ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
OBOR GEOMATIKA

DIPLOMOVÁ PRÁCE

VYUŽITÍ CHYTRÝCH MOBILNÍCH TELEFONŮ PRO URČENÍ POLOHY POMOCÍ NOVÉHO GOOGLE API

Vedoucí práce: Prof. Ing. Aleš Čepek, CSc.
Katedra geomatiky

červen 2019

Michael KALA

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta stavební
Thákurova 7, 166 29 Praha 6

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Kala Jméno: Michael Osobní číslo: 394824

Zadávající katedra: Katedra geomatiky

Studijní program: Geodézie a kartografie

Studijní obor: Geomatika

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce: Využití chytrých mobilních telefonů pro určení polohy pomocí nového Google API

Název diplomové práce anglicky: The use of smartphones for positioning using new Google API

Pokyny pro vypracování:
Student se seznámí a naučí pracovat s novým Google API, jež umožňuje získávat surové GNSS měření pomocí chytrého mobilního telefonu. Poté implementuje aplikaci pro jejich sběr a s použitím modifikace otevřeného software G-Nut/Anubis tato data zpracuje pro určení polohy uživatele. Hlavním cílem je porovnání polohy určené pomocí surových měření se standardními výstupy polohy z GNSS čipu.

Seznam doporučené literatury:
[1] The GSA GNSS Raw Measurements Task Force (2017), Using GNSS raw measurements on android devices, https://www.gsa.europa.eu/system/files/reports/gnss_raw_measurement_web_0.pdf

[2] J. Douša, P. Václavovic (2017), G-Nut/Anubis a tool for Multi-GNSS data quality control Tutorial 2017, http://www.epncb.oma.be/_newseventslinks/workshops/EPNLACWS_2017/pdf/Tutorials/2017-EUREF-LAC-AnubisTutorial

[3] P. Václavovic, J. Douša (2016), G-Nut/Anubis - open-source tool for multi-GNSS data monitoring, IAG Symposia Series, Springer, Vol. 143, pp. 775-782

Jméno vedoucího diplomové práce: Prof. Ing. Aleš Čepek, CSc.

Datum zadání diplomové práce: 20. 2. 2019 Termín odevzdání diplomové práce: 20. 5. 2019
Údaj uveďte v souladu s datem v časovém plánu příslušného ak. roku

[redacted]
Podpis vedoucího práce

[redacted]
Podpis vedoucího katedry

III. PŘEVZETÍ ZADÁNÍ

Beru na vědomí, že jsem povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je nutné uvést v diplomové práci a při citování postupovat v souladu s metodickou příručkou ČVUT „Jak psát vysokoškolské závěrečné práce“ a metodickým pokynem ČVUT „O dodržování etických principů při přípravě vysokoškolských závěrečných prací“.

Datum převzetí zadání

[redacted]
Podpis studenta(ky)

ABSTRAKT

Práce se zabývá zpracováním surových GNSS měření, která lze nově získat prostřednictvím mobilních telefonů s operačním systémem Android verze 7.0 a vyšší. V rámci práce je vytvořena vlastní aplikace GNSS Agent pro sběr takových dat. Pro analýzu dat jsou použity otevřené nástroje G-Nut/Anubis a G-Nut/Geb vyvíjené na Geodetické observatoři Pecný. Výpočet polohy ze surových pozorování je porovnáván s výpočtem polohy z mobilního GNSS čipsetu. Dále je v práci popsáno nové Google API, jež surová data poskytuje. Popsán je i základ programování pro Android zásadní pro vývoj aplikací.

KLÍČOVÁ SLOVA

GNSS, GNSS Agent, Android, surová GNSS měření, G-Nut/Anubis, G-Nut/Geb

ABSTRACT

This thesis deals with processing of raw GNSS measurements, that are newly available via using smartphones with Android version 7.0 and higher. New mobile application GNSS Agent is implemented for this purpose. Open-source tools G-Nut/Anubis and G-Nut/Geb created on Geodetic observatory Pecný are used for analysis of the received data. Computed location with usage of raw data and location that comes from mobile GNSS chipset are compared. Also the topics of new Google API for raw GNSS measurements and basics of Android programming are described in the thesis.

KEYWORDS

GNSS, GNSS Agent, Android, raw GNSS measurements, G-Nut/Anubis, G-Nut/Geb

PROHLÁŠENÍ

Prohlašuji, že diplomovou práci na téma „Využití chytrých mobilních telefonů pro určení polohy pomocí nového Google API“ jsem vypracoval samostatně. Použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval Ing. Janu Doušovi, Ph.D. za umožnění spolupráce na tématech týkajících se této diplomové práce, za konzultace a za vždy pozitivní přístup, jež mě motivoval pokračovat. Dále bych rád poděkoval prof. Ing. Alešovi Čepkovi, CSc. za vedení práce. Díky patří také Ing. Pavlu Václavovicovi, Ph.D. za cenné rady a odbornou pomoc přes elektronickou poštu i při četných návštěvách na Geodetické observatoři Pecný. V neposlední řadě děkuji i rodině a přátelům, jež mě podrželi, i když jsem nevěděl, že to potřebuji.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

ADT Android development tools (vývojářské nástroje pro Android)

API Application Programming Interface (rozhraní pro programování aplikací)

AVD Android virtual device (virtuální zařízení Android)

BNC BKG Ntrip Client

C/N_0 Carrier to noise density ratio (poměr signálu a šumu)

ESA European Space Agency (Evropská kosmická agentura)

GLONASS Globalnaja navigacionnaja sputnikovaja sistěma (Globální navigační družicový systém)

GMS Google Mobile Services (Google mobilní služby)

GNSS Global Navigation Satellite System (Globální družicový polohový systém)

GOPE Geodetická observatoř Pecný

GPS Global Positioning System (Globální polohový systém)

GUI Graphical user interface (grafické uživatelské rozhraní)

IGS International GNSS Service

PRN Pseudo-Random Noise (pseudonáhodný kód)

OS Operating system (operační systém)

RINEX Receiver Independent Exchange format (nezávislý výměnný formát)

SDK Software development kit (systémové vývojové nástroje)

TOW Time Of Week (čas v týdnu)

XML Extensible Markup Language

1 Úvod

Jednou z důležitých funkcí mobilního telefonu je určování aktuální polohy uživatele. Pro vývojáře je implementace takové úlohy více než přímočará, v podstatě stačí telefon o polohu požádat. Čipset zodpovědný za zjišťování polohy se chová jako tzv. 'černá skříňka', lze od něj získat pouze omezené informace týkající se pozorování GNSS družic a dalších výpočtů. V roce 2016 se s příchodem verze 7.0 Nougat operačního systému Android tento přístup změnil. Nově je možné od GNSS čipsetu získávat surová měření, která jsou pro výpočet polohy využívána. To umožňuje implementaci celé řady nových aplikací s vlastními algoritmy pro zjištění polohy nebo s jinými funkcionalitami využívajícími nově dostupné informace. Navíc na trh začínají vstupovat mobilní telefony podporující dvoufrekvenční pozorování GNSS, jež by se mělo projevit zvýšením přesnosti zjišťování polohy. Kombinací těchto nových telefonů a surových GNSS měření je možné spočítat polohu s přesností pod 1 metr.

Hlavním cílem této diplomové práce je otestování nového přístupu, potvrzení popř. vyvrácení tvrzení o zlepšení přesnosti měření a porovnání takto získané polohy s polohou získanou přímo z výpočtů GNSS čipsetu. Pro tyto účely je v rámci diplomové práce vytvořena vlastní mobilní aplikace GNSS Agent, která má sloužit pro sběr a základní zpracování surových GNSS měření. Nasbíraná data jsou dále zpracovávaná za využití otevřených nástrojů G-Nut/Anubis a G-Nut/Geb vyvíjených na Geodetické observatoři Pecný. Tyto nástroje slouží pro kontrolu kvality dat a výpočty polohy. Nástroj G-Nut/Anubis je dále použit i pro analýzu nasbíraných dat již existující aplikací Geo++ RINEX Logger pro porovnání rozdílů v implementaci s aplikací GNSS Agent. Pro sběr dat je využit mobilní telefon Xiaomi Mi 8, jež je prvním zařízením podporujícím dvoufrekvenční pozorování.

Vedle analýzy surových měření a vývoje aplikace GNSS Agent se diplomová práce zabývá úvodem do programování pro Android, jež je pro tvorbu takové aplikace esenciální. V několika kapitolách je popsána základní struktura projektu aplikace a její základní komponenty. Součástí diplomové práce je také popis a využití nového Google API, jež surová GNSS data poskytuje. V neposlední řadě práce obsahuje důležitý teoretický základ týkající se globálních navigačních systémů.

2 Globální navigační satelitní systémy

Tato kapitola slouží jako stručný úvod do problematiky globálních navigačních satelitních systémů, který by měl definovat základní pojmy a výpočty, na které se dále tato diplomová práce bude odkazovat.

Existuje několik globálních navigačních satelitních systémů, které lze využít k získání polohy místa na Zemi. Každý z těchto systémů se skládá z kosmického segmentu (konstelace satelitů) a řídícího segmentu. Hlavní myšlenkou GNSS je měření vzdáleností mezi družicemi a uživatelem na Zemi nebo v jejím nejbližším okolí (v řádu desítek až stovek kilometrů). Informace umožňující výpočet polohy satelitu jsou satelitem vysílány uživateli. Řídící segment zajišťuje provoz celého navigačního systému - monitoring družic, určování drah družic, komunikaci s družicemi atd. [1]. Základní globální navigační systémy jsou:

- **GPS** (Global Positioning System) je americký navigační systém spravovaný Letectvem Spojených států amerických. Kosmický segment GPS se v současné době skládá z 31 provozuschopných umělých družic se zárukou dostupnosti minimálně 24 družic se spolehlivostí minimálně 95% v každém okamžiku. Družice obíhají planetu Zemi na středních drahách ve výšce kolem 20 200 km nad povrchem Země. Družice jsou rozestaveny tak, aby alespoň čtyři byly viditelné z každého místa na Zemi. Informace o stavu družic a obecně celého systému lze najít na oficiálních webových stránkách GPS [2].

- **GLONASS** ('Globalnaja navigacionnaja sputnikovaja sistěma') je ruský navigační systém, jehož kosmický segment se skládá z celkového počtu 26 umělých družic. V době psaní této diplomové práce bylo operatibilních 24 družic, 1 záložní a 1 v testování. Družice se nacházejí na orbitě ve výšce 19 100 km. Podobně jako u GPS jsou v každé chvíli na každém místě na Zemi viditelné minimálně 4 družice [3].

- **BeiDou** je globální navigační systém vyvíjený Čínskou lidovou republikou. V současné době je v kosmickém segmentu BeiDou celkem 38 družic, z toho 17 plně operujících [4].

- **Galileo** je globální navigační systém vyvíjený Evropskou unií s pomocí Evropské kosmické agentury (ESA). V současné době je operatibilních 21 družic, v roce

2020 by jich mělo být 30. Na většině místech na Zemi bude viditelných vždy 6-8 družic. Družice obíhají ve výšce 23 222 km. Na rozdíl od ostatních globálních navigačních systémů není Galileo řízeno armádou [5].

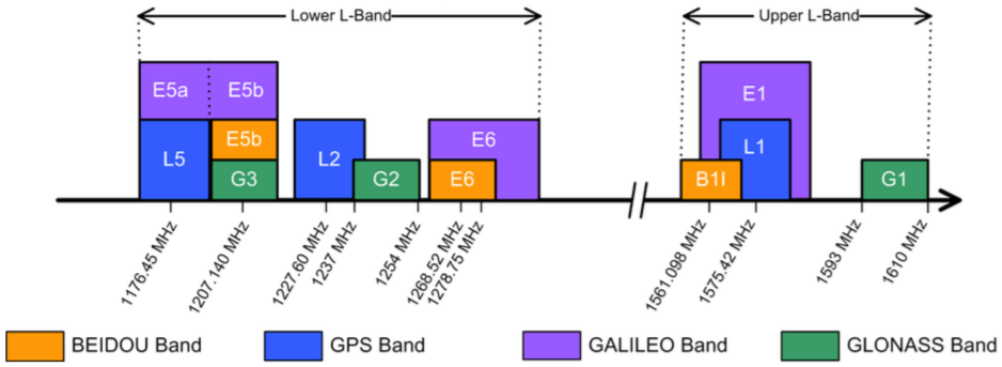
Tyto 4 systémy jsou dále doplňovány několika dalšími regionálními navigačními nebo jinými zpřesňujícími systémy.

2.1 GNSS signál

Družice nepřetržitě vysílají signál obsahující informace pro výpočet polohy přijímače. Tento signál se skládá ze 3 hlavních složek - nosné vlny, kódu a navigační zprávy [6].

2.1.1 Nosná vlna

Na následujícím obrázku 2.1 jsou zobrazena jednotlivá frekvenční pásma, na kterých navigační systémy vysílají.



Obrázek 2.1: Frekvenční pásma [7]

Pro tuto diplomovou práci jsou zásadní pásma L1 a L5 pro GPS, E1 a E5 pro Galileo, G1 pro GLONASS a B1 pro BeiDou. V současné době dostupná mobilní zařízení dokáží zpracovávat signál právě na frekvencích z těchto pásem.

2.1.2 Kód

S výjimkou GLONASS vysílají družice jednotlivých navigačních systémů signál s použitím shodných frekvencí. Nosná vlna je modulována sekvencí tzv. pseudonáhodného kódu ('Pseudo-Random Noise' - PRN), který mimo jiné slouží k rozpoznání,

o jakou družici se jedná. Pro každou družici je tento kód unikátní. Přijímač porovnává repliku těchto kódů a tak zjišťuje, ze kterého satelitu přijal data. Tento způsob se nazývá **CMDA** ('Code Division Multiple Access').

Kód PRN je pseudonáhodný, jelikož se na první pohled jedná o náhodnou sekvenci nul a jedniček, nicméně pro každý satelit je tato sekvence unikátní (kromě systému GLONASS). PRN dále slouží k určení časového rozdílu mezi vysláním a příjmem signálu.

Pro systém GLONASS je každá družice modulována stejným kódem, ale vysílá na jiné frekvenci odvozené od základní frekvence přičtením nebo odečtením násobku frekvenčního rozdílu. Těchto frekvencí je celkem 14:

$$G_{1_k} = G_{1_0} + k \cdot \Delta G_1, \tag{2.1}$$

kde G_{1_0} je základní frekvence (pro pásmo G1 je to $1602MHz$), ΔG_1 je malý frekvenční rozdíl ($0.5625MHz$) a $k = -7, ..., 0, ..., 6$. Tento postup se nazývá **FDMA** ('Frequency Division Multiple Acces'). Jelikož kanálů, na kterých vysílají družice GLONASS, je pouze 14, vysílají některé družice na stejných frekvencích. Nicméně systém je navržený tak, aby družice se stejnou frekvencí nebyly viditelné zároveň.

2.1.3 Navigační zpráva

Všechny družice získávají informace o své poloze z řídících stanic. Tyto informace potom vysílají nazpátek do přijímačů prostřednictvím navigační zprávy, jež obsahuje všechny podstatné údaje postačující k provedení výpočtu polohy. Důležité součásti navigační zprávy jsou [6], [1]:

- Efemeridy družice - údaje o její poloze
- Korekce atomových hodin na družici
- Informace o stavu satelitu
- Almanach - informace o ostatních družicích v systému pro určení jejich hrubé polohy
- Parametry ionosférického modelu (GPS a BeiDou)

2.2 Systematické chyby pozorování

Pozorování GNSS signálu jsou mimo náhodné chyby ovlivněna také řadou systematických jevů, u kterých je třeba v závislosti na požadované přesnosti a kvalitě měření (kódové a fázové měření, viz kap. 2.3) rozhodnout, zda je modelovat a zahrnout do výpočtů polohy. Dle [8] se dají nejdůležitější z těchto jevů rozdělit do tří kategorií, jež jsou stručně popsané v následujících podkapitolách. V kapitole je čerpáno především z [9].

2.2.1 Chyby způsobené družicí

- **Chyba hodin** - Přestože jsou družice vybavené atomovými hodinami, jejich přesnost stále není dostačující pro přesné určování polohy z měření tranzitního času při šíření elektromagnetického signálu z družice k přijímači. Posuny chodu družicových hodin jsou pozorovány monitorovacími stanicemi, jež počítají jejich korekce. Ty jsou poté vysílány uživateli v navigační zprávě.

- **Chyba efemeridy** - Řídící segment předpovídá efemeridy družic, jež jsou následně vysílány v navigační zprávě. Tyto předpovědi obsahují nepřesnosti, jež způsobí chybu ve výpočtu polohy družice v řádu až desítek metrů.

- Dalšími chybami na straně družice jsou např. excentricita a variace fázového centra antény. V režimu post-processing lze všechny chyby opravit pomocí přesných produktů poskytovaných službou IGS (International GNSS Service).

2.2.2 Chyby způsobené přijímačem

- **Chyba hodin** - Přijímač zpravidla neobsahuje přesné atomové hodiny (zejména z finančních důvodů). Hodiny přijímače tak mají oproti hodinám družice velmi nízkou přesnost. Pro eliminaci této chyby se při počítání polohy zavádí hodnota chyby hodin přijímače jako další neznámá. Z tohoto důvodu je třeba k určení polohy přijímače a opravy chodu hodin získat současné měření alespoň ze 4 družic.

- **Variace fázového centra antény** - Měření GNSS je vztahované k fázovému centru antény, k pomyslnému bodu, který není shodný s geometrickým středem antény. Jeho poloha je různá podle frekvence přijímaného signálu a závisí směru dopadu [8].

2.2.3 Chyby způsobené prostředím

- **Ionosférická refrakce** - Ionosféra je vrstva zemské atmosféry obsahující volné elektrony působící na šíření rádiového signálu. Ionosférická refrakce je jedním z efektů způsobujících největší systematické chyby v pozorování GNSS. Disperzní prostředí ionosféry působí různě na vlny o různých frekvencích. Čím nižší frekvence, tím je signál ovlivněn více. U jednofrekvenčních přijímačů je třeba pro eliminaci ionosférické refrakce využít ionosférického modelu. U dvoufrekvenčního přijímače lze efekt ionosférické refrakce téměř celý eliminovat pomocí lineární kombinace kódového nebo fázového měření [10].

- **Troposférická refrakce** - Troposféra je nedisperzní prostředí pro frekvence až do 15 GHz, což znamená, že všechny GNSS signály jsou troposférickou refrakcí ovlivňovány stejně. V důsledku tohoto efektu nelze na rozdíl od ionosférické refrakce tuto chybu eliminovat kombinací měření na dvou frekvencích. Přibližně 90% troposférické refrakce způsobuje suchá část atmosféry tvořená směsí suchých plynů (dusík, kyslík, argon...). Zbytek je způsoben mokrou složkou, která se skládá z vodních par a zkondenzované vody ve formě mraků. Suchou část lze modelovat při znalosti tlaku a teploty za použití zákonů ideálních plynů. Vytvořit model vlhké části je kvůli horší předvídatelnosti vstupních parametrů mnohem komplikovanější [8].

