ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK	Hallgató neve:	**Molnár Dóra Katalin**
2022	Hallgató törzskönyvi száma:	**T/006309/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Biomatika és Alkalmazott Mesterséges Intelligencia Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Molnár Dóra Katalin**
Törzskönyvi száma: T/006309/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Automatikus karakteranimáció módszer fejlesztésének támogatása mozgáskövető technológiával

Motion capture technology for the development of automatic character animation methodology

Intézményi konzulens: Dr. Galambos Péter
Külső konzulens: Lántzky Anna

Beadási határidő: 2021. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
IoT, beágyazott rendszerek és robotika specializáció

A feladat

A humanoid robotok mozgásának leírása és szabályozása az egyik legösszetettebb része a robotikának, ez kiváltképp igaz azon robotokra, melyek az emberhez hasonlóan két lábon járnak és egyensúlyoznak. A legelterjedtebb módszer ennek megvalósítására az emberi mozgás rögzítése mozgáskövető technológiával és a leírt pálya lemásolásának megtanítása a robot számára. Ez a módszer, azonban költséges és időigényes, mert a teszteléshez humán erőforrásra van szükség.
A dolgozat a Mollia Zrt. fejlesztési irányaihoz illeszkedve, a vállalat támogatásával valósul meg és fő célja a vállalat által fejlesztett automatikus karakteranimáció tesztelése és ellenőrzése fizikai humanoid roboton. Az ellenőrzéshez szükséges a robotot a szimulációval összekötő környezet elkészítése és az eredmények validálása mozgáskövető szoftver segítségével, valamint az ideális motorhajtás kiválasztása.

A dolgozatnak tartalmaznia kell:

- a generatív mozgásleírás, a humanoid robotmodellezés, a szervomotorok robotikában való alkalmazásának és a mozgáskövető technológia szakirodalmának áttekintését,
- az animált és a fizikai robot felépítésének leírását,
- az egyszerűsített, 1 szabadsági fokú modell felépítését és az ezen végzett tesztek leírását,
- a különböző szervók kimeneti adatainak feldolgozására megvalósítandó szoftver specifikációját és rendszertervét,
- a mozgáskövető rendszerrel végzett tesztek és a szimulációk összehasonlítását,
- megfelelően részletes fejlesztői és felhasználói dokumentációt,
- az eredmények összefoglaló értékelését és az optimális motorhajtást.

ÓBUDAI EGYETEM Neumann János Informatikai Kar
Ph.

.....
Dr. Eigner György
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022.05.01.

.....
hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: MOLNÁR DÓRA KATALIN

Neptun kód: [redacted]

Tagozat: NAPPALI

Telefon:

Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

AUTOMATIKUS KARAKTERANIMÁCIÓ MÓDSZER FEJLESZTÉSÉNEK TÁMOGATÁSA MOZGÁSKÖVETŐ TECHNOLÓGIÁVAL

Szakdolgozat / Diplomamunka[2] címe angolul:

MOTION CAPTURE TECHNOLOGY FOR THE DEVELOPMENT OF AUTOMATIC CHARACTER ANIMATION METHODOLOGY

Intézményi konzulens: DR. GALAMBOS PÉTER

Külső konzulens: LÁNTZKY ANNA

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.02.25	TÉMA ÁTBESZÉLÉSE	
2.	2021.03.12	IRODALOMKUTATÁS ÁTTEKINTÉSE	
3.	2021.04.20	A TERVEZETT RENDSZER SPECIFIKÁCIÓJA	
4.	2021.04.30	RENDSZERTERV EGYEZTETÉSE	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2021.05.03.

intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: MOLNÁR DÓRA KATALIN

Neptun kód: [redacted]

Tagozat: NAPPALI

Telefon:

Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

AUTOMATIKUS KARAKTERANIMÁCIÓ MÓDSZER FEJLESZTÉSÉNEK TÁMOGATÁSA MOZGÁSKÖVETŐ TECHNOLÓGIÁVAL

Szakdolgozat / Diplomamunka[2] címe angolul:

MOTION CAPTURE TECHNOLOGY FOR THE DEVELOPMENT OF AUTOMATIC CHARACTER ANIMATION METHODOLOGY

Intézményi konzulens: DR. GALAMBOS PÉTER

Külső konzulens: LÁNTZKY ANNA

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.09.14	IMPLEMENTÁCIÓS LEHETŐSÉGEK EGYEZTETÉSE	
2.	2021.10.08	IRÁNYÍTÁSI PROGRAM TERV	
3.	2021.11.03	TESZTELÉSI TERV	
4.	2021.12.01	VÉGSŐ ÁTTEKINTÉS, TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2021.12.07.

intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

TARTALOMJEGYZÉK

1. BEVEZETÉS

Napjaink egyik leggyorsabban fejlődő területe a robotika, az élet minden területén találkozhatunk velük az elmúlt évtizedek fejlesztéseinek köszönhetően, azonban még mindig számtalan megoldatlan kérdés és újítási lehetőség rejlik a témában. A robotikában beszélhetünk különböző felhasználású robotokról ilyen az ipari, gyógyászati, nano, katonai, fogyasztói vagy a humanoid robot. Ezek között a fő különbség a méret, a teherbírás és a pontosság. Felhasználásukból adódóan fontos tényező, hogy egy robotnak milyen kialakítása van, például nano méretűnek kell lennie, hogy kis helyre beférve képes legyen a feladatát véghezvinni vagy magas precizitással kell rendelkeznie, mint egy gyógyászati robotnak.

Egy az emberhez hasonlóan két lábon járó és egyensúlyozó humanoid robot mozgásának leírása és szabályozása igen összetett feladatot foglal magába. A dolgozatom a Mollia Zrt. fejlesztési tervein alapuló automatikus karakteranimáció tesztelése és ellenőrzése egy fizikai humanoid robot fejlesztéséhez. A robotdinamikában a robotok mozgásának leírása két tagból épül fel, ezek a csuklók (joint) és kartagok (link). Egy robot felépítése ezek láncolatából épül fel fa felépítésben. Jelen dolgozat fő témája a vállalat által fejlesztett robot csuklóinak tesztelése, hogy a tervezés további szakaszaira pontosan ismertek legyenek a cég által jelenleg használt szervómotorok jellemzői.

A dolgozat során egy mikroszervó tesztelési folyamatát mutatom be, ennek első lépése a szervó irányításához használt kód megírása volt, melyet a különböző tesztesetek rögzítése követett. A mikrokontrolleren keresztül vezérelt szervót az Óbudai Egyetem Mozgáslaborjában mozgáskövető technológiával rögzítettem, majd a kapott nyers adatokat egy python kód segítségével dolgoztam fel. A mozgáskövető technológia azért volt szükséges a szervó mozgásának validálására, hogy a motorba beépített feedback pontosságát is tesztelni tudjuk.

A humanoid robot mozgásánál két fontos elvárásunk van a robotot alkotó motorok felé a pontosság és a teherbírás. Annak érdekében, hogy a mozgás emberi legyen és az emberi szem által is hihetően emberinek tűnő mozgást kapjunk fontos, hogy a robot képes legyen minél kisebb léptékkel forogni. A magas terhelhetőségre főként azért van szükség, hogy az emberhez hasonlóan képes legyen bármely felvett pozícióját stabilan megtartani a gravitációs térben.

Összességében, a dolgozatom két fő célja egy szervómotor tesztelési folyamat összeállítása és dokumentálása, valamint egy kiválasztott motor megfelelőségének vizsgálata. A kapott eredményekből megállapítható lesz, hogy a kiválasztott motor megfelelő a célnak vagy sem.

2. IRODALOMKUTATÁS

2.1. Mollia Zrt. alapötlete

A Mollia Zrt. egy olyan matematikai modellt dolgozott ki, ami új utat nyithat meg a humanoid robotok mozgásának fejlesztésében. Céljuk, hogy az általuk készített szoftver segítségével a fizikailag megépített emberszerű robotokat is tanítani lehessen. Ennek segítségére létrehoztak egy olyan online játékot, mely központi motívuma a szimulált robotok mozgásának tanítása egy intuitív felhasználói interfészen keresztül. Fontos volt, hogy a gépi mozgástanítást hétköznapi felhasználók számára is hozzáférhetővé tegye szórakoztató formában. Ezért lett a cél egy tanuláson alapuló online játék.

2.2. Mollia Zrt. ötletéhez hasonló fejlesztések: Kinesthetic Bootstrapping

Heni Ben Amor, Erik Berger, David Vogt és Bernhard Jung tanulmánya a motorikus készségek tanításáról szól humanoid robotok számára egyfajta fizikai interakción keresztül. Első lépésként a tanító megmozgatja a robot ízületeit, hogy közvetítse a kívánt mozgást vagy viselkedést. A bemutatás során 20 Hz-es frekvenciával rögzítik a robot motorkonfigurációit. A tanulmányban használt robot egy Bioloid robot, ami 18 szervomotorral, azaz 18 szabadságfokkal rendelkezik. Minden egyes lépésben rögzítik az összes szervomotor állapotát, ami egy 18 dimenziós p vektort eredményez. Ha a tanítási fázis befejeződik, összegyűjtik az összes ilyen vektort, hogy kiszámítsák az alacsony dimenziós testtartás modelljét. Ezután ennek segítségével értékelik a mozgás különböző változatait. Ez a robot fizikai alapú virtuális valóság szimulációjában történik- A szimulátor lehetővé teszi a mozgás optimalizálását anélkül, hogy a robot hardverébe bele kellene avatkozni. Ha a bemutatott mozgás nagyon dinamikus (pl.: egy felálló mozgás) akkor fontos, hogy a robot megtanulja, hogyan számoljon az emberi tanító által korábban alkalmazott külső stabilizáló erőkkel. Az optimalizálási lépés befejezésével a tanult mozgás átkerül a fizikai robotra és a szimuláción kívül visszajátsszák.

2.3. Biológiailag inspirált robotok

A humanoid robotoknál fontosnak tartom megemlíteni az Asimov-féle három törvényt:

1. A robot nem támad emberre és óvja az embert
2. A robot engedelmeskedik az embernek, ha ez nem ütközik az 1. törvénybe.
3. A robot óvja magát, ha ez nem ütközik az 1. vagy a 2. törvénybe [1] [2].

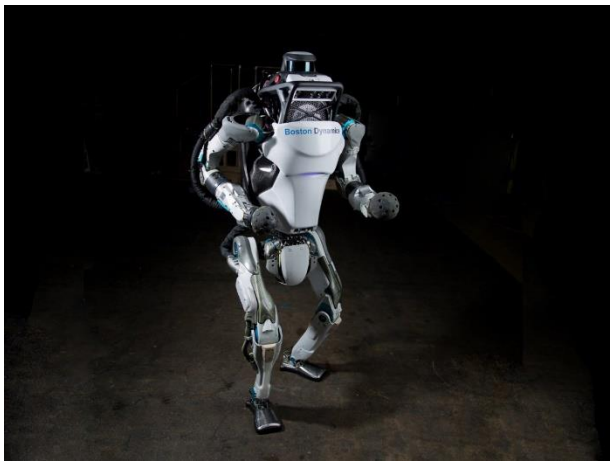
Ezen törvények figyelembevételével folynak az emberszerű robotok fejlesztései. Alapvetően képesek az emberrel együtt dolgozni és a kommunikációt is ki lehet úgy

alakítani, hogy emberi nyelven történjen az. A tanulás belső vezérléssel történik, mesterséges intelligencia segítségével [2].

A humanoid robotok jellegzetességei közt tartjuk számon, hogy komplex érzékelőkkel és az emberhez hasonlóan beavatkozó végtagokkal rendelkeznek. A robotot autonóm működés jellemzi, folyamatosan vizsgálja, elemzi, kiértékeli a környezetét. Dinamikusan változó környezethez is képesek alkalmazkodni. Az emberszerű felépítésük miatt képesek lesznek az embert pótolni bizonyos rendszerekben, képesek az ember helyett dolgozni vagy tudnak olyan feladatot is ellátni akár, ami az embereknek túl veszélyes vagy nem is lehetséges pl. egy más bolygón való tevékenység elvégzése (Perseverance, Csuzsung) Háborús környezetben is megoldást jelenthetnek sok veszélyes cselekvésre vagy más olyan területen is tudnak munkát végezni, amely az ember számára a szükséges fizikai erő hiánya miatt nem lehetséges.

A humanoid robotok akár emberi tulajdonságokkal is felruházhatók ilyen lehet a mimika, az artikuláció és az érzelmek kifejezése. Közlekedési módjukat tekintve megkülönböztethetünk sima, egyenletes felületen haladni képes robotot, illetve változatos, egyenetlen terepen közlekedőt.

A humanoid robotok egy fajtája a két lábon járó úgynevezett bipedális robot, mely az emberhez hasonlóan két lábon jár. Ez azonban a fejlesztői csapatok részéről igényel bonyolultabb tervezést, mert szükséges ilyen esetben az egyensúlyozásra képessé tenni a robotot. Azonban a két lábon járó megoldás a kerekeknél és a stabil robotoknál sokkal több előnnyel bír. Ilyen például, hogy képes önállóan mozogni, akár egy lépcsőt használni vagy egyenetlen terepen, törmelékeken is mozogni.



2.1. ábra Boston Dynamics Atlas

Jelenleg a világon több robotikai kutatócsoport is működik, melyek különböző technológiákat vesznek alapul a robot mozgásának leírásához és a robot felépítéséhez is. Az egyik legnagyobb megoldatlan kérdést a megfelelő áramforrás hiánya okozza, ugyanis

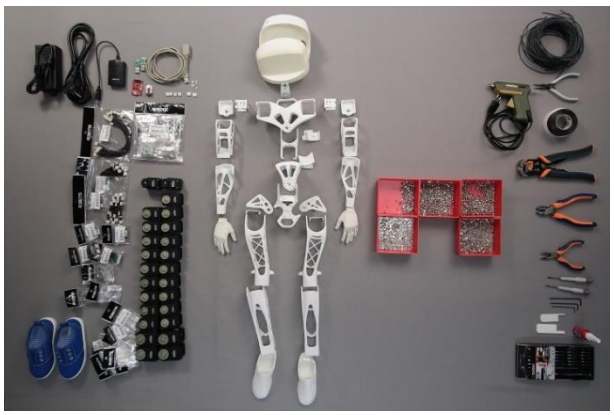
ha egy robot a hálózatról irányítható, akkor korlátozva van a mozgástere. Abban az esetben, ha a robot ellátása akkumulátorról történik, akkor olyan teherbírású robotra van szükség, amely képes az akkumulátorokat is mozgatni a saját testtömegén felül.

