Mehatronički model šake

Kostanjevac, Karlo

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:354841

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-23**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Stručni studij

MEHATRONIČKI MODEL ŠAKE

Završni rad

KARLO KOSTANJEVAC

Osijek, 2017.

1 UVOD ... 3
1.1 Zadatak i organizacija rada ... 3
2 **MEHATRONIČKI MODEL ŠAKE ... 4**
2.1 Modeliranje sustava ... 4
2.2 Mehatronički pogon ... 7
2.3 Fizički model mehatroničke šake ... 12
2.4 Upravljački i komunikacijski računalni sustav ... 13
2.5 Algoritam upravljanja ... 18
3 UPRAVLJANJE MEHATRONIČKIM MODELOM ŠAKE ... 21
3.1 Upravljačka rukavica ... 21
3.2 Aplikacijsko upravljanje modelom šake ... 32
4 TESTIRANJE I REZULTATI ... 36
4.1 Test kašnjenja pokreta mehatroničke šake kontrolirane upravljačkom rukavicom ... 36
4.2 Test točnosti pozicioniranja mehatroničke šake upravljane upravljačkom rukavicom 37
5 ZAKLJUČAK ... 40
LITERATURA ... 41
SAŽETAK ... 42
ABSTRACT ...
ŽIVOTOPIS ...
PRILOZI ...

1 UVOD

1.1 Zadatak i organizacija rada

U današnje vrijeme tehnologija se sve više primjenjuje u svakodnevnom životu, industriji, medicini, obrazovanju, prometu i slično. Česti primjer tehnologije u industriji i medicini je korištenje robotskih ruku. Robotske ruke danas uvelike zamjenjuju ljude ili koriste kao alat ljudima zato što mogu obavljati radnje koje su izvan fizičkih mogućnosti čovjeka a sa time postižu i povećanje kvalitete proizvoda ili radnje koju obavljaju. Robotske ruke u industriji koriste se za različite potrebe kao što su zavarivanje, montaža dijelova, lakiranje i slično. U medicini se također koriste za različite svrhe kao što su precizne operacije, protetika, asistencija pri operaciji i slično. Prednosti robotskih ruku, ovisno o području upotrebe su povećanje preciznosti radnji, povećanje produktivnosti, povećanje proizvodnje, ušteda energije i ekonomska isplativost.

U završnom radu potrebno je projektirati, izraditi i testirati sustav mehatroničkog modela šake. Omogućiti bežično upravljanje mehatroničkog modela šake sa upravljačkom rukavicom te računalnom aplikacijom. Mehatroničkog model šake mora izgledati i imati slične funkcije kao humanoidna šaka.

U ovom završnom radu biti će objašnjena izrada humanoidnog oblika mehatroničke šake, te dva sustava za upravljanje mehatroničke šake, upravljačku rukavicu i aplikaciju za računalo. Modeliranje sustava mehatroničke šake odvijat će se na osnovu analize dimenzija stvarne ljuske šake te dimenzija mehatroničkog pogona. Upravljački računalni sustav će pomoću algoritma za upravljanje procesirati primljene podatke i upravljati mehatroničkim pogonom. Upravljačka rukavica će biti modelirana na osnovu analize stvarne ljudske šake i sadržavati će senzore za detekciju položaja prsta. Upravljački računalni sustav rukavice će na osnovu algoritma za upravljanje očitavati vrijednosti senzora te ih procesirati i slati preko komunikacijskog sustava. Aplikacija za upravljanje mehatroničke šake će biti algoritam koji od korisnika prima podatke te ih šalje na mehatroničku šaku preko komunikacijskog sustava.

2 MEHATRONIČKI MODEL ŠAKE

2.1 Modeliranje sustava

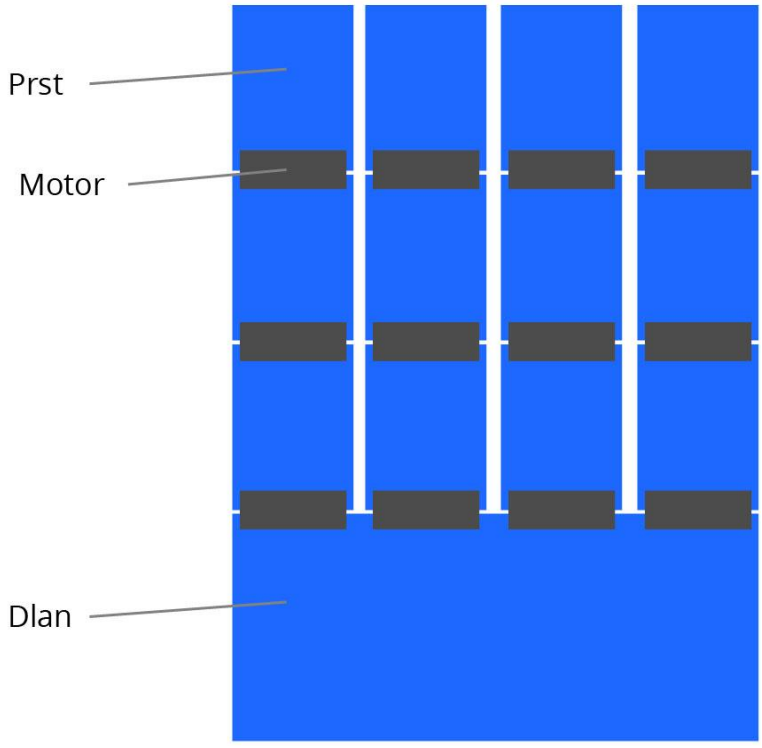
Šaka je dio ljudskog tijela koji sliži za hvatanje predmeta. Pri modeliranju mehatroničke šake napravljen je teorijski model mehatroničke šake te humanoidni model. Za potrebe izrade uzeti su uzorci veličina ljudskog dlana i prstiju te na osnovu dobivenih uzoraka i dimenzija motora napraviti mjere za fizički model šake. Mehatronički model šake ima funkciju zasebnog pomicanja svakog zgloba prstiju, osim prvog zgloba palca koji je izostavljen zbog ograničenosti izvedbe.

2.1.1 Teorijski model mehatroničke šake

Kako bi se dobila osnova za humanoidno modeliranje mehatroničke šake potrebno je napraviti teorijski model. Osnovni dijelovi mehatroničkog modela šake su dlan i prsti.

Oblik mehatroničke šake ovisi o obliku i količini motora koji pokreče zglobove te slobodi kretanja prstiju šake. Prsti šake mogu se kretati u jednom smjeru te je za takav način kretanja potreban jedan motor u svakom zglobu prsta.

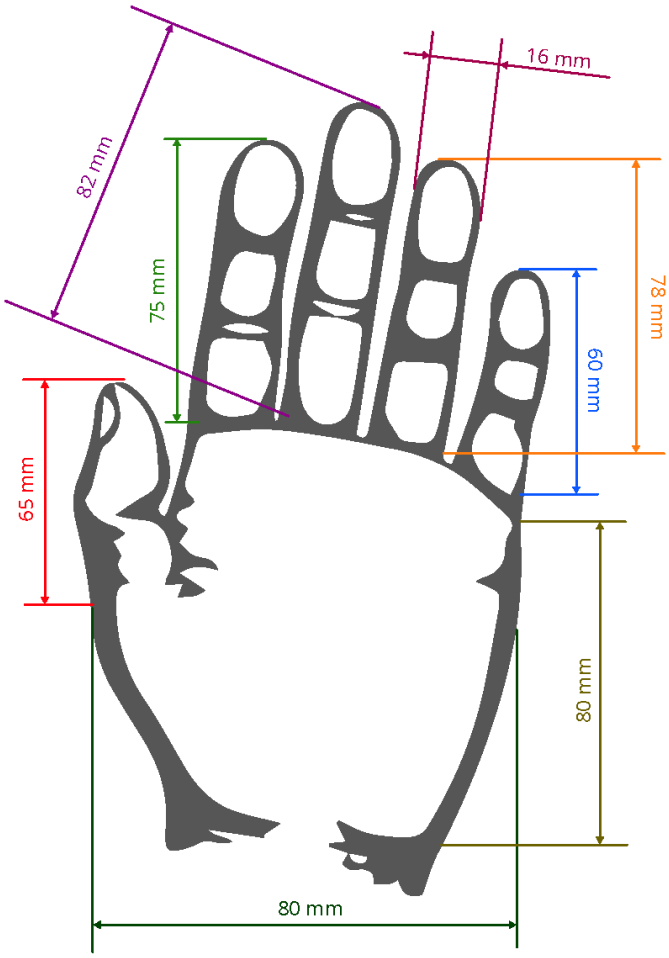
Teorijski model (Slika 2.1. Teorijski model šake) prikazuje raspored motora po zglobovima prstiju.



Slika 2.1 Teorijski model šake

2.1.2 Humanoidno modeliranje šake

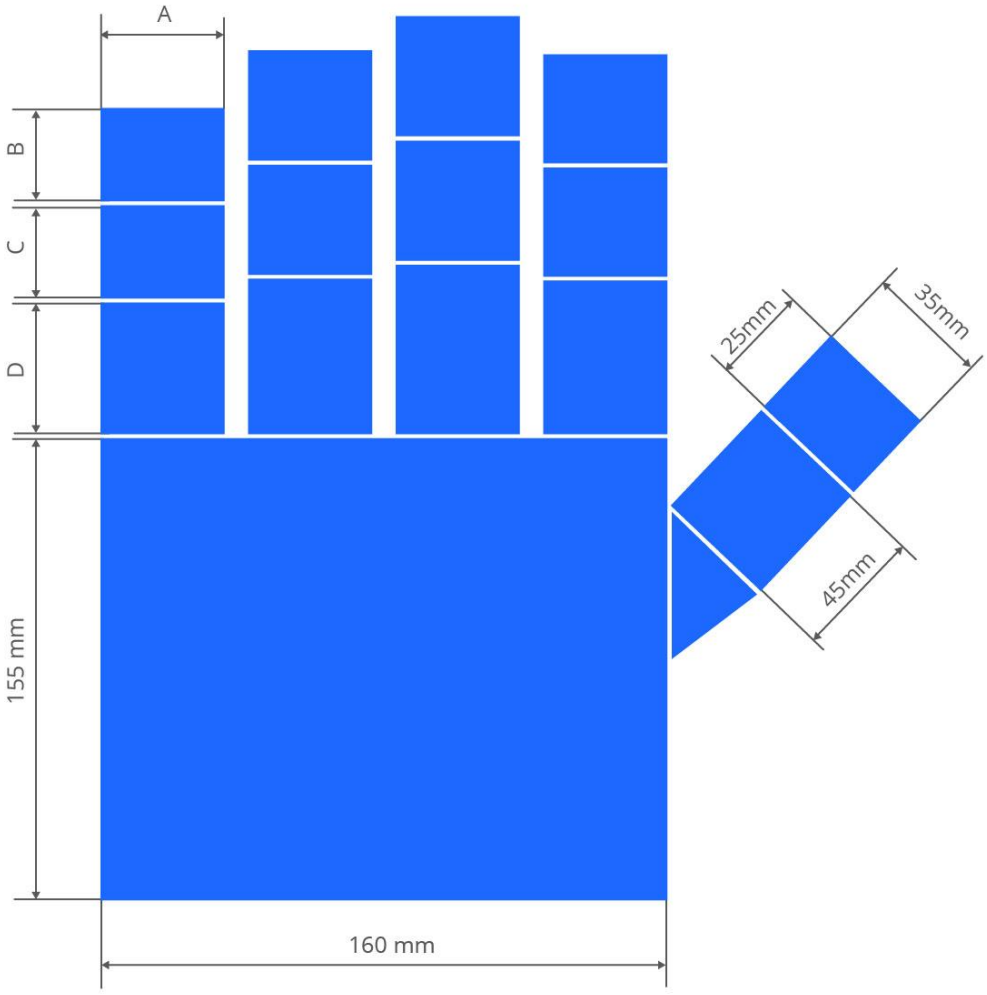
Ljudska šaka se sastoji od 27 kostiju, sa metakarpalnim kostima i člancima, kojih u svakom prstu ima po tri (izuzet palca sa samo dvije). Za potrebe projektiranja mehatroničke šake zbog ograničenosti izvedbe, kostur ljudske šake predstavljen je šakom sa četrnaest zglobova prstiju koji se mogu pomicati u jednom smjeru, prema dlanu.



Slika 2.2 Uzimanje uzoraka ljudske šake, [1]

Uzimanje uzoraka obavljeno je pomičnim mjerilom. Rezultati mjerenja mogu se vidjeti na slici 2.2, te se iz dobivenih rezultata mogu postaviti omjeri visine prstiju, širine dlana i debljine prstiju koji će koristiti za modeliranje mehatroničke šake.

Izračunate mjere nalaze se u tablici 2.1. Mjere mehatroničkog modela šake i na slici 2.3. Mehatronički mode šake.



Slika 2.3 Mehatronički model šake

Tablica 2.1 Mjere mehatroničkog modela šake

	A	B	C	D	Debljina
Kažiprst	35mm	30mm	40mm	60mm	30mm
Srednjak	35mm	35mm	43mm	72mm	30mm
Prstenjak	35mm	33mm	38mm	60mm	30mm
Mali prst	35mm	30mm	35mm	52mm	30mm

2.2 Mehatronički pogon

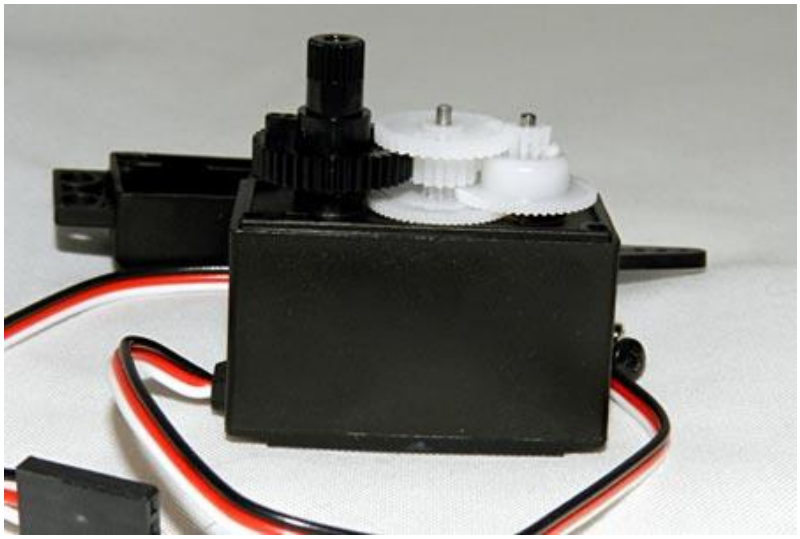
Zbog ograničenosti izvedbe mehatronička šaka ima omogućeno kretnje prstiju u jednom smjeru, prema dlanu. Mehatronička šaka pri stisku treba imati snagu dovoljnu za pomicanje prstiju tj. demonstraciju pokreta a ne za obavljanje radnji pomicanja predmeta i slično. Zbog navedenih svojstava, u izradi mehatroničke šake korišteni su mini servo motori.

