VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTERAKTIVNÍ ROZHRANÍ PRO VZDÁLENÉHO ROBOTA PRO ANDROID

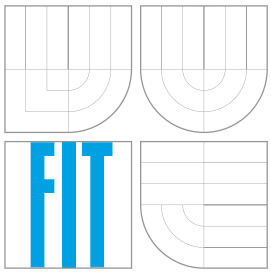
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LEOPOLD PODMOLÍK

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTERAKTIVNÍ ROZHRANÍ PRO VZDÁLENÉHO ROBOTA PRO ANDROID

INTERACTIVE INTERFACE FOR ROBOT REMOTE CONTROL FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LEOPOLD PODMOLÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL ŠPANĚL, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Bakalářská práce se zabývá problematikou vzdáleného ovládání robota. Cílem je navrhnout a naimplementovat uživatelské rozhraní na zařízení s dotykovou obrazovkou a operačním systémem Android. Uživatelské rozhraní je primárně vyvíjeno pro fakultního robota TB2, který využívá systém ROS Electric. Navržené uživatelské rozhraní se skládá z příchozího videa ze zařízení Microsoft Kinect, které je upevněno na robotovi a aktuální 2D mapy okolního prostředí. Robot je ovládán pomocí rozhraní, které je podobné aplikaci Street View od společnosti Google s tím rozdílem, že rotace na místě je řešena pomocí gest na obrazovce.

Abstract

This work describes an application to control a robot running the Robotic Operation System (ROS) remotely using Android-based touch devices. The application was primarily designed for our experimental faculty robot - TB2. Designed user interface shows a video stream coming from the Kinect mounted on the robot and the current 2D map. The robot movement is controlled by a Street View like user interface while rotation of the robot in place is solved by specific touch gestures on display.

Klíčová slova

Android, ROS, ROSJava, Robot, uživatelské rozhraní, vzdálené ovládání robota

Keywords

Android, ROS, ROSJava, Robot, UI, robot remote control

Citace

Leopold Podmolík: Interaktivní rozhraní pro vzdáleného robota pro Android, bakalářská práce, Brno, FIT VUT v Brně, 2013

Interaktivní rozhraní pro vzdáleného robota pro Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Španěla Ph.D.

.....
Leopold Podmolík
10. května 2013

Poděkování

Rád bych poděkoval Ing. Michalu Španělovi, Ph.D. za odborné vedení a konzultace. Dále bych rád poděkoval rodině a přítelkyni za psychickou podporu.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V době rozmachu mobilních zařízení, a to jak smartphonů, tak tabletů, se nabízí možnost pracovat s roboty právě pomocí těchto moderních zařízení. V současnosti již existují tzv. *low-cost* roboti a operační systémy pro takovéto roboty, například ROS [1]. Před dvěma lety přišla skupina lidí ze společnosti Google s rozšiřujícím balíčkem ROSJava [8] pro ROS a tím umožnila vývoj aplikací na systém Android pracující s robotem, který využívá systém ROS.

Cílem této práce je navrhnout uživatelské rozhraní (dále UI) pro intuitivní a snadné ovládání robota prostřednictvím dotykové obrazovky na platformě Android. Dílčím cílem je využít autonomních prvků robota jako je plánování trasy do zadaného bodu a celé rozhraní tak zjednodušit. Aplikace obsahuje prvky jako je video z robota, mapu, diagnostické informace, nastavení a ovládání *street-view*/*G-sensor*.

První kapitola diskutuje problematiku vzdáleného robota, kde je představena historie robotů a jejich UI a následně se seznámíme s dnešními trendy v robotice a moderními UI. V druhé kapitole se seznámíme s ROSem a ROSJavou. Třetí kapitola se již zabývá návrhem aplikace a jeho detailním popisem. V následující kapitole je rozebrána implementace. V předposlední kapitole je popsáno testování a to jak na simulaci robota, tak na reálném robotovi. Dále pak možný budoucí vývoj aplikace spolu s ukázky finálního UI. Závěrečná kapitola hodnotí a shrnuje celou práci.

Kapitola 2

Problematika vzdáleného ovládání robotů

V této kapitole se budeme zabývat historií robotů a dnešními trendy v robotice.

2.1 Historie

Pokud pomineme pokusy ze středověku - například stroje Leonarda DaVinciho, pak se historie robotů datuje od počátku dvacátého století. Je vhodné připomenout, že autorem pojmu (slova) robot je český spisovatel Karel Čapek, který ho jako první použil ve své hře R.U.R. a to již v roce 1921. Další důležitou postavou v historii robotiky je Issac Asimov, který v roce 1942 definoval tři základní zákony robotiky.

O prvních robotech, kteří připomínají dnešní roboty, můžeme mluvit až od roku 1950. Největšími průkopníky v robotice byly americké univerzity, které nejprve vytvořily výzkumné roboty. Postupem času se jejich aktivita zaměřila více na průmyslové roboty. Ovšem stále mluvíme o robotech, kteří byli umístěni na jedno místo a veškeré činnosti a pohyby byly vedeny většinou z pevně ukotveného místa, např. robotické paže apod. Nás ale budou zajímat mobilní roboti, tj. ty stroje, které jsou schopné se sami přemístit. Pokud budeme brát do úvahy historii v rozsahu posledních 20 let a chtěli bychom roboty rozdělit, pak bychom dostali tyto skupiny

- výzkumné,
- vojenské,
- civilní.

2.1.1 Výzkumné

Mezi nejznámějšími institucemi, které se zabývají vývojem robotů, patří NASA. Tato instituce vyvinula mnoho robotů, které vyslala do vesmíru. Nejznámějším robotickým vozítkem je MARS Rover Curiosity (obr. 2.1a).

2.1.2 Vojenské

Význam vojenských robotů většinou spočíval v roli průzkumníka terénu tam, kde hrozilo nebezpečí. Další využití robotů v armádě je při deaktivaci výbušnin, kdy roboti slouží jako

prodloužené ruce pyrotechniků. Posledním zásadnějším využitím vojenských robotů jsou záchranářské role. V diplomové práci [3] ze Švédska je pojednávano o využití amerického vojenského robota PackBot (obr. 2.1b) od USAR *(Urban Search and Rescue)* při záchranných pracech. Tento robot byl poprvé nasazen v roce 2001 při útoku na světové obchodní centrum, kde pomohl při záchraně lidských životů.

(a) Výzkumný robot - Mars Rover Curiosity.
Zdroj: coolsciencenews.blogspot.com

(b) Záchranářský robot - PackBot.
Zdroj: es.engadget.com

Obrázek 2.1: Výzkumný a záchraný robot.

2.1.3 Civilní

Do skupiny civilních robotů bezesporu zařadíme i roboty vytvořené za účelem zábavy. Mezi typické zástupce takovýchto robotů patří roboti "fotbalisti". S těmito roboty se mnoho let pořádají soutěže v tzv. "robofotbalu". Z počátku vypadali jako jezdící krabice (obr. 2.2a), ale postupem času se více přizpůsobovali podobě skutečných fotbalistů. Tyto roboty můžeme vidět na obrázku 2.2b.

(a) První verze robotů - fotbalistů.
Zdroj: msc.berkeley.edu

(b) Novější roboti - fotbalisti - Aldebaran Nao.
Zdroj: www.parismatch.com

Obrázek 2.2: Civilní roboti - fotbalisti

Do skupiny civilních robotů můžeme zařadit i hračky od společnosti LEGO, například jejich produkt Lego Mindstorms NXT. Tato hračka obsahuje základní programovatelný modul, který se propojuje s ostatními díly lega do konkrétního objektu, např. robota nebo robotické paže apod. Cílovým uživatelem těchto robotů jsou děti. Programování těchto robotů probíhá na velmi abstraktní úrovni přizpůsobené schopnostem dětí. Ukázku robota z lega vidíte na obrázku 2.3a.

2.2 Dnešní trendy

Jak již bylo uvedeno v předcházející části, novým trendem je, aby roboti vypadali jako lidé tzv. humanoidi. V dnešní době je největší rozmach uvedených typů robotů především v Japonsku, kde výzkumné skupiny vykazují velmi zajímavé výsledky (viz obr. 2.3b). Zde už nemluvíme o nějakém UI, ale o autonomních systémech, které pracují samostatně a jsou buď ovládany hlasem, či následují pohyb člověka. Pokud mluvíme o Hi-Tech robotech, pak bychom neměli zapomnenou na americké vojenské stroje, jako jsou například bezpilotní letouny, které můžeme také označit za roboty.

(a) Civilní robot z lega.
Zdroj: gadget.blog.hu

(b) Robot - humanoid - HRP-4C.
Zdroj: http://www.cbc.ca

Obrázek 2.3: Civilní roboti

2.2.1 "Low-cost" roboti

Pokud se oprostíme od Hi-Tech robotů a zaměříme se na nižší úroveň robotiky, pak můžeme v dnešní době mluvit o rozmachu tzv. "low-cost" robotů. Tito roboti se dají levně koupit či sestrojit a dovolují tak větší skupině lidí s nimi experimentovat. Aby byl tento vývoj možný je zapotřebí ještě rozvoje dalšího důležitého aspektu, a to volně dostupného operačního systému pro roboty. A právě takový je například ROS (viz kapitola 3). Tito "low-cost" roboti sice nevypadají jako lidé, což není jejich účelem, protože slouží pro účely experimentů a výzkumu nových technologií, kde není prioritou velký nárok na HW vybavení.

Turtlebot

Robot Turtlebot je typickým zástupcem "low-cost" robotů. Jeho základní části tvoří iRobot Roomba, notebook s ROS (více v kap. 3) a Microsoft Kinect. Tento robot je výhradně určen pro výzkumné a experimentální účely a můžeme ho vidět na obrázku 2.4a.

2.2.2 Experimentální roboti

Roboti PR2 *(Personal Robot 2)* a Care-O-bot již nepatří do kategorie "low-cost" robotů, ale stále slouží pro experimentování a výzkum. Tyto roboty si nemůže vytvořit kdokoliv, ale jsou vyvíjeny specializovanými společnostmi. Více informací najdeme v popisu jednotlivých robotů.

