

Integracija kamere i virtualne stvarnosti s robotskom rukom

Haraminčić, Fran

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: <https://urn.nsk.hr/urn:nbn:hr:235:674488>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2024-12-19**

Repository / Repozitorij:

[Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb](#)



SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Fran Haraminčić

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

Izv. prof. dr. sc. Tomislav Stipančić

Student:

Fran Haraminčić

Zagreb, 2024.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se profesoru Stipančiću i asistentu Korenu na pomoći oko izrade završnog rada. Također, zahvaljujem se obitelji na podršci.

Fran Haraminčić



SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija mehatronika i robotika



Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: 602 – 04 / 24 – 06 / 1	
Ur.broj: 15 – 24 –	

ZAVRŠNI ZADATAK

Student: **Fran Haraminčić**

JMBAG: **0035239741**

Naslov rada na hrvatskom jeziku: **Integracija kamere i virtualne stvarnosti s robotskom rukom**

Naslov rada na engleskom jeziku: **Integration of a camera and virtual reality with a robotic arm**

Opis zadatka:

Kao spona između fizičkog, virtualnog i proširenog svijeta koriste se različita sučelja u vidu ekrana i naočala za virtualnu ili proširenu stvarnost. Koristeći robote, softverski agenti iz digitalnog svijeta mogu činiti promjene u sklopu realne okoline.

Kroz rad treba razviti sustav koji integrira kameru montiranu na robotsku ruku s VR (engl. Virtual Reality) naočalama kako bi korisnik mogao upravljati pokretima robota i vizualno pratiti okolinu robota u realnom vremenu. Rad će istražiti mogućnost sinkronizacije pokreta glave korisnika s pokretima robota.

U radu je potrebno:

- ispitati tehničke specifikacije te ostvariti kompatibilnost između VR naočala, kamere i robota
- razviti softver koji će omogućiti komunikaciju i kontrolu između kamere, VR naočala te robotske ruke na temelju modela koji objedinjuje obradu slike te prijenos videa u realnom vremenu
- povezati i eksperimentalno evaluirati sve komponente sustava uključujući robotsku ruku, VR naočale i kameru u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava.

U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:

24. 4. 2024.

Zadatak zadao:

izv. prof. dr. sc. Tomislav Stipančić

Datum predaje rada:

2. rok (izvanredni): 11. 7. 2024.
3. rok: 19. i 20. 9. 2024.

Predviđeni datumi obrane:

2. rok (izvanredni): 15. 7. 2024.
3. rok: 23. 9. – 27. 9. 2024.

Predsjednik Povjerenstva:

izv. prof. dr. sc. Petar Čurković

SADRŽAJ

SADRŽAJ	I
POPIS SLIKA	II
POPIS OZNAKA	III
SAŽETAK.....	IV
SUMMARY	V
1. UVOD.....	1
2. KORIŠTENE KOMPONENTE I SOFTVER	2
2.1. LOGITECH KAMERA	2
2.2. UNREAL ENGINE.....	2
2.3. VIVE COSMOS ELITE UREĐAJ ZA VIRTUALNU STVARNOST	3
2.4. NDI SOFTVER.....	3
2.5. PYTHON	3
2.5.1. Korištene biblioteke	3
2.5.1.1. Socket biblioteka.....	4
2.5.1.2. Time biblioteka	4
2.5.1.3. Re biblioteka	4
2.5.1.4. Robolink biblioteka.....	5
2.6. UR5 ROBOTSKA RUKA	5
2.7. ROBODK.....	5
3. STVARANJE PROJEKTA U UNREAL ENGINE-U	7
4. POVEZIVANJE KAMERE S UNREAL ENGINE-OM	11
4.1. NDI ALATI.....	12
4.2. NDIIO PLUGIN U UNREAL ENGINE-U	13
4.3. NDIIO U UNREAL ENGINE PROJEKTU	14
4.3.1. NDI materijal	15
4.3.2. NDIREceiver_BP	16
5. UDP SOCKET SERVER I KLIJENT	17
5.1. UDP KLIJENT U UNREAL ENGINE-U	18
5.2. UDP SERVER	22
6. STVARANJE PROJEKTA U ROBODK.....	23
6.1. ORIENTATION_ROUTINE SKRIPTA	24
6.1.1. Dodavanje Robolink biblioteke	24
6.2. LOGIKA UPRAVLJANJA ROBOTSKOM RUKOM.....	25
7. POVEZIVANJE S UR5 ROBOTSKOM RUKOM.....	27
8. TESTIRANJE	28
9. ZAKLJUČAK.....	29
LITERATURA.....	30
PRILOZI.....	32

POPIS SLIKA

Slika 1	Logitech StreamCam	2
Slika 2	Python sintaksa za UDP protokol.....	4
Slika 3	UR5 robotska ruka.....	5
Slika 4	Stvaranje projekta.....	7
Slika 5	Unreal Engine projekt	7
Slika 6	Preglednik stavki	8
Slika 7	<i>Lock to HMD</i>	8
Slika 8	Player datoteka	9
Slika 9	Player_view <i>Blueprint</i>	9
Slika 10	Ravnina i <i>CineCameraActor</i>	10
Slika 11	NDIIO plugin datoteka.....	11
Slika 12	Plugins datoteka	11
Slika 13	Plugins datoteka projekta	12
Slika 14	NDI alati	12
Slika 15	Video prikaz StreamCam kamere.....	13
Slika 16	NDIIO plugin	13
Slika 17	<i>NDIReceiver</i> stavka	14
Slika 18	<i>NDIReceiver</i> datoteka	15
Slika 19	NDI materijal.....	16
Slika 20	<i>NDIReceiver_BP Blueprint</i>	16
Slika 21	Primjer rada UDP <i>socket</i> servera.....	17
Slika 22	<i>ObjectDeliverer</i> plugin.....	18
Slika 23	Orientation datoteka	18
Slika 24	<i>Sequence</i> čvor.....	19
Slika 25	<i>Deliverer Manager</i> skup čvorova.....	19
Slika 26	<i>Delivery Box</i> skup čvorova.....	20
Slika 27	<i>Start UDP socket</i> skup čvorova.....	21
Slika 28	Skup čvorova za zatvaranje UDP servera	21
Slika 29	Skup čvorova za dohvaćanje i slanje podataka	22
Slika 30	Inicijalizacija UDP servera.....	22
Slika 31	RoboDK projekt	23
Slika 32	<i>Python interpreter</i>	24
Slika 33	Povezivanje s robotskom rukom	25
Slika 34	Rutina	26
Slika 35	Povezivanje s UR5 robotskom rukom.....	27
Slika 36	Testiranje	28
Slika 37	Završni rezultat.....	28

POPIS OZNAKA

Oznaka	Opis
HD	Visoka definicija (engl. High definition)
FPS	Slike po sekundi (engl. Frames per second)
VR	Virtualna stvarnost (engl. Virtual reality)
AR	Proširena stvarnost (engl. Augmented reality)
NDI	Engl. Network Device Interface
UDP	Engl. User Datagram Protocol
IP	Engl. Internet Protocol
JSON	Engl. JavaScript Object Notation
UTF-8	Engl. Unicode Transformation Format – 8-bit
ASCII	Engl. American Standard Code for Information Interchange

SAŽETAK

U radu se koristi program za stvaranje virtualnog prostora zajedno sa aparatom za virtualnu stvarnost. Koristi se visoko kvalitetna kamera koja se preko programa za prijenos video prikaza prenosi u program u kojem se nalazi virtualni prostor. Kamera se montira na robotsku ruku koja se upravlja koristeći programski jezik.

Ključne riječi: virtualna stvarnost, aparat za virtualnu stvarnost, kamera, robotska ruka, programski jezik

SUMMARY

This paper utilizes a program for creating a virtual space and a headset for accessing that space. The feed from a high-quality camera is transmitted into the program which was used to create the virtual space by using a program for video streaming. The camera is attached to a robotic arm which is controlled by a programming language.

Key words: virtual reality, headset, camera, robotic arm, programming language

1. UVOD

U današnje vrijeme, tehnologija ubrzano napreduje i omogućuje nam integraciju fizičkog, virtualnog i proširenog svijeta na načine koji su nekada bili nezamislivi. Kroz razvoj sučelja poput ekrana i naočala za virtualnu ili proširenu stvarnost, stvorena je mogućnost interakcije s digitalnim svijetom na intuitivan i interaktivan način. Korištenje robota i softverskih agenata za povezivanje tih svjetova otvara nova područja primjene u industriji, medicini, obrazovanju i svakodnevnom životu. Jedan od inspirativnih primjera u ovom području je OriHime robot, dizajniran s ciljem omogućavanja socijalne inkluzije osobama s fizičkim invaliditetom. OriHime robot, razvijen od strane japanske kompanije OryLab, kreiran je kako bi omogućio ljudima s teškim tjelesnim oštećenjima da sudjeluju u društvenim aktivnostima, komuniciraju s voljenima i čak rade na daljinu. Ovaj robot funkcionira kao fizička prisutnost korisnika, omogućujući im da putem daljinskog upravljanja komuniciraju, promatraju i sudjeluju u raznim aktivnostima. Vođen ovom inspiracijom, cilj ovog rada je razviti sustav koji integrira VR tehnologiju s robotikom, omogućujući korisnicima da upravljaju pokretima robota i vizualno prate njegovu okolinu u stvarnom vremenu. Konkretno, ovaj rad istražuje mogućnost sinkronizacije pokreta glave korisnika, promatranih putem VR naočala, s pokretima robota, čime se omogućuje intuitivno i prirodno upravljanje robotom. Ovaj projekt predstavlja korak naprijed u stvaranju tehnoloških rješenja koja povezuju digitalni i fizički svijet, stvarajući pritom nove prilike za socijalnu uključenost, komunikaciju i suradnju, posebno za one kojima je tradicionalni oblik interakcije otežan.

2. KORISTENE KOMPONENTE I SOFTVER

U ovom radu je potrebno nekoliko komponenti kako bi se ostvarila željena funkcionalnost. Prvo je potrebno odabrati kameru čiji će se videosignal prenositi u softver s kojim će se napraviti virtualni prostor. Zatim je potrebno odabrati uređaj koji omogućuje pogled u taj virtualni prostor te omogućuje praćenje pokreta korisnikove glave. Naposljetku, potrebno je odabrati robotsku ruku koja će pomicati kameru sukladno s pokretima korisnikove glave. Također je potrebno odabrati programski jezik s kojim se mogu sve gore navedene komponente povezati.

2.1. LOGITECH KAMERA

Za kameru je odabrana StreamCam kamera proizvedena od strane kompanije Logitech [Slika 1]. StreamCam kamera uživo prenosi video u visokoj definiciji(HD) pri rezoluciji od šezdeset slika u sekundi(FPS). Njezine leće napravljene od visoko kvalitetnog stakla zajedno s pametnim automatskim fokusom omogućavaju kvalitetan video prikaz. Iz gore navedenih razloga je ova kamera odlična za ostvarivanje željene funkcionalnosti u ovom radu. [1]



Slika 1 Logitech StreamCam

2.2. UNREAL ENGINE

Unreal Engine je napredan grafički alat za razvoj video igara i interaktivnih aplikacija, kojeg je razvila kompanija Epic Games. Služi za stvaranje visokokvalitetnih trodimenzionalnih svjetova i simulacija te se koristi u industrijama kao što su videoigre, film, arhitektura i virtualna stvarnost(VR). Unreal Engine se često koristi za razvoj virtualne stvarnosti(VR) zbog svoje sposobnosti da generira realistične grafike i omogućuje interakcije u stvarnom vremenu. Platformski je neovisan, što znači da podržava razvoj za računala, konzole, mobilne uređaje, proširenu stvarnost(AR) i VR. Njegov vizualni alat *Blueprints* omogućuje jednostavno

stvaranje prototipa dok brojni plugin-ovi proširuju funkcionalnost, čineći ga prigodnim za ovaj rad. [2]

2.3. VIVE COSMOS ELITE UREĐAJ ZA VIRTUALNU STVARNOST

Vive Cosmos Elite je napredni VR uređaj razvijen od strane kompanije HTC, dizajniran je za visokokvalitetno iskustvo VR-a. Uređaj ima dva ekrana s visokom rezolucijom, što pruža detaljnu sliku te visoka stopa osvježavanja čine VR iskustvo glatkim i ugodnim. [3]

2.4. NDI SOFTVER

Network Device Interface(NDI) je protokol koji je razvila kompanija NewTek za prijenos video signala putem mreže u visokoj kvaliteti i s niskom latencijom. Omogućuje dijeljenje medijskih izvora između različitih uređaja putem lokalne mreže, bez potrebe za dodatnim hardverom. NewTek je također razvila NDI plugin za Unreal Engine koji omogućuje integraciju NDI tehnologije u igre i simulacije. Drugim riječima ovaj plugin omogućuje laku implementaciju video prikaza iz drugih uređaja ili softverskih izvora u Unreal Engine. To je posebno korisno u ovom radu jer omogućuje implementaciju video prikaza StreamCam kamere u Unreal Engine. [4]

2.5. PYTHON

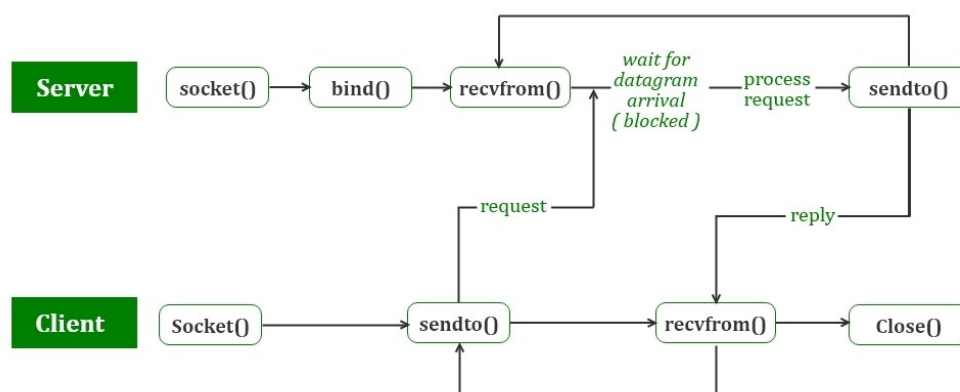
Python je interpretirani, objektno orijentirani programski jezik visoke razine s dinamičkom semantikom. Njegove ugrađene strukture podataka visoke razine, u kombinaciji s dinamičkim tipiziranjem i dinamičkim vezanjem, čine ga vrlo privlačnim za brzi razvoj aplikacija, kao i za korištenje kao jezik za povezivanje postojećih komponenti. Python podržava module i pakete, što potiče modularnost programa te širok raspon implementacije u različitim projektima. Također, budući da nema koraka kompilacije proces uređivanja, testiranja i otklanjanja grešaka je nevjerovatno brz. [5]

2.5.1. Korištene biblioteke

U ovom radu su korištene različite Python-ove biblioteke u cilju ostvarivanja željene funkcionalnosti.

