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.



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](#_Toc104419126)

[Abstract 7](#_Toc104419127)

[1 Bevezetés 8](#_Toc104419128)

[1.1 Pókszerű járó robot 8](#_Toc104419129)

[1.2 Új mechanikai modell 10](#_Toc104419130)

[1.3 Célkitűzések 11](#_Toc104419131)

[2 Mechanikai megvalósítás 13](#_Toc104419132)

[2.1 Mechanikai paraméterek 13](#_Toc104419133)

[2.2 Nyomtatás 15](#_Toc104419134)

[2.2.1 Csavarok, betétek tesztelése 15](#_Toc104419135)

[2.2.2 Alkatrész újra tervezés 17](#_Toc104419136)

[3 Omnidirekcionális járási algoritmus 18](#_Toc104419137)

[3.1 Láb elérési tartománya 19](#_Toc104419138)

[3.1.1 Az elérési tartomány 19](#_Toc104419139)

[3.1.2 Láb pályája 20](#_Toc104419140)

[3.1.3 Elérési tartomány középpontja 20](#_Toc104419141)

[3.1.4 Az elérési tartomány továbbfejlesztése 21](#_Toc104419142)

[3.2 Maximális lépéstávolság 21](#_Toc104419143)

[3.2.1 Láb pályájának metszete a lépési tartománnyal 21](#_Toc104419144)

[3.2.2 Lépéstávolság 22](#_Toc104419145)

[3.3 Lépéstávolság kiszámolása a teljes távolság függvényében 22](#_Toc104419146)

[3.4 Lépéssorrend kiválasztása mozgásirány függvényében 23](#_Toc104419147)

[3.4.1 Kritikus szög 23](#_Toc104419148)

[3.5 Láb új pozíciójának kiválasztása 23](#_Toc104419149)

[3.6 A mozgási algoritmus hibái 23](#_Toc104419150)

[4 Szimuláció 24](#_Toc104419151)

[4.1 Szimulátor működése új paraméterekkel 24](#_Toc104419152)

[4.1.1 Modell betöltése 24](#_Toc104419153)

[4.1.2 Modell paramétereinek kerekítési hibái 26](#_Toc104419154)

[4.2 Járási algoritmus tervezése szimulátorban 27](#_Toc104419155)

[5 Áttérés fizikai robotra 28](#_Toc104419156)

[5.1 Robot kalibrálása 28](#_Toc104419157)

[5.1.1 Ofszet szögek 29](#_Toc104419158)

[5.1.2 Motorok mozgási iránya 29](#_Toc104419159)

[5.1.3 Szervok beállítása 30](#_Toc104419160)

[5.2 Algoritmus átültetése a fizikai robotra 31](#_Toc104419161)

[6 ROS interfész firmware oldalon 32](#_Toc104419162)

[6.1 ROS message 32](#_Toc104419163)

[6.1.1 Sebesség-szögsebesség alapú vezérlés 32](#_Toc104419164)

[6.1.2 Pozíció-orientáció alapú vezérlés 33](#_Toc104419165)

[6.1.3 Konklúzió 33](#_Toc104419166)

[6.2 interfész implementálása 33](#_Toc104419167)

[7 ROS alapú vezérlés tesztelése 35](#_Toc104419168)

[8 Fejlesztési lehetőségek 36](#_Toc104419169)

[9 Irodalomjegyzék 37](#_Toc104419170)

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 – amellett, 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.

# 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 elő-re, hátra, oldalra, fel- és lelépkedni.

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

## 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, míg utóbbi a robot térbeli mozgásáról és elhelyezkedéséről biztosít pontos információkat. 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, és a RVIZ valamint a Hector SLAM segítségével környezet feltérképező algoritmust hozott létre.



. ábra: Massár Lóránt Mátyás által készített pókszerű robot.

## 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 algoritmusa 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]

## Célkitűzések

Célom az volt, hogy a robotot elődjeim munkájának megtartásával tudjam továbbfejleszteni, kiegészíteni. Továbbá elkészítsem a robot új mechanikai vázát, valamint új járási algoritmust tervezzek, mely alkalmas többirányú mozgás megvalósítására, mely nagyban kiterjesztené a robot alkalmazási lehetőségeit. Szeretném Lóránt munkáját - melyet az 0 pontban összefoglaltam - újra hasznosítani 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 nézési irányától.
* A robot mozgás közben képes legyen orientációjának megváltoztatására.
* A robot újraintegrálható legyen ROS platform alá.
* 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

# Mechanikai megvalósítás

Szakdolgozatom megkezdésekor először a robot legyártásával, illetve összeszerelésével foglalkoztam. A végleges gyártás előtt tesztelemeken próbáltam ki az összeszerelhetőségét, kijavítottam a felmerülő hibákat a tervben. A 0 pontban részletezem a robot mechanikai paramétereit, melyeknek nagy része megváltozott Mátyás tervei óta. Az alkatrészek gyártása során felmerült problémákat, változtatásokat a 2.2 fejezetben fejtem ki bővebben.