(a) Turtlebot ver.1.
Zdroj: ros.org/wiki/Robots/TurtleBot

(b) Care-O-Bot.
Zdroj: ftp.isr.ist.utl.pt

(c) Personal Robot 2.
Zdroj: robotliving.com

Obrázek 2.4: Roboti, kteří používají ROS.

Care-O-bot

Robot Care-O-Bot byl vyvinut německou firmou Fraunhofer IPA. Tento robot je využíván jak pro komerční, tak i výzkumné účely. Velkou výhodou tohoto robota je jeho robotická paže, kterou můžeme vidět spolu s celým robotem na obrázku 2.4b.

PR2

Robota PR2 vyvinula firma Willow Garage a je vyvíjen jednak za účelem experimentování, ale současně si klade za cíl být použitelným pro veřejnost, aby mohl poskytovat pomoc jak v práci, tak i v domácnosti. Robota můžeme vidět na obrázku 2.4c.

2.2.3 Domácí využití robotů

V poslední době se roboti dostali i do našich domácností, například v podobě tzv. inteligetních vysavačů. U těchto robotů také nemůžeme mluvit o nějakém UI, protože pracují většinou autonomně a to pomocí sensorů. V těchto robotech se většinou nachází operační systémy, pomocí kterých si robot postupně vytváří mapu místnosti na jejímž základě si pak dokáže sám rozpoznávat překážky a řešit jejich objíždění. Tyto inteligentní vysavače jsou součástí "low-cost" robotů, jako jejich základní stavební prvek. Ukázku inteligentního vysavače můžeme vidět na obrázku 2.5.

Obrázek 2.5: Robot - Roomba.
Zdroj: http://www.ok-produkt.cz

2.3 Uživatelské rozhraní robotů

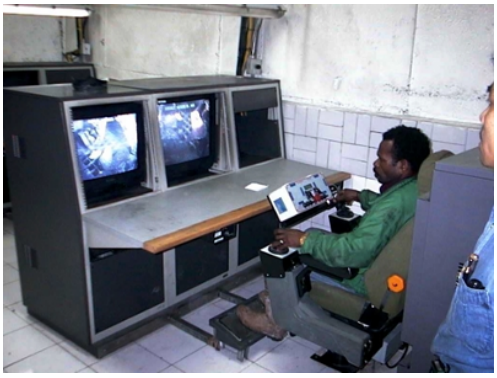
V této části textu budou rozebrány UI robota jak z historického hlediska, tak z pohledu moderních rozhraní.

2.3.1 Historie

První UI vypadaly jako malé místnosti, kde bylo mnoho monitorů a uživatel seděl v křesle a ovládal vzdáleného robota pomocí několika joysticků. Tento typ ovládání můžeme vidět na obrázku 2.6a, který je z článku [6] pojednávajícím o těžebních strojích v Austrálii. Na obrázku 2.6b je jasně vidět jak dříve bylo vše spojeno pomocí drátů a bylo problematické zobrazovat informace v lehce čitelné formě. Pokud bychom shrnuli poznatky z výše uvedených informacích o robotech, pak typickými prvky, které se vyskytují v UI pro roboty, jsou

- ovládání - HW joystick,
- obrazovky - typicky více monitorů či televizí,
- připojení - pomocí drátů,
- informace - většinou nepřehledné, snaha o co největší počet informací.

Postupem času se některá UI vyvíjela tak, že ovládání připomínalo ovládání herních konzolí, ovšem stále bylo zapotřebí drátového spojení. Nevýhodou těchto UI byla nízká nebo žádná mobilita a to díky jejich rozměrům a nutnosti drátové spojení.



(a) Ovládání težebního robota.

(b) Grafické UI.

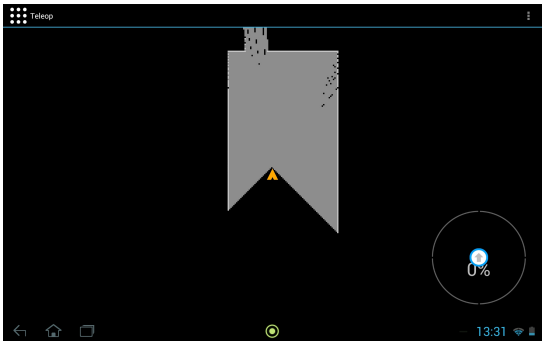
Obrázek 2.6: Težební robot z Austrálie.
Zdroj: [6]

2.3.2 Současnost

V současnosti, kdy jde vývoj mobilních zařízení rychle kupředu a to především v oblasti smartphonů a tabletů, se objevuje možnost pracovat s roboty právě pomocí těchto moderních zařízení. Díky těmto zařízení jsme mohli vyřešit hned první problém a to konektivitu. Protože jsou tyto přístroje vybaveny wi-fi a bluetooth adaptéry, můžeme využít těchto prostředků ke spojení s robotem a nemusíme tak řešit problém s drátovým připojením. Prostřednictvím těchto adaptérů se nám zvětšuje dosah pro ovládání robota. První typy ovládání na mobilních zařízeních byly podobné starším typům, především v podobě joysticku, ovšem ve virtuální podobně. Takový typ ovládání řadíme spíše mezi tradiční ovládání vzdálených robotů, které můžeme vidět na obrázku 2.7a z framework ROSJava (více v 3.3). Za zmínku stojí aplikace *Rviz on Android* [9], která neovládá robota, ale slouží jako vizualizační nástroj (více v 3.3.3). Další podobná aplikace, která robota ovládá pomocí virtuálního joysticku, byla vytvořena na Finské univerzitě [2] pro systém Maemo.

Dalším typem ovládání, který ovšem již patří mezi netradiční, je využití G-sensoru. Stejně jako výše zmíněné wifi a bluetooth adaptéry, tak i G-sensor obsahuje většina moderních zařízení. Tento způsob ovládání spočívá v naklánění zařízení do směru, kam chceme, aby se robot pohyboval. Ukázku takového UI můžeme vidět na obrázku 2.7b, ovšem v tomto UI je využito G-sensoru pouze pro pohyb vpřed a vzad, pro otáčení na místě jsou zde použita tlačítka. Pokud bychom shrnuli poznatky z výše uvedených aplikací, pak typickými prvky v UI pro roboty jsou

- on-line video stream,
- mapa okolí,
- laser-scan,
- ovládání - virtuální joystick.



(a) GUI s virtuálním joystickem.
Zdroj: [8]

(b) Ovládání k robotovi youBot.
Zdroj: http://mobile.xitaso.com

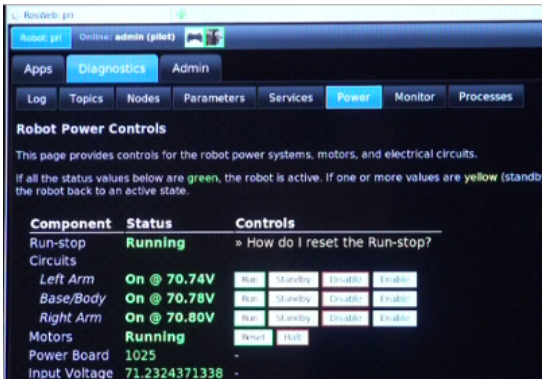
Obrázek 2.7: Moderní UI - mobilní aplikace

Posledním zajímavým typem je ovládání prostřednictvím webového prohlížeče. Protože většinou roboti obsahují wifi připojení, je možno se s nimi spojit právě pomocí webového prohlížeče. Jedním z takových UI je WebUI z ROSu, které můžeme vidět na obrázku 2.8b. Ovšem toto UI slouží pro diagnostiku robota a spouštění dalších aplikací na robotovi, nikoliv pro pohyb.

2.3.3 Experimentální uživatelské rozhraní robotů

Všechny doposud popsané typy ovládání využívaly moderní zařízení za pomoci interakce. V současnosti se již vyskytují i experimentální ovládání pomocí hlasu. Tento typ ovládání využívá hlasových pokynů k pohybu robota, ale toto řešení má zatím nedostatky, protože je těžko ovladatelné pokud je použito například v rušném okolí, čímž se efektivita ovládání výrazně snižuje. Dalším, spíše experimentálním, ovládáním je využití pohybu těla prostřednictvím kamery. Toto ovládání spočívá ve snímání lidského těla a následného rozpoznání pohybu například rukou a poté pohybem robota. Právě díky 3D kamerám, jako je například Microsoft Kinect (více v 4.1), byl umožněn takový způsob ovládání. Na obrázku 2.8a můžeme vidět jak se ovládá letající AR Drone pomocí pohybu rukou.

(a) Ovládání robota pohybem rukou.
Zdroj: http://booktype-demo.sourcefabric.org



(b) WebUI.
Zdroj: [1]

Obrázek 2.8: Moderní UI - jiné

Kapitola 3

ROS

ROS*(Robot Operating System)* [1] je open-source operační systém. ROS je podobný klasickému OS, takže nám vytváří abstraktní práci s HW, low-level kontrolu zařízení a práci s daty mezi procesy. Obsahuje i knihovny a podpůrné prostředky pro vytváření nových částí systému.

3.1 Historie

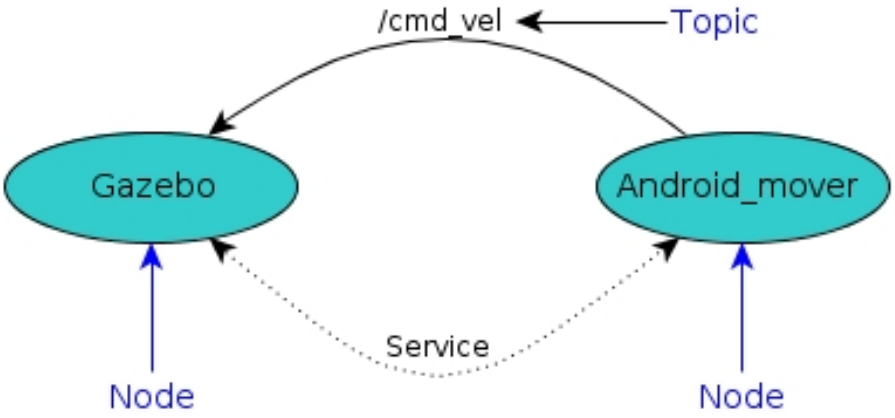
Za vznikem a vývojem ROSu stojí firma Willow Garage, která vznikla v roce 2006 za účelem vývoje robotů pro nevojenské použití. Nejvýraznějším robotem, který vznikl v této firmě, je PR2*(Personal Robot 2)* popsán v kapitole 2.2. Tvůrci ROSu nedávno založili tradici pořádání každoročního střetnutí vyvojářů zabývajících se ROSem tzv. *ROScon*, kde se kromě debat a seminářů představují i novinky.

