Model kvadrokoptéry řízený mikropočítačem

Lukáš Volčík

Bakalářská práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Lukáš Volčík**
Osobní číslo:	**A17160**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Model kvadrokoptéry řízený mikropočítačem**
Téma práce anglicky:	**A Model Quadrocopter Controlled by a Microcontroller**

Zásady pro vypracování

1. Popište existující řešení modelů kvadrokoptér se zaměřením na amatérské konstrukce.
2. Zvolte komponenty pro sestavení vlastního modelu kvadrokoptéry řízené mikropočítačem.
3. Navrženou kvadrokoptéru hardwarově realizujte.
4. Vytvořte obslužný software pro použitý mikropočítač.
5. Navrhněte a realizujte ovládání kvadrokoptéry pomocí mobilního telefonu.

Forma zpracování bakalářské práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol, CA: O'Reilly, 2005, xvi, 377 p. ISBN 0596007558.
2. CINNAMON, IAN, DIY Drones for the Evil Genius: Design, Build, and Customize Your Own Drones. McGraw-Hill Education TAB, 2016, 176 s. ISBN 978-1259861468.
3. PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN – technická literatura, 2004, 159 s. ISBN 80-7300-110-1.
4. VALVANO, Jonathan W. Embedded systems: Introduction to the Arm Cortex(TM)-M3 microcontrollers. 2nd ed. s.l.: CreateSpace, 2012, xii, 462 s. ISBN 978-1477508992.
5. VÁŇA, Vladimír. ARM pro začátečníky. Praha: BEN – technická literatura, 2009, 195 s. ISBN 978-80-7300-246-6.

Vedoucí bakalářské práce: **Ing. Jan Dolinay, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **15. ledna 2021**
Termín odevzdání bakalářské práce: **17. května 2021**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚSAN

prof. Mgr. Roman Jašek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Lukáš Volčík, v. r.
podpis studenta

ABSTRAKT

Práce se zabývá tvorbou modelu kvadrokoptéry, řízeného mikropočítačem. Cílem práce bylo vytvoření obslužného softwaru pro zvolený mikropočítač, tvorba mobilní aplikace a tvorba obslužného softwaru pro Wi-Fi modul, který nahrazuje běžně používané dálkové ovládání. Hlavním úkolem bylo vytvořit levné a nenáročné řešení, které lze dále modifikovat pro využití v konkrétních aplikacích. Navržený systém je ovládán pomocí mobilního telefonu, který komunikuje s Wi-Fi přijímačem pomocí WebSocket technologie. Vytvořený stabilizátor letu přijímá data z Wi-Fi modulu a stabilizuje funkci motorů pro dosažení požadovaného náklonu. V teoretické části je rozebráno řešení vlastního modelu kvadrokoptéry a jsou vysvětleny výhody a nevýhody řešení použitého pro tuto práci. V praktické části je detailněji popsán způsob řešení, aby se dal napodobit i pro jinak zvolené komponenty.

Klíčová slova: kvadrokoptéra, model, mikropočítač, Wi-Fi, mobilní telefon, WebSocket, stabilizátor letu

ABSTRACT

The thesis deals with the creation of a quadcopter model, controlled by microcomputer. The aim of the work was to create service software for the Wi-Fi module, which replaces commonly used remote controllers. The main goal was to create a cheap and easy solution that can be further modified for use in specific applications. The designed system is controlled by a mobile phone that communicates with the Wi-Fi receiver using WebSocket technology. The created flight controller receives data from the Wi-Fi module and it stabilizes the work of motors to achieve the required tilt. The theoretical part discussed the solution for the self-made quadcopter model and explained the advantages and disadvantages of this thesis's solutions. In the practical part, the solution is described in more detail so it could be imitated with differently chosen components.

Keywords: quadcopter, model, microcomputer, Wi-Fi, mobile phone, WebSocket, flight controller

Hlavní poděkování patří panu Ing. Janu Dolinayovi, Ph.D. za veškerou pomoc při tvorbě bakalářské práce a za samotné vzbuzení zájmu o probíranou problematiku v průběhu studia.

Na závěr bych také chtěl poděkovat svojí rodině, blízkým přátelům a svojí přítelkyni za nesmírnou podporu při studiu a po čas tvorby této práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Kvadrokoptéra se stává pořád oblíbenější v naší populaci, nejen jako koníček na trávení volného času, ale také má spoustu dalších praktických využití díky své výjimečné obratnosti ve vzdušném prostoru s kombinací s dalšími moderními technologiemi [16]. Průmysl s drony je považován ostatními průmysly za rušivý element kvůli tomu, kolik věcí změnil. Byl schopen vylepšit logistiku zejména v oblasti doručování zásilek, poskytnul fotografům a umělcům nový způsob pro zachycení úchvatných záběrů, a dokonce i v archeologii se stal nesmírně užitečným nástrojem což se prokázalo v roce 2018, kdy archeologové za pomoci dronů objevili kolem padesáti geoglyfů vyobrazených na planině Nazca [1].

Tato užitečná technologie tak neláká zájem pouze pro použití v průmyslu, ale také vyvolala zájem u mnoha běžných lidí, kteří by mohli nalézt další způsoby individuálního využití této technologie. Nejenže se dá model kvadrokoptéry pořídit v dnešní době za přijatelnou cenu, ale dokonce vzniklo několik volně dostupných řešení pro vyladění vlastního modelu kvadrokoptéry. Existující řešení však mají určitá omezení na hardware, který lze použít.

Při tvorbě vlastního algoritmu se však tato omezení dají obejít pouhým přidáním modulu pro měření náklonu. S velice minimálními požadavky tak lze využít široké škály mikropočítačů, které jsou schopny fungovat jako letový stabilizátor.

Tato Bakalářská práce by chtěla nabídnout šablonu pro tvorbu vlastních letových stabilizátorů z běžně dostupných generických mikropočítačů, které u existujících softwarových řešení nelze použít. Také využívá mobilního telefonu jako ovládacího zařízení a vyhýbá se tak potřebě kupovat modelářské dálkové ovladače a přijímače. Místo přijímače je využit Wi-Fi modul, který simuluje funkci přijímače přes protokol Wi-Fi. Celý projekt se tak dá sestrojit cenově podstatně efektivněji při použití vlastního kódu a při správném výběru komponent.

V teoretické části rozebírám historii kvadrokoptér a jednotlivé výhody a nevýhody možných řešení. V praktické části detailněji popisuji postup práce pro možné napodobení postupu pomocí odlišných komponent.

I. TEORETICKÁ ČÁST

1 HISTORIE

Pro pochopení moderního trendu kvadrokoptér je potřeba zjistit něco o jejich historii a jakým způsobem došlo k nárustu jejich popularity.

1.1 Definice

Nejdříve je potřeba zmínit, že slovo „Dron“ je označení pro jakékoliv bezpilotní vzdušné vozidlo (někdy také UAV z anglického názvu Unmanned Aerial Vehicle). Pojem „Kvadrokoptéra“ je tedy označení pro dron, který je poháněn čtyřmi motory a případně se může předpona „Kvadro“ nahradit podle počtu motorů konkrétního dronu [4].

1.2 Začátky

Podle definice dronu byl první spatřen v roce 1849, když rakouští vojáci útočili na Benátky pomocí bezpilotních horkovzdušných balónů naplněných výbušninami. Myšlenka dronu tak začala být zajímavá hlavně ve vojenském průmyslu. Útok však nebyl příliš úspěšný, jelikož se část balónů obrátila na vlastní jednotky, a tak se technologie vojenských dronů nevyužívala až do doby vynalezení prvního letadla v roce 1900. Po pouhých 16 letech od prvního letadla vytvořila Velká Británie první bezpilotní letoun [3].

Obrázek 1 Útok na Benátky [3]

1.3 Vojenské využití

První bezpilotní letadlo vytvořené Velkou Británií bylo prakticky létající bomba, která byla navržena podle designu Nicola Tesly a využívala dálkového ovládání, které bylo podobné jako u dnešních dronů (avšak v mnohem hrubším provedení). Mělo být určeno na sestřelování německých Zepelínů, ale po několika neúspěšných pokusech britská armáda začala věřit, že mají drony velice omezené využití v armádě. O rok později byla vytvořena americká alternativa a po působivém testu před americkou armádou byla vytvořena pokročilá verze určena k masové produkci jménem „the Kettering Bug“. Byl to div tehdejší technologie, ale v roce 1918 už bylo pozdě na jeho využití v první světové válce. UAV technologie se rapidně vylepšila obzvlášť po druhé světové válce a vznikly tak modernější vojenské drony. Poté v roce 1982, kdy Izrael zahájila útok na Sýrii pomocí UAV, začal se brát vyšší důraz na nebezpečnost vojenských dronů a celosvětově vzrostl zájem o jejich vlastnictví. Rok 1982 se bere jako začátek moderní války dronů [3].

Obrázek 2 SR-71 Blackbird (1960) [3]

Na rozdíl od prvního dronu vytvořeného Velkou Británií, moderní vojenské drony slouží hlavně ke dvěma účelům. První je, že pilot ovládá UAV pomocí dálkového ovládání a sbírá informace o nepřáteli. Druhý způsob využívá mini dronu, o velikosti podobné dnešním komerčním modelům, k taktickému průzkumu. Tento mini dron většinou letí na autopilot do předem určené pozice, kde pořídí snímky a vrátí se do základny. Je sice pravda, že se drony

dají využít i k přímému útoku na konkrétní cíle, ale takovou technologii se snaží udržet armáda v tajnosti ze zřejmých důvodů [3].

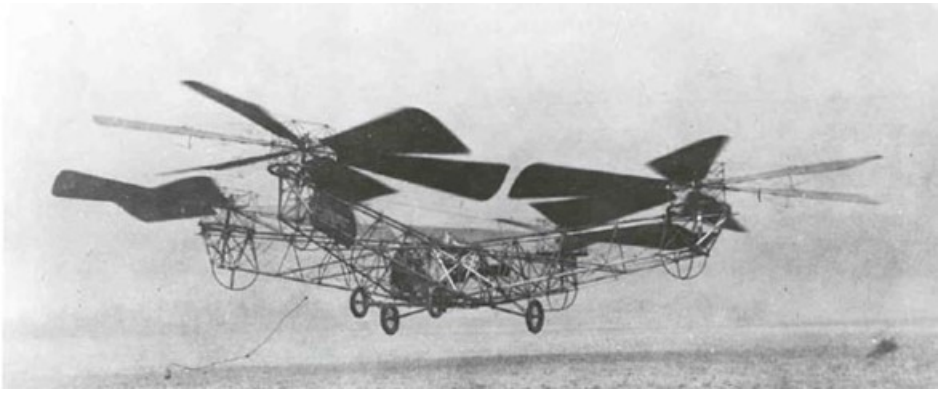
Obrázek 3 MQ-1 Predator (1995) [3]

Obrázek 4 Obrázek 4 RQ-11 Raven mini dron [5]

1.4 První kvadrokoptéry

V roce 1920 byla vytvořena Etiennem Omnichenem první kvadrokoptéra jménem „Omnichen 2“, která za svůj život dokončila úspěšně 1000 letů a dokázala doletět do rekordní vzdálenosti 360 m. Další fungující kvadrokoptéru sestrojil De Bothezat v roce 1923, kterou

se následně inspirovala společnost Curtis Wright a v roce 1958 vytvořila model „The Curtis Wright V27“ [4].