2.3 Měření

2.3.1 Měření času

Design GNSS technologií závisí na přesném měření času při příjmu a vysílání rádiového signálu. Pro úspěšné výpočty polohy je třeba ke každému signálu připojit přesnou časovou značku. Proto má každá družice velmi přesné atomové hodiny, které jsou synchronizovány pomocí řídícího segmentu [11].

Atomová sekunda byla na XIII. konferenci Mezinárodního komitétu pro váhy a míry v Paříži v roce 1967 definována následovně: "*Atomová sekunda je určena dobou 9 192 631 770 kmitů, které přísluší atomu cesia 133 při přechodu mezi hladinami $F = 4$, $M = 0$ a $F = 3$, $M = 0$ základního stavu $^2S_{1/2}$ bez vlivu vnějších magnetických polí*". Systém mezinárodního atomového času TAI je založen na váženém průměru více než 400 atomových hodin ve více než 50 národních laboratoří.

Čas UT1 je rotačním časem reflektujícím nerovnoměrnosti rotace Země. V časové škále UT1 Země rotuje rovnoměrně. Časy UT1 a TAI se rozcházejí, proto je zaveden čas UTC (Universal Time Coordinated), jež je odvozen z TAI a vázaný na UT1. Vztah mezi UT1 a UTC je:

$$|UT1 - UTC| < 0.9s. \tag{2.2}$$

Aby byla splněna podmínka v rovnici 2.2, jsou při potřebě zaváděny tzv. přestupné sekundy ('leap seconds') 30. 6. nebo 31. 12. [12]. Dle [13] byl v době psaní diplomové práce počet přestupných sekund, tedy rozdíl mezi časy UTC a TAI:

$$leapseconds = UTC - TAI = -37s. \tag{2.3}$$

Každý z navigačních systémů používá svůj vlastní referenční čas. Tyto časy a jejich vztah k TAI je uveden v následující tabulce 2.1:

Systém	**Referenční čas**	**Vztah k TAI**
GPS	GPST	$GPST = TAI - 19s$
Galileo	GST	$GST = TAI - 19s$
GLONASS	GLONASST	$GLONASST = TAI + 3h - leapseconds$
BeiDou	BDT	$BDT = TAI - 33s$

Tab. 2.1: Referenční časy GNSS [11]

V tabulce 2.1 hodnota *leapseconds* odpovídá počtu přestupných sekund mezi časy UTC a TAI (viz rovnice 2.3).

Začátek GPS času (GPST) je datován k 6. 1. 1980 v 00:00 (UTC). V této epoše byl rozdíl mezi UTC a TAI 19 sekund. GPST je určován pomocí dvou parametrů - číslo týdne a čas v týdnu (TOW) v sekundách. Čas systému Galileo (GST) byl definován tak, aby mezi ním a GPST nebyl žádný rozdíl [11].

2.3.2 Kódové měření

Kódové měření je metodou určování vzdálenosti přijímače od družice pomocí měření transitního času šíření signálu. Vynásobením tohoto času rychlostí světla je vypočtena tzv. pseudovzdálenost. Pseudovzdáleností se rozumí přibližná vzdálenost, která neodpovídá skutečné geometrické vzdálenosti, jelikož šíření signálu je ovlivněno různými negativními jevy (viz kap. 2.2)., které je třeba do výpočtu dále zahrnout [14].

2.3.3 Fázové měření

Určování vzdálenosti mezi přijímačem a družicí pomocí fázového měření je založeno na porovnání fáze nosné vlny přijaté z družice vůči replice generované přijímačem. V ideálním případě by pro získání vzdálenosti byl k rozdílu porovnávaných fází přičítán celkový počet cyklů vlny s přenásobením součtu vlnovou délkou signálu. Celkový počet vln však nelze měřením zjistit, proto vstupuje do výpočtu jako neznámá hodnota, jedná se o tzv. počáteční ambiguitu. Fázovým měřením lze tedy získat pouze relativní vzdálenost mezi epochami, nicméně jeho přesnost je o několik řádů vyšší než přesnost kódového měření. Počáteční ambiguita zůstává konstantní, dokud nedojde k přerušení signálu. Fázová měření je samozřejmě třeba dále opravovat o působení negativních jevů na šíření signálu [8].

2.3.4 Síla signálu

Síla signálu je uváděná jako C/N_0 (carrier to noise density ratio). Jedná se o poměr mezi sílou příchozího signálu a sílou šumu, jež signál ruší. Jednotkou C/N_0 je decibel-Hertz (dB-Hz) [15].

2.4 Datové formáty

2.4.1 RINEX

RINEX ('Receiver Independent Exchange format') je standardizovaný výměnný formát pro ukládání dat navigačních systémů. Umožňuje post-processingové zpracování s využitím dalších informací, jež v době měření nemusí být dostupné. RINEX se dělí na 3 druhy, observační, navigační a meteorologický. Každý z těchto souborů obsahuje hlavičku s údaji o měřící stanici a s dalšími informacemi o datech v souboru [16].

2.4.2 BNC

Formát BNC je nestandardizovaný jednoduchý formát navržený pro lokální sdílení observací v reálném čase. Na rozdíl od formátu RINEX tyto soubory neobsahují hlavičku [17].

3 Android

Obrázek 3.1: Logo OS Android [18]

Android je operačním systémem pro mobilní telefony, tablety a další zařízení jako jsou např. chytré hodinky, fotoaparáty nebo navigace. Spatřil světlo světa v roce 2003 a postupně se stal nejrozšířenějším mobilním operačním systémem. Android je otevřeným projektem (AOSP - 'Android Open Source Project'), který je veden společností Google pod hlavičkou konsorcia firem Open Handset Alliance. Google projekt používá jako základ pro tvorbu vlastní verze Androidu, která je dále využívána dalšími výrobci.

Operační systém Android je napsán primárně v jazyce Java a je založen na jádře Linuxu. První verze OS byla zveřejněna v listopadu 2007, první komerční verze Android 1.0 pak v září 2008. Pro zajímavost, verze jsou pojmenovávány abecedně vždy podle zákusku nebo jiné cukrovinky. Momentálně (v době psaní diplomové práce) je nejnovější verzí Android 9 Pie (koláč), předcházela verze 8 Oreo, předtím verze 7 Nougat atd.

OS Android je od společnosti Google doplňován o balíček vlastního softwaru, který se nazývá 'Google Mobile Services' (GMS). Ten je často předinstalován v zakoupených zařízeních a obvykle zahrnuje webový prohlížeč Google Chrome, službu Google Search (vyhledávání Google) a dále základní aplikace jako je např. emailová aplikace Gmail nebo Google Play pro stahování (zdarma či za úplatu) dalších aplikací od různých vývojářů.

Základní uživatelské rozhraní OS Android je založeno na přímé interakci, kdy uživatel se zařízením komunikuje pomocí dotykové obrazovky, která pro manipulaci zobrazovaných objektů podporuje pohyby jako je tažení, poklepání, 'roztahování' pomocí dvou prstů atd. Zařízení obsahují hardware jako je např. akcelerometr, gyroskop nebo čip pro určování polohy, který může být využíván různými aplikacemi,

např. mapami (resp. jakýmikoliv aplikacemi pracujícími s polohou) nebo jen při obyčejném otáčení obrazovky [19].

3.1 Vývoj aplikací

Právě díky otevřenosti projektu Android je možné přispívat vlastním kódem, připomínkami či vlastními aplikacemi. Nejjednoduším způsobem jak přispívat je hlášení nalezených chyb a nedostatků. Přímo na webových stránkách Android je k dispozici formulář pro nahlášení chyb v systému. Pro tvorbu vlastních aplikací existuje spousta zdrojů, které navedou začínající i pokročilé vývojáře, jak správně takovou aplikaci implementovat [20].

Aplikace rozšiřují funkcionalitu zařízení. Dle [21] mohou být psány v jazycích Java, Kotlin a C++ za využití Android SDK (Software development kit). Psaní aplikací v dalších jazycích je také možné, jsou to např. jazyky JavaScript, BASIC, Corona atd. Nicméně oficiálními jazyky pro psaní aplikací jsou zmíněné Java, Kotlin a C++. Android SDK obsahuje sadu vývojářských nástrojů. Ta zahrnuje debugger, Android emulátor, nástroj pro správu virtuálních zařízení, knihovny, ukázkové kódy, dokumentaci, tutoriály atd. SDK je dostupné pro všechny hlavní platformy OS GNU-/Linux, Windows a macOS. Jako oficiálně podporované vývojové prostředí do konce roku 2014 byla platforma Eclipse s využitím pluginu ADT (Android Development Tools) pro plnou podporu vývoje pro Android. Od roku 2015 je pak oficiální platformou pro vývoj aplikací Android Studio přímo od společnosti Google (ke konci roku 2015 byla ukončena podpora ADT pluginu) [22]. Dva základní koncepty pro Android aplikace jsou dle [23] následující:

- Do aplikací lze vstoupit z různých míst
  - Aplikace jsou sestavované jako kombinace komponent, které je možné vyvolat individuálně. Např. tzv. 'aktivita' je taková komponenta, která poskytuje uživatelské rozhraní. Hlavní aktivita ('main activity') je spuštěna při poklepání na ikonu aplikace a dá se z ní dostat do dalších komponent. Do nich se lze dostat i z jiných míst, např. z upozornění na horní liště zařízení nebo i z jiných aplikací. O komponentách aplikace je více pojednáváno v kap. 3.4.
- Aplikace se adaptují pro různá zařízení

– Aplikacím je možné např. poskytnout odlišná rozlišení pro různé velikosti obrazovek. Systém zařízení pak sám rozpozná, které z nabízených rozlišení je nejlepší. V případech, kdy aplikace vyžaduje nějaký specifický hardware, např. fotoaparát, je možné do aplikace implementovat opatření, které při případné absenci potřebného vybavení zablokuje danou funkcionalitu. Případně je možné nastavit některé prvky zařízení jako povinné. Pokud zařízení tyto požadavky nesplňuje, nástroj Google Play nedovolí danou aplikaci nainstalovat.

3.2 Ochrana soukromí

V Androidu existují opatření, která chrání soukromí uživatele. Aby měla aplikace přístup k internetu, kontaktům, fotoaparátu atp., je třeba jí udělit povolení. V závislosti na typu prvku, na který se opatření vztahuje, buď systém udělí povolení k používání automaticky nebo explicitně k jeho udělení vyzve uživatele. Myšlenkou bezpečnosti v Androidu je, že žádná aplikace nemá implicitně přístup k žádným operacím nebo prvkům, které by mohly nepříznivě ovlivnit další aplikace, operační systém nebo uživatele.

Aby bylo možné povolení udělit, je třeba, aby byla přidána do tzv. manifestu aplikace (kap. 3.3.3) [24].

3.3 Android Studio

Obrázek 3.2: Logo Android Studia [25]

Jak již bylo zmíněno, Android Studio je oficiální integrované vývojové prostředí pro vývoj aplikací pro Android. V době psaní diplomové práce byla aktuální verze Android Studia verze 3.4. Vedle editoru a vývojářských nástrojů nabízí platforma také další prvky, které zjednodušují a zlepšují práci při tvorbě aplikací [26].

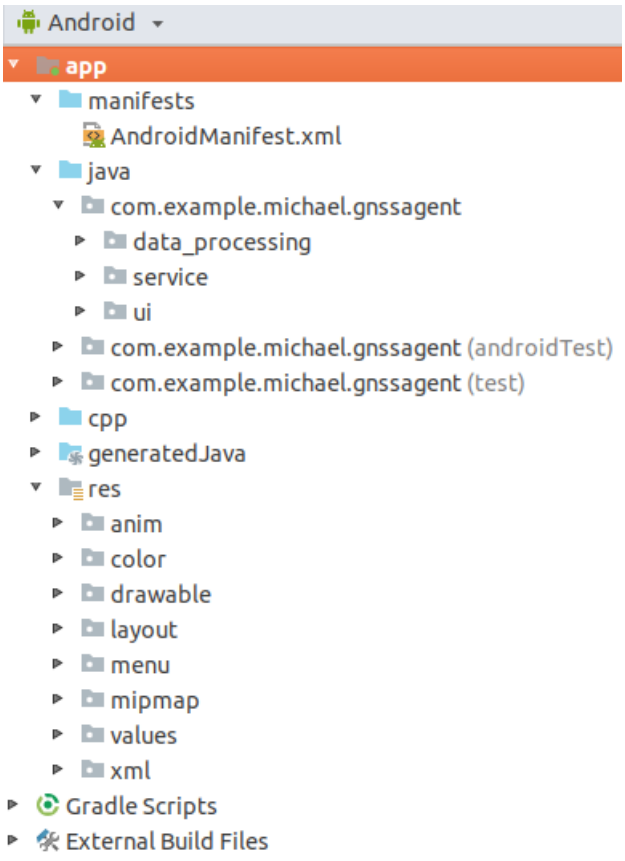
Je to např. rychlý emulátor, který bohatě nahrazuje fyzické zařízení, kdy je možné pomocí nástroje pro správu virtuálních zařízení AVD Manager (Android Virtual Device Manager) definovat charakteristiku daného mobilního telefonu, tabletu, hodinek atd. Lze vytvořit nová virtuální zařízení nebo editovat stávající, která přicházejí s distribucí Android Studia. Je možné nastavit rozměry zařízení, verzi OS, hardwarové komponenty atd. Emulátor poté skutečně funguje jako reálné zařízení, lze instalovat aplikace z Google Play, mazat data nebo např. simulovat pohyb telefonu umělým zadáním souřadnic. Především však lze v emulátoru testovat vyvíjené aplikace [27].

3.3.1 Vytvoření projektu

Při vytváření projektu v Android Studiu lze volit typ zařízení, pro které bude daná aplikace vyvíjená. Zároveň se volí verze Androidu, pro která má být aplikace kompatibilní. Aby bylo možné aplikaci pro danou verzi implementovat, je třeba mít stažené příslušné SDK podle verze API, na kterém Android běží. To lze stáhnout přes nástroj SDK Manager, který je součástí distribuce Android Studia. Dále lze zvolit, v jakém programovacím jazyce bude aplikace psána (Java/Kotlin) případně zdali má být zahrnuta podpora jazyka C++ [28].

3.3.2 Struktura projektu

Projekt aplikace je rozdělen na několik logických složek a souborů. Struktura je patrná z následujícího obrázku 3.3:



Obrázek 3.3: Struktura projektu (zdroj: vlastní)

Projekt se skládá z následujících důležitých částí:

- Soubor 'AndroidManifest.xml'
  - Soubor definující strukturu aplikace, její komponenty a požadavky. Více v kap. 3.3.3.
- Složka 'java'
  - Složka obsahující zdrojové kódy aplikace psané v jazyce Java. Soubory zdrojových kódů mohou být roztříděné do logických balíčků podobně jako v jiných projektech psaných v Javě.
- Složka 'res'
  - Zdroje, ze kterých čerpá grafické uživatelské rozhraní aplikace (GUI). Většina souborů je psaná v jazyce XML nebo jsou ukládané v jiném příslušném formátu

(např. vektorové a rastrové obrázky). GUI lze také dynamicky vytvářet a ovládat pomocí kódu v Javě, nicméně tvorba za využití XML umožňuje efektivně oddělit grafickou a funkční stránku aplikace. Jednotlivé zdroje obsahují identifikační údaj (`id`), přes který je možné k nim přistupovat (např. pro získání řetězce `S` slouží metoda `getString(R.string.S)`). V závislosti na funcionalitě jsou zdroje rozděleny do podsložek. Některé z podsložek jsou následující [29]:

- 'anim' - definice animací, např. plynulý přechod mezi jednotlivými obrazovkami,
- 'color' - definice barev a jejich změny např. při stisknutí příslušného tlačítka,
- 'drawable' - obrázky, ikony atp.,
- 'layout' - soubory definující grafické rozložení jednotlivých komponent, které se zobrazují uživateli, pro editaci lze využít grafický editor (metoda 'drag and drop'),
- 'menu' - soubory obsahující definici nabídek zobrazujících se po stisku příslušného tlačítka, např. menu s výběrem mapové vrstvy,
- 'mipmap' - ikona aplikace a její modifikace do různých rozlišení pro možnost zobrazení v jiných zařízeních nebo např. v aplikaci Google Play,
- 'values' - složka obsahující definice 'hodnot' jako jsou řetězce, barvy, barevné styly aplikace atd.,
- 'strings' - řetězce zobrazované v aplikaci a k nim přiřazená klíčová slova, pro lokalizaci je možné vytvořit soubor se stejnými klíčovými slovy a přeloženými řetězci. Aplikace dle nastavení systému telefonu rozpozná, který soubor použít.

- Složka 'Gradle Scripts' [30]
  - Složka obsahující skripty zodpovědné za sestavení projektu.
  - Gradle je otevřený nástroj pro automatizaci sestavovacího procesu umožňující definici vlastní konfigurace. Funguje nezávisle na vývojovém prostředí, je tedy možné projekt sestavovat např. pouze z příkazové řádky na počítači bez instalovaného Android Studia.
  - V souboru 'build.gradle' je definována minimální požadovaná verze Android API na cílovém zařízení

3.3.3 Manifest aplikace

Každý projekt musí obsahovat soubor `AndroidManifest.xml` obsahující nezbytné informace o aplikaci. Při vytváření projektu v Android Studiu je manifest vytvářen automaticky. Zároveň se do něj automaticky zapisují nově vytvářené komponenty aplikace. Příklad důležitých částí manifestu:

```
1  <manifest package="com.example.michael.gnssagent"
2    ... >
3    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
4    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
5    <uses-feature android:name="android.hardware.location.gps"/>
6    <uses-sdk android:minSdkVersion="24"/>
7
8    <application
9      android:label="GNSSAgent"
10     android:icon="@mipmap/ic_launcher"
11     ... >
12     <activity android:name=".MainActivity"
13       android:screenOrientation="portrait">
14       ...
15     </activity>
16
17     <service android:name=".LogService" />
18
19     <meta-data
20       android:name="com.google.android.geo.API_KEY"
21       android:value="YOUR_API_KEY" />
22     ...
23   </application>
24 </manifest>
```

Ukázka 3.1: AndroidManifest.xml (zdroj: vlastní)

Mimo jiné musí tento soubor obsahovat následující [31]:

- Název aplikačního balíčku (v ukázce 3.1 řádek 1).
- Všechny komponenty aplikace (kap. 3.4) s názvy jejich tříd (v ukázce 3.1 řádky 12-17).

- Hardware a software, který aplikace využívá. Z této informace pak čerpá služba Google Play, která (ne)umožní aplikaci nainstalovat na dané zařízení (v ukázce 3.1 řádek 5). Řádek 6 popisuje nejnižší úroveň API, kterou aplikace vyžaduje. Úroveň 24 odpovídá verzi Android 7.0.
- Potřebná povolení pro přístup k prvkům systému (v ukázce 3.1 řádky 3-4).

Používání Google Maps

Aby bylo v aplikaci možné zobrazit a využívat mapy od společnosti Google, je třeba aplikaci registrovat ve webové aplikaci 'Google Cloud Platform Console' a získat tak unikátní API klíč, který je nutno vložit do manifestu aplikace (viz v ukázce 3.1 řádky 20-21).

