

TDK-dolgozat

Szeghy Géza László

2024. november 13.



Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

TDK-dolgozat

HoloLens interfész fejlesztése OptiTrack és ROS által vezérelt drónokhoz

Szeghy Géza László
Mérnökinformatikus BSc
4. évfolyam

2024. november 13.

Témavezető: Bugár-Mészáros Barnabás
(tudományos segédmunkatárs, kutató - HUN-REN SZTAKI)
Belső konzulens: Dr. Benedek Csaba
(egyetemi tanár - PPKE ITK, kutatáskoordinációs igazgatóhelyettes - HUN-REN SZTAKI)

Kivonat

Az elmúlt években a dróntechnológia jelentős fejlődésen ment keresztül, és mára számos iparág számára nyitott új lehetőségeket. Eredetileg főként katonai célokra használták őket, de napjainkra a civil felhasználási területek palettája is jelentősen kibővült. Manapság a drónokat többek között szállítmányozási feladatokhoz, mezőgazdasági munkákhoz, térképezési projektekhez is alkalmazzák, miközben a katasztrófavédelem és a rendvédelmi szervek is egyre nagyobb mértékben támaszkodnak rájuk. A drónok kültéri felhasználása jellemzően GPS-alapú helymeghatározással történik, amely biztosítja a megbízható navigációt és az autonóm mozgást.

A dróntechnológia területén végzett kutatásokat a HUN-REN Számítástechnikai és Automatizálási Kutatóintézet (SZTAKI) többek között a mikroméretű repülőgépek számára kialakított MIMO (Micro aerial vehicle and Motion capture arena (Mikró repülőgép és gépi MOZgáskövetés aréna)) arénában valósítja meg. Az arénában tíz kamerából álló rendszer működik, amelyek infravörös technológia segítségével nagy precizitással képesek követni objektumok mozgását. Ez a környezet ideális a drónokkal kapcsolatos rendszerek tesztelésére és fejlesztésére, mivel lehetővé teszi a valós környezet pontos modellezését és a drón útvonalának nyomon követését.

A dolgozat fő célja egy olyan AR-alapú, HoloLens 2 eszközre fejlesztett interfész elkészítése, amely lehetővé teszi egy drón irányítását pusztán kézmozdulatokkal. A HoloLens 2 eszköz beépített szenzorai (például kamerák) lehetővé teszik a kéz- és fejmozdulatok pontos érzékelését, amely nagyban segítette a fejlesztés ötlet kivitelezését. Ez a megközelítés lehetőséget ad arra, hogy a drón irányítása egyszerűbbé és természetesebbé váljon, hiszen a felhasználónak nem kell külső vezérlőeszközre koncentrálnia, hanem közvetlenül a kezeivel vezérelheti a drónt a térben.

A működőképes rendszert először a SZTAKI 2024-es Kutatók Éjszakáján mutattuk be. A látogatók élőben láthatták és próbálhatták ki a technológiát, és közvetlen vissza-

jelzést adhattak a felhasználói élményről. Az eseményen a fejlesztett AR-alapú irányítási rendszer sikeresen bizonyította, hogy megfelel az elvárásoknak. Tervek között szerepel továbbá a munka összefoglalása egy tudományos cikkbe, amit konferencia kiadványban szeretnénk publikálni.

További kiegészítésként a rendszer képfeldolgozási technikákkal is bővíthető, amelyek lehetővé tennék a távolság- és akadályészlelést. Egy fedélzeti kamera alkalmazásával visszajelző rendszert is integrálhatunk, ami tovább növelné az irányítás biztonságát és pontosságát. Ezek a fejlesztések nagymértékben hozzájárulhatnának a technológia alkalmazhatóságának bővítéséhez különböző ipari és kutatási területeken.

Köszönetnyilvánítás

Szeretnék köszönetet mondani elsősorban témavezetőmnek, Bugár-Mészáros Barnabásnak, hasznos ötleteiért, lektorálásáért, valamint szakmai és kollegális segítségéért.

Köszönöm egyetemi konzulensemnek Dr. Benedek Csabának a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Kar egyetemi tanárának támogatását, mellyel jelentősen hozzájárult munkám elkészüléséhez.

Hálával tartozom Dr. Majdik András László intézményi konzulensemnek, a HUN-REN SZTAKI kutatójának, hogy lehetőséget biztosított és segítséget nyújtott dolgozatom megvalósításához.

Külön köszönöm a HUN-REN Számítástechnikai és Automatizálási Kutatóintézetnek is, hogy rendelkezésemre bocsátotta a dolgozat elkészítéséhez szükséges eszközöket és a megfelelő fejlesztői környezetet.

Tartalomjegyzék

1. Bevezetés	1
1.1. Áttekintés	1
1.2. Célkitűzés	2
1.3. A dolgozat felépítése	2
2. Előzmények	4
2.1. Irodalomkutatás	4
2.1.1. HoloLens 2 bemutatása	4
2.1.2. Crazyflie 2.1 kvadrokopter	5
2.1.3. OptiTrack kamerarendszer	5
2.1.4. A ROS2 keretrendszer	6
2.1.5. A ROS-TCP könyvtárak összehasonlítása	7
2.1.6. Unity platform bemutatása	7
2.1.7. MRTK3 könyvtárcsomag bemutatása	8
2.1.8. Kevert valóság alapú ember-robot interfész	9
2.1.9. Drónvezérlés egykezes gesztusokkal	10
2.2. A projektet megalapozó munkáim bemutatása	10
2.2.1. A drónvezérlő-rendszer megismerése	10
2.2.2. A fejlesztés második lépése	11
2.2.3. Harmadik fázis: Új hardverelem integrálása	11
2.2.4. Eredmények	15
3. Kézfelismerés alapú irányítás és virtuális környezet implementálása	17
3.1. Megoldási lehetőségek ismertetése és megvalósíthatóságuk értékelése . .	17
3.2. Megvalósítási folyamat	18
3.2.1. UDP szerver lecserélése ROS-TCP alapú kommunikációra	18
3.2.2. Drón irányítása kézmozdulatok felismerésével	19
3.2.3. Grafikus elemek: Valós tér modellezése	21

3.3. A drón kameraképének feldolgozása	22
3.3.1. AI deck képfeldolgozás	22
3.3.2. HoloLens kétirányú kommunikációjának lehetősége	25
4. Eredmények	26
4.1. A rendszer tesztelése	26
4.1.1. PTP szinkronizációs eljárás	27
4.1.2. Késleltetési eredmények kiértékelése	28
4.2. Továbbfejlesztési lehetőségek	31
5. Összefoglalás	32
5.1. Elvégzett feladatok összegzése	32
5.2. Kitekintés: Jövőbeli tervek	33
Felhasznált irodalom	37

1. fejezet

Bevezetés

1.1. Áttekintés

A drónokkal kapcsolatos kutatások az utóbbi évtizedben előtérbe kerültek, egyre többet hallhatunk ezen eszközök alkalmazásáról különböző területeken. Az ilyen kutatások ipari felhasználása jelentősen hozzájárul az emberi munka megkönnyítéséhez, mivel a légi robotok már képesek számos egyszerű fizikai feladat elvégzésére. A tenyérméretű drónokat leggyakrabban kutatási célokra használják, mivel ezek kiválóan alkalmasak a valós esetek kisméretű, laboratóriumi körülmények közötti modellezésére.

Jelen dolgozat már korábban elkezdett munkákra épül és több fázisban valósult meg. A feladatok elvégzésének helyszíne a HUN-REN SZTAKI MIMO [1] aréna. A munka első fázisa során egy precíziós drón vezérlő rendszer hibáinak felderítése és javítása után, a második fázisban a drónok vezérlésének AR (kiterjesztett valóság)-beli kiterjesztésével foglalkoztam. Ezen dolgozat részletesen tárgyalja egy olyan interfész kifejlesztését HoloLens 2 eszközhöz, mellyel kontroller nélkül, csupán kézmozdulatokkal lehet intuitív módon, valós időben irányítani egy drónt kamerák látótere által behatárolt térben.

A MIMO arénában elhelyezett OptiTrack [2] egy 10 kamerából álló beltérben felszerelt érzékelő rendszer. Ennek kamerái infravörös hullámokat bocsátanak ki és a követni kívánt objektumról visszaverődött sugárzást érzékelik. A drónok irányításáért két számítógép együttese felelős. A 10 kamerából szimultán érkező, visszavert infravörös sugárzás által keltett jeleket az első számítógép dolgozza fel, és ebből állapítja meg az objektum pontos térbeli helyzetét egy adott koordináta-rendszerben. A számolt adatok alapján rádiós adóval lehet a drónt stabilan vezérelni egy másik számítógép segítségével, mely a ROS (Robot Operating System - Robot Operációs Rendszer) [3] keretrendszert használja.

1.2. Célkitűzés

A projektben legfontosabb szerepet játszó hálózati végponti eszköz a Microsoft által kifejlesztett HoloLens 2¹ AR szemüveg, mely a beépített kameráknak és az ezeket használó szoftvernek köszönhetően képes a tér feltérképezésére. Ez lehetővé teszi egy olyan kevert valóság élmény létrehozását, ahol a környezetünkben megjeleníthetünk ténylegesen nem létező objektumokat, és ezeket közvetlenül manipulálhatjuk. A szemüveg a kamrák be-meneteit egy neurális hálózat segítségével [4] dolgozza fel és beilleszt a térbe egy 3D kéz modellt. Ezen virtuális kéz segítségével lehet interakcióba lépni a kivetített objektumokkal.

A sikeres kivitelezés esetén a kész rendszer segítségével egy drón irányítható pusztán kézmozdulatokkal. Ez lehetővé teszi, hogy a drónt precízen lehessen irányítani, előzetes kezelési tudás nélkül. Az irányítás kezünk mozgásával valósul meg, ami előny a hagyományos kontrollerrel való irányításhoz képest. Ez a módszer kiküszöböli a megosztott figyelem problémáját is vezérlés közben.

A feladatok végrehajtása során Unity² fejlesztői felület segítségével létre kellett hoz-zak egy interaktív kezelői felületet, mely hálózaton keresztül kommunikál a drón vezérlő szerverrel. A fejlesztendő interfész kiegészíti a teret egy virtuális drónnal és egy épülettel, melyek a valóságot modellezik. Az ütközésselkerülés fontos szempont, így ezt különösen figyelembe kell venni és megfelelő módon lekezelni, hogy a valóságban se csapódjon a drón épületbe.

1.3. A dolgozat felépítése

2. Fejezet Az irodalomkutatás során ismertetésre kerülnek a dolgozatban használt technológiák és eszközök, mint például a HoloLens 2, az OptiTrack rendszer, a ROS2 keretrendszer és a projekthez kapcsolódó szoftveres megoldások. Továbbá bemutatásra kerülnek a korábbi munkáim és ezek eredményei, amelyek a jelenlegi kutatás alapját képezik.

3. Fejezet Részletesen megismertetem az interfész és a szerver közötti új kommunikációs rendszert, amely a ROS-TCP Endpoint technológiára épül. Ezen fejezet elmagyarázza

¹<https://www.microsoft.com/hu-hu/hololens/hardware>

²<https://unity.com/>

az új vezérlési rendszer működési elvét, valamint azt, hogy hogyan lett a virtuális tér bevezetve és megjelenítve.

4. Fejezet Ez a fejezet részletesen bemutatja a rendszeren végzett tesztek és méréseket, valamint ismerteti azok menetét, valamint kiértékeli a kapott eredményeket a kitűzött célok fényében.

5. Fejezet Az utolsó fejezet összegezi az elvégzett feladatokat, valamint ismerteti a jövőbeli lehetséges fejlesztési irányokat, beleértve a rendszer optimalizálását, a funkcionálisok bővítését és a potenciális alkalmazási lehetőségeket különböző környezetekben.

2. fejezet

Előzmények

2.1. Irodalomkutatás

2.1.1. HoloLens 2 bemutatása

A Microsoft HoloLens 2 egy olyan kiterjesztett valóság (AR) szemüveg, amelyet különféle ipari és fejlesztési célokra is használhatunk. A HoloLens 2-ben alkalmazott technológia lehetővé teszi valós idejű holografikusan renderelt objektumok megjelenítését, amely számos kutatási területen nyújthat segítséget. Az 52 fokos szögátmérővel rendelkező kijelző precíz vizuális élményt biztosít, amely megkönnyíti a holografikus elemekkel való interakciót[5]. A HoloLens 2 a Windows Holographic operációs rendszert használja. Ez a fejlesztők számára egyszerűsíti a különféle alkalmazások létrehozását és használatát, mivel egy univerzális platformra fejlesztenek.