2.2.1 Servo motori

Servo motori (slika 2.) su uređaji koji rotiraju ili guraju dijelove stroja određenom silom, brzinom i udaljenosti. Servo motori se primjenjuju u područjima od igračka, kućne elektronike do automobila i zrakoplova. Jednostavnost servo motora je jedna od značajka koja ih čini toliko pouzdanima.

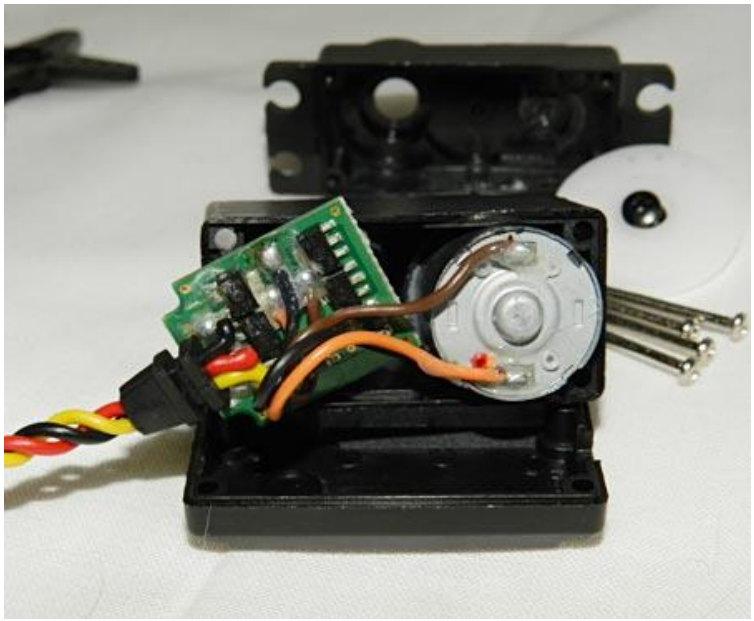
Slika 2.4 Servo motor

Pogon servo motora je istosmjerni motor koji ima karakteristike visokog broja okretaja u minuti i malu snagu okretnog momenta. Istosmjerni motor pokreće niz zupčanika, Slika 2.5 Zupčanici u servo motoru, koji smanjuje brzinu okretanja te podižu okretni moment, sa time je količina stvarnog rada ista ali korisnija.



Slika 2.5 Zupčanici u servo motoru, [2]

Na konačnom zupčaniku nalazi se senzor koji očitava poziciju izlaznog vratila. Senzor šalje signal u tiskanu pločicu koji govori koliko je izlazno vratilo zakrenuto. Elektronika na pločici dekodira signale senzora kako bi se utvrdilo koliko je izlazno vratilo zakrenuto, zatim uspoređuje položaj dobiven dekodiranjem signala senzora sa željenim položajem koji definira korisnik, te odlučuje u kojem smjeru okretati osovinu kako bi se dobio željeni položaj.



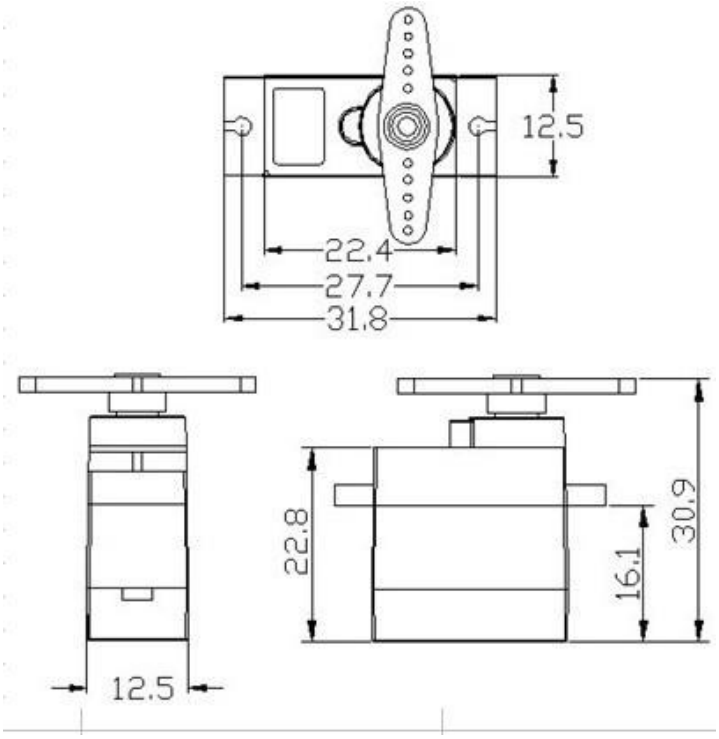
Slika 2.6 Tiskana pločica servo motora, [2]

2.2.2 Mini servo SG90

Za potrebe pomicanja prstiju mehatroničke šake koristi se mini servo SG90 servo motor zbog vrlo malih dimenzija, težine, dovoljnog zakretnog momenta i brzine.

[3] Tehničke specifikacije SG90:

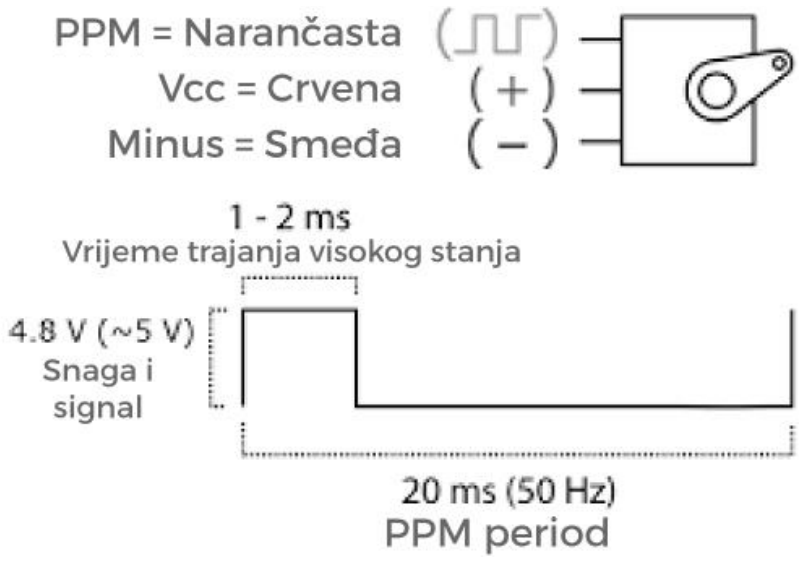
- Težina: 9 g
- Dimenzije: Slika 2.7 Dimenzije SG90
- Zakretni moment: 0.1765 Nm
- Brzina pomaka: 0.1 s/60 stupnjeva
- Radni napon: 4.8 V (~5V)
- Raspon radne temperature: 0 ºC – 55 ºC



Slika 2.7 Dimenzije SG90, [4]

SG90 ima tri ulaza, koji služe za signal i napajanje, slika 2.8 prikazuje ulaze i oblik PPM signala. Pozicija izlaznog vratila određuje se pulsno pozicijskim moduliranim signalom na PWM ulazu (slika 2.8). PPM signal šalje se frekvencijom od 50 hz, pomak vratila se određuje vremenom trajanja visokog stanja. Pozicija 0 stupnjeva odgovara 1.5 ms trajanja visokog

stanja, pozicija 90 stupnjeva odgovara 2ms trajanja visokog stanja, pozicija -90 stupnjeva odgovara 1ms trajanju visokog stanja.

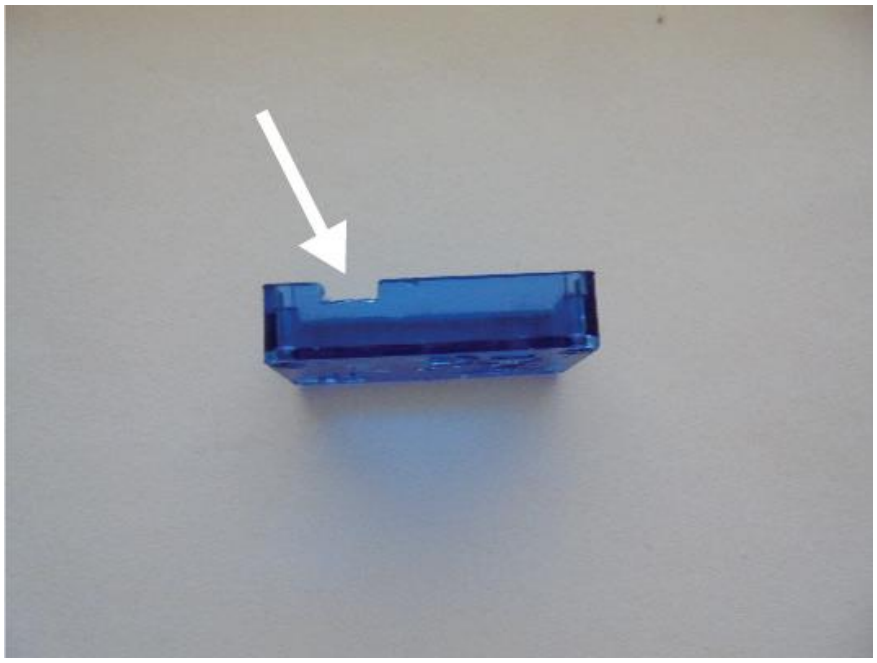


Slika 2.8 Ulazi i oblik PWM signala, [4]

2.2.3 Modifikacija Mini servo SG90 motora

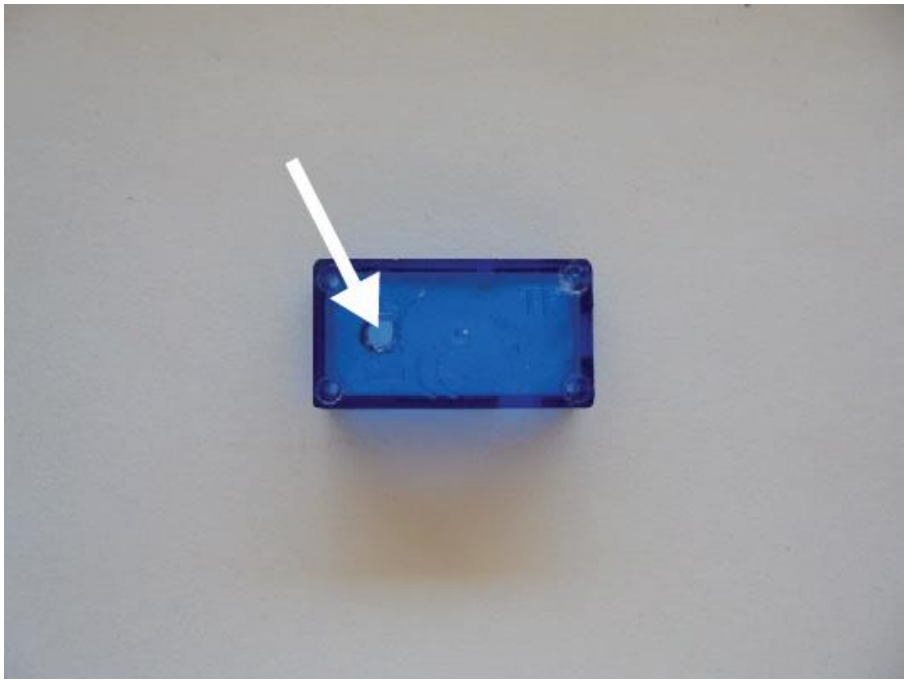
Kako bi SG90 mogao obavljati funkciju pomicanja prstiju potrebno je napraviti modifikaciju na samom kućištu servo motora. Potrebno je napraviti dvije modifikacije, premještanje ulaza na drugu stranu motora te dodavanje nepomičnog vratila na suprotnu stranu glavnog, izlaznog vratila servo motora.

Modifikacija premještanja ulaza , Slika 2.9 Modifikacija premještanja ulaza, na drugu stranu motora je potrebna zbog pozicije motora u prstu mehatroničke šake zbog toga što tvornički postavljani ulazi ne odgovaraju za ovakav položaj motora. Modifikacija je obavljena rastavljanjem servo motora te rezanjem utora na odgovarajućoj strani kućišta motora za žice ulaza signala i napajanja.



Slika 2.9 Modifikacija premještanja ulaza

Modifikacija dodavanje vratila, Slika 2.10 Modifikacija dodavanja nepomičnog vratila, na suprotnu stranu izlaznog vratila servo motora potrebna je za ležaj koji se nalazi na prstu mehatroničke ruke i izvedena je rastavljanjem motora, bušenjem rupe na odgovarajuće mjesto te montiranje valjkastog komada željeza u izbušenu rupu te učvršćivanje sa epoxy ljepilom.



Slika 2.10 Modifikacija dodavanja nepomičnog vratila

2.3 Fizički model mehatroničke šake

Na osnovu dimenzija dobivenih iz teorijskog modela mehatroničke šake i dimenzija servo motora, modeliran je fizički model mehatroničke šake. Odabir materijala za izradu ovisi o traženim fizikalnim i mehaničkim svojstvima te pristupačnosti opreme za njegovu obradu. Materijal korišten u izradi je polimetilakrilat, poznat kao i pleksiglas.