2.5.1.1. Socket biblioteka

Socket biblioteka omogućuje slanje i primanje podataka preko mreže. Općenito, *socket* je kranja točka za komunikaciju između dva uređaja preko mreže. Biblioteka se koristi za slanje i primanje podataka putem *User Datagram Protocol*(UDP). UDP je komunikacijski protokol za primjene u vremenski osjetljivim situacijama poput videoigara, dijeljenja videa ili u ovom slučaju prijenosa podataka iz Unreal Engine-a. UDP je jedan od brzih komunikacijskih protokola zato što ne provjerava ispravnost paketa koji se šalju preko mreže. Za ovaj rad je UDP protokol savršen jer se preko mreže šalje veliki broj podataka pri jako velikim brzinama te nije potrebno provjeravati ispravnost paketa niti je li se neki paket izgubio pri slanju [Slika 2].



Slika 2 Python sintaksa za UDP protokol

2.5.1.2. Time biblioteka

Time biblioteka pruža razne funkcije za rad s vremenom. Može se koristiti za dobivanje trenutnog vremena, formatiranje datuma te za umetanje pauza u izvršavanje programa pomoću funkcije *sleep()* koja se u ovom radu koristi kako bi se osiguralo dovoljno vremena robotu da se poveže s računalom.

2.5.1.3. Re biblioteka

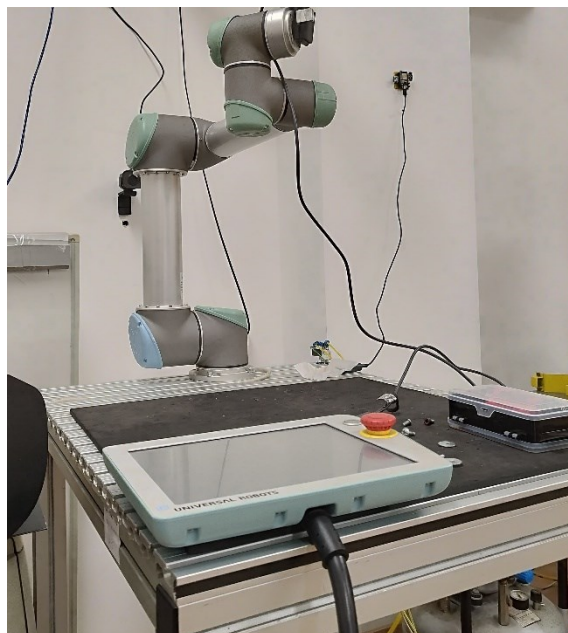
Re biblioteka omogućuje rad s regularnim izrazima, koji služe za pretraživanje i manipulaciju tekстом. U ovom radu se ova biblioteka koristi kako bi se otklonili neželjeni simboli iz podataka koji se šalju preko *Socket* servera.

2.5.1.4. Robolink biblioteka

Ova biblioteka je dio RoboDK softvera, koja omogućuje kontrolu i simulaciju industrijskih robota. *Robolink* omogućuje povezivanje s robotom, simulaciju pokreta i slanje naredbi za izvršavanje zadataka kao što su pozicioniranje robota, kretanje i upravljanje alatima robota.

2.6. UR5 ROBOTSKA RUKA

UR5 je kolaborativna robotska ruka koju je razvila kompanija Universal Robots [Slika 3]. Dizajnirana je za izvođenje preciznih zadataka u industrijskoj automatizaciji poput montaže, pakiranja, paletizacije i slično. UR5 je poznata po svojoj fleksibilnosti, sigurnosti i jednostavnosti programiranja. Ima šest stupnjeva slobode gibanja, što joj omogućuje složene pokrete, a podržava različite alate za integraciju, uključujući Python i RoboDK. [6]



Slika 3 UR5 robotska ruka

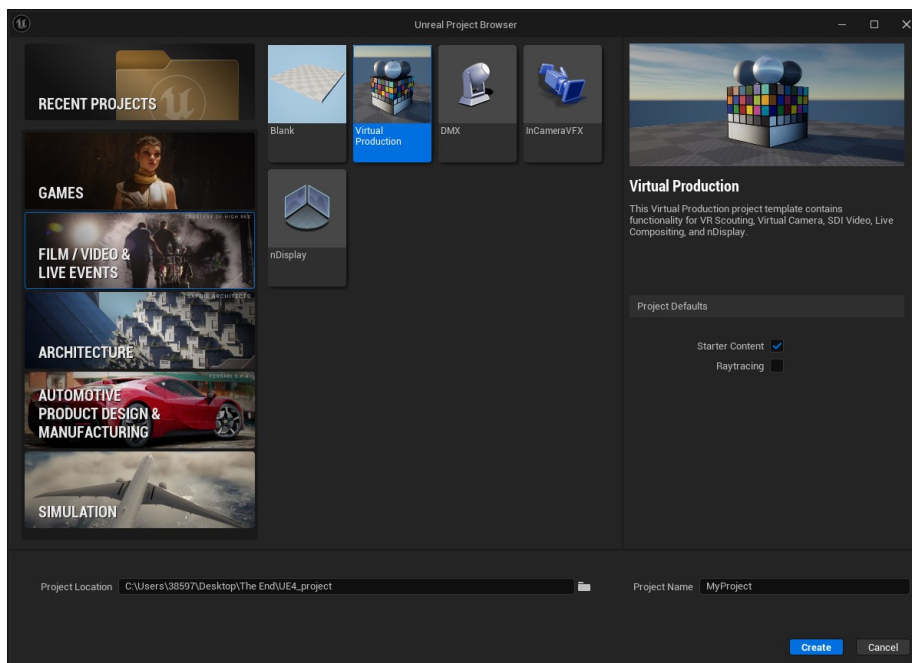
2.7. ROBODK

RoboDK je moćan i isplativ simulacijski alat za industrijske robote i programiranje robota razvijen od strane istoimene podružnice CoRo laboratorija na ETS sveučilištu u Kanadi. Omogućuje lako programiranje robota zahvaljujući svojem intuitivnom sučelju te također omogućuje programiranje robota dok su isključeni. Prednost korištenja RoboDK je to što on omogućuje programiranje robota izvan proizvodne okoline te direktnim programiranjem robota s korisnikovog računala smanjuje vrijeme proizvodnog zastoja. [7] RoboDK također ima veliku

biblioteku robota spremnih za uporabu. Iz njihove biblioteke je preuzet model UR5 robotske ruke korištene u ovom radu. [8]

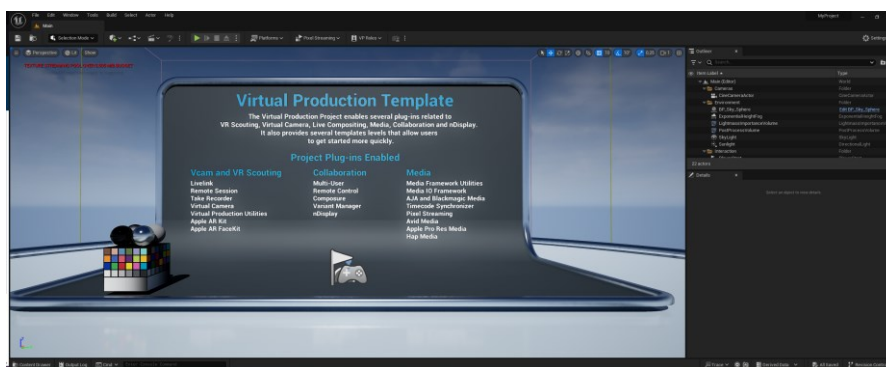
3. STVARANJE PROJEKTA U UNREAL ENGINE-U

Nakon pokretanja Unreal Engine preglednika nudi se opcija za stvaranje projekta. Odabire se projekt za virtualnu produkciju [Slika 4].



Slika 4 Stvaranje projekta

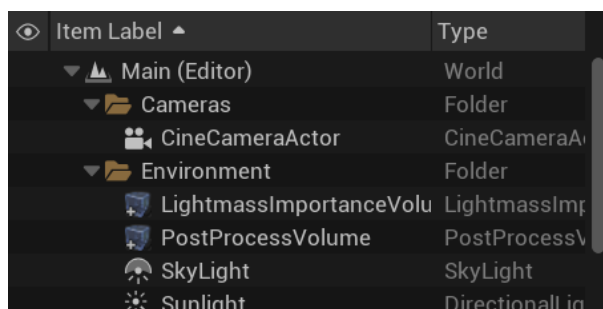
Nakon odabira lokacije na kojoj će se projekt spremiti i imena projekta pritisne se tipka *Create*. Nakon toga Unreal Engine stvori projekt [Slika 5]. Projekt u sebi sadrži razinu *Main* u kojoj se nalaze različite stavke (engl. Actor).



Slika 5 Unreal Engine projekt

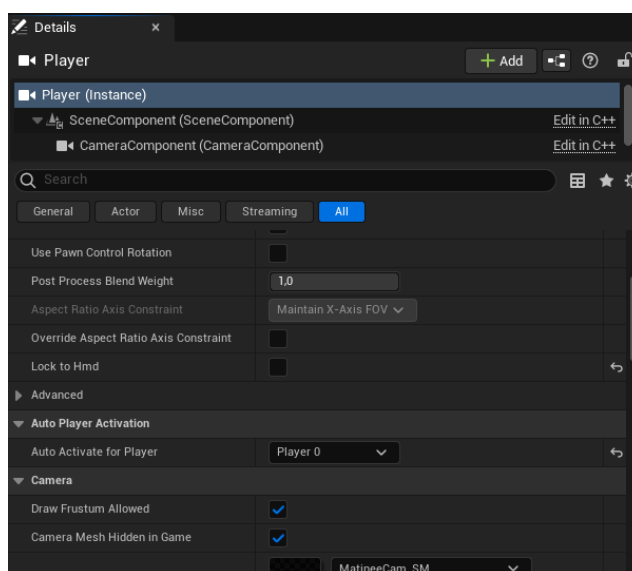
Nakon što je projekt generiran potrebno je obrisati nekoliko stvari kako bi se smanjilo nepotrebno opterećenje na grafičku karticu računala. Iz projekta se briše sve na način da ostanu *CineCameraActor*, *LightmassImportanceVolume*, *PostProcessVolume*, *SkyLight* i *Sunlight*

stavke projekta [Slika 6]. Sve stavke koje se nalaze u datoteci Enviroment služe za osiguravanje potrebnog osvjetljenja i interakcije VR uređaja s projektom.



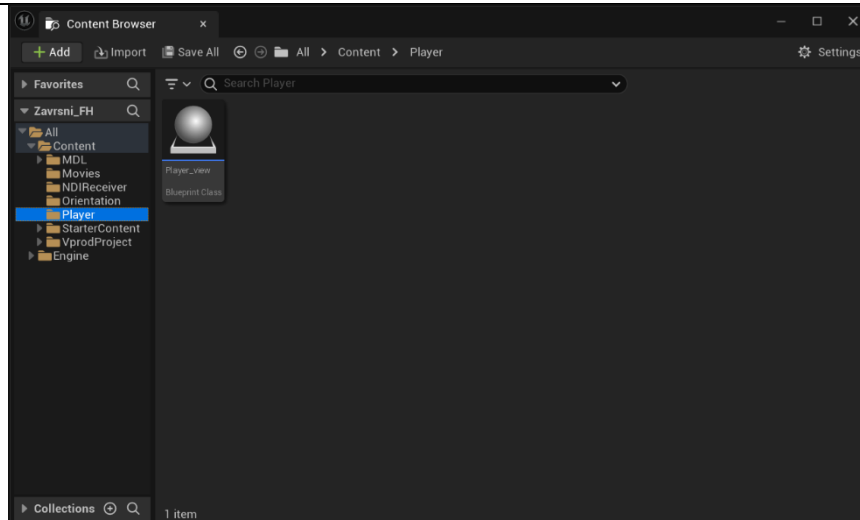
Slika 6 Preglednik stavki

CineCameraActor stavka je Unreal Engine kamera koja će služiti kao početna točka za korisnika kada se pokrene simulacija projekta. Međutim prvo se onemogućuje opcija *Lock to HMD* na stavci kako bi se onemogućilo korisniku skretanje pogleda [Slika 7].



