

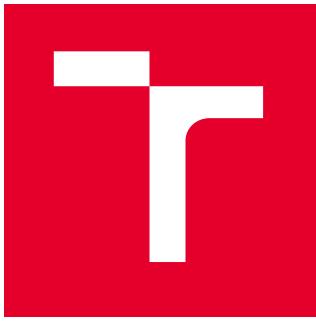
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

SEMESTRÁLNÍ PRÁCE

Brno, 2021

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

SEMESTRÁLNÍ PRÁCE

SEMESTRAL THESIS

AUTOR PRÁCE

AUTHOR

Bc. David Lindtner

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Petr Gábrlík, Ph.D.

BRNO 2021

Semestrální 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 bezpilotních letadel v prostředí ROS2/Gazebo a spočívá především ve výběru a testování vhodného programového vybavení.

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. Zvolené řešení zprovozněte a demonstrejte na jednoduché misi.

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: 20.9.2021

Termín odevzdání: 3.1.2022

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

doc. Ing. Petr Fiedler, Ph.D.

předseda rady studijního programu

Obsah

| | |
|--|-----------|
| Úvod | 6 |
| 1 Topologie řídícího systému dronu | 7 |
| 1.1 Řídící jednotka Pixhawk | 7 |
| 1.2 Palubní počítač | 8 |
| 1.3 Software pro řízení bezpilotních letadel | 9 |
| 1.3.1 ArduPilot | 9 |
| 2 PX4 autopilot | 10 |
| 2.1 Architektura PX4 | 10 |
| 2.1.1 PX4 jako samotný letový ovladač | 12 |
| 2.1.2 Letový ovladač a počítač pro řízení mise | 12 |
| 2.2 Licence | 13 |
| 2.3 QGround Control | 14 |
| 2.3.1 Nastavení konstrukce dronu | 15 |
| 2.3.2 Plánování bezpilotní mise | 15 |
| 3 Propojení PX4 a ROS 2 | 19 |
| 3.1 Struktura komunikace v ROS 2 | 19 |
| 3.2 Vnitřní komunikace v PX4 | 19 |
| 3.3 MicroRTPS agent a klient | 19 |
| 3.3.1 MicroRTPS klient | 20 |
| 3.3.2 MicroRTPS agent | 20 |
| 3.3.3 Komunikace mezi microRTPS agentem a klientem | 20 |
| 3.4 DDS®/RTPS middleware | 21 |
| 3.4.1 DDS® a ROS 2 | 22 |
| 3.4.2 Orientace na data (data centricity) | 22 |
| 3.4.3 Globální datový prostor | 22 |
| 3.4.4 Základní architektura DDS® | 23 |
| 3.4.5 Quality of Service | 23 |
| 3.4.6 RTPS protokol | 24 |
| 3.4.7 Dynamické hledání bodů v síti | 25 |
| 3.4.8 Využití DDS® v praxi | 25 |
| 4 Simulace ve virtuálním prostředí | 26 |
| 4.1 SITL simulační prostředí | 26 |
| 4.1.1 Seznam podporovaných simulátorů | 27 |
| 4.1.2 Gazebo | 27 |

| | | |
|---------------------------------|---|-----------|
| 4.1.3 | Ignition Gazebo | 28 |
| 4.2 | Instalace virtuálního prostředí | 28 |
| 4.2.1 | Instalace PX4 simulačního prostředí | 28 |
| 4.2.2 | Instalace ROS 2 Foxy | 29 |
| 4.2.3 | Nastavení ROS 2 prostředí | 29 |
| 4.2.4 | PX4 - ROS 2 bridge | 30 |
| 4.3 | ROS 2 misie | 31 |
| 4.3.1 | Offboard flight mode | 31 |
| 4.3.2 | Výsledky simulace | 31 |
| Závěr | | 33 |
| Literatura | | 34 |
| Seznam symbolů a zkratek | | 37 |

Seznam obrázků

| | | |
|-----|---|----|
| 1.1 | Řídící jednotka Pixhawk 4 | 8 |
| 2.1 | Komplexní strukturální architektura PX4 software | 11 |
| 2.2 | Jednoduchý systém PX4 založený na letovém ovladači (flight controller) . | 12 |
| 2.3 | Systém PX4 založený na letovém ovladači (flight controller) a palubním počítači pro řízení mise | 13 |
| 2.4 | Základní obrazovka QGroundControl | 14 |
| 2.5 | Základní obrazovka QGroundControl | 15 |
| 2.6 | Nastavení bezpilotní mise pro průzkum prostředí | 16 |
| 2.7 | Nastavení bezpilotní mise pro skenování konstrukcí | 17 |
| 2.8 | Nastavení bezpilotní mise pro skenování koridoru | 18 |
| 3.1 | Schéma komunikace mezi microRTPS agentem a klientem | 20 |
| 3.2 | Propojení DDS® middlewaru s aplikací a operačním systémem | 21 |
| 3.3 | Základní <i>publish/subscribe</i> struktura DDS® komunikace | 24 |
| 4.1 | Tok zpráv mezi simulátorem a PX4 | 26 |
| 4.2 | Diagram komunikace mezi PX4 a simulátorem | 27 |
| 4.3 | Software QGroundControl s dronem v <i>offboard flight mode</i> | 32 |
| 4.4 | Bezpilotní mise v simulátoru Gazebo | 32 |

Úvod

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 nízkourovňového řízení, tak z pohledu palubního počítače pro řízení složitých autonomních misí.

V práci bude popsaný 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 báze 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 4*.

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 *PX4* firmware přes ROS 2 *topic*. Komunikací mezi uvedenými platformami se budeme zabývat v následujících kapitolách

1.3 Software pro řízení bezpilotních letadel

Existují dvě hlavní větve vývoje firmware pro řídící jednotku Pixhawk. Jednou platformou je open source projekt ArduPilot [4], který podporuje velké množství bezpilotních prostředků a umožňuje vytváření autonomních misí. Druhým projektem je profesionální autopilot *PX4* [5]. 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 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 MAVROS¹

1.3.1 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). [4]

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

¹MAVROS je komunikační ovládač (*driver*) pro komunikaci mezi ROS a autopiloty s MAVLink protokolem

2 PX4 autopilot

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

2.1 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.