Az 2.1 ábrán a Boston Dynamics által kifejlesztett Atlas robot látható, mely jelenleg a legfejlettebb mozgású humanoid robot. Képes parkour mutatványokra és egyenetlen terepen is kiválóan megállja a helyét. Az Atlas készítése során távolságérzékelőket használ a környezet érzékelésére. A robotba beépített dinamikai modelljei alapján megjósolja, hogyan fog alakulni a mozgása idővel, és ennek megfelelően igazítja ehhez a mozgását.

2.3.1. Poppy

A Poppy Project egy nyílt forráskódú platform, ami interaktív 3D nyomtatott robotok létrehozásával, fejlesztésével, megosztásával foglalkozik. Ezeket a robotokat ugyanazon elvek alapján tervezték. Polimer alapanyagú 3D nyomtatott végtagokból és Dynamixel mikroszervókból épülnek fel. Beágyazott rendszereket használnak a vezérléshez pl.: Raspberry Pi vagy Odroid. Python könyvtáron, pypoton alapulnak, amely megkönnyíti a Dynamixel szervomotorok irányítását. Szimulált környezettel is rendelkeznek és Rest API-n keresztül is programozhatók, amely más programozási nyelvekkel is kompatibilissé teszi.

A Poppy Humanoid Robot egy 25 szabadságfokú robot. Kutatási, oktatási és művészeti célokra is használják laboratóriumok, mérnöki iskolák és művészeti projekteken dolgozók [3].



2.2. ábra Poppy project

2.3.1. Boston Dynamics Spot

A Boston Dynamics mérnökei számára fontos az innovatív technológiák és egyszerűsített munkafolyamatok előtérbe helyezése, ezáltal egy olyan problémára szerettek volna

megoldást találni, miszerint bizonyos iparágakban az adatgyűjtést gyakran korlátozott vagy veszélyes területekről szerezték be. Ezért létrehozták a Spot-ot, amely időt és erőforrást szabadít fel az alkalmazottak számára a magasabb szintű tevékenységekre való koncentrálás érdekében.

Egy külön csapat képezi ki a robotot neuronhálók segítségével. A neuronhálók egy tanulási algoritmusa a hiba-visszaterjesztéses. Ez azt jelenti, hogy az inputra a megadott számú input és output mintapárból összetevődő tanító adathalmazt adnak, és a kimeneten nem teljesen a mintapár szerint elvárt output értékének megfelelő kimenet jön létre [4] [5].

2.3. ábra Boston Dynamics Spot

2.1.3 Cassie

Cassie egy bipedal robot, vagyis két lábon sétáló robot. A dinamikus mozgásának egyik fontos szempontja a sétáló robotok sebességszabályozása, mivel például egy kísérő robot, aki az ember mellett sétál fontos, hogy stabilan járjon és a sebességét folyamatosan szabályoznia tudja a megfelelő távolság fenntartása érdekében. Különféle módszereket tártak fel ennek orvoslására, vannak, amelyek a visszacsatoló-vezérlőkön keresztül klasszikus szabályozási technikán alapulnak. Azonban vannak olyanok is, amik a gépi tanulási (machine learning) módszereket alkalmazzák.

A neurális hálózati szabályzók hatékony eredményei által az adaptív robotmanipulátorok szabályozására javasoltak egy szabályozási struktúrát, amely ötvözi az online tanulás előnyeit a klasszikus visszacsatolásos szabályozók robusztusságával, hogy kompenzálja a robot dinamikai tulajdonságaiban bekövetkező változásokat [6].

2.4. ábra Cassie robot

2.4. Mikroszervók

2.4.1. Lewansoul LX-16a

A Lewansoul LX-16a mikroszervó valós idejű visszajelzéseket ad a pozícióról, a hőmérsékletről, a feszültségről. A fogaskerék teljesen fém, ez a pontosság szempontjából előnyös és az élettartamot is jelentősen növeli. Kettős golyóscsapággyal rendelkezik az egyik a hajtó golyóscsapágy, a másik a segéd golyóscsapágy. A nagy pontosságú potenciométer segítségével a pontossága és a linearitása jelentősen javult a szervónak [7].

Sokszor használják robotokhoz, bár méreteit és technikai tudását tekintve nem egy felső kategóriás mikroszervó.

2.5. ábra Lewansoul LX-16A szervó

2.4.2. Herculex smart szervók

A Herculex egyike a legmodernebb moduláris intelligens szervóknak. Az egyenáramú motor képes akár 2,35 Nm nyomatékot biztosítani 7,4 V feszültség mellett. Minden szervóhoz két csatlakozó teszi lehetővé a soros és szükség esetén a párhuzamos csatlakozást. Különféle vezérlőalgoritmusokat hordoz, mint például PID, Feedforward és Trapezoidal Velocity Profile. A UART soros kommunikáció használatával egyszerűen megváltoztathatjuk a sebességet, a pozíciót, a LED-et, a leállást és az állapotot akár 254 szervónál egyidejűleg. Visszajelzéseket is kapunk a belső hőmérsékletről, a helyzetről és a túlterhelés érzékelőktől. A szervó hét különböző típusú hiba diagnosztizálására képesek, ezeket LED jelzi. A felhasználását különösen ajánlják mechanikus karokhoz, robotokhoz, ízületekhez [8].

2.6. ábra Herculex DRS-0201 szervó

2.4.3. Dynamixel szervó

A Dynamixel szervókat úgy tervezték, hogy modulárisak és láncoltak legyenek bármilyen roboton vagy mechanikus kivitelben az erőteljes és rugalmas robotmozgások érdekében. Egy moduláris szervó tartalmaz: teljesen integrált egyenáramú motort, fogaskereket, belső vezérlőt, motor meghajtó áramkört, láncolható hálózati kapcsolatot. Ezeken felül tudja azokat a funkciókat, amit korábban a Herkulex szervóknál már említettem [9].

2.7. ábra Dynamixel szervócsalád

2.4.4. Technikai összehasonlítás (állítható paraméterekkel)

	Lewansoul LX-16A	Herculex DRS 0201	Dynamixel MX-28T
Méret [mm]	45x24x35	44,5x24x32	35,6x50,6x35,5
Súly	52 g	60 g	77 g
Forgási szög	240°	320°	360°
Pontosság/eltérés (resolution)	0,24°	0,325°	0,0879 fok/impulzus
Visszajelzés	Pozíció, hőmérséklet, feszültség	Pozíció, hőmérséklet, gyorsaság, betöltés, feszültség	Pozíció, hőmérséklet, bemeneti feszültség, betöltés [7] [8] [9]

1. táblázat technikai paraméterek összehasonlítása 3 szervónál

2.5. Motion Capture

A Motion Capture (rövidítve: mocap, magyarul: mozgásrögzítés) egy olyan mozgáskövető, több kamerás élő felvételen alapuló rendszer, amely a térben nagyon pontosan vissza tudja adni a pontok helyzetét és ebből szoftveresen vissza lehet építeni a karakter mozgását.

2.8. ábra Motion Capture 6 kamerás rendszer

2.5.1. Felhasználási területei

Gyakran használják filmeknél, videójátékoknál ezt a technológiát. A cél egyrészt, hogy a mozgás minél valósághűbb legyen ezeknél a játékoknál, filmeknél, másrészt a realisztikus emberi mozgás digitális létrehozása időigényes és nagy szakértelmet igénylő feladat, ezért váltják ki a Motion Capture technológiával.

2.9. ábra Mocap technológia néhány filmben

A mocap-pel nem csak ezeken a területeken találkozhatunk ugyanis a robotikában, illetve a biztonságtechnikai területen is igen nagy jelentőséggel bír. Manapság számos biometrikus azonosító rendszerrel találkozhatunk például ujjlenyomatazonosítás, retinaazonosító, de akár az emberi mozgás is lehet annyira jellegzetes, hogy azáltal be tudjunk azonosítani egy konkrét személyt [10].

2.5.2. Működési elve és felépítése

A Motion Capture technológia két részből épül fel: a kamerákból és a kamerák által követett markerekből.

A markerek a testhezálló ruhán helyezhetők el, lehetnek aktív vagy passzív típusúak. Aktív markerek esetében LED jelölőkről beszélhetünk, ezeket megfelelően konfigurálni és szinkronizálni kell a rendszerrel. A legjobb eredményeket 850 nm-es megvilágítási hullámhosszal rendelkező markerrel érhetjük el.

Passzív jelölőknél pedig nagy odafigyelést igényel a marker felülete, annak karban tartása, hogy megfelelően visszaverhessék az infravörös sugarakat a kamerába.
A mocaphoz elengedhetetlen egy rögzítő rendszer, amely az emberi test vagy egy merev test mozgásait felveszi, ilyen például az Optritrack. Az Optitrack rendszer az utóbb említett passzív markereket használja. Szakdolgozatom során ezzel a típusú markerrel dolgozom.

2.10. ábra Optitrack marker

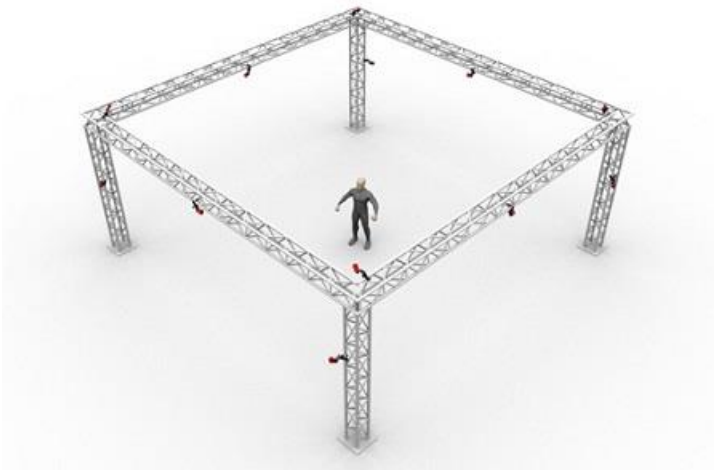
A markerek segítségével meghatározzuk a pontok térbeli helyzetét, majd ezeket az adathalmazokat használjuk később a modellezés során az összehasonlításra.
A jelölők elhelyezése, biztonságos rögzítése és megfelelő beállítása elengedhetetlen a rögzített információk megbízhatósága érdekében. A felrakott markerek pozíciója mellett a jelölések száma illetve specifikációi - mérete, visszaverődése is nagymértékben befolyásolja az eredményül kapott 3D koordinátákat.

A markerek használata során számos szabályt be kell tartanunk, hogy megbízható eredményeket kapjunk. Az egyik ilyen a testhezálló ruha használata, ugyanis el kell kerülnünk minden olyan lehetséges takarási lehetőséget, ami meggátolná, hogy a kamerák megfelelően lássák az adott markert. Azért is fontos a ruha, mert amikor az embernek egy pontját pontosan kell mérni, akkor jól kivehetően arra a helyre lehessen elhelyezni és ne mozduljon el mozgás közben. A markereket minimum 3 kamerának kell látnia, hiszen így tudjuk meghatározni a pont térbeli elhelyezkedését.

A jelölők elmozdulásait a minimális szintre kell leredukálnunk, hogy elkerüljük a későbbi zajt. A szimmetrikus elhelyezésre is figyelni kell egy teljes emberi test esetén, hogy később meg tudjuk különböztetni a jobb és bal oldalt [11].

Optitrack Flex 13 típusú 1.3 MP-es, 1280 x 1024-es felbontású, 120 FPS-re képes kamerákkal dolgozunk. A 8 kamera egy téglatest alakú vizsgálati helyszínt fed le számunkra [12].

Az optimális kameraelhelyezéseket vizsgálva több lehetőségünk is van. Egy körív mentén, egy négyzet vagy téglalap kerete mentén, a lényeg, hogy minél több szemszögből fedjük le az adott területet. Ha a kamerák hasonló helyzettel rendelkeznek, hasonló képeket rögzítenek az nem járul hozzá az esetlegesen eltakart területek lefedéséhez. Ez rontja a nagyobb kameraszám előnyeit is és megduplázza a kalibrációs folyamat számítási terhelését is [13].



2.11. ábra Optitrack kamera rendszer elhelyezkedése

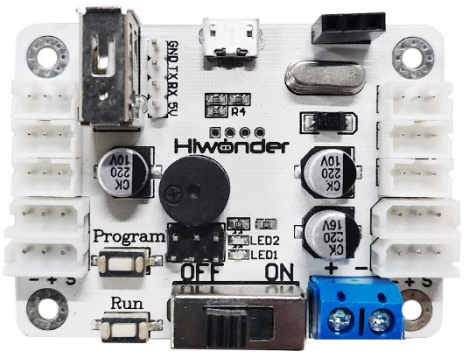
2.5.3. Motive szoftver

A Motive szoftver valós idejű leképezést ad a markerekkel felszerelt objektumunk vagy az emberi test helyzetéről. Visszajelzést kapunk arról, hogy megbízhatóan látják-e a kamerák a markereket. A szoftver segítségével rögzíthetünk egy adott mozdulatsort, képest csontvázkövetésre, illetve precíz a jelölőcímkézésben is. A kameraadatok feldolgozása során lehetőségünk van aktív vagy passzív követésre, a globális 3D pozíciók követésére és a forgási adatokat is biztosítják számunkra. A Motive folyamatos kalibrálást valósít meg, aminek segítségével csökkenti a szoftver kezelőjének a leterheltségét, és mindeközben javítja az adatok minőségét is. Egy kattintással létrehozhatóak különböző merev testek is. Az adatok streamelése és visszajátszása is nagyon hasznos és egyszerű folyamat a programban [14].

