ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

Katedra teoretické elektrotechniky

Diplomová práce

Řídící systém pro autonomní dron

Autor práce: Bc. Dominik Pauli
Vedoucí práce: Ing. Petr Kropík, Ph.D.

Plzeň 2017

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Dominik PAULI**

Osobní číslo: **E15N0005P**

Studijní program: **N2612 Elektrotechnika a informatika**

Studijní obor: **Elektronika a aplikovaná informatika**

Název tématu: **Řídicí systém pro autonomní dron**

Zadávající katedra: **Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Prostudujte problematiku algoritmů pro rozpoznávání okolí a řízení ve 3D prostoru.
2. Navrhněte řídicí systém pro autonomní dron pro pohyb a sledování v uzavřených členitých prostorách dle mapy uložené v paměti.
3. Navrhněte vhodný hardwarový sytém pro řízení dronu (senzory, řídicí moduly).
4. Navrhněte inteligentní řídicí algoritmy pro zpracování a vyhodnocení telemetrických dat a následné řízení dronu.
5. Vyřešte vzdálené řízení a přenos dat mezi základnou a dronem.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah kvalifikační práce: **40 - 60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **A. Hussein, A. Al-Kaff, A. de la Escalera and J. M. Armingol. "Autonomous indoor navigation of low-cost quadcopters", Service Operations And Logistics, And Informatics (SOLI), 2015 IEEE International Conference on, Hammamet, 2015, pp. 133-138. doi: 10.1109/SOLI.2015.7367607**

2. **T. T. Mac, C. Copot, A. Hernandez and R. De Keyser. "Improved potential field method for unknown obstacle avoidance using UAV in indoor environment", 2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, 2016, pp. 345-350. doi: 10.1109/SAMI.2016.7423032**

3. **L. V. Santana, A. S. Brando, M. Sarcinelli-Filho and R. Carelli. "A trajectory tracking and 3D positioning controller for the AR.Drone quadrotor", Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, Orlando, FL, 2014, pp. 756-767. doi: 10.1109/ICUAS.2014.6842321**

4. **A. Hornung at al. "OctoMap: an efficient probabilistic 3D mapping framework based on octrees" Autonomous Robots, vol. 34, num. 3, 2013, pp. 189-206. doi: 10.1007/s10514-012-9321-0**

5. **F. Cocchioni, A. Mancini and S. Longhi, "Autonomous navigation, landing and recharge of a quadrotor using artificial vision", Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, Orlando, FL, 2014, pp. 418-429. doi: 10.1109/ICUAS.2014.6842282**

Vedoucí diplomové práce: **Ing. Petr Kropík, Ph.D.**
Katedra teoretické elektrotechniky

Datum zadání diplomové práce: **14. října 2016**

Termín odevzdání diplomové práce: **19. května 2017**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 14. října 2016

Abstrakt

Tato diplomová práce se zabývá návrhem řídícího systému pro autonomní dron. V teoretické části je proveden návrh takovéhoto systému. Jako kontrolér pro navigaci je využito Raspberry Pi a jako dron AR Drone 2.0. Praktická část se zaměřuje na implementaci navigačního systému pro navigaci z mapy zadané uživatelem a uložené v paměti. Implementace je prováděna v jazycích JavaScript a TypeScript.

Klíčová slova

RaspberryPi, Dron, Navigační systém, Mapa, JavaScript, TypeScript

Abstract

Pauli, Dominik. *The Control system for autonomous drone* [*Řídící systém pro autonomní dron*]. Pilsen, 2017. Master thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Petr Kropík

---

This master's thesis deals with proposal of control system for an autonomous drone. In theoretical part this system is proposed. As controller for navigation the Raspberry Pi and as drone the AR Drone 2.0 are used. Practical part of thesis deals with implementing the navigation system from a map set by the user and saved in memory. Implementation is made using programming languages JavaScript and TypeScript.

Keywords

RaspberryPi, Drone, Navigation system, Map, JavaScript, TypeScript

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

V Plzni dne 17. května 2017

Bc. Dominik Pauli

.....

Podpis

Obsah

Seznam obrázků

Seznam symbolů a zkratek

A	Ampér
V	Volt
GB	Giga byte
MHz	Mega hertz
HDD	Hard disc
RPI	Raspberry Pi
ftp	File transfer protocol
csi	Camera Serial Interface
dsi	Display Serial Interface
sdk	Software developement kit

1

Úvod

Tato práce se zabývá řešením řídících systémů pro autonomní dron. V první části práce se nachází rešerše existujících řešení navigace v prostoru a navigačních systémů pro drony. Druhá část práce je zaměřena na popis návrhu takovéhoto systému pro autonomní dron. Třetí část se zabývá popisem návrhu a realizací takovéhoto systému a navržením prototypu. V závěru práce je zhodnocen navržený systém a je porovnám s existujícími systémy, jsou navrženy jeho další potřebné změny a budoucí vývoj.

Tato práce je řešena z důvodu existence rozsáhlého množství řešení navigace pro autonomní drony. Tato problematika je vysoce rozsáhlá a proto byla provedena rešerše existujících řešení. Pro předvedení této problematiky do praxe byl navržen navigační systém pro autonomní dron s navigací podle statické mapy zadané uživatelem. Toto řešení bylo poté implementováno a byl vytvořen prototyp.

V budoucnu by řídící systémy tohoto typu po delším vývojovém cyklu mohly sloužit například ve skladech pro rychlé vizuální kontroly namísto lidské přítomnosti, popřípadě se specializovaným typem dronu by bylo možné je využívat v nebezpečných prostorech nepřístupných lidem, popřípadě pro rychlou odezvu vůči narušiteli v objektu při vybavení kamerami či jinými výstražnými prvky.

2

Existující řešení navigačních systémů pro drony

Problematika navigačních systémů je v současné době aktivně řešena. Aktivně se řeší jak témata navigace dronu podle statické mapy, tak se řeší navigace pomocí rozpoznávání okolí a navigace v prostředí pomocí senzorů a to autonomně.

Například článek *A Trajectory Tracking and 3D Positioning Controller for the AR.Drone Quadrotor* [3] navrhuje způsob jakým by se dala určovat trajektorie a pozice dronu s využitím vizuálních a pohybových dat. Toto řešení působí vysoce sofistikovaně a podle uvedených experimentálních výsledků také jako velmi účinné.

Problém, podobný problému řešenému v této práci, je řešen ve článku *Autonomous Indoor Navigation of Low-Cost Quadcopters* [1] tento článek se zabývá také navigací dronu podle mapy a využívá vnitřního kontroléru dronu pro určení pozice, nicméně tato řešení jsou spíše ve formě simulace než pro použití v reálném zařízení.

Další článek zabývající se podobnou problematikou je *Improved Potential Field Method for Unknown Obstacle Avoidance Using UAV in Indoor Enviroment* [2]. Zde se navrhuje základní způsob navigace podle mapy a je zde navržena metoda pro orientaci s využitím potenciálového pole s využitím kamery. Tato metoda, ale vyžaduje další prostudování a otestování.

Problematika je také řešena za využití ROS což je operační systém pro robotiku [15]. Pro tento systém je navržena aplikace s navigací podle vizuálních dat za využití kamery. Tento systém momentálně není, ale aktivně vyvíjený. Pro navigaci z vizuálních dat také existuje kurz Vizuální navigace pro autonomní drony [14] kde je tato problematika řešena.

3

Návrh řešení navigace

Tato kapitola se zabývá návrhem řešení navigačního systému pro autonomní dron, navigací podle mapy zadané uživatelem, orientací v prostoru a ovládáním dronu. Je zde proveden návrh kompletního systému včetně neimplementovaných částí.

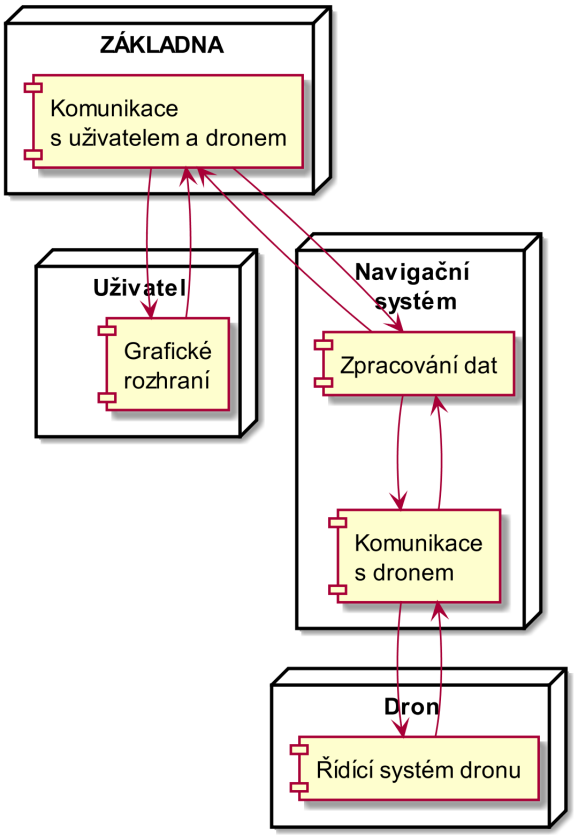
Celý systém jak navržený, tak implementovaný byl vytvářen za zjednodušení problému z 3D na 2D problém což znamená, že se předpokládá, že dron se pohybuje ve fixní výšce 1 m. Toto zjednodušení problému bylo provedeno z důvodu odstranění zbytečné komplexnosti řešení. Po kompletním návrhu systému jako 2D je přechod do 3D možný bez větších problémů.

3.1 Vize systému

Pro autonomní navigační systém pro dron je zapotřebí zajistit několik podstatných věcí. Pro samotné ovládání je třeba navrhnout způsob jakým může uživatel nastavit počátek a konec cesty pro dron, stejně tak je důležité zajistit detekci blízkých překážek popřípadě získání této informace od uživatele. Pro řešení tohoto problému je třeba navržení způsobu komunikace mezi řídícím systémem a uživatelem.

Dalším problémem je komunikace mezi dronem a řídícím systémem a poté samotné ovládání dronu. Samotná komunikace mezi uživatelem a řídícím systémem by měla být prováděna principem klient server, kde by řídící systém měl být dostupný vzdáleně nejlépe kontrolovaný pomocí základy a až ta by poskytovala přístup uživateli a prováděla samotnou komunikaci.