3.4 Komponenty aplikace

Stavebními kameny každé aplikace jsou základní komponenty realizované jako třídy v Javě. Jmenovitě jsou to: Aktivita ('Activity'), Služba ('Service'), 'Broadcast receiver' a Poskytovatel obsahu ('Content provider'). Každá z těchto komponent může fungovat jako vstup do aplikace. S rozmachem tabletů a nutnosti podpory uživatelského rozhraní zároveň pro malé a velké obrazovky byla do Androidu přidána nová komponenta Fragment ('Fragment'). Každá z uvedených součástí funguje samostatně (kromě komponenty fragment, která závisí na aktivitě) a má svůj vlastní životní cyklus, který definuje, jakým způsobem je vytvořena a rušena. V sekci jsou dále podrobněji popsány komponenty použité v praktické části diplomové práce. Čerpáno je ze zdrojů [32] a [21].

3.4.1 'Activity'

Aktivita je vstupním bodem pro interakci s uživatelem. Jedná se o jednu obrazovku zobrazující uživatelské rozhraní. Aplikace může disponovat více aktivitami, které si mezi sebou nějakým způsobem mohou předávat informace, nicméně každá z těchto aktivit pracuje samostatně a každou z nich lze (pokud je to povoleno) zavolat zvenčí aplikace. Např. při sdílení fotografie lze zavolat aktivitu aplikace Gmail pro psaní zprávy (tedy vstupem do aplikace není hlavní aktivita zobrazující se při otevření aplikace pomocí její ikony).

Při přechodu k jiné aktivitě se ta aktuální pozastaví a uloží do paměti do zásobníku ('Back Stack'), který funguje na principu LIFO (Last in, first out). Tento

zásobník umožňuje pomocí navigačního tlačítka na telefonu přecházet mezi aktuální a dříve spuštěnými aktivitami. V zásobníku nemusí být uloženy pouze aktivity, které jsou součástí jedné aplikace.

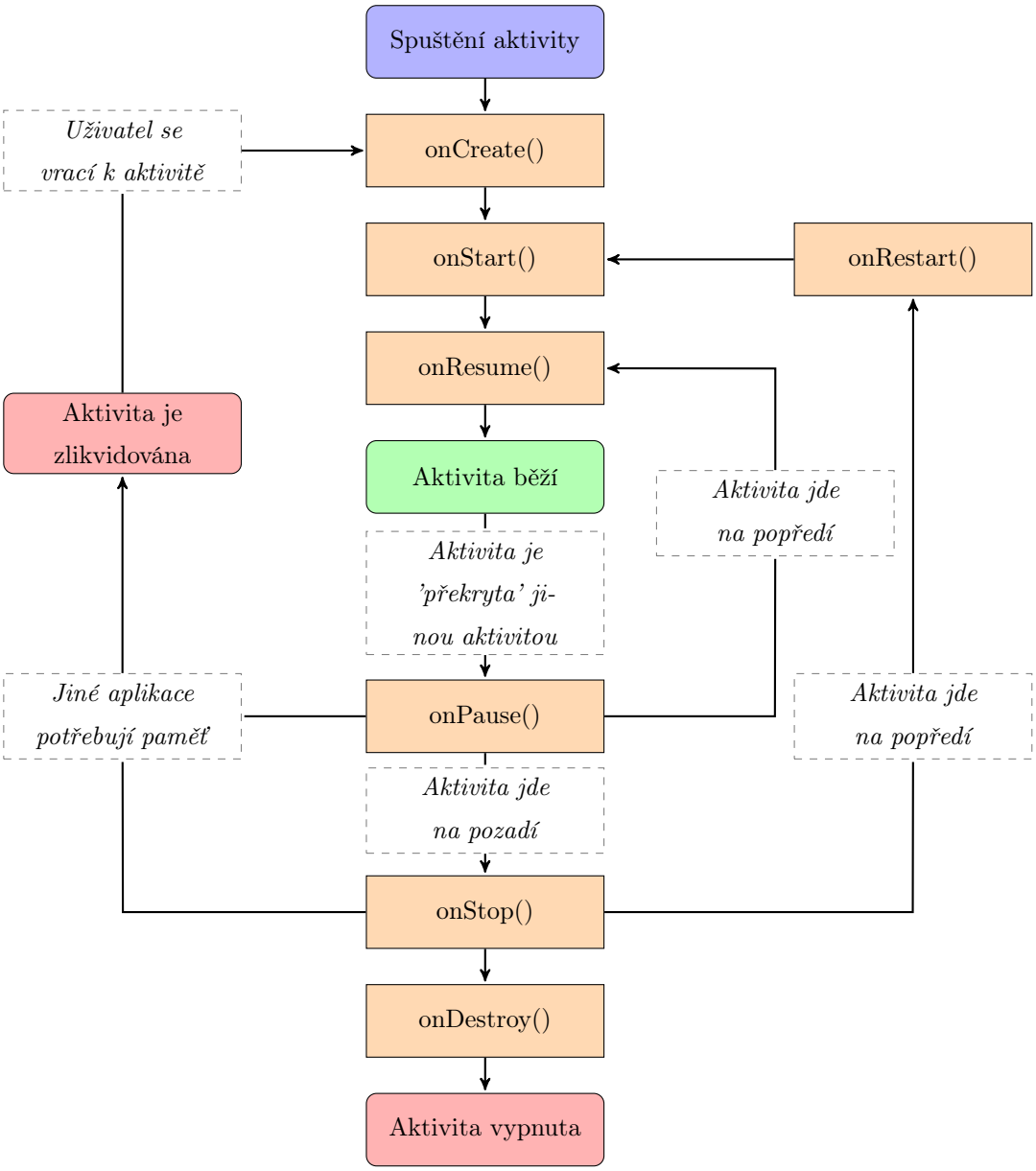
Pro správnou implementaci aktivity je třeba pochopit její životní cyklus a jakým způsobem s aktivitou nakládají různé procesy nebo uživatelské interakce. Životní cyklus aplikace a metody s ním související jsou patrné z diagramu na obrázku 3.4. Každá z metod je volána při vstupu aktivity do příslušného stavu.

Při opouštení aktivity jsou systémem zavolány metody pro její 'odklizení'. V některých případech může být tento proces pouze částečný; aktivita zůstává v paměti (např. při přepnutí do jiné aplikace). Pokud se uživatel k aktivitě vrátí, aktivita je znovu přesunuta na popředí a pokračuje od bodu, ve kterém byla uložena do paměti. Z toho vyplývají 3 hlavní fáze aktivity:

- **Aktivita je na popředí** - aktivita má 'fokus', tj. intereaguje s uživatelem. Aktivita se nachází ve stavu 'Created', 'Started' nebo 'Resumed'. V těchto stavech existuje minimální pravděpodobnost, že bude aktivita zlikvidována systémem z paměťových důvodů.

- **Aktivita je pozastavená** - aktivita je v paměti, může být částečně viditelná (např. překrytá dialogovým oknem), ztrácí 'fokus', tj. uživatel k ní nemá přístup. Aktivita se nachází ve stavu 'Paused'. Systém aktivitu v tomto stavu ukončí s vyšší pravděpodobností než v předchozích případech.

- **Aktivita je zastavená** - aktivita je stále v paměti, ale je kompletně překryta jinou. Aktivita se nachází ve stavu 'Stoped' nebo 'Destroyed'. V této fázi bude aktivita systémem z paměťových důvodů odklizena nejpravděpodobněji s porovnáním s předchozími případy.



Obrázek 3.4: Životní cyklus aktivity (zdroj: vlastní, data z [33])

onCreate()

Třída implementující aktivitu neobsahuje veřejný konstruktor, k její inicializaci slouží metoda `onCreate()`. V této metodě by měly být provedeny všechny úkony potřebné pro rozběhnutí aktivity a takové, které by se během života aktivity neměly opakovat. Pokud existuje uložený stav aktivity, je této metodě předáván jako argument.

onStart(), onResume()

Po dokončení metody `onCreate()` je ihned volána metoda `onStart()`, která je zodpovědná za zobrazení GUI. Podobně jako v předchozím případě je ihned po provedení této metody volána následující, `onResume()`, po jejímž zavolání je aplikace v příslušném stavu ('Resumed'), dokud je aktivita na popředí. V případě přerušení aktivity (např. z důvodu příchozího hovoru) je systémem spuštěna metoda `onPaused()`.

onPause()

Zavolání této metody indikuje, že uživatel opouští danou aktivitu, tj. aktivita přechází z popředí na pozadí. V této metodě by mělo být zajištěno pozastavení procesů, které nemusí běžet, pokud není aktivita na popředí (např. u navigace přechod od častému zjišťování přesné polohy k méně častému a s nižší přesností). U stavu 'Paused' se předpokládá, že se aktivita za krátký čas vrátí do stavu 'Resumed'.

Nově od Android 7 je možné rozdělit obrazovku telefonu pro zobrazení více aplikací. Pouze jedna z těchto aplikací má 'fokus', ostatní jsou ve stavu 'Paused', přestože jsou viditelné. V takových případech je u procesů, které ukazují nějakou informaci, doporučeno zařadit jejich pozastavení až do metody `onStop()`.

onStop()

Tato metoda je volána v případě, že aktivita vstoupila do pozadí a není viditelná. To se stane např. při zapnutí jiné aplikace, která kompletně překryje celou aktuální obrazovku. Systém může zavolat metodu `onStop()` před jejím ukončením.

onDestroy()

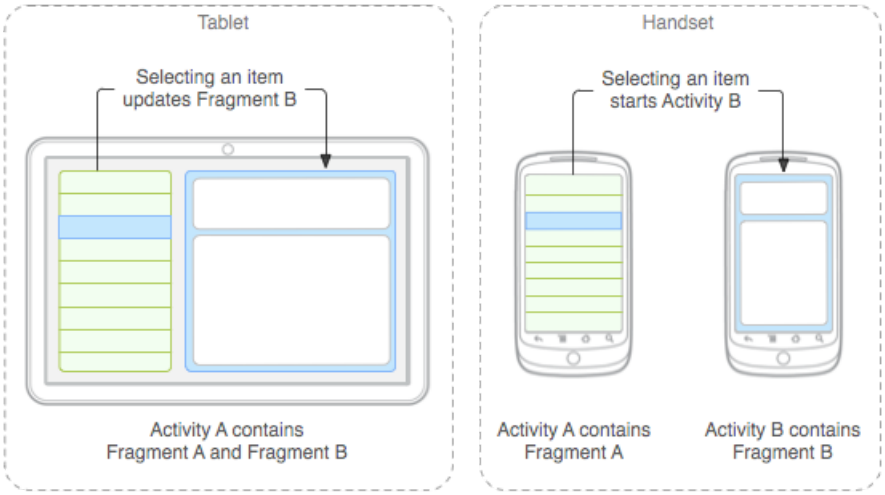
Metoda `onDestroy` je volána v případě ukončení aktivity nebo při změně její konfigurace, tj. např. při otočení přístroje nebo při rozdělení obrazovky na více aplikačních oken.

3.4.2 'Fragment'

S příchodem tabletů se vyskytla potřeba vytvářet takové aplikace, aby jejich GUI bylo kompatibilní s obrazovkou velkého i malého zařízení. Kompatibilní ve smyslu, aby při zobrazení na velkém zažízení nebyla většina obrazovky nevyužita a naopak na malém zařízení nebylo uživatelské rozhraní nečitelné. Aby nebylo potřeba nastavovat složitou hierarchii jednotlivých součástí grafického rozhraní, byla do Androidu zavedena nová komponenta - fragment. Fragment je hostován aktivitou a jeho životní

cyklus je přímo závislý na jejím. Na fragment se dá pohlížet jako na subaktivitu, jejíž instance mohou být použity v různých aktivitách. Rozdělením GUI na fragmenty je možné vzhled aplikace dynamicky upravovat za běhu.

Myšlenka fragmentů je patrná z obrázku 3.5. Např. v aplikaci pro příjem elektronické pošty jsou na tabletu vedle sebe na jedné obrazovce dva fragmenty zobrazující zároveň seznam přijaté pošty a text zvolené zprávy. Na telefonu je pak každý z těchto fragmentů na různé obrazovce.



Obrázek 3.5: Příklad fragmentů na telefonu a tabletu [34]

Na obrázku 3.5 je v případě telefonu každý z fragmentů hostovaný v jiné aktivitě, nicméně vícero fragmentů, které se nezobrazují na stejné obrazovce, lze hostovat i v aktivitě jediné. Příkladem může být aktivita s přepínacími panely (viz v aplikaci vyvíjené v rámci diplomové práce, kap. 5).

Díky hostování aktivitou může instance fragmentu jednoduše komunikovat s instancí aktivity a naopak. Aktivita může např. přijímat informace o poloze a následně je sdílet fragmentu, který polohu zobrazí na mapě. Příkladem může opět být aplikace vyvíjená v rámci diplomové práce a její hlavní aktivita obsahující 2 fragmenty, kterým sdílí informace o družicích (kap. 5).

Životní cyklus fragmentu je velmi podobný životnímu cyklu aktivity. V případě, že se aktivita dostane do stavu 'Paused', se do stejného stavu dostanou všechny fragmenty v ní hostované; v případě, že je aktivita zlikvidována, jsou zlikvidovány i její fragmenty atd. Doporučuje se implementovat 3 základní metody životního cyklu fragmentu - `onCreate()`, `onCreateView()` a `onPause()`. První a poslední z metod

fungují obdobně jako u aktivit, druhá z metod je pak systémem volána v případě potřeby prvního vykreslení grafického rozhraní fragmentu.

3.4.3 'Service'

Služba ('Service') je komponenta provádějící dlouhotrvající operace a operace na pozadí. Služba nevyžaduje grafické rozhraní. Může být spuštěna některou z jiných komponent a běží i v případě, že uživatel přejde k jiné aplikaci. Navíc některá z jiných komponent (i více než jedna) může být se službou 'svázaná', tj. může s ní interagovat a sdílet s ní data. Příkladem služby je hudební přehrávač, kdy hudba hraje i v případě, že uživatel vypne obrazovku. Existují 3 typy služeb:

- **Služba na popředí ('Foreground service')** - tento typ služby provádí operaci, jejíž průběh je zobrazován uživateli pomocí notifikace na horní liště zařízení (služba na popředí musí notifikaci zobrazovat). Např. může být tato služba využita pro zmíněný hudební přehrávač. Tato služba může být ukončena zavoláním příslušné metody z některé jiné komponenty nebo alternativně po ukončení její úlohy.

- **Služba na pozadí ('Background service')** - tato služba provádí operace, u kterých není potřeba, aby byly uživatelem viditelné. Např. může být taková služba zodpovědná za příjem sms zpráv. Tato služba je ukončována stejně jako služba na popředí.

- **Vázaná služba ('Bound service')** - služba se stane vázanou v případě, že se k ní 'naváže' některá z jiných komponent. Vázaná služba poskytuje rozhraní klient-server, díky čemuž může navázaná komponenta se službou komunikovat, posílat jí požadavky a získávat informace nazpět. Jak již bylo řečeno, k službě může být připojeno více komponent. S odpojením poslední se služba ukončí.

Od API 26 (Android 8.0) je ukládáno omezení na služby na pozadí pokud aplikace nemá 'fokus' (tj. aktivita není na popředí, je překryta aktivitou jiné aplikace nebo telefon má zhaslou obrazovku). Aby taková služba mohla být plně aktivní, je třeba jí implementovat např. jako službu na popředí (existují další způsoby v závislosti na operaci, kterou má služba provádět; zde je zmíněn ten, který je využit v autorské aplikaci).

Pro komunikaci mezi službou a jinou komponentou nemusí být služba nutně vázaná. Je to např. v případě, kdy komponenta předává službě parametry jen při spuštění a poté od služby pouze přijímá data. K tomu může služba využít tzv. vysílání zachytitelné přijímačem registrovaným v příslušné komponentě. Tento proces je více popsán v následující kapitole 3.4.4.

3.4.4 'Broadcast Receiver'

Aplikace může posílat nebo přijímat zprávy od jiných aplikací, systému nebo interně mezi komponentami. Model vysílání funguje na principu 'Publish-Subscribe'. Zpráva je vyslaná v případě, že proběhne zájmová událost. Např. systém vysílá takové zprávy při systémových událostech, jako je start systému nebo začátek nabíjení zařízení. Dalším příkladem může být příjem polohy službou a její vysílání aktivitám, které s polohou nějakým způsobem pracují. Pro příjem je třeba, aby aplikace (komponenta) zaregistrovala příjem příslušného vysílání. V případě vyslání zprávy systém automaticky upozorní aplikace (komponenty), které mají dané vysílání registrované.

3.4.5 'Content provider'

Tato komponenta umožňuje aplikaci spravovat přístup k vlastním uloženým datům nebo k datům uloženým jinou aplikací a poskytuje způsob jak sdílet data mezi aplikacemi.

3.5 Určování polohy

Existují 2 hlavní strategie, jak získávat aktuální polohu. První z nich využívá API z balíčku `android.location`. Druhá pak využívá 'Google Location Services API', které jsou součástí 'Google Play Services', což jsou služby zabalené do aplikace, které slouží k aktualizaci aplikací Google a aplikací z Google Play a poskytují funkcionality jako je např. synchronizace Google účtu nebo ověřování pro služby Google [35].

Od API 26 (Android 8.0) je omezeno získávání polohy pokud aplikace neběží na popředí; poloha je počítána s dlouhými intervaly a se sníženou přesností. V případě, že je třeba polohu získávat přesněji a častěji, lze např. zvolit podobný postup jako u komponenty aplikace 'service' - služba (kap. 3.4.3), totiž implementace získávání polohy ve službě, která běží na popředí [36].

V autorské aplikaci je pro zjišťování polohy využit balíček `android.location`, který je popsán v následující podkapitole.

3.5.1 Balíček android.location

Při implementaci získávání polohy je třeba nejprve určit, jak přesná poloha a jak často má být počítána nebo zdali má být zjistitelná i uvnitř budovy. V závislosti na potřebách aplikace je pak volen příslušný 'poskytovatel polohy', což můžou být buď družice GNSS (`GPS_PROVIDER`), vysílače a Wi-Fi (`NETWORK_PROVIDER`), pasivní poskytovatel (`PASSIVE_PROVIDER`) nebo jejich kombinace. Poslední z vyjmenovaných je speciálním typem poskytovatele, který využívá polohu zjištěnou jinou aplikací. Aby mohla být počítána poloha, je třeba dle kap. 3.2 explicitně definovat povolení v manifestu aplikace. Při využití `GPS_PROVIDER` a `PASSIVE_PROVIDER` se jedná o `ACCESS_FINE_LOCATION`, u `NETWORK_PROVIDER` je to `ACCESS_COARSE_LOCATION`. V případě kombinace poskytovatelů postačí první z vyjmenovaných. Od API 21 (Android 5.0) je dále v manifestu třeba definovat využívanou hardwarovou komponentu - `hardware.location.gps` [37].

