

Slovenská technická univerzita v Bratislave  
Fakulta informatiky a informačných technológií

# Projektová dokumentácia

Tím 1: LEGENDRONE

---

<b>Akademický rok</b>	2016/17
<b>Predmet</b>	Tímový projekt
<b>Študenti</b>	Bc. Tomáš Čičman Bc. Dušan Drábik Bc. Dušan Hatvani Bc. Patrik Kadlečík Bc. Mário Keszeli Bc. Martin Mocko Bc. Lukáš Visokai
<b>Vedúci tímu</b>	Ing. Viktor Šulák

# Obsah

## DOKUMENTÁCIA INŽINIERSKEHO DIELA

---

<b>7 Úvod</b>	<b>23</b>
<b>8 Globálne ciele</b>	<b>24</b>
<b>9 Analýza</b>	<b>26</b>
<b>10 Architektúra</b>	<b>31</b>
<b>11 Implementácia</b>	<b>39</b>
<b>12 Testovanie</b>	<b>42</b>
<b>Prílohy</b>	<b>43</b>
C Používateľská príručka	43
D Technická dokumentácia	46

# **Dokumentácia inžinierskeho diela**

## 7 Úvod

Obsahom tejto dokumentácie je pohľad na technickú stránku projektu vypracovaného v rámci predmetu Tímový projekt na tému *Simulácia správania bezpilotných lietadiel v roji*. Téma je pokračovaním minuloročného projektu. Cieľom je zdokonalenie nástroja na simuláciu správania bezpilotných zariadení v roji, zlepšenie ich vzájomnej komunikácie a rozšírenie simulácie o senzory umožňujúce efektívne plnenie úloh.

Kapitola 8 opisuje ciele, ktoré sme si stanovili pre zimný semester. Obsahuje tiež požiadavky na systém a globálne ciele. Základným cieľom pre zimný semester bolo vytvorenie MVP prototypu aplikácie. V letnom semestri sme stavali na základoch položených v zimnom semestri a rozšírili sme možnosti simulátora.

Obsahom kapitoly 9 je analýza projektu. V jej časti sa venujeme analýze existujúceho projektu spolu s vykonanými zmenami. Analýza obsahuje aj výsledky rešerše využitia bezpilotných lietadiel spolu s rôznymi senzormi, ktorými bezpilotné lietadlá disponujú.

Kapitola 10 sa zameriava na celkový pohľad na nový systém, ktorého súčasťou je architektúra, diagram tried a moduly systému. Jednotlivé časti sú popísané a vyobrazené na diagramoch v tejto kapitole.

## 8 Globálne ciele

### 8.1 Požiadavky produktového vlastníka na projekt

- definovanie UAV (určovanie pozície pomocou GPS, pohyb - rýchlosť, smerový vektor, kapacita a spotreba batérie)
- navrhnutie a vytvorenie spoľahlivej komunikácie (medzi-komponentová)
- definovanie senzorov (implementovať základné senzory, navrhnuť systém pre pridanie rôznych senzorov)
- definovanie prostredia (vytvorenie mapy a objektov reálneho sveta, simulovanie poveternostných podmienok)
- kolízie objektov s UAV
- navrhnuť a implementovať komunikáciu medzi UAV (na aplikačnej vrstve, rádius dosahu komunikácie)

### 8.2 Ciele pre zimný semester

#### Hlavné ciele pre zimný semester pred šprintom číslo 3

- analýza preberaného projektu
- pochopenie a zdokumentovanie preberaného projektu (refaktoring, komentovanie zdrojového kódu a príprava na doimplementovanie nových funkcionalít)
- zefektívnenie správania celého systému

Po šprinte č. 2 pri prediskutovaní kladov a záporov projektu bývalého tímu, sme sa rozhodli, po odsúhlasení vedúcim projektu, vytvoriť simulátor nanovo. Po tomto rozhodnutí sme si definovali nové požiadavky na systém a MVP pre zimný semester.

#### MVP pre zimný semester

Za cieľ v zimnom semestri sme si zvolili vytvorenie prototypu aplikácie, ktorý bude spĺňať základné funkcionality:

- navrhnutie modulárneho systému pre jednoduché pridávanie nových funkcionalít
- inicializácia prostredia (mapa, základňa, UAV)
- vytvorenie UAV a jeho základný pohyb po mape
- definovanie správ medzi dvoma UAV ako aj medzi UAV a základňou
- simulácia funkčnosti preposielania a zachytávania akýchkoľvek správ
- zobrazenie správneho správania sa aplikácie formou logov a zobrazenie do grafického používateľského rozhrania

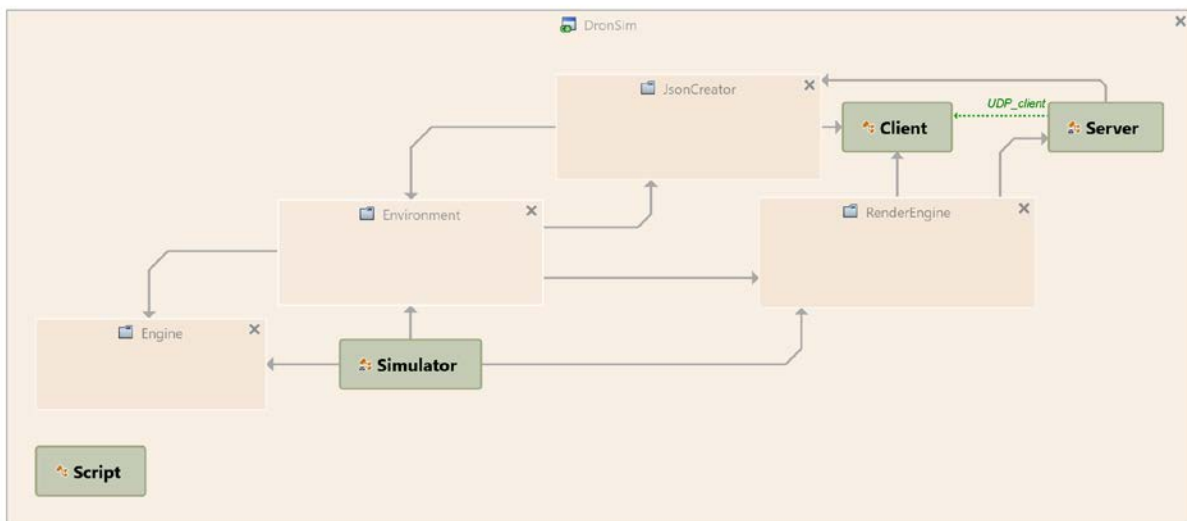
## **8.3 Ciele pre letný semester**

Po analýze existujúceho riešenia bývalého tímu a rozhodnutí implementovať vlastné riešenie v zimnom semestri sme následne v tom letnom prijali za cieľ rozširovanie funkcionality aplikácie nad rámec MVP podľa požiadaviek produktového vlastníka.

