplikace zaplňující celou obrazovku, ale může být také menší a být jen plovoucím dialogem. V minulosti bylo na mobilních platformách jistotou, že veškerý výkon mobilního zařízení byl soustředěn na jedinou aplikaci, která aktuálně byla spuštěna a uživatel interagoval s uživatelským rozhraním. Postupem vývoje rostl výkon a zájem uživatelů o možnost používat více aplikací najednou tak, jak tento způsob ovládání známe z klasického počítače. Na základě těchto požadavků byl vytvořen model dvou základních částí, které lze do větší míry použít dodnes. Jedná se především o aplikaci s grafickým uživatelským rozhraním reagující na požadavky uživatele a služby běžící na pozadí, které vykonávají svou práci bez ohledu na uživatele.

Jedná se o multitasking aplikací a služeb, který byl přejat z klasického modelu stolních počítačů. Na mobilní platformě je nutné řešit jiné problémy, než na klasickém počítači, a to především nároky na paměť, procesorový výkon a výdrž baterie. Naopak není nutné řešit přístup mnoha uživatelů na jedno mobilní zařízení. V závislosti na směru vývoje bylo nutné přizpůsobit životní cyklus mobilních aplikací. Na platformě Android je vytvořen specifický životní cyklus aplikace s grafickým uživatelským rozhraním 2.3.



Obrázek 2.3: životní cyklus aplikace s grafickým uživatelským rozhraním, přejato z [16]

Celý životní cyklus aplikace probíhá mezi voláními funkcí `onCreate()` a `onDestroy()`. Aktivita je viditelná mezi voláními funkcí `onStart()` a `onStop()`. Uživatelský vstup lze zpracovávat mezi voláními funkcí `onResume()` a `onPause()`. Životní cyklus aktivity nemusí být vždy kompletní. Pokud aktivita zpracovala systémové volání funkce `onPause()`, systém ji může při nedostatku prostředků ukončit.

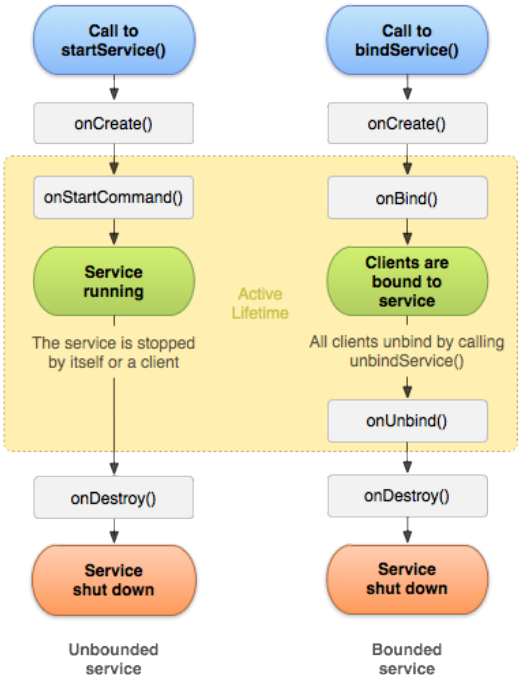
Funkce `onStop()` a `onDestroy()` tudíž nemusí být zavolány. Z tohoto důvodu je funkce `onPause()` využívána pro ukládání stavu dané aktivity. Při vývoji je nutné uvažovat možné ukončení aplikace a tím optimalizovat využívání dostupných zdrojů a jejich zpětné uvolňování. Na vývoj aplikací primárně pro platformu Android je vydáno několik doporučení, která je vhodné respektovat. Tato doporučení jsou od vývojářů systému Android a zabývají se především správnou prací s přidělenými prostředky, senzory, šetření baterie a v neposlední řadě bezpečností. Při vývoji aplikací s uživatelským rozhraním je vhodné respektovat doporučení na tvorbu uživatelského rozhraní, které umožňuje přizpůsobit velikost grafických prvků pro různá rozlišení a různé velikosti obrazovek. Pro návrh grafického rozhraní existuje dokument, který pochází z dílny designerů, kteří navrhovali vzhled samotného operačního systému. Doporučení pro vývoj aplikací a návrh grafického uživatelského rozhraní lze nalézt na oficiální stránce platformy Android [3].

Druhou možností je vytvořit aplikaci, která běží na pozadí a nepotřebuje ke svému běhu

grafické uživatelské rozhraní. Tento typ aplikace se nazývá služba a slouží například pro síťovou komunikaci nebo sběr dat z některého ze senzorů. Životní cyklus služby je mírně odlišný od životního cyklu aktivity, její stavy a systémová volání jsou popsány na obrázku 2.4. Na platformě Android existují dva typy služby: vázaná a nevázaná.

Nevázaný typ služby je po spuštění například z aktivity samostatně pracující proces a po ukončení aktivity, ze kterého byla služba vytvořena, je stále aktivní. Služba provádí jednoduchou úlohu a neposkytuje žádné návratové hodnoty. Typickou úlohou této služby je například stažení souboru z internetu, kde po stažení souboru je služba ukončena.

Vázaný typ nabízí známé rozhraní klient-server, které umožňuje komponentě (aktivitě) komunikovat s danou službou, zasílat požadavky a získat odpovědi. Tento typ služby je aktivní do doby než je implicitně ukončen z dané aktivity nebo je ukončena aktivita, ke které je vázán. Je zde možnost navázání více komponentů na jednu službu a pokud jsou všechny aktivity ukončeny, je služba také ukončena.



Obrázek 2.4: schéma životního cyklu služby, přejato z [18]

Na základě schéma lze opět jednoznačně určit, že služba probíhá mezi voláními funkcí `onCreate()` a `onDestroy()`. Hlavní aktivita dané služby je spuštěna po volání systémové funkce `onStartCommand()` u nevázané služby a `onBind()` u vázané služby. Před ukončením je u vázané služby volána systémová funkce `onUnbind()`, která ukončuje danou službu z důvodu, že není již žádný zájemce o službu, u služby nevázané je běh ukončen po vykonání služby. Detailnější informace lze nalézt v literatuře, ze které tento text pochází [18] nebo [16].

2.2 Vývojové nástroje

Pro vývoj na platformě Android je dostupný balíček knihoven, aplikací, dokumentace a doporučených postupů tzv. Software Development Kit (SDK) od společnosti Google, který je základem pro vývoj na tuto platformu. Primárním programovacím jazykem pro platformu Android je Java a je pro tento jazyk vytvořena podpora v podobě dokumentace. Na vývoji pro platformu Android v dnešní době spolupracuje mnoho společností či jednotlivci, kteří tvoří obrovskou vývojovou komunitu. Vývojáři mezi sebou sdílí návody, rady a řešení problémů, se kterými se lze při vývoji aplikací setkat.

Na oficiálních stránkách platformy Android lze stáhnout archiv obsahující Android Developer Tools. Jedná se o Android SDK včetně vývojového prostředí Eclipse. Vývoj platformy a zároveň vývojového prostředí je podporován společností Google. Tento balíček je dostupný pro dnes nejpoužívanější platformy jako je Linux, Windows a Mac. Pro vývoj aplikací, jejichž odezva je kriticky důležitá (například hry), lze využít další sadu knihoven, podporovanou společností Google, Android Native Development kit (NDK). NDK je balík knihoven, podobně jako SDK, pro vývoj aplikací na platformu Android, ale lišící se programovacím jazykem. Android NDK je primárně určen pro vývoj v jazyce C/C++, ale není tak kvalitně zdokumentován a společnost Google. nedoporučuje využívat tento balíček pouze za účelem programování aplikace v jiném jazyce. Primárně je plně podporován jazyk Java, který je považován za nativní programovací jazyk pro platformu Android. Pro usnadnění tvorby aplikací, zkoušení a ladění jsou v SDK dostupné nástroje, které ocení každý vývojář. Jednotlivé nástroje jsou propojeny s vývojovým prostředím Eclipse a při návrhu a tvorbě mobilních aplikací jsou tyto nástroje používány při programování, ladění, kompilaci a testování. Následuje výpis a popis nejpoužívanějších nástrojů.

Android Emulator

Android SDK archiv obsahuje plug-in do vývojového prostředí Eclipse, označovaný jako Android Development Tools (ADT), obsahující nástroje pro vývoj na platformě Android. Není nutné využívat prostředí Eclipse. Lze vyvíjet software v jakémkoli jiném prostředí a překlad provést přes příkazovou řádku. Oba přístupy podporují emulátor, který lze využít pro spuštění, ladění a testování vyvíjené aplikace.

Emulátor je implementace Dalvik VM, vytvářející korektní platformu pro spouštění aplikací pro Android. Jelikož není hardwarově závislý tak pokrývá většinu dostupných fyzických zařízení. Pro tvorbu aplikace není nutné vlastnit fyzické zařízení s operačním systémem Android v 90% případů. Lze nalézt omezení emulátoru, která zahrnují například USB připojení, zachytávání obrázků (fotografií) či videa, Bluetooth, Wi-Fi, NFC a OpenGL ES .Na základě plné konektivity lze upravovat rychlost připojení k internetu, simulovat zpoždění připojení či úplný výpadek spojení. Mezi další a důležitý prvek emulátoru patří možnost simulovat příchozí hovor či textovou zprávu. Zároveň lze sledovat chování vyvíjené aplikace při reakci na tyto události. V případě použití plug-inu do vývojového prostředí Eclipse, je emulátor automaticky spuštěn při zahájení testování aplikace. Pokud vývoj probíhá mimo Eclipse, lze na emulátoru spustit projekt explicitně přes příkazovou řádku.Ve spojení s emulátorem je vhodné uvést Android Virtual Device (AVD), který popisuje virtuální zařízení. Toto zařízení obsahuje základní popis hardwaru daného zařízení a operační systém Android. Vlastnosti virtuálního zařízení mohou být jak reálné - nastaveny dle existujícího zařízení, tak upravené - například jiné hardwarové prvky či obrovské rozlišení. Čerpáno z [16].

Nástroje potřebné pro vývoj

Jak již bylo řečeno Andoid SDK obsahuje mnoho nástrojů, které ulehčují práci s vývojem, laděním a následnou distribucí aplikací. Jedná se o mnoho nástrojů a tato práce není zaměřena pouze na popis platformy Android. Z tohoto důvodu následuje základní popis nejdůležitějších nástrojů, které s největší pravděpodobností budou využity při implementaci výsledné aplikace. Následující informace o jednotlivých nástrojích pocházejí z oficiální dokumentace platformy Android a jejího SDK, které je dostupné on-line [3].

Android Debug Bridge

Android Debug Bridge neboli ADB je všestranný nástroj pro příkazovou řádku, který umožňuje komunikaci s emulátorem či připojeným fyzickým zařízením s operačním systémem Android. Nástroj je v jádru klient-server aplikace obsahující následující tři části.

- Klient, který běží na stroji (fyzickém či emulovaném) a lze s ním komunikovat na základě ADB příkazů. Nástroje jako například zmíněný ADT plug-in také vytváří ADB klienta.
- Server, který je spuštěn jako proces v pozadí na pracovním stroji (fyzickém či emulovaném). Server obsluhuje komunikaci mezi klientem a ADB démonem běžícím na zařízení nebo v emulátoru.
- Démon, který je spuštěn v pozadí na každém emulátoru či zařízení.

V případě, že je spuštěn ADB klient, tak je nejprve hledán běžící ADB server. Pokud není, je spuštěna instance serveru a následně přiřazen TCP port 5037, kde je naslouchán příkazům. Příkazy jsou zasílány z ADB klientů, kteří používají port 5037 pro komunikaci se serverem. Server poté vytvoří spojení na všech běžících emulátorech či fyzických zařízeních. Zjišťování dostupných zařízení probíhá skenováním lichých čísel portů počínající portem 5555 a končící portem 5585. Když server nalezne ADB démona, vytvoří připojeni na daném portu.

Poté co server vytvoří všechna připojení se všemi zařízeními či emulátory, lze používat ADB příkazy pro přístup k těmto zařízením. Jelikož server obsluhuje spojení a komunikaci mezi mnoha ADB klienty, lze tímto způsobem manipulovat s jakýmkoli emulátorem či zařízením z kteréhokoli klienta.

Device Monitor

Android Device Monitor je samostatná sada nástrojů poskytující grafické uživatelské rozhraní pro ladění a analýzu některých aplikací. Tato sada nepožaduje instalaci či integraci do vývojového prostředí, ale lze ji využívat přímo z příkazové řádky ihned po stažení od výrobce. Device Monitor zahrnuje několik důležitých celků, které jsou popsány následovně.

- DDMS ladící nástroj Dalvik Debug Monitor Server poskytující možnost zachycení obrazovky zařízení, informaci o vláknech, paměti, příchozích hovorech či SMS zprávách a další informace týkající se aktuálního běhu aplikace a vnějších vlivů.
- Tracer for OpenGL ES je nástroj pro analýzu OpenGL Embedded Systems (ES) kódu v aplikaci Android. Tento nástroj umožňuje zachytávat OpenGL ES příkazy a obrázky snímek po snímku, které mohou získat náhled na výstup a získat přehled, jak jsou jednotlivé příkazy provedeny.

- Hierarchy Viewer umožňuje ladění a optimalizaci uživatelského rozhraní. Poskytuje vizuální reprezentaci hierarchie jednotlivých komponent na daných aktivitách.
- TracerView grafický prohlížeč pro exekuční záznamy uložené v aplikaci. Slouží pro ladění aplikace a profilování výkonu.

UI/Application Exerciser Monkey

Zkráceně je tento program nazýván pouze Monkey a slouží pro generování pseudo-náhodného proudu uživatelských událostí, jako je například kliknutí, různá gesta, stisknutí několika kláves současně a mnoho systémových událostí. Využívá se tedy hlavně ve fázi testování, kde je tento způsob testování pojmenován stress-test. Program je spouštěn z příkazové řádky a lze jej spouštět jak na fyzickém zařízení, tak v emulátoru. Testování umožňuje zvolit mnoho událostí, která mají být spouštěna. Tyto události je možné kategorizovat následovně.

- základní konfigurace - obsahuje například počet událostí, které mají být vyvolány (testovány)
- funkční omezení - omezení týkající se testování, například pouze jednoho konkrétního balíčku
- typy událostí a jejich frekvence
- ladící informace

Když program Monkey běží, generuje události a zasílá je systému, současně sleduje systém, který testuje, a sleduje tři základní podmínky hrozby pádu aplikace. Tyto tři podmínky jsou následující.

- Pokud jsou v programu Monkey nastaveny funkční omezení na jeden či více balíčků, tak je sledováno, zda není pokus o přesun do jiného balíčku předtím, než je testován a tento balíček je blokován.
- Pokud aplikace havaruje nebo obsluhuje neošetřenou výjimku, program Monkey se zastaví a nahlásí chybu.
- Pokud aplikace není schopen odpovědět na chybu, program Monkey se zastaví a nahlásí chybu.

Zipalign

Zipalign je nástroj pro zarovnání archivů, poskytující důležitou optimalizaci pro Android aplikační soubory APK. Hlavním účelem je zajistit, že všechny nekomprimované údaje začínají určitým uspořádáním vhledem k počátku souboru. Konkrétně způsobí, že všechna nekomprimovaná data, jako jsou obrázky nebo syrové soubory, jsou zarovnána na 4 byty. Tento způsob zarovnání umožňuje, aby všechny části byly přístupné pomocí tzv. `mmap` funkce. Výhodou je snížení množství paměti RAM spotřebované při spuštění aplikace.

Tento nástroj je používán před samotnou distribucí aplikace. Pokud je aplikace programována s využitím doporučeného vývojového prostředí a s jeho pomocí i vytvářen výsledný APK balíček, je tento nástroj použit automaticky.

S touto platformou přichází upravená verze programovacího jazyka Java pro virtuální stroj Dalvik tak, jak bylo zmíněno v předchozí části. Jelikož programovat mobilní aplikace

pouze v jednom programovacím jazyce je omezující pro některé vývojáře či společnosti zabývající se mobilním vývojem. Omezující ve smyslu programování mobilních aplikací na více platforem. Pro každou platformu je nutné se učit jiný programovací jazyk, postup vývoje a specifické API rozhraní pro přístup k hardwarovým prvkům. Z tohoto důvodu postupem let vzniklo několik projektů umožňujících vývoj v různých jazycích a na různé platformy včetně multiplatformních.

Nejprve byla snaha programovat aplikace ve skriptovacích jazycích, jako je například Python nebo Ruby a nyní se jedná především o jazyk C# . Aktuálně jsou populární vývojové nástroje, které umožňují vyvíjet aplikace na webových technologiích HTML, CSS a JavaScript. Na jejich základě je vybudována základní funkčnost aplikace a pro specifické vlastnosti umožňují přistupovat k nativnímu API rozhraní.

Mobilní telefon je z dnešního pohledu chápán jako osobní počítač, který je dostupný uživateli po celý den. Chytré telefony jsou levnější než klasické počítače nebo notebooky, ale jsou více vhodné pro svou mobilitu a neustále se zvyšující výkon a dostupnost služeb, které nejsou na počítači dostupné. Příkladem takové služby může být získání aktuální polohy. S postupným vývojem a rozvojem mobilních technologií lze pozorovat zvyšující se zájem o přístup na internet právě skrz mobilní platformu. Na základě těchto potřeb rozvíjející se společnosti a požadavkům kladených právě na vývoj a provoz chytrých telefonů lze spoléhat na to, že každý chytrý mobilní telefon obsahuje internetový prohlížeč. Pro tento prohlížeč lze programovat aplikace stejně dobře a jednoduše, jako aplikace na klasickém PC či v SDK pro jednotlivé platformy.

2.3 Alternativní vývoj

Před existencí frameworků umožňující vývoj aplikací pro různé platformy, bylo nalezeno řešení vývoje skrz zabudované webové rozhraní implementované jako nativní aplikace. Jednalo se o praktickou cestu, jak vytvořit mobilní aplikaci rychle a zároveň multiplatformní. Grafické uživatelské rozhraní je reprezentováno jako série několika obrazovek. Z vyššího pohledu lze tento způsob přirovnat k rozhraní s klasickým webovým rozhraním známé z internetové sítě na klasických stolních počítačích.

V mobilních aplikacích je zvykem pro téměř každé kliknutí zobrazit novou obrazovku právě tak, jak je to u tradičních internetových aplikací, kde kliknutím je načtena nová stránka. Výhodou tohoto přístupu je fakt, že aplikace napsaná pro webové uživatelské rozhraní nemusí nutně používat internetové připojení, ale pouze využívá prohlížeč k zobrazení svého obsahu uživateli. Hlavní předností pro vývojáře je právě dostupnost známých technologií HTML, CSS a JavaScript, které jsou dnes považovány za internetové standardy. Na základě těchto předností vzniklo několik frameworků pro vývoj mobilních aplikací.

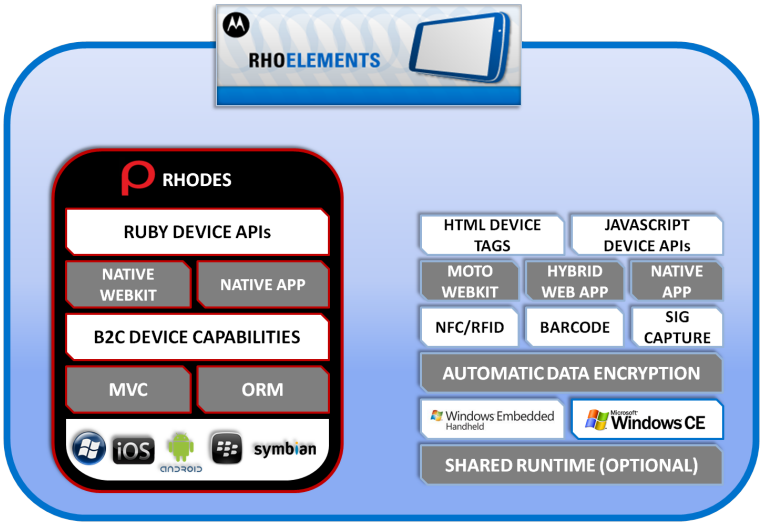
Tyto frameworky umožňují přistupovat k nativnímu API rozhraní jednotlivých platforem za účelem získání plné kontroly nad hardwarovými prostředky. Výsledkem je aplikace, kterou lze uložit do instalačního balíčku určeného pro cílovou platformu a následně umístit na internetový obchod nebo nainstalovat přímo na konkrétní fyzické mobilní zařízení.

Rhodes

Rhodes je jedním z předních multiplatformních aplikačních frameworků určených pro mobilní platformu. Jedná se o komerčně podporovaný open source projekt spadající pod licenci MIT. Je dostupný především pro platformy iPhone, Android, Windows Mobile, Symbian a nově BlackBerry. Framework umožňuje tvorbu aplikací s využitím HTML, CSS, JavaScriptu

a programovacího jazyka Ruby. Je zaměřen na vývojáře se zkušenostmi s vývojem webových aplikací, kteří chtějí vytvářet mobilní aplikace bez učení se novým postupům či jiného programovacího jazyka včetně specifického SDK. Mezi přednosti tohoto frameworku lze zařadit využití skriptovacího jazyka Ruby a dosáhnout tím úsporu času a velikosti kódu. Podmínkou je přítomnost SDK cílové platformy na vývojovém stroji.

Rhodes je primárně určen pro vývoj Enterprise aplikací. Umožňuje rychlý a jednoduchý vývoj aplikací reprezentující sérii obrazovek obsahujících standardní uživatelské rozhraní. Tento framework není vhodný pro vývoj her nebo aplikací vyžadujících složitější interaktivní grafické prvky. Další informace o frameworku Rhodes lze nalézt v [5].



Obrázek 2.5: architektura frameworku Rhodes, přejato z [5]

Na obrázku 2.5 je zobrazeno schéma RhoElements, pod které patří zmiňovaný Rhodes. V levé části je popsána architektura frameworku, kde na nejnižší úrovni je API samotného zařízení, které je naprogramované v jazyce Ruby. Do následující vrstvy jsou zahrnuty zobrazovací elementy, kde primární částí je nativní jádro prohlížeče nebo nativní aplikace. B2C funkce zařízení poskytují operace s uživatelem, které jsou zde reprezentovány komunikací uživatele s webovým prohlížečem na daném zařízení. Vykreslování dat je především přes MVC nebo ORM mode. Více se o těchto modelech lze dozvědět v literatuře [5].