## 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 elvégeznem (a paraméterek kerekítéséből adódó problémákat, és azok megoldását a 0 pontban fejtem ki részleteiben). Másrészt az alábbiakban definiált paramétereket, s azok származtatását a dolgozatom során későbbiekben vonatkozási pontként is használom.

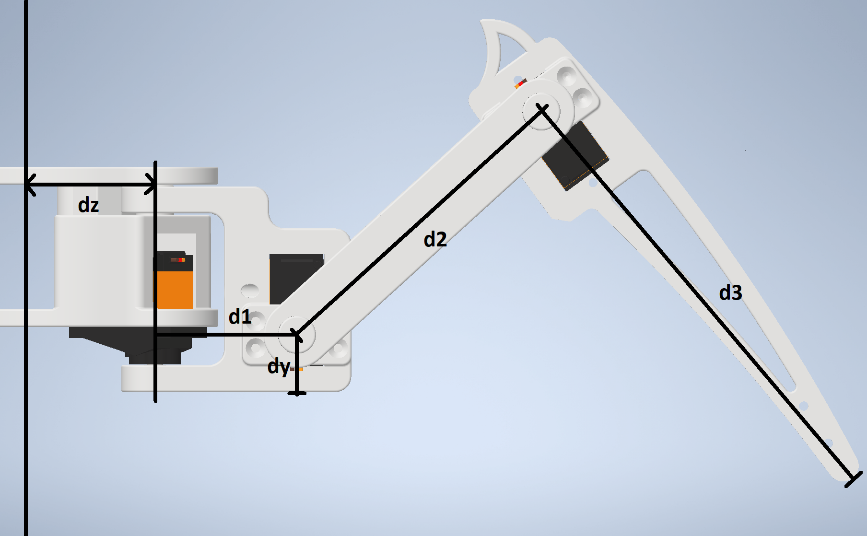
A geometriai modell az 1. táblázatban szereplő paraméterek alapján eltolásokból és forgási pontokból épül fel, ahol a dz, dx, dy paraméterek a teljes lábat transzformálják annak első forgási pontjába. Innen a láb további részeit az első motor szöge forgatja, majd a d1 tovább tolja. A második csukó forgatása után a d2 paraméter tolja tovább a harmadik csuklóba a láb végpontját, majd a d3 és d3x által eltolt pont adja meg a robot láb végpontjának tényleges pozícióját. Ezzel meghatározható a láb előrefele számolt geometriai egyenlete, melyet Babits Mátyás jegyzetéből [5] emeltem át a (2.1) egyenlettel együtt, ami pedig mátrix szorzásokkal hűen leírja a fent említett eltolásokat és forgatásokat. Az egyenletben lévő Ci és Si paraméterek koszinusz és szinusz műveleteknek felelnek meg.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Láb | dx (cm) | dy (cm) | dz (cm) | d1 (cm) | d2 (cm) | d3 (cm) | d3x (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 |

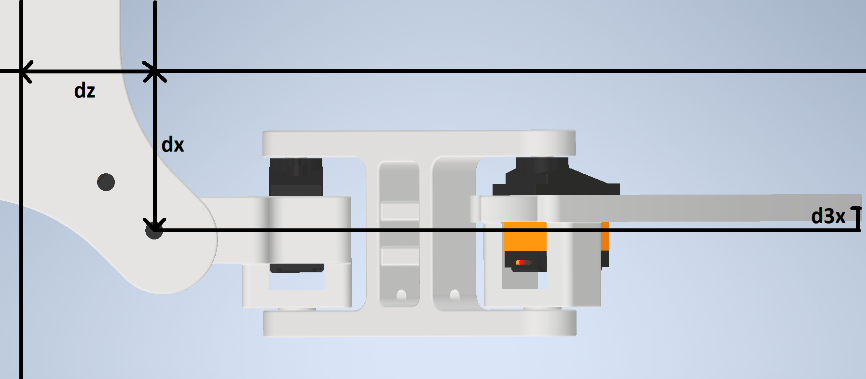
1. táblázat: A robot lábaihoz tartozó mechanikai paraméterek

(.)