# 9 Analýza

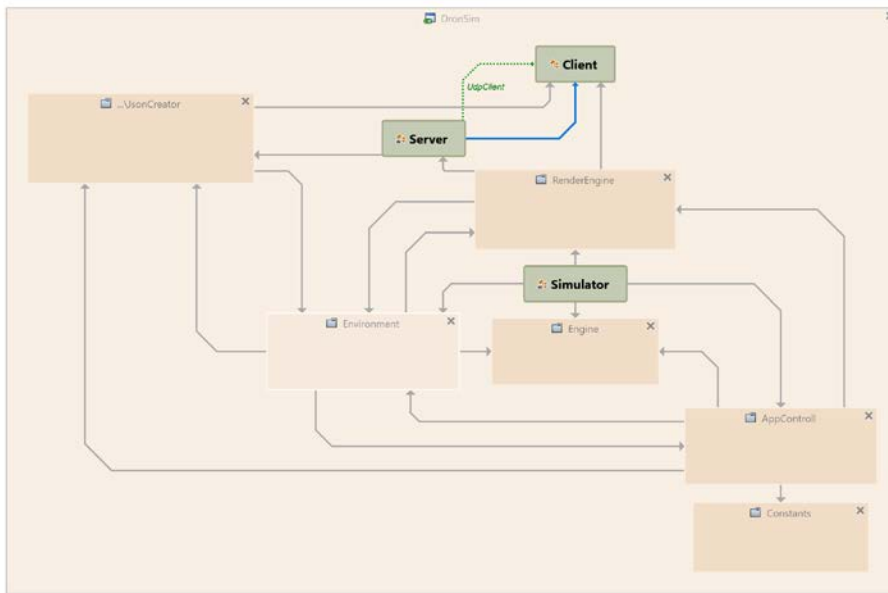
## Analýza existujúceho riešenia

Pri analýze existujúceho riešenia bývalého tímu sme začali s naštudovaním dokumentácie a zdrojového kódu. Pri študovaní dokumentácie sme zistili, že z nezanedbateľnej časti nezodpovedá dodanému projektu a jeho zdrojovému kódu. Diagram tried v dokumentácii taktiež nereprezentoval skutočnú architektúru a štruktúru kódu. Konflikt sme objavili aj pri triede zodpovednej za spúšťanie celého simulátora. Zatiaľ, čo v dokumentácii je uvedené, že trieda Engine je spúšťačom celej aplikácie (konkrétne metóda `onStart(script: Script): int`), po inšpekcii zdrojového kódu triedy Engine sme zistili, že táto trieda je prázdna a nič nespúšťa. Následne sme pokračovali s naštudovaním zdrojového kódu bez dokumentácie. Zistilo sa pri tom, že spúšťačom aplikácie je metóda `Main(string[] args)` v triede Simulator. V tejto triede sa vykonávalo vytvorenie prostredia (Environment) a jeho inicializácia. Následne sme po dôkladnej analýze zdrojového kódu pristúpili k zlepšovaniu kvality kódu (refaktoring). Obrázok 1 zachytáva pohľad na pôvodnú vysokoúrovňovú štruktúru modulov a prepojení.



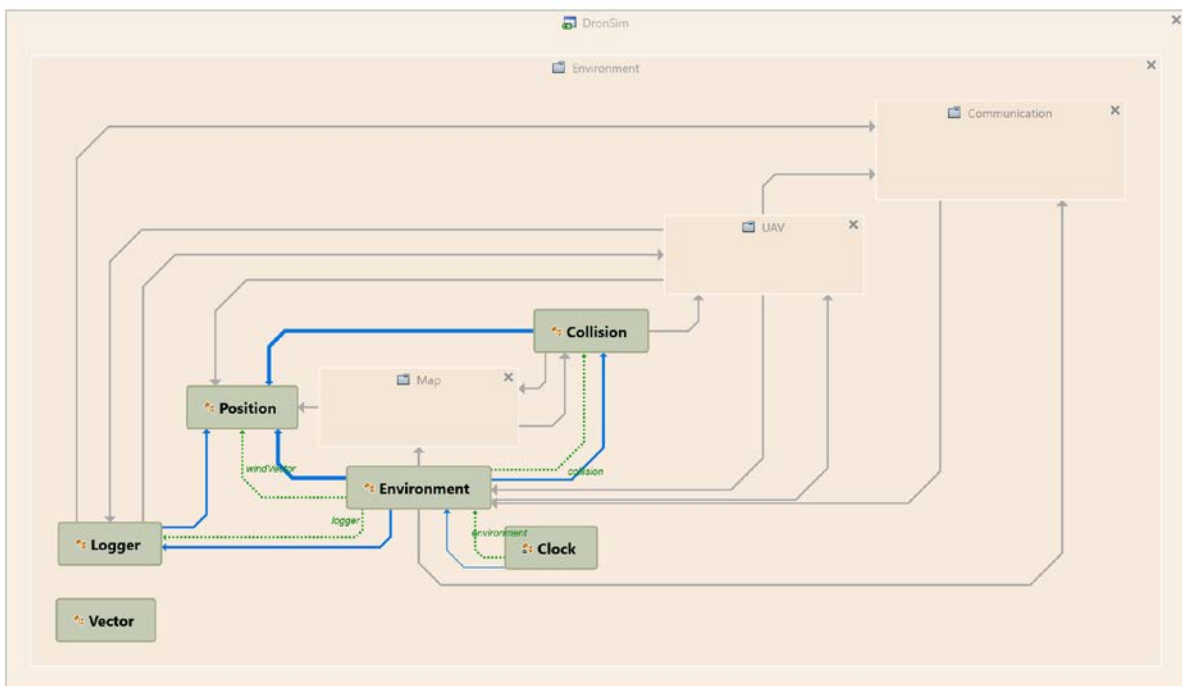
Obr. 1: Celkový pohľad na vysokoúrovňovú štruktúru pôvodného projektu bývalého tímu

Na obrázku 2 vidno prepracovanú štruktúru prepojení a modulov spolu s ich vzťahmi po našej úprave pôvodného kódu. Spúšťačom aplikácie zostala trieda Simulator, ktorá volala metódu `Main(string[] args)`. V nej sa vytvorí objekt triedy AppControl inicializujúci Environment a prípadne aj Unity pre grafické používateľské rozhranie simulátora. Oproti pôvodnej štruktúre pribudli dva moduly AppControl a Constants.



Obr. 2: Celkový pohľad na vysokoúrovňovú štruktúru upraveného projektu

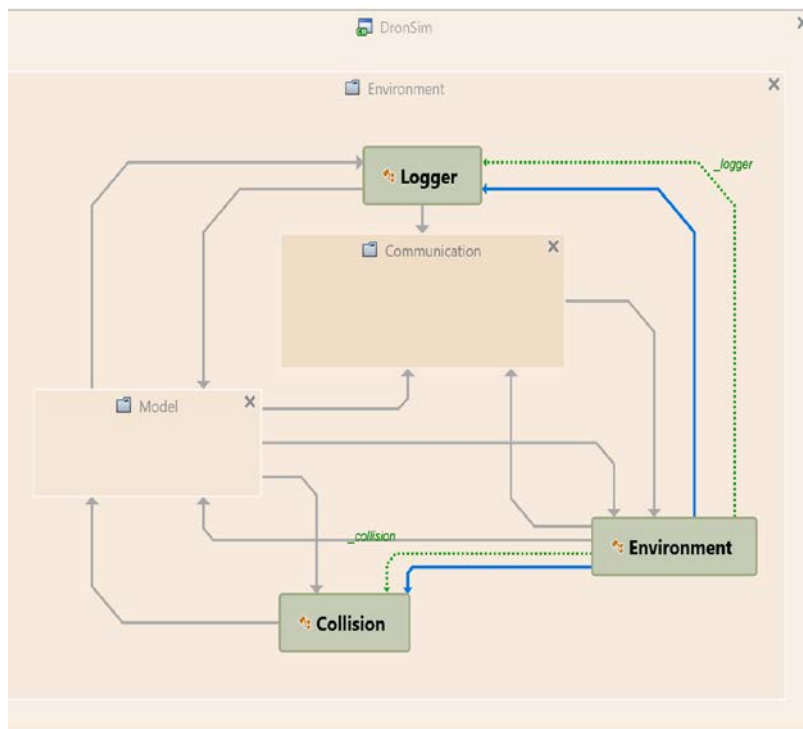
Obrázok 3 znázorňuje pôvodnú štruktúru modulu Environment, ktorý obsahoval navyše zložky UAV, Communication a Map.



Obr. 3: Architektúra modelu Environment pôvodného projektu



Obrázok 4 zobrazuje upravenú modulu Environment. V upravenej časti sme nechali pôvodnú zložku Communication, ale pridali sme novú zložku Model. Spravili sme tak pre lepšiu organizáciu tried do skupín, ktoré spolu súvisia. Zložka Model obsahuje dve ďalšie zložky, ktoré boli pôvodne v modeli Environment - zložky UAV a Map. UAV navyše obsahuje novú súčasť UAV\_State, ktorá definuje aktuálny stav UAV.



