Koncept umjetne kože za fizičku interakciju čovjeka i robota

Ljubičić, Ivo

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Ivo Ljubičić

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Prof. dr. sc. Bojan Jerbić, dipl. ing.

Student:

Ivo Ljubičić

Zagreb, 2020.

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

Zahvaljujem se mentoru prof. dr. sc. Bojanu Jerbiću na stručnoj pomoći i korisnim savjetima tijekom izrade ovog rada.

Isto tako, želim se zahvaliti svima koji su me podržali tijekom studiranja, ali i cijelog života.

Ivo Ljubičić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment,
inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602 - 04 / 20 - 6 / 3	
Ur. broj: 15 - 1703 - 20 -	

DIPLOMSKI ZADATAK

Student: **IVO LJUBIČIĆ** Mat. br.: 0035188613

Naslov rada na hrvatskom jeziku: **Koncept umjetne kože za fizičku interakciju čovjeka i robota**

Naslov rada na engleskom jeziku: **Artificial skin concept for human-robot physical interaction**

Opis zadatka:

Tijekom socijalne interakcije ljudi kombiniraju različite senzorske podražaje kako bi interpretirali specifična neverbalna značenja. Značajno mjesto u neposrednoj interakciji predstavlja fizička interakcija. Implementacija slične sposobnosti razumijevanja fizičke interakcije kod robota može značajno poboljšati primjenu robota u kooperativnim zadaćama gdje ljudi i roboti dijele radni prostor i surađuju. Osim unapređenja intuitivnosti fizičke interakcije čovjek-robot važno je istaknuti povećanje sigurnosti primjene robota u zadaćama u kojima može doći i do nenamjernih kontakata. Jedna od pretpostavki za ostvarivanje fizičkih interakcijskih sposobnosti robota je razvoj taktilne senzorike, odnosno robotske umjetne kože koja ima sposobnost osjeta dodira, blizine objekta, sile pritiska i temperature dodira.

U ovom radu je potrebno istražiti mogućnosti primjene mikro i mini senzorskih komponenata za izradu mreže senzora koji bi tvorili tzv. umjetnu kožu robota. Za odabrane senzorske i upravljačke komponente potrebno je izraditi prototip za koji treba razviti osnovne algoritme za obradu i interpretaciju signala. Prototip će se potom ispitati na jednom od robota u Laboratoriju za projektiranje izradbenih i montažnih sustava.

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

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
5. ožujka 2020.	7. svibnja 2020.	11. svibnja do 15. svibnja 2020.

Zadatak zadao: Predsjednica Povjerenstva:

prof. dr. sc. Bojan Jerbić prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
U	V	Napon
I	A	Jakost struje
R	Ω	Električni otpor
m	kg	Masa
Vcc	V	Izlazni napon kontrolera (5V)
FSR	Ω	Otpor senzora

SAŽETAK

U ovom radu osmišljen je i napravljen sustav za interakciju čovjeka i robota. U njemu će biti prikazana reakcija robota na vanjski podražaj. Ulazna jedinica je *Arduino* mikrokontroler s četiri senzora pritiska. Podaci se pomoću neuronske mreže i filtera obrađuju te *Arduino* prema robotu šalje uputu putem bežične komunikacije (*WiFi*). Robot je simuliran u RoboDK programskom paketu.

Cilj ovog rada je prikazati kako mikrokontroler, uz primjenu neuronskih mreža na ulaznim signalima, može osposobiti robota za interakciju s čovjekom te nakon toga prikazati i primjenu u slučajevima osiguranja sigurnosti, učenju i naredbama upravljanom gibanju.

Ključne riječi: upravljanje, bežična komunikacija, interakcija, simulacija, neuronske mreže.

SUMMARY

This thesis presents a system for human-robot interaction. It shows how a robot reacts when it receives an input from the microcontroller. Input will be gathered via four pressure pads by the *Arduino* controller and then passed through a neural network and send as a command to the robot through WIFI connection. The robot is simulated in Robed package.

The main goal is to show how a microcontroller with implemented neural network can be applied to a robot and enable that robot interact with a human. It also shows different applications like secure interaction, learning and gesture control.

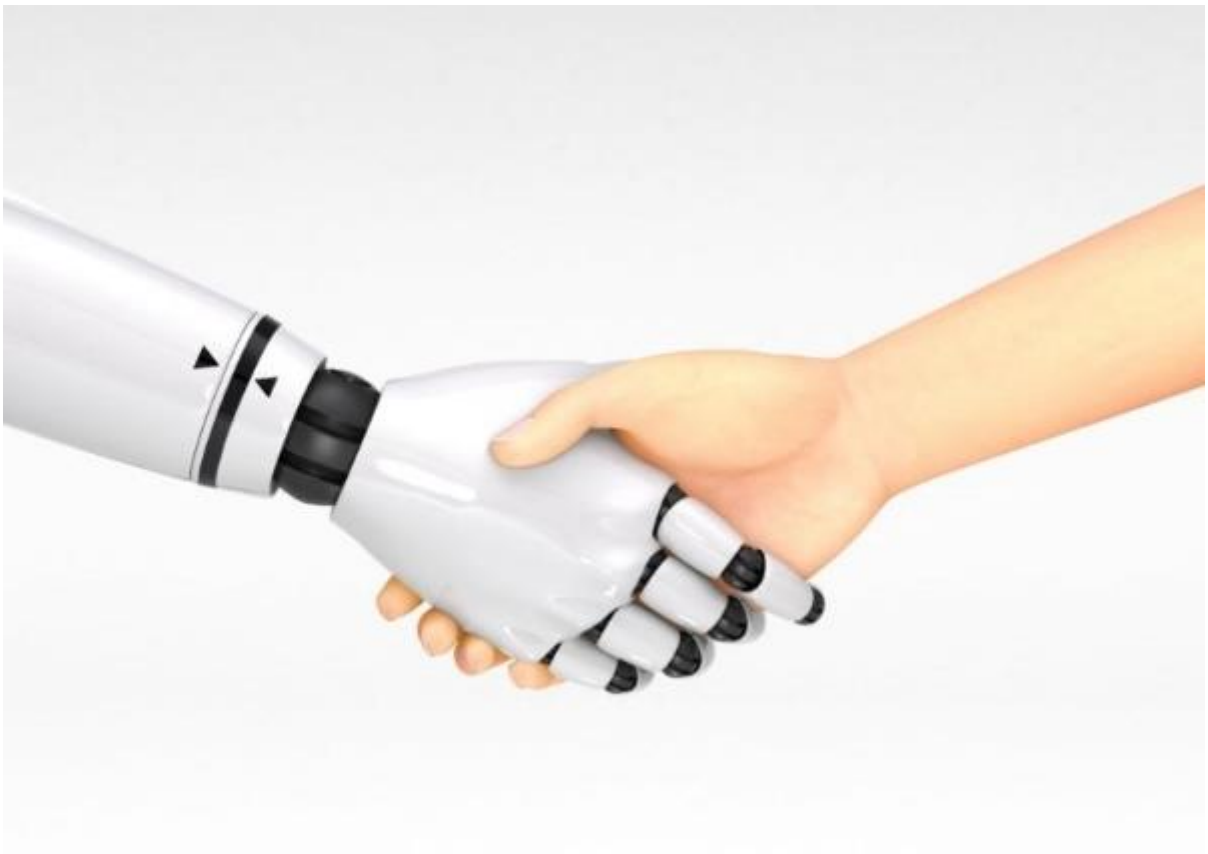
Key words: controlling, wireless communication, human-robot interaction, simulation, neural network.

1. UVOD

S obzirom na napredak tehnologije u industriji, pogotovo Industrije 4.0, primjena robota je u konstantnom porastu. Potreba nadogradnje starijih robota, kao i novijih robota bez senzora ključna je u razvoju nove industrije. Kako današnji roboti i dalje često surađuju s ljudima, u operativnom prostoru robota potrebno je osigurati sigurnost ponajprije čovjeka, a onda i opreme. Kako bi se to omogućilo, robotima su potrebni senzori.

Smisao ovog rada je prikazati kako jedan takav sustav robota, neuronske mreže i senzora, djeluje. Naime, robotima koji su stariji ili napravljeni bez senzora, potrebno je dodati i kontroler koji bi podatke sa senzora obrađivao te potom slao robotu.

Ovim se pristupom osigurava lagana nadogradnja robota koja omogućava kolaboraciju čovjeka i robota ili čak robota i robota tijekom cijelog proizvodnog procesa.



Slika 1. Interakcija čovjeka i robota[1]

1.1. Kolaborativni roboti

Kolaborativni roboti, zvani i coboti, jesu roboti koji određenu radnju izvršavaju u suradnji s čovjekom ili u njegovoj neposrednoj blizini.

Kolaborativni roboti mogu se podijeliti na četiri vrste, ovisno o načinu interakcije:

- rad u blizini, bez zajedničke zadaće
- dijeljenje prostora i zadaće, ali ne istovremeno
- robot i čovjek istovremeno izvršavaju postupak na istom proizvodu/stroju
- robot upravljan čovjekovim kretnjama u stvarnom vremenu

Najvažniji zahtjev za ove robote je sigurnost, a najčešće se koriste kako bi se olakšali monotoni ili čovjeku teški, naporni ili nezdravi postupci. Kolaborativni se roboti koriste u medicini, autoindustriji i drugim industrijama. Najpoznatiji primjer kolaborativnog robota u Republici Hrvatskoj je RONNA G5 robot[2], koji je proizvod suradnje Fakulteta strojarstva i brodogradnje i Kliničke bolnice Dubrava. RONNA se koristi u medicinske svrhe kao prvi robot za stereotaktičku neuronavigaciju.

U ovom radu će se kolaborativnom robotu UR5 dodati senzori kako bi mogao sigurnije i bolje izvršavati zadane zadatke.