Verze ROSu dostávají názvy stejně tak, jako verze u klasických OS. Poslední verze se jmenují Electric, Fuerte a nejnovější Groovy.

3.2 Koncept a použití

Základním programem ROSu je *roscore*, na který se připojují další programy. ROS funguje na principu peer-to-peer komunikace, kdy další programy potřebují znát lokaci *roscore*.

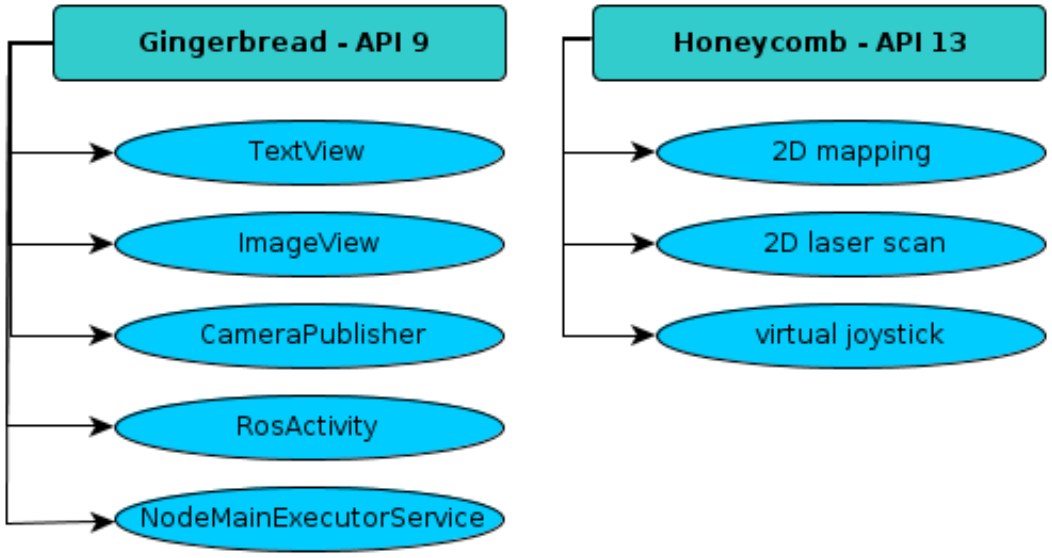
Koncept ROSu se rozděluje na tři úrovně a to Filesystem level, Computation Graph level, a Community level. Filesystem level obsahuje *Packages*, což jsou zdrojové kódy podprogramů, *Stacks*, což jsou kolekce *Packages*. V Computation Graph level jsou *Nodes*, *Topic* a *Services* (obr. 3.1), kde *Node* představuje spuštění *Package*, *Topic* představuje prostor, kde si jednotlivé *Node* ukládají (*Publisher*) nebo čtou (*Subscriber*) data a nakonec *Services* umožnuje komunikaci typu dotaz-odpověď. Poslední částí Community level se rozumí prostor pro výměnu software v komunitě a to v rámci repozitářů nebo Wiki stránek.



Obrázek 3.1: Jednoduché schéma ROSu.

3.3 ROSJava a Android

ROSJava [8] je implementace ROSu v Jave, která umožňuje jednoduše pracovat se všemi prvky ROSu. ROSJava obsahuje dva základní balíčky a to *rosjava_core* a *android_core*. První zmiňovaný balíček obsahuje základní implementaci a druhý balíček obsahuje základní prvky pro vývoj na systému Android. Primárním účelem ROSJavy je umožnit intreakci mezi ROSem a systémem Andorid.



Obrázek 3.2: Přehled prvků z balíčku *android_core*.

3.3.1 Historie

ROSJavu vytvořil v roce 2011 tým lidí ze společnosti Google pod vedením Damona Kohlera. I když se zdá tento framework velmi mladý, již stihl projít velkými změnami, jako je na-

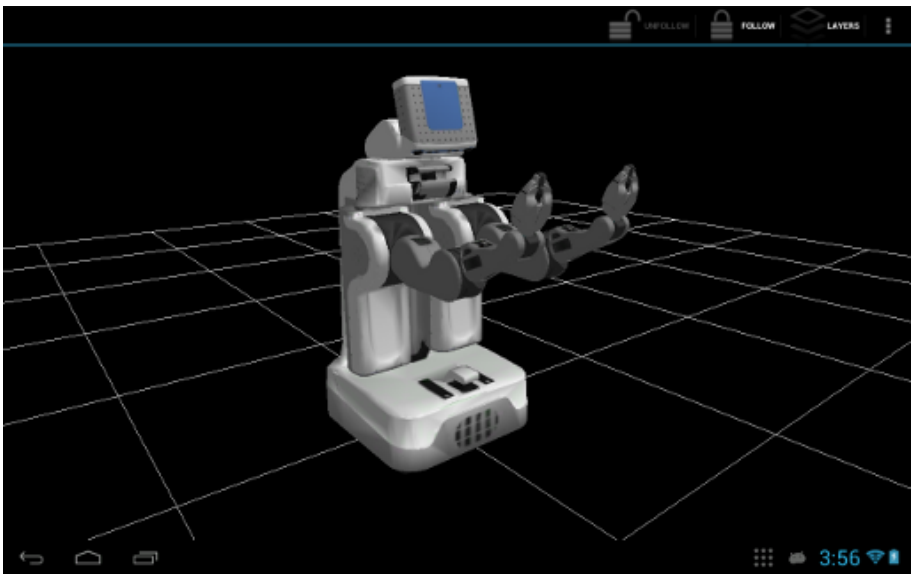
příklad změna systému na překlad zdrojových kódů. Obecně lze říci, že vývoj ROSJavy je velmi dynamický a ani oficiální dokumentace neobsahuje plně funkční příklady zdrojových kódů, což stěžuje tvorbu aplikací založených na tomto frameworku.

3.3.2 Použití

Balíček *android_core*, který obsahuje základní prvky pro Android, se rozděluje na dva menší balíčky. První menší balíček obsahuje prvky pro verze Androidu *Gingebread* (API 9). Tento balíček obsahuje základní stavební prvky `RosActivity` a `NodeMainExecutorService`. Dále pak tento balíček obsahuje grafické prvky jako `RosImageView`, který slouží pro zobrazování obrazových dat z robota a to, jak v "surové", tak i v komprimované podobě, a `RosCameraView`, který umožnuje posílání obrazu z kamery umístěné na zařízení se systémem Android. Druhý menší balíček je určen pro verze Androidu *Honeycomb* (API 13). Tento balíček rozšiřuje první balíček pouze o grafické prvky jako jsou virtuální joystick, zobrazování 2D mapy a vizualizaci laser-scanu.

3.3.3 Vzdálené ovládání robota

V současné době již existuje několik aplikací založených na základě ROSJavy. Za zmínku stojí aplikace *Rviz on Android* (obr. 3.3), která slouží jako vizualizační nástroj. Tuto aplikaci [9] vytvořil A. Zimmerman, americký student, přímo ve firmě Willow Garage. Zajímavostí této aplikace je fakt, že všechny grafické prvky byly přeprogramovány z druhého podbalíčku ROSJavy, protože tyto prvky v některých případech nefungují správně, nebo jen na některých zařízeních.



Obrázek 3.3: Ukázka aplikace *Rviz for Android*.

3.4 Simulační nástroj Gazebo

I když v dnešní době je snadnější vlastnit robota, je stále důležité používat simulační nástroje, které nám pomocí simulace nahrazují fyzické roboty. Obecně pro simulaci robotů existuje mnoho nástrojů jako jsou Microsoft Developer Studio, Webots, MORSE a Gazebo. Právě posledně jmenovaný nástroj Gazebo je pravděpodobně nejlepším nástrojem pro práci s ROSem, protože je plně integrován do toho systému. Tento simulační nástroj nám kromě zobrazování robota, umožnuje vytvářet rozmanité světy na základě velké databáze objektů.

Dalším nástrojem, o kterém bych zde rád zmínil, je RViz, což je vizualizační nástroj pro ROS, který zobrazuje jak data z robota, tak i samotného robota. Mezi zobrazovaná data patří například video z kamery robota, mapa a laser-scan.

Kapitola 4

Experimentální robot TB2

Robot TB2 (obr. 4.1) byl vytvořen na FIT VUT v Brně. Tento robot je vyvíjen pracovní skupinou Robo@FIT a jeho autorem je Ing. Materna. Tento robot slouží k testování nových alogritmů, například pro mapování či manipulaci s předměty.

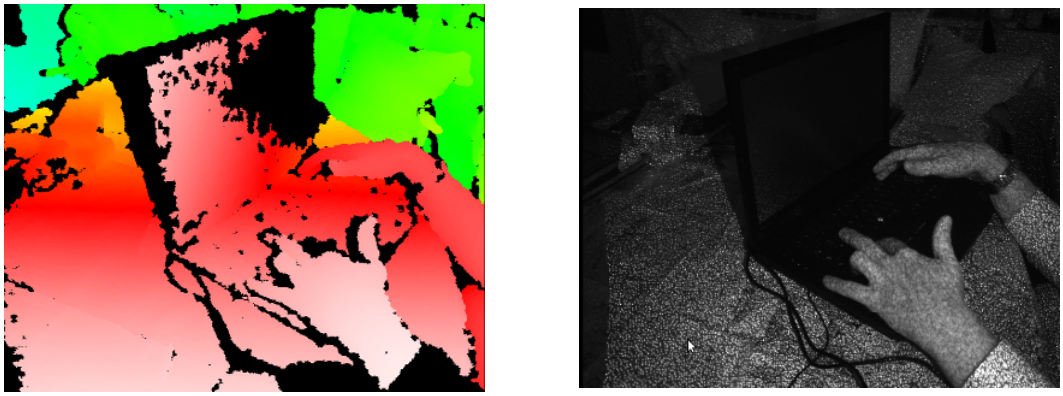
Obrázek 4.1: Fakultní robot TB2.

