

Gymnázium, Praha 6, Arabská 14

Programování



ROČNÍKOVÝ PROJEKT

Bezpilotní letadlo

Vypracoval:

Havránek Kryštof 4E

Vedoucí práce:

ing. Daniel Kahoun

Únor 2022

Prohlašuji, že jsem jediným autorem tohoto projektu, 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 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 controller by the pilot remotely from software on a computer. Pilot should receive all the necessary telemetry and video stream from the aircraft. Software should be written in a style that allows for future development.

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

Obsah

Úvod	iv
1 Rozložení práce	1
1.1 Komunikační protokol	1
1.1.1 Struktura protokolu	1
1.1.2 Specifikace protokolu	2
2 Hardware a Raspberry Pi	4
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
3 Client na ovládání	10
3.1 Organizace kódu	11
3.2 Zpracování data z ovladače	12
4 Budoucnost projektu	13
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	14
4.1.4 Stabilnější forma přenosu dat	14
Závěr	v
Seznam obrázků	vi

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. Komunikaci mezi dronem a pilotem probíhá přes TCP protokol. Raspberry Pi tak funguje jako Access Point, konfigurační soubory pro nastavení Access Pointu jsou součástí práce.

Celý kód práce je dostupný na github pod licencí MIT – <https://github.com/havrak/UAV-project>. V rámci práce na projektu byly napsány dvě doprovodné knihovny – <https://github.com/havrak/Raspberry-JY901-Serial-I2C> a <https://github.com/havrak/raspberry-pi-ina226>. Obě 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 protitoku založené na TCP rodině. Přenos videa je zprostředkovávám pomocí UDP. Samotné video je kódováno do jpg, v porovnání s dalšími styly přenosu byla latence u jpg menší.

1.1 Komunikační protokol

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

Protokol prozatím není zašifrovaný a nepočítá s nevalidními daty. Šifrování není příliš nutné, dronu a počítač operují na vlastní síti, kde Raspberry Pi funguje jako Wi-Fi Access-Point. Struktura protokolu ovšem nabízí možnost jednoduše naimplementovat symetrickou kryptografií.

1.1.1 Struktura protokolu

Každý paket začíná hlavičkou o velikosti 5 byteů.

1. typ zprávy
2. prioritá – 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 hlavničce musí být dělitelný sedmi.

Po hlavičce 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 nevykona jí.

Program sleduje aktivitu na socket pomocí příkazu `select`. Při běžné provozu tak většina vláken komunikací 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.