3.2 Diagram systému



Obr. 3.1: Diagram navrženého systému

3.3 Ovládání dronu

Samotný dron by ideálně měl být vybaven kamerami pro zachycení obrazu a následnou detekci překážek. Standardně se při těchto úkonech využívá kamera v přední části a kamera ve spodní části mířící dolů. Pro účely určení výšky se také využívá ultrazvukového senzoru mířícího dolů. Pro přesnější určování vzdálenosti od překážek je také možné využít laserové senzory vzdálenosti.

Pro řešení se nabízí návrh vlastního dronu z existujících volně dostupných dílů, které jsou v současné době již velmi snadno k sehnání. Výhodou tohoto řešení by byla možnost spojení řídícího kontroléru pro samotnou kontrolu motorů a navigačního kontroléru do jednoho kontroléru, tím by se snížil odběr a u využití komerčního dronu potřeba přidané druhé baterie pro navigační kontrolér. Nevýhodou tohoto řešení, ale je nemožnost využití navigačního kontroléru jako samostatného řešení pro libovolný typ dronu.

V případě navigace z mapy zadané uživatelem je zapotřebí zajistit určení uletěné vzdálenosti i v případě, že se neprovádí detekce překážek což je možné provést několika způsoby. Například pomocí výpočtu z doby a rychlosti letu popřípadě pomocí využití spodní kamery dronu jako reference a následné analýzy změny obrazu. Uživatel by v tomto případě měl sám určit startovní a cílovou lokaci včetně pozic překážek umístěných v prostoru.

V případě navigace pomocí kamer by se mělo nejprve provést mapování prostoru a poté vygenerovat mapu překážek v prostoru podle které by probíhala samotná navigace. U tohoto řešení by bylo zapotřebí vysokého výkonu kontroléru a s velkou pravděpodobností se toto řešení hodí spíše na zařízení řízené z externího počítače, ale poté se začne projevovat problém přenosu velkého množství obrazového materiálu a nutnost jeho rychlého zpracování.

3.4 Hledání cesty

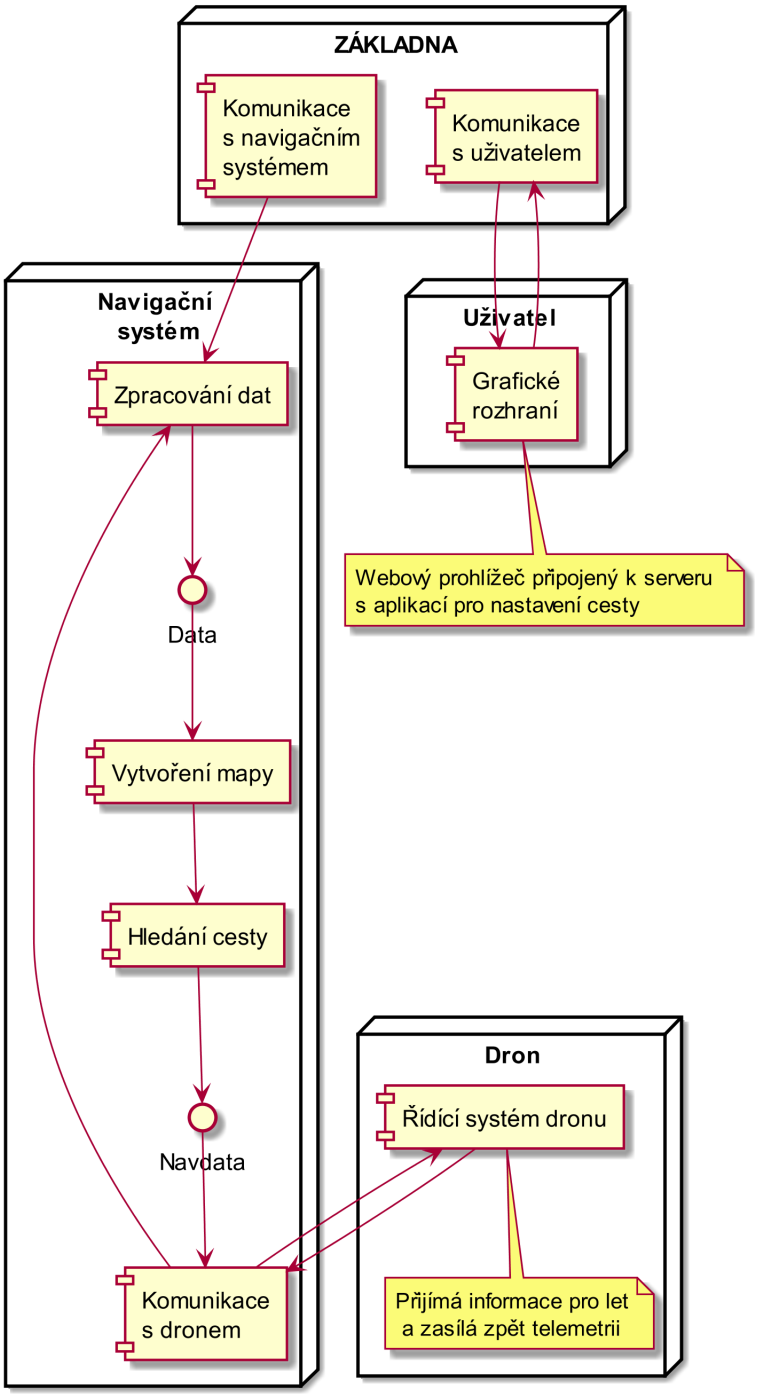
V případě navigace podle mapy uložené v paměti je potřeba v mapě nalézt startovní a cílovou pozici pomocí prohledání mapy. Po nalezení startovní a cílové pozice je zapotřebí nalézt v mapě překážky, popřípadě začít s vyhledáváním cesty v mapě. Toto hledání je ideální provádět iteračním do doby než bude nalezena nejkratší možná cesta. Pro dosažení nejkratší možné cesty je třeba definovat zastavovací podmínku pro vyhledávací algoritmus, která algoritmus zastaví v případě, že se po určitý počet iterací nemění nejkratší cesta.

Pro samotné hledání cesty existuje mnoho algoritmů. Jako nejjednodušší jednoduchý algoritmus typu brute force, který vyzkouší většinu možností a vrátí nejkratší možnou cestu. Jako další možnosti je možné použití složitějších algoritmů jako typu A* a D*. Tyto algoritmy se používají zejména při vývoji počítačových her, ale pro tento případ se tyto problémy velice podobají.

3.5 Struktura navigačního softwaru

Navigační software by měl být modulární. Hlavní modul by se měl skládat ze serveru, který je zde jako prvek programu pro komunikaci a spouštění ostatních podprogramů pro řízení dronu a jeho zpětnou vazbu. Tento modul je zde pro samotné řízení dronu a získání telemetrie z dronu, tento modul by měl být jediný modul, který je třeba pozměnit při změně dronu a měl by být jediným bodem interakce mezi navigačním systémem a dronem. Dalším modulem by měl být modul pro navigaci, který získá data od serveru a převede data od uživatele na navigační data pro navigační modul. Poslední modul by měl být modul s uživatelským rozhraním, tento modul má za úkol získat data o startovní a konečné pozici od uživatele a v případě navigace z mapy získat od uživatele mapu.

3.6 Vývojový diagram softwaru



Obr. 3.2: Vývojový diagram navrženého softwaru

4

Popis použitého hardwaru

Tato kapitola se zabývá popisem specifikací použitého hardwaru pro prototyp s ohledem na jeho výhody a nevýhody s odůvodněním jeho použití.

4.1 Dron AR Drone

Jako prototyp pro testování byl využit dron AR Drone 2.0 od francouzké firmy Parrot. Tento model se začal vyrábět v roce 2012 a v současné době byl již nahrazen novým modelem BEBOP s poměrně jinými parametry.

Obr. 4.1: AR Drone 2.0

Dron se skládá ze základní konstrukce s rozměry 50 cm x 50 cm a k dispozici je trup pro externí použití, který zakrývá pouze kontrolér a baterii a trup pro použití v interiéru, který kromě zakrývání kontroléru obsahuje ještě ochranné části okolo vrtulí, tak aby se

nepoškodily menším nárazem do objektu. Oba tyto trupy jsou vyrobeny z polystyrenu. Dron je osazen čtyřmi motory. Motory jsou bezkartáčové s výkonem 14,5 W a dosahují 28 500 otáček za minutu se samomaznými bronzovými ložisky. Výkon je přenášen pomocí nylonových ozubených kol na vrtule s převodem 1 : 8,75 na vrtule s rozměry 12 x 0,5 x 9 palců.

Dron je osazen kontrolérem s 1 GHz ARMv7 procesorem a 800 MHz video DPS TMS320DMC64x s pamětí 1 Gb DDR2 RAM na 200 MHz a 32 MB NAND Flash paměti. Komunikace je řešena pomocí Wi-Fi čipu 802.11/b/g/n Atheros. Dron je opatřen 3-osým akcelerometrem s přesností +-50 mg, 3-osým gyroskopem s přesností 2000 °/s a tlakovým senzorem +-10 Pa. Dále je osazený 40 kHz ultrazvukovým senzorem mířícím dolů. Dron je také osazen dvěmi kamerami. Kamera mířící dolů je VGA kamera s rozlišením 320 x 240 při 60 snímcích za sekundu. Kamera mířící dopředu je HD kamera s rozlišením 1280 x 720 při 30 snímcích za sekundu s čočkou zabírající 92°. Dron je napájen z baterie 1000 mAh LiPo 10c, která dodává dronu letovou dobu přibližně 12 minut a má dobu nabíjení přibližně 90 minut.

