

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
SZÁMÍTÁSTECHNIKA SZAK**



SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM

**IPARI ROBOTOK TÁVVEZÉRLÉSE ANDROID
APPLIKÁCIÓVAL**

Országos Tudományos Diákköri Konferencia Dolgozat

Témavezetők:
Dr. Márton Lőrinc,
Egyetemi tanár
Dr. Szántó Zoltán,
Egyetemi adjunktus

Hallgató:
Deák Adrienn

2022

**SAPIENTIA HUNGARIAN UNIVERSITY OF
TRANSYLVANIA
FACULTY OF TECHNICAL AND HUMAN SCIENCES
COMPUTER SCIENCE SPECIALIZATION**



SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA

**REMOTE CONTROL OF INDUSTRIAL ROBOTS
WITH ANDROID APPLICATION**

Conference of Scientific Students Paper

Scientific advisors:
Dr. Márton Lőrinc,
Full Professor
Dr. Szántó Zoltán,
Lecturer

Student:
Deák Adrienn

2022

Kivonat

Napjainkban egyre nagyobb teret hódítanak az IoT (Internet of Things) technológia alapú rendszerek, amelyek segítségével hatékonyan megvalósítható az ipari folyamatok okos eszköz alkalmazással történő távvezérlése, felügyelete valamint irányítása. Az okos eszközökön futó ipari szoftverek esetén kritikus az ipari folyamat megbízható monitorizálása, a gyors beavatkozás biztosítása, valamint a jogosultság megfelelő kezelése. Ezért az ilyen jellegű alkalmazások fejlesztése egy összetett folyamat.

Az ipari robotok rendelkeznek saját vezérlőeszközökkel, melyek segítségével programozhatóak, felügyelhetőek, de ezek a vezérlők általában helyhez kötöttek. Éppen ezért, ezen vezérlőeszközök mellé, szinte tökéletes kiegészítő eszközként szolgálhatnak a telefonos-alkalmazások.

Jelen dolgozatom célja egy olyan Android applikáció fejlesztése, mely az iparban használt robotok távvezérlésére alkalmazható. A fejlesztett alkalmazás képes a robot mozgásának felügyeletére videó és érzékelő információk alapján, képes parancsokat, parancssorozatokat küldeni a robotnak, valamint biztosítja a robot kézi vezérlését videó információ alapján. Az előbbiekben említett funkciók mellett az alkalmazás képes a felhasználó személyzet távvezérlési technikájának elemzésére. Ez a modul segíthet a kézi vezérlést alkalmazó felhasználók kiértékelésében, oktatásában.

Kulcsszavak: távvezérlés, Android alkalmazás, robotirányítás, IoT.

Abstract

Nowadays, IoT (Internet Of Things) technology-based systems are gaining more ground, enabling the efficient remote control, monitoring, and management of industrial processes with smart device applications. In the case of industrial software running on smart devices, reliable monitoring, rapid intervention, and proper management of the access control are critical. Hence, this kind of application development is a complex process.

Industrial robots have their own control devices to program and monitor, but these tools are usually immovable. Therefore, in addition to these devices, mobile phone applications can effectively serve as an extension.

This work aims to develop an Android application that can be used to remotely control robots used in the industry. The developed application can monitor the motion of the robot based on video and sensor informations, it can send commands and control scripts to the robot, and it ensures the manual control of the robot based on video information. In addition to the aforementioned functionalities, the application can analyze the remote control technique of the users. This module can help to evaluate and educate manual robot control techniques.

Keywords: remote control, Android application, robot control, IoT.

Tartalomjegyzék

1. Bevezető	7
1.1. Témaválasztás indoklása	7
1.2. Szakirodalmi áttekintő	8
1.3. Kutatási kérdések	9
1.4. Célkitűzések	10
2. Rendszerspecifikáció	11
2.1. Felhasználói felület	11
2.1.1. Felhasználói követelmények	11
2.1.2. Rendszerkövetelmények	12
2.2. Kommunikációs interfész	13
2.2.1. Rendszerkövetelmények	14
2.3. Kamera interfész	14
2.3.1. Rendszerkövetelmények	14
3. Rendszerarchitektúra	15
3.1. Smartel applikáció	16
3.2. Kommunikációs interfész	18
3.3. Intel RealSense kamera interfész	20
3.4. UR3 robot	21
4. Megvalósítás	25
4.1. Smartel, telefonos applikáció	25
4.2. ZMQ kommunikáció	30
4.3. Intel RealSense kamera interfész	32
5. Mérések	33
5.1. Mérési módszer	33
5.1.1. A mérés leírása	33
5.2. Mérési eredmények	35
6. Összefoglaló	37
6.1. Következtetések	37
6.2. Fejlesztési lehetőségek	37
Ábrák jegyzéke	39

Táblázatok jegyzéke	40
Irodalomjegyzék	42
Függelék	43

1. fejezet

Bevezető

1.1. Témaválasztás indoklása

Napjainkban a technológia már annyira fejlett, hogy a személyi számítógépünk szinte teljes tartalma elfér a zsebünkben. A telefonok egyre kisebbek lesznek, viszont egyre több mindenre képesek. Lassan mindenki számára elérhetőek lesznek ezek az okos eszközök, mivel az alacsonytól az irreálisnak tűnő magas árkategória mindegyikében megfordulnak. Ennek a rohamos fejlődésnek köszönhetően ezeken az eszközökön már nem csak adatokat tárolunk, hanem egy időben különböző más eszközöket is vezérelhetünk. Ezt a vezérlést az IoT (Internet of Things) segítségével lehet megvalósítani. Ez az IoT egy még újnak számító technológia, mely különböző okos eszközök és gépek közötti interakciót teszi lehetővé interneten keresztül.

Az ipar a munkamenet fejlesztése érdekében, a gépek és az emberi erőforrás automatizálása mellett egyre nagyobb hangsúlyt fektet az IoT alapú rendszerek bevezetésébe, ezzel meggyorsítva a termelést, jobban nyomon tudják követni a termékek áramlását, illetve optimalizálni tudják a költségeket, éppen ezért egyre elterjedtebbek az ipari robotok. Az ipari robotok vezérlésére szolgáló rendszerek nagyon komplexek, ezért általában használatuk egy előre betanított személyt feltételeznek. Ebből az okból kifolyólag egy olyan intuitív megoldásra van szükség, mely nem feltételezi a felhasználó jelenlétét, ahhoz, hogy monitorizálni tudja a robot állapotát és bizonyos esetekben beavatkozzon annak működésébe [12].

Az ipari robotok rendelkeznek egy táblagép szerű teach pendant-el, melyek segítségével programozhatóak, illetve állapotaik felügyelhetőek. Ezekkel a táblagépekkel az a gond, hogy általában egy darab van belőlük, nagyok és fizikailag össze vannak kötve a robottal, mely jelentősen megnehezíti a használatukat. Éppen ezért szinte tökéletes kiegészítőeszközként szolgálhatnak a telefonos alkalmazások, melyek segítségével nem csak navigálhatjuk az adott robotot, hanem vizuális visszajelzést is kaphatunk annak aktuális állapotáról.

Ezeknél az alkalmazásoknál kritikus pont a jogosultságok lekezelése, ami azt jelenti, hogy nem minden felhasználó navigálhat a robottal, illetve bizonyos felhasználók csak a robot állapotáról kaphatnak információt.

Az előbbiekben említett információkból kiindulva, dolgozatomban egy olyan Kotlin programozási nyelvben írt Android alkalmazást valósítottam meg, mely képes egy UR3-as ipari robottal kommunikálni egy Python programozási nyelvben implementált szerveren

keresztül, illetve egy másik Python-ban implementál szerver segítségével a felhasználó egy Intel RealSense USB kamera képének segítségével vezérelhet. Választásom azért esett egy Android alkalmazásra, mivel az Android rendszerek az okos eszközök piacának több mint 70%-át uralják. Az Android egy Linux alatt fejlesztett rendszer, melyet a Google menedzsel, okos eszközök operációs rendszere. A robot és az eszköz közötti információcserét ZMQ (Zero Message Queue) üzenetező könyvtár segítségével oldottam meg, míg a jogosultság lekezelését a Firebase autentikációs technológiájával sikerült megvalósítanom, illetve letiltanom azokat a funkciókat, melyekhez nincs joga az adott felhasználónak.

Továbbiakban méréseket is végeztem a felhasználó kiértékelése érdekében. Az alkalmazás egyik funkciója videóinformáció alapján történő navigálása a robotnak az USB kamera segítségével. A kiértékelés érdekében a felhasználó egy előre meghatározott, egyszerű feladatsort kap, mely megértése és megjegyzése után végre kell hajtania ugyan azt a mozgatsort. A végrehajtás befejeztével kiértékelésre kerül a műveletsora különböző metrikákat figyelembe véve. Ezek a kiszámított metrikák alapján a vezérlést végző felhasználó visszajelzést kap, hogy mennyire sikerült elsajátítania a robotvezérlési technikát. Ez abból a szempontból is fontos lehet, hogy nem csak a felhasználó, de feletteseik is fontos információkat nyerhetnek ki a meghatározott metrikákból [9] [10] [11].

1.2. Szakirodalmi áttekintő

Mai napig rengeteg olyan alkalmazást fejlesztettek és továbbra is fejlesztenek, mely lehetővé teszi valamely robot távvezérlését. A távvezérlés vagy más néven teleoperálás annyit jelent, hogy egy robotot vagy bármely más rendszert bizonyos távolságból vezérlünk. Ez a fajta vezérlés akkor hatékony, ha a rendszer környezete veszélyes, megközelíthetetlen, megközelítése túl költséges vagy időigényes, így a navigálás megvalósulhat egy dinamikusán változó, strukturálatlan környezetben is elkerülve a baleseteket. Erre egy jó példa a mobilis robotok távvezérlése, így bizonyos feladatokat végezhetünk vele fizikai beavatkozás nélkül [7] [15].

A teleoperálás történhet manuálisan, hang alapján, illetve akár billentyűzettel vagy egerrel. A hangvezérelt rendszerekkel az a probléma, hogy maga a hangfelismerő rendszer nem megfelelő kifejlesztése alapvető hibákhoz vezethet és nem utolsósorban jó minőségű mikrofont feltételez. A billentyűzet vagy az eger vezérelt robotokkal a probléma, hogy ezeknek a hardver eszközöknek külön helyet kell szánni a rendszer megépítésekor [11].

Továbbá ezeket az alkalmazásokat általában olyan felhasználók is használhatják, akiknek nem szükséges külön képzést végezniük, ahhoz, hogy a robotot vezérelni tudják. Carlos Matteo és munkatársai által létrehozott *Hammer* alkalmazás is egy ilyen Android alkalmazás, melynek célja, hogy egy programozási alapismeretekkel rendelkező, viszont nem képzett személy is vezérelni tudjon egy COMAU robotot [14].

Az Android telefonos alkalmazással vezérelhető rendszerek nagy előnye, hogy nem igényel nagy költségű komponenseket, ahhoz, hogy a követelményeknek megfelelő rendszert hozzanak létre. Éppen ezért kitűnően használhatóak hobbi projektekhez, oktatási, illetve kutatási célokra. Nicolas Oros és Jeffrey L. Krichman csapata éppen ebből az okból kifolyólag egy olyan távvezérelhető járművet építettek, mely Android alkalmazásból vezérlehető Wi-fi-n keresztül, illetve videó és szenzor információkat küld vissza egy személyi számítógépnek. Az *Android Robotic Platform*-nak nevezett alkalmazásuk úgy

tervezték meg, hogy az épített jármű komponensei cserélhetőek legyenek a nélkül, hogy az alkalmazásban nagyon kis vagy egyáltalán ne legyen módosításra szükség [17].

Jan Nádoρνík és munkatársa oktatási célra egy olyan Android mobilalkalmazást fejlesztettek ki, mely egy mobilis robot vezérlését teszi lehetővé Bluetooth kapcsolat által. Az általuk választott robot egy Lego Mindstorms összeszerelhető, kis robot. Ez egy olyan alkalmazás, mely nem csak manuálisan vezérelhető, de a hangvezérlést is lehetővé teszi a felhasználó számára [9].

A Bluetooth kommunikáció egyik nagy hátránya a nagyon korlátolt a hatókör, éppen ezért a felhasználásuk is korlátolt a Wi-fi kommunikáción alapuló rendszerekkel szemben [8].

Napjainkig több olyan rendszert valósítottak meg, mely az előbbieken említett két technológiát ötvözve valósult meg. Egy ilyen rendszer J. Azeta és munkatársai által létrehozott, Android alkalmazással történő, mobilis robot távvezérlésére szolgáló rendszer, mely a robot vezérlésére egy internet protokollt használ, míg a robotra szerelt kamera képének továbbítása az alkalmazás fele Bluetooth kommunikációval történik [6].

Egy másik, az általam megvalósított rendszerhez közelebb álló rendszert valósított meg Sung Wook Moon és munkatársai, akik olyan Android alkalmazást hoztak létre, mely egy általuk épített mobilis robottal képes TCP (Transmission Control Protocol) kommunikációs protokoll által kapcsolatba lépni [16].

A Sapientia Erdélyi Magyar Tudományegyetemen több kutatást is végeztek a mobilis robotok távvezérlésével kapcsolatosan. Ezek közül az egyik Dr. Szántó Zoltán, Dr. Márton Lőrinc és munkatársaik által végzett *Performance analysis of WLAN based mobile robot teleoperation* című kutatás, melyben megállapították, hogy egy mobilis robotot távvezérlő rendszer megvalósításánál kiemelkedően fontos szerepet tölt be a kommunikációs protokoll helyes megválasztása. A kutatásuk során különböző módszerekkel vizsgálták, hogy adott mobilis robot távvezérlése során, hogyan kerülheti el az akadályokat, az ütközést, ha az egy olyan munkatérben dolgozik, ahol több robot is tartózkodik. Végül megállapították, hogy a rendszerük valós idejűsége érdekében UDP kommunikációs protokollt használnak annak gyorsasága miatt, mivel esetükben az ütközés elkerülése egy kritikus pont a rendszerben [18].

Összegezve a fentieket, több olyan alkalmazás létezik, mely különböző kis, és nagy méretű ipari robotok távvezérlésére lett fejlesztve, viszont kutatásom során nem találkoztam olyan rendszerrel, mely Android alkalmazással képes egy UR3 ipari robotot vezérelni.

1.3. Kutatási kérdések

A rendszer megvalósítása során a következő kérdésekre kerestem a választ:

- Hogyan valósítom meg a kommunikációt az alkalmazás és az ipari robot között?
- Milyen kommunikációs protokoll felel meg a legjobban a rendszerem kivitelezéséhez?
- Hogyan kezelem le a felhasználók jogosultságait?
- Melyek azok az alapvető parancsok, melyekkel a felhasználó már könnyen vezérelheti a robotot?
- Hogyan fogom az USB kamera képét átvinni az alkalmazásba?