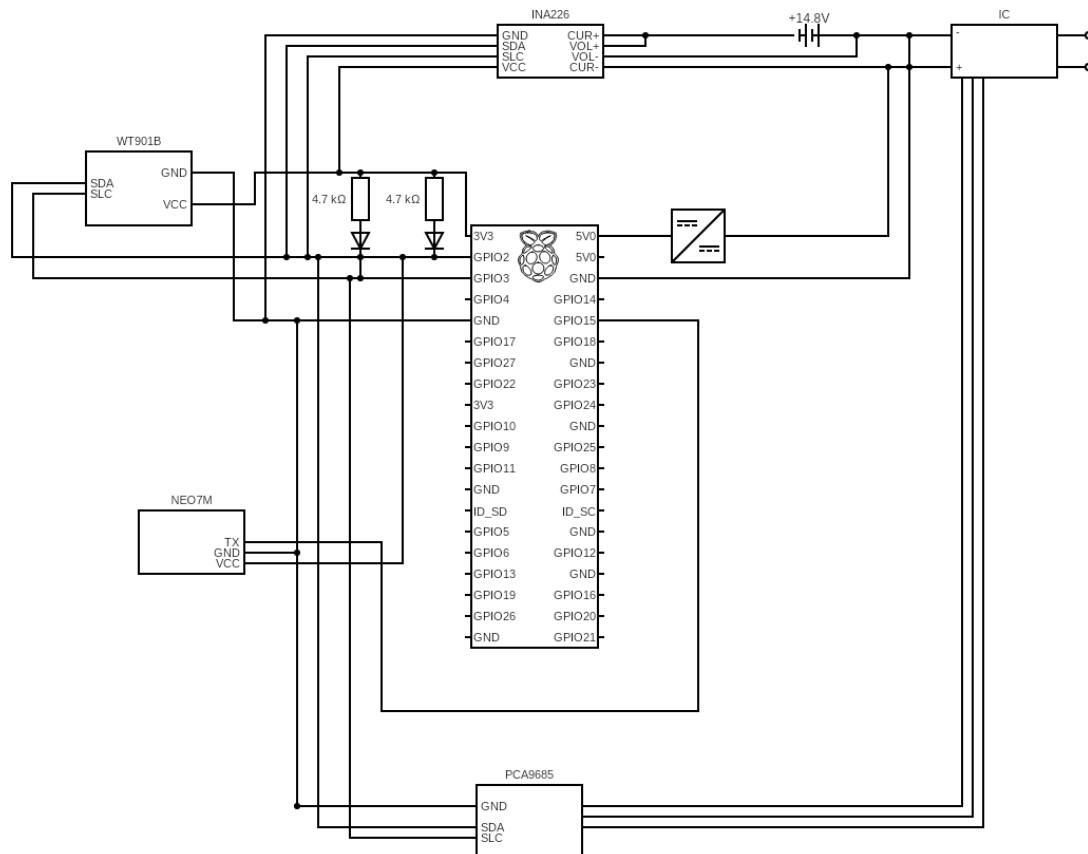
Poslední kategorií je telemetrie, které má zbytek rozsahu do 0xFF. Tu ve standardním chování posílá Raspberry každých pár stovek milisekund. Chce li client aktuálnější data může si je vyžádat odesláním zprávy o stejném typu, jako jsou požadovaná data. Speciální případ představují chybové hlášky, které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

2. Hardware a Raspberry Pi

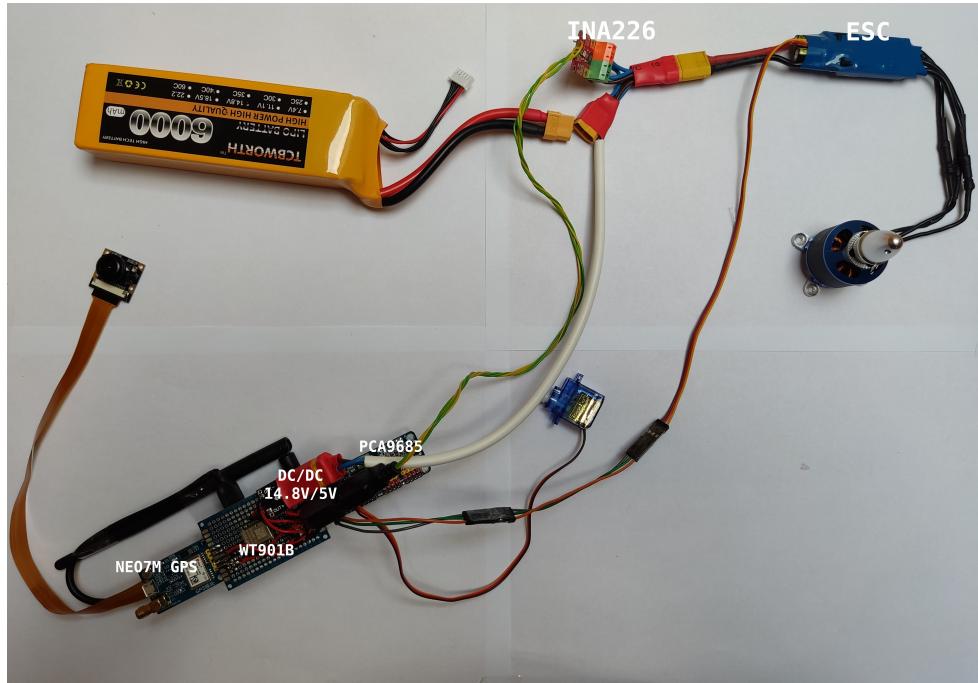
Jádro samotného letadla tvoří malý počítač Raspberry Pi Zero 2. Ostatní periférie jsou: polohový senzor WT901B, voltmetr a ampérmetr INA226, ovladač na serva PCA9865, ublox NEO-7M GPS modul a BEATLES 40A ESC (Electronic speed control). Vše s výjimkou GPS komunikuje s Raspberry Pi prostřednictvím I2C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

