

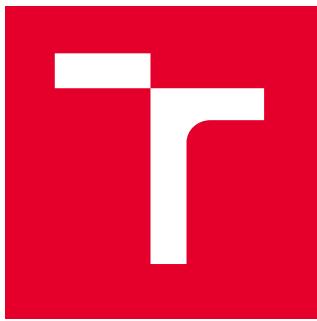
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2022

Bc. David Lindtner



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

**FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ**

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

**SIMULACE BEZPILOTNÍCH LETADEL VE VIRTUÁLNÍM
PROSTŘEDÍ**

SIMULATION OF UNMANNED AIRCRAFTS IN A VIRTUAL ENVIRONMENT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. David Lindtner

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Petr Gábrlík, Ph.D.

BRNO 2022



Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. David Lindtner

ID: 196815

Ročník: 2

Akademický rok: 2021/22

NÁZEV TÉMATU:

Simulace bezpilotních letadel ve virtuálním prostředí

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je simulace misí jednoho či více autonomních bezpilotních letadel v prostředí ROS2/Gazebo.

1. Seznamte se s frameworkm ROS2 (Robot Operating System 2) a simulačním prostředím Gazebo, naučte se je používat na systému Linux (Ubuntu).
2. Prozkoumejte open-source projekty pro řízení bezpilotních letadel nabízející integraci do simulačního prostředí.
3. Nastudujte a navrhněte jak správně ve zvoleném nástroji či frameworku implementovat mise autonomních letadel.
4. Zobecněte bod č. 3 tak, aby bylo možné simulovat mise s více bezpilotními letadly najednou.
5. Demonstrujte funkčnost simulace na misích jako je let po waypoints, sledování dynamického objektu, let podle dat z doplňkových senzorů atp.
6. Výsledky práce průběžně verzujte v GIT repositáři, který bude respektovat běžnou strukturu ROS projektů a umožní rychlé zprovoznění simulátoru na kompatibilních zařízeních či integraci do jiných ROS projektů.

DOPORUČENÁ LITERATURA:

PYO, YoonSeok, HanCheol CHO, RyuWoon JUNG a TaeHoon LIM. ROS Robot Programming. Republic of Korea: ROBOTIS Co., 2017. ISBN 979-11-962307-1-5.

Termín zadání: 7.2.2022

Termín odevzdání: 18.5.2022

Vedoucí práce: Ing. Petr Gábrlík, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.

předseda rady studijního programu

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Semestrální práce se zabývá problematikou simulace misí bezpilotních letadel ve virtuálním prostředí. Cílem práce je demonstrovat jednoduchou autonomní leteckou misi v simuovaném prostředí Gazebo - ROS2. Práce dává důraz na stabilitu softwarového řešení a vnitřní komunikace mezi kritickými komponenty letecké mise.

KLÍČOVÁ SLOVA

Autonomní mise, Robotika, Dron, Pixhawk, PX4, Autopilot, uORB, DDS, ROS2, Gazebo, Simulace

ABSTRACT

The semester thesis deals with the issue of simulation of drone missions in a virtual environment. The aim of the work is to demonstrate a elementary autonomous air mission in the simulated environment Gazebo - ROS2. The work emphasizes the stability of the software solution and internal communication between critical components of the air mission.

KEYWORDS

Autonomous mission, Robotics, Drone, Pixhawk, PX4, Autopilot uORB, DDS, ROS2, Gazebo, Simulation

LINDTNER, David. *Simulace bezpilotních letadel ve virtuálním prostředí*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2022, 65 s. Diplomová práce. Vedoucí práce: Ing. Petr Gábrlík, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora: Bc. David Lindtner

VUT ID autora: 196815

Typ práce: Diplomová práce

Akademický rok: 2021/22

Téma závěrečné práce: Simulace bezpilotních letadel ve virtuálním prostředí

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno

.....
podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. XXX YYY, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Poďakovanie Milošovi Cihlářovi za gazebo world

Obsah

Úvod	11
1 Topologie řídícího systému dronu	12
1.1 Řídící jednotka Pixhawk	12
1.2 Palubní počítač	13
2 Firmware pro řízení bezpilotních letadel	15
2.1 BetaFlight	15
2.2 INAV	15
2.3 Firmware pro řídící jednotku Pixhawk	15
2.3.1 ArduPilot	16
3 PX4 autopilot	17
3.1 Architektura PX4	17
3.1.1 Systém s PX4 řídící jednotkou	19
3.1.2 Systém s PX4 řídící jednotkou a počítačem pro řízení mise	19
3.2 Licence	20
3.3 Letové režimy PX4	21
3.3.1 Manuální režimy	21
3.3.2 Asistované režimy	21
3.3.3 Autonomní režimy	22
3.4 QGround Control	22
3.4.1 Nastavení konstrukce dronu	25
3.4.2 Plánování bezpilotní mise	25
4 Propojení PX4 a ROS 2	29
4.1 Komunikace pomocí pomocí DDS [®]	29
4.1.1 Struktura komunikace v ROS 2	29
4.1.2 Vnitřní komunikace v PX4	29
4.1.3 MicroRTPS agent a klient	29
4.1.4 DDS [®] / RTPS middleware	30
4.1.5 Základní principy DDS [®]	32
4.1.6 Využití DDS [®] v praxi	35
4.2 Komunikace pomocí MAVLink	36
4.2.1 MAVLink zprávy	36
4.2.2 MAVLink mikroslužby	37
4.2.3 Využití protokolu MAVLink	37
4.2.4 Přemostění ROS 2 a MAVLink	38

5 Simulace ve virtuálním prostředí	40
5.1 SITL simulační prostředí	40
5.1.1 Seznam podporovaných simulátorů	41
5.1.2 Gazebo	41
5.1.3 Ignition Gazebo	42
5.2 Instalace virtuálního prostředí	42
5.2.1 Instalace PX4 simulačního prostředí	42
5.2.2 Instalace ROS 2 Foxy a všech závislostí	43
5.2.3 Komunikace pomocí RTPS	43
5.2.4 Komunikace pomocí MAVLink	45
6 Implementace robotických misí	47
6.1 Offboard flight mode	47
6.2 Struktura robotické mise	47
6.2.1 Základní funkce dronu	48
6.2.2 Struktura řídícího algoritmu	49
6.2.3 Stavový automat pro řízení mise	50
7 Výsledky simulace	54
7.1 Gazebo svět	54
7.2 Let po waypointech	54
7.2.1 Let podle lokálních waypointů	54
7.2.2 Let podle globálních waypointů	54
7.3 Let podle setpointů rychlosti	54
7.4 Mise pro sledování dynamického objektu	55
7.5 Simulace mise s více drony	55
Závěr	58
Literatura	59
Seznam symbolů a zkratek	62
Seznam příloh	63
A Přílohy	64
B Obsah elektronické přílohy	65

Seznam obrázků

1.1	Řídící jednotka Pixhawk 4	13
3.1	Komplexní strukturální architektura PX4 software	18
3.2	Jednoduchý systém založený na řídící jednotce s PX4 firmware	19
3.3	Systém založený na PX4 řídící jednotce a palubním počítači pro řízení mise	20
3.4	Diagram přepínání mezi letovými režimy	23
3.5	Základní obrazovka QGroundControl	24
3.6	Základní obrazovka QGroundControl	25
3.7	Nastavení bezpilotní mise pro průzkum prostředí	26
3.8	Nastavení bezpilotní mise pro skenování konstrukcí	27
3.9	Nastavení bezpilotní mise pro skenování koridoru	28
4.1	Schéma komunikace mezi microRTPS agentem a klientem	31
4.2	Propojení DDS® middlewaru s aplikací a operačním systémem	32
4.3	Základní <i>publish/subscribe</i> struktura DDS® komunikace	34
4.4	Strura MAVLink zprávy	36
5.1	Tok zpráv mezi simulátorem a PX4	40
5.2	Schéma komunikace mezi PX4 a simulátorem	41
6.1	Stavový automat řídící robotickou misi	51
7.1	Řídící struktura systému PX4	55
7.2	Software QGroundControl s dronem v <i>offboard flight mode</i>	56
7.3	Software QGroundControl s dronem v <i>offboard flight mode</i>	56
7.4	Software QGroundControl s dronem v <i>offboard flight mode</i>	57
7.5	Software QGroundControl s dronem v <i>offboard flight mode</i>	57

Seznam tabulek

6.1 Přehled využitých objektů typu publisher, subscriber a client 50

Úvod

DOKONČIŤ

Téma bezpilotních letadel a autonomních leteckých misí v poslední době nabývá na popularitě. S drony se začínáme setkávat v běžném životě, ale také v dalších odvětvích jako jsou armáda, lesnictví, zemědělství a stavitelství. Simulace je vhodným přístupem k vývoji autonomních letadel, protože snižuje finanční nároky a urychluje vývoj a testování.

Tato práce se bude zabývat problematikou simulace misí bezpilotních letadel ve virtuálním prostředí. Cílem práce je seznámit se s simulačním ekosystémem Gazebo/ROS2 a vytvořit a demonstrovat jednoduchou autonomní misi.

Práce bude pojednávat o topologii řídícího systému dronu, jak ze stránky nízkoúrovňového řízení, tak z pohledu palubního počítače pro řízení složitých autonomních misí.

V práci bude popsán firmware řídící jednotky, možnosti nastavení parametrů letu dronu, možnosti komunikace s okolím a hlavně komunikace s ROS 2.

Práce se věnuje instalaci a nastavení simulačního prostředí. Bude zde přiblížen praktický test simulované autonomní mise v prostředí PX4 - Gazebo, v které dron dostává povely z nadřazeného systému palubního počítače na bázi ROS 2.

1 Topologie řídícího systému dronu

Práce se zaobírá v první řadě simulací bezpilotních letadel ve virtuálním prostředí, ale z důvodu možné budoucí realizace projektu je nutné pracovat s konkrétními hardwarovými a softwarovými řešeními.

Topologii řídícího systému dronu jsme rozdělili na dvě části. Palubní počítač řídí bezpilotní misi a řídící jednotka ovládá kritické procesy v dronu. Na trhu existuje ne-přeberné množství různých komponent pro autonomní mise. Na základě existujících projektů, například *Multi Robot Systems Group* z ČVUT [1] jsme zvolili modul Pixhawk jako řídící jednotku. Jako palubní počítač, který ovládá samotnou autonomní misi jsme zvolili jednodeskový počítač Raspberry PI.

S touto kombinací jsme schopni řídit velké množství různých autonomních letadel a vozidel jako jsou:

- dron v různých konfiguracích
- delta VTOL (Vertical Take-Off and Landing)
- letadlo v různých konfiguracích
- vrtulník
- rover
- ponorka

1.1 Řídící jednotka Pixhawk

Pixhawk je otevřený hardwarový projekt, jehož cílem je poskytnout standard pro snadno dostupné a vysoce kvalitní návrhy hardwaru autopilota pro akademické, hobby a vývojářské komunity. [2]

Řídící jednotka Pixhawk se využívá pro řízení kritických procesů v dronu jako jsou *fail safe* funkce, ovládání pohonů, stabilizace a čtení kritických snímačů jako jsou GPS, barometr a IMU (Inertial Measurement Unit). Do jednotky Pixhawk je možné posílat řídící povely z RC vysílačky a je možné do ní nahrát jednoduchou bezpilotní misi pomocí software QGroundControl nebo MissionPlanner.

Existuje několik variant řídící jednotky od značky Pixhawk:

- Pixhawk (výroba byla ukončena)
- Pixhawk 2 (výroba byla ukončena)
- Pixhawk 3 Pro
- Pixhawk 4
- Pixhawk 4 Mini
- Pixhawk 5X
- Pixracer
- Auterion Skynode (průmyslové řešení)

Na obrázku 1.1 je zobrazena řídící jednotka Pixhawk 4. Můžeme si na ní všimnout redundantní porty pro napájení (POWER1, POWER2, USB), sběrnice I2C, SPI, CANbus, UART, S.BUS. Velkou výhodou je nízká hmotnost, která činí jenom 15,8g. [3]



Obr. 1.1: Řídící jednotka Pixhawk 4 [3].

Mezi hlavní výhody řídících jednotek Pixhawk patří kvalitní softwarová podpora, automatické aktualizace firmware pomocí QGroundControl, flexibilita z hlediska připojených hardwareových periferií a silná komunita uživatelů a vývojářů. Jednotka Pixhawk podporuje aj simulaci HITL (*Hardware In The Loop*), což umožňuje test firmwaru a komunikace ještě před samotným vzletem dronu.

1.2 Palubní počítač

Jako palubní počítač se na experimentální letecké mise využívá jednodeskový počítač Raspberry PI. Na trhu existuje množství různých variant tohoto široce rozšířeného

počítače v rozmezí od levného, malého a úsporného *Raspberry PI Pico* až po vysoko výkonný počítač *Raspberry PI 4B*.

Důležitými výhodami pro aplikace v robotických misích jsou integrace na jeden plošný spoj, nízká proudová spotřeba a možnost spuštění linuxových distribucí. Nejběžnější podporované distribuce jsou *Raspberry PI OS*, *Raspbian*, *Ubuntu desktop* a server.

Hlavní důvod pro použití *Raspberry PI* jako palubního počítače je možnost spolehlivé komunikace s firmware pro řízení bezpilotních letadel (*PX4*, *ArduPilot*, ...) přes *ROS 2 topic*, nebo *MAVLink*. Komunikací mezi uvedenými platformami se budeme zabývat v následujících kapitolách.

2 Firmware pro řízení bezpilotních letadel

2.1 BetaFlight

BetaFlight je *open-source* firmware pro řízení letu používaný k létání s multikoptérami a s letadly s pevnými křídly. Firmware se zaměřuje na letový výkon, špičkové funkce a širokou podporu plaforem.

BetaFlight podporuje velkou škálu řídících jednotek, které mají alespoň *STM32F4* procesor. Software pro nastavování parametrů (*BetaFlight Configurator*) běží na Windows, Mac OS, Linux a Android.

Firmware Betaflight podporuje komunikaci s dálkovými ovládači od většiny výrobců, jako jsou FrSky, Graupner, Spektrum, DJI a FlySky. Řídící jednotku s BetaFlight firmware je možné povelovat i z nadřazeného systému pomocí MSP¹. [4]

2.2 INAV

INAV autopilot je odvozený od BetaFlight a zaměřuje se hlavně na autonomní lítání. Podporuje velké množství modelů od závodních a freestyle dronů, přes letadla až pod rovery a jiné kolové vozidla.

Firmware podporuje širokou škálu řídících jednotek od různých výrobců, podporuje zpracování dat ze senzorů a umožňuje ukládání dat o letu a vnitřním stavu dronu v reálném čase (*blackbox*). [8]

Software *INAV Configurator* nabízí plánování *waypoint* mise pro jakýkoliv autonomní model s řídící jednotkou, v které běží INAV firmware.

2.3 Firmware pro řídící jednotku Pixhawk

Existují dvě hlavní větve vývoje firmware pro řídící jednotku Pixhawk. Jednou platformou je open source projekt ArduPilot [5], který podporuje velké množství bezpilotních prostředků a umožňuje vytváření autonomních misí. Druhým projektem je profesionální autopilot PX4 [6]. Velkou výhodou obou projektů je, že jsou do velké části kompatibilní, například software pro pozemní stanice QGroundControl z platformy PX4 může komunikovat s řídící jednotkou s firmware od ArduPilot a

¹MSP (Multiwii Serial Protocol) je komunikační protokol pro posílání informací o telemetrii, dat ze snímačů a ovládání [5]

naopak. Vzájemná kompatibilita je zajištěna využitím stejného komunikačního protokolu MAVLink. Výhoda protokolu MAVLink spočívá aj v tom, že oba firmware můžou komunikovat s ROS (Robot Operating System) přes MAVROS²

2.3.1 ArduPilot

ArduPilot je open source autopilot podporující mnoho typů vozidel, například multikoptéry, tradiční vrtulníky, letadla s pevnými křídly, čluny, ponorky, rovery a další. Projekt ArduPilot má společnou historii s PX4, ale v minulosti se tyto dva projekty oddělili. Proto mají oba projekty do velké míry podobné využití a míří na stejnou platformu.

Zdrojový kód projektu ArduPilot je vyvíjen širokým spektrem profesionálů a nadšenců, takže výhodou je velká komunita poskytující řešení na řadu problémů.