Obrázek 5 De Bothezatova helikoptéra (1923) [4]

1.5 Komerční využití

Rok 2006 byl prvním rokem, kdy FAA vydala povolení ke komerčnímu využití dronu a vládní organizace je začala využívat na pomoc při katastrofách, hlídání hranic a hašení požárů. V letech 2005–2010 také pokrok v elektronice umožnil vývoj lehkých a levných letových stabilizátorů, akcelerometrů, GPS a kamer. Díky tomu začal být design kvadrokoptéry populární hlavně v menším provedení, jak ho známe dnes. Až když v roce 2013 výkonný ředitel Amazonu prohlásil, že společnost zvažuje využití dronů jako způsob doručení, veřejnost začala mít velký zájem o tuto technologii. V roce 2015 FAA vydala 1000 povolení na komerční použití dronů a v roce 2016 toto číslo stouplo na 3100. Od té doby se zájem stále zvyšuje [3]. Také Pizzerie Domino’s si na Novém Zélandu v roce 2016 získala pozornost prvním využitím dronů na rozvoz pizzy pomocí speciálně navrženého dronu pro převoz krabice s pizzou [1].

2 ZPŮSOBY ŘEŠENÍ

V této kapitole probereme nejpoužívanější řešení pro tvorbu letového stabilizátoru u vlastního modelu kvadrokoptéry. Stabilizátor obsahuje mikropočítač, který se chová jako mozek systému, který přijímá data z gyroskopu, analyzuje je a posílá upravené hodnoty do motorů během milisekund. K analýze hodnot a jejich úpravě využívá PID algoritmus, který je schopen vytvořit robustní systém, který reaguje na okolní vlivy [6].

2.1 Existující Open Source řešení

Velice oblíbeným a efektivním řešením pro tvorbu vlastního modelu kvadrokoptéry je využití existujícího softwaru pro letový stabilizátor. Aktuálně jsou nejčastěji používané Ardupilot, Betaflight a CleanFlight, ale existuje mnoho dalších alternativ. V takovém případě je potřeba zvolit jednu z podporovaných desek pro vybraný software. Další komponenty lze zvolit libovolně podle aplikace projektu a lze sestrojit vlastní amatérskou konstrukci vhodnou i pro vlastní případy užití pro které zatím neexistují komerční řešení. Podporované desky obsahují veškeré potřebné komponenty pro balancování zařízení za pomoci IMU. Také mohou obsahovat doplňky pro kamery a jiné populární modifikace navíc. Mohou však být podstatně dražší a méně dostupné oproti často používaným mikropočítačům jako je například Arduino, STM32F1 a jiné často používané desky pro řešení široké škály problémů [21].

2.2 Tvorba vlastního algoritmu

Díky rostoucímu zájmu o embedded hardware je v dnešní době podstatně snadnější pro programátory, inženýry ale také fanoušky elektroniky začít s tvorbou vlastních embedded systémů a lze vybrat z mnoha druhů mikropočítačů vhodných pro různé aplikace podle našich potřeb [17]. Proto je tvorba vlastního algoritmu pro letový stabilizátor také velice populární možností a může být přínosná obzvlášť pro studenty informatiky a elektrotechniky.

Psát vlastní kód sice zabere práci navíc, ale umožní to větší kontrolu nad celým projektem, a to zejména při výběru komponent. Lze tedy zvolit libovolný mikropočítač, který bude schopen komunikovat s libovolně zvoleným modulem gyroskopu a také přijímat data z bezdrátového modulu (RC přijímač, Wi-Fi modul apod.).

2.3 Volba komponent

Vhodná volba komponent je silně individuální pro konkrétní aplikaci projektu. Platí však, že 8bitové mikropočítače (např. Arduino Uno) nejsou vhodné na letový stabilizátor kvůli

omezené výpočetní síle. Jsou sice schopny fungovat, ale řadí se do nižší třídy, protože kvalita vyrovnávání je ovlivněna rychlostí čtení hodnot z gyroskopu a také rychlostí počítání PID hodnot.

2.3.1 Jádro letového stabilizátoru

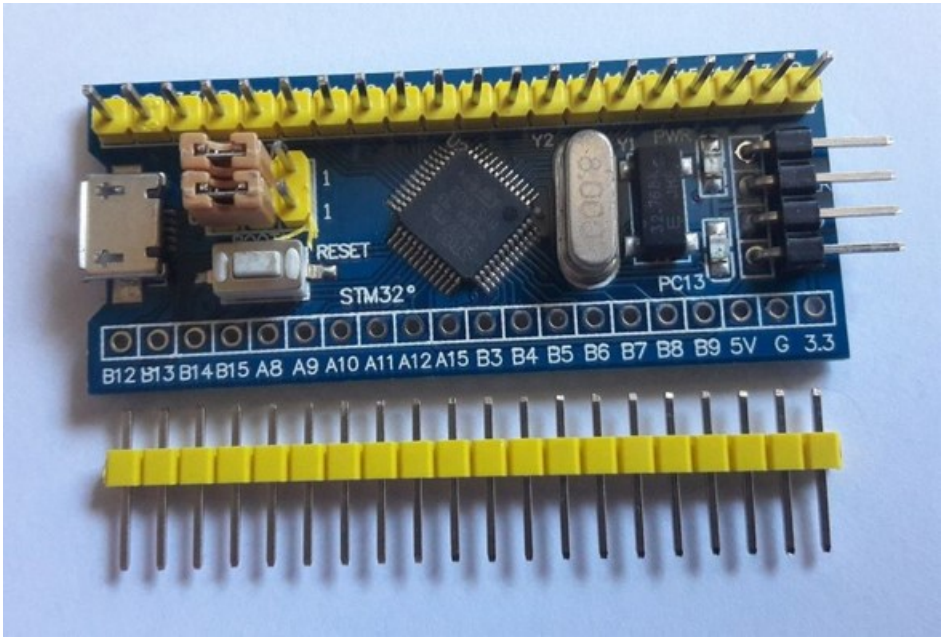
Letový stabilizátor přijímá data z gyroskopu přes I^2C linku a za pomoci PID algoritmu upravuje rychlosti jednotlivých motorů pro dosáhnutí požadovaného náklonu.

Efektivním řešením je volba vhodného 32bitového mikropočítače, který splňuje všechny podmínky pro letový stabilizátor. Tyto podmínky jsou:

- Alespoň 4 PWM piny pro posílání signálů do motorů
- Schopnost komunikovat po sériové lince s bezdrátovým modulem
- Umožňuje I^2C komunikaci s modulem gyroskopu

V případě této bakalářské práce byl jako jádro letového stabilizátoru zvolen mikropočítač STM32F103 „Bluepill“. Jedná se o 32bitový mikropočítač s procesorem ARM32 CortexTM-M3 s rychlostí 72MHz, který se dá pořídit za podobnou cenu jako klon Arduino Nano a také má podobné rozměry. Nabízí však mnohem větší výpočetní výkon díky vyšší rychlosti a většímu počtu bitů se kterými zvládne operovat v jednom cyklu. Oproti 8bitovému systému má velikou výhodu při počítání s delšími čísly, protože dokáže sečíst 32bitové číslo během jedné operace, zatímco 8bitový mikropočítač musí úlohu rozdělit nejméně na 4 operace. To se může projevit i na délce zkompilovaného algoritmu díky menšímu počtu operací [2], [15]. Také má víc pinů. Jako nevýhodu však má, že je potřeba do něj nahrávat program pomocí speciálního programátoru ST-Link V2 a abychom mohli využít jeho micro USB port k nahrávání programu, musíme vymazat jeho bootloader a nahrát do něj bootloader pro

STM32duino. Také je potřeba nainstalovat STM32duino v manažéru desek v Arduino IDE [10].

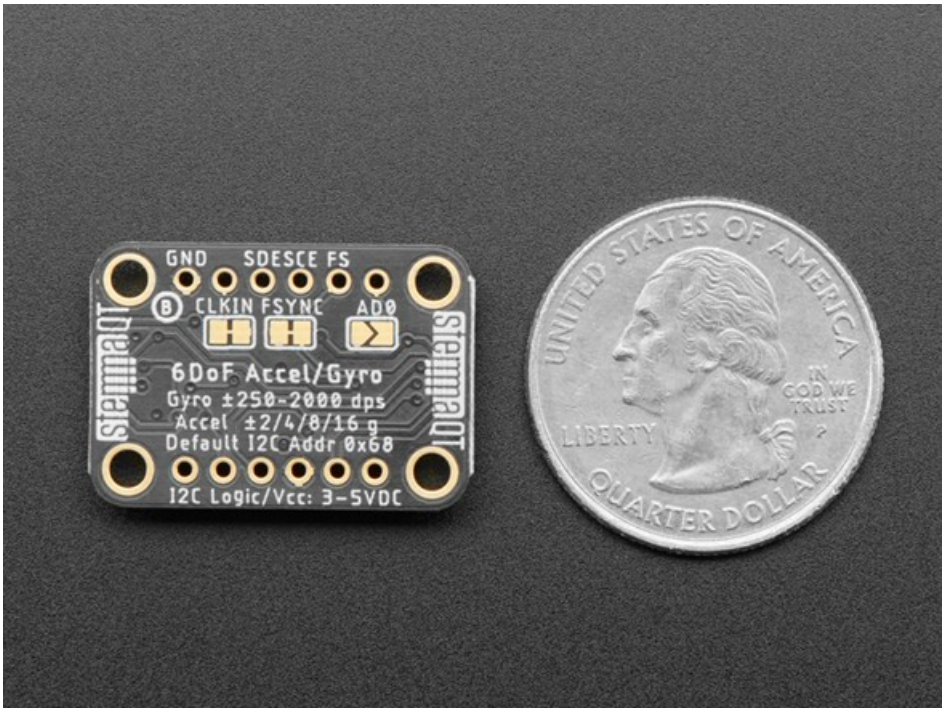


Obrázek 6 STM32 „Bluepill“ [10]

2.3.2 Inerciální Měřící Jednotka

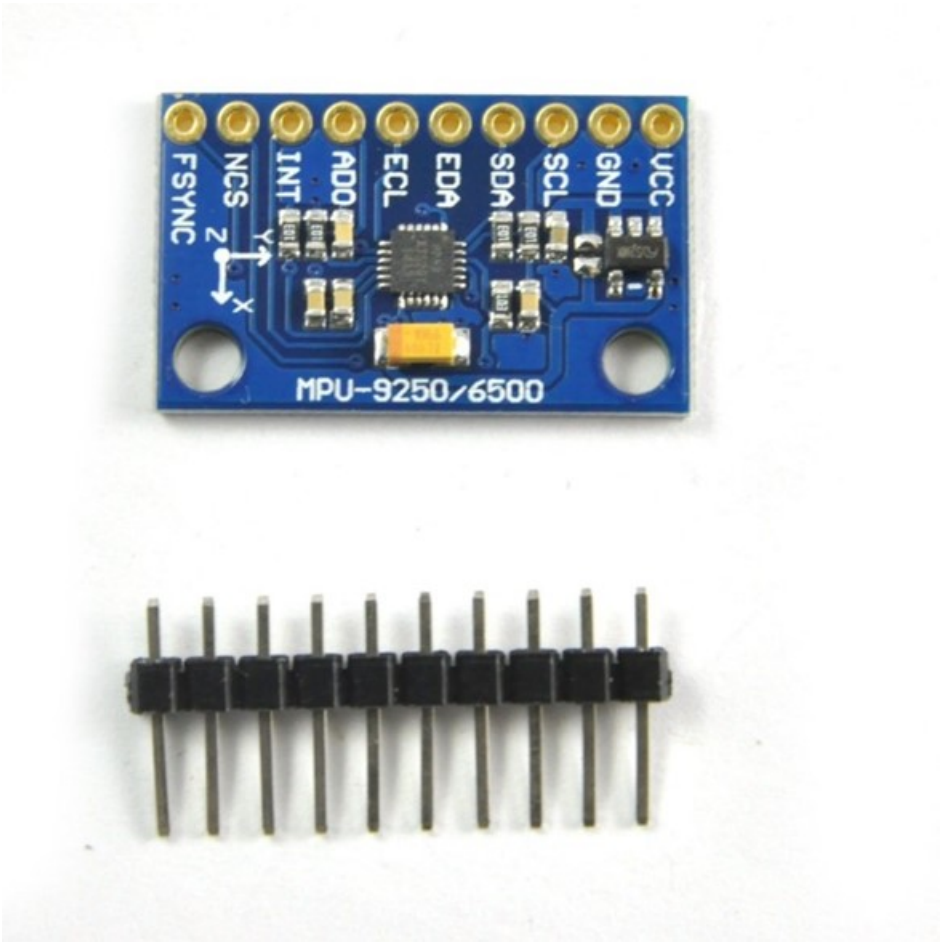
Inerciální Měřící Jednotka je hlavním potřebným modulem pro tvorbu vlastního letového stabilizátoru. Bez ní by kvadrokoptéra nebyla schopna letu. Jako Inerciální Měřící Jednotka (IMU z anglického „Inertial Measurment Unit“) se běžně označuje kombinace akcelerometru a gyroskopu [18]. Pokud by neprobíhalo upravování rychlosti motorů v závislosti k náklonu, tak by se jakákoliv nerovnost okamžitě exponenciálně zvětšila a vedla k nekontrolovatelnému pádu [16].

