

**Gymnázium, Praha 6, Arabská 14**

Programování



**MATURITNÍ PRÁCE**

**Bezpilotní letadlo**

Vypracoval:

Havránek Kryštof 4E

Vedoucí práce:

ing. Daniel Kahoun

Březen 2022

Prohlašuji, že jsem jediným autorem této práce, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V ..... dne .....

Podpis autora

Název práce: Bezpilotní letadlo

Autor: Havránek Kryštof 4E

**Abstrakt:** Cílem práce je vytvořit autonomní letadlo, které bude pilot dálkově ovládat prostřednictvím softwaru na počítači. Z letadla bude během provozu dostávat potřebnou telemetrii a přesnos video. Software by měl být napsán stylem, který umožňuje budoucí vývoj.

Klíčová slova: (UAV), (Gtk), (C++), (Raspberry Pi zero 2)

---

Title: Drone

Author: Havránek Kryštof 4E

**Abstract:** The aim of the work is to create an unmanned aerial vehicle system which will be controlled remotely by the pilot from software on a computer. Pilot should receive all the necessary telemetry in addition to video stream from the aircraft. Software should be written in a style that allows for future development.

Key words: (UAV), (Gtk), (C++), (Raspberry Pi zero 2)

---

Title: Drohne

Autor: Havránek Kryštof 4E

**Abstrakt:** Ziel der Arbeit ist es, ein unbemanntes Luftfahrzeugsystem zu schaffen, das vom Piloten aus der Ferne von einer Software auf einem Computer gesteuert wird. Der Pilot sollte alle erforderlichen Telemetrie und Videostreams vom Flugzeug erhalten. Software sollte in einem Stil geschrieben werden, der eine zukünftige Entwicklung ermöglicht.

Schlüsselwörter: (UAV), (Gtk), (C++), (Raspberry Pi zero 2)

# Obsah

<b>Úvod</b>	<b>iv</b>
<b>1 Rozložení práce</b>	<b>1</b>
1.1 Komunikační protokol . . . . .	1
1.1.1 Struktura protokolu . . . . .	2
1.1.2 Specifikace protokolu . . . . .	3
1.2 Konfigurace . . . . .	3
<b>2 Hardware a Raspberry Pi</b>	<b>4</b>
2.1 Program . . . . .	7
2.2 Interface s Hardwarem . . . . .	7
2.3 Organizace v kódu . . . . .	8
2.4 Ovládání letadla . . . . .	9
2.4.1 Autopilot . . . . .	9
<b>3 Client na ovládání</b>	<b>10</b>
3.1 Organizace kódu . . . . .	11
3.2 Zpracování data z ovladače . . . . .	12
<b>4 Budoucnost projektu</b>	<b>13</b>
4.1 Další možné vlastnosti . . . . .	13
4.1.1 Upozornění na letovou zónu . . . . .	13
4.1.2 Spojení s Assault Tactical Android Kit . . . . .	13
4.1.3 Autopilot . . . . .	13
4.1.4 Stabilnější forma přenosu dat . . . . .	14
<b>Závěr</b>	<b>v</b>
<b>Seznam použité literatury</b>	<b>vi</b>
<b>Seznam obrázků</b>	<b>vii</b>

# Úvod

Cílem práce bylo sestavit bezpilotní letadlo a napsat doprovodný software – a to jak pro samotný dron, tak počítač, který letadlo ovládá. Uživatel systému by měl mít možnost sledovat video stream z letadla a letadlo dálkově ovládat. Programová část kódu by měla být psána v souladu s klasickými návrhovými vzory a umožňovat jednoduchou implementaci dalších funkcí.

Dron je postavený okolo malého počítače Raspberry Pi Zero 2, ke kterému jsou připojeny ostatní periférie přes sériovou a I2C linku. Komunikace mezi dronem a pilotem probíhá přes TCP protokol. Raspberry Pi tak funguje jako Access Point. Konfigurační soubory pro nastavení Access Pointu jsou přiložené u kódu.

Celý kód práce je dostupný na Github pod licencí MIT – <https://github.com/havrak/UAV-project>. V rámci práce na projektu byly napsány tři doprovodné knihovny – <https://github.com/havrak/Raspberry-JY901-Serial-I2C>, <https://github.com/havrak/raspberry-pi-ina226> a <https://github.com/havrak/PCA9685-rpi>. Všechny jsou volně dostupné k použití také pod licencí MIT.

# 1. Rozložení práce

Práci lze rozdělit na dvě základní komponenty. První část software běží na počítači přes který se bezpilotní letadlo ovládá, druhá na samotném dronu. Obě části jsou psané v jazyce C++.

Software pro počítač používá grafickou knihovnu Gtk3 a byl vyvíjen primárně na operačním systém Linux. Gtk3 je multiplatformní toolkit, port na další operační systémy je tak možný a kód je psán stylem, aby toto umožnil.

Jádro samotného dronu představuje malý počítač Raspberry Pi Zero 2. Vyšší výkon verze Zero 2 není nutný pro fungování, práce může fungovat na libovolném Raspberry Pi. Program využívá jen zlomek zdrojů a to jak po stránce paměti, tak výpočetního výkonu.

