

## Kivonat

A dolgozat témája terepen alkalmazható mobilis robothoz köthető feladatok elvégzése, és mérések a valós rendszerrel. A robot egy négykerekű csúszáskormányzású SSMR típusú mobilis robot, a négy kereket egy nagy fogaskerék áttétel segítségével hozzuk mozgásba, az áttétel aránya 1:70, és a kerék maximális forgási szögsebessége  $90^\circ/\text{s}$ , a kerekek átmérője 0.40m. A kerekek forgási sebességének a szabályzására PID típusú szabályzókat alkalmaztam, amelyeket FPGA alapú fejlesztőlapon és szoftveresen valósítottam meg. Matlab segítségével megbecsültem a rendszer átviteli függvényét és ezután felhasználva terveztem meg a PID szabályzókat. A rendszer tervezésénél figyeltünk a kutatás-fejlesztési lehetőségekre, így lehetőség van más típusú hardveres és szoftveres szabályzókat is integrálni a rendszerbe minimális erőfeszítéssel.

A rendszer központi feladatait egy kis méretű számítógép végzi, amelyen Ubuntu Linux alapú operációs rendszer fut. A számítógépen futó programokat ROS környezetben implementáltam C/C++ nyelven.

Inkrementális szenzorok segítségével mérem a kerekek forgási sebességét, hall effektusra alapuló szenzorokkal a fogaskerék áttételeket hajtó DC motorok áramát.

A mért értékeket FPGA segítségével dolgozom fel, és küldöm tovább a központi számítógépnek.

A dolgozat egyik fő témája az FPGA és ROS integrációjának lehetséges megoldásainak vizsgálata volt. A ROS által nyújtott ROS-serial lehetséges általános integrációs megoldás széles körben alkalmazható alacsony szintű hardverek integrálására, de korlátai miatt nem képes nagy méretű üzenetcsomagok kezelésére.

Az eredmények alapján a legjobban működő kommunikációs lehetőség UART protokollra épülő saját protokoll készítése. Ezen megoldással a rendszer 1ms mintavételezési periódussal robusztusan működik. Jelen pillanatban az üzenet hossza a központi számítógép és az FPGA modulok között maximum 200 byte lehet, ami szükség esetén kiterjeszhető. Az FPGA-ban levő modulokat Matlab/System Generator használatával valósítottam meg pl: PWM, UART kommunikációs modul.

A kifejlesztett saját protokoll és az FPGA előnyei, hogy az FPGA-ba beélező adatok feldolgozását a hardver végzi, így könnyítve a processzor leterheltségén. Abban az esetben, ha a kommunikációs protokollt a uBlaze soft core processzor végezte, a sok megszakítás miatt a rendszer használhatatlan volt már 20ms mintavételezsnél.

Az FPGA kommunikációs modulba beépített direkt utasítás értelmező képes a PWM kimenetet és a szabályzókat leállítani abban az esetben, ha a kommunikáció megszakad vagy ha a központi egység ezt kéri. A HeartBeat biztonsági megoldás segítségével figyeli a ROS rendszerben futó összes létfontóságú Node működését, beleértve a FPGA-k működését is. A HearBeat node periodikusan küld engedélyező jeleket és vár is a létfontóságú moduloktól.

A keret figyelését és a speciális karakterek értelmezését a FPGA hardvere végzi, a beérkező adatokat a processzorral közös memóriába írja bele. Ha a csomag teljes egészében a memóriában van akkor, kéri a processzort, hogy szakítsa meg a munkáját és dolgozza fel a csomagot.

A robot dőléseit és gyorsulásait IMU modul segítségével mérem.

A 2D LIDAR szenzor segítségével mérem a robot távolságát a környezetében levő tárgyakhoz képest, 8m távolságig,  $360^\circ$ , 2mm pontossággal. A LIDAR mérésein felhasználva, HectorSlam segítségével térképet készítetek a környezetről, amelyben a robot képes lokalizálnia magát. A LIDAR alapvetően jól használható a térképezési feladok elvezésére, abban az esetben, ha a környezet tartalmaz üvegfalat és nincs más stabilan mérhető tárgy

---

a közelben, zajok jelennek meg a térképezésben.

### 0.0.1. Mérések a robottal

A robottal méréseket végeztünk különböző terepviszonyok között, körpályán való mozgást és a helyben forgást tanulmányozva. A robot lépcsőn felfele való mozgását tanulmányozva az eredmény az, hogy a lépcsőt hatékonyabb súlyponttal előre, és nem merőlegesen kell megközelíteni. A homokos talajon való mozgás esetén a kerekek kis fordulatszáma hatékonyabb, mert így a kerekek nem ássák be magukat a homokba.

### 0.0.2. Pozíció szabályzása

A pozíció szabályzása a robotnak két részfeladatra osztható: fordulás a cél irányába, és csökkentsd a távolságot a cél és az aktuális pozíció között. Az irány esetében szöget szabályzunk, míg a távolság esetében sebességet. Mindkét mennyiséget esetében PI típusú szabályzót alkalmaztam, a távolság szabályzó bemenetét súlyozva a szögpozíció szabályzó hibájának a fordított értékével.

Kulcsszavak: SSMR,FPGA,ROS,Robot,SLAM

# Tartalomjegyzék

0.0.1. Mérések a robottal . . . . .	2
0.0.2. Pozíció szabályzása . . . . .	2
1. Bevezető	8
2. Bibliográfiai tanulmány	9
2.1. Robotok . . . . .	9
2.2. Mobilis Robotok Helyváltoztatása . . . . .	9
2.3. Kültéri mobilis robotok . . . . .	10
2.4. Robot Operációs Rendszer . . . . .	12
2.4.1. Új robot integrációja ROS-hoz . . . . .	14
2.5. Mobilis robotok modellezése . . . . .	16
2.5.1. Merőleges Nyomóerő (N) . . . . .	16
2.5.2. Súlypont (X,Y) komponensének a meghatározása . . . . .	19
2.5.3. Súlypont meghatározása mérésekkel . . . . .	20
2.5.4. Kerék Dinamikája . . . . .	20
2.6. Kinematikai Modell . . . . .	21
2.7. Dinamikai Modell . . . . .	23
2.8. Robot Platform Sebesség Szabályzása . . . . .	26
2.8.1. Előírt nyomatékkal . . . . .	26
2.8.2. Elírt kerekszögsebessegekkel . . . . .	27
3. A rendszer implementálása	28
3.1. Robot felépítése . . . . .	28
3.2. Alacsony szintű modulok . . . . .	29
3.2.1. Microblaze szoftvere . . . . .	37
3.2.2. FPGA és UART alapú kommunikációs protokoll . . . . .	39
3.2.3. Paraméterek FPGA modul . . . . .	40
3.2.4. Kommunikáció sebessége . . . . .	41
3.2.5. Biztonsági megoldások . . . . .	41
3.3. ROS . . . . .	43
3.3.1. Üzenet típusok (.msg) . . . . .	44
3.3.2. FPGA kommunikációs modul ROS oldali integració . . . . .	47
3.3.3. Előírt értékek . . . . .	49
3.3.4. Vonatkoztatási Rendszerek . . . . .	49
3.4. Kerekek Pid Szabalyzo hangolas . . . . .	52
3.4.1. Kisebbik fokozatban . . . . .	53
3.5. Pályakövetési feladatok . . . . .	55
3.6. Mérések . . . . .	58
3.6.1. Differenciális Forgás Vízszintes Talajon . . . . .	58
3.6.2. Féloldali kerekek blokkolva kavicsos talajon . . . . .	58

3.6.3.	Kavicsos talajon helyben forgás . . . . .	61
3.6.4.	Márvány padlón helyben forgás . . . . .	63
3.6.5.	Kavicsos talajon körpályán 50/15 . . . . .	66
3.6.6.	Kavicsos talajon körpályán 50/25 . . . . .	69
3.6.7.	Körpályák összehasonlítása . . . . .	71
3.6.8.	Homokos Lejtő . . . . .	72
3.6.9.	Lepcson . . . . .	74
3.6.10.	Ismeretlen terep térképezése és robot lokalizálása (SLAM) . . . . .	78
4.	Eredmények Kiértékelése	80
4.1.	Megvalósítások . . . . .	80
4.2.	Hasonló rendszerekkel való összehasonlítás . . . . .	81
4.3.	További fejlesztési irányok . . . . .	82

## Ábrák jegyzéke

2.1.	iRobo 510 lánctalpas packbootForrás: . . . . .	10
2.2.	Négykerekű mobilis platform.Forrás: . . . . .	11
2.3.	ecorobotix mezőgazdasági robotForrás: . . . . .	11
2.4.	Spirit nevű Mars járó robot.Forrás: . . . . .	12
2.5.	ROS kommunikációs mechanizmus szervizhívásokraForrás: . . . . .	13
2.6.	ROS kommunikációs mechanizmus adatfolyamokraForrás: . . . . .	13
2.7.	Ros Control modulokForrás: . . . . .	15
2.8.	Merőleges nyomóerő a talajra 4W – <i>SSMR</i> típusú robot esetében. . . . .	17
2.9.	4W – <i>SSMR</i> típusú robot lejtőn felfele oldal nézetből. . . . .	17
2.10.	4W – <i>SSMR</i> típusú robot lejtőn első nézetből. . . . .	18
2.13.	Kinematikai modell az 4W – <i>SSMR</i> típusú robotnak. . . . .	22
2.14.	Kinematikai modell az <i>SSMR</i> típusú <i>MR</i> robotnak. . . . .	24
2.15.	Kinematikai modell az <i>SSMR</i> típusú <i>MR</i> robotnak. . . . .	26
2.16.	Kinematikai modell az <i>SSMR</i> típusú <i>MR</i> robotnak. . . . .	28
3.1.	CmodA7 FPGA-ban kialakított architektúra amely a szenzorok és motor hajtások jeleinek feldolgozását és előállítását valósítja meg . . . . .	29
3.2.	FPGA ban megvalósított kommunikációs modul a ?? leírt protokoll alapján. . . . .	31
3.3.	FPGA hardver/MicroBlaze processzor és ROS node közti kommunikáció megvalósítása UART protokoll alapján . . . . .	32
3.4.	FPGA ban megvalósított szoftprocesszor rendszer, legfelső négyzete. . . . .	33
3.5.	FPGA-ban megvalósított szabályzok A és B . . . . .	35
3.6.	FPGA controller modul . . . . .	36
3.7.	MicroBlaze processzoron futó szoftver diagramja . . . . .	38
3.8.	FPGA kommunikacios protokol altalanos csomag szerkezet . . . . .	39
3.9.	Robothoz csatlakozás a Wifi-n keresztül. . . . .	43
3.10.	ROS graph . . . . .	46
3.11.	ROS integrálása Uart protokolhoz. . . . .	48

3.12. A megvalósított robot VNR-k közti reláció . . . . .	51
3.13. Nagy fokozat Hammerstein-Wiener becsült modell válasza és a mért értékek. . . . .	53
3.14. Kis fokozat Hammerstein-Wiener becsült modell válasza és a mért értékek. . . . .	54
3.15. Robot pozíció szemlélődése . . . . .	55
3.16. Robot pozíció szemlélődése . . . . .	57
3.17. <i>SSMR – 4W</i> típusú robot pozíciója, X és Y tengelyekre bontva, keréksebességek BL=FL=0 és a FR=BR= 50°/s . . . . .	58
3.18. <i>SSMR – 4W</i> típusú robot által leírt pálya, kerekesebességek BL=FL=0 és a FR=BR= 50°/s . . . . .	59
3.19. <i>SSMR – 4W</i> típusú robot orientációja,ha a kerékszögsebességek BL=FL=0 és a FR=BR=50°/s . . . . .	59
3.20. <i>SSMR – 4W</i> típusú robot fordulási szögsebessége Giroszkóp és LIDAR által mért értékek, ha a kerékszögsebességek BL=FL=0 és a FR=BR= 50°/s . . . . .	60
3.21. <i>SSMR – 4W</i> típusú robot súlypontjának sebessége a globális VNR-ben, X és Y tengelyekre bontva, ha a kerékszögsebességek BL=FL=0 és a FR=BR= 50°/s . . . . .	60
3.22. <i>SSMR – 4W</i> típusú robot mozgása, tengelyekre bontva, ha a kerék szögsebességek BL=FL=50°/s és a FR=BR=-50°/s . . . . .	61
3.23. <i>SSMR – 4W</i> típusú robot mozgása, ha a kerék szögsebességek BL=FL=50° °/s és a FR=BR=-50°/s . . . . .	62
3.24. <i>SSMR – 4W</i> típusú robot által leírt pálya, ha a kerék szögsebességek BL=FL=50° °/s és a FR=BR=-50°/s . . . . .	62
3.25. <i>SSMR – 4W</i> típusú robot orientációja, ha a kerék szögsebességek BL=FL=50° °/s és a FR=BR=-50°/s . . . . .	63
3.26. <i>SSMR – 4W</i> típusú robot fordulási szögsebessége Z tengely körül, ha a kerék szögsebességek BL=FL=50°/s és a FR=BR=-50°/s . . . . .	63
3.27. <i>SSMR – 4W</i> típusú robot mozgása, tengelyekre bontva, ha a kerék szögsebességek BL=FL=70°/s és a FR=BR=-70°/s . . . . .	64
3.28. <i>SSMR – 4W</i> típusú robot mozgása, ha a kerék szögsebességek BL=FL=70° °/s és a FR=BR=-70°/s . . . . .	65
3.29. <i>SSMR – 4W</i> típusú robot által leírt pálya, ha a kerék szögsebességek BL=FL=70° °/s és a FR=BR=-70°/s . . . . .	65
3.30. <i>SSMR – 4W</i> típusú robot orientációja, ha a kerék szögsebességek BL=FL=70° °/s és a FR=BR=-70°/s . . . . .	66
3.31. <i>SSMR – 4W</i> típusú robot fordulási szögsebessége Z tengely körül, ha a kerék szögsebességek BL=FL=70°/s és a FR=BR=-70°/s . . . . .	66
3.32. <i>SSMR – 4W</i> típusú robot mozgása, tengelyekre bontva, keréksebességek BL=FL=50°/s és a FR=BR=15°/s . . . . .	67
3.33. <i>SSMR – 4W</i> típusú robot mozgása, tengelyekre bontva, kerekészögsebességek BL=FL=50°/s és a FR=BR=15°/s . . . . .	67
3.34. <i>SSMR – 4W</i> típusú robot által leírt pálya, kerekészögsebességek BL=FL=50° °/s és a FR=BR=15°/s . . . . .	68
3.35. <i>SSMR – 4W</i> típusú robot fordulási szögsebessége, kerekészögsebességek BL=FL=50° °/s és a FR=BR=15°/s . . . . .	68
3.36. <i>SSMR – 4W</i> típusú robot egyenesvonalú sebességei, kerekészögsebességek BL=FL=50°/s és a FR=BR=15°/s . . . . .	69
3.37. <i>SSMR – 4W</i> típusú robot által leírt pálya, ha kerékszögsebességek BL=FL=25° °/s és a FR=BR=50°/s . . . . .	69

3.38. <i>SSMR</i> – 4W típusú robot motorvezérlő jelei, ha kerékszögsebességek $BL=FL=25^\circ$ ◦/s és a $FR=BR=50^\circ$ /s . . . . .	70
3.39. <i>SSMR</i> – 4W típusú robot mozgása, tengelyekre bontva, ha kerékszögsebességek $BL=FL=25^\circ$ /s és a $FR=BR=50^\circ$ /s . . . . .	70
3.40. <i>SSMR</i> – 4W típusú robot fordulási szögsebessége, ha kerékszögsebességek $BL=FL=25^\circ$ /s és a $FR=BR=50^\circ$ /s . . . . .	71
3.41. Kulombozo korpalyak . . . . .	72
3.42. Homokbat súlyet kerek $45^\circ$ lejtön, 10 cm mélyen,elakadt robot a lejtőn 0.5m megtett ut után . . . . .	73
3.43. Homokos és kavicsos lejtőn felfele mozgás . . . . .	73
3.44. Lépcsőn lefelé mozgás. . . . .	74
3.45. Lépcsőn lefelé mozgás, három lépcsőfok. . . . .	75
3.46. Lépcsőn felfele mozgás, kétlépcsőfok. . . . .	75
3.47. Lépcsőn felfele mozgás . . . . .	76
3.48. Lépcsőn $60^\circ$ irányból felfele haladva 8 s. . . . .	77
3.49. Lépcsőn felfele mozgás $60^\circ$ szöget bezárva a lépcsőfokokkal. . . . .	77
3.50. Térkép készítése miközben távirányítással halad a robot. . . . .	78
3.51. Térkép készítése Sapientia Aula. . . . .	79

## Jelölősek

$\theta$  A robot z és x tengelyek által bezárt szög *rad*

${}^nS_{ik}$  Valamilyen § fizikai mennyiség a  $n$  vonatkoztatási rendszerben, az *in* indexek a mennyiség helyet jelöli.

$F$  erők *N*

$i$   $i \in (L, R)$  ahol L és R a robot bal és jobb oldali kerekeit jelöli.

$k$   $k \in (F, B)$  ahol F és B a robot első és hátulsó oldali kerekeit jelöli.

$n$   $n \in (G, R)$  ahol G és R világ és robot vonatkoztatási rendszer.

$q_2$  A *MR* állapot vektora  $q \in \mathbb{R}^3$   $q = [X \ Y \ \theta]$  ahol a X és Y a robot x illetve y tengelyen mért pocizója, és a  $\theta$  a robot orientációja a globális X tengelyhez képest

$X$  A robot x tengelyen mért pozíciója *m*

$Y$  A robot y tengelyen mért pozíciója *m*



Analóg bemenet.

- 
- Digitális bemenet.
  - Digitális tömbbites multiplexer.
  - Digitális kimenet.
  - Megszakítás a processzornak.
  - Multiplexert szelektáló regiszter.
  - Micriblaze processzor címtartományában levo regiszter.

#### Rövidítsek

4W-SSMR 4Wheel Skid-Steered Mobile Robot - 4 kerekű csúszókormányzású mobilis robot.

AMCL Adaptíve Monte Carlo Localization

AMR Autonomus Mobilis Robot

COG Center Of Geometry

COM Center Of Mass

CTM Computed Torque Method - Kiszámított Nyomatékok módszere

DDV Diferential Driven Vehicles

DSC Distributed Control System

GUID Globally Unique Identifier -globálisan egyedi azonosító

HLC Angol elnevezésből származó rövidítése a magas szintű szabályzónak (High Level Controller)

ICR Instantaneous Center of Rotation - pillanatnyi forgás-középpont

IMO in my opinion, saját vélemeny

IMU Inertial Measurement Unit gravitációs gyorsulást és szög elfordulást mérő szenzor.

LIDAR Light Detection and Ranging, 2D vagy 3D lézeres kép alkotó műszer

MR Mobilis Robot

OMP Outdor Mobile Robot - kültéri mobilis robot

ROS Robot Operation System

SMC Sliding Mode Control - Csuszás alapú szabályzás

SSMR Skid-Steered Mobile Robot - csúszókormányzású mobilis robot.

UART két vezetékes(Tx,Rx) két irányú kommunikáció

URDF Unified Robot Description Format, XML alapú robotleíró nyelv

VFO Vector Field Orientation - Vektor alapú iránymeghatározás

VNR Vonatkoztatási Rendszer

## fejezet 1

### Bevezető

A robotokat széles körben alkalmazzák, egyre több feladatra és most már az átlag emberek életében is. Néhány nagyobb vállat mint pl: ABB, Kuka nagy területet foglalt el az iparban robot karok gyártásában. Emellett egyre több kisebb vállalat jelent meg, amelyek háztartási vagy fél katonai eszközöket gyártanak pl.: iRobo. A mezőgazdaságban is próbálnak alkalmazni autonóm robotokat pl.: echorobotics amelyek segítségével hatóanyag mentes élelmiszereket állíthatnak elő.

A dolgozat célja hogy felderítse az aktuális legmodernebb technikákat amelyeket robotokon alkalmaznak, és ezeket alkalmazza egy kültéri mobilis robot megépítése során. Az előzőleg már egy hasonló rendszer kivitelezésekor elteszíték modulok továbbfejlesztése és integrálása az új rendszerbe. A robot mozgása négy csiga áttétel és négy DC motor segítségével valósul meg. A motrok szögsebességét és felvett arámokat merjük, inkrementális szenzor és elektromágneses jelenségre alapuló aramerő modul segítségével.

A beavatkozás feszültség formájában történik, amelyet H-híddal állítunk elő azáltal hogy PWM jelet generálunk.

Az inkrementális és aramerő szenzorok jeleit FPGA alapú fejlesztőlapokkal oldjuk meg. A roboton helyet foglal egy kis méretű számítógép is amely USB vezetékeken keresztül csatlakozik az FPGA lapokhoz és a roboton található más szenzorokhoz pl: LIDAR, IMU, GPS.. Az FPGA modulok segítségével valosulnak meg a kerekek szögsebesseg szabalyzása, a dolgozatban targyalásra kerül az egyedi hardver integracioja ROS keretrendszerben, amelyet egy koztes nodal valositunk meg. A robot a csiga áttelek miatt nagy nyomatékokat tud előállítani, ami a mozgási sebesség rovására vált. A robot kepés 0.3 m/s sebességél mozogni előre, és 20 °/s forgási sebességet generálni súlypontja körül.

A dolgozat célja az elkészített kültéri mobilis robottal meréseket végezni különböző terepviszonyok között és azokat összehasonlítási. A terepviszonyok mit pl: füves, kavicsos, aszfalt, márvány stb. A meréseket szeretnek összehasonlítani a terepviszonyok függvényében: a robot mozgását és a szükséges energia nagyságát, a környezetében okozott behatások.

A robot SLAM algoritmus dolgozza fel a LIDAR által mért adatokat és egy 2D térképet állít elő, amelyen kepés egyidőben behatárolnia pozícióját és a térképet építeni is. Ezen meglevő programok használatával kültéri meréseket végezni, térképeket készíteni és majd egy előírt pályán végigmenni, ezen terepeken a térképet használva. Az esetleges zavao tenyezok felderites amelyek befolyasolják a terkepeses es a lokalizacio pontosagat.

## fejezet 2

# Bibliográfiai tanulmány

### 2.1. Robotok

Az olyan gépek, amelyeket arra terveztek, hogy bizonyos feladatokat automatikusan elvégezzenek gyorsabban és pontosabban mit az emberek. Manapság már sok típusú robot létezik, ezek közül néhány repül, földön gurul vagy mászik, de léteznek már emberhez hasonlóak is.

A robotok legelőször a második világháború alatt jelentek meg mint irányított bombák. A háború után W.Grey Walter neurológus fejlesztett ki egy kiméretű robotot (Elmer and Elsie) mely fényszennorral, és nyomásszenzorral volt ellátva. 1961-1963 Johns Hopkins Beast robot képes volt a fal menten végigmenni és megtálalni a töltőállomást. 1970 -ben Shakey robot képes volt kamera segítségével egy vonalat követni. A 1990 után a fejlesztések felgyorsultak és azóta már robot jár a naprendszer más bolygóján is.

Ezen dolgozat csak a *MR* típusú robotokkal fog részletesebben kitérni. A *MR* típusú robotok képesek a saját pozíciójukat megváltoztatni a környezetükhez képest. Az *AMR* típusú robotok képesek saját környezetük felderítésére ismeretlen környezetben is különböző szenzorokat használva melyekkel képesek a környezetük paramétereinek a megmérésére. Az *AMR* típusú robotokat manapság leginkább tároló helyiségekben használjak, ahol anyagokat, palettákat vagy különböző dolgokat kel egyik helyszínről a másikra szállítani (például Mobile Industrial Robots ApS vállalat fejlesztései).

### 2.2. Mobilis Robotok Helyváltoztatása

A *MR* kerekekkel, hernyótalpakkal, vagy lábakkal képesek helyüket megváltoztatni. A leginkább használatos és robusztusnak bizonyult kültéri terepen a kerekek és a hernyótalpak.

Legismertebb robot felépítések: packboot pl: iRobo510,  
négy hagyományos kerék pl: husky, négy kormányozható kerek, hat kerék, amiből négy kormányozható pl.: Curiosity Mars Rover.

---

Az utóbbi években a kerek fogalma is újraértelmeződött, a hagyományos kerekek pl: RHex Rough-Terrain amely a kerek és a lábak keveréke.

### 2.3. Kültéri mobilis robotok

#### iRobo510

A robot külterén és belterén is egyaránt használható, felderítésekre és kisebb beavatkozásra alkalmas a manipulátor karát használva. A robotot 2007-ben dobták piacra. A katonaság használta bombák hatástalanítására vagy felderítésre.



2.1. ábra. iRobo 510 lánctalpas packbootForrás:

Tulajdonság	Mértékegység
Méretek	X 0.521 m
	Y 0.686 m
	Z 0.178 m
Önsúly	10.89 kg
Max Sebesség	9.3 km/h
Gyorsulásmérő és Giroszkóp	Igen

#### Husky

Nagy teherbírású kültéri mobilis robot, leginkább kutatási célokra használják, nagy keréknyomatéka miatt nehéz terepen is jól boldogul. Nagy felbontású inkrementális szenzorokkal szerelték fel, alacsony mozgási sebességre is képes. Működése 3 óra átlagos használat mellett, támogatja teljes mértékben ROS-t.



2.2. ábra. Négykerekű mobilis platform. Forrás:

Tulajdonság	Mértékegység		
Méretek	X	0.99	m
	Y	0.67	m
	Z	0.39	m
Önsúly	50 + 75	kg	
Max Sebesség	3.6	km/h	
Gyorsulásmérő és Giroszkóp	Igen		

## Ecorobotix

Mezőgazdaság számára kifejlesztett robot, négy kerékkel rendelkezik amelyek közül kettőt motor hajt, és a másik kettő ön-beálló kerék. Kamera segítségével ismeri fel a növényeket és a pozíciójukat, és ha szükséges akkor be is avatkozik.

A lokalizációra egy nagy pontosságú GPS RTS alkalmaznak.



2.3. ábra. ecorobotix mezőgazdasági robot. Forrás:

Tulajdonság	Mértékegység	
Méretek	X	2.2 m
	Y	1.70 m
	Z	1.30 m
Önsúly	130	kg
Max Sebesség	1.4	km/h
Gyorsulásmérő és Giroszkóp	Igen	

## Spirit

Marsi körülményekre tervezett robot 6 kerekkel rendelkezett, amelyből 4 kormányzott négy sztereó kamerával ellátva, 30° lejtőt volt képes megmászni. Működési ideje 6 év és 2 hónap volt a Marson, a végen homokba süllyedve egy sziklára akadva érte a marsi tél, amely ahhoz vezetett, hogy a nap elemei nem szolgáltattak elegendő energiát és így végleg leállt és kihűlt.



2.4. ábra. Spirit nevű Mars járó robot. Forrás:

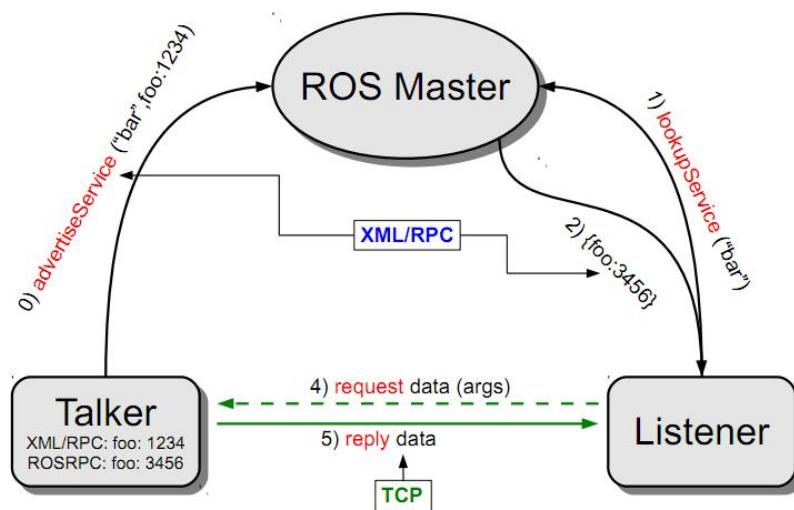
Tulajdonság	Mértékegység	
Méretek	X	1.6 m
	Y	2.3 m
	Z	1.5 m
Önsúly	35 (felszereléssel 180)	kg
Max Sebesség	0.05 (avg 0.01)	km/h
Gyorsulásmérő és Giroszkóp	Igen	

## 2.4. Robot Operációs Rendszer

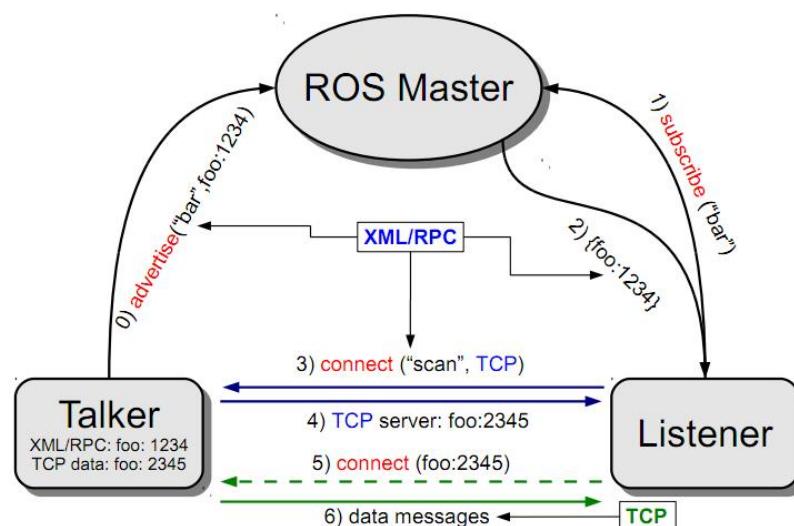
A ROS 2007-ben jelent meg, gyorsan elterjedt az egész világon, manapság szinte minden robotokkal foglalkozó cég termékeit kapcsolódnak a ROS-hoz.

A ROS működéséhez szükséges egy gazda operációs rendszer a UNIX vagy Windows alapú amelyen a központi node fut (rosmaster). A master kezeli a többi node közti kommunikációt, paramétereket, sérvízhívásokat.

A kommunikáció a nodok között TCP protokolra épül, amely XML/PRC teknoloiat használ, RPC távoli eljárás hívást jelent [8]. minden node jelzi a rosmasternek milyen adatokat szeretne megosztani ezeket az advertise függvényhívások jelzik. A nodok ugyanakkor feliratkozhatnak ezeket a subscribe függvényekkel valósíthatjuk meg pl.: [23]. A szervízhívások 5 lépéses folyamatból állnak, látható a ábra 2.4.5 valamint az adatfolyamok 8 lepésből ábra 2.4.6.



2.5. ábra. ROS kommunikációs mechanizmus szervízhívásokraForrás:



2.6. ábra. ROS kommunikációs mechanizmus adatfolyamokraForrás:

A [12] [14] segítségével megalapozhatjuk a tudásunkat. Számos előnyel rendelkezik a ROS használata egy új robot fejlesztésében mert már elkészített eszközöket használhatunk

---

pl: SLAM<sup>1</sup>, *AMCL*, vagy előre elkészített eszközök segítenek a fejlesztésben pl: Rviz<sup>2</sup>, Gazebo<sup>3</sup>, interfészt biztosít a szenzoroknak és beavatkozóknak pl: *LIDAR*, *IMU*.

A [12] említést tesz arról hogy más hasonló keret-rendszerekhez képest a ROS stabilabb pl: ha egy modulban futás-idejű hiba lép fel az nem terjed ki a többi csomópontra. Egy-szerűbb fejlesztést lehetséges azáltal hogy kisebb modulokat fejlesztünk és nem egy nagy több szálon futó kódot. Annak ellenére hogy a forrás-kódja nyílt a keret-rendszernek nagyon jó szupportja van, rengeteg fórumon keresztül kaphatunk megoldásokat az esetleges hibákra. Több technológiát képes összekapcsolni mint pl.: tensorflow, matlab, simulink, opencv, V-Rep...

Hátrányai között említhető a Gazebo szimulátor mert a használata nem egyszerű el-lentében a V-Rep<sup>4</sup> programmal. A robot modellezése nem egyszerű dolog, *URDF* fájl szükséges hozzá, csak SolodWork<sup>5</sup> biztosít lehetőséget arra hogy modellt exportálhassuk.

#### 2.4.1. Új robot integrációja ROS-hoz

Egy új robot integrációja során meg kell vizsgálni hogy milyen mérési adatok állnak rendel-kezésünkre alacsony szinten, a rotációs csukló paraméterek lehetnek pl: szög pozíció, szög sebesség, kifejtett, nyomaték, ezeket a paramétereket mérhetjük, illetve referenciaiként is előírhatjuk.

Az integrációra több megoldás is lehetséges:

- (a) ROS Serialon keresztül.
- (b) ROS control használata.
- (c) Osztott Rendszer.

##### Ros Serial

Egy megoldás a hardver integrációjára a ROS Serial amely UART, vagy TCP protokollra épülő, soros vagy hálózati kommunikációt használva. Korlátai miatt [7] nem képes nagy méretű üzentek használatára, valamit a nodok száma is korlátozott lásd. Szükséges a ROS csomagok használata a hardveren ami nem minden előnyös, függőség alakul ki a hardverfejlesztése közben a szoftver irányába.

A [1] cikkben egy arduino típusú fejlesztő lappal valósítja meg a robot alacsony szintű szabályzását, megemlíti hogy a rosserial-t nem tudja használni a limitációk miatt. A feldolgozó oldalon elkészít egy saját szoftveres modult amely megvalósítja a ROS és a hardver közti kapcsolatot. A kommunikációra soros *UART* protokollt használ.

A [6] könyv 8-ik fejezetben leírja hogyan lehet használni a rosserial-t, arduino valamint Raspberry Pi fejlesztő lapokon de viszont nem tesz említést a hátrányairól.

##### Ros Control

A ros controller [4] használatával összeköthetjük az alacsony szintű hardvert a ROS keret-rendszerben fejlesztett modulokkal, implementálva [17] a *hardware\_interface::RobotHW* interfészt és létrehozva minden egyes rotációs csuklónak egy *hardware\_interface::JointStateHandle*-t. A ábra 2.4.7 látható *read()* és *write()* függvényeken

<sup>1</sup> Simultaneous Localization and Mapping  
egyidejű térképezés és lokalizáció

<sup>2</sup> ROS környezet vizualizációs eszköze

<sup>3</sup> ROS környezet szimulációs eszköze

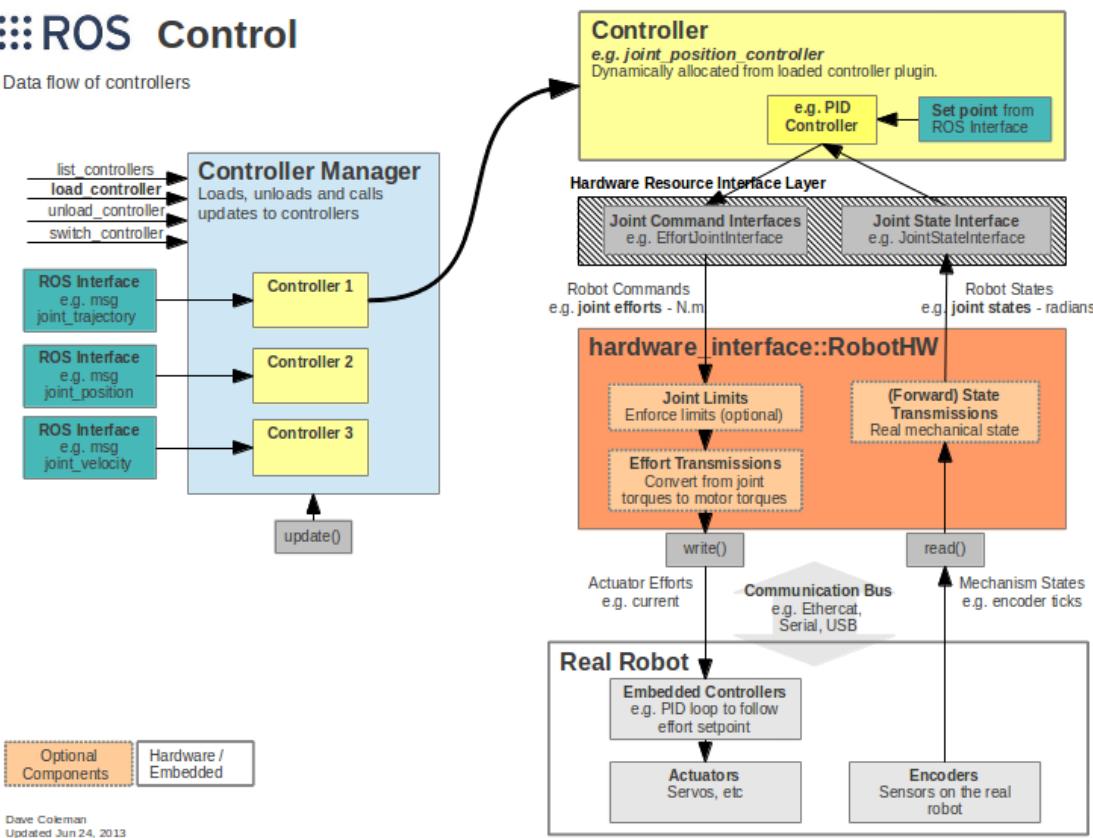
<sup>4</sup> Robot szimulációs szoftver amely támogatja  
a ROS-t

<sup>5</sup> 3D modellező szoftver

keresztül kell megvalósítani a kommunikációt a hardverrel, ez történhet hálózaton vagy soros porton keresztül.