2.6. Fejlesztői panelek

2.6.1. Hiwonder Serial Bus Servo controller

A Hiwonder szervó vezérlő támogatja az online hibakeresést, programozást, a TTL soros portos kommunikációt és a PC grafikus hibakeresést. A vezérlő riaszt, ha alacsony feszültség lép fel. A CPU egy STM32-es chipet használ ARM Cortex-M3 core-ral. Tartozik hozzá egy grafikus program, amelyben akár 40 szervót is képesek vagyunk vezérelni, akciókat megadni nekik vagy folyamatokat létrehozni [15].

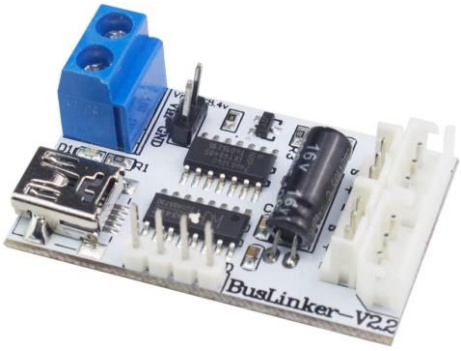


2.12. ábra Hiwonder Serial Bus Servo Controller

2.6.2. Hiwonder TTL/USB Debugger Board

A Hiwonder TTL/USB debugger board egy panel, amely képes csatlakoztatni a szervót a PC szoftverhez vagy más eszközökhöz és ennek segítségével be lehet állítani a szervók paramétereit. A szervó által megtett szöget és magát a pozíciót is képes leolvasni. A mérete 25 mm x 40 mm és rendelkezik USB porttal, serial porttal, servo porttal.

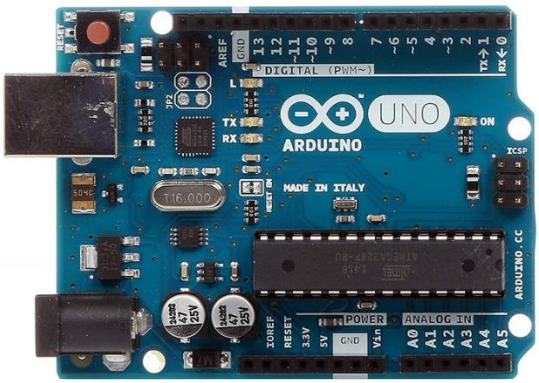
A panel kezeléséhez egy dedikált szoftver tartozik, ennek segítségével be tudunk állítani konkrét paramétereket, mint például a sebességet és a pozíciót [16].



2.13. ábra Hiwonder TTL/USB Debugger Board

2.6.3. Arduino Uno R3

Az egyik legnépszerűbb Arduino modell az Uno, Rev3-mas kiadása. Az Arduino alapvetően egy nyílt forráskódú platform, amely programozható eszközök fejlesztését teszi lehetővé. 14 digitális inputtal és outputtal rendelkezik. A panelen található egy LED is, amely állítható, illetve rendelkezik egy csatlakozóval, amely USB-B típusú. Ezen keresztül van lehetőség a számítógéppel való kommunikációra soros port segítségével.



2.14. ábra Arduino Uno

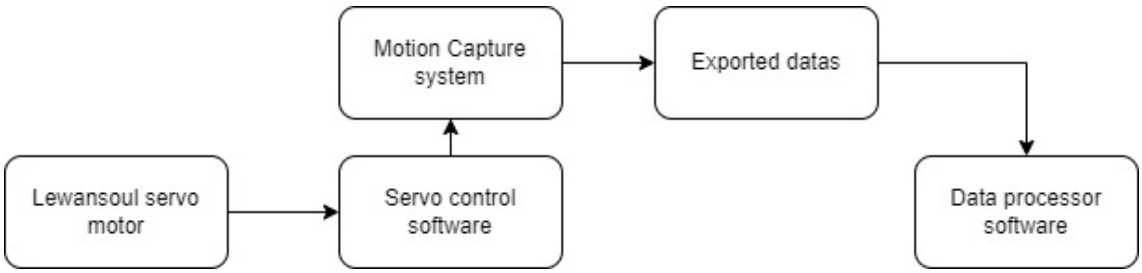
3. A TERVEZETT RENDSZER FELÉPÍTÉSE

3.1. Rendszerterv

A rendszer megvalósításához szükség van egy Motion capture rendszerre, ami jelen esetben Optitrack kamerákkal van felszerelve. Ezen felül szükségünk van a mikroszervóra, a szervót irányító boardra, tápegységre, egy 3D nyomtatással készült tartóra és egy a szervót rögzítő eszközre, ami egy satu lesz.

A rendszer hardverrészein felül szükség van még két szoftverre. Az egyik a szervót irányító szoftver, a másik pedig egy adatfeldolgozó szoftver.

Az alábbi ábrán a rendszerterv látható:



3.1. ábra Rendszerterv

3.2. A rendszer specifikációja

Az alapvető elvárásunk a rendszerrel szemben, hogy a különböző forrásokból származó adatokat összevetve meg tudjuk határozni a hiba mértékét. Az irányító szoftvernek képesnek kell lennie kommunikálni a szervóval a megfelelő kommunikációs protokollon keresztül. Ezen felül a paramétereknek állíthatónak kell lenniük. Illetve a befolyásoló tényezőket is át kellett gondolni, hogyan hat a különböző feszültség, amper, tömeg, sebesség, pozíció az adott méréssorozatra. Az adatfeldolgozó szoftvernek pedig fel kell tudnia dolgozni a Motion Capture rendszerből kiexportált adatokat, majd ezeken el kell végeznie adott műveleteket és a végén ezeket egy ábrán meg kell jelenítenie.

3.3. A tesztelés terve

A tesztelés során különböző teszteseteket jelöltünk ki a különböző mozgások teszteléséhez. Ezekhez a tesztesetekhez fontos volt, hogy a paramétereink állíthatóak legyenek. Meghatároztunk több pozíciós értéket, illetve több időértéket, ami alatt mozoghat a szervó. Teszteltük, hogy mi a legkisebb idő és a legnagyobb idő érték, ami alatt megteszi az utat. Több és kevesebb feszültségellátással is próbálkoztunk. A

különböző esetek közül az értékadás volt az első. Majd ezt összekötöttük egy értéklekérdezésssel, hogy ellenőrizni tudjuk a megkapott értéket. A következő a Lineáris mozgás volt, ahol egy pozíciót és egy fix időértéket adtunk meg. Itt az érték kiolvasások miatt beleraktunk egy várakozási időt is, így az idő alatt ki tudta olvasni a szervó helyzetét a motor. A Lineáris mozgás második tesztseteénél különböző időértékkel teszteltünk. Aztán következett a Gyorsuló mozgás, itt 3000 ms-ról 1500 ms-ra szerettük volna felgyorsítani eszközünket. Illetve ezt követte a Változó gyorsulású mozgás, amikor is random pozíciós értékekhez rendeltünk random időértéket.

A tesztelési terv során megkülönböztettük a tömeg nélküli és a 125 g-mos tömeggel való méréseket. Az utóbbihoz egy rudat használtunk, mert fontos volt, hogy szimmetrikus legyen a tömegünk, illetve hogy ne tudjon mozogni a motorunkon. A rúd egy 3D nyomtatással készült tartóba került bele, csavarokkal rögzítettük, így tudtuk vele a méréseket megtenni.

Az egész eszközt fixen kellett tartani, hogy csak a mozgó részt vehesse fel a Motion Capture rendszer és más mozgás ne legyen benne. Ennek megoldására egy satut használtunk és ezt egy asztal széléhez rögzítettük.

A teszteléshez szükségünk volt még a Motion Capture rendszerre. Ez az Óbudai Egyetem Mozgáslaborjában volt elhelyezve. A rendszer rendelkezett 8 kamerával, ami téglalap alakban helyezkedett el és minden oldalán 2-2 kamera foglalt helyet. Ezeknek a kalibrálásához két különböző eszközünk volt: egy kalibráló bot és egy háromszög.

3.4. Tesztelés a tracker munkaterének különböző pontjain

A mocap rendszer munkatere egy téglatest formájú terület, amit a 8 kamera lefed és ezen területen helyezzük el a méréshez használt mikroszervónkat. Alapvetően, ha 3 kamera lát egy markert, akkor lehetünk biztosak abban, hogy valós adatot kapunk vissza. Ha egy kamera nem lát rá az adott markerre, akkor azt jelzi. Motive szoftverben lévő háromdimenziós felület segítségével láthatjuk eszközünket, illetve hogy melyik kamera lát rá az adott markerre. Itt figyelhetjük meg azt is, ha nem teljesen kerül a kamera látószögébe, akkor az adott markerhez tartozó vonal pulzálással jelzi, hogy nem stabil a rálátása.

A munkatér szélein már egyre kevesebb kamera lát rá az adott pontra, így az adott markerre vonatkozó helyzet adatokat nem adja vissza a rendszer.

Olyat is tapasztaltunk a mérések során, hogy az előbb említett helyzetben lévő markerre, ha pár másodpercen keresztül nem lát rá a kamera majd újra igen, akkor összekeveri a markerek azonosítóját és egy másikról hiszi azt, hogy az az eredeti.

Ezen felül pedig „nan” vagyis „not a number” értéket kapunk vissza, ha a marker a kamera számára láthatatlan.

Több hibás működést is tapasztaltunk az alábbi esetekben:

- ha a markert takarja valami
- ha az eszközünkön megcsillan valami
- ha a munkaterületen becsillan valamilyen tárgy
- ha a munkaterület széléhez közelítünk
- ha mozgás közben elveszíti az egyik kamera az adott markert (nem lát rá az adott időpillanatban)
- ha egy zavaró tárgy van a kamera és a marker között
- ha nincs elég fény a helyiségben
- ha túl alacsonyan/magasan van az eszközünk

Ezen tesztmérések után döntöttünk úgy, hogy a mérési terület közepén helyezzük el egy asztalon az eszközünket, ezzel is biztosítva a legstabilabb információszerzést.

4. MICROSZERVO TESZTELÉSE ÉS IRÁNYÍTÁSA

4.1. Microszervo irányítása

4.1.1. Lewansoul LX-16A

A Mollia által tervezett humanoid robot első verziójában a Lewansoul Lx-16A szervók kerülnek beépítésre, így elengedhetetlen, a megfelelő fejlesztési irányok meghatározásához, hogy ezeket megfelelően tudjuk tesztelni, illetve irányítani.

4.1.2. Kommunikációs protokoll

Alapvetően soros kommunikáció és 9600-as átviteli sebesség jellemzi ezt az eszközt. A kommunikációs protokoll hexadecimális értékeket vár el és 4 fő részből áll. Az első a fejléc ebben két egymást követő 0x55 –öt küld, ami azt jelzi, hogy az adatcsomagok megérkeztek. Ezután jön az adat hossza, ami az alábbi módon jön létre: paraméterszám plusz egy parancs plusz az adathossz által elfoglalt bájthossz vagyis Hossz = n + 2. A harmadik rész a parancs, a különféle vezérlési utasítások, majd ezt követi a paraméter 1-től n-ig, ami vezérlési információ hozzáadását teszi lehetővé. Ha a felhasználó a megfelelő adatokat továbbítja a szervovezérlőnek a vezérlőn lévő kék led egyszer felvillan jelezve, hogy megfelelő adat érkezett. Ha a felhasználó rossz adatot ad át, akkor a kék led nem reagál, folyamatosan világít.

Header	Data Length	Command	Parameter
0x55 0x55	Length	Cmd	Prm 1.. Prm N

2. táblázat Lewansoul LX-16A szervó kommunikációs protokollja

A táblázaton a kommunikációs protokoll látható, ami a szervó és az irányító kód közti kapcsolatot biztosítja.

Főbb parancsokhoz tartozó kommunikáció felépítése [17]:

1. CMD_SERVO_MOVE (parancs értéke: 3)
   Ezzel a paranccsal mozgathatjuk akármelyik szervót. A hossznak egyenlőnek kell lennie a vezérlő szervók számának háromszorosával plusz öttel.
   Paraméter 1: A vezérlendő szervók száma
   Paraméter 2: Az időérték alacsonyrendű 8 bitje
   Paraméter 3: Az időérték magasrendű 8 bitje
   Paraméter 4: A szervó ID-ja

Paraméter 5: A szögpozíció alacsonyrendű 8 bitje
Paraméter 6: A szögpozíció magasrendű 8 bitje
További paraméterek megegyeznek a 4, 5, 6-os paraméterrel

Példa: Az 1-es szervót vezéreljük, úgy hogy 1500 ms-on belül (1,5 másodpercen belül) 200 pozícióba forduljon.
Fejléc: 0x55 0x55
Hossz: 0x08 – mert az első szervót szeretnénk vezérelni vagyis: 1*3+5 = 8
Parancs értéke: 0x03 – 3-mas érték hexadecimálisan megadva
Paraméterek: 0x01 – a szervók száma amit irányítani szeretnénk , 0xdc – az idő alacsonyrendű bitjei, 0x05 – az idő magasrendű bitjei, 0x01 – az adott szervó ID-ja, 0xc8 – a szögpozíció(rate-ben) alacsonyrendű bitjei, 0x00 – a szögpozíció magasrendű bitjei

2. CMD_ACTION_GROUP_RUN (parancs értéke: 6)
   Ezzel a paranccsal a műveletcsoportok futtatásának vezérlését tudjuk megtenni. Beállíthatjuk, hogy hányszor fusson le, illetve lehetőség van folyamatos futásra is ehhez az értéket 0-ra kell állítani. Az adat hossza 5.
   Paraméter 1: A futtatandó műveletcsoportok száma
   Paraméter 2: A műveletcsoport futási idejének alacsonyrendű 8 bitje
   Paraméter 3: A műveletcsoport futási idejének magasrendű 8 bitje

   Példa: Ha a 8-as csoportot akarjuk egyszer futtatni, akkor az alábbi kódot adjuk meg.
   Fejléc: 0x55 0x55
   Hossz: 0x05 – 5 hosszúságú
   Parancs értéke: 0x06 – 6-os parancs hexadecimálisan megadva
   Paraméter: 0x08 – műveletcsoport száma jelen esetben 8-as, 0x01 – futási idő alacsonyrendű bitjei, 0x00 – futási idő magasrendű bitjei

3. CMD_ACTION_STOP (parancs értéke: 7)
   Ez a futó műveletcsoport leállítására szolgál. Ha a műveletcsoport nem fut, a parancs nem befolyásol semmit. Az adat hossza 2.