Obě části mezi sebou komunikují prostřednictvím protokolu založeném na rodině TCP. Přenos videa je zprostředkovávám pomocí UDP. Samotné video je kódováno do jpg, v porovnání s dalšími styly přenosu byla u jpg menší latence.

Rychlosť latence byla testována pomocí jednoduché aparatury, kde malý kód četl obrazovku, kde byl zobrazen záznam z kamery, a ovládal LED na ESP32. Systém jednoduše měřil za jak dlouho se na obrazovce objevila rozsvícená dioda (Předem vybraný pixel změnil barvu) poté co skript poslal příkaz k rozsvícení

## 1.1 Komunikační protokol

Pilotův počítač a Raspberry mezi sebou komunikují prostřednictvím jednoduchého protokolu postaveném na rodině TCP. Raspberry Pi přebírá roli serveru, to mimo jiné umožňuje, aby bylo letadlo ovládáno z více stanic.

Protokol prozatím není zašifrovaný a nepočítá také s nevalidními daty. Šifrování však není příliš nutné – dron a počítač operují na vlastní síti, kde Raspberry Pi funguje jako Wi-Fi Access-Point. Struktura protokolu ovšem nabízí možnost jednoduše implementovat symetrickou kryptografi. Ta by byla zcela postačující, její klíče by byly uloženy v konfiguračních souborech programu.

### 1.1.1 Struktura protokolu

Každý packet začíná hlavičkou o velikosti 5 bytů.

1. typ zprávy
2. priorita – Program prozatím prioritu nebere v potaz.
3. 8 horních bitů velikosti zprávy
4. 8 dolních bitů velikosti zprávy
5. dopočet do kontrolního součtu – součet čísel v hlavníčce musí být dělitelný sedmi.

Po hlavníčce následuje samotná zpráva zakončená posloupností pěti bytů – 0x00, 0x00, 0xFF, 0xFF, 0xFF. Zakončovací řetězec slouží pro případ, kdy nějaký packet nedorazí, či příchozí zpráva byla nějak poškozena. Nastane li tato situace, program postupně čte byty ze socketu a hledá tuto sekvenci. Maximální délka zprávy je půl kilobytu, samotná data jsou posílána v balíčcích o maximální velikosti 250 bytů. Omezení velikosti zprávy je zavedeno hlavně kvůli šetření paměti – program má pro každého klienta vytvořené jedno pole, do kterého se musí vejít celá zpráva.

Protokol funguje na základě posílání struktur, definovaných v `protocol_spec.h`. Pro odeslání dat je nejprve nutné nahrát data do struktury. Dále se z téhoto dat vytvoří tzv. `SendingStructure`, které se odešle do thread poolu, ten zprávu eventuálně odešle. Při přijímání se data načítají do bufferu spojeným s klientem, ze kterého se později zkopiují objektu `ProcessingStructure`. Příslušná část programu si pak data přes `memcpy` opět zkopiuruje do příslušné struktury.

Pro příjem a zpracování zpráv slouží dva thread pooly – `SendingThreadPool` a `ProcessingThreadPool`. Jednotlivé části programu tak nemusí čekat, až se uvolní socket, aby se mohla data odeslat, či čekat na jejich zpracování pro přijetí dalších. Speciální případem je zpracování příkazů, které přímo ovládají pohyb letadla. Pro ovládaní existuje samostatné vlákno, které příkazy vykonává sekvenčně. Také je implementován mechanismus, kde příliš staré příkazy se nevykonají.

Program sleduje aktivitu na socket pomocí příkazu `select`. Při běžné provozu tak většina vláken komunikace spí a zátěž není příliš velká.

Z hlediska programové stránky jsou veškeré součásti komunikace implementovány jako singletony. Program je tak řešen, aby umožnil jednoduše odesílat zprávy z různých částí programu.

### 1.1.2 Specifikace protokolu

Protokol obsluhuje tři základní části provozu dronu – nastavení, ovládaní a telemetrie. Pro nastavení je vyhrazen rozsah od 0x00 po 0x20 a zahrnuje věci, jako je odpojení klienta, nastavení kamery, či restart.

Další kategorie představuje ovládaní, pro které je vyhrazen prostor mezi 0x21 až 0x40. Projekt implementuje dva základní typy – standardní ovládaní (pohyb joysticku, atd.) a speciální. Standardní slouží pro prosté manuální ovládání dronu. Speciální například resetuje polohoměr, či zapíná autopilota.

Poslední kategorií je telemetrie, která má zbytek rozsahu do 0xFF. Tu ve standardním chování posílá Raspberry každých pár stovek milisekund. Pokud klient potřebuje aktuálnější data, může si je vyžádat odesláním zprávy o stejném typu, jako jsou požadovaná data. Speciální případ představují chybové hlášky, které vyžádat pochopitelně nejdou a pilotovi se zobrazují v dialogovém okně.

## 1.2 Konfigurace

Některé vlastnosti programu, lze specifikovat v konfiguračním souboru. Ten se nachází ve složce `$HOME/.config/uav_control/config.ini` u počítačů s operačním systémem postaveným na UNIXu. Soubor se psán stylem INI, který je lehce čitelný a vhodný na malý rozsah konfigurace. Pro zpracování je použita knihovna `inih`<sup>1</sup>.

