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 2018

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2017/2018**

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: **10. října 2017**

Termín odevzdání diplomové práce: **24. května 2018**

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

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 10. října 2017

Abstrakt

Tato diplomová práce se zabývá návrhem řídícího systému pro autonomní dron. V teoretické části je vysvětlen pojem autonomie a jsou obecně popsány možné funkce takovéhoto systému dále je pak proveden návrh systému pro specifické zadání. Jako dron je využit dron AR Drone 2.0, který je ovládán pomocí serverové aplikace zpracovávající data od uživatele a řídící tento dron. Praktická část se zaměřuje na implementaci navigačního systému pro navigaci z mapy zadané uživatelem a uložené v paměti. Je vysvětlena realizace serveru, funkčnost a implementace webové aplikace pro zadávání mapy uživatelem a způsob ovládání dronu. Implementace je prováděna v jazycích JavaScript a TypeScript za využití webového frameworku NodeJS pro server, jazyka HTML5 pro grafické rozhraní a SDK pro ovládání dronu.

Klíčová slova

Dron, Navigační systém, Mapa, JavaScript, TypeScript, NodeJS, JSON, Server

Abstract

Pauli, Dominik. *The Control system for autonomous drone* [*Řídící systém pro autonomní dron*]. Pilsen, 2018. 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 autonomy is explained and there is described functionality for similar system also a proposal of system is made for specific assignment. As drone the AR Drone 2.0 is used which is controlled using server application processing data from user and controlling this drone. Practical part of thesis deals with implementing the navigation system from a map set by the user and saved in memory. Realization of server is explained, functionality and implementation of web application for map input from user is described as well as the drone control. Implementation is made using programming languages JavaScript and TypeScript using web framework NodeJS for server, language HTML5 is used for graphical interface and SDK is used for drone control.

Keywords

Drone, Navigation system, Map, JavaScript, TypeScript, NodeJS, JSON ,Server

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 20. května 2018

Bc. Dominik Pauli

.....

Podpis

Obsah

Seznam obrázků

Seznam symbolů a zkratek

A	Ampér
V	Volt
GB	Giga byte
MHz	Mega hertz
HDD	Hard disc
ftp	File transfer protocol
csi	Camera Serial Interface
dsi	Display Serial Interface
sdk	Software developement kit
Pa	Pascal
mAh	mili ampér hodin
HD	High definition
PID	Proporcionálně integračně derivační
HTTP	Hyper Text Protocol

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 a je vysvětlena problematika autonomie, jsou představeny způsoby využití a používání autonomie v praxi společně s členěním druhů autonomie. Druhá část práce je zaměřena na popis návrhu takovéhoto systému pro autonomní dron a je proveden návrh systému pro budoucí implementaci. Třetí část se zabývá popisem návrhu, 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, také jsou popsány problémy při implementaci systému a nutné změny designu pro jeho implementaci. Z důvodu velice dlouhého vývojového cyklu je do hloubky popsána a provedena pouze implementace prvního prototypu, která řeší některé body navrženého systému a slouží jako start pro další vývoj výsledné aplikace a jako způsob hledání problémů při budoucím vývoji výsledné aplikace. Tyto problémy jsou do hloubky popsány spolu s návrhem jejich řešení.

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. Také bylo provedeno vysvětlení základních pojmů autonomie a představeny problémy z praxe. Pro předvedení této problematiky do praxe byl navržen navigační systém pro autonomní dron s navigací podle statické mapy zadané uživatelem. Byl proveden detailní návrh systému a jeho architektury. Toto řešení bylo poté implementováno v pozměněné formě jako serverová aplikace a byl vytvořen prototyp tohoto systému pro testovací účely.

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. S využitím ovládání pomocí serverové aplikace by také systémy tohoto druhu mohly fungovat s větším množství dronů popřípadě jako roj dronů kde hlavní dron funguje jako server a ovládá ostatní.

2

Existující řešení

Tato část práce se zabývá popisem existujících způsobů realizace navigačních systémů pro drony a také popisem komerčních systémů zabývajících se podobnou problematikou. Jsou také popsány publikace zabývající se podobnou problematikou.

2.1 Existující podobná řešení navigačního systému

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. Využití pohybových dat je vysoce komplexní záležitostí a vyžaduje vysoký výpočetní výkon spolu se složitými algoritmy.

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. Také se zabývá určováním pozice dronu z doby letu a jeho rychlosti pouze odhadem.

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 je vysoce komplexní a sofistikovaná, 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.

Obecně lze říci, že tato problematika je řešena poměrně rozsáhle a drony a jejich

navigace jsou a budou v budoucnu využívány stále více a více. Existuje vysoké množství dostupných materiálů zabývajících se touto problematikou a zde byla zmíněna pouze hrstka existujících materiálů. Pro hlubší nastudování je zapotřebí rozsáhlého vyhledávání a zhodnocení existujících řešení podle konkrétních potřebných parametrů, protože tato problematika je tak rozsáhlá není možné uvádět veškerá řešeni.

3

Autonomie

Tato kapitola se zabývá problematikou autonomie a jejím rozdělením na několik stupňů. Také jsou zde představeny způsoby implementace a oblasti využití jednotlivých stupňů na teoretické úrovni.

3.1 Základy autonomie

Autonomie jako taková je v současné době velmi skloňované téma, zejména v automobilovém průmyslu, kde jsou také v současné době největší pokroky ve vývoji autonomních zařízení. Autonomní zařízení vyžadují vysoký výpočetní výkon pro zpracování dat, nicméně toto není jediný problém. Mezi další problémy patří samotné získávání dat což se často řeší kombinací senzorů různých veličin. Mezi takovéto veličiny patří vzdálenost pro jejíž měření se často využívá kombinace laserových radarů a ultrazvukových senzorů. Pro přesné určení pozice vůči překážkám se pak využívá videozáznam v kombinaci s analýzou obrazu. Tyto senzory poskytují data prakticky okamžitě, nicméně zde nastává problém s jejich analýzou neboť množství takovýchto dat může být až několik petabytů za vteřinu a je potřeba je zpracovat okamžitě s co nejnižší časovou prodlevou a ihned zareagovat na výsledek tohoto zpracování. Tato data se využívají pro zpřesnění ovládání, samotná navigace je poté většinou řešena pomocí satelitních navigačních systémů jako je GPS. Tato zařízení jsou velmi často vybavena komunikačními rozhraními schopnými komunikovat s ostatními podobnými zařízeními a mohou tedy korigovat svoji pozici oproti ostatním zařízením.

Problematika autonomie a navigace v uzavřeném prostoru je oproti těmto technikám odlišná zejména tím, že vzdálenosti, na které se zařízení pohybuje jsou poměrně malé a satelitní navigace tedy nepřipadá v úvahu. Ve známém prostředí se nabízí možnost generování mapy prostoru a její následné využití pro samotnou navigaci spolu s korigováním přesné polohy pomocí senzorů. Pro navigaci v neznámém prostředí je zapotřebí nejprve tuto mapu vytvořit což vyžaduje nasnímání okolního prostoru. Samotný pohyb ve známém prostředí je možné provádět bez jakéhokoliv využití senzorů a využívat senzory pouze pro neočekávané situace což vysoce snižuje vyžadovaný výpočetní výkon, nároky

na zpracování dat a jejich objem.

3.2 Druhy autonomie

Autonomie se rozděluje na několik stupňů, kde stupně určují do jaké míry je možné aby zařízení pracovalo bez zásahu člověka do jeho činnosti. Pro vysvětlení jsou využity příklady z automobilového průmyslu. V automobilovém provozu se například využívá rozdělení do pěti úrovní autonomie.

Nultý stupeň autonomie, zde se ještě nejedná o autonomní zařízení, jako příklad je možné použít tempomat, kde uživatel nastaví rychlost a automobil je schopný tuto rychlost sám udržovat. Toto není autonomie neboť systém sám neprovádí žádná rozhodnutí, ale pouze vykonává příkaz daný uživatelem.

První stupeň autonomie, zde se již jedná o autonomní systém, ale je zde stále vyžadován vstup od uživatele. Jako příklad zde poslouží adaptivní tempomat, kde je toto zařízení schopné zareagovat na pomalu jedoucí automobil před sebou a snížit samovolně rychlost. Zde je nutné podotknout, že samotné ovládání stále ještě provádí sám uživatel.

Druhý stupeň autonomie je již schopný samostatného ovládání pokud jsou splněny specifické podmínky. Tyto zařízení jsou schopná se sama udržet na vozovce a reagovat na vozidla jedoucí ve stejném směru a upravovat svoji rychlost. Tyto zařízení stále ještě nejsou považována za plně autonomní a jako hlavní bezpečnostní prvek je stále využíván zásah uživatele, který musí být schopný okamžitě převzít kontrolu a mít neustálý přehled nad chováním vozu.

Třetí stupeň autonomie, zde je zařízení schopné samostatně monitorovat okolní prostředí a dynamicky reagovat na jeho změny. U této úrovně je vyžadováno aby uživatel reagoval na jakýkoliv požadavek systému na převzetí kontroly, který nastane v případě nerozhodnosti systému.

Čtvrtý stupeň autonomie je případ, kdy je zařízení schopné provádět většinu úkonů řízení samostatně. U tohoto stupně je stále možnost aby uživatel převzal plnou kontrolu a také je možné požadovat vstup od uživatele v případě, že jsou nevyhovující podmínky pro bezpečnou funkci systému nebo senzorů.

Pátý stupeň autonomie je stav, kdy pouze stačí zadat cílovou lokaci a systém je sám schopný dostat se na místo bez jakéhokoliv vstupu od uživatele. Jakýkoliv systém vyžadující zásah od uživatele není schopen pátého stupně autonomie.

4

Návrh řešení navigace

Tato kapitola se zabývá návrhem řešení navigačního systému pro autonomní dron. Navigace u tohoto typu systému je řešena pouze v předem známém prostoru. Samotný prostor je systému zadán předem uživatelem pomocí grafického rozhraní. Systém je poté schopen autonomně vyhledat optimální cestu v zadané mapě podle zadané cílové pozice a provést samotné ovládání dronu.

Pro zjednodušení problému byl systém navržen tak, aby se 3D problém převedl na problém 2D. Zde se tedy uvažuje o fixní výšce letu, toto omezení bylo provedeno z důvodu snížení komplexnosti řešení algoritmů pro prototypový systém.

4.1 Vize systému

Pro dosažení autonomního chování systému a navržené základní funkce je zapotřebí zajistit několik podstatných podmínek.

Je zapotřebí navrhnout způsob jakým je možné zadat informace o prostoru, ve kterém se má systém pohybovat v dostatečné přesnosti na to, aby byla možná navigace, ale také zjednodušit zadávání tak, aby samotné zadávání nevyžadovalo od uživatele znalost samotného systému a bylo dostatečně intuitivní. Pro splnění těchto požadavků by bylo nejlepší vytvořit grafické rozhraní, které by fungovalo ideálně ve formě webové aplikace, kde je ve většině případů zaručena kompatibilita mezi větším množstvím systémů, než při použití dedikované aplikace.

Je zapotřebí také navrhnout způsob, jakým bude tato informace komunikována back-end systému a samotný back-end systém. V tomto případě bude ideální vytvoření serveru, který bude poskytovat samotné grafické rozhraní a získávat informace pomocí předem definovaných webových protokolů.

Tento server by se také dále dal využít pro samotné hledání cesty, neboť toto může být také poměrně výpočetně náročné. Server by také mohl poskytovat již finální trasu pro samotnou navigaci, čímž by se umožnilo i obsluhování více zařízení najednou.

Samotné hledání optimální cesty, tedy navigace v mapě by měla být prováděna v samostatném procesu, který získá jako vstup reprezentaci mapy pro navigaci. Tyto mapy

by mohly být uloženy se specifickým kódem a bylo by možné v případě stejného prostoru využít již existující mapu a pouze změnit cílovou a počáteční pozici. Pro hledání cesty by měl být využit algoritmus hledání cesty v grafu s váhováním jednotlivých cest. Po hledání cesty by měla být provedena validace, jestli je cesta opravdu platná. Jako výstup tohoto procesu by měly být již navigační data pro samotné zařízení.

Ovládání by mělo být řešeno pomocí získaných navigačních dat, ale uživatel by měl vždy mít možnost vynutit si nouzový režim ovládání, popřípadě přistání. Dále by měl mít uživatel informace o stavu dronu a jeho poloze. Samotný systém by měl reagovat na nečekané překážky, úpravou své trasy nebo návratem na původní umístění v případě, že pokračování v trase není možné. Uživatel by měl být o všech těchto stavech informován pomocí stejné aplikace jako pro nastavení mapy a cílové pozice pro zařízení.

Samotný dron by také 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 uražené 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.

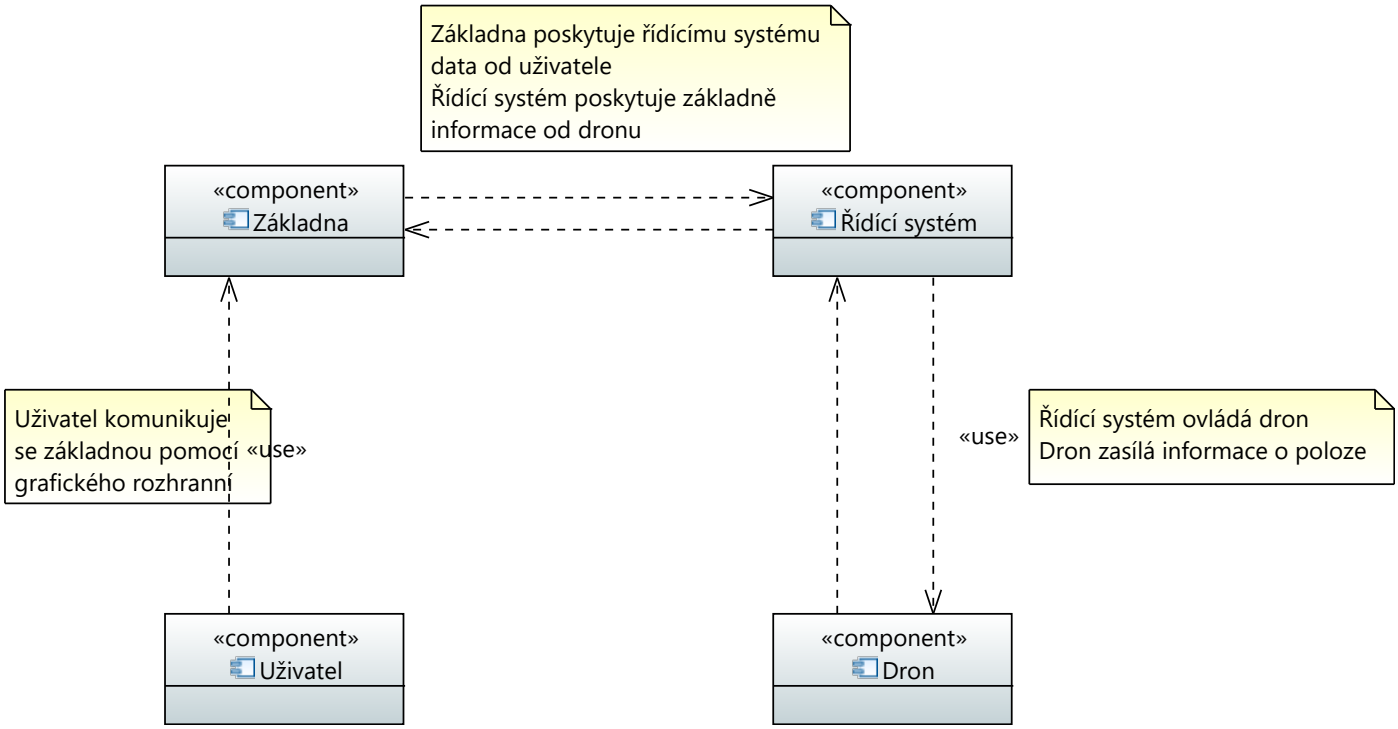
V případě navigace pomocí kamer by se mělo nejprve provést mapování prostoru a poté vygenerovat mapu překážek v prostoru, podle které by probíhala samotná navigace. U tohoto řešení by bylo zapotřebí vysokého výkonu kontroléru a s velkou pravděpodobností se toto řešení hodí spíše na zařízení řízené z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

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

nalezení nejkratší cesty existuje celá řada algoritmů od velmi jednoduchých algoritmů typu brute-force až po vysoce sofistikované algoritmy. Zvolení vhodného algoritmu s ohledem na rychlost výpočtu, jeho náročnost a chybovost je jedním z nejdůležitějších kroků u tohoto přístupu.