A szemüveg egyik különlegessége az, hogy lehetővé teszi az AR élmények fokozott mértékű interaktivitását és személyre szabhatóságát, mivel nem csupán passzív megjelenítő eszközként szolgál, hanem aktív módon támogatja a kézmozdulatok felismerését és a fejmozgás követését is. Ezáltal a felhasználó természetes gesztusokkal tud manipulálni virtuális objektumokat, forgatni, méretezni vagy akár áthelyezheti azokat a valóságban látható tárgyakhoz viszonyítva. A fejmozgás figyelembevételével a szemüveg dinamikusan alkalmazkodik a nézőpontváltozásokhoz is, bármerre fordul a felhasználó, az AR elemek stabilan illeszkednek a környezethez, tartva az elhelyezkedésüket.

A kézfelismerési rendszer mélységi érzékelőket és mesterséges intelligencia alapú algoritmusokat alkalmaz, hogy precízen követni tudja a kéz mozdulatait és helyzetét. [4]

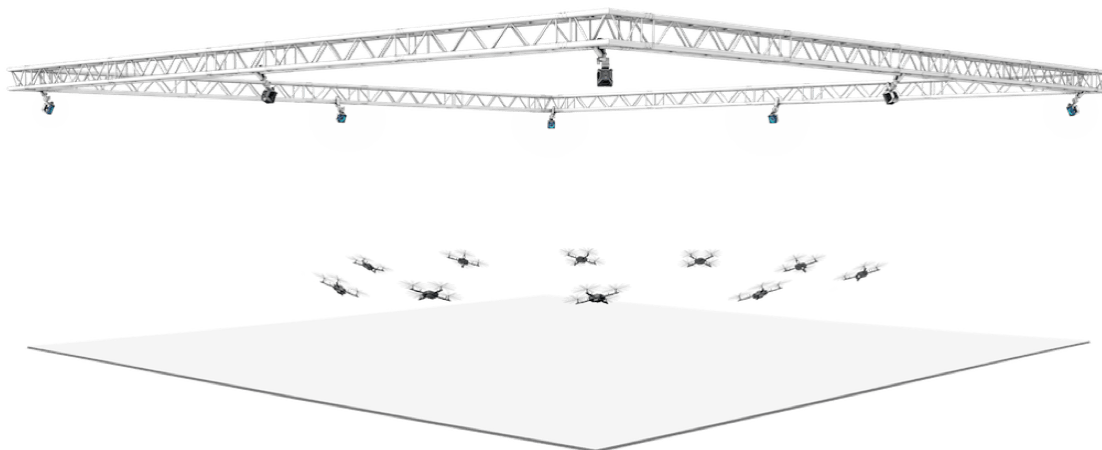
Ezáltal a felhasználóknak nem szükséges fizikai vezérlőket használniuk, ami természetesebb interakciót tesz lehetővé a környezetükkel. A rendszer specifikációi között szerepel az ujjmozgások, különböző gesztusok, valamint az ujjbegyek pozíciójának felismerése és lokalizálása, amely lehetővé teszi a megfelelően pontos manipulációt. Ez különösen hasznos olyan helyzetekben, ahol a felhasználók komplex holografikus eszközökkel dolgoznak, például orvosi beavatkozások során vagy virtuális eszközök kezelésekor.

2.1.2. Crazyflie 2.1 kvadrokopter

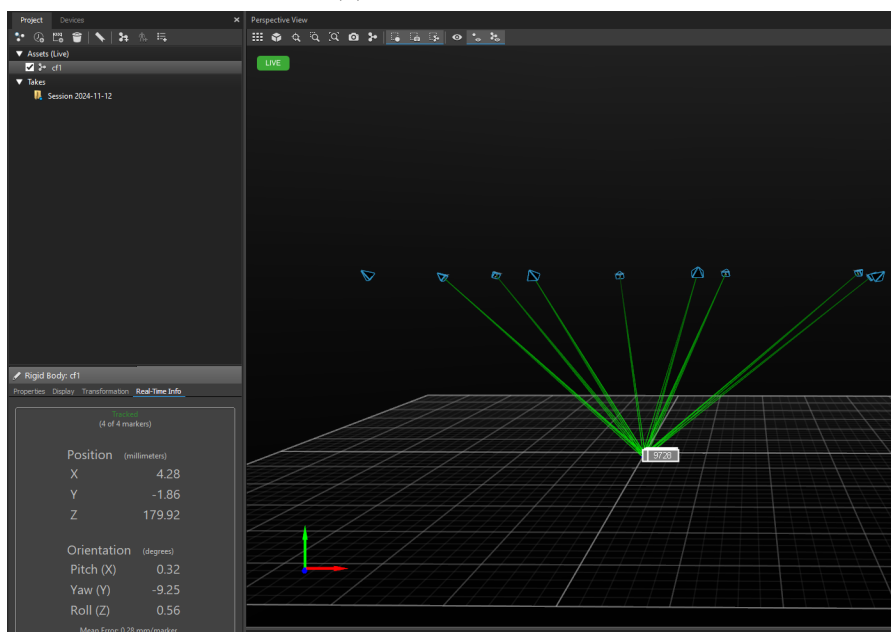
A Crazyflie 2.1 [3.2 ábra] egy kompakt, moduláris drónplatform, amelyet oktatási és kutatási célokra terveztek. Ez a drón ideális választás az olyan felhasználási területeken, ahol szükség van a gyors prototípus-készítésre és a rugalmas konfigurációra. A Crazyflie drón integrálható a ROS2-t alkalmazó projektekbe [6], így lehetségessé válik a robotikai alkalmazásokkal való együttműködés. A drón (15 grammos teherbírását figyelembe véve) szabadon bővíthető különféle szenzorokkal, például kamerával és mozgásérzékelőkkel, ami lehetőséget ad az adatelemzésre és a komplex feladatok végrehajtására. Ezen kívül a Crazyflie rendelkezik egy stabil, mégis könnyen kezelhető vezérlési mechanizmussal, amelyet akár mobil eszközökről is használhatunk, így növelve az irányítási lehetőségek sokféleségét.

2.1.3. OptiTrack kamerarendszer

Az OptiTrack egy megbízható és széles körben alkalmazott mozgáskövető rendszer, amely kiválóan teljesít különböző területeken, mint a biomechanika, robotika és virtuális valóság. Egy kutatás [7] feltárta az OptiTrack alkalmazásainak sokféleségét, és a rendszer hitelesítési vizsgálataiból kiderült, hogy megbízható és egyben egy megfizethető választásnak bizonyul. Az OptiTrack rendszer lehetőséget biztosít kisebb költségvetésű kutatócsoportok számára is, ezáltal növelve a hozzáférhetőséget és a kutatások sokféleségét. A rendszer számos infravörös kamerát alkalmaz, amelyek a mozgó tárgyakat jelölők segítségével azonosítják és követik a kamerák által lefedett térben. Az OptiTrack alkalmazása különösen fontos a bonyolult mozgások és gyorsan változó helyzetek pontos leképezésében, ami hasznos a robotikában és az emberi interakciókat vizsgáló kutatásokban.



(a) Kamerarendszer¹



(b) Motive² kezelőfelület

2.1. ábra. OptiTrack

2.1.4. A ROS2 keretrendszer

A Robot Operating System 2 (ROS2) egy nyílt forráskódú keretrendszer, amelyet kifejezetten robotikai és automatizálási alkalmazásokhoz fejlesztettek ki. A ROS2 továbbfejlesztette az eredeti ROS funkcionalitását [8], különös figyelmet fordítva a párhuzamosított adatmegosztásra. Ez a decentralizált keretrendszer sokkal rugalmasabb környezetet hoz létre az ipari és kutatási alkalmazások számára. A ROS2 felépítése hasonló a ROS-hoz, de optimalizált kommunikációs protokollokat használ, amelyek nagyobb megbízható-

¹<https://www.optitrack.com/systems/#robotics/primex-13/8>

²<https://www.optitrack.com/software/>

ságot biztosítanak. A ROS2-ben az adatkapcsolatok közvetlen peer-to-peer kapcsolatokat tesznek lehetővé, miközben megőrzi a ROS által kínált moduláris rendszert, amelyben különböző csomópontok és szolgáltatások használhatók fel. Az üzenetkezelési rendszer segítségével létrehozhatunk különféle csatornákat a csomópontok közötti adatcseréhez. Az üzenetek küldése és fogadása publikálással és feliratkozással lehetséges. A ROS2 tehát alkalmas olyan alkalmazások kommunikációs struktúrájának felépítésére, ahol nagy pontosság és időzíítési követelmények szükségesek, például autonóm járművek vagy ipari robotok irányítása során.

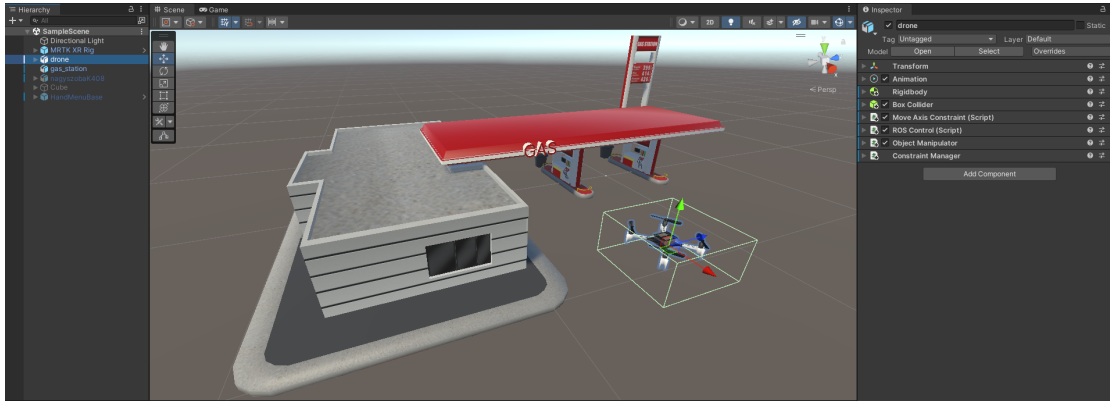
2.1.5. A ROS-TCP könyvtárak összehasonlítása

A ROS-TCP könyvtárak használata lehetővé teszi a ROS és Unity közötti kommunikációt, ami különösen hasznos a VR/AR alkalmazások esetében, ahol az adatok valós idejű szinkronizációjára van szükség különböző platformok között. Egy tanulmány [9] összehasonlít különböző implementációkat késleltetés, publikálási sebesség és CPU kihasználás szerint. Kiemelten három könyvtárral foglalkozik, ezek a ROS-TCP-Connector, ROS# és ROS.NET. A ROS-TCP-Connector könyvtár, amelyet a Unity fejlesztett ki, kisebb és közepes méretű üzeneteknél mutatott kiemelkedő teljesítményt. Ugyanakkor nagyobb adatmennyiség esetén a ROS.NET bizonyult hatékonyabbnak, mivel közvetlen kapcsolatot tesz lehetővé, így elkerülve a köztes csomópontok által okozott késleltetéseket. A ROS# esetében a teljesítmény jelentősen visszaesett a nagyobb üzenetek esetén, amit főként az üzenetek JSON alapú szerializációja okozott. A teszteredmények alapján a ROS-TCP-Connector hatékony választás kisebb adatforgalom esetén, míg a ROS.NET előnyösebb a több gépre kiterjedő alkalmazások és nagyobb adatmennyiségek esetében.

2.1.6. Unity platform bemutatása

A Unity egy széles körben használt, több platformon működő fejlesztői környezet, amely különösen népszerű a 3D-s és 2D-s játékok, interaktív alkalmazások, valamint virtuális és kiterjesztett valóság (VR/AR) megoldások fejlesztésében. Az eszközt először 2005-ben mutatták be, és azóta a játékipar egyik alapvető eszközévé vált. Ma már széles körben alkalmazzák az ipar és a tudomány más területein is, például a szimulációk, a képzési rendszerek és az oktatási platformok fejlesztésében. A Unity lehetőséget biztosít a fejlesztők számára, hogy egyetlen projektből többféle eszközre, például PC-re, konzolokra, mobil eszközökre, valamint VR és AR eszközökre is fejleszthessék az alkalmazásaikat. A fejlesztői környezet vizuális szerkesztője intuitív, amelyben különféle komponensek segít-

ségével lehet összerakni a jeleneteket (Scenes) és az objektumokat, miközben a programozási feladatokat főként C# nyelven lehet megvalósítani. A Unity kiterjeszthetősége lehetővé teszi a külső eszközök és könyvtárak integrálását.