## ROS Control

Data flow of controllers



2.7. ábra. Ros Control modulokForrás:

A ábra 2.4.7 ábrán látható az interfészek kapcsolatai és a fontosabb függvényhívások. A hardverrel való integrációt write() és read() függvényhívásokkal valósul meg, ebben a két függvényben kell elkészíteni a programokat amelyek képesek kiolvasni és beírni az eszközben a kívánt jeleket. Itt használhatunk több típusú alacsonyszintű kommunikációs protokolt pl: TCP,UART.., vagy bármilyen interfészt amit a gazga operációs rendszer elismer.

## Osztott Rendszer

A [19] cikkben leír egy megoldást arra hogyan lehetne smart eszközöként tekinteni a szenzorokra és beavatkozó modulokra. Osztott rendszert *DSC* -t alkalmaz, ahol minden szenzornak saját mestere van, ezáltal minden node független lesz a hálózaton. Enelkül a hálózat konfigurációját teljes mértékben ismerni kell minden nodenak a IP címét, de ezzel a megoldással futásidőben módosíthatjuk a hálózatot, *GUID* -t használ a új maszter bejelentésre a hálózaton, valamit ezzel is oldja meg az információk áramlását is. A kommunikációt a mesterek között ROS MultiPeer Architecture (RMPA) nevezet architektúrával oldja meg.

---

Más rendszerekhez képest kétszer jobb időkésést valósított meg. Ami a hátránya, a szenzorok mellett egy mcu is szükséges amely kepés egy operációs rendszert futtatni és egy rós masztert. Valóban robusztusabb modulárisabb lesz a rendszer ezáltal de drágább is. Kisebb rendszereknél, inkább hátrány mint előny, de viszont komplex, térben nagy területet lefedő rendszernél előnyös.

Egy másik hasonló megoldás ahol több ROS masztert lehet összekötni és felügyelni a multimaster\_fkie megoldja a futásidőben való új maszter szinkronizációját, a topikok és a szervizek kezelését is.

## 2.5. Mobilis robotok modellezése

### Négy kerekű modell

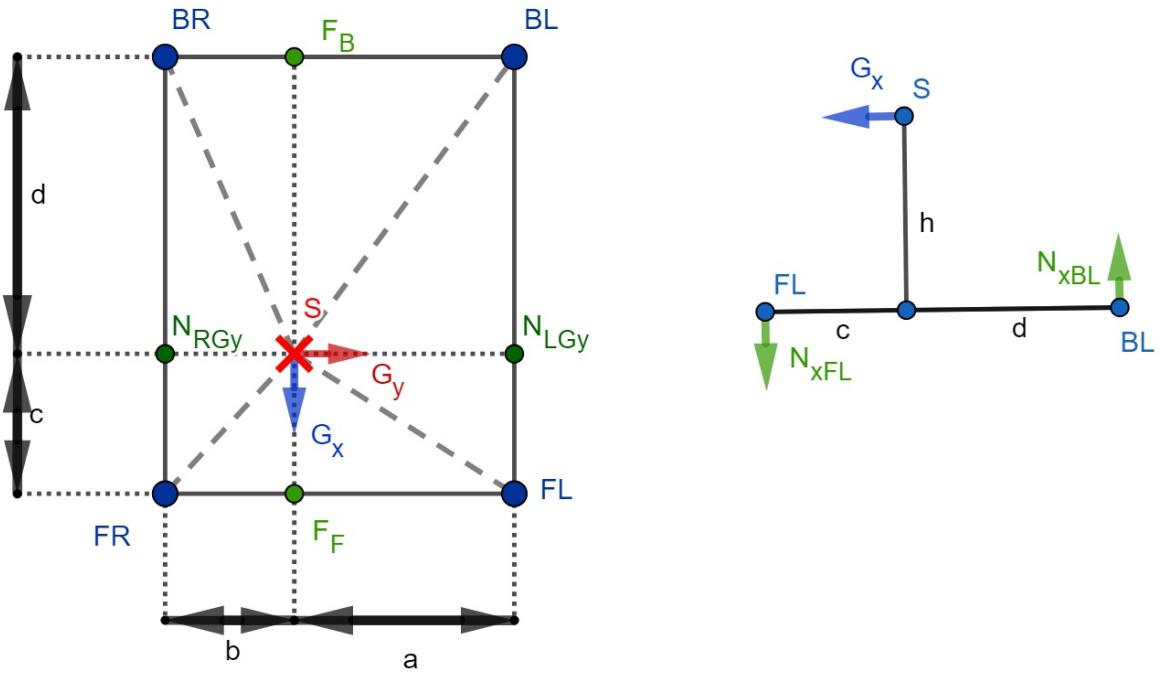
A mobilis robotok leginkább kerekekkel oldják meg a helyváltoztatásukat. Egy fontos probléma ezekkel a robotokkal az hogy milyen kölcsönhatások lepnek fel a kerék és a talaj között [21] [2] [3], és ezeket az erőket hogyan lehet modellezni. A [21] cikkben kidolgoz egy modellt amely segítségével képes meghatározni egy négykerekű robot pozícióját a kerekek forgási sebességéét felhasználva. A fent említett irodalmakban a **SSMR** típusú **OMP** vizsgálnak módélezés és pályakövetés szempontjából. Annak függvényében hogy a MR-t mozgató motorokat tekintve a következő variánsok lehetségesek: 4 kerék - 4 motor, 4 kerék - 2 motor, azonos oldalon levő kerekek összecsatolva fogas-szíjjal ez második megoldás egyszerűbb kevesebb szenzort és hajtó motort igényel. A [21] [3] irodalmakban a **HLC** sebesség referencia jelet ír elő a kerekeken, [2] a cikkben nyomatéket ír elő amelyet az alacsony szintű szabályozóknak. Az **ICR** meghatározásával választotta. Több dolgot is feltételez: a robot forgásközpontja a robot középpontjában van, a robot azonos oldalán levő kerekek ugyanazzal a szögsebességgel forognak, a négy kerék minden érintkezik a talajjal és méretben is megegyeznek.

A [2] kifejezetten a SSMR jól ismert a robusztus félépítése miatt, nagyon jól alkalmazható terepen. Általában **DDV** mert a fordulást azáltal oldják meg hogy a jobb és bal oldali kerekek különböző sebességgel vagy különböző kifejtett nyomatéket fejtenek ki a talajra. A [2] cikkben alkalmazott technológiáiak: **VFO CTM** a robot kerekeit szabályzó motoroknak nyomaték van előírva amit követniük kell.

#### 2.5.1. Merőleges Nyomóerő (N)

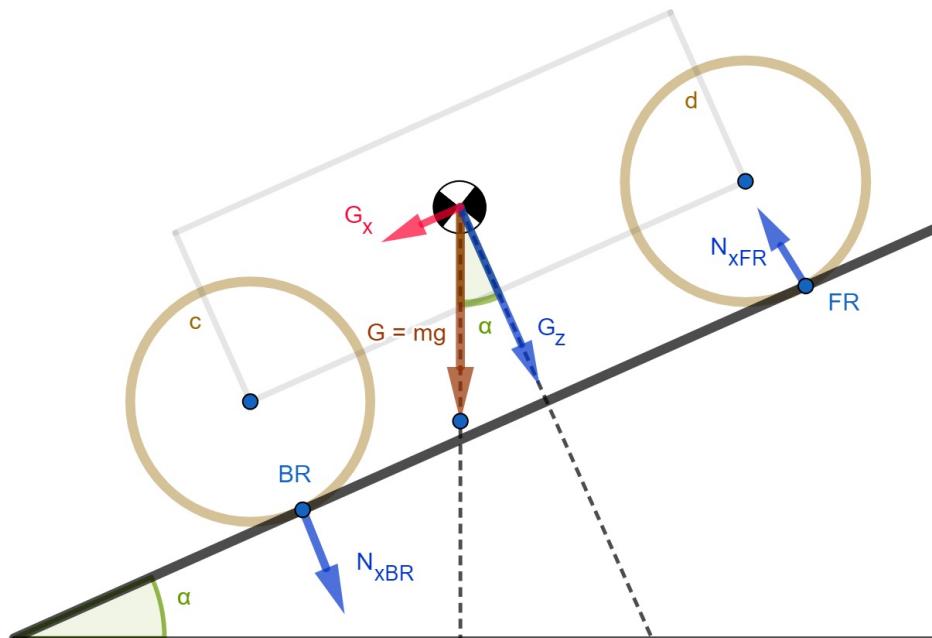
A kerekek és a talaj egy pontban érintkeznek, ezeket a pontokat jelölje a BR,BL,FL,FR a ábra 2.5.8 ábrán. Jelölje S a robot súlypontkát, G - a súlyából származó erők a robot alapjához rendelt kordinát rendszerben felbontva a három tengely menten, N a merőleges nyomóerő a talajra az adott pontban.

A robot fizikai méretei:  $a+b = 0.66\text{m}$  és  $c+b = 0.58\text{m}$ ;

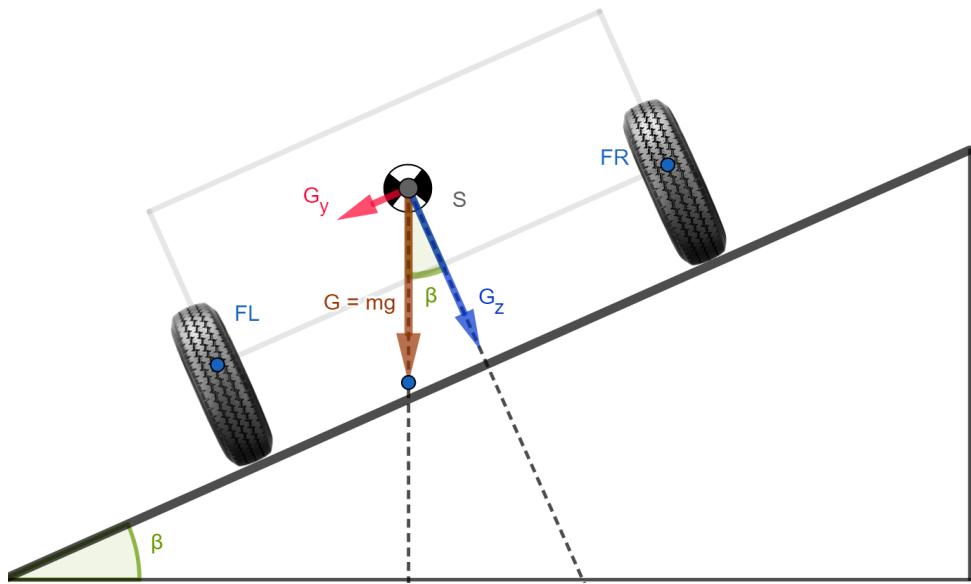


2.8. ábra. Merőleges nyomóerő a talajra 4W – SSMR típusú robot esetében.

Jelölje az  $\alpha$  szög ha a lejtőn felfele halad a robotre fig:S MR4WLejtoOldalrol,  $\beta$  ha a robot a lejtőn oldalra halad ábra 2.5.10. Ha ismerjük a robot súlypontjának a pozícióját mindenkor tengelyen akkor kicsuszamolhatjuk a kerekek merőleges nyomóerejét a talajra a következő módszerrel.



2.9. ábra. 4W – SSMR típusú robot lejtőn felfele oldal nézetből.



2.10. ábra. 4W – SSMR típusú robot lejtőn első nézetből.

Egy test nyugalomban van ha a rá ható erők eredője és a forgatónyomatékok eredője zero, ismerve a súlypont pozíciójának a koordinátáit a robot VNR-be akkor az 2.1 egyenlettel meghatározzuk a  $G_x$  erő által létrehozott nyomóerőket a  $N_F$  és  $N_B$  pontokban.

$$N_{FGx} = \frac{hG_x}{c+d}, \quad N_{BGx} = -N_{FGx} \quad (2.1)$$

Meghatározzuk a  $G_y$  erő által létrehozott nyomóerőket a  $N_{RGy}$  és  $N_{LGy}$  pontokban.

$$N_{RGy} = \frac{hG_y}{a+b}, \quad N_{LGy} = -N_{FGy} \quad (2.2)$$

Ismerve a  $N_{LGy}$  és  $N_{RGy}$  pontokban ható erőket kiszámítjuk ezek eloszlását a robot kerekeire nézve, így megkapjuk azokat a nyomóerőket amelyet a ábra 2.5.10 ábrán látható állapotban a  $G_y$  gravitációból származó erő hoz létre.

$$N_{yBL} = \frac{dN_{LGy}}{c+d}, \quad F_{yFL} = -N_{yBL} \quad (2.3)$$

$$N_{yBR} = \frac{dN_{RGy}}{c+d}, \quad F_{yFR} = -N_{yBR} \quad (2.4)$$

Meghatározzuk a gravitáció Z komponense által létrehozott nyomóerőket a  $F_F$  és a  $F_B$  pontokban amelyhez hozzáadjuk a X komponens által létrehozott nyomóerőket ugyan ezekben a pontokban.

$$F_F = \frac{G_z d}{c+d} + N_{FGx}, \quad F_B = G_z - F_F + N_{BGx} \quad (2.5)$$

Ismét kiszámoljuk a kerekekre nézve a nyomóerőket ismerve az  $F_F$  és  $F_B$  erőket.

$$F_{BR} = \frac{aF_B}{a+b}, \quad F_{BL} = F_B - F_{BR} \quad (2.6)$$

$$F_{FR} = \frac{aF_F}{a+b}, \quad F_{FL} = F_F - F_{FR} \quad (2.7)$$

A merőleges nyomóerő vektor az X,Y,Z gravitációs erők által létrehozott nyomóerők összegzéséből áll.

$$N_{\perp} = [F_{FL} + N_{yFL} \quad F_{BL} + N_{yBL} \quad F_{FR} + N_{yFR} \quad F_{BR} + N_{yBR}]^T \quad (2.8)$$

### 2.5.2. Súlypont (X,Y) komponensének a meghatározása

Ismerve a robot méreteit  $W$  jelölje a szélességét míg a  $L$  hosszúságát, kerék középpont között mérve.

A robotot vízszintes helyzetbe helyezzük, és minden kerek merőleges nyomóerejét megmérve mérleg segítségével megkapjuk a  $N_{FL}, N_{FR}, N_{BL}, N_{BR}$  nyomóerőket.

$$W = a + b, \quad L = c + d \quad (2.9)$$

Meghatározzuk a  $a$  érteket ismerve a  $F_{FR}$  pontban a nyomóerőt és kiszámolva a  $F_F$  pontban a nyomóerőt a 2.11 és 2.12 egyenletek segítségével.

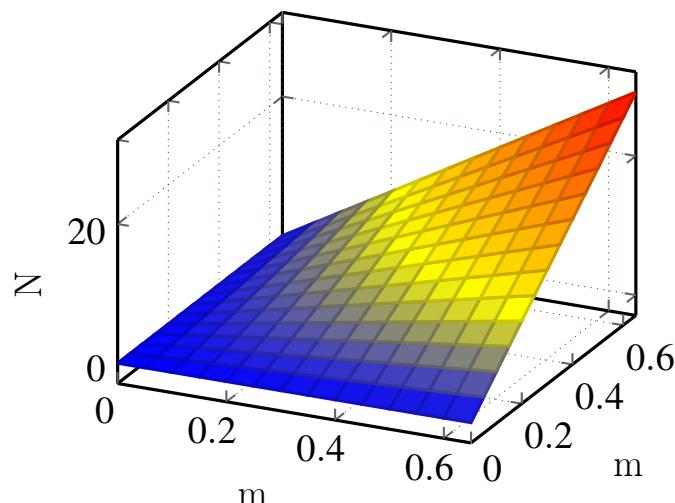
$$F_{FR} = \frac{aF_F}{W} \Rightarrow a = \frac{F_{FR}W}{F_F} \quad (2.10)$$

$$G_z = F_{FR} + F_{FL} + F_{BR} + F_{BL} \quad (2.11)$$

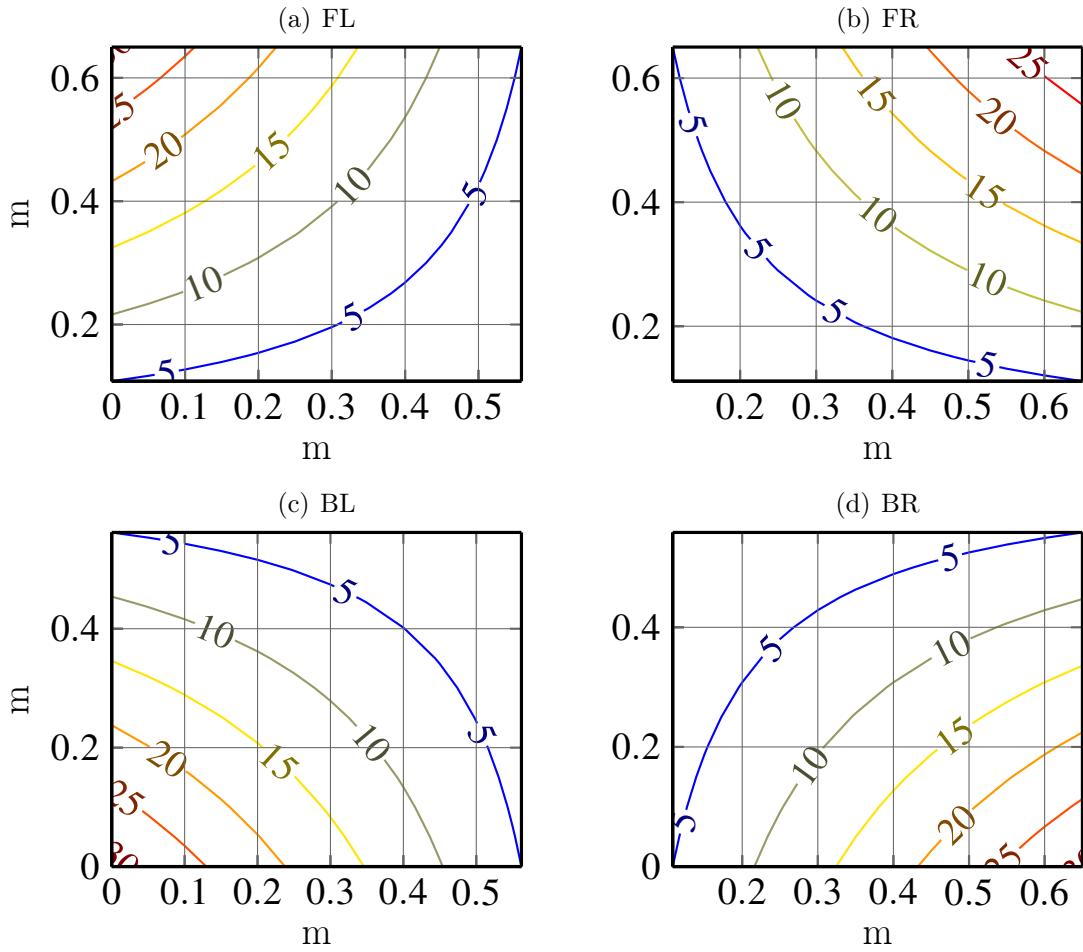
$$F_F = \frac{dG_z}{L} \Rightarrow d = \frac{F_F L}{G_z} \quad (2.12)$$

Merőleges nyomoero alakulása a súlypont fugvenyeben

A tételezzük fel hogy a robot súlya 28kg, a  $BR$  kerek közepe legyen a  $(0,0)$  pont,  $W \in (0m, 0.6m)$  és  $L \in (0m, 0.6m)$  értéket vehet fel. A ábra 2.5.11 ábrán látható a  $FR$  pontban a merőleges nyomóerő változása a súlypont pozíójának a függvényben.



2.11. ábra. Kerek nyomoero valtozasa a súlypont fugvenyeben ha  $\alpha = 0$  es  $\beta = 0$



2.12. ábra. *SSMR – 4W* tipusu robot kereknyomoerok kerekenkeni változása a súlypont függvényében

### 2.5.3. Súlypont meghatározása mérésekkel

A robot súlypont meghatározása egy mérleg segítségével lemérve sorra minden kerék merőleges nyomóerőjét a talajra nézve. A mért adatok vízszintes helyzetben:

Kerek	Nyomó erő	Mértékegység
FL	11,8	kg
FR	13,2	kg
BL	17,1	kg
BR	17,9	kg
$F_F$	25,0	kg
$F_B$	35,0	kg
Összsúlya	60,0	kg

A súlypont pozíciója:  $b = 0.34$  és  $c = 0.24m$  távolság a COG és COM között:  $0.053m$

### 2.5.4. Kerék Dinamikája

Az  $I_w \in \mathbb{R}^4$  tartalmazza a kerekek inerciáját a forgás tengelyükhez képest.  $\Omega \in \mathbb{R}^4$  a kerekek szögsebessége. A  $W_r \in \mathbb{R}^4$  a kerekek sugara,  $\tau \in \mathbb{R}^4$  a kerekek forgatónyomatéka.

---


$$I_w \dot{\Omega} = \tau - W_r F_x \quad (2.13)$$

$$I_w = \begin{bmatrix} I_{FL} & 0 & 0 & 0 \\ 0 & I_{BL} & 0 & 0 \\ 0 & 0 & I_{FR} & 0 \\ 0 & 0 & 0 & I_{BR} \end{bmatrix}, \quad W_r = \begin{bmatrix} r_{FL} & 0 & 0 & 0 \\ 0 & r_{BL} & 0 & 0 \\ 0 & 0 & r_{FR} & 0 \\ 0 & 0 & 0 & r_{BR} \end{bmatrix}$$

$$\tau = [\tau_{FL} \quad \tau_{BL} \quad \tau_{FR} \quad \tau_{BR}]^T, \quad \Omega = [\omega_{FL} \quad \omega_{BL} \quad \omega_{FR} \quad \omega_{BR}]^T$$

## 2.6. Kinematikai Modell

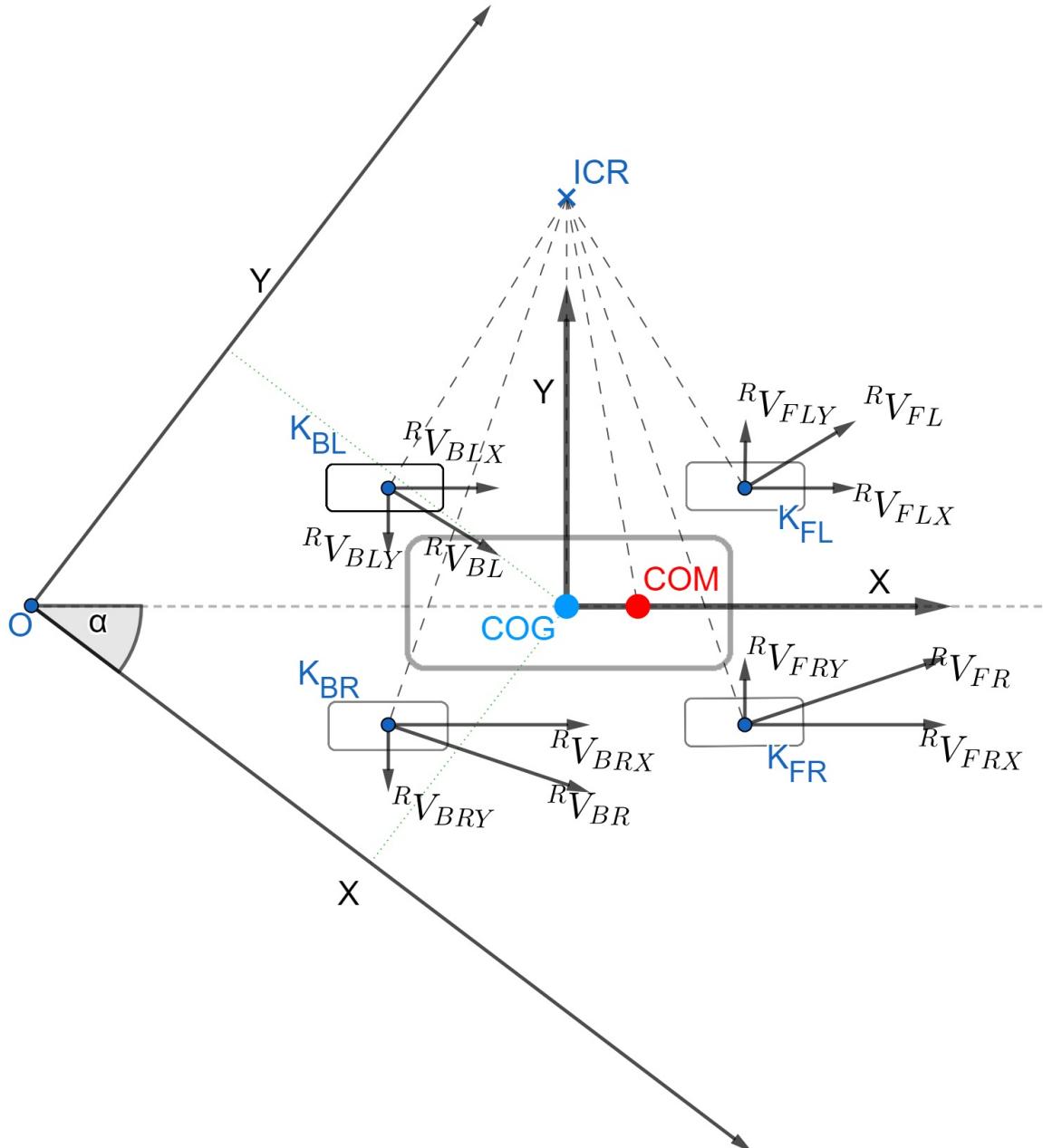
A ábra 2.6.13 látható a *4W – SSMR* kinematikai modellje. Néhány feltételezés: a robot minden kereke mindenkor érintkezik a talajjal, a kerekek nem csúsznak forgásuk közben, külön van kezelve a laterális és a longitudinális súrlódás, a robot egy tömeg központtal van jellemezve, az alacsony szintű szabályzok tökéletesen követik az előírt referenciát.

A robot a *ICR* pont körül fordul, és csak a robothoz rendelt vonatkoztatási rendszer x tengelye mentén tud elmozdulni. Az y irányú sebességeket azt okozza hogy a jobb és bal oldali kerekek forgási sebessége eltér és így létrejönne egy oldal irányú csuszás.

Jelölje a rendre a  $K_{ik}$  a kerekek a talajjal való érintkezési pontját,  ${}^R V_{ik}$  a  $K_{ik}$  pontok sebességét a robothoz rendelt *VNR*-ben,  ${}^R V_{ikX}$  és  ${}^R V_{ikY}$  rendre a  ${}^R V_{ik}$  sebesség X és Y komponense robothoz rendelt *VNR*-ben. A  ${}^R V_{ikX}$  megfelel a kerekek kerületei sebességének. A  ${}^R V_{ik}^{COM}$  a *COM* pont sebességet a robothoz rendelt *VNR*-ben, illetve a  ${}^R V_{ikX}^{COM}$  és  ${}^R V_{ikY}^{COM}$  az X és Y komponense.

A robot és a globális *VNR* x tengelye között bezárt szög  $\theta$  valamint *X* és *Y* a robot pozíciója a *O* ponthoz viszonyítva.

Az *ICR* pont helyzete a  ${}^R V_{ik}$  sebesség vektorokra merőleges egyenesek metszés pontjában található és mindenkor a robothoz rendelt *VNR* Y tengelyen helyezkedik el.



2.13. ábra. Kinematikai modell az  $4W - SSMR$  típusú robotnak.

A  $\dot{q}$  a  $4W - SSMR$  síkban modellezett állapot vektora a globális  $VNR$ -ben.  ${}^R\omega^{COG}$  az  $COG$  pont körüli szögsebesség a robothoz rendelt  $VNR$ -ben. Az  $\eta$  jelölje a bemeneti értékeket. A  $d$  a  $COG$  és a  $COM$  pontok közti távolság.

A  $COM$  pontban mert értékek az egyenlet (2.14) segítségével számolhatjuk a globális  $VNR$ -be. A  $COM$  pont sebességének  $y$  komponense megadható az egyenlet (2.15) segítségével. A nemholomonikus megkötés egyenlet (2.16) biztosítja azt hogy a robot nem tud oldal irányú mozgást végezni.

$$\dot{q} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^R V_x^{COM} \\ {}^R V_y^{COM} \\ {}^R \omega^{COG} \end{bmatrix} \quad (2.14)$$

---


$${}^R V_y^{COM} = d\omega \quad (2.15)$$

$$\begin{bmatrix} -\sin \theta & -\cos \theta & -d \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = A(q)\dot{q} = 0 \quad (2.16)$$

$$\dot{q} = S(q)\eta \quad (2.17)$$

$$S(q) = \begin{bmatrix} \cos \theta & -d \sin \theta \\ \sin \theta & d \cos \theta \\ 0 & 1 \end{bmatrix}, \eta = \begin{bmatrix} {}^R V_x^{COM} \\ {}^R \omega_{COG} \end{bmatrix}$$

$$S^T A^T = 0 \quad (2.18)$$

Az ICR pont meghatározása a robot paraméterei és a mért sebességek alapján [24].

$${}^R V_{COG_X} = \frac{V_L + V_R}{2} \quad (2.19)$$

$${}^R \Omega_{COG} = \frac{V_L - V_R}{b + a} \quad (2.20)$$

$${}^R ICR_X = -\frac{{}^R V_{COG_Y}}{{}^R \Omega_{COG}} \quad (2.21)$$

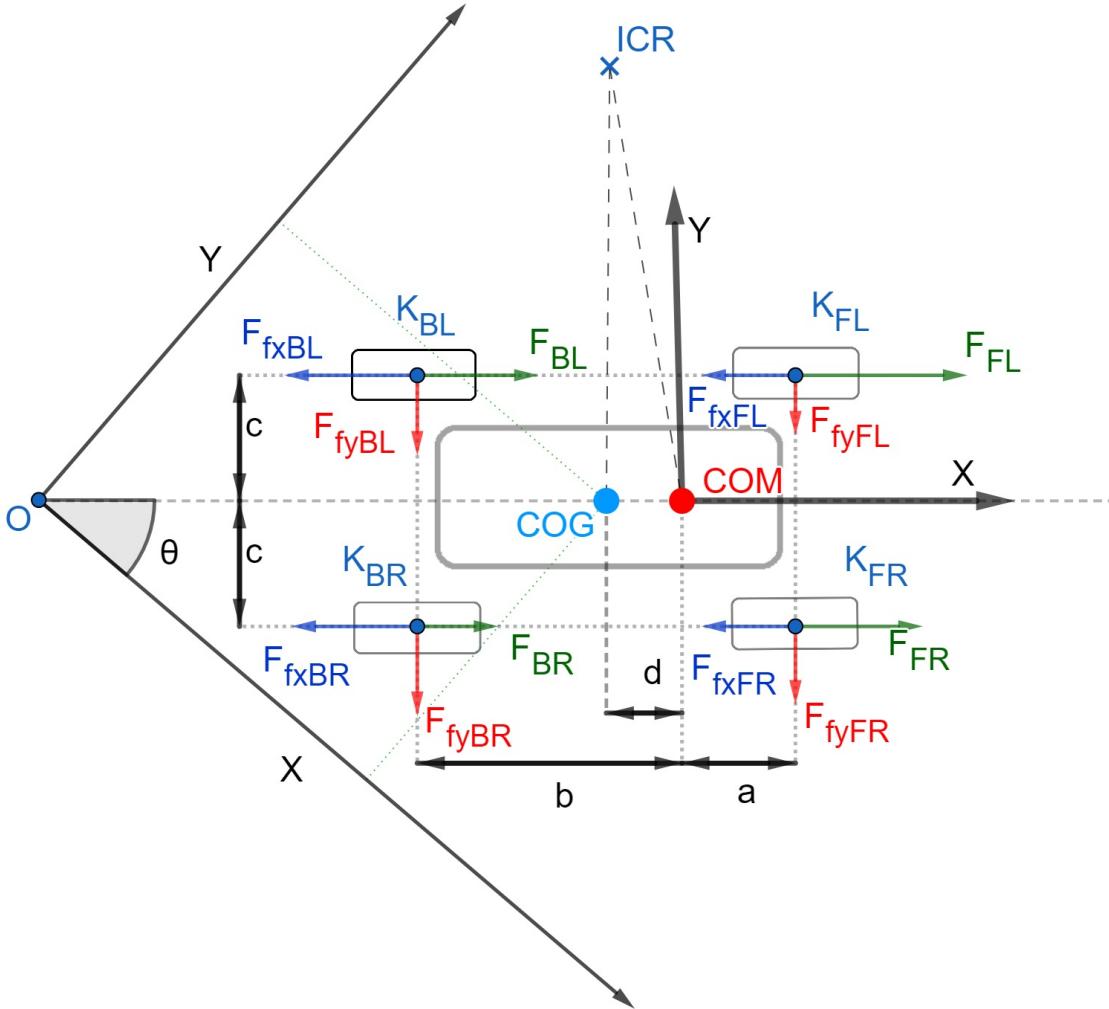
$${}^R ICR_Y = \frac{{}^R V_{COG_X}}{{}^R \Omega_{COG}} \quad (2.22)$$

$$R = \sqrt{{}^R ICR_X^2 + {}^R ICR_Y^2} \quad (2.23)$$

## 2.7. Dinamikai Modell

A ábra 2.7.14 látható a  $4W - SSMR$ -ra ható erők rendszerre. Jelölje a  $F_{ik}$  a  $K_{ik}$  pontokban a kerekek a talajra kifejtett erőt,  $F_{fxik}$  és a  $F_{fyik}$  rendre az x és y irányba ható súrlódási erőket a  $K_{ik}$  pontokban.

Az egyenletek (2.24)-(2.26) leírják a robot mozgását a globális rendszerben, felhasználva a robot  $VNR$  - ben mert erő hatásokat.



2.14. ábra. Kinematikai modell az *SSMR* típusú *MR* robotnak.

Az  $F_x \in \mathbb{R}^4$  tartalmazza  $F_{ik}$  kerekek által a talajra kifejtett erőket. Az  $F_{sx} \in \mathbb{R}^4$  és  $F_{sy} \in \mathbb{R}^4$  súrlódási erők x és y tengely mentén a robot *VNR*-ben. A  $F \in \mathbb{R}^2$  tartalmazza a jobb és bal oldali kerek által a talajra kifejtett erők összegét. Jelölje  $I$  a robot inerciáját a z tengely körül,  $M_a$  nyomatékok összege amelyeket a kerekek hoznak létre,  $M_r$  a nyomatékok összege amelyeket a súrlódások hoznak létre.

A  $K_x \in \mathbb{R}^4$  és  $K_y \in \mathbb{R}^4$  jelölje a súlypont pozíciója a kerek és talaj érintkezési pontokhoz viszonyítva a ábra 2.5.8 alapján.

Az  $N \in \mathbb{R}^4$  tartalmazza a merőleges nyomóerőket talajra nézve, a  $K_{ik}$  pontokban.

$$m\ddot{X} = \xi F_x \cos \theta - \xi F_{sx} \cos \theta + \xi F_{sy} \sin \theta \quad (2.24)$$

$$m\ddot{Y} = \xi F_x \sin \theta - \xi F_{sx} \sin \theta - \xi F_{sy} \cos \theta \quad (2.25)$$

$$I\ddot{\theta} = M_a + M_r, \quad (2.26)$$

$$\xi = [1 \ 1 \ 1 \ 1]$$

$$F_{sx} = [F_{sxFL} \ F_{sxBL} \ F_{sxFR} \ F_{sxBR}]^T, \quad F_{sy} = [F_{syFL} \ F_{syBL} \ F_{syFR} \ F_{syBR}]^T$$

$$F_{sxik} = N_{ik} \mu_{xik} S_{xik}, \quad F_{sy} = N_{ik} \mu_{yik} S_{yik}$$

$$\mu_x = [\mu_{xFL} \ \mu_{xBL} \ \mu_{xFR} \ \mu_{xBR}]^T, \quad \mu_y = [\mu_{yFL} \ \mu_{yBL} \ \mu_{yFR} \ \mu_{yBR}]^T$$

$$S_x = [sgn(V_{xFL}) \quad sgn(V_{xBL}) \quad sgn(V_{xFR}) \quad sgn(V_{xBR})]^T$$

$$S_y = [sgn(V_{yFL}) \quad sgn(V_{yBL}) \quad sgn(V_{yFR}) \quad sgn(V_{yBR})]^T$$

$$M_r = M_{rx} + M_{ry} \quad (2.27)$$

$$M_{rx} = K_x^T F_{sx}, M_{ry} = K_y^T F_{sy}, M_a = K_x^T F_x \quad (2.28)$$

$$K_x = [a \ a \ b \ b]^T, \quad K_y = [c \ d \ c \ d]^T$$

$$F_x = [F_{FL} \ F_{BL} \ F_{FR} \ F_{BR}]^T$$

$$F = [F_{FL} + F_{BL} \ F_{FR} + F_{BR}]^T$$

Általános formában a  $4W - SSMR$  dinamikai modellje a egyenlet (2.29) adható meg a [2] alapján. Jelölje  $M \in \mathbb{R}^{3 \times 3}$  a tömegek és inerciák mátrixa,  $R \in \mathbb{R}^3$  ellenálló nyomatékok és erők mátrixa,  $B \in \mathbb{R}^{3 \times 2}$  bemeneti mátrix,  $A$  a megkötések vektora egyenlet (2.16) alapján,  $\lambda$  Lagrange együtthatók vektora.  $F_d \in \mathbb{R}^3$  zajok vektora.

