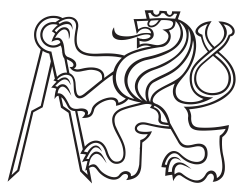


Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Navigace v budovách za použití mobilních telefonů

Bc. Miroslav Voda

Vedoucí: Kopřiva Štěpán MSc.
Květen 2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Miroslav Voda**

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: **Navigace v budovách za použití mobilních telefonů**

Pokyny pro vypracování:

- 1) Nastudujte možnosti využití mobilních telefonů pro navigace v uzavřených budovách (nemocnice, školy), zejména možnosti low-energy technologií, wi-fi a akcelerometrů a gyroskopů mobilních telefonů. Zejména se zaměřte na low-energy technologie (iBeacons).
- 2) Vytvořte grafické rozhraní pro vytváření plánů budov použitelné na mobilním telefonu.
- 3) Vytvořte aplikaci, která umožní provozovateli navigace umístit do plánu budov existující navigační prvky (wi-fi, iBeacons).
- 4) Navrhněte metodu navigace v místnostech. Metoda má umožňovat navigovat uživatele pomocí mobilního telefonu z bodu A do bodu B v uzavřené budově za pomoci existujících a přidaných informačních majáků - wifi, iBeacons.
- 5) Naimplementujte metodu z předchozího bodu.
- 6) Naimplementujte aplikaci, která bude uživatele navigovat v budově mezi význačnými body. Body budou nadefinovány v aplikaci předem.
- 7) Otestujte aplikaci z předchozího bodu na syntetickém scénáři.

Seznam odborné literatury:

- 1) Stuart J. Russell and Peter Norvig. 2003. Artificial Intelligence: A Modern Approach (2 ed.). Pearson Education.
- 2) Matthew S. Gast : Building Applications with iBeacon: Proximity and Location Services with Bluetooth Low Energy
Publisher: O'Reilly Media

Vedoucí: MSc. Štěpán Kopřiva, MSc.

Platnost zadání: do konce zimního semestru 2016/2017


doc. Ing. Filip Železný, Ph.D.
vedoucí katedry




prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 8. 10. 2015

Poděkování

Tímto bych chtěl poděkovat své rodině a přítelkyni za jejich podporu po celou dobu studia. Dále bych chtěl poděkovat vedoucímu této práce za jeho profesionální přístup a rady.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 25. 5. 2016

Abstrakt

Cílem této práce je nastudovat možnosti využití mobilních telefonů pro navigaci ve vnitřním prostředí a navrhnout navigační metodu. Tuto metodu následně implementovat do mobilní aplikace a otestovat na reálném prostředí.

Pro realizaci navigační metody bylo nutné navrhnout a implementovat vlastní krokoměr a kompas. Navigační metoda je postavena na metodě *PDR* s odhadem počáteční polohy pomocí *WiFi* a *Bluetooth Low Edition fingerprinting*. Změna polohy uživatele je filtrována částicovým filtrem, který udržuje odhadnutou polohu v dosažitelných místech mapy. Pro vytváření mapových podkladů byla vytvořena separátní aplikace. Obě aplikace byly implementovány pro operační systém *Android*.

Testování navržené navigační metody prokázalo její použitelnost v reálném prostředí.

Klíčová slova: navigace, smartphone, Android, iBeacon, EddyStone, WiFi, Bluetooth, BLE, IMU, částicový filtr, fingerprinting, kompas, krokoměr

Vedoucí: Kopřiva Štěpán MSc.
Katedra počítačů (13136)
Fakulta elektrotechnická
České vysoké učení technické v Praze

Abstract

This thesis aims to analyze the possibilities of using a mobile device for indoor navigation. It also aims to propose a method for navigation in this environment, implement this method as a mobile application and test the resulting application in real environment.

It was necessary to propose and implement pedometer and compass algorithms in order to create the method for navigation. The proposed method is based on *PDR* method with initial detection of the location by *WiFi* and *BLE fingerprinting*. A particle filter, which limits observed position in a reachable area of the map, is used to improve the detection of the change of the user's position. A separate application was implemented to create the map layers. Both of these applications were implemented for the *Android* system.

Usability of the proposed method for indoor navigation was proven by performed tests.

Keywords: navigation, smartphone, Android, iBeacon, EddyStone, WiFi, Bluetooth, BLE, IMU, Particle Filter, Fingerprinting, Compass, Pedometer, Dead Reckoning, PDR

Title translation: Navigation in buildings using a smartphone

Obsah

1 Úvod	1		
1.1 Cíle práce	1		
1.2 Struktura práce	2		
2 Přehled technologií pro navigaci	3		
2.1 Globální družicový polohový systém	3		
2.2 Bezdrátové technologie	4		
Bluetooth	4		
Wi-Fi (WLAN)	5		
NearFieldCommunication (NFC)	6		
2.3 Senzory pohybu a orientace	6		
Akcelerometr	6		
Gyroskop	7		
Senzor lineární akcelerace	8		
Magnetometr	9		
Krokoměr	10		
Senzor gravitace	10		
Senzor orientace	11		
2.4 Ostatní způsoby detekce polohy	12		
3 Současná řešení	13		
4 Definice problému	15		
4.1 Co je to navigace?	15		
4.2 Navigační proces	15		
4.3 Prostředí	16		
4.4 Lokalizace	16		
4.4.1 Problém lokalizace dle GNSS	16		
4.4.2 Magnetická rušení	16		
4.5 Změna polohy uživatele	17		
4.6 Umístění navigačního zařízení	17		
4.7 Hledání cesty	18		
4.7.1 Požadovaná přesnost navigační metody	18		
4.7.2 Omezení navigační metody	18		
5 Návrh řešení	19		
5.1 Mapové podklady	19		
5.1.1 Specifikace požadavků	19		
5.1.2 Návrh struktury mapových podkladů	20		
5.2 Návrh uživatelského rozhraní	24		
5.2.1 Persona	24		
5.2.2 Struktura obrazovek	25		
5.3 Lokalizace	36		
5.3.1 Bezdrátové technologie	36		
5.3.2 <i>Pedestrian Dead Reckoning</i>	42		
5.3.3 Kombinace <i>PDR</i> a <i>fingerprintingu</i>	52		
5.3.4 Výsledný proces lokalizace	56		
5.4 Navigace	57		
5.4.1 Význačné body	57		
5.4.2 Hledání cesty	57		
5.4.3 Výsledný proces navigace	59		
5.4.4 Návrh uživatelského rozhraní	59		
6 Implementace	65		
6.1 Zastoupení operačních systémů	65		
6.2 Mobilní platforma	65		
6.3 Vývojové prostředí	65		
6.4 Vývojové zařízení	66		
6.4.1 Senzory vývojového zařízení	67		
6.5 Architektura systému Android	67		
6.6 Data ze senzorů	68		
6.7 Aplikace pro vytváření mapových podkladů	69		
6.7.1 Komponenta pro vykreslování mapových podkladů	69		
6.7.2 Grafické rozhraní	69		
6.7.3 Reprezentace a persistence map	75		
6.8 Lokalizace	75		
6.8.1 Nativní krokoměr a senzor orientace	75		
6.8.2 Podpůrné aplikace	76		
6.8.3 Implementace krokoměru	77		
6.8.4 Implementace kompasu	77		
6.8.5 Implementace lokalizace pomocí bezdrátových technologií (<i>fingerprinting</i>)	78		
6.8.6 Částečný filtr	78		
6.9 Navigace	80		
6.9.1 Hledání cesty	80		
6.9.2 Výsledná implementace lokalizace a navigace	80		
6.9.3 Grafické rozhraní navigační aplikace	81		
6.10 Použité knihovny	83		
7 Testování	85		
7.1 Testovací zařízení	85		
7.2 Testovací prostředí	85		
7.3 Testování navržené metody	86		
7.3.1 Krokoměr	88		
7.3.2 Kompas	89		

7.3.3 Bezdrátové majáky	90
7.3.4 Porovnání metod <i>fingerprintingu</i>	91
7.3.5 Zdroje chyb	94
7.4 Testování implementace	94
8 Závěr	95
8.1 Práce do budoucna	96
Literatura	97
A Struktura projektu	101
B Seznam senzorů testovacího zařízení	103
C JSON struktura mapových podkladů	105
D Obsah CD	107

Obrázky

2.1 Souřadný systém vůči zařízení (zdroj: developer.android.com) . . .	6	5.16 Frekvenční pásma <i>Wi-Fi</i> a <i>Bluetooth Low Edition</i> (převzato z [14])	38
2.2 Akcelerace - zařízení fixováno .	7	5.17 Vliv lidské chůze na akceleraci působící na mobilní zařízení (převzato z [35])	43
2.3 Úhlová rychlost otáčení - zařízení fixováno	8	5.18 Velikost akcelerace na jednotlivých osách v průběhu chůze	44
2.4 Lineární akcelerace - zařízení fixováno	9	5.19 Velikost normy akcelerace a její filtrovaný průběh	44
2.5 Velikost magnetického pole - zařízení fixováno	10	5.20 Souřadný systém orientace zařízení v prostoru (převzato z [41])	49
2.6 Velikost gravitační síly - zařízení fixováno	11	5.21 Proces lokalizace	56
2.7 Orientace zařízení - zařízení fixováno	11	5.22 Relace mezi obrazovkami (<i>screenflow</i>)	60
4.1 Umístění navigačního zařízení	17	5.23 Obrazovka instrukcí (<i>wireframe</i> , <i>mockup</i>)	60
5.1 Souřadný systém mapových podkladů a umístění grafického podkladu	21	5.24 Obrazovka výběru cíle (<i>wireframe</i> , <i>mockup</i>)	61
5.2 Grafický podklad a množina překážek	22	5.25 Obrazovka lokalizace uživatele (<i>wireframe</i> , <i>mockup</i>)	61
5.3 Grafický podklad, množina překážek a body zájmu	22	5.26 Obrazovka navigace (<i>wireframe</i> , <i>mockup</i>)	62
5.4 Relace mezi obrazovkami (<i>screenflow</i>)	26	5.27 Obrazovka dosažení cíle (<i>wireframe</i> , <i>mockup</i>)	63
5.5 Hlavní obrazovka (<i>wireframe</i> , <i>mockup</i>)	27	6.1 <i>Samsung Galaxy S6</i> (zdroj: Samsung.com)	66
5.6 Krok 1 (<i>wireframe</i> , <i>mockup</i> . .	28	6.2 Architektura operačního systému <i>Android</i> (převzato z [49])	68
5.7 Krok 2 (<i>wireframe</i> , <i>mockup</i> , nápopvěda)	29	6.3 Hlavní obrazovka (snímek obrazovky)	70
5.8 Krok 3 (<i>wireframe</i> , <i>mockup</i> , nápopvěda)	29	6.4 Krok 1 a nápopvěda (snímek obrazovky)	70
5.9 Krok 4 (<i>wireframe</i> , <i>mockup</i> , nápopvěda)	30	6.5 Krok 2 a nápopvěda (snímek obrazovky)	71
5.10 Krok 5 (<i>wireframe</i> , <i>mockup</i> , nápopvěda)	31	6.6 Krok 3 a nápopvěda (snímek obrazovky)	71
5.11 Krok 6 (<i>wireframe</i> , <i>mockup</i> , nápopvěda)	32	6.7 Krok 4 a nápopvěda (snímek obrazovky)	72
5.12 Krok 7 (<i>wireframe</i> , <i>mockup</i> , nápopvěda)	32	6.8 Krok 5 a nápopvěda (snímek obrazovky)	72
5.13 Krok 8 (<i>wireframe</i> , <i>mockup</i> , nápopvěda)	33	6.9 Krok 6 a nápopvěda (snímek obrazovky)	73
5.14 Krok 9 (<i>wireframe</i> , <i>mockup</i>) .	34		
5.15 Detail mapových podkladů (<i>wireframe</i> , <i>mockup</i>)	35		

6.10 Krok 7 a nápověda (snímek obrazovky)	73
6.11 Krok 8 a nápověda (snímek obrazovky)	74
6.12 Krok 9 a obrazovka Detail (snímek obrazovky)	74
6.13 Podpůrné aplikace (MotionRecorder, Pedometer, Azimuth)	76
6.14 Úvodní obrazovka (vlevo) a obrazovka instrukcí (vpravo) (snímek obrazovky)	81
6.15 Obrazovka výběru cíle a obrazovka lokalizace uživatele (snímek obrazovky)	82
6.16 Obrazovka navigace a obrazovka dosažení cíle (snímek obrazovky)	82
7.1 Plán testovacího prostředí (poskytl: Inovacetrum)	85
7.2 Porovnání navržené metody s metodou <i>PDR</i>	86
7.3 Distribuční funkce odchylky pro metodu <i>PDR</i>	87
7.4 Distribuční funkce odchylky pro navrženou metodu	87
7.5 Azimut uživatele při chůzi ...	89
7.6 Rozmístění <i>iBeacon</i> majáků v testovacím prostředí	90
7.7 Změny síly signálů	91
7.8 Obsazení komunikačních kanálů	91
7.9 Lokalizace pomocí metody KNN pro $K=5$	92
7.10 Lokalizace pomocí metody <i>NN</i>	93
7.11 Lokalizace kombinací <i>WiFi</i> a <i>BLE</i> majáků	93
A.1 Část struktury projektu	101

Tabulky

6.1 Základní technické parametry vývojového zařízení	67
7.1 Porovnání testování metod ...	87
7.2 Krokoměr - porovnání počtu kroků	88
7.3 Krokoměr - porovnání délky kroků	88
B.1 Seznam senzorů testovacího zařízení	104
C.1 JSON struktura mapových podkladů	106

Kapitola 1

Úvod

Dnešní doba je charakteristická svojí uspěchaností, každý stále někam pospíchá a snaží se svůj čas využít naplno. Proto v případě, že musí na něco čekat nebo by něco šlo udělat rychleji, je nespokojený a cítí, že by svůj čas šlo využít lépe. Ať už nějakou konstruktivní činností či zaslouženým odpočinkem po celodenním shonu. Právě takovou situaci, kdy uživatel lidově řečeno "zabíjí čas", je hledání něčeho. Ať už věci či nějakého místa. Právě hledání nějakého místa je téměř jistota, pokud se uživatel nachází v daném prostředí poprvé. Řešením tohoto problému je navigace, která nás bez velkého úsilí dovede na hledané místo. Takovým místem může být například město nebo ulice, ale také například místnost. Za účelem dosažení místa ve venkovním prostředí člověk běžně používá *GPS* navigaci. Dnešní technika umožňuje zabudovat tento šikovný nástroj do zařízení, které většina z nás nosí celý den u sebe. Ano, řeč je o mobilním telefonu. Právě mobilní telefon je skvělý nástroj k šetření, ale i v některých případech k mrhání našeho času. Bohužel k nalezení místa uvnitř budovy, kterým může být například místnost, nelze použít technologii *GPS*. Zde se nabízí myšlenka vytvořit nástroj, který bude šetřit náš čas i při hledání místa v uzavřeném prostředí. Možná tento nástroj neušetří hodiny času, jako *GPS* navigace v autě, ale v dnešní době je každá ušetřená minuta důležitá. Proto jsem se rozhodl vytvořit mobilní aplikaci určenou pro navigaci uživatele v budovách s co největší možnou přesností.

1.1 Cíle práce

Tato práce se zaměřuje na následující cíle:

1. Nastudovat možnosti využití mobilních telefonů pro navigaci

Abych mohl navrhnout metodu navigace ve vnitřním prostředí, bylo nutné provést analýzu technologických možností, které nám mobilní telefony poskytují. Analýza musela obsahovat silné a slabé stránky jednotlivých technologií a jejich možné uplatnění v navigačním procesu.

2. Navrhnout metodu navigace ve vnitřním prostředí

Pro účel navigace uživatele ve vnitřním prostředí bylo potřeba navrhnout vhodnou metodu lokalizace a navigace s ohledem na možnosti mobilního telefonu.

3. Vytvořit aplikaci pro vytváření mapových podkladů

Navigační proces vyžaduje mapové poklady obsahující podobu navigačního prostředí. Pro tvorbu těchto podkladů bylo potřeba navrhnout a implementovat mobilní aplikace pro vytváření mapových podkladů.

4. Vytvořit navigační aplikaci

Na základě navržené metody navigace byla navržena a implementována mobilní aplikace určená k navigaci uživatele vnitřním prostředím.

5. Otestovat navrženou metodu

Navrženou metodu navigace bylo nutné otestovat v reálném prostředí, aby byla potvrzena její funkčnost.

1.2 Struktura práce

Práce je členěna do následujících kapitol:

- **Kapitola 2** popisuje technologické možnosti chytrých mobilních telefonů za účelem zaměření a detekce změny polohy uživatele.
- **Kapitola 3** popisuje současné metody určení polohy uživatele ve vnitřním prostředí.
- **Kapitola 4** definuje problematiku navigace uvnitř budov s ohledem na navigační prostředí a změnu polohy uživatele.
- **Kapitola 5** navrhuje strukturu mapových podkladů a grafické rozhraní aplikace pro jejich tvorbu. Dále tato kapitola obsahuje návrh navigační metody včetně vlastního krokoměru a kompasu. Pro použití navigační metody je zde také navržena navigační aplikace.
- **Kapitola 6** obsahuje výběr mobilní platformy, implementaci navržené metody navigace a implementaci výsledných mobilních aplikací.
- **Kapitola 7** testuje a vyhodnocuje navržené řešení navigace uvnitř budovy a jeho konkrétní implementace pomocí mobilní aplikace.

Kapitola 2

Přehled technologií pro navigaci

V této kapitole budou popsány technologické možnosti chytrých mobilních telefonů za účelem zaměření a detekce změny polohy uživatele. Současná mobilní zařízení jsou vybavena desítkami senzorů, které prostřednictvím mobilních aplikací umožňují rozšířit funkcionalitu *smartphonu*. Obecně mezi technologie vhodné k určení polohy uživatele můžeme zařadit družicové polohové systémy, bezdrátové technologie (*Bluetooth*, *WiFi*, *NFC*), senzory pohybu a orientace (akcelerometr, gyroskop, magnetometr, krokoměr) a ostatní technologie (např. mikrofon, fotoaparát, ...).

2.1 Globální družicový polohový systém

Dle [1] je globální družicový polohový systém (anglicky *GlobalNavigationSatelliteSystem (GNSS)*) “služba umožňující za pomoci družic autonomní prostorové určování polohy s celosvětovým pokrytím. Uživatelé této služby používají malé elektronické rádiové přijímače, které na základě odeslaných signálů z družic umožňují vypočítat jejich polohu s přesností na desítky až jednotky metrů. Přesnost ve speciálních nebo vědeckých aplikacích může být až několik centimetrů až milimetrů.” Většina globálně prodávaných telefonů je vybavena čipem pro americký globální družicový systém *GlobalPositioningSystem (GPS)* (pozn.: přesněji *Navstar GPS*). V některých případech jsou mobilní zařízení také vybavována čipem pro ruský družicový systém *GLObalNavigationSatelliteSystem (GLONASS)* či čínský *Beidou*. Kombinací více globálních družicových systémů lze zajistit rychlejší a přesnější fixaci uživateli polohy. Některé regionální družicové polohové systémy (např. čínský *Beidou* [2]) nejsou na našem území dostupné.

GPS čipy instalované do mobilních telefonů nedosahují tak vysoké přesnosti zaměření polohy jako zařízení používaná pro vojenské účely, ale pro použití běžného uživatele jsou zcela dostatečné. Odchylka skutečné polohy od polohy získané pomocí *GPS* pro civilní použití je dle [3] v ideálních podmínkách v rámci několika metrů. Pro správnou funkčnost družicových polohových systémů je nutná přímá viditelnost několika satelitů služby, tudíž je použitelnost této technologie mimo venkovní prostředí značně limitována. Zejména konstrukční řešení budov nedovoluje přímou viditelnost družic *GPS*, čímž může přesnost určení polohy pomocí této technologie klesnout na desítky metrů či tato služba může být zcela nedostupná.

2.2 Bezdrátové technologie

Bezdrátové technologie jsou technologie komunikující prostřednictvím radiofrekvenčního vlnění. Mezi dvě nejvýznamnější technologie dostupné v mobilních zařízeních patří *Wi-Fi* a *Bluetooth*. Mezi méně rozšířené technologie lze zařadit technologii *NearFieldCommunication (NFC)*. Princip určení polohy pomocí bezdrátových technologií je založen analýze přijatého signálu, nejčastěji na vyhodnocování síly přijatého signálu (anglicky *ReceivedSignalStrength (RSS)*). V reálném prostředí ale dochází při šíření elektromagnetického vlnění prostorem k interferencím, absorpcím a odrazům vysílaného signálu, což značně ovlivňuje podobu přijatého signálu a tudíž i přesnost odhadu vzdálenosti od vysílače. Dalším faktorem ovlivňujícím podobu přijatého signálu je konstrukční řešení antény přijímače, které může mít pro různá zařízení odlišný vliv na sílu přijatého signálu za stejných podmínek. Podobu přijatého signálu také může ovlivňovat způsob držení mobilního zařízení, kdy se uživatelská ruka může nacházet mezi anténou přijímače a vysílače, tudíž může dojít k částečné či úplné absorpci vysílaného signálu.

Bluetooth

Bluetooth je dle [4] standard určený pro bezdrátovou komunikaci mezi zařízeními na krátkou vzdálenost. Komunikace probíhá na veřejném frekvenčním pásmu od 2400 do 2483.5 MHz. Většina běžně dostupných mobilních telefonů je vybavena *Bluetooth* čipem. Od verze *Bluetooth 4.0* lze využít technologii zvanou *BluetoothLowEnergy (BLE)*. *BluetoothLowEnergy (BLE)* není zpětně kompatibilní se staršími verzemi *Bluetooth* a je tedy nutné mít zařízení vybavené čipem podporující verzi minimálně 4.0. Díky nižším energetickým nárokům, menší latenci a vyššímu teoretickému limitu komunikační vzdálenosti mezi zařízeními vytvořila firma *Apple, Inc.* technologii *iBeacon*.

iBeacon

iBeacon je proprietární technologie, která dle [5] v pravidelných intervalech vysílá specifická data, která obsahují unikátní identifikátor vysílacího zařízení - *Major + Minor UUID*. Na základě tohoto identifikátoru je možné po přijetí dat identifikovat vysílače a například vyvolat konkrétní akci (otevřít aplikaci, zobrazit notifikaci, atp.). Komunikace při vysílání těchto dat z *iBeacon* je jednosměrná, tudíž není možné monitorovat zařízení, která data přijala. Odesílaná data dle [5] mj. obsahují informaci o síle vysílaného signálu (anglicky *TX power*), což umožňuje dle skutečné síly přijatého signálu *ReceivedSignalStrength (RSS)* odhadnout přibližnou vzdálenost, ve které se vysílače od přijímače nachází. S dostatečným počtem vhodně napevno umístěných vysílačů lze tuto technologii využít pro určení polohy přijímače v prostoru [6]. Technologie *iBeacon* je již v základu podporována v mobilním operačním systému *Apple iOS* od verze 7.0 a výše. S využitím knihoven lze protokol *iBeacon* používat i na operačním systému *Android* od verze 4.3, ale pouze na zařízeních s modulem podporujícím technologii *BLE*.

■ EddyStone

Google EddyStone je dle [7] konkurenční *open-source* alternativa k technologii *Apple iBeacon* vytvořená společností *Google, Inc.* pod otevřenou licencí *Apache 2.0*. Protokol *EddyStone* pracuje na stejném principu a nabízí podobnou funkcionalitu jako *Apple iBeacon*. Rozdíly mezi těmito technologiemi jsou pouze na softwarové straně. Například *beacon* zařízení od firmy *Estimote* mohou dle [8] pracovat buď v režimu *iBeacon* nebo v režimu, kdy se chovají jako *EddyStone* zařízení. Technologie *EddyStone* je použitelná na systémech *Android* od verze 4.3 a výše buď nativně nebo za použití příslušných knihoven a na operačních systémech *iOS* od verze 7 a výše s využitím knihovny.

■ Ostatní Bluetooth zařízení

Všechna *Bluetooth* zařízení, která jsou viditelná (vysílají *beacon frame*), lze využít jako referenční body pro určení polohy. Pokud se ale nejedná o technologii *BLE*, zařízení mají menší dosah, vyšší latenci a delší periodu vysílání rámce, který zajišťuje viditelnost pro ostatní zařízení. Všechna *Bluetooth* zařízení lze identifikovat dle *MAC* adresy.

■ Wi-Fi (WLAN)

WirelessLocalAreaNetwork (WLAN) je dle [9] (standard *IEEE 802.11*) technologie umožňující připojit elektronická zařízení do bezdrátové počítačové sítě. Tato technologie využívá dle [9] frekvenční pásma 2.4GHz, 3.6GHz (*802.11y*), 5GHz a 60GHz (*802.11ad*). Dnešní chytré mobilní telefony jsou již v základu vybavovány *Wi-Fi* čipy, které umožňují mobilní zařízení připojit k bezdrátovým sítím, standardně na frekvencích 2.4GHz a 5GHz. Pro využití bezdrátových sítí k určení polohy uživatele je důležitý fakt, že každá bezdrátová síť je identifikována identifikátorem sítě *ServiceSetIdentifier (SSID)*, který je periodicky vyslán v tzv. *Beacon Frame* z přístupového bodu (zařízení zprostředkovávající bezdrátovou síť) směrem k ostatním zařízením, aby bylo možné bezdrátovou síť detekovat. Jedna bezdrátová síť se může skládat z několika přístupových bodů, proto se v *Beacon Frame* společně s identifikátorem *ServiceSetIdentifier (SSID)* také posílá identifikátor *BasicServiceSetIdentifier (BSSID)*, který rozlišuje jednotlivé přístupové body. Jedná se tedy o jakousi *MAC* adresu přístupového bodu. Periodické vysílání rámce *Beacon Frame* lze přirovnat k periodickému vysílání, které provádí *Bluetooth Beacon* zařízení, tudíž lze technologii *WirelessLocalAreaNetwork (WLAN)* využít za účelem lokalizace uživatele obdobným způsobem jako *Beacon* technologii, ale bohužel s určitými limitacemi, které vyplývají z odlišného návrhu a využití technologie *WLAN*. Mezi tyto limitace lze zařadit fakt, že *Beacon Frame*, který je vyslán z přístupového bodu, neobsahuje informaci o síle vysílaného signálu, tudíž je nutné sílu vysílaného signálu (*TX power*) zjistit jiným způsobem - například manuálně změřit. Dále je nutné ověřit, že přístupový bod není vybaven technologií pro adaptivní změnu síly vysílaného signálu v čase a v závislosti na aktuálním stavu prostředí (např. zarušení prostředí ostatními přístupovými body). Díky statickému umístění přístupových bodů a periodickému

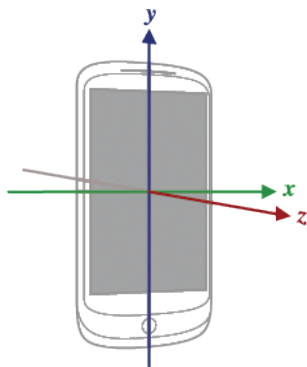
vysílání *Beacon Frame*, lze přístupový bod využít jako referenční bod pro určení polohy uživatele obdobně *Beacon* technologie *BLE*.

■ NearFieldCommunication (NFC)

NFC je dle [10] technologie určená k bezdrátové komunikaci mezi dvěma zařízeními na velmi krátkou vzdálenost a to konkrétně na vzdálenost 4cm a méně. Při použití této technologie lze použít *NFC* štítky (anglicky *NFC tags*) jako referenční body k určení polohy uživatele. Bohužel velkou limitací při použití technologie *NFC* je právě velmi krátká komunikační vzdálenost, kdy uživatel musí zařízení přiložit k referenčnímu zařízení - například k *NFC* štítku. Proto tato technologie je použitelná pro určení polohy uživatele s přesností na několik centimetrů s tím, že se musí uživatel aktivně podílet určení své polohy. Tato technologie tedy není vhodná pro autonomní lokalizaci uživatele.

■ 2.3 Senzory pohybu a orientace

Chytré mobilní telefony jsou vybaveny senzory, které umožňují určit, v jaké poloze se telefon nachází a jakým způsobem mění svou polohu vůči okolnímu prostředí. Mezi takové senzory lze zařadit akcelerometr, gyroskop a magnetometr. Tyto senzory jsou nezbytné jak pro základní funkce mobilního telefonu, jako je například automatické otáčení obrazovky, tak pro pokročilé funkce, jako jsou aplikace virtuální (*VR*) nebo umělé (*AR*) reality. V této sekci bude popsána funkce jednotlivých senzorů a jejich možné využití pro určení polohy uživatele a jeho navigaci prostředím. Pro vztahný systém, kterým je tělo mobilního zařízení se bude brát souřadný systém dle obrázku 2.1.

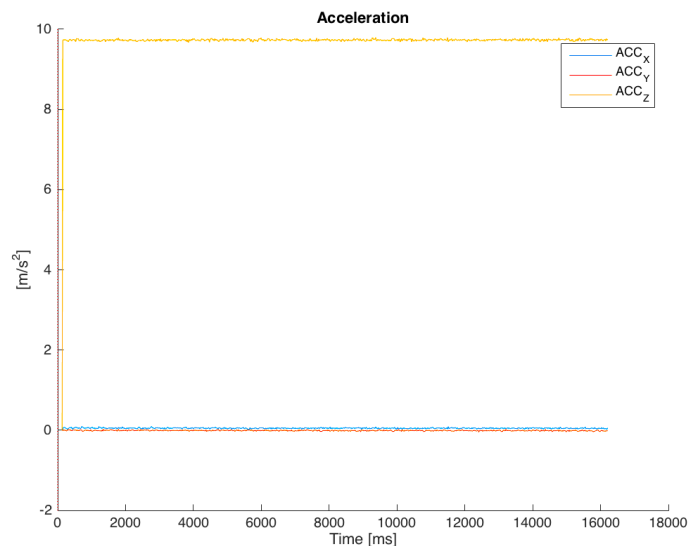


Obrázek 2.1: Souřadný systém vůči zařízení (zdroj:developer.android.com)

■ Akcelerometr

Dle [11] je akcelerometr elektronická součástka, která je schopná detekovat a měřit velikost zrychlení. Jednotkou zrychlení jsou m/s^2 . V běžných mobilních zařízeních jsou senzory schopné měřit akceleraci ve třech na sebe kolmých osách (x, y, z) - viz obrázek 2.1. Akcelerometr také detekuje tíhové zrychlení (9.81 m/s^2), které je rozloženo na daných osách v závislosti na natočení zařízení vůči zemskému povrchu,

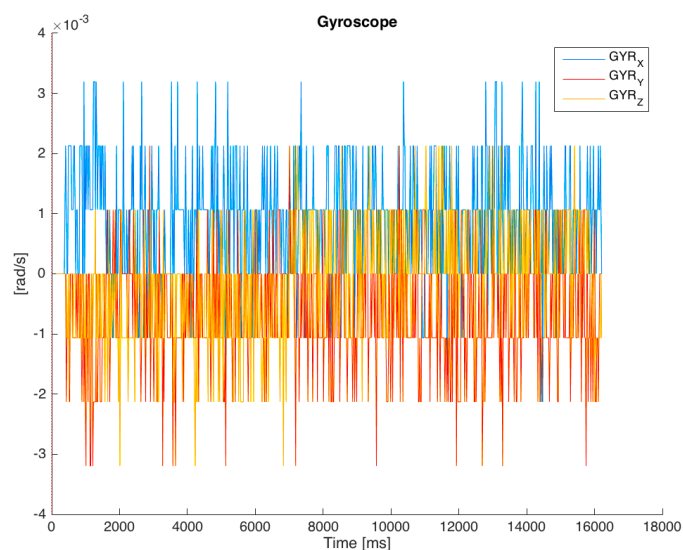
díky čemuž je možné určit naklonění telefonu vůči horizontální rovině zemského povrchu. Obrázek 2.2 zachycuje velikost akcelerace působící na zařízení položené na stole.



Obrázek 2.2: Akcelerace - zařízení fixováno

■ Gyroskop

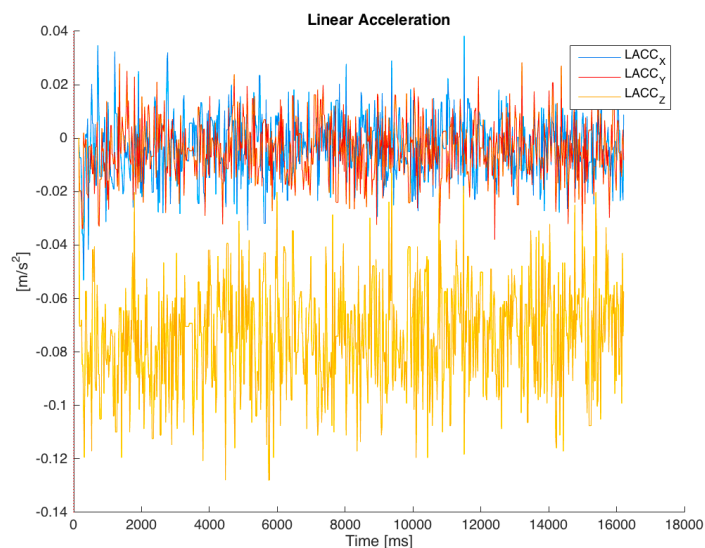
Dle [12] je gyroskop elektronická součástka, která umožňuje měřit velikost rychlosti otáčení. Jednotkou úhlové rychlosti otáčení jsou rad/s. Gyroskop je schopen, stejně jako akcelerometr, měřit data ve třech na sebe kolmých osách (x,y,z). Integrací naměřených hodnot lze získat relativní úhel otočení. Po stabilizaci zařízení v prostoru by měly být naměřené hodnoty nulové. Z obrázku 2.3 je patrné, že naměřená data obsahují chybovou složku, která se integrací kumuluje a může mít negativní vliv na odhadnutý úhel otočení. Tomuto jevu se říká *drift* gyroskopu a musí být eliminován za pomoci jiných senzorů.



