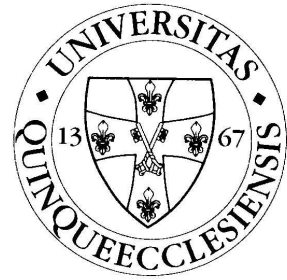


Pécsi Tudományegyetem

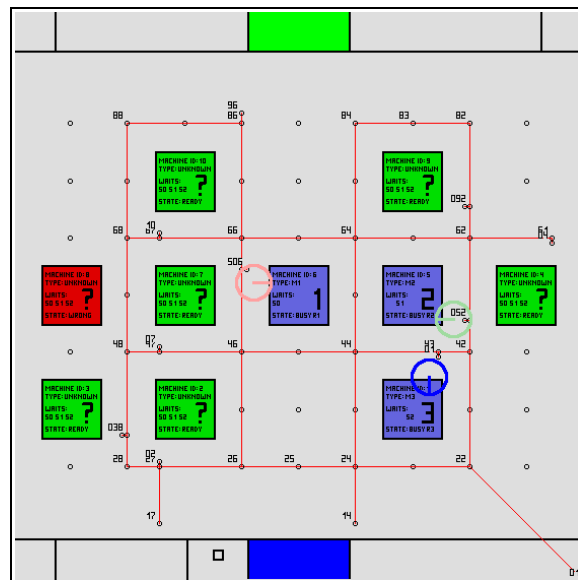
Pollack Mihály Műszaki és

Informatikai Kar

Automatizálási tanszék



Virtuális valóság alapú szimulációs környezet fejlesztése autonóm intelligens robotokhoz



Készítette: **Malkó Tibor**

villamosmérnök hallgató

Konzulens: Megyeri Péter

Tanszékvezető: Helmich József

Pécs

2012.

Tartalomjegyzék

1.Bevezetés és célkitűzések.....	3
2.Általános áttekintés.....	4
2.1.Robotino felépítése, általános hardverismertető.....	4
2.2.Robotino hajtása, motor objektum.....	5
2.3.Robotino Sim ismertetése és hiányosságai.....	6
2.4.Virtuális valóság alapú szimulációs környezet.....	7
3.VRE - Virtuális Robot Emulátor.....	7
3.1.Tervezés.....	7
a)Matematikai modell.....	7
b)Virtuális Robot Emulátor tulajdonságai.....	9
3.2.Implementálás.....	10
a)Alap Blokkokból felépített Virtuális Robot.....	10
b)Lua Script Virtuális Robot.....	11
c)Virtuális Robot Blokk.....	12
d)Az elvégzett kísérletek.....	15
4.Vizualizációs környezet.....	18
4.1.Vizuális környezet kiválasztása.....	18
4.2.Vizualizáció alapjai.....	19
4.3.Virtuális tér megalkotása.....	20
5.Tesztelés.....	21
5.1.Robotino programozás.....	21
a)Általános Robotino programozás.....	21
b)Pollino specifikus Robotino programozás.....	23
5.2.VRE használata és programba illesztése.....	23
6.Eredmények.....	24
6.1.Alkalmazási lehetőségek.....	24
6.2.Továbbfejlesztés.....	24
7.Értékelés és következtetések.....	26
8.Köszönetnyilvánítás.....	27
9.Melléklet.....	27
10.Irodalomjegyzék.....	30
11.Nyilatkozat.....	30

1. Bevezetés és célkitűzések

A mai fejlődő világban egyre nagyobb szerepet kapnak az ipari robotok. Az elsőt az 1960-as évek elején állították munkába, ekkor még önálló egységként, egy feladat elvégzésére volt képes. A technika fejlődésével egyre bonyolultabb felépítésű, nagyobb tudású és pontosabb robotok álltak munkába. Ezek már komplex feladatok elvégzésére is alkalmasak voltak. Közös jellemzőjük a nagy tömeg, magas ár mellett a helyhez, vagy pályához kötöttség volt. Ezek a gépek akár egymás után is képesek voltak egy folyamaton belül több részfeladatot elvégezni. Napjainkban már ott tart a robottechnika, hogy egyre több helyen szerepet kapnak a helyváltoztatásra képes robotok:

- intelligens raktár
- katasztrófa felderítés
- robbanásveszélyes környezet
- robbanószerkezet hatástalanítás
- sugárszennyezett környezet

Célom egy olyan sokoldalúan használható környezet kifejlesztése volt, ahol az ilyen jellegű feladatokat lehet szimulálni. Számos szempontot figyelembe kellett venni a tervezés során, továbbá az alábbi elvárásoknak kellett megfelelni:

- mindenki számára elérhető, ingyenes rendszer
- könnyű kezelhetőség
- széleskörű alkalmazhatóság
- oktatásban felhasználható
- ideális és reális robot emulálás

A választás egy ingyenesen használható mobilrobot programozó rendszerre esett. A FESTO Robotino View program, Robotino-k programozásához használható. Ilyen robotokkal számos középiskolában és egyetemen is találkozhatunk, mivel ez egy oktatási és kutatási célokra kifejlesztett mobil robot.

2. Általános áttekintés

Mielőtt nekiláthatnánk az emulátor és a környezet megtervezésének meg kell ismerni a robotokat.

2.1. Robotino felépítése, általános hardverismertető



1. ábra. Robotino

Mi a Robotino és miből áll? A 3 szabadsági fokú robot két fő részre bontható: az alsó felében találhatóak a mechanikus alkatrészek, a távolságérzékelő szenzorok és a tápellátást biztosító akkumulátorok. A felső részben található a control board, valamint egy ipari pc, a kezelőszervek egy lcd kijelzővel és egy vezeték nélküli router. Valójában ez egy omnidirekcionális hajtással felszerelt robot, mely ennek a

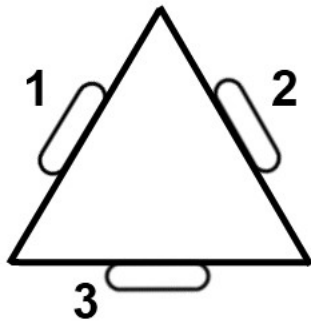
rendszernek köszönhetően szabadon tud mozogni a tér 2 dimenziójában, tehát két szabadsági fok a pozíciót és a harmadik az orientációt mutatja meg. [1]

A mozgatót megnehezíti kicsit hogy két külön koordináta rendszerrel kell dolgozni. A bázis rendszer ami a pályához van rögzítve és a robot koordináta rendszere amiben a mozgásokat adhatjuk meg. A bázis koordináta rendszert a szokásos koordináta rendszer 90 fokos elforgatásával kapjuk. Függőlegesen felfelé kerül az X tengely és vízszintesen balra az Y tengely. Ez a tengelyelrendezés a robot felépítéséből adódik, azért hogy ne kelljen a pc-jének felesleges transzformációkat elvégeznie futás közben. Ezzel nem is lenne még probléma, viszont a robot mozgása, a robothoz rögzített koordináta rendszerben van megadva, tehát ha a bázis koordináta rendszerben szeretne elmozdulni, akkor az elmozdulás vektort el kell forgatni a robot orientációjával, hogy az a megfelelő irányba mozduljon.[2]

A kommunikációhoz és külső egységek csatlakoztatására rendelkezésre állnak:

- USB, Ethernet, Analóg be-, Digitális be- és kimeneti portok
- 1 Aktuátor kimenet encoder bemenettel

