«Քվանտ» վարժարան

Հետազոտական աշխատանք

Թեմա՝ Արհեստական ձեռքի ստեղծում: Կառավարում

Դասարան՝ 11^1

Առարկա՝ Ֆիզիկա

Աշակերտ՝ Մանուչարյան Արգիշտի

Ղեկավար՝ Աղաբաբյան Գ.

Երևան-2024

Բովանդակություն

ԳԼՈՒԽ 1. Աշխատանքի նպատակ, խնդրի սահմանում և առանձնահատկություն

Աշխատանքի նպատակը

Հետազոտական աշխատանքի նպատակն է ուսումնասիրել 3D մոդելավորման և ծրագրավորման հնարավորությունների միջոցով մարդկային ձեռքին նմանվող պրոթեզի(արհեստական կամ բիոնիկ ձեռքի) ստեղծման հիմունքները, ինչպես նաև ձեռք բերել որոշակի հմտություններ 3D տպիչներից օգտվելու, arduino միկրոկոնտրոլլերի հետ աշխատելու գործում: Մինչև այս թեման ընտրելը նախատեսել էի որպես հետազոտականու թեմա ընտրել մանիպուլյատորի ստեղծում, բայց հետո ոգեշնչվելով OpenBioncs կազմակերպության աշխատանքով, տեսնելով նրանց 3D տպիչով ստեղծած պրոթեզները, ինքս էլ որոշեցի ուսումնասիրել այդպիսի պոռթեզների ստեղծման հիմունքները:

Ձեռքի պրոթեզները սարքեր են, որոնք նախատեսված են այն մարդկանց համար, ովքեր չունեն ձեռքի, նախաբազկի կամ ձեռքի մի մասը: Դրանք կարող են օգտագործվել այն մարդկանց կողմից, ովքեր վերջույթ են կորցրել վնասվածքի, դժբախտ պատահարի կամ հիվանդության հետևանքով, ինչպես նաև նրանց, ովքեր վերջույթների տարբերություն ունեն բնածին հիվանդության պատճառով: Ձեռքի պրոթեզները կամ վերին վերջույթների պրոթեզները լինում են ինչպես պարզ և հասարակ, այնպես էլ բարձր տեխնոլոգիական "բիոնիկ" ձեռքեր, որոնք թե տեսքով, թե գործառույթով չեն զիջում մարդկային ձեռքին: Իհարկե չենք կարող ասել որ ձեռքի պրոթեզները ամբողջովին կարող են փոխարինել մարդկային ձեռքին, բայց նրանք կարող են կատարել սովորական առօրյա գործողությունները, որոշ չափով հեշտացնելով դրանք կրողի կյանքը: Լավ պրոթեզները ոչ միայն ֆիզիկապես են օգնում մարդուն այնպես էլ հոգեբանորեն, օգնելով նրան լինել ավելի վստահ:

Ձեռքի Պրոթեզները լինում են 3 տեսակի՝

- Պասիվ կամ կոսմետիկ
- Մարմնի կողմից կառավարվող
- Միոէլեկտրական

Պասիվ կամ կոսմետիկ պրոթեզները նախատեսված են այն մարդկանց համար, ովքեր ցանկանում են ունենալ հնարավորինս բնական տեսքով պրոթեզ, սրանք նախագծվում են ըստ մի քանի գործոնի՝ մարդու գույնի, սեռի, ֆիզիկական հատկությունների: Բայց ի տարբերություն մյուս տեսակի պրոթեզների պասիվ պրոթեզները չեն կատարում մարդկային ձեռքի գործառույթները, նրանք նման են սովորական ձեռքին, բայց չեն գործում որպես ձեռք:

Մարմնի կողմից կառավարվող պրոթեզները ըստ իս ամենաբարդ կառավարվողներն են, քանզի դրանց կառավարման համար անհրաժեշտ է մարդկային ուժ, սրանք կառավարվում են ուսին կապված գոտիների շնորհիվ: Իհարկե սրանք կատարում են որոշակի գործողություններ, բայց միևնույն է այս տեսակը շատ չի նմանվում սովորական ձեռքին՝ իր գործառույթով:

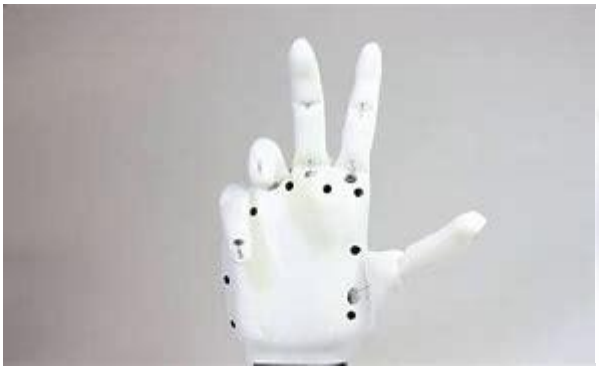
Միոէլեկտրական պրոթեզները մինչ օրս առկա ամենակատարյալ և Ունիվերսալ ձեռքի պրոթեզներն են։ Երբեմն կոչվում են "բիոնիկ" ձեռքեր, միոէլեկտրական պրոթեզները կառավարվում են մարդու սեփական մկանային ազդանշաններով և սնուցվում են մարտկոցով։ Ձեռք վրա գտնվող էլեկտրոդները կարող են գրավել նյարդային ազդակները օգտագործողի մկաններից և օգտագործել դրանք ձեռքի պրոթեզի շարժիչները կառավարելու համար։ Վերջին շրջանում ստեղծվում են նաև պրոթեզներ որոնք կառավարվում են համակարգչային տեսողությամբ։ Հետազոտական աշխատանքի սկզբում ես ունեի նպատակ իմ արհեստական ձեռքը կառավարել մկանների շնորհիվ, ցավոք աշխատանքի վերջին շրջանում սենսորը, որը հաշվարկում էր էլեկտրոդների շնորհիվ ստացված նյարդային ազդակները և փոխանցում միկրոկոնտրոլլերին փչացավ, ես չունեմ հնարավորություն արհեստական ձեռքի կառավարումը մկանների շնորհիվ ցուցադրել, ինչպես նաև նկատի ունենանք այն փաստը, որպեսզի ստանանք հաստատուն և ճշգրիտ արժեքներ նյարդային ազդակների մասին, անհրաժեշտ է օգտագործել այնպիսի սենսոր, որը հնարավորություն կտա աշխատել ամենաքիչը 6 էլեկտրոդի հետ, որպեսզի ստանանք տեղեկություն ամեն մատի համար պատասխանատու մկանների մասին։ Եվ ահա այս ամենի պատճառով ես ստեղծել եմ c# ծրագրավորման լեզվի միջոցով հասարակ տեսքով UI ծրագիր, որի շնորհիվ կարող ենք կառավարել ձեռքը։