Konfigurace je nutná jak na straně pilota, tak na straně dronu. U pilota se nastavuje například typ ovladače, či port, na kterém má být přijímán záznam z kamery. Konfigurace na straně dronu určuje například, jaká je orientace polohového senzoru, či rozložení křídel.

V aktuální verzí se konfigurační soubor nevygeneruje sám, program však má definované hodnoty, ke kterým se uchylí v jeho absenci.

---

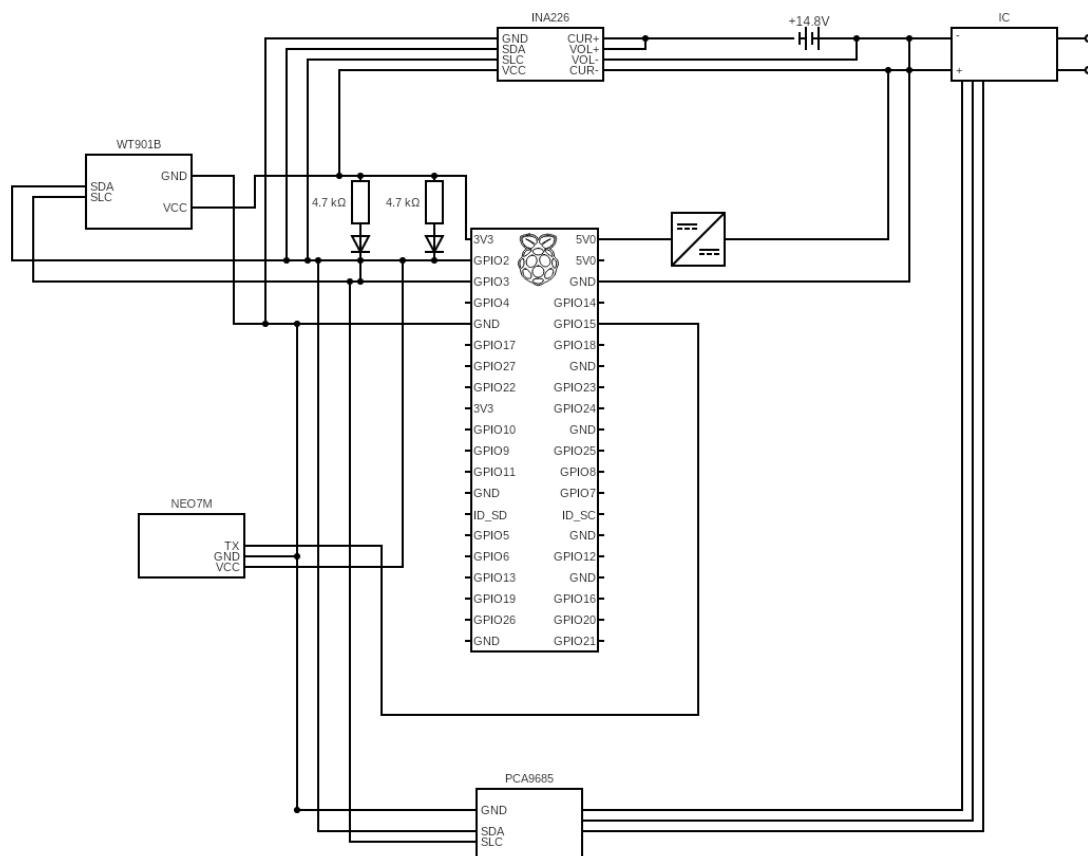
<sup>1</sup>HOYT, B. *inih*. GitHub. Dostupné z <https://github.com/benhoyt/inih>. [cit 2022-28-3]

## 2. Hardware a Raspberry Pi

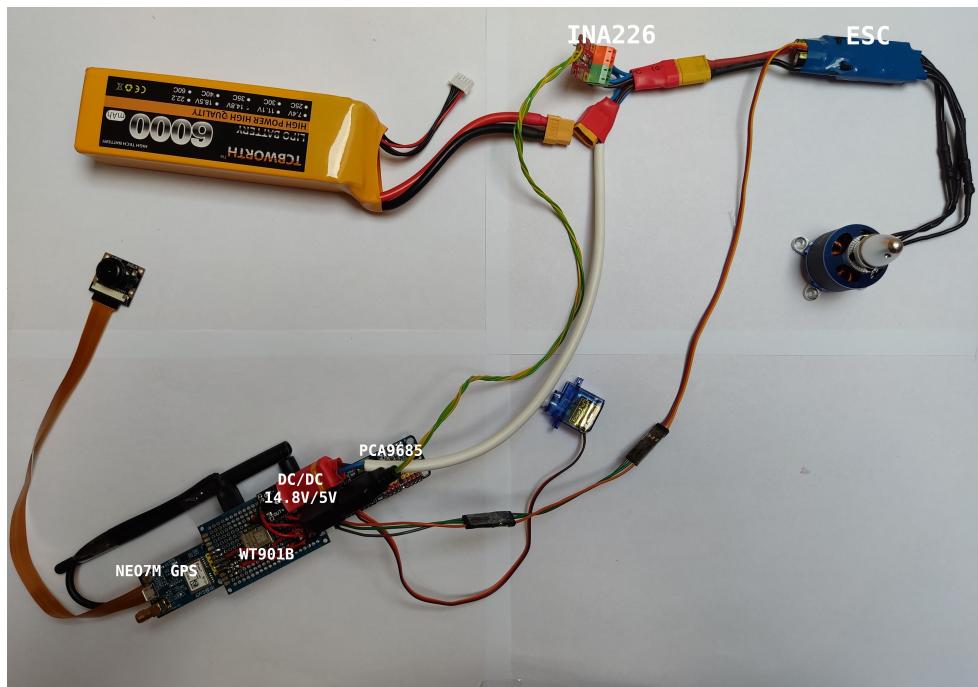
Jádro samotného letadla tvoří malý počítač Raspberry Pi Zero 2. Ostatní periférie jsou: polohový senzor WT901B, voltmetr a ampérmetr INA226, ovladač na serva PCA9865, ublox NEO-7M GPS modul a BEATLES 40A ESC (Electronic speed control). Vše s výjimkou GPS komunikuje s Raspberry Pi prostřednictvím I2C protokolu. GPS modul využívá sériovou linku.