A egyenlet (2.29) az állapotok gyorsulását megkapjuk ha az egyenlet (2.17) időben deriváljuk, így az egyenlet (2.30)-t kapjuk.

Felhasználva a egyenlet (2.18) és egyenlet (2.30) és egyenlet (2.17) a egyenlet (2.29) egyenletet egyszerűbb alakra írhatjuk azáltal hogy minden tagot beszorzunk balról  $S^T$ -vel, így a egyenlet (2.31) kapjuk.

$$M(q)\ddot{q} + R(\dot{q}) + F_d = B(q)F + A^T \lambda \quad (2.29)$$

$$M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix}, \quad B(q) = \begin{bmatrix} \cos \theta & \cos \theta \\ \sin \theta & \sin \theta \\ -a & b \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}^T, \quad R(\dot{q}) = \begin{bmatrix} \xi F_{sx} \cos \theta - \xi F_{sy} \sin \theta \\ \xi F_{sx} \sin \theta - \xi F_{sy} \cos \theta \\ M_r \end{bmatrix}$$

$$\ddot{q} = S(q)\dot{\eta} + \dot{S}(q)\eta \quad (2.30)$$

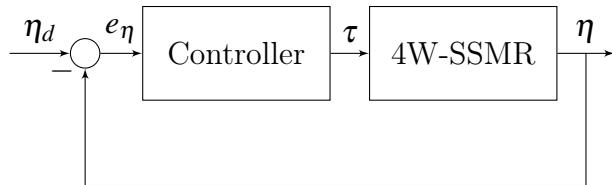
$$\bar{M}\dot{\eta} + \bar{C}\eta + \bar{R} + \bar{F}_d = \bar{B}F \quad (2.31)$$

$$\bar{M} = S^T M S, \quad \bar{C} = S^T M \dot{S}, \quad \bar{R} = S^T \dot{R}, \quad \bar{F}_d = S^T F_d, \quad \bar{B} = S^T B$$

## 2.8. Robot Platform Sebesség Szabályzása

### 2.8.1. Előírt nyomatékkal

A kerekek előírt nyomatékát megkapjuk ha a egyenlet (2.32) -t használva. Az  $u$  szabályzó jelet kiszámíthatjuk ha az egyenlet (2.33)-t használjuk. Jelölje a  $K_\eta$  a szabályzó paramétereit. Csuszás szabályzó  $\sigma_\eta$  paraméteri  $\rho_v$  lineáris sebességért, és  $\rho_w$  szögsebességért felelős. Mindkét paraméter nagyobb kell legyen mint a zaj  $n$  megfelelő érteké.



2.15. ábra. Kinematikai modell az *SSMR* típusú *MR* robotnak.

$$\tau = W_r D \bar{B} [\bar{M} u + \bar{C} \eta + \bar{R}] + I_w \dot{\Omega} \quad (2.32)$$

$$u = \dot{\eta}_d + K_\eta e_\eta + \sigma_\eta \quad (2.33)$$

$$e_\eta = \eta_d - \eta \quad (2.34)$$

$$e_\eta = [e_v \ e_w]^T \in \mathbb{R}^2, \quad \sigma_\eta = [\sigma_v \ \sigma_w]^T \in \mathbb{R}^2, \quad \sigma_v = \rho_v sgn(e_v), \quad \sigma_w = \rho_w sgn(e_w) \quad (2.35)$$

#### Mesterséges Erő Módszere

A [5] cikk alapján egy másik megközelítést használva modellezzi a robot. A  $q$  állapotokat meg kiegészíti a jobb és bal oldali kereke szögsebességével. Feltételezi hogy a kerekek sugara  $r$  minden a négy kerekénél egyenlő, és a *COM* pont a robot szimmetriatengelyén helyezkedik el. Jelölje  $F$  a ellenálló erők és nyomatékok vektora. Hasonlóképpen az egyenlet (2.29) -hez a Lagrange egyenletet ír fel a dinamikai modellre. Az  $\eta$  tartalmaz az előírt sebességek vektora, az  $\eta_3$  a sebességek vektora, a 3-dik eleme tartalmazza a generált sebességet amelyet úgy kell előírnunk hogy a hozzá tartozó előírt kerek sebesség nulla legyen. Az  $u \in \mathbb{R}^3$  a jobb és bal oldali kerekek előírt erőleadása a talajra.

$$A(q)\dot{q} = \begin{bmatrix} \cos \theta & \sin \theta & -c & -r & 0 \\ \cos \theta & \sin \theta & c & 0 & -r \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \\ \Omega_L \\ \Omega_R \end{bmatrix} \quad (2.36)$$

$$M(q) = \begin{bmatrix} m & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I_{FL} + I_{BL} & 0 \\ 0 & 0 & 0 & 0 & I_{FR} + I_{BR} \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$F(q, \dot{q}) = [F_x \cos \theta - F_y \sin \theta \quad F_x \sin \theta - F_y \cos \theta \quad M_r \quad 0 \quad 0]^T \quad (2.37)$$

$$\dot{q} = G_e(q) \boldsymbol{\eta} = \begin{bmatrix} \cos \theta & \cos \theta & -\sin \theta \\ \sin \theta & \sin \theta & \cos \theta \\ \frac{1}{c} & -\frac{1}{c} & 0 \\ 0 & \frac{2}{r} & 0 \\ \frac{2}{r} & 0 & 0 \end{bmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{pmatrix} \quad (2.38)$$

$$\underbrace{G_e^T M G_e}_{M^*} \dot{\boldsymbol{\eta}} + \underbrace{G_e^T (M \dot{G}_e + C G_e)}_{C^*} \dot{\boldsymbol{\eta}} + \underbrace{G_e^T F}_{F^*} = \underbrace{G_e^T B u}_{B^*} \quad (2.39)$$

$$M^*(q) = \begin{bmatrix} m + \frac{I}{c} + 4 \frac{I_k}{r^2} & m - \frac{I}{c^2} & 0 \\ m - \frac{I}{c^2} & m + \frac{I}{c^2} + 4 \frac{I_k}{r^2} & 0 \\ 0 & 0 & m \end{bmatrix}, \quad F^*(q, \dot{q}) = \begin{bmatrix} -F_x - \frac{M_r}{c} & -F_x + \frac{M_r}{c} & -F_y \end{bmatrix}^T$$

$$C^*(q) = \begin{bmatrix} 0 & 0 & -m\dot{\theta} \\ 0 & 0 & -m\dot{\theta} \\ m\dot{\theta} & m\dot{\theta} & 0 \end{bmatrix}$$

$$B^* = \begin{bmatrix} \frac{2}{r} & 0 & 0 \\ 0 & \frac{2}{r} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (B^*)^{-1} = \begin{bmatrix} \frac{r}{2} & 0 & 0 \\ 0 & \frac{r}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{\eta}_r = \begin{pmatrix} \eta_{r_1} \\ \eta_{r_2} \\ \eta_{r_3} \end{pmatrix}$$

$$u = (B^*)^{-1} \{ M^* \dot{\boldsymbol{\eta}}_r + C^* \boldsymbol{\eta}_r + F^* - K_d e \} \quad (2.40)$$

$$\boldsymbol{\tau} = W_r D [u_1, u_2]^T \quad (2.41)$$

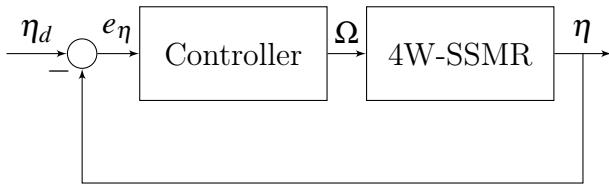
$$e_{\boldsymbol{\eta}_i} = \boldsymbol{\eta}_i - \boldsymbol{\eta}_{r_i} \quad (2.42)$$

$$\dot{\boldsymbol{\eta}}_{r_3} = \frac{m\dot{\theta}(\boldsymbol{\eta}_{r_1} + \boldsymbol{\eta}_{r_2}) - F_y - K_d(\boldsymbol{\eta}_3 - \boldsymbol{\eta}_{r_3})}{m} \quad (2.43)$$

$$\boldsymbol{\eta}_{r_1} = \frac{{}^R V_x^{COM} - {}^R \omega_r^{COG} c}{2}, \quad \boldsymbol{\eta}_{r_2} = \frac{{}^R V_x^{COM} + {}^R \omega_r^{COG} c}{2}$$

### 2.8.2. Elirt kerekszogsebessegekkel

A [3] cikkben a kerekek sebességét szabályozza, A jobb és bal oldali kerekek modelljét ARX becsléssel meghatározza a matematikai modellt és pólusáthelyezéses módszerrel a kívánt modellt állítja elő.



2.16. ábra. Kinematikai modell az *SSMR* típusú *MR* robotnak.

## fejezet 3

# A rendszer implementálása

### 3.1. Robot felépítése

A robot egy négykerekű kültéri mobilis robot *4W – SSMR*, amelyet négy különálló csiga áttétel és egy DC motor mozgat.

Paraméter	érték	Mértékegység
Szélesség	80	[cm]
Hosszúság	80	[cm]
Magasság	40	[cm]
Önsúly	54	[kg]
Max sebessége	25	[cm/s]
Max fordulási sebesség	25	[°/s]
Kerék-átmérő	40	[cm]
Maximális kerék forgatónyomaték	1000	[N/m]

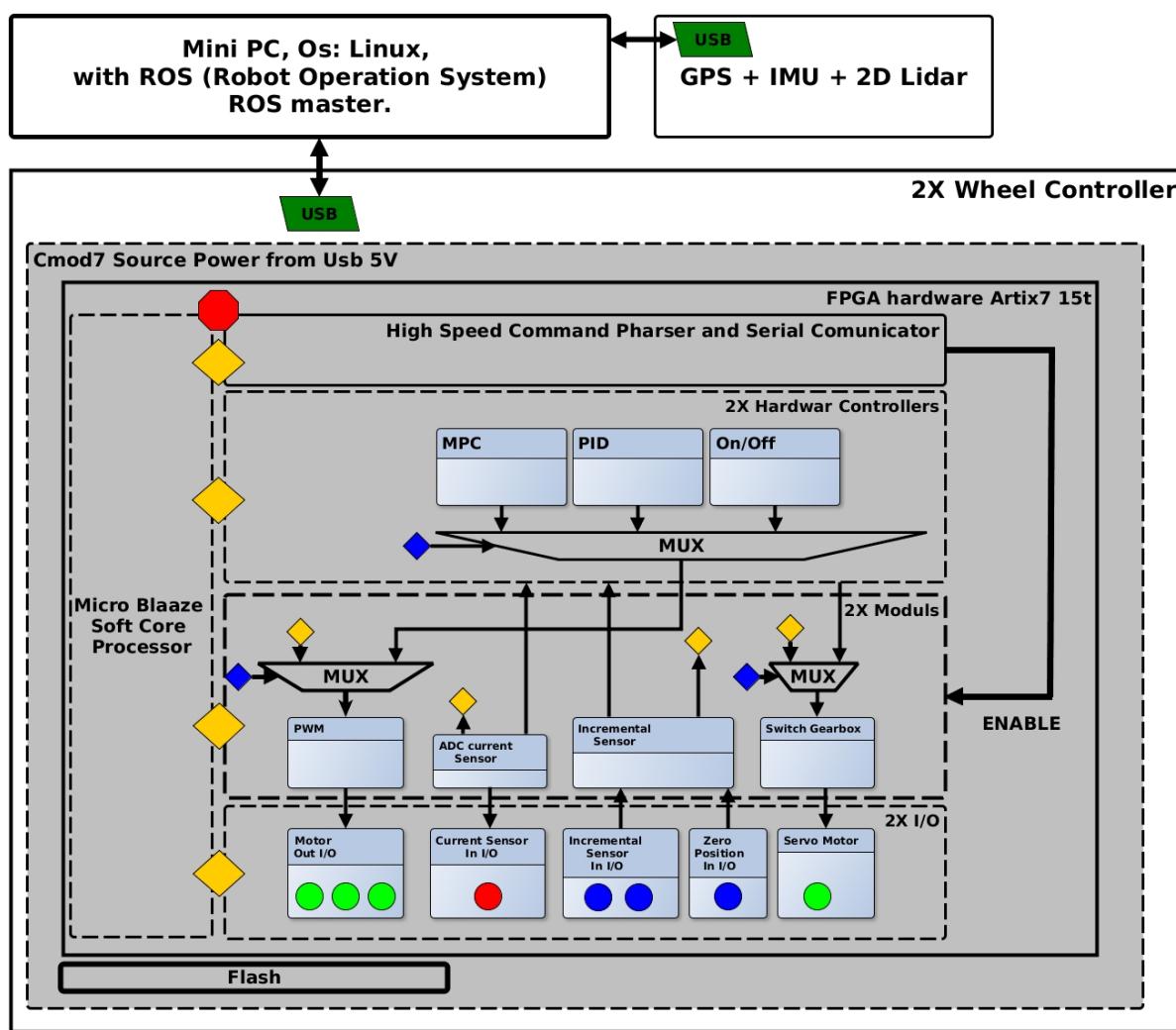
A roboton, egy performáns számítógép felelős a ROSmaster és egyéb kiszolgáló nodok futtatásáért, a rendszerhez USB ábra 3.2.1 csatlakozóval kapcsolódó modulok:

Modulnév	Leírás
2xFPGA	CmodA7 <a href="https://reference.digilentinc.com/reference/programmable-logic/cmod-a7/reference-manual">https://reference.digilentinc.com/reference/programmable-logic/cmod-a7/reference-manual</a>
IMU	ESP8266 arduino alapú rendszer, SPI protokollon keresztül olvassa az IMU mert értékeit, átalakítás után string formában uart protokollon küldődik a kiszolgáló ROS nodnak
GPS	Uart porton NMEA protokoll alapján ROS node fogadja az értékeket.
LIDAR	Leírás: <a href="https://www.robotshop.com/media/files/pdf2/rpk-02-datasheet.pdf">https://www.robotshop.com/media/files/pdf2/rpk-02-datasheet.pdf</a> , gyártó által biztosított ROS <a href="http://wiki.ros.org/rplidar">http://wiki.ros.org/rplidar</a> integráció.

### 3.2. Alacsony szintű modulok

A roboton alacsony szintű feladatait amelyek a motor hajtásokhoz kapcsolódó szenzorokat és vezérlő jelek előállítására hivatott a két CmodA7 20T FPGA fejlesztőlap. Négy tengely szögsebességet kell szabályozni, egy FPGA két hajtást valósít meg. A ábra 3.2.1 alapján egy hajtáshoz tartozó I/O-k:

Név	Darabszám	Típus
Inkrementális szenzor	2	Digitális Input
Árammérő szenzor	1	Analóg Input
Motor vezérlő	3	Digitális Output
Végállás kapcsoló	1	Digitális Input



3.1. ábra. CmodA7 FPGA-ban kialakított architektúra amely a szenzorok és motor hajtások jeleinek feldolgozását és előállítását valósítja meg

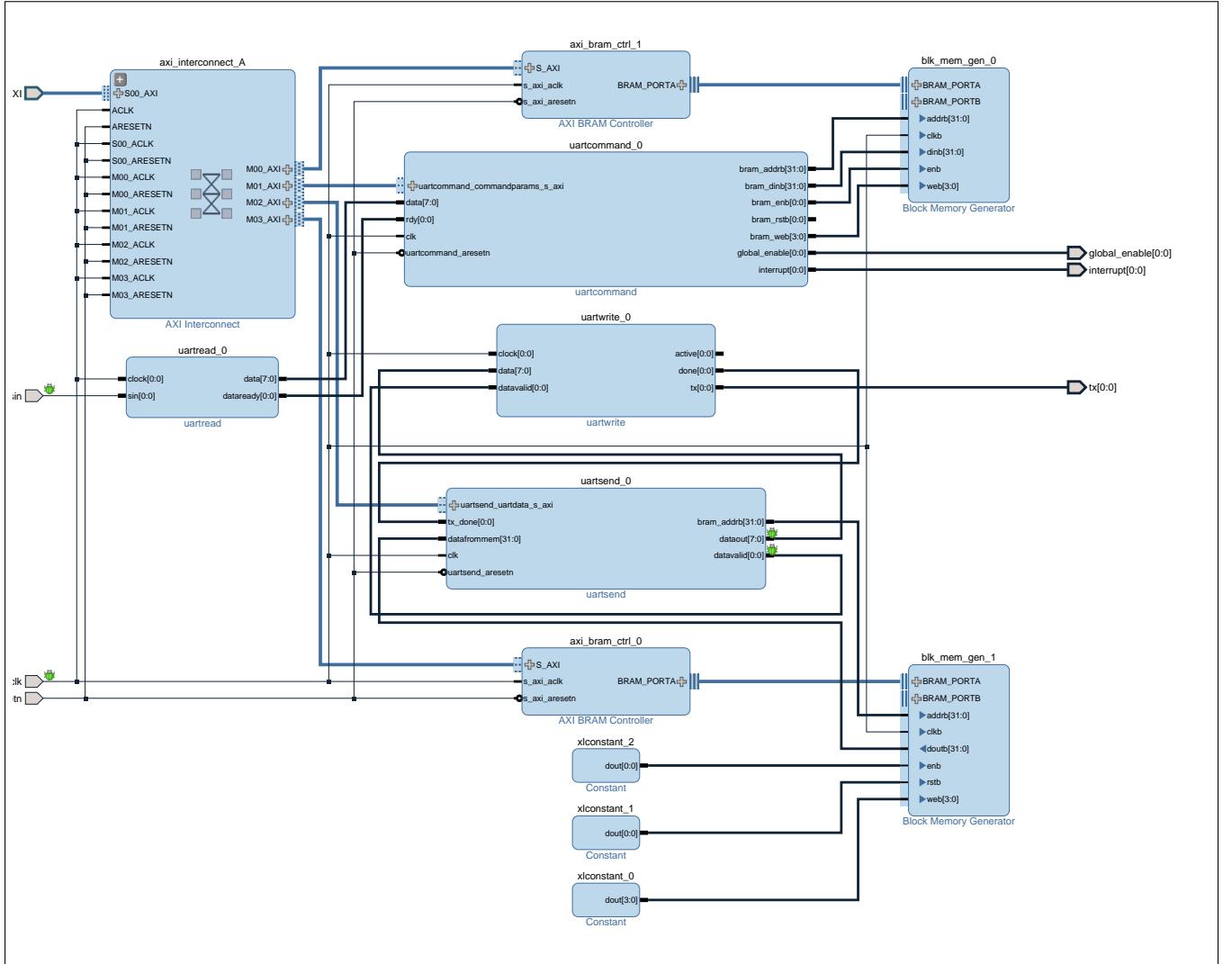
A ábra 3.2.4 a Vivado tervezőprogramban FPGA-ban megvalósított processzor architektúra látható, tartalmaz egy szoftcore processzort (microBlaze). Az utasítás és az

---

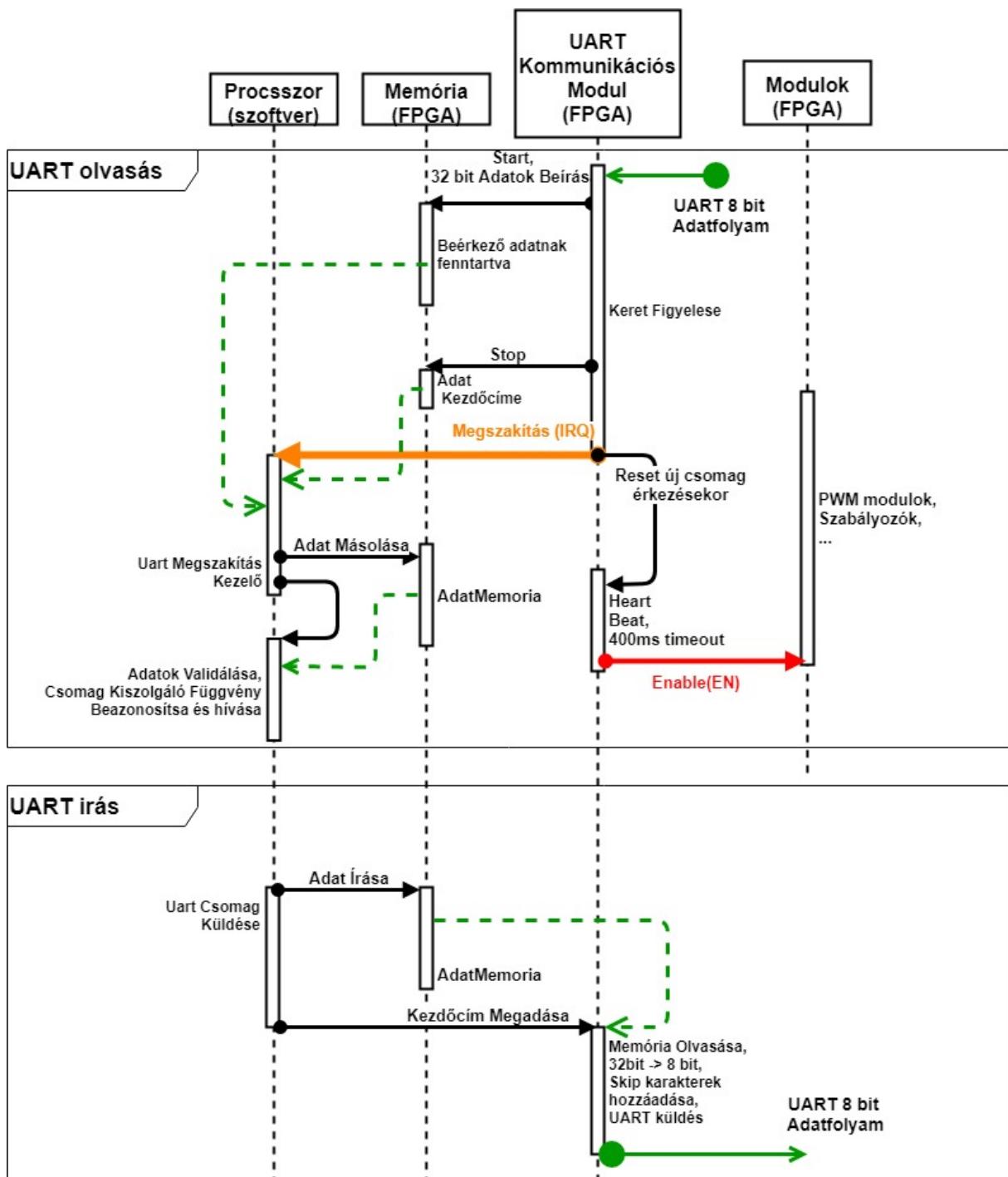
adatmemória az *microblaze\_0\_local\_memory*-ban található. A memória mérete 24Kbyte 32bit sávszélességgel, amely az FPGA block memóriájából van összerakva. A programkód maga egy külső memóriában van tarolva, QSPI alapú interfésszel kapcsolódik a FPGA hoz, az *ExternalMemory*s kapcsolja be az AXI busz rendszeren keresztül a processzor memória címtartományába. A külső memória mérete 30 MByte. A processzor működéséhez szükséges a *ClockAndReset* modul előállítja az órajelet ami 100 MHz, ezen a frekvencián működik az processzor és a memória is. Az *mdm\_1* modul a rendszerben jeleinek és a programkód megfigyelésére hivatott.

A ábra 3.2.4 ábrán látható *UartCom* modul belső szerkezete látható az ábra 3.2.2 ábrán. A modul *uartcommand\_0* szerepe a kommunikációs protokoll implementálása hardveres szinten, a modul fogadja az UART protokollon érkező 8 bites csomagokat. A protokoll által meghatározott keretezési byte-okat figyelve (Start,Stop,Skip). Ezek paraméterként megadhatók a microBlaze processzoron keresztül. A processzor induláskor beállítja ezeket a következő értékekre: Start='S', Stop='P', és Skip='.'. Start keret byte érkezésétől kezdve az összes byte négyesével bekerül a 32 bit, 400 byteos, *blk\_mem\_gen\_0* memóriába jogfolytonosan. Abban az esetben ha az üzenet meghaladja a 200 byte-t akkor a modul úgy tekinti hogy hibás adatok érkeztek és visszatérte a kiinduló állapotba. Abban az esetben ha bináris adatokat küldünk tartalmazhat speciális protokoll keretezésben résztvevő karaktereket így ezeket előtt Skip karakter kell hogy legyen. A skip karakterek nem kerülnek bele a memóriába ezeket a hardver kiveszi és az utána következő karaktert nem értelmezi.

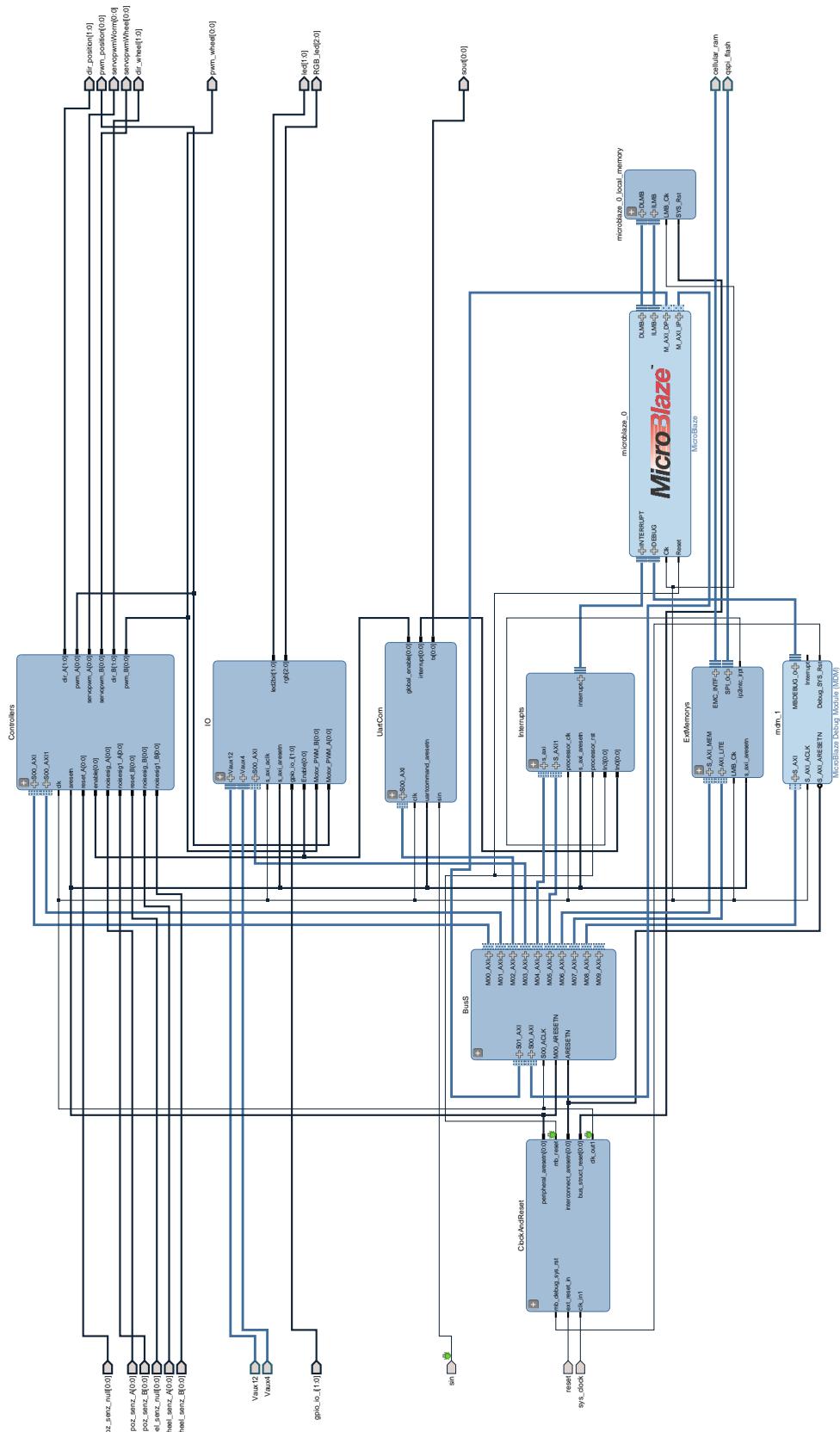
A kommunikáció folyamatát a ábra 3.2.3 ábra is végigkövethetjük.



3.2. ábra. FPGA ban megvalósított kommunikációs modul a ?? leírt protokoll alapján.



3.3. ábra. FPGA hardver/MicroBlaze processzor és ROS node közti kommunikáció megvalósítása UART protokoll alapján



3.4. ábra. FPGA ban megvalósított szoftprocesszor rendszer, legfelső négyzete.

---

A keret véget jelző byte érkezésekor a modul generál egy megszakítást a microBlaze processzornak az *interrupt*[0 : 0] kivezetésen keresztül. A megszakítás kiszolgálásakor a processzor kiolvassa az *uartcommand*<sub>0</sub> modulból az új üzenet kezdőcímét. A processzor az adatokat az *axi\_bram\_ctrl\_1* modulon keresztül kepés kiolvasni, az új üzenetre kezdőcímére, mutató pointer értékét megkapjuk, ha az *uartcommand*<sub>0</sub> modul processzor memóriájába illesztésétt kezdőcímét és a modul által jelzett kezdőcímét összeadjuk. Abban az esetben, ha az adatok feldolgozása előtt új csomag érkezik a hardver kommunikációs modulhoz, elkezdődik annak beírása a memóriába, az előző adatok felülírás nélkül.

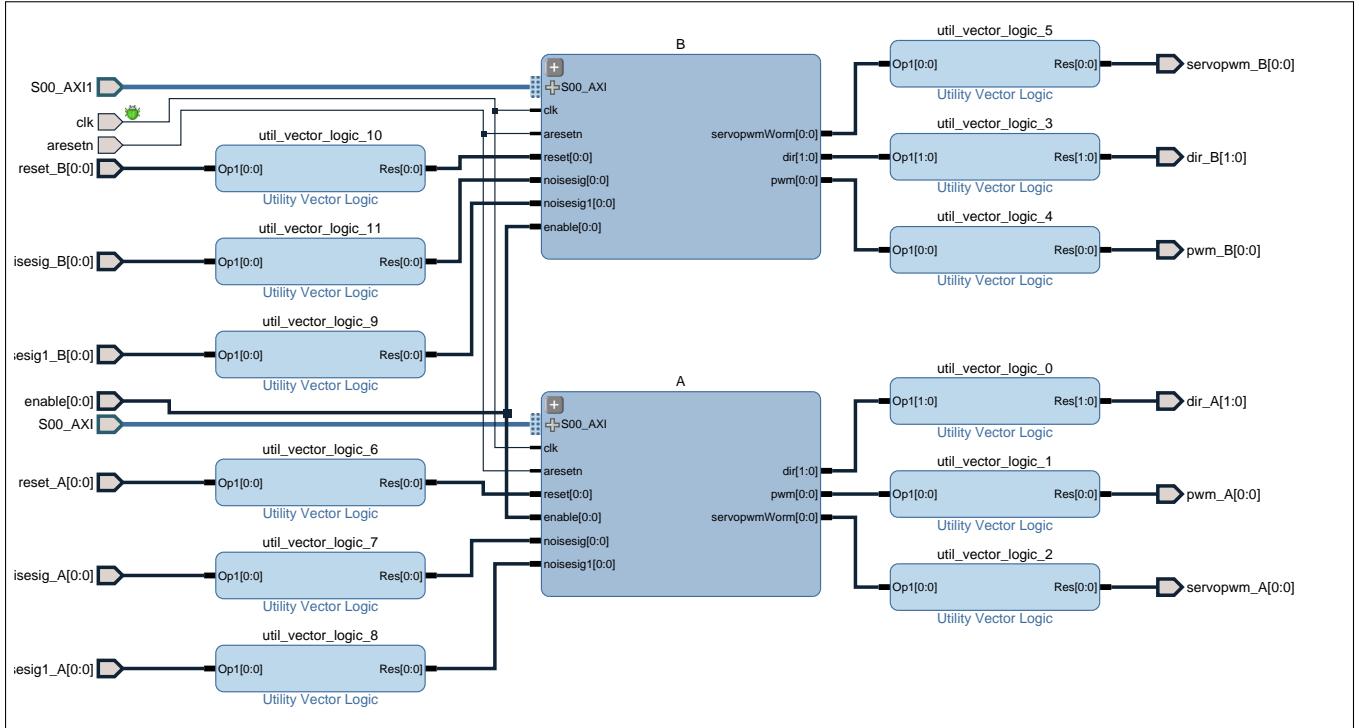
A *uartcommand*<sub>0</sub> modul kimenete *global\_enable*[0 : 0] engedélyező jel, abban az esetben ha nem kap a hardver *SREP* üzenetet legalább 300ms periódussal, akkor az engedélyező jelet alacsonyra állítja vagy ha *SRDP* üzenet érkezett.

Az *uartsend\_0* modul hasonlokeppen működik csak az adatok küldésére hivatott. A microBlaze proceszor az *axi\_bram\_ctrl\_0* modulon keresztül beleirodnak a *blk\_mem\_gen\_1* [25] 32bites és 400byte méretű memóriába. Az írás végével a processzor beállítja a küldés jelet, amely egy paramétere a hardvernek. A modul elkezdi kiküldeni az adatokat. Abban az esetben ha a csomag tartalmaz speciális protokollt leíró byte-t akkor a modul automatikusan elküldi előtte a Skip karaktert.

Az *uartread\_0* az UART protokoll implementálja FPGA hardverebe, a *sin* az RX bemenet, a *data*[7 : 0] a 8 bites szeles adatvezeték ami tartalmazza az utolsó beérkezett byte-t. A *dataready*[0 : 0] felfutó él jelzi az új adat érkezését. Az *uartwrite\_0* modul a *data*[7 : 0] 8bites sinen érkező adatot küldi ki UART protokoll alapján a *tx*[0 : 0] kimenetere abban az esetben ha a *datavalid*[0 : 0] bemenetén egy felfutó él érkezik.

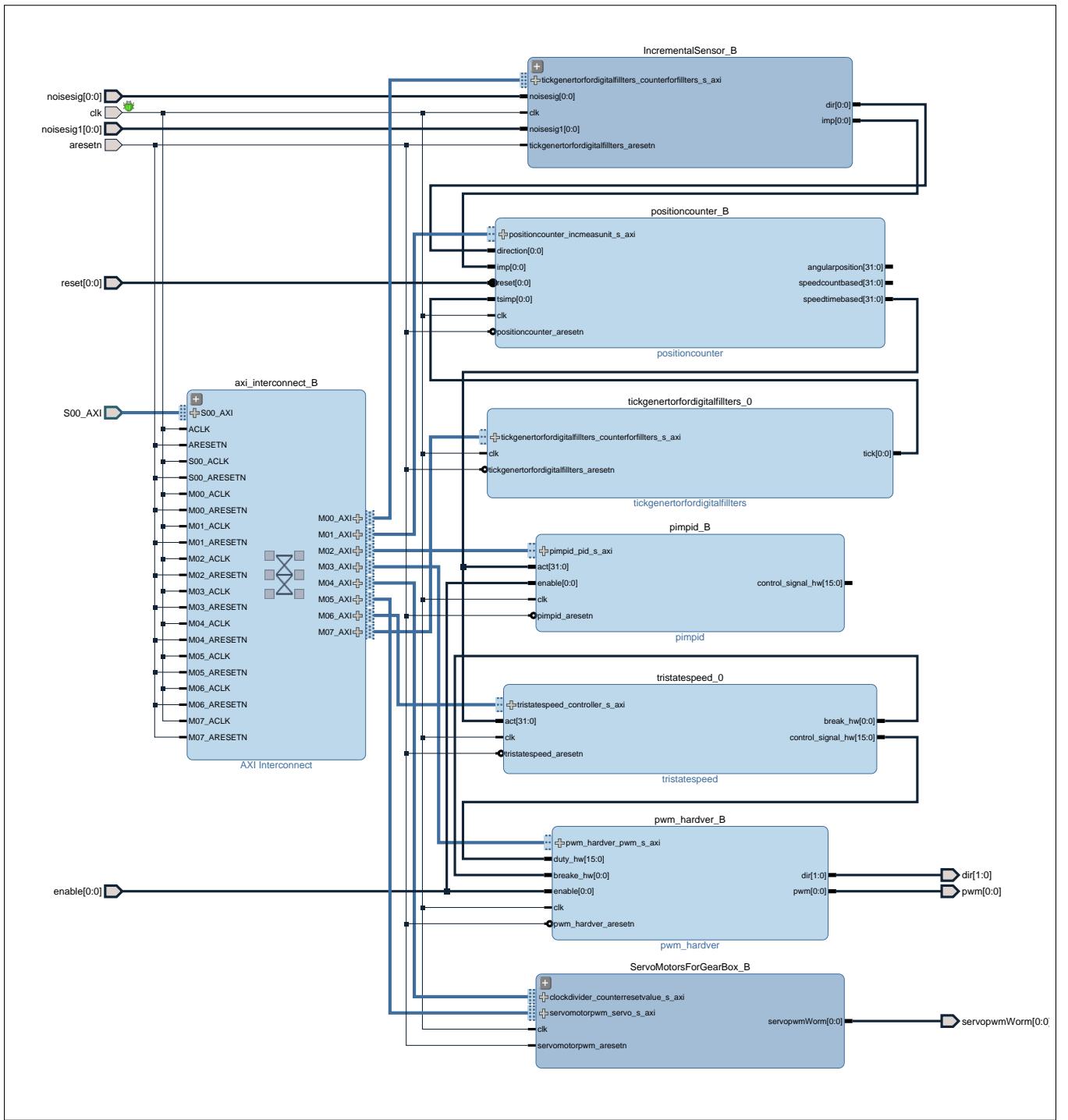
Az ábra 3.2.5 látható *A* és *B* modul feladatak egy-egy DC motor szabályzását/vezérlését lássák el. Mindkét modulban ugyan azon alegységek találhatók meg az *A* modul az robot első, míg a *B* modul a robot hátsó kerekét szabályozza.