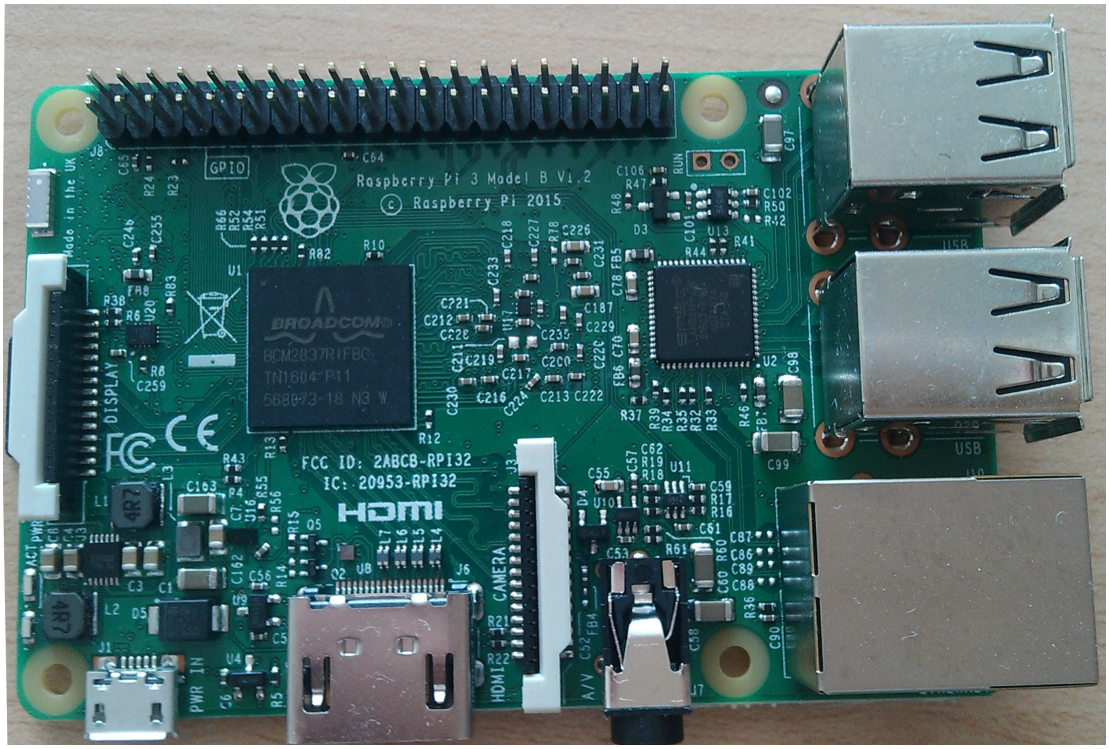
Kontrolér je opatřen uzavřenou distribucí operačního systému Linux verze 2.6.32.9(BusyBox). Nainstalované služby jsou Parrot aplikace pro ovládání a FTP, DHCP a Telnet klient.

AR Drone 2.0 byl vybrán z důvodu úspory času, kde vlastní návrh dronu by zabral času velké množství a stejně tak jeho konstrukce. AR Drone 2.0 byl také zvolen díky existenci knihoven umožnujících využití jeho Wi-Fi rozhraní pro snažší kontrolu bez nustnosti vytváření vlastního komunikačního protokolu pro vlastnoručně navržený dron. Další nespornou výhodou je rozsáhlá komunita zabývající se problematikou programování tohoto typu drona a relativně jednoduché programování v porovnání s konkurencí. Další nespornou výhodou je osazení dvěmi kamerami a jednoduchý přístup k jejich výstupům. Nevýhodou tohoto drona je, že samotný kontrolér drona je opatřen uzavřeným systémem a aplikace tedy musí být na externím kontroléru. Toto lze také brát jako výhodu při designu prototypu, protože je poté možné existující aplikaci při vhodném oddělení výstupů pro dron snadno změnit pro jakýkoliv jiný dron.

4.2 Raspberry Pi

Raspberry Pi je počítač postavený na procesoru ARM. Pro účely této práce byl vybrán Raspberry Pi 3 model B jako řídící počítač. Raspberry Pi je napájené z baterie v případě umístění na dronu a v případě umístění mimo dron ze sítě pomocí adaptéru. Napájecí napětí je 5 V a proud 2 A, napájení je na Raspberry Pi 3 řešeno pomocí micro USB rozhraní.

Raspberry Pi je postavena na SoC Broadcom BCM2837. Osazena je procesorem se 4 jádry ARM Cortex-A53 taktovanými na frekvenci 1.2 GHz. Procesor je také možné mírně taktovat za použití firmwaru. Procesor disponuje 1 GB LPDDR2 RAM paměti



Obr. 4.2: Raspberry Pi 3

pracující na frekvenci 900 MHz. Na rozdíl od předchozích modelů disponuje také grafickým procesorem Broadcom VideoCore IV. Pro připojení k síti lze využít ethernet 10/100 kde je Raspberry Pi 3 osazena standardním ethernetovým konektorem, popřípadě využít wi-fi v pásmi 2,4 GHz 802.11n. Je možné také připojení pomocí Bluetooth 4,1. Úložný prostor na Raspberry Pi 3 je řešen pomocí externí microSD karty.

Raspberry Pi disponuje mnoha vstupy a výstupy zejména 40 programovatelnými GPIO piny, 3,5 mm audio-video jack konektorem, čtyřmi USB konektory, ethernetovým konektorem a sériovými interface pro kameru a display (CSI a DSI). V neposlední řadě disponuje také konektorem HDMI pro připojení monitoru.

Výhodou Raspberry Pi 3 je zejména nízká cena mikropočítače pohybující se do 1 000 Kč spolu s tím, že poskytuje poměrně vysoký výkon a zachovává si poměrně malé rozměry (85 mm x 56 mm). Další výhodou jsou poměrně četně vstupy a výstupy zajišťující flexibilitu této platformy a velmi rozsáhlá komunita vývojářů pracujících s touto platformou, která zajišťuje podporu u velkého množství softwaru a velké množství podkladů a informací pro práci s touto platformou. Neposlední výhodou je využití operačního systému Linux na této platformě ve vlastních distribucích zvaných Raspbian což je upravená distribuce debian pro tuto platformu.

Nevýhodou Raspberry Pi 3 je zejména u modelu 3 vysoká spotřeba energie, která je zejména způsobena přidáním wi-fi komunikačního rozhraní. Druhou nevýhodou je neúplný přístup k registrům procesoru z důvodu využívání operačního sytému na platformě.

Platforma Raspberry Pi 3 byla zvolena z důvodu možnosti využití operačního systému Linux na platformě malých rozměrů s poměrně vysokým výkonem. Raspberry Pi 3 plat-

forma byla také zvolena z důvodu dobré kompatibility s frameworkem Node JS, který byl zvolen pro vývoj celé aplikace a je na Raspberry Pi 3 přímo podporován. Dalším rozhodujícím faktorem byla přítomnost wi-fi přímo na kontroléru a napájení přes mikro USB což podstatně zjednodušuje hardwarové řešení prototypu.

5

Popis použitého softwaru

Tato kapitola se zabývá popisem použitého software a odůvodněním jeho použití s příklady implementace v práci. V této sekci jsou také popsány využité knihovny spolu s příklady jejich využití a popisem jejich funkce.

5.1 Jazyk HTML5

Html5 je popisovací jazyk a je to současná verze Html standartu. Narozdíl od předchozích verzí Html5 obsahuje nově canvas prvek, který dovoluje vytváření grafických prvků přímo v prohlížeči a také podporuje nově i video a audio prvky. Díky implementaci těchto prvků v současné době nahrazuje zastaralou technologii pro audio a video Adobe flash.

5.2 Programovací jazyk JavaScript

JavaScript je jazyk určený zejména pro vytváření webových aplikací, ale se zvyšujícím se výpočetním výkonem je stále více využíván také pro back-end vývoj softwaru. JavaScript je interpretovaný jazyk což znamená, že není přímo překládán do zdrojového kodu, ale využívá takzvaný interpreter, který program již rovnou spouští z již existujících dříve vytvořených podrutin.

Výhodou JavaScriptu je multiplatformovost a možnost vývoje grafického rozhraní uživatele za využití html webového rozhraní a back-endu aplikace s využitím frameworku bez nutnosti využití jiného jazyku.

JavaScript je využíván současně s TypeScriptem, kdy TypeScript je do JavaScriptu kompilován. Pro uživatelské rozhraní je výhodou snadná práce s html5 canvas objektem, který dovoluje vykreslování objektů a efektivita, protože je tento skript spouštěn až u uživatele ve webovém prohlížeči. U back-endu je výhodou možnost kombinace s frameworkem NodeJS, který dovoluje interakci s hardwarem.

Výhodou je také zjednodušení vývoje díky možnosti využití jazyku TypeScript, který dovoluje použití objektově orientovaného programování a také existence velkého množství

knihoven a rozsáhlé komunity využívající tento jazyk.

Nevýhodou tohoto jazyka je jistý over-head, kde není přistupováno přímo k hardwaru a lze tedy hovořit o nižší efektivitě než při použití kompilovaných jazyků jako je C/C++, kde je vývoj podstatně více časově náročný.

5.3 Programovací jazyk TypeScript

Typescript je jazyk pro vytváření aplikací v jazyce JavaScript. TypeScript přidává do jazyka JavaScript další typy a objektovou funkčnost jako jsou třídy a moduly. TypeScript se kompiluje do čitelného JavaScriptu. Pro kompilaci se využívá kompilátor tsc a je momentálně udržován firmou Microsoft.

Výhodou TypeScriptu je možnost práce s objekty ve skriptovacím jazyce, což samotný JavaScript nepodoporuje a proto je zde vývoj a návrh složitějších aplikací podstatně problematičtější a větší aplikace jsou poté složitější na návrh.

TypeScript byl zvolen z důvodu snažšího návrhu architektury aplikace a její realizace za využití objektového programování. Je možné kod rozčlenit do jednotlivých tříd a podstatně jej tak zpřehlednit a využít dědičnosti prvků. Dalším důvodem pro zvolení TypeScriptu je zachování stejné efektivity kodu jako u samotného JavaScriptu tím, že se kompiluje do JavaScriptu a až ten je používán na aplikační úrovni. Stejně tak je výhodou podpora tohoto jazyka ze strany vývojových prostředí a velké množství materiálů od firmy Microsoft a od komunity využívající tento jazyk. Výhodou je také multiplatformovost, která je zaručena již tím, že tato aplikace využívá NodeJS jako back-end a front-end je pojat jako webová aplikace. Díky kompilaci do JavaScriptu je tento jazyk také kompatibilní s NodeJS a knihovnami pro kontrolu drona AR Drone a ardrone-autonomy. Pro tuto práci byla zvolena momentálně aktulní verze (TypeScript 2.2) dostupná z [6].

5.4 Framework NodeJS

NodeJS je framework řízený asynchronními událostmi navržený pro vývoj webových aplikací. Výhodou NodeJS je využití neblokujících událostí narozdíl od standartních vláknových aplikací, kde se mohou jednotlivé aplikace blokovat. NodeJS využívá standartně jazyk JavaScript, ale pro účely této práce byl zvolen jazyk TypeScript, který je také kompatibilní. JavaScript je interpretován pomocí JavaScriptového enginu Google V8.

Výhodou NodeJS je multiplatformovost a existence runtime i na hardware jako je RaspberryPi a také možnost využití NodeJS jako webserver, tak jako back-end celé aplikace. Výhodou je také neustálý vývoj a rostoucí popularita webových aplikací a možnost využití stejného jazyka jako pro uživatelské rozhraní.

