Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Szakdolgozat

Kamera alapú beltéri navigáció az AR drone eszközön

Petrovicz Benedek
Mérnök informatikus BSc
2017

Témavezető:
dr. Zsedrovits Tamás

Alulírott Petrovicz Benedek a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Karának hallgatója kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem és a szakdolgozatban csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen a forrás megadásával megjelöltem. Ezt a Szakdolgozatot más szakon még nem nyújtottam be.

_____

Petrovicz Benedek

Tartalomjegyzék

Kivonat

A dolgozat arra törekszik, hogy az automatizált drónvezérlés témában érdekelt személyeknek, eligazítást és kiindulási alapot adjon. Amellett, hogy elméleti síkon, általános áttekintést ad a témáról, mellékletként tartalmazza ezen gondolatok tényleges megvalósítását is.

A kidolgozás kifejezetten olyan drón kutatókat céloz, akik pontos méréseket szeretnének végezni és ezáltal hatékony algoritmusokat fejleszteni. A kutatás során felhasznált összes eszköz, fontos részét képezi a rendszernek, mivel bármelyik elhagyása esetén használhatatlanná válik a kidolgozott kód. Bár az eszközök szimulálására van lehetőség, de az nagy mértékben ronthatja a mérések eredményét.

A kutatás során felhasznált drón a Parrot cég AR.Drone 2.0 nevű terméke. Viszonylag kis térben is biztonságos körülmények között használható. A biztonságot maga a drónra szerelt kemény szivacs alapú keret adja, mely amellett, hogy védi a drón propellereit, azt is megakadályozza, hogy a környezetében nagyobb kárt tudjon tenni. További biztonságot nyújt a drón fedélzeti számítógépén futtatott szkript, ami azonnal leállítja a drón rotorjait, amint azt érzékeli, hogy az eszköz elérte a 90 fokos dőlésszöget, ami által könnyedén megállítható az akkumulátorhoz való hozzáférés nélkül is.

A felhasznált eszközök közül a második legfontosabb, az OptiTrack cég mozgáskövetésre alkalmas kamerarendszere. Ez képes akár centiméteres pontossággal lekövetni és háromdimenziós térben ábrázolni azon tárgyakat, amik fel vannak szerelve a kamerákhoz kidolgozott fényvisszaverő markerekkel.

Az eddig felsorolt eszközök kommunikációját pedig, egy Linux rendszerrel ellátott gépen futó Robot Operating System (ROS) teszi lehetővé. Ezt a nyílt forráskódú rendszert széles körben alkalmazzák a robotikával foglalkozó cégek és kutatók, így rengeteg eszközhöz érhetők el különböző felhasználásra szánt implementációk.

A repülés közbeni utasítás közvetítést, QR kódok és a drón 2 kamerája segíti elő. A jelek felismerését a kamera képéből az ar_track_alvar nevű ROS csomag valósítja meg. Ez a csomag kifejezetten jól működik gyenge felbontású kamera remegő képével is, amivel a projekt során többször is találkozhattunk. Az egyetlen hátrány

abból adódik, hogy a drón egyszerre csak az egyik kamerája képét tudja közvetíteni a rendszernek.

A megvalósított forráskód egy ROS csomag formájában használható fel. Fel van készítve a rendszer minden összetevőjének vezérlésére és az adataik feldolgozására. A felhasználónak csupán az eszközök IP címeit kell pontosan megadnia a csomag számára, illetve, hogy drón melyik kameráját szeretné használni a QR kódok felismerésére.

A QR kódon szereplő adat alapján több különböző előre megírt algoritmust is képes végrehajtani. A projektben csak előre kiszámolt animációkat kapott meg parancsként a drón, amelyek segítségével képes átrepülni vagy egy irányban kikerülni az előtte lévő akadályt.

További fejlesztések során már csak ezzel a részével kell foglalkoznia az adott kutatónak. További szenzorok adatainak a felhasználása is egyszerűen megoldható. Mivel a kód python nyelven készült, gyorsan és problémamentesen lehet tovább alakítani.

Abstract

The thesis pursues at giving a guidance and a starting-point to people who are intrested in automated drone control. Besides that it gives a general theoretical overview of the topic, it contains the implementation of these thoughts as an attachment.

This work mostly aims at those drone researchers, who want to take accurate measurements and to develop effective algorithms with them. All of the items used in this research are important parts of the system, because leaving out any of them makes the code unusable. Although simulating them is another possibility, but it spoils the measurement results greatly.

The drone used in this research is the product of the Parrot company called AR.Drone 2.0. It can be used in relatively small places, under safe conditions also. The main reason of this safety is the hull of the drone made out of some hard foam. Besides that it protects the propellers of the drone, ensures that the drone is not causing any great damage in the environment around it. A script running on the on-board computer of the drone provides additional security, because it blocks the rotors of the drone as soon as it detects that the drone is tilted by 90 degrees so it can be stopped easily without reaching its battery.

The second most important item used in the research is the camera system of OptiTrack that is capable of motion capture. It is able to track with centimeter accuracy and represent in a three dimensional space essentially anything that is equipped with the reflective markers designed for these cameras.

These two items are communicating through Robot Operating System (ROS) on a computer with Linux. This open-source framework is used widely by companies and researchers intrested in robotics, so there is a wide variety of implementations for a lot of tools and devices.

Giving instructions mid-air, is solved by detecting and decoding QR codes from the video feed of the two camera on the drone. The detection from the camera feed is done by the ar_track_alvar ROS package. This package works really good with shaky image of low resolution cameras, that is frequent in projects like this. The only drawback is, that the drone can broadcast the video feed of only one camera at once.

The implemented source code can be used as a ROS package. It is suitable for controlling all of the components in the system and processing their data. The user is only needed to give the correct IP address of the items, and to choose which camera is used for detecting QR codes.

It is possible to call multiple different algorithms according to the data in the QR code. In the project there were only pre defined animations for bypassing over or next to an obstacle that is before the drone.

In further development processes this part is only intended to be worked on by the researcher. Data of additional sensors can be used easily. As the code has been written in python it can be formed rapidly fast and without any problem.

1. Bevezetés

A kutatás fő célja a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Karán elindított UAV laboratórium alapjainak kidolgozása és lefektetése. Számtalan lehetőség rejlik ebben a gyerekcipőben járó technológiában, így fontosnak tartom, hogy minél előbb megismerjék a hallgatók és részesei lehessenek a fejlesztésének.

A tervezés fő szempontja egy olyan rendszer megvalósítása, ahol az AR drón az általunk írt szkriptek segítségével, automatizált módon képes bizonyos feladatokat végrehajtani. Az OptiTrack kamerarendszer közreműködésével, egy olyan teszt környezetet hozhatunk létre, ami amellett, hogy megbízható, rengeteg plusz információval szolgál a hatékonyság növelésének érdekében. Könnyedén szimulálhatunk vele különböző háromdimenziós terekben való mozgást.

Fontos lépés lehet ez az egyetem életében, mivel a jó alapok megteremtése ebben a témában, nagyon jó lehetőségeket nyithat a jövő hallgatói felé. Aki nem zárkózik el teljes mértékben a közösségi médiától, az nagy eséllyel hallott vagy olvasott már drónokról és azok alkalmazásáról szóló cikket. Úgy gondolom ma pontosan abban a világban élünk, ahol ez az egyik leginkább felkapott téma fiatalok és idősebbek között is, mivel olyan piaci területeket nyitott meg a technológia, ami eddig még ismeretlen volt sokak számára.

1.1. Drónok

Sokan nincsenek tisztában a drón mint fogalom pontos jelentésével. A legtöbben a katonai alkalmazásra gondolnak, ahol csapásmérő vagy kémkedő eszközként hasznosítják. Ma már talán többen is vannak, akiknek a szó hallatán a játékra vagy videófelvételre alkalmas 4 rotoros, viszonylag kis méretű eszközök jutnak az eszébe.

A drón eredeti neve az angolban UAV-ként megismert Unmanned Aerial Vehicle vagy magyarul pilóta nélküli légi jármű. Ez kevésbé hangzatos ám annál beszédesebb név. Bármilyen eszközt nevezhetünk drónnak ami alkalmas repülésre úgy, hogy nincs pilóta a fedélzetén. Ebből következik, hogy egy drón bármilyen kicsi vagy nagy lehet, a határokat csak a fizika törvényei szabják meg. A parancsokat általában távvezérléssel kapja valamilyen rádió frekvencián vagy a fedélzetén található szenzorok adatai alapján generálja.

1.2. Alkalmazási területek

Jogos kérdés lehet, a drónok fontossága és gyakorlati haszna vagy egyáltalán ezen projekt értelme. Manapság ezekből egyre többet látni és hallani mind a nagy cégek alkalmazásában mind pedig a halandó kisembereknél is.

Természetesen a legkorábbi fejlesztések olyan problémákat igyekeztek megoldani, mint például a városon belüli csomagszállítás, [1] vagy az emberéletek megmentése közben fellépő akadályok legyőzése. [2] Azt hiszem kijelenthetjük, hogy a fő alkalmazási módokat a nagy multinacionális cégek, a hadsereg, illetve az egészségügy határozza meg, mint ahogy az sok más eszköz esetében is előfordul. Sajnos az is elmondható, főleg a nagy cégek esetében, hogy nem fordítanak elég figyelmet az alkalmazásuk közben fellépő veszélyforrásokra. Sajnos a drónok a legtöbb esetben nem rendelkeznek a megfelelő biztonsági rendszerrel.

