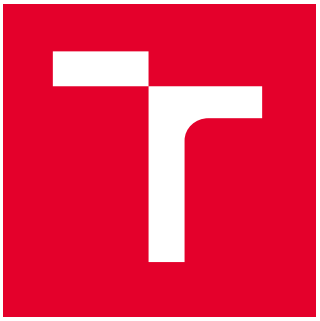


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

VZDÁLENÉ ŘÍZENÍ DRONŮ V REÁLNÉM ČASE

REAL-TIME REMOTE CONTROL OF DRONES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Filip Ferko**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Králík, Ph.D.**

SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Filip Ferko**
Studijní program:	Mechatronika
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Jan Králík, Ph.D.**
Akademický rok:	2024/25

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

Vzdálené řízení dronů v reálném čase

Stručná charakteristika problematiky úkolu:

S rostoucím stupněm automatizace začíná přibývat provozů, které mohou pracovat ve zcela autonomním režimu, ovšem I tyto provozy je nutné nějakým způsobem kontrolovat. V případech kdy statické dohledové systémy nejsou dostatečně vhodným řešením přichází na řadu možnost dálkově ovládaného kontrolního dronu. Cílem práce je vytvořit systém, který podporuje vzdálené řízení dronu na základě komunikačního řetězce klient – server – dron, kde klientské ovládání bude řešeno pomocí webového rozhraní.

Cíle bakalářské práce:

1. Proveďte rešerši problematiky vzdáleného ovládání dronu.
2. Realizujte komunikace dronu se serverem.
3. Realizujte komunikace serveru s klientem.
4. Vytvořte GUI pro ovládání dronu a plánování misí.

Seznam doporučené literatury:

[1] VALÁŠEK, M.: Mechatronika, Vydavatelství ČVUT 1995.

[2] SMITH, Steven W. Digital signal processing: a practical guide for engineers and scientists. . Amsterdam : Newnes, c2003. ISBN 0-7506-7444-X.

[3] MOHANTY, Sachi Nandan, RAVINDRA, J. V. R., SURYA NARAYANA, G., PATTNAIK, Chinmaya Ranjan and SIRAJUDEEN, Yoosuf Mohamed (eds.). Drone technology: future trends and practical applications. . Hoboken, NJ, USA : Wiley, [2023]. ISBN 978-1-394-16653-4.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2024/25

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

Abstrakt

Tato bakalářská práce se zaobírá problematikou vzdáleného ovládání dronů s cílem vytvořit řídicí systém prostřednictvím architektury klient – server – dron. Pomocí webové aplikace bude uživatel moct naplánovat trasu mise, kterou dron má automaticky vykonat. Součástí je zobrazování telemetrie v reálném čase a grafická vizualizace dat z proběhlých letů.

Summary

This bachelor's thesis deals with remote drone control, with the aim of creating a control system using a client-server-drone architecture. By utilizing the web application, users have the capability to design a mission route that the drone will autonomously execute. The application provides real-time telemetry presentation as well as a graphical representation of data acquired from prior flights.

Klíčová slova

dron, server, klient, databáze, webová aplikace, uživatelské rozhraní

Keywords

drone, server, client, database, web application, user interface

Bibliografická Citace

FERKO, F. *Vzdálené řízení dronů v reálném čase*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2025. 50 s., Vedoucí diplomové práce: Ing. Jan Králík, Ph.D..

Prohlašuji, že tato bakalářská práce je původní a je výsledkem mé vlastní činnosti. Veškeré použité informační zdroje jsem řádně citoval a uvedl v seznamu literatury, který je úplný.

Filip Ferko

Brno . . . .

Rád bych poděkoval vedoucímu své závěrečné práce Ing. Janu Králíkovi, Ph.D. za jeho odborné vedení, cenné rady a vstřícný přístup při řešení této práce. Děkuji všem členům studentského spolku Drone Research Center za inspirativní prostředí, technickou podporu a za vytvoření podmínek, díky nimž jsem mohl získat cenné zkušenosti s praktickou prací s drony. Také bych chtěl poděkovat celé své rodině za podporu a trpělivost po dobu celého mého studia.

Filip Ferko

Obsah

1 Úvod

Bezpilotní letadla (UAV) jsou technická zařízení, která v této a minulé dekádě prošla značným rozvojem. Vývoj dronů probíhá zrychlujícím se tempem a je závislý na inovacích ve všech oblastech technických věd. Od návrhu konstrukce, kde je kladen důraz na aerodynamiku a lehkost materiálů, přes výkonovou elektroniku a pohonu, které zajišťují efektivní využití energie, až po sofistikované informační a řídící systémy, které umožňují autonomní řízení a adaptabilitu v různorodých podmínkách. Nejvýznamnějším trendem v této oblasti je pokles cen a velikosti komponent, což mělo za následek zpřístupnění těchto zařízení i pro širší veřejnost a výrobcům umožnilo začlenit sofistikovanější systémy do menších a levnějších zařízení. Rukou v ruce jde s technickým pokrokem i legislativa, která nově řeší autonomní provoz těchto zařízení. V rámci Evropské unie jsou to například U-space prostory.

Důsledkem těchto změn je, že bezpilotní letadla a jiné drony se stávají nedílnou součástí moderní společnosti díky jejich širokému spektru využití napříč různými oblastmi. K nejstarším formám jejich aplikací patří vojenské využití. Tato technologie přetváří způsob, jakým se vedou konflikty, a nabízí nové možnosti pro ochranu strategických objektů. V průmyslu se drony uplatňují při inspekci kritické infrastruktury. Do této kategorie spadají například mosty a elektrické vedení. Zvyšují bezpečnost a efektivitu stavebních projektů a napomáhají jednodušší diagnostikovatelnosti současných staveb. V zemědělství se drony používají k postřiku plodin a monitoringu vegetace. Záchranné týmy využívají drony k vyhledávání pohřešovaných osob či k přenosu obrazu z nebezpečných oblastí, což usnadňuje rozhodování v krizových situacích. V logistice se drony testují pro účely doručování zásilek, s průkopníky jako Amazon či Meituan. Navíc tato zařízení umožňují sběr vědeckých dat z odlehlých a obtížně přístupných míst, což přispívá k pokroku ve výzkumu v různých oblastech.

Dalším trendem je směr k autonomii a zjednodušení vzdáleného ovládání dronů. To je nezbytné pro zvýšení bezpečnosti práce a efektivity samotných operací. Tato potřeba rovněž vede ke snížení počtu operátorů nezbytných k ovládání dronů, což umožňuje jednomu operátorovi řídit více dronů současně. Centralizace řízení prostřednictvím jednoho místa může být uskutečněna přes internet, čímž se zvyšuje kontrola a dostupnost dronů téměř odkudkoliv. Tento koncept se postupně vyvíjí v podobě „Internet of Drones“ (IoD), který je analogií k již zavedenému „Internet of Things“ (IoT). Spojení více dronů prostřednictvím sítě nabízí nové možnosti pro jejich koordinaci, sdílení dat a efektivní správu celé flotily.

Motivací této bakalářské práce je návrh vlastního řídicího systému pro studentský spolek. Tento požadavek vychází z potřeby vytvořit flexibilní a modulární řešení, které by bylo v souladu s nejmodernějšími technologiemi a současným stavem techniky. Takový systém by umožnil rychlejší experimentální vývoj a testování reálných dronů v prostředí s omezenými zdroji.

2 Rešerše

Tato kapitola se věnuje rešerši dostupných řešení a komunikačních prostředků používaných pro ovládání a řízení dronů. Cílem je vytvořit přehled technologií, které lze využít při návrhu a realizaci vlastního řídicího systému. Nejprve jsou popsány softwarové nástroje a ucelené řešení pro ovládání dronů. Následně se kapitola věnuje hardwaru sloužícímu k řízení letu bezpilotních letadel.

2.1 Software pro ovládání a řízení dronů

2.1.1 Komunikační protokoly

Při návrhu komunikačního systému pro bezpilotní prostředky je kladen důraz na nízkou latenci, spolehlivost a efektivní přenos dat. Protokol musí být dostatečně lehký, aby nezatěžoval přenosovou kapacitu spojení, a zároveň robustní vůči výpadkům komunikace.

Na trhu existují proprietární řešení od výrobců komerčních dronů, nicméně v oblasti open-source neexistuje jiný komunikační protokol s takovou mírou rozšíření, podpory a komunitního vývoje, jako má MAVLink. Díky těmto vlastnostem se stal de facto standardem pro komunikaci mezi bezpilotními systémy a jejich řídicími stanicemi. Následující část podrobně popisuje jeho princip a možnosti využití.

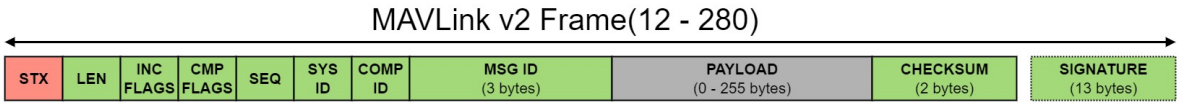
MAVLink

Micro Air Vehicle Link (MAVLink)[1] je komunikační protokol navržený pro bezpilotní systémy, zahrnující bezpilotní letadla (UAV) a další drony. Umožňuje efektivní a standardizovanou obousměrnou komunikaci, a to nejen pro spojení dronu s externím zařízením (vnější komunikace), ale i pro jednotlivé komponenty dronu (vnitřní komunikace). Externím zařízením bývá většinou pozemní stanice (GCS), nicméně MAVLink lze využít i mezi drony při tvorbě a ovládání rojů[2]. Příkladem využití MAVLinku mezi jednotlivými částmi dronu je třeba spojení doprovodného počítače s autopilotem.

Jeho hlavní předností je efektivita a malá velikost zpráv, díky které může protokol fungovat napříč různými komunikačními prostředky jako je USB, WiFi, ethernet či dokonce sub-GHz vysílačky. Další významnou vlastností je jeho podpora napříč různými mikrokontroléry (ARM7, ATMega, dsPic, STM32), operačními systémy (Windows, Linux, MacOS, iOS a Android) a programovacími jazyky (C, JavaScript, Python, Rust, Lua, Ada)[1].

První verzi vydal v roce 2008 Lorenz Meier, jenž je také zakladatelem i dalších open-source projektů pro drony jako například Pixhawk (hardwarová platforma pro vestavěné systémy dronů), PX4 (Autopilot) a QGroundControl (GCS). Od té doby na přelomu let 2015-2017 vznikla druhá a vylepšená verze MAVLink 2.0, která oproti té původní umožňuje definovat více zpráv (16 777 216 oproti 256) a díky možnosti podepisování paketů může být bezpečnější. Zároveň je tato verze zpětně kompatibilní s verzí 1.0, se kterou se v dnešní době setkáme už jen u staršího (legacy) softwaru[3].

MAVLink dosahuje vysoké účinnosti serializací dat do binárního formátu. Struktura zpráv je pevně daná a její sekvence je pro verzi 2.0 popsána na obrázku 2.1 a v tabulce 2.1.



Obrázek 2.1: Struktura packetu pro MAVLink 2.0, převzato z [4]

Tabulka 2.1: Popis jednotlivých částí MAVLink packetu, převzato z [4]

Jméno	Význam	Velikost v bytech
STX	Začátek nového packetu (Start-of-Text), má vždy hodnotu 0xFD.	1
LEN	Velikost payload sekce v bytech	1
INC FLAGS	Nekompatibilní rozšíření, pokud jim přijímač nerozumí, neměl by zprávu zpracovat. Typickým takovým rozšířením je signature.	1
CMP FLAGS	Kompatibilní rozšíření, přijímač může zprávu zpracovat i bez znalosti rozšířeních.	1
SEQ	Pořadové číslo packetu, slouží k detekci ztráty packetu.	1
SYS ID	ID systému, slouží rozlišení různých zařízeních v sítí jako GCS anebo vícero dronů.	1
COMP ID	ID komponent	1
MSG ID	Typ zprávy	3
PAYLOAD	Data zprávy, struktura obsahu závisí na samotném typu zprávy	0–255
CHECKSUM	Kontrolní součet, slouží k ověření, zda při přenosu nedošlo k poškození zprávy.	2

Packet vždy začíná počátečním bytem (STX) a podle obrázku 2.1 sekvence postupuje zleva doprava. Celková velikost jedné zprávy je proměnlivá. Pohybuje se v rozsahu 12 až 280 bytů. Maximální podporovaný počet systémů v síti je 255. Toto číselné omezení platí i pro počet komponent jednotlivého systému.

U části, která klasifikuje typ zprávy (MSG ID), obecně platí, že zprávy s hodnotami menší než 255 jsou zpětně kompatibilní s MAVLink 1.0. Zprávy se podle obsahu dělí do dvou kategorií. Jestliže obsahem je stav systému (např. poloha, kapacita baterie, výška), hovoříme o stavové zprávě. Druhým typem je příkazová zpráva, kterou typicky vysílá GCS k ovládání a řízení dronu[3].

Tabulka 2.2: Vybrané příkazové zprávy, převzato z [3]

Hodnota	Jméno	Význam
21	LAND	Příkaz ke spuštění automatického přistání.
22	TAKEOFF	Příkaz ke vzletu. UAV vzlétne do předem dané výšky. Výška se dá také nastavit jako parametr zprávy (PAYLOAD).
400	ARM/DISARM	Příkaz k rozjezdu motorů nebo jejich zastavení.

Tabulka 2.3: Vybrané stavové zprávy, převzato z [3]

Hodnota	Jméno	Význam
0	HEARTBEAT	Pravidelná zpráva (většinou o frekvenci 1 Hz), kterou každá jednotka posílá pro signalizaci své přítomnosti, typu, režimu a stavu. Pokud systém nevyšle HEARBEAT do 3 sekund považuje se za nepřítomný/vypnutý.
1	SYS_STATUS	Informuje o systémovém stavu, např. o napětí baterie, senzorech a systémových chybách.
20-23	PARAM	Slouží k výměně a úpravě parametrů mezi systémy (např. načtení/nastavení parametrů řízení letu).
37-47, 51	MISSION	Slouží k nahrávání, čtení a správě misí.

