Haladási napló

1. hét:

Létrehoztam a github repot

Ismerkedtem a ROS-al, csináltam egy ubuntu VS-t, amire telepítettem is elméletileg, bár még nem látom át teljesen

Kicsit utánajártam, hogy milyen frameworköt érdemes használni egyszerűbb VR alkalmazások fejlesztésére, utánaolvastam a Unitynek, illetve Unreal Engine-nek, végül a Unity mellett döntöttem.

Video streamelésére és akár a ROS futtatására is lehet egy RPi-t használnék, az általam talált információk alapján mindkettőre van lehetőség

2. hét

Telepítettem a unity fejlesztői környezetét

VRTK letöltése, ismerkedés a toolkittel

3. hét

Ismerkedés ROSsal, próbaprogramok írása

UR5 robot package telepítése, szimulátor kifagyásának debuggolása (VM reinstallok, nem megfelelő Ubuntu verzió az adott ROS-hoz)

Specifikáció, feladatleírás, ütemterv elkészítése

4. hét

Specifikáció pontosítása

Kerestem, hogy milyen protokollt lenne érdemes használni a vezérlés és visszairányú adatforgalomra

Ezen cikk alapján (<https://www.confluent.io/blog/kafka-fastest-messaging-system/>) a RabbitMQ a leggyorsabb kis adatmennyiség esetén az általa felsoroltak közül, ami feltétel teljesülni fog szerintem jelen esetben. A listában második helyen lévő eszköz a Kafka, aminek késleltetése 5ms, ami jóval nagyobb, mint az 5g-é (ami potenciálisan 1 ms), viszont jóval több adatot tud átvinni.

Mivel azzal már foglalkoztam, illetve a késleltetése elméletileg kisebb, inkább a RabbitMQ-t választottam

Videó továbbításához sima RTMP streamet ki kell próbálni, hogy elég gyors-e, ha nem megfelelő, meg kell nézni az ffmpeg által nyújtott egyéb lehetőségeket

További kutatásom eredményeként megállapítottam, hogy az rtmp nagyon lassú lesz (3-30 s) így érdemesebb lenne a RTC, FTL vagy SRT protokollok egyikét használni inkább.

Igyekeztem az UR5 robot moveit konfigurációját életre kelteni, de még csak a panda robotot sikerült mozgatnom, ami a moveit tutorialban szerepel

Dilemmáztam a python és a cpp között, de végül mivel a döntésem után a cpp-t néhány óra alatt sem sikerült működésre bírnom a moveit-tal, viszont a python szinte azonnal működött és egyébként is bizonytalan voltam, hogy melyik legyen, áttértem a pythonra.

5. hét

Működésre bírtam pythonból az UR5 kart a szimulátorral, meg is mozgattam. Ehhez készítettem egy MoveIt configot az assistanttal.

6. hét

Egyeztettem Ákossal és megismerkedtem a robotkarral, illetve a laborban használt ROS konfigurációval

A megismert config alapján itthon is összeraktam a gazebo szimulátorral is a működést egy új teszt projektben.

Elkezdtem foglalkozni a projekt kommunikációs részével is, kipróbáltam a RabbitMQ-t pythonnal, illetve elkezdtem gondolkodni a kommunikációs csatornákon.

7. hét

A korábban kitalált kommunikációs utak alapján elkezdtem megvalósítani a kommunikációt ROS-on kívülről

sikerült odáig eljutni, hogy kívülről érkező parancs hatására a robot megmozdult a szimulátorban

8. hét

Beszéltem Lacival a VR setuppal kapcsolatban, arra a megállapodásra jutottunk, hogy előbb készítsem el az alkalmazást alap 3D-s unityben és ha elkészült, akkor könnyen át lehet ültetni VR headsetre. Ezen az úton indultam el utána.

Unityvel való ismerkedés, platform tanulása

RTC videótovábbítással való ismerkedés

9-11. hét

Unity videófogadással való kísérletezés, RTC-n való videótovábbítás kipróbálása

Rengeteg idő megy el a különböző python verziók által okozott kompatibilitási problémákkal

Elkezdtem átlátni a WebRTC működését, félig elkészült egy teszt kommunikációs projekt a Unity és a virtuális gépen futó RTC python script között, viszont sok idő elment arra, hogy rájöjjek milyen library-k működnek, mit mivel érdemes használni.

