



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

**NÁVRH STRUKTURY ŘÍDICÍHO SYSTÉMU PRO
KONVOJ AUTONOMNÍCH VOZIDEL S VYUŽITÍM
FRAMEWORKU ROS**

AUTONOMOUS VEHICLE CONVOY SYSTEM ARCHITECTURE BASED ON ROS FRAMEWORK

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

Hynek Buchta

VEDOUCÍ PRÁCE

SUPERVISOR

doc. Ing. Stanislav Věchet, Ph.D.

BRNO 2017

Zadání bakalářské práce

Ústav: Ústav automatizace a informatiky
Student: **Hynek Buchta**
Studijní program: Strojírenství
Studijní obor: Aplikovaná informatika a řízení
Vedoucí práce: **doc. Ing. Stanislav Věchet, Ph.D.**
Akademický rok: 2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Návrh struktury řídicího systému pro konvoj autonomních vozidel s využitím frameworku ROS

Stručná charakteristika problematiky úkolu:

Návrh struktury řídicího systému vhodného pro konvoj autonomních vozidel s využitím frameworku ROS. Navržený řídicí systém musí umožňovat integraci různých typů senzorů pro zajištění rošiřitelnosti do budoucích aplikací.

Cíle bakalářské práce:

- 1) Seznamte se s aktuálním stavem frameworku ROS.
- 2) Prostudujte možnosti meziprocesové komunikace frameworku ROS.
- 3) Navrhněte vhodnou strukturu řídicího systému vhodného pro autonomní vozidla v konvoji.
- 4) Navržený systém otestujte.

Seznam doporučené literatury:

THRUN Sebastian, BURGARD Wolfram, and FOX Dieter. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series). Intelligent robotics and autonomous agents. The MIT Press, August 2005.

CHOSSET Howie, LYNCH Kevin M., HUTCHINSON Seth, KANTOR George, BURGARD Wolfram, KAVRAKI Lydia E., and THRUN Sebastian. Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents). The MIT Press, June 2005.

ROS.org. ROS.org | Powering the world's robots. [online]. 2.11.2016 [cit. 2016-11-02]. Dostupné z: <http://www.ros.org/>.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan fakulty

ABSTRAKT

Náplní této bakalářské práce je návrh architektury řídicího systému pro konvoj autonomních vozidel s využitím softwarových nástrojů frameworku Robot Operating System. Navržený řídicí systém musí umožňovat integraci různých typů senzorů pro zajištění rozšiřitelnosti do možných budoucích aplikací. Systém je realizován na platformě Pi 3, jednodeskovém počítači společnosti Raspberry. Hlavní vstupní periferií je kamera strojové vidění připojená přes rozhraní USB. Další důležitou součástí je modul bezdrátové sítě sloužící ke komunikaci mezi vozidly a vzdálenému přístupu k jednotlivým vozidlům konvoje. O řízení motorů se stará mikrokontroler mbed LPC1768 připojený prostřednictvím sériové linky.

ABSTRACT

This thesis deals with the problematics of designing control system for convoy of autonomous vehicles using software tools offered as part of the ROS framework, which must take into account possible future implementations of various types of sensors. Designed system is realized on a single-board computer Pi 3 platform, the Raspberry company product. The main input sense is camera stream connected via USB interface. Another important part of the system is a wireless network module allowing communication between vehicles in the convoy as well as remote access to each of them. The engines are directly controlled by a mbed LPC1768 microcontroller communicating through serial link.

KLÍČOVÁ SLOVA

Semiautonomní konvoj, řídicí systém, Raspberry Pi 3, ROS framework, Ubuntu, Linux, Python.

KEYWORDS

Semiautonomous convoy, control system, Raspberry Pi 3, ROS Framework, Ubuntu, Linux, Python.

BIBLIOGRAFICKÁ CITACE

BUCHTA, H. *Návrh struktury řídicího systému pro konvoj autonomních vozidel s využitím frameworku ROS*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 41 s. Vedoucí bakalářské práce doc. Ing. Stanislav Věchet, Ph.D.

PODĚKOVÁNÍ

Děkuji vedoucímu práce doc. Ing. Stanislavu Věchetovi, Ph.D. za poskytnuté rady a vstřícný přístup.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením doc. Ing. Stanislava Věcheta, Ph.D. a s použitím literatury uvedené v seznamu literatury.

V Brně dne 1. 5. 2017

.....

Hynek Buchta

OBSAH

1	ÚVOD.....	15
2	SEMIAUTONOMNÍ KONVOJ	17
2.1	Využití	17
2.2	Platooning	18
2.2.1	Přínos	18
2.2.2	Úskalí	18
2.3	Inteligentní transportní systémy	18
3	RASPBERRY PI 3	19
3.1	Využití	19
3.2	Parametry	20
4	GNU/LINUX.....	21
4.1	Historie	21
4.2	Ubuntu	21
4.3	Programovací nástroje	21
4.3.1	SSH (Secure Shell)	21
4.3.2	Python	22
4.3.3	Bash	23
4.3.4	Formát JSON	23
5	ROS FRAMEWORK	24
5.1	Master	24
5.2	XML-PRC protokol.....	25
5.3	Server parametrů.....	25
5.4	Uzel.....	26
5.5	Transport témat.....	27
5.6	Navázání připojení k tématu.....	27
6	NÁVRH STRUKTURY ŘÍDÍCÍHO SYSTÉMU.....	28
6.1	Instalace operačního systému	28
6.2	Příprava systému ROS.....	29
6.3	Vstupní periferie	29
6.4	Meziprocesová komunikace	30
6.5	Implementace frameworku ROS	30
6.5.1	Publikující uzel	31
6.5.2	Přijímací uzel	32
6.6	Inicializace	34
6.7	Přístup a editace	35
7	ZÁVĚR	37
8	SEZNAM POUŽITÉ LITERATURY.....	39
9	SEZNAM PŘÍLOH.....	41

1 ÚVOD

Programování pohybu a chování robotů je obecně považováno za obtížnou disciplínu z hlediska realizace, hlavně z důvodu velkého množství zdánlivě zanedbatelných vnějších vlivů, které však mohou mít na výsledek rozhodující vliv [1, 2]. Donedávna návrh takovýchto řídicích systémů vyžadoval velkém množství práce nutné k získání jen základní funkcionality.

Zásadní změnu do této oblasti přinesla volně dostupná sada nástrojů pro programování robotů souhrnně nazývaná Robot Operating System, jejíž možnosti a využití k návrhu řídicího systému pro konvoj autonomních vozidel jsou hlavní náplní této práce. V úvodu je však pozornost věnována stručnému seznámení s deskovým počítačem Pi 3, produktem společnosti Raspberry, na kterém je tento řídicí systém realizován. Dále se práce zabývá operačním systémem Linux, konkrétněji jeho distribucí Ubuntu Mate, jejíž prostředí a nástroje jsou využity pro návrh a běh řídicího systému a řadě dalších nástrojů a programovacích jazyků, které byly při návrhu použity.