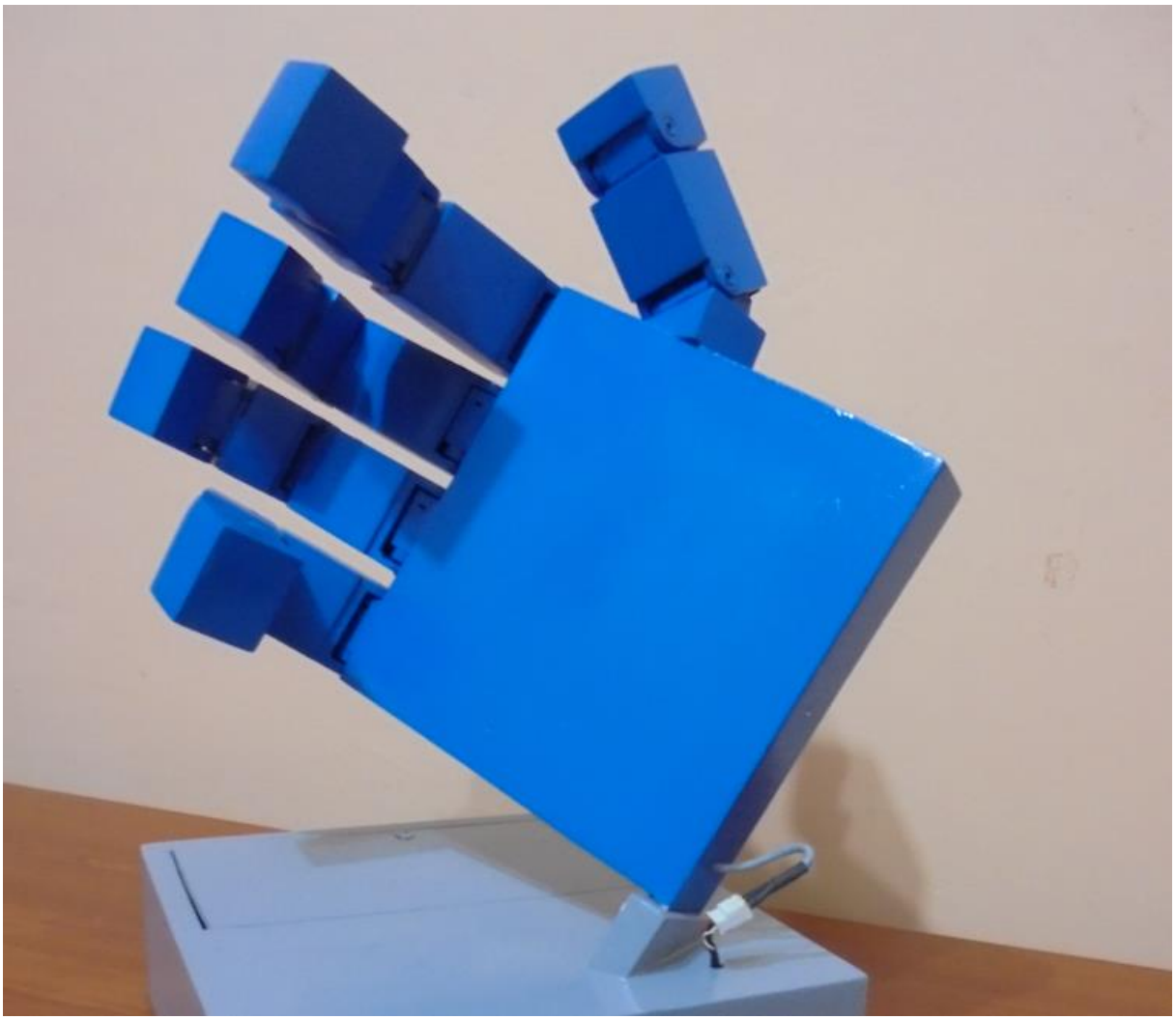
2.3.1 Pleksiglas

Pleksiglas je jak, tvrd i lagan materijal sa gustoćom od 1.17 do 1.20 g/cm^3 što ga dobrim materijalom za izradu mehatroničke šake.

Mehanička obrada pleksiglas obavlja se pomoću alata i strojeva za obradu metala i drveta. Moguće ga je pilit, glodati, tokariti, polirati, bušiti i brusiti. Spajanje pleksiglasa može se izvesti superljepilom, toplinskim varenjem ili kemijskim varenjem a rezanje navedenim tehnikama te dodatnim tehnikama kao što je lasersko rezanje.

2.3.2 Izrada fizičkog modela mehatroničke šake

Pleksiglas se prema dobivenim dimenzijama iz teorijskog modela mehatroničke šake reže pilom za željezo na pločice potrebnih dimenzija. Nakon rezanja, svim izrezanim dijelovima obrađuju se rubovi kako bi se uklonile nepravilnosti i ostatci od rezanja. Pločice se tada ljepe superljepilom i lakiraju lak bojom te se dobije fizički model mehatroničke šake sa slike 2.11



Slika 2.11 Fizički model mehatroničke šake

2.4 Upravljački i komunikacijski računalni sustav

Mehatronički model šake za komunikaciju i upravljanje servo motora, sa time i prstima šake, treba upravljačku jedinicu. Za upravljačku jedinicu koristi se arduino mega mikrokontroler.

2.4.1 Arduino mega

Arduino mega [5], slika 2.11, je mikrokontroler sa 54 digitalnih ulazno / izlaznih pinova (od kojih 14 može se koristiti kao PWM izlaza), 16 analognih ulaza, 4 UARTa (hardver serijski portovi), 16 MHz kristal oscilator, USB priključak, ulaz za napajanje i gumb za resetiranje.

Slika 2.11 Arduino mega

Specifikacije arduino mega:

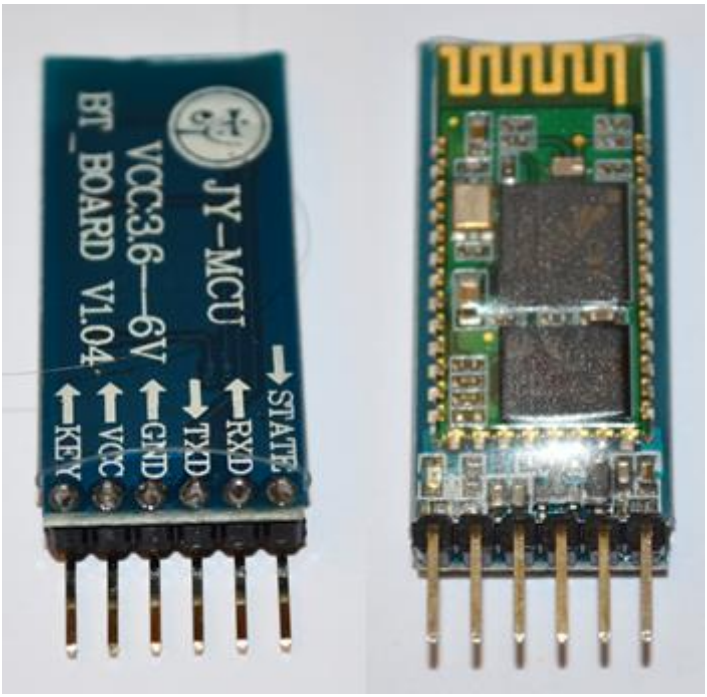
- Mikrokontroler: ATmega1280
- Radni napon: 5V
- Ulazni napon (preporučeni): 7-12V
- Ulazni napon (granice): 6-20V
- Digitalni I/O pinovi: 54 (od kojih 15 omogućavaju PWM izlaze)
- Pinovi za analogne ulaze: 16
- Istosmjerni iznos struje po I/O pinu: 40 mA
- Istosmjerni iznos struje za 3.3V po pinu: 50 mA
- Flash memorija: 128 KB
- SRAM: 8 KB
- EEPROM: 4 KB
- Brzina procesora: 16 MHz

2.4.2 Komunikacijski sustav

Komunikacija prema mehatroničkom modelu šake obavlja se isključivo bežično preko bluetooth veze. Za uspostavljanje bežične veze koristi se arduino bluetooth modul HC05 [6].

HC 05, slika 2.13, je jednostavan bluetooth SPP (Serial Port Protocol) modul namijenjen za uspostavu bežične serijske veze.

Bluetooth modul je u potpunosti osposobljen Bluetooth V2.0 + EDR (Enhanced Data Rate) 3 Mb modulacija s potpunim 2.4GHz radio primopredajnikom.



Slika 2.13 HC 05

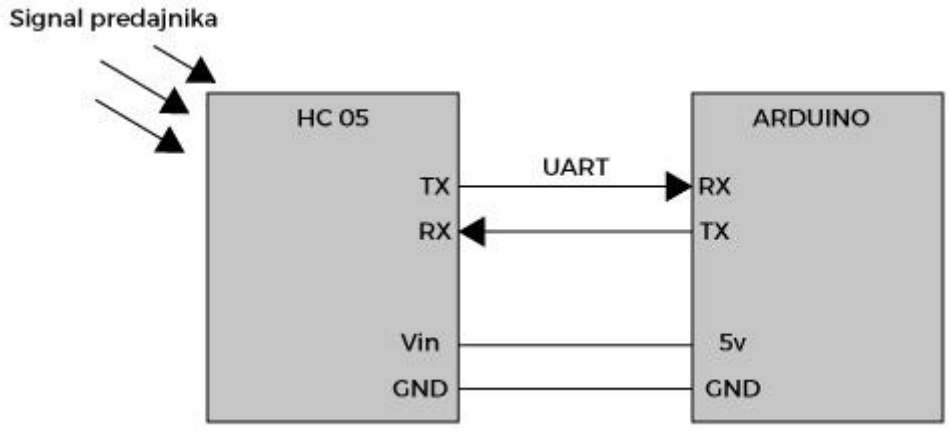
Hardware specifikacije:

- -80dBm osjetljivost
- Do +4dBm RF snaga odašiljanja
- PIO kontrola
- UART sučelje sa promjenjivim frekvencijskim područjem
- Integrirana antena

Software specifikacije:

- Tvornički postavljeni Baud rate: 38400, podatkovni bitovi:8stop bitovi:1, parity: ne, kontrola podataka: da.
- Podržava baud rate: 9600,19200,38400,57600,115200,230400,460800.
- Sa pozitivnim pulsom na PIO0 veza će biti prekinuta.
- PIO10 i PIO11 mogu biti spojeni na crvenu i plavu led diodu odvojeno neovisno o master i slave modu
- Kada je veza ostvarena, crvena i plava led dioda svijetle jednom u intervalu od dvije sekunde, dok veza nije ostvarena samo plava dioda svijetli dva puta u sekundi.
- Tvornički postavljeno automatsko spajanje na zadnji spojeni uređaj.

HC 05 bluetooth modul i arduino mega zajedno čine komunikacijski sustav mehatroničkog modela šake. Na blok dijagramu, slika 2.14, je prikazana veza između HC 05 modula i arduina. Bluetooth modul prima signal predajnika i preko pina za serijsku komunikaciju (TX) šalje primljene podatke u UART obliku na pin za primanje podataka (RX) na arduinu. HC 05 bluetooth modul koristi arduino za napajanje preko Vin (ili Vcc) i GND ulaza.



Slika 2.14 Veza HC 05 bluetooth modula i arduina

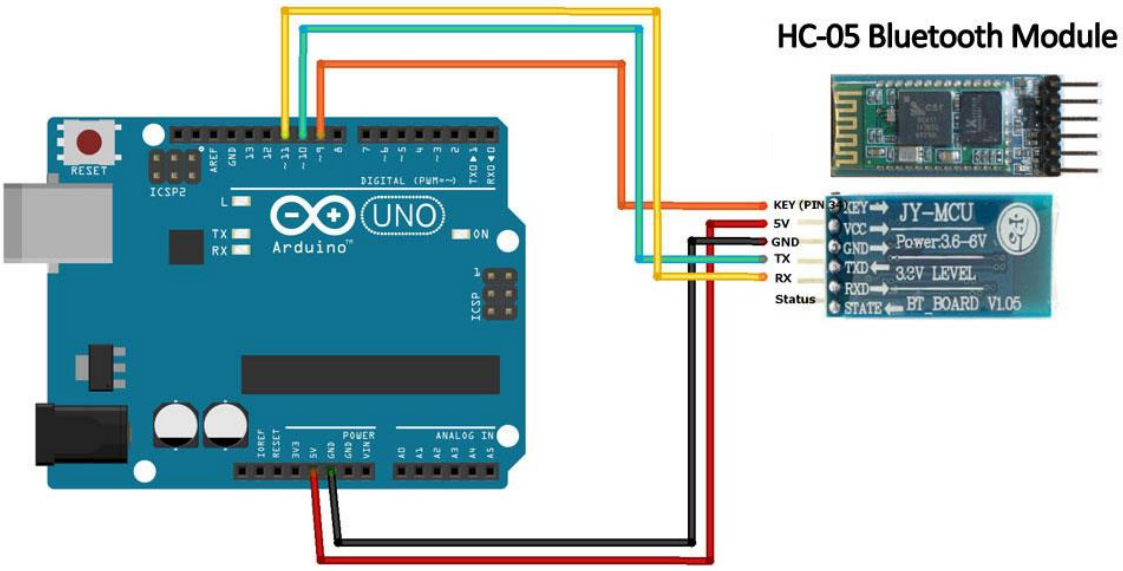
2.4.3 Podešavanje HC 05

1. Ulazak u AT mod

Za ulazak u mod podešavanja potrebno je spojiti bluetooth modul na arduino te arduino spojiti sa računalom.

Spajanje se izvodi na sljedeći način, Slika 2.15 Spajanje modula za AT način rada :

- **KEY** - Spaja se na 5 volti.
- **VCC** - Kao što je naznačeno spaja se na raspon od 3.6V do 6V.
- **GND** – spaja se na GND pin na arduino.
- **TXD**: serijski *output* od modula mora biti spojen na RX pin arduina.
- **RXD**: serijski *input* od modula mora biti spojen na TX pin arduina.



Slika 2.15 Spajanje modula za AT način rada

2. Unos AT komandi

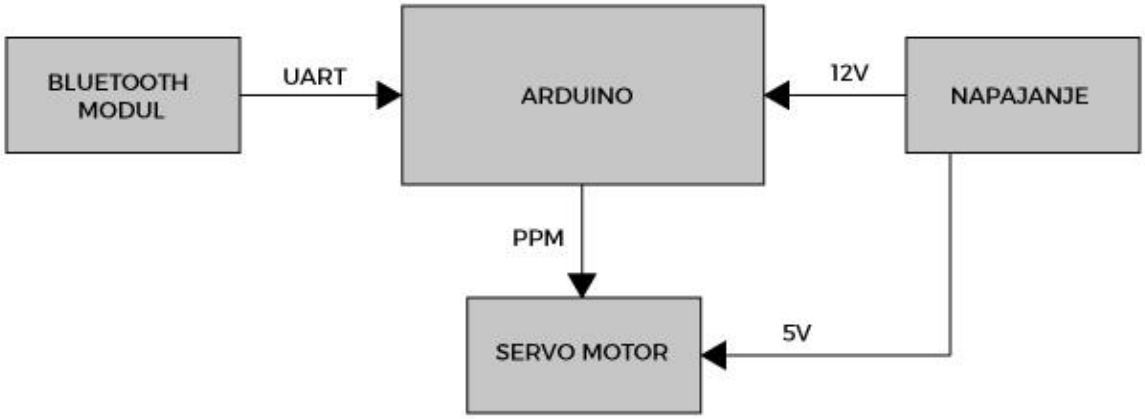
Pomoću AT komandi potrebno je postaviti ime, pin te baud rate modula. AT komande unose se preko računala na arduino desktop aplikaciji pomoću prozora "serial monitor".

- Postavljanje imena „mehatronickasaka" pomoću komande AT+NAMEmehatronickasaka
- Postavljanje šifre „1234" pomoću komande AT+PIN1234

- Postavljanje baud rate-a na 38400 pomoću komande AT+BAUD6

2.4.4 Blok shema sustava

Arduino i servo motori napajaju se sa napajanja koje je smješteno u bazi mehatroničke šake. Napajanje je istosmjerni izvor 12 i 5 volti. Arduino se napaja sa 12 volti a servo motori sa 5 volti. Bluetooth modul je spojen sa arduino računalnim sustavom na način da je ostvarena UART podatkovna veza i napajanje modula. Arduino računalni sustav je povezan sa svakim servo motorom tako da može upravljati njegovim položajem, upravljanje se odvija pomoću PPM signala. Pojednostavljena blok shema prikazana je na slici 2.16 na kojoj je prikaz spoja bluetooth modula, napajanja, arduina i servo motora. Kompletna shema spoja nalazi se u prilogu br. 3.