Obrázek 2.3: Úhlová rychlost otáčení - zařízení fixováno

■ Senzor lineární akcelerace

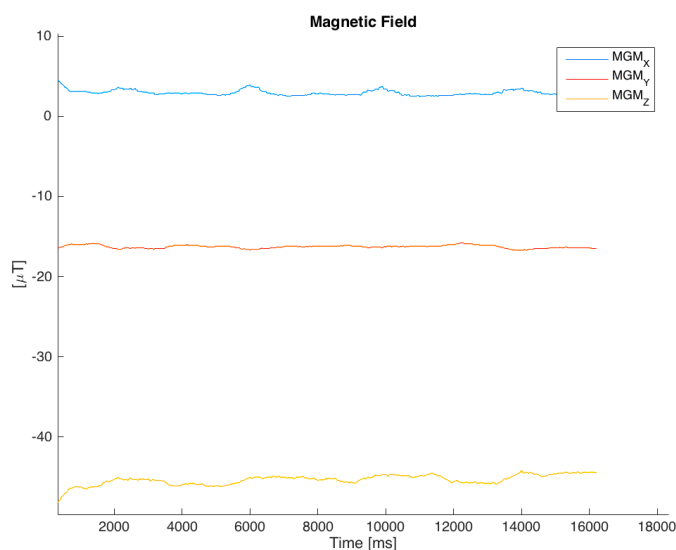
Po odečtení gravitační složky z naměřených dat lze získat samotné zrychlení zařízení - takzvané lineární zrychlení. Stabilizováním zařízení v prostoru by měly být naměřené hodnoty ze všech tří os rovny nule. V operačních systémech je tento senzor dostupný jako softwarový senzor. Softwarový senzor není hardwarová součástka, ale pouze výsledek výpočtů založených na datech z dostupných hardwarových senzorů. V tomto případě softwarový senzor počítá lineární zrychlení. První integrací lineárního zrychlení lze spočítat rychlost, jejíž integrací lze získat dráhu. Z obrázku 2.4 je patrné, že naměřená data stabilizovaného zařízení opět obsahují chybu měření, což se dvojitou integrací dat projeví na velikost chyby kvadraticky. Takto vypočítaná dráha se bude po určité době značně lišit od dráhy skutečné.



Obrázek 2.4: Lineární akcelerace - zařízení fixováno

■ Magnetometr

Magnetometr je dle [13] elektronická součástka určená pro měření síly elektromagnetického pole. Velikost síly magnetického pole se udává v μT . V mobilních telefonech se tento senzor používá převážně k určení světové strany. Magnetometr, jakožto i běžný kompas, je náchylný na změny v okolním elektromagnetickém poli. Většina mobilních telefonů vybavených magnetometrem obsahuje senzor schopný měřit velikost elektromagnetického pole opět ve třech na sebe kolmých osách (x,y,z). Obrázek 2.4 zachycuje velikost síly magnetického pole působícího na stabilizované zařízení. V naměřených datech jsou patrné změny okolního elektromagnetického pole, způsobené elektromagnetickým rušením. K určení světové strany je nutné znát orientaci telefonu vůči horizontální rovině Země.



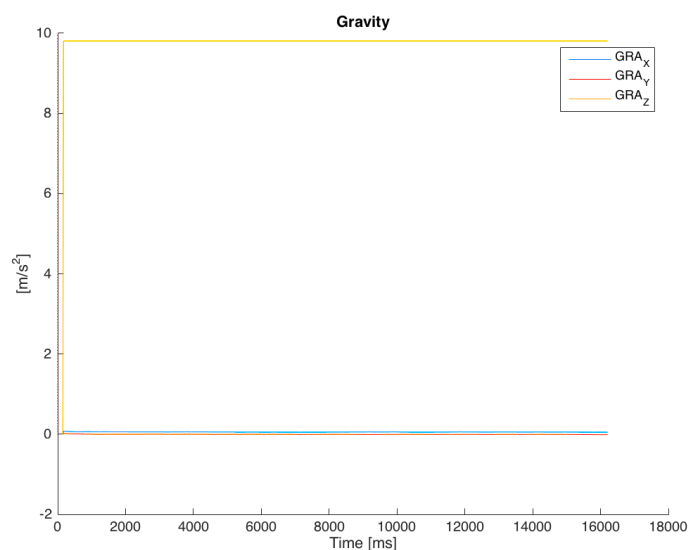
Obrázek 2.5: Velikost magnetického pole - zařízení fixováno

■ Krokoměr

S rostoucím využitím mobilních telefonů při sportování a při sledování zdravého životního stylu jsou dnešní mobilní telefony vybavovány dedikovaným čipem pro počítání ušlých kroků - digitálním krokoměrem. Digitální krokoměr je tedy elektronická součástka detekující kroky. Tato elektronická součástka umožňuje počítat kroky i v případě, že je telefon v režimu spánku (režimu nízké spotřeby). Data z krokoměru lze využít především na sledování uživateli aktivity za určitý čas. Jelikož hardwarový krokoměr, na rozdíl od softwarového krokoměru, neprovádí výpočty na hlavním procesoru, neklade vyšší nároky na výpočetní výkon a umožňuje procesoru přechod do režimu nízké spotřeby, aniž by byl krokoměr pozastaven. V některých telefonech lze nalézt pouze softwarovou implementaci tohoto senzoru.

■ Senzor gravitace

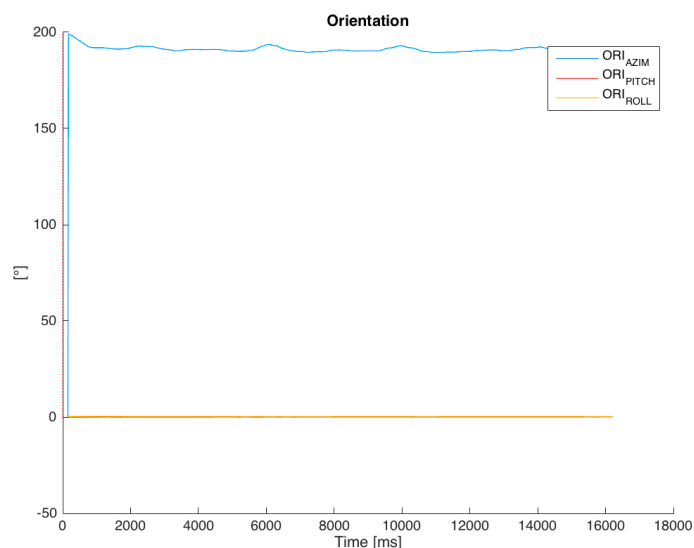
Senzor gravitace je softwarový senzor, který poskytuje informace o velikosti gravitační síly působící na zařízení opět na všech třech osách (x,y,z). Na rozdíl od dat z akcelerometru, senzor gravitace neobsahuje složku lineárního zrychlení působícího na zařízení při změně polohy v prostoru. Obrázek 2.6 zachycuje působení gravitační síly na stabilizované zařízení. Výstup toho senzoru se používá především k určení orientace zařízení vůči horizontální rovině Země.



Obrázek 2.6: Velikost gravitační síly - zařízení fixováno

■ Senzor orientace

Senzor orientace je softwarový senzor, který kombinuje data ze senzoru gravitace, gyroskopu a magnetometru. Výstupem tohoto senzoru je orientace zařízení vůči okolnímu prostředí. Orientací je myšlen úhel naklonění zařízení kolem osy X (tzv. *Pitch*), úhel naklonění zařízení kolem osy Y (tzv. *Roll*) a úhel naklonění zařízení kolem osy Z (tzv. *Yaw* či *Azimuth*). Orientace *Yaw* odpovídá azimutu severního magnetického pólu, proto jsou v obrázku 2.7 patrné změny v elektromagnetickém poli detekované magnetometrem.



Obrázek 2.7: Orientace zařízení - zařízení fixováno

■ 2.4 Ostatní způsoby detekce polohy

Chytrý mobilní telefon je dále vybaven dalšími komponentami a součástkami, které je možné využít za účelem určení polohy uživatele v prostoru. Mezi tyto komponenty lze zařadit například fotoaparát a mikrofon. Určení polohy za pomoci fotoaparátu lze docílit například vyfocení unikátního referenčního bodu (čárový kód, *QR* kód) nebo pomoci počítačového vidění, které vytváří vizuální otisk okolního prostředí. Určení polohy za pomoci mikrofonu lze docílit vysíláním zvukového signálu (nejlépe v neslyšitelném spektru) z referenčního bodu. K této metodě je nutné vybavit prostředí speciálním hardwarem. Jelikož je tato diplomová práce zaměřená především na bezdrátové technologie, nebudou tyto metody lokalizace za použití dalších komponent rozvíjeny.

Kapitola 3

Současná řešení

Tato kapitola bude popisovat existující metody určení polohy uživatele ve vnitřním prostředí.

Práce [14] popisuje možnosti použití metody *fingerprinting* pro určení polohy pomocí *Bluetooth Low Energy* majáků (anglicky *beacons*). Dále tato práce obsahuje podrobnou analýzu propagace signálu prostředím, vliv konfigurace s hustoty majáků na přesnost určení polohy včetně podrobně popsané metody *fingerprintingu* pro *BLE* zařízení.

V práci [15] je navržena a popsána metoda určení polohy pomocí zařízení s operačním systémem *Android* na základě senzorických dat ze senzorů mobilního zařízení. Navrženým řešením je lokalizace uživatele na základě integrace dat z akcelerometru a gyroskopu. V práci je použit nevšední způsob umístění mobilního zařízení, jímž je položení mobilního telefonu na vozík, který lokalizovaný uživatel tlačí před sebou. Pro počáteční zjištění polohy využívají *WiFi fingerprinting*.

Dokument [16] popisuje metodu lokalizace osoby ve vnitřních prostorách. Metoda je založena na částicovém filtru, který modeluje možnou polohu uživatele. Částice jsou propagovány na základě délky a směru kroku a jsou z množiny částí odstraněny v případě, že "narazí" do neprůchozí překážky. Za účelem vygenerování apriorní množiny částic je použit opět *WiFi fingerprinting*. Navržená metoda dosahuje přesnosti pod jeden metr, ale bohužel díky vysokým požadavkům na výpočetní výkon ji není možné použít k výpočtu v reálném čase na mobilním zařízení.

Práce [17] pojednává o návrhu metody lokalizace ve vnitřním prostředí pouze na základě dat z akcelerometru a gyroskopu. Jedná se o metodu na způsob *PDR*.

Autoři práce [18] popisují opět metodu *PDR* na základě detekce kroků a směru chůze. Zvýšení přesnosti metody zajistili použitím částicového filtru na mapové podklady vnitřního prostředí.

Dokument [19] popisuje využití částicového filtru pro zvýšení přesnosti lokalizace ve vnitřním prostředí s využitím mapových podkladů a pravděpodobnostního modelu.

V práci [20] je popsán způsob navigace ve vnitřním prostředí za použití *Bluetooth 2.0 fingerprintingu*, částicového filtru a mobilního telefonu s operačním systémem *Symbian*.

Práce [6] navrhuje metodu lokalizace *PDR* s využitím hrubé sítě *BLE* majáků, kdy se při průchodu v těsné blízkosti majáku určí poloha v daném prostředí.

Většina výše zmíněných prací používá pro lokalizaci metodu *PDR*, kterou vhod-

Kapitola 4

Definice problému

V této kapitole bude popsána problematika navigace uvnitř budov za použití chytrého mobilního telefonu. Obsahem bude také charakteristika prostředí, způsob změny polohy uživatele a způsob držení mobilního zařízení.

4.1 Co je to navigace?

Navigace je proces hledání cesty z jednoho bodu do bodu jiného za pomoci instrukcí, díky kterým se uživatel dostane do cílového bodu. Mezi nejrozšířenější typy navigací patří automobilová navigace, která využívá k určení polohy technologii GPS. Automobilová navigace má za cíl navést řidiče bezpečně a bez potřeby znalosti prostředí z aktuálního bodu do bodu cílového. Podobným způsobem by měly fungovat i ostatní druhy navigace. Aby proces navigace mohl fungovat správně, je potřeba znát aktuální polohu navigovaného a znát podobu prostředí. Struktura prostředí je zachycena v mapových podkladech, jež obsahují všechny potřebné informace k nalezení cesty v daném prostředí. Mapové podklady tedy musí mít definované cesty, po kterých je možné se pohybovat a/nebo překážky, kterým je nutné se vyhnout. Aby navigace umožňovala navigovat uživatele na konkrétní místo, aniž by uživatel znal konkrétní souřadnice cílového bodu, musí mapové podklady obsahovat seznam význačných bodů. Význačný bod obsahuje souřadnice a název. Tyto body mohou být například adresy nebo tzv. body zájmu (*PointofInterest (POI)*).

Na základě těchto požadavků bude v práci definován navigační proces a mapové podklady umožňující zachytit podobu vnitřního prostředí společně s množinou význačných bodů.

4.2 Navigační proces

Navigační proces se skládá z několika částí. První částí je určení polohy (lokalizace) navigované osoby v daném prostředí. Poté následuje proces nalezení cesty z aktuální polohy uživatele do cílového bodu na základě mapových podkladů. V průběhu navigačního procesu je nutné uživateli poskytovat instrukce, jaké další akce má provést, aby se dostal k cílovému bodu. Tyto dva procesy se opakují do té doby, než uživatel dosáhne cílového bodu. Pro návrh konkrétní implementace navigačního

procesu je nutné vzít v potaz charakteristiku navigačního prostředí, způsob pohybu uživatele, technologické možnosti a způsob držení navigačního zařízení.

Dle popisu obecného navigačního procesu je potřeba v práci definovat metodu lokalizace použitelnou v prostředí definovaném níže.

4.3 Prostředí

Druhy navigace lze rozdělit dle prostředí na dvě základní skupiny. Na navigaci ve venkovním prostředí (námořní, letecká, pozemní, atp.) a na navigaci ve vnitřních (uzavřených) prostorách - budovách. Obě výše zmíněná prostředí se od sebe značně liší a nelze tak bez určitých modifikací a optimalizací použít standardní navigační techniky z jednoho prostředí ve druhém a naopak. Budova je prostředí uzavřené, fyzicky vymezené stěnami a jinými fyzickými objekty. Rozloha vnitřního prostředí je tedy limitována konstrukčním řešením budovy. Konkrétními prostředími pro využití navigace uvnitř budov jsou školy, nemocnice a muzea.

Tato práce se omezuje na vnitřní prostředí.

4.4 Lokalizace

Lokalizace je proces určení polohy, která je relativní vůči určité vztažné soustavě. V případě lokalizace ve vnějším prostředí za pomoci *GPS* navigace je vztažnou soustavou Země a souřadnice jsou zeměpisná šířka a délka.

4.4.1 Problém lokalizace dle GNSS

Dle kapitoly 2 většina chytrých mobilních telefonů obsahuje čipy pro *Global Navigation Satellite System (GNSS)*, které se běžně používají pro lokalizaci uživatele ve vnějším prostředí. Tyto čipy dosahují akceptovatelné přesnosti zaměření polohy ve vnějším prostředí, ale pro použití dle kapitoly 2 vyžadují přímou viditelnost na družice satelitního systému, což je díky konstrukčnímu řešení budov téměř nemožné. Absencí přímé viditelnosti družic klesá přesnost navigačního systému až na úroveň, kdy není lokalizace uživatele technologicky možná.

Nedostupnost přímé viditelnosti družic je hlavním problémem, proč není možné technologii lokalizace dle *GNSS* použít k určení polohy uvnitř budovy a je nutné navrhnout alternativní metodu za využití jiných senzorů a technologií dostupných v mobilních telefonech.

4.4.2 Magnetická rušení

Díky konstrukčnímu řešení budov je prostředí obklopeno různými předměty z různých materiálů (železné stropní traverzy, zděné stěny, dřevěné podlahy, nábytek, atp.), kde tyto objekty mohou absorbovat, odrážet či jinak deformovat různá elektromagnetická vlnění. Vliv na šíření elektromagnetického vlnění ve vnitřním prostředí mají také osoby [21], které se mohou v daném prostředí vyskytovat. Nejvýraznějším zdrojem elektromagnetického vlnění jsou různá elektronická zařízení a elektroinstalace. Tato

V této práci bude problém přítomnosti elektromagnetického rušení v navigačním prostředí brán v potaz při návrhu robustní navigační metody.

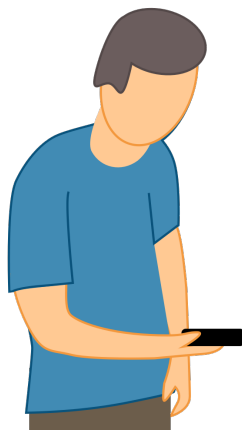
4.5 Změna polohy uživatele

Pro návrh lokalizační a navigační metody je nutné definovat způsob změny polohy uživatele. Jediná možnost změny polohy osoby pro tuto práci bude běžná chůze a jiný způsob přesunu uživatele nebude přípustný. Lidská chůze je specifický způsob pohybu, který se u každého lidského jedince nepatrně liší a je tedy nutné v najít společné rysy a celkovou charakteristiku chůze. Dle kapitoly 2 jsou chytré mobilní telefony vybaveny množstvím senzorů pohybu a orientace, jimiž je možné pohyb uživatele registrovat.

Tato práce si klade za cíl tuto změnu polohy detekovat.

4.6 Umístění navigačního zařízení

Pro tuto práci se předpokládá, že navigovaná osoba je osoba, která nosí navigační zařízení s sebou. Jelikož zdrojem informací bude obrazovka mobilního telefonu, předpokládá se, že uživatel drží mobilní telefon v pravé či levé ruce v pozici před sebou (viz obrázek 4.1), aby viděl na displej zařízení. Při návrhu navigačního algoritmu bude brán zřetel na způsob držení mobilního telefonu při pohybu uživatele.



Obrázek 4.1: Umístění navigačního zařízení

■ 4.7 Hledání cesty

Součástí navigačního procesu je hledání cesty z aktuální polohy do polohy cílové. Tato práce bude řešit problém nalezení cesty ve vnitřním prostředí.

■ 4.7.1 Požadovaná přesnost navigační metody

Aby bylo možné prohlásit navrženou navigační metodu za použitelnou, je nutné stanovit požadovanou přesnost, které by měla navržená metoda dosahovat. Požadovanou přesností navigační metody definuji odchylku tří metrů od skutečné polohy. V případě vyšší odchylky roste pravděpodobnost, že uživatel nenalezne hledaný cíl a navigace bude pro jeho použití nepoužitelná. Odchylkou je myšlena Euklidovská vzdálenost mezi odhadnutou a skutečnou polohou.

■ 4.7.2 Omezení navigační metody

Za účelem relaxace problému hledání polohy uživatele ve vnitřním prostředí se tato práce bude omezovat pouze na navigaci a lokalizaci v rovině namísto v třídimenzionálním prostoru. Nadmořská výška uživatele nebude tedy brána v potaz a bude se předpokládat, že se uživatel pohybuje pouze na jednom podlaží.

Kapitola 5

Návrh řešení

Tato kapitola bude navrhovat strukturu mapových podkladů a grafické rozhraní aplikace pro jejich tvorbu. Dále tato kapitola bude obsahovat návrh navigační metody včetně vlastního krokoměru a kompasu. Pro použití navigační metody zde bude navržena struktura a grafická podoba navigační aplikace.

5.1 Mapové podklady

Tato sekce bude definovat strukturu a obsah mapových podkladů. Dále zde bude popsán návrh aplikace pro vytváření mapových podkladů na mobilních zařízeních.

5.1.1 Specifikace požadavků

Jak bylo zmíněno v kapitole 4, pro funkční navigační proces je potřeba poskytnout vhodné mapové podklady. Mapové podklady budou splňovat následující požadavky:

1. Mapové podklady musí mít jednotný souřadnicový systém.
2. Mapové podklady musí obsahovat překážky zaznamenané v programově čitelném formátu, aby bylo možné v mapových podkladech hledat cestu z jednoho bodu do druhého bodu.
3. Mapové podklady musí obsahovat body zájmu, které uživatel může použít za účelem navigace na místo, kde se bod zájmu nachází.
4. Mapové podklady musí být možné jednoduše rozšířit o dodatečné informace potřebné pro konkrétní metodu určení polohy uživatele (např. referenční body).
5. Mapové podklady musí být platformě nezávislé - přenositelné mezi různými zařízeními a platformami.
6. Vizuální (grafická) stránka mapových podkladů musí být intuitivně interpretovatelná a musí charakterizovat reálné prostředí.

■ 5.1.2 Návrh struktury mapových podkladů

V této sekci bude definována struktura a obsah mapových podkladů na základě požadavků ze sekce 5.1.1.

Mapové podklady se skládají ze dvou hlavních částí. Z vizuální části, která je viditelná pro uživatele ve formě mapy, a z informační (datové) části, kterou využívá navigační aplikace k určení polohy uživatele, nalezení cesty a k poskytování navigačních instrukcí. Jako základ pro vizuální část jsem se rozhodl použít rastrovou grafiku, která bude vizuálně reprezentovat podobu vnitřního prostředí a bude vykreslována uživateli. Toto rozhodnutí jsem udělal na základě faktu, že každá budova musí mít stavební plány, ve kterých je přesně definována struktura a rozměry daných prostor, tudíž je možné dané plány dodat v rastrovém formátu (export, sken, fotografie, ...). Dalším argumentem, proč zvolit rastrový soubor a nepožadovat po uživateli, aby kreslil prostředí na mobilním zařízení, je poměr úsilí potřebného k přesnému nakreslení celého plánu budovy vůči přínosu vektorové grafiky. Tato přesná rastrová grafika bude použita jako podklad pro definování překážek v daném prostředí. Bez použití tohoto grafického podkladu by bylo velmi komplikované a nepohodlné definovat vizuální podobu prostředí a dané překážky vzhledem k omezením, které vyplývají z konstrukčního řešení mobilního telefonu, zejména z malé velikosti obrazovky telefonu, chybějícím vstupním zařízením (myš a klávesnice) a způsobu ovládání dotykem.

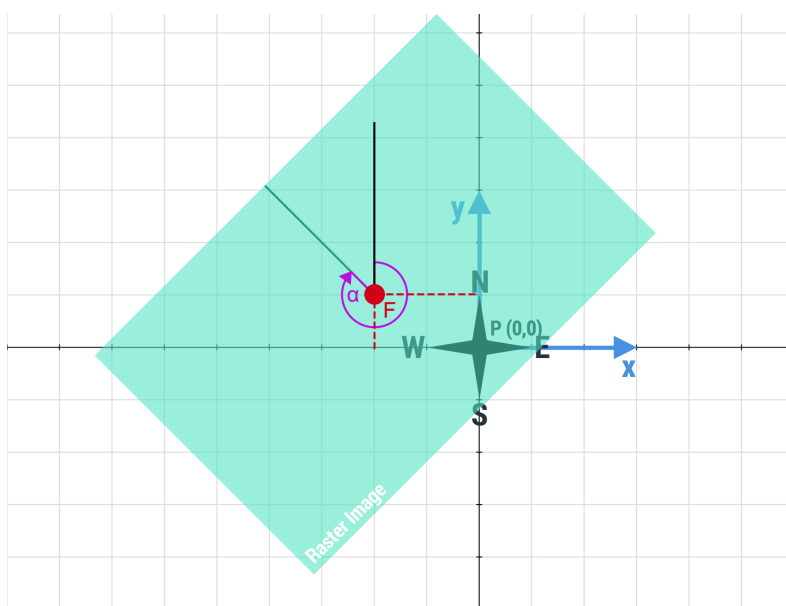
Datovou část jsem rozdělil na čtyři pod-části. První pod-části jsou základní informace o mapových podkladech (název a číslo podlaží), které jsou nutné k rozlišení jednotlivých podkladů v rámci jedné aplikace. Druhou částí je množina překážek (stěny, vybavení místností, atp.) ve formě polygonů a lomených čar, které vymezují prostor, ve kterém je možné se pohybovat a ve kterém lze hledat cestu. Třetí částí je množina bodů zájmu (*POI*) ve formě datových objektů, které obsahují název bodu zájmu pro možnost zobrazení seznamu bodů zájmu uživateli, kategorii bodu zájmu (například výtah, východ, recepce, ...) a souřadnice bodu v souřadném systému mapových podkladů. Poslední datovou částí je struktura obsahující měření hodnot bezdrátových majáků, potřebná pro metodu určení polohy uživatele v rámci prostředí. Všechny výše zmíněné části musí být definovány v jednotném souřadném systému, aby pozice a rozměry objektů z jednotlivých datových částí odpovídaly pozicím a rozměrům příslušných objektů v grafické části. Proto následující podsekcce definuje podobu jednotného souřadného systému použitého v navržené podobě mapových podkladů.

■ Souřadný systém mapového podkladu

Jelikož se jedná o mapové podklady budov, lze předpokládat, že mapové podklady budou mapovat oblast v okolí menším než několik stovek metrů. Na základě tohoto předpokladu lze zanedbat zakřivení země a lze provést aproximaci prostředí horizontální rovinou. Proto jsem soustavou souřadnic zvolil rovinnou kartézskou soustavu souřadnic, ke které jsou vztaženy všechny části mapového podkladu, což zajišťuje zachování pozice a rozměrů mezi jednotlivými vrstvami a objekty. Dle tradičního použití kartézské soustavy budou osy označeny písmeny x , y a počátek P bude definován v bodě $(0, 0)$ viz obrázek 5.1. Kde kladný růst hodnot na ose x

odpovídá pohybu na východ, pokles odpovídá pohybu na západ. Logicky kolmo k ose x odpovídá růst hodnot na ose y pohybu směrem na sever a pokles hodnot odpovídá pohybu směrem na jih. Základní jednotkou tohoto souřadného systému bude 1cm .

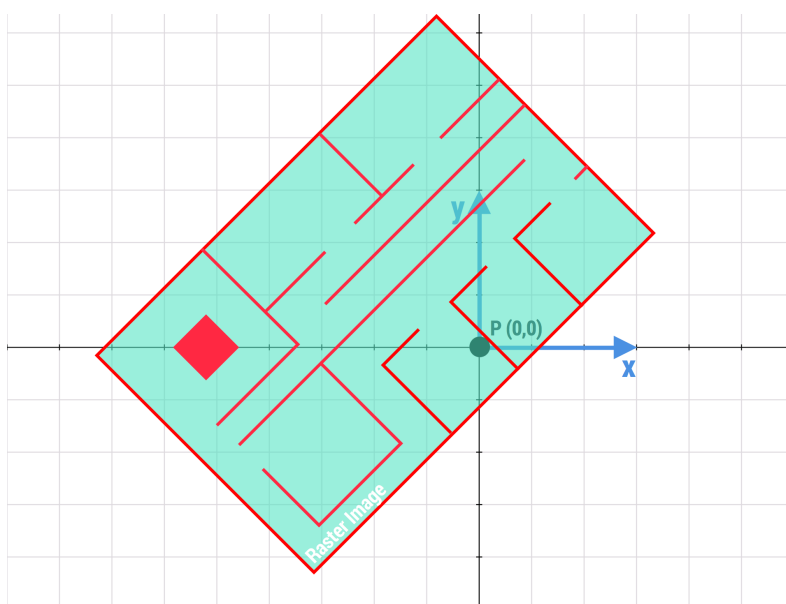
Relativně k tomuto souřadnému systému bude umístěna grafická část (dále také grafická vrstva), jejíž umístění vůči počátku souřadného systému bude dáno bodem posunutí F a úhlem natočení α (viz obrázek 5.1). Bod F odpovídá geometrickému středu rastrové grafiky. Úhel α je rotace rastrové grafiky v bodě F ve směru hodinových ručiček. Rotací rastrové grafiky musí být zajištěno, aby při pohybu směrem na sever (tzn. v kladném směru osy y) odpovídal směr pohybu směru pohybu v grafickém podkladu. Posunutí bodu F vůči počátku umožňuje umístit počátek souřadného systému do libovolné části grafické vrstvy.



Obrázek 5.1: Souřadný systém mapových podkladů a umístění grafického podkladu

■ Množina překážek

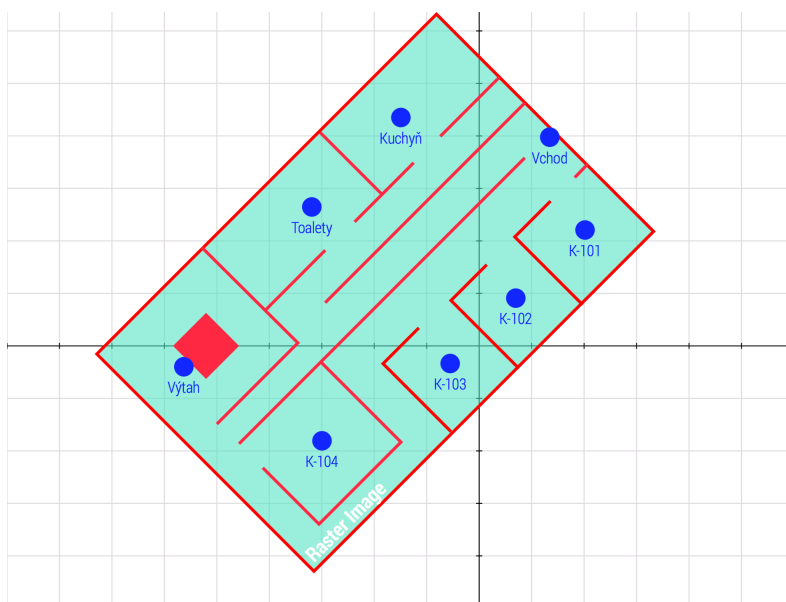
Obrázek 5.2 zobrazuje mapové podklady včetně vrstvy překážek, které jsou definovány červeně zbarvenými polygony a lomenými čarami. Polygony a lomené čáry jsou tvořeny množinou bodů, které po spojení vytvoří výsledný tvar a vymezí tak prostor, ve kterém je možné se pohybovat. Viditelnost této vrstvy je doporučena pouze pro režim vytváření mapových podkladů. Navigovaný uživatel by měl vidět pouze grafickou vrstvu (rastrovou grafiku).



Obrázek 5.2: Grafický podklad a množina překážek

■ Množina bodů zájmu

Obrázek 5.3 zobrazuje mapové podklady včetně vrstvy překážek a včetně zakreslené množiny bodů zájmu, které jsou v obrázku vyznačeny modře zbarvenými body. Body zájmu je možné umístit do libovolné dosažitelné části mapových podkladů. Pokud bude do podkladů zanesen bod zájmu, do kterého díky definici překážek nelze nalézt cestu, nebude možné uživatele na takové místo navigovat.



Obrázek 5.3: Grafický podklad, množina překážek a body zájmu

■ Referenční body pro bezdrátové majáky

Struktura referenčních bodů pro bezdrátové majáky bude navržena dle navržené lokalizační metody.

■ Formát mapových podkladů

Na základě požadavků na mapové podklady ze sekce 5.1.1 a na základě výše uvedené podoby mapových podkladů jsem navrhl vlastní formát mapových podkladů postavený na datovém formátu *JavaScriptObjectNotation (JSON)* s vlastní strukturou. Tento způsob reprezentace je vhodný zejména díky své platformní nezávislosti (numerické hodnoty jsou reprezentovány v textovém formátu), univerzálnosti (vlastní struktura), čitelnosti (lze otevřít v libovolném textovém editoru) a snadné rozšiřitelnosti (rozšíření struktury o další strukturu). Dále je možné takto definované mapové podklady snadno distribuovat i pro *offline* použití. Strukturu tohoto formátu lze nalézt v přílohách v tabulce C.1.

■ Proces vytvoření mapových podkladů

Tato sekce popisuje posloupnost akcí potřebných k vytvoření mapových podkladů specifikovaných v předchozích sekcích. Na základě tohoto procesu bude dále navrženo prostředí mobilní aplikace pro vytváření mapových podkladů.

1. Vložení názvu mapových podkladů - textový řetězec
2. Vložení čísla podlaží - celé číslo
3. Vložení rastrového obrázku - obrázek ve formátu *JPEG* nebo *PNG*
4. Výběr měřítka rastrového obrázku - kolik pixelů v obrázku je 100cm v reálném prostředí
5. Výběr počátku souřadného systému v rámci rastrového obrázku
6. Nastavení rotace grafického podkladu vůči severu tak, aby kladná část osy *y* směřovala na sever
7. Zakreslení překážek prostřednictvím polygonů a lomených čar
8. Vyznačení bodů zájmu
9. Vyznačení referenčních bodů (*WiFi AP*, *BLE* zařízení)
10. Naměření hodnot signálu z bezdrátových majáků v prostoru
11. Uložení vytvořeného mapového podkladu

■ 5.2 Návrh uživatelského rozhraní

Návrh uživatelského rozhraní (*UI*) je komplexní proces, jehož cílem je, aby navržené rozhraní bylo intuitivní, snadné na ovládání a mělo by minimalizovat možnost vzniku chyb. Chybou je míněno, že aplikace provede odlišnou akci, než uživatel očekával. Dále by navržené rozhraní mělo respektovat tzv. *design guidelines* pro danou platformu. *Design guidelines* je sada pravidel definujících, jaký by měly jednotlivé prvky uživatelského rozhraní mít vzhled a jaká omezení a doporučení co do rozmístění a velikosti by měly splňovat. Tato pravidla je důležité dodržovat zejména při návrhu UI pro mobilní telefony, jelikož mobilní telefony disponují podstatně menší obrazovkou než stolní počítače či notebooky a ovládání si proto vyžaduje větší požadavky na přesnost při interakci s prvky uživatelského rozhraní. Další důležitou rolí *design guidelines* je snaha o sjednocení způsobu ovládání velkého množství aplikací napříč celou platformou. Mobilní aplikace pro tuto práci budou postavené na platformě *Google Android* (volba platformy je zmíněna v kapitole 6), tudíž při návrhu uživatelského rozhraní budu dodržovat *Android Design Guidelines* [22].

Návrh uživatelského rozhraní začíná vymezením, pro jakou cílovou skupinu uživatelů je aplikace určena. Definováním cílové skupiny uživatelů lze odhadnout, jakým způsobem uživatel bude aplikaci ovládat a jaké jsou jeho teoretické znalosti a schopnosti. Jedním ze způsobů specifikace cílové skupiny je modelování osoby. *Persona* je model smyšlené osoby, která danou cílovou skupinu vystihuje a lze z její charakteristiky odhadnout, jak se v různých situacích při používání aplikace zachová. Model osoby může být výstupem průzkumu cílové skupiny nebo může být *persona* definována s ohledem na požadavky na uživatele ze strany aplikace/produktu.