Základní třídou, která poskytuje přístup k polohovým službám systému, je třída `LocationManager`. Instanci objektu této třídy lze získat za využití metody `getSystemService(Context.LOCATION_SERVICE`. Pro získávání polohy je třeba požadavek zaregistrovat zavoláním metody `locationManager.requestLocationUpdates(`*`String provider, long minTime, float minDistance, LocationListener listener`*`)`, kde `locationManager` je instance třídy `LocationManager`. Existuje několik přetížení této metody. Zde jsou uvedeny ty, které jsou využity v autorské aplikaci. Čtyři argumenty metody jsou:

- *provider* - poskytovatel polohy (viz předchozí odstavec)
- *minTime* - minimální časový interval mezi aktualizacemi polohy v milisekundách
- *minDistance* - minimální vzdálenost mezi aktualizacemi polohy v metrech
- *listener* - rozhraní, jež naslouchá aktualizaci polohy

`LocationListener` je rozhraní sloužící k příjmu notifikací od `LocationManager` v případě změny polohy. Implementuje několik metod, kde jednou z nich je `onLocationChanged`, jež je volána právě při změně polohy [38].

Třídou pro reprezentaci polohy je třída `Location`, jejíž instance obsahuje informace o zeměpisné šířce a délce, nadmořské výšce, aktuálním čase, přesnosti atd. U všech instancí této třídy generovaných pomocí `LocationManager` je garantované,

že obsahují informace o zeměpisných souřadnicích a času (aktuální čas ve formátu UTC a uběhlý čas od nastartování systému) [39].

Při registrování požadavku na polohu je ještě třeba zkontrolovat příslušná oprávnění. Výhodou IDE Android Studia je, že samo upozorňuje např. právě na chybějící kontrolu oprávnění a dokonce samo doplní příslušnou programovou konstrukci [32].

3.5.2 Nový přístup

Modul zodpovědný za příjem a zpracování dat k určování polohy se chová jako 'černá skříňka', tedy neposkytuje metody ani data potřebná k výpočtům. S příchodem API 24 (Android 7.0) v roce 2016 se tento přístup změnil. Nově lze přímo získávat surová měření přijatá z GNSS družic, díky čemuž je možné provádět vlastní analýzy satelitních dat, implementovat vlastní algoritmy pro výpočet polohy a snažit se dosahovat vyšších přesností, což bylo doposud omezeno pro oblast profesionálnějších GNSS přijímačů. Toto téma je dále více rozebráno v kapitole 4.

4 Surová GNSS měření

Počínaje verzí OS Android 7.0 je API z balíčku `android.location` (kap. 3.5.1) obohaceno o nové vlastnosti a funkce, které rozšiřují možnosti v oblasti zjišťování polohy. Hlavními novými třídami jsou:

- `GNSS Clock`, která obsahuje čas přijímače (pro výpočet pseudovzdáleností) a chyby hodin přijímače,
- `GNSS Navigation Message`, která obsahuje části navigační zprávy (všechny GNSS systémy) a její stav,
- `GNSS Measurement`, která obsahuje čas příjmu (pro výpočet pseudovzdáleností), kód a fázi.

Všechna tato data pocházejí z GNSS čipu, nicméně přímý přístup k čipu umožněn není. Využití nového API není tak přímočaré, jak se na první pohled může zdát. Prvním důvodem je, že ne všechna zařízení, která mají OS Android 7.0 a vyšší toto API podporují (o dostupných zařízeních, jež nové API podporují, pojednává kapitola 4.1). Surová data nejsou ukládána ve standardních formátech (ukázka formátu je na obrázku 4.1). Dále např. přijatá data neobsahují přímo pseudovzdálenosti a je třeba je filtrovat od neuspokojivých čtení.

V této kapitole je popsáno nové API včetně rešerše zařízení, která je může využívat a dále jsou zde stručně představeny některé stávající aplikace, které nové API využívají. V této kapitole je čerpáno především z [11].

4.1 Dostupná zařízení

Jak již bylo zmiňováno, aby mělo zařízení umožněno přístup k surovým GNSS měřením, je třeba, aby disponovalo operačním systémem alespoň Android 7.0 (Nougat). Většina zařízení s tímto OS (a vyšší verzí) vyrobených v roce 2016 a později surová data poskytuje. Nicméně není zárukou, že každé takové zařízení má přístup ke všem datům, která je možné získat.

Na webových stránkách vývoje pro Android [40] je uvedeno, která zařízení mají jaké vlastnosti související se získáváním surových dat. Nejedná se o definitivní výčet, tedy pro zjištění, zdali daný telefon skutečně podporuje vše uvedené (příp. co podporuje neuvedený telefon), je lepší konzultace přímo s výrobcem. Seznam zařízení je

průběžně aktualizován. Tabulka 4.1 uvádí zkrácenou verzi výčtu z webu. Jsou zde uvedeny telefony použité při testování autorské aplikace.

Model	Verze OS	Nav. zpráva	Fáze	L5	GNSS
Xiaomi Mi 8	9.0	Ano	Ano	Ano	GPS, GLO, GAL, BDS
Huawei P20	9.0	Ano	Ano	Ne	GPS, GLO

Tab. 4.1: Mobilní telefony s novým API, zkrácený výčet [40]

V tabulce jsou navzdory údajům na webu pozměněné údaje o operačním systému, v době psaní diplomové práce obě zařízení disponují verzí OS Android 9.0. Dále bylo experimentálně zjištěno, že i druhý z uvedených telefonů podporuje systém Galileo a BeiDou. Atribut **L5** uvádí podporu příjmu druhé frekvence pro systémy GPS a Galileo (kap. 2.1).

Doposud mobilní telefony tradičně přijímaly signál pouze v jednom frekvenčním pásmu - L1/E1 (GPS, Galileo). Telefon Xiaomi Mi 8 v květnu 2018 přišel jako první s čipem, který podporuje duální frekvenci v pásmech L1/E1 a L5/E5. V době psaní diplomové práce existuje již více modelů se stejnou vlastností, jsou to např. alternativy řady Huawei Mate 20.

4.2 Popis a využití nového API

V této sekci jsou popsána především získávaná observační data a jejich využití v dalších výpočtech. API týkající se navigačních zpráv je zde popsáno pouze stručně, jelikož se jimi autorská aplikace nezabývá.

4.2.1 Observační data

K ilustraci, jaká observační data mohou být zařízením přijímána, poslouží ukázka na obrázku 4.1 s jednou epochou dat zjednodušených na jednu družici. Každá epocha přijatých dat začíná informacemi o hodinách přijímače a pokračuje výčtem observačních dat pro každou viditelnou družici nezávisle na družicovém systému.

```
[ GnssMeasurementsEvent:

GnssClock:
   LeapSecond      = null
   TimeNanos       = 158992000000   TimeUncertaintyNanos       = null
   FullBiasNanos   = -1236957563008277892
   BiasNanos       = 0.0            BiasUncertaintyNanos       = 12.015784320075461
   DriftNanosPerSecond = null       DriftUncertaintyNanosPerSecond = null
   HardwareClockDiscontinuityCount = 0

GnssMeasurement:
   Svid                          = 2
   ConstellationType             = 1
   TimeOffsetNanos               = 0.0
   State                         = CodeLock|BitSync|TowDecoded|TowKnown
   ReceivedSvTimeNanos           = 141721925294311
   ReceivedSvTimeUncertaintyNanos = 19
   Cn0DbHz                       = 34.26625061035156
   PseudorangeRateMetersPerSecond = 85.26081646442424
   PseudorangeRateUncertaintyMetersPerSecond = 0.0035162782296538353
   AccumulatedDeltaRangeState    = Valid
   AccumulatedDeltaRangeMeters   = 930.3621861563192
   AccumulatedDeltaRangeUncertaintyMeters = 0.0017581391148269176
   CarrierFrequencyHz            = 1.57542003E9
   MultipathIndicator            = Unknown
   SnrInDb                       = null
   AgcLevelDb                    = 49.046180725097656
]
```

Obrázek 4.1: Ukázka formátu surových observačních dat (zdroj: vlastní)

Nové API je součástí balíčku `android.location` a podobně jako v případě získávání polohy z GNSS čipu je třeba i zde požadavek registrovat pomocí `LocationManager` (viz kap. 3.5.1). K tomu slouží metoda `locationManager.registerGnssMeasurementsCallback(myGnssMeas)`. Argument metody je instance třídy `GnssMeasurementEvent.Callback`, do níž jsou 'doručována' přijatá data a jež je vnořenou třídou třídy `GnssMeasurementEvent`, která slouží jako kontejner pro ukládání surových měření. Při vytváření objektu třídy `GnssMeasurementEvent.Callback` je třeba přetížit metodu `onGnssMeasurementsReceived(GnssMeasurementEvent eventArgs)`, která je volána v případě příjmu měření a ukládá data do zmíněného kontejneru. Z něho lze pomocí metody `eventArgs.getClock()` získat data hodin přijímače jako instanci třídy `GnssClock`. Podobně za využití metody `eventArgs.getMeasurements` lze dostat kolekci měření z družic v rámci jedné epochy. Tato měření jsou typu `GnssMeasurement`.

V následující tabulce 4.2 jsou uvedeny použité parametry observace a jejich význam. Každý z parametrů lze získat zavoláním metody `get*()`, kde za `*` je dosazen název atributu. Dále se lze u některých atributů dotázat na jejich existenci zavoláním metody `has*()` opět s dosazením názvu atributu.

Třída GnssClock	
Atribut	**Význam**
LeapSecond $[s]$	Rozdíl mezi časy GPS a UTC
TimeNanos $[ns]$	Čas hodin přijímače
FullBiasNanos $[ns]$	Rozdíl mezi první GPS epochou o půlnoci 6. 1. 1980 a časem hodin přijímače
BiasNanos $[ns]$	Subnanosekundový bias hodin přijímače
HardwareClockDiscontinuityCount	Počet diskontinuit hodin přijímače, důležité pro parametr FullBiasNanos
Třída GnssMeasurement	
Atribut	**Význam**
Svid	ID družice
ConstellationType	Typ systému GNSS
State	Validita observace
ReceivedSvTimeNanos $[ns]$	Čas hodin družice
TimeOffsetNanos $[ns]$	Časový posun, ve kterém bylo měření přijato
Cn0DbHz $[dB - Hz]$	Síla signálu
AccumulatedDeltaRangeMeters $[m]$	Ambiguitní fázové měření
AccumulatedDeltaRangeState	Validita fázového měření
CarrierFrequencyHz $[Hz]$	Frekvence nosné vlny

Tab. 4.2: Parametry observace [11]

Cílem následujících podkapitol je stručné představení jednotlivých parametrů a výpočtů důležitých pro sestavení observační zprávy, jež je také výstupem autorské aplikace (viz 2.4.2).

Validace observací

Pro kontrolu validity observace slouží parametr *State*. Ten může obsahovat několik hodnot najednou. Chtěnými stavy pro GPS a BeiDou jsou *STATE_CODE_LOCK* a *STATE_TOW_DECODED*, pro Galileo *STATE_CODE_LOCK* a *STATE_GAL_E1C_2ND_CODE_LOCK* a pro GLONASS je to *STATE_CODE_LOCK* a *STATE_GLO_TOD_DECODED*.

Výpočet GPS času

Nové API neposkytuje přímo GPS čas, nicméně ho lze vypočítat pomocí údajů o hodinách přijímače a rozdílem mezi nimi a první epochou GPS (půlnoc 6. 1. 1980):

$$GpsTime = TimeNanos - (FullBiasNanos + BiasNanos)\ [ns]. \tag{4.1}$$

V případě, že zařízení odhadlo čas pomocí jiného systému než GPS, je třeba od *GpsTime* (4.1) odečíst rozdíl v referenčích časech těchto systémů (viz kap. 2.3.1).

Hodnoty *FullBiasNanos* a *BiasNanos* vstupují do výpočtu ve všech epochách nezměněné od posledního zafixování času, tj. od chvíle prvního určení platné pseudovzdálenosti a od chvíle, kdy nebyla monitorována diskontinuita hodin přijímače (nezměněný parametr *HardwareClockDiscontinuityClock*).

Určení pseudovzdálenosti

Konstrukce pseudovzdálenosti ϱ je založena na rozdílu času vysílání a příjmu signálu:

$$\varrho = \frac{t_{Rx} - t_{Tx}}{1E9} \cdot c, \tag{4.2}$$

kde t_{Tx} je čas vysílání, t_{Rx} čas příjmu a c je rychlost světla ve vakuu. Hodnota t_{Tx} odpovídá přijatému parametru *ReceivedSvTimeNanos*. Postup pro určení t_{Rx} se liší v závislosti na družicovém systému. Společně pro všechny systémy je nejprve za využití rovnice 4.1 vypočteno:

$$t_{Rx_{GNSS}} = GpsTime + TimeNanos\ [ns]. \tag{4.3}$$

Pro jednotlivé systémy pak:

- GPS a Galileo se statusem *STATE_ TOW_ DECODED*:

$$t_{Rx} = mod(t_{Rx_{GNSS}}, NumberNanoSecondsWeek)\ [ns], \tag{4.4}$$

  kde *NumberNanoSecondsWeek* je týden v nanosekundách.

- BeiDou se statusem *STATE_ TOW_ DECODED*:

$$t_{Rx} = mod(t_{Rx_{GNSS}}, NumberNanoSecondsWeek) + 14s\ [ns]. \tag{4.5}$$

- Galileo se statusem *STATE_GAL_E1C_2ND_CODE_LOCK*:

$$t_{Rx} = mod(t_{Rx_{GNSS}}, NumberNanoSeconds100Milli)\,[ns], \tag{4.6}$$

kde *NumberNanoSeconds100Milli* je 100 milisekund v nanosekundách.

- GLONASS se statusem *STATE_GLO_TOD_DECODED*:

$$t_{Rx} = mod(t_{Rx_{GNSS}}, NumberNanoSecondsDay) + 3h - LeapSecond\,[ns], \tag{4.7}$$

kde *NumberNanoSecondsDay* je den v nanosekundách a *LeapSecond* je aktuální časový skok mezi časy UTC a GPS. Při testování měření surových dat byla druhá z hodnot vždy vracena jako `null`, tj. je třeba jí explicitně v aplikaci definovat. V době psaní diplomové práce byl skok mezi časy 18 sekund.

Fázové měření

Fázové měření je obsaženo v parametru *AccumulatedDeltaRangeMetres* v metrech. Pro přepočet na cykly:

$$phaseCycles = \frac{AccumulatedDeltaRangeMetres}{\lambda}, \tag{4.8}$$

$$\lambda = \frac{c}{CarrierFrequencyHz}, \tag{4.9}$$

kde λ je vlnová délka nosné vlny a c rychlost světla ve vakuu. V případě jednofrekvenčního přijímače je hodnota *CarrierFrequencyHz* vracena vždy jako `null`. Frekvence nosných vln je pak nutné podobně jako u parametru *LeapSecond* explicitně definovat a volit na základě informace o typu družicového systému.

Vypočtená hodnota *phaseCycles* určuje počet cyklů mezi epochami. Platnost a stav fázového měření je možné zjistit pomocí parametru *AccumulatedDeltaRangeState*. Tyto stavy jsou následující:

Status	Hodnota	Význam
ADR_STATE_CYCLE_SLIP	4	Fázový skok
ADR_STATE_RESET	2	Detekován reset
ADR_STATE_VALID	1	Platný stav
ADR_STATE_UNKNOWN	0	Neplatný nebo neznámý stav

Tab. 4.3: Hodnoty *AccumulatedDeltaRangeState* [11]

Systém GNSS a číslo satelitu

Pro určení GNSS systému slouží číselný parametr *ConstellationType* a pro číslo satelitu pak *Svid*. Následující tabulka 4.4 obsahuje seznam GNSS systémů a hodnoty uvedených parametrů:

Systém	*ConstellationType*	*Svid*
GPS	1	1-32
SBAS	2	120-151, 183-192
GLONASS	3	1-24 (93-106)
QZSS	4	193-200
BDS	5	1-37
Galileo	6	1-36
Neznámý	9	-

Tab. 4.4: Hodnoty *ConstellationType* a *Svid* [11]

U systému GLONASS je v parametru *Svid* uvedeno číslo družice nebo číslo frekvenčního kanálu (od -7 do 6) navýšené o 100. Druhý případ vznikne, pokud přijímač nedokáže rozhodnout, která z družic vysílajících na stejné frekvenci je právě ta observovaná (viz kapitola 2.1.2).

4.2.2 Navigační zprávy

Požadavek pro příjem navigační zprávy je podobně jako u observací nutno registrovat pomocí metody `locationManager.registerGnssNavigationMessageCallback(myGnssNav)`, kde `myGnssNav` je instancí třídy `GnssNavigationMessage.Callback`. Postup k získání navigační zprávy je shodný s postupem k získání observací popsaným v kap. 4.2.1.

4.3 Existující možnosti získání surových měření

V této sekci jsou krátce představeny některé z dostupných mobilních aplikací, jež využívají nové API pro sběr surových GNSS měření.

4.3.1 GNSS Logger

GNSS Logger je otevřená aplikace vyvinutá přímo společností Google. Aplikace demonstruje základní implementaci sběru surových GNSS měření. Ukládá data tak,

jak byla přijata s minimální úpravou pro zmenšení velikosti záznamu. Aplikace dále např. obsahuje výpočet polohy, mapové okno s aktuální polohou nebo graf se sílou signálu [40].

4.3.2 GNSS Compare

Otevřená aplikace GNSS Compare v základu počítá polohu telefonu za využití surových GNSS měření. Dále umožňuje uživateli volit výpočet polohy buď pouze z družic systému Galileo resp. GPS nebo pomocí jejich kombinace. V aplikaci je umožněno logování surových měření do souboru ve stejném formátu jako je ukládá aplikace GNSS Logger [41].

4.3.3 Geo++ RINEX Logger

Aplikace Geo++ RINEX Logger je aplikace vyvíjená německou společností Geo++. Ukládá surová GNSS měření ve formátu RINEX [42]. Výstupy z této aplikace byly v rámci diplomové práce zpracovávány a porovnávány s výsledky autorské aplikace GNSS Agent.

5 Aplikace GNSS Agent

Tato část textu se zabývá především popisem implementace autorské aplikace GNNS Agent, její funkcionality a jejích výstupů. Dále jsou zde stručně zmíněné některé již existující aplikace využívající surová GNSS měření.

Obrázek 5.1: Ikona aplikace GNSS Agent (zdroj: David Micheletti)

Důvody k vytvoření vlastní aplikace, jež se zabývá sběrem surových dat, byly následující:

- Testování schopnosti příjmu GNSS dat a jejich kvality pomocí mobilních zařízení jakožto nízkonákladových přijímačů.
- Zhodnocení náročnosti implementace aplikace pro Android a použitelnosti (a srozumitelnosti) nového API.

5.1 Funkcionality aplikace

Aplikace GNNS Agent má následující funkce:

- příjem surových GNSS měření z družic a jejich zpracování do čitelné formy,
- získávání polohy z čipsetu a její zápis do čitelné formy,
- indikátor počtu viditelných družic různých systémů,
- indikátor počtu využitelných družic pro určování polohy, jejich zařazení do příslušného systému a přiřazení příslušné frekvence,
- dynamický graf ukazující sílu signálu využitelných družic,
- mapa zobrazující aktuální polohu uživatele,
- prohlížeč zpracovaných souborů.