1.2. Mikrokontroler i senzori

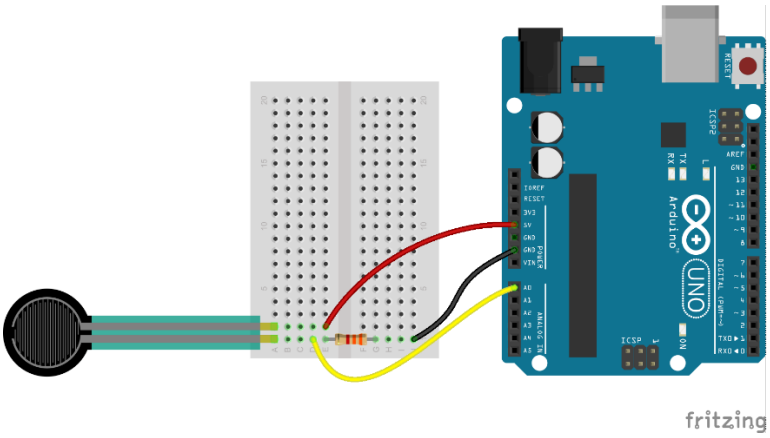
Za početak će se prikazati senzor pritiska te njegov način rada, nakon toga bit će objašnjen način prikupljanja i obrade podataka sa senzora u mikrokontroleru. Potom će obrađeni podaci u specifičnom formatu biti poslani robotu koji će se, ukoliko je došlo do aktivacije senzora, zaustaviti ili odraditi neku drugu operaciju.

Za taj dio obrade podataka zaslužni su filter, koji smanjuje šumove te neuronska mreža, koja od nekoliko podataka kreira konkretnu odluku, treba li se poslati uputa robotu ili ne.

S obzirom da je brzina prijenosa podataka iznimno važna jer nekoliko milisekundi više može značiti razliku između oštećenja opreme, ozljede ili nečega goreg, izbjeći će se korištenje zasebnog web servera te će tu ulogu preuzeti mikrokontroler, dok će robot biti klijent koji će osluškivati komunikacijski kanal u potrazi za mogućim uputama.

Istražit će se sposobnost kontrolera od kojeg se očekuje da omogući rad s čovjekom u direktnom kontaktu, da uči, uočava prepreke, sprječava ozljede i slično. Robot u industriji bez takvih mogućnosti slijep je te se, uz ubrzano povećanje primjene Industrije 4.0, mnogi roboti

neće moći nositi sa spomenutim zahtjevima. U ovom radu provjerit će se može li se taj robot ispravno nadograditi i pravovremeno reagirati na događaje u svojoj neposrednoj okolini.



Slika 2. Mikrokontroler i senzor pritiska[2]

1.3. Neuronska mreža

Neuronske mreže koncept su koji predstavlja presliku rada ljudskog mozga. Koriste se u analizi podataka.

Kako se od robota očekuje da bude spreman učiti nove naredbe, načine rada i sprječavanja nezgoda, u ovom će se radu proučiti i način implementacije neuronske mreže za obradu signala sa senzora i razumijevanje uzoraka, ukoliko su isti naučeni. Time se želi postići sposobnost robota da se pravim uzorkom da posebna naredba kao što je stani, kreni, nauči trenutnu poziciju. Neuronska mreža treba moći naučiti po zadanom predlošku te nakon toga naučeno primijeniti na podacima iz okoline. Zadat će se četiri podatka od kojih ona treba odlučiti koju naredbu treba izvršiti te istu pripremiti za slanje robotu.

Cilj je implementiranu mrežu prikazati kao način nadogradnje neinteligentnih robota *software*-om koji će im omogućiti konkuriranje na tržištu. Iako jednostavna, ova mreža treba prikazati mogućnosti analize podataka.

1.4. Robot

Slika 3. UR5

Tijekom ovog rada robot će biti simuliran u programskom paketu RoboDK. Njemu će u simulaciji biti zadana neka operacija koju treba izvršiti. Ukoliko tijekom izvršavanja te operacije neki od senzora bude aktiviran, robot će dobiti uputu da stane.

Odabran je robot UR5 zbog svoje jednostavnosti i svestrane primjene. Također, ukoliko isključimo sigurnosni stop izazvan otporom gibanju motora (sila od 50+ *N*), nema ni senzore koji bi spriječili dodir bez oštećenja. Simulirani robot treba moći reagirati na naredbe dobivene od kontrolera.

2. SENZORI

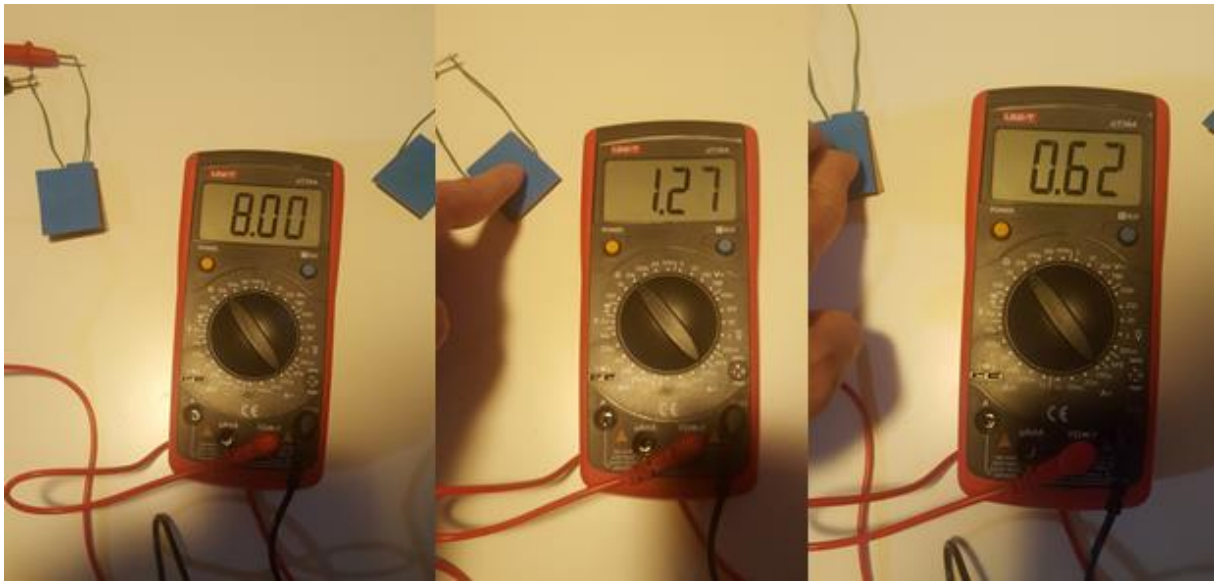
U ovom poglavlju bit će pobliže opisan princip rada i primjena senzora.

Ono što robotima nedostaje kako bi se više ponašali kao ljudi, jesu senzori. Iako ih zovemo osjetima, senzori nam omogućavaju da reagiramo na podražaje iz okoline. Programski kod sam za sebe to ne može, kao što to ne mogu ni ljudi čiji osjeti ne funkcioniraju. Možemo zamisliti robota bez kamera (vizualni senzor) kao slijepog čovjeka. On te podražaje neće moći registrirati. Zato se na robote, kao npr. na ovog koji će biti korišten u radu (Universal robots UR5), postavljaju dodatni senzori. Ukoliko roboti nemaju mogućnost direktne ugradnje senzora, isti se mogu povezati putem vanjskog kontrolera koji će slati robotu poznate podatke.

U ovom radu bit će korišteni senzori pritiska jer se robotu želi omogućiti da prepozna dodir s objektima.

2.1. Elektronički seenzori pritiska

Elektronički senzori pritiska koji će se koristiti rade na principu sličnom potenciometru. Otpor unutar senzora pada s pritiskom, što mikrokontroler može prihvatiti na analognom ulazu. Kada senzor nije pritisnut, otpor je velik i jako mala struja prolazi prema ulaznom pinu te će vrijednost na pinu biti približno 0. Ukoliko se senzor pritisne, otpor pada te se na ulazu registrira vrijednost veća od 0. Ova pojava se može vidjeti na Slici 1

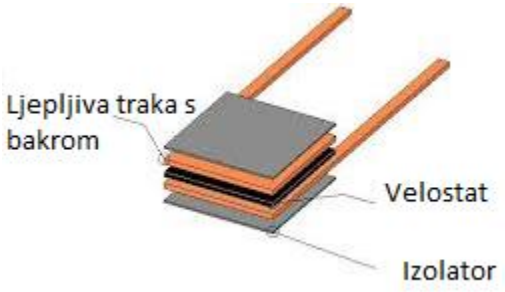


Slika 4. Otpor senzora

Na slici iznad vidljiv je senzor izrađen od materijala zvanog velostat[4]. Razlog izrade senzora u kućnoj radinosti je potreba da ti senzori u nekom trenu pokriju robote po mjeri jer gotovi senzori su specifičnih dimenzija koje bi bilo vrlo teško i vrlo skupo poslagati da pokriju traženu površinu robotske ruke. S obzirom na to, u ovom poglavlju razradit će se izrada budućih senzora koje uz poboljšanje proizvodnog procesa možemo primijeniti na robotskoj ruci.

2.1.1. Izrada senzora

Nakon proučavanja opcija i različitih vrsta senzora, senzori napravljeni od velostata dali su dobre vrijednosti.