   Példa: Állítsuk le a futó műveletcsoportot
   Fejléc: 0x55 0x55
   Hossz: 0x02 – 2 hosszúságú lesz az adat
   Parancs értéke: 0x07 – 7 lesz
   Paraméter itt nincs

4. CMD_ACTION_SPEED (parancs értéke: 11)

A vezérlő akciócsoportjainak sebességét lehet ezzel beállítani. Ha az 1-es számú munkacsoportnak szeretnénk megadni, hogy az eredeti sebesség kétszerese legyen, akkor a szám 200 lesz, mert százalékos formában kell megadnunk tehát ez 200% lesz. Az összes műveletcsoport sebességének beállítását a 0xFF kóddal tudjuk megtenni. A szervónak is megvan a saját határsebessége, így erre nagyon kell figyelni, hogy a megadott fordulatszám ne lépje túl a maximális fordulatszámot. Az adat hossza itt 5.
Paraméter 1: A módosítandó műveletcsoport száma
Paraméter 2: A sebesség százalékos értékének alacsonyrendű 8 bitje
Paraméter 3: A sebesség százalékos értékének magasrendű 8 bitje

Példa: A 8-as számú akciócsoport 50%-os sebességgel működjön
Fejléc: 0x55 0x55
Hossz: 0x05
Parancs értéke: 0x0B
Paraméter: 0x08 – műveletcsoport száma, 0x32 – alacsonyrendű 8 bit, 0x00 – magasrendű 8 bit

5. CMD_GET_BATTERY_VOLTAGE (parancs értéke: 11)
   Le tudjuk kérdezni a szervó vezérlő feszültségét milivoltban. A vezérlő azonnal visszaadja az adatokat a parancs elküldése után. A visszaadott adat egy adatcsomag két paraméter értékkel. Az adat hossza 2. Itt viszont nincs paraméter, amit meg kell adnunk. Mivel a két parancs értéke ugyanúgy 11-11, ezért úgy tudja megkülönböztetni a program, hogy nézi a paramétereket, mert ha rendelkezik 3 paraméterrel a parancs, akkor a korábban már említett sebesség beállítás fut le, ha viszont nem rendelkezik parancsokkal, akkor a feszültséglekérés fut le.

   Példa: Feszültség lekérdezése
   Fejléc: 0x55 0x55
   Hossz: 0x02 – az adat hossza 2
   Parancs értéke: 0x0F – 11 lesz
   Paraméter nincs

   Visszakapott csomag:
   Fejléc: 0x55 0x55
   Hossz: 0x04 – az adat hossza 4
   Parancs értéke: 0x0F – 11 lesz
   Paraméter: 0x4C -7500mV alacsonyrendű 8 bitje, 0x1D – 7500mV magasrendű 8 bitje

6. CMD_MULT_SERVO_UNLOAD (parancs értéke: 20)

Több szervó együttes kikapcsolása, hogy utána a megfelelő szervót tetszés szerint tudjuk forgatni kézzel. Az adathossz az a vezérelt szervók mennyisége plusz 3.
Paraméter 1: a vezérelt szervók mennyisége
Paraméter 2: az első szervó ID-ja
Paraméter 3: a második szervó ID-ja
Paraméter.. : a sorban következő szervó ID-ja

Példa: Kapcsolja ki a 1,2,3 szervókat és motorjaikat
Fejléc: 0x55 0x55
Hossz: 0x06 – 3 vezérelt szervó plusz 3, tehát a hossz 6 lesz
Parancs értéke: 0x14 – vagyis 20
Paraméter: 0x03 – a szervók száma, 0x01 – az első szervó ID-ja, 0x02 – a második szervó ID-ja, 0x03 – a harmadik szervó ID-ja

7. CMD_MULT_SERVO_POS_READ (parancs értéke: 21)
   Kiolvassa több szervó pozíciójának értékét. Az adathossz az olvasni kívánt szervók mennyisége plusz 3.
   Paraméter 1: az olvasni kívánt szervók mennyisége
   Paraméter 2: az első szervó ID-ja
   Paraméter 3: A második szervó ID-ja
   Paraméter… : A soron következő olvasni kívánt szervó ID-ja

   Példa: Olvassa ki a 1,2,3,4,5,6 szervók pozíciójának értékét.
   Fejléc: 0x55 0x55
   Hossz: 0x09 – a vezérelt szervók száma 6 plusz 3 tehát az összesen 9 lesz.
   Parancs értéke: 0x15 – vagyis 21
   Paraméter: 0x06 – a szervók száma, 0x01 – az első olvasni kívánt szervó ID-ja, 0x02 – a második szervó ID-ja, 0x03 – a harmadik szervó ID-ja, 0x04 – a negyedik szervó ID-ja, 0x05 – az ötödik szervó ID-ja, 0x06 – a hatodik szervó ID-ja

   Visszakapott csomag:
   Adathossz: A kiolvasott szervók száma
   Paraméter 1: A kiolvasott szervók száma
   Paraméter 2: A szervó ID-ja
   Paraméter 3: a pozíció értékének alacsonyrendű 8 bitje
   Paraméter 4: a pozíció értékének magasrendű 8 bitje
   Pataméret … : A formátum megegyezik a 2-es, 3-mas, 4-es paraméterrel

   Példa: Minden pozícióra visszakapott értékünk 500 lesz
   Fejléc: 0x55 0x55
   Hossz: 0x15 – az adat hossza 21

Parancs értéke: 0x15 – vagyis 21
Paraméter: 0x06 – 6 szervó pozícióját olvassa ki, 0x01- az első szervó ID-ja, 0x01 – a pozíció vagyis az 500 alacsonyrendű 8 bitje, 0xF4 – a pozíció vagyis az 500 magasrendű bitjei, 0x02 – a második szervó ID-ja, 0x01 - az 500 alacsonyrendű 8 bitje, 0xF4 - az 500 magasrendű 8 bitje, 0x03 – a harmadik szervó ID-ja, 0x01 - az 500 alacsonyrendű 8 bitje, 0xF4 - az 500 magasrendű 8 bitje, 0x04 – a negyedik szervó ID-ja, 0x01 - az 500 alacsonyrendű 8 bitje, 0xF4 - az 500 magasrendű 8 bitje, 0x05 – az ötödik szervó ID-ja, 0x01 - az 500 alacsonyrendű 8 bitje, 0xF4 - az 500 magasrendű 8 bitje, 0x06 – a hatodik szervó ID-ja, 0x01 - az 500 alacsonyrendű 8 bitje, 0xF4 - az 500 magasrendű 8 bitje. [18] [19]

```
def send(self,packet):
    self.serial.write(packet)

def sendReceivePacket(self,packet,receiveSize):
    t_id = packet[0];t_command = packet[2]
    self.serial.flushInput();self.serial.timeout=0.1;self.sendPacket(packet)
    r_packet = self.serial.read(receiveSize+3); return r_packet

def motorOrServo(self,id,motorMode,MotorSpeed):
    packet = struct.pack("<BBBBBh",id,7,29,motorMode,0,MotorSpeed)
    self.sendPacket(packet)# motorMode 1=motor MotorSpeed=rate, 2=servo

def moveServo(self,id,position,rate=1000): #rate=1000
    packet = struct.pack("<BBBHH",id,7,1,position,rate)
    self.sendPacket(packet)  # Move servo 0-1000, rate(ms) 0-30000(slow)

def readPosition(self,id):
    packet = struct.pack("<BBB",id,3,28)
    rpacket = self.sendReceivePacket(packet,5)
    s = struct.unpack("<BBBBBhB",rpacket);return s[5]

def LoadUnload(self,id,mode):
    packet = struct.pack("<BBBB",id,4,31,mode)
    self.sendPacket(packet)#Activate motor 0=OFF 1 =Active
```

4.1. ábra Irányító szoftver

Alapvető funkciókat szerettünk volna tesztelni a Lewansoul LX-16A szervón, ilyen például az értékadás és az értéklekérdezés. (get, set). Ezután meg kellett néznünk, hogy egy adott pozícióba milyen pontossággal ment el. Ezt a moveServo paranccsal tudtuk megadni, majd kiolvasni a tényleges adatot a readPosition paranccsal tudjuk megtenni. A szervó 0° és 240° között tud mozogni, ezen a tartományon 0 és 1000 közötti pozíciós értékeket vesz fel. A kezdeti 0°-ban 0 értéket vesz fel, a végpontban 240°-nál pedig 1000 értéket, 150 és 850 pontoknál vízszintben van, 500-nál pedig függőlegesen.

Ezeket az előre definiált mozgásokat felvesszük a mocap segítségével, majd kiexportáljuk egy csv fájlba a markerek helyzetének adott időpillanatban mért értékeit.

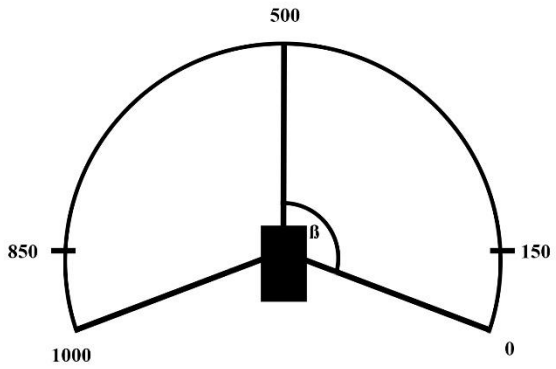
4.2. Microszervo tesztelése

4.2.1. Lewansoul LX-16A

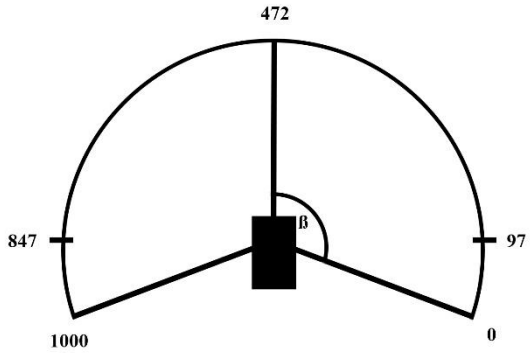
A szervóhoz különböző teszteseteket írtunk elő annak függvényében, hogy mi volt az az alapvető funkciólista, amit meg akartunk valósítani. A legfőbb esetek a következőek:

1. Értékadás és értéklekérdezés
2. Lineáris mozgás I.
3. Lineáris mozgás II.
4. Gyorsuló mozgás
5. Változó gyorsulású mozgás

A szervó mozgási tartománya 240°, amit 0-1000 közötti rátában tudunk kezelni. Ennek a főbb pontjait jelöltük ki: 100, 300, 472, 600, 700, 1000-nél. A függőleges értéknek 500-nak kellett volna, hogy legyen, már a kezdeti szakaszban az értékadásnál láttuk, hogy ott eltérés van. 120°-ban volt a függőleges helyzetünk – ez 500- at jelentett viszont, szögmérő és a vizuális kép alapján megállapítottuk, hogy ebben a helyzetben nem függőleges. A legközelebbi érték a 472 volt a függőlegeshez, így a mérések során ezt használtuk. Az alábbi képen az elméleti felosztást láthatjuk.

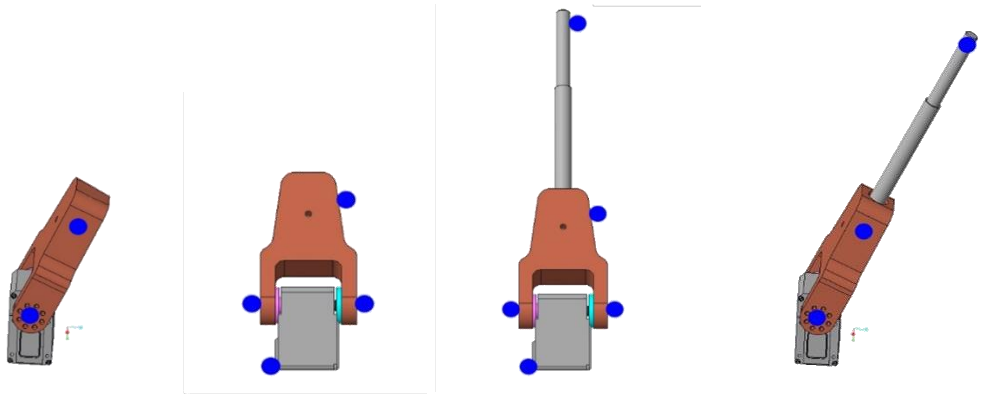


4.2. ábra Elméleti felosztás 0-1000 között



4.3. ábra Általunk valósnak ítélt ráta értékek

A szervóra kezdetben – amikor még tömeg nélkül mértünk – 4 markert helyeztünk el, majd később, amikor már tömeggel mértünk, akkor már 5 markerrel dolgoztunk. Ez azért fontos, mert minden egyes marker egy pontot jelöl a térben és később ezekkel tudunk pontos számításokat végezni a mért eredmények és az elvárt értékek összevetésénél.
Az első két ábrán a markerek elhelyezkedése látszik a tömeg nélüli méréseknél. A következőd két képen pedig a markerek pozíciója látszódik a tömeggel való méréskor.