2.2. Robotino hajtása, motor objektum

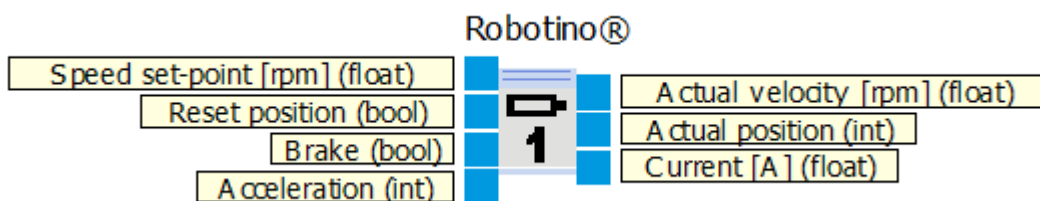


2. ábra. Hajtás elrendezés



3. ábra. Mecanum kerék

A hajtás 3 azonos felépítésű Mecanum¹ kerekekkel szerelt motoregységből áll. A motorok egy bordás szíjas hajtással meghajtanak egy bolygóműves áttételt, és az így megnövelt nyomaték hajtja meg a Mecanum kerekeket. Ezek a kerekek két síkba rendezett speciális görgőkből állnak, melyek csak a forgástengelyre merőlegesen képesek a talajra nyomatékot átvinni, más irányokban szabadon gördülnek. A motorok tengelyén helyet kapott még egy (2000 imp./ford.) encoder tárcsa is, mellyel a motor fordulatszámát és a tengely pozícióját mérhetjük. Ezek a motoregységek 120 fokos szöget zárnak be egymással, ezáltal elérhető, hogy bármely irányba elmozdulhasson, elfordulhasson vagy akár mozgás közben foroghasson is a robot. A motorokat külön-külön is vezérelhetjük, megadhatunk nekik egy fordulatszámot, mely a control boardon megvalósított PID szabályzót vezérli. Ez a vezérlési mód meglehetősen idegen és átláthatatlan, ezért ehelyett egy omnidrive blokk használható. Ezen keresztül a robothoz rögzített Descartes koordináta rendszerben lehet megadni a mozgást és a forgást, melyeket az egy 3 változós függvény segítségével motorfordulatszámokká transzformál.[3]



4. ábra. Robotino motor objektum

Amint az a 4.ábrán is látható, a motorobjektumoknak 3 kimenete van, ezek segítségével lehet lekérdezni a motor áramfelvételét, a tengely aktuális pozícióját és a motor tényleges sebességét. A bemeneteit tekintve megadható a sebesség, ezen kívül

¹ Mecanum, vagy svéd kerék

lenullázhatóak a pozíció számlálók, aktiválhatóak a fékek (ekkor a control boardon található H-híd rövidre zárja a motor kapcsait, ezzel valósítva meg a fékezést), valamint beállítható egy gyorsulás is. Mivel a motor mindkét irányba foroghat, a gyorsulás beállításánál figyelni kell arra, hogy ez tulajdonképpen a sebesség változásának meredekségét adja meg, mely gyorsulásnál és lassulásnál egyaránt hat.

2.3. Robotino Sim ismertetése és hiányosságai

A Robotino szoftvercsomag része egy Robotino Sim nevű szimulációs program, mellyel Robotino nélkül is dolgozhatunk. Ez egy teljesen különálló rendszer mely hálózaton keresztül kommunikál a Robotino View-al. A szimulátorban egy 3 dimenziós virtuális térben lehet a robottal mozogni, azonban apróbb hiányosságai, hátrányai azért vannak:

- nagy erőforrásigény
- motor encodereket nem kezeli
- részletes grafika
- ingyenes verzióban a pálya nem módosítható
- csak egy robotot támogat

Miért problémák ezek? A Robotino hardvere igen szűk keresztmetszet, a benne elhelyezett pc104 processzora mindössze 500MHz-es, ami nem elegendő a Robotino Sim futtatásához.

A kettős koordináta rendszer és a különböző orientációkban elvégzendő azonos feladatoknál folyamatosan koordináta transzformációkat kellene számolni, mely szintén erőforrás igényes. Ez sokkal hatékonyabban, kevesebb erőforrást felhasználva, megoldható egy másodlagos odometria használatával, mely pontosságához elengedhetetlen a motor encoder használata.

A részletes grafikával elveszik az a lehetőség, hogy minden folyamatot egyszerűen nyomon lehessen követni.

Új pályák szerkesztését nem támogatja a szoftver ingyenes verziója.

Autonóm intelligens rendszerek tesztelésénél, szimulálásánál szükség lehet egynél több robotra is.

A tervezendő környezet nem rendelkezhet ezekkel a hiányosságokkal.

2.4. Virtuális valóság alapú szimulációs környezet

Az eddig bemutatott tulajdonságokat összegezve látható, hogy a környezetnek az alábbi kritériumoknak kell megfelelnie:

- kis erőforrásigény
- Robotino Viewban futtatható
- motor encoderek kezelése
- átlátható, egyszerű grafika
- egyszerű pályaszerkesztés
- több robot kezelése

Ezeket figyelembe véve a szimulációs környezet két fő részre bontható. Egyrészt a Virtuális Robot Emulátor, melynek feladata a fizikai robot helyettesítése, valamint a Robot Monitor, ami a megjelenítésért és a robot szenzorainak kezeléséért felel. Ennek a felosztásnak köszönhetően, ha más típusú (jellegű) robotokat szeretnénk szimulálni, akkor az a VRE módosításával egyszerűen megtehető.

3. VRE - Virtuális Robot Emulátor

3.1. Tervezés

a) Matematikai modell

Első lépésként kapcsolatot kell teremteni a motorok forgása és a robot elmozdulása között. Az elmozdulás sebessége és a motorok fordulatszám konverzióját az omnidrive

blokk végzi, az általa szolgáltatott értékeket kell a VRE-nek sebességekké konvertálni és azokból az aktuális pozíciót és orientációt meghatározni. Ezt a leképezést legegyszerűbben projekciós mátrix segítségével lehet megadni, amint az az alábbi kódrészletben is látszik.[4]

```
void Encoder::projectVelocity( float* m1, float* m2, float* m3, float
vx, float vy, float omega, const DriveLayout& layout )

{
    static const double PI = 3.14159265358979323846;

    //Projection matrix
    static const double v0[2] = { -0.5 * sqrt( 3.0 ), 0.5 };
    static const double v1[2] = { 0.0 , -1.0 };
    static const double v2[2] = { 0.5 * sqrt( 3.0 ), 0.5 };

    //Scale omega with the radius of the robot
    double vOmegaScaled = 2.0 * PI * layout.rb * (double)omega / 360.0 ;

    //Convert from mm/s to RPM
    const double k = 60.0 * layout.gear / ( 2.0 * PI * layout.rw );

    //Compute the desired velocity
    *m1 = static_cast<float>( ( v0[0] * (double)vx + v0[1] * (double)vy
+ vOmegaScaled ) * k );
    *m2 = static_cast<float>( ( v1[0] * (double)vx + v1[1] * (double)vy
+ vOmegaScaled ) * k );
    *m3 = static_cast<float>( ( v2[0] * (double)vx + v2[1] * (double)vy
+ vOmegaScaled ) * k );
}
```

Ezt a kódot leegyszerűsítve és a konstansokat behelyettesítve az alábbi összefüggés jön ki, Robotinora specifikálva: [5]

$$1. \quad k = \frac{60 \cdot 16}{2 \cdot \pi \cdot 40}$$

$$2. \quad M1 = (-0,5 \cdot \sqrt{3} \cdot vx + 0,5 \cdot vy + \frac{2 \cdot \pi \cdot 125 \cdot vw}{360}) \cdot k$$

$$3. \quad M2 = (-vy + \frac{2 \cdot \pi \cdot 125 \cdot vw}{360}) \cdot k$$

$$4. \quad M3 = (0,5 \cdot \sqrt{3} \cdot vx + 0,5 \cdot vy + \frac{2 \cdot \pi \cdot 125 \cdot vw}{360}) \cdot k$$

Ebből az egyenletrendszerből kell kifejezni a számunkra fontos v_x , v_y , v_w sebességeket:

$$5. \quad v_x = \frac{M3 - M1}{\sqrt{3} \cdot k} \quad 6. \quad v_y = \frac{M3 - 2 \cdot M2 + M1}{3 \cdot k} \quad 7. \quad v_w = \frac{v_y + M2/k}{\pi/1,44}$$

Ahol: v_x, v_y - x,y irányú sebesség (mm/sec)
 v_w - phi forgási sebesség (fok/sec)
 $M1, M2, M3$ - 1,2,3-as motorok fordulatszáma (RPM)