Firmware ArduPilot je založený na ChibiOS RTOS (Real-Time Operating System). ArduPilot taky poskytuje možnost simulace letového kódu jak simulací SITL (Software In The Loop), tak simulací HITL (Hardware In The Loop). [5]

Pro další pokračování v práci jsme zvolili projekt PX4, takže další části budou pojednávat o firmware PX4.

²MAVROS je komunikační ovládač (*driver*) pro komunikaci mezi ROS a autopiloty s MAVLink protokolem

3 PX4 autopilot

PX4 je profesionální autopilot, vyvinutý světovými vývojáři z průmyslu a akademické obce a podporovaný aktivní celosvětovou komunitou. Pohání všechny druhy modelů od závodních a nákladních dronů až po pozemní vozidla a ponorky.

3.1 Architektura PX4

PX4 firmware se skládá ze dvou hlavních vrstev:

- Flight stack (letový zásobník)
 - Letový zásobník je systém pro odhad a řízení letu.
- PX4 middleware
 - PX4 middleware je obecná robotická vrstva, která podporuje jakýkoli typ autonomního robota a poskytuje interní/externí komunikaci a integraci s hardware.
 - Middleware navíc obsahuje simulační vrstvu, která umožňuje spuštění letového kódu PX4 na desktopovém operačním systému a ovládání počítačem modelovaného vozidla v simulovaném prostoru.

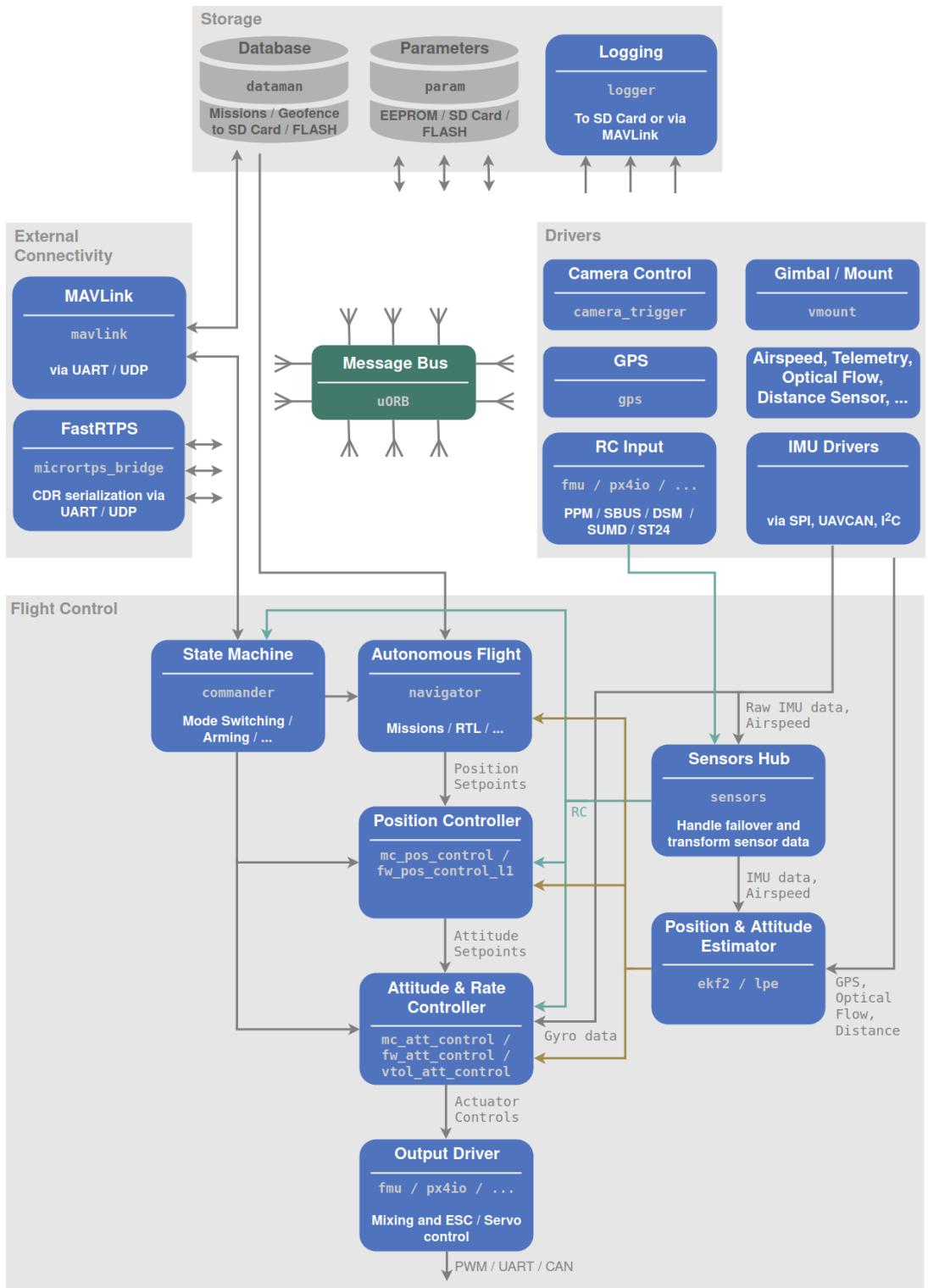
Všechny podporované typy vozidel (drony, letadla, čluny, rovery, ponorky atd.) sdílejí jedinou kódovou základnu, která má následující vlastnosti: [3]

- Veškerá funkčnost je rozdělena na vyměnitelné a opakovatelné komponenty
- Komunikace probíhá pomocí asynchronního předávání zpráv
- Systém se dokáže vypořádat s různou pracovní zátěží

Na obrázku 3.1 je zobrazen diagram, který poskytuje podrobný přehled stavebních bloků PX4. Horní část diagramu obsahuje middlewareové bloky, zatímco spodní část zobrazuje komponenty letového zásobníku (*flight stack*).

Moduly spolu komunikují prostřednictvím sběrnice *publish-subscribe* s názvem *uORB*. Hlavní výhody schématu *publish-subscribe* v PX4 jsou následující:

- Systém je asynchronní a aktualizuje se okamžitě, když jsou k dispozici nová data
- Všechny operace a komunikace jsou plně paralelní
- Systémová komponenta může zpracovávat data odkudkoli (globální datový prostor)

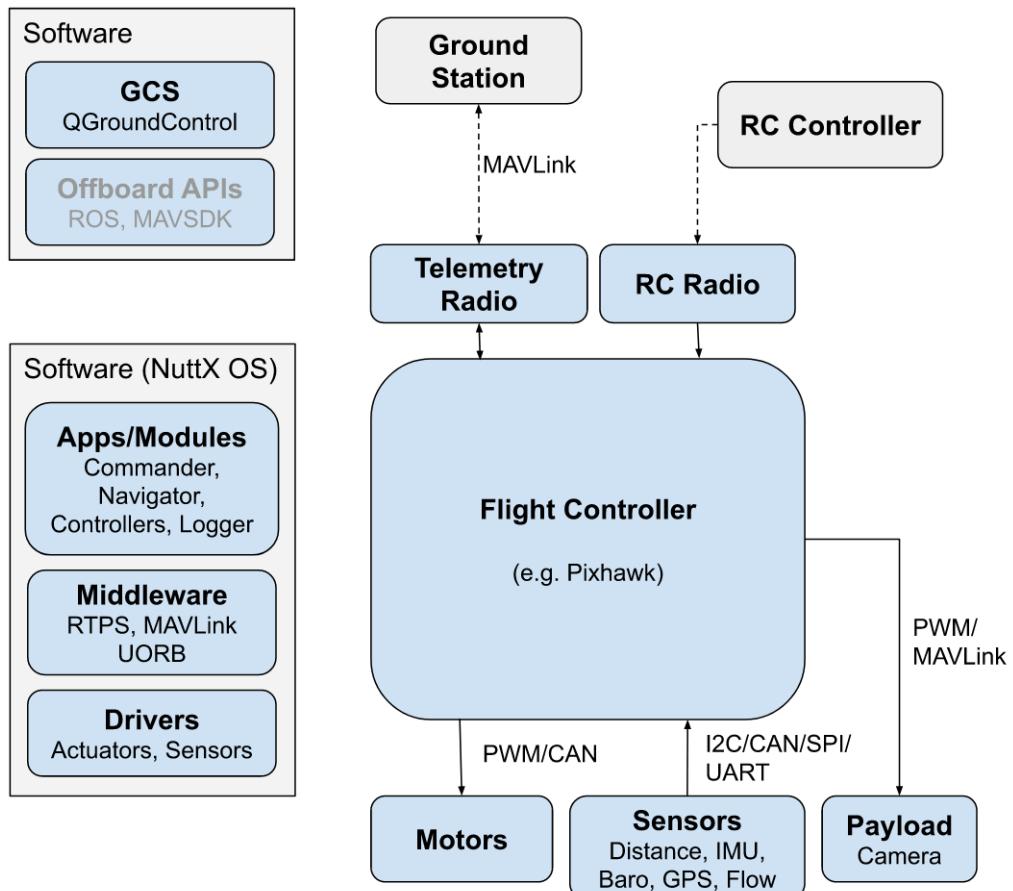


Obr. 3.1: Komplexní strukturální architektura PX4 software [3].

3.1.1 Systém s PX4 řídící jednotkou

Na obrázku 3.2 je naznačená architektura systému založeného na řídící jednotce (*flight controller*). Řídící jednotka zabezpečuje sběr dat ze senzorů, ovládání akčních členů dronu (pohony, serva, ...), řízení dronu na základě dat ze senzorů a povelů z RC rádia a v neposlední řadě stabilizaci dronu.

Do systému PX4 je možné zasílat povely z pozemní stanice (*Ground Station*) přes protokol *MAVlink* pomocí telemetrické rádiové soupravy.

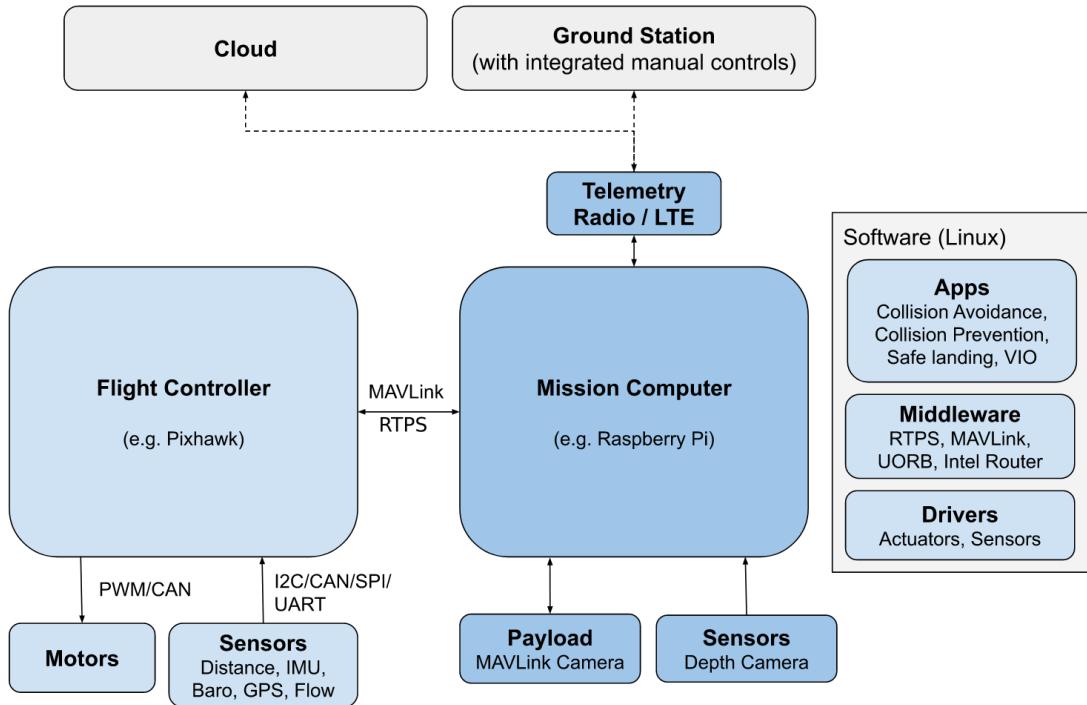


Obr. 3.2: Jednoduchý systém založený na řídící jednotce s PX4 firmware [3].

3.1.2 Systém s PX4 řídící jednotkou a počítačem pro řízení mise

Na obrázku 3.3 je zobrazená architektura systému PX4 založeného na řídící jednotce a palubním počítači pro řízení mise. Řídící jednotka zabezpečuje sběr dat ze senzorů a ovládání akčních členů dronu (pohony, serva, ...). Komunikaci mezi řídící jednotkou a palubním počítačem zabezpečuje protokol MAVlink, nebo RTPS (Real Time Publish Subscribe protocol).

Palubní počítač pro řízení mise poskytuje pokročilé funkce, jako jsou například vyhýbání se objektům. Obvyklým operačním systémem je Linux s ROS (ROS 2) z důvodu, podpory knihoven na předcházení kolizím a z důvodu, že ROS 2 a PX4 využívají pro komunikaci s okolím stejný komunikační DDS[®] (Data-Distribution Service[®]) middleware.



Obr. 3.3: Systém založený na PX4 řídící jednotce a palubním počítači pro řízení mise [3].

3.2 Licence

Kód PX4 je zdarma k použití a úpravě za podmínek permisivní licence *BSD 3-clause license* [7].

Redistribuce a použití ve zdrojové a binární formě, s úpravami nebo bez nich, jsou povoleny za předpokladu, že jsou splněny následující podmínky:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Jméno držitele autorských práv ani jména jeho přispěvatelů nesmí být použita k podpoře nebo propagaci produktů odvozených z tohoto software bez předchozího výslovného písemného povolení.

3.3 Letové režimy PX4

Letové režimy určují chování PX4 autopilota a tím výrazně ovlivňují vlastnosti dronu během letu. Režimy poskytují uživateli různé úrovně pomoci autopilota od automatizace jednoduchých úkolů, jako je vzlet nebo přistání až po komplexní řízení autonomní mise.

Přepínat mezi letovými režimy je možné pomocí pozemní stanice v software QGroundControl, pomocí nadřazeného počítače pro řízení mise, nebo pomocí dálkového ovladače. Obrázek 3.4 zobrazuje logiku přepínání letových režimů v řídící jednotce s firmware PX4. [3]

3.3.1 Manuální režimy

V manuálních režimech má uživatel přímou kontrolu nad dronem pomocí RC ovladače. Pohyb vozidla vždy následuje pohyb páky, ale úroveň nebo typ odezvy se mění v závislosti na letovém režimu. Pro multikoptéry existují 2 manuální režimy:

Manual/Stabilized

Vstupy z RC vysílačky se interpretují jako příkazy o uhlu náklonu, úhlu sklonu a rychlosti stáčení. Příkaz z plynové páky je předán přímo do výstupního směšovače. Autopilot reguluje náklon dronu na nulu, když jsou páky na RC vysílači vycentrovány. V tomto režimu se poloha může měnit vlivem větru.

Acro

Vstupy z RC vysílačky se interpretují jako příkazy o míře naklánění, sklánění a stáčení. Pokud jsou páky na RC vysílačce vycentrovány, náklon multikoptéry se nezmění. Tento režim umožňuje lepší manévrovatelnost dronu, ale zároveň vyžaduje větší letecké zručnosti pilota.

3.3.2 Asistované režimy

Asistované režimy jsou také ovládány uživatelem, ale nabízejí určitou úroveň autonomní pomoci (držení polohy nebo směru). Asistované režimy často značně usnadňují získání nebo obnovení řízeného letu.

Altitude control

Vstupy naklápění, sklonu a stáčení jsou stejné jako při stabilizovaném režimu. Vstup plynové páky na RC vysílači indikuje rychlosť stoupání nebo klesání. Při vycentrované plynové páce RC vysílačky dron udržuje konstantní výšku.

Position control

Vstupy naklápění a sklonu z RC vysílačky ovládají rychlosť otáčení v daných osách. Plynová páka má stejnou funkci jako v *altitude mode*. Z toho vyplývá, že pozice dronu v osách X , Y , Z je při vycentrovaných pákách RC vysílače udržována stabilně i při nepříznivých povětrnostních podmírkách.

3.3.3 Autonomní režimy

Autonomní módy vyžadují od pilota jenom malý, nebo žádný zásah do řízení dronu. Orientace a rychlosti ve všech osách jsou řízeny autopilotem.