Az 1. táblázat lábanként tartalmazza azokat a paramétereket, melyeket az inverz kinematikai modell felhasznál a motorok új pozíciójának állításához. [5] A robot koordinátarendszere a szokványostól eltérő módon épül fel, mivel az a robot szemszögéből határozza meg az irányokat, így 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 dz és dx paraméterek minden lábra meghatározzák az első motor tengelyét a robot testének nullapontjához képest, ezzel eltolva a teljes lábat abba a forgáspontba. A dy 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 d1 eltolja a második motor tengelyét sugárirányba, ezzel meghatározza az első és a második csukló tengelyei közötti távolságot. A d2 hasonló módon a harmadik motor tengelyét helyezi odébb, ezzel meghatározva a második és harmadik motor tengelyei közötti távolságot. A d3 paraméter a láb végpontja és a harmadik csukló közötti távolságot határozza meg. Végül a d3x 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. A fenti magyarázat megértését a 4. ábra és az 5. ábra támogatja, melyek jól reprezentálják a lábakhoz tartozó paramétereket.



4. ábra: A robot lábának paraméterei oldalnézetből.



5. ábra: A robot lábának paraméterei felülnézetből.

## Nyomtatás

A robot gyártásának megkezdése előtt fontosnak tartottam pár tesztnyomtatás 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.

### 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.

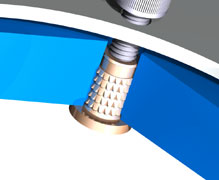


6. ábra: Teszt panel modellje [1]

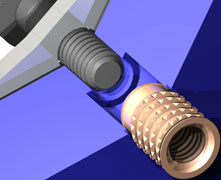


7. ábra: Panel a tesztelés után.

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 belecsavarni 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 [6]



9. ábra: Perem nélküli insert [6]

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.

### 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 kellet 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 csuklóin.

## A fizikai robot megvalósításának értékelése

A robot összeszerelése kifejezetten problémamentes volt. Az előzetes tesztelések meghozták eredményüket, minden alkatrész pontosan összeillett egymással, együtt egy erős vázat alkotva. A csuklókban minimális mozgás lépett fel, melyeken talán már csak a csapágy használata segítene.

# Omnidirekcionális járási algoritmus

Egy négylábú robot mozgása elsőre egyszerűbbnek tűnhet, mint egy hat, vagy nyolc lábúé, hiszen kevesebb láb irányítását és lépési pozícióját kell megvalósítani. Gondoljunk csak a hangyák és pókok mozgására, melyek igen kaotikusnak tűnnek, nehéz megállapítani hogyan mozognak. Azonban kicsit jobban megvizsgálva elsődleges becslésként kimondhatjuk, hogy egyensúlyozás nélküli, statikus [HIVATKOZÁS?] járás esetén azon robotok, melyeknek több lába van stabilabb és gyorsabb mozgást tudnak megvalósítani. Ez abból adódik, hogy a robot stabilitásának megtartása érdekében legalább három lábának mindenféleképpen a földet kell érintse és a robot tömegközéppontja a lábak által alkotott háromszögön belül kell maradjon. Amíg egy négy lábú robot a fentiek alapján egyszerre csak egy lábát tudja felemelni egy hat vagy nyolc lábbal rendelkező esetben már három, illetve négy lábat, a szimmetria megtartása érdekében, emel egyszerre a levegőbe, ezzel egy sokkal gyorsabb és stabilabb mozgást elérve. A fentiek alapján sejthetjük, hogy egy négy lábú statikus mozgást megvalósító algoritmus sokkal lassabb, szaggatottabb, talán egy fokkal bonyolultabb eredményhez vezet, mint egy több lábbal rendelkező esetben.

A járási algoritmus újra tervezésekor elsődleges szempontom volt, hogy a régi programmal ellentétben a robot ne csak egyenesen tudjon menni és a tengelye körül tudjon forogni [HIVATKOZÁS], hanem bármelyik irányba el tudjon indulni, függetlenül attól, hogy milyen irányba áll az eleje. A robot újra tervezésekor, e célt figyelembe véve, úgy alakítottam ki a terveket, hogy az a tengelyekre szimmetrikus legyen, így a robot előre és oldalra a lépéssorrenden kívül ugyanúgy tud lépni. Azt változtatva, hogy mennyit lépjen előre és oldalra, a robot tetszőleges irányba mozgatható úgy, hogy végig egy irányba néz, tehát mechanikailag minden irányba mozgatható.