Jedním z nejpopulárnějších IMU je MPU 6050. Není to příliš dávno, kdy jednotka IMU dosahovala rozměrů mikrovlnné trouby a její cena se pohybovala okolo 50 000 dolarů [18]. Dnes MPU 6050 může mít mnoho provedení, ale nejčastěji prodávané generické modely se pohybují v cenové kategorii okolo 50 korun a většinou nepřesahují velikost kovové mince. Nízká cena a vysoká spolehlivost jsou nepochybně nejsilnější faktory určující její oblíbenost.



Obrázek 7 Alternativní verze MPU 6050 [18]

Jako IMU pro tuto práci byl vybrán 9osý akcelerometr/gyroskop/magnetometr MPU 9250, který je založen na svém předchůdci MPU 6050. Díky funkci magnetometru je však schopen určit přesnější orientaci v 3D prostoru v kombinaci s funkcemi akcelerometru a gyroskopu [11].



Obrázek 8 Modul MPU-9250 [14]

2.4 PID algoritmus

PID (Proporcionální, Integrální, Derivační) algoritmus pracuje tak, že kontroluje výstupní proměnnou, aby zpracovávaná proměnná dosáhla požadované hodnoty, kterou chceme [9]. Tento algoritmus se dá použít pro velmi širokou škálu projektů, které vyžadují kontrolovaný přírůstek hodnoty (např. výkon topení pro dosažení určité teploty, síla motorů kvadrokoptéry pro dosažení náklonu atd.), který může být narušen okolními vlivy.

2.4.1 Složka P

Proporcionální složka je vypočtena násobením příbytku P (kP) s chybou Err (Rozdíl mezi aktuální hodnotou a požadovanou hodnotou) [9]. Jedná se tak o hlavní složku, která především určuje rychlost růstu proměnné.

P = kP * Err

2.4.2 Složka I

Integrální složka je vypočtena opět násobením příbytku I (kI) s chybou (Err) a následně ještě s časovou délkou jednoho cyklu našeho algoritmu (dt). To má za následek, že se hodnota I postupně akumuluje a čím déle nám trvá dosáhnout požadované hodnoty, tím větší má hodnota I vliv na naši proměnnou [9].

I = kI * Err * dt

2.4.3 Složka D

Derivační složka násobí příbytek D (kD) s hodnotou, která určuje rychlost stoupání. Tato složka se snaží předurčit kam ovlivněná proměnná míří a působí v protisměru oproti složkám P a I. Tím se snaží zamezit přestřelení žádané hodnoty (např. kvůli kinetické energii) a zmenšuje rychlost stoupání, pokud se blížíme k cíli [9]. Zpravidla zpomaluje rychlost vyrovnávání, ale zvyšuje spolehlivost, že nepřestřelíme požadovanou hodnotu.

D = kD * (Err – prevErr) / dt

2.4.4 Výsledná hodnota

Výsledek získáme jednoduše sečtením všech hodnot dohromady [9]. Přidáním konstant do našeho algoritmu můžeme doladit vliv jednotlivých složek na výslednou hodnotu, aby hodnota odpovídala našim potřebám.

PIDhodnota = P + I + D

3 BEZDRÁTOVÉ OVLÁDÁNÍ

Pro ovládání UAV je potřeba bezdrátové ovládání. Existují sice způsoby jak předprogramovat letovou dráhu dronu, ale nejjednodušší způsob, jak začít je pořídit si dálkové ovládání operující pomocí rádiových vln (velikosti vlnových délek se mohou lišit podle modelu ovladače). Tyto rádiové dálkové ovládání zpravidla obsahují vysílač a přijímač [16]. V dnešní době však existuje mnoho vyvinutých bezdrátových technologií a lze jako alternativní řešení využít protokol Wi-Fi nebo dokonce i Bluetooth, LoRa a dalších existujících technologií. Každá tato technologie poskytuje určité výhody a nevýhody a záleží na naší aplikaci. Například technologie LoRa je schopna vysílat a přijímat signál do vzdálenosti větší až 21 km a její maximální rychlostní limit 256Kb za sekundu je dostačující pro ovládání dronu. Díky tomu je technologie LoRa populární moderní alternativou pro aplikace vyžadující ovládání na veliké vzdálenosti [20].

3.1 Standardní dálkové ovládání

Nejčastěji jsou používány dálkové ovladače využívající rádiové frekvence 2.4 GHz a 5.8 GHz. Využívají tak nejčastěji frekvenci 5.8 GHz na ovládání dronu a frekvence 2.4 GHz na přenos FPV videa. Staré modely mohou využívat frekvence 433 MHz. Lze také vytvořit vlastní dálkové ovládání pomocí těchto frekvencí [19].

Standardní ovladače mají spoustu funkcí nejen pro modely kvadrokoptér, ale také pro jiné typy UAV modelů. Od toho se také odvíjí jejich cena.

Obrázek 9 Standardní RC vysílač [19]

3.2 Ovládání pomocí Wi-Fi

Toto řešení využívá Wi-Fi modulu pro komunikaci s mobilním telefonem. Pro rámec této práce postačí mobilní aplikace obsahující slider pro určení síly motorů a joystick pro určení požadovaného náklonu. Je však možno vylepšit uživatelské prostředí a přidat libovolné množství ovládacích prvků což je v případě standardních ovladačů omezeno počtem mechanických páček a tlačítek.

Toto řešení se tak vyhýbá se potřebě kupovat standardní modelářské ovladače a přijímače, které se v případě kvalitnějších zařízení mohou pohybovat v řádech několika tisíc korun. Mobilní telefon tedy nahrazuje vysílač a Wi-Fi modul se chová jako přijímač, který přeposílá data přijatá z mobilního telefonu do stabilizátoru letu.

3.2.1 WebSocket technologie

Za pomoci Websocket technologie jsme schopni vytvořit dostatečně rychlou komunikaci pro bezpečné ovládání s nízkou latencí a můžeme tak efektivně využít mobilní telefon, anebo

dokonce i notebook či počítač s bezdrátovou kartou. Lze také využít desktopový počítač při použití USB adaptéru pro bezdrátovou komunikaci.

Požadavkem je pouze aplikace, která se chová jako WebSocket klient a odesílá správná data do našeho Wi-Fi modulu na kterém běží WebSocket server.

II. PRAKTICKÁ ČÁST

4 NÁVRH HARDWARU

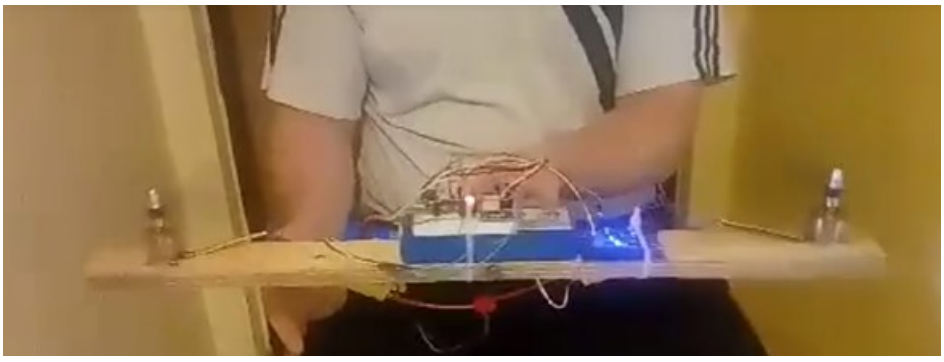
4.1 Sestavení konstrukce

Konstrukci lze zakoupit nebo vyrobit vlastní. V mém případě jsem se rozhodl vyrobit vlastní, aby byl celý projekt takzvaně „from scratch“. V takovém případě je nutno vhodně zvolit materiál konstrukce. Nejvhodnější jsou pravděpodobně karbonové konstrukce díky jejich vysoké odolnosti a výjimečně nízké hmotnosti. Je však možné vyrobit například i dřevěnou konstrukci, pokud to naše aplikace umožňuje a vyladíme správně faktor hmotnosti a odolnosti. V mém případě jsem tedy zvolil dřevěnou konstrukci pro oba prototypy, především kvůli možnosti jednoduché úpravy či vylepšení v budoucnosti.

Pro lepší parametry konstrukce by se dal design finální konstrukce zhotovit z karbonových tyčí o stejných rozměrech jako použité dřevěné tyče (nebo o rozměrech menších díky vyšší odolnosti).

4.1.1 Testovací prototyp

Nejdříve bylo potřeba sestrojit prototyp, který využívá dva motory a balancuje se pouze po jedné ose. Na takovém prototypu byla úspěšně ozkoušena komunikace mezi Wi-Fi modulem, letovým stabilizátorem a klientskou aplikací. Byly na něm také vyladěny konstanty pro PID algoritmus pro efektivní dosažení správného náklonu.



Obrázek 10 Testovací prototyp za běhu [zdroj vlastní]

4.1.2 Finální verze

Pro konečnou verzi kvadrokoptéry bylo potřeba navrhnout konstrukci pro všechny čtyři motory, která bude mít co nejnižší hmotnost a přijatelnou odolnost. První konstrukce byla vyrobena z vlastnoručně vyřezaných a vybroušených tyčí. Kvůli nespokojenosti s drobnými

nerovnostmi a výslednou tloušťkou tyčí menší než 1 cm po dobroušení, byly zakoupeny tyče přesně o tloušťce 1x1cm a z nich byla zhotovena finální konstrukce na kterou byl nainstalován veškerý hardware.