Senzor WT901B sice podporuje přímé napojení GPS senzoru, při pokusu o konfiguraci senzoru však přestala fungovat sériová komunikace obecně.

Systém má dva přívody proudu – hlavní motor a serva jsou napájeny prostřednictvím ESC. To však nebylo schopné poskytovat dostatečný proud pro Raspberry Pi a senzory na něj napojené. Z baterie je tak vyvedená linka napojená na step down converter, který 14.8 V sniží na 5V. Tento výstup je pak připojen k 5V pinu Raspberry Pi.

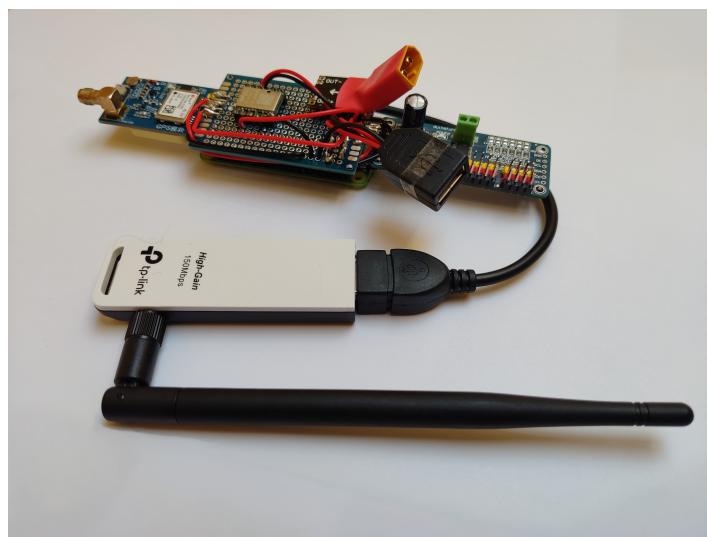


Obrázek 2.1: Schéma zapojení obvodu

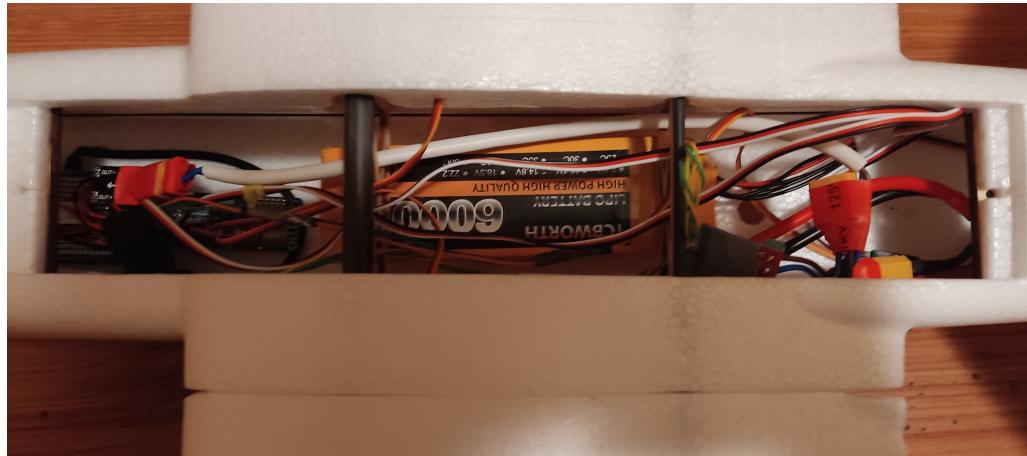


Obrázek 2.2: Reálné zapojení

Většina komponentů je umístěna na pájivé pole, které lze jednoduše nasadit na Raspberry Pi. Výjimku představuje voltmetr/ampérmetr, který je volně a v letadle se nachází v ocase. Ke zbytku se připojuje přes koncovku USB, ostatní koncovky (tj. ty pro napájení) jsou XT60.