NodeJS byl zvolen z důvodu existence knihoven pro zvolený dron, ale také proto, že je jej možné výhodně propojit s uživatelským rozhraním bez ztráty efektivity a poslední verze jsou po optimalizaci schopné konkurovat rychlostí i programům kompilovaným v jazyce

C/C++. Dalším důvodem je rozsáhlá komunita a velké množství dostupných materiálů a knihoven.

5.5 Použité knihovny

Pro vývoj navrženého softwaru byly použity dvě hlavní knihovny. Tyto knihovny byly využity hlavně z důvodu usnadnění kontrolu dronu bez nutnosti vyvíjet vlastní komunikační protokol pro kontrolu dronu. Pro další vývoj by bylo vhodné navrhnout vlastní řešení problémů, které momentálně řeší knihovny zejména pro možnost využití jiných typů dronů.

Jako hlavní knihovna je použita knihovna nodecopter. Instalace této knihovny je možná pomocí package manageru NodeJS. Využitím příkazu:

```
1 npm install ar-drone
```

Výrobce dronu AR-Drone 2.0 Parrot nabízí SDK v jazyce C s poměrně dobrou dokumentací a pomocí tohoto SDK je možné kromě samotné kontroly také získávat video z kamer a data ze senzorů dronu. Nicméně toto SDK je poměrně nízkoúrovňové, což se nehodí při práci s vysokoúrovňovými koncepty. Pro usnadnění proto existuje knihovna Nodecopter, která převádí interface pro kontrolu do jazyka JavaScript a zjednodušuje tak práci s dronem. Tato knihovna funguje jako klient, který poté zasílá příkazy přímo do dronu. Na příkladu je vidět využití knihovny kde dron vzlétne poletí po určitou dobu vpřed a přistane. Zde je vidět nevýhoda knihovny v tom, že namísto vzdálenosti se udává čas pohybu. Tato knihovna je distribuována pod MIT licencí. Dostupná z [7].

```
1  var drone = require('ar-drone');
2  var client = drone.createClient();
3
4  client.takeOff();
5
6  client.after(1000, function()
7  {
8    this.front(0.5);
9  })
10 .after(1000, function ()
11 {
12   this.stop();
13   this.land();
14 });
```

Příklad aplikace s použitím knihovny Nodecopter

Tato knihovna se, ale neosvědčila pro přesnější pohyb, a proto byla využita ještě knihovna ardrone-autonomy. Tato knihovna je přímo určena k automatizaci AR-Drone. Instalace knihovny je možná také pomocí package manageru NodeJS pomocí příkazu:

```
npm install ardrone-autonomy
```

Knihovna je postavena na předchozí knihovně, ale na rozdíl od kontroly pomocí času, zde je možné přímo vytvořit takzvané mise s definovanou výškou letu a vzdáleností pro let. Pro samotné určení vzdálenosti tato knihovna využívá odhad výpočet uražené vzdálenosti (vzdálenost = rychlost x čas), čímž se získá odhad polohy relativní ke startovní pozici. Knihovna také podporuje navigaci pomocí značek, kde se využívá kamery a zpětné projekce pro určení pozice s využitím rozšířeného Kalmanova filteru pro korekci chyby. Tato metoda se také využívá při navigaci kdy koriguje uraženou vzdálenost pomocí detekce vzoru na podlaze. Pro samotný let dronu na určení místo je poté využit PID regulátor. Tato knihovna se již hodí pro tento projekt podstatně více a proto je extenzivně využívána. Knihovna je dostupná pod MIT licencí z [8].

```
var auto = require('ardrone-autonomy');
var mission = auto.createMission();

mission.takeoff(); // Vzlétnutí dronu
mission.zero(); // Zde se nastaví počáteční bod pro let
mission.altitude(1); //Nastavení výšky letu na 1 m
mission.forward(1); //Let vpřed na vzdálenost 1m
mission.hover(100); //Zůstat na místě pod dobu 100 ms
mission.land(); //Přistát

mission.run(function (error, result) // Spuštění mise
{
  if(error)
  {
    console.trace('ERROR: %s', error.message);
    mission.client.stop();
    mission.client.land();
  }
  else
  {
    console.log('SUCCESS');
    process.exit(0); // Ukončení programu drona
  }
});
```

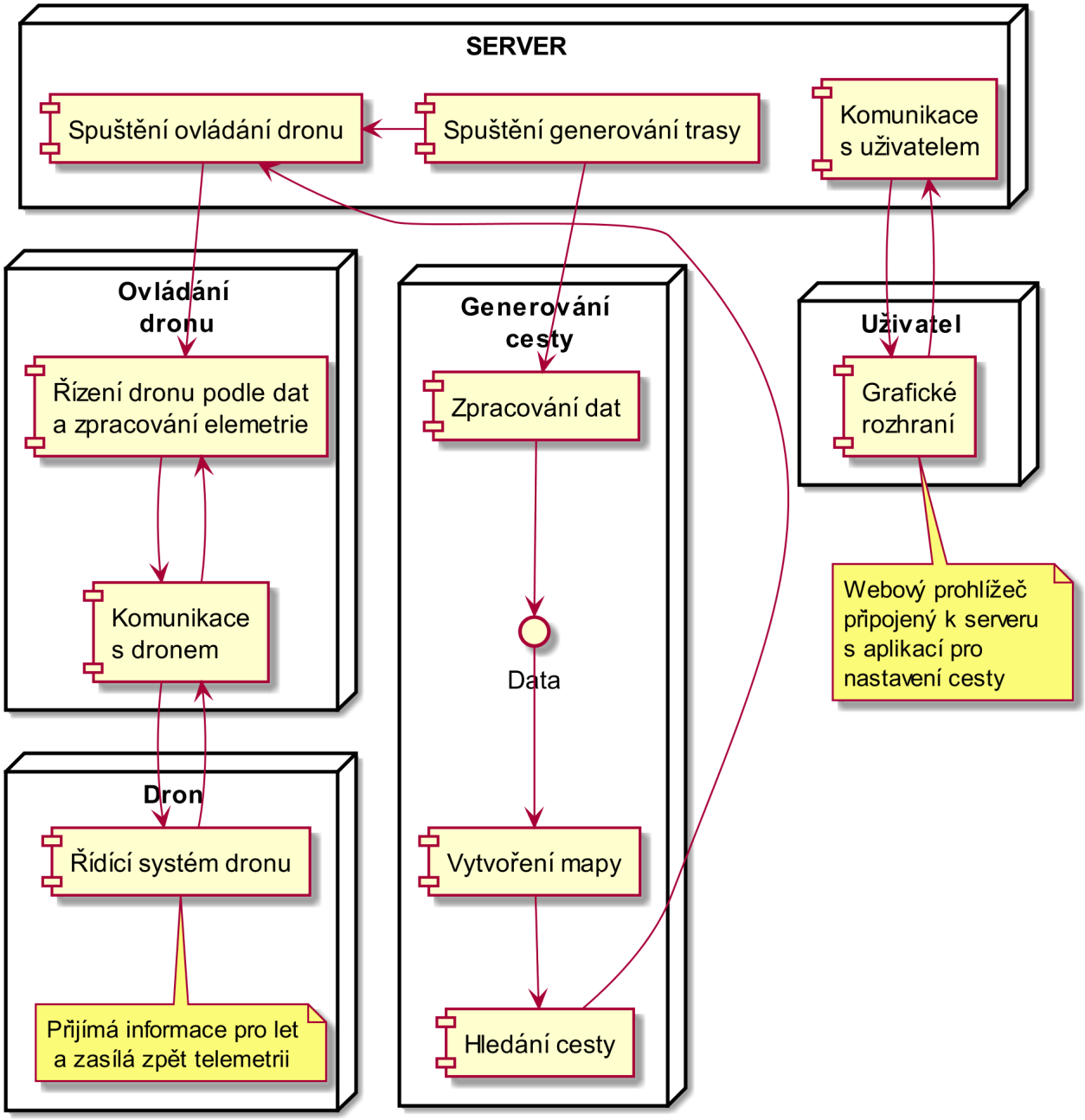
Příklad použití knihovny ardrone-autonomy

6

Implementace navrženého řešení

Během implementace řešení bylo rozhodnuto, pro vytvoření systému založeného na navigaci, podle předem uložené mapy v paměti generované uživatelem. Toto rozhodnutí bylo provedeno z důvodu vysoké komplexnosti řešení vyžadující hluboké znalosti algoritmů pro zpracování obrazu.

Navigační systém je implementován za využití programovacího jazyku TypeScript viz 5.3 a je poté překládán do jazyku JavaScript viz 5.2. Pro kompilaci TypeScriptu do JavaScriptu je využit kompilátor tsc viz 5.3. Pro implementaci je značně využíván také framework NodeJS viz 5.4. Tento software je možné dělit do tří hlavních částí: Grafické rozhraní, ovládání dronu a server.



Obr. 6.1: Diagram implementovaného řešení

6.1 Hardware systému použitý při implementaci

Systém je implementován za využití komerčně dostupného dronu AR Drone 2.0 viz 4.1 z důvodu vysoké náročnosti při samotné konstrukci dronu a tento dron poskytuje ještě mnohé další výhody pro testování i samotný provoz prototypu. Samotný navigační systém je poté implementován do mikropočítače Raspberry Pi 3 viz 4.2. Během vývoje systému bylo rozhodnuto pro sloučení navigační jednotky pro dron se základnou pro ovládání uživatelem a tento mikropočítač tedy slouží jako základna a také jako samotná řídící jednotka. Vzdálený přístup uživatele je zajištěn po připojení na wi-fi síť ve formě webové stránky a pro současné účely tedy momentálně potřeba použití externí základny odpadá. Využití externí základny by bylo vhodné v budoucnu až při potřebě automatického nabíjení dronu a plně automatického letu například jako součást většího monitorovacího systému.