- Hogyan lehet kiértékelni a kézi vezérlést alkalmazó felhasználót?
- Hogyan fogom kinyerni az adatokat a kézi vezérlés során?
- Hol és hogyan tárolom a kinyert adatokat?
- Hogyan lehet feldolgozni a mérési adatokat?

1.4. Célkitűzések

Az előbbiekben említett nyílt forráskódú applikáció megvalósítása során több célt fogalmaztam meg, mely két nagy kategóriát foglal magába. Az első és a legfontosabb cél, hogy egy megbízható kommunikációt hozhassak létre az alkalmazás és az ipari robot között. E mellett a második kritérium az, hogy az applikáció legyen képes, bizonyos érzékelő és videó információk alapján felügyelni a robot állapotát.

Az előbbiek mellett több kisebb célt fogalmaztam meg, melyek közül az egyik, hogy az alkalmazás információt kapjon a robot aktuális pozíciójáról és azok változásairól. Ezek után fontos volt, hogy a felhasználó ne csak a robot helyzetének változásait láthassa, hanem különböző parancsokat, illetve előre implementált parancssorozatok (szkript) is küldhessen annak. Ezek a funkcionálisok mellett fontos volt, hogy az alkalmazás tegye lehetővé a kézi vezérlést is videó információ alapján.

A fent felsorolt funkcionálisok mellett számomra szükséges volt az is, hogy egy olyan alkalmazást hozhassak létre, melyet csak a felhatalmazott személyek használhatnak. Ezeket a személyeket is két kategóriára osztottam, mégpedig egyik kategóriába tartoznak azok, akiknek minden joguk megvan, hogy információt kaphassanak a robot állapotáról, illetve vezérelhessék is azt. A másik kategóriába azok a felhasználók tartoznak, melyek csak információkat kaphatnak a robot állapotáról.

Végül, de nem utolsó sorban a kézi vezérlést alkalmazó felhasználó személyzet elemzése is a céljaim közé tartozik, melyben a felhasználó pontosságát és gyorsaságát vizsgálom. Ez a funkcionális segíthet a személy kiértékelésében és oktatásában.

2. fejezet

Rendszerspecifikáció

A rendszer specifikációja három nagy részre osztható, melyek közül az egyik a felhasználói felület specifikációja, mely egy Android alkalmazás és lehetővé teszi a felhasználó interakcióját a robottal, a másik a kommunikációs interfész, mely megvalósítja a robot és az Android komponens közötti kommunikációt, míg a harmadik a kamera interfész, mely tartalmazza egy Intel RealSense USB kamera képének átvitelét az Android komponensre, ez által lehetővé téve a robot navigálását videó információ alapján. A rendszer bemutatásához elengedhetetlen diagramokat a Draw.io nevű online szoftver segítségével hoztam létre [2]. Az alkalmazás logóját (F.0.9 ábra) Canva [1] tervező programban alkottam meg.

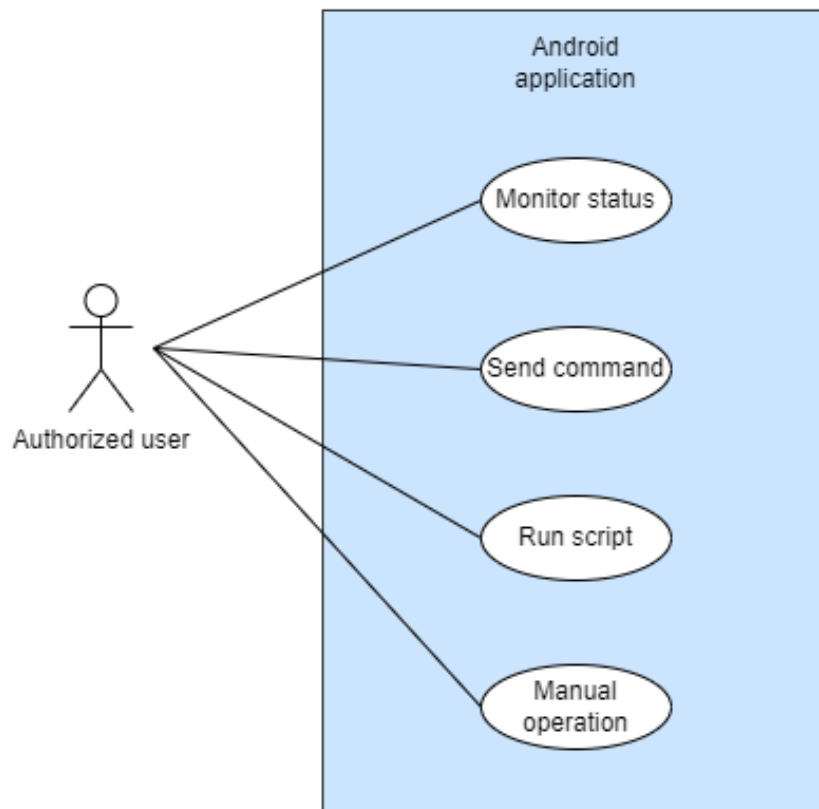
2.1. Felhasználói felület

A rendszer felhasználói felületét egy Android applikáció segítségével hoztam létre. Ennek a felületnek a segítségével a felhasználó interakcióba léphet az ipari robottal, figyelheti annak állapotát, illetve operálhatja is azt.

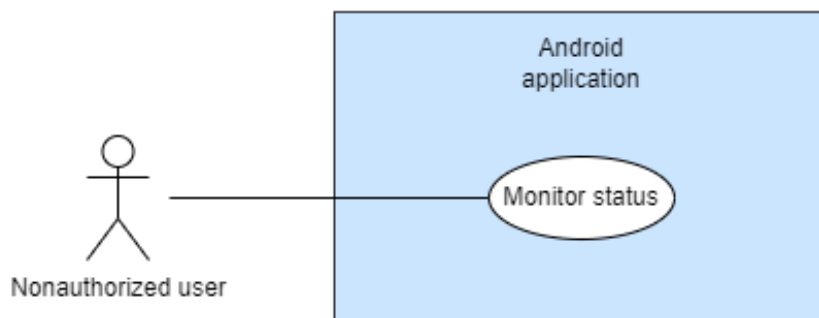
2.1.1. Felhasználói követelmények

Az applikáció használatához kétféle felhasználótípust különítettem el jogosultságaik szempontjából:

- Az első kategóriába azok a felhasználók tartoznak, melyeknek minden joga megvan, hogy információt kaphassanak a robot állapotáról különböző érzékelők, illetve videó információk alapján, e mellett különböző parancsokat, parancssorozatokat (szkript) küldhessenek annak. Ezek a funkcionalitások mellett még joguk van a robot kézi vezérlésére, melynek mozgását videó információ segítségével ellenőrizhetik (2.1. ábra).
- A második kategóriába a korlátozott jogosultságokkal rendelkező személyek tartoznak. Számukra csak a robot állapotának különböző információi érhetőek el, parancsot nem küldhetnek annak (2.2. ábra).



2.1. ábra. Use case diagram - Autorizált felhasználó



2.2. ábra. Use case diagram - Nem autorizált felhasználó

2.1.2. Rendszerkövetelmények

- a) **Funkcionális követelmények:** A felhasználó sikeres bejelentkezése után a kezdőoldal látható, melyen a robot aktuális pozíciói figyelhetőek meg. Annak érdekében, hogy a robot pontos helye legyen elérhető a felhasználó számára két pozíció lát-

ható, melyek közül egyik a csuklók pozíciója és a másik a végberendezés aktuális pozíciója. Mivel a felhasználónak fontos a gyorsaság, így egy fontos kritérium, hogy ezen pozíciók változása nagyon kis idő ($\approx 0.5\text{mp}$) alatt legyen elérhető a felhasználó számára, illetve, hogy hosszú mozgások esetén a koordináták változása szinte valós időben nyomon követhetőek legyenek.

Egy menü segítségével a felhasználó válthat az oldalak között, annak függvényében, hogy van-e jogosultsága a többi menüpont eléréséhez. Ha nincs meg a joga, hogy mozgassa a robotot, így nem tud rálépni ezekre az oldalakra és a megfelelő hibaüzenet értesíti őt.

A kívánt parancsot a parancsoknak szánt menüpontból lehet elérni, melyen belül kiválaszthatóak a különböző parancsok, melynek kiküldésére a felhasználó begépel a kívánt pozíciót, sebességet, illetve gyorsulást, majd kiküldi azokat.

Ahhoz, hogy egy már implementált parancssorozatot tudjon kiküldeni a felhasználó, a megfelelő menüponton egy gomb lenyomásával kiválasztja az eszköze tárhelyéről a megfelelő fájlt és ha az megfelel a robot program standard formátumának, akkor sikeresen kiküldheti.

Abban az esetben, ha a felhasználó szeretné kézzel vezérelni a robotot, akkor az erre a célra létrehozott menüpontot választja, melyben nyilak segítségével operálhatja a robotot x, y és z irányok mentén, illetve ezek a tengelyek menti forgatásra is lehetőség van. Annak érdekében, hogy a felhasználó jobban láthassa, hogy hova navigál egy robotra szerelt kamera képét figyelheti.

Bármely felhasználói felület mellé elengedhetetlenek a UI/UX megkötések, vagyis nem csak funkcionalitás, hanem külalak szempontjából is megfelelő kell legyen egy jól elkészített felület. Az applikáció dizájnja minimalista és könnyen kezelhető azon felhasználók számára is, akiknek nem mindennapi az ilyen és ehhez hasonló alkalmazások használata.

b) **Nem funkcionális követelmények:**

- Android alapú okos eszköz
- Android: 5.0+
- Min API Level: 21(Lollipop)
- Tárhely: 36MB

Az applikáció kizárólag csak Android alapú rendszerek számára elérhető, melyek verziója legalább 5.0 (Min API Level: 21) és rendelkeznek legalább 36MB tárhellyel, hogy a rendszer elérhető és jól működő legyen.

2.2. Kommunikációs interfész

Ez a felület teszi lehetővé az indirekt kommunikációt a felhasználó és a robot között, lényegében ez egy kommunikációs szempontból kialakított szerver. Valójában ezzel a modullal kommunikál az applikáció, nem pedig a robottal. Továbbítja a felhasználó kéréseit a robot fele, illetve a robot által küldött információkat is tovább küldi a felhasználó számára.

2.2.1. Rendszerkövetelmények

a) Funkcionális követelmények:

Mivel a felhasználó számára ez a modul nem látható, módosítható, így ezzel szemben csak a gyorsasággal kapcsolatos megkötései lehetnek.

b) Nem funkcionális követelmények:

- Windows 10 operációs rendszer
- Python 3.7
- pyzmq könyvtár
- ur-rtde könyvtár
- Tárhely: 26MB

A rendszer kommunikációs része Windows 10 operációs rendszeren lett fejlesztve, illetve tesztelve, éppen ezért szükséges ezt a szervert ilyen operációs rendszerrel ellátott eszközön futtatni, melyen legalább 26MB szabad tárhely van a szerver tárolásához és futtatásához. A futtatáshoz szükségesek a pyzmq és az ur-rtde könyvtár-csomagok, illetve 3.7-es vagy annál nagyobb Python verzió is szükséges [4] [5].

2.3. Kamera interfész

A kamera interfész lehetővé teszi az Intel RealSense kamera képének megjelenítését egy weboldalon, majd erről a weboldalról kerül az okos eszköz képernyőjére a videó.

2.3.1. Rendszerkövetelmények

a) Funkcionális követelmények:

A kamera képének megjelenítése során a felhasználónak fontos, hogy a felvétel aránylag jó minőségű legyen, ne legyen szemcsés a kép és ne késsen sokat az adatok átvitele.

b) Nem funkcionális követelmények:

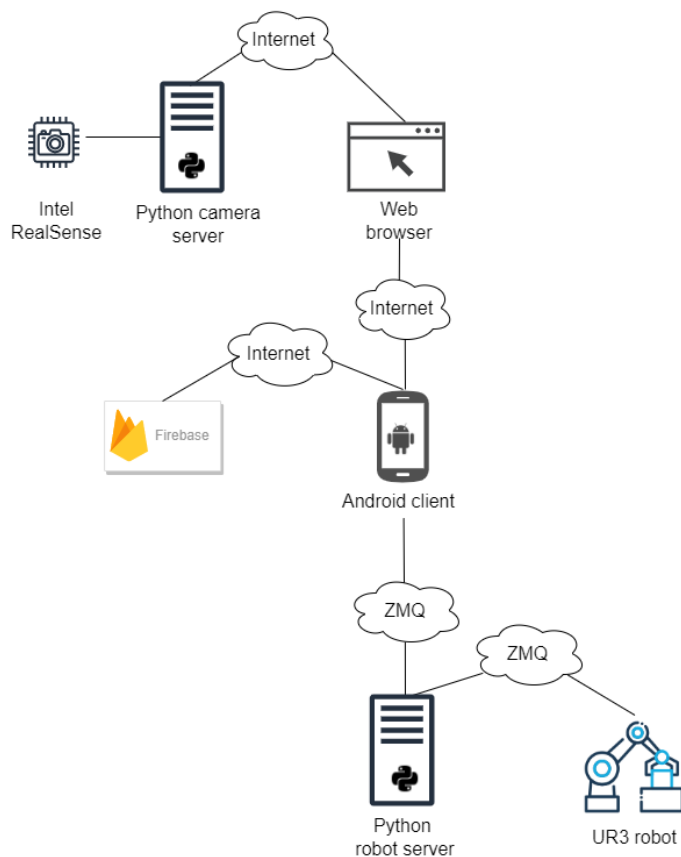
- Windows 10 operációs rendszer
- Python 3.7
- flask könyvtár
- opencv-python könyvtár
- Tárhely: 170MB

A kamera interfész megírása és tesztelése Windows 10 operációs rendszerrel rendelkező számítógépen történt, e miatt ajánlott azon futtatni. E mellett a futtatáshoz Python 3.7 vagy ennél újabb verziójú Python szükséges, e mellett még szükséges két könyvtár-csomag, a flask és az opencv-python könyvtárak. A modul 170MB tárhelyet foglal el.

3. fejezet

Rendszerarchitektúra

A dolgozatban kivitelezett rendszer három nagyobb, különálló komponensből áll, melyeket különböző kommunikációs protokollok segítségével sikerült összekötni.