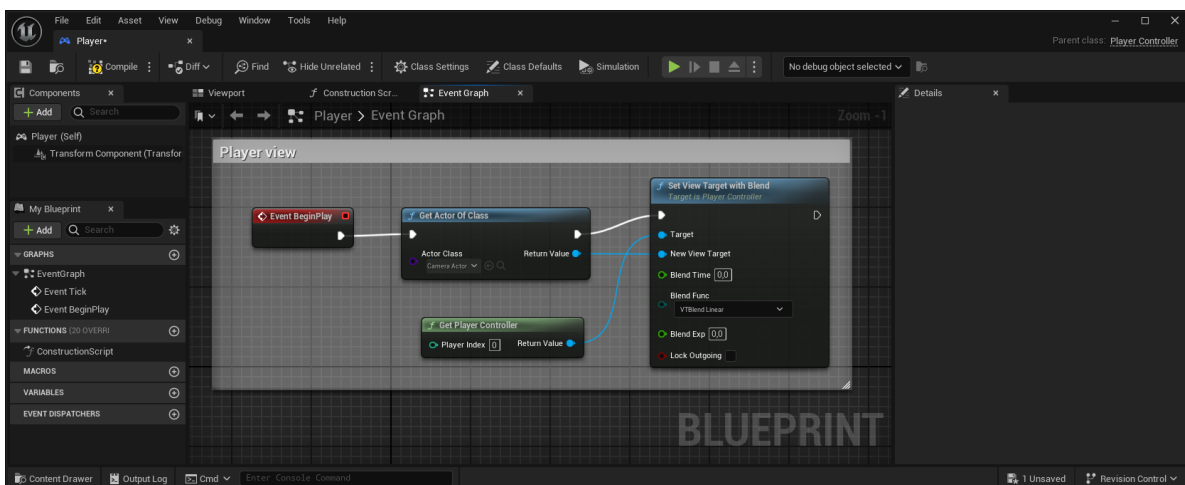
Slika 7 *Lock to HMD*

Sada se *CineCameraActor* neće moći okretati niti pomicati. Nadalje je potrebno povezati pogled korisnika sa stavkom kada se pokrene simulacija projekta. U Unreal Engine-u to se ostvaruje uporabom njegovog ugrađenog vizualnog alata. *Blueprint Visual Scripting system* je sustav za stvaranje programskih skripti korištenjem sučelja baziranog na čvorovima (engl. Node) za stvaranje i manipulaciju elementima i stavkama unutar Unreal Engine-a. Kao i drugi programski jezici koristi se za definiranje objektno orijentiranih klasa ili objekata. Prvo je potrebno napraviti novu datoteku Player u kojoj se kreira nova stavka klase *Blueprint* imena *Player_view* [Slika 8].



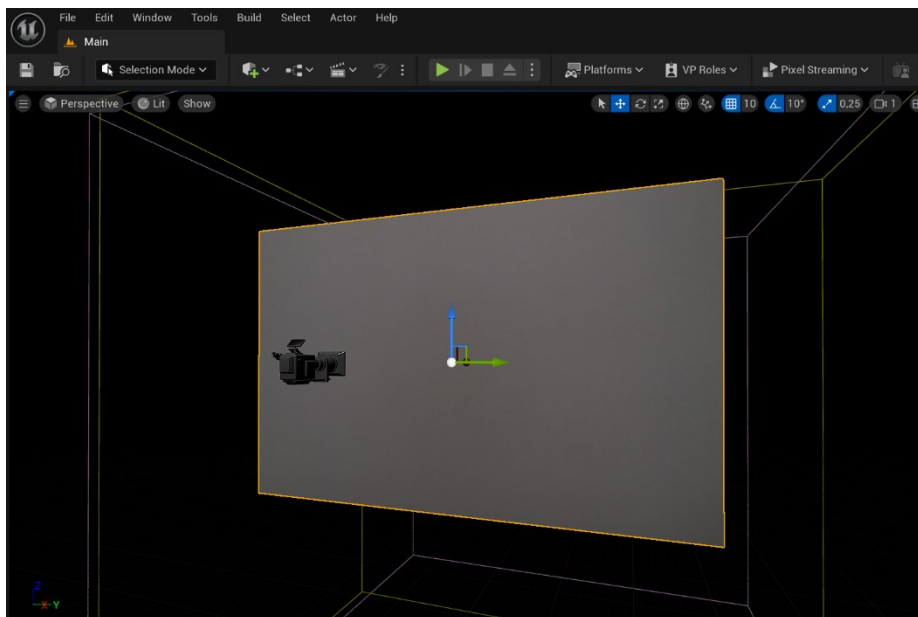
Slika 8 Player datoteka

Kada se otvori *Blueprint* on sadrži tri osnovna čvora koji se uključe pri određenom događaju. Ovdje je potrebno uzeti čvor *Event BeginPlay* koji će se uključiti kada se pokrene simulacija projekta. On se zatim spaja na čvor *Get Actor Of Class* koji ima padajući izbornik u kojem se nalaze klase svake dostupne stavke unutar Unreal Engine-a. Ovdje je potrebno odabrati klasu *Camera Actor* tako da čvor obuhvati *CineCameraActor* stavku. Nadalje se *Get Actor Of Class* čvor povezuje sa *Set View Target with Blend* čvorom. Ovaj čvor ima nekoliko ulaznih parametara, na *New View Target* ulazni parametar se mora povezati izlazni parametar iz *Get Actor Of Class* čvora. Zatim se dodaje novi *Get Player Controller* čvor na kojem se indeks igrača definira kao nula te se time osigura da će čvor uvijek obuhvatiti prvi VR uređaj koji se pojavi pri pokretanju simulacije projekta. Izlazna vrijednost iz *Get Player Controller* čvora se povezuje na *Target* ulazni parametar od *Set View Target with Blend* čvora [Slika 9].



Slika 9 Player_view Blueprint

Ovim *Blueprint*-om se pogled korisnika stavlja na stavku *CineCameraActor* pri pokretanju simulacije projekta. Sada je potrebno dodati površinu na koju će se staviti video prikaz sa StreamCam kamere. U Unreal Engine-u se klikne na alatnu traku i odabire se opcija *Place new actor*. Pod kategorijom *Shapes* se nalazi ravnina (engl. Plane) koja se dodaje u projekt. Normala ravnine gleda u smjeru z osi pa ju je potrebno zarotirati oko y osi u pozitivnome smjeru. Sada normala ravnine gleda u negativnom smjeru x osi, u smjeru *CineCameraActor* stavke [Slika 10].

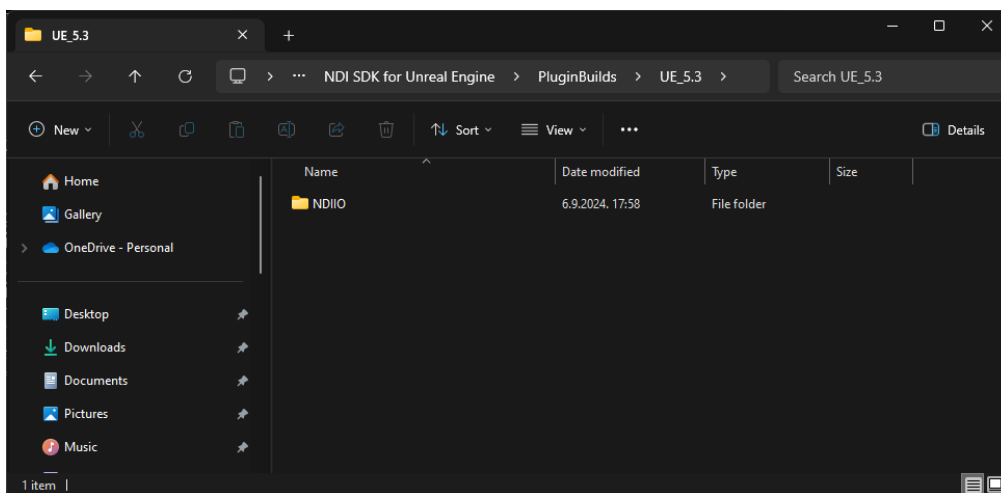


Slika 10 Ravnina i *CineCameraActor*

Sada je projekt spreman za povezivanje StreamCam kamere s Unreal Engine-om.

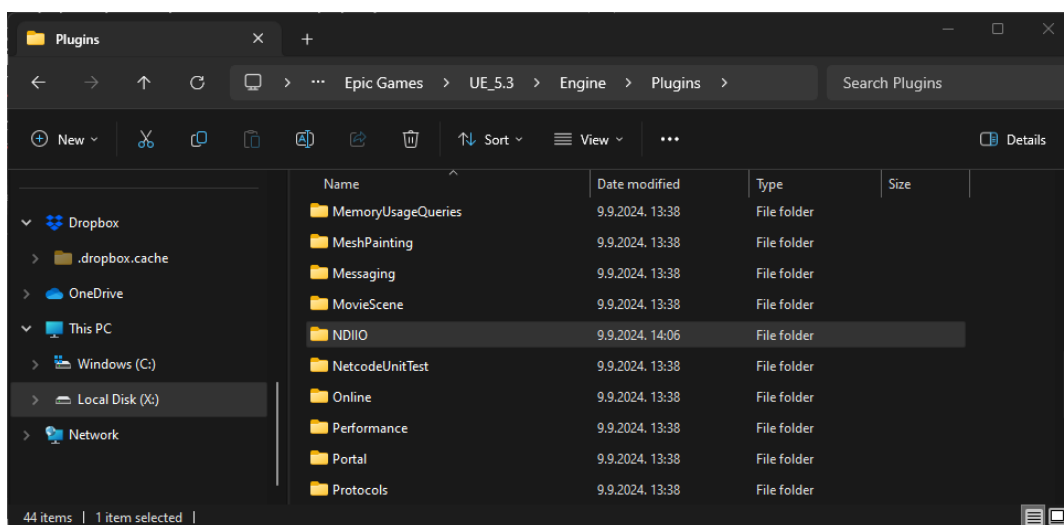
4. POVEZIVANJE KAMERE S UNREAL ENGINE-OM

Za povezivanje kamere se koriste NDI alati i plugin za Unreal Engine. Prvo je potrebno instalirati NDI alate s njihove službene stranice [9] i slijediti upute za instalaciju [10]. Prvo se instalira plugin za Unreal Engine. Prije nego li se plugin može koristiti u Unreal Engine-u potrebno je otvoriti Eksploraer za datoteke i otići na mjesto instalacije NDI plugin-a za Unreal Engine (C:\Program Files\NDI\NDI SDK for Unreal Engine). Nadalje se otvara datoteka PluginBuilds i otvara se datoteka za odgovarajuću verziju Unreal Engine-a (UE_5.3). Odavde se kopira datoteka NDIIO koja u sebi sadrži plugin [Slika 11].