Obr. 4: Architektúra modelu Environment po úprave a refaktoringu projektu

## Analýza použitia bezpilotných lietadiel

Bezpilotné lietadlá (angl. unmanned aerial vehicle - UAV) sú v súčasnosti čoraz viac rozšírené v rôznych oblastiach a spôsoboch použití. Vývoj dronov bol iniciovaný pre armádne využitie najmä kvôli ochrane vlastných pilotov. Môžu byť pri tom nasadené na špehovanie nepriateľského prostredia pomocou rôznych senzorov, prípadne ochranu hraníc, ale aj ako útočné lietadlá. Použitie na súkromné účely najčastejšie zahŕňa natáčanie videa a fotografovanie. Vo verejnej sfére však pre ne existujú podstatne širšie možnosti nasadenia. Obrovský prínos predstavujú napríklad pre záchranné alebo policajné zložky, ktorým uľahčujú prehľadávanie prostredia aj v náročnom teréne, či pri nevyhovujúcich poveternostných podmienkach. Veľkou výhodou je cena UAV, keďže náklady na jeho obstaranie sú výrazne nižšie ako napríklad pri obstarávaní helikoptéry. Takisto je známe aj využitie týchto zariadení v poľnohospodárstve, či už pri mapovaní poľnohospodárskej pôdy, alebo pri chove zvierat, kde sa pomocou bezpilotných lietadiel môže mapovať pohyb a poloha zvierat a taktiež ochraňovať ich napr. pred pytlíkmi. Taktiež majú veľký význam pre meteorológov. Bepilotné lietadlá bez ohrozenia ľudskej posádky možno vyslať na prieskum hurikánov a analyzovať a spracovávať nazbierané dáta v reálnom čase. V rôznych sférach je tiež užitočné prehľadávanie miestností rojom dronov alebo vytváranie mapy priestorov prípadne aj v spolupráci s UGV (pozemné vozidlo).

## Rešerš použitia senzorov v bezpilotných lietadlách

Existuje množstvo senzorov, ktoré je možné pripojiť k dronom a tým zvýšiť ich efektívnosť pri plnení úloh. Základným prvkom drona je systém, pomocou ktorého sa dokáže orientovať v danom prostredí. Väčšina dronov používa globálny satelitný navigačný systém (GNSS), ktorého hlavnou časťou môže byť GPS (Global Positioning System), satelitná navigácia (INS) alebo laserový diaľkomer. Pri plnení rôznych typov úloh ako simultánna lokalizácia a mapovanie vnútorných priestorov sa používajú miniatúrne laserové senzory (HokuyoURG a IMU Xsens MTI-G) a ultrazvukové senzory. Na orientáciu vo vnútornom prostredí objektu sa používa magnetometer tiež nazývaný elektronický kompas. Ako už bolo spomenuté, bohaté uplatnenie nachádzajú UAV v armáde obzvlášť pri sledovaní cieľov. Úlohy týkajúce sa sledovania cieľov môžeme rozdeliť na dve skupiny:

1. Automatické sledovanie statických cieľov
2. Sledovanie pohyblivých cieľov pohybujúcich sa rýchlosťou do 5 km/h

Pri automatickom sledovaní statických cieľov sa väčšinou používajú tri typy systémov:

1. Systém GALIFIR – hlavnou zložkou systému je infra-kamera s teleobjektívom s dvomi zornými poľami.
2. Systém STIS (Stabilised Thermal Imaging System) – hlavnou zložkou tohto systému je senzor MINI-FLIR s teleobjektívom s tromi ohniskovými vzdialenosťami, ten je doplnený nízkoúrovňovou CCD TV kamerou a laserovým diaľkomerom.
3. MKD 400/600 elektro-optické zariadenie, ktoré obsahuje infra-kameru a CCD TV kameru

Ďalšou možnosťou využitia v armáde je sledovanie pohyblivých cieľov pohybujúcich sa rýchlosťou do 5 km/h. Pri tomto type úlohy sa používajú systémy, ktorých hlavnou zložkou je radar. Často používané systémy na sledovanie pohyblivých cieľov:

1. SAR (Synthetic Aperture Radar) radar
2. TESAR (Tactical Endurance Synthetic Aperture Radar) – systém obsahujúci zariadenie pre automatické rozpoznávanie pozemných cieľov
3. STARLOS (SAR Target Recognition and Location System)
4. EL/M-2055 radar
5. Rádiolokačný senzor AWARD (All Weather Airborne Reconnaissance Drone Sensor)

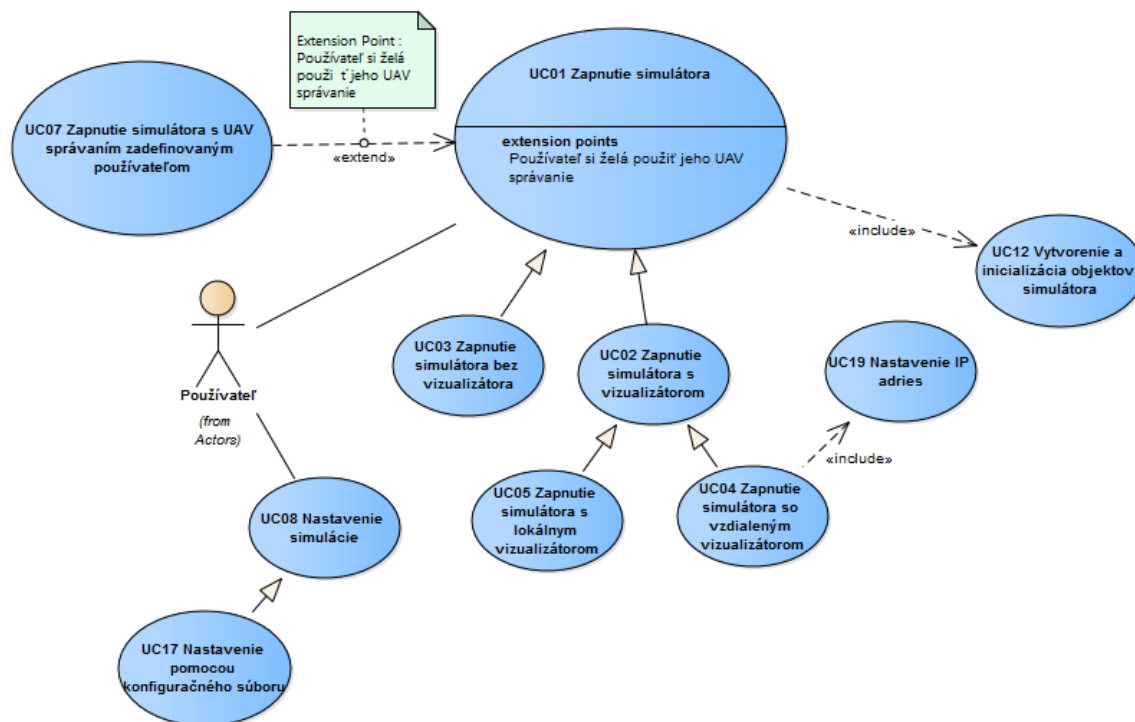
Armádne nasadenie dronov bolo hlavnou motiváciou k skonštruovaniu dronov. Popri použití v armáde však existuje mnoho ďalších odvetví, kde nájdu svoje uplatnenie. Medzi ne patrí poľnohospodárstvo, monitorovanie ovzdušia, monitorovanie znečistenia pôdy, resp. vody, monitorovanie sopečných aktivít, či monitorovanie poškodenia elektrického vedenia. Drony používané na meranie koncentrácie plynu v ovzduší sú vybavené GSS (Gas Sensing System) systémom, ktorý pracuje na základe MEMS senzoru plynu USM-VOC 3.0 od firmy Unitronic. Tento senzor meria koncentráciu CO<sub>2</sub>, NO<sub>2</sub> a SO<sub>2</sub> v ovzduší. Na monitorovanie nebezpečných chemických a toxických látok v prostredí sa používajú drony obsahujúce LSCAD (Lightweight Stand-off Chemical Agent Detector) detektor, ktorý obsahuje televíznu kameru a infračervený interferometer.