Robot obsahuje tyto části:

- **Roomba 530** - základna robota
- **notebook** - na kterém běží "mozek" robota a to ROS Electric
- **laser-scaner** - přesněji Sick LMS100 s rozsahem 270°
- **robotické rameno** - CrustCrawler AX-18A Smart Robotic Arm
- **Microsoft Kinect** - kamera, umístěna v horní části robota

4.1 Microsoft Kinect

Microsoft Kinect byl nejprve vyvíjen jako ovládací prvek k herní konzoli XBOX 360, ale později Microsoft uvolnil potřebné knihovny pro využití i mimo tuto herní konzoli. Kinect obsahuje RGB kameru, sensor pro zjištění hloubky a mikrofon. Nebo-li Kinect, produkuje RGB-D stream o rozlišení 640x480 pixelů o 30 Hz. Právě údaje o hloubce jsou velmi důležité pro výpočet vzdáleností předmětů od robota. Na obrázcích 4.2 můžete vidět ukázky dat z Kinectu.



(a) Barevné zobrazení

(b) Ve stupni šedi

Obrázek 4.2: Zobrazení deepth dat.

Kapitola 5

Návrh řešení

Návrh aplikace vychází ze základního požadavku - intuitivní ovládání robota. Pokud UI pro robota je vyvíjeno na zařízení s dotykovou obrazovkou, pak je nejintuitivnější ovládání pomocí gest. V tomto směru byla inspirací společnost Google a jejich *street-view* mapy, které se ovládají pomocí určení pozice na snímku.

Obrázek 5.1: Jednoduché schéma ovládání.

Schématický návrh aplikace můžeme vidět na obrázku 5.2, kde je v levém dolním rohu umístěn prvek zobrazující mapu, laser-scan a polohu robota. Dále v horním pravém rohu je umístěno tlačítko zobrazující nastavení aplikace a v horním levém rohu je umístěno tlačítko zobrazující diagnostické informace. Video z robota je umístěno na celém pozadí aplikace tak, aby uživatel mohl jednoduše označit (určit) požadované místo.

Obrázek 5.2: Schéma navrhované aplikace.

Výsledný návrh tedy obsahuje jak standardní prvky

- **video** - on-line video stream z Kinectu,
- **mapa** - 2D náhled mapy okolí,
- **scan** - 2D vizualizace laser-scan,
- **diagnostické informace**,
- **nastavení** - editace názvu *topics* a přepínání ovládání,

tak i prvky nové jako

- **ovládání** - *street-view* s rotací pomocí gest,
- **alternativní ovládání** - *g-sensor*.

5.1 Diagnostické informace

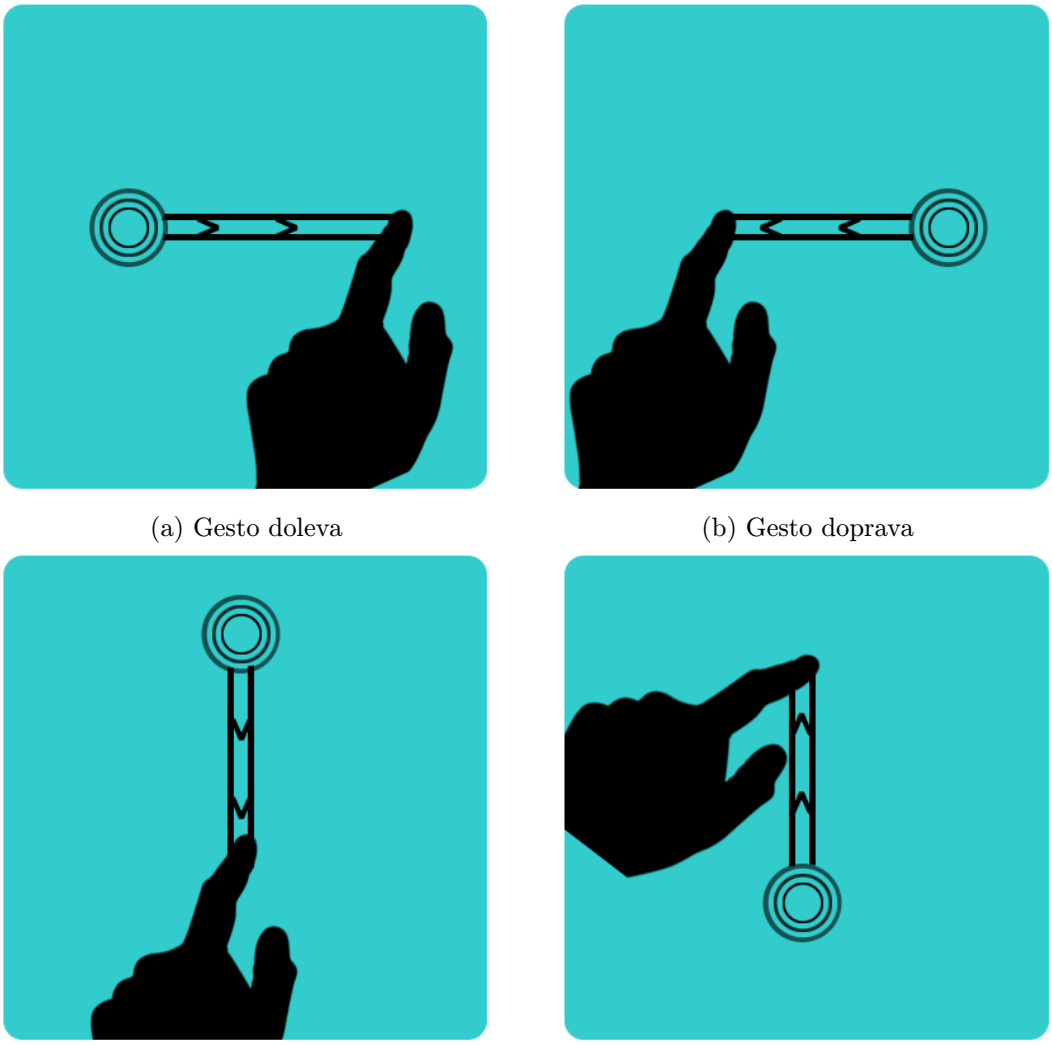
Při návrhu diagnostických informací byl největší inspirací program *robot_monitor*, který využívá *diagnostic_aggregator* pro získávání dat. Protože zatím neexistuje upravený *diagnostic_aggregator* pro TB2, vycházel jsem při návrhu z dostupných dat pro PR2. Na základě těchto skutečností by prostor pro diagnostické údaje měl obsahovat následující skupiny informací

- CPU,
- Notebook,
- Sensor,
- Network,
- Motors,
- Others.

5.2 Gesta

Na obrázcích 5.3 můžeme vidět jak budou rozpoznávána gesta na obrazovce. Tato gesta určují následující akce

- **doleva** - obr. 5.3a - rotace robota směrem doleva,
- **doprava** - obr. 5.3b - rotace robota směrem doprava,
- **dopředu** - obr. 5.3c - mírný pohyb robota směrem vpřed,
- **dozadu** - obr. 5.3d - mírný pohyb robota směrem vzad.



(a) Gesto doleva

(b) Gesto doprava

(c) Gesto dopředu

(d) Gesto dozadu

Obrázek 5.3: Grafické znázornění gest.

5.3 Alternativní ovládání

Jako alternativní ovládání bude aplikace používat *G-sensor*, který obsahuje většina zařízení se systémem Android. Tento princip ovládání spočívá v naklánění zařízení ve směru, v kterém chceme, aby se robot pohyboval.

5.4 Další možnosti UI

U většiny robotů jsou *topics* různě pojmenované, proto je u vytvořené aplikace důležité, aby mohla dynamicky měnit názvy těchto *topics* ve svém nastavení (menu). V rámci možného budoucího vývoje obsahuje vytvořená aplikace další funcionality. Jedna z funkcí bude akce dlouhého stisku na obrazovku a tím vyvolání submenu, které by mohlo obsahovat například položku *follow person*.

Jako jazyk pro aplikaci byla zvolena angličtina, která je pro aplikace v robotice standardem.

5.5 Ovládání inspirované Street View aplikací

Aplikace *street-view* pochází od společnosti Google a využívá se k navigaci v mapě. Robot se bude navigovat obdobným principem jako tato aplikace. V případě navrhovaného UI bude princip navigace robota spočívat v označení bodu na dotykové obrazovce zařízení, kde bude zobrazeno video stream z robota (viz obr. 5.1). Pomocí Kinectu, který produkuje RGB-D data, se na základě informací o hloubce v obrazu (*depth data*) vypočítají cílové 3D souřadnice. Otáčení robota na místě je řešeno pomocí gest doleva a doprava tak, že se vždy robot otočí cca o 20°. Mírný pohyb vpřed a vzad je řešen pomocí gesta přiblížit resp. oddálit.

5.6 Mobil vs. Tablet

Jak již jsem uvedl výše, bude aplikace fungovat, jak na mobilních zařízeních, tak na tabletech. Zde se nastoluje otázka týkající se dynamického zobrazování videa v aplikaci, protože Kinect produkuje obrazová data v rozlišení 640x480 pixelů, ale rozlišení na zařízeních se systémem Android jsou různá. Pro lepší uživatelský komfort se video z robota zobrazuje na maximální plochu na obrazovce, takže je potřeba při označení cílové pozice přepočítat souřadnice do intervalu hodnot pro Kinect.