5.2 Výstupy z aplikace

Výstupy z aplikace se v zařízení ukládají do složky `Documents/GNSSAgent`. Jména souborů se odvíjejí od času a data počátku observace, např. `log_2019_04_17_17_-41_31_*.txt` je soubor, jež byl vytvořen 17. dubna 2019 v 17:41:31. Za symbol `*` je dosazován typ souboru:

- Zpracované sekundové observace ve formátu BNC (kap. 2.4.2) pro systémy GPS, Galileo, GLONASS a BeiDou viz ukázka na následujícím obrázku. Do souboru se zapisují pouze platné observace (využitelné pro další zpracování). Do názvu souboru s tímto záznamem je vložen řetězec `obs`. V ukázce je prezentována část observovaných družic v jedné epoše. Každá epocha začíná časovou značkou (GPS týden a sekunda v týdnu). Důležitými hodnotami jsou:
  - určení systému a číslo družice (2. sloupec),
  - pseudovzdálenost (za kódem `C1C` pro první frekvenci nebo `C5X` pro druhou),
  - počet fázových cyklů s neznámou ambiguitou (za kódem `L1C` pro první frekvenci nebo `L5X` pro druhou),
  - síla signálu C/N_0 (za kódem `S1C` pro první frekvenci nebo `S5X` pro druhou).

  Záznam v ukázce byl pořízen s využitím mobilního telefonu Xiaomi Mi8.

```
> 2049 315653
GEOP G05 C1C 23355655.516   L1C 79456.821    17  D1C -339.071    S1C 41.885
GEOP G16 C1C 23105782.699   L1C -1206320.358 17  D1C 3624.579    S1C 35.230
GEOP G21 C1C 21154506.740   L1C -693280.131  17  D1C 1832.754    S1C 36.841
GEOP G25 C1C 23328113.583   L1C 1099718.391  17  D1C -3085.096   S1C 35.380
    C5X 23328122.877    L5X 1076633.544 0    D5X -2303.820   S5X 36.098
GEOP G26 C1C 21078322.281   L1C -862646.640  20  D1C 2310.711    S1C 36.935
    C5X 21078324.979    L5X -856603.423 0    D5X 1725.772    S5X 24.002
GEOP G29 C1C 21128682.318   L1C 608376.448   17  D1C -1737.417   S1C 41.850
GEOP G31 C1C 22392005.637   L1C 760539.107   17  D1C -2183.402   S1C 43.656
GEOP R06 C1C 23326532.178   L1C -1275306.294 17  D1C 3447.071    S1C 28.879
GEOP R22 C1C 19242922.102   L1C -391834.135  17  D1C 972.563     S1C 32.679
GEOP E25 C1C 24327589.260   L1C 753291.828   17  D1C -2155.140   S1C 38.516
    C5X 24327591.359    L5X 635092.591  0    D5X -1609.410   S5X 37.990
```

Obrázek 5.2: Ukázka záznamu surových observací (zdroj: vlastní)

Alternativně lze observace také ukládat nezpracované v surovém formátu, ve kterém jsou dostupné přímo z API. Tyto observace nejsou nijak upravované ani ověřované. Ukázka neupravených observací je v kap. 4.2.1 na obrázku 4.1.

- Sekundový záznam polohy z čipsetu viz ukázka na následujícím obrázku 5.3. Do názvu souboru s tímto záznamem je vložen řetězec `pos`. V ukázce jsou uvedené 3 po sobě jdoucí epochy. Ukázka je pro přehlednost upravená (prázdné řádky mezi epochami a odsazení). Vypsané údaje v pořadí jsou: *GPS týden, sekunda v GPS týdnu, zeměpisná šířka, zeměpisná délka, výška, udávaná přesnost měření, seznam použitých družic (u GPS a Galileo jsou uvedená i frekvenční pásma)*. Záznam v ukázce byl taktéž pořízen s využitím mobilního telefonu Xiaomi Mi8.

```
2049 315635 14.785619110643086  49.913692087490084  591.2583220158420
     ±4.0 R22 E36_E1 E36_E5 E02_E1 E02_E5 R12 G21_L1 E11_E1 E11_E5
     G26_L1 G26_L5 G29_L1 C23 E25_E1 E25_E5 C25 R21 G31_L1 R13 R11
     G25_L1 G25_L5 G16_L1 E30_E1 E30_E5 C05 G05_L1 R23 R05 C12 C22

2049 315636 14.785618284200750  49.913692904578376  591.2605681324432
     ±4.0 R22 E36_E1 E36_E5 E02_E1 E02_E5 R12 G21_L1 E11_E1 E11_E5
     G26_L1 G26_L5 G29_L1 C23 E25_E1 E25_E5 C25 R21 G31_L1 R13 R11
     G25_L1 G25_L5 G16_L1 E30_E1 E30_E5 C05 G05_L1 R23 R05 C12 C22

2049 315637 14.785618145831222  49.913693492596850  591.2464876768748
     ±4.0 R22 E36_E1 E36_E5 E02_E1 E02_E5 R12 G21_L1 E11_E1 E11_E5
     G26_L1 G26_L5 G29_L1 C23 E25_E1 E25_E5 C25 R21 G31_L1 R13 R11
     G25_L1 G25_L5 G16_L1 E30_E1 E30_E5 C05 G05_L1 R23 R05 C12 C22
```

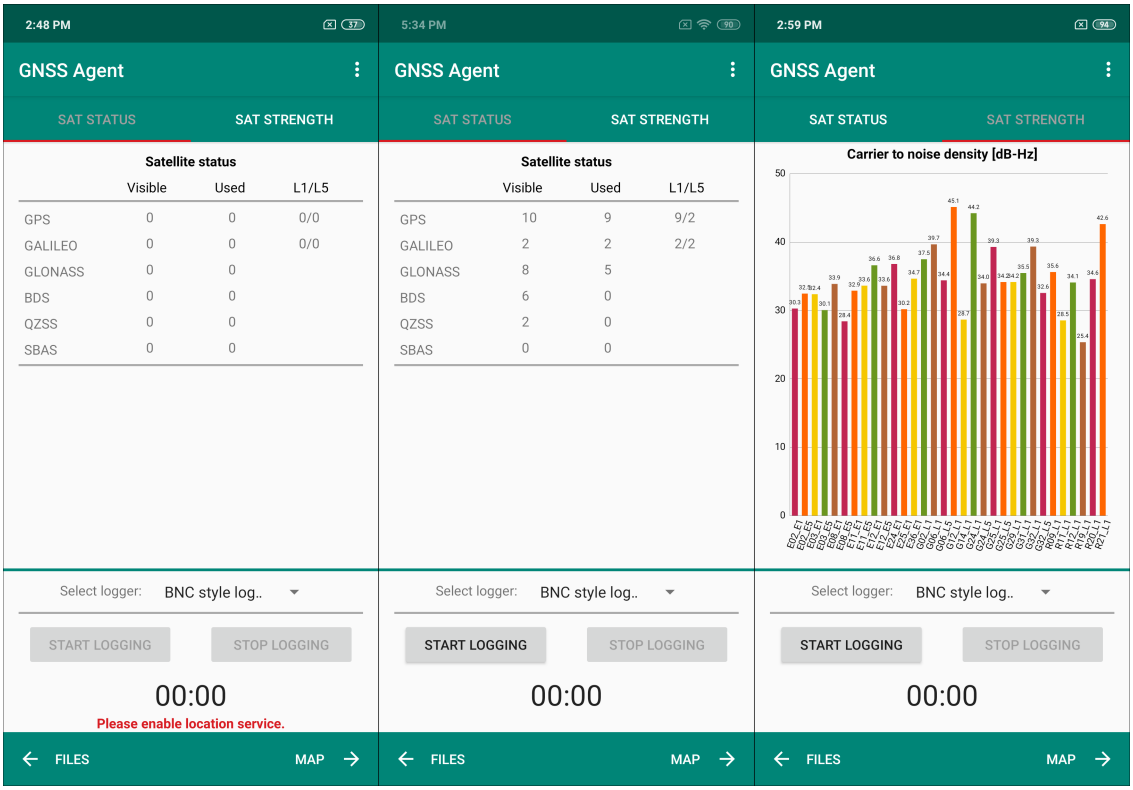
Obrázek 5.3: Ukázka záznamu polohy z čipsetu (zdroj: vlastní)

5.3 Grafické uživatelské rozhraní

Aplikace GNSS Agent se skládá celkem ze 3 hlavních obrazovek (aktivit), kde některé z nich se dělí na další podobrazovky (fragmenty). Všechny obrazovky mají společnou horní a spodní lištu. Ve spodní liště se nacházejí navigační tlačítka sloužící pro přesun mezi obrazovkami. V horní liště je uveden název aplikace a menu se základním nastavením (volba výstupního formátu) a přístupem k systémovým informacím o aplikaci (využití paměti, správce notifikací, oprávnění atp.).

V dalších podkapitolách této sekce jsou představeny jednotlivé obrazovky aplikace.

5.3.1 Hlavní obrazovka



Obrázek 5.4: Hlavní obrazovka aplikace (zdroj: vlastní)

V případě, že v zařízení není aktivované zjišťování polohy, se ve spodní části hlavní obrazovky objeví upozornění (obr. 5.4 vlevo). Hlavní obrazovka aplikace je rozdělena na 2 části.

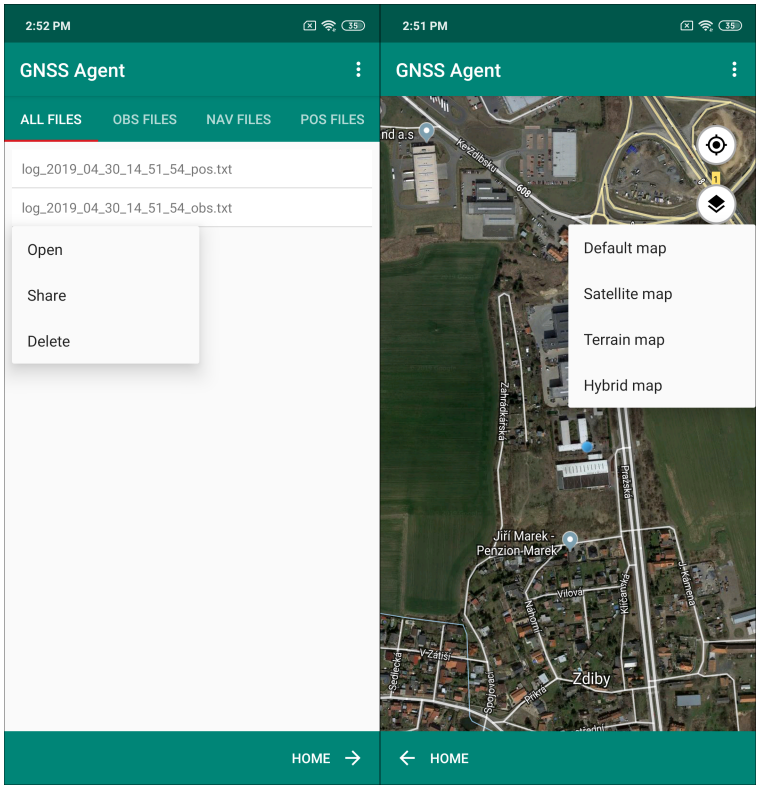
Ve spodní části lze volit typ výstupu pro surové observace. Dále se zde nachází tlačítka ovládající spuštění a vypnutí záznamu. Spouštěcí tlačítko je deaktivované, pokud není v zařízení zaplé zjišťování polohy nebo pokud záznam již běží; tlačítko pro zastavení se naopak aktivuje po spuštění záznamu. Pod tlačítky je pak zobrazen čas uplynulý od spuštění.

Horní část obsahuje 2 podobrazovky, mezi kterými se lze přesouvat přetažením nebo pomocí navigačních tlačítek. Na první podobrazovce (obr. 5.4 vlevo a uprostřed) se zobrazuje aktuální počet viditelných a použitelných družic všech systémů. Pro GPS a Galileo se navíc ukazuje kolik z použitelných družic vysílá v jakém frekvenčním pásmu (pro přehlednost jsou v nadpisu uvedeny pásma pouze pro GPS).

Na druhé podobrazovce (obr. 5.4 vpravo) je zobrazována síla signálu jednotlivých použitelných družic včetně rozřazení do frekvenčních pásem (GPS a Galileo).

5.3.2 Správce souborů a mapové okno

Na dalších 2 obrazovkách se nachází správce souborů a mapové okno.



Obrázek 5.5: Správce souborů a mapové okno (zdroj: vlastní)

Správce souborů

Obrazovka se správcem souborů (obr. 5.5 vlevo) je rozdělená na 4 podobrazovky, mezi kterými lze přecházet přetažením nebo pomocí navigačních tlačítek v horní části. Na každé z nich je zobrazen seznam uložených souborů rozdělen do logických skupin (podle názvu podobrazovky) - všechny soubory, observační soubory, navigační soubory, soubory s pozicí z čipsetu. Získávání navigačních dat není v aplikaci implementováno, proto je vždy příslušný seznam prázdný. Po výběru položky v seznamu se zobrazí menu s akcemi, které se dají se souborem provádět. Při výběru možnosti otevření a sdílení se uživateli ukáže nabídka s dostupnými možnostmi (jiné nainstalované aplikace) na provedení akce.

Mapové okno

Na další obrazovce (obr. 5.5) se nachází mapa (Google maps) se zobrazením aktuální polohy získané z čipsetu. Stisnutím příslušných tlačítek lze přepínat mapové podklady (základní, satelitní, terénní, hybridní - satelitní mapa s bodovými a liniovými prvky a názvy) nebo vystředit mapu a sledovat polohu zařízení (modrá tečka).

5.4 Práce s aplikací

Při prvním spuštění aplikace je uživatel požádán o udělení povolení k přístupu k poloze a do souborů zařízení (za účelem ukládání záznamů). Tato povolení se dají dále spravovat v systémových informacích o aplikaci, jež jsou dostupné buď z nastavení telefonu nebo přímo z aplikace z menu na horní liště.

Aplikace je vyvíjena tak, aby byla co nejintuitivnější, proto v zásadě obsahuje pouze 2 důležitá tlačítka, která jsou potřebná ke spuštění a vypnutí záznamu. Při spuštění záznamu se v horní liště zařízení objeví notifikace informující uživatele o tom, že je aplikace aktivní. Tato notifikace zároveň slouží k uchování běhu aplikace ikdyž není na popředí (o službách na popředí pojednává kap. 3.4.3). Díky tomu lze během záznamu zapínat jiné aplikace nebo vypnout displej zařízení pro šetření energie baterie. Při testování aplikace bylo zjištěno, že i přes uvedené opatření čipset přestává po několika minutách počítat polohu (ze dvou testovaných telefonů se tento úkaz objevil pouze na jednom - Xiaomi Mi8). Pro eliminaci tohoto problému je třeba aplikaci "uzamknout" ve správci běžících aplikací.

Aplikace vyžaduje připojení k internetu pouze pro zobrazení mapy v mapovém okně. Poloha z čipsetu je počítána s využitím družic navigačních systémů.

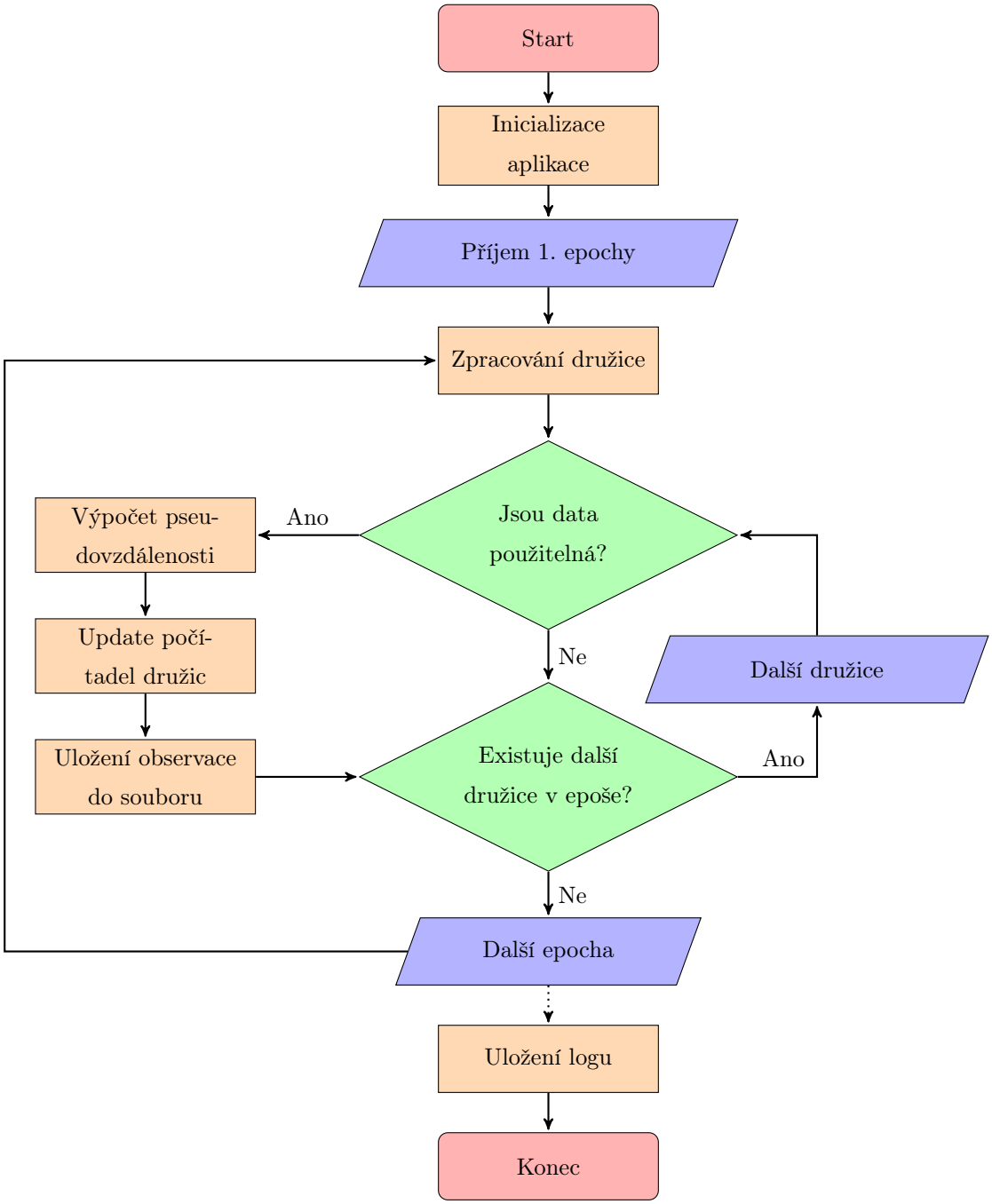
5.5 Implementace

5.5.1 Verze platformy

Před samotnou tvorbou aplikace bylo třeba vybrat minimální verzi OS, pro který má být aplikace dostupná. Vzhledem k API, které GNSS Agent využívá, byla nastavena na Android 7 Nougat (API 24). Z průzkumů trhu mezi březnem 2018 a březnem 2019 by měla být aplikace dostupná cca. pro 58% všech zařízení s OS Android [43].

5.5.2 Kostra aplikace

Základním kamenem aplikace je příjem a základní analýza dat z družic. Další funkcionality aplikace jsou na tomto přímo závislé. Na následujícím vývojovém diagramu (obrázek 5.6) je stručně naznačeno, jakým způsobem aplikace pracuje. Z vývojového diagramu je pro zjednodušení vynechán proces spuštění a zastavení záznamu.