Az elmozdulások számításához a sebességen kívül a ciklusidőre is szükség van, (ez egy egyszerű rendszeridő lekérdezéssel megoldható) ezek szorzata adja az elmozdulást. Elsőként a robot orientációját kell meghatározni. Az utolsó orientációhoz hozzá kell adni a $(v_w \cdot cycle_time)$ szorzatot:

- 8. $pos_phi += (v_w \cdot cycle_time)$, majd ezt a $[-\pi; \pi]$ tartományra kell vetíteni. Az x,y elmozdulásokat $(v_x \cdot cycle_time)$ és $(v_y \cdot cycle_time)$ a megfelelő orientációval kell a pozícióhoz hozzáadni. A robot mozgásának vizsgálata közben kiderült, hogy ha ezt az integrálást az előző, vagy az aktuális orientációval végezzük akkor egy folyamatosan növekvő hiba keletkezik a rendszerben, ezért az:

$$- 9. \quad ori = (pos_phi - \frac{v_w \cdot cycle_time}{2}) \text{ orientációval kell figyelembe venni.}$$

Ekkor az aktuális pozíciót a :

$$- 10. \quad pos_x += (\cos(ori) \cdot v_x - \sin(ori) \cdot v_y) \cdot cycle_time$$

$$- 11. \quad pos_y += (\sin(ori) \cdot v_x + \cos(ori) \cdot v_y) \cdot cycle_time$$

összefüggésekből kapjuk.

b) Virtuális Robot Emulátor tulajdonságai

A VRE² tervezésekor fő szempont volt az hogy a Robotino View egy olyan blokkal egészüljön ki, amit az teljes értékű robotnak lát. Rendelkeznie kell minden robotspecifikus tulajdonsággal. Magában kell foglalnia a motor objektumokat és az

odometriát is. Rendelkeznie kell tehát:

- M1, M2, M3, odo_X, odo_Y, odo_Phi, odo_Reset bemenetekkel
- pos_X, pos_Y, pos_Phi, M1_E, M2_E, M3_E kimenetekkel

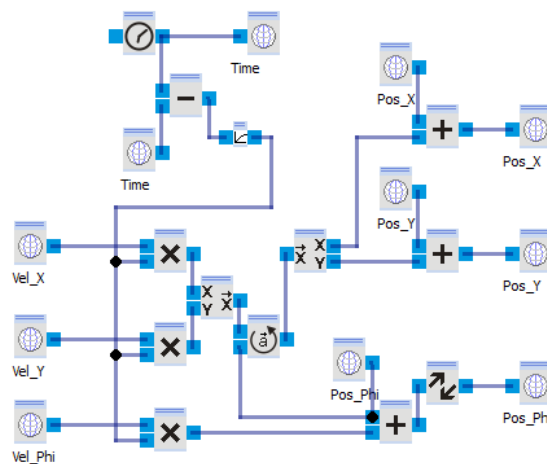
3.2. Implementálás

A matematikai modellezést követően, a Robotino View több lehetőséget is kínál virtuális robotok létrehozására: felépíthetők az alap blokkokból, megírhatóak Lua Script nyelven, melyet a program kezel, illetve létrehozható egy önálló blokk is C nyelven. Nézzük őket a fejlesztés folyamatában.

a) Alap Blokkokból felépített Virtuális Robot

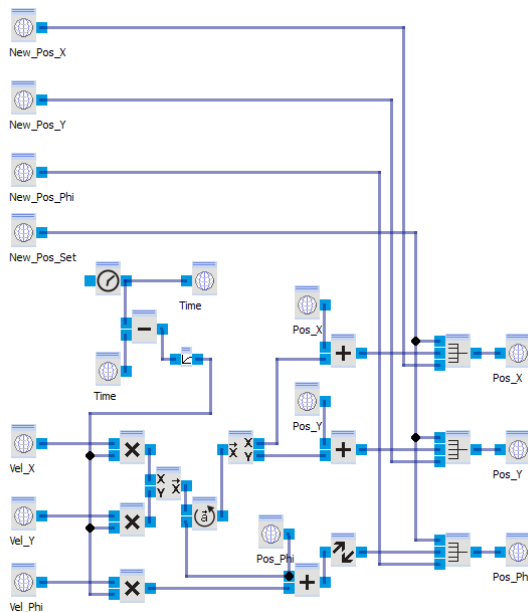
A legegyszerűbb megoldás virtuális robot létrehozására, ha a már meglévő elemekből építjük fel, melyet az 5. ábrán láthatunk.

Ennél a verziónál nem a motorok fordulatszáma szolgál bemenetként, hanem az x-, y-, phi-sebességek. Ez egy könnyen megérthető ábra: baloldalt felül a ciklusidő, alul az elmozdulás számítások, középen az elmozdulás vektor elforgatása, majd jobb oldalt a pozíciók integrálása látható.



5. ábra. Virtuális Robot alap blokkokból

Ez az első verzió fontos az emulátor működésének megértéséhez, de a tudása még messze nem éri el a kívánt szintet. Továbbfejlesztve lehetőség adódik a robot pozíciójának a beállítására. Az integrátorokat egy multiplexer sorral kiegészítve, már helyettesíthető vele az odometria, az omnidrive és a motorobjektumok. Felépítése ekkor már bonyolult és módosítása nehézkes, ezt a 6. ábra mutatja.



6. ábra. VR pozíció inicializálással

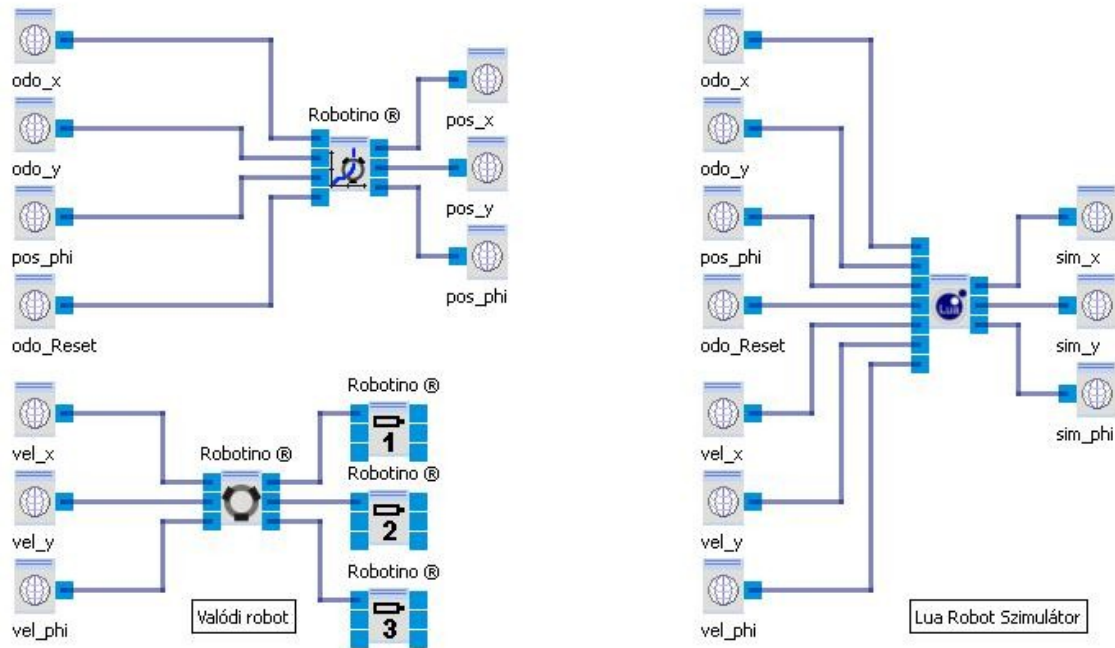
A New_Pos_Set változótól függően lehet váltani a robot pozíciójának inicializálása és a mozgása között. Ez a megoldás a program timerének hibája miatt újraindítás nélkül csak 35 percig képes működni, mivel az ekkor túlsordul és a robot rövid ideig fénysebességgel mozog, elhagyva minden pályát.