Ez felveti olyan algoritmusok és eszközök kidolgozását, amik minél egyszerűbben alkalmazhatók és megelőzik ezeket. Itt jönnek szóba az olyan kutató csoportok, aminek mi is szándékozunk az alapjait lefektetni. Rengeteg alap algoritmus fejlesztésére szükség van, olyanokra, mint például, a virtuális háló, ami nem engedi, hogy a drón elhagyjon egy megadott területet. Természetesen ezekre vannak már megoldások, viszont ezek sok esetben nem felelnek meg azoknak az elvárásoknak, amik szerint nem okozhatnak kárt a környezetükben.

Vannak persze olyan alkalmazási területei is a drónoknak, ahol csekély a környezetre mért károk esélye és mégis hatalmas hasznot tud hozni. Ilyen például a drónok mezőgazdaságban való felhasználása. [3] Képes a gazda által birtokolt terület megfigyelésére és különböző adatok elemzésére. Alkalmas például felmérni, hogy melyik területek azok, amik több gondozást igényelnek. Ezen kívül alkalmazható még veteményezésre is, ami igen hasznos lehet nagy területek esetén. Nagy előnye itt a drónoknak, hogy sokkal gyorsabban képesek elvégezni a feladatukat az eddig használt megoldásokhoz képest.

Hasznosítják ezeken kívül még bányáknál a kibányászott nyersanyag mértékének vagy minőségének megállapítására. [4] Továbbá nagy ipartelepek esetén biztonsági megfigyeléshez is felhasználhatók. [5]

1.3. Robot Operating System

A Robot Operating System vagy rövidebben csak ROS egy nyílt forráskódú, általánosított „operációs rendszer” robotokhoz. Megvalósítja a hardver szintű absztrakciót, képes kezelni a hardverek vezérlését és az egyes folyamatok közti kommunikációt.

A ROS rengeteg olyan eszközt és könyvtárat biztosít, ami lehetővé teszi a különböző platformokon való fejlesztést. Támogatja a kód újra felhasználást, ami nagymértékben megkönnyíti a fejlesztők munkáját. Rendelkezik beépített vizuális megjelenítővel is, ami kiváló segítséget nyújt a teszt környezetünk megjelenítéséhez a kódot futtató gépen. A kutatás során készített forráskód is egy ROS csomagként érhető el.

1.4. Motion Capture

A Motion Capture vagy magyarul mozgásrögzítés, lényegében bármilyen objektum mozgásának a felvételét takarja, ami háromdimenziós térben egy számítógép és sok szenzor segítségével történik. Esetünkben az objektumot fényvisszaverő markerekkel szereljük fel, amiknek a térbeli adatait az OptiTrack kamerarendszer állapítja meg, majd integrálja ezeket az adatokat a saját rendszerébe, amiket utána képes közvetíteni a ROS számára is. A technológia segítségével pontos méréseket és szimulációkat végezhetünk, amik a valós térből származó adatokkal dolgoznak.

2. Előzmények

A témában fellelhető szakirodalmi alkotások többsége sajnálatos módon csak fizetés ellenében érhető el. Lévén, hogy egy olyan technológiáról van szó, ami csupán pár éve törte be magát a köztudatba, nehéz olyan ingyenes forrásokat találni, amik megbízható, kutatói háttérrel rendelkeznek. A kutatásom során igyekeztem ezekre hagyatkozni.

Az ehhez hasonló fejlesztések esetében még a nyílt forráskódú projektekből dokumentációjából is lehet hasznos információkhoz jutni, ám ebben az esetben nagyon körültekintőnek kell lenni, mivel ezek a projektek sokszor rejthetnek hibákat. Ezek a hibák egy nagyobb méretű drón esetében akár komoly következményekkel is járhatnak. A 2017-ben rendezett Golden State Race Series biciklis versenyen is történt egy baleset, ahol az egyik résztvevő biciklijét találta el egy drón miután becsapódott egy közeli fába. [6] Bár ebben az esetben emberi hibából történt baleset, nem volt olyan biztonsági rendszer a drón szoftverében, ami képes lett volna megakadályozni azt.

2.1. Korábbi kutatások

A kutatást megelőzően már egy félévet foglalkoztam a rendszer felépítésével az önálló laboratórium keretein belül. Itt felkutattam a számomra elérhető drónok jellemzőit és több tesztet is végeztem a kiválasztott AR.Drone 2.0-val. Megismerkedtem a ROS rendszerrel, ami kifinomult megoldásokat tartalmaz a robotikával kapcsolatos kutatásokhoz. Beüzemeltük az egyetem birtokában lévő OptiTrack kamerarendszert és több tesztet is végeztünk vele. Megtanultam rendszer szoftverének a Motive-nak a használatát, a kalibrálás folyamatát, a pontos paraméter beállításokat és a kamerák megfelelő elhelyezését a saját tesztkörnyezetünkben. Létrehoztam több különálló objektumot, amiket viszonylag hatékonyan el tud különíteni a szoftver, ezzel elősegítve több projekt egyidejű fejlesztését és tesztelését.

A projekt eredményeként létrehoztam egy olyan teszt környezetet, ahol a ROS segítségével a rendszer képes volt egy helyen kezelni a drón és az OptiTrack adatait. Ehhez viszont elengedhetetlen a ROS beható ismerete, mivel innentől már minden fejlesztés ott történik. Meg kellett tanulnom ROS csomagot készíteni, fejleszteni és használni. Megismerkedtem a csomagok közti kommunikációs csatornákkal és az egyes programok paramétereit tároló szerver működésével.

2.2. Szakirodalom tanulmányozása

Fontos előzménye volt a kutatásoknak a drón dokumentációjának [7] és a kapcsolódó szakirodalom tanulmányozása és minél részletesebb megismerése. Ezek elengedhetetlen részei a drónnal végzett hatékony munkának. Ezáltal teljes képet kaphatunk azokról az eszközökről, amiket később hasznosíthatunk a gördülékenyebb munkafolyamat érdekében.

2.3. Repülő megismerése, irányításának megtanulása

A dokumentáció után fontos volt, hogy tisztában legyek a drón specifikációjával és a hozzá kapcsolódó eszközök használatával. Ilyen eszköz például az ardrone_autonomy [8] nevű ROS csomag, ami lehetővé teszi a drón irányítását egy számítógépen keresztül. A csomag egy magas szintű nyelvezetet biztosít a kezelője számára. További segítséget nyújthat az ardrone_tutorials [9] csomag ismerete, ami egy további absztrakciós szintet biztosít a drón minél egyszerűbb irányításához.

2.4. QR kód felismerésére alkalmas megoldás keresése

Az összetettebb feladatok megoldásához szükség a volt a QR kód felismerésre. Az erre alkalmas algoritmust a drón kamerájával készült képeken kellett futtatni. Ehhez felkutattam a feladat szempontjából releváns ROS csomagokat. Három különböző csomag jött szóba, amiket egyenként megvizsgáltam, hogy melyik lenne számomra a leginkább alkalmas. A választásomat a drón jellemzői alapján kialakított feltételek befolyásolták. Működnie kellett a kamerák alacsony felbontása, és a repülés közben keletkező instabil kép esetén is.

Az első a zbar_ros nevű csomag volt. Aki valaha foglalkozott QR vagy vonalkód detektálással és dekódolással az valószínűleg találkozott már a ZBar nevű nyílt forráskódú programmal. Sok nyelven megtalálható az implementációja és a széleskörű támogatottság eredményeként képes nagyon gyorsan detektálni egy képi kódot. Nagy hátránya viszont, hogy állóképekre van optimalizálva, és néhány további tervezési tulajdonság miatt, mint például, hogy képtelen egyszerre több kódot detektálni önállóan, alkalmatlannak bizonyult a projekt számára.

A második a visp_auto_tracker. Ez a csomag már több lehetőséggel rendelkezik. Képes a képen megkeresni egy QR kódot, majd lekövetni azt és figyelnie a pozícióját és az orientációját. Ez már több kód egyidejű követésére is alkalmas ellentétben a ZBar-ral. Ez a csomag a Visual Servoing Platform része, ami tartalmaz még sok más kapcsolódó programot is. Ilyen például a visp_camera_calibration ami egyedi minták

alapján segíti a kamera kalibrációját a minél hatékonyabb felismerés érdekében, illetve a visp_tracker ami már nem a QR kódok hanem háromdimenziós objektumok felismerésére és követésére képes, ami igazán hasznos lehet további kutatások esetén. Bár ez a csomag már képes a legtöbb feladat elvégzésére, amit a projekt sikeressége igényel, de még mindig nem elég megbízható a mozgás közbeni kamera képéről való detektálásra.

Végül rátaláltam a visp_auto_tracker egyik versenytársára az ar_track_alvar nevű kódfelismerő csomagra, ami a harmadik és egyben az utolsó megvizsgált program a listában. Egy fórum bejegyzés [10] szerint ilyen instabil kép esetén, ami az AR drónnál előfordul, sokkal hatékonyabb ez a csomag, mint az előző. Ez is hasonló képességekkel rendelkezik de, az online vélemények szerint a különböző szűrő algoritmusok itt hatékonyabbak a rossz minőségű videókép feldolgozása esetén.