Správnost dat zprávy se ověřuje pomocí modifikovaného algoritmu cyklického redundantního součtu pro 16bitové číslo (CRC-16)[5]. Odesilatel vypočte z dat zprávy číslo, které pošle příjemci pod kontrolním součtem (CHECKSUM). Příjemce stejným algoritmem zkontroluje přijatou zprávu. Pokud se kontrolní součet shoduje, považuje se zpráva za platnou.

Pro zlepšení bezpečnosti komunikace se volitelně může přidat podpis zprávy. Ten zajišťuje, že zpráva pochází od důvěryhodného zdroje a také nebyla během přenosu upravena. To slouží například k zabránění spoofingu[3]. Struktura podpisu je popsána v tabulce 2.4.

Tabulka 2.4: Popis jednotlivých částí podpisu MAVlink zprávy, převzato z [6]

Jméno	Význam	Velikost v bytech
LINK ID	ID spojení – jakým komunikačním prostředkem přišla zpráva (USB, WiFi)	1
TM STAMP	Timestamp – čas uběhlý v 10 mikrosekundách od 1.1.2015 GMT	6
SIGNATURE	Bezpečnostní podpis – závisí na čase, samotné zprávě a tajného klíče	6

Zde je nutné podotknout, že každá následující zpráva musí mít větší hodnotu časového razítka (TM STAMP). Tajný klíč musí být zabezpečen v trvalém úložišti a za žádných okolností nesmí být odhalen prostřednictvím veřejně dostupných komunikačních protokolů. Je zvlášť důležité, aby klíč nebyl zahrnut v parametrech MAVLink, ani v logovacích souborech MAVLink nebo dataflash, které by mohly být zpracovány pro analýzu veřejných protokolů[6].

Přestože druhá verze komunikačního protokolu MAVLink obsahuje mechanismus pro podepisování paketů, stále vykazuje vážné bezpečnostní nedostatky. Tyto nedostatky se nejvíce projevují v oblasti důvěrnosti, tedy ochrany obsahu zpráv před neautorizovaným přístupem nebo odposlechem ze strany třetích stran[3].

Konkrétně to znamená, že pokud se útočníkovi podaří připojit na síť, ve které právě probíhá komunikace pomocí MAVLink, může velmi snadno odposlechnout a přečíst obsah

přenášených zpráv. Protokol totiž sice umožňuje ověřit pravost odesílatele (díky podpisům), ale sám o sobě nezajišťuje šifrování dat. To představuje závažný problém zejména v případech, kdy se přenášejí citlivé informace.

Jako reakce na tento nedostatek byl vyvinut rozšířený protokol s názvem **MAVSec**, který přidává šifrování zpráv k původnímu MAVLink protokolu[7]. Tento přístup výrazně zvyšuje bezpečnost komunikace a chrání před útoky typu man-in-the-middle nebo jednoduchým pasivním odposlechem[3].

Alternativní a ještě bezpečnější možnost představuje použití **standardizační dohody NATO pro rozhraní systému řízení bezpilotních letadel a pro interoperabilitu bezpilotních letadel (STANAG 4586)**[8]. Tento standard definuje rámec pro interoperabilní komunikaci mezi různými systémy bezpilotních prostředků a jejich pozemními řídicími stanicemi. I když byl původně navržen pro vojenské účely, je možné jej využít i u běžně dostupných komerčních modelů[9].

V takovém případě se STANAG 4586 může používat pro komunikaci mezi GCS a doprovodným počítačem. Následně se zprávy převádějí do MAVLink formátu, který slouží ke komunikaci s autopilotem. Jelikož tento úsek komunikace obvykle probíhá v rámci jednoho systému (např. uvnitř dronu přes sériový port), riziko neoprávněného odposlechu nebo zásahu je zde minimální[9].

MAVSDK

Micro Air Vehicle Software Development Kit (MAVSDK) je sada knihoven pro různé programovací jazyky (např. C++, Python, Swift)[10]. Cílem těchto knihoven je usnadnit vývoj, testování a nasazení systémů využívajících MAVLink protokol. Její přední vlastností je abstrakce nad strukturou protokolu. Vývojáři díky tomu mohou využívat jednodušší API volání a většinou nepotřebují detailní znalost komunikačního protokolu, jako je například struktura, typ zpráv a jejich hodnota. Je zejména určena pro vývoj aplikací pro UAV. Současné verze MAVSDK podporují pouze nejnovější verzi MAVLinku[11].

Samotné jádro knihovny je psané v programovacím jazyku C++ a pro ostatní jazyky se používá wrapper[11]. Wrapper definuje rozhraní v druhém programovacím jazyku a volá podkladové funkce z jádra. Díky tomu má vývojář dojem, že pracuje čistě v druhém programovacím jazyku. Struktura SDK je tedy napříč různými programovacími jazyky přibližně stejná, nicméně může mít omezenou funkčnost. Konkrétním příkladem omezení je v současné době například absence třídy *MavlinkPassthrough* v Pythonu, která umožňuje s MAVLinkem přímo pracovat na nízké úrovni. To má za následek, že uživatel může s dronem dělat pouze to, co umí MAVSDK pro Python (dále jen pyMAVSDK)[12].

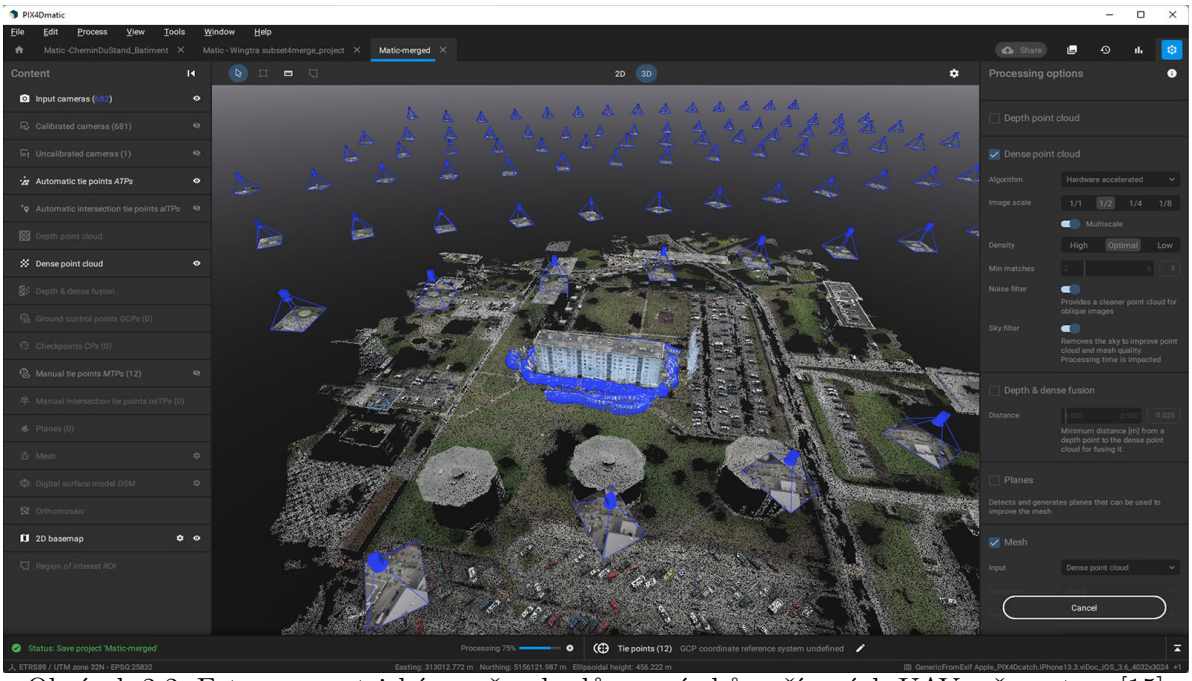
Architektura SDK je asynchronní, umožňuje spouštět a vykonávat více operací najednou, aniž by se navzájem blokovaly. V praxi to například znamená, že můžeme číst telemetrii z různých komponentů dronu a zároveň posílat příkazové zprávy, aniž bychom museli čekat na hodnoty telemetrie. To je obzvlášť výhodné, pokud SDK běží na serveru, který je z podstaty věci také asynchronní. V Pythonu bývá asynchronní kód doprovázen klíčovými slovy *async* a *await*. Je nutné zde také podotknout, že přítomnost asynchronního kódu neznamená skutečný paralelní běh, ale spíš že program umí efektivně čekat na pomalé operace, aniž by se na nich zasekl. Jestliže by součástí asynchronního kódu byla rychlá smyčka, může se stát, že neumožní jiným asynchronním funkcím dostat se ke slovu, což může vést k zamrznutí programu. Běžným řešením je přidat do smyčky *await asyncio.sleep()*.

Vložený kód 2.1: Ukázkový skript pyMAVSDK, převzato a upraveno z [13]

```
import asyncio
from mavsdk import System

async def run():
    drone = System()
    await drone.connect(system_address="udp://:14540")

    print("Waiting for drone to connect...")
    async for state in drone.core.connection_state():
        if state.is_connected:
            print("-- Connected to drone!")
            break

    print("-- Arming")
    await drone.action.arm()

    print("-- Taking off")
    await drone.action.takeoff()

if __name__ == "__main__":
    asyncio.run(run())
```

2.1.2 Software pro pozemní řídící stanice

Moderní aplikace běžící na GCS (Ground Control Station) přináší uživatelům řadu možností a funkcí, jak pracovat s dronem, popřípadě s rojem dronů. Jednou z nejvýznamnějších schopností je plánování a management misí, které umožňují uživatelům řídit a organizovat mise efektivně. Návrh trajektorií a jejich bodů (waypointů) nemusí být zadáván uživatelem ručně. Většina aplikací již v dnešní době podporuje automatické generování tras. To se například často používá pro fotogrammetrii anebo při průzkumných misích, kde je potřeba rychle naplánovat misi a precizně ji odletět[14]. Neméně důležitou schopností je monitoring stavu dronů a misí a přehledné zobrazení telemetrie v reálném čase. Vizualizace letu často probíhá prostřednictvím map, grafů a video streamingu. Další důležitou vlastností je diagnostika, která umožňuje sledování stavu baterie, motorů, senzorů a dalších parametrů dronu – s upozorněním při detekci problému. Analýza dat a reporty jsou klíčové pro vyhodnocení úspěšnosti misí a plánování budoucích operací. Export dat a vizualizace historických misí pomáhají při detailním zkoumání minulých letů.



Obrázek 2.2: Fotogrammetrická mračna bodů ze snímků pořízených UAV, převzato z [15]

Software pro pozemní stanice můžeme rozdělit do dvou kategorií, a to na základě rozdílů v architektuře, dostupnosti, výkonu a způsobu komunikace.

Native (lokální) software je provozován přímo na zařízení pozemní stanice, kterým může být notebook, tablet, mobilní telefon nebo jiné specializované zařízení určené ke kontrole dronů. Komunikace s dronem je realizována přímo prostřednictvím sériového portu, USB, radiových vln nebo přímým UDP/TCP spojením. Přímé spojení je výhodné především díky nízké latenci, avšak nevýhodou je nutnost instalace a správa aktualizací na každém jednotlivém zařízení. Mezi typické příklady tohoto softwaru patří Mission Planner[16] a QGroundControl[17]. Obě aplikace jsou dostupné pod open-source licencí.

Obrázek 2.3: Automatické plánování trasy v programu Mission Planner

Cloudový software je provozován na vzdáleném serveru a je zpřístupněn uživatelům prostřednictvím webového rozhraní nebo API. Tato cloudová řešení využívají spojení přes WiFi nebo LTE sítě a komunikují pomocí protokolů jako HTTP, UDP nebo TCP[18]. Obecně se architektura takového softwaru skládá z následujících částí:

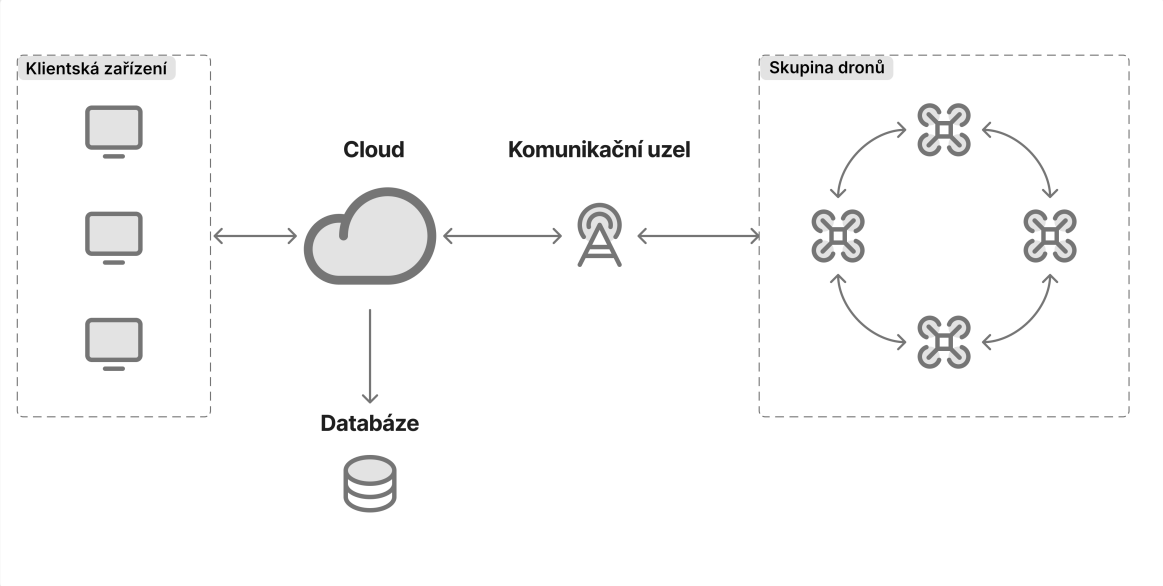
- **Klientská zařízení** reprezentují uživatelské rozhraní webové aplikace. Uživatelé (může jich být zároveň několik) mohou z těchto zařízení posílat příkazy, plánovat mise nebo sledovat stav dronů. Komunikace probíhá přes cloud.