3.1. ábra. A teljes rendszer architektúrája

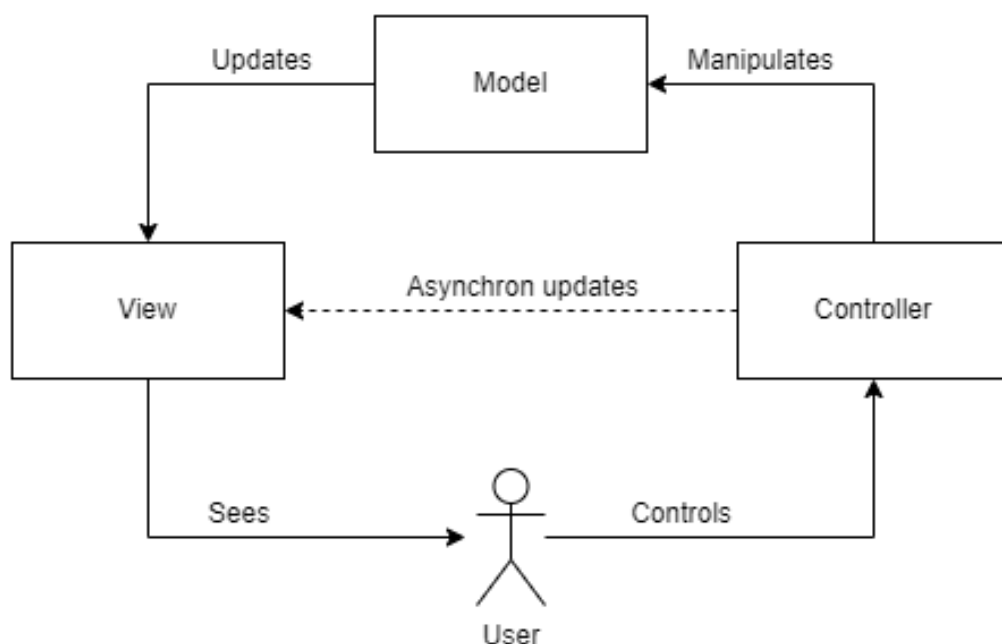
Az első komponens egy Intel RealSense kamera, melynek az egyszerű USB kamera funkcionalitását használtam, hogy videó információt kaphasson a felhasználó a másik nagy entitáson, az Android okos eszközén. Ahhoz, hogy az előbb említett kapcsolat létrejöhessen egy Python-ban írt kamera szerver interneten keresztül Flask keretrendszer segítségével egy weboldalt hozok létre, melyen megjelenik a kamera képe, ahonnan ismét internetes kapcsolat segítségével jelenik meg az eszközön a kép.

Egy másik nagy komponens maga az UR3 robot, melyet ZMQ kommunikációs protokollon keresztül, egy Python-ban implementált szerver segítségével kötöttem össze az Android eszközzel, így küldhet, illetve fogadhat információkat az eszköz a robotnak, illetve a robottól.

Firebase technológiát használok a felhasználók adatainak tárolására és autentikációjára. Ezen felül a méréshez szükséges adatokat is a Firebase által fenntartott valós idejű NoSQL adatbázisba mentem el [3].

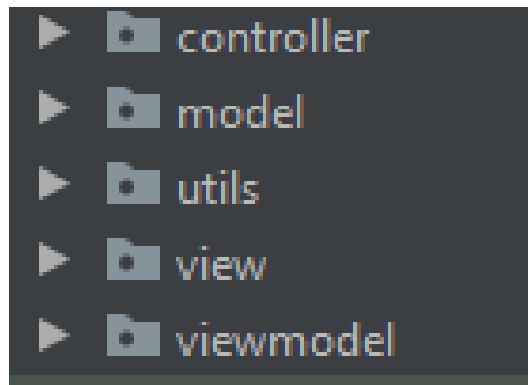
3.1. Smartel applikáció

Az applikáció neve Smartel (Smart Teleoperation), mely a rendszerben lényegében a kliens szerepét tölti be (3.1. ábra).



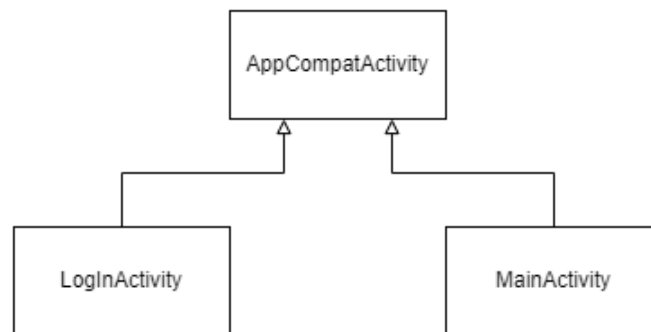
3.2. ábra. Blokk diagram - MVC architektúra

Annak érdekében, hogy a projekt átlátható legyen MVC (Model-View-Controller) architektúrát próbálok követni (3.2. ábra). Mivel ez sem egy tökéletes minta, így nem sikerült teljes mértékben az említett architektúrát megvalósítanom (3.3. ábra).



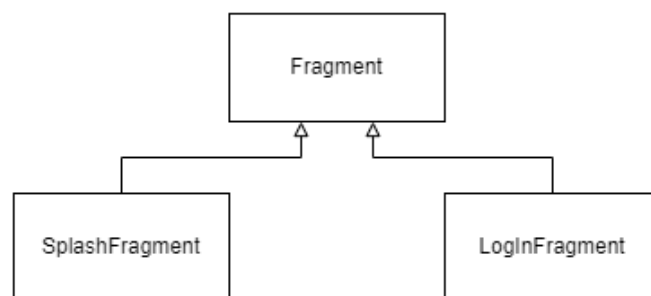
3.3. ábra. A projekt struktúrája

Azért, hogy a rendszer gyorsabb legyen, csak két activity-t használok, majd ezek tartalmát cserélem ki (3.4. ábra).



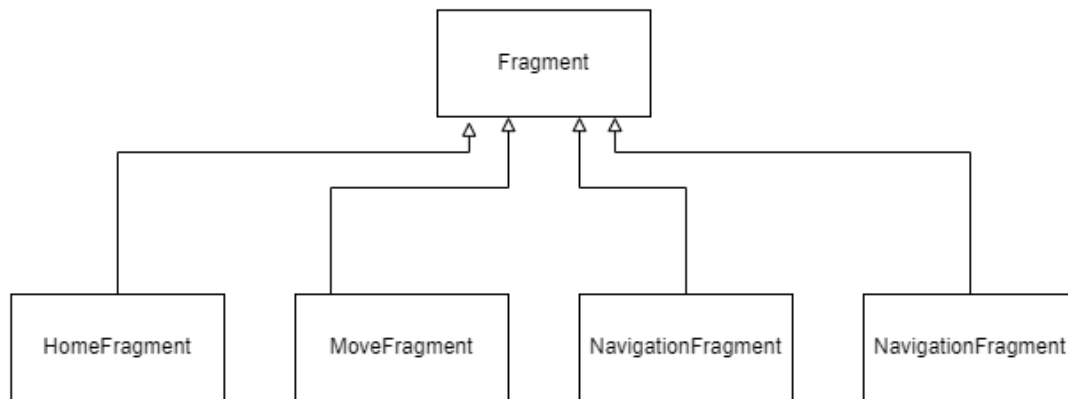
3.4. ábra. Blokk diagram - Az applikáció két activity-je

Az alkalmazás megnyitásakor a Splash, majd a LogIn fragmentek jelennek meg, hogy a felhasználó bejelentkezhessen (3.4. ábra).



3.5. ábra. Blokk diagram - Fragmentek a bejelentkezés activity-hez kapcsolódóan

Ahhoz, hogy a felhasználó könnyedén kommunikálhasson a robottal a UI letisztult és érthető kell legyen számára. Éppen ebből az okból kifolyólag szükséges menüpontok segítségével az alkalmazás funkcionálisainak külön fragmenteket létrehozni (3.6. ábra).



3.6. ábra. Blokk diagram - fragmentek a különálló parancsoknak

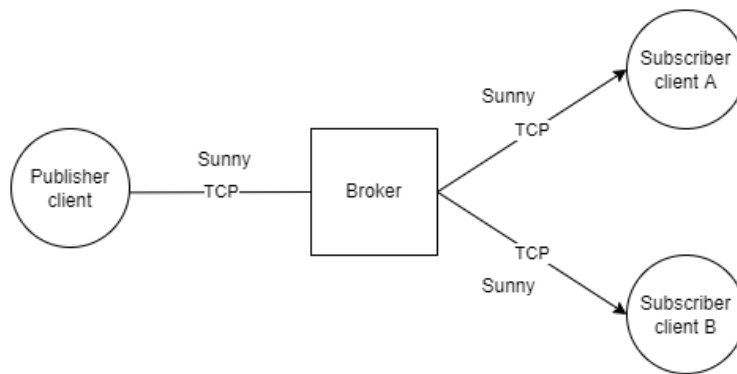
3.2. Kommunikációs interfész

Dolgozatom egyik létfontosságú eleme a kommunikáció, melyhez elengedhetetlen a jól megválasztott kommunikációs protokoll. A rendszerem több oldalról is elemeztem, hogy melyek azok a szempontok, melyek kihagyhatatlanok a számomra megfelelő protokoll kiválasztásakor. Annak ellenére, hogy a TCP kommunikációs protokoll megfelel az általam megfogalmazott kritériumoknak és egyszerű implementálni, úgy döntöttem, hogy nincs szükségem alacsony szintű technológiára. Számomra fontos volt, hogy olyan könyvtárral dolgozzak, mely nyílt forráskódú, különböző programozási nyelvekbe könnyen injektálható, wirelessly keresztüli kommunikáció megvalósítására képes és ezek mellett rendelkezik a TCP pozitív tulajdonságaival, mint a megbízhatóság, garantálja a sorrendhelyes adatátvitelt, illetve hiba esetén újraküldi a csomagot. Az előbb említett szempontok alapján két üzenetező könyvtárat hasonlítottam össze, ezek az MQTT és ZMQ könyvtárak.

Az MQTT (Message Queue Telemetry Transport) egy TCP alapú Subscribe/Publish típusú, pehelysúlyú, két gép közötti kommunikáció megvalósítására kifejlesztett üzenetező protokoll. Ebben a protokollban egy bróker segítségével jut el az üzenet a publikáló kienstől az adott témára feliratkozott kliensek fele (3.7. ábra).

A ZMQ egy nyílt forráskódú univerzális üzenetező könyvtár, mely lehetőséget ad több kommunikációs minta megvalósítására. A nevében szereplő Zero arra utal, hogy kommunikáció lebonyolítása érdekében nincs szükség egy harmadik entitásra, a brókerre.

Annak érdekében, hogy melyik protokoll a számomra megfelelő, különböző tulajdonságok szerint hasonlítottam össze.



3.7. ábra. Blokk diagram - MQTT protokoll architektúrája

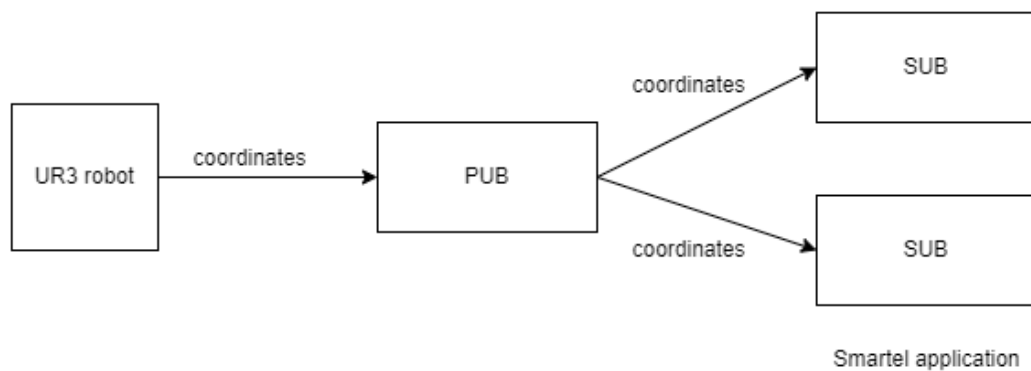
Tulajdonságok	MQTT	ZMQ
TCP protokoll pozitív tulajdonságai	igen	igen
Ingyenes	igen	igen
Nyílt forráskódú	igen	igen
Megbízható	igen	igen
Kétirányú kommunikáció	igen	igen
Bróker jelenléte	igen	nem
Wireless kommunikáció	igen	igen
Több kommunikációs minta	nem	igen

3.1. táblázat. MQTT és ZMQ összehasonlítása.

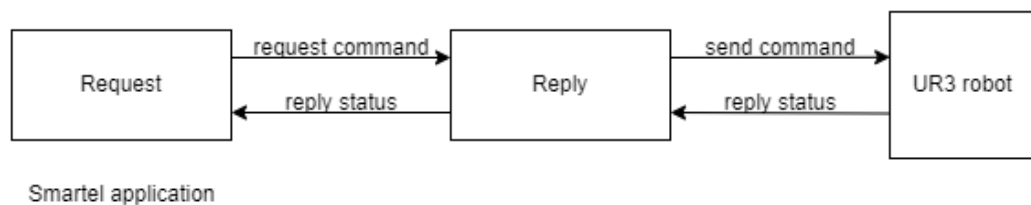
A 3.1 táblázat alapján, végül a döntésem a ZMQ üzenetező könyvtárra esett, mely könnyen implementálható különböző programozási nyelvekbe, ingyenes, nyílt forráskódú, kétirányú wireless kommunikációt tesz lehetővé, nem szükséges egy harmadik entitás, a bróker és több kommunikációs mintára nyújt megvalósítási lehetőséget.

Nem utolsó szempontként a rendszerben jól elkülöníthető kommunikációs minták vehetőek észre. Míg a kommunikációs interfész folyamatosan kéri le a robot aktuális pozícióját és kiküldi az Android kliens számára, addig egy másik szálon a kliens különböző parancsvégrehajtást kezdeményezhet, melynek a megfelelő kérést kiküldi az interfésznek, amely továbbítja az UR3 robot fele. A ZMQ kommunikációs mintái közül az előbb említett mintákra kettő is használható. A Publish/Subscribe minta lehetővé teszi, hogy egyik szálon a kommunikációs interfész folyamatosan lekérje a robot pozícióját, illetve továbbítsa azt az Android kliens fele. A Publish/Subscribe kommunikációs minta abból áll, hogy a Publisher, vagyis a küldő szerver anélkül, hogy ismerné a rácsatlakozott klienseket, folyamatosan küldi ki a meghatározott üzenetet. Annak érdekében, hogy a kliens, vagyis a Subscriber megkapja a szerver által kiküldött üzenetet, fel kell iratkozzon rá (3.8. ábra).

A másik szálon történik a parancs lekezelése, melyre alkalmazni tudtam a Request/-Reply kommunikációs mintát. Ez a minta a legegyszerűbb kliens/szerver mintára épül, tehát abból áll, hogy a kezdeményező fél kérelmez valami végrehajtást, erre a fogadó fél válaszol (3.9. ábra).