4.2 Diagram systému

Na tomto diagramu je vidět návrh softwaru využívající základnu. Tento návrh je pro finální verzi systému. Pro vývoj aplikace a obecné používání je možné vynechat základu a nechat uživatele komunikovat přímo s řídícím systémem čímž prakticky dojde ke spojení komponenty základna a komponenty řídící systém v jednom komponentu. Celý tento systém poté vyžaduje pouze jeden kus hardwaru kde běží veškerá komunikace a server najednou. Výhodou umístění komponentů řídícího systému a základy na stejném místě je odstranění nutnosti komunikačního kanálu mezi řídícím systémem a základnou je tedy nutná pouze komunikace s uživatelem a dronem.



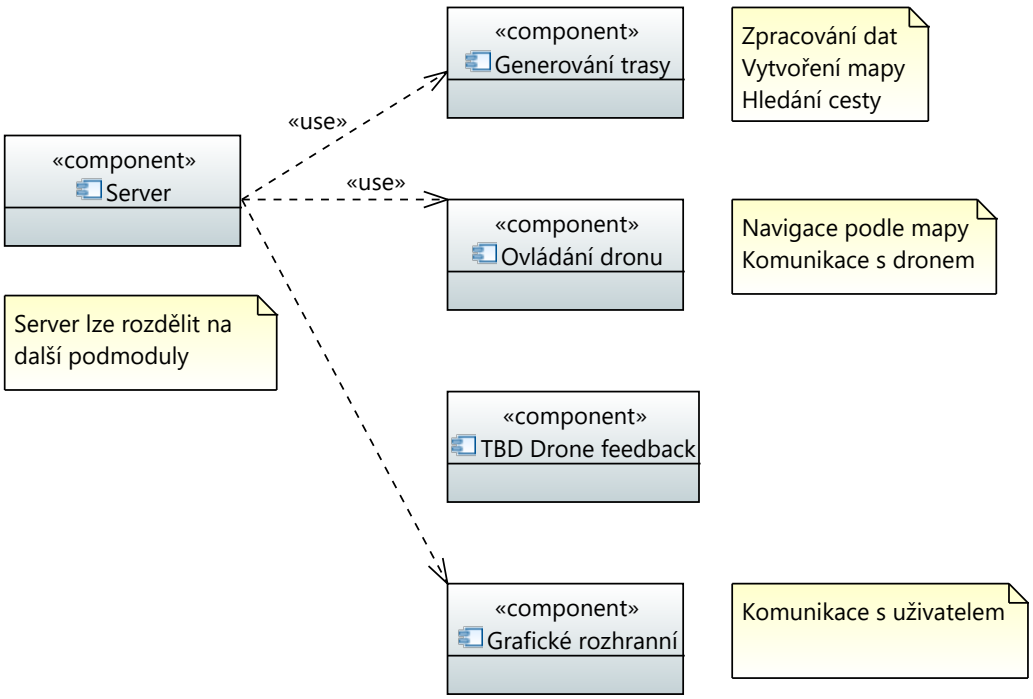
Obr. 4.1: Diagram navržených modulů systému

4.3 Navržený navigační software

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, mělo by se tedy jednat o takzvaný wrapper oddělující hardwarové ovládání od samotné logiky, uživatelského rozhraní a dalších částí softwaru jejichž zakomponování přímo do ovládání dronu by přinesly pouze potíže při dalším využití popřípadě při změně hardwaru či dokonce pouze nějaké funkce firmwaru což jsou změny prováděné relativně často u komerčních produktů. Tento modul je zde pro samotné řízení 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.

Přenos těchto informací by měl být řešen za pomoci nějakého definovaného protokolu ve známém tvaru. Zde se nabízí implementace vlastního datového typu pro přenos mapy od uživatele do serveru popřípadě využití již existujících formátů po jejich důkladném prostudování.

4.4 Vývojový diagram softwaru

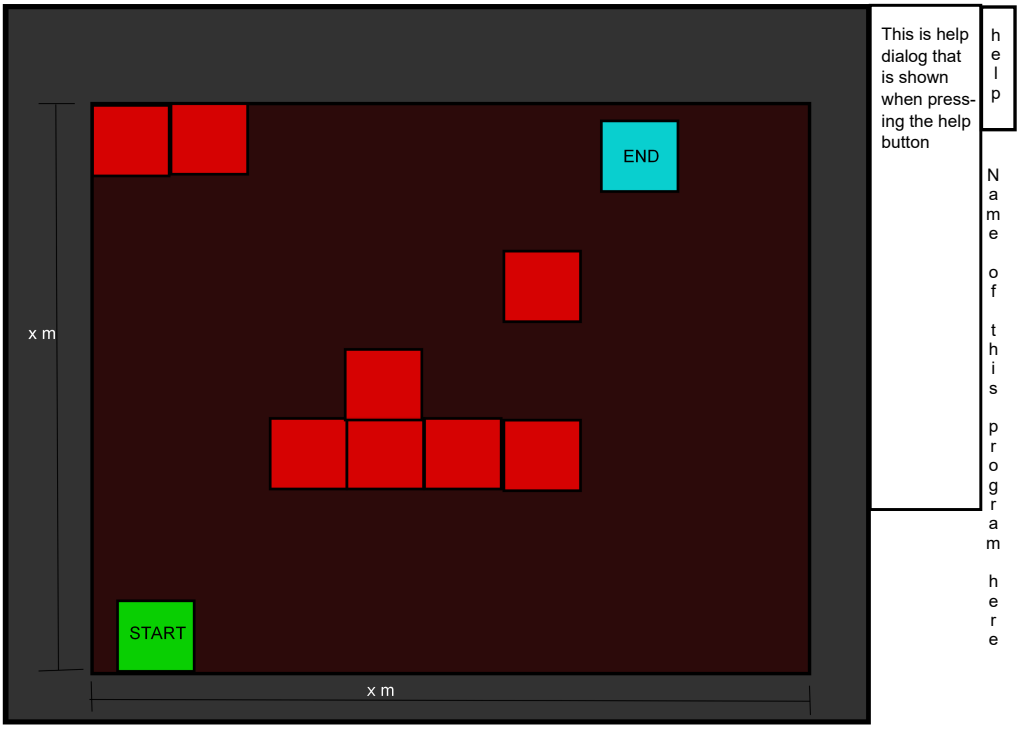


Obr. 4.2: Vývojový diagram navrženého softwaru

4.5 Navržené grafické rozhraní

Grafické rozhraní by mělo být pro uživatele hlavně srozumitelné a ovládání by mělo být jednoduché tak, aby nebyly vyžadovány žádné znalosti funkce systému, ale pouze znalost základního ovládání.

Pro první prototyp aplikace by mělo grafické rozhraní obsahovat pouze ovládání v jedné místnosti. Příklad takovéhoto grafického rozhraní je níže 4.3.

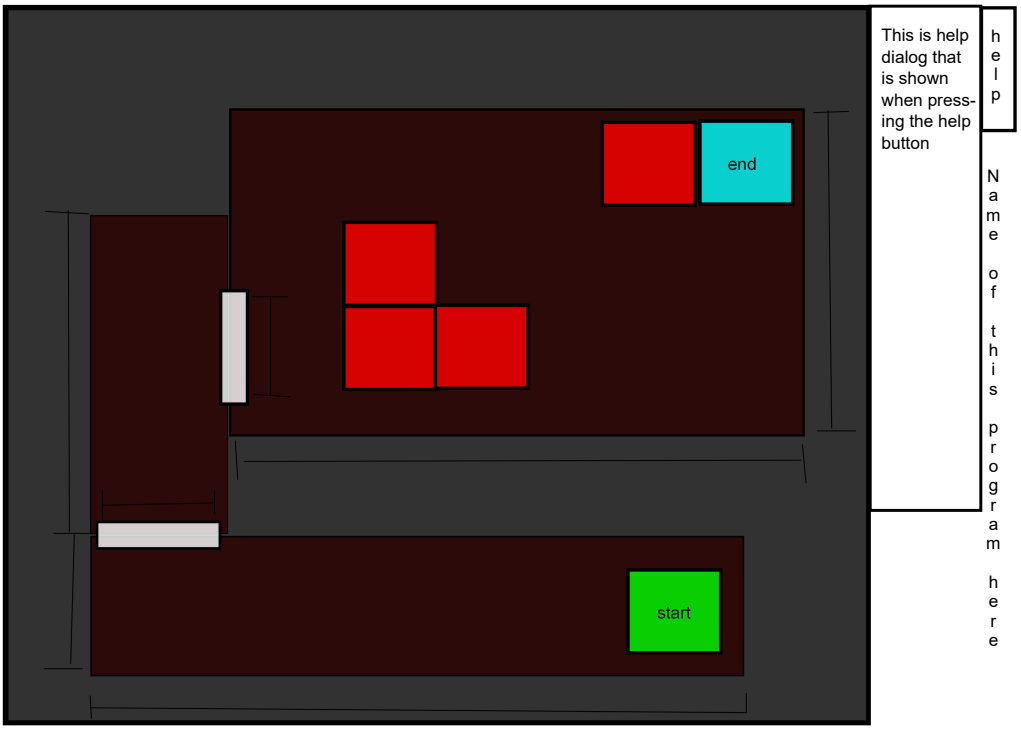


Obr. 4.3: Náčrtek grafického rozhraní pro prototyp

U takto navrženého grafického rozhraní by se poté mělo ovládat tak, že stisknutím pravého tlačítka myši by se otevřelo uživatelské menu ze kterého by bylo možné vybrat umístění prvku. Tímto způsobem by mělo být možné umístit jak jednotlivé překážky, kde u prototypu považujeme překážky za překážky jednotné velikosti 1 m. Také by mělo být možné umístit grafické prvky označující startovní a cílovou pozici, které budou nadále klíčové pro navigaci. Vhodná by také byla implementace možnosti nastavení velikosti místnosti a implementace nápovědy, kterou je možné zobrazit. V případě implementace jako webové stránky je ještě zapotřebí na této stránce vytvořit tlačítka pro přechod mezi jednotlivými stránkami aplikace pro jednodušší navigaci v aplikaci.

Pro budoucí a finální aplikaci by bylo ideálním krokem rozšíření tohoto grafického rozhraní o možnost přidávat další místnosti a mít tedy možnost navigace ve více místnostech najednou. Náčrtek tohoto rozhraní je níže 4.4.

U takovéhoto řešení by bylo zapotřebí přidání možnosti nakreslení jednotlivých místností a nastavení jejich jednotlivých velikostí. Je také nutné navrhnout způsob ukládání pozice dveří, které obvykle jsou menšího rozměru, než stěny. Dveře by měly být implemen-



Obr. 4.4: Náčrtek grafického rozhraní pro budoucí aplikaci

továny jako společný objekt mezi dvěma místnostmi, podle tohoto objektu by poté bylo možné skládat zpětně mapu pro navigační algoritmy, ale stále by bylo možné místnosti ukládat separátně. Samotné ovládání by mělo být velice podobné původní prototypové verzi, kde je implementace ovládání navržena pomocí menu po stisknutí pravého tlačítka myši. V této vývojové fázi by měl také začít přesun z překážek statické velikosti na překážky velikosti dynamické stále ještě obdélníkového tvaru.

Kompletně finální podoba tohoto řešení by ještě měla obsahovat možnost přepnout na stejné grafické rozhraní s načtenou mapou uživatele, kde by uživatel byl schopný v reálném čase sledovat pozici dronu a byl by schopný také přímého ovládání dronu pomocí tlačítek popřípadě by bylo vhodné řešení zobrazení konzole, která by přijímala příkazy pro ovládání a obcházela a tím vyřadila celý navigační systém pro zotavení z chybových stavů. V této naprosto finální podobně by také měla být možnosti nastavení libovolného tvaru a velikosti překážky. Nyní v tomto kroku je také možnost tuto aplikaci rozvinout a začít s prototypováním verze aplikace s navigací v 3D prostoru, toto je také ideální stav pro implementaci video playbacku z kamery dronu na webové stránce pro uživatele. Tato část je vytvářena velice pozdě ve vývojovém cyklu neboť je tato část vysoce výkonově náročná a jakákoliv práce s obrazem přináší vysokou míru komplexnosti a problémů s kompatibilitou pro navrhovaný systém a vylučuje se tedy se způsobem vývoje navrženým na postupném zdokonalování softwaru bez přidávání vysoce komplexních částí do nedokončeného softwaru.

4.6 Shrnutí návrhu

V této části práce byl proveden detailní návrh systému pro navigaci dronu, byl navržen modulární systém a detailně zdokumentovány všechny jeho moduly. Dalším krokem ve vývoji tohoto systému by měla být implementace tohoto řešení, které bylo navrženo. Při implementaci je důležité vzít v potaz, že design je prováděn bez znalostí implementace a detailní funkčnosti jednotlivých komponent což znamená, že rozumná míra odklonu od designu je očekávána. Tento návrh byl také prováděn pro kompletní aplikaci obsahující mnoho komplexních částí a pro prototypové účely se nepočítá s jeho kompletní implementací. Tento návrh byl také prováděn v grafické formě a tato dokumentace je dostupná v přílohách.

5

Popis použitého hardwaru

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

5.1 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. 5.1: 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, 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.

6

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.

6.1 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.