6.2 Grafické rozhraní

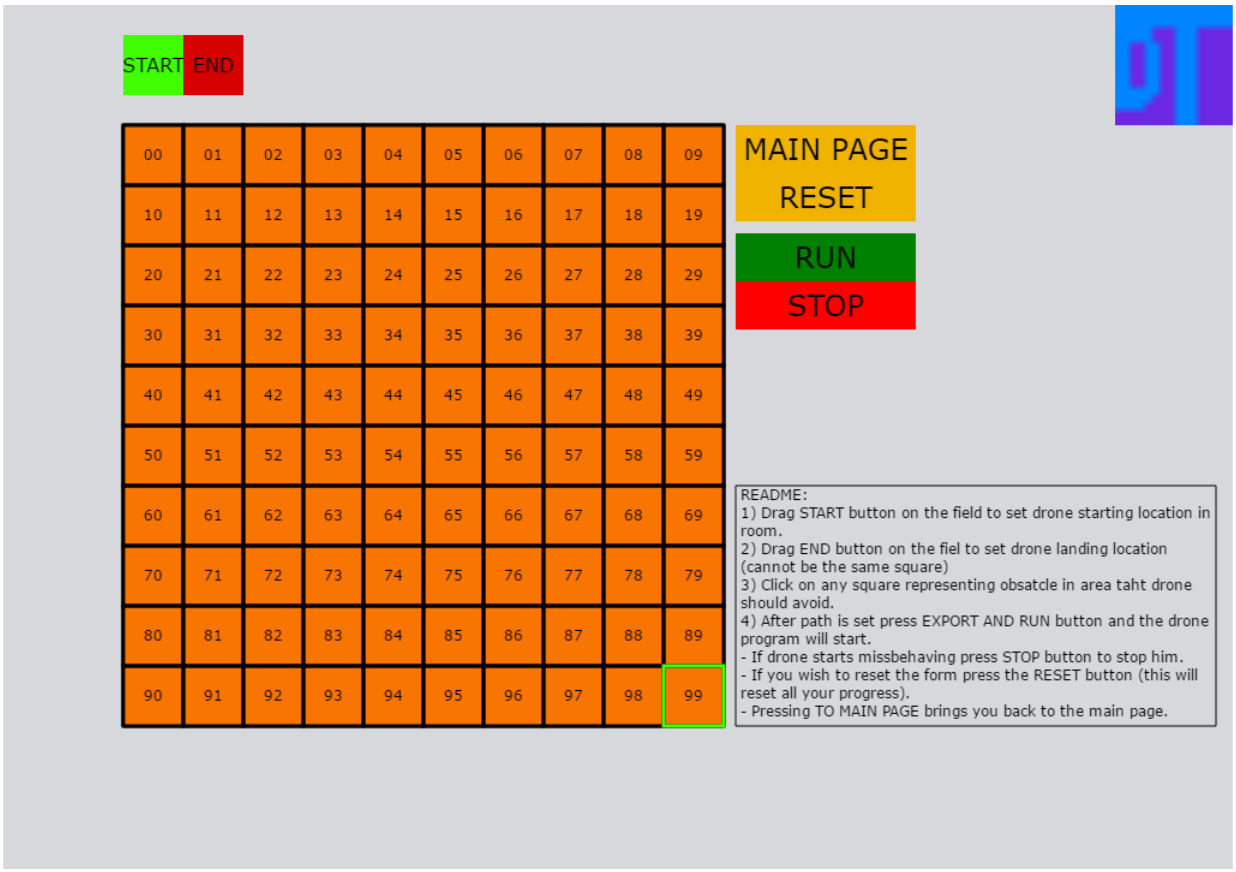
Při spuštění je jako první viditelná stránka s jednoduchým menu pro vybírání funkce. Hlavní část grafického rozhraní je na webové stránce s nastavením cesty pro dron 6.2. Tato stránka má podobu webové stránky kde v hlavní části se nachází mapa pro nastavení překážek a vedle ní jednoduché ovládací menu s popisem ovládání pod ním. Všechny prvky jsou poté postupně překreslovány při každém obnovení stránky.

V menu 6.3 se nachází 4 tlačítka 1. tlačítko slouží pro návrat na předchozí stránku, 2. tlačítko slouží pro resetování mapy uživatelem, 3. tlačítko slouží pro exportování mapy od uživatele do další části programu, která po vygenerování trasy začne vlastní let s dronem a let vykoná. Posledním tlačítkem je tlačítko stop, které zastaví dron a přistane s ním v krajním případě potřeby přerušení letu.

Nejdůležitější částí grafického rozhraní na této stránce je bezesporu mapa pro nastavení překážek pro trasu dronu. Tato mapa reprezentuje prostory ve kterých se má dron pohybovat, každý čtverec reprezentuje 1 m. Na této mapě uživatel nastaví počátek a konec cesty pro dron přetažením tlačítek *START* a *END* umístěných nad mapou na požadované místo odpovídající reálnému prostoru. Nastavení překážek v prostoru se provede jednoduše kliknutím na požadované místo v mapě odpovídající relativní pozici překážky v prostoru.

Tato mapa je implementována stejně jako celé grafické rozhraní za využití html5 viz 5.1 objektu canvas. Canvas dovoluje vykreslování do souřadnicového systému a jsou opatřeny detekcí kliknutí myší. Implementace tlačítka je dostupná v přílohách A.

Tlačítko pro export poté projde všechna tlačítka a vytvoří výstupní řetězec, který je zaslán do serveru pomocí *POST* požadavku viz 6.2, kde se vytvoří *POST* požadavek z třídy *XMLHttpRequest* a zašle se s daty. Exportování dat je dostupné v přílohách B.

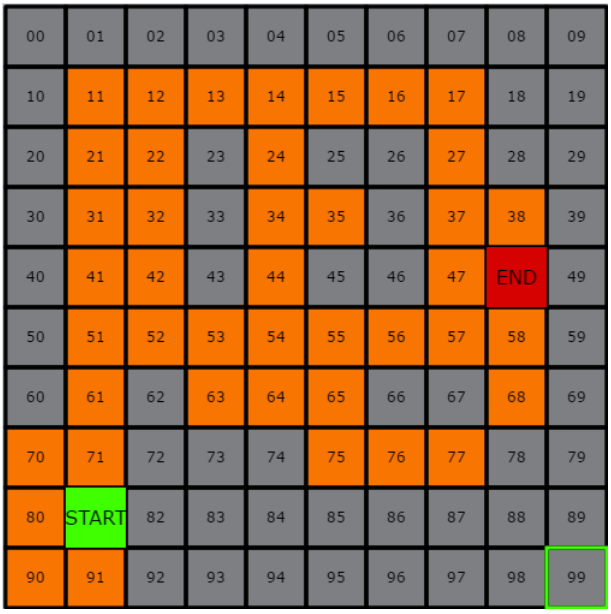


Obr. 6.2: Grafické rozhraní aplikace

Obr. 6.3: Ovládací tlačítka aplikace

```
1  /**
2   * This function sends POST request with data for server
3   *
4   * @param url target url to send data
5   * @param data data in any readable form
6   */
7  function SendData(url, data) : void
8  {
9      var request = new XMLHttpRequest();
10     request.open('POST', url, true);
11     request.send(data);
12 }
```

Vytváření *POST* požadavku a zasílání dat serveru



Obr. 6.4: Příklad nastavené trasy pro dron na mapě

Data pro server jsou ve tvaru řetězce obsahujícího znaky *U*, *D*, *E*, *S*. Tyto znaky jsou přímou reprezentací uživatelem navržené mapy.

6.3 Server

Server je implementován jako standardní NodeJS http server. Server je vytvářen pomocí třídy Server frameworku NodeJS na portu 8080 běžící na adrese *http://127.0.0.1*. Při požadavku *GET* od webového prohlížeče se odpoví na požadavek poskytnutím dat ze serveru.

```
1  if (request.method == 'GET')
2        {
3          console.log('GET');
4          fs.readFile(filename, function  (error, data)
5          {
6              console.log(error)
7              if (error)
8              {
9                  response.writeHead(500);//internal error
10                 return response.end('Cannot load index.html');
11             }
12
13             var extension = path.extname(filename)
14             console.log(extension);
15             var cType : string = getCT(extension.toString());
16             console.log(cType);
17             response.setHeader('Content-type', cType);
```

```
18              response.writeHead(200);//OK
19              response.end(data);
20          });
21      }
22  }
```

Vyřizování GET požadavku

V případě požadavku POST musí server rozeznat na kterou cestu je *POST* požadavek zaslán, což zjistí z adresy požadavku a v případě požadavku *POST* pro navigační cestu jsou přečtena a uložena do csv souboru viz 6.3 kde D reprezentuje překážku a U reprezentuje volné místo, S startovní pozici a E cílovou pozici. Po uložení dat je vytvořen nový proces, kterým je navigační program, do kterého se zašlou data. Po ukončení činnosti navigační proces zašle data zpět serveru, který do té doby čeká. Po příchodu dat vytvoří nový proces, který ovládá samotný let dronu ze zpracovaných navigačních dat.

```
1  else if(request.url == '/p')// /p route
2          {
3            var data : String = new String();
4
5            request.on('data', function (rData)
6            {
7                data += rData.toString();
8                //Uncoment for data logging
9                //console.log('RecData ' + data);
10           });
11           request.on('end', function ()
12           {
13
14               createFile(navDataFile, data);
15               console.log('Full data: \n' + data);
16
17               // calling dronecontroll
18               var dcPath = bbdir + '/nav/main.js';
19               var fork = require('child_process').fork; // start the
                     navigation process
20               var child = fork(dcPath);
21               child.send(data); // send data to navigation process
22
23               child.on('message', (navData) =>
24               {
25                 console.log('NAVDATA: ', navData);
26
27                 var dcPath2 = bbdir + '/nav/DFLY.js';
28                 var fork2 = require('child_process').fork; // start the
                       drone control
29                 var child2 = fork(dcPath2);
```

```
30                    child2.send(navData); // send data to navigation process
31                   });
32
33                  data = new String(); // clear data
34
35                  response.end();//CLOSE RESPONSE
36              });
37          }
```

Vyřizování POST požadavku a spouštění podprocesů

```
1  U U U U U U U U U U
2  U U U U D D D D D D
3  U U U U D E U U U D
4  U U U U D U D D U D
5  U U D D D U D D U D
6  U U D U U U U U U D
7  U U D U D D D D D D
8  U U D U D U U U U U
9  U U D S D U U U U U
10 U U D D D U U U U U
```