PX4 autopilot může ovládat dron v těchto autonomních leteckých režimech: [3]

Take Off

- Dron vystoupá do definované výšky pro vzlet

Land

- Dron přistane na místě, kde byl letový režim aktivován

Hold

- Dron se vznáší na GPS pozici v konstantní výšce

Return to launch

- Dron přistane na místě, z kterého vzlehl

Mission

- Dron vykoná předdefinovanou misi, která byla nahrána do řídící jednotky

Follow Me

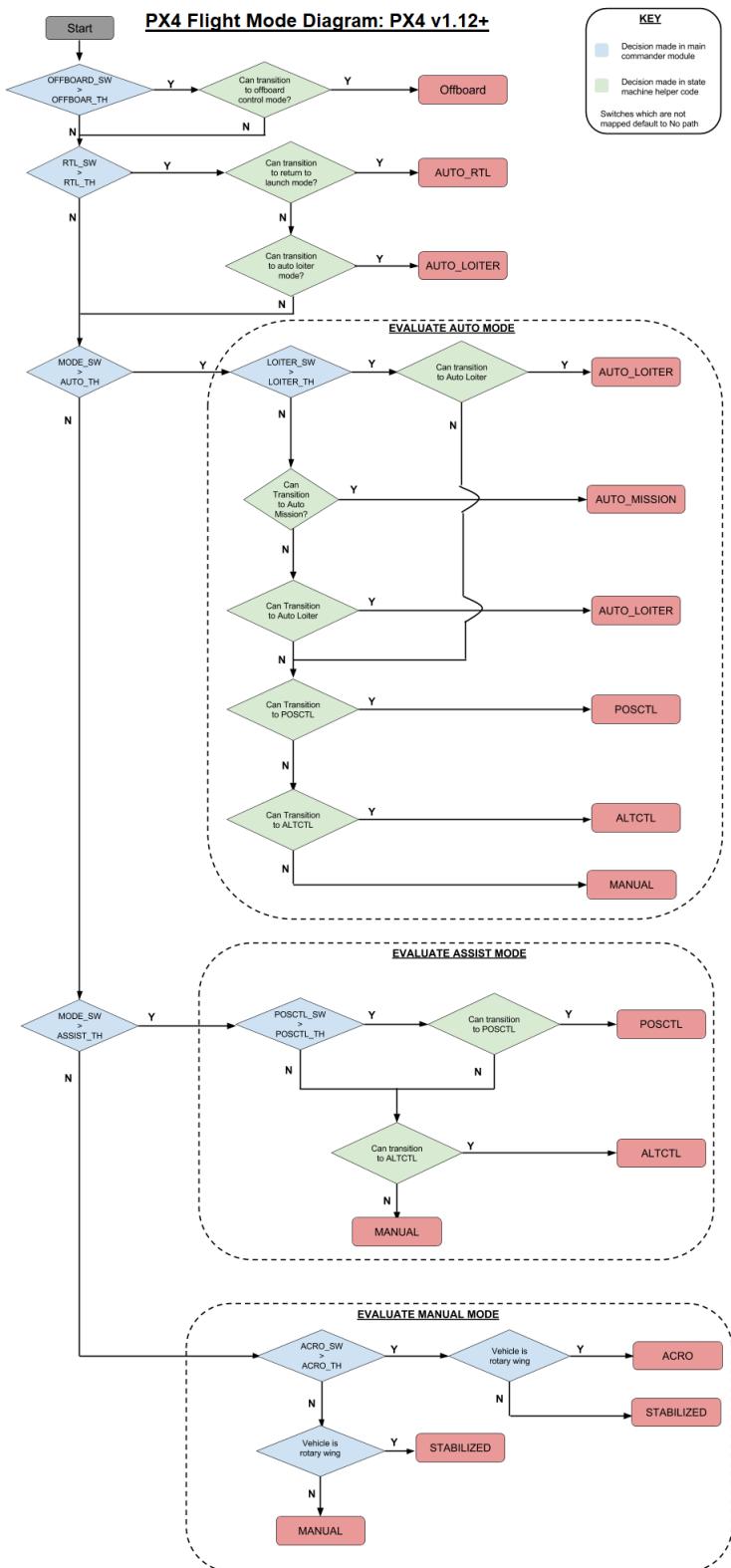
- Dron automaticky následuje osobu se zařízením, v kterém běží software QGroundControl

Offboard

- Dron se řídí nastavenými hodnotami pozice, rychlosti, nebo natočení poskytovanými z nadřazeného systému (přes MAVLink, nebo RTPS)

3.4 QGround Control

QGroundControl je software, který poskytuje plnou kontrolu letu a plánování mise pro jakýkoli dron s podporou MAVLink. Jeho primárním cílem je snadné použití

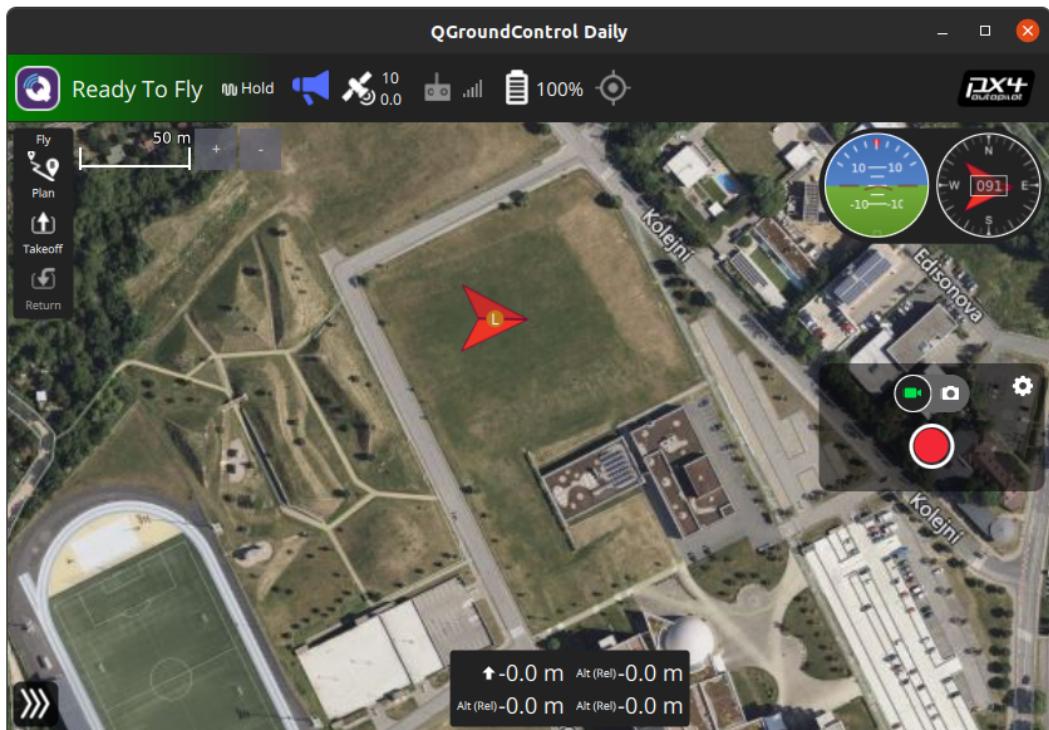


Obr. 3.4: Diagram přepínání mezi letovými režimy. [3]

pro profesionální uživatele a vývojáře. Veškerý kód je open source, takže je možné přispívat a dál ho vyvíjet [9].

Pomocí QGroundControl je možné nahrát nejnovější PX4, nebo ArduPilot firmware do řídící jednotky, nastavit typ konstrukce dronu, ladit parametry regulace a vytvářet autonomní mise pomocí waypointů. Veškeré nastavení dronu a misí je jednoduché a velmi intuitivní.

Obrázek 3.5 zobrazuje základní obrazovku software QGroundControl.



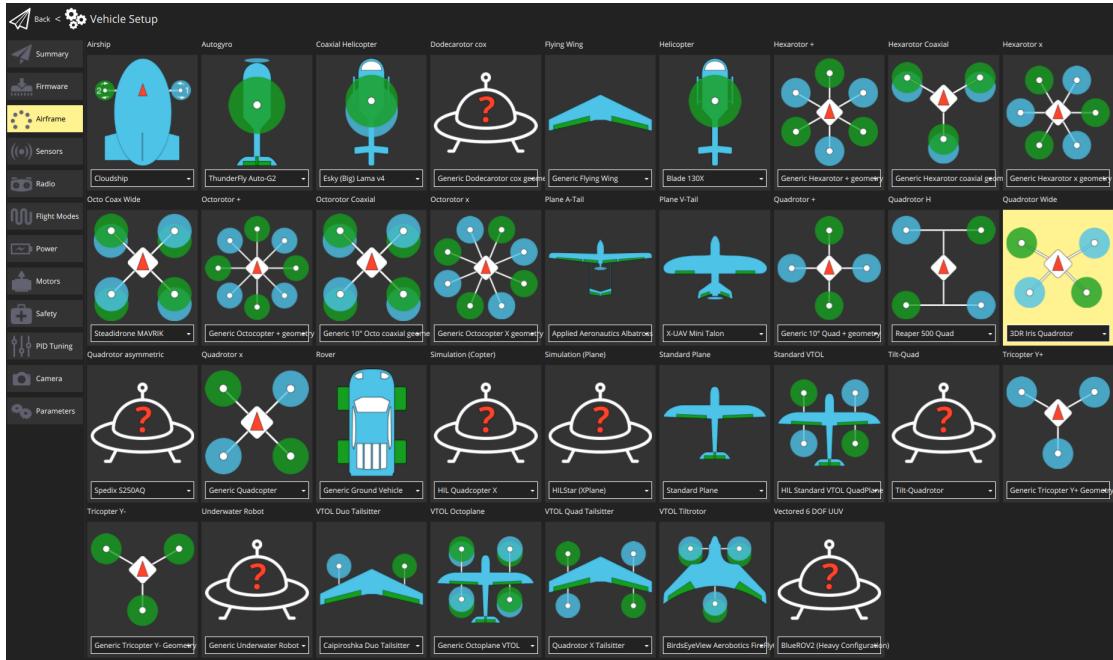
Obr. 3.5: Základní obrazovka QGroundControl.

Na základní obrazovce jsou zobrazeny stavové informace o dronu:

- Relativní výška
- Azimut
- Orientace dronu (pitch, roll)
- RSSI (*Received Signal Strength Indication*)
- Počet připojených GPS satelitů
- Procento nabití akumulátoru
- Letový mód

3.4.1 Nastavení konstrukce dronu

Systém QGroundControl podporuje velké množství typů konstrukcí dronů, letadel, roverů a ponorek. Obrázek 3.6 zobrazuje všechny podporované typy zařízení, které dokáže systém PX4 ovládat.



Obr. 3.6: Základní obrazovka QGroundControl.

3.4.2 Plánování bezpilotní mise

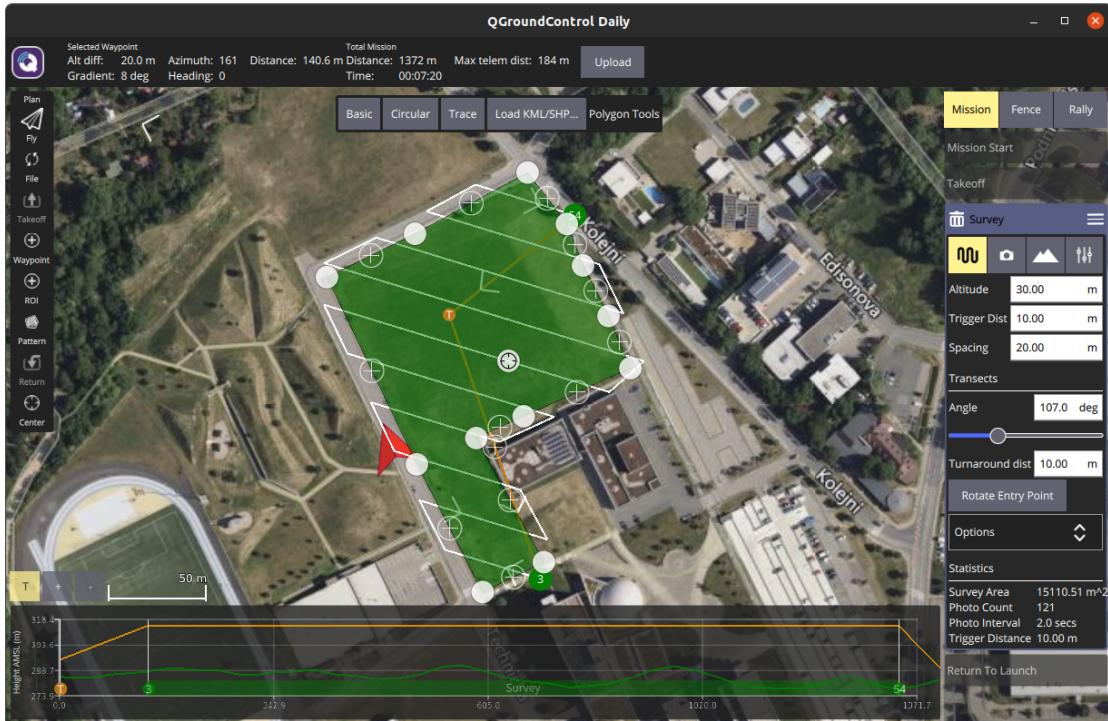
Software QGroundControl nabízí možnost intuitivního plánování bezpilotních misí. Vlastní misi je možné naplánovat pomocí waypointů, nebo pomocí 3 přednastavených plánů pro bezpilotní mise: [10]

- Průzkum prostředí
- Skenování koridoru
- Skenování konstrukcí

Průzkum prostředí

Pomocí tohoto plánu se vytvoří mřížkový letový vzor nad polygonální oblastí definovanou uživatelem. Oblast je možné definovat jako mnohoúhelník nebo kruh. Je zde taky možné nastavit mřížku přeletů, výšku nad povrchem jako absolutní, nebo

relativní vůči terénu. Program umožňuje nastavení snímkování prostředí s podporou *geotagging*¹. Obrázek 3.7 zobrazuje nastavení autonomní mise s průzkumem prostředí.

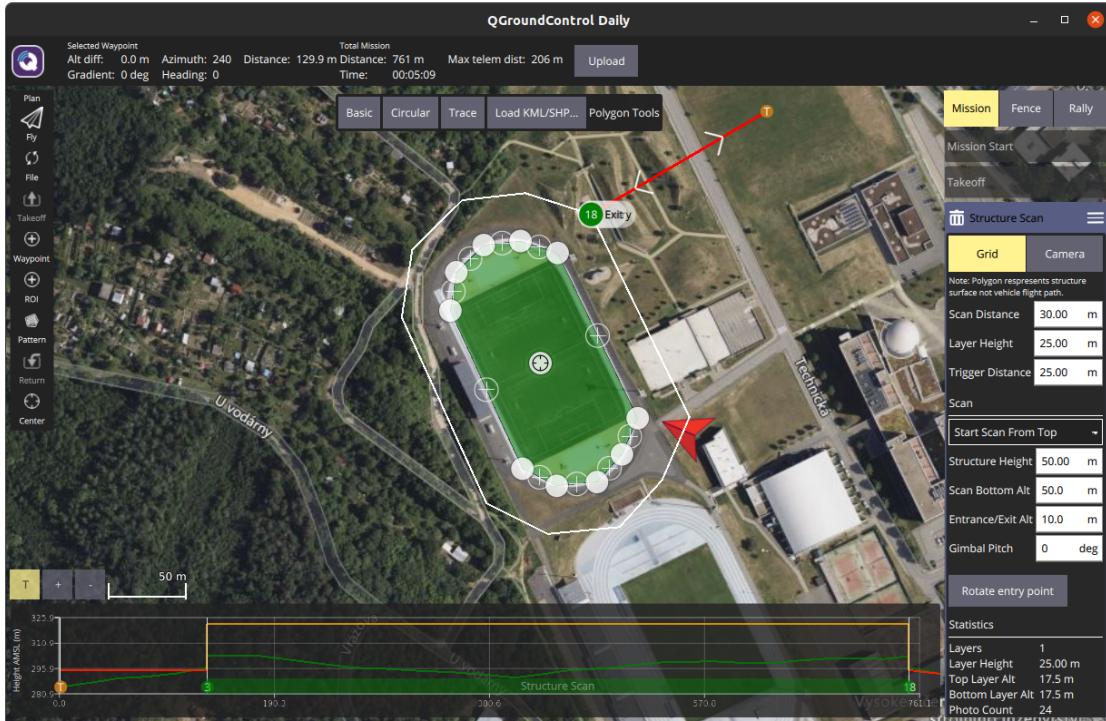


Obr. 3.7: Nastavení bezpilotní mise pro průzkum prostředí.