Ez a hiba küszöbölhető ki a következő verzióval.

b) Lua Script Virtuális Robot

Ezen a nyelven egy egyszerűsített robotot hoztam létre, mellyel számos elem kiváltható, (omnidrive, motorobjektumok és odometria) azonban ez a robot még mindig nem kezeli a motor encodereket. A működése időben nem korlátozott és referencia felvételre is alkalmas. Egyszerű feladatok és mozgások tesztelésére használható. A nyelv sajátosságainak köszönhetően gyorsan lehet fejleszteni, módosítani és a szerzett tapasztalatokat felhasználva később önálló blokk alkotható. Alkalmazására a 7. ábra mutat egy példát, melyen jól látható a méretbeli fejlődés az előző verzióhoz képest. Bal oldalt felül található az odometria, mely a valódi robot navigációjáért, és alatta az omnidrive a motorobjektumokkal, amik a mozgásért felelnek. Jobb oldalt látható a Lua Robot, melyben egyesítve megtalálhatóak az előbb említett blokkok. [6]

Ez az összeállítás lehetőséget ad arra, hogy a létrehozott virtuális robot mozgása összehasonlítható legyen a Robotino Sim Robottal, vagy akár egy valódival is. A Robotino Sim-el összehasonlítva az eredmények még holomonikus mozgás esetén sem mutattak 0,078 %-nál nagyobb hibát. Már ez az egyszerű felépítésű szimulátor is jól megközelítette a gyári szoftver pontossági paramétereit, töredék erőforrás mellett. A szimulátor forrása egyszerű és könnyen megérthető. [7]



7. ábra. Azonos funkciójú, Valódi és Lua Robot

:Lua Robot Simulator:

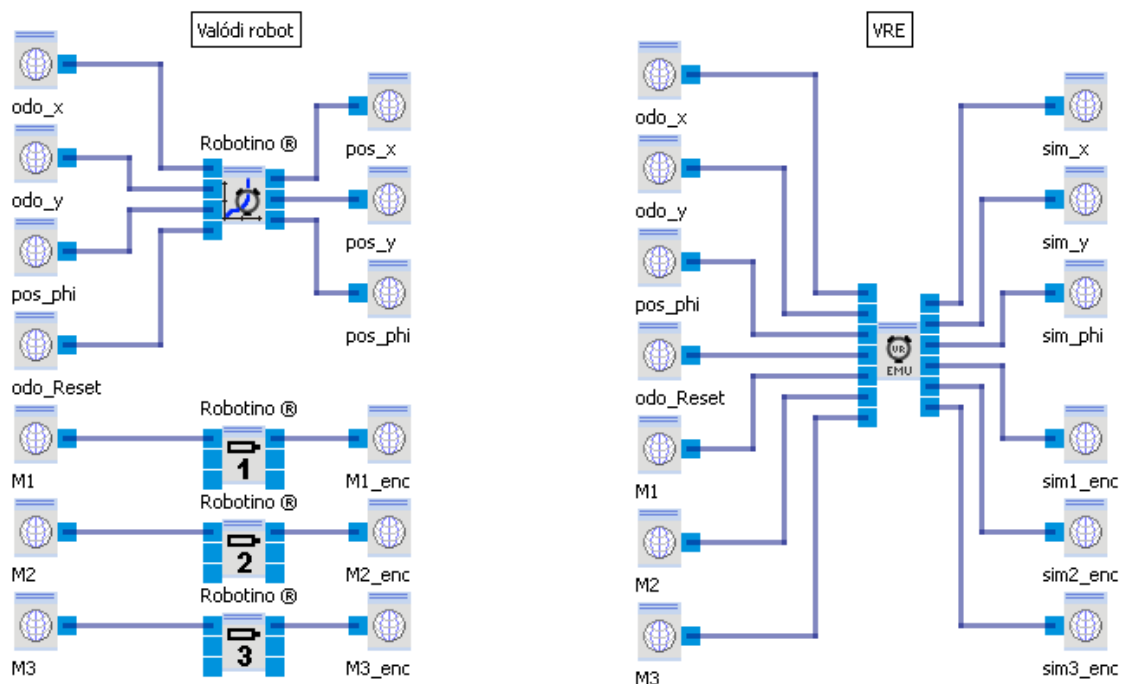
```
cycle_time=(os.clock()-os_time);
os_time=os.clock();
if in4==0 then
    speed=(in5^2+in6^2)^0.5;
    dir=math.atan2(in5,in6);
    pos_phi=pos_phi+cycle_time*in7;
    pos_x=pos_x+cycle_time*speed*math.cos(math.rad(pos_phi)+dir);
    pos_y=pos_y+cycle_time*speed*math.sin(math.rad(pos_phi)+dir);
else
    pos_x=in1; pos_y=in2; pos_phi=in3;
end;
if pos_phi<=-180 then pos_phi=pos_phi+360;
end;
if pos_phi>=180 then pos_phi=pos_phi-360;
end;
out1=pos_x; out2=pos_y; out3=pos_phi;
```

Ennél bonyolultabb felépítésű emulátort már önálló blokkban érdemes létrehozni.

c) Virtuális Robot Blokk

A Lua Robot Szimulátort ki kell egészíteni pár funkcióval, hogy teljes értékű VRE-ként használható legyen. Ezzel kicsit bonyolultabbá válik az alkalmazása, de cserében minden feladat tesztelhető vele. Bemenetként a motorok fordulatszámát, az új pozíciókat, illetve az odometria resetet kell kezelnie. Kimenetként szolgáltatnia kell a

pozíciót, orientációt és a motor encoderek jelét. Egy alkalmazási példa a 8. ábrán látható: [8]



8. ábra. Azonos funkciójú, Valódi robot és VRE

Mint az a 8. ábrán látszik, az emulátor be- és kimenetei úgy lettek kialakítva, hogy az, a helyettesített blokkokéval megegyezzen, ezzel is könnyítve a használatát.

A VRE működése -ideális robot szimulálás mellett- sokban hasonlít a Lua Script Virtuális Robotéra. A fő különbség a kettő között, hogy itt a motorok fordulatszám jeléből állíthatóak elő az encoder értékek úgy, hogy a ciklusidővel megszorozva a fordulatszámokat, a kapott értékeket kell integrálni. Ezek a szorzatok az elmozdulás meghatározásához is kellenek, és ebből azonos módon számolhatóak már tovább, mint az LSVR³-nél. Ilyen üzemmódban az összeállított Robotino View projektben RT⁴-nél gyorsabban is lehet emulálni, mely felhasználható például a kommunikációk, vagy a logikai folyamatok tesztelésére is.

Miért is van szükség a VRE Valós üzemmódu használatára? Ha egy valós és egy ideális VRE összehasonlításra kerül, szembetűnőek lesznek a különbségek a mozgásban és a pozíciókban. Itt 2,13%-os átlagos eltérés tapasztalható. Mi a különbségek oka? Ennek legfőbb oka a robot inerciájából adódik, mivel rá nem a virtuális tér, hanem a

3 Lua Script Virtuális Robot

4 Real Time – Valós idő

valós környezet fizikai törvényei érvényesek. Egy módosítást elvégezve azt tapasztaltam, hogy a hibaszázalék jelentősen lecsökkent: 1,3% alá. Itt nem tettem más csak kis mértékben módosítottam a kódot, bevezetve a fizika korlátait. A motoroknak nem engedtem meg dinamikus fordulatszám változást, csak olyan mértékben változhattak, amit a robot még követni tudott. Ahhoz hogy a szimulátor ezeket a fizikai korlátokat kezelni tudja, be kellett vezetni a robot momentumát, impulzus momentumát, és a mecanum kerekek tapadási és gördülési súrlódását. Az súrlódási együtthatókat kísérleti úton lehetett meghatározni, márvány és műanyag padlóra, valamint a Festo Robotino pálya RESOPAL⁵ anyagára. Ezeket az értékeket még a mecanum kerék görgőinek állapota (tisztasága és elhasználódottsága) is befolyásolja kis mértékben, de ettől jelentéktelensége miatt eltekintettem.