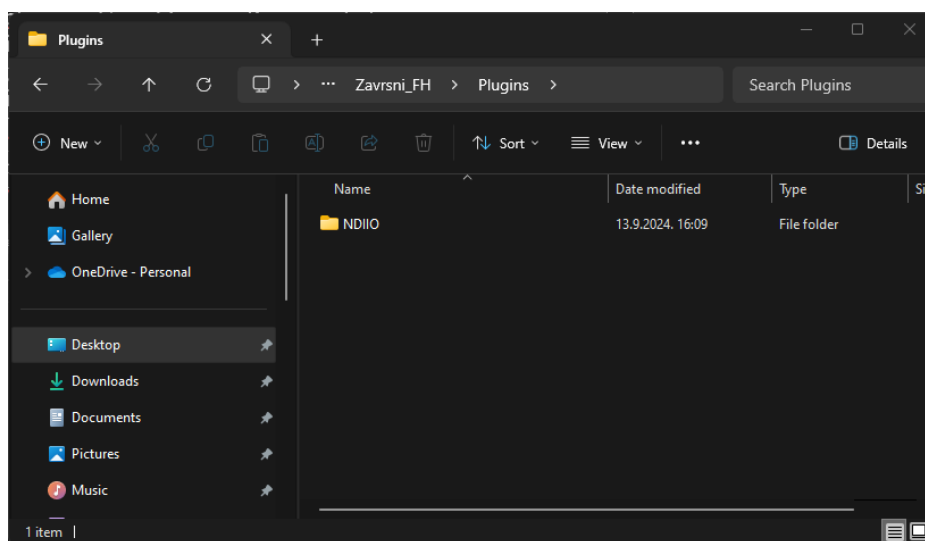
Slika 11 NDIIO plugin datoteka

Ta datoteka se može kopirati ili u datoteku od projekta ili u datoteku od Unreal Engine-a. Ukoliko se kopira u datoteku od Unreal Engine-a potrebno je otići na lokaciju Plugins datoteke [Slika 12]. [11]



Slika 12 Plugins datoteka

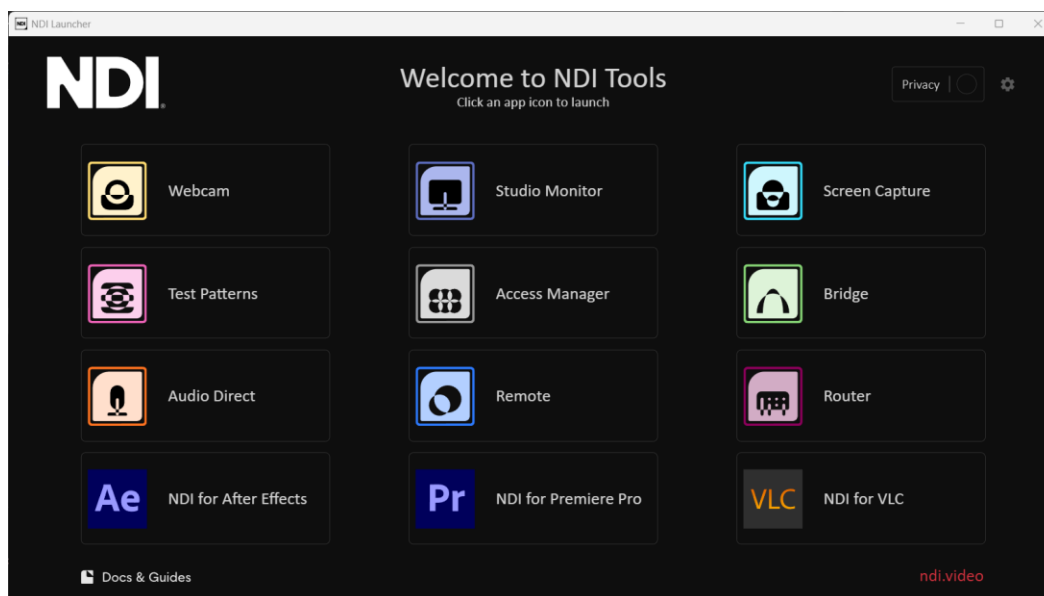
Općenita preporuka je da se kod rada s posebnim plugin-ovima, oni kopiraju u datoteku od projekta. Ovaj rad prati tu preporuku pa je potrebno otići na mjesto gdje je spremljen projekt i napraviti datoteku Plugins te unutar nje kopirati datoteku s plugin-om [Slika 13].



Slika 13 Plugins datoteka projekta

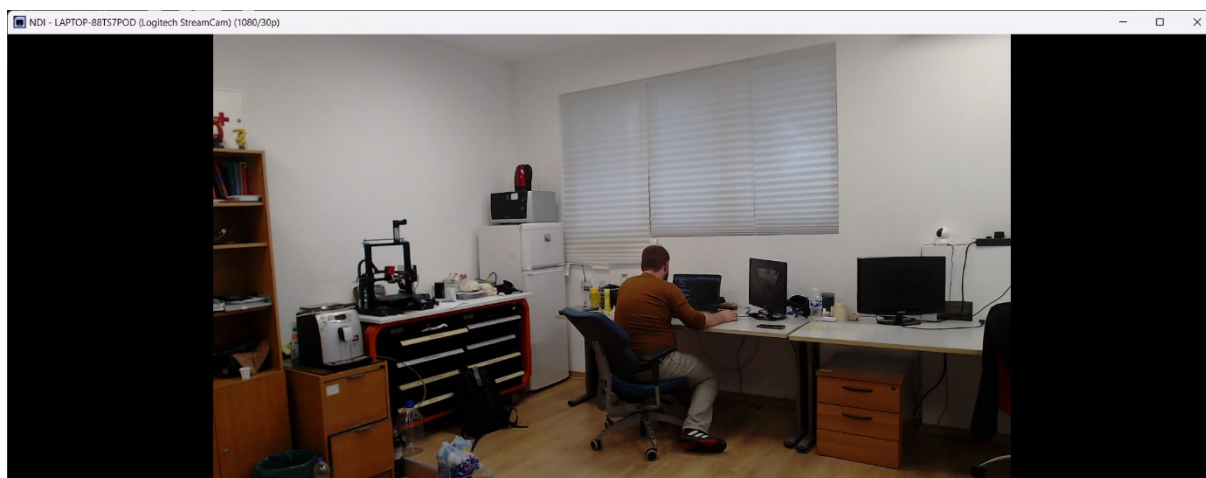
4.1. NDI ALATI

Nakon instalacije NDI alata potrebno ih je pokrenuti. S glavnog izbornika odabire se *Screen Capture* alat koji će omogućiti prikaz videa sa StreamCam kamere [Slika 14].



Slika 14 NDI alati

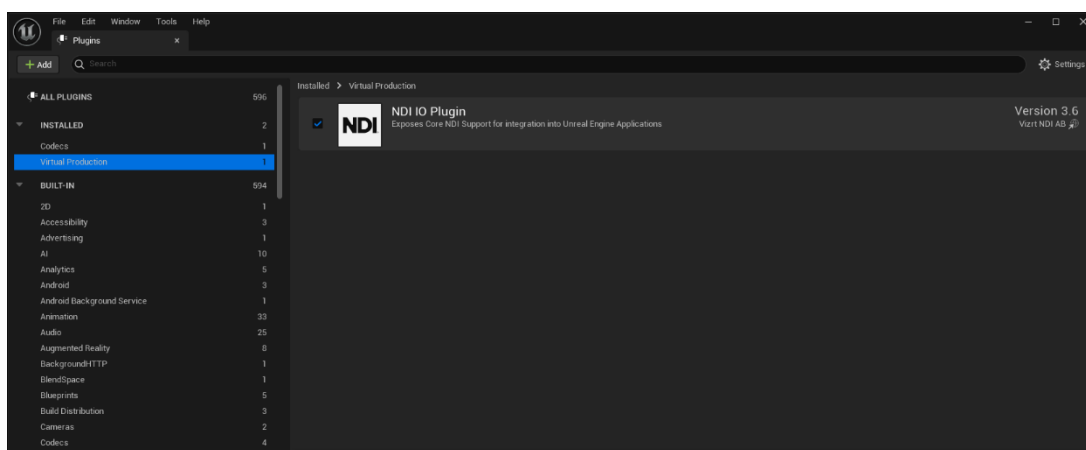
Dok je *Screen Capture* alat upaljen video prikaz s kamere se može vidjeti na *Studio Monitor* alatu. Stoga se s glavnog izbornika otvara *Studio Monitor* alat kako bi se provjerilo je li instalacija bila uspješna i vidi li se video prikaz sa StreamCam kamere [Slika 15].



Slika 15 Video prikaz StreamCam kamere

4.2. NDIIO PLUGIN U UNREAL ENGINE-U

Video prikaz s kamere radi i plugin se nalazi unutar datoteke u kojoj je projekt. Sada se ponovno otvara Unreal Engine i na alatnoj traci se odabire *Edit* te zatim *Plugins*. Otvara se prozor s instaliranim plugin-ovima među kojima se sada i nalazi NDIIO plugin. Međutim prije nego li se može koristiti potrebno ga je aktivirati pritiskom na praznu kućicu. Nakon toga je potrebno ponovno pokrenuti Unreal Engine te opet otvoriti prozor s plugin-ovima. Pored NDIIO plugin-a se nalazi plava kvačica što znači da je plugin aktivan [Slika 16].

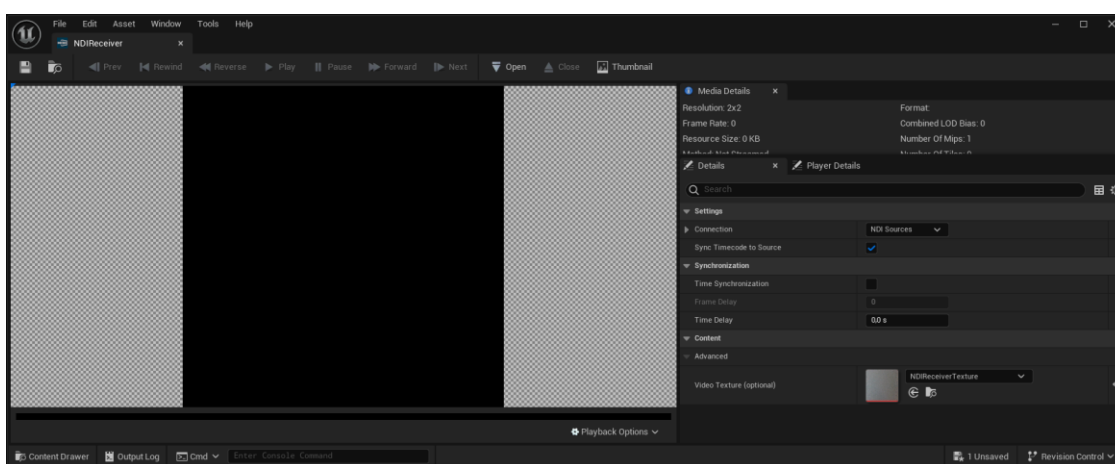


Slika 16 NDIIO plugin

Sada je moguće koristiti NDIIO plugin u Unreal Engine projektu napravljenom u trećem poglavlju.

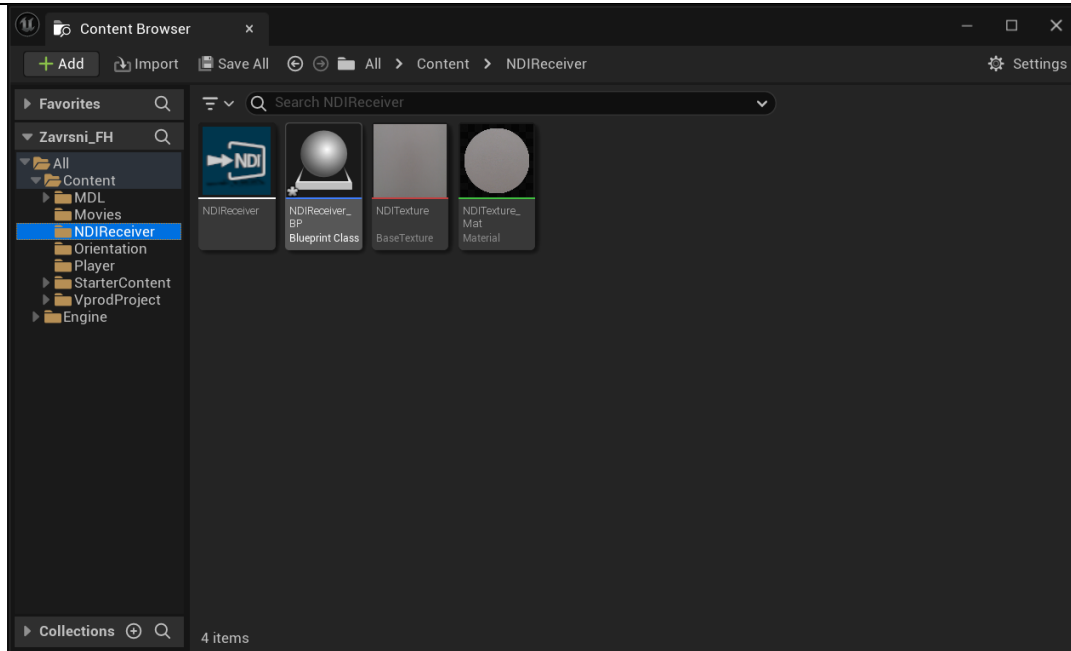
4.3. NDIIO U UNREAL ENGINE PROJEKTU

U Unreal Engine projektu se napravi nova datoteka *NDIReceiver* u koju se spremaju sve komponente potrebne za rad *NDIReceiver* stavke NDIIO plugin-a. Desnim klikom na datoteku otvara se izbornik na kojem se pod *Media* kategorijom nalazi stavka *NDI Media Receiver*. U postavkama stavke *NDI Media Receiver* se pod *Video Texture* kategorijom na padajućem izborniku odabire opcija *NDI Media Texture2D* [Slika 17].