Drony monitorujúce morské prostredie, morské aktivity a riziká (napr. tsunami) sú vybavené video kamerou s vysokým rozlíšením, výškovým senzorom, senzorom barometrického tlaku, senzorom vlhkosti a meteorologickým senzorom. Ďalším dôležitým odvetvím, kde sa drony používajú, je monitorovanie znečistenia vody. Pri tomto type úlohy nesú drony fotoaparát Canon 50D spolu s niekoľkými multispektrálnymi kamerami. Na monitorovanie sopečných aktivít v oblastiach s vysokou pravdepodobnosťou výskytu sopečnej aktivity sa používajú drony s termálnymi kamerami TC36000. Tieto typy kamier sledujú teplotu bahennej pôdy popri sopkách.

Zaujímavým využitím dronov a ich senzorov je použitie na monitorovanie typu vegetácie a pôdy v ťažko dostupných lokalitách. Takéto drony obsahujú multispektrálne kamery, ktoré pracujú na princípe odrazu svetla od rôznych typov vegetácie. Pri monitorovaní fotovoltaiických systémov a elektrického vedenia sú drony vybavené termovíznymi kamerami, ktoré detegujú praskliny a detegujú oblasti s vysokou teplotou. Drony monitorujúce elektrické vedenie používajú ultrafialové kamery a infračervené kamery typu FLIR, ktoré majú za úlohu merať infračervené emisie v okolí elektrických vodičov.

# 10 Architektúra

V tejto kapitole rozoberieme podrobný návrh novej architektúry aplikácie.

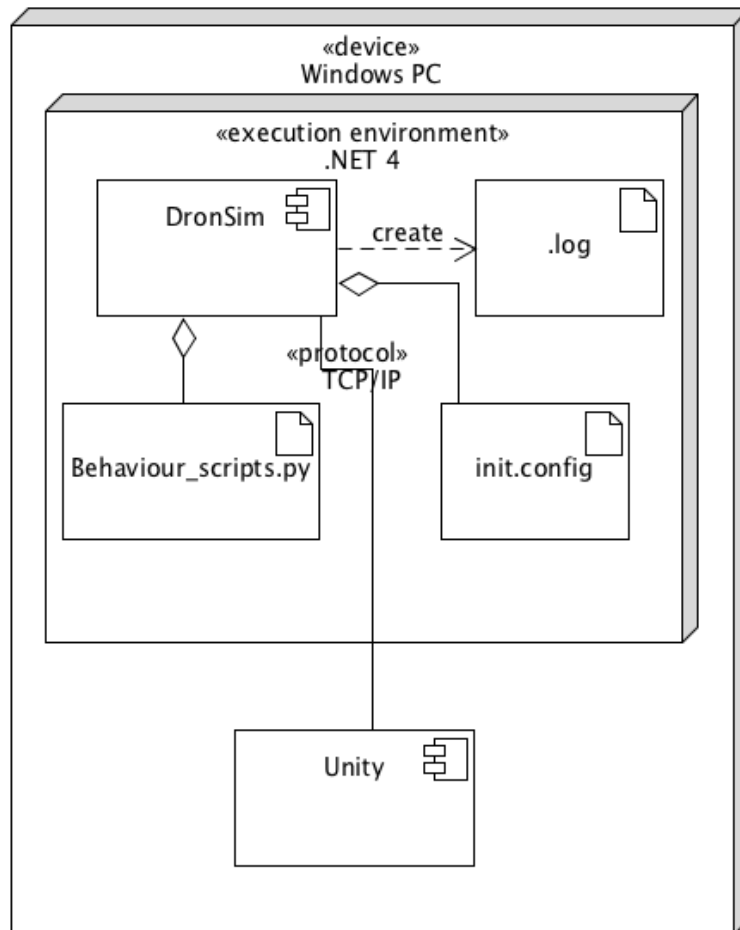


Obr. 5: Diagram prípadov použitia – prehľad funkcionality

## Diagram nasadenia

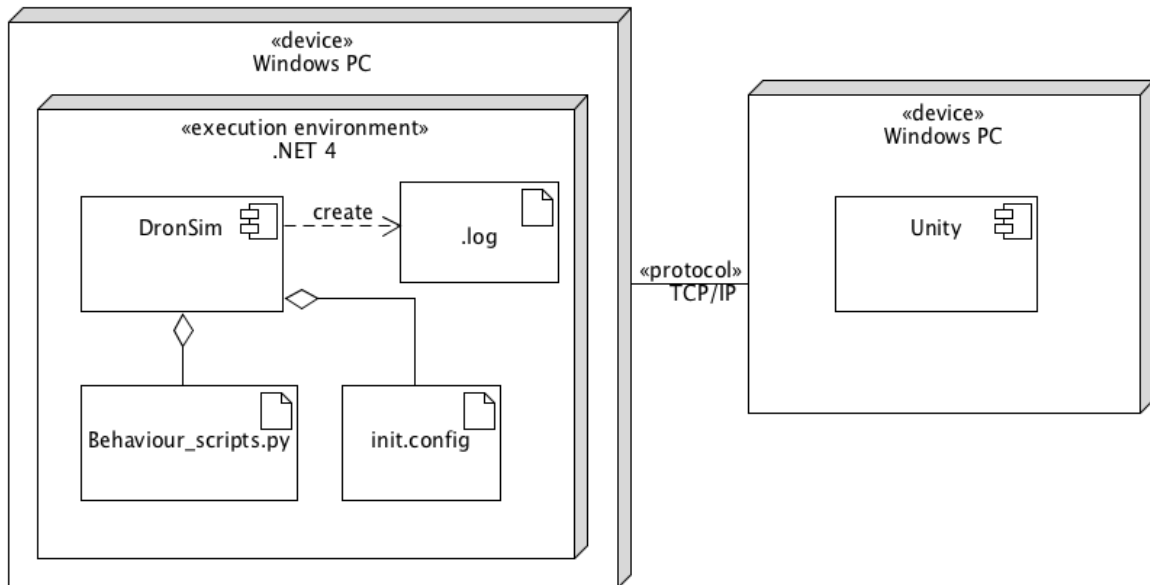
Nami navrhnutá architektúra aplikácie je použiteľná v troch rôznych rozloženiach nasadenia komponentov.

Prvý diagram nasadenia na obrázku 6 popisuje situáciu, kedy sú oba komponenty riešenia nasadené na jednom stroji s operačným systémom Windows. Pre spustenie samotného simulátora je potrebná aj inštalácia programového rámca .NET Framework 4.5.2. Časti riešenia potom komunikujú pomocou protokolu TCP/IP.



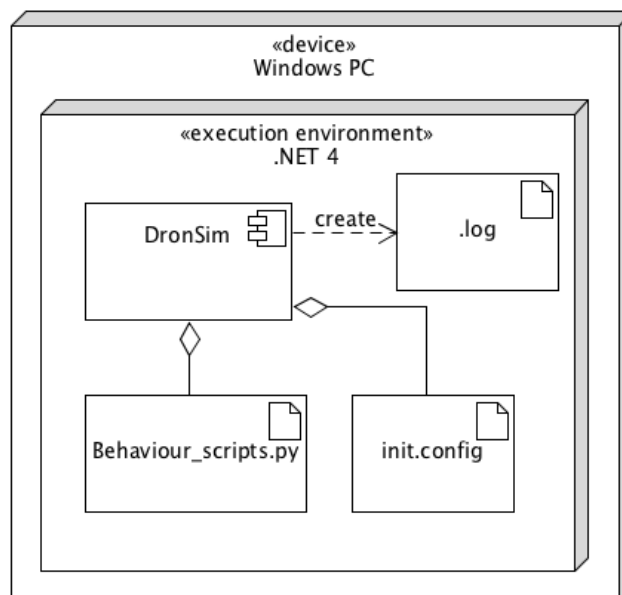
Obr. 6: Diagram nasadenia prvého prípadu

Diagram na obrázku 7. popisuje nasadenie aplikácie na dvoch samostatných zariadeniach s operačným systémom Windows. Tak ako v prvom prípade, na zariadení so simulátorom je pre spustenie potrebný programový rámec .NET Framework 4.5.2.