6.1.1 HTML5 canvas

Canvas prvek je prvek pro využití grafiky za pomoci skriptovacího jazyku JavaScript 6.2. Tento prvek základně obdélníkového tvaru je možné využít k zobrazování libovolné grafiky a je také možné přidávat prvkům vlastní animace a je také možné jej využít pro zobrazování 3D akcelerované grafiky za využití WebGL. Canvas vyžaduje pouze inicializaci v html dokumentu a poté je možné přistupovat k němu jako k jakémukoliv jinému objektu.

```
1 <canvas id="canvas"></canvas>
```

Canvas je podporován v současné době všemi internetovými prohlížeči podporujícími HTML5 6.1 standart a je využíván velkým množstvím knihoven. [18]

6.2 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.

6.3 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 aktuální verze (TypeScript 2.2) dostupná z [6].

6.4 Datový formát JSON

JSON je v současné době jedním z nejvíce rozšířených datových formátů pro výměnu dat u webových aplikací, kde úspěšně konkuruje složitějšímu formátu xml. Výraznou výhodou formátu JSON je to, že JSON je validním kodem v jazyce JavaScript a je možné pracovat přímo s těmito objekty bez nutného převádění. V jazyce JavaScript je přímo podporován přechod mezi formátem JSON a string pomocí funkcí *JSON.parse()* 6.4.1 a *JSON.stringify()* 6.4.2. Tato vlastnost dovoluje využití při zasílání dat pomocí POST a GET http requestů a jejich jednoduché zpracování.

6.4.1 JSON.parse()

JSON.parse() je JavaScriptová 6.2 metoda, která zkonstruuje z prvku typu string validní JavaScriptovou 6.2 hodnotu ve formátu JSON 6.4. V této funkci je také možno specifikovat druhý vstupní parametr jako funkci, která provede transformaci výsledných hodnot. [19]

```
1 JSON.parse(text[, funkce])
```

6.4.2 JSON.stringify()

JSON.stringify() je JavaScriptová 6.2 metoda, která konvertuje JavaScriptovou 6.2 hodnotu ve formátu JSON 6.4 na hodnotu ve formátu string. V této funkci je možné definovat druhý parametr jako funkci, která provede nahrazení specifikovaných znaků jinými specifikovanými znaky. [20]

```
1 JSON.stringify(JSON[, funkce[co nahradit, čím nahradit]])
```

6.5 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.

6.6 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].

```
var drone = require('ar-drone');
var client = drone.createClient();

client.takeOff();

client.after(1000, function()
{
  this.front(0.5);
})
.after(1000, function ()
{
  this.stop();
  this.land();
});
```

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].

```
1  var auto = require(‘ardrone-autonomy‘);
2  var mission = auto.createMission();
3
4  mission.takeoff(); // Vzlétnutí dronu
5  mission.zero(); // Zde se nastaví počáteční bod pro let
6  mission.altitude(1); //Nastavení výšky letu na 1 m
7  mission.forward(1); //Let vpřed na vzdálenost 1m
8  mission.hover(100); //Zůstat na místě pod dobu 100 ms
9  mission.land(); //Přistát
10
11 mission.run(function (error, result) // Spuštění mise
12 {
13   if(error)
14   {
15     console.trace('ERROR: %s', error.message);
16     mission.client.stop();
17     mission.client.land();
18   }
19   else
20   {
21     console.log('SUCCESS');
22     process.exit(0); // Ukončení programu drona
23   }
24 });
```

Příklad použití knihovny ardrone-autonomy

7

Implementovaný systém

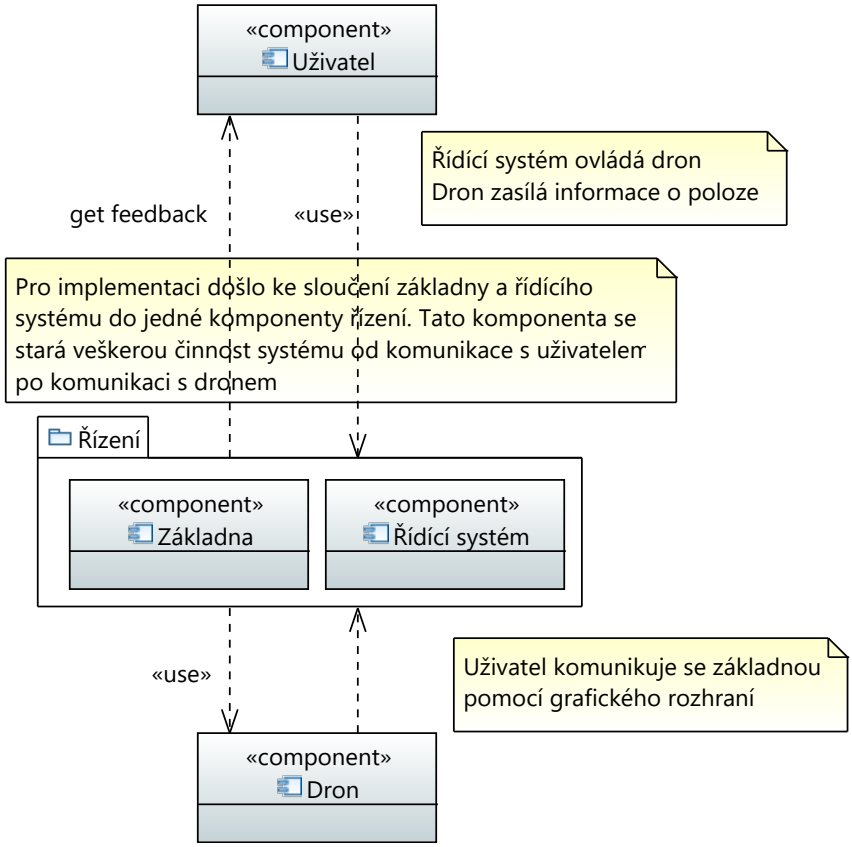
V této kapitole je popsán systém, který byl experimentálně implementován pro ověření funkčnosti tohoto návrhu. Tento systém neodpovídá návrhu ve všech bodech z důvodu velice složité implementace. Pro implementaci bylo také třeba provést rozhodnutí o zjednodušení systému a vynechání několika pokročilých funkcí. Větší pozornost byla věnována softwarové části řešení systému a hardwarové řešení bylo řešeno pouze povrchně.

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.

Systém je implementován za využití programovacího jazyku TypeScript 6.3, který je poté překládán do jazyku JavaScript 6.2. Pro kompilaci TypeScriptu 6.3 je využit kompilátor tsc. Tento software je možné dělit do tří hlavních částí: Grafické rozhraní, ovládání dronu a server. Pro implementaci serverové části aplikace byl využit framework NodeJS 6.5. Software si stále zachovává navrženou modularitu a je tedy možné popsat jeho jednotlivé části samostatně.

7.1 Hardware systému použitý při implementaci

Systém je implementován za využití komerčně dostupného dronu AR Drone 2.0 viz 5.1 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 možné spustit na jakémkoliv PC připojeném k wi-fi síti. 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.

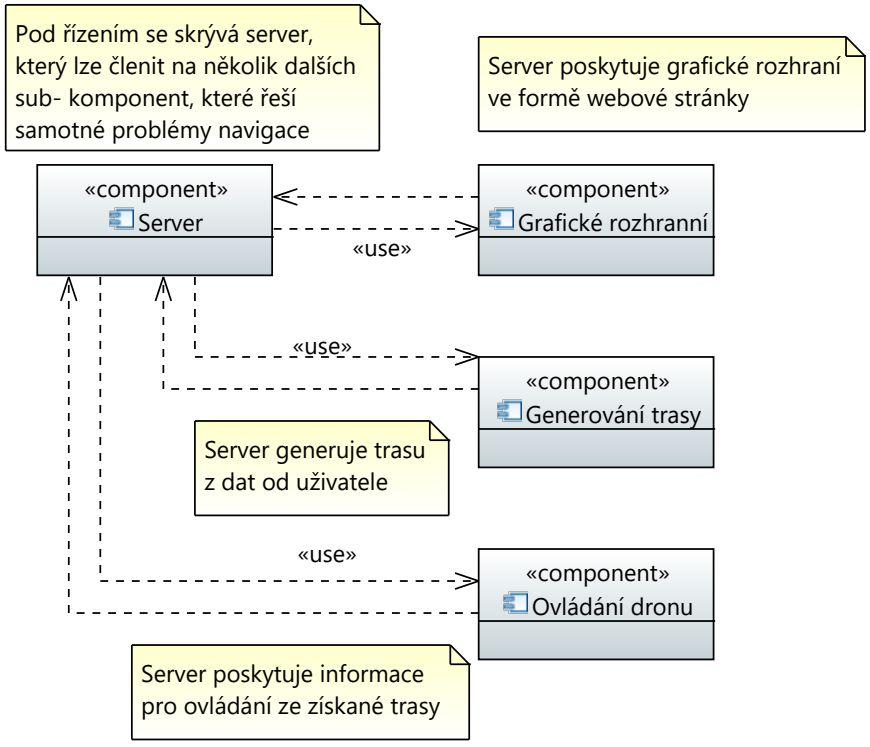


Obr. 7.1: Diagram implementovaného řešení

8

Popis implementace aplikace

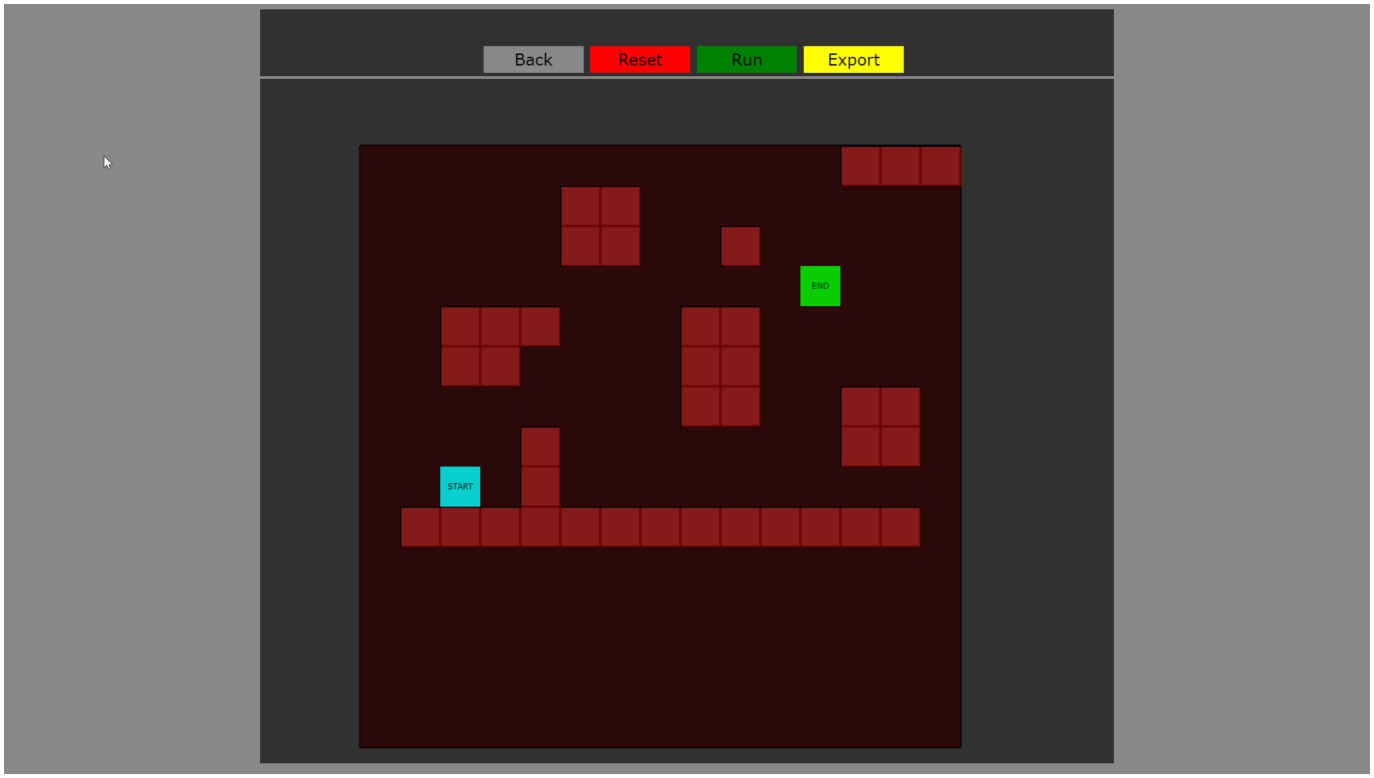
Tato část práce pojednává do hloubky o implementaci celé aplikace, je zde vysvětlena funkčnost aplikace a implementace jednotlivých částí včetně diagramů, popřípadě částí kódu pro lepší pochopení implementace. Implementovaná řídící aplikace se skládá z několika částí, které jsou v této kapitole popsány do hloubky včetně jejich implementace.



Obr. 8.1: Diagram implementovaného systému

8.1 Grafické rozhraní uživatelské aplikace

Při spuštění je jako první viditelná úvodní stránka s jednoduchým menu pro vybírání funkce. Hlavní část aplikace se skrývá pod tlačítkem nastavení mapy. Na této stránce se nachází hlavní část uživatelské aplikace pro nastavení mapy pro dron, startovní a koncové pozice. V hlavní části se nachází mapa pro nastavení oblasti pro ovládání a navigační tlačítka pro přepínání mezi jednotlivými částmi aplikace. Všechny prvky jsou poté postupně překreslovány při každém obnovení stránky což je hlouběji popsáno v sekci implementace grafického rozhraní.



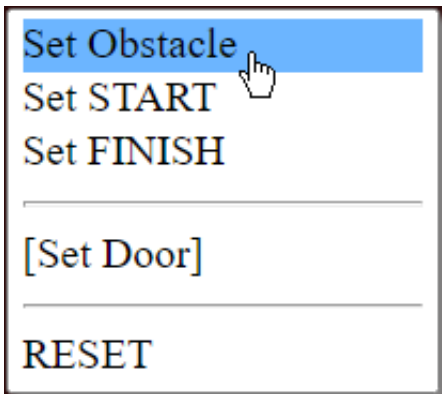
Obr. 8.2: Grafické rozhraní aplikace

V menu 8.3 se nachází tlačítka sloužící pro přesun mezi stránkami aplikace, další tlačítko slouží pro resetování mapy uživatelem a tlačítko slouží pro exportování mapy od uživatele a zaslání mapy serveru, po které se po vygenerování trasy začne vlastní let dronu. Posledním tlačítkem je tlačítko stop, které zastaví dron a přistane s ním v krajním případě potřeby přerušení letu.

Obr. 8.3: Ovládací tlačítka aplikace

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 pohy-

bovat, u tohoto prototypu každý čtverec reprezentuje 1 m. Na této mapě uživatel nastaví počátek a konec cesty pro dron pomocí menu zobrazeného po stisknutí pravého tlačítka myši 8.4 pomocí voleb *START* a *END* na požadovaném místě odpovídajícím reálnému prostoru. Nastavení překážek v prostoru se provede jednoduše kliknutím pravého tlačítka myši na požadované místo v mapě odpovídající relativní pozici překážky v prostoru. Po vybrání překážky se změní barva pole na barvu označující překážku. Momentálně není možné nastavit parametry jako je výška překážky a podobně. Tato funkcionalita momentálně není podporována grafickým rozhraním. Implementace této mapy je do hloubky popsána níže, ale jedná se o prvek canvas 6.1.1 jazyku HTML5 6.1.