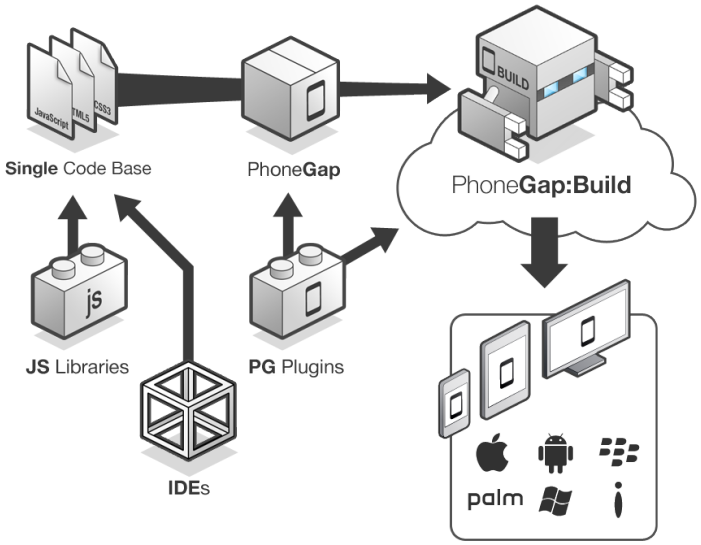
PhoneGap

PhoneGap je open source framework pro tvorbu nativních mobilních aplikací s využitím HTML, CSS a JavaScriptu. Jedná se o stejné hlavní technologie jako u předešlého Rhodes frameworku. Podporovány jsou především platformy iPhone, Android, BlackBerry, Palm webOS a Symbian WRT. PhoneGap je perfektní nástroj pro převedení webové aplikace do nativní verze. Tento framework je přizpůsoben pro vývojáře webových technologií. Není nutné ovládat další programovací jazyky či samotné SDK výrobců jednotlivých platforem. Naopak je kladen důraz na pokročilou znalost jazyka JavaScript.

PhoneGap poskytuje bohatou kolekci klientských JavaScriptových knihoven usnadňujících vývoj. Výhodou tohoto frameworku je vývoj webové aplikace a přeložena je do nativní aplikace, kterou koncový uživatel může nainstalovat na mobilní zařízení nebo distribuovat.

Nativní aplikaci je umožněn přístup k určitým funkcím, které nejsou z webové aplikace dostupné. Mezi takové funkce lze zařadit kontakty, geolokace, fotoaparát a různé senzory. Tento přístup je umožněn přes PhoneGap JavaScript API.

Vývoj aplikace probíhá stejně jako v předešlém případě psaním aplikace v HTML, CSS a JavaScriptu. Nejsou zde kladena žádná omezení. Pokud vývojář již v minulosti vytvořil jakoukoli webovou aplikaci, nyní ji lze bez větších problémů převést na mobilní aplikaci. V poslední verzi frameworku PhoneGap je již začleněna podpora HTML5 a CSS3. Další informace lze najít v literatuře [5].



Obrázek 2.6: schéma překladu s využitím frameworku PhoneGap, přejato z [5]

Na obrázku 2.6 je schéma vývoje mobilní aplikace s využitím frameworku PhoneGap. Schéma začíná napsáním zdrojového kódu v jakémkoli vývojovém prostředí. Při programování jsou využity knihovny JavaScriptu umožňující přístup k hardwarovým prvkům. Při překladu do koncové podoby pro konkrétní zařízení jsou přidruženy specifické knihovny pro danou platformu a zdrojový kód v jazyce JavaScript je transformován na nativní kód dané platformy.

Appcelerator Titanium

Posledním jmenovaným frameworkem založeným na technologii HTML, CSS a JavaScriptu je Appcelerator Titanium, většinou nazývaný pouze Titanium. Stejně jako předešlé knihovny umožňuje vývoj v jazycích specifických pro webové rozhraní. Podporovány jsou především platformy iPhone a Android.

Proces překladu generuje zdrojový kód pro cílovou platformu. Nabízí jednoduché API pro tvorbu mobilních aplikací. Od předchozích frameworků se liší svým přístupem ke grafickému uživatelskému rozhraní. Pro standardní grafické prvky jsou nadefinovány konstantní prvky Titanium API, které jsou při překladu nahrazeny nativními grafickými prvky cílové platformy.

Primárně je vývoj grafického uživatelského rozhraní směřován do jazyka JavaScript, kde se programově sestavuje vzhled jednotlivých obrazovek a jejich posloupnost. Reakce na stisk, posun či pohyb, znázorňující nějaké gesto, je řešený na základě událostí tak, jak je zvykem pracovat v tomto jazyce při vývoji klasických internetových aplikací.

Přístup ke specifickým částem mobilního zařízení, které není dostupné pro webové rozhraní je řešeno stejně jako u předešlého frameworku, tedy přes JavaScriptové knihovny určené pro cílovou platformu. Existuje mnoho dostupných oficiálních knihoven pro usnadnění práce, které lze nalézt na oficiálních stránkách projektu.

Kapitola 3

Senzory

Dostupnost senzorů je jedna z klíčových vlastností systému Android, který jej odlišuje od ostatních systémů klasického počítače. Bez těchto senzorů se jedná o normální operační systém s mobilním internetovým prohlížečem a malou obrazovkou. Snímače neboli senzory jsou to, co umožňuje Androidu vyniknout. Uživatelé jsou dnes na snímače zvyklí a mohou provádět spousty dnes již přirozených gest s telefonem, které by bez senzorů pohybu nebyly možné.

Senzor je nějaký hardwarový prvek, který je připojen k zařízení umožňující získávat data z fyzického světa do softwaru. Aplikace využívající data ze senzorů, slouží k informování uživatele o fyzickém světě, ovládání her, vytváření rozšířené reality nebo poskytuje použitelný nástroj pro práci v reálném světě. Senzory pracují pouze v jednom směru, jsou tzv. pouze na čtení (read-only). Výjimkou je senzor NFC, který komunikuje obousměrně, ale není využit pro implementaci aplikace vycházející z této práce. Využití senzorů je z tohoto hlediska přímočaré. Na daném zařízení v určité aplikaci se nastaví poslech dat z konkrétního senzoru a následně se jen zpracovávají data, která z daného senzoru přicházejí. Text vychází z literatury [19, 8].

Platforma Android rozděluje senzory do tří základních skupin:

- Senzory pohybu - tyto senzory měří síly akcelerace a rotace ve třech základních osách. Tato kategorie zahrnuje akcelerometry, senzory gravitace, gyroskopy a senzory rotačního vektoru.
- Senzory prostředí - jedná se o senzory, které měří různé parametry okolí. Mluvíme zde například o měření teploty okolního vzduchu a tlaku, osvětlení či vlhkosti. Skupina zahrnuje barometry, tepelné snímače a fotometry.
- Senzory pozice - tato kategorie senzorů měří fyzickou polohu zařízení. Lze sem zařadit senzory orientace a geomagnetické snímače.

Android umožňuje přístup k senzorům přes tzv. senzor framework, který je obsažen v Android SDK. Některé senzory jsou čistě hardwarové a některé jsou softwarové. Hardwarové senzory jsou fyzické komponenty zabudované do daného mobilního zařízení nebo tabletu. Tyto senzory odvozují svá data na základě přímého měření specifických vlastností jako je například zrychlení nebo síla geomagnetického pole. Snímače na bázi softwaru nejsou fyzické komponenty, i když napodobují hardware na bázi čidel. Softwarové senzory odvozují svá data z jednoho nebo více hardwarových snímačů a jsou někdy nazývány virtuálními senzory nebo syntetickými senzory. Příkladem může být lineární senzor zrychlení nebo gravitační senzor.

Velké množství dnešních chytrých telefonů a tabletů obsahují senzory typu akcelerometr a geomagnetický snímač. Naopak nebývá pravidlem, že mobilní zařízení vždy obsahuje barometr či teploměr. Lze se setkat s mobilním zařízením, které obsahuje více senzorů stejného typu, kde tyto senzory mají rozdílný rozsah hodnot. Mobilní platforma Android umožňuje získávat data od různých senzorů současně. Tento přístup dovoluje vytvářet aplikace, které slouží pro různé účely. Typicky se jedná o úkony, pro které nebyl daný senzor navržen. Například na základě dat z akcelerometru, gyroskopu a geomagnetického snímače lze získat údaj o natočení displeje daného zařízení.

Následuje detailní popis základních senzorů, které jsou dnes zabudovány do většiny mobilních zařízení včetně tabletů a jsou využity pro implementaci výsledné aplikace.

3.1 Akcelerometr

Akcelerometr je elektromechanické zařízení, které měří zrychlení sil. Tyto síly mohou být statické jako tíhová síla, nebo dynamické - způsobeny pohybem nebo vibrováním akcelerometru. Akcelerometry dovolují měřit jak statické, tak dynamické gravitační zrychlení. U mnohých měření statického gravitačního zrychlení lze zjistit úhel vychýlení vzhledem k zemskému povrchu. Při měření dynamických zrychlení je možné analyzovat směr, kterým se zařízení pohybuje. Měření náklonu a zrychlení se nezdá být všechno.

Ve světě počítačů u společností IBM a Apple se nedávno začaly používat akcelerometry v laptopech pro ochranu pevných disků před poškozením. V případě, že náhodou upustíte laptop, akcelerometr zjistí náhlý volný pád a vypne pevný disk tak, že hlavy nenarazí na plotny. V podobném stylu jsou v průmyslu řešeny standardní způsoby odhalování autonehod a vystřelení airbagů ve stejném čase.

Je mnoho různých způsobů jak vytvořit akcelerometr. Některé akcelerometry využívají piezoelektrický jev - obsahují mikroskopické krystalové struktury, které při namáhání způsobeném zrychlením začnou generovat napětí. Další způsob je snímání kapacitance. Když máte vedle sebe dvě mikrostruktury, je mezi nimi určitá kapacitance. Jestliže zrychlovací síla pohybuje jednou z těchto struktur, pak se kapacitance mění. Při dodání nějakých obvodů k přeměně kapacitance na napětí lze s nadsázkou sestrojit akcelerometr. Jsou ještě další metody obsahující použití piezorezistivního jevu, horkých vzduchových bublin a světla.

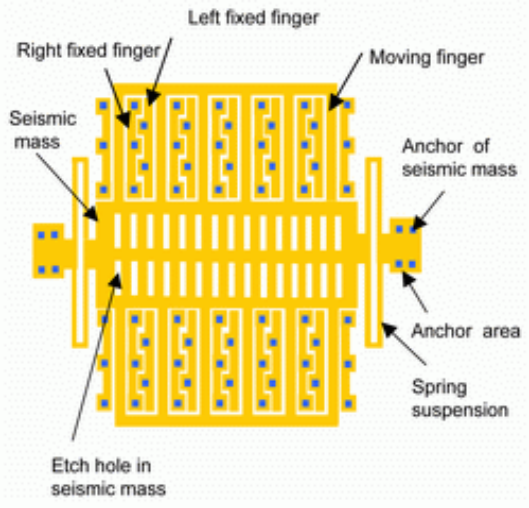
Akcelerometry v mobilních zařízeních jsou založeny na proměnné kapacitě tříelektrodového vzduchového kondenzátoru. Využívá se zde známé nelineární závislosti kapacity C na vzdálenosti elektrod kondenzátoru d (velikosti vzduchové mezery). Vzájemné závislosti jsou popsány matematickým vztahem 3.1, kde x je konstanta a S znamená plocha elektrod.

$$C = x.S \ /d \tag{3.1}$$

Pokud tedy jednu elektrodu uděláme pohyblivou a její pohyb bude závislý na působícím zrychlení, získáme kapacitní akcelerometr. A protože taková struktura pohyblivých malých nosníků (elektrod) je snadno realizovatelná MEMS (Micro-Electro-Mechanical Systems) technologií, vznikne nám MEMS akcelerometr.

Výsledná funkční struktura ovšem není tak jednoduchá, jak se na první pohled zdá. Hlavní je právě zajistit lineární a dostatečně citlivý převod zrychlení na mechanický posuvný pohyb. Ten totiž určuje samotný měřící rozsah senzoru, tj. maximální a minimální měřitelné zrychlení.

Obrázek 3.1 popisuje princip akcelerometru. Zde se vychází ze základního vztahu pro působení síly při zrychlení, tento vztah je popsán vzorcem 3.2.



Obrázek 3.1: schéma akcelerometru, přejato z [10]

$$F = m.a \tag{3.2}$$

Ve vzorci je F síla vzniklá působením zrychlení a na hmotu m (Seismic mass). Síla se pak přes pružiny (Spring suspension) převádí na posuv nosníku (Seismic mass), jejíž některé části tvoří pohyblivé elektrody vzduchového kondenzátoru (Moving finger). Jejich pozice vůči levým pevným elektrodám (Left fixed fingers) a pravým pevným elektrodám (Right fixed fingers) určuje elektronicky měřenou hodnotu kapacity takto vzniklého kondenzátoru.

Výše uvedená a popsaná struktura však umožňuje měření zrychlení jen v jednom směru, kolmém na pohyblivé elektrody = 1D akcelerometry. Technologicky vcelku není problém na čipu přidat další stejnou strukturu, pouze proti té předchozí, pootočenou o 90°. Vznikne tak 2D akcelerometr, který například měří v osách XY nebo XZ, dle natočení senzoru. Složitější je již vytvořit jednočipový 3D akcelerometr, protože se musí přidat výškově pohyblivá struktura v ose Z. detailní popis akcelerometru lze najít v literatuře, ze které tento text čerpá [10].

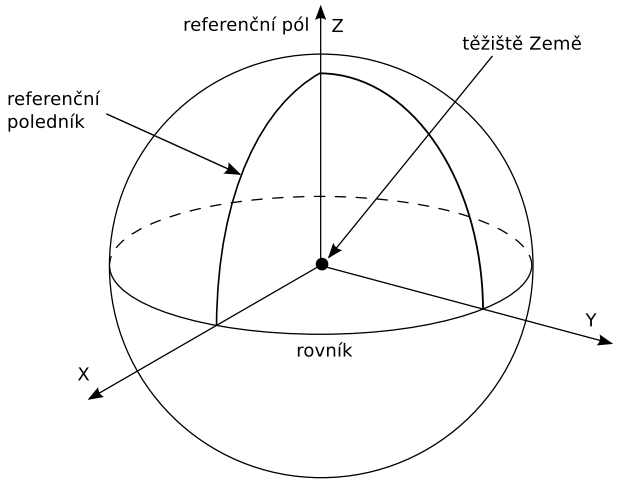
3.2 Získání polohy

Hlavním obsahem této části je popis systému získaní polohy, zejména je popis zaměřen na systém GPS. Systém GPS lze chápat i jako službu, která je poskytována zdarma pro všechna zařízení disponující GPS modulem. Služba GPS není časově omezená a lze ji využívat bez ohledu na čas a počasí. Pro doplnění dalších možností, jak získat polohu, jsou popsány základy dalších systémů. Systém WGS-84 je celosvětově uznávaný geodetický standard, který definuje souřadnicový systém, podrobnosti lze nalézt dále v textu, a to zejména jak tento systém funguje a k čemu slouží. Text kapitoly je čerpán z literatury [15], [22]

WGS-84

Vzorec pro výpočet vzdálenosti dvou pozic na Zemi využívá geodetický standard World Geodetic System 1984 (WGS-84), který je celosvětově uznávaným standardem pro použití

v kartografii, geodézii a navigaci vydaný americkým ministerstvem obrany roku 1984. WGS-84 definuje souřadnicový systém, referenční rotační elipsoid pro geodézii a navigaci.V současné době standard WGS-84 využívá modelu gravitačního pole Země definovaného v roce 1996. Model gravitačního pole Země (EGM96) byl naposledy revidován v roce 2004, kde došlo ke zpřesnění modelu k aktuálnímu stavu gravitačního pole. Samotné spojení rotačního elipsoidu se Zemí je takové, že počátek souřadnic leží v těžišti Země, osa Z odpovídá ose rotace Země, osa Y leží v rovině rovníku 90 stupňů východně od osy X. Spojení referenčního rotačního elipsoidu s reálnou Zemí je dáno souřadnicemi základních stanic systému WGS-84 rozmístěných po celé zeměkouli. Pro reálnější představu je zobrazen standard WGS-84 na obrázku 3.2.



Obrázek 3.2: WGS 84, přejato z [15]

Standard WGS-84 je výchozím a referenčním modelem pro Global Positioning System neboli GPS, který je schopný získat pozici zařízení s přesností ±1 cm. Poslední revidovaná verze standardu WGS-84 z roku 2004 začíná být zastaralá a neodpovídá nárokům dnešní moderní doby, které jsou rychlost a přesnost. Z tohoto důvodu vzniká revize nová. Nová revize zpřesňuje výpočty pozic, které jsou důležité pro orbitální satelity, ale mají praktický vliv na typické topografické použití. Zpřesnění lze očekávat také u modelu gravitačního pole Země, které bylo naposledy revidováno v roce 2004.

GPS

Název systému GPS označuje zkratka NAVSTAR GPS, která je v celém názvu Navigation Satellite Timing and Ranging Global Position System neboli globální systém sloužící k určování polohy na Zemi. Systém je veden americkým ministerstvem obrany a původně sloužil pouze pro potřeby armády Spojených států amerických. Technologie GPS byla na počátku využívána jen jako přesný vojenský lokalizační a navigační prostředek pro sledování pozic vojenských jednotek, zaměřování cílů, apod., v 80. letech 20. století americká vláda rozhodla o jeho uvolnění i pro civilní účely.

Poté došlo k mohutnému rozšíření technologie GPS do všech oblastí lidské činnosti. Pro pokrytí signálem GPS celé Země bylo projektováno využívání 24 satelitních družic, ale potřeba získat přesnější polohu si vyžádala přidat osm satelitních družic, na aktuální počet 32 satelitních družic. Pro určení dvojrozměrné polohy (nejčastěji zeměpisná délka a

šířka) postačí příjem signálu z minimálně tří družic (výpočet tří pseudo-vzdáleností), pro určení trojrozměrné polohy (navíc výška) minimálně ze čtyř družic. Příjem menšího počtu družic znemožňuje výpočet polohy, vyšší počet družic naopak určení polohy dále zpřesňuje. Na základě pasivity GPS přijímačů (pouze přijímají data, ale nevysílají zpět k satelitům) je systém GPS schopen obsloužit neomezený počet uživatelů a nelze systém GPS zaměřit nepřítelem.

A-GPS

A-GPS nebo někdy také aGPS je zkratka Assisted GPS a slouží k rychlejšímu počátečnímu výpočtu pozice. Protože signál navigačních družic špatně proniká do budov, byl vyvolán zájem o bezdrátovou podporu GPS - Wireless Assisted-GPS (A-GPS) a o vysoce citlivé přijímače GPS. To vše s požadavkem na HW/SW integraci do mobilních zařízení. Je-li signál z družic slabý, nemusí se vždy podařit dekódování navigačního souboru dat, a v takovém případě mají udržet polohu v běžícím navigačním systému údaje z mobilní sítě. Jedná se o ekvivalentní údaje družic vyslané do časoměrně-dálkoměrného systému z mobilní sítě.

Technologie A-GPS je založena na IP technologii. Lokalizační server komunikuje přímo s mobilním přístrojem prostřednictvím protokolu vyšší vrstvy IP, který je zaveden do mobilní sítě. Ústřední a rádiová část mobilní sítě tak zůstávají beze změny. Výměna informací mezi telefonem a lokalizačním serverem probíhá přes běžná datová spojení, např. GPRS. Toto řešení tedy využívá protokolů a rozhraní, které již v ústřední a radiové části sítě existují. Již zmíněné lokalizační servery jsou schopny vyhodnotit pozici mobilního telefonu přímo na zemi, provést některé potřebné početní úkony a zjištěné výsledky spojit se signálem satelitu GPS. Tím tedy odpadne problém nedostupnosti satelitního signálu ve stíněných oblastech. Polohu v takovém případě nebude určovat satelit, ale telefon, odvodí se podle okolních stanic BTS. Tato informace se zpracuje a lokalizační server již zajistí komunikaci se satelitem. Konečné údaje jsou poté poskytovány mobilnímu telefonu opět po bezdrátové mobilní síti.

Pomocí technologie A-GPS získává mobilní zařízení data o oběžných drahách, frekvencích a provozních schopnostech satelitů z mobilní sítě, a dokáže tak analyzovat i slabší signály ze satelitů během několika sekund. U GPS trvá zjištění informace poněkud delší dobu. S A-GPS uživatel pouze zadá požadavek, výpočty se provedou na lokalizačních serverech a zpět je poslán pouze výsledek. Vše trvá jen několik málo sekund, což šetří nejen čas, ale také energii. Nevýhodou základní technologie A-GPS je její přesnost, kde je závislost na pokrytí dané oblasti BTS stanicemi. Tento nedostatek je vyřešen kombinací získání základní polohy dle mobilní sítě a následné zpřesnění pomocí signálu z orbitálních družic systému GPS. Čerpáno z [22].

Kapitola 4

Zpracování signálu ze senzorů

Tato kapitola popisuje zpracování signálu ze senzorů tak, jak probíhá na platformě Android. Zároveň jsou zde popsány chyby, které mohou při zpracování signálu nastat a jejich případné řešení. Text této kapitoly je čerpán z literatury [18], [19].

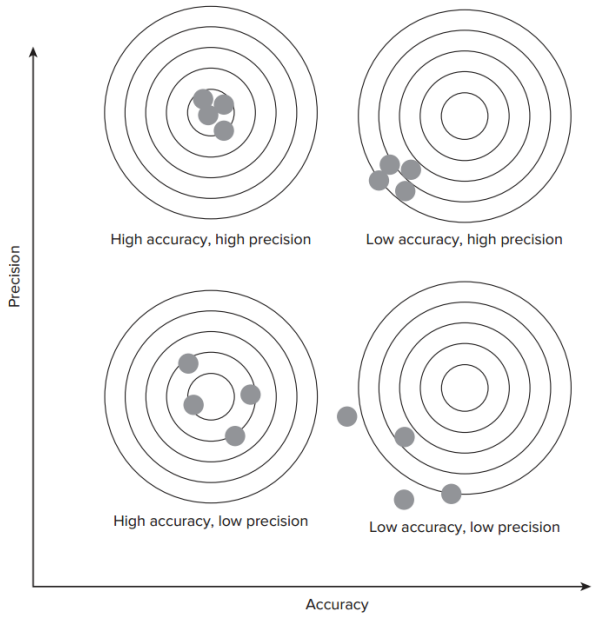
Některé senzory v mobilních zařízeních nejsou schopny získat absolutně přesné hodnoty měření. Poskytují data obsahující nesprávné hodnoty, například v závislosti na okolním rušení či digitálním šumu. Oba tyto vlivy napomáhají k získání chyb v měření a je nutné se jimi zabývat. Naopak jsou zde senzory, které měřením poskytují svá data s nepřesnými hodnotami či šumem, ale na celkovém výsledku se nic nemění. Příkladem takového senzoru je senzor měřící vzdálenost, který udává jen dvě hodnoty - blízko a daleko.