Obrázek 11 První konstrukce bez hardware [zdroj vlastní]

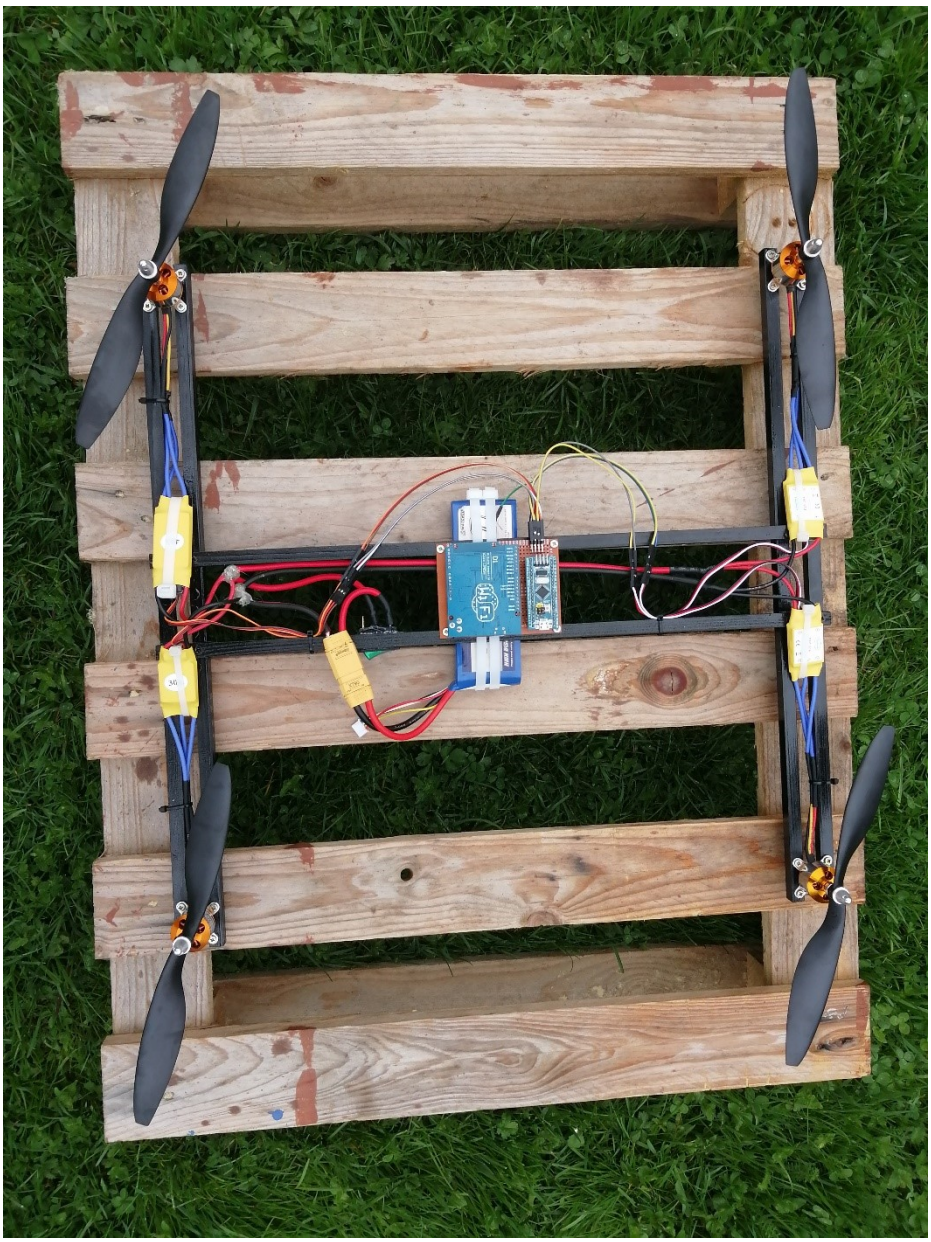
Před instalací veškerého hardwaru byla konstrukce nabarvena na černo. Po připevnění jednotlivých komponent byly na pevno připájeny kontakty ESC s baterií a jednotlivými motory. Všechny kontakty byly dále izolovány tepelně smrštitelnými bužírkami. Výsledný model kvadrokoptéry má hmotnost 1070 gramů, včetně baterie a vrtulí.

Nejdříve byla konstrukce připevněna plastovou utahovací páskou k dřevěné paletě. To mělo zajistit bezpečnost práce a zamezit zničení modelu v případě, kdyby bylo potřeba ještě doladit PID konstanty. Po úspěšném ověření schopnosti stabilizovaní náklonu byla páska odstraněna.

Po prvním vzlétnutí se však projevila vada jednoho z ESC, které vyhořelo a způsobilo pád celé kvadrokoptéry. Po nahrazení vyhořelé součástky však bylo zjištěno, že model ESC zakoupeného na českém trhu není kompatibilní s ESC stejného typu zakoupeného z čínského trhu. Jednotlivé modely ESC vytváří odlišné rychlosti motoru při stejné délce přijímaného pulzu. Jediným reálným řešením je tak zakoupit všechny ESC od stejného prodejce. I přes pokus o softwarové řešení tohoto problému je PID algoritmus vysoce nestabilní při použití

jiného modelu ESC a je vysoká pravděpodobnost výskytu chyby ve vyrovnávání, kterou následuje pád celého modelu.

Existující řešení softwaru pro letový stabilizátor jako například Ardupilot se tento problém také pokouší vyřešit softwarově, ale ani tento spolehlivý software nedokáže kompletně zamezit nežádoucí chování způsobené odlišným typem ESC a lze předpokládat chybu ve stabilizaci náklonu.

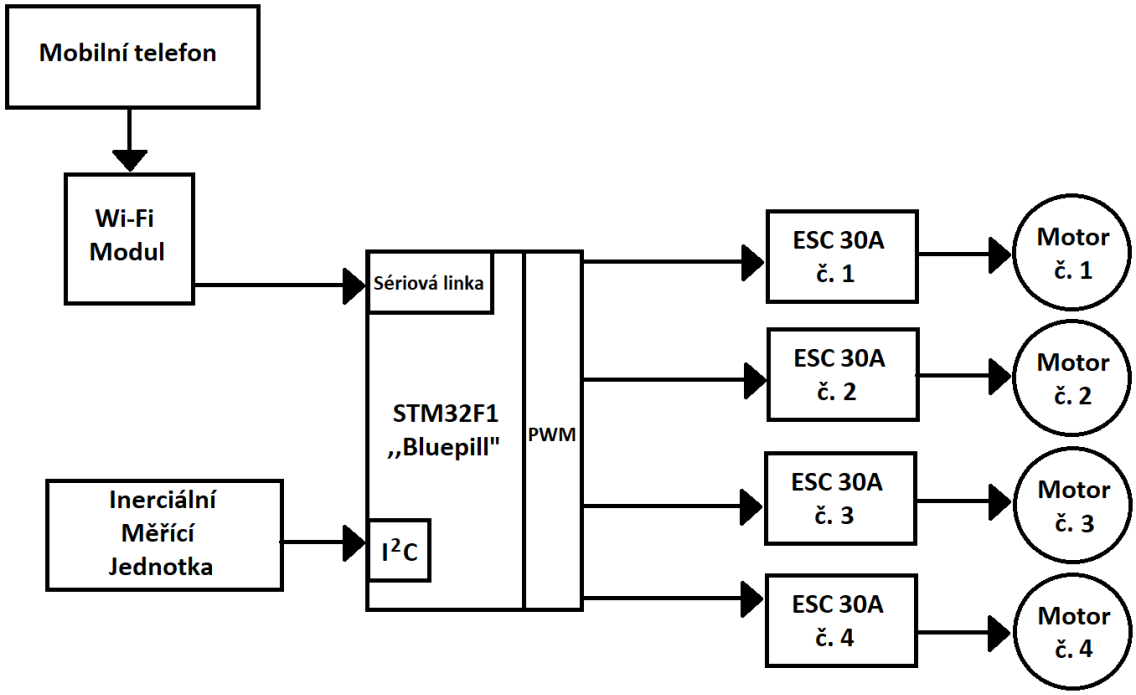


Obrázek 12 Finální model kvadrokoptéry [zdroj vlastní]

4.2 Zapojení celého systému

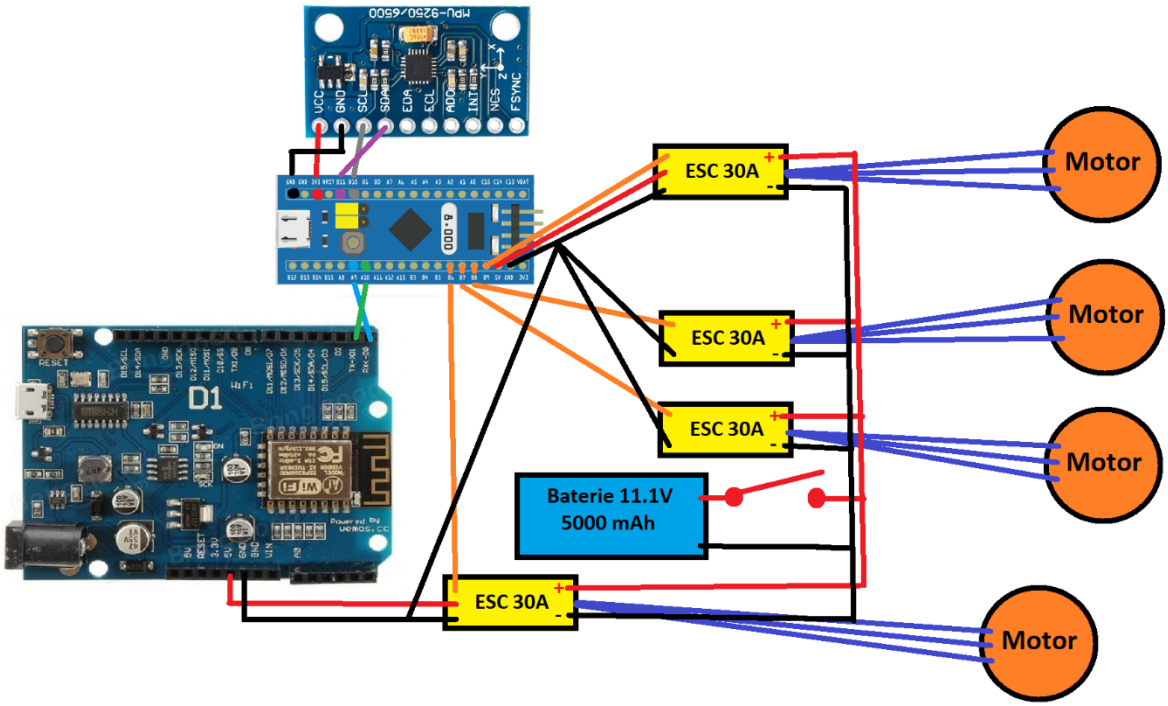
4.2.1 Schéma zapojení celého systému

Pro ujasnění zapojení celého systému bylo vytvořeno blokové schéma zobrazující jednotlivé komponenty.



Obrázek 13 Blokové schéma systému [zdroj vlastní]

Před tvorbou na pevno připájených kontaktů bylo potřeba ještě ujasnit zapojení jednotlivých kontaktů. Pro tento účel bylo vytvořeno barevné schéma zapojení celého systému.