Obrázek 2.3: Detail Raspberry Pi



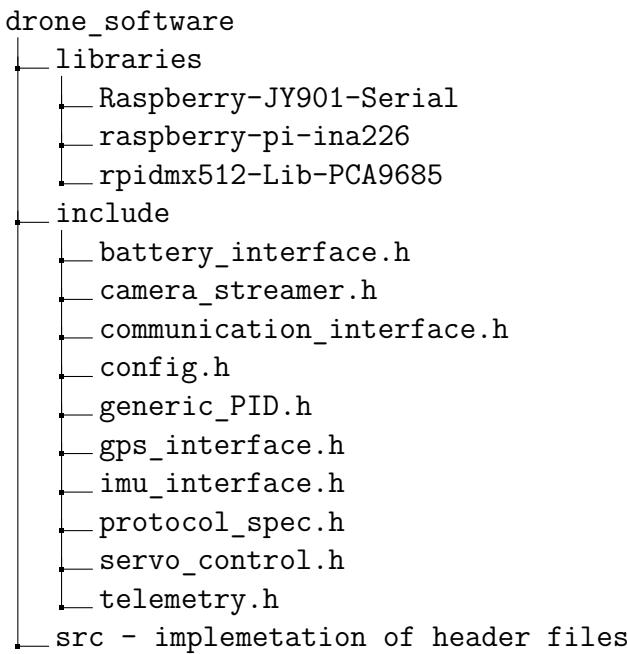
Obrázek 2.4: Detail těla dronu



Obrázek 2.5: Celý dron

Kostroum dronu je model Mini Talon od čínské společnosti X-UAV. Rozpětí křídel je 130cm délka 85cm. Jedná se o tzv. V-Tail konfiguraci, ocas tedy nemá standardní tři kontrolní plochy, ale pouze dvě. V aktuální verzi program má naimplementované ovládání pouze pro tuto konfiguraci.

## 2.1 Program



Obrázek 2.6: Struktura souborů frontend

Program na Raspberry Pi používá tři základní knihovny pro komunikaci s WT901B, INA226 a PCA9685. Přenos z kamery pak zprostředkovává OpenCV, který jako backend používá gstreamer. V budoucnu je tak možné přidat analýzu videa přes OpenCV přímo na Raspberry Pi – například detekce věcí na obrazu.

## 2.2 Interface s Hardwarem

Jak už bylo řečeno, s hardwarem Raspberry komunikuje prostřednictví I2C spojení a sériové linky. Pro zpracování dat z INA226 a WT901B byly napsány nové knihovny. Jelikož knihovny pro Raspberry Pi totiž buďto neexistovaly, nebo nebyly zdaleka kompletní.

Knihovna pro WT901B vznikla forkem knihovny pro Arduino<sup>1</sup> a byla předělána v souladu s dokumentací protokolu<sup>2</sup>. Kvůli problémům se senzorem se byla i dopsána I2C komunikace, kterou původní knihovna neimplementovala.

<sup>1</sup>TIAN, P. *Arduino-JY901-Serial*. GitHub. Dostupné z <https://github.com/paul-tian/Arduino-JY901-Serial>. [cit 2022-27-2]

<sup>2</sup>WT901 Datasheet. WitMotion Shenzen Co. Ltd. Dostupné z <https://www.wit-motion.com/gyroscope-module/Witmotion-wt901-ttl-i2c.html>. [cit 2022-27-2]

V případě senzoru INA226 bylo forknuto demo<sup>3</sup> z GitHubu a dopsáno v plnohodnotnou knihovnu.

Obě knihovny využívají knihovnu wiringPi. Ta byla koncem roku 2021 spolu s vydáním raspbian bullseye označena ze deprecated. Existuje, ale fork, kde se na vývoji dále pokračuje. Samotné wiringPi je navíc defacto pouze wrapper pro přímý přístup k smbus.

Vzhledem k přímočarosti výstupu z GPS modulu je celá implementace přímo součástí hlavního kódu. Program zpracovává GPGGA packet, který obsahuje veškerá důležitá data. Zpracování dalších packetů nebylo pro práci podstatné.

Knihovna pro ovládaní server byla lehce přepsána pro potřeby tohoto projektu. Původní verze knihovny<sup>4</sup> využívala driver bcm2835 ke komunikaci s I2C linkou. Docházelo tak ke kolizím s časováním, které vedly k narušení integrity dat ze senzorů. Byla tak přepsána za účelem, aby také používala wiringPi.

## 2.3 Organizace v kódu

Z programu se k perifériím přistupuje přes několik singletonů – ImuInterface pro WT901B, BatterInterface pro INA226, GPSInterface pro ublox NEO 7M a ServoControl pro PCA9865. Důvod řešení přes singletony je opět stejný jako u komunikace. Data ze senzorů jsou potřeba v různých částech programu a reálně reprezentují pouze jeden senzor.

Díky tomu program nepotřebuje jednu centrální třídu, která by vše organizovala. Mizí tak problémy s řadou lokálních proměnných a nutností je udržovat aktuální. Jenkož celý program je vícevláknový, implementují tak singletony ochranu před kolizí několika vláken a vzniku nevalidních dat.

Data ze senzorů, jsou agregována v třídě telemetry. Ta se stará o nastavení všech senzorů a zároveň zajišťuje vytvoření packetů pro odeslání klientovi. Ze stejných důvodů jako další třídy, i telemetry je implementována jakožto singleton.