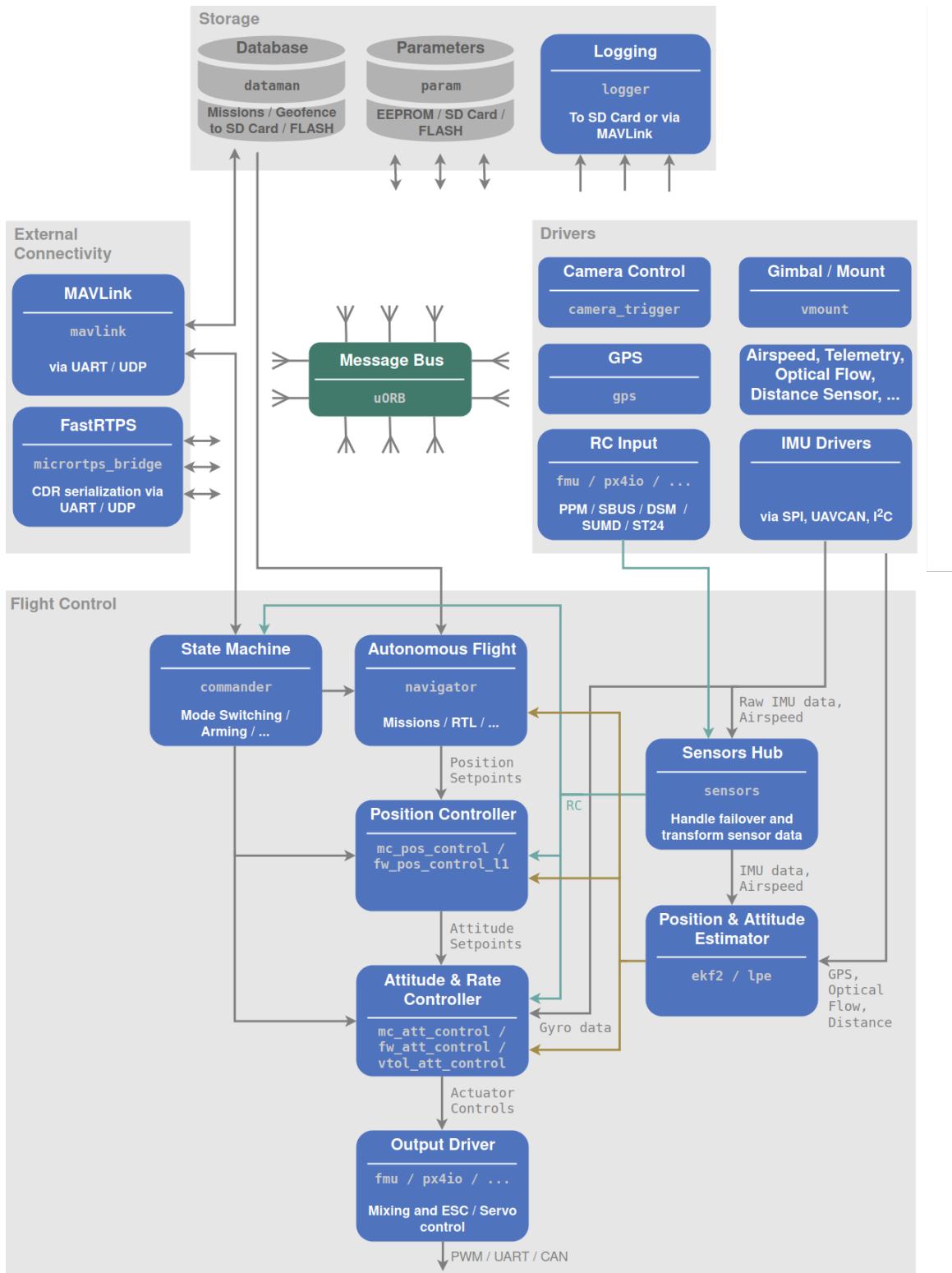
Všechny podporované typy vozidel (drony, letadla, čluny, rovery, ponorky atd.) sdílejí jedinou kódovou základnu, která má následující vlastnosti: [7]

- 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 2.1 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)

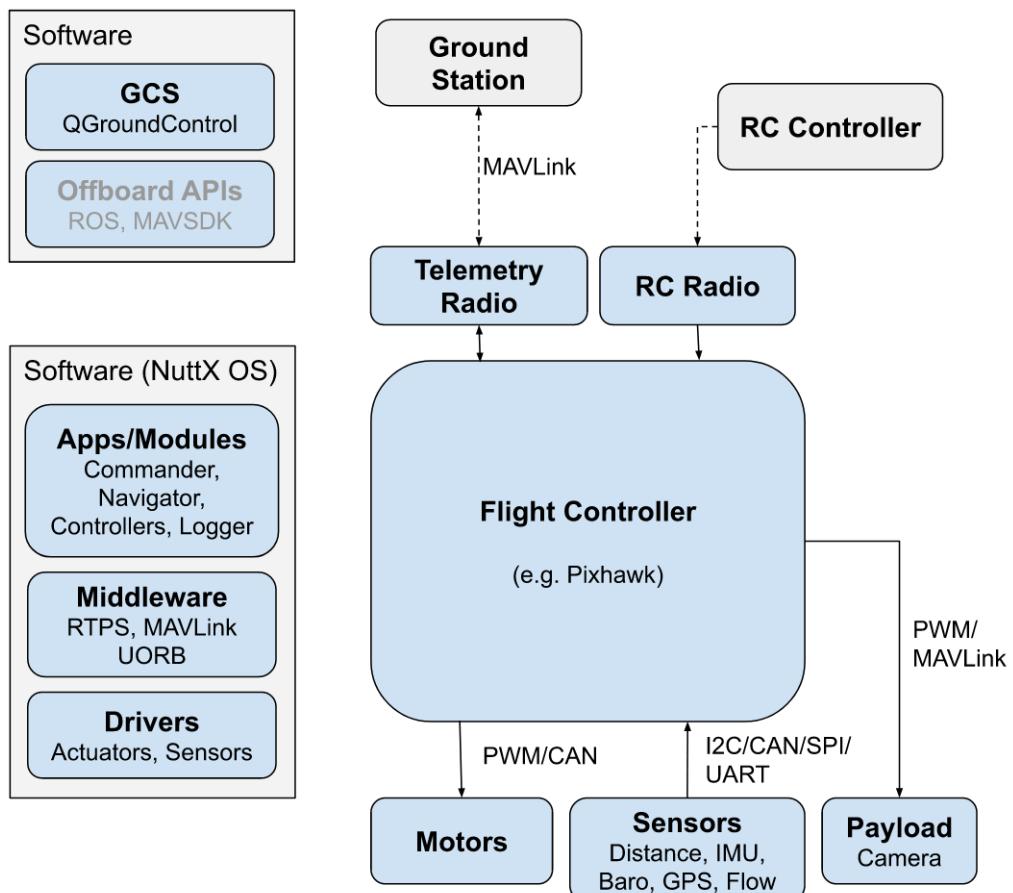


Obr. 2.1: Komplexní strukturální architektura PX4 software [7].

2.1.1 PX4 jako samotný letový ovladač

Na obrázku 2.2 je naznačená architektura systému založeného na letovém ovladači (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.