3.8. ábra. Blokk diagram - Publish/Subscribe kommunikációs minta

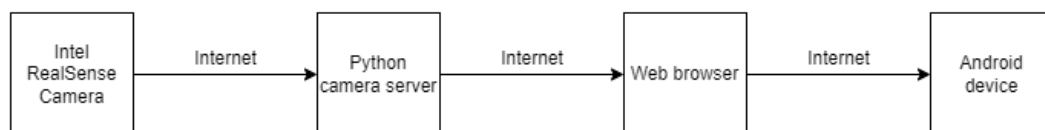


3.9. ábra. Blokk diagram - Request/Reply kommunikációs minta

3.3. Intel RealSense kamera interfész

Az Intel RealSense kamera egy olyan platformfüggetlen mélységi kamera, mely dolgozatomban USB kameraként kerül a robotra.

Ahhoz, hogy a kamera képe megjelenhessen az Android eszközön egy Python programozási nyelven implementált szerver szükséges, ami által a videó betölthető egy weboldalra. Az okos eszközre erről a weboldaltól került a kamera képe (3.10. ábra).



3.10. ábra. Blokk diagram - Intel RealSense kamera szerver

3.4. UR3 robot

Az ipari robotok olyan újra programozható gépek, melyek egy előre meghatározott mozgásszekvencia nagyon pontos végrehajtására képesek. Ezek a robotok, más néven manipulátorok felépítését két részre oszthatjuk:

- Végberendezés: a manipulátor azon része, mely által kapcsolatba lép környezetével. Ez lehet például egy megfogó.
- Bázis: a mechanikai eszköz alapját képi.

A manipulátor szegmensek és csuklók sorozatából tevődik össze. A szegmensek merev tárgyak, melyeket a robot mozgó részei, a csuklók kötnek össze. A robot szabadságfoka a tér dimenziója, ami primitív csuklók esetén megegyezik a csuklók számával. Továbbá megkülönböztetünk két pozíciót, melyek segítségével pontosan leírható a robot helyzete:

- Csuklók pozíciói
- Végberendezés pozíciója [13]

Az ipari robot manipulátorok rendelkeznek egy táblagép szerű eszközzel (teach pendant), melyek segítségével parancsok, parancssorozatok küldhetőek ki a robotnak, programozhatóak, felügyelhetők és ami talán a legfontosabb, hogy tartalmazzák a vészleállító gombot. Ezeknek az eszközöknek a hátrányai, hogy helyhez kötöttek és nagy méretűek.

A dolgozatban megvalósított alkalmazás hatékonyságát egy UR3 roboton vizsgáltam. Az UR3 robot manipulátor egy 6 szabadságfokú ipari robot.

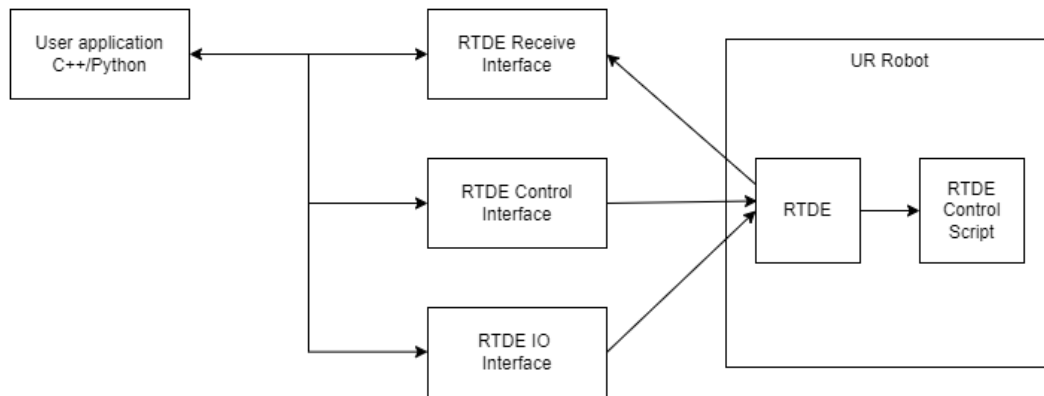
Az UR robotok, a KUKA robotokkal szemben, jobban megfelelnek oktatási célokra, mivel érzékeny szenzorokat tartalmaznak, így biztonságosak, könnyebb az üzembe helyezésük és könnyen mozgathatóak. Ezzel ellentétben a KUKA robotokat inkább az iparban használják a nagyobb teherbírásuk miatt.

Ahhoz, hogy külső entitások is kapcsolatba léphessenek a robottal egy C++-ban fejlesztett könyvtárat hoztak létre, az ur-rtde-t, melyet a C++ mellett, a megfelelő módon Pythonban is használható. Az Universal Robot Real-Time Data Exchange interfész lehetőséget ad, egy TCP/IP kapcsolat kialakítása által a robot kontrollálására és bizonyos adatainak lekérdezésére (3.11. ábra).

Ezeket a műveleteket három interfészen keresztül lehet megvalósítani:

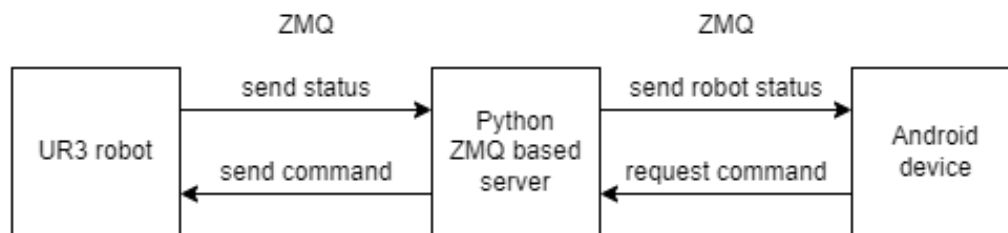
- RTDE Control - elsősorban az robot irányítására használatos interfész
- RTDE Receive - biztosítja a robottól érkező adatok fogadását
- RTDE IO - digitál/analóg IO, illetve sebesség beállításokhoz szükséges

Azért szükséges szétválasztani a kontroll és az IO műveleteket két interfészre, mivel így lehetőség nyílik a robot működése közbeni IO vezérlésre [4].



3.11. ábra. Blokk diagram - UR-RTDE interfész

Tehát az alkalmazás és a robot az előbb említett ur-rtde könyvtár segítségével kapcsolható össze (3.12. ábra).



3.12. ábra. Blokk diagram - UR3 robot és az Android eszköz közötti kommunikáció

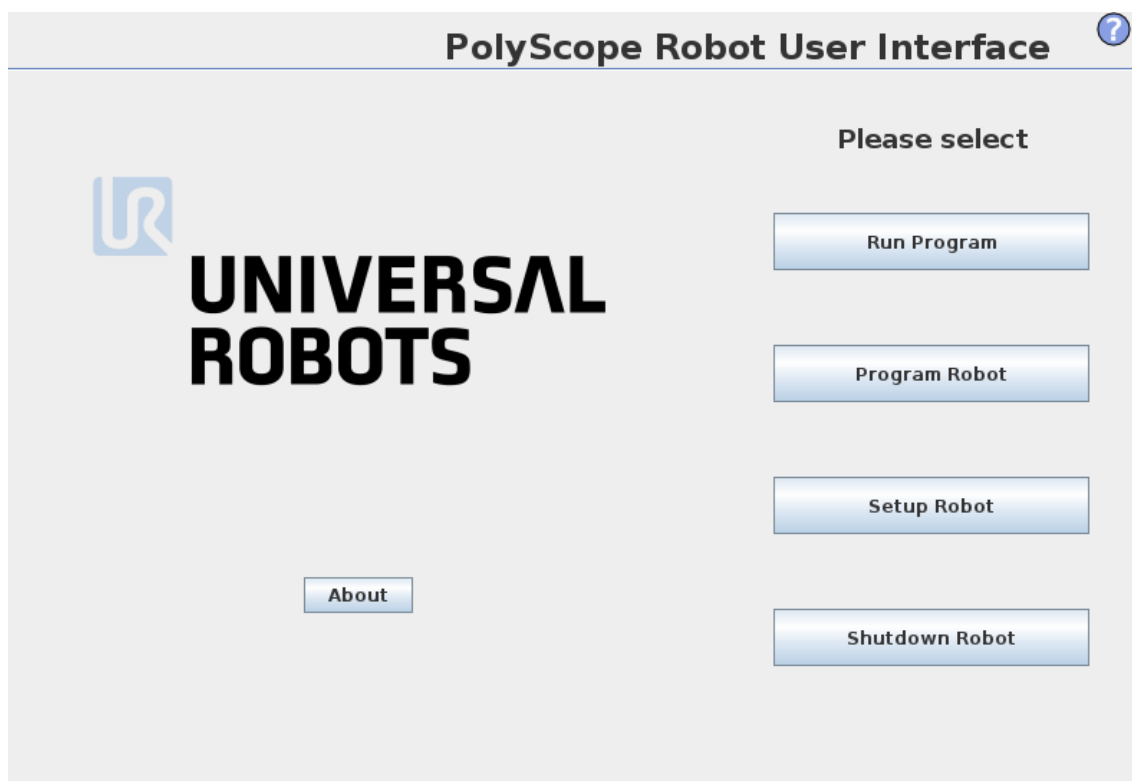
Annak érdekében, hogy kezdetben az alkalmazást ne a fizikai roboton teszteljem, így annak offline szimulátorát használtam. A szimulátort végül Windows rendszeren használok, ezért szükségem volt egy virtuális gépre, amelyen belül egy Ubuntu rendszeren fut a szimulátor.

Különböző menüpontok érhetőek el kezdőoldaltól, innen indítható a robot programozása, illetve már meglévő program (.urp kiterjesztésű) tölthető be. Az „About” pontból elérhetőek el robot adatai, mint például az IP címe, ez által C++ vagy Python szkriptből is valós időben programozható a robot. Ehhez még szükséges a robot portja, ami esetemben a 30004 (3.13. ábra).

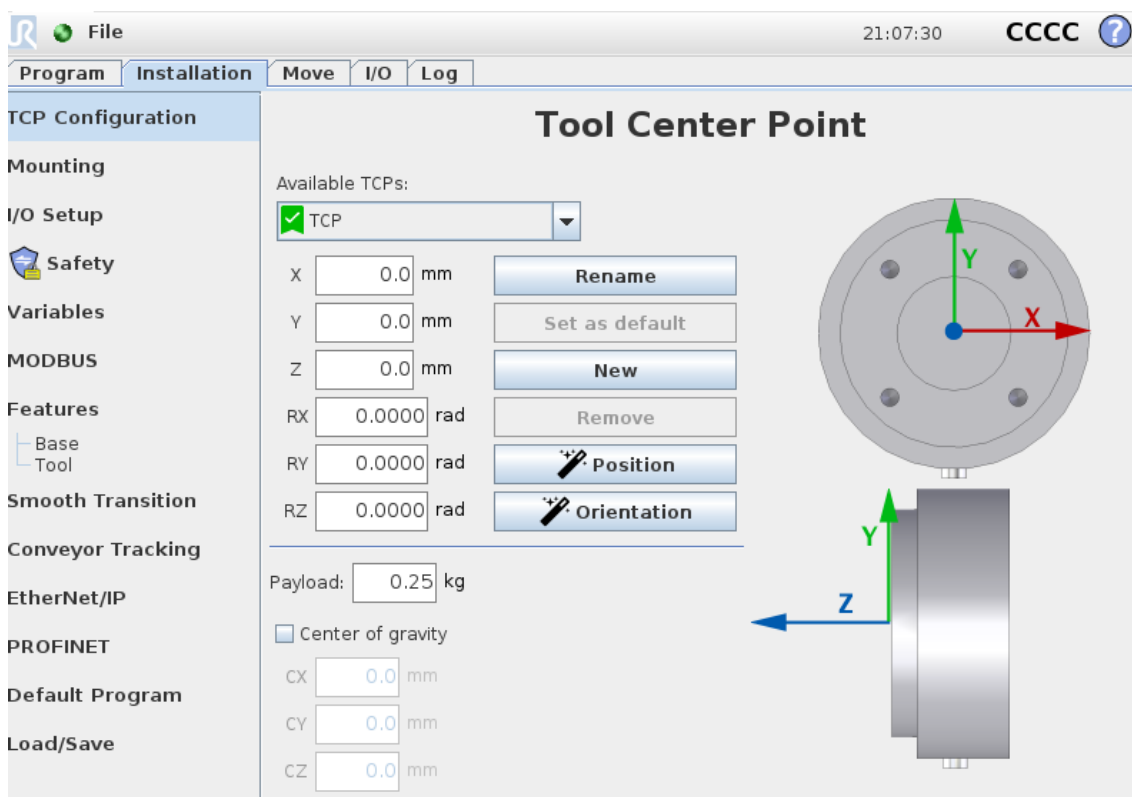
A beállítások oldal teszi lehetővé, például a TCP (Tool Central Point) koordináták beállítását vagy akár a hasznos teherét (3.14. ábra).

A 3.15. ábrán szemléltetett ablakon keresztül finomhangolható, beállítható a robot TCP koordinátái, a csuklók pozíciói.

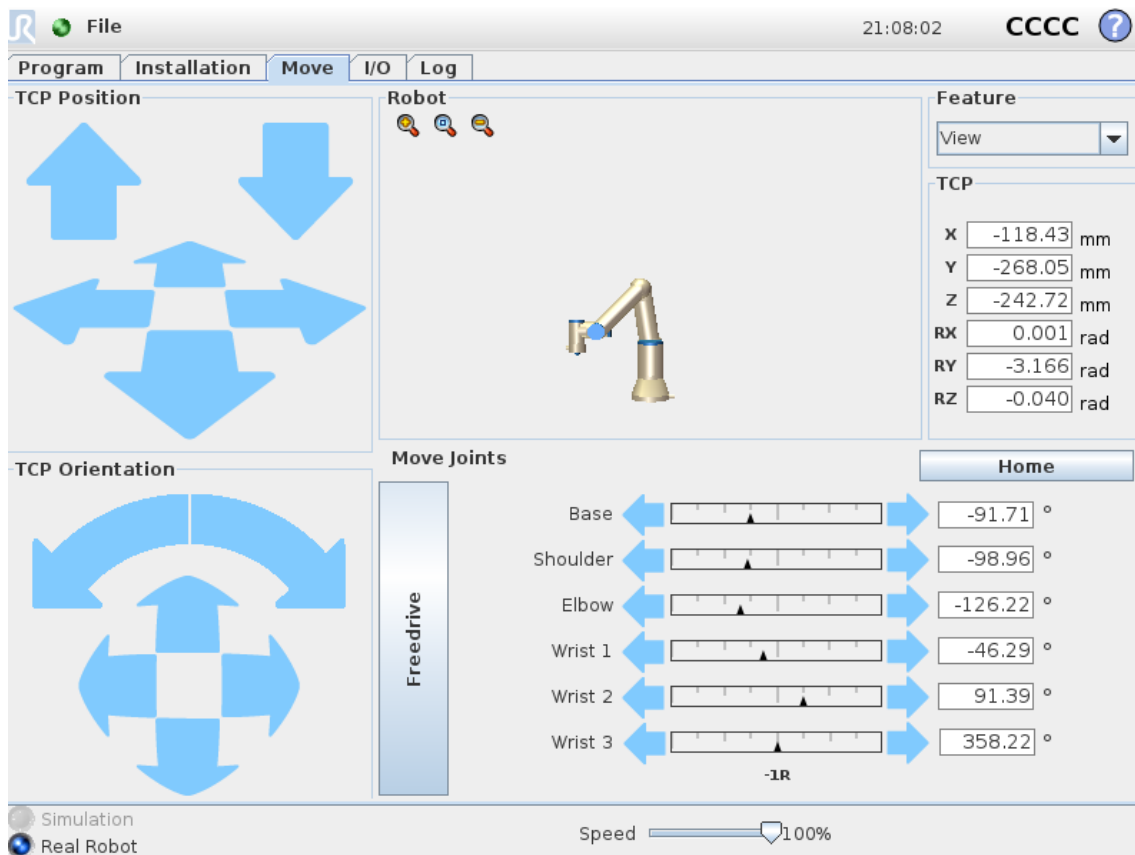
A szimulátorból, vagy más szkriptből történő programozása után a felhasználót egy animációs ablak fogadja, hogy navigáljon a kezdő pozícióba (3.16. ábra).



