Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Bőr István

ÚJ MEGOLDÁSOK KERÉK ÉS IMU ALAPÚ ODOMETRIA FUZIONÁLÁSÁRA

KONZULENS

Dr. Fehér Gábor

BUDAPEST, 2020

Tartalomjegyzék

Hallgatói nyilatkozat

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

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzé tegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Jelen dolgozatot más tudományos diákköri szekcióba vagy konferenciára nem neveztem, nem adtam le mesterszakos szakdolgozatként, nem publikáltam.

Kelt: Budapest, 2020. 10. 28.

...…….
Bőr István

Összefoglaló

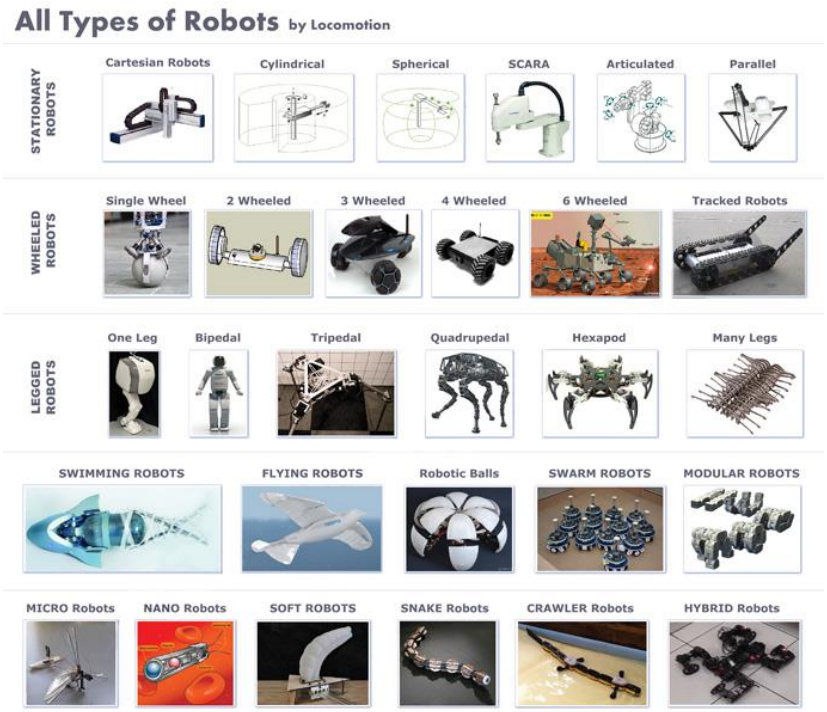
Az elmúlt években egyre nagyobb teret hódítottak az önjáró mobil robotok, amiket röviden AMR-nek (Automated Mobile Robot) is neveznek. Hasznosságuk nem merül ki abban, hogy vezeték nélkül képesek működni, mivel képesek önállóan döntést hozni és végre is hajtani azt. Azonban ahhoz, hogy egy robot ilyen módon működhessen, létfontosságú az, hogy minden pillanatban tudja a saját pozícióját és orientációját, azaz röviden az odometriáját. A kutatásom témája a robot rendszerekre, azon belül is a különböző odometria számítási eljárások összehasonlítására és későbbi fuzionálására irányult. Két fő eljárás köré csoportosultak a vizsgálataim, az egyik a kerékelfordulás által számított odometria, a másik a különböző szenzorok, az úgynevezett IMU-k (Inertial Measurment Unit) használatával történő odometria vizsgálata. Mindkét módszer rendelkezik a rá jellemző negatívumokkal és kizáró körülményekkel, így a cél ezen negatívumok kiküszöbölése és az így elért javulás számszerűsítése volt. Az odometria definíció szerint a mozgásváltozást mérő szenzorok mérési eredményeinek felhasználása, amely által megbecsülhető a pozíció időbeli változása. A kutatásom eredményeképp létrejött egy olyan rendszer, amely egyesíti a két rendszer előnyeit, ily módon előállítva egy olyan odometriát, amely pontosabb az előzőknél, továbbá jól láthatóan mutatja a több szenzor használatának az előnyeit egy robotikus rendszerben. A kutatásomban vizsgáltam egy, kettő, illetve több szenzor használatának a hatását az odometriára, és ezeknek eredményeit számszerű adatokkal támasztottam alá. A kutatásom további haszna az, hogy ezen eredmények átültethetök olyan rendszerekbe is, ahol a robot nem tartalmaz kerekeket (példának okáért a drónok), és csak a szenzorok fuzionálásával lehet finomítani az odometriát.

Abstract

In the last few years, the popularity of the Automated Mobile Robots (AMR) has risen greatly due to their compactness and the fact, that they do not require wired connections. Their usefulness doesn't end at this point, because their great advantage comes from their ability to make decisions on their own, and to be able to carry them out too. However to be able to operate this way, a robot should always be able to determine it's own location and position, or in other words, it's odometry. The definition of odometry is the use of data from motion sensors to estimate the change in position over time. My research is focused on different robot systems and on the different methods to measure, compare, and fusion the odometry. I examined two methods to calculate odometry, those are the odometry based on the wheels and the odometry based on the IMU (Inertial Measurement Unit) data. Both of the methods has it's own disadvantages and circumstances in which they cannot function properly, so my goal was to eliminate these negative sides and combine the two systems in a way, that they only keep their advantageous properties, so the odometry which is produced this way, will be more accurate than the previous ones, furthermore, it represents the positive effects of using more sensors in a robotic system. In my paper, I examined the effect of one, two, and more sensors on the odometry, and I provided a quantitative representation of it too. Further use of my paper that, can be implemented in systems where the robot does not contain wheels (i.e. drones) and only can rely on the data acquired from the IMU.

1 Bevezetés

A mai világban egyre elterjedtebbek a robotok, mint a hétköznapi élet kiegészítői, mivel pontosabban és gyorsabban hajtják végre a mindennapi feladatokat. Ahogy haladt előre a tudomány ezen ága, úgy jelent meg a felhőrobotika és a kooperatív robotika fogalma is. A felhőrobotika jelentése röviden azt takarja, hogy a robot, mint olyan nem tartalmazza „a fedélzetén" a komoly számításokért felelő hardvert, hanem csak egy olyan vezetéknélküli hálózati kapcsolatra képes eszközt, ami továbbítja a mérések eredményét, és a számításokhoz feltétlenül szükséges adatokat a felhőbe, ahol a tényleges számítás megtörténik. Ezzel a lépéssel közelebb kerültünk a kooperatív robotika fogalmához is, ami magyarul azt jelenti, hogy különböző robotok egymással együttműködve hajtják végre a feladatot. A felhő ezt a műveletet nagyban megkönnyíti, mert egy helyen tárolódnak az adatok, ami lehetővé teszi, hogy minden eshetőségre felkészítse a robotokat. Azonban ez az 5G megjelenéséig csak elméleti síkon valósulhatott meg, mivel az addig jelenlévő hálózatok túl nagy késleltetéssel rendelkeztek, és túl kevés eszközt voltak képesek stabilan ellátni, ahhoz, hogy egy robot flotta gyorsan és pontosan működhessen. Alább a legelterjedtebb robot típusokat láthatjuk az 1.ábrán.



1. ábra Robot típusok

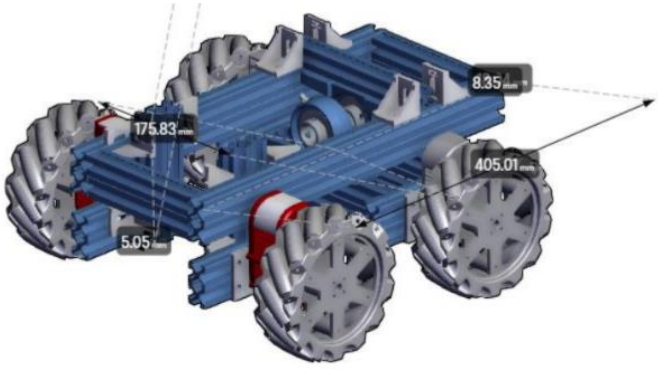
A kutatásomban az általam épített négykerekű, mecanum kerékkel felszerelt robot mozgását és kinematikáját vizsgálom, ami leginkább az 1.ábra második sorának a negyedik oszlopában található robotra hasonlít. Az ilyen típusú robotok leginkább kutatási célokra és logisztikára vannak használva. Példának okáért ilyen típusúak a holdjárók, és a gyárakban az alapanyag szállító robotok is. Komoly fejlesztések folynak például annak irányába, hogy az igen nagy tisztaságú szilíciumlapkákat képesek legyenek elszállítani pusztán robotok emberi beavatkozás nélkül a gyár egyik részéből a másikba fennakadás nélkül úgy, hogy a többi robottal kommunikálnak, közösen térképet építve és az útjukba kerülő akadályokat elkerülve. Az én dolgozatom is ilyen irányú, azonban míg a hagyományos kerekekkel felszerelt robotoknak 6 szabadsági fokuk (azon irányok száma, amelyek irányába a robot állóhelyből el tud mozdulni) van, addig az általam tervezett robotnak, a mecanum kerekes meghajtás révén 8 szabadsági foka van. Ezért is hívják őket angol terminológiában omnidirectional, azaz magyarul bármely irányba elmozdulni képes robotoknak. Ez a fajta mozgás külön figyelmet és komplexebb matematikai műveleteket eredményez, és ez adja a létjogosultságát a kutatásomnak. Továbbá a robotikán belül is előfordulnak olyan területek, melynek vizsgálata fontos és esszenciális. Ilyen például az odometria, amely annak a tudománya, hogy közelítjük egy *Automated Mobile Robot* (továbbiakban **AMR**) mozgását csupán a kerékelfordulás, vagy a beépített szenzorok által közölt mérési adatok alapján. Ez azért fontos, mert az egész mozgás erre épül, ez adja meg a robotnak a viszonyítási alapot a környezethez, és így tudja magát elhelyezni a világban, továbbá így lehet a robotnak komplex utasításokat kiadni, például a térképalkotás és a környezet felfedezése. Továbbá így tudja megbízhatóan kikerülni az útjába kerülő akadályokat, ami, ha nem pontos az odometria jóformán lehetetlen. Több robot alkalmazása esetén meg különösen fontos, hogy minél kevesebb tévedés legyen a pozíció meghatározásában, mivel a tévedések összeadódnak és hatványozott hibát okoznak és összeütközhetnek ezáltal. A minél megbízhatóbb odometria előállítása miatt komplex matematikai segédeszközöket és szenzorokat alkalmaztam, amit, később bővebben is kifejtek.

2 A kutatáshoz szükséges környezet előállítása

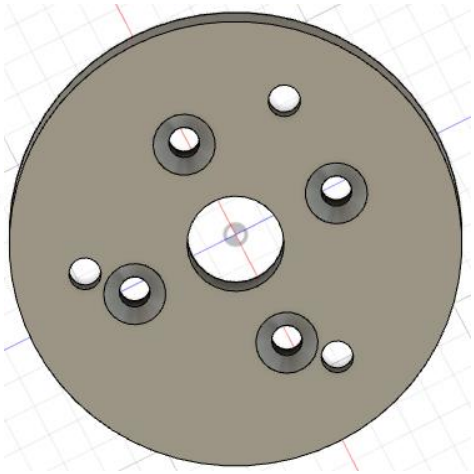
Az AMR-t előre meghatározott követelmények szerint kellett megépítenem, ezek a követelmények magukba foglalták azt, hogy a hajtáshoz kettő darab a HEBI fejlesztőcég által fejlesztett X5-1 típusú és kettő darab HEBI X5-9 típusú motorokat használjak fel, amik úgynevezett mecanum kerekeket hajtanak meg. A fő feladat egy olyan interfész megalkotása volt, ami megbízhatóan implementálja egy ilyen kerekekkel rendelkező AMR mozgását, és kapcsolatot teremt a felhasználói réteg és a HEBI motorok vezérlése között. A szenzorok és a kiegészítő funkciók is meghatározásra kerültek a továbbiakban. Fontos kitétel, hogy a rendszer képes legyen vezeték nélkül is megbízhatóan működni, mivel ez egy úgynevezett „research AMR", így fontos, hogy ne legyen helyhez és számítógéphez kötött, hálózat és táplálás szempontjából. Ezt eleinte egy WiFi képes Raspberry Pi 4 fejlesztőpanel teszi lehetővé, azonban a későbbiekben, ha a technológia már rendelkezésre áll, 5G-n fog működni. Az energiaellátást egy 6 cellás LiPo (lítium-polimer) akkumulátor biztosítja, ami képes ellátni a HEBI motorok és a *Raspberry* (továbbiakban **RPi**) feszültségigényét. További követelmény az, hogy az AMR képes legyen térképet alkotni dinamikusan a környezetéről. Ezt a térképalkotást egy LIDAR A1 típusú szenzor biztosítja, ami lézernyalábok segítségével pontfelhőt alkot a környezetében található tárgyak formájáról és távolságáról. További funkció, hogy *Robot Operating System*-en (továbbiakban **ROS**) keresztül hozzá lehet férni az akkumulátor aktuális feszültségéhez. Ehhez építenem kellett egy kiegészítő áramkört, amit a későbbiekben részletezek. Az energiaellátás során megoldandó problémaként merült fel a különböző hardverek különböző feszültségszint igénye, mivel az akkumulátor 25.2 voltot szolgáltat, míg az RPi maximális bemenő feszültsége ~5 voltnál van meghatározva, ennél magasabb feszültségérték maradandó károsodást okozhat a hardverben. Ezért erre a célra egy előre legyártott áramkört, egy úgy nevezett step-down modult alkalmaztam, amely képes a bemeneti 25.2 voltot 5 volttá alakítani, Hovatovább ez nem a legjobb módszer, mert így rengeteg energia disszipálódik el a levegőbe, azaz alakul át hőenergiává, így fűtve az áramkört, amely így az üzemi hőmérsékletet meghaladva felmelegszik. Az ilyen körülmények között hosszantartó használat miatt az élettartama jelentősen rövidül az eszköznek, ezért egy hűtőbordával is felszereltem az áramkört. Miután ily módon előállítottam a szükséges tápellátást az RPi-nek és a motoroknak, áttértem a hardver összeépítésére.

2.1 A hardver összeállítása

Az egyik legfontosabb feladat a sok közül, a hardver összeállítása volt, amelyet úgy kellett véghez vinnem, hogy minél pontosabban illeszkedjenek az egyes elemek egymáshoz, ezzel elkerülve azt, hogy a mozgás helytelen lesz amiatt, hogy a kerekek nem érintkeznek a talajjal teljesen vagy pedig nincsenek egy vonalban. A munka megkönnyítésére létrehoztam egy méretarányos modellt, amit az 2.ábrán lehet látni. Ezen a modellen az előzetes teszteket elvégezhettem, mielőtt összeépítettem volna a fizikai mását. Az első nehézség itt mutatkozott meg, mivel a HEBI és a kerék kialakítása olyan, hogy nem lehet összecsavarozni őket összekötő elem nélkül. Itt a projekt egyedisége miatt nem állt rendelkezésre ilyen alkatrész, ezért terveztem egy, a 3.ábrán is látható kerékadaptert a Fusion360 nevű CAD szoftverben. Ezt a modellt 3D nyomtatóval kinyomtattam, majd az AMR-t összeállítottam és megkezdtem az előzetes teszteléseket.