Slika 2.16 Blok shema sklopa

2.5 Algoritam upravljanja

Mikroprocesor i drugi logički sklopovi imaju svoj vlastiti programski jezik koji se naziva strojni jezik [7], a sastoji se od nizova binarnih riječi koje predstavljaju instrukcije logičkim sklopovima i podatke koje treba obraditi. Program napisan u strojnom jeziku se naziva izvršni program ili izvršni kod budući da ga mikroprocesor može neposredno izvršiti. Strojni jezik je određen arhitekturom mikroprocesora, a definira ga proizvođač hardwarea. Izvršni program je strojno zavisan, što znači da se kod napisan na jednom računalu može izvršavati jedino na računalima istog tipa.

Pisanje instrukcija u binarnom kodu posve je nepraktično pa su razvijeni simbolički jezici u kojima su binarne instrukcije zamijenjene mnemoničkim oznakama. Programer unosi program napisan u mnemoničkim oznakama u tekstualnu datoteku pomoću editora teksta i zatim poziva program koji mnemoničke oznake prevodi u binarne instrukcije strojnog jezika. Program koji vrši konverziju naziva se asembler, a sam se programski jezik naziva asemblerski jezik ili jednostavno asembler.

Viši programski jezici (C, Pascal, FORTRAN, C++, Java, Perl, Python, . . .) razvijeni su kako bi se prevladali nedostaci asemblerskog jezika. Oni oslobađaju programera potrebe poznavanja arhitekture računala ili mikroprocesora te omogućavaju prenosivost programa između mikroprocesora različitih arhitektura te brže i jednostavnije programiranje.

Programski jezik C viši je programski jezik opće namjene i jedna od primjena je programiranje arduino mikrokontrolera.

Svi prsti mehatroničke šake upravljaju se na isti način. Prvo se deklariraju potrebne varijable za određeni prst pa se zatim otvara serijska veza te čitaju podaci sa serijskog porta. Pročitani podaci spremaju se u odgovarajuće varijable za daljnju obradu.

```
int PinkyOne = 0;
int PinkyTwo = 0;
int SavePinkyOne = 0;
int SavePinkyTwo = 0;
int PinkyOneFix = 50;
int PinkyTwoFix = 20;
int PinkyTwooFix = 60;
BTserial.begin(9600);
PinkyOne = BTserial.read();
 PinkyTwo = BTserial.read();
```

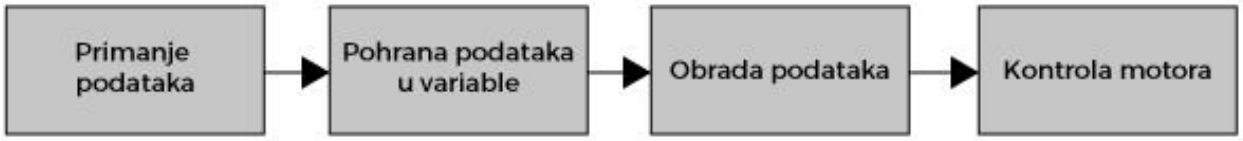
Iz navedenog primjera za mali prst mehatroničke šake vidi se prvo deklariranje potrebnih varijabli pa pohrana očitanih vrijednosti. Slika 2.17. prikazuje blok dijagram za navedeni primjer, do bloka pohrane podataka u varijable.

Nakon što su primljeni podaci spremljeni u varijable oni se šalju na servo motore kako bi motori na osnovu njih promijenili položaj osovine.

```
ServoPinkyOne.write(PinkyOne - PinkyOneFix);
```

```
ServoPinkyTwo.write(PinkyTwo + PinkyTwoFix);

ServoPinkyTwoo.write(PinkyTwo - PinkyTwooFix);
```

Nastavak algoritma za mali prst mehatroničke šake prikazuje obradu podataka te kontrolu servo motora. Također slika 2.17. prikazuje blok dijagram za navedeni primjer, od bloka obrade podataka do kontrole motora.



Slika 2.17 Blok dijagram algoritma upravljanja

Kada se servo motori montiraju u zglobove prstiju mehatroničke šake nije moguće svaki servo motor postaviti u istu poziciju pa se zato koriste varijable čije ime završava na „Fix“ koje služe za fino podešavanje početnog položaja servo motora.

Kompletni algoritam korišten za upravljačku sustav mehatroničke šake nalazi se u prilogu 1.

3 UPRAVLJANJE MEHATRONIČKIM MODELOM ŠAKE

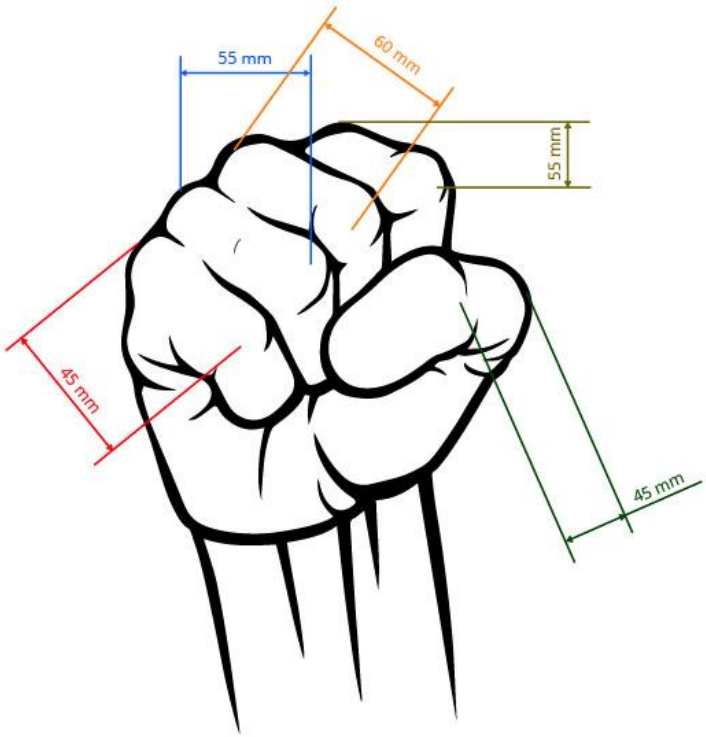
3.1 Upravljačka rukavica

Upravljačka rukavica jedan je od načina upravljanja mehatroničkim modelom šake. Rukavica obavlja funkciju praćenja pokreta prstiju ruke te slanja podataka koji govore o poziciji prstiju mehatroničkoj šaci. Upravljačka rukavica prilagođena je za korištenje na desnoj šaci.

3.1.1 Teorijski model upravljačke rukavice

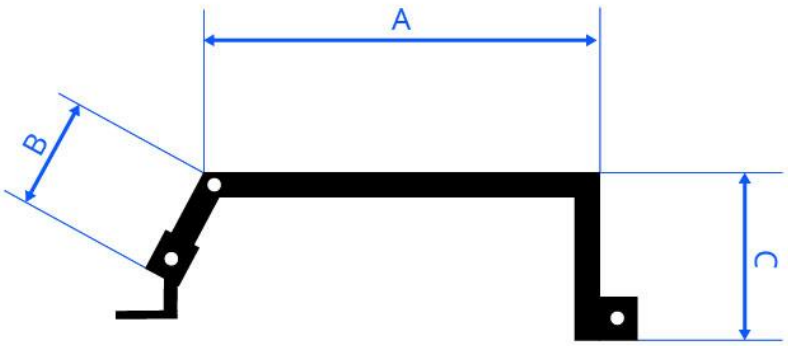
Upravljačka rukavica ima funkciju praćenja pokreta prstiju kada se prsti pomiču prema dlanu, zbog ograničenosti izvedbe upravljačka rukavica nema funkciju praćenja raširenosti prstiju. Pri modeliranju upravljačke rukavice bilo je potrebno uzeti uzorke desne otvorene i zatvorene šake čovjeka te prema tim uzorcima napraviti model.

Uzorkovanje raširene ljudske šake opisano je u poglavlju broj 2, slika 2.1 Uzimanje uzoraka ljuske šake. Uzorkovanje stisnute ljudske šake obavljeno je na desnoj šaci pomičnim mjerilom slika 3.1 Uzorkovanje stisnute ljudske šake.



Slika 3.1 Uzorkovanje stisnute ljudske šake [8]

Prsti upravljačke rukavice su C oblika zbog prilagođavanja prstima pri stisnutoj šaci. Slika 3.2 Prst upravljačke rukavice, prikazuje izgled jednog prsta sa univerzalnim dimenzijama. Bijeli krug prikazuje pomični dio koji se može rotirati do 180 stupnjeva.



Slika 3.2 Prst upravljačke rukavice

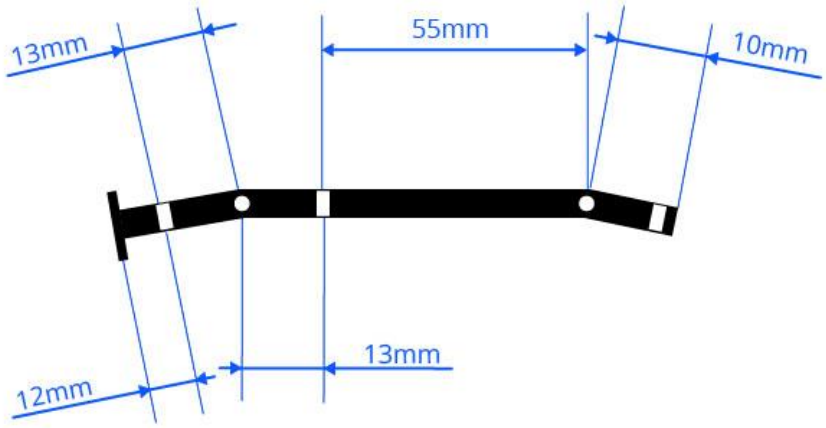
Na osnovu izmjerenih dimenzijama ljudske šake proračunate su dimenzije za prste upravljačke rukavice koje su prikazane u tablica 3.1 Dimenzije prstiju upravljačke rukavice.

Tablica 3.1 Dimenzije prstiju upravljačke rukavice

	A	B	C
Kažiprst	60mm	35mm	40mm
Srednjak	80mm	35mm	50mm
Prstenjak	60mm	35mm	40mm
Mali prst	50mm	35mm	35mm

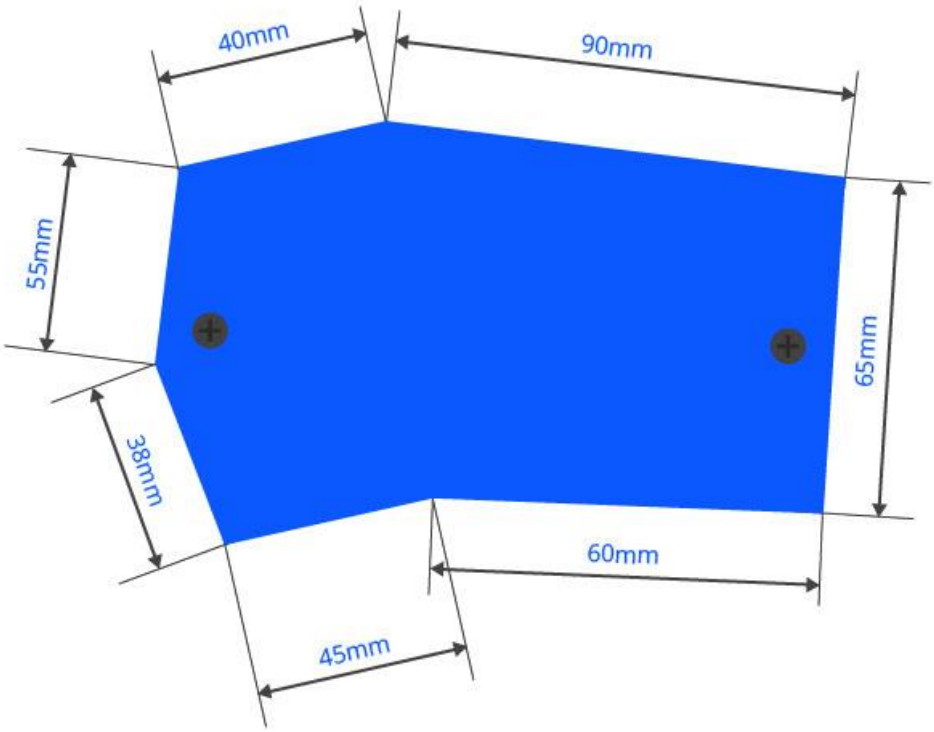
Svi prsti upravljačke rukavice izvedeni su na jednak način osim palca koji je morao biti izveden drugačije zbog funkcije da se pomiče u puno više smjerova nego ostali prsti. Palac

upravljačke rukavice, slika 3.3 Palac upravljačke rukavice, sastoji se od 5 pomičnih zglobova koji omogućuju nesmetano pokretanje palca ruke tokom praćenje pokreta.



Slika 3.3 Palac upravljačke rukavice

Tijelo upravljačke rukavice napravljeno je u dimenzijama (Slika 3.4 tijelo upravljačke rukavice) koje su potrebne da se smjesti baterija, upravljački i komunikacijski sustav uzeći u obzir dimenzije ljudskog dlana.

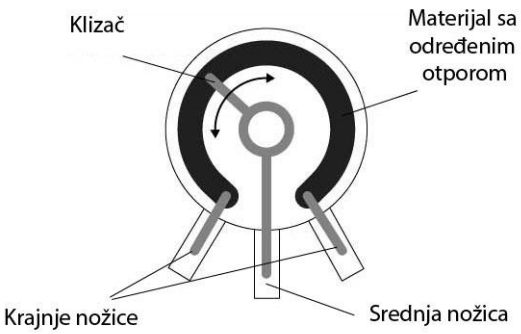


Slika 3.4 Tijelo upravljačke rukavice

3.1.2 Senzori

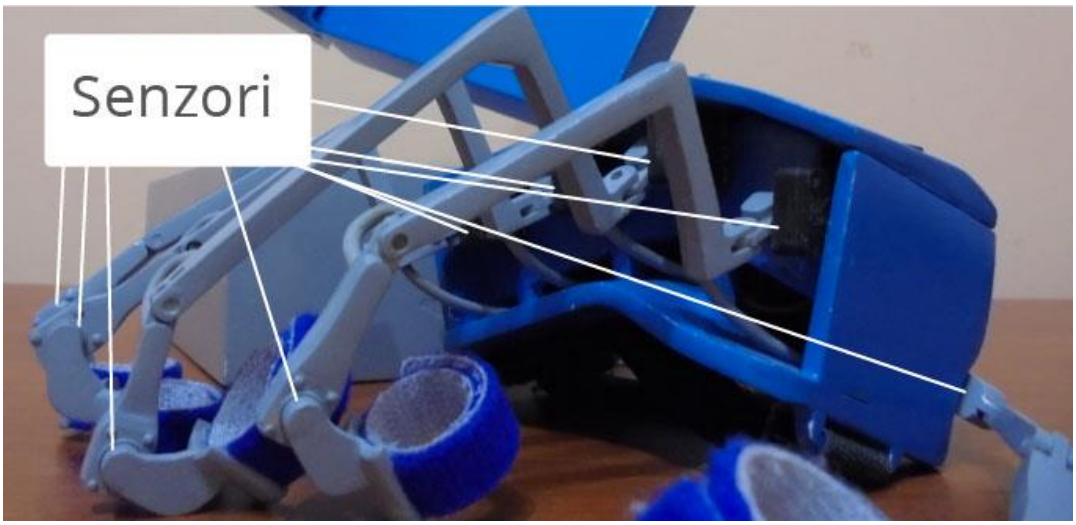
Pri pomicanju prstiju potrebno je koristiti senzore koji mehanički pokret pretvaraju u električnu veličinu. Senzori korišteni za praćenje pokreta su vertikalni potenciometri.