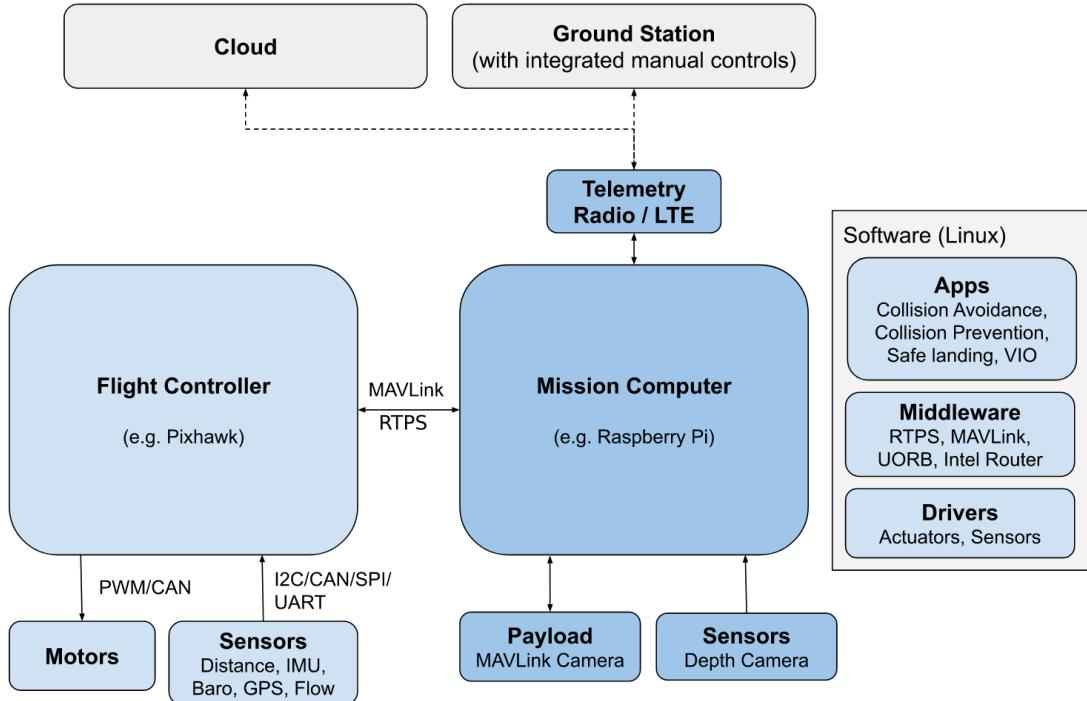
Obr. 2.2: Jednoduchý systém PX4 založený na letovém ovladači (flight controller) [8].

2.1.2 Letový ovladač a počítač pro řízení mise

Na obrázku 2.3 je zobrazená architektura systému PX4 založeného na letovém ovladači (řídící jednotce) a palubním počítači pro řízení mise. Řídící jednotka zabezpečuje sběr dat ze senzorů a ovládání akčních členů dronu (pohony, serva, ...). Komunikace mezi řídící jednotkou zabezpečuje protokol MAVlink, nebo RTPS (Real Time

Publish Subscribe protocol).

Palubní počítač pro řízení mise poskytuje pokročilé funkce, jako je vyhýbání se objektům a předcházení kolizím. 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 DDS® (Data-Distribution Service®) middleware.



Obr. 2.3: Systém PX4 založený na letovém ovladači (flight controller) a palubním počítači pro řízení mise [8].

2.2 Licence

Kód PX4 je zdarma k použití a úpravě za podmínek permisivní licence *BSD 3-clause license* [6].

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

1. Redistribuce zdrojového kódu musí obsahovat výše uvedenou poznámku o autorských právech, tento seznam podmínek a následující prohlášení o vyloučení odpovědnosti.

2. Redistribuce v binární formě musí reprodukovat výše uvedenou poznámku o autorských právech, tento seznam podmínek a následující prohlášení o vyloučení odpovědnosti v dokumentaci a/nebo jiných materiálech dodávaných s distribucí.

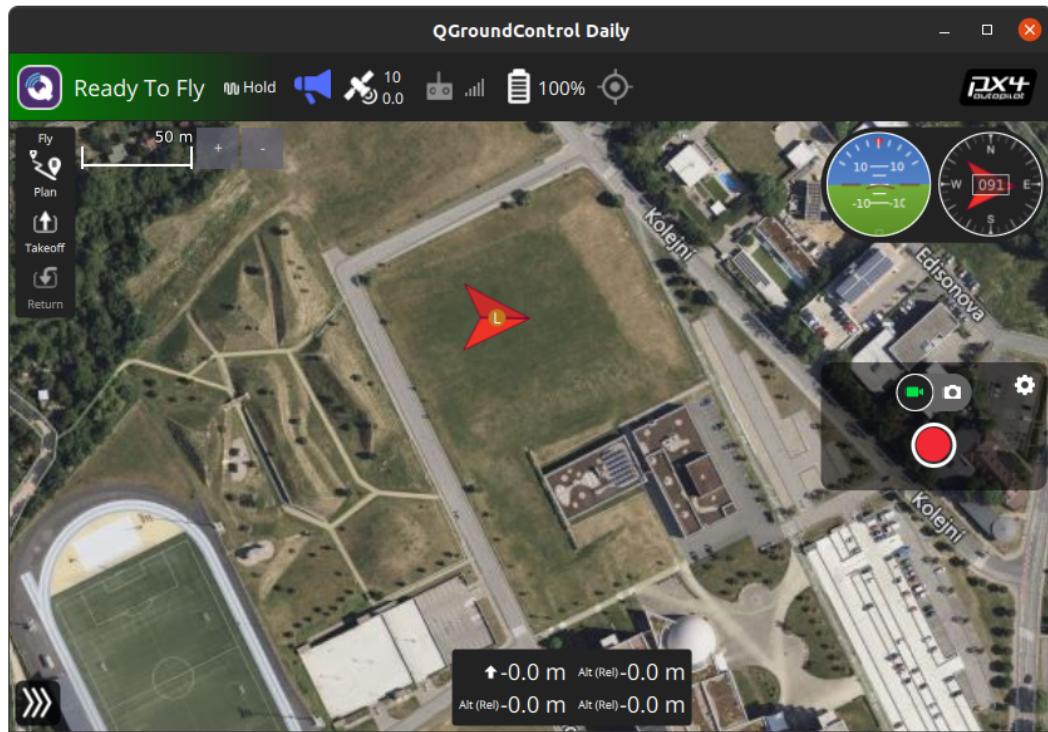
3. Jméno držitele autorských práv ani jména jeho přispěvatelů nesmí být použita k podpoře nebo propagaci produktů odvozených z tohoto software bez předchozího výslovného písemného povolení.

2.3 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 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 letového ovladače (*flight controller*), nastavit typ konstrukce dronu, ladit parametry regulace a vytvářet autonomní mise pomocí waypointů. Veškeré nastavení dronu a misí je jednoduché a velmi intuitivní.

Obrázek 2.4 zobrazuje základní obrazovku software QGroundControl.



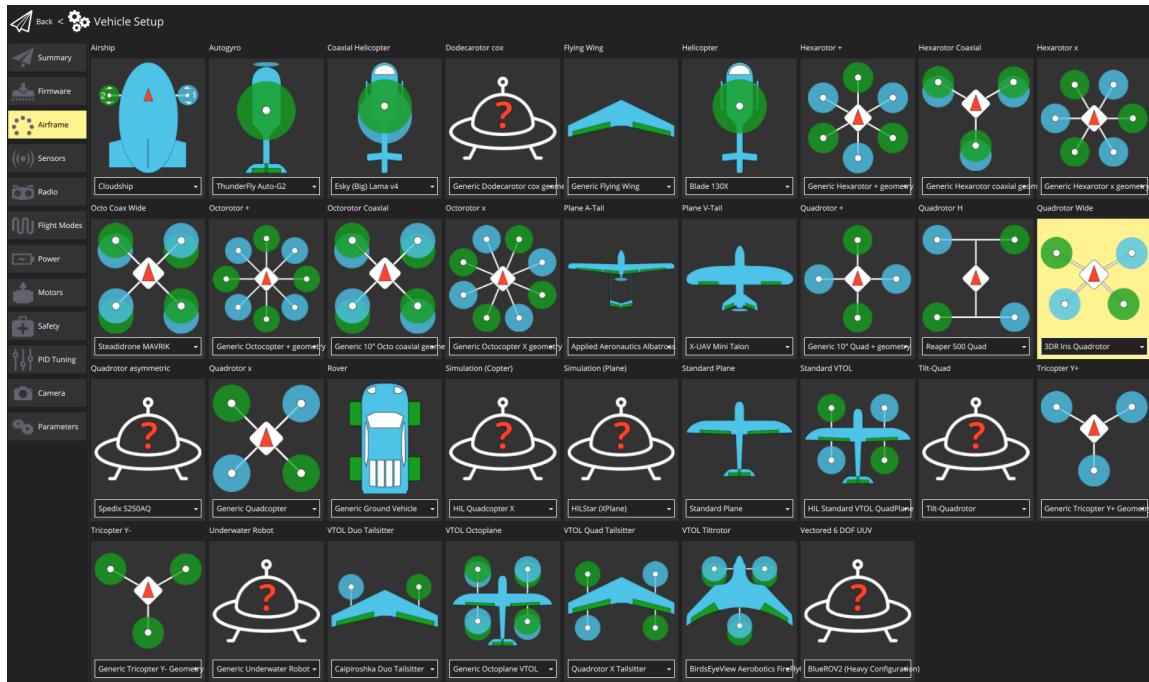
Obr. 2.4: Základní obrazovka QGroundControl.