2. ábra Az AMR modellje



3. ábra A kerékadapter modellje

2.2 A meghajtás

A mozgatást illetően több választás elé voltam állítva. Dönthettem a hagyományos szervo motoros meghajtás vagy a jóval komplexebb HEBI típusú motorok alkalmazása mellett. Mivel a HEBI verziója sokkal testre szabhatóbb és „mérnökibb”, ezért arra esett a választás. Azon belül is, két X5-1 és két X5-9 típusú motorok kaptak helyet az AMR-n. Ahogy a 4.ábrán is látható, hogy a HEBI motorok kialakítása olyan, hogy van kitüntetett eleje, amit a motor egyfajta viszonyítási pontnak is használ, vagyis

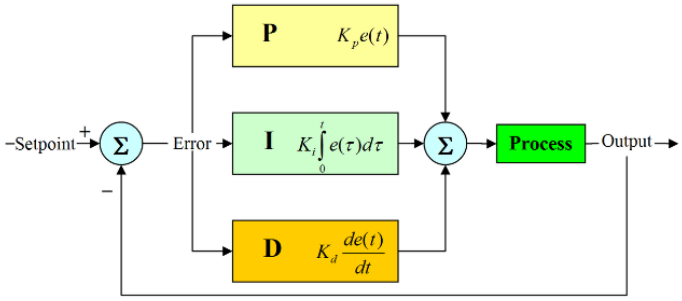
a szoftvernél és a szimmetria kialakításánál is figyelembe kellett vennem a referencia pontot. Ennél fogva a motorok elhelyezkedése volt a következő döntési pont, mivel ez egy olyan AMR, amely minden irányban tud mozogni, nincs kitüntetett „eleje” és „hátulja”, így szimmetrikus elrendezésre kellett törekednem. Mivel mind a négy kerék rendelkezik meghajtással, arra törekedtem, hogy tengelyenként egyező motorok kerüljenek a kerekekre, így minimalizálva az ebből eredő hibákat. A hibákat, melyek onnan származtak, hogy a motorok különböző fizikai paraméterekkel bírnak, ennek okán nem tudják ugyanolyan pontosan végrehajtani a kapott parancsokat, vagyis az AMR siklani, vagy konyhanyelven fogalmazva „kacsázni” fog, hiába ugyan azt a parancsot kellene végrehajtaniuk. Ezt a hibát PID szabályzókkal lehet tovább csökkenteni, melyekről a szabályozás alfejezetben ejtek szót.

4.ábra A HEBI motor

2.3 A szabályozás

Mint korábban említettem egy AMR vezérlésében nagyon fontos szerepet kap a szabályozás. Szabályozás alatt a motorok olyan vezérlését értjük, amely magában foglalja a folyamatos visszacsatolást és beavatkozást, hogy a kimenő jel a bemenő jel minél pontosabb mása legyen. A motorok helyes működéséhez tehát szokás szabályzót tervezni, amely folyamatosan méri a bemenetet, illetve a kimenetet, és ha egy megadott tűrésből kilép a motor, akkor beavatkozik oly módon, hogy a kimenet minél jobban hasonlítson a bemenetre. Itt volt lehetőségem saját szabályozót írni, de használhattam a tanulmányaim során megismert PID szabályzó rendszert, ami egyrészt sokkal megbízhatóbbnak bizonyult, mint bármely általam tervezett szabályozás, másrészt a HEBI rendszerei tartalmaznak PID szabályzókat a motor minden tényezőjéhez (nyomaték, sebesség, pozíció). A PID szabályzóról röviden írva, fontos elmondani, hogy amint a 5.ábrán is

látható, 3 tagból áll a beavatkozójel. A hibajelből, a hibajel integráltjából és a hibajel deriváltjából. A különböző tényezők különböző hatásokat (túllövés, felfutás etc.) befolyásolnak, és a motorok PID szabályzóit aprólékosan beállítva elértem a kívánt hatást.

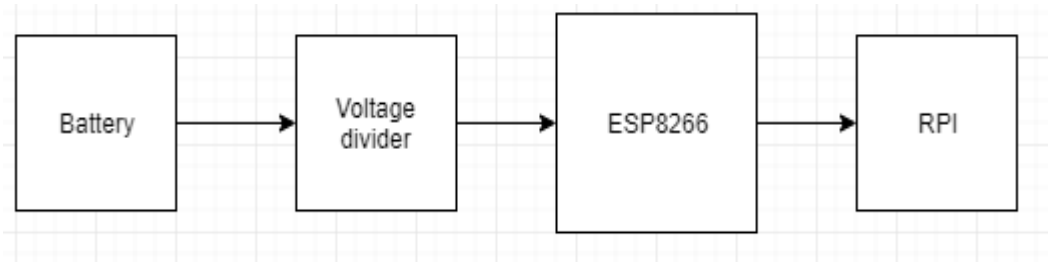


5.ábra A PID szabályzó blokkvázlata (forrás: Wikipédia)

2.4 Tápellátás és a batteryCheck áramkör

Mint említettem, a tápellátást egy 6 cellás akkumulátor biztosítja, melynek maximális kapocsfeszültsége 25.2 Volt. Az összes igényt ez az akkumulátor elégíti ki, így egy stabilizátor áramkörre is szükségem volt, ami az RPI által igényelt 5V-ot állítja elő a bemeneti 25.2-ből. Ezt egy step-down konverterrel oldottam meg. Az általam használt akkumulátor (LiPo) érzékeny a túlmerítésre és tüzet, illetve az általa ellátott eszközök meghibásodását okozhatja, ha egy cella 3V alá merül. Ennek az elkerülésére elkészítettem egy egyszerű áramkört, ami képes arra, ha adott feszültség alá merül az akkumulátor, akkor világítani kezd egy LED. Az alapelve egy ilyen áramkörnek, hogy analóg-digitális átalakítást (ADC) kell végeznünk, mivel a feszültség egy analóg érték, amit nem tud így egyből értelmezni a processzor. További probléma, hogy az RPi nem rendelkezik ilyen típusú átalakítóval, ezért az egyszerűség végett egy ESP8266 típusú mikrovezérlőt alkalmaztam erre a célra, amelynek kialakítása a 7.ábrán található. Az ESP rendelkezik egy 10 bites ADC-vel, de ez a bemenet csak 3.1 Voltot tud fogadni, ezért a feszültségosztás elvét alkalmazva előállítottam ezt a 25.2 Voltból és ezután már csak ezt az arányt alkalmaztam. A feszültségosztás egy olyan alapvető elektronikai fogalom, ami azt takarja, hogy egy magasabb feszültségből előállítunk egy számunkra szükséges alacsonyabb feszültséget két ellenállás használatával. Én a 25.2 voltból állítottam 3.1 voltot, ami az ESP maximális bemeneti feszültsége az ADC portján, és ezt egy 72 kOhmos és egy 10 kOhmos ellenállás soros kapcsolásával tettem meg. Az 6.ábrán látható a batteryCheck rendszer blokkvázlata, ami elvégzi a feszültségosztást. Az ESP, mivel 10

biten ábrázolja a bejövő feszültséget, 1024 részre osztja fel azt. Így pár alapvető matematikai művelettel kinyerhető az aktuális telepfeszültség, amit ezután soros porton közöltem az RPi-vel. Továbbá az ESP beépített LED-je világít, ha 23V alá merül az akkumulátor.



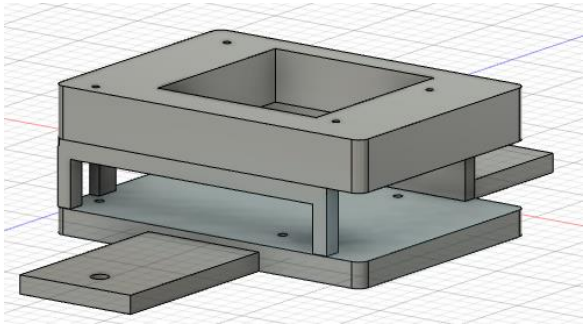
6. ábra A blokkdiagram

7.ábra Az ESP

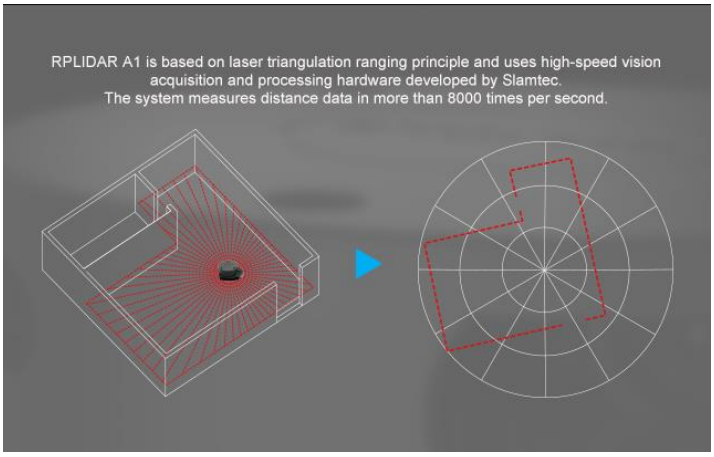
2.5 A Raspberry Pi 4 és a LIDAR

Az egész rendszer vezetéknélküli működéséért az RPi felel. Ez egy olyan fejlesztőpanel, melynek teljesítménye közelíti egy gyengébb asztali számítógép teljesítményét, de ahogy a 10.ábrán is látható, úgy lehet használni a lábait, mint egy hagyományos fejlesztőpanel lábait. Az RPi felcsatlakozik a Wifi-re és utána oszt IP-címet a HEBI motoroknak DHCP segítségével. Az RPi-n egy headless Ubuntu szerver van, ami nem rendelkezik GUI-val, így SSH segítségével lehet kapcsolódni hozzá. Ahhoz, hogy feltehessem a vázra, terveztem az RPi-nek egy, a 8.ábrán is látható foglalatot, ami szabadon hagyja a szükséges portokat, megfelelő védettséget biztosít neki, valamint lehetővé teszi, hogy felrakhassam az AMR-re. Ezt is kinyomtattam a 3D nyomtatóval és ezt követően teszteltem. A rendszer fő szenzorát egy LIDAR alkotja, amely az RPi-hez közvetlenül kapcsolódik USB kapcsolattal. A LIDAR egy olyan eszköz, ami a 9.ábrán látható módon lézeres alapú távolságmérést végez úgy, hogy másodpercenként sok, egymástól független lézernyalábot bocsát ki és méri a lézer kibocsátása és visszaérkezése között eltelt időt. Ennek és a fénysebességnek az ismeretében meghatározható a megtett távolság és innen következtetni lehet a lézert visszaverő objektum alakjára is. Ez a

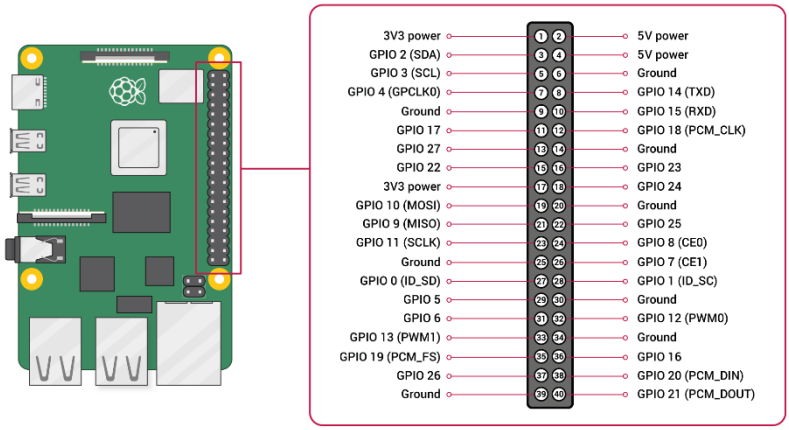
szenzor a legegyszerűbb módon kínál olyan pontfelhőt, amely használatával valósághű térképet lehet alkotni, így kiváltja a 3D kamera használatát.



8. ábra Az RPi foglalat



9.ábra A LIDAR működése (forrás: https://www.slamtec.com/en/Lidar/A1)



10.ábra Az RPi (forrás: https://www.raspberrypi.org/documentation/usage/gpio/)

2.6 A szoftverkörnyezet

A szoftverkörnyezet kialakításánál figyelemmel kellett lennem arra, hogy az RPi kompatibilis legyen vele. Fontos szempont volt az is, hogy egy olyan környezetet válasszak az AMR-nek, amihez rendelkezésre áll elégséges szintű segédanyag. A ROS interfészt választottam, ami az egyik legelterjedtebb módja a robotok vezérlésének. Ebben az interfészben különféle, előre elkészített package-k egyszerűsítik a munkát. A ROS struktúrája olyan, hogy az egész program egy központi maghoz, a roscore-hoz kapcsolódik, és különböző node-ok végzik a számításokat. Egy-egy node megfeleltethető egy programrészletnek, avagy gráfelméleti szempontból a csúcsoknak. A különböző node-okon topic-okon keresztül kommunikálva juttatják el az adatot egymás között. A topic-ok megfeleltethetőek a gráfelméletben is alkalmazott éleknek, amik a csúcsokat kötik össze. A ROS tehát egy olyan operációs rendszer, ami elérhető közelségbe hozza a robotok programozását alacsony szinten. Ez azért is fontos, hogy minél kompaktabb, kisebb legyen a kód, így lecsökkentve az esetleges késleltetéseket a hardver és a szoftver között. A ROS Melodic verzióját használtam, mivel ez az utolsó stabil verzió, amelyhez nagy mennyiségben elérhetőek segédanyagok, illetve ez kompatibilis a „tf” nevezetű csomaggal, ami a lokalizáció alapját adja. Fontos része volt még a projektnek az RVIZ nevezetű vizualizációs alkalmazás, amely valós időben jeleníti a végrehajtott kódot egy modellen keresztül. Ahogy már azt említettem, headless Ubuntu server fut az RPi-n, mivel a ROS Linux alapú rendszerekkel kompatibilis.

A program elkészítésére több lehetőségem is volt, mivel a HEBI támogatja a Matlab, C, C++ és Python nyelveket is. Én a Python mellett döntöttem, figyelembe véve annak egyszerűségét.

2.7 A szoftver

A hardver összeállítása után a szoftver elkészítése volt a következő lépés. Azt fontos tudni, hogy a motorok egyfajta beágyazott rendszerek, tehát közvetlenül lehet programozni őket. Az első nehézséget az szolgáltatta, hogy a motorok különböző típusúak voltak, így különbözik a nyomatékok, sebességük és minden egyéb tulajdonságuk is, ahogy az 1.táblázatban látni lehet.