- **Cloud (centrální server)**, zde má úlohu zprostředkovatele komunikace mezi uživatelskými zařízeními a samotnými drony. Cloud může zahrnovat rozličné prvky, jako je backendová část realizovaná pomocí serveru, autentizační mechanismy, či logickou strukturu upravenou pro zpracování a vyřizování požadavků. Z tohoto místa se rovněž uskutečňuje přístup k databázovému úložišti.

- **Databáze** slouží k ukládání dat. Může zahrnovat údaje o dronech, včetně jejich unikátních identifikačních čísel, aktuálního stavu a pozice, ale také záznamy o misích, telemetrická data a data týkající se uživatelských oprávnění. Databázový systém může běžet tandemově s cloudovou částí na stejném serveru nebo fungovat jako nezávislý celek.

- **Komunikační uzel** zajišťuje spolehlivý přenos dat mezi cloudovou strukturou a drony. Může k tomu využívat různé technologie jako jsou LTE sítě či Wi-Fi spojení a současnou telekomunikační infrastrukturu. Fyzicky bývá tato část umístěna v blízkosti dronů, aby se zajistilo rychlé a stabilní spojení.

- **Skupina dronů** interaguje s cloudovou infrastrukturou, ale zároveň může komunikovat mezi sebou.



Obrázek 2.4: Schéma cloudové architektury pro správu dronů

V kontextu obranného průmyslu se také setkáme s decentralizovanou architekturou, která umožňuje každému uzlu částečnou autonomii, čímž je zaručena funkčnost jednotlivých dronů i při výpadku spojení s centrálním serverem[19].

Výhodou cloudového řešení je jeho praktičnost, která umožňuje přístup odkudkoliv, za předpokladu, že všechny komponenty mají internetové připojení. Díky tomu fyzická vzdálenost mezi dronem a uživatelem přestává být relevantní. Navíc lze systém snadno škálovat pro větší počet dronů.

Ovšem s touto technologií přichází i nevýhody, jako je vyšší komplexita systému, zvýšená latence a složitost bezpečnostních opatření. Nezbytné je zavedení autentizace uživatelů, šifrování dat pomocí například technologií AES-256 a nastavení firewallu pro ochranu před neautorizovaným přístupem[20].

2.2 Hardwarové aspekty bezpilotních systémů

2.2.1 Letové ovladače

Letové ovladače hrají klíčovou roli při stabilizaci dronů a při řízení letu. Skládají se z vestavěného systému, který je postaven na mikrokontroléru (MCU). Na rozdíl od mikroprocesoru (MPU) integruje tento mikrokontrolér nejen paměťové komponenty, jako je RAM a Flash paměť, ale také různé periferie (GPIO, UART). Běžně se pro drony používají mikrokontroléry řady ARM Cortex[21].

Letové ovladače obsahují vestavěné senzory, které jsou nezbytné pro regulaci letu. Ve většině případů je v jednom letovém ovladači integrováno několik stejných senzorů, typicky ve dvojicích nebo trojicích, aby se zajistila spolehlivost v případě selhání některého z nich. Zpravidla se jedná o následující tři senzory:

- **Inerciální měřící jednotka (IMU)** se v aplikacích pro UAV skládá ze tříosého gyroskopu a tříosého akcelerometru. Měří lineární zrychlení, úhel natočení a úhlovou rychlost. Využívá se při odhadu trajektorie (tzv. sensor fusion)[22].

- **Barometr** se používá k odhadu výšky od hladiny vzletu (relativní výška). Měří tlak vzduchu, který klesá se vzrůstající výškou. Na delších časových intervalech může barometr ukazovat jiné hodnoty pro stejnou konstantní výšku, a tak se pro korekci používají i data z GPS[22].

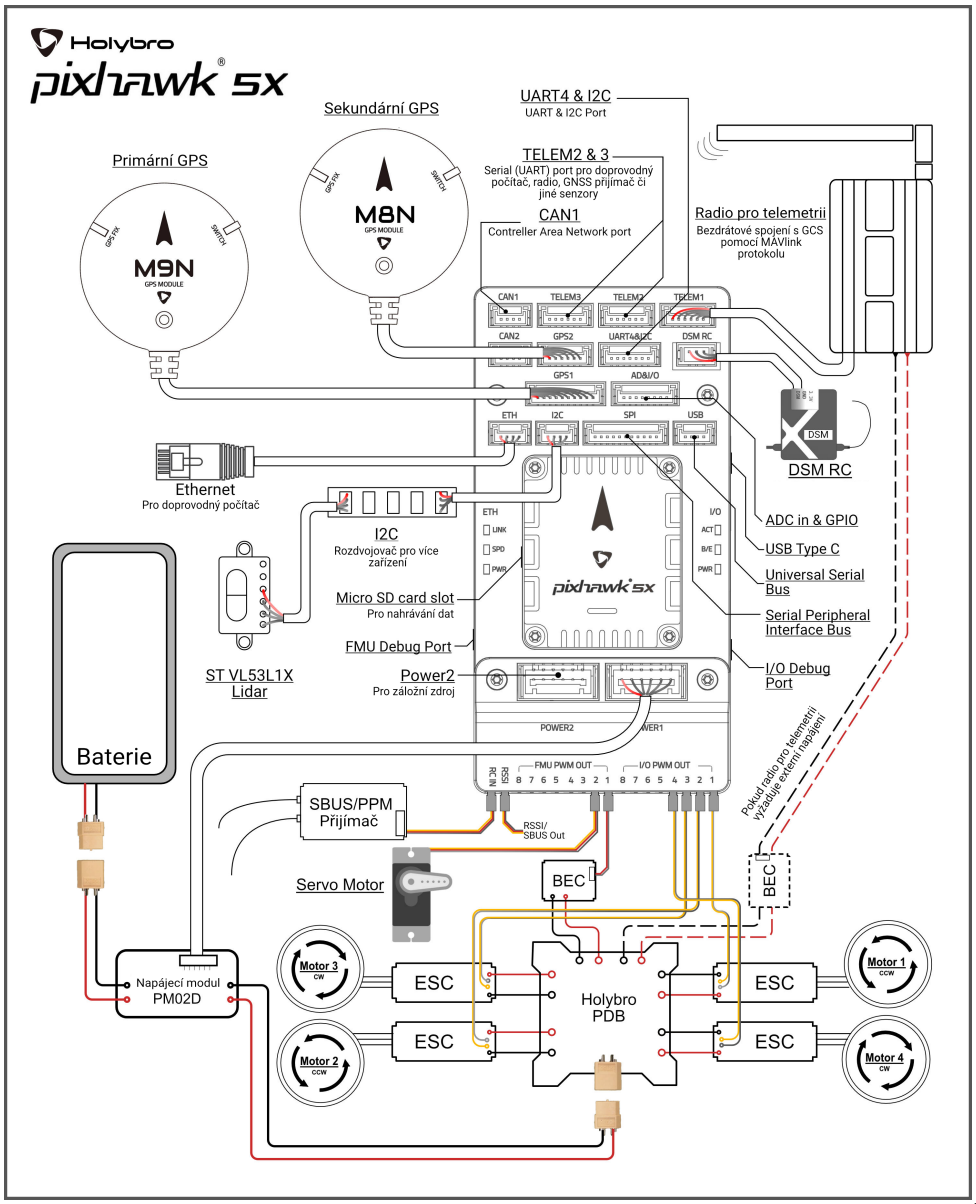
- **Magnetometr** určuje absolutní směr vůči Zemi (azimut). Výrazně ulehčuje navigaci pomocí GPS bodů. Může být i součástí IMU[22].

I tak musí mít letové ovladače porty pro další externí senzory. Pro většinu autonomních dronů je nezbytným externím senzorem **GNSS (Global Navigation Satellite System)** přijímač, pomocí kterého jsme schopni zjistit přesnou polohu dronu. Dalšími externími senzory mohou být **LiDAR** nebo **sonar** pro detekci blízkých překážek[22].

Mimo externí senzory se k ovladači připojuje napájecí deska, přes kterou se posílá PWM signál do motorů. Zásadní je také spojení rádiového modulu pro zajištění komunikace s pozemní stanicí či využití sériového portu pro přenos dat s doprovodným počítačem. Pro větší drony je nezbytné zahrnout Remote ID, které slouží k vysílání identifikačních údajů. Remote ID je u dronů ekvivalencí k registrační značce u automobilů. Pro komunikaci s externími komponenty se často využívají sběrnice I2C, UART, USART nebo SPI[23].

Aby letové ovladače mohly správně fungovat a řídit dron na nízké úrovni, potřebují k tomu **autopilot firmware**. Typicky takový firmware běží na operačním systému reálného času (RTOS), který zaručuje, že úlohy běží v přesně stanovených časech a jsou dokončeny do určitého časového intervalu. Jakákoliv neočekávaná latence by mohla zapříčinit nestabilitu systému a pád dronu.

Mezi nejrozšířenějšími platformami pro firmware autopilotů patří především **Ardupilot** a **PX4**. Obě zmíněné platformy podporují různé typy a konfigurace dronů jako například quadkoptéry, VTOL letadla a pozemní drony. Také obsahují failsafe mechanismy pro případ neočekávané poruchy nebo jiného mimořádného stavu, jako je nízká baterie či ztráta spojení. Díky nim může dron bezpečně přistát nebo pokračovat v bezpečném režimu, čímž se předchází poškození dronu nebo ohrožení osob v okolí. Podpora protokolu MAVLink je u obou platforem standardem.



Obrázek 2.5: Schéma zapojení pro ovladač Pixhawk 5X, převzato a upraveno z [24]

2.2.2 Doprovodné počítače

Letové ovladače se starají o řízení na nízké úrovni. To ale často nestačí u autonomních dronů, kde je nutné vyhodnotit situace bez zásahu uživatele nebo programů běžící mimo dron. To vyžaduje značnou výpočetní sílu, kterou jsme schopni získat pomocí doprovodného počítače.

Doprovodné počítače fungují na běžných desktopových operačních systémech jako například Linux a úlohy, které řeší, nepotřebují být nutně zpracovány v rámci několika milisekund. Používají se převážně pro následující účely:

- **Zpracování obrazu** pro detekci objektů a jejich klasifikaci. Také se používá pro metodu SLAM (Simultaneous Localization and Mapping), díky kterému dron dokáže zjišťovat svou polohu v neznámém prostředí a zároveň toto prostředí mapovat.[25]
- **Autonomní navigace**, pomocí které se dron například může bez zásahu operátora vyhýbat nečekaným překážkám. Využívá se také v situacích, kdy předpokládáme slabé spojení či jeho výpadek s uživatelem.[26]
- **Zpracování dat z externích senzorů** může probíhat již na dronu. Tok dat ze senzorů může být větší, než je kapacita sítě mezi uživatelem a dronem. Proto je nutné data na dronu přímo zpracovat a uživateli posílat jenom výsledky měření, které mají menší velikost.[26]
- **Integraci robotických systémů** lze provést například pomocí ROS (Robot Operating System). Díky doprovodnému počítači můžeme například ovládat dron i manipulátor, který je k dronu připojen[27].
- **Správa komunikace** mezi letovým ovladačem a pozemní stanicí, cloudem či jinými drony. Můžeme využívat kompresi dat, šifrování nebo vlastní komunikační protokoly.[28]

Obrázek 2.6: Dron s doprovodným počítačem typu Intel NUC, převzato z [29]

Z hlediska hardwaru se často používají následující typy počítačů, kde jejich společnou charakteristikou je kompaktní velikost a nízká spotřeba energie.

Prvním z nich je **Raspberry Pi**, který se vyznačuje především svou nízkou cenou a miniaturními rozměry[30]. Díky široké podpoře od komunity je oblíbený pro různé projekty a aplikace, ačkoli jeho výkon je omezený, což může představovat překážku pro náročnější aplikace.

Další populární variantou je **NVIDIA Jetson**, což je vestavěná deska vybavená výkonným grafickým procesorem (GPU)[30]. Tato zařízení mají mnoho výpočetních jader, což je činí vhodnými pro úlohy v oblasti zpracování obrazu a umělé inteligence.

Poslední zmíněnou kategorií je **Intel NUC**, který nabízí vyšší výpočetní výkon díky své univerzální x86 architektuře[30]. Podporuje různé operační systémy, jako jsou Linux a Windows.

3 Formulace cílů

Cílem této bakalářské práce je realizovat řídící systém pro komunikaci mezi uživatelem a dronem pomocí řetězce klient – server – dron, přičemž server v rámci této práce bude umístěn na pozemní stanici (tzv. "on premise"). Díky tomuto umístění se práce soustředí čistě na funkční návrh a realizaci komunikačních modulů, aniž by bylo nutné řešit dodatečné náklady a komplikace spojené s cloudovými službami (hostování, autentizace, šifrování apod.).

Nedílnou součástí systému bude grafické uživatelské rozhraní (GUI) pro správu a plánování misí, přehled proběhlých letů a nastavení spojení s dronem.

Systém musí být kompatibilní s platformou ArduPilot a bude testován na kvadrokoptéře s doprovodným počítačem Intel NUC.

Pro naplnění výše uvedeného cíle práce byly definovány následující konkrétní úkoly:

1. Navrhnout a implementovat modul pro komunikaci dronu se serverem.
    - Jelikož doprovodný počítač nebude připojený k RC vysílači, komunikace mezi pozemní stanicí bude probíhat prostřednictvím WiFi sítě. Doprovodný počítač bude spravovat komunikaci se zemí a zároveň zpracovávat data z letového ovladače. Dále bude sloužit k řízení na vyšší úrovni, s budoucím potenciálem rozšíření na autonomní navigaci (viz kapitola 2.2.2).
2. Navrhnout a implementovat modul pro komunikaci serveru s klientskou aplikací.
    - V souladu s běžnou architekturou klient–server (viz obrázek 2.4) bude nutné mezi těmito celky vytvořit aplikační rozhraní, přes kterou bude probíhat komunikace. Součástí bude databáze propojená se serverem, kde se budou ukládat trasy misí a uplynulé lety.
3. Vytvořit grafické uživatelské rozhraní pro ovládání dronu a plánování misí.
    - Rozhraní by mělo obsahovat interaktivní mapu s polohou dronu a vizualizací plánované trasy, podobně jako v programu Mission Planner (viz obrázek 2.3). Uživatel bude moci trasu zadávat ručně, přičemž by měla být k dispozici i interaktivní tabulka s body trasy propojená s mapou. Pro implementaci těchto funkcí bude vhodné využít moderní frontendový framework.