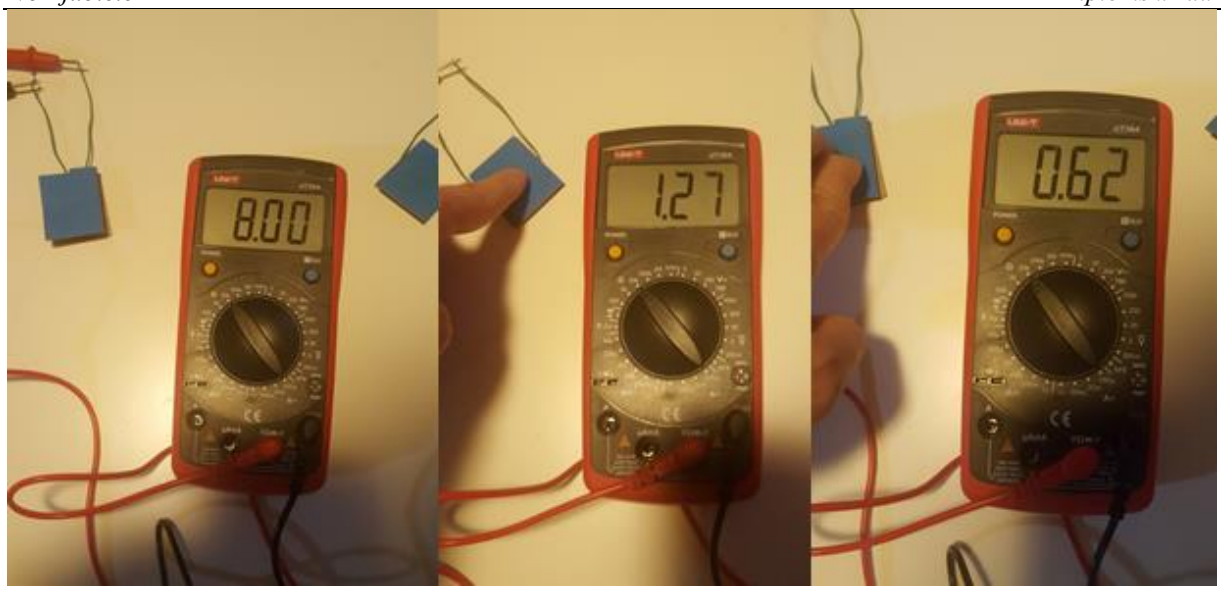


Slika 5. Elektronički senzor pritiska

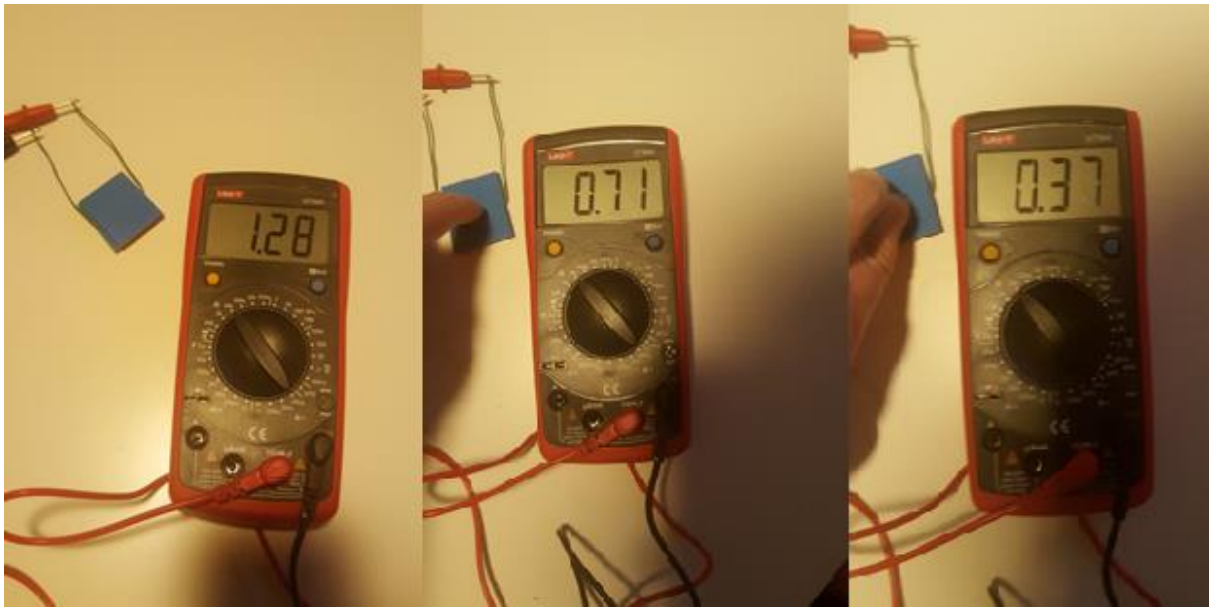
Kao što je vidljivo na slici 3., velostat se nalazi u sredini. S obje strane postavljene su ljepljive trake s bakrenim filmom, a svaka je traka spojena na jednu od žica. Kao što je navedeno, razliku otpora opuštenog i pritisnutog senzora očitavamo uz pomoć kontrolera koji tu razliku čita kao pad napona.

velostat nije jedini takav materijal, ali se kroz mjerenja pokazao puno reaktivnijim u odnosu na ostale slične materijale, poput provodljive pjene. Ona je dala veće razlike napona, ali joj je bilo potrebno više vremena da se vrati u početnu vrijednost, što uzrokuje kašnjenje u reakciji.

Ovu razliku u razlici otpora možemo vidjeti na slici 4 i slici 5.



Slika 6. Pad otpora provodljive pjene



Slika 7. Pad otpora velostata

Tablica 1. prikazuje odnos pada otpora i vrste materijala

Tablica 1. Pad otpora uslijed primjene sile

Tip materijala	Otpor bez pritiska kΩ	Lagani pritisak kΩ	Jak pritisak kΩ
Provodljiva pjena	8	1.27	0.62
velostat	1.28	0.71	0.37

Iako veći pad otpora ima svoje prednosti, velostat je kontinuirano brže reagirao na promjenu pritiska, što je u konačnici prevagnulo jer je bitno postići brzu reakciju robota na dodir kako bi se spriječio udarac i ozljeda čovjeka ili oštećenje opreme.

2.1.2. Mjerenja

Kako bi se pokazala konstantnost mjerenja i pouzdanost napravljenih senzora, uz pomoć nekoliko utega, napravljena su tri mjerenja.

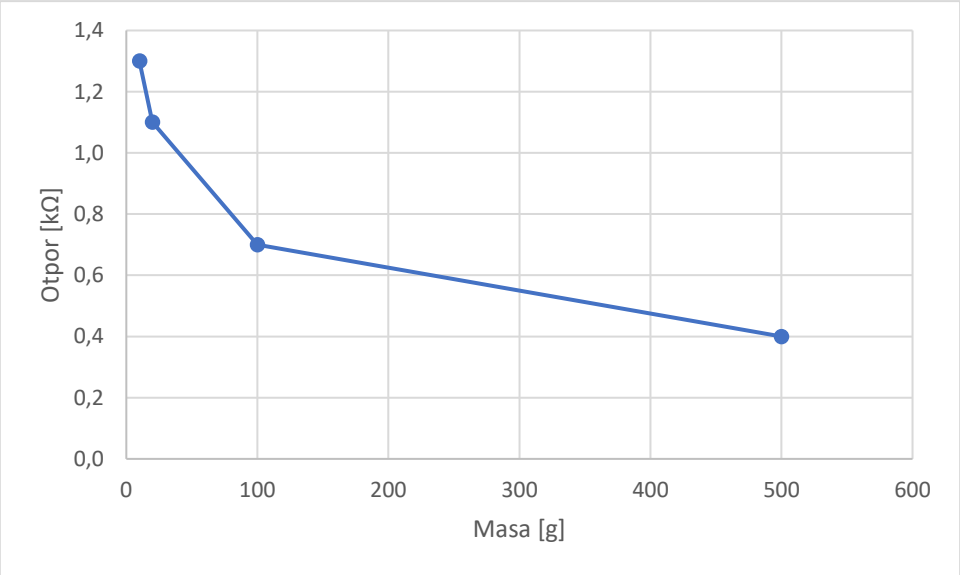
Podaci tih mjerenja vidljivi su u tablici 2.

Tablica 2. Mjerenja pada otpora senzora

Masa	1. mjerenje	2. mjerenje	3. mjerenje
10 g	1.3kΩ	1.29 kΩ	1.25 kΩ
20 g	1.1 kΩ	1.12 kΩ	1.1 kΩ
100 g	0.7 kΩ	0.68 kΩ	0.7 kΩ
500 g	0.4 kΩ	0.4 kΩ	0.4 kΩ

Kao što se u tablici vidi, senzori reagiraju već na pritiske od oko 20g. Iz tablice se može uočiti karakteristika promjene otpora. Ista je zbog bolje vidljivosti prikazana na slici 8. za prvo mjerenje.

Slika 8. Karakteristika senzora



Iako u ovom radu nije obrađeno takvo mjerenje, treba napomenuti da se isti senzor, ako je napravljen savitljivim, može koristiti za mjerenje fleksije.

Zbog nemogućnost izrade više senzora istih karakteristika za ovaj rad, nakon pomnog odabira, odlučeno je koristiti gotove senzore koji se u budućnosti mogu zamijeniti kvalitetnim senzorima željenih dimenzija.

3. MIKROKONTROLERI

S obzirom na temu ovog rada, odabir mikrokontrolera ovisio je o nekoliko parametara: bežičnoj komunikaciji, modularnosti (skalabilnost broja i vrsta senzora) te neizostavnoj procesorskoj moći.

Postojalo je nekoliko opcija: Arduino, ESP32, čak i Raspberry PI. Iako su Arduino i ESP32 kontroleri prilično slabiji, što ih čini sporijima u učenju neuronske mreže, pogodniji su za brzu obradu signala dobivenih od senzora te ne zahtijevaju cijeli operativni sustav. Nadalje će se u radu koristiti samo usporedba dva odabrana kontrolera.

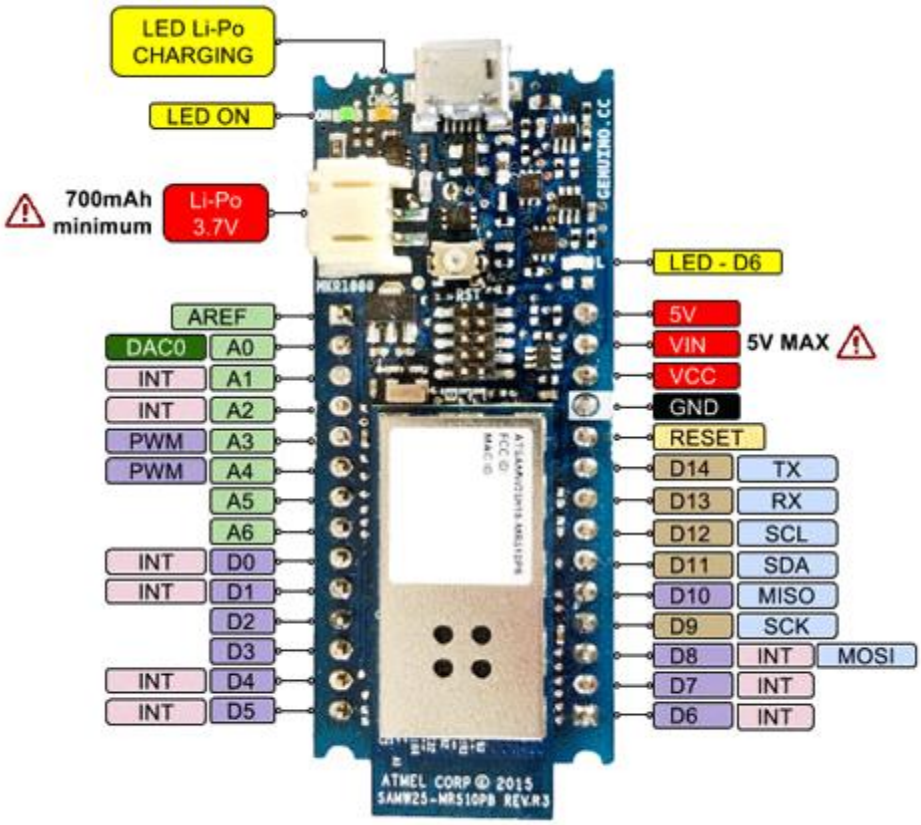
3.1. MKR1000

Arduino MKR1000 programabilni je *open source* modul s ugrađenim WiFi modulom za bežičnu komunikaciju. Osnovne karakteristike ove pločice navedene su u Tablici 3.

