SZAKDOLGOZAT

Molnár Tamás

Debrecen,
2009

Debreceni Egyetem
Műszaki Kar
Villamosmérnöki és Mechatronikai Tanszék

LEGO MINDSTORMS NXT 2.0 PROGRAMOZÁSA JAVA NYELVEN

Készítette:

Molnár Tamás
mérnökinformatikus hallgató

Témavezető:
Dr. Husi Géza,
Tanszékvezető főiskolai docens

Külső konzulens:
Lugosi Péter,
Tesztmérnök, National Instruments

Debrecen,
2009

TARTALOMJEGYZÉK

TÉMAVÁLASZTÁS INDOKLÁSA

Harmadéves tanulmányaim során nyílt lehetőségem a LEGO robotok közelebbi megismerkedésével, és ez után döntöttem amellett, hogy szeretném a szakdolgozatomat is ebben a témakörben megírni. Valamint az is indokolta ezen irányú témaválasztásomat, hogy életemet egyre nagyobb részben befolyásolják a robotok jelenléte. Az oktatási intézmények is egyre jobban szorgalmazzák, hogy a fiatalok minél korábban megismerkedhessenek a robotok világával. Ami a Debreceni Egyetemen sincs másképp.

Szakdolgozatom célkitűzésének egy, az iparban is használt anyagmozgatási folyamatot tekintetek. Melynek modellezését a LEGO Mindstorms NXT készletekkel próbálom megvalósítani.

Előzetes információim alapján úgy gondolom, ha valaki robotok programozását szeretné közelebbről megismerni, és szeretne ezzel foglalkozni, annak ideális választás a LEGO Mindstorms NXT készlete. Hogy én is így fogok-e vélekedni arról a dolgozatom végén, a megszerzett tapasztalataimból kiindulva ki fog derülni.

ROBOTOKRÓL RÖVIDEN

ROBOTOK FOGALMA

A robot szó a szláv „rabota” kifejezésből ered, aminek jelentése szó szerint szolgamunka. Magát a robot szót Karel Čapek nevéhez köthetjük, ugyanis ő használta először ezt, amit 1921-ben mondott ki, egyik drámájában, melynek címe Rossum Univerzális Robotjai [1.]. Ebben a darabban a gépek fellázadnak az emberek ellen, és átveszik a hatalmat a Föld felett. Mára már ez a szó nemzetközivé vált. A hétköznapi szemlélet szerint a robot, az emberre hasonlító, szellemi vagy fizikai munkát végző szerkezet. Azonban egy szerkezetről eldönteni, hogy robot-e, azt nem a megjelenéséről kell megítélnünk, hanem sokkal inkább, hogy milyen képességekkel rendelkezik. Ilyen képesség lehet például az, hogy tud-e reagálni a környezeti változásokra, ingerekre vagy, hogy éppen mennyi szabadsági fokkal rendelkezik. Ezen okok miatt egy gépjárművet nem tekinthetünk önmagában robotnak, de ha a végcélt megadjuk neki és képes lesz emberi beavatkozás nélkül eljutni oda, így ezek után joggal tekinthetünk rá, mint olyan szerkezetre, amelyet nem csak a köznyelv nevez robotnak.

A robot fogalmára nagyon sok megfogalmazást adtak különféle szervezetek más-más szemszögből vizsgálva azt. Íme, ezekből néhány definíció:

- 1979-ben a Robot Institute of America azt mondta ki, hogy a robot definíciója nem más, mint egy „Újraprogramozható, többfunkciós manipulátor anyagok, eszközök, részegységek mozgatására, megváltoztatására programozott mozdulatsor segítségével különféle feladatok elvégzése” [2.].
- A Webster szótár azt mondja, hogy a robot „egy önműködő eszköz, mely ember formájú vagy olyan feladatokat hajt végre, amelyeket emberinek nevezünk” [3.].
- Végül a Robotics FAQ, úgy véli, hogy a robot az „erő és intelligencia együttese, ahol a mesterséges intelligencia találkozik a valós világgal” [4.].

Az előbb említett definíciókban vannak közös tényezők, amelyek az alábbiak:

- A szerkezet legyen képes hely vagy helyzetváltozásra. Azaz mozgása során szabadsági fokkal kell, hogy rendelkezzen a szerkezet.
- Megadott program alapján képes legyen önállóan egy tevékenységet részben vagy teljesen irányítani.

Ezen dolgok ismeretében a műszaki terület művelői között van egy általános érvényű definíció, ami azt mondja ki, hogy „a robot egy olyan gép, amely szállítási és manipulációs feladatok automatikus végrehajtására szolgál, és melynek működése programozható”.

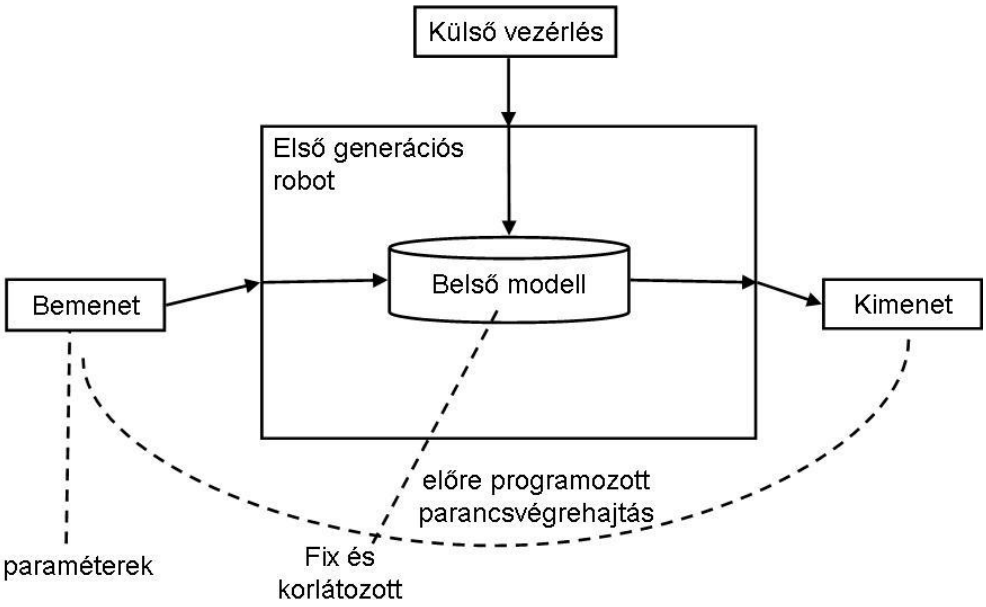
A ROBOTIKA HÁROM ALAPTÖRVÉNYE

A robotika három alaptörvényének a megfogalmazása Isaac Asimov nevéhez kötődik. Az „Én a robot” című novellásgyűjteményében fogalmazta meg ezeket a törvényeket, amelyek így szólnak [5.] :

- Első törvény: „a robot nem árthat az embernek, és nem nézheti tétlenül, ha az embert veszély fenyegeti.”
- Második törvény: „a robot engedelmeskedni tartozik az emberek parancsainak, kivéve, ha ezek a parancsok az Első Törvénybe ütköznek.”
- Harmadik törvény: „a robot köteles megvédeni magát mindaddig, amíg ez nem ütközik az Első vagy a Második Törvénybe.”

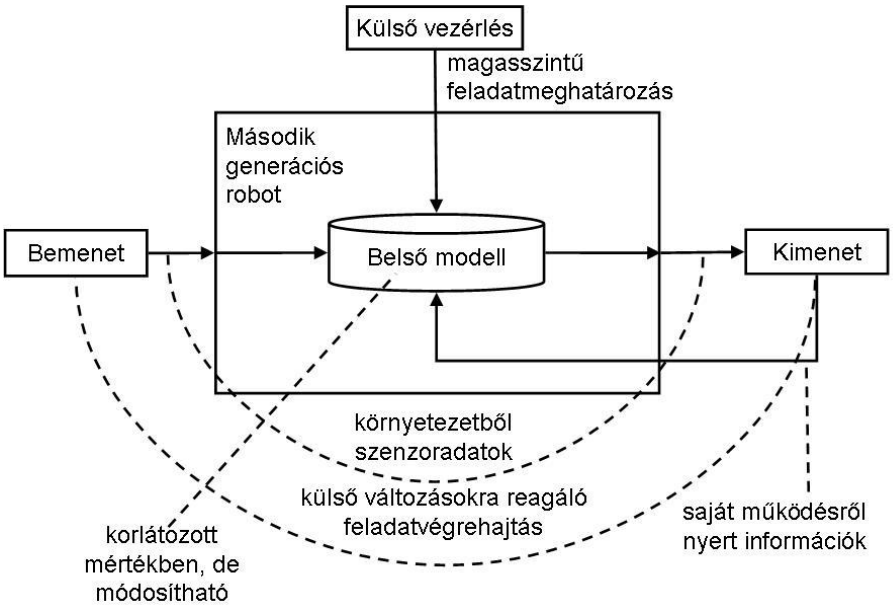
ROBOTGENERÁCIÓK

A robotok generációja három nagyobb csoportra különülnek el egymástól. Az első generációs robotokat (1. ábra), az 1960-as években hozták létre, és főképp anyagmozgatásra használták. Kezdetekkor nem rendelkeztek különféle érzékelőkkel, ugyanis főképp kötött pályán mozogtak, és futószalagok kiszolgálására voltak optimalizálva. Nem sokkal később az ipari munka világában is használhatóvá váltak a robotok. Itt főképp már az ember egészségére ártalmas környezetben működtek.



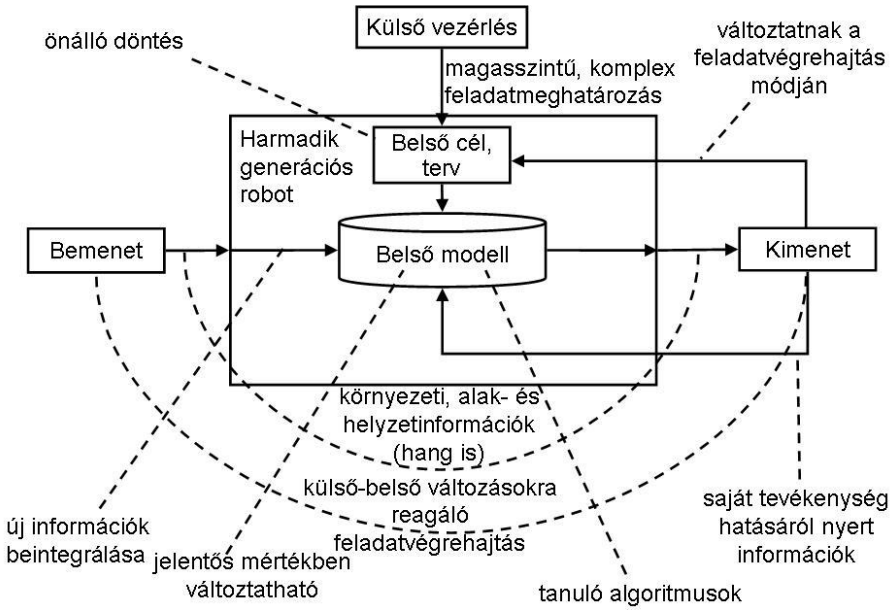
1. ábra: Az első generációs robotok modellje (forrás: [6.]).

Az iparban nagyon jól tudták hasznosítani ezeket a szerkezeteket, így a második generációk is ugyan csak ezen a területen jelentek meg (2. ábra). De a fő alkalmazási terület a mikroelektronika lett. Az 1970-es évekre szerelték fel a robotokat először érzékelőkkel, amelyek vizsgálták a környezetet és a környezeti változások hatására képessé váltak arra, hogy tevékenységüket megváltoztassák.



2. ábra: A második generációs robotok modellje (forrás [7.]).

A harmadik generáció a mai korunkat jellemzi főképp, és ezek kutatása folyik jelenleg is (3. ábra). Mára már jellemzően a jelfeldolgozás és az információ kiválasztása, valamint ezek kombinációinak a variálása a feladatuk. Ekkor jelentek meg azon szerkezetek, amelyek mesterséges intelligenciával lettek felruházva, és amelyekben önálló viselkedésű algoritmusok, és döntési rendszerek vannak integrálva. Ezen döntési rendszerek tették lehetővé, hogy az emberi gondolkodáshoz lehessen hasonlítani a szerkezeteket.



3. ábra: A harmadik generációs robotok modellje (forrás: [8.]).

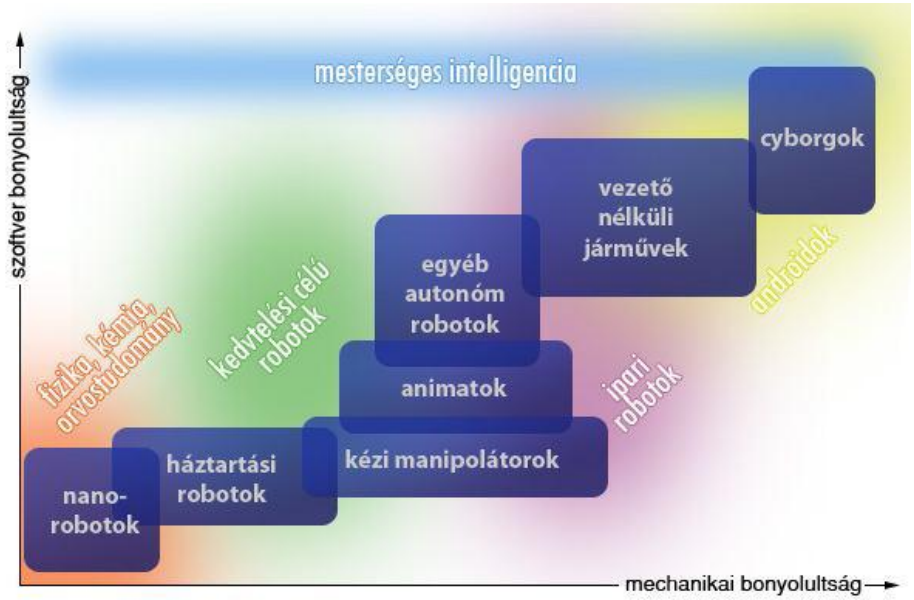
ROBOTOK OSZTÁLYZÁSA

Osztályozni ezen szerkezeteket egyáltalán nem egyszerű feladat, ugyanis nem könnyű eldönteni, hogy mit is vizsgálnunk meg, például fizikai megjelenést, vagy esetleg képesség szerint csoportosítunk-e, és így tovább. Ahhoz, hogy egy robotot be lehessen sorolni bármilyen csoportba, meg kell vizsgálni először is, hogy milyen intelligencia szinttel rendelkezik, másodszor milyen a külső megjelenése, azaz például el kell tudni dönteni, hogy a tárgyalt eszköz mobil robot, technológiai robot, vagy esetleg robotkar. Harmadszor figyelembe kell venni a szerkezet pályavezetését, és végül pedig nem szabad megfeledkezni az alkalmazási területének a besorolásáról sem.

A mobil robotokat tovább szokás bontani alcsoportokba. Ilyenek az androidok, animátorok, ember nélküli járművek, szórakoztató robotok, az általános autonóm robotoknak külön-külön alcsoportot alkotnak.

Az önálló helyzetváltoztatásra nem képes robotok, azaz a statikus robotok is nagyon hasznosak tudnak lenni, ebbe a csoportba tartoznak a háztartási és ipari robotok, robotkarok.

A hagyományos robotszemléletet nagyon megváltoztatta a nanorobotok megjelenése. A fizika és a kémia határterületén megszületett „gépek” nem csak méretükben, hanem működésükben is jelentősen eltérnek a hétköznapi szemléletben kialakult robotképtől.



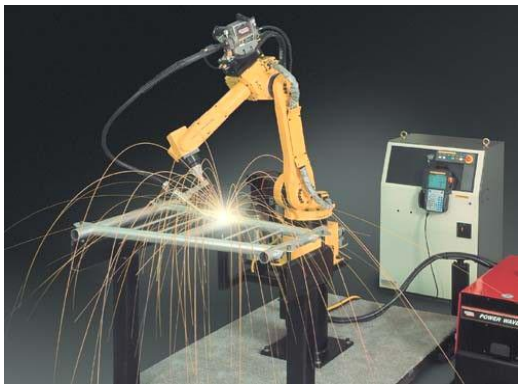
4. ábra: Robotok besorolása felépítési és működési bonyolultságuk alapján (forrás [9.]).

ROBOTOK FELADATAI

- Veszélyes feladatok: ide kerülnek az űrkutatás, vegyi anyagok, bombafelderítés, katasztrófa elhárítással foglalkozó robotok.
- Unalmas és/vagy ismétlődő feladatok: azonos műveletek, állandó mozgások, gyártási válogatást végző robotok.
- Magas pontosság vagy nagy gyorsaság feladatok: elektronikatesztelő, sebészet, precíziós megmunkáló robotok.

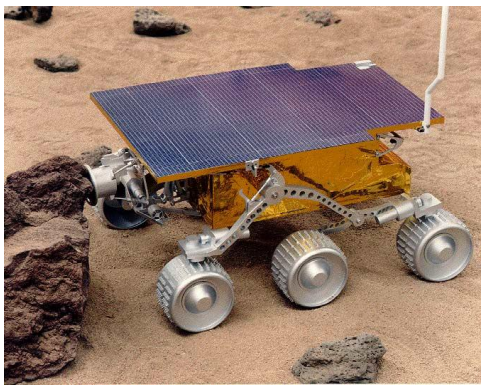
A ROBOTOK ALKALMAZÁSUK SZERINTI CSOPORTOSÍTÁSA

- Az iparban használt robotok: ezekhez tartoznak a technológiai feladatot ellátó robotok, az anyagmozgató robotok – szerelőrobotok, hegesztőrobotok, festőrobotok (5. ábra). Emberi mozgásformákkal végrehajtható vagy ahhoz hasonló feladatok elvégzésére alkalmas gépek. Az ipari robotok mechatronikai szerkezetek, mely nyílt kinematikai láncú mechanizmust és intelligens vezérlést tartalmaz. Előírt autonóm irányított programozható feladatok elvégzésére képesek.