A fent említett általános mozgási algoritmus tervezésekor figyelembe vettem, hogy a robot ROS [HIVATKOZÁS] környezetbe való integrálása a célom, melyre az XY pontban térek ki részletesebben. A mozgási parancs két paramétert vár, a cél pozíció koordinátait a robot koordináta tengelyében, azaz az elmozdulás vektort, illetve azt az elfordulás mennyiséget, melyet a célpontba érve elforduljon a robot. Algoritmusom több különálló részre bontható, először kiszámolja a legnagyobb láb lépés távolságot, melyet a lábak elérési tartománya határol. Majd kiszámolja a lelépendő távolság és az elfordulandó szög alapján azt a lépéstávolságot, illetve lépésenkénti elfordulást, melyet a robot ténylegesen lépni fog. A tényleges lépéstávolság nem minden esetben egyenlő a maximummal, hogyha a lelépendő távolság nem 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.

## Láb elérési tartománya

A mozgási algoritmus elkészítése elött, 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 [HIVATKOZÁS?]. A legtöbb ezzel a témakörrel foglalkozó cikkben egy téglalappal egyszerűsítik le a lábak elérési tartományát az egyszerűbb számolás érdekében [HIVATKOZÁS]. 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 3.1.1 fejezetben részletezek. Az elérési tartomány megválasztásakor figyelembe vettem a robot lábainak számának bővítési lehetőségét is, melyre a 3.1.4 fejezetben térek ki bővebben.

### Az elérési tartomány

A továbbiakban bemutatom egy láb elérési tartományának felépítését és paramétereit, elegendő egy tartományt részleteznem, mert a többi ennek tükrözésivel 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 0 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 α 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 paraméterei korlátozzák, hogy mennyire messzire tud nyúlni, ezzel egy maximum nyúlási távolságot számolható ki. A körszelet nagyságát tovább korlátozza az első forgó pontba épített szervo forgási korlátja, melynek teljes tartománya ugyan 145° viszont a tipikus használati tartománya csak 90° fok így én az utóbbi értéket használtam fel. 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 értéjéig képes behúzni azt. Ebből következik, hogy az elérési tartomány határolja egy kisebb körív is, melyen belűre nem tudja húzni a lábát a robot.

Összefoglalva …kép alapján összefoglalni… meg képre hivatkozni stb.

### Láb pályája

A robot lábának mozgási pályáját egy egyenes határolja. A pálya járás közben minden esetben érinti a 3.1.3 pontban részletezett 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 eléréi tartományt, ezen két pont segítségével könnyedén kiszámolható a láb lépésének hossza, melyet a 3.2 fejezet tárgyal részletesebben.

### 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 Ci középpontját [HIVATKOZÁS]. Az xy képen látható Ci pontot a 3.1.1 fejezetben részletezett elérési tartományba illesztettem be, így annak koordinátáit sugár-szög párral adtam meg. A szimmetria megtartásának érdekében a Ci szöge a teljes lefedettség felével egyezik meg, míg sugara az elérési tartomány két körívének átlaga.

képletet beszúrni erről

A mozgás irányok közötti váltást tovább egyszerűsíti a Ci pont, ugyanis így mozgásiránytól függetlenül létezik olyan pont mely mindig részese a mozgás pályájának. A járási algoritmus befejezésekor keletkezik olyan pont, mely ebben a középpontban van.

### Az elérési tartomány továbbfejlesztése

## 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.

### 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.

### Lépéstávolság

A maximális lépéstávolság kiszámítása elött a négy láb lépéstávolságát számolja ki az algoritmus. A 3.2.1 részben leírtak alapján megkapott két pont közül az lesz a megfejelő, melyet kivonva az elérési tartomány középpontjából a robot mozgásával közel megegyező irányú vektort kapunk. A számítás során a föntebbiekhez 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 pontatlan, azon legnagyobb távolságra utal, melyet még minden láb leléphet mielőtt elérné az elérési tartomány szélét. Ezek után egyértelművé válik, hogy a négy lépéstávolság minimumát kell venni az érdekében, hogy semmikor ne léphessen egyik láb sem ki az elérési tartományból.