Navržená architektura umožňuje autonomní pohyb vozidel v konvoji, zejména na základě údajů kamery sledující značku na zádi vozidla předchozího, popřípadě příkazů a bezdrátové komunikace. Zatímco vedoucí vozidlo konvoje je řízeno rádiově obsluhou, pohyb dalších vozidel v konvoji je plně v režii navrženého řídicího systému. Navržený řídicí systém je následně experimentálně otestován.

2 SEMIAUTONOMNÍ KONVOJ

2.1 Využití

Automatizace dopravy si pomalu hledá cestu do našich každodenních životů. Za několik málo let budeme na silnicích dosti možná potkávat malé konvoje několika vozidel jedoucích v těsném závěsu za sebou, které se chovají jako jeden celek. Automatizace dopravy slibuje zvýšení bezpečnosti přepravy, její efektivity a úspory času obsluhy, neboť se nemusí nutně aktivně věnovat řízení. Podobné technologie samozřejmě nalézají své využití i v důležitějších uplatněních, a to všude tam, kde mají potenciál zachraňovat lidské životy, jako v případě konvojů armádních vozidel, které se mohou stát cílem útoků a nástražných výbušných zařízení. Komerční vývoj těchto technologií je v současnosti značně finančně náročný, přesto existuje řada armádních i civilních projektů [3]. Zejména v nákladní dopravě se uchytil pojem *platooning*, označující vozidla pohybující se ve skupinách (*platoons*). Hlavní výhodou v nákladní dopravě je, že všechna vozidla cestují stálou, stejnou rychlostí. Důsledkem je zajištění vyšší plynulosti dopravy, a to především ve vytížené evropské dopravní síti.



Obr. 1: Zkušební autonomní konvoj v podání nákladních aut Scania [4].

2.2 Platooning

Je metoda navyšování kapacit dopravní infrastruktury shlukováním vozidel do menších semiautonomních a autonomních konvojů vozidel jedoucích v těsném závěsu za sebou, což je umožněno elektronickými řídicími, popřípadě i mechanickými prostředky. Díky vzájemné komunikaci mezi jednotlivými členy konvoje, je umožněno ideálně souběžné zrychlování i brzdění, což elegantně zabraňuje potenciálně značným materiálním škodám [5]. Budoucí chytrá auta by byla schopná se automaticky připojit a opustit konvoje dle potřeby.

2.2.1 Přínos

- Pohyb vozidel v těsném závěsu výrazně snižuje odpor vzduchu. S tím se váže úspora paliva a snížení emisí vypouštěných do ovzduší.
- Přístup slibuje zkrácení přepravních časů během hodin dopravní špičky.
- Při cestování na delší vzdálenosti vozidla ve sledovacím módu jsou v provozu bez nutnosti obsluhy.
- Celkově menší počet dopravních nehod a výrazná redukce škod, při nepředvídatelných situacích.

2.2.2 Úskalí

- Řidiči mají daleko menší, až žádnou možnost kontroly nad vozidlem. Jsou plně odkázáni na řídicí počítač a software.
- Z důvodu dlouhodobě snížených nároků obsluha nemusí být schopna reagovat dostatečně rychle v případě selhání řídicího systému.

2.3 Inteligentní transportní systémy

Chytré konvoje, často obecně shrnuté pod označením *vehicle platoons* či *platooning*, jsou problematikou samy o sobě a přesto, že jsou převážně stále ve stádiu vývoje a inovací, jsou de facto spolu s plně autonomními vozidly základním prvkem stále častěji skloňovaného fenoménu inteligentních transportních systémů.

3 RASPBERRY PI 3

3.1 Využití

Jedná se o jednodeskový počítač určený k univerzálnímu použití ve vestavěných systémech, k řízení, správě a realizaci bezdrátových sítí a podobně. Jako celek je tvořen komponenty sestavenými na jediné desce bez nutnosti aktivního chlazení, což spolu s využitím flash paměti oproti běžným stolním počítačům zaručuje vyšší spolehlivost. Univerzálnost jeho použití je definována zejména širokými komunikačními možnostmi, které jsou dány přítomností bezdrátového modulu Wi-Fi, Ethernetu a Bluetooth doplněných možnostmi komunikace přes sériovou linku GPIO (*General Purpose Input Output*) se čtyřiceti piny, čtyři porty USB 2 [6]. Grafický výstup na zobrazovací zařízení je možný prostřednictvím portu micro HDMI (*High Definition Multimedia Interface*).



Obr. 3: Jednodeskový počítač Raspberry Pi 3 [7].

3.2 Parametry

Jednodeskový počítač s těmito parametry tedy již nabízí daleko více možností než mikroprocesory, které by se pro tento účel daly rovněž použít, ale realizace veškerých komunikačních a řídicích procedur by byla zdlouhavá a pracná. Značnou výhodou, kterou výkonný hardware přináší je možnost instalace libovolné distribuce operačního systému GNU/Linux, nebo verze operačního systému Windows 10 IoT upraveného pro jednodeskové počítače založené na architektuře ARM. V našem případě se jedná o distribuci Ubuntu ve verzi MATE 16.04, neboť je k dispozici verze určená přímo pro počítače Raspberry Pi 3.

Komponent	Popis
Chipset	Broadcom BCM2837
CPU	4x Cortex A-53 (ARMv8, max. 1,2 GHz)
GPU	Broadcom VideoCore IV (1080p, max. 300 MHz)
RAM	1 GB LPDDR2 (900 MHz)
Drátová síť	10/100 Ethernet
Bezdrátová síť	802.11n 2,4 GHz
Bluetooth	4.1 Classic, Bluetooth Low Energy
GPIO	40 univerzálních vstupních/výstupních pinů
Porty	Micro HDMI, 3,5mm analog audio-video jack, 4x USB 2.0, CSI (Camera Seriál Interface), DSI (Display Seriál Interface)

Tab. 1: Technické parametry jednodeskového počítače Raspberry Pi 3 [8, 9].

4 GNU/LINUX

4.1 Historie

Linux v pravém slova smyslu je pouze jádrem operačního systému, napsaného Linusem Torvaldsem, po vzoru systému Unix s ohledem na plnění standardu POSIX. [10] Distribuce operačního systému Linux, jako je např. Arch Linux, CentOS, Debian, Fedora, Gentoo, openSUSE, Ubuntu a další, kromě jádra obsahují množství dalšího svobodného softwaru, který poskytuje mimo jiné uživatelské prostředí, nástroje pro úpravu a správu multimediálních souborů, kancelářské nástroje a řadu dalších, které jako celek tvoří moderní operační systém, jak jej známe dnes. [11]

Původně byl navržen pro osobní počítače založené na architektuře x86, ale brzy se rozšířil na více platforem, než jakýkoliv jiný operační systém. Linuxového jádra využívá i v současnosti nejrozšířenější mobilní operační systém Android a je častou volbou pro servery a superpočítače. Své uplatnění nalézá také v celé řadě dalších aplikací včetně vestavěných systémů, chytrých televizí, hodinek i herních konzolí. Zdrojový kód je volně dostupný k užítku, úpravám a distribuci, komerční i nekomerční, kýmkoliv v souladu s licenčními podmínkami jako např. všeobecná veřejná licence GNU, k jeho vývoji tedy dílčími vylepšeními přispívají uživatelé z celého světa.

