VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

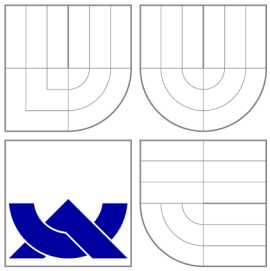
VYUŽITÍ KAMERY KVADROKOPTÉRY PRO SLEDOVÁNÍ KONTEXTU SCÉNY VE VENKOVNÍM PROSTŘEDÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

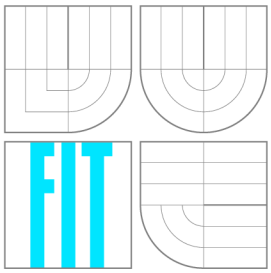
AUTOR PRÁCE
AUTHOR

PETR BĚL

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYUŽITÍ KAMERY KVADROKOPTÉRY PRO SLEDOVÁNÍ KONTEXTU SCÉNY VE VENKOVNÍM PROSTŘEDÍ

QUADCOPTER CAMERA UTILIZATION FOR SCENE CONTEXT MONITORING

IN OUTDOOR ENVIRONMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR BĚL
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÍTĚZSLAV BERAN, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací uživatelského rozhraní pro ovládání kvadrokoptéry mobilním zařízením. Nejdůležitějším aspektem návrhu je schopnost sledování scény kamerou nesenou kvadrokoptérou. Zvolené kritérium bylo řešeno ovládáním gesty kreslenými na displej zařízení, a tedy vypuštěním zbytných grafických ovládacích prvků. Komfort sledování scény je zajištěn více módy letu za použití stejné sady gest. V práci je pro spojení robota s mobilním zařízením zvolena platforma Robot Operating System. Její přenesení na mobilní zařízení bylo uskutečněno knihovnou RosJava pro zvolený operační systém Android. Hlavním přínosem je vytvoření mobilní aplikace umožňující pilotovat kvadrokoptéru a pozorovat přenášený obraz z její kamery nerušený ovládacími prvky. Tato přednost je důležitá především pro zařízení s menšími displeji, jako jsou například chytré telefony.

Abstract

This bachelor thesis deals with design and implementation of user interface for quadcopter control by mobile device. Most important design aspect is ability to scene monitoring by camera carried by quadcopter. Selected criterion was solved by control gestures drawn on device display, thus deletion of unnecessary graphical control elements. Comfort for scene monitoring is ensured by multiple flight modes using same set of gestures. For purpose of connecting robot with mobile device in this work has been chosen platform Robot Operating System. Its transfer to mobile device was implemented by RosJava library for the selected operating system Android. Main contribution of this thesis is creation of mobile application which allows to pilot quadcopter and watch transmitted image from its camera not being disturbed by control elements. This advantage is most important for devices with smaller displays like smartphones.

Klíčová slova

ROS, kvadrokoptéra, řízení letu gesty, Android, chytrý telefon, AscTec Pelican, RosJava, sledování scény

Keywords

ROS, quadcopter, gestures flight control, Android, smartphone, AscTec Pelican, Rosjava, scene monitoring

Citace

Petr Běl: Využití kamery kvadrokoptéry
pro sledování kontextu scény
ve venkovním prostředí, bakalářská práce, Brno, FIT VUT v Brně, 2014

Využití kamery kvadrokoptéry pro sledování kontextu scény ve venkovním prostředí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Běl
21. května 2014

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Vítězslavu Beranovi, Ph.D, za odbornou pomoc a kontruktivní konzultace. Dále bych chtěl poděkovat panu Ing. Václavu Šimkovi za pomoc řešením hardwarových problémů.

Obsah

Kapitola 1

Úvod

Kvadrokoptéry, tedy létající roboti se čtyřmi rotory, se poslední dobou těší velké popularitě. Pro svou obratnost nacházejí využití v mnohých oborech, například ve výzkumu teorie řízení letu, letecké navigace, robotice. Mimo výzkumu se využívají kvadrokoptéry v praxi, kdy je používají novináři, bezpečnostní složky pro průzkum terénu a v neposlední řadě ve vojenském průmyslu. Také velké firmy zvažují použití kvadrokoptér či hexakoptér pro doručování balíků koncovým zákazníkům.

Tyto přístroje jsou schopné velmi obratného letu a mohou se držet ve vzduchu na jednom místě, oproti strojům používajícím k letu pevná křídla. Toto velmi rozšiřuje možnosti zkoumání okolí a určitých bodů zájmu různými senzory. Tyto stroje s použitím moderních baterií mají dostatečný výkon, aby dovolovaly nést nejen základní senzory, ale krom specializované řídicí desky také výkonnější počítač, který umožnuje dívat se na řízení kvadrokoptéry s vyšší abstrakcí a rovněž umožňuje komunikaci s dalšími zařízeními včetně moderních chytrých telefonů a tabletů.

Tato bakalářská práce má za cíl prozkoumat právě možnost intuitivního a efektivního ovládání kvadrokoptéry mobilním zařízením při sledování scény ve venkovním prostředí. Navrhnout uživatelské rozhraní, které umožňuje získávat přehledně informace z kvadrokoptéry, například obraz z kamery, či údaje o poloze, a zároveň dokáže řídit kvadrokoptéru pohodlněji než užitím klasického rádiového ovladače.

Na poli ovládání kvadrokoptér mobilními zařízeními existuje již několik většinou proprietárních řešení, proto bude prvním krokem zjištění existujících řešení a jejich srovnání. Dalším krokem bude studium jednotlivých mobilních zařízení. Chytré telefony nebo tablety nabízejí spoustu možností, jak ovládat pohyb kvadrokoptéry, například multitouch, hlasové ovládání, gesta, gyroskop, či ovládání kamerou. Při samotném návrhu se nesmí opomenout fakt, že zařízení podporují různé operační systémy, a je tedy nutné zvolit vhodné nástroje pro vývoj mobilních aplikací.

Všechny experimenty budou prováděny na platformě AscTec Pelican pojmenované Tyra. Tato platforma má dvě desky - specializovanou desku s vestavěným systémem autopilota a Atomboard, kde je nainstalován UNIXový operační systém a framework Robot Operating System (`ROS`). V zadání této práce je využít knihovnu `ROS` pro komunikaci s kvadrokoptérou, správu dat, experimenty a vizualizaci.

Práce obsahuje velké množství zkratek a pojmů, proto je v příloze B uveden seznam pojmů s vysvětlením či odkazem do textu.

Kapitola 2

Teorie

Tato kapitola přibližuje teorii nezbytnou pro pochopení celé práce. Nejdůležitější je pochopení platformy Robot Operating System (ROS) a její nástavby pro komunikaci s jinými zařízeními. Bude uvedena problematika vývoje aplikací pro mobilní zařízení a především pro ty s operačním systémem Android. Další část pojednává o konkrétních kvadrokoptérách a jejich dosavadních možnostech ovládání. Čtenář bude také seznámen s kvadrokoptérou AscTec Pelican, na které budou probíhat experimenty.

2.1 Robot Operating System

Stabilizace letu kvadrokoptéry není předmětem této práce, a proto bude ovládání příkonu jednotlivých motorů ponecháno softwaru třetí strany. Tato sekce seznamuje čtenáře se systémem zajišťující abstrakci tohoto problému, a tím je Robot Operating System. Důležitým zdrojem informací krom webových stránek byla kniha Learning ROS for Robotics Programming[1].

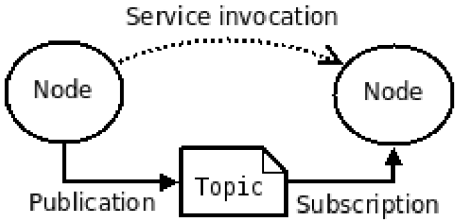
Tento open-source projekt si klade za cíl zjednodušit vývoj softwaru v oblasti robotiky. Jedná se o framework s celou řadou nástrojů a rozšíření. Tento framework se neustále vyvíjí a je podporován širokou komunitou vývojářů i uživatelů po celém světě.

Robot Operating System (dále ROS) se původně jmenoval Switchyard a byl zaštítěn projektem STAIR[1] ve Stanfordské laboratoři umělé inteligence SAIL[2]. V roce 2008 pokračoval především v robotickém výzkumném inkubátoru Willow Garage[3] s více než dvaceti spolupracujícími institucemi.

Celá kolekce nástrojů, knihoven a konvencí zjednodušuje vytváření robustního univerzálního softwaru pro roboty. ROS přináší standardní služby operačních systémů, jako například implementaci běžně používaných funkcí, nízkoúrovňové ovládání zařízení, hardwarovou abstrakci, správu balíčků a zasílání zpráv mezi procesy.

Architektura ROS frameworku se dá popsat jako architektura grafu. Jedná se o peer-to-peer síť, která zpracovává data dohromady. Základními jednotkami výpočetního grafu jsou uzly, Master, Parameter Server, zprávy, služby a vlákna. ROS poskytuje různé styly komunikace, například synchronní RPC komunikaci službami, asynchronní stream dat skrze vlákna a uložení dat na Parameter Server[4].

[1] STanford AI Robot http://stair.stanford.edu
[2] Stanford Artificial Intelligence Laboratory http://ai.stanford.edu
[3] https://www.willowgarage.com
[4] http://wiki.ros.org/Parameter_Server



Obrázek 2.1: Základní koncept systému ROS, Zdroj: [11]

ROS je určen pro UNIX systémy. Plně podporovaný a testovaný je na systémech Ubuntu a Mac OS X, avšak komunita kolem ROS sdílí podporu i pro systémy Fedora, Gentoo, Arch Linux a další. Port pro Microsoft Windows je možný, ale není zatím implementován.

Wikistránka ROS uvádí pět hlavních cílů při spolupráci a sdílení ROS frameworku:

- ROS-nezávislé knihovny: upřednostňovaný model vývoje je psaní ROS-nezávislých knihoven s čistými funkčními rozhraními.
- Nezávislost na jazyku: ROS framework je jednoduché implementovat do jakéhokoli moderního programovacího jazyku. Zatím byl implementován do jazyků Python, C++, Lisp a existují experimentální knihovny v jazycích Java a Lua.
- Jednoduché testování: ROS má zabudován testovací framework nazvaný rostest, který zjednodušuje testování a zavádění oprav.
- Škálování: ROS je vhodný pro velké runtime systémy i velké vývojové procesy.
- "Lightweight": ROS je navržen tak, aby byl co nejtenčí. Kód napsaný pro ROS může být použit ostatními softwarovými frameworky pro roboty. Přímým důsledkem toho je, že ROS je jednoduše integrovatelný s ostatními frameworky: ROS byl integrován s OpenRAVE, Orocos a Player.

Balíčky pro ROS

V této části seznámíme čtenáře s balíčky pro ROS, které umožňují komunikaci mezi zařízeními. Zajímají nás především ty, které se dají použít pro komunikaci s mobilním zařízením. Z existujících řešení jsou volně dostupné zejména tyto: RosJava, Rospy on android, Rosbridge suite s knihovnou roslibjs. Poté zmíníme balíček Asctec Mav Framework, který slouží pro práci se zařízeními firmy Ascending Technologies.

RosJava

Rosjava byla představena na konferenci RosCon2012[7] a od té doby se stává nejrozšířenějším nástrojem pro spojení robotiky a platformy Android. Rosjava je implementace ROS v jazyce Java, která si klade za hlavní cíl přinést ROS na Android. Nejdůležitějším balíčkem celého projektu je rosjava_core, který obsahuje implementaci roscore a android_core obsahující sadu komponent a užitečných příkladů pro vyvíjení ROS aplikace pro Android. Mobilní zařízení se poté chová transparentně jako ROS zařízení.

RosJava umožňuje nahrát binární soubory pro propojení projektů bez potřeby kompilace v ROS prostředí, což ocení především uživatelé s malými či žádnými zkušenostmi s ROS. Mohou také využít maven repozitáře.

RosJava používá několik kompilačních nástrojů:

- Java - openjdk-6
- Gradle
- Maven
- Catkin

Rospy on android

Rospy on android[5] je projekt týmu Distributed Multimodal Information Processing Group of Technische Universität München (TUM). Jedná se o port Rospy[6] na zařízení s operačním systémem Android a je založen na Python on Android. Na zařízení s OS Android běží naportovaný roscore napsaný v jazyce Python. Prozatím jsou implementovány základní balíčky Rospy, roslib, std_msgs, sensor_msgs, geometry_msgs a cv_bridge. Je možné automaticky nakonfigurovat roscore na Android zařízení pouhým naskenováním vygenerovaného QR kódu. Implementována je i základní podpora pro OpenCV a ROS image topicy.

Tento projekt byl naposled aktualizován 25.2. 2011, ale pořád se jedná o funkční řešení.