2.2. ábra. Unity szerkesztő

2.1.7. MRTK3 könyvtárcsomag bemutatása

Az MRTK3 (Mixed Reality Toolkit 3) [10] egy nyílt forráskódú eszközkészlet, amelyet a Microsoft fejlesztett ki a kiterjesztett valóság (AR) és a vegyes valóság (MR) alkalmazások fejlesztésének megkönnyítésére Unity környezetben. Az MRTK3 a korábbi MRTK2 továbbfejlesztett változata, amelyben számos új funkciót és optimalizációt vezettek be, hogy a fejlesztők egyszerűbben hozhassanak létre intuitív, interaktív MR élményeket különböző eszközökön, mint például a HoloLens 2 és egyéb VR/AR headsetek. Az MRTK3 moduláris felépítésének köszönhetően a fejlesztők kiválaszthatják és testre szabhatják azokat a funkciókat, amelyek legjobban illeszkednek a projektjük igényeihez, így csökkentve az alkalmazás erőforrás-igényét és növelve a teljesítményt.

Az MRTK3 integrált támogatást nyújt a különböző bemeneti típusokhoz, beleértve a kézmozdulatokat, a hangvezérlést és a szemmozgás-követést, így a fejlesztők a felhasználók számára természetesebb interakciókat biztosíthatnak a virtuális objektumokkal. Emellett az MRTK3 új rendszerei lehetőséget biztosítanak a felhasználói felületek gyors létrehozására, mivel előre definiált elemeket, például menüket, gombokat és slider komponenseket tartalmaz, amelyeket kifejezetten MR és AR környezetekhez optimalizáltak. A rendszer támogatja az OpenXR protokollt, így több platformon is könnyen adaptálható és biztosítja, hogy az alkalmazások zökkenőmentesen működjenek a Unity és más MR platformok között.

Az MRTK3 kiválóan integrálható a különböző Unity-alapú projektekbe, és a fejlesztők munkáját jelentősen megkönnyíti a részletes dokumentáció, útmutatók és példaprojektek széles választéka, amelyek különösen hasznosak a kezdők számára. A közösségi támogatás és a rendszeres frissítések biztosítják, hogy az MRTK3 egy folyamatosan fejlődő eszközkészlet maradjon, amely lehetővé teszi a gyors prototípusok létrehozását és a stabil, professzionális szintű MR alkalmazások fejlesztését. A teljesítményre optimalizált, rugalmasan testreszabható megoldások támogatják a valós idejű interakciókat és az összetett rendszerek szinkronizálását, így ideális választás a különböző AR/MR alkalmazásokhoz.

2.1.8. Kevert valóság alapú ember-robot interfész

A „Design and Evaluation of a Mixed Reality-based Human-Robot Interface for Teleoperation of Omnidirectional Aerial Vehicles” című cikkben [11] egy kevert valóság (MR) alapú interfészt mutatnak be, amely segítségével a drónpilóták számára egy hat szabadságfokú (6DoF) OMAV (Omnidirectional Aerial Vehicles) irányítását. A kutatómunka kiemeli az MR-alapú irányítás intuitív használhatóságát és hatékonyságát, ami független a felhasználók korábbi tapasztalataitól. Ez a megközelítés lehetőséget nyújt a pilóta számára, hogy egy drónt irányíthasson a térben a hagyományos hardverek okozta akadályok nélkül, mint például joystickok használata.

A szakirodalom alapján a szerzők a hagyományos vizualizációs módszerekkel szemben az MR nyújtotta előnyökre helyezik a hangsúlyt. Míg a haptikus és joystick alapú rendszerek korlátozott irányíthatóságot biztosítanak, az MR interfészek lehetőséget nyújtanak új irányítási és visszacsatolási módszerek felfedezésére, mivel ezek bármilyen kívánt formában és pozícióban megjeleníthetők. Az eredmények azt mutatják, hogy a felhasználók MR tapasztalata hatással van a feladat teljesítési idejére, ezáltal javasolható a technológia oktatási, képzési környezetben való alkalmazása is. A cikk további kutatási irányként javasolja az MR alkalmazhatóságának vizsgálatát különböző környezetekben és OMAV típusoknál.

Ez a megközelítés egy ígéretes alternatívát kínál a teleoperációs felületek megtervezéséhez. Az MR segítségével az irányítók az OMAV teljes pozícióját és orientációját, valamint környezetét részletesebben láthatják, ami biztonságosabb és pontosabb vezérlést eredményezhet. Az MR-alapú irányítás tehát olyan megoldás, amely jelentősen csökkentheti a pilóta kognitív terhelését, különösen összetett környezetekben.

Jelen dolgozat egy a fentiekben leírt rendszerhez elvben hasonló, de működésében és kivitelezésében teljesen különböző rendszert és hozzá tartozó interfészt, illetve annak fejlesztési lépéseit és körülményeit mutatja be.

2.1.9. Drónvezérlés egykezes gesztusokkal

A „Drone Control in AR: An Intuitive System for Single-Handed Gesture Control, Drone Tracking, and Contextualized Camera Feed Visualization in Augmented Reality” című cikkben [12] egy olyan AR-alapú rendszer kidolgozásáról számolnak be, amely egykezes gesztusokkal lehetővé teszi a drón irányítását. A HoloLens 2 segítségével a pilóták intuitív gesztusokkal vezérelhetik a drónt, emellett AR-ben követhetik annak pozícióját és figyelemmel kísérhetik a drón kameraképét. Az innováció lényege, hogy a pilóták folyamatosan szem előtt tarthatják a drónt és környezetét anélkül, hogy átváltanának a fizikai és a virtuális nézetek között, így csökkentve a kognitív terhelést.

Az egykezes gesztusvezérlés jelentősen növelheti a drónvezérlés intuitív használatát és csökkenti annak tanulási időigényét. Ezenkívül az AR alapú kamerakép megjelenítés hozzájárul a helyzetfelismerés javításához, különösen akkor, amikor a drón távol van vagy rossz fényviszonyok között üzemel. A gesztusok finomsága és az AR használata révén a rendszer egy természetes felhasználói interfészt biztosít, amely a hagyományos távirányítós vezérlésnél jobban kihasználható stresszes helyzetekben, például katasztrófaelhárítás során.

2.2. A projektet megalapozó munkáim bemutatása

2.2.1. A drónvezérlő-rendszer megismerése

HUN-REN SZTAKI kutatóintézetben az első feladataim közé tartozott a környezet (OptiTrack kamerarendszer, Crazyflie drónok [13]) alapos megismerése. Ezek után a MIMO arénában telepített, előzetesen megismert eszközöket használó drón vezérlő rendszer [14] lokális stabilitási hibáit kellett felderíteni és kijavítani. Többek között fizikai (infravörös zaj, visszaverődés) és szoftveres problémákat oldottam meg, de lehetőségem nyílt elmerülni a Docker és a ROS működésének tanulmányozásában is, ami nagy mértékben segített megvalósítani a projektet. Az itt ismertetett rendszer lehetővé teszi egy tenyér méretű drón térbeli milliméter pontosságú pozícionálását és mozgásának irányítását. Felépítése a következő: a drónra erősített jelölők (marker) segítségével az OptiTrack kamerarendszer szoftvere meghatározza a drón térbeli helyét és irányát/orientációját. A

kapott adatokat egy másik számítógép feldolgozza és ezek alapján ad instrukciókat a drónnak. A vezérlés egy controllerrel történik, melynek segítségével egy virtuális pont irányítható a térben. Ezt a kvadrokopter folyamatosan próbálja megközelíteni a pont felé konvergálva.

2.2.2. A fejlesztés második lépése

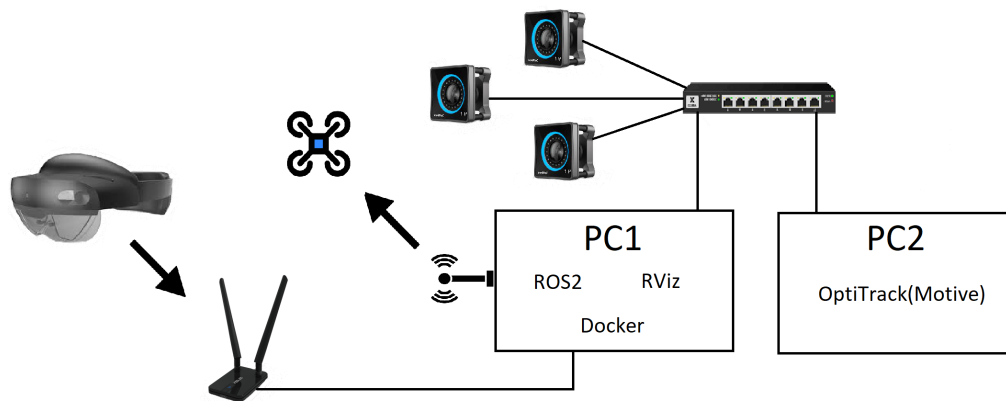
A munkának ebben a szakaszában, ami a szakmai gyakorlatom folyamán történt, telepítettem a Crazyflie projekt egy továbbfejlesztett, már ROS2-t alkalmazó verzióját [6] az arénában. Ehhez össze kellett kapcsolni azt a SZTAKI-ban található eszközökkel, megoldva a kompatibilitási problémákat. Ezek után elkezdtem az előző projektből átvinni az általam újraimplementált vezérlési algoritmust. Ehhez alaposan meg kellett ismernem a ROS2 környezetet is, mivel a ROS előző verziója nem kompatibilis vele. Egyes szolgáltatások eltérően működnek. A projekt átmozgatásának folyamatát ROS migrációnak [15] nevezzük. Az intézetben általam használt számítógépen Ubuntu 20.04 (Focal Fossa) operációs rendszer található, amivel a ROS2 Galactic verziója volt kompatibilis.

A migráció során a legnagyobb kihívást az jelentette, hogy a ROS2 más kommunikációs protokollal működik. Az első verzió még egy központosított rendszert használt, ahol a becsatlakozó felek a mesteren keresztül kommunikálnak. A ROS2 már egy decentralizált rendszert valósít meg, ahol a kommunikáló felek csomópontokként (node) értelmezhetőek, és képesek párhuzamosan adatot közvetíteni egymás között. A kommunikáció témák (topic-ok) segítségével történik, az ezekre feliratkozó csomópontok részt vehetnek a beszélgetésben. A megfelelő működés eléréséhez Python szálkezelést kellett alkalmaznom, mivel a ROS2 futása blokkolta volna a drón vezérlő programot. A kód migrálását sikeresen véghezvitettem, és az előző projekttel hasonló működést sikerült implementálnom egy stabilabb, robusztus környezetben.

2.2.3. Harmadik fázis: Új hardverelem integrálása

Az eddigi rendszer refaktorálása és továbbfejlesztése után áttértem az új vezérlő rendszer megvalósítására. A bevezetésben megemlítettem, hogy az új vezérlő rendszer a Hololens 2 augmentált valóság szemüveggel lesz implementálva. Önálló Laboratórium során sikerült elkészíteni az interfész egy kezdetleges verzióját. Unityben létrehoztam egy alkalmazást, mely képes kommunikálni hálózaton keresztül a drónt felügyelő szerverrel. A megvalósításhoz egy egyszerű kocka objektumot hoztam létre, mely a drón pozícióját reprezentálta térben. A szemüveg koordináta-rendszere szerinti pozíciót kellett eljuttatni

a szerverhez, a távirányító megoldásához hasonlóan. A kommunikációt C# programozási nyelven írt szkripttel valósítottam meg, mely egy UDP (User Datagram Protocol) klienst hoz létre. Ez ciklusonként lekérdezi és stringbe csomagolt üzenetként elküldi a kocka pozícióját a szervernek. Ezt követően a pozíció és rotáció vektorok feldolgozásra kerülnek és a drón korrigálja helyzetét a térben az utasításoknak megfelelően. A felhasználó mozgathatja a kockát a kezével. Ehhez egy összecsíppő mozdulatot kell imitálnia, melyet a HoloLens a kamerái segítségével észlel és innentől a kéz mozgásának megfelelően manipulálható a kocka pozíciója, illetve orientációja. Természetesen a kocka csak a földhöz képest függőleges tengely körül forgatható. A kommunikációs rendszer felépítése szemléletesen:



2.3. ábra. A rendszer vázlatos rajza

Az első lépés egy Unity projekt elkészítése volt, ami kompatibilis a HoloLens 2 eszközzel. Ezt egy Microsoft oktatóanyag [16] követésével hoztam létre. A projekt beállítása során megismerhettem a szoftver alapvető működését és elsajátíthattam annak alapszintű kezelését. Miután beállítottam az AR szemüveghez szükséges paramétereket, létrehoztam egy kocka alakú új objektumot. A későbbiekben ezt mozgatva lesz lehetséges a drónt irányítani. A kocka oldalait az OptiTrack Motive szoftver tengelyszínei alapján színezttem be, illetve megoldottam, hogy az objektum kényszerítetten csak a függőleges tengelye körül legyen forgatható. Ez azért szükséges, hogy a drónt repülés közben ne utasíthassuk olyan művelet végrehajtására, amely annak lezuhanását vagy ütközését okozhatná.