4.4. ábra Markerek elhelyezkedése

Eredmények:

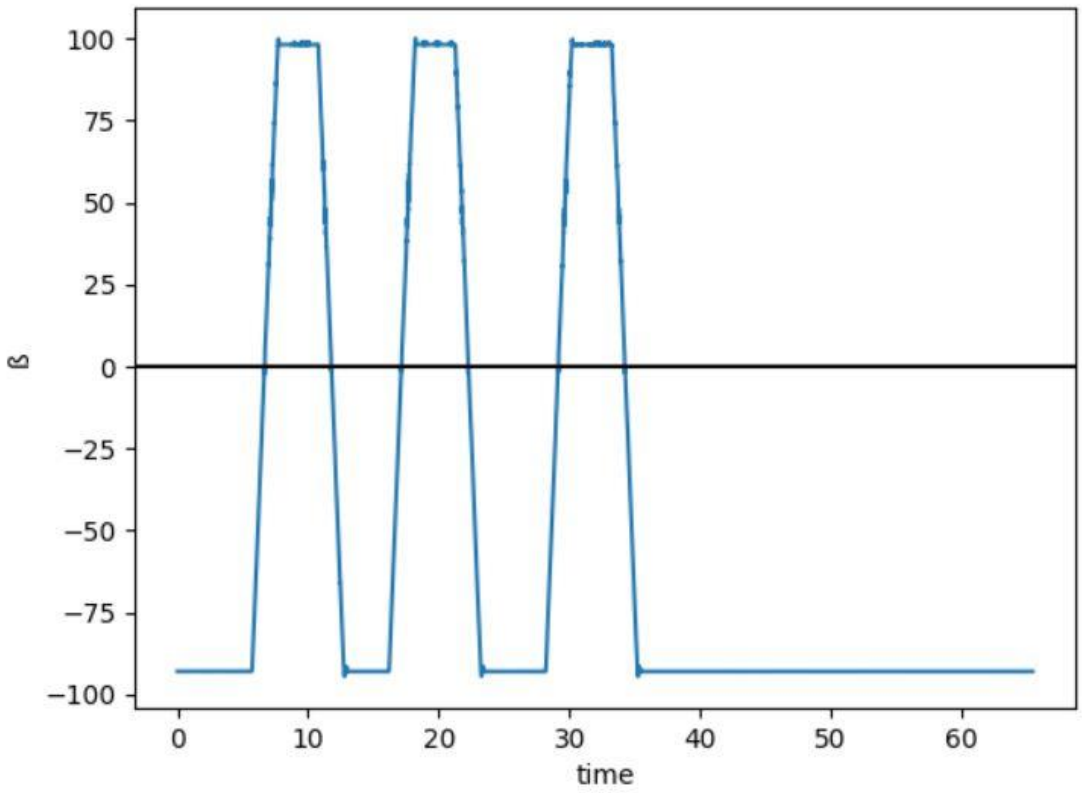
Értékadás, értéklekérdezés tömeg nélküli mérés esetén

Az értékadásokkor meg kell adnunk, hogy milyen sebességgel, milyen pozícióba kell mozognia a szervónak. 0 és 240° között tudjuk mozgatni, ez 0 és 1000 közötti értékre van

felbontva, így egyszerűen leosztjuk 4,167-el, így tudunk neki fokból egy elmozdulási értéket számolni. A mérés során az értékadást egy érték lekérdezéssel ellenőriztük.

Előírt szög	**Visszakapott szög**	**Hibatartomány**	**Időérték**
24°	23,8°; 24,2°	-0,2° és 0,2° között	2000 ms
72°	72,4°; 72,47°; 72°	0° és +0,47° között	2000 ms
113,3°	112,8°; 113,5°; 113,3°	-0,3° és +0,4° között	2000 ms
144°	143,5°; 143,3°	-0,7° és + 0,5° között	2000 ms
168°	169°; 168,2°	+0,2° és 1° között	2000 ms
216°	216°; 215,7°	-0,3° és 0° között	2000 ms

3. táblázat értékadás, értéklekérdezés tömeg nélküli mérés esetén



4.5. ábra Értékadás 900

A méréshez tartozó kód az alábbi:

```
def getset(self, b):
    for a in range(0,2):
     m1.moveServo(1,b,2000) #id, position #speed
     sleep(5)
     print(m1.readPosition(1))
     m1.moveServo(1,100,2000) #fix point
```

4.6. ábra értékadás és értéklekérdezés függvénye

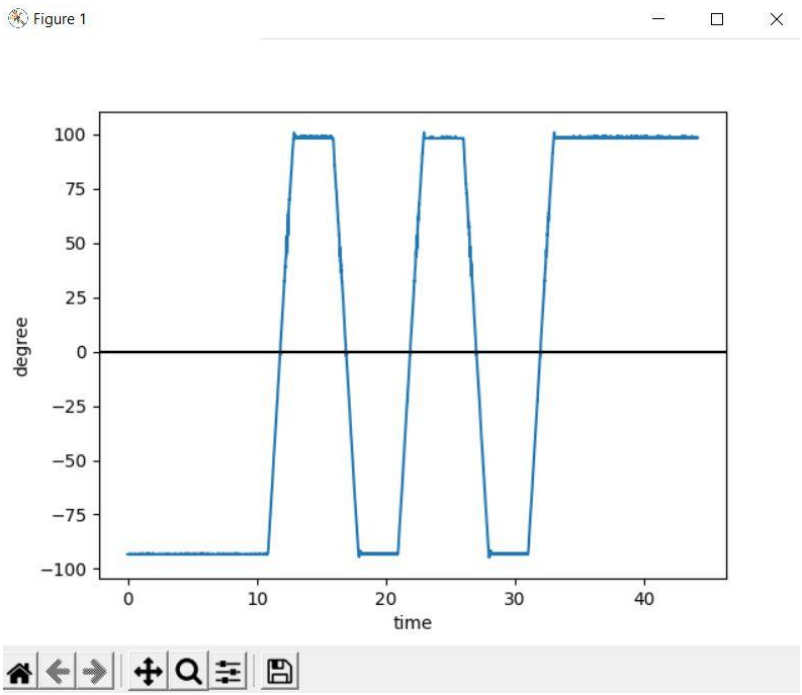
A függvényünk egy paraméterrel rendelkezik. Az értékadásnál 2 másodpercet adtunk meg mindenhol fixen a szervó mozgási idejének, ezt később változtatjuk majd, illetve mindig a 100 értékből indul a mérés és a b változó - ami a függvény egy paramétere - megkapja a mérni kívánt értéket. Ez a ciklus háromszor fut le, így 3 értéket olvasunk ki összesen. A táblázatban, ha egy értéket kétszer kaptunk vissza, nem kerül külön feltüntetésre. Ami még a kódban látható, hogy belekerült egy sleep(5) vagyis egy 5 másodperces szünet, ez az érték kiolvasása miatt volt szükségszerű. Ezt van ahol

módosítottuk 3 másodpercre. Mindig egy fix pontból szerettük volna indítani a méréseket, ezért lett mindenhol ez 100.

Lineáris mozgás I. tömeg nélküli mérés esetén

Előírt szög	Visszakapott szög	Hibatartomány	Időérték
24°-72°	72°; 23,8°, 71,8°; 23,8°, 72°; 23,8°	-0,2° és 0,2° között	2000 ms
24°-113,3°	113,5°; 24,2, 112,8°; 24°, 112,8°; 23,8°	-0,4° és +0,3° között	2000 ms
24°-144°	143,5°;23,8°; 143,3°;23,8°; 143,3°;23,8°	-0,7° és -0,2° között	2000 ms
24°-216°	23,8°;216,5°; 23,8°;216°; 23,8°	-0,2° és +0,5° között	2000 ms

4. táblázat lineáris mozgás tömeg nélküli mérés során



4.7. ábra Lineáris mozgás 100-900 között

A méréshez tartozó kód az alábbi:

```
def linear_move(self,a,b):
    for a in range(0,2):
     m1.moveServo(1,b,2000)
     sleep(5)
     print(m1.readPosition(1))
     m1.moveServo(1,a,2000)
     sleep(5)
     print(m1.readPosition(1))
```

4.8. ábra Lineáris mozgatás függvénye

A függvényünk két paraméterrel rendelkezik ez az ’a’ és a ’b’. Ezek jelentik a ráta kezdeti és végponti értékét. A szervó sebessége 2000 ms lesz, és itt is beépítésre került egy 5 másodperces várakozás az értéklekérdezés miatt. A ciklus háromszor fut le és a lineáris mozgás az ’a’ és ’b’ érték között zajlik.
A táblázatból jól kivehető, hogy a 100-900 közötti mérés esetén az utolsó körben, nem ment vissza az ’a’ értékbe a szervó, így ott csak 5 mért értékünk van. Ezt jól szemlélteti a 4.10 ábránk, ahol is látszik, hogy az utolsó kört csak félig tette meg.

Lineáris mozgás II. tömeg nélküli mérés esetén

Előírt szög	Visszakapott szög	Hibatartomány	Időérték és a várakozási idő
24°- 72°	71,8°;23,8°; 71,8°;23,8°; 71,8°;23,8°	-0,2°	1500 ms és 8000 ms
24°- 72°	72°;23,8°; 72,2°;23,8°; 72°;23,8°	-0,2° és +0,2° között	4000 ms és 8000 ms
24°-113,3°	112,8°;23,8°;112,8°;23,8°;112,8°; 23,8°	-0,4° és -0,2° között	1500 ms és 8000 ms
24°-113,3°	113,5°;23,8°;113,5°;23,5°; 113,5°;23,8°	-0,5° és +0,3° között	4000 ms és 8000 ms
24°-144°	143,7°;23,8°;143,7°;23,8°;143,7°; 23,8°;	-0,3° és -0,2° között	1500 ms és 8000 ms
24°-144°	144,2°;23,8°;144,2°;23,8°;144,2°; 23,8°;	-0,2° és +0,2° között	4000 ms és 8000 ms
24°-216°	215,7°;23,8°; 215,7°;23,8°;215,7°;23,8°;	-0,3° és -0,2° között	1500 ms és 8000 ms
24°-216°	216,5°;24,2°;216,2°;23,8°;216,5°; 23,8°	-0,2° és +0,5° között	4000 ms és 8000 ms

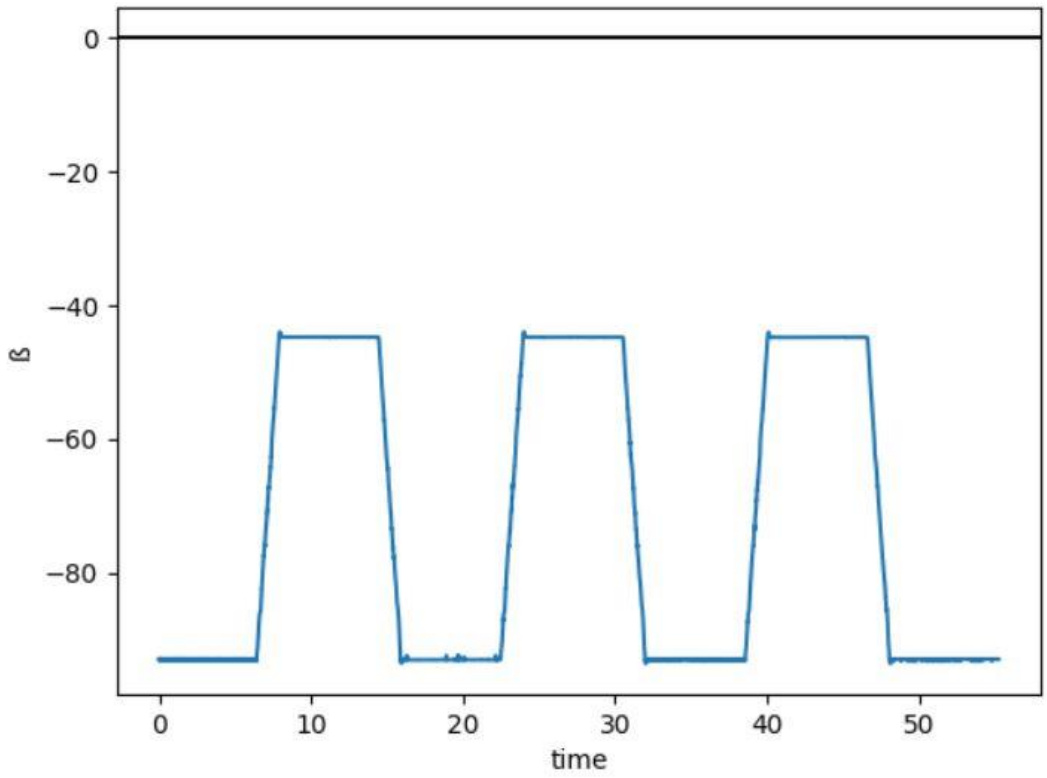
5. táblázat Lineáris mozgás II. tömeg nélkül

A méréshez tartozó kód az alábbi:

```
def linear_move_two(self,a,b,time):
    m1.moveServo(1,b,time)
    sleep(8)
    print(m1.readPosition(1))
    m1.moveServo(1,a,time)
    sleep(8)
    print(m1.readPosition(1))
```

4.9. ábra Lineáris mozgás II. függvénye

A függvényünk három paraméterrel rendelkezik: egy kezdeti ráta értékkel, egy végponti ráta értékkel és egy idő értékkel. Másfél másodperc és négy másodperces időkkel néztük meg a különböző lineáris eseteket. A várakozási idő pedig 8 másodperc volt. A mérések során azt láthattuk, hogy a hosszabb távokat pl.: 100-900 között egyenletesebben tette meg, mint a rövidebb távokat pl.: 100-300 között. Utóbbi esetben sokszor darabosan mozgott, ez látható a 4.13 ábrán.



4.10. ábra Lineáris mozgás 100-300 között tömeg nélkül

Gyorsuló mozgás tömeg nélküli mérés esetén

Előírt szög	Megfigyelés	Várakozási idő két ciklus között	Időérték
24°-216°	Egy teljes kört megy, tehát 100-900-ig és 900-100-ig	5 másodperc	3000 ms -1500 ms-ig csökken az érték
24°-216°	Két kört megy	5 másodperc	3000 ms-1500 ms-ig csökken az érték

24°-216°	Leggyorsabb idő, ami alatt átviszi	5 másodperc	500 ms, ami fél másodpercnek felel meg
24°-216°	Leglassabb idő, ami alatt átviszi	40 másodperc	35000 ms, ami 35 másodpercnek felel meg

6. táblázat Gyorsuló mozgás tömeg nélkül

A méréshez tartozó kód az alábbi:

```
def accelerate_move(self, pos, time):
    for a in range(0,3):
     pos = pos + 100
     time = time - 500
     m1.moveServo(1, pos, time)
     pos = pos + 100
     m1.moveServo(1, pos, time)
    sleep(5)

    pos = 900
    time = 3500

    for a in range(0,3):
     pos = pos - 100
     time = time - 500
     m1.moveServo(1, pos, time)
     pos = pos - 100
     m1.moveServo(1, pos, time)
    sleep(5)
```

4.11. ábra Gyorsuló mozgáshoz tartozó függvény

A függvény két paraméterrel rendelkezik, az első a kezdeti pozíció, a második az időérték. A függvény folyamatosan csökkenti az időértéket, ezzel elérve a gyorsulást. A pozíciós tartomány a mérés során 100-900 között volt, ezt 100-ról növeltük folyamatosan 100-asával. Egy oda és egy vissza kört tett meg a szervó. A két kör között egy 5 másodperces várakozási időt tartott.