Tablica 3. Tehničke karakteristike MKR1000

Mikrokontroler	SAMD21 Cortex-M0+ 32bit low power ARM MCU
Radni napon	3.3 V
Ulazni napon	5 V
Analogni pinovi	7 (A0 – A6)
Digitalni pinovi	8
Jakost struje na *I/O* pinu	7 mA
Jakost struje na 3.3 V pinu	50 mA
Flash memorija	256 KB
SRAM	32 KB
Radna freekvencija	32.768 kHz (RTC), 48 MHz
Komunikacija pinova	I2C, SPI, USART

Na slici 9 prikazan je raspored pinova dostupnih za korištenje.



Slika 9. Raspored dostupnih pinova

Komunikacija korištena kako bi se signal od senzora prenio do simuliranog robota je WiFi. Wifi modul, kao što je već navedeno, ugrađen je u pločicu. Kontroler se u oba slučaja (i kod ESP32) koristi kao server koji čeka spajanje klijenta (simuliranog ili pravog robota) te mu šalje podatke.

Prednost ovog kontrolera su male dimenzije, iznimno mala potrošnja energije te niska cijena. Nažalost, procesorska moć je slaba, a memorija mala, o čemu više u idućim poglavljima prilikom usporedbe s ESP32.

3.2. NodeMCU ESP-32S

ESP32 devkit v4, proizvod kompanije Espressif, pločica je za prototipiziranje nešto većih mogućnosti od Arduinovog MKR1000 modula. Iako MKR1000 zadovoljava sve zahtjeve ovog procesa, tijekom učenja neuronske mreže potreban mu je skoro jedan sat da dobije tražene težinske faktore neuronske mreže. ESP32 za dobivanje tih faktora (zapravo sličnih, više u poglavlju o neuronskoj mreži) treba otprilike dvije minute.

Razlike u performansama možemo vidjeti kroz osnovne karakteristike ovog kontrolera.

Tablica 4. Tehničke karakteristike ESP32

Mikrokontroler	Tensilica 32-bit Single-/Dual-core CPU Xtensa LX6
Radni napon	3.3 V
Ulazni napon	7-12 V
Analogni pinovi	10
Digitalni pinovi	28
Flash memorija	4 MB
SRAM	520 KB
Radna freekvencija	240 MHz
Komunikacija pinova	I2C, SPI, UART

Iz tablice je vidljivo da je čak i jednojezgrena verzija ovog kontrolera pet puta jače procesorske snage, ima više memorije i više pinova.

Zašto onda ne odbaciti MKR1000? Zato što se može pokazati da i slabiji kontroleri mogu odraditi zadatke s neuronskim mrežama. Iako sporije uče, to ih ne sprječava da dođu do ispravnih težinskih faktora te nakon dobivanja tih faktora nema neke razlike u radu ova dva kontrolera. Ti su se faktori mogli dobiti i na računalu pa prenijeti u programski kod mikrokontrolera, o čemu također iscrpnije u poglavlju o neuronskim mrežama.

4. UMJETNA NEURONSKA MREŽA

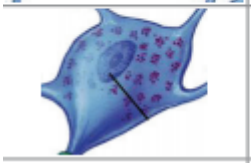
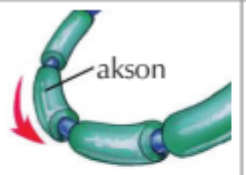
Umjetna inteligencija je dio znanosti o računalima koja se bavi projektiranjem inteligentnih računalnih sustava koji predočavaju karakteristike koje povezujemo s inteligencijom u ljudskom ponašanju. [1]

Pod vrstu umjetne inteligencije spadaju i umjetne neuronske mreže. One su pokušaj oponašanja neuronske mreže u našem mozgu. Njihove mogućnosti do danas nisu dostigle razinu ljudskog mozga, posebno u segmentu učenja.

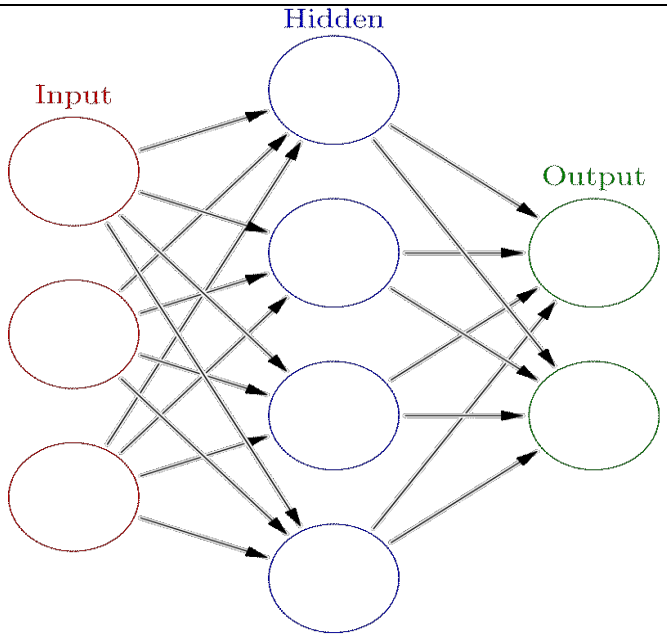
Ljudski se mozak sastoji od mnoštva (10^{11}) neurona, dok su najveće umjetne neuronske mreže na manje od 1% tog broja.

Usporedba osnovnih karakteristika biološkog i umjetnog neurona nalazi se u tablici 5.

Tablica 5. Biološki i Umjetni neuron[1]

Biološki neuron		Umjetni neuron
	Prima ulazni signal putem dendrida (sinaptičke veze)	Prima ulaze (*i*) koji su određeni težinskim koeficijentima (*w*)
	Obrada signala u somi	Obrada ulaza, unutarnji prag – bias (*b*)
	Pretvara obrađeni ulaz u izlaz putem aksona	Pretvara ulaze u izlaz (prijenosna funkcija)
	Šalje informacije putem sinapsi do svih neurona s kojima je neuron povezan	Šalje informaciju prema izlazu i sljedećim neuronima

Neuroni u mozgu i u računalima funkcioniraju na sličan način. S jedne strane imamo ulazne vrijednosti koje se, koristeći razne funkcije, pretvaraju u izlazne vrijednosti. Ulaza i izlaza može biti i više od jednog.



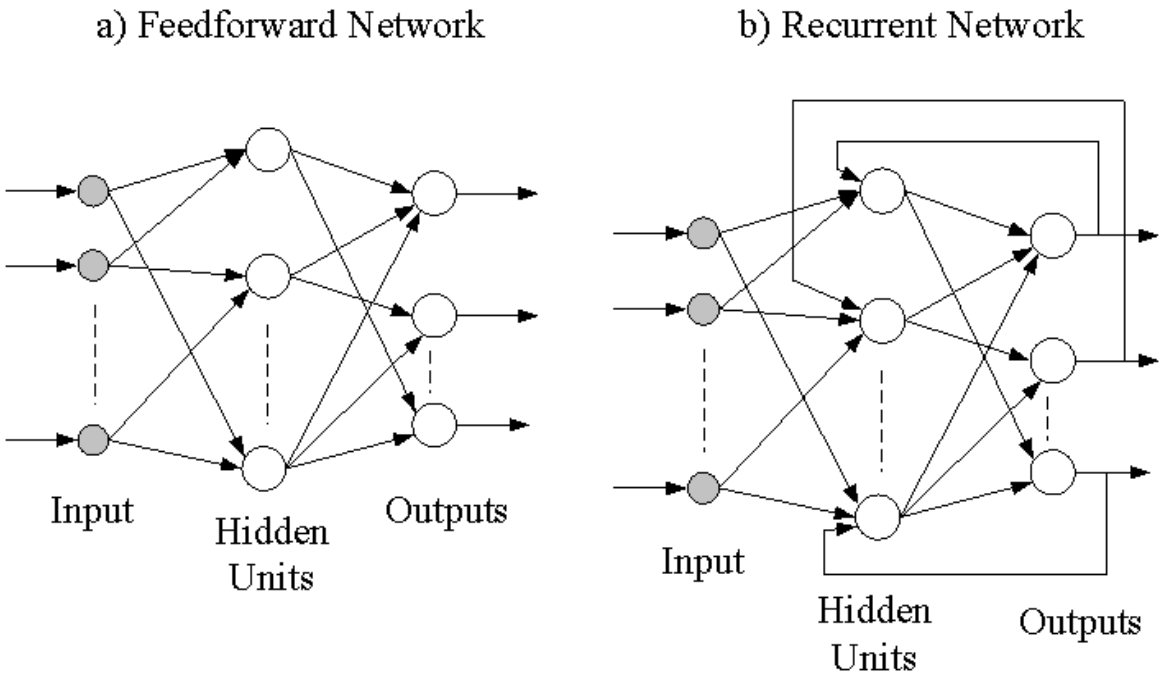
Slika 10. Slojevi neuronske mreže

Kod računala postoji nekoliko tih funkcija, koje zovemo aktivacijskim funkcijama. Najučestalija takva neuronska mreža sastoji se od skupa neurona između kojih putuju signali. Neuroni su raspoređeni u slojeve od kojih razlikujemo: ulazni, skrivene te izlazni sloj (slika 10).

4.1. Podjela neuronskih mreža

Neuronske se mreže mogu podijeliti na mnogo načina, ali najčešće su podjele prema arhitekturi, načinu učenja i vrsti signala.

Prema arhitekturi dijele se na zatvorene (povratne – recurrent neural networks) i otvorene (Feedforward neural networks). Kod zatvorene mreže izlaz se povratnom vezom vraća kao ulaz te koristi za korekciju rješenja, dok kod otvorenih mreža krug nije zatvoren te je smjer signala samo jedan.