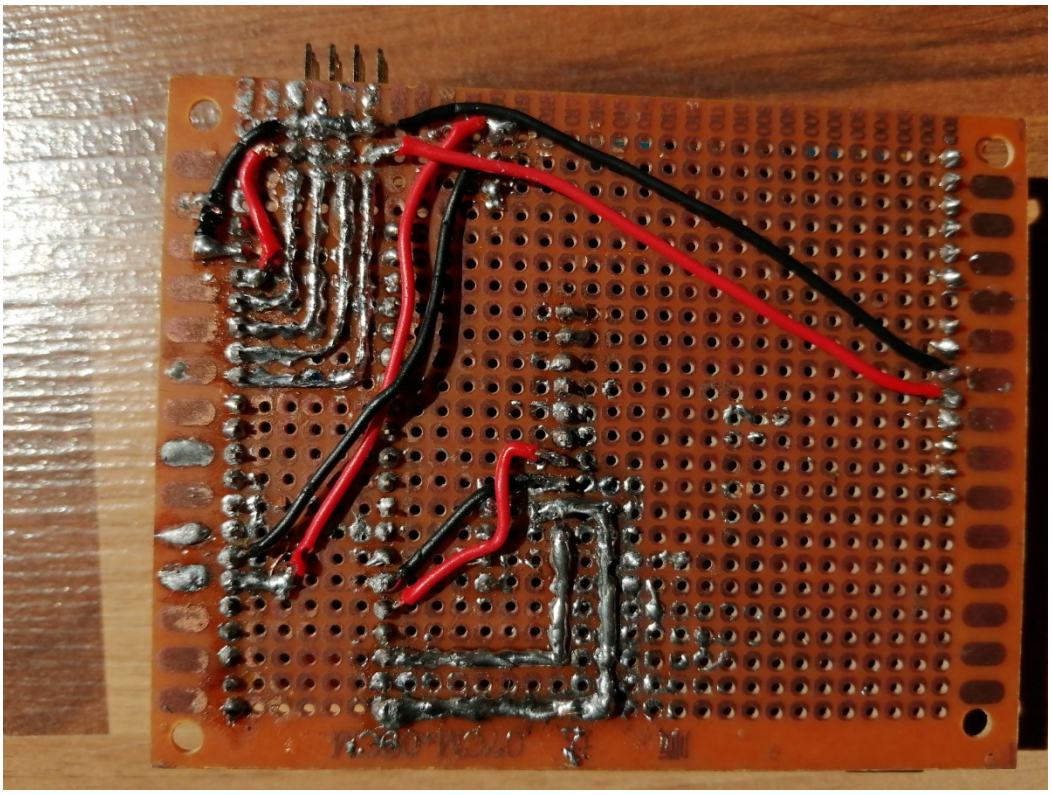


Obrázek 14 Schéma zapojení celého systému [zdroj vlastní]

4.2.2 Tvorba vlastní PCB desky

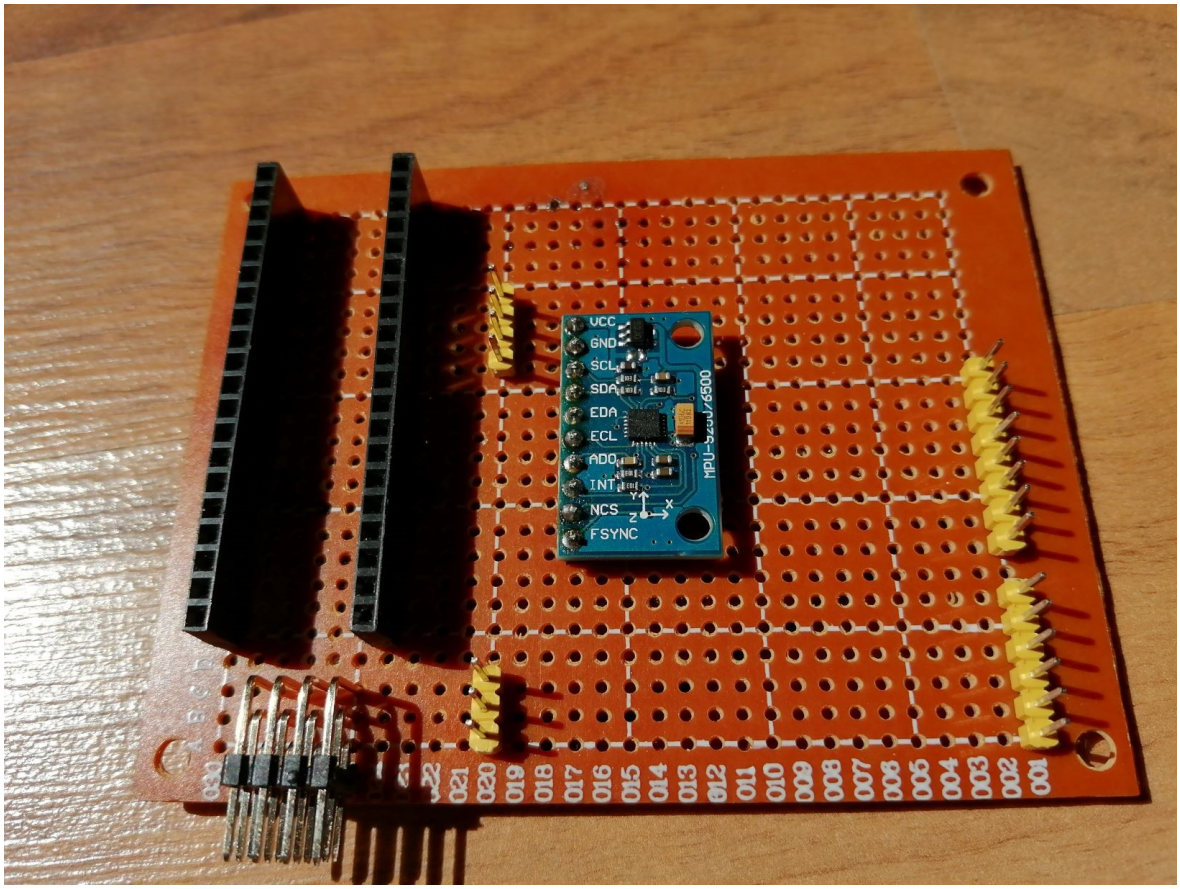
Pro zmenšení hmotnosti a praktickou manipulaci byla vytvořena vlastní PCB deska. Díky ní jsou všechny výpočetní jednotky přehledně propojeny. Také umožňuje v případě potřeby

odpojit všechny ESC přes odpojitelné piny či prohazovat jejich pozice pro správnou funkci algoritmu.



Obrázek 15 Připájené kontakty PCB desky [zdroj vlastní]

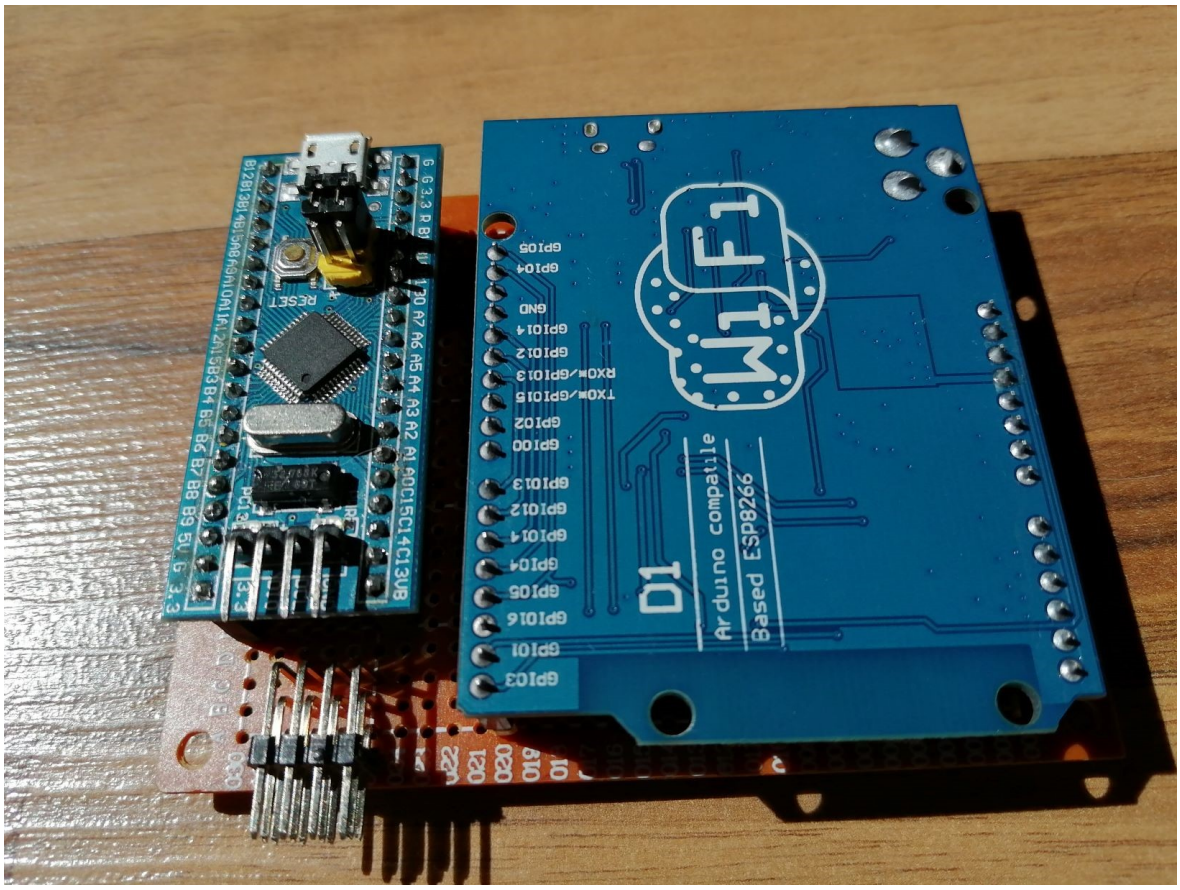
Pouze MPU-9250 je připájeno na pevno, aby nedocházelo ke zbytečným otřesům navíc či nerovností způsobenou zakřivením pinů. STM32F1 má svůj vlastní socket, stejně jako deska Wemos D1 R1.



Obrázek 16 Sockety pro mikropočítače [zdroj vlastní]

V případě poškození tak lze pouze poškozenou jednotku nahradit za novou nebo lze komponenty využít na jiný projekt a potom vše jednoduše vrátit zpět. Pod deskou STM32F1 si také můžeme všimnout čtyř trojitých pravoúhlých pinů, které jsou určeny na zapojení všech potřebných kontaktů s ESC, pomocí kterých letový stabilizátor řídí rychlost motorů. ESC

také dodávají proud do všech modulů skrz prostřední piny. Levý pin slouží k napájení STM32 s MPU9250 a pravý prostřední pin slouží k napájení desky Wemos D1 R1.



Obrázek 17 Letový stabilizátor připraven k použití [zdroj vlastní]

5 NÁVRH SOFTWARU

V této části podrobně popisujeme důležité části kódu u všech programů, které bylo nutno pro náš systém vytvořit. Tedy pro letový stabilizátor, Wi-Fi modul a mobilní aplikaci.

5.1 Wi-Fi modul

Pro naprogramování Wi-Fi modulu postačí vytvoření Websocketu a při zprávě přes WebSocket, přepošle zprávu po sériové lince do stabilizátoru letu řízeného STM32F103.