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

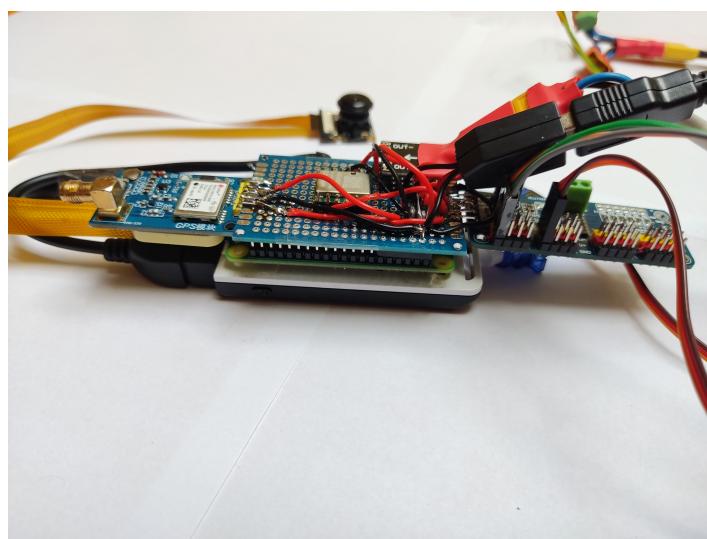
Systém má dva přívody proudu – hlavní motor a serva jsou napájeny prostřednictvím ESC. To však nebylo schopné poskytovat dostatečný proud pro Raspberry Pi a senzory na něj napojené. Z baterie je tak vyvedená linka napojená na step down converter, který 14.8 V sníží na 5V. Tento výstup je pak připojen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



Obrázek 2.1: Schéma zapojení obvodu

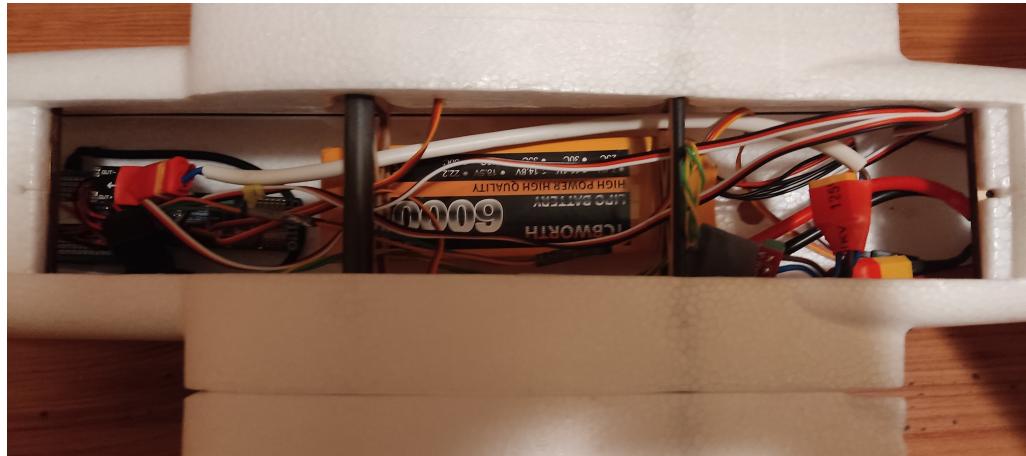
Obrázek 2.2: Reálné zapojení¹

Většina komponentů je napájena na bread board, který 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 jsou XT60. V aktuální verzi je zapojení relativně nepraktické – z ocasu do přídě vedou tři kabely.