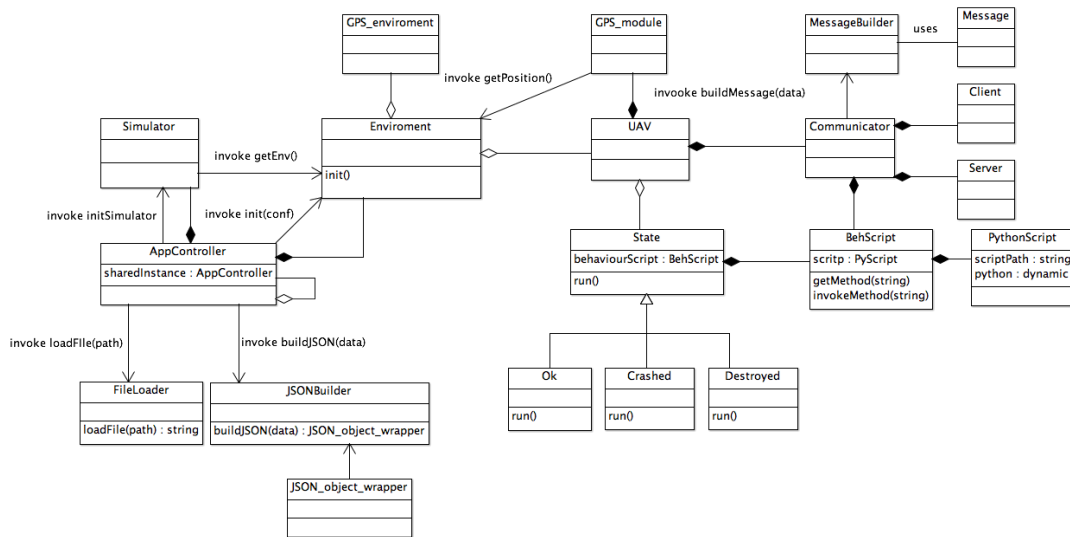
Obr. 7: Diagram nasadenia druhého prípadu

Obrázok 8 opisuje nasadenie aplikácie na jednom zariadení s operačným systémom Windows a programovým rámcom .NET Framework 4 bez nasadeného vizualizátora Unity 3D. V tomto nasadení sa zapisujú len záznamy o testovaní vo forme súborov *názov.log*.



Obr. 8: Diagram nasadenia tretieho prípadu

## Diagram tried



Obr. 9: Návrh diagramu tried novej implementácie

Diagram tried opisuje štruktúru tried novej implementácie aplikácie.

### AppController

Trieda určená na ovládanie toku aplikácie. Načítava konfiguračné súbory a na základe zadaných informácií inicializuje moduly Simulator a Environment. Tiež spúšťa a ovláda simuláciu.

### Simulator

Stará sa o samotnú simuláciu dronov. Aplikuje pohyb a vplyvy prostredia, s ktorými bolo prostredie Environment zinicilizované.

### Environment

Predstavuje prostredie, v ktorom budú drony simulované. Obsahuje informácie o dronoch, prekážkach na mape a mape samotnej.

### UAV

Objekt predstavuje dron podľa používateľskej konfigurácie z konfiguračného súboru, ktorý obsahuje informácie o rozmeroch, tvare a ostatných vlastnostiach dronov.

### **State**

Reprezentuje stav drona, ktorý bude obsahovať správanie drona v jednotlivých stavoch, definovaných v Python skriptoch.

### **Communicator**

Implementuje metódy, ktoré slúžia na komunikáciu dronov s prostredím Environment a prostredia s vizualizátorom Unity 3D.

### **JSON\_object\_wrapper**

Obaľuje JSON objekt a implementuje nad ním metódy na prístup k jeho reprezentácia.

### **GPS\_enviroment**

Modul pre počítanie GPS pozícií na základe pozícií dronov v prostredí a poskytovania pozícií metódami.

### **BehScript**

Reprezentuje triedu obsahujúcu Python skripty s opisom správania drona, komunikácie dronov a definícií senzorov dronov. Tiež implementuje metódy na volanie dynamických metód z Python skriptu.

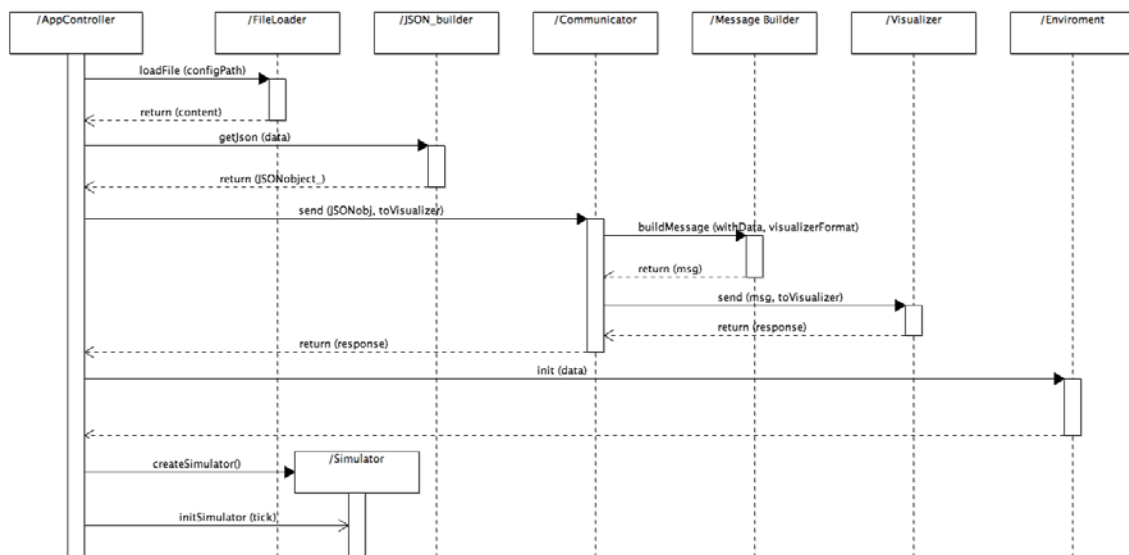


## Sekvenčné diagramy

V tejto kapitole popíšeme navrhnuté sekvenčné diagramy kľúčových funkcií navrhutej aplikácie.

### Inicializácia aplikácie

Obrázok 10 reprezentuje návrh volania metód pri inicializácii prostredia simulátora v novej implementácii projektu. Zahrňuje inicializáciu vizualizátora Unity 3D na základe konfiguračného súboru. Tiež sa tu inicializuje prostredie Environment dronmi a ostatnými parametrami z konfiguračného súboru. Ďalším krokom inicializácie je vytvorenie modulu simulátora Simulator a jeho nastavenie pre potreby scenára v konfiguračnom súbore.