2.5. Hasonló alkotások

Rövid keresés után rátaláltam egy kísértetiesen hasonló projektre, aminek a célja az volt, hogy a drón alsó kamerájának segítségével detektáljon egy QR kódot, majd landoljon. A szerzője nem valódi hardverekkel, hanem szimulált környezetben végezte a fejlesztéseket. Érdekes volt számomra, hogy a repülés végeztével a landolást egy neurális háló segítségével hajtotta végre a drón, a QR kód tetején. Sajnos a projektet már két éve nem fejlesztik, emiatt a jelenleg használt ROS verzióval nem kompatibilis. A kompatibilitási problémák megoldása aránytalanul sok munkát igényelt volna, így csak a forráskódokat vizsgáltam át.

Bár hiányzott a projektből az OptiTrack kamerarendszer, ennek ellenére is sikerült pár jó gondolattal fejlesztenem a saját kódomat. Ez volt számomra az első olyan projekt, ahol az általam is használt ardrone_tutorials csomag vezérlésért felelős részét felhasználták. Sikerült pár ötletet merítenem az ide kapcsolódó kódok szervezésével kapcsolatban. Eleinte nem tudtam összekapcsolni az ar_track_alvar és a drón rendszerét. Ennek a problémának a megoldását is ebben a projektben találtam meg, ami csupán a helyes paraméterezésen múlt.

A szimulált környezet miatt nagyon egyszerűnek tűnt a feladat megoldása, mert a fejlesztőjének nem kellett lekezelnie a drónra ható fizikai tényezőket. A valóságban sokkal több kivételkezelésre volt szükség.

3. A tervezés részletes leírása

Miután sikeresen teszteltem a drón, az OptiTrack [11] kamerarendszer a Robot Operating System és a QR kód detektálás működését, kezdődhetett a fejlesztői munka. Ez a fejezet részleteibe menően bemutatja a fejlesztés során fellépő problémákat és azok megoldásait.

3.1. A munka során felhasznált eszközök

A kutatás elvégzéséhez sok hardveres és szoftveres eszköz nyújtott segítséget. Ebben az alfejezetben ezeket szeretném ismertetni és a fontosabb jellemzőiket bemutatni. Fontos megjegyeznem, hogy a fejlesztés során kialakult rendszer nem indokolja, hogy pontosan ezeket az eszközöket használja fel valaki hasonló munkája során. Ez csak egy olyan irányvonalat képvisel, ami megbízható hátteret biztosít a fejlesztői munka támogatásához.

3.1.1. OptiTrack kamerák

Korábban már szerepelt a motion capture technológia jelentősége és gyakorlati haszna. A hardver, ami mindezt biztosítja számunkra, az OptiTrack cég által forgalmazott kamerarendszer. Sokféle kamerát tartalmaz a repertoárjuk, de nem mindegyik alkalmas a dolgozatban bemutatott munka támogatására. Ezek a speciális kamerák, az infravörös fényt megfelelő módon átengedő lencsékkel és szűrőkkel vannak ellátva, valamint a lencséjük körül egy gyűrűvel, ami infravörös fény kibocsátására alkalmas diódákat (LED) tartalmaz. Továbbá fel vannak még szerelve egy állapotjelző LED-eket tartalmazó másik gyűrűvel is a lencséjük körül, ami segíti a kalibrálás menetét.

Az egyetem 8 darab Prime 13 típusú kamerával rendelkezik, amik kiválóan használhatók olyan méretű terek lefedésére, mint például az egyetem edzőterme, ami végül a tesztek helyszínéül is szolgált. Ezek a kamerák egyenként 13 megapixelesek, 1280 x 1024 pixel arányokkal. Képesek 240 képkocka per másodperc sebességgel képet rögzíteni, horizontálisan 56 míg vertikálisan 46 fokos betekintési szög mellett. Az átlagos késleltetésük 4,2 ms ami igazán gyorsnak mondható és nem okoz problémát a valós idejű mozgáskövetésben. A 8 kamerára a csekély betekintési szögek miatt volt szükség, mivel legalább ennyi kamera kell egy nagyjából 40 négyzetméteres terület lefedéséhez.

Ezek a kamerák ethernet kábelen keresztül kommunikálnak az őket vezérlő számítógéppel és ugyanezen a kábelen tudják felvenni a működésükhöz szükséges

áramot. Ebből kifolyólag olyan switch szükséges hozzájuk, ami képes a Power over Ethernet funkcióra.

3.1. ábra. Prime 13 típusú kamera [12]

3.1.2. Markerek

A motion capture technológiához, speciális markerekre is szükség van, ezek is elengedhetetlen részei a rendszernek. Ezek a markerek lényegében olyan golyók, amiknek a felülete magas intenzitással képes visszaverni a fényt. Ezek közül is az infra tartományba tartozó, magas hullámhosszú fényt tudja a leghatékonyabban reflektálni. Így a kamerák, jól el tudják különíteni őket a környezetüktől. A markerek gömb alakúak és emiatt pontforrásként látják őket a kamerák.

3.2. ábra. Fényvisszaverő marker és egy hozzá tartozó rögzítő talp [13]

3.1.3. Motive

Az OptiTrack rendszer hivatalos szoftvere a Motive. Képes a kamerák által modellezett háromdimenziós tér megjelenítésére és annak konfigurálására. [14] Ennek segítségével végrehajthatunk különböző koordináta transzformációkat, definiálhatunk objektumokat, amik több markert csoportosítanak össze és közvetíthetjük a detektált markerek pozícióit, illetve az objektumok orientáció adatait.

A kamerákat UTP kábelekkel csatlakoztathatjuk egy switchen keresztül a Motive szoftvert futtató géphez, ami előzetes beállítások nélkül azonnal felismeri őket. A kamerák kalibrálásához néhány ezer mintát kell gyűjteniük a megfigyelt térből. A mintákat egy erre a célra készített kalibráló rúddal generálhatjuk manuálisan. [15]

3.3. ábra. Kalibráló rúd, 3 markerrel felszerelve [16]

A kalibrálást követően a szoftver kiszámítja a kamerák egymáshoz viszonyított pozícióit és elkészül a modellezett tér, amiben az eszközök helyzetét valós időben figyelhetjük meg.

3.1.4. Robot Operating System

A Robot Operating System vagy rövidebben csak ROS egy nyílt forráskódú, általánosított „operációs rendszer” robotokhoz. Megvalósítja a hardver szintű absztrakciót, képes kezelni a hardverek vezérlését és az egyes folyamatok közti kommunikációt.

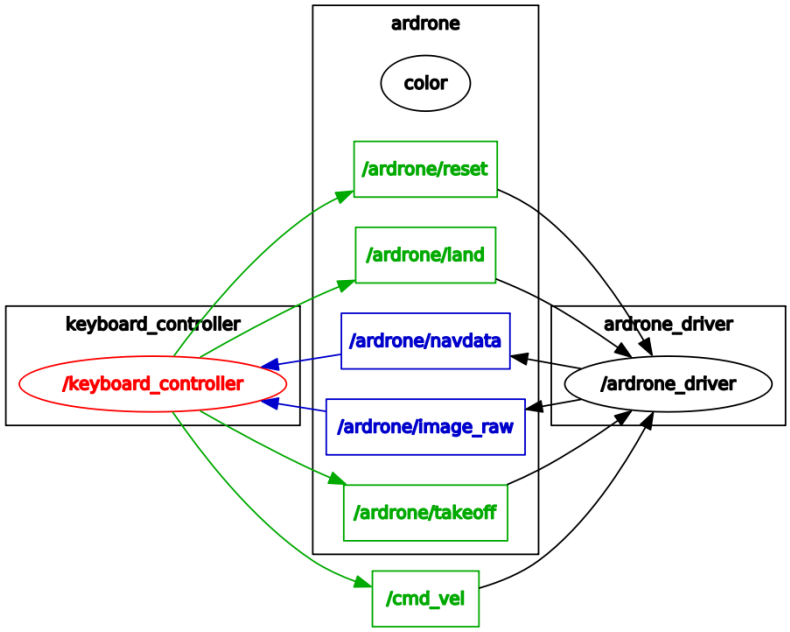
A ROS rengeteg olyan eszközt és könyvtárat biztosít, ami lehetővé teszi a különböző platformokon való fejlesztést. Támogatja a kód újra felhasználást, ami nagymértékben megkönnyíti a fejlesztők munkáját. Rendelkezik beépített vizuális megjelenítővel is, ami kiváló segítséget nyújt a teszt környezetünk szemléltetéséhez a kódot futtató gépen.

A kutatás során a Kinetic verziójú ROS-t használtam, ami teljes mértékben kompatibilis a Xenial verziójú Ubuntu rendszerrel és a kapcsolódó alprogramok is mind, teljes körűen támogatják.

A ROS rendszer építőelemei:

- **Node**: futtatható állomány, ami a ROS-t használja a többi node-dal való kommunikációhoz
- **Message**: primitív típusú adat vagy tömb, amit a node-ok küldhetnek és fogadhatnak
- **Topic**: a node-ok üzeneteket publikálhatnak bele, míg mások feliratkozhatnak rá, hogy megkapják az üzeneteket. Lényegében ezekből épül fel a binárisok közti kommunikációs csatorna
- **Service**: egy node-ban implementált szolgáltatás, ami a ROS-on keresztül meghívható
- **Master**: a ROS rendszer központi eleme, ami számon tartja a különböző node-okat, topic-okat, feliratkozásokat és publikálásokat
- **Parameter Server**: központi tároló, ami a node-ok paramétereit tárolja és közvetíti

[17]



3.4. ábra. Példa a ROS node-ok közötti kommunikáció felépítésre

3.1.5. Számítógépek

A rendszer működtetéséhez 2 darab számítógépre volt szükség. Az egyik Windows 10 operációs rendszerrel, míg a másik VirtualBox-on keresztül Ubuntu Xenial operációs rendszerrel felszerelve. A Windows-os gépen található a Motive ami közvetíti a hálózaton keresztül az objektumok adatait. A másik gép a Robot Operating System és a hozzá kapcsolódó programok futtatásáért felelős. Itt összpontosul a kamerarendszerből kapott és a drón által szolgáltatott összes adat és az itt futtatott programokon keresztül lehetséges a parancsok közvetítése a drón rendszeréhez.