Naštěstí existují algoritmy a techniky pro řešení těchto problémů, které vedou k omezení chyb a nalezení filtru výstupních dat konkrétního senzoru tak, aby bylo možné získat co nejpřesnější hodnoty měření dané veličiny. Další možností je nalézt filtr pro usměrnění dat, která jsou výsledkem sloučení několika senzorů. Android obsahuje určitou sadu algoritmů, které jsou aplikovány na výstupy senzorů automaticky.

Signál ze senzorů na platformě Android je získáván v diskrétní podobě. Každý senzor poskytuje svá data s určitým časovým zpožděním, kde každý senzor má toto zpoždění jiné. Operační systém Android dovoluje nastavit rychlost, se kterou bude data z určitého senzoru získávat, ale nezaručuje, že budou data získána přesně v tomto intervalu. Mezi obecný interval po sobě následujících hodnot dat z jednoho senzoru je předpokládán 20 ms.

K naslouchání nějakému senzoru se lze přihlásit a pak jen pracovat s daty, které vrací. Data jsou vždy v definovaných jednotkách, které jsou pro každý senzor jiné. Každý senzor měří něco jiného, ale data se posílají přes jednu sběrnici a proto je definován formát, který musí všechny senzory dodržovat. Tento formát definuje, že data mohou nabývat maximálně tří hodnot. Například u akcelerometru či gyroskopu lze očekávat měření pro každou hlavní osu. V případě senzoru s jednou veličinou, například dříve zmíněný senzor světla, jsou data pouze v první složce a zbylé dvě jsou nulové.

Pro hodnocení přesnosti a opakovatelnosti senzorů je zapotřebí určit dvě čísla. Prvním je aktuální skutečná hodnota (například vlhkost, teplota nebo akcelerace), kterou se daný senzor snaží změřit. Druhým číslem je naměřená hodnota, kterou naměří určený senzor a poskytne ji jako výsledek. Vysoká přesnost znamená, že naměřená hodnota daným senzorem je blízko aktuální skutečné hodnotě. Podobně vysoká opakovatelnost ukazuje, že naměřené a získané hodnoty ze senzoru jsou shlukovány blízko nějaké hodnoty, ať už se jedná o skutečnou aktuální hodnotu či ne. Tento princip je popsán na obrázku 4.1, na kterém jsou popsány principy přesnosti (Accuracy) a opakovatelnosti (Precision). Obrázek je přejat z knihy [19, 8].



Obrázek 4.1: přesnost vs. opakovatelnost, přejato z [19]

Na obrázku je několik náčrtů určitého rozptylu hodnot, které znázorňují data poskytnutá ze senzoru. Ve středu jednotlivých rozptylů je vždy aktuální reálná hodnota.

4.1 Typy chyb

Při komunikaci s jakýmkoli senzorem, je důležité rozumět a počítat s možným výskytem chyb. Schopnost porozumět, proč může senzor navracet špatná data, je klíčová při navrhování algoritmu, který má tyto chyby eliminovat či alespoň zmírnit jejich špatný vliv.

- Systematické, náhodné chyby a chyby způsobené člověkem - Systematické chyby jsou chyby, které jsou způsobeny nepřesností měření. Jsou konstantní a v některých případech jim lze předejít, nebo je úplně odstranit, například kalibrací v neutrálním prostředí. Příkladem může být měření magnetometrem, které může být stále rušeno magnetem, který se nachází v blízkosti senzoru. Náhodné chyby, jako je například šum, nelze odstranit technikou kalibrace a velice špatně se tento typ chyby odhaluje. Důvodem je nahodilý výskyt, který se nemusí vyskytnout při vývoji. Chyby způsobené člověkem patří do stejné kategorie, ale není nutné se jimi dále zabývat.

- Šum - Jedná se o náhodné změny měřených veličin. Ačkoli samotný šum může být kategorizován do několika skupin (bílý šum, hnědý šum . . .) a statisticky kvantifikován. Tento typ chyb se velice špatně odhaduje. Pro částečné řešení a omezení vlivu těchto chyb se vytváří tzv.filtry dolní propusti.

- Drift chyby - Tento typ chyb popisuje pomalé, dlouhotrvající oddalování naměřených hodnot od hodnot reálných. Vyskytuje se u opakovaného čtení senzoru, který si pamatuje předchozí hodnotu a přičítá nebo odečítá od ni rozdíl. Nastává akumulace chyby způsobená uložením desetinných čísel v binární podobě.

- Zero offset - Pokud výstup měření některého senzoru není nula i v případě, že reálná hodnota je rovna nule, nastává tento typ chyby. Někdy je chyba nazývána jako typ bias (posun). Tento jev lze pozorovat při měření hodnot z akcelerometru mobilního zařízení, které je položeno na rovné ploše (například na stole). Výstupem by měla být ve dvou osách nula a v poslední ose gravitační konstanta 9,80665 $\frac{m}{s^2}$, tohoto přesného výsledku se bohužel nikdy nedočkáme. Podobně lze pozorovat hodnoty získané z gyroskopu, kde v klidové poloze nezískáme přesně nulu ve všech osách.
- Časová zpoždění nebo ztráta dat - Jelikož platforma Android není real-time operační systém (RTOS), tak některé naměřené hodnoty mohou být zpožděny, získány v jiný časový okamžik než jsou očekávány. Data se mohou i ztratit, pokud je zařízení plně vytíženo jinou činností.

4.2 Techniky řešení chyb

Předchozí část popisovala jednotlivé typy chyb, které mohou nastat při práci s daty, která jsou získávána používáním senzorů na platformě Android. Při zpracování údajů ze senzorů je vhodné, aby aplikace byla schopna některé typy chyb, které jsou přítomny, redukovat a pracovat tak s co nejpřesnějšími daty. Následující část popisuje techniky, které se snaží řešit některé typy chyb při zpracování signálu (dat) ze senzorů.

- Re-zeroing - Pokud je naměřen určitý offset (posun) v naměřených datech, který se vyskytuje ve všech měřeních, může být vhodné provést vynulování senzorového měření. Jedná se o jednoduché uložení kalibrované hodnoty. Kalibrace je buď dopředu známá, nebo je nabízena uživateli explicitně (například tlačítkem v aplikaci). Kalibrace se provede na rovné ploše, kde se výsledná hodnota uloží a při každém měření je tato hodnota odečtena od naměřené hodnoty. Smysl kalibrace je vhodný pro přesná měření, například u senzorů typu gyroskop či akcelerometr.
- Filtry - Dolní propusť filtruje vysoko-frekvenční signál nebo šum. Filtrovaná data jsou na první pohled "vyhlazená". Horní propusť filtr omezuje pomalý drift a offset a mění vyšší frekvence. Ořízne frekvenci nad nebo pod hranicí dat, která jsou filtrována. Pásmový filtr odstraňuje nízkofrekvenční a vysokofrekvenční data a zanechává pouze rozsah frekvence našeho zájmu.
- Sensor fusion (získání údajů z více senzorů) - Při použití více senzorů, kde se využívají přednosti každého jednotlivého senzoru a potlačení jeho slabostí. Příkladem může být využití akcelerometru pro získání relativně přesného směru zařízení, ale nelze tímto senzorem získat natočení do hlavních světových stran. S tímto problémem naopak může pomoci geomagnetický senzor. Při skládání jednotlivých senzorů do jednoho celku je nutné brát zřetel na jejich přednosti a nedostatky. Součástí nedostatků je jejich přesnost a chyby, které je nutné řešit globálně. Tímto je myšleno například využití metody re-zeroing, kterou získáme kalibraci akcelerometru na celková data z obou senzorů.

4.3 Filtry

Filtry na mobilních zařízeních jsou již ve většině případů integrovány v samotném zařízení. Přesto se vyskytnou požadavky na aplikaci nestandardní a je zapotřebí data filtrovat ještě

jednou nebo i vícekrát. Získaná data teoreticky obsahují menší procento chyby. Následuje popis několika metod filtrace jako je dolní a horní propusť.

Dolní propusť

Ačkoli jsou senzory v mobilních zařízeních stále zdokonalovány a jejích výstup neustále přesnější, jsou zde aplikace, které vyžadují formu vyhlazených či průměrných hodnot. Známé jako filtrování dolní propustí, které odstraňuje vysokofrekvenční šum a propouští nízké frekvence nebo pomalé změny na daném signálu (na daných hodnotách).

Horní propusť

Jedná se o frekvenční lineární filtr, který nepropouští signál o nízkých frekvencích. Převážně je tento typ filtru používán v audio technice, kde usměrňuje vysoké tóny nebo pro odrušení nízkofrekvenčního šumu. V oblasti mobilních zařízení nachází své uplatnění při práci se senzory, zvlášť u akcelerometru. S využitím horní propusti lze z naměřených dat akcelerometru odstranit gravitační zrychlení, které je při měření vždy přítomné.

Váhové vyhlazování

Jednoduchá a velmi používaná metoda pro filtrování dolní propustí, na vyhlazení data na základě váhového koeficientu. Kde následující (nově) naměřená hodnota je částečně závislá na předcházející. Váhový (vyhlazovací) koeficient se používá dle následujícího vzorce 4.1.

$$new\_value = last\_value + x_i.a - last\_value.a \tag{4.1}$$

Jinak řečeno poslední vypočítaná hodnota je připočtena k x_i - poslední získaná hodnota, která je ovlivněna váhovým parametrem a. Předchozí hodnota je opět ovlivněna stejným váhovým parametrem a je od celkové sumy odečtena. Pokud je váhový koeficient a blízko hodnotě 1, nová hodnota bude x_i, a pokud je a blízko hodnotě 0, nová hodnota se nezmění - tento princip umožňuje libovolnou úroveň vlivu na novou hodnotu. Algoritmus po matematické úpravě dostává tvar, který je popsán rovnicí 4.2.

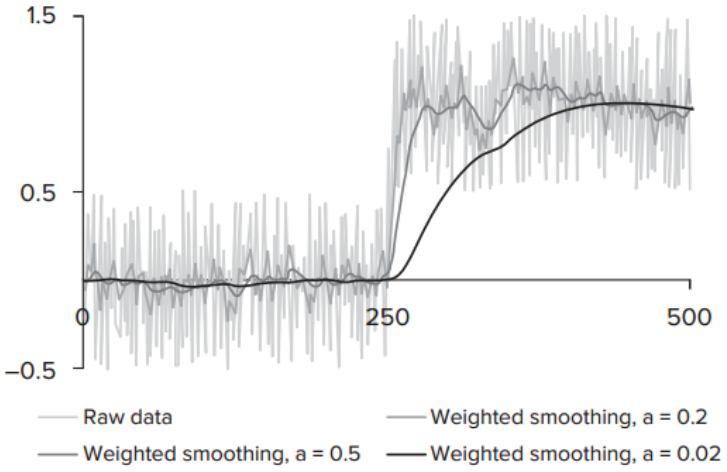
$$new\_mean = last\_value.(1 - a) + x_i.a \tag{4.2}$$

Na obrázku 4.2 je zobrazena závislost filtru dolní propusti na parametru a, který byl popsán výše. Originální signál pochází z akcelerometru a je upravován tímto filtrem. Při změně tohoto parametru lze pozorovat i změnu signálu. Tato změna je znázorněna pro hodnoty 0.5, 0.2 a 0.02.

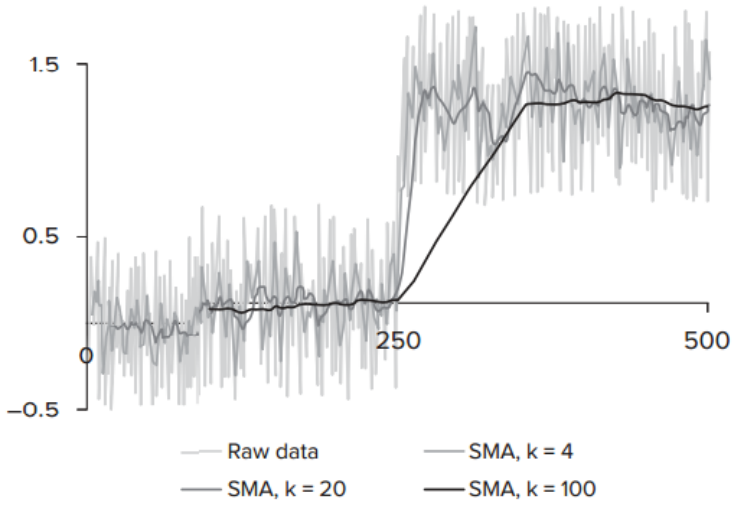
Pohyblivý průměr

Tento jednoduchý filtr SMA (Simple Moving Average), který lze napsat na pár řádků kódu, poskytuje vyhlazování tzv. špiček v signálu. Někdy je tento typ filtru nazýván kolébavým nebo běžícím průměrem. Princip této metody je založen na získání aritmetického průměru posledně získaných k dat ze vstupního proudu. Hodnota k zde vystupuje jako velikost posuvného (klouzavého) okna. Metoda nepracuje dokud není k dispozici alespoň k vstupních dat. Pro prvních $k - 1$ hodnot, lze uvažovat libovolný průměr (například 0), nebo výpočet pohyblivého průměru může být odložen, až bude nashromážděno k hodnot.

Na obrázku 4.3 je podobně jako u předchozí metody znázorněn signál z akcelerometru, který je nezměněn a zároveň několik signálů, které jsou upraveny filtrem dolní propusti. Je



Obrázek 4.2: váhové vyhlazování - závislost parametru `a`, přejato z [19]



Obrázek 4.3: pohyblivý průměr - závislost parametru `k`, přejato z [19]

zde znázorněna změna signálu v závislosti na změně koeficientu k, který byl popsán dříve vzorcem 4.2. Byly zvoleny hodnoty koeficientu na 4, 20 a 100.

Kapitola 5

Internetové technologie a služby

Obsahem kapitoly je stručný popis internetových (webových) technologií, které jsou v dnešní době používané pro sdílení obsahu. V první části budou představeny jazyky, které slouží pro uspořádání obsahu dokumentu nebo internetové stránky. Především se jedná o značkovací jazyk HTML a kaskádové styly CSS, umožňující upravovat vzhled dokumentu. Po lehkém úvodu je popsán skriptovací jazyk JavaScript. Tento jazyk je zmiňován především z pohledu jeho využití, například při tvorbě dynamických akcí na straně uživatele. Pokud se mluví o uživatelské straně, tak je ve většině případů myšleno v prohlížeči internetového obsahu. Se skriptovacím jazykem JavaScript se setkáme u zobrazování map, který daný jazyk používá pro interaktivní komunikaci s uživatelem. Příkladem může být posuv mapy nebo přiblížení konkrétní polohy.

Kapitola obsahuje popis skriptovacích jazyků, které jsou využívány pro tvorbu dynamického webu. Především se jedná o jazyky PHP, ASP či Python. Skriptovací jazyky mají společnou vlastnost, že jsou vždy prováděny na straně serveru a uživatel s nimi není v přímém kontaktu. Existuje mnoho jazyků umožňujících dynamicky sestavovat webový dokument, jmenované patří mezi nejpoužívanější. Mají podporu práce s databázemi, které jsou v této kapitole zmíněny také. Především se jedná o technologie MySQL či SQLite. U jednotlivých databázových systémů jsou popsány jejich výhody, a nevýhody a pro jaká data jsou vhodné. Mezi nejpoužívanější kombinace lze zařadit jazyk PHP a databázový systém MySQL. Vývojáři jazyka PHP vytvořili velice úzkou spolupráci s databázovým systémem MySQL. Vývojáři malých a středně velkých systémů volí jako implementační technologie právě PHP a MySQL, nebo alespoň nějaký framework postavený na jmenované technologii.

Poslední částí kapitoly jsou mapy dostupné na internetu, které lze využít pro aplikace webové nebo mobilní. Ideální mapy jsou takové, které jsou dostupné zdarma, jsou kvalitně zpracované a jsou dostupné pro velké množství platforem. Velký důraz je kladen na dostupné API rozhraní pro komunikaci mezi aplikací a mapami.

5.1 HTML, CSS

HTML

První definici jazyka HTML vytvořil v roce 1991 Tim Berners-Lee jako součást projektu WWW, který měl umožnit vědcům zabývajícím se fyzikou vysokých energií komunikaci a sdílení výsledků výzkumu po celém světě. Ne náhodou proto celý projekt vznikal v CERNu (Centre Européenne de Rechere Nucléaire, Evropské centrum jaderného výzkumu), který leží na švýcarsko-francouzských hranicích nedaleko Ženevy. Tato verze HTML je známá

pod označením HTML 0.9. Umožňovala text rozčlenit do několika logických úrovní, použít několik druhů zvýraznění textu a zařadit do textu odkazy a obrázky. Berners-Lee při návrhu HTML nepředpokládal, že by autoři WWW stránek museli tento jazyk znát.

První verze softwaru podporující zobrazení WWW stránek byla napsána pro operační systém NextStepa, který obsahoval jak prohlížeč, tak i integrovaný editor. Když však Marc Anderssen se svými kolegy z NCSA (National Center for Supercomputing Applications) psal známý prohlížeč Mosaic, považoval za příliš obtížné implementovat do programu rovnou i editor HTML. Díky tomuto rozhodnutí a tomu, že ne každý provozuje na svém počítači NextStep, je dnes nutné, aby autoři profesionálních stránek znali HTML. Požadavky uživatelů na WWW stránky vzrůstaly, a tak producenti různých prohlížečů obohacovali HTML o některé nové prvky. Aby byla zachována kompatibilita mezi jednotlivými modifikacemi HTML, vytvořil Berners-Lee pod hlavičkou IETF (Internet Engineering Task Force) návrh standardu HTML 2.0, který zahrnoval všechny v té době běžně používané prvky HTML. Verze HTML 2.0 má zároveň dvě úrovně. První z nich (Level 1) pouze rozšiřuje předchozí verzi HTML. Level 2 navíc definuje práci s formuláři. Specifikaci HTML 2.0 nalezneme v RFC dokumentu číslo 1866.

Další rozšíření jazyka známé jako HTML+, zahrnují zejména rozšíření HTML o vytváření tabulek a matematických vzorců. Rovněž se zde objevují prvky, které umožňují precizněji kontrolovat výsledný vzhled textu a výrazně lepší obtékání obrázků textem a styly dokumentů. Dave Raggett z laboratoří Hawlett-Packard jazyk HTML formalizoval a vytvořil jeho deklaraci DTD (Document Type Declaration) v jazyce SGML. Na jaře roku 1995 tak vznikl návrh standardu HTML 3.0. Některé prvky HTML 3.0, jako např. tabulky, podporovaly novější verze prohlížečů Mosaica Netscape. Kompletní podporu pro všechny rysy HTML 3.0 nabízel pouze experimentální prohlížeč Arena. Ten je bohužel k dispozici pouze pro operační systémy typu Unix. Na počátku roku 1996 již bylo jasné, že HTML 3.0 bylo tak mohutným skokem vpřed, že se nenašel nikdo, kdo by dokázal implementovat prohlížeč s jeho podporou.

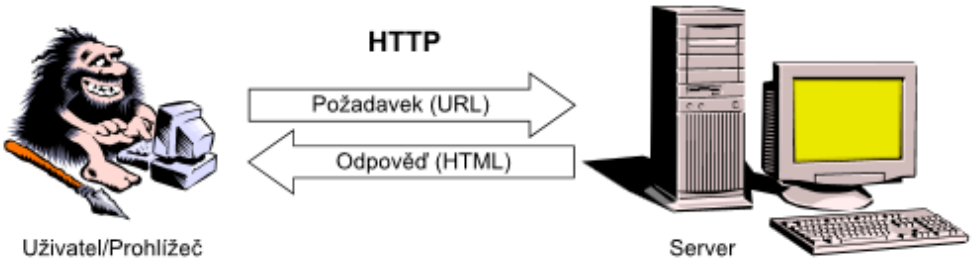
Vývoj standardů Webu v té době již koordinovalo konsorcium W3C, jehož členy jsou mimo jiné přední softwarové firmy. Členové W3C se tedy shodli na vlastnostech, o které rozšíří HTML 2.0, a vytvořili tak HTML 3.2. HTML 3.2 však zdaleka neobsahuje vše z HTML 3.0. Z verze 3.0 zůstaly v podstatě jen tabulky. Ostatní nové prvky HTML 3.2 jsou doplňky, které v té době podporovaly nejnovější prohlížeče. Opakoval se tedy v podstatě stejný postup jako při vzniku verze 2.0. Jazyk se sjednotil na průniku možností těch nejrozšířenějších prohlížečů. Kromě tabulek přibyly ve verzi 3.2 zejména možnosti lepší kontroly formátování, včetně mnohem volnějšího výběru použitých druhů písma a logický ústupek požadavkům na graficky perfektně vypadající stránky. Další podstatné rozšíření se týkalo podpory Java-apletů.

Tato verze HTML nese kódové jméno Wilbur a od ledna 1997 je doporučením konsorcia W3C, které znamená to, že by ji měli všichni používat, aby byla ve webu zajištěna kompatibilita. Posledním hitem, který s HTML úzce souvisí, jsou kaskádové styly dokumentů (CSS), které jsou doporučením W3C od prosince 1996. Kaskádové styly jsou popsány v následující části této kapitoly. Na jaře roku 1997 zveřejnilo W3C další plány na rozšíření HTML pod kódovým názvem Cougar. Cougar v sobě zahrnuje HTML 3.2 společně s běžně používanými konstrukcemi jako jsou rámy, skripty a obecné vkládání objektů. Dalšími novinkami byla podpora vícejazyčných dokumentů. Na začátku července 1997 uveřejnilo W3C návrh HTML 4.0, který vznikl drobnými úpravami Cougaru a vytvořením jediného komplexního dokumentu popisujícího návrh standardu.

Aktuálně je ve vývoji verze jazyka HTML 5, jejíž specifikace jsou dostupné a začínají se

pomalu stávat součástí moderních prohlížečů. Jazyk si klade za cíl, mimo používání dalších nových značek (tagů), také integrovat mnoho služeb, mezi kterými lze nalézt vzdálenou komunikaci, přehrávání videa a audia. Na základě těchto požadavků vznikají spory o to, který audio či video kodek bude použit. Velkou roli v tomto sporu hrají hlavně vývojáři internetových prohlížečů. Některé prohlížeče mají uzavřené kontrakty o poskytování služeb konkrétních kodeků, a tím je vývoj v tomto směru omezován. Konečná specifikace jazyka by měla být schválena do konce roku 2014. Texty vychází z literatury [17] a informace týkající se jazyka HTML 5 jsou čerpány z [14].