4.2 Ubuntu

Ubuntu je operační systém, který vychází z linuxové distribuce Debian. Je vyvíjen společností Canonical Ltd. se sídlem ve Velké Británii, založenou a dotovanou jihoafrickým podnikatelem Markem Shuttleworthem. Nová verze vychází každého půlroku s bezplatnou podporou na devět měsíců, poskytující aktualizace a ladění chyb. První vydání vyšlo v srpnu 2004 pod označením Ubuntu 6.06. Každé dva roky pak vychází verze LTS (Long Term Support) s pětiletou podporou. [12]

4.3 Programovací nástroje

Návrh řídicího systému z podstaty věci znamená tvorbu software, návrh komunikace jednotlivých dílčích prvků softwaru a jejich propojení s využitím vhodných programovacích nástrojů.

4.3.1 SSH (Secure Shell)

Secure Shell je kryptografickým síťovým protokolem umožňujícím provozování síťových služeb bezpečně přes nezabezpečenou síť. Nejtypičtějším použitím, jako i v našem případě, je vzdálený přístup uživatele k účtu do počítačových Unix-like systémů,

ačkoliv omezené využití se nabízí také pro systémy Windows [13]. Využit je model klient-server. Záměrem návrhu byla náhrada *Telnetu* a nezabezpečených protokolů jako *Berkeley rlogin*, *rsh* a *rexec*, které posílají informace, jako i hesla, ve formě čistého textu a vystavují je tak potenciálnímu odhalení pomocí analýzy paketů. Zajímavostí je, že podle informací zveřejněných na webu *WikiLeaks*, *National Security Agency* má k dispozici algoritmy, kterými je za určitých okolností možné dešifrovat komunikaci SSH a získat přístup k přenášeným datům [14].

4.3.2 Python

Python je vysokoúrovňovým programovacím jazykem univerzálním, určeným pro širokou škálu uplatnění. Byl navržen Guidem van Rossumem jako následovník jazyka ABC a první verze byla přestavena veřejnosti v roce 1991. Hlavním záměrem jeho návrhu bylo poskytnout jazyk se zvýšenou přehledností a čitelností. Hlavním znakem tohoto přístupu je členění kódu odsazením, takzvanými bílými znaky, namísto závorkování nebo použití klíčových slov, jako je tomu v jiných programovacích jazycích. Také syntaxe jazyka umožňuje programátorům vyjádření zamýšlených konceptů menším množstvím řádků kódu, než je možné např. v jazyce C++, nebo Java. Python je jazyk multiparadigmatický a disponuje automatickou správou paměti. Jeho překladač je dostupný pro celou řadu operačních systémů a platforem. Základní filozofie jazyka je shrnuta v práci *Zen of Python* [15], která je souhrnem zásad jako jsou např.:

- Hezké je lepší než škaredé.
- Jednoduchost spíše než složitost.
- Čitelnost se počítá.
- Komplexnost je lepší než komplikovanost.
- Explicitnost je lepší než implicitnost.

4.3.3 Bash

Bash je název pro shell systému Unix a příkazový jazyk navržený Brianem Foxem jako součást projektu GNU, jako svobodná náhrada komerčního Bourne shell. Následně se uchytí jako výchozí pro distribuce systému Linux a MacOS společnosti Apple a je dostupná i verze pro Windows 10. Bash je ve své podstatě příkazovým procesorem, typicky běžícím v prostředí textového okna, které umožňuje vykonávání požadovaných akcí zadáváním příkazů. Spouštěny mohou být také sekvence příkazů zapsané v textovém souboru zvaném skript. [16]

Syntaxe příkazů Bash je rozšířením příkazů Bourne shell, v důsledku čehož umožňuje spouštění drtivé většiny skriptů původně napsaných pro Bourne shell. Mezi jeho vlastnosti usnadňující práci v textovém prostředí patří automatické dokončování a návrh příkazů stisknutím obvykle klávesy Tab. Implementováno je i automatické dokončování příkazů.

4.3.4 Formát JSON

Název JSON, je zkratkou anglického názvu JavaScript Object Notation. Jedná se o formát navržený kolem roku 2000 Douglasem Crockfordem, který využívá člověkem snadno čitelný text k přenosu datových objektů, sestávajících z prvků a jejich hodnot. Jeho nejčastějším využitím je realizace asynchronní komunikace, kde často nahrazuje formát XML. Výhodou formátu je jeho nezávislost na konkrétním jazyce [17].

Jeho konstrukce vychází ze syntaxe jazyka JavaScript, jak název napovídá, přesto v současnosti spousta různých programovacích jazyků zahrnuje nástroje pro generování a zpracování dat uložených ve formátu JSON.

5 ROS FRAMEWORK

ROS (Robot Operating System) je soubor softwarových nástrojů určených k ovládání a návrhu robotických komponentů a robotů šířených pod licenci BDS, která je považována za jednu z nejsvobodnějších. Řídicí systém založený na frameworku ROS sestává z řady nezávislých uzlů, které mezi sebou komunikují na principu modelu vysílač – přijímač, popřípadě klient – server. Sensor integrovaný do systému může tedy být snadno realizován jako uzel, který průběžně publikuje data získaná čidlem jako proud zpráv, které mohou být zpracovány libovolným množstvím jiných uzlů, jako jsou filtry, loggery i různými jinými vysokoúrovňovými systémy. Značnou výhodou systému ROS je míra jeho multiplatformnosti, navrhovaný systém tedy může být provozován na vícero počítačích s různými operačními systémy.

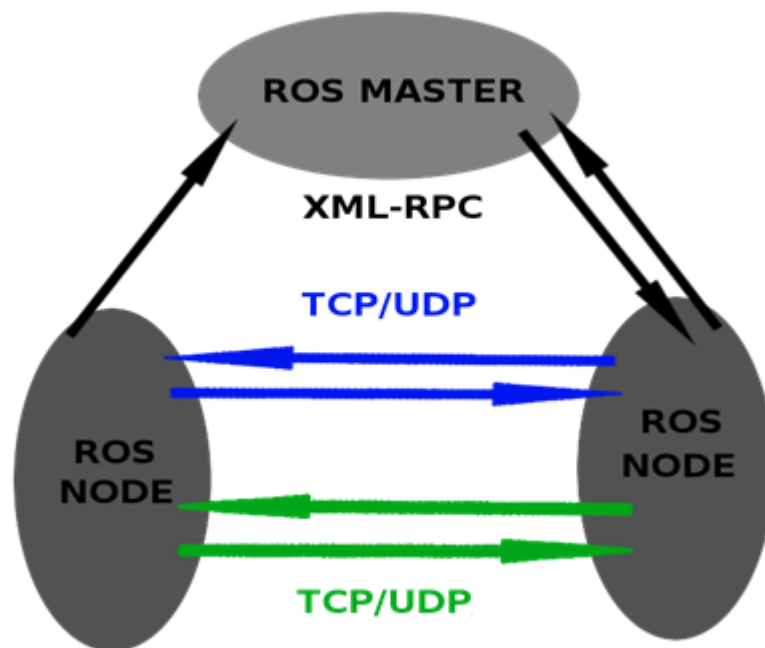
V praxi tedy není žádný problém realizovat komunikaci uzlů, z nichž každý běží na jiném z podporovaných operačních systémů, na nichž je ROS instalován. Plné podpory se doposud těší systémy Ubuntu nebo Debian, některé další jsou prozatím podporovány jen experimentálně jako Gentoo, MacOS a do jisté míry i Android. Toto činí ROS velmi všestranným nástrojem s otevřeným zdrojem, to znamená, že k vývoji a údržbě přispívá široká řada uživatelů z celého světa. Základním prvkem systému ROS je uzel Master, který umožňuje ostatním částem uzlů navázat vzájemnou komunikaci, což značně ulehčuje práci, není totiž nutné adresovat proud dat na určitou adresu. Místo toho lze jednoduše směřovat komunikaci z jednoho do libovolného počtu jiných uzlů a veškerá režie komunikace a toku dat je řízena automaticky uzlem Master. Jednotlivé uzly jsou registrovány u uzlu Master, na nějž lze pohlížet jako na tabulku s údaji a adresami jednotlivých uzlů a míst určení dat.