3.1.6. Switch

Az OptiTrack kamerái nem rendelkeznek külön tápellátással. A működésükhöz szükséges áramot a kommunikációhoz is használt ethernet porton keresztül veszik fel. Ezt a funkciót egy átlagos switch nem képes ellátni, mivel nem képes akkor feszültség felvételére, ami ehhez szükséges. Ebből kifolyólag egy olyan switchre volt szükség, ami képes egyszerre akár 8 kamera árammal való ellátására is. A kutatás során használt switch a NETGEAR ProSafe GS728TPP. Ez a switch 24 darab gigabit ethernet porttal rendelkezik, amelyekből mindegyik képes a Power over Ethernet funkció ellátására. Összesen 384 watt leadására képes, ami bőségesen elég a 8 darab Prime 13 típusú kamera működtetéséhez. Továbbá ez a switch alkalmas lesz akkor is, ha az egyetem több kamerával bővítené a tárházát.

3.5. ábra. NETGEAR ProSafe GS728TPP típusú switch patch panele [18]

3.1.7. Router

Rendelkezésre állt még egy LINKSYS WRT1900AC típusú router is, ami képes automatikusan csatlakozni a drón által közvetített Wi-Fi hálózatra. Ez azért lehet fontos, mert a drón fedélzeti számítógépén található rendszer csak olyan hálózathoz tud csatlakozni, amik nincsenek ellátva titkosítással. Így megoldható, hogy a vezérlést végző laptop ne legyen kábelekhez kötve. Ez a router alkalmas akár a legújabb AC szabványú Wi-Fi hálózat létrehozására is, amit tanácsos használni mivel ez az 5 GHZ-es tartományban kommunikál az eszközökkel nem pedig a régi routerek által használt 2,4 GHZ-es tartományban, ami napjainkban kezd kissé túl zsúfolt lenni. Ráadásul ezzel a szabvánnyal sokkal gyorsabb kommunikációt is elérhetünk.

3.6. ábra. Linksys WRT1900AC típusú router és a rajta látható állapotjelző sáv [19]

3.1.8. Drón

A kutatás során a Parrot cég által gyártott AR.Drone 2.0 volt a segítségemre, ami egy távolról vezérelhető, kifejezetten beltéri használatra felkészített négyrotoros pilóta nélküli repülőgép. Wi-Fi kapcsolaton keresztül képes kommunikálni más eszközökkel. Tartalmaz egy 4GB-os tárhelyet ami lehetővé teszi, hogy egyedi algoritmusokat töltsünk fel és automatizáljuk a működését. A drón alsó részén található egy ultrahangos távolságmérő és egy légnyomásmérő szenzor, továbbá be van építve még egy 3 tengelyű giroszkóp, egy 3 tengelyű gyorsulásmérő szenzor és egy magnetométer, amik a drón stabilan tartását hivatottak biztosítani. [20]

3.7. ábra. A Parrot cég AR.Drone 2.0 típusú drónja [21]

3.2. A tesztkörnyezet felépítése

Ez az alfejezetben a teszteléshez használt környezet felépítését és kalibrálásának folyamatát szeretné bemutatni az olvasó számára. Itt is megjegyezném, hogy ezek nem szigorú, inkább csak ajánlott feltételei a helyes használatnak.

3.2.1. Tesztelési helyiségek

Alapvetően két különböző helyszínen történt a mérés. Az egyik a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Karának oktatói klubja, míg a másik ugyanezen Kar edzőterme volt.

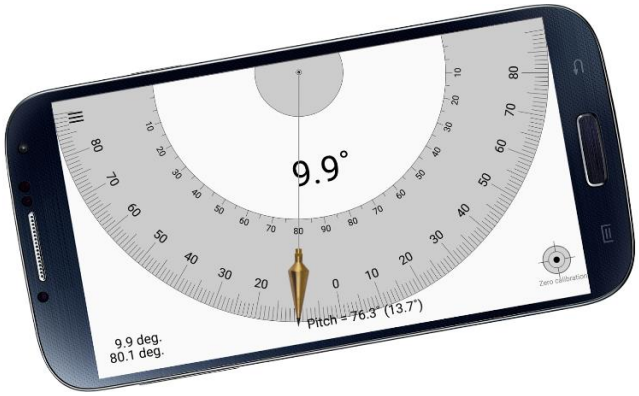
Az oktatói klub alapterülete nem haladja meg a 15-20 négyzetmétert, amiből további lényeges területet foglalnak el az oda tartozó bútorok. Így csekély hely maradt a drónnal való repülésekhez. A belmagasság nem okozott különösebb problémát, viszont a drón közelében lévő tárgyak erősen befolyásolták a repülés trajektóriáját. Ez főleg Bernoulli törvényéből adódik, ami azt mondja ki, hogy egy közeg áramlásakor a sebesség növelése a nyomás csökkenésével jár. Vegyük például azt az esetet mikor a drón közel repül egy falhoz. Ilyenkor a drón mindkét oldalán ugyanakkora sebességgel forognak a rotorok. A légnyomás mindkét oldalon alacsonyabb lesz az eredetihez képest. Azon az oldalon, ahol nincs fal, van elegendő levegő, ami a csökkent nyomású területre áramlik. Ellenben a másik oldallal, ahol a fal miatt nem jut elegendő levegő a területre, így az a drón másik oldala felől áramlik át, ami által a fal mintegy magához szívja a drónt. A jelenség megtekinthető a mellékletben található sucked_by_object.mp4 felvételen.

Néhány alkalommal lehetőségem adódott az edzőteremben való tesztelésre is. Itt már nem jelentkeztek az előző helyiségnél említett problémák és szinte tökéletesen tudta tartani a drón a kívánt pályát repülés közben. A hatalmas belmagasság és a közel 40 négyzetméteres, tesztelésre kijelölt terület nem szabott több határt a repülések közben.

3.2.2. Kamerák elhelyezése

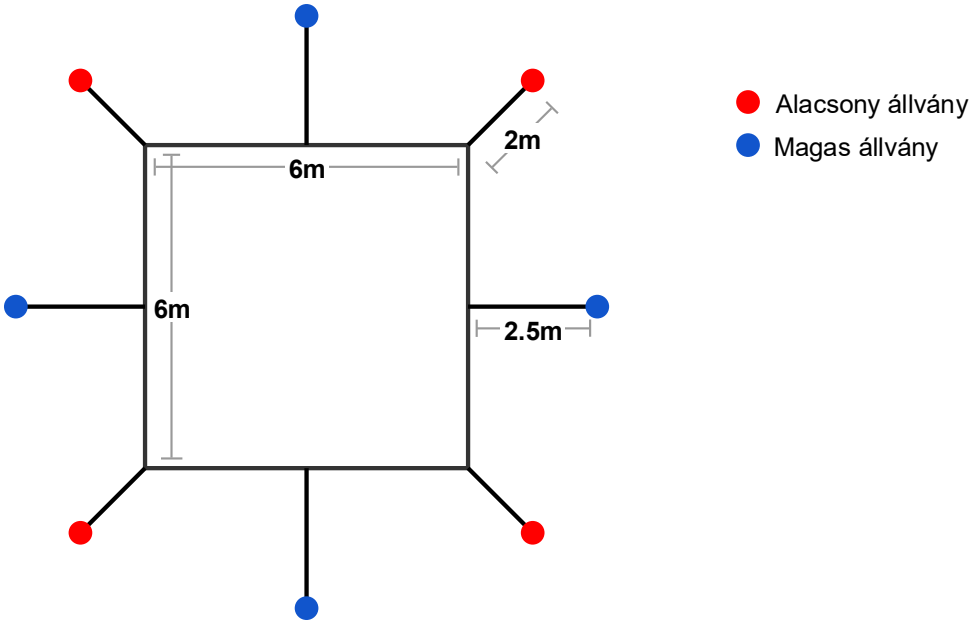
A kamerák, speciálisan erre a célra készített állványokon helyezkedtek el. Ezek az állványok nagy súllyal rendelkeznek, hogy a drón ne tudja őket belengetni repülés közben, ami elrontaná a kamerarendszer kalibrációját. Négy darab alacsonyabb és négy darab magasabb állvány állt rendelkezésre. A magasabb állványok 3 méteres magasságra voltak kiengedve, míg az alacsonyabbak körülbelül 1,5 méteres magasságra. Az alacsonyabb állványokon elhelyezkedő kamerák vízszintesen előre

néztek, míg a magasabb állványokon lévők 50 fokos szöget zártak közre a vízszintessel. A dőlésszög lemérését a Protractor nevű Android alkalmazás segítette.



3.8. ábra. A Protractor nevű Android alkalmazás

Az állványok elhelyezése egy négyzet alapú területen történt, az alábbi képen látható módon.