Příklad komunikace mezi uživatelovým počítačem (internetovým prohlížečem) a serverem (poskytovatelem obsahu) je znázorněn na obrázku 5.1. Na tomto obrázku je na levé části zobrazen uživatel, který se dotazuje (požadavek URL) na obsah nějaké internetové stránky a v pravé části je server, který na daný požadavek odpovídá a zasílá uživateli HTML dokument, který je předán uživateli ve formě zobrazení na monitoru. Obrázek je přejat z literatury [17].



Obrázek 5.1: komunikace uživatele se serverem a přenos dat v rámci HTML, přejato z [17]

CSS

Tato část volně navazuje na předešlý článek o jazyku HTML a obsah je čerpán z literatury [20]. Koncept kaskádových stylů je takový, že doplní dokument HTML (XHTML či jiný značkovací jazyk) o vlastnosti formátování. Tato formátování zahrnují například barvu, písmo nebo určení polohy samotných částí dokumentu. Cílem je propojit všechny stránky do jedinečného vzhledu na základě šablony stylů. Definovat všechny styly jednou, namísto opakovaně v každém dokumentu. Důvodem je oddělit obsahovou a vzhledovou část dokumentu a usnadnění úpravy vzhledu. Technologie stylů se nazývá CSS, což je zkratka pro Cascading Style Sheed neboli kaskádové styly. CSS je jazyk, který definuje styl jako je například typ písma, barva a poloha. Tyto definice jsou pro webový prohlížeč, který na základě těchto informací naformátuje prvky dokumentu.

CSS styly je možné uložit přímo do souboru dokumentu a stávají se součástí HTML stránky, bývá dobrým zvykem oddělovat styly do samostatného souboru a tím dosáhnout výše zmiňovaného oddělení obsahové části od vzhledu. Pravidlo stylu je formátovací instrukce, která může být použita (přiřazena) k prvku na webové stránce, příkladem takového prvku může být odstavec textu či odkaz. Styl se zpravidla skládá z jednoho nebo více vlastností a příslušné přidružené hodnoty. Kaskádové styly za sebou mají dlouhý vývoj, stejně jako jazyk HTML, kde od roku 1996 byla představena CSS úrovně 1 a ve velice krátké době upravena na úroveň 2. Verze 2 je od roku 1998 stále nejpoužívanější verzí kaskádových stylů,

ale je postupně nahrazována úrovní 3, která si získává svou oblibu a podporu v moderních internetových prohlížečích.

5.2 Skriptovací jazyky

Pro interaktivní úkoly je nutné použít sofistikovanější jazyk, programovací jazyk. Ačkoli mnoho programovacích jazyků jsou komplexní, skriptovací jazyky jsou obvykle jednoduché. Webové skriptovací jazyky umožňují kombinovat programování s HTML jazykem a vytvářet tak interaktivní internetový obsah. Výstupem internetových (webových) skriptovacích jazyků je primárně dokument ve značkovacím jazyce HTML. Obsahem této části je popis nejpoužívanějších skriptovacích jazyků pro tvorbu dynamického obsahu webových dokumentů. Záměrně byly vybrány dvě nejznámější technologie pro tvorbu menších a středně velkých projektů. Není cílem této práce vytvářet velké systémy.

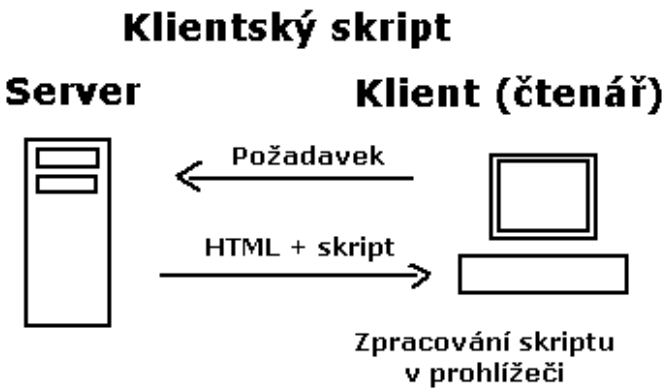
JavaScript

Jak bylo zmíněno v předešlé části, internet, konkrétně internetový obsah ve formátu www dokumentů, začínal pouze jako textové médium. První prohlížeče nepodporovali zobrazování obrázků či animací. Od počátků uplynulo mnoho let, jelikož dnešní internetový obsah je plný bohatých vizuálních a interaktivních funkcí a zároveň užitného obsahu, mezi který lze zařadit grafiku, zvuky, animace a video. Webové skriptovací jazyky, kam JavaScript beze sporu patří, určují způsob, kterým jsou webové dokumenty obohaceny o vzájemnou interakci s uživatelem.

Byl představen jazyk HTML [17, 14, 12], který je základním kamenem internetových dokumentů a popisuje základní obsah dokumentu se základním členěním stránky na jednotlivé části. Tento jazyk nemůže reagovat na uživatele a provádět rozhodnutí nebo automatizovat opakující se úkoly. Jazyk JavaScript nabízí interaktivitu, rozhodování, automatické vyhodnocení daného problému a mnoho dalších úkonů spojených s internetovým obsahem. JavaScript obvykle běží v prohlížeči internetového obsahu. Nejedná se tedy o jazyk běžící v pozadí na serveru. Z tohoto důvodu je popsán v této části kapitoly. V další části jsou popsány skriptovací jazyky, které běží v pozadí internetového provozu na serveru a generují internetový obsah na základě požadavků uživatele. Nyní zpět k jazyku JavaScript, který interaktivně reaguje na uživatele ihned bez nutnosti komunikovat přes internet s dalším počítačem (serverem).

JavaScript byl vyvinut společností Netscape Communications, která jej integrovala do svého internetového prohlížeče Netscape. JavaScript byl první webový skriptovací jazyk, který byl podporován prohlížeči a zároveň si ve velmi brzké době získal oblibu vývojářů, kteří jej v mírně modifikované formě používají dodnes. Mezi základní funkce jazyka lze zařadit zobrazování zpráv uživateli (návštěvníkovi internetové stránky) jako součást webové stránky v prohlížeči, nebo například kontrolu obsahu formuláře před samotným odesláním na server. Další funkcionalitou je animace nebo tvorba obrázků s ohledem, na pohyb myši, tvorba tzv. banerů, které interagují s uživatelem, než jen zobrazení klasického statického obrázku. Detekce prohlížeče, a s tím související použití funkcí v závislosti na verzi prohlížeče, zjištění nainstalovaných doplňků, úprava části nebo celého webového dokumentu bez nutnosti znovu-načtení dokumentu ze serveru a mnohé další funkcionality nabízí jazyk JavaScript. Podrobnější informace lze nalézt v literatuře, ze které tato kapitola čerpá [11, 20].

Primárním důvodem pro vznik jazyka JavaScript byla nedostupnost interaktivity a automatizace stále se opakujících úkolů. Především byl v dané době kladen důraz na co nejmenší objem dat přenesených mezi serverem a uživatelem. A na základě těchto požadavků byl představen skriptovací jazyk, který se vykonává na straně klienta a nezasahuje do další síťové komunikace. Názorná ukázka funkčnosti je na obrázku 5.2, který je převzat z internetové stránky [2].



Obrázek 5.2: komunikace klient-server, přejato z [2]

Rostoucí zájem o přístup do sítě internet a s postupným nárůstem rychlosti sítě, je tento přenos zanedbatelný. Obliba jazyka nezůstala pouze v minulosti, ale i v dnešní době je jazyk neustále vyvíjen a doplňován novými funkcemi. Nezůstává pouze u samotného jazyka, ale jsou dostupné frameworky pro různé typy animací, úpravy vzhledu, přehrávání audio či video obsahu. V této kapitole lze nalézt příklad použití jazyka JavaScript na službách poskytující mapy. JavaScript se stal primárním jazykem při vývoji mobilních aplikací pro více platforem. Především se jedná o různé frameworky, ve kterých je mobilní aplikace tvořena především jazyky HTML a CSS, jež definují grafické rozhraní aplikace a jazyk JavaScript implementující kompletní funkcionalitu a logiku dané aplikace.

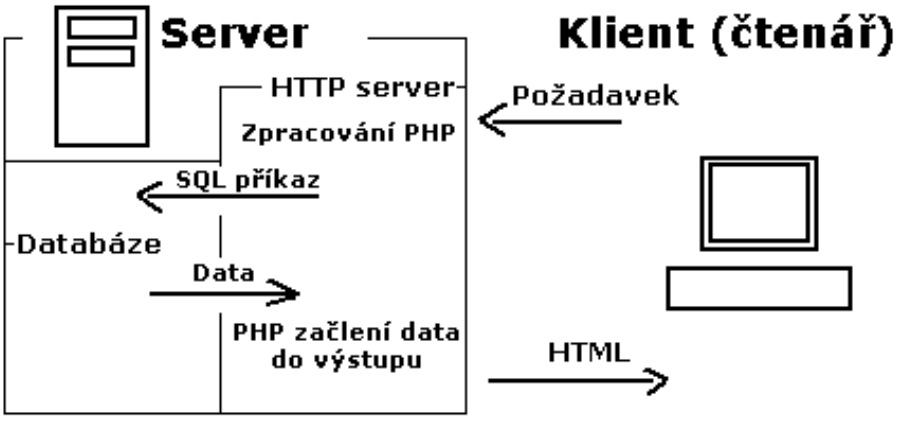
PHP

Na počátku zrodu systému stál Rasmus Lerdorf. Psal se rok 1994 a Rasmus si ve volném čase vytvořil v programovacím jazyce Perl jednoduchý systém pro evidování přístupu k jeho stránkám. Jelikož neustálé spouštění interpretu Perlu velmi zatěžovalo www server, přepsal autor systém do jazyka C. Ačkoliv byl celý systém původně určen pro osobní Rasmusovo použití, zalíbil se i ostatním uživatelům serveru a začali ho používat. Systém se stal oblíbeným a používalo jej stále více uživatelů. Ti přicházeli s požadavky na vylepšení celého systému. Autor proto systém rozšířil, doplnil o dokumentaci a uvolnil jej pod názvem Personal Home Page Tools, který se později změnil na Personal Home Page Construction Kit. V téže době autor zprovoznil elektronickou konferenci, která sloužila jako prostor pro výměnu zkušeností mezi uživateli systému. Kromě již zmíněného systému pro evidování přístupů ke stránkám vytvořil pan Lerdorf i nástroj, který umožňoval začleňování SQL dotazů do stránek, vytváření formulářů a zobrazování výsledků dotazů.

Program, který umožnil zpřístupnění databází na webu, se jmenoval Form Interpreter (FI). Celosvětovou proslulost si získal systém PHP/FI 2.0. Tento systém vznikl spojením dvou předchozích programů autora. V této podobě se jednalo o jednoduchý programovací

jazyk, který se zapisoval přímo do HTML stránek. PHP//FI 2.0 se rozšířilo opravdu po celém světě a pracovalo i na mnoha českých serverech. První verze pro systém Windows byla 3.0 a přidala mnoho nových funkcí. Projekt se rozšířil a na jeho vývoji pracuje Rasmus Lerdorf s několika dalšími vývojáři. Obsah zkratky PHP nyní zcela ztrácí na svém původním významu a doporučené označení celého systému je hypertextový preprocesor PHP (v originálním znění PHP: Hypertext Preprocessor). Nejčastěji se uvádí pouze zkratka PHP, která daný systém vystihuje bez ohledu na konkrétní význam. Do jazyka jsou přidávány stále nové možnosti, které umožňují efektivnější vytváření internetových aplikací.

PHP je skriptovací jazyk určený pro vývoj webových aplikací, který je vždy spuštěn na straně serveru. Schéma komunikace uživatele přes internetový prohlížeč se serverem obsahující PHP je znázorněn obrázkem 5.3. Na obrázku je v pravé části klient (uživatel s webovým prohlížečem) požadující obsah webového dokumentu. Na požadavek reaguje server obsahující PHP a případně pracující s databázovým systémem. Data jsou zpracována interpretem PHP a je vytvořena HTML stránka, která je následně zaslána klientovi, který si může výsledek prohlédnout ve svém prohlížeči.



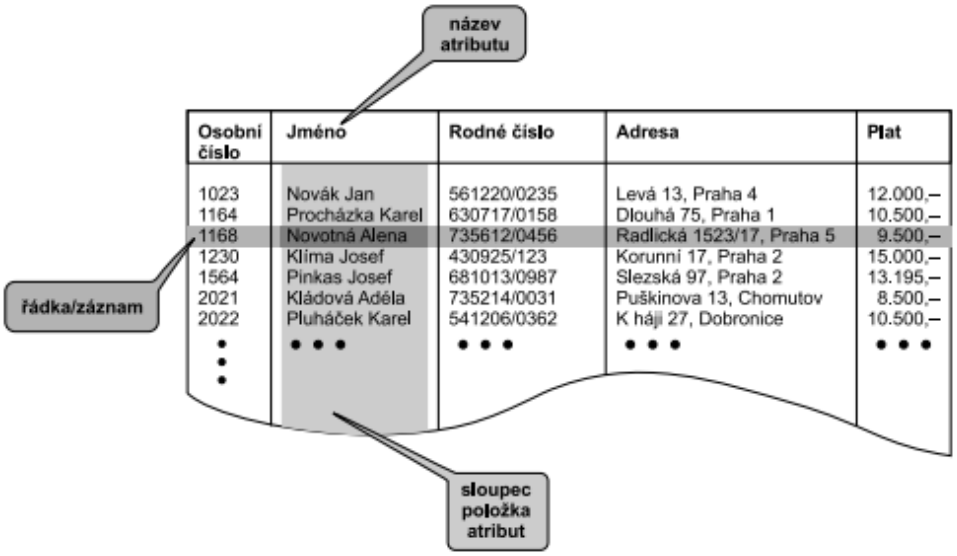
Obrázek 5.3: komunikace uživatele se serverem a přenos dat v rámci PHP, přejato z [2]

Nejčastěji je nasazován pro tvorbu dynamického webu, ale je možné jej použít jako univerzální programovací jazyk. Příkazy jazyka jsou vloženy přímo do značkovacího jazyka HTML a dynamicky je tvořen obsah jednotlivých částí. Jazyk za dob svého vývoje dosáhl velkých změn, ale některé původní záměry a funkce jsou zachovány, dodnes mezi nejznámější patří syntaxe podobná jazyku Perl, možnost psát příkazy jazyka mezi značkovací HTML jazyk nebo využívat proměnné z vyplněného formuláře. Aktuálně je nejnovější verze jazyka PHP 5, která umožňuje využít několika paradigmat včetně objektově orientovaného. Jádro jazyka pojmenované Zend bylo přepracováno a z původní první verzí již nemá téměř nic společného. Dosáhlo se desetinásobného zrychlení vykonávání příkazů oproti první verzi a vývoj stále pokračuje. Oblíbenost jazyka dokládá více než 2,1 milionu serverů, na kterých je PHP interpret aktuálně spuštěn. Pro jazyk je vytvořeno mnoho frameworků, které zabezpečují jednodušší a rychlejší vývoj aplikací. Více se o technologii PHP lze dočíst v literatuře [7].

5.3 Databáze

V této části je rozebrána problematika ukládání dat, kterou je z dnešního pohledu možné řešit dvěma způsoby a to ukládat do samostatných souborů či využít technologii databází. Pryč jsou doby, kdy pro kvalitní prezentaci firmy na webu stačilo pár statických stránek provázaných odkazy. Většina dnes veřejných webových aplikací pracuje s velkým množstvím dat, které jsou poměrně dobře strukturována a ukládají se proto do databází. S databází přichází mnoho dalších pojmů, které je nutné znát pro pochopení správného principu databází a následného správného vytváření tabulek.

Databázi si lze představit jako místo, kam se ukládají všechny potřebné údaje. Přístup k údajům uloženým v databázi obstarává program, kterému se říká SŘBD neboli Systém Řízení Báze Dat. Tento poněkud krkolomný název vznikl přeložením původního anglického termínu DBMS neboli DataBase Management System. Mezi SŘBD patří takové programy jako Oracle, MS SQL Server, Sybase, Informix, Progress či InterBase. Vyjmenované programy se řadí mezi placený software, vzhledem k zaměření projektu je vhodné se poohlédnout po alternativách. Mezi zdarma dostupné lze zařadit mSQL, MySQL a PostgreSQL.Převážná většina dnes používaných SŘBD při uspořádání údajů v databázi vychází z relačního modelu dat. Název tohoto modelu vychází z relační algebry, což je matematický aparát, na kterém relační model dat staví. V tomto modelu jsou údaje uspořádány do tabulek . Tabulka zpravidla shromažďuje údaje o jednom druhu objektů. Můžeme tak mít například tabulku s osobními údaji zaměstnanců. Jednotlivé řádky odpovídají jednotlivým zaměstnancům. Sloupce pak obsahují informace o pracovnících. Sloupce tabulky jsou obvykle nazývány v databázové terminologii položky nebo atributy. Jednotlivé řádky se pak nazývají záznamy. Předchozí pojmy jsou graficky znázorněny na obrázku 5.4, který je přejat z literatury [7].



Obrázek 5.4: tabulka v relačním modelu dat, přejato z [7]

Aby bylo možné s tabulkami a v nich uloženými údaji pracovat, musí být nějak jednoznačně identifikovány. Každý sloupec je proto pojmenován vlastním názvem. Tento název je poté použit, když je nutné se odvolávat na obsah určitého atributu, a ne na celý záznam.

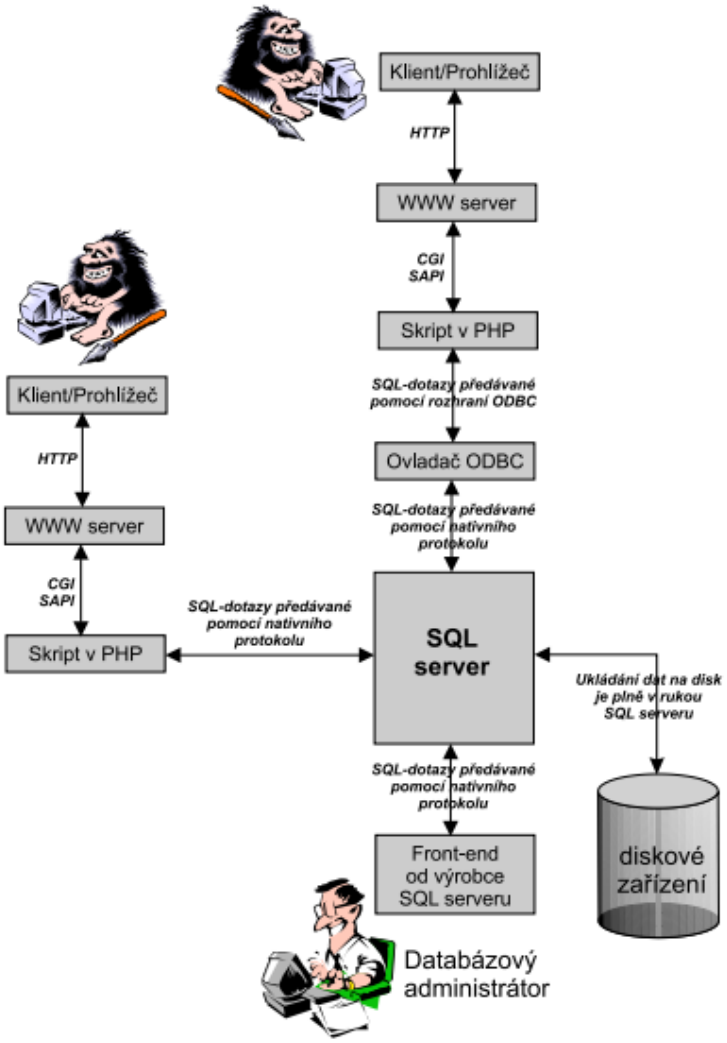
Častou operací prováděnou v tabulkách je změna obsahu jednoho atributu u určitého záznamu. Pro provedení této operace je však nutné mít k dispozici způsob, jak jednoznačně určit požadovaný záznam. Pro tyto účely by každá tabulka měla obsahovat tzv. primární klíč. Primární klíč je atribut, jehož hodnota je pro každý záznam jedinečná. Pro každý atribut tabulky se určuje, jaký typ dat může obsahovat. Mezi nejběžněji používané typy patří celá čísla, znakové řetězce a logické hodnoty (ano/ne). Další velmi často používané typy jsou reálná čísla, měnové údaje, datum a čas. Mnoho SŘBD podporuje i složitější typy, jako je obrázek, video či audio klip.

Databáze může samozřejmě obsahovat větší množství tabulek, záleží na tom, co vše za údaje je potřeba do databáze zaznamenat. Každá tabulka má proto své jméno, které ji v rámci databáze jednoznačně identifikuje. Přístup k údajům uloženým v databázích obstarává SŘBD. Aby mohly být údaje z databáze přístupné ostatním aplikacím, musí SŘBD nabízet rozhraní, pomocí kterého s ním mohou spolupracovat ostatní programy. Způsob komunikace se SŘBD je velice obdobný komunikaci s webovým serverem. Dnes je SŘBD nejčastěji nepřetržitě spuštěn jako démon (na Unixu) nebo jako služba (ve Windows) a na určitém socketu očekává požadavky klientů (ostatních aplikací). Na tyto požadavky pak odpovídá. Lze vyhodnotit, že i zde funguje osvědčený model klient/server. V roli serveru je nyní SŘBD a někdy se mu proto také říká databázový server. Pro zadávání požadavků na databázový server aplikace nejčastěji používají jazyk SQL (Structured Query Language). Tento jazyk prošel dlouhým vývojem a v různé míře jej dnes podporují téměř všechny běžně používané databázové servery. Někdy se proto databázovým serverům říká zjednodušeně SQL servery. Jazyk SQL nabízí vše potřebné pro vytváření, modifikování a rušení tabulek a pro práci s údaji v tabulce, jako je například vyhledávání, přidávání, modifikování a mazání údajů.