Skenování konstrukcí

Pomocí plánu na skenování konstrukcí je možné naplánovat bezpilotní misi zaměřenou na snímkování svislých povrchů. Snímky se používají na vizuální kontrolu budov, nebo tvorbu 3D modelů konstrukcí. Obrys konstrukce je možné zadat jako mnohoúhelník nebo kruh. Dron bude oblétat celou budovu předem v nastavených výškách tak, aby kamera směrovala vždy na budovu. Po zadání konkrétní kamery, objektivu a rozlišení snímkování [cm/px] si program sám dopočítá vhodnou letovou vzdálenost od budovy a výšku jednotlivých letových vrstev. Na obrázku 3.8 je zobrazeno nastavení autonomní mise pro skenování konstrukcí.

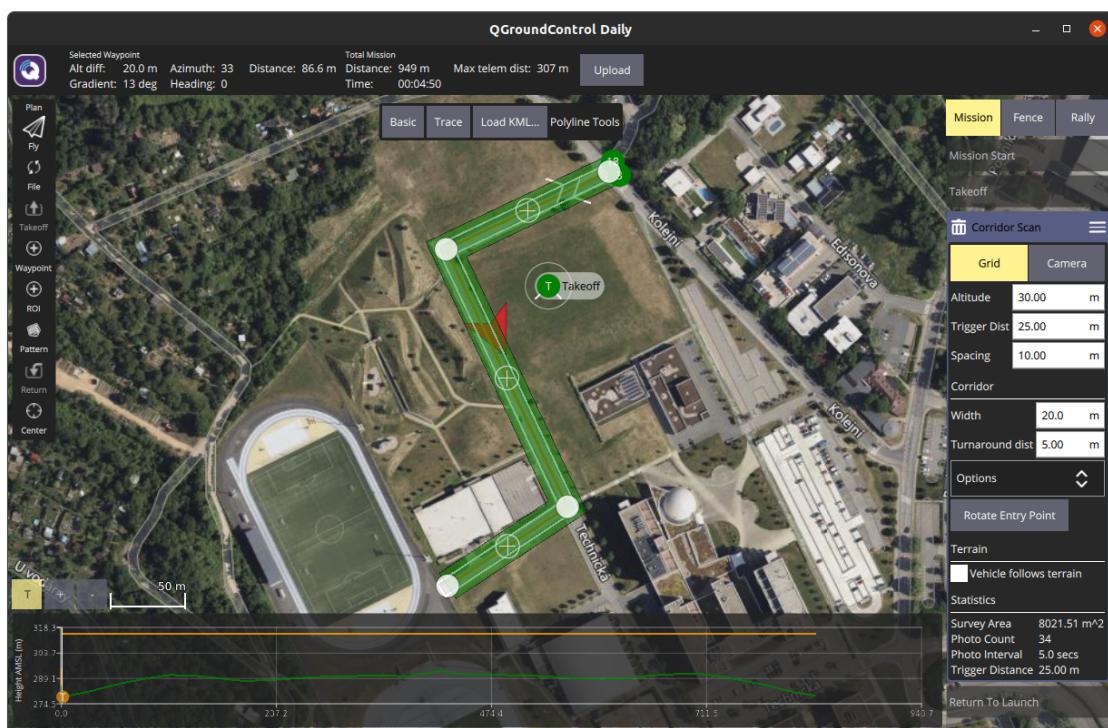
¹Geotagging představuje přidávání geografických metadat k různým médiím jako jsou např.: obrázky.



Obr. 3.8: Nastavení bezpilotní mise pro skenování konstrukcí.

Skenování koridoru

Letový plán pro skenování koridoru je vhodný pro sledování křivky (například pro průzkum silnice). Je možné si zde nastavit parametry letového vzoru jako jsou výška přeletu a hustota přeletů nad oblastí. Po zadání parametrů kamery a rozlišení snímkování [cm/px] si program sám dopočítá optimální letový vzor. Obrázek 3.9 zobrazuje nastavení letového plánu pro skenování koridoru.



Obr. 3.9: Nastavení bezpilotní mise pro skenování koridoru.

4 Propojení PX4 a ROS 2

Existují dva hlavní přístupy pro výměnu dat mezi ROS 2 middleware a PX4 firmware:

- Přímá komunikace s využitím DDS® middleware
- Komunikace pomocí MAVLink protokolu

V následující kapitole jsou detailně popsány obě možnosti komunikace.

4.1 Komunikace pomocí pomocí DDS®

4.1.1 Struktura komunikace v ROS 2

Velká změna, která se udála v ROS ekosystému při přechodu z ROS na ROS 2 bylo zvolení DDS® (Data-Distribution Service®) middleware jako hlavního komunikačního prostředku.

Každý ROS 2 uzel (*node*) je namapovaný do vlastního DDS® procesu. Pokud je spuštěno více ROS 2 uzlů, tak každý uzel je namapovaný do samostatného DDS® procesu. [11]

4.1.2 Vnitřní komunikace v PX4

uORB je asynchronní API pro zasílání zpráv *publish* a *subscribe* používané pro komunikaci mezi vlákny a procesy PX4. *uORB* API je automaticky spuštěno při startu PX4 systému, protože na něm závisí množství vnitřních procesů PX4.

V systému PX4 je definováných 167 *uORB* zpráv. Úplný seznam je zveřejněný v uživatelském manuálu PX4: https://docs.px4.io/master/en/msg_docs/ [3]

4.1.3 MicroRTPS agent a klient

PX4-Fast RTPS(DDS®) Bridge, který je také označován jako microRTPS Bridge, přidává k PX4 Autopilot rozhraní RTPS (Real Time Publish Subscribe protocol), které umožňuje výměnu zpráv *uORB* mezi různými interními komponenty PX4 a „mimo-palubní“ komunikaci s DDS® aplikacemi v reálném čase.

RTPS umožňuje lepší integraci s aplikacemi spuštěnými a propojenými v doménách DDS® (včetně uzlů ROS 2), což usnadňuje sdílení dat ze senzorů, příkazů a dalších informací o dronu.

Vhodným případem použití RTPS jsou aplikace, kde je nutné komunikovat časově kritické informace (informace v reálném čase) mezi letovým ovladačem a „mimo-palubními“ (*offboard*) komponenty. RTPS je užitečný, kdy je mimo-palubní software

kritický pro bezpilotní misi a rovnocenný s procesy běžící v PX4. Mezi možné příklady použití patří komunikace s uzly v ROS 2, které pracují s daty v reálném čase a jsou nezbytné pro ovládání dronu.

Výhoda RTPS oproti protokolu MAVlink je v tom, že dokáže komunikovat na vyšších frekvencích. [3]

MicroRTPS klient

MicroRTPS klient je *deamon*, který běží na řídící jednotce (*flight controller*) PX4. Klient se přihlásí k odběru (*subscribe*) zpráv *uORB* publikovanými interními komponenty autopilota PX4 a odesílá zprávy agentovi (přes UART nebo UDP). Klient taky přijímá zprávy od agenta a publikuje je jako *uORB* zprávy do systému PX4. [3]

MicroRTPS agent

MicroRTPS agent běží jako *deamon* na palubním počítači. Agent sleduje aktualizační *uORB* zprávy od klienta a publikuje je přes RTPS do DDS® domény. Agent taky přijímá zprávy strukturované jako *uORB* z jiných aplikací v DDS® doméně a přeposílá je klientovi.

Komunikace mezi microRTPS agentem a klientem

Jak je zobrazeno na obrázku 4.1 agent a klient jsou propojeni přes sériovou linku (UART) nebo UDP síť a informace *uORB* jsou serializovány pomocí CDR (Common Data Representation)¹ před odesláním. Micro RTPS agent a všechny DDS® aplikace jsou propojeny přes UDP a mohou být na stejném nebo jiném zařízení.

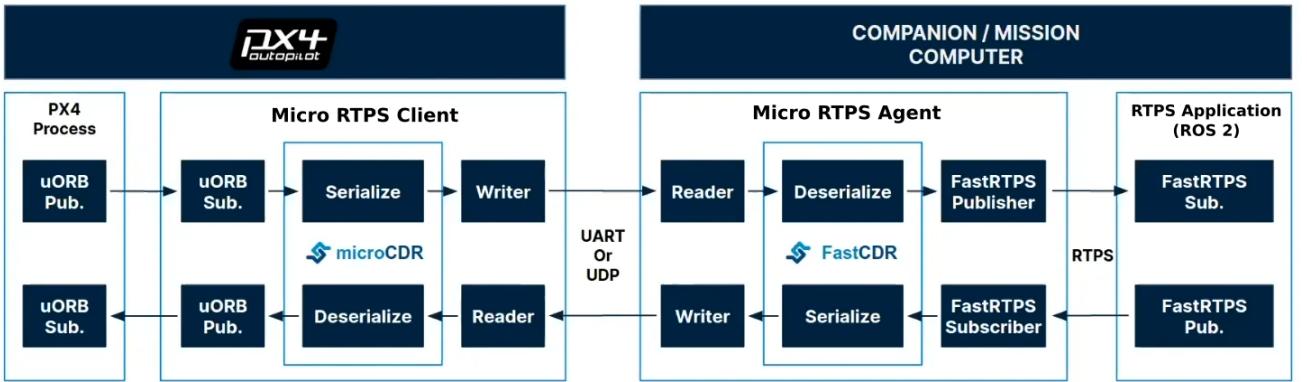
4.1.4 DDS® / RTPS middleware

Služba OMG² Data-Distribution Service for Real-Time Systems®(DDS®) je prvním otevřeným mezinárodním middlewareovým standardem, který přímo řeší komunikaci pro *publish and subscribe* systémy v reálném čase.

DDS® představuje virtuální globální datový prostor, kde mohou aplikace sdílet informace pouhým čtením a zápisem datových objektů. Datové objekty jsou adresované pomocí definovaného názvu (*topic*) a klíče (*key*). DDS® integruje součásti

¹Common Data Representation je způsob nízkoúrovňové serializace dat mezi různými platformami [12]

²Object Management Group®(OMG®) je mezinárodní neziskové konsorcium pro technologické standardy s otevřeným členstvím.



Obr. 4.1: Schéma komunikace mezi microRTPS agentem a klientem. [3]

systému dohromady a poskytuje datovou konektivitu s nízkou latencí, extrémní spolehlivost a škálovatelnou architekturu. Nabízí rozsáhlé řízení parametrů QoS (Quality of Service), včetně spolehlivosti, šířky pásma a limitů zdrojů [14].

DDS® middleware komunikuje *peer-to-peer* a nevyžaduje, aby byla data zprostředkována serverem nebo cloudem.

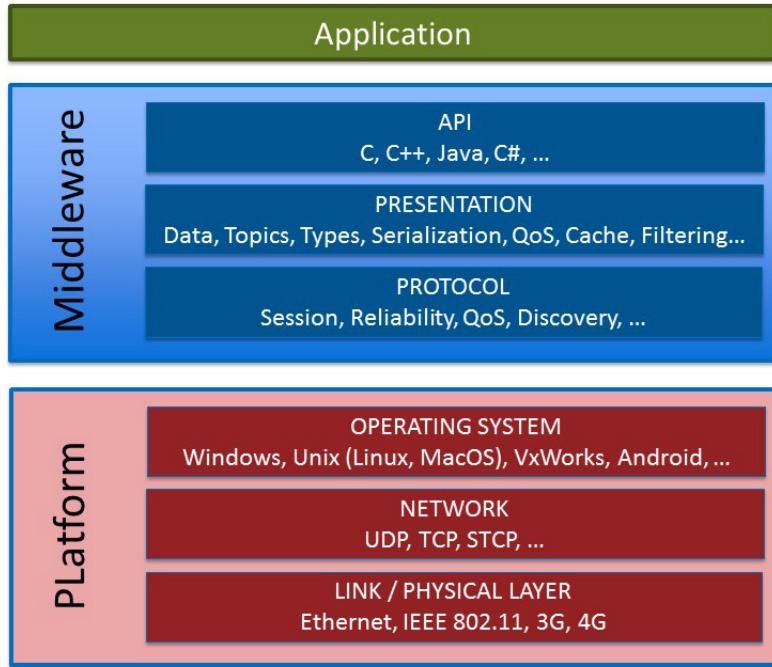
Jak je zobrazeno na obr 4.2, DDS® middleware je software vrstva, která odděluje aplikaci od detailů operačního systému, síťového přenosu a nízkoúrovňových datových formátů. Nízkoúrovňové detaily, jako je formát dat v paměti, spolehlivost, protokoly, výběr přenosového kanálu, QoS, zabezpečení atd. jsou spravovány middlewarem.

DDS® a ROS 2

DDS® middleware je základní komunikační interface pro ROS 2. Middleware zajišťuje dynamické hledání bodů v síti, serializaci a *publish/subscribe* přenos dat. ROS 2 skrývá velkou část složitosti DDS® pro většinu uživatelů, ale taky poskytuje přístup k základní implementaci DDS® pro uživatele, kteří potřebují vyřešit extrémní případy nebo potřebují integraci s jinými stávajícími DDS® systémy. [19]

ROS 2 podporuje více implementací DDS®/RTPS. Při výběru implementace middleware můžete zvážit mnoho faktorů, jako je licence, dostupnost platformy nebo výpočetní náročnost. Existují varianty middleware, které jsou vytvořeny pro mikrokontroléry, nebo varianty, které se zaměřují na aplikace vyžadující speciální bezpečnostní certifikace.

Podpora několika implementací DDS® middleware je důležitá pro zajištění toho, aby kódová základna ROS 2 nebyla vázána na žádnou konkrétní implementaci, protože uživatelé mohou chtít implementace přepínat v závislosti na potřebě jejich projektu. [20]



Obr. 4.2: Propojení DDS® middleware s aplikací a operačním systémem [15].

ROS 2 podporuje tyto DDS® implementace:

- RTI (Real-Time Innovations) Connex
- Eclipse Cyclone DDS
- eProsima Fast DDS

4.1.5 Základní principy DDS®

Orientace na data (data centricity)

DDS® middleware je jeden z mála middleware, který je zaměřen na data. Orientace na data zajišťuje, že všechny zprávy obsahují správné kontextové informace, které aplikace potřebuje k pochopení dat, která přijímají.

Podstatou orientace na data je, že DDS® middleware má informaci o tom, jaká data ukládá, a řídí, jak tato data sdílet. Programátoři používající tradiční middleware zaměřený na zprávy musí napsat kód, který odesílá zprávy. Programátoři používající middleware orientovaný na data píší kód, který určuje, jak a kdy sdílet data a poté přímo sdílejí datové hodnoty. DDS® přímo implementuje řízené, spravované a bezpečné sdílení dat. [15]

Globální datový prostor

DDS® aplikace má přístup k lokálnímu úložišti dat nazývanému *globální datový prostor*. *Globální datový prostor* vypadá pro aplikaci jako lokální paměť přístupná přes API (Application programming interface). DDS® middleware posílá zprávy k aktualizaci příslušných úložišť na vzdálených komunikačních uzlech (*nodes*).

V DDS® aplikacích neexistuje žádné globální datové místo, kde by všechna data existovala. Každá aplikace si lokálně ukládá jen to, co potřebuje a jen tak dlouho, jak to potřebuje. [15]

DDS® se zabývá daty v pohybu - globální datový prostor je virtuální koncept, který je ve skutečnosti pouze souborem lokálních úložišť. Každá aplikace, téměř v jakémkoli jazyce, běžící na jakémkoli systému, vidí lokální paměť ve svém optimálním nativním formátu. Globální datový prostor sdílí data mezi vestavěnými, mobilními a cloudovými aplikacemi v rámci jakéhokoli přenosu, bez ohledu na jazyk nebo systém, a s extrémně nízkou latencí.