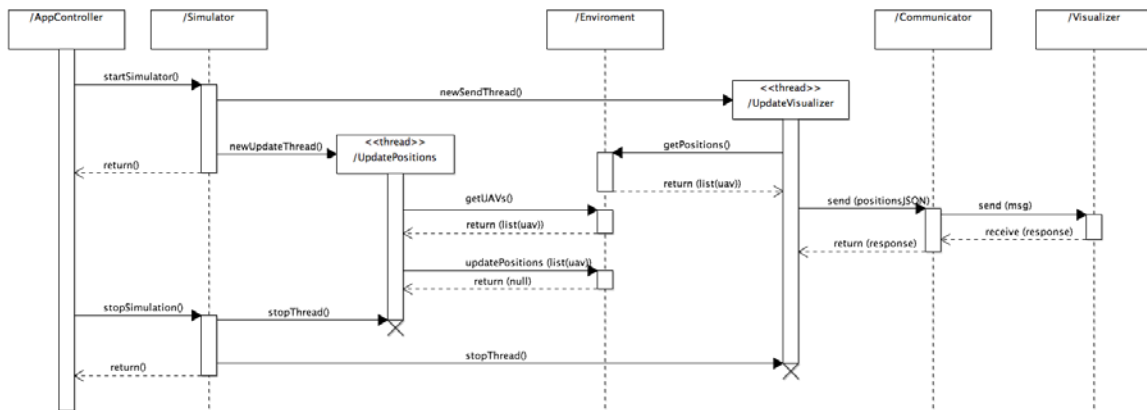
Obr. 10: Sekvenčný diagram inicializácie aplikácie

### Obnovovanie polohy

Na obrázku 11 je sekvenčný diagram návrhu volaní metód v procese simulácie. Obsahuje dve základné časti so samostatnými vláknami, ktoré sa spustia pri spustení simulácie.

Jedným je vlákno na komunikáciu s vizualizátorom Unity 3D, ktoré v určených intervaloch zasiela informácie o polohách dronov a ich zmene.

Ďalším je vlákno pre výpočet pohybu dronov v prostredí na základe scenárov a správania definovaného v skriptoch správania dronov. Zmenené pozície sú uložené v prostredí Environment.



Obr. 11: Sekvenčný diagram obnovovania polohy

## Návrh komunikácie medzi prostredím a vizualizátorom

Podľa nových požiadaviek na komunikáciu systému, v ktorých je definované, že má byť komunikácia rozšírená na komunikáciu po lokálnej sieti, je nutné prerobiť koncept pôvodne navrhnutej komunikácie. Momentálne je komunikácia zabezpečená pomocou UDP protokolu na rovnakom stroji (localhost). Keďže má byť komunikácia rozšírená na lokálnu sieť, je potrebné zmeniť protokol komunikácie na TCP.

Pôvodný návrh správ, ktoré si medzi sebou vizualizátor a prostredie vymieňajú, je v dobrom stave a môže byť ponechaný v rovnakej podobe. Formát správ zostáva aj naďalej JSON.

Vo finálnom návrhu bola TCP komunikácia medzi prostredím a vizualizátorom navrhnutá nasledovne: Po zapnutí TCP servera na strane prostredia server čaká na pripojenie klienta. Ihneď po pripojení klienta je serverom odoslaný `init_json`, ktorý obsahuje dimenzie mapy, objekty v mape a prvotné pozície dronov. Po prijatí a spracovaní `init_jsonu` na strane klienta nastáva periodický update pozícií posielaním json správ z prostredia do vizualizátora. Správy sa posielajú s oneskorením 1ms, aby toľko nezaťažovali procesor a taktiež sa neposielajú duplicitne, ak nenastal žiaden pohyb dronov.

Posielanie správ z prostredia do vizualizátora je riešené v dvoch krokoch. Najskôr klient očakáva presnú veľkosť json správy zakódovanú do 4B, čo znamená, že najväčší možný json, čo môže klient prijať je 9999B – predstavuje to približne 60 dronov v prostredí. Toto číslo nie je žiaden problém navýšiť, stačí zmeniť na koľko bajtov sa bude kódovať veľkosť správy a na strane klienta podobne navýšiť, koľko bajtov bude očakávať. Po prijatí veľkosti klient ihneď čaká na samotný json. Keďže TCP sockety posielajú správy o nepravidelnej veľkosti, prijímanie json správ je riešené v cykle. V tomto cykle sa prijímajú správy, až kým nám nepríde presne taký počet, čo sme dostali v prvej správe. Potom už len prebehne spracovanie jsonu a následne vyzobrazenie vo vizualizátore.

## **Návrh vizualizátora**

Vizualizátor vypracovaný bývalým tímom je postačujúci a momentálne nie je potrebné ho meniť. Doposiaľ neboli zadané žiadne ďalšie požiadavky pre vizualizátor.

# 11 Implementácia

Aktuálna verzia našej implementácie ponúka dve možnosti simulácie. Prvou možnosťou je simulácia bez grafického rozhrania, pri ktorej je možné jej priebeh sledovať prostredníctvom vytváraných záznamov v textovej forme (tzv. logov).

Záznamy sú ukladané do podadresárov priečinka „Logs“ nachádzajúceho sa v koreňovom adresári projektu. Podadresáre majú názov vo formáte „session\_<dd-MM-rrrr\_HH-mm-ss>“, kde dd-MM-rrrr\_HH-mm-ss je čas inicializácie záznamníka. Obsahom tohto podadresára sú jednotlivé záznamy každého simulovaného UAV. Názov súboru je vo formáte „uav\_<id>.log“, kde id je identifikačné číslo daného UAV.

Príklad obsahu záznamu vidno na obrázku nižšie. Obsahom záznamu sú zmeny pozície UAV a tiež detaily o odoslaných a prijatých správach.

```
16:55:55 28.04.2017
| current position
|   x      102,3
|   y      20,136
|   z      10

16:55:55 28.04.2017
| current position
|   x      102,4
|   y      20,14
|   z      10

16:55:55 28.04.2017
| message sent
|   To      -1
|   Type    WiFi
|   Protocol Routing

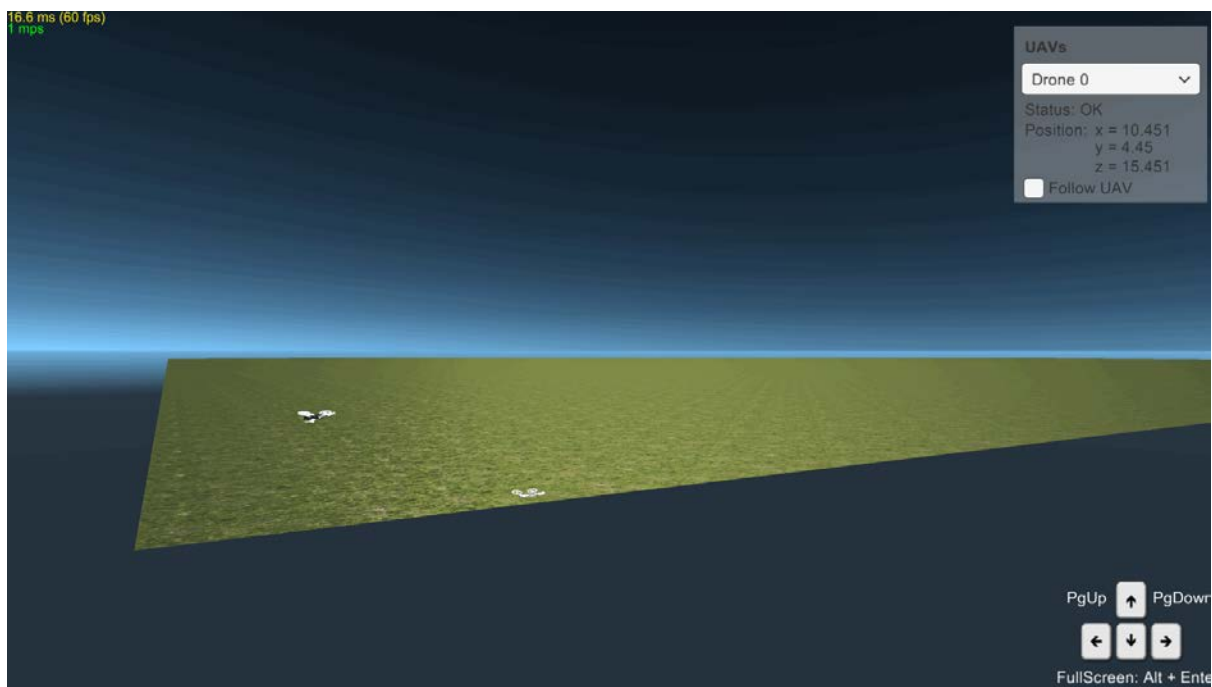
16:55:55 28.04.2017
| message received
|   From    2
|   Type    WiFi
|   Protocol Routing

16:55:55 28.04.2017
| current position
|   x      102,5
|   y      20,144
|   z      10
```