3.9. ábra. Kameraállványok elhelyezésének alaprajza

3.2.3. Hálózati beállítások

A kamerák a már korábban is említett switchbe voltak közvetlen csatlakoztatva. A switchez csatlakozott még a Windows-os számítógép és a router is. A számítógép fogadta a kamerák képeit, majd közvetítette az objektumok kiszámolt adatait a switchen keresztül a hálózatra. A router létrehozott egy vezeték nélküli alhálózatot, amihez csatlakozhatott a másik számítógép és lehetőség szerint a drón is. A routeren keresztül az internetet is be lehet vezetni a hálózatba, így a tesztelés közben nem kell átváltani másik Wi-Fi-re.

Mivel az Ubuntu rendszert egy virtuális gépen keresztül futtattam, akadtak nehézségek a hálózati kommunikációval. A drón hiába csatlakozott a router-hez, azt a virtuális gépről nem értem el. Megoldásként a gépet ethernet porton keresztül a switchbe csatlakoztattam és a vezeték nélküli hálózati kártyával a drón Wi-Fi hálózatához csatlakoztam. A VirtualBox a gép ethernet portját bridgelve, míg a vezeték nélküli hálózatot NAT-olva érte el. Így lehetőség adódott mindegyik eszközzel való kommunikációra, bár a gép így kábelhez kötve maradt.

3.2.4. Drón konfigurálása

A drón konfigurálható, hogy csatlakozzon egy titkosítatlan vezeték nélküli hálózathoz. Ehhez először fel kell csatlakozni a drón Wi-Fi-jére, majd telnet segítségével bejelentkezni a fedélzeti rendszerébe. Létre kell hozni rajta egy shell scriptet, ami a következőket tartalmazza:

```
killall udhcpd
ifconfig ath0 down
iwconfig ath0 mode managed essid [SSID]
ifconfig ath0 [DESIRED IP] netmask 255.255.255.0 up
route add default gw [GATEWAY IP]
```

A szkriptben látható kék részek hálózat függőek, ezért azokat előbb ki kell deríteni az adott hálózatnál, majd behelyettesíteni őket. A fájl elmentése után engedélyezni kell a futtatását, amit a `chmod +x [FILE PATH]` paranccsal tehetünk meg. Végül zárjuk a telnet kapcsolatot.

Ezek után a legközelebbi használat esetén csak fel kell csatlakozni a drón hálózatára és lefuttatni ezt a parancsot: `echo "./[FILE PATH]" | telnet [DRONE IP]`

3.2.5. Mesterséges akadálypálya tervezése és építése

Az akadálypálya tervezésre és építésre azért volt szükség, hogy a kutatás során egy jól definiált környezetben, egyértelmű feladatokat tudjak definiálni, amivel jól mérhető a haladásom mértéke. Így egy kézzelfogható cél lebegett előttem, amit igyekeztem minél precízebben megvalósítani.

Olyan akadálypályát terveztem, amit egyszerű összeállítani, mégis sok különböző feladat tesztelésére alkalmas lehet. Összesen csak a kalibráló rúd felső része, valamint a drón dobozára volt szükség. Ezen kívül a QR kódokat kellett kinyomtatni a pálya véglegesítéséhez. Praktikus ez a megoldás, mert nem kell plusz eszköz a felállításához. A kalibráló rúd három markerét egyesítettem egy cél objektummá,

aminek a jelentősége, hogy ezen objektum fölé kellett berepülnie a drónnak a saját pozíciója ismeretében. Innen kellett detektálnia a dobozon elhelyezett QR kódot, majd a kód alapján egy bizonyos feladatot végrehajtania. Így nem volt szükség arra, hogy a kezdeti problémát is egy másik keresési algoritmussal kelljen megoldanom, illetve ez által ki lehetett küszöbölni a felszállás közben keletkező, pozícionálási hibákat.

A projekt végén két ilyen feladat valósult meg. Az egyik a doboz felett való átrepülés, majd landolás, a másik a doboz mellett való elrepülés, majd landolás a doboz mögött. Mindkét feladat sikeres tesztje megtekinthető a mellékletben található videó felvételeken.

3.10. ábra. A drón doboza a ráragasztott QR kóddal

3.3. A munka során felhasznált forráskódok

A kutatás folyamán sok olyan nyílt forráskódú programot használtam fel, amik segítik lekezelni az alacsony szintű, hardverközeli programozást és interfészként szolgálnak számomra, hogy magasabb szinten tudjam lekezelni a problémákat. Ezek mind a Robot Operating System részeként működnek együtt és legtöbbjük folyamatos fejlesztés alatt áll, így fontos volt ez esetben is a körültekintő hozzáállás a fejlesztés folyamán.

3.3.1. ardrone_autonomy

Az ardrone_autonomy egy ROS driver a Parrot AR.Drone 1.0 és 2.0 típusú drónokhoz. A ROS-on belül egy külön node-ként jelenik meg, ami a következő fontos topic-okat közvetíti:

- **/ardrone/navdata**: a drón navigációs adatait közvetíti
- **/ardrone/image_raw**: a kamerák képeit közvetíti

Továbbá még számos *service* is meghívható benne, amivel a drón fedélzeti számítógépét vezérelhetjük és előre megírt animációkat hívhatunk elő. [8]

A fejlesztés során problémám akadt a csomag indításáért felelő fájl paraméterezésével. Az indításkor megadható egy paraméter, amivel a drón két kamerája közül tudunk választani, mivel egyszerre csak az egyiket tudja közvetíteni. Ez teljesen jól működik addig, amíg fel nem szeretnénk használni annak a kamerának a képét más programokban. Ez esetben voltak olyan változók, amiknek az értéke nem állítódott át a választott kamera által elvárt helyes értékekre. Ezért szükség volt, a saját kódomban található toggleCam() függvény definiálására, ami ad-hoc módon javítja a problémát.

A csomag alapjaként tekintett ardrone_driver alprogrammal is volt egy gond. Alapvetően ez felelős a drónnak küldött parancsok feldolgozásáért, ami szerint végül a drón cselekszik. Pozíció változtatás esetén elküldésre kerül, hogy a drón melyik tengelyén mekkora szögben dőljön meg. Úgy tapasztaltam, hogy ez a dőlésszög a felszállást követő pillanatokban sokkal nagyobb, mint utána pár másodperccel. Ez a tulajdonság szerintem annak tudható be, hogy az indításkor átadott paraméterek egy kis késleltetés elteltével aktiválódnak csak. Így a drón a dőlésszögre meghatározott maximum értéket is képes átlépni a felszállás utáni röpke idő intervallumban. Ez eléggé instabillá tudja tenni a rendszert.

3.3.2. ardrone_tutorials

Az ardrone_autonomy fölé épült absztrakciós szint, ami segít a drón vezérlését megkönnyíteni. Sajnálatos módon akadt a forráskódjának olyan része, ami nem átgondolt tervezésre utal. Ez abban a részben található, ahol a drónnak szánt parancsok feldolgozása és elküldése történik. Itt meg van fogalmazva egy olyan feltétel, ami szerint felszállás után csak akkor küld további parancsokat a drónnak ha az már nyugalmi, lebegő állapotba került. A tapasztalat azt mutatta, hogy ez nem így történik, sőt a talaj elhagyásának pillanatban máris küldi tovább a parancsokat. Ez okozhat néhány félreértést a fejlesztők számára.

A továbbiakban igyekszem javasolni néhány változtatást a fejlesztői felé. Ebből a tervezési hibából adódóan úgy kellett néhány dolgot lekezelnem a saját programomban, amik szembe mennek a saját fejlesztői elveimmel és így bár kis mértékben de instabilitást okozhatnak. Ugyanakkor érdemes felhasználni, mert a későbbi javítások esetén a saját kódunkon nem vagy csak kis mértékben kell változtatni.

3.3.3. ar_track_alvar

Mint korábban már említettem ez a kis program képes a megadott kameraképen detektálni és dekódolni a QR kódokat. Helyette van lehetőség más megoldások alkalmazására, de a fejlesztés során ez bizonyult a leginkább alkalmasnak.

3.3.4. Virtual-Reality Peripheral Network

A VRPN egy platform független rendszer, ami interfészként működik szoftverek és fizikai eszközök között. Képes kezelni a ROS-sal való topic orientált kommunikációt. Szerver – Kliens modell alapján működik és Internet Protokoll segítségével kommunikál más eszközökkel. [22] A Motive is ezen a rendszeren keresztül képes publikálni az objektumok mért adatait, amit aztán a ROS dolgoz fel és végül lekérdezhető a saját forráskódunkban is. Ehhez szükség van a vrpn_client_ros nevű csomag feltelepítésére.

Sajnos ezen a rendszeren keresztül nem képes a Motive közvetíteni a gyorsulás irányára vonatkozó adatokat. Ennek a kiváltó okára nem találtam logikus magyarázatot sem az interneten, sem pedig az Optitrack fórumain. Amennyiben ez a probléma megoldódik a későbbiekben, ezekkel az adatokkal is tovább lehet fejleszteni a szkript döntési algoritmusát.

3.4. Saját ROS csomag

A fejlesztői munka során egy saját ROS csomagot hoztam létre, ami minden szükséges összetevőt tartalmaz a program futtatásához. Törekedtem arra, hogy a letöltése után szinte azonnal futtatható legyen, különösebb konfigurálások nélkül is. Az egész ROS csomag megtalálható a mellékletek között.

3.4.1. Nyelvezete

A forráskódot python nyelven írtam, amit teljeskörűen támogat a ROS. Saját könyvtárral rendelkezik, ami az egész utasításkészletét támogatja python nyelven. Ez a könyvtár a rospy [23] csomagon belül érhető el. Azért választottam ezt a nyelvet, mert tisztában voltam vele, hogy a futtatásához elegendő egy jól működő interpreter és nem kell fordításokkal bajlódni. A rospy saját megoldásokkal rendelkezik a hibakereséssel és a konzolra íratással kapcsolatban, ami remekül működik közre a ROS többi részével.