## Lépéstávolság kiszámolása 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 csak akkor egyenletes, ha a lelépendő távolság nem egész többszöröse a maximális lépéstávolságnak a lépések legvégén egy maradék, kisebb, lépés marad, mely megtöri a mozgást. A robot algoritmusa 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 a kapott eredmény a lépésszám minimumát képezi. 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 a lépéstávolságot, mellyel egységes lépésekkel lépi le a robot a kiadott távolságot. A fentihez hasonló módon kiszámolja a lépésenkénti forgási szöget is.

Abban az esetben, ha a robotnak forognia is kell mozgás közben, elképzelhető olyan variáció, jellemzően kis lelépendő távolság esetén, hogy a kiszámolt lépésszám megtétele alatt nem a robot nem tud elegendőt forogni. Ekkor az algoritmus újra számolja a lépésszámot a maximális lépésenkénti elfordulással és a fentihez hasonló módon kiszámolja a lépésenkénti lépéstávolságot és elfordulási szöget.

## Lépéssorrend kiválasztása mozgásirány függvényében

### Kritikus szög

## Láb új pozíciójának kiválasztása

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

## A mozgási algoritmus hibái

A robot forgása

# Szimuláció

Mielőtt a járási algoritmust ténylegesen implementáltam volna a firmware-be, melyet a 3. fejezetben bemutattam, le kellett tesztelnem annak működését. 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 kifejeztetetten ezen robot mozgás algoritmusának megtervezéséhez. [7] Annak ellenére, hogy a szimuláció kifejezetten ehhez a robothoz készült, nehéz volt számomra 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 **Hiba! A hivatkozási forrás nem található.** részben mutatom be. A szimuláció a kezdeti nehézségek ellenére kifejezetten hasznosnak bizonyult, miután 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 kifejtem.

## Szimulátor működése 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 modell tervező 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 0 pontban részletezem.

### Modell betöltése

A modell betöltése során elsődleges problémám az általam használt Standard Tessellation Language (STL) és a Babits Mátyás által használt úgynevezett .mdl formátum közötti különbség volt. 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 reményével, hasonlóan kevés sikerrel. Próbáltam továbbá az általam használt tervezői környezetből kiexportálni ebbe a formátumba a modellt, azonban így sem jutottam közelebb a célomhoz. Végül Babits Mátyáshoz fordultam segítségért, aki elküldte a többnyire általa készített konvertert, mely segítségével 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 egyedi.

A fájlformátum okozta problémák megoldása után ugyan sikerült az általam tervezett modelleket megjeleníteni a szimulátorban, azonban azok nagyon rossz orientációban és rossz méretarányban jelentek meg. Ezeket a modell tervező szoftverben az x, y, z tengelyek orientációja, valamint a modell STL formátumba való exportálásának méretaránya okozzák. Én elsőre a szimulátorba beépített pozíció, orientáció, méretarány paraméterekhez adott ofszettel terveztem kijavítani a keletkezett hibákat, sikertelenül. A szimulátorban lévő modell úgy épül fel, hogy az egyes láb darabok 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 robot test modelljének méretarány változtatása után a test belső koordinátarendszere megváltozott centiméteres arányra. Így hiába értem el azt, hogy a robot teste újra jó méretű volt a szimuláció többi részéhez képest, miután az inverz geometriai algoritmus egyszerre dolgozik a roboton kívüli és azon belüli paraméterekkel, és méretarány hibát vittem a számításokba.

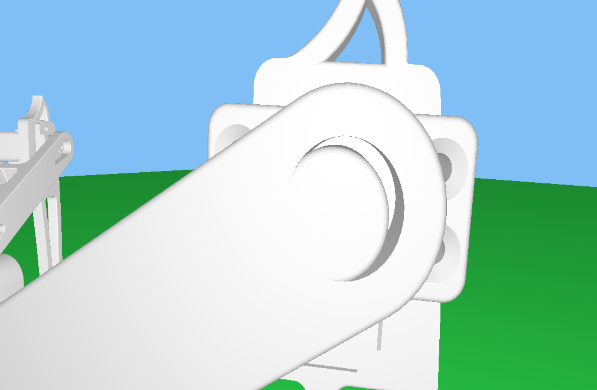
A méretarány különbségből adódó hiba mellett megjelent egy másik is, mely hatással volt az inverz geometriai számításokra és a lábak megfelelő összeállítására. Ezen probléma a modell szerkesztőben az x, y, z tengelyek orientációjából adódott. A modellek saját koordinátatengelye a legtöbb esetben nem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

Ezen két problémára egyetlen és legjobb megoldásként az bizonyult, a szimulátor nagyfokú megváltoztatása nélkül, hogy magának a modellnek kellett megváltoztatnom az orientációját és méretét. A modellt felépítő elemek orientációját úgy változtattam, 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 változattam meg a modellek tényleges méretét, hogy a nemlétező deciméteres export helyett centiméteres után újra visszakapjam az eredeti méreteket. Az így megváltozatott fájlokkal már tökéletesen működött a szimulátor kinematikai algoritmusa.