4 Realizace

Tato kapitola popisuje návrh systému pro ovládání UAV. Na začátku je představena celková architektura řešení. Dále jsou detailně popsány použité technologie a nástroje, které byly zvoleny s ohledem na požadavky, mezi které patří rychlost vývoje, modularita a snadná rozšiřitelnost systému.

4.1 Návrh architektury systému

Systém používaný k řízení bezpilotního letounu se skládá ze dvou hlavních součástí, konkrétně z pozemní stanice a samotného dronu. Pro zajištění efektivní a spolehlivé komunikace se využívají různé protokoly.

Hypertext Transfer Protocol (HTTP) je základní protokol pro výměnu informací na webu, přičemž komunikace probíhá formou požadavku a odpovědi. Požadavek v HTTP zahrnuje vždy metodu jako je GET, POST, PUT nebo DELETE a URL a může obsahovat tělo zprávy. Odezva serveru pak přichází s příslušnými status kódy, které indikují úspěšnost či neúspěšnost zpracování požadavku a odpovědí.

Websocket (WSS) umožňuje obousměrnou (duplex) komunikaci v reálném čase. Tento pevně navázaný kanál přenáší data bez nutnosti předchozích žádostí od klienta. Díky tomu nemusíme posílat dronu několikrát za sekundu požadavky o telemetrii a zároveň se nám sníží latence spojení.

Jelikož zde posílá živá data jenom dron, bylo by možné použít i jednostranný přenos pomocí **Server-Sent Events (SSE)**. Toto řešení by bylo ale nevhodné, pokud bych například později chtěl realizovat ovládání pomocí joysticku.

MAVLink je protokol, který se využívá pro komunikaci mezi doprovodným počítačem a letovým ovladačem. Komunikace je zajištěna kabelovým spojením přes sériový port. Podrobnější informace o MAVLink protokolu lze najít v kapitole 2.1.1.

Pro komunikaci s databází se také používá PostgreSQL wire protokol, který je založený na rodině protokolů **TCP/IP**.

Pozemní stanice může mít několik různých provedení. Může nabývat formy realistického kokpitu v samostatné místnosti, nebo se může jednat o robustní vojenské kufry vybavené více displeji a integrovanými periferními zařízeními, jako jsou například joysticky, v kombinaci s pokročilými komunikačními technologiemi[31]. Pro tento účel však postačí použití notebooku, který poskytuje spojení přes Wi-Fi hotspot, přičemž může být provozován na operačních systémech Windows nebo Linux. Na pozemní stanici běží tři části systému.

Frontend představuje uživatelské rozhraní systému, se kterou operátor přímo pracuje. Je zodpovědný za grafické zpracování dat z dronu a zároveň reaguje na uživatelské vstupy, které poté validuje a transformuje do formátu srozumitelného pro backend. Uživateli se tato část zobrazuje jako běžná webová stránka. Tato část je spravována serverem, který se nachází na portu `:5173`.

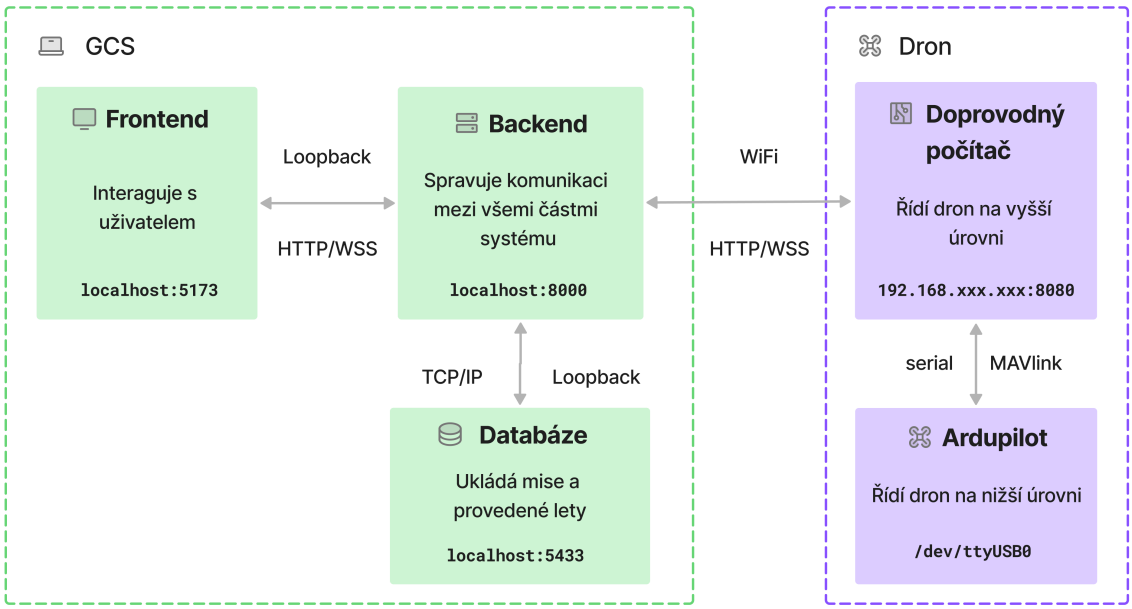
Backend funguje jako prostředník mezi dronem, databází a frontendem. Funguje

na bázi REST API a umožňuje operace typu CRUD (Create, Read, Update, Delete), které slouží ke správě dat v databázi. Tato komunikace probíhá přes HTTP protokol. Backend zároveň umožňuje navázat WSS spojení mezi frontendem a dronem.

Databáze je klíčovým prvkem pro ukládání dat, jako jsou mise, waypointy, zaznamenané lety a naměřená telemetrie během letu. Poskytuje úložiště pro dlouhodobé uchování a snadný přístup k těmto informacím, což je nezbytné pro analýzu a plánování operací dronu.

Na samotném dronu se nachází dvě části systému. Prvním z nich je **doprovodný počítač**. Jeho hlavním úkolem je spravovat spojení s pozemní stanicí a překládat HTTP požadavky do formátu srozumitelného pro letový ovladač. Kromě toho se zde nachází také logika pro ukládání dat, která jsou formou požadavků zasílána do backendu. Nachází se zde ovládání dronu na vyšší úrovni (např. příkazy "vzlet", "přistání"). Jelikož se nachází na stejné síti jako GCS, jeho IP adresa je ve tvaru `192.168.x.x`.

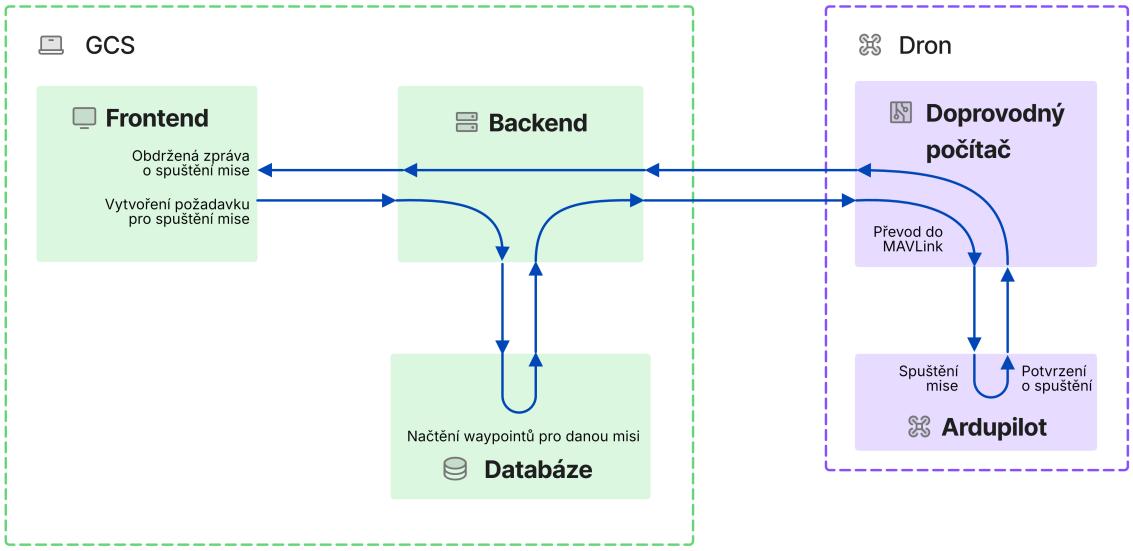
Druhou částí je letový ovladač, který se stará o řízení na nižší úrovni (např. regulace letu, provedení úkonů nutných pro úspěšný vzlet). Tento systém běží na platformách Ardupilot a Pixhawk, které poskytují hardware a software pro základní řízení letu.



Obrázek 4.1: Grafické znázornění architektury systému

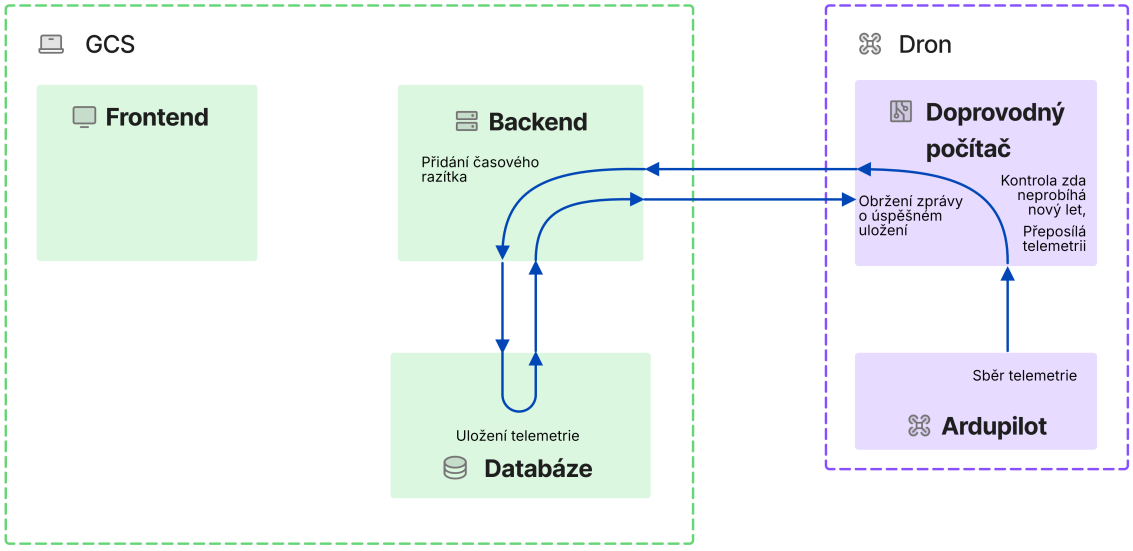
Pomocí obrázku 4.1 lze odvodit, jak pro takový systém fungují běžné operace jako je posílání misí do dronu, ukládání bodů tras či telemetrie.

Pro spuštění mise uživatel zmáčknutím tlačítka vyšle z frontendu (`localhost:5173`) požadavek. Backend (`localhost:8000`) tuto zprávu zpracuje a z databáze (`localhost:5433`) obdrží všechny waypointy pro žádanou misi. Pokud taková mise neexistuje, backend vyšle zprávu zpět uživateli s HTTP kódem 404 (mise nenalezena). V případě úspěchu požadovaná data přepošle doprovodnému počítači (`192.168.xxx.xx:8080`). Zde se verifikuje formát mise a převede se do MAVLink formátu. Poté doprovodný počítač čeká na odpověď z letového ovladače. Konečná zpráva o spuštění mise následně putuje z doprovodného počítače zpět k uživateli přes backend.



Obrázek 4.2: Tok dat v systému při spouštění mise

Během letu doprovodný počítač shromažďuje data z letového ovladače (např. informace o poloze, výšce, rychlosti) prostřednictvím protokolu MAVlink. Tato data jsou odesílána v pravidelném intervalu zpět na backend. Backend je následně ukládá do databáze, která slouží jako centrální úložiště historických dat a misí. Doprovodný počítač obdrží zpětně zprávu o úspěšném uložení (HTTP kód 200 – OK).



Obrázek 4.3: Tok dat při ukládání telemetrie letu

Podobně lze popsat i situace, kdy uživatel vytváří, upravuje či maže data z databáze (smyčka frontend – backend – databáze), nebo kdy se posílají živá data uživateli (jednosměrný tok Ardupilot – doprovodný počítač – backend – frontend).

4.2 Použité technologie

4.2.1 Webový klient

Před samotnou tvorbou frontendu je důležité si definovat, jaké vlastnosti má webový klient splňovat a k tomu vybrat vhodné technologie. Jednotlivé komponenty by měly reagovat na data přijímaná v reálném čase z dronu a také interagovat s uživatelem. Aby tvorba misí a ovládání dronů byla intuitivní, součástí grafického rozhraní by měly být pokročilé interaktivní prvky, jako jsou dialogová okna, kombinovaná pole a datové tabulky. Přechody mezi jednotlivými stránkami by měly být plynulé, a proto by měl být klient navržen jako **jednostránková webová aplikace (SPA)**.

Jednostránková aplikace interaktivně načítá obsah na jednu stránku a dynamicky aktualizuje (hydratuje) jednotlivé části uživatelského rozhraní, aniž by znovu načítala a překreslila celou stránku. Oproti tomu běžná webová stránka znovu načítá veškerý obsah při jakékoliv změně rozhraní, což může způsobit pomalejší odezvu.

Proto pro vytvoření takové aplikace nestačí běžný technologický balíček Hypertext Markup Language (HTML), kaskádové styly (CSS) a JavaScript (JS) a je potřeba využít **framework**, který nám umožní rychlejší vývoj.

Framework je sada vývojářských nástrojů a knihoven, které usnadňují vývoj tím, že poskytují základní kostru, na které vývojáři staví své aplikace. Tato kostra řeší za vývojáře opakující se problémy spojené s návrhem aplikace.