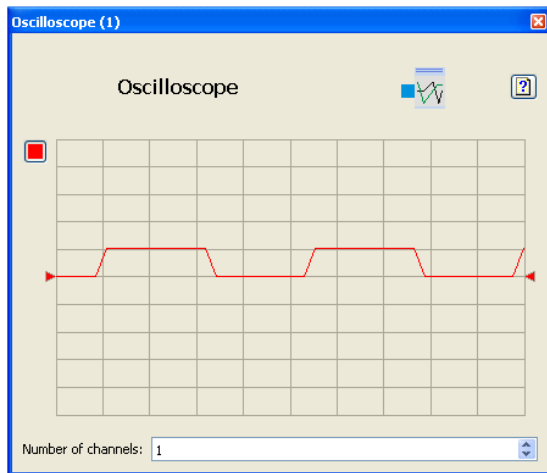
A VRE Valós üzemmódban a beérkező (M1, M2, M3) fordulatszám értékeket transzformálja a fenti fizikai mennyiségek figyelembevételével, majd a sebességeket újraszámolja és eltárolja a következő futásáig. Az újabb tesztek lefuttatása után a hibát sikerült 1% körülire csökkenteni, amely már elfogadható érték. Amennyiben valódi robottal kerül elvégzésre ugyanaz a futási teszt, a 3 lehetséges futtatási módban a kapott eredmények 2,8%-os eltérést mutatnak. Ezek a módok lehetnek:

- roboton futtatva
- számítógépen futtatva vezetékes kapcsolattal
- számítógépen futtatva vezeték nélküli kapcsolattal

Természetesen a hibaszázalék meghatározása során a robotokon a korrekciókat be kellett állítani, hogy a saját odometriájuk pontos legyen. A VRE realizálása során nem vettem még figyelembe a mecanum kerekek ezen verziójának azon hátrányos sajátosságát, mely szerint a hajtó görgők távolsága nem állandó a robot középpontjától.

Amint az az 1. és a 3. ábrán is látszik, ezek a görgők két síkban helyezkednek el. Amennyiben a mecanum kerekek azonos helyzetben állnak a 3 tengelyen, és állandó fordulatszámmal forognak, a robot szögsebessége a 9. ábrán látható módon alakul. A kerekek ezen tulajdonságát figyelembe véve a különbség a valódi és a VRE között 0,6%-ra csökkenthető.

⁵ magas nyomáson gyártott, réteges kialakítású, melamin gyanta réteggel bevont lemez



9. ábra. Szögsebesség változás

erőforrásigénye jelentősen megnőne a csekély hibaszázalék csökkenés mellett. Mivel ez enélkül is belül van a robotok különböző futtatási lehetőségek pontatlanságain, ezért ezt nem célszerű megvalósítani.

Azonban ennek elengedhetetlen feltétele az, hogy a kerekeket minden tesztelés előtt kézzel a megfelelő pozícióba forgassuk. Ezt a műveletet azért manuálisan kell elvégezni, mert a motorok tengelyére felszerelt encodereknek nincs alaphelyzet érzékelőjük. Amennyiben a robot kerekei nem azonos pozícióban állnak, a 9.ábránál sokkal kaotikusabbnak tűnő szögsebesség görbe alakul ki. Ezt figyelembe véve a VRE

d) Az elvégzett kísérletek

A VRE jelenlegi verziója a robot fizikai paramétereit egy külső fájlból (VRE.cfg) olvassa be.

```
:Virtual Robot Emulator Configuration:
<VRE_CFG>
<ROBOT>
Robotino;    //Rootino/Custom
</ROBOT>
Rootino/Custom
<MODE>
Real;        //Real/Ideal
</MODE>
<TIME_MULTIPLIER>
Real;        //Real/0,1 .. 10
</TIME_MULTIPLIER>
<FLOOR>
Resopal;     //Resopal/Stone/Plastic
</FLOOR>
<WEIGHT>
0;           //kg
</WEIGHT>
<DIAMETER>
0;           //mm
</DIAMETER>
<GEAR_RATIO>
0;           //const
```

```

<GEAR_RATIO>
<WHEEL>
0; //mm
</WHEEL>
<MOTOR_ACC>
0; //rpm/sec
</MOTOR_ACC>
</VRE_CFG>

```

Amint az a mellékelt fájlban látszik, a számos fizikai paraméter mellett az üzemmód választás is itt kapott helyet.

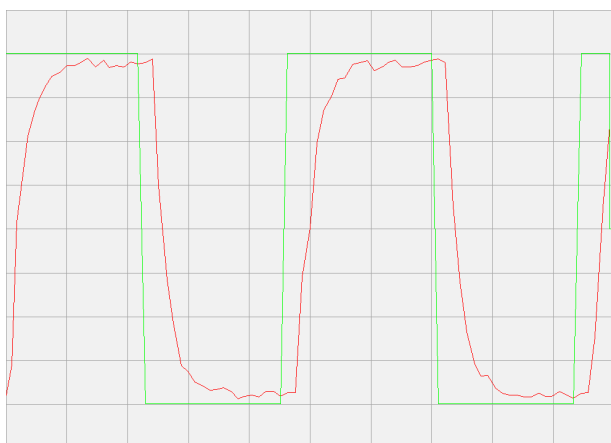
Robot választásnál kiválasztható, hogy Robotinot szeretnénk használni, vagy más, Custom robotot. A Robotino paraméterei már integrálva vannak a VRE-be, így ezeket nem szükséges megadni. Ahol az emulátor 0 értéket talál, ott a beépített értékkel dolgozik. Amennyiben valamilyen hardveres változás történt, lehetőségünk van a korrekcióra. Például a zselés ólomakkumulátorok Li-Polimerre történő cseréje 3 kilogrammal is csökkentheti a robot tömegét, mely jelentősen befolyásolja a mozgását.

Mód választásnál megadható, hogy valós vagy ideális módban szeretnénk-e futtatni, valamint egy idő szorzó is, mely segítségével lehet valós időnél gyorsabb, vagy adott esetben lassabb tesztek futtatni. A padló anyaga is itt definiálható. Ezekre a beállításokra a gyári szimulátornál nincsen lehetőség.

Fizikai paraméterek:

Vannak köztük olyanok, amiket egy egyszerű méréssel meg lehet határozni: tömeg, átmérő és áttétel. A motorok gyorsulását már nehezebben lehet mérni.

Első kísérleti összeállításnál a motorokat vizsgáltam. A méréseket különböző padlóanyagokkal végeztem, szerepelt köztük műkö, műanyagpadló és Resopal is. Mivel X, Y és Phi irányú elmozdulásokhoz különböző motor használat tartozik, ezért külön vizsgáltam azokat. Forgásnál mindhárom motor azonos irányban, azonos sebességgel forog, előrehaladásnál csak a két első motor, míg oldalazásnál szintén három motor hajt, de különböző fordulatszámmal. A kísérletek célja az volt, hogy meg tudjam határozni a gyorsulások értékét. Első lépésben egy meredek felfutású sebesség görbét adtam a robotnak, és figyeltem a tényleges sebességét. Amikor ezt a sebességet elérte, szintén nagy meredekséggel, ellentétes irányú mozgásra utasítottam. A kísérlet eredményét egy oszcilloszkópon jelenítettem meg, mely a 10. ábrán látható. [9]



10. ábra. Forgás oszcillogram

Sajnos a Robotino View nem támogatja az oszcilloszkópok OSD⁶ funkcióját, ezért a tengelyek beosztása nem kerülhetett fel az ábrára. A vízszintes tengelyen az idő 1000 ms/div⁷, a függőlegesen pedig a sebesség (középen a 0 értékkel) 100 (mm/sec)/div található. A zöld színű görbe a nagy meredekségű vezérlőjel, a

piros a robot tényleges sebessége. Az ábrán középen található a 0 sebesség. Látszik, hogy ettől távolodva pozitív és negatív irányban is, a meredekség kisebb mint közeledve felé. A tényleges sebesség görbén megfigyelhető a PID szabályzó működése is. Ezeket a kísérleteket elvégezve (a már említett irányokban: X, Y, Phi) és az eredményeket összegezve, átlagolva sikerült meghatározni a Robotinora jellemző paramétereket.