5. ábra: Robotkar (forrás [10.]).

- A kutatásban használt robotok: három nagyobb csoportot tudunk elkülöníteni a kutatásban használatos robotok között az első az általános mobil robotok. Melyek kerekes, lépegető, önállóan tájékozódni tudó robotokat jelenti. Valamint a telerobotok is ide tartoznak, amelyeket például a mikro sebészetben alkalmaznak. A második az animátorok, azaz az állatokhoz hasonlító robotok, leginkább a mozgásuk leutánzása jelenti a fő kihívást. És végül a harmadik nagy alcsoportot az androidok alkotják.



6. ábra: Mars űrjármű: Pathfinder robot (forrás [9.]).

- A speciális feladatok megoldására alkalmazott robotok lehetnek: mikro-robotok, nanorobotok és a gyógyászatban alkalmazott robotok.
- Háztartási robotok: ehhez a csoporthoz tartoznak az automatikusan és intelligensen működő háztartási gépek, a lakás egyes egységeit irányító rendszer, amely reagálni képes a környezeti változásokra, és a számítógépes vagy egyéb hálózaton keresztül kapcsolatot tartó berendezések, például vagyonbiztonsági rendszerek.

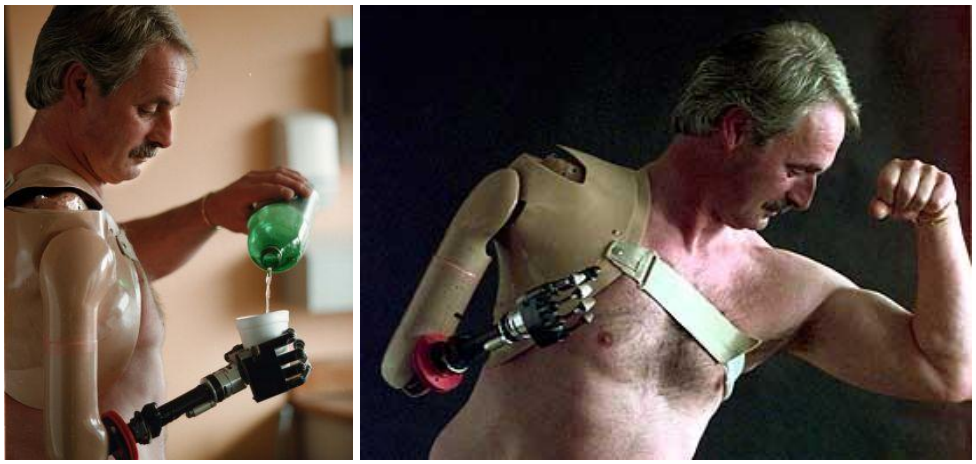
7. ábra: Mobil Robot: Aibo, a Sony cég új szórakoztatóelektronikai terméke (forrás [9.]).

8. ábra: ASIMO a Honda bérelhető, már sorozatban is gyártott androidja (forrás [9.]).

CYBORGOK

A cyborg megnevezés azokra a részben ember, részben pedig gép képződményekre vonatkozik, amelyek már nem tartoznak a robotokhoz (9. ábra). De nagyon nehéz eldönteni, hogy milyen az a keverék, ami még ember, vagy már robot. Az orvostudomány mára már képessé vált, hogy újra működésre bírja a sérült idegpályákat. Elérték, hogy az ember az

idegpályára kötött mesterséges testrésszel képes legyen befolyásolni a működését. Gondoljunk csak egy olyan balesetre, ahol egy személy elveszítette valamely végtagját. Ezen emberek számára újra „élhető” életet tudnak nyújtani. Ebből következik, hogy a legjelentősebb eredményeket a művégtagok pótlásának területén érték el. Amit előrejelzések alapján várhatóan 2020–2030 körül a fejlett országokban általánosan elérhetőek lesznek a művégtagok.



9. ábra: Teljes kar pótlása művégtag segítségével (forrás [11.]).

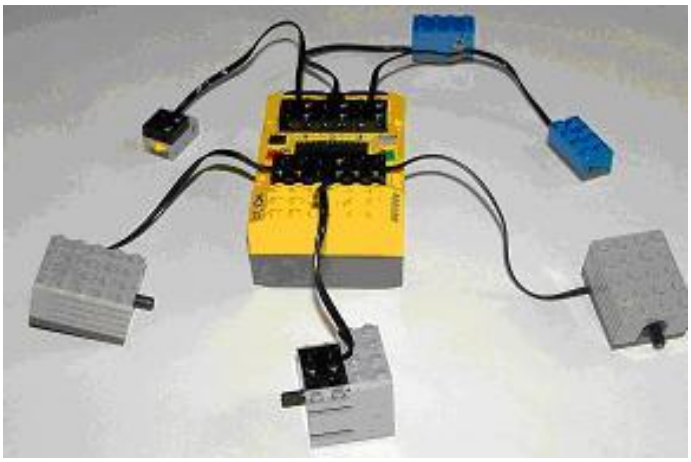
LEGO ROBOTOK

A robotika hazánkban is elérhető eszközei közül az egyik legnépszerűbb, legsokoldalúbban használható a LEGO cég által kifejlesztett egységei. Amelyeket kezdetben oktatási célokra készítettek, de ma már egyéb helyeken is használnak, ilyen például a játék világa. A LEGO cég több csomagot is kifejlesztett az évek során, ezek közül az oktatásban leginkább hasznosíthatóak az RCX, valamint az NXT csomagok. Szakdolgozatom készítésekor NXT készleteket használtam föl.

A LEGO 1998-ban indította útjára az úgynevezett NXT intelligens tégla projektet, aminek elődje az RCX volt, majd ezt követte a már említett Mindstorms NXT, míg végül a legújabb generáció a 2009-es évben érkezett, amit egyszerűen NXT 2.0-nek kereszteltek el. A kezdetekkor az előbb már említett RCX programozható egység volt, melyet 1987-ben egy kutatási project keretében Prof. Seymour Papertnek munkálatainak köszönhető. Az RCX egység hat szenzor értékeit tudja feldolgozni és négy motor segítségével képes különféle mozgásokat végezni.

LEGO RCX

Hazánkban, a játékboltokban nagyon könnyen hozzájutható eszközről van szó, közel 700 elemet tartalmaz a készlet (10. ábra).



10. ábra: Az RCX robot és főbb tartozékai (forrás [9.]).

Az RCX téglához egyszerre alaphelyzetben 3 motor és 3 érzékelő csatlakoztatható. A nyomás és a fényérzékelő a készlet része, de beszerezhető forgás, hő és időérzékelő is. Ezen nagyszámú érzékelők miatt a fizikai kísérletek és vizsgálatok során is jól használható a készlet.

A 9 V-os motorok igen fürgék, viszonylag nagy nyomatékúak. Az alapkészlet csak kettő motort tartalmaz, amihez beszerezhető több típusban is különböző motor, amint az a képen is látható (10. ábra).

LEGO NXT

A LEGO Mindstorms mára már annyira közkedvelté vált, hogy nem csak egyetemeken, főiskolákon tudják hasznosítani, hanem egyre inkább jellemző, hogy középiskolák, valamint általános iskolák is beszerzik és oktatnak ezekkel a csomagokkal. Igaz ez inkább Magyarországtól nyugatabbra lévő országokra jellemző. Hazánkban számottevően egyelőre csak a felsőoktatásban van jelen a Mindstorms NXT (11. ábra).

11. ábra: A legújabb NXT csomag (forrás [12.]).

Az NXT alapú robot elemek több kiszerelésben is kapható, sajnos hazánkban elég körülményes ezeknek a beszerzése jelen pillanatban. A 8527-es sorszámú készlet alkalmas az egyéni munkára. A 9797-es jelzésű készlet már kimondottan iskolai munkálatokra lett kialakítva, az FLL verseny elvárásainak megfelelően [13.]. Ez a készlet a 9648-as jelzésűvel kiegészítve már igazi oktatási készletet jelent.

Az alapcsomaghoz viszonylag alacsony áron lehet hozzájutni. Magyarországon, százezer forint alatt, így sokkal olcsóbban lehet beszerezni ezen készleteket, mint egyes manipulátorokat, robotkarokat. A műanyag borításának köszönhetően sokkal kevésbé érzékeny a külső behatásokra. Ugyanakkor a rendszer kellőképpen összetett ahhoz, hogy ne csak szórakozásra lehessen használni, hanem komoly kihívások elé is állítsák a vállalkozó kedvű fejlesztőket. Azért különösen jó ez a LEGO csomag, mert egyszerre enged bepillantást mind az informatika, mind a műszaki világ rejtelmeibe, problémáiba. Valamint a rendszer lehetővé teszi, hogy az ügyes robotépítők rövid idő leforgása alatt képesek legyenek felépíteni és beprogramozni egy robotot.

A rendelkezésre álló eszközök között szenzorok széles köre válik elérhetővé, ennek köszönhetően az ezekkel épített robotok érzékelik a fényt, a hangokat, a mozgásokat, és a kapcsolásokat. Hivatalosan a kapcsoló érzékelőt, a LEGO Touch Sensor-nak (érintő szenzornak) nevezi, melyre későbbiekben én is így, vagy nyomásérzékelőként fogok hivatkozni. A rendszer szíve egy mikroprocesszor, amelyet PC-ről lehet programozni. A

kommunikáció USB csatolón, vagy vezeték nélküli Bluetooth kapcsolaton keresztül lehetséges, amely lehetővé teszi számunkra, hogy a távolból irányítsuk, vagy beavatkozzunk robotunk cselekmény sorozatába. A programírás, és a program feltöltés után a robot saját életet él, a számítógéptől függetlenül.

12. ábra: Az NXT robot és főbb tartozékai (forrás [12.])

A standard LEGO Mindstorms NXT 2.0 készletben találunk egy NXT 2.0 intelligens téglát, amelyet egy 32 bites mikroprocesszorral láttak el. 256 Kbyte FLASH és 64 Kbyte RAM memóriával rendelkezik. A téglának négy bemeneti és három kimeneti portja van, és egy LCD mátrix kijelzővel rendelkezik. Valamint kapunk még a csomaggal három szervomotort, két érintő szenzort, egy színérzékelőt és egy távolság mérésére alkalmas szenzort. Kommunikáció USB vagy Bluetooth használatával lehetséges.

KÉSZLETEK

Jelen pillanatban több lehetőség van az alkatrészek beszerzésére. Az NXT tégla, a motorok és a különböző típusú érzékelők a LEGO honlapjáról vagy nagyobb web áruházakból, mint például a LUGNET, elemenként is megrendelhetőek, de kifizetődőbb, ha készletben vásárolunk.

8527-es játékbolti csomag, ez kapható több nyelvi verzióban is, de a magyar kereskedelemben csak a kézikönyvet kapjuk meg lefordítva. Az 8527-es jelzésű készlet - amely a 2007-es évben a NextWork verseny alapcsomagja is volt [14.]-, igazából a játékboltok számára, otthoni használatra szánt, összeállított készlet. A LEGO cég nem rég

piacra dobta a 9797-es jelzésű oktatási verziót is. Ez az USA-ban és már világszerte is igen népszerű FLL verseny szabályainak megfelelően tartalmazza az érzékelőket, és a csomag része a tölthető akkumulátor is, valamint az RCX-hez kifejlesztett érzékelők csatlakoztatásához szükséges átalakító kábeleket is.

Az NXT készlet tartozékai – Szenzorok bemutatása

NXT egység (Brick)

Az alábbi képen az NXT készlet intelligens központi vezérlő része van, amit „téglának” is szokás nevezni. Külön érdekesség, hogy Java alatt egy virtuális operációs rendszer fog futni az NXT egységen, de ennek pontos megismerése a későbbiekben lesz kifejtve. Továbbá az NXT egységen található egy grafikus LCD kijelző, mely képes szöveg, számértékek, és rajz objektumok megjelenítésére. Ezekre, a műveletekre 8 sor áll rendelkezésre, amelyekben soronként 16 karaktert lehet egymás mellett megjeleníteni. Tehát 100 x 64 pixelnyi az a terület, amely a megjelenítésre szolgál az LCD kijelzőn. A sorok egymást követően a bal felső sarokból kezdve kell értelmezni.

Érzékelők (Sensors)

A szenzor egy olyan eszköz, amely fizikai mennyiséget (távolság, hőmérséklet) a vezérlés- és szabályozástechnikában jobban felhasználható, jobban kiértékelhető jellé alakít át.

Az NXT készletben látszólag négy különböző érzékelő típus található meg, de valójában öt van. A nyomásérzékelő mellett fényérzékelő, hangerősség, távolság érzékelő, és az említett ötödik pedig a motor, más néven szögelfordulás érzékelő, mivel a motor képes forgásérzékelésre is. Így egy időben hét szenzor használata válik elérhetővé.

Az előbb említett érzékelők bemutatása, jellemzése Java programozási nyelv alatt:

KAPCSOLÁSI ÉRZÉKELŐ (TOUCH SENSOR)

Ez a szenzor három állapotot tud egymástól megkülönböztetni és figyelni. Ez a három állapot figyelés a benyomódás, elengedés, illetve a klikkellés. Ha jellemezni kellene ezeket az állapotváltozásokat, azt lehetne mondani, hogy a benyomás megfelel a logikai 1-es értéknek. Az elengedés lesz a logikai 0, és végül a váltás pedig a 0-ról 1-re való átváltás. A benyomáshoz szükséges kifejtett erőnek meg kell haladnia a 0,33 N erőt. De erőkar használatával ez az érték természetesen csökkenthető. Egy elég fontos tényező, hogy ennél az érzékelőnél végállásig kell nyomni ahhoz, hogy jelet észleljen.

FÉNYÉRZÉKELŐ ÉS LÁMPA (LIGHT SENSOR)

Ez a szenzor fel lett szerelve egy lámpával is, ami azt a célt szolgálja, hogy pontosabb mérést tudjon elérni a szenzor. Ugyanis így kevésbé méri a környezetből adódó fényeket. Természetesen lehetőség van a lámpa kikapcsolására, de az előbb említett ok miatt ajánlott bekapcsolt állapotban használni a fényszenzorral. A fényszenzor a mért egész értéket egy 0-tól – 100-ig terjedő skálán adja vissza. Nagyon érdekes dolog, hogy a segédfény bekapcsolásával a fényérzékelőt használhatjuk távolság érzékelésére is, habár ez egyáltalán nem ajánlatos. Ugyanis a mérési eredményeket számos tényező befolyásolhatja ilyenek például a pontos színe az objektumnak, a szín homogenitása, valamint a környezet megvilágítása.

HANGERŐSSÉG ÉRZÉKELŐ (SOUND SENSOR)

Az érzékelő hangfrekvenciát nem tud mérni, csak is erősséget mér, ami az előző szenzorhoz képest szintén egy 0-tól – 100-ig terjedő skálára vetíti vissza. Oda kell figyelni arra, hogy a hangerősség mérése erősen függ attól, hogy milyen távol van az objektumtól, aminek a jelszintjét mérni szeretnénk. A szenzor használata előtt, le kell mérni a környezetből adódó jelszintet, és e fölé kell helyezni azt az értéket, amelyet mérni szeretnénk. Beszédfelismerés esetben, az utasítás időbeli erősségének a lefolyását kell vizsgálni.

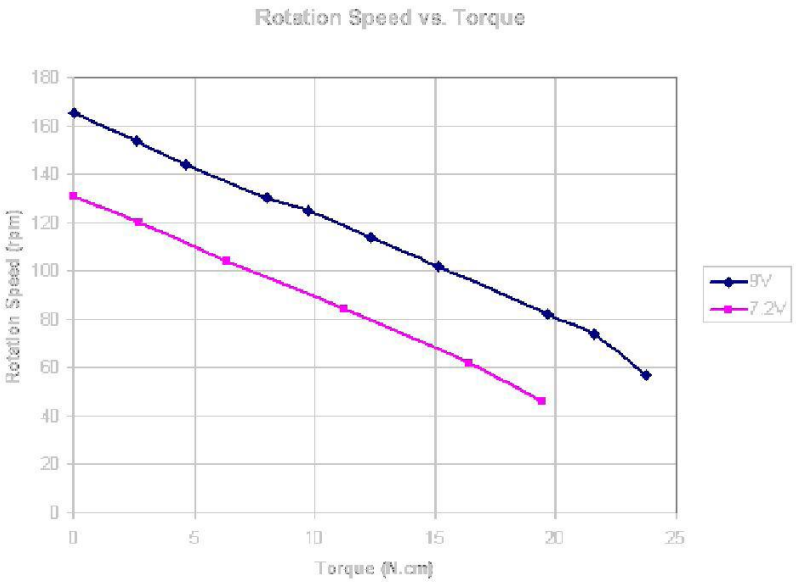
TÁVOLSÁG ÉRZÉKELŐ (ULTRASONIC SENSOR)

Ez a szenzor valójában 5 - 255 cm tartományban képes a távolság meghatározására, nagyjából 2-3 %-os tévedési aránnyal. Nagy hátránya ennek az érzékelőnek, hogy csak akkor mér pontos értéket, ha megfelelő mennyiségű ultrahang jel verődik vissza a felületről, így gyakran előfordul a pontatlanság. Egy lehetséges alkalmazási területe például a mozgó robotoknál az ütközések elkerülése.