Po vytvoření osoby následuje proces návrhu obrazovek aplikace, tento proces se nazývá *UI prototyping*. *Prototyping* začíná na nejnižší úrovni grafického návrhu, který bere v potaz pouze strukturu aplikace, umístění ovládacích prvků a jejich funkcionalitu. Tyto grafické návrhy nejnižší úrovně se nazývají drátové modely (anglicky *wireframes*). Podoba drátových modelů se během návrhu velmi často mění, tudíž je vhodné použít specializovaný software pro návrh *wireframes*, kde lze za běhu relativně rychle měnit jejich podobu. Další tradiční možností modelování *wireframes* je skládání obrazovek z vystřižených částí papíru s nakreslenými ovládacími prvky, které lze rapidně měnit. Po navržení drátových modelů, čímž je určeno rozložení ovládacích prvků, je možné začít navrhovat *mockupy*. *Mockup* je stejně jako *wireframe* grafický návrh obrazovky obsahující prvky uživatelského prostředí. *Mockupy* se již nezaměřují na rozvržení jednotlivých ovládacích prvků, ale na jejich vizuální podobu. Tudíž *mockupy* mají daleko blíže k finálnímu vzhledu a v praxi slouží tedy jako grafický návrh dané obrazovky. *Mockupy* lze vytvářet ve specializovaných nástrojích na návrhy UI, ale také v libovolném grafickém programu. Níže přiložené drátové modely i *mockupy* jsou nakresleny v aplikaci *Sketch* od společnosti *Bohemian Coding* ve verzi 3.7.1.

■ 5.2.1 Persona

Mobilní aplikace pro vytváření mapových podkladů je aplikace, jejíž použití je omezené pro úzkou skupinu uživatelů zaměřující se instalací navigačního systému

do vnitřních prostor. Instalace navigačního systému vyžaduje znalost problematiky navigačního systému, tudíž cílovou skupinou jsou uživatelé se znalostí této problematiky. Na základě této definice cílové skupiny byla vytvořena následující persona charakterizující fiktivního uživatele.

Petr, muž, 25 let

Informační technik - zaměstnanec IT firmy zabývající se návrhy a realizací elektrických, LAN a WLAN sítí. Realizuje instalaci bezdrátových sítí. Vystudoval České vysoké učení technické v Praze, obor Otevřená informatika. Mezi jeho zájmy patří moderní technologie, fotografování a sport.

5.2.2 Struktura obrazovek

V této podsekci bude popsána struktura uživatelského prostředí celé aplikace pro návrh mapových podkladů (plánů budov), jeho vizuální a funkční podoba. Dále zde bude popsán vztah mezi jednotlivými obrazovkami.

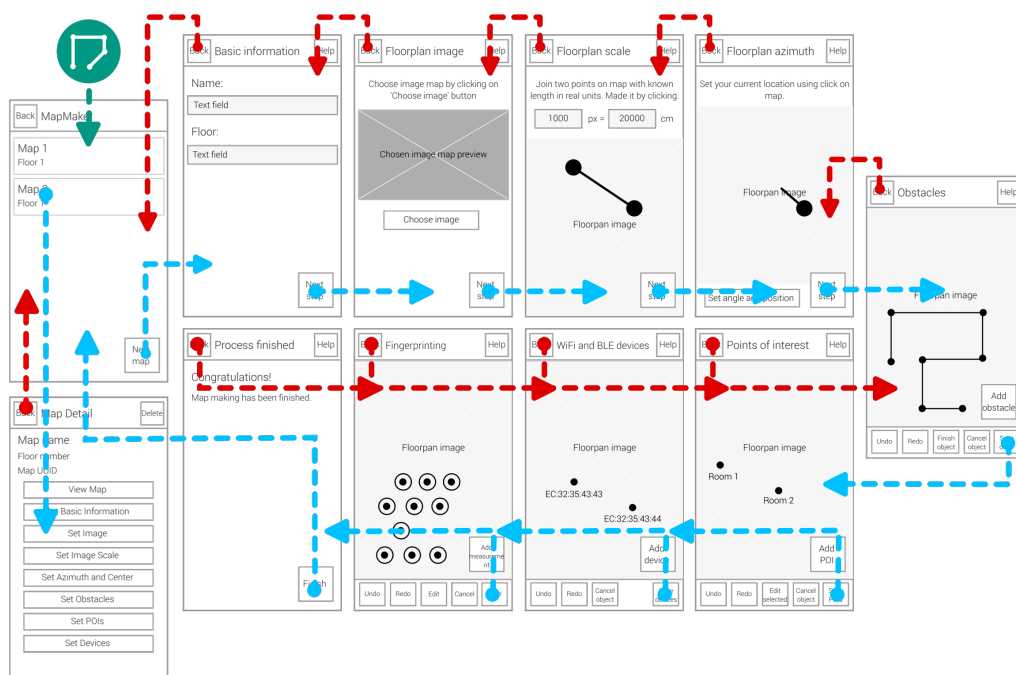
Nejrozsáhlejší částí aplikace pro vytváření mapových podkladů bude proces vytvoření podkladů. Tento proces se skládá z několika kroků, které je nutné provést ve správném pořadí a nad správnými daty. Je nutné tedy, aby uživatelské prostředí minimalizovalo možnost vložení nesprávných dat a naopak, aby maximalizovalo snadnost ovládání. Správné pořadí jednotlivých činností lze zajistit vhodným rozdělením obrazovek, které zajistí, že chronologické uspořádání obrazovek bude odpovídat požadovanému pořadí daných činností. Minimalizaci možnosti vložení nesprávných dat a minimalizaci možnosti provedení činnosti nevhodným způsobem lze zajistit účelnou instruktáží uživatele s informacemi, jaká data se od něj očekávají, v jakém tvaru a jakým způsobem je získat. Dále je nutné vložená data validovat a v případě nesplnění validačních omezení uživatele vhodným způsobem upozornit.

Rozdělení obrazovek jsem navrhl dle navrženého procesu vytváření mapových podkladů ze sekce 5.1.2. Hlavní obrazovkou je seznam již vytvořených mapových podkladů, který lze rozšířit vytvořením nových mapových podkladů. Jednotlivé mapové podklady musí být možné upravovat i po vytvoření, proto za tímto účelem byla do aplikace zařazena obrazovka "Detail", která se otevře po kliknutí na vytvořený mapový podklad. Pro účel vytvoření mapového podkladu jsem navrhl sadu obrazovek, které na sebe v pevně definovaném pořadí navazují a vytváří tak intuitivního průvodce procesem. Tato sada obrazovek je určena jednotlivými kroky procesu tvorby plánu definovaných v sekci 5.1.2.

- Prvním krokem je poskytnutí základních informací o mapovém plánu.
- Druhým krokem je vložení rastrové grafiky obsahující grafický plán podlaží, pro který je nutné ve
- třetím kroku nastavit měřítko respektující reálné rozměry objektů v grafickém plánu.
- Čtvrtým krokem je nastavení počátku souřadného systému a určení rotace mapového podkladu vůči magnetickému severu.

- Pátým krokem je obrazovka, ve které je nutné vyznačit všechny překážky a definovat fyzickou strukturu prostředí, aby bylo možné později uživatele v mapových podkladech navigovat.
- Šestým krokem je umístění bodů zájmů do mapového plánu pod příslušným názvem a kategorií.
- Sedmý krok je obrazovka pro umístění bezdrátových zařízení jako jsou *Wi-Fi AP* a *Bluetooth Low Energy* zařízení (*iBeacon*, *EddyStone*) rozmístěné v mapovaném prostředí. Poslední obrazovkou je minimalistická obrazovka informující o dokončení procesu vytváření plánu podlaží.
- Osmý krok je samotné naměření množiny referenčních bodů pro bezdrátové majáky.

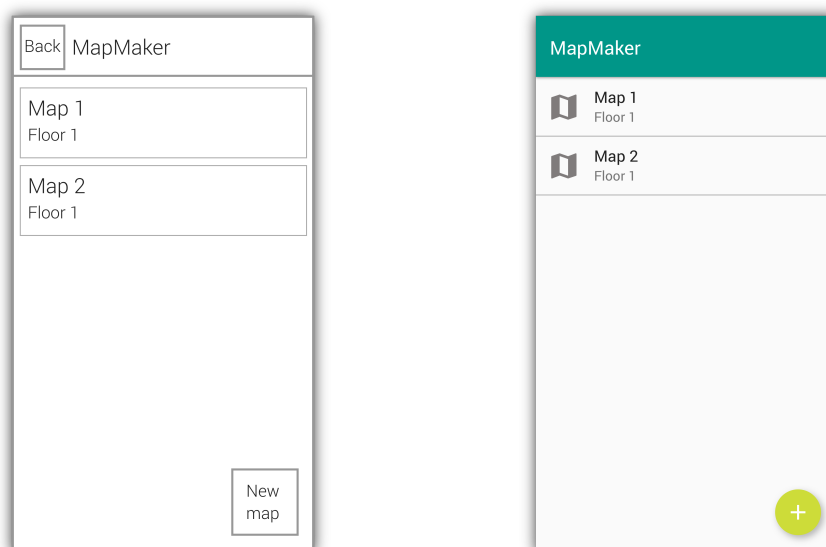
Struktura aplikace a relace mezi obrazovkami je zachycena v obrázku 5.4. Šipky charakterizují přechod mezi jednotlivými obrazovkami. Červeně zbarvené šipky odpovídají návratu z aktuální obrazovky na předchozí obrazovku. Z důvodu přehlednosti ilustrace neobsahuje diagram relace mezi obrazovkou detailu a jednotlivými obrazovkami pro úpravu dat. Tyto relace budou popsány dále v popisu obrazovky "Detail". Níže budou detailně popsány jednotlivé obrazovky celé aplikace pro vytváření mapových podkladů.



Obrázek 5.4: Relace mezi obrazovkami (*screenflow*)

Hlavní obrazovka

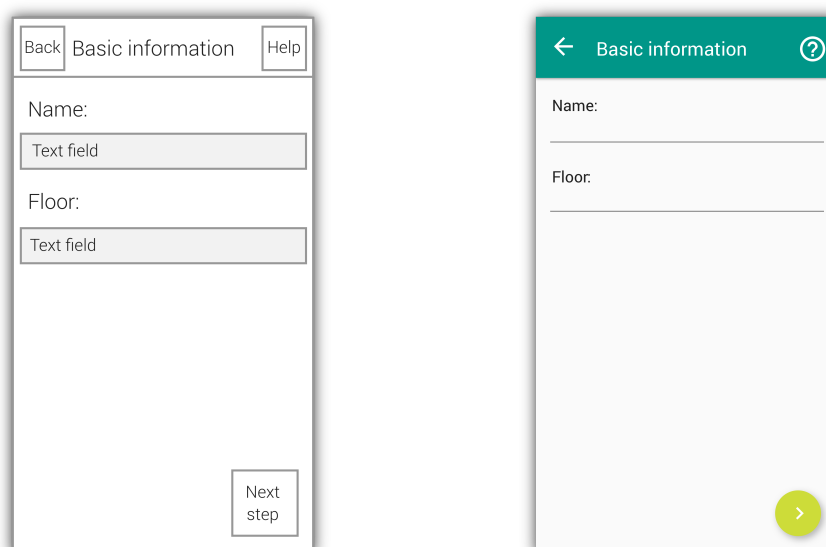
Hlavní obrazovka, jak bylo uvedeno výše, obsahuje pouze seznam existujících mapových podkladů. Jednotlivé položky seznamu obsahují základní informace o plánu, jako je jeho název a číslo podlaží. Dále hlavní obrazovka obsahuje akční tlačítko pro zahájení procesu tvorby nového plánu. Vzhled a rozmístění jednotlivých prvků respektuje *design guidelines* pro platformu *Android* [22] (návrh uživatelského rozhraní byl vytvořen až o volbě cílové platformy). Obrázek 5.5 zachycuje grafickou podobu drátového modelu obrazovky, dle kterého byl navržen *mockup*. Po kliknutí na akční tlačítko pro přidání nového plánu se otevře obrazovka pro vložení základních informací.



Obrázek 5.5: Hlavní obrazovka (*wireframe*, *mockup*)

Obrazovka krok 1 - základní informace

První obrazovka (viz 5.6) tvorby nového plánu obsahuje dvě vstupní pole. Tlačítko pro nápovědu a akční tlačítko pro pokračování. Prvním vstupním polem je textové pole pro vložení názvu nového plánu. Druhým vstupním polem je opět textové pole očekávající celé číslo, které odpovídá číslu podlaží vytvářeného plánu. Obrazovka dále obsahuje tlačítko pro nápovědu s tradiční ikonou otazníku. Po kliknutí na tlačítko nápovědy se zobrazí vyskakovací dialog s instrukcemi, jaké informace má uživatel vyplnit do vstupních polí vyplnit. Po kliknutí na tlačítko pro přechod na následující krok se provede validace dat vstupních polí a pokud je obsah validní, přejde aplikace na následující obrazovku. V opačném případě aplikace informuje uživatele o tom, že vložená data nejsou správná.

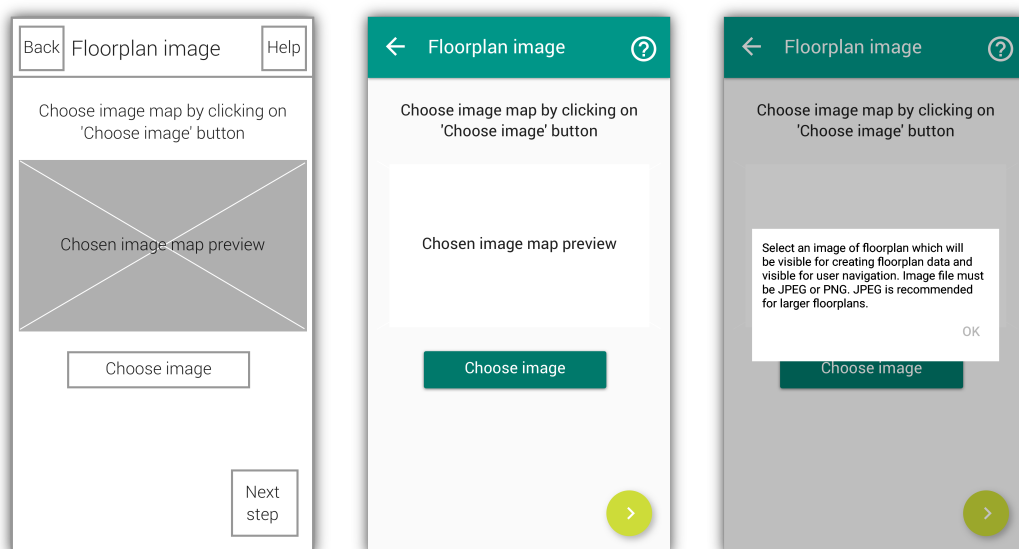
Obrázek 5.6: Krok 1 (*wireframe, mockup*)

■ Obrazovka krok 2 - nahrání grafického podkladu

Druhým krokem procesu je nahrání grafického podkladu plánu ve formě rastrového obrázku, který je uložen v paměti telefonu. Ihned po otevření obrazovky (obrázek 5.7) jsou uživateli zobrazeny instrukce prostřednictvím vyskakovacího dialogu. Zobrazením dialogu s vhodnými instrukcemi snížíme šanci, že uživatel nahraje nevhodný obsah, jelikož je mu sdělena informace, že očekávaný obsah je plán podlaží. Po zavření dialogu je uživateli umožněno kliknutím na tlačítko vybrat obrázek ve formátu *JPEG* nebo *PNG*. Po vybrání příslušného obrázku je možné přejít na další obrazovku kliknutím na plovoucí tlačítko. V případě, že uživatel nenahrál rastrový plán podlaží, nebude možné přejít na následující obrazovku.

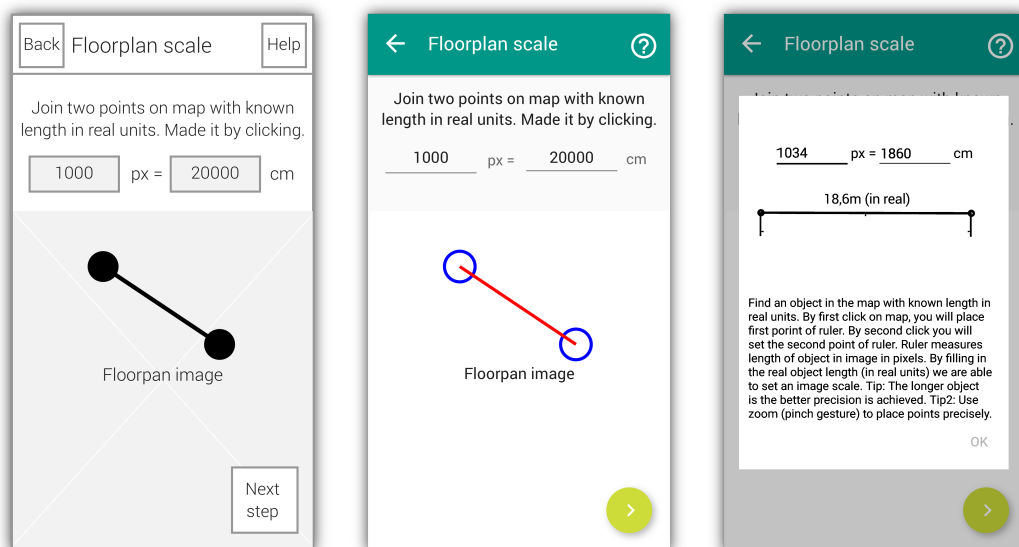
■ Obrazovka krok 3 - nastavení měřítka

Třetí obrazovkou (obrázek 5.8) procesu je obrazovka pro nastavení měřítka nahraného obrázku. Aby odpovídaly velikosti a souřadnice v grafickém podkladu skutečnosti, je nutné nastavit, v jakém poměru jsou jednotky obrázku (*pixels*) vůči reálnému prostředí. Jelikož se opět nejedná o triviální krok, je ihned po otevření obrazovky zobrazen dialog s podrobnými instrukcemi. Uživatel má na výběr ze dvou možností, jak nastavit správné měřítko. V případě, že je uživateli znám poměr obrazových bodů (*pixelů*) obrázku vůči reálnému prostředí, může přímo vyplnit dané hodnoty do dvou vstupních polí. V případě, že uživatel nemá tyto informace dostupné, jednoduše dvěma body v obrázku vyznačí objekt (např. stěnu), u kterého zná délku v reálných jednotkách. Vyznačení bodů provede kliknutím na mapovou komponentu. První kliknutí vyznačí první bod, druhé kliknutí bod druhý. Při lichém kliknutí se vyznačí první bod a při sudém kliknutí bod druhý. Mapová komponenta musí umožňovat uživateli mapu posouvat a měnit přiblížení tradičními gesty. Poté, co jsou hodnoty vyplněny a uživatel klikne na tlačítko pro přechod na následující obrazovku, provede



Obrázek 5.7: Krok 2 (wireframe, mockup, nápověda)

program validaci dat a v případě úspěchu pokračuje na čtvrtý krok.

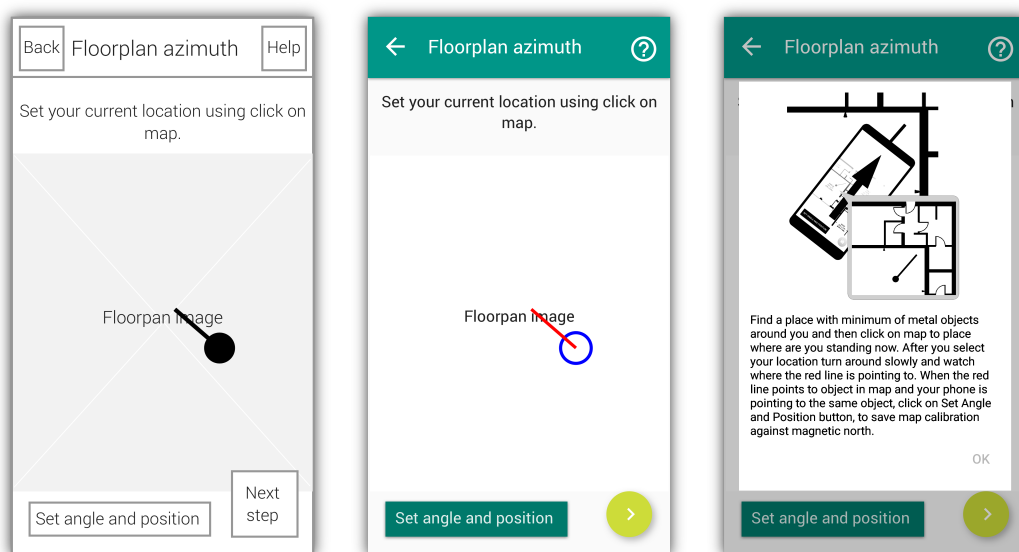


Obrázek 5.8: Krok 3 (wireframe, mockup, nápověda)

■ Obrazovka krok 4 - kalibrace rotace a počátku

Tato obrazovka (obrázek 5.9) má za cíl co nejvíce usnadnit kalibraci rotace grafického podkladu vůči magnetickému severu. Po otevření se opět zobrazí instrukce s grafickou ilustrací. Po uživateli se vyžaduje, aby kliknutím do mapové komponenty vyznačil místo, na kterém se fyzicky nachází. Vybrané místo by mělo být tam, kde se v nejbližším okolí nenachází žádné objekty rušící geomagnetické pole, protože je

nutné, aby byl magnetický sever zaměřen správně. Poté, co uživatel označí místo, kde se právě nachází, se bude na obrazovce otáčet šipka určující směr magnetického severu v závislosti na otočení uživatele vůči okolnímu prostředí. Uživatel pomocí plynulého otáčení na místě kolem své vertikální osy sleduje, jakým směrem šipka v mapě ukazuje. Pokud šipka v mapě směřuje na objekt, na který ukazuje uživatel středem telefonu, klikne na tlačítko pro nastavení azimutu a pozice. Tím se nastaví potřebné hodnoty rotace grafických podkladů vůči magnetickému severu a také se nastaví počátek mapových podkladů do tohoto místa. Detekce magnetického severu je realizována kompasem. Po nastavení dat je možné přejít na další obrazovku.

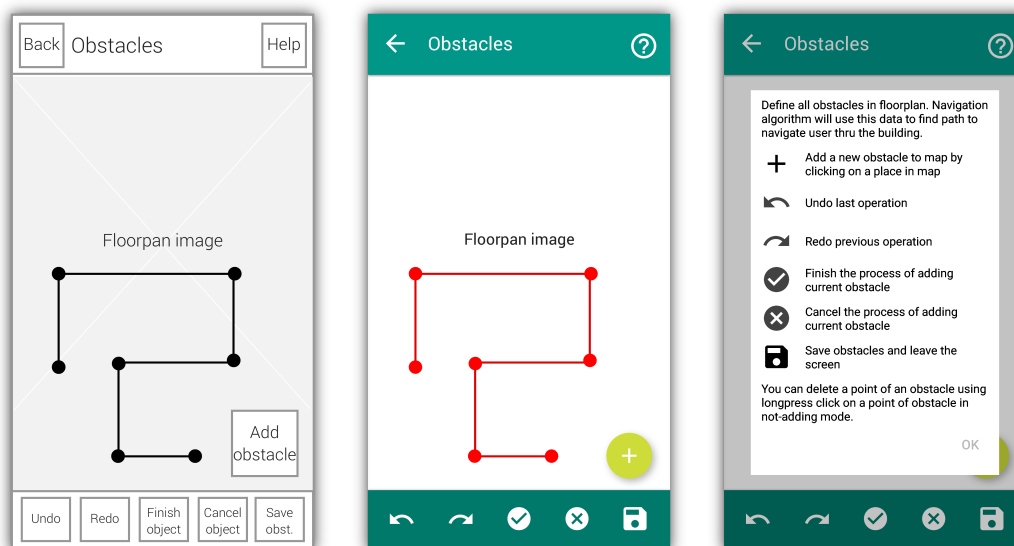


Obrázek 5.9: Krok 4 (wireframe, mockup, nápověda)

■ Obrazovka krok 5 - vyznačení překážek a objektů

Pátým krokem procesu je označení překážek a objektů, přes které není možné vést cestu a kterým je nutné se vyhnout. Mezi tyto překážky patří zejména stěny, sloupky, zábradlí, nábytek atp. Opět se nejedná o triviální krok, proto je uživatel instruován pomocí informačního dialogu. Hlavním prvkem této obrazovky (obrázek 5.10) je mapová komponenta, která již po předchozích krocích zobrazuje grafický podklad ve správném měřítku a se správným posunutím vůči počátku. Dále grafické rozhraní obsahuje spodní lištu nástrojů potřebných k vyznačení objektů. Na této obrazovce je, na rozdíl od předchozích, dominantní akcí přidání dalšího objektu. Z tohoto důvodu se na akčním tlačítku objevuje místo akce přechodu na další obrazovku akce pro přidání objektu. Tato změna byla provedena na základě *design guidelines*. Po kliknutí na tlačítko pro přidání objektu přejde komponenta do režimu přidávání a po kliknutí na libovolné souřadnice v mapové komponentě se přidá bod lomené čáry, která reprezentuje objekt. Opakovaným přidáváním bodů vzniká celý objekt, který lze při vytváření posledního bodu uzavřít, tím že klikne na první bod lomené čáry. Tím se změní lomená čára na uzavřený polygon. Režim přidávání objektu je nutné dokončit kliknutím na tlačítko s ikonou "fajfky" nebo zrušit kliknutím na tlačítko

s ikonou "křížku". Pokud není komponenta v režimu přidávání, je možné jednotlivé body přesouvat tažením prstu z daného bodu na místo jiné. Dlouhým podržením prstu na bodu objektu lze bod odstranit. Provedené akce nad objekty v mapě lze vrátit zpět či opakovat (z angličtiny akce *undo*, *redo*). Poté, co jsou vloženy všechny objekty, je možné kliknutím na ikonu "diskety" uložit vyznačené objekty a přesunout se na obrazovku pro vložení bodů zájmu.



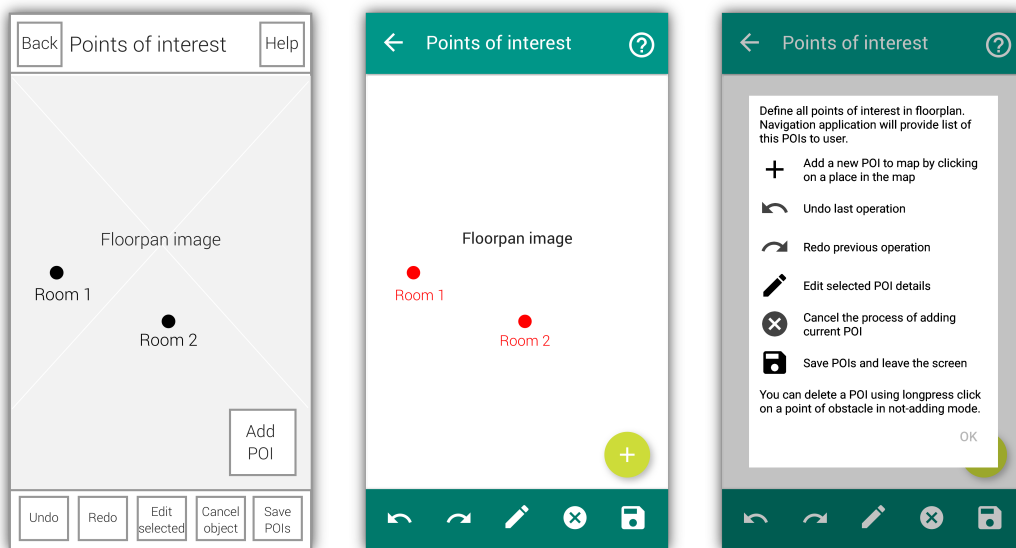
Obrázek 5.10: Krok 5 (wireframe, mockup, nápověda)

■ Obrazovka krok 6 - vyznačení bodů zájmu

Šestá obrazovka (obrázek 5.11) slouží k přidání bodů zájmu, do kterých bude možné uživatele navigovat. Opět se ihned po otevření obrazovky zobrazí dialog s instrukcemi. Prostředí obrazovky je téměř totožné s obrazovkou předchozí, s tím rozdílem že místo překážek se přidávají body zájmu, což jsou doslova body v mapě s přiřazeným názvem a kategorií. Opět lze tyto body mimo režim přidávání přesouvat a odstraňovat. Navíc je zde možné body zájmu editovat. Editace je možná kliknutím na konkrétní bod zájmu. Následným kliknutím na ikonu "tužky" se zobrazí dialog pro editaci názvu a kategorie vybraného bodu zájmu. Kliknutím na ikonu "diskety" dojde k uložení bodů a k přesunu na obrazovku pro přidání bezdrátových zařízení.

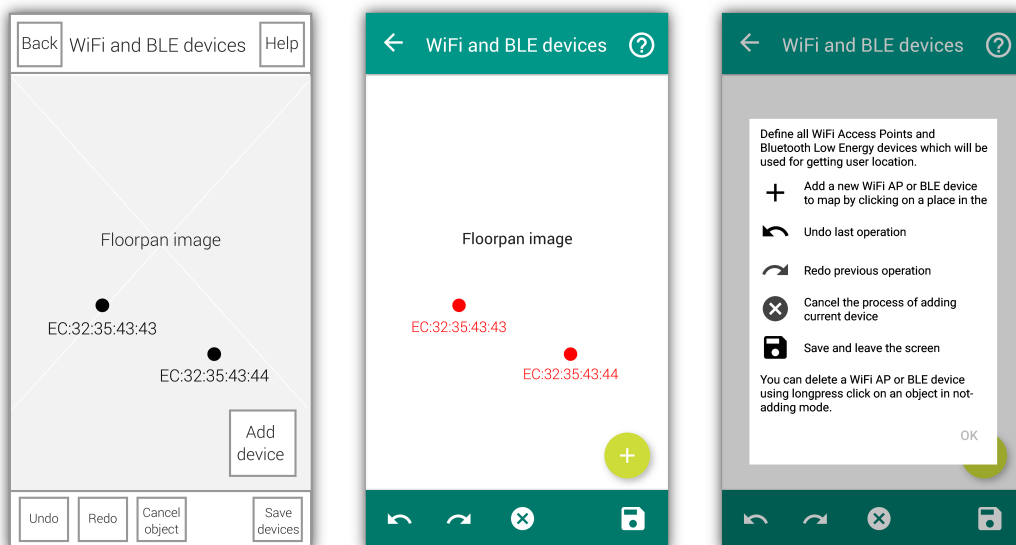
■ Obrazovka krok 7 - přidání bezdrátových zařízení

Sedmá obrazovka (obrázek 5.12) je určena pro rozmístění bezdrátových zařízení typu *Wi-Fi AP* a *Bluetooth Low Edition (iBeacon, EddyStone)* v mapových podkladech. Tato obrazovka po otevření opět instruuje uživatele pomocí dialogového okna. Rozhraní této obrazovky je převzato z obrazovky předchozí. Recyklací stejného uživatelského rozhraní se využívá nabytých zkušeností uživatele z předchozí obrazovky, jelikož si je uživatel vědom, co který ovládací prvek znamená a jakou akci vyvolá. Po přidání bodu do mapy se zobrazí vyskakovací dialog se seznamem všech



Obrázek 5.11: Krok 6 (wireframe, mockup, nápověda)

přístupových bodů *Wi-Fi* a *Bluetooth Low Edition* zařízení v dosahu mobilního telefonu. Výběrem položky v seznamu zařízení se uloží k bodu identifikátor zařízení (definováno dále). Po rozmístění všech potřebných zařízení lze přejít kliknutím na tlačítko uložení na poslední obrazovku.

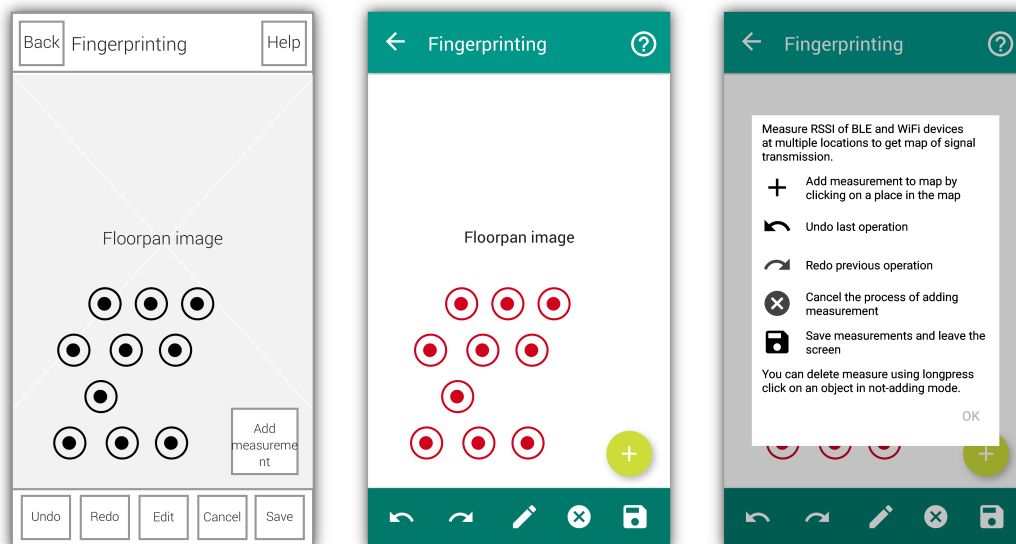


Obrázek 5.12: Krok 7 (wireframe, mockup, nápověda)

■ Obrazovka krok 8 - fingerprinting

Osmá obrazovka (obrázek 5.13) je určena pro provedení *offline fáze fingerprintingu* (definováno dále). Po kliknutí na tlačítko přidat přejde obrazovka do stavu přidávání

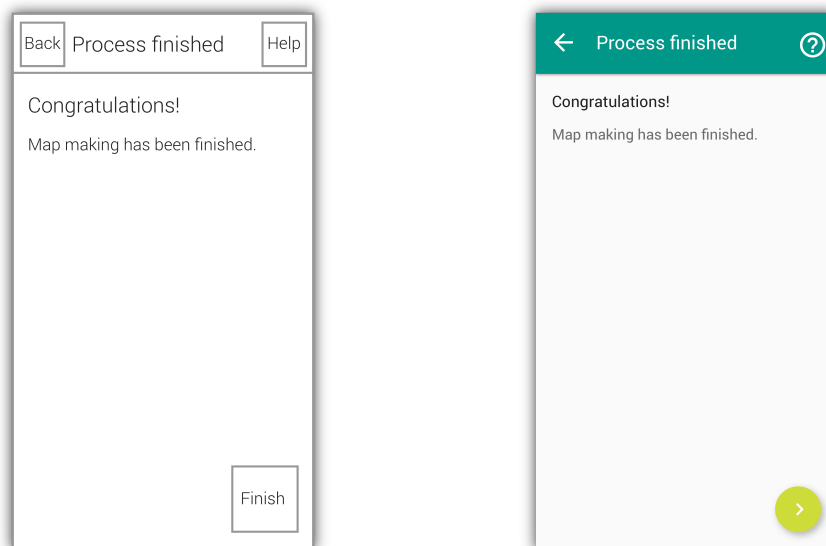
referenčních míst, pro které budou naměřeny přijaté síly signálů z jednotlivých bezdrátových zařízení. Proces měření signálu bude zahájen ihned po vyznačení referenčního místa v mapě. Průběh měření bude doprovázet dialogové okno informující o počtu naměřených vzorků pro dané místo. Po naměření všech potřebných zařízení lze kliknutím na tlačítko pro uložení celý proces tvorby mapového podkladu dokončit.