Umožňují lépe strukturovat projekt díky rozdělení aplikace na znovupoužitelné části, známé jako komponenty. Tyto komponenty mohou zahrnovat prvky, jako jsou tlačítka, seznamy nebo modální okna. Zároveň umožňují dělit kód na základě toho, co každá část dělá.

Současné webové frameworky jsou založeny na principu reaktivního programování a umožňují automatickou aktualizaci Document Object Modelu (DOM – struktura reprezentující HTML dokument) při změně proměnných. Například se automaticky aktualizuje mapa v okamžiku přijetí nových telemetrických údajů, což zajistí, že zobrazené informace jsou vždy aktuální a přesné.

Vložený kód 4.1: Nereaktivní kód v JavaScript

```
let b = 1;

let c = 2;

let a;


a = b + c;

b = 10;


console.log(a); // Výstup: 3
```

Vložený kód 4.2: Reaktivní kód ve Svelte

```
let b = 1;

let c = 2;

let a;


$: a = b + c; // Reaktivní výraz

b = 10;


console.log(a); // Výstup: 12
```

Pro tento projekt jsem zvolil **Svelte 4**, který běží na **Node JS** serveru. Svelte patří mezi novější frameworky a je charakteristický tím, že postrádá běhové prostředí (runtime engine)[32]. Místo toho kompiluje kód přímo do nativního JavaScriptu[32]. To vede

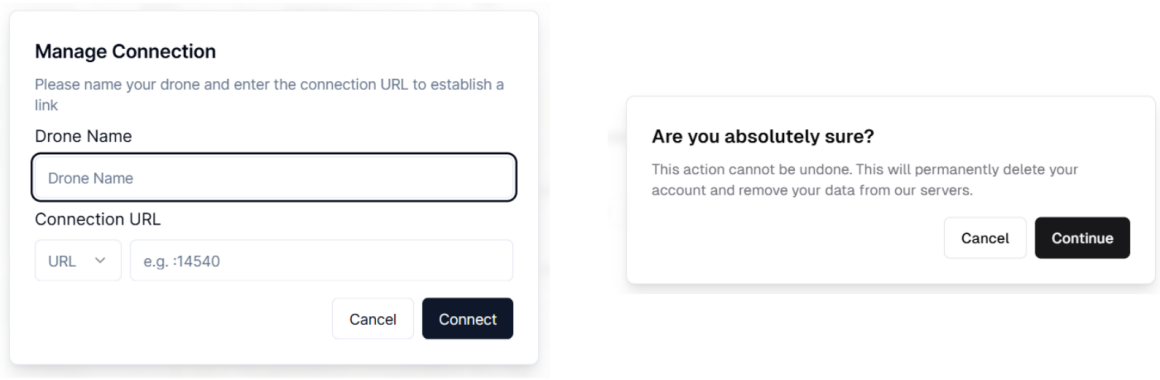
k rychlejšímu běhu a menší velikosti výsledné aplikace; konkrétně je velikost základního balíčku přibližně 25krát menší ve srovnání s React[33]. Svelte má také srozumitelnější syntaxi, a je proto i snazší na naučení. Během psaní této práce byla vydána nová verze, Svelte 5, avšak vzhledem k nejasné podpoře použitých knihoven jsem se rozhodl setrvat u verze předchozí.

Aby si uživatelské rozhraní zachovalo jednotný vzhled a chování, použil jsem znovupoužitelné komponenty z kolekce **shadcn**. Shadcn je původně určený pro framework React, ale existuje i neoficiální port pro vývojáře používající Svelte, kde většina komponentů je převzata z Bits UI a Melt UI[34].

Vložený kód 4.3: Použití dialogového okna pomocí knihovny shadcn-svelte, převzato z [35]

```
<script lang="ts">
 import * as AlertDialog from "$lib/components/ui/alert-dialog";
</script>


<AlertDialog.Root>
 <AlertDialog.Trigger>Open</AlertDialog.Trigger>
 <AlertDialog.Content>
  <AlertDialog.Header>
   <AlertDialog.Title>Are you absolutely sure?</AlertDialog.Title>
   <AlertDialog.Description>
    This action cannot be undone. This will permanently delete your account
    and remove your data from our servers.
   </AlertDialog.Description>
  </AlertDialog.Header>
  <AlertDialog.Footer>
   <AlertDialog.Cancel>Cancel</AlertDialog.Cancel>
   <AlertDialog.Action>Continue</AlertDialog.Action>
  </AlertDialog.Footer>
 </AlertDialog.Content>
</AlertDialog.Root>
```

U tradičních UI knihoven, jako je Bootstrap, jsou komponenty stahovány jako externí závislosti. U shadcn se namísto toho přímo kopírují jejich zdrojové kódy. Tím získává vývojář plnou kontrolu nad prvky a může je přizpůsobit podle potřeby jako na obrázku 4.4.



Obrázek 4.4: Srovnání upraveného dialogového okna (vlevo) s původní verzí (vpravo)

Součástí shadcn je také framework pro kaskádové styly **Tailwind CSS**, pomocí kterého jsou stylizovány veškeré komponenty. Tento framework umožňuje psát styly pomocí utility tříd přímo v HTML prvcích[36]. Díky tomu není nutné vytvářet vlastní třídy ani přepínat mezi CSS a HTML, a to je obzvlášť výhodné při práci se Svelte. Zároveň má předdefinovanou škálu barev, mezer a velikostí. To přispívá ke konzistentnosti napříč celým uživatelským rozhraním.

Vložený kód 4.4: Ukázka použití Tailwind CSS

```
<div class="bg-gray-100 flex flex-col items-center justify-center rounded-xl
    p-2 min-w-[8rem] h-16">
    <p class="text-center text-xs">{name}</p>
    <p class="text-center font-bold uppercase">{value}</p>
</div>
```

Při přepisu vloženého kódu 4.4 by bylo nutné nejprve podle osvědčených postupů vytvořit separátní soubor pro kaskádové styly. Poté by bylo potřeba ke každému HTML prvku přiřadit vlastní třídu a nakonec každou třídu stylizovat. Alternativně lze styly psát přímo v HTML, tak jako je tomu ve vloženém kódu 4.5, přesto je ale kód v Tailwind přehlednější a lépe se píše.

Vložený kód 4.5: Přepsání tříd z Tailwind CSS do CSS

```
<div style="background-color: #f3f4f6; display: flex; flex-direction: column;
    align-items: center; justify-content: center; border-radius: 0.75rem;
    padding: 0.5rem; min-width: 8rem; height: 4rem;">
     <p style="text-align: center; font-size: 0.75rem;">{name}</p>
     <p style="text-align: center; font-weight: bold; text-transform: uppercase;
    ">{value}</p>
</div>
```

Pro zobrazení pozice dronu na mapě a vykreslování trasy misí byla použita knihovna **svelte-maplibre**. Byla zvolena zejména proto, že je open-source, vizuálně ladí s komponentami ze shadcn a nevyžaduje použití API klíčů. Maplibre je fork poslední veřejně dostupné verze Mapbox GL a ve spojení s OpenMap Tiles umožňuje tvorbu map zcela bez použití placených či freemium služeb[37]. Díky existujícímu portu pro Svelte bylo možné knihovnu i přes částečně chybějící dokumentaci snadno integrovat do aplikace.

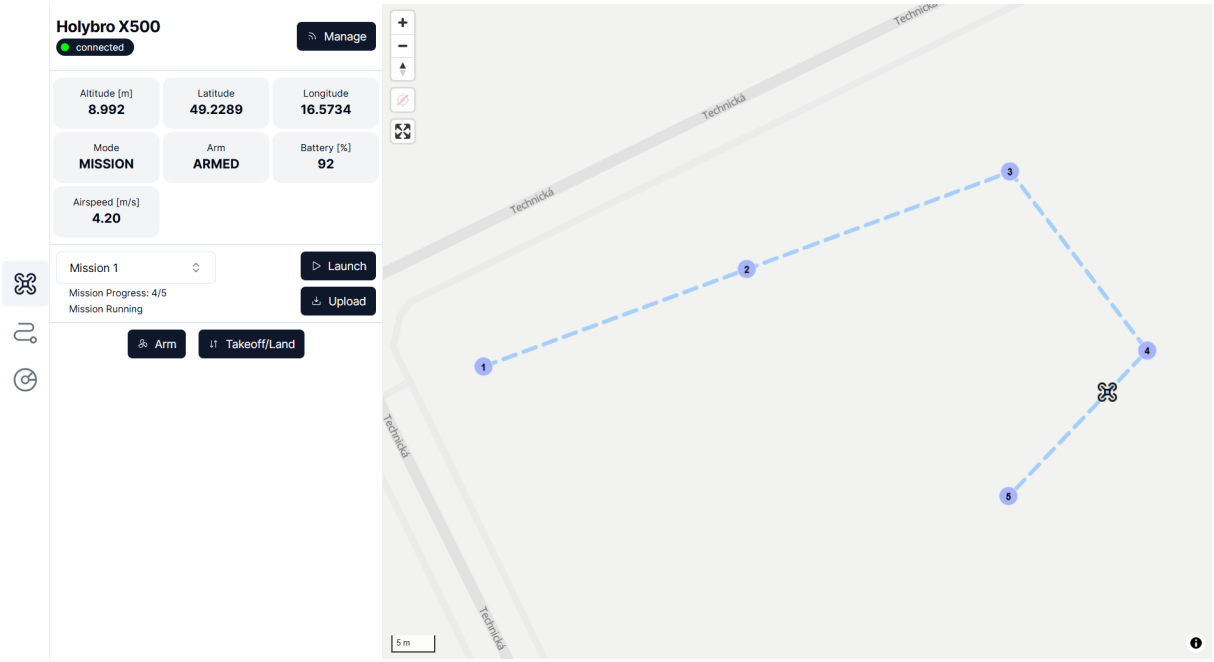
Data ze záznamu letu jsou vizualizována pomocí knihovny **Apache ECharts**, která podporuje širokou škálu grafických výstupů – od liniových a sloupcových grafů po koláčové či mapové vizualizace. Vysoká míra přizpůsobitelnosti umožnila sjednotit jejich vzhled se zbytkem uživatelského rozhraní. [38]

Také jsem využil knihovnu **Iconify**, která poskytuje přístup k obrovskému množství ikon z různých populárních služeb a projektů. Je navržena tak, aby byla snadno integrovatelná do webových stránek a aplikací, přičemž umožňuje použití ikon v SVG formátu nebo jako i fonty. Iconify obsahuje ikony z mnoha open-source projektů a nabízí uživatelsky přívětivé API pro jejich efektivní spravování a stylizaci[39]. Shadcn komponenty používají **Lucide Icons** [40], bohužel zde chybí značná část ikonek pro potřeby této práce (např. ikona kvadrokoptéry), a proto jsem se rozhodl použít převážně **Phosphor Icons**[41].

Struktura webového klienta

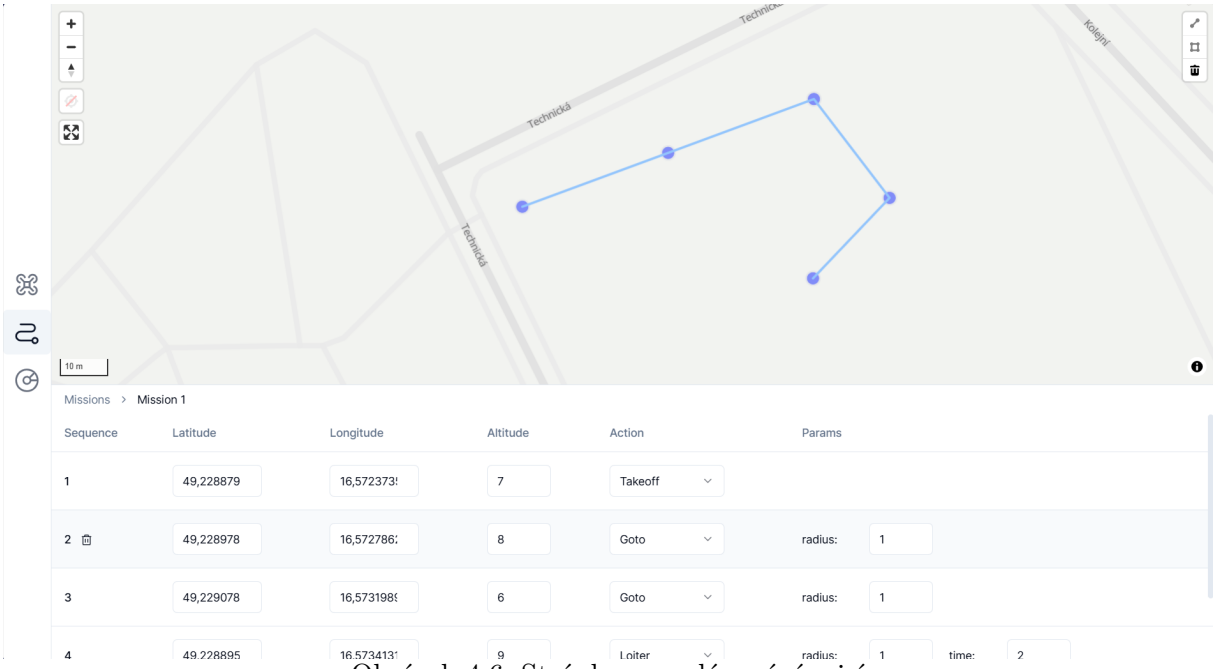
Webová aplikace se skládá ze tří podstránek: *analyze, fly* a *plan*. Zároveň obsahuje globální soubory, které zahrnují funkce a komponenty, jež nejsou závislé na konkrétní vybrané stránce. Konkrétně se v globálních souborech nacházejí například funkce pro zpracování dat z WSS a logika pro přepínání mezi jednotlivými stránkami.

Na podstránce *fly* uživatel ovládá dron. Zde má možnost spravovat připojení s dronem. Uživatel si může zvolit jeden ze tří komunikačních protokolů pro připojení k letovému ovladači a zároveň si k nim vybrat odpovídající port. Na stránce jsou rovněž zobrazována živá data z dronu jako je stav baterie, výška, rychlost či letový režim. Z kombinovaného pole si může vybrat uloženou misi a pomocí bočních tlačítek ji nahrát nebo rovnou spustit na dronu. Níže se nachází tlačítka pro přímé ovládání: tlačítko *arm* pro roztočení vrtulí a tlačítko *takeoff/land* pro vzlet nebo přistání v závislosti na tom, jestli je dron ve vzduchu.