SZÖGELFORDULÁS ÉRZÉKELŐ (SERVOMOTOR)

A motorokban beépített forgásérzékelő van, amely egy fok pontossággal képes elfordulást mérni. Így finom mozgás elérésére is képes. Az NXT a szervomotorok segítségével tud bármilyen irányú mozgást elvégezni, ilyenek például: emelés, járás, vagy gurulás. A motor egyben forgásérzékelő is, emiatt mind kimenet, mind bemenet funkciója is van. A motor alapvetően két fajta üzemmódban működhet, szabályozás nélkül és szabályozottan. Bármilyen robot helyváltoztatásához legalább egy motor szükségeltetik.

Ami a LEGO NXT motorjának belső szerkezetét illeti, a motor tartalmaz egy biztonsági elemet, amely megakadályozza, hogy a motor ne égjen le, ha éppen nem forog. Motorok sebességét külső áttételekkel lehet megváltoztatni a ráadott feszültség mellett. Minél jobban felgyorsítjuk a forgási sebességet, annál kisebb lesz a kapott forgatónyomaték, azaz egyszerűen megfogalmazva erőtlenebb a motor. A szervomotor a legnagyobb erőt 6 darab 1,5 V-os egyszer használható elemek esetében képes elérni. Tölthető ceruza akkumulátorok, vagy Lítium-ion akkumulátor esetében a feszültségszint körülbelül 20-30%-a elvész a maximális 9 V-hoz képest (13. ábra). Az utóbbi esetekben a feszültség csak 7,2 - 7,4 V.



13. ábra: Forgási sebesség és a forgató nyomaték grafikonja, akkumulátor és elemek használata esetén (forrás [9.]).

A motorok érdekessége, hogy ha manuálisan vannak forgatva, akkor áramot termelnek, azaz generátorként is képesek működni.

TOVÁBBI BESZEREZHETŐ SZENZOROK

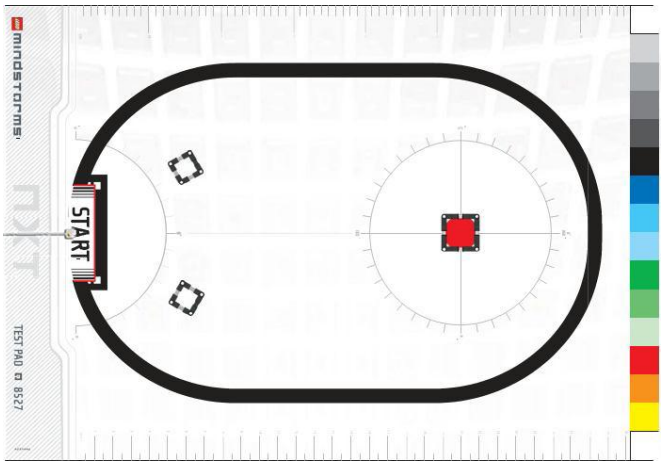
Az alapcsomagon kívül lehetőség van még szenzorok beszerzésére, habár ezek az érzékelők drágábbak a csomagban fellelhetőktől. A képen egy színérzékelő van, amellyel pontos színérzékelést érhetünk el. Ez a szenzor hat különböző színt tud egymástól elkülöníteni.

Giroszkóp szenzor képes arra, hogy a robot térbeli mozgását érzékelje, bármilyen irányú elmozdulást tud jelezni.

Iránytű. Külsőre nagyon hasonlít a giroszkóp érzékelőre, de ez a szenzor a mágneses teret képes érzékelni. Ez az érzékelő 0-359° között tud mérni. Ezeket az információkat elengedhetetlenül szokták alkalmazni a programozók a robotfoci versenyeken, ahol a támadás irányát rögzítik le vele.

TESZTPÁLYA

Minden alapkészlet tartalmaz egy tesztpályát (14. ábra), amin egyrészt kalibrálhatóak a robotok, másrészt egyszerű feladatok megoldására is alkalmas. Ilyen tipikus feladat lehet például a vonalkövetés, melyet nagyon szeretnek használni a robotprogramozók.



14. ábra: A csomagban található tesztpálya (forrás [9.]).

TÁPELLÁTÁS

A robotok igen fontos eleme az energiát biztosító rész (15. ábra). Mindkét robot (RCX, NXT) használható 6 db 1,5 V-os egyszer használatos elemekkel és újratölthetőekkel is, az első esetben viszont az összesen 9 V-os feszültség miatt a robot motorjai lényegesen gyorsabbak lesznek.

Az NXT-t alapértelmezetten újratölthető elemek által leadott 7,4 V-os feszültségre tervezték, ezt az összfeszültséget az AA vagy AAA jelzésű újratölthető ceruzaakkumulátorok, vagy a tölthető saját Lítium-ion akkumulátorok biztosítják. Ez utóbbi tervezhető teljesítményű: 1400 mAh.

15. ábra: Az NXT saját akkumulátora (forrás [9.]).

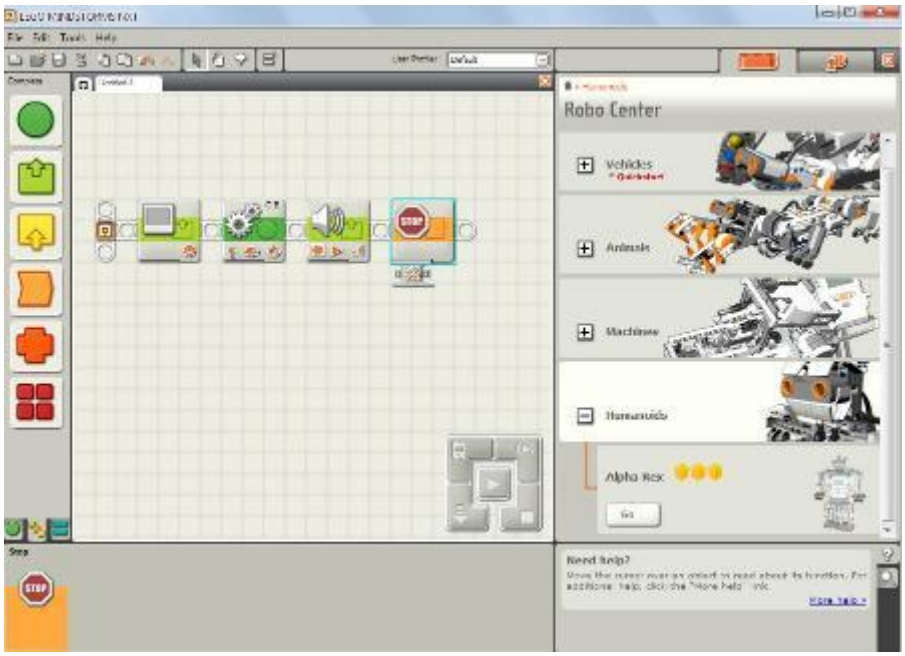
A LEGO MINDSTORMS NXT PROGRAMOZÁSI LEHETŐSÉGEI

Az NXT tégla több különféle programozási nyelven is vezérelhető. Az összes rendszer nem kerül bemutatásra, de a Java nyelv sajátosságaiba mindenképp betekintést fogok adni. Fontos megjegyezni, hogy lelkes programozóknak köszönhetően szinte naponta jelennek meg újabbnál újabb NXT kezelő rendszerek, és a már meglévő rendszerekhez frissítések.

NXT-G

Azaz a LEGO Mindstorms NXT Graphics Code.

Bármilyen NXT készlet megvásárlását követően, a csomagban található egy CD lemez, amely tartalmazza az NXT-G nevű grafikus felületű egyszerűen használható szoftvert. Ezt a környezetet azoknak hozták létre, akik a programozási nyelvekben kevésbé jártasabbak, ugyanis nem programozói szemszögből közelíti meg a problémákat. Sőt egyes irodalmak azt állítják, hogy a programozási előismeretekkel rendelkező emberek számára ez a szoftver csomag kissé átláthatatlan. Előnye ennek a programnak más nyelvekkel szemben, hogy gyorsan lehet egyszerű programokat összeállítani benne. Sajnos ezen a felületen meglehetősen bonyolult úton érhető el az NXT összes vezérlési szerkezete, és összetettebb programozási feladatok meglehetősen nehezen oldhatóak meg benne. A nagyobb programok már nem egyértelműen átláthatóak. Újabb hátránya, hogy maga a tárolt program nagyméretű, és a lefordított kód is nagy helyet foglal le a tégla memóriájából. Ezek ellenére vannak, akik mégis csak e felület mellett döntenek, mert nem kell külön telepíteni és hozzárendelni a téglához. Ebben a grafikus fejlesztő környezetben csak be kell, hogy húzogassuk az elemeket és beállítsuk a paramétereit. Majd ezek után nincs más feladatunk, csak áttölteni a programkódot.



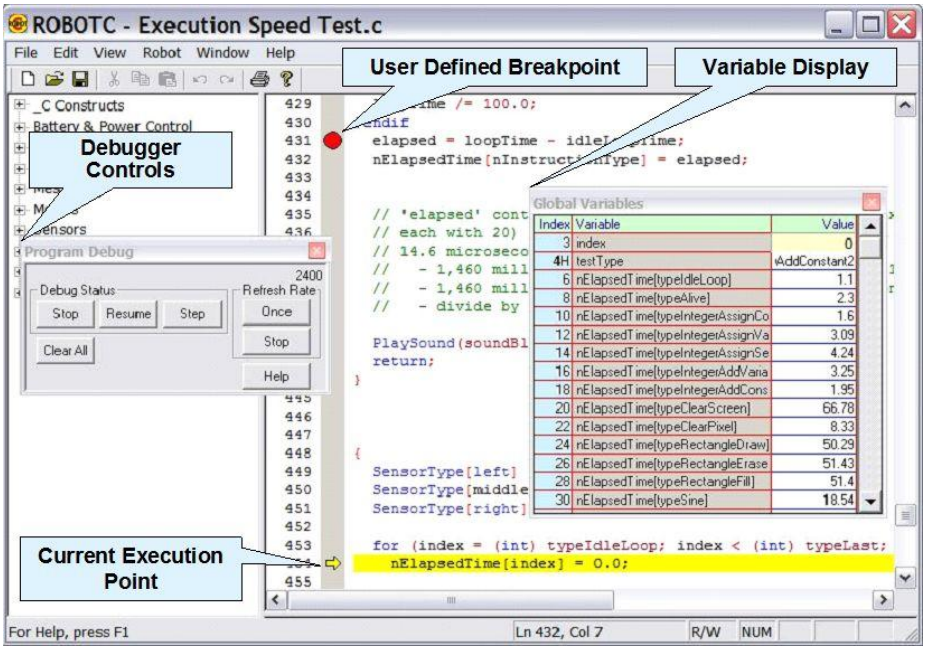
16. ábra: NXT-G szoftver képernyőmaszkja (általam készített print screen).

NBC/NXC

Lehetőség van a programozásra egy ingyenesen letölthető nyelvpáros használatával, ez az NBC/NXC. A két nyelv közül az NBC az NXT assembly szintű programnyelve, az NXC a magasabb szintű, amely a neve alapján is következtetést enged levonni: Not eXactly C. Az NBC (Next Byte Codes) az NXT egység bájtkódjának felel meg, az NXC programok is elsőre, erre a nyelvre fordítódnak le. Ez a két programozási nyelv gyakorlott programozók számára készült, mélyebb szintű betekintést enged az NXT programozási világába. Az eredményül kapott kód gyorsabb, kisebb méretű, mint a LEGO MindStorms NXT-G Software-rel előállított.

ROBOTC

Ezt a nyelvet a Carnegie Mellon Egyetem robotakadémiája fejlesztette ki, amely a robotikában komoly hírnévnek örvend. Ez a környezet nem ingyenes, de létezik egy egyhónapos próbaverzió is. Ez a nyelv teljesen a C nyelvre épül, és az NXC-vel ellentétben a C nyelv teljes funkcionalitását biztosítja és ad támogatást.

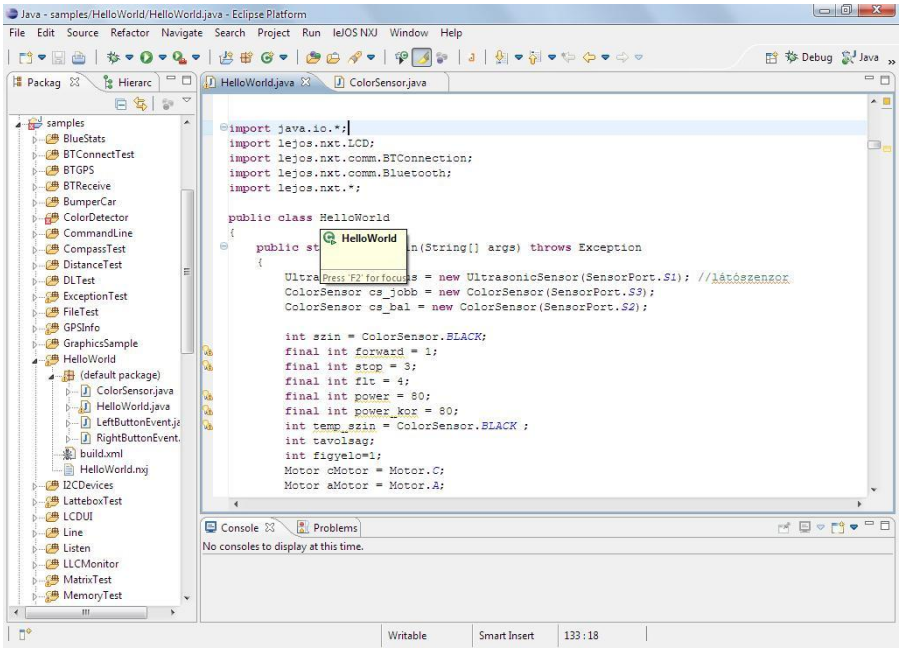


17. ábra: A RobotC képernyő-felépítése (forrás [15.]).

LEJOS

Azaz a Lego Java Operating System.

A LeJOS az a Java alapú operációs rendszer, amelyet az Eclipse vagy a NetBeans nevű objektumorientált programozási környezetet ki lehet terjeszteni. Ezáltal valósul meg az értelmezhetőség az NXT és a Java programozási nyelv között, mely az előző programozási környezetekhez képest nagymértékben túlszárnyalja őket. Ugyanis le kell cserélni az NXT teljes operációs rendszerét, ahhoz, hogy használhatóvá váljon. Ennek hatására fog megjelenni a téglán egy kis Java virtuális gép, melyhez külön dokumentációt is írtak a fejlesztők. Ennek szerkezete meglehetősen hasonlít a Java API-hoz. Ez a megoldás sem tekinthető még igazán kiforrottnak, de néhány havonta új verziók kerülnek ki a felhasználók számára.



18. ábra: Java Eclipse fejlesztői környezet képernyője. (általam készített print screen).

LEGO NXT ÉS AZ ECLIPSE

Az NXT programozására az Eclipse nevű nyílt forráskódú, platform független fejlesztő eszközzel dolgoztam.

Ez a fejlesztői környezet egyben egy felhasználói csoportot is rejt önmagában, amely folyamatosan bővíti az Eclipse alkalmazások területeit. Java nyelven sokkal erőteljesebb és rugalmasabb programokat tudtam írni, szemben azzal a szoftverrel, amelyet a LEGO biztosít a felhasználók számára. Az Eclipsenek egy modulja teszi lehetővé, hogy a megírt Java kódot át lehessen tölteni a téglára.

A LeJOS az a program környezet, amely értelmezi és engedélyezi a LEGO robotok programozását Java környezet alatt. Ez áll, egy Firmware-ból, amely tartalmazza a Java Virtuális gépét, egy classes.jar állományt, amely azokat az osztályokat tartalmazza, melyek felhasználhatóak a programok írása során.

A felhasználó által készített java osztályokhoz referenciaként hozzá kell adni a külső classes.jar állományt. Majd egy úgynevezett Linker felépít egy bináris fájlt, melybe az általunk írt java osztályokon kívül a classes.jar-ban tárolt azon osztályok kerülnek bele, amelyek használva vannak a kódban. Így egy futtatható állomány fog keletkezni, és ezt az állományt kell áttölteni az NXT-re.

JAVA PROGRAM SZERKEZETE

A Java nyelv egy objektum orientált programozási nyelv, melyben nem a műveletek megalkotása áll a középpontban, hanem az egymással kapcsolatban álló programegységek hierarchiájának a megtervezése. Ezen gondolkodásmód alapja lényegében a valós világ modelljén alapul. Minden Java programnak kötelezően tartalmaznia kell egy osztályt, és az osztály nevének, valamint a fájl nevének ugyan annak kell lennie. Főprogramnak tekinthető az osztály egy statikus metódusa.

A program felső részében importálni kell azon osztályokat, melyekre éppen szükség lesz a kódban. Ezen osztályokban vannak megírva olyan metódusok, amelyekre hivatkozásokat lehet tenni. Ilyen például a távolsági szenzor getDistance() függvénye, mely visszaadja azt a távolságot, amelyet éppen a szenzor érzékelt közte és az objektum között. Majd az importálás után jön a statikus rész, amely egy Black Box-nak is tekinthető [16.]. Melynek vannak bemenetei és van, vagy vannak kimenetei. Hogy mi játszódik le konkrétan a belsejében, azt pedig a programozónak kell meg írni. Továbbá feladat a példányosítás, deklarációk elvégzése, kezdő állapot inicializálás, és így tovább.

Dolgozatomban Java kódrészleteket fogok bemutatni, melyekben a kettős perjel szolgál a megjegyzés beszúrására. Ezt a legtöbb programozási nyelv is így értelmezi. Azon karaktereket, melyek kettős perjel mögött egy sorban vannak, azokat a fordító figyelmen kívül hagyja, és nem kerülnek lefordításra. Ilyen megjegyzés továbbá a „/* … */” jelek is, különbség csak az, hogy ennek használatakor több sornyi megjegyzést is elhelyezhetünk.