A következő lépésként meg kellett oldanom a valós idejű kommunikációt a szemüveg és a drónt vezérlő számítógép között. Mivel a HoloLens 2 csak Wi-Fi segítségével képes adatokat küldeni, USB kábel nélkül, ezért mindenképpen ebből kellett kiindulnom. Két hálózati protokoll merülhetett fel a megoldáshoz: a TCP (Transmission Control Protocol)

és az UDP. A készítendő rendszer elvárásai közé tartozik a valós idejű irányíthatóság, így az UDP mellett döntöttem, hiszen ez a protokoll nem vár visszaigazolást a hálózati csomagokról, így folyamatosan, várakozás nélkül lehet adatokat küldeni.

A kommunikáció megvalósításához a C# és a Python programozási nyelveket használtam. Létrehoztam egy szervert Python segítségével a fogadó számítógépen, illetve egy klienst C#-ban, amely program az AR szemüvegen fut. A kliens működése a következőképpen alakul: minden iterációban lekérdezi a hozzárendelt objektum pozícióját (jelen esetben az előzetesen létrehozott kocka), és azt UDP csomagba foglalva továbbítja egy megadott IP címre. A szerver hasonlóan, végtelen ciklusban futva hallgat az előre definiált porton, és ha hálózati csomag érkezik, akkor azt dekódolja.

Ahhoz, hogy ez a rendszer működjön, egy hálózatra kellett csatlakoztassam a kommunikációban részt vevő eszközöket. A SZTAKI belső hálózata erre nem volt alkalmas, mivel bizonyos okokból le van tiltva a közvetlen kommunikáció. Ezt a problémát azzal küszöböltem ki, hogy a telefonomat az említett hálózatra csatlakoztatva hozzáférési pontként használtam, így létrehoztam egy saját lokális hálózatot, amely már nincs letiltva. Ehhez csatlakoztatva a két eszközt a kommunikáció akadálytalan. A program megfelelő működéséhez még be kellett állítanom, hogy a fogadó számítógép statikus IP címet használjon, hiszen nem szerencsés ha a kliensben megadott érték eltér a valóságban használttól az automatikus DNS beállítások miatt.

A munka során a drón függőleges tengely körüli forgatását (yaw) is meg kellett oldanom. Ekkor még nem tudtam, hogy létezik implementált függvény, amellyel közvetlenül le lehet kérni az adott objektum Euler szögét, így megismerkedtem a kvaterniók fogalmával a probléma megoldása érdekében. A drón irányításához a számítógép egy kvaterniót is kap bemenetként a pozíció mellett, és ez alapján utasítja a drónt.

A kvaternió egy olyan matematikai fogalom, amelynek segítségével leírható a térbeli objektumok orientációja. Négy komponensből áll: egy skalár (w) és három vektoriális (x , y , z) rész. Ez a következő egyenlettel írható fel:

$$q = w + xi + yj + zk$$

Mivel az Euler szöget le tudtam kérni a Unity projektben, ezt küldtem át a számítógépnek feldolgozásra. A rendszer kvaterniókkal dolgozik, melyek az alábbi módon számíthatóak ki egy Euler szögből:

- Euler szögek felosztása félszögekre:

$$\begin{aligned} cr &= \cos\left(\frac{roll}{2}\right) & sr &= \sin\left(\frac{roll}{2}\right) \\ cp &= \cos\left(\frac{pitch}{2}\right) & sp &= \sin\left(\frac{pitch}{2}\right) \\ cy &= \cos\left(\frac{yaw}{2}\right) & sy &= \sin\left(\frac{yaw}{2}\right) \end{aligned}$$

- Komponensek kiszámítása:

$$\begin{aligned} q_w &= cr \cdot cp \cdot cy + sr \cdot sp \cdot sy \\ q_x &= sr \cdot cp \cdot cy - cr \cdot sp \cdot sy \\ q_y &= cr \cdot sp \cdot cy + sr \cdot cp \cdot sy \\ q_z &= cr \cdot cp \cdot sy - sr \cdot sp \cdot cy \end{aligned}$$

A roll és a pitch értékeket 0-ra kellett állítsam, hiszen most csak a függőleges tengely körüli forgatásról van szó, a yaw értéket pedig közvetlenül le tudtam kérni.

Az OptiTrack kamerarendszer koordináta rendszere nem egyezik meg a szemüveg koordináta rendszerével, egyes tengelyek iránya és elhelyezkedése különbözik, így koordináta transzformációt kellett végrehajtanom a kapott adatokon a helyes működés eléréséhez. Ez implementálva a fogadó fél oldalán következőképpen néz ki:

```

1  def setMsgFromHoloData(self):
2      data, addr = self.sock.recvfrom(1024)
3      coords = data.decode()
4      if coords != None:
5          data = coords.split(',')
6          x, y, z, rx, yaw, rz = map(float, data)
7          self.msg.pose.position.x = float(z)
8          self.msg.pose.position.y = -float(x)
9          self.msg.pose.position.z = float(y)
10
11         quaternion = quaternion_from_euler(0, 0, radians(float(-yaw)))
12         self.msg.pose.orientation.x = quaternion[0]
13         self.msg.pose.orientation.y = quaternion[1]
14         self.msg.pose.orientation.z = quaternion[2]
15         self.msg.pose.orientation.w = quaternion[3]
16

```

Ahol a `quaternion_to_euler(roll, pitch, yaw)` függvény a fentebb bemutatott számításokat végzi el.

2.2.4. Eredmények

A projekt tesztelése során több problémával szembesültem. Az egyik egy késleltetési hiba volt, amely jelentősen rontotta a program teljesítményét. Egy másik probléma a Hololens 2 kéz detektáló algoritmusával függött össze. Az emberi kéz korlátozott rotációs képessége miatt a forgatás fel van skálázva, így az objektum érzékeny az ilyen jellegű mozdulatokra. Ha a felhasználó túl gyorsan próbálja forgatni a drónt, az nem biztos, hogy képes követni a mozdulatot, és így könnyen kipördülhet a pályáról.

Tesztelés során kezdetben nagyon instabil volt a rendszer. A célpont (kocka) elmozgatása után a drón csak később, nagyjából 3 másodperc múlva reagált. Ez nem ideális, hiszen így nem érvényesül a rendszer valós idejű tulajdonsága. Először arra gondoltam, hogy a probléma az új eszköz bevezetéséből adódó hálózati hiba lehet, hiszen a reptetés kontrollerrel késleltetés nélkül működött. A rendszert hibalokalizációs módszerrel teszteltem, egyesével, részletekben mértem az egyes szakaszok futási idejét. Az eredmény az lett, hogy a hiba a feltételezéssel ellentétben mégse az AR szemüveg és a számítógép közötti kapcsolatból adódott. Mivel tovább nem tudtam szűkíteni a hibalehetőségeket, ezért más irányból közelítettem meg a problémát. Észrevettem, hogy a tesztelés során a kiíratásnál a késleltetési idő inicializáláskor még rendben volt, viszont idővel mindig nagyjából 4 másodperchez konvergált. Ebből arra következtettem, hogy az átvitel rendben van, viszont valami csökkent a üzenetek közlekedésének sebességét. Rájöttem, hogy a hálózatban torlódás lép fel, a hálózati csomagok nem jutnak el időben a célponthoz. Ezt egy hálózati paraméter okozta, amelyet a megváltozott körülményekhez kellett hangolni. Ez a paraméter a `rotrate`, mely szabályozza azt az intervallumot, amely időközönként a ROS rendszer meghívja az előre definiált függvényeket.

A paraméter átállítása után is végeztem méréseket, de így is túl magas értékeket kaptam. Ez azért történt, mert a számítógép belső órája nem volt szinkronizálva. Szinkronizálás után már elfogadható, 0.1 másodperc alatti késleltetést kaptam, ami az emberi reakcióidő tartományába esik.

A fenti módosításokkal mért eredményeket a következő táblázat foglalja össze:

A fent leírt rendszer segítségével a felhasználó képes irányítani egy tenyér méretű drónt milliméter pontossággal az OptiTrack által lefedett térrészen kézmozdulatokkal, valós időben. Ahogy már említettem a drón forgatásra a vártnál intenzívebben reagál bizonyos szögek esetében. Ezt ki lehetne küszöbölni a drón forgási sebességének lekorlátozásával vagy egy másik irányítási módszer kifejlesztésével.

	Átlagos késleltetés [ms]
Paraméter módosítása és szinkronizáció nélkül	4167
Paraméterrel, szinkronizáció nélkül	1423
Paraméterrel és szinkronizációval	96

2.1. táblázat. Tesztek eredményei

A Hololens 2 és az irányítást végző számítógép közötti kommunikációt sikerült megvalósítani az egyik irányban, azonban a számítógépről küldött adatokat a szemüveg nem fogadta. Ha sikerülne az adatátvitelt a másik irányban is megvalósítani, a drón valós fizikai helyzetét lehetne továbbítani, és erre ráilleszteni egy drón modellt, amely megjelenne az eszköz kijelzőjén. Ezzel felhasználóbarátabb lenne az alkalmazás, ugyanis jelenleg a vezérlésre szánt objektum ráilleszkedik a drónra, ezzel részben kitakarva azt.

3. fejezet

Kézfelismerés alapú irányítás és virtuális környezet implementálása

3.1. Megoldási lehetőségek ismertetése és megvalósíthatóságuk értékelése

Az eddig ismertetett rendszer az alapvető elvárásokat teljesíti, viszont működése jobbra tehető bizonyos kiegészítésekkel. Felmerül a hálózati kommunikáció megbízhatósága, illetve hatékonysága. Mint azt tudjuk az UDP protokoll nem használ nyugtázást, így nem lehetünk biztosak a hálózati csomagok megfelelő ütemben való beérkezésében. A TCP ezzel ellentétben tartalmaz megerősítő üzeneteket (Acknowledgement Number), viszont ezért lassabb a kommunikáció.

Erre a problémára megoldást a ROS alapú kommunikáció jelenthet, hiszen a drón vezérlő rendszer is ezt használja. Ehhez többféle implementációt is használhatunk. Ilyenek a ROS#, a ROS-TCP Endpoint könyvtár és a ROS.NET. Jelen dolgozatban a ROS-TCP Endpoint könyvtárat használtam, hiszen kis méretű adatok hálózaton való átküldése esetén ez a leggyorsabb, ahogyan azt az irodalomkutatásban is említettem.

Problémát jelentett továbbá az eddigi rendszer működésében a hirtelen mozdulatokra való érzékenység. A bemeneti gesztusok változtatása ezt megoldhatja a mozdulatok tovább bontásával. Ezen módszer jóval intuitívabb, a pilótának nem szükséges egy külön irányító eszközre figyelni, folyamatos közvetlen rálátása van a drónra.

További kiegészítésként létrehozhatunk egy virtuális modellt épületekről a valóságot reprezentálva. Ez lehetővé teszi, hogy a kívánt környezetet szimuláljuk lekicsinyített formában, így a pilótának könnyítve a feladatát.

3.2. Megvalósítási folyamat

3.2.1. UDP szerver lecserélése ROS-TCP alapú kommunikációra

Az eddigi implementációban egy UDP szerver segítségével működött a HoloLens 2 és a célszámítógép közötti kapcsolat. A küldött csomagokban stringeket használtam az üzenetek kódolásához. A következő lépésben ezt átalakítottam és bevezetésre kerültek a ROS-TCP Endpoint [17] és a ROS-TCP Connector [18] könyvtárak. Az első a kliensen futtatható, a második pedig a szervernek felel meg. Ezeket megfelelően beállítva gyors és megbízható kommunikáció hozható létre. Mivel így a szemüveg Wi-Fi-n keresztül közvetlenül a ROS csomagokat továbbítja, a szerveren elég csak létrehozni egy, a csomagok fogadására képes csomópontot. Ekkor az adatok objektumszerűen (JSON formátumban) feldolgozhatóak minden további nélkül.