Základní architektura DDS®

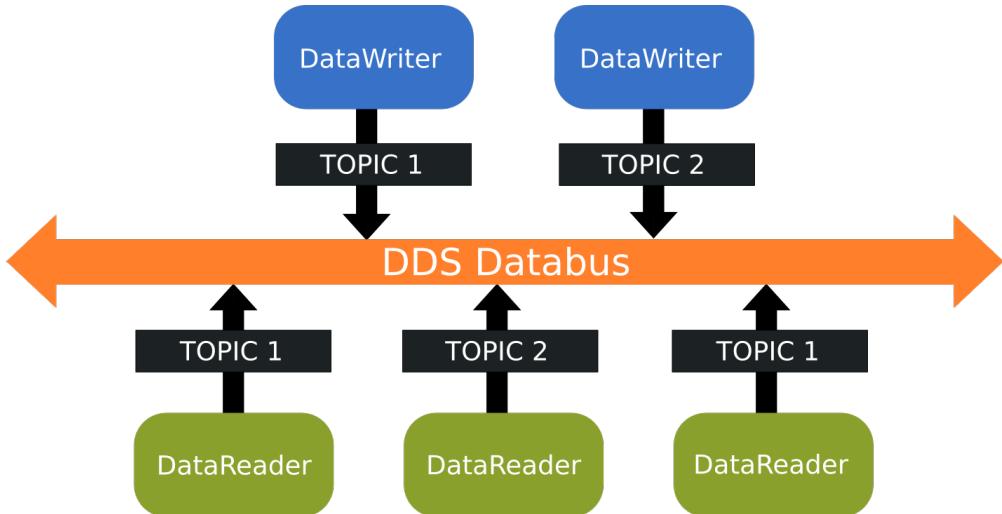
Nejzákladnějším komunikačním vzorem podporovaným DDS® je model publikování/přihlášení (*Publish/Subscribe*). *Publish/Subscribe* je komunikační model, kde producenti dat publikují data a spotřebitelé dat se přihlašují k odběru dat. Tito producenti a spotřebitelé o sobě nemusí vědět předem, protože se navzájem dynamicky objevují za běhu. Data, která sdílejí, jsou popsána tématem (*topic*) a producenti a spotřebitelé odesírají a přijímají data pouze pro téma, která je zajímají. V tomto vzoru může mnoho producentů publikovat stejně téma a mnoho spotřebitelů se může přihlásit k odběru stejného tématu. Spotřebitelé dostávají data od všech producentů, se kterými sdílejí téma. Producenti odesírají data přímo Spotřebitelům, aniž by potřebovali brokera nebo centralizovanou aplikaci pro zprostředkování komunikace. [16]

Jak je zobrazeno na obrázku 4.3 v DDS® se objekty, které publikují data, nazývají *DataWriters* a objekty, které odebírají data, jsou *DataReaders*. *DataWriters* a *DataReaders* jsou spojeni s jedním objektem *Topic*, který tato data popisuje.

Quality of Service

V DDS® aplikacích je možné sdílet data s flexibilními specifikacemi kvality služeb (QoS), jako jsou spolehlivost, stav systému, živost, zabezpečení a mnohé další.

DDS® middleware posílá jen data, které jsou nutné pro správnou funkci systému. Pokud se zprávy nedostanou do zamýšlených cílů, middleware automaticky zvýší spolehlivost na daných datových cestách. Když se komunikační systémy změní,



Obr. 4.3: Základní *publish/subscribe* struktura DDS® komunikace [16].

middleware dynamicky zjistí, kam poslat data, a informuje účastníky o změnách. Pokud je celková velikost dat obrovská, DDS® inteligentně filtruje a odesílá pouze data, která každý koncový bod skutečně potřebuje a tím snižuje nároky na komunikační kanál. Když je potřeba, aby byly aktualizace rychlé, DDS® odesílá multicastové zprávy pro aktualizaci mnoha vzdálených aplikací najednou. U aplikací kritických z hlediska zabezpečení DDS® řídí přístup, vyhodnocuje cesty toku dat a šifruje data za běhu. [15]

RTPS protokol

DDSI-RTPS™ (DDS Interoperability Wire Protocol™), jinak RTPS (Real Time Publish Subscribe protocol) je protokol pro spolehlivou komunikaci *publish/subscribe* přes nespolehlivé přenosy, jako je UDP pro unicast i multicast. [13]

RTPS byl standardizován OMG (Object Management Group) jako protokol pro implementaci Data-Distribution Service® (DDS®), standard široce používaný v leteckém a obranném sektoru pro aplikace v reálném čase.

Kromě implementací RTPS zabudovaných do různých implementací DDS® existují samostatné lehké implementace RTPS (eProsima Fast RTPS).

Protokol RTPS je založen na různých modulech, které řídí výměnu informací mezi různými DDS® aplikacemi: [18]

- *Structure module* - definuje komunikační koncové body (*endpoints*) a mapuje je na jejich protějšky DDS®.
- *Message module* - definuje, které zprávy si mohou tyto koncové body vyměňovat a jak jsou sestaveny.

- *Behavior module* - definuje sadu povolených interakcí a jejich vliv na každý z koncových bodů.
- *Discovery module* - definuje sadu vestavěných koncových bodů, které umožňují automatické zjišťování.

Dynamické hledání bodů v síti

DDS® poskytuje dynamické zjišťování producentů a spotřebitelů dat. Dynamické hledání umožňuje rozšíritelnost všech DDS® aplikací. To znamená, že aplikace nemusí znát nebo konfigurovat koncové body pro komunikaci, protože je automaticky za běhu zjistí DDS®.

Při dynamickém hledání bodů v síti DDS® zjistí, zda koncový bod publikuje data, odebírá data nebo obojí. Zjistí typ dat, která jsou publikována nebo odebírána. Zjistí také komunikační charakteristiky nabízené producentem a komunikační charakteristiky požadované spotřebitelem dat. Všechny tyto atributy jsou brány v úvahu při dynamickém zjišťování a párování účastníků DDS®. [15]

Účastníci DDS® komunikace mohou být na stejném počítači nebo v síti. Není potřeba znát nebo konfigurovat IP adresy nebo brát v úvahu rozdíly v architektuře strojů, takže přidání dalšího účastníka komunikace je snadné.

4.1.6 Využití DDS® v praxi

Data-Distribution Service® je základem některých průmyslových standardů: [17]

- OpenFMB (Open Field Message Bus)
- Adaptive AUTOSAR (AUTomotive Open System ARchitecture)
- MD PnP (Medical Device „Plug-and-Play“)
- GVA (Generic Vehicle Architecture)
- NGVA (NATO Generic Vehicle Architecture)
- ROS2 (Robot Operating System 2)

DDS® má rozsáhlý seznam různých instalací, které jsou často kritické. [19]

- Bitevní lodě
- Velké inženýrské instalace, jako jsou přehrady
- Finanční systémy
- Vesmírné systémy
- Letové systémy
- Systémy vlakových rozvaděčů

4.2 Komunikace pomocí MAVLink

MAVLink (*Micro Air Vehicle communication protocol*) je „lehký“ protokol využívaný už od roku 2009 pro komunikaci mezi vnitřními procesy na palubě dronu, nebo mezi pozemní stanicí a dronem v náročných komunikačních podmírkách jako je vysoká latence, nebo šum prostředí.

Základní komunikační schémata využívána protokolem MAVLink jsou *publish-subscribe* a *point-to-point*.

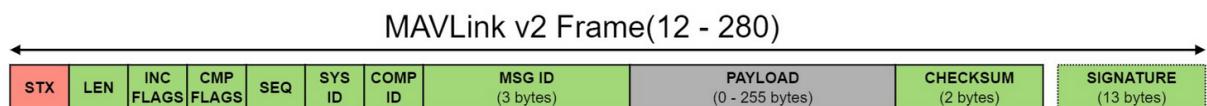
Knihovny, které implementují funkce a definují zprávy MAVLink protokolu, jsou pro většinu programovacích jazyků vydány s MIT licencí, takže je lze bez omezení využívat v libovolné aplikaci s uzavřeným zdrojovým kódem. Proto je možné MAVLink protokol využívat na různých platformách, jako jsou Windows, Linux, iOS a Android, ale i na mikrokontrolérech včetně ARM7, ATMega, dsPic, STM32. [26]

4.2.1 MAVLink zprávy

Společná sada zpráv MAVLink obsahuje standardní definice, které jsou spravovány protokolem MAVLink. Tyto definice zahrnují hlavně funkce, které jsou považovány za užitečné pro většinu pozemních řídicích stanic a autopilotů. Od všech systémů kompatibilních s protokolem MAVLink se očekává, že budou tyto definice používat tam, kde je to možné.

Seznam standardních definic zpráv protokolu MAVLink je zveřejněn zde: <https://mavlink.io/en/messages/common.html>

Na obrázku 4.4 je zobrazena struktura MAVLink zprávy.



Obr. 4.4: Struktura MAVLink zprávy [26].

- **STX:** Indikace začátku paketu (hodnota 0xFD)
- **LEN:** Délka sekce s datama - **PAYLOAD**
- **INC FLAGS:** Příznak nekompatibility
 - Definuje kritické funkce, které musí daná implementace MAVLinku podporovat, aby bylo možné zpracovat paket.
- **CMP FLAGS:** Příznak kompatibility
 - Obsahuje doplňující informace, které nejsou nutné k správnému zpracování paketu.

- **SEQ**: Pořadové číslo paketu pro zprávy skládající se z více paketů
- **SYS ID**: Číslo systému v síti (vozidlo)
- **COMP ID**: Číslo komponenty na definovaném systému (kamera, autopilot, ...)
- **MSG ID**: Číslo zprávy
- **PAYLOAD**: Data
 - Struktura dat je definována číslem zprávy (**MSG ID**)
- **CHECKSUM**: Informace pro detekci, jestli zpráva dorazila v pořádku
- **SIGNATURE**: Podpis, který zajistí zprávu proti neoprávněné manipulaci

4.2.2 MAVLink mikroslužby

MAVLink mikroslužby definují protokoly vyšší úrovně, pomocí kterých mohou systémy MAVLink komunikovat s vyšší efektivitou. Například pozemní stanice (QGroundControl, ArduPilot) a autopilot PX4 sdílejí společný příkazový protokol pro zasílání zpráv vyžadující potvrzení (*acknowledge*).

Mikroslužby zajišťují, jak se rozdělí a znova sestaví data, které mají větší výšku, než je velikost jedné zprávy. Jestliže jsou nějaká data ztracena v průběhu přenosu, mikroslužby iniciují opětovné odeslání dat.

Většina mikroslužeb využívá vzor *client-server*, například pozemní stanice (klient) iniciuje požadavek a vozidlo (server) odpovídá daty.

4.2.3 Využití protokolu MAVLink

Velké množství autopilotů, pozemních stanic, API, projektů a dalších softwarových balíčků používá protokol MAVLink.

Autopilot

Na trhu existuje velké množství hobby autopilotů, nebo profesionálních řešení pro modely letadel a dronů. Níže uvedený seznam zobrazuje aktivní vyvýjené autopiloty, které využívají komunikační MAVLink protokol. [26]

- PX4
- ArduPilot
- AutoQuad 6 AutoPilot
- iNAV
- SmartAP Autopilot

Pozemní stanice

- QGroundControl
- AutoQuad GCS

- SmartAP GCS
- Yuneec Datapilot
- Sentera Groundstation
- WingtraPilot
- APM Planner 2.
- Mission Planner
- MAVProxy
- UgCS (Universal Ground Control Station)
- Side Pilot
- JAGCS
- Flightzoomer
- Inexa Control
- Synturian Control

API

Pro zjednodušení interakce s autopiloty, kamerami, pozemními stanicemi a mnoha dalšíma uzly s podporou MAVLink protokolu bylo vytvořeno několik vysokoúrovňových API. Tyto API poskytují implementace hlavních mikroslužeb a jednoduché komunikační rozhraní pro odesílání příkazů a k příjmu informací o vozidle. V níže uvedeném seznamu jsou aktuálně udržované implementace.

- MAVSDK - MAVLink API Library (C++, Python, Swift, Java, JS)
- Dronecode Camera Manager - Přidává rozhraní pro kamery připojené k Linuxovým systémům
- Rosetta Drone - MAVLink vazba na DJI SDK (Software development kit) - propojení DJI dronu a pozemní stanice podporující MAVLink
- Pymavlink - MAVLink vazba na Python
- MAVROS - Přemostění ROS (ROS 2) a MAVLink
- DroneKit - MAVLink API pro Python a Android (optimalizované pro ArduPilot)

4.2.4 Přemostění ROS 2 a MAVLink

K přemostění ROS (ROS 2) a protokolu MAVLink slouží balíček Mavros. [27]

Mavros je rozšiřitelný komunikační MAVLink uzel (*node*) pro ROS (ROS 2). Poskytuje komunikační ovladač pro různé autopiloty s komunikačním protokolem MAVLink. Dále poskytuje komunikační UDP kanál pro pozemní řídicí stanice jako je například QGroundControl nebo ArduPilot.

Vlastnosti balíčku Mavros:

- Komunikace s autopilotem nebo pozemní stanicí přes sériový port, UDP nebo TCP
- Nástroje pro manipulaci s parametry
- Nástroje pro manipulaci s bodmi trasy (*waypoints*)
- Podpora módu OFFBOARD

- Konverze geografických souřadnic

Práce s Mavros uzlem

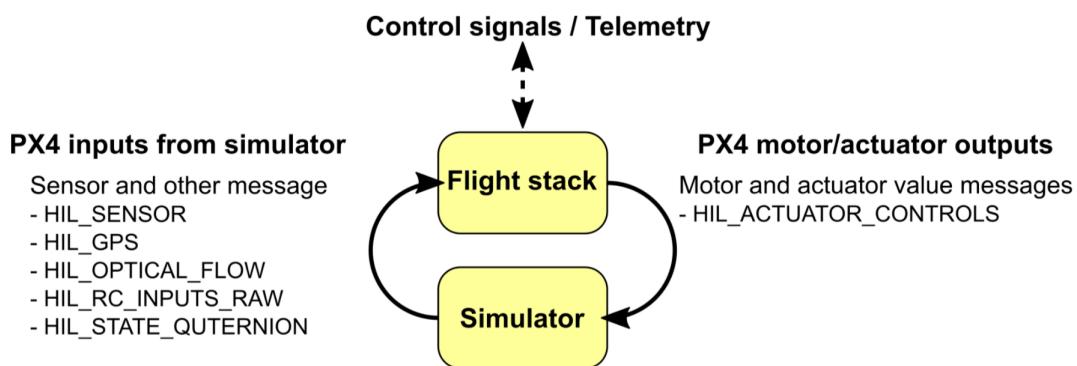
Po vytvoření Mavros uzlu (*node*) jsou téma (*ROS topic*) a služby (*ROS services*) související s autopilotem připraveny k použití. Pak stačí vytvořit ROS (ROS 2) uzel pro zpracování dat a povolení dronu, který bude komunikovat na dostupná téma, nebo služby. [28]

5 Simulace ve virtuálním prostředí

Simulátory umožňují letovému kódu PX4 ovládat počítačem namodelovaný dron v simulovaném „světě“. S tímto dronem je možné komunikovat stejně jako se skutečným dronem pomocí QGroundControl, offboard API nebo rádiového ovladače/gamepadu.

PX4 podporuje jak simulaci SITL (*Software In The Loop*), kde PX4 firmware (program řídící jednotky) běží na počítači, nebo simulaci HITL (*Hardware In The Loop*), kde PX4 firmware běží na skutečné řídící jednotce a vše ostatní (fyzika reálného světa) je spuštěno na počítači. [3]

Všechny simulátory mohou komunikovat s firmware PX4 pomocí *Simulator MAVLink API*, které definuje sadu zpráv MAVLink. Na obrázku 5.1 je zobrazen tok zpráv mezi „simulovaným světem“ a mezi firmware PX4.



Obr. 5.1: Tok zpráv mezi simulátorem a PX4 [3].