Sajnos a fenti problémák orvoslása igen sok időt vett igénybe a rendelkezésre álló közel fél évből, viszont kijelenthetem, hogy nagy szükség volt a szimulátor használatára a járási algoritmus tervezésekor. A szimulátor nagy biztonságot nyújtott a robot mozgásának vizsgálatakor és tesztelésekor. Így annak ellenére, hogy a program működésbe hozására kénytelen voltam több időt szánni a vártnál, mindenképpen megérte.

### Modell paramétereinek kerekítési hibái

A modell file-ok sikeres betöltése után egyértelművé vált, hogy az Önálló laboratóriumom során a 3D-s terv paramétereinél használt kerekítések elég nagyok és 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

A képen szöveg látható

Automatikusan generált leírás

12. ábra: az új paraméterekkel pontosított szimuláció

Nagy valószínűséggel az ilyen magasfokú pontosság alkalmazása értelmetlen, viszont így tényleg a pontos adatokkal számolnak az algoritmusok. A jövőre való tekintettel fontos tanulság, hogy robot modelljét azon paraméterek alapján érdemes tervezni, melyek később a számításokban is felhasználásra kerülnek.

## 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.

# Áttérés 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 0 alfejezetben tárgyalom. továbbá a 0 pontban részletezem a járási algoritmus firmware-be való implementálását és annak hibáit.

## 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, melyet az 0 és az 5.1.3 pontokban részletezek. Annak érdekében, hogy a szervomotorok univerzálisabbak legyenek külön darabban adják magát a motort és a hozzá tartozó szervokart. Ennek szerepe az, hogy annak ellenére, hogy a magának a motornak a működési tartománya limitált, többféleképpen összeszerelhető. Az általam használt motor adatait az 5.1.3 részben fejtem ki bővebben.

A fizikai robotnak lábanként van még további hat paramétere, mert minden láb csuklójához tartozik egy ofszetszög, amiket a csuklók nulla helyzetének beállításához kell használni [5], valamint szervonként további három paraméter meghatározza az alapértelmezett forgásirányt.

Babits Mátyás robotjához tartozó paraméterekhez képest az ofszetszöget meg kellett változtatnom, mert az én új mechanikámban a szervok elfordulási szöge csak +/- 0,34π.

A robot kalibrálásakor a 2.1 pontban kifejtett paraméterek mellett továbbiak meghatározására volt szükség, amelyek nagy részét ugyan Babits Mátyás már definiált munkájában [5] [8], úgy éreztem dolgozatom során is érdemes kifejtenem.

A paramétereket és azok összefüggéseit az alábbi UML diagrammok szemléltetem:

### Ofszet szögek

A robot lábaihoz tartozik három paraméter, melyek a robot csuklóihoz tartozó ofszet szögek és a csuklók nulla helyzetének beállítására használhatók. Ezeket a paramétereket meg kellett változtatnom, ugyanis az általam választott új szervok ugyan erősebbek, viszont forgási tartományuk kisebb [8]. A szervok beállítása során azt tapasztaltam, hogy a motorok a középértéküktől φ = +/- 0.34π radiánba állíthatóak, ezt tovább részletezem az 5.1.3 pontban. A kisebb mozgási tartomány miatt nekem ezen paramétereket pontosabban be kellett állítani. Az xy ábrán mutatom be azt az esetet, mikor mind az a1, a2, a3 szögek nulla helyzetben állnak. Jól látható, hogy a motorok mozgatásához ezen nullértékek nem előnyösek, hiszen azoknak a jó működés érdekében a mechanikai mozgási tartományok közepére kellene mutassanak. Az új paraméterek beállításakor - melyek megértését az xy és xy ábrák segítik - fontos volt számomra, hogy a motorok mozgási tartományát maximalizáljam. Ezért az a1 szöget 45 fokra, azaz 0.25π radiánra állítottam, így a láb a null érték megadása esetén az elérési tartományának közepére mutat, melyet a **Hiba! A hivatkozási forrás nem található.**. alfejezetben részletezek. Az a2 és a3 meghatározásakor hasonló elvet követtem azzal a különbséggel, hogy ezen csuklók esetében fizikai határok szabják meg, hogy a robot mennyire képes behúzni 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.