Uzel, který má přístup k datům ze sensorů publikuje data na určité téma a libovolný počet jiných uzlů, data publikovaná k určitému tématu přímo přijímá a zpracovává prostřednictvím protokolu TCP/IP. V případě modelu klient – server může libovolný uzel podobně jako v předchozím případě registrovat u uzlu Master specifickou službu. V tomto případě si uzel nejprve vyžádá data od uzlu, který má přístup k datům ze sensoru a obratem je zašle žadateli.

5.1 Master

Základním softwarovým prvkem frameworku ROS je uzel Master, který je implementován využitím protokolu XML-RPC (*Extensible Markup Language Remote Procedure Call*). Tento protokol byl zvolen hlavně z důvodu nenáročnosti, nevyžaduje stavuplné připojení a nabízí snadnou dostupnost v řadě použitelných programovacích jazyků. Například v jazyce Python lze pomocí jakéhokoliv překladače a zahájit komunikaci přímo s ROS masterem. Součástí uzlu Master je také registrační API (*Application Programming Interface*), které umožňuje uzlům registrovat se jako publisher, subscriber nebo zprostředkovatele služeb. Každý Master uzlu slouží k

identifikaci URI, jehož hodnota je nadále uložena v proměnném prostředí `ROS_MASTER_URI` a sestává z hodnot `host:port` aktuálního XML-PRC serveru, který master provozuje. Ve výchozím stavu je využit port 11311 [18].



Obr. 7: Schéma komunikace jednotlivých uzlů mezi sebou a k tomu použité protokoly.

5.2 XML-PRC protokol

Jedná se o komunikační protokol založený na protokolu HTTP (*Hyper Text Transfer Protocol*). XML-RPC funguje na základě zasílání HTTP požadavků na server protokolu. Klientem v tomto případě je typicky program, který požaduje volání jediné metody vzdáleného systému.

5.3 Server parametrů

Server parametrů je sdílený slovník dostupný prostřednictvím síťového API. Uzly tento server používají k ukládání a získávání parametrů za běhu. Vzhledem k tomu, že hlavním záměrem při jeho návrhu nebylo dosažení vysokého výkonu, nejlépe se uplatňuje především pro statická, nebinární data, jako jsou konfigurační parametry a podobné.

Přestože server parametrů je vlastně součástí uzlu Master, je vhodné na jeho API pohlížet odděleně. Stejně jako v případě uzlu Master využívá protokolu XML-RPC kvůli následné snadné integraci s knihovnami klientů a rovněž poskytuje výraznou typovou flexibilitu při ukládání a přijímání dat [19].

Server parametrů je schopen ukládat základní celočíselné hodnoty XML-RPC protokolu (32bit celočíselné hodnoty, *boolean* hodnoty, řetězce, *doubles*, data normy iso8601, seznamy a base64-kodovaná binární data). Možné je i ukládání slovníků (tzv. struktur), tyto však mají speciální význam. Je použita takzvaná „slovníky slovníků“ reprezentace pro jmenné servery, kde každý slovník představuje jednu úroveň jmenné hierarchie a každý klíč v libovolném slovníku představuje jmenný prostor. Je-li některá hodnota slovníkem, předpokládá se, že jde o hodnotu jmenného prostoru. XML-RPC programovací aplikační rozhraní umožňuje snadnou integraci volání serveru parametrů bez nutnosti použití knihoven ROS klienta.

5.4 Uzel

Uzly ROS mohou využívat několik typů API:

- XML-RPC Slave API, které zastává dvě role. Jednak příjem volání od Master uzlu a zřízení připojení s ostatními uzly.
- Implementace protokolu pro transport témat (TCPROS a UDPROS). Uzly mezi sebou navazují připojení dohodnutým protokolem. Nejobecnějším protokolem je TCPROS, který využívá přetrvávajícího připojení prostřednictvím socketu TCP/IP.
- API příkazové řádky. Každý uzel musí podporovat mapovací argumenty příkazové řádky, což umožňuje konfiguraci jmen v rámci uzlu za běhu programu.

Každý uzel, stejně tak jako uzel Master, má svůj identifikátor URI. XMLRPC, server neslouží k přenosu témat ani dat, ale pouze pro sjednání připojení s ostatními uzly a komunikace s uzlem Master. Je vytvořen a spravován v rámci knihoven ROS klienta, ale obecně není viditelný pro uživatele těchto knihoven. Server může být vázán na jakýkoliv port hosta na kterém uzel běží.

XMLRPC server nabízí slave API, které uzlu umožní příjem aktualizacích volání od uzlu Master. Tato volání obsahují název tématu a seznam identifikátorů pro uzly, které zasílají data na dané téma. Rovněž přijímá volání uzlů, které vyžadují připojení k určitému tématu. Přenosy tématu jsou navázány, když si přijímací uzel vyžádá připojení k tématu prostřednictvím XMLRPC serveru publikujícího uzlu, následně zašle seznam podporovaných protokolů. Publikující uzel zvolí vhodný protokol ze zaslaného seznamu a obratem zašle nezbytné údaje nastavení pro zvolený protokol (např. IP adresu a port). Přijímací uzel nakonec naváže nezávislé připojení na základě dohodnutých podmínek. [20]

5.5 Transport témat

Existuje mnoho způsobů přenosu dat po síti, z nichž každý má své výhody i nevýhody odvíjející se především od zamýšleného využití. Protokol TCP je hojně využíván, protože poskytuje jednoduchý komunikační proud. Packety jsou vždy přijímány ve stejném pořadí jako byly odeslány a ztracené packety jsou opětovně posílány, dokud nedojde k jejich úspěšnému přijetí. Tento přístup je vhodný pro stabilní připojení – zejména rozhraním Ethernet. Značné obtíže však mohou nastat je-li přenosovým rozhraním slabá bezdrátová síť, pro kterou je daleko vhodnější protokol UDP. Je-li na jediné podsíti více přijímacích uzlů, nejvhodnějším způsobem komunikace s jediným vysílacím uzlem je jednotné zasílání dat právě s využitím protokolu UDP. [21]

5.6 Navázání připojení k tématu

Úkony nezbytné k navázání komunikace mezi dvěma uzly:

- 1) Přijímací uzel zjistí, který název tématu použít na základě mapovacích argumentů.
- 2) Vysílací uzel zjistí, který název tématu použít na základě mapovacích argumentů.
- 3) Registrace přijímacího uzlu u uzlu Master (XMLRPC).
- 4) Registrace vysílacího uzlu u uzlu Master (XMLRPC).
- 5) Uzel Master informuje přijímací o vysílacím uzlu (XMLRPC).
- 6) Přijímací uzel obeznámí uzel vysílací o požadavku na připojení k tématu a dohodnutí přenosového protokolu (XMLRPC).
- 7) Vysílací uzel zašle uzlu přijímacímu informace o nastavení pro zvolený přenosový protokol (XMLRPC).
- 8) Přijímací uzel se připojí k uzlu vysílacímu s využitím dohodnutého protokolu (TCPROS, UDPROS).