5.1.1 Setup

Ve funkci Setup je klíčové začít sériovou komunikaci a vytvořit softAP ke kterému se připojí naše mobilní zařízení. Po vytvoření WebSocket serveru na portu 81 je také potřeba přiřadit „webSocketEvent“ jako funkci, která se zavolá při každé WebSocket události.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <WebSocketsServer.h>

/* Přihlašovací údaje. */
const char *ssid = "Server Test";
const char *password = "thereisnotest";

ESP8266WebServer server(80);
WebSocketsServer websocket(81);

void setup() {
  delay(1000);
  Serial.begin(500000);
  Serial.setTimeout(10);

  WiFi.softAP(ssid, password);

  server.on("/", handleRoot);
  server.on("/on", pinOn);
  server.on("/off", pinOff);
  server.on("/post", handleArg);

  server.begin();
  websocket.begin();

  websocket.onEvent(webSocketEvent);


  pinMode(14, OUTPUT);
}
```

Obrázek 18 Inicializace WebSocketu [zdroj vlastní]

5.1.2 WebSocketEvent

Ve funkci webSocketEvent se zpracovávají příchozí data podle typu události. Důležitá je část označená jako „Příchod textových dat od klienta“, ve které veškerá přijatá textová data přeposíláme po sériové lince do stabilizátoru letu.

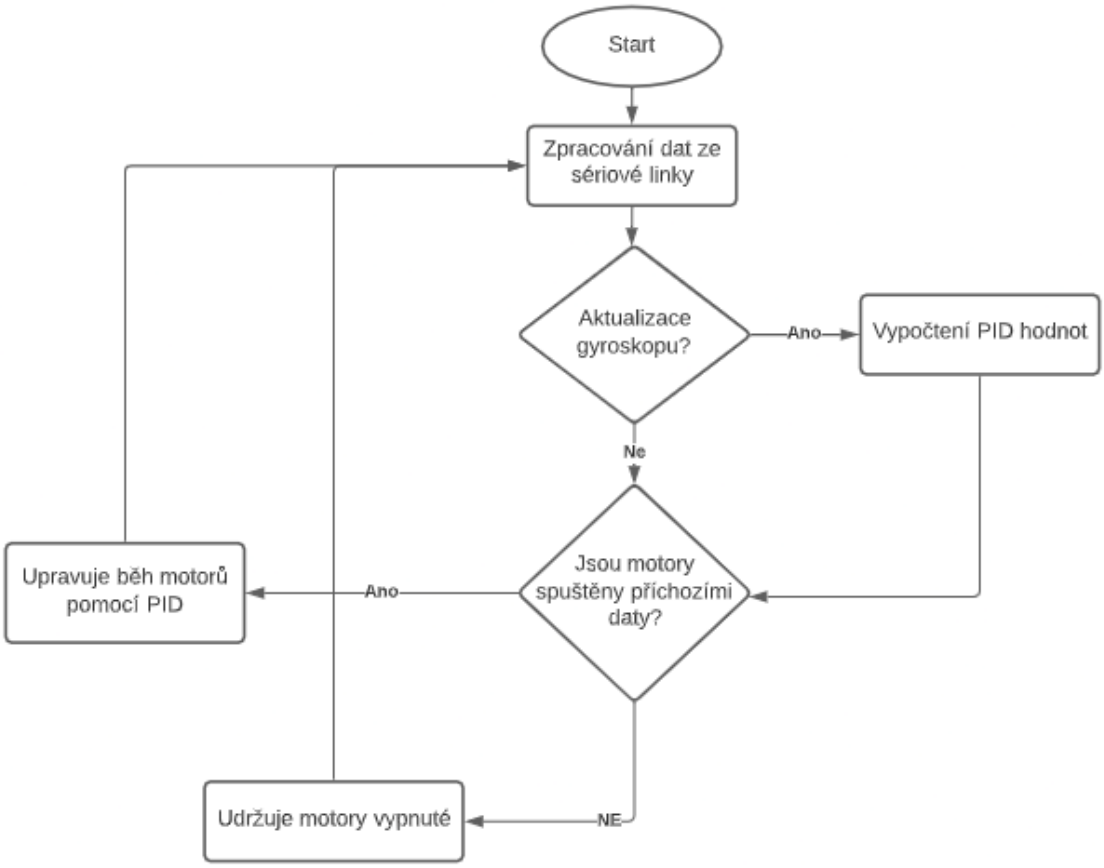
```
void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {
    switch(type) {
        case WStype_DISCONNECTED:
            Serial.printf("[%u] Disconnected!\n", num);
            break;
        case WStype_CONNECTED:
            {
                IPAddress ip = websocket.remoteIP(num);
                Serial.printf("[%u] Connected from %d.%d.%d.%d url: %s\n", num, ip[0], ip[1], ip[2], ip[3], payload);

        // send message to client
        websocket.sendTXT(num, "Connected");
            }
            break;
        case WStype_TEXT: /* PŘÍCHOD TEXTOVÝCH DAT OD KLIENTA */
            Serial.print((char *)payload);
            Serial.print("\r");

            break;
        case WStype_BIN:
            Serial.printf("[%u] get binary length: %u\n", num, length);
            hexdump(payload, length);
            break;
    }
}
```

Obrázek 19 Funkce WebSocketEvent [zdroj vlastní]

5.2 Stabilizátor letu

5.2.1 Vývojový diagram stabilizátoru

Stručně zobrazuje průběh hlavního cyklu stabilizátoru. Chod jednotlivých funkcí je dále detailněji rozebrán.



Obrázek 20 Vývojový diagram stabilizátoru letu [zdroj vlastní]

5.2.2 Setup

Stabilizátor využívá knihovny Servo.h pro psaní PWM signálu na ovladače motorů, MPU9250.h na čtení z modulu IMU [11] a knihovny string.h pro operace s řetězci.

Funkce Setup se při spuštění vykoná pouze jednou a probíhají v ní jednotlivé inicializace:

- ESC piny se stanoveným maximálním a minimálním rozsahem pulzu.
- Sériová komunikaci s rychlostí 500 000 baud a zmenšeným timeoutem pro rychlejší odezvu.
- Inicializace modulu IMU.

```
void setup() {

    topLeftEsc.attach(PB6,1000,2000);
    topRightEsc.attach(PB7,1000,2000);
    botRightEsc.attach(PB8,1000,2000);
    botLeftEsc.attach(PB9,1000,2000);


    Serial.begin(500000); // USB serial
    Serial1.begin(500000); // WiFi serial
    Serial1.setTimeout(10);
    Serial.setTimeout(10);
    while(!Serial1){};
    Wire.begin();
    delay(2000);

    // Nastavení Accel/Gyro/Mag lze provádět až po inicializaci na správné adrese modulu
    if (!mpu.setup(0x68)) {
        while (1) {
            Serial.println("MPU connection failed. Please check your connection with `connection_check` example.");
            delay(5000);
        }
    }

    // Kalibrační hodnoty
    mpu.setAccBias(46.79, 10.24, 8.79);
    mpu.setGyroBias(-0.84, 0.37, -1.27);
    mpu.setMagBias(-346.09, -1.83, -875.03);


    mpu.setMagneticDeclination(5.183333);

}
```

Obrázek 21 Funkce Setup() [zdroj vlastní]

5.2.3 Loop

Ve funkci Loop probíhá hlavní cyklus, který se opakuje. Princip fungování této funkce byl stručně popsán ve vývojovém diagramu.

Nejdříve kontrolujeme příchozí data na sériové lince. Dále, pokud proběhl update v MPU9250, tak voláme funkci PIDController, která nám vypočítá správné PID hodnoty pro úpravu jednotlivých motorů. V poslední fázi aplikujeme vypočítané PID hodnoty na

jednotlivé motory jen v případě, že uživatel zapnul motory pomocí bezdrátového ovládání (když motory neběží, neaplikuje se PID kvůli možnému nekontrolovatelnému chování).

```
void loop() {

    processSerialData();

    if (mpu.update()) {
        static uint32_t prev_ms = millis();
        if (millis() > prev_ms + 25) {

            PIDController(); // Každý update z gyroskopu se vypočitají PID hodnoty pro synchronizované balancování
            //print_roll_pitch();
            prev_ms = millis();
        }
    }

    if(throttle > 1030) // Bez běhu motorů nezapíná PID
      ApplyPID();
    else {
      topLeftEsc.writeMicroseconds(1000);
      topRightEsc.writeMicroseconds(1000);
      botLeftEsc.writeMicroseconds(1000);
      botRightEsc.writeMicroseconds(1000);
    }
}
```

Obrázek 22 Funkce loop [zdroj vlastní]

5.2.4 PIDController

Ve funkci PIDController() probíhá výpočet PID hodnot pro obě osy. Funkce CalculatePitch a CalculateRoll jsou téměř totožné, každá pouze pracuje s vlastními proměnnými. Nejdříve tedy vypočítáme rozdíl mezi požadovaným náklonem a aktuálním náklonem. Dále vypočítáme jednotlivé hodnoty P, I a D, které nakonec sečteme pro dosažení finálního výsledku.

K výsledným proměnným se dá přistupovat z kterékoliv části kódu díky tomu, že byly definovány globálně.

```
void PIDController(){
  CalculatePitch();
  CalculateRoll();
}

void CalculatePitch(){
  float pitchRead = mpu.getPitch();
  pitch_PID_error = set_pitch - pitchRead; // Vypočítá chybu mezi aktuální hodnotou a požadovanou hodnotou

  //Výpočet P
  pitch_PID_p = 0.01*kp * pitch_PID_error;

  //Výpočet I
  pitch_PID_i = 0.01*pitch_PID_i + (ki * pitch_PID_error);

  //Pro D potřebujeme reálnou informaci o uplynutí času
  pitch_timePrev = pitch_Time;                          // předchozí čas je uložen před přepsáním aktuálního času
  pitch_Time = millis();
  pitch_elapsedTime = (pitch_Time - pitch_timePrev) / 1000;
  //Výpočet D
  pitch_PID_d = 0.01*kd*((pitch_PID_error - pitch_previous_error)/pitch_elapsedTime);

  //Konečný výsledek je vypočten sečtením všech hodnot P + I + D
  pitch_PID_value = pitch_PID_p + pitch_PID_i + pitch_PID_d;

  pitch_previous_error = pitch_PID_error;
}
```

Obrázek 23 Funkce PIDController() [zdroj vlastní]

5.2.5 ApplyPID

Funkce ApplyPID se stará o správné aplikování PID hodnot u konkrétních motorů, aby došlo ke stabilizaci polohy. Jednotlivé počítání správných hodnot provádí funkce ApplyPitch() a ApplyRoll(). Výsledky se ukládají do globálních proměnných pro každý motor. Před

výsledným posláním pulzů ještě ošetří, aby po aplikování PID hodnot nedošlo k překročení minimální nebo maximální délky pulzu.

```
void ApplyPID() {
  ApplyPitch();
  ApplyRoll();
  /* Ošetření maximální a minimální šířky pulzu */
  if(topLeftThrottle > 2000)
    topLeftThrottle = 2000;
  else if(topLeftThrottle < 1000)
    topLeftThrottle = 1000;
  if(topRightThrottle > 2000)
    topRightThrottle = 2000;
  else if(topRightThrottle < 1000)
    topRightThrottle = 1000;
  if(botLeftThrottle > 2000)
    botLeftThrottle = 2000;
  else if(botLeftThrottle < 1000)
    botLeftThrottle = 1000;
  if(botRightThrottle > 2000)
    botRightThrottle = 2000;
  else if(botRightThrottle < 1000)
    botRightThrottle = 1000;

  /* Aplikace hodnot do motorů */
  topLeftEsc.writeMicroseconds(topLeftThrottle);
  topRightEsc.writeMicroseconds(topRightThrottle);
  botLeftEsc.writeMicroseconds(botLeftThrottle);
  botRightEsc.writeMicroseconds(botRightThrottle);
}
```

Obrázek 24 Funkce ApplyPID() [zdroj vlastní]

5.2.6 ApplyPitch a ApplyRoll

Funkce ApplyPitch a ApplyRoll fungují podobně. Obě upravují globální proměnné jednotlivých motorů pro dosažení požadovaného náklonu. Rozdíl mezi nimi je pouze v tom, že se předpokládá volání ApplyPitch jako první. Ta nastaví hodnoty podle aktuální síly motorů a připočte/odečte od nich PID hodnoty k jednotlivým motorům. Poté ApplyRoll pouze přičte/odečte další PID hodnoty k vypočítaným hodnotám pro jednotlivé motory. PID

používáme nejen k zesilování motorů, ale také k zeslabení motorů na opačné straně pro efektivnější vyrovnávání [16].