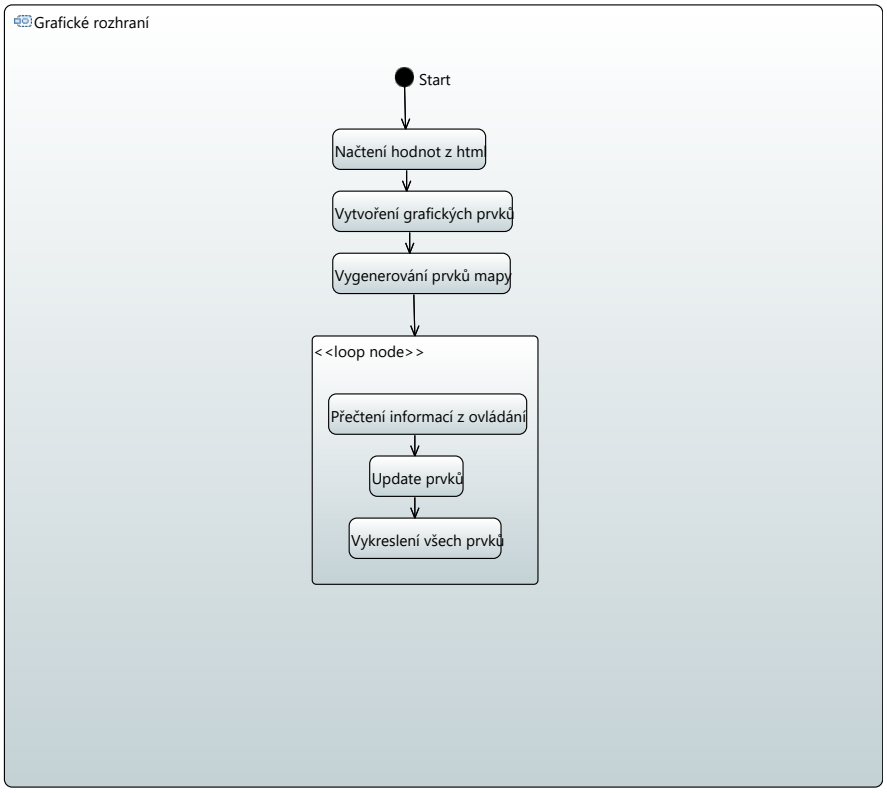


Obr. 8.4: Menu zobrazené po kliknutí pravého tlačítka

8.2 Realizace grafického rozhraní

Celé grafické rozhraní je vytvořeno v jediném objektu *canvas* 6.1.1 jazyka HTML5 6.1 a skripty pro samotné ovládání jsou poté realizovány v jazyce JavaScript 6.2. Pro samotnou funkčnost grafického rozhraní je vyžadována implementace dvou klíčových funkcí, jedná se o funkce setup() a mainLoop().

Funkce *setup()* 8.2 je inicializační funkce, ve které dochází k vytváření všech počátečních objektů v paměti. Všechny tyto objekty jsou poté uloženy do vykreslovacího pole, toto pole obsahuje všechny objekty, které je potřeba vykreslovat.



Obr. 8.5: Diagram realizace grafického rozhraní

```
/**
 * This is setup function for user interface
 *
 * @param none
 * @return none
*/
function setup()
{
    //Setup canvas and rendering context
    canvas = <HTMLCanvasElement>document.getElementById('cnvs');
    ctx = canvas.getContext("2d");

    canvasTop = <HTMLCanvasElement>document.getElementById('cnvsTop');
    ctxTop = canvasTop.getContext("2d");

    //load images from server using HTML
    img1 = <HTMLImageElement>document.getElementById('img1');

    //clean drawArray
    drawArray = new Array<IDrawable>();
    //clean output array
    outArray = new Array<string>();

    // GENERATION CODE
    //setup parameters and position
    var startX = 150;
    var startY = 100;
    var buttonSize = 60;
    var buttonLineCount = 15;
    var color = "#2B0909";

    //setup text for static textFields
    //Create grid
    btnGrid = new ControlGrid(ctx, canvas, startX, startY, buttonSize, "
        black", color, buttonLineCount, drawArray);

    //Create user UI
    controlUI = new ControlButtonUI(ctxTop, canvasTop, (canvasTop.width
        / 2) - (150 * 2) - 5, canvasTop.height - 40 - 5, 150, 40, "#
        efb300", 24, drawArray);

    //Uncomment for serverside logging
    //SendData(serverURL, "%CLIENT ONLINE%");
}
```

Inicializační funkce grafického rozhraní

Funkce *mainLoop()* 8.2 je funkcí, která se stará o samotnou funkčnost aplikace. Tato funkce je volána z webové stránky pomocí eventu *onLoad()* 8.2 a poté je tato funkce volána s každým animačním framem v určitých intervalech. Toto je dáno samotným vykreslovacím enginem canvasu. Zde probíhá vykreslování pomocí jednoduchého cyklu, ve kterém se vykreslí veškeré prvky vykreslovacího pole. Jednoduchost této funkce je dosažena díky zdědění interfacu *IDrawable* 8.2 veškerými vykreslitelnými prvky což zaručuje implementaci vykreslovací funkce.

```
1  /**
2   * This interface is used for objects able to be drawn to canvas
3   * This interface implements x,y position and draw() method
4   */
5  interface IDrawable
6  {
7      context : CanvasRenderingContext2D;
8
9      x : number;
10     y : number;
11
12     draw() : void;
13 }
```

Interface IDrawable

Mezi vykreslované prvky patří samostatná tlačítka pro ovládání, jednotlivé prvky mapy kde je vytvořena síť těchto prvků a každému prvku je přiřazeno jeho ovládání a proměnné. Tato síť má vlastní vykreslovací metodu, která se stará o vykreslování všech prvků mapy, pro vykreslování je nutné ukládat stav jednotlivých prvků aby nedocházelo ke ztrátě dat při překreslení a podle těchto dat je nutné překreslení provádět B.

```
1  window.onload = () =>
2  {
3      //setup app
4      setup();
5      //start main application loop
6      mainLoop();
7  }
8
9  /**
10  * This is main function for user interface and main cyclicly called
        function
11  *
12  * @param none
13  * @return none
14 */
15 function mainLoop()
16 {
17     requestAnimationFrame(mainLoop);
18
19     //Clear canvas every frame
20     ctx.fillStyle = "#313131";
21     ctx.fillRect(0, 0, 1280, 1024);
22
23     ctxTop.fillStyle = "#313131";
24     ctxTop.fillRect(0, 0, 1280, 100);
25
26     //logic
27     exportButtonUI(controlUI, btnGrid, tList, tGrid);
28     startButtonUI(controlUI, btnGrid, tList, tGrid);
29     backButton(controlUI);
30     resetButton(controlUI);
31
32     //draw call
33     for(var i : number = 0; i < drawArray.length; i++)
34     {
35         var drawable : IDrawable = drawArray[i];
36         drawable.draw();
37     }
38 }
```

Spouštění grafického rozhraní

8.3 Implementace ovládání uživatelské aplikace

Hlavní ovládací tlačítka pro ukládání a startování letové procedury jsou tlačítka implementovaná jako prostý obdélníkový objekt v canvasu HTML5 6.1. Tyto objekty jsou poté stylizovány pomocí css stylu 8.3 pro nastavení samotného vzhledu. Interakce s těmito tlačítky je řešena zachytáváním událostí ze vstupních zařízení (klávesnice, myš), zachytávání je řešeno samotným canvasem. Implementace vyžaduje pouze vytvoření obsluhy pro tyto stavy. V případě těchto tlačítek je testováno zda bylo stisknuto levé tlačítko myši. V případě, že se tak stalo je zapotřebí zjistit, kde bylo tlačítko stisknuto, protože tento event je obsluhován u všech existujících tlačítek a v menší míře také u samotné mapy. Pro zjištění správnosti tlačítka je zapotřebí získat pozici myši, které se získává pouze ve vztahu k celému oknu, je zde tedy zapotřebí tyto koordináty přepočítat pro pozici v canvas objektu. Po tomto přepočtu je možné z velikosti tlačítek a jejich umístění zjistit, kterého tlačítka se stisknutí týká a aktivovat jeho reakci.

Implementace mapy je poměrně podobná tlačítkům, nicméně tyto prvky reagují na pravé tlačítko myši. Je provedeno stejné vyhledávání stisknutého tlačítka, ale v tomto případě je reakce poměrně složitější. Pro implementaci vlastního menu pro stisknutí pravého tlačítka tak aby bylo podporováno většinou současných webových prohlížečů. Zapotřebí je definovat toto menu předem v html dokumentu pomocí div notace 8.3 jednotlivé prvky tohoto menu musí být definovány jako samostatná třída a je také nutné navrhnout kaskádový styl v css souboru 8.3. Pokud jsou provedeny tyto kroky, je možné vypnout funkci standardního menu pravého tlačítka a nahradit ji tímto vlastním menu. Pro funkčnost tohoto menu je zapotřebí přidat listener na stisknutí levého tlačítka pro všechny prvky tohoto menu a stejně tak jejich obsluhu.

U prvků mapy jsou to možnosti: *set obstacle*, *set START*, *set END* a *reset*. Tyto funkce mění stav samotných prvků mapy, tento stav je poté informace, která se po úpravách zasílá serveru.

```
1     <div class="menu">
2         <div class="menu-item1">Set Obstacle</div>
3         <div class="menu-item2">Set START</div>
4         <div class="menu-item3">Set FINISH</div>
5         <hr>
6         <div class="menu-item4">[Set Door]</div>
7         <hr>
8         <div class="menu-item5">RESET</div>
9     </div>
10    <div id="APP" align="center">
11        <canvas id="cnvsTop" height="100" width="1280"></canvas>
12        <canvas id="cnvs" height="1024" width="1280"></canvas>
13    </div>
```

div notace pro vytvoření menu

```
body
{
    background-color: #888888;
}

h1
{
    color: #000000;
    margin-left: 20px;
}
.menu
{
    width: 150px;
    box-shadow: 3px 3px 5px #888888;
    border-style: solid;
    border-width: 1px;
    border-color: grey;
    border-radius: 2px;
    padding-left: 5px;
    padding-right: 5px;
    padding-top: 3px;
    padding-bottom: 3px;
    position: fixed;
    display: none;
    background-color: #ffffff;
}

.menu-item1, .menu-item2, .menu-item3, .menu-item4, .menu-item5
{
    height: 20px;
}

.menu-item1:hover, .menu-item2:hover, .menu-item3:hover, .menu-item4:
   hover, .menu-item5:hover
{
    background-color: #6CB5FF;
    cursor: pointer;
}
```

css styl uživatelské aplikace

8.4 Ukládání mapy

Ukládání mapy je řešeno za pomoci objektového formátu JSON 6.4. Samotná práce se soubory je řešena pomocí standardního IO systému pro soubory a je využito synchronního čtení. Soubory jsou čteny i zapisovány znak po znaku pomocí knihovních funkcí.

Mapa je v softwaru uložena jako třída *RoomMap* E. Tato třída obsahuje veškerá data o místnosti, koordináty jsou jednotlivě uloženy ve vlastní třídě *Coordinate* F tato třída je využívána jako kontejner pro jednu sadu koordinátů a obsahuje také jejich stav a validitu, každý koordinát má také vlastní ID, které je využíváno pro určení umístění v místnosti. Veškerá data o místnosti včetně koordinátů, které jsou u třídy *RoomMap* E uloženy jako pole instancí třídy *Coordinate* F jsou uloženy do prvku data, který je poté převeden do formátu JSON 6.4 TypeScriptovou knihovní funkcí *JSON.stringify()* 6.4.2, tato funkce vyprodukuje string s veškerými daty ve formátu string. Tento string je poté možné zapsat jednoduše do souboru.

Pro získání koordinátů z uložené mapy je zapotřebí přečtení souboru do formátu string a jeho následné převedení na formát JSON 6.4 na což se využívá knihovní funkce *JSON.parse()* 6.4.1. Takto získaná data ještě nejsou použitelná je zapotřebí tato data převést na třídu *RoomMap* E. Pro převedení informace o koordinátech je ještě zapotřebí separovat tyto data a provést parsování těchto elementů a následně vytvořit instanci třídy *Coordinate* F pro každý jednotlivý koordinát a až tyto koordináty je možné uložit do třídy *RoomMap* E jako pole.

```
1  /**
2   * This function loads room data and returns new RoomMap instance
3   *
4   * @param path tom saved JSON file
5   * @return new RoomMap instance
6  */
7  public loadSingleRoom = (savePath : string) : RoomMap =>
8  {
9      let FH = new FileHelper();
10     var rawData = FH.readFile(savePath);
11     var parsedData = JSON.parse(rawData);
12
13     var ary : Array<Coordinate>  = new Array<Coordinate>();
14     parsedData.markers.forEach(element => {
15         ary.push(new Coordinate(element.ID, element.x, element.y,
               element.z,
16         element.valid, element.idstring));
17     });
18
19     var room : RoomMap = new RoomMap(parsedData.ID,
20         parsedData.WIDTH, parsedData.HEIGHT,
21         parsedData.START, parsedData.END,
22         parsedData.idString, ary);
23
24     return room;
25 }
```

Načítání dat z JSON souboru

8.5 Komunikace uživatelského rozhraní se serverem

Realizace přenosu nastavené mapy od uživatele je prováděna pomocí http POST požadavku 8.5. Tento požadavek je dále zpracováván serverem a na straně uživatele je pouze převedena mapa do objektového formátu JSON 6.4. Výhodou využití formátu JSON 6.4 je jeho nativní podpora v JavaScriptu 6.2 a podpora tohoto formátu u POST požadavků 8.5. Díky této vlastnosti je možné přesouvat poměrně velké množství informací mezi klientem a serverem bez obav ztráty informace a nutnosti vymýšlet vlastní přenosový formát.