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

2.3.1 Nastavení konstrukce dronu

Systém QGroundControl podporuje velké množství typů konstrukcí dronů, letadel, roverů a ponorek. Obrázek 2.5 zobrazuje všechny podporované typy zařízení, které dokáže systém PX4 ovládat.



Obr. 2.5: Základní obrazovka QGroundControl.

2.3.2 Plánování bezpilotní mise

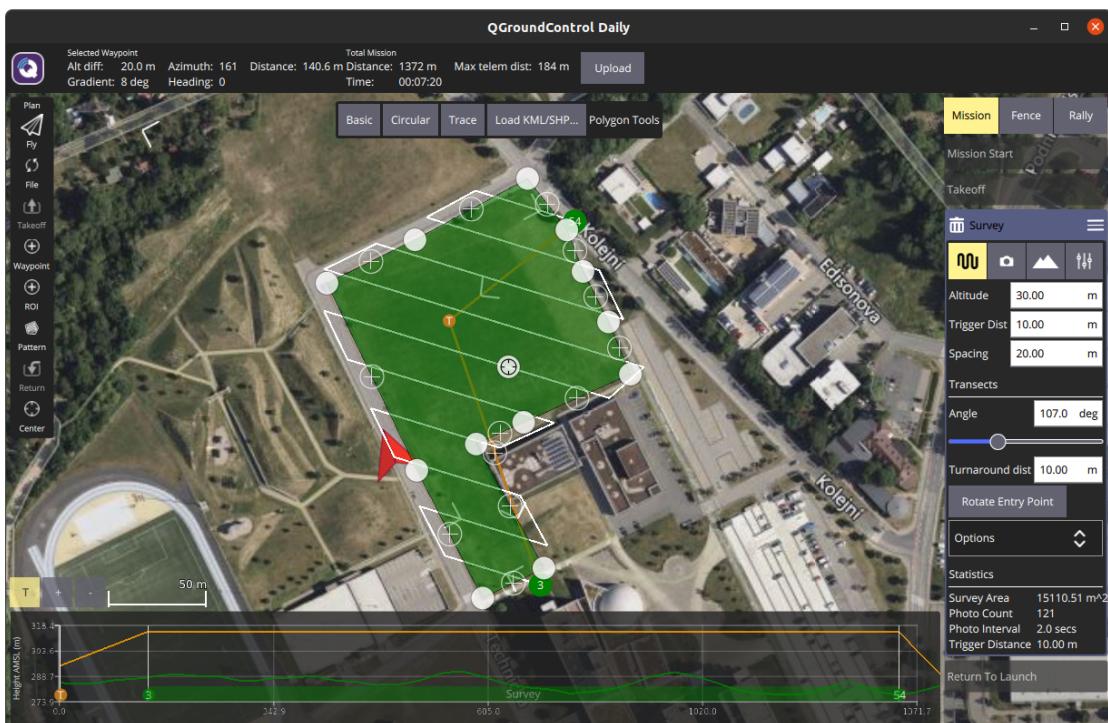
Software QGroundControl nabízí intuitivní plánování bezpilotních misí. Vlastní misi je možné naplánovat pomocí waypointů, nebo pomocí 3 přednastavených plánů pro bezpilotní mise: [10]

- 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 geotagging¹. Obrázek 2.6 zobrazuje nastavení autonomní mise s průzkumem prostředí.



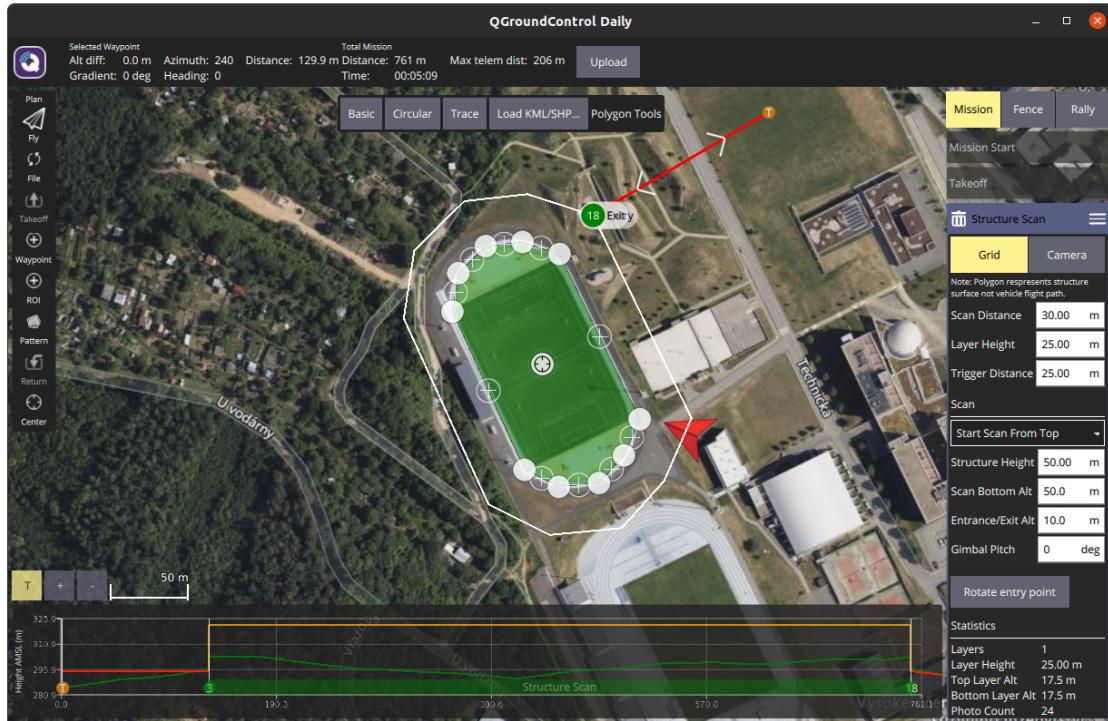
Obr. 2.6: Nastavení bezpilotní mise pro průzkum prostředí.