A fejlesztések alatt végig igyekeztem tartani magam a PEP8 által meghatározott szigorú szabályokhoz. Ez nagyrészt sikerült is egy-két kivétel mellett. Fontosnak tartottam, hogy jól olvasható, könnyen megérthető kódot adjak ki a kezeim közül, így a szabályok betartása mellett, igyekeztem minél érthetőbb kommentelést is vezetni a kódban. Így nem kell elvesznie az olvasónak a részletekben.

3.4.2. Felépítése

Örömömre szolgált az a felismerés, hogy szkriptnyelv lévén a python igazán barátságos megoldásokat kínál az objektum orientált szemléletű embereknek is. A kódomat jól átlátható, logikailag elkülöníthető blokkokba szerveztem. Az egész forráskód lényegében három logikai egységre lett tagolva, plusz egy úgynevezett launch fájlra, ami egy ROS program lényegi részeinek elindításáért felelős.

Az első és legfontosabb logikai egység egy központi rész, ahol összefut minden szál. Ezt én operator-nak neveztem el, mivel ez felelős a többi egység irányításáért. Itt definiálódik a drón controller objektuma és a másik két egység is. Mindegyik egység megkapja a drón irányításáért felelős objektumot, miután az operator elindította a drónt.

A második egység, az OptitrackController osztály. A elnevezés onnan ered, hogy ez az objektum felelős a Motive által szolgáltatott adatok alapján való tájékozódásért és manőverezésért. Az inicializáló függvényében megkapja a drón irányításához

szükséges objektumot, definiálja a repülés során használt változókat és feliratkozik minden, számára fontos topic-ra, amik a következők:

- **/vrpn_client_node/ardrone/pose**
  A drón pozíció és orientációs adatai egyenesen a Motive VRPN szerverétől
- **/vrpn_client_node/destination/pose**
  A kalibráló rúd pozíció adatai szintén a Motive VRPN szerverétől
- **/ardrone/pose**
  A drón pozíció adatainak transzformált változata. Ezek megegyeznek a drón által értelmezett koordinációs rendszer béli adatokkal.
- **/visualization_marker**
  A detektált QR kódok drónhoz viszonyított, relatív pozíció és orientációs adatai az ar_track_alvar rendszeréből. Itt csak akkor érkezik üzenet, ha sikerült a detektálás.

Az irányításért felelős, lényegi rész a *flying()* függvényében található. Ez a blokk akkor kerül meghívásra, ha a Motive által szolgáltatott helyadatok frissítésre kerülnek. Felmerülhet a kérdés, hogy itt nem okoz e bajt, ha a drón kirepül a megfigyelt területről. Én erre a kérdésre azt a választ tudom adni, hogy tervezési szempontból a tesztek alatt nem volt indokolt egy ilyen hiba lekezelése, mivel ez sokkal inkább az Optitrack rendszer helytelen elhelyezésének vagy kalibrációjának tudható be. Ugyanakkor a szkript futása közben, manuálisan bármikor kiadható a drón számára a landolás megkezdését előidéző parancs: ’`rostopic pub /ardrone/land std_msgs/Empty -1`’, továbbá egy KeyboardInterrupt kivétel előidézésével azonnal megszakítható a további parancsok elküldése, így a drón az aktuális pozícióját fogja megtartani. További szempont, hogy a kód így sokkal átláthatóbb maradt, ami egy tesztelés folyamán csak előnyt jelent.

A *flying* függvény pontos feladata, hogy a drón és a célpont hely adatainak az ismeretében, olyan parancsokat küldjön a drónnak, amik azt a célpont irányába mozdítják el. Egészen addig végzi ezt a műveletet, amíg el nem éri a célpont egy bizonyos nagyságú környezetét. Ezt a méretet a függvény belsejében definiált treshold változóval lehet növelni vagy csökkenteni. Amint a drón beért a kívánt terület fölé elkezdi csökkenteni a magasságát, amíg nem detektálja a doboz oldalán található QR kódot. Erre azért van szükség, mert a drón felszállás utáni alap magassága nagyobb értékre van definiálva, mint a doboz magassága. Ha detektálta a QR kódot, akkor az OptitrackController működése leáll és átadja az irányító szerepét a harmadik egységnek.

A harmadik egység a LandingController osztály. Ezt az elnevezést még a projekt korai stádiuma alatt kapta, mikor az eredeti cél az volt, hogy a drón az általa detektált QR kód felületére landoljon. Az eredeti feladathoz képest változott a

szerepe, de ennek ellenére, a benne található kódok nagyrésze felhasználható maradt. Két úgymond fő függvényt tartalmaz, amik közül az egyik a performAction(), míg a másik a toggleCam().

A toggleCam() lényege, hogy beállítsa a QR kód felismerésére szolgáló kamerát. Ezt kiválasztani a csomag launch fájljában lehet, majd a választást itt kezeli le a program. Így könnyedén lehet kamerát váltani és ugyanakkor olvashatóbbá teszi a launch fájlt.

A performAction() függvény azért felelős, hogy QR kód detektálás esetén a drón a kódhoz rendelt megfelelő animációt hajtsa végre. Tartalmaz egy elágazást, ami biztosítja a felhasználót arról, hogy csak akkor futtatja le az animációt, ha a drón már a kívánt célterület felett van és még nem kezdett bele egyetlen másik animációba sem.

Ezeken kívül van még definiálva két különböző animációs függvény, amiknek a tesztelési eredménye a mellékelt videó fájlokban megtekinthetők. További animációt egyszerűen új függvények definiálásával lehet létrehozni, ahol felhasználható bármilyen külső szenzor adata is a hatékonyság növeléséhez.

3.4.3. Verziókövetés

A fejlesztés során git verziókövető rendszert használtam, amiből adódóan teljes mértékben végig követhető a forráskódok fejlődési szakasza, továbbá könnyedén le lehet ágaztatni újabb branch-eket a hasonló fejlesztésekhez.

3.5. Elkészült munka és mérések dokumentálása

Fontos volt, hogy a munka megfelelően legyen dokumentálva, mivel ez az egyetemen egy teljesen új irányvonal az eddigi kutatások között, így bármilyen információ fontos szerepet játszhat a később bekapcsolódó hallgatók számára. Ehhez én összesen négy platformot vettem igénybe, hogy a munka minden mozzanata és fontos részlete megfelelően legyen dokumentálva. Az első és egyben talán a legfontosabb ez a tanulmány, ami a legtöbb információt és tanulságot hordozza magában. Összefoglalja a kutatás folyamatát az elejétől a végéig és minden lényeges információt tartalmaz, ami a céljaink és ambícióink megértéséhez szükségesek. A második a laboratóriumunk weboldala, [24] ahol igyekszünk összegyűjteni a jövő hallgatói számára az igazán lényeges információkat, amik a projektjeink során készült programok futtatásához és megértéséhez elengedhetetlenek. A harmadik a kódban elhelyezett kommentjeim, amiket igyekeztem minél inkább minimalizálni és

egyértelművé tenni. Olyan programozóként, akinek a szakmai tapasztalatának nagy része objektum orientált környezethez köthető, jól elkülöníthető blokkokat definiáltam a fejlesztéseim során. Így az a kevés komment is egy jól átlátható dokumentációként szolgálhat a fejlesztők számára. Végül az utolsó platform a GitHub, [25] ahol azért vezettem dokumentációt, hogy ne csak a mi egyetemünk hallgatói számára szolgáljak kutatási eredményekkel, hanem a világon bárki számára is, aki érdeklődik a téma iránt. Amellett, hogy itt megtalálható az a leírás, ami alapján a kódom beüzemelhető és futtatható, végig követhető a projektem forráskódjainak fejlődési szakasza, ami további információval szolgálhat, a kutatásom során alkalmazott ötleteimmel kapcsolatban.

4. Értékelés

Alapvetően én egy szoftver közeli embernek tartom magam. Ez előtt nem vettem még részt olyan projektben, ahol úgynevezett kiber-fizikai rendszerekkel kellett dolgoznom. Itt egymás közreműködésével működik a szoftvertechnológia illetve a mechanika és elektronika párosa. Különös volt megtapasztalni, hogy mennyire más gondolkodásmódot követel meg egy ilyen rendszer. Az eddigi fejlesztéseim során hozzá voltam szokva a többnyire lineáris és determinisztikus környezetekhez. Ez eleinte sokszor meglepetéseket szült számomra, majd lassan adaptálódtam ehhez a számomra még új világhoz. Már-már rutin szerűvé vált a fizikai tényezők megismerése egy-egy új program részlet fejlesztése előtt. Annak ellenére, hogy dolgoztam már szimulációs rendszerrel, ott nem érvényesült ilyen szinten a fizika. Örülök, hogy ilyen sok tapasztalattal gazdagodtam a projekt során.

4.1. Az elkészült műszaki alkotás kritikai elemzése

Bár az általam készített program képes végrehajtani azt, amit én célul kitűztem magam elé, ez nem jelenti azt, hogy hibátlan. Természetesen egyetlen program sem lehet hibamentes, így ez sem az. Már említettem az OptitrackController által le nem kezelt hibalehetőséget, ami lehetővé teszi, hogy ha a drón kirepül a megfigyelt területről, akkor olyan parancsokat kapjon, amik esetleg olyan irányba terelik amerre mi nem kívánjuk. Az erre hozott magyarázat meglehet, hogy indokolja ezt a tervezési malőrt, de az biztos, hogy nem oldja meg. Biztos vagyok benne, hogy tudnék rá találni megfelelően működő megoldást, ám ebben az esetben ez csupán időt vett volna el a lényeges részek fejlesztésétől és tesztelésétől.