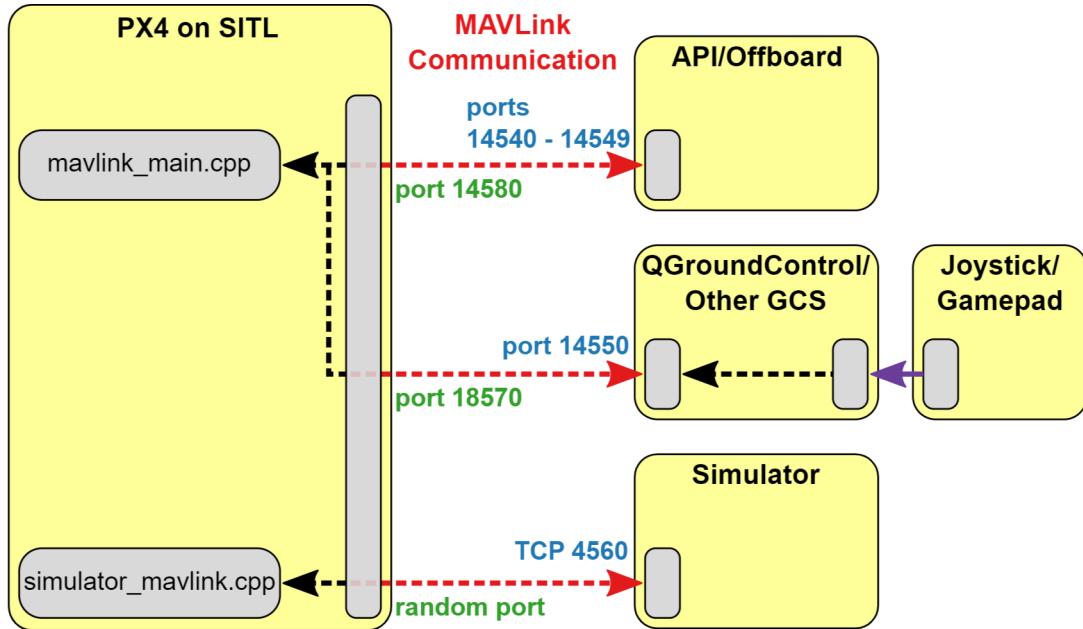
5.1 SITL simulační prostředí

Software In The Loop simulace umožňuje rychlé a „levné“ ladění robotických misí, protože pro simulaci SITL se nevyužívá žádný specializovaný hardware, ale jenom počítač s nainstalovaným simulačním prostředím.

Obrázek 5.2 zobrazuje typickou konfiguraci simulace SITL pro PX4 firmware a pro kterýkoliv z podporovaných simulátorů. Simulátor a PX4 firmware mohou spolu komunikovat pomocí MAVLink API. Pro přímou interakci s PX4 je možné využít komunikaci přes *Micro-RTPS bridge*, kde se komunikují přímo uORB zprávy.

Různé části simulačního prostředí spolu komunikují pomocí UDP (User datagram protocol) a mohou být spuštěny na stejném počítači, nebo jiném počítači

ve stejné síti. Pomocí sériové komunikace je možné připojit joystick, gamepad nebo RC soupravu pro ruční ovládání simulovaného dronu přes QGroundControl.



Obr. 5.2: Schéma komunikace mezi PX4 a simulátorem [3].

5.1.1 Seznam podporovaných simulátorů

Následující simulátory jsou podporovány systémem PX4:

- Gazebo
- FlightGear
- JSBSim
- jMAVSIM
- AirSim
- Ignition Gazebo

5.1.2 Gazebo

Simulátor Gazebo je vysoce doporučen pro PX4 *Software In The Loop* simulaci.

Gazebo je dobře navržený a otestovaný simulátor, umožňující rychlé testování algoritmů, návrh robotů a trénování systémů s umělou inteligencí v realistických scénářích. Nabízí možnost přesně a efektivně simulovat populace robotů ve složitých vnitřních i venkovních prostředích. Gazebo se běžně používá pro práci s ROS (ROS 2). [21]

Podporovaná vozidla SITL simulace PX4 v simulátoru Gazebo jsou:

- dron
- delta VTOL (Vertical Take-Off and Landing)
- letadlo
- rover
- ponorka

5.1.3 Ignition Gazebo

Vývojový tím simulátoru Gazebo se v budoucnu zaměří na vývoj simulátoru Ignition, takže Gazebo 11 je poslední verze známého simulátoru Gazebo. Oba simulátory jsou kompatibilní ve formátu popisu prostředí („světa“) a některých pluginů. [22]

Z důvodu, že simulační firmware PX4 je v této době lépe přepojený se simulátorem Gazebo, tak v této práci budeme pracovat s simulátorem Gazebo 11.

Jediné vozidlo, které je podporované *Software In The Loop* simulací PX4 v simulátoru Ignition je dron.

5.2 Instalace virtuálního prostředí

Pro vývoj a simulaci bezpilotních misí v prostředí PX4 a ROS 2 je nutné nainstalovat jednotlivé komponenty vývojového prostředí:

- PX4 simulační firmware
- Gazebo simulátor
- ROS 2
- Komunikační most mezi PX4 firmware a ROS 2
 - PX4 - ROS 2 *bridge*
 - Mavros

V následujících podkapitolách si popíšeme instalaci jednotlivých částí. Návod je určený pro linuxovou distribuci Ubuntu 20.04 *Focal Fossa*.

5.2.1 Instalace PX4 simulačního prostředí

Prvním krokem je instalace PX4 prostředí. Při reální misi (nebo HITL simulaci) bude PX4 firmware spuštěný přímo na řídící jednotce Pixhawk. Při SITL simulaci je nutné PX4 letový kód nainstalovat na počítač. Doporučená platforma pro PX4 prostředí je Ubuntu, ale je možné ho spustit na Windows 10 nebo na Mac OS.

Samotná instalace prostředí PX4 je jednoduchá a skládá se z 2 kroků. [3]

```
1 # 1. Stáhnout zdrojový kód PX4:
2 $ git clone https://github.com/PX4/PX4-Autopilot.git --recursive
```

```

3
4 # 2. Spustit script ubuntu.sh bez argumentů
5 # pro instalaci všech závislostí
6 $ bash ./PX4-Autopilot/Tools/setup/ubuntu.sh

```

Script `ubuntu.sh` nastaví vývojové prostředí PX4, dále nainstaluje simulátory Gazebo a jMavSim a sadu nástrojů *NuttX*¹/*Pixhawk*

5.2.2 Instalace ROS 2 Foxy a všech závislostí

Instalace ROS 2 je dobře popsaná na dokumentačních stránkách [24]. Po úspěšné instalaci ROS 2 Foxy je nutné doinstalovat několik závislostí:

```

1 # 1. Instalační proces ROS 2 by měl nainstalovat
2 #     nástroje pro sestavení colcon, ale v případě,
3 #     že se tak nestane, můžete nainstalovat ručně:
4 $ sudo apt install python3-colcon-common-extensions
5
6 # 2. Eigen3 se používá v knihovně transformací:
7 $ sudo apt install ros-foxy-eigen3-cmake-module
8
9 # 3. Další závislosti Pythonu musí být také nainstalovány:
10 $ sudo pip3 install -U empy pyros-genmsg
11 $ pip3 install transforms3d
12
13 # 3. Nainstalovat pluginy do simulátoru gazebo:
14 $ sudo apt install ros-foxy-gazebo-ros-pkgs

```

5.2.3 Komunikace pomocí RTPS

Jak je popsáno v kapitole 4.1.3 MicroRTPS agent a klient, pro komunikaci mezi ROS 2 *node* a PX4 slouží *eProsimma Fast DDS*, který umožňuje výměnu zpráv uORB mezi PX4 a ROS 2.

Pro nastavení PX4 - ROS 2 *bridge* potřebujeme nainstalovat následující balíčky:

- Java JDK (Open source JDK 11)
- Gradle
- Fast-RTPS-Gen

¹NuttX je RTOS (Real-Time Operating System) pro provoz PX4 na řídící jednotce Pixhawk. Je to open source (BSD licence), lehký, výkonný a velmi stabilní operační systém. [3]

Po úspěšné instalaci balíčků je nutné sestavit ROS 2 workspace.

Java JDK

Java JDK je nutná závislost pro správné sestavení komunikačního RTPS agenta.

```
1 $ sudo apt install openjdk-11-jdk
```

Gradle

Gradle je univerzální nástroj pro sestavení software v jazycích C++, Swift, Java. V našem případě je nutný pro sestavení komunikačního RTPS agenta.

Gradle je možné nainstalovat pomocí následujících příkazů [23]:

```
1 # Stažení, extrahování a vytvoření odkazu
2 $ wget https://services.gradle.org/distributions/gradle-6.3-bin.zip \
3   -P /tmp
4 $ sudo unzip -d /opt/gradle /tmp/gradle-6.3-bin.zip
5 $ sudo ln -s /opt/gradle/gradle-6.3 /opt/gradle/latest
```

Pro přidání Gradle adresářu do systémové proměnné PATH je nutné vytvořit soubor `/etc/profile.d/gradle.sh` a přidat následnou konfiguraci.

```
1 $ sudo nano /etc/profile.d/gradle.sh
1   export GRADLE_HOME=/opt/gradle/latest
2   export PATH=$GRADLE_HOME/bin:$PATH
```

Následně je nutné změnit oprávnění skriptu pomocí příkazu:

```
1 $ sudo chmod +x /etc/profile.d/gradle.sh
```

Fast RTPS Gen

Fast-RTPS-Gen je nástroj pro generování IDL (Interactive Data Language) kódu pro Fast RTPS (DDS®).

Fast-RTPS-Gen je možné nainstalovat pomocí následujících příkazů:

```
1 # Načtení proměnných prostředí
2 $ source /etc/profile.d/gradle.sh
3
4 # Stažení a sestavení Fast DDS Generátoru
5 $ git clone https://github.com/eProsima/Fast-DDS-Gen.git \
6   --recursive -b v1.0.4 ~/Fast-RTPS-Gen \
7   && cd ~/Fast-RTPS-Gen \
8   && gradle assemble \
9   && sudo env "PATH=$PATH" gradle install
```

Sestavení ROS 2 pracovního prostoru

Pro vytvoření ROS 2 pracovního prostoru (*workspace*) podporujícího simulaci v PX4 s komunikací pomocí RTPS je nutné stáhnout dva ROS 2 balíčky. [3]

Na stáhnutí a sestavení obou balíčků můžeme použít tyto příkazy:

```
1 # Naklonování obou balíčků:  
2 $ mkdir -p ~/px4_ros_missions/src  
3 $ git clone https://github.com/PX4/px4_ros_com.git \  
4   ~/px4_ros_missions/src/px4_ros_com  
5 $ git clone https://github.com/PX4/px4_msgs.git \  
6   ~/px4_ros_missions/src/px4_msgs  
7  
8 # Balíček px4_ros_com obsahuje scripty na sestavení workspace:  
9 $ cd ~/px4_ros_missions/src/px4_ros_com/scripts  
10 $ source build_ros2_workspace.bash
```

Balíček px4_msgs

V balíčku px4_msgs je nadefinovaných více jak 200 uORB zpráv pro komunikaci mezi ROS 2 *node* a PX4 firmware.

Balíček px4_ros_com

Balíček px4_ros_com generuje přepojení (*MicroRTPS_agent*) mezi ROS 2 a PX4 firmware pomocí Fast RTPS (DDS®).

RTPS agent se na palubním počítači spouští pomocí příkazu:

```
1 $ source ~/px4_ros_missions/install/setup.bash  
2 $ micrortps_agent -t UDP
```

5.2.4 Komunikace pomocí MAVLink

Pro komunikaci ROS 2 s autopilotem PX4 pomocí protokolu MAVLink je nutné nainstalovat ROS 2 balíček Mavros.

Pro instalaci balíčku Mavros slouží následující příkaz:

```
1 $ sudo apt install ros-foxy-mavros
```

Pro správnou funkci balíčku Mavros je nutné doinstalovat geografický dataset (*geographiclib dataset*) který slouží na převod mezi zeměpisnými, UTM, UPS, MGRS, geocentrickými a místními kartézskými souřadnicemi, pro výpočty gravitace, výšky geoidu a geomagnetického pole a pro řešení geodetických úloh. [29]

K tomu slouží následující příkazy:

```
1 $ wget https://raw.githubusercontent.com/mavlink/mavros\  
2   /ros2/mavros/scripts/install_geographiclib_datasets.sh  
3 $ sudo bash install_geographiclib_datasets.sh  
4 $ rm install_geographiclib_datasets.sh  
5  
6 # Nainstalování python závislostí:  
7 $ sudo apt install python3-lxml libxml2-utils
```

6 Implementace robotických misí

Jedna možnost pro vytvoření autonomní mise v systému PX4 je pomocí programu QGroundControl, jak je popsáno v kapitole 3.4.2 Plánování bezpilotní mise. V tomto software je možné vytvořit jednoduché mise na průzkum prostředí, skenování kori-doru, skenování konstrukcí a obecnou misi pomocí *waypoints*. Celá mise musí být naplánovaná před startem, takže dron nemůže pohotově reagovat na různé situace, které nastanou v průběhu letu.

Další možnost je vytvoření bezpilotní robotické mise v nadřazeném palubním počítači pomocí ROS 2 *node*, který úkoluje řídící jednotku dronu. Tato možnost je vhodná pro složité mise, kde není před startem známá celá trajektorie letu a řídící algoritmus v palubním počítači ji dopočítává na základě čtení ze snímačů a kamer, nebo na základě povelů z pozemní stanice.

Pro tento případ jsme vytvořili několik ROS 2 uzelů (v C++), které komunikují s PX4 firmware pomocí Fast RTPS, nebo pomocí MAVLink protokolu. Zdrojový kód je zveřejněný na platformě GitHub [25].

Pro ovládání dronu z palubního počítače musí mít dron aktivovaný *offboard flight mode* (mimopalubní letový režim).

6.1 Offboard flight mode

Offboard flight mode se využívá vždy, když je dron řízený z jiného zdroje, než z Pixhawk řídící jednotky (PX4 firmware), například z palubního počítače.

V *offboard* letovém režimu se dronu posílají hlavě zprávy pro let na definovanou pozici (absolutní nebo relativní), let určitým azimutem a rychlostí, nebo zprávy definující zrychlení dronu ve všech osách. Je vhodné, aby se pro kritické operace jako jsou vzlet, přistání, návrat na startovací pozici (*Return to Launch*) využívali odpovídající letecké režimy řídící jednotky Pixhawk.

Aby zůstal *offboard* letový režim aktivní, palubní počítač musí posílat zprávy o aktivitě módu (`OffboardControlMode message`) s frekvencí $> 2 \text{ Hz}$.

V případě poruchy palubního počítače, nebo nedostatečné rychlosti posílání zpráv se *offboard* letový režim vypne a bude aktivovaný předem bezpečný, defi-novaný letový režim, například *land* letový režim, takže dron přistane na daném místě. [3]

6.2 Struktura robotické mise

Jedním z cílů práce je navrhnout a implementovat ukázkovou robotickou misi pro autonomní létání dronů. V této kapitole se budeme zabývat námi implementovaným

řešením robotické mise.

6.2.1 Základní funkce dronu

Vytvořili jsme základní třídu v C++ pro komunikaci s dronem pomocí protokolu MAVLink, která implementuje nezbytné funkce k řízení dronu. Při vytváření nové robotické mise se tato základní třída jednoduše zdědí a programátor se již nemusí zabývat implementací *publisherů* a *subscriberů* a může se soustředit na programování složitějších algoritmů robotické mise.

Dron je možné povelovat pomocí následujících funkcí, které jsou implementované v základní třídě:

Funkce `arm`

Funkce `arm` slouží k přepnutí dronu do *arm* stavu, v kterém jsou motory aktivní a dron je schopen letu. Mimo tohoto stavu není možné dron ovládat.

Funkce `disarm`

Funkce `disarm` přepíná dron do neaktivního (bezpečného) stavu, kdy není možné spustit motory dronu.

Funkce `setFlightMode`

Funkce `setFlightMode` mění letové režimy dronu. Víc o letových režimech je popsáno v kapitole 3.3 Letové režimy PX4.

Funkce `pullParam`

Pro změnu vnitřních parametrů PX4 přes ROS 2 je nutné, aby spuštěný Mavros *node* měl informaci o všech aktivních parametrech PX4. Funkce `pullParam` vyžádá všechny parametry z PX4.

Funkce `preflightCheck`

Funkce `preflightCheck` nastaví parametry PX4, které jsou nezbytné pro robotickou misi, jako jsou výška vzletu, horizontální rychlosť, povolení *offboard* módu a chování dronu po výpadku *offboard* módu. Dále tato funkce zkонтroluje, jestli proběhla korektní geolokace (GPS *fix*).

Funkce publish_traj_setp_position

Funkce `publish_traj_setp_position` posílá do dronu *setpoint* pro *offboard* mód s relativní pozicí dronu vůči startovacímu místu. Když je dron přepnutý do *offboard* módu a posíláme dronu setpointy s relativní pozicí, tak musí být tato funkce volána s frekvencí $> 2 \text{ Hz}$.