Az új kommunikációs rendszer integrálásához a következő lépéseket kellett megtegyem. A már meglévő Unity projekthez hozzá kellett adni a ROS-TCP Endpoint könyvtárat és ez után beállítani a megfelelő kommunikációs paramétereket. Ilyenek voltak a ROS protokoll kiválasztása, illetve a késleltetési időintervallum (sleep time) beállítása. ROS protokollnak ROS2-t választottam, hiszen a drónvezérlő rendszer is ezt használja. A késleltetéssel először nem foglalkoztam, de tesztelés közben azt tapasztaltam, hogy a csomagok nagyon lassan érkeznek meg, így ezen paraméter csökkentésével a hibát kiküszöböltem. Természetesen itt is be kellett állítani a számítógép statikus IP címét, hogy a hálózati kapcsolat létrejöhessen. Az üzenet és szolgáltatás struktúrákat legenerálva a HoloLens kliens üzemképes.

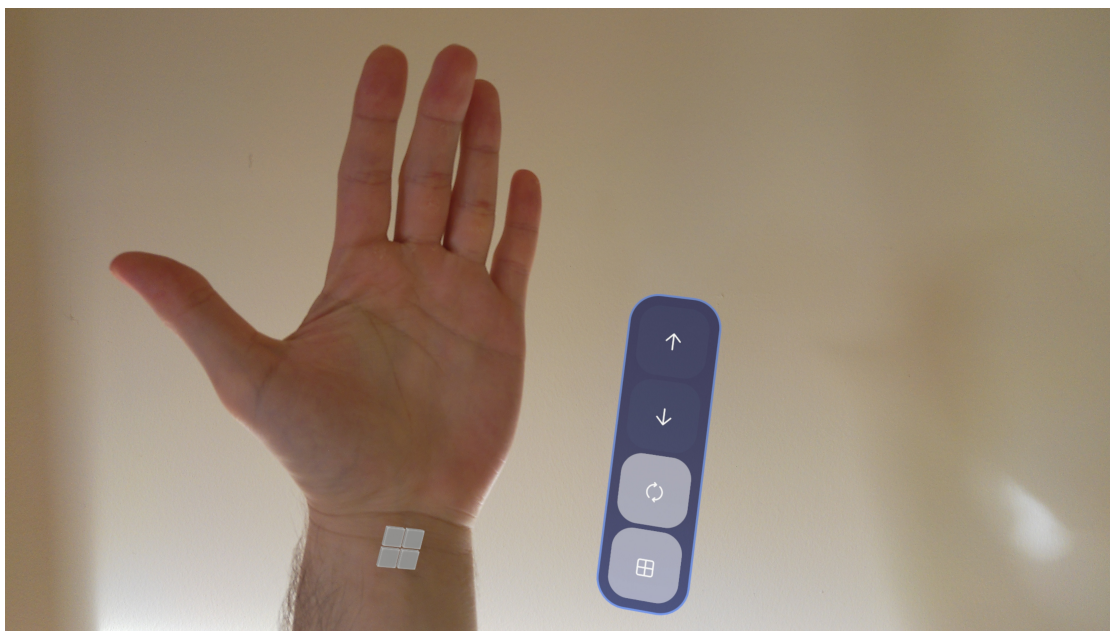
A szerver oldalon a ROS-TCP Connector könyvtárat kellett felhasználnom. Mivel a projekt konténerizált, az új fájlokat hozzá kellett adjam a rendszerhez. A ROS csomagok értelmezéséhez a szervernek is ismernie kell az üzenet struktúráját, így ezt le kellett generálnom. Ezt a megfelelő üzenet deklarálásával oldottam meg a megfelelő CMakeLists.txt fájlban. A továbbiakban létrehoztam egy új csomópontként az indítófájlban (launch file) a szolgáltatást és specifikáltam a programon belül, hogy hallgassa a rendszer a megfelelő topic-ot. Az érkező adatok feldolgozása így már problémamentesen megtörténhetett.

3.2.2. Drón irányítása kézmozdulatok felismerésével

A következőkben arra fókuszáltam, hogy egy egyszerűbb, de mégis stabilabb, irányítási módot implementáljak a hibalehetőségek minimalizálásával. Próbáltam a lehető legegyszerűbb megoldást megtalálni, ami végül a kéz közvetlen lekövetése lett. Az irányító mozdulatokat a következőképpen definiáltam:

- **Vízszintes irányú mozgás (roll, pitch):** Vizsgáljuk a kézfej a vízszintessel bezárt szögét és ennek megfelelően mozgatjuk el az adott irányba a drónt a szöggel arányos sebességgel.
- **Függőleges irányú mozgás (throttle):** A kézfej egy beállított referenciaponttól való távolsága befolyásolja ezen irányú mozgást, azaz ha a pont alatt van a kézfej, lefele, míg ha felette, akkor felfele mozdul el a drón a távolsággal arányos sebességgel.
- **Biztonság szempontjából** ha a másik kézfejünk is bekerül a kamerák látómezejébe a drón tartja a legutolsó pozícióját. Ekkor az irányítás nem működik, viszont beállíthatjuk az előzőleg említett referenciapontot az irányító kéz tetszőleges helyre való mozgatásával.
- Fontos, hogy a pilóta milyen irányba néz, hiszen összezavaró lehet, ha az irányok felcserélődnek ennek eredményeképp. Ennek kiküszöbölésére a program figyelembe veszi ezt és folyamatosan frissíti a saját koordináta rendszerét, azaz tetszőleges irányba nézhetünk, az irányítás nem fog változni a pilóta szemszögéből.
- **Függőleges tengely körüli forgást (yaw)** nem definiáltam, ugyanis ez jelen projekt szempontjából nem releváns. Azt azonban érdemes megjegyezni, hogy ezt kétféleképpen is lehetne implementálni. Egyik lehetőség, hogy a drón a pilóta irányítottságát kövesse, tehát folyamatosan arra nézzen amerre a pilóta néz. Így már nem szükséges foglalkozni irányítás közben a forgatással, viszont megszűnik a precíz orientáció beállításának lehetősége. A másik megoldás az lenne, hogy a függőleges irányú mozgatáshoz hasonlóan beállíthassunk egy referencia szöget a kezünknek, és a kézfejünket jobbra illetve balra elmozgatva lehetne forgatni a drónt.
- A program továbbá tartalmaz egy **panelt**, melyen a **felszálláshoz és leszálláshoz** szükséges gombok találhatóak. Ez a panel a biztonsági kéz mellett található, azaz ha fel- vagy le szeretnénk szállni a drónnal, ezt csak a drón landolt vagy lebegő

állapotában tehetjük meg. Ezen kívül két másik gomb található a panelon. Az egyikkel válthatunk a fentebb kifejtett irányítási mód és az eredeti között ("fogd és vidd" módszer, amivel precízen manipulálhatjuk a drón pozícióját). A másik gomb pedig a teret lekorlátozó virtuális kerítés ki/be kapcsolásáért felelős. Segítségével az adott térből akkor sem juthat ki a drón, ha esetleg a pilóta elvesztené a szemét elől.



3.1. ábra. Kézi menü (panel)

A kézmozdulatokkal történő irányítás számos előnnyel jár a hagyományos joystick-kontrollerhez képest, különösen a felhasználói élmény és a precizitás szempontjából. Mivel a drón közvetlenül a kéz mozgására reagál, a pilóta nem egy közvetett vezérlő eszközt használ, hanem közvetlen kapcsolatba lép a drónnal, ami egy magával ragadó élményt nyújt. Ennek köszönhetően csökken a megosztott figyelem igénye, hiszen a felhasználónak nem szükséges a joystick-ra vagy egyéb vezérlőeszközre koncentrálnia, a drón irányítása automatikusan követi a kéz mozgulatait. Ez a módszer különösen hasznos olyan helyzetekben, ahol gyors reakcióidő és nagyfokú pontosság szükséges, mivel a kézmozdulatokkal egyértelműen lehet utasítást adni a drónnak, nem kell két külön bemenet értékeit pontosan állítani, mint egy rádiós vezérlő esetében. Ez az irányítási mód lehetőséget ad arra is, hogy a felhasználó figyelmét teljes mértékben a drónra és a környezetre összpontosítsa. Az interaktív AR környezetben a kézmozdulatokkal való vezérlés egyszerre természetes és hatékony, ami elősegíti a drón precíz irányítását, megkönnyítve a komplex manőverek végrehajtását a virtuális és valós tér közötti integrációban.

3.2.3. Grafikus elemek: Valós tér modellezése

Az irányítás implementálása után elkezdtem foglalkozni a grafikus elemekkel is. Először is az eddig kockaként megjelenített pont helyett beillesztettem egy Crazyflie modellt [19] az applikációba, ami 1:1 arányban reprezentálja a laborban használt drónt. Ez valójában a fizikai drón virtuális ikerpárja. A másik felhasznált objektum egy benzinkút modellje [20] volt a Gazebo gyűjteményből, melyet egy valós épülethez képest tízszeresen leskáláztam [2.2 ábra]. A projekt célja egy olyan augmentált valóság létrehozása, amelyben bármilyen környezetet tudunk szimulálni laboratóriumi körülmények között kicsinyített formátumban.

A drón modellt glTF (Graphics Library Transmission Format) formátumban könnyen lehetett kezelni Unity-ben, viszont a Gazebo modell .dae (Digital Asset Exchange) formátuma nem tartalmazta a színeket. A problémát Blender használatával oldottam meg, itt a mellékelt textúrákat hozzá lehetett rendelni a modellhez majd ebből glTF fájlt generálva már megjelentek a színek is a HoloLens applikáción belül.

A valóság reprezentálásához a modellekhez hozzá kellett rendelni bizonyos fizikát implementáló komponenseket. Ahhoz, hogy ezek megfelelően viselkedjenek, a benzinkút modellt statikus objektumként kell kezelni, hiszen ez nem mozog a valóságban sem. Ezzel ellentétben a drón nem statikus, így működni fog az ütközésük. A benzinkútra hálózati, míg a drónra téglatest alakú ütközés figyelő komponenst (collider) helyeztem. A drón modellt ezen kívül merev testként kellett definiálni (rigid body component) és megbizonyosodni arról, hogy nem használok hozzá gravitációt, illetve, hogy nem kinematikus testként értelmezi a program. Az ütközés detektálást folyamatosan kellett állítani, mivel ha diszkrét lenne, nem mindig működne helyesen. Ezen beállítások után az ütközés már helyesen működött, viszont egyes esetekben furcsa jelenséget figyelhettem meg. Ha túl sokáig tartott az ütközés a drón kapott egy ellentétes irányú sebességet és elkezdett sodródni az adott irányban. Rájöttem, hogy ez annak volt a következménye, hogy ütközéskor a drón forgási sebességet is kapott a nem tökéletes ütközés miatt. Ezt azzal tudtam kiküszöbölni, hogy a merev test komponens megszorítások részén befagyasztottam a tengelyek körüli forgathatóságot. Ezek után már helyesen működött a szimuláció.

A kitűzött célt sikerült elérni: létrehoztam egy olyan virtuális környezetet, amiben kezünk segítségével irányíthatunk egy virtuális drón modellt, ami nem ütközik a környezetében található objektumokkal. Mivel a drón modellt a hálózati kommunikáció segítsé-

gével valós időben leköveti a fizikai drón, így elkerülhető a drón ütközése az előre definiált tetszőleges épülettel. Természetesen mivel a laboratóriumban csak a kamerák által határolt részen dolgozhattam, így a szimulációba definiálnom kellett láthatatlan falakat is, amik nem engedik kirepülni a drónt ebből a térből.

3.3. A drón kameraképének feldolgozása

A Crazyflie drón egy moduláris platform, számos egyéb eszközzel felszerelhető. Ilyen például az AI deck¹ modul is, amely egy erőteljes, kisméretű feldolgozó egységet biztosít a drón számára. Az AI-Deck a kis energiaigényű HIMAX kamerája (320x320px) mellett tartalmaz egy GAP8 nevű mikroprocesszort is. Ez neurális hálózatok futtatására optimalizált, így lehetővé teszi a valós idejű képfeldolgozást közvetlenül a drónon, anélkül, hogy további számítási kapacitásra lenne szükség. Az AI-Deck Wi-Fi kapcsolaton keresztül hálózati végponti egységekkel való kommunikációra is képes. Mivel az eddig létrehozott interfész nem rendelkezik semmiféle visszacsatolással, azaz a pilóta nem kap visszajelzést a dróntól, rendkívül hasznos lenne visszajelzésként kameraképet eljuttatni a felhasználóhoz. A későbbiekben a képfeldolgozó egység is hasznos lehet a projekt szempontjából.