Ve výchozím stavu je připojení služeb bezstavové (*stateless*), to znamená, že žádné informace o připojení jednotlivých uzlů v systému nejsou uchovávány pro budoucí využití. Pro každé připojení, které chce klient navázat, jsou opakovány kroky vyhledávání služby u ROS mastera a výměny požadavku/odezvy prostřednictvím nového připojení. Bezstavový přístup je obecně robustnější, neboť umožňuje snadný restart připojení uzlů, což ovšem může být výkonnostně problematické při častém opakovaném volání stejné služby [22]. ROS umožňuje přetrvávající připojení ke službě, poskytující vysoký proud dat při opakovaném volání služeb. V tomto případě připojení mezi klientem a službou je udržováno otevřené, aby klient mohl pokračovat v zasílání požadavků pomocí navázaného připojení. V případě těchto přetrvávajících připojení je nutno dát si pozor, neboť objeví-li se nový zprostředkovatel služby, již probíhající přetrvávající připojení tím nejsou zrušena. Podobně selže-li navázané přetrvávající připojení, automaticky neprobíhá žádný pokus o opětovné navázání.

6 NÁVRH STRUKTURY ŘÍDÍCIHO SYSTÉMU

Úkol plynoucí ze zadání práce je návrh řídicího systému s využitím frameworku ROS, který umožňuje, aby druhé a další vozidla v konvoji byla řízena automaticky vestavěným řídicím systémem, na základě bezdrátové komunikace a údajů ze vstupních periférií. Řídicí systém má dále umožnit integraci dalších sensorů do budoucích aplikací, což je umožněno možností bezdrátové komunikace, užitečnými nástroji frameworku ROS a řady periférií, kterým zvolený jednodeskový počítač disponuje.

Jako vhodný hardware pro náš řídicí systém jsme zvolili jednodeskový počítač Raspberry Pi 3, který jsme vybavili operačním systémem Ubuntu, konkrétně verzí MATE, která se nabízí ve verzi určené právě pro Raspberry Pi 3.

6.1 Instalace operačního systému

Deskový počítač Raspberry Pi 3 je prodáván bez jakéhokoliv předem nainstalovaného operačního systému a je čistě na uživateli, jakým z řady kompatibilních systémů jej vybaví. V našem případě je takzvaný obraz instalačního disku ISO dostupný ke stažení z oficiálních webových stránek ubuntu-mate.org v několika verzích. Mezi základní verze patří nejnovější, momentálně *State-of-the-Art* verze označená 17.04 prozatím ve stádiu beta, novější verze ve finálním provedení (16.10) a verze s dlouhodobou podporou s označením 16.04.2 LTS. Pro naše účely jsme tedy zvolili právě verzi s dlouhodobou podporou, neboť slibuje stabilní chod a vyladěnost. Pro instalaci je potřeba vytvoření tzv. spouštěcího disku. Může se jednat o disk USB, ale stejně tak lze použít kartu microSD. Existuje řada programů třetích stran, které jeho tvorbu snadno umožňují. Využili jsme tedy vlastností programu Rufus.

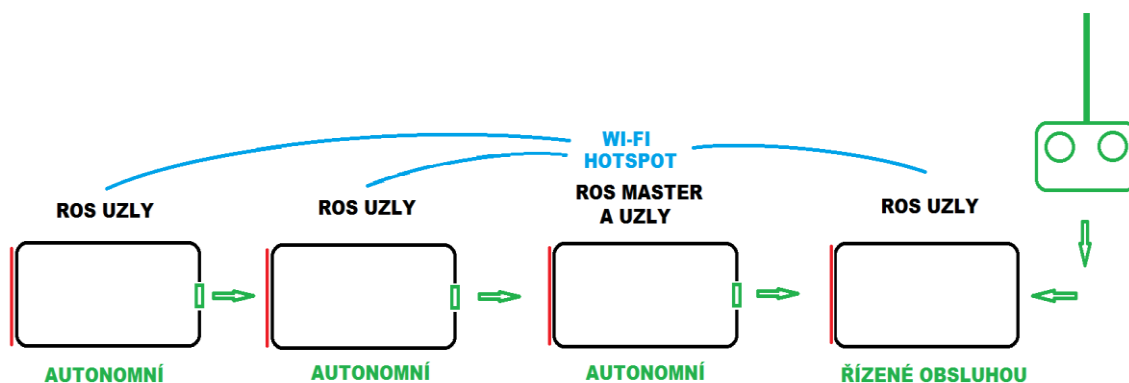
Takto připravený disk využívající souborový systém FAT, nebo FAT32, přítomný při spuštění počítače pak lze snadno vybrat z nabídky systému BIOS, respektive UEFI. Není-li přítomen žádný operační systém, průvodce instalace se zobrazí automaticky a po jejím dokončení je plně funkční operační systém připraven k použití.

6.2 Příprava systému ROS

Dalším nezbytným krokem ve funkčním prostředí systému Ubuntu je instalace samotného frameworku ROS. Jak již z principu práce v tomto systému vyplývá, instalace spočívá především v řadě příkazů zadávaných do příkazové řádky. K instalaci je nutné mít práva administrátora (*superuser*). Tato lze snadno získat příkazem *sudo passwd* a následnou volbou hesla. Před samotnou instalací je potřeba nastavit prostředí konfigurací repositářů, nastavením klíčů a aktualizací indexů správce balíčků Debian. Samotná instalace je dostupná v několika provedeních. Plná verze, která kromě základní funkcionality frameworku ROS obsahuje řadu nástrojů, jako je *rviz*, knihovny základních robotických funkcí a 2D/3D simulační nástroje. Samotná instalace je víceméně plně automatická doprovázená pouze výpisem informací v konzoli. Předtím než je možné začít používat framework ROS, je potřeba inicializace nástroje *rosdep*, který umožňuje snadnou instalaci systémových závislostí nezbytných pro běh některých komponentů.