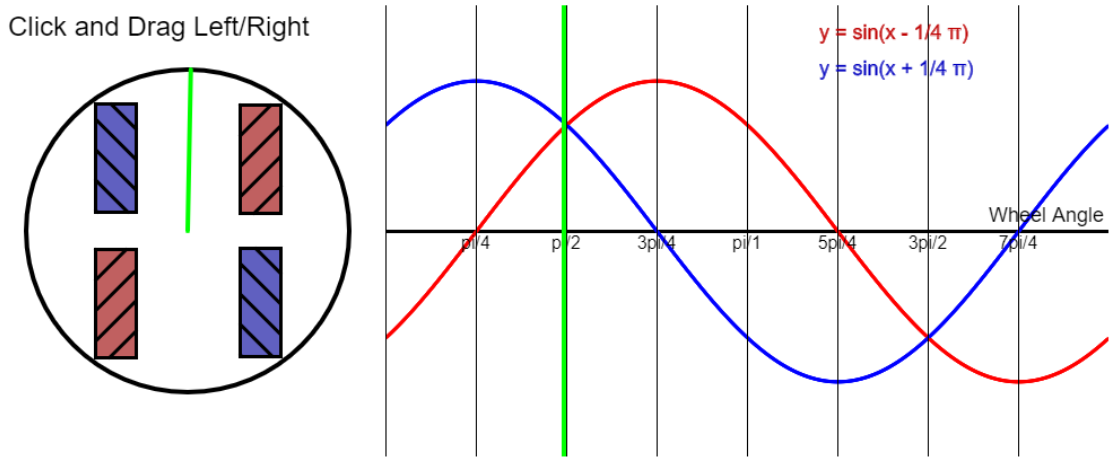
Motor típusa	Áttétel	Nyomaték állandó	Sebesség állandó
X5-1	272.222 : 1	1.1 Nm / A	5.6 RPM / V
X5-9	1742.222 : 1	7.1 Nm / A	0.9 RPM / V

1.táblázat HEBI összehasonlító táblázat

Mindezen tulajdonságok figyelembevételével a motorok PID szabályzóinak módosításával lehet finomhangolni a motorokat. A PID szabályzó tömören azt biztosítja, hogy a motor élethűen le tudja követni a bemenő jelet. Ez azért fontos a projekt során, mivel egy kis eltérés is nagy csúszásokat okoz a mozgás során. Miután a PID szabályzókat konfiguráltam, a következő választás a vezérlést érintette, mivel három döntés állt előttem. Vezérelhettem a sebesség szerint, a pozíció szerint vagy a nyomaték szerint. Mivel az általános felfogás szerint az olyan AMR-eket, amelyek ROS környezetet használnak, úgynevezett Twist üzenetekkel kézenfekvő vezérelni, ezért én is a sebesség szerinti vezérlést választottam. A Twist üzenetek felépítésük szerint tartalmazzák a lineáris és az anguláris sebességeit a vezérlendő objektumnak, így első lépésként létrehoztam a rebiControl vezérlő szkriptet, ami a Twist üzeneteket lefordítja a HEBI motorok nyelvére. Ehhez a HEBI Python API-t [16] vettem segítségül. Ha az AMR mecanum kerekek helyett hagyományos kerekekkel épült volna fel, akkor a vezérlés helyes működéséhez ennyi elég is lett volna. Azonban a mecanum kerék segítségével az AMR holonomikus mozgásra is képes, azaz bármely irányba képes elmozdulni. A hogy a 11. ábrán látható, a mecanum kerék struktúrája olyan, hogy egymáshoz képest 45°-al elforgatott görgőkből áll. Így ez a struktúra lehetővé teszi az ilyen típusú típusú mozgást.

11.ábra A mecanum kerék

Amennyiben felvázoljuk az egyes kerekek mozgását, ahogy ez a 12.ábrán mutatva van, akkor látszik, hogy az egymással párhuzamos kerekek mozgását egymáshoz képest 90°-al eltolt fázisú szinuszhullámok írják le, valamint az átlós kerekek megegyeznek fázisban és értékben egyaránt.



12.ábra A mozgás matemaikája (forrás: https://seamonsters-2605.github.io)

A lineáris mozgások után a fordulásra helyeztem a hangsúlyt, amit a Twist üzenet anguláris részének a *z* irányú komponense ír le. Forgómozgást úgy lehet ezzel létrehozni, hogy a lineáris keréksebességekhez hozzáadjuk az anguláris komponenst, ahogy ez a [2]. hivatkozáson is látható.

3 Az odometria

Mivel az odometria létfontosságú és a dolgozatom erre irányul, így egy külön fejezetet szentelnék neki. Szakszerű megfogalmazása az odometriának az, hogy egy olyan becslés, ami megadja az adott objektum pozícióját és sebességét szabad térben. Kétféleképpen közelítettem meg az odometria számítását, a kerékelfordulás és a IMU szenzorok alapján. Az előzetes várakozás az, hogy a kerékelfordulás által biztosított adat sokkal pontosabb lesz, mint az IMU által szolgáltatott, mivel ezek a szenzorok sokszor pontatlanok, így a kis változásokat nem tudják hűen visszaszolgáltatni. Azonban az ilyen rendszerek jó tulajdonsága az, hogy csak akkor jeleznek változást, ha az tényleg megtörtént, amit a kerékelfordulásból nem lehet kikövetkeztetni. Példának okáért, ha elakadt a jármű, akkor a kerekek ugyanúgy tudnak forogni, de a robot nem halad sehova. Ez az odometriába beleszámít és hibát okoz, ellenben a szenzorok nem fognak elmozdulást jelezni, így ez a hiba megszüntethető. További hibaforrás az, hogy az eszközök nem ideálisak, így például a kerekek mozgásában is tapasztalható csúszás, ami a végső odometriában hibát eredményez. A végső cél az, hogy a két módszer jó tulajdonságait (kerékodometriánál a pontosságot, szenzoroknál az élethűséget) egyesítsem, így előállítva egy olyan odometria számító rendszert, ami mindenre fel van készítve. Tehát az ilyen projektek során a lokalizáció több mint létfontosságú, mivel a térképalkotáshoz muszáj tudni, hogy aktuálisan épp hol jár az AMR és épp milyen irányba néz. Az odometria pont ezt hivatott meghatározni.

Mint említettem, több mód is létezik az odometria meghatározására. Az általam alkalmazott egyik módszer elve, hogy vesszük a kerék végső és kezdő elfordulási pozícióinak a különbségét és ennek a különbségnek a hányadosát az eltelt idővel. Így megkapjuk $\frac{rad}{s}$ mértékegységben a kerekek elfordulásainak a sebességét. Majd ezeket a keréksebességeket az AMR paramétereit figyelembe véve átváltjuk globális (AMR) sebességre a 13. és a 14. ábrán látható mátrixok segítségével. Utolsó lépésként kiszámolom a várható elmozdulást és publikálom azt az adott ROS topic-ra.

$$\begin{bmatrix} v_x \\ v_z \\ \omega_0 \end{bmatrix} = \frac{r}{4} * \begin{bmatrix} 1 & \frac{1}{tan\alpha} & -\frac{L_1 * tan\alpha + L_2}{tan\alpha} \\ 1 & -\frac{1}{tan\alpha} & \frac{L_1 * tan\alpha + L_2}{tan\alpha} \\ 1 & -\frac{1}{tan\alpha} & -\frac{L_1 * tan\alpha + L_2}{tan\alpha} \\ 1 & \frac{1}{tan\alpha} & \frac{L_1 * tan\alpha + L_2}{tan\alpha} \end{bmatrix}$$

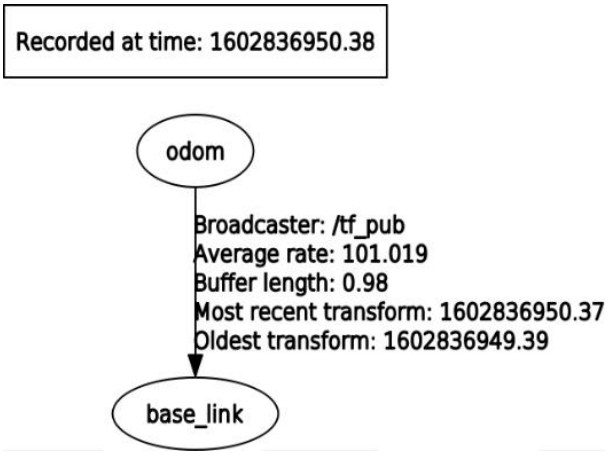
13.ábra Az általános mátrix az átváltáshoz

$$\begin{bmatrix} v_x \\ v_z \\ \omega_0 \end{bmatrix} = \frac{r}{4} * \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -\frac{1}{L_1 + L_2} & \frac{1}{L_1 + L_2} & -\frac{1}{L_1 + L_2} & \frac{1}{L_1 + L_2} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix}$$

14.ábra A mecanum kerékre vonatkozó mátrix

Másik bevált módszer az úgynevezett IMU (Inertial Measurement Unit) használata, amely olyan szenzor adatokat szolgáltat, amelyekből a valósághoz hűen ki lehet számítani az odometria egyes részeit, vagyis az AMR aktuális pozícióját. Ilyen szenzorok például a gyorsulásmérő, a giroszkóp és a magnetométer. Ezek a szenzorok más-más részét adják meg az odometriának, például a giroszkóp és a magnetométer szenzorok által szolgáltatott adatok kombinálásával a szögsebesség és a szögelfordulás nagysága adható meg. A kombinálás elvégzésére az úgynevezett Kálmán-szűrő a bevált módszer. A Kálmán-szűrő úgy működik, hogy a beérkező adatokat fuzionálja, majd ad egy becslést a kimenetre, amely gyakran pontosabb annál, mintha csak egy mérést végeztünk volna. A további méréseknél a Kálmán-szűrő kiszámolja a bizonytalanságot és az aktuális állapotváltozókat, majd az új mérések súlyozott átlagát veszi [17]. Odometria számolásra létrehoztak egy kiterjesztett Kálmán-szűrőnek [18] nevezett változatát az eddig ismertetett Kálmán-szűrőnek. Ez a kiterjesztett Kálmán-szűrő a nemlineáris formája a Kálmán szűrőnek, amely manapság a navigációs rendszerek alapkövének számít. A teljesség igénye miatt leírom, hogy az IMU által szolgáltatott adatokból csak pontatlanul lehet odometriát számolni, így szükséges egy további szenzor a pontos adatok számításához. Legtöbbször ez egy LIDAR-t vagy egy 3D kamerát jelent, ami fel tudja mérni a megtett távot és a környezete változását. Azonban ebben a kutatásban nem használtam a LIDAR-t, csakis az általam számolt odometriákra támaszkodtam.

A ROS-on belül az odometria egy külön üzenettípusként van értelmezve, amely négy tagból áll. Az első az úgynevezett header, amit magyarul fejlécnek lehet lefordítani. Ez adja meg a frame nevét, amiről a transzformációt végrehajtjuk. A következő tag a „child frame id”, amely annak a frame-nek az azonosítója, amelyre a transzformáció el lesz végezve. A maradék kettő egy pose típusú és egy Twist típusú változó a megfelelő kovarianciamátrixszal. A pose típusú változó az a fejlécben szereplő frame-hez viszonyított elmozdulás, míg a Twist üzenet pedig a belőle származó frame-re származtatott elmozdulási sebesség. A kovariancia mátrix pedig azt adja meg, hogy egy többtagú rendszerben, ahol több helyről lehet ugyanazt az információtípust kinyerni, mennyire van súlyozva, azaz mennyire fontos azt figyelembe venni. A koordinátarendszerek, amiket használok a robotnál a map, odom és a base_link rendszerek. Ezek a megszokott konvencióknak megfelelnek, a *REP-105* szabvány alapján. A map frame a világhoz van rögzítve, tehát a mozgás, amit a robot véghezvisz, ehhez képest történik. Ennél fogva ennek a koordinátarendszernek nem szabad elmozdulnia az idő múlásával. A következő frame az „odom frame”, amit az odometria számító node hoz létre. Ez a frame időben folytonos és megkötések nélkül el tud mozdulni, így hosszútávon nem alkalmas arra, hogy az AMR mozgását ehhez viszonyítsuk, ugyanis minél több hiba csúszik az odometria számításába, annál inkább pontatlanabb lesz ez a frame. Ez a pontatlanság pedig az időben előre haladva halmozódik. Az odom frame iránya mindig a robot valós irányába fog mutatni. A base_link frame a robot mozgásának megfeleltethető frame, tehát amerre a valós AMR mozog, arra fog mozogni a szimulációban a frame is. A frame-k egymáshoz való viszonya az alábbi 15.ábrán látható:



15.ábra A TF tree

3.1 Odometria kerékelfordulásból

Ahogy fentebb írtam, az odometria számítás egyik módszere a motorokban található enkóder kimenetéből számolható. Az enkóder ugyanis azt hivatott megmondani, hogy mekkora szögelfordulást hajtott végre a motor. Ezután ezt felhasználva és a 14.ábrán látható mátrix segítségével kiszámítható az elmozdulás. A számítást elvégezve látható, hogy a kerekek mozgását egy szinusz függvénnyel lehet leírni, ami úgy módosul, hogy az egymással szemben lévő kerekek egymáshoz képest $\frac{\pi}{2}$ szögeltolásssal rendelkeznek. Ez alapján felállítottam az [1] és [2] egyenleteket, amik megadják a mecanum kerék mozgásának matematikáját:

$$Left = (\sin\left(forg_{rad} + \frac{\pi}{4}\right) * magnitude * \sqrt{2} + forg_{deg}) * 6.57 * 2$$

1.egyenlet A baloldali keréksebességek kiszámítása

$$Right = (-\sin\left(forg_{rad} + \frac{\pi}{4}\right) * magnitude * \sqrt{2} + forg_{deg}) * 6.57 * 2$$

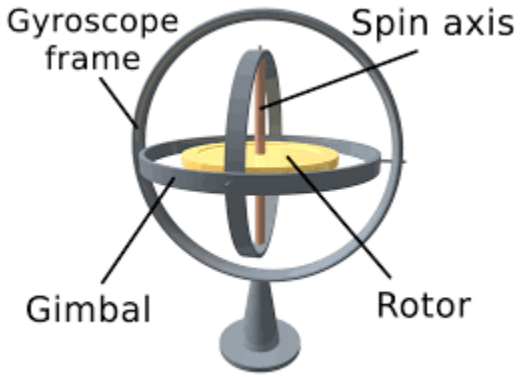
2.egyenlet A jobboldali keréksebességek kiszámítása

Ahol a $forg_{deg}$ az autó forgása fokokban megadva, a $forg_{rad}$ az autó forgása radiánban megadva és a *magnitude* a mozgásvektor nagysága. Úgy adódott, hogy az így kiszámított odometria megfelelően pontos, habár nem szabad elfeledkezni azon hibájáról, hogy elakadás esetén nem tudja meghatározni, hogy valójában haladt-e az AMR.