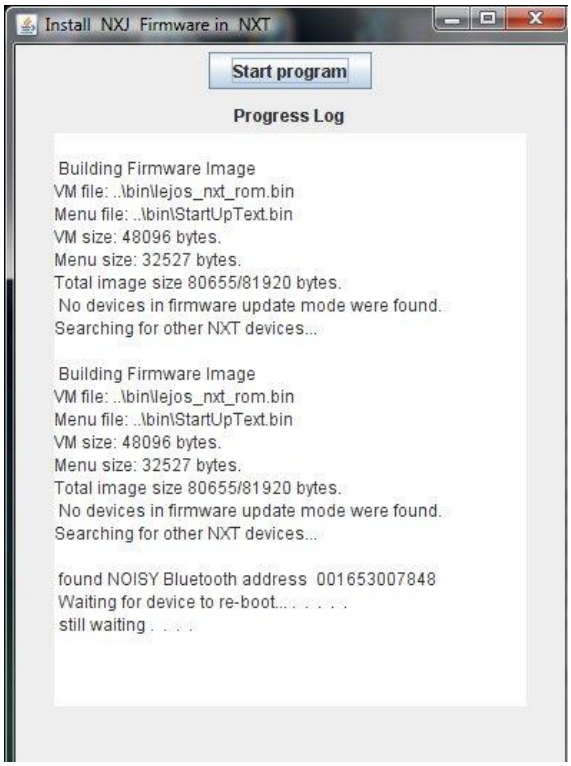
```
    import lejos.nxt.*; // osztályok importálása
    import java.io.*;
    import javax.bluetooth.RemoteDevice;
    import lejos.nxt.LCD;
//a távolság érzékelő példányosítása, amely az S1 portra lett bekötve.
  Továbbiakban a példány nevével kell rá hivatkozni, ez jelen esetben az
  „us”.
      UltrasonicSensor us = new UltrasonicSensor(SensorPort.S1);
// Itt egy színérzékelő példány jön létre, melynek a port száma 3-as
      ColorSensor cs_jobb = new ColorSensor(SensorPort.S3);
      int figyelo = 1; // Egy egyszerű egész szám deklarálása
      Motor bMotor = Motor.B; // B_motor inicializálása
      bMotor.setSpeed(340); // a B motor sebesség beállítása
```

A LEJOS ELŐNYEI MÁS PROGRAMOZÁSI KÖRNYEZETEKHEZ KÉPEST

- A Java általános szabványát használja. Ebből adódóan objektumorientált programozást tesz lehetővé,
- Nyílt forráskódú,
- Vannak saját beépülő moduljai,
- Dokumentációk és tapasztalatok alapján az NXT-G-nél sokkal gyorsabb,
- A Bluetooth-nak, USB-nek, I2C-nek, RS485 protokolloknak nyújt teljes támogatást, amely pontos irányítást tesz lehetővé,
- Támogatja a szenzorérzékelést,
- Internetes fórumok vannak, ahol a problémáinkat, ötleteinket megoszthatjuk másokkal,
- Támogatja a GPS vevőkészülékkel a Bluetooth kapcsolatot,
- Könnyen kezelhető menürendszerrel rendelkezik,
- Széles körben használják egyetemek, főiskolák és más oktatási intézmények.

Ahhoz, hogy a LeJOS fusson a számítógépen, vannak bizonyos előfeltételek, melyeknek meg kell felelni. Először is megfelelő USB vezérlésre van szükség, amely elérhetővé válik, ha a szabványos LEGO Mindstorms szoftvert feltelepítjük. Továbbá szükség van a JDK-ra (Java Development Kit [17.]) a számítógépen. Ajánlatos a JDK 1.6-ot feltelepíteni, mert előző verziókkal nem biztos, hogy működő képes lesz.

A LeJOS NXJ szoftver letölthető az internetről, melynek pontos címét a szakdolgozat végén lehet elérni [18.]. Ajánlatos időnként mindig a legújabbat letölteni, mert folyamatosan adnak ki újabb verziókat, melyekben kijavítanak hibákat, és új funkciókat integrálnak. A LeJOS-ban megtalálható néhány Java forrás fájl, amelyek példaként szerepelhetnek a programozó számára. Amikor befejeződött a telepítés a program felajánlja, a firmware frissítését az NXT-n (19. ábra). Ez a művelet mindent töröl a téglán. A régi fájlok nem kompatibilisek az új firmware-vel, e miatt sem lehet használni ezeket.



19. ábra: Firmware frissítés az NXT-n. (általam készített print screen).

OPTIMALIZÁLÁS

A környezetre való reagálás, illetve a robotmozgási sebessége is javítható, ha hatékonyabb programot készít a programozó. Az is előfordulhat, hogy egy összetettebb és így nagyobb méretű program esetleg már nem fér el a tégla memóriájába, ezért a program méretére is különös tekintettel kell lenni.

Optimalizálásnál megemlítésre méltó az NXT-G MOTOR és MOVE blokkja. Mert ez a két blokk meglehetősen nagyon hasonló. Ha mind a két blokk jelen van a programkódban, akkor kénytelenül mind a két változat fel fog töltődni az NXT-re. Ezek meglehetősen sok memóriát foglalnak le, így ha megoldható, akkor csak az egyik szerepeljen a kódban. Ha két külön-külön MOTOR blokk van hozzárendelve a szervomotorokhoz, akkor valószínű, hogy nem teljesen egyenesen fog előre gurulni a robot, hiába szabályozottak a szervo motorok. Ezért ajánlatosabb a MOVE blokk használata, mert szinkronizálja a motormozgásokat függetlenül a terheléstől. De ezzel csak egy motor meghajtása érhető el.

Hangok esetén meg kell próbálni minél rövidebb hangmintákat használni, mivel a minták meglehetősen sok helyet foglalnak le az NXT memóriájából. Egy 80 kHz mintavételű

WAV hang fájl, ami körülbelül 10 másodperc, az 80 KB helyet foglal le. Ezen okból törekedni kell a MIDI fájlok használatára.

Optimalizáláshoz ajánlott, hogy figyeljük a firmware-k megjelenését és mindig a legújabbra cseréljük le.

MEMÓRIA FELSZABADÍTÁSA

Az NXT memóriája nem túl nagy, 64 KByte. Éppen ezért nem lényegtelen, hogy az elkészített program mennyi helyet foglal le belőle. Befolyásoló tényező továbbá, hogy mennyi szabad hely áll rendelkezésre alaphelyzetben. Törölni lehet a Demó és Try- kezdetű programokat, valamint a használaton kívüli hangfájlokat is, ami például a bejelentkezési hang, gombok hangja stb.

FIRMWARE FELADATA

A Frimware egy beágyazott szoftver az NXT-ben. Lehetővé teszi a tégla számára, hogy a perifériák (motorok, szenzorok) egymással viszonylag egyszerűen kommunikáljanak. A LEGO újabb és újabb verziókat ad ki, amikor egy már meglévő hibát javítottak ki, vagy esetleg új funkcióval látták el az NXT-t. Időnként ajánlatos a firmware-t lecserélni az NXT-n a megbízhatóság és a teljesítménynövelés miatt. Szinte minden programozási környezet alatt lehetőség van a firmware frissítse. De gondosan kell ezeket telepíteni, lépésről-lépésre kell elvégezni a frissítéseket, mert a lépések nem megfelelő sorrendben történő végrehajtása esetén probléma léphet fel.

LOGISZTIKAI RENDSZEREK

A szakdolgozatomnak gyakorlati, a való világban is használatos szerepe van. Hiszen a komissiózás, raktározás jelen van az ellátási rendszerekben. E miatt tartom fontosnak, hogy kifejtsem azon fogalmakat, definíciókat, melyek szerepet játszanak a dolgozatomban.

Logisztikai rendszer alatt egy vagy több ellátó és fogyasztó közötti anyag- és információáramlás megvalósításában közreműködő eszközöket, létesítményeket és szervezeteket értünk.

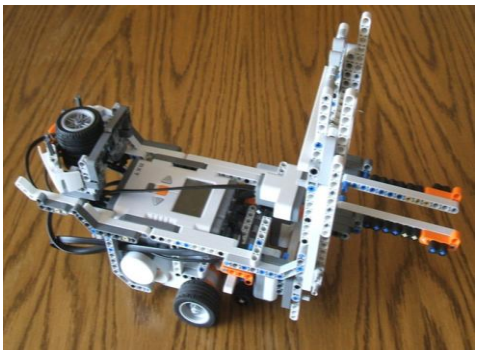
Logisztikai eszközök, azok a rendszeren belüli anyag- és információáramlásban résztvevő eszközöket jelenti, melyek gépeknek tekinthetőek. Az anyagáramlás megvalósításában fontosabb eszközök a közúti, a vasúti, a vízi, a rakodó és a szállító járművek.

Logisztikai létesítményeknek tekinthető az anyag- és információáramlásban érintett utak, szállítópályák, területek és épületek. Létesítmények közül külön kiemelendőek a raktárépületek.

Szállítás alatt az áruk olyan helyzetváltozását értjük, mely viszonylag rövid idő alatt helykoordinálással járnak.

Az általam megépített emelőtargonca a szakaszos működésű anyagmozgató gépek nagy csoportjába tartozik. Ezen belül is a gépi hajtású targoncák csoportjába, mely az anyagmozgatást valósítja meg a logisztikán belül. Pontosabban a termelési rendszer és a raktári rendszer közötti rakodó, anyagmozgató feladatok ellátásáért felelős.

A TARGONCA



20. ábra: A megépített targonca (saját kép)

Szakdolgozatomban három NXT csomagot használtam fel az építéshez. Az elsőből egy targoncát építettem össze, amely az anyagmozgatási feladatok ellátását szolgálja. Jelen esetben, hogy a szétválogatott elemeket beszállítsa a raktárba.

A targoncákat a valós világban, a komplex anyagmozgató rendszerek területén jelentkező feladatok elvégzéséhez használják. A targoncák szállítási feladatait funkciók szerint több csoportba osztjuk, ezt a 21. ábra mutatja.

21. ábra: A targoncás feladatok funkciók szerinti csoportosítása (forrás: [19.] átdolgozott ábra).

- Üzem-üzem közötti szállítás:

A gyakorlati életben sokszor előfordul, hogy egyes termékek előállítása külön technológiai sorrend alapján valósul meg. Azaz ezeket külön jól elhatárolt üzemekben állítják elő. Így az egyes üzemek között jelentkező szállítási feladatokat a technológiai sorrendnek megfelelően biztosítani kell.

- Üzemek-raktárok közötti szállítás:

Az üzemek az eltérő technológiai idő miatt a termelési-szolgáltatási folyamatokat raktárak létesítésével oldják meg. Ez nyújt segítséget számukra, hogy folyamatos anyagáramlást tudjanak biztosítani. Ezen szállítás esetében kettős feladat elvégzését kell megoldani. Az első feladat, hogy az üzem tevékenységei számára a szükséges anyagáramlást biztosítani kell a raktárból. Másodrészt fontos, hogy az üzemben a tevékenységek által kapott termékeket biztosítani kell a raktár felé.

- Raktáron belüli szállítás, rakodás:

Ilyen feladatok lehetnek a raktár bemeneti pontjaihoz érkező termékek raktáron belüli elhelyezése. Valamint az innen kikerülő, a raktár kimeneti pontjaihoz történő kiszállítás. Ezen feladatokhoz tartoznak továbbá a raktár átrendezésekor felmerülő anyagmozgatások. Végül a

raktárral összefüggő, a raktáron belül megvalósuló egységrakomány-képzéssel, és komissiózással járó anyagmozgató feladatok.

- Üzemen belüli szállítás, rakodás:

Ez a targoncás feladat fordul elő leggyakrabban. Egy adott üzemen belül gyakran szállítási, rakodási feladatok jelentkeznek, melyek megoldására a legoptimálisabb választás a targoncás anyagmozgatás.

- Szállító járművek ki- és berakodása:

A kiszállítás esetében a szállító eszközöket meg kell rakodni termékekkel, illetve beszállításra érkezett szállító járműveket ki kell rakodni. Amit gyakran targoncákkal végeznek el.

Természetes dolognak tekinthető, hogy különféle rakodó, szállító igények felmerülésekor különböző targoncák alkalmazására kerül sor. A szállítási feladatokra való alkalmasság alapvető szempontjait a 22. ábra mutatja.

22. ábra: A targoncák szállításra való alkalmasságát befolyásoló alapvető tényezők. (forrás [19.] átdolgozott ábra).

Az ábrából kiderül, hogy a szállításra kiválasztott targoncának több paramétere is van, mely befolyásolja az alkalmasságot. Ezek a következők:

Geometriai paraméterek: befoglaló méret, súly, …

Üzemeltetési paraméterek: diesel, elektromos, gáz, …

Műszaki paraméterek: terhelés, gyorsulás, kormányzás, teljesítmény, fordulatszám, ...

Logisztikai paraméterek: vontatásra alkalmas, rakodásra alkalmas, elemelésre alkalmas, speciális megfogásra alkalmas, …

Karbantartási paraméterek: szervizelések, megbízhatóság.

Rendszerbe való illeszthetőség paraméterei: ár, átfutási idő, …

A targoncás anyagmozgatás számítógépes irányítása azért kiemelt jelentőségű, mert nagymértékben tudja fokozni a logisztikai feladatok elvégzésének hatékonyságát, a termelési-szolgáltatási teljesítőképességet, és igen fontos tényező, hogy nagymértékben csökkenti a költségeket.

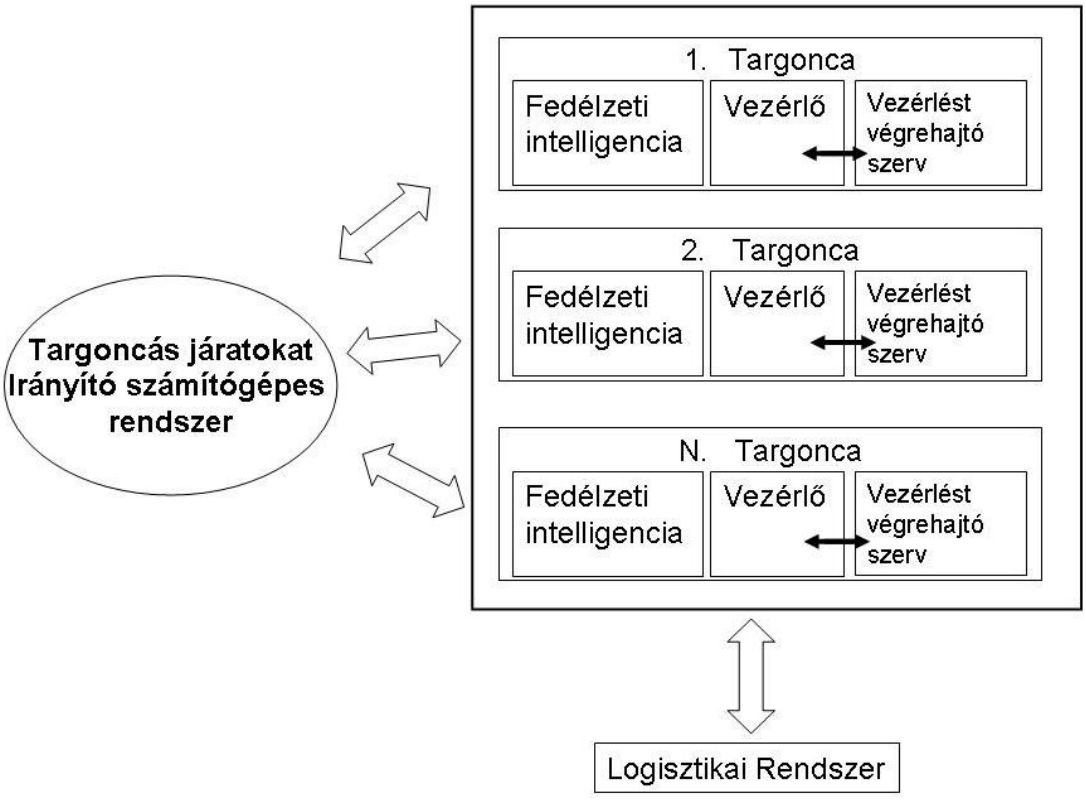
A TARGONCÁS ANYAGMOZGATÁS SZÁMÍTÓGÉPES VEZÉRLÉSEINEK ELŐNYEI

- Növelheti a targoncákkal kiszolgált termelő és szolgáltató egységek kihasználtságát, teljesítőképességét.
- A targoncás szállítást végző anyagmozgatást és a kapcsolódó anyagokat és árukat követhetővé teszik, ezek adatait a vállalat számítógépes információs rendszerébe beépíthetik.
- Használhatja a vállalat számítógépes integrált adatrendszerét, ezért az irányítás megfelelően friss, a teljes rendszer áttekinthető, meghozható információk alapján történhetnek a diszponálások.
- Csökkenthető a szükséges targoncaszám, ezáltal, valamint diagnosztikai adatok gyűjtésével a karbantartási költségek is csökkenthetők.

A számítógépes targoncás anyagmozgatás kétségtelenül nagy beruházást jelent, de nem minden esetben vonja magával, hogy új targoncákat kell beszerezni. Elegendő bizonyos változtatásokat eszközölni, ilyen lehet például az intelligens vezérlők telepítése. Valamint az

esetek döntő többségében rendelkezésre áll egy vállalati számítógépes információs rendszer, amelyhez való kapcsolódás általában, bizonyos interfészek kiépítésével megoldható.