Obrázek 5.13: Krok 8 (*wireframe*, *mockup*, nápověda)

■ Obrazovka krok 9 - dokončení

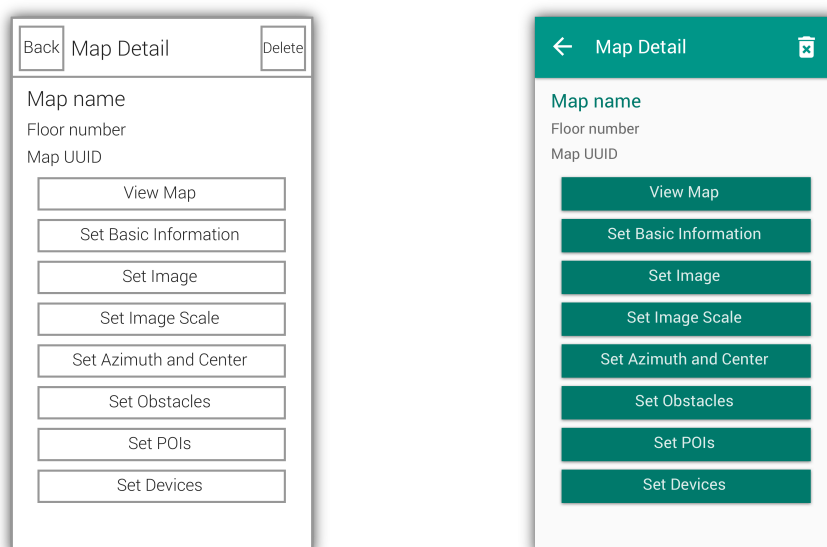
Poslední obrazovka (obrázek 5.14) procesu vytváření mapových podkladů informuje uživatele o úspěšném vytvoření mapových podkladů. Kliknutím na tlačítko pro pokračování aplikace přejde na hlavní obrazovku a zobrazí aktualizovaný seznam mapových podkladů.



Obrázek 5.14: Krok 9 (*wireframe, mockup*)

■ Obrazovka Detail

Obrazovka detailu (obrázek 5.15) poskytuje základní informace o mapovém plánu a slouží k jeho editaci. Dále tato obrazovka umožňuje přejít do jiných aplikací, které mohou daný mapový podklad použít. V případě potřeby je možné mapový plán odstranit prostřednictvím tlačítka s ikonou "popelnice". Odstranění je nevratné, proto je zobrazen potvrzovací dialog. Pro editaci jednotlivých částí mapových podkladů slouží skupina tlačítek uspořádaných v pořadí, v jakém byly informace vkládány při procesu vytváření mapového podkladu. Po kliknutí na příslušné tlačítko se otevře asociovaná obrazovka. Z této obrazovky se lze kliknutím na tlačítko zpět vrátit na obrazovku detailu. Obdobně tlačítko, které sloužilo v procesu vytváření k přechodu na následující krok, uloží změněná data a přejde zpět na obrazovku detailu.



Obrázek 5.15: Detail mapových podkladů (*wireframe, mockup*)

■ 5.3 Lokalizace

V této kapitole bude navržen a popsán algoritmus určení polohy uživatele uvnitř budovy s ohledem na technologické možnosti mobilních zařízení z kapitoly 2. Výsledný algoritmus bude zaměřen zejména na lokalizaci pomocí bezdrátových technologií a senzorů orientace a pohybu.

■ 5.3.1 Bezdrátové technologie

Dle kapitoly 2 jsou mobilní telefony vybaveny čipy určenými pro bezdrátovou komunikaci. Práce [23, 24, 25, 26, 27, 28] potvrzují, že je možné pro lokalizaci uživatele použít *Wi-Fi* sítě. Dle [28, 20, 6, 15] lze využít technologii *Bluetooth Low Edition* za účelem lokalizace obdobným způsobem jako sítě *Wi-Fi*. Lokalizace pomocí těchto dvou bezdrátových technologií je dle [23] možná několika způsoby:

ToA (Time of Arrival) - měření doby od vyslání do přijetí signálu, kde při známé rychlosti šíření signálu, lze odhadnout vzdálenost přijímače od vysílače,

AoA (Angle of Arrival) - měření úhlu dopadu signálu přijatého přijímačem na základě rozdílu času dopadu na jednotlivé antény přijímače, čímž lze odhadnout úhel, pod jakým se vysílač nachází vůči přijímači,

RSS (Received Signal Strength) - měření síly přijatého signálu.

Pro metodu *AoA* je nutné, aby přijímač disponoval několika anténami, na kterých lze přijatý signál zachytit. Pro obě metody (*ToA* a *AoA*) je nutné mít přístup na hardwarovou úroveň čipu pro danou bezdrátovou komunikaci. Jelikož operační systémy mobilních telefonů z bezpečnostních důvodů neposkytují přístup přímo na hardwarovou úroveň, není tedy možné použít způsoby *ToA* a *AoA*. Jediným možným způsobem pro lokalizaci uživatele lze využít informaci o síle přijatého signálu (*RSS*), které poskytují všechny mobilní platformy.

■ RSS

Jak bylo již výše zmíněno, *RSS* je zkratka pro sílu přijatého signálu. Operační systémy tyto informace získávají z *Wi-Fi* a *Bluetooth* čipů a propagují je dále prostřednictvím *Application Programming Interface (API)* pro komunikaci s hardwarem.

Velikost síly přijatého signálu v určité vzdálenosti od vysílače odpovídá v ideálním případě teoretickému modelu propagace daného signálu prostředím. V ideálních podmínkách dle matematické rovnice útlumu síly signálu z [29] s rostoucí vzdáleností mezi přijímačem a vysílačem síla přijímaného signálu klesá. Na základě známých pozic vysílačů a velikostí síly přijatých signálů z jednotlivých vysílačů lze za dostatečného množství vhodně umístěných vysílačů v téměř ideálním prostředí určit polohu přijímače pomocí metody multi-laterace. V praxi ale dochází k útlumu síly signálu v závislosti na struktuře reálného prostředí, tudíž skutečný útlum síly signálu neodpovídá matematickému modelu a nelze spolehlivě multi-lateraci použít [30]. Například cihlová zeď má dle [29] útlum 6dB . Mezi další faktory ovlivňující sílu

přijatého signálu patří odrazy, absorpce a jiné deformace signálu způsobené při propagaci signálu reálným prostředím. Alternativou multi-laterace je *fingerprinting* prostředí, jež je založen na principu uložení konkrétních hodnot *RSS* pro konkrétní polohu. Výhodou je, že není potřeba znát matematický propagační model signálu pro dané prostředí. Pro odfiltrování výkyvů v síle signálu lze dle [14] použít plovoucí průměr či medián. Plovoucí průměr může brát N posledních hodnot či průměrovat vzorky za určitý časový interval.

■ Porovnání *Wi-Fi* a *Bluetooth Low Edition*

Dle [9] a [4] jsou 2.5 Ghz *Wi-Fi* sítě a *Bluetooth* zařízení provozovány na stejném frekvenčním pásmu a to přibližně na 2.5 Ghz, tudíž způsob šíření signálu prostředím je pro obě technologie podobný.

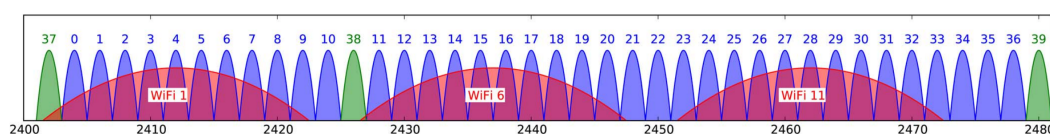
Obrázek 5.16 zachycuje 40 komunikačních kanálů pro *Bluetooth Low Edition* a tři nejčastěji okupované *Wi-Fi* kanály. Dle [14] propagace BLE zařízení probíhá pouze na třech kanálech (37,38,39), tudíž je pro zachycení *beacon* rámce vysílaného BLE zařízením nutné přepnout mezi třemi kanály, oproti technologii *Wi-Fi*, kde pokud bereme v potaz 2.5 GHz a 5 GHz sítě, je nutné přepnout mezi 50 kanály, což má za důsledek delší periodu mezi skenováními dostupných *Wi-Fi* sítí. Délka periody skenování je nepřímo úměrná frekvenci získávání hodnot *RSS*. Dle [14] platí, že větší množství vzorků *RSS* za jednotku času nám umožňuje použít menší časový interval pro průměrování. Délka okna pro průměrování by měla být co možná nejmenší, aby naměřená data v daném okně odpovídala co nejmenší ploše měření *RSS*. Pokud bychom brali průměr z pěti posledních měření a dostávali bychom pouze jedno měření za sekundu, tak by bylo nutné k naplnění okna pět vteřin. Za pět sekund, ale může uživatel běžnou chůzí ujít několik metrů a naměřená data by neodpovídala jeho aktuální pozici.

Výhody a nevýhody *Bluetooth LE* technologie:

- + vysoká frekvence detekce *beacon* rámce (měření *RSS*)
- + nízká spotřeba
- + možnost přesného umístění
- nutnost vybavení prostředí novými zařízeními

Výhody a nevýhody *Wi-Fi* technologie:

- + existující instalace v budovách
- nízká frekvence detekce *beacon* rámce (měření *RSS*)
- již stanovená pozice
- nutnost externího napájení



Obrázek 5.16: Frekvenční pásma *Wi-Fi* a *Bluetooth Low Edition* (převzato z [14])

Fingerprinting

Fingerprinting je dle [25] způsob lokalizace založený na porovnávání velikosti aktuálně přijímaného signálu vůči záznamům (otiskům) v předem naměřené databázi *RSS* hodnot. Tento proces lokalizace se skládá ze dvou hlavních částí - *online* a *offline* fáze.

Online fáze je vytvoření databáze referenčních bodů, které obsahují souřadnice v prostoru a množinu hodnot *RSS* pro jednotlivé majáky (*Wi-Fi* přístupové body, *Bluetooth LE* zařízení). Pro výpočet hodnoty *RSS* lze dle [28] použít filtr, aby se v měření odfiltrovaly fluktuace síly signálu. Pro jednotlivé majáky v konkrétním referenčním bodu je možné spočítat rozptyl *RSS* hodnot, aby bylo možné určit, jak velké jsou fluktuace síly signálu pro daný maják a konkrétní referenční bod. Majáky, pro které v daném referenčním bodu signál značně fluktuuje by neměli být do databáze zařazeny. Při vytváření referenčního bodu je nutné pro dané místo vytvořit otisk *RSS* signálu ze všech stran. Tudíž při měření *RSS* je potřeba, aby osoba provádějící měření se plynule otáčela kolem své vertikální osy na místě tak, aby vznikl *RSS* otisk 360° okolí. Jelikož je databáze referenčních bodů silně vázána na strukturu prostředí v době její tvorby, je nutné při významnější strukturální změně prostředí vytvořit novou databázi či databázi vhodným způsobem aktualizovat.

Offline fáze spočívá v porovnání aktuálně naměřených hodnot síly přijatého signálu vůči databázi referenčních bodů. Čím méně se liší naměřené hodnoty *RSS* jednotlivých majáků od hodnot *RSS* daného referenčního bodu uloženého v databázi, tím větší je pravděpodobnost, že se uživatel nachází právě v okolí daného referenčního bodu. Z tohoto důvodu je nutné stanovit metriku, která bude určovat rozdíl mezi naměřenými hodnotami *RSS* vůči konkrétnímu referenčnímu bodu. Pro nalezení konkrétní lokace přijímače na základě databáze referenčních bodů lze efektivně použít algoritmus *Nearest Neighbor* (*NN*) a jeho modifikace (*KNN*, *WKNN*).

Nearest Neighbor

Nearest Neighbor (*NN*) je klasifikační algoritmus, který z množiny referenčních vzorků vybere právě ten, který se od naměřených hodnot liší nejméně. Tento rozdíl budeme nazývat vzdáleností naměřených hodnot od hodnot referenčních. Pro určení této vzdálenosti se používá konkrétní metrika, která reprezentuje vzdálenost, jak se vzorek liší od referenčního vzorku. V našem případě je vzorek množina majáků s přiřazenými hodnotami měření *RSS* pro konkrétní lokaci. Dle [26] lze efektivně použít několik metrik pro měření vzdálenosti naměřených vzorků. Nejčastěji používané metriky jsou *Euklidovská* a *Manhattanská* vzdálenost naměřených hodnot, respektive součet hodnot metriky pro všechny majáky v daném referenčním bodu. Rovnice 5.1 definuje *Euklidovskou* vzdálenost aplikovanou na naše vzorky:

$$ED_i = \sqrt{\sum_{j=1}^{L_i} (R_{i,j} - M_j)^2} \quad (5.1)$$

a rovnice 5.2 definuje *Manhattanskou* vzdálenost aplikovanou na naše měření:

$$MD_i = \sum_{j=1}^{L_i} |R_{i,j} - M_j| \quad (5.2)$$

, kde

i je i -tý referenční bod,

j je j -tý maják,

L_i je počet majáků pro i -tý referenční bod,

$R_{i,j}$ je hodnota síly signálu pro i -tý referenční bod a j -tý maják,

M_j je naměřená hodnota síly signálu pro j -tý maják.

Algoritmus 1 popisuje proces nalezení souřadnic nejbližšího referenčního bodu na základě naměřených hodnot *RSS*.

Data: R, M

Result: Souřadnice nejbližšího referenčního bodu

Krok 1: vypočítej vzdálenost měření vůči každému referenčnímu bodu

Krok 2: seřaď referenční body dle vzdálenosti vzestupně

Krok 3: vrať souřadnice (x, y) referenčního bodu s nejnižší vzdáleností

Algoritmus 1: Algoritmus *Nearest Neighbor (NN)*

, kde

R je databáze referenčních bodů,

M jsou naměřené hodnoty RSS.

■ *k-Nearest Neighbors*

Algoritmus *k-Nearest Neighbors (kNN)* je upravený algoritmus *Nearest Neighbor*, který hledá K nejbližších referenčních bodů. Lze říci, že algoritmus *NN* je *kNN* s $K = 1$. Výstupem algoritmu 2 jsou opět souřadnice. Tyto souřadnice (x, y) jsou průměrné souřadnice K nejbližších referenčních bodů. Průměrné souřadnice lze vypočítat pomocí následujícího vzorce:

$$x = \frac{\sum_{i=1}^K X_{S_i}}{K}, y = \frac{\sum_{i=1}^K Y_{S_i}}{K} \quad (5.3)$$

, kde

K je počet nejbližších referenčních bodů,

S je množina indexů nejbližších referenčních bodů,

i je i -tý index referenčního bodu,

X je množina x-ových souřadnic referenčních bodů,

Y je množina y-ových souřadnic referenčních bodů,

x je x-ová souřadnice výsledného bodu,

y je y-ová souřadnice výsledného bodu.

Algoritmus 2 popisuje proces nalezení nejbližšího referenčního bodu na základě naměřených hodnot RSS .

Data: Databáze referenčních bodů R , naměřené hodnoty RSS M

Result: Souřadnice nejbližšího referenčního bodu

Krok 1: vypočítej vzdálenost měření vůči každému referenčnímu bodu

Krok 2: seřaď referenční body dle vzdálenosti vzestupně

Krok 3: vrať průměrné souřadnice (x, y) K referenčních bodů s nejnižší vzdáleností

Algoritmus 2: Algoritmus *k-Nearest Neighbors* (kNN)

■ *Weighted k-Nearest Neighbors*

Algoritmus *Weighted k-Nearest Neighbors* ($WkNN$) [25] je upravený algoritmus *k-Nearest Neighbors*, který opět hledá K nejbližších referenčních bodů a jehož výstupem jsou opět souřadnice. Tyto souřadnice (x, y) jsou váženým průměrem souřadnic K nejbližších referenčních bodů, kde váhy jsou nepřímo úměrné vzdálenosti měření. Váhu daného referenčního bodu lze spočítat následujícím vzorcem:

$$W_i = \frac{1}{D_i} \quad (5.4)$$

, kde

KD_i je vzdálenost i -tého referenčního bodu,

W_i je váha i -tého referenčního bodu.

Vážený průměr souřadnic lze spočítat následujícím vzorcem:

$$x = \frac{\sum_{i=1}^K w_{S_i} \cdot X_{S_i}}{\sum_{i=1}^K w_{S_i}}, y = \frac{\sum_{i=1}^K w_{S_i} \cdot Y_{S_i}}{\sum_{i=1}^K w_{S_i}}, \quad (5.5)$$

, kde

K je počet nejbližších referenčních bodů,

S je množina indexů nejbližších referenčních bodů,

i je i -tý index referenčního bodu,

W_{S_i} je váha referenčního bodu,

X je množina x-ových souřadnic referenčních bodů,

Y je množina y-ových souřadnic referenčních bodů,

x je x-ová souřadnice výsledného bodu,

y je y-ová souřadnice výsledného bodu.

Algoritmus 3 popisuje proces nalezení nejbližšího referenčního bodu na základě naměřených hodnot RSS a vah jednotlivých referenčních bodů.

Data: Databáze referenčních bodů R , naměřené hodnoty RSS M

Result: Souřadnice nejbližšího referenčního bodu

Krok 1: vypočítej vzdálenost měření vůči každému referenčnímu bodu

Krok 2: seřaď referenční body dle vzdálenosti vzestupně

Krok 3: vrať vážený průměr souřadnic (x, y) K referenčních bodů s nejnižší vzdáleností

Algoritmus 3: Algoritmus *Wk-Nearest Neighbors (WkNN)*

■ 5.3.2 Pedestrian Dead Reckoning

Dead Reckoning (DR) je způsob lokalizace na základě změny polohy v čase. Poloha je relativní vůči bodu, kdy sledování polohy započalo. Nová poloha sledovaného objektu je určena z předchozí polohy a informací o změně polohy (směr, vzdálenost, ...).

Pedestrian Dead Reckoning (PDR) je speciální případ metody lokalizace *Dead Reckoning*. Tato metoda odhaduje polohu jdoucího uživatele a předpokládá se, že uživatel mění polohu pouze chůzí.

Několik studií [18, 31, 32, 33, 34, 28, 6, 15] ověřilo, že *PDR* je vhodná metoda k určení polohy uživatele ve vnitřním prostředí pomocí mobilního telefonu za použití senzorů orientace a pohybu.

■ Změna polohy uživatele chůzí

Způsob změny polohy lidského jedince prostřednictvím běžné chůze je založen na provedení kroku vpřed. Tudíž uživatelská poloha se změní o délku jednoho kroku ve směru, jakým udělal krok. Při informaci o tom, že uživatel udělal krok, lze při známé délce kroku a směru určit, jak se uživatelská poloha změnila vůči předchozí poloze, než uživatel provedl daný krok. Předpokládáme, že jsme schopni detekovat krok. Nechť je L_s délka kroku a Θ úhel směru, jakým uživatel udělal krok. Rovnice pro výpočet nové polohy z polohy předchozí v horizontální rovině (bez ohledu na vertikální polohu uživatele) vypadá následovně:

$$\begin{aligned}x_i &= x_{i-1} + L_s \cdot \cos(\Theta) \\y_i &= y_{i-1} + L_s \cdot \sin(\Theta)\end{aligned}\tag{5.6}$$

, kde

x_{i-1} je x-ová souřadnice před provedením kroku,

y_{i-1} je y-ová souřadnice před provedením kroku,

x_i je x-ová souřadnice po provedení kroku,

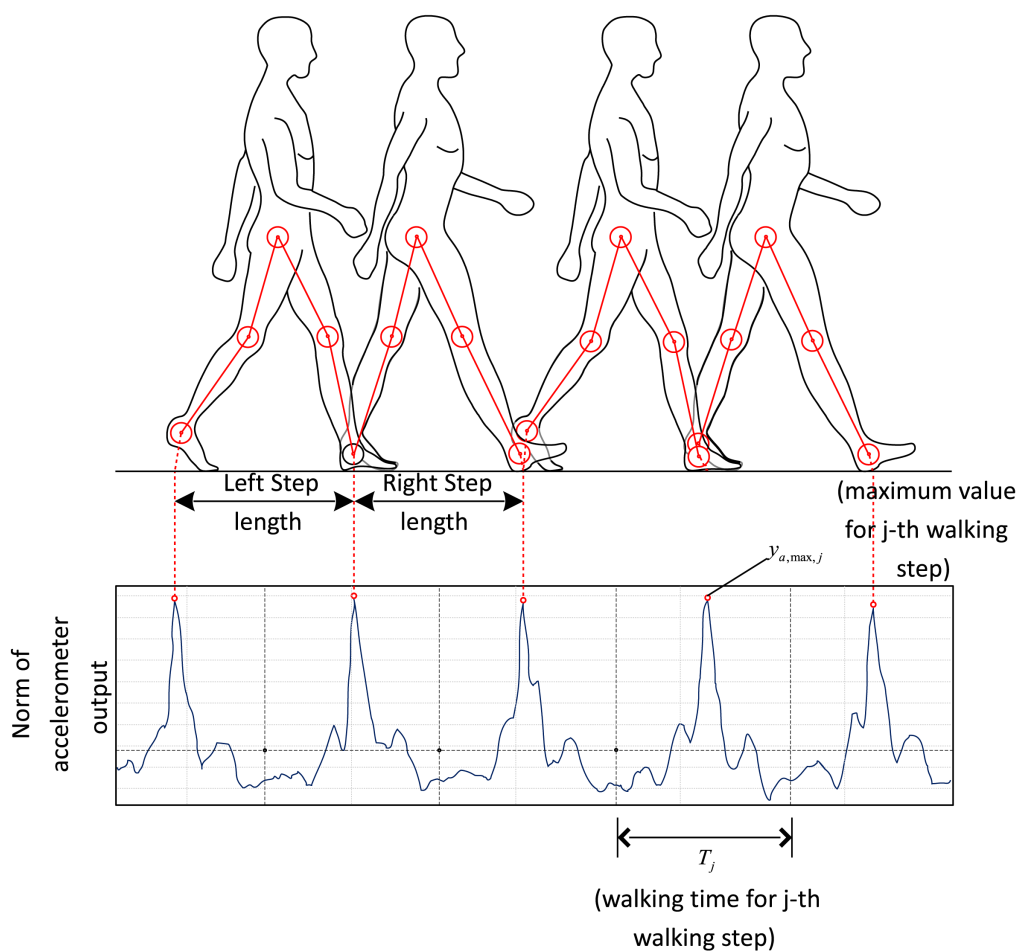
y_i je y-ová souřadnice po provedení kroku,

L_s je délka provedeného kroku,

Θ je azimut, kterým byl krok proveden.

■ Detekce kroku

Mobilní telefony jsou dle kapitoly 2 vybaveny senzory akcelerace (akcelerometry), tudíž je možné detekovat změny polohy lidského těla způsobené přesunem z jedné dolní končetiny na končetinu druhou. Jedná se o pohyb charakteristický a periodický, proto je možné v naměřených hodnotách akcelerace detekovat krok na základě změny velikosti akcelerace působící na mobilní zařízení, které uživatel drží v horní končetině. Obrázek 5.17 zachycuje velikost akcelerace v jednotlivých fázích chůze.



Obrázek 5.17: Vliv lidské chůze na akceleraci působící na mobilní zařízení (převzato z [35])

Mobilní telefony poskytují data o akceleraci na všech třech osách (x,y,z) a v závislosti na orientaci telefonu a způsobu držení zařízení se velikost akcelerace projevuje na různých osách akcelerometru. Dle [36] je možné zanedbat orientaci mobilního telefonu použitím Euklidovské normy na tří-dimenzionální vektor z akcelerometru. Dále se tedy velikostí akcelerace, pokud není řečeno jinak, bude myslet *Euklidovská* norma akceleračního vektoru, kterou lze spočítat následujícím vzorcem:

$$\|a\| = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (5.7)$$

, kde

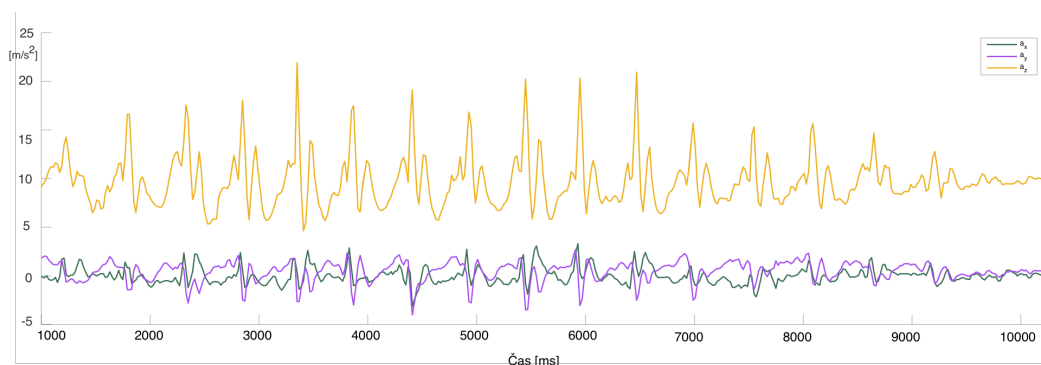
a_x je velikost akcelerace působící na osu x,

a_y je velikost akcelerace působící na osu y,

a_z je velikost akcelerace působící na osu z,

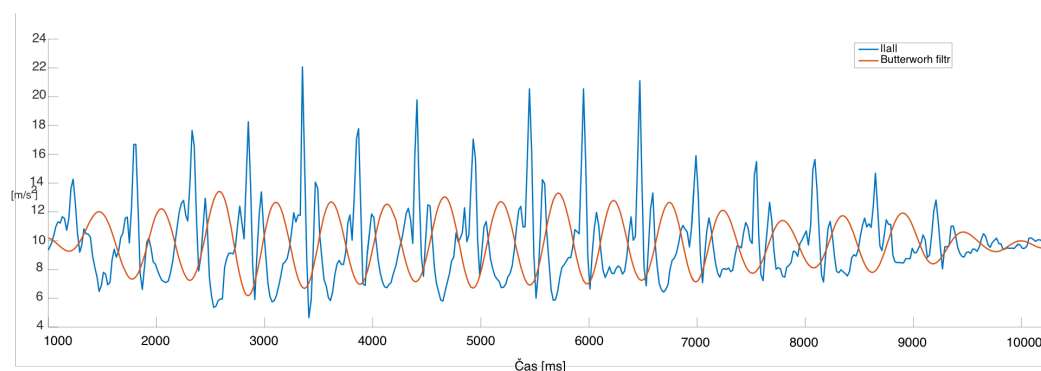
$\|a\|$ je Euklidovská norma akceleračního vektoru.

Dle [18, 37] je vhodné před detekcí kroků data filtrovat pomocí filtru dolní propusti, aby došlo k odstranění vysokofrekvenčních kmitů, které neodpovídají vzoru chůze. Obě zmíněné práce používají k filtraci *Butterworth* filtr, který v naměřených datech odstraní vysokofrekvenční kmit. Obrázek 5.18 zachycuje velikost akcelerace na jednotlivých osách akcelerometru v průběhu chůze.



Obrázek 5.18: Velikost akcelerace na jednotlivých osách v průběhu chůze

Kdežto obrázek 5.19 zobrazuje Euklidovskou normu složek akcelerace a její filtraci pomocí *Butterworth* filtru šestého stupně. Z obrázku je patrné, že filtrace má vliv na amplitudu a způsobuje zpoždění signálu. Oba tyto faktory musí být brány v potaz při detekci kroku a následné synchronizaci dat z krokoměru s daty jiných senzorů.



Obrázek 5.19: Velikost normy akcelerace a její filtrovaný průběh

Z obrázků 5.17 a 5.19 lze vidět lokální maxima a minima akcelerace vyvolané chůzí. Každý krok je charakterizován lokálním maximem (vrcholem) následovaný lokálním minimem (údolím). Detekcí těchto extrémů lze detekovat kroky, které uživatel provede.

Pro detekci kroků v takto filtrovaných datech z akcelerometru lze využít hraniční (anglicky *threshold*) metodu, kdy se stanoví velikost akcelerace, která se od naměřených dat odečte a následně se pak detekuje přechod ze záporných hodnot akcelerace do hodnot kladných a naopak. Tyto přechody určují, zda se bude hledat globální minimum či maximum hodnot akcelerace. Proto, aby mohl být po sobě jdoucí pár maximum-minimum označen jako krok, je nutné, aby průběh hodnot mezi

jednotlivými přechody splňoval určité omezující podmínky na rozdíl extrémů a jejich časový rozestup. Práce [38] definuje vhodnou velikost hranice akcelerace jako velikost složky tíhového zrychlení ($9.81m/s^2$), jež je obsažena ve výstupu akcelerometru.

Algoritmus 4 popisuje metodu [39], která pro dané vstupní parametry vrátí logickou hodnotu, která je pravdivá v případě detekce kroku. .

```

Data:  $S_{i-1}, a_i, t_i, \tau, \delta_{jMin}, \delta_{tMin}, \delta_{tMax}$ 
Result: Boolean
 $S_i = \text{"unknown"}$ 
result = false
if  $a_i > \tau$  then
     $S_i = \text{"rising"}$ 
    if  $a_i > accelerationMax$  then
        accelerationMax =  $a_i$ 
    end
else if  $a_i < \tau$  then
     $S_i = \text{"falling"}$ 
    startTimeMin =  $t_i$ 
    if  $a_i < accelerationMin$  then
        accelerationMin =  $a_i$ 
    end
end
if  $S_i == \text{"unknown"}$  then
     $S_i = S_{i-1}$ 
end
if  $S_i \neq S_{i-1}$  then
    if  $S_i == \text{"rising"}$  then
        if  $accelerationMax - accelerationMin > \delta_{jMin}$  and
             $t_i - startTime > \delta_{tMin}$  and  $t_i - startTime < \delta_{tMax}$  then
            result = true
        end
        startTime =  $t_i$ 
        accelerationMin =  $+\infty$ 
        accelerationMax =  $a_i$ 
    end
end
return result

```

Algoritmus 4: Detektor kroků - Threshold metoda

, kde

S_{i-1} je předchozí stav,

a_i je velikost akcelerace v čase t_i ,

t_i je časové razítko (timestamp) zaznamenání hodnoty a_i ,

τ je hranice pro detekci přechodu mezi extrémy,

δ_{jMin} je minimální rozdíl akcelerace mezi extrémy,

$\delta_{tMin}, \delta_{tMax}$ je minimální a maximální délka trvání kroku,

Z principu je hraniční metoda závislá na velikosti hranice (*thresholdu*), což v případě, že dojde posunutí minima nad hranici či maxima pod hranici, zamezí detekci kroku. K takové situaci může dojít při změně orientace zařízení v průběhu kroku. Řešením může být adaptivní výpočet hranice či použití alternativní metody. Alternativní metodou je hledání extrémů bez potřeby definování hranice. [36, 39] navrhuje metody detekce kroku bez potřeby definice hranice. Krok je detekován v případě, že po lokálním maximu následuje lokální minimum a rozdíl akcelerace mezi extrémy a jejich časový rozestup opět splňuje omezující podmínky.

Aby bylo možné detekovat změnu polohy uživatele co nejpřesněji, je potřeba navrhnout algoritmus, který minimalizuje počet *false positive* (detekován krok, který nebyl proveden) a *false negative* (nebyl detekován krok, který by proveden) detekcí kroku a poskytuje přijatelnou odezvu detekování kroku v *realtime* použití. [39] navrhuje metodu, která je výrazně spolehlivější při detekování kroků pro různé styly

chůze. Tento algoritmus jsem modifikoval a definoval v algoritmu 5.

Data: $S_{i-1}, a_i, t_i, \delta_{jMin}, \delta_{tMin}, \delta_{tMax}, lastPeak, lastValley, paceBuffer,$
 $jerkBuffer, J$

Result: Boolean

```

 $S_i = "unknown"$ 
result = false
if  $a_i > a_{i-1}$  then
|  $S_i = "rising"$ 
else if  $a_i < a_{i-1}$  then
|  $S_i = "falling"$ 
end
if  $S_i == "unknown"$  then
|  $S_i = S_{i-1}$ 
end
if  $S_i \neq "unknown"$  and  $S_{i-1} \neq "unknown"$  and  $S_i \neq S_{i-1}$  then
| if  $S_i == "rising"$  then
| | if  $lastPeak$  is defined then
| | | durationAvg = Mean(durationBuffer)
| | | jerkAvg = Mean(jerkBuffer)
| | | duration = lastPeak[0] -  $t_i$ 
| | | jerk = lastPeak[1] -  $a_i$ 
| | | if  $jerk \geq \delta_{jMin}$  and ( $jerk \geq \frac{jerkAvg}{2}$  or  $jerk \geq 2 * J$ ) then
| | | | if  $duration \geq \frac{durationAvg}{2}$  and  $duration \leq 3 * duration$  then
| | | | | stepDuration =  $2 * duration$  if  $stepDuration \geq \delta_{tMin}$ 
| | | | | and  $stepDuration \leq \delta_{tMax}$  then
| | | | | | jerkBuffer.add(jerk)
| | | | | | durationBuffer.add(duration)
| | | | | | result = true
| | | | end
| | | end
| | end
| | end
| | lastValley =  $\{t_i, a_i\}$ 
| else if  $S_i == "falling"$  then
| | lastPeak =  $\{t_i, a_i\}$ 
| end
end
return result

```

Algoritmus 5: Detektor kroků - modifikovaný *Adaptiv* algoritmus

, kde

S_{i-1} je předchozí stav,

a_i je velikost akcelerace v čase t_i ,

t_i je časové razítko (*timestamp*) zaznamenání hodnoty a_i ,

δ_{jMin} je minimální rozdíl akcelerace mezi extrémy,

δ_{tMin} je minimální délka trvání kroku,

δ_{tMax} je maximální délka trvání kroku.

J je výchozí průměrný rozdíl akcelerace mezi extrémy,

lastPeak je poslední lokální maximum,

lastValley je poslední lokální minimum,

durationBuffer je zásobník N posledních rozdílů časů mezi extrémy,

jerkBuffer je zásobník M posledních rozdílů akcelerace mezi extrémy,

Pro odhad *délky kroku* je možné dle [40] použít vzorec 5.8, kde konstanta K je potřeba nastavit dle uživatelského stylu chůze.

$$L_s = K \cdot \sqrt[4]{a_{max} - a_{min}} \quad (5.8)$$

, kde

L_s je výsledná délka kroku,

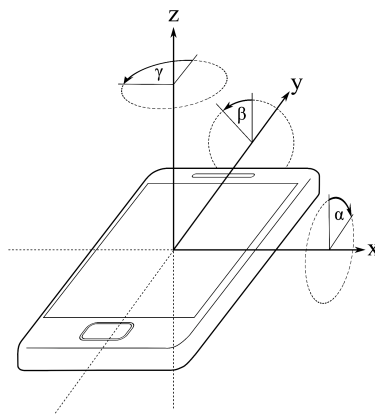
K je koeficient určený trénováním krokoměru,

a_{max} je velikost maxima normy akcelerace pro daný krok,

a_{min} je velikost minima normy akcelerace pro daný krok.