3.2 Szenzorodometria

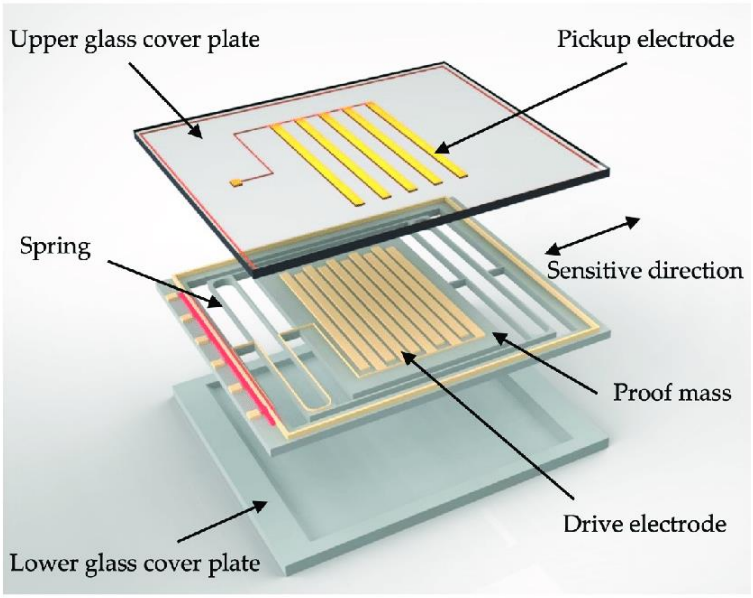
A másik vizsgált módszer az odometria számítására a szenzoradatokból való számítás. A HEBI motorok tartalmaznak giroszkópot és gyorsulásmérőt, amik képesek számomra szükséges gyorsulás, illetve elfordulás adatokat szolgáltatni. Kitekintésképp, a giroszkóp egy olyan eszköz, amely szögelfodulást és szögsebességet képes mérni, a perdületmegmaradás elve alapján. Ahogy a 16.ábrán láható, kezdetben ez lényegében egy tengelyre függesztett korong volt, amelynek forgástengelye bármilyen szöget felvehet. A HEBI-ben lévő giroszkóp mikroelektronikai technológiával készül és egy három elemű vektorban tárolja a különböző tengelyeken felvett szögsebességeket. A másik általam használt szenzor a gyorsulásmérő, amely egy szintén három elemű vektort állít elő a három dimenziónak megfelelően.

Itt azt is figyelembe kell venni, hogy a függőlegesen mutató tengelyen megjelenik egy konstans lefelé irányuló 9.81 $\frac{m}{s^2}$ nagyságú gyorsulás, ami a gravitációs gyorsulásnak feleltethető meg. Manapság ezt a két szenzort kombinálják egy áramkörben: ezt hívják mikro elektro-mechanikai rendszereknek (**MEMS**). Az ilyen rendszereket kifinomult technológiával gyártják, amelyek már képesek egy 6 szabadságfokú gyorsulás vektort előállítani. A 6 szabadságfok az x, y, z irányú lineáris és szöggyorsulások.



16.ábra A giroszkóp

Ahogy a 17.ábrán is látható, a MEMS gyorsulásmérő úgy működik, hogy gyorsulás hatására a két elektróda közeledik vagy távolodik egymástól a gyorsulás irányától függően, így változik a kettő elektróda között mérhető kapacitás nagysága és ezt felhasználva lehet kiszámítani a gyorsulást.



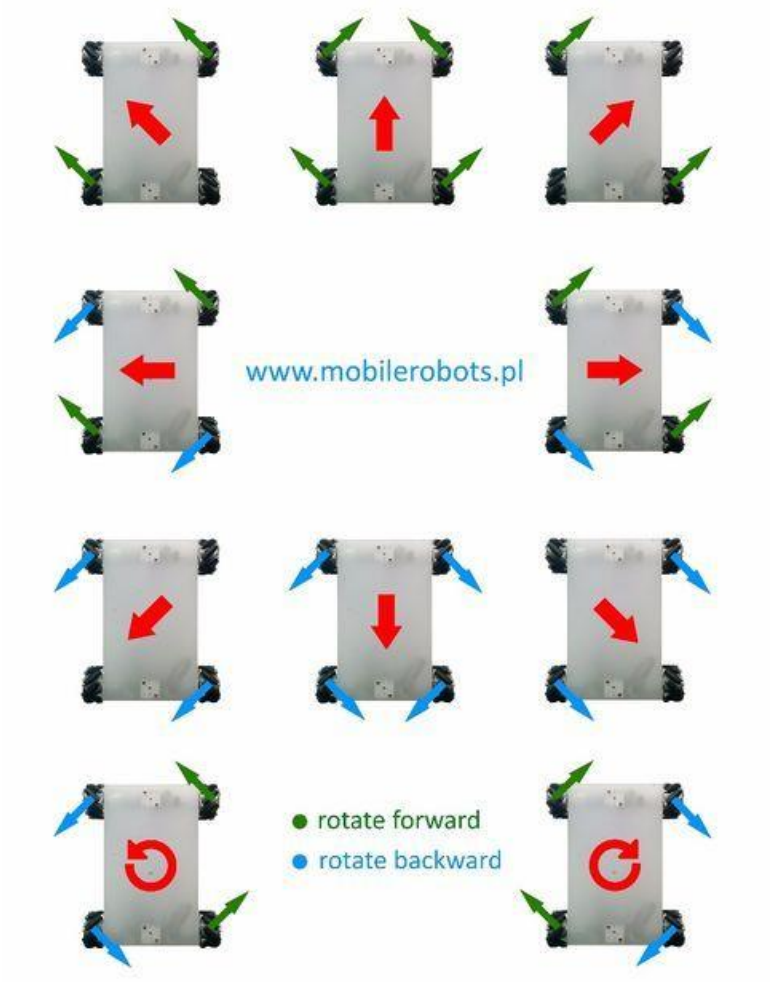
17.ábra MEMS gyorsulásmérő (forrás: Wikipédia)

Az én szempontomból ezek a szenzorok azért fontosok, mivel a lineáris gyorsulásból az odometria adatokat ki tudom számítani, ha az idő szerint folyamatosan integrálom a beérkező adatokat.

4 Előzetes eredmények a témában

A témában ezidáig sok kutatás zajlott, amelyek az én kutatásomat érdemben elősegítették, így ebben a fejezetben elemzem és kiértékelem a kutatáshoz felhasznált irodalmat és tudományos cikkeket, amik hasznosnak bizonyultak a kutatás során.

Először a *Yunwang Li, Sumei Dai, Yuwei Zheng, Feng Tian és Xucong Yan* [1] által elért eredményeket szeretném kiemelni, amik abban segítettek, hogy a jármű mozgását a valósághoz hűen tudjam modellezni Python nyelven. A mecanum kerekek kialakítása miatt, ugyanis a mozgást bonyolultabb előállítani, mint hagyományos kerekű társaik esetében. A 18.ábrán látható a kerekek meghajtásának a logikája, valamint az így elért irányok.



18.ábra A mozgás logikája (forrás: www.mobilerobots.pl)

Alább látható a 2.táblázat, ami egy olyan általam elkészített táblázat, amit a fentiek alapján állítottam össze és kifejezetten az általam épített robotra vonatkozik (a neve Rebi).

REBI mozgás

Kerékazonosító	Előre	Hátra	Jobbra	Balra	Jobbfel	Balfel	Jobble	Balle	ForgásJobbra	ForgásBalra
FrontLeft	+	-	+	-	+	0	0	-	+	-
BackLeft	+	-	-	+	0	+	-	0	+	-
FrontRight	-	+	+	-	0	-	+	0	+	-
BackRight	-	+	-	+	-	0	0	+	+	-

2.táblázat Rebi mozgás

További eredményük, amiket felhasználtam a 13. és 14. ábrákon található mátrixok, amik a globális autó sebességét váltják át kerék sebességekre bármilyen mozgás esetén.

Másik megközelítése a problémának a *Dawson Bowhay* által megalkotott modellen [2] keresztül történt. Az ő kutatása is célravezető volt, azonban ő arra a felismerésre alapozta a kutatását, hogy a kerekek mozgása megfeleltethető két szinusz hullámmal, amik egymáshoz képest 90 fokkal vannak eltolva. Az elmélete, amely leírja a mecanum kerekes robotok fordulásának kinematikáját, a teszteken az elvárt működést mutatta. A kódomban apróbb módosításokkal felhasználásra került a vezérlés kialakításánál.

Ezeket a kutatásokat tanulmányozva és felhasználva a kerékodometria számító algoritmust, amit a végső kódban is használok. Miután már a kerékodometriát megbízhatóan elő tudtam állítani a következő lépés a szenzorodometria előállítása volt. Ezen a területen is előzték meg kutatások az enyémet így ezeket felhasználva alakítottam ki a saját rendszeremet.

Először is *J. Shen, D. Tick és N. Gans* kutatását [3] vizsgáltam, ahol a vizuális, kerék és IMU odometria fuzionálásának jótékony hatását vizsgálták. Itt szembesültem a Kálmán–szűrő használatának előnyeivel, amely kitűnően használható olyankor, ha több szenzor

kimenetét akarjuk fuzionálni, hogy redukáljuk az egyes szenzorok pontatlanságából, kvantálási hibájából, vagy drift-jéből eredő halmozódó hibát. A tanulmányból mélyebben elsajátítottam a Kálmán-szűrő működését, így az általam használt Python implementációja a Kálmán-szűrőnek is pontosabb eredményekkel szolgált.

További eljárásokat ismertem meg a *Song, Y., Nuske, S., és Scherer*, S. által végzett kutatásból [4], amely légi járművek helyzetbecslését vizsgálta kiterjesztett Kálmán-szűrő (EKF) és IMU-k használatával. Ez a kutatás a szűrők alkalmazásának és a gravitációs komponens módszerével behatóan foglalkozik, amely hasznos volt a kutatásom során, mivel a motorokban elhelyezett gyorsulásmérő nem kompenzálta ki a gravitációs gyorsulásból eredő komponenst, amely konstans hibát okoz a rendszerben.

Hasznos tanulmány volt továbbá a *Xue, H., Fu, H. és Dai, B*. [5] által kifejlesztett megközelítés. A kutatásuk egy újfajta megközelítést ajánl az IMU kimenetének a feldolgozására. Létrehoztak egy olyan rendszert, amely folyamatosan, több lépésben fuzionálja az IMU kimenetét a LIDAR kimenetével. Ebben a kutatásban ugyan használtak LIDAR-t, tehát az én szempontomból nem volt olyan konkrét, releváns eredmény, amit felhasználhattam volna, de a módszertan elsajátítására, és az ilyen rendszerek részleteinek elemzésére kiváló kutatás.

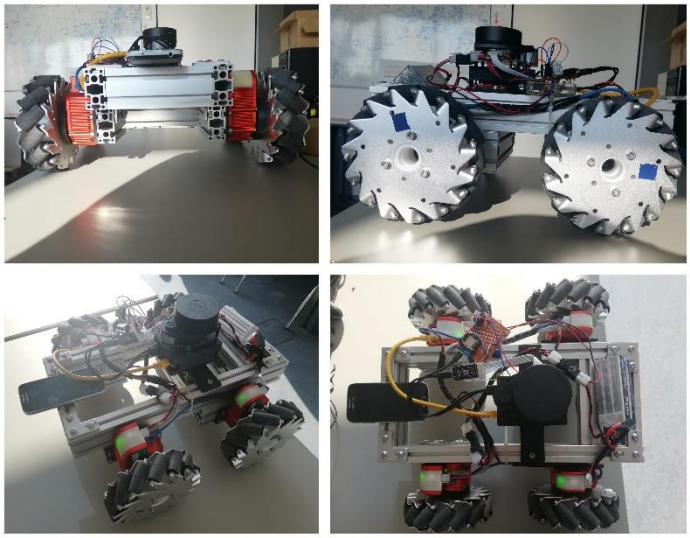
Egy másik megközelítést tartalmaz a *R. K. Megalingam, D. Nagalla, K. Nigam, V. Gontu és P. K. Allada* kutatása [6], amely egy többfajta terepen is közlekedni tudó, PID szabályzó által vezérelt robotot vizsgál. A kutatásomban a PID szabályzók paramétereinek beállításában és algoritmikus megközelítésében nyújtott segítséget ez a tanulmány. Közvetve a PID szabályzók pontosságának és az ebből eredő hibák minimalizálását segítette elő az általuk kidolgozott algoritmus, ami azért volt jelentős, mivel a motorok vezérlése és a vezérlés pontos kivitelezése nagyban befolyásolja az odometria megbízhatóságát.

Az odometria számítását egy már bevált módszer alapján csináltam, a [13]. forrás által megadott példakód mintájára. Ez a kód szabad felhasználású, a ROS wikipédia oldalán van feltüntetve a segédanyagok között.

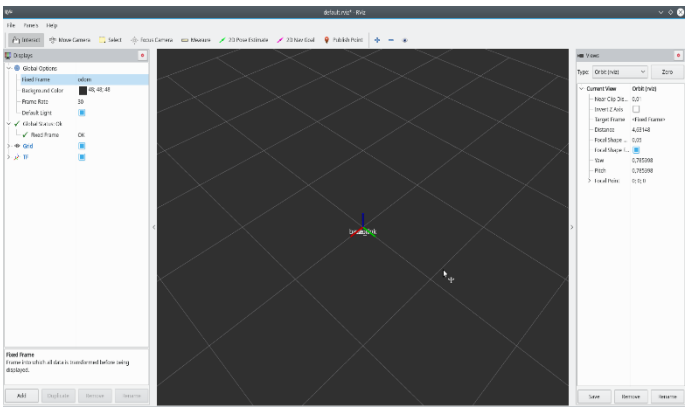
A kutatásom során hasznosnak találtam még különböző fórumokon fellelhető segédanyagokat, szabványokat, példakódokat és tutorial-okat, ezeket az irodalomjegyzékben egyesével feltüntettem.

5 A kutatás menete

A kutatásom során a 19.ábrán is látható, általam épített négykerekű AMR-n (továbbiakban Rebi) keresztül vizsgáltam az egyes módszerek hatékonyságát és pontosságát. A következőkben ezeket a módszereket fogom ismertetni. Minden mérés a Budapesti Műszaki és Gazdaságtudományi Egyetem I épületének a 326. laborjában zajlott, illetve az előtte elhelyezkedő folyosón. A kutatás eredmények alátámasztására az RVIZ nevű 3D vizualizációs eszközt használtam, ami a ROS tartozéka, és közvetlenül tud ROS üzenetekkel dolgozni. Az RVIZ-ben az odometria számításának az eredménye nyomon követhető, úgy mintha ideális körülmények között zajlott volna a mérés, így ellenőrizhető, hogy az odometria jellegre és nagyságra helyes eredményeket produkál-e. Az RVIZ ellenőrzésére a valóságban is elvégezhettem a távolságmérést. Ezt egy, a padlóra felvázolt 1 méter hosszú és 1 méter széles négyzetrácson végeztem el. Az RVIZ felülete, és az odometria nyomon követhetősége az 20. ábrán látható.