Potenciometri su uređaji koji imaju sposobnost promjene otpora tj. funkcioniraju kao razdjelnik napona. Slika 3.5 prikazuje ilustraciju rada potenciometra. Srednja nožica povezana je sa klizačem koji se kreče po materijalu koji ima određeni otpor i na taj način se mijenja otpor između srednje nožice i krajnjih nožica.



Slika 3.5 Potenciometar [10]

Potenciometri su montirani na krajeve i početke prstiju (Slika 3.6 položaji senzora) upravljačke rukavice te pri pomicanju prstiju mijenjaju otpor i preko vodiča šalju signal položaja prstiju upravljačkom sustavu. Kod svih prstiju se senzore nalaze na istim mjestima osim kod palca gdje se zbog mogućnosti izvedbe koristi jedan senzor na početku prsta, uz tijelo rukavice.



Slika 3.6 Položaji senzora

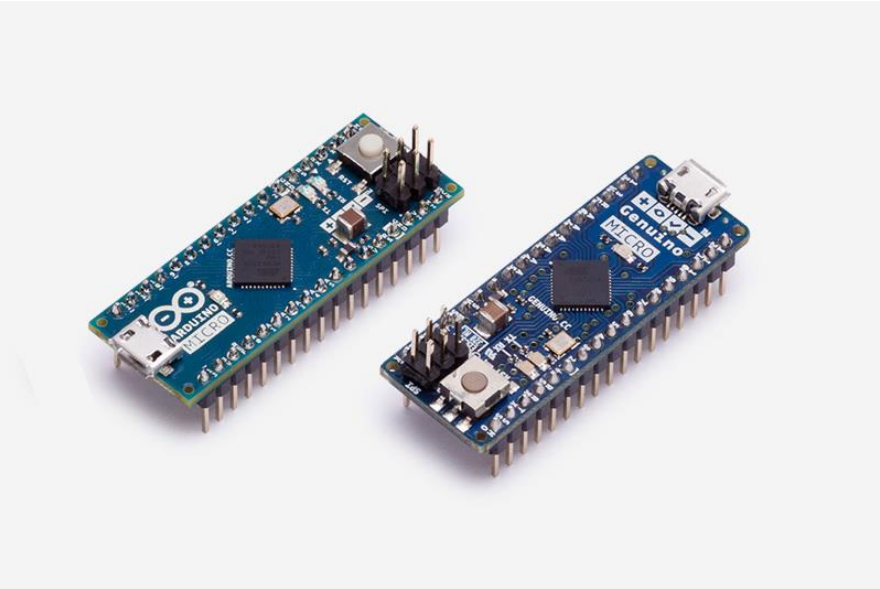
3.1.3 Fizički model upravljačke rukavice

Prema dimenzijama iz teorijskog modela konstruiran je fizički model upravljačke rukavice Slika 3.7 Upravljačka rukavica. Materijal za izradu upravljačke rukavice je pleksiglas (opisan u 2.3.1 Pleksiglas), koji je korišten i u izradi fizičkog modela mehatroničke šake.

Slika 3.7 Upravljačka rukavica

3.1.4 Upravljački i komunikacijski sustav

Upravljačka rukavica za prikupljanje i procesiranje podataka sa senzora te komunikaciju treba upravljačku jedinicu. Za upravljačku jedinicu koristi se arduino micro mikrokontroler Slika 3.8 Arduino micro.



Slika 3.8 Arduino micro [11]

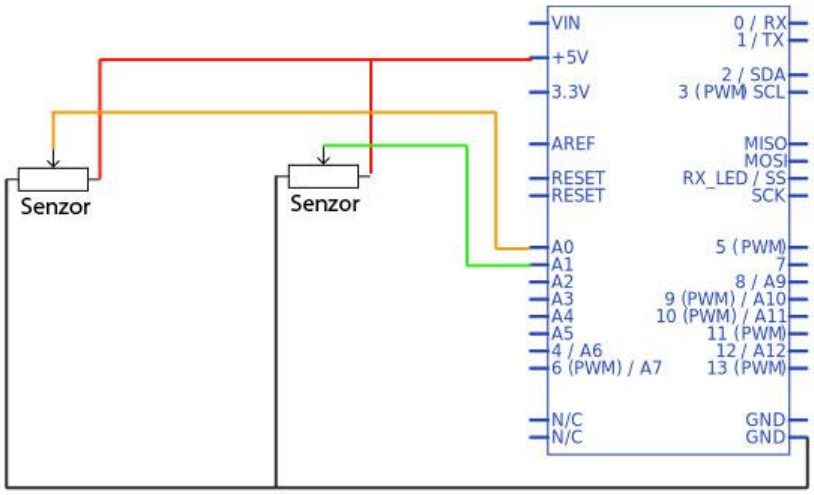
Arduino micro je korišten zbog potrebe spajanja 9 potenciometara na analogne ulaze , svoje male potrošnje koja je dovoljno mala da se cijeli sklop stabilno napaja sa jednom baterijom od 9 volti te vrlo malih dimenzija.

Arduino micro specifikacije su:

- Mikrokontroler: Atmega32U4
- Radni napon: 5V
- Ulazni napon (preporučeni): 7-12V
- Ulazni napon (granice): 6-20V
- Digitalni I/O pinovi: 20
- Pinovi za analogne ulaze: 12
- Istosmjerni iznos struje po I/O pinu: 20mA
- Istosmjerni iznos struje za 3.3V po pinu: 50 mA
- Flash memorija: 32KB
- SRAM: 2.5 KB
- EEPROM: 1KB
- Brzina procesora: 16 MHz
- Širina: 18mm
- Dužina: 48mm
- Težina: 13g

Komunikacija prema mehatroničkom modelu šake obavlja se isključivo bežično preko bluetooth veze. Za uspostavljanje bežične veze koristi se arduino bluetooth modul HC05. Specifikacije i način podešavanja opisani su u poglavlju 2.3.3 i 2.3.4.

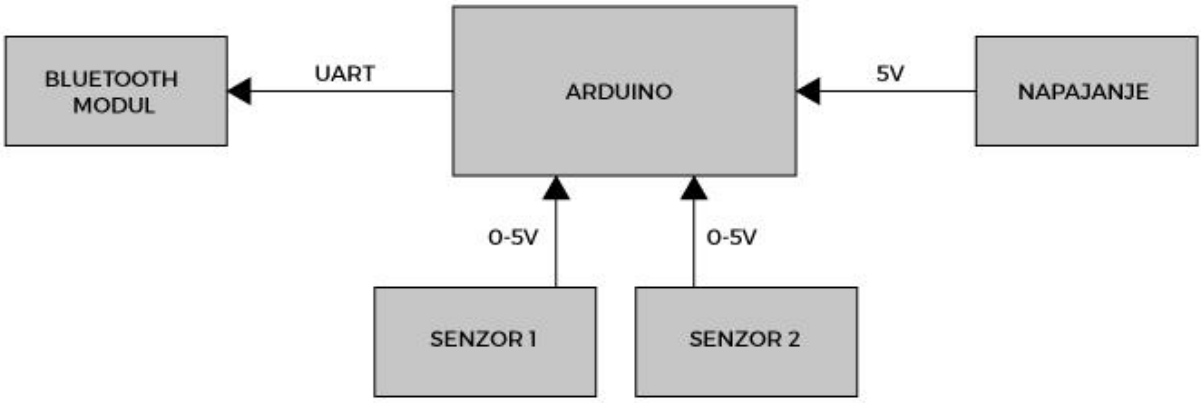
Spajanje senzora i arduina prikazano je sa shemom, slika 3.9 na primjeru jednog prsta. Senzori za ostale prste spajaju se identično samo na druge analogne ulaze a kompletna shema spajanja nalazi se u prilogu 2.



Slika 3.9 Primjer spajanja senzora jednog prsta upravljačke rukavice

3.1.5 Blok shema sustava

Arduino računalni sustav za napajanje koristi bateriju od 9v. Komunikacijski sustav povezan je sa arduino računalnim sustavom na način koji omogućuje napajanje komunikacijskog sustava i podatkovnu vezu sa računalnim sustavom. Arduino računalni sustav preko analognih ulaza prima podatke sa senzora koje dalje obrađuje i šalje na komunikacijski sustav tj. bluetooth modul. Slika 3.10 prikazuje blok shemu sustava sa senzorima jednog prsta upravljačke rukavice. Kompletna shema spoja nalazi se u prilogu br. 2.



Slika 3.10 Blok shema upravljačke rukavice

3.1.6 Algoritam upravljanja

Algoritam upravljanja napisan je u c/c++ programskom jeziku. Upravljački sustav spojen je sa senzorima i komunikacijskim sustavom. Za njihovu kontrolu potreban je algoritam upravljanja koji uzima podatke sa senzora, pretvara ih u odgovarajući oblik te šalje pomoću (serijske) bluetooth veze.

Algoritam upravljanja, po principu rada, jednak je za sve prste upravljačke rukavice. Algoritam započinje sa deklariranjem varijabli koje su potrebne za prvi i drugi senzor jednog prsta rukavice:

```
int ResPinkyOne = A0;
int ResPinkyTwo = A1;
int OpenPinkyOne = 0;
int OpenPinkyTwo = 0;
int ClosePinkyOne;
int ClosePinkyTwo;
int PinkyOne = 0;
int PinkyTwo = 0;
```

Nakon deklariranja varijabli započinje glavna petlja u kojoj se očitavaju vrijednosti sa senzora. Vrijednosti sa senzora očitavaju se preko funkcije analogRead(). Arduino sadrži 8 kanalni 10-bit analog to digital pretvarač što znači da će mapirati iznos pada napona između 0 i 5v u vrijednosti između 0 i 1023. Za jedno očitavanje analogne vrijednosti potrebno je 100

milisekundi. Na slici 3.11. prvi i drugi blok prikazu očitavanje podataka i spremanje očitanih podataka u varijable što predstavlja dolje navedeni algoritam.

```
PinkyOne = analogRead(ResPinkyOne);
PinkyTwo = analogRead(ResPinkyTwo);
```

Kada su vrijednosti sa senzora učitane one se uspoređuju sa prijašnjim vrijednostima. Korisnik upravljačke rukavice je odmah nakon paljenja rukavice obavezan nekoliko puta stisnuti i ispraviti šaku kako bi sljedeći dio algoritma mogao obaviti svoju funkciju. Funkcija sljedećeg djela algoritma je je dodjeljivanje vrijednosti varijablama koje služe za položaj potpuno ispravljenog prsta i potpuno stisnutog prsta. Na ovaj način rukavica nije ograničena na jednu veličinu ruke nego se može prilagoditi na različite veličine ruke i jednako će dobro raditi.

```
if(PinkyOne > OpenPinkyOne)
    OpenPinkyOne = PinkyOne;
 if(PinkyTwo > OpenPinkyTwo)
    OpenPinkyTwo = PinkyTwo;
 if(PinkyOne < ClosePinkyOne)
    ClosePinkyOne = PinkyOne;
 if(PinkyTwo < ClosePinkyTwo)
    ClosePinkyTwo = PinkyTwo;
```

Nakon određivanja početnog i krajnjeg položaja prstiju mapira se varijabla koja će se slati na mehatronički model šake. Navedeni algoritam predstavlja blok „obrada podataka“ na slici 3.11.

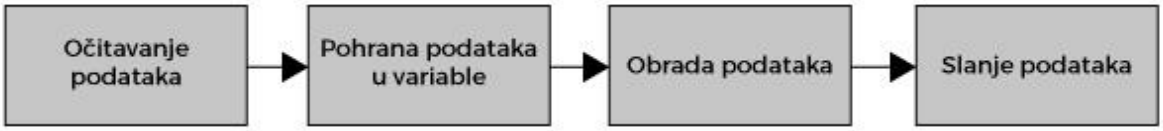
Mapiranje se obavlja sa map() funkcijom. Sintaksa map funkcije (Tablica 3.2. Parametri map() funkije) je map (value, fromLow, fromHigh, toLow, toHigh).

Tablica 3.2. Parametri map() funkcije

value	Broj koji se mapira
fromLow	Gornja granica raspona broja koji se mapira
fromHigh	Doljnja granica raspona broja koji se mapira
toLow	Doljnja željena granica
toHigh	Gornja željena granica
return	Mapirana vrijdnost koja je u željenim granicama i ovisna o broju koji se mapirao

Mapirani i prilagođeni podatci se dalje šalju serijskom vezom preko bluetooth modula na mehatronički model šake pomoću write() funkcije. Write() funkcija šalje mapirane podatke serijskom vezom na bluetooth modul koji ih dalje prenosi na mehatronički model šake.

```
BTserial.write(PinkyOne);
BTserial.write(PinkyTwo);
```



Slika 3.11 Blok dijagram upravljačke rukavice

Algoritam za ostale prste upravljačke rukavice je isti samo se izmjenjuju imena varijabli za svaki prst. Iznimka kod logaritma je za prst palac u koji ima jednu varijablu iz razloga što palac koristi jedan senzor. Cijeli algoritam nalazi se u prilogu 4.