AscTec Mav Framework

Tento framework slouží k získávání dat a ovládání pozice kvadrokoptér od firmy Ascending Technologies. Obsahuje řadiče, nelineární ovladač pozice a další nástroje pro kvadrokoptéry vybavené deskou AutoPilot. Tento framework je založen na programovaní high level procesoru desky AutoPilot na rozdíl od knihovny asctec_drivers, která komunikuje přímo s low level procesorem této desky. Obsahuje tři základní balíčky:

- asctec_hl_comm - obsahuje definice zpráv frameworku
- asctec_hl_interface - rozhraní ROS pro HLP, skripty pro monitorování a debugování
- asctec_hl_firmware - firmware, který je potřeba nahrát do HLP (provést flash), aby framework fungoval

Zmíněný balíček asctec_hl_interface obsahuje uzel hl_interface, který nabízí všechny topicy a nastavení důležité pro tuto práci. Nejdůležitějším je topic /fcu/control, který přijímá příkazy k ovládání kvadrokoptéry. Topic /fcu/status obsahuje stavové informace důležité pro pilota (např. nabití baterie). Tento uzel má také konfigurovatelné statické nastavení. Mezi nimi je třeba zmínit parametr /postion_control, který nabízí výběr mezi třemi módy ovládání kvadrokoptéry. Při nastavení "off"se let ovládá zprávami pro zrychlení v jednotlivých osách a sílu motorů, jednotkou jsou radiány. Pro osu kolmou k zemi (yaw) jsou jednotkou radiány za sekundu. Nastavení "GPS"využívá stabilizační algoritmy LLP[7]

[5] https://projects.vmi.ei.tum.de/ros/wiki/Androidinstall
[6] Klientská knihovna pro ROS napsaná v jazyce Python http://wiki.ros.org/rospy
[7] Low Level Processor desky Autopilot

kvadrokoptéry a ovládá se zprávami rychlosti v jednotlivých směrech včetně stoupání i klesání. Jednotkou jsou metry za sekundu s vyjímkou osy kolmé k zemi, tam jsou opět použity radiány za sekundu. Třetí a poslední mód při nastavení "HighLevel"používá ke stabulizaci HLP[8], a tedy mohou být do stabilizace zapojeny další senzory a data vypočtené na Atomboardu, a bez externích algoritmů nefunguje. Tento mód využívá zprávy typu position a lze nastavit rychlost průletu jednotlivými body.

2.2 Android

Pro vzdálené ovládání robotů lze použít celou škálu metod a zařízení. Zadání však klade za cíl vytvořit ovládání na mobilním zařízení, které je dostupné a rozšířené. Dnešní chytré telefony jsou vybaveny operačním systémem a dostatečným výpočetním výkonem pro zvládnutí této problematiky. Následující text seznamuje čtenáře s operačním systémem Android, pro který bude vytvořena aplikace řídící let kvadrokoptéry.

Android je operační systém založený na Linuxovém jádře, určený především pro mobilní zařízení, jako například mobilní telefony či tablety. Na počátku byl vyvíjen firmou Android, Inc., kterou později koupila společnost Google. První zařízení používající operační systém Android bylo uvedeno do prodeje v říjnu roku 2008 firmou HTC a neslo název HTC Dream.

Zdrojové kódy systému Android jsou dostupné pod Apache License. Nejen výrobci zařízení tímto získávají možnost volně modifikovat a distribuovat tento systém podle potřeby, a proto je většina smartphonů a tabletů dodávana s kombinací open-source a proprietárního software. Velkou výhodou pro vývojáře je taky možnost zdarma umístit svou aplikaci na internetový obchod Google Play (dříve Android Market). Platforma Android je také uživatelsky nejrozšířenějším operačním systémem pro mobilní zařízení s podílem 71% v období duben-květen 2013.[3]

Vývoj pro Android

Existuje mnoho nástrojů a způsobů pro vývoj aplikací na mobilní platformu Android. Nejčastější je programování v jazyce Java a použití oficiálního nástroje Android Software Development Tools. Společnost Google však nabízí další dva oficiální přístupy. Druhou oficiální cestou je překlad knihoven `C/C++` a dalších jazyků do strojového kódu a jejich následná instalace pomocí nástroje Native Development Kit. Podporovány jsou architektury ARM, MIPS a x86. Třetí možností je využití Android Open Accessory Development Kit a protokolu Android Open Accessory, jež umožňuje spolupracovat s externím USB zařízením a se zařízeními s OS Android ve speciálním módu "accessory".

K dispozici jsou i vývojové prostředky třetích stran. App Inventor for Android byl původním projektem společnosti Google, ta ale v druhé polovině roku 2011 zveřejnila zdrojové kódy, ukončila webovou podporu a poskytla financování The MIT Center for Mobile Learning a nyní je projekt čistě v držení MIT. Další variantou je využití projektu Qt for Android, který umožňuje spouštět aplikace vytvořené v QT 5. Jsou podporovány všechny moduly krom Qt WebKit, Qt Bluetooth, Qt NFC, Qt Positioning, Qt Serial Port a modulů závislých na konkrétní platformě. Oficiální podpora systému Android byla vydána až 12. prosince 2013 s verzí Qt 5.2[8], do té doby bylo nejčastějším řešením použití Necessitas - portu Qt na Android. Toto však vyžadovalo mít Necessitas nainstalovaný v zařízení s OS Android, anebo jeho přiložení k aplikci, což však znamená značné zvětšení objemu aplikace.

[8] High Level Processor desky Autopilot

Jako další nástroje třetích stran můžeme zmínit například komerční The Simple Project a Basic4android inspirované programovacím jazykem Microsoft Visual Basic nebo SDL, který umožnil vzniknout například portu hry Jagged Alliance 2 na OS Android.

Co se týče vývojových prostředí, nabízí se dvě oficiální varianty. Stabilní ADT plugin[9] pro prostředí Eclipse a Android Studio, které je však ještě ve vývoji a dostupné pouze ve verzi 0.4.6 "early access preview"(březen 2014). Na webových stránkách ROS.org se doporučuje instalace prostředí Android Studio.[10]

2.3 AscTec Pelican - Tyra

Tato sekce seznámí čtenáře s konkrétní kvadrokoptérou, se kterou budou prováděny veškeré experimenty. Je proto důležité nastudovat tuto platformu a veškeré její možnosti. Nejedná se pouze o dodaný hardware, ale taky o software obsahující algoritmy pro stabilizaci a let stroje, který je možné v řešení vyžít.

Kvadrokoptéra firmy Ascending Technologies je zaměřena na co možná největší modularitu, tedy tak, aby na ni uživatel mohl dle libosti přidávat potřebné senzory a další zařízení. Kvůli nosnosti je navržena pro vysoký výkon a zároveň je na ni dostatek místa pro již zmíněné senzory. Lehká věž ve středu kvadrokoptéry umožňuje jednoduše přidávat a spravovat veškerou elektroniku na kvadrokoptéře. Stroj také může nést desku AutoPilot a jeden ze dvou výkonných x86 PC nabízených firmou Ascending Technologies. Mezi další znaky patří:

- Minimální váha: 630g
- Maximální nosnost: 650g
- Maximální doba letu s plnou zátěží: 15 min
- Programovatelné přes AscTec SDK and AscTec Simulink toolkit
- Přizpůsobitelná věž kvadrokoptéry
- Může nést výkonný PC (např. AscTec Mastermind nebo AscTec Atomboard)

Tyra je pracovní název konkrétního stroje vlastněného FIT VUT[11]. Kvadrokoptéra nese dvě desky, a to desku AutoPilot firmy Ascending Technologies a nízkoodběrovou desku Atomboard. Dále je Tyra osazena řadou senzorů, deska AutoPilot nese tyto: gyroskop, akcelerometr, barometr, kompas, GPS modul. K Atomboardu je pak připojená kamera a WLAN modul.

AutoPilot

Hlavním cílem desky AutoPilot je udržet kvadrokoptéru stabilní a usnadnit její ovládání, a tím usnadnit řízení i pilotům bez větších zkušeností. Deska je osazena dvěma ARM7 procesory Low Level Procesor (dále LLP) a High Level procesor (dále HLP). LLP zajišťuje

[9] Android Development Tools https://developer.android.com/tools/sdk/eclipse-adt.html

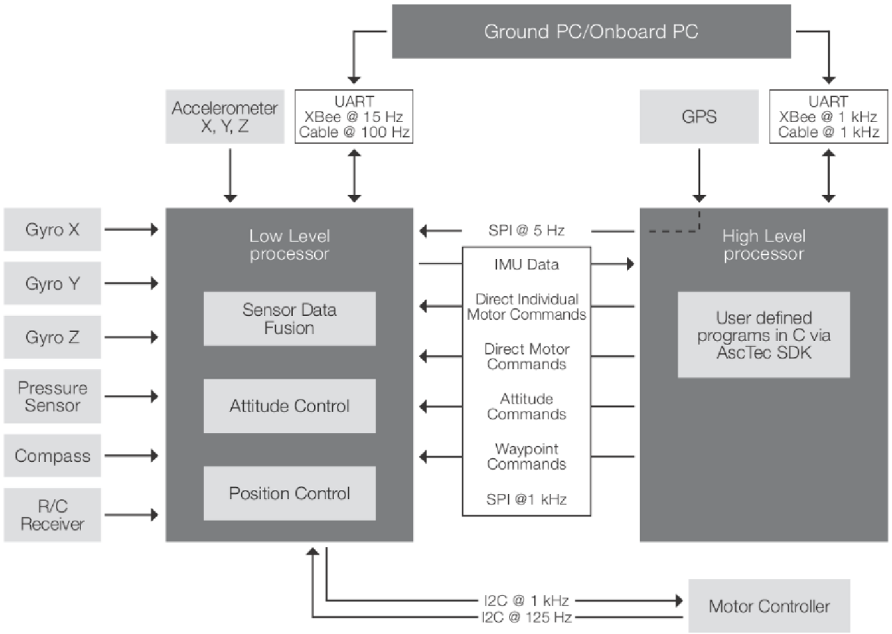
[10] http://wiki.ros.org/android/Tutorials/hydro/Installation%20-%20Ros%20Development%20Environment

[11] Fakulta Informačních Technologií Vysokého Učení Technického.

Obrázek 2.2: Kvadrokoptéra Tyra, Zdroj: [12]

získávání dat ze senzorů a stabilizaci kvadrokoptéry, zatímco HLP nabízí jednoduchou programovatelnost pro uživatelské řídicí algoritmy. Řadou senzorů a zpracováním jejich dat zařizuje funkci IMU (Inertial Measurement Unit).

Je více způsobů, jak ovládat kvadrokoptéru s využitím technologie AutoPilot. Kvadrokoptéra může přijímat data z ovladače a sériové data z PC umístěného na zemi, nebo může být program ovládání nahrán do HLP pro automatizované létání.



Obrázek 2.3: schéma desky AutoPilot, Zdroj: [2]

Low Level procesor

Jak již je zmíněno výše, LLP zajišťuje zpracování dat ze senzorů. Dalším důležitým účelem je zajištění rychlé a stabilní kontroly nad přístrojem s obnovovací frekvencí 1000 Hz. Existují tři druhy letových módů, které lze přepínat dálkovým ovladačem kvadrokoptéry:

- GPS mód (aktivování kontroly pohybu, výšky a pozice)
- Výškový mód (aktivování kontroly pohybu a výšky)
- Manuální mód (aktivování kontroly pohybu)

Všechny kvadrokoptéry AscTec mají nouzový režim, který se automaticky aktivuje při ztrátě signálu mezi strojem a dálkovým ovladačem. Je k dispozici více strategií nouzového režimu.

Data ze senzorů mohou být z LLP získána pomocí sériového rozhraní, dále je možno tímto rozhraním zasílat příkazy pohybu (úhel stoupání, úhel otočení, rychlost stáčení a sílu motoru), nebo body, kterými má proletět. Bezdrátová sériová komunikace Xbee umožňuje obnovovací frekvenci do 15 Hz, připojením sériovým kablem lze dosáhnout až 100 Hz.

High Level procesor

Při programování HLP je výhodou, že funguje jako jakási záchranná brzda při testování svých řídicích algoritmů, protože může pilot kdykoli na ovladači přepnout do bezpečného prostředí ověřených algoritmů na LLP. Programování mikroprocesoru se provádí přes oficiální AscTec SDK v jazyce C. Poté se už jenom provede flash HLP a kvadrokoptéra může začít používat uživatelské řídicí algoritmy.

Oba mikroprocesory komunikují na vysoké frekvenci 1000 Hz, takže všechna data ze senzorů jsou dostupná i pro HLP. Na stejné frekvenci mohou být příkazy samozřejmě posílány i zpět do LLP, například příkazy rychlosti rotace pro jednotlivé motory, příkazy pohybu (stoupání, otočení atd.) nebo jednotlivé body průletu.

HLP nabízí více rozhraní (UART, SPI, I2C, simple I/O) jak přidávat další uživatelské senzory, jako například kameru, či dále zmíněný Atomboard. Napájení dalších zařízení může být prováděno také přímo z desky AutoPilot buď 5V nebo 12 V.