Funkce publish_traj_setp_speed

Funkce `publish_traj_setp_speed` posílá do dronu příkazy pro let určitou lineární rychlostí v 3 osách a úhlovou rychlostí v 1 ose (*yaw*). Při povelování dronu pomocí rychlostí vynecháváme poziční regulátor v firmware PX4, takže si ho musíme implementovat sami. Tato funkce musí být volána s frekvencí $> 2 \text{ Hz}$, aby systém PX4 nevyhodnotil výpadek *offboard* módu a neukončil misi předčasně.

Funkce publish_traj_setp_geo

Funkce `publish_traj_setp_geo` posílá dronu globální poziční *setpoint* v souřadničovém systému WGS 84. Pro setrvání v *offboard* módu musí být tato funkce volána s frekvencí $> 2 \text{ Hz}$.

Funkce isG1SetpReached

Funkce `isG1SetpReached` je pomocná funkce, která zjišťuje, zda dron v *offboard* módu doletěl na nastavenou globální souřadnici.

Výše zmíněné funkce komunikují s dronem pomocí Mavros uzlu přes ROS 2 *publishery*, *subscribers* a *clients* (*services*). Komunikace pomocí ROS 2 *services* je výhodná pro kritické údaje kvůli tomu, že po odeslání požadavku z ROS 2 uzlu (například na změnu letového režimu, nebo nastavení vnitřních parametrů PX4) dostaneme od PX4 systému asynchronní odpověď, že požadavek byl zpracován.

Tabulka 6.1 zobrazuje a popisuje všechny ROS 2 *publishery*, *subscribers* a *clients* (*services*), které využívá základní třída dronu.

6.2.2 Struktura řídícího algoritmu

Všechny námi implementované robotické mise mají stejnou základní strukturu algoritmu. Pro řízení mise je využitý jeden hlavní stavový automat, který ovládá všechny úkony od vzletu až po ukončení mise.

Kromě vzletu, kde je použitý *take off* letový režim je v celé robotické misi využíván *offboard* letový režim, v kterém dron vykonává všechny další úkony. Podle typu

Tab. 6.1: Přehled využitých objektů typu *publisher*, *subscriber* a *client* v základní třídě dronu.

Objekt	ROS 2 zpráva	Popis
Subscriber	<code>mavros_msgs::msg::State</code>	Stav řídící jednotky
	<code>mavros_msgs::msg::Altitude</code>	Nadmořská výška
	<code>sensor_msgs::msg::NavSatFix</code>	Pozice podle WGS 84
	<code>geometry_msgs::msg::PoseStamped</code>	Relativní pozice
Publisher	<code>geometry_msgs::msg::PoseStamped</code>	Setpoint pro lokální pozici
	<code>geometry_msgs::msg::TwistStamped</code>	Setpoint pro nastavení rychlostí
	<code>geographic_msgs::msg::GeoPoseStamped</code>	Setpoint pro globální pozici
Client	<code>mavros_msgs::srv::CommandBool</code>	Nastavení <code>arm</code> nebo <code>disarm</code>
	<code>mavros_msgs::srv::SetMode</code>	Nastavení letových režimů
	<code>mavros_msgs::srv::ParamSetV2</code>	Nastavení parametrů PX4
	<code>mavros_msgs::srv::ParamPull</code>	Vyžádání všech parametrů

robotické mise řídící systém posílá dronu v *offboard* letovém režimu lokální nebo globální poziční waypoints, nebo rychlostní setpointy.

6.2.3 Stavový automat pro řízení mise

Na obrázku 6.1 jsou zobrazeny kroky základního stavového automatu pro řízení námi implementovaných misí.

V následující části jsou popsány stavy a podmínky přechodů v stavovém automatu pro řízení mise:

Stav 0: Čekání na start mise

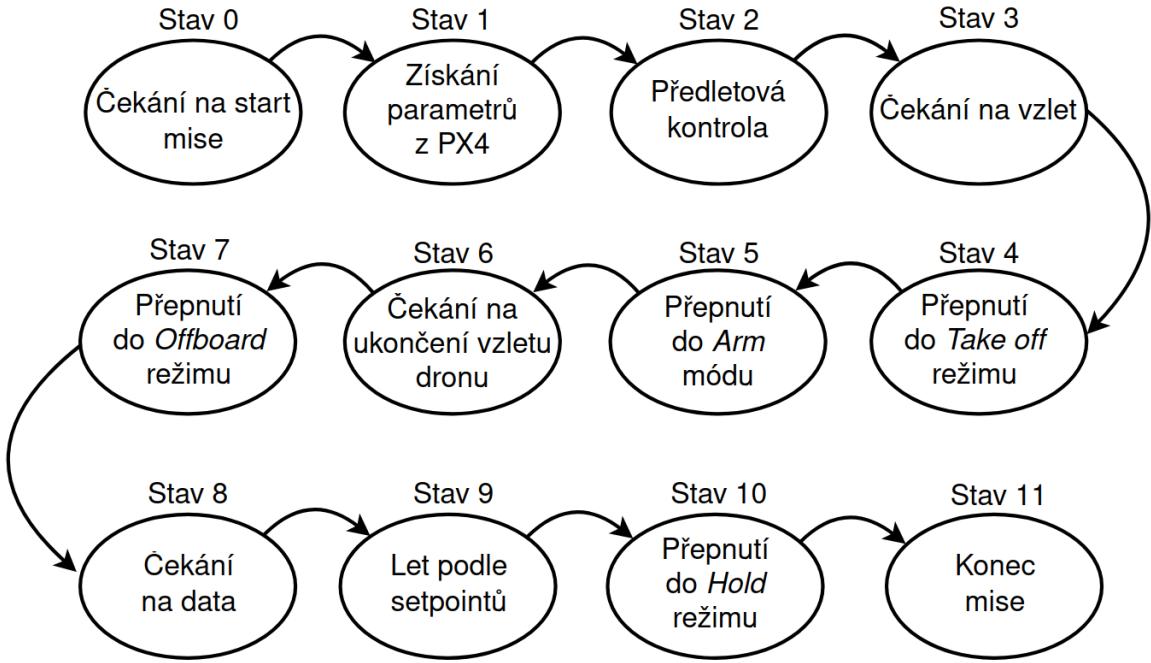
Dron je na zemi ve vypnutém stavu.

Podmínka přechodu: Uběhne uživatelem stanovený čas.

Stav 1: Získání parametrů z PX4

Dron vyžádá aktivní parametry ze systému PX4 kvůli tomu, aby bylo možné měnit parametry pomocí komunikačního uzlu Mavros.

Podmínka přechodu: Komunikační uzel Mavros pošle nadřazenému uzlu, který řídí misi informaci, že systém PX4 poskytl všechny aktivní parametry.



Obr. 6.1: Stavový automat řídící robotickou misi.

Stav 2: Předletová kontrola

V tomto kroku dron vykoná nezbytné kontroly letové způsobilosti jako je kontrola korektní geolokace (GPS *fix*) a nastaví důležité parametry robotické mise jako je výška vzletu, horizontální rychlosť, povolení *offboard* módu a chování dronu po výpadku *offboard* módu.

Podmínka přechodu: Systém PX4 odešle informaci, že všechny parametry byly změněny správně a GPS na dronu je aktivní.

Stav 3: Čekání na vzlet

Dron je na zemi ve vypnutém stavu.

Podmínka přechodu: Uběhne uživatelem stanovený čas.

Stav 4: Přepnutí dronu do *take off* letového režimu

Nadřazený systém pošle požadavek pro změnu letového režimu na *take off* letový režim.

Podmínka přechodu: Systém PX4 pošle odpověď, že letový režim dronu byl úspěšně změněný.

Stav 5: Přepnutí dronu do *arm* módu

Nadřazený systém pošle požadavek pro změnu letového režimu.

Podmínka přechodu: Systém PX4 pošle odpověď, že mód dronu byl úspěšně změněný na *arm*.

Stav 6: Čekání na ukončení vzletu dronu

Dron vykonává sekvenci úkonů pro vzlet.

Podmínka přechodu: Nadřazený systém detekuje změnu letového režimu na *hold* letový režim z důvodu ukončení vzletu.

Stav 7: Přepnutí dronu do *offboard* letového režimu

Nadřazený systém pošle požadavek pro změnu letového režimu na *offboard* letový režim.

Podmínka přechodu: Systém PX4 pošle odpověď, že letový režim dronu byl úspěšně změněný.

Stav 8: Čekání na data

Dron se vznáší v *offboard* letovém režimu na místě, kde byla ukončená vzletová sekvence. Tento stav je aktivní jenom v misích, kde jsou data pro let dronu generována a vysílána jiným ROS 2 uzlem, například v misi pro sledování dynamického objektu.

Podmínka přechodu: ROS 2 uzel pro řízení mise přijme první paket dat.

Stav 9: Let podle setpointů

Dron v *offboard* letovém režimu vykonává jednotlivé úkony robotické mise. Nadřazený ROS 2 uzel pro řízení mise posílá dronu buď poziční waypoints (lokální nebo globální), nebo řídí přímo lineární rychlosti v osách *X*, *Y*, *Z* a rychlosť rotace kolem osy *Z*.

Podmínka přechodu: Nadřazený uzel pro řízení mise již nepřímá data pro let dronu (po určitou dobu definovanou uživatelem), nebo dron vykoná všechny úkony robotické mise.

Stav 10: Přepnutí do *hold* režimu

Dron se v tomto stavu vznáší v *offboard* letovém režimu na místě, kde ukončil poslední úkon robotické mise. V této fázi je nutný zásah pilota kvůli tomu, že nechceme, aby dron zahájil přistávací manévr bez vědomí a povolení pilota.

Podmínka přechodu: Pilot dronu převezme řízení dronu.

Stav 11: Konec mise

Pilot dronu přistává, nebo zahajuje další autonomní misi.

7 Výsledky simulace

7.1 Gzebo svět

Na základě diplomové práce studenta bc. Miloše Cihlářa

7.2 Let po waypointech

Funkčnost simulace jsme demonstrovali na robotické misi jejímž cílem byl let dronu po waypointech. Pro tento účel jsme implementovali dvě alternativy, a to řízení dronu na lokální, a globální waypointy. V obou případech jsou data o trajektorii letu známá před misí. Tyto data se načítají z `.yaml` souboru jako ROS 2 parametry, takže je možné je měnit bez komplikace celého zdrojového kódu mise.

7.2.1 Let podle lokálních waypointů

Zprávy typu `geometry_msgs::msg::PoseStamped` jsou z nadřazeného uzlu publikovány do ROS 2 témy (*topic*) `/mavros/setpoint_position/local` s frekvencí 10 Hz (pro aktivitu *offboard* letového režimu musí být zprávy posílány s frekvencí > 2 Hz).

7.2.2 Let podle globálních waypointů

Zprávy typu `geographic_msgs::msg::GeoPoseStamped` jsou z nadřazeného uzlu publikovány do ROS 2 témy (*topic*) `/mavros/setpoint_position/global` s frekvencí 10 Hz.

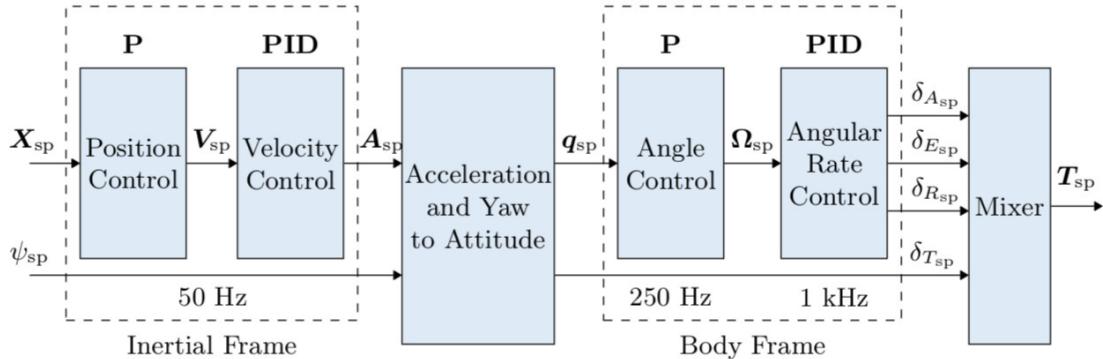
7.3 Let podle setpointů rychlosti

Další robotická mise, na které jsme otestovali funkčnost simulace je založená na řízení dronu podle lineární rychlosti v osách X, Y, Z a rychlosti rotace kolem osy Z.

Řízení tohoto typu je vhodné v případě, že regulační smyčka v PX4 (obrázek 7.1) nepostačuje našim požadavkům na regulaci a tudíž musíme implementovat vlastní, komplexnější řídící strukturu. Obrázek 7.1 zobrazuje, že při řízení dronu pomocí setpointů rychlosti vynecháme PX4 poziční regulátor¹ z řídící struktury.

¹Poziční regulátor V PX4 je složený z P složky a saturace

Další možnosti, kterou jsme neimplementovali je řízení dronu pomocí setpointů zrychlení. Tímto krokem by jsme vyneschali taky rychlostní regulátor² a měli by jsme volnou ruku při implementaci komplexnějších řídících struktur.



Obr. 7.1: Řídící struktura systému PX4. [3]

Zprávy typu `geometry_msgs::msg::TwistStamped` jsou z nadřazeného uzlu publikovány do ROS 2 témy (*topic*) `/mavros/setpoint_velocity/cmd_vel` s frekvencí 10 Hz.

7.4 Mise pro sledování dynamického objektu

Funkčnost simulace jsme také demonstrovali na složitější robotické misi jejímž cílem bylo sledování dynamického objektu.

Pro tuto úlohu jsme využili data z měření radiace pomocí všeobecného radiometrického detektoru, který byl umístěný na pozemním robotu. Robot pomocí částicového filtru (*particle filter*) estimoval pozici radioaktivního zářiče a na základě této estimace se k zdroji radiace přibližoval.

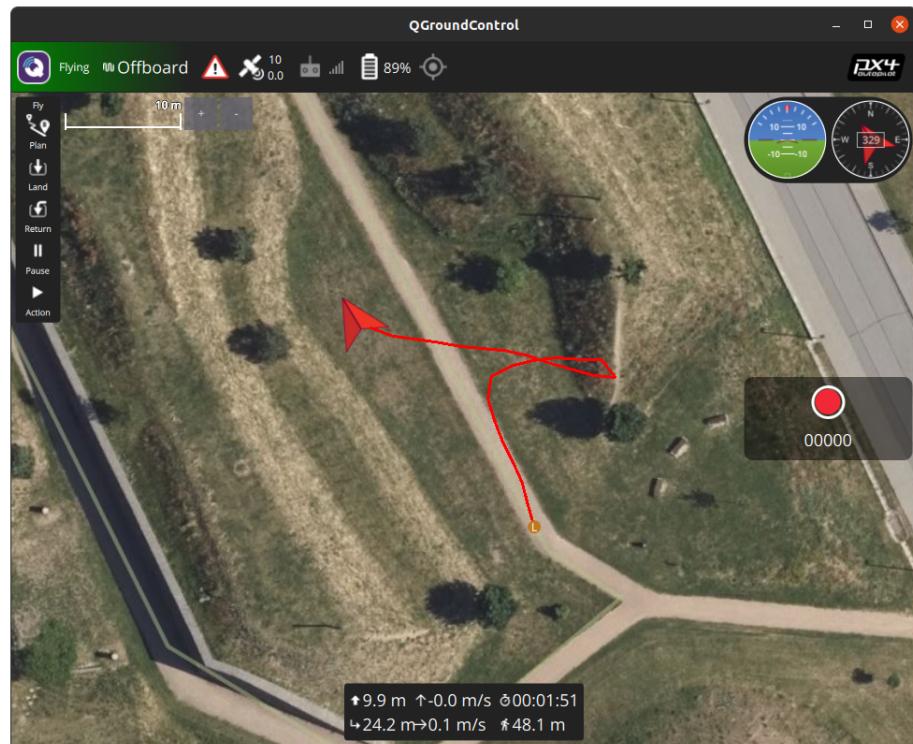
Data z měření jsme měli k dispozici jako `rosbag` soubor³, který obsahuje zprávy typu `geographic_msgs::msg::GeoPoseStamped` a publikuje je do ROS 2 tématu `/estimated_source_location`. Tyto zprávy obsahují globální souřadnice estimovaného zdroje radiace podle standardu WGS 84. Dron se v průběhu celé mise pohybuje v konstantní výšce, která je definovatelná uživatelem pomocí ROS 2 parametru.