3.2 Aplikacijsko upravljanje modelom šake

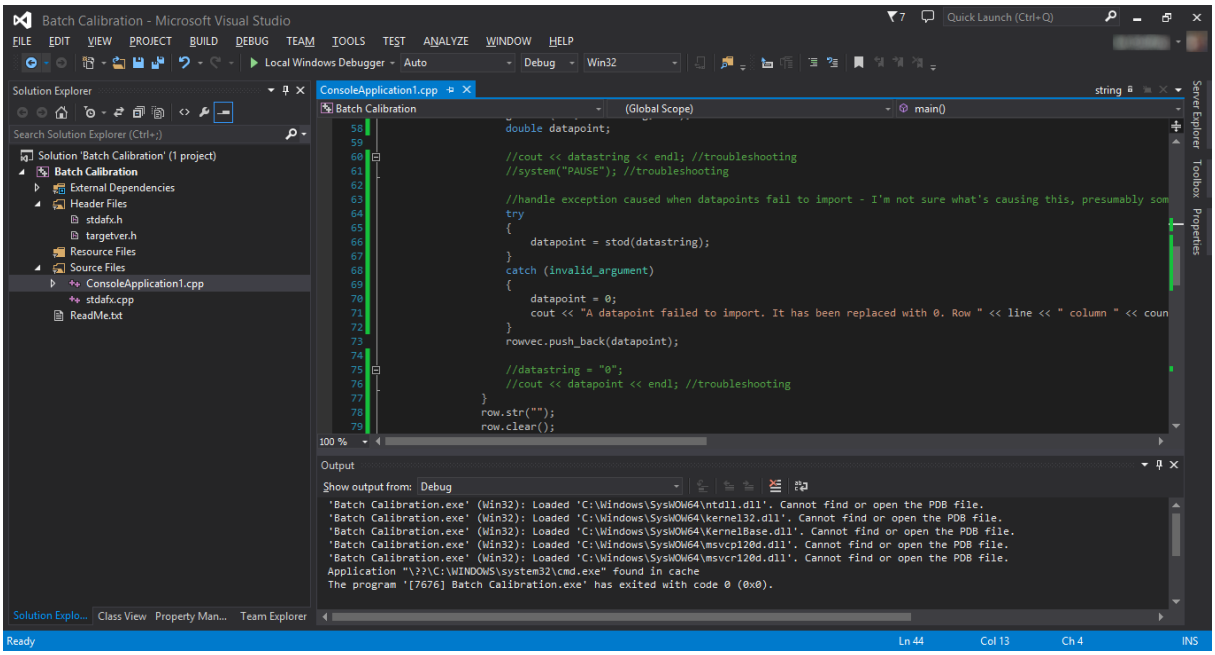
3.2.1 Model upravljačke aplikacije

Modularni pristup izradi mehatroničkog modela šake omogućava kontrolu šake sa svim uređajima koji mogu uspostaviti bluetooth vezu. Osobno računalo je uređaj koji sa bluetooth hardverom može uspostaviti vezu sa mehatroničkom šakom a samim time i njom upravljati.

Upravljačka aplikacija je programsko rješenje za upravljanje mehatroničkim modelom šake preko osobnog računala na način da se preko digitalnog klizača pojedinačno upravlja položaj prstiju šake. Upravljačka aplikacije uspostavlja bluetooth vezu sa mehatroničkom šakom te očitava podatke sa klizača i šalje ih u odgovarajućem formatu te na taj način mijenja položaj prstiju na mehatroničkoj šaci.

3.2.2 Alati i programska podrška

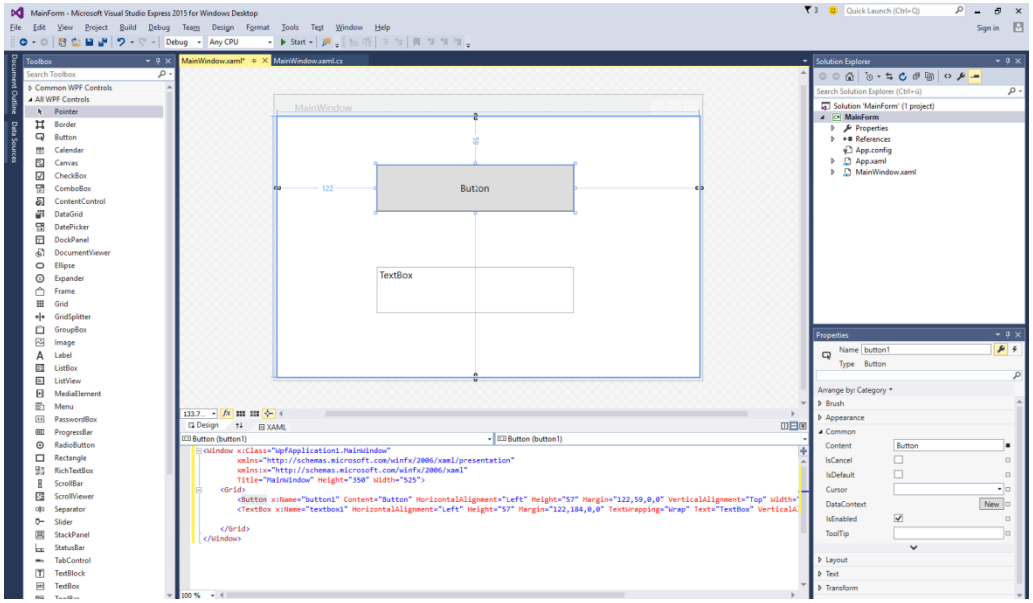
Upravljačka aplikacija napisana je pomoću alata Visual Studio, slika 3.12. Microsoft Visual Studio [12]. Visual Studio je razvojno okruženje (eng.Integrated Development Environment odnosno IDE) namijenjeno razvoju raznih vrsta aplikacija u raznim programskim jezicima na Windows platformi. Najčešće korišteni jezici su Microsoft Visual



Basic, C#, C++.

Slika 3.12. Microsoft Visual Studio

Visual studio podržava i dizajnerski (grafički) pristup izradi programskog algoritma, slika 3.13. U dizajnerskom načinu rada ne treba pisati algoritam nego se programiranje može obaviti raspoređivanjem grafičkih ikona koje simboliziraju dijelove koda pri čemu se upisuju samo osnovne značajke tih dijelova algoritma.



Slika 3.13. Dizajnerski pristup izradi programskog koda

Visual studio podržava windows 10, 8.1, 8, 7 SP1, Server 2012 R2, Server 2012, Server 2008 R2 SP1 operacijske sustave.

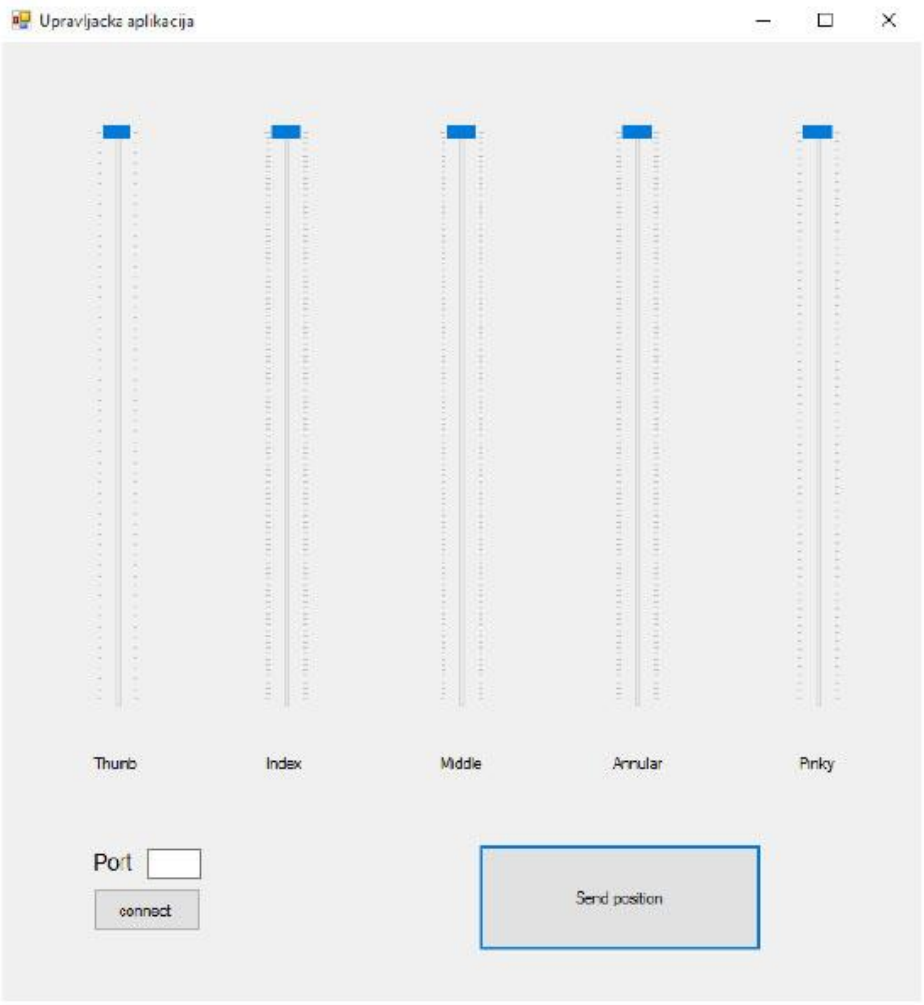
Hardwerski zahtjevi za korištenje Visual Studia su:

- 1.6 GHz ili brži procesor
- 1 GB RAM ili više
- 4 GB prostora na tvrdom disku
- 5400 RPM tvrdi disk
- DirectX 9 kompatibilnu grafičku karticu

3.2.3 Programsko rješenje

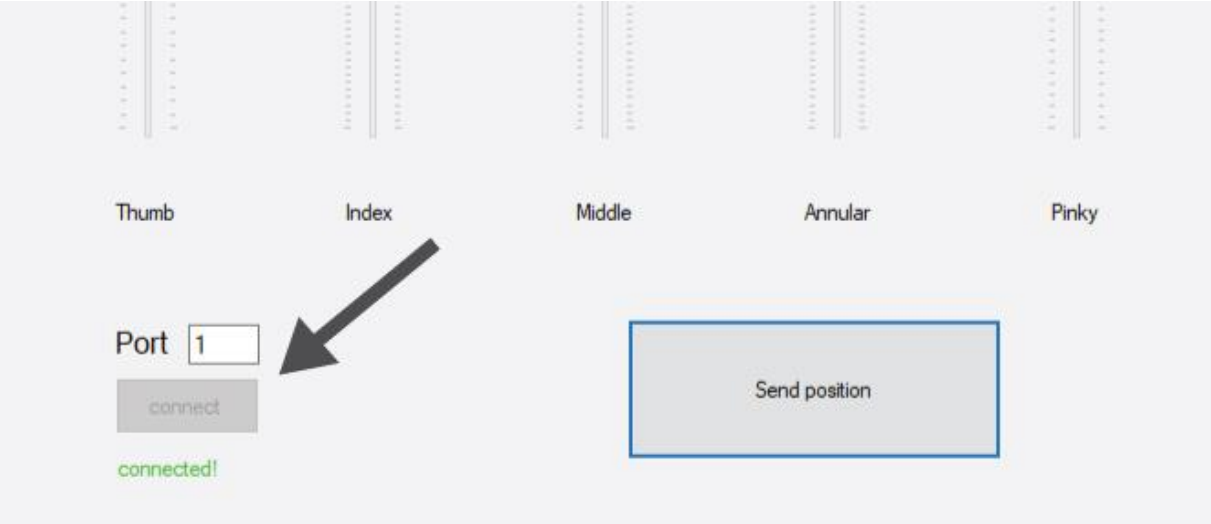
U izradi upravljačke aplikacije korišten je c# programski jezik. C# (CSharp) [13] je nastao u tvrtki Microsoft s ciljem da bude jednostavan, siguran, moderan, objektno orijentiran jezik visokih performansi za .NET platformu. C# je nastao na temelju objektno orijentiranih jezika Java, C++ i Visual Basic. Vrlo je sličan Javi i C++ jeziku (sintaksa i semantika je velikim dijelom preuzeta iz Jave, koja je kao i C# objektno orijentirani jezik). Ali C# za razliku od Jave nije neovisan o platformi, tj. operativnom sustavu, već je kreiran za izradu stolnih (desktop) i Internet aplikacija u .Microsoft .NET okruženju.

Upravljačka aplikacija sastoji se od dva gumba, pet klizača te jednog polja za unos podataka, slika 3.14. Upravljačka aplikacija.



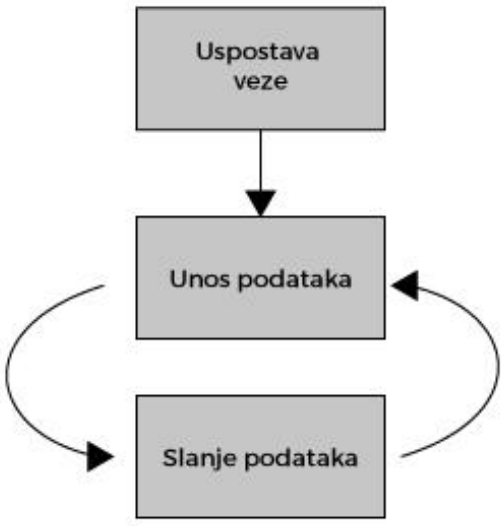
Slika 3.14. Upravljačka aplikacija

Upravljačka aplikacija prvo uzima upisani broj porta bluetooth uređaja mehatroničke ruke te nakon klika na gumb „connect“ stvara bluetooth vezu. Bluetooth veza je uspostavljena kada ispod gumba zelenim slovima piše „Connected!“, slika 3.15.



Slika 3.15. Uspostavljena veza sa mehatroničkom

Nakon uspostavljanja bluetooth veze na klik „Send position“ gumba vrijednosti sa klizača se šalju na mehatroničku šaku i ovisno o njima prsti mehatroničke šake se pomiču u poslani položaj. Blok dijagram na slici 3.16. prikazuje proces povezivanja aplikacije sa mehatroničkom šakom te očitavanje i slanje podataka. Algoritam cijele aplikacije nalazi se u prilogu 5.