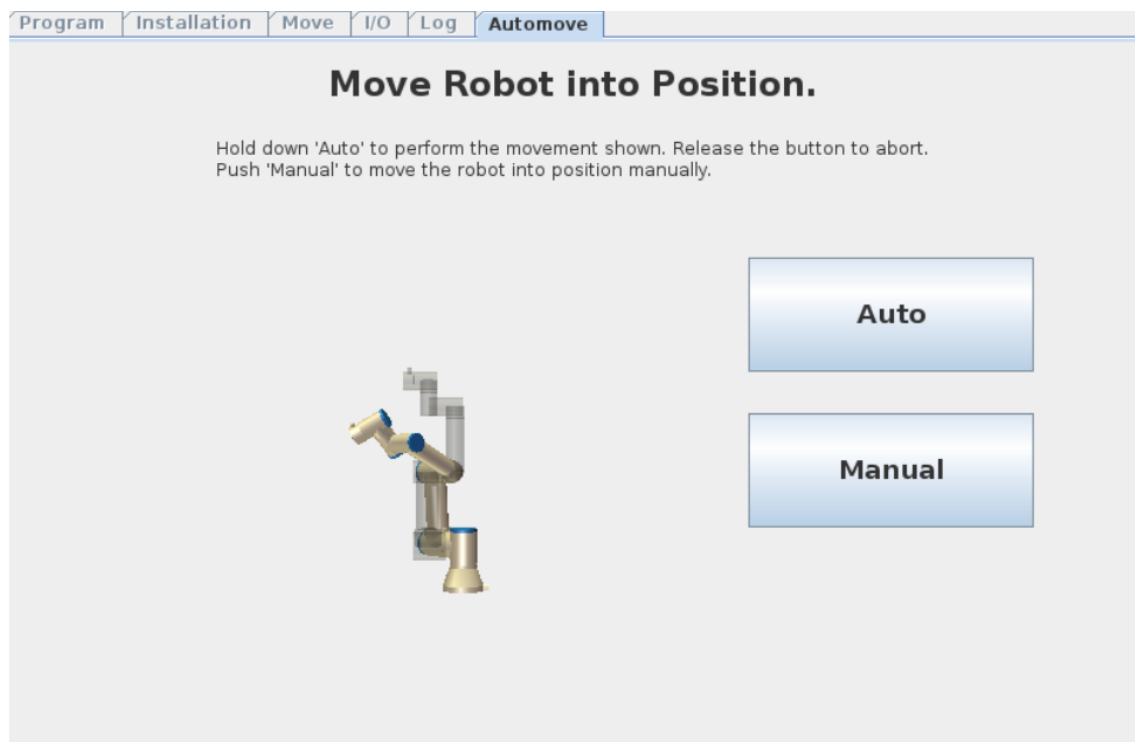
3.13. ábra. Szimulátor - Kezdőoldal



3.14. ábra. Szimulátor - Beállítások



3.15. ábra. Szimulátor - Navigálás



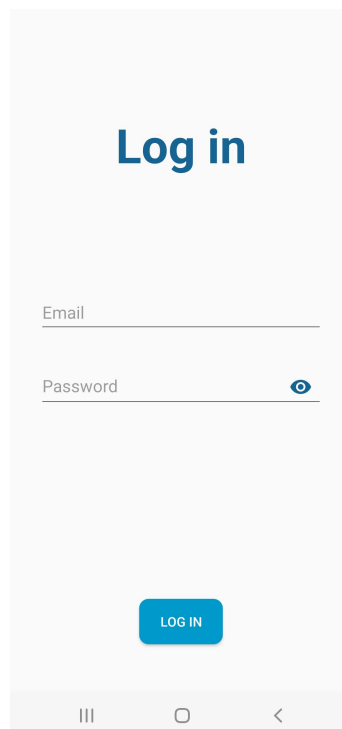
3.16. ábra. Szimulátor - Mozgatás adott pozícióba

4. fejezet

Megvalósítás

4.1. Smartel, telefonos applikáció

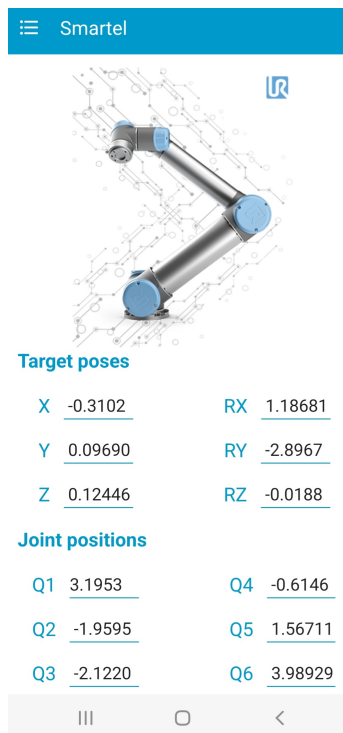
Az alkalmazás megnyitásakor a felhasználó a Splash animáció után a bejelentkezés oldalon találja magát (4.1. ábra).



4.1. ábra. Applikáció - Bejelentkezés

Az applikációban nincs regisztrációra lehetőség, így kiküszöbölve az illetéktelen felhasználást. A regisztrációt a rendszergazda végzi, illetve a jogok megadása is az ő feladatkörébe tartozik. A felhasználók a Firebase technológia, autentikációs lehetőségével vannak eltárolva egy valós idejű adatbázisba. Ebben az adatbázisban a felhasználó e-mail címe, egy random generált azonosító, a rendszerbe való hozzáadásának ideje, utolsó bejelentkezésének ideje, illetve jelszavának hash értéke van eltárolva.

A bejelentkezés validálását is a Firebase végzi, illetve sikertelen bejelentkezés esetén a megfelelő hibaüzenet jelenik meg a felhasználónak. Sikeres bejelentkezés után a kezdő oldal jelenik meg, amely tartalmazza a robot aktuális pozícióit. Az 4.2. ábrán látható a kezdőoldal. A kommunikáció létrejötte után a felső hat számjegy tartalmazza a végbrendezés pozícióit milliméterben. Az alsó hat szám a csuklópozíciók, amelyek fokban láthatóak.



4.2. ábra. Applikáció - Kezdőoldal

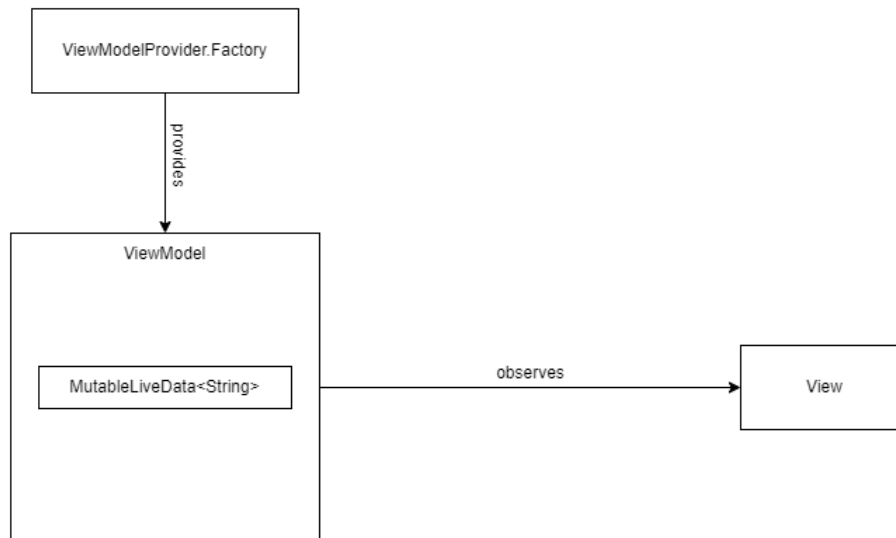
Annak érdekében, hogy az adatok változását észlelje a fragmens, a koordinátákat egy MutableLiveData típusú objektumba tárolom el, majd egy ViewModelProvider.Factory segítségével előállítom a ViewModel-t, mely tartalmazza az előbb említett változót (4.3. ábra).

A balról jobbra, fiókszerűen kihúzható vagy a menüt jelző gombbal elérhető menü tartalmazza a többi elérhető oldalt, melyek segítségével különböző parancsok, parancssorozatok küldhetők a robotnak (4.4. ábra).

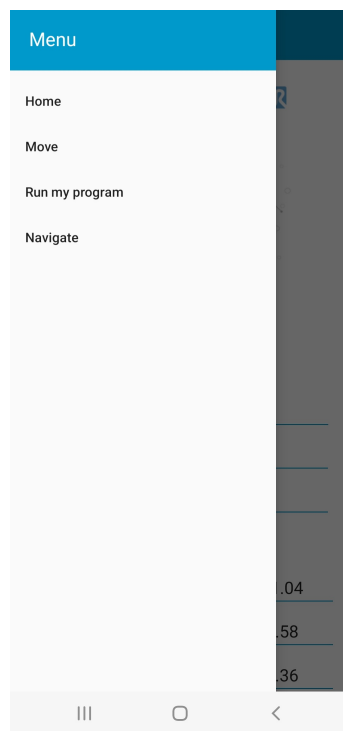
A Move menüpont segítségével elérhető a mozgások kiküldésére létrehozott oldal, melyen kezdetben a moveJ parancs kiküldéséhez szükséges oldal jelenik meg. A MoveJ oldalon a csukló pozíciókat (Q1, Q2, Q3, Q4, Q5, Q6 [rad], illetve ennek függvényében a sebességet (V) [rad/s]-ban, míg a gyorsulást (A) [rad/s²]-ben kell megadja a felhasználó, melyeket a kiküldés után végrehajt a robot (4.5. ábra).

A fenti jobb sarokban található lenyíló menü segítségével elérhető a MoveL oldalon, melyen a pozíció (X, Y, Z, RX, RY, RZ [mm]), sebesség (V[mm/s]), illetve gyorsulás (A[mm/s²]) megadása és kiküldése után lineárisan, egy vonal mentén a meghatározott pozícióba irányítható a robot (4.6. ábra).

A Run script oldalon betölthető egy, az eszközön tárolt, megfelelő struktúrájú fájl, mely kiküldhető, mint parancssorozat (4.7. ábra).



4.3. ábra. Blokk diagram - Live data



4.4. ábra. Applikáció - Menü

Az utolsó menüpontról elérhető a Navigation oldal, mely tartalmaz egy WebView nézetet, mely tartalmazza azt a weboldalt, melyen keresztül nyomon követhető a robot mozgása a rászerezelt kamera segítségével, így könnyítve a felhasználó számára a navigálást, a nyilakat, melyekkel navigálható a robot, illetve egy csúszkát, mely segítségével állítható az operálás sebessége (4.8. ábra).

Smartel

MoveJ ▼

[rad]

Q1 Q4

Q2 Q5

Q3 Q6

V A

[rad/s] [rad/s^2]

SEND

III □ <

4.5. ábra. Applikáció - Oldal a moveJ parancs küldéséhez

Smartel

MoveL ▼

[mm] [rad]

X RX

Y RY

Z RZ

V A

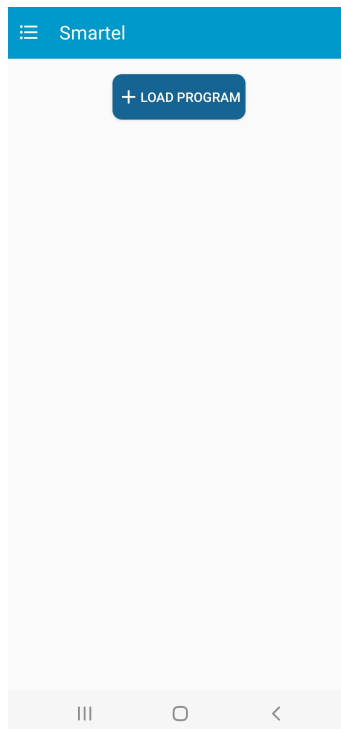
[mm/s] [mm/s^2]

SEND

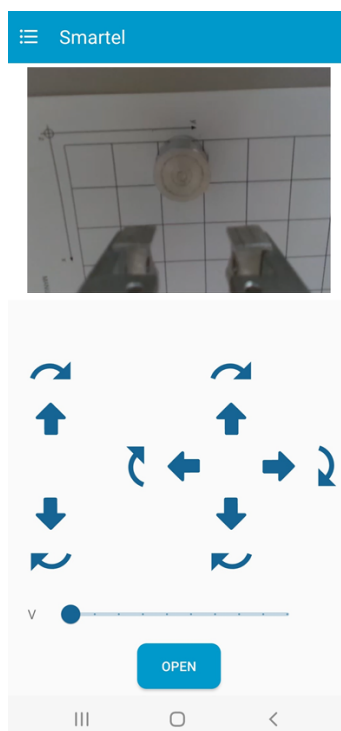
III □ <

4.6. ábra. Applikáció - Oldal a moveL parancs küldéséhez

Az applikáció Coroutin-ok segítségével egy szálon küldi ki a struktúrált üzenetet a szervernek, illetve egy másik szálon történik a koordináták folyamatos kezelése.