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

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

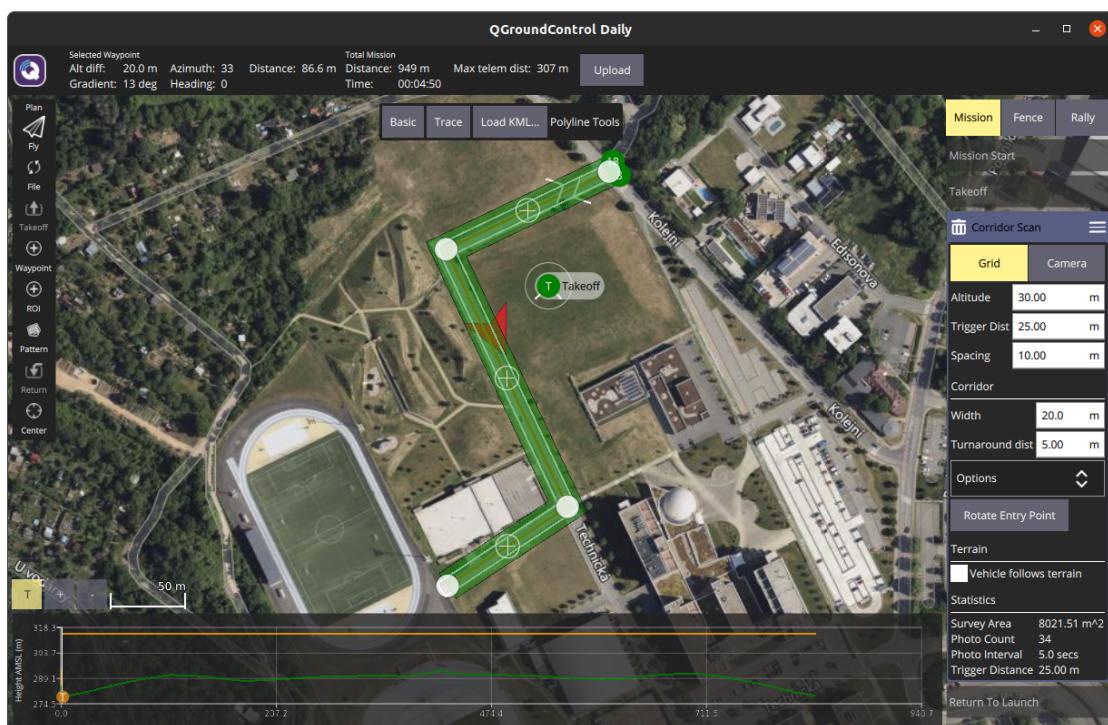
výškách tak, aby kamera směřovala vždy na budovu. Po zadání konkrétní kamery, objektivu a rozlišení snímkování [cm/px] si program sám dopočítá vhodnou letovou vzdálenost od budovy a výšku jednotlivých letových vrstev. Na obrázku 2.7 je zobrazeno nastavení autonomní mise pro skenování konstrukcí.



Obr. 2.7: 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 2.8 zobrazuje nastavení letového plánu pro skenování koridoru.



Obr. 2.8: Nastavení bezpilotní mise pro skenování koridoru.

3 Propojení PX4 a ROS 2

3.1 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 hlavní komunikační prostředek.

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]

3.2 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. [12]

V systému PX4 je definovaných 167 uORB zpráv. Úplný seznam je zveřejněný v uživatelském manuálu PX4: https://docs.px4.io/master/en/msg_docs/ [13]

3.3 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. [14]

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.

3.3.1 MicroRTPS klient

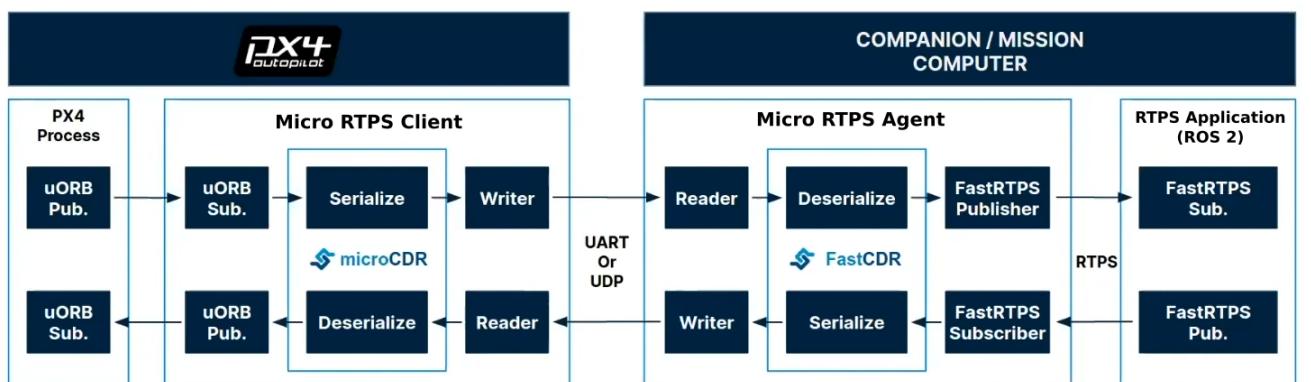
MicroRTPS klient je *deamon*, který běží na řídící jednotce (*flight controller*) PX4. Klient se přihlásí k odběru (*subscribe*) zpráv uORB publikovanými interními komponenty autopilota PX4 a odesílá zprávy agentovi (přes UART nebo UDP). Klient taky přijímá zprávy od agenta a publikuje je jako uORB zprávy do systému PX4. [14]

3.3.2 MicroRTPS agent

MicroRTPS agent běží jako *deamon* na palubním počítači. Agent sleduje aktuální 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.

3.3.3 Komunikace mezi microRTPS agentem a klientem

Jak je zobrazeno na obrázku 3.1 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.



Obr. 3.1: Schéma komunikace mezi microRTPS agentem a klientem. [14]

¹Common Data Representation je způsob nízkoúrovňové serializace dat mezi různými platformami [15]

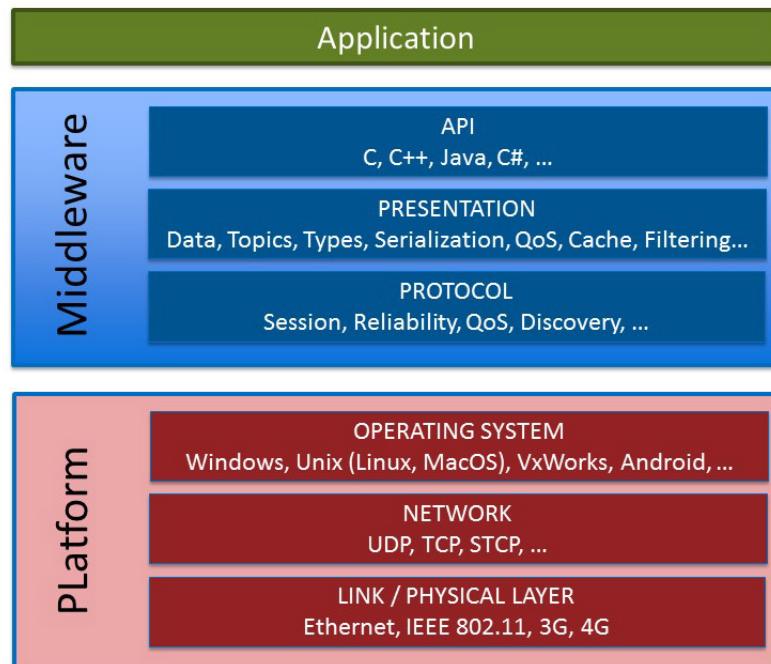
3.4 DDS®/RTPS middleware

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 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 [17].