Slika 17 *NDIReceiver* stavka

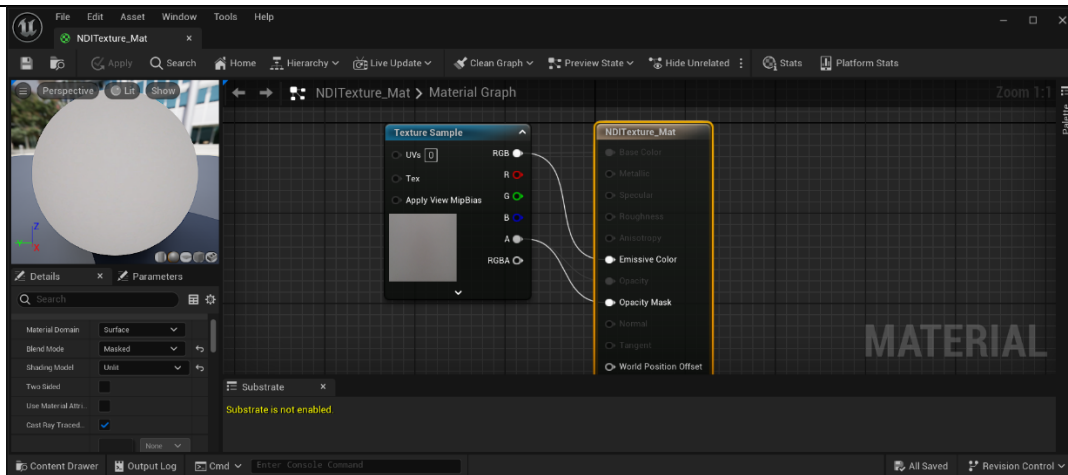
Novostvorenoj teksturi se dodjeljuje ime *NDITexture* i ona se sprema u datoteku *NDIReceiver*. Tekstura se zatim povlačenjem miša stavlja na ravninu kreiranu u drugom poglavlju. Nadalje se kreira novi *Blueprint* imena *NDIReceiver_BP*. Sada *NDIReceiver* datoteka sadrži *NDIReceiver* i *NDITexture* stavke te *NDITexture_Mat* materijal i *NDIReceiver_BP Blueprint* [Slika 18]. [12]



Slika 18 NDIReceiver datoteka

4.3.1. NDI materijal

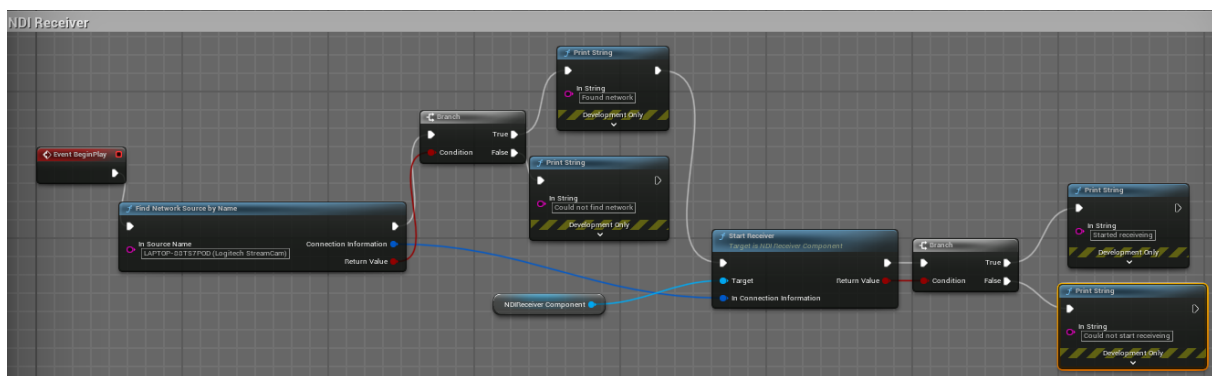
NDITexture_Mat materijal se kreirao automatski nakon što je NDITexture stavka stavljena na ravninu. U Unreal Engine-u materijali definiraju površinska svojstva objekata u sceni nekog projekta. Ovdje ovaj materijal određuje površinska svojstva ravnine na koju je stavljen. Odlaskom u postavke materijala otvara se novi prozor u kojem se nalazi *Blueprint* za manipulaciju svojstvima materijala. U *Blueprint*-u se nalaze *Texture Sample* i *NDITexture_Mat* čvorovi. Jedan od izlaznih parametara iz *Texture Sample* čvora je *RGB* koji se spaja na *Emissive Color* ulazni parametar *NDITexture_Mat* čvora. Zatim se u postavkama materijala u padajućem izborniku pod kategorijom *Shading Model* odabire opcija *Unlit* kako drugi izvori svjetla unutar projekta ne bi utjecali na njegovo osvjetljenje [Slika 19].



Slika 19 NDI materijal

4.3.2. NDIReceiver_BP

U NDIReceiver_BP Blueprint-u se ponovno nalazi čvor *Event BeginPlay*. On se spaja na čvor *Find Network Source by Name* u kojem se definira izvor videa (naziv izvora mora biti isti kao i ime prozora od *Studio Monitor* alata). Izlazni parametar *Connection Information* se spaja s ulaznim parametrom *In Connection Information* čvora *Start Receiver*. Ovaj čvor je zadužen za početak primanja video prikaza sa StreamCam kamere. Između ta dva čvora se dodavaju čvorovi koji daju povratne informacije tijekom obavljanja procesa [Slika 20].

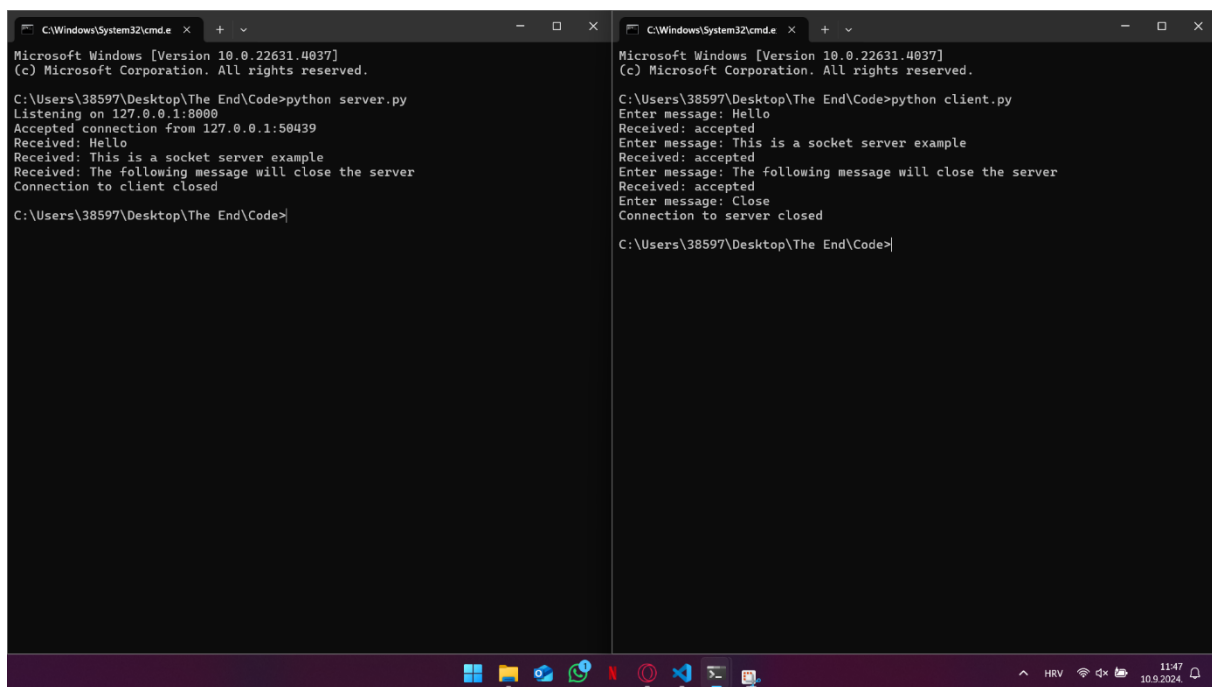


Slika 20 NDIReceiver_BP Blueprint

Sada se video s kamere preko *Screen Capture* alata prikazuje u *Studio Monitor* alatu koji služi kao izvor za NDIReceiver stavku u Unreal Engine-u. Nakon pokretanja simulacije projekta video s kamere prikazuje se na ravni u projektu.

5. UDP SOCKET SERVER I KLIJENT

UDP server se u ovom radu koristi za ekstrakciju podataka o rotaciji VR uređaja u Unreal Engine-u. Ti podaci se zatim obrađuju u Python skripti za daljnju uporabu. U prilogu su priložene dvije Python skripte koje pokazuju uobičajeni rad UDP servera i klijenta. Jedna skripta `server.py` sadrži sintaksu potrebnu za pokretanje i rad UDP servera, dok druga `client.py` skripta sadrži potrebnu sintaksu za pokretanje i rad UDP klijenta. Kada se pokrene UDP server skripta ispisuje se povratna informacija o *Internet Protocol*(IP) adresi na kojoj se server nalazi te preko kojeg priključka (engl. Port) server očekuje pakete od klijenta. Nakon pokretanja klijenta UDP server vraća povratnu informaciju o prihvatanju veze s klijentom. Odatve je moguće preko klijenta slati poruke koje će server zaprimiti te poslati povratnu informaciju o prihvatanju poruke klijentu [Slika 21]. [13]



```
C:\Windows\System32\cmd.exe x + v
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\38597\Desktop\The End\Code>python server.py
Listening on 127.0.0.1:8000
Accepted connection from 127.0.0.1:50439
Received: Hello
Received: This is a socket server example
Received: The following message will close the server
Connection to client closed

C:\Users\38597\Desktop\The End\Code>

C:\Windows\System32\cmd.exe x + v
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\38597\Desktop\The End\Code>python client.py
Enter message: Hello
Received: accepted
Enter message: This is a socket server example
Received: accepted
Enter message: The following message will close the server
Received: accepted
Enter message: Close
Connection to server closed

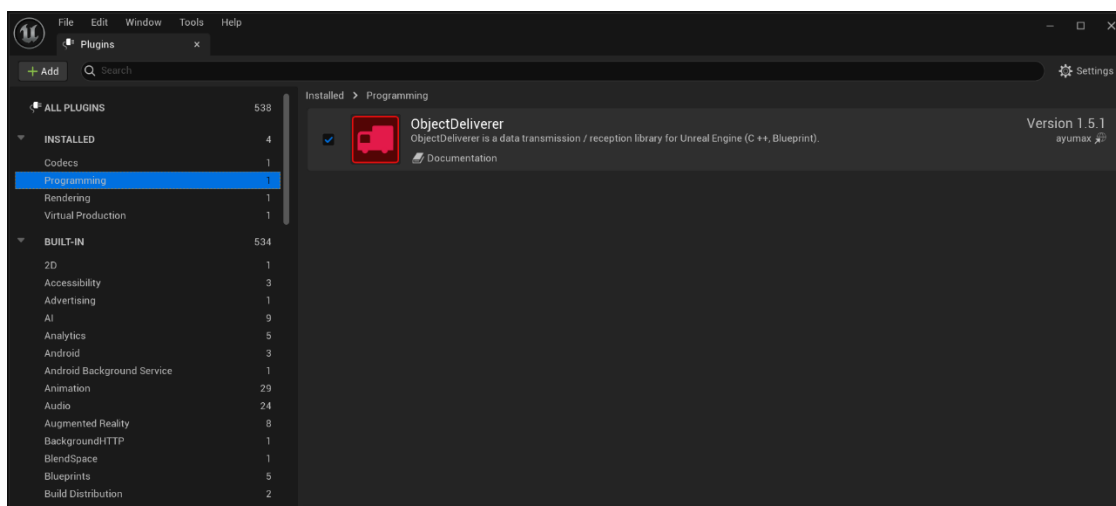
C:\Users\38597\Desktop\The End\Code>
```

Slika 21 Primjer rada UDP socket servera

Sada je potrebno napraviti klijent u Unreal Engine-u koji će slati podatke o orijentaciji VR uređaja na *Socket* server koji će se pokretati iz nove Python skripte unutar koje će se obrađivati zaprimljeni podaci i upravljati UR5 robotskom rukom.

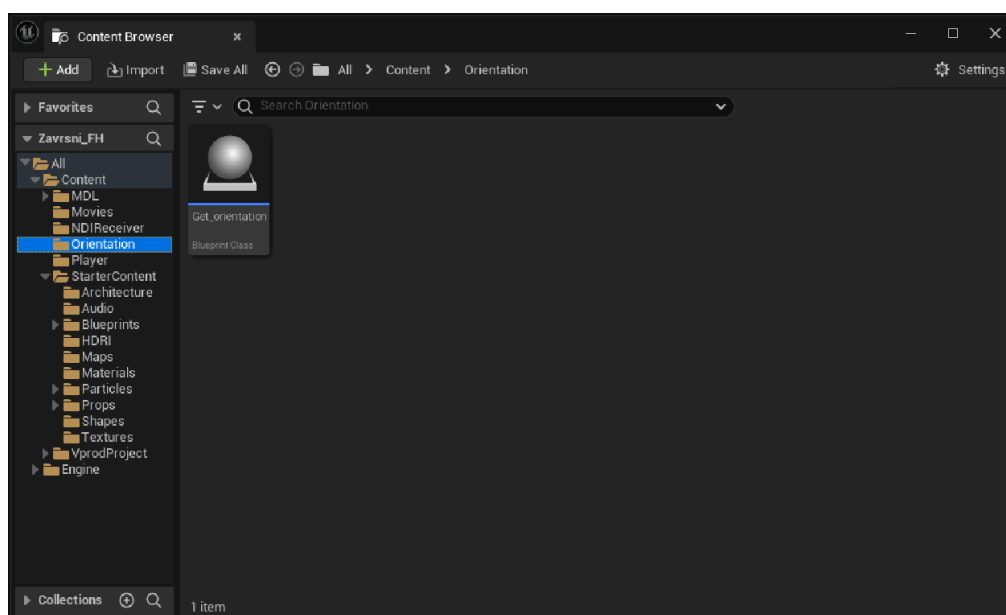
5.1. UDP KLIJENT U UNREAL ENGINE-U

S *Marketplace*-a na Epic Games pregledniku se pronalazi prikladan plugin. Koristi se *ObjectDeliverer* plugin koji se, među ostalim stvarima, koristi i za stvaranje UDP klijenta unutar Unreal Engine projekta. Nakon instalacije plugin-a otvara se Unreal Engine projekt i na alatnoj traci se odlazi na *Edit* te zatim *Plugins*. Otvara se prozor sa svim instaliranim plugin-ovima te kako je *ObjectDeliverer* standardan plugin on je odmah i aktivan [Slika 22].