O přístup ke kameře se stará třída CameraStreamer. Ta se vytváří dynamicky na

---

<sup>3</sup>ARIAS, M. *raspberry-pi-ina226*. GitHub. Dostupné z <https://github.com/MarioAriasGa/raspberry-pi-ina226>. [cit 2022-27-2]

<sup>4</sup>ARJAN. *Library PCA9685*. GitHub. Dostupné z <https://github.com/vanvugt/rpidmx512/tree/master/lib-pca9685>. [cit 2022-27-2]

základě požadavku od klienta. Jelikož aktuálně není implementována žádná analýza obrazu z kamery přímo na Raspberry Pi, proces stremu vznikne jako fork procesu, ve kterém je spuštěn stream.

## 2.4 Ovládání letadla

O ovládání kontrolních ploch a motoru se stará třída ServoControl. V aktuální podobě přímo zpracovává data z ovladače, která posílá server. Není tak možné jednoduše změnit jaké ovládací prvky ovladače, budou letadlo ovládat.

Na počátku zpracování nových příkazů jsou hodnoty analogové páčky přepočítány do čtvercové plochy. Standardně se totiž mapují ovladače na kruh – páčka čistě nahoru tak má výrazně vyšší hodnotu osy Y, než páčka pod úhlem 45 stupňů, to i přesto, že urazily stejnou vzdálenost. Bez této korekce ovládání působilo velmi neresponzivně, kdy praktické byly pouze polohy jih, sever, západ, východ.

V případě konfigurace ocasu do tvaru písmena V je interpretace ovládání relativně jednoduchá. Ocasní plochy (Ruddervator) ovládají pitch (klopění) a yaw (bočení). Plochy na křídlech nastavují roll (klonění) letadla, teoreticky mohou ovládat i yaw, v tomto případě to však nebylo třeba.

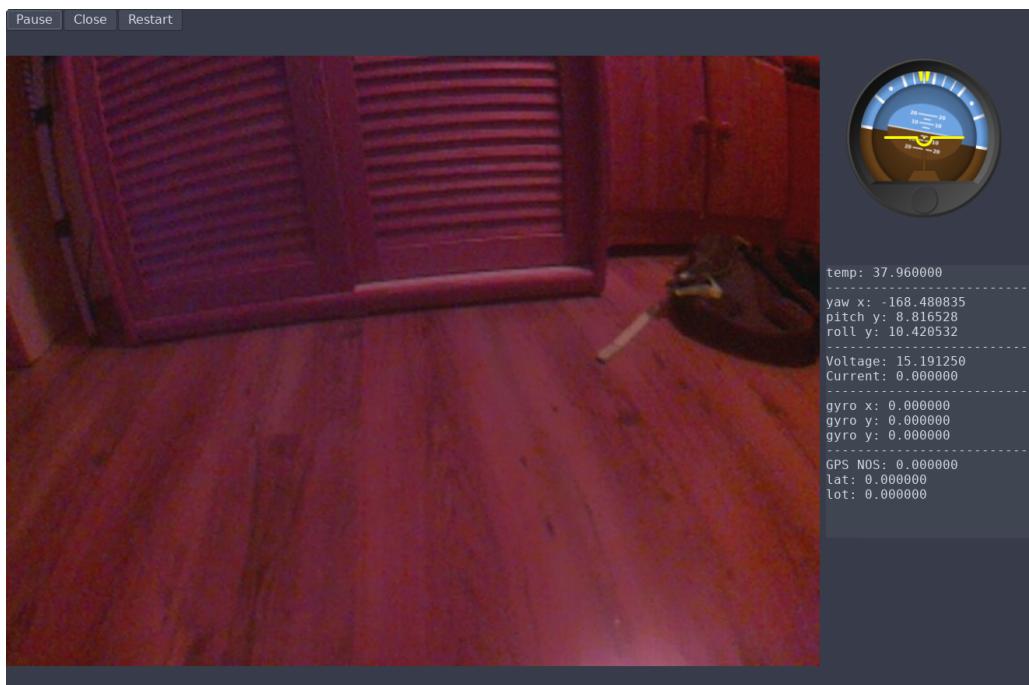
### 2.4.1 Autopilot

Projekt implementuje i jednoduchého autopilota, který je schopen držet letovou hladinu. Vyrovnávání řeší dva proporcionalní-integrační-derivační ovladače, které se snaží minimalizovat chybu. Jeden minimalizuje odchylku v klopení, druhý pak odchylku v klonění. Roll minimalizovat není třeba – pro roll musí být letadlo nakloněno.

Teoreticky by mohlo jít tento systém rozšířit aby se snažil minimalizovat odchylku nikoliv od původního směru, ale od pomyslného bodu určeného souřadnicemi GPS. Letadlo by tak nejenže letělo více rovně (aktuálně je autopilot náchylný na změnu posunutí v příčné ose), ale mohlo by u samo doletět na specifikovaný bod.

### 3. Client na ovládání

Ovládací klient pro počítač je stejně jako software pro Raspberry Pi psán v jazyce C++. K vykreslení uživatelského prostředí je použita grafická knihovna Gtk3. Konkrétně pak wrapper gtkmm, který zprostředkovává funkce Gtk z jazyka C++. Ačkoliv byl celý systém vyvíjen pro Linux je možný port na další operační systémy, jelikož Gtk je multiplatformní toolkit.