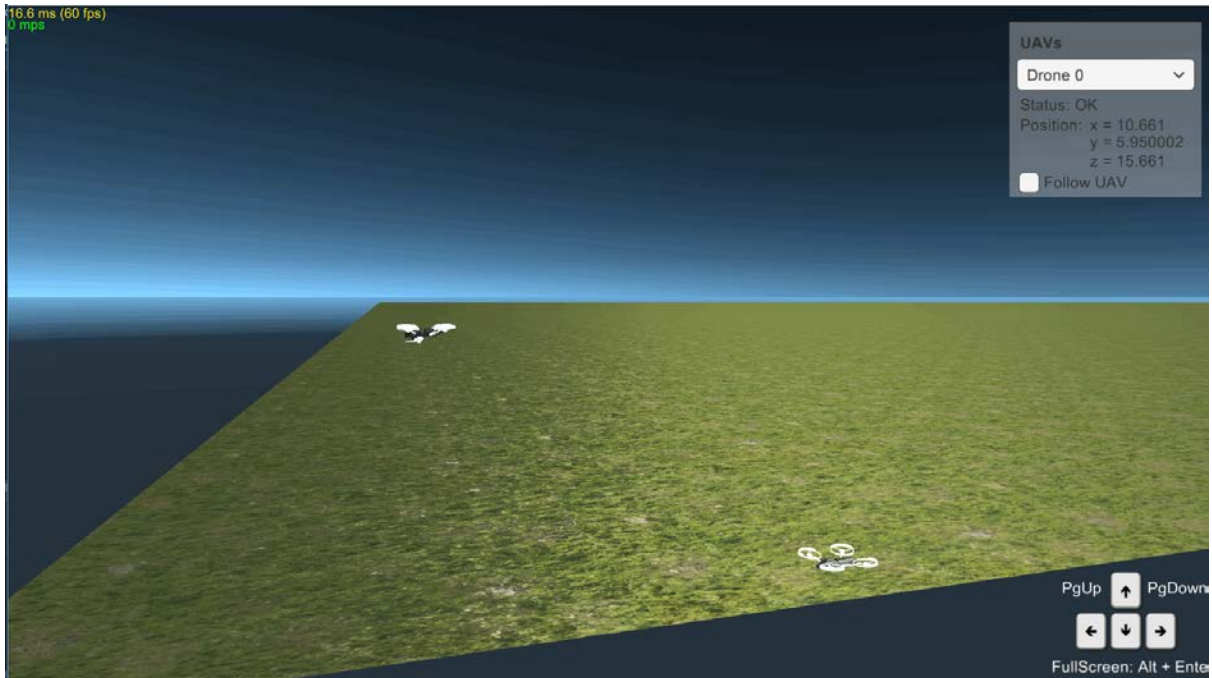
Obr. 12: Záznam vygenerovaný zo spustenej simulácie

Druhou možnosťou je zobrazenie s konfigurovateľným grafickým rozhraním. Používateľ v rámci neho môže nastaviť rozlíšenie, úroveň detailov, či zobrazenie v režime na celú obrazovku, resp. v okne. Vyššie spomenuté súbory so záznamami sú vytvárané aj počas spusteného grafického zobrazenia.

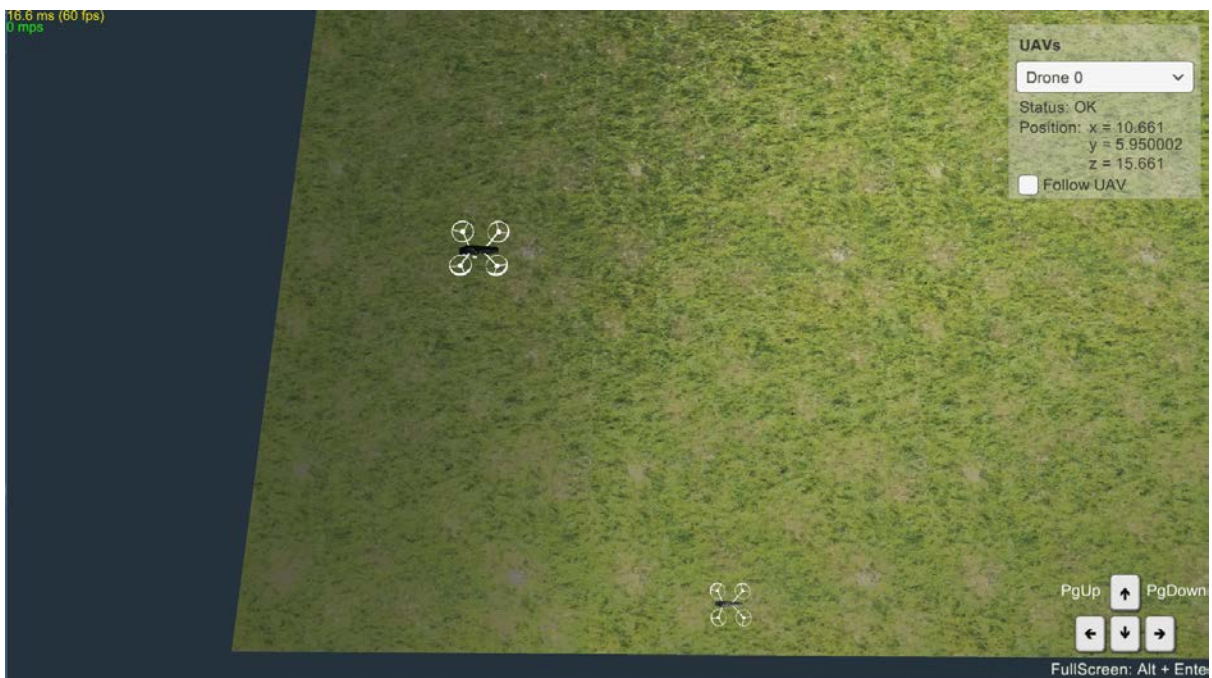
Samotné grafické rozhranie a vykresľovanie simulácie je implementované v nástroji Unity 3D. Keďže samotná logika simulácie prebieha v C# konzole, bolo potrebné prepojiť tieto dva komponenty. Komponenty sú prepojené TCP soketmi a vymieňajú si medzi sebou údaje. Pre tento účel sme sa rozhodli využiť JSON súbory, ktoré budú v sebe obsahovať potrebné informácie. Na začiatku existuje jeden hlavný konfiguračný JSON súbor, na základe ktorého sa nastavia počítačové údaje. Ide konkrétne o veľkosť mapy, počet dronov a ich vlastnosti v podobe počítačovej pozície, rozmerov a senzorov. Taktiež tento súbor obsahuje počet, veľkosť a pozície jednotlivých objektov nachádzajúcich sa na mape. Počas behu simulácie prostredie pravidelne oznamuje vizualizátoru aktuálne stavy a pozície dronov pomocou JSON súborov.