Slika 3.16. Blok dijagram upravljačke aplikacije

4 TESTIRANJE I REZULTATI

4.1 Test kašnjenja pokreta mehatroničke šake kontrolirane upravljačkom rukavicom

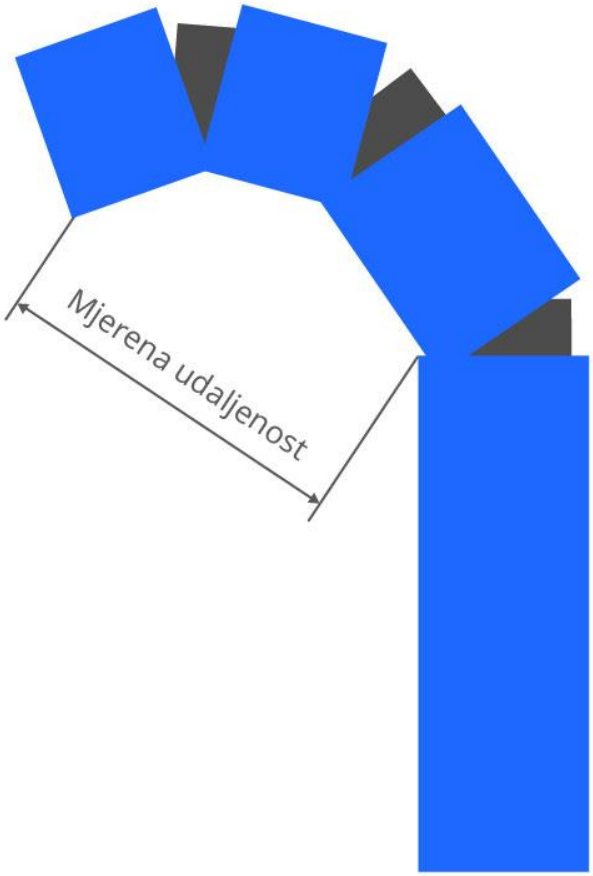
Test kašnjenja mehatroničke šake upravljane upravljačkom rukavicom prikazuje koliko vremena mehatronička šaka "kasni" sa pokretima za ljudskom rukom. Za test je potrebno potpuno saviti kažiprst prema dlanu iz položaja kada je potpuno ispravljen. Test će se obavljati samo na kažiprstu iz razloga što je vrijeme kašnjenja jednako za sve prste. Brojanje vremena započinje u isto vrijeme ali zasebno za mehatronički šaku i ljudsku šaku (sa upravljačkom rukavicom). Brojanje vremena započinje kada se prst ljudske šake počinje pomicati te završava kada prst mehatroničke šake završi kretanje. Vrijeme kašnjenja pokreta mehatroničke šake je razlika vremena pokreta prsta mehatroničke šake i prsta ljudske šake. Testiranje se obavlja pet puta te se kao rezultat uzima prosječna vrijednost.

Tablica 4.1. Rezultati testiranja kašnjenja pokreta mehatroničke šake kontrolirane upravljačkom rukavicom

	Ljudska šaka [s]	Mehatronička šaka [s]
	0.30	0.51
	0.29	0.53
	0.31	0.56
	0.33	0.51
	0.30	0.48
Prosjek	0.306	0.518
Kašnjenje [s]	0.212	

4.2 Test točnosti pozicioniranja mehatroničke šake upravljane upravljačkom rukavicom

Test točnosti pozicioniranja mehatroničke šake upravljanje upravljačkom rukavicom prikazuje odstupanja pozicije prsta mehatroničke šake pri istim pokretima prsta šake sa upravljačkom rukavicom. Za obavljanje testa potrebno je kažiprst ruke na kojoj je upravljačka rukavica pomicati u isti položaj te mjeriti položaj kažiprsta mehatroničke šake. Test će se obavljati samo na kažiprstu iz razloga što su za sve prste korišteni isti motori i preciznost će biti jednaka. Udaljenost koja se mjeri je na mehatroničkoj šaci od početka dlana do vrha prsta. Testiranje se obavlja pet puta i kao rezultat se računa odstupanje razlikom pojedinih mjerenja.



Slika 4.1. Mjerena udaljenost na mehatroničkoj šaci

Tablica 4.2. Rezultati testiranja točnosti pozicioniranja mehatroničke šake upravljane upravljačkom rukavicom

	Udaljenost na mehatroničkoj šaci [mm]
	80
	77
	75
	78
	75
Maksimalna razlika [mm]	5

Iz rezultata testiranja točnosti, tablica 4.2 Rezultati točnosti pozicioniranja mehatroničke šake upravljane upravljačkom rukavicom, zaključuje se da prilikom istih pokreta ljudske ruke dolazi do odstupanja na mehatroničkom modelu šake u rasponu od 5mm.

5 ZAKLJUČAK

U ovom završnom radu napravljen je uređaj pod nazivom mehatronički model šake koji se sastoji od mehatroničke šake, upravljačke rukavice i upravljačke aplikacije. Mehatronička šaka je humanoidnog oblika i sastoji se od upravljačkog računalnog sustava, servo motora te kućišta. Upravljački računalni sustav upravlja servo motorima na osnovu podataka koje primi od upravljačkih jedinica. Upravljačka rukavica je jedna od upravljačkih jedinica. Upravljačka rukavica se sastoji od upravljačkog računalnog sustava, senzora i kućišta. Upravljačka jedinica prima podatke iz senzora, obrađuje ih u prikladan oblik te preko bluetooth veze šalje na mehatroničku šaku. Druga upravljačka jedinica je windows aplikacija koja preko digitalnih klizača prima željeni položaj prstiju mehatroničke šake te preko bluetooth veze šalje podatke na mehatronički model šake. Mehatronički model šake testiran je na brzinu i preciznost pokreta.

LITERATURA

[1] The evolution of human and ape hand proportions, www.researchgate.net (kolovoz, 2015)

[2] Introduction to Servo Motors, www.sciencebuddies.org (prosinac 2012)

[3] SG90 9 g Micro Servo, http://www.micropik.com/PDF/SG90Servo.pdf (25.7.2016)

[4] S0090 plastic gear 9g servo, www.domanrchobby.com (veljača 2015)

[5] Arduino Mega https://www.arduino.cc/en/Main/arduinoBoardMega (25.7.2016)

[6] HC 05 Bluetooth to serial port module http://www.robotshop.com/media/files/pdf/rb-ite-12-bluetooth_hc05.pdf (25.7.2016)

[7] Programski jezik C https://web.math.pmf.unizg.hr/~singer/Prog_Add/c.pdf (25.7.2016)

[8] Fist, www.shutterstock.com

[9] AnalogRead () https://www.arduino.cc/en/Reference/AnalogRead (25.7.2016)

[10] Rotary Potentiometer Diagram, www.techwomen.co, ožujak 2017

[11] Arduino, www.robotprostore.kz

[12] Microsoft https://hr.wikipedia.org/wiki/Microsoft (25.7.2016)

[13] Objektno orijentirano programiranje - Uvod u c# http://www.etfos.unios.hr/~lukic/oop/Auditorne_vje%C5%BEbe_5.pdf (25.7.2016)

SAŽETAK

U ovom azvršnom radu opisan je proces projektiranja, modeliranja, izrade i testiranja mehatroničkog modela šake. Opisan je način izrade, rada i komunikacije upravljačke rukavice i upravljačke aplikacije.

Detaljno je prikazan proces projektiranja mehatroničke šake humanoidnog oblika, uloga njenog upravljačkog računalnog sustava, način bežične komunikacije te servo motori koji pokreču prste šake.

Prikazan je način komunikacije upravljačke rukavice sa mehatroničkom šakom te uloga senzora i obrade podataka sa senzora u upravljačkom računalnom sustavu.

Opisani su alati za korištenje pisanja algoritma te način rada algoritma windows aplikacije za upravljanje mehatroničke šake.

Testiranje mehatroničkog modela šake provedeno je za vrijeme kašnjenja pokreta te preciznost pri istim pokretima kada je mehatronička šaka upravljana upravljačkom rukavicom.

Ključne riječi:

Arduino, bežična komunikacija, servo motor, robotska ruka.

ABSTRACT

This final paper it describes the process of design, modelingd, development and testing of humanoid mechatronic fist. Methods of constructions and communication ways of controlling modules are described in details.

It is written about designing and modeling of mehatronic fist, role of its computer system, wireless communication and control of servo motors

It is written about ways of communications of control glove and mechatronic fist, so as role of the sensors and data processing in its control unit.

Tools form making windows applications are described in details so as a application aswel.

Tesing of mechatronic glove are made to finde out delay of movements and how accurate tat movement are.

Keywords: Arduino, wireless communication, servo motor, robotic arm.

ŽIVOTOPIS

Karlo Kostanjevac rođen je 30. Ožujka 1993. u Osijeku. U Elektrotehničkoj i prometnoj školi Osijek je završio smjer elektrotehnike a u školi Edunova - škola za informatiku i menadžment u Osijeku je završio tečaj za web dizajnera. Trenutno je student apsolventske godine stručnog studija elektrotehnike, smjer informatika.

Tijekom studija bio je demonstrator iz kolegija digitalna elektronika i osnove elektronike te sudjelovao i uspješno ispunio sve uvjete u fakultetskom natječaju „Pro student“ godine 2016. i 2015. sa maketama „Robotska ruka upravljana ljudskom rukom“ i „Kugla i stol“ te uspješno izradio maketu za fakultet pod nazivom „Robotska ruka upravljana ljudskom rukom v2“.

Od kolovoza 2015. zaposlen je kao student u firmi Bamboo lab na mjestu web dizajnera i frontend developera gdje je uspješno radio na projektima kao što su izrada web stranica za slastičarnu Gold by Waldinger, Elektromodul i Ato.