Atomboard

Smyslem desky Atomboard je poskytnout kvadrokoptéře dostatečný výkon a možnost programování obecných algoritmů na architektuře x86. Tato deska umožňuje mít nainstalován plnohodnotný operační systém se systémem ROS a připojení dalších periferií. Toho se může využít ke komunikaci s jinými zařízeními, jako například s dalšími roboty, počítači či mobilními zařízeními, na kterých poběží systém ROS.

Na Tyře není použit Atomboard od firmy Ascending Technologies, ale zejména kvůli ceně byla pořízena levnější deska MIO-2261 značky Advantech. Protože jde o obecný počítač, neztratila se téměř žádná přidaná hodnota. AscTec Atomboard má pouze navíc speciálně navržený chladič kvůli snížení hmotnosti a dodává se s předinstalovaným systémem Ubuntu. Formát desky Pico-ITX umožňuje její umístění na kvadrokoptéru stejně tak dobře jako desku AscTec. Je osazena dvoujádrovým procesorem Intel Atom, jehož průměrná spotřeba je udávaná 5,52W a maximální 9,6W, čímž spadá do kategorie nízkopříkonových. Je vybavena 4GB DDR3 pamětí frekvence 1333MHz a 30GB SSD mSata diskem. Toto vybavení řadí kvadrokoptéru Tyra mezi komerčně dostupné létající stroje s nejvyšším výpočetním výkonem.

Deska dále obsahuje mnoho rozhraní. USB se používá pro spojení s deskou AutoPilot pomocí USB UART převodníku. Dále pak pro připojení USB kamery. Pro spojení s dalšími roboty se bude používat Wi-Fi modul. Mezi další rozhraní patří například VGA výstup, Audio vstup i výstup, Ethernet, I2C, či Mini PCI Express.

2.4 Gazebo

Ačkoli tato práce směřuje k provádění experimentů navrženého systému na fyzickém zařízení, je nezbytné provést základní testy na simulátoru. Použití simulátoru nebrání jen případné havarii kvadrokoptéry z důvodů chybně implementovaných algoritmů, ale umožňuje provádění testů bez závislosti na nabití baterií. Následující text představuje nejrozšířenější simulátor robotů a robotických zařízení Gazebo, který byl používán při počáteční fázi této práce.

Gazebo je simulátor nejen robotů jako celku, ale také jednotlivých částí robotů, či skupin robotů. Simulátor pracuje v trojrozměrném prostředí, generuje realistické odezvy senzorů a věrohodné interakce mezi objekty. Je v něm implementována přesná fyzika tuhých těles. V roce 2009 John Hsu integroval `ROS` a `PR2`[12] do Gazeba, čímž z něj udělal přední nástroj pro simulace používané `ROS` komunitou. Pro tuto bakalářskou práci byl využíván zejména balíček `Hector Quadrotor` jakožto simulátor UAV[13] a probíhaly na něm testy dříve, než byly prováděny na kvadrokoptéře Tyra.

Hector Quadrotor

Ještě do března roku 2012 neexistovalo dostupné obecné řešení pro simulace kvadrokoptér za použití nástrojů `ROS`. `Hector Quadrotor` je soubor balíčků pro `ROS` spojený s modelováním, ovládáním a simulováním kvadrokoptér. Simulace probíhá v prostředí `Gazebo` stejně jako u pozemních robotů, čili je možné snímat data ze senzorů a testovat všechny aspekty řízení. Testováním v simulátoru se vyhýbá nepříjemnostem, jako je například pád kvadrokoptéry špatně odladěnými stabilizačními algoritmy.

`Hector Quadrotor` se skládá ze šesti balíčků, kde se každý stará o jiný aspekt simulace:

- `hector_quadrotor_description` poskytuje obecný URDF model a jeho varianty s různými senzory.
- `hector_quadrotor_model` implementuje dynamické modely pro pohony a aerodynamiku kvadrokoptéry
- `hector_quadrotor_controller` obsahuje ovládání otočení, pozice a motorů pro kvadrokoptéru založené na `ros_control`, `hector_quadrotor_controller_gazebo` implementuje hardwarové rozhraní kvadrokoptéry jako plugin pro `gazebo_ros_control`
- `hector_quadrotor_gazebo` obsahuje nezbytné spouštěcí soubory a informace o závislostech pro simulaci kvadrokoptéry v `Gazebu`
- `hector_quadrotor_teleop` obsahuje uzel umožňující ovládat kvadrokoptéru gamepadem
- `hector_quadrotor_gazebo_plugins` poskytuje pluginy specifické pro simulaci kvadrokoptéry v `Gazebu`

[12] Robot vyvíjený ve Willow Garage https://www.willowgarage.com/pages/pr2/overview
[13] Unmanned Aerial Vehicle - bezpilotní letoun

2.5 Existující řešení

Cílem této části je prozkoumat existující řešení z hlediska ovládání kvadrokoptér mobilním zařízením. Důležitým aspektem je najít nejen proprietární řešení, ale také zjistit, zda existují nějaké open-source alternativy a zda nějaká platforma využívá `ROS`. Zde jsou uvedeny obecné informace o aktuální situaci a přehled hotových řešení.

Byly prozkoumány mnohé dostupné kvadrokoptéry, avšak pouze málo z nich umožňuje ovládání mobilním zařízením. Vlastní aplikaci pro Android a iOS nabízí pouze firmy Parrot ke svému AR.Drone a firma Walkera ke třem ze svých modelů.

Parrot AR.Drone

Mezi nejrozšířenější modely nejen mezi nadšenci kvadrokoptér patří bezpochyby AR.Drone od firmy Parrot. Tato kvadrokoptéra byla představena na CES 2010 v Las Vegas jako AR.Drone 1.0. Předváděno bylo ovládání iOS pomocí aplikace AR.Freeflight a aplikace AR.Race, což je závodní hra navržena pro AR.Drone, která měří čas průletu uživatelem určenou trasou a má i možnost multiplayeru s dalšími AR.Drony. Kvadrokoptéra se neomezuje pouze na iOS, ale lze ji také ovládat zařízením s operačním systémem Android. V roce 2012 přichází firma Parrot s následovníkem AR.Drone 2.0.

AR.Drone 1.0

Tato kvadrokoptéra nese počítač s operačním systémem Linux a komunikuje s pilotem skrze Wi-Fi hotspot. Na palubě je umístěna řada senzorů, které umožňují stabilizaci v prostoru, patří mezi ně i ultrazvukový výškoměr, který umožňuje přesnou vertikální stabilizaci do výšky 6 metrů. Dále má AR.Drone VGA kameru a spolu se softwarem od firmy Parrot umí sestavovat 3D mapy okolí, sledovat objekty a další AR.Drony nebo hrát hry s použitím rozšířené reality. Rotory jsou napájeny 15 watty 11,1 Voltovou Li-Pol baterií, což umožňuje dobu letu přibližně 12 minut rychlostí 5 m/s. Další zařízení lze připojit pomocí rozhraní USB 2.0.

Kvadrokoptéra měří 57 cm a její tělo je vyrobeno z nylonových a karbonových vláken. Dodává se se dvěma kryty pro vnitřní a venkovní použití. Vnitřní je vyrobena z pěny a zajišťuje větší ochranu vrtulí na rozdíl od venkovní, která je vyrobena z plastu a umožňuje větší manévrovací schopnosti.

AR.Drone 2.0

Druhá verze AR.Drone zaznamenala spoustu vylepšení. Senzory zajišťující stabilizaci kvadrokoptéry za letu byly vyměněny za citlivější tak, aby umožňovaly pilotu větší kontrolu letu. Gyroskop akcelerometr a magnetometr byly nahrazeny jejich tříosými variantami a k ultrazvukovému výškoměru přibyl senzor měřící atmosférický tlak. Namísto VGA kamery je na stroji HD kamera zaznamenávající obraz v rozlišení 720p s frekvencí 30fps. Přibyla také možnost zaznamenávat let přímo do paměti Flash připojené do USB rozhraní. V roce 2013 Parrot uvádí Flight Recorder for AR.Drone 2.0. Přidává kvadrokoptéře 4GB paměti a GPS modul. Toto umožňuje pilotovi zadat několik bodů, kterými má AR.Drone proletět.

Ovládání letu

Ovládání AR.Drone probíhá přes volně stažitelnou oficiální aplikaci AR.FreeFlight na Google Play. Po zapnutí aplikace ověří aktualizace (případně je stáhne), spojí se s kvadro-

koptérou a následně ukáže obraz z kvadrokoptéry, který překrývají ovládací prvky. Jedná se o dva virtuální joysticky a dvě stavové lišty umístěné na horní a dolní části obrazovky. Horní lišta obsahuje tyto ikony a tlačítka (zleva):

- Tlačítko zpět do aplikace (3)
- Ikona stavu nabití baterie (4)
- Tlačítko nastavení (5)
- Ikona síly wi-fi signálu (6)
- Tlačítko nouzového vypnutí (7)
- Tlačítko přepínání kamer (8)
- Tlačítko nahrávání videa (9)
- Tlačítko pořízení fotografie (10)

Spodní lišta:

- Výška od země či jiného objektu pod kvadrokoptérou (11)
- Tlačítko vzlétnutí (1)
- Rychlost kvadrokoptéry (12)

Největší část obrazovky zaujímá dvojice výše zmíněných virtuálních joysticků. Mohou pracovat třemi různými způsoby. První možností je ovládání stejné jako na běžném RC ovladači, tzn. pravým joystickem se ovládá výška a otočení v ose kolmé k zemi, levým pak pohyb vpřed, vzad a do stran (joypad mode on). Druhá možnost je přepnout levý joystick do módu ovládaném akcelerometry (joypad mode off). V tomto případě levý virtuální joystick funguje pouze jako tlačítko a směry letu se ovládají nakláněním mobilního zařízení. Třetí možností je mód Absolute control, při jehož aktivaci se stane referenčním bodem k ovládání mobilní zařízení bez ohledu na to, kterým směrem je natočena přední část kvadrokoptéry (tzn. kam se dívá kamera).

Obrázek 2.4: Ovládání AR.Drone 2.0, Zdroj: [9]

Ovládání kamery

8. ledna 2013 představila firma Parrot na CES[14] placené rozšíření AR.Freeflight nazvané Director Mode, které umožňuje efektní natáčení prostředí kolem sebe. Ovládání používá čtyři módy ovládání, a to:

- Traveling (cestování) - fixuje kvadrokoptéru v dané výšce a lze se pohybovat pouze čtyřmi směry z pohledu kamery (dopředu, dozadu, vpravo, vlevo)
- Panorama - umožňuje pouze otáčení vpravo a vlevo kolem vlastní osy
- Crane (jeřáb) - umožňuje pouze pohyb nahoru a dolů
- Steady mode (stabilní mód) - zakáže kvadrokoptéře jakýkoli pohyb

Tlačítka ovládání zabírají téměř polovinu displeje a jejich poloprůhlednost sice zlepšuje pohled na zkoumanou scénu, avšak na světlejších površích nejsou dobře vidět. Průhlednost tlačítek je spolu s joysticky nastavitelná v menu mimo letovou obrazovku.

Obrázek 2.5: Ovládání AR.Drone 2.0 s Director Mode, Zdroj: [10]

Walkera UAVs

Firma Walkera je čínská firma vyrábějící dálkově ovládané modely helikoptér, ale v posledních letech rozšířila svou působnost o výrobu malých kvadrokoptér v návaznosti na předchozí úspěch s helikoptérami. Firma především prodává modely pro civilní použití, ale měla také zakázky pro čínskou armádu na dodej levných modelů, jež sloužily k tréninku před použitím drahých zařízení, které jsou pak nasazovány ve vojenských misích. Všechny stroje jsou ovladatelné dodaným dvoujoystickovým ovladačem a nesou kameru a pouze některé modely jsou opatřeny Wi-Fi modulem pro ovládání mobilní aplikací.

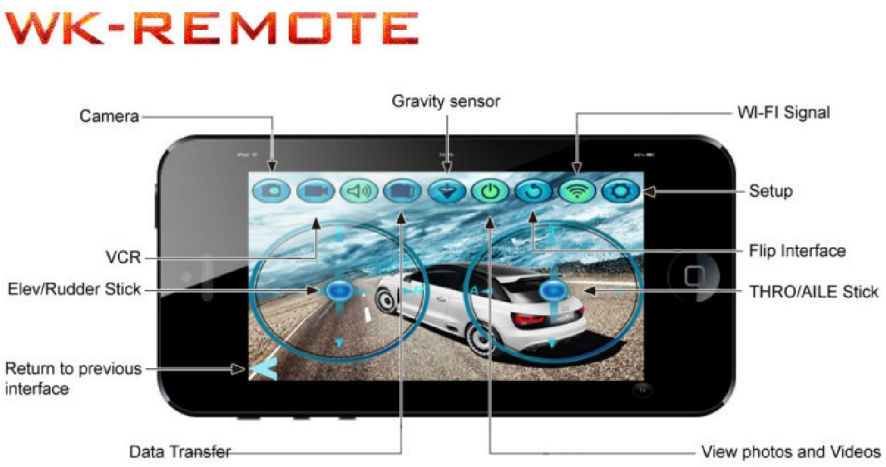
Modely umožňující ovládání mobilním zařízením:

- Brushed Hotten X wifi
- QR W100(S)

[14] Consumer Electronics Show

- QR X350

Na rozdíl do konkurenčního AR.Drone, který je určen pouze k ovládání chytrým telefonem či tabletem, nenabízí Walkera krom základní aplikace k ovládání žádný další software a rozšířenou funkčnost. Základní aplikace WK-remote dostupná pro iOS a Android ovládá kvadrokoptéru virtuálními joysticky anebo nakláněním zařízení (tlačítko gravity sensor). V nastavení jsou čtyři módy ovládání, které ale znamenají pouze přehození páček a jejich orientaci.