Itt került letesztelésre, hogy melyik az a legkisebb, illetve legnagyobb időérték, ami alatt átmegy a szervó. A leggyorsabb időre fél másodperc jött ki, a leglassabbra pedig 35 másodperc.

Változóan gyorsuló mozgás tömeg nélküli mérés esetén

Itt alapvetően random értékekkel dolgoztunk, ezek kerültek hozzáadásra a pozícióhoz és az időértékhez. A következő pozíció-idő értéket kaptuk meg: 233 – 2232 ms , 998-2192 ms, 114-2102 ms, 2-2232 ms. A pozíció random számát 0 és 1000 között állapítottuk meg, az idő random értékét pedig 100 és 300 között és ezt 2000 ms-hoz adtuk hozzá.

Értékadás, értéklekérdezés 125 g-mos tömeggel való mérés során

Előírt szög	Visszakapott szög	Hibatartomány	Időérték és várakozási idő
113,3°	112,6°; 112,8°; 113,3°	-0,7° és 0° között	2000 ms és 5000 ms
72°	71,5°;71,3°	-0,7° és -0,5° között	2000 ms és 5000 ms
168°	168,9°;169,2°	+0,9° és 1,2° között	2000 ms és 5000 ms
216°	216,5°;216,7°	+0,5° és +0,7° között	2000 ms és 5000 ms

7. táblázat Értékadás és értéklekérdezés tömeggel való mérés során

A mérés során 100-472 közötti esetben jól látszott, hogy a tömeg kilengett, illetve amikor 900-ba kellett mennie, vagyis 216°-ba – ez lett volna a vízszintes – akkor lejjebb ment. Itt is 3 kör ment mindből, ezért kapunk 3 értéket vissza. (az ugyanolyan értékek nem kerülnek kiírásra)

Lineáris mozgás I. 125 g-mos tömeggel való mérés során

Előírt szög	Visszakapott szög	Hibatartomány	Időérték és várakozási idő
24°-72°	71,8°;23,5°; 72°;23,8°; 72°;23,3°	-0,7° és 0° között	2000 ms és 8000 ms
24°-113,3°	113,3°;23,3°; 113,3°;23,3°; 113,3°;23,3°;	-0,7° és 0° között	2000 ms és 5000 ms

24°-144°	144°;23,3°; 144°;23,8°; 144,2°;23,8°	-0,7° és +0,2° között	2000 ms és 5000 ms
24°-216°	216,5°;22,8°; 216,5°;23,3°; 216,5°;22,6°	-1,4° és +0,5° között	2000 ms és 5000 ms

8. táblázat Lineáris mozgás tömeggel való mérés során

Ezen lineáris mérések során háromszer is előfordult, hogy kiakadt a szervó és nem ment tovább az adott körben, illetve nem tudta megtartani a vízszintes helyzetet 900-nál. A beállítások során 3 kört kértünk a szervótól, 2 másodperc alatt és a várakozási idő 5 másodperc és 8 másodperc volt.

Lineáris mozgás II. 125 g-mos tömeggel való mérés során

Előírt szög	**Visszakapott szög**	**Hibatartomány**	**Időérték és várakozási idő**
24°-72°	71,3°;22,8°; 71°;23°; 71,8°;23°	-1,2° és -0,2° között	1500 ms és 8000 ms
24°-72°	71,5°;24°; 71,3°;23°; 71,3°;22,8°	-1,2° és 0° között	4000 ms és 8000 ms
24°-72°	71°;22,8°; 71,8°;22,6°; 71,8°;22,8°	-1,4° és -0,2° között	1500 ms és 5000 ms
24°-72°	71°;-1°	-25° és -1° között	35000 ms és 40000 ms
24°-113,3°	113°	-0,3°	1500 ms és 8000 ms
24°-113,3°	113°;22,8°; 112,6°;23°; 113°;22,6°	-1,4° és -0,3° között	1500 ms és 8000
24°-113,3°	112,8°;23,5°; 112,8°;23,3°; 113°;23°	-1° és -0,3° között	4000 ms és 8000 ms

24°-113,3°	113,5°;23°	-1° és +0,2° között	35000 ms és 40000 ms
24°-144°	144°;23,3°; 143,7°;23,3°; 144°;23,3°	-0,7° és 0° között	1500 ms és 8000 ms
24°-144°	144,7°;23,3°; 144,7°;23°; 144,7°;23,5°	-1° és +0,7 ° között	4000 ms és 8000 ms
24°-144°	144,2°	+0,2°	35000 ms és 40000 ms
24°-144°	144,5°;24°	0° és +0,5° között	35000 ms és 40000 ms
24°-216°	216,7°	0,7°	1500 ms és 8000 ms
24°-216°	216,7°;23,5°; 216,5°;23,5°	-0,5° és +0,7° között	1500 ms és 8000 ms
24°-216°	216,7°;23,3°; 216,7°;23°; 216,7°;23,3°	-1° és +0,7° között	4000 ms és 8000 ms
24°-216°	216,7°	+0,7°	35000 ms és 40000 ms

9. táblázat Lineáris mozgás II. tömeggel való mérés esetén

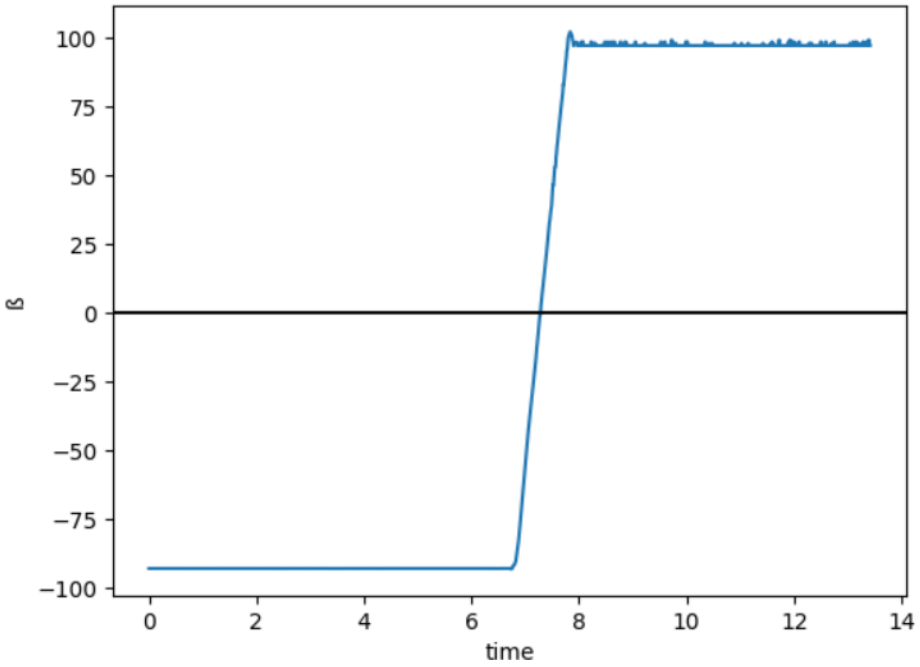
A tömeg nélküli és a 125g-mos tömeggel való lineáris mérések eredményeiből a két táblázatra nézve is rögtön látszik, hogy az utóbbinál nagyobb a hibatartomány, illetve a mérések során is többször tapasztaltunk olyat, amikor nem volt képes vízszintesen tartani a szervót. A szakaszos mozgás jellemzi legtöbbször, a függőleges pozíciónál kileng a tömeg – ez a 100-472 tartomány mérése során jött elő, mert ott stabilan kéne tartania a tömeget, de ez nem sikerül neki.

A Lineáris mozgás II. tesztelések során az elvárás a 3 kör futása lett volna, azonban többször tapasztaltunk olyat, amikor 1 kör után megállt és nem ment tovább. 35 másodperces mérésnél pedig csak 1 kört futtattunk, amit szakaszosan ugyan, de meg tudott tenni még 35 másodperc alatt. Ennél magasabb időtartamnál már nem viszi át a szervót vagy olyan hiba is előfordult, amikor megállt egy bizonyos pontban és várt, aztán továbbment. 4 másodperces időtartamnál 100-600 között fél percig várt majd elindult és megcsinálta a 3 kört. A 35 másodperces 100-300 közötti futás során került előtérbe a

legnagyobb hibaérték, amikor is meg kellett volna tartania a vízszintes pozícióban a tömeget, de ezt nem tette meg, hanem lement -1°-ba.

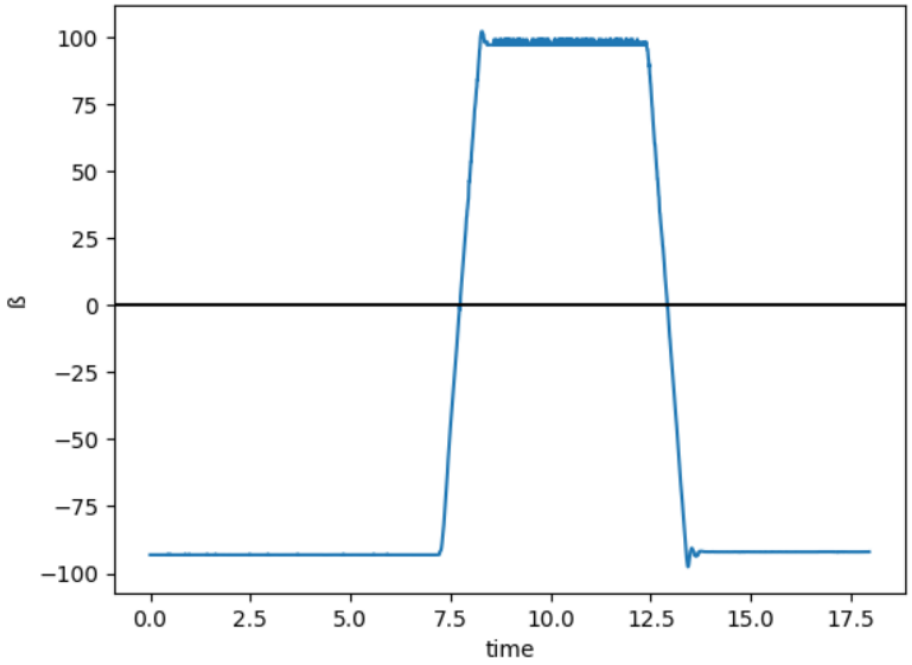
Gyorsuló mozgás 125g-mos tömeggel való mérés esetén

A korábban használt gyorsuló mozgásra írt függvényt használtuk itt is, kezdetben 3500 ms-ról vagyis 3,5 másodpercről indul, majd ezt folyamatosan csökkentjük 500-al, így kapum 3 másodpercet, 2,5 másodpercet, 2, másodpercet majd a végén 1,5 másodpercet. A pozíció pedig 100-asával növekszik, illetve a második körben csökken. A megfigyeléseink a következőek voltak 900-nál nem tudta megtartani vízszintesen a tömeget és másfél kört ment le, aztán kiakadt.



4.12. ábra Gyorsuló mozgás során előjött hibás működés

A második mérésnél azonban megcsinálta jól, ez az alábbi ábrán látható.



4.13. ábra Gyorsuló mozgás helyes működése

A leggyorsabb illetve a leglassabb értéket korábban említettem, itt is azokkal az értékekkel működtek: 500 (1,5 másodperc) és 35000 (35 másodperc).

Változóan gyorsuló mozgás 125g-mos tömeggel való mérés esetén

Az első mérés értékelhetetlen lett, mert a harmadik kör során nem értékelte ki a program az első random számot, így nem tudta azt átadni a szervónak. A következő mérés azonban jól sikerült, az első körben a pozíció-idő páros az alábbi volt: 678, 2182. A második körben: 826, 2267. A harmadik körben pedig: 379, 2122.

5. ADATFELDOLGOZÓ SZOFTVER

A Motion Capture rendszer által mért adatok feldolgozásához szükségünk volt egy adatfeldolgozó szoftverre. Az adatok a Motive szoftverből kiexportálásra kerültek a megfelelő elnevezéssel, így egy csv fájllal kellett dolgoznunk. Ebben szerepelnek az adott időpillanathoz tartozó markerek térbeli helyzetének x, y, z koordinátái. A tömeg nélküli mérések során 4 markert helyeztünk el az eszközünkre.

A szoftver Python nyelvben került megírásra. A Python 3 verziót használtuk. Azért esett erre a programozási nyelvre a választásunk, mert sok hasznos library található hozzá, például a matematikai függvények könnyű implementálása terén, ami egy lényeges szempont volt számunkra.