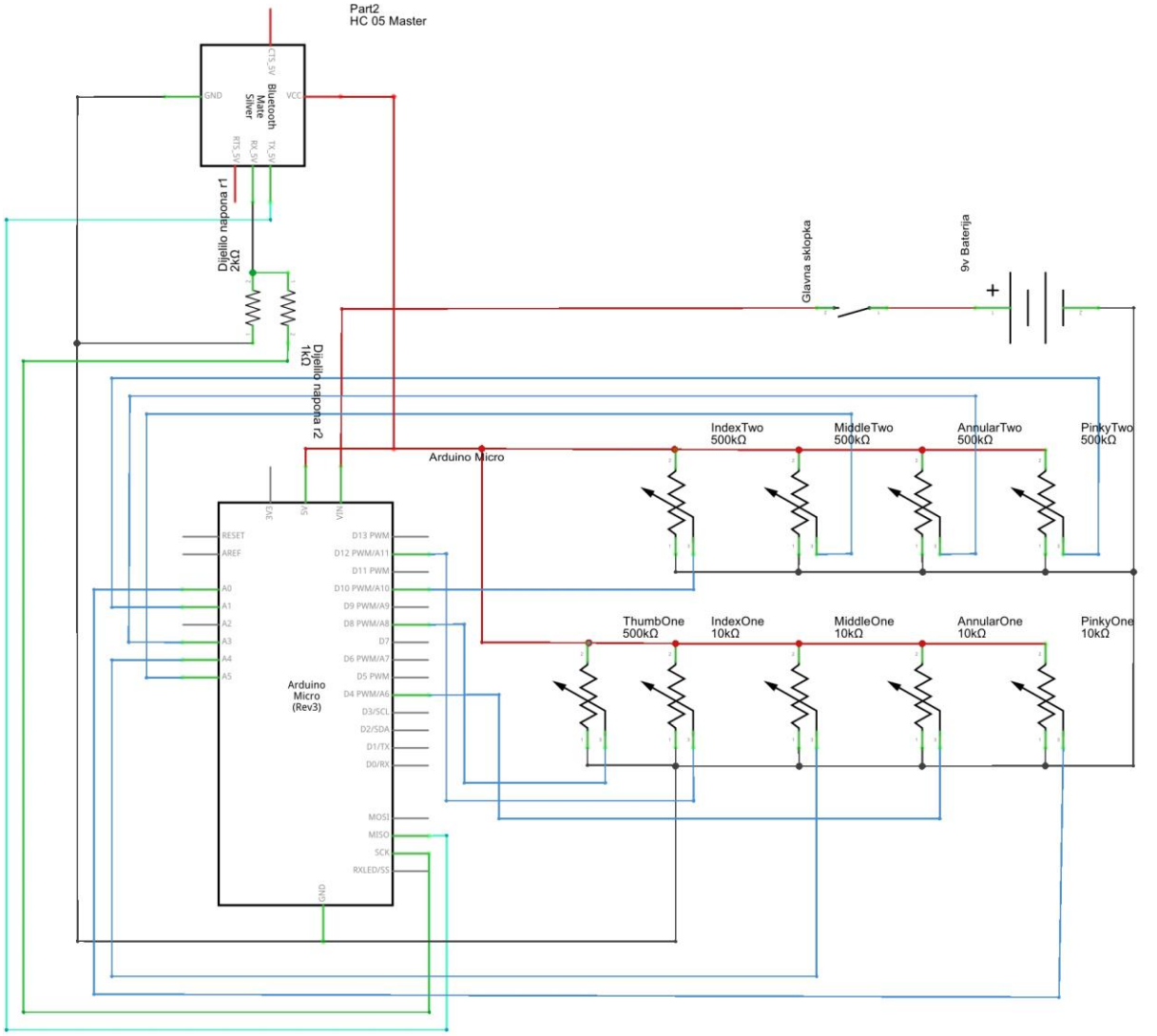
PRILOZI

Prilog 1

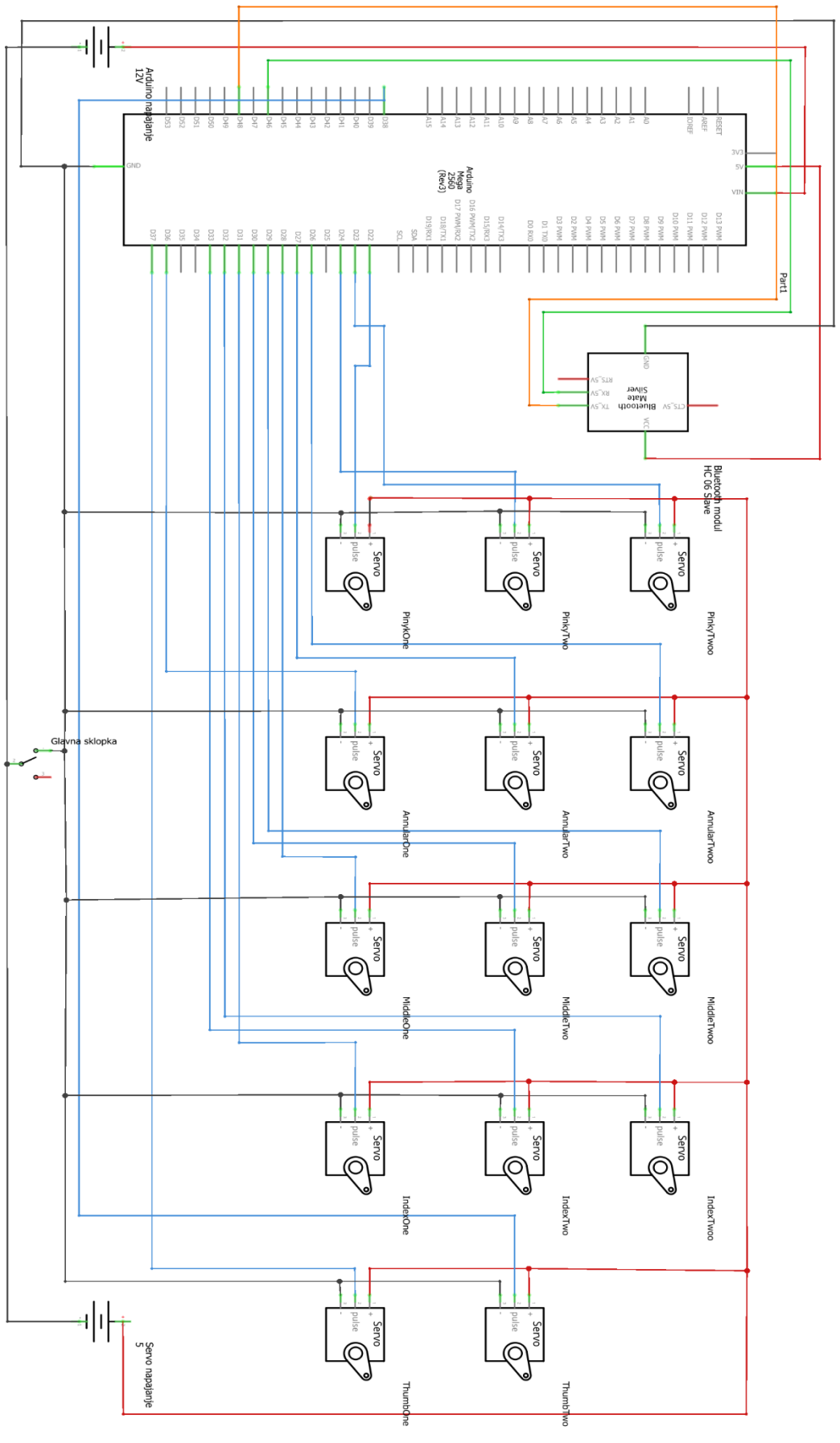
```
//PINKY
int ResPinkyOne = A0;
int ResPinkyTwo = A1;

int OpenPinkyOne = 0;
int OpenPinkyTwo = 0;

int ClosePinkyOne;
int ClosePinkyTwo;

int PinkyOne = 0;
int PinkyTwo = 0;

//ANNULAR
int ResAnnularOne = A6;
int ResAnnularTwo = A3;

int OpenAnnularOne = 0;
int OpenAnnularTwo = 0;

int CloseAnnularOne;
int CloseAnnularTwo;

int AnnularOne = 0;
int AnnularTwo = 0;
//MIDDLE
int ResMiddleOne = A4;
int ResMiddleTwo = A5;

int OpenMiddleOne = 0;
int OpenMiddleTwo = 0;

int CloseMiddleOne;
int CloseMiddleTwo;

int MiddleOne = 0;
int MiddleTwo = 0;

//INDEX
int ResIndexOne = A11;
int ResIndexTwo = A10;

int OpenIndexOne = 0;
int OpenIndexTwo = 0;

int CloseIndexOne;
int CloseIndexTwo;

int IndexOne = 0;
int IndexTwo = 0;

//THUMB
int ResThumbOne = A8;

int OpenThumbOne = 0;

int CloseThumbOne;

int ThumbOne = 0;

#include <SoftwareSerial.h>
SoftwareSerial BTserial(14, 15); // RX | TX

void setup()
{
   Serial.begin(9600);
   BTserial.begin(9600);

   ClosePinkyOne = analogRead(ResPinkyOne);
   ClosePinkyTwo = analogRead(ResPinkyTwo);

   CloseAnnularOne = analogRead(ResAnnularOne);
   CloseAnnularTwo = analogRead(ResAnnularTwo);

   CloseMiddleOne = analogRead(ResMiddleOne);
   CloseMiddleTwo = analogRead(ResMiddleTwo);

   CloseIndexOne = analogRead(ResIndexOne);
   CloseIndexTwo = analogRead(ResIndexTwo);

   CloseThumbOne = analogRead(ResThumbOne);


}

void loop()
{

   PinkyOne = analogRead(ResPinkyOne);
   PinkyTwo = analogRead(ResPinkyTwo);

   if(PinkyOne > OpenPinkyOne)
      OpenPinkyOne = PinkyOne;
   if(PinkyTwo > OpenPinkyTwo)
      OpenPinkyTwo = PinkyTwo;

   if(PinkyOne < ClosePinkyOne)
      ClosePinkyOne = PinkyOne;
   if(PinkyTwo < ClosePinkyTwo)
      ClosePinkyTwo = PinkyTwo;

   PinkyOne = map(PinkyOne ,ClosePinkyOne ,OpenPinkyOne
,80,180);
   PinkyTwo = map(PinkyTwo ,ClosePinkyTwo ,OpenPinkyTwo,
90,180);

   AnnularOne = analogRead(ResAnnularOne);
   AnnularTwo = analogRead(ResAnnularTwo);

   if(AnnularOne > OpenAnnularOne)
      OpenAnnularOne = AnnularOne;
   if(AnnularTwo > OpenAnnularTwo)
      OpenAnnularTwo = AnnularTwo;

   if(AnnularOne < CloseAnnularOne)
      CloseAnnularOne = AnnularOne;
   if(AnnularTwo < CloseAnnularTwo)
      CloseAnnularTwo = AnnularTwo;

   AnnularOne = map(AnnularOne ,CloseAnnularOne
,OpenAnnularOne ,90,170);
   AnnularTwo = map(AnnularTwo ,CloseAnnularTwo
,OpenAnnularTwo, 70,170);

   MiddleOne = analogRead(ResMiddleOne);
   MiddleTwo = analogRead(ResMiddleTwo);
   if(MiddleOne > OpenMiddleOne){
    if (MiddleOne < 460){
      OpenMiddleOne = MiddleOne;
    }
   }
    Serial.println(OpenMiddleOne);
   if(MiddleTwo > OpenMiddleTwo)
      OpenMiddleTwo = MiddleTwo;

   if(MiddleOne < CloseMiddleOne)
      CloseMiddleOne = MiddleOne;
   if(MiddleTwo < CloseMiddleTwo)
      CloseMiddleTwo = MiddleTwo;

   MiddleOne = map(MiddleOne ,CloseMiddleOne ,OpenMiddleOne
,0,190);
   MiddleTwo = map(MiddleTwo ,CloseMiddleTwo ,OpenMiddleTwo,
70,170);

   IndexOne = analogRead(ResIndexOne);
   IndexTwo = analogRead(ResIndexTwo);

   if(IndexOne > OpenIndexOne)
      OpenIndexOne = IndexOne;
   if(IndexTwo > OpenIndexTwo)
      OpenIndexTwo = IndexTwo;

   if(IndexOne < CloseIndexOne)
      CloseIndexOne = IndexOne;
   if(IndexTwo < CloseIndexTwo)
      CloseIndexTwo = IndexTwo;

   IndexOne = map(IndexOne ,CloseIndexOne ,OpenIndexOne
,80,170);
   IndexTwo = map(IndexTwo ,CloseIndexTwo ,OpenIndexTwo,
80,170);

   ThumbOne = analogRead(ResThumbOne);

   if(ThumbOne > OpenThumbOne)
      OpenThumbOne = ThumbOne;

   if(ThumbOne < CloseThumbOne)
      CloseThumbOne = ThumbOne;

   ThumbOne = map(ThumbOne ,CloseThumbOne ,OpenThumbOne
,90,170);

   BTserial.write("+");
   BTserial.write(PinkyOne);
   BTserial.write(PinkyTwo);
   BTserial.write(AnnularOne);
   BTserial.write(AnnularTwo);
   BTserial.write(MiddleOne);
   BTserial.write(MiddleTwo);
   BTserial.write(IndexOne);
   BTserial.write(IndexTwo);
   BTserial.write(ThumbOne);

   delay(25);
```

}

Prilog 2


Part2
HC 05 Master
Bluetooth Mate Silver
GND
VCC
Djelilo napona r1 2kΩ
Djelilo napona r2 1kΩ
Glavna sklopka
9v Baterija
Arduino Micro
IndexTwo 500kΩ
MiddleTwo 500kΩ
AnnularTwo 500kΩ
PinkyTwo 500kΩ
ThumbOne 500kΩ
IndexOne 10kΩ
MiddleOne 10kΩ
AnnularOne 10kΩ
PinkyOne 10kΩ
Arduino Micro (Rev3)
fritzing

Prilog 3


Arduino napajanje 12V
Arduino Mega 2560 (Rev3)
Part1
Bluetooth Mate Silver
Bluetooth modul HC 06 Slave
Glavna sklopka
Servo napajanje 5
PinkyOne
PinkyTwo
AnnularOne
AnnularTwo
MiddleOne
MiddleTwo
IndexOne
IndexTwo
ThumbOne
ThumbTwo
Servo
fritzing

Prilog 4

```
#include <SoftwareServo.h>
#include <AltSoftSerial.h>
AltSoftSerial BTserial; // RX | TX
SoftwareServo ServoPinkyOne, ServoPinkyTwo, ServoPinkyTwoo,
    ServoAnnularOne, ServoAnnularTwo, ServoAnnularTwoo,
    ServoMiddleOne, ServoMiddleTwo, ServoMiddleTwoo,
    ServoIndexOne, ServoIndexTwo, ServoIndexTwoo,
    ServoThumbOne, ServoThumbTwo;
byte StartPackage;
//PINKY
int PinkyOne = 0;
int PinkyTwo = 0;
int SavePinkyOne = 0;
int SavePinkyTwo = 0;
int PinkyOneFix = 50;
int PinkyTwoFix = 20;
int PinkyTwooFix = 60;

//ANNULAR
int AnnularOne = 0;
int AnnularTwo = 0;
int SaveAnnularOne = 0;
int SaveAnnularTwo = 0;
int AnnularOneFix = 10;
int AnnularTwoFix = 20;
int AnnularTwooFix = 30;

//MIDDLE
int MiddleOne = 0;
int MiddleTwo = 0;
int SaveMiddleOne = 0;
int SaveMiddleTwo = 0;
int MiddleOneFix = 20;
int MiddleTwoFix = 5;
int MiddleTwooFix = 10;

//INDEX
int IndexOne = 0;
int IndexTwo = 0;
int SaveIndexOne = 0;
int SaveIndexTwo = 0;
int IndexOneFix = 10;
int IndexTwoFix = 12;
int IndexTwooFix = 15;

//THUMB
int ThumbOne = 0;
int SaveThumbOne = 0;
int ThumbOneFix = 0;
int ThumbOneeFix = 15;

void setup()
{
  Serial.begin(9600);
  BTserial.begin(9600);

  ServoPinkyOne.attach(22);
  SoftwareServo::refresh();
  delay(300);
  ServoPinkyTwoo.attach(23);
  SoftwareServo::refresh();
  delay(300);
  ServoPinkyTwo.attach(24);
  SoftwareServo::refresh();
  delay(300);
  ServoAnnularOne.attach(36);
  SoftwareServo::refresh();
  delay(300);
  ServoAnnularTwoo.attach(26);
  SoftwareServo::refresh();
  delay(300);
  ServoAnnularTwo.attach(27);
  SoftwareServo::refresh();
  delay(300);
  ServoMiddleOne.attach(28);
  SoftwareServo::refresh();
  delay(300);
  ServoMiddleTwoo.attach(29);
  SoftwareServo::refresh();
  delay(300);
  ServoMiddleTwo.attach(30);
  SoftwareServo::refresh();
  delay(300);
  ServoIndexOne.attach(31);
  SoftwareServo::refresh();
  delay(300);
  ServoIndexTwoo.attach(32);
  SoftwareServo::refresh();
  delay(300);
  ServoIndexTwo.attach(33);
  SoftwareServo::refresh();
  delay(300);
  ServoThumbOne.attach(37);
  SoftwareServo::refresh();
  delay(300);
  ServoThumbTwo.attach(38);
  SoftwareServo::refresh();
  delay(30);
}
void loop()
{
  if (BTserial.available())
  {
   StartPackage = BTserial.read();
   PinkyOne = BTserial.read();
   PinkyTwo = BTserial.read();

   AnnularOne = BTserial.read();
   AnnularTwo = BTserial.read();

   MiddleOne = BTserial.read();
   MiddleTwo = BTserial.read();

   IndexOne = BTserial.read();
   IndexTwo = BTserial.read();

   ThumbOne = BTserial.read();

  }

  if(StartPackage == '+'){
   if(PinkyOne < 90 ){}
   else if (PinkyOne == SavePinkyOne){}
   else
   {
     ServoPinkyOne.write(PinkyOne - PinkyOneFix);
     Serial.println(PinkyOne + PinkyOneFix);
   }
   SavePinkyOne= PinkyOne;

   if(PinkyTwo < 90 ){}
   else if (PinkyTwo == SavePinkyTwo) {}
   else
   {
     ServoPinkyTwo.write(PinkyTwo + PinkyTwoFix);
     ServoPinkyTwoo.write(PinkyTwo - PinkyTwooFix);
   }
   SavePinkyTwo = PinkyTwo;
   if(AnnularOne < 70 ){}
   else if (AnnularTwo == SaveAnnularTwo ){}
   else
   {
     ServoAnnularOne.write(AnnularOne - AnnularOneFix);
   }
   if(AnnularTwo < 70 ){}
   else if (AnnularTwo == SaveAnnularTwo ){}
   else
   {
     ServoAnnularTwo.write(AnnularTwo - AnnularTwoFix);
     ServoAnnularTwoo.write(AnnularTwo + AnnularTwooFix);
   }
   SaveAnnularTwo = AnnularTwo;
   SoftwareServo::refresh();
   if(MiddleOne < 60 ){}
   else if (MiddleOne == SaveMiddleOne ){}
   else {
    ServoMiddleOne.write(MiddleOne + MiddleOneFix);
    }
   if(MiddleTwo < 70 ){}
   else if (MiddleTwo == SaveMiddleTwo ){}
   else {
     ServoMiddleTwo.write(MiddleTwo + MiddleTwoFix);
     ServoMiddleTwoo.write(MiddleTwo - MiddleTwooFix);
   }
   SoftwareServo::refresh();


   if(IndexOne < 80 ){}
   else if (IndexOne == SaveIndexOne ){}
   else {
    ServoIndexOne.write(IndexOne + IndexOneFix);
   }
   if(IndexTwo < 80 ){}
   else if (IndexTwo == SaveIndexTwo ){}
   else {
     ServoIndexTwo.write(IndexTwo + IndexTwoFix);
     ServoIndexTwoo.write(IndexTwo - IndexTwooFix);
   }
   SoftwareServo::refresh();
   if(ThumbOne < 90 ){}
   else if (ThumbOne == SaveThumbOne ){}
   else {
    ServoThumbOne.write(ThumbOne + ThumbOneFix);
    ServoThumbTwo.write(ThumbOne + ThumbOneeFix);
    }
  }
  delay(25);
}
```

Prilog 5

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
using System.Windows.Forms;
using System.IO.Ports;

namespace mehatronickimodelsake
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.PortName = "COM" + port_name.Text;
            serialPort1.BaudRate = 9600;
            serialPort1.Open();
            if (serialPort1.IsOpen)
            {
                connection_state.Text = "connected!";
                button2.Enabled = false;
            }
        }
        private void button1_Click(object sender, EventArgs e)
        {

            if (serialPort1.IsOpen)
            {
                    serialPort1.Write(     pinky.Value.ToString()     +     "."     +
pinky.Value.ToString()    +    "."    +    annular.Value.ToString()    +    "."    +
annular.Value.ToString()    +    "."    +    middle.Value.ToString()    +    "."    +
middle.Value.ToString() + "." + index.Value.ToString() + "." + index.Value.ToString()
+ "." + thumb.Value.ToString());
            }
        }


    }
}
```