Obrázek 2.6: Aplikace WK-remote pro ovládání kvadrokoptér Walkera, Zdroj: [5]

Kapitola 3

Návrh řešení

Následující kapitola představuje nejrozšířenější existující řešení problému ovládání kvadrokoptéry mobilním zařízením. Je popsán cíl řešení a jsou zde rozebrány jednotlivé části z pohledu ovládání robota. Tato část dále obsahuje schématické znázornění bloků na straně klienta (mobilního zařízení se systémem Android) a robota (Atomboard s `ROS`).

3.1 Popis cíle a analýza řešení

Cílem je navrhnout řešení, které by umožňovalo řízení kvadrokoptéry pomocí mobilního zařízení a sledování scény. Tento návrh vychází z existujících řešení popsaných výše a snaží se docílit modifikace, která by více vyhovovala pilotům kvadrokoptér při zkoumání scény nesenou kamerou. Návrh řešení se nezaměřuje na rychlé akrobatické manévry, ale na jednoduché plynulé ovládání, kde důležitou roli hraje kvalita záběru kamery kvadrokoptéry.

Pilot potřebuje mít kvadrokoptéru stále pod kontrolou, a to i za předpokladu, že není pilotem viditelná (například se nachází za překážkou). Řešení je určeno i pro piloty, kteří nemají žádné předchozí zkušenosti s ovládáním kvadrokoptér klasickými rádiovými ovladači či jinými způsoby, tudíž by mělo být ovládání velmi jednoduché. Pilot by měl zvládnout základní manévry bez jakékoli instruktáže školitele či zkušenějšího pilota. Důležitým aspektem, proč pilot toto řešení potřebuje, je sledování určeného objektu či bodu, proto by měla co největší část displeje patřit čistě obrazu, bez ovládacích a dalších prvků. Obraz by také neměl být zakrýván prsty pilota neustálým držením displeje a pro pohodlné sledování by neměl být pilot nucen držet mobilní zařízení v určité poloze. Nahýbání by mohlo vést k odrážení odlesků slunce, či jiných světel, a sledování by tak bylo rušeno nežádoucím osvěcováním displeje nebo oslňováním pilota. Pilot by měl mít kvadrokoptéru stále pod kontrolou a řešení by mělo být dostatečně intuitivní a používat zaběhnuté stereotypy, aby ovládání nevedlo k chybným manévrům, či dokonce k havárii.

Shrnutí požadavků pilota:

- Žádné vstupní znalosti pilotování pro základní manévry
- Ovládání kvadrokoptéry mimo viditelnost pilota
- Co největší plocha obrazu z kamery
- Pohodlí při sledování (nahýbání neovládá kvadrokoptéru)

3.2 Prvky uživatelského rozhraní

Návržené GUI ovládání kvadrokoptéry musí zobrazovat obraz z kamery po celou dobu letu, tímto se zajistí, že pilot bude moct ovládat kvadrokoptéru i mimo jeho zorné pole. Dalším aspektem je zajistit co nejkomfortnější sledování objektu. Proto bude obraz z kamery zabírat celý displej. Sledování by také nemělo být rušeno žádnými dalšími ovládácími prvky. Tohoto záměru se docílí odlišným způsobem ovládání, než jsou virtuální joysticky u existujících řešení (viz kapitola 2.5), které zabírají velkou část displeje. Moderní dotykové displeje umožňují dostatečnou kontrolu ovládání různými tahy po obrazovce a gesty. Lze případně využít i vícedotykové varianty gest, které se běžně používají v celé řadě aplikací. Použitím gest zajistíme pilotovi nejčistší obraz, jakého lze docílit, a vyhneme se i ovládání nahýbáním mobilního zařízení, což byl také jeden z požadavků na aplikaci.

Po spuštění aplikace bude zobrazena jednoduchá obrazovka pro spárování robota s mobilním zařízením. V systému `ROS` půjde o zadání URI adresy robota a portu, na kterém běží `roscore`. Na párovací obrazovce půjde taky zvolit možnost načtení QR kódu. Po úspěšném spárování přejde aplikace na hlavní obrazovku, na níž by pilot měl vidět obraz z kamery kvadrokoptéry, ovládací tlačítka a stavovou lištu (viz obrázek 3.1).

Dominantou by měla být tři hlavní tlačítka přepínající módy ovládání, a to:

- Mód ovládání kamery
- Mód ovládání letu kvadrokoptéry
- Mód sledování scény

Řídicí úkony budou pořád stejné, avšak při rozdílném aktivovaném módu se kvadrokoptéra zachová jinak. Kvadrokoptéra se bude ovládat gesty kreslenými na plochu obrazovky, kam je promítán obraz z kamery. Dostupná gesta budou tato:

1. rovný tah prstem po displeji (nahoru, dolů, doprava, doleva)
2. nakreslení kružnice (v obou směrech)
3. nakreslení stříšky (nahoru, dolů)

Popis chování v jednotlivých módech udává následující přehled:

`Mód kamery`
1. natočení kamery nahoru, dolů, doprava, doleva
2. kvadrokoptéra se točí kolem vlastní osy
3. zoom (ostření)

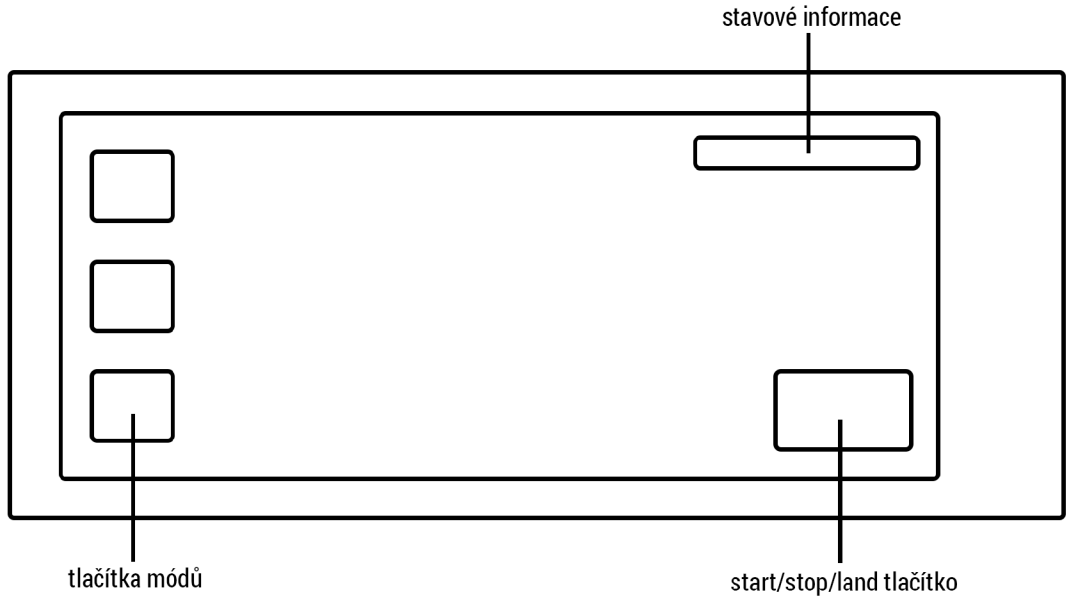
`Mód letu`
1. let kvadrokoptéry při stabilní výšce dopředu, dozadu, doprava, doleva
2. kvadrokoptéra se točí kolem vlastní osy
3. let nahoru dolů

`Mód sledování objektu`
1. let kvadrokoptéry při stabilní výšce dopředu, dozadu, doprava, doleva - rozdíl oproti módu letu v rychlosti letu a délce provádění manévru.

2. Oblet objektu po kružnici ve stabilní výšce čelem k objektu. Poloměr je nastavitelný v menu.
3. let nahoru a dolů - stejné s módem letu.

Rozdílný mód má jinou strategii letu. Zatímco mód letu, který je určen k přeletu kvadrokoptéry, bude provádět svižnější manévry v krátké době, mód sledování objektu bude kvadrokoptéru ovládat velmi pomalu po delší dobu, aby mohl pilot pohodlně přesně sledovat objekt. Toto řešení dává aplikaci jistou autonomii při letu, proto po dvojím kliku kdekoli na displeji zastaví veškerý pohyb kvadrokoptéry a bude ji držet na místě ve vzduchu. Dále nebude chybět tlačítko start/land, kterým se bude kvadrokoptéra startovat a také provádět přistávací manévr na zem.

Dále bude v horní části obrazovky úzký panel informující o stavu kvadrokoptéry, konkrétněji o nabití baterie a GPS signálu.

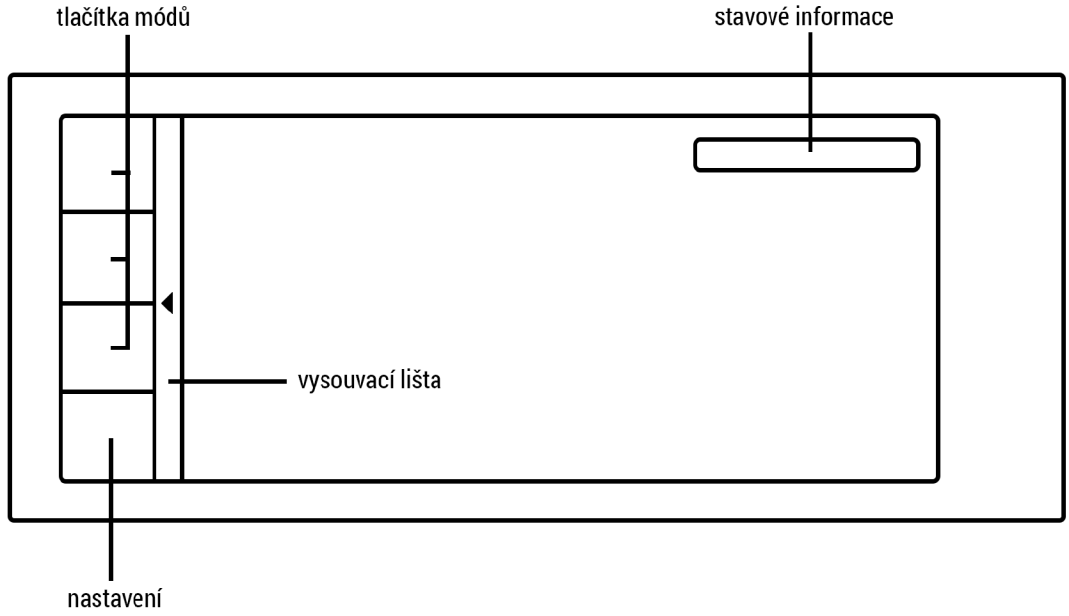


Obrázek 3.1: základní návrh UI

Další návrh seskupuje tlačítka do vysouvací lišty, která je na zařízeních Android běžná, například pro nastavení a notifikace (viz obrázek 3.2). Tímto se docílí ještě větší plochy obrazu oproti předešlému návrhu. Tento návrh počítá již s tlačítkem nastavení. Po jeho zmáčknutí se zobrazí formulář pro konfiguraci rychlostí jednotlivých módů, otáčení i obletu.

3.3 Koncept navrženého řešení

Celý systém pilotování kvadrokoptéry musí vyhovovat nejen kritériím pilota pro kvalitu GUI, ale musí také zajišťovat, aby nedošlo k havárii vlivem selhání systému. Proto musí být dostatečně robustní a minimalizovat možnosti chyb každé části. Je třeba vyřešit stabilitu kvadrokoptéry, ať není pilot zatížen vyvažováním při stání na místě či dokonce ovládáním jednotlivých motorů. Dále je nezbytné zajistit připojení kamery a připojit Wi-Fi modul s odpovídající anténou. Wi-Fi síť bude zajišťovat přenos dat mezi mobilním zařízením a kvadrokoptérou.