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

Jak je zobrazeno na obr 3.2, 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.



Obr. 3.2: Propojení DDS[®] middlewareu s aplikací a operačním systémem [18].

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

3.4.1 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. [22]

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. [23]

ROS 2 podporuje tyto DDS® implementace:

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

3.4.2 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. [18]

3.4.3 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. [18]

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.

3.4.4 Základní architektura DDS®

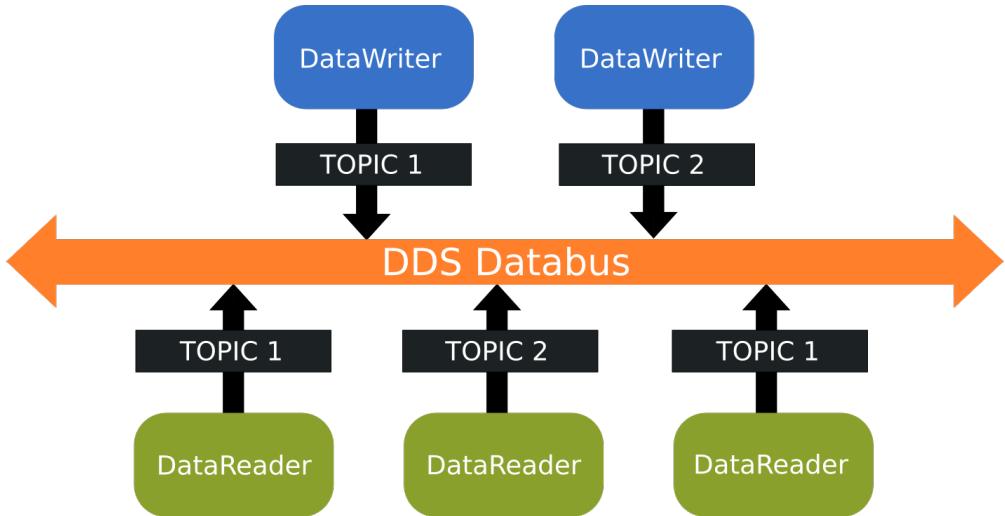
Nejzákladnějším komunikačním vzorem podporovaným DDS® je model publikování/přihlášení (*Publish/Subscribe*). *Publish/Subscribe* je komunikační model, kde producenti dat publikují data a spotřebitelé dat se přihlašují k odběru dat. Tito producenti a spotřebitelé o sobě nemusí vědět předem, protože se 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. [20]

Jak je zobrazeno na obrázku 3.3 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.

3.4.5 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, 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



Obr. 3.3: Základní *publish/subscribe* struktura DDS® komunikace [20].

hlediska zabezpečení DDS® řídí přístup, vyhodnocuje cesty toku dat a šifruje data za běhu. [18]

3.4.6 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. [16]

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: [21]

- *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.

3.4.7 Dynamické hledání bodů v síti

DDS® poskytuje dynamické zjišťování producentů a spotřebitelů dat. Dynamické hledání umožňuje rozšířitelnost 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®. [18]

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.

3.4.8 Využití DDS® v praxi

Data-Distribution Service® je základem některých průmyslových standardů: [19]

- 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. [22]

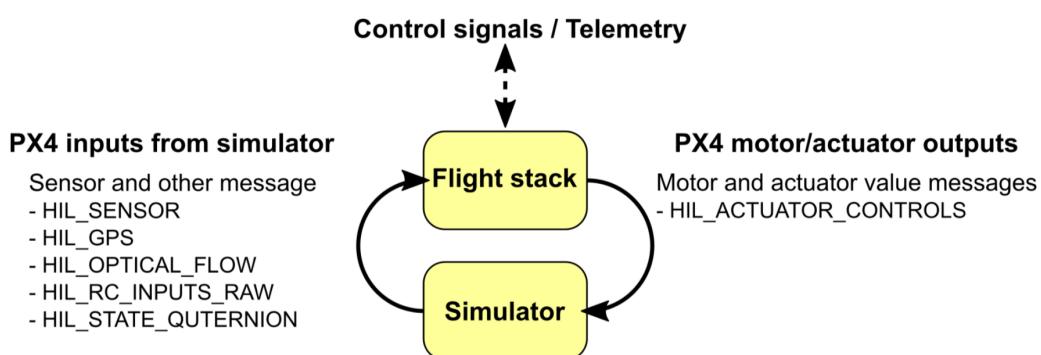
- Bitevní lodě
- Velké inženýrské instalace, jako jsou přehrady
- Finanční systémy
- Vesmírné systémy
- Letové systémy
- Systémy vlakových rozvaděčů

4 Simulace ve virtuálním prostředí

Simulátory umožňují letovému kódu PX4 ovládat počítačem modelované vozidlo v simulovaném „světě“. S tímto vozidlem můžete komunikovat stejně jako se skutečným vozidlem pomocí QGroundControl, offboard API nebo rádiového ovladače/gamepadu.

PX4 podporuje jak simulaci SITL (*Software In The Loop*), kde program letového ovladače běží na počítači, nebo simulaci HITL (*Hardware In The Loop*), kde simulační firmware běží na skutečném letovém ovladači (řídící jednotce). [24]

Všechny simulátory mohou komunikovat s firmware PX4 pomocí *Simulator MAVLink API*, které definuje sadu zpráv MAVLink. Na obrázku 4.1 je zobrazen tok zpráv mezi „simulovaným světem“ a mezi firmware PX4. Další možnost komunikace mezi simulátorem a PX4 firmware využívá *Micro-RTPS bridge*.



Obr. 4.1: Tok zpráv mezi simulátorem a PX4 [24].