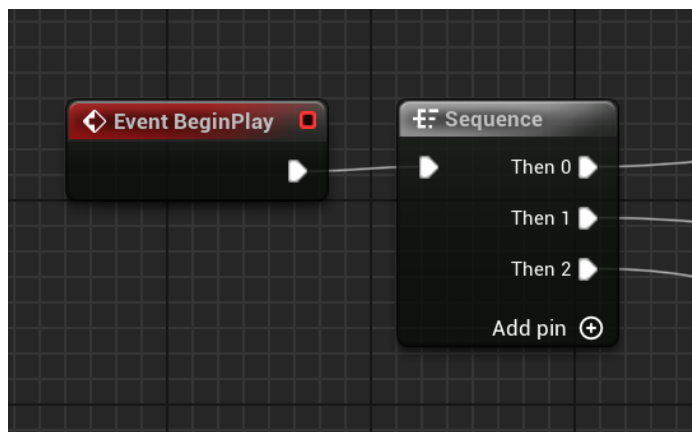
Slika 22 *ObjectDeliverer* plugin

U projektu se kreira nova datoteka imena *Orientation* unutar koje se kreira novi *Blueprint* imena *Get_orientation*. Ovaj *Blueprint* će dobavljati podatke o kutu zakreta uređaja oko z osi (engl. yaw) te će zatim te podatke slati na UDP server [Slika 23].



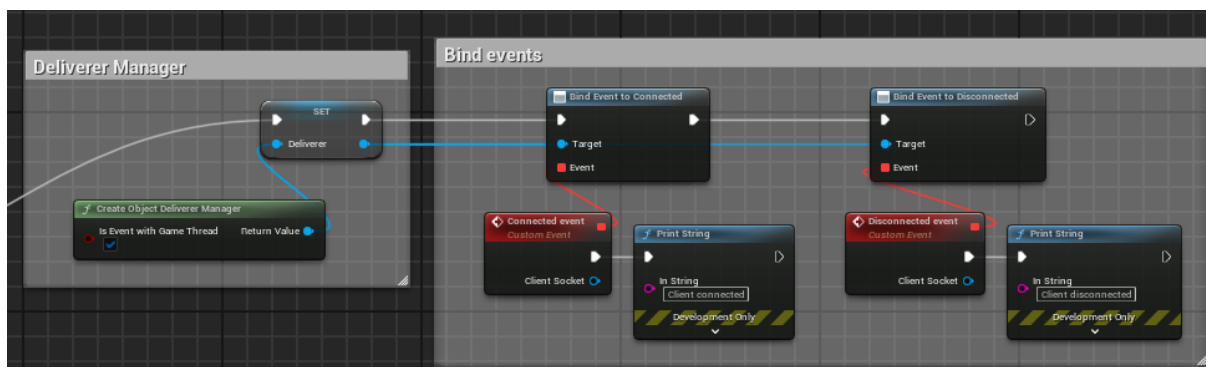
Slika 23 *Orientation* datoteka

Otvora se *Get_orientation Blueprint* unutar kojeg se *Event BeginPlay* čvor veže na *Sequence* čvor. *Sequence* čvor je jedan od osnovnih *Flow Control* čvorova koji se koristi za organizaciju toka događaja. Njegova glavna funkcija je omogućiti izvođenje više akcija ili događaja uzastopno, po određenom redoslijedu. *Sequence* čvor može imati bezbrojno mnogo izlaza, ovdje ih ima tri [Slika 24].



Slika 24 *Sequence* čvor

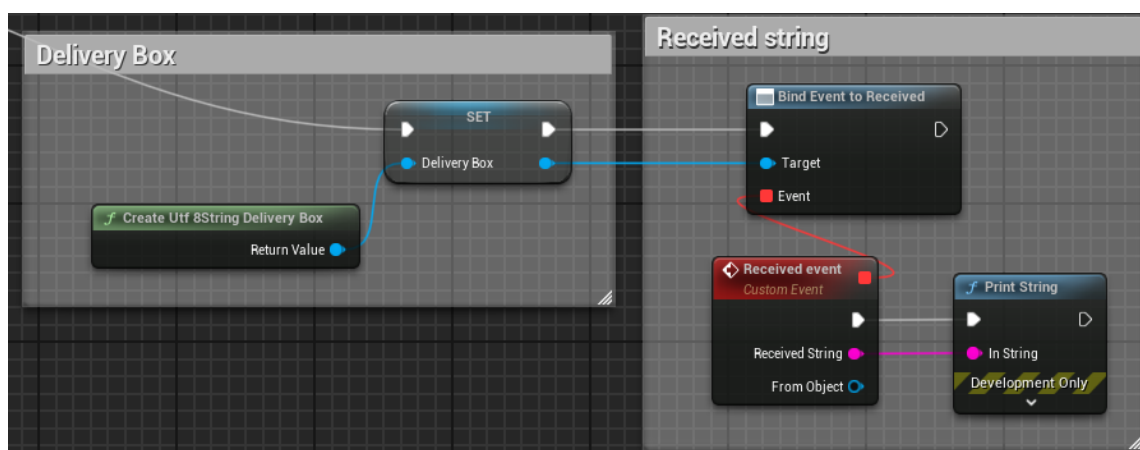
Za pravilnu kreaciju UDP klijenta koristeći *ObjectDeliverer* plugin za Unreal Engine *Sequence* čvor osigurava pravilnu izvedbu operacija potrebnih za stvaranje UDP klijenta. Prvo se pokreće *Deliverer Manager* skup čvorova koji je zaslužan za postavljanje komunikacijskih kanala između klijenta i servera. On upravlja uspostavljanjem veze, prijenosom podataka i zatvaranjem veze. U skupu čvorova se nalaze dva *Bind Event* čvora koji provjeravaju kada se klijent spoji i odspoji na server. Prilikom spajanja na server šalje se povratna informacija *Client connected* i prilikom odspajanja šalje se povratna informacija *Client disconnected*.



Slika 25 *Deliverer Manager* skup čvorova

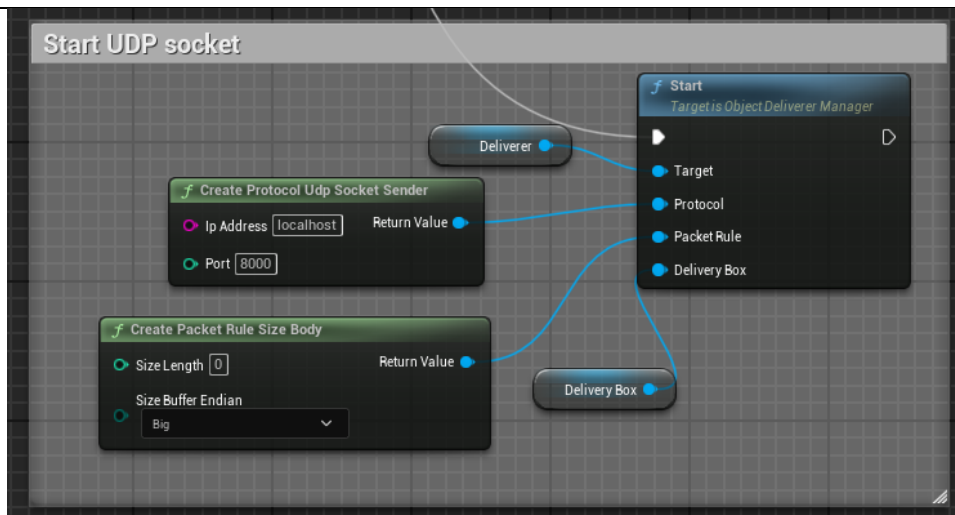
Zatim se pokreće *Delivery Box* skup čvorova koji funkcionira kao spremnik za prijenos podataka. On definira kako će se podaci slati ili zaprimati unutar određenog komunikacijskog

kanala. *ObjectDeliverer* plugin sadrži dvije vrste *Delivery Box* čvorova. Prva je *JavaScript Object Notation*(JSON) koja se najčešće koristi za razmjenu podataka između klijentskih i serverskih aplikacija. Podaci se spremaju u ključ/vrijednost parovima gdje ključ mora biti tekst (engl. String) dok vrijednost može biti bilo koji drugi tip podatka. JSON je lagan format za razmjenu podataka koji je ljudima jednostavan za čitanje i pisanje, ali i strojevima za analizu i generiranje. Druga vrsta čvora šalje podatke serveru u obliku teksta. Koristi *Unicode Transformation Format – 8-bit*(UTF-8) koji je jedan od najčešće korištenih načina kodiranja znakova u tekstualnim datotekama i internetskim protokolima. UTF-8 koristi između jednog i četiri bajta za predstavljanje svakog znaka, ovisno o složenosti znaka. Ovdje se na drugi izlazni parametar *Sequence* čvora veže skup *Delivery Box* čvorova koji koriste UTF-8 za stvaranje spremnika. Moguće je i koristiti JSON, međutim ovdje je to nepotrebno jer se u paket sprema samo jedan podatak. Također se dodaje čvor *Bind Event to Received* koji vraća povratnu informaciju o zaprimanju paketa [Slika 26].



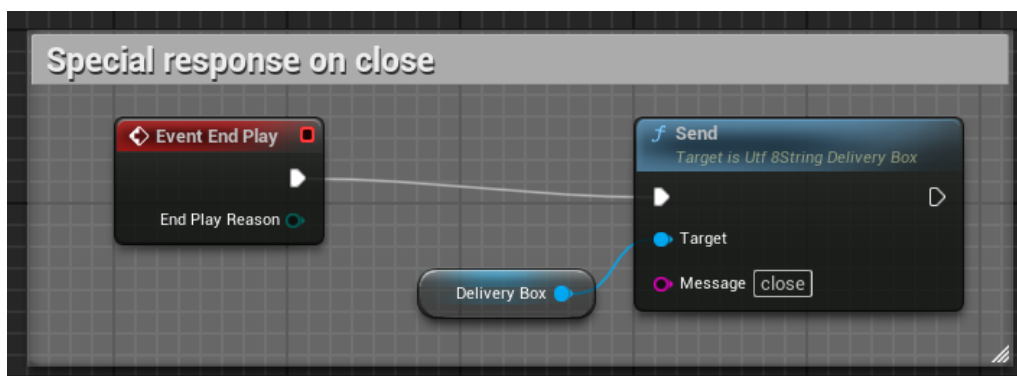
Slika 26 *Delivery Box* skup čvorova

Naposljetku se dodaje skup čvorova koji će pokrenuti UDP *socket*. Dodaje se čvor *Start* koji za jedan ulazni parametar koristi objekt koji je stvoren s *Delivery Manager*-om, a za drugi ulazni parametar koristi objekt koji je stvoren s *Delivery Box*-om. Ostala dva ulazna parametra su *Protocol* i *Packet Rule* parametri. Na *Protocol* se veže izlazni parametar iz čvora *Create Protocol UDP Socket Sender* koji služi za definiranje IP adrese i priključka preko kojeg će klijent slati podatke serveru. *Packet Rule* ulazni parametar služi za definiranje pravila o veličini paketa koji će se slati. Ovdje se na njega veže izlazni parametar iz *Create Packet Rule Size Body* čvora koji sadrži dva ulazna parametra koji određuju veličinu paketa [Slika 27]. [14]



Slika 27 Start UDP socket skup čvorova

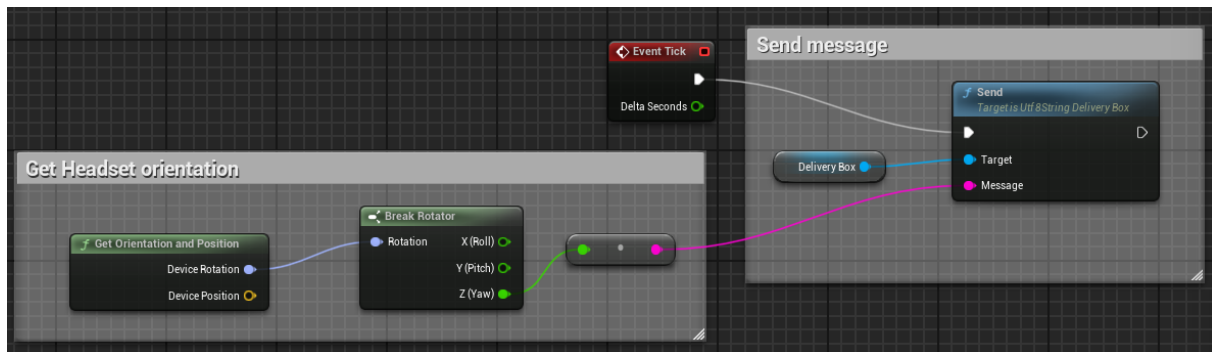
S ova tri skupa čvorova će se uspješno kreirati i pokrenuti UDP klijent koji će slati podatke na UDP server. Međutim, kada klijent prestane slati podatke dodaje se posebna poruka koja će, kada ju on zaprimi, zatvoriti UDP server. Na čvor *Event End Play*, koji će se uključiti kada se simulacija projekta zaustavi, veže se *Send* čvor od *ObjectDeliverer* plugin-a. *Send* čvor sadrži dva ulazna parametra. Na prvi parametar *Target* se veže *Delivery Box* objekt kako bi se poruka spremila u paket i poslala serveru. Drugi ulazni parametar je sama poruka koja glasi „close“ [Slika 28].