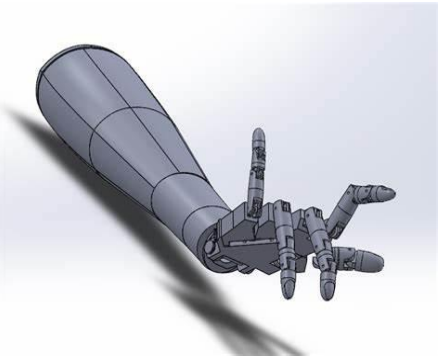
Առանձնահատկություն

Հետաքրքիր է այն փաստը, որ շուկայի գրեթե բոլոր պրոթեզները բավական բարձր գնային արժեք ունեն, իսկ 3D տպիչով տպված պրոթեզը հնարավորություն է տալիս ստեղծել ավելի ցածր գնային արժեք ունեցող պրոթեզ։ Ցանկանում եմ վստահեցնել, որ 3D տպված պրոթեզները ամրությամբ գրեթե չեն զիջում շուկայի կարբոնե պրոթեզներին։ Սա նաև հնարավորություն է տալիս ցանկացած անսարքության դեպքում շատ հեշտությամբ փոխարինել ցանկացած մասնիկ։ Այս հետազոտական աշխատանքի արդյունքում ստեղծված արհեստական ձեռքը կառավարվում է ձայնային հրահանգների շնորհիվ(ներկայիս պահին միայն անգլերեն տարբերակով)։ Ձեռքի յուրաքանչյուր մատի շարժման՝ գործառույթի համար պատասխանատու են սեռվո շարժիչները(յուրաքանչյուր մատի համար մեկ սեռվո շարժիչ)։ Սեռվո շարիչները աշխատում arduino nano միկրոկոնտրոլլերի հետ՝ գլխավորությամբ, որն իր հերթին միացված է համակարգչին usb լարով, համակարգիչը մենք կարող ենք փոխարինել Raspberry Pi մինի համակարգչով, որի չափսերը ավելի պրակտիկ են, և որը մեր արհեստական ձեռքը կդարձնի ավելի լիարժեք և անհրաժեշտություն չի լինի այն անմիջական միացնել համակարգչին։ Սեռվո շարժիչների համար հոսանքի աղբյուր ծառայում են 2 հատ լիթիում է մարդկոցները, յուրաքանչյուրը 3.7v լարումով։

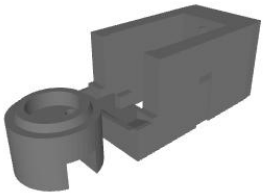
ԳԼՈՒԽ 2. Պրոթեզի 3D Մոդել: Մոդելի տպում

Աշխատանքը սկսելիս ամենից առաջ անհրաժեշտ էր պրոթեզի մոդել, աշխատանքի սկզբում եղան փորձեր սեփական մոդելը ստեղծելու գործում, ցավոք չունենալով բավականին գիտելիքներ 3D մոդելավորում, որոշում կայացվեց գտնել արդեն պատրաստի մոդել և այն ձևափոխել՝ փոխելով ինչպես տեսքը, այնպես էլ աշխատելու մեխանիզմը: Ընտրությունը իրականացվում էր OpenBionics -ի ADA HAND-ի և INMOOV ռոբոտի ձեռքի մոդելների միջև:

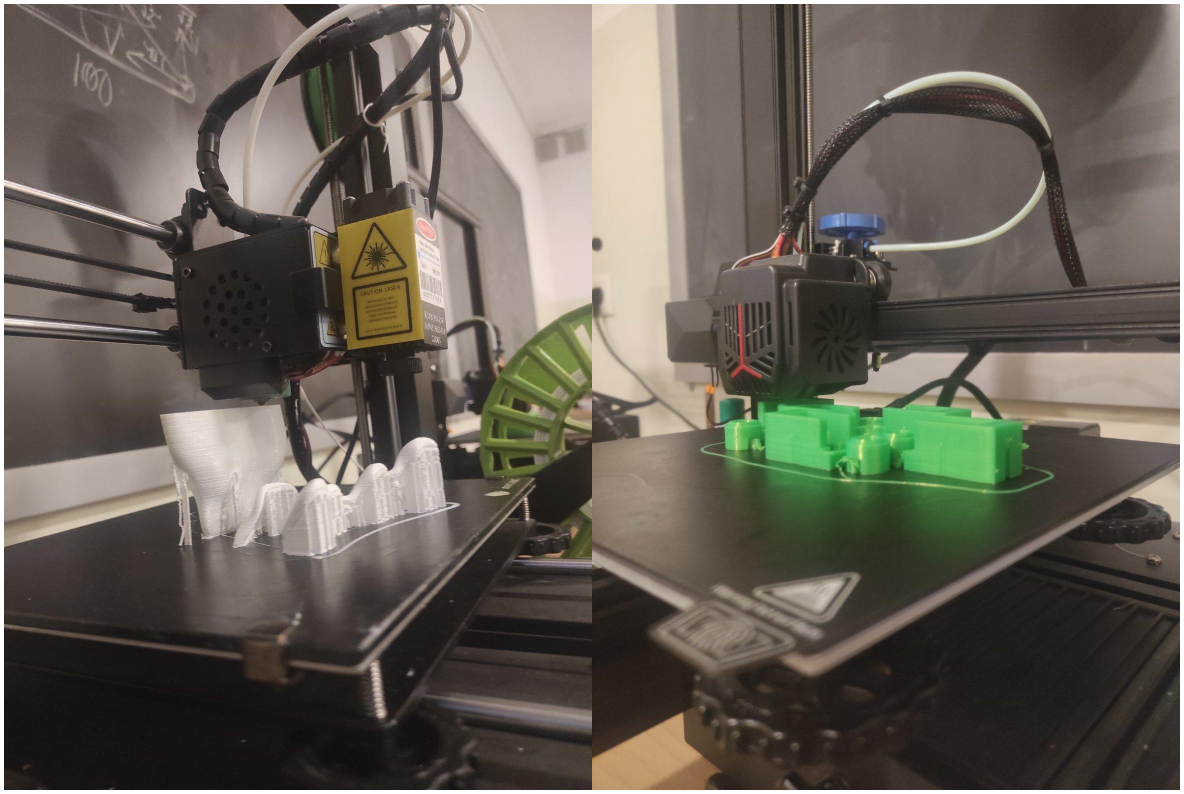




Աշխատանքի համար ընտրվեց INMOOV ռոբոտի ձեռքի մոդելը, բայց այս մոդելի վատ կողմն այն էր, որ սրա շարժիչները տեղադրվում էին բազուկի հատվածում՝ զբաղեցնելով բավականին մեծ տարածք, այդ իսկ պատճառով կայացվեց որոշում ձևափոխել մոդելի ափի հատվածը, դարձնելով այն այնպիսին, որ շարժիչները տեղադրվեն ափի մեջ: Աշխատանքի մեղանիզմում սկզբում եղան որոշակի խոչընդոտներ, քանի որ երբ շարժիչները երբ տեղադրվում էին բազուկի հատվածում, նրանք կանգնեցված դիրքով էին և զբաղեցնում էին շատ տարածք, բայց չէին խանգարում միմյանց, այդ իսկ պատճառով մոդելավորվեցին սովորական շարճիչի համար նախատեսված գլխիկներ, որոնք տալիս էին գրեթե նույն արդյունքը ինչ կանգնեցված շարժիչները:



Բազուկի հատվածը ամբողջովին վերցվել է INMOOV ռոբոտի ձեռքի մոդելից: Որպեսզի ափի չափսերը համընկնեն բազուկի չափսերի հետ, 9g սեռվո շարժիչների համար ստեղծվել են պատյաններ, այնուհետև դրանք փոխարինելով ավելի մեծ շարժիչների, որոնք հնարավորություն կտան ավելի ծանր իրեր բարձրացնել: Մոդելը տպվել է AnyCubic I3 Mega և Ender 3D տպիչվներով: Օգտագործվել է PLA ֆիլամենտ, մասերը տպվել են 30% լցոնությամբ, 2մմ պատի հաստությամբ, բացի բազուկի հատվածներից մնացած մասերը տպվել են օգնող հատվածներով(support):

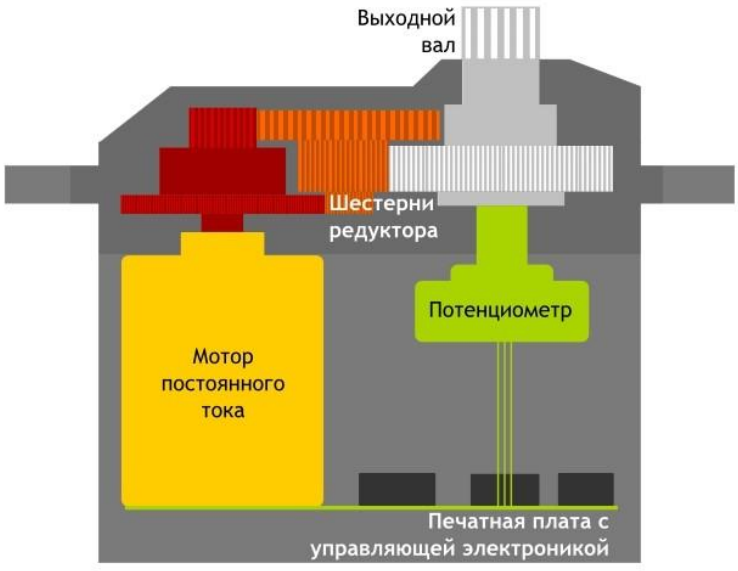


ԳԼՈՒԽ 3. Էլեկտրոնիկա

Օգտագործված էլեկտրոնիկան՝

- Arduino Nano
- 2 հատ 3.7v մարտկոց
- Dc-dc փոխարկիչ
- 5 հատ 9g սեռվո շարժիչ

9g սեռվո շարժիչը՝ ամենափոքրը սեռվո շարժիչներից, ստացել է իր անվանումը նրանից, որ նրա զանգվածը մոտ 9 գրամ է: Սեռվո շարժիչին միացվում են 3 լար, սև կամ շագանակագույն լարը դա - ն է, կարմիրը + ը, իսկ դեղինը սիգնալի լարն է, որը միացվում է Arduino - ին: Սեռվո շարժիչը իր մեջ ներառում է էլեկտրական շարժիչ, ռեդուկտոր՝ ատամնանիվներ, պոտենցիոմետր և պլատա: Էլեկտրական շարժիչը էլեկտրական իմպուլսները դարձնում է մեխանիկական էներգիայի: Իսկ ռեդուկտորը մեզ հնարավորություն է տալիս այդ մեխանիկական էներգիան դարձնել օգտակար աշխատանք: Պոտենցիոմետրը հնարավորություն է տալիս իմանալ թե տվյալ ինչքան աստիճանի տակ է թեքված սեռվո շարժիչը: Այն աշխատում է գլխավոր ատամնանիվի հետ, և փոխում իր դիմադրությունը, կախված ատամնանիվի դիրքից:



Քանի որ 9g սեռվո շարժիչին հարկավոր է 4.8v - 6v լարում, իսկ մենք օգտագործում ենք 2 հատ 3.7v լարում ունեցող մարտկոցներ, այդ իսկ պատճառով օգտագործում ենք dc-dc փոխարկիչ, որպեսզի ստանանք 5v լարում 7.4v լարումից: Մենք օգտագործում ենք Arduino Nano բայց կարող ենք օգտագործել նաև Uno կամ ուրիշ