■ Orientace

Obrázek 5.20 zachycuje souřadný systém mobilního zařízení použitý pro reprezentaci orientace zařízení v prostoru. Úhly α , β a γ reprezentují úhly naklonění zařízení vůči zemskému povrchu. Úhel α se nazývá *pitch*, úhel β *roll* a úhel γ se nazývá *azimuth*.



Obrázek 5.20: Souřadný systém orientace zařízení v prostoru (převzato z [41])

Pro odhad nové polohy vyžaduje propagační rovnice 5.6 velikost azimutu (úhel γ), jakým směrem uživatel provedl krok. Za účelem zjištění azimutu se používá zařízení zvané kompas, které určuje směr, jakým se nachází magnetický severní pól. Dle kapitoly 2 jsou moderní mobilní telefony vybavovány magnetometrem, který měří velikost síly magnetického pole působícího na zařízení. Magnetometr měří sílu na třech osách (x, y, z) vůči vztažné soustavě, kterou je mobilní telefon (relativně vůči tělu mobilního zařízení). Aby bylo možné určit směr, jakým se nachází magnetický severní pól, je nutné transformovat naměřená data ke vztažné soustavě, kterou je horizontální rovina Země, takzvaně vykompenzovat naklonění zařízení. K transformaci velikosti naměřených dat magnetometru ze vztažné soustavy telefonu na vztažnou soustavu Země, je potřeba znát orientaci mobilního telefonu vůči horizontální rovině - úhly α a β . Tyto úhly natočení zařízení lze dle [32] získat z akcelerometru, jelikož akcelerometr registruje tíhové zrychlení, které se promítá na jednotlivé osy v závislosti na natočení mobilního zařízení vůči Zemi. Dle [32] lze

úhly natočení α a β získat vyřešením následující rovnice:

$$\begin{aligned}
 R_x(\theta) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \\
 R_y(\gamma) &= \begin{bmatrix} \cos \gamma & 0 & -\sin \gamma \\ 0 & 1 & 0 \\ \sin \gamma & 0 & \cos \gamma \end{bmatrix} \\
 C = R_y(\gamma) \cdot R_x(\theta) &= \begin{bmatrix} \cos \gamma & \sin \gamma \sin \theta & -\sin \gamma \cos \theta \\ 0 & \cos \theta & \sin \theta \\ \sin \gamma & -\sin \theta \cos \gamma & \cos \theta \cos \gamma \end{bmatrix} \\
 \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = C \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} &= \begin{bmatrix} \cos \gamma & \sin \gamma \sin \theta & -\sin \gamma \cos \theta \\ 0 & \cos \theta & \sin \theta \\ \sin \gamma & -\sin \theta \cos \gamma & \cos \theta \cos \gamma \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}
 \end{aligned}$$

, kde

a_x, a_y, a_z jsou jednotlivé složky vektoru akcelerace získaného z akcelerometru,

α je hledaný úhel *pitch*,

β je hledaný úhel *roll*,

g je velikost tíhového zrychlení.

Po získání úhlů *pitch* a *roll* z vektoru akcelerace je nutné dle [32, 42] provést transformaci vektoru velikosti magnetického pole mezi souřadnými systémy jednotlivých vztažných soustav pomocí následujícího vzorce:

$$\begin{aligned}
 m_x^p &= m_x^b \cos \gamma + m_z^b \sin \gamma \\
 m_y^p &= m_x^b \sin \theta \sin \gamma + m_y^b \cos \theta - m_z^b \sin \theta \cos \gamma
 \end{aligned}$$

, kde

m_x^b, m_y^b, m_z^b jsou jednotlivé složky vektoru síly magnetického pole získaného z magnetometru vůči vztažné soustavě, kterou je mobilní zařízení,

m_x^p, m_y^p jsou horizontální složky vektoru síly magnetického pole vůči vztažné soustavě, kterou je zemský povrch,

α, β jsou úhly *pitch* a *roll*,

g je velikost tíhového zrychlení.

Pro získání azimutu z vektoru velikosti magnetického pole v horizontální rovině XY lze dle [32] použít následující vzorec:

$$\gamma = \text{atan2}(m_x^p, m_y^p)$$

, kde

m_x^p, m_y^p jsou horizontální složky vektoru síly magnetického pole,

γ, β je výsledný azimut.

Z principu funkce je dle kapitoly 2 magnetometr náchylný na elektromagnetická rušení způsobená elektronickými zařízeními či strukturou okolí. Typickým elektromagneticky zarušeným prostředím jsou právě budovy, které jsou vybaveny různými elektronickými zařízeními a struktura prostředí může obsahovat prvky ovlivňující magnetické pole. Vysokofrekvenční magnetické rušení je možné odstranit filtrem typu dolní propusti, tím ale dojde k umělému zpoždění odezvy a ztrácíme tak dynamiku změny orientace. Alternativním řešením pro zjištění orientace zařízení je použití gyroskopu, který dle popisu v kapitole 2 měří úhlovou rychlost otáčení zařízení okolo všech tří os a neregistruje magnetická rušení jako magnetometr. Integrací úhlové rychlosti po určitý čas získáme úhel otočení, který je relativní vůči absolutnímu úhlu z počátku integrace. Pro získání absolutního úhlu orientace je tedy nutné před integrací přičíst počáteční úhel orientace získaný z alternativních senzorů (akcelerometr+magnetometr). Integraci úhlové rychlosti rotace je dle [43] nutné převést z Eulerovy reprezentace na reprezentaci pomocí Kvaternionů, aby při integraci nedošlo k uvíznutí (stav *gimbal lock*). Jelikož naměřená data gyroskopu obsahují chybovou složku, integrací dochází k její kumulaci a výsledný úhel po určitém čase obsahuje nezanedbatelnou chybovou složku, která výrazně ovlivňuje velikost úhlu. Vhodným zkombinováním obou způsobů určení orientace lze získat ztracenou dynamiku a potlačit působení magnetického rušení na určení orientace pomocí magnetometru. Kombinaci orientace z magnetometru s integrací dat z gyroskopu lze dle [41] docílit pomocí komplementárního filtru, který kombinuje naměřená data stejného typu ze dvou různých zdrojů. Následující vzorec popisuje komplementární filtr:

$$\gamma_t = \lambda \cdot (\gamma_{t-1} + \Delta\gamma_{gyro}) + (1 - \lambda) \cdot \gamma_{magn} \quad (5.9)$$

, kde

γ_t je výsledný úhel,

γ_{t-1} je předchozí úhel,

$\Delta\gamma_{gyro}$ je relativní úhel získaný integrací dat z gyroskopu,

γ_{magn} je absolutní úhel získaný z magnetometru (a akcelerometru),

λ je koeficient určující váhu jednotlivých dat.

Vhodnou volbou váhy lze docílit, že dynamické změny průběhu budou získané z gyroskopu a stabilní (absolutní) úhel orientace bude získán z filtrovaného magnetometru.

Díky rozsahu hodnot $< 0^\circ, 360^\circ$ azimutu nelze použít běžné průměrování (vážení), protože pokud by jeden úhel byl 10° a druhý 350° , očekával by se průměr 0° , ale pouhým zprůměrováním by byl výsledný úhel 180° , což při krajních hodnotách rozsahu může způsobit chybný výstup. Řešením je dle [44] rozložení obou úhlů pomocí goniometrických funkcí $\sin$ a $\cos$, zprůměrování výstupů jednotlivých funkcí a poté opětovné převedení na úhel dle následujícího algoritmu

```
Data:  $\gamma_t, \gamma_1, \gamma_2, \lambda$ 
sin1 =  $\sin(\gamma_1)$ 
sin2 =  $\sin(\gamma_2)$ 
cos1 =  $\cos(\gamma_1)$ 
cos2 =  $\cos(\gamma_2)$ 
sinPart =  $\lambda \cdot \sin1 + (1 - \lambda) \cdot \sin2$ 
cosPart =  $\lambda \cdot \cos1 + (1 - \lambda) \cdot \cos2$ 
 $\gamma = \text{atan2}(\sinPart, \cosPart)$ 
return  $\gamma$ 
```

Algoritmus 6: Algoritmus pro poměrovou kombinaci dvou úhlů

5.3.3 Kombinace PDR a fingerprintingu

PDR metoda pro určení absolutní polohy vyžaduje počáteční polohu, kde se uživatel nachází před detekcí prvního kroku. Aby metoda lokalizace pracovala autonomně, bez potřeby interakce s uživatelem, je nutné získat počáteční polohu uživatele pomocí určení polohy na základě bezdrátových technologií - konkrétně metoda *WiFi* a *Bluetooth Low Energy fingerprintingu*. Následující algoritmus popisuje základní proces lokace uživatele s využitím PDR a fingerprintingu:

```
/* Získej počáteční polohu z fingerprintingu */
[x,y] = getLocationUsingFingerprinting()
reportNewLocation(x,y)
while true do
    /* Čekej na detekci dalšího kroku */
    [stepLength, azimuth] = waitForStep()
    x = x + stepLength * cos(azimuth)
    y = y + stepLength * sin(azimuth)
    reportNewLocation(x,y)
end
```

Algoritmus 7: Algoritmus lokace uživatele s využitím PDR a fingerprintingu

Map-matching

Dle [40] vznikají při lokalizaci uživatele pomocí metody PDR chyby způsobené nepřesným odhadem délky kroku a nepřesným odhadem azimutu. Tyto chyby se kumulují v průběhu chůze a s každým dalším krokem roste odchylka lokalizace vůči skutečné poloze. Tuto chybu je možné snížit použitím informací o prostředí,

ve kterém je uživatel lokalizován. Této metodě využití mapových plánů se anglicky říká *map-matching*. Díky *map-matchingu* je možné detekovat nereálné změny polohy uživatele způsobené nepřesným odhadem nové polohy. Mezi nereálné změny polohy lze zařadit například průchod zdí či překážkou.

Vhodnou realizací *map-matchingu* je dle [19, 34, 40, 31, 18, 20] použití částicového filtru.

■ Částicový filtr

Částicový filtr je dle [19] numerická metoda, která nám poskytuje aproximaci aposteriorního rozdělení nepozorovaného stavu na základě sekvence akcí a pozorování. Nepozorovaným (skrytým) stavem je v našem případě poloha a orientace uživatele, jež se snažíme odhadnout. Na základě aproximace aposteriorního rozdělení nepozorovaného stavu jsme schopni odhadnout, kde se uživatel s největší pravděpodobností nachází. Aposteriorní rozložení nepozorovaného stavu je založeno na sekvenci podmíněně nezávislých akcí a na sekvenci nepřesných pozorování, které dostáváme ve formě propagačního modelu změny polohy (akce) a ve formě pozorování. Pozorování v našem případě je informace, zda je stav (poloha uživatele) dle mapových podkladů reálný a zda se daná poloha nachází v dosažitelných částech mapy. V případě částicového filtru je aposteriorní rozložení definováno pomocí konečné množiny částic. Jedna částice je jeden vhodně (náhodně zašuměný) umístěný vzorek nepozorovaného stavu s určitou vahou určenou na základě pozorování. Pomocí částic lze definovat libovolné pravděpodobnostní rozložení. Výsledný, nejpravděpodobnější, odhad nepozorovaného stavu je dán váženým průměrem částic. Metodu částicového filtru lze dle [19] rozdělit na tři hlavní části. Na predikci, korekci a převzorkování (anglicky *resampling*). Krok predikce změny umístění všech částic na základě akce (provedení kroku) dle propagačního modelu změny polohy. Krok korekce přiřadí váhu jednotlivým částicím na základě pozorování, kterým je v našem případě ověření, zda částice leží v dosažitelném místě mapy. Pokud ano, přiřadí se částici váha 1, v opačném případě váha 0. Jelikož pravděpodobnost aposteriorního rozložení je dána denzitou částic, musí být proveden *resampling*, kdy se částice s malou vahou odstraní z množiny částic a částice s velkou vahou se duplikují. Poté se všem částicím váhy znormalizují, aby jejich součet byl roven 1. V případě, kdy váhy nabývají binárních hodnot, není nutné dělat *resampling*. Důvod generování nových částic na základě váhy spočívá v tom, že aposteriorní rozložení pravděpodobnosti odpovídá hustotě rozložení částic a pro částice s vyšší vahou musí být zajištěna vyšší pravděpodobnost náhodného výběru. Zároveň hustota rozložení částic ovlivňuje výsledný odhad nepozorovaného stavu. Proces predikce, korekce a *resamplingu* se provádí po každém kroku. Před první predikcí musí být vygenerována množina částic charakterizující apriorní rozložení pravděpodobnosti polohy uživatele. Bez znalosti přibližné polohy uživatele získané z *fingerprintingu* bychom museli vygenerovat uniformě rozmístěné částice na celém dosažitelném prostoru budovy, jelikož se uživatel může nacházet v libovolné části budovy se stejnou pravděpodobností. V případě přibližné polohy z *fingerprintingu* lze vygenerovat uniformě rozmístěné částice pouze v určitém radiusu okolo přibližné polohy získané pomocí *fingerprintingu*. Pokud bychom přibližnou polohou nezredukovali počáteční počet částic, požadovala by metoda čas-

ticového filtru dle [19, 45] více výpočetních operací, jelikož počet výpočtů je úměrný počtu částic a mohli bychom tak ztratit možnost použít částicový filtr v reálném čase, jelikož by požadavky na výpočetní výkon přesahovaly možnosti mobilního telefonu. Algoritmus 8 popisuje algoritmus částicového filtru inspirovaný dle [19].

```

Data:  $S_{k-1}, y_k$ 
 $w_{SUM} = 0$ 
 $\tilde{S}_k = \{\}$ 
/* Propaguj částice a spočítej váhy */
for  $i = 1$  to  $N$  do
     $\tilde{s}_k^i = \text{moveParticle}(s_{k-1}^i, v_k)$ 
     $\tilde{w}_k^i = \text{particleWeight}(s_k^i, y_k)$ 
     $w_{SUM} = w_{SUM} + \tilde{w}_k^i$   $\tilde{S}_k = \tilde{S}_k \cup \{\tilde{s}_k^i; \tilde{w}_k^i\}$ 
end
/* Normalizuj váhy */
for  $i = 1$  to  $N$  do
     $w_k^i = \frac{\tilde{w}_k^i}{w_{SUM}}$ 
end
/* Převzorkuj množinu částic */
 $S_k = \text{resample}(\tilde{S}_k)$ 
return  $S_k = \{s_k^i; w_k^i\}$ 

```

Algoritmus 8: Algoritmus částicového filtru

, kde

S_{k-1} je množina částic,

S_k je nová množina částic,

w_k^i je váha nové částice,

v_k je akce (informace o kroku),

y_k je pozorování.

Algoritmus 9 popisuje metodu pro výpočet váhy částice s_k na základě pozorování y_k . Jelikož v našem případě je váha binární a určuje se na základě základě mapových podkladů, odpovídá y_k mapovým podkladům.

```

Data:  $s_k = \{x_{c_k}, y_{c_k}, \gamma_k\}, y_k$ 
if  $\text{locationIsReachableInMap}(\{x_{c_k}, y_{c_k}\}, y_k)$  then
    | return 1
else
    | return 0
end

```

Algoritmus 9: Algoritmus výpočtu váhy částice

, kde

s_k je částice,

x_{c_k}, y_{c_k} jsou x-ová a y-ová souřadnice částice,

y_k jsou mapové podklady.

Alogritmus 10 popisuje propagační funkci pro propagaci částice dle délky kroku a rozdílu azimutu vůči kroku předchozímu.

Data: $s_{k-1} = \{xc_{k-1}, yc_{k-1}, \gamma_{k-1}\}$, $v_k = \{L_s, \Delta\gamma\}$
 $L_{s_{noised}} = \mathcal{N}(L_s, \sigma_{L_s}^2)$
 $\gamma_k = \gamma_{k-1} + \mathcal{N}(\Delta\gamma, \sigma_\gamma^2)$
 $xc_k = xc_{k-1} + L_{s_{noised}} \cdot \cos(\gamma_k)$
 $yc_k = yc_{k-1} + L_{s_{noised}} \cdot \sin(\gamma_k)$
return $s_k = \{xc_k, yc_k, \gamma_k\}$

Algoritmus 10: Algoritmus propagace částice

, kde

s_{k-1} je původní částice,

s_k je propagovaná částice,

xc, yc jsou x-ová a y-ová souřadnice částice,

γ je orientace (azimut),

L_s je délka kroku,

$\Delta\gamma$ je změna úhlu mezi s_{k-1} a s_k ,

$\sigma_{L_s}^2, \sigma_\gamma^2$ jsou rozptyl délky kroku a rozptyl rozdílu úhlu.

■ Částicový filtr s rozšířeným pozorováním

Práce [20] používá jako model pozorování polohu odhadnutou metodou *fingerprintingu*. Algoritmus 11 popisuje metodu pro výpočet váhy částice s_k na základě pozorování y_k , kterým je kombinace polohy z *fingerprintingu* a pravděpodobnost, že se částice nachází v dosažitelných částech mapových podkladů. Čím blíže je částice k odhadnuté poloze z *fingerprintu*, tím větší je její váha.

Data: $s_k = \{xc_k, yc_k, \gamma_k\}$, y_k
if *locationIsReachableInMap*($\{xc_k, yc_k\}, y_k$) **then**
 $w = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp - \frac{(xc - xc_f)^2 + (yc - yc_f)^2}{2\sigma^2}$
 return w
else
 return 0
end

Algoritmus 11: Algoritmus výpočtu váhy částice na základě polohy z *fingerprintingu* a mapových podkladech

, kde

s_k je částice,

xc, yc jsou x-ová a y-ová souřadnice částice,

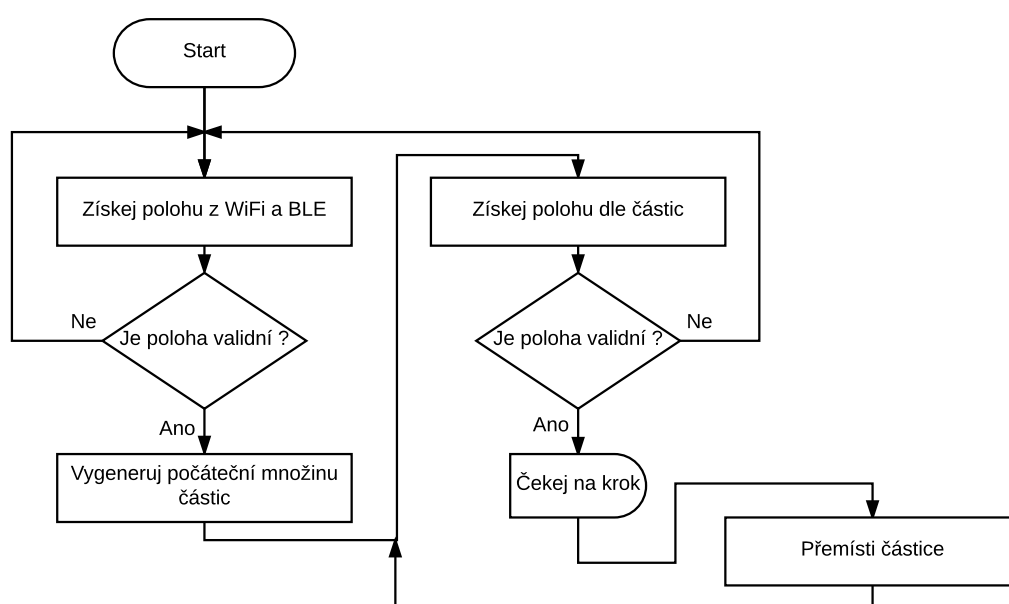
$y_k = \{m, xc_f, yc_f, \sigma\}$ je pozorování, kde

m jsou mapové podklady,

x_{cf}, y_{cf} jsou souřadnice polohy uživatele získané pomocí *fingerprintingu* a σ je rozptyl naměřené polohy z *fingerprintu*.

5.3.4 Výsledný proces lokalizace

Výsledný proces lokalizace je kombinace obou výše navržených metod pro určení polohy. Proces kombinuje odhadnutou polohu z *fingerprintingu* s polohou získanou pomocí *PDR*. Souřadnice polohy se musí nacházet v dosažitelných částech mapových podkladů (tzv. *map-mapmatching*). Kombinace těchto metod lokalizace a metody *map-matchingu* je realizována částicovým filtrem, který na základě pozorování získaného z *fingerprintingu* a akcí z *PDR* aproximuje rozložení pravděpodobnosti polohy uživatele. Obrázek 5.21 popisuje celý proces lokalizace.



Obrázek 5.21: Proces lokalizace

5.4 Navigace

Jedním z cílů této práce je navrhnout metodu navigace ve vnitřním prostředí. Toto prostředí je definováno pomocí mapových podkladů, které obsahují množinu překážek a význačné body. Metoda navigace by měla být schopna dovést uživatele z bodu A do bodu B. Bodem A je aktuální poloha uživatele a bodem B je význačný bod, do kterého se chce uživatel dostat.

5.4.1 Význačné body

Význačný bod je objekt umístěný v mapových podkladech. Tento objekt má definovány souřadnice, na kterých se nachází. Dále tento objekt má název, dle kterého může uživatel jednotlivé význačné body rozlišit.

5.4.2 Hledání cesty

Kritériem pro hledání cesty v rámci vnitřního prostředí bude nejkratší délka cesty z bodu A do bodu B. Omezením pro hledání cesty je podoba vnitřního prostředí, která je definována pomocí polygonů a lomených čar. Nalezená cesta tedy musí být co možná nejkratší a musí se nesmí vést přes neprůchozí překážky. Jako funkci pro definici vzdálenosti mezi dvěma body použijí Euklidovskou vzdálenost, která je definována následujícím vzorcem:

$$d(\{x_1, y_1\}, \{x_2, y_2\}) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (5.10)$$

Pro hledání nejkratší cesty v množině polygonů je dle [46] vhodné vytvořit graf viditelnosti (anglicky *visibility graph*). Graf viditelnosti je graf, jehož uzly tvoří vrcholy jednotlivých polygonů a lomených čar. Mezi dvěma uzly vede hrana pouze tehdy, když je mezi danými uzly přímá viditelnost. Přímá viditelnost znamená, že spojnice těchto dvou bodů v mapě neprotne žádnou hranu polygonu či lomené čáry. Hrana je ohodnocena Euklidovskou vzdáleností mezi dvěma vrcholy, jež hrana spojuje. Pro hledání nejkratší cesty v grafu viditelnosti lze dle [46] použít informovaný A^* (*A-star*) algoritmus, jehož stavovým prostorem bude právě graf

viditelnosti. A* algoritmus je popsán v algoritmu 12.

```

 $N_{location}.gCost = 0$ 
 $N_{location}.hCost = distance(N_{location}, N_{destination})$ 
 $openList = \{\}$ 
 $closedList = \{\}$ 
 $openList = openList \cup N_{location}$ 
while  $openList \neq \{\}$  do
   $N = openList[0]$ 
   $closedList = closedList \cup N$ 
  if  $N == N_{destination}$  then
    | Zrekonstruuuj a vrať cestu do cíle
  else
    for  $\forall N_{adj} \in adj(N)$  do
      if  $N_{adj} \notin closedList$  then
        if  $N_{adj} \notin openList$  then
           $openList = openList \cup N_{adj}$ 
           $N_{adj}.previous = N$ 
           $N_{adj}.gCost = N.gCost + distance(N, N_{adj})$ 
           $N_{adj}.hCost = distance(N_{adj}, N_{destination})$ 
          Seřaď  $openList$  dle součtu polí  $gCost$  a  $hCost$  vzestupně
        else if  $N_{adj}.gCost > N.gCost + distance(N, N_{adj})$  then
           $N_{adj}.previous = N$ 
           $N_{adj}.gCost = N.gCost + distance(N, N_{adj})$ 
        end
      end
    end
  end
end
  Vrať prázdnou cestu

```

Algoritmus 12: Algoritmus A* pro hledání nejkratší cesty

, kde

$N_{location}$ je počáteční vrchol,

$N_{destination}$ je cílový vrchol,

$openList$ je množina vrcholů připravená k prohledání,

$closedList$ je množina již prohledaných vrcholů,

$.gCost$ je cena cesty z počátečního,

$.hCost$ je heuristický odhad vzdálenosti do cílového vrcholu,

$.previous$ je reference na vrchol, ze kterého se do daného vrcholu přišlo,

$distance$ je metoda pro výpočet Euklidovské vzdálenosti.

■ Instrukce

Po úspěšném nalezení cesty z aktuální polohy uživatele do cílového bodu, musí je uživatel informován prostřednictvím instrukcí, které jej dovedou do cílového bodu. Těmito instrukcemi bude nejkratší cesta vykreslená do mapových podkladů společně s aktuální polohou uživatele. Dále bude uživateli poskytnuta informace, jakým směrem stojí.

■ 5.4.3 Výsledný proces navigace

Algoritmus 13 slovně popisuje navržený navigační proces. Navigační proces využívá lokalizační proces k nalezení aktuální polohy uživatele a nejkratší cestu hledá dle navrženého algoritmu hledání nejkratší cesty.

Poskytni uživateli instrukce

Nabídní uživateli seznam význačných bodů

Čekej, než uživatel vybere cílový bod

Lokalizuj uživatele na místě

while *Poloha není v úzkém radiusu cíle* **do**

 Nalezni nejkratší cestu z aktuální polohy do cílového bodu

 Poskytni uživateli zpětnou vazbu

 Čekej na změnu polohy

end

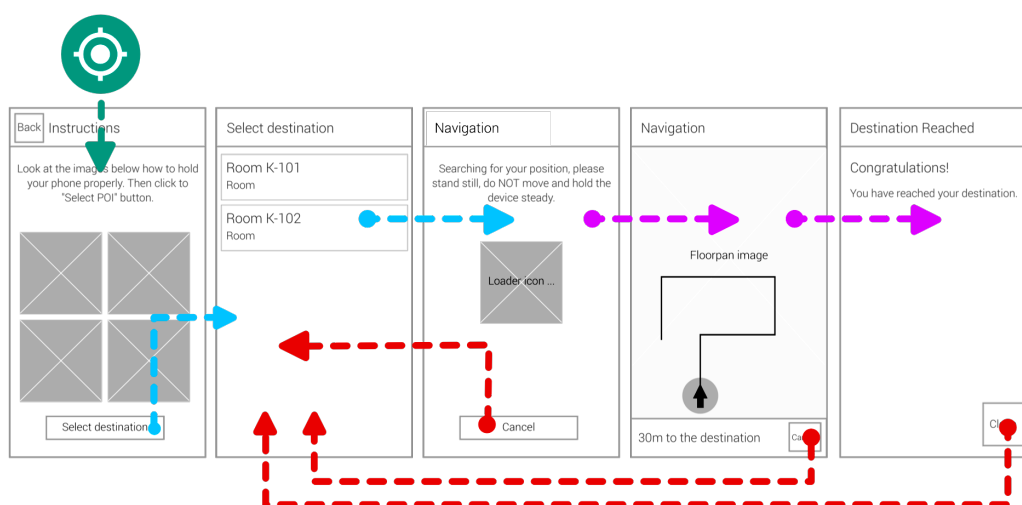
Ukonči navigaci Zobraz informaci o dosažení cíle

Algoritmus 13: Algoritmus výsledného procesu navigace

■ 5.4.4 Návrh uživatelského rozhraní

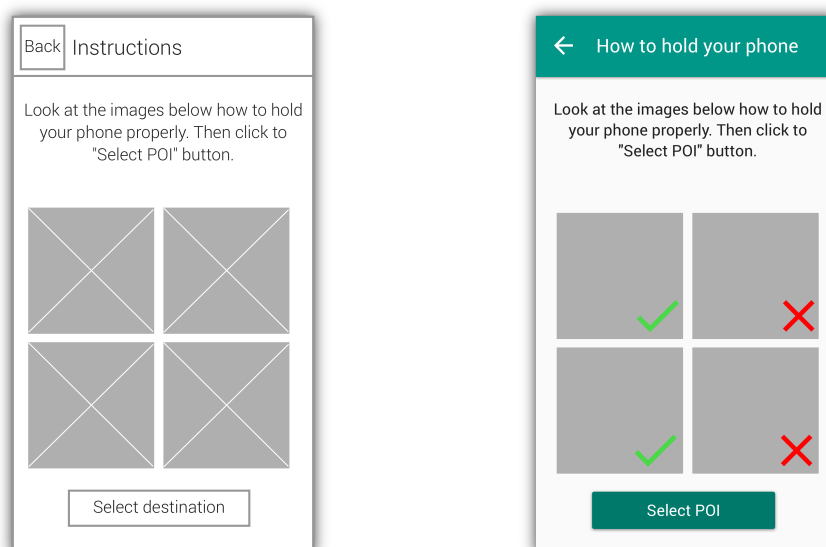
■ Relace mezi obrazovkami

Struktura navigační aplikace a relace mezi obrazovkami je vyobrazena v obrázku 5.22. Šipky charakterizují přechod mezi jednotlivými obrazovkami. Červeně zbarvené šipky odpovídají návratu z aktuální obrazovky na předchozí obrazovku. Modré šipky přechod na další obrazovku na základě uživatelské interakce, kdežto fialové šipky odpovídají přechodům na základě změny stavu lokalizačního či navigačního procesu.

Obrázek 5.22: Relace mezi obrazovkami (*screenflow*)

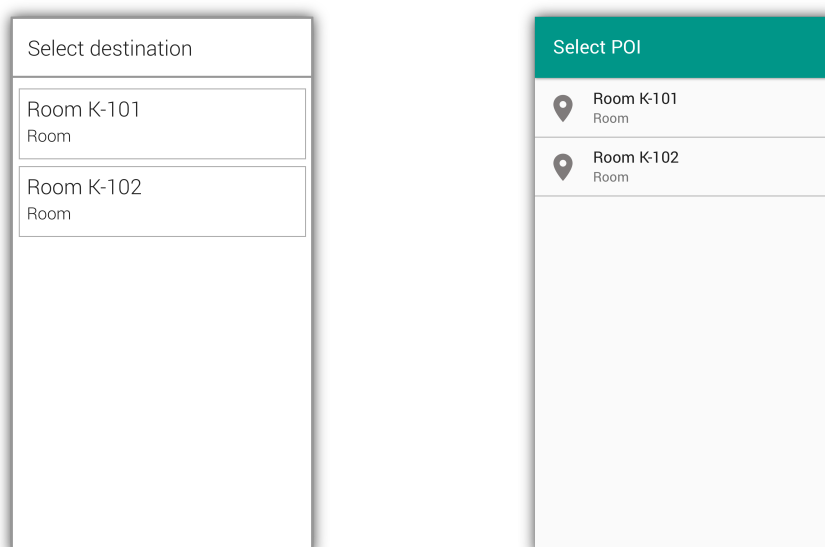
■ Obrazovka instrukcí

Obrazovka instrukcí (viz obrázek 5.23) je první obrazovkou, která se zobrazí po spuštění navigační aplikace. Tato obrazovka má seznámit uživatele s tím, jak má v průběhu navigačního procesu držet mobilní zařízení. Po kliknutí na tlačítko pro výběr význačného bodu (*POI*) aplikace přejde na obrazovku pro výběr cílového bodu.

Obrázek 5.23: Obrazovka instrukcí (*wireframe, mockup*)

■ Obrazovka výběru cíle

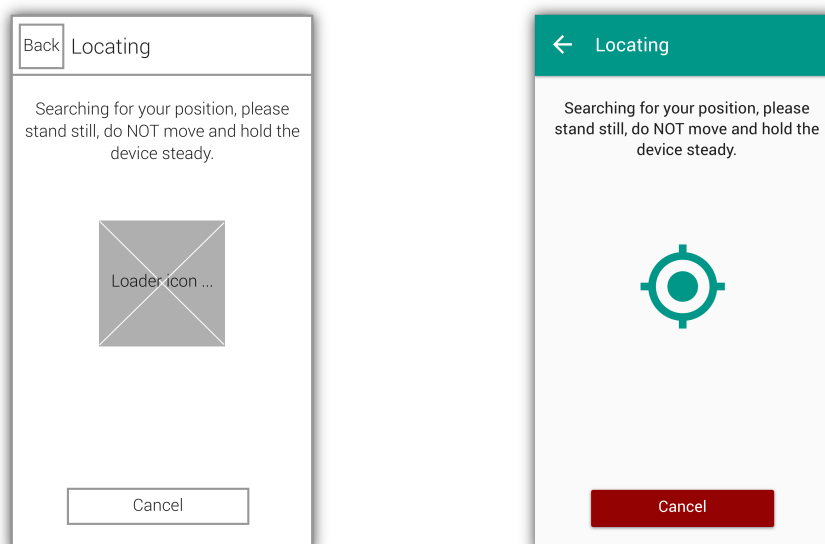
Tato obrazovka (viz obrázek 5.24) obsahuje seznam všech význačných bodů uložených v mapových podkladech. Po kliknutí na konkrétní položku přejde aplikace do režimu lokalizace polohy uživatele, pro který je vyhrazena zvláštní obrazovka.



Obrázek 5.24: Obrazovka výběru cíle (wireframe, mockup)

■ Obrazovka lokalizace uživatele

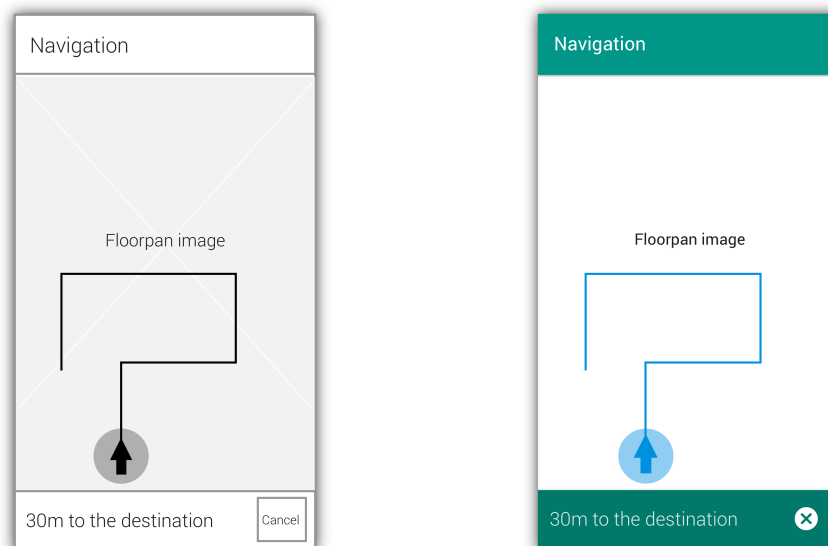
Obrazovka lokalizace (viz obrázek 5.25) žádá uživatele, aby neměnil svou polohu a aby nehýbal mobilním zařízením po dobu, než aplikace nalezne jeho konkrétní polohu. Proces získávání polohy lze zrušit kliknutím na tlačítko pro zrušení, v takovém případě se aplikace vrátí na obrazovku výběru cíle. Poté co je uživatelská poloha zaměřena, zobrazí aplikace obrazovku navigace a spustí navigační proces.



