

FELADATKIÍRÁS

Az elektronikusan beadott változatban ez az oldal törlendő. A nyomtatott változatban ennek az oldalnak a helyére a diplomaterv portálról letöltött, jóváhagyott feladatkiírást kell befűzni.



M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék

Komáromi Sándor

ROS ALAPÚ PÓKSZERŰ ROBOT FEJLESZTÉSE

KONZULENS

Nagy Ákos

BUDAPEST, 2022

Tartalomjegyzék

Összefoglaló	6
Abstract.....	7
1 Bevezetés	8
1.1 Pókszerű járó robot	8
1.2 Új mechanikai modell	10
1.3 Célkitűzések.....	11
2 Mechanikai megvalósítás	13
2.1 Mechanikai paraméterek	13
2.2 Nyomtatás	15
2.2.1 Csavarok, betétek tesztelése	15
2.2.2 Alkatrész újra tervezés	17
2.3 A fizikai robot megvalósításának értékelése.....	18
3 Omnidirekcionális járási algoritmus	19
3.1 Láb elérési tartománya.....	19
3.1.1 Az elérési tartomány	20
3.1.2 Láb pályája.....	21
3.1.3 Elérési tartomány középpontja.....	23
3.1.4 Az elérési tartomány továbbfejlesztése.....	Hiba! A könyvjelző nem létezik.
3.2 Maximális lépéstávolság	23
3.2.1 Láb pályájának metszete a lépési tartománnyal.....	24
3.2.2 Lépéstávolság.....	25
3.3 Lépéstávolság kiszámolása a teljes távolság függvényében.....	25
3.4 Lépéssorrend kiválasztása mozgásirány függvényében.....	25
3.4.1 Kritikus szög	26
3.5 Láb új pozíciójának kiválasztása	26
3.6 A mozgási algoritmus hibái	26
4 Szimuláció.....	27
4.1 Szimulátor működése új paraméterekkel	27
4.1.1 Modell betöltése.....	27
4.1.2 Modell paramétereinek kerekítési hibái.....	27
4.2 Járási algoritmus tervezése szimulátorban.....	29

5 Áttérés fizikai robotra	31
5.1 Robot kalibrálása	31
5.1.1 Ofszet szögek.....	31
5.1.2 Motorok mozgási iránya	33
5.1.3 Szervok beállítása	34
5.2 Algoritmus átültetése a fizikai robotra.....	36
6 ROS interfész firmware oldalon	36
6.1 ROS message	38
6.1.1 Sebesség-szögsebesség alapú vezérlés	38
6.1.2 Pozíció-orientáció alapú vezérlés	38
6.1.3 Konklúzió.....	39
6.2 interfész implementálása.....	39
7 ROS alapú vezérlés tesztelése	40
8 Fejlesztési lehetőségek	41
9 Irodalomjegyzék.....	42

HALLGATÓI NYILATKOZAT

Alulírott **Komáromi Sándor**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző, cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2022. 05. 19

.....
Komáromi Sándor

Összefoglaló

A szakdolgozat, vagy diplomaterv elkészítése minden egyetemi hallgató életében egy fontos mérföldkő. Lehetőséget ad arra, hogy az egyetemi évei során megtanultakat kamatoztassa és eredményeit szélesebb közönség előtt bemutassa, s mérnöki rátermettségét bizonyítsa. Fontos azonban, hogy a dolgozat elkészítésének folyamata számos csapdát is rejt magában. Rossz időgazdálkodás, hiányos szövegszerkesztési ismeretek, illetve a dolgozat készítéséhez nélkülözhetetlen „műfaji” szabályok ismeretének hiánya könnyen oda vezethetnek, hogy egy egyébként jelentős időbefektetéssel készült kiemelkedő szoftver is csak gyengébb minősítést kapjon a gyenge minőségű dolgozat miatt.

E dokumentum – amellet, hogy egy általános szerkesztési keretet ad a dolgozatodnak – összefoglalja a szakdolgozat/diplomaterv írás írott és íratlan szabályait. Összeszedjük a Word kezelésének legfontosabb részeit (címsorok, ábrák, irodalomjegyzék stb.), a dolgozat felépítésének általános tartalmi és szerkezeti irányelveit. Bár mindenkire igazítható sablon természetesen nem létezik, megadjuk azokat az általános arányokat, oldalszámokat, amelyek betartásával jó eséllyel készíthetsz egy színvonalas dolgozatot. A részletes és pontokba szedett elvárás-lista nem csupán a dolgozat írásakor, de akár más dolgozatok értékelésekor is kiváló támpontként szolgálhat.

Az itt átadott ismeretek és szemléletmód nem csupán az aktuális feladatod leküzdésében segíthet, de hosszútávon is számos praktikus fogással bővítheti a szövegszerkesztési és dokumentumkészítési eszköztáradat.

Abstract

Ide jön a ½-1 oldalas angol nyelvű összefoglaló, amelynek szövege a Diplomaterv Portálra külön is feltöltésre kerül. Ez a magyar nyelvű összefoglaló angolra fordított változata.

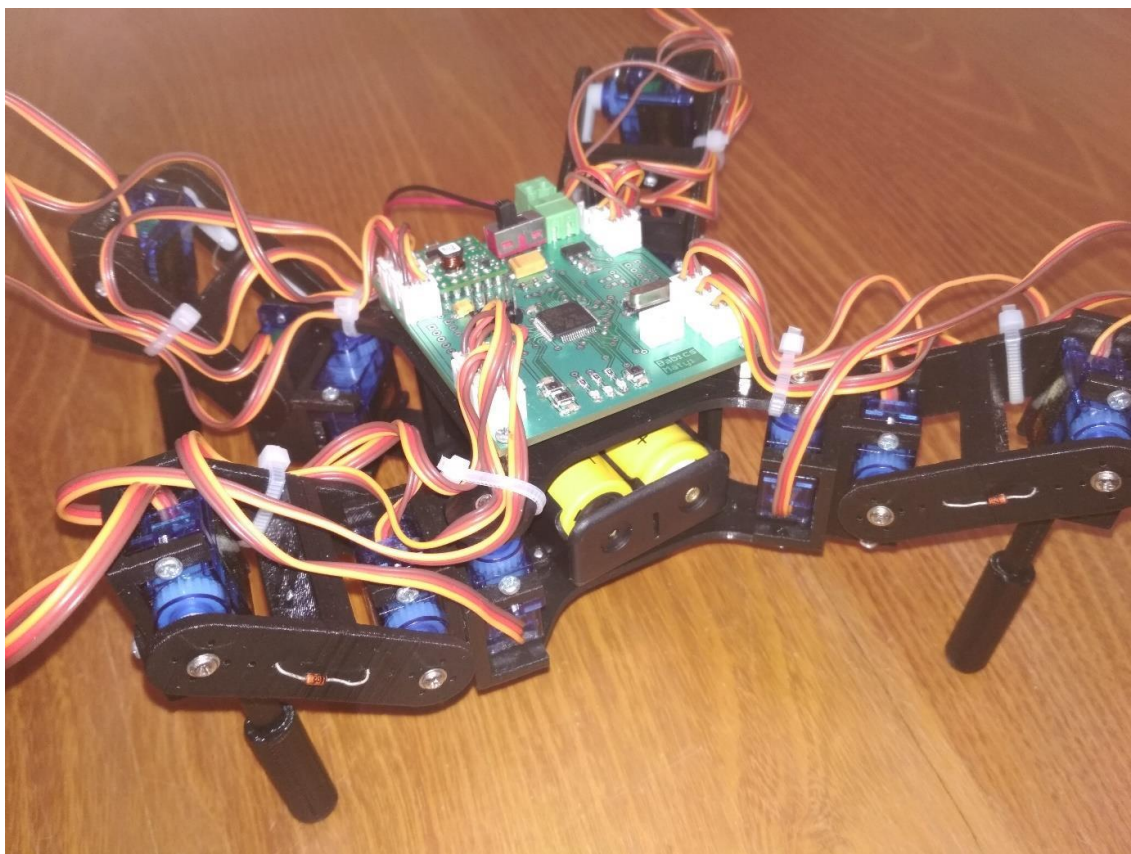
1 Bevezetés

Manapság egyre inkább előtérbe kerül a mobil robotok fejlesztése, felhasználása, napjainkban ezek elsődlegesen az iparban jelennek meg, de egyéb más alkalmazásterületük is feltörekvőben van. A mobil robotok egyik speciális területe a járó robotok, melyek felépítése bonyolultabb a hagyományos kerékmeghajtású robotoknál, viszont sokkal alkalmasabbak a talaj egyenetlenségeinek kompenzálására mozgásuk közben. Ezen robotok elsődleges feladata általában a környezetük feltérképezése, és az abban való problémamentes navigáció megvalósítása. Eme robotok mechanikai kialakításából következik az, hogy képesek többirányú mozgás megvalósítására, vagyis könnyedén tudnak előre, hátra, oldalra, fel- és lelépkedni.

A szakdolgozatom megvalósítása során két másik hallgató munkáját folytattam. Elsődleges feladatomban az önálló laboratóriumom [1] során tervezett robot háromdimenziós (3D) nyomtatással készített alkatrészeinek legyártása és összeszerelése volt. Továbbá célkitűzésem egy olyan új járási algoritmus elkészítése, mely megvalósítja a fent említett mozgási sémát, miközben a robot orientációjának változtatásáért is felel. S végül, de nem utolsósorban az új mechanikával rendelkező robot integrálása is elvárás volt a részemről a Robot Operating System (ROS) [2] interfész alá.

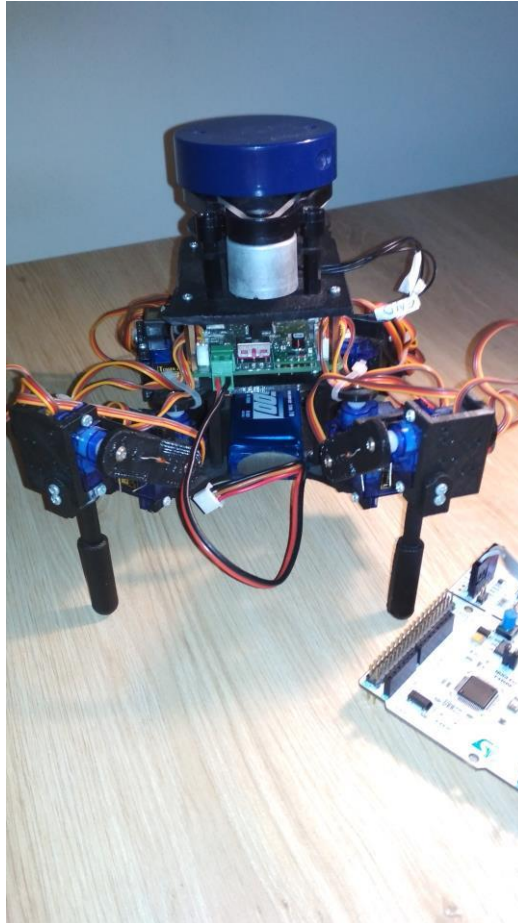
1.1 Pókszerű járó robot

A robot készítése Babits Mátyás munkájával kezdődött. [3] Ő egy négylábú robotot alkotott meg, mely jellegét tekintve pókszerű volt. A robot négy lába egyenként három-három csuklóból állt, olyan elrendezésben, mint a pókok, rákok, vagy a hangyák lábai. A robot alkatrészei 3D nyomtatással készültek. Lábanként három hajlási pont biztosítja az elegendő szabadsági fokot ahhoz, hogy egy-egy láb minden olyan pozíciót fel tudjon venni, amelyet a járási algoritmus megkövetel. A Mátyás által készített robot rendelkezett egy olyan beágyazott szoftverrel (firmware), mely be tudta mutatni a robot járási képességeit. Kezelte a szervomotorokat, valamint a lábakra kiadott végpontokból kiszámolta az inverz geometriai algoritmus segítségével a motorok szögét. A robot mozgása limitált volt; tudott előre menni, valamint jobbra és balra forogni.