Slika 28 Skup čvorova za zatvaranje UDP servera

Sada će se UDP klijent pravilno kreirati i zatvoriti UDP server kada se prekine simulacija projekta. Posljednji skup čvorova koji se dodaje je zaslužan za dohvaćanje podataka o rotaciji VR uređaja i slanje tih podataka UDP serveru. Skup se dijeli u dva djela gdje je prvi dio zaslužan za dohvaćanje podataka, a drugi dio šalje te podatke preko *UDP socket*-a serveru. Za prvi dio se izlazni parametar *Device Rotation* čvora *Get Orientation and Position* veže na ulazni

parametar čvora *Break Rotator* koji rastavlja rotaciju VR uređaja na rotacije oko x, y i z osi. Nadalje se zakret (rotacija oko z osi) pretvara u tekst i veže na ulazni parametar *Message* čvora *Send*. Slično kao i kod skupa čvorova za zatvaranje UDP servera ovdje će *Send* čvor slati podatke o zakretu VR uređaja serveru. Dodatno se čvor *Event Tick* veže na *Send* čvor kako bi se podaci slali tijekom svakog okvira (engl. Frame) simulacije projekta [Slika 29].



Slika 29 Skup čvorova za dohvaćanje i slanje podataka

5.2. UDP SERVER

Prvo je potrebno uvesti *Socket* biblioteku naredbom *import socket*. Zatim se definiraju dvije varijable *udp_ip* i *udp_port* koje u sebi sadrže IP adresu i priključak koji odgovaraju IP adresi i priključku definiranim u *Get_orientation Blueprint-u* u Unreal Engine-u. Nakon toga se definira novi objekt server u kojem se definira vrsta priključka. Zatim se objekt server veže na varijable *udp_ip* i *udp_port* funkcijom *bind()* [Slika 30].

```
# Set up the UDP server
udp_ip = "127.0.0.1" # IP address
udp_port = 8000 # Port

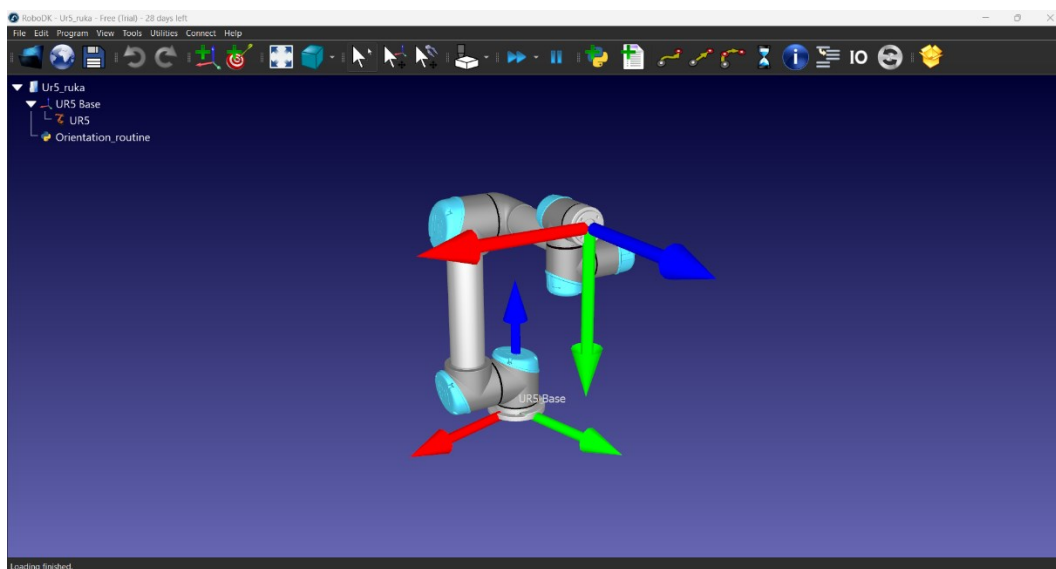
# Create the UDP socket
server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server.bind((udp_ip, udp_port))
```

Slika 30 Inicijalizacija UDP servera

Sada će se UDP server pravilno inicijalizirati, na pravilnoj IP adresi i na pravilnom priključku te će moći primiti podatke od UDP klijenta u Unreal Engine-u.

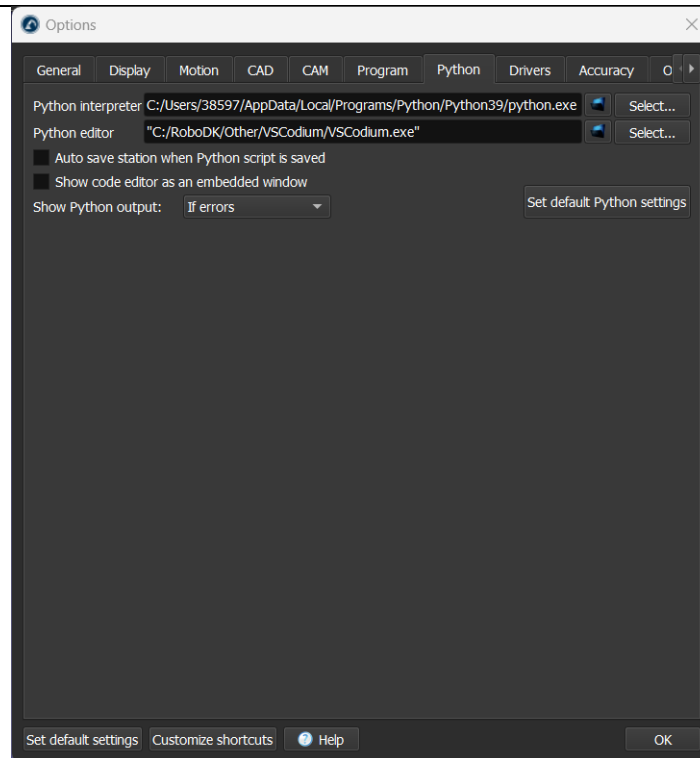
6. STVARANJE PROJEKTA U ROBODK

Nakon instalacije RoboDK i preuzimanja modela UR5 robotske ruke s njihove biblioteke, klikom miša na model otvara se novi projekt (engl. Station) u RoboDK s modelom UR5 robotske ruke. Na alatnoj traci se odabire opcija *Add Python Program*. Novonastala Python skripta se preimenuje u *Orientation_routine* i u nju će se spremati sintaksa za inicijalizaciju UDP servera kao i sintaksa za upravljanje robotom [Slika 31].



Slika 31 RoboDK projekt

RoboDK sadrži osnovni Python interpreter koji već ima instaliranu *Robolink* biblioteku potrebnu za rad s robotskom rukom. Međutim, kako bi se mogle koristiti ostale potrebne biblioteke (*socket*, *re* i *time*) potrebno je na alatnoj traci pod opcijom *Tools* odabrati opciju *Options*. Otvara se novi prozor *Options* u kojem se odlazi u kategoriju *Python* gdje se u polje *Python interpreter* dodaje put do izvršne Python datoteke. Ova izvršna datoteka se odnosi na Python verziju koja je instalirana na računalu [Slika 32].



Slika 32 Python interpreter

Nakon toga se klikne na tipku *OK* kako bi se spremile novouvedene promjene. [15]

6.1. ORIENTATION_ROUTINE SKRIPTA

U skriptu se dodaje sintaksa navedena u petom poglavlju za inicijalizaciju UDP servera zajedno sa ostalim Python bibliotekama navedenim u uvodnom poglavlju.

6.1.1. Dodavanje Robolink biblioteke

Robolink biblioteka se uvodi na isti način kao i *Socket* biblioteka, međutim ovdje se koristi naredba `from robodk.robolink import Robolink` jer je potrebna samo *Robolink()* funkcija. Prvo se funkcijom *Connect()* ostvaruje veza s RoboDK softverom, zatim se koristi funkcija *setRunMode(6)* s kojom se računalo definira kao klijent koji će slati naredbe robotskoj ruci koja se ponaša kao server. Definira se novi objekt *robot* koji je određen funkcijom *Item()* s kojom se obuhvaća model UR5 robotske ruke u RoboDK. Zatim se nad objektom izvršava funkcija *Connect()* koja u sebi sadrži IP adresu fizičke robotske ruke. Također se dodaju dvije funkcije *print()* koja će ispisati povratne informacije o statusu konekcije s fizičkom robotskom rukom. Dvije *print()* funkcije su odvojene sa *sleep()* funkcijom iz Python biblioteke *time* [Slika 33].

```
RDK = Robolink()
# Establish a connection with RoboDK
RDK.Connect()
# PC is the client, robot is the server
RDK.setRunMode(6)

# Connect to UR5 robot
robot = RDK.Item('UR5')
robot.Connect('192.168.0.25') # robots IP

# Check if PC and robot are connected
print(robot.ConnectedState())
time.sleep(20)
print(robot.ConnectedState())
```

Slika 33 Povezivanje s robotskom rukom

Sada se nakon pokretanja skripte računalo može spojiti s fizičkom UR5 robotskom rukom. Još je potrebno dodati logičku petlju s kojom će se robotska ruka pomicati. [16]

6.2. LOGIKA UPRAVLJANJA ROBOTSKOM RUKOM

Prilikom pokretanja simulacije projekta u Unreal Engine-u podaci se konstantno šalju na UDP server koji se inicijalizira iz *Orientation_routine* Python skripte. Sada je potrebno konstantno obrađivati zaprimljene podatke te ih pretvarati u oblik prigodan za upravljanje robotskom rukom. Definira se početna pozicija robotske ruke u obliku liste koja se koristi kao ulazni argument za funkciju *MoveJ()* koja pripada *Robolink* biblioteci. Sada se robotska ruka pri pokretanju skripte pomiče u početni položaj. Također se definira nova varijabla *start_yaw* čija je vrijednost nula i ona predstavlja staru vrijednost zakreta VR uređaja. Rutina za pomicanje robotske ruke se stavlja u *while True* petlju. Na taj način će se robotska ruka pomicati dok se ne izađe iz petlje. Unutar petlje se prvo, koristeći funkcije *recvfrom()* i *decode()*, podaci zaprimaju i dekodiraju prema UTF-8 standardu. Pojavljuje se problem gdje se nakon dekodiranja podataka uz brojevnne vrijednosti pojavljuju posebni simboli. Problem se pojavljuje zbog nepravilnog dekodiranja podataka. Stoga se koristi funkcija *sub()* iz *re* Python biblioteke zajedno s funkcijom *strip()* kako bi se uklonili ti posebni znakovi. Funkcija *sub()* kao jedan od ulaznih argumenata prima tekst „\x20\x7E“. Ovaj tekst predstavlja niz koji koristi heksadecimalni zapis za predstavljanje znakova koji pripadaju *American Standard Code for Information Interchange*(ASCII) standardu. Drugim riječima ovaj niz obuhvaća osnovni raspon