A megkapott eredmények pontosságának ellenőrzésére további kísérleteket kellett végezni, itt a Robotino navigációs és helymeghatározó rendszerétől független, külső szenzorokkal ellenőriztem a sebességeket. Forgási sebesség ellenőrzésére egy CruizCore XG1000 típusú giroszkópot alkalmaztam, melyet a Robotino is képes kezelni, és az általa mért adatokat oszcillogramon megjeleníteni. Az X, Y irányú sebességek ellenőrzésére egy Balluff BOD 66M-LA04-S92-C lézeres távolságmérő szenzort használtam, melyet a robot analóg bemeneteire illesztettem, ezáltal ezeket az adatokat is meg lehetett jeleníteni. A mérések menete azonos volt a már előbb ismertetettel, annyi különbséggel, hogy itt a tényleges sebesség értékeket a külső szenzorok szolgáltatták.

Ezen kísérletek alapján az előbb meghatározott paramétereket kis mértékben módosítani kellett.

6 On Screen Display – magyarul képernyő kijelző

7 Osztás

4. Vizualizációs környezet

4.1. Vizuális környezet kiválasztása

A vizualizációs környezetet, a Robot monitort egy komplex példán keresztül mutatom be. Ez a példa egy, a Pécsi Tudományegyetem Pollack Mihály Műszaki és Informatikai Karának Pollino csapatához -melynek jómagam is tagja vagyok- közelálló projekt szimulálására alkalmas, amely a RoboCup Festo Logistic League feladata. Itt egy gyártócsarnokban kell 3 robot segítségével, autonóm kooperációban alapanyagokból készterméket előállítani. A termelésre 10 megmunkáló (1,2,3-as típusú), 2 újrahasznosító és egy tesztelő állomást lehet felhasználni. A termelés során Empty, S0, S1, S2, P termékek fordulhatnak elő. A kezdetben ismeretlen típusú állomásokat fel kell deríteni, majd megfelelő sorrendben korongokkal (munkadarabokkal/alkatrészekkel) feltölteni. Az elkészült terméket az aktív kapuba juttatva fejezhető be a termelési folyamat. A gyártás menete:

<u>Alapanyag:</u>	<u>Állomás:</u>		<u>Termék:</u>
S0	1-es állomás	→	S1
S0 + S1	2-es állomás	→	Empty + S2
S0 + S1 + S2	3-as állomás	→	Empty + Empty + P
Empty	újrahasznosító	→	S0

A megmunkáló állomások állapotát egy az automatizálásban elterjedt (piros, sárga, zöld) lámpaoszlop jelzi, melyet egy általunk kifejlesztett speciális szenzor érzékel. A szimulátornak ezeket a lámpaoszlopokat is működtetnie kell.

Amit a Robot Monitornak összességében kezelnie kell:

- pálya alaprajza
- megmunkáló állomások helye
- megmunkáló állomások státusza
- navigációs csomópontok

- rendelkezésre álló közlekedési élek
- 3 robot és aktivitásuk megjelenítése
- állapotjelző lámpák

4.2. Vizualizáció alapjai

A vizualizáció megtervezésekor számos paramétert figyelembe kellett venni:

- egyszerű folyamatkövetés
- gyors szerkeszthetőség
- jó átláthatóság
- letisztult formák
- minimális hardverigény
- szemléletesség

Ezen szempontokat figyelembe véve alakítottam ki az alap vizualizációt. Mivel a megjelenítésnek a robotokon és számítógépeken is futniuk kell, ezért a multi layeres és vektor grafikus eszközöket kombinálva használtam. Lehetőséget adva ezzel az egyszerű szerkesztésre, mivel a pálya alaprajza, az akadályok, falak, felfestések külön bmp fájlokban vannak letárolva. A különböző szenzoroknak külön rétegeket hozhatunk létre, (induktív, infra távolságmérő és színérzékelő szenzorok), melyeket a programnak meg sem kell jelenítenie. Egy ilyen kép mérete 40 kbyte nagyságrendbe esik, ezért ezeket a rétegeket egy mátrixba kódoltam, (melyek mérete nem haladja meg az 1 kbyteot) és innen futás közben gyorsan, kis erőforrás szükséglettel lehet kiolvasni a szenzorok adatait. Ezeket az adatokat a RM⁸ egy kódolt csatornán továbbítja a dekódereknek, melyek azt kibontva elérhetővé teszik.

A megjelenített objektumok különbözőségeik miatt más-más frissítést igényelnek. A pálya alaprajzát elég csak egyszer kirakni, míg az állomásokat minden művelet után, a robotokat pedig másodpercenként többször is lehet frissíteni. Erőforrás menedzsmentet szem előtt tartva ezek a frissítések változtathatóak is. Ha számítógépen történik a

8 Robot Monitor

vizualizáció futtatása, akkor a robotok minden ciklusban újrarajzolhatóak. A robotról futtatva már nem célszerű ezt a megoldást választani, mivel a ciklusidőt indokolatlanul megnöveli, ezért itt azt a megoldást választottam, hogy csak adott távolságú elmozdulások után frissíték, ezzel nem kevés időt nyerve a futó szabályzó köröknek.

4.3. Virtuális tér megalkotása

Virtuális térről az embernek rögtön egy 3 dimenziós tér juthat az eszébe, ahol minden tökéletesen meg van szerkesztve (hasonlóan mint a Sim RoboCupTM-ben). Az ilyen megvalósításnak számos hátránya van, amellet az előnye mellett, hogy az átlag felhasználót elvarázsolja a vizuális megjelenése:

- magas hardver és erőforrás igény
- pályát folyamatosan mozgatni kell
- nem látjuk át az egész folyamatot
- kiegészítő információkat nem jelenít meg

Korlátozva a virtuális teret 2 dimenzióra és fentről szemlélve a pályát az egész tér belátható. Egyszerűsítettem a robotok formáját, mivel a modellben megállapítottam, hogy 3 szabadsági fokú robotokról van szó, ezért tudni kell a koordinátáit és az orientációját. A legegyszerűbb ábrázolási megoldás egy robot átmérőjű kör, mellyel a pozíció adható meg. Az orientáció szemléltetésére egy sugár irányú vonallal jelöltem a robot elejét. Ezeket az ábrákat mind vektorgrafikus úton lehet felvinni, melyeknek az erőforrásigénye töredéke egy textúrázott robot megrajzolásának.

Kiegészítő információkat is meg kell tudni jelenítenie a rendszernek.

- robot állapot
- korong információ
- feldolgozó állomások adata

Az alap szempontokat figyelembe véve, több eszköz kombinációjával lehet elérni a legjobb eredményt. Vannak kiegészítő információk melyeket egyszerű szín változással és vannak amiket alfanumerikus formában jeleníték meg. Egy robot vagy állomás

állapotát a színével lehet szemléltetni:

- Állomás: szabad: zöld
- Állomás: lefoglalt: kék
- Állomás: üzemen kívüli: piros
- Robot: aktív: világos szín
- Robot: passzív: sötét szín
- Lámpa szenzor: grafikus megjelenítés
- Útvonalak megjelenítése: grafikus megjelenítés
- Csomópontok: grafikus és numerikus megjelenítés
- Logisztikai információk: alfanumerikus megjelenítés

5. Tesztelés

Az elkészült VRE-t és szimulátort jelenleg a Pécsi Tudományegyetem Pollack Mihály Műszaki és Informatikai Karának Pollino csapatán kívül a Pécsi Tudományegyetem Babits Mihály Gyakorló Gimnázium és Szakközépiskolájában is tesztelik, használják.