Kapitola 6

Implementace

Následující kapitola se věnuje implementaci vytvořené aplikace, komunikaci mezi zařízením se systémem Android a robotem včetně popisu tříd. Aplikace byla implementována v jazyce Java a jako vývojové prostředí jsem použil Eclipse, protože v dnešní době nabízí asi největší komfort pro vývojáře na systému Android.

6.1 Verze Androidu

Pro vývoj aplikace jsem si vybral Android API 14, což je jiné označení pro verzi systému Android 4.0.4, a to z důvodu použití prvků ROSJavy z podbalíčku *android_core*. Na schématu 3.2 lze vidět, že základní prvky, jako je zobrazení videa z robota, jsou v nižším podbalíčku pro API 9, ale prvky zobrazující mapu a laser-scan jsou až v podbalíčku pro API 13. Dle článku [4] pojednávajícího o častých chybách při vývoji UI pro roboty je důležitým prvkem jak video, tak i informace o umístění robota v prostoru, případně vyznačení vzdáleností robota od předmětů. A proto jsem se rozhodl do aplikace zařadit jak mapu s laser-scanem, tak video.

6.2 Komunikace

Aplikace využívá ke komunikaci s robotem jak *topics*, tak i *services*. Samotné navázání komunikace s robotem je provedeno pomocí `NodeMainExecutorService` z frameworku. Protože je aplikace primárně vyvíjena pro fakultního robota TB2, tak i výchozí nastavení *topics* v aplikaci je přizpůsobeno pro tohoto robota. Následující tabulka 6.1 znázorňuje, jaké *topics* budou v aplikaci využity.

Výchozí topic	**Popis**
/map	Topic obsahující data pro 2D mapu okolí robota.
/scan	Topic obsahující data z laser-scanu.
/cam3d/rgb/ image_color_flipped/compressed	Topic obsahující komprimovaná data z Kinectu.
/cmd_vel_safe	Topic, který slouží pro základní pohyb robota.
/move_base_simple/goal	Topic, na který se publikuje zadaný cíl.

Tabulka 6.1: Přehled výchozích *topics*.

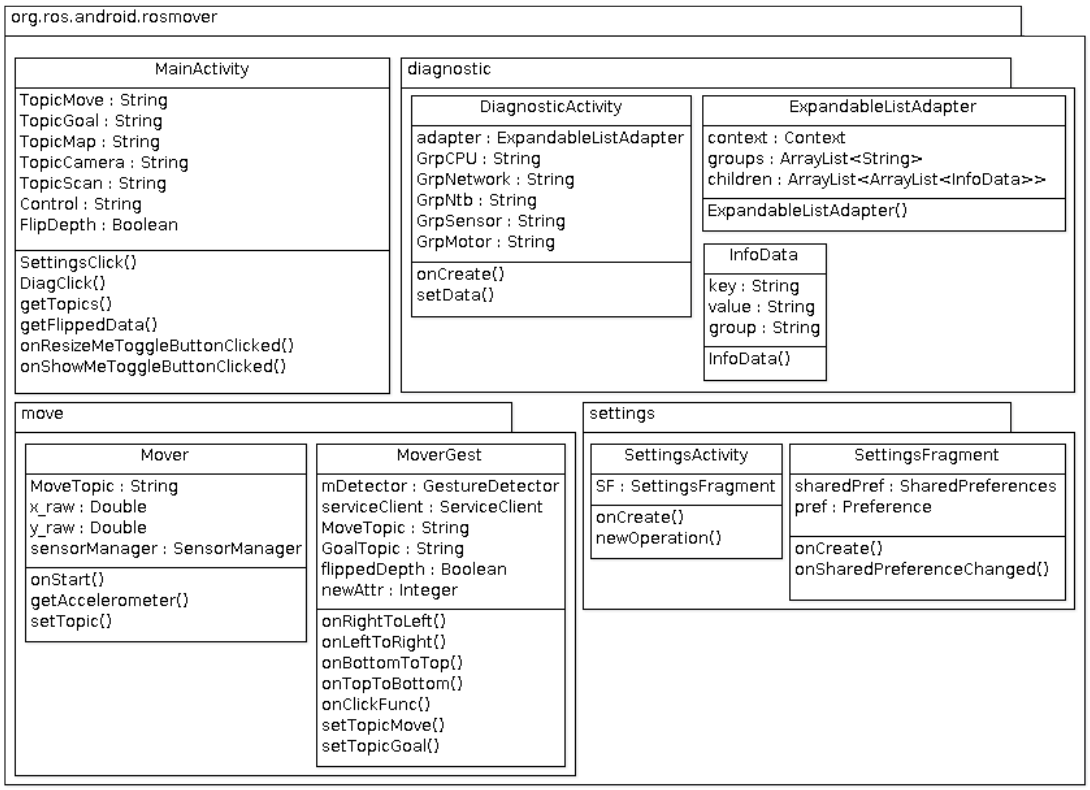
Uvedené *topics* reprezentují aktuální stav na robotovi. Aby byla aplikace použitelná po delší časový úsek, umožňuje editaci těchto *topic*ů pomocí nastavení.

6.2.1 BB_estimator

Jak již bylo uvedeno výše, bude aplikace využívat *services* a to **BB_estimator** [7]. Tento *service* byl vytvořen na FIT VUT v Brně studentem Tomášem Hodaňem pod vedením Dr. Španěla a slouží k výpočtu 3D souřadnic objektů na základě 2D hloubkových dat z Kinectu. Vytvořená aplikace bude tento *service* využívat pouze ve zjednodušené formě a to pro výpočet cílové souřadnice.

6.3 Popis tříd

Díky tomu, že jsem při implementaci využil skoro všechny dostupné prostředky z frameworku, nebylo nutné, abych všechny třídy sám implementoval, ale pouze jsem je použil, např. komunikaci s robotem. Na obrázku 6.1 je diagram tříd, které jsem implementoval. Jednotlivé třídy budou popsány níže.



Obrázek 6.1: Diagram tříd aplikace.

MainActivity

Třída `MainActivity` je hlavní okno aplikace. Při startu *MainActivity* se nejprve spustí okno pro navázání komunikace s robotem a poté se provede samotná třída. Tato třída načítá vlastnosti z nastavení aplikace a dle parametrů vytváří nové objekty pro video, mapu, laser-scan a ovládání. Prvek zobrazující mapu, laser-scan a polohu robota je `VisualizationView` z druhého podbalíčku ROSJavy. Pro zobrazení videa z robota je použit prvek `RosImageView` s komprimovaným zdrojem dat. Po inicializaci všech objektů aplikace provede zaregistrování objektů k jednotlivým *topics*, a to pomocí metody `execute`. Součástí této třídy jsou grafické prvky pro zobrazování videa a mapy a tlačítek pro zobrazení dalších oken aplikace. Dále tato třída umožňuje zobrazování submenu při akci dlouhého stisku na obrazovce.

Protože na zařízeních se systémem Android není vždy stejné rozlišení, musí se některé grafické prvky dynamicky přizpůsobovat. Takovým prvkem je právě prostor pro mapu, laser-scan a polohu robota. Tento prvek se dynamicky nastavuje při spouštění hlavní aktivity a to v rozměrech $0,3 * \textit{výška obrazovky}$ a $0,25 * \textit{šířka obrazovky}$.

SettingsActivity a SettingsFragment

Třídy `SettingsActivity` a `SettingsFragment` slouží pro zobrazení a ukládání nastavení aplikace. Třída `SettingsFragment` rozšiřuje třídu `PreferenceFragment`, která slouží pro standardní nastavení v systému Android. Data pro tento fragment jsou uložena v XML souboru *preferences.xml* ve složce `/res/xml`. Tento XML soubor, mimo popisků a definování jednotlivých způsobů nastavení, definuje i výchozí hodnoty pro jednotlivé možnosti. Tato třída také implementuje *listener* `OnSharedPreferenceChangeListener` sloužící pro zobrazování hodnot, které si uživatel aktuálně nastavuje.

Mover

Třída `Mover` implementuje pohyb robota pomocí G-sensoru. Hlavní částí této třídy je `SensorManager`, který zjisťuje aktuální polohu zařízení a následně posílá robotovi pokyny k pohybu. Pro lepší manipulaci s robotem je náklon zařízení směrem vpřed a vzad dělen konstantou `6`, aby ovládání bylo jednodušší. Při náklonech doprava a doleva je na robota posílána konstanta `+/- 2,5`.

MoverGest

Třída `MoverGest` implementuje pohyb robota pomocí označení cílové pozice na obrazovce a pomocí gest na obrazovce. Detekci gest na obrazovce zajišťuje `GestureDetector` a jejich rozpoznání na jednotlivá gesta je provedeno tak, jak bylo naznačeno na obrázku 5.3. V této třídě je volán klient *service* bb_estimator, který vrací cílovou pozici v reálném světě. Při úspěšném získání cílové pozice jsou poslána data na *topic* zajišťující autonomní naplánování trasy na určené souřadnice.

DiagnosticActivity a ExpandableListAdapter

Třídy `DiagnosticActivity` a `ExpandableListAdapter` slouží pro zobrazování diagnostických informací. Pro zobrazení podobné jako je přímo v aplikaci *robot_monitor* a nebo její odvozeniny pro Android *ROS Android Robot Monitor*, jsem zvolil prvek rozevírajícího seznamu, který jsem implementoval dle tutoriálu [5].

InfoData

Tato třída `InfoData` reprezentuje jednotlivé diagnostické informace, jejímiž atributy jsou klíč (*key*), hodnota (*value*), příslušnost do skupiny (*group*) a chybová hodnota (*error*).

6.4 Ovládání

Samotné ovládání robota pomocí aplikace bylo popsáno v kapitole 5.5 o návrhu, ale ovládání aplikace ještě nebylo zmíněno. Při spuštění aplikace se zobrazí obrazovka s editovatelným řádkem pro URL robota. Tuto URL je možné napsat přímo pomocí klávesnice nebo lze použít tlačítko *scan* a pomocí QR scaneru si načíst QR kód umístěný na robotovi. Po připojení aplikace k robotovi se zobrazí všechny požadované grafické prvky. V levém dolním rohu je usmítěna mapa. Z tutorial aplikace bylo převzato tlačítko *Center Me*, které slouží pro zobrazení robota ve středu vymezeného prostoru. Pro větší uživatelský komfort jsem přidal tlačítko *Max Map/Min Map* pro dvojnásobné zvětšení/zmenšení mapy.