միկրոկոնտրոլլեր: Ամբողջ էլեկտրոնիկան նախ հավաքում ենք breadboard ի վրա որպեսզի հասկանանք արդյոք ամեն բան նորմալ է, իսկ հետո արդեն ամեն բան կարող ենք միացնել իրար առանց breadboard -ի, որպեսզի սեռվոների միացումը arduino nano-ի հետ լինի հարմար, փոքր պլատայի վրա ստեղծվել են սեռվոների համար միացման կետեր, այդ պլատային է միանում հոսանքի աղբյուրը, իսկ պլատայից 5 լար միանում են միկրոկոնտրոլլերի թվային կետերին և մեկ լար միկրոկանտրոլլերի բացասականին: Զարմանալի է թե որքան պարզ միացումներից, որքան հետաքրքիր բան կարող ենք ստանալ: Միացումները, զոդումը անելուց հետո անցնում ենք սեռվոները շարժելու համար նախատեսված ծրագիրը գրելուն, որի մասին կխոսենք հաջորդ գլխում:

ԳԼՈՒԽ 4. Ծրագրային ապահովում

Սեռվոների կառավարում

Մինչև քննարկելը թե ինչպես է համակարգիչը ստանում մեր ձայնային հրահանգները և կառավարում ձեռքը, ինչպես նաև բացատրելը թե ինչպես է ստեղծվել ընհանուր ծրագիրը, եկեք քննարկենք և հասկանանք թե ինչպես է աշխատում սեռվոների կառավարումը: Ամբողջ կոդը գրվել և փորձարկվել է Arduino IDE ծրագրում: Համակարգչի և Arduino-ի միջև հաղորդակցությունը իրականացվում է serial մոնիտորի միջոցով: Պրոտոկոլը որով իրականացվում է հաղորդակցույթունը համակարգչի և Arduino-ի միջև կոչվում է USB (Universal Serial Bus):

Եկեք առանձնացնենք և մաս առ մաս հասկանանք կոդը՝

1. Գրադարանի միացում և փոփոխականների սահմանում

```
#include <Servo.h>
const int size = 5;
char input[size];
int servoPins[] = {6, 7, 8, 9, 10};
Servo servos[size];
```

- `#include <Servo.h>` միացնում ենք գրադարանը, որը հնարավորություն է տալիս կառավարել սեռվո շարժիչները
- `const int size = 5` սահմանում ենք սեռվո շարժիչների քանակը
- `char input[size]` այստեղ ենք պահելու serial մոնիտորում ներմուծվածը
- `int servoPins[] = {6, 7, 8, 9, 10}` այս զանգվածում մենք ներառել ենք բոլոր այն կետերը(pin) որոնց միանալու են սեռվո շարժիչները
- `Servo servos[size]` ստեղծում ենք սեռվո օբյեկտներից բաղկացած զանգված

2. Ֆունկցիա Setup()

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

    for (int i = 0; i < size; i++) {
        servos[i].attach(servoPins[i]);
    }
}
```

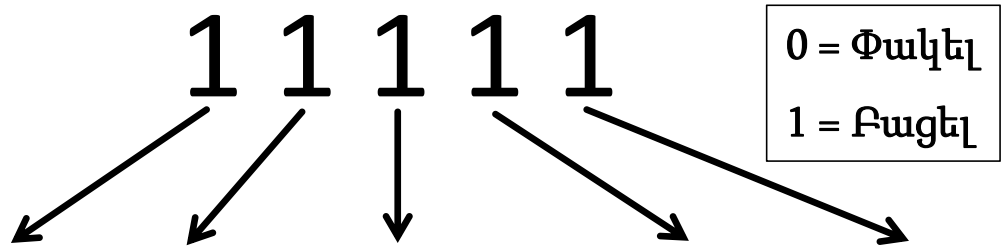
- `Serial.begin(9600)` սկսում ենք serial հաղորդակցությունը համակարգչի և arduino - ի միջև 9600 բիթ/վ
- `for (int i = 0; i < size; i++) {…}` միացնում ենք յուրաքանչյուր սեռվո շարժիչ arduino ի վրա իրեն համապատասխան կետին(pin)

3. Ֆունկցիա loop()