1. ábra: Babits Mátyás által készített pókszerű robot.

Ezt követően Massár Lóránt Mátyás fejlesztette tovább a robotot [4], akinek elsődleges célja a robot okosítása volt, ezért olyan szenzorokkal szerelte fel azt, melyek térképezési és kommunikációs feladatokat láttak el. Az okosítás jegyében Lóránt ellátta a robotot egy Light Detection and Ranging (LIDAR) egységgel, illetve egy Internal Measurement Unit (IMU) egységgel is. Előbbi egy olyan lézer alapú távolságérzékelő, mely 360°-ban meghatározza a robottól a külvilág távolságát [5], míg utóbbi a robot térbeli mozgásáról és elhelyezkedéséről biztosít pontos információkat [6]. A robot egy ESP-01 wifi modullal is gazdagodott, mely a távoli vezérlés megvalósítását biztosító kommunikációért felelt, s egyben lehetővé tette a robot irányítását a ROS platformon keresztül. A ROSSerial node segítségével kiépítette a robottal való kommunikációt [7], és a RVIZ [8] valamint a Hector SLAM [9] segítségével környezet feltérképező algoritmust hozott létre.



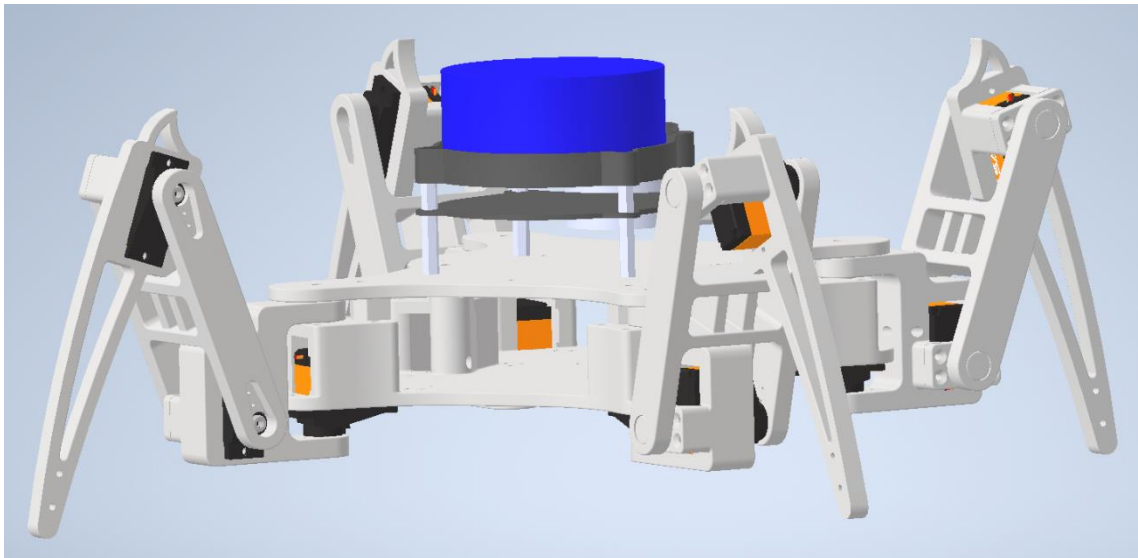
2. ábra: Massár Lóránt Mátyás által készített pókszerű robot.

1.2 Új mechanikai modell

Az önálló laboratóriumom kezdetekor, mikor kézhez kaptam a Lóránt által készített robotot, elsődlegesen a robot mechanikai hiányosságai voltak szembetűnőek. A robot több helyen el volt törve, és a szervomotorjai is nehezen bírták a LIDAR által megnövelt terhet. Ezért mechanikailag teljesen újraterveztem a robotot, erősebb illesztéseket használtam a csuklóknál, valamint a szerelhetőség megtartása érdekében réz betéteket (inserteket) használtam a csavarok rögzítésére. A szervokarok rögzítését megerősítettem, új, erősebb szervomotorokat választottam a robothoz, valamint további két új lábbal is kiegészíthetővé tettem a robot modelljét. Az újratervezés során fontos célkitűzésem volt az, hogy a robot lábainak kinematikai kialakítását ne változtassam meg, azaz a robot lábainak mozgási algoritmus a ugyanolyan maradjon, mint az eredetileg volt, csak paraméterei módosuljanak.

Az önálló laboratóriumom során a robot összeszerelésére már nem maradt időm, így csak annak megtervezése történt meg, melyet a 3. ábra mutat be. Ezen kívül

kiválasztottam a robot új meghajtó motorját, valamint az inverz kinematikai modell paramétereit is meghatároztam, melyeket a 0 pontban részletezek.



3. ábra: Önálló laborom alatt tervezett robot 3D-s terve. [1]

1.3 Célkitűzések

Céлом az volt, hogy a robotot elődjeim munkájának megtartásával tudjam továbbfejleszteni, kiegészíteni. Továbbá elkészítsem a robot új mechanikai vázát, valamint új járási algoritmust tervezek, mely alkalmas többirányú mozgás megvalósítására, mely nagyban kiterjesztené a robot alkalmazási lehetőségeit. Szeretném Lóránt munkáját - melyet az 0 pontban összefoglaltam - újrahasznosítani az általam tervezett roboton. Annak érdekében, hogy a fent említett céljaimat meg tudjam valósítani, megfogalmaztam az alábbi követelményeket:

- Készüljön el a robot mechanikai megvalósítása a kritikus alkatrészek letesztelésével együtt.
- A robot alkalmas legyen bármilyen irányú mozgás megvalósítására, függetlenül annak orientációjától, ezzel új járási algoritmust alkotva.
- A robot mozgás közben képes legyen orientációjának megváltoztatására.
- A robot új járási algoritmusának legyen ROS interfésze.
- A robot vezérelhetővé váljék ROS platformon keresztül.

A fentebb megfogalmazott céljaim eléréséhez először tesztelnem kell a robot tervének szerkezeti paramétereit és összeszerelhetőségét. Be kell kalibrálnom a robot

motorjain lévő szervokarokat, hogy azok megfelelő tartományban mozogjanak. Az általam tervezett járási algoritmus működését meg kell vizsgálnom, egy a Babits Mátyás által készített szimulációban, mielőtt azt a tényleges roboton tesztelném. Továbbá ki kell egészítenem a robot beágyazott kódját úgy, hogy az alkalmas legyen az új motorok használatára és a megváltozott mechanikai paraméterek feldolgozására. Végül újra kell integrálnom a robot rendszerét a ROS környezetbe.

Ábra

2 Mechanikai megvalósítás

Szakedolgozatom megkezdésekor először a robot legyártásával, illetve összeszerelésével foglalkoztam. A végleges gyártás előtt tesztelemeken próbáltam ki az összeszerelhetőségét, kijavítottam a felmerülő hibákat a tervben. A 2.1 pontban részletezem a robot mechanikai paramétereit, melyeknek nagy része megváltozott Mátyás terveitől. Az alkatrészek gyártása során felmerült problémákat, változtatásokat a 2.2 fejezetben fejtem ki bővebben.

2.1 Mechanikai paraméterek

Az Önálló laboratóriumi beszámolómban [1] már részleteztem a robot új mechanikai paramétereit, ám ebben a szakaszban két okból is érdemesnek tartottam ezeket felfrissíteni. Egyrészt a robot korrekt működése érdekében szükségét láttam a finomhangolásnak, amit a szimuláció során szabad szemmel is észlelt pontatlanságok orvoslása miatt kellett elvégezniem (a paraméterek kerekítéséből adódó problémákat, és azok megoldását a 4.1.2 pontban fejtem ki részleteiben). Másrészt az alábbiakban definiált paramétereket, s azok származtatását a dolgozatom során későbbiekben vonatkozási pontként is használom.

A geometriai modell az 1. táblázatban szereplő paraméterek alapján eltolásokból és forgási pontokból épül fel, ezt a továbbiakban a 4. ábra valamint az 5. ábra alapján részletezem. A d_z , d_x , d_y paraméterek a teljes lábat transzformálják annak első forgási pontjába. Innen a láb további részeit az első motor szöge forgatja, majd a d_1 tovább tolja. A második csukó forgatása után a d_2 paraméter tolja tovább a harmadik csuklóba a láb végpontját, majd a d_3 és d_{3x} által eltolt pont adja meg a robot láb végpontjának tényleges pozícióját. Ezzel meghatározható a láb előrefele számolt geometriai egyenlete, melyet Babits Mátyás jegyzetéből [10] emeltem át a (2.1) egyenlettel együtt, ami pedig mátrix szorzásokkal hüen leírja a fent említett eltolásokat és forgatásokat. Az egyenletben lévő C_i és S_i paraméterek koszinusz és szinusz műveleteknek felelnek meg.

Láb	d_x (cm)	d_y (cm)	d_z (cm)	d_1 (cm)	d_2 (cm)	d_3 (cm)	d_{3x} (cm)
Jobb első	5.2462	1.285	7.4962	3.22	7.5	10.7593	-0.45
Jobb hátsó	-5.2462	1.285	7.4962	3.22	7.5	10.7593	0.45

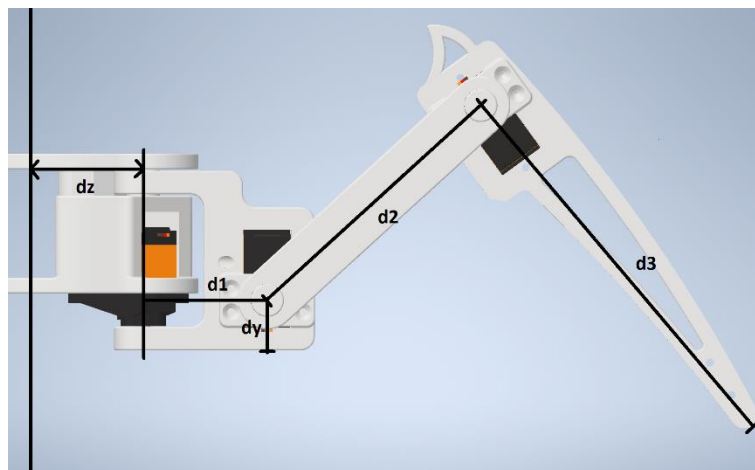
Bal első	5.2462	1.285	-7.4962	3.22	7.5	10.7593	0.45
Bal hátsó	-5.2462	1.285	-7.4962	3.22	7.5	10.7593	-0.45

1. táblázat: A robot lábaihoz tartozó mechanikai paraméterek

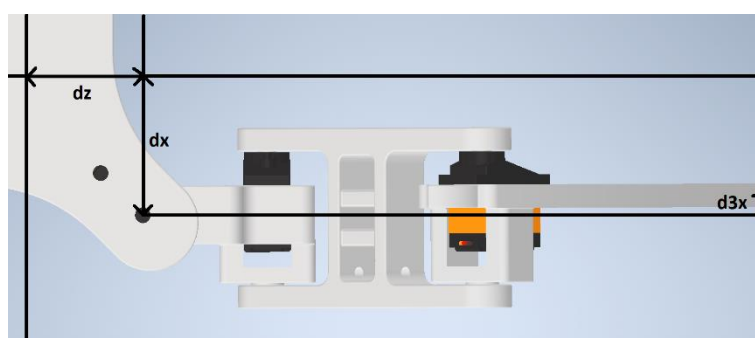
$$\begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} C_1 & 0 & S_1 & 0 \\ 0 & 1 & 0 & 0 \\ -S_1 & 0 & C_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C_2 & -S_2 & 0 \\ 0 & S_2 & C_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C_3 & -S_3 & 0 \\ 0 & S_3 & C_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & d_3x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(2.1)