További hibák adódhatnak a felhasznált ROS csomagok különböző tervezési elveiből adódóan. Mivel több különálló programot is felhasználok a sajátom működtetéséhez, így nagy az esélye, hogy valamelyik kettő között, valamilyen formában inkompatibilitás jöhet létre. Az általam használt verziók használata mellett a tesztelés folyamán nem jelentkeztek ilyen jellegű problémák, de ezt befolyásolhatják a csomagok olyan fejlesztései, amik megszüntetnek vagy bevezetnek különböző funkciókat. Így ebből a szempontból nem mondható időt állónak az általam készített program. Erre megoldásként megtehettem volna, hogy az összes csomag jelenlegi verziójának forráskódját bemásolom az én projektembe és egy közös ROS csomagba fordítom őket, de ez ugyanakkor megfosztott volna attól a lehetőségtől, hogy a különböző csomagokból folyamatosan a legújabb verziókat tudjam használni, elkerülve a linkelés és fordítás bonyodalmait.

4.2. A feladatkiírásban közölt feladatok alapján való értékelés

Ebben az alfejezetben a kiírt feladatok szerint szeretném elemezni a kutatásomat. Így szeretnék egybefüggő képet alkotni az általam végzett munka minőségéről a feladtok adta elvárások szemszögéből.

Az első fontos feladatom a szakirodalom tanulmányozása volt. Ezt igyekeztem minél jobban teljesíteni, tekintve, hogy ezáltal volt lehetőségem hozzájutni a megfelelő kompetenciákhoz. A kutatásokról írt cikkeken kívül rengeteg dokumentációt is olvastam, mert ezek szolgáltatták a legrészletesebb leírásokat a felhasznált eszközökkel kapcsolatban. Úgy vélem elegendő ismeretre tettem szert a témával kapcsolatban, hogy ez ne állja útját a többi feladat teljesítésének.

A következő feladatom a repülő megismerése és repülési tesztek végzése volt. A megismerésnél szerepet játszott az előbb említett dokumentáció olvasása. Azonban itt már azokkal az eszközökkel is jól meg kellett ismerkednem, amelyek a ROS rendszeréből kommunikálnak a drónnal és az irányításért felelősek. Ezek a már említett ardrone_autonomy és ardrone_tutorials csomagok. Mivel már a korábbi kutatásom során is betekintést nyertem a működésükbe, ez a feladat nem jelentett különösebb kihívást. Csupán az ismereteim elmélyítésére volt szükség. A két csomaggal kapcsolatban felfedezett hibák úgy gondolom igazolják, hogy mélységeiben megismertem ezeket a programokat.

A QR kódok felismerésére vonatkozó feladat olyan akadályokat gördített elém, amiknek a leküzdése első körben nem tűnt könnyűnek. A drón jellemzőiből adódó hátrányok mellett igazán nagy kihívásnak tekintettem ezt a feladatot. Úgy gondoltam, hogy nem fogok tudni olyan megoldást találni, ami robusztus módon képes detektálni a QR kódokat a drón alacsony felbontású kameráján keresztül. Meglepetésemre, három olyan megoldást is találtam, amik alkalmasak lehettek volna erre a célra. Így volt lehetőségem a három közül a legmegbízhatóbb megoldást választani.

Az akadálypálya tervezés eleinte fejtörést okozott, majd olyan megoldást találtam, ami egyben praktikus is és sok lehetőséget kínál. A pálya kialakításából adódóan tökéletes összhangban képes működni az Optitrack kamerarendszer és a QR kód felismerés. A kettő segítségével olyan algoritmusok fejleszthetők, ahol az adataik közreműködésével hajthatók végre a feladatok.

A drón irányítása a különféle szenzorok alapján, újabb kihívásokat tárt elém. A koordináta rendszerek nem voltak egységesek, a drón alsó kamerája csak erős fényben volt képes megkülönböztetni a sötét színeket, illetve több apró hiba is előfordult. Ezekre mind tudtam megoldást találni. A koordináta rendszereket transzformáltam, hogy egységes, átlátható rendszerhez jussak. Az alsó kamera gyengeségeit a drón alá helyezett fekete-fehér mintázatú lapokkal küszöböltem ki, így lebegés közben sokkal stabilabban tudta tartani a pozícióját.

Végül nem maradt más hátra, mint az elkészült munka és a mérések dokumentálása. Mint már említettem négy különböző platformon vezettem dokumentációt. Mindegyik más szempontból lehet hasznos. Úgy éreztem ez a négy dokumentáció szükséges ahhoz, hogy a kívánt célközönség minden tagja számára megfelelő minőségű és terjedelmű segédanyagot biztosítsak a kutatásaimmal kapcsolatosan.

4.3. Továbbfejlesztési lehetőségek

A kutatás során több olyan ötlet is megfogalmazódott bennem, ami kiváló témaként tudna szolgálni a többi, kutatást végző hallgatónak. Ebben az alfejezetben szeretném ismertetni az ide kapcsolódó gondolataimat és ötleteimet.

4.3.1. Swarming

Az első ötletem, a projektem továbbfejlesztése swarming feladatok elvégzésére. Akinek nem ismerős ez a szó, annak ismertetném a lényegét. Robotikával kapcsolatos területeken a swarming annyit tesz, hogy több robot egyidejűleg hajt végre egy közös feladatot, amihez összedolgoznak. Lehet ez egy bizonyos formáció megalkotása, vagy tárgyak eljuttatása egyik pontból a másikba. Az interneten további remek ötleteket lehet találni ilyen feladatokhoz.

A gyakorlati haszna egy ilyen kutatásnak, egyrészt az ebben rejlő lehetőségek feltérképezése, másrészt pedig annak a jelenségnek az átültetése a robotikába, amit a hangyák egymással közreműködő viselkedésében is megfigyelhetünk. Néhány összetett feladat elvégzése sokkal egyszerűbb sok kis egység közös munkája révén, mint egyedüliként. Sőt vannak olyan feladatok is amiknek a végrehajtása fizikai lehetetlenség a közreműködés nélkül. [26]

4.3.2. Mozgó objektum követése

A második ötletem még inkább eltávolodik a determinisztikus környezet fogalmától. Itt egy olyan továbbfejlesztett változatát képzeltem el a saját programomnak, ami által a drón képes lenne egy mozgásban lévő objektum követésére. Ez lehet akár a dobozra ragasztott QR kód elmozdulása esetén való követés is, vagy egy mozgó QR kódra való landolás.

4.3.3. Szimulátor

A harmadik ötletem, a korábban ismertetett ar2landing_neural nevű projektben lévő szimulátoros megoldás életre hívása az újabb ROS verziókban. Sajnos az évek során megszűnt a támogatottság, így sokat kell bajlódni a kompatibilitás megoldásában. Ugyanakkor kiváló lehetősége nyílna a tesztelésekre azoknak a kutatóknak, akik nem rendelkeznek a szükséges eszközökkel vagy épp nincs lehetőségük a velük való munkára. A fejlesztések kezdeti fázisában is hasznos lehet amikor az ember még nem igazán rendelkezik konkrét tervekkel és inkább csak ismerkedik a rendszerrel. Így nagyobb drónok biztonságos tesztelésére is nyílna lehetőség.

További komoly feladatként szolgálhat a szimulátor további fejlesztése a fizikai tényezők pontosabb szimulálására. Ez elősegítené a valóságban és a szimulátorban való tesztelés határainak közelebbi kapcsolatát. Így biztosabb lehetne a tesztelő abban, hogy amit a szimulátorban sikeresen letesztelt az a valóságban is hasonlóan fog tudni működni. Ehhez természetesen rengeteg irodalmazásra van szükség, amiben a többrotoros repülőgépek fizikai jellemzőit és a rá ható fizikai erőket kell felkutatni. A megvalósítása igazi kihívás lehet egy programozó számára, így érdemes olyan külső könyvtárakat keresni, amik rendelkeznek ehhez hasonló megoldásokkal.

4.3.4. Nano drón

A negyedik ötletem az egyre inkább elterjedőben lévő úgynevezett nano drónokkal kapcsolatos. Ezek a drónok arról ismeretesek, hogy társaikkal ellentétben igazán kis mérettel rendelkeznek, ezáltal akár egy kisebb lakásban is biztonságosan reptethetők. Így sokkal könnyebb ezeknek a drónoknak a tesztelése is, mert nem hatnak rájuk olyan erősen azok a fizikai tényezők, mint nagyobb társaikra. Például sokkal közelebb tudunk repülni velük tárgyakhoz anélkül, hogy a már említett Bernoulli törvény eredményeképpen, az a tárgy magához szippantsa a drónt. és ezáltal nehezen irányíthatóvá váljon. Ezt lehet ötvözni akár a korábban említett swarming technológiával is, ami által több kisméretű drón akár egy nagyobb méretű, nehezebb objektumot is képes lenne felemelni és átmozgatni egyik pontból a másikba.