```
// Pitch se stará o sílu předních a zadních motorů
void ApplyPitch(){
  /* Pravá strana */
  if(pitch_PID_value > 0){
    botRightThrottle = throttle + pitch_PID_value;
    botLeftThrottle = throttle + pitch_PID_value;

    topLeftThrottle = throttle - pitch_PID_value;
    topRightThrottle = throttle - pitch_PID_value;
  } // Konec pravé strany
  /* Levá strana */
  else if (pitch_PID_value < 0){
    topLeftThrottle = throttle - pitch_PID_value;
    topRightThrottle = throttle - pitch_PID_value;

    botRightThrottle = throttle + pitch_PID_value;
    botLeftThrottle = throttle + pitch_PID_value;
  } // Konec levé strany
}
```

Obrázek 25 Funkce ApplyPitch() [zdroj vlastní]

```
// Roll se stará o sílu levých a pravých motorů
void ApplyRoll(){
  /* Pravá strana */
  if(roll_PID_value > 0){
    botRightThrottle += roll_PID_value;
    topRightThrottle += roll_PID_value;

    topLeftThrottle -= roll_PID_value;
    botLeftThrottle -= roll_PID_value;

  } // Konec pravé strany
  /* Levá strana */
  else if (roll_PID_value < 0){
    topLeftThrottle -= roll_PID_value;
    botLeftThrottle -= roll_PID_value;

    botRightThrottle += roll_PID_value;
    topRightThrottle += roll_PID_value;

  } // Konec levé strany
}
```

Obrázek 26 Funkce ApplyRoll() [zdroj vlastní]

5.2.7 ProcessData

Funkce processData() se stará o sériovou komunikaci s Wi-Fi modulem. Z řetězce přijatého sériovou komunikací vždy vyhledá pozici začínajícího čísla a uloží číslo do adekvátní proměnné, která upravuje buď sílu motorů nebo požadovaný náklon [12].

```
void processData(void)
{
  char* n1pos = strstr(g_buffer, "x=");
  int number1 = -1, number2 = -1, number3 = -1;   // přednastaví špatnou hodnotu pro number1

  // n1pos ukazuje na substring "x=".
  // strlen(n1pos) > 2 se ujišťuje, že následuje alespoň jeden znak po x=
  if ( n1pos != NULL && strlen(n1pos) > 2) {
    // pokud jsme našli první číslo, tak do funkce atoi vložíme číslo začínající za druhým znakem identifikátoru
    number1 = atoi(n1pos+2);

    char* n2pos = strstr(g_buffer, "y="); // Druhé číslo
    if ( n2pos != NULL && strlen(n2pos) > 2 )
      number2 = atoi(n2pos+2);

    char* n3pos = strstr(g_buffer, "z="); // Třetí číslo
    if ( n3pos != NULL && strlen(n3pos) > 2  )
      number3 = atoi(n3pos+2);
  }

  if ( number1 >= 0) {
    Serial.print("New params received. x = ");
    Serial.print(number1);

    throttle = number1;
    set_pitch = number2;
    set_roll = number3;

    Serial.print(", y = ");
    Serial.print(number2);
    Serial.print(", z = ");
    Serial.println(number3);
  } else {
    Serial.println("Wrong data format, please use x=123y=123z=123");
  }
}
```

Obrázek 27 Funkce processData() [zdroj vlastní]

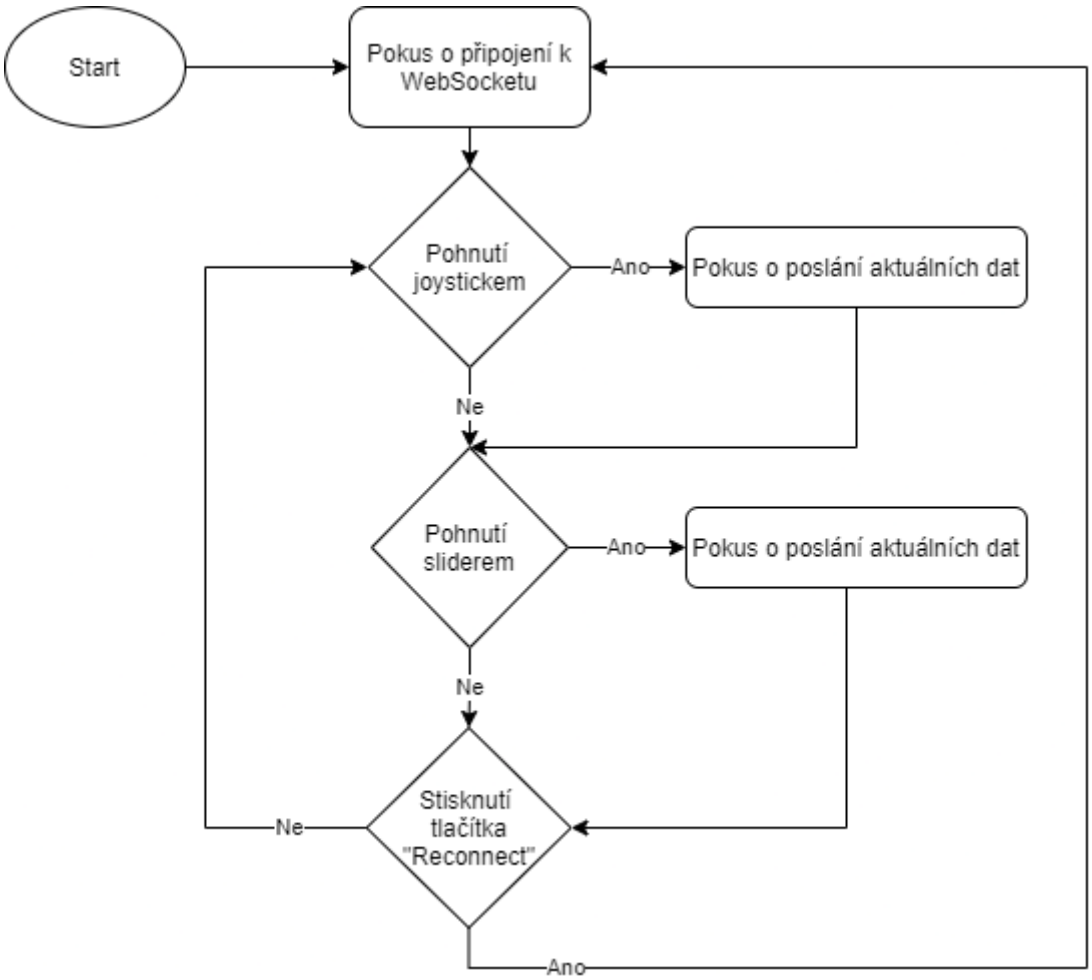
5.3 Mobilní aplikace

Pro tvorbu mobilní aplikace byl použit framework Unity Engine. Hlavním důvodem jeho volby před Android Studiem či Xamarin App byla možnost sestavit projekt pro široké spektrum zařízení. Mobilní aplikace se tak například dá testovat i bez sestavení na notebooku na kterém je projekt vyvíjen (Unity Engine nabízí „Play Mode“ ve kterém lze testovat aplikace bez nutnosti jejich sestavení [13]).

Další důvod je budoucí možnost přijímat zpět data od serveru z modulu IMU a pomocí těchto dat jednoduše v UE simulovat polohu kvadrokoptéry na 3D modelu. Také díky jazyku C#, který UE využívá lze využít knihovny navžené pro .Net framework jako například „System.Net.Websockets“.

5.3.1 Vývojový diagram aplikace

Pro obecný přehled fungování aplikace byl vytvořen vývojový diagram. Na něm je znázorněn logický cyklus vytvořené mobilní aplikace.



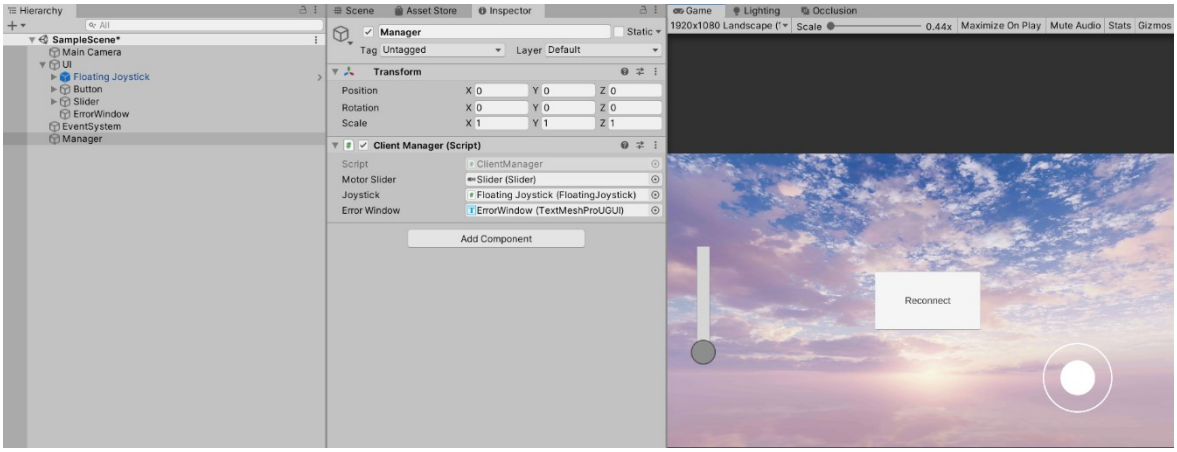
Obrázek 28 Vývojový diagram mobilní aplikace [zdroj vlastní]

5.3.2 Scéna projektu

V projektu Unity Engine musíme nejdřív vytvořit prázdný GameObject (kliknutím pravého tlačítka v Hierarchii projektu a vybráním objektu, který chceme vytvořit [13]), který

pojmenujeme „Manager“ a na který přidáme náš C# script „ClientManager.cs“. Dále vytvoříme Canvas, který pojmenujeme „UI“ a vytvoříme jednotlivé prvky uživatelského rozhraní. Těmto prvkům přiřadíme odpovídající metody z našeho scriptu.

Na levé straně je zobrazena Hierarchie projektu. Uprostřed je zobrazen detail objektu „Manager“, který obsahuje jediný skript, který se stará o celou aplikaci. Na pravé straně je zobrazen náhled výsledné aplikace.



Obrázek 29 Hlavní scéna v UE [zdroj vlastní]

5.3.3 ClientManager.cs

Skripty pro Unity Engine mají podobnou strukturu jako skripty pro mikropočítače, které byly v tomto projektu použity. Funkce Start() a Update() v Unity Engine slouží k téměř stejnému účelu jako funkce Setup() a Loop() v Arduino IDE. Unity však nabízí i mnoho dalších vestavěných funkcí jako například OnEnable() nebo OnDisable(), které se spustí pokaždé při vypnutí či zapnutí daného objektu se skriptem [13], [7]. V tomto projektu si však vystačíme pouze se základními funkcemi.

Pro komunikaci přes WebSocket využíváme knihovnu „System.Net.WebSockets“, která je určena pro .Net framework. V kódu nejdříve vytvoříme adresu WebSocketu. Počítáme s tím, že server si při spuštění vytvoří vlastní přístupový bod (neboli SoftAP, které vytváří Wi-Fi modul), který má defaultně přednastavenou adresu lokálního serveru na „192.168.4.1“ [7]. Port zvolíme takový, jaký jsme zvolili pro náš WebSocket v programu Wi-Fi modulu.

Adresu tedy není potřeba měnit a stačí pouze s mobilním zařízením využívající tuto aplikaci být připojen ke správnému přístupovému bodu.

Dále pouze škálujeme velikost slideru podle velikosti obrazovky zařízení. Další prvky UI jsou škálovány automaticky v Unity Engine díky ukotvení jednotlivých prvků na části obrazovky.

Na konci metody Start, která se provede při spuštění aplikace se automaticky pokusíme připojit k WebSocketu. Dále metodu Update využíváme pouze na čtení hodnot z joysticku a posíláme hodnoty pouze pokud se vstup od uživatele změnil (zamezení zbytečného posílání hodnot, pokud uživatel nepohnul s joystickem).

Ostatní prvky UI mají správnou metodu přiřazenou pomocí inspektoru projektu a využívají vestavěných funkcí UE pro zjištění, jestli se hodnota slideru změnila nebo bylo stisknuto tlačítko (k tomu slouží EventSystem, kterého si lze všimnout v hierarchii projektu). Není tedy potřeba se o tyto prvky starat v kódu aplikace.