V roli klienta pro SQL server může vystupovat i skript zapsaný ve skriptovacím jazyce PHP zmíněném výše. To znamená, že skripty mohou obsahovat příkazy zapsané v jazyce SQL a zpracovávat jejich výsledky po provedení na SQL serveru. Nic tedy nebrání tomu, aby byl přes Web zpřístupněn obsah nějaké databáze. Bohužel v praxi je vše samozřejmě o něco složitější. Každý SQL server má svůj vlastní protokol, kterým s ním může klient komunikovat. Pokud má klient umět komunikovat s více různými servery, musí podporovat více protokolů. To není zrovna nejšťastnější řešení, a proto na platformě Windows vzniklo rozhraní ODBC. To slouží jako prostředník mezi klientskou aplikací a databázovým serverem. Rozhraní ODBC se volá jednotně a ODBC ovladač pak požadavek předá databázovému serveru pomocí správného protokolu. Velkou výhodou ODBC tedy je, že stejným způsobem můžeme přistupovat k libovolné databázi. Pokud se tedy z nějakého důvodu změní SQL server, na kterém běží naše aplikace, nemusíme měnit v PHP skriptech žádný kód. Počáteční nevýhodou, která mluvila proti použití ODBC, byl nižší výkon oproti nativním ovladačům. Staré ODBC ovladače sloužily pouze jako mezistupeň mezi aplikací a nativním protokolem databáze. Novější ODBC ovladače jsou však optimalizovány a k databázovému serveru přistupují přímo. Jejich výkon je srovnatelný s použitím nativních ovladačů. Některé novější SŘBD obsahují jako svůj jediný nativní protokol právě ODBC. Předchozí dva typy komunikace jsou ilustrovány obrázkem 5.5, který je přejat z literatury [7].

MySQL

MySQL je databázový systém vytvořený švédskou společnosti MySQL AB, kterou později odkoupila společnost Oracle Corporation. Je považován za úspěšného průkopníka dvojího licencování. K dispozici je jak pod bezplatnou licencí GPL, tak pod komerční placenou li-



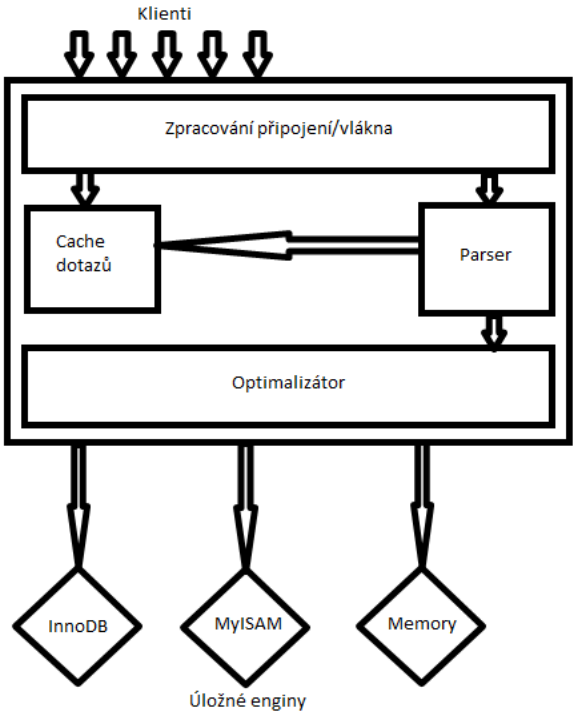
Obrázek 5.5: komunikace s SQL serverem, přejato z [7]

cencí. MySQL je multiplatformní databáze. Komunikace je tvořena SQL jazykem, který se stal pro databázové systémy standardem. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními. Pro svou snadnou implementovatelnost a nastavitelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně dostupný software, má vysoký podíl na v současné době používaných databázích. Velice populární u webhostingových poskytovatelů je kombinace Linux, MySQL, PHP a Apache jako základní software webového serveru.

MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení. Jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery, a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, kdy začaly nejčastějším uživatelům produktu, programátorům webových stránek, již poněkud scházet.

Architektura MySQL, popsána schématem na obrázku 5.6, se velmi odlišuje od architektur jiných databázových serverů, má široký záběr a je užitečná pro řešení mnoha různoro-

dých úloh. Vrstva, která je úplně nahoře, obsahuje služby, jež nejsou jedinečné pro MySQL. Obsluhují většinu potřebných nástrojů klient/server, které jsou založeny na síťovém provozu.Ve druhé vrstvě se nachází valná část funkcionality MySQL, včetně kódu pro rozbor tzv. `parser` , analýzu a optimalizaci pro všechny zabudované funkce. Tato funkcionalita je poskytována prostřednictvím úložných `enginů`. Text je přejatý z [1].



Obrázek 5.6: schéma MySQL databáze, přejato z [1]

Třetí vrstva obsahuje tak zvané úložné typy (`enginy`). Ty mají na starosti ukládání a získávání všech dat uložených v MySQL. Server komunikuje s úložnými `enginy` prostřednictvím API. Toto rozhraní skrývá rozdíly mezi jednotlivými úložnými typy a činí je na vrstvě dotazů velmi transparentními. Rozhraní obsahuje několik desítek nízkoúrovňových funkcí, které provádějí základní operace s databází jako je například zahájení transakce nebo získání řádku, který má daný primární klíč. Úložné typy nedělají rozbor SQL a nekomunikují mezi sebou, ale pouze odpovídají na požadavky serveru.

SQLite

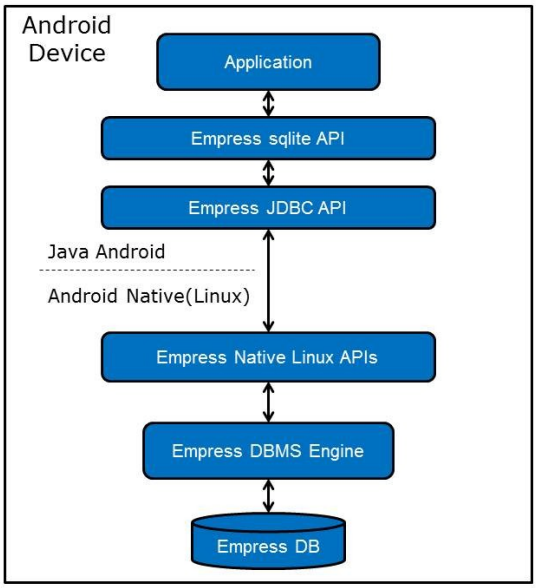
SQLite je relační databázový systém popisovaný knihovnou jazyka C/C++, poskytující rozhraní SQL jazyka bez nutnosti konfigurace. Databáze podléhá téměř plnému znění standardu SQL-92. Komunikace probíhá na základě SQL dotazů, které jsou známé i u ostatních relačních databázových systémů, a jsou definovány ve standardu SQL-92. Databáze je uložena v samostatném souboru a není součástí žádného většího systému, založeném na principu klient-server. Tento princip nepožaduje další software nebo knihovny pro svou funkcionalitu. Systém může být sdílen mezi různými počítači a platformami.

Výhodou tohoto přístupu je ten, že není zapotřebí spouštět samostatný proces, který

běží v pozadí a čeká na uživatelovy dotazy, požadující data v databázi. Požadavky na bezproblémový běh systému jsou minimalizovány, ať už z pohledu paměťové náročnosti, tak z pohledu využití procesoru. Pro implementační databázi není vyžadována konfigurace nebo administrace systému. Použití dané databáze se doporučuje pro menší množství zpracovávaných dat, nebo pro použití na mobilních zařízeních. Většina mobilních platforem používá pro ukládání dat SQLite databázový systém nebo modifikace tohoto systému. Podpora ze strany programovacích jazyků je v rámci API, a to především z jazyků C/C++ a Java, ale existují přístupy i z jazyka Python či Perl. Podpora systému se rozšiřuje s její přibývající popularitou právě na mobilních zařízeních.

Funkčnost vybrané databáze je založena na procesu, který se spouští až v případě potřeby, což znamená, že pokud se s databází nepracuje, neběží v systému žádný zbytečný proces. Pro většinu běžných operací nad databází je rychlejší, než je varianta systému klient-server. Rychlejší přístup k datům je omezen množstvím dat uložených v databázi. Dalším omezením je maximální možná velikost souboru či velikost úložného prostoru. SQLite databáze neobsahuje rozdělení datových typů pro jednotlivé záznamy, nejedná se o chybu nýbrž o záměr. Jedinou výjimku tvoří primární klíč, kde je striktně vyžadováno celé číslo. Databázový systém zkoumá datový typ pouze v případě uplatnění klauzule ORDER BY, určující seřazení jednotlivých požadujících záznamů. Další informace lze nalézt v literatuře, ze které vychází i tento text [21].

Databáze SQLite je aktuálně nejčastěji nasazovaným systémem pro ukládání dat na mobilních zařízeních. Na platformě Android je SQLite databáze použita například pro ukládání kontaktů, krátkých textových zpráv nebo historie volání. Ukládání do databáze je podporovány i při vývoji aplikací přes API v Android SDK. Implementace SQLite databáze na platformě Android je znázorněna obrázkem 5.7. Na obrázku je popsána architektura implementace včetně systému řízení báze dat (DBMS) a následné API rozhraní zapouzdřující komunikaci na jednotlivých vrstvách, až po aplikaci přistupující k databázi.



Obrázek 5.7: architektura SQLite databáze na platformě Android

5.4 Google Mapy

Mapy Google (dříve Google Local) je internetová služba a technologie poskytovaná společností Google, která umožňuje přístup k mapovým podkladům, které jsou prezentovány například webovými stránkami mapy Googlu. Službu mohou užívat aplikace třetích stran přes dostupné API. Mezi základní nabídku služby lze uvést mapy ulic, plánovač cesty pro cestování pěšky, autem, na kole (ve vývoji) nebo s veřejnou dopravou a lokátorem městských podniků v mnoha zemích po celém světě. Mapy nabízí i satelitní snímky, které nejsou aktualizovány v reálném čase, ale spíše několik měsíců či let. Mapy poskytují letecké nebo satelitní snímky většiny městských oblastí po celém světě. Většina světových satelitních snímků není starší než 5 let a aktualizace probíhá nepravidelně, což zahrnuje určité nepřesnosti. Ne všechny plochy z družicových snímků jsou zahrnuty ve stejném rozlišení, méně osídlené oblasti obvykle dostanou menší detaily. Přestože je použito slovo satelit, většina snímků měst s vysokým rozlišením je letecký snímek z letadla, které letí v určité výšce nikoli ze satelitů, zatímco většina ostatních snímků je ve skutečnosti z družic. Na obrázku 5.8 je zobrazena stejná lokalita, kde v levé části je zobrazena klasickým způsobem, uprostřed je zobrazen satelitní snímek a v pravé části je satelitní snímek včetně aktuálního stavu dopravy.

Obrázek 5.8: mapy Google

Stejně jako mnoho dalších webových aplikací společnosti Google, mapy používají skriptovací jazyk JavaScript ve velké míře. Když uživatel stiskne tlačítko myši nebo se dotkne prstem obrazovky a táhne mapu, čtverce mřížky jsou staženy ze serveru a vloženy na stránku. Když uživatel vyhledává, výsledky jsou staženy na pozadí pro vložení do bočního panelu a do mapy, stránka není zatížena a nenastává zaseknutí stránky způsobené právě načítáním dat. Na stránkách je také použit JSON pro přenos dat spíše než XML, z výkonnostních důvodů. Tyto techniky spadají pod široké působení Ajaxu.

Od roku 2005 je dostupné Google Maps API, které umožňuje vývojářům integraci map na svých internetových stránkách. Jedná se o bezplatnou službu, v současné době bez obsahu reklam, ale společnost uvádí ve svých podmínkách použití, které si vyhrazují právo zobrazovat reklamy v budoucnosti. Při použití Google Maps API je možné vložit mapy do externí webové stránky. Ačkoli zpočátku bylo toto API ryze pro jazyk JavaScript, posléze velice rychle rozšířeno do více modifikací. Například pro aplikace Adobe Flash.

Rozhraní API je poskytované zdarma pro komerční použití za předpokladu, že místo, na kterém je používán, je veřejně přístupné a neúčtuje si poplatky za přístup a není generováno více než 25 000 přístupů během jednoho dne. Pro rozšíření lze zaplatit licenci,

která předešlé omezení neobsahuje. Společnost Google, propojila své mapy s platformou Android a umožnila tak přístup na mapové podklady zákazníkům mobilních zařízení. Zároveň uvolnila další verzi svého API pro mobilní zařízení, které umožňuje vývojářům mobilních aplikací integrovat mapy do aplikací třetích stran. Podmínky, které jsou kladeny na provoz zdarma jsou totožné jako u klasické verze API. Google Maps for Mobile verze 2 umožňuje získat přístup k poloze bez přijímače GPS. Tato funkcionalita je řešena tak, že nejprve se poloha hledá u služby založené na GPS, následuje hledání na základě služeb WLAN, WIFI a nakonec u služeb mobilních operátorů respektive jejich vysílačů BTS. Podrobnější informace k získání polohy lze nalézt v kapitole 3.2. Aktuálně je služba dostupná pro následující mobilní platformy Android, iOS, Windows Mobile, Symbian, BlackBerry a Palm webOS.

Mapy používají zobrazování založené na Mercatorově projekci. Kdyby Země byla dokonale kulatá, projekce by přesně odpovídala Mercatorově projekci. Mapy používají specifické vzorce, které slouží pro převod mezi klasickými GPS souřadnicemi, používající WGS-84 model. Rozdíl spočívá v modelu, kde WGS-84 předpokládá model země jako elipsoid, zatímco Mercator pracuje s modelem Země jako s přesnou kulovou plochou. Přesnost je v globálním měřítku téměř nepozorovatelná, ale v lokálním měřítku se lze setkat s menšími nepřesnostmi.

Pro zobrazení jsou použity následující rovnice, které popisují způsob projekce zeměpisné šířky a délky na mapové podklady, které jsou viditelné uživateli. Výsledné hodnoty d_x a d_y určují vykreslující souřadnice ve dvou směrech, které jsou vykreslovány uživateli na zobrazovací plochu. Vykreslovací plochou je nejčastěji monitor. V rovnicích jsou hodnoty dE a dN reprezentanty místních souřadnic reprezentujících nekonečno v Mercatorově projekci. Úhel φ označuje zeměpisnou šířku, e je první excentricita elipsoidu Země, a označuje hlavní poloosu Země. V obou rovnicích se vyskytuje parametr a_{map}, jehož výpočet je uveden jako poslední, určující velikost mapy tak, jak je vykreslena uživateli a je závislý na stupni přiblížení.

$$d_x = a_{map}\left(\frac{a}{\sqrt{1-e^2\sin^2\varphi}}\right)^{-1}\sec\varphi dE \tag{5.1}$$

$$d_y = a_{map}\left(\frac{a(1-e^2)}{(1-e^2\sin^2\varphi)^{3/2}}\right)^{-1}\sec\varphi dN \tag{5.2}$$

$$a_{map} = \frac{256.2^{zoom}}{2\pi} pixels \tag{5.3}$$

Kapitola 6

Mobilní aplikace

Tento projekt je zaměřen na vytvoření aplikace pro systém Android, který byl zmíněn dříve, umožňující využití minimálně jednoho senzoru mobilního zařízení, který bude užitečný v kokpitu automobilu. Na základě těchto požadavků byla navržena aplikace pro měření výmolů a nerovností dopravní sítě. Primárně je aplikace navržena pro detekci větších nerovností, které mohou způsobit poškození vozidla a je vhodné na ně určitým způsobem upozornit. Aplikace měří povrch vozovky a zároveň upozorňuje uživatele na blížící se nekvalitní povrch vozovky.

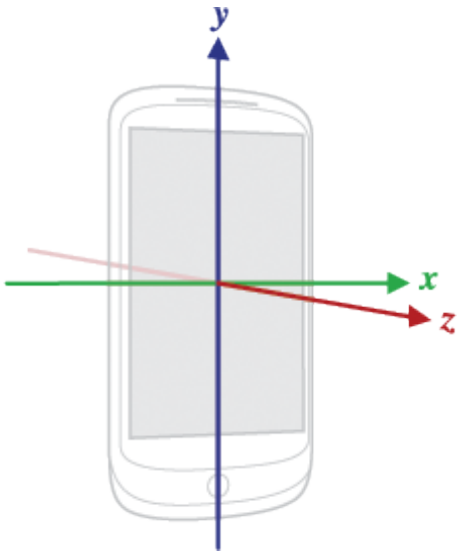
Uživatel je tedy sběratelem dat, která jsou nutná pro posouzení kvality vozovky. K mobilní aplikaci je dostupný server, kde jsou veškerá měření seskupována a vyhodnocována. Naměřená data je vhodné odesílat na tento server a aktualizovat tím databázi nerovností. Je v režii uživatele, zda a také kdy data bude na server odesílat. Uživatel dostává možnost odeslat a zároveň synchronizovat data s ostatními uživateli dle svého uvážení, a to především v závislosti na dostupnosti připojení na internet. Aplikace je vhodná pro majitele a společnosti, které jsou zodpovědné za stav dopravní sítě.

Tato kapitola pojednává o implementaci mobilní části tohoto projektu. Především se jedná o popis získávání dat ze senzorů GPS a akcelerometru, a následného zpracování signálu. Je zde řešena problematika a důvody otočení souřadného systému, který je pevně daný pro mobilní zařízení. Dále je v této kapitole řešen způsob uložení dat, jeho dostupnost a efektivita. Následuje popis grafického uživatelského rozhraní celé aplikace, zahrnující návrh aplikace z pohledu rozložení komponentů, okenních manažerů a služeb. Grafické uživatelské rozhraní obsahuje i popis služby umožňující zobrazení nerovností na mapě. Poslední částí je popis komunikace se serverem, který slouží pro sjednocení dat od všech uživatelů. Komunikace se serverem je důležitá pro synchronizaci dat a zajištění aktuálních dat uložených na serveru a následné stažení aktuálních oblastí do jednotlivých mobilních zařízení.

6.1 Rotace souřadného sytému

Senzory mobilního zařízení, které měří veličiny v prostoru, mají pevně zadaný souřadný systém. Tento systém je neměnný a pevně určen pro každé mobilní zařízení. Typicky je tento systém na platformě Android nastaven tak, že osa x měří kladné hodnoty ve směru doprava, osa y směrem vzhůru a osa z ve směru od displeje k uživateli. Tento souřadný systém je ilustrován na obrázku 6.1, který pochází z oficiální dokumentace platformy Android [3].

Na základě souřadného systému lze získat data ze senzoru v různých směrech. V tomto projektu je představitel tohoto senzoru akcelerometr. Pro měření vibrací, které jsou zrych-



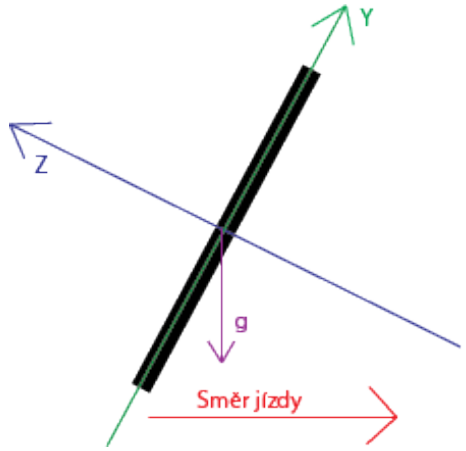
Obrázek 6.1: souřadný systém na platformě Android, přejato z [3]

lení ve vertikálním směru, je vhodné natočit mobilní zařízení tak, aby vertikální zrychlení bylo natočeno v jedné z hlavních os. V projektu je výsledné vertikální zrychlení promítnuto do osy y. Rotace souřadného systému je nutné řešit z několika důvodů. Při ukotvení mobilního zařízení do statického držáku v kokpitu automobilu, je zařízení pootočené vzhledem k osám reálného světa a vertikální zrychlení se promítá do všech os. Druhým a důležitějším problémem je zrychlení nebo zpomalování automobilu, které se do výsledného měření promítne pokud je mobilní zařízení natočeno ve směru jízdy, nebo natočeno ve směru jízdy v ose, ve které probíhá měření vertikálního zrychlení.

Pro odstranění těchto nežádoucích vlivů je řešena rotace souřadných os, lze také konstatovat jako promítnutí výsledných hodnot do jiného souřadného systému. Při odhadování úhlu, o který se bude v daných osách natáčet souřadný systém, je nutné zvolit pevný bod nebo vektor, který určí o jaký úhel je zařízení vychýleno. K tomuto účelu jsem využil gravitační zrychlení, které je stále kolmé k povrchu Země a zároveň je vždy do měření akcelerometru promítnuto. Promítnutí výsledných hodnot probíhá okolo dvou os x a z. Pokud bude zařízení natočeno jinak, než jak je ukázáno na obrázku 6.1, je výpočet řešen prohozením souřadných os a to tak, že se souřadný systém otočí kolem dané osy o 90° a výpočet je proveden stejně tak, jako kdyby byl v základní poloze.

Pro měření je primární otočení kolem osy x. Jelikož rotace kolem této osy má velký dopad na omezení vlivu zrychlení či zpomalení vozidla. Tento stav je znázorněn obrázkem 6.2. Na obrázku jsou znázorněny osy zařízení s tím, že osa x směřuje kolmo směrem k obrazovce. Hlavní černá linka značí mobilní zařízení ukotvené v kokpitu automobilu. Z obrázku je patrné, že zrychlení automobilu se bude velkou měrou promítat do osy z a částečně i do osy y. V případě, že se souřadný systém otočí z aktuální polohy o určitý úhel tak, aby osa y byla ve vertikálním směru, a tím se veškeré zrychlení automobilu promítne do osy z. Cílem je dosáhnout takové polohy zařízení, ve které lze jednoznačně určit vertikální zrychlení v jedné z hlavních os.