5.1. Robotino programozás

Ahhoz hogy a VRE és környezet alkalmazását könnyebben tudjam szemléltetni elengedhetetlen a Robotino View programozásának rövid bemutatása

a) Általános Robotino programozás

A programokat felépítő alapegységek (téglák) a blokkok,

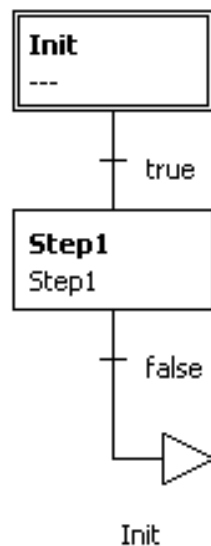
- változók
- matematikai, logikai műveletek

- időzítők, számlálók
- generátorok, megjelenítők
- bemeneti eszközök
- robot specifikus blokkok

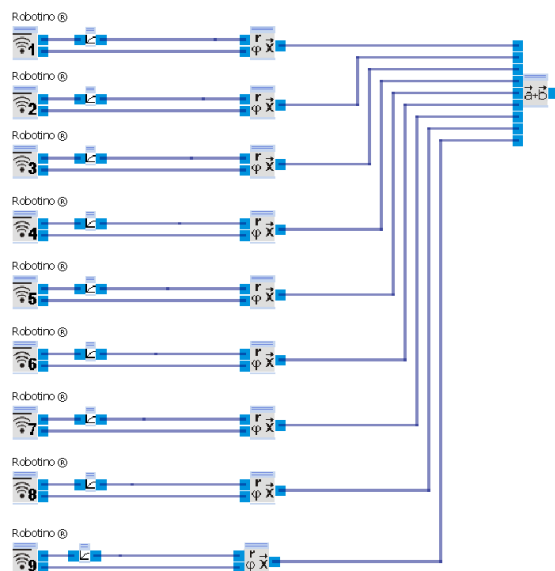
A robot specifikus blokkoknál érdemes kicsit vizsgálni. A Robotino View ezeken keresztül éri el a robotot. Mit is jelent ez? Adatokat kér le tőle, feldolgozza azokat, és utána visszaküldi. Maga a program nem csak robotos feladatok futtatására képes, önállóan robot nélkül is használható. Ha van egy saját fejlesztésű robot, ami képes hálózati, vagy soros kommunikációra, akkor a megfelelő interfészek megírása után akár Robotino View kompatibilissé is tehető.

A blokkokat összekötve hálózatokat (gráfokat) lehet létrehozni. Ezek alkotják az alprogramokat. Egy nagyobb projekt több alprogramból állhat, melyek a Main programban rendezhetők struktúrába. Lehetőség van belépési és kilépési feltételek megadására, -itt a C-nyelv szabályai érvényesek- melyek lehetnek egyszerű vagy összetett formulák is. A struktúrán belül lehetnek alternatív és párhuzamos ágak, melyek általában a bonyolultabb, összetettebb feladatokhoz szükségesek.

A következő ábrák egyszerű Main és alprogramot mutatnak.



11. ábra. Main program



12. ábra. Alprogram részlet

b) Pollino specifikus Robotino programozás

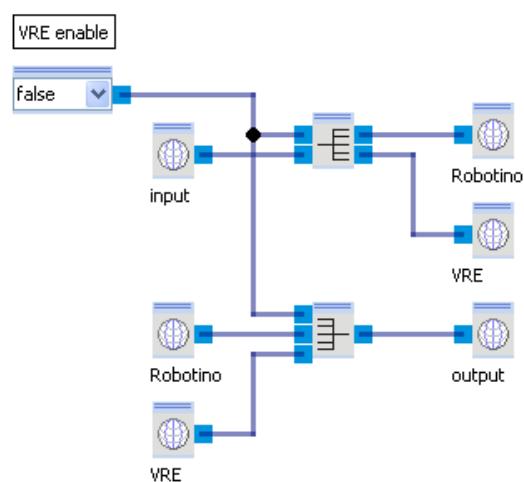
Komplex feladatok elvégzése során számos olyan alprogram van amelynek állandó jelleggel kell futnia. Mivel a Robotino View csak az aktív alprogramokat futtatja, ezért biztosítani kell ezek számára egy aktív szálát, mely a program futása során nem szakadhat meg. Ezt egy olyan -az egész programmal párhuzamos- szállal lehet elérni, mely a program indulásakor jön létre, és nincs kilépési feltétele. Itt helyezkedik el a kommunikáció, helymeghatározás (odometria), mozgásvezérlés (omnidrive), VRE és az RM. Amennyiben a virtuális valóság megkívánja a kiegészítő szenzorok használatát, (optikai távolságérzékelő, indukzív szenzorok, analóg bemenetek), az ehhez szükséges blokkok is itt találhatóak. Ilyen Main program és alprogramok szemléltetésére készültek példák a mellékletben.

5.2. VRE⁹ használata és programba illesztése

A VRE új, vagy már meglévő projektbe is illeszthető. A futási módok három csoportba oszthatóak:

- Roboton
- Helyi gépen
- Hálózati gépen

Az emulátor együtt tud futni a valós robottal is (mellyel össze lehet hasonlítani), vagy egy kapcsoló (változó) segítségével az emulált és a valós robot között lehet választani. Amennyiben így szeretnénk használni, akkor a bemeneteket és kimeneteket multi- és demultiplexereken keresztül kell megosztani a valós robot és a VRE között. A 13. ábra ilyen be és kimenetre mutat példát.



13. ábra. Robotino / VRE ki- és bemenetek bekötése

⁹ Virtuális Robot Emulátor

A célkitűzésben szerepelt, hogy egy robot nélküli szimuláció megalkotása a cél, a környezet futtatása teljesen azonos a roboton és a helyi számítógépen, mivel azonos operációs rendszer fut mindkettőn.

Helyi gépről futtatva a VRE-t a megfelelő ki és bemenetekkel kell összekötni. A futtatást csatlakozás és hálózat nélkül lehet elindítani. A vizualizációhoz egy kódátalakítóval kell illeszteni, így láthatóvá téve a mozgásokat.

Hálózati gépen futtatva lehetőség nyílik egy távoli megfigyelésre, nyomon követésre. Ez esetben az aktív robotok vagy a VRE-t futtató számítógépek egy hálózati UDP csatornán keresztül küldik el a vizualizáció számára szükséges adatokat. Ezeket az adatokat megkapva a hálózati gép képes megjeleníteni a kívánt robotokat.

A tesztelések nagy részét helyi gépen végeztem, mert ehhez csak egy számítógép és Robotino szükséges.

6. Eredmények

6.1. Alkalmazási lehetőségek

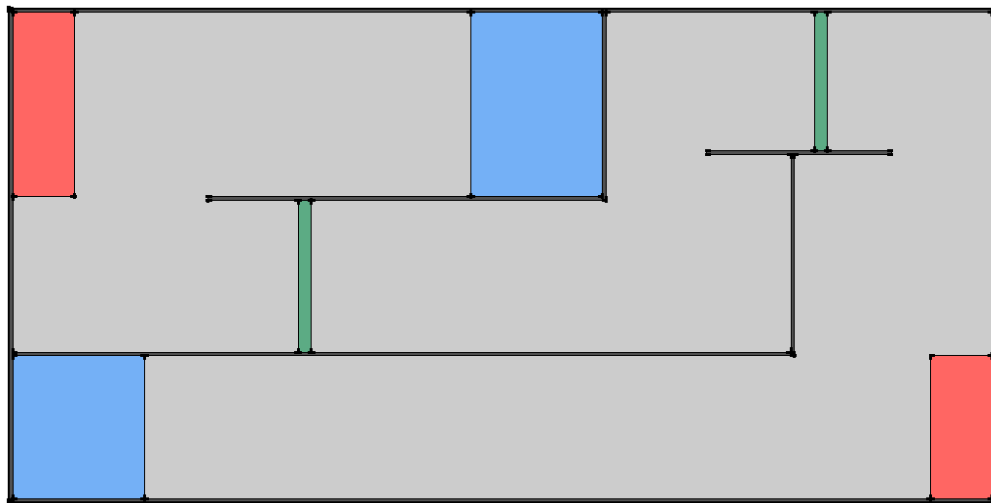
A kifejlesztett VRE és vizualizációs környezete főként oktatási cézzal készült. Hasznos kiegészítője lehet a robotika oktatásának, illetve nagy segítség azoknak a hallgatóknak, középiskolásoknak akik érdeklődnek a robotok programozása ránt.