|  |  |  |  |
| --- | --- | --- | --- |
| Láb | a1 (radián) | a2 (radián) | a3 (radián) |
| 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

### Motorok mozgási iránya

A robot mechanikai kialakítása miatt egyes motorokat fordítva kellett beszerelnem, mint másokat. Ezzel a problémával már Babits Mátyás is szembesült munkája során, ezért lábanként létrehozott további három paramétert [9]. A dir1, dir2, dir3 változók +1 és -1 értékeket vehetnek fel, ezzel állítva a motorok mozgás irányát. A főként tapasztalati úton beállított változókat az 3. táblázat mutatom be.

|  |  |  |  |
| --- | --- | --- | --- |
| Láb | dir1 | dir2 | dir3 |
| 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 parméterek

### Szervok beállítása

A robot tervezése során egyik első lépés volt az új robot mozgatásáért felelő szervomotorok kiválasztása [1]. Válogatás során célom az volt, hogy megtartva a micro típusú szervókat a sebesség ellenében 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 szervoja ez volt, mely adatait a kép mutatom be. [8]. A motor 4.8 volt tápfeszültség mellett 3.1 kg-cm-es nyomatékkal rendelkezik, ez több mint másfélszeres teljesítmény növekedést jelent. 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, kimondottan gyorsnak minősül. A szervo gyártói honlapja alapján ugyan a motor teljes működési tartománya 145°, a tapasztalatok nem ezt mutatták, ezért használtam végül a már fentebb megemlített +/-0.34π forgási tartományt, ami körülbelül 122°-nak felel meg.

A robot végső összeszerelése során, a szervokarok felhelyezésekor igyekeztem a legpontosabban feltenni azokat, azonban fogas kialakításuk miatt csak bizonyos pozíciókban összeszerelhetőek. Így a robot lábaiban mechanikai ofszet hiba keletkezett. Ezzel a problémával korábbiakban Babits Mátyás is találkozott, ezért motoronként létrehozott egy úgynevezett assemblyOffset paramétert, mely alkalmas ezen kis hibák szoftveres kiküszöbölésére. Ezeknek értékeit az xy táblázatban mutatom be.

táblázat

A fentebb részletezett mechanikai ofszet hibák megjelenése miatt több olyan helyzet is kialakult, mikor a motorok szélső határértékükben a lábak alkotóelemei összeértek és egymásnak feszültek. Ennek megakadályozásának érdekében létrehoztam motoronként két határértéket, egy posiveLimit és egy negativeLimit paramétert, ameddig a motorok elfordulhat. Ezen változóknak alapértékként beállítottam a már fentebb említett 0.34π értéket, így csak azon paramétereket mutatom be az xy táblázatban, melyek értékei változtak az összeütközési lehetőség miatt.

táblázat

## Algoritmus átültetése a fizikai robotra

A robot kalibrálása nem volt nehéz, azonban mégis több időt vett igénybe, mint azt először gondoltam. Megbonyolította a helyzetet az, hogy 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 nem okozott problémát. Tovább nehezítette a feladatot, hogy az egyik motor elromlott a limitek beállítása során, így azt ki kellett cserélnem.

A kalibrálás okozta nehézségeket enyhítette az, hogy amikor elkészült, 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 elején még némi probléma fellépett, például nehezen bírta el a saját testtömegét, viszont később kiderült, hogy a kalibrálás során egy másik motor is megsérült, mely később le is ált. A motor kicserélése után már nagyon jól, problémamentesen működött a berendezés.

# ROS interfész firmware oldalon

A robot ROS platformon keresztül történő mozgásvezérléséért egy a roboton futó ROSnode osztály felel. Ennek feladata a különböző üzenetek küldése és fogadása a számítógép és a robot között. Az üzenet fogadása a ROSnode osztály egy subscriber típusú tagváltozóján keresztül történik különböző mozgáshoz szükséges információk átadásával. Ugyan a mozgást vezérlő tagváltozó már létezett korábban is, a járási algoritmus megváltozásával megváltoztak azon paraméterek is melyeket a robot vár a ROS rendszeren keresztül. Az üzenet pontos tartalmát a 6.1 alfejezetben míg annak implementálását a 6.2 pontban mutatom be.