Získání úhlu natočení zařízení je provedeno lze s využitím akcelerometru, který do svého měření zahrnuje gravitační zrychlení Země. Při ukotvení zařízení do držáku a spuštění aplikace probíhá nejprve kalibrace zařízení a to tím způsobem, že pro prvních pár naměřených



Obrázek 6.2: vliv zrychlení automobilu na natočení mobilního zařízení

hodnot proběhne měření gravitačního zrychlení, které se promítne do všech os, vypočte se jeho průměrná hodnota a na základě této hodnoty získáme vektor v prostoru. Na základě tohoto vektoru gravitačního zrychlení lze provést výpočet úhlu mezi osou y a vektorem, a stejným způsobem pro úhel mezi osou z a vektorem. Výpočet úhlu mezi dvěma vektory je popsán rovnicí 6.1. Do rovnice lze například za vektor v dosadit souřadnice osy y, které jsou $(0, 1, 0)^T$, a za vektor u hodnoty gravitačního zrychlení promítnutého do všech os. Druhý vzorec je jen pro doplnění a jedná se o výpočet velikosti vektoru v prostoru. Více o vektorových prostorech a výpočtech z teorie vektorů lze nalézt v literatuře [6].

$$\cos(\varphi) = \frac{u_1.v_1 + u_2.v_2 + u_3.v_3}{|a|.|b|} \tag{6.1}$$
$$|x| = \sqrt{x_1^2 + x_2^2 + x_3^2}$$

Nyní když je znám úhel, o který je v daných osách vhodné souřadný systém otočit, je možné provést transformaci souřadného systému. Tato transformace je řešena tzv. transformační maticí, kterou se neměřené hodnoty vynásobí zprava. Přepočtené hodnoty jsou výslednými požadovanými hodnotami naměřené daným senzorem v cílovém souřadném systému. Pro otočení hodnot podle osy x je definována matice rotace 6.2, která je následována maticí rotace pro osu z. Pro rotaci v ose x a zároveň v ose z lze určit obecnou matici rotace a to tak, že se matice skládají (roznásobí) zprava a výsledná matice 6.3 je násobena naměřenými hodnotami. Je vhodné upozornit, že transformace patří do tzv. afinních operací. Mezi tyto operace lze zařadit například posunutí, změnu měřítka či zkosení a operace vzniklé jejich skládáním.Více o afinní geometrii lze nalézt v literatuře [6].

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix} R_z = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6.2}$$

$$R_{xy} = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \cos\varphi.\sin\phi & \cos\varphi.\cos\phi & -\sin\varphi \\ \sin\varphi.\sin\phi & \sin\varphi.\cos\phi & \cos\varphi \end{pmatrix} \tag{6.3}$$

6.2 Měření výmolů a nerovností

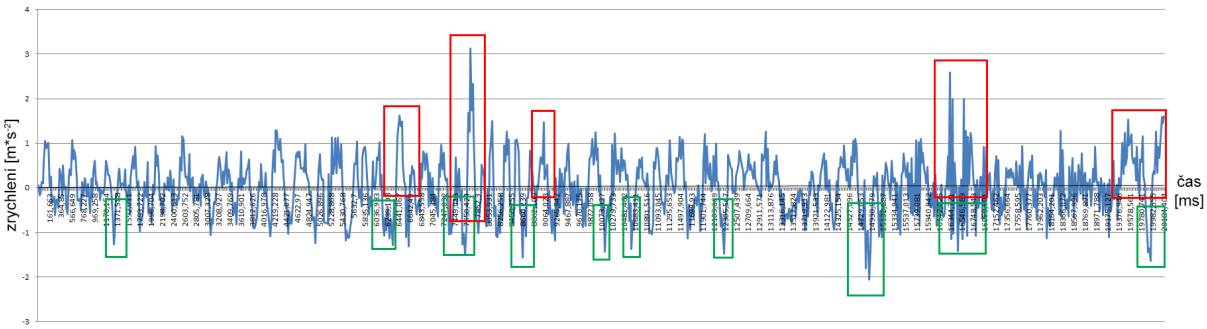
Odstranění gravitačního zrychlení

Pro měření je nutné mobilní zařízení pevně ukotvit v automobilu, kde proběhne kalibrace na základě natočení zařízení tak, jak bylo popsáno v předešlé části. Při samotném měření vertikálního zrychlení (vibrací ve vertikálním směru) se gravitační zrychlení projevuje jako nežádoucí hodnota ve svislém směru, a z tohoto důvodu je vhodné jej odstranit. Na odstranění gravitačního zrychlení z měření byl použit filtr horní propusti, který je doporučován v oficiální dokumentaci Android [3]. Filtr horní propusti, který je implementován v konečném řešení, je definován vzorcem 6.4.

Proměnná $gravity$, kterou je nutné eliminovat, obsahuje gravitační zrychlení a proměnná $result_{value}$ obsahuje požadované zrychlení, též nazývané jako lineární zrychlení. Tyto proměnné $gravity$ a $result_{value}$ jsou zde zjednodušeny na jednu dimenzi. Pro celkový výpočet ve 3D prostoru je nutné vypočítat stejným vzorcem hodnoty ve všech osách. Ve vzorci je α konstantou, která je pro tento typ filtru vypočtena ze vztahu 6.5, kde t je časová konstanta dolní propusti a dT značí průtok událostí. Vzorce jsou převzaty z oficiální dokumentace Android [3].

$$\alpha = 0.8$$
$$gravity = \alpha.gravity + (1 - \alpha).actual_{value}$$
$$result_{value} = actual_{value} - gravity \tag{6.4}$$
$$\alpha = t(t + dT) \tag{6.5}$$

Nyní lze z akcelerometru snímat data, která již obsahují zrychlení ve vertikálním směru a zobrazují vibrace. Ukázka takového měření je znázorněno na obrázku 6.3. Na obrázku jsou vyznačeny oblasti, které by měli být později detekovány jako nerovnosti. Zelená oblast určuje výmoly a prohlubně, a červená oblast označuje nerovnosti či překážky na vozovce.



Obrázek 6.3: měření kvality vozovky akcelerometrem

Zpracování signálu

Akcelerometr slouží pro získání hodnot o kvalitě vozovky a GPS přijímač získává polohu ze satelitních družic. Spojením těchto dvou senzorů lze jednoznačně určit oblast s výskytem výmolů či jiných typů nerovností. Údaje o frekvenci jednotlivých senzorů byly získány na

základě experimentů provedené v rámci této práce. Akcelerometr je schopen dodávat údaje v diskrétní formě s intervalem zhruba 20 ms. Tento interval byl získán na mobilním zařízení, které pouze měřilo kvalitu vozovky a při jiném testování probíhal telefonní rozhovor. Mezi jednotlivými měřeními nebyl rozdíl větší než 1 ms. Nutné bylo vyřešit přiřazení naměřených hodnot, s využitím akcelerometru, k jednotlivým souřadnicím, získaných z GPS senzoru. Interval mezi přijetí následující polohy byl až 1 s. Na základě těchto skutečností bylo nutné měření mezi jednotlivými body přiřadit k jednomu z nich. Bylo rozhodnuto, že měření bude vždy přiřazeno k prvnímu z bodů. Měří se tedy úsek od bodu A k bodu B a veškeré údaje, které budou mezi těmito body získány, jsou přiřazeny k souřadnicím určených bodem A.

V této části je vhodné uvést, jak velké chyby se lze při této úvaze dopustit. Maximální chyba bude v takovém případě, kdy je naměřena hodnota, určující výmol či nerovnost, těsně před získáním další GPS souřadnice. Chyba je závislá na rychlosti, kterou se vozidlo pohybuje. V následující tabulce 6.2 je uveden přehled velikosti chyb v závislosti na rychlosti. Velikost rychlosti vychází z nejčastěji uváděných rychlostí v České republice, kde je například rychlost v obci omezena na $50\frac{km}{h}$. Chybu lze zjednodušit na vzdálenost mezi dvěma body při určité rychlosti. Tyto vzdálenosti jsou uvedeny v tabulce pro dané rychlosti.

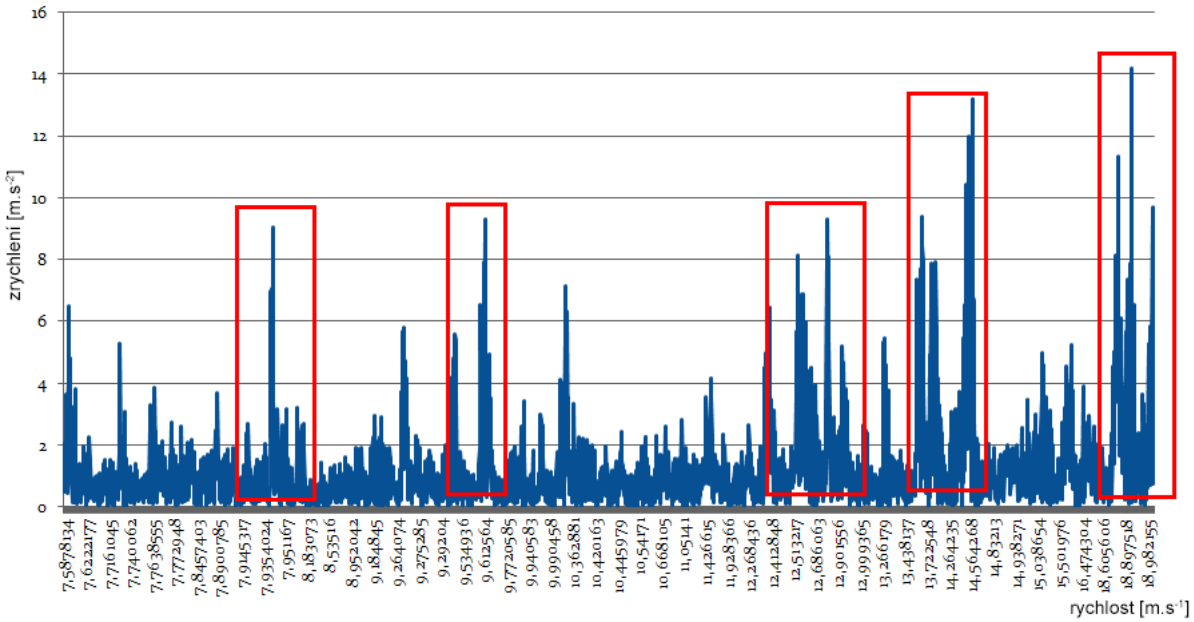
rychlost $[\frac{km}{h}]$	**maximální chyba** $[m]$
50	13,89
70	19,44
90	25
130	36,11

Při detekci výmolu či nějakého typu překážky je nutné identifikovat, zda se jedná o překážku či jen o malou nerovnost. Při experimentování bylo nutné zjistit vliv rychlosti automobilu na získané hodnoty z akcelerometru, při opakovaném projetí určité oblasti s výskytem těchto nerovností. Bylo zjištěno, že akcelerometr změří danou nerovnost v různých rychlostech jinými hodnotami. Touto problematikou se zabývá následující část tohoto textu.

Detekce výmolu nebo překážky na vozovce

Jak bylo vysvětleno dříve snímaní hodnot z akcelerometru probíhá mnohonásobně rychleji než získání další pozice z GPS senzoru. Pro detekování kvality povrchu dopravní sítě je vhodné měřit celý úsek mezi dvěma body a následně tento úsek vyhodnotit. Na základě předchozího rozhodnutí, je celý úsek přiřazen k předchozí GPS pozici a pokud se v daném úseku naměřila nerovnost či výmol, bude pozice zaznamenána. Nejprve je při měření získávána kvalita vozovky a do paměti je zaznamenána pouze maximální hodnota v absolutní hodnotě. Absolutní hodnota při měření byla zvolena na základě požadavku měřit jak výmoly, díry či propadliny,tudíž i nerovnosti, hrboly a překážky omezující provoz. Všeobecně se jedná o nerovnosti, které při přejezdu dané oblasti za nepřiměřené rychlosti, mohou poškodit některé součásti vozidla.

Byl experimentálně zjištěn vliv rychlosti na naměřené hodnoty akcelerometru. A proto jsem následně testoval přejezd několika nerovnostmi v různých rychlostech. Experiment je znázorněn na grafu 6.4, kde označené oblasti identifikují nerovnost, která by měla být detekována. Jedná se o stejnou nerovnost naměřenou v různých rychlostech. Na základě tohoto experimentu jsem se rozhodl definovat práh, který by určil zda se jedná o nerovnost tak závažnou, aby byla uložena či naopak aby byla ignorována a považována za uspokojivý stav vozovky.



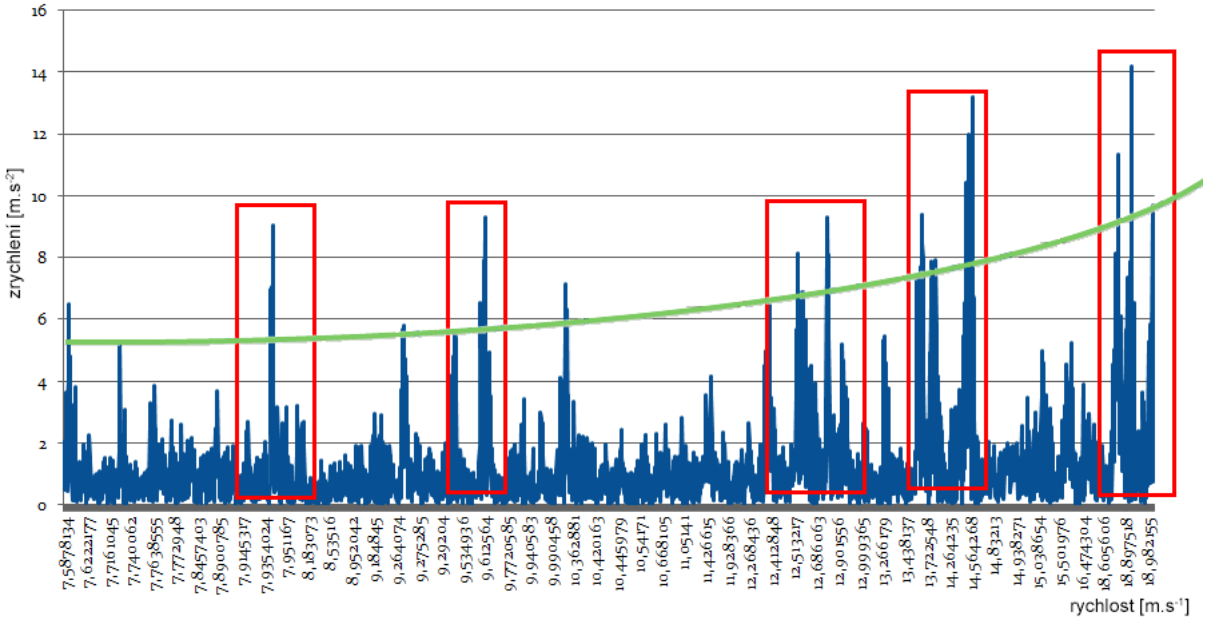
Obrázek 6.4: vliv rychlosti na měření vertikálního zrychlení akcelerometrem

Experiment byl proveden na několika různých místech a v odlišných rychlostech vozidla. Na základě dat, která byla získána měřením, byl stanoven práh jako exponenciální funkci, který zohledňuje aktuální rychlost automobilu. Tato exponenciální funkce má tvar 6.6. Neznámá proměnná v ve vzorci značí rychlost v základních jednotkách ($m.s^{-1}$). Při určování vlivu rychlosti a prahu je vhodné poznamenat, že měření akcelerometru probíhá pouze při pohybujícím se vozidle. Pokud automobil stojí na křižovatce, je měření pozastaveno. Měření probíhá pouze při rychlosti nad 10 $\frac{km}{h}$. Pro rychlost, která je menší, než stanovená minimální rychlost, bylo při experimentování velmi obtížné určit prahovou hodnotu. Měření při nízkých rychlostech bylo zaneseno z velké části chybami, které jsou tímto rozhodnutím odfiltrovány. Nevýhodou, který tento způsob měření přináší je, že pokud je v měřené oblasti například kolona či hustý provoz, tak neprobíhá měření, i když se vozidlo pohybuje. S tímto přístupem je spjata prahová exponenciální funkce, která je posunuta ve směru y o tři hodnoty v kladném směru. Tento posun je určení prahu v minimální rychlosti, tedy pro rychlost 10 $\frac{km}{h}$ je minimální naměřená hodnota zrychlení ve vertikálním směru 3 $\frac{m}{s^2}$ považována za výmol či nerovnost.

$$prah = 1,12^v + 3 \tag{6.6}$$

Při vynesení této prahové exponenciální funkce (vyznačena zelenou čarou) do předešlého grafu 6.5 naměřených hodnot lze jednoznačně určit, zda naměřenou oblast považovat za nerovnost nebo ji ignorovat. Prahová funkce je v obrázku vyznačena zelenou barvou

Rychlost mezi dvěma body (GPS souřadnicemi) je získána tak, že se vytvoří průměrná hodnota rychlostí. Z počáteční rychlosti a koncové rychlosti každého úseku je vytvořen aritmetický průměr. Z tohoto důvodu je vhodné řešit detekování oblasti tím, že se během měřeného úseku zaznamenává pouze maximální hodnota z akcelerometru, která se porovnává s prahovou funkcí.



Obrázek 6.5: prahová exponenciální funkce $1,12^v + 3$

Bylo vysvětleno měření z pohledu akcelerometru, získání důležitých hodnot a definována prahová exponenciální funkce. Pro uložení pozic oblasti obsahující nerovnosti je vhodné definovat směr jízdy. Směr jízdy je zde zaveden pro shlukování jednotlivých naměřených bodů do větších oblastí. Shlukování a zpracování je popsáno v následující kapitole obsahující webovou část. Zohlednění směru jízdy je zde zavedeno pro rozlišení jízdního pruhu či opačného směru vozovky. Služba GPS udává pozici s danou přesností dle dostupnosti (přímého kontaktu) orbitálních družic. Na základě této odchylky od přesné hodnoty může nastat situace, že naměřím oblast s výmoly či nerovnostmi v opačném jízdním pruhu. Při dalším zpracování jsou tyto informace zohledněny. Výpočet směru jízdy je prováděn dle následujících vzorců 6.7. Pro získání výsledku je nutné znát alespoň dvě GPS pozice, respektive souřadnice dvou bodů. Ve vzorci je význam proměnné *lat* zeměpisná šířka s indexem daného bodu a *lon* zeměpisná délka opět se stejným indexem.

$$\begin{aligned} y &= \sin(lon_2 - lon_1).\cos(lat_2) \\ x &= \cos(lat_1).\sin(lat_2) - \sin(lat_1).\cos(lat_2).\cos(lon_2 - lon_1) \\ smer &= \tan(y, x) \end{aligned} \tag{6.7}$$

Ze vzorce je dle vlastností funkce tangens získán výsledný úhel od $-180°$ do $180°$. Tento rozsah je vhodné převést na úhel v rozsahu $0°$ až $360°$. Tento převod je realizován vzorcem 6.8. Kde znak % označuje operaci modulo.

$$uhel = (\alpha + 180)\%360 \tag{6.8}$$

Úhel směru je použit pro následné shlukování do oblastí, kde jedním z kriterií shluku je právě směr. Tento směr je nutné rozšířit o určitou toleranci, důvodem je nepřesnost GPS

souřadnic a dle nich je výpočet směru prováděn. Rozhodl jsem se, dle naměřených dat, nastavit toleranci na ±10°. Velikost tolerance vychází z experimentálního měření, které ukázalo, že pokud automobil jede určitým směrem, odchylka vypočteného směru se pohybuje do dané tolerance.

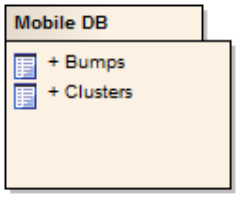
6.3 Uložení dat

Uložení dat v mobilním zařízení je řešeno pomocí dostupných a doporučovaných prostředků. Primárně je pro uložení pozic a shluků použit databázový systém `SQLite`. Přístup k databázi je řešen přes API rozhraní dostupné přes Android SDK. Data, která jsou ukládána do databáze lze rozdělit na dvě části.

První část jsou pracovní data, jedná se o naměřené jednotlivé body znázorňující nerovnost na daném místě včetně dalších informací, které byly popsány dříve. Druhou částí je databáze shluků. Tyto shluky jsou oblasti s nerovnostmi, které jsou potvrzené od více uživatelů. Shluky jsou tvořeny z dat, která jsou od uživatele nahrána na server, kde probíhá jejich vyhodnocení. Popis shlukování a proces zpracování dat je popsán v další kapitole. V této části je rozebrán způsob uložení dat a způsob práce s jednotlivými databázemi.

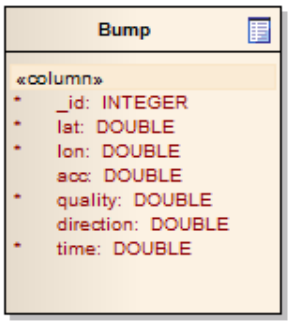
Databáze

Jak již bylo řečeno v úvodu této části, data jsou ukládána do databázového sytému SQLite. Jedná se o databázi tvořenou jedním souborem, který obsahuje jak funkcionalitu, tak jednotlivé záznamy tvořící data. Primárně je nutné ukládat naměřené hodnoty, které lze považovat za výmol či jakýkoli jiný typ nerovnosti. Druhým typem ukládaných dat jsou shluky tvořené z naměřených hodnot zpracované serverem. Pro ukládání těchto dat byla navržena databáze obsahující dvě tabulky. Schéma databáze je zobrazeno UML diagramem 6.6.



Obrázek 6.6: schéma databáze

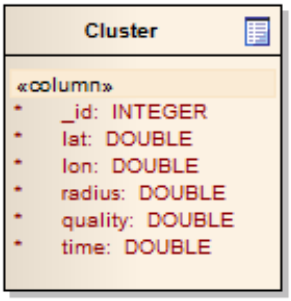
Pro uložení naměřených dat je nutné uchovat polohu, naměřenou velikost vertikálního zrychlení, rychlost automobilu, směr a čas. Návrh všech informací vhodných k uložení do databáze pozic je popsán schématem 6.7, které odpovídá implementačnímu řešení. První hodnota je identifikace daného záznamu unikátním číslem, které je primárním klíčem. Následuje popis pozice zeměpisnou šířkou a délkou. Zadané názvy sloupců `lat` a `lon` označují zkratky z anglických slov `latitude` a `longitude`. Poslední zkratkou je `acc`, která znamená `accuracy` a označuje přesnost dané pozice. Sloupec s názvem `quality` označuje velikost naměřeného vertikálního zrychlení, určující kvalitu dané oblasti, neboli kvalitu nerovnosti. Poslední hodnotou ukládanou do databáze je čas (`time`), kdy byl záznam měření proveden. Ve schématu (UML diagramu) je znak hvězdičky u záznamů, které nemohou být nulové.



Obrázek 6.7: schéma tabulky pro naměřené hodnoty

Databáze s naměřenými nerovnostmi je vždy po odeslání výsledků na server, kde jsou data zpracována, odstraněna. Není nutné na mobilním zařízení uchovávat historii všech naměřených pozic s nerovnostmi. Primárním důvodem pro odstranění dat z mobilní databáze je omezit datový přenos při synchronizaci a snížení datové náročnosti aplikace. Data jsou na serveru zpracována a jsou z nich vytvářeny shluky. Na mobilním zařízení je vhodné uchovávat a zároveň aktualizovat databázi shluků. Shluky obsahují data naměřená od všech uživatelů a mají vyšší odpovídající hodnotu o naměřených oblastech.