ASCII znakova, uključujući i posebne simbole. [17] Pravilno obrađeni i dekodirani podaci se spremaju u varijablu *cleaned_data* koja u sebi sadrži zakret VR uređaja u obliku teksta. Zatim se dodaje uvjetna naredba koja provjerava je li UDP klijent poslao tekst „close“. Ukoliko je uvjet ispunjen izlazi se iz petlje i server se zatvara funkcijom *close()*. Ukoliko uvjet nije ispunjen petlja se nastavlja i vrijednost iz varijable *cleaned_data* se pretvara u decimalni broj (engl. Float) koji se sprema u novu varijablu *new_yaw* koja predstavlja novu vrijednost zakreta VR uređaja. Na kraju se dodaje još jedna uvjetna naredba čiji je uvjet da apsolutna razlika između nove i stare vrijednosti zakreta mora biti veća od jedan. Na ovaj način se smanjuje opterećenje na privremenu memoriju (engl. Buffer) UDP servera i robotske ruke kako bi se robotska ruka glatko okretala. Inače robotska ruka neće moći pratiti promjene u zakretu VR uređaja. Ukoliko je uvjet ispunjen zglob robotske ruke se pomiče za iznos zakreta VR uređaja i u varijablu *start_yaw* se sprema trenutna vrijednost zakreta [Slika 34].

```
while True:
    # Receive data from Unreal Engine
    data, addr = server.recvfrom(32) # Buffer size is 1024 bytes
    # Decode the data to a string
    decoded_data = data.decode('utf-8')

    # Use regex to filter out non-printable characters
    cleaned_data = re.sub(r'^[\x20-\x7E]', '', decoded_data).strip()

    print(f"Received Yaw: {cleaned_data}")

    if cleaned_data == "close":
        break

    # Use data to move the UR5 base joint
    new_yaw = float(cleaned_data)

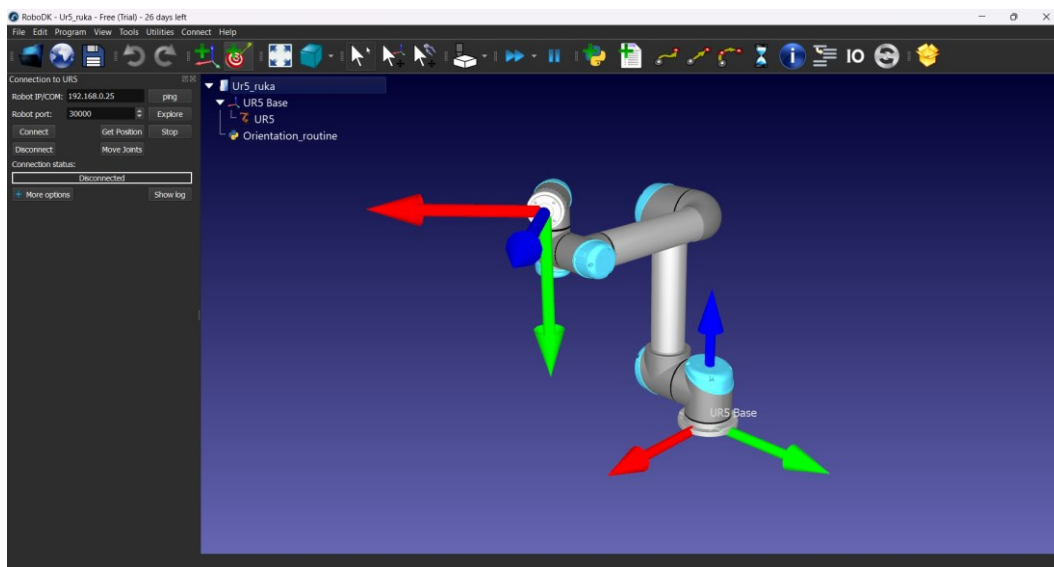
    if abs(new_yaw - start_yaw) >= 1:
        new_position = [0, -90, -90, 0, -(new_yaw - 90), 0]
        robot.MoveJ(new_position)
        start_yaw = new_yaw
```

Slika 34 Rutina

Unreal Engine odlično prati zakret VR uređaja, međutim pojavljuju se skokovi u zakretu u iznosu od dvjesto stupnjeva kada se uređaj okrene prema gore ili prema dolje. Razlog tomu je taj što Unreal Engine ne može utvrditi točnu poziciju uređaja oko z osi jer on tada gleda u njezinom smjeru. Kako bi se izbjegao ovaj problem u petlju se dodaje još jedna uvjetna naredba koja provjerava ako je razlika između nove i stare vrijednosti zakreta veća od sto. Ukoliko je uvjet ispunjen ta vrijednost zakreta se ignorira i uzima se sljedeća nova vrijednost. Na ovaj način se robotska ruka neće naglo okretati već će vjerodostojno pratiti zakret uređaja.

7. POVEZIVANJE S UR5 ROBOTSKOM RUKOM

Sada je cijeli projekt u RoboDK spreman za povezivanje s fizičkom robotskom rukom. Na alatnoj traci se odabire kategorija *Connect* unutar koje se nalazi opcija *Connect Robot*. Otvara se novi prozor u kojem se nalaze opcije za povezivanje s UR5 robotskom rukom. U prvo polje se upisuje IP adresa robotske ruke, a u drugo njezin priključak. Nakon toga pritiskom na tipku *Connect* računalo se povezuje s robotskom rukom koristeći standardnu skriptu za povezivanje koja se nalazi na robotskoj ruci [Slika 35].

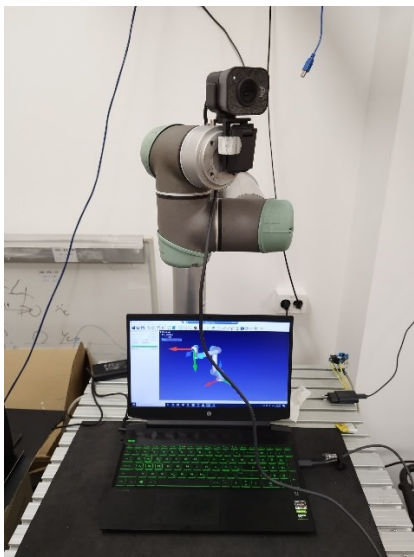


Slika 35 Povezivanje s UR5 robotskom rukom

Rad je potpun i sve komponente su povezane i spremne za testiranje.

8. TESTIRANJE

StreamCam kamera je uključena u računalo i montirana na robotsku ruku. Na računalu je upaljen projekt u RoboDK i računalo je spojeno s robotskom rukom [Slika 36].



Slika 36 Testiranje

Video sa StreamCam kamere se prikazuje na *StudioMonitor* alatu gdje se onda prikazuje u projektu u Unreal Engine-u. VR uređaj je spojen koristeći VivePort i SteamVR softver te je omogućen VR pretpregled u Unreal Engine-u. Nakon pokretanja simulacije projekta Unreal Engine je uspješno spojen na UDP server i video sa StreamCam kamere je vidljiv kroz VR uređaj. Sada se pokreće *Orientation_routine* skripta u RoboDK koja vraća povratne informacije o uspješnom povezivanju s robotskom rukom i nakon dvadeset sekundi počinje primati podatke o zakretu VR uređaja. Robotska ruka uspješno prati zakret VR uređaja bez ikakvih smetnji, a krajnji rezultat je prikazan na slici ispod [Slika 37].



Slika 37 Završni rezultat

9. ZAKLJUČAK

U ovom završnom radu uspješno je demonstrirana integracija kamere, virtualne stvarnosti i robotske ruke u cilju postizanja interaktivnog upravljanja robotskim sustavom u stvarnom vremenu. Kroz VR projekt razvijen u Unreal Engine-u prikazan je prijenos videa s Logitech StreamCam kamere putem NDI alata i Unreal Engine plugin-a, čime je omogućeno vizualno praćenje okoline u stvarnom vremenu. Korištenje Vive Cosmos Elite VR uređaja omogućilo je kontinuirano praćenje zakreta glave korisnika, čiji su podaci putem UDP servera proslijeđeni na Python skriptu unutar RoboDK softvera. Ova skripta je sinkronizirala rotaciju zgloba UR5 robotske ruke s rotacijom VR uređaja, čime je postignuta visoka razina interakcije između korisnika i robota. Rezultati ovog rada potvrđuju uspješnu implementaciju sustava koji omogućava intuitivno upravljanje robotskom rukom koristeći VR tehnologiju. Time je ostvarena spona između fizičkog i digitalnog svijeta u stvarnom vremenu, što otvara mogućnosti za daljnje primjene ove tehnologije. Rad je također demonstrirao važnost pravilne integracije softverskih i hardverskih alata u svrhu postizanja preciznog i pouzdanog upravljanja robotskim sustavima putem virtualne stvarnosti.

LITERATURA

- [1] StreamCam: <https://www.logitech.com/en-us/products/webcams/streamcam.960-001286.html>, pristupljeno 11.09.2024.
- [2] Unreal Engine: <https://www.unrealengine.com/en-US>, pristupljeno 11.09.2024.
- [3] Vive Cosmos Elite: <https://www.vive.com/sea/product/vive-cosmos-elite/features/>, pristupljeno 11.09.2024.
- [4] NDI: https://en.wikipedia.org/wiki/Network_Device_Interface, pristupljeno 12.09.2024.
- [5] Python: <https://www.python.org/doc/essays/blurb/>, pristupljeno 12.09.2024.
- [6] UR5: https://en.wikipedia.org/wiki/Universal_Robots, pristupljeno 12.09.2024.
- [7] RoboDK: <https://robodk.com>, pristupljeno 13.09.2024.
- [8] RoboDK biblioteka:
https://robodk.com/library?RDK&utm_source=RoboDK&utm_medium=Software&Lang=en&PC_OS=WIN64&ProgName=RoboDK&ProgVer=5.8.0&ProgDate=2024-09-06&PCID=d80068fea5e7a22f536897e9b772f867&LNH=467265652028547269616c29&id_login=9462015#filter?name=ur5, pristupljeno 13.09.2024.
- [9] NDI alati: <https://ndi.video/tools/download/>, pristupljeno 14.09.2024.
- [10] NDI za Unreal Engine:
https://www.youtube.com/watch?v=Cnmz1i3k4nE&list=PLWXvQZSGgrsMy6os7rBwH5L6t2_7ifh7-, pristupljeno 14.09.2024.
- [11] NDI plugin:
https://www.youtube.com/watch?v=7CK5wXLZvns&list=PLWXvQZSGgrsMy6os7rBwH5L6t2_7ifh7-&index=4, 14.09.2024.
- [12] *NDIReceiver*:
https://www.youtube.com/watch?v=sLHkmRpr6Tk&list=PLWXvQZSGgrsMy6os7rBwH5L6t2_7ifh7-&index=6, pristupljeno 14.09.2024.
- [13] *Socket* server: <https://www.datacamp.com/tutorial/a-complete-guide-to-socket-programming-in-python>, pristupljeno 15.09.2024.
- [14] *ObjectDeliverer* plugin: <https://github.com/ayumax/ObjectDeliverer?tab=readme-ov-file>, pristupljeno 15.09.2024.
- [15] Python biblioteke za RoboDK: <https://robodk.com/forum/Thread-Import-Python-libraries-in-RoboDK>, pristupljeno 15.09.2024.
- [16] Python za RoboDK: <https://robodk.com/doc/en/PythonAPI/index.html>, pristupljeno 15.09.2024.

[17] Regex vodič: <https://www.geeksforgeeks.org/write-regular-expressions/>, pristupljeno
15.09.2024.

PRILOZI

I. Programski kod – UDP klijent

```
import socket

def run_client():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    server_ip = "127.0.0.1"
    server_port = 8000

    client.connect((server_ip, server_port))

    while True:
        msg = input("Enter message: ")
        client.send(msg.encode("utf-8")[:1024])

        # receive message from the server
        response = client.recv(1024)
        response = response.decode("utf-8")

        # if the server sent "closed", break the loop
        if response.lower() == "closed":
            break

        print(f'Received: {response}')

    client.close()
    print("Connection to server closed")

run_client()
```

II. Programski kod – UDP server

```
import socket

def run_server():
    # socket object
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # set server IP to localhost
    server_ip = "127.0.0.1"
    port = 8000

    # bind the socket
    server.bind((server_ip, port))

    # listen for incoming connections
    server.listen(0)
    print(f'Listening on {server_ip}:{port}')

    # accept incoming connections
    client_socket, client_address = server.accept()
    print(f'Accepted connection from {client_address[0]}:{client_address[1]}')

    while True:
        request = client_socket.recv(1024)
        request = request.decode("utf-8") # convert bytes to string

        # if "close" is received from the client, break the loop
        if request.lower() == "close":
            client_socket.send("closed".encode("utf-8"))
            break

        print(f'Received: {request}')

    response = "accepted".encode("utf-8")
```

```
client_socket.send(response)

# close connection socket with the client
client_socket.close()
print("Connection to client closed")
server.close()

run_server()

III. Programski kod – upravljanje robotskom rukom
""" Fran Haramincic """
import socket
import time
import re
# Import RoboDK API and other necessary libraries
from robodk.robolink import Robolink

# Set up the UDP server
udp_ip = "127.0.0.1" # IP address
udp_port = 8000 # Port

# Create the UDP socket
server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server.bind((udp_ip, udp_port))

RDK = Robolink()
# Establish a connection with RoboDK
RDK.Connect()
# PC is the client, robot is the server
RDK.setRunMode(6)

# Connect to UR5 robot
robot = RDK.Item('UR5')
```

```
robot.Connect('192.168.0.25') # robots IP

# Check if PC and robot are connected
print(robot.ConnectedState())
time.sleep(20)
print(robot.ConnectedState())

start_position = [0,-90,-90,0,90,0]
start_yaw = 0.00
robot.MoveJ(start_position)

print(f'Listening for UDP packets on {udp_ip}:{udp_port}...')

while True:
    # Receive data from Unreal Engine
    data, addr = server.recvfrom(32) # Buffer size is 1024 bytes
    # Decode the data to a string
    decoded_data = data.decode('utf-8')

    # Use regex to filter out non-printable characters
    cleaned_data = re.sub(r'[^\x20-\x7E]', '', decoded_data).strip()

    print(f'Received Yaw: {cleaned_data}')

    if cleaned_data == "close":
        break

    # Use data to move the UR5 base joint
    new_yaw = float(cleaned_data)

    if abs(new_yaw - start_yaw) >= 1:
        new_position = [0, -90,-90,0,-(new_yaw - 90),0]
        robot.MoveJ(new_position)
        start_yaw = new_yaw
```

```
if abs(new_yaw - start_yaw) >= 100:  
    pass  
  
server.close()  
print("Server closed")
```