Obrázek 2.3: Detail Raspberry Pi

¹TP-LINK WN772N je aktuálně volně vedle Raspberry Pi, nikoliv pod



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 pak 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

```

drone_software
├── libraries
│   ├── Raspberry-JY901-Serial
│   ├── raspberry-pi-ina226
│   └── rpidmx512-Lib-PCA9685
├── include
│   ├── battery_interface.h
│   ├── camera_streamer.h
│   ├── communication_interface.h
│   ├── generic_PID.h
│   ├── gps_interface.h
│   ├── imu_interface.h
│   ├── protocol_spec.h
│   ├── servo_control.h
│   └── telemetry.h
└── src - implemetation of header files

```

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 Arduino² a byla předělána v souladu s dokumentací protokolu³. Kvůli problémům se seznorem byla i dopsána I2C komunikace, kterou původní knihovna neimplementovala.

V případě knihovny pro INA226 bylo forknuto demo⁴ z GitHubu a dopsáno v plnohodnotnou knihovnu.

²

³

⁴

Obě knihovny využívají knihovnu wiringPi. Ta byla koncem roku 2021 spolu s vydáním raspbian bullseye označena ze deprecated. Také dochází ke kolizím mezi částmi kódu, které přistupují k I2C přes knihovnu WiringPi a částmi kódu, které přistupují přímo přes bcm2835. Nejedná se o kritickou chybu, ale někdy je narušena integrita některých dat.

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.

V neposlední řadě k PCA9685 se také přistupuje přes knihovnu⁵. Knihovna je v základu psána pro Raspberry Pi a implementuje všechny funkce potřebné v projektu.

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.

Funkce tak nemusí mít celou řadu parametrů a program nepotřebuje jednu centrální třídu, která bude vše organizovat. Mizí tak i problémy s řadou lokálních proměnných a nutností je udržovat aktuální. Jelikož celý program je více vláknový implementují tak singletony ochranu před kolizí několika vláken a vzniku nevalidních dat.

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

O přístup ke kamere se stará třída CameraStreamer. Ta se vytváří dynamicky na 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. Jelikož se jendá o fork při ukončení rodičovského procesu se ukončí i stream a program samotný se o přenos nemusí nijak starat.

5

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 budou jak ovládat letadlo.

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) ovládání bočení a klopení. Plochy na křídlech pak pouze klonění, teoreticky mohou ovládat i klopení, v tomto případě to však nebylo třeba.