```
void loop() {
    if (Serial.available() >= size) {
        String servoStates = Serial.readStringUntil('\n');
        if (servoStates.length() == size) {
            for (int i = 0; i < size; i++) {
                if (servoStates.charAt(i) == '1') {
                    servos[i].write(0);
                } else {
                   servos[i].write(180);
                }
            }
        }
    }
}
```

- `if (Serial.available() >= size) {…}` ստուգում ենք արդյոք կա բավարար ինֆորմացիա ստանալու համար
- `String servoStates = Serial.readStringUntil('\n')` սրա մեջ պահում են ներմուծված տվյալները կարդում է մինչև Enter սեղմելը, '\n' նշանակում է նոր տող, ուր մենք անցնում ենք Enter սեղմելիս
- `if (servoStates.length() == size) {…}` ստուգում ենք արդյոք ներմուծվածը մեզ անհրաժեշտ երկարության է

- `for (int i = 0; i < size; i++) {…}` այս ցիկլով մենք անցնելու ենք բոլոր սեռվո շարժիչների վրայով սահմանելով նրանց դիրքերը՝ կախված ստացած տվյալներից
- `if (servoStates.charAt(i) == '1') {servos[i].write(0);}` ստուգում ենք servoStates-ի i-երորդ էլեմենտը եթե հավասար է 1-ի, թեքում ենք սեռվո շարժիչը 0^0-ի տակ
- `else {servos[i].write(180);}` հակառակ դեպքում թեքում ենք սեռվո շարժիչը 180^0-ի տակ

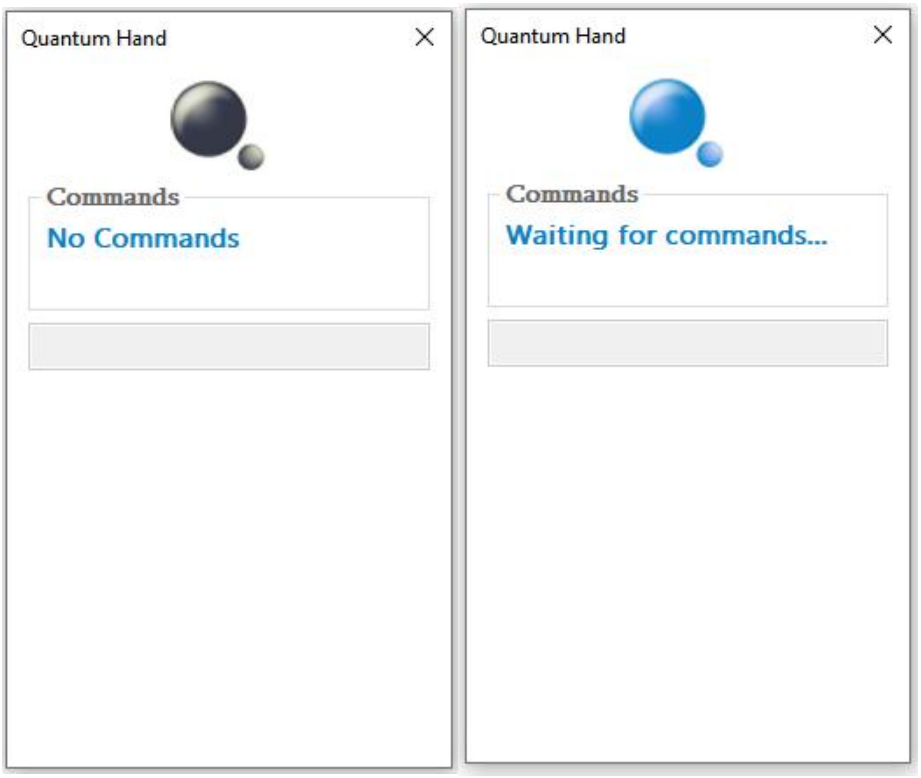


	Բութ	Ցուցամատ	Միջնամատ	Մատնեմատ	Ճկույթ
Զրո	0	0	0	0	0
Մեկ	0	1	0	0	0
Երկու	0	1	1	0	0
Երեք	1	1	1	0	0
Չորս	1	1	1	1	0
Հինգ	1	1	1	1	1

Մենք քննարկեցինք սեռվոների կառավարման հասարակ ծրագիրը, իսկ այժմ եկեք հասկանանք թե ինչպես է աշխատում ձայնային հրահանգներով կառավարումը։

Ինտերֆեյսի Ձևավորում

Ձայնային հրահանգներով կառավարումը իրականացվում է c# ծրագրավորման լեզվով, լեզուն բավական հնարավորություններ է տալիս թե ձայնային կառավարման ընդգրկման համար, թե User Interface ստեղծելու համար։

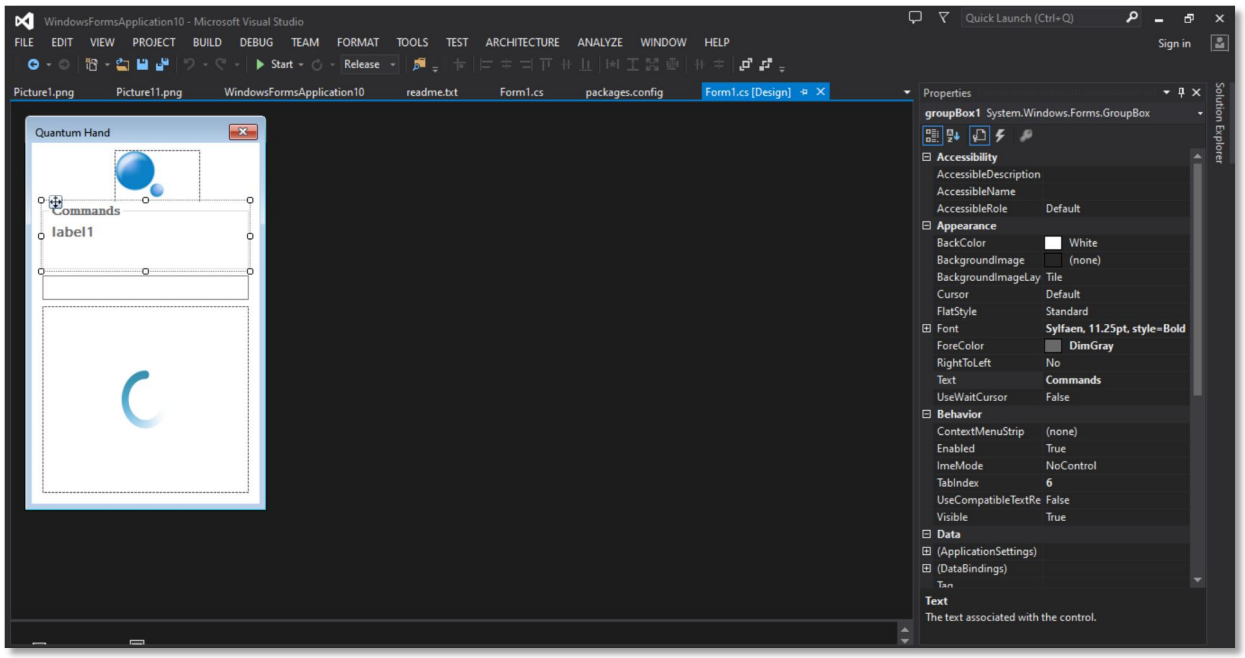


Ծրագիրն ունի բավական սովորական տեսք, երբ միացնում ենք ծրագիրը այն պասիվ վիճակում է, այսինքն ոչ մի հրահանգ չի կատարի, որքան էլ ասենք, որպեսզի ծրագիրը լսել մեզ, և կատարել հրահանգները, պետք է ասենք “Hey Quantum” արտահայտությունը կամ սեղմենք Քվանտի լոգոյին։ Երբ այն ձեռք բերի գույն և տեսնենք Waiting for commands արտահայտությունը, կարող ենք ասել ձեռքին թե ինչ անել։ Ինչպես օրինակ Google -ի օգնականը այնսպես էլ սրա դեպքում, ամեն անգամ հրահանգներ տալուց առաջ պետք է ասել **“Hey Quantum”** արտահայտությունը։

Հրահանգներից մենք կարող ենք ասել՝

- Open your fingers
- Close your fingers
- Exit
- Object is
  - Book
  - Apple
  - Banana
  - Ball
  - Small object
  - Large object

Երբ մենք ասում ենք **“Object is”** ծրագիրը մեզ ասում է **"Tell me what object is it or write"**, այնուհետև մենք ինչպես կարող ենք արտասանել վերևում նշվածները, այնպես էլ գրել մեկ այլ առարկա: Ծրագրի ինտերֆեյսը ձևավորվել է visual studio 2013-ում: Քվանտի լոգոն 2 սովորական picturebox -եր են:



[Ծրագիրը ներսից]

Ձայնային հրահանգներ

1. Եկեք հասկանանք թե ինչպես է իրականանում ձայնի ներմուծումը և արտածումը: Մենք ծրագրի ամենասկզբում հայտարարում ենք երկու գրադարան, որոնցից առաջինը թույլ է տալիս ծրագրին հասկանալ մեր հրահանգները, իսկ երկրորդը հնարավորություն է տալիս ծրագրին “խոսել”:

```
using System.Speech.Recognition;

using System.Speech.Synthesis;
```

2. Այնուհետև հայտարարում ենք 2 օբյեկտ՝ SpeechRecognitionEngine և SpeechSynthesizer տիպի, որպեսզի կարողանանք օգտվել գրադարաններից:

```
SpeechRecognitionEngine recognition = new SpeechRecognitionEngine();
SpeechSynthesizer synthesizer = new SpeechSynthesizer();
```

3. Այնուհետև սահմանում ենք արտածվող(համակարգչի կողմից արտասանվող) խոսքի տոնը, մենք կարող ենք ընտրել իգական, արական կամ նեյտրալ, տվյալ ծրագրում մենք օգտագոնծում ենք նեյտրալը:

```
synthesizer.SelectVoiceByHints(VoiceGender.Neutral);
```

4. Հետո սահմանում ենք Choices տիպի օօբյեկտ, որն իր մեջ ներառելու է բոլոր այն բառերը որոնք կարող են ներմուծվել:

```
Choices cmd = new Choices();

cmd.Add(new string[] { "Open your fingers", "Close your fingers",  "hey quantum", "object is", "apple", "banana", "a book", "cup", "ball", "small object", "large object", "Exit"});
```

5. Իսկ հետո ստեղծում ենք օբյեկտներ որոնց շնորհիվ համակարգիչը կհասկանա նախորդիվ ստեղծված Choices տիպի օբյեկտի պարունակությունը:

```
GrammarBuilder builder = new GrammarBuilder();

builder.Append(cmd);

Grammar grammer = new Grammar(builder);

recognition.LoadGrammarAsync(grammer);
```

6. Հաջորդ տողով մենք սահմանում ենք, որ 2-րդ կետում ստեղծած recognition օբյեկտը մեր խոսքը լսելու համար օգտագործի այն սարքը, որը համակարգչում սահմանված է որպես գլխավոր խոսափող:

```
recognition.SetInputToDefaultAudioDevice();
```

7. Այնուհետև հայտարարում ենք թե ինչ անի ծրագիրը, երբ հասկանա մեր կողմից տրված հրահանգը։ Որպես գործողություն սահմանում ենք այնուհետև գրված recognized_voice ֆունկցիան։

```
recognition.SpeechRecognized += recognized_voice;
```

8. recognized_voice ֆունկցիայում նշված են դեպքերը թե ինչ հրահանգ կարող է ստանալ ծրագիրը և ինչ պետք է անի դրանց դեպքում։ Ամբողջական ֆունկցիային և ծրագրին կարող եք ծանոթանալ Հավելված բաժնում։

Գլուխ 5. Ձեռքի կառավարման այլ մեթոդներ

Էլեկտրոմիոգրաֆիա

Իհարկե ձայնային հրահանգներով ձեռքի կառավարումը պրակտիկ չէ, եթե մենք ուզում ենք այն օգտագործել որպես պրոթեզ: Ինչպես արդեն նշել ենք ներկայումս բոլոր պրոթեզները կառավարվում են ի շնորհիվ մկանների կողմից ստեղծված էլեկտրական սիգնալների: Այս հետազոտական աշխատանքի ընթացքում ես ևս նպատակ էի դրել, կատարել ձեռքի շարժում էլեկտրոմիոգրաֆիայի շնորհիվ, քանի որ ես օգտագործում էի 1 ալիքանոց էլեկտրոմիոգրաֆիայի սենսոր, որն ուներ 3 էլեկտրոդ, սա սահմանափակում էր իմ գործողությունները, քանի որ սրա շնորհիվ հնարավոր չէր ճշգրիտ տեղեկություններ ստանալ ձեռքի յուրաքանչյուր մատի համար պատասխանատու մկանների շարժման մասին, առավելագույնը, որ կարող էր անել այս սենսորը, հասկանալ թե երբ ենք փակում մեր ափը: Բայց նույնիսկ այսպես, սա բավական հետաքրքիր էր, ցավոք սենսորի աշխատանքը խափանվեց, իսկ հետո դադարեց աշխատելը: Բայց մինչև դադարելը, ունեցել էի հնարավորություն գրել arduino ծրագիր, որի շնորհիվ ստանում էինք սենսորի կողմից տրվող արժեքները:

```
const int NUM_READINGS = 150;
int emgReadings[NUM_READINGS];
int i = 0;
boolean thresholdSet = false;
int thresholdValue = 50;
const int ledPin = 9;
void setup() {
Serial.begin(9600);
pinMode(ledPin, OUTPUT);
Serial.println("Sharjeq matery...");
unsigned long startTime = millis();
while (millis() - startTime < 30000 && i < NUM_READINGS) {
emgReadings[i] = analogRead(A0);
delay(100);
i++;
}
long sum = 0;
for (int j = 0; j < i; j++) {
sum += emgReadings[j];
```