Převod mapové informace do JSON 6.4 formátu je prováděn až po stisknutí tlačítka pro odeslání dat serveru. Tato informace je poté zpracovávána serverem pro navigaci. Detailní zpracování je popsáno v kapitole implementace serveru níže. Stručně se jedná o načtení JSON 6.4 objektu zpět do třídy obsahující informace o mapě a následně se provede samotné hledání cesty.

```
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 }
13 %%
```

Vytváření POST požadavku a zasílání dat serveru

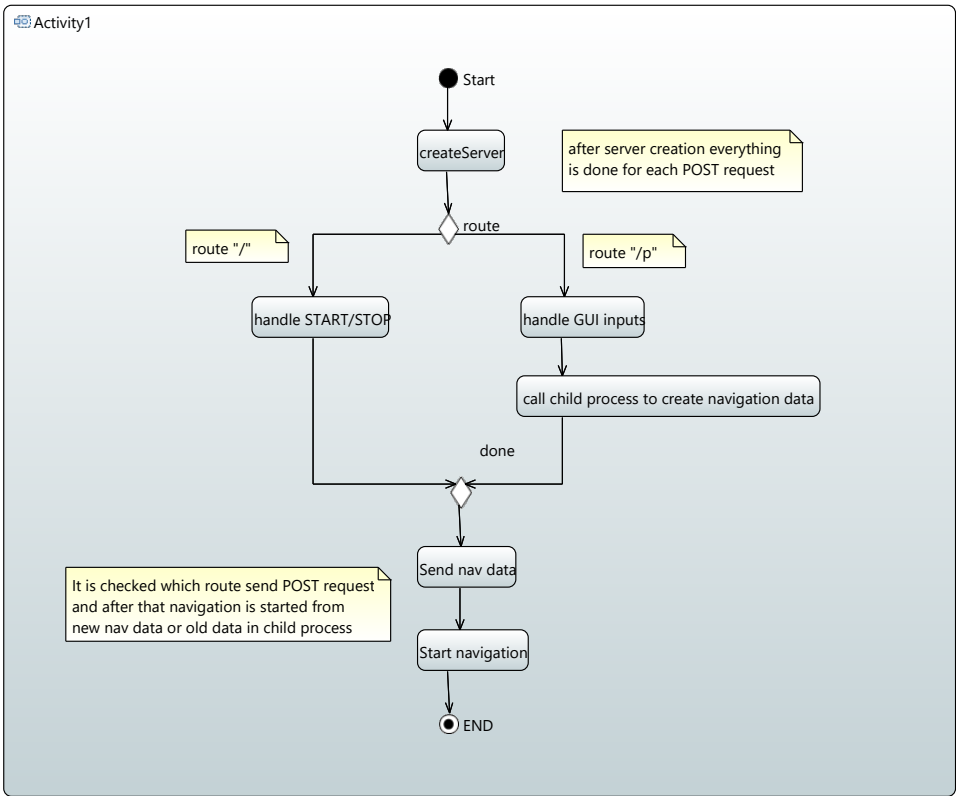
Tato funkce je funkcí pro přenos dat z uživatelské aplikace do serveru. Provede se vytvoření XMLHttp požadavku, který je otevřen a je označen jako požadavek typu POST 8.5. Tomuto požadavku je přiřazena adresa serveru a je odeslán pomocí protokolu http. Tato funkce je zavoláno po stisknutí tlačítka pro export dat. Pro zaslání dat serveru je zapotřebí provést úpravu dat na formát, který je možné zaslat serveru pomocí požadavku POST 8.5. Tento problém je řešen pomocí funkce 8.5, která získá JSON objekt s informacemi o mapě a vytvoří z něj reprezentaci ve formátu string, která je poté zaslána požadavkem POST 8.5 serveru.

```
1  var helpArray = grid.getButtonGrid().getButtonArray();
2
3  var id = 1;
4  var startCo;
5  var endCo;
6  var coords = new Array<Coordinate>();
7
8  for(var i = 0; i < helpArray.length; i++)
9  {
10   for(var j = 0; j < helpArray[0].length; j++)
11   {
12     if(helpArray[i][j].isStart)
13     {
14       startCo = new Coordinate(0, i, j, 0, true, "START");
15     }
16     if(helpArray[i][j].isEnd)
17     {
18       endCo = new Coordinate(1, i, j, 0, true, "END");
19     }
20     if(helpArray[i][j].down)
21     {
22       coords.push(new Coordinate(id++, i, j, 0, true, "X"));
23     }
24   }
25 }
26
27 var map : RoomMap = new RoomMap(0, helpArray.length, helpArray.length,
       startCo, endCo, "map", coords);
28
29 var outputString : string = map.returnJSONFormat();
```

Úprava dat na formát pro zaslání serveru

8.6 Implementace serveru

Server je implementován jako standardní NodeJS 6.5 http server. Server je vytvářen pomocí třídy *Server* frameworku NodeJS 6.5 na portu 8080 běžící na adrese *http://127.0.0.1*. Server je možné rozdělit na několik částí pro zpracování jednotlivých požadavků a obsluhu hledání cesty a ovládání dronu.



Obr. 8.6: Detailní design implementovaného serveru

Server obsluhuje několik požadavků HTTP protokolu. Při obdržení požadavku GET od webového prohlížeče je volána metoda pro jeho obsluhu. Požadavek GET je dotazovací metoda protokolu HTTP pro předávání proměnných mezi serverem a webovým prohlížečem popřípadě se pomocí této metody webový prohlížeč dotazuje dotazem GET když vyžaduje získání dat webové stránky ze serveru pro její zobrazení. Tato metoda Je využívána čistě pro zobrazení webové stránky. Pro používání je zde implementována metoda pro vyřizování tohoto požadavku. Tato metoda se používá tak, že webovému prohlížeči poskytne obsah vstupního souboru což jsou *.html*, *.js* a *.css* soubory nutné pro zobrazení webové stránky a její funkčnosti.

```
if (request.method == 'GET')
      {
        console.log('GET');
        fs.readFile(filename, function  (error, data)
        {
            console.log(error)
            if (error)
          {
              response.writeHead(500);//internal error
              return response.end('Cannot load index.html');
          }
          var extension = path.extname(filename)
             console.log(extension);
          var cType : string = getCT(extension.toString());
             console.log(cType);
          response.setHeader('Content-type', cType);
          response.writeHead(200);//OK
          response.end(data);
      });
    }
}
```

Vyřizování GET požadavku

V případě požadavku POST 8.6 musí server rozeznat na kterou cestu je tento požadavek zaslán, což zjistí z adresy požadavku a v případě požadavku POST 8.6 pro navigační cestu jsou přečtena a uložena do JSON souboru viz 8.6 kde jsou pouze uložené startovní a cílová pozice a také pole s uloženými pozicemi překážek. Po uložení dat je vytvořen nový proces, kterým je navigační program, do kterého se zašle informace o umístění dat. Tento proces provede zpracování těchto dat od uživatele a samotné vyhledávání cesty. Po úspěšném nalezení cesty se vytvoří další JSON soubor obsahující letová data. Tyto data jsou určena již pro samotný let dronu. 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 získaných z předešlého procesu.

```
else if(request.url == "/p")// /p route
{
  var data : String = new String();
  request.on('data', function (rData)
  {
    data += rData.toString();
  });
  request.on('end', function ()
  {
    if(data == "START")
    {
      var dcPath2 = bbdir + "/nav/Copter.js";
      var fork2 = require('child_process').fork; // start the drone
          control
      var child2 = fork2(dcPath2);
      child2.send(navDataFile); // send data to navigation process
      response.end();//CLOSE RESPONSE
    }
    else if(data == "STOP")
    {
      var dcPath = bbdir + "/nav/stop.js";
      var fork = require('child_process').fork; // start the navigation
          process
      var child = fork(dcPath);
      child.send("STOP"); // send data to navigation process
      response.end();//CLOSE RESPONSE
    }
    else
    {
      createFile(dataFile, data);
      var dcPath = bbdir + "/nav/main.js"; // call pathfinding process
      var fork = require('child_process').fork; // start the navigation
          process
      var child = fork(dcPath);
      child.send(dataFile); // send data to navigation process
      child.on('message', (navData) =>
      {
        createFile(navDataFile, navData);
      });
      data = new String(); // clear data
      response.end();//CLOSE RESPONSE
    }
  });
}
```

Vyřizování POST požadavku a spouštění podprocesů

```
{
  "ID":0,
  "WIDTH":15,
  "HEIGHT":15,
  "START":{
       "ID":0,
       "x":2,
       "y":3,
       "z":0,
       "optionalString":"START",
       "valid":true,
       "data":{"ID":0,"valid":1,"X":2,"Y":3,"Z":0,"idstring":"START"}
      },
  "END":{
       "ID":1,
       "x":5,
       "y":3,
       "z":0,
       "optionalString":"END",
       "valid":true,
       "data":{"ID":1,"valid":1,"X":5,"Y":3,"Z":0,"idstring":"END"}
      },
  "idString":"map",
  "markers":[
         {
            "ID":1,
            "x":1,
            "y":2,
            "z":0,
            "optionalString":"X",
            "valid":true,
            "data":{"ID":1,"valid":1,"X":1,"Y":2,"Z":0,"idstring":"X"}
         },
         {
            "ID":2,
            "x":1,
            "y":3,
            "z":0,
            "optionalString":"X",
            "valid":true,
            "data":{"ID":2,"valid":1,"X":1,"Y":3,"Z":0,"idstring":"X"}
         }
        ]
}
```

Příklad obsahu generovaného JSON souboru od uživatele

8.7 Generování cesty

Generování cesty je možné provádět buď přímo z dat zadaných uživatelem a zaslaných serverem, nebo pomocí uložené mapy v JSON souboru 8.6. Také je možné generování zpáteční cesty pomocí funkce, tato cesta je, ale generována pomocí zrcadlení předchozí cesty. Generování cesty je řešeno momentálně za využití brute force metody. Tato metoda není vysoce inteligentní, ale pro tyto účely naprosto dostačuje.

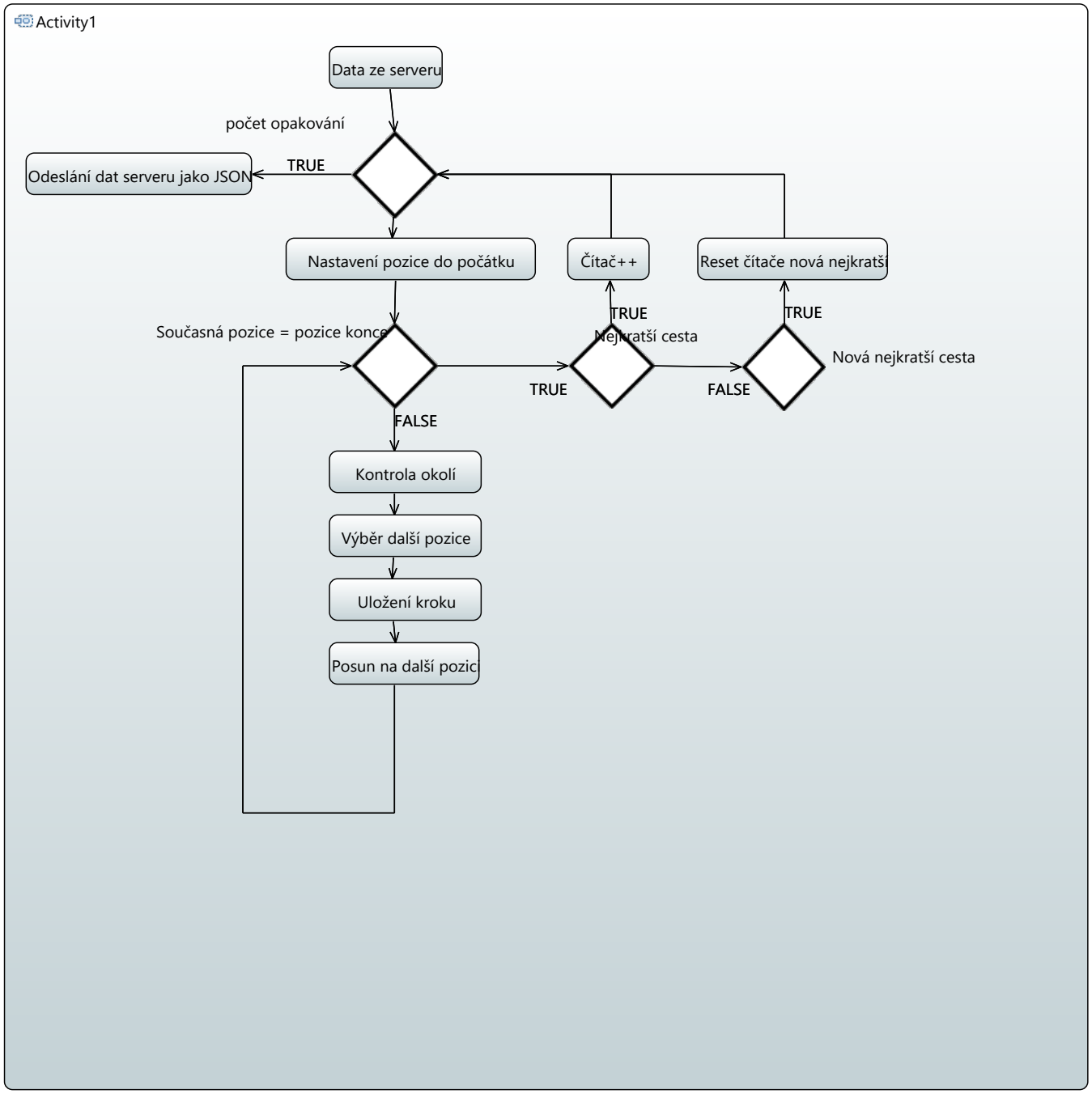
Mapa pro účely navigace a hledání cesty je implementována ve formě JSON souboru, který je poté převeden na dvourozměrné pole čísel. Standardní rozměry mapy jsou 15 x 15 kde každý prvek představuje oblast velikosti 1 m. Tato velikost je v současné době zvolena z důvodu bezpečnosti a přesnosti určení pozice dronu a je možné ji změnit pouze při překompilování celé aplikace. Mapa je vytvářena z řetězce dat, který je získán buď z uloženého JSON souboru 8.6, nebo přímo od serveru přímo od uživatele. Momentálně není možné si vybrat z uložených tras, ale pro navigaci je vždy použita poslední vygenerovaná cesta. Všechny tyto soubory jsou serverem ukládány do kořenového adresáře aplikace.

Samotné hledání cesty probíhá tak, že se provede parsování JSON souboru 8.6, který je vytvořen serverem a jehož název je zaslán serverem podprocesu pro navigaci. Tento podproces je vytvořen pomocí metody *.fork()* tato metoda dovoluje po vytvoření podprocesu pokračovat v komunikaci s tímto podprocesem. Parsování je provedeno pomocí JSON parseru, který provede samotné načtení souboru a extrahuje potřebná data. Takto přeparsovaná data jsou uložena do instance třídy *RoomMap* E takto upravená data se dále využívají. V prvním kroku se nastaví velikosti mapy, které jsou taktéž uloženy ve třídě *RoomMap* E. Po nastavení velikosti se přečte začátek a konec cesty a provede se uložení jejich souřadnic do pole čísel, které je požadované pro následné vyhledávání. Po přečtení startovních a cílových souřadnic se převede informace o souřadnicích překážek ze třídy *RoomMap* E tyto souřadnice se použijí pro vytvoření reprezentace mapy ve formě dvourozměrného pole ve kterém se vyznačí všechny známe souřadnice, pro prázdné místo se využije číslo 0, pro překážku 1, pro startovní pozici číslo 5 a pro cílovou pozici číslo 8. Dalším krokem je nastavení současné pozice na startovní pozici a následně se spouští samotný algoritmus vyhledávání cesty.