6.5 Problémy při implementaci

Při implementaci aplikace jsem narazil na několik problémů, které jsem musel vyřešit. Jeden z prvných problémů byl s QR scanerem, kdy jsem musel v prvním podbalíčku ROSJavy editovat soubor *MasterChooser.java* v kontrole výsledku čtečky. Dále při zobrazení mapy z robota jsem musel změnit prvek z druhého podbalíčku ROSJavy z `CompressedOccupancyGridLayer` na `OccupancyGridLayer` o proti ukázkové aplikaci z tutoriálu.

Nejčastější problémem, který se obecně vyskytuje u aplikace vytvořené na základě ROSJavy, je špatné či žádné nastavení globálních proměnných na robotovi. To znamená, že pro správnou funkci aplikace je potřeba na robotovi nastavit proměnné `ROS_IP` a nejlépe i `ROS_MASTER_URI`.

Kapitola 7

Testování a výsledky

V této kapitole bude popsáno testování aplikace, a to jak testování funkčnosti pomocí simulace, tak i testování na reálném robotu. Všechny testy byly provedeny jak na mobilním zařízení, tak i na různých tabletech. V následující tabulce 7.1 je uvedena bližší specifikace jednotlivých zařízení.

Název	**Verze Androidu**	**Velikost obrazovky**	**Rozlišení [px]**
Acer A701	4.1.1	10,1”	1920x1200
Prestigio PMP5570C	4.0.4	7”	1024x600
HTC One S	4.1.1	4,3”	960x540

Tabulka 7.1: Přehled zařízení použitých při testování.

7.1 Testování funkcionality

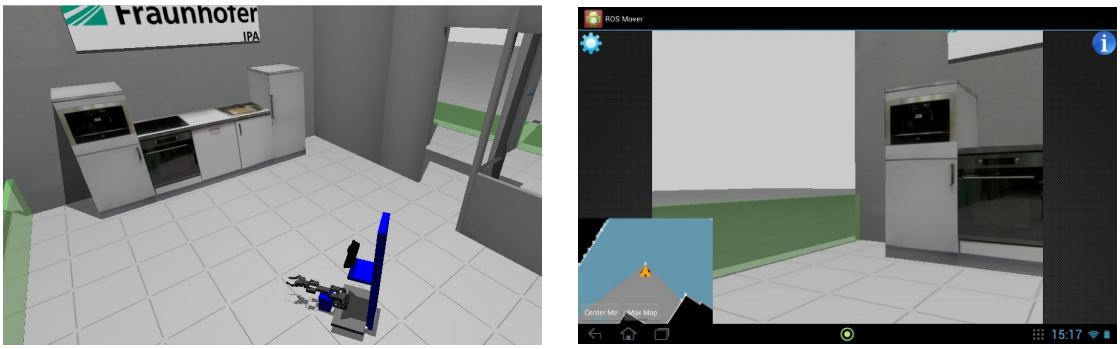
Základní testování funkcionality bylo provedno na simulovaném robotovi a to na platformě **ROS Electric** na *Ubuntu 11.10* a na platformě **ROS Fuerte** na *Ubuntu 12.04*. Nejprve byla aplikace vyzkoušena na základním robotovi *Turtlebot*, kde se zkoušela pouze základní funkčnost, protože při testování nového ovládání typu street-view simulace zapisovala do mapy špatné informace, takže ovládání nemohlo být plně odzkoušeno. Po zkoušce navázání komunikace, příjmu informací z kamery a poslání pokynů k pohybu s *Turtlebot*em, byla aplikace vyzkoušena se simulací fakultního robota *TB2*.

Při testování aplikace se simulací fakultního robota *TB2* byly zjištěny malé nedostatky, jako například udávání pozice na obrazovce. Tento problém byl zapříčiněn tím, že Kinect kamera je namontována na robotovi obráceně a pokud využíváme *topic*u, který již posílá inverzní data, pak se při označení místa na obrazovce musí nejprve pozice převrátit, aby byl proveden dotaz na správné místo. Po odstranění těchto dílčích nedostatků byla aplikace vyzkoušena na všech uvedených zařízeních v tabulce 7.1 a při všech dalších testech aplikace fungovala správně. U zařízení Prestigio se objevil problém, kde animace při zvětšování a zmenšování mapy nebyla tak plynulá, jak by měla být, ale byla skoková.

Při testování aplikace se simulací bylo využito všech standardních nástrojů pro ladění v upravené verzi Eclipse ADT (*Android Developer Tools*) jako je debugger a LogCat.

7.1.1 Ukázka testování se simulací

Na následujícím obrázku 7.1a je vidět simulovaný robot *TB2* v programu Gazebo a na obrázku 7.1b je zachycen snímek obrazovky z aplikace.



(a) Simulovaný robot *TB2*.

(b) Snímek z aplikace.

Obrázek 7.1: Testování aplikace se simulací robota.

7.2 Testování s reálným robotem

Testování s reálným robotem se uskutečnilo za účelem ověření funkčnosti aplikace v reálném světě. Toto testování proběhlo celkem 3x za odborným dohledem Ing. Materny, autora robota. Každé z těchto testování trvalo cca 4 hodiny. První testování bylo čistě zkušební v kanceláři, druhé proběhlo již ve větším prostoru. Poslední testování bylo také provedeno ve větších prostorách s finální verzí vytvořené aplikace.

První testování proběhlo v kanceláři. V jeho průběhu bylo odzkoušeno pouze navázání komunikace, příjem dat z kamery a poslání základních příkazů k pohybu. K testování nového ovládání street-view jsem se během této zkoušky nedostal z důvodu drobných problémů a dodatečného překládání dalších programů na robotovi.

Druhé testování již proběhlo na chodbě fakulty, kde byly zajištěny dostatečné prostorové možnosti pro vyzkoušení všech funkcionalit aplikace. Během tohoto testování byly nalezeny drobné chyby v aplikaci a také byla zjištěna potřeba nutnosti zvětšení mapy. Ani při tomto testování nebylo možné vyzkoušet nové ovládání street-view, protože se na robotovi nepodařilo vypočítat cílové souřadnice a to z důvodu problému v hloubkových datech získávaných z kamery Kinect. Tento problém během pár dnů Ing. Materna vyřešil a bylo pak možné přistoupit ke kompletnímu testování aplikace.

Poslední testování proběhlo opět ve větších prostorách a sloužilo ke kompletnímu testu aplikace. Během tohoto testování byla aplikace kalibrována tak, aby ovládání bylo plynulejší a to jak pro ovládání pomocí gest, tak i ovládání pomocí G-sensoru, protože při testování v simulaci byla reakce simulovaného robota mírně odlišná. V rámci testování byly odzkoušeny všechny funkcionality aplikace na všech zařízeních uvedných v tabulce 7.1. Úspěšná funkčnost aplikace byla povrzena testy na několika typech zařízení s odlišnými velikostmi obrazovek a s jinými verzemi systému Android.

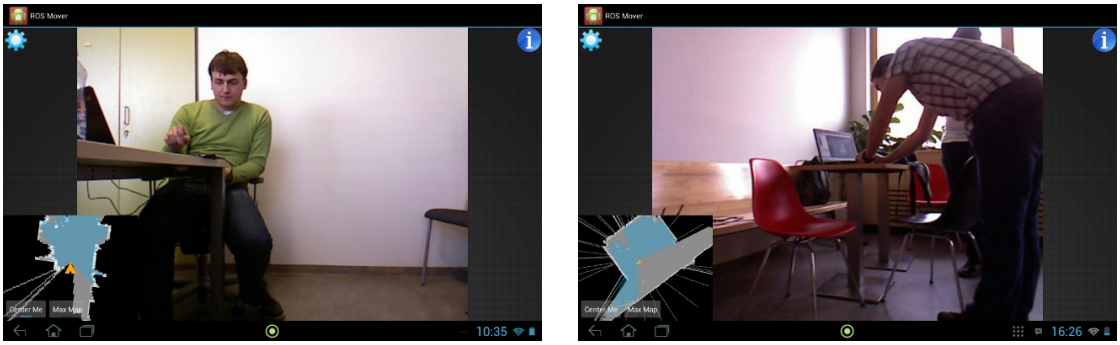
Funkcionalita	Acer A701	Prestigio PMP5570C	HTC One S
Zobrazení videa	✔	✔	✔
Zobrazení mapy a laser-scan	✔	✔	Funkční, ale v některých případech se nezobrazovalo
Nastavení	✔	✔	✔
Ovládání pomocí G-sensor	✔	převrácené osy XY	převrácené osy XY
Ovládání street-view s gesty	✔	✔	✔

Pozn: ✔ - plně funkční.
✔ - funkční, ale v jistých případech *BB_estimator* nevypočítal cílovou souřadnici

Tabulka 7.2: Přehled výsledku testování s různými zařízeními.

7.2.1 Ukázka z testování s reálným robotem

Na následujících obrázcích 7.2 lze vidět snímky obrazovky pořízené při testování s robotem *TB2*.



Obrázek 7.2: Testování aplikace s reálným robotem *TB2*.