4.7. ábra. Applikáció - Oldal szkript küldéshez

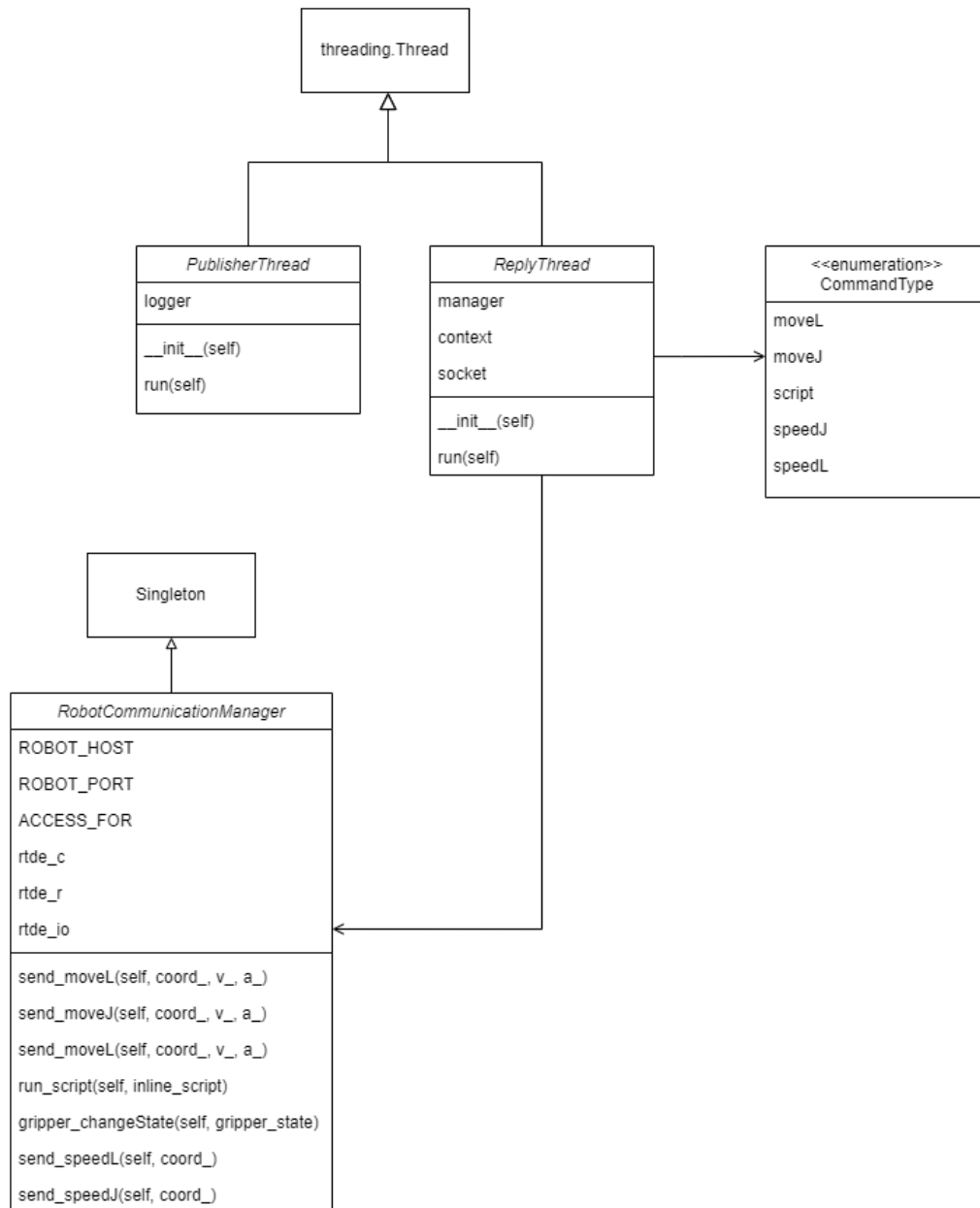


4.8. ábra. Applikáció - Oldal a robot kézi vezérléséhez

A mérési metrikákhoz szükséges adatokat a Navigation oldalról mentem ki egy ismét Firebase által nyújtott valós idejű NoSQL adatbázisba.

4.2. ZMQ kommunikáció

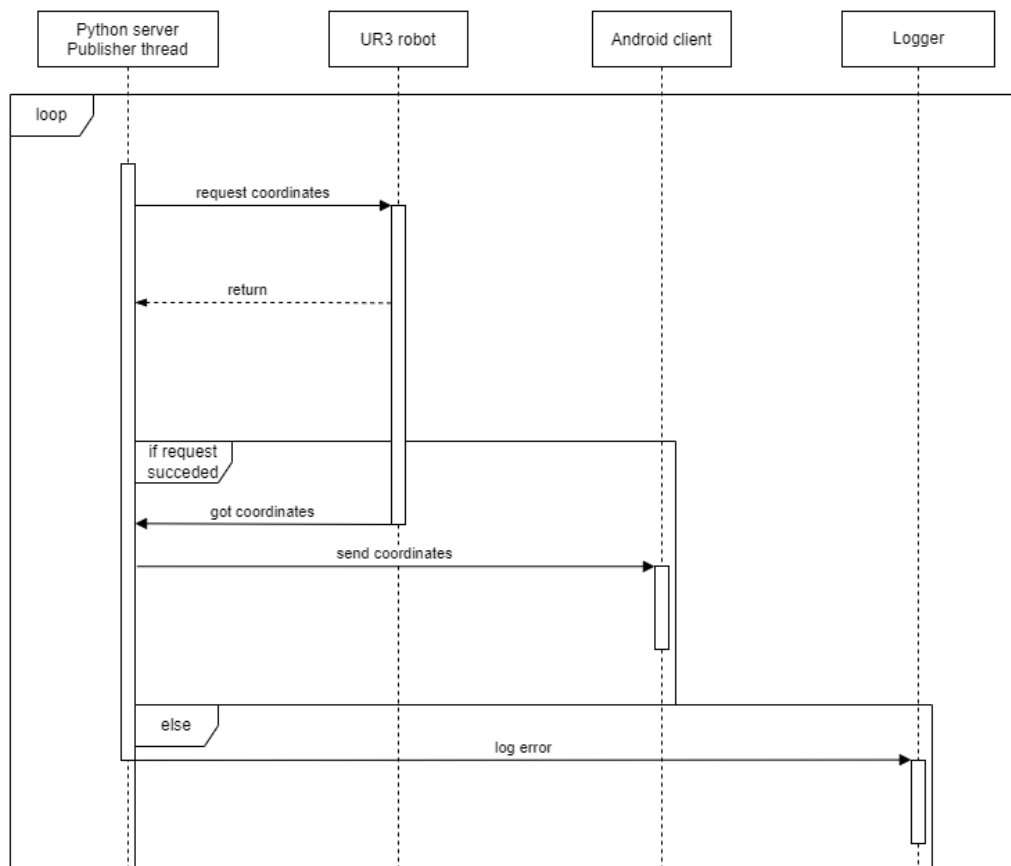
A ZMQ kommunikációs protokoll által nyújtott két minta segítségével valósítom meg a rendszer kommunikációs részét, a 3.2 fejezetben bemutatott Publish/Subscribe és Request/Reply mintákkal.



4.9. ábra. Osztály diagram

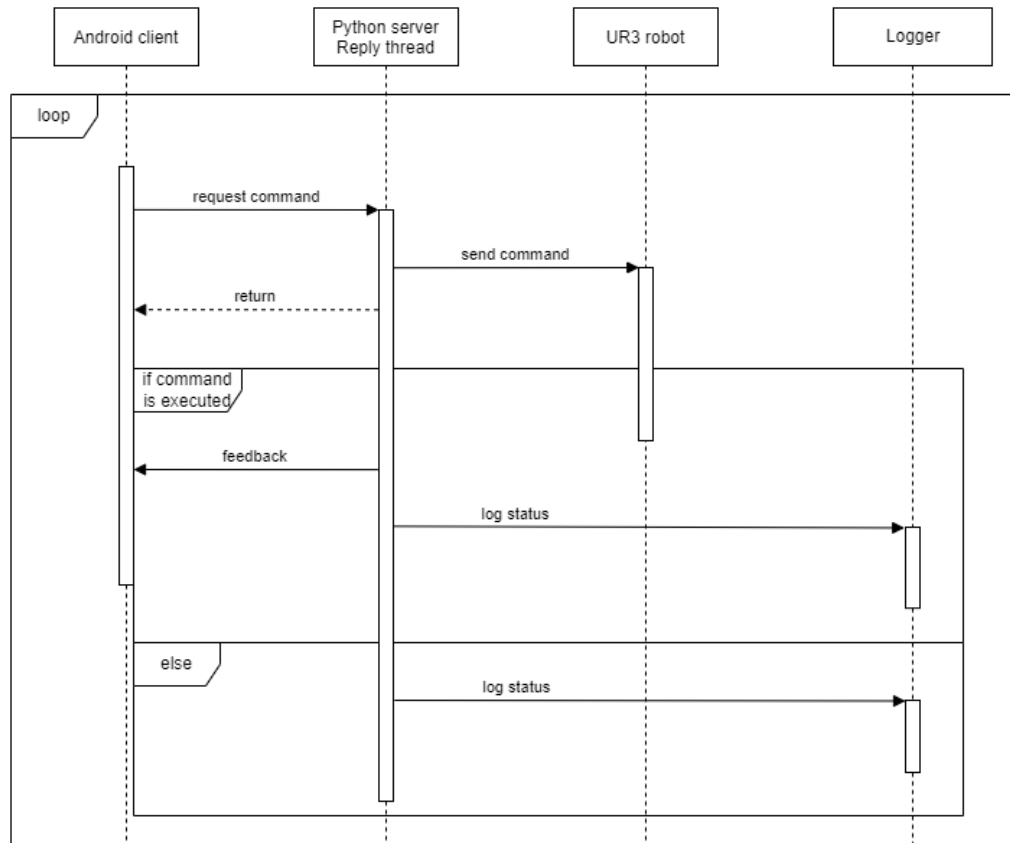
Míg egyik `PublisherThread` osztály felelős a koordináták folyamatos robottól való lekérdezéséért és a rá csatlakozott klienseknek való kiküldéséért, addig a másik `ReplyThread` osztálynak az a feladata, hogy fogadja a kliensek kéréseit és továbbítsa azt a ro-

bot fele. Az üzenetek JSON formátumban érkeznek meg a szerverhez. Ez a struktúrált üzenet tartalmazza a végrehajtandó folyamat azonosítóját (CommandType), a felhasználó azonosítóját, ami által eldönthető, hogy mihez van joga az adott felhasználónak (ACCESS_FOR listában szerepel-e), illetve még az elküldött, a parancs végrehajtásához szükséges adat szerepel: koordináták, sebesség, gyorsulás, esetlegesen a megfogó állapota és a kiküldeni kívánt szkript. A RobotCommunicationManager az a Singleton osztály, mely az Egyetlen Felelőség Elvének (Single Responsibility Principle) eleget téve különválasztja a szál, illetve a robot kommunikációs menedzser feladatait (4.9. ábra).



4.10. ábra. Szekvencia diagram - A koordinátákat folyamatosan küldő szál

A Publisher thread komponens előbb lekéri az UR3 robot aktuális pozícióit, majd ha az sikeresen megérkezett továbbküldi az Android komponensnek. Ellenben, ha valami hiba folytán nem sikerült lekérni a egy fájlba és konzolra naplózza a hibát (4.10. ábra).



4.11. ábra. Szekvencia diagram - Parancsot fogadó és továbbító szál

A Reply thread először megkapja a kérést az Android komponenstől és továbbküldi a robot fele. Ha a parancs végrehajtása sikeres volt, akkor a felhasználó vizuális visszajelzést kap és konzolra naplózza az eseményt. Ellenben, ha nem sikerül végrehajtani, akkor is a hibát konzolra naplózza (4.11. ábra).

A rendszer beüzemeléséhez szükséges adatok egy konfigurációs fájlba vannak elmentve, ami megkönnyíti az üzembe helyezés folyamatát.

4.3. Intel RealSense kamera interfész

Annak érdekében, hogy a felhasználó a navigálás során láthassa, hogy éppen hol tart a robot egy Intel Realsense kamerát használok. Ez a kamera könnyen csatlakoztatható személyi számítógépekhez, illetve egyszerűen megoldható, hogy a kamera képe egy böngésző oldalán jelenhessen meg. Ahhoz, hogy a videó mintavételezett képei megjelenjenek a Flask keretrendszer segítségével létrehozott weboldalon az open-cv könyvtárat használom.

5. fejezet

Mérések

5.1. Mérési módszer

A dolgozat második felében elemzem a kézi vezérlést alkalmazó felhasználókat gyorsaságuk és pontosságuk alapján.

A felhasználó egy előre definiált mozgássorozatot kell elvégezzen és a mérés során elmentett adatok feldolgozásával meghatározhatóak a metrikák. Az elvárt megérkezési pozíciót előre meghatározzuk. Az elvárt és a valós megérkezési pont közötti távolság adja meg a felhasználó pontosságát, amit három részre osztunk:

- x-y tengelyek menti mozgás pontosságát a

$$\Delta_{xy} = \sqrt{(x_r - x_u)^2 + (y_r - y_u)^2}$$

képlettel határozzuk meg, ahol x_r a robot pozíciója x tengely mentén, x_u a felhasználó pozíciója x tengely mentén, y_r a robot pozíciója y tengely mentén, y_u a felhasználó pozíciója y tengely mentén.

- z tengely menti mozgás pontosságát a

$$\Delta_z = |z_r - z_u|$$

képlettel határoztuk meg, ahol z_r a robot pozíciója z tengely mentén, z_u a felhasználó pozíciója z tengely mentén.

A felhasználó gyorsaságát a végrehajtási idő adja meg, ami az elindulás és a feladat befejezése közötti eltelt idő, melyet

$$\delta t = t_r - t_u,$$

képlettel határozzuk meg, ahol t_r a robot idejét, míg a t_u a felhasználó idejét jelölik.

5.1.1. A mérés leírása

A felhasználó egy előre definiált mozgássorozatot végez el négy alkalommal, melyből az első alkalom a próbakör, melynek eredménye nem kerül feldolgozásra. A mérőberendezés a [F.0.2](#) ábrán látható.

Az mérési adatokat a Navigation oldalról (4.8 ábra) mentem el egy valós idejű Firebase NoSQL adatbázisba. Az adatbázis kiválasztásakor fontos szempont volt, hogy a technológia könnyen implementálható legyen, e mellett legyen valós idejű, legyen rövid válaszideje és legyen mindig elérhető. Az adatbázis neve KeyLogger, melybe elmentem a gomb azonosítóját, a dátumot idővel, a csuklók pozícióit, a gomb lenyomásának és felengedésének idejét, a mozgás sebességét, a végberendezés pozícióit és a GDPR-ra való tekintettel csak a felhasználó azonosítóját (5.1 ábra).



5.1. ábra. Az adatbázis struktúrája

Annak érdekében, hogy felhasználó könnyebben navigálhasson végig az adott pályán a F.0.1. függelékben látható kézi vezérlést segítő lap úgynevezett pályaként szolgál számára.

Azért hogy a felhasználó a megfogó végeit is láthassa a videón keresztül, a kamerát bizonyos szögbe kellett felszerelni a robotra. Ez a dőlési szög miatt egy eltolás kerül a képbe, így akkor mondhatjuk, hogy a megfogó az adott tárgy felé kerül, ha a kamera képén a megfogó és a tárgy között egy egységnyi távolság van (5.2. ábra).