6.3 Vstupní periferie

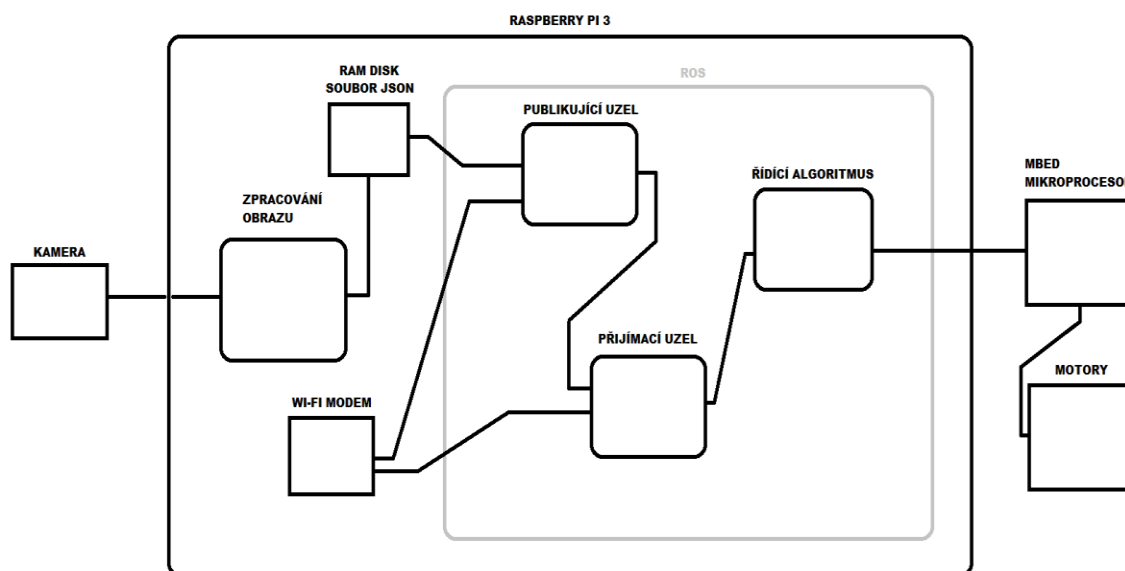
Hlavní vstupní periferií je kamera připojená prostřednictvím rozhraní USB 2. Úkolem webkamery je snímání prostoru před vozidlem. Data z kamery jsou dále zpracována programem strojového vidění, napsaného v jazyce C využívajícího standardu OpenCL. Jeho výstupem jsou informace o poloze předem definované značky, umístěné na zadní straně každého vozidla v konvoji, konkrétně úhel vybočení a relativní vzdálenost značky od kamery.



Obr. 8: Znázornění principu řízení a bezdrátové komunikace jednotlivých členů konvoje.

6.4 Meziprocesová komunikace

Informace získané z kamery jsou ukládány do souboru ve formátu JSON, který je z důvodu rychlosti čtení umístěn v mnohem rychlejší operační paměti, čehož jsme dosáhli pomocí nástroje RAM Disk. Uložiště na paměťové kartě by k tomuto účelu z důvodu nižších přenosových rychlostí nebylo příliš vhodné.



Obr. 9: Abstraktní schéma řídicího systému autonomního vozidla.

6.5 Implementace frameworku ROS

Při návrhu řídicího systému s využitím frameworku ROS se nabízí programovací jazyk C++ a Python, z nichž každý má své výhody i nevýhody. V případě C++ se jedná o programovací jazyk kompilovaný, přeložený do strojového jazyka pouze jednou před jeho používáním. Z těchto vlastností plyne relativně vysoký výkon, ale také obtíže při návrhu a ladění programu spojené s nutností opětovné kompilace zdrojového kódu při každé provedené úpravě. Při návrhu řídicího systému však volba padla na nástroje a výhody jazyka Python, který je ve své podstatě jazykem interpretovaným. Tento přístup znamená, že překlad zdrojového kódu do strojového jazyka probíhá automaticky těsně před každým spuštěním, tento způsob není natolik efektivní a výkonný jako jazyky kompilované. V dnešní době, kdy není nouze o slušný výpočetní výkon, se stále častěji dává přednost právě jazykům interpretovaným, umožňujícím rapidní vývoj a testování. Konstrukce frameworku ROS umožňuje, že jednotlivé uzly spolu komunikují bez rozdílu nezávisle na jazyce použitém k jejich tvorbě.

6.5.1 Publikující uzel

Prvním prvkem bezprostředně využívajícím vlastnosti zmíněného frameworku je uzel, jehož úlohou je získání dat poskytovaných softwarem strojového vidění ze souboru. Námi požadovaná data jsou reprezentována nejaktuálnějším řádkem souboru JSON.

```

17  def talker():
18      pub = rospy.Publisher('chatter', String, queue_size=1)
19      rospy.init_node('talker', anonymous=True)
20      rate = rospy.Rate(30) # (hz)
21      while not rospy.is_shutdown():
22          if os.access('/home/pi/ramdisk/output.json', os.W_OK):
23              f = open('/home/pi/ramdisk/output.json', 'r')
24              data = f.readline()
25              f.close()
26          else:
27              print '--- NO FILE OR ACCESS DENIED ---'
28
29          pub.publish(data)
30          SendOverWifi(data)
31          rate.sleep()
    
```

Obr. 10: Ukázka funkce zajišťující komunikaci mezi softwarem strojového vidění a ostatními prvky řídicího systému pomocí nástrojů frameworku ROS a jazyka Python.

Z důvodu zamezení kolizí čtení a zápisu dvěma nezávislými softwarovými komponenty do jediného souboru, program nejprve ověří, zda je soubor dostupný pro čtení a následně řádek s údaji uloží do proměnné data jako řetězec. Údaje jsou dále publikovány bezdrátově jednak pomocí nástrojů knihovny rospy, jsou tedy dostupné pro libovolný programový uzel na jakémkoliv vozidle v konvoji. Pro případ potřeby komunikace se zařízením, na kterém není přítomna instalace ROS, je uzel vybaven funkcí využívajících knihovny socket. Knihovna socket poskytuje univerzální nástroj pro sdílení, zasílání dat přímo na zvolený port a IP adresu příjemce, kde mohou být přijímána a zpracovávána libovolným jiným operačním systémem a programovacím jazykem.

6.5.2 Přijímací uzel

Dalším funkčním uzlem v rámci jediného vozidla je přijímací uzel. Jeho úkolem je jednak příjem dat zasílaných publikujícím uzlem a následná extrakce konkrétních hodnot úhlu odklonu sledovaného vozidla a jeho vzdálenosti.

```
1  #!/usr/bin/env python
2  import rospy
3  import json
4  import serial
5  from std_msgs.msg import String
6
7  def callback(data):
8      try:
9          parsed_json = json.loads(data.data)
10         Angl = parsed_json['A']
11         Dist = parsed_json['D']
12         print '\nAngl: %s\nDist: %s\n' % (Angl, Dist)
13         Controller(Angl, Dist)
14         print '\nAngl: %s\nDist: %s\n' % (Angl, Dist)
15
16     except ValueError:
17         print 'NO DATA'
```

Obr. 11: Funkce zpracování řídicích dat a předávání algoritmu trasování.

Uzel rovněž implementuje možnost příjmu dat ostatních vozidel a příkazů zasílaných bezdrátově a poskytuje tak možnost kooperace a komunikace jednotlivých členů konvoje stejně tak, jako příkazy z počítačů sloužících k ovládání konvoje. Umožňuje tak například okamžité zastavení konvoje v situacích, které to vyžadují. Získané údaje jsou dále směřovány do algoritmu řízení reprezentovaném samostatnou programovou funkcí *Controller*.