A ábra 3.2.6 ábrán látható az *A* és *B* modulok belső szerkezete. A *pwm\_hardver\_B* PWM jelet állít elő, a *pwm*[0 : 0] kimenetre, és egy irányjelet *dir*[0 : 0], a bemeneti 16bites előjeles számból áll. A PWM kitöltési tényezője -32000 tol 32000-ig terjed ki (15bit), az előjel meghatározza az irányt míg az abszolút érteke a pwm kitöltési értéket. A *enable*[0 : 0] jel engedélyezi a kimenetet, ha érteke alacsony akkor a pwm kimenet is alacsony.



3.5. ábra. FPGA-ban megvalósított szabályzok A és B

Az *InkrementalisSensor\_B* axi sínen keresztül kapcsolódik a processzorhoz, feladata az inkrementális szenzortól érkező A és B jeleknek a feldolgozása. A modul két jelet küld ki:  $dir[0 : 0]$  amely a tengely forgás irányát jelzi,  $imp[0 : 0]$  egy órajelperiódusig tartó felfutójelet küld ki az inkrementális szenzor minden egyes elmozdulására.



3.6. ábra. FPGA controller modul

A *positioncounter\_B* modul meri a szögsebességet és a szögpozíciót idő és impulzus számolási módszert használva. A processzor a saját mintavételezési frekvenciájával olvassa ki a modulból a mert értékeket és átalakítja kivánt mértékegységbe.

Az ábra 3.2.6 ábrán látható többi modul jelen pillanatban továbbfejlesztési lehetőségekkel jelenik meg. A *pimpid\_B* modul FPGA alapú PID szabályzó modul, *ServoMotorGearBox\_B* a DC motrokon levő fogaskerék kapcsolási lehetőséget hivatott majd állítani egy szervomotor segítségével.

A szabályzok referencia értékeit a *RefValues* funkcionális biztosítja, a megfelelő üzenetek érkezésekor a memóriába beállítódnak a kívánt referencia értékek, és majd a

---

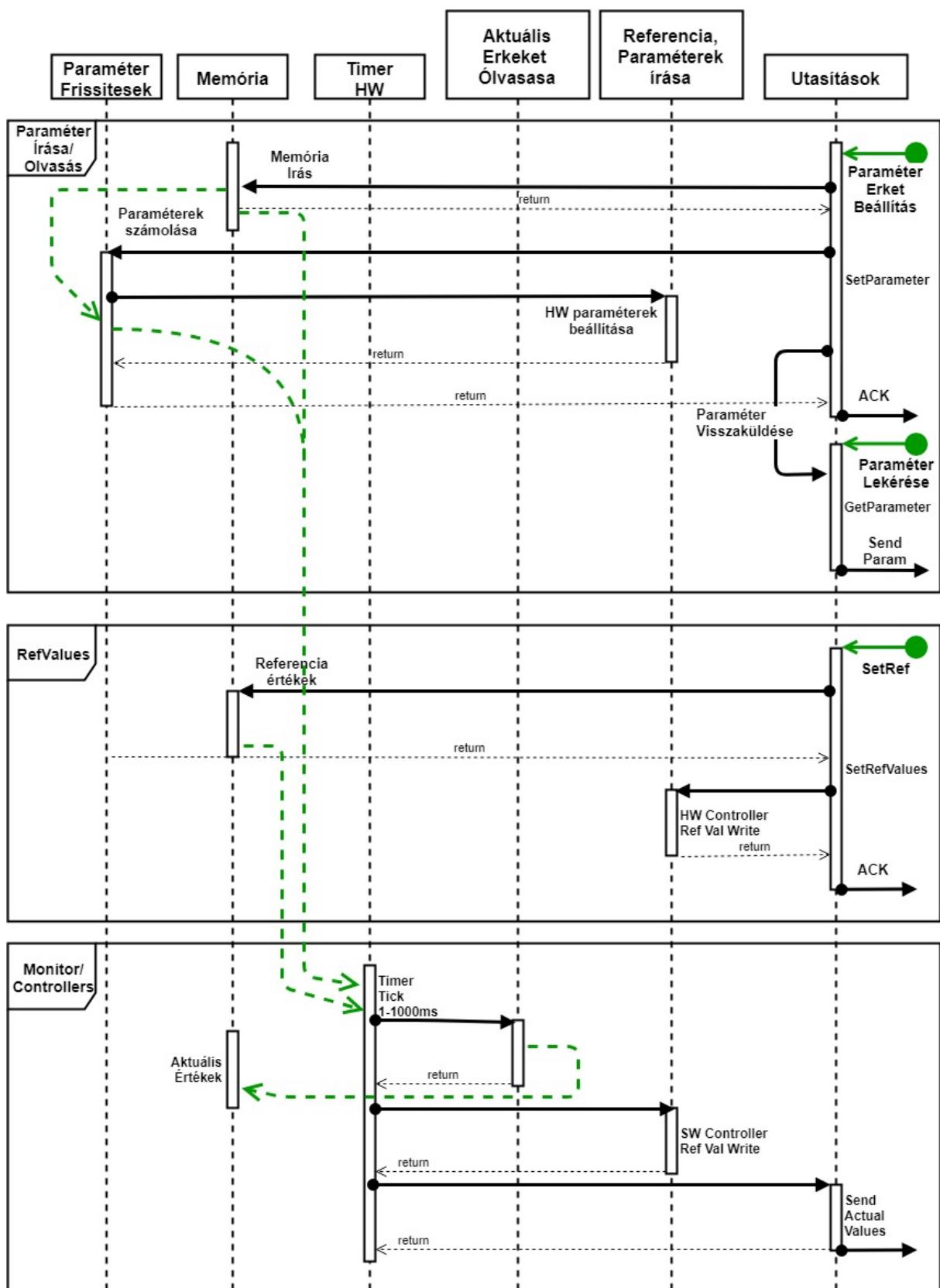
softveres szabályzok kiolvassák. A hardveres szabályzóknak pedig a beérkezés pillanatban leküldődik.

A *Monitors/Controllers* funkcionalitas egy parameterbol allithato periodusu idozito amely megszakitasokat general a proceszornak. A proceszor ezekre a megszakitasokra szamolja ki a beavatkozi jelek uj ertekeit, valamint a elkului a mert,es a kapott referencia ertekeket UART on keresztul.

### 3.2.1. Microblaze szoftvere

A rendszer tervezésénél a fő koncepció az volt hogy a rendszer architekturája dinamikus legyen a fejleszthetőséget tekintve, így a ábra 3.2.1 levő arhitekurát kaptuk.

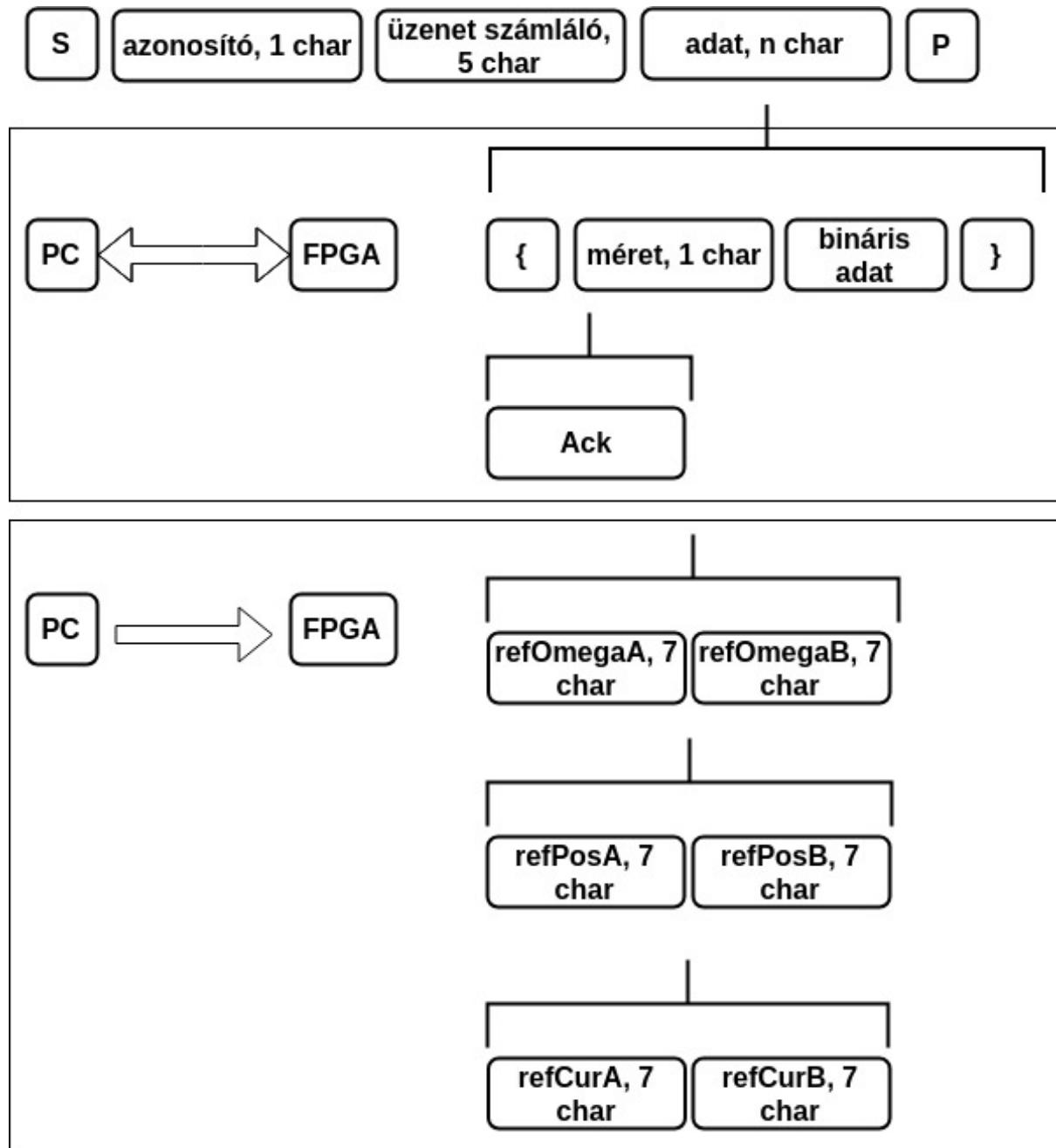
A microblaze processzor feladatai hardveres modulok kezdeti állapotainak a beállítása, softveresen futó szabályzók működtetése, a beérkezett üzenetek feldolgozása. A processzoron futó szoftvert az ábra 3.2.7 UML diagram szemlélteti. A paraméterek írása és olvasása funkcionálisban, az új beérkezet paraméter értékének a beállítási mente a következő: az üzenet beleíródik a memóriába, megszakítás generálódik a processzor irányában, a processzor feldolgozza az új üzenetet, észleli hogy paraméter értékének a beállítása típusú üzenet érkezett, meghívódik a kiszolgáló függvény. A továbbiakban az összes paraméter újraszámolódik, és újraküldődik a hardveres modulok irányába. Abban az esetben ha minden sikeresen beállítódott a processzor pozitív visszajelzést(ACK) üzenetet küld és visszaküldi a beállított paraméter értékét, acélból hogy a küldő is megbizonyosodjon a paraméter helyes beállításáról.



3.7. ábra. MicroBlaze processzoron futó szoftver diagramja

### 3.2.2. FPGA és UART alapú kommunikációs protokoll

Megvalósítva a kommunikációt a kiszolgáló ROS-node, amely UART protokollra épített saját üzenetekből áll. ábra 3.2.8.



3.8. ábra. FPGA komunikacions protokol altalanos csomag szerkezet

A protokoll keretek közé foglalt adatok rendiségre épül. Speciális karakterek<sup>1</sup> jelzik az üzenet kezdetét és véget jelen esetben az *S* csomag kezdetét, míg a *P* a csomag véget

<sup>1</sup> speciális karakterek amelyek jelzik az értelmező számára hogy olyan ka-

rakter következik amelyet a protokoll értelmezésében nem kell végrehajtani.

---

jelentik. Az csomag tartalmazhat bináris formátumú adatot is ezeket minden esetben a { és } speciális karakterek közé kel kerülniük, mert a köztük levő adat nem fog értelmezésre kerülni a feldolgozó által. Bináris reszt pl.: struktúra típusú adatra használhatjuk. Mindig a { utáni első karakter megadja a bináris adat hosszát így tudja az értelmező hogy hol kell várnia a lezáró karaktert. A bináris szekció hossza nem lehet nagyobb mint 254 byte, de amennyiben szükséges kiterjeszthető.

Minden üzenet tartalmaz egy egyedi azonosítót ami meghatározza az üzenet típusát ez alapján tudja eldönteni majd a fogadott fél hogy melyik kiszolgáló rutint kell felhívnia. Ezt követi egy üzenet számláló 5 char hosszú mező amely string formában tartalmaz egy egész számot, amely minden kiküldött üzenet után növelni kell. Az üzenetszámláló alól kivételt képeznek a direkt hardveres üzenetek pl.: *SREP* és a *SRDP*. A hasznos adat további felépítését meghatározható annak függvényében hogy mit szeretnénk. Jelen esetben a üzenet típus orientált parancsokat szerkesztünk amelyekkel a referencia értékeket, paramétereket állítjuk be, valamint bináris szekciót is tartalmazó struktúrát küld az FP-GA modul PC-nek amely a szabályzást és historizálást szolgálja.

### 3.2.3. Paraméterek FPGA modul

A paraméterek segítségével tudjuk beállítani a mintavételezési periódusokat, az inkrementális szenzorok felbontását, a szabályozok beállítását is ezáltal oldhatjuk meg. Konfigurációs paraméterek: Az alábbi táblázatban leírjuk a ROS-ban használt paramétereket. A paraméterek névének végen levő *X* jelölje *A* vagy *B*, attól függően hogy melyik DC motorhajtáshoz tartozik. Az ábra 3.2.5 ábrán látható modulokat tudjuk konfigurálni.

Id	Név (X lehet A vagy B)	Értékek		Típus	Leírás
		Min	Max		
1	TsTimerPeriod	1	1000	int16	Mintavételezési periódus [ms]ban.
2	GetDataPeriodical	0	1	int16	Kapcsoló ha 0 akkor nem küld az FPGA mért értékeket, különben a TsTimerPeriod periódusú mintavétellel küld.
3	TorqueCoefX			float16	Motor arám és nyomaték közti együttható.
4	ActiveControllerX	0	65535	int16	Válaszható szabályzó típusok hajtásoknál 0=Szoftvare PID szögsebesség, 1=Hardver PID szögsebesség, 2=Szoftver PID arám, 3=Hardver PID arám
5	MaxControlSiggnaX	0	32760	sint16	A beavatkozó PWM jel maximális kitöltési tényezője, lineárisan $0->0\%$ -tol $32760->100\%$ -ig.
6	IncSenzResX	0	65535	int16	Inkrementális szenzor által generált impulzusok száma egy teljes kerékfordulatra. FPGA oldalon ez a szám 10-el szorzandóik.
7	IncSenzCountDirectionX	-1	1	sint16	Inkrementális szenzor jeleit feldolgozó modul számolási irányá állítható be.
8	Kp_Whell_PidX	0		float16	szögsebesség szabályzó, PID erősítési paramétere.
9	Ti_Whell_PidX			float16	szögsebesség szabályzó, PID integrálási idő.
10	Td_Whell_PidX			float16	szögsebesség szabályzó, PID deriválási idő.

### 3.2.4. Kommunikáció sebessége

Az uart sebessége 1MBd<sup>2</sup> ami megfelel 131072 byte/s adatforgalomnak. A valóságban a kommunikáció hibátlanul működik 1 ms periodussal, küldött 100byte méretű üzeneteket. Összehasonlítva *AXI\_UART\_Lite* [26] modullal elérte eredmények, 50 ms periódust tudtunk csak elérni, mert a protokoll értelmezését a szoftver végezte és nem a hardver. A FIFO használata ellenére sem sikerült kisebb mintavételezési periódust elérni, hogy nem minden karakter után generált processzor megszakítást csak minden 16-dik karakter után.

$$frekvencia = \frac{131072}{SizeOfPachage} = \frac{131072[byte/s]}{100[byte]} = 1310,072[Hz] \quad (3.1)$$

### 3.2.5. Biztonsági megoldások

Abban az esetben ha kiküldtük az előírt értékeket a modulnak és ezután a kommunikáció megszakadt a modullal akkor a szabályzok próbálják tartani az előírt érteket annak el-

<sup>2</sup> megabaud

---

lenére is hogy az már lehet hogy nem aktuális. Erre a célra beépítésre került egy üzenet és egy logika (HeartBeat), a periodikusan érkező üzenet, amely csak a kommunikációs modulhoz érkezik meg, és jelzi hogy a ROS jól működik, és fordítva is küldődik periodikus üzenet amely jelzi az FPGA normális működést. Abban az esetben ha a megszakad leállítja a motorokat és a szabályzókat, ábra 3.2.3 alapján az Enable(EN) jelet használva.

Irány	Üzenet	Periódus
FPGA->ROS	SEP	300 ms kötelező
ROS->FPGA	mintavételezett értékek küldése	dinamikusan módosítható

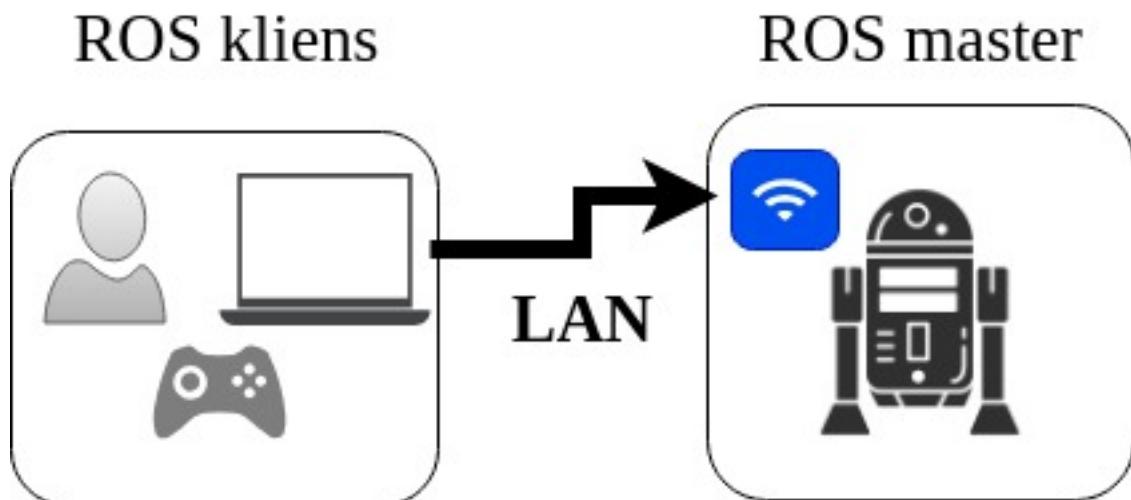
### 3.3. ROS

A robot irányítását megtehetjük azáltal, hogy csatlakozunk wifin keresztül a robothoz. Az irányításhoz szükséges egy számítógép, amelyre telepítve kell legyen a ROS. A kliens gépen fognak futni a következő felhasználói eszközök: -a rendszerben futó programok által loggolt üzeneteket szűrhetjük és tekinthetjük meg valós időben. -a paramétereket beállíthatjuk egy grafikus felhasználói felület segítségével (*dinamic\_reconfigure*).-rviz amelyel megjeleníti a robot 3D modelljét a térképen, a robot aktuális pozíójában és irányában a környezetéhez viszonyítva. *-JuglerPlot* segítségével online ábrázolhatunk mért értékeket mint pl.: a szabályzókörökben mért értékeket.

Jelenleg két működési mód közül választhatunk:

-távirányítóról vezérelve. A távirányítón található egy joystick, amely segítségével a robotnak referencia sebességeket írhatunk elő. A hozzárendelt referencia rendszerhez képest az X tengelyen m/s sebességet adunk meg, míg a Z tengelye körül °/s forgási sebességet adunk meg. A távirányító két gomb folyamatosan lenyomva tartásával a deadman kapcsolót valósíthatjuk meg. Abban az esetben, ha a két gomb nincsen lenyomva, a szabályzók működése és a pwm generátorok kimenete letiltva marad.

-automata mód, itt rviz program segítségével előírhatunk egy referencia pozíciót és irányt ahova szeretnénk, hogy a robot eljusson. Ez alatt a távirányítón a deadman kapcsolókat lenyomva kell tartanunk. A referencia értékeket megadhatjuk rviz használata nélkül azáltal, hogy a megfelelő típusú és nevű üzenetet beküldjük a ROS rendszerbe.



3.9. ábra. Robothoz csatlakozás a Wifi-n keresztül.

Az ábra 3.3.10 láthatóak a főbb nodok, topikok és a közöttük levő relációk. A nodok és a topikok leírását a ??

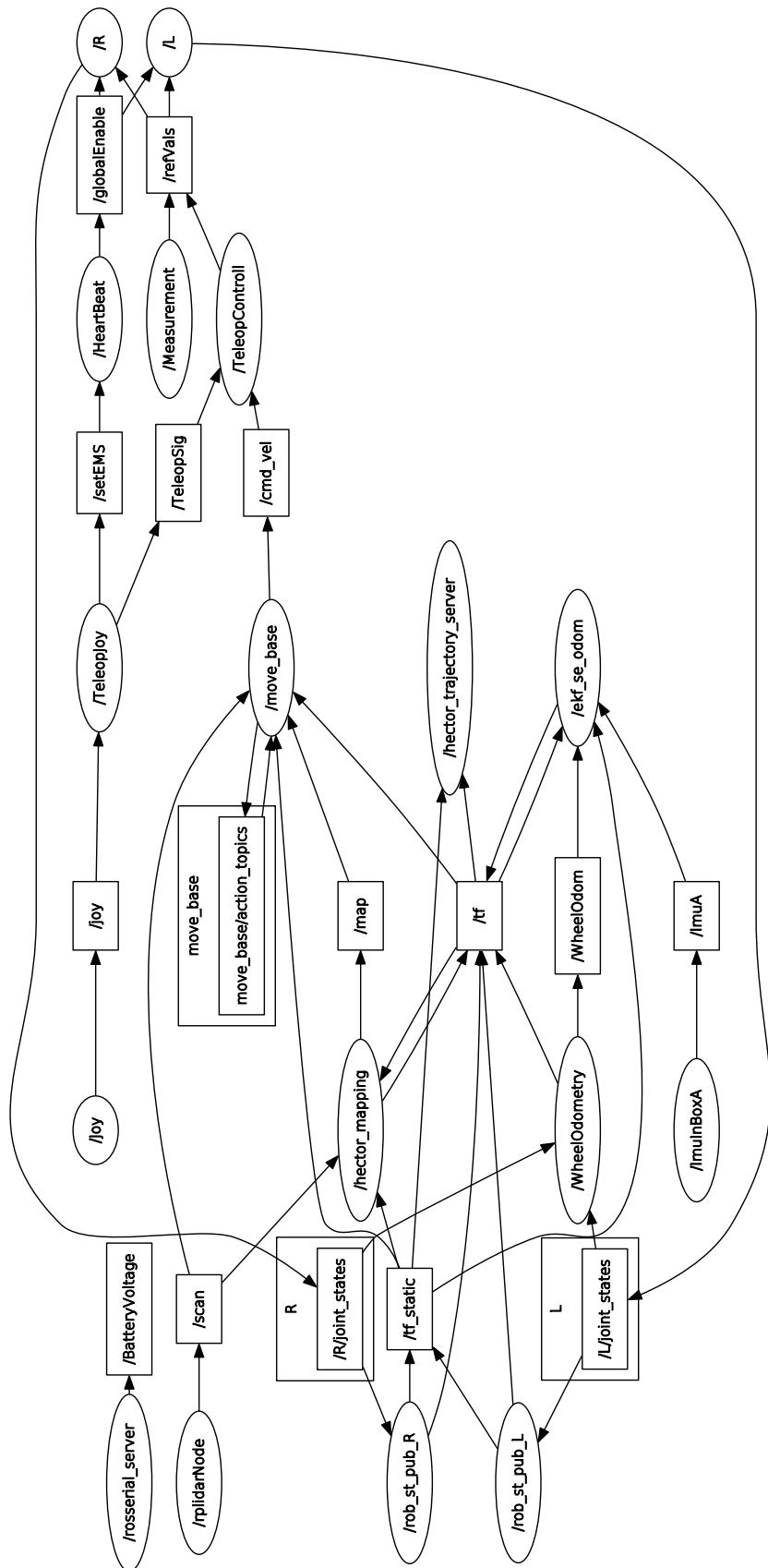
Node Név	Típus	Leírás
/R és /L	FPGA Communication Modul	Az FPGA-val való kommunikációért felelős. Adatokat fogad/küld a 3.2.2 fejezetben leírt protokoll alapján, amelyeket továbbít a ROS operációs rendszerben működő nodoknak.
/ImuInBoxA	Imuxxxx	Feladata szöveges formában érkező adatok feldolgozása és a ROS keretrendszerbe integrálását oldja meg. Mért fizikai mennyiségek:
/WheelOdometry		Kerekek mozgásából számolt robot elméleti pozíciója a térben
/TeleopJot		Feladata a joystick-től érkező parancsok fogadása. Megvalósítja a <i>DeadManSwitch</i> gomb kezelését, létrehoz egy globálisan engedélyező jelet a /setEMS-t és a /TeleopSigt, amely előírja a robot lineáris mozgási sebességét és a forgási sebességét a Z tengely körül.
/rplidarNode		A lidar mérési adatait olvassa ki és továbbítja a /scan topikban.
/rosserial_server		A megvalósítja a kommunikációt egy esp8266 fejlesztőlap és a ROS között, amely az akkumulátorok feszültségeinek a merését végzi [7]. Joystick eszközök integrációját valósítja meg [18].
/Joy		Lidar mérései alapján 2D térképet készít a környezetről, miközben lokalizálja a robotot ezen a térképen [13]
/hector_mapping		mérésék elvégzésére szolgáló node, amely egy előre beállított intervallumokban a megadott referencia értékeket küld ki az FPGA ban levő szabályzóknak.
/Measurement		Átalakítja a robot sebesség állapotait és kiszámolja nyílt hurokban a szabályzok előírt értékeit. Abban az esetben, ha a /movebase nodot használjuk, ez megoldja a robot pozíció szabályozását a térképen, így a /cmdvel csomagot csak átalakítja /refVals csomaggá azáltal, hogy azonos oldalon levő kereke ugyanazt a referencia értéket kell követniük.
/TelepoController		

### 3.3.1. Üzenet típusok (.msg)

Az alábbi táblázatban láthatjuk a ROS operációs rendszer által szolgáltatott .msg üzenetek kiterjesztését, amelyek lehetővé teszik az FPGA integrációját a ROS környezethez.

Üzenet típus	Értékek	Erkek típusa	Leírás
header	-	std_msgs/Header	Minden üzenet tartalmaz egy fejlécet, amely információkat tartalmaz az üzenetről.
	seq	uint32	minden üzenetet egyedileg beazonosító szám
	stamp	time	idő-bélyeg amely a küldés időpillanatát tárolja.
	frame_id	string	
/GlobalEnable	systemIsOk	int16	=0 - a HLC működik. <>0 - a HLC nem működik
/refVals	names	string[]	
	ref_position	float[]	előírt szög pozíció
	ref_velocity	float[]	előírt szög sebesség
	ref_effort	float[]	előírt forgatónyomaték
/setEMS	value	int16	=0 - Vészleállító aktív. <>0 - Vészleállító nem aktív
/joyControll	vx	float64	A robot X tengely mentén előírt sebessége m/s-ban.
	omega	float64	előírt szög pozíció A robot Z tengely körüli forgása °/s-ban.
	ControlMode	int64	Választhatunk a HLC szabályzok típusa vagy a manuális irányítás közül =0 move_base szabályzó, =1 manuális irányítás joystick segítségével.

A ábra 3.3.10 ábrán láthatjuk a nodok és az üzenetek közti kapcsolatot.



3.10. ábra. ROS graph

---

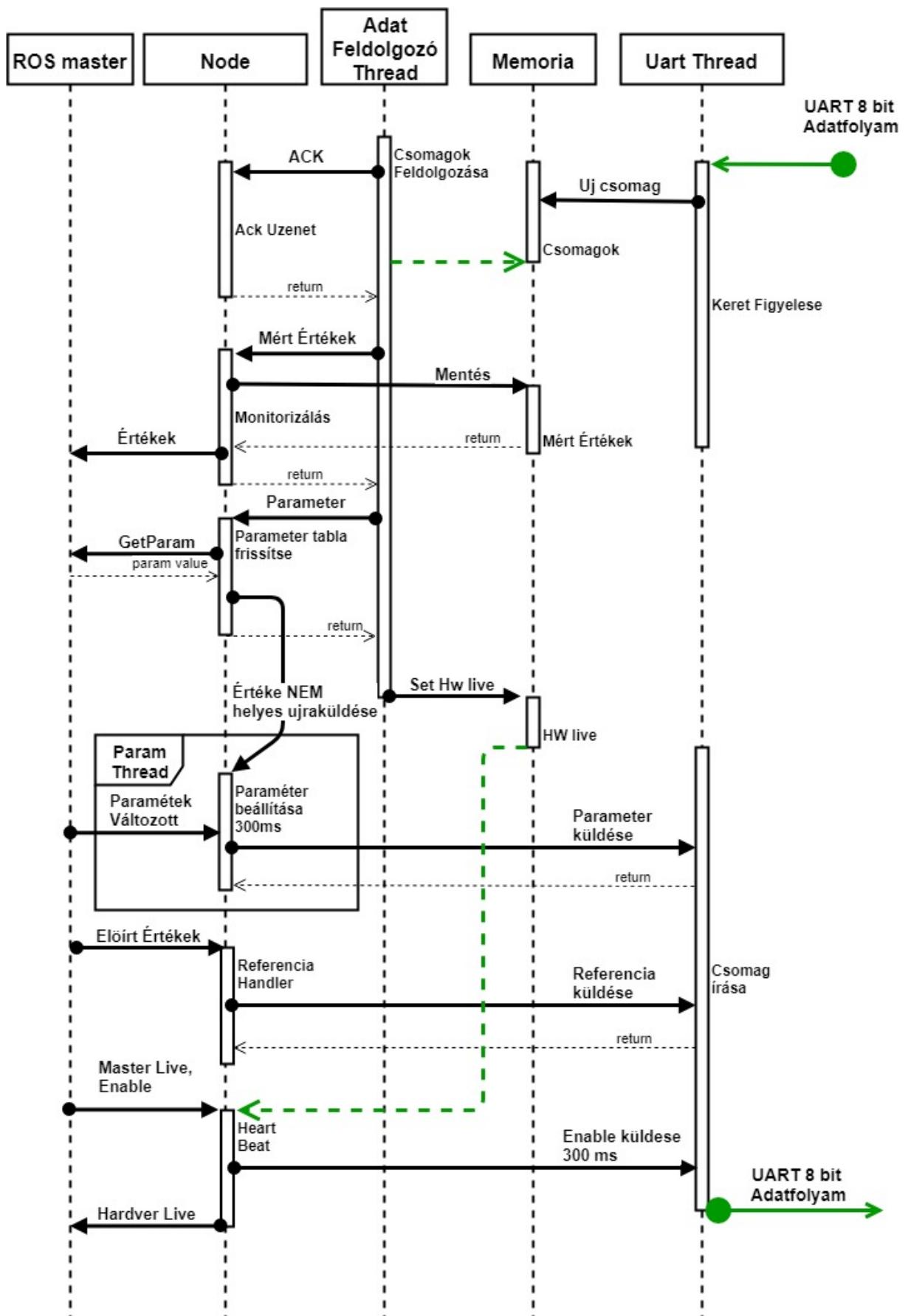
### 3.3.2. FPGA kommunikációs modul ROS oldali integració

A ROS biztosít a fejlesztőknek egy megoldást, amelyek képesek újonnan létrehozott robot integrálását ROS környezetben [7]. Előnye hogy gyorsan látványos eredményt érhetünk el, de a működés sebessége és az üzenetek méretében is korlátozott. Ezen hátrányokból kifolyólag sajátos integráció szükséges, amely integrálta az FPGA UART kommunikáció protokollt a ROS keretrendszerben működő más modulokhoz.

A ábra 3.3.11 diagramon a kommunikáció node technikai megvalósítását láthatjuk. Különálló szál gondoskodik az UART adatok olvasásáról és írásáról. Az üzenetek értelemezését egy külön szál végzi és hívja fel a kiszolgáló függvényeket. A paraméterek helyes beállításáról a ParamThread szál gondoskodik, paraméterek helyes beállításáról FPGA oldalon. Abban az esetben, ha a hardver kap egy új paramétert a ábra 3.2.7 alapján az FPGA visszaküldi a kapott paramétert, a visszajelzésből eldönthető, hogy a paraméter a hardverben helyesen állítódott-e. Abban az esetben, ha nem megfelelő, újraküldődik mindenkorábban amíg nem sikeres a beállítás.

A paraméterek kezelésére a ROS paraméter szerver a felelős [20], abban az esetben, ha egy paraméter megváltozott, amely az illető nodehoz köthető, akkor a ábra 3.3.11 ábrán látható ParameterValtozott esemény előidézi a megváltozott paraméter értékének az elküldését FPGAnak irányába.

A globális engedélyező jel a ábra 3.3.11 MasterLive Enable, /globalEnable típusú üzenettel engedélyezhetjük a szabályzok működését, a folyamatos működéshez 500ms periódussal kell érkeznie. Abban az esetben, ha a központi számítógép valami okból leállna, akkor a hardveres szabályzok is leállnak. A node 300ms periódussal küldi tovább az engedélyező jelet az FPGA modulnak. A HardverLive jel információt szolgáltat a többi ROS környezetben futó és a működés szempontjából kritikus nodenak, hogy az adott modul megfelelően működik-e. Ezen információ birtokában a HeartBeat node leállítja a rendszert, ha egyik FPGA modul nem válaszol.



3.11. ábra. ROS integrálása Uart protokolhoz.

### 3.3.3. Előirt értékek

A /refVals típusú üzenetben megadjuk minden egyes motor előírt értékét annak fövenyében °/s hogy sebesség alapján szabályzunk vagy  $N/m$  előírt nyomaték alapján.

### 3.3.4. Vonatkoztatási Rendszerek

A vonatkoztatási rendszerek szükségesek, mert a szenzorok és beavatkozó eszközök egymáshoz viszonyított helyzete és pozíciója is változhat. Sok esetben szükséges ismernünk egy adott eszköznek a múltbeli helyzetét, vagy egy másik vonatkoztatási rendszerhez képest a pozíóját vagy irányát. A ROS biztosít egy tf [22] nevű csomagot amely megvalósítja a szükséges transzformálásokat a VNR-k között. A ábra 3.3.12 látható a kialakított vonatkoztatási rendszerek a roboton amely hűen modellez a fizikai robot kialakítását. A vonatkoztatási rendszereket két csoportba oszthatók:

- (a) rögzített pozíció és szögek, szabadságok 0: Szenzorok laser, BODY\_link, wheel\_odom, ImuALink VNR je a base\_link a globális robot VNR hoz:
- (b) rögzített pozíció csak szögek változnak, szabadságok 1: Kerekek VNR je: FL\_link, BL\_link, FR\_link, BR\_link a BODY\_link hez képest csak Y körül foroghat.
- (c) pozíció és szög is változik, szabadságok 6: A robot base\_link az helymeghatározás odom, és az odometria a térképhez map viszonyítva.