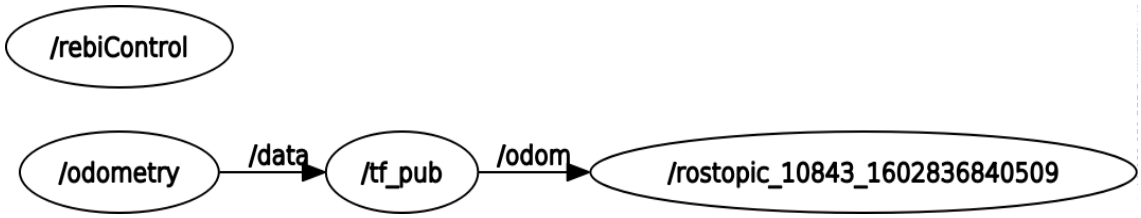
19.ábra a REBI



20.ábra Az RVIZ felülete

5.1 A kerékodometria

A kerékodometria számítására irányuló kutatást a hardver feltérképezésével kezdtem. Fontos megjegyezni, hogy a HEBI által gyártott motorok olyanok, mint egy beágyazott rendszer önmagukban, tehát többféle szenzort tartalmaznak az ilyen esetek megkönnyítésére. A kerekek elfordulását az enkóder kimenetéből egyszerűen kiszámíthattam, és az elfordulás hányadosa az eltelt idővel pedig a kerekek sebességét adta eredményül. Az így megkapott sebességeket pedig a már ismertetett 13.ábrán szereplő mátrix segítségével átváltottam AMR sebességre, ami az odometria számításához elengedhetetlen. Végül az odometriához szükséges helyváltozást szögfüggvények segítségével kiszámoltam. Mindezt becsomagoltam egy ROS üzenetbe és publikáltam az /odom topic-ra, amit az RVIZ is értelmezni tud, ez látható a 21.ábrán.

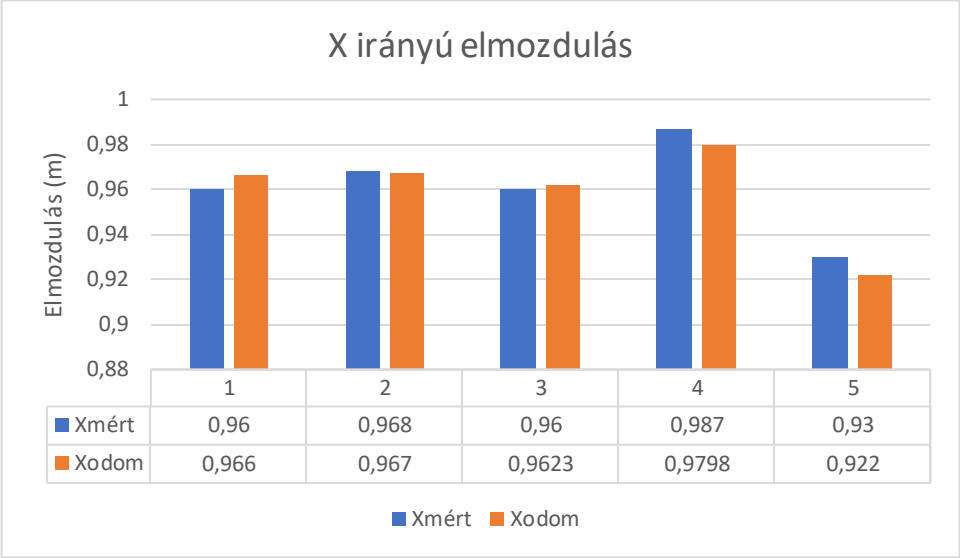


21.ábra A kerékodometria rendszere

Az így előállt odometriával egy kezdetleges navigációs rendszer is kialakítható már, azonban ez azért nem kielégítő eredmény, mivel sok hibalehetőség rejlik még ebben a rendszerben. Egyik például az, ha elakad a robot és a kerekek forognak, de nem halad a jármű, viszont az odometria továbbra is azt jelzi, hogy előrehaladás történt. További hibaforrás például, ha valamilyen súlyt kell vontatnia a robotnak, amire nincs felkészítve. Ez az eset is a kerekek csúszását eredményezi.

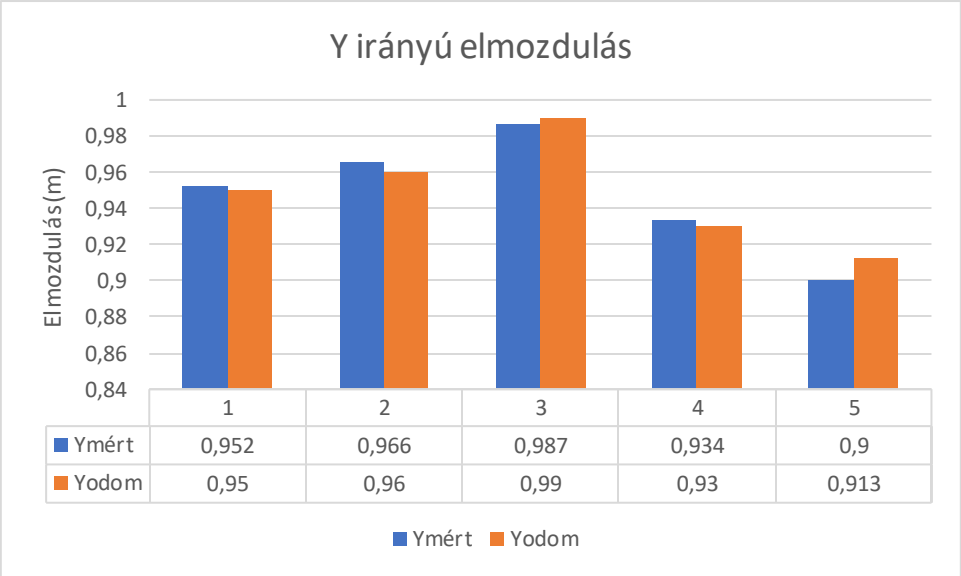
A 22. 23. 24. diagramokon a legfőbb mozgások teszteseteit mutatom be. A 22.ábrán az X irányba történő elmozdulást mutatom be kétfajta mérésből. Egyiket a robot számította, a másikat én mértem le a fentebb is említett padlóra vázolt négyzetrács segítségével. A mérést úgy végeztem el, hogy kiadtam a robotnak az utasítást, majd adott távolság után megállítottam és lemértem a mozgás X irányú komponensét, majd az eredményt összevetettem a robot által mért eredménnyel. A robot által előállított adatokat narancssárgával, az általam mért adatokat kék színnel ábrázoltam. Láthatjuk a 21.ábrán,

hogy a két mozgás jellegre megegyezik, az átlagos hiba 0.49%. Összeségében kijelenthető, hogy az X irányú elmozdulás megbízhatóan zajlik.



22. ábra Kerékodometria elmozdulás X irányba

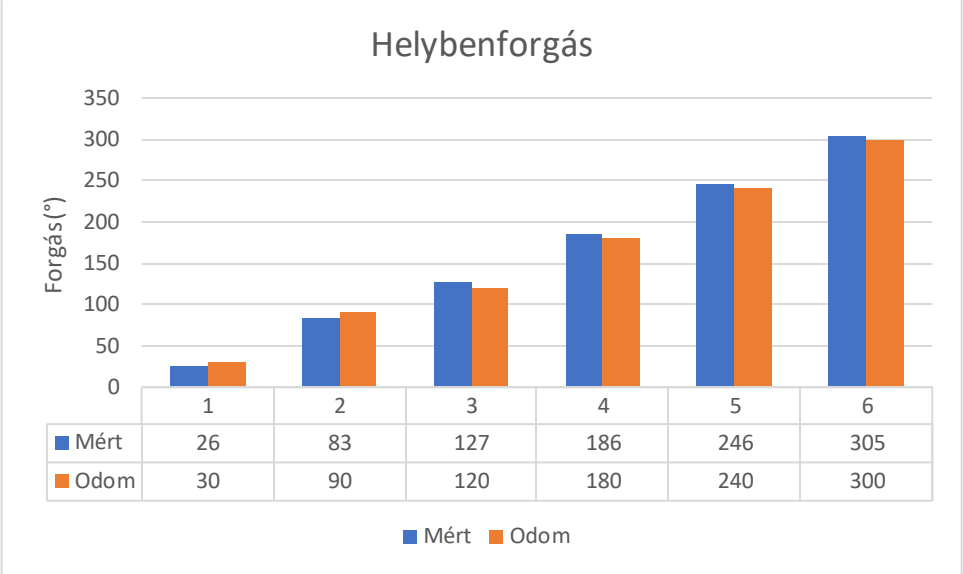
Az alább látható 23.ábra a tisztán Y irányú elmozdulásra vonatkozik. A mérési környezet, elv és a jelölésmód az előbbiekkel megegyezik, a konklúzió pedig az, hogy a mozgások jellegre itt is megegyeznek.



23.ábra Y irányú elmozdulás

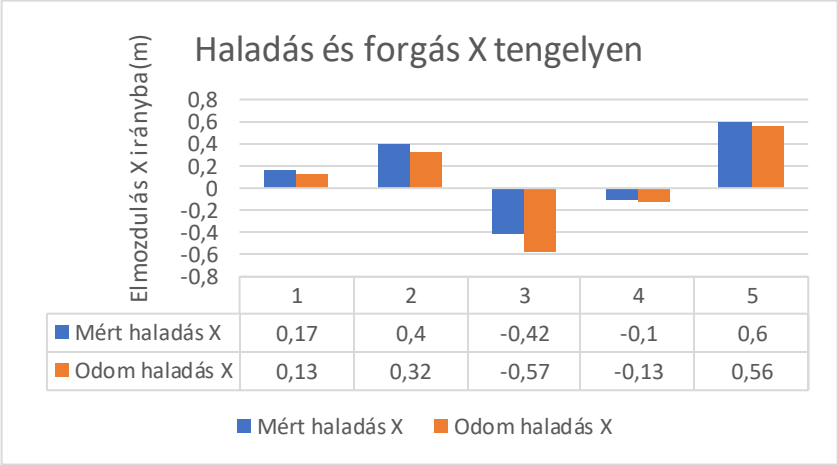
A hiba mértéke ebben az esetben 1.012%-nak adódott, de ez, az általam kívánt rendszerhez teljes mértékben megfelel.

A következő alapmozgás az egyhelyben való forgás volt. Itt a mérési elv az, hogy pontosan megjelöltem a padlón a középpontot, továbbá a robot elejének a kezdő és végső orientációját, majd meghatároztam a kettő közötti szöget. A 24.ábrán láthatóak a mérési eredmények. A rendszer ebben az esetben átlagosan 1.667%-ot téved, ami komoly rendszereknél a megengedhetetlen kategóriába esik.

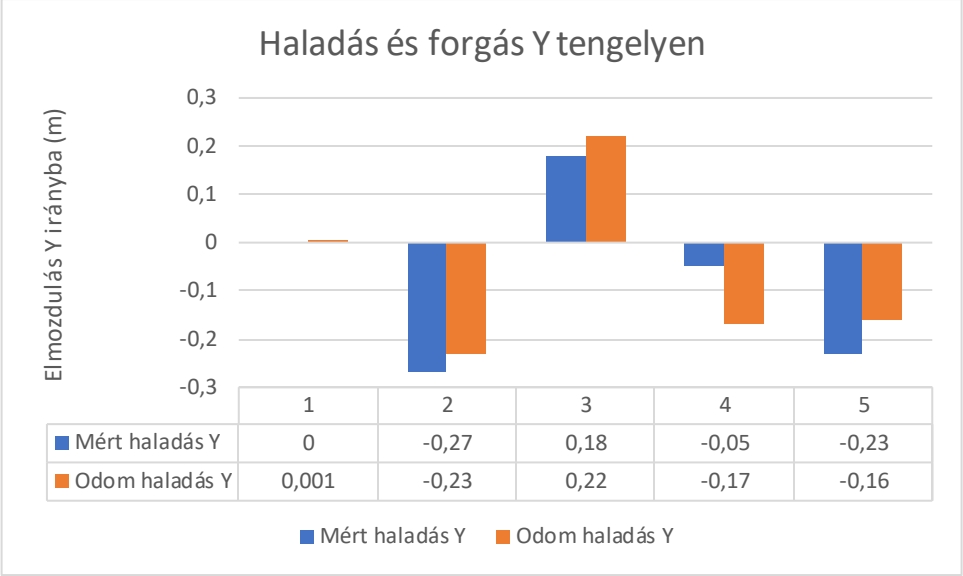


24.ábra A helybenforgás

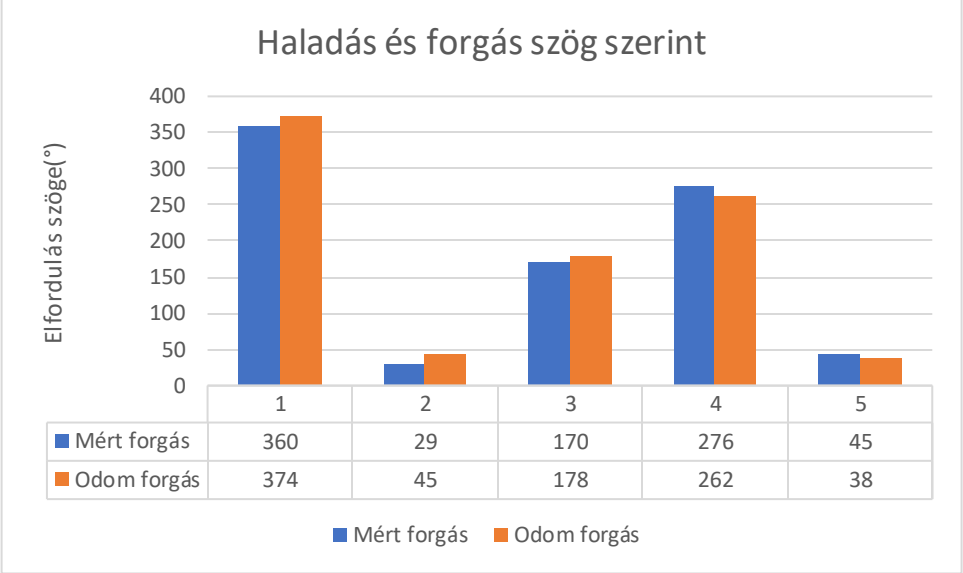
A következő három ábrát egyben tárgyalnám. A mozgást komponensekre bontva tárgyalom, a 25.ábrán az X irányú, a 26.ábrán az Y irányú és a 27.ábrán pedig forgás szerint. Látható, ha a két mozgást kombináltam, akkor az odometria lényegesen eltér a mért, valós odometriától. Ez az eltérés a környezet nem-ideális voltának, a kerekek csúszásának és a mérési hibának tekinthető. Az ábrákon az előzőknél is használatos jelölésmódot alkalmaztam.