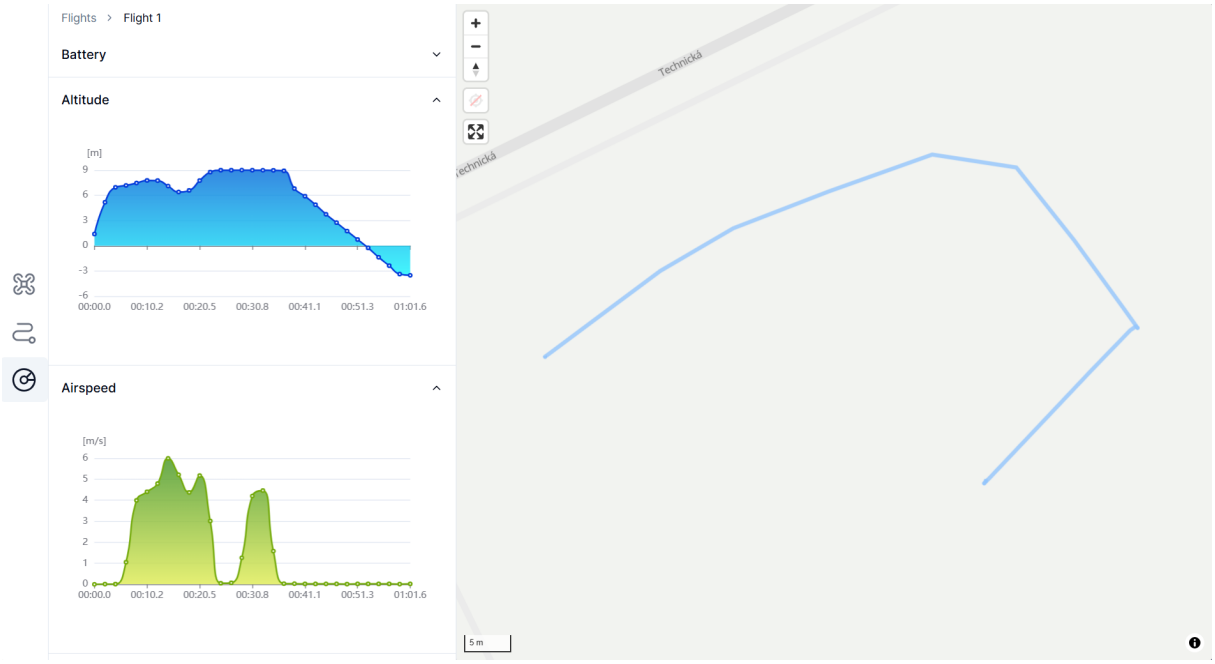
Obrázek 4.5: Stránka pro ovládání dronu

Podstránka *plan* je určena pro tvorbu a úpravu misí. Uživatel má možnost výběru mise k úpravě prostřednictvím tabulky, přičemž také může využít tlačítko pro vytvoření nové mise. Trasu může uživatel vytvářet, mazat a měnit pomocí mapy nebo interaktivní tabulky, které jsou mezi sebou propojeny. Jestliže uživatel upraví trasu na mapě, data zobrazená v tabulce se ihned změní. Přepínání zpět na výběr mise zajišťuje drobečková navigace. Aplikace zajišťuje automatické ukládání provedených změn v reálném čase, což znamená, že uživatel není povinen žádným způsobem potvrzovat uložené změny.



Obrázek 4.6: Stránka pro plánování misí

Na poslední podstránce má uživatel možnost vybrat záznam provedeného letu. Po výběru se na mapě zobrazí trajektorie letu a grafy měřených vlastností v čase. Grafy jsou barevně odlišeny a lze je pro lepší přehlednost zmenšit.



Obrázek 4.7: Stránka pro zobrazení záznamu letu

V následující struktuře souborového systému webového klienta nejsou zahrnuty soubory, které jsou běžně generovány automaticky, když je inicializován Node.js server.

- `src/`
  - `lib/` — Obsahuje shadcn komonenty
  - `stores/` — Proměnné sdílené mezi komponenty
  - `routes/`
    - `+layout.svelte` — Globální soubor
    - `sidebar.svelte` — Postranní lišta
    - `analyze/` — Soubory pro stránku s analýzou letu
      - `analyze.svelte` — Hlavní soubor pro stránku
      - `chart.svelte` — Šablona pro grafy
      - `delete-flight.svelte` — Tlačítko pro mazání záznamů
      - `flight-breadcrumb.svelte` — Drobečková navigace
      - `flight-table.svelte` — Tabulka letových záznamů
      - `telemetry-charts.svelte` — Okno s vizualizovanými daty
    - `fly/` — Stránka pro ovládání dronu
      - `fly.svelte` — Hlavní soubor pro stránku
      - `comms-module.svelte` — Widget pro připojení k dronu
      - `data-bubble.svelte` — Bublina pro zobrazení telemetrie
      - `drone-info.svelte` — Okno s ovládacími prvky
      - `mission-Module.svelte` — Widget pro výběr a spouštění misí
      - `telemetry-module.svelte` — Widget pro zobrazení telemetrie
    - `plan/` — Stránka pro plánování misí
      - `plan.svelte` — Hlavní soubor pro stránku
      - `add-mission.svelte` — Tlačítko pro přidání mise
      - `delete-mission.svelte` — Tlačítko pro odstranění mise
      - `delete-waypoint.svelte` — Tlačítko pro odstranění bodu
      - `mapPlan.svelte` — Mapa pro zobrazení trasy
      - `mission-breadcrumb.svelte` — Drobečková navigace
      - `mission-table.svelte` — Tabulka pro výběr misí
      - `waypoint-table.svelte` — Tabulka waypointů

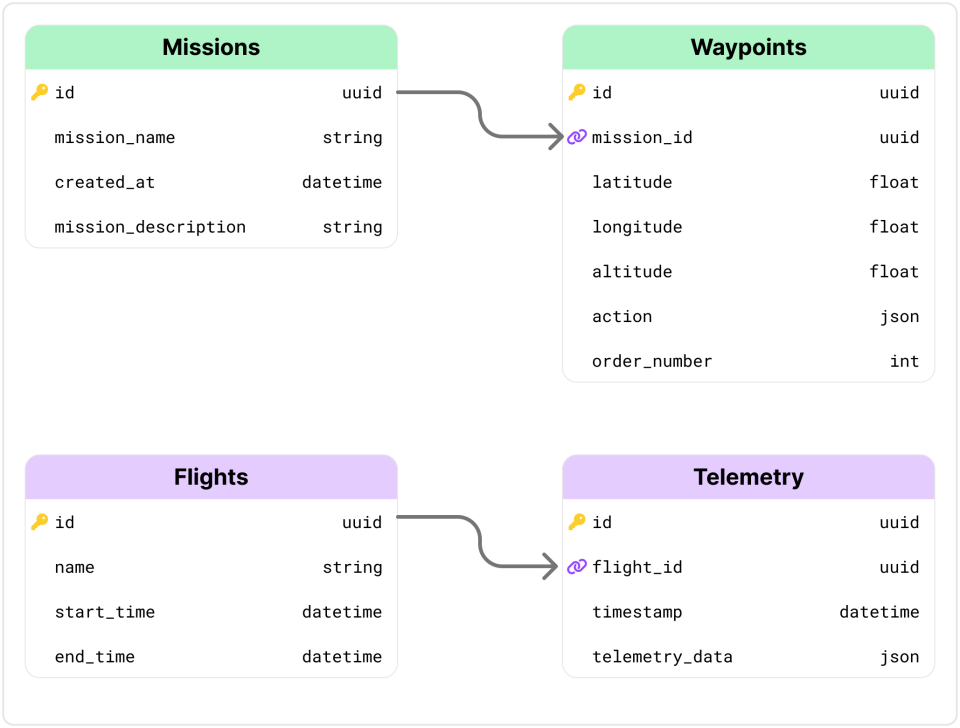
4.2.2 Backend server a databáze

Aby bylo možné začlenit server do systému, musíme znát místa, přes které povede komunikace. Taková místa se nazývají koncové body a lze je podle funkce rozdělit do tří kategorií:

- **Přímé ovládání** probíhá vždy přes HTTP GET požadavek. Server zprávu nijak nemodifikuje a přímo ji přeposílá dronu bez jakýchkoli úprav.
- **CRUD operace** manipulují s databází. Může se jednat o správu misí uživatelem anebo o tvorbu letových záznamů dronem. Součástí musí být i validace dat a požadavků, aby se zabránilo neočekávaným chybám, jako je například posílání mise s neznámými akcemi letovému ovladači.
- **Živá data** se posílají přes Websocket spojení. Koncový bod z pohledu webového klienta se nachází na `ws://127.0.0.1:8000/`. Podle IP adresy backend pozná, zda se připojil dron anebo klient.

Součástí backendového repozitáře je také relační databáze, která se skládá ze čtyř hlavních tabulek: `Missions`, `Waypoints`, `Flights` a `Telemetry`. Tabulka `Missions` uchovává základní informace o uložených misích (např. název, čas vzniku), zatímco `Waypoints` obsahuje konkrétní body, ze kterých se skládají mise. Tabulka `Flights` slouží k evidenci proběhlých letů a jejich záznam v čase zaznamenává tabulka `Telemetry`.

V relačních databázích je každá tabulka identifikována svým primárním klíčem (PK), který jednoznačně určuje každý záznam. Propojování mezi tabulkami (např. přiřazení waypointů k misi) je realizováno pomocí cizích klíčů (FK), které odkazují na primární klíče v jiných tabulkách. Díky těmto vazbám lze jednoduše navazovat související data, například získat všechny body trasy přiřazené ke konkrétní misi.



Obrázek 4.8: Schéma relační databáze

Backendový server spravuje webový framework **FastAPI** pro Python, postavený na Starlette. Patří mezi nejrychlejší Python frameworky a jeho hlavní výhodou je široká podpora různých nástrojů zaměřujících se na bezpečnost (OAuth2, OpenID Connect) anebo prací s databází (SQLAlchemy, Pydantic). Zároveň nabízí automatickou dokumentaci endpointů pomocí knihovny Swagger UI [42]. Seznam dokumentovaných koncových bodů pro GCS je dostupný na adrese `http://127.0.0.1:8000/docs`.

Vložený kód 4.6: Ukázka endpointu pro ovládání dronu

```
@api_router.get("/arm")
async def arm():
    async with httpx.AsyncClient() as client:
        response = await client.get(f"{DRONE_BACKEND_URL}/arm")
    return response.json()  # Přepošle odpověď zpět klientovi
```

Pro validaci dat pomocí datových modelů jsem použil knihovnu **Pydantic**, která automaticky ověřuje, zda data odpovídají zadaným typům v modelech, a v případě potřeby je konvertuje (např. převod řetězce na datum). Umožňuje definovat vlastní pravidla (číslo pořadí waypointu musí být nezáporné číslo) a mimo slovníky v Pythonu podporuje i JSON. [43]

Vložený kód 4.7: Datový model pro Waypoint

```
from pydantic import BaseModel


class WaypointSchema(BaseModel):
    latitude: float
    longitude: float
    altitude: float
    action: Union[GotoSchema, LoiterSchema, TakeoffSchema, LandSchema,
   RtlSchema, HomeSchema]
    order_number: int
```

Socket.io je knihovna, která v této práci usnadňuje manipulaci s WSS a poskytuje účinné rozhraní pro real-time komunikaci mezi klientem a serverem. Automaticky řeší časté problémy jako například obnova spojení po detekci výpadku, použití HTTP long-polling při neúspěchu navázání spojení nebo automatické parsování dat. Aby bylo možné využívat funkce této knihovny, musí být knihovna integrována jak v klientovi, tak v softwaru na straně dronu. [44]

Vložený kód 4.8: Ukázka použití Socket.io

```
import socketio
import asyncio


@sio.on("connect")
async def connect(sid):
   print(f"Frontend {sid} connected to GCS WebSocket")


@sio.on("disconnect")
async def disconnect(sid):
    print("Client Disconnected: "+" "+str(sid))
```

Správu databáze zajišťuje knihovna **SQLAlchemy**. Poskytuje sadu nástrojů pro práci s relačními databázemi pomocí objektově-relačního mapování (ORM) [45]. Toto mi umožnilo manipulovat databázovými záznamy pomocí Pythonových objektů bez nutného použití přímých SQL dotazů. SQLAlchemy komunikuje s databázovým systémem PostgreSQL.

Vložený kód 4.9: Ukázka endpointu pro výpis všech bodů jedné mise

```
from app.api.deps import SessionDep


@router.get("/flights/{flight_id}/telemetry/")
async def get_flight_records(async_session: SessionDep, flight_id):
    stmt = select(Telemetry).where(Telemetry.flight_id == flight_id)
    result = await async_session.execute(stmt)
    return list(result.scalars())
```

V struktuře backendového serveru níže soubory `__init__.py` slouží k tomu, aby označily adresář jako balíček (package). Bez tohoto souboru by importování balíčků z adresáře nebylo možné tradičním způsobem.

- `/`
  - `app`
    - `api` ..... Globální soubor
    - `db_models` ..... Databázové modely
      - `__init__.py`
      - `base.py` ..... Základní třída SQLAlchemy
      - `flightRecord.py` ..... Modely pro tabulky *Flights* a *Telemetry*
      - `missionPlan.py` ..... Modely pro tabulky *Missions* a *Waypoints*
      - `waypoint_schemas.py` .... Validační modely pro waypointy (např. land)
    - `routes`
      - `__init__.py`
      - `crud_flights.py` ..... Koncové body pro práci s tabulkou *Flights*
      - `crud_missions.py` ..... Koncové body pro práci s tabulkou *Missions*
      - `crud_telemetry.py` ..... Koncové body pro práci s tabulkou *Telemetry*
      - `crud_waypoints.py` ..... Koncové body pro práci s tabulkou *Waypoints*
      - `mission_router.py` ..... Koncové body pro posílání a spouštění misí
    - `__init__.py`
    - `config.py` ..... Nastavení databáze
    - `db.py` ..... Inicializace databáze
    - `deps.py` ..... Závislosti využívané napříč různými soubory
    - `router.py` ..... Hlavní router, spojuje koncové body z `routes/`
    - `sockets.py` ..... Správa Websockets
  - `__init__.py`
  - `default_db.json` ..... Výchozí obsah databáze, určen pro testování
  - `main.py` ..... Hlavní FastAPI soubor
  - `README.md` ..... Pokyny k instalaci
  - `requirements.txt` ..... Seznam Python knihoven ke stažení

4.2.3 Řídící server

Na základě architektury popsané v kapitole 4.1 jsem přistoupil k návrhu serveru, který běží na doprovodném počítači. Cílem bylo zde vytvořit prostředníka mezi pozemní stanicí a samotným autopilotem běžícím na dronu.