```
}
thresholdValue = sum / i;
thresholdSet = true;
Serial.println(thresholdValue);
}
void loop() {
int emgValue = analogRead(A0);
if (emgValue > thresholdValue) {
Serial.print("               ");
Serial.print(emgValue);
Serial.print("               ");
Serial.print(thresholdValue);
Serial.println("               ");
} else {
Serial.println(emgValue);
}
delay(100);
}
```

Այս ծրագրով մենք ստանում էինք սենսորի արժեքները, սկզբում հավաքում էնք որոշակի քանակության արժեքներ, հաշվում ենք դրանց միջին թվաբանականը, և գտնում այն արժեքը, որը ստանում էինք երբ շարժում էինք մեր ձեռքը: Քանի որ ես չունեի խորը գիտելիքներ կենսաբանության մեջ, ինչպես նաև մարդու ձեռքի կառուցվածքի, ես ձեռքի այն կետերը, որտեղ պետք է ամրացվեին էլեկտրոդները, գտել եմ փորձերի և դիտումների շնորհիվ: Տարբեր կետերում ամրացնելով էլեկտրոդներ գտել եմ այն կետերը որտեղ առավել ճշգրիտ արդյունքներ էին ստացվում: Տվյալ սենսորը աշխատում է arduino-ի հետ միասին, սենսորը սնուցվում է առանձին հոսանքի աղբյուրից ընհանուր: Մենք arduino-ով կարողանում ենք ստանալ արժեքները որովհետև, սենսորը մկաններից ստացած ինֆորմացիան վերածում է փոփոխական լարման, որը մենք կարող ենք ստանալ ցանկացած միկրոկոնտրոլլերի անալուգ մուտքով:

Եզրակացություն

Ամփոփելով այս հետազոտական աշխատանքը կարող ենք եզրակացնել, որն այն հաջողված է, ցավոք ոչ այնպես ինչպես սկզբնական նպատակն էր, այդ իսկ պատճառով աշխատանքը ավարտված չէ քանի դեռ չի ինտեգրվել էլեկտրոմիոգրաֆիայի միջոցով կառավարումը, որը հնարավորություն կտա ստեղծել լիարժեք պրոթեզ: Այս աշխատանքը ինձ հնարավորություն տվեց հասկանան թե որքան մեծ հնարավորություններ ունի 3D տպագրում:

Այս հետազոտական աշխատանքը նաև հնարավորություն ընձեռեց ծանոթանալ Arduino միկրոկոնտրոլլերների հետ՝ աշխատել դրանց հետ: Հմտություններ ձեռք բերել տարբեր սարքավորումների աշխատելու մեջ՝ առաջին հերթին զոդման մեջ, ինչպես նաև 3D տպիչների:

Այսպիսով ստացա գիտելիքներ որոնք կարող եմ կիրառել այլ նախագծերում, ակմ ինչու ոչ հենց այս նախագիծը դարձնել ավելին քան թե ուղղակի հետազոտական աշխատանք: Սա դժվար, բայց միաժամանակ հետաքրքիր և փորձություններով լի աշխատանք էր, որն օգնեց որոշակի չափով զարգանալ որպես ինժեներ:

Առանձնահատուկ շնորհակալությունս եմ ուզում հայտնել իմ գիտական ղեկավարին՝ ընկ. Աղաբաբյանին, աշխատանքի ընթացքում ինձ օժանդակելու և անհրաժեշտ իրերը տրամադրելու համար, ինչպես նաև ընկ. Աղամիրյանին:

Հավելված

Arduino -ի ծրագիր

```
#include <Servo.h>
const int size = 5;
char input[size];
int servoPins[] = {6, 7, 8, 9, 10};
Servo servos[size];