Při synchronizaci dat mezi mobilním zařízením a serverem dochází k nahrání naměřených dat od uživatele a zároveň k aktualizaci databáze shluků. Pro tabulku shluků jsou definovány následující sloupce. Tato databáze je podobná databázi naměřených hodnot. Obsahuje pozici shluku, velikost oblasti, hodnotu určující kvalitu a čas. Přesný popis sloupců, které jsou implementované v aplikaci jsou popsány schématem 6.8. Znak hvězdičky u jednotlivých záznamů znamená, že záznam nemůže obsahovat nulovou hodnotu. První hodnotou v tabulce je `_id` označující primární klíč jednoznačně identifikující shluk, následuje, stejně jako v databází naměřených nerovností, zeměpisná šířka `lat` a zeměpisná délka `lon` vycházející z anglických slov `latitude` a `longitude`. Sloupce s názvem `radius` označuje velikost oblasti, kterou daný shluk pokrývá. Jedná se o poloměr kružnice se středem v odpovídající zeměpisné poloze. Hodnota určující kvalitu daného shluku je uložena ve sloupci `quality` a časový údaj `time` určující čas poslední aktualizace shluku zakončují výpis všech potřebných hodnot k definici a vykreslení dané oblasti na mapový podklad.



Obrázek 6.8: schéma tabulky pro zpracované shluky

6.4 Synchronizace dat

Všechna mobilní zařízení, na kterých je při jízdě spuštěna aplikace, ukládají své výsledky do `SQLite` databáze. Všechna naměřená data jsou pouze lokální a jsou dostupná jen uživateli daného zařízení. Primárně se jedná o naměřené nerovnosti, které jsou pouze uloženy a nejsou uživateli zobrazeny na mapu. Pro získání nějaké užitné hodnoty z naměřených dat, je nutné využít synchronizaci. Ta provede několik úkonů, které jsou popsány v této části. Hlavním důvodem pro synchronizaci je aktuální dostupnost nerovností. Databáze nerovností je umístěna na serveru a je vhodné tyto informace stáhnout do mobilního zařízení a zobrazit je uživateli.

Uživatel sám rozhoduje kdy bude synchronizace provedena. Není kladeno omezení na neustálé připojení k internetu. Pro používání aplikace je vhodné vlastnit mobilní připojení, ale pro měření není vyžadováno. Pokud není potřeba zobrazovat nerovnosti na mapě není při měření nutné být on-line.

Druhou nedílnou částí synchronizace je naopak přispění do této databáze svými naměřenými hodnotami. Každý uživatel přidává své naměřené hodnoty, které jsou poté na serveru zpracovány. Výsledkem zpracování je aktuální databáze nerovností, kterou aplikace při synchronizaci stahuje a zobrazí uživateli výsledek na mapu. Při implementaci bylo zváženo využití samostatného vlákna na jednotlivé operace synchronizace. Při zavření aplikace obsahující grafického uživatelského rozhraní se vlákno ukončí a mohlo by dojít k přerušení jednotlivých operací. Takové přerušení by mohlo vést k nežádoucímu efektu či nekonzistentnímu stavu aplikace či databáze.

Pro synchronizaci jsem na platformě Android byla zvolena možnost vykonat operaci jako službu. Požadavky pro vykonání operací odeslání dat a příjem dat nových byla zvolena služba typu `IntentService`. Tento typ je služby připraven pro práci v samostatném vlákně a po dokončení všech svých úloh je automaticky ukončen. Služba v implementačním řešení obsahuje tři části, které jsou popsány v této části kapitoly.

Příprava dat z databáze

V první fázi synchronizace je nutné připravit naměřená data, která jsou uložena v databázi tak, jak bylo řečeno v předchozí části této kapitoly. Do databáze jsou s každým měřením přidávány výmoly či různé typy překážek vyskytující se na vozovce. Při velmi dlouhém provozu aplikace lze dosáhnout určité paměťové náročnosti, kterou není vhodné ignorovat. Z tohoto důvodu bylo rozhodnuto databázi naměřených nerovností, vždy po odeslání a uložení na server, vyprázdnit a připravit na další měření. Tím byla vyřešena náročnost na paměťový prostor.

Data jsou uložena v jediném souboru, který by bylo možné odeslat na server. Při tomto způsobu odeslání dat, jsou přenášena data a zároveň informace o databázi. Pro mobilní aplikace je vhodné navrhnout, co možná nejmenší datový přenos z a do mobilního zařízení. Dle uvážení byl vybrán jiný způsob, který nebude zatěžovat přenos dat dodatečnými informacemi. Zároveň je tento navržený způsob jednoduše převoditelný do celkové databáze. Z databáze byl vytvořen textový soubor, který obsahuje záznam s daty oddělená bílým znakem. Každý řádek reprezentuje jeden záznam v databázi. Tento soubor je následně připraven na přenos na server. Soubor není zabezpečen či nějakým způsobem šifrován. Předpokladem je, že není potřeba šifrovat data, která jsou měřena uživateli dobrovolně a bez možnosti získat finanční odměnu.

Odeslání dat na server

Pro vzájemnou komunikaci byl vybrán HTTP protokol, jehož specifikaci lze nalézt v RFC dokumentu [9]. Na základě tohoto protokolu probíhá komunikace se serverem jak při odesílání dat, tak při přijímání. Důvod pro využití tohoto protokolu je fakt, že se jedná o standard pro komunikaci klienta a serveru. Server je v tomto případě webové rozhraní a klient je mobilní zařízení požadující odeslat data.

Odesílání dat probíhá s využitím veřejně dostupné knihovny `HTTPClient` od společnosti Apache Software Foundations [4]. Tato knihovna je použita pro přenos vytvořeného textového souboru s naměřenými daty na server. Přenos je realizován protokolem HTTP v režimu `POST`. Cílem je pevná URL adresa obsahující `PHP` skript, který zpracuje nahraný soubor. Po úspěšném odeslání dat na server jsou odstraněny z databáze všechny záznamy obsahující naměřené hodnoty z akcelerometru. Tím je databáze připravena pro další měření a nenastává problém s paměťovou náročností aplikace.

Stažení dat ze serveru

Po nahrání dat na server, která jsou zpracována je vhodné stáhnout aktuální data obsahující pozice nerovností. Tyto pozice jsou poskytovány serverem přes PHP skript jako dokument, který zpřístupní data z databáze. Zpřístupnění je provedeno tak, že je vytvořen XML soubor obsahující přesně definovanou strukturu dat. Tato struktura zahrnuje popis všech částí nutných pro identifikaci jednotlivých shluků. Ukázka struktury s daty je popsána 6.9. Kořenový adresář se jmenuje `database` a obsahuje veškeré shluky, které jsou reprezentovány oblastí `cluster`. Shluk nebo v anglickém znění cluster obsahuje informace, které jsou potřebné pro vykreslení na mapový podklad. Jedná se zejména o zeměpisnou šířku a délku, velikost oblasti (poloměrem), kvalitu a čas.

```
<?xml version="1.0" encoding="utf-8"?>
<database>
        <cluster>
                <latitude>49.2278407815</latitude>
                <longitude>16.5924629537</longitude>
                <quality>7.2</quality>
                <radius>6.78573506469</radius>
                <time>1368052643</time>
        </cluster>

        <cluster>
                <latitude>49.2277122703</latitude>
                <longitude>16.5925559984</longitude>
                <quality>8.75</quality>
                <radius>8.68486751487</radius>
                <time>1368033829</time>
        </cluster>
</database>
```

Obrázek 6.9: struktura XML souboru s daty

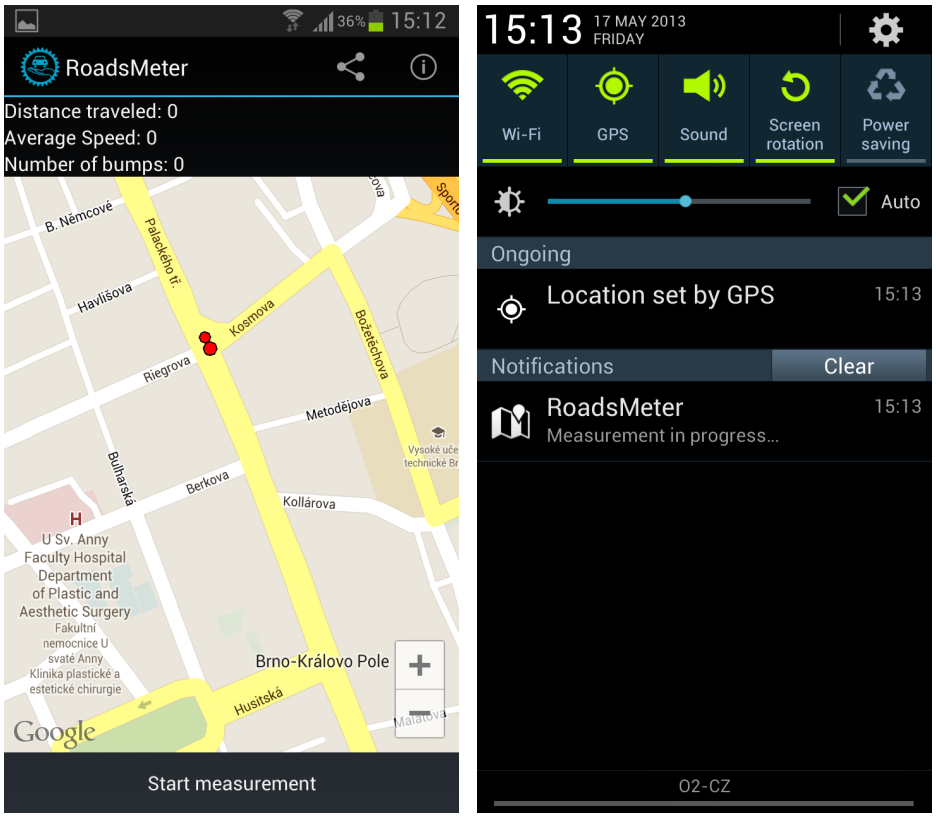
Stahování dokumentu obsahující všechny aktuální pozice nerovností (shluky) probíhá do

vnitřní paměti. Po úspěšném stažení je nutné nejprve vyčistit databázi obsahující předešlé záznamy o shlucích. Následuje čtení dat z vnitřní paměti a ukládání do prázdné databáze. Po skončení synchronizace proběhne asynchronně vykreslení jednotlivých pozic na mapu.

6.5 Grafické uživatelské rozhraní

Aplikace na mobilním zařízení je rozdělena na dvě samostatné části. Důležité jsou obě stejně, jelikož bez práce na pozadí by nebylo možné měření či synchronizace dat, ale uživatel přichází do přímého styku pouze s částí obsahující grafické uživatelské rozhraní. Ta část s uživatelským rozhraním umožňuje uživateli ovládat chování aplikace a to jak na popředí, tak zapínat či vypínat služby běžící na pozadí. Na základě doporučení od vývojářů platformy Android, které lze nalézt na oficiálních stránkách projektu [3], bylo navrženo univerzální grafické uživatelské rozhraní. Tento návrh zahrnuje splnění všeobecných podmínek od různých verzí operačního systému až po přizpůsobení se rozlišení obrazovky.

Při implementaci byly využity dostupné prostředky pro tvorbu vzhledu aplikace. Výsledkem je vzhled, který je převzat z nastavení individuálního zařízení a aplikace se jeví, jako kdyby byla tvořena pro dané zařízení. Výsledná aplikace na konkrétním zařízení vypadá tak, jak je zobrazeno na obrázku 6.10. Z levé části návrhu je patrné, že největší část obrazovky zabere zobrazení mapového podkladu. Na této ploše je zobrazena aktuální poloha řidiče a oblasti s výskytem výmolů nebo nějakého typu nerovností. Tyto oblasti jsou vyznačeny červeně, na levém obrázku jsou oblasti zobrazeny.



Obrázek 6.10: návrh grafického uživatelského rozhraní

Další částí jsou dostupné informace o aktuální jízdě. Jedná se o doplňkové informace zobrazující ujetou vzdálenost od počátku měření, průměrnou rychlost a počet naměřených nerovností. Ve spodní části návrhu lze nalézt tlačítko pro ovládání služby běžící na pozadí a provádějící měření povrchu vozovky. Na obrázku tlačítko obsahuje popis `Start measurement` a po stisknutí se popis změní na `Stop measurement`. Poslední oblastí, která umožňuje s aplikací manipulovat jsou ikony v hodní oblasti (napravo od názvu aplikace). První ikona umožňuje odeslat naměřené hodnoty na server za účelem zpracování. Server data přijme, zpracuje a zašle XML dokument obsahující aktuální oblasti s výskytem nerovností. Druhá ikona nabývá pouze informačního charakteru a po jejím stisknutí jsou zobrazeny informace o aplikaci a jejím použití.

Pravá část obrázku 6.10 ukazuje v nabídce upozornění (součást platformy Android) běžící službu. Grafickou části aplikace je možné zavřít a měření povrchu vozovky bude i nadále probíhat. Při kliknutí na upozornění této aplikace je grafická část aplikace opět spuštěna a probíhá vykreslení do mapy a zobrazení informací. Tento přístup byl zvolen pro možnost zapnout při jízdě jinou aplikaci nebo uskutečnění hovoru.

Kapitola 7

Internetová aplikace

Každý uživatel mobilní aplikace měří své individuální trasy a data jsou přístupná pouze jemu samotnému. Tato data pro jednoho uživatele nemají užitnou hodnotu, jelikož nejsou vykreslována na mapu, ale pouze zaznamenávána na dané zařízení. Data je vhodné shromažďovat a shlukovat na daném úložišti, které bude dostupné všem bez ohledu na čas a místo. Pro tyto účely byla navržena vytvořena internetová aplikace.

Veškeré naměřené hodnoty od uživatelů jsou zpracovávány a poskytovány zpět zdarma. Internetová aplikace je rozdělena na několik částí nebo operací, kde mezi hlavní lze zařadit ukládání dat jejich zpracovaní, zajištění aktuálnosti a poskytování výsledků jak mobilním uživatelům, tak návštěvníkům přes webový prohlížeč. Internetová aplikace je navržena tak, že se jedná o centrální synchronizační databázový server a zároveň je to místo, kde lze pozorovat zpracované údaje na mapě. Tyto údaje značí nerovnosti, které jsou potvrzené od více uživatelů. Potvrzením je myšleno, že daná nerovnost byla naměřena několikrát, na tomto principu funguje metoda shlukování, která je popsána dále v této kapitole.

V této části lze nalézt popis uložení dat, skripty v jazyce `PHP` popisující zpracování požadavků na synchronizaci od mobilních zařízení a popis implementace map.

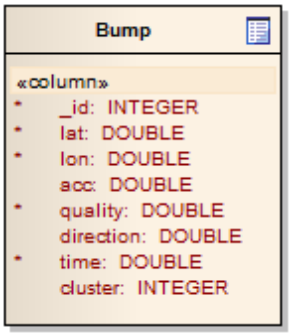
7.1 Uložení dat

V úvodu této kapitoly bylo řečeno, že server slouží pro ukládání dat, která uživatelé nahrávají při synchronizaci. Data od uživatelů jsou ukládána do databáze `MySQL`, která má své výhody a nevýhody. Tento typ databáze byl vybrán z důvodu jednoduchého přístupu z jazyka `PHP`. Databáze je rozdělena na dvě základní části nerovnosti `bumps` a shluky `clusters`. Tyto dvě části zároveň reprezentují název jednotlivých tabulek v databázi.

První částí je tabulka obsahující nerovnosti nebo výmoly. Stejně jako je tomu u databáze na mobilním zařízení i zde je nutné ukládat vše potřebné k identifikaci dané nerovnosti. Schéma této tabulky je popsáno na obrázku 7.1.

První sloupec je jednoznačný identifikátor záznamu nerovnosti `_id`. Následuje informace o pozici ve tvaru zeměpisná šířka `lat` a délka `lon` obě zkratky vycházejí z anglických pojmů `latitude` a `longitude`. K pozici patří údaj `acc`, pochází z anglického slova `accuracy`, reprezentující horizontální přesnost. Daná nerovnost obsahuje určitou kvalitu označující hodnotu naměřenou akcelerometrem, tato kvalita je uložena ve sloupci s názvem `quality`. Při určité toleranci, lze říci i přesnosti, je vhodné uvažovat směr jízdy automobilu. Tento směr je uložen v databázi pod názvem `direction`. Pro potřeby udržení dat v aktuální podobě je nutné uvádět časový údaj `time` a pro metodu shlukování je poslední sloupec

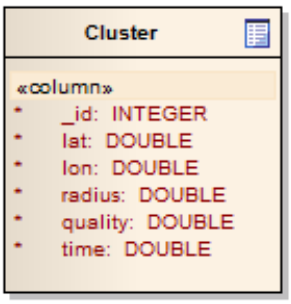
s názvem `cluster`. Ve schématu se vyskytuje symbol hvězdičky, který omezuje daný typ záznamu tak, že není možné ukládat nulovou hodnotu.



Obrázek 7.1: schéma tabulky pro data od uživatelů

Druhou částí je databáze shluků. Pod pojmem shluk si lze představit kruhovou oblast s daným poloměrem a středem, do které spadá určité množství naměřených hodnot. Tyto naměřené hodnoty jsou uloženy v databázi, která byla představena výše. Shluk je oblast, která je opakovaně naměřena uživateli jako nerovnost a do které patří minimální množství dat. Popis shlukování a přesné znění podmínek, za kterých shluk vzniká je popsán v jedné z následujících částí této kapitoly. Shluky jsou výsledkem naměřených dat a jedná se o vykreslovanou část na mapu. Pro shluk je nutné uložit pozici ve stejném tvaru jako naměřená data a poloměr. Uložení shluků je v tabulce `clusters`, jejíž popis je na schématu 7.2.

První nutnou hodnotou, jak je zvykem, je identifikátor jednotlivých shluků. Následuje pozice středu shluku ve tvaru zeměpisná šířka `lat` a délka `lon`. Dalším údajem je `radius` označující poloměr kruhové oblasti a čas `time`. Časový údaj je zde pro zajištění aktuálnosti dat. Stejně jako v předešlém schématu má symbol hvězdičky u jednotlivých záznamů význam nemožnost uložení nulové hodnoty.



Obrázek 7.2: schéma tabulky pro zpracování shluků

7.2 Shlukování

Nabízí se možnost proč je nutné data shlukovat. Především se jedná o zpřesnění a ověření naměřených dat. Naměřené hodnoty jsou přiřazovány k nejbližší předchozí GPS pozici a

údaj o naměřené nerovnosti není přesný. Další nepřesnost je zahrnuta v horizontální přesnosti systému GPS, kde velkou roli hraji počet dostupných orbitálních družic. Pokud by byly brány v úvahu všechny naměřené události od všech uživatelů, vykreslovalo by se na mapu obrovské množství bodů. Body by se nacházely po celé mapě, ale přesto by docházelo k určitému shlukování k místům s výskytem určité nerovnosti. Shlukování je v tomto případě takový filtr, který odstraní body, které se vyskytují samostatně a naopak zvýrazní oblasti, kde se jednotlivé body nachází v dostatečně velkém počtu. Primárně je shlukování v tomto projektu pro filtraci a pro malou paměťovou a procesorovou náročnost na mobilních zařízení.

Nyní je definováno jak jsou data na serveru uložena a co je nutné pro uložení jednotlivých záznamů. Při ukládání uživatelských dat je vyvolán skript, který je pojmenován `upload.php`. Jeho význam je takový, že mobilní zařízení tomuto skriptu pošlu textový soubor obsahující naměřené hodnoty akcelerometrem s přiřazenými pozicemi. Skript, jak název napovídá, je napsán v jazyce `PHP`. Tento skript na základě regulárních výrazů provede získání jednotlivých záznamů o naměřených hodnotách a uloží je do databáze naměřených hodnot. Všem novým záznamům je přiřazena hodnota shluku rovna nule. Pokud má záznam hodnotu ve sloupci `cluster` rovnou nule, znamená to, že není přiřazen k žádnému shluku. Po uložení všech pozic do databáze je spuštěna metoda shlukování a probíhá zpracování těchto nových dat na základě metody, která je popsána v této části.

Metriky

Pro shlukování dat byla vybrána metoda, která požaduje jako vstupní parametr vzdálenost. Výběr implementační metody je popsán další části této kapitoly. Co to vzdálenost je a jak je definována v matematice je obsahem této části.

Metrika na množině X je funkce, která je nazývána vzdáleností $d : X.X \rightarrow R$, kde R je množina všech reálných čísel. Pro všechna x, y a z patřící do množiny X je požadováno splnění následujících podmínek 7.2.

1. $d(x, y) \geq 0$
2. $d(x, y) = 0$, pouze pokud $x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \leq d(x, y) + d(y, z)$

V oblasti shlukování se nejvíce osvědčily Euklidovská a Manhattanská metrika. Obě mají specifický výpočet vzdáleností, který splňuje výše uvedené podmínky. Při implementaci výpočtu vzdálenosti bylo vycházeno ze vzorce Euklidovské metriky. Euklidovská metrika je někdy nazývána obecná nebo běžná metrika mezi dvěma body.

Vzdálenost

Výpočet vzdálenosti u Euklidovské metriky je řešen s využitím Pythagorovy formule. Příklad výpočtu vzdálenosti v kartézském souřadném systému v Euklidově n-dimenzionálním prostoru je znázorněn vzorcem 7.1. Ve vzorci jsou hodnoty $p = (p_1, p_2, \ldots, p_n), q = (q_1, q_2, \ldots, q_n)$ n-dimenzionální vektory. Výsledná hodnota funkce $d(p, q$ je vzdáleností mezi dvěma vektory p a q.