Obrázek 5.25: Obrazovka lokalizace uživatele (wireframe, mockup)

■ Obrazovka navigace

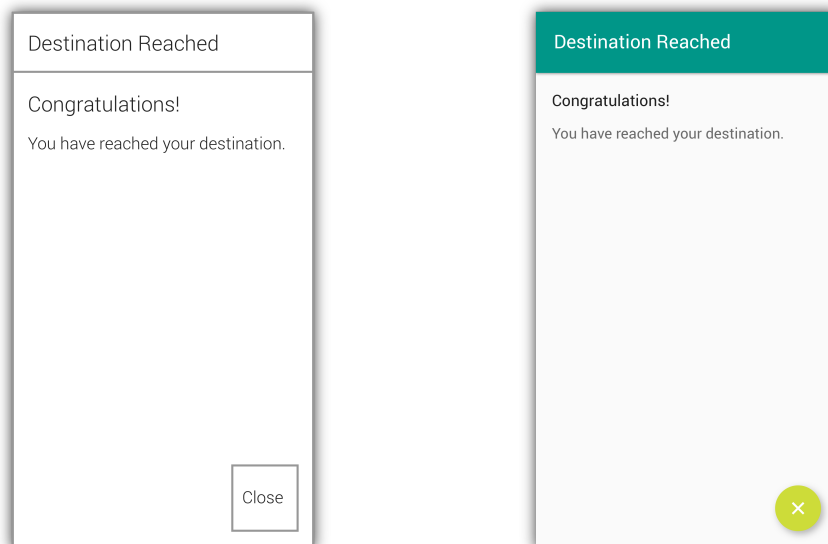
Obrazovka navigace (viz 5.26) slouží k zobrazení mapových podkladů prostředí společně s aktuální polohou uživatele a nejkratší cestou z aktuální polohy do cílového bodu. Aplikace vykresluje společně s aktuální polohou také orientaci uživatele vůči okolnímu prostředí. V případě, že uživatel dosáhne cílového bodu, zobrazí se obrazovka dosažení cíle.



Obrázek 5.26: Obrazovka navigace (*wireframe, mockup*)

■ Obrazovka dosažení cíle

Obrazovka dosažení cíle (viz 5.27) je čistě informativní obrazovka, která uživateli sděluje, že dosáhl cílové lokace. Touto obrazovkou je proces navigace zakončen.



Obrázek 5.27: Obrazovka dosažení cíle (wireframe, mockup)

Kapitola 6

Implementace

Tato kapitola bude obsahovat výběr mobilní platformy, implementaci navržené metody navigace a implementaci výsledných mobilních aplikací.

6.1 Zastoupení operačních systémů

Na trhu s chytrými mobilními zařízeními se nachází několik operačních systémů. Mezi nejrozšířenější mobilní platformy v roce 2015 globálně patřily dle [47] následující operační systémy:

1. Google Android (82.8%)
2. Apple iOS (13.9%)
3. Microsoft Windows Phone (2.6%)
4. Ostatní (0.7%)

6.2 Mobilní platforma

Jako mobilní platformu jsem vybral operační systém *Google Android*. Tuto volbu jsem zvolil díky otevřenosti a rozšířenosti operačního systému *Android*. *Android* je více otevřený a poskytuje vývojáři rozsáhlejší *API*, jak vyčítat data z hardwarových a softwarových senzorů. Dalším argumentem pro výběr této platformy bylo skenování *WirelessFidelity (Wi-Fi)* sítí, které není v konkurenčních operačních systémech možné. Operační systém *Apple iOS* je proprietární systém a běžnému vývojáři s důrazem na ochranu soukromí uživatele poskytuje minimum informací, které by mohli narušit soukromí uživatele. Operační systém *Microsoft Windows* jsem z výběru vyřadil z důvodu uzavřenosti a nízké rozšířenosti oproti vybrané platformě.

6.3 Vývojové prostředí

Na základě výběru vývojové platformy jsem vybral nativní vývojové prostředí pro platformu Android, kterým je *SoftwareDevelopmentKit (SDK) Android Studio*. Toto prostředí umožňuje vývoj jak aplikací psaných v jazyce *JAVA*, tak aplikací psaných

v nativních kódech (*C, C++, ...*) s využitím sady nástrojů Android NDK. Nativní Android aplikace bude psána v jazyce *JAVA* s využitím *Android SDK API*, pomocí kterého se komunikuje s operačním systémem Android. V poslední stabilní verzi *SDK API 23* bylo dle informací o změnách [48] optimalizováno skenování *BLE* zařízení, tudíž minimální podporovaná verze *API* pro vytvořenou *Android* aplikaci bude 23.

6.4 Vývojové zařízení

S ohledem na výběr operačního systému a minimální verzi *SDK API* jsem jako vývojové mobilní zařízení zvolil mobilní telefon *Samsung Galaxy S6 (SM-G920F)* (viz obrázek 6.1) s poslední stabilní verzí operačního systému *Android (6.0.1)*. Hlavními důvody pro volbu tohoto zařízení byla aktuální verze operačního systému, výkonný několika-jádrový procesor pro zpracování a měření testovacích dat, podpora nejmodernějších technologií jako je *Wi-Fi 802.11ac*, *Bluetooth v4.1*, *NFC* a v neposlední řadě široká škála senzorů pohybu a orientace (akcelerometr, gyroskop, magnetometr, krokoměr, ...). Základní technické parametry zařízení jsou zaznamenány v tabulce 6.1. Jako druhé testovací zařízení jsem zvolil *LG Nexus 5*, které také splňuje hardwarové i softwarové požadavky.



Obrázek 6.1: *Samsung Galaxy S6* (zdroj: Samsung.com)

Výrobce, model	Samsung, Galaxy S6 (SM-G920F)
Operační systém	Android 6.0.1 Marshmallow (23)
Display	SuperAmoled 5.1" 2560x1440px
CPU	Exynos 7420 ARMv8-A (A57+A53) 4x@2.1GHz 4x@1.5GHz
RAM	3GB LPDDR4
GSM	2G GSM, 3G WCDMA, 3G TD-SCDMA, 4G FDD LTE, 4G TDD LTE
Bezdrátová rozhraní	ANT/ANT+, Bluetooth 4.1, IEEE 802.11 a/b/g/n/ac 2.4G+5GHz, VHT80 MIMO, Infrared (IrDA), NFC
Navigace	A-GPS, GLONASS, Beidou
Senzory	accelerometer, ambient light sensor, baro- meter, geomagnetic sensor, gesture, gyro sensor, hall sensor, heart rate, proximity sensor

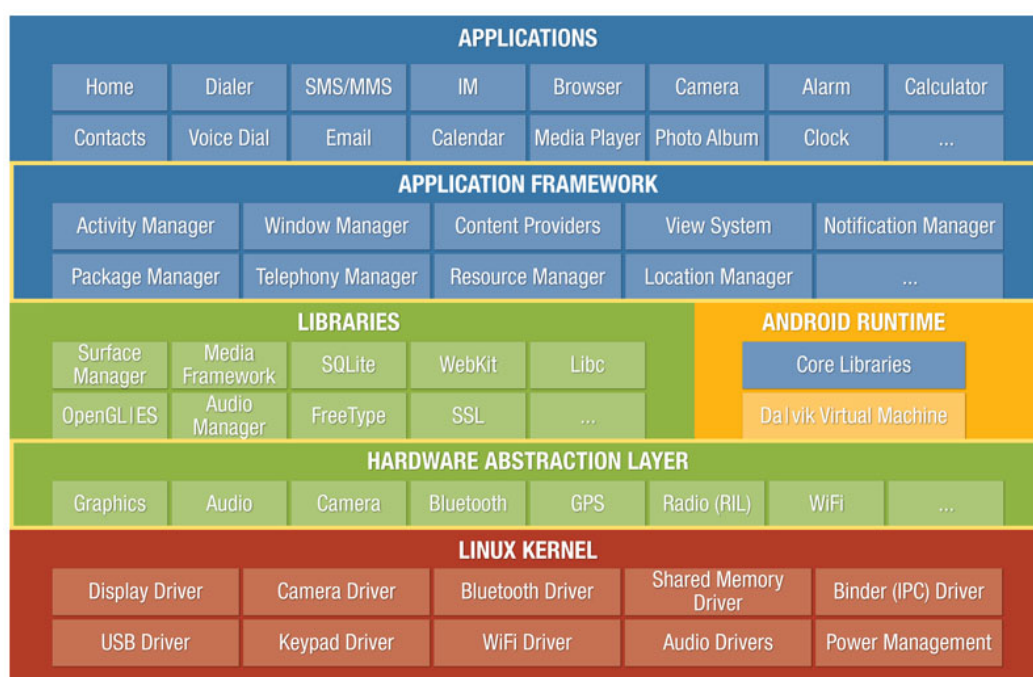
Tabulka 6.1: Základní technické parametry vývojového zařízení

6.4.1 Senzory vývojového zařízení

V přílohách v tabulce B.1 je seznam všech dostupných senzorů vývojového zařízení. Z tabulky je patrné, že akcelerometr a gyroskop jsou ve společném čipu a že lze získávat data z akcelerometru a gyroskopu s frekvencí až 200Hz, respektive minimální prodleva je 5ms. Dále je patrné, že zařízení obsahuje hardwarový krokoměr. Za zmínku stojí, že výrobce vybavil telefon i senzory, které nejsou v operačním systému Android standardně přítomny. Jedná se například o senzor srdečního tepu (*HRM*).

6.5 Architektura systému Android

Obrázek 6.2 popisuje architekturu operačního systému *Android*. Dle obrázku je architektura systému rozdělena do několika vrstev. Nejnížší vrstvou je jádro operačního systému postavené na *Linuxu*. *Linuxové* jádro obsahuje ovladače jednotlivých zařízení a periférií. Nad touto vrstvou je abstrakční hardwarová vrstva (anglicky *hardware abstraction layer*), která poskytuje rozhraní pro komunikaci s jednotlivými zařízeními a perifériemi bez závislosti na konkrétních ovladačích daného zařízení. Nad touto vrstvou běží samotný systém *Android*, který zajišťuje přes vrstvu *application framework* právě komunikaci mezi jednotlivými aplikacemi a zařízeními. Samotné *Android* aplikace tvoří nejvyšší vrstvu architektury systému.



Obrázek 6.2: Architektura operačního systému *Android* (převzato z [49])

6.6 Data ze senzorů

Klíčovou částí aplikací, které implementují navržené řešení, je čtení dat z jednotlivých senzorů mobilního zařízení. Pro přístup k datům z hardwarových senzorů a zařízení je nutné použít *API*, které poskytuje vrstva *application framework*. Implementované aplikace vyžadují senzory a zařízení, které jsou společně se službami poskytujícími přístup k datům, vypsány v následujícím seznamu 6.6:

Wi-Fi android.hardware.wifi (WifiManager)

Bluetooth Low Edition android.hardware.bluetooth_le (BluetoothManager)

GPS android.hardware.location.gps (LocationManager)

Gyroskop android.hardware.sensor.gyroscope (SensorManager)

Akcelerometr android.hardware.sensor.accelerometer (SensorManager)

Magnetometr android.hardware.sensor.compass (SensorManager)

Krokoměr android.hardware.sensor.step_counter (SensorManager)

Čtení dat je realizováno registrováním posluchače (anglicky *listener*) k dané službě. Jelikož jsou data poskytována asynchronně, je nutné při registraci posluchače definovat maximální prodleva mezi vzorky měření dat. Tato prodleva je zdola limitována hardwarovými omezeními jednotlivých senzorů a zařízení. Tyto limity jsou uvedeny v tabulce s výpisem senzorů testovacího zařízení v příloze B.1. Tím,

že Android neposkytuje čtení dat v pevně definovaném intervalu, je nutné tato data vzorkovat. Pevně definovaný interval, neboli konstantní frekvence, čtení dat je nutná zejména pro filtrování a případnou integraci naměřených hodnot.

6.7 Aplikace pro vytváření mapových podkladů

Dle návrhu jsem implementoval nativní Android aplikaci pro vytváření mapových podkladů, která je použitelná na mobilních zařízeních. Takto vytvořené mapové podklady musí být použitelné v aplikaci pro navigaci ve vnitřním prostředí, která je implementována dále.

6.7.1 Komponenta pro vykreslování mapových podkladů

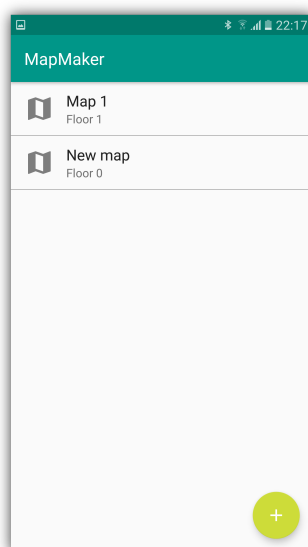
Pro aplikaci na vytváření mapových podkladů a pro navigační aplikaci bylo nutné implementovat grafickou komponentu, která vykresluje mapové podklady, umožňuje uživateli manipulaci s objekty a umožňuje navigovat uživatele v rámci mapy. Implementace je realizována třídou `MapView`, která rozšiřuje nativní komponentu `SurfaceView`. Tato třída se stará o zobrazení grafického výstupu a interakci s uživatelem. `MapView` je součástí abstraktní třídy `Map`, která zajišťuje propojení grafické komponenty s mapovými podklady (třída `MapData`) a třídou pro vykreslování (`MapRenderer`). Třída `Map` je abstraktní implementací, jelikož navržené aplikace vyžadují různé použití mapové komponenty a je zřejmé, že některé procesy a způsoby interakce s mapou se budou lišit na základě různého použití. Například v případě návrhu mapových podkladů je nutné na mapě měřit různá data a vytvářet objekty, kdežto v navigační aplikaci komponenta pouze vykresluje cestu s aktuální polohou. Podrobnější popis této komponenty v této práci není nutný. Konkrétní implementaci lze nalézt ve zdrojovém kódu v balíčku `cz.mvoda.map.common`.

6.7.2 Grafické rozhraní

Grafické rozhraní a struktura obrazovek aplikace byly vytvořeny na základě návrhu s minimálním počtem změn v grafickém prostředí.

Hlavní obrazovka

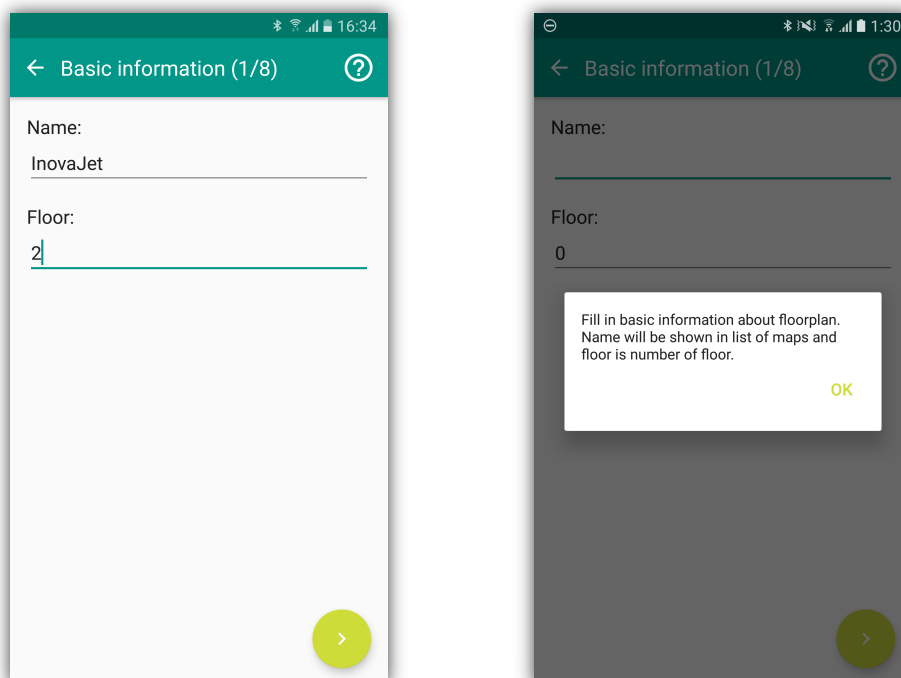
Hlavní obrazovka aplikace byla implementována na základě navrženého *mockupu* (viz snímek obrazovky 6.3). Hlavní obrazovka vypisuje seznam mapových podkladů získaný ze třídy `MapDataProvider`.



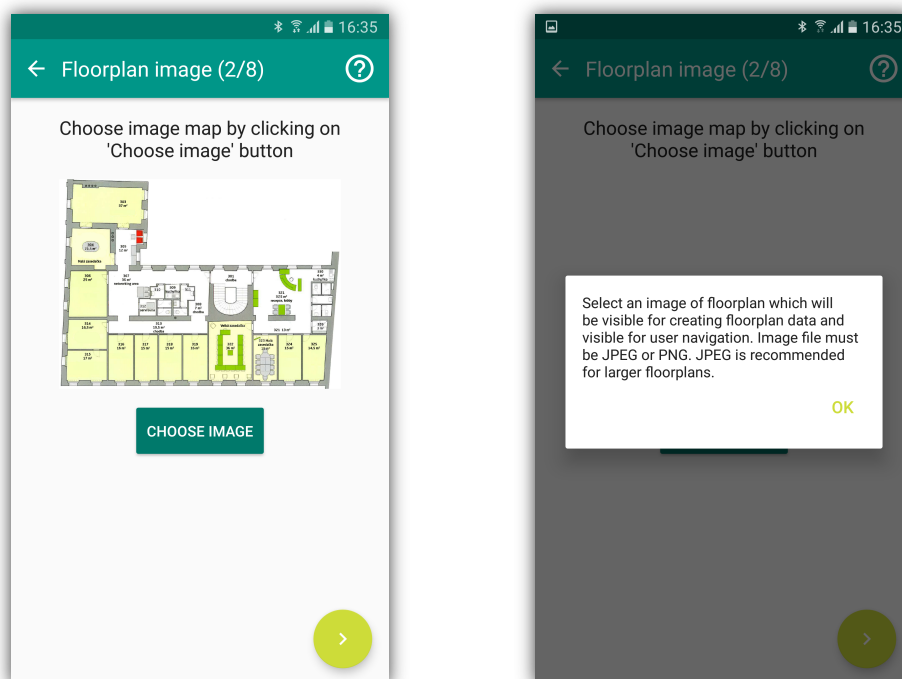
Obrázek 6.3: Hlavní obrazovka (snímek obrazovky)

■ Obrazovky vytváření mapového podkladu

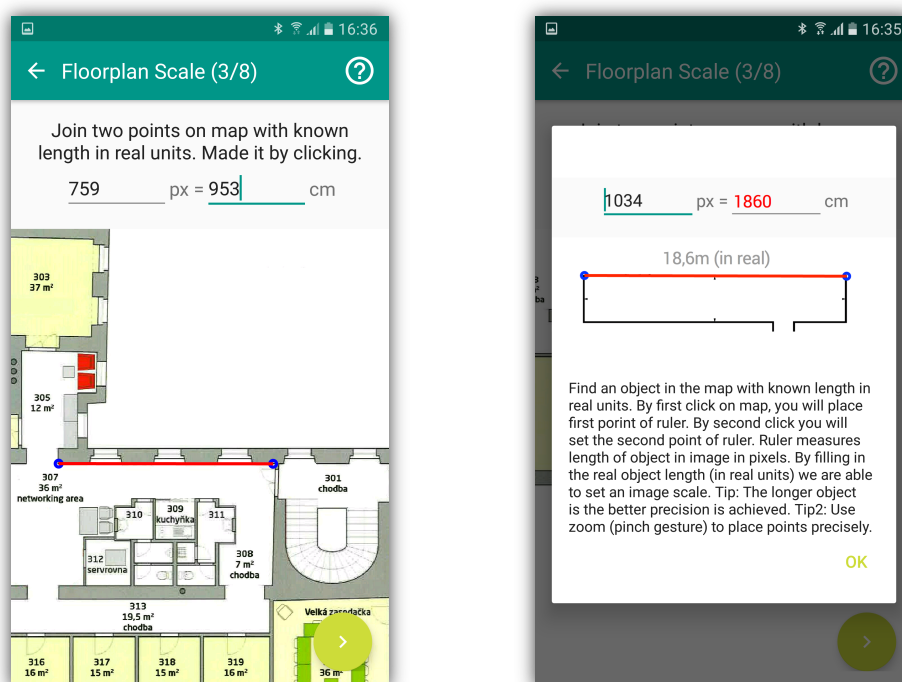
Tato podsekcce obsahuje snímky obrazovek konkrétní implementace navrženého uživatelského rozhraní. Implementace respektuje navržené *mockupy* a nijak se zásadně neliší od návrhu.



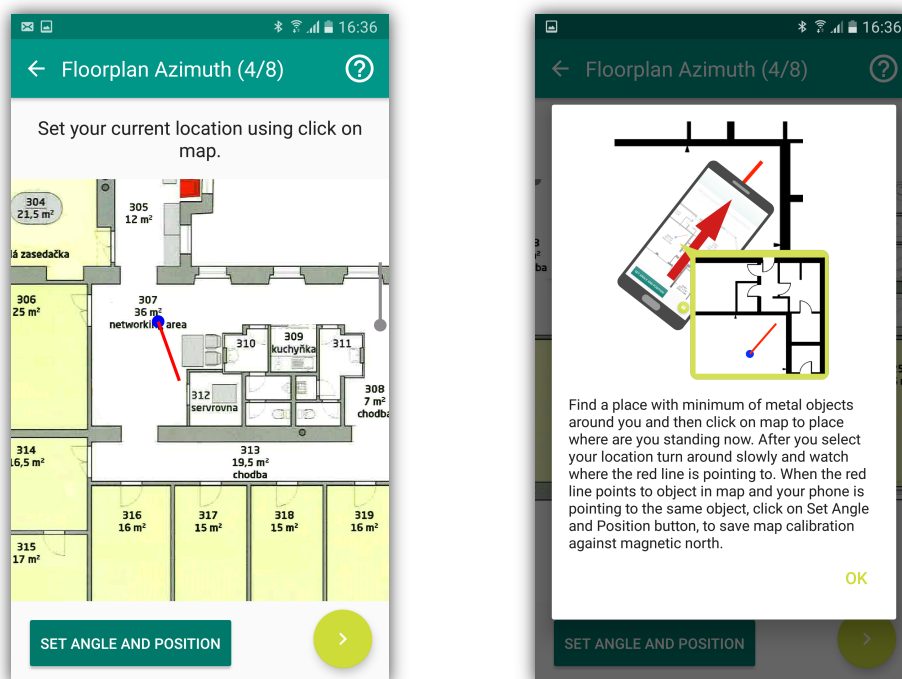
Obrázek 6.4: Krok 1 a nápověda (snímek obrazovky)



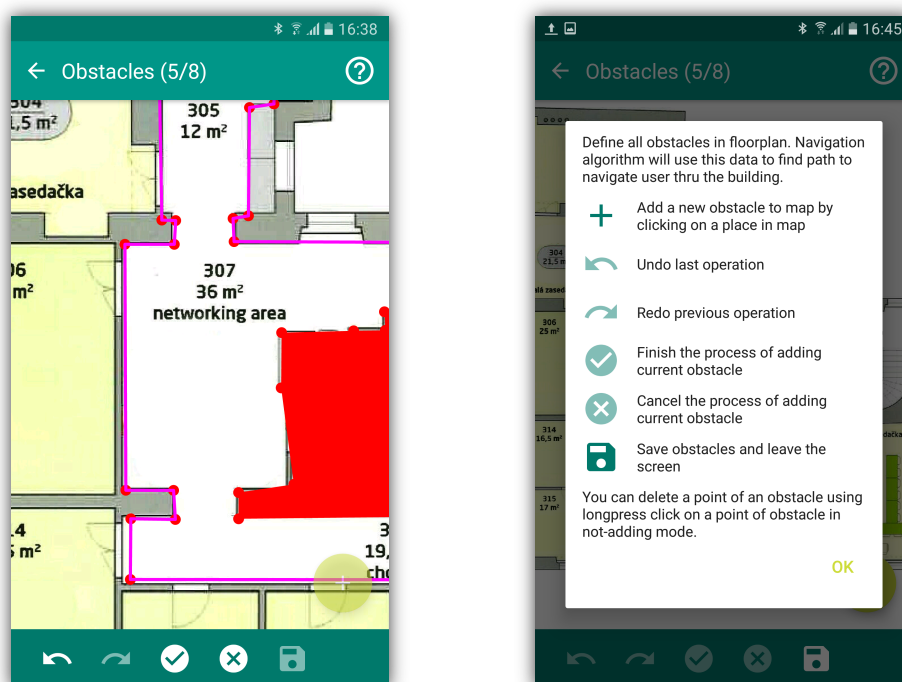
Obrázek 6.5: Krok 2 a nápověda (snímek obrazovky)



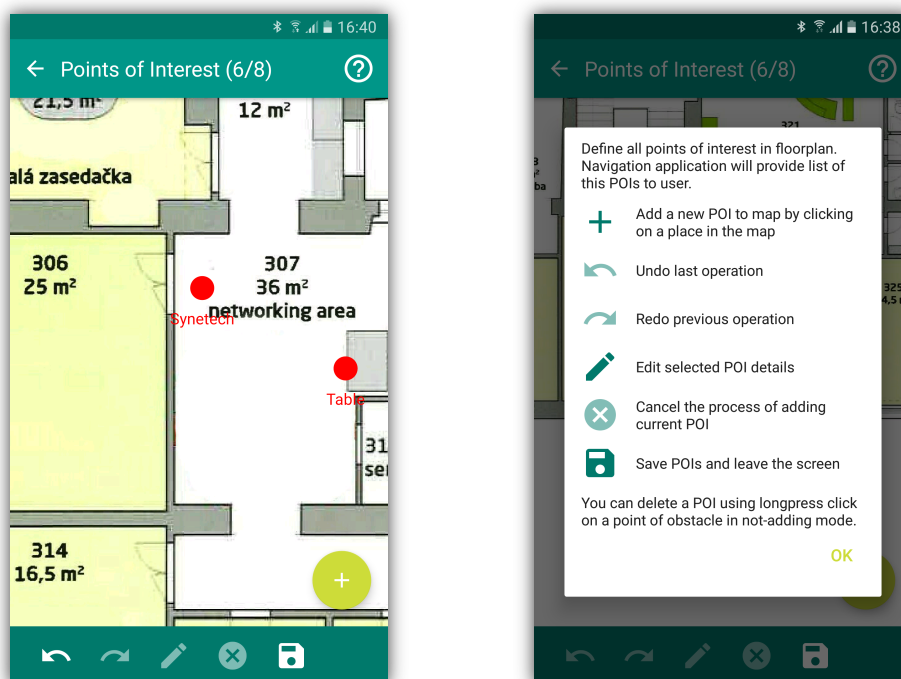
Obrázek 6.6: Krok 3 a nápověda (snímek obrazovky)



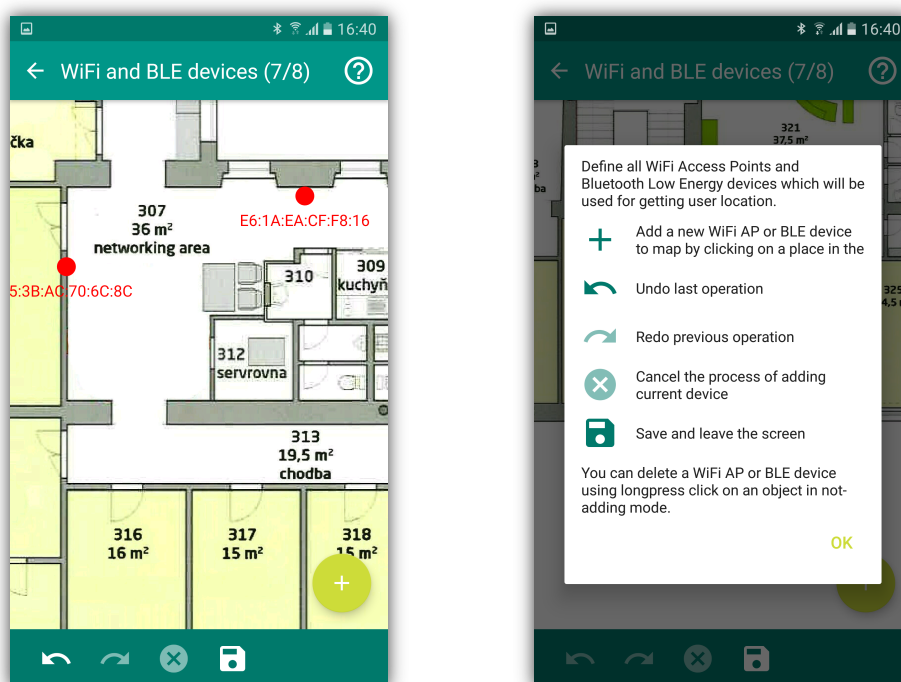
Obrázek 6.7: Krok 4 a nápověda (snímek obrazovky)



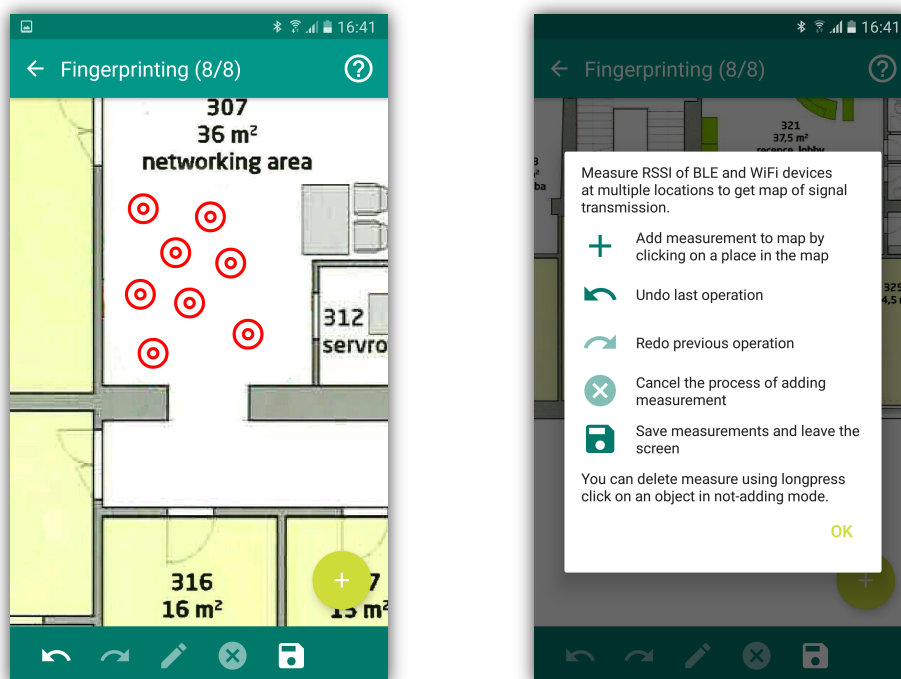
Obrázek 6.8: Krok 5 a nápověda (snímek obrazovky)



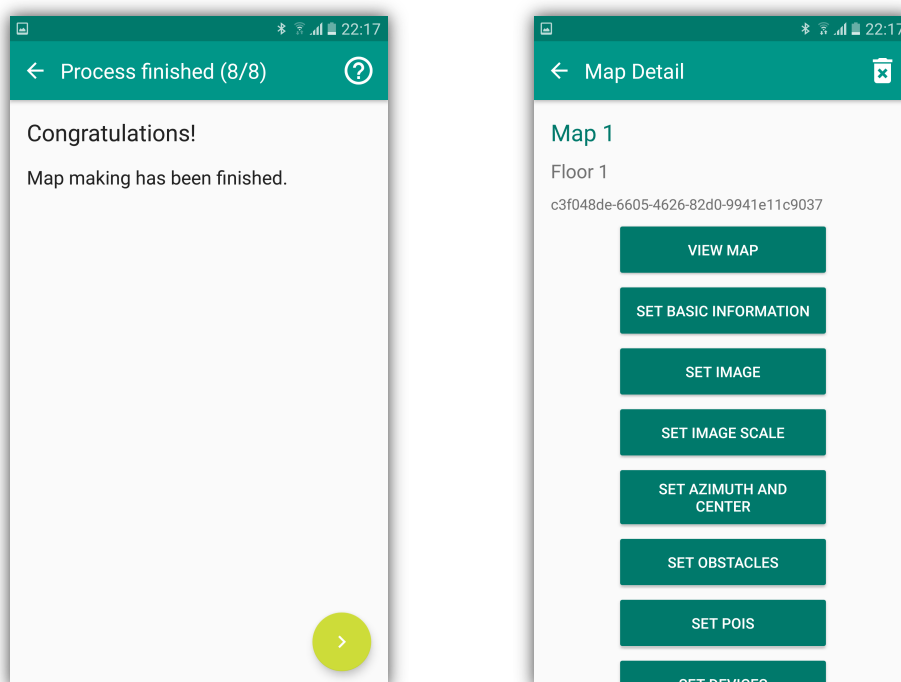
Obrázek 6.9: Krok 6 a nápověda (snímek obrazovky)



Obrázek 6.10: Krok 7 a nápověda (snímek obrazovky)



Obrázek 6.11: Krok 8 a nápověda (snímek obrazovky)



Obrázek 6.12: Krok 9 a obrazovka Detail (snímek obrazovky)

■ 6.7.3 Reprezentace a persistence map

Reprezentaci map jsem zvolil vytvořením tzv. PlainOldJavaObject (POJO) objektů, které reprezentují navrženou strukturu z kapitoly 5. Kořenem této stromové struktury je objekt třídy `MapModel`, který obsahuje objekty či pole objektů charakterizující mapu prostředí. Takto reprezentované mapové podklady se serializují do textového formátu JSON s využitím knihovny *GSON*¹. Takto serializovaná data jsou uložena do souboru v paměti mobilního zařízení, konkrétně v podsložce `MapMaker` ve složce určené pro dokumenty. Takto uložené mapové podklady je možné jednoduše distribuovat mezi zařízeními. Pro manipulaci s těmito soubory se používá *singleton* třída `MapDataProvider`.