Obrázek 3.2: další návrh UI pro zvětšení polohy pozorování

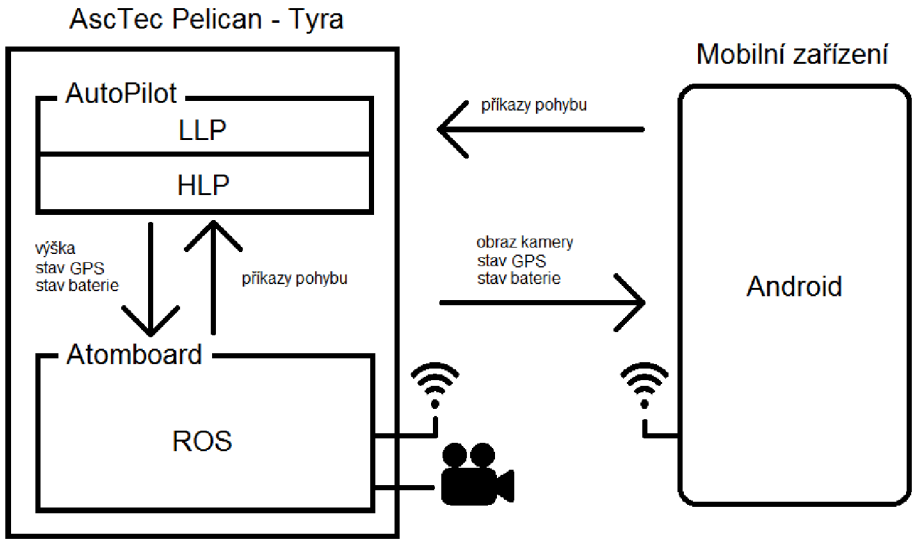
Navržený systém se skládá ze tří částí, kvadrokoptéra ponese dvě části a třetí část se bude nacházet v mobilním zařízení. První částí je systém zajšťující nízkoúrovňovou stabilitu letu a zisk dat ze všech namontovaných senzorů. Tato část se bude nacházet na desce AutoPilot. Druhá část bude umístěna na desce Atomboard s UNIXovým operačním systémem a bude zde nainstalován ROS. Jejím úkolem bude získávat a zpracovávat data z části AutoPilot a poskytovat API vyšší úrovně pro ovládání mobilním zařízením. Třetí část systému bude na mobilním zařízení a bude obsahovat především GUI pro komunikaci s uživatelem a bude ovládat kvadrokoptéru pomocí API poskytnuté části na desce Atomboard.

Přenášená data

Přenos dat bude probíhat na dvou rozhraních. Mezi deskami AutoPilot a Atomboard pomocí USB UART převodníku a mezi deskou Atomboard a mobilním zařízením pomocí Wi-Fi. Na desce Atomboard může běžet Wi-Fi AP, ale může se využít i silnější externí Wi-Fi sítě pro zvětšení dosahu signálu. Z desky AutoPilot budou přenášena především data ze senzorů (výška, informace o nabití baterie) a zároveň bude tato deska přijímat informace, které určují pohyb kvadrokoptéry. Atomboard je prostředníkem mezi ovládáním uživatelem skrze mobilní zařízení a deskou AutoPilot, která přímo ovládá rotory kvadrokoptéry. Směrem do mobilního zařízení se budou přenášet data z kamery a stavové informace o síle baterie. Na Atomboard budou přenášena a zpracovávána řídicí data pohybu kvadrokoptéry.

Spojení mobilního zařízení s kvadrokoptérou

Pro zajištění spojení mobilního zařízení se systémem `ROS`, běžícím na desce Atomboard kvadrokoptéry, je nejlepší využít některý z nástavbových balíčků `ROS` pro Android. Ukázalo se, že Rospy on Android Mnichovské univerzity je funkční, avšak je velmi omezený a od roku 2011 neudržovaný, naproti tomu `RosJava` s knihovnami `rosjava_core` a `android_core` jsou velmi živými projekty aktualizovanými i několikrát měsíčně. Další výhodou je jednotná implementace v jazyce Java jak na mobilním zařízení, tak na straně Atomboardu. Rosjava



Obrázek 3.3: Schéma návrhu

je vydána a zdokumentována pro `ROS hydro`, poskytuje také předpřipravené tutoriály pro rychlé pochopení vývoje `ROS` aplikace na Android.

3.4 Návrh uživatelské aplikace

Zařízení se před začátkem letu musí spojit s kvadrokoptérou pomocí Wi-Fi. V systému `ROS` existuje rozhraní pro spojování robotů, využije se tedy hotového řešení se správným nastavením. Pilotem zadaná URI adresa musí být kontrolována z hlediska validity ještě před pokusem o spárování, aby nedocházelo v důsledku špatné interpretace k chybnému spojení, a dopředu by měla nabízet přednastavenou hodnotu správného formátu. Aplikace by měla zobrazovat po opětovném spuštění poslední použitou URI, protože pilot opakovaně vlastní nejčastěji jednu kvadrokoptéru a ta si při připojení k Wi-Fi ponechává původní IP adresu, a URI se tedy nemění.

Ovládání kvadrokoptéry mobilním zařízením je hlavním předmětem této bakalářské práce. Pilot zadává instrukce pomocí gest, která jsou popsána v předešlé podkapitole, a tímto řídí pohyb kvadrokoptéry. Mobilní zařízení tedy musí od uživatele číst gesta, zpracovávat je a zasílat zprávy na daný `ROS` topic ovládající směr letu kvadrokoptéry. Zpracování gest musí odpovídat nejen intuitivnosti ovládání, ale musí být také správně nastavena citlivost rozpoznání gest. Řešení by mělo být přesně takové, aby se pilot nemusel zbytečně často dotýkat displeje při opakování gest a zároveň by mělo být ovládání dostatečně jemné k vykonání základních manévrů. Pilot musí být schopen přesně ovládat manévr kvadrokoptéry zadáním konkrétního tahu, jiné chování je nepřípustné, protože by mohlo vést k havárii a poškození. Programově se musí také vyřešit zakázání překrývání gest, to znamená, že v momentě, kdy pilot zadá nové gesto, musí se okamžitě zastavit předchozí pohyb a provést

pohyb aktuální. Vytvářet fronty gest a tedy manévrů není předmětem tohoto řešení. Zadávání gest dále nesmí kolidovat s jinými ovládacími prvky GUI, v tomto případě vytažením vysouvací lišty. Pilot musí být také uvědoměn o prováděném pohybu, tedy provedeném gestu. K experimentování jsou navrženy dvě varianty, a to zobrazení ve stavovém řádku, či zanechání zvýrazněné stopy po táhnutí prstem při kreslení gesta.

Aby měl pilot přehled o tom, kam kvadrokoptéra letí a kterým směrem je natočena, je důležité přenášet na displej obraz z kamery kvadrokoptéry. Ten by měl zaujímat celou plochu pro co největší komfort sledování scény, což znamená, že na tento prvek musí být programově umožněno kreslit gesta k ovládání kvadrokoptéry. Problémem může být zpoždění přenosu obrazu z kvadrokoptéry, které bude způsobeno přenosem přes Wi-Fi a použitím systému `ROS`. Protože však není cílem řešení provádět rychlé manévry ani jiné akce, pro které by bylo zpoždění kritickou veličinou, postačí pro optimalizaci zpoždění nastavením komprimace obrazu z kamery a topicu `ROSu`.

Dále je potřeba vizualizovat stavové informace o nabití baterie a síle Wi-Fi signálu. Síla Wi-Fi signálu bude odpovídat spojení mobilního zařízení s Wi-Fi AP[1]. Bude-li spojení probíhat přes externí Wi-Fi router, nebude tento údaj zahrnovat spojení mezi kvadrokoptérou a daným routerem, což může být matoucí a nebezpečné. Pilot s tímto musí být obeznámen před začátkem létání. Nabití baterie se bude získávat z topicu `ROSu` s vhodně zvolenou obnovovací frekvencí, aby se předešlo zbytečnému zatížení přenosové linky.

Mobilní zařízení s OS Android by mělo být co nejvíce odstíněno od výpočtů polohy a podobných algoritmů a mělo by být implementováno pro co nejrychlejší odezvu GUI. Kvůli komunikaci s `ROS` na něm však musí běžet také instance `android_core`.

3.5 Systém nesený Tyrou

Řešení na kvadrokoptéře musí obsahovat veškeré prvky řízení letu kvadrokoptéry. Z rozboru zadání vyplývá využití systému `ROS`, takže se bude pro řízení letu kvadrokoptéry používat zápis na dané topicy. Tyto topicy je nutné zpřístupnit tak, aby se na ně mohlo zapisovat z mobilního zařízení.

Na kvadrokoptéře se nachází dvě části navrhnutého systému, každá na jiné desce. Část AutoPilot zůstane beze změny oproti dodanému stavu od výrobce a bude ponechán i originální software. Deska Atomboard neobsahuje od výrobce žádný software, tutíž bude nutné navrhnout a naimplementovat vhodné řešení. Dále je nutné připojit Wi-Fi modul, připojit USB kameru a propojit obě desky pomocí USB UART převodníku.

AutoPilot

K desce AutoPilot jsou připojeny veškeré senzory, které jsou nutné k letu kvadrokoptéry. Jejím hlavním úkolem tedy bude zajšťovat autonomní držení výšky, aby bylo ovládání naprosto odstíněno od nutnosti řízení rychlostí jednotlivých rotorů. Na desce AutoPilot bude nainstalován proprietární software firmy Ascending Technologies, který umožňuje tři módy řízení:

- Manual mode - přímé ovládání síly rotorů
- Height mode - kvadrokoptéra drží výšku

[1] Access Point

- GPS mode - kvadrokoptéra drží výšku a vyrovnává vlivy vychylující do stran (například vítr)

Manual mode je pro řešení nevhodný, neboť gesta nemají takovou rozlišovací schopnost, aby měl pilot pod kontrolou každý pohyb kvadrokoptéry. Je tedy nutné se spolehnout na Height nebo GPS mód. Stabilizace má větší přesnost použitím druhého ze dvou módů, ale při nedostupnosti GPS signálu bude možné alespoň volně létat za využití Height módu. Pro plnou funkčnost však bude využíván GPS mode.

Dále bude deska AutoPilot poskytovat informace, které jsou důležité zaprvé pro pilota (stav baterie, dostupnost GPS signálu). Tato data budou přenášena desce Atomboard, která je druhou částí řešení.

Atomboard

Deska Atomboard je osazena výkonným procesorem a má dostatek paměti k nainstalovaní plnohodnotného UNIXového operačního systému, na kterém bude běžet `ROS`. Dosáhne se tím vyšší abstrakce ovládání kvadrokoptéry a do budoucna snadno rozšířitelného systému o další funkce. Toto řešení umožní také jednoduše připojit a zprovoznit Wi-Fi modul a USB kameru.

Atomboard je prostředníkem mezi deskou AutoPilot a mobilním zařízením, proto systém ROS musí umožňovat komunikaci s oběma částmi. Pro komunikaci s AutoPilot se bude používat `asctec_mav_framework`, jehož firmware se musí nahrát do HLP a poté lze využít poskytované rozhraní. Tento framework poskytuje potřebné `ROS` topicy a zprávy pro výměnu dat mezi deskami, zejména topic `/fcu/control`, na který se budou zapisovat data zrychlení v jednotlivých osách. Tímto způsobem se bude programově řídit let kvadrokoptéry. Komunikaci s mobilním zařízením bude obstarávat `RosJava`, která zajistí, že se mobilní zařízení bude chovat transparentně jako běžný počítač s `ROS`. Nezbytné je taky zpřístupnit data z připojené kamery v prostředí ROS. K tomu bude sloužit oficiální balíček `ueye_cam`, který také umožní kameru správně nastavit.

Kapitola 4

Realizace

V této kapitole bude uveden postupem sestavení jednotlivých částí, instalací nezbytného softwaru a následně implementačními detaily celého systému. Budou zde rozebrány nástroje a jednotlivé komponenty použité při vývoji aplikace.

4.1 Sestavení a instalace systému

Podstatnou částí této práce bylo samotné sestavení a instalace celého systému. Následující sekce přiblíží jednotlivé kroky vedoucí od dodaných dílů až k úspěšnému funkčnímu řešení ovládání kvadrokoptéry mobilním zařízením za použití systému `ROS`. Problémy samotného zařízení Tyra a jejich řešení jsou popsány v kapitole 4.3.

Na kvadrokoptéře byla v dodaném stavu namontována pouze deska AutoPilot, musela se tedy přimontovat deska Atomboard. Oproti `AscTec Atomboard` měla vybraná deska vyšší chladič a v základním stavu se na stroj nevešla, bylo tedy nutné přemontovat držák kamery na nižší pozici, což konstrukce Tyry umožňuje. Napájení Atomboardu probíhá DC/DC převodníkem, který byl připevněn k jedné z noh kvadrokoptéry a byl připájen na konektor pro připojení baterie. Pro potřebu propojit Atomboard s HLP desky AutoPilot byl vybrán USB UART převodník. Připojení přes Xbee má výhodu bezdrátového propojení s dalšími počítači, avšak od tohoto řešení bylo upuštěno kvůli špatné spolehlivosti spojení. Navázat spojení se podařilo až po několika pokusech a často docházelo k výpadku paketů. Kamera byla připevněna k dodanému držáku se schopností stabilizace a byla připojena k Atomboardu pomocí USB. Wi-Fi anténu nebylo možné přišroubovat k Tyře závitem k tomu určeným, proto byla přilepena lepicí páskou k horní části kvadrokoptéry. Toto řešení není optimální, ale pro účely této práce postačí.