25.ábra Haladás és forgás X irányú összetevő

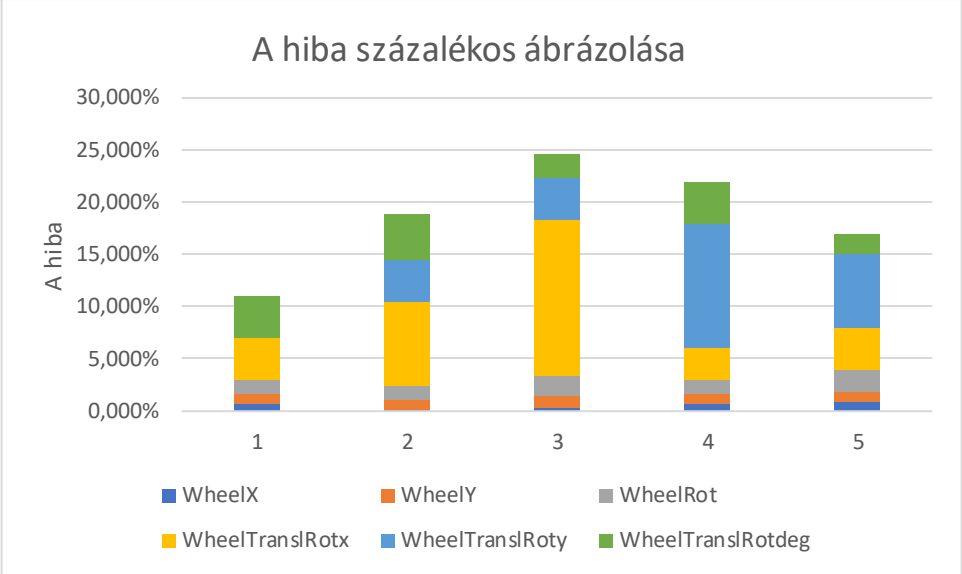


26.ábra Haladás és forgás Y irányú összetevő



27.ábra Haladás és forgás szöghibája

Szemléltetésképp készítettem egy összesített ábrát, amelyen feltüntetem az egyes komponensek mérésenként produkált százalékos eltérését a mért értékhez képest. Ezeket az értékeket összegeztem a 28. ábrán így látható, hogy az egyes hibák mekkora részét teszik ki a teljes hibának.



28.ábra A hibák százalékban kifejezve

A 28.ábrán látható, hogy a hiba nagy része akkor következik be, ha a mozgástípusokat kombinálom, tehát egyszerre történik a haladó és a forgó mozgás.

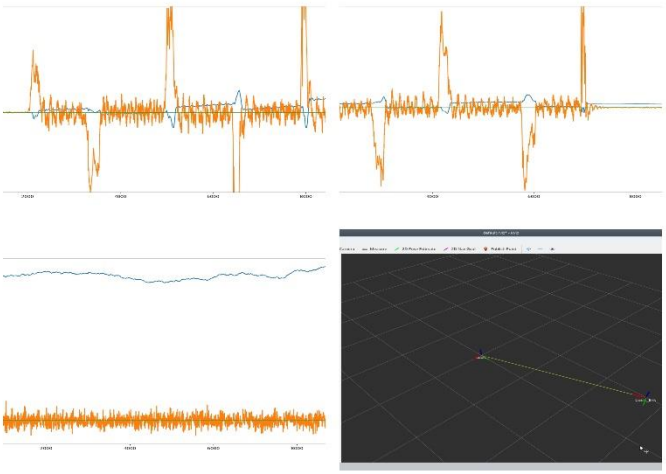
5.2 Az IMU odometria

A kutatás következő lépése egy megbízható odometria rendszer kidolgozása kizárólag az IMU által szolgáltatott adatokra hivatkozva. A HEBI-ben gyorsulásmérő és giroszkóp található, mint helymeghatározásra alkalmas szenzorok, így ezzel egy 6 szabadságfokú odometria rendszer alkotható. Ez azt takarja, hogy az elmozdulásokat csak és kizárólag a szenzorok által mért gyorsulás határozza meg. Mivel a gyorsulás a sebességvektor idő szerinti első deriváltja, így véges időtartammal a számolva a sebesség megkapható az 3.egyenletet átrendezve.

$$v = v_{init} * a * t$$

3.egyenlet A sebesség kiszámítása

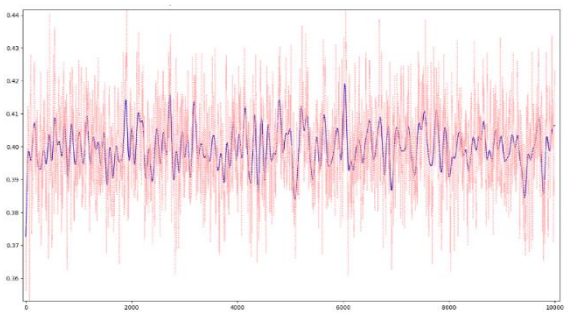
Azonban ez az eljárás csak nagyon rövid ideig helytálló, mivel a számunkra fontos adatot úgy kapjuk meg, hogy egy hibákkal rendelkező mérést integrálunk. Ezzel előidézve egy lavina-szerű effektust, ami azáltal keletkezik, hogy a mérési hibát folyamatosan integráljuk, ezáltal az exponenciálisan nő, végül elnyomva a valós jelet, így teljesen elrontva az adatot. A mérési eredményeim így csak az első másodpercekben voltak megbízhatóak, utána az odometria a végtelenhez konvergált.



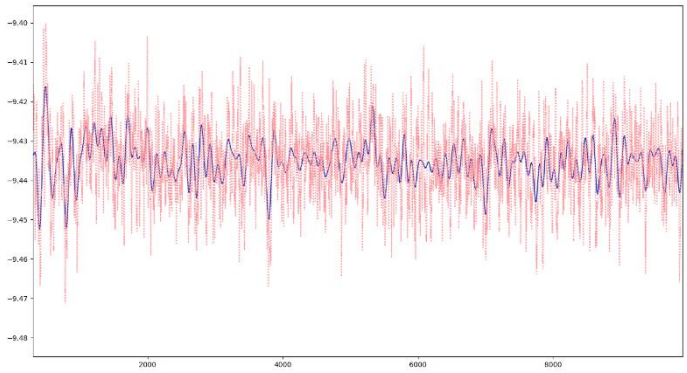
29.ábra Az IMU lavinahatás

A 29.ábrán az látható, ahogy az IMU interpretálja a robot előre és hátra való mozgását, majd teljes megállását. Narancssárga színnel az Y irányú gyorsulás, kékkel az ebből előállított sebesség látható. Az ábrán megfigyelhető továbbá az is, hogy a mozgás után megjelenik a jelben egy ofszet, ami azt eredményezi, hogy az odometria folyamatosan nő, álló helyzetben is. Az odometria hibáját a 29.ábra jobb alsó sarkában figyelhetjük meg, az RVIZ szoftverben. A többi alapmozgás mérésének eredményét nem tüntetem fel, mivel a hiba független a mozgás irányától.

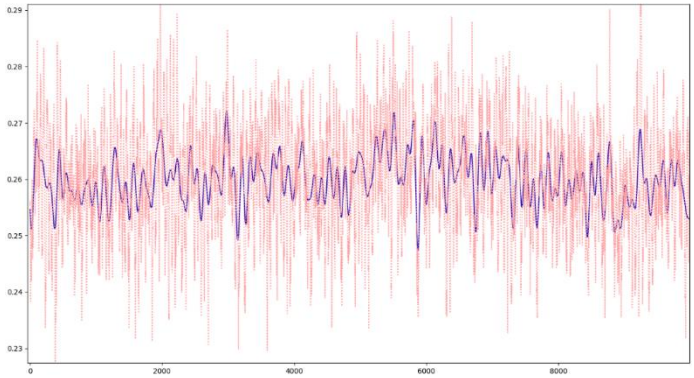
Elterjedt szokás ilyenkor az adatok előszűrése, hogy az integrálás során minél megbízhatóbb adatokkal dolgozhassunk. Én előzetes tanulmányaim alapján egy Butterworth szűrőt [15] választottam ezen cél elérésére, ami az egyik leggyakrabban használt szűrő mozgási folyamatok analizálásánál. A szűrő Python-ban való implementálását a scipy [14] Python csomaggal végeztem. Ezzel a szűrővel csillapítani tudtam a nem kívánt zajokat, csökkentve ezzel a szenzorok hibáit, ahogy az alábbi 30. 31. 32. ábrákon ez látható.



30. ábra Az IMU szűrt X irányú kimenete



31. ábra Az IMU szűrt Z irányú kimenete



32. ábra Az IMU szűrt Z irányú kimenete

A fenti 30. 31. és 32. ábrákon látható az IMU eredeti és a szűrővel megszűrt jele. Az eredeti halvány rózsaszínnel szaggatottan, a szűrt jel kékkel van ábrázolva. A függőleges tengely a gyorsulást ábrázolja $\frac{m}{s^2}$ -ben, míg a vízszintes tengely az időt ábrázolja *ms*-ban. Amint látható a szűrt és az eredeti jelalak mindegyik esetben megegyezik jellegre, illetve az is látható, hogy a 31.ábrán, az Y tengelyen látható a gravitációs komponens a várakozásoknak megfelelően. Ahogy a mérések során kiderült, további tényező is felmerült, ami elronthatta az odometriát. Ez a gravitáció jelensége volt, ugyanis a szenzorjaim nem lineáris gyorsulást mértek, hanem egy olyan gyorsulásvektort, amiben a gravitáció jelen van, mint egy konstans lefelé mutató, körülbelül 9.81 $\frac{m}{s^2}$ nagyságú gyorsulásvektor komponens. Ennek a kikompenzálására több szempont szerint megközelítettem a problémát.

5.2.1 A gravitáció kompenzálása

Az egyszerű megoldásoktól haladva az első irány, a gravitációval közvetlenül befolyásolt tengelyen a gyorsulás zérus kompenzálása volt, úgy, hogy kiegyenlítettem a gyorsulást egy konstans hozzáadásával. Ez nem hozott kielégítő eredményt, mivel a többi

tengelyen a padló egyenetlenségéből származó eltérés továbbra is jelen volt, illetve ez egy elég rugalmatlan megoldás, ami hosszútávon sem megfelelő. A következő megfontolás [7] egy aluláteresztő szűrő alkalmazása, ami nem kompenzálja ki a gravitációt közvetlenül, hanem megakadályozza a hirtelen változásokat úgy, hogy az előző eredmények átlagát veszi és az az új eredménynek a szűrési faktornak megfelelő hányadát veszi, ahogy ez a 4.egyenletnél látható. Ez a megoldás kiszűri a gravitációt, amit ezután az eredeti jelből kivonva megkapjuk a számunkra hasznos jelet.

$$g_{comp} = 0.9 * g_{comp} + 0.1 * accelvector$$

4.egyenlet Az aluláteresztő szűrő

Ez egy érdekes megközelítés volt jelfeldolgozási szempontból, de az én esetemben nem nyújtott maradandó megoldást. A lavina effektus megszűnt, de a mozgás nagyon pontatlan lett, mivel nem lehet gyors mozgásokat végrehajtani így.

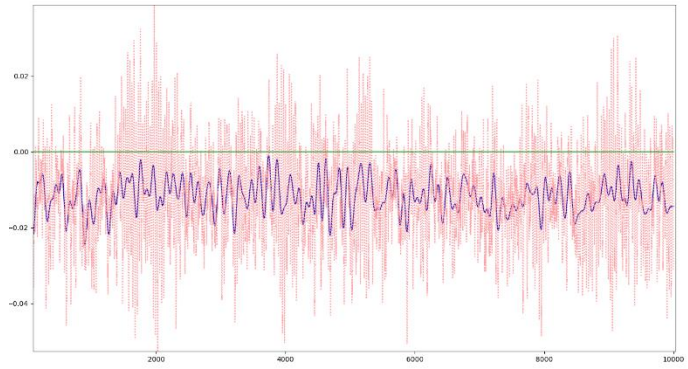
A harmadik, egyben utolsó megfontolás komoly matematikai számításokat igényelt, de kielégítő eredményt adott eredményül. Az elgondolás a módszer mögött az volt, hogy a szenzor képes orientációt is biztosítani. Az orientáció kvaterniók formájában írja le a forgást, így felhasználható a kompenzáláshoz is. A gyorsulást tehát a Föld rendszerébe forgatva, majd kivonva belőle a gravitációs vektort megkaphatjuk a tisztán lineáris gyorsulás értékeket. A kód, amit felhasználtam komoly matematikai kutatások eredménye, amit Fabio Varesano implementált [8]. Az algoritmust, amit megalkotott nem részletezném, csupán a könnyebb megértés kedvéért biztosítom a matematikai segédeszközt. A kódhoz ki kell számítanunk egy *k* kvaternióval elforgatott *v* vektort, amit az 5.egyenlet alapján tudunk kiszámítani:

$$\boldsymbol{v_{rot} = k * v * k^{-1}}$$

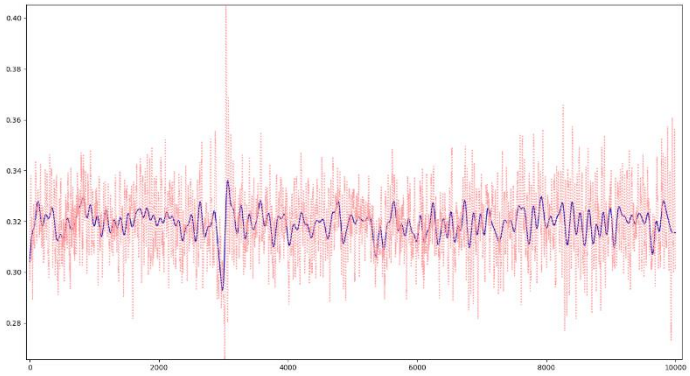
5.egyenlet Vektor forgatás adott rotációval

A használt matematikai eszközök bővebb megértéséhez ajánlom a [9] és [10] forrásokat, amelyek a Wikipédia és a Budapesti Műszaki Egyetem Számítógépes Grafika című Videotóriumos elérhetőségére mutatnak.

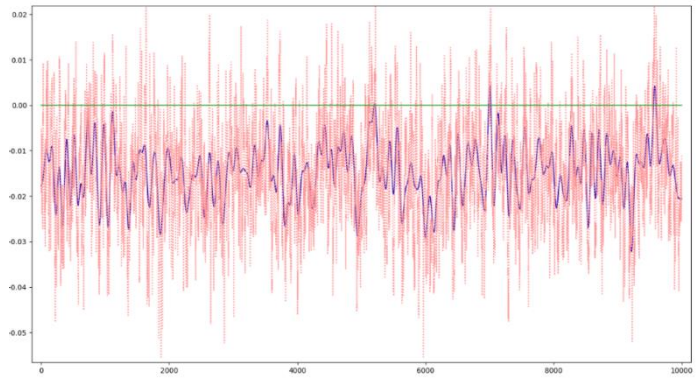
Ezt a módszert Python nyelven lekódolva és megvizsgálva elégséges eredményt adott. A tengelyeken állóhelyzetben közel zérus gyorsulás volt mérhető. Továbbá a kutatás során látszott, hogy a szűrő is javított a végső eredményen, így a két rendszert együtt használtam, amivel az alábbi, 33. 34. 35. ábrákon látható eredményeket mértem.