■ 6.8 Lokalizace

Pro lokalizaci uživatele v reálném čase bylo potřeba implementovat navržené metody z kapitoly 5, přičemž bylo nutné implementovat krokoměr, senzor orientace, dolnopropustní filtr, částicový filtr a skener bezdrátových zařízení (*WiFi*, *Bluetooth Low Edition*).

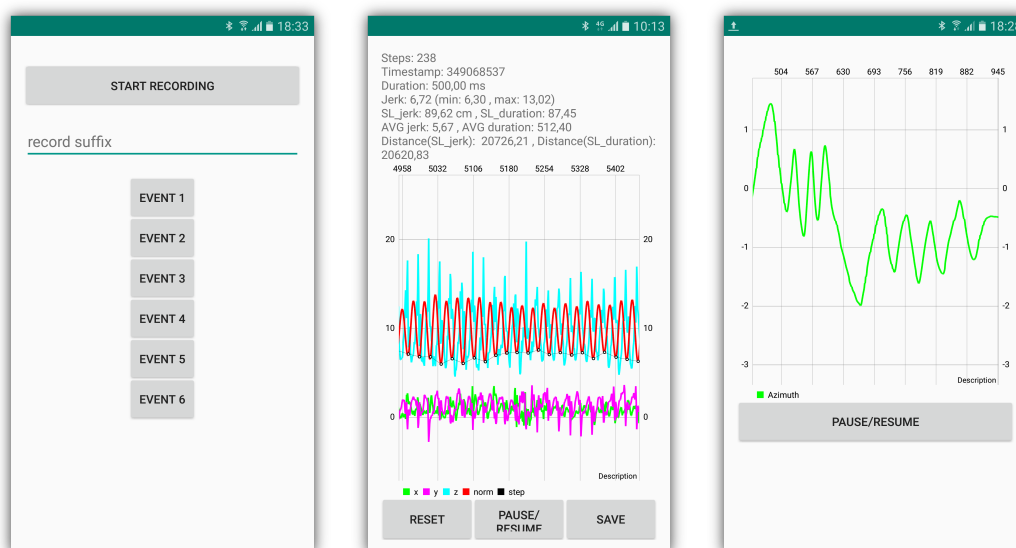
■ 6.8.1 Nativní krokoměr a senzor orientace

Obě výše zmíněná mobilní zařízení disponují nativním krokoměrem, který je optimalizovaný a realizovaný pro správnou detekci kroků. Operační systém Android umožňuje detekovat kroky pomocí asynchronních událostí. Tyto události jsou generovány s minimálním zpožděním a lze je tak použít za účelem detekce kroků. Aby proces detekce kroků minimalizoval šanci generování falešně detekovaných kroků, jsou tyto události generované až po několika po sobě jdoucích krocích, tedy až si je systém s jistotou vědom, že uživatel opravdu chodí. Díky této prodlevě dojde k tomu, že pro několik prvních kroků není generována událost, což v případě použití pro přesnou detekci ušlé vzdálenosti v reálném čase není možné. Pokud by uživatel ušel pouze dva kroky, které by nebyly detekovány, mohl by uživatel změnit svou polohu o několik metrů. Z tohoto důvodu bylo nutné implementovat vlastní krokoměr, který reaguje již na první krok. Implementace tohoto krokoměru bez tolerančního intervalu zvyšuje pravděpodobnost, že aplikace detekuje krok v případě, že se jednalo pouze o pohyb podobný kroku v rámci používání mobilního zařízení. Aplikace je navržena tak, aby nebylo nutné zařízení ovládat během navigace, tím se sníží pravděpodobnost detekce falešného kroku způsobeného ovládáním zařízení. Implementace procesu detekce kroku odpovídá algoritmu 5. Za účelem určení orientace jsou mobilní telefony se systémem Android vybaveny softwarovým senzorem orientace. Při pokusu použít tento nativní senzor orientace se ukázalo, že vliv magnetického rušení má značný vliv na odhadnutý azimut. Proto byl implementován kompas dle návrhu řešení - použití komplementárního filtru na azimut z integrace gyroskopu a na azimut získaný akcelerometru a magnetometru.

¹<https://github.com/google/gson>

6.8.2 Podpůrné aplikace

Aby bylo možné získat a analyzovat data z vývojového zařízení, vytvořil jsem několik podpůrných Android aplikací. Aplikace **MotionRecorder** je schopna souběžně nahrávat data z několika senzorů a ukládat je do souborů typu *CommaSeparatedView* (CSV). Aplikace je také schopna skenovat *Wi-Fi* sítě a všechna dosažitelná *BLE* zařízení. Pro analýzu naměřených dat jsem vytvořil skript pro aplikaci **MATLAB**, která po připojení mobilního telefonu s aplikací **MotionRecorder** v režimu vývojáře, je schopna zaznamenaná data zkopírovat, vykreslit a analyzovat. Aplikace dále umožňuje uživateli zaznamenat až 6 asynchronních, uživatelských událostí, které se poté promítnou do grafů. Snímek 6.13 zachycuje obrazovky podpůrných aplikací. Příložené grafy v kapitole 2 jsou výstupem skriptu v *MATLABu* s naměřenými daty z vývojového zařízení (*Samsung Galaxy S6*) pomocí této aplikace. Druhou implementovanou aplikací je aplikace **Pedometer**, která zobrazuje data z akcelerometru a krokoměru. Tato aplikace byla nutná k optimalizaci algoritmu krokoměru. Třetí podpůrnou aplikací je aplikace **Azimuth**, která zobrazuje data z implementovaného kompasu v reálném čase.



Obrázek 6.13: Podpůrné aplikace (MotionRecorder,Pedometer,Azimuth)

6.8.3 Implementace krokoměru

Krokoměr byl implementován dle algoritmu 5 návrhu řešení. Implementace je realizována třídou `AdaptivPedometer`, která obsahuje metodu

```
boolean processSample(long timestamp, double x, double y, double z)
```

, kde

timestamp je čas vzorku v milisekundách,

x,y,z jsou jednotlivé složky akcelerace,

návratová hodnota určuje detekci kroku.

Metoda vrací pozitivní logickou hodnotu pouze v případě, že je pro daný vzorek detekován krok. Při detekci kroku se také volá metoda `onNewStepDetected` pro přiřazeného posluchače.

```
void onNewStepDetected(long tStepCenterTimestamp, double stepDuration,  
                        double stepLength, double aMin, double aMax)
```

, kde

timestamp je průměr časů mezi extrémy akcelerace v milisekundách,

stepDuration je délka kroku v milisekundách,

stepLength je délka kroku v cm,

aMin, aMax jsou hodnoty extrémů normy akcelerace.

Při detekci kroku dochází k prodlevě, která je způsobena dolnoproustním filtrem typu *Butterworth* šestého stupně s mezní frekvencí 2.5Hz . Tento filtr odstraňuje vysokofrekvenční složky z hodnot akcelerace, které nejsou charakteristické pro běžnou chůzi.

6.8.4 Implementace kompasu

Kompas byl implementován dle navrženého algoritmu v sekci 5 návrhu řešení. Kompas charakterizuje třída `AzimuthSensorFusionEstimator`, která obsahuje metodu

```
void onSensorChanged(SensorEvent event)
```

, kde

event je objekt obsahující data z akcelerometru, gyroskopu či magnetometru.

Metoda `onSensorChanged` je asynchronně volána službou `SensorManager`, která poskytuje data interních senzorů. Jelikož jsou data poskytována asynchronně, obsahuje třída časovač (*timer*), který zajišťuje správnou funkčnost komplementárního filtru, který kombinuje data z gyroskopu a magnetometru.

Pro asynchronní zjištění aktuálního azimutu je určena metoda `getLastAngle`, která vrací poslední odhadnutou hodnotu azimutu. Třída `AzimuthSensorFusionEstimator` mimo jiné obsahuje metodu `getDelayedAngle`, která vrací hodnotu úhlu před určitou časovou konstantou. Tato metoda se používá při výpočtu směru kroku, protože kroky jsou díky filtraci zpožděné o určitou konstantu a mohlo by dojít ke špatnému odhadu směru kroku.

```
double getLastAngle()
double getDelayedAngle()
```

, kde

návratová hodnota určuje velikost azimutu v radiánech.

6.8.5 Implementace lokalizace pomocí bezdrátových technologií (fingerprinting)

Pro zjištění absolutní polohy byla implementována metoda lokalizace pomocí bezdrátových technologií *WiFi* a *Bluetooth Low Edition*, založená na měření hodnot síly přijatého signálu. Třída implementující tuto metodu se nazývá `FingerprintingLocationManager`. Při inicializaci třídy je nutné poskytnout množinu referenčních bodů. Skenování okolních *WiFi* sítí a *BLE* zařízení za účelem sběru hodnot síly signálu pro konkrétní zařízení se spustí po zavolání metody `start`. Metodou `stop` lze skenování ukončit. Pro získání odhadu polohy na základě naměřených dat slouží metoda `getLocation`.

```
PointModel getLocation(boolean weighted, boolean bleEnabled, boolean wifiEnabled)
```

, kde

weighted určuje, zda se má použít algoritmus *KNN* či *NN*,

bleEnabled určuje, zda se má při odhadu použít měření z *BLE* zařízení,

wifiEnabled určuje, zda se má při odhadu použít měření z *WiFi* zařízení,

návratová hodnota obsahuje souřadnice polohy či hodnotu `null` v případě, že poloha nebyla nalezena.

Alternativní způsob zjištění polohy je metoda `getLocation` bez parametrů, která vrací objekt obsahující polohy získané z různých zdrojů měření (*WiFi*, *BLE*, *WiFi* + *BLE*). Tato varianta je vhodná v případě, že není známo, zda je v daném místě dostupný zdroj měření, například pokud některá místa nejsou pokryta jednou z technologií.

6.8.6 Částicový filtr

Pro zamezení nepřesného a nereálného určení polohy mimo dosažitelné části mapy byl implementován částicový filtr, který pomocí částic reprezentovaných třídou `Particle`, modeluje aktuální polohu uživatele. Pro správnou funkčnost částicového

filtru je nutné vygenerovat počáteční množinu částic reprezentující apriorní pravděpodobnost polohy uživatele. Generování počátečních částic probíhá pomocí metody `generateRandomParticlesInRadius`, která přidá do množiny částice, které jsou rovnoměrně rozmístěné v radiusu určitého bodu, jímž je počáteční odhad polohy uživatele. Poté se při každém kroku volá metoda `moveAllParticles`, která na základě informací o kroku přemístí částice daným směrem. Částice, které "narazí" do neprůchozí překážky, jsou z množiny odstraněny. Při detekci průchodu částice překážkou je nutné znát rozmístění překážek, které poskytují mapové podklady. Výsledná poloha dle částicového filtru je dána průměrováním částic v množině. V případě binární pravděpodobnosti pozorování (validní/nevalidní částice) není nutné částice převzorkovávat (anglicky *resampling*). V případě odstranění částic je nutné náhodně vybrat částice z původní množiny částic a přemístit je.

```
void generateRandomParticlesInRadius(boolean clear, int number,
                                     double x, double y, double r)
```

, kde

clear určuje, zda se má před generováním množina částic vyprázdnit,

number je počet generovaných částic,

x,y jsou souřadnice, okolo kterých budou částice rozmístěny,

r určuje radius v jakém budou částice rozptýleny.

```
boolean moveAllParticles(double angle, double distance,
                         double angleNoise, double distanceNoise)
```

, kde

angle určuje velikost změny orientace uživatele,

distance charakterizuje délku kroku,

angleNoise je hodnota zašumění částic,

distanceNoise je hodnota zašumění délky kroku.

```
ParticlePosition getAverageParticle()
```

, kde

návratová hodnota určuje zprůměrovanou polohu částic.

Struktura částice je reprezentována následovně:

```
public class Particle implements Comparable<Particle> {
    private double x, y, orientation = 0;
    private double weight = 1;
    ...
}
```

, kde

x,y jsou souřadnice částice,

orientation je azimut částice v radiánech,

weight je váha částice.

■ 6.9 Navigace

■ 6.9.1 Hledání cesty

Součástí procesu navigace je hledání cesty z aktuální polohy do polohy cílové. Proces hledání cesty byl implementován za pomoci algoritmu A^* , který hledá cestu mezi překážkami. A^* prohledává stavový prostor, jímž je graf viditelnost. Pro generování grafu viditelnosti pro dané prostředí byla implementována třída **VisibilityGraph**, která z množiny překážek a polygonu hranic mapy vygeneruje graf viditelnosti. Generování grafu viditelnosti je realizováno hledáním párů vrcholů z množiny všech vrcholů daných polygonů. Aby se zamezilo vygenerování cest příliš úzkými mezerami mezi překážkami, zvětšil jsem všechny překážky o několik desítek centimetrů. Samotná detekce viditelnosti dvou vrcholů je realizována s využitím knihovny *JTS*, která poskytuje metody pro zjištění průniků různých geometrických objektů jako jsou polygony, lomené čáry a body. Graf viditelnosti se generuje pouze jednou v rámci jednoho spuštění aplikace a to právě při spouštění. Při následném hledání cesty mezi aktuální polohou a cílovou polohou, se graf viditelnosti rozšíří o bod aktuální polohy a o bod cílové polohy, tudíž není nutné generovat celý graf viditelnosti znovu. Po nalezení cesty se opět body reprezentující aktuální polohu a cílovou polohu z grafu viditelnosti odeberou. O hledání konkrétní cesty v grafu viditelnosti se stará třída **AStar**, jež je inicializována s parametrem obsahujícím graf viditelnosti. Proces hledání cesty se vyvolá metodou **findPath**.

```
List<Coordinate> findPath(Coordinate location, Coordinate destination)
```

, kde

location je aktuální poloha uživatele,

destination je cílová poloha uživatele,

návratová hodnota je seznam souřadnic, jejichž spojením vznikne výsledná cesta.

■ 6.9.2 Výsledná implementace lokalizace a navigace

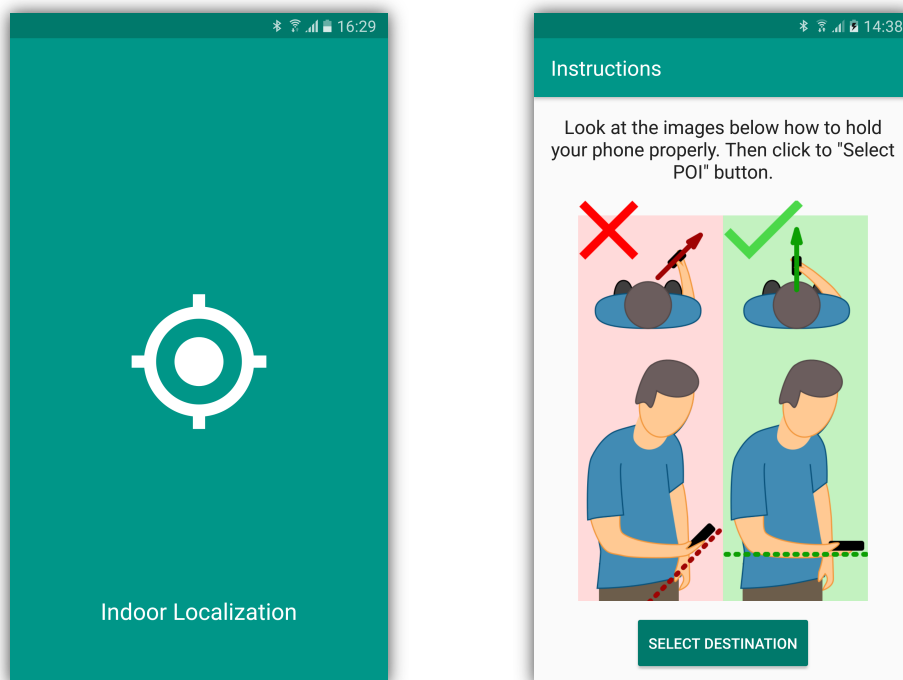
Proces lokalizace a navigace zajišťuje třída **IndoorLocator**, která je schopna uživatele lokalizovat a v případě poskytnutí cílového bodu i navigovat. Po vytvoření instance této třídy s parametry obsahujícími mapové podklady, lze zavolat metodu **start**, která spustí proces lokalizace. Prvním stavem je počáteční zaměření polohy trvající několik sekund, než dojde ke sběru všech potřebných informací pro lokalizaci. Až poté třída přejde do režimu lokalizace/navigace v reálném čase. Níže

uvedené rozhraní `LocatorListener` vystihuje způsob komunikace s třídou používající `IndoorLocator`. Názvy metod vystihují jejich účel, tudíž je nebudu blíže popisovat.

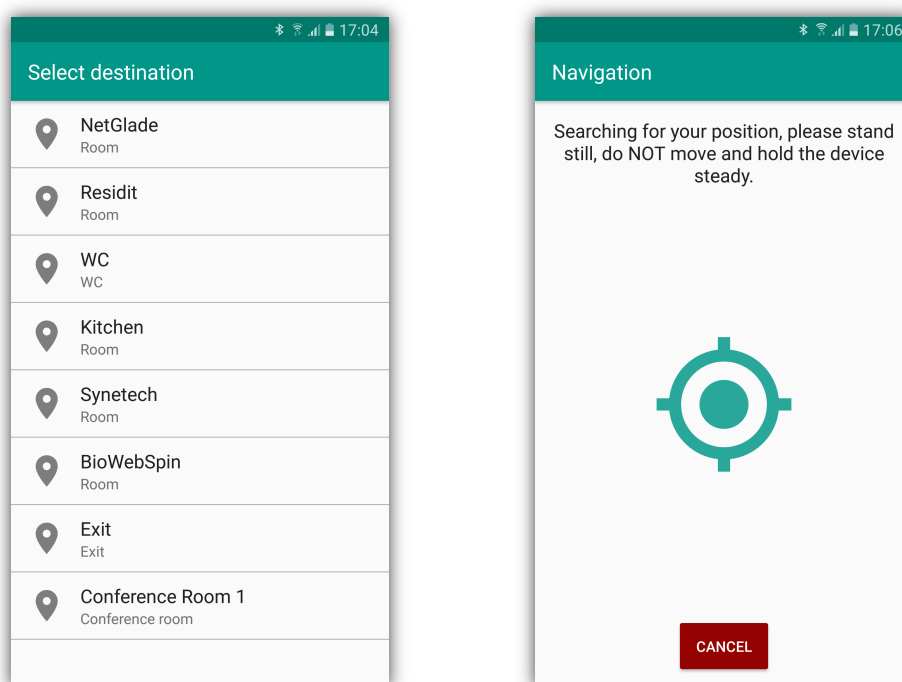
```
interface LocatorListener {
    void onStateChanged(IndoorLocator.LocatorState state);
    void onLocationChanged(ParticlePosition lastPosition, boolean valid);
    void onInitialPositionNotFound();
    void onLocationLost();
    void onDistanceToDestinationUpdated(double distance);
    void onDestinationReached();
}
```

6.9.3 Grafické rozhraní navigační aplikace

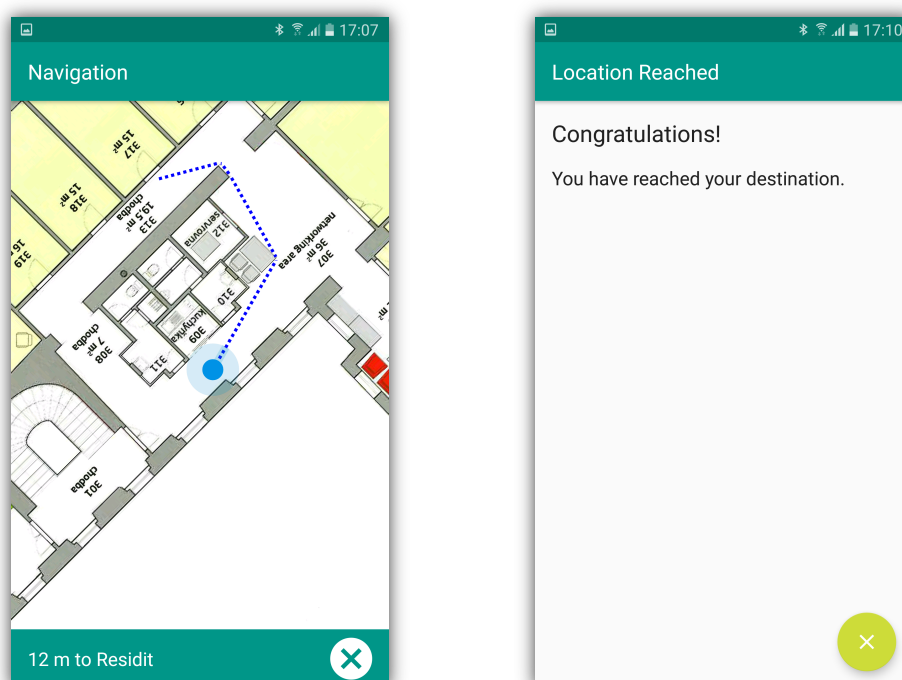
Navigační aplikace byla také implementována na základě navržených *mockupů* s tím rozdílem, že bylo nutné přidat úvodní obrazovku. Úvodní obrazovka je zobrazena při načítání mapy, kdy současně probíhá generování grafu viditelnosti, jež může na některých zařízeních trvat delší dobu.



Obrázek 6.14: Úvodní obrazovka (vlevo) a obrazovka instrukcí (vpravo) (snímek obrazovky)



Obrázek 6.15: Obrazovka výběru cíle a obrazovka lokalizace uživatele (snímek obrazovky)



Obrázek 6.16: Obrazovka navigace a obrazovka dosažení cíle (snímek obrazovky)

■ 6.10 Použité knihovny

Při implementaci aplikace byly použity knihovny ze seznamu 6.10.

GSON² Knihovna pro serializaci a deserializaci dat,

JTS³ knihovna od obsahující metody pro manipulaci s prostorovými objekty (např. polygons),

MPAndroidChart⁴ knihovna pro vykreslování grafů,

Android Gesture Detector⁵ knihovna pro detekci dotykových gest,

Fabric⁶ nástroj pro distribuci a testování mobilních aplikací.

Kapitola 7

Testování

V této kapitole bude otestováno navržené řešení navigace uvnitř budovy a jeho konkrétní implementace pomocí mobilní aplikace.

7.1 Testovací zařízení

Pro účel testování byly vybrány dva mobilní telefony různých výrobců a různých implementací operačního systému *Android*. Prvním zařízením je *Samsung Galaxy S6 (SM-G920F)*, druhým *LG Nexus 5 (D-821)*. Oba mobilní telefony splnily hardwarové a softwarové požadavky stanovené navrženou metodou.

7.2 Testovací prostředí

Testovací prostředí jsem volil dle velikosti rozlohy, která je limitována omezeným počtem mně dostupných *iBeacon* zařízení. Pro testování jsem zvolil prostory podnikatelského inkubátoru *ČVUT - InovaJET*. Obrázek 7.1 zachycuje podobu testovacího prostředí.

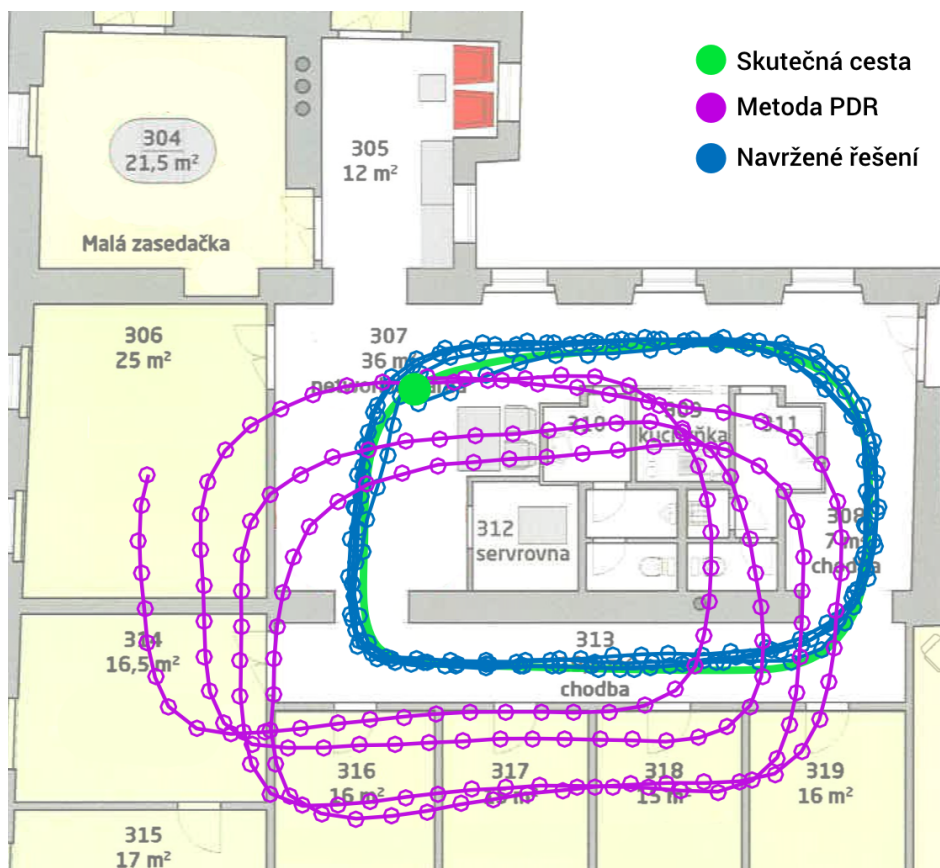


Obrázek 7.1: Plán testovacího prostředí (poskytl:Inovacetrum)

7.3 Testování navržené metody

Testování navržené metody lokalizace probíhalo vyznačením testovací trasy. Přesné souřadnice této trasy nebylo možné zjistit, jelikož jsem neměl dostupný žádný referenční lokalizační systém, který by v reálném čase monitoroval moji polohu. Pro vyhodnocení použitelnosti metody za účelem navigace je použito vizuální porovnání reálné trasy s trasou získanou z navržené metody.

Obrázek 7.2 zachycuje vizuální porovnání metod určení polohy uživatele. Zelenou barvou je vyznačena testovací trasa, která byla čtyřikrát opakována. Fialově je vykreslena trasa odpovídající metodě *PDR*, bez jakékoliv korekce a modrou barvou je výstup navrženého řešení. Při porovnání vykreslených tras je patrná náchylnost metody *PDR* na odchylky odhadu směru a délky kroků v kombinaci s ne vždy přesným držením mobilního zařízení ve směru chůze. Navržené řešení potvrdilo svoji odolnost a schopnost tyto odchylky kompenzovat a je možné jej použít pro navigaci uživatele daným prostředím. Počáteční poloha byla odhadnuta dle naměřených signálů z bezdrátových majáků. Stejným způsobem byla navržená metoda otestována na druhém testovacím zařízení (LG Nexus 5) s naměřenými daty z prvního testovacího zařízení. I přesto, že se jedná o rozdílná zařízení různých výrobců, bylo možné použít navrženou metodu i s naměřenými hodnotami z jiném zařízení.



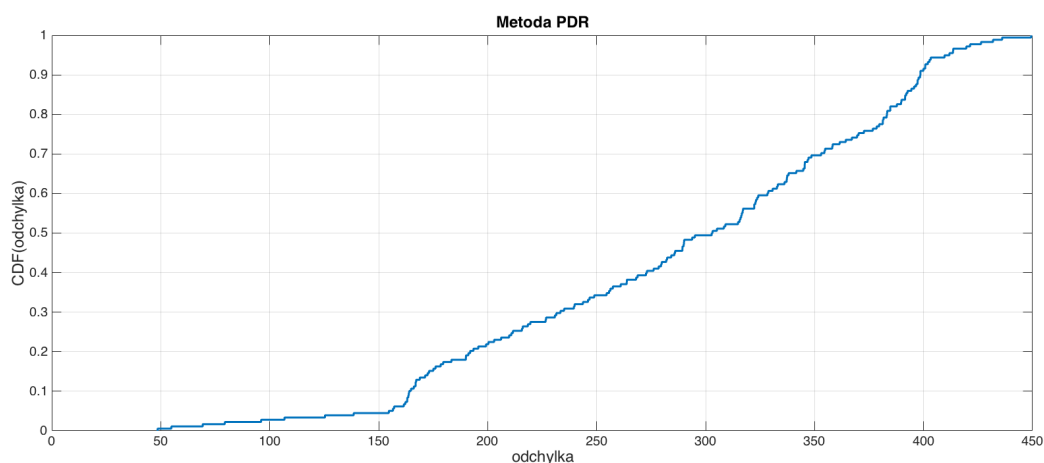
Obrázek 7.2: Porovnání navržené metody s metodou *PDR*

Metoda	Průměrná odch.	Směrodatná odch.	Maximální odch.
PDR	289cm	92cm	450cm
Navržená metoda	81cm	45cm	172cm

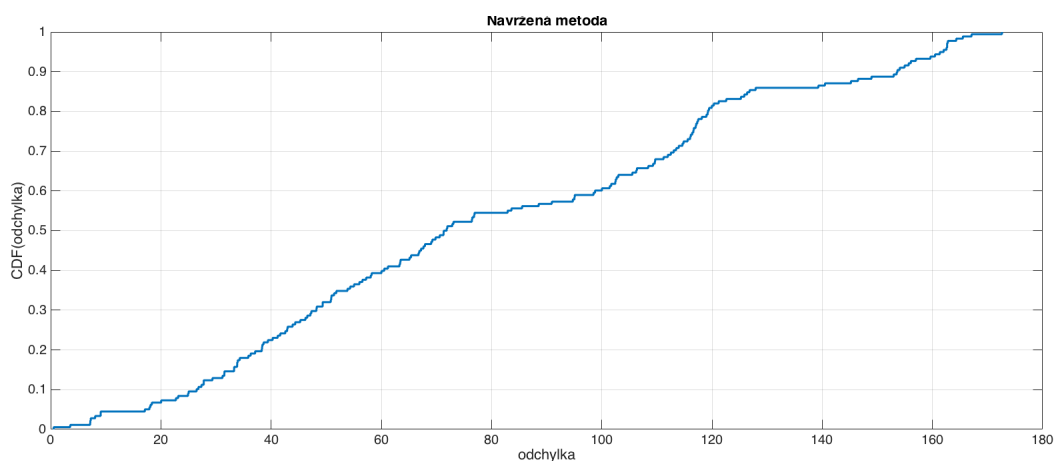
Tabulka 7.1: Porovnání testování metod

Tabulka 7.1 obsahuje výsledky testování jednotlivých navigačních metod. Navržená metoda měla průměrnou odchylku (Euklidovskou vzdálenost) 81cm, kdežto metoda PDR dosahovala průměrné odchylky 289cm od referenční trasy. Velký vliv na přesnost navržené metody má částicový filtr, jež dle návrhu pracuje velmi dobře v členitém prostředí. V otevřeném prostředí by byla odchylka vyšší, protože částice by omezovány překážkami.

Obrázky 7.3 a 7.4 zachycují distribuční funkci odchylky pro jednotlivé metody. Z grafů je patrné, že navržená metoda má odchylku menší než 154cm po 90% času, kdežto metoda PDR má po 90% času odchylku menší než 399cm.



Obrázek 7.3: Distribuční funkce odchylky pro metodu PDR



Obrázek 7.4: Distribuční funkce odchylky pro navrženou metodu

7.3.1 Krokoměr

Přesnost implementovaného krokoměru byla otestována na sedmi osobách, mezi kterými bylo šest mužů a jedna žena. Výška testerů byla v rozmezí od 165cm do 194cm. Kritérii kvality krokoměru bude počet odhadnutých kroků vůči skutečnému počtu a rozdíl délek vzdálenosti. Data byla naměřena podpůrnou aplikací **Pedometer**. Tabulka 7.2 porovnává počet reálně provedených kroků vůči počtu detekovaných kroků. Z výsledků testování krokoměru na sedmi osobách vyšlo, že přesnost detekce počtu kroků je 99%, což je pro účely navigace dostačující. V rámci testovací trasy bylo naměřeno 179 kroků ze 179 skutečných.

Test č.	Skutečný p.kroků	Detekovaný p.kroků	Přesnost
1	433	431	99,5%
2	85	84	98,9%
3	81	80	98,7%
4	85	83	97,6%
5	49	49	100%
6	90	89	98,9%
7	93	92	98,9%
Celkem	916	908	99,1%

Tabulka 7.2: Krokoměr - porovnání počtu kroků

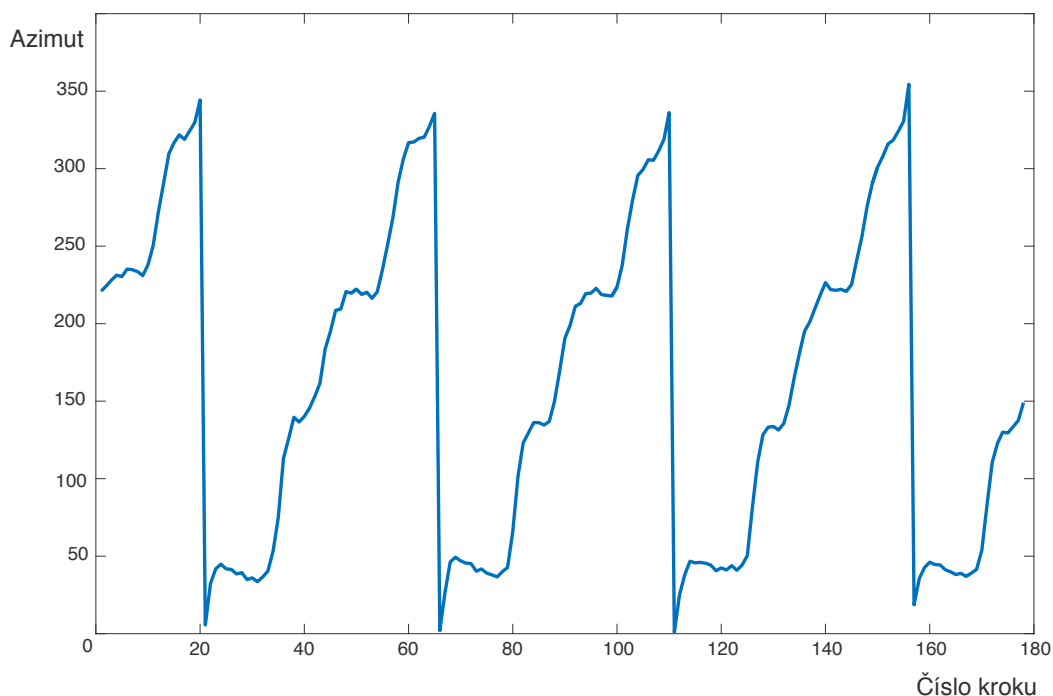
Tabulka 7.3 porovnává délku reálně ušlé vzdálenosti vůči odhadnuté vzdálenosti z implementovaného krokoměru. Z výsledků testování na sedmi osobách vyšlo, že přesnost detekce ušlé vzdálenosti na množině testerů je přibližně 93%. Na testování jsem vyznamenal, že pro běžný styl chůze odhad funguje relativně přesně a nezáleží ani tak na výšce testované osoby, jako na jejím stylu chůze. U testerů, kteří dělali velmi malé kroky, byla přesnost výrazně nižší.