Az 1. táblázat lábanként tartalmazza azokat a paramétereket, melyeket az inverz kinematikai modell felhasznál a motorok új pozíciójának állításához [10]. A robot koordinátarendszere a szokványostól eltérő módon épül fel, mivel az a robot szémszögéből határozza meg az irányokat, ezeket az xy ábrán mutatom be. A z koordinátatengely mutat előre, az x tengely pedig a z koordinátatengely irányából nézve mutat jobbra. Az előbb említett két tengely metszéspontja határozza meg a robot középpontját felülnézetből nézve. Az y tengely ebből az origóból mutat fölfelé, nullapontja pedig a robottest legalsó pontján van. Ezen analógiát követve a d_z és d_x paraméterek minden lábra meghatározzák az első motor tengelyét a robot testének nullapontjához képest, ezzel eltolva a teljes lábat abba a forgáspontba. A d_y paraméter eltolja a láb kezdőpontját a második motor tengelyének magasságába, ám ez természetesen az első motorra nincs visszahatással, hiszen annak tengelye mentén történik az eltolás, viszont a második motor e tengely menti eltolását így már be lehet állítani. A d_1 eltolja a második motor tengelyét sugárirányba, ezzel meghatározza az első és a második csukló tengelyei közötti távolságot. A d_2 hasonló módon a harmadik motor tengelyét helyezi odébb, ezzel meghatározva a második és harmadik motor tengelyei közötti távolságot. A d_3 paraméter a láb végpontja és a harmadik csukló közötti távolságot határozza meg. Végül a d_3x a második és harmadik csukló tengelye mentén tolja el a láb végpontját az első motor tengelyéhez képest. Erre azért van szükség, mert a láb utolsó alkatrésze kissé eltolva érinti a földet az első forgástengelyhez képest.



4. ábra: A robot lábának paraméterei oldalnézetből.



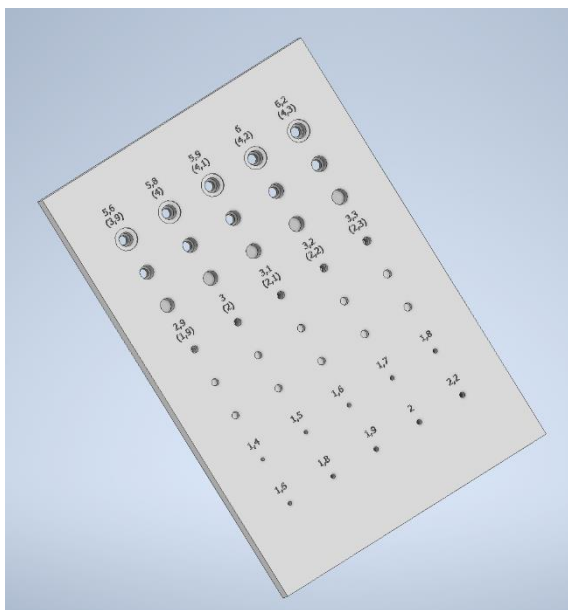
5. ábra: A robot lábának paraméterei felülnézetből.

2.2 Nyomtatás

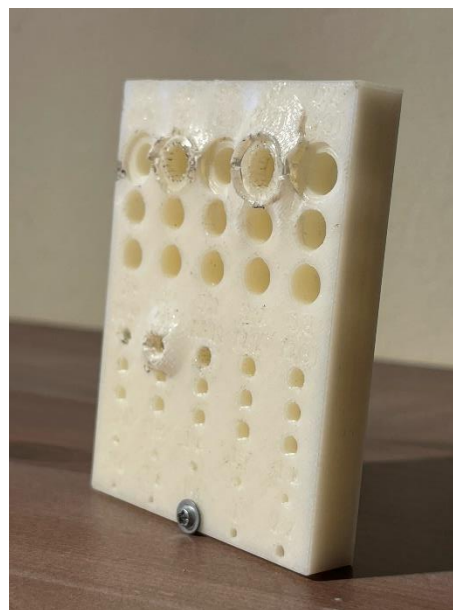
A robot gyártásának megkezdése előtt fontosnak tartottam pár tesztgyomtatás elvégzését, ugyanis ezen prototípust megelőzően még nem használtam műanyagba ágyazható réz menetes betétet, így az külön tesztelést igényelt, melyről a 0 pontban számolok be részletesebben. Továbbá leteszteltem egy teljes láb összeszerelését is, melyet a 0 pontban ismertetek.

2.2.1 Csavarok, betétek tesztelése

A menetes betétek, valamint minden egyéb csavar tesztelése érdekében létrehoztam egy egyszerű tesztelési panelt, melyen tizedmilliméter átmérő különbségű furatok vannak. Így a végleges tervre olyan átmérőjű furatok kerülnek, melyekbe a legjobban illeszkednek a betétek és egyéb csavarok. A 6. ábra segítségével mutatom be a tervezett panelt, melyen jól láthatóak a különböző méretű furatok. Minden átmérőjű furatból többet terveztem, hogy eltérő rögzítési lehetőségeket is ki tudjak próbálni.

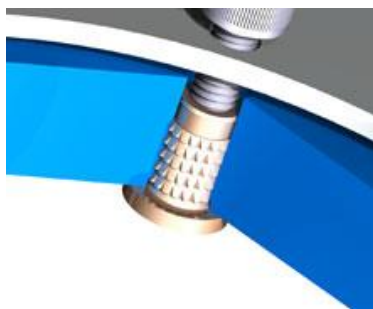


6. ábra: Teszt panel modellje [1]

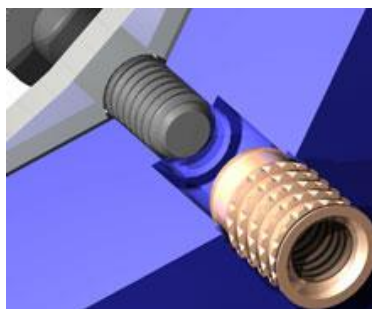


7. ábra: Panel a tesztelés után.

A betétek, melyeket a robotnál használok, fogakkal rendelkeznek, és ezeknél is többféle behelyezési módszerre találtam példát. Az egyik esetben egy satu segítségével lehet belepréselni a műanyagba a rézbetétet, és a csavart ebbe fentről rögzíteni, ekkor terhelés hatására kizárólag az előbb említett fogak tartják a betétet. Ennél biztosabb megoldás a peremes betét használata, melybe a peremmel ellentétes oldalról érdemes belesavarni a csavart. Ezen felhasználáskor a tengelyirányú terhelést a perem, a csavaró erőhatást pedig jellemzően a fogak veszik fel. Erre a technológiára a 8. ábra mutat példát. Azon a helyeken, ahol megoldható volt, én ezt alkalmaztam a rögzítésre. Egyéb esetekben a 9. ábra látható insertet használtam fel, ám én gyártástechnológiai okokból a csavart a behelyezés irányából csavartam bele. További lehetőség a betét behelyezése közben melegítés használata, melyre egy forrasztópáka alkalmas, ekkor a réz jó hővezetőképességéből adódóan a betét körüli műanyag megolvad, és a nyomtatáskor létrejött rétegrendek elmosódnak, így jobban körülölelik a betét fogait. Ezzel egy sokkal erősebb és biztosabb rögzítés készíthető, mely a tesztek során is beigazolódott. A tesztek eredményét a 7. ábra segítségével mutatom be. Azon esetekben, amikor a panel belülről tört ki és felpúposodott, ott melegítést is alkalmaztam. Itt igen látványos volt az, hogy az hordozó anyag egyben szakadt ki a betéttel együtt, és a belső réteghatárokon tört szét.



8. ábra: Peremmel rendelkező insert [11]

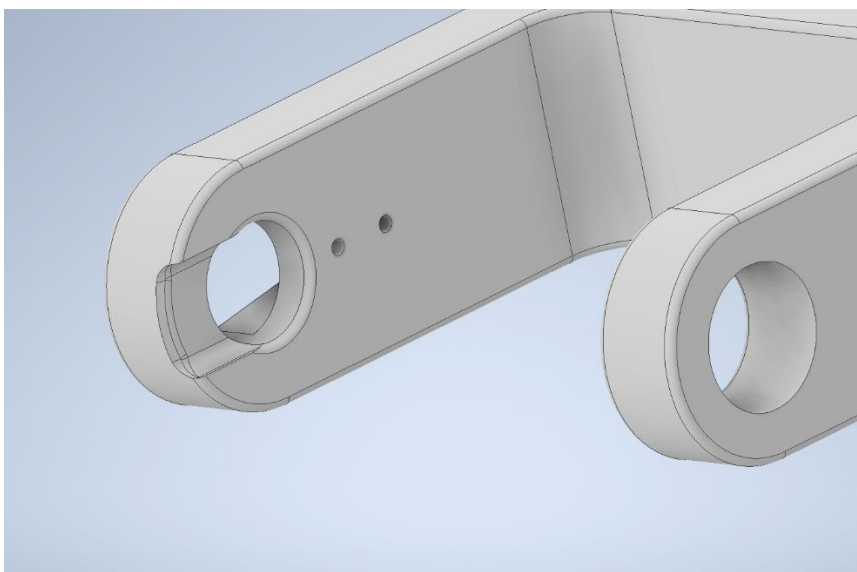


9. ábra: Perem nélküli insert [11]

Ezt az erősebb rögzítési lehetőséget végül nem alkalmaztam a végleges megoldásban, mert a melegítés hatására túlzottan megolvadt műanyag, s így a fellépő oldalirányú erőkkel szemben nem tartotta a betétet, ezért a folyamat közben eldőlt az alkatrész. Emiatt nem tudtam minden esetben biztosítani a pontos behelyezést, ráadásul a melegítés nélküli változat is megfelelő tartóerővel rendelkezett.

2.2.2 Alkatrész újra tervezés

A robot alkatrészeinek végleges elkészítése előtt kinyomtatásra került egy lábnyi alkatrész tesztelés céljából. A teszt alkatrészek összeszerelése során egy fő probléma lépett fel, melyre előzetesen számítottam. A lábak csuklóinak összeszerelésekor a 10. ábra által bemutatott elem két szárát túlzottan szét kellett feszíteni. A végleges tervnél a képen is látható bevágásokat alkalmaztam az összeszerelés megkönnyítése érdekében. A bevágás mellett megnagyobbítottam a távolságot a két szár között, ezzel is elősegítve az összeszerelést, ez így nem visz tengelyirányú mozgást a csuklóba, mert a szervokarok felhelyezésekor csupán az egyik oldalra húzza magát a csuklót.



10. ábra: A könnyebb összeszerelhetőség kedvéért bevágás a robot csuklóján.

2.3 A fizikai robot megvalósításának értékelése

A robot összeszerelése kifejezetten problémamentes volt. Az előzetes tesztek meg hozták eredményüket, minden alkatrész pontosan összeillett egymással, együtt egy erős vázat alkotva. A csuklóban minimális holtjátékot tapasztaltam, ezen talán már csak a csapágó használata segített volna. A réz betétek egytől-egyig tökéletesen működtek, a robot könnyedén szétszerelhetővé vált tőlük, melyet alkalmaznom is kellett, mikor az egyik motor tönkrement. Azonban, ha újra tervezném a robot vázát, valószínűleg kissé máshogy kezdenék neki. A robot jelenleg fölöslegesen tartalmazza az ötödik, hatodik lábhoz tartozó rögzítési pontokat, ez növelte a robot tömegét és nagyította a fő elem méretét. Jelenleg elég távlati célkitűzésnek tűnik a további két láb használata, mert teljesen új mikrokontrollert is kellene használni hozzá. Az akkumulátor elhelyezésén is változtatnék, jelen helyzetében túlzottan hátulra helyeződik a tömegközéppontja a robotnak miatta.