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \quad (7.1)$$

Výpočet vzdálenosti dvou souřadnic na Zemi je charakterizován vzorcem 7.2. Vzorec vychází z předešlých podmínek a výpočtu vzdálenosti Euklidovské metriky. Ve vzorci je φ zeměpisná šířka, λ zeměpisná délka a R poloměr Země, odpovídající 6,371 km. Zeměpisná délka a šířka je uvedena v radiánech. Výsledná vzdálenost je uložena v proměnné d.

$$\begin{aligned} a &= \sin^2(\frac{\Delta\varphi}{2}) + \cos(\varphi_1).\cos(\varphi_2).\sin^2(\frac{\Delta\lambda}{2}) \\ c &= 2.\arctan(\sqrt{a}, \sqrt{1-a}) \\ d &= R.c \end{aligned} \quad (7.2)$$

Pro implementaci byl zvolen výpočet vzdálenosti, který vychází ze dvou předchozích vzorců. Jedná se o upravení vzorce 7.2. Výsledná implementace výpočtu vzdálenosti je řešena pomocí vzorce 7.3, který je efektivnější při zohlednění náročnosti na rychlost výpočtu. V tomto vzorci vystupují stejné proměnné jako ve vzorci, ze kterého výpočet vychází. Proměnná φ označuje zeměpisnou šířku, λ zeměpisnou délku a R je poloměr Země. Výsledná vzdálenost dvou souřadnic je uložena v proměnné d.

$$d = \arccos(\sin(\varphi_1).\sin(\varphi_2) + \cos(\varphi_1).\cos(\varphi_2).\cos(\Delta\lambda)).R \quad (7.3)$$

Metoda shlukování

Shlukování naměřených dat je nutné řešit takovou metodou, která při spuštění nevyžaduje znát počet shluků. Z tohoto výběru lze eliminovat metodu k-means, která je jednoduchá a velmi používaná. Pro tento typ úlohy byl zvolen algoritmus DBSCAN (Density Based Spatial Clustering of Applications with Noise), který je navržen na shlukování prostorových dat dle hustoty výskytu. Nevýhoda této metody spočívá v citlivosti vstupních parametrů. Pro spuštění metody DBSCAN je nutné znát dva parametry. Prvním je minimální počet bodů a druhým je vzdálenost. Pokud do dané vzdálenosti je nalezeno minimální množství bodů je vytvořen shluk a jsou do něj přiřazeny všechny body.

Pseudokód algoritmu DBSCAN je popsán v 7.2. Jedná se o základní princip fungování algoritmu. Existuje mnoho modifikací, které upravují jednotlivé parametry a zároveň jsou různá doporučení jak zvolit vstupní parametry `Eps` označující vzdálenost a `MinPts` určující minimální počet bodů. Posledním parametrem je databáze obsahující jednotlivé prostorové body `D`.

Implementační řešení vychází z velké části z algoritmu DBSCAN, ale bylo potřeba navrhnout několik modifikací. Do shlukování je nutné zahrnout dvě podmínky, které jsou součástí požadavků na výslednou aplikaci. Prvním požadavkem je dodržení aktuálnosti dat. Z tohoto důvodu jsou do databáze nerovností (bumps) a oblastí (clusterů) přidány hodnoty reprezentující čas. Druhým požadavkem je vytvoření sluku dat na základě podmínek kladených algoritmem DBSCAN. Tyto podmínky zahrnují minimální počet bodů a vzdálenost, do které se okolní body hledají. Byla zvolena tolerance 15 m, která je v městské zástavbě maximální přípustná přesnost GPS pozice, je nutné určit ještě jeden klíčový parametr. Tímto parametrem je směr jízdy. Pokud by nebyl uveden směr jízdy, mohla by nastat

Algorithm 1 pseudokód algoritmu DBSCAN

```
function DBSCAN(D, Eps, MinPts)
    C = 0
    for all  unvisited point P in dataset D do
        mark P as visited
        NeighborPts = regionQuery(P, Eps)
        if sizeof(NeighborPts) < MinPts then
            mark P as NOISE
        else
            C = new cluster
            expandCluster(P, NeighborPts, C, Eps, MinPts)
        end if
    end for
end function

function expandCluster(P, NeighborPts, Eps, MinPts)
    add P to cluster C
    for all point N in NeighborPts do
        if N is not visited then
            mark N as visited
            NeighborPts' = regionQuery(N, Eps)
            if sizeof(NeighborPts' then >= MinPts
                Neighbors = Neighbors joined with NeighborPts'
            end if
        end if
        if N is not yet member of any cluster then
            add N to cluster C
        end if
    end for
end function

function regionQuery(P, Eps)
    return all points within P's eps-neighborhood (including P)
end function
```

situace, že do shluku jsou přiřazeny nerovnosti z jiného jízdního pruhu či opačného směru. Uživatel mobilní aplikace by mohl být zmaten při zobrazení na mapě oblasti s výskytem nerovnosti, která je v opačném směru jízdy, v jeho jízdním pruhu.

Z předchozích podmínek vyplývá, že algoritmus DBSCAN je modifikován v několika bodech. Především se jedná o výběr okolních bodů, kde je zahrnut směr jízdy. Algoritmus výběru těchto okolních bodů je popsán pseudo-algoritmem 7.2. Ve vzorci je porovnání vzdálenosti, která je vypočtena vzorcem, který byl uveden v předešlé části této kapitoly. Podmínka stejného směru je tolerována do odchylky $\pm 10°$. Druhým požadovaným parametrem je časový údaj. Ten je zanesen do implementace při výběru jednotlivých bodů z databáze. Jsou vždy vybírány takové body, které nejsou starší než jeden měsíc.

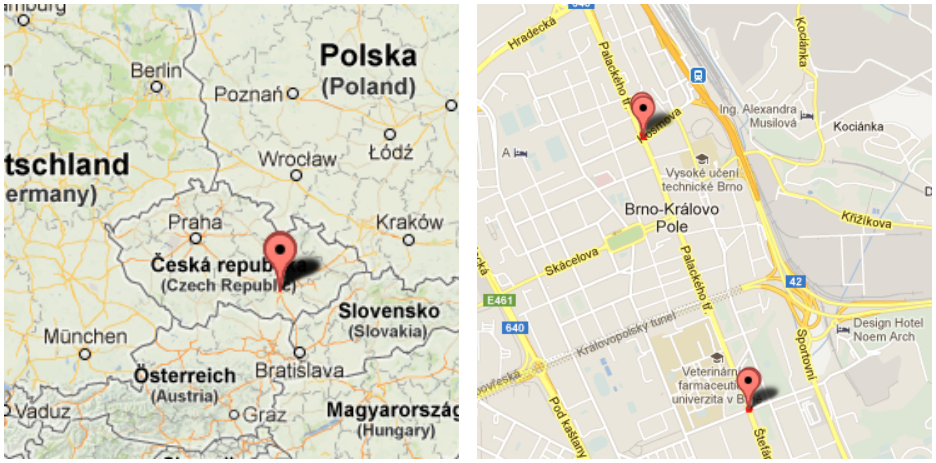
Algorithm 2 výběr okolních bodů z databáze

```
function getNeighbors(D, P, Eps, Direction)
    neighbors = []
    for all points in dataset D do
        if (point has the same direction as P) and (distance(point, P) < Eps) and point
is not P then
            add point into neighbors
        end if
    end for
    return neighbors
end function
```

7.3 Grafické uživatelské rozhraní

Internetová aplikace zpracovávající data z mobilních zařízení zobrazuje výsledky na mapový podklad. Pro zobrazení byla veškerá plocha využita pro vykreslení mapy s oblastmi výskytu výmolů nebo jiného typu nerovnosti omezující plynulou jízdu. Tyto oblasti jsou stejně jako v mobilní části vykresleny červenou barvou a zároveň opatřeny ukazatelem středu. Smysl tohoto ukazatele středu je v jeho vnitřní implementaci. Ta umožňuje zobrazit tento popisek i ve velké výšce mapy, kde by vykreslená oblast nebyla okem patrná. Příkladem může být vykreslená oblast na nějaké křižovatce při pohledu na mapu v rámci světa, kontinentu nebo i státu nebude tato oblast pozorovatelná. Při použití popisu středu bude alespoň v rámci města viditelný jeden ukazatel.

Tento jev je ilustrován obrázkem 7.3. V levé části je náhled naměřených a zpracovaných dat v rámci kontinentu. Přiblížení stejných dat lze konstatovat, že se jedná o tři samostatné oblasti a je pozorovatelné i vykreslení kruhových oblastí. Na rozdíl od mobilní aplikace, zde jsou vždy vykreslovány aktuální oblasti. Aktuálnost je dodržována dle časové hodnoty uložené u každé naměřené hodnoty a při vytváření nebo aktualizaci jednotlivých shluků.

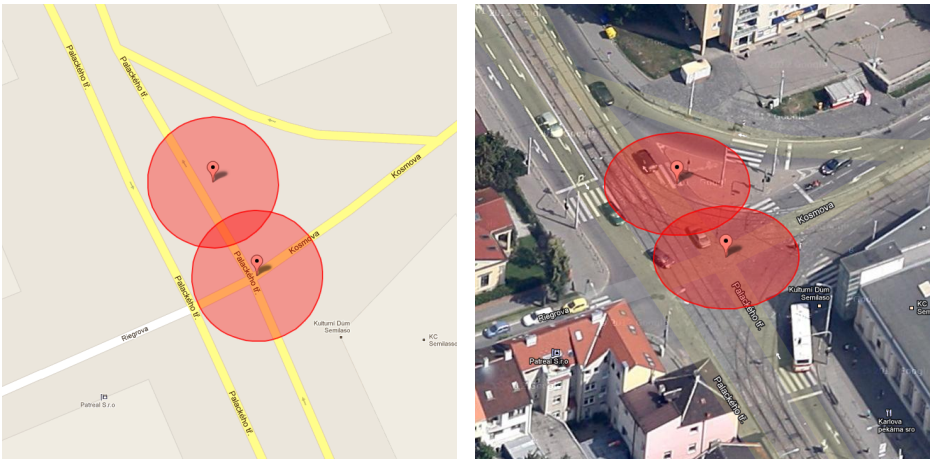


Obrázek 7.3: zobrazení popisu oblastí

Mobilní zařízení může, ale zároveň nemusí obsahovat aktuální databázi oblastí. Aktuální informace jsou vždy zdarma dostupné na serveru, ale na individuálním mobilním zařízení je aktualizace přenechána na potřebě uživatele. Uživatel se rozhoduje zda a hlavně kdy bude synchronizovat svá data.

Severová část aplikace slouží jako náhled na veškerá naměřená a zpracovaná data od všech uživatelů. Zde lze nalézt výsledky jednotlivých měření a případně vybrat trasu cesty dle kvality vozovky. Na výsledné hodnoty (naměřená a zpracovaná data do shluků) lze nahlížet i jinak, než z pohledu řidiče. Společnosti zabývající se správou silniční sítě mohou stejná data využít pro opravu jednotlivých nebezpečných úseků nebo provést opatření k jejímu omezení. Na základě zobrazených dat lze odhadnout i rychlost průjezdu určitou oblastí a případné zdržení.

Při prohlížení oblastí je možnost zobrazit si fotografický typ mapy, který je poskytován od služby map. Ukázka tohoto typu mapy je zobrazena na obrázku 7.4. Na obrázku je v levé části zobrazena naměřená oblast s nerovnostmi klasickým způsobem. V pravé části téhož obrázku je zobrazena stejná oblast s fotografickým podkladem. Fotografický podklad lze



Obrázek 7.4: zobrazení oblastí s různým mapovým podkladem

využít pro detailnější náhled na konkrétní oblasti. Další výhodou při náhledu u fotografické části mapy je ten, jehož lze s menší námahou identifikovat oblast výskytu nerovnosti.

Kapitola 8

Dosažené výsledky

Byla vytvořena aplikace pro platformu Android umožňující využití senzorů v automobilu a serverová část zpracovávající naměřené hodnoty z mobilních zařízení. Tato aplikace zaznamenává vertikální vibrace akcelerometrem a zpracovává je. Při zpracování probíhá rozhodnutí, zda se jedná o výmol nebo nějaký typ nerovnosti, který by mohl způsobit poškození vozidla. Pokud aplikace zhodnotí naměřená data jako možné riziko, uloží je do své interní paměti. Uživatel aplikace dostává možnost svá naměřená data sdílet. Při sdílení dat je provedeno jejich zpracování, na základě několika kriterií algoritmem DBSCAN, a uložení do databáze na serveru. Při sdílení dat z mobilního zařízení jsou zpracovaná data ze serveru obsahující shluky stažena do mobilního zařízení. Tyto data obsahují oblasti s možným rizikem poškození vozidla, která jsou naměřena více uživateli (nebo stejným uživatelem několikrát) a jsou ve stejném směru jízdy. Zároveň jsou tyto data aktuální. Aktuálnost dat je hlídána na serveru dle časové hodnoty u jednotlivých záznamů oblastí.

Aplikace byla primárně testována na mobilním zařízení Samsung Galaxy S3, ze kterého jsou i naměřené výsledné hodnoty s vytvořenými shluky v daných oblastech. U testovaného automobilu proběhla změna podvozku z pohledu výměny zimních a letních pneumatik. Testování probíhalo nejprve v zimním období, kde bylo provedeno několik měření. Převážná část měření byla na letních pneumatikách a nebyly nalezeny výrazné rozdíly v měření. Na základě daných měření byla nalezena prahová funkce, která definuje, zda jsou naměřené hodnoty nebezpečné nebo se jedná o klasický stav vozovky. Klasický stav vozovky je porovnáván dle stavu silnic v České republice, jsou zde zahrnuty oblasti jak s větší hustotou obyvatel, tak i dálniční síť. Hlavní testovací oblastí bylo město Brno a přilehlé oblasti včetně dálnice D1.

Na vytipovaných místech bylo naměřeno několik hodnot, které ve výsledném zpracování vytvořili shlukovou oblast. Jedná se například o křižovatku u Semilassa, kde při přejezdu tramvajových kolejí lze naměřit dvě po sobě jdoucí nerovnosti a stejného výsledku bylo dosaženo i měřením. Výsledný shluk z této oblasti byl ilustrován na obrázku 7.4 v předešlé kapitole, kde bylo diskutováno o různých typech zobrazení mapového podkladu.

Součástí tvorby aplikace bylo vytvoření krátkého demonstračního videa, sloužící pro základní pochopení a koncept ovládání aplikace přes grafické uživatelské rozhraní. Ve videu je znázorněno jak aplikace pracuje v normálním provozu i při přejezdu v oblasti s výskytem nerovnosti a jak probíhá záznam takové oblasti. Video je uloženo na nosiči CD přiloženém k této práci.

Naměřená data

Jak bylo uvedeno v úvodu této kapitoly, měření probíhalo z velké části v obydlené oblasti s výskytem dopravní městské hromadné dopravy. Na základě naměřených hodnot lze kategorizovat v této oblasti několik typů nerovností.

Prvním typem jsou výmoly, díry a různé typy prohlubní, ve kterých je velké nebezpečí vzniku defektu kola nebo jiného typu poškození při přejezdu oblasti i v nižších rychlostech.

Druhým typem těchto oblastí jsou překážky. Do této oblasti lze zařadit retardéry, nájezdné stanice městské hromadné dopravy nebo různé typy vyvýšenin. Při přejezdu ve vyšších rychlostech může dojít k poškození vozidla, ale výskyt těchto oblastí je z určité vzdálenosti viditelný a lze předejít poškození.

Posledním typem je zmíněná městská hromadná doprava. Kde jsou naměřeny především přejezdy kolejí ať už vlakové nebo tramvajové. Přejezdy kolejí, které jsou naměřeny z převážně příčné jízdy, tedy kolmo na danou trať kolejiště. Tento typ oblastí není až tak kriticky nebezpečný na přejezd vozidlem, ale i zde je riziko poškození a je doporučeno snížit rychlost. Data, která byla doposud naměřena jsou součástí této práce a jsou přílohou tohoto dokumentu na doprovodném CD disku.

Rozšíření

Pokračování vývoje by směřovalo na testování různých typů automobilů. Především s různým typem podvozku, kde by na základě naměřených hodnot u oblastí, které jsou již zaznamenané probíhalo vyhodnocení prahové funkce a přizpůsobení tohoto prahu pro konkrétní typ automobilu. Druhou možností jak eliminovat testování různých typů automobilů, je umístit zařízení snímající vibrace na tlumící soustavu pérování vozidla.

Rozšíření týkající se softwarové aplikace je především v grafickém uživatelském rozhraní, kde by interakce s uživatelem v podobě hlasové navigace nebo upozornění při blížící se oblasti s nerovnostmi byla vítána velkou skupinou řidičů. V aktuálním stavu je aplikace použitelná pro měření stavu a grafické uživatelské rozhraní slouží pro zajímavé informace o dané trase, ale chybějící funkce navigace může mít za následek spuštění služby na pozadí a zavření aktivního okna s grafickým rozhraním.

Aplikace je plně funkční při dostupnosti orbitálních družic k určení GPS pozici Země, ale například v oblastech kde je signál omezen nebo nedostupný je funkčnost omezena. Typické oblasti jsou například tunely, kde GPS signál není. Nabízí se rozšíření využít další senzor, který je na mobilním zařízení dostupný a to gyroskop. Na základě tohoto senzoru a již používaného akcelerometru by bylo možné interpolovat polohu automobilu dle informací o zrychlení a případném odbočení vozidla.

V serverové částí by rozšíření mohlo rozdělovat naměřené oblasti do shluků a přidělovat je do skupin. Například v rámci státu, aktuálně jsou do mobilního zařízení stahovány všechny oblasti s výskytem výmolů a nerovnostmi. Při globálním používání bude mobilní zařízení a přenos mezi serverem a zařízením zatížen daty, které uživatel nemusí využít. Pro vytvoření a udržení komunity by bylo možné vytvořit upravené grafické rozhraní serverové části, kde by bylo možné diskutovat a vyměňovat si názory s oblasti měření povrchu vozovky a získat tak zpětnou vazbu na daný projekt.

Kapitola 9

Závěr

Cílem práce bylo získat ucelené informace o tom, zda je možné využít senzory mobilního zařízení v kokpitu automobilu. Zda je tu možnost rozšířit vybavenost dnešních moderních automobilů a jejich senzorů. Součástí práce bylo navrhnout a implementovat aplikaci na mobilním zařízení a otestovat ji v automobilu. Přesné výsledky jsou shrnuty v předešlých kapitolách této práce, ale lze shodnotit, že lze využít mobilní zařízení v kokpitu automobilu. Automobil je možné rošířit o senzory v mobilním zařízení, ale je vhodné implementační řešení připravit na různé modifikace dle typu vozidla. Naměřené hodnoty se vždy budou lišit v závislosti na typu podvozku.

Implementace probíhala na platformě Android, základní popis platformy včetně vývoje a možnosti přístupu k senzorům lze nalézt v kapitole 2. Pro práci se senzory bylo nutné nastudovat, jak akcelerometr pracuje a jaká data lze očekávat na výstupu. Problematika zpracování diskrétního signálu a práce se senzory je popsána v kapitole 3. Signál ze senzorů je nutné zpracovat a filtrovat a tím se zabývá kapitola 4.

Návrh mobilní aplikace se zabýval snímání povrchu vozovky a rozhodnutelnosti, zda se jedná o oblast s výskytem výmolů, překážek nebo jiného typu nerovnosti. Tato nerovnost by mohla způsobit poškození automobilu při přejezdu touto oblastí ve vyšší rychlosti. Informace týkající se mobilní aplikace lze nalézt v kapitole 2. Tato kapitola obsahuje řešení problémů při vývoji aplikace a způsob zpracování a uložení dat na mobilním zařízení. Mobilní aplikace je spojena s dostupným serverovým řešením, které je navrženo pro zpracování dat ze všech mobilních zařízení. Návrh, implementace a řešení shlukování naměřených dat od uživatelů je řešeno v části o internetové aplikaci a tím se zabývá kapitola 7.

Shrnutí navržené aplikace je hodnoceno v kapitole 8. V této kapitole lze nalézt i případná rozšíření a možné úpravy do budoucna.

Literatura

[1] MySQL [online]. http://en.wikipedia.org/wiki/MySQL, 2012-05-03 [cit. 2013-05-04].

[2] Jak psát web [online]. http://http://www.jakpsatweb.cz, 2012-12-06 [cit. 2013-05-02].

[3] Dokumentace k platformě Android [online]. http://developer.android.com, 2013-01-01 [cit. 2013-01-09].

[4] HTTP Components. http://hc.apache.org, 2013-03-24 [cit 2013-5-10].

[5] Allen, S.; Graupera, V.; Lundrigan, L.: *Pro smartphone cross-platform development: iPhone, blackberry, windows mobile and android development and distribution.* Apress, 2010, iSBN 978-1430228684.

[6] Bouchala, J.: Úvod do funkcionální analýzy. *Matematika pro inženýry*, ročník 21, 2012.

[7] Coggeshall, J.: *PHP 5 unleashed.* Sams, 2005, iSBN 978-0672325113.

[8] Coskun, V.; Ok, K.; Ozdenizci, B.: *Professional NFC Application Development for Android.* Wrox, 2013, iSBN 978-1-118380093.

[9] Fielding, R.; Gettys, J.; Mogul, J.; aj.: RFC 2616-HTTP/1.1, the hypertext transfer protocol. 1999.

[10] Fraden, J.: *Handbook of modern sensors : physics, designs, and applications.* American Institute of Physics, 1993, iSBN 978-1563961083.

[11] Goodman, D.; Morrison, M.; Eich, B.: *Javascript bible.* John Wiley & Sons, Inc., 2007, iSBN 978-0470526910.

[12] Graham, I. S.: *The HTML sourcebook.* John Wiley & Sons, Inc., 1995, iSBN 978-0471257240.

[13] Hashimi, S.; Komatineni, S.; MacLean, D.: *Pro Android 2.* 2, Apress, 2010.

[14] Hickson, I.; Hyatt, D.: Html5. *W3C Working Draft WD-html5-20110525, May*, 2011.

[15] Hojgr, R.; Stankovič, J.: *GPS: Praktická uživatelská příručka.* Computer Press, 2007, 256 s., iSBN 978-80-251-1734-7.

[16] Komatineni, S.; MacLean, D.: *Pro Android 4.* Apress, 2012, iSBN 978-1430239307.

[17] Kosek, J.: *HTML tvorba dokonalých WWW stránek: Podrobný průvodce.* Grada, 1998, iSBN 807-1-696080.

[18] Meier, R.: *Professional Android 4 application development.* Wrox, 2012, iSBN 978-1118102275.

[19] Milette, G.; Stroud, A.: *Professional Android Sensor Programming.* Wrox, 2012, iSBN 978-1118183489.

[20] Moncur, M. G.: *Teach yourself JavaScript in 24 hours.* Sams Publishing, 2002, iSBN 978-0672336089.

[21] Owens, M.: *The Definitive Guide to SQLite.* Apress, 2006, iSBN 978-1-59059-673-9.

[22] Tuček, J.: *GIS - Geografické informační systémy: Principy a praxe.* Computer Press, 1998, 438 s., iSBN 80-7226-0691-X.

Příloha A

Obsah CD

- Zdrojové soubory k mobilní aplikaci
- Zdrojové soubory k internetové aplikaci
- Instalační soubor mobilní aplikace (APK balíček)
- Demonstrační video
- Obrázky a grafy použité v této práci
- Naměřená data