Server byl navržen jako aplikace postavená na `FastAPI`, čímž se zachovala jednotnost v rámci celého systému. Důležitým úkolem řídícího serveru je efektivní práce s knihovnou pyMAVSDK, která umožňuje přímou komunikaci s dronem prostřednictvím MAVLink protokolu. V rámci této komunikace bylo nutné řešit zpracování asynchronních datových toků – zejména telemetrie – a zároveň poskytovat jednoduché, srozumitelné rozhraní pro základní ovládací akce (např. vzlet, přistání, spuštění mise).

Dron je zde implementován jako samostatný objekt – třída, která slouží jako abstrakce nad pyMAVSDK. Skládá se ze tří podtříd `Controls`, `Telemetry` a `Mission` a díky tomu je možné přehledně rozdělit kód do bloků podle druhu využití. Tento objekt je schopen:

- navazovat spojení s dronem,
- pracovat s misemi (načítání, nahrávání, zahájení, pozastavení),
- zpracovávat a poskytovat aktuální živou telemetrii pomocí asynchronních generátorů,
- a zároveň poskytovat jednoduché rozhraní pro základní řídicí příkazy.

Vložený kód 4.10: Ukázka struktry třídy pro ovládání dronu s asynchronním generátorem

```
from mavsdk import System, mission_raw

class DroneManager:

    def __init__(self,connection_address):

        self.drone = System() # Nová instance MAVSDK třídy

        self.telemetry = self.Telemetry(self) # Propojení DroneManager s
    Telemetry


    class Telemetry:

        def __init__(self, manager):

            self.manager = manager


        async def flight_mode(self):

            async for mode in self.manager.drone.telemetry.flight_mode():

                yield mode.name
```

- `/`
  - `deps.py` — Závislosti využívané napříč různými soubory
  - `flight_recorder.py` — Automaticky zaznamenává parametry letu
  - `main.py` — Hlavní FastAPI soubor
  - `schemas.py` — Validační modely pro přijaté mise
  - `objects/`
    - `__init__.py`
    - `drone.py` — Objekt dronu
  - `routes/`
    - `__init__.py`
    - `router.py` — Spravuje HTTP komunikaci
    - `sockets.py` — Spravuje Websocket komunikaci
  - `README.md` — Pokyny k instalaci
  - `requirements.txt` — Seznam Python knihoven ke stažení

5 Testování

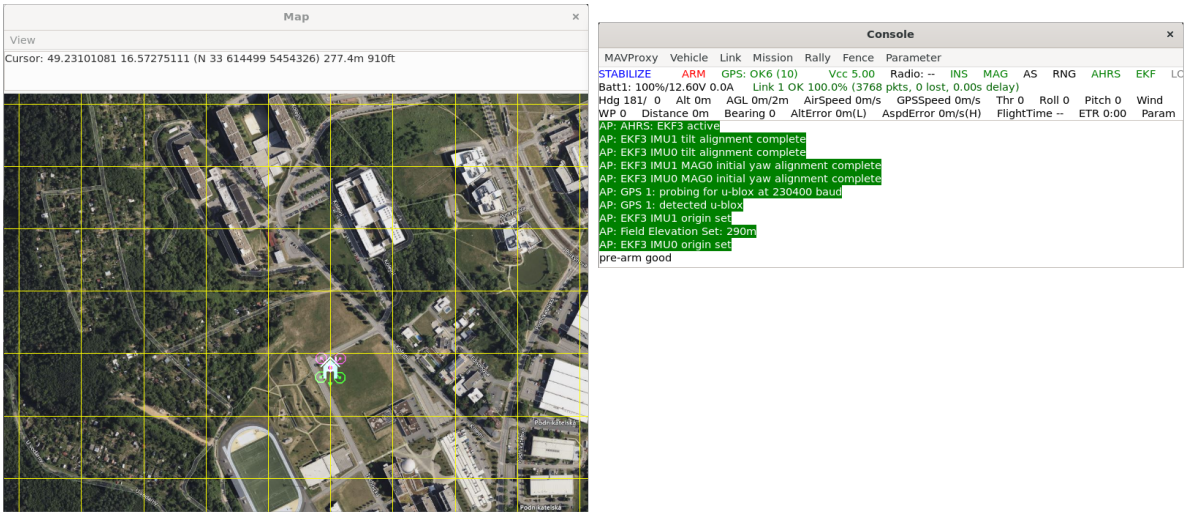
Tato kapitola se věnuje testování navrženého systému, a to jak ve virtuálním prostředí, tak i v reálných podmínkách s fyzickým dronem. Cílem testování bylo ověřit funkčnost jednotlivých částí systému, jejich vzájemnou komunikaci a schopnost spolehlivě řídit UAV.

Testování v simulaci

V průběhu řešení práce jsem průběžně testoval systém na vlastním počítači bez použití hardwaru, který se nachází na dronu. Díky tomu jsem měl možnost včasně zjistit nedostatky a zabránit chybám, které by mohly vést k neúspěšnému vzletu, nebo pádu a ztrátě dronu.

K tomu, abych mohl lokálně spustit řídící server na dronu, stačilo přepsat IP adresu v `.env` souboru, který se nachází v backendovém repozitáři. V takovém případě je IP adresa řídícího serveru `127.0.0.1:8080`.

Dron a letový ovladač jsem nahradil programem **Ardupilot SITL (Software in the Loop)**, který umožňuje například simulovat letadla (Plane), pozemní drony (Rover) a kvadrokoptéry (Copter) [46]. Součástí simulátoru je konzole, kde je možné si zobrazit přijaté zprávy, základní telemetrii a mapu s aktuální pozicí dronu. Komunikace se systémem probíhala pomocí protokolu MAVLink přes UDP port `:14540`. Pro testování jsem výhradně používal simulaci kvadrokoptéry a vývoj probíhal na WSL (Windows Subsystem for Linux).



Obrázek 5.1: Uživatelské rozhraní pro Ardupilot SITL

K otestování a ladění aplikačního programového rozhraní (API) na backendovém serveru a na doprovodném počítači jsem využil nástroj **Postman**[47]. Umožňuje jednoduše posílat HTTP požadavky (GET, POST, PUT, DELETE) na různé koncové body a zobra-

zovat odpovědi. Také umožňuje rychle modifikovat hlavičky a těla zpráv a automatizovat testování. Tento nástroj umí generovat dokumentaci, kterou lze sdílet s ostatními vývojáři.

Testování na reálném dronu

K testování mi byl v rámci spolku vyhrazen dron Holybro X500 s letovým ovladačem řady Pixhawk 6C a doprovodným počítačem typu Intel NUC. Vzdálené připojení dronu s počítačem bylo zajištěno přes WiFi hotspot na pozemní stanici. Stav připojení zobrazovala konzole backend serveru. Přestože celý systém byl v simulaci testován na Linux zařízení, backendový server a klient pracoval na Windows 11.

K letovému ovladači byl během testování připojen zároveň RC vysílač, který sloužil jako bezpečnostní prvek. K RC vysílači byl připojen program Mission Planner, přes které se nastavovaly parametry určující maximální rychlost letu a virtuální plot (geofence), přes který dron nesměl proletět. V případě nečekané poruchy bylo možné přepnout letový režim a dron přímo ovládat.

Obrázek 5.2: Dron Holybro X500 použitý pro testování

Manuální spouštění serveru na doprovodném počítači by znamenalo nutnost před letem zapojit k NUC počítači klávesnici s myší a monitor a manuálně buď napsat Shellový skript anebo zmáčknout tlačítko start v textovém editoru. Proto by měl server automaticky běžet se startem počítače. K tomu bylo nutné vytvořit `systemd .service` soubor.

Ve vloženém kódu 5.1 definujeme, že server se spustí 30 sekund poté, co se počítač připojí síti a zařízení musí být spojeno s letovým ovladačem přes port `/dev/tty/USB0`. Zároveň musíme programu udělit práva pro čtení a posílání dat přes tento port. Práva se nastavují pomocí příkazu `chmod`. V případě neúspěšného startu serveru se `systemd` pokusí restartovat FastAPI server.

Vložený kód 5.1: Script pro automatické spouštění FastAPI serveru

```
[Unit]
Description=FastAPI Drone Server
Wants=network-online.target dev-ttyUSB0.device
After=network-online.target dev-ttyUSB0.device

[Service]
User=username
Group=dialout
PermissionsStartOnly=true
WorkingDirectory=/home/username/drone/drone-server
Environment=PATH=/home/username/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/
    sbin:/usr/bin
ExecStartPre=/bin/sleep 30
ExecStartPre=/bin/chmod 666 /dev/ttyUSB0
ExecStart=/home/username/.local/bin/uvicorn main:app \
    --host 192.168.137.222 --port 8080
Restart=on-failure
Type=simple

[Install]
WantedBy=multi-user.target
```

Zhodnocení výsledků

V simulátoru byla testována tvorba misí a nahrávání waypointů. Dále byla zkoumána funkčnost posílání a spouštění misí. Součástí testování bylo také armování dronu a aktivace tlačítka pro vzlet a přistání. Posílání misí a spouštění bylo také otestováno i na skutečném dronu. Ten byl schopen odletět automaticky naplánovanou trasu v požadovaných letových hladinách. Přestože signál se posílal přes WiFi hotspot, který není určený k vzdálenému ovládání, komunikace fungovala bez zvýšené latence i na vzdálenost přibližně 50 metrů.

Přesto testování poukázalo na nedostatky systému, které můžou zhoršit uživatelskou zkušenost s ovládáním dronu. Mezi známé problémy patří nespolehlivé zobrazení kapacity baterie na skutečném dronu. Letový ovladač se zapnutím vždy zobrazuje 100% kapacitu baterie, přestože baterie nemusí být plně nabitá. Uživatel tak nemůže spolehlivě určit stav baterie. Řešením by zde mohlo být použití chytrých baterií, které měří více parametrů a lépe odhadují rychlost vybíjení. Dalším možným řešením je zkalibrovat parametry autopilota pro danou baterii. Chybné měření kapacity baterie nemá vliv na failsafe me-

chanismus pro detekci vybité baterie, neboť tento mechanismus srovnává naměřené napětí baterie s minimální povolenou hodnotou, která je uložená jako parametr v Ardupilot autopilotu.

Občas doprovodný počítač není schopen získat živou telemetrii, pokud není k letovému ovladači nejprve připojena RC vysílačka. Pravděpodobným problémem je, že knihovna pyMAVSDK v některých případech po připojení nezačne pravidelně vysílat zprávy typu heartbeat. Četnost tohoto problému se liší dle použitého dronu i pyMAVSDK skriptu. Řešením by bylo manuálně v pravidelných intervalech tyto zprávy posílat do letového ovladače, ale to v současné době pyMAVSDK nepodporuje. Tento problém se ani jednou nevyskytl během testování v simulaci.

Nevýhodou webového klienta je nutnost připojení k internetu pro zobrazení mapy s polohou dronu. Pro venkovní testování pozemní stanice byla data z internetu posílána z mobilní sítě přes hotspot z telefonu. Některé notebooky mají slot pro SIM kartu a umožňují tak získávat data přímo z mobilní sítě.

Obrázek 5.3: Testovací let dronu

6 Závěr

Tato bakalářská práce se podrobně zabývala problematikou dálkového řízení dronů, přičemž hlavním cílem bylo navrhnout a implementovat komplexní řídicí systém, který využívá architekturu založenou na interakci mezi klientem, serverem a samotným dronem.

V rešeršní části byla rozebrána problematika efektivního přenosu dat pomocí protokolu MAVLink. V rámci této analýzy byly prozkoumány klíčové charakteristiky protokolu, včetně jeho struktury a způsobu, jakým zajišťuje spolehlivost a bezpečnost přenosů informací. Práce se dále zabývá knihovnou MAVSDK, která nabízí abstrakci nad daným protokolem. Součástí bylo také porovnání nativní a cloudové architektury softwarových řešení pro pozemní stanice s konkrétními příklady existujících programů.

Byla také popsána hardwarová část řídících systémů, do které spadal princip funkce letového ovladače s autopilotem, který běžel jako operační systém v reálném čase. Byly popsány základní senzory nutné k regulaci letu a možné využití doprovodných počítačů pro pokročilejší funkce, jako je správa komunikace, lokální zpracování dat a autonomní plánování trasy.

Dále se práce zabírala samotným návrhem architektury systému, kde se backendový server nacházel přímo na pozemní stanici (tzv. on premise řešení). Komunikace mezi klientem, serverem a doprovodným počítačem probíhala přes webové protokoly HTTP a WSS, zatímco doprovodný počítač komunikoval s letovým ovladačem pomocí protokolu MAVLink.

Grafické rozhraní webového klienta umožňovalo spravovat připojení a ovládat dron, přehledně zobrazovat živá data z letového ovladače, navrhovat trasy mise a interaktivně pomocí mapy a grafů zobrazovat v čase měřené parametry proběhlých letů. Reaktivní chování aplikace zajišťuje framework Svelte a k urychlení vývoje a sjednocení vizuálního stylu byly použity komponenty z kolekce shadcn. Součástí této kolekce byl také framework pro psaní kaskádových stylů Tailwind CSS. Kreslení trajektorií a zobrazování polohy dronu umožňovala knihovna Maplibre a zobrazení grafů naměřené telemetrie knihovna Apache ECharts.