3 Omnidirekcionális járási algoritmus

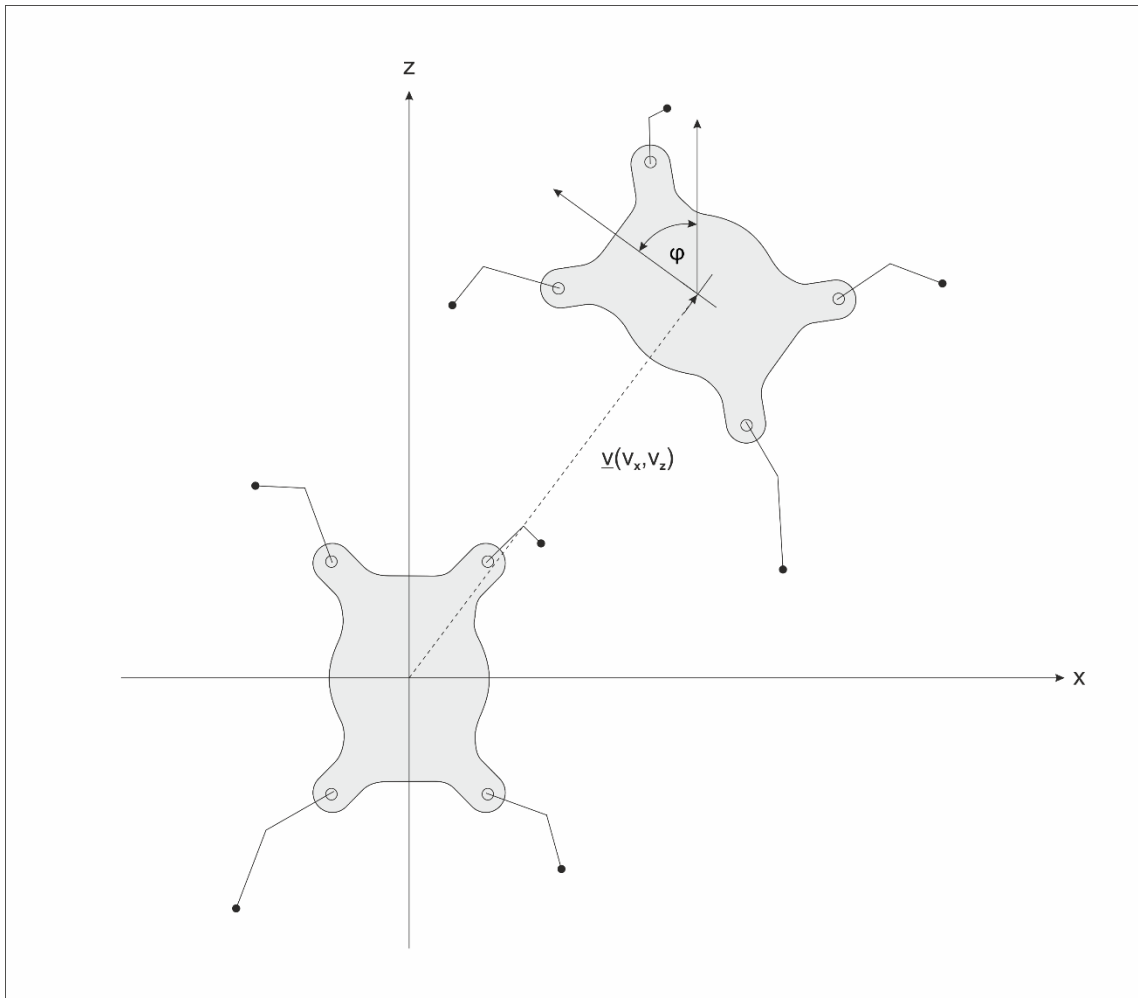
Egy négylábú robot mozgása elsőre egyszerűbbnek tűnhet, mint egy hat, vagy nyolclábúé, hiszen kevesebb láb irányítását és lépési pozícióját kell megvalósítani. Gondoljunk csak a hangyák és pókok mozgására, melyek igen kaotikusnak tűnnek, nehéz megállapítani azt, hogy hogyan mozognak. Azonban ezt kicsit jobban megvizsgálva, elsődleges becslésként kimondhatjuk, hogy egyensúlyozás nélküli, statikus járás esetén azon robotok, melyeknek több mint négy lába van, stabilabb és gyorsabb mozgást tudnak megvalósítani. Ez abból adódik, hogy a robot stabilitásának megtartása érdekében legalább három lábának mindenféleképpen a földet kell érintenie, és a robot tömegközéppontjának is a lábak által alkotott háromszögön belül kell maradnia. Amíg egy négylábú robot a fentiek alapján egyszerre csak egy lábát tudja felemelni, addig egy hat vagy nyolc lábbal rendelkező már három, illetve négy lábat - a szimmetria megtartása érdekében - emel egyszerre a levegőbe, ezzel egy sokkal gyorsabb és stabilabb mozgást érve el. Ezen robotok, a négy lábbal rendelkezőkkel szemben, egyszerre lábaik számának felét fel tudják emelni, ezért két lépés után ismétlődik a mozgási algoritmusuk. A fentiek alapján sejthetjük, hogy egy négylábú statikus mozgást megvalósító algoritmus sokkal lassabb, szaggatottabb, talán egy fokkal bonyolultabb eredményhez vezet, mint egy több lábbal rendelkező esetben.

A járási algoritmus újratervezésekor elsődleges szempontom az volt, hogy a régi programmal ellentétben a robot ne csak egyenesen tudjon menni és a tengelye körül tudjon forogni [12], hanem bármelyik irányba el tudjon indulni, függetlenül attól, hogy milyen a robot orientációja. A robot modelljének újratervezésekor, e célt figyelembe véve, úgy alakítottam ki a tervet, hogy az a tengelyekre szimmetrikus legyen, így a robot előre és oldalra a lépéssorrenden kívül ugyanúgy tudjon lépni. Azt változtatva, hogy mennyit lépjen előre és mennyit oldalra, a robot tetszőleges irányba mozgatható úgy, hogy végig egy irányba néz, tehát ily módon mechanikailag minden irányba mozgathatóvá válik.

A fent említett általános mozgási algoritmus tervezésekor figyelembe vettem azt, hogy a robot ROS környezetbe való integrálása is a céljaim között szerepel, melyre a **Hiba! A hivatkozási forrás nem található.** és 6.3 pontokban térek ki részletesebben. A mozgási parancs két paramétert vár, a cél pozíció koordinátáit a robot koordináta

tengelyében, azaz az elmozdulás vektort, illetve a robot relatív elfordulásának mértékét. Az algoritmus által várt paramétereket az xy ábrán mutatom be. Algoritmusom több különálló részre bontható, először kiszámolja a legnagyobb láblépés távolságot, melyet a lábak elérési tartománya határol. Majd kiszámolja a lelépendő távolság és az elfordulandó szög alapján azt a lépéstávolságot, illetve elfordulási szöget lépésenként, melyet a robot ténylegesen meg fog tenni. A tényleges lépéstávolság nem minden esetben egyenlő a maximummal, hiszen a legtöbb esetben a lelépendő távolság sem egész számú többszöröse a maximum lépéstávolságnak. A továbbiakban az algoritmus a mozgás irányának függvényében összeállítja a lábak lépésének sorrendjét, majd végül kiszámolja minden egyes lábhoz a hozzá tartozó új pozíciót.

A mozgás tervezése során elsődlegesen hivatkozások



3.1 Láb elérési tartománya

A mozgási algoritmus elkészítése előtt, meg kellett vizsgálnom a robot lábainak elérési tartományát, melyek a mechanikai kialakításuk határol. A robot egy lábának

elérési tartományának nevezzük azt a területet, melyen belül a láb mechanikailag képes kinyúlni. A legtöbb ezzel a témakörrel foglalkozó cikkben egy téglalappal egyszerűsítik le a lábak elérési tartományát az egyszerűbb számolás érdekében [13]. Az egyszerűsítést én nem alkalmaztam, mert a téglalap alakú elérési tartomány esetében a robot előre nagyobb távolságot tud lépni, mint oldalra. Számomra elsődleges cél volt a robot szimmetriájának megtartása, így a felhasználtam a teljes mechanikailag megengedett tartományt, melyet 0 fejezetben részletezek. Az elérési tartomány megválasztásakor figyelembe vettem a robot lábainak számának bővítési lehetőségét is, melyre a **Hiba! A hivatkozási forrás nem található.** fejezetben térek ki bővebben.

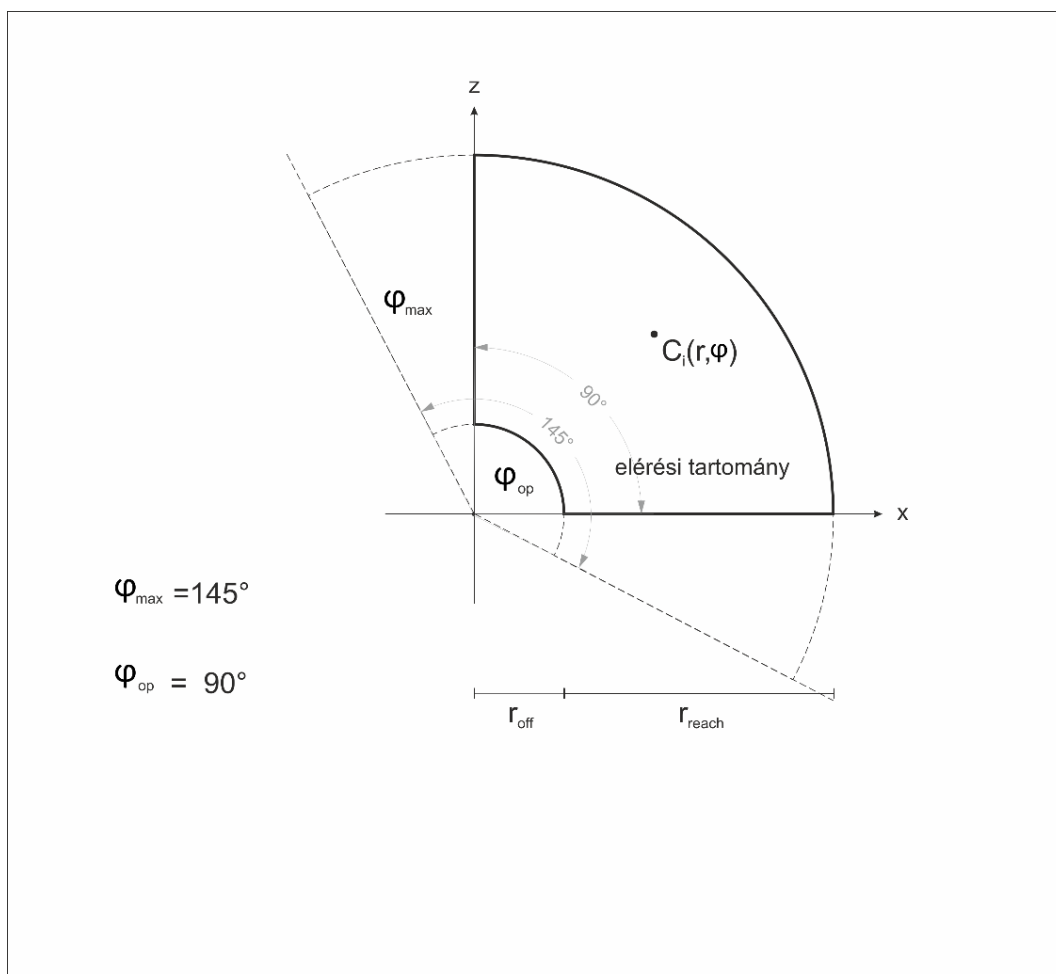
3.1.1 Az elérési tartomány

A továbbiakban bemutatom egy láb elérési tartományának felépítését és paramétereit, elegendő egy tartományt részleteznem, mert a többi ennek tükrözésével kiszámolható. A robot koordináta rendszere úgy épül fel, hogy a robot testének középpontja az origó és a négy láb a négy síknegyeden helyezkedik el. A robot eleje a z tengely fele, míg a jobb oldala az x tengely fele néz, így a jobb mellső lábának koordinátái pozitívek. Az y tengely mentén a robot hasmagassága állítható.