A robot modellt ROS környezetben URDF robot leíró, xml alapú fájjal lehetjük meg, [9] [10] [11]. Az <origin> tag az xzy paraméter alatt, megadhatjuk a csukló pozíóját mindenáron tengelyen, méterben kifejezve a <parent> tagban szereplő linkhez képest. Az rpy paraméter alatt az elfordulásokat rendre x, y, z tengelyek mentén radiánban ki- fejezve. Az <axis> tagban beállíthatjuk a kényszereket, jelen esetben csak az y tengely körüli forgás engedélyezett, azaz a kerekek esetében. Az <link> tagban robot elemeket hozhatunk létre. Az alábbi XML-ben látható a robot fizikai leírása, amely megfelel a valós szerkezetnek.

```
<robot name="mobile_robot_platform_4Wheel">
    <link name="base_link" > </link>
    <link name="FL_link" > </link>
    <link name="BR_link" > </link>
    <link name="BL_link" > </link>
    <link name="BODY_link"> </link>
    <link name="ImuALink"> </link>
    <link name="laser"> </link>

    <joint name="FL" type="continuous">
        <parent link="BODY_link"/>
        <child link="FL_link"/>
        <origin xyz="0.29 -0.33 0" rpy="0 0 0" />
        <axis xyz="0 1 0" />
    </joint>

    <joint name="FR" type="continuous">
        <parent link="BODY_link"/>
```

---

```

        <child link="FR_link"/>
        <origin xyz="0.29 0.330 0" rpy="0 0 0" />
        <axis xyz="0 1 0" />
    </joint>

    <joint name="BL" type="continuous">
        <parent link="BODY_link"/>
        <child link="BL_link"/>
        <origin xyz="-0.29 -0.330 0" rpy="0 0 0" />
        <axis xyz="0 1 0" />
    </joint>

    <joint name="BR" type="continuous">
        <parent link="BODY_link"/>
        <child link="BR_link"/>
        <origin xyz="-0.29 0.330 0" rpy="0 0 0" />
        <axis xyz="0 1 0"/>
    </joint>

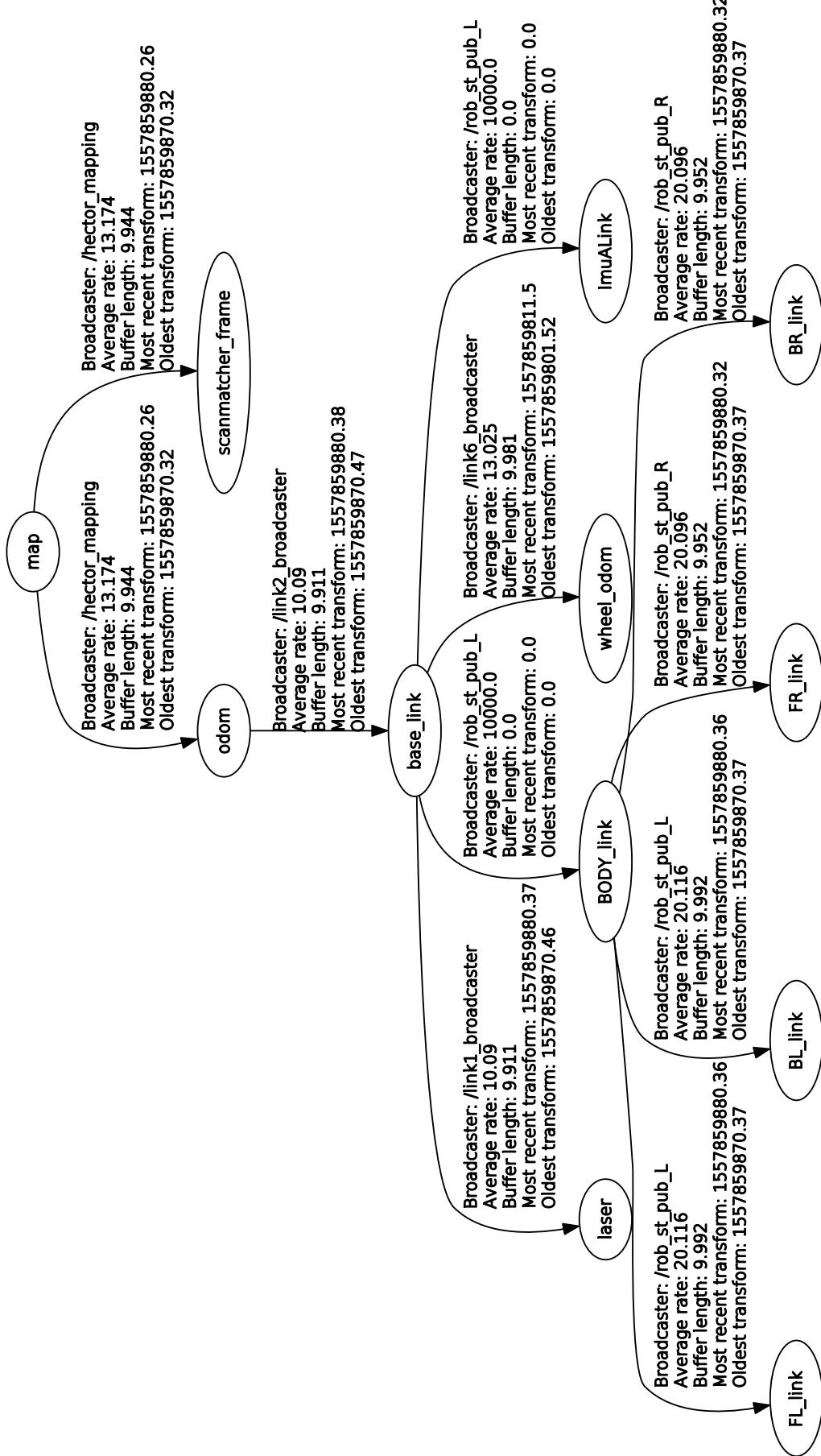
    <joint name="imuAandGPS" type="fixed">
        <parent link="base_link"/>
        <child link="ImuALink"/>
        <origin xyz="0.125 0.03 0.11" rpy="0 0 0" />
    </joint>

    <joint name="laserAJoin" type="fixed">
        <parent link="base_link"/>
        <child link="laser"/>
        <origin xyz="0.39 -0.02 0.23" rpy="0 0 3.14" />
    </joint>

    <joint name="contact" type="fixed">
        <parent link="base_link"/>
        <child link="BODY_link"/>
    </joint>
</robot>
```

A ábra 3.3.12 láthatjuk, hogy a robot törzsét a *BODY\_link* alkotja, amelyhez kapcsolódnak a kerekek: *BL\_link*, *FL\_link*, *BR\_link*, *FR\_link*. A *base\_link* és a *BODY\_link* egybe esnek. A szenzorok a *laser*, amely a lidarnak felel meg, *ImuALink* IMU szenzor ezek a *base\_link*-hez kapcsolódnak. A *map* VNR a térképnak, amelyen meghatározzuk a robot pozícióját *odom*.

Recorded at time: 1557859811.5

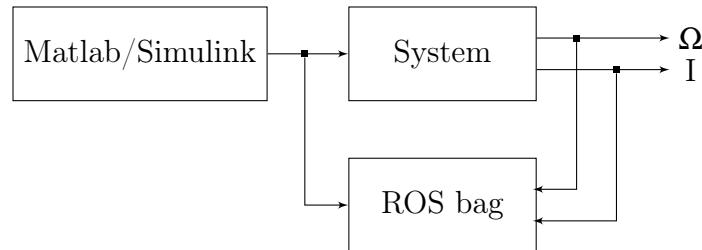


3.12. ábra. A megvalósított robot VNR-k közti reláció

### 3.4. Kerekek Pid Szabalyzo hangolas

A pid, a legelterjedtebb szabályozó egyszerű feladatok elvégzésére, esetünkben is elegendő a kerekek szögsebesség szabályzására kerekenkénti egy PID szabályzóval. A PID szoftveresen fut a uBlaze processzoron. Bemenete egy előírt forgási sebesség °/s -ban és kimenete egy -32000 és 32000 egész típusú értek. A kimenti érteke a PWM kitöltési tényezőt jelenti, az előjel pedig a beavatkozás irányát. Matlab/Simulink környezetben használva a Robotix Toolbox segítségével direktben pwm beavatkozó referencia érteket írtam elő a motoroknak. A beavatkozó jel előállítása és elküldési a fizikai eszköznek 0-100%-ig 10% lépcsőkben, amelyek 0% kitöltésekkel vannak megszakítva. A mért adatokat rosbag csomagba mentve majd a System Identification Toolbox használatával identifikáljuk a rendszer modellt. A rendszer bemenete egy beavatkozó jel, ami fizikailag feszültségnek fele meg 0V és 12V között. A kimenetek a forgási sebesség. A mért adatokat Matlab/System Identification használatával megbecsüljük a rendszer modellekét. Nemlinearis modellt becslünk Hammerstein-Wiener model [16] használva, 1 kimenet és 1 bemenet, a lineáris átviteli függvény fokszáma: zérusok nb = 2, pólusok nf = 3, késés a bemenet és a kimenet között nk = 1. A becsült adatok 94%-ban megfelelnek a mért rendszernek.

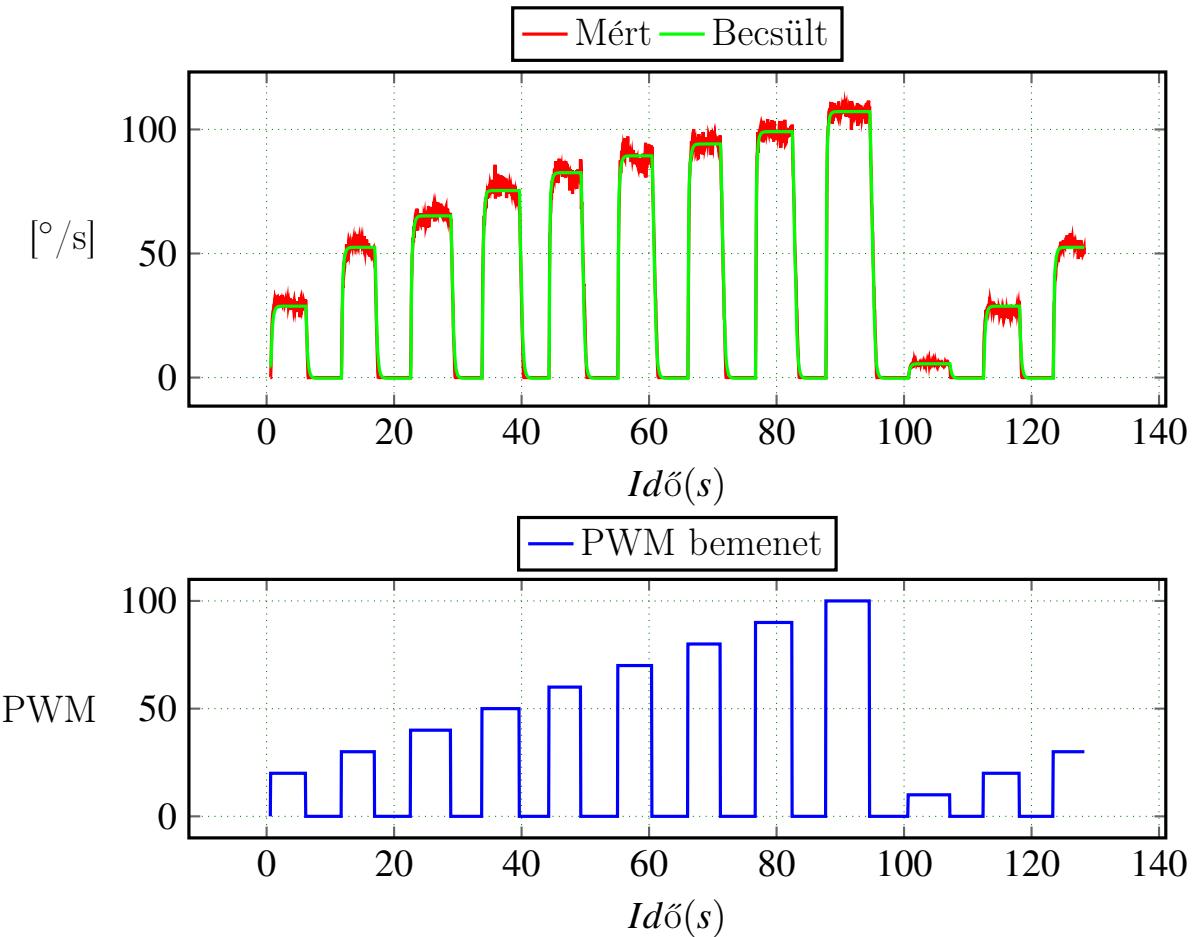
A méréseket a robot kerelei és a talaj érintkezése nélkül végeztem. A becsült modellt a bemenet 50/% körül linearizáljuk és a linearizált modellből átviteli függvényt készítünk.  $tf = tf(linearize(model,16000))$ ; utasítást használva Matlab környezetben. A linearizált modellt Matlab/PidTuning eszközt használva behangolunk, kiszámítjuk a megfelelő PID szabályzó paramétereit.



A becsült rendszer átviteli függvénye  $H_s(z)$ , mintavételézesi periódus  $T_s = 0.05s$ .

Nagyobbik fokozatban

A becsült modellt összehasonlítva a mért értékkal a ábra 3.4.13, a modell nem lineáris becsült modell megfelel a mért értékeknek.



3.13. ábra. Nagy fokozat Hammerstein-Wiener becsült modell válasza és a mért értékek.

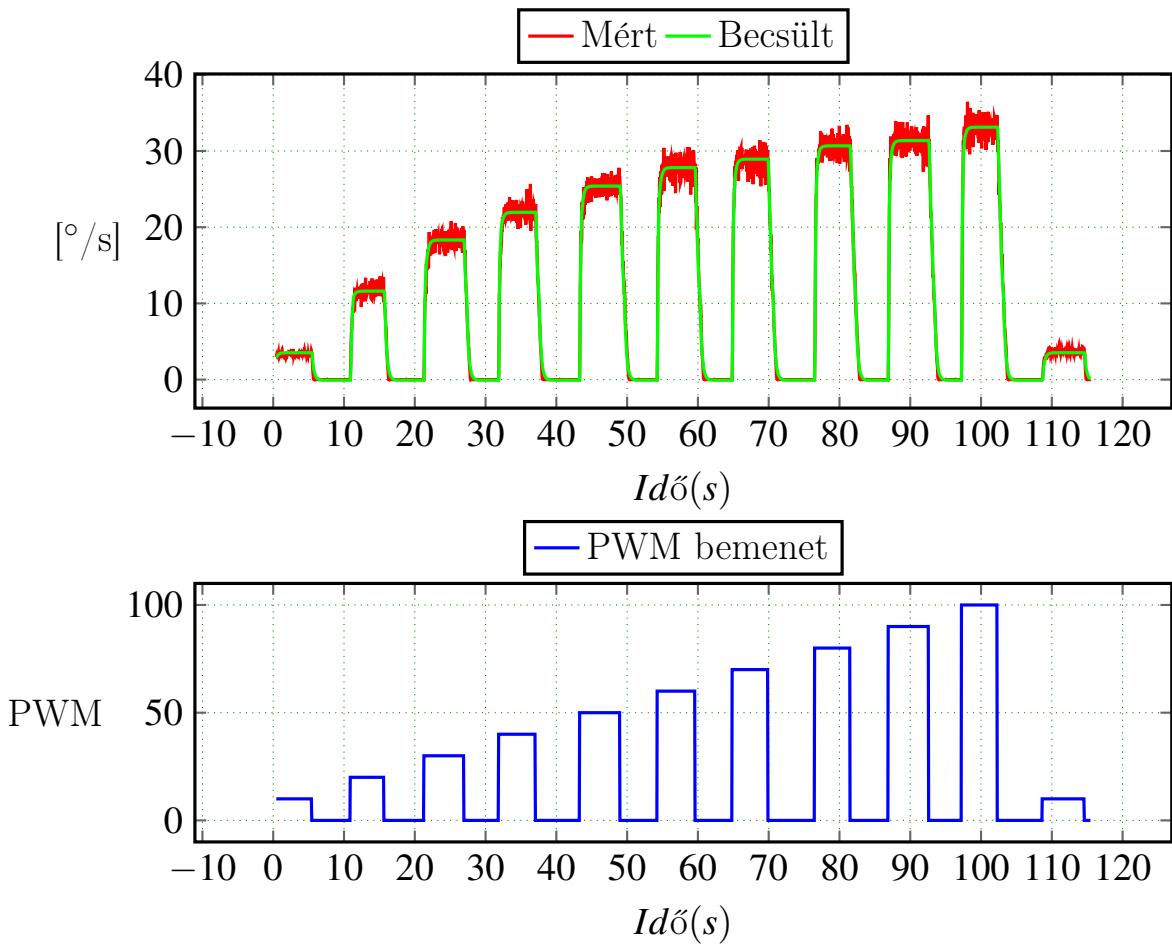
Az átviteli függvény a bemenet 50/% körül linearizálva.

$$H_s(z) = \frac{-0.07017z^{-2} - 0.053z^{-1}}{-0.2117^{-3} + 0.7321z^{-2} - 1.393z^{-1} + 1} \quad (3.2)$$

A tervezett PID szabályozó paramétere Kp: 7.11 , Ti: 23.66 , Td: 0.43

### 3.4.1. Kisebbik fokozatban

A becsült modellt összehasonlítva a mért értékkal a ábra 3.4.14, a modell nem lineáris becsült modell megfelel a mért értékeknek.



3.14. ábra. Kis fokozat Hammerstein-Wiener becsült modell válasza és a mért értékek.

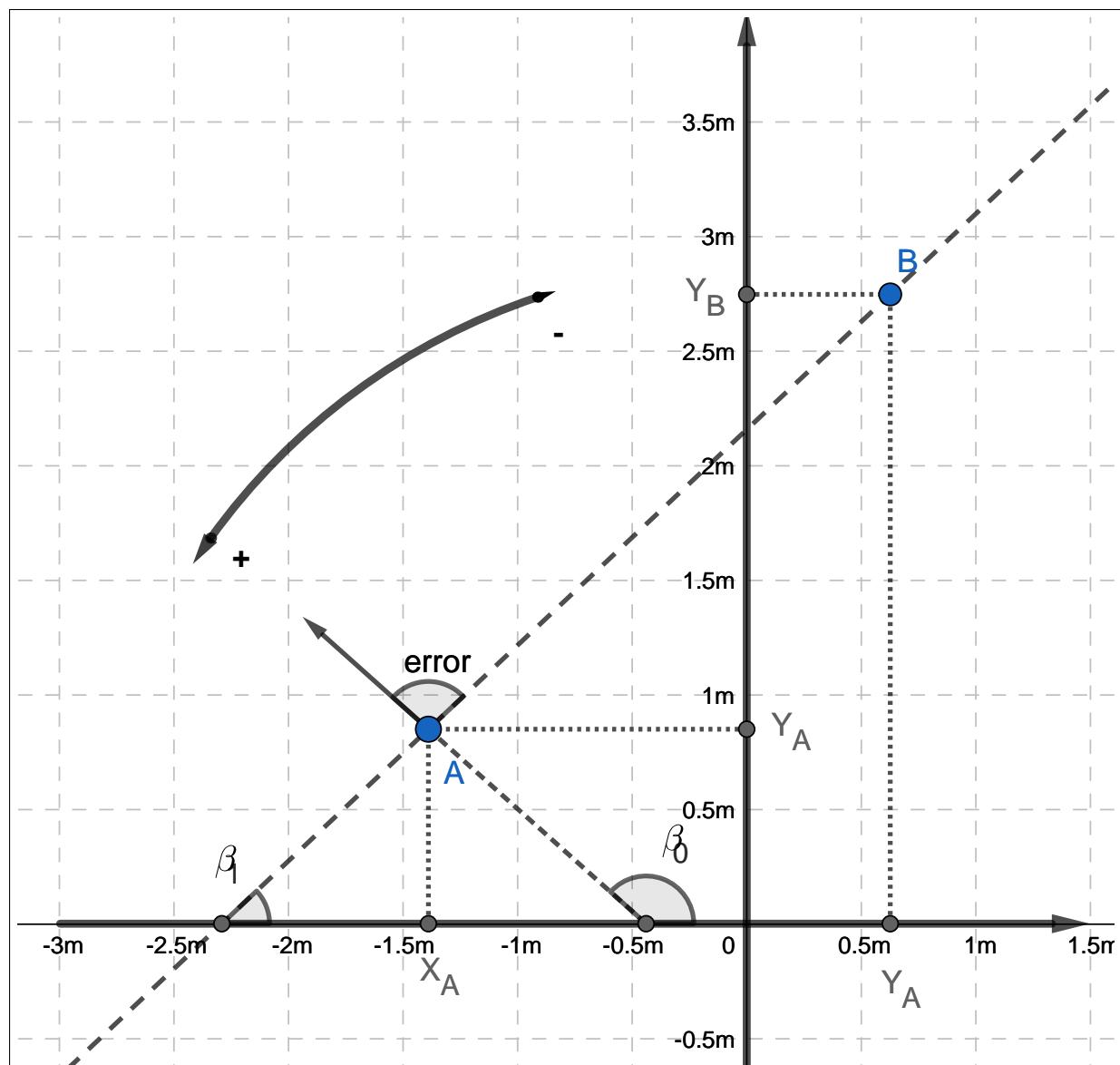
Az átviteli függvény a bemenet 50/% körül linearizálva.

$$H_s(z) = \frac{-0.0291z^{-2} - 0.009263z^{-1}}{-0.198z^{-3} + 0.7058z^{-2} - 1.394z^{-1} + 1} \quad (3.3)$$

A tervezett PID szabályozó paraméterek: Kp: 15.96 , Ti:51.51 , Td:1.237

### 3.5. Pályakövetési feladatok

A robot pályakövetési feladata megfogalmazhatjuk úgy hogy tudjunk eljutni A pontból B pontba ha ismerjük a robot síkbeli pozíciót és az orientációját egy térképen amely megfelel a környezetnek látható az ábra 3.5.15 ábrán. A ábra 3.5.15 ábrán látható hogy felveszünk egy pozitív és egy negatív irányt a szögre nézve, az orientációkat mindig  $[0^\circ, 360^\circ]$  között kell megadjuk. A robotnak mindenkorra kell fordulnia amerre a szög a legkisebb azért hogy minél kevesebbet keljen mozogjon. A kiinduló állapotok a robot kezdetben az A pontban van és az orientációja  $\beta_0$  és a B pontba szeretnénk eljutni egyenes vonalban. Így a robotnak kezdetben a célra kell fordulnia és ezután akadhat előre, miközben korrigálja a orientációs hibákat. Első lépésben a robotnak fordulnia kell *error* szöget hogy  $\beta_1$  irányba mutasson és ezután haladhat a cél fele.



3.15. ábra. Robot pozíció szemléldése

A pályakövetsí algoritmus látható alább. Az előírt irányt  $\tan^{-1}$  függvény segítségével határozzuk meg ismerve X és Y tengelyen a hibák nagyságát. A  $\text{Rotate}(e_\alpha, \Omega_{max})$  fövenyben alkalmazunk egy PI típusú szabályzót, melynek a feladata a szöghiba 0-hoz való. A

---

$Vx(d, V_{max})$  a robot lineáris sebességének a szabályzását látja el egy PI típusú szabályzó, célja a robot és a kitűzött cél távolságának a csökkentése. A távolságszabályzó bemenetét súlyozuk  $1/e_\alpha$  értékkel hogy ameddig nincs a robot irányban addig ne domináljon távolságszabályzó.

---

#### Algorithm 1 Pályakövetés Algoritmusa

---

```

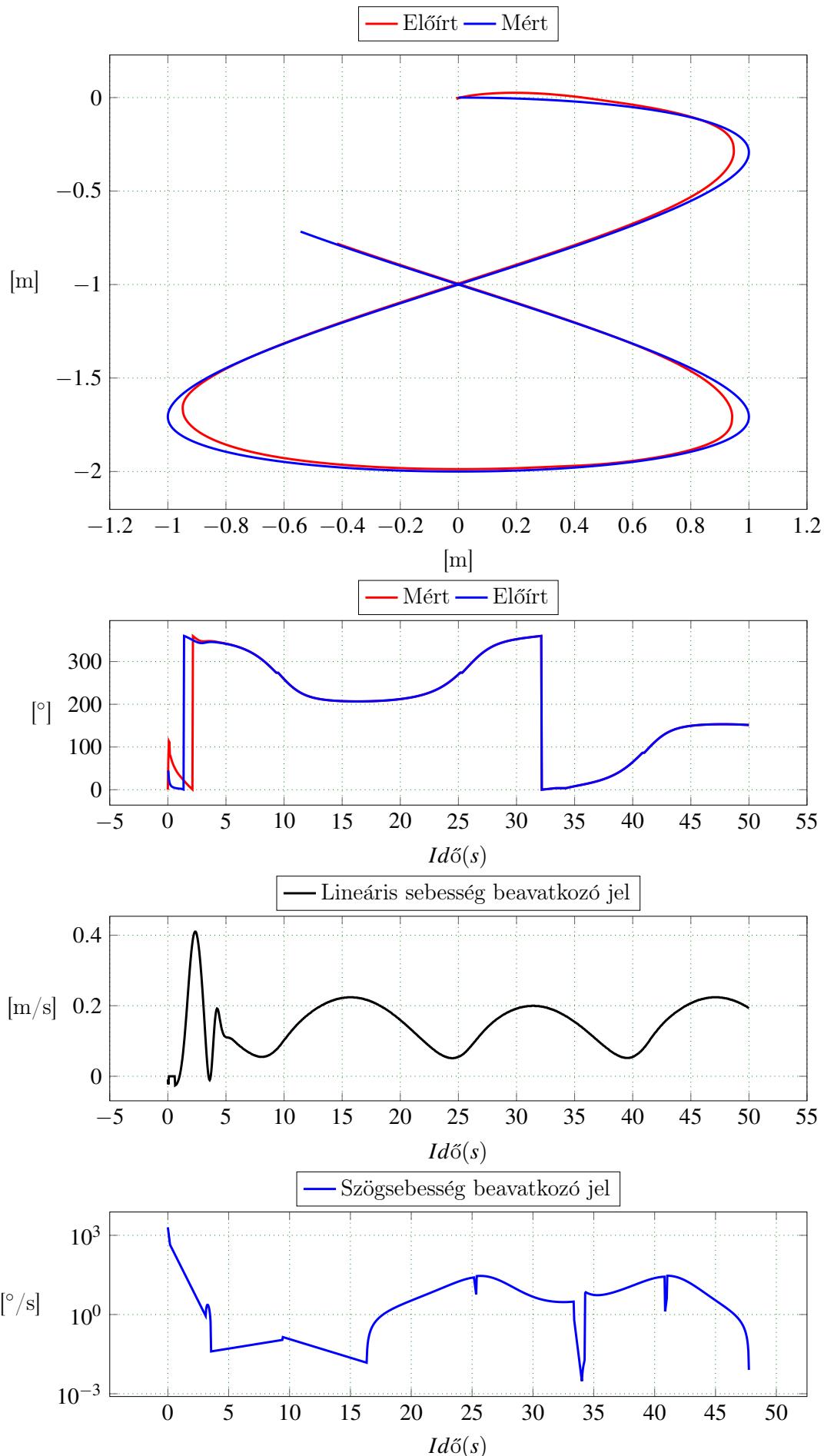
1: function GetNextControl( $X_a, Y_a, \alpha_a, X_t, Y_t, \alpha_t, Tr, Tr_\alpha, \Omega_{max}, V_{max}$ )
2:    $e_x = X_a - X_t$ 
3:    $e_y = Y_a - Y_t$ 
4:    $d = \sqrt{e_x^2 + e_y^2}$ 
5:    $\alpha_i = dir(X_a, Y_a, X_t, Y_t)$            ▷ Két ponton átmenő egyenes iránytényezője
6:    $e_\alpha = \alpha_i - \alpha_a$ 
7:   if  $e_\alpha > Tr_\alpha$  then                  ▷ Fordulj a cél felé
8:      $\Omega = Rotate(e_\alpha, \Omega_{max})$ 
9:      $V_x = 0$ 
10:    else                                     ▷ Haladj a cél felé és korrigáld az élfordulást
11:       $\Omega = Rotate(e_\alpha, \Omega_{max})$ 
12:       $V_x = Vx(d * 1/e_\alpha, V_{max})$ 
13:    end if
14:    if  $e_\alpha < Tr_\alpha$  és  $d < Tr$  then      ▷ Kívánt pozícióban
15:       $\Omega = 0$ 
16:       $V_x = 0$ 
17:    end if
18:  end function

```

---

Az algoritmus tesztelésére MATLAB/Simulink környezetet használtam. A robot kinematikai modellt az 2.14 egyenlet alapján modelleztem, a pozíciók ( $X, Y$ ) és a irány meghatározására integráltam a lineáris és szögsebességeket.

A ábra 3.5.16 ábrán láthatjuk a szimulációs eredményeket amint a robot egy végtelen jelhez hasonló pályát követ.



3.16. ábra. Robot pozíció szemlélődése

### 3.6. Mérések

Ebben a fejezetben tanulmányozásra kerül a robot viselkedése abban az esetekben, különböző terepviszonyok között. A Spinit mars roverrel (2006, Március, 13) [15] amikor az első jobb kereke meghibásodott. A megoldás az volt, hogy a robot mozgása optimálisabb lesz energia felhasználás szempontjából ha háttal megy. Az energia ellátása is véges volt, kiszolgáltatott volt a napsütésnek, a napelemekre rakodott por miatt csökkentek a azok hatásfoka így sokkal alaposabb mozgás pálya tervezésre volt szükség. Problémák adódtak a homokos talajjal is, a Spirit mars járónak, kerekei a homokba süllyedtek és beragadtak, a földi irányító csapat egy másolat segítségével próbálta kimozdítani a csapdából. A hasonló eseteket elkerülhetők lennének, ha ismerve a robot korlátait olyan mozgás pályát határoznak meg, amellyel elkerülhetjük ezen akadályokat, vagy időben detektálhatjuk ezen problémákat pl: homokba süllyedés érzékelése.

A robottal a következő méréseket fogjuk elvégezni:

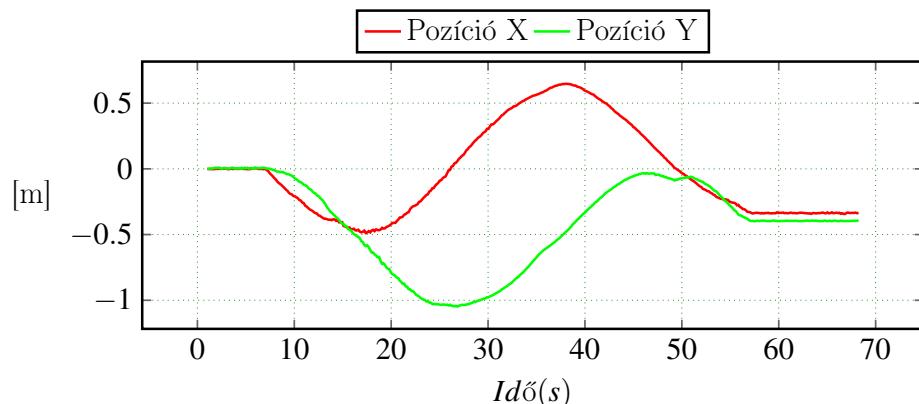
- (a) Helyben forgás súlypontja körül.
- (b) Körpályán haladás
- (c) Lépcsőn fel és le mozgások

#### 3.6.1. Differenciális Forgás Vízszintes Talajon

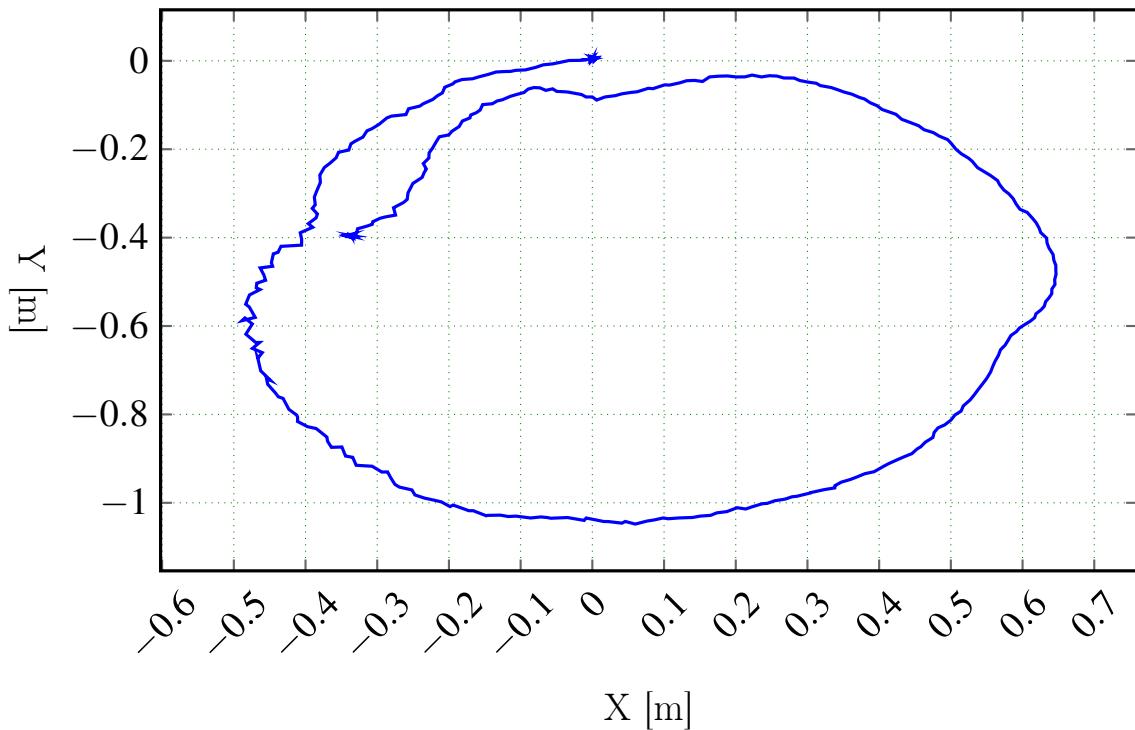
Differenciális forgásnak nevezzük azt, amikor a jobb és bal oldali kerekek sebessége megegyező, csak irányukban ellenkező, így a robot belső területen belül jön létre a ICR pont a COG közelében kellene legyen ábra 2.6.13.

#### 3.6.2. Féloldali kerekek blokkolva kavicsos talajon

A robot baloldali kerekei leblokkolva és a jobboldali kerekei  $50^\circ/\text{s}$  szögsebességgel forognak. Az eredmények alapján a ábra 3.6.18 látható a robot által leírt pálya. A mozgás során több mint  $360^\circ$ -t fordul és mondhatni körpályát írt le. A talajjal való súrlódások miatt a robot nem tökéletesen fordul ez látható abból is hogy a másodszori fordulás már az előzőhöz képest más középponttal rendelkezik.



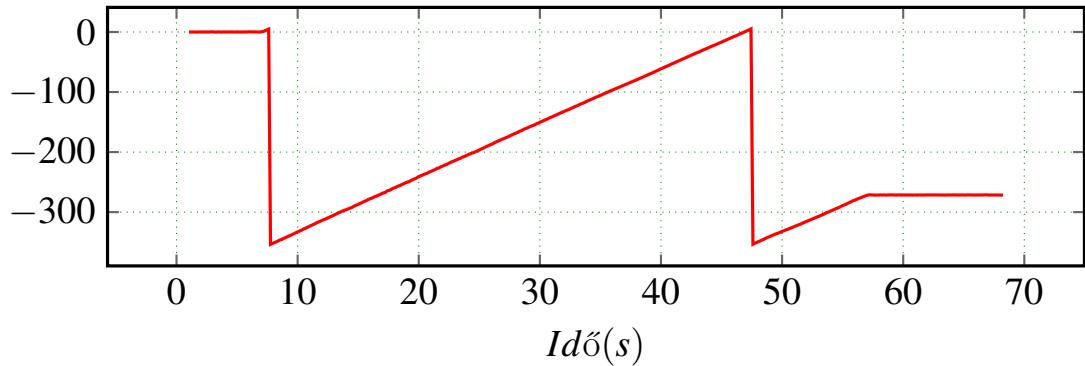
3.17. ábra. *SSMR-4W* típusú robot pozíciója, X és Y tengelyekre bontva, keréksebességek  $BL=FL=0$  és a  $FR=BR=50^\circ/\text{s}$



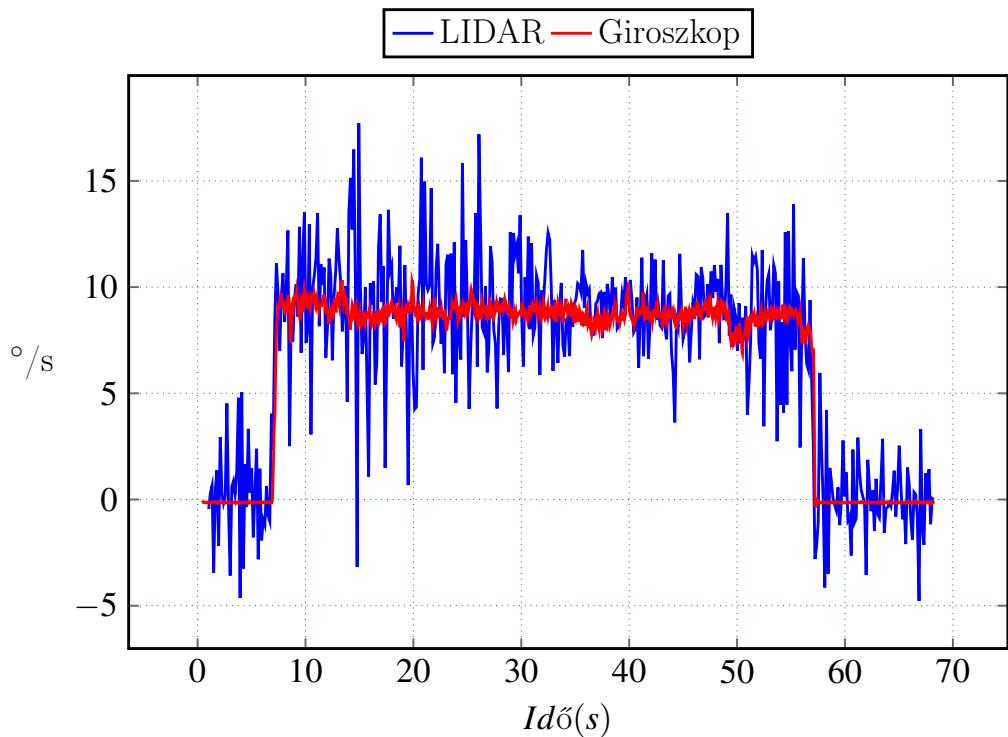
3.18. ábra. *SSMR – 4W* típusú robot által leírt pálya, kerekesebességek  $BL=FL=0$  és a  $FR=BR=50^\circ/s$

A mérés során a fordulási szögsebesség  $9^\circ/s$  látható a ábra 3.6.19 ábrán. A LIDAR és HectorMap segítségével mért abszolut szögsebesség zajosabb mint a giroszkóp által mért. A LIDAR-al mért szögsebesség előnyösebb mert a zajokat nem kell integrálni ahhoz hogy megkapjuk a szögsebességet a giroszkóppal ellentétben.