Test č.	Skutečná vz.	Odhadnutá vz.	Odchylka	Přesnost
1	928cm	896cm	32cm	96.55%
2	938cm	926cm	12cm	98.72%
3	963cm	968cm	5cm	99.48%
4	1080cm	1145	65cm	93.98%
5	948cm	873cm	75cm	92.09%
6	940cm	1025cm	85cm	90.96%
7	1050cm	1270cm	220	79.05%
Celkem	6847cm	-	494 cm	92.79%

Tabulka 7.3: Krokoměr - porovnání délky kroků

7.3.2 Kompas

Testování kompasu bylo provedeno podpůrnou aplikací **Azimuth**. Prostřednictvím této aplikace jsem zkoumal stabilitu odhadnutého azimutu a citlivost na magnetická rušení. Při porovnání implementace kompasu pomocí komplementárního filtru s implementací kompasu odhadující azimut pouze pomocí magnetometru, byl rozdíl v citlivosti na magnetická rušení zcela zřejmý. V hodnotách azimutu pouze z magnetometru, bylo při průchodu zarušenou částí patrné nárazové vychýlení odhadnutého úhlu. Při průchodu stejnou částí prostředí v případě kombinovaného kompasu byla potvrzena vyšší odolnost proti nárazovému vychýlení úhlu v případě magnetického rušení. Pokud jsem na silně magneticky zarušeném místě setrval delší dobu, azimut po určitém čase konvergoval do hodnoty, kterou skokově registrovala implementace nekombinovaného kompasu. Obrázek 7.5 zachycuje azimut uživatele v průběhu testovací cesty. V grafu je patrné několikanásobné opakování testovací trasy.



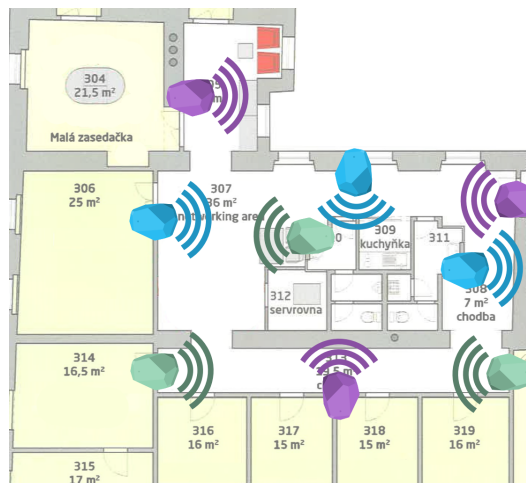
Obrázek 7.5: Azimut uživatele při chůzi

7.3.3 Bezdrátové majáky

Použité majáky

Pro testování jsem použil devět *BLE* majáků od firmy *Estimote, Inc.*. Majáky byly nastaveny do režimu *iBeacon* na interval vysílání *beacon* rámce každých 100ms s nastavením vysílacího výkonu na -8dBm a byly umístěny do výšky 180cm nad podlahou.

Obrázek 7.6 Zachycuje rozmístění *iBeacon* majáků v testovacím prostředí.

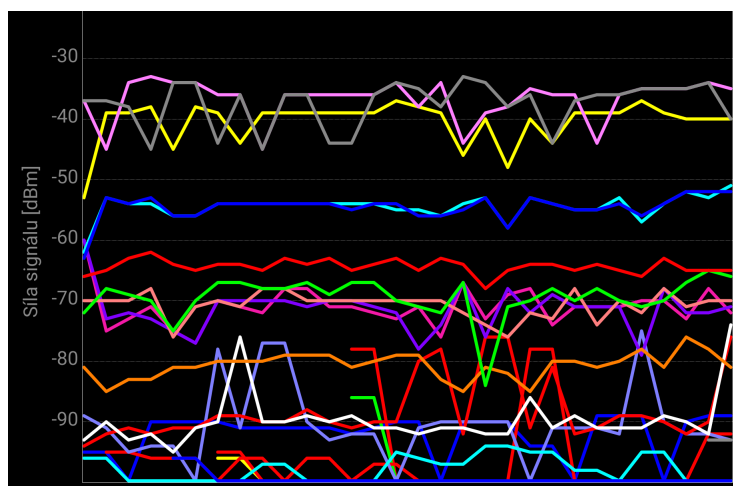


Obrázek 7.6: Rozmístění *iBeacon* majáků v testovacím prostředí

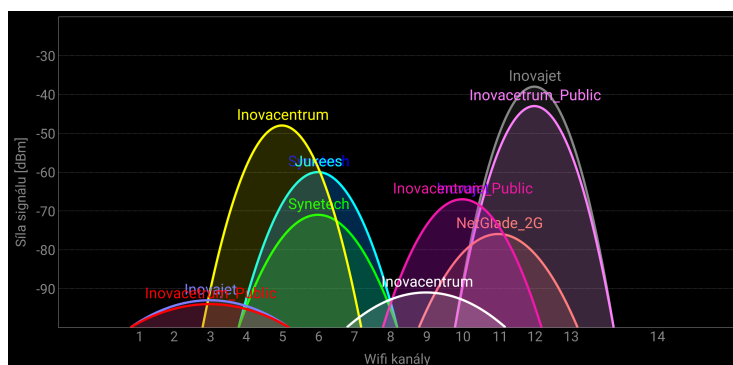
Obsazení komunikačních kanálů

Obrázky 7.7 a 7.8 zachycují bezdrátové sítě přítomné v testovacím. Obrázek 7.7 zachycuje kolísání síly signálu pro jednotlivé bezdrátové sítě, kdežto obrázek 7.8 zachycuje obsazení jednotlivých komunikačních kanálů ve stejném místě. Oba obrázky byly vytvořeny aplikací *Wifi Analyzer*¹ na testovacím zařízení *Galaxy S6*. Kolísání síly signálu má negativní vliv na lokalizaci uživatele v daném prostředí. I přesto, že metoda lokalizace pomocí bezdrátových majáků odstraňuje extrémy, dochází k nesprávnému odhadu polohy.

¹<https://play.google.com/store/apps/details?id=com.farproc.wifi.analyzer>



Obrázek 7.7: Změny síly signálů



Obrázek 7.8: Obsazení komunikačních kanálů

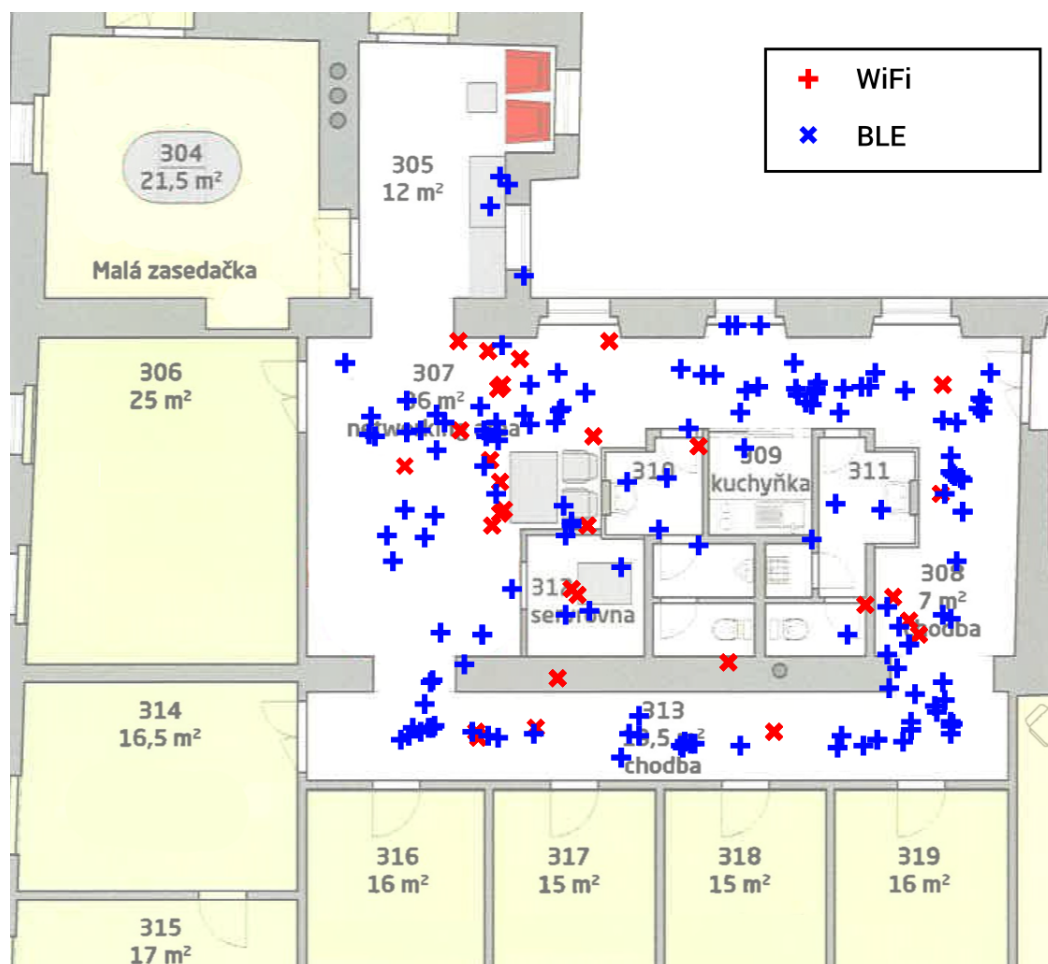
■ Způsob měření referenčních bodů

Referenční body pro *fingerprinting* byly naměřeny setrváním na konkrétním místě po dobu potřebnou k naměření dostatečného počtu vzorků (alespoň 5 vzorků pro *WiFi* sítě a 10 vzorků pro *beacon* zařízení). Doba měření jednoho referenčního bodu se pohybovala mezi 15 až 20 vteřinami. Měření jsem prováděl za plynulého otáčení kolem své osy, aby došlo k zachycení signálu ze všech stran. Při měření jsem zjistil, že pokud operační systém skenuje obě pásma (2.5GHz i 5Ghz), tak není možné získat měření *WiFi* signálu vícekrát než jednou za tři sekundy. Při zakázání skenování 5GHz sítí na LG Nexus 5 se interval mezi dvěma měřeními zkrátil ze tří vteřin na jednu. Bohužel na zařízení *Samsung S6* není možné vypnout skenování sítí z 5GHz pásma. Celkem bylo naměřeno 105 referenčních bodů.

■ 7.3.4 Porovnání metod *fingerprintingu*

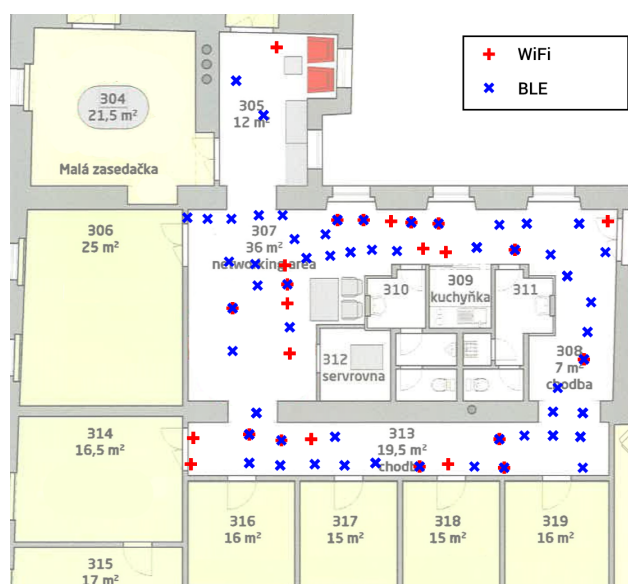
Obrázek 7.9 zachycuje polohy odhadnuté v průběhu testovací trasy z *WiFi* a *BLE* majáků pomocí metody KNN s $K=5$. Z obrázku je patrné, že četnost poloh odhadnutých z *BLE* majáků značně převyšuje počet poloh odhadnutých z *WiFi*

majáků. Tato četnost je ovlivněna intervalem mezi jednotlivými vzorky měření síly signálu. Interval mezi *WiFi* vzorky byl při skenování 2.5GHz a 5GHz sítí 3 vteřiny, kdežto pro *BLE* vzorky byl 100ms. Dále je z odhadnutých poloh patrné, že při průměrování výsledných souřadnic dochází k umístění polohy na nedosažitelná místa jako jsou překážky a zdi. Takto zaměřené polohy nejsou v případě počáteční lokalizace uživatele brány v potaz.

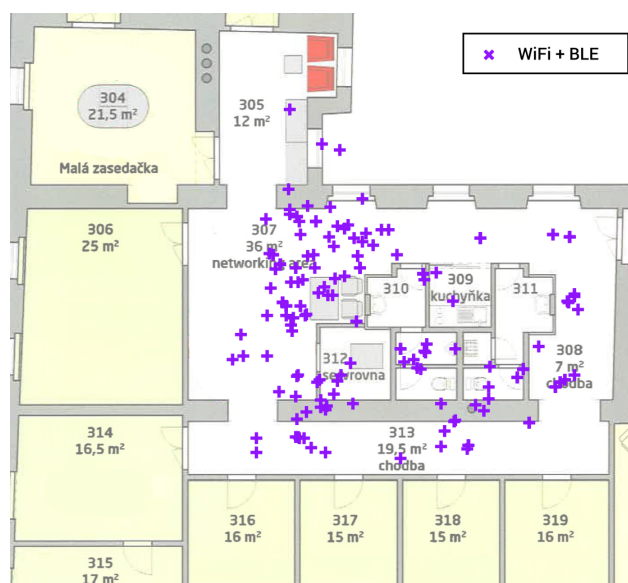


Obrázek 7.9: Lokalizace pomocí metody KNN pro $K=5$

Obrázek 7.10, podobně jako předchozí obrázek, zachycuje polohy z *WiFi* a *BLE* s tím rozdílem, že použitou metodou hledání nejbližšího referenčního bodu je metoda *NN*. Jelikož se souřadnice neprůměrují z několika referenčních bodů, ale používají se souřadnice konkrétního referenčního bodu, nedochází k zaměření polohy mimo zmapované prostory. Bohužel je tato metoda náchylnější na fluktuace signálu.

Obrázek 7.10: Lokalizace pomocí metody *NN*

Obrázek 7.11 zachycuje jednotlivé polohy získané kombinací *WiFi* a *BLE* majáků, kde výsledná poloha je odhadnuta z *K* referenčních bodů.

Obrázek 7.11: Lokalizace kombinací *WiFi* a *BLE* majáků

V průběhu vývoje a experimentálního testování výše zmíněných algoritmů je nejvhodnější použít metodu *KNN* pro *BLE* zařízení s validací, zda je nalezená souřadnice validní (na dosažitelném místě).

■ 7.3.5 Zdroje chyb

Mezi hlavní zdroje chyb ovlivňující přesnost lokalizace lze zařadit:

- nepřesný odhad délky kroku, který je způsoben individuálností stylu chůze každé osoby,
- nepřesný odhad azimutu, jež je ovlivněn magnetickými rušeními a *driftem* gyroskopu,
- nepřesně zkalibrované mapové podklady (azimut, měřítko grafických podkladů),
- fluktuace síly přijatého signálu, způsobená šířením signálu prostředím,
- nepřesné odhadnutí počáteční polohy.

■ 7.4 Testování implementace

Provést komplexní testování implementace aplikací je nad rámec rozsahu této práce. Do aplikací byl implementován *framework Fabric*² na sledování stability a detekci výjimek a pádů aplikace. Tudiž testování aplikací probíhá kontinuálně na všech uživateliích v reálném čase.

²<https://fabric.io/>

Kapitola 8

Závěr

Všechny stanovené cíle této práce byly splněny. V kapitole 2 byly podrobně prozkoumány možnosti chytrých mobilních telefonů pro navigaci uživatele ve vnitřním prostředí. Analýza obsahuje veškeré senzory mobilního zařízení, které lze použít pro lokalizaci a detekci změny polohy uživatele. Na základě této analýzy, současných řešení zmíněných v kapitole 3 a definici problému popsaneho v kapitole 4 byl navržen navigační proces, který je podrobně popsán v kapitole 5. Součástí návrhu navigačního procesu bylo vytvořit mapové podklady, které bude navigační aplikace používat ke zjištění podoby vnitřního prostředí. Aby bylo možné vytvořit mapové podklady, byla také navržena aplikace pro operační systém Android pro vytváření těchto detailních podkladů. Aplikace umožňuje nahrát grafický plán budovy, nastavit správné měřítko a orientaci. Dále aplikace vyžaduje zakreslení překážek ve formě polygonů a lomených čar. Pro účel navigace je nutné v této aplikaci umístit význačné body, do kterých je možné uživatele navigovat. Aplikace také poskytuje možnost vytvořit množinu referenčních bodů reprezentujících hodnoty signálů jednotlivých bezdrátových majáků. Jelikož použitelnost vestavěného krokoměru a kompasu byla nedostačující, navrhl a implementoval jsem algoritmus krokoměru a kompasu s využitím dostupných senzorů orientace a pohybu. Výsledná metoda navigace se skládá z odhadnutí počáteční polohy pomocí bezdrátových majáků, detekce změny polohy uživatele na základě krokoměru a kompasu. Dále metoda využívá částicový filtr ke korekci polohy uživatele dle fyzické struktury prostředí. Pro hledání nejkratší cesty mezi cílovou a aktuální polohou uživatele byl použit algoritmus A^* , který hledá cestu za využití grafu viditelnosti vytvořeného dle podoby prostředí. Tato navigační metoda tvoří základ navigační aplikace, jejíž návrh byl popsán také v kapitole 5. Veškeré navržené aplikace a navigační metoda byly implementovány v kapitole 6. Testování navigační metody je popsáno v kapitole 7. Testování v reálném prostředí potvrdilo použitelnost navržené metody a funkčnost implementace aplikace pro operační systém Android. Běžná metoda navigace pomocí *PDR* dosahovala průměrné přesnosti 289cm (<399cm po 90% času), kdežto přesnost navržené metody pro testované prostředí byla 81cm (<154cm po 90% času). Jelikož nebyla získána přesná referenční poloha, obě metody testování obsahují chybu zapříčiněnou nepřesným odhadem skutečné polohy. Navržená metoda tedy splnila požadavek na přesnost pro použití jako navigační metoda ve vnitřním prostředí.

■ 8.1 Práce do budoucna

- Pro zvýšení možnosti použití navržené metody navigace by bylo vhodné rozšířit navigační proces o autonomní navigaci mezi jednotlivými podlažními, aby mezi nimi uživatel nemusel ručně přepínat.
- Při testování navrženého řešení se zrodila myšlenka, jaký vliv na přesnost lokalizace by mělo umístění bezdrátových majáků na strop. Zda by tato orientace měla menší vliv na absorpci signálu lidským tělem v případě pohybu většího počtu osob v daném prostředí.
- Výrazným faktorem ovlivňujícím přesnost detekce kroků, je změna akcelerace zařízení při jeho ovládání, která vede k detekci falešných kroků. Vytvořená navigační aplikace neposkytuje prostor k jejímu ovládání v průběhu navigace prostředím, tudíž tento faktor implementovanou aplikaci ovlivňuje minimálně. Nabízí se tedy prostor pro implementaci metody, která nějakým způsobem rozliší tyto dva podobné průběhy změny akcelerace.
- Další možnou optimalizací je rozdělení mapových podkladů na menší části tak, aby se urychlil výpočet hledání cesty a propagace částic.
- Jelikož je navržená metoda postavena na *fingerprintingu*, je nutné vytvořit síť referenčních bodů, které budou obsahovat naměřená data z bezdrátových majáků. Počet potřebných referenčních bodů roste s plochou, kterou je nutné proměřit a vždy určit polohu referenčního bodu v mapových podkladech. Za účelem redukce množství měření referenčních bodů by bylo vhodné navrhnout metodu, která by umožňovala uměle dopočítat referenční body.
- Pro masové použití aplikace by bylo možné implementovat server, který by distribuoval mapové podklady mezi jednotlivými uživateli. Jelikož jsou mapové podklady uloženy ve formátu *JSON*, nabízí se možnost implementace *REST* serveru.



Literatura

- [1] Wikipedia, “Globalni družicovy polohovy system.” https://cs.wikipedia.org/wiki/Globální_družicový_polohový_systém, 2016.
- [2] C. S. N. Office, “Beidou navigation satellite system open service performance standard.” http://www2.unb.ca/gge/Resources/beidou_open_service_performance_standard_ver1.0.pdf, 2013.
- [3] GPS.gov, “Global positioning system (gps) civil positioning performance specification.” <http://www.gps.gov/technical/ps/2009-civil-monitoring-performance-specification.pdf>, 2009.
- [4] Bluetooth SIG, Inc., “Specification of the bluetooth system, v4.2.” https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439&_ga=1.233179166.1662416632.1456600186, 2014.
- [5] Apple, Inc., “Getting started with ibeacon.” <https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>, 2014.
- [6] Z. Chen, Q. Zhu, H. Jiang, and Y. C. Soh, “Indoor localization using smartphone sensors and ibeacons,” in *Industrial Electronics and Applications (ICIEA), 2015 IEEE 10th Conference on*, pp. 1723–1728, IEEE, 2015.
- [7] Google, Inc., “Beacons overview.” <https://developers.google.com/beacons/overview>, 2015.
- [8] I. Estimote, “Estimote beacons.” <http://estimote.com/>, 2016. Navštíveno: 20.ledna 2016.
- [9] IEEE, “Wireless lan medium access control (mac) and physical layer (phy) specifications.” <http://standards.ieee.org/getieee802/download/802.11-2012.pdf>, 2012.
- [10] Google, Inc., “Near field communication.” <http://developer.android.com/guide/topics/connectivity/nfc/index.html>, 2016.
- [11] Techopedia Inc., “What is accelerometer?.” <https://www.techopedia.com/definition/24430/accelerometer>, 2016.

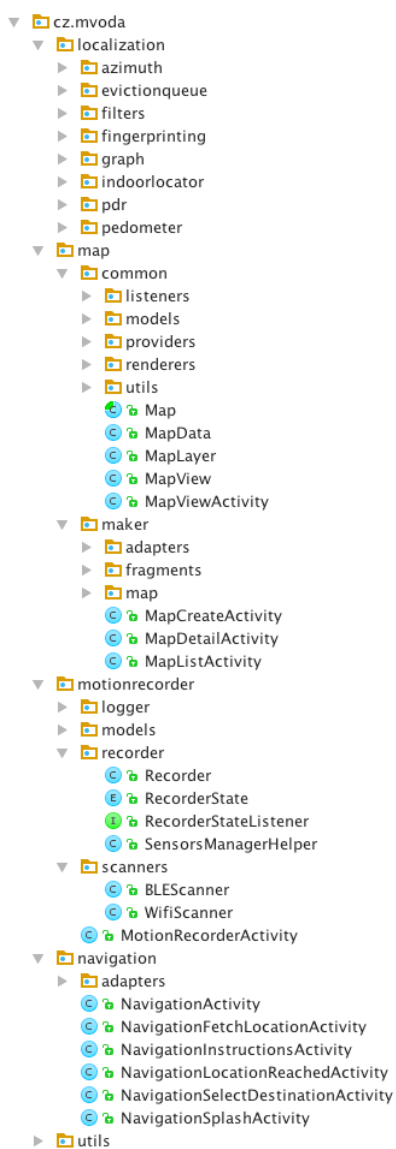
- [12] Google, Inc., "Using the gyroscope." http://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-gyro, 2016.
- [13] Wikipedia, "Mems magnetic field sensor." https://en.wikipedia.org/wiki/MEMS_magnetic_field_sensor, 2016.
- [14] R. Faragher and R. Harle, "Location fingerprinting with bluetooth low energy beacons," *IEEE Journal on Selected Areas in Communications*, vol. 33, pp. 2418–2428, Nov 2015.
- [15] U. Shala and A. Rodriguez, "Indoor positioning using sensor-fusion in android devices," 2011.
- [16] O. Woodman and R. Harle, "Pedestrian localisation for indoor environments," in *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 114–123, ACM, 2008.
- [17] H.-H. Hsu, W.-J. Peng, T. K. Shih, T.-W. Pai, and K. L. Man, "Smartphone indoor localization with accelerometer and gyroscope," in *Network-Based Information Systems (NBIS), 2014 17th International Conference on*, pp. 465–469, IEEE, 2014.
- [18] D. Pai, I. Sasi, P. S. Mantripragada, M. Malpani, and N. Aggarwal, "Padati: A robust pedestrian dead reckoning system on smartphones," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 2000–2007, June 2012.
- [19] J. Straub, "Pedestrian indoor localization and tracking using a particle filter combined with a learning accessibility map," *Bachelor Thesis*, 2010.
- [20] S. P. Subramanian, J. Sommer, F. P. Zeh, S. Schmitt, and W. Rosenstiel, "Pbil pdr for scalable bluetooth indoor localization," in *2009 Third International Conference on Next Generation Mobile Applications, Services and Technologies*, pp. 170–175, Sept 2009.
- [21] V. Kumar, R. Vats, S. Goyal, S. Kumar, and P. Pathak, "Interaction of electromagnetic radiation with human body," *Indian Journal of Radio & Space Physics*, vol. 37, pp. 131–134, 2008.
- [22] I. Google, "Android design guidelines." <http://developer.android.com/design/>, 2016. Zobrazeno:8.4.2016.
- [23] C. Yang and H. r. Shao, "Wifi-based indoor positioning," *IEEE Communications Magazine*, vol. 53, pp. 150–157, March 2015.
- [24] S. Boonsriwai and A. Apavatjrut, "Indoor wifi localization on mobile devices," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2013 10th International Conference on*, pp. 1–5, May 2013.

- [25] B. Shin, J. H. Lee, T. Lee, and H. S. Kim, "Enhanced weighted k-nearest neighbor algorithm for indoor wi-fi positioning systems," in *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, vol. 2, pp. 574–577, April 2012.
- [26] A. Farshad, J. Li, M. K. Marina, and F. J. Garcia, "A microscopic look at wifi fingerprinting for indoor mobile phone localization in diverse environments," in *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pp. 1–10, Oct 2013.
- [27] F. Lemic, "Benchmarking of quantile based indoor fingerprinting algorithm," *Telecommunication Networks Group, Technische Universität Berlin, Tech. Rep. TKN-14-001*, 2014.
- [28] S. He and S.-H. G. Chan, "Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons," 2015.
- [29] A. Zanella, "Best practice in rss measurements and ranging," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2016.
- [30] A. T. Parameswaran, M. I. Husain, S. Upadhyaya, *et al.*, "Is rssi a reliable parameter in sensor localization algorithms: An experimental study," in *Field Failure Data Analysis Workshop (F2DA09)*, p. 5, 2009.
- [31] T. Akiyama, H. Ohashi, A. Sato, G. Nakahara, and K. Yamasaki, "Pedestrian dead reckoning using adaptive particle filter to human moving mode," in *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pp. 1–7, Oct 2013.
- [32] X. Yunqiang, Z. Yanshun, W. Z. Qing, and L. Ming, "Research for pedestrian navigation positioning method based on mems sensors," in *Control Conference (CCC), 2015 34th Chinese*, pp. 5315–5318, July 2015.
- [33] L. Berbakov, B. Pavković, and S. Vraneš, "Smart-phone application for autonomous indoor positioning," in *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, pp. 670–674, May 2015.
- [34] Y. Sun, Y. Zhao, and J. Schiller, "An indoor positioning system based on inertial sensors in smartphone," in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 2221–2226, March 2015.
- [35] Y. S. Suh and B. Amarbayasgalan, "Indoor magnetic signature based localization algorithm without person-dependent parameter calibration," *Sensors*, vol. 14, no. 8, pp. 14375–14398, 2014.
- [36] H.-h. Lee, S. Choi, and M.-j. Lee, "Step detection robust against the dynamics of smartphones," *Sensors*, vol. 15, no. 10, pp. 27230–27250, 2015.
- [37] B. Zhou, Q. Li, Q. Mao, W. Tu, and X. Zhang, "Activity sequence-based indoor pedestrian localization using smartphones," *IEEE Transactions on Human-Machine Systems*, vol. 45, pp. 562–574, Oct 2015.

- [38] M.-S. Pan and H.-W. Lin, “A step counting algorithm for smartphone users: Design and implementation,” *Sensors Journal, IEEE*, vol. 15, no. 4, pp. 2296–2305, 2015.
- [39] R. B. Dan Murray, “Adaptiv: An adaptive jerk pace buffer step detection algorithm.” <https://github.com/danielmurray/adaptiv>, 2015. Navštíveno: 17.dubna 2016.
- [40] H. Bao and W. C. Wong, “An indoor dead-reckoning algorithm with map matching,” in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 1534–1539, July 2013.
- [41] P. Lawitzki, “Application of dynamic binaural signals in acoustic games,” *Hochschule der Medien Stuttgart*, 2012.
- [42] L. Fang, P. J. Antsaklis, L. A. Montestruque, M. B. McMickell, M. Lemmon, Y. Sun, H. Fang, I. Koutroulis, M. Haenggi, M. Xie, and X. Xie, “Design of a wireless assisted pedestrian dead reckoning system - the navmote experience,” *IEEE Transactions on Instrumentation and Measurement*, vol. 54, pp. 2342–2358, Dec 2005.
- [43] Google Inc., “Motion sensors: Using the gyroscopes.” https://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-gyro, 2016. Navštíveno: 5.března 2016.
- [44] Google Inc., “Motion sensors: Using the gyroscopes.” <http://christine-coenen.de/blog/2014/07/02/smooth-compass-needle-in-android-or-any-angle-with-low-pass-filter/>, June 2014. Navštíveno: 4.dubna 2016.
- [45] A. Doucet, N. De Freitas, and N. Gordon, “An introduction to sequential monte carlo methods,” in *Sequential Monte Carlo methods in practice*, pp. 3–14, Springer, 2001.
- [46] A. Patel, “Map representations.” <http://theory.stanford.edu/~amitp/GameProgramming/MapRepresentations.html>, 2009. Navštíveno: 13.dubna 2016.
- [47] IDC, “Smartphone os market share.” <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>, 2016.
- [48] Google, Inc., “Android 6.0 apis.” <http://developer.android.com/about/versions/marshmallow/android-6.0.html>, 2016.
- [49] A. Agrawal, “Android application security part 2-understanding android operating system.” <https://manifestsecurity.com/android-application-security-part-2/>, Jan 2015. Navštíveno: 18.dubna 2016.

Příloha A

Struktura projektu



Obrázek A.1: Část struktury projektu

Příloha B

Seznam senzorů testovacího zařízení

Senzor	Výrobce	Max.hodn.	Přesnost	P.(ms)
Acceleration	Invensense MPU6500	39,2266	0,001197101	5
Gyroscope	Invensense MPU6500	34,906586	0,001065265	5
Gyroscope UnCalibrated	Invensense MPU6500	34,906586	0,001065265	5
Magnetic	Yamaha YAS537	1200	0,1	10
Uncalibrated Magnetic	Yamaha YAS537	1200	0,1	10
Linear Acceleration	Samsung Electr.	19,6133	5,96E-08	10
Rotation Vector	Invensense MPL	1	5,96E-08	10
Game Rotation Vector	Invensense MPL	1	5,96E-08	10
Orientation	Samsung Electr.	360	0,00390625	10
Gravity	Samsung Electr.	19,6133	5,96E-08	10
Step Detector	Samsung Inc.	1	1	N/A
Step Counter	Samsung Inc.	4,29E+09	1	N/A
Significant Motion	Samsung Inc.	1	1	N/A
Tilt Detector	Samsung Inc.	1	1	N/A
Pick Up Gesture	Samsung Inc.	1	1	N/A
Barometer	STM LPS25H	1260	1	200
Proximity	AMS TMD4903	8	8	N/A
RGB	AMS TMD4903	60000	1	200

Nestandardní senzory (rozšíření výrobce):					
Interrupt	Gy-	Invensense	34,906586	0,001065265	N/A
roscope					
RGB IR		AMS TMD4093	60000	1	200
Grip		SEMTECH	5	5	N/A
Screen	Orien-	Samsung Electr.	255	255	N/A
tation					
HRM		Samsung Electr.	200	1	N/A

Tabulka B.1: Seznam senzorů testovacího zařízení



Příloha C

JSON struktura mapových podkladů

name	Název plánu podlaží	String
floor	Číslo podlaží	Int
image	Rastrová grafika	Object<Image>
pois	Pole bodů zájmu	Array<POI>
obstacles	Pole objektů odpovídajících překážkám (neviditelné pro uživatele)	Array<Obstacle>
measurements	Vyhrazená struktura pro dodatečné informace	Object
<i>Object(Image)</i>		
binary	Soubor formátu JPEG nebo PNG kódovaný v BASE64	String
offset	Posunutí (bod F) grafického plánu vůči souřadnému systému	Object<Point>
rotation	Úhel rotace (α) středu grafického plánu ve směru hodinových ručiček (ve stupních)	Float
scale	Měřítko grafického plánu. Udává kolik pixelů v grafickém podkladu odpovídá kolika centimetrům ve skutečnosti	Double
<i>Object<POI></i>		
name	Název bodu zájmu	String
type	Typ bodu zájmu (room, lift, entrance, wc)	String
coordinates	Souřadnice bodu zájmu vůči souřadnému systému	Object<Point>
<i>Object<Obstacle></i>		
type	Typ překážky (<i>polygon</i> , <i>line</i>)	String
points	Pole bodů definující překážku	Array<Point>
<i>Object(Point)</i>		
x	souřadnice na ose x v jednotkách souř.systému	Double
y	souřadnice na ose y v jednotkách souř.systému	Double

Tabulka C.1: JSON struktura mapových podkladů



Příloha D

Obsah CD

thesis_vodamiro.pdf PDF verze této práce

source.zip Zdrojové kódy

README.txt Soubor s informacemi