Működést tekintve a targoncákat irányító számítógép utasításokat ad a szállítóeszközöknek, és visszajelzéseket kap az elvégzett munkáról. A targoncát vezérelheti vezető, vagy fedélzeti intelligencia. A rendszerről való adatgyűjtésnek két típusa van, online (automatikus), vagy offline (kézi). Az online féle adatgyűjtést, úgy kell értelmezni, hogy a rendszerből folyamatos adatgyűjtés történik. Továbbá csak olyan adatokat lehet gyűjteni, amelyekhez érzékelők vannak hozzá kötve. A targoncás járatokat irányító számítógép és a targoncák között az információs kapcsolat lehet online (folyamatos) adatátvitel, illetve offline (vezetékes). A targoncás járatokat irányító számítógép információs kapcsolatot tart a logisztikai részrendszerek valamelyik számítógépével illetve a termelés, tervezés és irányítás számítógépével (23. ábra).



23. ábra: A targoncás anyagmozgatás számítógépes irányításának általános elve. (forrás [19.] átdolgozott ábra).

A TARGONCA PROGRAMKÓDJÁNAK RÉSZLETEIBE VALÓ BETEKINTÉS

A targonca feladata, hogy egy kezdőpontból eljusson a felrajzolt vonal mentén a válogató robothoz. Majd ott felszedje a kosarakat, melyben a szétválogatott elemek vannak, és ezeket elszállítsa a raktárba, megadott útvonalon keresztül. Ha az anyagmozgatási folyamat megtörtént, akkor jusson vissza az előre definiált kezdőpontra, amely felfogható úgy is, mint a targonca egy parkoló helye.

Ebben a fejezetben hozok példákat a targonca mozgásának Java nyelven történő megvalósításaira.

A „Java program szerkezete” című fejezetben lett megemlítve, hogy hogyan épül fel a program váza. Konkrét példán keresztül lett bemutatva, hogy hogyan tudjuk deklarálni az érzékelőinket. Ebben a fejezetben lesz bemutatva a vonalkövetési program működése.

Alapvetően nagyon egyszerű a működése, amikor elindul a program részlet azelőtt kalibrálni kell az érzékelőket, melyet külön-külön kell elvégezni a két szenzorra. A kalibrálás annyit jelent ebben az esetben, hogy a szenzoroknak meg kell mutatni a fekete és a fehér színt, melyeknek értékei letárolásra kerülnek külön változókba. Így érhető el az, hogy a robot végig tudjon menni a felrajzolt vonalon. Azért fontos a két szenzort külön kezelni, mert a két fényérzékelő sem egyforma, eltérő értékeket mutat ugyanarra a színre. Amennyiben külön vannak kezelve az érzékelők, akkor meglehet előzni egy sor problémát.

```
public ColorSensor(SensorPort p)
{
        ls = new LightSensor(p);
        ls.setFloodlight(true);
}
private int read(String color)
{
        int lightValue=0;
        while (Button.ENTER.isPressed());
              LCD.clear();
              LCD.drawString("Press ENTER", 0, 0);
              LCD.drawString("to callibrate", 0, 1);
              LCD.drawString(color, 0, 2);
        while( !Button.ENTER.isPressed() )
```

```
        {
              lightValue = ls.readValue();
              LCD.drawInt(lightValue, 4, 10, 2);
              LCD.refresh();
        }
        return lightValue;
}
```

Ez után annyi teendő van, hogy egy cikluson belül le kell kérni azt a szint, amelyet éppen lát az adott pillanatban a szenzor, és ha korrigálnia kell, akkor javítani fog a mozgásán a két oldalra felszerelt motorok segítségével. Konkrétan a kód úgy lett megírva, hogy ha a jobb szenzor érzékel fekete színt, akkor a bal motor javítson a mozgáson. Ha pedig a bal érzékelő lát feketét, ezen esetben pedig a jobb motor korrigálja a mozgást. Amennyiben nincs szükség a korrigálásra, a robotnak előre kell mennie.

```
public void LeftCorrection()
  {
        rightMotor.setSpeed(0);
        leftMotor.setSpeed(470);
  }

  public void RightCorrection()
  {
        leftMotor.setSpeed(0);
        rightMotor.setSpeed(470);
  }
  public void Forward()
  {
        leftMotor.setSpeed(340);
        rightMotor.setSpeed(340);
        leftMotor.forward();
        rightMotor.forward();
  }
```

Gyakorlati példa lehet, ha egy robot, mely emberi közbeavatkozás nélkül végzi a munkáját felkészült legyen arra, hogy olyan helyzetek is előállhatnak, melyre különös kép tekintettel kell lennie. Ilyen lehet, ha egy gyárban a szállító robot végzi feladatát és egy

váratlan helyzetben útjába kerül egy ember, vagy egy másik robot. Ezen esetben fontos dolog, hogy a robot felfüggessze tevékenységét, ne kerüljön ütközési helyzetbe. Addig ne tegyen semmit, míg biztonságosan nem tud tovább haladni, és az akadály ki nem került a targonca hatóköréből. Ezzel csökkenthető a balesetek kialakulásának kockázata. A targonca esetében is meg van ugyan ez, mindaddig felfüggeszti a program futását, amíg el nem került az útjából az általunk oda tett torlasz.

A következő kód azt fogja bemutatni, hogy ha a targonca távolságérzékelő szenzorja 20 cm távolságban érzékel egy tárgyat, akkor megáll és vár, amíg el nem kerül az útjában álló tárgy. A távolságérzékelő által mért érték folyamatosan kiírásra kerül az LCD képernyőre.

```
LCD.clear();
LCD.drawString("cm1:", 0, 0);
LCD.drawInt(us.getDistance(), 5, 0);
LCD.refresh();
tavolsag = us.getDistance();
if(tavolsag <= 20)
{
      aMotor.stop();
      bMotor.stop();
      cMotor.stop();
      LCD.drawString("stop", 3, 4);
      continue;
}
```

A VÁLOGATÓ ROBOT

24. ábra: A megépített válogató robot (saját kép)

A raktározás szempontjából fontos funkciót lát el a válogató robot. Jelen esetben, mint ahogy a képen is látható a szétválogatott elemek külön kosarakba kerülnek, melyek a könnyű mozgathatóság figyelembe vételével lettek megtervezve. A robot a rászerelt fényérzékelő segítségével végzi el az objektumok megkülönböztetését, ez nyújt segítséget a különböző színek szétválogatásában.

Pontosabban két féle szín közül dönti el, hogy épp az adott golyó milyen színű és ez alapján pakolja bele a kosárba a golyókat. A Mindsorms NXT új készletben négy féle színű golyót találhatunk meg, ezek színei a kék, a sárga, a piros és a zöld. Alapvetően az volt a célom, hogy két fajta színt jól meg tudjak különböztetni egymástól. Mérések alapján döntöttem a sárga és a zöld golyók mellett. A fényszenzor a kék színt néha zöldnek látta, a piros szint pedig néha sárga golyónak érzékelte, a fényviszonyok okozta változások miatt. De a sárga és a zöld színt alapjában véve jól el tudta különíteni egymástól.

A programot tekintve a válogatást a következő képen végzi el a robot. Mindegy, hogy milyen színű az első golyó, a szenzor megvizsgálja és elteszi az egyik kosárba. Természetesen egy változóban letárolta, hogy milyen értékkel dolgozott. Majd sorra jönnek a golyók és, amelyiknek a színe különbözik az elsőtől, azt fogja a másik rekeszbe pakolni. Ezzel lehetett kiküszöbölni azt a problémát, melyet a fényviszonyok okoznak a programozó számára.

```
else if( Math.abs(cs.readValue() - szin1) <= 2)
 {
       Thread.sleep(750);
       bMotor.rotate(50);
       Thread.sleep(750);
       aMotor.rotate(-70);
       Thread.sleep(750);
       aMotor.rotate(70);
       Thread.sleep(750);
```

```
        bMotor.rotate(-50);
        Thread.sleep(750);
        darab = darab - 1;
        kezdo_irany_bal = true;
    }
    else if(Math.abs(cs.readValue() - szin2) <= 2)
    {
        Thread.sleep(750);
        bMotor.rotate(-50);
        Thread.sleep(750);
        aMotor.rotate(-70);
        Thread.sleep(750);
        aMotor.rotate(70);
        Thread.sleep(750);
        bMotor.rotate(50);
        Thread.sleep(750);
        darab = darab - 1;
        kezdo_irany_jobb = true;
    }
```

A kosár, melybe kerülnek a golyók egy saját tervezésű viszonylag egyszerű, téglalap alapú hasáb, melyet elég szélesre kellett megépíteni ahhoz, hogy a targonca villái közéférjenek, és hogy ne essen le szállítás vagy a forgolódás közben.

A válogató robot esetében nem elemeket, hanem akkumulátort használtam, így a tesztelések során nem kellett szétszerelni a robotot. Ez nagy segítséget jelentett, mert egyszerűen a hálózati áramról valósult meg a töltés.

BLUETOOTH KOMMUNIKÁCIÓ

A Bluetooth egy távközélési ipari szabvány és protokoll vezeték nélküli adatátvitelre. Segítségével kis rádiós kapcsolatot tudunk kialakítani számítógépek, mobiltelefonok vagy épp jelen esetben NXT-k között. Érdekesség kép azt érdemes tudni, hogy maga a Bluetooth szó honnan is származik [20.]. Az elnevezést I. Harald dán királyról kapta, aki nagyon szerette az áfonyát, ezért kék lett a foga. Azért volt nevezetes Harold király, mert egyesítette a dán, a norvég és a svéd törzseket. A Bluetooth-t is arra használják, hogy különböző, eltérő eszközöket össze lehessen kötni.

NXT és NXT között USB csatlakozás sajnos nem lehetséges. A Bluetooth kapcsolat esetében egy úgynevezett Master – Slave viszony valósul meg. Egy időpillanatban a Master mindössze három Slave egységgel tud kapcsolatba lépni. A Master fogja a kapcsolatot létrehozni, ő fogja küldeni az adatot. Számítani fog a Slave visszajelzésére, ha ez nem történik meg akkor egy hibaüzenetet fog kiírni a kijelzőre. Figyelni kell arra, hogy sajnos előfordulhat, hogy a kommunikáció során elvesznek egyes adatok. Ez a következő miatt lehetséges, hogy az üzenet küldésekor mindig meghatározódik egy postaláda, és ha több NXT kapcsolódik egy Master-hez, akkor sorban állási helyzet alakul ki. Az NXT postaládák öt üzenet tárolására képesek, és amikor egy üzenet mindenféle képen be szeretne kerülni egy már megtelt sorba, akkor a legrégebbi üzenet törlődni fog. Ez a szituáció csak a Slave oldaláról fordulhat elő, a sorban állási kényszerítettség miatt. Üzenetvesztés azon esetben is előfordulhat, ha a Master hamarabb kezdi el küldeni az adatokat, mint azt a Slave fogadni tudná. Üzenetek elvesztése ellen megoldást nyújthat, ha ellenőrizzük a sorok állapotát, és csak akkor küldünk, ha nincs a sor telítődve.

KOMMUNIKÁCIÓ NXT ÉS NXT KÖZÖTT

Dolgozatomban kommunikációra volt szükségem az NXT egységek között. Konkrétan, amikor a válogatás befejeződik, ezt jeleznem kell a szállító járműnek. Amely a jel fogadása után a rekeszekért indul.

Először úgy szerettem volna megoldani, hogy Morse jeleket küldök a targonca számára. Amelyhez a hangszenzor nyújtott volna segítséget. De aztán biztosabb és egyszerűbb

megoldásnak bizonyult, ha a már említett Bluetooth kommunikáció használatával oldom meg ezt a problémát.

Alapvetően a kommunikáció a következő hat lépésből áll:

- A vevőkészülék: a kapcsolat kialakítására várakozik,
- A küldő készülék: csatlakozik a vevőhöz,
- A vevőkészülék: ha kialakult a kapcsolat, akkor a vevő adatra vár a küldőtől,
- A küldő készülék: elküld egy egész számot a vevő számára,
- A vevőkészülék: ha fogadta az egész számot, azt feldolgozza és az eredménynek megfelelően fog dönteni.
- A küldő és a vevő lezárja a kapcsolatot.

```
LCD.clear();
LCD.drawString("Connecting...", 0, 0);
LCD.refresh();

try
{
   RemoteDevice sender = Bluetooth.getKnownDevice(name);
         if (sender == null)
   throw new IOException("no such device");
   BTConnection connection = Bluetooth.connect(sender);
         if (connection == null)
   throw new IOException("Connect fail");

   LCD.drawString("connected.", 1, 0);
   DataOutputStream output = connection.openDataOutputStream();
   if ( (kezdo_irany_bal == true ) && (kezdo_irany_jobb == true))
         {
               output.writeInt(1);           // A_motor fordult el
               output.flush();
         }
   else if( (kezdo_irany_jobb == true) || (kezdo_irany_bal == true ) )
         {
               //kezdo_irany_bal = false;
               output.writeInt(2);            // C_motor fordult el
               output.flush();
         }
```

```
LCD.drawString("Sent data", 2, 0);
output.close();
connection.close();
LCD.drawString("Bye ...", 5, 0);
}catch(Exception ioe)
      {
            LCD.clear();
            LCD.drawString("ERROR", 0, 0);
            LCD.drawString(ioe.getMessage(), 2, 0);
            LCD.refresh();
      }
```

A fenti programkód részlet valósítja, meg a targoncával való vezeték nélküli összekapcsolódás egyik oldalát. Konkrétabban megpróbál összekapcsolódni egy másik NXT egységgel, amelyik az adatcsatornát figyeli, hogy jön e jel. Amennyiben megvalósul a kapcsolódás akkor a válogató robot elküld egy számot. Amely vagy az 1, vagy a 2 egész szám egyike. Ebből az információból a targonca tudni fogja, hogy melyik irányba, melyik úton haladjon végig.

TÁVIRÁNYÍTÓ JOYSTICK

25. ábra: A joystick (saját kép)

A harmadik NXT csomagból egy joystick-ot építettem össze, mely képessé teszi a targoncát a távolsági vezérelésre, vezeték nélkül. Itt is a Buletooth kommunikációt lehetett hasznosítani. Ehhez a távvezérlő készülékhez szükség volt két motorra és két érintő szenzorra. A motorokkal valósult meg, hogy bármely irányba el tudjon mozdulni a targonca.

Valamint ezen esetben a motorok bemeneti funkciót látnak el. A két érintő szenzor pedig a targoncán lévő emelőszerkezet mozgatásáért felelős. Elvi működését tekintve figyelni kell, hogy a motorok és a kapcsolók milyen állapotban vannak. Ehhez pedig getTachoCount() függvény bizonyult hasznosnak, mely a motorok forgási irányáról ad információt. Az egyik

motorral a jobbra-balra való elfordítás, a másikkal pedig az előre-hátra való haladás valósult meg. Az előbb említett getTachoCount() függvény úgy működik, hogy az vissza ad egy értéket a programozó számára. Az első esetben, ha az érték nulla, akkor az azt jelenti, hogy nem történt elmozdítás. Ha pozitív számot ad vissza, akkor azt jelenti, hogy az egyik irányba elforgattuk a motort, ha pedig negatív számmal tér vissza, akkor a másik irányba történt elmozdítás. Ennek a megoldásnak a program részlete a következő néhány sorban olvasható. A következő sorokban nyolcirányú mozgás van lekódolva, melyet a targonca képes végezni a távirányító által vezérelve.

```
a = aMotor.getTachoCount();
b = bMotor.getTachoCount();
LCD.clear();
LCD.drawString("elore:", 0, 1);
LCD.drawInt(a , 9, 1);
LCD.drawString("fordulas:", 0, 2);
LCD.drawInt(b, 11, 2);
LCD.refresh();

if( (a<20 && a>-20 ) && b<-20  )       // csak jobbra_fordul a targonca
 {
       output.writeInt(Command.RIGHT);
       output.flush();
 }
else if( (a<20 && a>-20) && b>20 )     // csak balra_fordul a targonca
 {
       output.writeInt(Command.LEFT);
       output.flush();
 }
else if( a>20 && ( b>-20 && b<20 ) )      // előre megy a targonca, ha
az A_Motor szögelfordulása nagyobb, mint 20, a B_Motoré pedig -20 és
20 között van.
 {
       output.writeInt(Command.FORWARD);
       output.flush();
 }
else if ( a<-20 && ( b>-20 && b<20 ) )    // hátra megy a targonca, ha
az alábbi feltétel teljesül.
```

```
 {
       output.writeInt(Command.BACKWARD);
       output.flush();
 }
else if ( a<-20 &&  b<-20  )     // jobbra_előre megy a targonca, ha a
feltétel igazzá válik
 {
       output.writeInt(Command.FORWARD + Command.RIGHT);
       output.flush();
 }
else if ( a<-20 &&  b>-20 )       // balra_előre megy a targonca, ha a
feltétel teljesül
 {
       output.writeInt(Command.FORWARD + Command.LEFT);
       output.flush();
 }
else if ( a>20 &&  b<-20 )       // jobbra_hátra megy a targonca, ha a
feltétel teljesül
 {
       output.writeInt(Command.BACKWARD + Command.RIGHT);
       output.flush();
 }
else if ( a>20 &&  b>-20 )       // balra_hátra megy a targonca, ha a
feltétel teljesül
 {
       output.writeInt(Command.BACKWARD + Command.LEFT);
       output.flush();
 }
else         // stoppolja a motorokat, a targonca áll
 {
       output.writeInt(Command.STOPLEFT + Command.STOPRIGHT);
       output.flush();
 }
```

USB KOMMUNIKÁCIÓ

Ez méltán népszerű kommunikációs megoldás, hiszen az eszközök túlnyomó többségén megtalálható ez a csatoló felület. Valamint nagy előnye, hogy kevés vezeték használatával lehet információt átvinni. Azonban vannak hátrányai, amelyek közül első az, hogy az általa nyújtott adatátviteli sebesség viszonylag kicsi. Továbbá pont-pont típusú kapcsolat miatt mindössze egy készülék kapcsolható hozzá. Valamint használatához negatív tápfeszültség is szükségeltetik. Az USB busz egy mester-szolga típusú félduplex kommunikációt valósít meg. A kommunikáció vezérlését mindig a mester végzi. Az USB2.0 szabvány 480 Mbit/sec adatátviteli sebesség elérését is képes lehetővé tenni.