Příklad obsahu generovaného csv souboru

6.4 Generování cesty

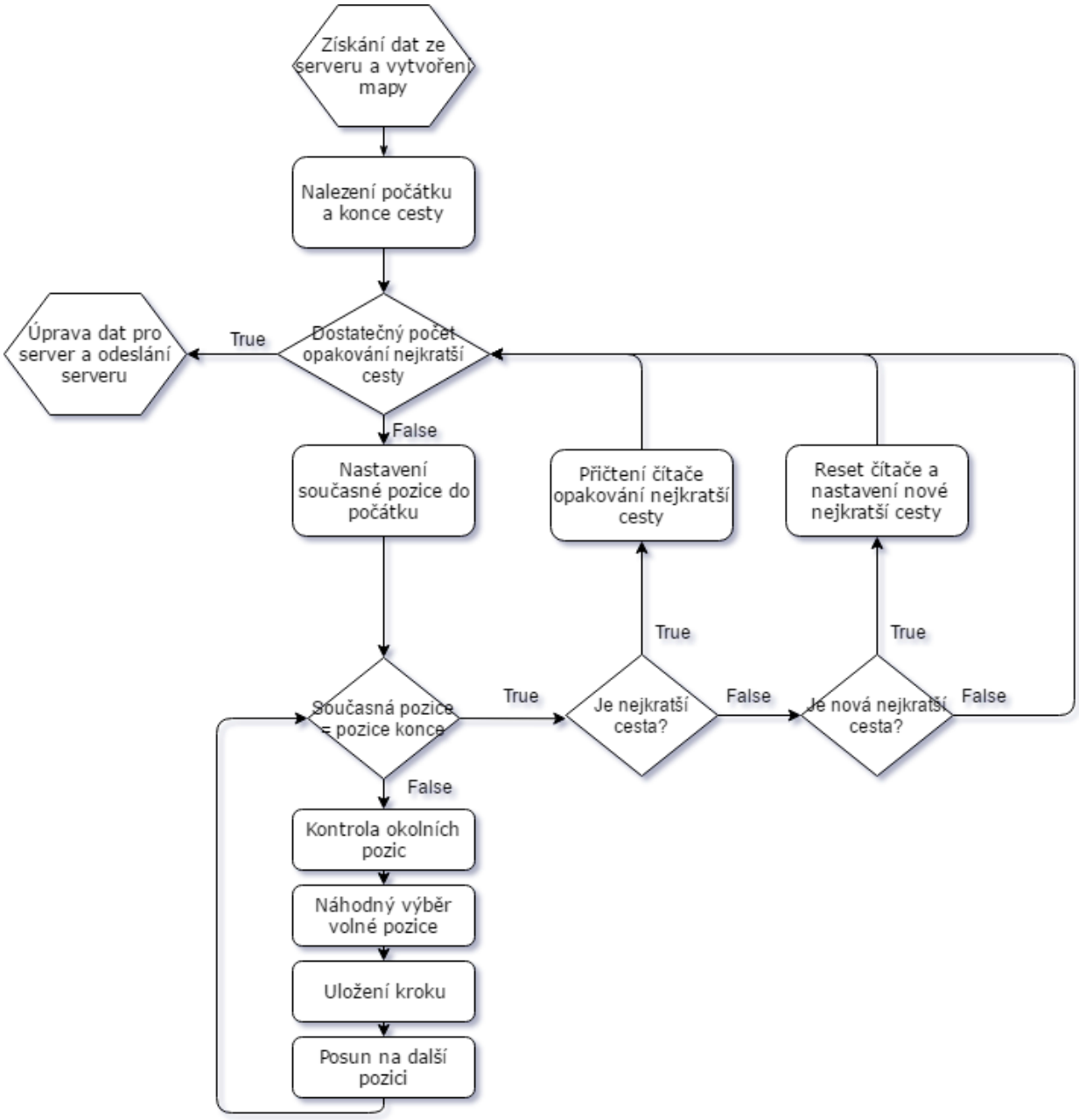
Generování cesty je možné provádět buď přímo z dat zadaných uživatelem a zaslaných serverem, nebo pomocí uložené mapy v csv souboru 6.3. Také je možné generování zpáteční cesty pomocí funkce tato cesta je, ale generována pomocí zrcadlení předchozí cesty. Generování cesty je řešeno momentálně za využití metody brute force. Tato metoda není vysoce inteligentní, ale pro tyto účely naprosto dostačuje.

Mapa pro účely navigace a hledání cesty je implementována ve formě dvourozměrného pole čísel implementace v příloze C. Standardní rozměry mapy jsou 10 x 10 kde každý prvek představuje oblast velikosti 1 m. Tato velikost je v současné době zvolena z důvodu bezpečnosti a přesnosti určení pozice dronu. Mapa je vytvářena z řetězce dat, který je získán buď z uloženého csv souboru 6.3, nebo přímo od serveru přímo od uživatele.

Samotné hledání cesty probíhá tak, že nejprve se nalezne začátek a konec cesty a uloží se jeho souřadnice v mapě. Zde, protože mapa je malých rozměrů vyhledávají se tyto souřadnice pouze pomocí procházení dvourozměrného pole. Dalším krokem je nastavení současné pozice na startovní pozici a následně se spouští samotný algoritmus vyhledávání cesty.

Všechny tyto kroky algoritmu jsou poté uloženy do pole čísel a podle jeho velikosti lze snadno zjistit počet kroků nutných pro dosažení cíle cesty implementace v příloze D. Tento algoritmus v každém kroku kontroluje, zda nedorazil do cílové pozice pomocí porovnávání souřadnic současné a cílové pozice. V případě, že nedorazil algoritmus provede rozhlédnutí po okolí současné pozice, kde zkontroluje, které pozice jsou volné a na kterých se nachází překážka, data z tohoto kroku jsou uložena. Po rozhlédnutí algoritmus zvolí náhodně jednu z volných pozic a posune současnou pozici na tuto volnou pozici s označením předchozí pozice jako využité. V případě, že neexistuje žádná volná pozice algoritmus se vrátí zpět po svých předchozích krocích. Z každého kroku jsou vygenerovány mapy a souřadnice použitých kroků. Z důvodu náhodné volby cesty je tedy nutné provádět tento proces opakovaně do nalezení nejkratší cesty.

Celý tento proces se opakuje do doby, než je splněna zastavovací podmínka algoritmu. Tato podmínka je počet opakování nejkratší sekvence musí být vyšší než 10. Tato podmínka není ideální, ale alespoň v testovacích případech se osvědčila jako dostačující. Výsledkem tohoto algoritmu je pole obsahující čísla 1, 2, 3 a 4 (vpřed, vzad, vlevo, vpravo), která značí směr kterým se má dron vydat po každém uraženém metru cesty.



Obr. 6.5: Vývojový diagram hledání cesty

6.5 Ovládání dronu

Ovládání dronu je z větší části řešeno pomocí knihovny ardrone-autonomy 5.5. Tato knihovna dovoluje přesnou kontrolu dronu a zajišťuje ovládání dronu tím, že poskytuje přístup k ovládacímu rozhraní dronu upraveném pro využití ve vysokoúrovňových aplikacích. To znamená, že tato knihovna je schopná přijmout data ve stylu určení směru a vzdálenosti letu a je schopná jej dodržet samostatně bez dalšího nutného zásahu, většina telemetrie je také momentálně využívána touto knihovnou, kde využívá informace o naklonění rotaci a otočení spolu s rychlostí dronu z jeho čidel. Tyto informace jsou využívány k odhadu pozice a korekce odhadu je poté prováděna v knihovně pomocí využití spodní kamery. Informace je také možné z knihovny získat a dále s nimi pracovat. Pomocí knihovny ar-drone 5.5 je také možné získat informace o stavu baterie, které by v budoucnu mohly sloužit pro automatický návrat na základu v případě nízkého napětí.

Data zpracovaná v podprogramu pro generování cesty jsou tedy zaslána do podprogramu pro řízení dronu pomocí serveru ve formě řetězce po tom, co je dokončeno jejich zpracování v podprogramu pro generování cesty a jsou zpracována do pole čísel. Pro každý prvek jsou vytvořeny samostatné mise, které se řídí podle vygenerovaných dat pro směr letu a je také udržována konstantní výška letu z důvodu zjednodušení aplikace. Vytváření samostatných misí bylo zvoleno z důvodu implementace nuceného zastavení dronu a zlepšení zpětné vazby od dronu, knihovna sice podporuje více prvkové mise, ale testováním bylo zjištěna lepší chování dronu v případě využití samostatných misí. Tato část softwaru se při implementaci jevila jako nejproblematičtější a dalším krokem pro úpravu je odstranění využívané knihovny a navržení vlastního řešení pro přímou kontrolu dronu, to ale nebylo z časových důvodů realizováno.

7

Popis instalace navigačního systému a potřebného software a jeho spouštění

Tato kapitola se zabývá instalací potřebných programů pro chod navigačního systému, instalací samotného navigačního systému a poté jsou vysvětleny úkony nutné pro jeho spouštění.

7.1 Instalace operačního systému na Raspberry Pi

Instalace operačního systému na Raspberry Pi je nejlepší provádět za připojení monitoru pomocí HDMI portu přímo na Raspberry Pi. Pro instalaci je potřeba upravit paměťovou kartu naformátováním jejího obsahu a poté zkopírováním NOOBS instalátoru [9] na tuto paměťovou kartu. Tento instalátor existuje v online a off-line verzi u online verze je třeba před instalací připojit Raspberry Pi k internetu pomocí dialogového okna které se zobrazí při spuštění.

7.2 Instalace Node JS

Instalace NodeJS na Raspbery Pi je velice jednoduchá a provádí se pomocí package manageru pomocí jednoduchých příkazů:

```
1 curl -sL https://deb.nodesource.com/setup_7.x | sudo -E bash -
2 sudo apt-get install -y nodejs
```

Po instalaci NodeJS je ještě vhodné nainstalovat nástroje pro kompilaci:

```
1 sudo apt-get install -y build-essential
```

7.3 Instalace navigačního software

Instalace navigačního software na PC je velice jednoduchá vyžaduje pouze rozbalení archivu s navigačním softwarem do kořenového adresáře disku. Stejně tak instalace na Raspberry Pi probíhá obdobně rozbalením archivu v kořenovém adresáři.