Slika 11. Otvorene (lijevo) i zatvorene (desno) neuronske mreže[2]

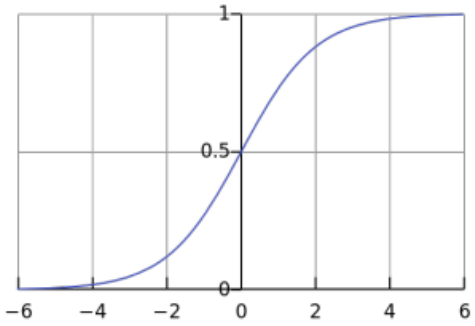
Iduća podjela, prema načinu učenja, ovisi o utjecaju na mrežu tijekom učenja. Nadzirane uzimaju gotove ulazne podatke te dobiju očekivani izlaz, pojačane uzimaju podatke iz okoline, a nenadzirane moraju same razvrstati podatke po nekom parametru što znači da dobiju ulazne podatke, ali bez izlaza.

Zadnja podjela je prema signalima, a oni mogu biti kontinuirani (na primjer između 0 i 1) te diskretni (0 ili 1).

U ovom radu koristit će se nadzirana otvorena neuronska mreža s kontinuiranim signalima.

Aktivacijske funkcije jesu funkcije koje se aktiviraju ako se postigne neki uvjet. Postoji mnogo aktivacijskih funkcija, ali u ovom radu koristit će se samo sigmoidna funkcija čija je formula:

$$\sigma(y) = \frac{1}{1+e^{-y}}$$



Slika 12. Sigmoidna funkcija

4.2. Primjena neuronskih mreža

Neuronske se mreže koriste posvuda – od ekonomije, računalstva, robotike pa sve do medicine. Mogućnosti su neograničene. Npr. obrada podataka koje nije lako ili uopće moguće opisati linearnom vezom, ali ipak imaju neku zajedničku karakteristiku.

Kontroler treba uzimati podatke sa senzora i pretvarati ih u informacije koje se prosljeđuju robotu. Razlog korištenja neuronske mreže je prikaz njenih mogućnosti na jednostavnom i skalabilnom primjeru.

Skalabilnost je iznimno bitna u današnjoj industriji. Proizvodni sustavi sve su kompliciraniji, a isto vrijedi i za robote. Program koji bi jednostavno očitao senzore i za svaku kombinaciju (ili stotine njih) dao nekoliko različitih naredbi, ne bi bio skalabilan klasičnim programiranjem, već bi se kontroler morao neprestano preprogramiravati sa svim mogućim ulazima i izlazima.

Tu u prvi plan ulazi neuronska mreža kojoj bismo mogli dati samo nekoliko opcija i ona bi kroz trening uspjela (uz manju grešku) naučiti sve opcije. U slučaju da se pojave nove opcije, samo se dodaju novi ulazni i izlazni podaci te ponovno pokrene učenje.

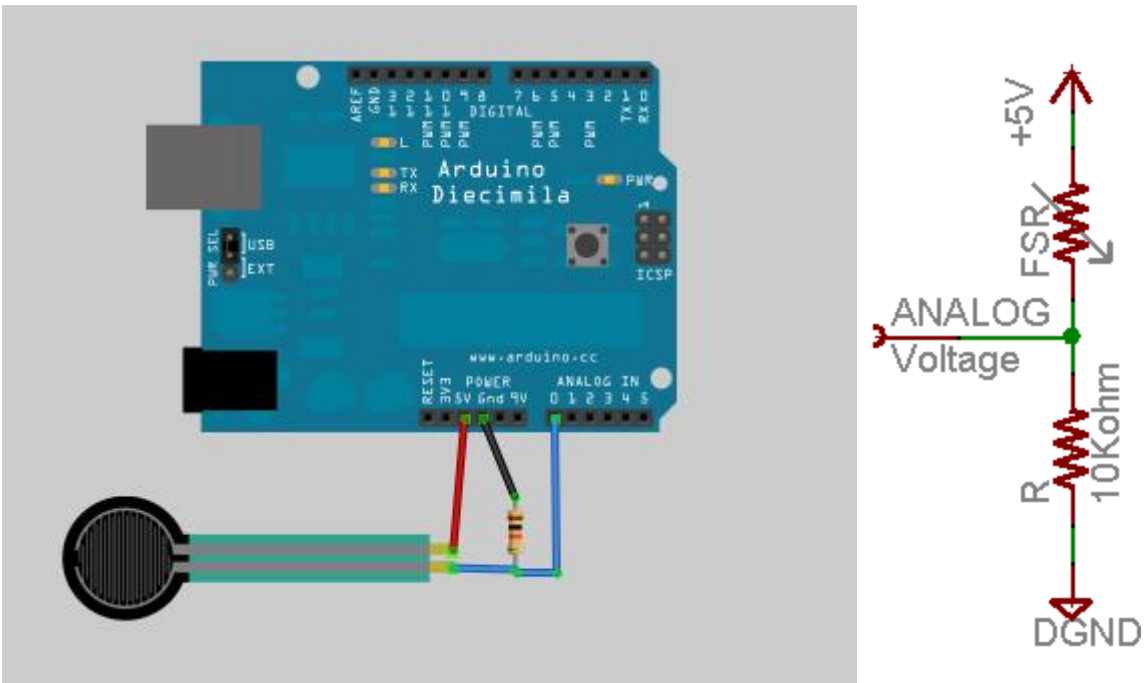
5. OBRADA SIGNALA

U ovom poglavlju obradit će se način prikupljanja podataka sa senzora te njihova obrada kako bi se oni mogli koristiti u neuronskoj mreži. Dosad je prikazan pad otpora pomoću multimetra. Kontroler ne može mjeriti sam otpor, ali može mjeriti pad napona.

5.1. Prikupljanje podataka sa senzora

Elektornički senzori osjetljivi na pritisak su vrsta otpornika s dinamičkim otporom koji se mijenja s obzirom na silu. Njihov otpor bez pritiska je visok, kao što je već objašnjeno. Način na koji su podaci registrirani na analognom inputu mikrokontrolera proizlazi iz činjenice da otpor materijala u senzoru pada s pritiskom.

Senzor se povezuje na 3 pina kontrolera, Vcc (5V) izvor struje, GND ili uzemljenje te analogni pin, što je vidljivo na slici 10.



Slika 13. Povezivanje senzora i kontrolera

Ukoliko senzor nije pritisnut, otpor mu je visok, a struja ne prolazi do analognog pina te kontroler očitava 0. Ukoliko se senzor pritisne, ovisno o pritisku, registrirat će se vrijednost koju mapiramo između 0 i 1023 (za Arduino MKR1000).

Pad napona možemo izračunati formulom

$$\mathbf{Vo} = \mathbf{Vcc}\left(\frac{\mathbf{R}}{\mathbf{R} + \mathbf{FSR}}\right)$$

koja proizlazi iz desnog dijela slike 10. Ako uvrstimo nekoliko različitih vrijednosti sile te napon iz kontrolera od 5V (Vcc), dobijemo vrijednosti iz tablice 6.

Tablica 6. Odnos sile i karakteristika struje[4]

Sila ***N***	**Otpor senzora (R)** **Ω**	**Otpor senzora i otpornika (FSR)** **Ω**	**Struja kroz senzor i otpornik** ***mA***	**Pad napona na otporniku** ***Vo***
0	∞	∞	0	0
0.2	30	40	0.13	1.3
1	6	16	0.31	3.1
10	1	11	0.45	4.5
100	250	10.25	0.49	4.9

U kodu kontrolera, ova se vrijednost dobije funkcijom analogRead() koja daje vrijednost između 0 i 1023. Kako bismo dobili napon, koristimo se sljedećim kodom:

```
//Očitanje sa senzora
int sensor1 = analogRead(A1);
//Preračunavanje u napon
float voltage1 = sensor1 * (5.0 / 1023.0);
```

5.2. Korištenje neuronske mreže

5.2.1. Učenje

Kako bismo gore dobivene podatke mogli koristiti u neuronskoj mreži, prvo ju trebamo naučiti što napraviti s tim ulaznim podacima. Učenje je proces koji se odvija na različite načine, ovisno o primjeni mreže. U ovom slučaju odabrana je neuronska mreža s nadgledanim učenjem gdje

se mreži daju ulazni podaci i izlazni podaci koje očekujemo. Za ovaj program korištena je gotova *open-source* biblioteka *NeuralNetwork*[6] napisana u C++ jeziku te se može koristiti na slabijim uređajima kao što su to Arduino kontroleri.

Ovoj mreži za učenje daje se 10 ulaznih podataka (*inputs*) te 10 odgovarajućih izlaznih podataka (*expectedOutputs*).

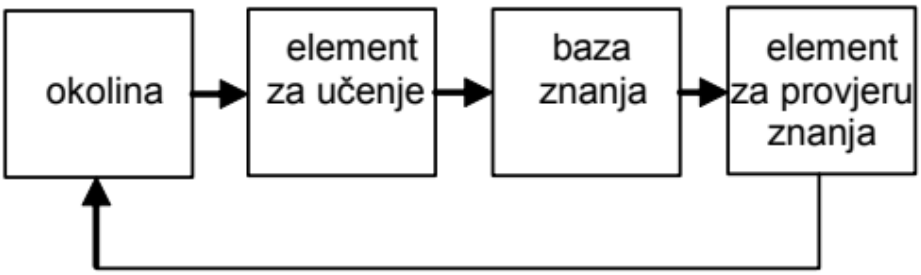
```
const float inputs[PatternCount][InputNodes] = {
 {0, 0, 0, 0},
 {0, 0, 0, 0},
 {0, 0, 0, 0},
 {0, 0, 0, 0},
 {1, 0, 0, 0},
 {0, 1, 0, 0},
 {0, 0, 1, 0},
 {0, 0, 0, 1},
 {1, 1, 0, 0},
 {1, 0, 0, 1},
 {0, 1, 1, 0},
 {0, 0, 1, 1},
 {1, 0, 1, 0}
};

const float expectedOutput[PatternCount][outputNodes] = {
   {0,0},
   {0,0},
   {0,0},
   {0,0},
   {1,0},
   {1,0},
   {1,0},
   {1,0},
   {0,1},
   {0,1},
   {0,1},
   {0,1},
   {0,1}
};
```

Ulazni su podaci spremljeni u 2D niz, a kako je u ovoj programskoj biblioteci za neuronsku mrežu odabrana sigmoidna funkcija, ulazi moraju biti između 0 ili 1.