7.5 Simulace mise s více drony

DOKONČIT

²Rychlostní regulátor V PX4 je složený z PID složky a saturace

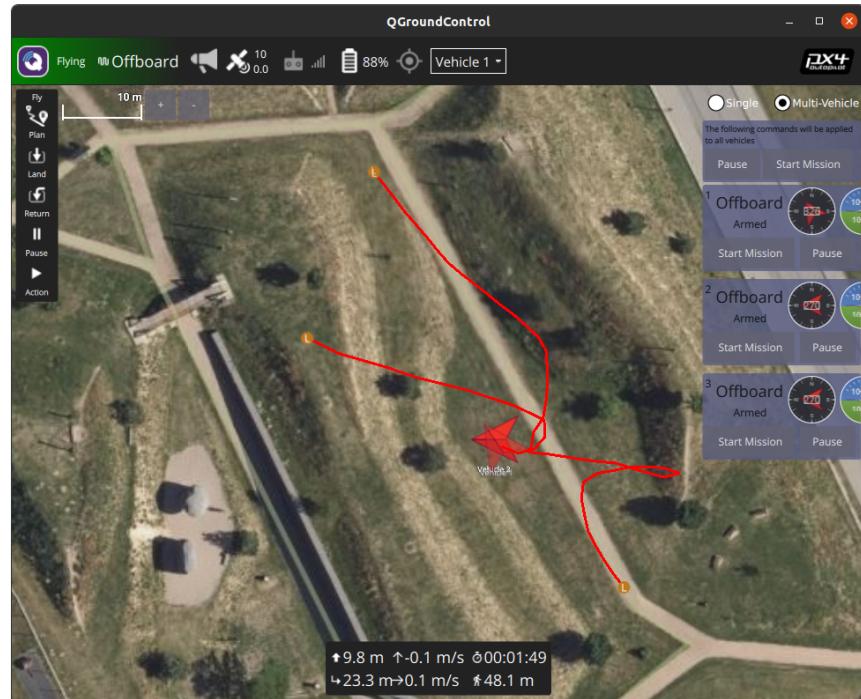
³Rosbag soubor slouží na zálohování dat v podobě ROS 2 témat (*topic*)



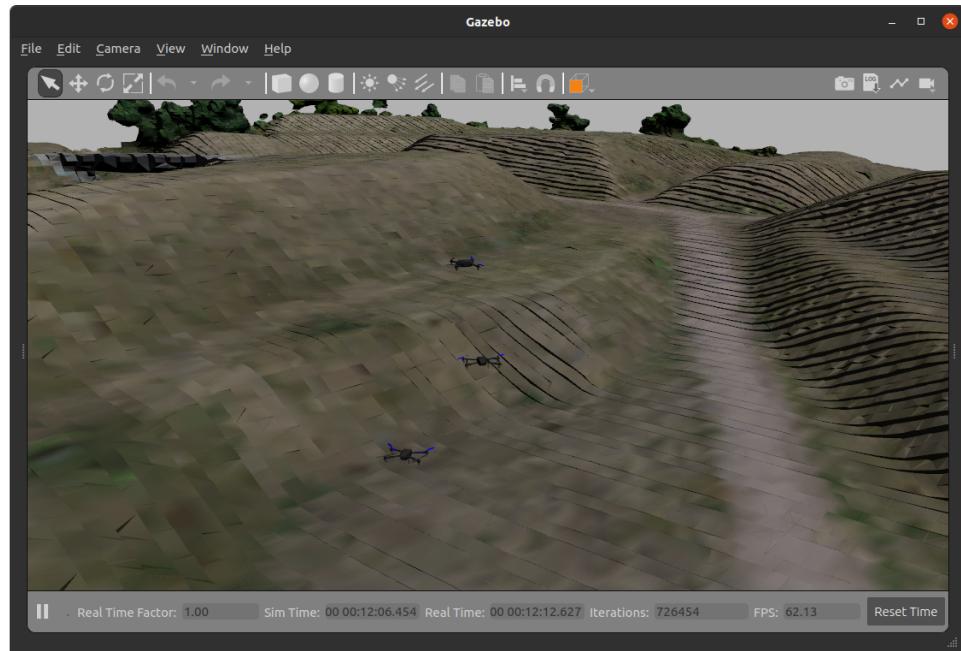
Obr. 7.2: Software QGroundControl s dronem v *offboard flight mode*.



Obr. 7.3: Software QGroundControl s dronem v *offboard flight mode*.



Obr. 7.4: Software QGroundControl s dronem v *offboard flight mode*.



Obr. 7.5: Software QGroundControl s dronem v *offboard flight mode*.

Závěr

ZÁVER DOKONČIT

Tato semestrální práce se zabývala problematikou simulace misí bezpilotních letadel ve virtuálním prostředí ROS2/Gazebo. V první kapitole je popsána topologie dronu z pohledu řídícího systému. Zaobírá se jak řídící jednotkou pro nízkoúrovňové řízení dronu, tak palubním počítačem pro řízení složitých autonomních misí.

V další kapitole je popsán firmware pro řídící jednotku Pixhawk. Kapitola pojednává o architektuře firmware PX4 a o možnostech propojení a komunikace s ostatními periferiemi. Nedílnou součástí ekosystému PX4 je software QGroundControl pro vzdálenou kontrolu letu, plánování různých typů bezpilotních misí a nastavení dronu.

Důležitou částí práce je popis komunikace mezi ROS 2 a PX4 firmware přes Fast RTPS (DDS[®]) bridge. Práce rozebírá samotný DDS[®] (Data-Distribution Service[®]) middleware a jeho komunikační protokol RTPS (Real Time Publish Subscribe protocol) pro kritické aplikace, základní vlastnosti DDS[®] a hlavně propojení mezi ROS 2 a DDS[®] komunikačním middleware.

Poslední kapitola popisuje simulaci ve virtuálním ekosystému PX4 - Gazebo - ROS 2. Shrnuje instalaci všech závislostí a postup sestavení simulačního prostředí (*workspace*). Kapitola se zabývá tvořením a plánováním autonomní robotické mise, která je realizována pomocí ROS 2 balíčku a její testováním v simulačním prostředí. Cílem jednoduché autonomní mise je ovládat dron v tzv. *offboard* módu pomocí ROS 2 balíčku. Podařilo se nám odsimulovat bezpilotní misi složenou z vzletu dronu, přeletu přes několik relativních souřadnic a následného přistání.

Literatura

- [1] MRS UAV System - open source platform for UAV research. *Multi-robot Systems Group* [online]. Praha: Multi-robot Systems Group, c2022 [cit. 2022-01-01]. Dostupné z: <http://mrs.felk.cvut.cz/>
- [2] The open standards for drone hardware. *Pixhawk* [online]. San Francisco: Dronecode, 2018 [cit. 2022-01-01]. Dostupné z: <https://pixhawk.org/>
- [3] *PX4 User Guide: PX4 Autopilot User Guide (master)* [online]. San Francisco: Dronecode, 2021 [cit. 2022-03-23]. Dostupné z: <https://docs.px4.io/master/en/>
- [4] Betaflight Home: Are you ready to fly?. *Betaflight* [online]. Betaflight, c2022 [cit. 2022-03-25]. Dostupné z: <https://betaflight.com/>
- [5] Ardupilot, versatile, trusted, open. *Ardupilot* [online]. Ardupilot, c2022 [cit. 2022-01-03]. Dostupné z: <https://ardupilot.org/>
- [6] PX4: Open Source Autopilot For Drone Developers. *PX4* [online]. San Francisco: Dronecode, c2021 [cit. 2022-01-03]. Dostupné z: <https://px4.io/>
- [7] The 3-Clause BSD License: BSD-3-Clause. *Open Source Initiative* [online]. West Hollywood: Open Source Initiative, 2014 [cit. 2021-11-17]. Dostupné z: <https://opensource.org/licenses/BSD-3-Clause>
- [8] INAV: Navigation capable flight controller. *GithHub* [online]. California: GitHub, 2021 [cit. 2022-03-27]. Dostupné z: <https://github.com/iNavFlight/inav>
- [9] QGroundControl. *QGroundControl* [online]. San Francisco: Dronecode, 2019 [cit. 2021-11-24]. Dostupné z: <http://qgroundcontrol.com/>
- [10] QGroundControl User Guide. *QGroundControl docs* [online]. San Francisco: Dronecode, 2021 [cit. 2021-12-21]. Dostupné z: <https://docs.qgroundcontrol.com/master/en/PlanView/Pattern.html>
- [11] ROS 2 middleware interface: Mapping between DDS and ROS concepts. *ROS 2 Design* [online]. California: Open Source Robotics Foundation, 2017 [cit. 2021-12-21]. Dostupné z: https://design.ros2.org/articles/ros_middleware_interface.html
- [12] eProsima Fast Buffers. *eProsima the middleware experts* [online]. Madrid: eProsima, 2014 [cit. 2021-12-23]. Dostupné z: <https://www.eprosima.com/docs/fast-buffers/0.3.0/html/index.html>

- [13] The Real-time Publish-Subscribe Protocol (RTPS) DDS Interoperability Wire Protocol Specification. *Object Management Group* [online]. Milford: OMG, c2021 [cit. 2021-11-25]. Dostupné z: <https://www.omg.org/spec/DDS-RTSPS/2.3>
- [14] Data Distribution Service (DDS). *Object Management Group* [online]. Milford: OMG, 2018 [cit. 2021-11-25]. Dostupné z: <https://www.omg.org/omg-dds-portal/>
- [15] What is DDS? *DDS Foundation: Join DDS Foundation* [online]. Milford: DDS Foundation, c1997-2021 [cit. 2021-11-25]. Dostupné z: <https://www.dds-foundation.org/what-is-dds-3/>
- [16] Introduction to Publish/Subscribe. *Real-Time Innovations: RTI / Intelligent, distributed and real world systems* [online]. Sunnyvale: RTI, c2020 [cit. 2021-11-28]. Dostupné z: https://community.rti.com/static/documentation/connext-dds/6.0.1/doc/manuals/connext_cpp11/intro_pubsub_cpp.html
- [17] Why choose DDS?: Industry standards build on top of DDS. *DDS Foundation: Join DDS Foundation* [online]. Milford: DDS Foundation, c1997-2021 [cit. 2021-11-25]. Dostupné z: <https://www.dds-foundation.org/why-choose-dds/>
- [18] RTPS Introduction: What is RTPS? *eProsim: The Middleware experts* [online]. Madrid: eProsim, c2013-2021 [cit. 2021-11-29]. Dostupné z: <https://www.eprosima.com/index.php/resources-all/whitepapers/rtps>
- [19] ROS on DDS: Technical Credibility of DDS. *ROS 2 Design* [online]. California: Open Source Robotics Foundation, c2021 [cit. 2021-11-29]. Dostupné z: https://design.ros2.org/articles/ros_on_dds.html
- [20] About different ROS 2 DDS/RTPS vendors: Supported RMW implementations. *ROS 2 Documentation: Foxy* [online]. California: Open Robotics, c2021 [cit. 2021-12-09]. Dostupné z: <https://docs.ros.org/en/foxy/Concepts/About-Different-Middleware-Vendors.html>
- [21] *Gazebo* [online]. California: Open Source Robotics Foundation, c2014 [cit. 2021-12-30]. Dostupné z: <http://gazebosim.org/>
- [22] THACKSTON, Allison. Ignition vs Gazebo. *Allison Thackston* [online]. San Francisco: Thackston, c2014-2021 [cit. 2021-12-31]. Dostupné z: <https://www.allisonthackston.com/articles/ignition-vs-gazebo.html>

- [23] How to Install Gradle on Ubuntu 20.04. *Linuxize* [online]. Linuxize, 2020 [cit. 2022-01-02]. Dostupné z: <https://linuxize.com/post/how-to-install-gradle-on-ubuntu-20-04/>
- [24] Installing ROS 2 via Debian Packages. *Documentation: Foxy* [online]. California: Open Robotics, c2021 [cit. 2022-01-02]. Dostupné z: <https://docs.ros.org/en/foxy/Installation/Ubuntu-Install-Debians.html>
- [25] PX4_ros2_missions. *Github* [online]. California: GitHub, 2022 [cit. 2022-01-03]. Dostupné z: https://github.com/DavidLindtner/px4_ros2_missions
- [26] MAVLink: MAVLink Developer Guide [online]. San Francisco: Dronecode, [2022] [cit. 2022-04-29]. Dostupné z: <https://mavlink.io/en/>
- [27] ROS: MAVROS [online]. California: Open Robotics, 2018 [cit. 2022-05-05]. Dostupné z: <http://wiki.ros.org/mavros>
- [28] INDRIYANTO, T., A. R. RIZKI, M. L. HARIYADIN, M. F. AKBAR a A. A. A. SYAFI. Centralized swarming UAV using ROS for collaborative missions [online]. In: . 2020, s. 030012- [cit. 2022-05-05]. Dostupné z: doi:10.1063/5.0002616
- [29] KARNEY, Charles. GeographicLib: GeographicLib library. *Sourceforge* [online]. New Jersey: Sourceforge, 2021 [cit. 2022-05-06]. Dostupné z: <https://geographiclib.sourceforge.io/html/index.html>

Seznam symbolů a zkrátek

IMU	Inertial Measurement Unit
MSP	Multiwii Serial Protocol
SITL	Software In The Loop
HITL	Hardware In The Loop
CDR	Common Data Representation
API	Application programming interface
DDS®	Data-Distribution Service®
RTPS	Real Time Publish Subscribe protocol
DDSI-RTPS™	DDS Interoperability Wire Protocol™
QoS	Quality of Service
UDP	User datagram protocol
VTOL	Vertical Take-Off and Landing
RTOS	Real-Time Operating System
IDL	Interactive Data Language
SDK	Software development kit

Seznam příloh

A Přílohy	64
B Obsah elektronické přílohy	65

A Přílohy

B Obsah elektronické přílohy

Elektronická příloha je často nedílnou součástí semestrální nebo závěrečné práce. Vkládá se do informačního systému VUT v Brně ve vhodném formátu (ZIP, PDF ...).

Nezapomeňte uvést, co čtenář v této příloze najde. Je vhodné okomentovat obsah každého adresáře, specifikovat, který soubor obsahuje důležitá nastavení, který soubor je určen ke spuštění, uvést nastavení kompilátoru atd. Také je dobré napsat, v jaké verzi software byl kód testován (např. Matlab 2018b). Pokud bylo cílem práce vytvořit hardware zařízení, musí elektronická příloha obsahovat veškeré podklady pro výrobu (např. soubory s návrhem DPS v Eagle).

Pokud je souborů hodně a jsou organizovány ve více složkách, je možné pro výpis adresářové struktury použít balíček `dirtree`.

```
/..... kořenový adresář přiloženého archivu
  logo ..... loga školy a fakulty
    BUT_abbreviation_color_PANTONE_EN.pdf
    BUT_color_PANTONE_EN.pdf
    FEEC_abbreviation_color_PANTONE_EN.pdf
    FEKT_zkratka_barevne_PANTONE_CZ.pdf
    UTKO_color_PANTONE_CZ.pdf
    UTKO_color_PANTONE_EN.pdf
    VUT_barevne_PANTONE_CZ.pdf
    VUT_symbol_barevne_PANTONE_CZ.pdf
    VUT_zkratka_barevne_PANTONE_CZ.pdf
  obrazky ..... ostatní obrázky
    soucastky.png
    spoje.png
    ZlepseWilsonovoZrcadloNPN.png
    ZlepseWilsonovoZrcadloPNP.png
  pdf ..... pdf stránky generované informačním systémem
    student-desky.pdf
    student-titulka.pdf
    student-zadani.pdf
  text ..... zdrojové textové soubory
    literatura.tex
    prilohy.tex
    reseni.tex
    uvod.tex
    vysledky.tex
    zaver.tex
    zkratky.tex
  sablona-obhaj.tex ..... hlavní soubor pro sazbu prezentace k obhajobě
  sablona-prace.tex ..... hlavní soubor pro sazbu kvalifikační práce
  thesis.sty ..... balíček pro sazbu kvalifikačních prací
```