Obrázek 5.6: Kostra aplikace (zdroj: vlastní)

Aplikace je psána v jazyce Java. Třídy zajišťující chod aplikace jsou rozděleny do 3 logických balíčků. Jsou to balíčky `ui`, `service` a `data_processing`. V následujících podkapitolách jsou tyto balíčky popsány.

5.5.3 Balíček `ui`

Balíček `ui` spravuje uživatelské rozhraní; obsahuje třídy, jež stojí za chodem komponent aplikace, které se zobrazují uživateli. Balíček obsahuje další 2 podbalíčky `main` a `files_managing`. Třídy mimo podbalíčky jsou:

- Třída `BaseActivity`
  - Abstraktní třída, která implementuje část uživatelského rozhraní, jež je společné pro všechny aktivity v aplikaci. Tím je aplikační menu na horní liště aplikace. Všechny aktivity dědí z této třídy.
- Třída `MapsActivity`
  - Tato třída implementuje mapové okno a všechny funkcionality s ním spojené. Důležitou metodou této třídy je metoda `setUpBroadcastReceiver`, která inicializuje přijímač dat vysílaných třídou `LogService` z balíčku `service`, podle kterých je aktualizována pozice na mapě za využití další metody `updateLocation`.

Podbalíček `main`

Třídy tohoto balíčku zobrazují uživatelské rozhraní hlavní obrazovky a implementují její funcionality.

- Třída `MainActivity`
  - Jedná se o hlavní aktivitu, v níž je inicializována služba `LogService` z balíčku `service`. Před voláním zmíněné služby kontroluje, zdali jsou udělena všechna potřebná oprávnění. Implementuje metody reagující na uživatelské akce, které začínají a ukončují logování. Přebírá hodnoty důležité pro zpracování dat zadané uživatelem (resp. jednu hodnotu - typ výstupního formátu záznamu surových observací). Tato aktivita dále hostuje fragmenty implementované v následujících třídách.

- Třída `SatStatusFragment`
  - Jedná se o fragment zobrazující aktuální počty observovaných a použitelných družic včetně frekvenčních pásem. Instance třídy je inicializována v hostující aktivitě. Potřebná data získává voláním metody `setSateliteStatus` v hlavní aktivitě.
- Třída `SatSignalStrengthFragment`
  - Jedná se o fragment zobrazující sílu signálu družic. Stejně jako v předchozím případě je tato třída inicializována v hostující aktivitě a data získává voláním metody `updateChart`. Data jsou vykreslována za využití knihovny `MPAndroidChart` dostupné z `https://github.com/PhilJay/MPAndroidChart`.

Podbalíček `files_managing`

V tomto balíčku je hlavní třídou třída `FilesActivity`. Ta hostuje 4 fragmenty implementované v třídách `AllFilesFragment`, `PosFilesFragment`, `ObsFilesFragment` a `NavFilesFragment` (pouze příprava na možnou budoucí implementaci).

5.5.4 Balíček `service`

Hlavní třídou tohoto balíčku je třída `LogService`. Jedná se o službu, jež je primárně zodpovědná za příjem dat, jejich distribuci do ostatních komponent aplikace a za vytváření notifikace udržující aplikaci v chodu i když není na popředí. Důležitými metodami jsou:

- Metoda `processRawMeas`
  - Do metody vstupuje kontejner typu `GnssMeasurementEvent` obsahující přijatá surová data v jedné epoše. V této metodě je vytvářena instance třídy `OneEpoch` z balíčku `data_processing` jež měření zpracovává (popsána dále v podkapitole 5.5.5). Část zpracovaných dat, která se mají ukazovat uživateli, je odtud odesílána do příslušných tříd z balíčku `ui` zavoláním další metody `sendDataToMainActivity`. Zpracovaná data jsou dále sdílena třídám z balíčku `data_processing` zodpovědným za zápis dat do souborů (třídy `BncCoder` a `RawCoder`).

- Metoda `processPosition`
  - Argument typu `Location` vstupující do této metody obsahuje informace o poloze získané z čipsetu v jedné epoše. Odtud je volána metoda pro zápis těchto dat do souboru (metoda `parseData` třídy `ChipsetPositionCoder` z balíčku `data_processing`).
- Metody `sendDataToMainActivity` a `updateMapActivity`
  - Tyto metody jsou zodpovědné za vysílání dat do hlavní resp. mapové aktivity (viz kap. 3.4.4). Na základě těchto dat aktivity aktualizují počty družic a sílu jejich signálu resp. pozici na mapě.

5.5.5 Balíček `data_processing`

V balíčku `data_processing` se nacházejí třídy potřebné ke zpracování surových dat a dat z čipsetu. Obsahuje 2 statické třídy - `Constants`, která pouze obsahuje konstantní hodnoty potřebné pro výpočty (např. rychlost světla, přestupné sekundy mezi časy UTC a TAI atd.) a `TimeConverter` obsahující metody pro převody mezi časy UTC a GPS. Další třídy jsou:

Třída `BaseCoder`

`BaseCoder` je abstraktní třídou, ve které jsou implementované společné metody tříd pro zápis do souboru (vytváření souboru a zápis jedné položky).

Třídy `BncCoder`, `RawCoder` a `ChipsetPositionCoder`

Všechny tyto třídy dědí od třídy `BaseCoder` a navíc implementují metody, ve kterých je formátován výstup podle zvoleného typu záznamu.

Třída `OneEpoch`

Ve třídě `OneEpoch` jsou zpracovávána surová měření jedné epochy. V konstruktoru třídy je vytvářen kontejner pro zpracované observace. Důležitými metodami jsou:

- Metoda `determineConst`
  - V této metodě probíhá rozdělování jednotlivých observací podle typu systému, pod který observovaná družice spadá. Podle typu systému je pro observaci volána příslušná metoda (viz metody v následujícím bodě tohoto výčtu).

- Metody `processGps, processGalileo, processGlonass, processBeidou`
  - V těchto metodách jsou testována kritéria pro platnost observací. V případě, že jsou kritéria splněna je vypočtena pseudovzdálenost a zavolána metoda `processBase`, ve které je každou observací naplňován kontejner typu `OneObs`.
- Metoda `processBase`
  - V této metodě je pro každou observaci naplňován kontejner typu `OneObs`.

Třída `OneObs`

Tato třída slouží jako kontejner pro jednu observaci. Pro každý údaj z observace definuje členskou proměnnou. Důležitou metodou je metoda `toString`, která je volána třídou pro zápis zpracovaných observací do souboru (`BncCoder`). V této metodě je definován formát zápisu jedné observace.

Třída `VisibleUsableSatelites`

Tato třída je kontejnerem pro počty viditelných a použitelných družic a pro počty družic v příslušných frekvenčních pásmech (pro systémy GPS a Galileo).

6 Analýza měření

V této kapitole jsou prezentovány výsledky analýzy dat získaných z měření pomocí autorské aplikace GNSS Agent. Některé výstupy jsou dále porovnávány s výstupy z analýzy dat získaných pomocí aplikace Geo++ RINEX Logger (kap. 4.3.3, dále jen Geo++), GNSS čipsetu a z geodetického přijímače na Geodetické observatoři Pecný (GOPE). K měření byl použit mobilní telefon Xiaomi Mi8 umožňující sběr dvoufrekvenčních pozorování. Cílem této kapitoly je zhodnotit schopnosti testovaného mobilního telefonu a vytvořené autorské aplikace.

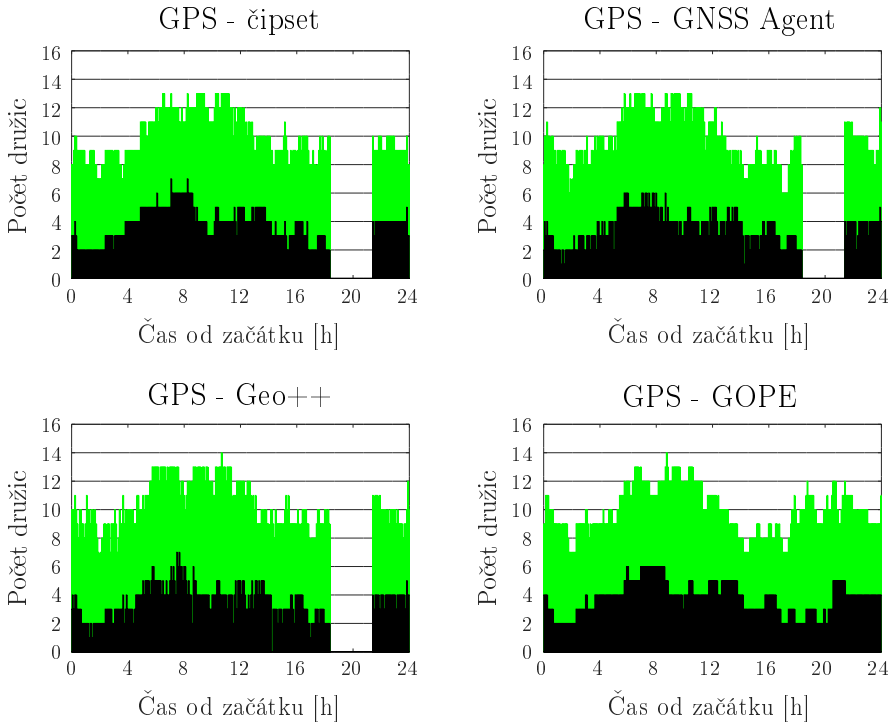
Pro analýzu byly zvoleny vzorky dat sebrané během dubna a května 2019 na GOPE a v Adamově u Českých Budějovic. Mobilní telefon byl vždy umístěn na střechu s co nejlepším výhledem na oblohu, v případě GOPE navíc blízko geodetické antény. Cílem bylo sebrat vždy alespoň 24 hodinové observace. Tento záměr byl ale provázen technickými problémy; mobilní telefon občas sám vypínal příjem dat a zjišťování polohy, dále kvůli aplikaci Geo++ bylo potřeba po celou dobu měření nechat rozsvícenou obrazovku telefonu, což velmi urychluje vybíjení baterie. Pro eliminaci hrozby vybití baterie byl telefon připojen k záložní baterii.

6.1 Kvalita dat

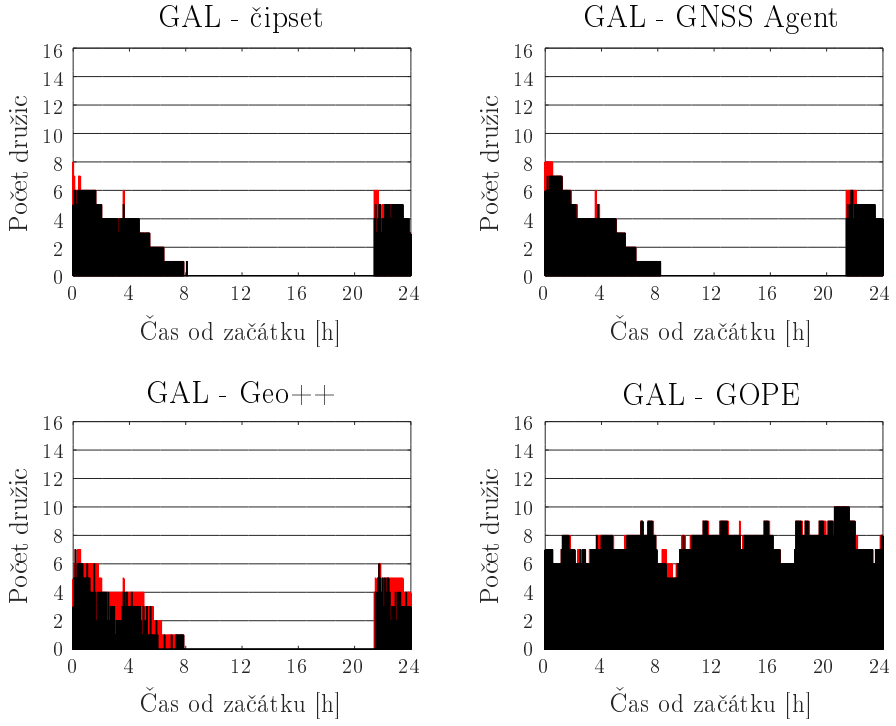
Pro zhodnocení kvality dat byl použit software G-Nut/Anubis. G-Nut/Anubis je nástroj vytvořený za využití knihovny G-Nut vyvíjené na GOPE od roku 2011. Autory jsou J. Douša a P. Václavovic. Anubis byl vyvinut pro hodnocení kvality a kvantity GNSS dat uložených ve formátech RINEX 2 a 3 nebo i v neoficiálním formátu BNC. Software je dostupný ke stažení na webových stránkách GOPE pod licencí GNU GPL [44].

6.1.1 Počet družic

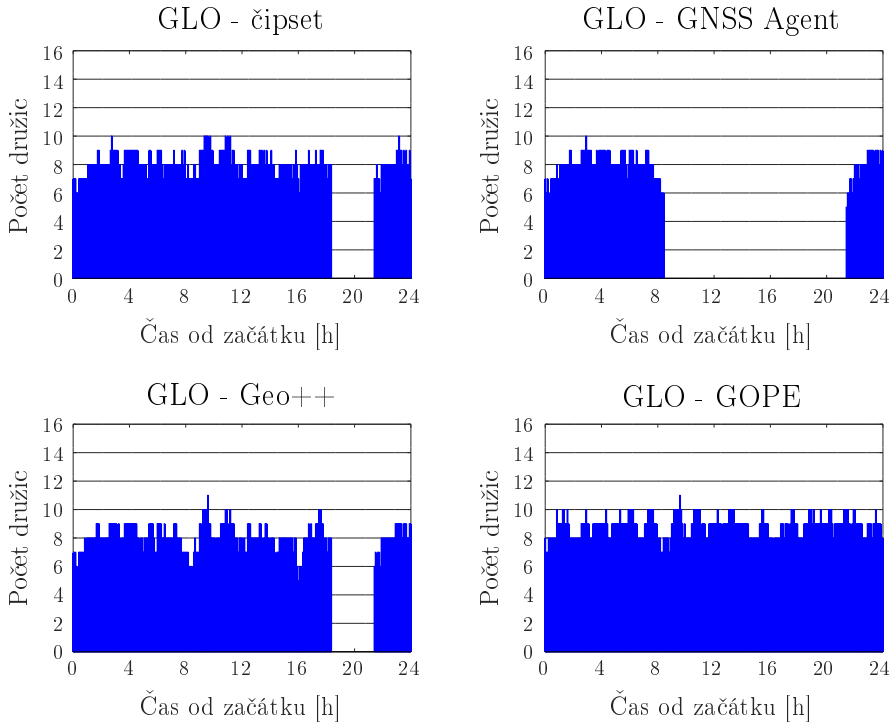
Do grafů na následujících obrázcích 6.1 - 6.4 jsou vynesené počty družic jednotlivých systémů. Data pocházejí z měření 2. května 2019 na GOPE. Na ose x je zobrazen čas od počátku měření, vzorkování je nastaveno na 5 minut. Na ose y je pak zobrazen počet družic. Zelená (GPS), červená (Galileo), modrá (GLONASS) a purpurová (BeiDou) barva indikuje celkový počet družic v dané epoše, černá barva (GPS a Galileo) pak indikuje počet družic s dvoufrekvenčními observacemi.



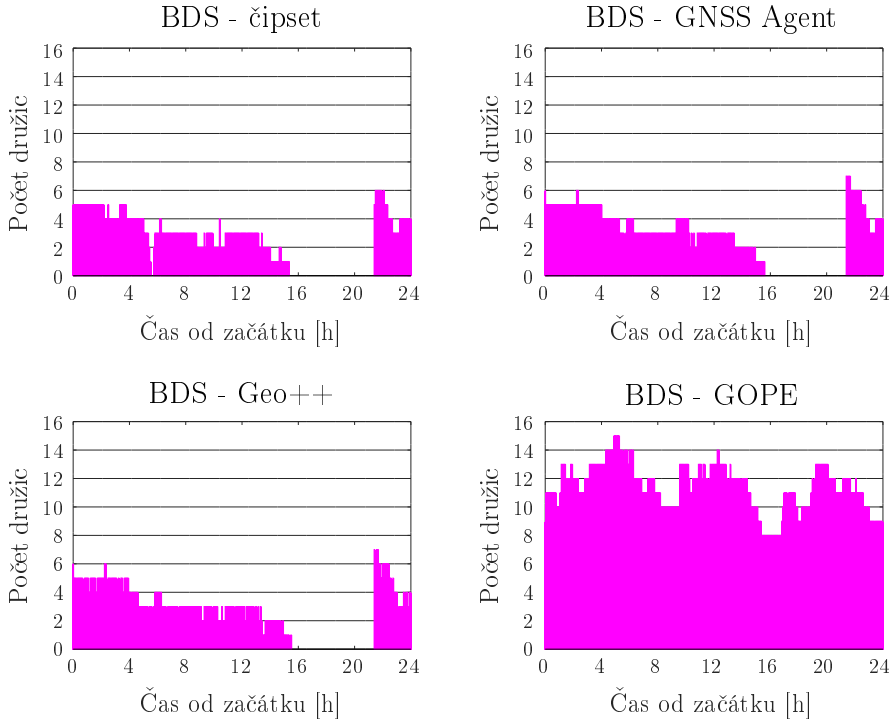
Obrázek 6.1: Počet družic systému GPS 2. 5. 2019 (zdroj: vlastní)



Obrázek 6.2: Počet družic systému Galileo 2. 5. 2019 (zdroj: vlastní)



Obrázek 6.3: Počet družic systému GLONASS 2. 5. 2019 (zdroj: vlastní)



Obrázek 6.4: Počet družic systému BeiDou 2. 5. 2019 (zdroj: vlastní)

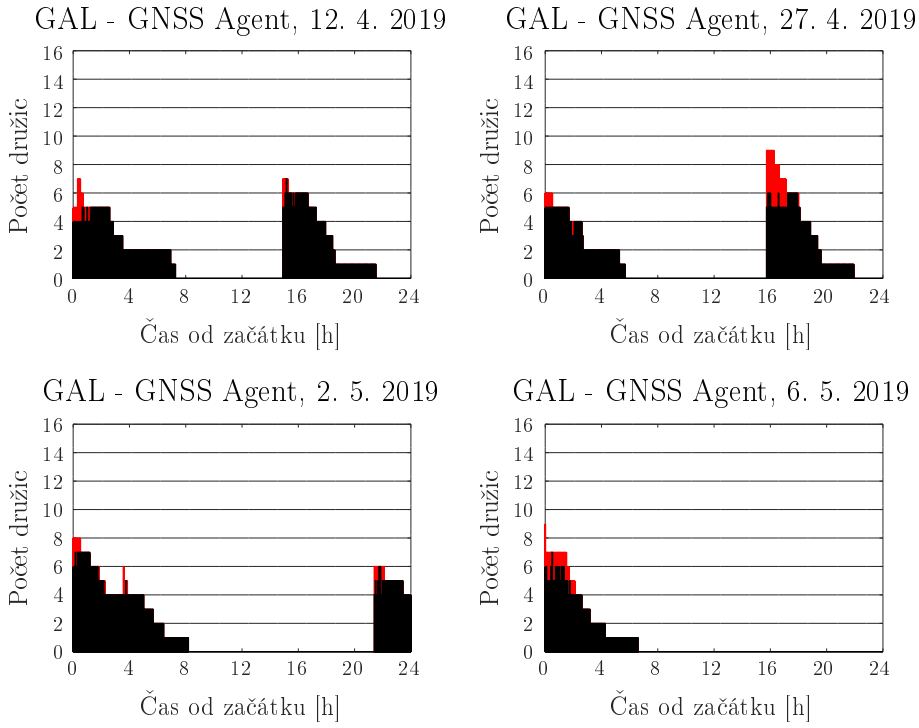
Z uvedených obrázků lze odvodit několik závěrů:

- Zejména z obr. 6.1 pro GPS, jež je pro následující tvrzení nejrelevantnější, je patrné, že po 18 hodině měření přestal telefon přibližně na 3 hodiny přijímat data. Tento úkaz se objevil ve všech třech testovaných aplikacích i v jiných dnech testování. Ze společného výpadku dat u obou aplikací i získání polohy z vlastního čipsetu lze usuzovat, že se jedná o interní problém čipsetu. Nalezená data s experimenty na webu jsou vždy krátkodobé (observace v řádu několika minut), nikde proto nebyl nalezen report uvádějící stejné problémy.