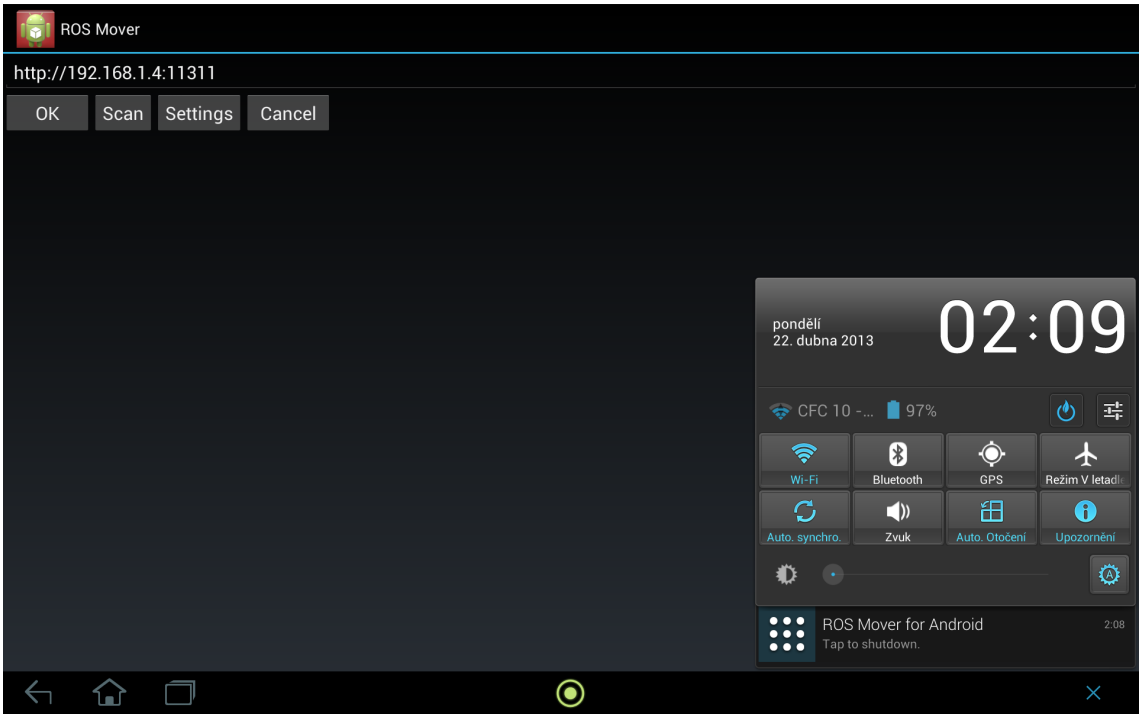
7.3 Poznatky z testování

Během testování bylo provedeno několik změn v ovládání. Jak už bylo zmíněno výše u ovládání pomocí G-sensoru, byl tento typ ovládání upraven tak, aby se robot ovládal jednodušeji. Úprava spočívala ve snížení citlivosti G-sensoru, takže pohyby robota při změně polohy senzoru jsou nyní plynulejší. Další změna proběhla u rozpoznávání gest. V první fázi bylo nutné provést několik gest k tomu, aby se robot výrazněji otočil. Po úpravě ovládání se nyní robot otáčí pomocí jednoho gesta o cca 20°. Dalším problémem, který se vyskytl v průběhu testování, je nepříliš přesný pohyb vpřed a vzad. Tato nepřesnost je způsobená mechanickými parametry koleček, která se neotáčí vždy stejně. Poslední zkušenost z testování je spojená s posíláním robota na určité místo - v současné verzi neumožňuje framework okamžitě robota zastavit. Pokud má robot nastaven cíl, snaží se do tohoto cíle dostat a není možné ho zastavit dříve, než dosáhne stanoveného místa.

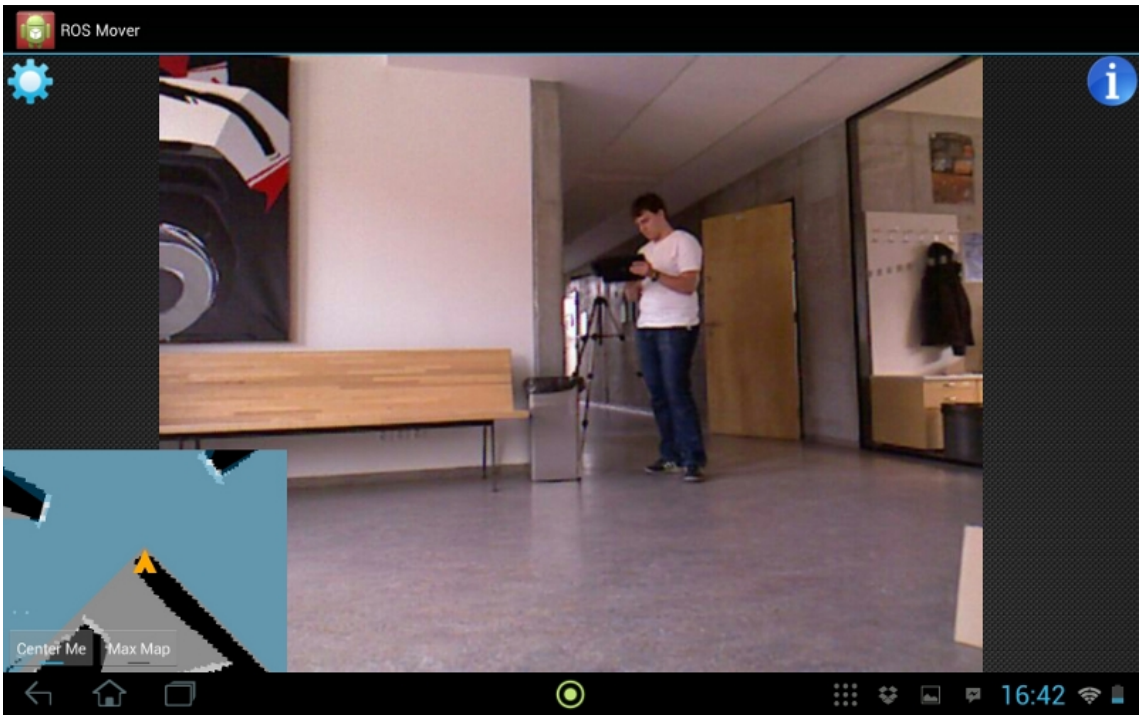
7.4 Ukázky UI

Na následujících obrázcích je možno vidět ukázky UI.

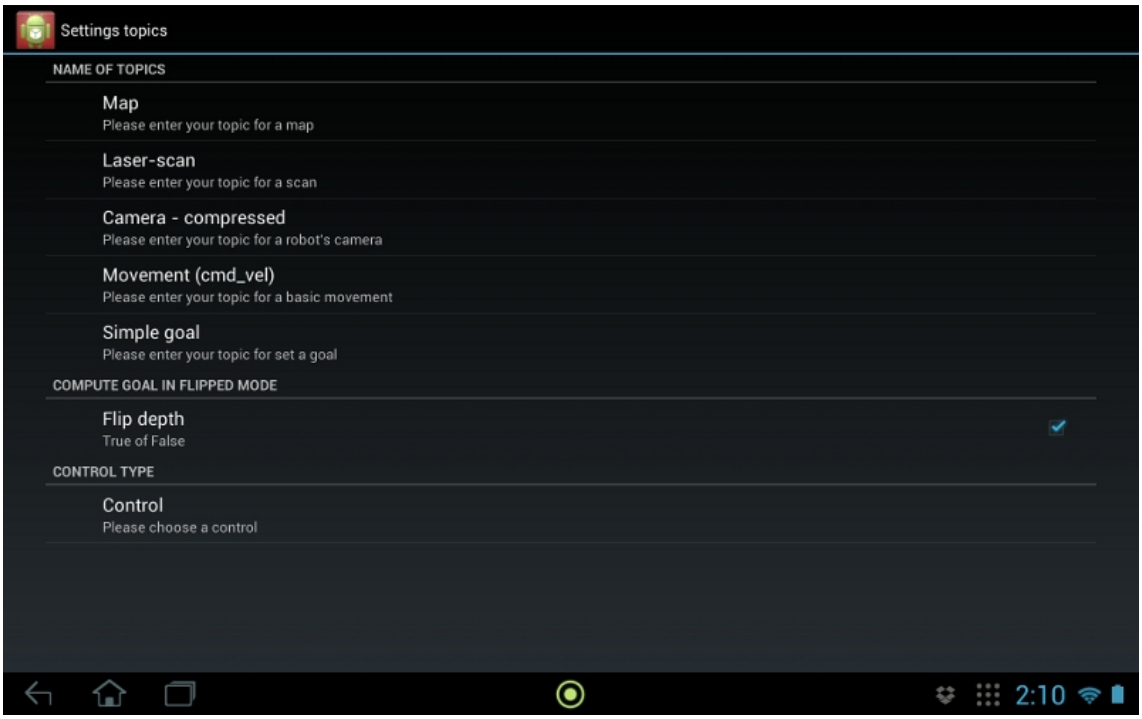
- **Úvodní obrazovka** - obr. 7.3 - zobrazuje obrazovku, kdy se buď zadá přímo URL robota nebo se načte pomocí QR čtečky,
- **Hlavní obrazovka** - obr. 7.4 - zobrazuje všechny grafické prvky, které byly zmíněny v návrhu aplikace,
- **Nastavení aplikace** - obr. 7.5 - na této obrazovce lze provést nastavení aplikace s tím, že se změny projeví až po restartování aplikace,
- **Zobrazení submenu** - obr. 7.6 - jedna z možných rozšiřujících funkcí,
- **Diagnostická obrazovka** - obr. 7.7 - zobrazuje diagnostické údaje z robota, ale obsahuje fiktivní data.