3.3.1. AI deck képfeldolgozás

Az AI-Deck modul megismerése során felfedeztem annak hozzáférési és vezérlési lehetőségeit. Az eszköz egy átalakító segítségével JTAG csatlakozón keresztül érhető el, de ugyanakkor rádión keresztül is lehetséges a kommunikáció az eszközzel, ami kényelmesebb megoldást nyújt fejlesztési folyamatok során. Az eszköz programozása C nyelven lehetséges, ami ugyan biztosítja a hatékony és erőforrástakarékos futtatást, viszont sok esetben nem egy egyszerű feladat. A munka során több példa programot [21] is kipróbáltam, melyek közül a klasszifikáló és az arcfelismerésre használható projektek voltak a legérdekesebbek. Ezek egy konvolúciós neurális hálózatot használnak és a MobileNetV2 tanított modellt veszik alapul. Míg a projektekből rendkívül sokat tanulhattam, a felismerési pontosság és stabilitás nem volt elég megbízható. Ezért végül úgy döntöttem, hogy képfeldolgozási feladatokat a drónt irányító számítógépen futtatom, ahol nagyobb számítási kapacitás áll rendelkezésre, így futtatható akármilyen neurális hálózat, és a programozási nyelvre sincs megkötés.

Az AI-Deck modulra fejlesztett Wi-Fi kommunikációs példa projekt lehetőséget ad

¹<https://www.bitcraze.io/products/ai-deck/>

az élő képátvitelre, viszont ez csak 5 képet visz át másodpercenként. Ez a sebesség nem elegendő a valós idejű képfeldolgozási célokra, így módosítottam a kódot, hogy növeljem a sebességet. A képfeldolgozáshoz szükséges átviteli sebességet végül úgy sikerült 15 fps-re növelni, hogy a képeket kisebb felbontásban továbbítom. Az eredeti kép minősége (244x324px a példakód alapján) ugyan kis mértékben csökkent (120x160px), de mivel eleve kis felbontású volt a kimenet, ez a módosítás nem okozott jelentős képminőségromlást. Az optimalizált Wi-Fi streamelés eredményeképpen sikerült gyorsan és hatékonyan adatokat továbbítani az AI-Deck-ről a földi egység felé, így támogatva megbízhatóbb képfeldolgozási eredményt.

A következő lépésben a fogadott képek feldolgozásához megismertem és a gyakorlatban is kipróbáltam a YOLO² (You only look once) objektum detektáló neurális hálózatot. Ehhez megfelelő mennyiségű tanuló adatot kellett készítenem. A képeket a drónra szerelt kamera segítségével készítettem el, melyhez írtam egy egyszerű szkriptet a felvételek lementéséhez. Ezzel a módszerrel közel kétezer képet készítettem el, majd ezeket annotáltam a Roboflow³ online szolgáltatása segítségével. Az annotálás és az adatbővítési (data augmentation) lépés után keletkezett két tanítóhalmazom, ezek 2578 és 3500 képből álltak. A képeket 75-25%-os arányban bontva tanító és validálási halmazokra, elkezdhettem a tanítási folyamatot. Ehhez a YOLOv5⁴ modellt tanítottam tovább (transfer learning), mivel ez a legelterjedtebb már előtanított modell, de próbálkoztam YOLOv8⁵-al is. A továbbtanítás sikeres volt, 83%-os pontossággal ismerte fel az célobjektumot a modell annak ellenére, hogy a képek felbontását csökkentettem az adatátvitel sebességének növelése miatt.

Annak ellenére, hogy a képfeldolgozó algoritmusok eredményesen működtek, belefutottam a Crazyflie drón egy olyan problémájába, amit az eszköz fejlesztői (Bitcraze) a mai napig nem tudtak megoldani. A drón nem képes egyszerre repülni és Wi-Fi segítségével képet továbbítani, azaz stabilan működtetni az AI deck modult. Több próbálkozási kísérletem volt a probléma elhárítására, de egyik sem bizonyult működőképesnek. Először arra gondoltam, hogy interferencia léphet fel a drón rádiós adója és az AI deck Wi-Fi modulja között, mivel közel azonos frekvencián működnek. Ezért alufóliát helyeztem a két elem közé blokkolva ezzel a hullámokat, de a képátvitel így is megszakadt felszálláskor.

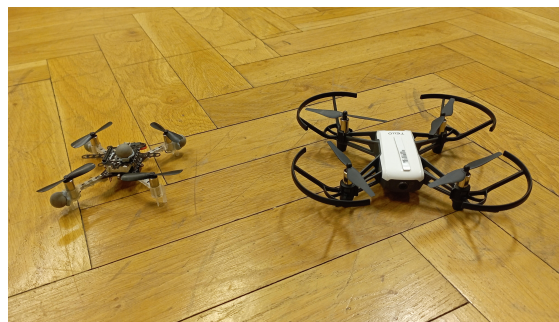
²<https://docs.ultralytics.com/>

³<https://roboflow.com/>

⁴<https://github.com/ultralytics/yolov5>

⁵<https://github.com/ultralytics/ultralytics>

Az AI deck Wi-Fi-t használó C nyelven megírt kommunikációs kódjában is próbáltam módosítani a megfelelő részeket, de így sem jártam sikerrel. Megpróbáltam felkeresni a gyártót, de nem kaptam kielégítő választ, tulajdonképpen megerősítést nyert, hogy ez egy ismert hiba. Végül a megoldás abban rejlett, hogy egyszerűen áttértem egy másik drón használatára (DJI Tello⁶ [3.2]) amely sokkal robusztusabb rendszerének köszönhetően minden további nélkül képes repülni és képet továbbítani. A drón vezérelhető egy nyílt forráskódú API (Application Programming Interface) segítségével. Ennek köszönhetően a jövőben az előbb említett funkciók felhasználhatóak lesznek további projektekhez is.

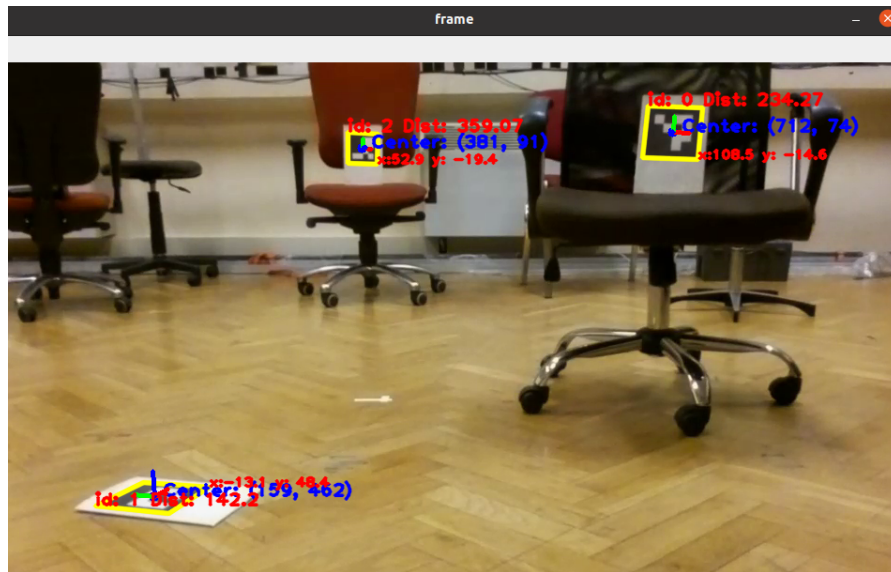


3.2. ábra. Crazyflie 2.1 és DJI Tello drónok

Az új kamerával egy másik projektet is kipróbáltam. A feladat egy ArUco marker detektálása és távolságának meghatározása volt a kamerától számítva. Az ArUco⁷ marker egy speciális, QR-kódhoz hasonló jelölő, amelyet gyakran használnak objektumok azonosítására és követésére. Ezek a jelölők egyedi azonosítók, melyeket Python könyvtárak segítségével lehet felismerni. A felismeréshez használt projektben [22] össze kellett kötni a drónon található kamera adatcsatornáját a felismerő algoritmussal, illetve be kellett állítani a megfelelő ArUco marker méretet. A detektálás és távolság meghatározás az eddigi próbálkozásaimhoz képest rendkívül jól működött. A távolságbecslés hibája közelre milliméteres tartományba esett, míg távolra is csupán néhány centiméter eltérés volt a kis felbontású kamerán is.

⁶<https://www.ryzerobotics.com/tello/>

⁷https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html



3.3. ábra. ArUco jelölő detekció

3.3.2. HoloLens kétirányú kommunikációjának lehetősége

További problémát jelentett az is, hogy bárhogy próbáltam, a HoloLens nem fogadott semmilyen hálózati csomagot. Emiatt nem lenne lehetséges visszacsatolási funkciókat fejleszteni az interfészhez sem. A megoldás megtalálását továbbá az is nehezíti, hogy fejlesztés során nincs mód a programban hibakeresésre, mivel ezt az eszköz nem támogatja. Alapos kutatás után találtam egy megoldási irányt, ahol a Windows stream socket könyvtárát felhasználva lehetséges karaktereket küldeni a HoloLens eszközre, így a jövőben ez rengeteg új lehetőséget nyit meg.

4. fejezet

Eredmények

4.1. A rendszer tesztelése

A rendszer fizikai tesztelését a funkciók kipróbálásával kezdtem. A kézi menü jól működik, csak akkor jelenik meg amikor a felhasználó maga felé fordítja a tenyerét. A panel ekkor sem foglal el túl nagy területet, nem zavaró a jelenléte, ugyanakkor a gombok könnyen hozzáférhetőek, a kezelés egyszerű és letisztult. A drón kézmozdulatokkal való irányítása számomra egyszerű volt, nagyon könnyen tudtam alkalmazkodni hozzá. Érdekes lenne a projekt tesztelését független felhasználókkal is elvégezni, mivel érdekes eredmények születhetnek. Ez lehetőséget adna arra, hogy következtetéseket vonjunk le arra vonatkozóan, mennyi időre van szükségük a vezérlés elsajátításához különböző tapasztalati szintek mellett. Emellett új fejlesztési irányok is kirajzolódhatnak a visszajelzések alapján.

Ezen az irányvonalon elindulva a 2024-es Kutatók Éjszakáján [23] a HUN-REN SZTAKI-ban a projekt egy fejletlenebb verziója került bemutatásra. A résztvevők kipróbálhatták a kevert valóságban megjelenő drónt, kezükkel irányíthatták azt a benzinkút modell körül. A tapasztalatok azt mutatták, hogy pár próbálkozás után a gyerekek is tudták vezetni a virtuális drónt. Kiegészítési lehetőségként merültek fel a biztonsági falak alkalmazása, illetve egy szimuláció újraindító gomb integrálása. Mivel ezek az elemek jelentősen hozzájárulnak a felhasználói élmény javításához és a biztonságosabb drónvezérlés biztosításához, ezért beépítésre kerültek az interfészbe. A fel és leszállás a bemutató során még a számítógépről volt vezérelhető, ez a funkció is jelenleg már az kezelőfelület része.

A tesztelés következő fázisa a program késleltetésének lemérése. Ehhez a pontosság érdekében foglalkoztam számítógép órák szinkronizációjával is. Abban az esetben, ha a program működéséhez csak Linux operációs rendszert használó számítógép kellene, akkor

ez egy nagyon egyszerű folyamat lenne. Viszont a rendszerben három számítógép található, ebből egy Linux, egy Windows 10 és egy Windows alapú, a HoloLensre fejlesztett operációs rendszer működik. Mivel az OptiTrack kamerarendszer szoftvere kizárólag Windows alapon működik és az időbélyegeket a kamerák ehhez a géphez igazítják, szükség van egy szinkronizációs megoldásra. A Windows rendszereken nehéz óra egyeztetést végezni, ugyanis nem lehet egyszerűen hozzáférni ezekhez a beállításokhoz. Az alapértelmezett szinkronizációs folyamat Windowson az NTP (Network Time Protocol) szerverekkel való kommunikáció. Ezt a módszert használtam eddigi munkáim során a mérési eredményekhez, de ez 1-50 ms mérési hibához vezethet, hiszen a számítógépek nincsenek folyamatosan egymáshoz egyeztetve. A problémára egyetlen megoldási lehetőséget találtam, ez a PTP (Precision Time Protocol) alapú szinkronizáció.