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

    for (int i = 0; i < size; i++) {
        servos[i].attach(servoPins[i]);
    }
}


void loop() {
    if (Serial.available() >= size) {
        String servoStates = Serial.readStringUntil('\n');
        if (servoStates.length() == size) {
            for (int i = 0; i < size; i++) {
                if (servoStates.charAt(i) == '1') {
                    servos[i].write(0);
                } else {
                   servos[i].write(180);
                }
            }
        }
    }
}
```

Ձայնային հրահանգներ

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Drawing.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Speech.Recognition;
using System.Diagnostics;
using WindowsFormsApplication10.Properties;
using System.Speech.Synthesis;
using System.IO;
using System.Drawing.Imaging;
using System.Media;
namespace WindowsFormsApplication10
{
    public partial class Form1 : Form
    {
        SpeechRecognitionEngine recognition = new SpeechRecognitionEngine();
        SpeechSynthesizer synthesizer = new SpeechSynthesizer();
        bool command = false;
        bool query = false;
        public Form1()
        {
            InitializeComponent();
           serialPort1.Open();
            label1.ForeColor = Color.FromArgb(12, 129, 199);
            label1.Text = "No Commands";
            textBox1.KeyPress += TextBox1_KeyPress;
            button2.Visible = false;
```

```
            synthesizer.SelectVoiceByHints(VoiceGender.Neutral);
            Choices cmd = new Choices();
            cmd.Add(new string[] { "Open your fingers", "Close your fingers",  "hey
quantum", "object is", "apple", "banana", "a book", "cup", "ball", "small object",
"large object", "Exit"});
            GrammarBuilder builder = new GrammarBuilder();
            builder.Append(cmd);
            Grammar grammer = new Grammar(builder);
            recognition.LoadGrammarAsync(grammer);
            recognition.SetInputToDefaultAudioDevice();
            recognition.SpeechRecognized += recognized_voice;
            recognition.RecognizeAsync(RecognizeMode.Multiple);
        }
private void recognized_voice(object sender, SpeechRecognizedEventArgs e)
{
    switch (e.Result.Text)
    {
        case "hey quantum":
            if (query == false)
            {
                pictureBox2.Visible = false;
                Enable_cmd();
            }
                break;
        case "Open your fingers":
            if (command==true && query==false)
            {
                synthesizer.Speak("Opening fingers");
                serialPort1.Write("11111");
                Disable_cmd();
            }
            break;
        case "Close your fingers":
            if (command == true && query == false)
            {
                synthesizer.Speak("Closing fingers");
```

```
            serialPort1.Write("00000");
            Disable_cmd();
        }
        break;
    case "object is":
        if (command == true && query == false)
        {
            recognition.RecognizeAsyncStop();
            synthesizer.Speak("Tell me what object is it or write");
            textBox1.Text = "";
            recognition.RecognizeAsync(RecognizeMode.Multiple);
            textBox1.Enabled=true;
           query = true;
        }
        break;
    case "Exit": if (command == true && query == false)
        {
            Application.Exit();
        }
        break;
    case "a book":
    case "cup":
    case "apple":
    case "banana":
    case "ball":
    case "small object":
    case "large object":
        if (e.Result.Text=="a book")
        {
            serialPort1.Write("01000");
        }
        else if (e.Result.Text=="apple")
        {
            serialPort1.Write("00000");
        }
        else if (e.Result.Text == "cup")
```

```
            {
                serialPort1.Write("00111");
            }
            else if (e.Result.Text == "ball")
            {
                serialPort1.Write("00000");
            }
            else if (e.Result.Text == "banana")
            {
                serialPort1.Write("00111");
            }
            else if (e.Result.Text=="small object")
            {
                serialPort1.Write("00111");
            }
            else if(e.Result.Text=="large object")
            {
                serialPort1.Write("00000");
            }
            if (command==true && query==true)
            {
                textBox1.Enabled = false;
                textBox1.Text = e.Result.Text;
                WriteToQueryFile(textBox1.Text);
                string datasetPath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "dataset");
                string queryFolderPath = Path.Combine(datasetPath, e.Result.Text);
                if (Directory.Exists(queryFolderPath))
                {
                    string[] imageFiles = Directory.GetFiles(queryFolderPath, "*.jpg")
                                .Concat(Directory.GetFiles(queryFolderPath, "*.png"))
                                .ToArray();

                    if (imageFiles.Length > 0)
                    {
                        string imagePath = imageFiles[0];
```

```
                        pictureBox2.Visible = false;
                        pictureBox1.Image = Image.FromFile(imagePath);
                        serialPort1.Write("00000");
                    }
                    else
                    {
                        RunExternalExe();
                        pictureBox2.Visible = true;
                    }
                }
                else
                {
                    RunExternalExe();
                    pictureBox2.Visible = true;
                }
                query = false;
                Disable_cmd();
            }
break;
    }
}
private void TextBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)Keys.Enter)
    {
        if (textBox1.Enabled && !string.IsNullOrWhiteSpace(textBox1.Text))
        {
            string inputText = textBox1.Text.Trim();
            WriteToQueryFile(inputText);
            string datasetPath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "dataset");
            string queryFolderPath = Path.Combine(datasetPath, inputText);
            if (Directory.Exists(queryFolderPath))
            {
                string[] imageFiles = Directory.GetFiles(queryFolderPath, "*.jpg");
                if (imageFiles.Length > 0)
```

```
                {
                    string imagePath = imageFiles[0];
                    pictureBox2.Visible = false;
                    pictureBox1.Image = Image.FromFile(imagePath);
                    serialPort1.Write("00000");
                }
                else
                { RunExternalExe();
                    pictureBox2.Visible = true;
                }
            }else{
                RunExternalExe();
                pictureBox2.Visible = true;
            } textBox1.Enabled = false;
            query = false;
            Disable_cmd();
        }
    }
}
private void RunExternalExe()
{
        string currentDirectory = AppDomain.CurrentDomain.BaseDirectory;
        string externalExeFileName = "main.exe";
        string externalExePath = Path.Combine(currentDirectory, externalExeFileName);
        if (File.Exists(externalExePath))
        {
            Process.Start(externalExePath);
        }
}
         private void WriteToQueryFile(string itemName)
        {
                string desktopPath =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
                string filePath = Path.Combine(desktopPath, "query.txt");
                if (File.Exists(filePath))
                {
```

```
                File.Delete(filePath);
            }
            using (StreamWriter writer = File.AppendText(filePath))
            {
                writer.WriteLine(itemName);
            }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Enable_cmd();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            Disable_cmd(); }
        private void Enable_cmd()
        {
            button2.Visible = true;
            button1.Visible = false;
            command = true;
            label1.Text = "Waiting for commands...";
            SystemSounds.Exclamation.Play();
        }
        private void Disable_cmd()
        {
            button1.Visible = true;
            button2.Visible = false;
            command = false;
            label1.Text = "No Commands";
        }
        private void Form1_Load(object sender, EventArgs e)
        {
        }
        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
        }   }
}
```

Օգտագործված գրականության ցանկ

- https://learn.microsoft.com/ru-ru/dotnet/api/system.drawing.bitmap?view=dotnet-plat-ext-7.0
- https://metanit.com/sharp/windowsforms/1.1.php
- https://www.codeguru.com/csharp/windows-forms-controls/
- https://www.mbmproject.com/blog/tutorials/windows-forms-c-loops-and-arrays
- https://www.zebra0.com/csharp/arrays/
- Настройка UART соединения в Ардуино | АрдуиноПлюс (arduinoplus.ru)
- Arduino Servo, как подключить сервопривод + скетч, схема (arduino-site.ru)
- Images - Computer Vision Zone
- Сервоприводы: подключение, управление, скетчи Ардуино (podkluchaemvse.ru)
- Electromyography - Wikipedia
- Overview | Getting Started with MyoWare Muscle Sensor | Adafruit Learning System
- МЕТОДЫ И АЛГОРИТМЫ АНАЛИЗА ПАТТЕРНОВ СИГНАЛОВ ЭЛЕКТРОМИОГРАФИИ НА ОСНОВЕ ЗАДАЧИ КЛАССИФИКАЦИИ ДАННЫХ | Кабанов Артемий Андреевич
- Методы и алгоритмы обработки электромиографического сигнала для управления механическими системами | Унанян Нарек Новлетович
- Hand and Forarm - InMoov
- Cognitive vision system for control of dexterous prosthetic hands: Experimental evaluation - PMC (nih.gov)