- Aplikace GNSS Agent má zřejmě chybu v implementaci pro pozorování družic systému GLONASS (obr. 6.3), kdy po několika hodinách dochází k náhodnému výpadku. Ani po několikanásobné revizi zdrojového kódu aplikace se tato chyba nepodařila odhalit.

- Až na menší výjimky (a zmíněnou chybu v implementaci GNSS Agent) lze obecně říct, že mobilní telefon ve všech aplikacích zpracovává podobný počet družic každého systému.

- Z grafů týkajících se systému Galileo (obr. 6.2) je patrný souvislý pokles počtu observovaných družic až k nule. Okamžik oživení příjmu družic Galileo koresponduje s koncem okna, ve kterém čip vůbec neposkytoval data. Tento 'reset' tedy zřejmě napomohl i restartovat příjem družic systému Galileo. Situace je shodná pro všechny metody získání dat z čipu. Následující obrázek 6.5 poukazuje na opakování popsaného trendu během čtyř nezávislých dnů. Porovnání je provedeno na datech získaných aplikací GNSS Agent. Při bližším prozkoumání získaných dat bylo zjištěno, že stejně jako u dat z 2. 5. 2019 koresponduje jejich opětovné získání s koncem celkové výpadku měření. Měření ze dne 27. 4. 2019 probíhalo v Adamově u Českých Budějovic, ostatní probíhala na GOPE. Měření ze dne 6. 5. 2019 bylo o několik hodin kratší, přičemž nedošlo k resetu příjmu. Podobný úkaz lze sledovat také u systému BeiDou, kdy počet observovaných družic postupně klesal a vymizel ještě před začátkem celkového výpadku. Příjem BeiDou byl opět restartován ve stejný čas jako u ostatních systémů. Z analýzy vyplývá, že problém také souvisí s vlastním čipem a projevuje se stejně u všech metod získání informací o poloze.

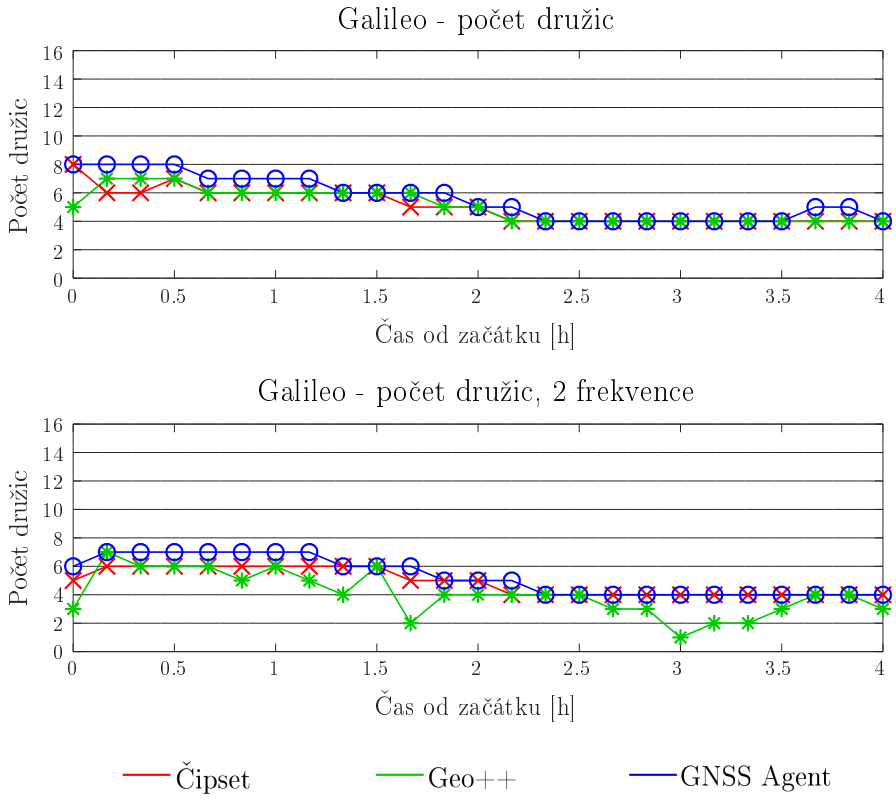


Obrázek 6.5: Trend úbytku družic systému Galileo (zdroj: vlastní)

- V porovnání s geodetickou anténou přijímá mobilní zařízení jen omezený počet družic systémů Galileo a BeiDou, což je pravděpodobně způsobeno problémem přímo v čipu.
- Počet družic systémů GPS a GLONASS zjišťovaný mobilním telefonem je na první pohled srovnatelný s počtem zjištěným geodetickou anténou. Tyto systémy jsou na rozdíl od dvou zbývajících stabilní po celou dobu měření (s výjimkou výpadku).
- Počet družic s dvoufrekvenčními daty v případě GPS je v průměru poměrně nízký (také u měření geodetickou anténou), pohybuje se kolem 4 družic. To je zřejmě dané tím, že na frekvenci v pásmu L5 vysílá dosud pouze 12 nejnovějších družic [2]. Naopak všechna pozorování družic systému Galileo jsou dvoufrekvenční, jelikož jde o družice stejné generace (s výjimkou aplikace Geo++, viz diskuze na konci této podkapitoly).

Při bližším prozkoumání počtu družic z různých aplikací na mobilním telefonu bylo zjištěno, že u systému Galileo se objevují významnější rozdíly. Ostatní systémy až na občasné výjimky mají počty družic stejné. Na následujícím obrázku 6.6 jsou

detailněji porovnány počty družic systému Galileo. Data jsou omezena na první 4 hodiny observací a bez újmy na obecnosti je pro přehlednost zvýšen interval mezi epochami na 10 minut.



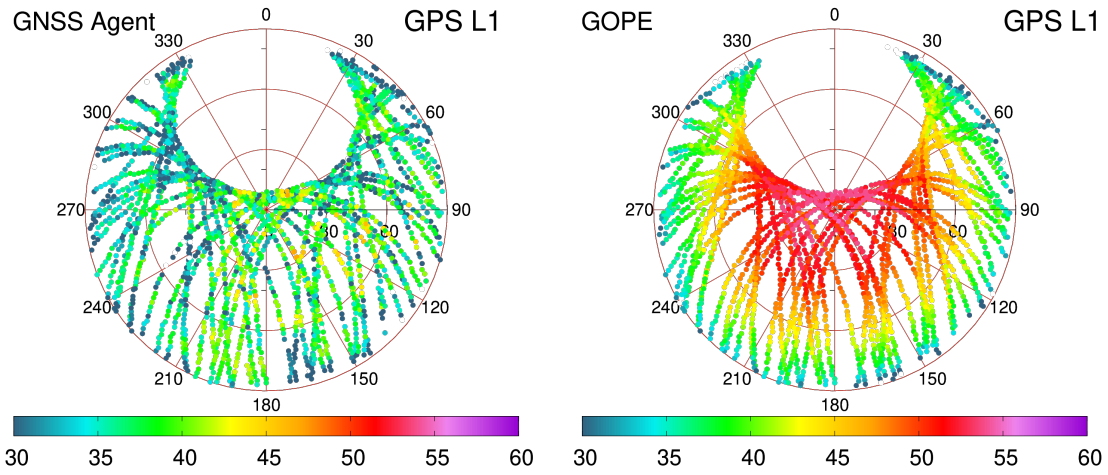
Obrázek 6.6: Rozdíl v počtu družic Galileo (zdroj: vlastní)

Z uvedeného obrázku je patrné, že aplikace Geo++ obsahuje téměř po celou dobu menší počet dvoufrekvenčních pozorování než GNSS Agent a vlastní čipset. Tento rozdíl se objevoval i v průběhu dalších dnů měření. Aplikace GNSS Agent obecně poskytuje data více družic systému Galileo než čipset a Geo++. Rozdíly jsou zřejmě způsobené implementací různých kritérií pro validaci observací (kap. 4.2.1). Ověřit tato tvrzení přímo ze zdrojového kódu aplikace Geo++ nelze, jelikož není volně dostupný.

6.1.2 Sky plot a síla signálu

V grafech na následujícím obrázku 6.7 jsou vynesené vybrané 'sky ploty' se zobrazením poměru signálu a šumu v elevacích a azimutech z jednotlivých družic v průběhu měření. Každý bod v grafu je označen barvou, jež indikuje sílu signálu (kap. 2.3.4).

Měření opět pochází ze dne 2. 5. 2019, je dlouhé 24 hodin a obsahuje výpadek diskutovaný v předchozí podkapitole 6.1.1.



Obrázek 6.7: Sky plot GPS L1, porovnání GNSS Agent a GOPE (zdroj: vlastní)

Na obrázku 6.7 jsou porovnávány signály L1 systému GPS získané aplikací GNSS Agent a geodetickou anténou. Z obou grafů je patrné velmi podobné zaplnění oblohy, ovšem s velkým rozdílem v síle signálu. Zatímco pozorování geodetickou anténou vykazují souvislý nárůst síly signálu od horizontu směrem k zenitu nad přijímačem, pozorování mobilním telefonem jsou velmi proměnlivá po celou dobu měření. Měření mobilním telefonem navíc zdaleka nedosahuje největší síly signálu dostupné při měření geodetickou anténou. Tento efekt je pochopitelný vzhledem ke kvalitě, pořizovací ceně a velikosti antén. Obrázky pro další signály jsou v principu velmi podobné.

V následující tabulce 6.1 jsou uvedené průměrné hodnoty síly signálu v prvních 4 hodinách měření pro různé systémy a měřící zařízení (mobilní aplikace a geodetická anténa).

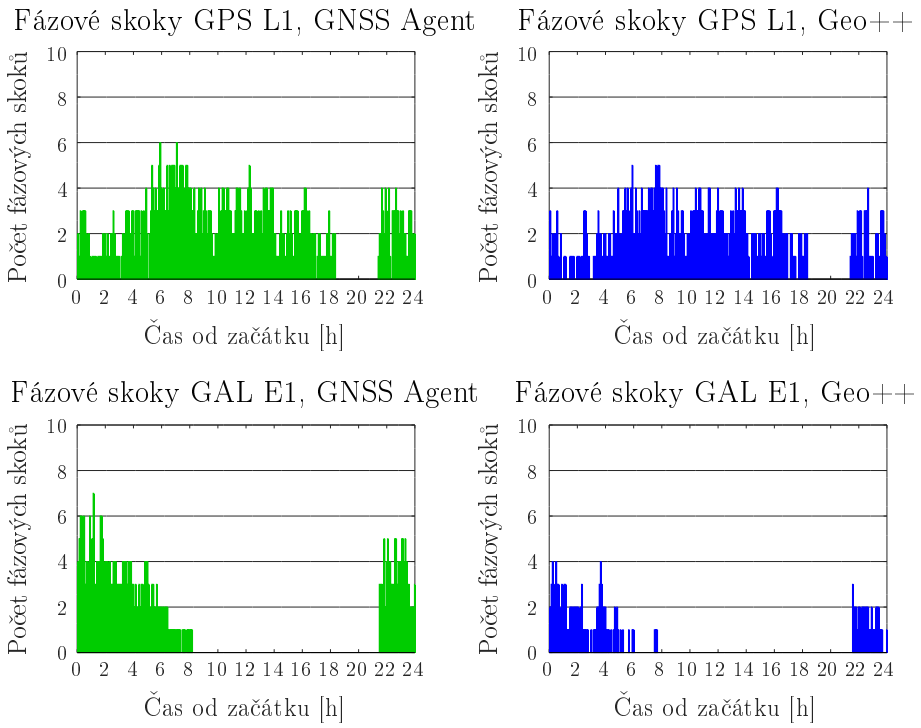
	GPS L1	**GPS L5**	**GAL E1**	**GAL E5**	**GLO**	**BDS**
GOPE	44.4	47.0	44.1	46.2	45.4	42.4
GNSS Agent	34.6	30.7	30.0	30.2	32.6	33.5
Geo++	34.0	30.2	33.5	31.8	32.6	33.9

Tab. 6.1: Průměrné síly signálu v $[dB - Hz]$ (zdroj: vlastní)

Stejně jako z obrázku 6.7 je i zde dle očekávání patrné, že síla signálu přijatého geodetickou anténou je téměř ve všech případech o třetinu vyšší než u signálu přijatého mobilního telefonu. Rozdíly mezi mobilními aplikacemi se objevují především u systémů GPS a Galileo. Aplikace GNSS Agent má vyšší průměr u systému GPS a naopak nižší u Galileo. Každá z aplikací tedy zřejmě lépe filtruje přijatá data jiných systémů. S tvrzením o lepší filtraci družic systému Galileo aplikací Geo++ koresponduje i analýza prováděná v kap. 6.1.1, ze které vyšlo najevo, že u tohoto systému GNSS Agent zpracovává více měření.

6.1.3 Fázové skoky

S kvalitou síly signálu a filtrací družic na základě síly signálu souvisí i počet fázových skoků (přerušení pozorování). Následující obrázek 6.8 ukazuje počty fázových skoků zjištěných při měření ze dne 2. 5. 2019. Jsou zde porovnávány výstupy z aplikací GNSS Agent (na obrázcích vlevo) a Geo++ (na obrázcích vpravo) pro systémy GPS a Galileo. V grafech jsou uvedeny pouze první frekvence, jelikož druhé mají shodný průběh.



Obrázek 6.8: Fázové skoky (zdroj: vlastní)

Z grafů je na první pohled zřejmé, že zejména u systému Galileo mají observace z aplikace Geo++ méně fázových skoků. Množství observací s fázovými skoky včetně procentuálního zastoupení vzhledem k celkovému počtu během prvních 4 hodin měření je uvedeno v následující tabulce 6.2.

Systém	**L1/E1**		**L5/E5**	
	GNSS Agent	**Geo++**	**GNSS Agent**	**Geo++**
GPS	74 (18%)	54 (13%)	74 (61%)	54 (44%)
Galileo	191 (71%)	90 (37%)	191 (75%)	90 (48%)

Tab. 6.2: Součet fázových skoků, 4 hodiny (zdroj: vlastní)

Procentuální zastoupení fázových skoků je ve všech případech nižší u aplikace Geo++. U systému Galileo toto podporuje hypotézy, jež byly přednesené v přechozích analýzách; Geo++ více filtruje družice se slabším signálem, proto je observací i fázových skoků méně. Díky vyššímu počtu fázových skoků v aplikaci GNSS Agent i u systému GPS je možné vyslovit tvzení, že aplikace Geo++ nefiltruje data pouze na základě síly signálu, ale i možná podle dalších kritérií, bohužel detailní informace nelze získat.

Počty fázových skoků u měření geodetickou anténou byly téměř nulové, nejsou tedy ve srovnání uvedeny. Nicméně tato informace může sloužit jako demonstrace značně rozdílné úrovně kvality dat přijatých různými přijímači a tudíž potenciálním problémům při využití přesných fázových měření na mobilních zařízení.

6.2 Poloha

Pro výpočet polohy byl využit software G-Nut/Geb. Jedná se o otevřený nástroj, který je podobně jako G-Nut/Anubis vytvořen za využití knihovny G-Nut. Zpracovávaná data byla nasbírána 2. 5. 2019 aplikací GNSS Agent.

6.2.1 Použité strategie

Za asistence Ing. Pavla Václavovice Ph.D. byl software nakonfigurován pro výpočet několika strategiemi se stupňující se přesností.

První strategie by měla simulovat metodu použitou při určení polohy čipsetem mobilního telefonu. Zároveň tato strategie slouží k přímému porovnání s G-Nut/Anubis, jehož výstupem je také vypočítaná poloha. Pro určení polohy využívá G-Nut/Anubis

pouze dvoufrekvenční data, díky kterým je možné eliminovat působení ionosféry bez použití externích modelů. Z důvodu nízké kvality i kvantity získaných fázových měření na dvou frekvencích, bylo přistoupeno k jednofrekvenčnímu řešení, které G-Nut/Anubis neposkytuje. První nakonfigurovaná strategie v G-Nut/Geb je proto v podstatě modifikací druhého softwaru za předpokladu, že by mohl využít jednofrekvenční data.

Použité strategie jsou definované v následující tabulce 6.3:

	1. strategie	**2. strategie**	**3. strategie**
Kódová měření	Ano	Ano	Ano
Fázová měření	Ne	Ne	Ano
2. frekvence	Ne	Ne	Ne
Navigační zpráva	Ano	Ano	Ano
Ionosférický model	Ano	Ano	Ano
Přesné korekce hodin	Ne	Ano	Ano
Přesné korekce drah	Ne	Ano	Ano

Tab. 6.3: Popis strategií pro výpočet polohy (zdroj: vlastní)

6.2.2 Přesnost vypočtené polohy

Z nasbíraných dat byl použit počáteční úsek dlouhý 1 hodinu a 15 minut především z důvodu rychlého úbytku počtu observovaných družic systému Galileo. První strategií byla vypočtena poloha za použití všech systémů a poté zvlášť za použití systémů GPS a Galileo pro porovnání jejich kvality. Druhá strategie byla pak použita pro výpočet polohy za využití družic všech systémů. Třetí strategie, jež by měla dávat nejlepší výsledky, byla nejprve použita rovněž za využití všech systémů, poté byly odstraněny družice systému Galileo pro zjištění, jakým způsobem výpočet zpřesňují nebo zhoršují.