Egy láb mechanikai paramétereit és összeállítását a 2.1 fejezet részletezi, az elérési tartomány szempontjából az a fontos, hogy a három forgó pont közül az első az y tengely körül forog, míg a második és a harmadik az x tengely körül. A robot lábának inverzkinematikai modellje x, y, z koordinátákat használ a pozíció megadására. Viszont a robot lábának kialakítása miatt a láb egy pozíciója sokkal jobban leírható egy φ szög és egy r sugár segítségével, ahol előbbi az első forgó pont szögének felel meg, utóbbi a második és harmadik forgó ponttal egy egyenes mentén növelhető a távolság.

A fent leírtak alapján következik, hogy az elérhető tartomány egy kör valamilyen szelete, a láb mechanikai paramétereit korlátozzák, hogy mennyire messzire tud nyúlni, ezzel egy maximum nyúlási távolság számolható. A körszelet nagyságát tovább korlátozza az első forgó pontba épített szervó forgási korlátja, melynek teljes tartománya ugyan 145° viszont a tipikus használati tartománya csak 90° fok így én az utóbbi értéket használtam fel [14]. Egy láb elérési tartományát tovább szűkíti az, hogy a végpont nem képes teljesen a y tengely körüli forgó pont alá hajolni, hanem csak egy adott paraméter értékéig képes behúzni azt. Ebből következik, hogy az elérési tartomány határolja egy kisebb körív is, melyen belüre nem tudja húzni a lábát a robot. Összefoglalva a robot

lábának elérési tartománya egy fánkseleltre hasonlít, melyet az xy ábrán mutatok be. Korlátait elsősorban mechanikai korlátok adják.



3.1.2 Láb pályája

A robot lábának mozgási pályáját egy egyenes határolja az z - x síkon. A pálya járás közben minden esetben érinti a 0 pontban részletezett és xy ábrán látszó C_i középpontot. A pálya irányultságát elsősorban a robot mozgási iránya szabja meg, melyet az xy képen α szög jelez. A robot mozgási pályája minden esetben két ponton metszi az elérési tartományt, ezen két pont segítségével könnyedén kiszámolható a láb lépésének hossza, melyet a 3.2 fejezet tárgyal részletesebben. A láb pályáját az xy ábrán mutatom be, melyen a p_1 és a p_2 pontok által határolt egyenes a láb pályája.

KÉP

A robot lábának y tengely menti mozgását, Babits Mátyás munkája alapján [12] az (3.1) képlet adja meg. A képletben szereplő d_y paraméter a robot hasmagasságát, a

h változó pedig a láb végpontjának elemelkedését határozzák meg. A t paraméter az időt reprezentálja $[0, 1]$ intervallumon, a mozgás kezdetétől annak végéig.

$$y = -d_y + (1 - (2t - 1)^2) * h \quad (3.1)$$

3.1.3 Elérési tartomány középpontja

A mozgási algoritmus tervezését nagyban megkönnyíti az, ha a láb pályája, függetlenül annak szögétől, minden esetben átmegy egy fix ponton, ezzel meghatározva egy láb elérési tartományának C_i középpontját [15] 4.3 fejezet. Az xy képen látható C_i pontot a 0 fejezetben részletezett elérési tartományba illesztettem be, így annak koordinátáit sugár-szög párral adtam meg. A szimmetria megtartásának érdekében a C_i szöge a teljes lefedettség felével egyezik meg, míg sugara az elérési tartomány két körívének átlaga, az xy ábra alapján.

$$r_{Ci} = \frac{r_{off} + (r_{off} + r_{reach})}{2} \quad (3.2)$$

$$\varphi_{Ci} = \frac{\varphi_{op}}{2} \quad (3.3)$$

A mozgás irányok közötti váltást tovább egyszerűsíti a C_i pont, ugyanis így mozgásiránytól függetlenül létezik olyan pont mely mindig részese a mozgás pályájának. A járási algoritmus befejezésekor az egyik láb mindig ezen pontot érinteni fogja.

3.2 Maximális lépéstávolság

A maximális lépéstávolság az a távolság, amely a robot mozgás iránya függvényében meghatározza mi az a legnagyobb lépés, amely megtétele esetén még éppen egyik láb sem lép ki az elérési tartományából. Ezen szám kiszámításához először ki kell számolni mind a négy lábhoz tartozó lépéstávolságot. A járási algoritmus egyszerűsítése érdekében a négy lépéstávolság minimumát vettem, mert így biztosan egyik láb sem fog kilépni a saját területéről. Ezzel az algoritmust lábmozgatásért felelős részét nagyban leegyszerűsítettem, mert így minden láb ugyanakkorát fog lépni, így nem szükséges külön lábanként figyelni aktuális pozíciójukra, hiszen a minimum

választásának köszönhetően biztosan tudom, hogy minden láb benne marad a mozgási tartományában.

3.2.1 Láb pályájának metszete a lépési tartománnyal

A fent említett lábankénti lépéstávolság meghatározásához először elérési tartomány és a láb pályájának metszéspontjait számoltam ki, mivel a láb pályája minden esetben érinti az elérési tartomány középpontját, így minden esetben két metszéspont keletkezik. A számítás során kihasználtam azt, hogy az elérési tartomány egy 90° -os, körívhez hasonlít, mert így a számítás során felhasználhattam azt, hogy a négy láb elérési tartománya leír egy teljes kört, melyet a síknegyedek határolnak. A számítás során elegendő volt a lábak forgáspontját az origóba tolnom, így az elérési tartományt határoló két körív a négy lábra két kört alkotott. A tartományt határoló két egyenes pedig a tengelyekre esett.

A metszéspontok kiszámítását egy lábhoz számolja ki a függvény, minden lábhoz tartozik egy síknegyed és mindegyiket reprezentáltam egy olyan vektorral, melynek mind az x , mind az y koordinátájának abszolút értéke egy, tehát csak előjelükben van különbség. Ezzel létrehozva egy eltolás vektort.

A láb pályájának egyenesét megszorozva ezzel az eltolás vektorral kiválasztottam, hogy melyik lábhoz tartozó metszéspontokat számolja az algoritmus. Függetlenül attól, hogy melyik síknegyedhez tartozó metszéspontok fontosak, mindig mindkét körre, x tengelyre és y tengelyre meghatározza az összeset. Egy egyenes és egy körnek legfeljebb két metszéspontja lehet, míg két egyenesnek egy-egy. Ezért a legfeljebb hat metszéspontot megszorozza a fentebb említett eltolásvektorral, ennek eredménye az lesz, hogy azon pontok kerülnek, a két kör esetében, a pozitív tartományba, a két tengely esetében a két kör sugarának pozitív értékei közé, melyek a keresett elérési tartományhoz tartoznak. A megfelelő tartományba eső pontokat elmenti az algoritmus egy pontokból álló listába későbbi felhasználásra. A lista használata azért előnyös, mert így, ha egy harmadik pont hibásan bekerülne a metszéspontok közé leellenőrizhető lesz.

Mivel a számítások a szimuláció után a mikrokontrolleren fognak megvalósulni float változók felhasználásával. Ezek nem a legpontosabb számábrázolást valósítják meg, ezért felhasználtam egy epsilon változót a számítási pontosság tűréshatárának beállítására.

3.2.2 Lépéstávolság

A maximális lépéstávolság kiszámítása előtt a négy láb lépéstávolságát számolja ki az algoritmus. A 3.2.1 részben leírtak alapján megkapott két pont közül az lesz a megfelelő, melyet kivonva az elérési tartomány középpontjából a robot mozgásával közel megegyező irányú vektort kapunk. A számítás során a fentebbiekhez hasonlóan felhasználtam egy epsilon változót a pontatlanságból eredő hiba elkerülésére.

A négy lábhoz tartozó lépéstávolság kiszámítása után a maximális lépéstávolság kiszámítása már nem okozott problémát. Ugyan a változó neve ebből a környezetből nézve pontatlanul azon legnagyobb távolságra utal, melyet még minden láb leléphet, mielőtt elérné az elérési tartomány szélét. Ezek után egyértelművé válik az, hogy a négy lépéstávolság minimumát kell venni annak érdekében, hogy soha ne léphessen ki egyik láb sem az elérési tartományból.

3.3 Lépéstávolság kiszámítása a teljes távolság függvényében

A 3.2 fejezetben részletezett maximális lépéstávolság számításakor a robot fizikai korlátait használtam fel, azonban a tényleges lépéstávolság kiszámításhoz további paramétereket is figyelembe kell venni. Ezek a lelépendő távolság, illetve az út alatti elfordulás, melyet a későbbiekben relatív elfordulásnak nevezek.

Jelen helyzetben a robot mozgása akkor nem egyenletes, ha a lelépendő távolság sem egész többszöröse a maximális lépéstávolságnak, így a lépések legvégén egy maradék, kisebb lépés marad, mely megtöri a mozgást. A robot algoritmusában ezen kis lépés eltüntetése érdekében a lelépendő távolságot elosztva a 3.2 fejezetben számolt maximális lépéstávolsággal kiszámolja azt, hogy hány lépés alatt tudná a robot megtenni az utat. A lépésszám számolásakor a maximális lépéstávolságot használta az algoritmus, ezért az így kapott eredmény a lépésszám minimumát képezi. Mivel a robot csak egész számú lépést tud megtenni, ezért az előbb számított lépésszámot egész számra felkerekíti, majd elosztja vele a lelépendő távolságot, így megkapva azt az új lépéstávolságot, mellyel immár egységes lépésekkel lépi le a robot a kiadott távolságot. A fentihez hasonló módon kiszámolja a lépésenkénti forgási szöveget is.

Abban az esetben, ha a robotnak forognia is kell mozgás közben, elképzelhető olyan variáció is - jellemzően kis lelépendő távolság esetén -, ahol a kiszámolt lépésszám megtétele alatt a robot nem tud elegendőt forogni. Ekkor az algoritmus újraszámolja a

lépésszámot a maximális lépésenkénti elfordulással együtt, és a fentihez hasonló módon kiszámolja a lépésenkénti lépéstávolságot és elfordulási szöget is.