A mérési folyamat kezdetét a megfogó kinyitása jelzi, mígy a folyamat végét a megfogó becsukása.



5.2. ábra. A kamera képe a céltárgy felett

5.2. Mérési eredmények

A mérési eredmények által nem csak a felhasználó, de feletteseik is értékes információkhoz juthatnak a kézi vezérlést alkalmazó személy robotvezérlési technikájáról.

A kísérlet során öt személyt kézi vezérlési technikáik alapján vizsgálunk. Mind az öt személy a pálya és a feladatsor megismerése és egy próbakör után három alkalommal végezte el ugyan azt a mozgásszekvenciát. Az adatokat manuálisan egy Excel táblázatba mentettem el.

UID	Δ_{xy} [mm]	Δ_z [mm]	δt [s]
F1	6	38	16,333
F2	6	15	19,667
F3	18	36	24,667
F4	4	8	26,000
F5	8	5	20,667

5.1. táblázat. Metrikák átlagai felhasználónként.

Az összes kísérlet elvégzése során azt a következtetést vontuk le, hogy a végrehajtási idő növekedésével csökken a pontosság mértéke, illetve az adatokból az is kiolvasható, hogy a z tengely menti mozgás kevésbé pontos az xy engelyek menti mozgásokhoz képest szinte mindegyik felhasználó esetében (5.3. 5.4. ábrák).



5.3. ábra. Összes felhasználó x-y tengelyek menti végrehajtási idő szerinti pontosság vizualizálása.



5.4. ábra. Összes felhasználó z tengely menti végrehajtási idő szerinti pontosság vizualizálása.

6. fejezet

Összefoglaló

6.1. Következtetések

Dolgozatomban sikerült egy olyan Android alkalmazást megvalósítanom, mely egy szerveren keresztül képes kommunikálni egy UR3 ipari robottal. Az alkalmazás képes adatot fogadni a robottól, az arra jogosult személyek különböző mozgásokat, előre implementált mozgássorozatokot küldhetnek ki a robotnak, e mellett lehetőséget ad a robot kézi vezérlésére is, melyről videó információ alapján visszajelzést kaphat.

Az alkalmazás képes adatok mentésére és kiolvasására egy valós idejű adatbázisba, melyek segítik az alkalmazás felhasználóinak kiértékelését.

A mérések alapján három féle metrikát határoztunk meg: x-y tengelyek menti pozíció pontosságát, z tengely menti pozíció pontosságát és a szekvencia végrehajtásának sebességét. Ezen metrikák alapján megállapítottuk, hogy a sebesség növekedésével csökken a x, y, z tengelyek menti mozgások pontossága.

6.2. Fejlesztési lehetőségek

A megvalósított rendszer fejlesztésére több lehetőség is van. Első sorban azt a célt fogalmaztam meg, hogy több mérést végezzek el. E mellett célom, hogy az alkalmazás több parancs kiküldésére legyen képes. A jövőbeli terveim közé tartozik az is, hogy komplexebb feladatokon is méréseket végezzek. Valamint célom, hogy a méréseket különböző sebességeken is elvégezzem. Ezek mellett fontosnak tartom további mérési lehetőségek biztosítását az orientáció változtatásával.

Hosszú távú célokat is megfogalmaztam, melyek közül az egyik a hekkertámadások elleni védelem, mert az IoT alapú rendszerek nagy mértékben ki vannak téve az ilyen típusú támadásoknak, így a jövőben szükségesnek látom valamilyen kriptográfiai algoritmus, hash függvény bevezetését a kommunikációba.

Egy másik távolabbi célom egy olyan keretrendszer bevezetése, mint például a Flutter, mely által az applikáció platformfüggetlen lehet.

Ábrák jegyzéke

2.1. Use case diagram - Autorizált felhasználó	12
2.2. Use case diagram - Nem autorizált felhasználó	12
3.1. A teljes rendszer architektúrája	15
3.2. Blokk diagram - MVC architektúra	16
3.3. A projekt struktúrája	17
3.4. Blokk diagram - Az applikáció két activity-je	17
3.5. Blokk diagram - Fragmentek a bejelentkezés activity-hez kapcsolódóan	17
3.6. Blokk diagram - fragmentek a különálló parancsoknak	18
3.7. Blokk diagram - MQTT protokoll architektúrája	19
3.8. Blokk diagram - Publish/Subscribe kommunikációs minta	20
3.9. Blokk diagram - Request/Reply kommunikációs minta	20
3.10. Blokk diagram - Intel RealSense kamera szerver	20
3.11. Blokk diagram - UR-RTDE interfész	22
3.12. Blokk diagram - UR3 robot és az Android eszköz közötti kommunikáció	22
3.13. Szimulátor - Kezdőoldal	23
3.14. Szimulátor - Beállítások	23
3.15. Szimulátor - Navigálás	24
3.16. Szimulátor - Mozgatás adott pozícióba	24
4.1. Applikáció - Bejelentkezés	25
4.2. Applikáció - Kezdőoldal	26
4.3. Blokk diagram - Live data	27
4.4. Applikáció - Menü	27
4.5. Applikáció - Oldal a moveJ parancs küldéséhez	28
4.6. Applikáció - Oldal a moveL parancs küldéséhez	28
4.7. Applikáció - Oldal szkript küldéshez	29
4.8. Applikáció - Oldal a robot kézi vezérléséhez	29
4.9. Osztály diagram	30
4.10. Szekvencia diagram - A koordinátákat folyamatosan küldő szál	31
4.11. Szekvencia diagram - Parancsot fogadó és továbbító szál	32
5.1. Az adatbázis struktúrája	34
5.2. A kamera képe a céltárgy felett	34
5.3. Összes felhasználó x-y tengelyek menti végrehajtási idő szerinti pontosság vizualizálása.	35
5.4. Összes felhasználó z tengely menti végrehajtási idő szerinti pontosság vizualizálása.	36

F.0.1Manuális vezérlést segítő lap	43
F.0.2Mérőberendezés.	44
F.0.3A mérés menete.	45
F.0.41-es felhasználó mérési eredményeinek vizualizálása.	46
F.0.52-es felhasználó mérési eredményeinek vizualizálása.	46
F.0.63-as felhasználó mérési eredményeinek vizualizálása.	47
F.0.74-es felhasználó mérési eredményeinek vizualizálása.	47
F.0.85-ös felhasználó mérési eredményeinek vizualizálása.	48
F.0.9A tervezett logó.	48

Táblázatok jegyzéke

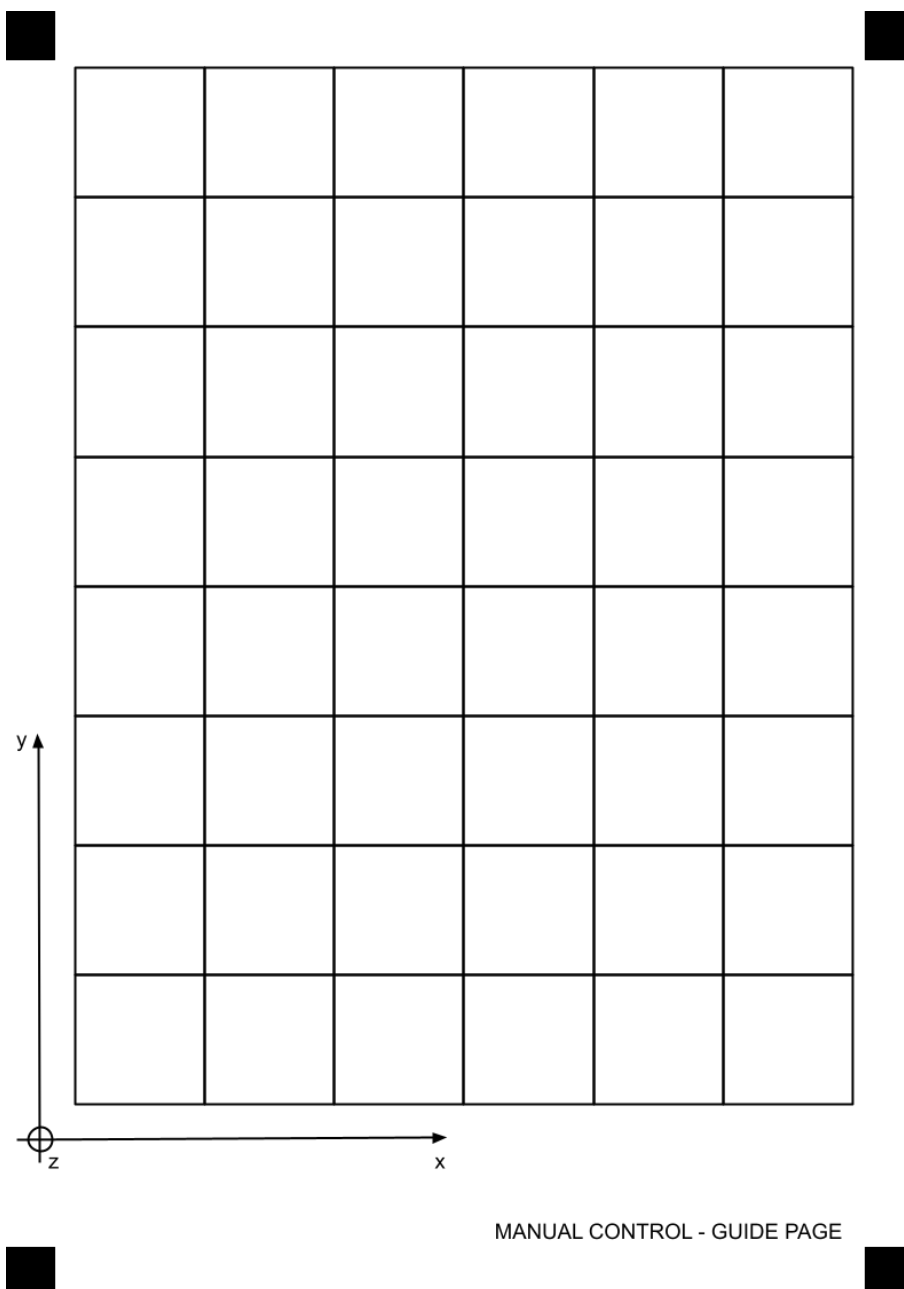
3.1. MQTT és ZMQ összehasonlítása.	19
5.1. Metrikák átlagai felhasználónként.	35

Irodalomjegyzék

- [1] Canva. <https://www.canva.com/>.
- [2] Draw.io online felület a diagramok elkészítéséhez. <https://app.diagrams.net/>.
- [3] Firebase dokumentáció. <https://firebase.google.com/docs/>.
- [4] Universal robot rtde c++ interfész dokumentáció. https://sdurobotics.gitlab.io/ur_rtde/.
- [5] Zmq dokumentáció. <https://zeromq.org/>.
- [6] AZETA, J., BOLU, C., HINVI, D., ABIOYE, A., BOYO, H., ANAKHU, P., AND ONWORDI, P. An android based mobile robot for monitoring and surveillance. *Procedia Manufacturing* 35 (2019), 1129–1134. The 2nd International Conference on Sustainable Materials Processing and Manufacturing, SMPM 2019, 8-10 March 2019, Sun City, South Africa.
- [7] BOBOC, R., MOGA, H., AND TALABA, D. A review of current applications in teleoperation of mobile robots. *Bulletin of the Transilvania University of Brasov. Engineering Sciences. Series I* 5, 2 (2012), 9.
- [8] BOKADE, A. U., AND RATNAPARKHE, V. R. Video surveillance robot control using smartphone and raspberry pi. In *2016 International Conference on Communication and Signal Processing (ICCSP)* (2016), pp. 2094–2097.
- [9] JAN, N., AND PAVEL, S. Remote control robot using android mobile device. In *Proceedings of the 2014 15th International Carpathian Control Conference (ICCC)* (2014), pp. 373–378.
- [10] LEE, I., AND LEE, K. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons* 58, 4 (2015), 431–440.
- [11] LU, X., LIU, W., WANG, H., AND SUN, Q. Robot control design based on smartphone. In *2013 25th Chinese Control and Decision Conference (CCDC)* (2013), pp. 2820–2823.
- [12] LUTOVAC BANDUKA, M. Remote monitoring and control of industrial robot based on android device and wi-fi communication. *Automatika–Journal for Control, Measurement, Electronics, Computing and Communications* 56, 3 (2015).
- [13] LŐRINC, M. *Irányítástechnika*. Scientia, 2009.

- [14] MATEO, C., BRUNETE, A., GAMBAO, E., AND HERNANDO, M. Hammer: An android based application for end-user industrial robot programming. In *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)* (2014), pp. 1–6.
- [15] MONTÚFAR-CHAVEZNAVA, R., PAISANO, C., CAÑAS, J. M., AND DE JÓDAR, E. Simulating telerobotics by cellular telephony. In *ICINCO* (2005), pp. 329–334.
- [16] MOON, S. W., KIM, Y. J., MYEONG, H. J., KIM, C. S., CHA, N. J., AND KIM, D. H. Implementation of smartphone environment remote control and monitoring system for android operating system-based robot platform. In *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)* (2011), pp. 211–214.
- [17] OROS, N., AND KRICHMAR, J. L. Smartphone based robotics: Powerful, flexible and inexpensive robots for hobbyists, educators, students and researchers. *IEEE Robotics & Automation Magazine* 1, 3 (2013).
- [18] SZÁNTÓ, Z., MÁRTON, L., HALLER, P., AND GYÖRGY, S. Performance analysis of wlan based mobile robot teleoperation. In *2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing (ICCP)* (2013), pp. 299–305.