## ROS message

A robot ROS alapú mozgásvezérlés tervezése során a két lehetőség merült fel bennem, a sebesség-szögsebesség alapú és a pozíció-orientáció alapú vezérlés. Előbbit a 6.1.1 pontban, míg utóbbit a 6.1.2 pontban fejtem ki bővebben, valamint részletezem előnyeiket hátrányaikat.

### Sebesség-szögsebesség alapú vezérlés

Ahogyan azt már a 3 fejezet elején is részleteztem a robot járási algoritmusa 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, azonban nehézkesebb. 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 valamekkora minimumút választására van szükség, ami alapján vezérelhető lenne a robot. Ez a módszer a jelenlegi mozgás végrehajtásért felelős alrendszer mellett bonyolultabb és nem hatékony implementálást igényelt volna.

### Pozíció-orientáció alapú vezérlés

A sebesség-szögsebesség alapú vezérléssel szemben a pozíció-orientáció alapú vezérlést nagyságrendekkel könnyebb implementálni, ugyanis az algoritmust meghívó AddPathElementMove() függvény eleve a robot új pozícióját és új orientációját várja. A vezérlés egy nagy hátránya a 6.1.1 pontban részletezettekkel szemben, hogy az általános ROS által használt pályatervező algoritmusok nem képesek irányítani így a robotot.

### Konklúzió

Szerettem volna a sebesség-szögsebesség alapú vezérlés mellett dönteni, hiszen az sokkal tágabb izgalmasabb működést tenne lehetségessé, azonban a szakdolgozatom során elkezdtem kifutni, valamint a járási algoritmus megtervezésének kezdetén eldöntöttem, hogy nem szeretnék változtatni a járást végrehajtó modulon. Így végül a Pozíciót-orientációt alkalmazó vezérlés mellett döntöttem, melynek implementálásáról a 6.2 bekezdésben írok.

## interfész implementálása

A robot által várt ROS üzenetet egy geometry\_msgs::Pose típúsú üzenetként [10] várja a program, mert az üzenet egy három koordinátából álló pozíció koordinátából, valamint egy kvaterniókban megadott elfordulási szögekből. Míg előbbivel a robot új pozícióját adom meg, addig az utóbbival a robot elfordulását. 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 a callback függvénye a gyors lefutás érdekében csak átmásolja a pozíció változókat, valamint a kvaerniókból kiszámolja a robot elfordulásának paraméterét. Továbbá a program igaz állapotba állít egy flaget, mely jelez hogy kiértékelhető a walk\_Command tagváltozó tartalma.

Az adatok kiértékelése elött a ROSnode futtatásáért felelős taszk a walk\_command\_status igaz értéke esetén elküldi az adatokat a járásértfelelős taszk számára:

**if**(**true** == node.GetWalk\_commandRequest())

{

xMessageBufferSend(RosToWalkMessage,**void**\*)&(node.GetWalk\_Command()), **sizeof**(WalkCommand), 0u);

node.GetWalk\_commandRequest() = **false**;

}

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 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\_walkScipt.AddPathElementMove(mth::float2(command.x,command.y), command.phi);

}

A paraméterek kitalálása és a folyamat megértése igen egyszerű volt, már ebből az egy példából jól látszik, hogy miért ennyire elterjedt a ROS használata és milyen egyszerű megérteni az alapjait. A robot ROS-on keresztüli tesztelése is hasonlóan könnyed, egyértelmű volt, ezt a 7 fejezetben fejtem ki bővebben.

# ROS alapú vezérlés tesztelése

# Fejlesztési lehetőségek

# 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] | M. Babits, „Direkt és inverz geometria,” in *Járó robot készítése*, 2018, pp. 6-9.. |
| [6] | Tappex, „Multisert brass threaded insert,” [Online]. Available: https://www.tappex.co.uk/products/brass-threaded-inserts/multisert. [Hozzáférés dátuma: 20 05 2022]. |
| [7] | M. Babits, „Szimulátor,” in *Járó robot készítése*, 2018, pp. 14-15.. |
| [8] | Savöx, „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]. |
| [9] | M. Babits, „Áttérés fizikai robotra,” in *Járó robot készítése*, 2018, p. 15.. |
| [10] | 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]. |
| [11] | Wikipedia, „Lidar,” [Online]. Available: https://en.wikipedia.org/wiki/Lidar. [Hozzáférés dátuma: 25 05 2022]. |
| [12] | Wikipedia, „Inertial measurement unit,” [Online]. Available: https://en.wikipedia.org/wiki/Inertial\_measurement\_unit. [Hozzáférés dátuma: 25 05 2022]. |