4.1.1. PTP szinkronizációs eljárás

A PTP segítségével számítógépek órája hálózaton keresztül szinkronizálható nanoszekundumos pontossággal [24]. A pontos mérések elvégzésének érdekében ez lenne a legjobb módszer. Először a linuxptp csomaggal próbálkoztam, ezen belül a ptp4l¹ nevű implementációval. Ezzel nem sikerült létrehozni a számítógépek között kapcsolatot, így áttértem a PTPd-re. Ezzel a megoldással Linux operációs rendszert futtató számítógépeket sikeresen szinkronizáltam. A PTPd a mester-szolga architektúrát használja. Az először elindított példány viselkedik mesterként, a többi a hálózaton megtalálja a mestert és automatikusan alárendeli magát. Parancssorból is beállíthatóak a szerepek, viszont egy alhálózaton belül nem lehet több mester egyszerre. A protokoll szerint a hálózaton a résztvevő gépek másodpercenként értesítik egymást és frissítik az órájukat.

Windows 10 rendszereken ez a megoldás nem működik egyből. Ahhoz, hogy egyáltalán a számítógép fogadja a PTP üzeneteket, a rendszerleíró adatbázisában (registry) módosítani kell a megfelelő bejegyzést a funkció bekapcsolásához. Ezen kívül létre kell hozni tűzfal szabályokat a megfelelő portokhoz, hogy a hálózati kommunikációt fogadja a rendszer. Az időkritikus eseményüzenetek (Sync, Delay_req, Pdelay_Req és Pdelay_Resp) a 319-es portot használják, míg az általános üzenetek (Announce, Follow_Up, Delay_Resp, Pdelay_Resp_Follow_Up, management and signaling) a 320-ast [25]. Az ezekhez kapcsolódó megfelelő beállítások elvégzése után automatikusan szinkronizál a rendszer, de csak abban az esetben, ha unicast módon küldjük neki a szinkronizációs üzeneteket. Mivel az elvégezni kívánt méréshez több számítógépet kell összekapcsolni,

¹<https://manpages.debian.org/unstable/linuxptp/ptp4l.8.en.html>

ezért ez a megoldás nem kielégítő.

Felmerült megoldásként a virtuális számítógépek alkalmazása. Ki kell nevezni a Windows 10 operációs rendszert futtató gépet "host"-nak, amin egy virtuális Linuxot kell futtatni. Ezt én az Oracle VirtualBox² virtualizációs szoftver segítségével oldottam meg, mely a szimulált rendszertől érkező hálózati csomagokat is képes tovább irányítani a gazda gép fizikai hálózati kártyájára. Ezek után a "guest" operációs rendszeren lehetséges a PTPd elindítása mesterként, így már akárhány számítógép óráját lehet szinkronizálni a virtuális számítógéphez. Sajnos a szimulált és a gazda számítógép között a szoftver nem oldja meg a szinkronizációt. Ez a megoldási módszer valószínűleg kudarcra van ítélve a Windows process kezelési prioritizálási folyamatai miatt. Így egyedüli megoldásként marad az NTP szerverekhez való manuális szinkronizáció.

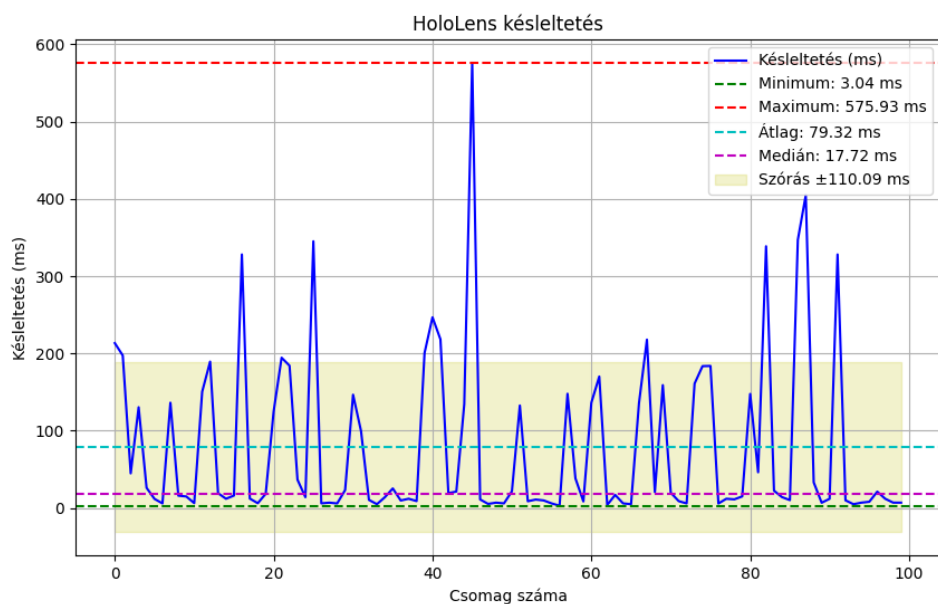
4.1.2. Késleltetési eredmények kiértékelése

A késleltetés le méréséhez végül azt találtam ki, hogy mivel végül sikerült adatot küldeni a HoloLens-re, így lehetőségessé vált a megfordulási idő (Round Trip Time) mérése. A teljes kommunikációs szekvencia három lépésből áll:

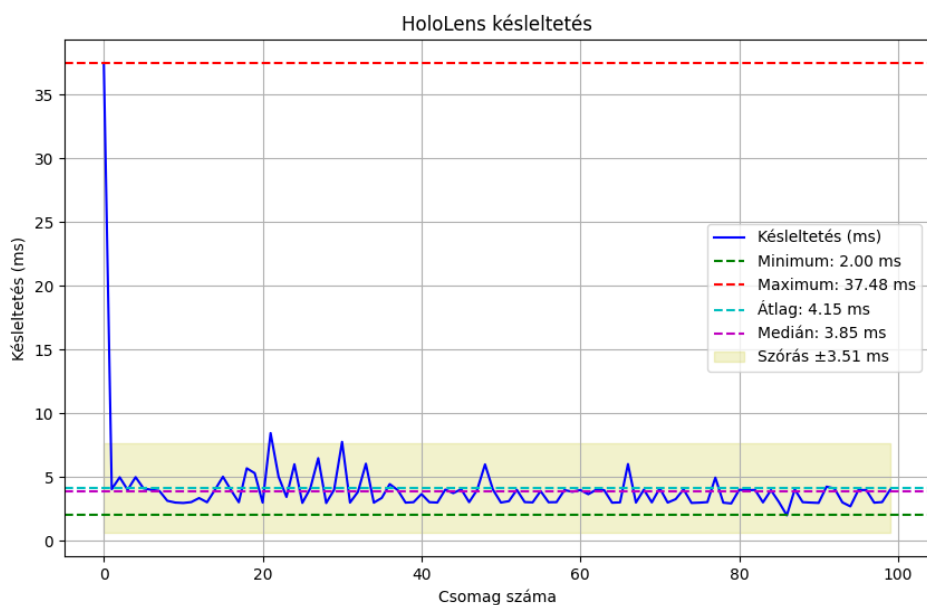
1. A HoloLens elküldi a koordinátát a szervernek.
2. A szerver feldolgozza a kapott adatokat.
3. A szerver a rádió segítségével kiküldi a megfelelő CRTP (Crazy RealTime Protocol) parancsot a drónnak.

Az első intervallum le méréséhez randomizált (pozíció, rotáció) párosokat küldtem a szemüvegnek, ami amint megkapta vissza is küldte azt. A mérést két összeállításban is elvégeztem, az egyiknél a telefonom hálózat megosztási funkcióját használtam, a másik esetben pedig egy nagy sávszélességű (1000 MBit/s) útválasztót. Az ebből le mért időt megfeleltetve megkapjuk a keresett késleltetés első részét. A mérés során száz adatot küldtem, ezekből átlagot és szórást számoltam, melyeket a következő grafikonok szemléltetnek:

²<https://www.virtualbox.org/>



(a) Telefon használatával

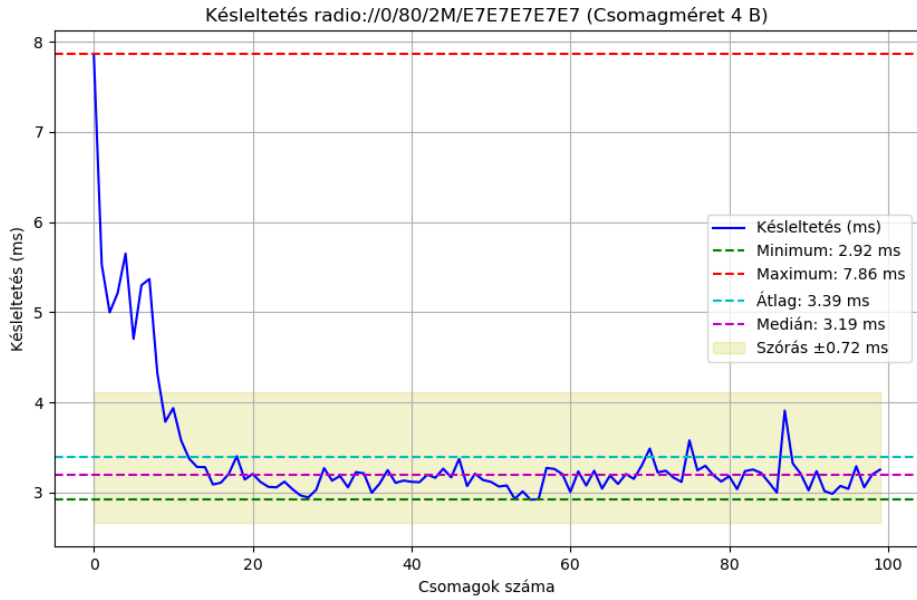


(b) Útválasztó használatával

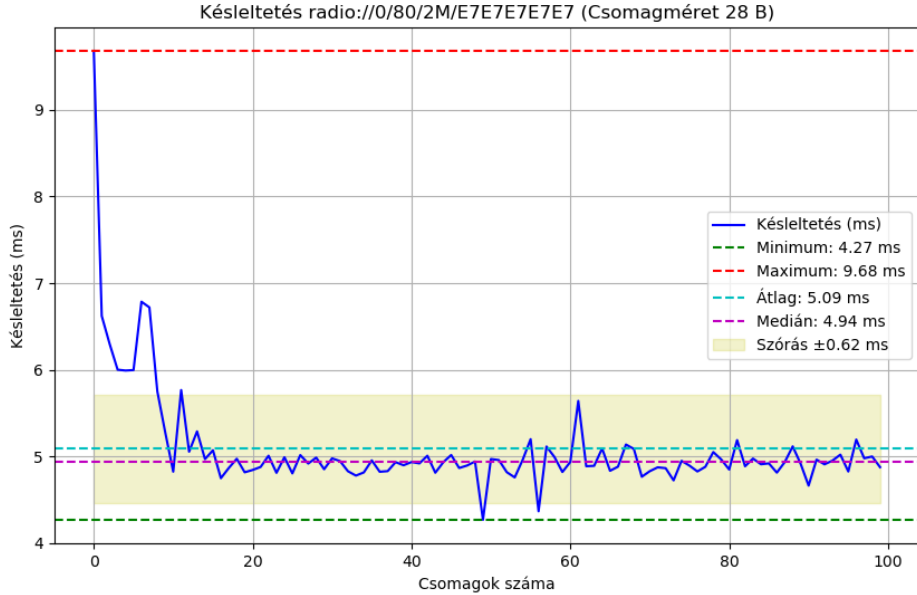
4.1. ábra. HoloLens hálózati csomag megfordulási idő

A számításokból kiderül, hogy nagyjából 2 ms szükséges, hogy a koordináták eljussanak a számítógéphez. Ez után a szerver fel kell dolgozza a kapott adatot, amely elhanyagolható időt vesz igénybe, hiszen ekkor a ROS kommunikáció is egyazon számítógépen zajlik. A ROS csomagok mérési eredményeim alapján 1 ns alatt megérkeznek.

Utolsó lépésként a szerver CRTP parancsot küld a drónnak. Ezt szintén lemértem, a megfordulási idő az alábbi ábrákon látható:



(a) 4 bájt adat esetében



(b) 28 bájt adat esetében

4.2. ábra. CRTP megfordulási idő

A mérési eredmények alapján a megfordulási idő fele nagyjából 2 ms, ami összhangban áll mások méréseivel [26].

Ezekből az következik, hogy nagy sávszélességű útválasztót használva, a legrosszabb esetben is a 0.1 szekundumról a rendszer késleltetése 4 ms alá lett csökkentve. Ez nagyságrendekkel kisebb az emberi reakcióidőnél, tehát a drón irányíthatósága az emberi reakcióidőn kívül nagyrészt az eszköz fizikai tehetetlenségéből adódik.