Obr. 13: Grafické rozhranie (vzdialený pohľad)



Obr. 14: Grafické rozhranie (približený pohľad)



Obr. 15: Grafické rozhranie (horný pohľad)

Prostredníctvom komunikačného rozhrania aplikácia poskytuje model kooperácie UAV zariadení. Napríklad jedno UAV posiela správu druhému UAV, čím mu oznamuje, aby zmenil smer pohybu. Požiadavka na zmenu vektora pohybu je obsahom odoslanej správy.

## 12 Testovanie

V rámci testovania uplatňujeme testovanie pomocou unit testov. Tieto testy sme sa rozhodli vykonávať pomocou NUNIT frameworku, ktorého adaptér sme implementovali do Visual Studia ako rozšírenie. Princíp písania testov je zadaný v metodike testovania.

# Prílohy

## C Používateľská príručka

### Inštalčná príručka

Inštalácia aplikácie je možná dvomi spôsobmi:

#### 1. Bez buildovania

Na nainštalovanie aplikácie bez buildovania je potrebné mať vybuildovanú verziu projektu na relatívnej adrese `<project_folder>/bin/Debug`. Taktiež je potrebné mať na relatívnej adrese `<project_folder>/Config/` súbor `init.json` a python skripty, ktorých cesty sú zadefinované v tomto `init.json` konfiguračnom súbore (štandardne v priečinku `<project_folder>/PythonScripts/`). Ich predvolené názvy by mali byť `script_1.py`, `script_2.py`, `script_3.py` pre 3 UAV defaultne nakonfigurované v konfiguračnom súbore. V prípade potreby použitia vizualizátora (spustenia simulátora s vizualizátorom) je potrebné mať vizualizátor s názvom `unity.exe` v priečinku `<project_folder>/UnityExecutable/` taktiež so súborom `ip.txt` a priečinkom `unity_Data`. Posledným krokom je pridanie nainštalovanej verzie knižníc Pythonu do syspath Pythonu, čo sa robí priamo v každom Python skripte pre UAV nasledujúcim spôsobom:

```
import sys
sys.path.append("<cesta_k_python_lib>")
teda napríklad: sys.path.append("C:\\Python27\\Lib")
```

Po týchto krokoch je aplikácia úspešne nainštalovaná a je ju možné spustiť.

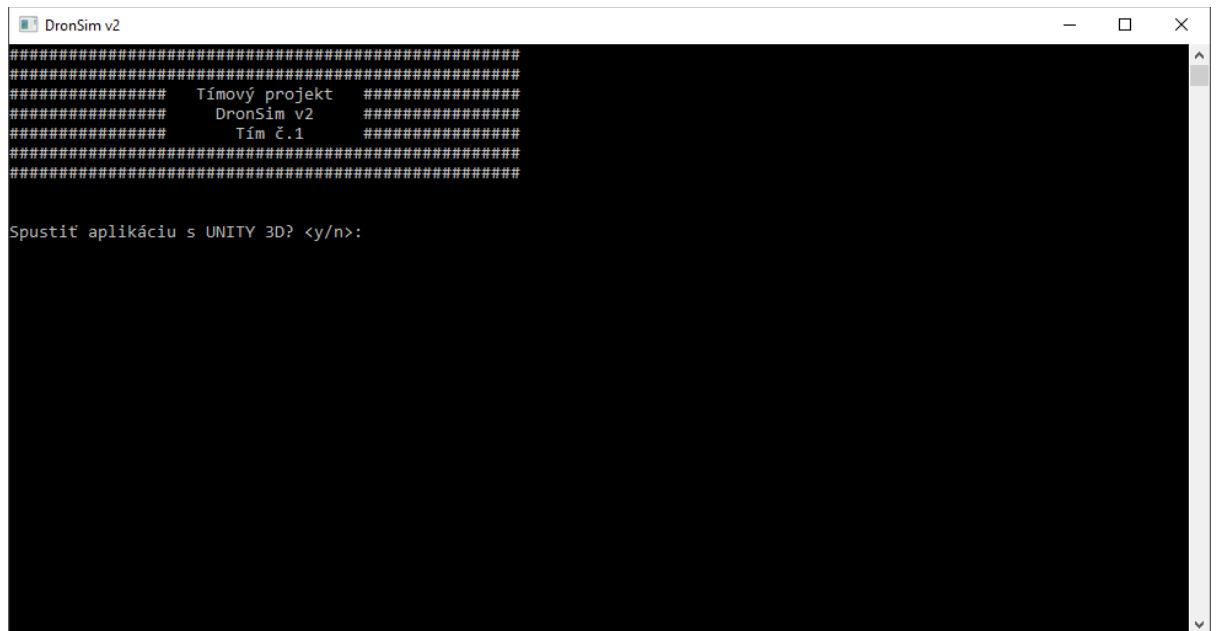
#### 2. S buildovaním

Posledné verzie projektu je možné vybuildovať iba pod operačným systémom Microsoft Windows (odporúča sa verzia 7 a vyššia). Na vybuildovanie odporúčame nástroj Visual Studio (postačujúca je aj základná bezplatná verzia Community). Testované boli verzie Visual Studio 2015 a 2017. Na vybuildovanie je potrebné mať k dispozícii celý zdrojový kód projektu a vo Visual Studiu otvoriť súbor s názvom `LegendroneSim.sln` na otvorenie projektu. Pre samotné vybuildovanie je potrebné zvoliť konfiguráciu „Debug“ alebo „Release“ a bitovú sadu „x86“. Konfigurácia bitovej sady „x64“ nie je v súčasnosti podporovaná. Po vybuildovaní programu sa pokračuje bodom 1.



## Použitie vizualizátora

1. Pri spustení aplikácia dáva na výber možnosť spustenia simulácie s vizualizátorom (y) alebo bez vizualizátora (n).



```
DronSim v2
#####
#####          Tímový projekt          #####
#####          DronSim v2              #####
#####          Tím č.1                  #####
#####
Spustiť aplikáciu s UNITY 3D? <y/n>:
```

Používateľ zadá svoju voľbu a potvrdí ju stlačením klávesy Enter.

V prípade, ak si používateľ zvolil možnosť spustenia simulácie bez vizualizátora (n), simulátor bude fungovať v režime konzolovej aplikácie a všetky udalosti bude zároveň zaznamenávať do logovacích súborov.

2. Naopak, ak si používateľ zvolil možnosť spustenia simulácie s vizualizátorom, aplikácia dáva na výber možnosť použitia lokálneho (y) alebo vzdialeného (n) vizualizátora.

Lokálny vizualizátor predstavuje vizualizátor na rovnakom stroji, na ktorom beží aplikácia simulátora, zatiaľ čo vzdialený vizualizátor sa nachádza na inom stroji a komunikuje sa s ním prostredníctvom LAN siete.

Používateľ opäť zadá svoju voľbu a potvrdí ju stlačením klávesy Enter.

Pri zvolení možnosti lokálneho vizualizátora sa automaticky spustí vizualizátor a po zvolení prvotných nastavení funguje simulátor v režime grafickej aplikácie.

3. V prípade, že používateľ nezvolí možnosť lokálneho vizualizátora a rozhodne sa použiť vzdialený vizualizátor, aplikácia vyzve na zadanie IP adresy servera.

Tu je potrebné zadať IP adresu existujúceho rozhrania na stroji, na ktorom sa má previesť komunikácia po LAN sieti s vizualizátorom. V prípade zadania nesprávnej IP adresy nie je možné spustiť server a aplikácia nepobeží.

- a) Po úspešnom spustení servera je potrebné manuálne spustiť vizualizátor. Pri spustiteľnom .exe súbore vizualizátora sa nachádza .txt súbor s IP adresou, do ktorého je potrebné zapísať IP adresu servera. Je to rovnaká IP adresa, aká bola zadaná v kroku 3 do konzoly simulátora.
- b) Následne už stačí len spustiť vizualizátor a po zvolení prvotných nastavení funguje simulátor v režime grafickej aplikácie.

*Poznámka:* Dôležité je, aby server vždy bežal ako prvý.

## **D Technická dokumentácia**