Az elvárásaink a szoftverrel szemben:

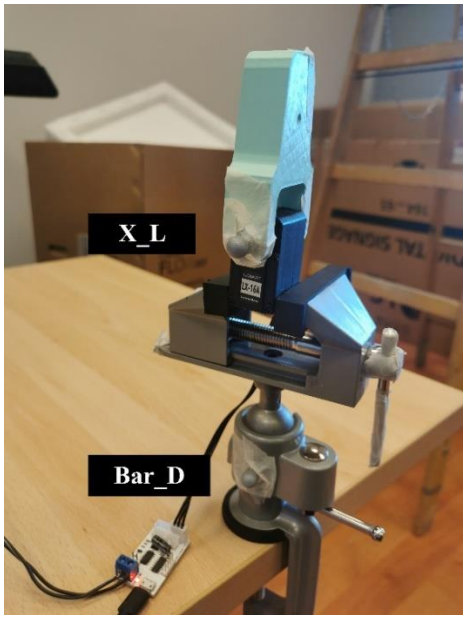
- Beolvassa az értékeket a csv fájlból
- Elkészít két egyenest két-két markerből
- Kiszámolja a két egyenesünk által bezárt szöget
- Ezt a szöget kirajzoljuk egy koordináta rendszerbe

Az alábbi képen az értékek beolvasását láthatjuk. A for ciklus elején beolvassuk az első időpillanatot, ez később fontos lesz a kirajzoltatás során. Aztán a következő lépésben tisztáznunk kellett, hogy mi történjen akkor, ha egy cellát üresen találja a beolvasó ciklus. Mivel ilyenkor az egész sorral nem tud tovább számolni, ezért ha egy cella üres, akkor kihagyja az egész sort. Ez egy radikális megoldásnak tűnhet, viszont a Mocap rendszer stabilitása miatt kevés ilyennemű hibába futottunk bele. A „plusormin” változót kinullázzuk, erre azért van szükség minden sornál, hogy el tudjuk dönteni, hogy a szervó éppen, melyik oldalon van a függőleges egyeneshez képest. A függőleges helyzetet megkeresve hasonlítottuk hozzá az aktuális z_d marker y koordinátáját. Ha átlépi az értéket később a szögelfordulás számítása során kap egy -1-es értéket, így pontosan láthatjuk majd, ha áthalad a nullán a marker. A markerek x, y, z koordinátáiból készítünk egy vektort és ezt a vektort adjuk hozzá az adott marker listájához.

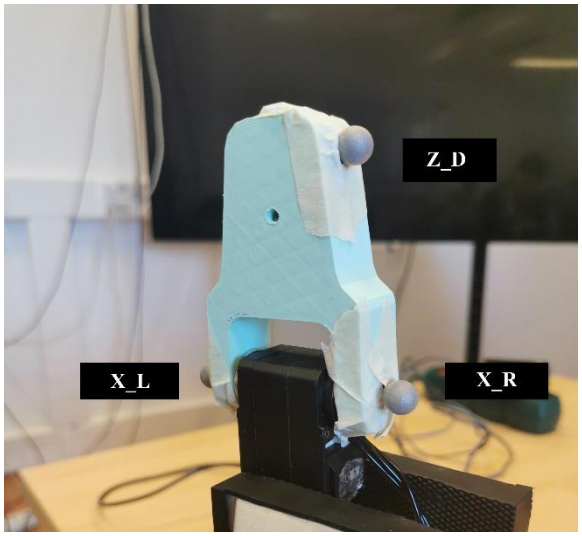
```
i = -1
for row in file_csv_read:
    time_list.append((float)(row[1]))
    if not row[2] or not row[3] or not row[4] or not row[5] or not row[6] or not row[7] or not\
        row[8] or not row[9] or not row[10] or not row[11] or not row[12] or not row[13]:
        next(file_csv_read)
    else:
        i=i+1
        plusormin = 0

        marker_bar_d = [(float)(row[5]), (float)(row[3]), (float)(row[7])]  #be in line with the x_l marker
        vector1 = np.array(marker_bar_d)
        bar_d_list.append(vector1)

        marker_x_l = [(float)(row[5]), (float)(row[6]), (float)(row[7])]
        vector2 = np.array(marker_x_l)
        x_l_list.append(vector2)

        marker_x_r = [(float)(row[8]), (float)(row[9]), (float)(row[10])]
        vector3 = np.array(marker_x_r)
        x_r_list.append(vector3)

        if((float)(row[11]) < -0.125233): #Examine when it moves to the other side
            plusormin = 1

        marker_z_d = [(float)(row[11]), (float)(row[12]), (float)(row[10])]
        vector4 = np.array(marker_z_d)
        z_d_list.append(vector4)

        marker_bar_r = [(float)(row[8]), (float)(row[3]), (float)(row[10])]
        vector5 = np.array(marker_bar_r)
        bar_r_list.append(vector5)
```

5.1. ábra CSV fájlt beolvasó kódrészlet Python nyelvben

A bar_d és az x_r markerből készült egyenes lesz a függőleges egyenesünk, ehhez egy apró módosítást kellett eszközölnünk a markerek értékeiben. A bar_d markert eltoltjuk az x_r marker alá, így kapjuk meg a bar_r-t, amivel tovább dolgozunk. Az alábbi kép jól szemlélteti, hogy a markereket, hogyan helyeztük el a szervón.



5.2. ábra A szervó rögzítve és a két marker elhelyezkedése



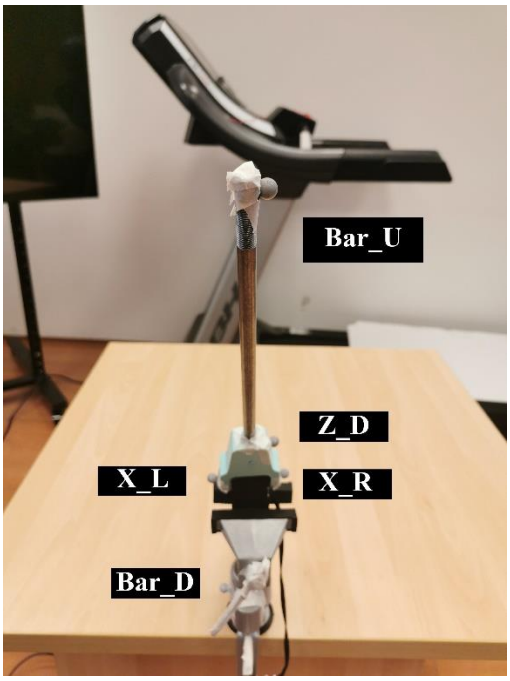
5.3. ábra A szervó és rajta a 3 marker

A markerek listába való betöltése után a szögelfordulás számítása következett. Ehhez két egyenesre volt szükség a függőlegesre, ami fix volt és a mozgó marker egyenesére. Ennek a két egyenesnek a bezárt szögét nézem minden egyes időpillanatban, aztán a korábban vizsgált „plusormin” változóm szerint, ha mínusz tartományban mozog, akkor egy -1-szeres szorzást végzünk el rajta, majd hozzáadom a szögeket tartalmazó listához. Ezután egyszerűen kirajzoltatom egy python library a matplotlib függvényének segítségével.

```
calculate - x_r - bar_r will be the vertical line

    fugg_vector= x_r_list[i] - bar_r_list[i]
    fugg_vector_length = np.linalg.norm(fugg_vector)

    #z_d x_r
    mozgo_vect =  z_d_list[i] - x_r_list[i]
    mozgo_vect_length = np.linalg.norm(mozgo_vect)

    skalar = np.multiply(fugg_vector,mozgo_vect)
    skalar2 = skalar[0] + skalar[1] + skalar[2]

    res1 = skalar2 / (fugg_vector_length * mozgo_vect_length)
    res2 = np.arccos(res1)
    res2todeg = np.rad2deg(res2)

    if(plusormin == 1):
        angle_list.append(res2todeg*(-1))
    else:
        angle_list.append(res2todeg)
```

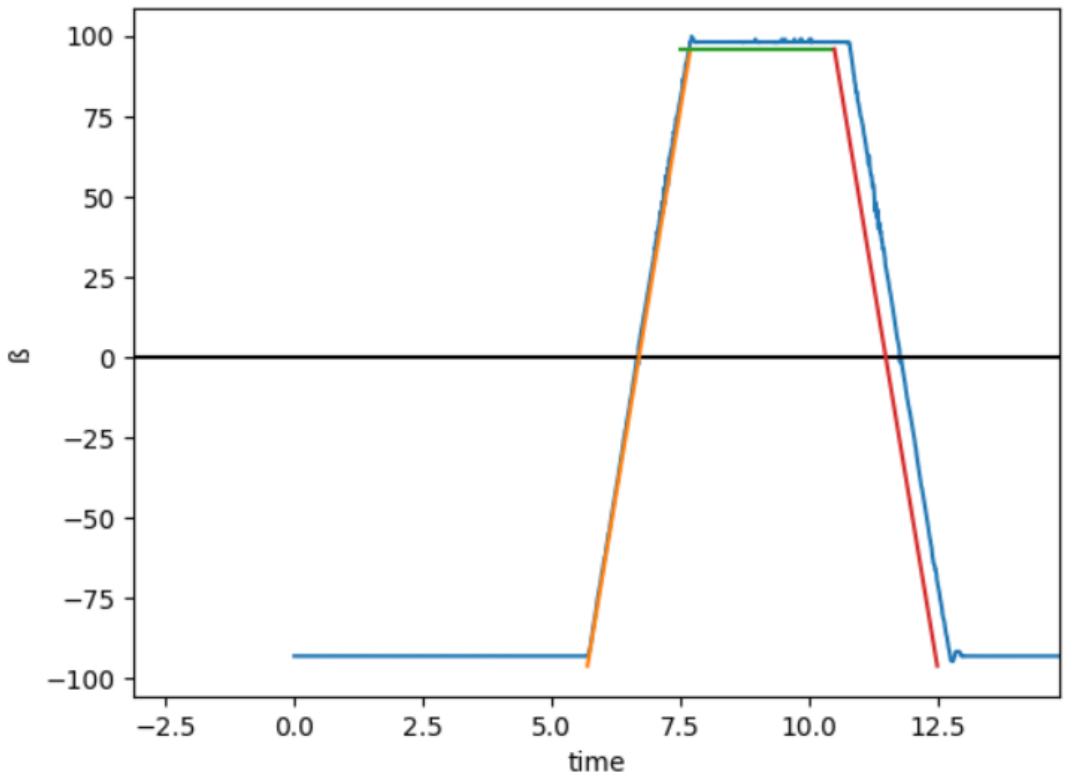
5.4. ábra A számítást elvégző rész

A tömeggel való mérés során egy plusz markert helyeztünk a tömeg – jelen esetben egy fém rúd – végére. Az alábbi ábra jól szemlélteti a markerek elhelyezkedését.



5.5. ábra A szervón elhelyezkedő markerek tömeggel való mérés során

6. ELMÉLETI ÖSSZEHASONLÍTÁS



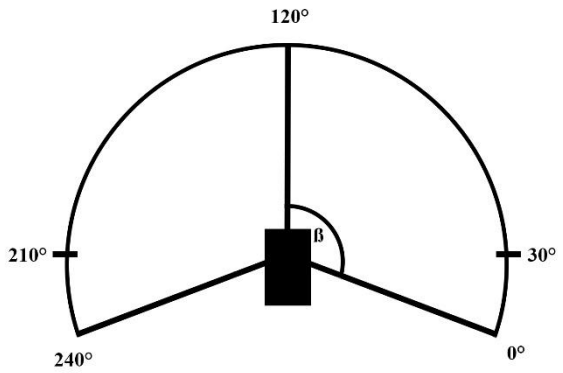
6.1. ábra 900 értékadás a mocap szerint és egy elméleti helyzet

A 6.1 ábrán a 900 értékadás látszódik tömeg nélküli mérés esetén, illetve az ehhez tartozó elméleti pozíció. A szervónak 2 másodperc alatt kell a megadott helyzetbe érnie, ezt még egész jól le tudja követni, aztán 3 másodpercet várakozik, majd 2 másodperc alatt kell visszaérnie a kiinduló helyzetébe. -96° és + 96° között mozog, de már látni lehet a visszaérkezésnél, hogy van egy kis megingás. Illetve az értékadásnál is van egy minimális bemozdulás.

Ezt az elméleti függvényt egy szinusz függvénnyel tudjuk megadni, egy apró módosítással, mert ugye nekünk van benne egy 3 másodperces várakozási időnk. Itt, ha a deriváltunk 0 akkor 3 másodpercig legyen konstans a függvényünk.

7. EREDMÉNYEK ÉRTÉKELÉSE

A következő képen pedig a szögtartomány elhelyezkedését láthatjuk. A szervó a 0°-tól méri az eredményeit a Readservo paranccsal és 0 és 1000 között kapjuk vissza az adatokat, viszont ezt 4,167-el leosztva át tudjuk számolni fokba. A Motion Capture rendszerből kinyert értékeket viszont egy függőleges egyeneshez hasonlítjuk, így számoljuk ki az elfordulás szögét. Ez egy kicsit módosítja az értékeinket, mert a 7.1ábrán látható 120°-ból méri az elfordulást, vagyis ha 100-300 között mozgatjuk a szervót, akkor a kezdeti érték, amit visszakapunk a mocap rendszertől az -96° és -48° között mozog.



7.1. ábra A szervónk szögtartománya, amit lefed

- **Értékadásból származó adatok összevetése tömeg nélküli mérés során**

Elvár fok	Motion Capture rendszerrel mért fok	Szervó által visszaadott fok	Hiba Mocap	Hiba Szervó
-96°	-96,3°	-95,8°	0,3°	0,2°
-48°	-48,1°	-47,5°	0,1°	0,5°
-6,7°	-6,09°	-7,2°	0,61°	0,5°
24°	24,4°	23,3°	0,4°	0,7°
48°	50,7°	48,2°	2,7°	0,2°

96°	95,5°	95,7°	0,5°	0,3°

10. táblázat Értékadás összehasonlítása mocap és szervó eredményeivel

- **Lineáris mozgásból származó adatok összevetése tömeg nélküli mérés során**

Elvárt szög	**Motion Capture rendszerrel mért szög**	**Szervó által visszaadott szög**	**Hiba Mocap**	**Hiba Szervó**
-48°	-47,8°	-48,2°	-0,2°	-0,2°
-6,7°	-5,96°	-7,2°	0,74°	0,5°
24°	24,1°	23,3°	0,1°	0,7°
96°	95,6°	96,5°	0,4°	0,5°

11. táblázat Lineáris mozgásból származó eredmények összehasonlítása

- **Lineáris mozgás II.-ből származó adatok összevetése tömeg nélküli mérés során 1,5 másodperces gyorsasággal**

Elvárt szög	**Motion Capture rendszerrel mért szög**	**Szervó által visszaadott szög**	**Hiba Mocap**	**Hiba Szervó**
-48°	-48°	-48,2°	0°	0,2°
-6,7°	-6,06°	-7,2°	0,64°	0,5°
24°	25°	23,7°	1°	0,3°
96°	95,7°	95,7°	0,3°	0,3°

12. táblázat Lineáris mozgás II. –ből származó eredmények összehasonlítása 1,5 másodperces idő esetén

- **Lineáris mozgás II.-ből származó adatok összevetése tömeg nélküli mérés során 4 másodperces gyorsasággal**

Elvárt szög	**Motion Capture rendszerrel mért szög**	**Szervó által visszaadott szög**	**Hiba Mocap**	**Hiba Szervó**
-48°	-47,8°	-47,8°	0,2°	0,2°
-6,7°	-5,56°	-6,5°	1,14°	0,2°

24°	24,9°	24,2°	0,9°	0,2°
96°	95,9°	96,5°	0,1°	0,5°

13. táblázat Lineáris mozgás II. –ből származó eredmények összehasonlítása 4 másodperces idő esetén

- **Gyorsuló mozgás tömeg nélküli mérés során**

Elvárt szög	**Motion Capture rendszerrel mért szög**	**Hibatartomány**	**Megjegyzés**
96°	95,54°	0,46°	fél kört megy
96°	95,94°	0,06°	Egy kört megy
96°	95,1°	0,9°	Leggyorsabb idő ami alatt átviszi (0,5 s)
96°	94,4°	1,6°	Leglassabb idő ami alatt átviszi (35 s)

14. táblázat Gyorsuló mozgás eredményeinek összehasonlítása

Ezen tesztesetnél került leteszteléssre, hogy leggyorsabban és leglassabban hány másodperc alatt képest megtenni az utat. Ennek eredménye a 14-es táblázatban látható, 5 másodperc és 35 másodperc jött ki eredményül.