K dispozici jsou dva přístupy trasování. Prvním, který zajišťuje přímé sledování značky napsaným jako součást řídicího systému a sofistikovanějším algoritmem řízení, kterým se tato práce blíže nezabývá. Smyslem přímého sledování bylo umožnění základní funkcionality během vývoje a dále umožňuje demonstrovat rozdíl funkcionality prostého sledování a již zmíněného algoritmu trasování, který je hlavní náplní jiné bakalářské práce. Výstupy trasovacích funkcí jsou následně zasílány přes sériovou linku, za využití prostředků knihovny *serial*.

```

47     def SendToHW(SteeringOut, PowerOut):
48         z = int(SteeringOut)
49         y = int(PowerOut)
50         i = '//A{0}.{1}&'.format(str(z), str(y))
51         ser = serial.Serial(
52             port='/dev/ttyUSB0',
53             baudrate=115200,
54             timeout = 1
55         )
56         ser.write(i)

```

Obr. 12: Komunikace mezi algoritmem řízení a mikroprocesorem MBED prostřednictvím sériové linky

6.6 Inicializace

Pro požadovanou funkcionalitu je po připojení deskového počítače ke zdroji elektrické energie a dokončení spouštění operačního systému nezbytné spuštění množství softwarových nástrojů. Ruční spouštění je zdlouhavé a ve finální verzi řídicího systému musí probíhat automaticky. Příkazy jsou tedy sepsány v *bash* skriptu, k jehož automatickém spuštění využíváme vestavěné funkce operačního systému.

```
1  #!/bin/bash
2  cd /home/pi/workspace/TRACKMARKER/Release/
3  ./TRACKMARKER &
4  source /opt/ros/kinetic/setup.bash
5  source ~/catkin_ws/devel/setup.bash
6  roslaunch bakalarka nodes.launch
```

Obr. 13: Ukázka skriptu implementovaného v jazyce Bash, který postupně spouští jednotlivé funkce řídicího systému.

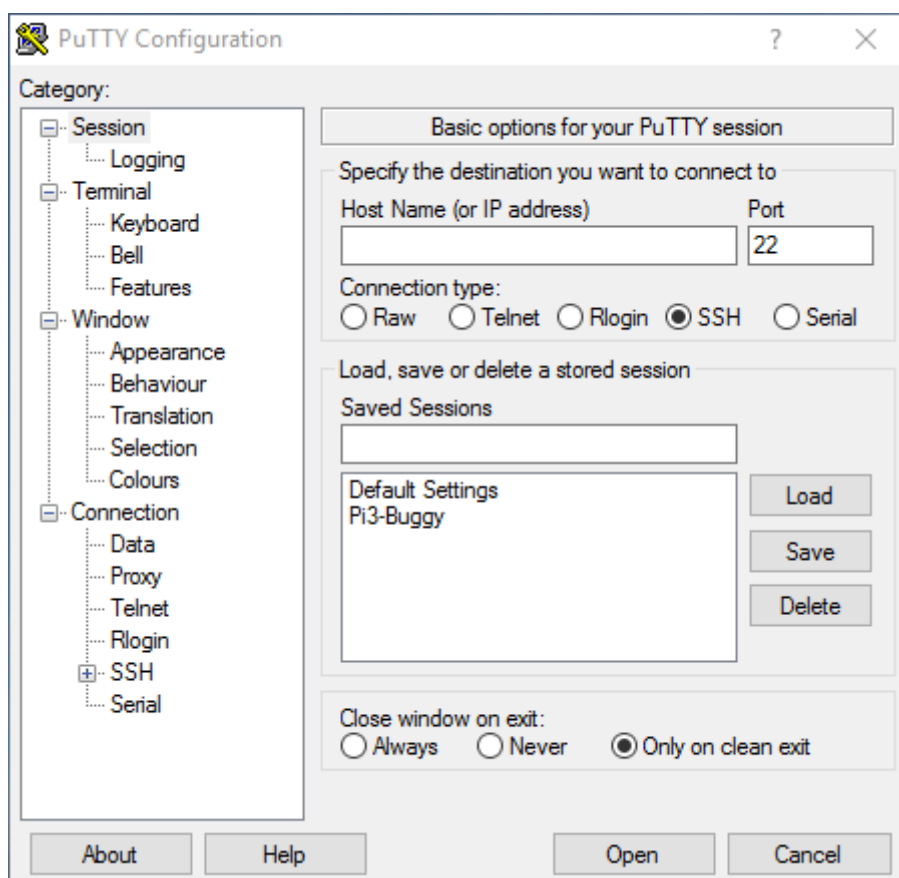
Spouštěcí skript obsahuje specifikaci kompilátoru následovanou příkazem spouštějícím software strojového vidění *./TRACKMARKER* a příkazem *roslaunch bakalarka nodes.launch*, který umožňuje spuštění vícero uzlů tvořících jeden funkční celek lokálně, nebo vzdáleně prostřednictvím SSH. *Roslaunch* pracuje s příkazy ve formátu XML sepsanými v souboru *nodes.launch*, které popisují uzly, jenž mají být spuštěny, popřípadě parametry, které mají být nastaveny a ostatní atributy spuštění sbírky ROS uzlů. Výhodou je, že příkaz *roslaunch* automaticky spustí uzel master, není-li již spuštěn.

```
1  <launch>
2  <master auto="start"/>
3    <!-- a respawn-able subscriber node -->
4    <node name="subscriberOLD" pkg="bakalarka" type="listener" respawn="true" />
5    <!-- a respawn-able publisher node -->
6    <node name="publisher" pkg="bakalarka" type="talker" respawn="true" />
7  </launch>
```

Obr. 14: Náhled na atributy automatického spuštění uzlů v souboru *nodes.launch*.

6.7 Přístup a editace

V průběhu návrhu se nabízí možnost využití portu HDMI pro připojení deskového počítače k externímu monitoru a pohodlné ovládání USB myši a klávesnicí. V průběhu testování a ladění je z podstaty mobility projektu nemožné drátové připojení periférií, a proto s výhodou využijeme přítomnosti Wi-Fi modulu, kterým je každé vozidlo v konvoji vybaveno. Open Source nástroj PuTTY a znalost IP adresy konkrétního vozidla poskytuje možnost připojení a přihlášení do počítače v prostředí příkazové řádky. Prohlížení adresářů a editace kódu je možná běžnými příkazy linux shellu a základními nástroji jako je správce souborů *Midnight Commander*, nebo miniaturní textový editor *nano*.



Obr. 15: Okno programu PuTTY umožňujícího vzdálený přístup prostřednictvím SSH.

7 ZÁVĚR

V úvodní části práce v kapitole 2 proběhlo pojednání o využití, výhodách a úskalích semiautonomních konvojů obecně a seznámení se s pojmem *platooning* v dopravě. Dále byl zmíněn pojem inteligentních transportních systémů, které de facto zahrnují již zmíněné pojmy.