Všechny tyto kroky algoritmu jsou postupně ukládány 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. Kontrola se realizuje tak, že se provede rozhlédnutí po nejbližších souřadnicích v mapě a uloží se stav souřadnice ve všech čtyřech směrech. Poté co se provede rozhlédnutí provede se vyhodnocení této informace v případě, že se v těchto datech nachází souřadnice cíle je automaticky zvolena, v případě, že se nachází volné pozice je mezi těmito pozicemi náhodně vybráno, překážky jsou v tomto případě ignorovány. Pokud se nenachází žádné volná souřadnice ani cílová souřadnice je zapotřebí provést návrat na předchozí pozici, návrat je možný díky tomu, že pro každém kroku se označí předešlá souřadnice číslem 2. Množství souřadnic o které se provede návrat

je monitorováno a pokud je jejich množství příliš vysoké označí se tato cesta jako ztracená a začne se s vyhledáváním znovu. Z každého kroku je vygenerována mapa použitých kroků a je tedy možné sledovat přesně chování vyhledávacího algoritmu. Je také zaznamenáván počet kroků při navigaci, tato délka navigace je poté využita při určení ideální cesty.

Celý tento proces se opakuje do doby, než je splněna zastavovací podmínka algoritmu. Tato podmínka je počet opakování nejkratší sekvence. Z důvodu náhodné volby cesty je tedy nutné provádět tento proces opakovaně do nalezení nejkratší cesty momentálně je tento proces nastaven na 10 opakování, v budoucnu je zde potřeba vylepšit tuto podmínku tak, že se kontroluje konvergence k určité hodnotě a pokud se algoritmus dostane na dostatečně blízkou vzdálenost hledání označit za dostatečné. Momentálně použitá 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. Tato sekvence je převedena na JSON objekt 8.7. V tomto objektu jsou uloženy informace o směru letu pro dron. Tato data v JSON formátu jsou odeslána serveru jako odpověď po dokončení vyhledávání cesty, po odeslání těchto dat je celý tento podproces ukončen.



Obr. 8.7: Vývojový diagram hledání cesty

```
{
  "1":"FORWARD",
  "2":"LEFT",
  "3":"BACK",
  "4":"RIGHT",
  "ID":0,
  "size":3
}
```

Příklad obsahu generovaného JSON souboru pro navigaci

8.8 Ovládání dronu

Komunikace na hardwarové úrovni je u prototypu řešena pomocí protokolu wi-fi, kde dron je možné buď připojit k síti a nebo je možné provozovat samotný dron jako hot-spot. U prototypu byl vybrán způsob komunikace tak, že dron poskytuje uživateli hot-spot. K tomuto hot-spotu se poté uživatel připojí se serverovou aplikací. Toto řešení není použitelné pro budoucí využívání nicméně je tento způsob jednodušší pro testování, neboť připojování dronu k wi-fi není vždy konzistentní a vyžaduje zásah do operačního systému dronu. Změna hot-spotu dronu za externí wi-fi je poměrně složitá a vyžaduje zásah do firmwaru dronu nicméně na implementaci pro testování se osvědčilo využívat přímo dron jako zdroj sítě.

Ovládání dronu je z větší části řešeno pomocí knihovny ardrone-autonomy 6.6. 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. Knihovna si získává informace sama přímo z dronu a proto není přímo implementováno získávání telemetrie z dronu. 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 6.6 je také možné získat informace o stavu baterie, které by v budoucnu mohly sloužit pro automatický návrat na základu v případě nízkého napětí.

Knihovna se sama stará o stabilizaci dronu a jeho letovou kontrolu. Řízení letu v této knihovně probíhá tak, že se pro vytvoření letových misí používá vzdálenost, kterou dron urazí uražená vzdálenost si získává knihovna z čidel umístěných na dronu a z informací z těchto čidel a doby letu odhadne vzdálenost. Tato vzdálenost je ještě nadále kalibrována pomocí spodní kamery dronu. I když se jedná pouze o odhad vzdálenosti při testování bylo zjištěno, že při standardních klimatických podmínkách je tento odhad velice přesný

a odpovídá realitě, tuto odchylku je také možné ignorovat z toho důvodu, že stabilizace dronu při nízké letové výšce není tak kvalitní a dochází k neustálému pohybu dronu, který je ovlivňován vzduchovým prouděním a software se jej stále snaží udržet ve stabilní pozici.

Server tedy provede spuštění podprogramu pro kontrolu dronu metodou *fork*. Do podprogramu pro generování cesty je tedy zaslán název JSON souboru, který obsahuje navigační data vygenerovaná do podprogramu pro řízení dronu. Tato aplikace provede inicializaci knihoven pro kontrolu dronu a vytvoří jejich instance. Po vytvoření instancí těchto knihoven je vytvořena letová mise, která je prvkem knihovny 6.6 tato mise obsahuje veškeré příkazy, které bude dron vykonávat. Jako první příkaz se provede vynulování mise a zavolá se vzletová funkce, pro kterou se nastaví výška, momentálně je výška konstantní 1 m. Data získaná z JSON souboru je třeba parsovat a získat z nich letové informace. Tyto informace jsou uloženy do pole, které je v cyklu procházeno a jsou cyklicky vytvářeny jednotlivé prvky mise po přidání těchto směrových příkazů se do mise nastaví ještě přistávací proces, který provede přistání dronu. Vytváření jediné mise má jako nevýhodu nemožnost přerušit misi externě nicméně se osvědčilo jako lepší řešení než implementace samostatných misí pro každý směr letu. Tato část softwaru se při implementaci jevila jako nejproblematičtější a mnoho problémů je způsobeno využitím této knihovny. V případě, že nastane chyba během mise je mise automaticky ukončena a je zobrazena chybová hláška. Po ukončení mise ať už úspěšném či neúspěšném je tento proces ukončen a je navrácena kontrola serveru.

8.9 Loggování stavu aplikace

Důležitou součástí pro vývoj a údržbu jsou rutiny pro loggování aktivity této aplikace. Pro tuto funkci byla vytvořena loggovací třída *activity_logger* G. Tento logger je vlastně pole samostatných loggerů, které je možné libovolně přidávat a odstraňovat dle potřeby a také je možné ukládat veškeré informace do souboru popřípadě uložit pouze konkrétní logy z konkrétních částí kódu nebo provádět real-time loggování informací ať už do souboru nebo příkazové řádky.

Tento způsob ukládání informací o softwaru je v této aplikaci hojně využíván a byl extenzivně využíván při jejím vývoji. Veškerá chybová hlášení jsou stále řešena pomocí této funkcionality.

9

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, nastavení a je vysvětlen způsob ovládání systému.

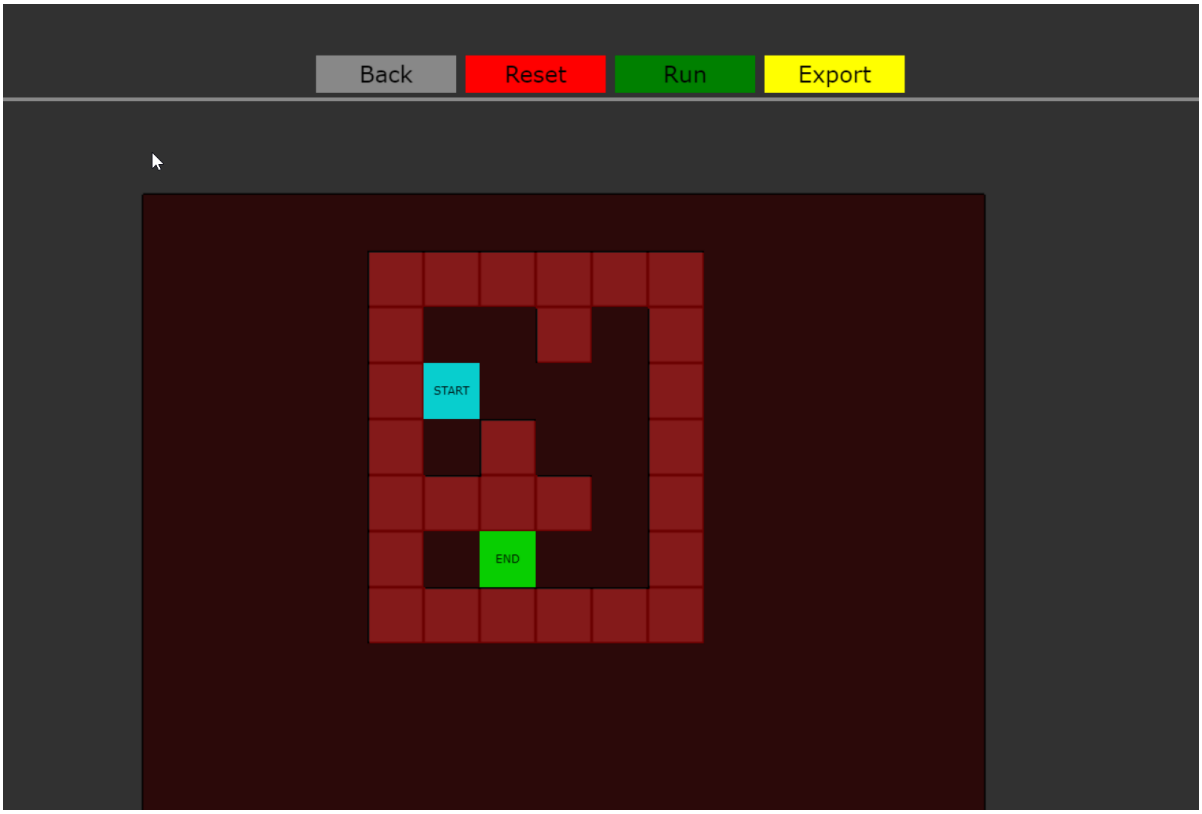
9.1 Nastavení serverové aplikace pro navigaci

Instalace tohoto systému je vysoce jednoduchá. Pro instalaci stačí pouze extrahovat archiv obsahující aplikaci do libovolného adresáře na disku ze kterého bude poté aplikace spouštěna. Pro zaručení funkčnosti u prototypu je momentálně nutné zapnout dron a připojit se k jeho wi-fi síti. Po provedení tohoto připojení je možné nastartovat server. Nastartování serveru se provede jednoduše spuštěním dávkového souboru RUN.bat v kořenovém adresáři aplikace. Po nastartování serveru je možné se připojit do uživatelského rozhraní z libovolného zařízení připojeného k této wi-fi síti. Na aplikaci uživatelského rozhraní je možné se dostat zapsáním ip adresy 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.),na této adrese se zobrazí grafické rozhraní pomocí kterého můžeme navigovat z nastavené mapy. V tomto grafickém rozhraní se bude provádět většina činností pro nastavení ovládání dronu a spouštění jeho činností.

9.2 Ovládání uživatelské aplikace

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 9.1 a umístit dron na startovní pozici relativně v prostoru pro navigaci. Nastavení se provede prostým kliknutím pravého tlačítka myši na startovní a cílové souřadnice na kontrolní mapě pod tlačítky a výběrem možností START a FINISH. 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. Možné je také resetovat všechny tlačítka pomocí tlačítka RESET v menu po kliknutí pravého tlačítka myši. 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. Také je možné stisknout tlačítko run a začít navigaci z již předem uložené poslední mapy. Po tomto procesu je možné pokračovat s novým nastavením mapy. Pro chod navigace už jen stačí dron umístit na startovní pozici v prostoru a stisknout tlačítko RUN je důležité brát v potaz orientaci dronu vzhledem k mapě.



Obr. 9.1: Nastavená kompletní aplikace připravená pro spuštění

10

Zhodnocení vypracované práce

Tato kapitola se zabývá zhodnocením výsledků práce a jejím porovnáním s jejím zadáním a navrženým systémem. Jsou zde prezentovány výsledky testování a porovnána navržená a implementovaná funkcionalita společně s odůvodněním odlišností od původního designu, je také navrženo řešení odstranění těchto problémů.

10.1 Průběh řešení práce

Práce na řešení této práce začala volbou hardwaru kde byl zvolen dron a volbou způsobu ovládání dronu, kde se jako nejpraktičtější ukázalo využití JavaScriptové knihovny 6.6 pro ovládání s frameworkem NodeJS 6.5. Po této volbě začala fáze vytváření designu aplikace a paralelně s tím prototypování jednotlivých systémů. Návrh byl postupně převeden do návrhového prostředí Papyrus za využití jazyku UML. Tento způsob práce se ukázal jako efektivní a proto byl při implementaci systému paralelně vytvářen diagram implementovaného systému jako jeho dokumentace. Pro zálohování a verzování systému bylo využito verzovacího systému GIT, tento systém dovoluje vytváření jednotlivých verzí systému s možností se vrátit na předchozí verzi popřípadě se oddělit na jinou branch systému a provádět paralelní vývoj. Další výhodou tohoto systému je, že bylo možné provádět off-site zálohy systému na serveru a na dalších počítačích, neboť při korektním využívání GITu je na všech počítačích používajících tento systém a repozitář stejná verze programu a pokud dojde k její ztrátě je možné ji vždy obnovit. Také je výhodou možnost dohledání změn s možností dohledání prvního výskytu chyb v softwaru.

Po této fázi začala samotná implementace ve fázi prototypování se ukázalo jako vhodné nepsat kód přímo v jazyce JavaScript 6.2, ale využít jazyk TypeScript 6.3 pro implementaci a poté překládat kód do JavaScriptu 6.2. Toto rozhodnutí bylo provedeno z důvodu jednoduššího způsobu zápisu podobnému jazyku Java a přehlednější práci s objekty. Samotná implementace začala implementací uživatelského rozhraní a webové stránky. Vývoj probíhal za využití debuggeru v prohlížeči Google Chrome. V první řadě bylo prototypováno a poté využito komponenty canvas 6.1.1 a následně byly navrženy a implementovány jednotlivé generické grafické prvky. Pro následné využití bylo vytvořeno ovládání těchto

prvků a zaručeno jejich ukládání stavu s korektním vykreslováním. Po ověření funkčnosti této stránky začala práce na implementaci serveru. U implementace serveru bylo zapotřebí zaručit funkčnost skriptů i při využití serveru a bylo tedy zapotřebí je upravit a vytvořit podporu pro obsluhu požadavků POST a GET z prohlížeče. V této fázi vývoje byla také vytvořena úvodní webová stránka. Po zprovoznění funkčnosti aplikace v prohlížeči bylo zapotřebí implementovat přenos mapy od uživatele na server. Tato část byla původně implementována s vlastním datovým formátem .dmap. Toto řešení se ukázalo jako vysoce neefektivní a problémové a proto bylo v pozdější fázi vývoje nahrazeno datovým formátem JSON 6.4 toto vyžadovalo zásadní změny ve funkci aplikace. Po zprovoznění této komunikace začala práce na implementaci hledání cesty v mapě a ovládání dronu.