33.ábra Az IMU kompenzált kimenete X



34.ábra Az IMU kompenzált kimenete Y



35.ábra Az IMU kompenzált kimenete Z

Ahogy a fenti 33. 34. 35. ábrákon látható, sikerült egy zajos, ofset-el terhelt jelet egy kezelhető, mérsékelten zajos jellé szűrni. Miután sikerrel végeztem a kompenzálásra irányuló kutatásokkal, a mért és immár kompenzált eredményekből előállítottam az odometriát, ami immáron kizárólag IMU adatokra támaszkodott. A mérések azt bizonyítják, hogy a lavinahatás továbbra is fennállt, ami annak köszönhető, hogy az IMU-

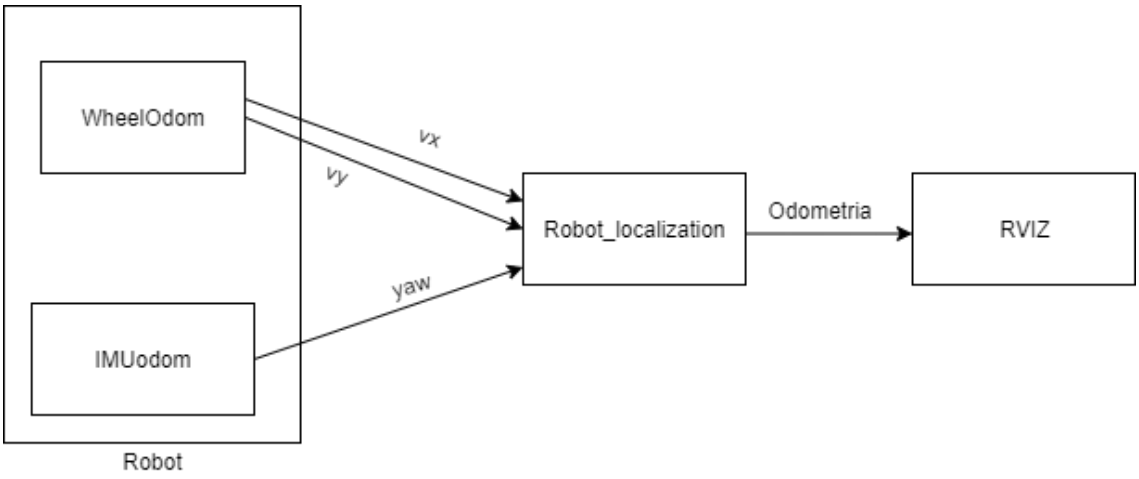
k elmozdulás után változó hibával mérnek, tehát nincs egy konstans ofset -je, amivel számolni lehetne, így az odometria hosszútávon szintén pontatlannak bizonyult.

Erre irányuló fejlesztés és lehetséges megoldás az, ha adott időközönként frissítem a kezdeti sebességet zérus kezdőértékkel. Ez azonban olyan mértékű pontatlanságat vitt be a rendszerbe feleslegesen, amely már nem volt tolerálható. Ugyanis azzal, hogy manuálisan beállítottam a kezdőértéket adott időközönként, a program futása ennyivel eltolódott és az odometria nem lett sokkal pontosabb, de még késett is a valósághoz képest. Így tehát a kutatás során bebizonyosodott, a már előrevetített eredmény, miszerint az odometria, amely kizárólag a kereskedelemben megtalálható IMU-k által szolgáltatott adatokra támaszkodik, hosszútávon nagyon pontatlan, rövidtávon meg lassú. Ellenben az iparban és a hadiparban használatos IMU-k pontossága már önmagukban meghaladják azt a szintet, ahol már kellő precizitással meghatározható az odometria pusztán az IMU-k kimenetéből. Mivel nekem nem állt rendelkezésemre ilyen szintű techológia, a minél pontosabb eredmény érdekében a telefonomat használtam, mint IMU, mivel a mai mobilkészülékek a gyorsulásmérő és giroszkóp mellett tartalmaznak magnetométert is, melynek segítségével pontosabb odometria adatokat lehet velük előállítani. Az ilyen rendszerekhez azonban a megfelelő eredmény érdekében szükséges illeszteni további szenzorokat, ez lehet kamera, GPS, enkóder, LIDAR, vagy bármilyen, pozíció meghatározására alkalmas szenzor. Itt pedig szükségessé válik a különféle szenzorok fúziója, amelyre jelenleg korlátozott számú módszer áll rendelkezésre, mecanum kerekes rendszerre pedig elenyésző mennyiségű kutatás létezik.

5.3 Szenzorfúzió

A fúzióra bevett eljárás a *robot_localization* [11] nevű ROS package, ami egy olyan nemlineáris állapot becslést végez, amely képes kiszámolni a bemenetből a várható kimenetet. Ez a csomag bármennyi bemeneti forrást képes fogadni, illetve kiszámolni belőlük a várható mozgás sebességét és az elmozdulás mértékét. Tehát fel tudja dolgozni IMU-k vagy akár kész odometriák adatait is, fuzionálva őket a felhasználó alapján bekalibrált kovariancia adatok alapján. A kovariancia- mátrixban megadhatjuk, hogy a beérkező adatunk, avagy maga a szenzorunk mennyire megbízható, mekkora mérési bizonytalanságot tartalmaz. Ez a csomag akkor működik optimálisan, ha sok bemeneti forrást tudunk biztosítani, mivel a sok zajos jelből képes előállítani egy megbízhatót.

Esetemben, ahogy a 36.ábrán is láthatjuk, azonban csak két forrás áll rendelkezésre, a kerékodometria és a telefonom IMU-ja által szolgáltatott odometria. Az IMU odometria ugyanakkor csak egyes aspektusaiban megbízható, így rontana a helyzeten, ha minden komponenst belevennék a fúzióba. Tekintve, hogy az IMU sokkal több szenzor kimenetéből tud dolgozni, így az általa előállított orientáció is sokkal pontosabb. Köszönhető ez főként a magnetométernek, ami Hall effektus segítségével méri a mágneses térerősség változását, amiből akár az égtájakat is meghatározhatjuk. Így van egy biztos viszonyítási alap, ami sokkal megbízhatóbbá teszi a fordulás, avagy konvenció szerint a yaw (fordulás a Z tengely körül) mérése. Így a minél nagyobb pontosság eléréséhez célravezető az IMU yaw mérését, a kerékodometria lineáris mérésével. Azonban ez még mindig okoz tévedést, mivel a robot_localization package figyelembe veszi a kerékelfordulás alapján számolt fordulást is, megjósolja azt, így rontva a végső odometrián, tehát további javításokra volt szükség.



36.ábra A robot_localization felépítése a rendszerben

5.4 Az általam kidolgozott algoritmus

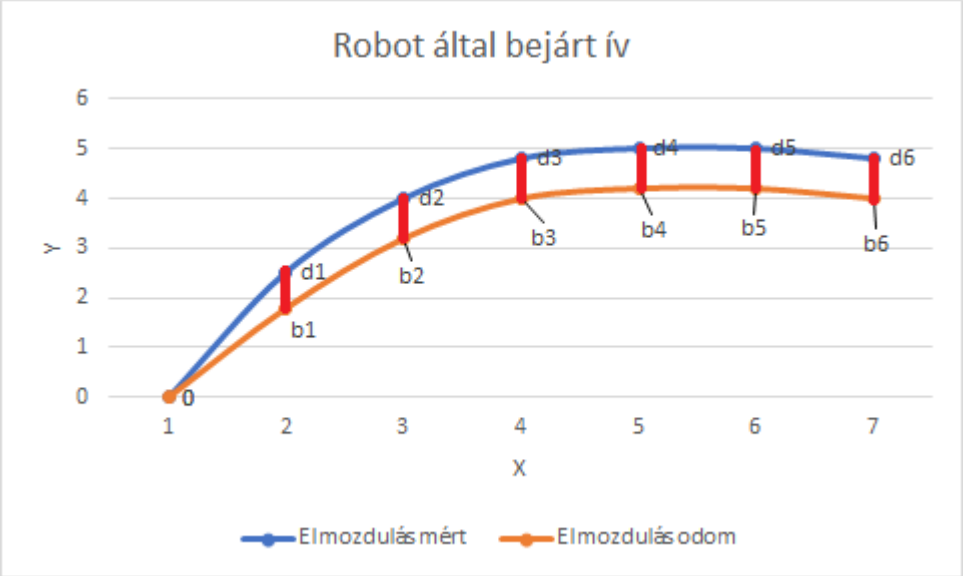
Az előző megfigyelések alapján új megfontolásra volt szükség, amivel tovább javítható a rendszer pontossága. Az eddigi állapot szerint, a robot útvonalában eltérések mutatkoznak a valósághoz képest, ami nem elhanyagolható. Legfőképp a fordulásnál tapasztalhatók problémák, így ott kezdem el vizsgálni, hogyan lehetne kompenzálni az eltérést. A 6.egyenlet adja meg, hogy a keréksebességekből hogyan lehet kiszámolni a

robot elfordulását, ahol az $\omega_{1\text{-}4}$ megadja az egyes kerekek szögsebességét $\frac{rad}{s}$ -ban, az r a kerék sugarát méterben, az L_1 a párhuzamosan lévő kerekek középpontjainak a távolságának a fele, és az L_2 pedig az egy tengelyen lévő első kerekek középpontjai és a hátsó tengelyen lévő kerekek közötti távolság fele.

$$\frac{\vartheta_{glob}}{\Delta t} = \frac{\frac{r}{4} * \omega_1}{L_1 + L_2} + \frac{\frac{r}{4} * \omega_2}{L_1 + L_2} - \frac{\frac{r}{4} * \omega_3}{L_1 + L_2} + \frac{\frac{r}{4} * \omega_4}{L_1 + L_2}$$

6.egyenlet Az AMR szögsebesség kiszámolása

További javulást lehet elérni egy korrekciós változó bevezetésével, ami sikerrel csökkenti a kerekek csúszásából eredő hibát. A hibát illetően további méréseket végeztem, annak a meghatározására, hogy a bejárt görbe mennyivel tér el az ideálistól, és ezt milyen mérőszámmal lehet leginkább jellemezni. A legmegbízhatóbbnak az bizonyult, ha vettem a két görbe alatti területet, és kivontam őket egymásból, így megkapva a 37.ábrán is látható, két görbe által közrefogott területet.



37.ábra A hibaszámításhoz szükséges görbék alatti terület szemléltetése

Az ábrán a pirossal jelölt terület meghatározása a cél, amelyhez a két területet integrálással határoztam meg, amihez még adott időközönként felvettem az aktuális pozícióját és orientációját a robotnak. A b_n és d_n az egyes mintákat jelentik, amiket a mérés során felvettem. Ezután a mérések különbségéből előállítottam a lineáris és rotációs hibákat, amelyet a 7.egyenlet alapján összegeztem.

$$y = \sqrt{\sum_{i=0}^{n} {x_n}^2}$$

7.egyenlet A hibaszámítás

Összesen tehát 4 hibakomponens volt: az X irányú, az Y irányú, a forgási és a görbék alatti területben jelentkező hibakomponens. A két irány béli és a forgási hiba összege ideálisan a terület béli eltérést eredményezi.

Miután ezt a hibát meghatároztam, az egyenletbe beépítettem oly módon, hogy az aktuális keréksebességektől függően adaptálódik a korrekciós változó (ε), folyamatosan javítva az eredményt. Látható, hogy a 6. egyenletben szereplő változók között sok a konstans, így ezeket átrendezve és a ϑ_{glob} helyére behelyettesítve a két szög (az odometria és a ground truth) mérésének különbségét, valamint hozzáadva az ε korrekciós változót a 8.egyenlet adódik.

$$\frac{\vartheta_{glob}}{\Delta t} + \varepsilon + rob_{const} = \omega_1 + \omega_2 + \omega_3 + \omega_4$$

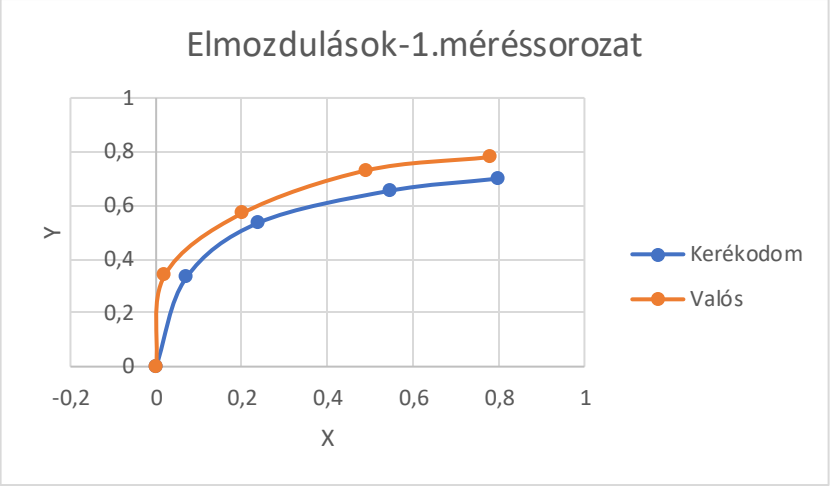
8.egyenlet Az algoritmussal kompenzált formula

A fent közölt egyenletben a kerekek szögsebességei ismeretlenek, de ugyanakkor mérhetők, tehát ezzel leredukálva az egyenletet egy egyismeretlenes egyenletté, megoldhatóvá válik a probléma. Az így elért kerékodometriával fuzionálva az IMU adatait, már egy jobb eredményt tudtam elérni, mint bármelyik előző módszerrel. Az elméletet pedig alább konkrét mérésekkel is alátámasztom.

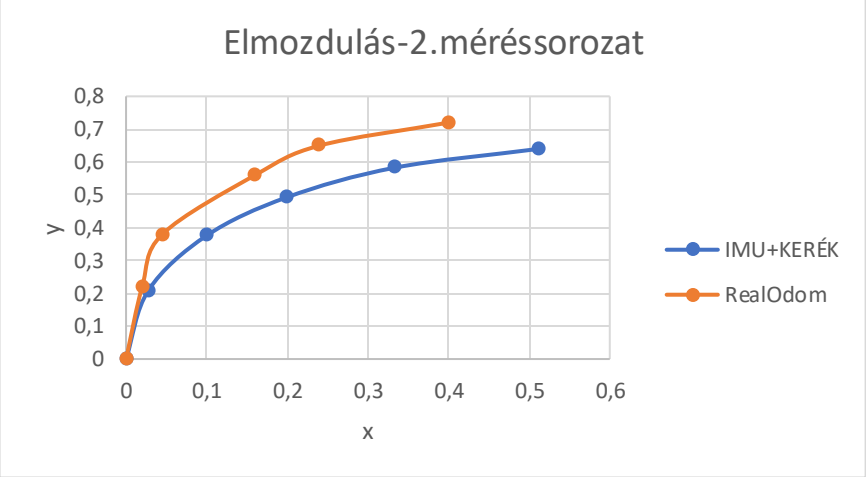
5.5 Az algoritmus kidolgozása

Alább az algoritmus kidolgozását részletezem, amely a végső megoldást biztosította. A méréseket az első esetben 1, a második esetben 2 másodperces periódusokban végeztem. Minden periódus elején és végén feljegyeztem a kerékodometria és az IMU által szolgáltatott adatokat, illetve a valós odometriához a robot mögé rögzítettem egy filctollat, ami a valós pályáját rajzolta a robotnak. Ezen a pályán bejelöltem a periódusonként megtett távolságot, illetve az orientációját a robotnak. Ezután a kezdő orientációhoz érintőt illesztve létrehoztam az adott orientációt használva egy derékszögű háromszöget, amelynek lemértem az oldalait, megkapva így az x-y elmozdulást. Majd a szöget pedig alapvető trigonometria alkalmazásával számítottam ki.