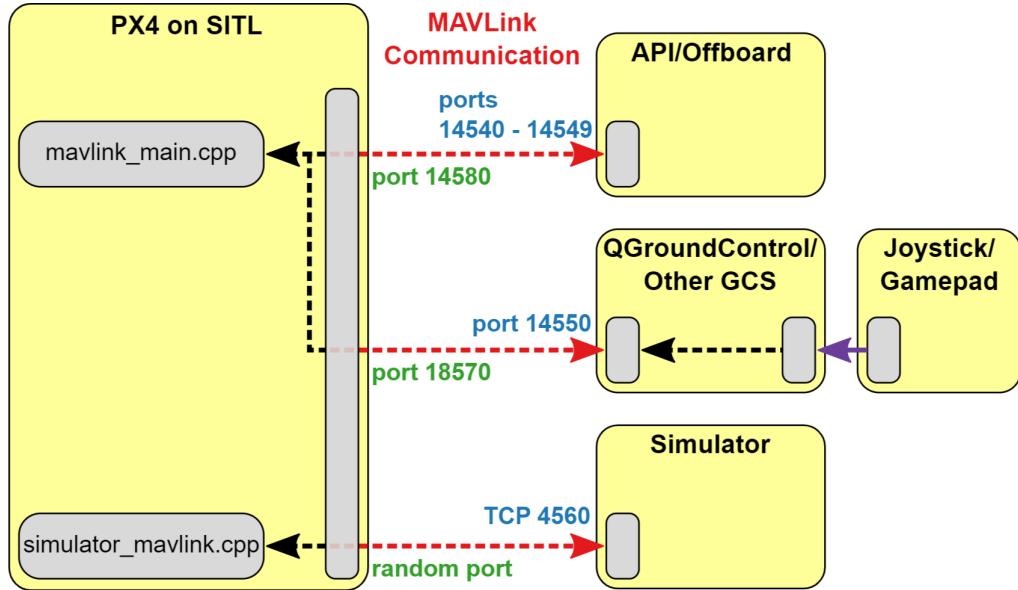
4.1 SITL simulační prostředí

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 4.2 zobrazuje typické prostředí simulace SITL v PX4 pro kterýkoliv z podporovaných simulátorů. Simulátor a PX4 firmware mohou spolu komunikovat pomocí MAVLink API. Pro přímou interakci s PX4 se využívá komunikace 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. 4.2: Diagram komunikace mezi PX4 a simulátorem [24].

4.1.1 Seznam podporovaných simulátorů

Následující simulátory jsou podporovány systémem PX4:

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

4.1.2 Gazebo

Simulátor Gazebo je vysoce doporučen pro PX4 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). [25]

Podporovaná vozidla SITL simulace PX4 v simulátoru Gazebo jsou:

- dron

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

4.1.3 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. [26]

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

Jediné vozidlo, které je podporované SITL simulací PX4 v simulátoru Ignition je dron.

4.2 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 Foxy
- PX4 - ROS 2 *bridge* (komunikační most mezi PX4 firmware a ROS 2)

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

4.2.1 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. [27]

```

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 *NuttX¹/Pixhawk*

4.2.2 Instalace ROS 2 Foxy

Instalace ROS 2 je dobře popsaná na dokumentačních stránkách [30]. 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 nástroje 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. Závislosti Pythonu musí být také nainstalovány:
10 $ sudo pip3 install -U empy pyros-genmsg setuptools
```

4.2.3 Nastavení ROS 2 prostředí

Pro vytvoření ROS 2 pracovního prostoru (*workspace*) podporujícího simulaci v PX4 je nutné stáhnout dva ROS 2 balíčky. [29]

Na stáhnutí a sestavení obou balíčků můžeme použít tyto příkazy:

```
1 # Naklonování obou balíčků:
2 $ mkdir -p ~/px4_ros_com_ros2/src
3 $ git clone https://github.com/PX4/px4_ros_com.git \
4   ~/px4_ros_com_ros2/src/px4_ros_com
5 $ git clone https://github.com/PX4/px4_msgs.git \
6   ~/px4_ros_com_ros2/src/px4_msgs
7
8 # Balíček px4_ros_com obsahuje scripty na sestavení workspace:
9 $ cd ~/px4_ros_com_ros2/src/px4_ros_com/scripts
10 $ source build_ros2_workspace.bash
```

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. [7]

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 mezi ROS 2 a PX4 firmware pomocí Fast RTPS (DDS®). V tomto balíčku je možné vytvářet ROS 2 nároky pro autonomní řízení dronu.

4.2.4 PX4 - ROS 2 bridge

Jak je popsáno v kapitole 3.3 MicroRTPS agent a klient, pro komunikaci mezi ROS 2 node a PX4 slouží *eProsimá 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

Java JDK a Gradle

Java JDK a Gradle jsou nutné závislosti pro správnou funkci Fast-RTPS-Gen. Pro instalaci je možné použít návod zveřejněný zde:

<https://linuxize.com/post/how-to-install-gradle-on-ubuntu-20-04/> [28].

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 následujícího příkazu [31]:

```
1 git clone --recursive https://github.com/eProsimá/Fast-DDS-Gen.git \
2   -b v1.0.4 ~/Fast-RTPS-Gen \
3   && cd ~/Fast-RTPS-Gen \
4   && gradle assemble \
5   && sudo env "PATH=$PATH" gradle install
```

RTPS agent se na palubním počítači spouští pomocí příkazu:

```
1 $ source ~/px4_ros_com_ros2/install/setup.bash
2 $ micrortps_agent -t UDP
```

4.3 ROS 2 misie

Jedna možnost pro vytvoření autonomní mise v systému PX4 je pomocí programu QGroundControl, jak je popsáno v kapitole 2.3.2 Plánování bezpilotní mise. Zde je možné vytvořit jednoduché mise na průzkum prostředí, skenování koridoru, 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 ROS 2 *node* (v C++), který komunikuje s PX4 firmware přes Fast RTPS. Zdrojový kód je zveřejněný na platformě GitHub [32].

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

4.3.1 Offboard flight mode

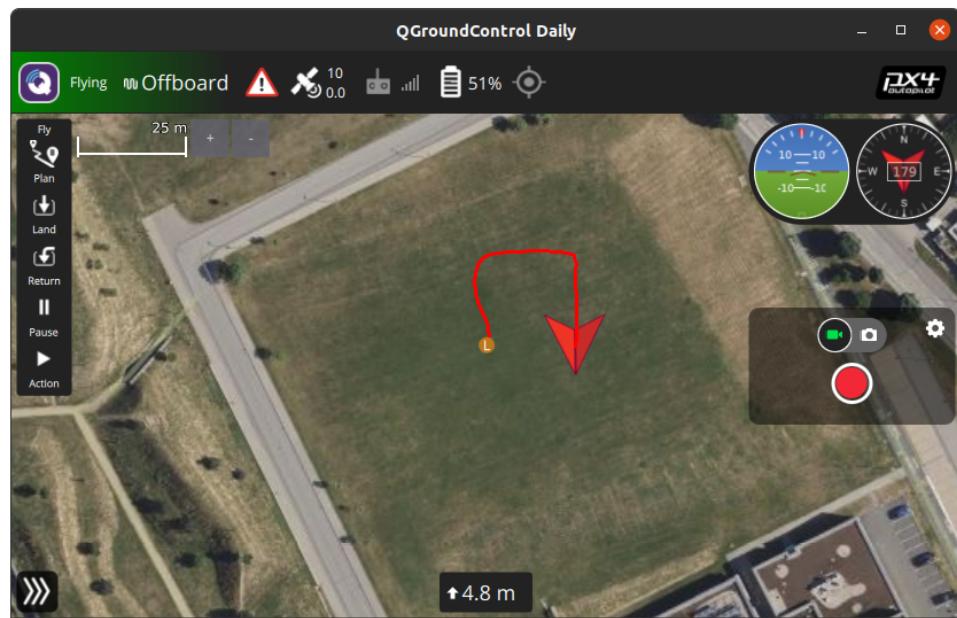
Offboard flight mode se využívá vždy, když je dron řízený z jiného zdroje, než z Pixhawk řídící jednotky (PX4 firmware) například z palubního počítače.