Implementace serveru začala tím, že byla vytvořena obsluha požadavku GET od webového prohlížeče. Tato obsluha pouze poskytne prohlížeči html, javascript a css soubory, prohlížeč poté již sám stránku zobrazí, je zapotřebí pouze pozměnit cesty k těmto souborů na relativní k umístění spouštěcímu souboru serveru. Po implementaci obsluhy požadavku GET byla provedena implementace nástroje na ukládání souborů ze serveru. Tato rutina využívá funkce frameworku NodeJS k asynchronnímu ukládání souborů. Po vytvoření této rutiny byla implementována obsluha požadavku POST od webové aplikace. Tato obsluha byla vysoce problémová, neboť musela provádět obsluhu několika webových stránek. Nakonec bylo rozhodnuto, že tato obsluha bude rozdělena na cesty "/" a "/p" tyto cesty jsou přímo cesty v adresovém řádku prohlížeče a reakce na POST požadavek mezi nimi rozlišuje. Většina obsluhy je umístěna v cestě "/p" tato cesta přijme a uloží data z aplikace podle těchto dat se rozhodne, který z podprogramů aktivovat. Implementace podprogramů byla problematická z důvodů složitého debugování paralelních procesů, nakonec bylo debugování realizováno pomocí loggování výpisů těchto pod-aplikací. Kvůli přechodu na JSON formát bylo zapotřebí kompletně přepsat celý server a funkčnost reakcí na POST požadavky. Bylo zejména zapotřebí implementovat ukládání jednotlivých JSON souborů.

Implementace hledání cesty prošla několika fázemi vývoje, zde byl největším problémem algoritmus hledání cesty tento algoritmus stále není úplně funkční a stále má problémy s okrajovými případy při nastavení okrajových podmínek. U tohoto algoritmu bylo nutné zredukovat počet cyklů a množství kopi9 polí, které byli vytvářeny způsobovaly jeho vysokou neefektivitu. Zejména z těchto důvodů došlo na přechod z .dmap formátu na JSON formát tento formát zaručuje jednodušší přenos mezi komponenty a snížilo se množství cyklů potřebné pro získání dat z uloženého souboru. Další výhodou tohoto nového systému je, že je nyní možné uložit již vypočtenou trasu popřípadě data od uživatele a navigovat z již uložených dat.

Implementace ovládání dronu neustále prochází zásadními změnami nicméně současná podoba se již osvědčila. Zde bylo zapotřebí zvolit vhodnou knihovnu pro ovládání a poté upravit způsob doručování dat do této knihovny. Zásadní změna, která proběhla při implementaci tohoto modulu je využívání JSON formát pro přenos dat mezi serverem a pod-

procesem. Díky této změně je možné ukládat letová data pro kontrolu nebo znovuvyužití. Také byla změněna architektura serveru což zlepšilo možnost získávání informací modulů, neboť v prvotních fází vývoje byl tento modul volán z procesu pro hledání cesty a nebylo možné sledovat činnost tohoto modulu ani pomocí logů.

Během vývoje musela projít většina systémů a komponent projít prototypovací fází. V této fázi bylo zejména testováno hledání cesty v mapě a také bylo vytvořeno několik prototypů pro ukládání mapy a přenos na server. Pro návrh samotné mapy byl ze začátku vytvářen prototyp v jazyce Java z důvodu rychlejšího vývoje a lepších nástrojů pro rychlý vývoj. Dalším prototypovaným problémem bylo ovládání dronu, zde bylo třeba pomocí prototypů ověřit jakým způsobem fungují ovládací knihovny a jakým způsobem by bylo nejefektivnější předávat data pro ovládání. Zdaleka nejvíce prototypů bylo vytvořeno pro server, zde bylo zapotřebí definovat chování a zaručit kompatibilitu s webovými prohlížeči spolu s korektní odezvou na požadavky, spolu s problematickým debugováním tato část byla velice časově náročná.

10.2 Problémy při implementaci systému a jejich řešení

Během implementace bylo objeveno mnoho problémů vycházejících ať z předem vytvořeného designu tak také způsobených hardwarovými popřípadě softwarovými limitacemi. Největším limitujícím faktorem pro implementaci byl však čas. Z těchto důvodů bylo rozhodnuto vytvoření aplikace pouze pro navigaci v 2D prostoru s konstantní výškou letu. Jako další limitující faktor se ukázal způsob ovládání dronu zde bylo nakonec rozhodnuto využití knihovny pro ovládání dronu. Pro vývoj byl využíván dron AR-Drone 2.0, tento model dronu je již poměrně zastaralý a SDK poskytované výrobcem již není nijak aktualizováno a stále se potýká s problémy s kompatibilitou.

Problémem při vývoji bylo také objevení mnoha slepých cest, kde v některých případech bylo zapotřebí provést kompletní změnu způsobu implementace. Jako největší a nejdéle trvající slepá cesta se jevilo využití kontroléru Raspberry Pi jako základny a serveru. Toto řešení se ukázalo jako nevhodné poté co byla započata implementace serveru a vyhledávání cesty. Tyto funkce značně zpomalovaly běh systému i odezvu serveru a proto bylo rozhodnuto o nevyužití kontroléru Raspberry Pi. Raspberry Pi kontrolér měl být také nesen na samotném dronu, ale toto se také ukázalo jako vysoce nepraktické, protože bylo potřeba nést ještě baterii pro napájení Raspbery Pi a také samotné propojení a komunikace s dronem byla možná pouze pomocí wi-fi popřípadě se značným zásahem do hardwaru dronu. Z těchto důvodů bylo učiněno rozhodnutí, že bude využito stolní PC jako server pro hledání cesty i pro ovládání.

Další slepou cestou bylo vytváření vlastního formátu pro přenos dat mezi serverem a uživatelskou aplikací. Toto bylo řešeno pomocí vlastního datového formátu DMap. Tento formát se ukázal jako neefektivní a bylo zapotřebí jej nahradit efektivnějším formátem

a byl proto zvolen formát JSON. Bohužel přechodem na formát JSON vyžadovalo přepracování veškeré aplikace což způsobilo značnou časovou ztrátu při vývoji, neboť bylo nutné adaptovat veškerou komunikaci se serverem a algoritmy pro navigaci a hledání cesty.

Jako další problém při implementaci se ukázal pokus o implementaci čidel a využití kamer pro navigaci v prostoru. Implementace této funkcionality by vyžadovala kompletní změnu architektury a funkčnosti jak serveru, tak aplikace. Pro tuto činnost by bylo zapotřebí aby server odpovídat v reálném čase také na požadavky od dronu a pružně zpracovával všechna vstupní data od dronu a čidel. Toto se současnou architekturou není možné a vyžadovalo by to kompletní přepracování myšlenky a návrhu systému. Stejně tak zobrazování živého přenosu obrazu z dronu, zde je problémem nutnost implementace dalšího serveru pro přijímání těchto dat z dronu a jejich zpracování. Tento server by je poté mohl poskytovat ve formě webové stránky. Tato implementace by byla možná, ale stále by se nejednalo o data v přímém přenosu.

10.3 Testování systému

Většina komponent systému byla testována již při prototypování. Každá z jednotlivých komponent byla vytvářena a testována separátně na jejich funkčnost. Po otestování všech komponent byl systém integrován do celku a ten byl také testován nicméně testování systému jako celku bylo prováděno pouze povrchně z důvodu rychlého vybíjení baterie dronu a vysoké prodlevy mezi možnými cykly testování. Během testování se objevilo mnoho problémů zejména díky implementované knihovně pro navigaci. Samotná funkčnost systému byla testována pouze vytvářením jednoduchých scénářů pro navigaci neboť navigace v úzkých prostorách se u dronu takových rozměrů jako je použitý dron jeví jako vysoce nepraktická. V úzkých prostorách dochází ke změně vzduchového proudění a tím ztrátě stability tohoto dronu a může dojít až k nehodě. Dalším problémem při testování je nemožnost externě ovlivnit již vykonávanou misi dronu. Jakmile jsou dronu zaslány letové informace dron tyto informace vykonává a přerušení by vyžadovalo zásah přímo do operačního systému dronu. Dalším problémem je problém, kdy se dron zasekne u příkazu pro změnu výšky a dochází k neustálému a nepřerušitelnému stoupání do doby než je externě dron zastaven pomocí hrubé síly popřípadě otočen trupem vzhůru což provede jeho nouzové zastavení. Obecně tedy říci, že tento systém byl testován, ale v současné podobně je tato verze vhodná pouze pro testovací účely.

11

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.

11.1 Vylepšení hardwaru systému

Hardware současného implementovaného systému je potřeba pro budoucí řešení systému výrazně pozměnit. 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. Také by bylo vhodné zvolit novější verzi dronu, protože v současné době již pro momentálně používaný dron není podporováno SDK. 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. Také by bylo možné již kontrolér nevyužívat a využívat pouze server, který by sám prováděl všechny výpočty a pouze získával informace od dronu a ovládal jej. 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.

11.2 Vylepšení navigačního softwaru systému

V prvním kroku vylepšení software do budoucna je třeba odstranění využívané knihovny ardrone-autonomy a její nahrazení sofistikovanějším řešením. 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. Proto nebyla vytvářena nová knihovna, neboť se očekává, že každý uživatel s jiným typem dronu si vytvoří svoji ovládací část sám.

V současné době nejlepším vylepšení softwaru by bylo zlepšení získávání zpětné vazby od dronu přímo do serveru, 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. Také by bylo možné zanechat tento systém jako serverovou aplikaci, která by byla dostupná přímo z internetu. Pro zprovoznění této funkce by bylo zapotřebí využívání externí wi-fi sítě pro ovládání dronu a také úprava serveru, tak aby umožňoval přihlášení uživatele do systému a nebylo možné jej zneužít.

Uvedená vylepšení jsou pouze základními vylepšeními a bylo by možné najít mnohem více věcí pro zlepšení u tohoto systému, lze tedy říci, že pro další vývoj by bylo zapotřebí vytvořit prototyp nové generace založený na současném implementovaném řešení, ale stále vyžadující kompletní přepracování systému.

12

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 v prostoru.

V návrhové části do hloubky proveden návrh celého systému od hardwarového řešení po softwarové. Je navržený způsob komunikace a specifikovány parametry dronu. U návrhu softwaru je navržena architektura systému jako serverové aplikace. Je zde navrženo chování serveru, způsob komunikace serveru s jeho podprogramy a jsou detailně navrženy tyto podprogramy pro hledání cesty v mapě od uživatele a ovládání dronu. Je také proveden návrh grafického rozhraní jak pro grafické, tak po implementační stránce. Systém byl navržen po vývoj několika generací pro implementaci.

V implementační části je popsán využitý hardware pro implementaci dron AR Drone 2.0. V této části je také popsán využitý software pro vývoj jako je jazyk TypeScript a JavaScript a framework NodeJS. Je popsána implementace prvního prototypu určeného pro testovací účely a pro ověření proncipů, které byly navrženy v návrhové části. V další části je poté popsán způsob ovládání implementovaného systému a v následující části je popsán průběh implementace spolu s problémy při implementaci a zhodnocení této práce.

Tento systém je implementován jako systém pro navigaci podle mapy nastavené uživatelem a uložené v paměti. Implementace je provedena jako serverová aplikace kde se uživatelské rozhraní spouští pomocí webového prohlížeče. Pomocí tohoto grafického rozhraní aplikace uživatel nastaví mapu po které se má dron pohybovat spolu s počátkem a cílem cesty. Poté co je provedeno nastavení mapy spustí server podprogram pro vyhledávání cesty v této mapě a navigaci. Jsou vygenerována navigační data, která jsou předána dalšímu podprogramu, který provede samotné ovládání dronu podle zadané uložené mapy.

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 odstraněna a veškerou komunikaci provádí uživatel přímo se serverem 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.

Obecně lze říci, že tento systém vyžaduje podstatná vylepšení, ale cílem bylo pouze vytvoření prvního prototypu systému což se podařilo a je možné provádět navigaci v omezené míře. Problémem tohoto systému je, že momentálně zvolený dron je příliš rozměrný a nehodí se pro navigaci v interiéru neboť je příliš ovlivňován změnou vzduchového proudění od překážek což způsobuje značnou nestabilitu. Dalším problémem, je způsob implementace ovládání dronu, které momentálně nepodporuje převzetí ovládání uživatelem a úprava tohoto chování by vyžadovala změny ve firmwaru dronu. Tento systém je v současné době schopen navigace pouze v jednoduchých scénářích z důvodu ztráty stability což se negativně promítá na navigaci. Závěrem tedy říci, že vytvoření prvního prototypu sloužícího jako ověření navrženého designu se povedlo, ale tento prototyp ještě není připraven na používání a pro další využití je zapotřebí vytvoření další generace tohoto systému s novým designem založeným na tomto existujícím designu a řešícím nebo se vyhýbajícím chybám, které byly v této práci popsány.

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

[18] Mozzila.org *Canvas API*[online][Cit. 7.5.2018]. Dostupné z WWW: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

[19] Mozzila.org *JSON.parse()*[online][Cit. 7.5.2018]. Dostupné z WWW: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse

[20] Mozzila.org *JSON.stringify()*[online][Cit. 7.5.2018]. Dostupné z https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/str