7.4 Spouštění aplikace a její používání

Aplikaci je potřeba spustit pomocí dávkového souboru (*run.bat* pro PC a *run.sh* pro Raspberry Pi) v kořenovém adresáři aplikace. Tímto se spustí server, který bude mít na starosti veškerou komunikaci s uživatelem a kontrolu dronu. Samotnou aplikace je poté možné spustit ve webovém prohlížeči na adrese *http://127.0.0.1:8080* v případě přístupu do aplikace přímo ze zařízení pro ovládání (standardně v případě použití PC pro kontrolu), nebo v případě zařízení přímo na dronu přístupem přes adresu zařízení na síti (standardně *http://192.168.1.2:8080* pokud se připojujeme k hotspotu vytvářeného dronem, popřípadě jinou přidělenou adresou pokud je připojován dron k jiné wi-fi síti.) Pro připojení k hotspotu dronu na Raspberry Pi popřípadě jiné wi-fi síti je potřeba upravit soubor *wpa_supplicant.conf*. Toto nastavení je nutné na řídícím Raspberry Pi pouze jednou.

```
1 sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Příkaz pro upravení souboru pro připojení k wi-fi dronu

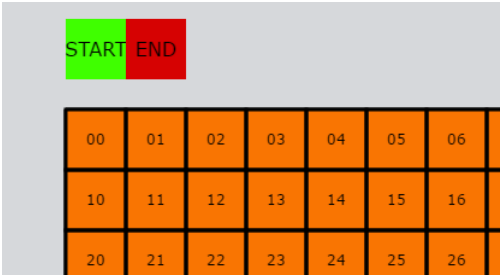
```
1 network={
2     ssid="ssid sítě pro připojení(u dronu ARdrone)"
3     psk="heslo k síti v případě dronu lze vynechat"
4 }
```

Text vložený do souboru

Samotné ovládání aplikace je po připojení vysoce jednoduché při otevření webové adresy se zobrazí úvodní obrazovka ze které se po přepnutí dostaneme na adresu *http://(ip-adresa):8080/p*. Na této stránce běží řídící aplikace pro ovládání dronu.

V prvním kroku je třeba nastavit počátek a konec cesty pro dron. Toto nastavení se provádí pomocí tlačítek *START* a *END* 7.1 a umístit dron na startovní pozici relativně v prostoru pro navigaci. Nastavení se provede prostým přetažením na startovní a cílové souřadnice na kontrolní mapě pod tlačítky 6.4. Po nastavení začátku a konce cesty je potřeba umístit na mapě překážky, kterým se má dron vyhnout to se provede jednoduchým kliknutím na jednotlivé prvky na mapě, které změní barvu pro signalizaci překážky. V případě potřeby změnit překážku na volný prostor stačí kliknutí opakovat na stejný

prvek nebo v případě potřeby kompletního resetování stisknout tlačítko *RESET*. Pro spuštění programu slouží tlačítko *RUN*, toto tlačítko zašle serveru data z mapy pro zpracování a po zpracování ihned spustí program ovládající dron, ten začne vykonávat cestu. Případě nutného přinuceného zastavení je třeba stisknout tlačítko *STOP*. Toto tlačítko provede nucené zastavení, pro vrácení k běžnému programu je poté potřeba kompletní restart serveru a resetování dronu provedené odpojením a připojením napájení. Po tomto procesu je možné pokračovat s novým nastavením mapy.



Obr. 7.1: Tlačítka START A END pro nastavení cesty

8

Návrh dalšího rozvoje řídícího systému

Tato kapitola se zabývá návrhem dalšího rozvoje navigačního systému pro autonomní dron, jak ve stránce hardwarové kdy jsou popsány další typy čidel a další prvky pro zlepšení funkčnosti a přesnosti systému. Stejně tak je navržen způsob zlepšení softwaru systému do pro budoucí účely.

8.1 Vylepšení hardwaru systému

Hardware současného implementovaného systému jak dron, tak řídící mikropočítač vyhovují zadání, ale pro budoucí využití by bylo vhodné vybavit dron dalšími senzory. Nabízí se zejména laserové měřiče vzdálenosti pro určení přesné vzdálenosti od překážek a také pro přistávání. Pro přistávání by byla vhodná přistávací platforma se základnou, která by byla schopná bezdrátově automaticky dron nabíjet tak, aby byl vždy připraven pro další let. Samotný dron by mohl být upraven pro jednodušší připevnění navigačního kontroléru, popřípadě by dron mohl být nahrazený dronem nové konstrukce kde by se využívalo jednoho kontroléru pro navigaci i řízení motorů. Samotné uživatelské rozhraní by poté bylo možné ovládat ze základny, která by byla připojena k internetové síti a dron by byl poté ovládán pomocí rádiového ovládání, které je bezpečnější než v případě využití wi-fi a dosahuje podstatně vyšší vzdálenosti.

Nabízí se také možnost dovybavit dron dalšími kamerami a výkonnějším kontrolérem, kde by se provádělo zpracování přímo na dronu, popřípadě využití většího množství dronů a rozdělení práce mezi nimi podobně jako na výpočetním clusteru, kde by jeden dron působil jako řídící pořizovací data a ostatní by sloužily jako přidaný výpočetní výkon. Celý takovýto systém by poté mohl provádět i složitější úkony.

8.2 Vylepšení navigačního softwaru systému

V prvním kroku vylepšení software do budoucna je třeba odstranění využívané knihovny ardrone-autonomy. Tato knihovna momentálně nahrazuje nutnost provádění výpočtů a určování přesné pozice nicméně způsobuje problémy s integrací do softwaru a dále již není vyvíjena. Nahrazení této knihovny je, ale velice problematické a neexistuje žádná alternativa kompatibilní s tímto softwarem, a proto by bylo potřeba vytvoření vlastní implementace přesného určování pozice což je velice náročné.

Pro využití jiného typu dronu je také možné vytvoření nových knihoven pro navigaci s jiným dronem, kde je třeba pouze využívat data zpracovaná serverem pro let a vytvořit pouze ovládací program pracující s těmito daty.

V současné době nejlepším vylepšení softwaru by bylo zlepšení získávání zpětné vazby od dronu, popřípadě přechod na kompletní systém v reálném čase. Dalším podstatným vylepšením by byla implementace detekce překážek pomocí čelní kamery a reakce dronu na překážku vyhnutím se. Spolu s tímto by bylo možné přejít zpět ze zjednodušeného 2D problému na 3D problém. Dalším možným vylepšením je přechod z mapy generované uživatelem na mapu generovanou dronem pomocí kamery. Zde by bylo potřeba nasnímat prostor a poté tyto informace převést do mapy. Popřípadě ponechat mapu zadávanou uživatelem, ale upravit ji do 3D s možností nastavení výšek překážek. V neposlední řadě by systém mohl zobrazovat přesnou polohu dronu na mapě u uživatele.

9

Závěr

Tato práce se zabývá řídícím systémem pro autonomní navigaci dronu v prostoru. Práce je členěna do části návrhové, kde je proveden návrh takovéhoto systému a části implementační, kde je provedena implementace takovéhoto systému a je vytvořen prototyp. V první části práce byla provedena stručná rešerše existujících systémů a způsobů navigace ve 3D prostoru.

V návrhové části je navržen potřebný hardware pro realizaci jako je dron a mikropočítač. Je navržen systém pro navigaci podle mapy zadané uživatelem a jsou také popsány jiné možnosti implementace jako je systém s využitím kamer a mapování pro zjištění pozice v prostoru.

V implementační části je popsán použitý hardware dron AR Drone 2.0 a mikropočítač Raspberry Pi, a jako senzory je využíváno základní vybavení dronu AR Drone 2.0 což jsou kamery a ultrazvukový senzor. Pro realizaci byl využíván programovací jazyk TypeScript a JavaScript. V této části je také popsána architektura řídící aplikace a způsob jejího používání.

Systém je implementován jako systém navigující podle mapy nastavené uživatelem a uložené v paměti a je implementován na principu serverové aplikace kde se uživatelské rozhraní spouští v internetovém prohlížeči vzdáleně. V tomto rozhraní uživatel nastaví danou mapu a s počátkem a cílem cesty. Po tomto nastavení server spustí podprogram, který provede určení cesty pro dron a tyto data předá dalšímu podprogramu, který poté provede samotnou kontrolu dronu.

Samotný přenos dat je řešen pomocí samotné funkčnosti aplikace, která funguje jako server a přenos dat se řeší pomocí wi-fi sítě generované dronem a lze jej také řešit pomocí existující wi-fi sítě po připojení dronu a serveru. Základna byla po návrhu sloučena dohromady s navigačním mikropočítačem a veškerou komunikaci provádí uživatel přímo s ním a základna byla tedy shledána jako nepotřebná. Potřeba základy by se začala objevovat až v případě potřeby automatického nabíjení a plně automatického provozu bez zásahu uživatele.

Literatura

[1] A. Hussein, A. Al-Kaff, A. de la Escalera and J. M. Armingol. *"Autonomous indoor navigation of low-cost quadcopters"*, Service Operations And Logistics, And Informatics (SOLI), 2015 IEEE International Conference on, Hammamet, 2015, pp. 133-138. doi: 10.1109/SOLI.2015.7367607

[2] T. T. Mac, C. Copot, A. Hernandez and R. De Keyser. *"Improved potential field method for unknown obstacle avoidance using UAV in indoor enviroment"*, 2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, 2016, pp. 345-350. doi: 10.1109/SAMI.2016.7423032

[3] L. V. Santana, A. S. Brando, M. Sarcinelli-Filho and R. Carelli. *"A trajectory tracking and 3D positioning controller for the AR.Drone quadrotor"*, Unmanned Aircraft Systems (ICUAS), 2014 Internation Conference on, Orlando, FL, 2014, pp. 756-767. doi: 10.1109/ICUAS.2014.6842321

[4] A. Hornung at al. *"OctoMap: an efficient probabilistic 3D mapping framework based on octrees"*, Autonomous Robots, vol. 34, num. 3, 2013, pp. 189-206. doi: 10.1007/s10514-012-9321-0

[5] F. Cocchioni, A. Mancini and S. Longhi. *"Autonomous navigation, landing and recharge of quadrotor using artificial vision"*, Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, Orlando, FL, 2014, pp. 418-429. doi: 10.1109/ICUAS.2014.6842282

[6] TYPESCRIPT *TypeScript*[online][Cit. 9.5.2017]. Dostupné z WWW: http://www.typescriptlang.org/

[7] AR-DRONE (github) *ar-drone*[online][Cit. 9.5.2017]. Dostupné z WWW: https://github.com/felixge/node-ar-drone

[8] ARDRONE-AUTONOMY (github) *ardrone-autonomy*[online][Cit. 9.5.2017]. Dostupné z WWW: https://github.com/eschnou/ardrone-autonomy

[9] NOOBS *NOOBS*[online][Cit. 9.5.2017]. Dostupné z WWW: https://www.raspberrypi.org/downloads/noobs/

[10] TYPESCRIPT *TypeScript basic types*[online][Cit. 9.5.2017]. Dostupné z WWW: https://www.typescriptlang.org/docs/handbook/basic-types.html

[11] ARDRONE-AUTONOMY (github) *ardrone-autonomy*[online][Cit. 9.5.2017]. Dostupné z WWW: https://github.com/AutonomyLab/ardrone_autonomy

[12] HTML5 CANVAS TUTORIALS *HTML5 canvas tutorials*[online][Cit. 9.5.2017]. Dostupné z WWW: http://www.html5canvastutorials.com/

[13] NODECOPTER CORE *Nodecopter*[online][Cit. 9.5.2017]. Dostupné z WWW: http://www.nodecopter.com/hack#npm-modules

[14] COMPUTER VISION GROUP *Visual Navigation for Flying Robots*[online][Cit. 9.5.2017]. Dostupné z WWW: http://vision.in.tum.de/teaching/ss2013/visnav2013

[15] ROS.ORG *tum_ardrone*[online][Cit. 9.5.2017]. Dostupné z WWW: http://wiki.ros.org/tum_ardrone

[16] TUM_ARDRONE (github)*tum_ardrone*[online][Cit. 9.5.2017]. Dostupné z WWW: https://github.com/tum-vision/tum_ardrone

[17] LAURENT ESCHENAUER *Advanced programming with #nodecopter*[online][Cit. 9.5.2017]. Dostupné z WWW: https://www.slideshare.net/eschnou/20130807-advanced-programming-with-nodecopter

Příloha A

Implementace tlačítka v mapě