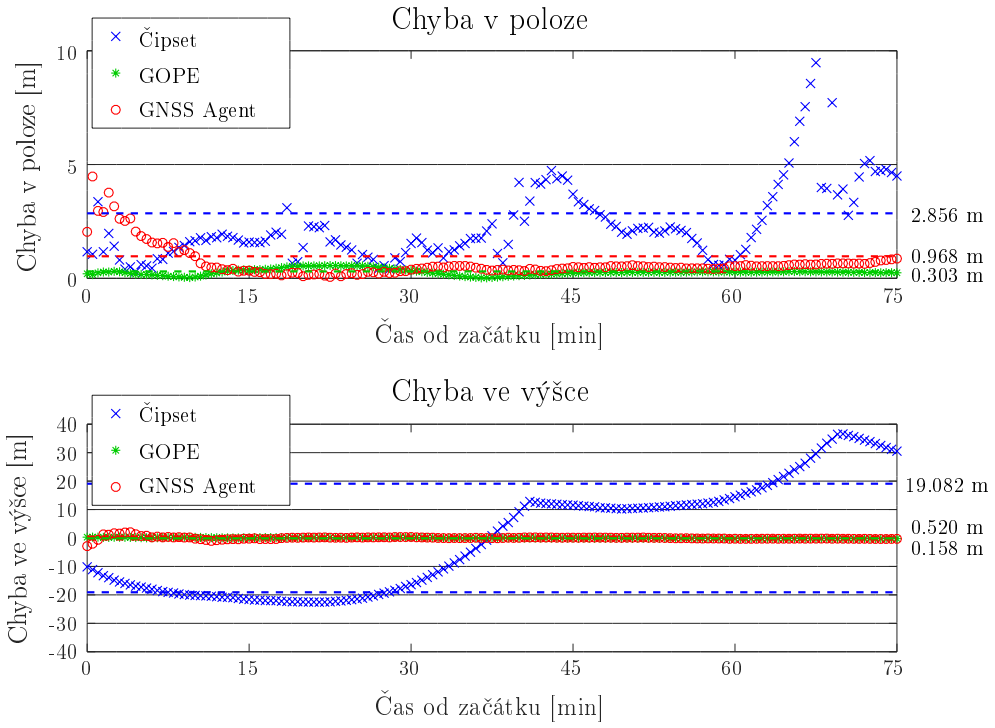
Poloha byla vypočtena se vzorkováním 30 sekund. Ze všech vypočtených hodnot byla vždy zjištěna směrodatná odchylka od průměru v poloze a ve výšce. Všechny vypočtené směrodatné odchylky jsou porovnány v následující tabulce 6.4:

	1. strategie			2. strategie	3. strategie	
	GNSS	GPS	GAL	GNSS	GNSS	Bez GAL
σ_p [m]	5.810	9.065	8.945	5.649	0.968	1.637
σ_v [m]	7.478	11.666	10.036	7.371	0.520	3.614

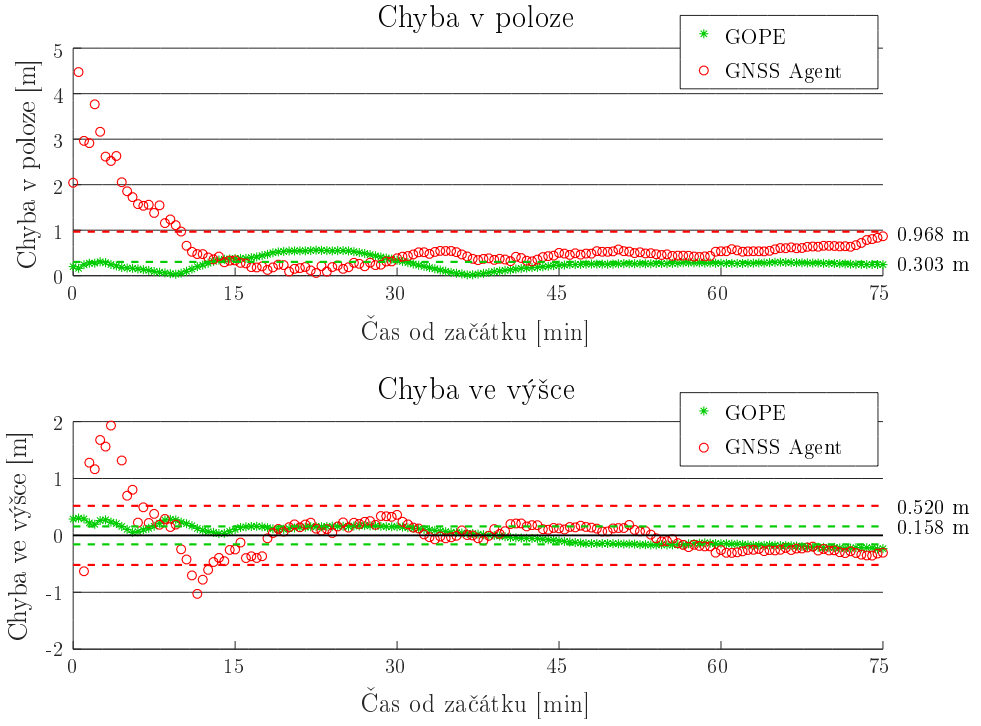
Tab. 6.4: Směrodatné odchylky GNSS Agent, různé metody (zdroj: vlastní)

Z tabulky 6.4 je patrné, že v případě všech systémů se přesnost výpočtu zlepšuje s využitím přesnějších strategií. Při porovnání první strategie samostatně pro systémy GPS a Galileo je znatelné, že ve výšce a i v poloze, leč minimálně, dává Galileo lepší výsledky. Zároveň z experimentu s použitím třetí strategie vyplývá, že i v současné době nekompletní systém Galileo tento výpočet také zpřesňuje.
Pro další srovnání byla za stejný časový úsek a se stejným vzorkováním vypočtena poloha za využití dat získaných geodetickou anténou na GOPE. K výpočtu byla použita třetí strategie za shodné konfigurace. Dále byly opět vypočteny směrodatné odchylky v poloze a ve výšce. Zároveň byly vypočteny směrodatné odchylky z určené polohy čipsetem telefonu.

Nejlepší výsledky jsou srovnány na grafech na následujícím obrázku 6.9. Přerušované čáry s hodnotami vpravo označují směrodatné odchylky, body pak určují rozdíl polohy a výšky od průměru. Jelikož hodnoty pro GNSS Agent a GOPE splývají, jsou prezentovány s větším rozlišením v grafech na dalším obrázku 6.10.



Obrázek 6.9: Porovnání nejpřesnějšího výpočtu polohy (zdroj: vlastní)



Obrázek 6.10: Porovnání nejpřesnějšího výpočtu polohy, detail (zdroj: vlastní)

Porovnání je dále pro přehlednost prezentováno v následující tabulce 6.5 se směrodatnými odchylkami:

	GNSS Agent	**Čipset**	**GOPE**
σ_p [m]	0.968	2.856	0.303
σ_v [m]	0.520	19.082	0.158

Tab. 6.5: Směrodatné odchylky nejpřesnějšího výpočtu polohy (zdroj: vlastní)

Nejprve je třeba pro úplnost připomenout, že i výpočet s daty z GOPE je prováděn pouze jednofrekvenčně. Vícefrekvenční řešení by zlepšilo přesnost do řádu centimetrů.

Při srovnání tabulek 6.4 a 6.5 je vidět, že pro první dvě strategie dává čipset lepší výsledky v poloze. Čipset pro výpočet polohy zřejmě využívá kódové měření, které je vyhlazeno s pomocí fázových měření (patrné z grafů na obrázku 6.9).

Z výsledků lze s potěšením konstatovat, že za využití surových GNSS observací lze dosáhnout výsledků o řád lepších než s využitím výpočtu polohy čipsetem. Nicméně je nutné zdůraznit, že pro výpočet byla použita nejlepší data, která se podařila telefonem nasbírat. Měření může jinak kolísat od submetrové až po metrovou přesnost (na rozdíl od geodetického přijímače, jež má přesnost stabilně vysokou).

7 Závěr

Cílem této diplomové práce bylo zpracování nově dostupných surových GNSS měření prostřednictvím mobilního telefonu a porovnání výpočtu polohy s polohou vypočtenou mobilním GNSS čipsetem. Dále bylo třeba se seznámit se základy programování pro Android a s novým Google API, jež surová měření poskytuje.

V rámci diplomové práce byla vytvořena vlastní aplikace GNSS Agent sloužící pro sběr surových měření a jejich ukládání do formátu, jež je čitelný pro další nástroje určené pro post-processingové zpracování. Aplikace dále ukládá polohu vypočtenou GNSS čipsetem pro všechny epochy surových měření, kterou zároveň zobrazuje na mapě. Vedle sběru dat aplikace dále v reálném čase zobrazuje počty observovaných družic a sílu jejich signálů. Aplikace také umožňuje základní správu vytvořených souborů - jejich otevírání, mazání a sdílení.

Pro sběr dat byl použit mobilní telefon Xiaomi Mi 8 umožňující dvoufrekvenční pozorování. Výstupy z aplikace GNSS Agent a z již existující aplikace Geo++ RINEX Logger (dále jen Geo++) byly zpracovány pomocí otevřených nástrojů G-Nut/Anubis a G-Nut/Geb. Pro srovnání byla také zpracována data získaná geodetickou anténou na střeše Geodetické observatoře Pecný. Z analýzy počtu observovaných družic vyplývají malé rozdíly v implemetaci použitých aplikací. Zejména se jedná o systém Galileo, kdy aplikace Geo++ nasbírala méně dvoufrekvenčních observací. Při další analýze bylo ukázáno, že Geo++ zřejmě více filtruje přijatá data, jelikož vykazuje nižší počty fázových skoků než aplikace GNSS Agent. Tyto rozdíly v kódech aplikací bohužel nelze porovnat, jelikož aplikace Geo++ není otevřená. Aplikace GNSS Agent byla implementována především dle oficiálního dokumentu agentury GSA (European GNSS Agency), jež popisuje využití surových GNSS měření pomocí mobilních telefonů. Zdrojový kód je dostupný z autorova GitHub repozitáře na `https://github.com/kalator/GnssAgent`.

Obě použité aplikace a zároveň informace z mobilního GNSS čipsetu o počtu využitých družic vykazují trend úbytku družic Galileo během prvních několika hodin pozorování. Cílem bylo sbírat data po celý den, ve všech pokusech se ukázalo, že po určitém čase na několik hodin přestal telefon data sbírat. Poté se sběr vždy znovu zrestartoval a tím zapříčinil i obnovu pozorování družic systému Galileo. Trend

úbytku u Galilea se ovšem opět opakoval a byl shodný pro všechny metody sběru dat z čipu.

Dále byla analyzována síla signálu družic. Signál by měl být nejsilnější ve stavu, kdy se družice nachází v nadhlavníku a nejslabší v blízkosti obzoru. Toto očekávání se potvrdilo v případě analýzy dat získaných geodetickou anténou, nicméně síla signálu pozorování mobilním telefonem zůstává velmi proměnlivá během celé cesty družice po obloze a zdaleka nenabývá maximálních hodnot jakých dosahují geodetické antény.

Pro výpočet polohy byly použity různé strategie využívající kódové a fázové měření, navigační zprávy a další přesné produkty (ionosférický model, přesné korekce hodin, přesné dráhy družic). Při výpočtech se ukázalo, že dvoufrekvenční pozorování jsou v důsledku jejich nízkého počtu v podstatě nepoužitelná. S jednofrekvenčními daty se podařilo dosáhnout přesnosti v poloze i ve výšce o řád lepší než při výpočtu polohy GNSS čipsetem, což demonstruje budoucí potenciál mobilních zařízení. Zároveň byla potvrzena teorie, že lze za pomocí měření mobilním telefonem získat polohu s přesností pod 1 metr.

S možností získávání surových GNSS měření se nízkonákladová levná mobilní zařízení (ve srovnání s cenou geodetických přijímačů) posouvají o 1-2 řády směrem k přesnějšímu určení polohy (v řádu decimetrů) vůči poloze získávané z čipsetu. Pro běžné aplikace jistě postačují méně přesné výpočty polohy GNSS čipsetem také vzhledem k tomu, že při měření je jednou z priorit šetření energie baterie. Vysoké nároky na spotřebu energie bohužel ovlivňuje i sběr surových měření a dnešní mobilní telefony často nejsou pro neustálé sledování družic uzpůsobené. U novějších modelů mobilních telefonů včetně Xiaomi Mi 8 lze nastavit, aby družice byly či nebyly sledovány neustále. Toto nastavení se nicméně skrývá v možnostech pro vývojáře, které nejsou implicitně viditelné. Situaci dobře ilustrují popisované několikahodinové výpadky měření.

S vývojem aplikace GNSS Agent je počítáno i do budoucna. V plánu je doplnit získávaná data o navigační zprávy, které momentálně aplikace neposkytuje. Při dalším vývoji se předpokládá propojení s nástrojem G-Nut/Anubis a G-Nut/Geb pomocí integrace kódu v jazyce C++ pro určování polohy s využitím fázových měření a s podporou přesných produktů. Dále je počítáno s integrací enkódérů pro

standardní soubory dat formátu RINEX a podporou posílání měření v datovém proudu přímo v reálném čase ve formátu RTCM.

Literatura

[1] PETROVSKI, Ivan G. *GPS, GLONASS, Galileo and BeiDou for mobile devices.* Cambridge: Cambridge University Press2014. ISBN 978-1-107-03584-3.

[2] *GPS: Space segment* [online]. [cit. 2019-05-04]. Dostupné z: `https://www.gps.gov/systems/gps/space`.

[3] *Information and Analysis Center for Positioning: GLONASS* [online]. [cit. 2019-05-04]. Dostupné z: `https://www.glonass-iac.ru/en/GLONASS`.

[4] *BeiDou Navigation Satellite System* [online]. [cit. 2019-05-04]. Dostupné z: `http://en.beidou.gov.cn`.

[5] *European GNSS Service Centre: Galileo programme* [online]. [cit. 2019-05-04]. Dostupné z: `https://www.gsc-europa.eu/galileo-gsc-overview/programme`.

[6] *European Space Agency navipedia: GNSS signal* [online]. [cit. 2019-05-04]. Dostupné z: `https://gssc.esa.int/navipedia/index.php/GNSS_signal`.

[7] *Explore Labs: Designing A GPS Receiver* [online]. [cit. 2019-05-04]. Dostupné z: `http://www.explorelabs.com/blog/designing-a-gps-receiver`.

[8] MERVART, Leoš. *Globální polohový systém.* Praha: Vydavatelství ČVUT1994. ISBN 80-01-01221-2.

[9] ČÁBELKA, Miroslav. *Úvod do GPS* [online]. [cit. 2019-05-04]. Dostupné z: `https://www.natur.cuni.cz/geografie/geoinformatika-kartografie/ke-stazeni/vyuka/gps/skriptum-uvod-do-gps`.

[10] *European Space Agency navipedia: Ionospheric Delay* [online]. [cit. 2019-05-04]. Dostupné z: `https://gssc.esa.int/navipedia/index.php/Ionospheric_Delay`.

[11] The GNSS Raw Measurements Task Force. *Using GNSS raw measurements on Android devices (white paper).* Luxembourg: Publications Office of the European Union2017. ISBN 978-92-9206-033-6.

[12] KOSTELECKÝ, Jan; KLOKOČNÍK, Jaroslav a KOSTELECKÝ, Jakub. *Kosmická geodézie.* České vysoké učení technické v Praze: Nakladatelství ČVUT2008. ISBN 978-80-01-04059-1.

[13] *International Earth Rotation and Reference Systems Service: IERS Bulletin C (leap second announcements - latest issue* [online]. [cit. 2019-05-04]. Dostupné z: `https://www.iers.org/SharedDocs/News/EN/BulletinC.html`.

[14] *European Space Agency navipedia: GNSS Basic Observables* [online]. [cit. 2019-05-04]. Dostupné z: `https://gssc.esa.int/navipedia/index.php/GNSS_Basic_Observables`.

[15] *European GNSS Agency: Test your satellite navigation performance on your Android device, Glossary* [online]. [cit. 2019-05-04]. Dostupné z: `https://www.gsa.europa.eu/sites/default/files/understanding_gnss_performance_on_android_using_the_gps_testc_app.pdf`.

[16] *European Space Agency navipedia: Interfaces and Protocols* [online]. [cit. 2019-05-04]. Dostupné z: `https://gssc.esa.int/navipedia/index.php/Interfaces_and_Protocols`.

[17] WEBER, Georg et al. *BKG Ntrip Client (BNC)* [online]. [cit. 2019-05-04]. Dostupné z: `https://software.rtcm-ntrip.org/export/HEAD/ntrip/trunk/BNC/src/bnchelp.html`.

[18] *Android Source: Android green robot* [online]. [cit. 2019-05-04]. Dostupné z: `https://source.android.com/setup/images/Android_greenrobot.png`.

[19] *Wikipedia: Android (operating system)* [online]. [cit. 2019-05-04]. Dostupné z: `https://en.wikipedia.org/wiki/Android_(operating_system)`.

[20] *Android Source: Contributing* [online]. [cit. 2019-05-04]. Dostupné z: `https://source.android.com/setup/contribute`.

[21] *Android Developers: Application Fundamentals* [online]. [cit. 2019-05-04]. Dostupné z: `https://developer.android.com/guide/components/fundamentals.html`.

[22] *Wikipedia: Android software development* [online]. [cit. 2019-05-04]. Dostupné z: https://en.wikipedia.org/wiki/Android_software_development.

[23] *Android Developers: Build your first app* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/training/basics/firstapp.

[24] *Android Developers: Permissions overview* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/guide/topics/permissions/overview.

[25] *Wikimedia Commons: Android Studio icon* [online]. [cit. 2019-05-04]. Dostupné z: https://commons.wikimedia.org/wiki/File:Android_Studio_icon.svg.

[26] *Android Developers: Android Studio* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/studio.

[27] *Android Developers: Create and manage virtual devices* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/studio/run/managing-avds.

[28] *Android Developers: Create an Android project* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/training/basics/firstapp/creating-project.

[29] *Android Developers: App resources overview* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/guide/topics/resources.

[30] *Android Developers: Configure your build* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/studio/build.

[31] *Android Developers: App Manifest Overview* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/guide/topics/manifest/manifest-intro.html.

[32] LACKO, Luboslav. *Mistrovství - Android.* Brno: Computer Press2017. ISBN 978-80-251-4875-4.

[33] *Android Developers: Understand the Activity Lifecycle* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/guide/components/activities/activity-lifecycle.html.

[34] *Android Developers: Fragments* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/guide/components/fragments.

[35] *Google Play: Služby Google Play* [online]. [cit. 2019-05-04]. Dostupné z: https://play.google.com/store/apps/details?id=com.google.android.gms.

[36] *Android Developers: Background Location Limits* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/about/versions/oreo/background-location-limits.html.

[37] *Android Developers: Location strategies* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/guide/topics/location/strategies.

[38] *Android Developers: LocationManager* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/reference/android/location/LocationManager.

[39] *Android Developers: Location* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/reference/android/location/Location.html.

[40] *Android Developers: Raw GNSS Measurements* [online]. [cit. 2019-05-04]. Dostupné z: https://developer.android.com/guide/topics/sensors/gnss.

[41] *GNSS Compare Documentation* [online]. [cit. 2019-05-04]. Dostupné z: https://gnss-compare.readthedocs.io.

[42] *Geo++: Logging Of GNSS Raw Data On Android* [online]. [cit. 2019-05-04]. Dostupné z: http://www.geopp.de/logging-of-gnss-raw-data-on-android.

[43] *StatCounter: Mobile and Tablet Android Version Market Share Worldwide* [online]. [cit. 2019-05-04]. Dostupné z: http://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide.

[44] *Geodetic observatory Pecný: G-Nut/Anubis APPLICATION* [online]. [cit. 2019-05-04]. Dostupné z: http://software.pecny.cz/anubis/.