Na Atomboard byl nainstalován OS `Xubuntu 12.04 LTS`, který je oficiálně podporován pro použití `ROS`. Po startu je systém upraven tak, aby se nezapínal systém `X Window` pro rychlejší start a snížení nároků na výkon. Nastaveno bylo automatické připojení k Wi-Fi síti. Po této úpravě k práci není nutné připojovat monitor a klávesnici, ale dá se na něj poté připojit přes SSH a vzdáleně zapínat nezbytné služby, zejména `asctec_mav_framework`. První verzí `ROS`, která byla nainstalována, byla `ROS Groovy`, jakožto aktuální vydaná verze. Později však vyšla stabilní verze `ROS Hydro Medusa`, pro kterou byla plná podpora projektu `RosJava`, takže došlo ke změně a verze `ROS` byla povýšena na `Hydro Medusa`. Další důležitou částí byla instalace `asctec_mav_framework`, který zajišťuje komunikaci s HLP na desce AutoPilot. Část frameworku `asctec_hl_firmware` obsahuje firmware v souboru `main.hex`, který byl nahrán do procesoru `HLP` (provedl se flash HLP). Nezbytný byl také balíček

ueye_cam, který zpřístupňuje data z kamery na příslušné topicy. Je třeba obraz z kamery správně nastavit, protože v základním nastavení vytěžuje běh balíčku procesor téměř na sto procent a obraz v plném rozlišení by kladl příliš velké nároky na šířku přenosového pásma. Dalším nezbytným balíčkem byl sensor_msgs, který definuje zprávy pro běžně užívané senzory. V této práci jsou použity zprávy NavSatFix pro přenos informací o GPS pozici.

4.2 Implementace aplikací

Pro vývoj aplikace bylo zvoleno vývojové prostředí Android studio, doporučované na stránkách ROSu[1]. Výhodou tohoto prostředí je vydání knihoven a jádra rosjava, které pracují bezešvě s tímto IDE a také obsahují předprogramované základní tutoriály (např. spojení robota s mobilním zařízením, přenos obrazu). Aplikace byla testována na mobilním zařízení ZTE Skate Pro (též označován ZTE Blade III) s verzí Android 4.0.4, odpovídající API Level 15, což odpovídá 82,7% zařízení s OS Android[6]. Toto zařízení se řadí do kategorie low-end, takže by aplikace neměla mít problém běžet na drtivé většině zařízení. Implementační základy veškerých komponent vychází ze zabudované nabídky Android Studia nebo byly převzaty z knihy Android 4: Průvodce programováním mobilních aplikací[4].

Použité ROS topicy a zprávy

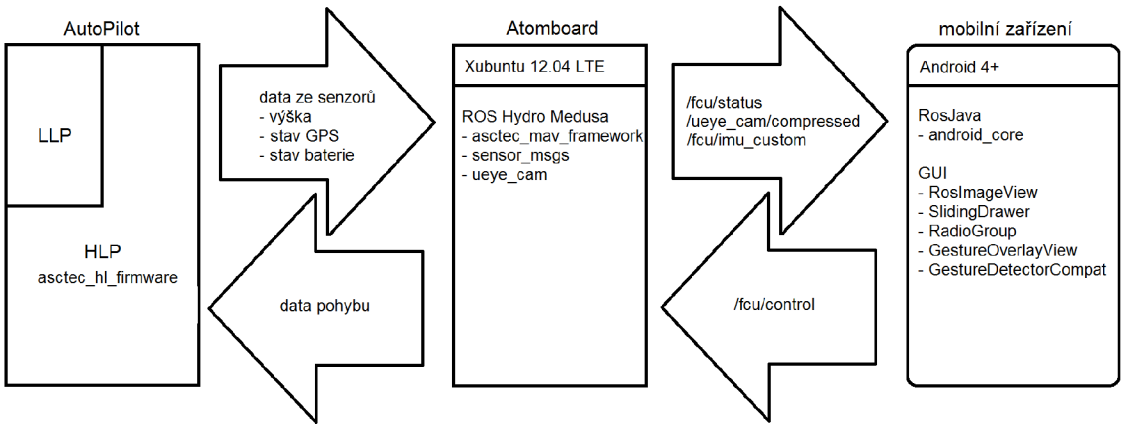
Podle návrhu bylo potřeba implementovat komunikaci mezi všemi třemi částmi systému. Systém přenosu mezi mobilním zařízením a Atomboardem probíhá zveřejňením topiců systému ROS na Atomboardu. Síťové spojení systému ROS pak umožní aplikaci na mobilním zařízení číst tyto topicy a zapisovat na ně.

Mezi jednotlivými deskami nesenými na kvadrokoptéře zajišťuje komunikaci soubor balíčků asctec_mav_framework. Část frameworku na Atomboardu získává data z HLP, na němž je nahrána část asctec_hl_firmware, a pro další použití zveřejňuje a přijímá data na topicy za použití zpráv systému ROS. Zprávy jsou definovány ve dvou balíčcích. Obecné zprávy senzorů jsou v balíčku sensor_msgs a ty specifické pro kvadrokoptéru AscTec Pelican v balíčku mav_hl_comm, jenž je součástí zmíněného asctec_mav_framework. Všechny topicy se nachází v balíčku asctec_hl_interface. Ovládání kvadrokoptéry probíhá pomocí topicu /fcu/control, na nějž se zapisují zprávy typu mav_ctrl. Mezi další využívané topicy patří /fcu/status, ze kterého se používají informace o stavu nabití baterie a GPS signálu. Typ zpráv je mav_status. Potřebná informace o výšce se získává z topicu /fcu/imu_custom zprávami typu mav_imu, protože informace vypočtená z IMU je přesnější než hodnota altitude z GPS modulu. Komprimovaný obraz z kamery je také zveřejňován, a to na topic /ueye_cam/compressed.

Stabilizace letu

Implementovány jsou dvě možnosti kontroly letu různým nastavením řídicích zpráv. V případě dostupnosti GPS signálu je přepnuto autonomní vyvažování pomocí LLP desky AutoPilot. Při tomto módu se nastaví typ zpráv na velocity_body a hodnoty parametrů pro pohyb v osách (x, y, z) udávají rychlost v metrech za sekundu. Dostupnost GPS signálu se získává z topicu /fcu/status a přepnutí na tento mód značí hodnota "GPS fix".

[1] http://wiki.ros.org/android/Tutorials/hydro/Installation%20-%20Ros%20Development%20Environment



Obrázek 4.1: Schéma ROS zpráv a topiců

`Asctec_mav_framework` neumožňuje využívání Height módu pro stabilizaci kvadrokoptéry. Proto při nedostupném GPS módu je nutné využívat přímé ovládání zrychlení motorů kvadrokoptéry v jednotlivých osách. Zprávy jsou přepnuty na typ acceleration a hodnoty parametrů pro pohyb v osách (x, y, z) se udávají v radiánech. Návrh řešení počítá s použitím autonomního držení výšky, proto se po přepnutí do tohoto módu předpokládá, že bude ovládání obtížné a nepohodlné.

Ovládání letu

Ze tří navržených módů ovládání byly realizovány pouze dvě, a to mód letu a mód sledování objektu. Důvodem byla nemožnost ovládání držáků kamery pomocí `asctec_mav_frameworku`. Řízení letu funguje následovně: Uživatel zadá gesto, které program rozpozná, a poté se publikuje zpráva `mav_ctrl` na topic `/fcu/control` ve formátu, který určuje zvolený mód. Při sledování objektu se publikují `ROS` zprávy delší dobu s menšími rychlostmi. Pohyb kvadrokoptéry bude pomalejší a bude trvat delší dobu. Oproti tomu při módu letu jsou na daný topic publikovány zprávy kratší dobu s vyšší hodnotou rychlosti. Toto není jediný rozdíl, gesto kružnice znamená v každém módu vykonání rozdílného manévru. Při letovém se kvadrokoptéra natáčí kolem vlastní osy, avšak u druhého z nich začne kvadrokoptéra sledovaný objekt obletávat. Poloměr je určen rychlostí otáčení a rychlostí letu v daném směru, tyto parametry jsou nastavitelné v menu aplikace.

Rozpoznání gest

OS Android je uspůsoben především pro ovládání dotekem, proto existuje více způsobů a knihoven pro rozpoznávání gest. Z návrhu vyplývá, že řešení musí být schopno rozpoznávat složitější gesta, než jen jednoduché tahy, například kružnici či šipku. Proto se jako první řešení nabízelo využití `GestureOverlayView`, které navíc umožňuje překrývat prvek pod sebou, tedy snímat gesta kreslená přes obraz z kvadrokoptéry. `GestureOverlayView` rozpoznává gesta vytvořená v aplikaci `Gestures Builder`, která je volně ke stažení na Google Play. Knihovna `Gesture Library` rozpoznává gesta jako konkrétní tahy, nikoli jako podobnost obrazců, záleží tedy na počátku kreslení gesta. Bylo tedy nejjednodušším řešením

přidat gesta v opačném směru (např. kreslení šipky vzhůru zleva i zprava). Problémem tohoto řešení bylo však kreslení obyčejných čar v horizontálním a vertikálním směru, sloužící pro základní pohyb kvadrokoptéry do stran. Prvek `GestureOverlayView` musí mít nastavenou orientaci v rozmezí vertical/horizontal a v závislosti na tomto nastavení rozpoznává rovné tahy pouze v daném směru. Toto chování je způsobeno obecnou implementací prvku, kde by gesto v daném směru kolidovalo s akcí scrollování. Při horizontálním nastavení tedy nerozpoznával vertikální tahy. Nepomohlo ani nastavení `GestureStrokeAngleTreshold`, které udává hranici natočení rozpoznávaných gest. Vertikální gesta byla sice při hodnotách blízkých 90.0f rozpoznávána, avšak docházelo k častému chybnému rozpoznání, tedy zaměnění s gestem horizontálním.

Obrázek 4.2: Gesta zachycená v Gestures Builderu

Dalším možným řešením bylo použít `GestureDetectorCompat` z podpůrné knihovny Androidu (android.support.v4). Požadavek kreslení gest přes obraz z kamery kvadrokoptéry se vyřešil nastavením prvku `RosImageView` funkcí `setOnTouchListener`. Rozpoznání základních vertikálních a horizontálních gest proběhlo přetížením metody `onFling` a výpočtem pomocí počátečních a koncových bodů a nastavením minimální délky a rychlosti tahu. Rozpoznání gesta kružnice bylo obtížnější a metoda matematické aproximace z bodů se ukázala velmi nepřesná.

Celkové řešení bylo implementováno kombinací obou přístupů. `GestureOverlayView` se použil pro rozpoznání složitějších gest (kružnice, šipka vzhůru/dolů) a v případě nerozpoznání (tj. `predictionScore` menší než 1) se pomocí funkce `dispatchTouchEvent` předaly informace o tahu o úroveň níže modulu `GestureDetectorCompat`, který rozpozná tahy ve čtyřech základních směrech. Je také rozpoznávané gesto `onDoubleTap`, které zastaví jakýkoli pohyb kvadrokoptéry.

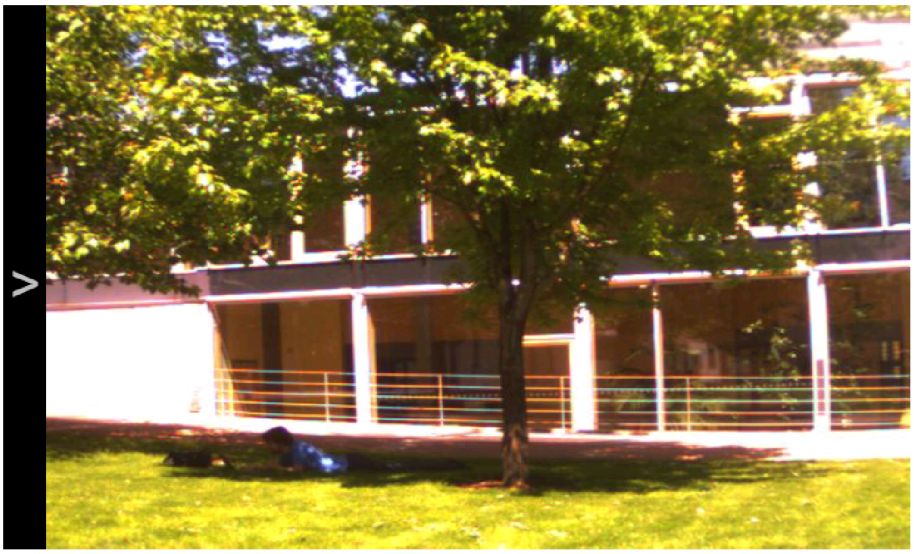
Implementace GUI

Celou obrazovkou pokrývá obraz z kamery, který byl nejprve implementován komponentou `RosImageView` převzatou z tutoriálu `image_transport_tutorial` obsaženém v `android_core`. Jedná se o rozšíření standardní komponenty `ImageView` o rozhraní Node systému `ROS`. Zobrazování touto cestou nebylo kvalitní a docházelo ke zpožděním vlivem vysoké náročnosti na výpočetní výkon. Byl nalezen patch, který namísto `ImageView` používá komponentu

`SurfaceView`[2].