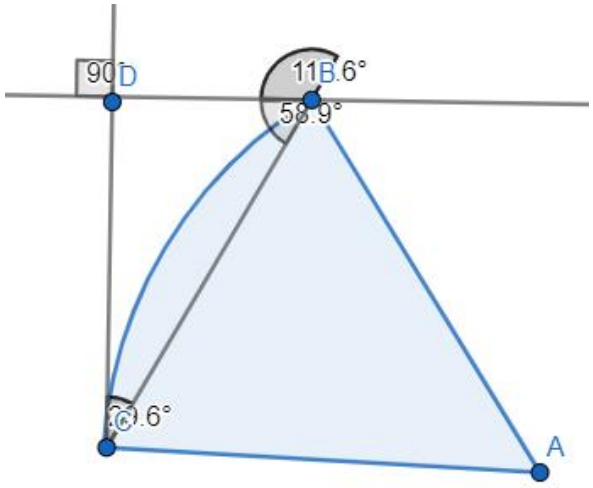
A lemért és a számított pozíció adatokat a 38. és a 39. ábrán mutatom be. A módszer könnyebb megértéséhez mellékeltem a 40.ábrán egy példát az alkalmazásra. Az ábrán a **C** pontból haladt a robot a **B** pontba. A mérések után az előzőekben is alkalmazott hibaszámítást alkalmaztam a hiba meghatározására. A következőkben pedig a diagramok és a szögelfordulás táblázatai láthatóak.



38.ábra Elmozdulás a hibaszámításhoz



39.ábra Elmozdulás a hibaszámításhoz



40.ábra A módszer vizualizálva

Az orientációkban található különbségeket az alábbi, 3. és 4. táblázatokban foglaltam össze.

Kerék	IMU+Kerék	Valós
21°	17°	18°
37°	36°	36°
51°	45°	46°
66°	56°	57°
84°	66°	67°

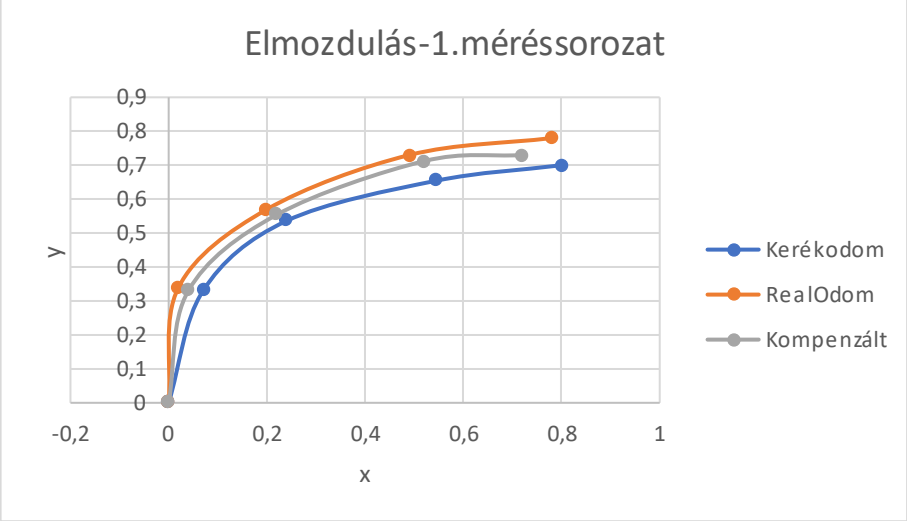
3.táblázat 1.mérés orientáció

Kerék	IMU+Kerék	Valós
32°	37°	38°
56°	55°	55°
86°	73°	71°
107°	115°	114°

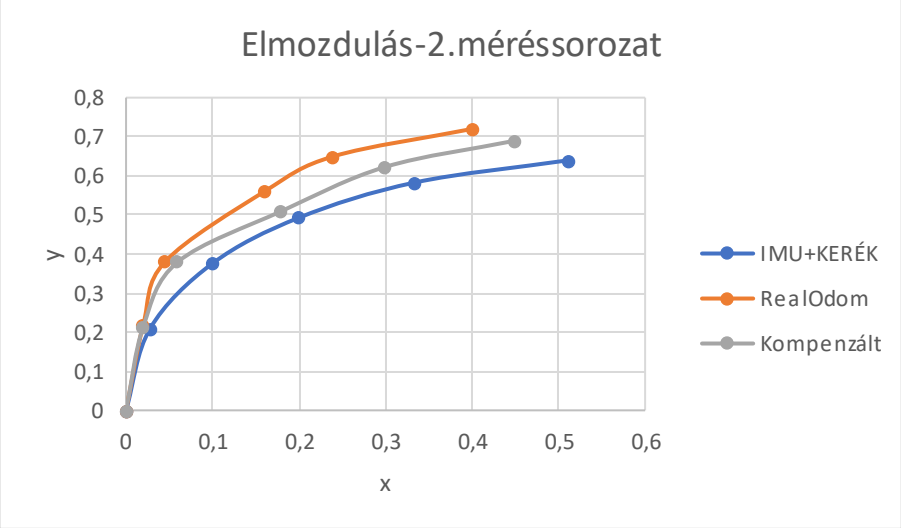
4.táblázat 2.mérés orientáció

Tehát a két területet kiszámoltam, majd kivontam egymásból, így megkapva a kettő közötti eltérést, vagyis a hibáját a rendszernek. Az első rendszernek 0,04770784 a második rendszernek 0,02972837 volt a hibája, a [12]. forrásban található módszerek segítségével kiszámolva. Mivel a hibák a rendszerre vonatkoznak és szorosan összefüggenek, ezért összegezhetőek is.

A 8.egyenletet így megoldva meghatároztam a korrekciós változót és a teszteket lefuttatva dokumentáltam az -immár kompenzált- eredményeket. A kódban az alaputasítás feldolgozásánál módosítom a keréksebességeket a korrekciós változóval, így a lehető legkevesebb torzuláson megy keresztül. Az így elért elmozdulás az előzőkhez viszonyítva a 41. és 42. ábrákon látható. A teszt során ugyanazt a parancsot adtam ki, mint az előző két esetben, így jól összehasonlíthatóak az eredmények. A hiba minél pontosabb meghatározásához az eredményeket az idő szerint szükséges normalizálni, mivel minél nagyobb időintervallumot tekintünk, annál széttartóbb lesz a két görbe, tehát annál nagyobb lesz a hiba is.



41.ábra A kompenzált odometria-1.mérés



42.ábra A kompenzált odometria-2.mérés

Ahogy a diagramokon jól látszik, a kompenzálás sikeres volt, a kerék csúszásból eredő hibát átlagosan 53%-al csökkentettem. A maradék hiba a mérési pontatlanságból, az esetleges egyszerűsítésekből illetve a kerekítésekből adódik.

6 Eredmények

Ebben a fejezetben összefoglalnám az általam elért eredményeket, azaz a kontribúciót, ami indokolja a dolgozatom létrejöttét. A dolgozat célja egy olyan mecanum kerekes rendszer létrehozása volt, amely szenzorfúzió segítségével állítja elő az odometriáját. Az odometriát előállítottam pusztán kerékodometriából, IMU-ból, a kettő fúziójából és végül az általam kidolgozott algoritmus segítségével. Ahogyan a mérési eredmények ezt alátámasztják, haladó és forgó mozgás esetén az általam előállított módszer bizonyult a legpontosabbnak a 3 megközelítés közül. Ezáltal kijelenthetem, hogy sikerrel létrehoztam egy olyan algoritmust, amely megfelelően kompenzálja a kerékcsúszást egy 4-kerekű mecanum kerekes robot esetében, ahol szenzorfúziót alkalmazunk.

7 Következtetések és fejlesztési lehetőségek

A dolgozat végére többféle konklúziót lehet levonni az ilyen rendszerekkel való munka mivoltáról. Legfontosabb tanulságként az szolgált, hogy a mecanum kerekek odometriáját jóval körülményesebben lehet számítani, mint a hagyományos kerekekét, és pusztán az enkóderek kimenetére hagyatkozva komplex mozgások esetén, az odometria lényeges eltér a valósághoz képest. Ennek kapcsán fontos észrevétel, hogy az IMU önmagában pontatlan és teljesen alkalmatlan ilyen típusú mérésekre, viszont remekül lehet vele szenzorfúziót véghez vinni. A kutatás további kérdéseket vetett fel, amit még érdemes lenne megvizsgálni. Ilyen például többféle IMU szenzor együttes alkalmazásának számszerű előnyei, algoritmus a minél jobb eredmény elérésére és annak megvizsgálása, hogy hány szenzorig költséghatékony az alkalmazásuk. További kutatási lehetőség, ha a LIDAR kimenetét is fuzionálom a rendszerhez, amellyel elméletben szintén nagymértékben javítható az odometria.

8 Köszönetnyilvánítás

Legfőképp szeretném megköszönni a konzulensemnek, *Dr. Fehér Gábornak* a határtalan segítségét, mivel Nélküle és a meglátásai nélkül nem valósulhatott volna meg a dolgozat.

Továbbá *családtagjaimnak, lakótársaimnak* és a laborban munkájukat végző *hallgatótársaimnak*, akik közvetlenül vagy közvetve, de mind elősegítették a dolgozat létrejöttét.

Végül, de nem utolsósorban pedig a *HEBI munkatársainak*, akik fáradtságot nem kímélve válaszolták meg az összes, hardverrel kapcsolatos kérdésemet.

9 Irodalomjegyzék

[1] **Yunwang Li, Sumei Dai, Yuwei Zheng, Feng Tian, Xucong Yan, "Modeling and Kinematics Simulation of a Mecanum Wheel Platform in RecurDyn", Journal of Robotics, vol. 2018, Article ID 9373580, 7 pages, 2018. https://doi.org/10.1155/2018/9373580**

[2] **https://seamonsters-2605.github.io/archive/mecanum/#:~:text=The%20front%2Dright%20and%20back,x%2B1%2F4%CF%80 (Megtekintve: 2020 szeptember 8.)**

[3] **J. Shen, D. Tick and N. Gans, "Localization through fusion of discrete and continuous epipolar geometry with wheel and IMU odometry," Proceedings of the 2011 American Control Conference, San Francisco, CA, 2011, pp. 1292-1298, doi: 10.1109/ACC.2011.5990946**

[4] **Song, Y.; Nuske, S.; Scherer, S. A Multi-Sensor Fusion MAV State Estimation from Long-Range Stereo, IMU, GPS and Barometric Sensors. Sensors 2017, 17, 11.**

[5] **Xue, H.; Fu, H.; Dai, B. IMU-Aided High-Frequency Lidar Odometry for Autonomous Driving. Appl. Sci. 2019, 9, 1506.**

[6] **R. K. Megalingam, D. Nagalla, K. Nigam, V. Gontu and P. K. Allada, "PID based locomotion of multi-terrain robot using ROS platform," 2020 Fourth International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2020, pp. 751-755, doi: 10.1109/ICISC47916.2020.9171152.**

[7] **https://stackoverflow.com/questions/6911900/android-remove-gravity-from-accelerometer-readings (Aug 2 '11 13:38 Kay felhasználó által, 2020.09.03 állapot)**

[8] **http://www.varesano.net/blog/fabio/simple-gravity-compensation-9-dom-imus (2020 szept. állapot)**

[9] **Wikipédia: Kvaterniók https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation#Using_quaternion_rotations (revision 08:01, 20 September 2020)**

[10] **https://bme.videotorium.hu/hu/category/1083/szamitogepes-grafika**

[11] **http://docs.ros.org/en/melodic/api/robot_localization/html/index.html (2020 szept.)**

[12] **http://foundation04.chem.elte.hu/Specik/(1)/Mereselmelet_merestechnika_2_resz_Hibaszamitas.pdf**

[13] **http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom (Megtekintve: 2020. szeptember 4.)**

[14] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html (Megtekintve: 2020.szeptember 14.)

[15] https://en.wikipedia.org/wiki/Butterworth_filter (revision 23:51, 8 March 2020)

[16] https://docs.hebi.us/docs/python/0.95/ (Megtekintve: 2020.szeptember 12.)

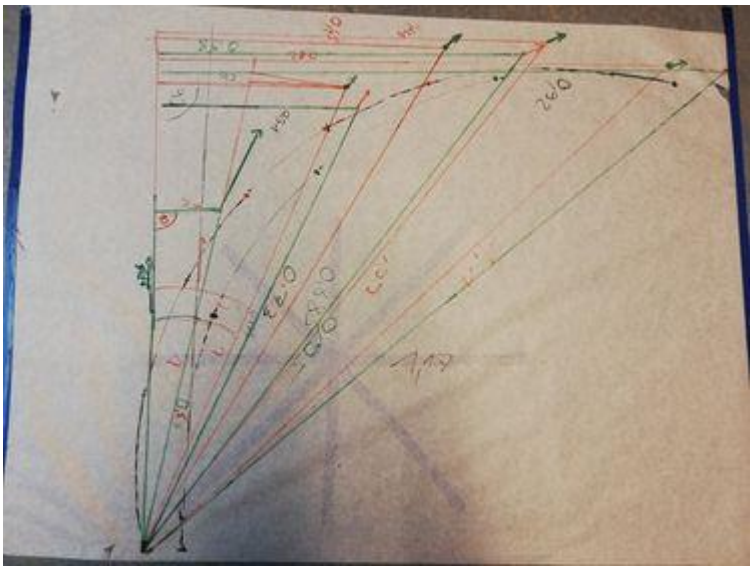
[17] R.E. Kalman (1960). "Contributions to the theory of optimal control". Bol. Soc. Mat. Mexicana: 102–119. CiteSeerX 10.1.1.26.4070

[18] Julier, S.J.; Uhlmann, J.K. (2004). "Unscented filtering and nonlinear estimation". Proceedings of the IEEE. 92 (3): 401–422. doi:10.1109/jproc.2003.823141

10 Függelék

Ebben a fejezetben mellékelem azokat a dolgokat, amelyeknek úgy vélem volt szerepe a kutatás sikerességében, de szigorúan nem kötődik a dolgozathoz.

A mérési környezet:



1.sz. melléklet