```
const string webSocketAddress = "ws://192.168.4.1:81/";
ClientWebSocket socket;

float lastX, lastY;
Zpráva Unity | Počet odkazů: 0
private void Start()
{
    // Inicializace slideru podle velikosti monitoru
    motorSlider.GetComponent<RectTransform>().sizeDelta = new Vector2(Screen.width * 0.1f, Screen.height - (Screen.height*0.2f));

    SocketInit();
}
Zpráva Unity | Počet odkazů: 0
private void Update()
{
    if(lastX != joystick.Horizontal || lastY != joystick.Vertical)
    {
        lastX = joystick.Horizontal;
        lastY = joystick.Vertical;
        SendData();
    }
}
```

Obrázek 30 Funkce Start() a Update() [zdroj vlastní]

V metodě SocketInit() se pokoušíme připojit k zadanému WebSocketu. Pokud se nám připojení povede, chybové okno vymaže veškerý svůj text. Pokud se nám připojení nepovede, v chybovém okně se zobrazí hláška s důvodem chyby. Stejný princip využívá metoda SendData(), která nám vyhodí do chybového okna hlášku o chybě v případě výpadku spojení za letu, jinak se pokouší asynchronně posílat data na WebSocket server. Ve funkci SendData

je také potřeba převést string na ArraySegment<byte> pro možnost poslání dat přes WebSocket.

```
Počet odkazů: 2
public async void SocketInit()
{ // Metoda je veřejná kvůli přistupu tlačítka "Reconnect" k této metodě
    try
    {
        socket = new ClientWebSocket();
        await socket.ConnectAsync(new Uri(webSocketAddress), CancellationToken.None);

        errorWindow.text = "";
    }
    catch (Exception e)
    {
        errorWindow.text = e.Message;
        throw;
    }

}
Počet odkazů: 2
async void SendData()
{
    byte[] encoded = Encoding.UTF8.GetBytes(ValueToSend());
    ArraySegment<byte> buffer = new ArraySegment<Byte>(encoded, 0, encoded.Length);
    try
    {

        await socket.SendAsync(buffer, WebSocketMessageType.Text, true, CancellationToken.None);

        errorWindow.text = "";
    }
    catch (Exception e)
    {
        errorWindow.text = e.Message;
        throw;
    }
}
```

Obrázek 31 Funkce SocketInit() a SendData() [zdroj vlastní]

Poslední důležitá funkce je „private string ValueToSend()“, která vrací správně formátovaný řetězec. Využívá funkce Map, která převede hodnotu z rozsahu od -1 po 1 na hodnoty v rozsahu od -45 po 45. Tím je nastavena maximální velikost náklonu na 45° kvůli bezpečnosti.

Výslednou hodnotu převede na celé číslo, aby jej algoritmus stabilizátoru letu zvládnul zpracovat.

```
Počet odkazů: 1
string ValueToSend()
{
    string valueToSend =
        "x=" + (1000 + motorSlider.value) +
        "y=" + Convert.ToInt32(Map(-1,1,-45,45, joystick.Vertical)) +
        "z=" +  Convert.ToInt32(Map(-1,1,-45,45, joystick.Horizontal));
    return valueToSend;
}

Počet odkazů: 2
float Map(float origMin, float origMax, float resultMin, float resultMax, float number)
{
    float normalized = (number - origMin) / (origMax - origMin);
    return normalized * (resultMax - resultMin) + resultMin;
}
```

Obrázek 32 Funkce ValueToSend() a Map() [zdroj vlastní]

ZÁVĚR

Cílem této práce bylo navrhnout vlastní systém pro fungující model kvadrokoptéry, řízený mikropočítačem. Navržený systém se skládá ze stabilizátoru letu, Wi-Fi modulu a mobilní aplikace. Díky použití mobilního telefonu jako ovládacího zařízení je především možno obejít hardwarové omezení běžně používaných dálkových ovladačů a lze přidat libovolný počet ovládacích prvků, který by byl jinak omezen počtem páček a tlačítek u konkrétního modelu dálkového ovladače.

Pro začátek bylo potřeba otestovat funkčnost všech zařízení a byl sestrojen prototyp využívající klíčových funkcí. Tento prototyp využíval dvou motorů pro stabilizaci polohy na pouze jedné ose a přijímal data z mobilní aplikace o požadované rychlosti motorů. Po úspěšném vyladění PID konstant pro správný chod stabilizace polohy proběhl návrh softwaru pro finální konstrukci se všemi čtyřmi motory a byla přidána možnost přijímat také data o požadovaném náklonu. Konstrukce kvadrokoptéry byla vytvořena z dřevěných hranolů, které mají přijatelné parametry pro hmotnost a odolnost a také umožňují jednoduchou modifikaci pro budoucí přidání modulů.

Celý model je tak otevřen libovolným modifikacím a v budoucnu se plánuje přidání zpětné vazby od modelu do mobilní aplikace. Také je v plánu vylepšit uživatelské rozhraní mobilní aplikace a přidání vizuálních simulací pomocí zpětné vazby z modelu kvadrokoptéry. Celková cena kvadrokoptéry se pohybuje kolem 2 tisíc korun českých a může se lišit podle zdrojů použitých součástek.

SEZNAM POUŽITÉ LITERATURY

[1] ZUCKERMAN, Arthur. 68 DRONES STATISTICS: 2020/2021 MARKET SHARES, APPLICATIONS & FORECASTS. *CompareCamp* [online]. 24.05.2020 [cit. 2021-5-12]. Dostupné z: https://comparecamp.com/drones-statistics/

[2] VÁŇA, Vladimír. ARM pro začátečníky. Praha: BEN - technická literatura, 2009, 195 s. ISBN 978-80-7300-246-6

[3] FORD, Justin. The History Of Drones (Drone History Timeline From 1849 To 2019). *Dronethusiast* [online]. 2019 [cit. 2021-4-8]. Dostupné z: https://www.dronethusiast.com/history-of-drones/

[4] The history of drones and quadcopters. *Quadcopter Arena* [online]. [cit. 2021-4-9]. Dostupné z: https://quadcopterarena.com/the-history-of-drones-and-quadcopters/

[5] 12 Military Drones Employed By The US Military. *Operation Military Kids* [online]. 2021 [cit. 2021-4-12]. Dostupné z: https://www.operationmilitary-kids.org/military-drones/

[6] TAGAY, Adilet, Abylkaiyr OMAR a Hazrat ALI. *Development of control algorithm for a quadcopter: 5th International Conference on Computer Science and Computational Intelligence* [online]. Elsevier, 2021 [cit. 2021-4-23]. Dostupné z: doi:https://doi.org/10.1016/j.procs.2021.01.003

[7] PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN – technická literatura, 2004, 159 s. ISBN 80-7300-110-1

[8] Arduino for STM32: Everything relating to using STM32 boards with the Arduino IDE and alternatives. *Stm32duino* [online]. [cit. 2021-5-9]. Dostupné z: https://www.stm32duino.com/

[9] PID Controller Explained. *PID Explained* [online]. [cit. 2021-5-9]. Dostupné z: https://pidexplained.com/pid-controller-explained/

[10] TEEL, John. Introduction to the STM32 Blue Pill (STM32duino). *Predictable Designs* [online]. 01.04.2020 [cit. 2021-5-9]. Dostupné z: https://predictable-designs.com/introduction-stm32-blue-pill-stm32duino/

[11] TAI, Hideaki. MPU9250: Arduino library for MPU9250. *GitHub* [online]. [cit. 2021-5-10]. Dostupné z: https://github.com/hideakitai/MPU9250

[12] DOLINAY, Jan. Processing data from serial line in Arduino. *Code Project* [online]. 6.8.2018 [cit. 2021-5-11]. Dostupné z: https://www.codeproject.com/Articles/1255400/Processing-data-from-serial-line-in-Arduino

[13] Unity User Manual. *Unity Documentation* [online]. Unity Technologies, 2020 [cit. 2021-5-11]. Dostupné z: https://docs.unity3d.com/Manual/index.html

[14] Gyroskop, akcelerometr a magnetometr MPU-9250. *Drátek Návody* [online]. Havlíčkův Brod: Eclipsera [cit. 2021-5-11]. Dostupné z: https://navody.dratek.cz/navody-k-produktum/gyroskop-akcelerometr-a-magnetometr-mpu-9250.html

[15] VALVANO, Jonathan W. Embedded systems: Introduction to the Arm Cortex(TM)-M3 microcontrollers. 2nd ed. s.l.: CreateSpace, 2012, xii, 462 s. ISBN 978-1477508992

[16] CINNAMON, IAN, DIY Drones for the Evil Genius: Design, Build, and Customize Your Own Drones. McGraw-Hill Education TAB, 2016, 176 s. ISBN 978-1259861468

[17] CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol, CA: O'Reilly, 2005, xvi, 377 p. ISBN 0596007558

[18] SIEPERT, Brian. MPU6050 6-DoF Accelerometer and Gyro. *Adafruit* [online]. 6.11.2019 [cit. 2021-5-11]. Dostupné z: https://learn.adafruit.com/mpu6050-6-dof-accelerometer-and-gyro

[19] DUKOWITZ, Zacc. Drone Controllers: A Look at How They Work, Important Terminology, and Why They're Unique in the RC Aircraft World. *UAV Coach* [online]. 19.09.2019 [cit. 2021-5-12]. Dostupné z: https://uav-coach.com/drone-controller/

[20] Zoon: 2.4Ghz Long Range (LoRa), High Data Rate Radio Telemetry Module. *Dronee* [online]. Dronee, 2021 [cit. 2021-5-12]. Dostupné z: https://dronee.aero/pages/zoon

[21] Autopilot Hardware Options. *Ardupilot* [online]. [cit. 2021-4-5]. Dostupné z: https://ardupilot.org/copter/docs/common-autopilots.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UAV Unmanned Aerial Vehicle.

IDE Integrated Development Environment.

PWM Pulse Width Modulation.

UE Unity Engine

ESC Electronic Speed Controller

IMU Inertial Measurement Unit

FAA Federal Aviation Administration

FPV First Person View

UI User Interface

PID Proportional-Integral-Derivative

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD

Přiložené CD obsahuje:

- Bakalářskou práci ve formátu .pdf.
- Všechny zdrojové kódy a instalační soubor pro mobilní aplikaci v souboru .zip.