Obrázek 3.1: Grafické prostředí aplikace

V aktuální verzi je grafická stránka aplikace relativně strohá. Většinu obrazovky zabírá výstup z kamery a polevě straně se nachází informace z telemetrie. Ty jsou prozatím zprostředkovány pomocí umělého horizontu a textového výpisu.

Kvůli absenci předešlých zkušeností s Gtk toolkitem a nástrojem glade na vytváření uživatelského prostředí byla použita boilerplate aplikace<sup>1</sup>. Boilerplate mimo jiné u uka-zoval propojení s webkamerou, část logiky okolo zobrazení obrazu tak byla využita pro stream z Raspberry Pi. Systém byl však udělán více robustní a byla přidána například podpora zvětšení velikosti obrazu v závislosti na velikosti okna.

Grafika umělého horizontu je generována prostřednictvím knihovny Cairo. Ta je na

<sup>1</sup>DOLERON. *gtk3-opencv3-ux-sampling*. GitHub. Dostupné z <https://github.com/doleron/gtk3-opencv3-ux-sampling>. [cit 2022-27-2]

manipulaci s obrázky je mnohem efektivnější, než nástroje standardně nabízené v Gtk. Samotné textury pro umělý horizont byly převzaty z jiného projektu<sup>2</sup>. Jejich použití je v souladu s MIT licencí, pod kterou jsou distribuovány.

### 3.1 Organizace kódu

```
desktop_client
├── img
└── include
    ├── camera.h
    ├── communication_interface.h
    ├── config.h
    ├── control_interpreter.h
    ├── controller_interface.h
    ├── drone_telemetry.h
    ├── linux_controller_implementation.h
    ├── main_window.h
    └── protocol_spec.h
src - implementation of header files
```

Obrázek 3.2: Struktura souborů klienta

Veškerá logika grafického prostředí je koncentrována v souboru `mainwindow.h`. Knihovna Gtk není příliš vstřícná objektově orientovanému programování a efektivní granularizace kódu se dosahuje těžce. Jelikož aktuální velikost kódu není příliš velká, nepředstavuje tato limitace větší problém. Přesto je aktuální organizace dočasným řešením, zvláště pak v případě logiky týkající zpracování kamery.

Podobně jako na Raspberry Pi i počítačový klient má soubor, ve kterém je agregována veškerá telemetrie. Při získání nové telemetrie třída také požádá Gtk o zavolání metody, která přepíše data na obrazovce.

Soubory `protocol_spec.h` a `communication_interface.h` se starají o implementaci komunikačního protokolu. Ze stránky kódu jsou velmi podobné implementaci, která je na Raspberry Pi. Součástí kódu s komunikací je i třída ControllerDroneBridge, ta zprostředkovává interpretaci událostí z ovladače a posílá jeho aktuální rozložení Raspberry Pi.

---

<sup>2</sup>MAREK CEL. *QFlightinstruments*. GitHub. Dostupné z <https://github.com/marek-cel/QFlightinstruments>. [cit 2022-27-2]

## 3.2 Zpracování data z ovladače

Zpracování dat z ovladače je koncipované okolo faktu, že systém nemá standardně callbacky na události vyvolané ovladačem. Přístup přes callbacky je však relativně příčetivý, a tak je tato funkcionality implementována v programu. V souboru `linux_controller_implementation` je proto definovaný cyklus, který běží v sólo vlákně. Ten čte z file descriptoru a v případě větší změny na ovladači si data zapíše do vlastní struktury. Data ve struktuře přímo neodpovídají vstupu z ovladače – v případě os jsou posunuté o maximální hodnotu (32767). Díky tomu je zabráněno záporným hodnotám, které by mohly dělat problémy v matematice aplikované na datech.

Při zapsání nového stavu ovladače se zároveň vygeneruje událost, která se pošle všem observerům. Logika pro návrhový vzor observer je deklarovaná v třídě `ControllerInterface`. Toto rozdělení konkrétní implementace ovladače od základní deklarace prostředí pro přístup k ovladači umožňuje lehce implementovat komunikaci s ovladačem pro další operační systémy.

Veškeré observery musí implementovat abstraktní třídu `ControlInterpreter`. Aktuálně se může tento systém jevit jako zbytečně složitý. Byl však vytvořen s ohledem k budoucí funkcionality projektu. Ovladačem by totiž mělo být možné ovládat celé uživatelské prostředí – pohyb na D-Padu by například přepínal mezi různou telemetrií na obrazovce, zatímco analogový joystick by ovládal samotné letadlo.

# 4. Budoucnost projektu

Bohužel vývoj práce byl překvapivě komplikovaný a řada původní zamýšlených vlastností nebyla implementována.

Výrazně by měla být přepracována stránka vykreslování na obrazovku – grafický výstup je nyní relativně nevhledný. Grafické prostředí by také mělo mít zakomponované další indikátory, jako je směrový gyroskop či ukazatel rychlosti.

## 4.1 Další možné vlastnosti

Následující sekce uvádí další vlastnosti, které dron může implementovat a v budoucnu pravděpodobně bude.