U svakom redu su četiri vrijednosti koje odgovaraju vrijednostima ulaznih čvorova (senzora), dok svaki red predstavlja jedan trening primjer. U ovom radu odabrano je korištenje dva skrivena sloja (*layer*) s po devet čvorova (*node*). Izlaz ima samo dvije vrijednosti, svaka odgovara jednom izlaznom neuronu.

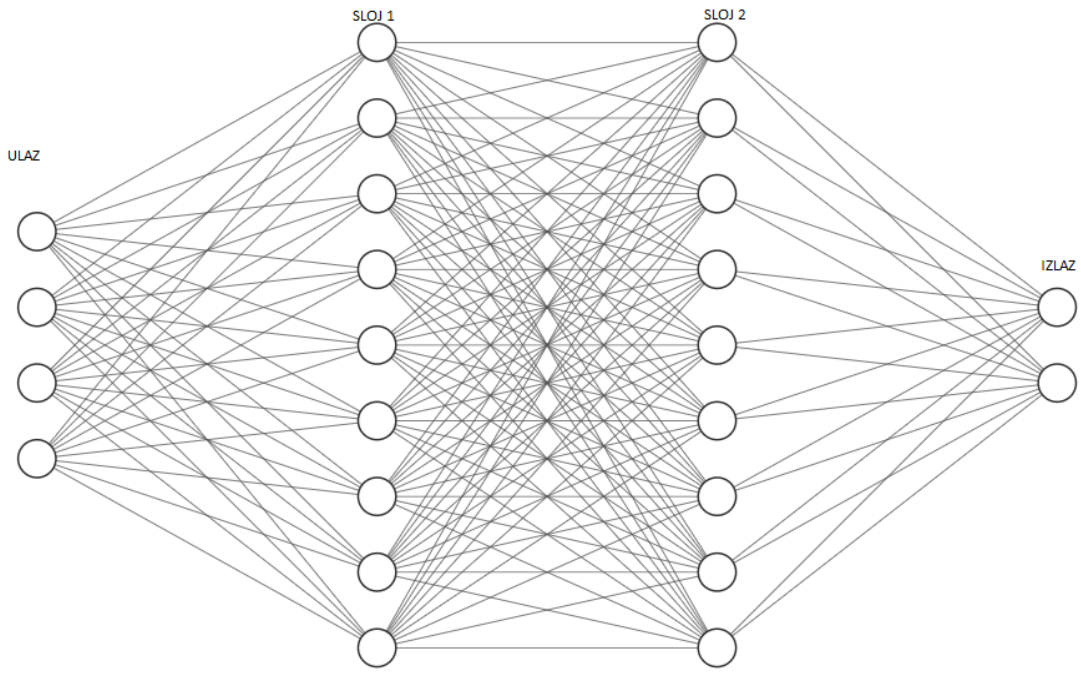
Primjera radi, peti ulaz javlja da je aktiviran senzor1 (1,0,0,0) te očekujemo da će nam vratiti vrijednost izlaza od 1,0.



Slika 14. Proces učenja[7]

Kada smo zadali primjere i rješenja, definiramo slojeve i generiramo neuronsku mrežu:

```
unsigned int layers[] = { 4, 9, 9, 3 };
NeuralNetwork NN(layers, NumberOf(layers));
```



Slika 15. Arhitektura generirane neuronske mreže

Kada se generira neuronska mreža, može se započeti s treningom. U mrežu se *FeedForward* funkcijom predaju primjeri. Zatim se funkcijom *BackProp (BackPropagation)* u mrežu dodaju očekivani rezultati te se prati pogreška. Ovaj proces traje od par do nekoliko tisuća ciklusa. U ovom slučaju stavljeno je 8000 ciklusa nakon kojih će program izbaciti težinske faktore i faktore pomaka.

Sljedeći kod izvučen je iz treninga neuronske mreže. U prvom *for loop-u* vidimo koliko će ciklusa biti izvršeno, a u drugom program prolazi svaki primjer koji smo zadali.

```
for (int i = 0; i < 8000; i++)
{
  for (int j = 0; j < NumberOf(inputs); j++)
  {
    NN.FeedForward(inputs[j]);
    NN.BackProp(expectedOutput[j]);
  }

  Serial.print("Cycle: "); Serial.println(i);
}
```

Isto tako, ispisat će rezultate zadnjeg ciklusa treninga koje možemo vidjeti u tablici 7. Bitno je napomenuti da će neuronska mreža, s obzirom na broj ciklusa, imati određenu pogrešku. Ta se pogreška povećanjem broja ciklusa smanjuje. Nakon 8000 ciklusa ona iznosi oko 2%.

Tablica 7. Rješenja primjera u zadnjem ciklusu

Primjer	Očekivano rješenje		Dobiveno rješenje	
1	0	0	0.0155442	0.0000001
2	0	0	0.0155442	0.0000001
3	0	0	0.0155442	0.0000001
4	0	0	0.0155442	0.0000001
5	1	0	0.9802952	0.0275535
6	1	0	0.9859030	0.0118255
7	1	0	0.9815304	0.0223503
8	1	0	0.9854775	0.0119085
9	0	1	0.0048916	0.9836176

10	0	1	0.0050975	0.9830105
11	0	1	0.0084837	0.9873014
12	0	1	0.0083083	0.9874284
13	0	1	0.0271974	0.9799543

Ovaj proces iznimno je zahtjevan za slabije uređaje. Arduino MKR1000 za ovaj proces treba oko jednog sata, dok ESP32 treba oko pet minuta. Za usporedbu¸ modernom računalu ne trebaju niti sekunde. Jako je bitno napomenuti da se ovaj proces radi samo jednom na početku programa ili se može odraditi na drugom računalu pa vrijednosti faktora direktno ubaciti u program kontrolera koji će se koristiti, što bitno skraćuje vrijeme.

5.2.2. Obrada senzorskih podataka u neuronskoj mreži

Kada je neuronska mreža gotova s učenjem, izbacit će vrijednosti težinskih faktora i faktora pomaka. Težinski faktori su vrijednosti poveznica među neuronima, dok su faktori pomaka vrijednosti utjecaja određenog sloja na konačno rješenje. Kako neuronskoj mreži podaci trebaju biti između 0 i 1 zbog toga što koristi sigmoidnu aktivacijsku funkciju, napon koji koristimo kao ulaz moramo podijeliti s 5.

Iako smo neuronskoj mreži za trening zadali podatke kao 0 ili 1, ona će moći prihvatiti bilokakvu vrijednost između 0 i 1 kao ulaz. Osim prebacivanja na raspon između 0 i 1, potrebno je i filtrirati šum kad senzor nije pritisnut, a koji nastaje zbog nekonzistentnosti napajanja senzora i njegove nepreciznosti. Iz tog razloga filtriraju se vrijednosti ulaza manje od 0.05. Neuronskoj mreži dali smo tri moguća rješenja (0,0) , (1,0) , (0,1), od kojih svako predstavlja jednu naredbu koju želimo prihvatiti. Ukoliko nema pritiska na robotove senzore, neuronska mreža prihvaća vrijednosti 0 sa svakog od četiri senzora. Učenjem je neuronska mreža spremna te vrijednosti pretvoriti u izlaz (0,0), odnosno približnu vrijednost koja će biti filtrirana na (0,0). Ukoliko robot naiđe na čovjeka ili predmet, neki od senzora će se aktivirati te će neuronska mreža vratiti rezultat (1,0) koji ćemo poslati robotu putem WiFi konekcije.

5.3. Slanje podataka prema robotu

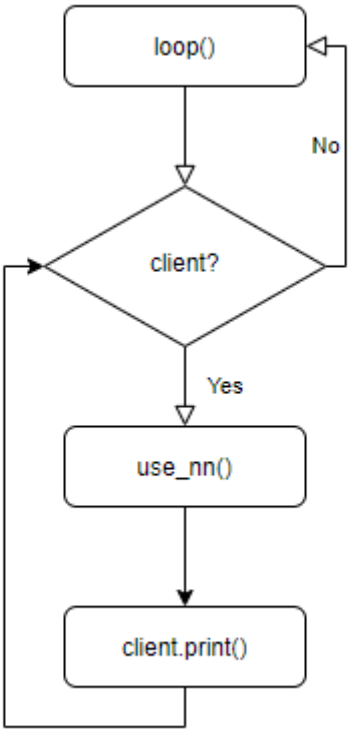
Nakon što su podaci obrađeni, potrebno ih je pripremiti za slanje i poslati. Iako podaci mogu biti poslani u mnogo formata i većem broju znakova, zbog brzine prijenosa i reakcije u ovom slučaju bolje je poslati samo jedan *byte*. Kontroler svakih 10 milisekundi dobije rezultate stanja senzora te ih želimo što prije poslati robotu kako bi isti pravovremeno reagirao. Što je podatak koji šaljemo veći, dulje je i vrijeme slanja, posebno iz razloga što robotu javljamo koliko podataka treba očekivati.

Iz tog razloga, svaka od naredbi bit će poslana u obliku broja (jedan byte nam omogućava brojeve od 0 do 256) koji će biti interpretiran na robotu u naredbu.

U sljedećem komadu koda, vidi se ponavljajuća *loop*() funkcija kontrolera. U njoj nalazimo tri različita segmenta. Prvo i najbitnije je da kontroler ostvari konekciju prema klijentu (*client*). Nakon toga, ukoliko je konekcija ostvarena, kontroler koristi neuronsku mrežu kako bi došao do podataka koje treba poslati (*use_nn*). Naime, *use_nn* funkcija sadrži cijeli dio iščitavanja podataka sa senzora. Dokle god je kliijent spojen (*client.connected()*), kontroler će provjeravati stanje senzora te slati podatke (*client.print(strOutput)*).

```
void loop() {
  WiFiClient client = server.available();
  if (client) {
    while (client.connected()) {
      use_nn();
      if (strOutput != "255") client.print(strOutput);
    }
    client.stop();
    Serial.println("client disonnected");
  }
  delay(1000);
}
```

Isto se može vidjeti u sljedećem dijagramu toka:



Slika 16. Dijagram toka obrade i slanja podataka

6. RoboDK

U ovom poglavlju objasnit će se simuliranje i programiranje robota u programskom paketu RoboDK.

RoboDK, alat je za simulaciju robota. Program nije besplatan, ali se može koristiti u probnoj verziji. Ovaj programski paket vrlo je jednostavan način za isprobavanje rada robota, bez opasnosti od ozljeda ili šteta na pravom robotu. Opcija je mnogo, na robota se mogu staviti razni alati i isprobavati pretprogramirane funkcije kao što su zavarivanje, lijepljenje i slično. Osim toga, u paketu je ugrađen i python *compiler* koji nam omogućava da u simulaciju ubacimo i određene skripte u pythonu koje su pokrivene RoboLink API-em. U ovom radu odabran je robot UR5 kompanije Universal Robots.