```
1  class Button implements IDrawable
2  {
3      public context : CanvasRenderingContext2D;
4      public canvas : HTMLCanvasElement;
5      public down : boolean;
6      public x : number;
7      public y : number;
8      public width : number;
9      public height : number;
10     public color : string
11     public halfW : number;
12     public halfH : number;
13     public text: string;
14     public fontSize: number;
15     private prevColor : string;
16
17     constructor(context : CanvasRenderingContext2D, canvas :
           HTMLCanvasElement, x : number, y : number, width : number, height
            : number, color : string, text : string, fontSize : number = 32
           )
18     {
19         this.context = context;
20         this.canvas = canvas;
21         this.x = x;
22         this.y = y;
23         this.width = width;
24         this.height = height;
25         this.color = color;
26         this.halfW = width / 2;
27         this.halfH = height / 2;
28         this.text = text;
29         this.fontSize = fontSize;
30         this.prevColor = color;
31         this.down = false;
```

```
32          this.canvas.addEventListener("mousedown", this.mouseDown, false)
                ;
33      }
34      public draw = () : void =>
35      {
36          this.context.save();
37          this.context.beginPath();
38          this.context.textAlign = "center";
39          this.context.textBaseline = "middle";
40          this.context.fillStyle = this.color;
41          this.context.font = this.fontSize + "px Verdana";
42          if (this.down == true)
43          {
44              this.color = "blue";
45              this.context.fillText(this.text, this.x + 2, this.y + 2);
46          }
47          else
48          {
49              this.color = this.prevColor;
50              this.context.fillText(this.text, this.x, this.y);
51          }
52          this.context.restore();
53          this.context.save();
54          this.context.lineWidth = 2;
55          this.context.strokeStyle = this.color;
56          this.context.fillStyle = this.color;
57          if (this.down == true)
58          {
59              this.context.globalAlpha = 0.5;
60              this.context.fillRect(this.x - this.halfW + 2, this.y - this
                    .halfH + 2, this.width, this.height);
61          }
62          else
63          {
64              this.context.fillRect(this.x - this.halfW, this.y - this.
                    halfH, this.width, this.height);
65          }
66          this.context.stroke();
67          this.context.restore();
68      }
69      public mouseDown = (event: MouseEvent) : void =>
70      {
71          var x: number = event.x - this.canvas.offsetLeft;
72          var y: number = event.y - this.canvas.offsetTop + window.
                pageYOffset;
73
74          if (x > this.x - this.halfW && y > this.y - this.halfH &&
75              x < this.x + this.halfW && y < this.y + this.halfH)
```

```
76          {
77              //switch button state to up/down
78              if(this.down == false)
79              {
80                  this.down = true;
81              }
82              else
83              {
84                  this.down = false;
85              }
86          }
87      }
88      //force button into NOT pressed state
89      public setUP = () =>
90      {
91          this.down = false;
92      }
93      //force button into pressed state
94      public setDOWN = () =>
95      {
96          this.down = true;
97      }
98  }
```

Příloha B

Exportování dat z mapy pro server

```
1  function exportButton(UI : ControlButtonUI, grid : ControlGrid,
       targetList : Array<DropTarget>, targeGrid : TargetGrid) : boolean
2  {
3      var exportButton : BButton = UI.getExportButton();
4      var bGrid : BtnGrid2 = grid.getButtonGrid();
5      var bkarray : Array<Array<Button>> = new Array<Array<Button>>();
6      var strtX : number = 0;
7      var strtY : number = 0;
8      var endX : number = 0;
9      var endY : number = 0;
10
11     for(var i : number = 0; i < targetList.length; i++)
12     {
13         if(targetList[i].getIsOverStart() == true)
14         {
15            var abPos = i;
16            strtX = abPos % targeGrid.getButtonLineCount();
17            strtY = parseInt((abPos / targeGrid.getButtonLineCount()).
                 toString());
18         }
19     }
20     for(var i : number = 0; i < targetList.length; i++)
21     {
22         if(targetList[i].getIsOverEnd() == true)
23         {
24             var abPos = i;
25             endX = abPos % targeGrid.getButtonLineCount();
26             endY = parseInt((abPos / targeGrid.getButtonLineCount()).
                  toString());
27         }
28     }
29     if(exportButton.clicked == true)
30     {
31         bkarray = bGrid.getButtonArray();
```

```
32          var outputString : String = new String();
33          for(var i : number = 0; i < bGrid.getButtonArray().length; i++)
34          {
35              for(var j : number = 0; j <  bkarray[i].length ; j++)
36              {
37                  if(strtX == j && strtY == i)
38                  {
39                      outputString += 'S;';
40                  }
41                  else if(endX == j && endY == i)
42                  {
43                      outputString += 'E;';
44                  }
45                  else
46                  {
47                      if(bkarray[i][j].down == true)
48                      {
49                          outputString += 'D;';
50                      }
51                      else if(bkarray[i][j].down == false)
52                      {
53                          outputString += 'U;';
54                      }
55                  }
56              }
57              outputString += '\n';
58          }
59          exportButton.clicked = false;
60          SendData(serverURL, outputString);
61          return true;
62      }
63      return false;
64  }
```

Příloha C

Implementace mapy pro ukládání dat od uživatele

```
1  import { FileHelper } from "./FileHelper";
2
3  export class DMap
4  {
5      private xSize : number = null;
6      private ySize : number = null;
7      private map : Array<Array<number>> = null;
8      private text : string = null;
9
10     constructor(xSize : number, ySize : number)
11     {
12         this.xSize = xSize;
13         this.ySize = (ySize * 2) + 1;
14     }
15     public createMapFromText = (text : string) : void =>
16     {
17         this.text = text;
18         var cnt = 0;
19         this.map = new Array<Array<number>>();
20         var Hmap : Array<number> = new Array<number>();
21
22         for(var i : number = 0; i < this.xSize; i++)
23         {
24             Hmap = new Array<number>();
25             for(var j : number = 0; j <  this.ySize; j++)
26             {
27                 if(this.text.charAt(cnt) == 'D')
28                 {
29                     Hmap.push(1);
30                     cnt++;
31                 }
32                 else if(this.text.charAt(cnt) == 'U')
```

```
33                  {
34                      Hmap.push(0);
35                      cnt++;
36                  }
37                  else if(this.text.charAt(cnt) == 'S')
38                  {
39                      Hmap.push(5);
40                      cnt++;
41                  }
42                  else if(this.text.charAt(cnt) == 'E')
43                  {
44                      Hmap.push(8);
45                      cnt++;
46                  }
47                  else
48                  {
49                      cnt++;
50                  }
51
52              }
53              this.map.push(Hmap);
54          }
55      }
56
57      public getArrayClone = () : Array<Array<number>> =>
58      {
59          var clonedArray : Array<Array<number>> = new Array<Array<number
                >>();
60          for(var i : number = 0; i < this.xSize; i++)
61          {
62              clonedArray[i] = this.map[i].slice(0, this.ySize);
63          }
64
65          return clonedArray;
66      }
67
68      public getMapArray = () : Array<Array<number>> =>
69      {
70          return this.map;
71      }
72
73      public setMapArray = (newMapArray : Array<Array<number>>) : void =>
74      {
75          this.map = newMapArray;
76      }
77
78      public getXSize = () : number    =>
79      {
```

```
80          return this.xSize;
81      }
82
83      public getYSize = () : number    =>
84      {
85          return this.ySize;
86      }
87
88  }
```

Příloha D

Implementace hledání cesty v mapě

```
1  private iteratePath = () =>
2  {
3          var mapStepList : Array<Array<Array<number>>> = new Array<Array<
               Array<number>>>();
4          var directionData : Array<number> = new Array<number>();
5          var iterationCounter : number = 0;
6          var sPos : Array<number> = new Array<number>();
7          var ePos : Array<number> = new Array<number>();
8
9          sPos = this.findSTART();
10         ePos = this.findEND();
11
12         var currPos : Array<number> = sPos.slice(0, 2);
13         var nextPos : Array<number> = new Array<number>();
14         var surroundings : Array<number> = new Array<number>();
15         var chosenPath : number = null;
16         var nextStep : DMap = new DMap(this.map.getXSize(), this.map.
               getYSize());
17
18         nextStep.setMapArray(this.map.getArrayClone());
19         var prevStep : DMap = new DMap(this.map.getXSize(), this.map.
               getYSize());
20         prevStep.setMapArray(this.map.getArrayClone());
21         mapStepList.push(prevStep.getMapArray());
22
23         while(this.checkIfEnd(currPos, ePos) == false)
24         {
25             iterationCounter++;
26
27             surroundings = this.lookAround(currPos, nextStep);
28
29             chosenPath = this.chosePath(surroundings);
30
31             directionData.push(chosenPath);
```

```
32
33              nextPos = this.getNextPos(currPos, chosenPath);
34              prevStep.setMapArray(nextStep.getArrayClone());
35              nextStep = this.movePos(currPos, nextPos, prevStep);
36              currPos = nextPos;
37              mapStepList.push(nextStep.getMapArray());
38          }
39          return {
40                      iterationCounter : iterationCounter,
41                      mapStepList : mapStepList,
42                      directionData : directionData
43          };
44  }
```