4.3.5. Objektum detektálás

Az utolsó ötletem arra irányul, hogy a könnyen felismerhető QR kódok helyett él detekciók és szegmentálások segítségével egész objektumokat legyen képes felismerni a drón. Így sokkal több információval rendelkezne egy-egy körülötte elhelyezkedő tárgyról és pontosabb manőverek végrehajtására is lehetőség adódna. Bár ez a téma inkább a képfelismerés világába kalauzolja el a kutató egyént, de szintén kihívást jelent ezekből az adatokból kiszámolni a megfelelő irányokat, amibe a drónnak el kell mozdulni például egy ütközés elkerüléséhez.

5. Összefoglalás

Úgy gondolom, hogy olyan feladatot választottam, ami méltó kihívásokat állított elém a kezdetektől fogva. Eleinte úgy gondoltam, hogy ez a feladat is hasonló lesz azokhoz a fejlesztői projektekhez, amikben eddig részt vettem az életem folyamán. Az első akadályt a robotikával kapcsolatos csekély mennyiségű ismeretem állította. Bár mindig is érdeklődő szemmel tekintettem ezen témák irányába, sajnos soha nem adódott lehetőségem kipróbálni magamat bennük. Ezen kívül semmilyen pilóta nélküli repülőgépet nem irányítottam a kutatásaim előtti időszakban, így ez is teljesen új tapasztalatokkal szolgált számomra. Véleményem szerint kellő mennyiségű ismeretet szereztem a témában ahhoz, hogy a feladatomat kellő hozzáértéssel és a kívánt minőségben tudjam elvégezni.

A második legnagyobb akadály a python nyelv használata volt, mivel ezelőtt még soha nem használtam a projektjeim során. Már említettem, hogy az egyetemi éveim és az eddigi pályafutásom alatt kivétel nélkül, mindig objektum orientált környezetben és ugyanazzal a szemléletmóddal oldottam meg minden elém gördülő problémát. Eleinte okozott néhány fejtörést a szkriptnyelv használata, viszont hamar megbarátkoztam vele. Tekintve, hogy támogatja saját objektumok definiálását és használatát, szinte mindenre találtam olyan megoldást, ami ennek a szemléletnek sem teljesen idegen. A kódot a PEP8 irányelvei szerint vezettem, így biztos lehetek abban, hogy más rendszereken való használatakor sem fog problémákat okozni a felhasználójának. Igyekeztem minimalizálni a felesleges kódok halmozását, próbáltam mindenre olyan megoldást találni, ami röviden és tömören igyekszik megoldani a problémákat.

A szakdolgozatom célját tekintve, úgy gondolom elértem mindazt, amit szerettem volna. A teljesítendő feladat első felének azokat a részeket tekintettem, ahol a cél az volt, hogy megismerkedjek a tesztkörnyezet minden elemével és tisztázzak magamban minden olyan kérdést, ami a kutatás során felmerülhet. Továbbá azért terveztem egy olyan akadálypályát aminek van kiinduló és végpontja, valamint egy olyan része, amit a drónnak a két pont között be kell járnia, hogy ezáltal könnyedén mérhető legyen az általam elért teljesítmény mértéke. Tekintve, hogy ezen elvárások második fele, azaz a tesztelés sikeresen zajlott és egy kész programot tudhatok magaménak, magában hordozza az első részben meghatározott feltételek sikeres abszolválását is, mivel ezek nélkül igazán nehéz lett volna ezeket véghez vinni. Biztosan akad még fejleszteni való része a programomnak, de a szakdolgozatomban kitűzött feladatokat részemről teljesítettnek tekintem.

6. Köszönetnyilvánítás

Megjegyezném, hogy az egyetem által biztosított eszközökért iszonyúan hálás vagyok, mivel kevés korom béli fiatal fejlesztő mondhatja el magáról, hogy ilyen technológiákkal volt lehetősége dolgozni.

Továbbá köszönet illeti a felhasznált nyílt forráskódú programok és könyvtárak fejlesztőit, akik remek munkát végeztek.

Szeretnék még köszönetet mondani

- témavezetőmnek, **Zsedrovits Tamásnak**, aki lehetővé tette számomra, hogy megismerkedjek a drónok világával és idejét nem sajnálva, tanácsaival és jelenlétével folyamatosan segítette a munkámat

- kutató társamnak, **Bartha Andrásnak**, aki IP hálózati témában mindig tudott logikus magyarázatot adni a felmerülő problémákra

- kedvesemnek, **Tóth Noéminek**, aki végig támogatott és türelemmel viselte a legnehezebb időszakokat

- szüleimnek, **Petrovicz Andreának** és **Petrovicz Attilának**, akik az egyetem eleje óta vakon támogattak céljaim elérésében

7. Irodalomjegyzék

[1] B. Simran, R. Ralph, R. Vishal, R. George és Y. Andrew, „Drones for Deliveries," 08 11 2015. [Online]. Available: http://scet.berkeley.edu/wp-content/uploads/ConnCarProjectReport-1.pdf.

[2] M. Silvagni, „Multipurpose UAV for search and rescue operations in mountain avalanche events," 07 10 2016. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/19475705.2016.1238852.

[3] L. P. Hafsal, „Precision Agriculture with Unmanned Aerial," 2016. [Online]. Available: https://brage.bibsys.no/xmlui/bitstream/handle/11250/2394352/Hafsal.pdf?sequence=1.

[4] E. K. S. A. P. Bamford T., „A real-time analysis of rock fragmentation using," [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1607/1607.04243.pdf.

[5] Airobotics. [Online]. Available: http://www.airobotics.co.il/industrial-facilities/.

[6] E. Team, „Drone crashes into cyclist during Golden State Race Series," 12 05 2017. [Online]. Available: https://www.eedesignit.com/drone-crashes-into-cyclist-during-golden-state-race-series/.

[7] N. B. P. E. F. D. Stephane Piskorski, „AR.Drone Developer Guide," 21 05 2012. [Online]. Available: https://jpchanson.github.io/ARdrone/ParrotDevGuide.pdf.

[8] „ardrone_autonomy — ardrone_autonomy indigo-devel documentation," 04 2014. [Online]. Available: http://ardrone-autonomy.readthedocs.io/en/latest/. [Hozzáférés dátuma: 05 05 2017].

[9] M. Hamer, „Up and flying with the AR.Drone and ROS: Getting started," 10 12 2012. [Online]. Available: http://robohub.org/up-and-flying-with-the-ar-drone-and-ros-getting-started/.

[10] M. Monajjemi, „ROS Answers," 14 02 2016. [Online]. Available: https://answers.ros.org/question/225436/quadrocopter-ardrone-detection-and-tracking-qrcode/.

[11] „OptiTrack - Motion Capture Systems," [Online]. Available: http://optitrack.com/.

[12] „OptiTrack - Prime 13 - Technical Specifications," OptiTrack, [Online]. Available: http://optitrack.com/products/prime-13/specs.html.

[13] „OptiTrack - Motion Capture Markers," OptiTrack, [Online]. Available: https://optitrack.com/products/motion-capture-markers/.

[14] „Motive Documentation - NaturalPoint Product Documentation," 19 08 2016. [Online]. Available: http://wiki.optitrack.com/index.php?title=Motive_Documentation. [Hozzáférés dátuma: 06 05 2017].

[15] „OptiTrack - Calibration Tools," 2017. [Online]. Available: https://optitrack.com/products/tools/. [Hozzáférés dátuma: 06 05 2017].

[16] „OptiTrack - Calibration Tools," OptiTrack, [Online]. Available: https://optitrack.com/products/tools/.

[17] „Documentation - ROS Wiki," 31 03 2017. [Online]. Available: http://wiki.ros.org/. [Hozzáférés dátuma: 05 05 2017].

[18] „GS728TPP | Product | Support | NETGEAR," NETGEAR, [Online]. Available: https://www.netgear.com/support/product/GS728TPP.aspx.

[19] „Linksys WRT1900AC AC1900 Dual-Band Wi-Fi Router,” Linksys, [Online]. Available: https://www.linksys.com/ca/p/P-WRT1900AC/.

[20] „Parrot AR.DRONE 2.0 Power Edition | Parrot Store Official,” Parrot, 2017. [Online]. Available: https://www.parrot.com/us/drones/parrot-ardrone-20-power-%C3%A9dition. [Hozzáférés dátuma: 04 05 2017].

[21] „Parrot AR.Drone 2.0 Review & Rating | PCMag.com,” PCMag, [Online]. Available: https://www.pcmag.com/article2/0,2817,2424345,00.asp.

[22] „vrpn Wiki,” 21 01 2017. [Online]. Available: https://github.com/vrpn/vrpn/wiki. [Hozzáférés dátuma: 06 05 2017].

[23] „rospy - ROS Wiki,” ROS.org, [Online]. Available: http://wiki.ros.org/rospy.

[24] A. Bartha és B. Petrovicz, „uavLab,” 2017. [Online]. Available: https://uavlab.itk.ppke.hu/. [Hozzáférés dátuma: 08 05 2017].

[25] L. Bradford, „What is GitHub and Why Should I Use It?,” 18 02 2017. [Online]. Available: https://www.thebalance.com/what-is-github-and-why-should-i-use-it-2071946.

[26] F. M. Iñaki Navarro, „An Introduction to Swarm Robotics,” 19 06 2012. [Online]. Available: http://dx.doi.org/10.5402/2013/608164.

8. Mellékletek

A dolgozathoz mellékelt lemezen és a http://thesis.petrovicz.com linken megtalálhatóak a következők:

- ardrone_landing ROS csomag
- 3 videófelvétel a kutatás során megvalósított program működéséről
- Ez a dolgozat