6.1. UR5

UR5 kolaborativni je i multifunkcionalni robot. Manjih je dimenzija, maksimalnog opterećenja do 5kg. Vrlo je lagan te jako praktičan za testiranje ovih senzora.

Slika 17. UR5 kolaborativni robot

U tablici niže mogu se vidjeti osnovne specifikacije ovog robota.

Tablica 8. Specifikacije robota UR5

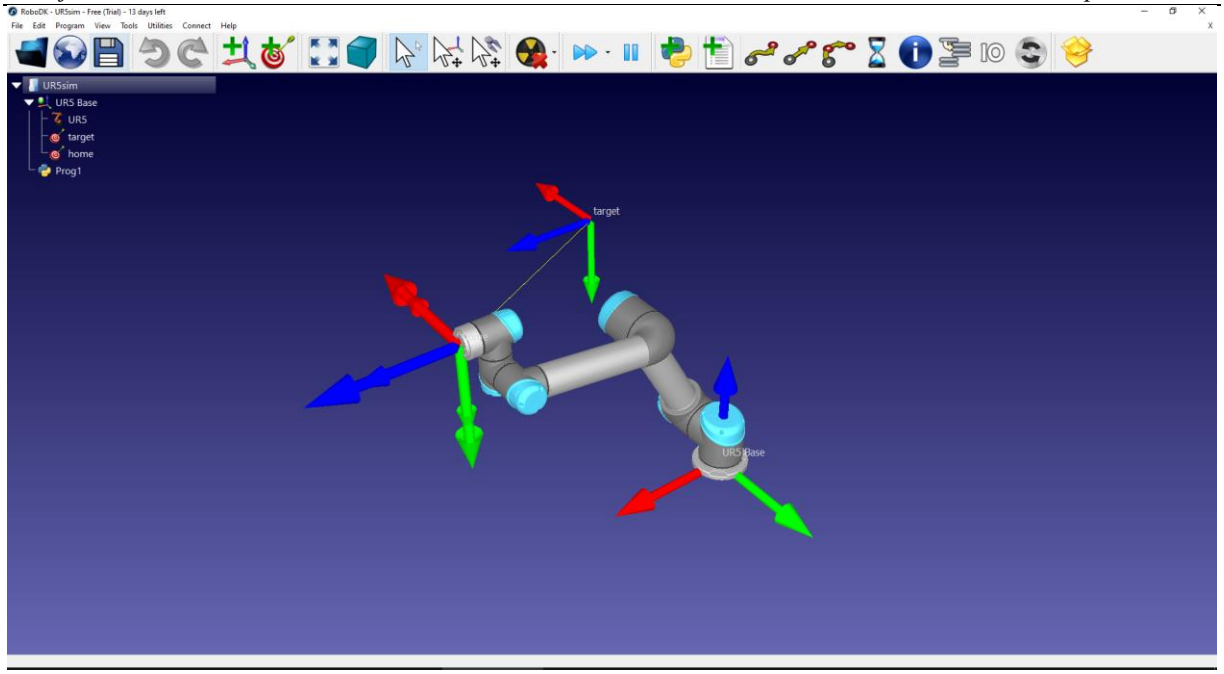
Masa	20.6 *kg*	
Maksimalni teret	5 *kg*	
Doseg	850 *mm*	
Stupnjevi slobode	6	
Osi	Raspon okreta	Maksimalna brzina
• baza	±360°	±180°/s
• rame	±360°	±180°/s
• lakat	±360°	±180°/s
• zglob1	±360°	±180°/s
• zglob2	±360°	±180°/s
• zglob3	±360°	±180°/s
IP standard	IP54	
Komunikacija	Modbus, PROFINET, Ethernet/IP, USB	

Osim gore navedenih osnovnih karakteristika, bitno je napomenuti da robot u sebi ima implementirane sigurnosne značajke. On se, u slučaju nailaska na prepreku, može staviti u *safety stop* stanje. Iako slično funkciji ovih senzora, senzori koji se ugrađuju na ovog robota ipak su dosta finiji te mogu samo privremeno zaustaviti robota. Ukoliko robot ode u *safety stop* stanje, mora ga se ručno opet upaliti preko pripadajućeg tableta za programiranje. S ovim senzorima robot bi samo pauzirao dok se prepreka ne ukloni.

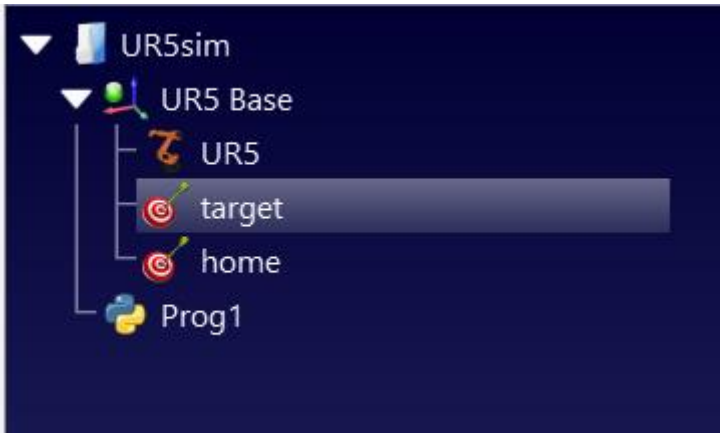
Na ovaj robot mogu se ugraditi mnogi alati kao npr. sisaljke, hvataljke i sl.

6.2. Simulacija UR5 u RoboDK programskom paketu

Kao što je gore navedeno, jedan od robota koji mogu biti simulirani je i UR5. Na slici niže može se vidjeti izgled sučelja RoboDK programa.



Slika 18. RoboDK simulacija robota UR5



Slika 19. komponente simulacije

Na slici 16 vidi se stablo s komponentama simulacije.

- UR5sim – naziv programa
- UR5 Base – referenca na bazu robota
- UR5 – model robota
- target – pozicija na koju želimo pomaknuti glavu robota
- home – početna pozicija glave robota
- Prog1 – programska skripta koja će se izvršiti tijekom simulacije

Iako se robota može programirati i bez skripte, zbog jednostavnosti, preglednosti i pozadinske komunikacije, skripta je uvrštena u program. Skripta je pisana u pythonu i sadrži 2 funkcije koje se vrše u dva *threada,* odnosno dvije niti koje predstavljaju odvojene procese. Naredbe za slušanje komunikacijskog kanala i naredbe za pokret su blokirajuće pa ne mogu biti korištene u jednoj niti.

Skripta se sastoji od tri segmenta:

- postavljanje - dodavanje programskih biblioteka i početne vrijednosti varijabli
- komunikacijska nit – u njoj se vrši komunikacija
- simulacijska nit – u njoj se više kretnje robota i reakcije na dobivene naredbe

Tijekom faze postavljanja, u program se uključuju vanjske biblioteke funkcija, potom se definiraju robot, njegova pozicija i tražene pozicije pomaka (njih se vidi na slici 16. pod *target* i *home*).

```
6   from robolink import *    # RoboDK API
7   from robodk import *      # Robot toolbox
8   import socket
9   import threading
10  RDK = robolink.Robolink()
11  robot = RDK.Item('UR5')      # retrieve the robot by name
12  home = RDK.Item('home')
13  pos2 = RDK.Item('target')
14
15  run = True
16  program=True
17  interrupted = False
18  target = pos2
```

Slika 20. Postavljanje varijabli

Nakon te faze pokreće se prva nit *commThread* u kojoj se vrši komunikacija s kontrolerom. Tu se otvara kanal prema robotu korištenjem naredbe *create_connection* iz biblioteke socket. Ukoliko je uređaj uspješno povezan na kontroler, počinje čitanje podataka. Dok s kontrolera stiže vrijednost 0, robot nastavlja s kretanjem. Ukoliko dođe vrijednost 1, robot je interpretira kao STOP te se poziva zaustavljanje robota u simulacijskoj niti. Na slici 18. može se vidjeti cijela funkcija u kojoj se vrši komunikacija te pokretanje niti *commThread* u kojoj se ta funkcija poziva.

```
#komunikacijska funkcija
def communication():
    global run
    global current
    olddata="999"
    while True:
        rt_socket = socket.create_connection(("192.168.0.25",30000)) #otvaranje komunikacijskog kanala
        rt_socket.setblocking(0)    #postavljanje neblokirajućih funkcija
        buf = b''
        data=''
        while rt_socket:  #provjera stanja konekcije
            data=''
            rt_socket.settimeout(1)
            try:
                data = rt_socket.recv(1).decode("utf-8") #čitanje vrijednosti s kontrolera
            except Exception as e:
                print(e) #ispisivanje greške čitanja
                continue
            finally:
                #interpretacija podataka
                if olddata!=data:
                    if data == '0':
                        print("OK")
                        run=True
                    elif data == '1':
                        print("STOP")
                        run=False
                    elif data == '2':
                        run=False
                        print("DO SOMETHING")
                    elif data == '9':
                        print("UNDEFINED")
                    else:
                        print("data: ",data)
                    olddata=data
        rt_socket.close()
commThread = threading.Thread(target=communication)
commThread.daemon = True
commThread.start()
```

Slika 21. Komunikacijska nit i izvršena funkcija

Simulacijska nit, kao što je navedeno, izvršava kretanje ili zaustavljanje robota ukoliko je to potrebno.

```
60  #simulacijski program
61  while program:
62      #ako nije zaustavljen robot nastavi
63      if run:
64          #ako je kretnja bila prekinuta nastavit prema toj poziciji
65          if interrupted:
66              interrupted=False
67              target = currentMovementTarget
68          robot.SearchL(target,False)
69      #provjeravaj naredbe s kontrolera
70      while robot.Busy():
71          if not run:
72              interrupted = True
73              currentMovementTarget = target
74              robot.Stop()
75      if target == home:
76          target = pos2
77      else:
78          target = home
```

Slika 22. Glavni dio programa - simulacija

7. Primjena sustava

Ovaj koncept primjer je sustava koji omogućava kolaboraciju čovjeka i robota, uz manje nadogradnje i više robota. Jedan ovakav kontroler može se koristit i na dvama robotima u isto vrijeme. Kolaboracija dvaju robota preko jednog kontrolera zahtijevala bi manje nadogradnje zbog povećanja broja senzora. Kao što je navedeno, Arduino ima samo osam analognih ulaza. To se može riješiti uz jedan multiplexer, uređaj koji se ponaša kao prekidač i usmjerava kontroler na čitanje drugog ulaza.