A LEGO Mindsorms NXT esetében is lehetséges az USB kommunikáció. Melyet a programozás során úgy hasznosítottam, hogy a válogató robotot USB-n keresztül összekötöttem a számítógéppel. A számítógép feladata volt, hogy összekapcsolódjon a robottal, és ezen keresztül kell megadni a szétválogatni kívánt golyók számát. A következő kódrészlet szintén hiányos, de működését tekintve a futtatás után összekapcsolódik egy NXT-vel USB csatolón. Majd az Eclipse konzoljára kiírja, hogy „Adja meg a szétválogatni kívánt golyók számát:”. Ezután ezt a számot elküldi a válogató robot számára. Majd miután a robot befejezte a golyók szelektálását, jelzi a számítógép számára. Amely úgy ad vissza információt erről, hogy kiírja a konzolra a „a szétválogatni kívánt golyók száma:… Ebből a szétválogatott golyók száma:…”, és végül lebontja a kapcsolatot.

```
if (!conn.connectTo("usb://"))
      {
            System.err.println("Nincs NXT kapcsolva a sz.gephez USB-n");
            System.exit(1);
      }
      ...
      System.out.println("\nAdja meg a szétválogatni kívánt golyók
      számát:");
      ...
      {
      try
```

```
{
      outDat.writeInt(be);
      outDat.flush();
} catch (IOException ioe)
      {
            System.err.println("IO Exception writing bytes");
      }
try
{
      x = inDat.readInt();
} catch (IOException ioe)
      {
            System.err.println("IO Exception reading reply");
      }
System.out.println("A szétvállogatni kívánt golyókszáma: " +be +
"\nEbből a szétvállogatott golyók száma: " + x);
}
try
{
      inDat.close();
      outDat.close();
      System.out.println("Adat küldés lebontva");
} catch (IOException ioe)
      {
            System.err.println("IO Exception Closing connection");
      }
try {
      conn.close();
      System.out.println("Kapcsolat lebontva");
      } catch (IOException ioe)
      {
            System.err.println("IO Exception Closing connection");
      }
```

GRAFIKUS FELHASZNÁLÓ FELÜLET

A mai világban a Windows rendszerek meglehetősen nagymértékben megváltoztatták az emberek elvárásait a felhasználói programokkal szemben. Ilyen elvárások például, hogy legyen grafikus felülete a programnak, egeret lehessen használni a programban.

Egy ideig az Eclipse konzol ablakát használtam adat bevitelre, és szöveg kiíratásra. Majd ezt követően gondoltam, hogy készítek egy grafikus felhasználói ablakos alkalmazást.

Mivel a Java rendszer független nyelv, ezért különböző operációs rendszereken képes futtatható kódot generálni. Java programozási nyelv alatt a felhasználó felület készítése két osztálykönyvtáron alapul, ezek a Swing és AWT. Ez biztosítja a grafikus felület és a felhasználó közötti kommunikációt.

Egy felhasználó felület megjelenítéséhez objektumokat kell létrehozni, és a tagfüggvényeit meghívni. Grafikus felhasználói felület esetében a komponensekre, és tárolókra kell tekintettel lenni. A komponens a felhasználói felület egy önálló része, a tároló pedig egy olyan komponenst jelöl, mely több más komponenst is tartalmaz. Felhasználói felület készítése során először a tárolót kell kialakítani, melyekben lesznek a komponensek. A tároló gyakran egy keret, egy ablak. Amelyek a felhasználó asztalán jelennek meg.

A saját grafikus felhasználói ablakom tartalmaz egy képet arról a pályáról, melyet a targonca bejár, valamint található még egy „Ok” gomb, amely az adatbevitel érvényesítésére szolgál. Továbbá van két szövegmező is értelmezve a grafikus felületben. Az egyik adatbevitelre szolgál, a másik pedig adat kiírásra.

```
public Display()
      {
      //képernyő_keretének_a_címke_beállítása
            super("NXT képernyő");
            setSize(450,250);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      //szövegmező_amelybe_kiírásra_kerül_a_robot_művelet
            labelLoger = new JLabel("A Robot épp ezt hajtotta végre:
            ", JLabel.RIGHT  );
            txtFieldLogger = new JTextField(20);
```

```
        add(labelLoger);
        add(txtFieldLogger);
        txtFieldLogger.addActionListener(this);

    //a_golyók_bevitele_a_szövegmezőbe
        labelBallNumber = new JLabel("Adja meg a szétválogatni
        kívánt golyók számát: ", JLabel.RIGHT  );
        txtFieldBallNumber = new JTextField(10);
        add(labelBallNumber);
        add(txtFieldBallNumber);
        txtFieldBallNumber.addActionListener(this);

    //ok_gomb_megrajzolása
        ok = new JButton("Ok");
        add(ok);
        ok.addActionListener(this);

    //pálya.jpg_megjelenítése
        labelBackgroundPicture = new JLabel();
        labelBackgroundPicture.setIcon(new
        ImageIcon("C:\\palya.jpg"));
        add(labelBackgroundPicture);
        pack();
        usbs = new USBSend();
        usbs.addMyEventListener(this);
        setLayout(new FlowLayout());
    }//Display_vége
```

Egy grafikus felhasználói interfész (felület) eseményvezéreltnek kell, hogy legyen. Ami azt jelenti, hogy a programot a program futása közben bekövetkező események vezérlik. Események keletkezhetnek a programon kívül is. Ilyenek például a billentyűk leütése, egér elmozdítása. Az esemény is egy objektum, amely eljut a megfelelő figyelőobjektumhoz, melyek feldolgozzák az eseményeket.

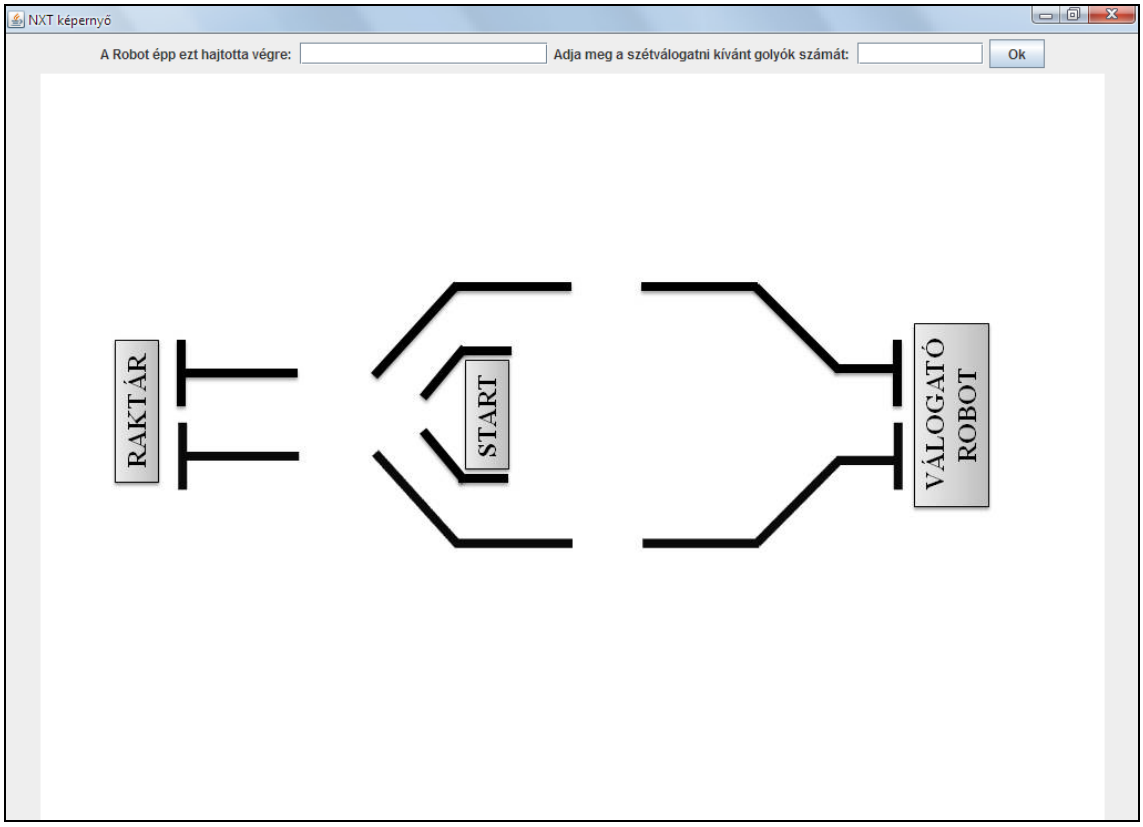
Jelen esetben ahhoz, hogy a keret figyelje és kezelje, a nyomógomb eseményeit, annyit kell tenni, hogy definiálni kell a java.awt.event csomagot. Majd a figyelő osztálynak kötelezően implementálnia kell az ActionListener interfészt. Ezt az interfészt implementálni kell, és kifejteni annak egyetlen actionPerformed(ActionEvent ev) metódusát. Ezt követően a

nyomógomb tulajdonságait fel kell jegyezni a tulajdonosi hierarchiába. Majd hozzá kell fűzni a figyelőláncba.

```
import java.awt.event.*;

public class Display extends JFrame implements ActionListener,
MyEventListener, EventListener
{
//ok_gomb_megrajzolása
      ok = new JButton("Ok");
      add(ok);
      ok.addActionListener(this);
      public void actionPerformed(ActionEvent evt)
      {
            Object source = evt.getSource();
            if (source == ok)
            {
                  try
                  {
                  int number =
                  Integer.parseInt(txtFieldBallNumber.getText());
                  usbs.SendNumber(number);
                  }
                  catch (Exception ioe)
                  {
                        System.err.println("Nem számot adott meg!");
                  }
            }
      } //public_void_actionPerformed_vége
```

Ahhoz, hogy tökéletes program fusson a számítógépen, egyszerre több feladattal kell foglalkozni egy időben. A Java nyelv egyik legjelentősebb tulajdonsága, hogy képessé válunk több szálon futó programot írni benne. A Java programokban azokat a feladatokat, amelyeket a számítógép egy időben kezel, szálaknak nevezik. Ezzel lehet elérni, hogy egyszerre több dologgal tudjon foglalkozni a program. Minden hosszabbideig tartó végrehajtandó feladatot érdemes külön szálba tenni, azaz külön osztályként kell meghatározni. A hosszabb ideig tartó

feladatok külön szálba tételével megelőzhető a grafikus felhasználói felület lelassulása, vagy épp rendellenes működése.



26. ábra: Grafikus felhasználói felület (általam készített print screen).

Ezen a grafikus felületen (26. ábra), valósul meg az értesítés a felhasználó számára, hogy a robot melyik műveletet hajtotta végre. Ezt a képernyő bal felső sarkában található „A Robot épp ezt hajtotta végre:” melletti szövegmezőben írja ki.

```
public void NumberRecievedEvent(EventArgs args)
      {
            switch (args.getNumber())
            {
                  case DisplayMessage.MOVE          :
txtFieldLogger.setText("A targonca elindult a Parkoloból.  ");
break;
                  case DisplayMessage.ARRIVE        :
txtFieldLogger.setText("Megérkezett a válogató robothoz.   ");
break;
```

```
                case DisplayMessage.UP             :
txtFieldLogger.setText("Felveszi a rakományt.              ");
break;
                case DisplayMessage.TAKEAWAY       :
txtFieldLogger.setText("A rakomány elszállítása a raktárba.");
break;
                case DisplayMessage.ARRIVERAKTAR   :
txtFieldLogger.setText("A targonca a raktárba érkezett.    ");
break;
                case DisplayMessage.DOWN           :
txtFieldLogger.setText("Rakomány elhelyezése.              ");
break;
                case DisplayMessage.MOVEPARKOLO    :
txtFieldLogger.setText("Beállás a parkolóba.               ");
break;
                default                            :
txtFieldLogger.setText("A targoca autómatikus irányítás alatt van.");
break;
            } //switch_vége
      } //NumberRecievedEvent_vége
```

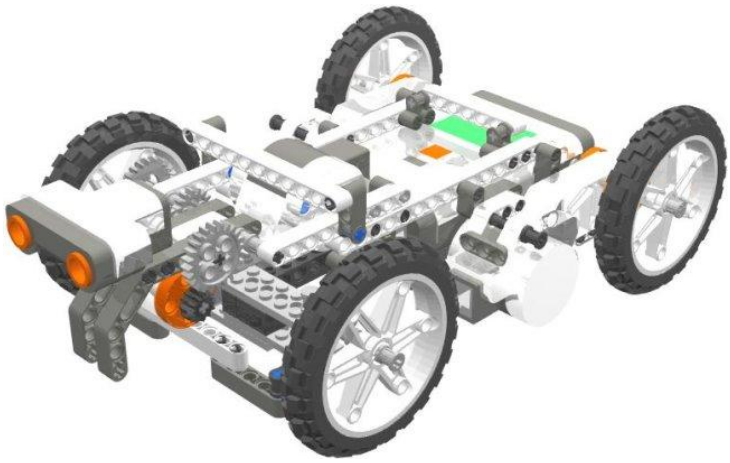
FŐBB FELMERÜLŐ PROBLÉMÁK ÉS MEGOLDÁSUK

ÉPÍTÉS

Az interneten keresztül nagyon sok leírást, útmutatást lehet fellelni. Amelyekből különböző robotokat lehet össze építeni lépésről-lépésre. Mind a szállító, mind a válogató robot építése esetében egy előzetes építési útmutatót követtem. Azonban sokszor el kellett térnem az építési útmutatótól. Nem tudtam mindig egy előre rögzített építési sorrendet követni, mert minden helyzet más és más szituációt, problémát vetett fel. Így sokszor mérlegelni kellett az új helyzeteket.

KEREKEK

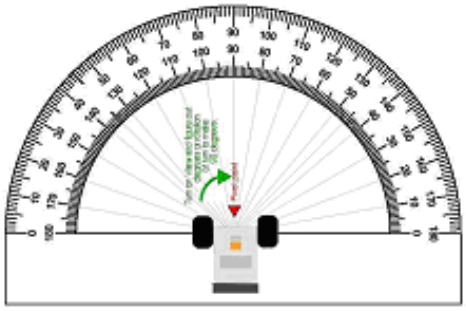
A targonca a kerék alapú járművek nagy csoportjába tartozik. Itt az első fontos kérdés az volt, hogy hány kerékkel is szereljem föl. Ugyanis vannak alapvető szabályok, amik betartása kötelező. Ha négykerekű járművet építünk (27. ábra), akkor tudnunk kell ezekről, hogy nagyobb sebességet képesek elérni, mint ha kevesebb kerékkel szerelnénk fel. Fontos tényező, hogy négy kerék esetében könnyű szerkezetet kell kialakítanunk. Itt külön kormányzási szerkezettel lehet megoldani a fordulást, és így egy körívet írnak le fordulásnál.



27. ábra: Négykerekű NXT (forrás [9.]).

Az összeépített targonca egy másik nagy csoportba a háromkerekűek közzé tartozik. Ezeknél a típusoknál a hajtott kerekek egyben a kormányzott kerekek is. Két kerékhez két

motor fog kelleni. Ezen esetben, ha egy irányba forognak a kerekek, akkor előre illetve hátrafelé haladhat a járművünk. Ha ellentétes irányba fordítjuk el a motorokat, akkor így el tudjuk fordítani a hajtott tengely középpontja körül a targoncát (28. ábra). Ez az indok is hozzájárult, hogy háromkerekű szállításra alkalmas legót építettem, mert a vonalkövetés szempontjából ez a legkiválóbb. Az is igaz, hogy általában a háromkerekű járműveknél nem az első kerekeket kormányozzák, hanem az előbb említett módon, a közös tengelyen lévő kerekeket mozgatják.



28. ábra: Az NXT forgása (forrás [9.])

29. ábra: LEGO-ból megépített Segway (forrás: [9.])

Végül megemlíteném a harmadik nagy csoportot, melyeket a kétkerekű robotok alkotják. Ezen a képen látható, amint az NXT készletből egy segway imitátor is kirakható (29. ábra), és működése szimulálható. A segway egy olyan önegyensúlyozó robot, amelyet a közlekedésben használnak. Melynek egy lehetséges megoldása, amikor giroszkóp szenzor figyeli az eldőlést.

A kép csalóka, mivel nem a szenzor támasztja meg a lego-t!

VÁLOGATÓ ROBOT MEGKÖZELÍTÉSE

A targonca képes egyenes vonalú mozgás közben megközelíteni a megemelendő terhet. Első verzióban az erre szerelt ultrahangos szenzor segítségével oldottam meg, hogy ha érzékelte a megemelendő rakományt, akkor vegye fel azt a villájára. De ezt később elvetettem, mivel nem minden esetben pont ugyan ott állt meg a robot, ahol a legoptimálisabb

lett volna. Ennek a magyarázata pedig az ultrahangos érzékelő működésével magyarázható, melynek a pontos magyarázatát „a szenzorok bemutatása” című fejezetben ki is fejtettem. Valamint ennek a szenzornak a jellemzése a későbbiekben még olvasható lesz.