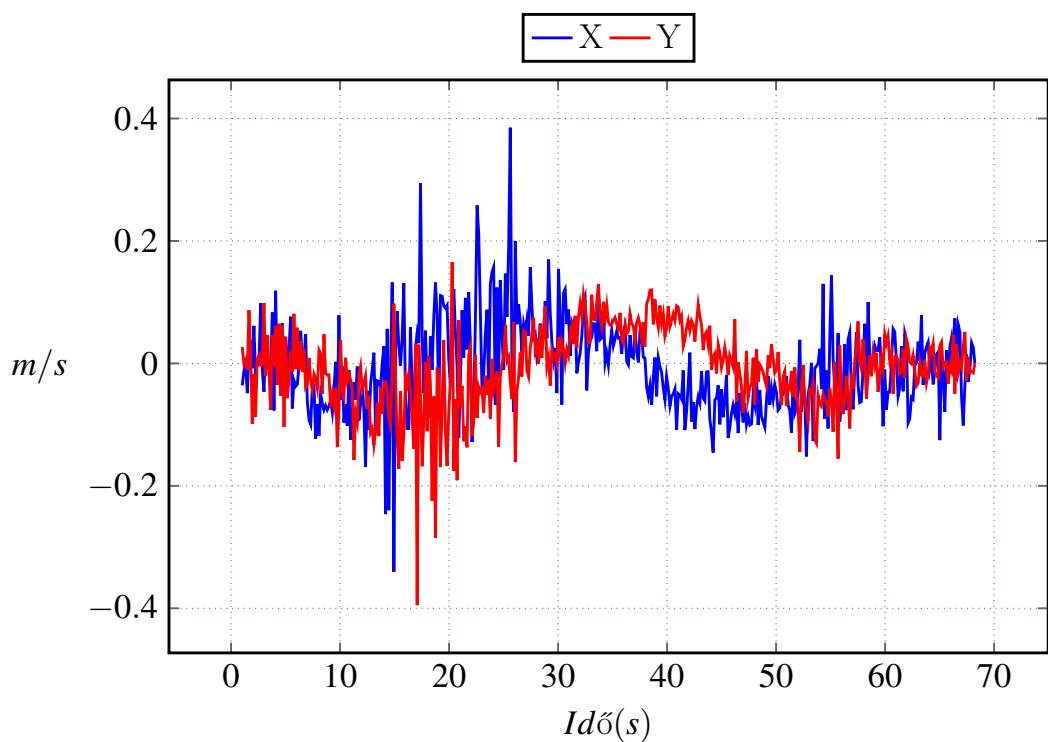
A lineáris sebességeket tekintve ábra 3.6.20 szinuszosan változnak, az X és Y tengelyeken, megfigyelhető egy  $90^\circ$  eltolódás az X és Y tengelyeken mért szinuszos mozgásban. A kerületi sebesség  $0.1 \text{ m/s}$  körül adható meg a forgás sugara  $0.5\text{m}$  körülire tehető.



3.19. ábra. *SSMR – 4W* típusú robot orientációja,ha a kerékszögsebességek  $BL=FL=0$  és a  $FR=BR=50^\circ/s$



3.20. ábra.  $SSMR - 4W$  típusú robot fordulási szögsebessége Giroszkóp és LIDAR által mért értékek, ha a kerékszögsebességek  $BL=FL=0$  és a  $FR=BR= 50^{\circ}/\text{s}$

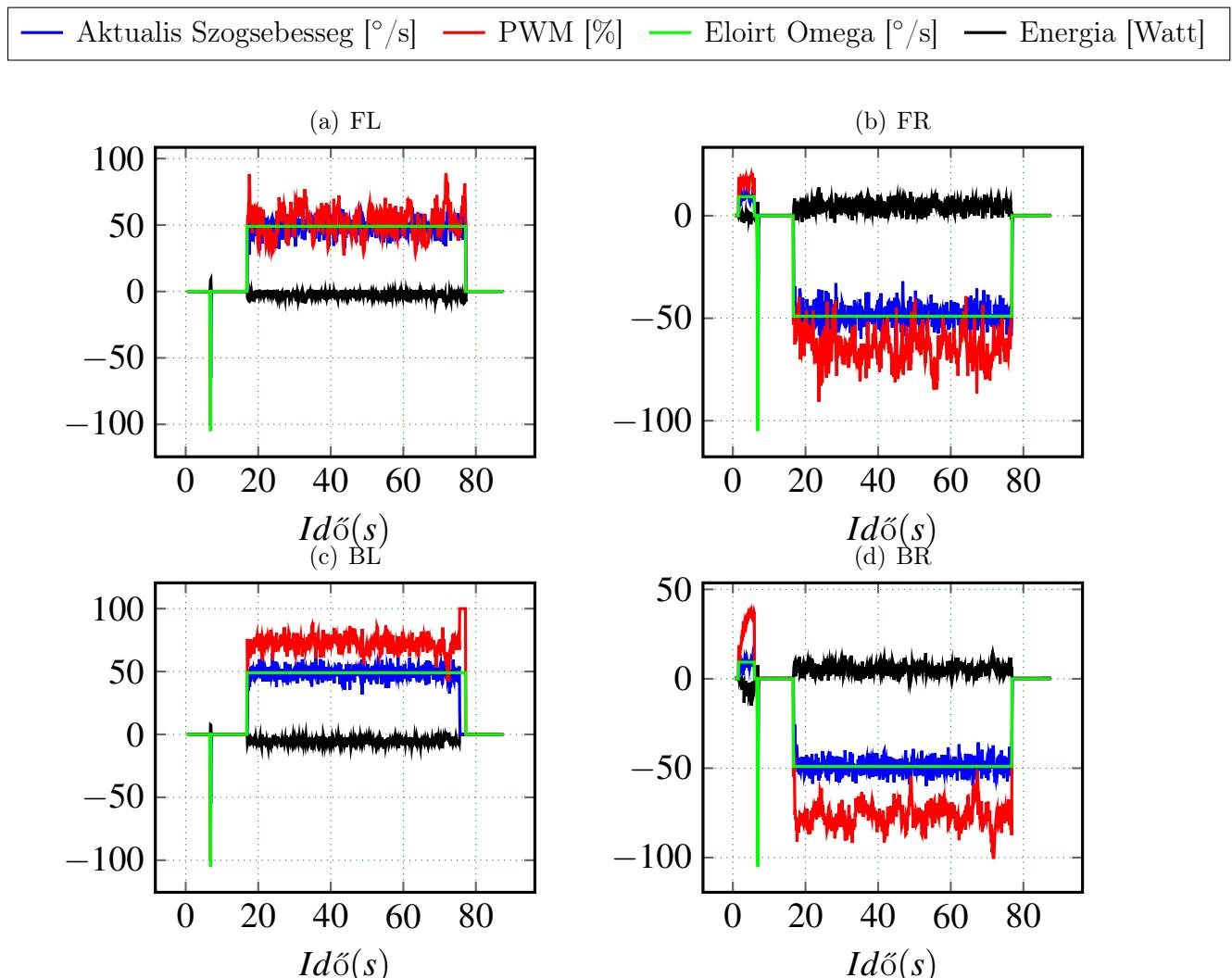


3.21. ábra.  $SSMR - 4W$  típusú robot súlypontjának sebessége a globális VNR-ben, X és Y tengelyekre bontva, ha a kerékszögsebességek  $BL=FL=0$  és a  $FR=BR= 50^{\circ}/\text{s}$

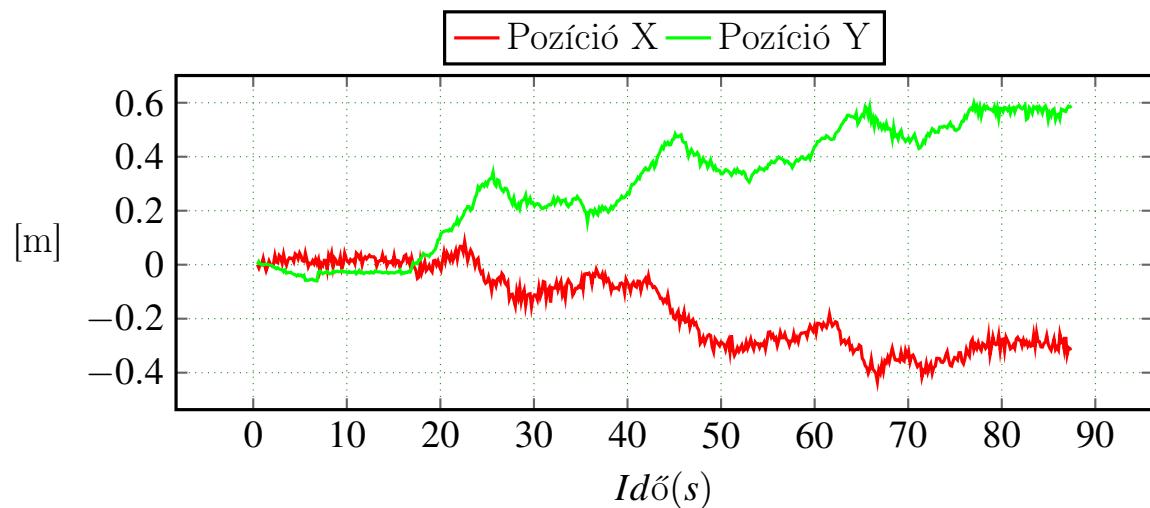
### 3.6.3. Kavicsos talajon helyben forgás

A ábra 3.6.23 megfigyelhető amint a robot kavicsos talajon differenciálisan fordul 60 másodpercen keresztül, ezalatt háromszor teljen körbefordul. A pályát tekintve a robot középpontja elmozdul, az X tengelyen 0.22m-t és a Y tengelyen 0.6m-t. Az oldalirányú mozgás a nem egyenlő surlodási erők miatt jön létre. A fordulás közben a kerekek követik az előírt referencia szögsebességeket, amint az ábra 3.6.22 ábrán is látható. A fordulási szögsebesség  $20^{\circ}/s$ , az X és Y tengelyen való sebesség elhanyagolható nagyságú, de jelen van mivel a robot forgási középpontja elmozdul ábra 3.6.24.

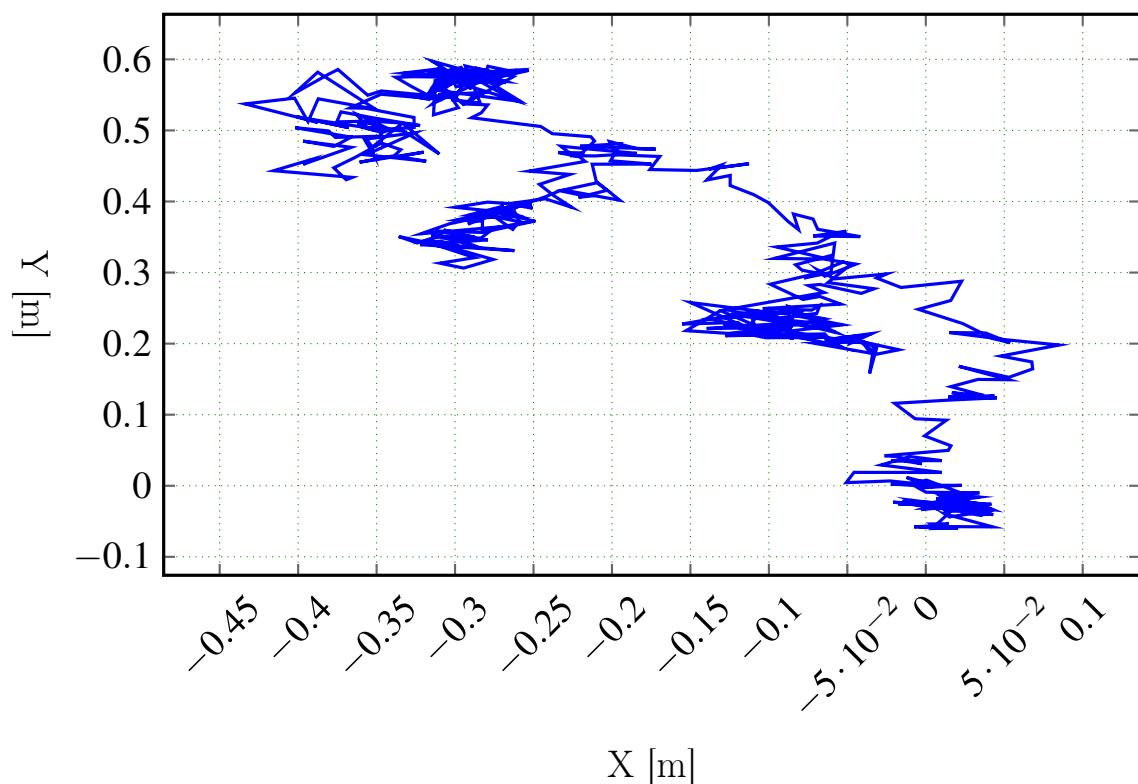
A talajon levő 0.8-1.1cm átmérőjű kavicsok miatt a súrlódások is megváltoznak, az kereke oldalirányú súrlódásai kisebbek lesznek, emiatt a robot könnyebben fordul. A kavicsok hasonlóképpen viselkednek a csapágyakban található görgőkhöz. Abban az esetben, ha a talajt a kavicsok nem teljes mértékben fedik be a ábra 3.6.23 látható mozgáspálya keletkezik, azáltal hogy csak néha kerül a kerék alá a gördülékeny kavics a négy kerék eltérő nyomatékot fog kifejteni és így a robot forgásközpontja is elmozdul.



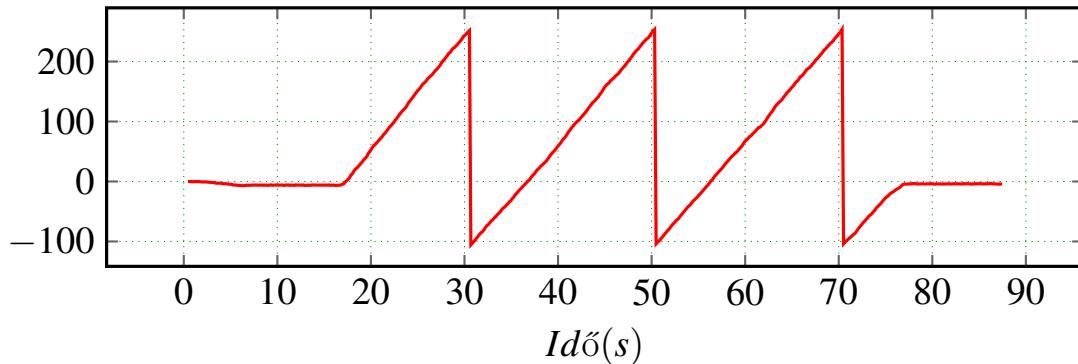
3.22. ábra. *SSMR – 4W* típusú robot mozgása, tengelyekre bontva, ha a kerék szögsebességek  $BL=FL=50^{\circ}/s$  és a  $FR=BR=-50^{\circ}/s$



3.23. ábra. *SSMR – 4W* típusú robot mozgása, ha a kerék szögsebességek  $BL=FL=50^\circ/s$  és a  $FR=BR=-50^\circ/s$

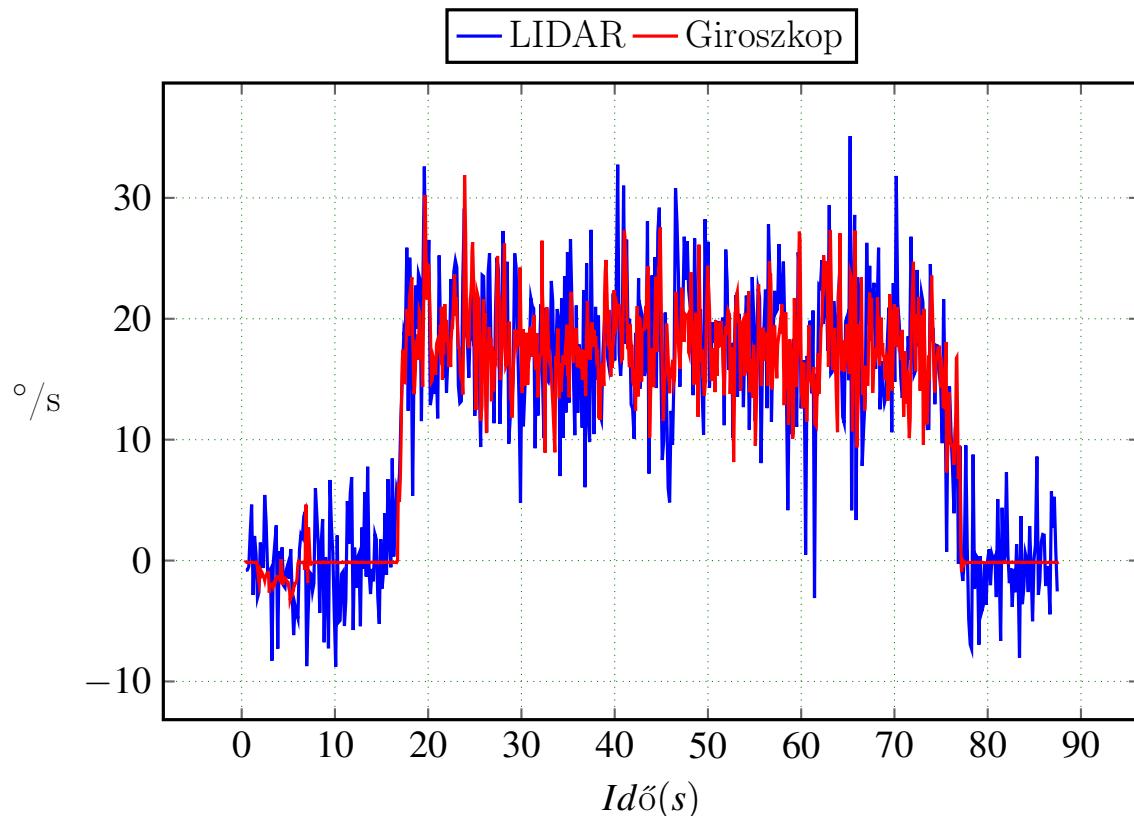


3.24. ábra. *SSMR – 4W* típusú robot által leírt pálya, ha a kerék szögsebességek  $BL=FL=50^\circ/s$  és a  $FR=BR=-50^\circ/s$



3.25. ábra. *SSMR-4W* típusú robot orientációja, ha a kerék szögsebességek  $BL=FL=50^\circ/s$  és a  $FR=BR=-50^\circ/s$

A ábra 3.6.20 és a ábra 3.6.26 összehasonlítva, megfigyelhető, hogy a Giroszkóp által mért érték zajosabb ha a robot fordulási sebessége nagyobb, míg a LIDAR és a HectorMap szögsebessége nem zajosodik ilyen mértékbén.



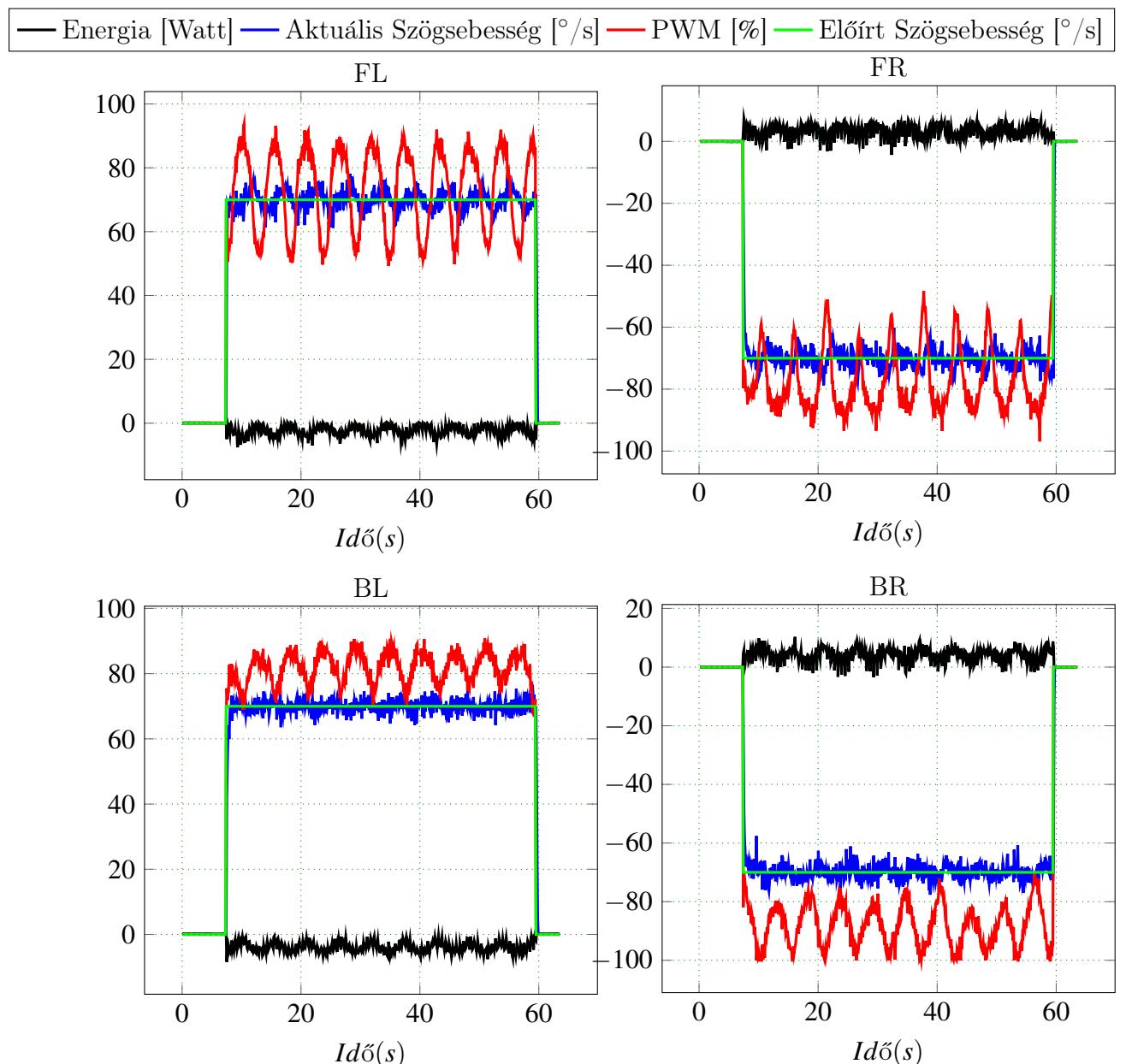
3.26. ábra. *SSMR-4W* típusú robot fordulási szögsebessége Z tengely körül, ha a kerék szögsebességek  $BL=FL=50^\circ/s$  és a  $FR=BR=-50^\circ/s$

### 3.6.4. Márvány padlón helyben forgás

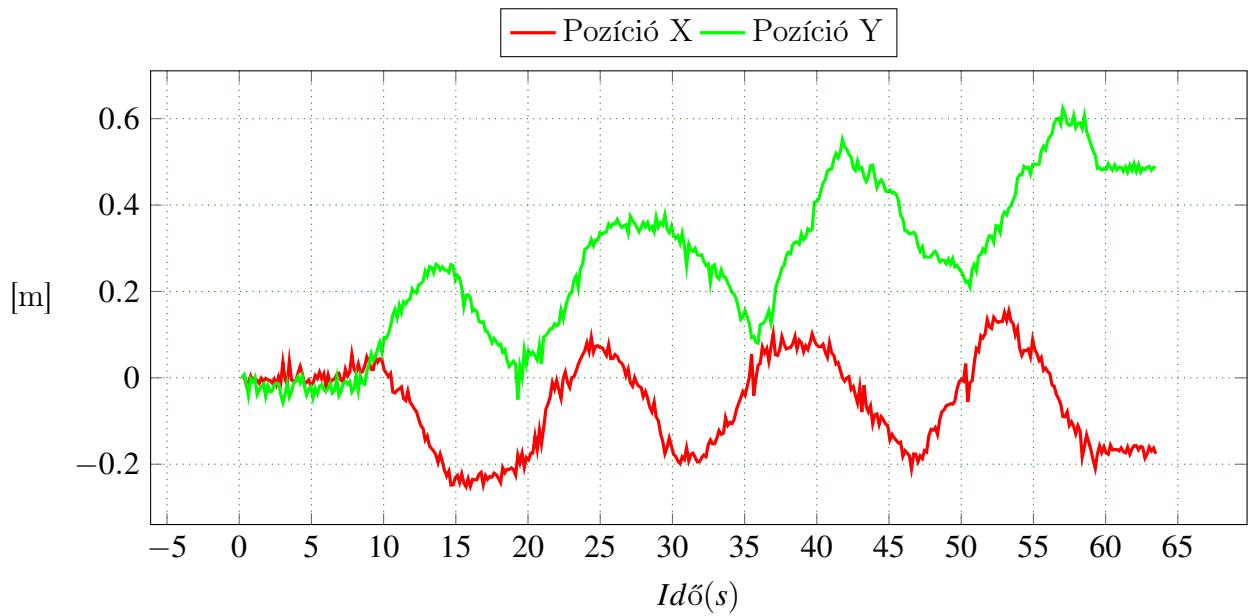
A ábra 3.6.28 megfigyelhető amint a robot márviny padlón differenciálisan fordul 50 másodpercen keresztül, ezalatt háromszor teljen körbefordul. A pályát tekintve a robot központja elmozdul, az X és a Y tengelyen is. Az oldalirányú mozgás a nem egyenlő súrlódási erők miatt jön létre. A fordulás közben a kerekek követik az előírt referencia

szögsebességeket, amint az ábra 3.6.27 ábrán is látható. A fordulási szögsebesség  $25^{\circ}/s$ , nagyobb mint a ábra 3.6.26, mivel nagyobb referencia értékek íródtak elő a kerekeknek.

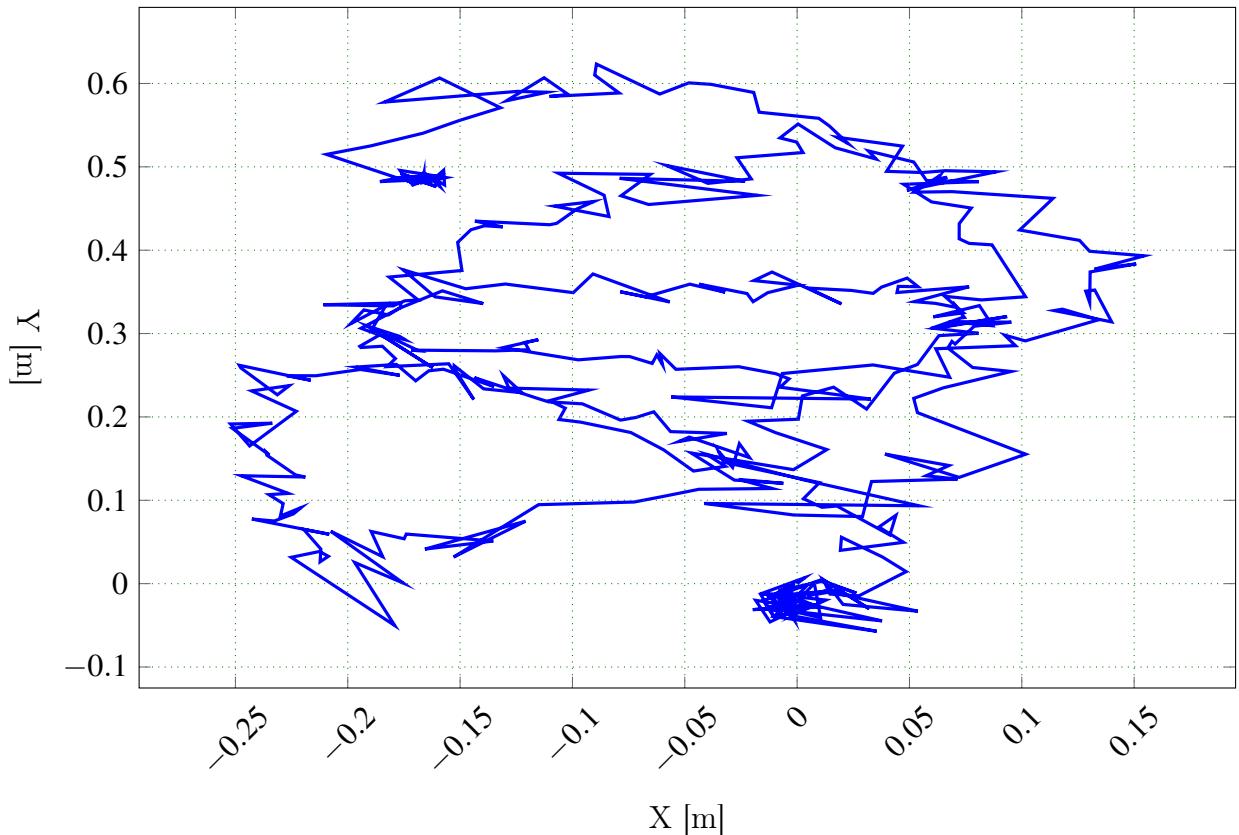
A szabályzok működése közben a kitöltési tényező hullámzik mind a négy kerék esetében ami a többi mérés esetében nem fordult elő.



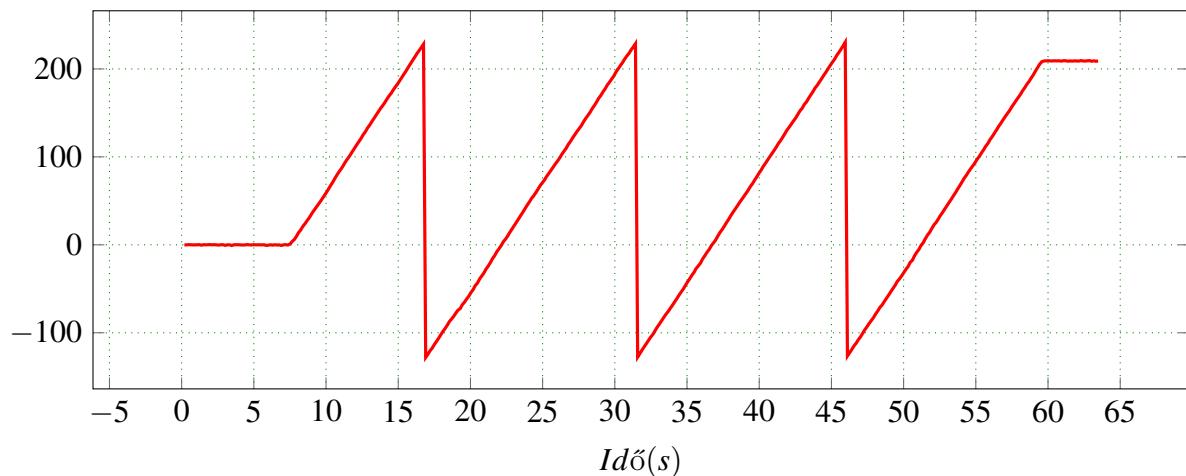
3.27. ábra. *SSMR – 4W* típusú robot mozgása, tengelyekre bontva, ha a kerék szögsebességek  $BL=FL=70^{\circ}/s$  és a  $FR=BR=-70^{\circ}/s$



3.28. ábra. *SSMR – 4W* típusú robot mozgása, ha a kerék szögsebességek  $BL=FL=70^\circ/s$  és a  $FR=BR=-70^\circ/s$

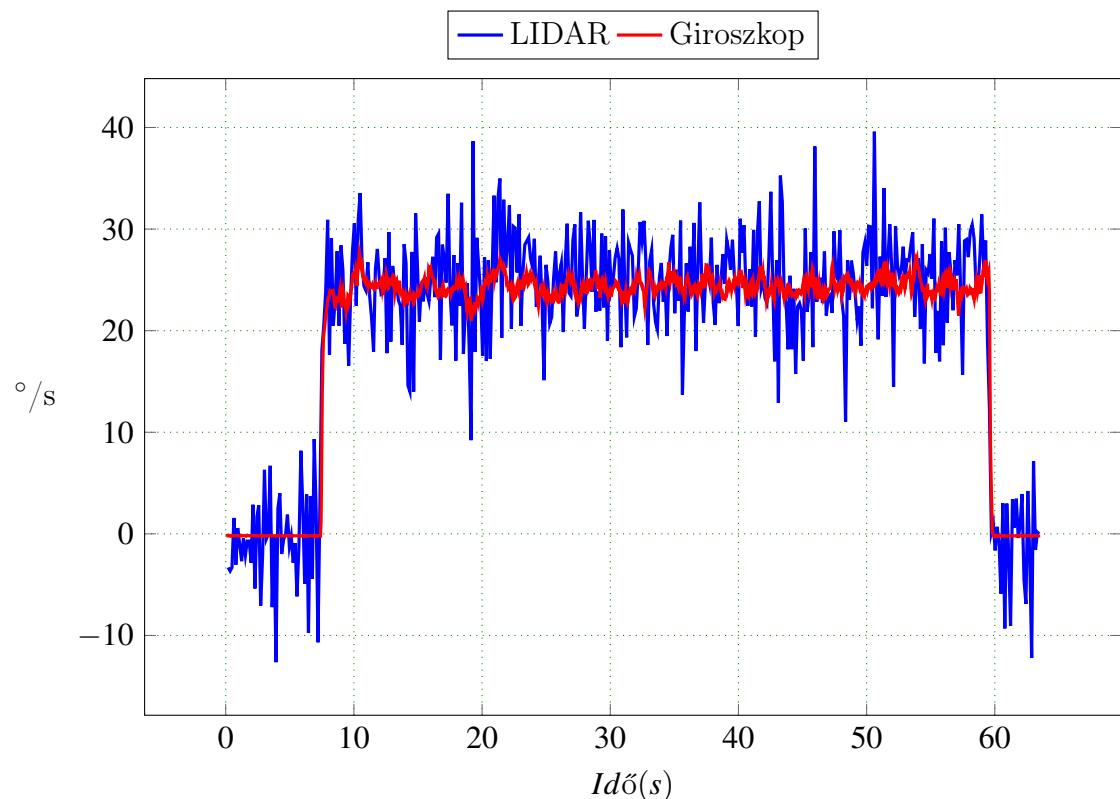


3.29. ábra. *SSMR – 4W* típusú robot által leírt pálya, ha a kerék szögsebességek  $BL=FL=70^\circ/s$  és a  $FR=BR=-70^\circ/s$



3.30. ábra. *SSMR-4W* típusú robot orientációja, ha a kerék szögsebességek  $BL=FL=70^\circ/s$  és a  $FR=BR=-70^\circ/s$

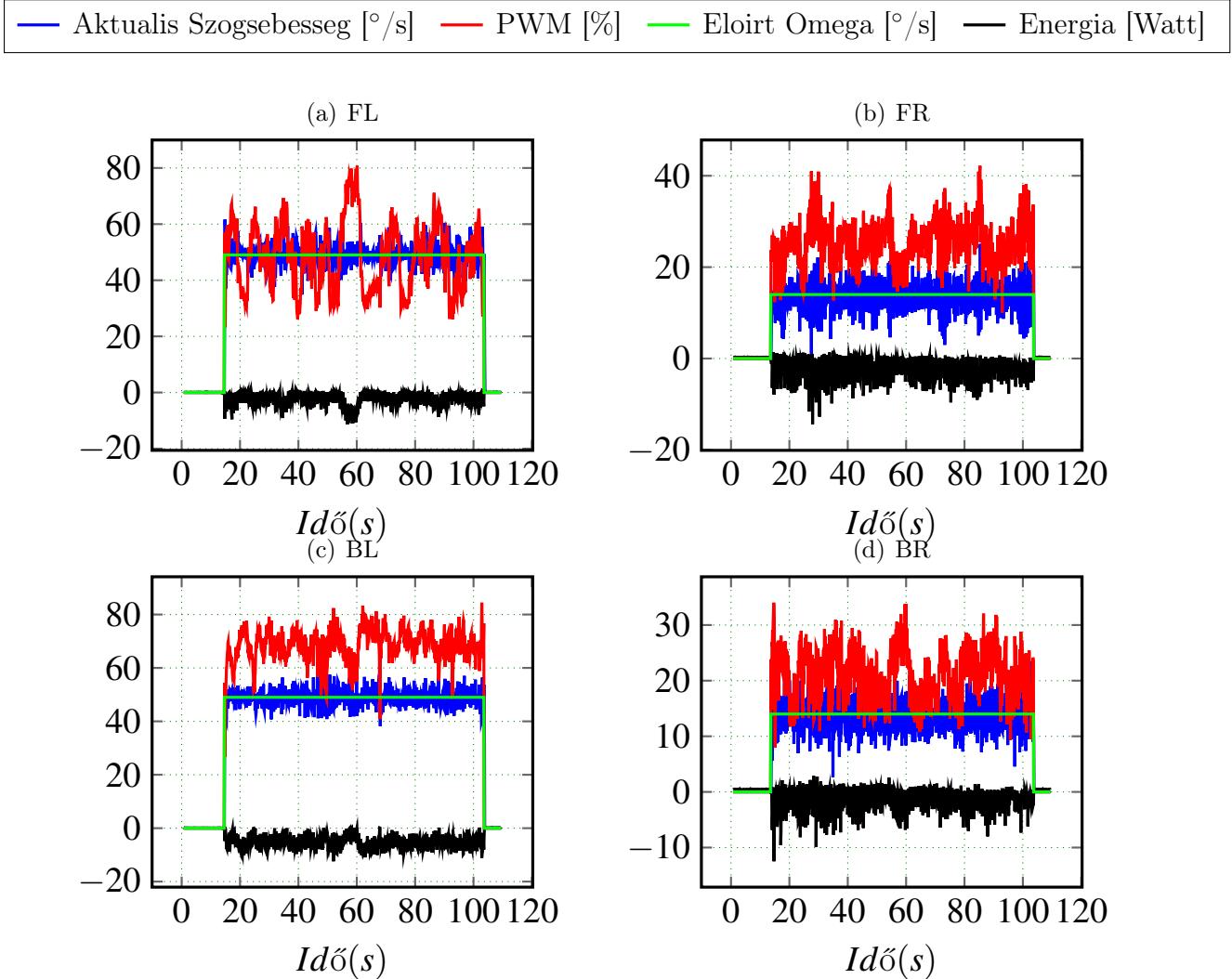
A ábra 3.6.31 megfigyelhető hogy a LIDAR által mért szögsebesség sokkal kevésbé zajos mint ábra 3.6.20, amiatt hogy a márvány padló sokkal egyenesebb mint a kavicsos talaj, így nincsenek jelen kisebb nagyobb bukkánok amik zajforrásként jelenek meg a LIDAR térképezés számára.