Účelem bakalářské práce vyplývajícím z jejího zadání byl návrh struktury řídicího systému pro vozidlo pohybující se autonomně v konvoji realizovaného na jednodeskovém počítači Raspberry Pi 3. Pro vhodný návrh řídicího systému bylo nezbytné věnovat pozornost jeho hardwarovým a komunikačním možnostem rozebraným v kapitole 3.2, která následuje po kapitole pojednávající o obecných možnostech a využití tohoto jednodeskového počítače.

Jak bylo řečeno v kapitole pojednávající o použitém jednodeskovém počítači, jeho dostatečný výkon umožňuje instalaci a plnohodnotný běh řady operačních systémů založených převážně na systému Linux, kterému je věnována kapitola 4. V našem případě bylo s výhodou využito distribuce operačního systému Ubuntu ve verzi MATE popsané v následující kapitole. Využitými nástroji jsou programovací jazyky Python, Bash a formát JSON, které jsou probrány v kapitole 4.3. Zadání bakalářské práce rovněž vyžaduje využití softwarových nástrojů frameworku ROS, kterému je věnována největší část práce a jeho základní prvky a vlastnosti jsou detailněji probrány v podkapitolách 5.1 až 5.6.

S hardwarovou platformou vybavenou operačním systémem může začít samotný návrh struktury řídicího systému, kterému je věnována závěrečná kapitola 6. Nejprve je pozornost věnována samotné přípravě operačního systému, frameworku ROS a ostatních nástrojů následovaná popisem implementace a jejich využití při návrhu řídicího systému a konkrétního využití. Navržený řídicí systém byl ve finále úspěšně experimentálně otestován.

8 SEZNAM POUŽITÉ LITERATURY

- [1] THRUN Sebastian, BURGARD Wolfram, and FOX Dieter. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series). Intelligent robotics and autonomous agents. The MIT Press, August 2005.
- [2] CHOSET Howie, LYNCH Kevin M., HUTCHINSON Seth, KANTOR George, BURGARD Wolfram, KAVRAKI Lydia E., and THRUN Sebastian. Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents). The MIT Press, June 2005.
- [3] *Self-driving cars*: <https://www.theguardian.com/technology/2016/apr/07/convoy-self-driving-trucks-completes-first-european-cross-border-trip> [online], c2017 [cit. 2017-01-01]. Dostupné z: <https://www.theguardian.com/technology/2016/apr/07/convoy-self-driving-trucks-completes-first-european-cross-border-trip>.
- [4] Scania lines up for platooning trials. In: [Www.scania.com](http://www.scania.com) [online]. 2017 [cit. 2017-05-05]. Dostupné z: <https://www.scania.com/group/en/scania-lines-up-for-platooning-trials/>.
- [5] *Creating next generation mobility: What is Truck Platooning?* [online]. [cit. 2017-01-08]. Dostupné z: <https://www.eutruckplatooning.com/About/default.aspx>.
- [6] *Raspberry Pi 3 Model B: The third generation Raspberry Pi*. [online], c2014. [cit. 2017-03-08]. Dostupné z: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b>.
- [7] Raspberry Pi 3. In: [Www.raspberrypi.org](http://www.raspberrypi.org) [online]. [cit. 2017-05-05]. Dostupné z: https://www.raspberrypi.org/app/uploads/2016/02/IMG_4090.jpg.
- [8] *Benchmarks / Tech: Broadcom VideoCore-IV* [online], c2016. [cit. 2017-04-10]. Dostupné z: <https://www.notebookcheck.net/Broadcom-VideoCore-IV.116484.0.html>.
- [9] *The official Raspberry Pi magazine: Raspberry Pi 3 is out now! Specs, benchmarks & more* [online]. [cit. 2017-04-10]. Dostupné z: <https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/>.
- [10] *Difference Between Linux and UNIX: What is the difference between Linux and UNIX operating systems?* [online], Sep. 8, 2016 [cit. 2017-04-10]. Dostupné z: <https://www.cyberciti.biz/faq/what-is-the-difference-between-linux-and-unix/>.
- [11] News for the Open Source Professional: What is Linux? *Linux* [online], c2016. [cit. 2017-04-10]. Dostupné z: <https://www.linux.com/what-is-linux>.
- [12] *Ubuntu MATE: For a retrospective future* [online], c2017. [cit. 2017-03-05]. Dostupné z: <https://ubuntu-mate.org/>.
- [13] What is secure shell: Secure Shell (SSH). *SearchSecurity.TechTarget* [online]. [cit. 2017-04-12]. Dostupné z: <http://searchsecurity.techtarget.com/definition/Secure-Shell>.
- [14] Secure Shell. In: *Wikipedia* [online]. 2017 [cit. 2017-05-15]. Dostupné z: https://en.wikipedia.org/wiki/Secure_Shell.
- [15] Zen of Python: Principles. *Wikipedia* [online], c2017. [cit. 2017-02-22]. Dostupné z: https://en.wikipedia.org/wiki/Zen_of_Python.
- [16] Bash (Unix shell). In: *Wikipedia* [online]. 2017 [cit. 2017-05-15]. Dostupné z: [https://en.wikipedia.org/wiki/Bash_\(Unix_shell\)](https://en.wikipedia.org/wiki/Bash_(Unix_shell)).

- [17] *JSON: JSON encoder and decoder* [online], Mar. 26, 2017 [cit. 2017-03-10]. Dostupné z: <https://docs.python.org/3/library/json.html>.
- [18] *ROS/Technical Overview: Master* [online]. [cit. 2017-04-10], c2014. Dostupné z: <http://wiki.ros.org/ROS/Technical%20Overview>.
- [19] ROS / Documentation: Parameter Server. *Technical Overview* [online], c2013. [cit. 2017-04-12]. Dostupné z: <http://wiki.ros.org/Parameter%20Server>.
- [20] *ROS/Technical Overview: Node* [online]. [cit. 2017-04-10], c2014. Dostupné z: <http://wiki.ros.org/ROS/Technical%20Overview>.
- [21] *ROS/Technical Overview: Topic Transports* [online], c2014. [cit. 2017-04-10]. Dostupné z: <http://wiki.ros.org/ROS/Technical%20Overview>.
- [22] *ROS/Technical Overview: Establishing a topic connection* [online], c2014. [cit. 2017-04-10]. Dostupné z: <http://wiki.ros.org/ROS/Technical%20Overview>.
- [23] *ROS/TCPROS: TCPROS* [online], c2013. [cit. 2017-04-10]. Dostupné z: <http://wiki.ros.org/ROS/TCPROS>.
- [24] *ROS/Technical Overview: Persistent service connections* [online], c2014. [cit. 2017-04-10]. Dostupné z: <http://wiki.ros.org/ROS/Technical%20Overview>.
- [25] *ROS/Technical Overview: Persistent service connections* [online], c2014. [cit. 2017-04-10]. Dostupné z: <http://wiki.ros.org/ROS/Technical%20Overview>.
- [26] *ROS/Concepts* [online], c2016. [cit. 2017-04-10]. Dostupné z: <http://wiki.ros.org/ROS/Concepts>.

9 SEZNAM PŘÍLOH

CD-R: Elektronická kopie dokumentu ve formátu PDF.