4.2. Továbbfejlesztési lehetőségek

A jelenleg elkészített rendszer önmagában kielégítően működik, viszont vannak lehetőségek a továbbfejlesztésre. Mivel a munka folyamán sikerült adatot küldeni a HoloLens szemüvegnek, így fontos lenne létrehozni egy visszajelző rendszert a pilóta számára. Két kiegészítő funkció lehet szükséges, az egyik egy kamerakép, melynek segítségével figyelemmel lehetne kísérni a drón közvetlen környezetét. A másik lehetőség a fizikai drón helyzetének számszerűsítése, például a felhasználótól való távolság megjelenítése. A továbbfejlesztett rendszert tesztelni kellene emberek csoportján kezelhetőség szempontjából. Meg lehetne figyelni, hogy mennyivel nehezebb egy valós drónt irányítani kézmozdulatokkal, mint csak a virtuális modellt.

Ezekon kívül lehetséges más virtuális modellek beillesztése is a környezetbe. Egy másik fejlesztési irány lehet a dinamikus feltérképezés implementálása, mellyel a drón mozgása során a kameraképeken futtatott 3D rekonstrukciós algoritmus által előállított modellt jeleníti meg a HoloLens. Ehhez a projektet kültéren is lehetne tesztelni az OptiTrack kamerarendszer nélkül, így bővítve a felhasználási lehetőségeket. Ekkor már figyelembe kellene venni a kültéri időjárási viszonyokat is, ezért sokkal megbízhatóbb drónt kellene felhasználni hozzá.

5. fejezet

Összefoglalás

A projekt célja egy olyan intuitív, kiterjesztett valóság alapú interfész fejlesztése volt, amely lehetővé teszi a HoloLens 2 AR szemüveg segítségével egy drón vezérlését pusztán kézmozdulatokkal. A munkához egy Crazyflie drónt használtam fel, melyet az OptiTrack kamerarendszer segítségével reptettem beltéren milliméter pontossággal. A kész interfésszel két módon vezérelhetjük a kvadrokoptert, adott célpontot kijelölve, illetve folytonosan a kézmozdulatok lekövetésével. Az új vezérlési módszer intuitívabbá teszi a térben való manőverezést, illetve kiküszöböli a figyelem megosztásának problémáját repülés közben.

5.1. Elvégzett feladatok összegzése

A munkafolyamat során először létrehoztam az interfész egy egyszerűbb verzióját, ez még nem tartalmazott sok elemet. Egy virtuális kockát lehetett megfogni és elmozgatni a térben, amit a HoloLens projektált a felhasználó elé. UDP kommunikációt használva és a szemüveg koordináta rendszerét a globálishoz egyeztetve a kocka pozícióját eljuttattam a drónt vezérlő szerverig, így a drón képes volt lekövetni a kocka mozgását valós időben.

A projektet tovább fejlesztve lecseréltem az UDP kommunikációt ROS TCP üzenetekre, mely egy sokkal megbízhatóbb csatornát biztosított, nem csökkentve az átviteli sebességet. A virtuális kockát lecseréltem egy, a fizikai drónnak megfelelő modellre és definiáltam irányító kézmozdulatokat, mellyel a drón intuitívan irányítható. Implementáltam egy virtuális menüt is, amely tartalmazza a fel- és leszálláshoz szükséges gombokat. További funkciók a menüben a virtuális kerítés, illetve az irányítási mód váltása a régi és az új módszer között.

Az interfész kiegészítéseként foglalkoztam képfeldolgozással is. A Crazyflie drónhoz gyártott AI deck modult használtam fel a képek forrásaként, melyeket végül a drónról átküldve egy számítógépen dolgoztam fel a YOLO modellt újratanítva. A képfeldolgozás hatásfokának növeléséhez lecsökkentettem a képek minőségét a képátvitel sebességének növelése érdekében, így elértem 15 fps sebességet, ami már megfelelő volt. A képminőség csökkentése nem befolyásolta számottevően a neurális hálózat pontosságát. Kiderítettem, hogy a Crazyflie drónnal egy gyártói hiba miatt nem lehetséges egyszerre repülni és képet átvinni hálózaton keresztül, ezért egy másik drónra implementáltam a képfeldolgozást.

Végeredményben elmondható, hogy létrehoztam egy olyan ember-robot interfészt, amely megkönnyíti az olyan feladatokat, ahol a drónok reptetéséhez emberi beavatkozásra van szükség. Segítségével rendkívül precízen és biztonságosan vezérelhető a drón, a megfelelő beállítások mellett nem csapódik bele környező épületekbe sem.

5.2. Kitekintés: Jövőbeli tervek

A létrehozott interfészt a jövőben számos további funkcióval szeretném bővíteni, amelyek mind a felhasználói élmény fokozását, mind a rendszer funkcionalitásának javítását szolgálnák. Az egyik fontos fejlesztési irány egy részletes visszajelző felület beépítése lenne, amely lehetővé tenné, hogy a pilóta valós időben, átlátható módon kövesse a drón helyzetét és mozgását. Ezen a felületen megjeleníthető lenne a drón kameraképe, így a felhasználó pontos vizuális visszajelzést kapna a drón környezetéről. A távolban egy virtuális drónmodell jelezhetné a fizikai drón pontos pozícióját, megkönnyítve ezzel a drón útjának követését, különösen nagyobb távolságokon vagy komplex terepen.

A projekt bővítésének egy másik iránya a szabadtéri környezetben való alkalmazás lehetőségének megteremtése. Ezzel elhagyhatnám az OptiTrack kamerarendszer használatát, amelyet jelenleg a drón precíz lokalizációjához és pozicionálásához alkalmazok beltéren. Szabadtéren egy GPS-modullal rendelkező drónnal is végezhetnénk a teszteket, ami lehetővé tenné a nagyobb távolságú, szabadtéri manőverezést, ahol az AR-alapú vezérlés előnyei, mint a gyors és intuitív irányítás, még inkább érvényesülnének.

Jövőbeli terveim között szerepel az eddig tesztelt képfeldolgozási megoldások integrálása olyan projektekbe, ahol szükség lehet objektumok detektálására vagy távolság meghatározására drónok segítségével. Egyéb képfeldolgozási módszereket is szeretnék alkalmazni, amelyekkel lehetségessé válhat épületstruktúrák térbeli modellezése, robotok

autonóm navigációja vagy biztonsági zónák monitorozása. A továbbfejlesztés részeként a rendszer vezérlési mechanizmusait is érdemes lehet bővíteni. Például a kézmozdulatok felismerésének finomítása neurális hálózatok segítségével, vagy akár egyedi irányítási profilok létrehozása felhasználói szokások alapján. Így az interfész a felhasználó tapasztalatai szerint finomhangolható lenne, és az irányítás még gördülékenyebbé válna.

Irodalom

- [1] S. Gazdag, A. Kiskároly, T. Szirányi és A. Majdik, „Autonomous racing of micro air vehicles and their visual tracking within the Micro aerial vehicle and Motion capture (MIMO) arena“, *54th International Symposium on Robotics, ISR Europe 2022*, Berlin: VDE Verlag GmbH, 2022, 333–340. old. cím: <https://eprints.sztaki.hu/10466/>.
- [2] *OptiTrack documentation*, en. cím: <https://docs.optitrack.com/>.
- [3] *Robot Operating System documentation*, en. cím: <https://docs.ros.org/>.
- [4] M. Lopez, „The making of the hololens 2: How advanced AI built Microsoft’s vision for ubiquitous computing“, *Microsoft Conexiones*, 2019. cím: <https://blogs.microsoft.com/conexiones/2019/11/20/the-making-of-the-hololens-2-how-advanced-ai-built-microsofts-vision-for-ubiquitous-computing/>.
- [5] *HoloLens2 Specs*, en. cím: <https://vr-compare.com/headset/microsofthololens2>.
- [6] *Crazyswarm 2*, en. cím: <https://imrclab.github.io/crazyswarm2/index.html>.
- [7] G. Nagymáté és R. M. Kiss, „Application of OptiTrack Motion Capture Systems in Human Movement Analysis: A systematic Literature Review“, *Recent Innovations in Mechatronics*, 5. évf., 1. sz., 1–9. old., 2018. DOI: <https://doi.org/10.17667/riim.2018.1/13>.
- [8] K. Albert, „Kisméretű légi járművek hatékony vezérlése“, hu, Szakdolgozat, Budapesti Műszaki és Gazdaságtudományi Egyetem (BME), 2019.
- [9] J. Allspaw, G. LeMasurier és H. Yanco, „Comparing Performance Between Different Implementations of ROS for Unity“, *Proceedings of the HRI-23 Workshop on Virtual, Augmented, and Mixed Reality for Human-Robot Interaction*, Stockholm,

- Sweden: ACM, 2023, 1–6. old. DOI: <https://doi.org/10.1145/nnnnnnn.nnnnnnn>.
- [10] *Mixed Reality Toolkit 3*, en. cím: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/>.
 - [11] M. Allenspach, T. Kötter, R. Bähneemann, M. Tognon és R. Siegwart, „Design and Evaluation of a Mixed Reality-based Human-Robot Interface for Teleoperation of Omnidirectional Aerial Vehicles“, *ICUAS 2023 - International Conference on Unmanned Aircraft Systems*, Warsaw, Poland, 2023. jún., 1168–1174. old. DOI: [10.1109/ICUAS57906.2023.10156426](https://doi.org/10.1109/ICUAS57906.2023.10156426).
 - [12] K. Konstantoudakis, K. Christaki, D. Tsiakmakis és tsai., „Drone Control in AR: An Intuitive System for Single-Handed Gesture Control, Drone Tracking, and Contextualized Camera Feed Visualization in Augmented Reality“, *Drones*, 6. évf., 2. sz., 2022. DOI: <https://doi.org/10.3390/drones6020043>. cím: <https://www.mdpi.com/article/10.3390/drones6020043>.
 - [13] *Crazyflie 2.1*, en. cím: <https://www.bitcraze.io/products/old-products/crazyflie-2-1/>.
 - [14] J. A. Preiss*, W. Hönig*, G. S. Sukhatme és N. Ayanian, „Crazyswarm: A large nano-quadcopter swarm“, *IEEE International Conference on Robotics and Automation (ICRA)*, Software available at <https://github.com/USC-ACTLab/crazyswarm>, IEEE, 2017, 3299–3304. old. DOI: [10.1109/ICRA.2017.7989376](https://doi.org/10.1109/ICRA.2017.7989376). cím: <https://doi.org/10.1109/ICRA.2017.7989376>.
 - [15] *ROS migration*, en. cím: <https://docs.ros.org/en/galactic/The-ROS2-Project/Contributing/Migration-Guide.html>.
 - [16] Marlenaklein-Msft, „Hololens 2 fundamentals: Develop mixed reality applications - training“, *Training | Microsoft Learn*, accessed May 17, 2024. cím: <https://learn.microsoft.com/en-us/training/paths/beginner-hololens-2-tutorials/>.
 - [17] *ROS-TCP-Endpoint*, en. cím: <https://github.com/Unity-Technologies/ROS-TCP-Endpoint>.
 - [18] *ROS-TCP-Connector*, en. cím: <https://github.com/Unity-Technologies/ROS-TCP-Connector>.

- [19] bitcraze_se, *Crazyflie 2.X*, <https://sketchfab.com/3d-models/crazyflie-2x-7b3ef11ebced4bfdb91bd756bfae46a5>, Licensed under CC-BY-4.0, available at <http://creativecommons.org/licenses/by/4.0/>, 2022.
- [20] *Gazebo gas station*, en. cím: https://github.com/osrf/gazebo_models/tree/master/gas_station.
- [21] *AI deck examples*, en. cím: <https://github.com/bitcraze/aideck-gap8-examples>.
- [22] *Distance Estimation Using Opencv and ArUco Marker*, en. cím: <https://www.instructables.com/Distance-Estimation-Using-Opencv-and-ArUco-Marker/>.
- [23] *Kiterjesztett Valóságban Koordinált Drónok a Mikro repülőgép és gépi MOZgás-követés (MiMo) Arénában*, hu. cím: <https://app.kutatokejszakaja.hu/esemenyek/hun-ren-szamitastechnikai-es-automatizalasi-kutatoi-intezet-hun-ren-sztaki/kiterjesztett-valosagban-koordinalt-dronok-a-mikro-repulogep-es-gepi-mozgaskovetes-mimo-arenaban>.
- [24] *What is Precision Time Protocol?*, en. cím: <https://www.masterclock.com/network-timing-technology-ntp-vs-ptp.html#:~:text=PTP%2C%20or%20Precision%20Time%20Protocol,or%20even%20picosecond%20level%20synchronization..>
- [25] „IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems“, *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, 1–269. old., 2008. DOI: 10.1109/IEEESTD.2008.4579760.
- [26] *Crazyflie radio latency*, en. cím: <https://forum.bitcraze.io/viewtopic.php?t=1708>.