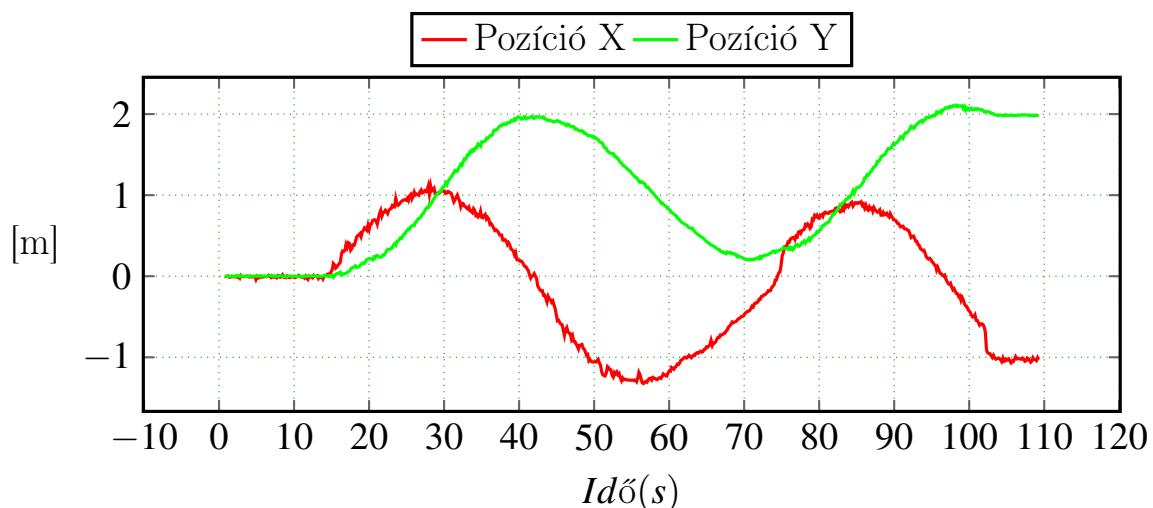
3.31. ábra. *SSMR-4W* típusú robot fordulási szögsebessége Z tengely körül, ha a kerék szögsebességek  $BL=FL=70^\circ/s$  és a  $FR=BR=-70^\circ/s$

### 3.6.5. Kavicsos talajon körpályán 50/15

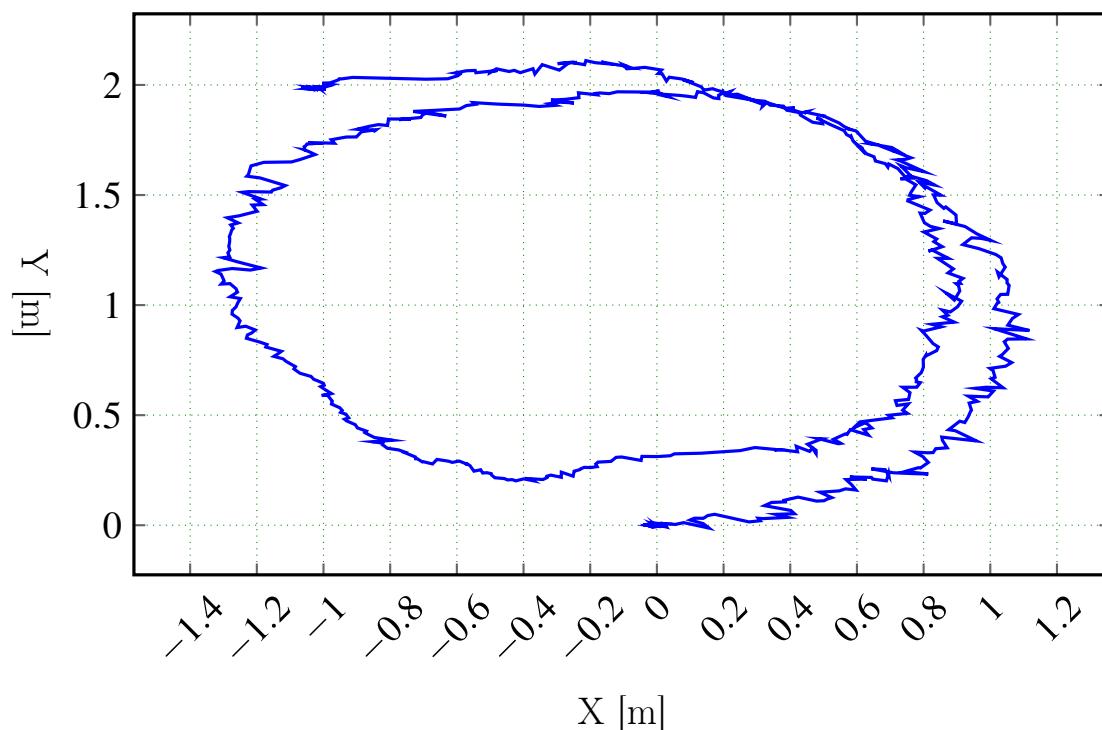
A ábra 3.6.33 megfigyelhető amint a robot kavicsos talajon differenciálisan fordul 80 másodpercen keresztül, ezalatt másfél szer körbefordul.



3.32. ábra. *SSMR – 4W* típusú robot mozgása, tengelyekre bontva, keréksebességek  $BL=FL=50^{\circ}/s$  és a  $FR=BR=15^{\circ}/s$

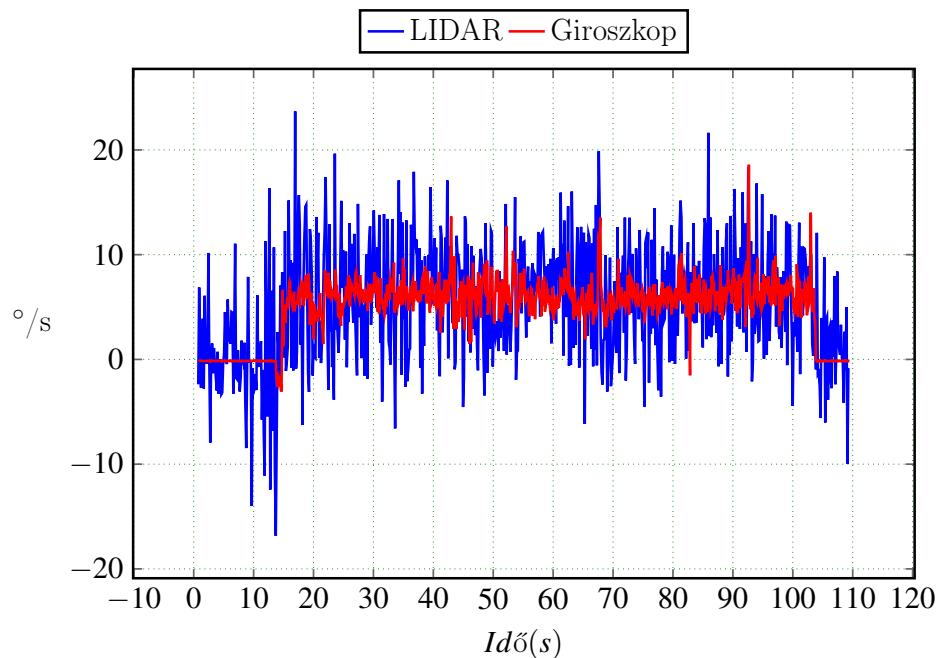


3.33. ábra. *SSMR – 4W* típusú robot mozgása, tengelyekre bontva, kerekszögsebességek  $BL=FL=50^{\circ}/s$  és a  $FR=BR=15^{\circ}/s$

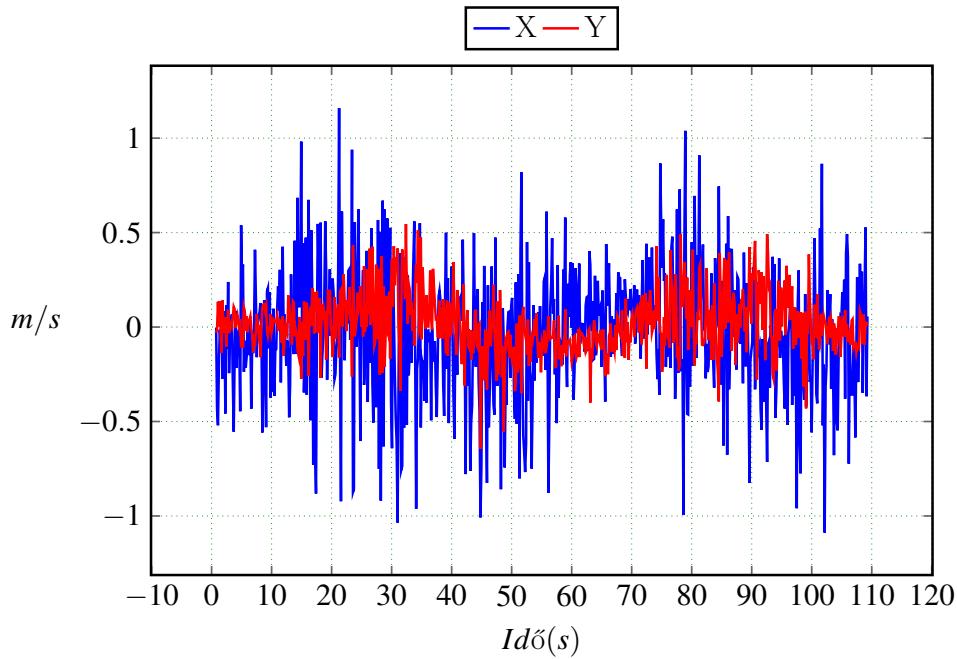


3.34. ábra. *SSMR – 4W* típusú robot által leírt pálya, kerekszögsebességek  $BL=FL=50^\circ/s$  és a  $FR=BR=15^\circ/s$

A körpályán, mozgás során a robot eltér a szabályos körtől, és látható a ábra 3.6.34 ábrán. A mérések nyilthurokan törénnek, nincs szabályzókör a pozicióra és a sebességekre. A szögsebességet tekintve a robot  $7^\circ/s$  szögsebességet generál.



3.35. ábra. *SSMR – 4W* típusú robot fordulási szögsebességek, kerekszögsebességek  $BL=FL=50^\circ/s$  és a  $FR=BR=15^\circ/s$

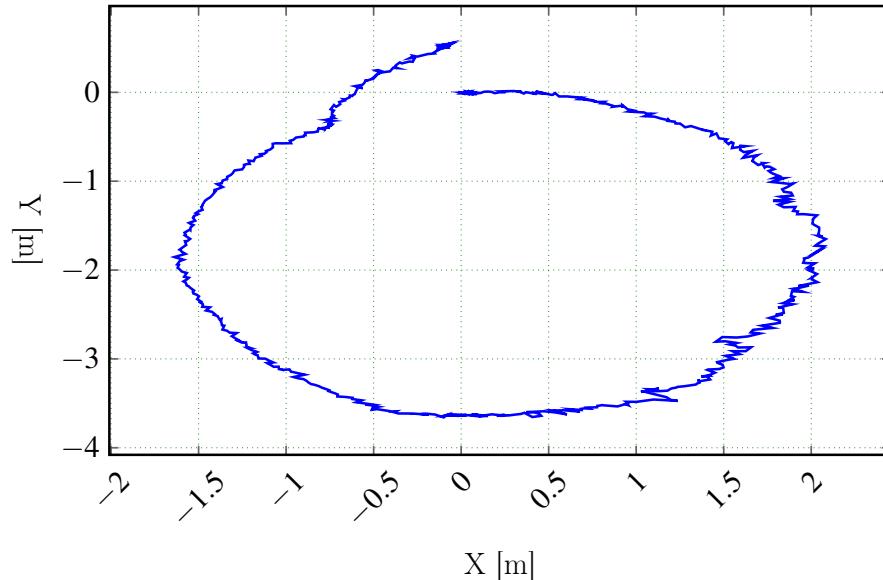


3.36. ábra.  $SSMR-4W$  típusú robot egyenesvonalú sebességei, kerekszögsebességek  $BL=FL=50^\circ/s$  és a  $FR=BR=15^\circ/s$

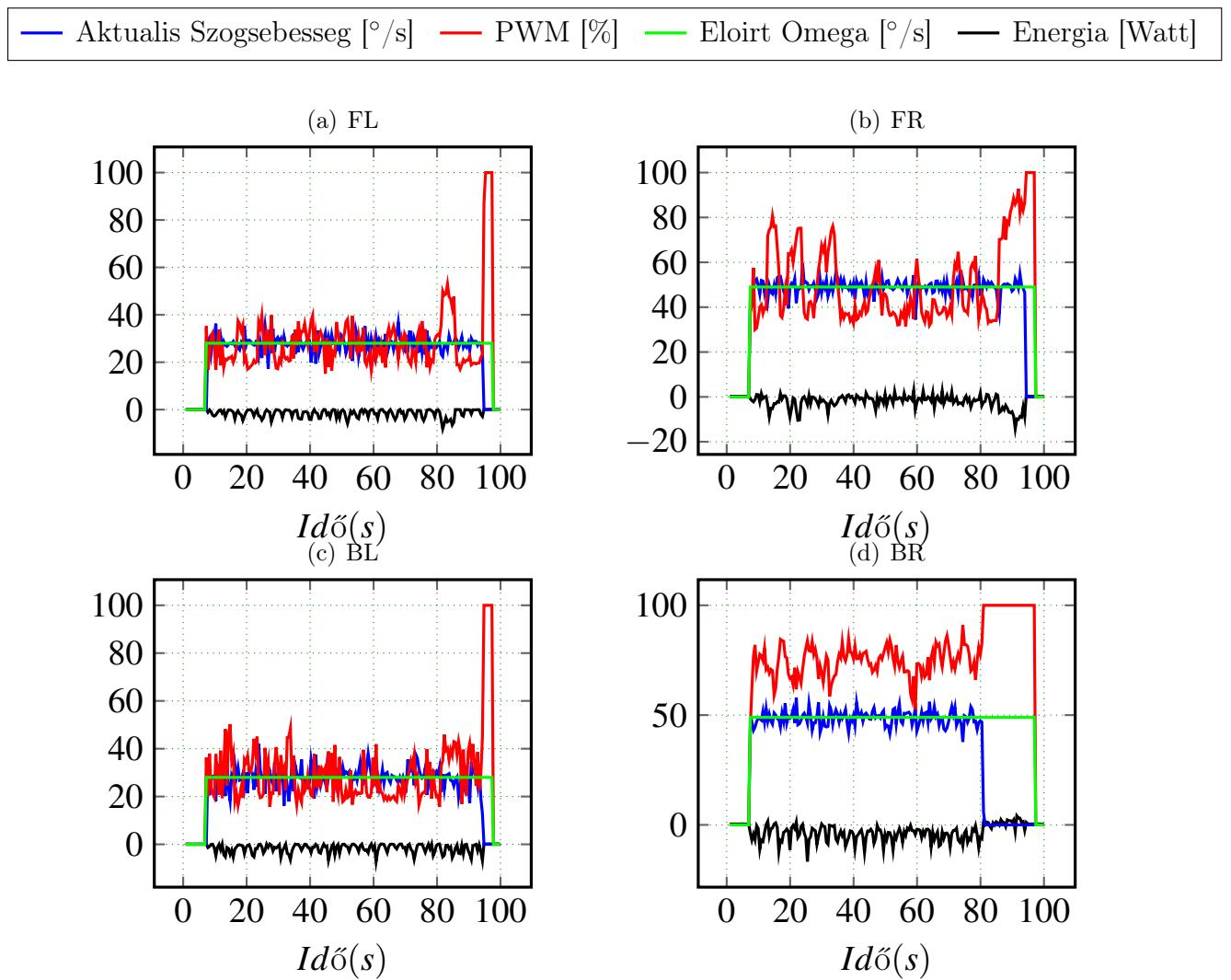
A robot sebességét tekintve a ábra 3.6.36 látható, hogy szinuszosan változik a pozícióhoz hasonlóan, a robot kis sebességű mozgása miatt a mérés zajos.

### 3.6.6. Kavicsos talajon körpályán 50/25

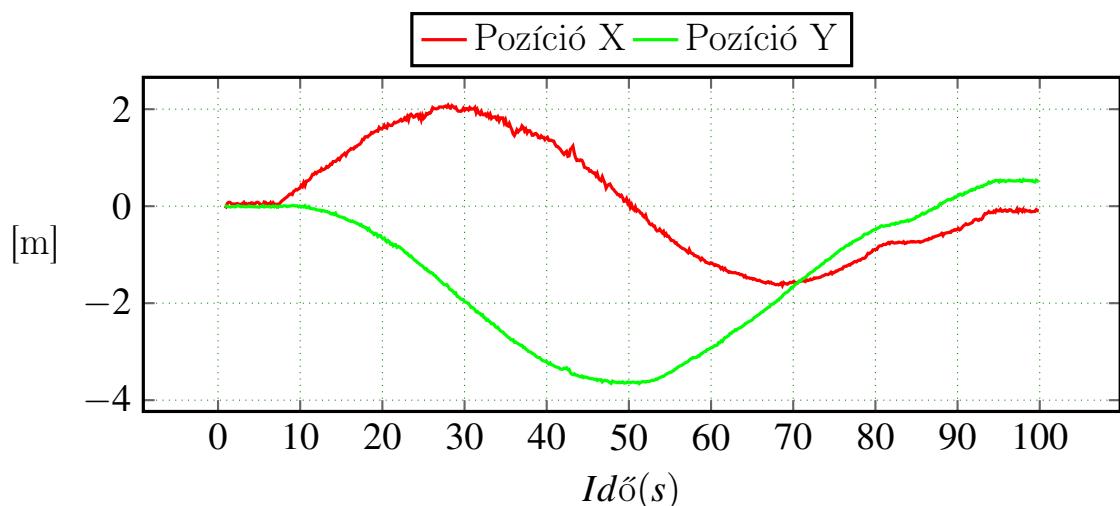
Az alábbi mérésekben 80-90-s között a robot jobboldali kerekeit vezérlő H-híd túlmelegedése miatt a beléjük épített védelmi funkcióknak köszönhetően leálltak így a jobboldali kerek leblokkoltak, így a mozgás pályája is megváltozott. A körpálya sugara 1.8m re tehető.



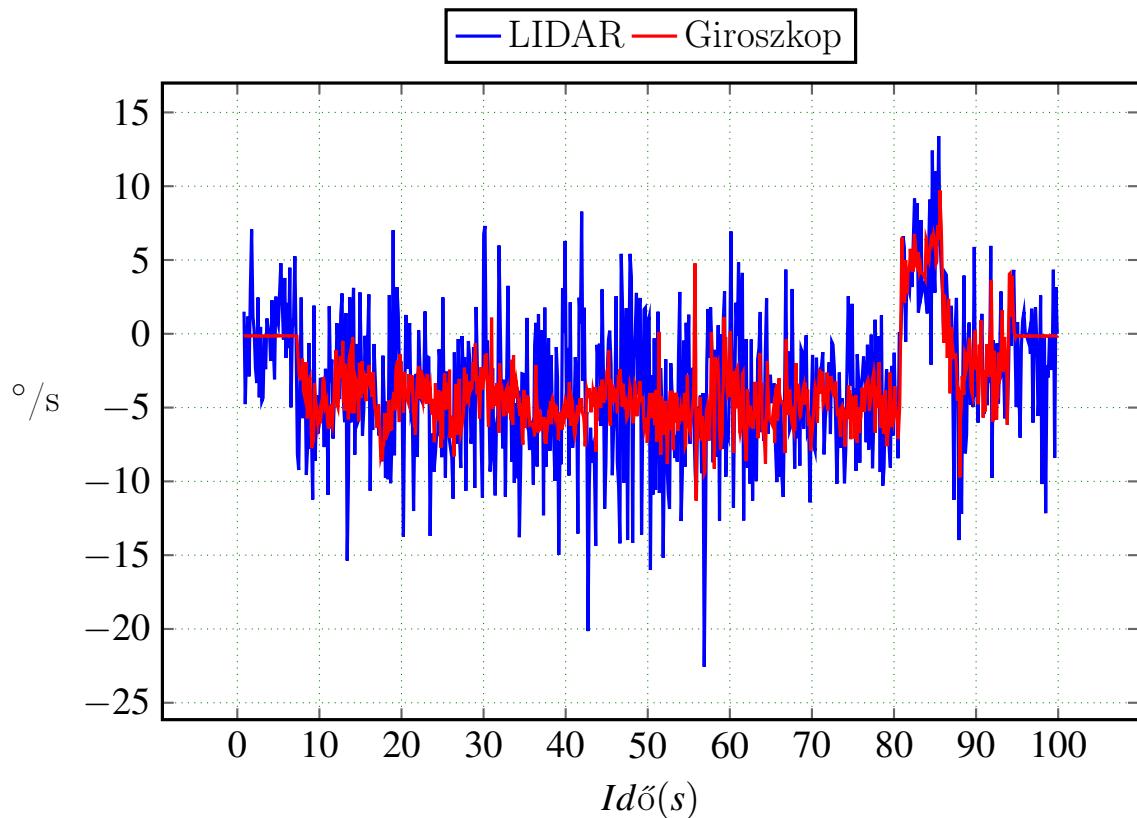
3.37. ábra.  $SSMR-4W$  típusú robot által leírt pálya, ha kerékszögsebességek  $BL=FL=25^\circ/s$  és a  $FR=BR=50^\circ/s$



3.38. ábra. *SSMR – 4W* típusú robot motorvezérlő jelei, ha kerékszögsebességek  $BL=FL=25^{\circ}/s$  és a  $FR=BR=50^{\circ}/s$

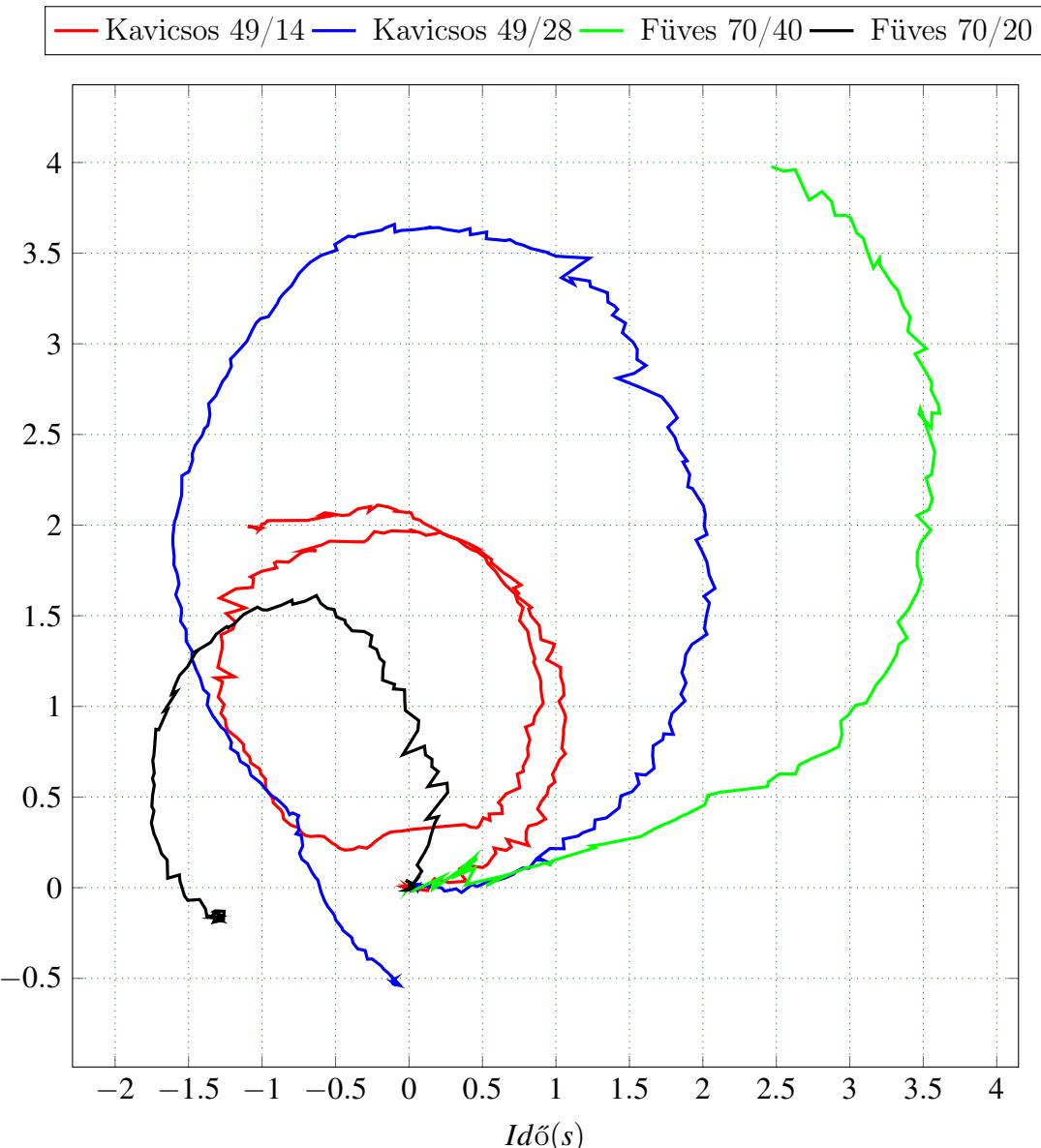


3.39. ábra. *SSMR – 4W* típusú robot mozgása, tengelyekre bontva, ha kerékszögsebességek  $BL=FL=25^{\circ}/s$  és a  $FR=BR=50^{\circ}/s$



3.40. ábra. *SSMR – 4W* típusú robot fordulási szögsebessége, ha kerékszögsebességek  $BL=FL=25^{\circ}/s$  és a  $FR=BR=50^{\circ}/s$

A lenti ábrán látható a kavicsos illetve füves terepen mért körpályák ábrázolása kétfelé körpályát irtunk elő, a füves talajon nagyobb sebességél haladtunk.



3.41. ábra. Kulombozo korpalyak

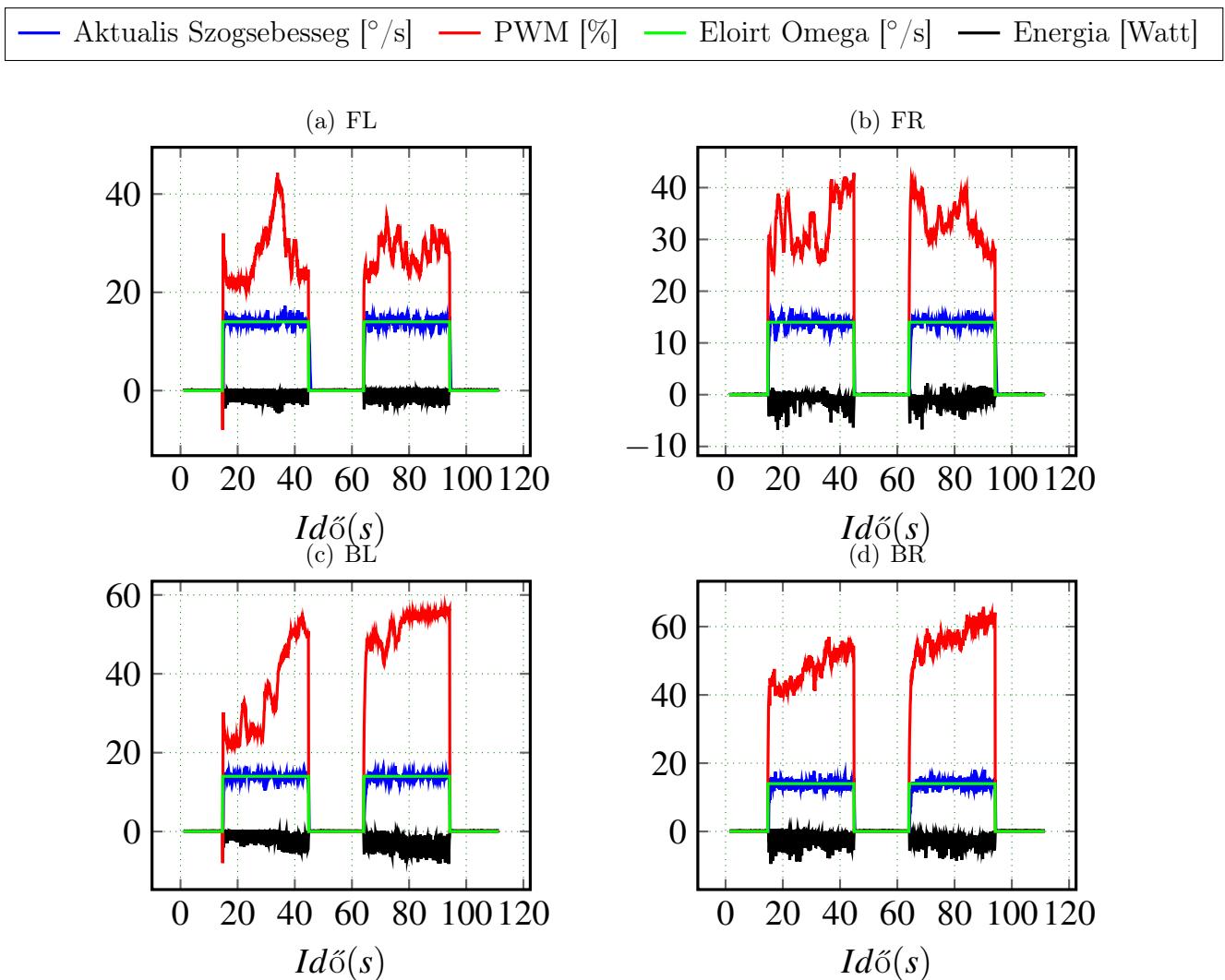
### 3.6.7. Homokos Lejtő

A lejtő meredeksége  $45^\circ$ , a szerkezete homokos, nagyobb méretű szilárd darabokkal, amelyek elérik a 4cm átmérőt is. A lejtőt az FL és FR kerekekkel közelítjük meg, így a BL és BR kerekekre nagyobb merőleges nyomóerő jut. A ábra 3.6.43 látható a PWM beavatkozó jelek a BL és BR kerekeken 20% nagyobb beavatkozó jelet szükséges ugyanannak a referencia jelnek a követésére.

A kereke előirt forgási sebessége  $15^\circ/s$ , 0.5m ábra 3.6.42 megtétele után a BL és a BR kerek 10 cm mélyen a homokba süllyedtek és a robot elakadt ábra 3.6.42. Egy másik észrevétel hogy a FL és FR kerek egyáltalán nem süllyedtek be a talajba forgás közben, ami a kisebb merőleges nyomóerőnek tulajdonítható.



3.42. ábra. Homokbat sülyet kerek 45°lejtön, 10 cm mélyen, elakadt robot a lejtőn 0.5m megtett ut után

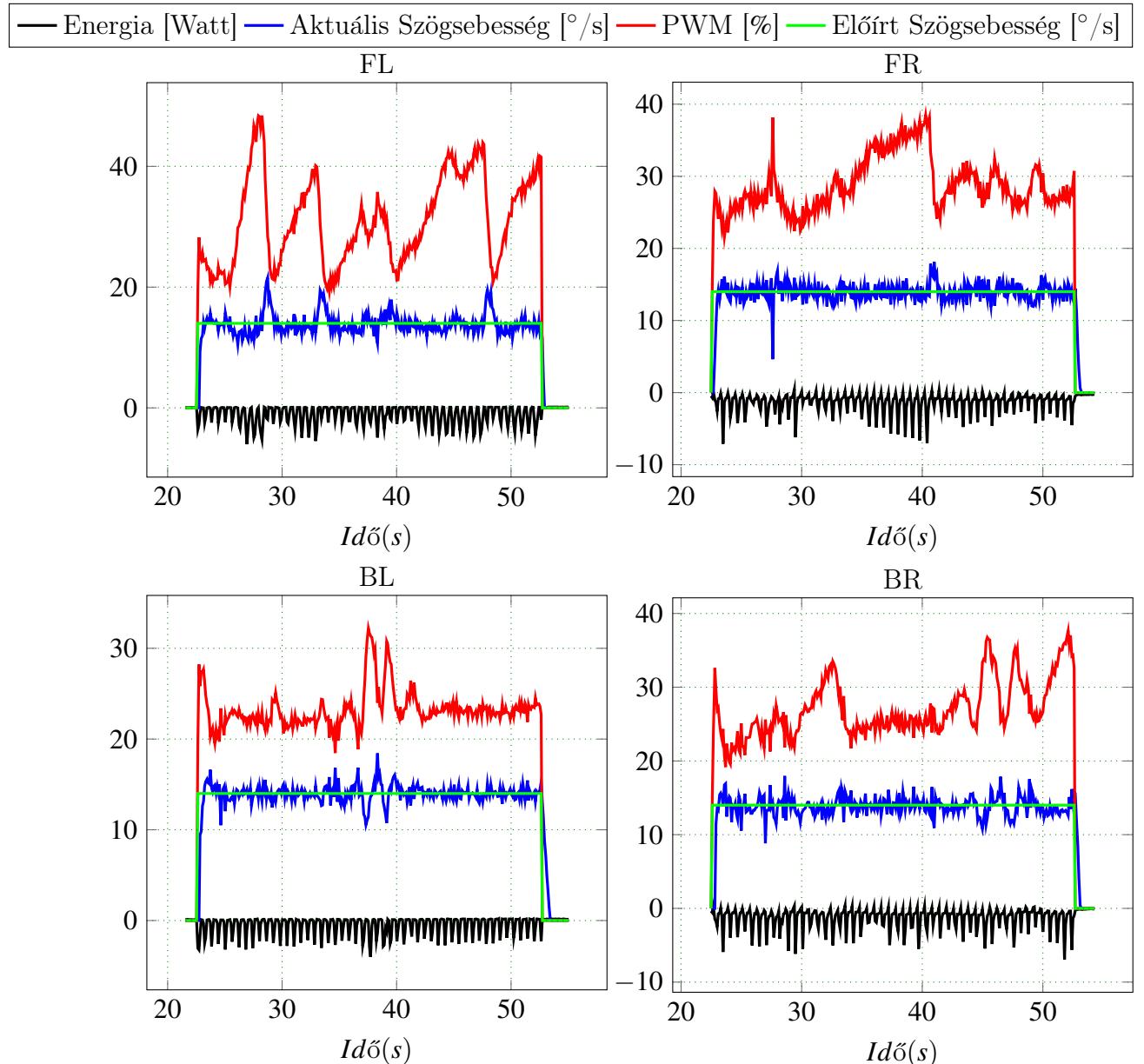


3.43. ábra. Homokos és kavicsos lejtőn felfele mozgás

### 3.6.8. Lepcson

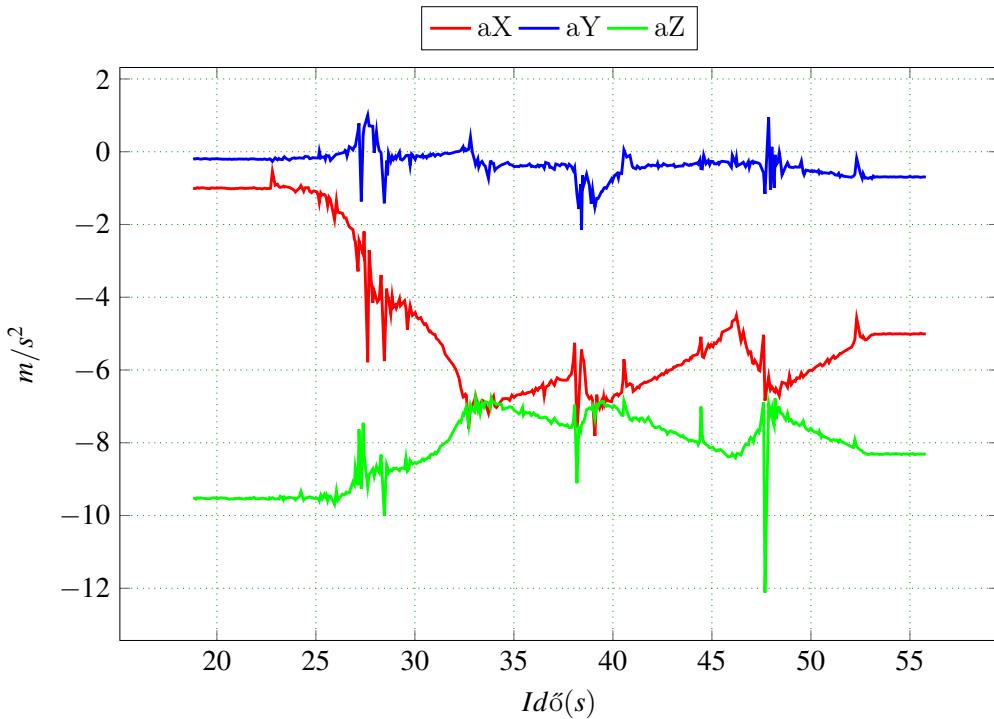
A következő mérések során a robot egy 40 °lépcsőn lefele és felfele is mozog, a lépcsőre merőleges irányban. A robot viselkedése a mozgás során, lefele könnyedén megy gond nélkül, felfele viszont a kerekek a következő lépcsőfok éléről lecsúszva visszaesnek.