Miután megtörtént a válogatás, akkor kerül rá a vezérlés a szállító robotra. Amely érte megy a rekeszekért, melyekben a golyók vannak, megfordul és elszállítja a raktárba, a vonalat követve. Majd a fordulás után visszatér a kezdő kiinduló pontba, amit felfoghatunk úgy, mint parkoló területet, ahol a munkagépek helyezkednek el.

Fontos dolog lehet, hogy az elem csere ezen eszköz esetében kissé körülményes, hiszen a tégla köré lett építve a targonca váza, így az elemcsere során meglehetősen szét kellett szereljem a robotot darabjaira.

FÉNYSZENZOR

A fényszenzor használatával nagyon sok feladat oldható meg. Ilyen lehet például egy felrajzolt vonalon történő vonalkövetés, vagy épp elemek szelektálása. A következő fejezetekben a fényszenzor alkalmazhatóságát fejtem ki, és a használata közben adódott problémák egy lehetséges megoldására nyújtok útmutatást.

VONALKÖVETÉS

Vonalkövetés szempontjából a targoncára fel kellett szereljek két fényszenzort, mivel ezek segítségével oldottam meg a vonalkövetést. Egy érzékelővel is megoldható a vonalkövetés. De úgy gondoltam, hogy a gyorsabb és folyamatosabb mozgás eléréséhez jobb, ha két szenzort használok. Egy szenzor használata esetében a megoldás az lehetett volna, ha a követendő vonal fölött helyezkedik el a szenzor. Ha az érzékelő letér erről a színről, akkor korrigálásra van szükség. Esetemben az előbb említett megoldás ellentétje valósult meg. A két szenzort egymástól 4-5 cm-re helyeztem el azonos tengelyen, a talajtól mért 2-3 cm távolságban, mivel ez volt a legideálisabb, amit ajánlott a fényérzékelő gyártója. Ez viszont néhány problémát idézett elő. Az első volt, hogy a targonca emelőszerkezetét előrébb kellett helyezni, mert a szenzorok miatt nem fért már el. Majd ezen probléma megoldása, idézte elő azt a szituációt, ami miatt kellett a targoncára ellensúlyt felszerelni. Ezzel valósult meg a targonca kiegyensúlyozása. Ugyanis, ha túl elől van a súlypont, abból az következik, hogy a

hírtelen megállásoknál a targonca belibegett, a legrosszabb eset pedig az volt, amikor előre is bukott. A lesúlyozott hátsó rész miatt került hátrébb a súlypont.

A targoncaemelő villáját is, úgy kellett megépíteni, hogy az képes legyen egy általam tervezett kis rekesznek a felemelésére, amelyekbe előzőleg bele kerültek a kiválogatott elemek. Fontos dolog, hogy nem csak arra kellett oda figyelni, hogy alá nyúljon és felemelje, hanem arra is, hogy ne csússzon majd le a rekesz a forgolódás és megállás során.

A válogató robot építése során is problémák merültek fel. Mégpedig az, hogy a szenzort megfelelő távolságra, és megfelelő beesési szöggel kellett elhelyezni a válogató elemek fölött.

EGYENES VONALÚ EGYENLETES MOZGÁS

Az első próbálkozásaimkor a korrigálást a vonalon úgy próbáltam megoldani, hogy a motornak egy blokkoló utasítással adtam meg, hogy mennyit javítson a mozgásán, de ez azért volt hibás, mert ilyenkor a robot mozgása darabossá vált. A szaggatott mozgás pedig bizonytalanná tette a szállítást, félő volt, hogy leesik a kosár. Tehát más megoldást kellett kitalálni, míg végül arra jutottam, hogy elég, ha csak a motorok forgási sebességét állítom át. Mert ebben az esetben a motorok nem blokkolva, hanem lassulva állnak le. Ez a következő példában is látszani fog, ha a jobb szenzor érzékeli a fekete színt, akkor az ezzel azonos oldalon lévő motor forgási sebességét leállítom nullára, az ellenkező oldalon lévő motor sebességét pedig megnövelem akkorára, hogy jól el tudjon fordulni a vonal mentén akár élesebb szögben is. Ennek a megvalósítását a következő kódrészlet mutatja be.

```
else if (cs_jobb.getColor() == szin)
        {
              cMotor.setSpeed(0);
              aMotor.setSpeed(470);
        }
else if (cs_bal.getColor() == szin)
        {
              cMotor.setSpeed(470);
              aMotor.setSpeed(0);
        }
```

A GOLYÓK SZÉTVÁLOGATÁSA

A válogató részben az alap probléma ott kezdődött, hogy az Mindstorms NXT-hez adtak egy színérzékelőt. De sajnos ezt nem lehet Java alatt használni, mivel ennek a szenzornak a használatát még nem írták bele a LeJOS legújabb verziójába. Én a szakdolgozatom készítésekor a 0.8beta verzió számmal ellátott szoftvert használtam. Így vissza kellett nyúlni az első NXT csomaghoz, melyben fényérzékelő van. Ezzel lehetett megoldani a színérzékelést Java programozási nyelv alatt. A fényszenzorról köztudott, hogy erősen fényviszony függő. Más értékeket mutatott ugyanarra a színre reggel, délben és éjszaka. Én viszont olyan programot szerettem volna írni, melyet fényviszonyoktól nem függve lehet alkalmazni a nap bármely pillanatában.

Ezt sikerült is megvalósítanom, melynek a működési váza a „válogató robot” fejezetben van kifejtve. Természetesen külön megoldást kellett találni arra, hogy ugyanolyan színű golyót is mérhet különböző értékkel a szenzor. Erre konkrét példa, egy sárga golyónak a fénymélysége lehet 48 értéknyi, de akár 47 vagy 49 is, ugyanazon fényintenzitások mellett. Erre oda kellett figyelni, hogy ilyen eltérések esetén is ugyanolyan színű golyóként kezelje le a válogató robot.

```
else if( Math.abs( colorSensorValue - szin1 ) <= 2 )
{
        sorter.BalraValogat();
        darab = darab - 1;
 }
 else if( Math.abs( colorSensorValue - szin2 ) <= 2 )
 {
        sorter.JobbraValogat();
        darab = darab - 1;
 }
 public void JobbraValogat()
 {
 try
 {
        Thread.sleep(750);
        bMotor.rotate(-50);
        Thread.sleep(750);
        aMotor.rotate(-72);
```

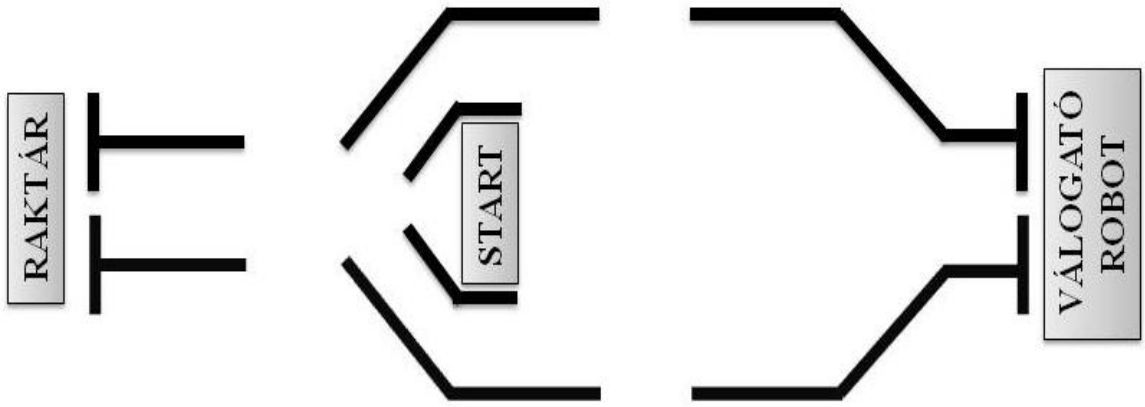
```
        Thread.sleep(750);
        aMotor.rotate(72);
        Thread.sleep(750);
        bMotor.rotate(50);
        Thread.sleep(750);
    }
    catch (InterruptedException e)
        {
        }
    } //jobbraválogat_vége

    public void BalraValogat()
    {
    try
    {
        Thread.sleep(750);
        bMotor.rotate(50);
        Thread.sleep(750);
        aMotor.rotate(-72);
        Thread.sleep(750);
        aMotor.rotate(72);
        Thread.sleep(750);
        bMotor.rotate(-50);
        Thread.sleep(750);
    }
    catch (InterruptedException e)
        {
        }
    } //balraválogat_vége
```

A PÁLYA MEGTERVEZÉSE

Az általam tervezett pálya meglehetősen sok időt és gondolkodást igényelt (30. ábra). Hiszen nem tudtam megoldani azt, hogy a targoncán lévő fényszenzor több ellentétes színű vonalat is tudjon követni. Az alap probléma az volt, hogy a targoncára felszerelt fényszenzor más-más fényviszonyok között más és más értéket adott vissza ugyan arra a színre. Mely abból is adódott, hogy a targonca folyamatos mozgásban volt, és a szenzor már nagyon kicsi

értékváltozásra is reagált, ilyen volt például, mikor érzékelte saját árnyékát. Próbálkoztam piros, fekete és sárga vonalak követésével is. De a szenzor alapjában véve csak a fekete és a fehér színt tudta elkülöníteni egymástól folyamatos mozgás esetén, mert ezek értékei már elég nagyok voltak ahhoz, hogy jól el lehessen különíteni egymástól. A többi szín esetében sokszor előfordult, hogy egyes színeket rosszul értelmezett. Ez abból kifolyólag állt elő, hogy a szenzor a fekete színre 25-30 közötti értéket adott vissza a 100-as skálán. A fehérre pedig olyan 45 körüli értéket látott. Így az összes többi szín e két érték közötti tartományból vehette volna fel az értéket, de túl gyakran fordult elő, hogy árnyékban a piros vonalra fekete tartományú értékeket látott. Ezen okok miatt kellett kizárnom az összes szint, és csak a fehérre és feketére koncentrálni. Így alakult ki az a helyzet, hogy a pályát fehér kartonlapokból ragasztottam össze, és a vonal pedig fekete ragasztószalag lett.

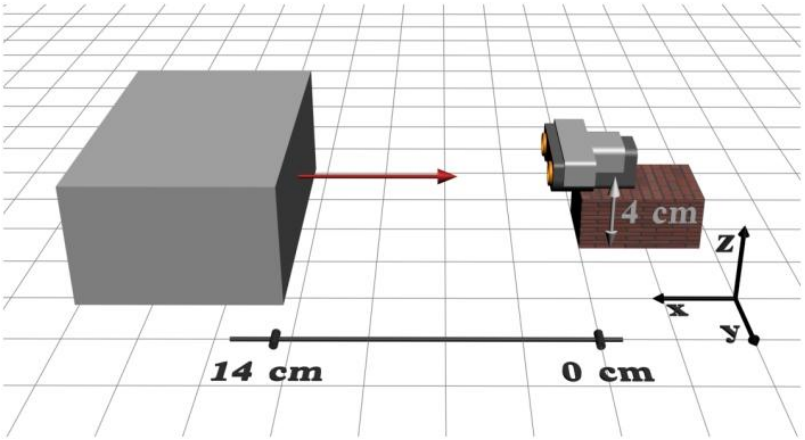
A pálya alakját tekintve is elég különlegesnek mondható, hisz meg kellett azt oldani, hogy amikor két vonal keresztezi egymást, akkor mikor melyik irányba forduljon el a szállító járművünk. Hosszas gondolkodás után jutottam arra az egyszerű ötletre, hogy a kereszteződések pontjában nem ragasztok fekete vonalat a papírra. Így képessé vált a robot, hogy áttérjen egy másik útvonalra, anélkül, hogy mesterséges intelligenciát használtam volna a programkódba. Ezt az alábbi kép mutatja.



30. ábra: Az általam tervezett pálya (általam készített ábra).

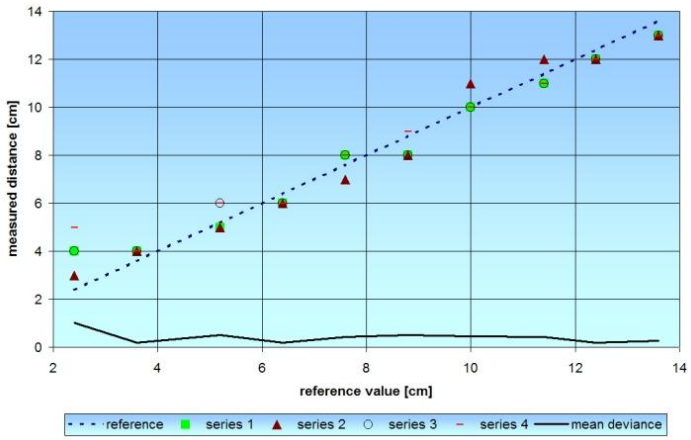
TÁVOLSÁGÉRZÉKELŐ

A távolságérzékelő szenzor 2-3 %-os tévedési aránnyal képes működni. E mellett fontos tényező, hogy megfelelő mennyiségű ultrahang jelnek kell vissza verődnie az objektumról, hogy pontos értékkel tudjon szolgálni a felhasználó számára. Tesztelések során több probléma is akadt ezzel a szenzorral. Az egyik legjelentősebb, hogy legoptimálisabb esetben síkfelületű objektumra van szükség, ha szeretnék elérni a pontos mérést. A következőkben bemutatásra kerül e szenzornak a pontos vizsgálata. Az első kép mutatja a szenzor telepítési viszonyát egy síkfelülettel rendelkező objektumhoz képest (31. ábra).



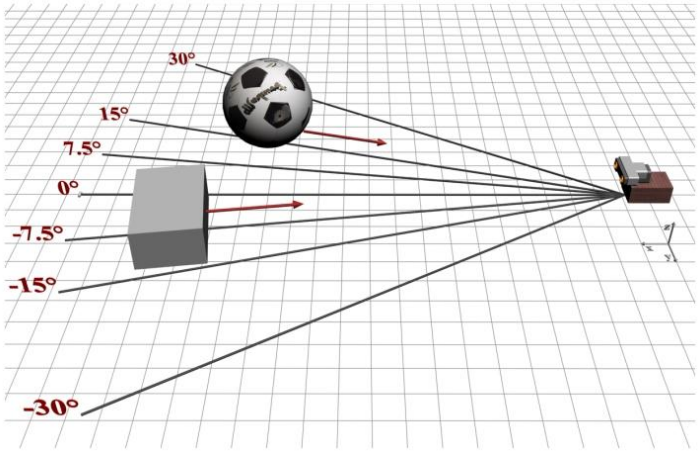
31. ábra: Az ultrahangos szenzor telepítési helyzete (forrás [21.]).

A következő kép fogja mutatni azt a derivált értékeket az idő függvényében, amint egy 14 cm x 9 cm x 6cm-es dobozhoz közeledik az ultrahangos szenzor. A képről látszik, hogy közel 3cm-nél kevesebbet nem tud a szenzor mérni (32. ábra).



32. ábra: Az ultrahangos szenzor által mért értékek deriváltjai (forrás [21.]).

A következő kép azt mutatja be, hogy mekkora a szenzor látási mezője. Ebből kiderül, hogy mindössze 60° az a horizontális mező, melyben képes az érzékelésre. Szemben az emberével, amely 180° (33. ábra).



33. ábra: Az ultrahangos szenzor látószöge (forrás [21.]).

Az eredmények kimutatják, hogy az ultrahangos érzékelőt mindig vízszintes helyzetbe kellene tenni, annak érdekében, hogy jó eredményeket adjon vissza. Különféle mérések, tesztek alapján rájöttem, hogy a szenzoron lévő bal szem, mintha kissé vak lenne. Ennek az a magyarázata, hogy a bal szem az ultrahangos hullám vevő készüléke, míg a jobb szem a küldő szerepét látja el.

MEGFELELŐ TÁPELLÁTOTTSÁG

Fontos dolog külön kiemelni, hogy ha nincs megfelelő tápellátottsága az NXT-nek, akkor olyan problémák merülhetnek fel, amelyeket nem biztos, hogy értünk. Tapasztalataim alapján észrevettem, hogy bármit is teszek a robottal, még sem működik megfelelően. Erre példa, hogy nem engedte a programot feltölteni a memóriájába, vagy egyszerűen programindításnál nem volt hajlandó működni. Ez abból adódott, hogy nem teljesen helyt álló az a kis ikon az LCD kijelzőn, amely jelzi számunkra, hogy mennyire van feltöltve az elemünk, vagy éppen az akkumulátorunk. Esetemben még szinte félig jelezte a töltöttségi szintet, de valójában már nem volt elég feszültség arra, hogy a már említett programot feltöltse, vagy megmozgassa a motorokat.

ÁLTALÁNOS TAPASZTALATOK

Az alábbi lista az NXT-vel kapcsolatos tapasztalatok összessége, szemben az egyetemi kutatásokhoz vagy mérnöki játszadozáshoz használható egyéb robotokkal.