Vysouvací lišta je implementována pomocí komponenty `SlidingDrawer`. V základní implementaci lze nastavit směr vysouvání z dolní nebo pravé strany obrazovky, což je určeno nastavením parametru `orientation`. Aby se dosáhlo navrženého vysouvání lišty z levé strany obrazovky za využití této komponenty, byl jí nastaven parametr `rotation` na hodnotu 180, což odpovídá otočení komponenty o 180 stupňů. Ostatní komponenty, nacházející se uvnitř `SlidingDrawer`, musí mít tento parametr nastaven taktéž z důvodů zachování jejich původní správné orientace.

Tlačítka přepínání módu jsou implementována jako `RadioGroup` nastylovaný do podoby tlačítek. Oproti návrhu není obsaženo tlačítko módu kamery, neboť `asctec_mav_framework` nemá možnost ovládání naklopení kamery.



Obrázek 4.3: Screenshot aplikace se zasunutou ovládací lištou

4.3 Testy funkčnosti Tyry

Na počátku se Tyra (kvadrokoptéra AscTec Pelican) nacházela ve stavu, kdy se nedařilo navázat spojení mezi deskami AutoPilot a Atomboard. Nefungoval přenos přes UART USB převodník ani přes Xbee a jediným funkčním spojením, které však nebylo spolehlivé, bylo spojení sériovou linkou. Její převodník byl zkonstruovaný pouze v laboratoři na nepájivém poli. I přesto však často docházelo k rozsynchronizování spojení. Vina byla připsána prostředí toho času nainstalovaném OS Ubuntu 12.04 `ROS Groovy`. Přeinstalací na OS Arch Linux se problém synchronizace částečně vyřešil, ale nebylo to řešení, které by zajistilo stabilní ovládání kvadrokoptéry systémem ROS. Dalším oznámeným problémem byla samotná schopnost držení pozice systémem AutoPilot a získání dostatečného GPS signálu tvz. GPS Lock.

Začaly se tedy testovat obě desky zvlášť. Deska Atomboard byla předána na kontrolu, aby se vyloučila možnost poškození předešlým pádem kvadrokoptéry. Z časových důvodů se však nečekalo na výsledky kontroly a rovnou byla objednána deska nová, na které probíhaly veškeré následující instalace a experimenty.

[2] https://code.google.com/p/rosjava/issues/detail?id=141

Obrázek 4.4: Screenshot aplikace s vysunutou ovládací

AutoPilot se testoval nejprve připojením Xbee k LLP a sledováním dat ze senzorů. Při bezletových testech v laboratoři velmi kolísal údaj výšky, ostatní údaje neukazovaly na žádný problém. Signál GPS se nepodařilo získat dostatečně silný, aby proběhl GPS Lock indikovaný LED diodou, což se přisoudilo prostředí laboratoře. Testování manuálního módu ovládání radiovým ovladačem nevykazovalo žádné chyby, proto se mohlo přejít k letovým testům prováděným ve venkovním prostředí v blízkosti fakulty.

Aby se předešlo dalším haváriím v případě disfunkce některých senzorů či systému AutoPilot, byl sestrojen jednoduchý **držák** kvadokoptéry. Musel dávat kvadrokoptéře dostatečnou volnost pohybu, nesměl překážet letu a zároveň však musel zabránit kvadrokoptéře havarovat. Nejlepším řešením se ukázalo přivázat cca 2 metrový lehký provaz s gumou k horní části kvadrokoptéry a ten pak připevnit na teleskopickou tyč. Toto řešení vyžadovalo k testování minimálně jednoho dalšího člověka, který operoval držákem, avšak poskytovalo možnost dosažení výšky cca 3m a při rychlosti chůze neomezený prostor do stran.

Byly testovány všechny tři módy, a to Manual, Height a GPS. Manual mód fungoval bezchybně, při snaze o držení výšky se pohybovala kvadrokoptéra nízkou rychlostí kupředu vlevo. Ovládání bylo pro nezkušeného pilota zpočátku náročné. Height mód byl testován po bezpečném vzletu na Manual mód. Kvadrokoptéra zpočátku držela výšku bez nutnosti zasahovat, pohyb dopředu vlevo však zůstal problémem. Ovládání bylo pohodlnější než použití módu Manual, po dobu pětiminutového letu však držení výšky několikrát selhalo a kvadrokoptéra začala nečekaně stoupat či klesat. Během testování Height módu se podařilo získat dostatečný GPS signál k GPS Locku a problém navázání GPS spojení již poté nenastal. GPS mód byl testován také až po vzletu módem Manual. Ovládání bylo nejpohodlnější a nedocházelo ani k větším odchylkám držení výšky. Mírný pohyb dopředu vlevo setrval. V tomto módu se podařilo proletět nižší rychlostí předem určenou trajektorii.

Ke korekci nežádoucího pohybu do stran by měla sloužit posuvná tlačítka vedle joysticku na dálkovém ovládání. Tlačítko pro pohyb vlevo však nefungovalo, což ztěžovalo další testování. Bylo rozhodnuto, že závada není natolik závažná k okamžité opravě, ta by totiž zdržela další testování a vývoj systému.

Během letů byla kvadrokoptéra spojena s notebookem, na němž běžel `ROS` pomocí Xbee,

a přestože bylo spojení značně nestabilní a navázat je se podařilo vždy až na více pokusů, bylo nahráno několik záznamů letu pomocí služby `rosbag` k dalšímu zkoumání.

4.4 Návrh uživatelských testů GUI

Navržené testy musí potvrzovat či vyvracet jasně dané hypotézy o GUI, které vychází z analýzy zadání a požadavků uživatele. Základní hypotézou je bezpečné a bezproblémové ovládání kvadrokoptéry tak, aby ji měl pilot pod kontrolou, po každém úkonu přesně věděl co vykoná a vyhnul se tak havárii. Touto otázkou se budou zabývat dva testy. První má za úkol zjistit, jestli je ovládání dostatečně intuitivní a předvídatelné. Druhý otestuje možnosti a přesnost pilotování kvadrokoptéry na volném prostranství. Další hypotéza se zabývá sledováním scény. Je nutné ověřit, zda pilotovi dostačují funkce sledování objektu a zda mu ovládací a informační prvky nepřekáží. Prvky však musí plnit svou funkci. Toto ověří v pořadí třetí test. Jednotlivé bodové příkazy jsou přiloženy v příloze.

Test č.1 - intuitivnost GUI

K potvrzení či vyvrácení hypotézy intuitivnosti řešení (tj. pilot přesně ví co a jak dělá) aniž by byla kvadrokoptéra vystavena možnosti havárie, bude první test probíhat bez létání. Zvoleným řešením bude dotazník. Pilotovi budou postupně oznamovány příkazy k ovládání, které budou vyhodnocovány tazatelem. Tazatel do formuláře napíše výsledek každého úkolu ze dvou možností, a to "splnil", či "nesplnil". V případě, že pilot nevykoná úkon do deseti sekund, výsledek se počítá jako "nesplnil". Výsledkem testu bude procento správných odpovědí, tudíž čím bude procento vyšší, tím je řešení považováno za intuitivnější.

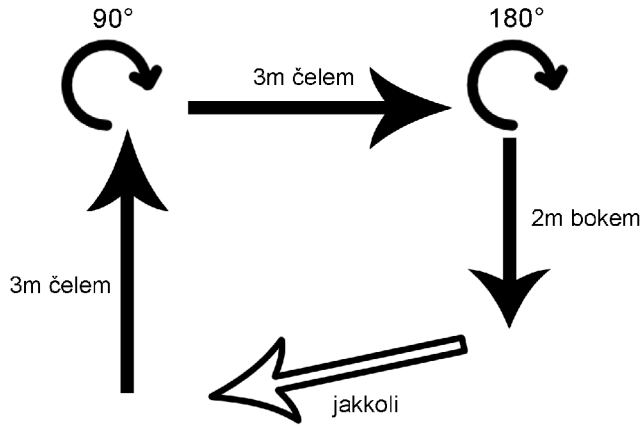
Test č.2 - Ovladatelnost při letu

Tento test zjišťuje, zda platí hypotéza o jednoduché manipulaci letu. Pilot tedy bude testovat funkční aplikaci a přímo ovládat kvadrokoptéru ve vzduchu. Pilot dostane seznam mánevrů, které bude muset vykonat. Hodnotícím faktorem bude čas, za který stihne soubor manévrů udělat. V případě vybočení z letového plánu chybným manévrem a nemožností ho napravit návratem do pozice před chybou se celý test hodnotí jako zmařený a je nutné jej opakovat. Zaznamenává se také, po kolika pokusech pilot zvládl projít celým testem. Čas dosažen navrženou aplikací se bude porovnávat s časem dosaženým rádiovým ovladačem a mobilní aplikací používající virtuální joysticky (vyvinutou Ondřejem Kleinem).

Aby se otestovalo, zda je řešení použitelné i za okolností, kdy není kvadrokoptéra vidět, bude test opakován ještě jednou, avšak po druhé bude pilot otočen zády k předpokládané manévrovací ploše.

Test č.3 - Sledování scény

Hypotézu, zda je návrh řešení vhodný pro sledování bodu budeme testovat stejně jako při předešlém testu za letu. Pilot dostane seznam úkolů, které má vykonat. Měřit se bude opět čas stejně jako v předešlém testu. Smyslem testu je přiblížit se k předem připravené tyči (popřípadě stromu, rohu budovy), na které budou ze dvou stran svírajících úhel 90 stupňů tabule s textem, který bude muset pilot přečíst. Abychom minimalizovali chybu měření schopností čtení pilota, budeme používat pouze několik vět rozprostřených po větší ploše.



Obrázek 4.5: Manévrovací plocha

Čas dosažen navrženou aplikací se bude porovnávat vůči času dosaženého mobilní aplikací používající virtuální joysticky (vyvinutou Ondřejem Kleinem). Doplňkem časového údaje bude dotazník pro subjektivní zhodnocení pozorovacích vlastností řešení.

4.5 Vyhodnocení testů

Tato sekce se zabývá především vyhodnocením kvality splnění jednotlivých aspektů výsledné aplikace, které jsou popsány v kapitole 3.1. Data byla získána provedením a vyhodnocením jednotlivých testů navržených v předešlé kapitole 4.4. Při vyhodnocení jsem nevycházel pouze z naměřených hodnot, ale důležitou roli hrála konzultace s pilotem a vlastní sledování přesnosti pohybu kvadrokoptéry při letu. Vyhodnocení je rozděleno do tří částí podle prováděných testů, a to intuitivnost GUI, ovladatelnost letu a schopnost sledování scény.

Testování intuitivnosti GUI

V této části jsou popsány testy, které probíhaly na prototypu aplikace na mobilním zařízení bez přítomnosti kvadrokoptéry. Intuitivnost řešení byla testována na osmi dobrovolnících, kteří prováděli pokyny podle seznamu uvedeném v příloze A. Před začátkem jim byl na dvacet sekund ukázán obrázek s možnými gesty.

Tabulka 4.1 obsahuje všechny záznamy osmi testovaných subjektů a barevně odlišuje části podle toho, co dané příkazy testují. Celkový výsledek testu nabývá hodnoty 85%, což považuji za indikaci dostatečně intuitivního uživatelského rozhraní. Dále tabulka ukazuje, že používání gest, ať už běžných, či pokročilých, nedělá pilotům problém, což naznačují vysoké hodnoty 94% a 91%. Slabším výsledkem je intuitivnost při ovládání scény. Beremeli však v potaz, že se jedná o zcela nový koncept ovládání, považuji hodnotu 79% pro tuto práci jako dostačující.

Příkaz č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	%
Pilot č.1	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	100%
Pilot č.2	S	S	S	S	S	N	S	S	S	S	S	N	S	S	S	N	S	82%
Pilot č.3	S	S	N	S	S	N	S	S	S	S	S	S	S	S	N	N	S	76%
Pilot č.4	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	94%
Pilot č.5	S	S	N	N	S	S	S	S	N	S	S	N	N	S	N	N	N	53%
Pilot č.6	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	100%
Pilot č.7	S	S	S	S	S	N	N	S	N	S	S	S	S	S	S	N	S	76%
Pilot č.8	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	100%
				94%				91%				79%						**85%**

Tabulka 4.1: Tabulka vyhodnocení testu GUI, vyšší procento úspěšnosti je lepší. Procenta jsou zaokrouhlena na celá čísla. Žlutě jsou zvýrazněny příkazy základních pohybů, modře pokročilých gest a zeleně příkazy pro sledování scény.