A ábra 3.6.44 latható amit a lépcsőn lefele, a mozgató motrok mért értékei. és a ábra 3.6.47 viszafele mozgás során a mért értékek. A beavatkozó jel nagysága 10% -al nagyobb viszafele mozgas során. A mérések elvezetéskor a hajtást végző motrok a kisebbik átétfelkészítéssel voltak, így nagyobb forgatónyomatéket adtak le.

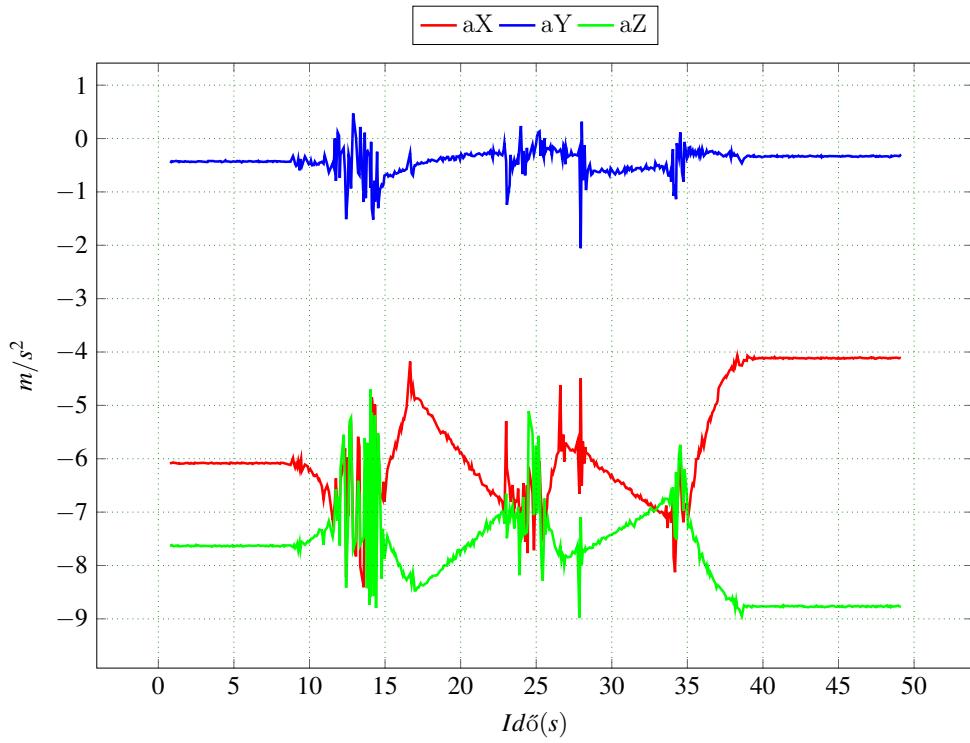


3.44. ábra. Lépcsőn lefele mozgás.

A roboton IMU szenzora által mért értékek mutatják amint a  $g = 9.81 \text{m/s}^2$  gravitációs gyorsulás megjelenik a  $aZ$  tengelyen ábra 3.6.45. Kezdetben a robot vízszinteshez közeli állapotban van X és Y. A lépcson lefele mozgás során a  $g$  fokozatosan átevődik az  $aX$  tengelyre is amiatt, hogy a robot előre dől. A robot három lépcsőfokon halad át ami látható az ábrán is.



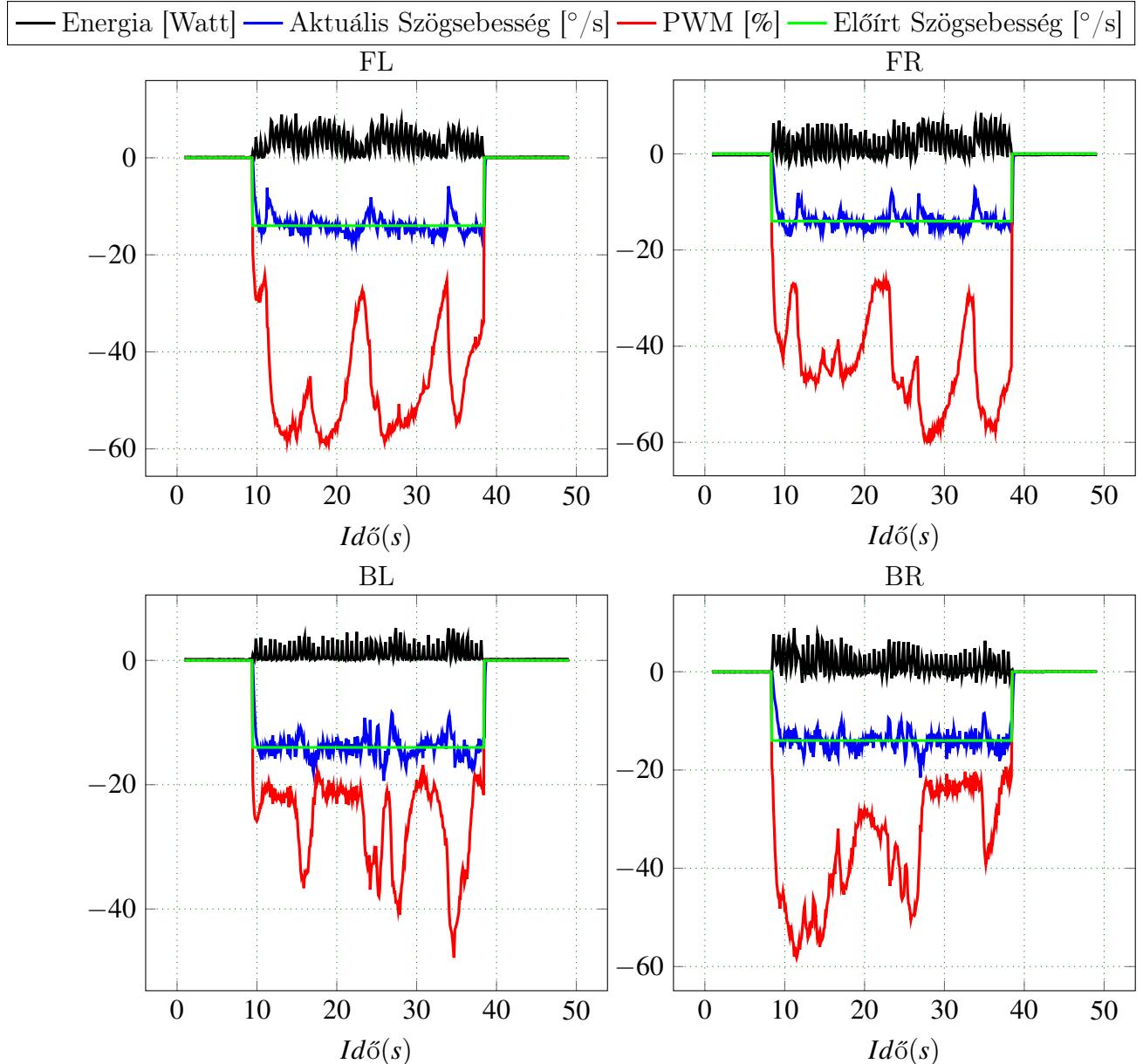
3.45. ábra. Lépcsön lefele mozgás, három lépcsőfok.



3.46. ábra. Lépcsőn felfele mozgás, kétlépcsőfok.

A lépcsőn felfele mozgás során a robot az előző állapotból indul visszafele. Azokban a pillanatokban, amikor a kerekek lecsúsznak a lépcső éléről, a kerekek szögsebessége megnő, mert a súrlódási erő lecsökken.

Az ábra 3.6.47 az *FL* és *FR* kerekeken nagyobb beavatkozó jel esik, amiatt hogy megnő a merőleges nyomóerő.

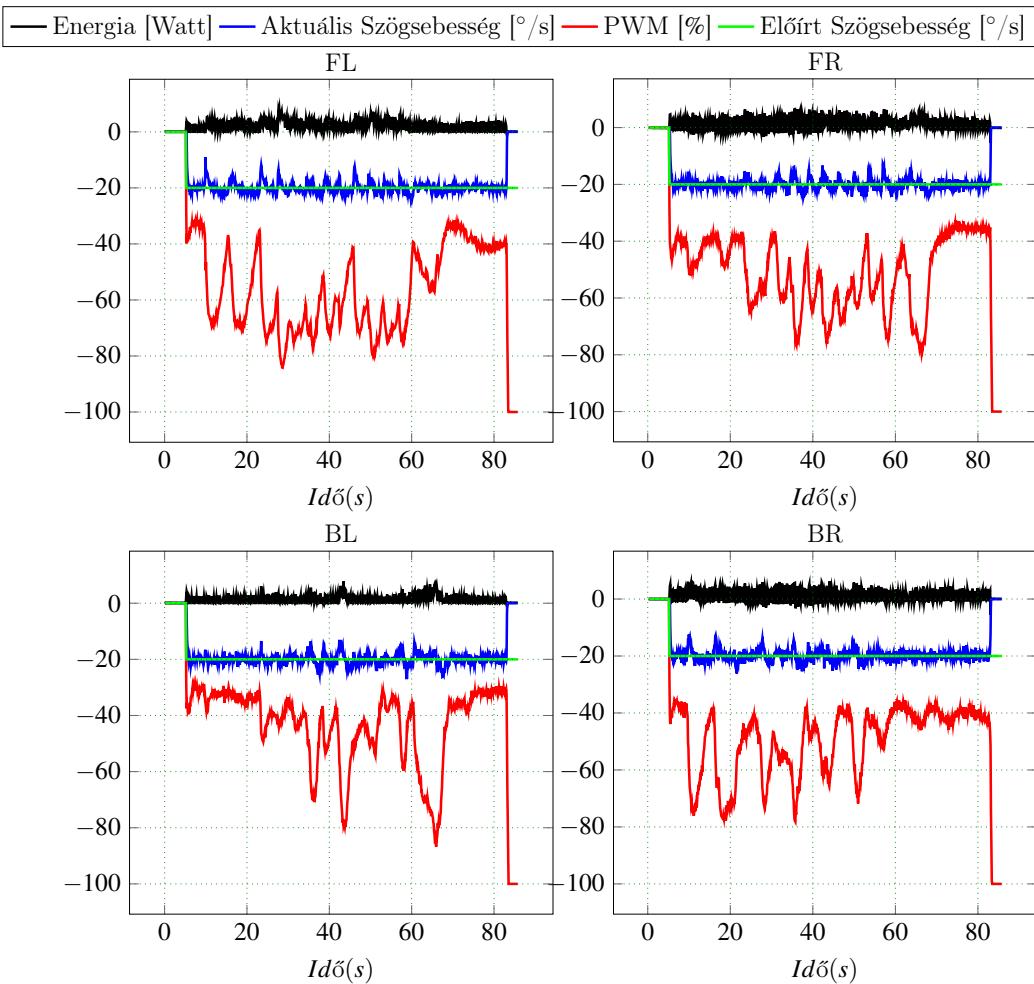


3.47. ábra. Lépcsőn felfele mozgás

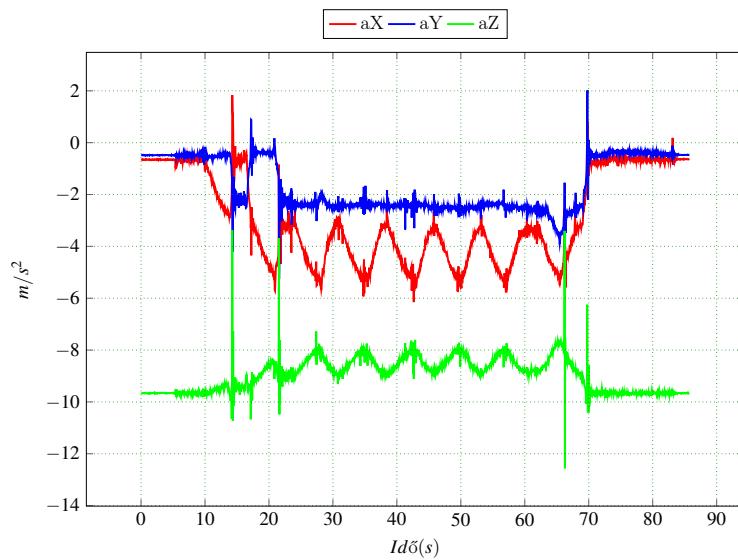
Abban az esetben ha a rontottal  $90^\circ$  kisebb szög alatt közelítjük meg a lépcsőt akkor a robot mozgása a lépcsőn felfele könnyebben tud haladni. Ha a robot súlypontja a hátul van és így közelítjük meg a lépcsőt megtörténhet az hogy a robot eleje elemelkedik és hátrabukik amiatt hogy a hátsó kerek beszorulnak a lépcsőfokba és a nyomaték így elemeli az első kerekeket.

Összevetve a ábra 3.6.46 és a ábra 3.6.49 látható hogy minden esetben az X és Y tengelyen tapasztaltunk bukdácsolást, ha  $60^\circ$ szög alatt közelítjük meg akkor az Y tengelyen is megjelenik egy dőlési szög.

A a pwm kitöltési tényezőjét tekintve az első kereke minden esetben nagyobb kitöltési tényezővel dolgoznak amiatt hogy a robot hattal megy fel a lépcsőn, a hátsó részben találhatóak az akkumulátorok.



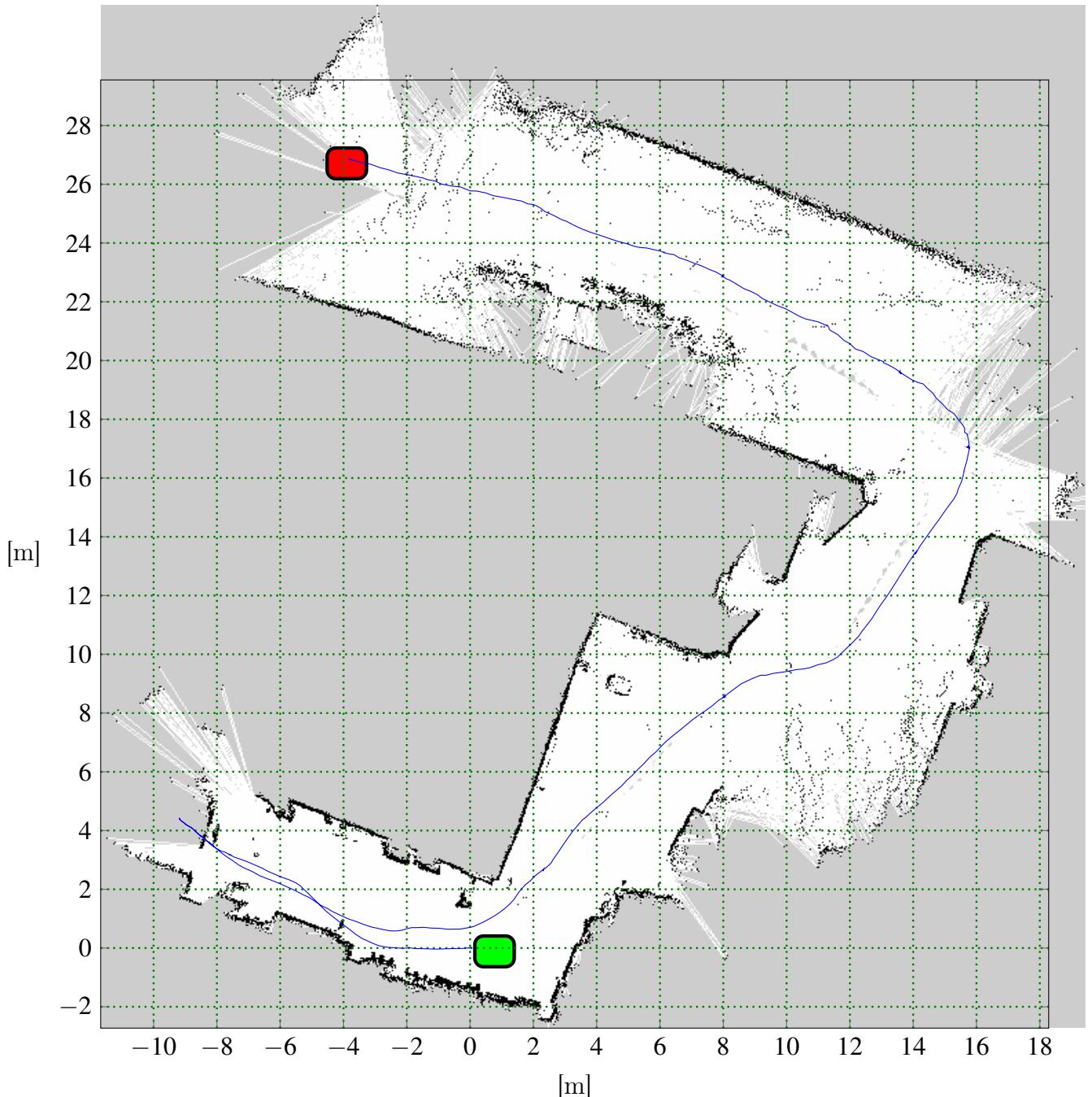
3.48. ábra. Lépcsőn  $60^{\circ}$ irányból felfele haladva 8 s.



3.49. ábra. Lépcsőn felfele mozgás  $60^{\circ}$ szöget bezárva a lépcsőfokokkal.

### 3.6.9. Ismeretlen terep térképezése és robot lokalizálása (SLAM)

A mérés során egy 25x30m udvart jártunk be a robottal távirányítóval vezérelve. A roboton található LIDAR és HectorSlam segítségével a ?? látható térképet készítettem el. A mérés során azt tapasztaltam hogy a fixen álló nagyobb tárgyakhoz jól megtudja határozni a robot pozíóját. Abban az esetben ha LIDAR üveget vagy magasra nőt füves terpen lokalizálja magát a mért értéket zajosabbak lesznek, pl: a ?? (0-8,18-22) celláiban füves terep volt.



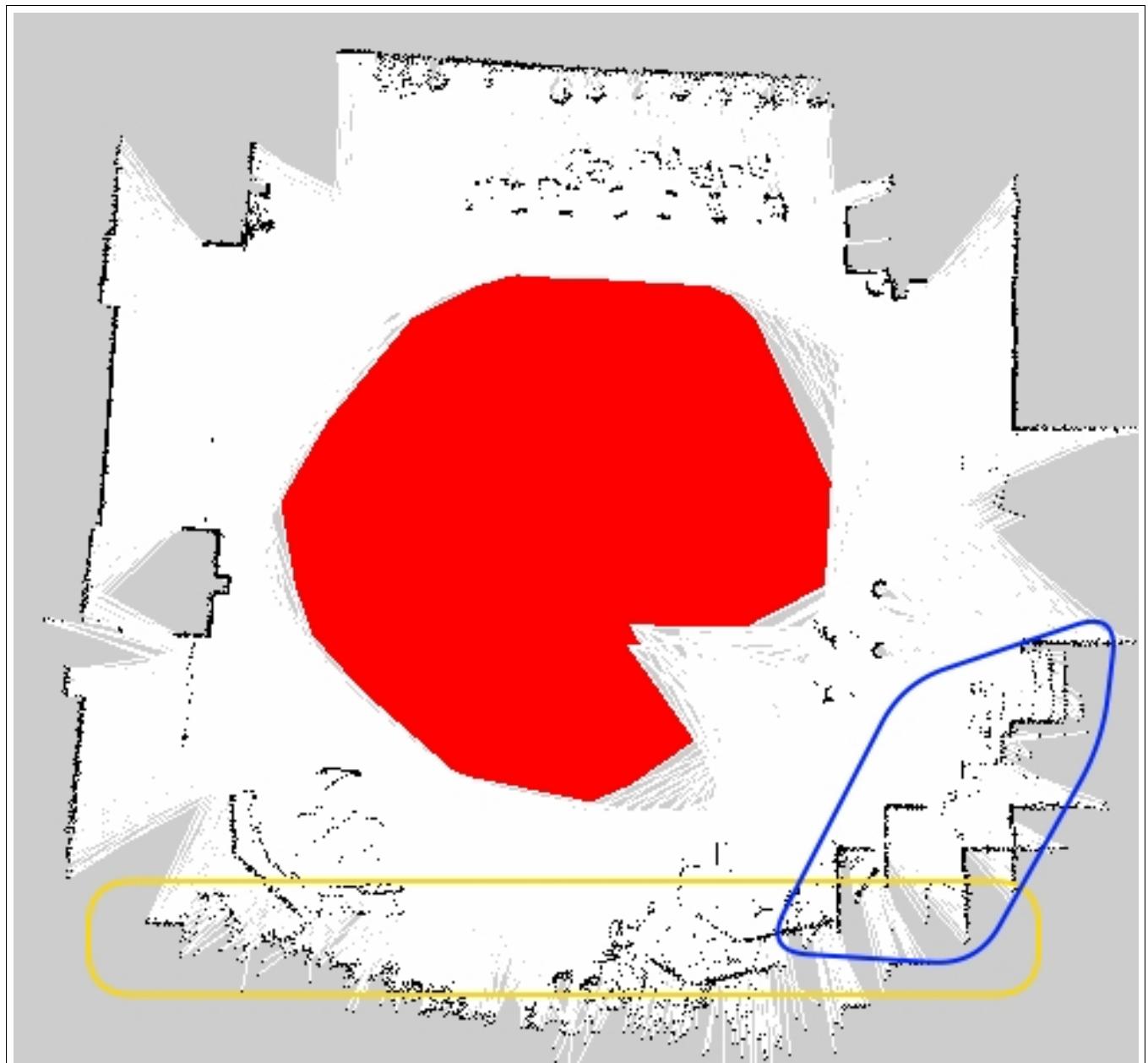
3.50. ábra. Térkép készítése miközben távirányítással halad a robot.

A LIDAR mérési tartománya 7m-re volt állítva maximálisan 8m-ig tud mérni, abban az esetben ha a robot messze esik azon fix tárgyaktól amelyekhez kezdetben lokalizálta

magét, akkor elveszíti a pozícióját és orientációját. Ezért nyilt terepen nem alkalmazható egy rövidtávú LIDAR egymagában. Két LIDAR használatával egy hőszűtavú (200m) maximális mérési távolsággal, és egy rövidtávú ami pontosabb.

A ?? látható a Marosvásárhely Sapientia Erdély Magyar Tudományegyetem Marosvásárhely karának aulájában készített térkép. Az aula méretei miatt a középső pirossal jelölt szekció a SLAM számára ismeretlen marad mivel a LIDAR mérési tartománya nem terjed ki addig. A robottal a fehér színnel jelölt zónában haladtunk végig. A sárga dobozzal jelölt részen üvegfal található amit a LIDAR nem tud pontosan megmérni ezért a térkép zajokkal lesz teli ezen a részen.

A kékkel jelölt zónában a fixen álló tárgy található csak a robot ezen a részen a sárga mezőhöz képest próbálja bepozicionálni magét és ezért a térképen bizonytalanságok jelennek meg.



3.51. ábra. Térkép készítése Sapientia Aula.

## fejezet 4

# Eredmények Kiértékelése

### 4.1. Megvalósítások

Ezzel a dolgozattal négy év folyamatos munkájának egy szakaszát szeretném lezárni, a következő eredményeket sikerült felmutatni, kronológiai sorrendben:

A mechanikai szerkezet fejlesztése, a 2011 be fejlesztett verzióhoz képest, amely sokkal robusztusabb, kültérre alkalmasabb mint az előző, és egyszerűbb is. Ugyan megmaradt az a tendencia hogy csiga-áttételeket alkalmazzak a kerekék meghajtására. Az áttételeket magam terveztem és gyártattam le, a kezdeti alacsony költségvetés miatt. Sok alkatrész kivitelezésére használtam 3D nyomtatasi technikát. Próbálkoztam inkrementális szenzor tárcsát készíteni ami bevált de nem lehetett alkalmazni robusztusan a rendszeren, mert a mechanikai rendszerben 2-3 mm kotyogás van, ami nem az jeleni, hogy nem is lehetséges csak túl sok időt igényelt volna.

Az alkatrészeket 3D tervező programban elkészítettem, és 3D nyomtatóval megvalósítotuk, a tapasztalom ezekkel az alkatrészekkel: nagy mechanikai terhelés elviselésére nem alkalmasak hosszútávon, ezért történt meg hogy a csiga tengely csapágyháza terhelés alatt széttört.

A következő lépésben a robot alacsonyrendű vezérlőáramköröket terveztem meg és viteztem ki CNC marógép segítségével.

A vezérlő logika implementálására FPGA alapú fejlesztőlapokat használtam mert, flexibilisek és testreszabható rendszert lehet elkészíteni segítségükkel.

Vivado környezetet használtam az FPGA fejlesztésére, megvalósítottam egy uBlaze processzorrendszer kialakítását és több hardveres modult is amelyek a következők: PWM modul, UART protokoll csomag értelmező amely támogatja a nagy sebességű kommunikációt és sikerült elérni az 1ms mintavételezési periódust, globális engedélyező jel, ezeket a modulokat System Generator-ban valósítottam meg és IP mag készült ezekből.

Robotokhoz kapcsolódó keretrendszer használata lett szükséges így került sor a ROS keretrendszer ismereteinek az elsajátítására.

Megterveztem az integrációt a ROS és FPGA UART alapú kommunikációjának kiépítésére a jelen pillanatban a robot specifikus, dinamikusan nem hasznosulható újra más FPGA alapú robotoknál anélkül hogy ne kellene számottevő programkódot írni, de lehetséges automatizálni az integrációt amennyiben ez szükséges.

A ROS használata számos előnyel járt, sikerült bekonfigurálni és elindítani a ROS keretrendszerben levő egyéb eszközöket is pl: logolás a nodok között, eclipse fejlesztőkörnyezet bekonfigurálása a fejlesztéshez, HectorMap térképez és lokalizálás LIDAR alapján, robotmodell elkészítése Rviz 3D megjelenítő számára. A alacsonyrendű paraméterek szinkronizációja a ROS rendszerben levő paraméterekkel.

Sikerült elkészíteni egy 90GB méretű virtuális gépet amelyen minden eszköz meg-található a robot fejlesztéséhez: Vivado,Matlab,Arduino,ROS lunar,eclipse.

Sikerült kiépíteni egy deploy mechanizmust amely segítségével a forráskódot tudjuk telepíteni a robotra. A SSH használatával a meglevő kódokat átmásoljuk a robotra ahol majd újra fodorítjuk a szükséges részeket.

Sikeresen elsajtottam a ROS alapjait, és megterveztem egy sajátos kommunikációt FPGA alapú rendszer és a ROS között. Az integráció a robot és a ROS között jól működik, minden egyes szenzor mért adata bekerül a ROS környezetbe.

A rendszer elindítása után, a megbecsültem MATLAB segítségével a kereke átviteli függvényeit, és majd ezekre PID szabályozót terveztem PID tuning toolbox segítségével.

A hectormap segítségével, az ismeretlen terep térképezésével, a roboton lokalizálva, sikeresen bekonfiguráltam a movebase nevű eszközt, amely segítségével a robotot egy adott pozícióba és irányba tudjuk elvinni. A move base megoldja az akadályok kikerülését is.

A térképezéssel kapcsolatban a tapasztalatok a mérések során, a LIDAR és Hector-Map zajosak külső terepen egyrészt a környezeti tényezők miatt amelyek befolyásolják a robot dőlésszöget, így zajosítva a méréseket, valamit a lokalizáció nem pontos ha üvegen keresztülhalad a lézersugár.

A robottal végzet mérések során a robot szerkezete és alacsonyrendű szabályzása megbízhatónak bizonyult, képes több legalább 100kg függőleges irányú terhelést elviselni és akar egy személygépkocsit is elmozdítani.

A mérések alapján a lépcsőre könnyebben tud felmenni ha egy 90 °nál kisebb szög alatt közelítjük meg.

A környezetre fordulás esetén gyakorolja a robot a legnagyobb behatást, pl helyben forgás esetén a súrlódások miatt a füves talajt a fekete földig leszedi. Mezőgazdasági alkalmazásra előnyösebb volna ha mind a négy kereke kormányozhatnák.

A lépcsőn és a lejtőt is előnyösebb a SSMR al úgy megközelíteni hogy a súlypont a robot elején legyen ahogy felfele haladunk így a merőleges nyomóerők is egyenletesebben eloszlanak. Lépcső esetében ügyelni kell a hátsó kerekek lépcsőfokba való beragadása miatt,mert a kerekek nagy forgatónyomatéka, a robot hátra billentheti, ez abban az esetben állhat fent ha a lépcsőt 90°szög alatt közelítjük meg.

## 4.2. Hasonló rendszerekkel való összehasonlítás

Az általam kivitelezett robot legjobban a Husky robotra hasonlít amely szintén négy kerekkel rendelkezik. A legnagyobb előnye a sebessége mivel nem használ de kisebb a kerekei forgatónyomatéka. Hátránya ha hátrabukik akkor a kerekek nem fognak érintkeznek a talajjal, míg a SapRover minden két irányban képes működni.

Tulajdonság	Husky Robot	Előny	SapRover
széleség	0.67m	?	0.78m
hosszúság	0.99m	?	0.80m
magaság	0.39m	?	0.40m
ROS kompatibilitás	Igen	=	Igen
Max sebesség	1m/s	>	0.25 m/s
Bépitett számítógép	IGEN	=	IGEN
Önsúly + hasznos teher	50+75 kg	<	60+100 kg
Hátraborulhat	NEM	<	IGEN

### 4.3. További fejlesztési irányok

A robotplatformot a következő lépésekben felkelne szerelni egy nagyobb méréstartományú 3D LIDAR, sztereó kamera, és nagyobb pontóságú GPS vevővel. A platform anonimitása lenne a fő cél a növénytermesztésben, legyen képes eljutni A pontból B pontba autonóm módon anélkül hogy kártenne a haszonnövényekben. Eközben legyen képes kiszolgálni a majdan rászerelhető pl: robotkar kéréseit.

A mechanika tovább fejlesztése: célszerű lenne a robot minden a négy kerekét kormányozóval tenni egyedileg, így a csuszás kormányzással járó károk megszűnnek és energia fogyasztása is hatékonyabb lenne.

Az FPGA és a ROS integrációját flexibilisébe lehetne tenni azáltal hogy egy XML alapú konfigurációs fájlban leírva a kívánt üzenet típusokat egy értelmező segítségével kigenerálni a szükséges c++(.cpp és .h) állományokat, ROS és uBlaze oldalra. A UART kommunikációért felelő IP magot csak konfigurálni kellene szoftverből.

Napemseles energiaforrással ellátni, és töltőállomást elkészíteni amelyre automatikusan kapcsolódhatna a robot. Az akkumulátorok kezelésére egy energia processzor lenne szükséges.

## Irodalomjegyzék

- [1] David Couceiro Micael Rocha Rui Araújo, André Portugal. Integrating arduino-based educational mobile robots in ros, 2013.
- [2] S. Arslan and H. Temeltaş. Robust motion control of a four wheel drive skid-steered mobile robot. In 2011 7th International Conference on Electrical and Electronics Engineering (ELECO), pages II–415–II–419, Dec 2011.
- [3] Marissa G. Campa, J.L. Gordillo, and Rogelio Soto. Speed and point-to-point control for trajectory tracking of a skid-steered mobile robot. In 11th IEEE International Conference on Control & Automation (ICCA). IEEE, jun 2014.
- [4] Sachin Chitta, Eitan Marder-Eppstein, Wim Meeussen, Vijay Pradeep, Adolfo Rodríguez Tsouroukdissian, Jonathan Bohren, David Coleman, Bence Magyar, Genaro Raiola, Mathias Lüdtke, and Enrique Fernández Perdomo. ros\_control: A

generic and simple control framework for ros. The Journal of Open Source Software, 2017.

- [5] Mateusz Cholewiński and Alicja Mazur. Influence of choosing the extending column in trajectory tracking control of ssmp platform using artificial force method. Advances in Intelligent Systems and Computing, 323:129–140, 01 2015.
- [6] Carol Fairchild and Dr. Thomas L. Harman. ROS Robotics By Example - Second Edition: Learning to control wheeled, limbed, and flying robots using ROS Kinetic Kame. Packt Publishing, 2017.
- [7] Michael Ferguson. rosserial @ONLINE, November 2018.
- [8] Scripting News Inc. Xml-rpc, 2018.
- [9] Jackie Kay Ioan Sucan. Urdf, 2018.
- [10] Jackie Kay Ioan Sucan. urdf/xml/joint, 2018.
- [11] Jackie Kay Ioan Sucan. urdf/xml/link, 2018.
- [12] Lentin Joseph. Mastering ROS for Robotics Programming. Packt Publishing, 12 2015.
- [13] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR). IEEE, November 2011.
- [14] Anis Koubaa, editor. Robot Operating System (ROS). Springer International Publishing, 2016.
- [15] Jet Propulsion Laboratory. Spirit struggles to survive the martian winter, 2006.
- [16] Matlab. link, 2018.
- [17] Wim Meeussen. Create your own hardware interface @ONLINE, November 2018.
- [18] Kevin Watts Blaise Gassend Morgan Quigley, Brian Gerkey. joy, 2018.
- [19] Eduardo Munera, Jose-Luis Poza-Lujan, Juan-Luis Posadas-Yague, Jose Simo, and J. Francisco Blanes Noguera. Distributed real-time control architecture for ros-based modular robots. IFAC-PapersOnLine, 50(1):11233 – 11238, 2017. 20th IFAC World Congress.
- [20] ROS.org. Parameter server, 2018.
- [21] Maciej Trojnacki. Dynamics Model of a Four-Wheeled Mobile Robot for Control Applications – A Three-Case Study, volume 323. 01 2014.
- [22] Wim Meeussen Tully Foote, Eitan Marder-Eppstein. tf, 2018.
- [23] wiki.ros.org. Writing a simple publisher and subscriber (c++), 2018.
- [24] X. Wu, M. Xu, and L. Wang. Differential speed steering control for four-wheel independent driving electric vehicle. In 2013 IEEE International Symposium on Industrial Electronics, pages 1–6, May 2013.

[25] Xilinx. Block memory generator v8.2, 2015.

[26] Xilinx. Axi uart lite v2.0, 2017.