3.4 Lépéssorrend kiválasztása mozgásirány függvényében

3.4.1 Kritikus szög

3.5 Láb új pozíciójának kiválasztása

írni a végrehajtó részről is, arról is hogy nem akarom átírni, 6 osból reflektáni rá

3.6 A mozgási algoritmus hibái

A robot forgása

4 Szimuláció

Mielőtt a járási algoritmust ténylegesen implementáltam volna a firmware-be, melyet a 3. fejezetben mutattam be, le kellett tesztelnem annak működését is. Erre legalkalmasabb egy szimuláció használata, melyben előre elkészíthettem a mozgástervezést anélkül, hogy a megjelenő hibák tönkretették volna a robotot. Szerencsémre Babits Mátyás is gondolt erre munkája során, így elkészített egy szimulátort, kifejeztetten ezen robot mozgási algoritmusának megtervezéséhez. [14] Annak ellenére, hogy a szimuláció kifejezeten ehhez a robothoz készült, számomra nehéz volt lecserélni a régi mechanikai modellt az általam tervezettre, valamint az inverz geometriai modell új roboton való alkalmazása során is kihívásokba ütköztem. Ezen nehézségeket részletesebben a 4.1 részben mutatom be. A szimuláció a kezdeti nehézségek ellenére kifejezeten hasznosnak bizonyult, mivel a szimulátor forráskódjának azon része, ami a robot lábainak irányításáért felel, teljesen megegyezik a firmware-en lévő függvényekkel. A teljes mozgási algoritmust meg tudtam tervezni a szimuláción belül, melynek implementálását a 0 pontban kifejttem.

4.1 Szimulátor működése az új paraméterekkel

A Babits Mátyás által készített szimulátort átalakítottam úgy, hogy az általam készített 3D modellel is problémamentesen működjön. A szimulátorba nem teljesen ugyanazon modelleket töltöttem be, mint amelyek ki lettek nyomtatva, mert abban az esetben a szimulátorban kellett volna összeszerelni a robotot is, és az nagyban megbonyolította volna azt. A szimulátorba betöltött elemeket külön a modelltervező programban készítettem el a kinyomtatott alkatrészek alapján. Így a robot szimulációs modellje csak a csuklók miatt áll több darabból. Azon alkatrészeket, melyeket a szimulátorban használtam, egyenként az **xy** ábrától az **xy** ábráig mutatom be. A modellek szimulációba való betöltésekor több hibába is ütköztem, melyeket a 4.1.1 pontban részletezem.

4.1.1 Modell betöltése

A modell betöltése során az elsődleges problémámat az általam használt Standard Tessellation Language (STL) és a Babits Mátyás által használt úgynevezett .mdl formátum közötti különbség okozta. Az általa használt formátummal kapcsolatban az

interneten semmilyen használható információt nem találtam. Ezért tovább próbálkoztam az általa készített fájlok megnyitásával és a szimuláció tanulmányozásával, az abból szerzett információ felhasználhatóságának reményében, ám sajnos itt sem jártam sikerrel. Próbáltam továbbá az általam használt tervezői környezetből kiexportálni ebbe a formátumba a modellt, azonban így sem jutottam közelebb a célomhoz. Végül közvetlenül Babits Mátyáshoz fordultam segítségért, aki elküldte a főképpen általa készített konvertert, melynek segítségével immár sikeresen el tudtam készíteni a szimulátorba beolvasható formátumú modell fájlokat. Azért nem jártam sikerrel a próbálkozásaim során, mert a Mátyás által használt fájlok kódolása és dekódolása is egyedi szintaxissal bírt.

A fájlformátum okozta problémák megoldása után ugyan sikerült az általam tervezett modelleket megjeleníteni a szimulátorban, azonban azok számomra rossz orientációban és méretarányban jelentek meg. Ezeket a modell tervező szoftverben az x, y, z tengelyek orientációja, valamint a modell STL formátumba való exportálásának méretaránya okozták. Én elsőre ezt a szimulátorba beépített pozíció, orientáció, méretarány paraméterekhez adott ofszettel terveztem kijavítani, ám itt sem értem el sikereket. A szimulátorban lévő modell úgy épül fel, hogy az egyes lábdarabok hierarchiailag egymás alá tartoznak, így a fő modell méretarányának változtatásával a többi alkatrész is változott. Ez azt eredményezve, hogy míg a szimulációban eredetileg minden paraméter deciméteres méretarányban volt megadva, a robottest modelljének méretarány változtatása után a test belső koordináta-rendszere megváltozott centiméteres arányra. Így hiába értem el azt, hogy a robot teste újra jó méretű legyen a szimuláció többi részéhez képest, hiszen az inverz geometriai algoritmus egyszerre dolgozik a roboton kívüli és azon belüli paraméterekkel, így újfent méretarány hibát vittem a számításokba.

A méretarány különbségből adódó hiba mellett megjelent egy másik gond is, mely hatással volt az inverz geometriai számításokra és a lábak megfelelő összeállítására. Ezen probléma a modell szerkesztőben az x, y, z tengelyek orientációjából adódott. A modellek saját koordinátatengelye a legtöbb esetben nem esett egybe a geometriai számítások során felhasználtakkal, így a szimulátorban a várthoz képest másik irányba álltak az építőelemek.

Ezen két probléma megoldásaként - a szimulátor jelentős megváltoztatása nélkül - az kínálkozott, hogy magának a modellnek változtassam meg az orientációját és a méretét. A modellt felépítő elemek orientációit úgy változtattam meg, hogy azok a

kinematika során megadotthoz igazodjanak, ezzel kijavítva a hibák nagy részét. Ahogyan fentebb említettem, a szimulátor paraméterei deciméteres mértékegységben vannak megadva, viszont az Inventor nem alkalmas modell exportálására ebben a méretarányban. Ezért úgy változtattam meg a modellek tényleges méretét, hogy a nemlétező deciméteres export helyett centiméteres konverzió után újra visszakapjam az eredeti méreteket. Az így megváltoztatott fájlokkal már tökéletesen működött a szimulátor kinematikai algoritmus.

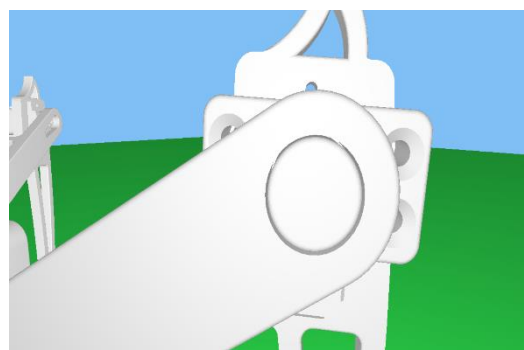
Sajnos a fenti problémák orvoslása igen sok időt vett igénybe a rendelkezésre álló gyakorlati időből, viszont ennek ellenére bizton kijelenthetem, hogy nagy szükség volt a szimulátor használatára a járási algoritmus tervezésekor. A szimulátor nagy biztonságot nyújtott a robot mozgásának vizsgálatok és tesztelések során. Így, annak ellenére, hogy a program működésbe hozására kénytelen voltam a vártnál több időt szánni, mégis határozottan megtérült a befektetett energia.

4.1.2 Modell paramétereinek kerekítési hibái

A modell file-ok sikeres betöltése után egyértelművé vált az, hogy az Önálló laboratóriumom során a 3D-s terv paramétereinél használt kerekítések túl nagyok és az ebből adódó hibák egyértelműen szembetűnőek voltak. A kerekítési hibákra egy példát az 11. ábra mutatok, melyet összevetve az 12. ábra látszik igazán a különbség. A hiányosságok észrevétele után ezredmilliméteres pontossággal adtam meg minden paramétert, melyeket a 1. táblázat részletezek.



11. ábra: a régi, hibás paraméterekkel rendelkező szimuláció



12. ábra: az új paraméterekkel pontosított szimuláció

Nagy valószínűséggel az ilyen magasfokú pontosság alkalmazása értelmetlen, viszont az algoritmusok használatában ez nem okozott erőforrás problémát. Számomra

fontos tanulság az, hogy robot modelljét azon paraméterek alapján érdemes megtervezni, amelyek később a számításokban is felhasználásra kerülnek.

4.2 Járási algoritmus tervezése szimulátorban

szimuláció tényleg a firmware kódja

uml diagram

paraméterek definiálása

Talán magáról a szimuláció integrálásról is itt kéne beszélnem diagrammokkal együtt. -> Az firmware iteráció miatti szimuláció és firmware közötti különbséget elrejtve beszélni az algoritmusról.

5 Áttérés a fizikai robotra

A fizikai robotra való áttérés során először annak lábaihoz és a szervok mozgatásához tartozó új paramétereket kellett beállítanom, azaz kalibrálnom kellett az eszközt. A beállítást és annak paramétereit az 5.1 alfejezetben tárgyalom. továbbá a 5.2 pontban részletezem a járási algoritmus firmware-be való implementálását és annak hibáit.

5.1 Robot kalibrálása

Mielőtt a szimulátorban már jól működő algoritmust kipróbálhattam volna magán a roboton, először kalibrálnom kellett azt. A robot összeszerelése során az utolsó lépést kihagytam, mely a szervokarok feltétele volt. A végső összeszerelésből és a fizikai robot működéséhez elengedhetetlen paraméterek beállításából épül fel a robot kalibrálása. Ezen paraméterek kiegészítik a 2.1 bekezdésben leírtak. A lábhoz tartozó értékeket összefoglalva az xy ábrán mutatom be, valamint az 0 és az 5.1.2 pontokban részletezem őket. Annak érdekében, hogy a szervomotorok univerzálisabbak legyenek, külön darabban adják magát a motort és a hozzá tartozó szervokart. Ennek az oka az, hogy annak ellenére, hogy magának a motornak a működési tartománya limitált, mégis többféleképpen összeszerelhető. Az általam használt motor adatait az 5.1.3 részben fejtem ki bővebben, valamint a motorokat kezelő osztályt az xy ábrán.

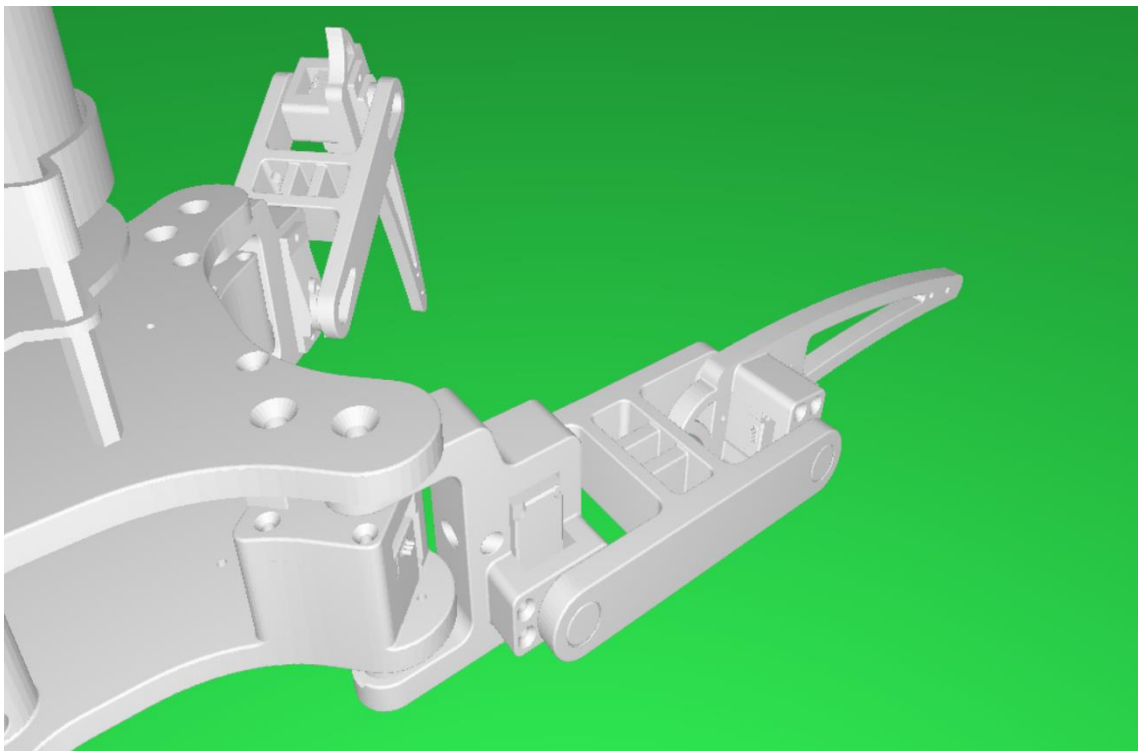
5.1.1 Ofszet szögek

A robot lábaihoz tartozik három paraméter, melyek a robot csuklóihoz tartozó ofszet szögek és a csuklók nulla helyzetének beállítására használhatók [10]. Ezeket a paramétereket meg kellett változtatnom, ugyanis az általam választott új szervok ugyan erősebbek, viszont forgási tartományuk kisebb, részletekbe menően a 5.1.3 bekezdésben mutatom be őket [15]. Az új szervok beállítása során azt tapasztaltam, hogy a motorok a középtértéküktől $\varphi = \pm 0.34\pi$ radiánba állíthatóak, szemben a Babits Mátyás által használtakkal, melyek $\varphi = \pm \pi$ forgási tartománnyal rendelkeztek, amire a továbbiakban még visszatérek az 5.1.3 pontban. A kisebb mozgási tartomány miatt nekem ezen paramétereket pontosabban be kellett állítani. Az 13. ábra mutatom be azt az esetet, amikor az a_1 , a_2 , a_3 szögek mindegyike nulla helyzetben áll. Jól látható, hogy a motorok mozgatásához ezen nullértékek nem előnyösek, hiszen azoknak - a jó működés érdekében