Időközben eljátszottam kicsit a gondolattal, hogy NDI-t használjak a videó átvitelére, hiszen nagyon jó minőségben tud átvinni videót minimális késleltetéssel, de végül felesleges overheadnek minősítettem és maradtam az RTC-nél.

A sok nem működő kód után elkezdtem csökkenteni az összetettségét a rendszernek teszt célokból és átálltam arra, hogy eleinte csak egy böngészőbe tudjak videót streamelni RTC-n, ez nem sokkal az átállás után sikerült is pythonból. Utána kezdtem el átültetni Unity-be az RTC videó fogadó felét, ami nem sikerült még sajnos. Valószínűleg ez a meglehetősen kezdetleges Unity tudásomból ered, még az sem teljesen tiszta, hogy pontosan milyen komponenshez kéne rendelni azt a kódot, ami a kommunikációért felel. Viszonylag sokat böngésztem példakódokat amelyek a Unity-WebRTC package-hez tartoznak, illetve más szemszögből kicsit próbáltam jobban megérteni az RTC-n alapuló kommunikáció felépítését, ami még mindig nem teljesen tiszta.

12. hét

Ebben az időszakban is még sajnos az RTC kommunikációt igyekeztem működésre bírni a ROS mellett futó python kliens és a Unity-ben futó felhasználói kliens között. Mostanra eljutottam már oda, hogy azt hiszem tényleg látom hogy kéne felépüljön a kapcsolat, viszont ténylegesen nem sikerült még megvalósítani sajnos.

Olyan problémákba is ütköztem folyamatosan, amikre nem számítottam, eleinte a RabbitMQ kommunikáció legalapvetőbb funkcióihoz tartozó függvénynek átadott paraméterek voltak hibásak, mint később kiderült amiatt, mert nem volt egy extra könyvtár telepítve, ami szükséges lett volna a Rabbit-et implementáló könyvtárnak. Mindez azért volt lehetséges, mert a nuget package-ket nem lehet csak úgy könnyen telepíteni Unity-ben, mint azt gondoltam, ennek megkerülése érdekében külön letöltöttem a csomagot és úgy importáltam a projektbe, viszont ez nem húzta magával a saját dependenciáit. A megoldást végül az jelentette, hogy találtam egy github repo-t, ami azt állította magáról, hogy AMQP implementáció és működik RabbitMQ-val Unityben, és ezt telepítettem végül. De ezzel sem volt vége az akadályoknak, mert valamiért a Rabbit csak a futtatás elején érzékeli a beérkező üzeneteket, ha indítás után küldök, azt nem észleli. Ezt a hibát próbáltam úgy áthidalni, hogy a Unity Update függvényébe teszem a Consume függvényt, azonban az hibát dobott futás közben és még indításkor sem kapta el az üzeneteket.

A Rabbittel való próbálkozásom lényege az lett volna (első körben), hogy összehozzam az RTC-s kommunikáció két felét kapcsolati szempontból, azonban sajnos ez sem sikerült a gondok miatt. Próbáltam manuálisan is továbbítani a csatlakozáshoz szükséges adatokat a két fél között, viszont ez már ötletnek sem volt jó.

A haladással való lemaradásom aggaszt, viszont sajnos egyéb feladatok és a megütközésekből való kedvtelenségem miatt nem tudtam sajnos ezen a héten sem jobban behozni. Remélem a következő hét változásokat hoz.

13. hét

A WebRTC hosszú sikertelensége után áttérést kezdtem RTP-re, amit viszonylag gyorsan sikerült működésre bírnom a virtuális ubuntu és a gazdagépen futó VLC között. Mivel a VLC-nek van Unity pluginje, ezért bizakodó vagyok.

Az RTP működésének másnapján reggel már nem működött a dolog, illetve egy virtuális gép restart után nem jelent meg semmilyen interaktálható grafikai elem az Ubuntu UI-on, amit egész problémás volt visszaszerezni.

A VLC Unity alá készült libVLC implementációjának segítségével sikerült rtp streamet megjeleníteni a virtuális gépről.