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

Vytváření mřížky tlačítek

```
1  /**
2   * @description
3   * This class is for creating button grid
4   * It's draw function creates square grid of square buttons of
        btnLineCount count
5   */
6  class BtnGrid2 implements IDrawable
7  {
8      public context : CanvasRenderingContext2D;
9      public canvas : HTMLCanvasElement;
10
11     public x : number = 0;
12     public y : number = 0;
13     private btnW : number = 0;
14     private btnH : number = 0;
15     private btnGap : number = 0;
16     private color : string = "red";
17     private btnLineCount : number = 0;
18
19     private buttonArray : Array<Array<Button>>;
20     private helpArray : Array<Button>;
21
22     private menuBox;
23     private menuBoxOn : boolean;
24
25     private menuItem1;
26     private menuItem2;
27     private menuItem3;
28     private menuItem4;
29     private menuItem5;
30
31
32     constructor(context : CanvasRenderingContext2D, canvas :
           HTMLCanvasElement,
```

```
33      x : number, y : number, buttonWidth : number, buttonHeight : number
            ,
34      buttonGap : number, color : string, buttonLineCount : number)
35      {
36          this.context = context;
37          this.canvas = canvas;
38
39          this.x = x;
40          this.y = y;
41          this.btnW = buttonWidth;
42          this.btnH = buttonHeight;
43          this.btnGap = buttonGap;
44          this.color = color;
45          this.btnLineCount = buttonLineCount;
46
47          this.buttonArray = new Array<Array<Button>>();
48
49          this.menuBox = null;
50          this.menuBoxOn = false;
51
52          this.menuItem2 = null;
53          this.menuItem3 = null;
54          this.menuItem4 = null;
55          this.menuItem5 = null;
56          this.menuItem1 = null;
57
58          this.canvas.addEventListener("contextmenu", this.cMenu, false);
59
60          //prepare grid
61          for(var j : number = 0; j < buttonLineCount; j++)
62          {
63              this.helpArray = new Array<Button>();
64              for(var i : number = 0; i < buttonLineCount; i++)
65              {
66                  var btn : Button = new Button(this.context, this.canvas,
67                  (this.x+this.btnW/2+(i*this.btnW)),
68                  (this.y+this.btnH/2+(j*this.btnH)),
69                  this.btnW, this.btnH, this.color, " ", 12);
70
71                  this.helpArray.push(btn);
72              }
73              this.buttonArray.push(this.helpArray);
74          }
75      }
76
77      public draw = () : void =>
78      {
79          //draw all
```

```
80          for(var i : number = 0; i < this.buttonArray.length; i++)
81          {
82              this.helpArray = this.buttonArray[i];
83              for(var j : number = 0; j < this.helpArray.length; j++)
84              {
85                  this.helpArray[j].draw();
86              }
87          }
88      }
89
90      public getButtonArray() : Array<Array<Button>>
91      {
92          return this.buttonArray;
93      }
94
95      public cMenu = (event: MouseEvent) : void =>
96      {
97          var left = event.clientX;
98          var top = event.clientY;
99
100         this.menuBox = window.document.querySelector(".menu");
101         this.menuBox.style.left = left + "px";
102         this.menuBox.style.top = top + "px";
103         this.menuBox.style.display = "block";
104
105         this.menuBoxOn = true;
106         event.preventDefault();
107
108         var mnbx = this.menuBox;
109         var mnbxon = this.menuBoxOn;
110
111         var ba = this.buttonArray;
112             this.menuItem1 = window.document.querySelector(".menu-item1"
                );
113             this.menuItem2 = window.document.querySelector(".menu-item2"
                );
114             this.menuItem3 = window.document.querySelector(".menu-item3"
                );
115             this.menuItem4 = window.document.querySelector(".menu-item4"
                );
116             this.menuItem5 = window.document.querySelector(".menu-item5"
                );
117             //this.menuBox.addEventListener("mousedown", this.setField,
                false);
118
119             this.menuItem1.addEventListener("mousedown", function(e)
120             {
121                 dow(e, ba, mnbx, mnbxon);
```

```
122                 e.target.removeEventListener(e.type, arguments.callee);
123             }, false);
124 
125             this.menuItem2.addEventListener("mousedown", function(e)
126             {
127                 dos(e, ba, mnbx, mnbxon);
128                 e.target.removeEventListener(e.type, arguments.callee);
129             }, false);
130 
131             this.menuItem3.addEventListener("mousedown", function(e)
132             {
133                 doe(e, ba, mnbx, mnbxon);
134                 e.target.removeEventListener(e.type, arguments.callee);
135             }, false);
136 
137             //NOT YET USED
138             //this.menuItem4.addEventListener("mousedown", this.
                    setField4, false);
139 
140             this.menuItem5.addEventListener("mousedown", function(e)
141             {
142                 dor(e, ba, mnbx, mnbxon);
143                 e.target.removeEventListener(e.type, arguments.callee);
144             }, false);
145 
146             function dor(event, ba, mnbx, mnbxon)
147             {
148                 ba.forEach(element => {
149                     element.forEach(element => {
150                         element.isStart = false;
151                         element.isEnd = false;
152                         element.setUP();
153                         element.text = "";
154                         element.color = element.prevColor;
155                     });
156                 });
157 
158                 mnbx.style.display = "none";
159                 mnbx = false;
160             }
161 
162             function doe(event, ba, mnbx, mnbxon)
163             {
164                 ba.forEach(element => {
165                     element.forEach(element => {
166                         var x = event.x - this.canvas.offsetLeft;
167                         var y = event.y - this.canvas.offsetTop + window
                                .pageYOffset;
```

```
168
169                         if ((x > element.x - element.halfW) && (y >
                                element.y - element.halfH) &&
170                             (x < element.x + element.halfW) && (y <
                                    element.y + element.halfH))
171                         {
172                             if(element.isStart)
173                             {
174                                 element.isStart = false;
175                             }
176                             if(element.down)
177                             {
178                                 element.setUP();
179                             }
180                             element.isEnd = true;
181                             element.text = "END";
182                         }
183                     });
184                 });
185
186                 mnbx.style.display = "none";
187                 mnbx = false;
188             }
189
190             function dos(event, ba, mnbx, mnbxon)
191             {
192                 ba.forEach(element => {
193                     element.forEach(element => {
194                         var x = event.x - this.canvas.offsetLeft;
195                         var y = event.y - this.canvas.offsetTop + window
                                .pageYOffset;
196
197                         if ((x > element.x - element.halfW) && (y >
                                element.y - element.halfH) &&
198                             (x < element.x + element.halfW) && (y <
                                    element.y + element.halfH))
199                         {
200                             if(element.isEnd)
201                             {
202                                 element.isEnd = false;
203                             }
204                             if(element.down)
205                             {
206                                 element.setUP();
207                             }
208                             element.isStart = true;
209                             element.text = "START";
210                         }
```

```
211                     });
212                 });
213 
214                 mnbx.style.display = "none";
215                 mnbx = false;
216             }
217 
218             function dow(event, ba, mnbx, mnbxon)
219             {
220                 ba.forEach(element => {
221                     element.forEach(element => {
222                         var x = event.x - this.canvas.offsetLeft;
223                         var y = event.y - this.canvas.offsetTop + window
                                .pageYOffset;
224 
225                         if ((x > element.x - element.halfW) && (y >
                                element.y - element.halfH) &&
226                             (x < element.x + element.halfW) && (y <
                                element.y + element.halfH))
227                         {
228                             if(element.down)
229                             {
230                                 element.setUP()
231                             }
232                             else
233                             {
234                                 element.setDOWN();
235                             }
236                         }
237                     });
238                 });
239 
240                 mnbx.style.display = "none";
241                 mnbx = false;
242             }
243     }
244 }
```

Příloha C

Exportování dat z mapy pro server

```
1  /**
2   * This function creates message for server about buttond grid buttons
        and sends it to server
3   *
4   * @param UI UI control button UI grid
5   * @param grid button grid for navigation
6   */
7  function exportButtonUI(UI : ControlButtonUI, grid : ControlGrid,
       targetList : Array<DropTarget>, targeGrid : TargetGrid) : boolean
8  {
9      var exportButton : BButton = UI.getExportButton();
10     var bGrid : BtnGrid2 = grid.getButtonGrid();
11
12     var mapValidity : boolean = false;
13
14     if(exportButton.clicked == true)
15     {
16
17         mapValidity = checkMapValidity(grid);
18         if(mapValidity == false)
19         {
20             exportButton.clicked = false;
21             return false;
22         }
23
24         var helpArray = grid.getButtonGrid().getButtonArray();
25
26         var id = 1;
27         var startCo;
28         var endCo;
29         var coords = new Array<Coordinate>();
30
31         for(var i = 0; i < helpArray.length; i++)
32         {
```

```
33              for(var j = 0; j < helpArray[0].length; j++)
34              {
35                  if(helpArray[i][j].isStart)
36                  {
37                      startCo = new Coordinate(0, i, j, 0, true, "START");
38                  }
39                  if(helpArray[i][j].isEnd)
40                  {
41                      endCo = new Coordinate(1, i, j, 0, true, "END");
42                  }
43                  if(helpArray[i][j].down)
44                  {
45                      coords.push(new Coordinate(id++, i, j, 0, true, "X")
                            );
46                  }
47              }
48          }
49
50          var map : RoomMap = new RoomMap(0, helpArray.length, helpArray.
                length, startCo, endCo, "map", coords);
51
52          var outputString : string = map.returnJSONFormat();
53
54          //uncomment for user logging
55          console.log(outputString);
56
57          exportButton.clicked = false;
58
59          //SEND DATA TO SERVER
60          SendData(serverURL, outputString);
61
62          //Uncomment for serverside logging
63          //SendData(serverURL, "!NODELIST!");
64
65          return true;
66      }
67
68      return false;
69  }
```

Příloha D

Implementace hledání cesty v mapě

```
1  /**
2   * This method iterates for set amount of trys and returns shortest
        possible path in direction data
3   *
4   * @param number count of iterations wanted before getting data
5   * @return Array<number> direction data for shortest possible path after
        @param iterations
6   */
7  public createDirectionData = (numIterations : number) =>
8  {
9      var currentIterationData;
10     var currentMapStepList : Array<Array<Array<number>>>;
11     var currentPath : Array<number>;
12     var currentIterations : number = 0;
13     var finalIterationData;
14     var mapStepList : Array<Array<Array<number>>>;
15     var shortestPath : Array<number>;
16     var leastIterations : number = null;
17     var iter : number = 0;
18
19     currentIterationData = this.iteratePath();
20     currentIterations = currentIterationData.iterationCounter;
21     currentPath = currentIterationData.directionData;
22     currentMapStepList = currentIterationData.mapStepList;
23
24     finalIterationData = currentIterationData;
25     leastIterations = currentIterations;
26     mapStepList = currentMapStepList;
27     shortestPath = currentPath;
28
29
30     while(iter < numIterations - 1)
31     {
32         currentIterationData = this.iteratePath();
```

```
33          currentIterations = currentIterationData.iterationCounter;
34          currentPath = currentIterationData.directionData;
35          currentMapStepList = currentIterationData.mapStepList;
36
37          if(currentIterations == 111111)
38          {
39              continue;
40          }
41          if(currentIterations == leastIterations)
42          {
43              finalIterationData = currentIterationData;
44              leastIterations = currentIterations;
45              mapStepList = currentMapStepList;
46              shortestPath = currentPath;
47              iter++;
48          }
49          if(currentIterations < leastIterations)
50          {
51              finalIterationData = currentIterationData;
52              leastIterations = currentIterations;
53              mapStepList = currentMapStepList;
54              shortestPath = currentPath;
55              iter = 1;
56          }
57      }
58      return finalIterationData;
59  }
```

Příloha E

Implementace Třídy RoomMap

```
1  /*
2   * File: RoomMap.ts
3   *
4   * Author : Dominik Pauli
5   *
6   * Year : 2018
7   *
8   * Description: This is data type that stores whole map of one room
9   *
10  */
11 import { Coordinate } from "./Coordinate";
12 import { StartCoordinate } from "./StartCoordinate";
13 import { EndCoordinate } from "./EndCoordinate";
14 import { start } from "repl";
15
16 export class RoomMap
17 {
18     private ID : number = null;
19     private magiNumber : number = null;
20     private mapWidth : number = null;
21     private mapHeight : number = null;
22     private startPos : StartCoordinate = null;
23     private endPos : EndCoordinate = null;
24     private optionalString : string = null;
25     private coordinateList : Array<Coordinate> = null;
26     private data = null;
27
28     constructor(id : number, mapWidth : number, mapHeight : number,
           startPosition : Coordinate, endPosition : Coordinate,
           optionalString : string, coordinateArray : Array<Coordinate>)
29     {
30         this.ID = id;
31         this.mapWidth = mapWidth;
32         this.mapHeight = mapHeight;
```

```
33          this.startPos = startPosition;
34          this.endPos = endPosition;
35          this.optionalString = optionalString;
36          this.coordinateList = coordinateArray;
37
38          this.data =
39          {
40              ID: this.ID,
41              WIDTH: this.mapWidth,
42              HEIGHT: this.mapHeight,
43              START: this.startPos,
44              END: this.endPos,
45              idString: this.optionalString,
46              markers: this.coordinateList
47          }
48      }
49
50  //SHORTENED BY REMOVING GETTERS AND SETTERS
51
52      /**
53       * This method returns coordinate list
54       *
55       * @return coordinate array
56       */
57      public getCoordList = () : Array<Coordinate> =>
58      {
59          return this.coordinateList;
60      }
61
62      /**
63      * This method returns coordinates in JSON format
64      *
65      * @return coordinates in JSON format
66      */
67      public returnJSONFormat = () : string =>
68      {
69          return JSON.stringify(this.data);
70      }
71
72  }
```

Příloha F

Implementace Třídy Coordinate

```
1  /*
2   * File: Coordinate.ts
3   *
4   * Author : Dominik Pauli
5   *
6   * Year : 2018
7   *
8   * Description: This is data type that stores single coordinate
9   *
10  */
11 export class Coordinate
12 {
13     private ID : number = null;
14     private x : number = null;
15     private y : number = null;
16     private z : number = null;
17     private optionalString : string = null;
18     private valid : boolean = false;
19     private data = null;
20
21     constructor(id : number, x : number, y : number, z : number,
22         valid : boolean, idString : string)
23     {
24         this.ID = id;
25
26         this.x = x;
27         this.y = y;
28         this.z = z;
29
30         this.valid = valid;
31
32         this.optionalString = idString;
33
34         this.data =
```

```
35          {
36              ID: this.ID,
37              valid: this.validityAsNumber(),
38              X: this.x,
39              Y: this.y,
40              Z: this.z,
41              idstring: this.optionalString
42          }
43      }
44
45  //SHORTENED BY REMOVING GETTERS AND SETTERS
46
47      /**
48       * This method returns validity
49       *
50       * @return number stored in z
51       */
52      public isValid = () : boolean   =>
53      {
54          return this.valid;
55      }
56
57      private validityAsNumber = () : number =>
58      {
59          if(this.isValid())
60          {
61              return 1;
62          }
63          return 0;
64      }
65
66      /**
67       * This method returns ID
68       *
69       * @return ID
70       */
71      public getID = () : number =>
72      {
73          return this.ID;
74      }
75
76      /**
77       * This method returns coordinates in JSON format
78       *
79       * @return coordinates in JSON format
80       */
81      public returnJSONFormat = () : string =>
82      {
```

```
83          return JSON.stringify(this.data);
84      }
85
86      /**
87       * This method prints stored variables to console
88       */
89      public printData = () : void =>
90      {
91          console.log(this.ID + ": " + "[" + this.x + ", " + this.y + ", "
                + this.z + "] " + this.valid + " " +this.optionalString + "\
               r\n");
92      }
93
94  }
```

Příloha G

Implementace loggeru

```
1  import { Logger } from "./Logger";
2  import { FileHelper } from "./FileHelper";
3
4  export class ActivityLogger
5  {
6      private loggers : Array<Logger> = null;
7      private lgrNum : number = null;
8
9      constructor()
10     {
11         this.loggers = new Array<Logger>();
12         this.lgrNum = 0;
13     }
14
15     public attach = (lgr : Logger) : number  =>
16     {
17         this.loggers.push(lgr);
18         this.lgrNum++;
19
20         return this.lgrNum;
21     }
22
23     public destroy = (logNR : number) : void =>
24     {
25         this.loggers.splice(logNR, 1);
26     }
27
28     public getSingleLog = (logNR : number) : Logger =>
29     {
30         return this.loggers[logNR];
31     }
32
33     public storeAllLogs = (path : string) : void =>
34     {
```

```
35          var completeData : string = "";
36          this.loggers.forEach(element => {
37              completeData += element.getLog();
38          });
39
40          let FH = new FileHelper();
41          FH.setPath(path);
42          FH.setText(completeData);
43          FH.createFile();
44      }
45
46      public storeSingleLog = (path : string, logNR : number) : void =>
47      {
48          this.loggers[logNR].outputFile(path);
49      }
50
51      public outputAllLogs = () : void =>
52      {
53          var completeData : string = "";
54          this.loggers.forEach(element => {
55              completeData += element.getLog();
56          });
57
58          console.log(completeData);
59      }
60
61      public outputSingleLog = (logNR : number) : void =>
62      {
63          this.loggers[logNR].outputConsole();
64      }
65  }
```