- Az NXT a forgalomban lévő LEGO készletek közül az egyik legdrágább, de a valódi robotot tartalmazó környezetek közül az egyik legolcsóbb.
- Előnye a flexibilitás, azaz könnyen és gyorsan átalakíthatóak a számunkra kívánt szerkezetté.
- Nagyon összetett robot állítható elő a szenzorok használatával.
- Szenzorok valódi robotokhoz képest elég egyszerűek és kis érzékelési tartománnyal bírnak.
- Alapkészletben kevés a szenzor, ugyanis egyes feladatok elvégzéséhez több érzékelőre lenne szükség. De ezt kompenzálták azzal, hogy továbbiakat tudunk vásárolni, igaz relatívan magas árral rendelkeznek ezen eszközök.
- A motorok számának korlátozása a növekvő energiaigény miatt is ésszerű, de ezzel sajnos túl kevés szabadságfok biztosítható.
- A beépített robotok építése jól dokumentált, építésük viszonylag gyorsan halad.
- Pozitív dolog, hogy a fiatalabb korosztályra is gondoltak. E miatt is szokás a középiskolákban oktatni ezekkel a termékekkel. Esetekben még korábbi korosztály számára.
- A fenti két megjegyzésből következően a fejlesztés hardver/szoftver aránya így nagyon a hardver felé tolódik el. Komolyabb programfejlesztéshez valószínűleg célszerű megépíteni egy az igényeknek megfelelő robotot, majd sokféle kísérletet elvégezni rajta különféle programok, vezérlések megírásával.
- A példarobotok rendkívül elegáns építmények, látszik rajtuk, hogy szakértők tervezték őket. Megközelítően jó struktúrájú, funkcionálisan megfelelő saját robot tervezéséhez sok gyakorlás szükséges.
- A hardver gyenge pontja a kábelezés. 3 motor és 4 szenzor bekötését kell megoldani úgy, hogy a kábelek ne akadályozzák egymást és a robot mozgását sem. Ez egyáltalán nem egyszerű. Különösen komoly problémát jelenthet, ha egy szenzor vagy egy motor a robot mozgó részén helyezkedik el, mivel így a kábelnek is mozognia kell. Azt sem

szabad figyelmen kívül hagynunk, hogy maguk a kábelek elég merevek, amelyek tömör drótszálból állnak.

- A szenzorok, ahogy az a robotikában megszokott, igen érzékenyek a környezetre. Egy fénymérés eltérő eredményt ad napos és felhős időben, távolságmérés eredménye változik az érzékelt felület anyagi minőségétől függően. Erre a problémára általános megoldást adni a robotika legnagyobb kihívásai közé tartozik.
- 6 ceruzaelemre - inkább akkura - van szükség a működéshez. Ezek intenzív használattal, vélhetően a robot nagy súlya miatt elég hamar lemerülnek. Ez még önmagában nem lenne baj, de az már probléma, hogy az energiafogyás kijelzése nem az igazi, mivel nagyon hirtelen válik szinte működésképtelenné az NXT. Ezért két kísérletezési alkalom között célszerű mindig tölteni.

ÖSSZEFOGLALÁS

Szakdolgozatom célkitűzése az volt, hogy egy az iparban is használt konkrét anyagmozgatási folyamatot modellezzem, az informatika által nyújtott támogatás segítségével. Ezt a célkitűzést véleményem szerint sikerült teljes mértékben megvalósítanom a LEGO Mindstorms NXT egységek alkalmazásával.

Projektemben három LEGO Mindstorms NXT készletből építkeztem és ezekre az általam épített eszközökre írtam külön programot, amelyek összehangolva is képesek az együttműködésre. Az első készletből egy válogató robotot építettem meg, mely képes különböző színű golyókat szétválogatni, és e golyókat más-más kosárba pakolni, miközben folyamatos kapcsolatban áll a számítógéppel. A második csomagból egy targoncát építettem, amely szállítási és anyagmozgatási feladatokat lát el a válogató robot és a raktár között. A válogató robot, a szelektálás után utasítja a targoncát a szállításra. Amelynek oda kell találnia a rekeszekhez, felvenni azokat, majd az elszállítás után a raktárban letenni. Közben pedig figyeli, ha valami akadály kerül elé, akkor megálljon. A szállítási folyamatok végeztével, a start helyre kell visszaállnia, hogy elölről tudjon kezdődni az egész folyamat, ha ez szükséges. Végül a harmadik egységből egy távirányító készüléket építettem, mely segítségével manuálisan tudtam vezérelni a targonca műveleteit, szimulálva ezzel egyfajta üzemzavart, amikor is kézi vezérlésre kell átállni.

Összességében sikerült megvalósítanom egy teljesen ipari automatikus rendszert. Melynek működést többször teszteltem különleges körülmények között is, és a modell kifogástalanul működött.

A dolgozatomban próbáltam arra rávilágítani, hogy milyen problémákkal szembesültem, és hogy ezekre milyen megoldásokat találtam. Valamint próbáltam információt adni a robotok világáról, hogy hol is tartunk jelenleg, és hogy milyen irányba fejlődnek tovább ezeket a szerkezetek.

A mechatronikában jelentős szerephez jut a robotokkal való megismerkedés robotok, robotkarok, manipulátorok programozása. Az általam, használt és kipróbált LEGO Mindstorms NXT készletről azt a következtetést vontam le, hogy igen használható ez a komplett rendszer és érdemes kihasználni adottságit, mert egyben nem csak szórakoztatott, de

el is gondolkodtatott bizonyos problémákról. Ezzel a hardver készlettel lehetőségem adódott komplex bonyolultabb programozási algoritmusok írására, megfizethető áron.

Tapasztalataim szerint a mai világban a robotok nagyon fontos feladatokat látnak el, és segítik, valamint helyenként felváltják az emberi munkálatokat. Széles körben, a világ minden pontjában használják és egyre gyorsabban fejlődő ágazatokról van szó. Egyre több multinacionális cég foglalkozik mobil és háztartási célú robotok készítésével. Gondoljunk csak a takarító robotokra, önjáró fűnyírókra, vagy Sony cég Aibo szórakoztató robot kutyájára. Azonban ne feledkezzünk el arról, hogy még úgymond „gyerekcipőben" jár ez az ágazat. A tengerentúlon komoly fejlesztések folynak a hadipar támogatásával, például aknakereső és felderítő robotok alkalmazására.

Úgy gondolom, ha valaki ezzel a témakörrel foglalkozik, biztos magával fogja ragadni a lehetőségek és ötletek tárháza, amit a robotok képesek végrehajtani valós időben. Személy szerint garantálni tudom, hogy sikerélményekben gazdag lesz annak a személynek az élete, aki ilyen témában dolgozik

Remélem sikerült felhívnom mindazok figyelmét, akik ilyen témával szeretnének foglalkozni, hogy mennyire sok lehetőség nyílik a programozó előtt, ha a LEGO Mindstorms készletével dolgozik.

IRODALOMJEGYZÉK

[1.] Čapek Karel. (1921). *Rossum Univerzális Robotjai.*

[2.] America, Institute of America. (1979). *Definition of a 'Robot'.*

[3.] Szótár, Webster. (1993). *A robot definiciója.*

[4.] FAQ Robotics. *A robot definiciója.*

[5.] Asimov Isac. (1950). *Én a robot.* Gnome Press.

[6.] Husi Géza. (2009). *Ipari robotok oktatási segédanyag.* Debreceni Egyetem – Műszaki Kar.

[7.] Husi Géza, Darai Gyula. (2008). *Building Services, Mechanical and Building Industry days.* Debreceni Egyetem – Műszaki Kar.

[8.] Tóth János. (2009). *Building Services, Mechanical and Building Industry days.* Debreceni Egyetem – Műszaki Kar.

[9.] Ismeretlen szerző. (2008). *Robotolj te is!* NextWork Robotépítő Verseny – dokumentáció.
http://www.nextwork.hu/files/imce/nxt_teljes.pdf,
letöltve: 2009. augusztus 30.

[10.] Husi Géza. (2009). *Ipari robotok oktatási segédanyag.* Debreceni Egyetem – Műszaki Kar.

[11.] Ismeretlen szerző. (2009). *Robotika és szimuláció.*
http://www.freeweb.hu/jataka/rics/speci/src/class1/intro.html,
letöltve: 2009. augusztus 20.

[12.] Ismeretlen szerző. LEGO Mindstorms NXT 2.0.
http://hertenberger.co.za/2009/07/09/lego-mindstorms-nxt-2-0/,
letöltve: 2009. szeptember 10.

[13.] Ismeretlen szerző. (2008). *Az FLL versenyről.*
http://sagv.gyakg.u-szeged.hu/fll/fllism.htm,
letöltve: 2009. október 20.

[14.] SG.hu. (2008). *NextWork robotépítő verseny LEGO alapon*
http://www.sg.hu/cikkek/58725/nextwork_robotepito_verseny_lego_alapon,
letöltve: 2009. október 20.

[15.] Ismeretlen szerző. (2007). *RobotC.net.*
http://www.robotc.net/,
letöltve: 2009. október 30.

[16.] Ismeretlen szerző. *Black boksz rendszer.*
https://miau.gau.hu/mediawiki/index.php/Black_box_rendszer,
letöltve: 2009. október 30.

[17.] WikipédiA.
http://hu.wikipedia.org/wiki/Java_Development_Kit,
letöltve: 2009. október 30.

[18.] http://sourceforge.net/projects/lejos/files/lejos-NXJ-win32/

[19.] Prof. Dr. Cselényi József, *A targoncás anyagmozgatás számítógépes irányításának lehetőségei I. ,* Miskolci Egyetem, Anyagmozgatási és Logisztika Tanszék, http://www.pointernet.pds.hu/ujsagok/transpack/2003-ev/03-jan-feb/tra-08.html,
letöltve: 2009. október 10.

[20.] WikipédiA.
http://hu.wikipedia.org/wiki/Bluetooth

[21.] Ismeretlen szerző. *LEGO Mindstorms – Ultrasonic Sensor.*
http://www.tik.ee.ethz.ch/tik/education/lectures/PPS/mindstorms/sa_nxt/index.php?page=tests_us
letöltve: 2009. október 3.

FELHASZNÁLT IRODALOM

- **Dr. Nagy Géza,** *Logisztika jegyzete,* Debreceni Egyetem–Műszaki Kar

- **LEGO Mindstorms NXT bemutatkozik**
  http://mindstorms.lego.com/

- **Építéshez ötletek, instrukciók**
  http://www.nxtprograms.com/

- **LeJOS Tutoriál**
  http://lejos.sourceforge.net/nxt/nxj/tutorial/index.htm
  http://www.bartneck.de/2008/03/04/java-lego-nxt-eclipse-tutorial/

- **LeJOS**
  http://lejos.sourceforge.net/index.php
  http://sourceforge.net/projects/lejos/files/lejos-NXJ-win32/

- **LeJOS API**
  http://lejos.sourceforge.net/nxt/nxj/api/index.html

- **Eclipse fejlesztői környezet**
  http://www.eclipse.org/downloads/

- **A LEGO NXT híroldala**
  http://mindstorms.lego.com/NXTLOG/default.aspx

- **NXT-vel kapcsolatos információk blogja**
  http://thenxtstep.blogspot.com/

- **LEGO webáruház**
  http://www.legoaruhaz.hu/

- **Oktatási segédanyag**
  http://www.nextwork.hu/tutorial,
  letöltve: 2009. szeptember 04.

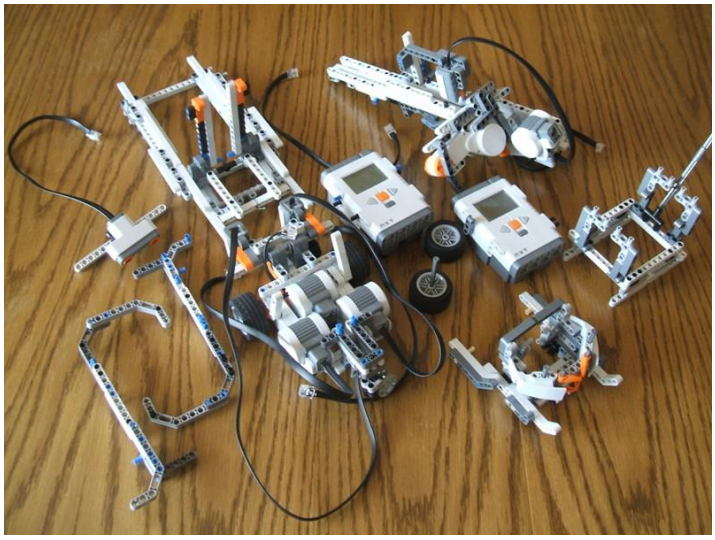
- **Osztályhierarchia**
  http://www.legorobotik.ch/NXTautonom/docs/overview-tree.html

- **Krámli György,** *Szenzorika tanfolyami jegyzete,* FESTO
  http://www.kekvilag.hu/didactic/letoltes/oktatas/Szenzorika_jegyzet.pdf,
  letöltve: 2009. október 30.

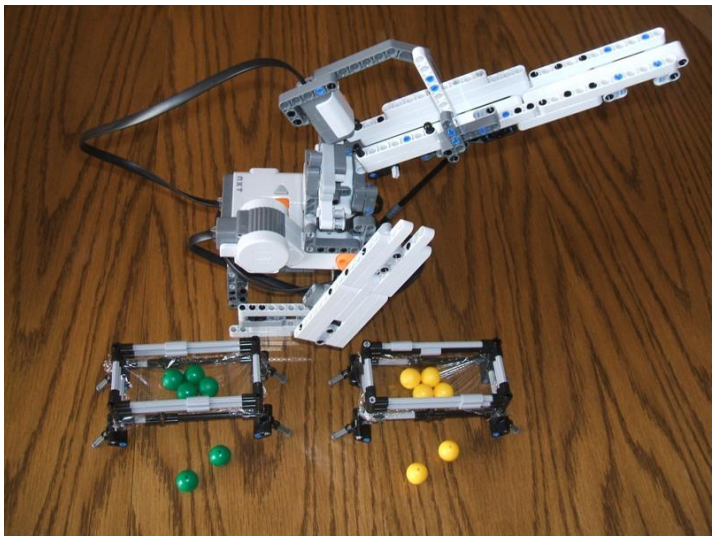
- **Bluetooth használata**
  http://hu.wikipedia.org/wiki/Bluetooth
  http://www.docstoc.com/docs/2133860/The-leJOS-NXJ-Tutorial,
  letöltve: 2009. október 2.

- **Angster Erzsébet,** (2003). *Objektumorientált tervezés és programozás JAVA I.-II.*

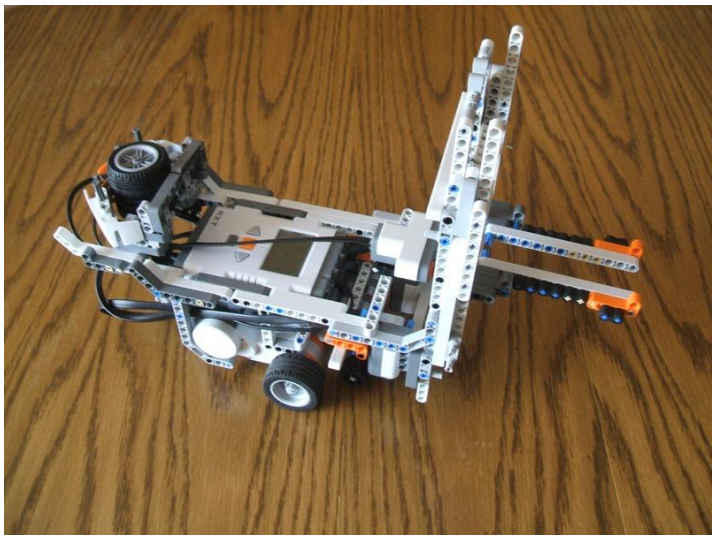
FÜGGELÉK



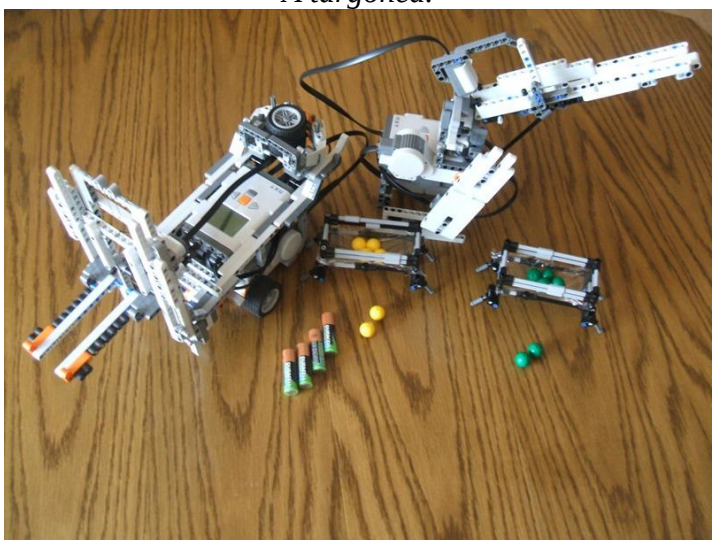
A LEGO robotok kissé szétszerelt állapotban.



A válogató robot.



A targonca.



A targonca és a válogató robot.

KÖSZÖNETNYILVÁNÍTÁS

Köszönetet szeretnék mondani Dr. Husi Gézának, a Debreceni Egyetem Műszaki Főiskolai Kar tanszékvezetői docensének a LEGO Mindstorms NXT 2.0 csomag használatához nyújtott segítségéért, tanácsaiért és ötleteiért. Valamint, hogy lehetőséget teremtett, hogy szakdolgozatom munkáját bemutathassam a Kutatók Éjszakáján. Amelynek célja, hogy bemutassa a kutatók hétköznapi arcát és a kutatási eredményeiket is közérthető, kipróbálható formában, ezáltal népszerűsítve a kutatói életpályát és a tudományt, elsősorban a fiatalok körében. Továbbá lehetőségem nyílt a XV. Épületgépészeti, Gépészeti és Építőipari Szakmai Napok keretében egy nemzetközi konferencia és kiállítás részvételében. Ahol bemutathattam a szakdolgozatom gyakorlati oldalát is.

Valamint külső konzulensemnek, Lugosi Péternek, a National Instruments tesztmérnökének a hasznos instrukcióiért, akinek hozzáállása, segítőkészsége és hasznos észrevételei nagyban segítette a szakdolgozatom munkálatait.