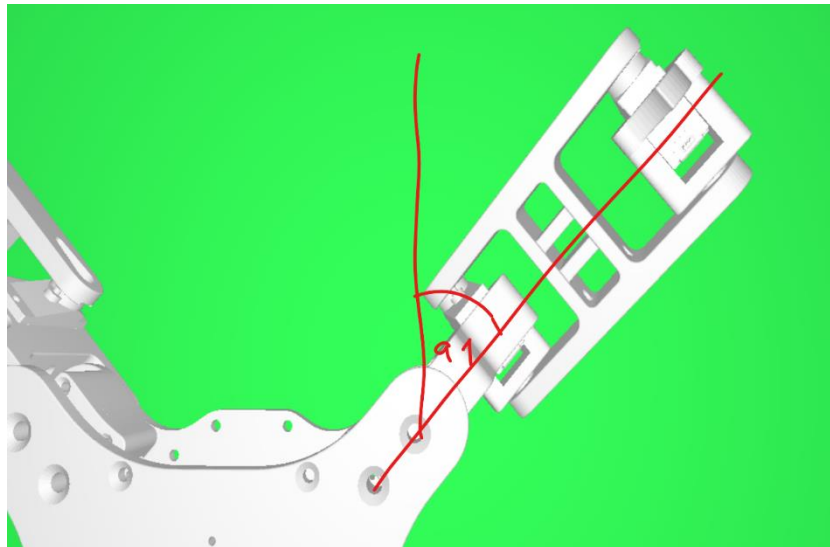
- a mechanikai mozgási tartományuk közepére kellene esniük. Az új paraméterek beállításakor - melyek megértését az 14. ábra és 15. ábra segítik - fontos volt számomra, hogy a motorok mozgási tartományát maximalizáljam. Ezért az a_1 szöget 45 fokra, azaz 0.25π radiánra állítottam, így a láb a null érték megadása esetén az elérési tartományának közepére mutat, melyet a 0 alfejezetben részletezek. Az a_2 és a_3 meghatározásakor hasonló elvet követtem azzal a különbséggel, hogy ezen csuklók esetében a fizikai határok szabják meg azt, hogy a robot mennyire képes behúzni a lábait, így a csuklók mozgási tartományának meghatározásakor a fizikai határokhoz közel állítottam be a mozgási tartományt is. A beállított szögeket a 2. táblázat mutatom be radiánban.

Láb	a_1	a_2	a_3
Jobb első	0.25π	-0.16π	0.5π
Jobb hátsó	0.75π	-0.16π	0.5π
Bal első	-0.25π	-0.16π	0.5π
Bal hátsó	-0.75π	-0.16π	0.5π

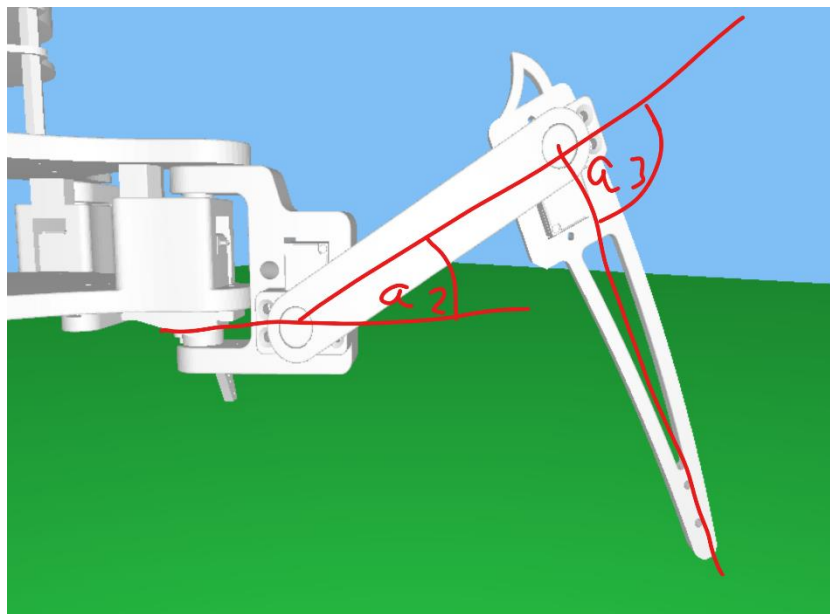
2. táblázat: a motorok beállításakor használt ofszet szögek lábanként, radiánban



13. ábra: a jobb oldali lábon a beállított ofszet szögek nulla értékűek, a bal oldali lábon a 2. táblázat megfelelő értékűek



14. ábra: Az ofszet szögek felülnézetből



15. ábra: az ofszet szögek oldalnézetből

5.1.2 Motorok mozgási iránya

A robot mechanikai kialakítása miatt egyes motorokat fordítva kellett beszerelnem a többiekhez képest. Ezzel a problémával már Babits Mátyás is szembesült munkája során, ezért lábanként létrehozott további három paramétert [16]. A dir_1 , dir_2 , dir_3 változók +1 és -1 értékeket vehetnek fel, ezzel állítva a motorok mozgás irányát. A főként tapasztalati úton beállított változókat az 3. táblázat segítségével mutatom be.

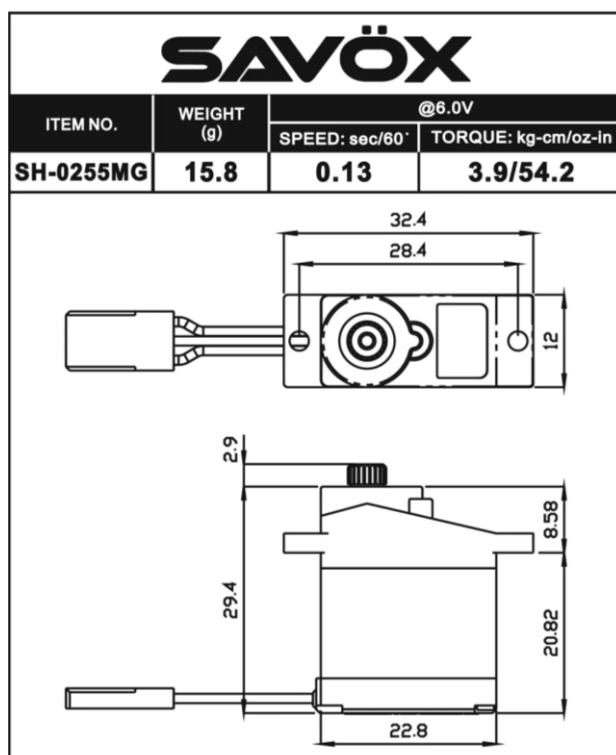
Láb	dir_1	dir_2	dir_3
Jobb első	-1	-1	1

Jobb hátsó	-1	1	-1
Bal első	-1	1	-1
Bal hátsó	-1	-1	1

3. táblázat: a motorok mozgás irányáért felelős paraméterek

5.1.3 Szervok beállítása

A robot tervezése során egyik első lépés volt az új robot mozgatásáért felelős szervomotorok kiválasztása [1]. Válogatás során céltom az volt, hogy megtartva a micro típusú szervókat [17], akár a sebesség rovására is, minél nagyobb nyomatékú motorra essen a választás. Hosszas keresés után végül a Savöx gyártmányú SH-0255MG típusú szervot választottam, mert a gyártó egyik legerősebb micro típusú servoja ez volt, mely adatait a 16. ábra mutatom be [15]. A motor 4.8 volt tápfeszültség mellett 3.1 kg-cm-es nyomatékkal rendelkezik, ez több mint másfélszeres teljesítmény növekedést jelent a kiinduló állapothoz képest. A motor fém fogaskerekekkel rendelkezik, így igen strapabíró, emellett 4.8 volt tápfeszültség mellett 0.16 másodperc alatt forog hatvan fokot, mely azt figyelembe véve, hogy nyomaték optimalizált szervot választottam, mégis kimondottan gyorsnak minősül. A servo gyártói honlapja alapján ugyan a motor teljes működési tartománya 145° , a tapasztalatok nem ezt mutatták, ezért használtam végül a már fentebb megemlített $\pm 0.34\pi$ forgásitartományt, ami 122.4° -nak felel meg.



16. ábra: a robotban használt Savöx szervomotor főbb adatai

A robot végső összeszerelése során, a szervokarok felhelyezésekor igyekeztem a legpontosabban feltenni azokat, azonban fogas kialakításuk miatt csak bizonyos pozíciókban szerelhetők össze. Így a robot lábaiban már a kiinduló állapotban mechanikai ofszet hiba keletkezett. Ezzel a problémával a korábbiakban Babits Mátyás is szembesült, ezért motoronként létrehozott egy úgynevezett *assemblyOffset* paramétert, mely alkalmas ezen kis hibák szoftveres kiküszöbölésére. Ezeknek értékeit az 4. táblázat mutatom be, az értékeket radiánban jelenítem meg.

Láb	assemblyOffset ₁	assemblyOffset ₂	assemblyOffset ₃
Jobb első	0.0	0.02	-0.1
Jobb hátsó	-0.21	0.1	0.005
Bal első	-0.12	0.02	0.05
Bal hátsó	0.0	0.02	-0.1

4. táblázat: a robot motorjainak összeszerelése során keletkezett ofszetet kompenzáló értékek, radiánban

A fentebb részletezett mechanikai ofszet hibák megjelenése miatt több olyan helyzet is kialakult, mikor a motorok szélső határértékükben a lábak alkotóelemei

összeérték és egymásnak feszültek. Ennek megakadályozásának érdekében létrehoztam motoronként két határértéket, egy *positiveLimit* és egy *negativeLimit* paramétert, ameddig a motorok elfordulhatnak. Ezen változóknak alapértékként beállítottam a már fentebb említett $\pm 0.34\pi$ értéket, így csak azon paramétereket mutatom be az 5. táblázat, melyek értékei változtak az összeütközési lehetőség miatt. A táblázatban azért jelennek meg az 4. táblázat szereplő összeszerelési ofszetek, mert sok esetben ezen kis változtatások is már elegendőek voltak a robot alkatrészeinek feszülésének megakadályozására.