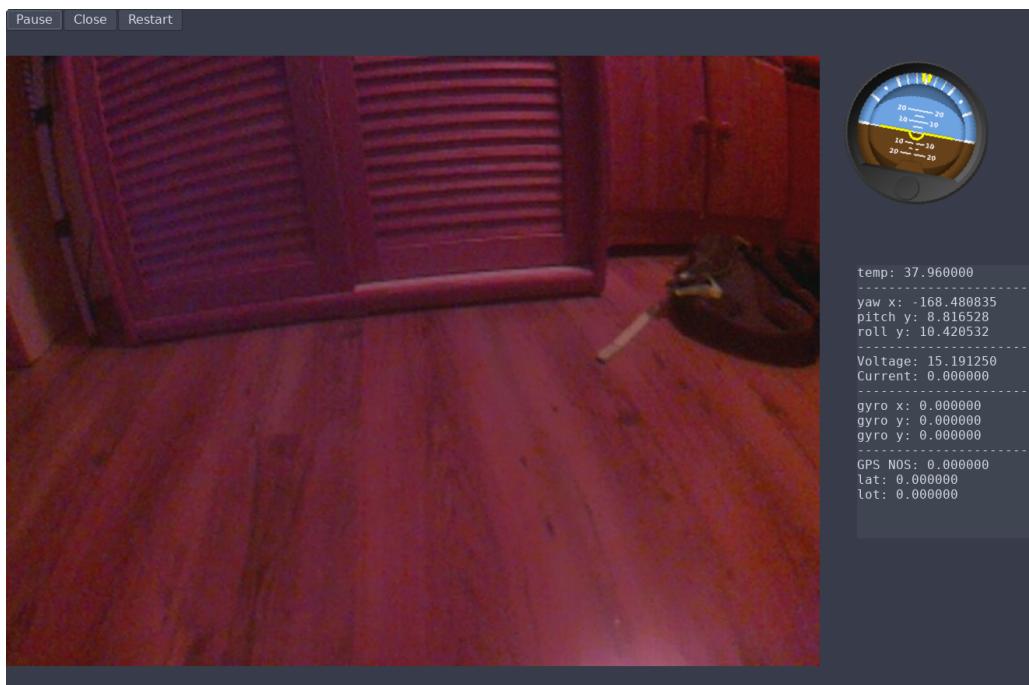
2.4.1 Autopilot

Projekt implementuje i jednoduchého autopilota, který je schopen držet letovou hladinu. Vyrovnavání řeší dva proporcionalní-integrační-derivační ovladače, které se snaží minimalizovat chybu. Jeden minimalizuje odchylku v klopení, druhý pak odchylku v klonění. Bočení minimalizovat není třeba – pro bočení 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 náchylní na změnu posunutí v příčné ose), ale mohlo 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 na Linuxu 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 po levé 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 jako boilerplate aplikace¹, která tyto vlastnosti využívala. Boilerplate mimo jiné ukazoval 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 vlastnost zvětšení velikosti obrazu v závislosti na velikosti okna.

Při vykreslování umělého horizontu je použit alogismus na zpracování obrázku.

¹

Standardně totiž PixBuf umožňuje otočení pouze o pravý úhel. Samotný metoda na rotaci byla převzata ze StackOverflow² a mírně upravena. Jedná se však dočasné řešení, jelikož práce s PixBuf je relativně nepraktická a eventuálně bude nahrazena knihovnou cairo.

Samotné textury pro umělý horizont byly převzány z jiného projektu³. Jejich použití je v souladu s MIT licencí.

3.1 Organizace kódu

```
desktop_client
├── img
└── include
    ├── communication_interface.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 `main_window.h`. Knihovna Gtk není příliš vstřícná objektově orientovanému programování a 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 u zavolání metody, která přepíše data na obrazovce.

Soubory `protocol_spec` a `communication_interface` se starají o implementaci komunikačního protokolu. Ze stránky kódu jsou velmi podobné implementaci, která je na Raspberry Pi. Byla však přidána integrace pro ControllerDroneBridge, který

²

³

zprostředkovává interpretaci událostí z ovladače a posílá aktuální rozložení na ovladači Raspberry Pi.

3.2 Zpracování data z ovladače

Zpracování data z ovladače je koncipované okolo faktu, že systém nemá standardně callbacky na události vyvolané ovladačem. Přístup přes callbacky je však relativně přívětivý 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 odpovídají výstupu z ovladače, ale v případě os jsou posunuté o maximální hodnotu (32767). Je tak

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

Veškeré observery musí implementovat abstraktní třídu `ControlInterpreter`. Aktuálně se může tento systém jevit jako zbytečně složitý. Byl však vytvořen s ohledem na budoucí funkcionality projektu. Ovladačem by mělo být možné ovládat celé uživatelské prostředí – pohyb na D-Pad by tak 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. Největší prioritu však má přepsání některých částí práce pro kompatibilitu s raspbian bullseye. K tomu patří i vyřešení kolizí při přístupu k I2C zařízením. Jako optimální řešení se nabízí přepsání knihoven pro WT901B a INA226, aby používaly nízkoúrovňový přístup přes knihovnu bcm2835.

Výrazně by měl být přepracována stránka vykreslování na obrazovku. Kvalita kódu této části je nedostačující a grafický výstup relativně nevhledný. Grafické prostředí by také mělo mít zakomponované další grafické indicátory 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 letectví 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, 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

Poněkud 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 také 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 odchylí 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 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 všesměrovou 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 watty.

Bylo by také vhodné vyvinout anténu, která dokáže letadlo sledovat. Systém by mohl být založený na platformě jako je ESP32/ESP8226, který by byl jednoduše z inicializovat se svojí polohou a orientací na jednu ze světových stran. Relativně jednoduše by tak šla dopočítat poloha, 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 specifikující budoucnost 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 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