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

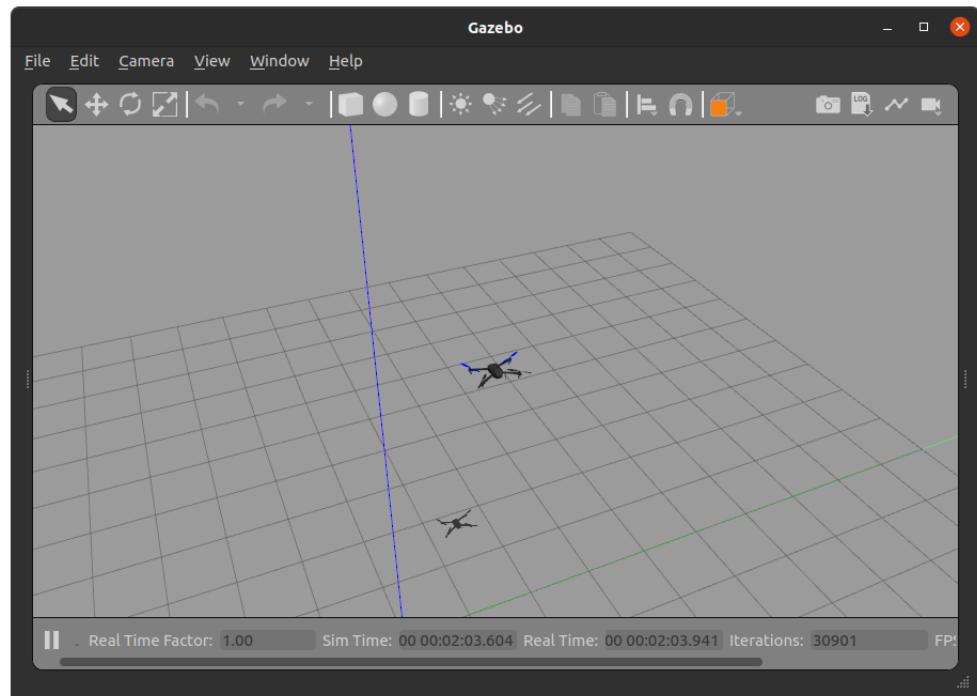
Aby zůstal *offboard* letový režim aktivní, palubní počítač musí posílat zprávu o aktivitě módu (`OffboardControlMode message`) s frekvencí $> 2 \text{ Hz}$. V případě poruchy palubního počítače, nebo nedostatečné rychlosti posílání zpráv se *offboard* letový režim vypne a bude aktivovaný *land* letový režim, takže dron přistane na daném místě. [33]

4.3.2 Výsledky simulace

Pomocí ROS 2 *node* se nám podařilo posílat zprávy do simulovaného PX4 firmware a tím ovládat dron v *offboard* letovém režimu. Na obrázku 4.3 v levém horním rohu je zobrazené, že dron je v režimu *offboard flight mode*. Obrázek 4.4 zobrazuje simulační prostředí Gazebo 11 které poskytuje fyzikální model „světa“ a graficky zobrazuje aktuální stav objektů v robotické misi.



Obr. 4.3: Software QGroundControl s dronem v *offboard flight mode*.



Obr. 4.4: Bezpilotní mise v simulátoru Gazebo.

Závěr

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] Pixhawk 4. *PX4* [online]. San Francisco: Dronecode, 2021 [cit. 2022-01-01]. Dostupné z: https://docs.px4.io/master/en/flight_controller/pixhawk4.html
- [4] Ardupilot, versatile, trusted, open. *Ardupilot* [online]. Ardupilot, c2022 [cit. 2022-01-03]. Dostupné z: <https://ardupilot.org/>
- [5] PX4: Open Source Autopilot For Drone Developers. *PX4* [online]. San Francisco: Dronecode, c2021 [cit. 2022-01-03]. Dostupné z: <https://px4.io/>
- [6] 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>
- [7] PX4 Architectural Overview. *PX4 Autopilot User Guide* [online]. San Francisco: Dronecode, 2021 [cit. 2021-12-19]. Dostupné z: <https://docs.px4.io/master/en/concept/architecture.html>
- [8] PX4 System Architecture. *PX4 Autopilot User Guide* [online]. San Francisco: Dronecode, 2021 [cit. 2021-12-20]. Dostupné z: https://docs.px4.io/master/en/concept/px4_systems_architecture.html
- [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] UORB Messaging. *PX4 Autopilot User Guide* [online]. San Francisco: Dronecode, 2021 [cit. 2021-12-21]. Dostupné z: <https://docs.px4.io/master/en/middleware/uorb.html>

- [13] UORB Message Reference. *PX4 Autopilot User Guide* [online]. San Francisco: Dronecode, 2021 [cit. 2021-12-24]. Dostupné z: https://docs.px4.io/master/en/msg_docs/
- [14] RTPS/DDS Interface: PX4-Fast RTPS(DDS) Bridge. *PX4 Autopilot User Guide* [online]. San Francisco: Dronecode, 2021 [cit. 2021-12-21]. Dostupné z: <https://docs.px4.io/master/en/middleware/micrortps.html>
- [15] EProsimá Fast Buffers. *EProsimá the middleware experts* [online]. Madrid: eProsimá, 2014 [cit. 2021-12-23]. Dostupné z: <https://www.eprosima.com/docs/fast-buffers/0.3.0/html/index.html>
- [16] 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/DDSI-RTPS/2.3>
- [17] Data Distribution Service (DDS). *Object Management Group* [online]. Milford: OMG, 2018 [cit. 2021-11-25]. Dostupné z: <https://www.omg.org/omg-dds-portal/>
- [18] 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/>
- [19] 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/>
- [20] 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
- [21] RTPS Introduction: What is RTPS? *EProsimá: The Middleware experts* [online]. Madrid: eProsimá, c2013-2021 [cit. 2021-11-29]. Dostupné z: <https://www.eprosima.com/index.php/resources-all/whitepapers/rtps>
- [22] 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

- [23] 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>
- [24] Simulation. *PX4 Autopilot User Guide* [online]. San Francisco: Dronecode, 2021 [cit. 2021-12-28]. Dostupné z: <https://docs.px4.io/master/en/simulation/>
- [25] *Gazebo* [online]. California: Open Source Robotics Foundation, c2014 [cit. 2021-12-30]. Dostupné z: <http://gazebosim.org/>
- [26] 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>
- [27] Ubuntu Development Environment. *PX4* [online]. San Francisco: Dronecode, 2021 [cit. 2022-01-02]. Dostupné z: https://docs.px4.io/master/en/dev_setup/dev_env_linux_ubuntu.html
- [28] 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/>
- [29] ROS 2 User Guide (PX4-ROS 2 Bridge). *PX4* [online]. San Francisco: Dronecode, 2021 [cit. 2022-01-02]. Dostupné z: https://docs.px4.io/master/en/ros/ros2_comm.html
- [30] 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>
- [31] Fast DDS Installation. *PX4* [online]. San Francisco: Dronecode, 2021 [cit. 2022-01-02]. Dostupné z: https://docs.px4.io/master/en/dev_setup/fast-dds-installation.html
- [32] PX4_SITL_missions. *GitHub* [online]. California: GitHub, 2021 [cit. 2022-01-03]. Dostupné z: https://github.com/DavidLindtner/PX4_SITL_missions
- [33] Offboard Mode. *PX4* [online]. San Francisco: Dronecode, 2021 [cit. 2022-01-02]. Dostupné z: https://docs.px4.io/master/en/flight_modes/offboard.html

Seznam symbolů a zkrátek

| | |
|-------------------|--------------------------------------|
| IMU | Inertial Measurement Unit |
| 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 |