Porovnání metod řízení letu kvadrokoptéry

V této sekci jsou uvedeny letové testy, které probíhaly ve venkovním prostředí za použití Wi-Fi hotspotu na mobilním zařízení. Testovací plocha měla jen několik metrů (viz obrázek 4.4), proto byla síla signálu dostatečná a během testů nedocházelo k výpadkům spojení. Cílem těchto testů bylo porovnat vytvořené řešení s dalšími typy ovládání. Konkrétně s rádiovým ovladačem dodaným ke kvadrokoptéře a s aplikací pro mobilní zařízení používající virtuální joysticky vyvinutou Ondřejem Kleinem. Obě tato ovládání byla testována nejprve v manuálním módu, tedy bez využití stabilizace AutoPilot, a také v módu se zapnutou stabilizací pomocí GPS. Vytvořená aplikace používá GPS stabilizaci.

Během testů byla kvadrokoptéra jištěna držákem, jež byl popsán v kapitole 4.3, aby se předešlo haváriím kvůli nezkušenostem pilotů a používání experimentálních ovládání. Z důvodu bezpečí bylo testování omezeno pouze na dvojici pilotů, kteří mají s ovládáním kvadrokoptéry zkušenosti. Tito piloti dostali instrukce obsažené v příloze A.

Výsledky byly zaneseny do tabulky 4.2. Každý let byl opakován maximálně pětkrát, poté byl test prohlášen za neúspěšný. Z hodnot lze vyčíst, že i zkušenější piloti, kteří za sebou mají řádově desítky letů, nebyli schopni vůbec, nebo jen velmi špatně, ovládat kvadrokoptéru bez použití stabilizačních algoritmů systému AutoPilot. Ovládání gesty nemá příliš odlišné hodnoty doby provedení testu, ale údaj počtu pokusů vypovídá o delší době zvykání si na zcela jiný způsob provádění manévrů. Z pozorování a výpovědí pilotů bylo zjištěno, že docházelo k občasnému rychlému rozkolísání výšky. U řízení rádiovým ovladačem a aplikací s virtuálními joysticky toto piloti řešili intuitivně manuálním zvýšením či snížením výšky. Gesty však tyto nuance nedokázali vyrovnat dostatečně hbitě.

Kvalitu a komfort letu ovládáním gesty hodnotím jako uspokojivou. Řadím ji na stejnou úroveň s dalšími dvěma druhy ovládání. Více pokusů na zvládnutí testu přičítám přechodu k jinému způsobu ovládání, než na jaký jsou piloti zvyklí.

Schopnost sledování scény kamerou kvadrokoptéry

Následující text popisuje testy módu sledování. Schopnost komfortního sledování scény je jeden z důležitých požadavků pro vytvořenou aplikaci. Ještě před provedením testů popsaných v kapitole 4.4 byla provedena řada testů, která ukázala na obtížnost tohoto řešení za použití stabilizace GPS systému AutoPilot.

	Rádiový ovladač		Virtuální joysticky Ondřej Klein		Ovládání gesty Petr Běl
Stabilizace	Manuální	GPS	Manuální	GPS	GPS
Pilot č.1	1:35 / 2	**1:08 / 1**	N/A / 5	**1:28 / 1**	**1:50 / 4**
Pilot č.2	N/A / 5	**1:30 / 2**	1:58 / 3	**1:35 / 2**	**1:26 / 3**

Tabulka 4.2: Tabulka vyhodnocení testu ovladatelnosti letu. Formát údaje je [čas / počet pokusů]. Čas je udáván ve formátu [minuty:sekundy].

Systém sledování byl navržen a implementován tak, aby pohyb byl co nejplynulejší kvůli tomu, aby obraz z kamery kvadrokoptéry nebyl rozmazaný. Předpoklady ke správnému chování výsledného řešení byly: Schopnost držení kvadrokoptéry na daném bodě ve vzduchu, což měl zajišťovat systém AutoPilot pomocí GPS, a schopnost pomalého plynulého letu v určeném směru. Až pilotní testy na Tyře ukázaly, že tyto předpoklady nejsou oproti očekáváním založených na nastudované teorii naplněny. Stabilita kvadrokoptéry náhodně kolísá, což nevadí letu, avšak sledování významně ruší. Řešení je funkční a použitelné, ale nenaplňuje kritérium komfortu sledování. Při oblétávání objektu, tedy pohybu ve dvou osách zároveň se navíc kvadrokoptéra destabilizuje úplně a je nutné přistát. Toto chování nastává při nastavení poloměru obletu v řádu metrů. Ovládání pomalým pohybem je nepřesné a chce-li pilot sledovat objekt na bližší vzdálenost v řádu metrů, je jednodušší přiblížit se módem letu a pro sledování kvadrokoptéru zastavit.

Schopnost sledování scény tímto letovým módem lze použít na větší vzdálenosti v řádu desítek metrů, na vzdálenosti v řádu metrů však hodnotím jeho schopnost jako nedostatečnou. Hlavním důvodem jsou nenaplněné předpoklady o stabilitě letu systémem AutoPilot.

Celkové zhodnocení výsledků

Výsledky jednotlivých testů zde přenesu do širšího kontextu celého systému. Hodnotící kritéria vychází ze zadání a cíle práce. Patří mezi ně tedy intuitivnost uživatelského rozhraní pro nezkušené piloty, možnost kvalitního sledování scény a řízení kvadrokoptéry komfortněji než rádiovým ovladačem.

Z tohoto pohledu systém splňuje všechny požadavky minimálně stejně dobře, jako další řešení s ním postavená do kontrastu. Ať už zkušení piloti, či nováčci, neměli problém používat vytvořenou aplikaci a v testech dopadli podobně, což potvrzuje splnění prvního bodu. Druhý bod je naplněn také. Mód sledování je použitelný sice jen pro pozorování objektů ve větší vzdálenosti, ale pozorování na kratší vzdálenosti je umožněno módem letovým. Při řízení gesty byly zaznamenány lepší časy, než při ovládání joysticky v manuálním módu. Výsledky s použitím stabilizace byly podobné. Má aplikace však nedovoluje extrémní vychýlení letu kvadrokoptéry a umožňuje zastavit pohyb poklepáním na displej, na rozdíl od joysticku, který je pro zastavení a držení výšky nutné dát přesně do středové polohy. Z tohoto důvodu hodnotím ovládání jako mírně komfortnější.

Kapitola 5

Závěr

Hlavním cílem této práce bylo navrhnout a implementovat systém ovládání kvadrokoptéry zaměřený na sledování scény. Tento cíl se podařilo splnit, jak vyplývá ze zhodnocení uvedeném v kapitole 4.5, přestože má řešení několik nedostatků.

Před návrhem byla prostudována dostupná literaturu především o platformě `ROS`, vývoji mobilních aplikací a konkrétním robotu AscTec Pelican. Další fází studia bylo získávání informací o existujících řešeních. Na základě těchto znalostí byl navržen systém odpovídající zadaným kritériím.

V rámci této bakalářské práce byl realizován systém umožňující ovládat kvadrokoptéru. Byla implementována aplikace pro mobilní zařízení s OS Android, která zajišťovala interakci s pilotem. Největší odlišností mého řešení od nalezených existujících je zvolený způsob ovládání gesty. Toto řešení vynechává grafické ovládací prvky, které zabírají místo obrazu, a proto je vhodné pro účel sledování scény. Dalším vylepšením tohoto hlediska je použití dvou módů letu, sledovacího a letového. Mód sledování řídí let kvadrokoptéry pomalu a plynule delší dobu bez nutnosti zakrývat displej mobilního zařízení zadáváním ovládacího gesta.

K prezentaci řešení bylo vytvořeno krátké propagační video a plakát. Tyto materiály jsou přiloženy k této bakalářské práci na CD.

Provedenými experimenty jsem zjistil, že pro sledování objektu v menší vzdálenosti je stabilizace systému AutoPilot nedostatečná pro pomalé rychlosti sledovacího módu a ty tímto ztrácejí na významu. Sledování na tyto vzdálenosti lze však pohodlně provádět módem letu.

Možný rozvoj mého řešení se může ubírat využitím systému GPS pro zadání konkrétního bodu zájmu a jeho sledování. Pilot by tak měl možnost zadat souřadnice, či označit bod na mapě, a kvadrokoptéra by při pohybu byla pořád natočena kamerou tímto směrem. Další možností je rozšířit `asctec_mav_framework` o balíček ovládající držák kamery.

Literatura

[1] Aaron Martinez, Enrique Fernández: *Learning ROS for Robotics Programming.* Packt, 2013, iSBN 1782161449.

[2] Alexander Maksymiw: AscTec AutoPilot. http://wiki.asctec.de/display/AR/AscTec+AutoPilot, [cit. 2014-2-1].

[3] Economics, D.: Developer Economics Q3 2013 analyst report. http://www.visionmobile.com/DevEcon3Q13.html, Červen 2013 [cit. 17-12-2013].

[4] Grant Allen: *Android 4: Průvodce programováním mobilních aplikací.* COMPUTER PRESS, 2013, iSBN 978-80-251-3782-6.

[5] Guangzhou Walkera Technology Co.,Ltd: Walkera QR W100S. http://www.walkera.com/en/showgoods.php?id=2548, [cit. 2014-2-2].

[6] Jerry Hildenbrand: Latest platform numbers are in: KitKat climbs to 8.5http://ardrone2.parrot.com/apps/director-mode/, květen 2014 [cit. 2014-2-1].

[7] Kohler, D.: Introduction to rosjava. Konference ROSCon2012, 19-20. květen 2012.

[8] Lars Knoll: Qt 5.2 Released — The Best Qt Yet. http://blog.qt.digia.com/blog/2013/12/12/qt-5-2-released-the-best-qt-yet/, [cit. 2014-2-12].

[9] Parrot SA.: Ardrone 2.0 User Guide. http://parrotcontact.emencia.net/website/user-guides/download-user-guides.php?pdf=ar-drone-2/AR-Drone-2_User-guide_Android_UK.pdf , [cit. 2014-2-1].

[10] Parrot SA.: Ardrone Director Mode. http://ardrone2.parrot.com/apps/director-mode/, [cit. 2014-2-1].

[11] Tully Foote: Concepts - ROS Wiki. http://wiki.ros.org/ROS/Concepts, [cit. 2013-12-15].

[12] Výzkumná skupina robotiky Robo@FIT: Vybavení. http://www.fit.vutbr.cz/research/groups/robo/eqp.php, [cit. 2014-4-13].

Příloha A

Zadání testů

Test č.1 příkazy:

1. spojte se s kvadrokoptérou
2. sdělte stav baterie
3. nastartujte motory
4. leťte vpřed
5. leťte vpravo
6. vraťte se na původní místo vzletu
7. otočte se ve směru hodinových ručiček
8. otočte se proti směru hodinových ručiček
9. vzleťte nahoru
10. sestupte dolů
11. přepněte na mód sledování objektu
12. přibližte se k bodu zájmu
13. vyleťte nad bod zájmu
14. posuňte se vpravo vůči bodu
15. vraťte se zpět na místo, odkud jste začali pozorovat
16. obleťte bod zájmu
17. přistaňte

Test č.2 příkazy:

1. vzlétněte
2. leťte kupředu cca 3 metry
3. otočte kvadrokoptéru o 90° po směru hodinových ručiček
4. vzlétněte výše o cca 1 metr
5. leťte kupředu o cca 3 metry
6. otočte kvadrokoptéru o 180° libovolným směrem
7. leťte doleva o cca 3 metry
8. vraťte se na místo vzletu a přistaňte

Test č.3 příkazy:

1. vzlétněte
2. zadejte GPS souřadnice bodu
3. přibližte se k bodu na 2 metry
4. přečtěte věty z přední části tabule (pohybujte se jakkoli v rámci sledovacího módu)
5. obleťte objekt o 90° v bezpečné vzdálenosti
6. přečtěte věty z boční části tabule
7. navraťte se zpět a přistaňte

Příloha B

Slovník pojmů

- ROS - Robot Operating System 2.1
- AscTec Pelikan - kvadrokoptéra firmy Ascending Technologies 4.3
- Tyra - pracovní pojmenování konkrétní kvadrokoptéry 4.3
- AutoPilot - systém stabilizace kvadrokoptéry firmy Ascending Technologies 2.3
- Atomboard - Deska s vysokým výpočetním výkonem, osazená nízkopříkonovým procesorem Atom
- HLP - High Level Processor desky AutoPilot umístěné na kvadrokoptéře AscTec Pelican 2.3
- LLP - Low Level Processor desky AutoPilot umístěné na kvadrokoptéře AscTec Pelican 2.3
- API - Application Programming Interface, rozhraní pro programování aplikací
- URI - Uniform Resource Identifier, jedinečný identifikátor
- GUI - Graphic User Interface, grafické uživatelské rozhraní
- SSH - Secure Shell
- Xbee - rádiový modul pro bezdrátovou komunikaci