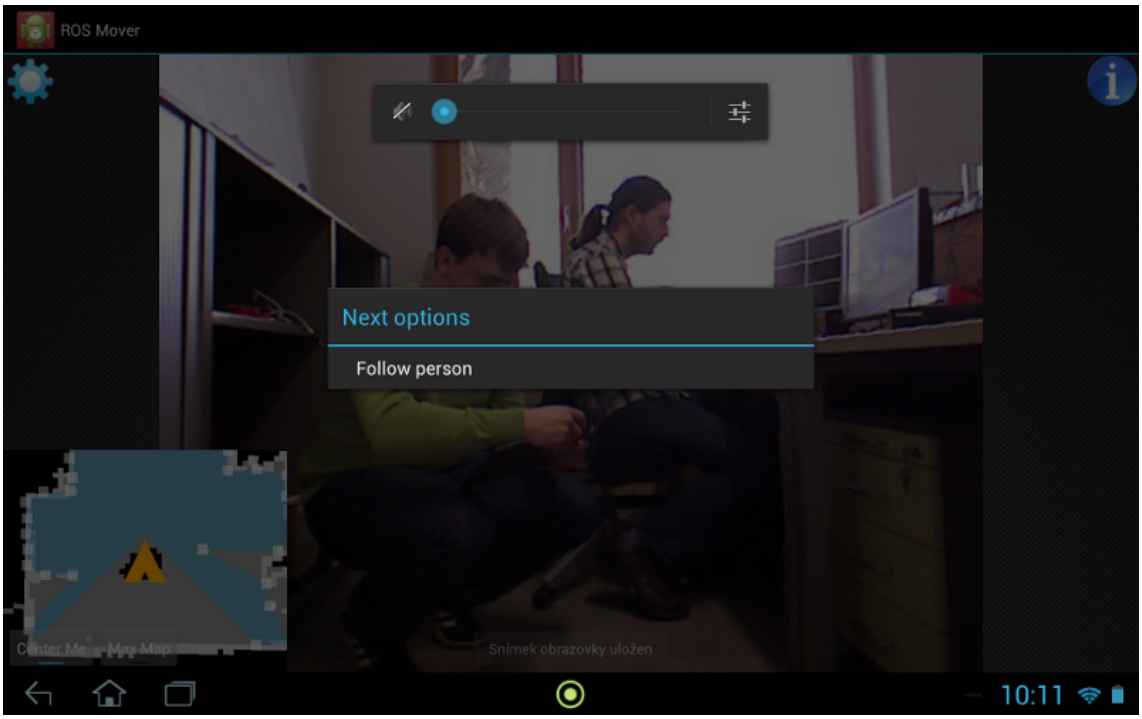
Obrázek 7.3: Ukázka UI - Okno pro navázání spojení a ukázkou vypínání aplikace.



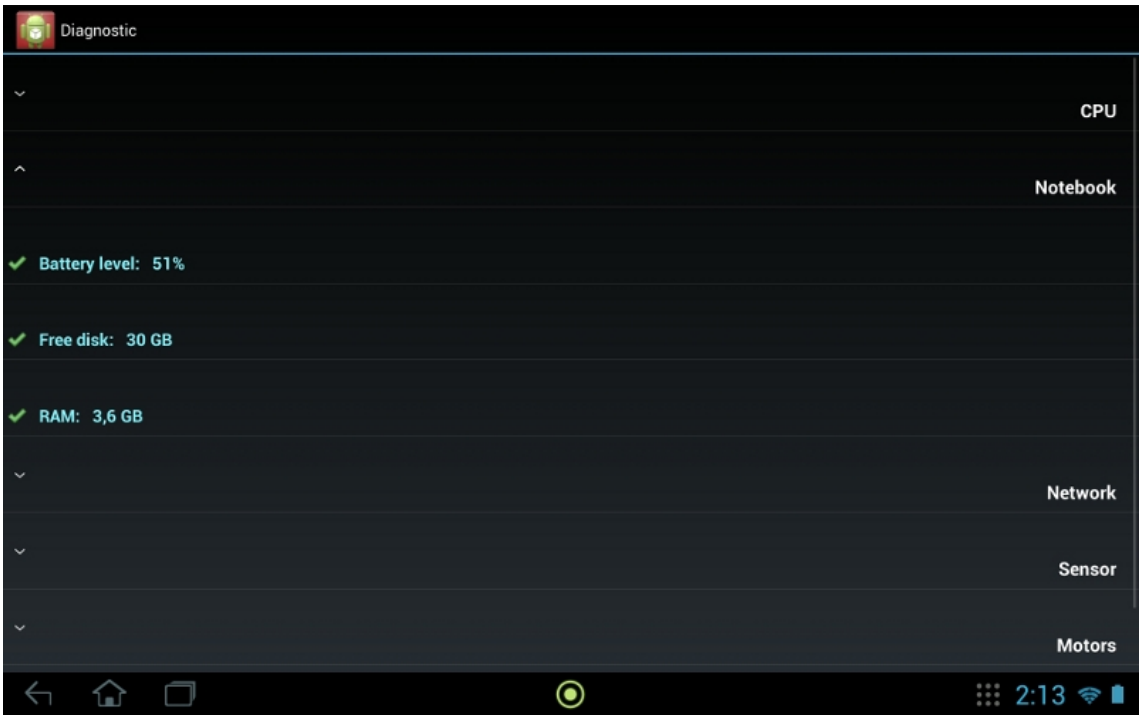
Obrázek 7.4: Ukázka UI - Hlavní okno aplikace.



Obrázek 7.5: Ukázka UI - Nastavení.



Obrázek 7.6: Ukázka UI - Zobrazení submenu.



Obrázek 7.7: Ukázka UI - Zobrazení diagnostických informací.

7.5 Budoucí vývoj

Samotné testování funkčnosti aplikace bylo časově velmi náročné a nebylo tak možné zajistit testování spojené s dotazníkem a nebo s několika testery. Bylo by vhodné v dalším vývoji zařadit do testování i tyto uvedené možnosti.

Při testování se ukázalo, že *service BB_estimator* nedokáže vždy vypočítat cílovou souřadnici, takže by bylo dobré v budoucím vývoji přepracovat výpočet cílového bodu přímo z hloubkové mapy (*depth map*). Zde je ovšem omezení v tom, že v simulaci nejsou tato data správně simulována, takže by se tato implementace musela intenzivě zkoušet přímo na robotovi, nebo pořídit záznam ze všech dat vycházejících z Kinectu a pracovat se záznamem. Dalším cílem budoucího vývoje by mohlo být napojení aplikace na reálný diagnostický *node* a zobrazovat tak reálná data.

Pro uživatele by mohla být také přínosná možnost změny zobrazení mapy a videa. Jednalo by se o prohození grafických prvků tak, aby mapa byla zobrazena na místě současného videa. Dalšími zajímavými vlastnostmi UI by mohlo být např. zobrazování současného cíle ve vizualizaci mapy nebo zobrazování vizualizace laser-scanu ve videu.

Kapitola 8

Závěr

Cílem mé bakalářské práce bylo vytvořit aplikaci pro systém Android, která ovládá vzdáleného robota, na kterém běží ROS. Během řešení této práce se nabídla možnost vytvořit nový typ ovládání inspirovaný aplikací Street View od společnosti Google. Funkčnost vytvořené aplikace byla ověřena jak v simulaci, tak s reálným robotem.

Aplikace byla primárně vyvíjena pro fakultního robota TB2, na kterém proběhlo i několik testování v rámci areálu FIT VUT v Brně. Toto testování proběhlo na několika zařízených se systémem Android, kde byly zástupci jak tabletů, tak telefonů. Hlavním typem ovládání je *street-view*, které spočívá v označení cíle na obrazovce, kde je zobrazováno video z kamery robota. Toto ovládání je doplněno o gesta pro rotaci a mírný pohyb vpřed/vzad robota. Dále aplikace umožnuje nastavení jednotlivých *topics* tak, aby aplikace byla použitelná pro všechny roboty, na kterých běží ROS. Poslední funkcionalitou je možnost zobrazení diagnostických informací o robotovi, které by v budoucím vývoji bylo potřeba napojit na diagnostický *node* na robotovi, který doposud není dostupný.

Přínosem této práce, mimo funkční aplikace, která ovládá vzdáleného robota dvěma způsoby, bylo vytvoření dokumentu, který zachycuje poznatky získané během implementace aplikace. Tento dokument by měl pomoci budoucím řešitelům, kteří budou pracovat s ROSJavou, k rychlejšímu pochopení daného frameworku.

Literatura

[1] Oficiální stránky ROS [online]. 22.02.2013, [cit. 2013-03-24].
URL http://www.ros.org/

[2] Aboudaya, E.: *Mobile Teleoperation of a Mobile Robot*. Diplomová práce, Lappeenranta University of Technology, Lappeenranta, 2010.

[3] Cavallin, K.; Svensson, P.: *Semi-Autonomous, Teleoperated Search and Rescue Robot*. Diplomová práce, Umeå University, Umeå, 2009.

[4] Chen, J. Y. C.; Haas, E.; Barnes, M.: *Human Performance Issues and User Interface Design for Teleoperated Robots*. 2007, IEEE transactions on systems, man, and cybernetics-part c:applications and reviews ,s. 1231-1245.

[5] Gupta, G.: Tutoriál - How to use ExpandableListView [online]. 01.2012, [cit. 2013-04-08].
URL androidword.blogspot.cz/2012/01/how-to-use-expandablelistview.html

[6] Hainsworth, D. W.: *Teleoperation User Interfaces For Mining Robotics*. Brisbane, 2000, CSIRO Exploration & Mining.

[7] Hodaň, T.: Oficiální stránky *service* BB_estimator [online]. 30.08.2012, [cit. 2013-04-08].
URL http://www.ros.org/wiki/bb_estimator

[8] Kohler, D.: Oficiální stránky projektu rosjava [online]. [cit. 2013-03-24].
URL https://code.google.com/p/rosjava/

[9] Zimmerman, A.: Oficiální stránky aplikace Rviz on Android [online]. 28.08.2012, [cit. 2013-03-24].
URL https://bitbucket.org/zimmrmn3/rviz-for-android/wiki/Home

Příloha A

Obsah CD

Přiložené CD obsahuje

- `src` - zdrojové kódy aplikace
- `apk` - výsledná aplikace ve formátu *.apk - ROSMover.apk
- `rest` - manuál k ROSJave, plakát a el. verze tech. zprávy
- Readme - základní popis

Příloha B

Manuál

Ve složce apk na přiloženém CD nalezneme soubor ROSMover.apk, který nahrajeme do zařízení se systémem Android. Nahrání na zařízení můžeme provést v příkazové řádce pomocí nasledujícího příkazu, pokud máme připojeno zařízení pomocí USB kabelu.

```
ant installd
```

Po nahrátí aplikace ji spustíme a budeme vyzváni k zadání IP adresy robota, na kterém běží ROS. Po správném zádáním a odsouhlasení adresy se zobrazí hlavní activita aplikace.

Příloha C

Ukázka vytvořeného plakátu