### 4.1.1 Upozornění na letovou zónu

Dle zákona o civilním letectvý by klient také měl upozornit pilota na aktuální leteckou zónu. Bohužel oficiální stránka řízení letového provozu nemá veřejné API, existují však služby jako [airmap.com](http://airmap.com), které poskytují potřebné informace. Přidání tohoto upozornění není příliš složité, jelikož z Raspberry Pi získáváme data o poloze.

### 4.1.2 Spojení s Assault Tactical Android Kit

Více zajímavější by byla integrace na ATAK, respektive jeho verzi pro civilní použití – CivTAK. Jedná se o nástroj pro koordinaci složek v terénu vyvinutý armádou Spojených států amerických. Vzhledem k svojí povaze nabízí relativně širokou škálu možností, jak začlenit do širšího systému autonomního drona. Existuje navíc open source software FreeTAKServer, který poskytuje REST API na použití těchto funkcí. I přes poměrně dobrou dokumentaci je celý systém velmi složitý a propracované propojení by bylo velmi časově náročné.

### 4.1.3 Autopilot

V aktuální verzi implementuje projekt jednoduchého autopilota, který je schopen držet letový kurz. Neudržuje však aktuální nadmořskou výšku a kurz může být libovolně

posunut ve směru lineárně závislém s aktuálním směrem. Letadlo se tak postupem času výrazně odchýlí od svého původního směru.

Již v jádru práce bylo navrhнуто řešení pro tento problém. Teoreticky by toto řešení mohlo být rozšířeno v plnohodnotného autopilota, který by byl schopen vždy dorazit na danou pozici. S autonomním letem souvislý i možnost letu na vzdálenost větší, než co umožní komunikace. Dron by tak měl mít možnost ukládat video, či pořizovat snímky.

#### 4.1.4 Stabilnější forma přenosu dat

V aktuální verzi se Raspberry Pi chová jako Wi-Fi Access Point. Dosah systému je sice dostačující – Raspberry má 5Dbi všeobecnou anténu přijímač 25Dbi Yagi anténu, ale slabý signál na delších vzdálenostech představuje problém. Situace by se dala teoreticky dala řešit posilovačem, který by posunou sílu signálu z řádu miliwattů na wattů.

Bylo by také vhodné vyvinout anténu, která dokáže letadlo sledovat. Systém by mohl být založený na mikročipu, jako je ESP32. Ten by byl zinicializován se svojí polohou a orientací. Relativně jednoduše by pak šla dopočítat polohu, kterou má anténa zaujmout v závislosti na datech z letadla.

# Závěr

Cíl práce specifikovaný v zadání byl splněn. Letadlo je schopné letu a uživatel má k dispozici stream z kamery na dronu.

Práci však komplikovala řada problémů a to jak s hardwarem, tak se softwarem. Před začátkem jsem měl jen velmi povrchní zkušenosti s jazykem C++ a žádné s jak Gtk3, tak vývojem pro Raspberry Pi. Práce tak nemohla být realizována tak, jak byla původně zamýšlena.

Tomu napovídá i dlouhý list s budoucností projektu. Ačkoliv tomu samotné řádky kódu nemusí napovídat, práce mi dala celou řadu nových znalostí. I proto plánuji dále na projektu pokračovat ve svém volém čase. Rád bych systém dostal do stavu, kdy se jedná o plnohodnotnou realizaci bezpilotního systému.

# Seznam použité literatury

ARIAS, M. *raspberry-pi-ina226*. GitHub. Dostupné z <https://github.com/MarioAriasGa/raspberry-pi-ina226>. [cit 2022-27-2].

ARJAN. *Library PCA9685*. GitHub. Dostupné z <https://github.com/vanvugt/rpidmx512/tree/master/lib-pca9685>. [cit 2022-27-2].

DOLERON. *gtk3-opencv3-ux-sampling*. GitHub. Dostupné z <https://github.com/doleron/gtk3-opencv3-ux-sampling>. [cit 2022-27-2].

HOYT, B. *inih*. GitHub. Dostupné z <https://github.com/benhoyt/inih>. [cit 2022-28-3].

MAREK CEL. *QFlightinstruments*. GitHub. Dostupné z <https://github.com/marek-cel/QFlightinstruments>. [cit 2022-27-2].

TIAN, P. *Arduino-JY901-Serial*. GitHub. Dostupné z <https://github.com/paul-tian/Arduino-JY901-Serial>. [cit 2022-27-2].

*WT901 Datasheet*. WitMotion Shenzhen Co. Ltd. Dostupné z <https://www.wit-motion.com/gyroscope-module/Witmotion-wt901-ttl-i2c.html>. [cit 2022-27-2].

# Seznam obrázků

2.1	Schéma zapojení obvodu . . . . .	4
2.2	Reálné zapojení . . . . .	5
2.3	Detail Raspberry Pi . . . . .	5
2.4	Detail těla dronu . . . . .	6
2.5	Celý dron . . . . .	6
2.6	Struktura souborů na dronu, vlastní tvorba . . . . .	7
3.1	Grafické prostředí aplikace . . . . .	10
3.2	Struktura souborů klienta . . . . .	11