Láb	positive ₁	negative ₁	positive ₂	negative ₂	positive ₃	negative ₃
Jobb első	0.25π	-0.25π	$0.34\pi+0.02$		$0.34\pi-0.1$	
Jobb hátsó	0.25π	-0.25π		$-(0.34\pi-0.1)$		$-(0.34\pi-0.005)$
Bal első	0.25π	-0.25π		$-(0.34\pi-0.02)$		$-(0.34\pi-0.05)$
Bal hátsó	0.25π	-0.25π	$0.34\pi+0.02$		$0.34\pi-0.2$	

5. táblázat: a robot motorjainak mozgási tartományát limitáló paraméterek, radiánban

5.2 Algoritmus átültetése a fizikai robotra

A robot kalibrálása nem volt nehéz, azonban mégis több időt vett igénybe, mint amivel előzőekben kalkuláltam. Megbonyolította a helyzetet az is, hogy technológiai okokból fordítva szereltem fel a Bassár Lóránt Mátyás által készített NYÁK-ot, így jelenleg a robot hátrafelé megy előre, de miután ez csak a motorok bekötése miatt van így, a járási algoritmusban ez nem okozott problémát. Tovább bonyolította a végrehajtást az is, hogy az egyik motor elromlott a limitek beállítása során, így azt ki kellett cserélnem.

A kalibrálás okozta nehézségeket viszont enyhítette az, hogy amikor ez elkészült, akkor a szimulátor hasonlósága miatt az abban elkészített járási algoritmust csak át kellett másolnom a firmware kódjába. Futtatás után a robot szinte tökéletesen végrehajtotta a mozgást. Az 5.1.2 részben kifejtett mozgási irányok beállítása után már jól lépegetett a robot. A tesztelés elején még némi problémát okozott az is, hogy a robot nehezen bírta el a saját testtömegét, ám később kiderült az, hogy a kalibrálás során egy másik motor is

megsérült, ezért nagyobb áramot vett fel, majd pedig le is állt. A motor kicserélése után már kiválóan, s egyben problémamentesen működött a berendezés.

6 Vezérlés ROS platformon

A robot ROS platformon keresztül történő mozgásvezérléséért egy a roboton futó ROS node osztály felel. Minden ROS platform alá tartozó egységet ROS nodenak nevezünk, melyek egyenként különálló egységeknek felelnek meg. Majd ezen nodeok kommunikálnak egymással. Ennek feladata az üzenetek küldése és fogadása a számítógép és a robot között. A mozgásért felelős üzenet fogadása a ROS node osztály egy subscriber típusú tagváltozóján keresztül történik, mely a ROS core-t futtató számítógépről kerül kiküldésre. Ugyan a mozgást vezérlő tagváltozó már létezett korábban is, a járási algoritmus megváltozásával megváltoztak azon paraméterek, melyeket a robot vár a ROS rendszeren keresztül. Az üzenet pontos tartalmát a 0 alfejezetben, míg annak implementálását a 0 pontban mutatom be. A robot ROS platformon keresztüli vezérlésének tesztelését, hiányosságait a 6.3 bekezdésben részletezem.

6.1 Robot mozgásért felelős ROS message

A robot ROS alapú mozgásvezérlés tervezése során a két lehetőséget vizsgáltam meg; a sebesség-szögsebesség- és a pozíció-orientációalapú vezérlést. Előbbit a 0 pontban, míg utóbbit a 0 pontban fejtem ki bővebben, valamint részletezem előnyeiket és hátrányaikat.

6.1.1 Sebesség-szögsebesség alapú vezérlés

Ahogy azt már a 3 fejezet elején is ismertettem, a robot járási algoritmus egy relatív pozíciót és egy orientációt vár. Ez felveti a kérdést, hogy miért fontoltam meg a sebesség-szögsebesség alapú vezérlést, hogyha az algoritmus nem így várja? A legtöbb ROS alapú globális és lokális pályatervező sebesség-szögsebesség párokkal vezérli a robotokat, mert így a robot mozgása közben is beleszól annak irányításába, ezzel jobb vezérlést elérve.

A vezérlés implementálása a jelenlegi mozgás algoritmusba megoldható, ám némi bonyodalmakkal jár. A tényleges mozgást végrehajtó rész first in first out (fifo) alapú, így az általam írt algoritmus csak elmenti a kiszámolt pozíciókat, melyeket később végrehajt. Ezért sebességre való vezérlés esetében elengedhetetlen egy minimumút választása, ami alapján vezérelhető lesz a robot. Emiatt ez a módszer - a jelenlegi mozgás végrehajtásért

felelős alrendszert alkalmazva - a hatékony implementálást megnehezítő procedúra bevezetését igényelte volna.

6.1.2 Pozíció-orientáció alapú vezérlés

A sebesség-szögsebesség alapú vezérléssel szemben a pozíció-orientáció alapú vezérlést nagyságrendekkel egyszerűbb implementálni, ugyanis az algoritmust meghívó *AddPathElementMove()* függvény eleve a robot új pozícióját és új orientációját várja. A vezérlés egy nagy hátránya a 0 pontban részletezettekkel szemben az, hogy az általános ROS által használt pályatervező algoritmusok nem képesek ezzel az eljárással irányítani a robotot. Viszont a fenti függvény megvalósításával kvázi egy ROS lokális pályatervezőt készítettem, mert ezen tervezők is célpozíciót kapnak paraméterül. [18]

6.1.3 Konklúzió

A sebesség-szögsebesség alapú vezérlés tehát pontosabb működést lehetővé tevő, több beavatkozási lehetőséget felkínáló, kreatívabb vezérlést megvalósító robotizációt biztosítana, ám a szakdolgozatom készítése során nekem a rendelkezésre álló időkerettel is gazdálkodnom kellett, ami viszont az időhatékony megoldás figyelembevételét helyezte előtérbe. Emellett már a járási algoritmus megtervezésének kezdetén eldöntöttem azt is, hogy nem kívánok változtatni a járást végrehajtó modulon. Így végül a pozíció-orientációt alkalmazó vezérlés mellett döntöttem, melynek implementálását a 0 bekezdésben ismertetem.

6.2 ROS interfész firmware oldalon

A robot által várt ROS üzenetet egy *geometry_msgs::Pose* típusú üzenetként [19] igényli a program, mely üzenet egy három koordinátából álló pozíciót, valamint egy kvaterniókban megadott elfordulási szöveget tartalmaz. Míg előbbivel a robot új pozícióját adom meg, addig az utóbbival a robot elfordulását határozom meg. Az üzenet lekezelését az alábbi kódrészlet tartalmazza:

```
void ROSNode::walk_inCommandCallback(const geometry_msgs::Pose& pose_msg)
{
    //Get position from pose
    walk_Command.x = pose_msg.position.x;
    walk_Command.z = pose_msg.position.z;

    //Get heading from quat
    walk_Command.phi = acos(pose_msg.orientation.x) * 2.0f;

    walk_command_status = true;
```

```
}
```

Jól látható, hogy a pozíció és szög paramétereinek átvétele igen egyszerű. Az üzenet callback függvénye a gyors lefutás érdekében csak átmásolja a pozícióváltozókat, valamint a kvaterniókból kiszámolja a robot elfordulásának paraméterét. Továbbá a program „igaz” állapotba állít egy flaget, mely jelezi, hogy kiértékelhető a *walk_Command* tagváltozó tartalma.

Az adatok kiértékelése előtt a ROS node futtatásáért felelős taszk a *walk_command_status* „igaz” értéke esetén elküldi az adatokat a járásért felelős taszk számára. Majd a járás működtetésért felelős taszk fogadja az üzenetet, végül a *walk.StoreCommand()* függvény segítségével eltárolja azt:

```
while(pdFALSE == xMessageBufferIsEmpty(RosToWalkMessage))
{
    xMessageBufferReceive(RosToWalkMessage, (void*)&(walk_Command),
    sizeof(WalkCommand), 0u);
    walk.StoreCommand(walk_Command);
}
```

A *walk.StoreCommand()* függvény már csak az általam készített *AddPathElementMove()* függvényt hívja meg, mely végrehajtja a szükséges számításokat és elmenti a lábak és a test pozícióit a mozgás végrehajtásához.

```
void Walk::StoreCommand(WalkCommand command)
{
    m_walkScript.AddPathElementMove(mth::float2(command.x, command.y),
    command.phi);
}
```

A paraméterek és az algoritmus értelmezése viszonylag egyszerű feladat volt, így ez példa is jól reprezentálja azt, hogy miért ennyire elterjedt a ROS használata, és miképpen garantálja a hatékony alkalmazás elsajátítását. A robot ROS-on keresztüli tesztelése is hasonlóan egyszerű, s egyben egyértelmű volt, ezt a 0 fejezetben fejtem ki bővebben.

6.3 ROS alapú vezérlés tesztelése

7 Fejlesztési lehetőségek

8 Irodalomjegyzék

- [1] S. Komáromi, „Pókszerű, járó robot fejlesztése,” 2021.
- [2] ROS, „Robot Operating System,” [Online]. Available: <https://www.ros.org/>.
- [3] M. Babits, „Járó robot készítése,” 2018.
- [4] L. M. Massár, „Pókszerű, járó robot fejlesztése,” 2020.
- [5] Wikipedia, „Lidar,” [Online]. Available: <https://en.wikipedia.org/wiki/Lidar>. [Hozzáférés dátuma: 25 05 2022].
- [6] Wikipedia, „Inertial measurement unit,” [Online]. Available: https://en.wikipedia.org/wiki/Inertial_measurement_unit. [Hozzáférés dátuma: 25 05 2022].
- [7] ROS, „ROS Serial,” 01 10 2018. [Online]. Available: <http://wiki.ros.org/roserial>. [Hozzáférés dátuma: 26 05 2022].
- [8] ROS, „RVIZ,” 16 05 2018. [Online]. Available: <http://wiki.ros.org/rviz>. [Hozzáférés dátuma: 26 05 2022].
- [9] ROS, „Hector SLAM,” 14 04 2014. [Online]. Available: http://wiki.ros.org/hector_slam. [Hozzáférés dátuma: 26 05 2022].
- [10] M. Babits, „Direkt és inverz geometria,” in *Járó robot készítése*, 2018, pp. 6-9..
- [11] Tappex, „Multisert brass threaded insert,” [Online]. Available: <https://www.tappex.co.uk/products/brass-threaded-inserts/multisert>. [Hozzáférés dátuma: 20 05 2022].
- [12] M. Babits, „Robot mozgása,” in *Járó robot készítése*, 2018, pp. 9-10..
- [13] Y. Jung-Min, „Crab walking of quadruped robots with a locked joint failure,” in *Advanced Robotics*, Online, Taylor and Francis Online, 2003, pp. 863-878..
- [14] M. Babits, „Szimulátor,” in *Járó robot készítése*, 2018, pp. 14-15..

- [15] Savox, „SH0255MG - Micro Digital Servo,” [Online]. Available: <https://www.savoxusa.com/collections/micro-servos/products/savsh0255mg-micro-digital-mg-servo-13-54#technical-details>. [Hozzáférés dátuma: 20 05 2022].
- [16] M. Babits, „Áttérés fizikai robotra,” in *Járó robot készítése*, 2018, p. 15..
- [17] Sparkfun, „Servos Explained,” [Online]. Available: <https://www.sparkfun.com/servos#types-of-servos>. [Hozzáférés dátuma: 28 05 2022].
- [18] ROS, „Base local planner,” 04 04 2019. [Online]. Available: http://wiki.ros.org/base_local_planner. [Hozzáférés dátuma: 28 05 2022].
- [19] ROS, „geometry_msgs/Pose Message,” 02 03 2022. [Online]. Available: http://docs.ros.org/en/api/geometry_msgs/html/msg/Pose.html. [Hozzáférés dátuma: 25 05 2022].