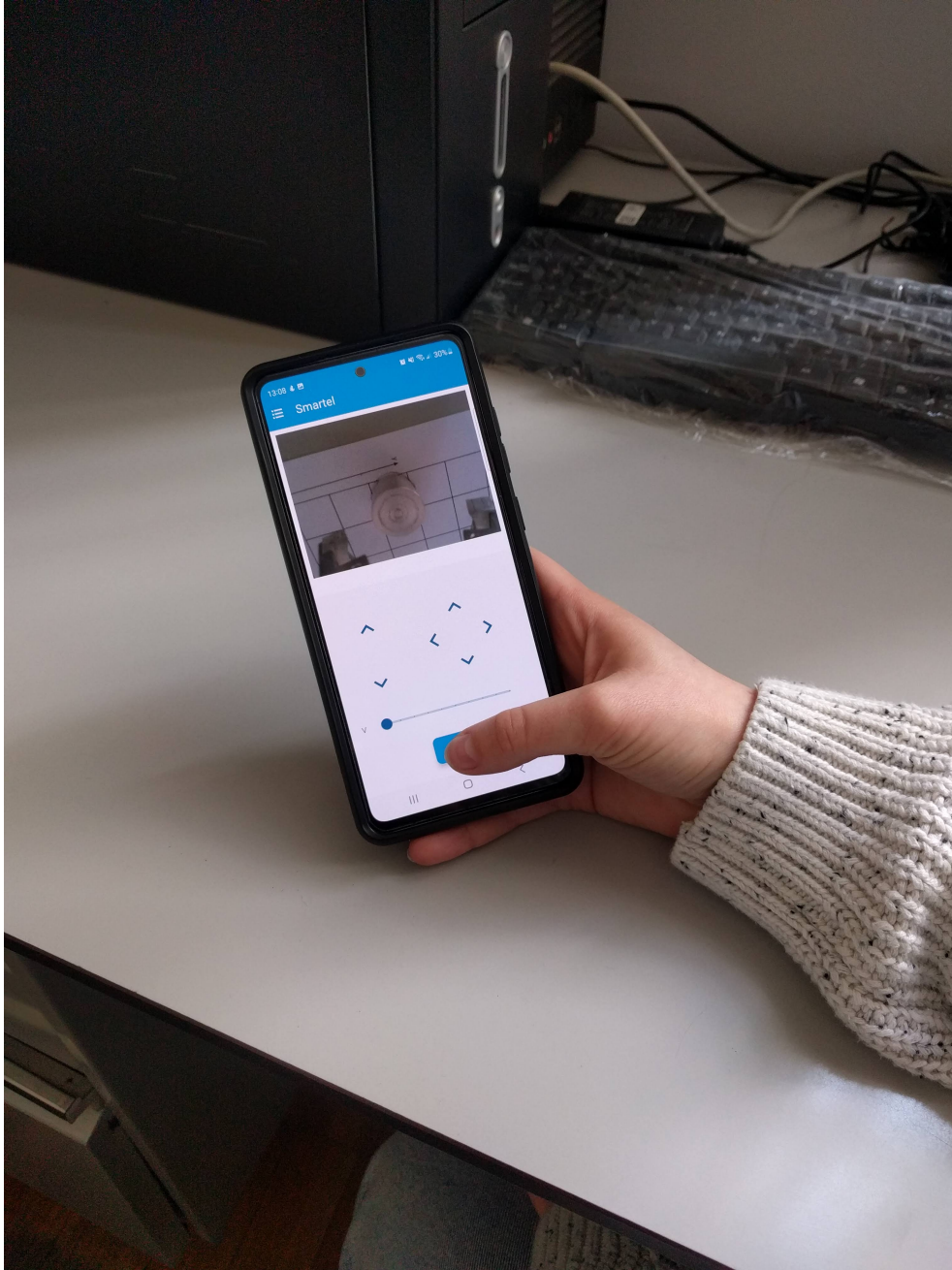
Függelék



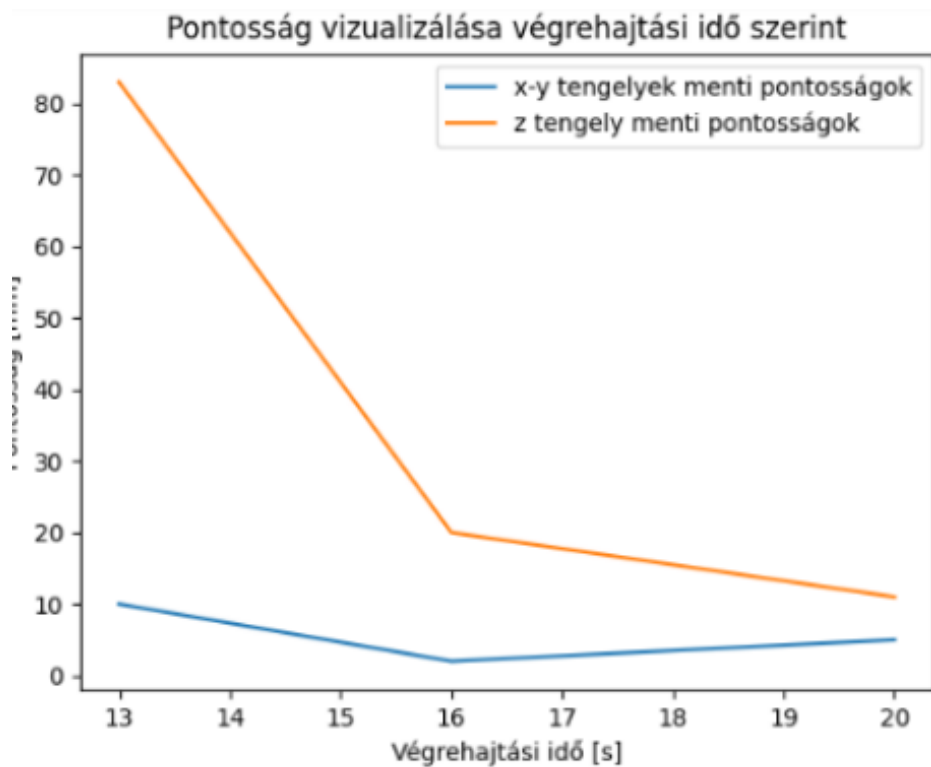
F.0.1. ábra. Manuális vezérlést segítő lap



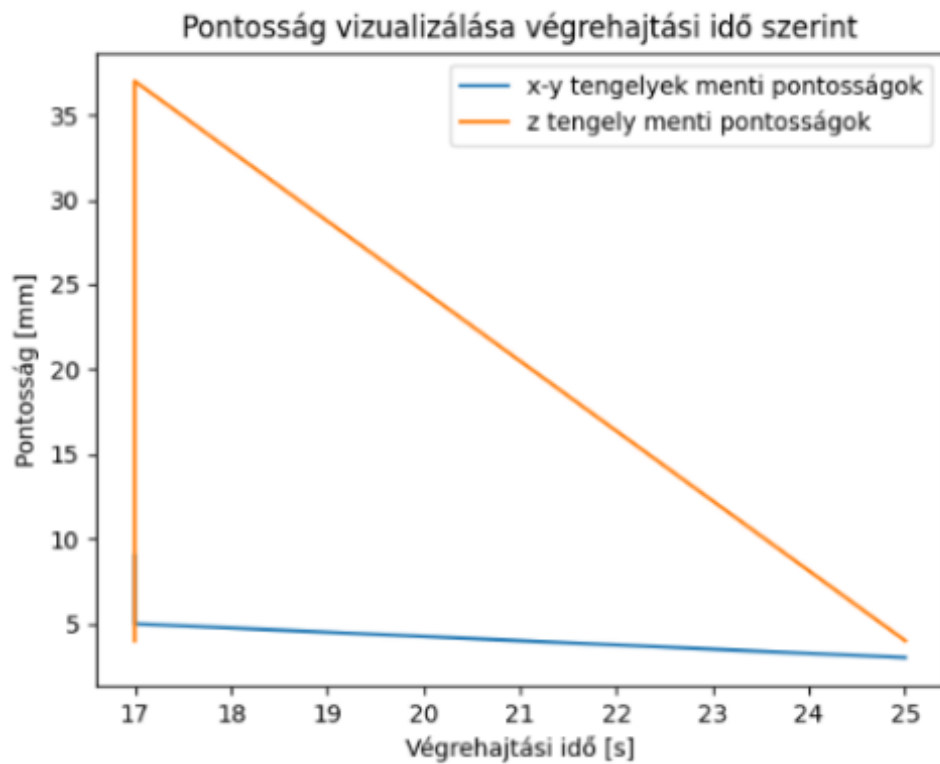
F.0.2. ábra. Mérőberendezés.



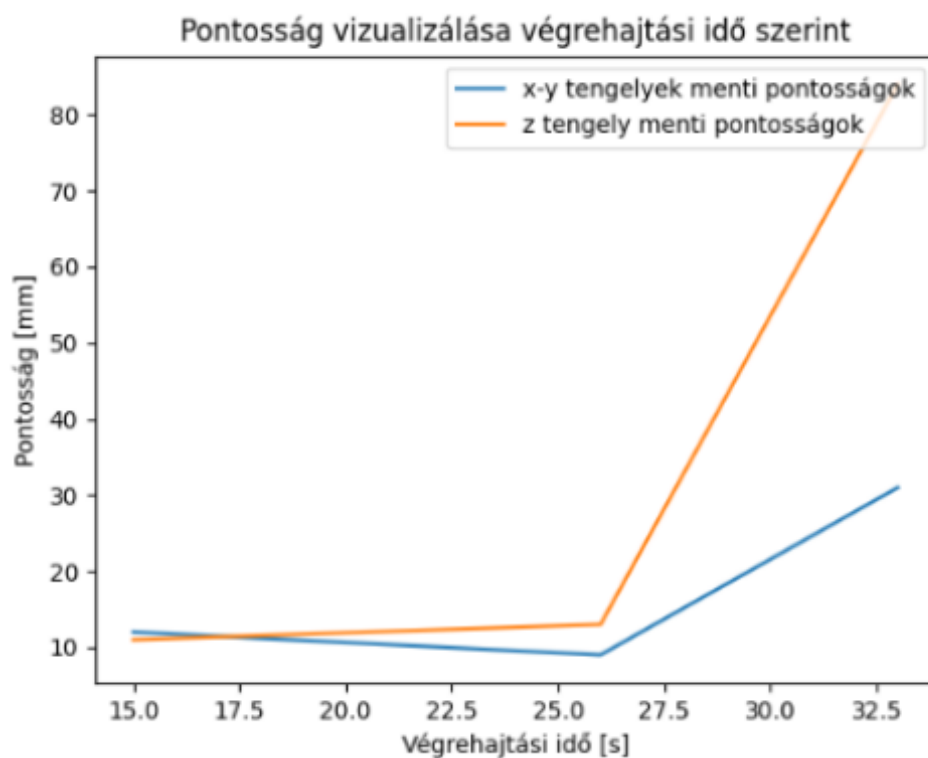
F.0.3. ábra. A mérés menete.



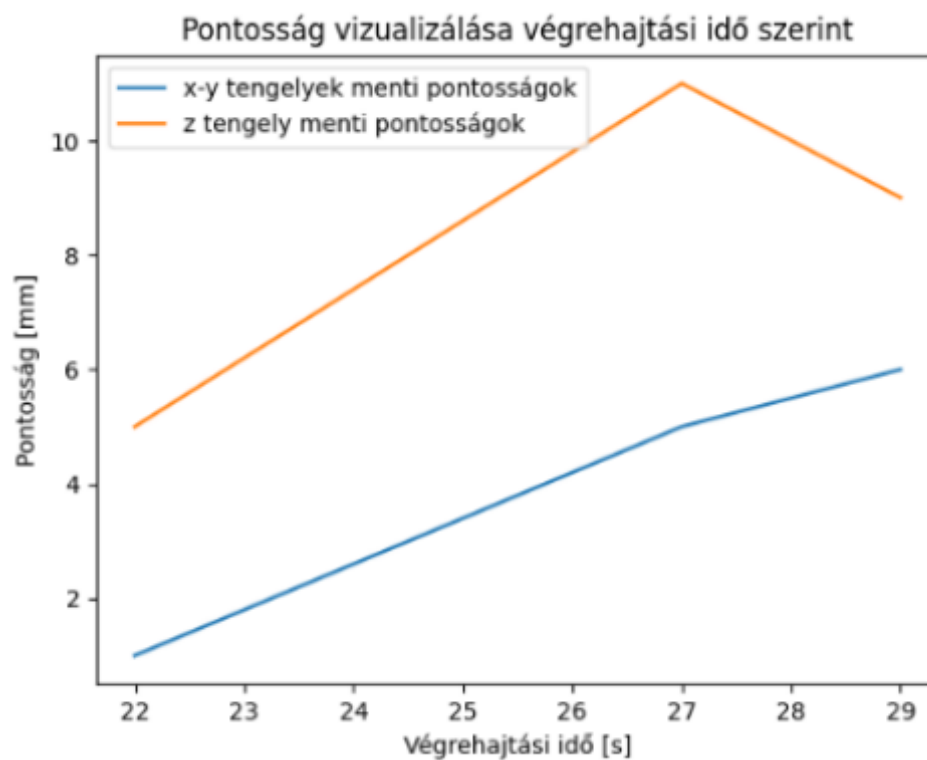
F.0.4. ábra. 1-es felhasználó mérési eredményeinek vizualizálása.



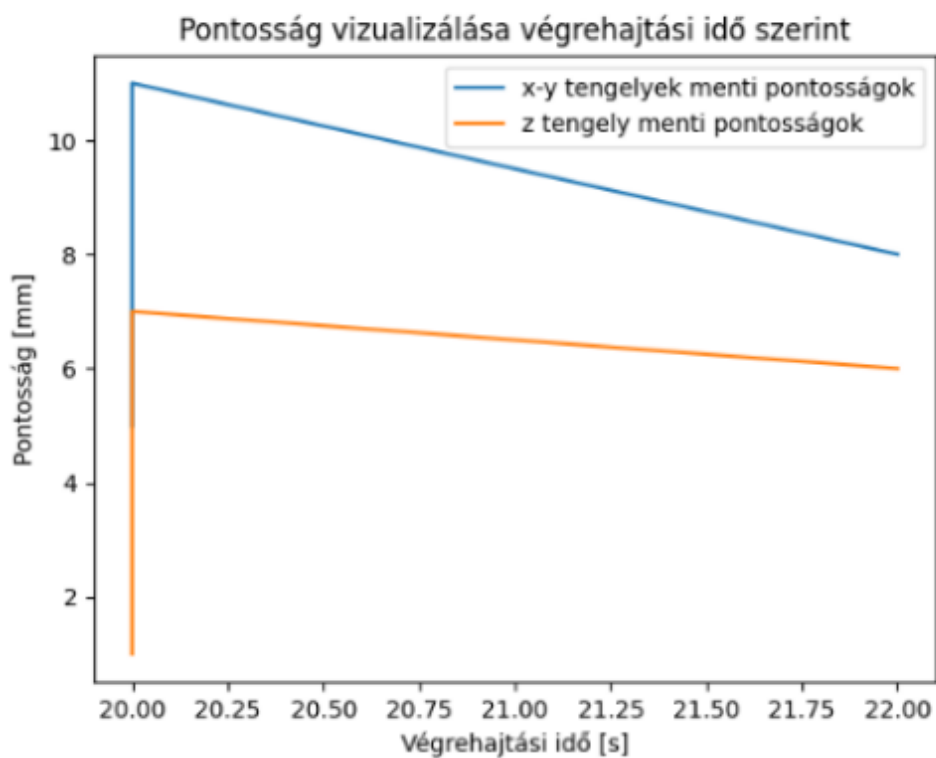
F.0.5. ábra. 2-es felhasználó mérési eredményeinek vizualizálása.



F.0.6. ábra. 3-as felhasználó mérési eredményeinek vizualizálása.



F.0.7. ábra. 4-es felhasználó mérési eredményeinek vizualizálása.



F.0.8. ábra. 5-ös felhasználó mérési eredményeinek vizualizálása.



F.0.9. ábra. A tervezett logó.