Robot koji je nadograđen s ovim sustavom može sigurno raditi uz čovjeka te bi se, u slučaju dodira, jako brzo zaustavio što bi spriječilo ozljede. Kako bi se dodatno osigurala čovjekova sigurnost, ispod senzora se može staviti tanak sloj spužve. Kašnjenje reakcije robota tako bi se riješilo da se spužva nakon registracije dodira stisne i amortizira udarac. Ovakav sustav omogućava čovjeku da uđe u radni prostor robota bez straha od ozljede. Druga primjena mogla bi biti davanje naredbi robotu taktilnim putem. Recimo, dodirivanjem dvaju ili više senzora u isto vrijeme ili dodirivanjem senzora po nekom uzorku, mogu robotu dati naredbu da stane, zapamti trenutnu poziciju, nauči novu putanju ili slično.

Takav način rada mogao bi olakšati preprogramiravanje robota, možda ga čak i izbjeći. Naredbi u jednom *byte*-u je mnogo, a može se koristiti i više *byteova*.

8. Zaključak

Ovaj rad pokazao je kako se uz primjenu nekoliko senzora i vanjskog kontrolera može nadograditi industrijski robot za primjenu u suradnji s čovjekom.

U uvodu su prikazani osnovni elementi i njihova zadaća u ovom radu. Nakon toga, objašnjeni su detalji istraživanja i izrade senzora pritiska. Senzori koji su napravljeni pokazali su da se vrlo lako mogu implementirati u ovakav sustav te vrlo lako izraditi u veličini i vrsti koja odgovara velikom broju uređaja. Potom su navedeni razlozi odabira određenog kontrolera koji se koristio za izvršenje svoje zadaće. Odabrani su procesorski slabiji kontroleri, ali jednostavnije primjene, što odgovara ovakvom sustavu. Iako nešto sporiji, kontroleri su tijekom učenja mogli odraditi oba zadatka, naučiti prema uzorcima i rješenjima te naučeno uspješno primijeniti kad su se umjesto uzoraka zadavali podaci s pravog senzora. To se može vidjeti u poglavlju o neuronskim mrežama. Učenje je trajalo nešto duže, ali se moglo obaviti i na računalu pa se podaci samo proslijediti kontroleru. Nakon učenja, podaci su obrađivani i slani s malim zaostatkom te se reakcija robota vidjela već nakon desetak milisekundi. Komunikacija je tu imala ključnu ulogu jer je podatke s kontrolera bilo potrebno prebaciti u simulator kako bi se vidjele reakcije robota na naredbe. Simulirani robot u RoboDK paketu programiran u Python programskom jeziku paralelno je slušao nove naredbe i izvršavao simulaciju.
Nakon što su svi parametri sustava profunkcionirali, došlo se do zaključka da je ovakav sustav, iako jednostavan, vrlo skalabilan i iskoristiv u nadogradnji robota. Vrlo bi se jednostavno moglo dodati više senzora ili druge tipove senzora, a neuronska mreža bi uz ponovno učenje bila sposobna reagirati na sve zahtjeve.

9. LITERATURA

[1] Željka Ujević Andrijić (2019), Kemija u industriji : Časopis kemičara i kemijskih inženjera Hrvatske, Vol. 68 No. 5-6, 2019.

[2] FSB robot RONNA u neurokirurgiji, https://100.fsb.hr/hr/119/FSB+robot+RONNA+u+neurokirurgiji+u+Zagrebu, zadnji pristup 22.4.2020.

[3] Pressure pad interface with arduino, https://create.arduino.cc/projecthub/ashish21senapati/pressure-pad-interfacing-with-arduino-efacad, zadnji pristup 20.4.2020.

[4] Chandra, Rohitash (2012), Problem Decomposition and Adaptation in Cooperative Neuro-evolution

[5] Using an FSR, https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr, zadnji pristup: 18.04.2020.

[6] SimpleMLP biblioteka, https://github.com/GiorgosXou/NeuralNetworks, zadnji pristup 18.4.2020.

[7] Neuronske mreže: Uvod, https://www.fer.unizg.hr/_download/repository/01-Uvod[3].pdf, zadnji pristup 20.4.2020.

PRILOZI

I. Programski kod za učenje neuronske mreže

II. Programski kod za primjenu neuronske mreže

III. Programski kod za simulaciju robota

PRILOG 1. Programski kod za učenje neuronske mreže

PrimjenaMreze.ino:

```
#define NumberOf(arg) ((unsigned int) (sizeof (arg) / sizeof (arg [0])))

#include <NeuralNetwork.h>
const int PatternCount = 13;
#define InputNodes 4
#define outputNodes 2
#define Layer1 9
#define Layer2 9
unsigned int layers[] = { InputNodes, Layer1, outputNodes };
float* outputs;

const float inputs[PatternCount][InputNodes] = {
   {0, 0, 0, 0},
   {0, 0, 0, 0},
   {0, 0, 0, 0},
   {0, 0, 0, 0},
   {1, 0, 0, 0},
   {0, 1, 0, 0},
   {0, 0, 1, 0},
   {0, 0, 0, 1},
   {1, 1, 0, 0},
   {1, 0, 0, 1},
   {0, 1, 1, 0},
   {0, 0, 1, 1},
   {1, 0, 1, 0}
};

const float expectedOutput[PatternCount][outputNodes] = {
   {0,0},
   {0,0},
   {0,0},
   {0,0},
   {1,0},
   {1,0},
   {1,0},
   {1,0},
   {0,1},
   {0,1},
   {0,1},
   {0,1},
   {0,1}
};
void setup()
{
   Serial.begin(115200);
   NeuralNetwork NN(layers, NumberOf(layers));
   for (int i = 0; i < 8000; i++)
```

```
    {
      for (int j = 0; j < NumberOf(inputs); j++)
      {
        NN.FeedForward(inputs[j]);
        NN.BackProp(expectedOutput[j]);
      }

      Serial.print("Cycle: "); Serial.println(i);
    }
    for (int i = 0; i < NumberOf(inputs); i++)
    {
      outputs = NN.FeedForward(inputs[i]);
      for (int o = 0; o < outputNodes; o++)
      {
        Serial.print(outputs[o], 7);
        Serial.print(",");
      }
      Serial.println();
    }
    NN.print();
}
void loop() {

}
```

PRILOG 2. Programski kod za primjenu neuronske mreže

UcenjeMreze.ino:

```
#define NumberOf(arg) ((unsigned int) (sizeof (arg) / sizeof (arg [0])))

#include <NeuralNetwork.h>
const int PatternCount = 13;
#define InputNodes 4
#define outputNodes 2
#define Layer1 9
#define Layer2 9
unsigned int layers[] = { InputNodes, Layer1, outputNodes };
float* outputs;

const float inputs[PatternCount][InputNodes] = {
  {0, 0, 0, 0},
  {0, 0, 0, 0},
  {0, 0, 0, 0},
  {0, 0, 0, 0},
 { 1, 0, 0, 0 },
 { 0, 1, 0, 0},
 { 0, 0, 1, 0},
 { 0, 0, 0, 1 },
 { 1, 1, 0, 0 },
 { 1, 0, 0, 1 },
 { 0, 1, 1, 0 },
 { 0, 0, 1, 1 },
 { 1, 0, 1, 0 }
};

const float expectedOutput[PatternCount][outputNodes] = {
   {0,0},
   {0,0},
   {0,0},
   {0,0},
   {1,0},
   {1,0},
   {1,0},
   {1,0},
   {0,1},
   {0,1},
   {0,1},
   {0,1},
   {0,1}
};
void setup()
{
   Serial.begin(115200);
```

```
  NeuralNetwork NN(layers, NumberOf(layers));
  for (int i = 0; i < 8000; i++)
  {
    for (int j = 0; j < NumberOf(inputs); j++)
    {
      NN.FeedForward(inputs[j]);
      NN.BackProp(expectedOutput[j]);
    }

    Serial.print("Cycle: "); Serial.println(i);
  }
  for (int i = 0; i < NumberOf(inputs); i++)
  {
    outputs = NN.FeedForward(inputs[i]);
    for (int o = 0; o < outputNodes; o++)
    {
      Serial.print(outputs[o], 7);
      Serial.print(",");
    }
    Serial.println();
  }
  NN.print();
}
void loop() {

}
```

PRILOG 3. Programski kod za simulaciju robota

Prog1.py:

```
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox
import socket
import threading
RDK = robolink.Robolink()
robot = RDK.Item('UR5')      #poveznica na simuliranog robota
home = RDK.Item('home')      #poveznica na početnu poziciju
pos2 = RDK.Item('target')    #poveznica na željenu poziciju

run = True
program=True
interrupted = False
target = pos2

#komunikacijska funkcija
def communication():
    global run
    global current
    olddata="999"
    while True:
        rt_socket = socket.create_connection(("192.168.0.25",30000)) #otvaranje komunikacijsk
og kanala
        rt_socket.setblocking(0)    #postavljanje neblokirajućih funkcija
        buf = b''
        data=''
        while rt_socket:  #provjera stanja konekcije
            data=''
            rt_socket.settimeout(1)
            try:
                data = rt_socket.recv(1).decode("utf-8") #čitanje vrijednosti s kontrolera
            except Exception as e:
                print(e) #ispisivanje greške čitanja
                continue
            finally:
                #interpretacija podataka
                if olddata!=data:
                    if data == '0':
                        print("OK")
                        run=True
                    elif data == '1':
                        print("STOP")
                        run=False
                    elif data == '2':
                        run=False
                        print("DO SOMETHING")
```

```
                elif data == '9':
                    print("UNDEFINED")
                else:
                    print("data: ",data)
                olddata=data
        rt_socket.close()
commThread = threading.Thread(target=communication)
commThread.daemon = True
commThread.start()

#simulacijski program
while program:
    #ako nije zaustavljen robot nastavi
    if run:
        #ako je kretnja bila prekinuta nastavit prema toj poziciji
        if interrupted:
            interrupted=False
            target = currentMovementTarget
        robot.SearchL(target,False)
    #provjeravaj naredbe s kontrolera
    while robot.Busy():
        if not run:
            interrupted = True
            currentMovementTarget = target
            robot.Stop()
    if target == home:
        target = pos2
    else:
        target = home
```