- **Változóan gyorsuló mozgás tömeg nélküli mérés esetén**

Elvárt szög	**Motion Capture rendszerrel mért szög**	**Hibatartomány**	**Megjegyzés**
-64,1°	-64,3°	0,2°	2232 ms
119,5°	116,2°	3,2°	2192 ms
-92,6°	-91,5°	1,1°	2102 ms

-119.5°	-118,7°	0,8°	2232 ms

15. táblázat Változóan gyorsuló mozgás eredményei

- **Értékadás 125 g tömeggel való mérés során**

Elvár szög	**Motion Capture rendszerrel mért szög**	**Szervó által visszaadott szög**	**Hiba Mocap**	**Hiba szervó**
-96°	-96,8°	-97,2°	0,8°	1,2
-48°	-48°	-48,7°	0°	0,7
-6,7°	-5,9°	-7,5°	0,8°	0,8
48°	51°	49,1°	3°	1,1
96°	98,2°	96,7°	2,2°	0,7

16. táblázat Értékadás eredményei tömeggel való mérés során

- **Lineáris mozgás I. 125 g tömeggel való mérés során**

Elvárt szög	**Motion Capture rendszerrel mért szög**	**Szervó által visszaadott szög**	**Hiba Mocap**	**Hiba Szervó**
-48°	-47,6°	-48,2°	0,4°	0,2°
-6,7°	-5,4°	-7,4°	1,3°	0,7°
24°	25,6°	24,2°	0,4°	0,2°
96°	97°	96,5°	1°	0,5°

17. táblázat Lineáris mozgás I. eredményei tömeggel való mérés során

A függőleges pozícióba való mozgatás során egy olyan hibát vettünk észre miszerint, nem tudja megállítani a tömeget a függőleges helyzetben, hanem ez láthatóan kileng.

- **Lineáris mozgás II. 125 g tömeggel való mérés során**

Elvárt szög	**Motion Capture rendszerrel mért szög**	**Szervó által visszaadott szög**	**Hiba Mocap**	**Hiba Szervó**

-48°	-47,6°	-49°	0,4°	1°
-6,7°	-5°	-7,4°	1,7°	0,7°
24°	27°	24,8°	3°	0,8°
96°	96,9°	96,7°	0,9°	0,7°

18. táblázat Lineáris mozgás II. eredményei tömeggel való mérés során

- **Gyorsuló mozgás 125 g tömeggel való mérés során**

Elvárt szög	**Motion Capture rendszerrel mért szög**	**Szervó által visszaadott szög**	**Hiba Mocap**	**Hiba Szervó**
96°	97,6°	-96,7°	1,6°	0,7°

19. táblázat Gyorsuló mozgás eredménye tömeggel való mérés során

- **Változóan gyorsuló mozgás 125 g tömeggel való mérés során**

Elvárt szög	**Motion Capture rendszerrel mért szög**	**Hiba Mocap**
42,7°	46,8°	4,1°
78,2°	79,6°	1,4°
29°	27,8°	1,2°
1,2°	1,5°	0,3°

20. táblázat Változóan gyorsuló mozgás tömeggel való mérés során

A tesztelés során nagyon sokszor tapasztaltunk a szervó részéről pontatlanságokat. Ilyen volt a rossz függőleges helyzet felvétele, vagy amikor egyes mérések során nem hajtotta végre a feladatát, hanem megállt minden előjelzés nélkül és nem mozdult tovább. Egyes ciklusokat csak félig tett meg, de ugyanazokkal a paraméterekkel ugyanazokkal a körülményekkel a következő indítás során viszont jól működött. Ezek azok a stabilitási problémák, amik miatt nehéz erről a szervóról csak az általa visszaadott értékei alapján megbecsülni a pontosságát, illetve valós hibatartományuk elfogadását.

A Motion Capture rendszer által mért adatok is sok helyen tértek el nagymértékben. A tömeggel való mérés esetén a tömeg kilengése miatt esetlegesen nagyobb hibatartományt tapasztalhattunk, mint a tömeg nélküli méréseknél.

Viszont a mérési hibatartományok valószínűsíthető, hogy amiatt ilyen nagyok, hogy a tengelyeket nem tudtuk tűpontosan beállítani - ezt kézzel kellett beállítanunk. Amikor nagyjából párhuzamosra állítottuk, akkor nagyságrendileg kisebb hibával mozog az eszközünk. Illetve a satuban sem tökéletesen vízszintesen állt a szervó. Egy ennyire manuálisan beállított rendszert nagyon nehéz jól beállítani, ahhoz, hogy olyan kis hibákat fel tudjunk tárni vele, mint a szervó tévedése az általa visszaadott adathoz képest.

8. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

A rendszernek vannak továbbfejlesztési lehetőségei, elsősorban a mikroszervót illetően, egy komplexebb, nagyobb tudású szervó valószínűsíthető, hogy jobb eredményeket tudna elérni. Ilyen lenne a Dynamixel MX-28T típusú eszköz vagy a Herculex DRS-0201 típusú mikroszervó. Ezekkel szintén végig lehetne mérni a különböző teszteseteket, és össze lehetne hasonlítani a hibatartományt akár mind a 3 említett eszköznél.

Ezen felül lehetne bővíteni még a szoftveres ellenőrzést egy szimulációval, ami a szervónak a mozgását ellenőrzi le.

Egy másik fejlesztés, ami inkább már fizikai szintű, hogy mivel nem lehet tűpontosan párhuzamosan beállítani a tengelyeket, emiatt is pontatlanságokat tapasztaltunk a mocap részéről és ezt még fejleszteni kéne a mérési berendezésen, hogy akkora pontossággal tudjunk mérni, hogy mekkora a hibája a szervónak.

Ezen kívül tovább lehetne fejleszteni automatizálással, több marker használatával és egy olyan 3D nyomtatással készült tartóval, ami alakjából adódóan fixen vízszintesen tud állni a satuban, és egy olyan szervóburkolatot, ahol egy síkban lehet rögzíteni a markereket.

TARTALMI KIVONAT

Szakdolgozatomban egy mikroszervó irányítási és tesztelési folyamatát mutatom be, amelyet az Óbudai Egyetem Mozgáslaborjában mozgáskövető technológiával rögzítettem, majd az így kapott nyers adatokat egy adatfeldolgozó szoftver segítségével dolgoztam fel.

Kezdésként bemutatok különböző felhasználásra tervezett humanoid robotokat, majd több mikroszervót is ismertetek, ami szóba jöhetett a kiválasztás során, ezeket egy technikai összehasonlításban is szemléltetem. Bemutatom magát a Motion Capture technológiát, felhasználási területeit, illetve működési elvét, amiben fontos szerepet játszanak a markerek, az ember által viselt ruházat elhelyezése és a nagy fps-el bíró kamerák. Ezen felül több fejlesztői panelt is vizsgálok, amelyek alkalmasak lehetnek a szervók vezérlésére.

A munka tervezési fázisában kialakítottam a rendszertervet és meghatároztam a rendszer specifikációját. Majd következett a szervó irányításához szükséges irányító szoftver megírása. Ehhez fontos volt tanulmányozni az adott szervó kommunikációs protokollját, hiszen a két eszköz közti kommunikációnak stabilnak kellett lennie. Ezt a tesztesetek megírása követte, amit több részre bontottam fel, majd megkezdődött a szervó tesztelése. Fontos volt, hogy az elvártaknak megfelelően viselkedik-e a motorunk.

Az irányítási és tesztelési folyamatok során, megtörténtek a mérések a Motion Capture rendszerrel is, így megkaptam a nyers adatokat. Az első fő részt tömeg nélküli mérések tették ki, a másodikat pedig 125 g-mos tömeggel való mérések. Ennek feldolgozásához szükség volt egy adatfeldolgozó szoftverre. Az adatfeldolgozás során megkapott értékeket összevetjük a szervó által visszaadott, illetve az elméleti értékkel.

SUMMARY / ZUSAMMENFASSUNG

In my thesis, I present the control and testing process of a micro servo, which I recorded in the Motion Lab at the University of Óbuda using motion tracking technology and then processed the raw data using a data processing software.

A start by presenting humanoid robots designed for different applications, then I describe several micro servos that could be considered for the selection and illustrate them in a technical comparison. I will present the motion capture technology, its applications and its working principle, in which markers, the clothes worn by humans and cameras with high fps play an important role. In addition, I will examine several controller panel that may be suitable for controlling servos.

In the design phase of work, I develop the system design and defined the system specifications. Next I developed the control software. To do this, it was important to study the communication protocol of the servo. Communication between two devices had to be stable. This was followed by the writing of test cases, which I divided into several parts, and then the testing of the servo began. It was important to make sure that our motor should behave as expected.

During the management and testing process, I also took measurements with the Motion Capture system to get the raw data. The first main part consisted of measurements without mass, the second part of measurements with 125 g mass. To process this, a data processing software was needed. The values obtained from the data processing were compared with the values returned by the servo and the theoretical value.

IRODALOMJEGYZÉK

[1] I. Asimov, Én, a robot, Budapest: Kossuth Kiadó, 1966.

[2] N. Bátfai, C. Csukonyi, D. Papp, J. Szabó, S. L. Tóth és F. Kovács, „HKK-Hackers: a halálos robotfegyverek és az asimovi három törvény," *Hadtudományi szemle,* p. 6, 2020.

[3] „Poppy project," [Online]. Available: https://docs.poppy-project.org/en/getting-started. [Hozzáférés dátuma: 12 4 2021].

[4] L. Dudás, Alkalmazott Mesterséges Intelligencia, 2011.

[5] „Boston Dynamics," [Online]. Available: https://www.bostondynamics.com/spot/resources/pomerleau. [Hozzáférés dátuma: 13 4 2021].

[6] G. A. Castillo, B. Weng, T. C. Stewart, W. Zhang és A. Hereid, „Velocity Regulation of 3D Bipedal Walking Robots with Uncertain Dynamics Through Adaptive Neural Network Controller," 2020.

[7] „Hiwonder," [Online]. Available: https://www.hiwonder.hk/products/hiwonder-lx-16a-full-metal-gear-serial-bus-servo. [Hozzáférés dátuma: 13 4 2021].

[8] „Herculex," [Online]. Available: http://www.sgbotic.com/products/datasheets/robotics/HerkuleX0602manual.pdf. [Hozzáférés dátuma: 13 4 2021].

[9] „Dynamixel," [Online]. Available: http://www.dynamixel.com/list.php?dxl=x. [Hozzáférés dátuma: 14 4 2021].

[10] D. Salánki és K. Sarvajcz, „Járásfelismerő kamerarendszer kidolgozása NI LAB-View programozási környezetben," Műszaki Tudományos Közlemények,

Debreceni Egyetem, Műszaki Kar Mechatronikai Tanszék, Debrecen, Magyarország, 2019.

[11] „Optitrack Active Marker," [Online]. Available: https://v22.wiki.optitrack.com/index.php?title=Active_Marker_Tracking. [Hozzáférés dátuma: 14 4 2021].

[12] „Optitrack Cameras," [Online]. Available: https://optitrack.com/cameras/flex-13/. [Hozzáférés dátuma: 14 4 2021].

[13] „Optitrack Camera Placement," [Online]. Available: https://v22.wiki.optitrack.com/index.php?title=Camera_Placement. [Hozzáférés dátuma: 14 4 2021].

[14] „Optitrack Motive," [Online]. Available: https://optitrack.com/software/motive/. [Hozzáférés dátuma: 15 4 2021].

[15] „Hiwonder Collections," [Online]. Available: https://www.hiwonder.hk/collections/servo-controller. [Hozzáférés dátuma: 15 4 2021].

[16] „Hiwonder Debug Board," [Online]. Available: https://images-na.ssl-images-amazon.com/images/I/91DLnW8nTnL.pdf. [Hozzáférés dátuma: 15 4 2021].

[17] Lewansoul, „LSC Series Servo Controller Communication Protocol V1.2".

[18] alexeden, „Github-serial-servo," [Online]. Available: https://github.com/alexeden/serial-servo/find/master. [Hozzáférés dátuma: 11 10 2021].

[19] „Optitrack Documentation," [Online]. Available: https://v22.wiki.optitrack.com/index.php?title=OptiTrack_Documentation_Wiki. [Hozzáférés dátuma: 14 4 2021].

[20] „Github - pendulum,” [Online]. Available: https://github.com/apf99/Simple-Pendulum-Model/blob/master/simple_pendulum.py. [Hozzáférés dátuma: 1 12 2021].

ÁBRAJEGYZÉK

TÁBLÁZATOK JEGYZÉKE