REST API pro backendový server byl napsán v Pythonovském frameworku FastAPI. Pro komunikaci mezi aplikací a PostgreSQL databází byl implementován ORM systém knihovnou SQLAlchemy.

Všechny funkce systému byly úspěšně testovány na simulátoru Ardupilot SITL i na skutečném dronu. Zvolené technologie se ukázaly jako efektivní a vhodné pro dané úkoly. Aplikace je funkční a dron je automaticky schopen odletět naplánované mise.

Přesto se během testování objevily nedostatky systému. Mapa ve webovém klientu je závislá na připojení k internetu, který nemusí být vždy dostupný. Zároveň systém neumí spolehlivě zobrazit kapacitu baterie, přesto fungují bezpečnostní mechanismy, které zabraňují pádům dronu při jejím vybití.

V rámci spolku se předpokládá další rozvíjení systému. K doprovodnému počítači se v budoucnosti připojí dvě kamery, nichž jedna bude snímat infračervené spektrum. Cílem bude detekce objektů pomocí neuronových sítí typu YoLo (You only Look Once).

Seznam zkratek

CRC	Cyclic Redundancy Check
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DOM	Document Object Model
FK	Foreign Key
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
GPU	Graphical processing unit
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IMU	Inertial Measurement Unit
IP	Internet Protocol
JS	JavaScript
MAVSDK	Micro Air Vehicle Software Development Kit
MAVlink	Micro Air Vehicle Link
MCU	Microcontroller Unit
MPU	Microprocessor Unit
ORM	Object Relational Mapping
PK	Personal Key
RTOS	Real-Time Operating System
SPA	Single Page Application
SSE	Server-Sent Events
STANAG	Standardization Agreement

TCP Transmission Control Protocol

UAV Unmanned Aerial Vehicle

WSS Websocket

Literatura

[1] WILLEE, Hamish. *MAVLink Developer Guide*. 2017. Dostupné také z: https://mavlink.io/en/.

[2] DIETRICH, Thomas; ANDRYEYEV, Oleksandr; ZIMMERMANN, Armin; MITSCHELE-THIEL, Andreas. Towards a Unified Decentralized Swarm Management and Maintenance Coordination Based on MAVLink. In: *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2016, s. 124–129. Dostupné z DOI: 10.1109/ICARSC.2016.64.

[3] KOUBÂA, Anis; ALLOUCH, Azza; ALAJLAN, Maram; JAVED, Yasir; BELGHITH, Abdelfettah; KHALGUI, Mohamed. Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey. *IEEE Access*. 2019, roč. 7, s. 87658–87680. Dostupné z DOI: 10.1109/ACCESS.2019.2924410.

[4] MEIER, Lorenz; WILLEE, Hamish; OES, Julian; ŁUKASIEWICZ, Marek S.; BOES, Cedric. *Packet Serialization*. 2017. Dostupné také z: https://mavlink.io/en/guide/serialization.html.

[5] MAVLINK DEVELOPERS. *checksum.h - MAVLink C Library v2* [online]. 2025. [cit. 2025-04-20]. Dostupné z: https://github.com/mavlink/c_library_v2/blob/master/checksum.h. GitHub repository.

[6] WILLEE, Hamish. *Message Signing*. 2017. Dostupné také z: https://mavlink.io/en/.

[7] ALLOUCH, Azza; CHEIKHROUHOU, Omar; KOUBÂA, Anis; KHALGUI, Mohamed; ABBES, Tarek. MAVSec: Securing the MAVLink Protocol for Ardupilot/PX4 Unmanned Aerial Systems. In: *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*. 2019, s. 621–628. Dostupné z DOI: 10.1109/IWCMC.2019.8766667.

[8] *STANAG 4586 Edition 4: Standard Interfaces of UA Control System (UCS) for NATO UA Interoperability – AEP-84 Edition A*. NATO Standardization Office, 2017. Schváleno CNAD/AC/141 NNAG/JCGUAS. Dostupné z: https://nso.nato.int/nso/nsdd/main/standards/stanag-details/8989/EN.

[9] RODRIGUES, Alexandre Valério; CARAPAU, Rodolfo Santos; MARQUES, Mario Monteiro; LOBO, Victor; COITO, Fernando. Unmanned systems interoperability in military maritime operations: MAVLink to STANAG 4586 bridge. In: *OCEANS 2017-Aberdeen*. IEEE, 2017, s. 1–5.

[10] NOES, Julian; WILLEE, Hamish. *MAVSDK (main)*. 2017. Dostupné také z: https://mavsdk.mavlink.io/main/en/.

[11] NOES, Julian; WILLEE, Hamish. *MAVSDK | FAQ*. 2017. Dostupné také z: https://mavsdk.mavlink.io/main/en/faq.html.

[12] MAVLINK DEVELOPERS. *MAVSDK-Python*. GitHub, 2025. Dostupné také z: https://github.com/mavlink/MAVSDK-Python.

[13] MAVLINK DEVELOPERS. *telemetry_takeoff_and_land.py – MAVSDK-Python Example*. GitHub, 2025. Dostupné také z: https://github.com/mavlink/MAVSDK-Python/blob/main/examples/telemetry_takeoff_and_land.py.

[14] NEX, Francesco; REMONDINO, Fabio. UAV for 3D mapping applications: a review. *Applied Geomatics*. 2014, roč. 6, č. 1, s. 1–15. ISSN 1866-928X. Dostupné z DOI: 10.1007/s12518-013-0120-x.

[15] PIX4D. *PIX4Dmatic 1.43: project merging and indoor workflows!* 2023-04. Dostupné také z: https://www.pix4d.com/blog/PIX4Dmatic-project-merging-indoor-workflows/. [cit. 2025-04-23].

[16] MICHAEL OBORNE AND ARDUPILOT DEVELOPMENT TEAM. *Mission Planner: Ground Control Station for ArduPilot*. 2025. Dostupné také z: https://github.com/ArduPilot/MissionPlanner. [cit. 2025-05-23].

[17] DRONECODE PROJECT. *QGroundControl: Cross-platform Ground Control Station for MAVLink Drones*. 2025. Dostupné také z: https://github.com/mavlink/qgroundcontrol. [cit. 2025-05-23].

[18] HU, Lyuyang; PATHAK, Omkar; HE, Zeyu; LEE, Hunkyu; BEDWANY, Mina; MICA, Jace; BURKE, Peter J. "CloudStation:" A Cloud-Based Ground Control Station for Drones. *IEEE Journal on Miniaturization for Air and Space Systems*. 2021, roč. 2, č. 1, s. 36–42. Dostupné z DOI: 10.1109/JMASS.2020.3027520.

[19] ANDURIL INDUSTRIES. *Lattice Mesh*. 2025. Dostupné také z: https://www.anduril.com/lattice-mesh/. [cit. 2025-04-23].

[20] DJI. *Drone Security White Paper, Version 3.0*. 2024-04. Tech. zpr. DJI. Dostupné také z: https://terra-1-g.djicdn.com/851d20f7b9f64838a34cd02351370894/trust%20center/DJI%20Drone%20Security%20White%20Paper.pdf. [cit. 2025-04-23].

[21] EBEID, Emad; SKRIVER, Martin; TERKILDSEN, Kristian Husum; JENSEN, Kjeld; SCHULTZ, Ulrik Pagh. A survey of Open-Source UAV flight controllers and flight simulators. *Microprocessors and Microsystems*. 2018, roč. 61, s. 11–20. ISSN 0141-9331. Dostupné z DOI: https://doi.org/10.1016/j.micpro.2018.05.002.

[22] GARG, P. K. *Unmanned Aerial Vehicles*. Sciendo, 2021. Dostupné z DOI: 10.1515/9781683927082.

[23] PIXHAWK SPECIAL INTEREST GROUP. *DS-009 Pixhawk Connector Standard*. 2020. Tech. zpr. Dronecode Foundation. Dostupné také z: https://github.com/pixhawk/Pixhawk-Standards/blob/master/DS-009%20Pixhawk%20Connector%20Standard.pdf. Verze 0.1.0.

[24] PX4 AUTOPILOT PROJECT. *Holybro Pixhawk 5x Wiring Quick Start*. 2025. Dostupné také z: https://docs.px4.io/main/en/assembly/quick_start_pixhawk5x.html. [cit. 2025-05-03].

[25] AL-KAFF, Abdulla; MARTÍN, David; GARCÍA, Fernando; ESCALERA, Arturo de la; MARÍA ARMINGOL, José. Survey of computer vision algorithms and applications for unmanned aerial vehicles. *Expert Systems with Applications*. 2018, roč. 92, s. 447–463. ISSN 0957-4174. Dostupné z DOI: https://doi.org/10.1016/j.eswa.2017.09.033.

[26] MCENROE, Patrick; WANG, Shen; LIYANAGE, Madhusanka. A Survey on the Convergence of Edge Computing and AI for UAVs: Opportunities and Challenges. *IEEE Internet of Things Journal*. 2022, roč. 9, č. 17, s. 15435–15459. Dostupné z DOI: 10.1109/JIOT.2022.3176400.

[27] GRAY, Alexander G.; GONZALEZ, Felipe; VANEGAS, Fernando; GALVEZ-SERNA, Julian; MORTON, Kye. Design and Flight Testing of a UAV with a Robotic Arm. In: *2023 IEEE Aerospace Conference*. 2023, s. 1–13. Dostupné z DOI: 10.1109/AERO55745.2023.10115880.

[28] CAMPION, Mitch; RANGANATHAN, Prakash; FARUQUE, Saleh. *A Review and Future Directions of UAV Swarm Communication Architectures*. 2017. Tech. zpr. University of North Dakota, Department of Electrical Engineering. Dostupné také z: https://und.edu/research/rias/_files/docs/swarm_ieee.pdf.

[29] ROZBOUD, Jakub. *Multiagentní systém dronů a robotů řízený umělou inteligencí pomůže armádě* [https://www.vut.cz/i/media/document_images/fotogalerie_doc/ostra/282938/FEKT_-_UARS_C__t_reporta__z__-3_1600.jpg]. 2025. Fotografie publikovaná na webu VUT.

[30] PX4 AUTOPILOT PROJECT. *Companion Computers*. 2025. Dostupné také z: https://docs.px4.io/main/en/companion_computer/. [cit. 2025-05-22].

[31] GENERAL ATOMICS AERONAUTICAL SYSTEMS, INC. *Ground Control Stations*. 2025. Dostupné také z: https://www.ga-asi.com/ground-control-stations/. [cit. 2025-05-23].

[32] SVELTE CONTRIBUTORS. *Svelte v4: Cybernetically enhanced web apps*. 2023. Dostupné také z: https://v4.svelte.dev/. [cit. 2025-05-17].

[33] HERNANDEZ, Ian. *Svelte vs. React: The Ultimate JavaScript Framework Showdown*. 2024-08. Dostupné také z: https://www.dreamhost.com/blog/svelte-vs-react/. [cit. 2025-05-08].

[34] SHADCN; HUNTABYTE. *Shadcn-svelte: Beautifully designed components that you can copy and paste into your apps. Accessible. Customizable. Open Source.* 2025. Dostupné také z: https://www.shadcn-svelte.com/. [cit. 2025-05-17].

[35] SHADCN; HUNTABYTE. *Alert Dialog – shadcn-svelte*. 2024-08. Dostupné také z: `https://www.shadcn-svelte.com/docs/components/alert-dialog`. [cit. 2025-05-08].

[36] TAILWIND LABS. *Tailwind CSS: A Utility-First CSS Framework*. 2025. Dostupné také z: `https://tailwindcss.com/`. [cit. 2025-05-17].

[37] IMFELD, Daniel. *Svelte MapLibre: Svelte Bindings for the MapLibre Mapping Library*. 2025. Dostupné také z: `https://svelte-maplibre.vercel.app/`. [cit. 2025-05-17].

[38] APACHE SOFTWARE FOUNDATION. *Apache ECharts: An Open Source JavaScript Visualization Library*. 2025. Dostupné také z: `https://echarts.apache.org/en/index.html`. [cit. 2025-05-17].

[39] TRUSHKIN, Vjacheslav. *Iconify: All Popular Icon Sets, One Framework*. 2025. Dostupné také z: `https://iconify.design/`. [cit. 2025-05-17].

[40] FENNIS, Eric; RIGÓ, Karsa; GUDDAS, Jakob. *Lucide: Beautiful & Consistent Icons Made by the Community*. 2025. Dostupné také z: `https://lucide.dev/`. [cit. 2025-05-17].

[41] ZHANG, Helena; FRIED, Tobias. *Phosphor Icons: A Flexible Icon Family for Interfaces*. 2025. Dostupné také z: `https://phosphoricons.com/`. [cit. 2025-05-17].

[42] RAMÍREZ, Sebastián. *FastAPI: Modern, Fast (High-Performance) Web Framework for Building APIs with Python*. 2025. Dostupné také z: `https://fastapi.tiangolo.com/`. [cit. 2025-05-17].

[43] COLVIN, Samuel. *Pydantic: Data Validation and Settings Management Using Python Type Annotations*. 2025. Dostupné také z: `url%7Bhttps://docs.pydantic.dev/latest/%7D`. [cit. 2025-05-17].

[44] RAUCH, Guillermo. *Socket.IO: Bidirectional and Low-Latency Communication for Every Platform*. 2025. Dostupné také z: `https://socket.io/`. [cit. 2025-05-17].

[45] BAYER, Michael. *SQLAlchemy: The Python SQL Toolkit and Object Relational Mapper*. 2025. Dostupné také z: `https://www.sqlalchemy.org/`. [cit. 2025-05-17].

[46] ARDUPILOT DEVELOPMENT TEAM. *SITL Simulator (Software in the Loop)*. 2025. Dostupné také z: `https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html`. [cit. 2025-05-17].

[47] POSTMAN, INC. *Postman: The World's Leading API Platform*. 2025. Dostupné také z: `https://www.postman.com/`. [cit. 2025-05-17].

Seznam příloh

- Soubor **`frontend.zip`** obsahuje zdrojový kód ke klientskému serveru
- Soubor **`backend.zip`** obsahuje zdrojový kód k backendovému serveru a nastavení databáze
- Soubor **`drone-server.zip`** obsahuje zdrojový kód k řídícímu serveru dronu