6.2. Továbbfejlesztés

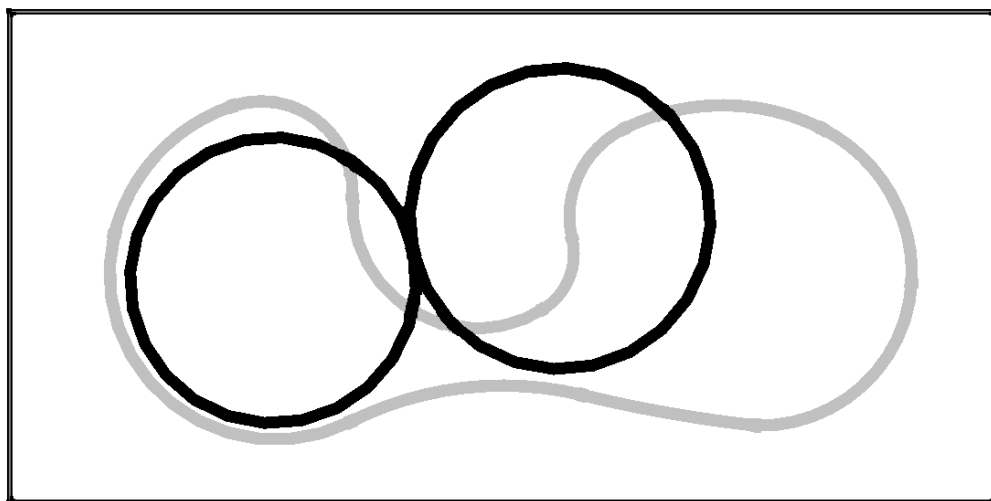
A VRE jelenleg nem terjed ki a robot fedélzeti elektronikus szenzorainak kezelésére, értem itt a fedélzeti feszültség és a pillanatnyi áramfelvétel mérését. Az a cél, amellyel ez a keretrendszer készült, ezt nem is teszi indokolttá, de későbbiekben ezzel is bővíthető. A valóságos robotszimuláció már a fizikai erőhatásokat is figyelembe veszi, melyekből a motorok áramfelvétele számítható, majd azokból egy egyszerűbb integrálással az akkumulátor feszültség csökkenése is meghatározható.

A részletesen bemutatott RoboCup™ szimulátor mellett elkészült még 2 mintaalkalmazás. Az egyik (14. ábra) a 2010-es II. Robotino Olimpia minősítő

feladatának tesztelését teszi lehetővé az infra távolságmérő szenzorokat használva. A sötét szürke vékony vonalak ábrázolják a falakat, a kék mezők a start és a cél zóna, a piros a tiltott terület és zöld színt kaptak az időmérő kapuk. A másik (15. ábra) egy vonalkövetési példa, ahol tesztelhetők az optikai és induktív érzékelők. Jól látható a két elkülönülő szín, a fekete az optikai érzékelőknek, míg a szürke az induktív érzékelőknek kialakított pálya. Az ábrák a pálya megjelenési képét szemléltetik. A szenzorok jeleit egy meg nem jelenített kép adja, melyen külön vannak választva az optikai, induktív és az infra távolságmérő szenzorok. Ezeket az információkat futás alatt a fentebb már említett kódolt mátrix tartalmazza.



14. ábra. Minősítő feladat



15. ábra. Vonalkövetés induktív vagy optikai szenzorokkal

A környezethez elkészült keretrendszer számos alkalmazási lehetőséget rejt még magában:

- Intelligens raktár
- Különböző logisztikai feladatok
- Betegkísérő rendszer
- Komplex robottechnika oktatás

Jelen állapotába a Robot monitor csak egy szimulációs környezetet tud futtatni, amit mappa másolással, áthelyezéssel lehet változtatni. A továbbfejlesztési tervekben szerepel egy olyan opció, hogy ezt egyszerűen a Robotino View-n belül tehesük meg.

7. Értékelés és következtetések

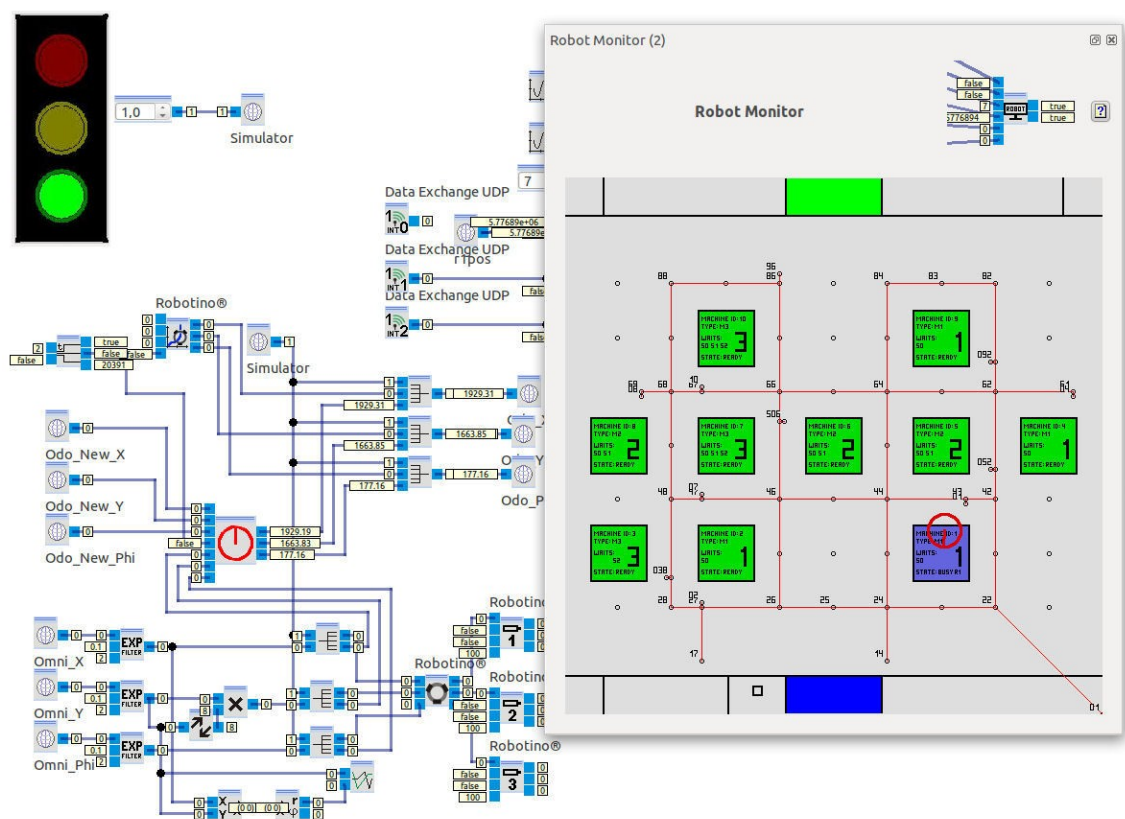
Az elkészült RM és VRE megbízhatóan és stabilan működött a tesztek alatt és a használat során is. A célkitűzés minden pontjának sikerült megfelelő környezetet teremteni, ezzel létrehozva egy olyan rendszert amely nem csak országos szinten egyedülálló. Segítségével számos feladat és probléma bemutatható az oktatásban. Használata nagy segítséget jelent a RoboCup Logistic League versenyére készüléskor, értékes időt megtakarítva a fejlesztő csapat számára.

A környezet kipróbálásához és használatához szüksége fájlok letölthetők a <http://pollino.hu/Downloads> weboldalról. A telepítési, használati és pályaszerkesztési leírás ugyanitt megtalálható, de a forráskódokat saját szellemi termék révén nem osztom meg. A felhasználók az általuk készített pályákat megoszthatják a többi felhasználóval a pollino.hu oldalon keresztül, és az arra érdemesek a soron következő verzióba is beépülnek majd.

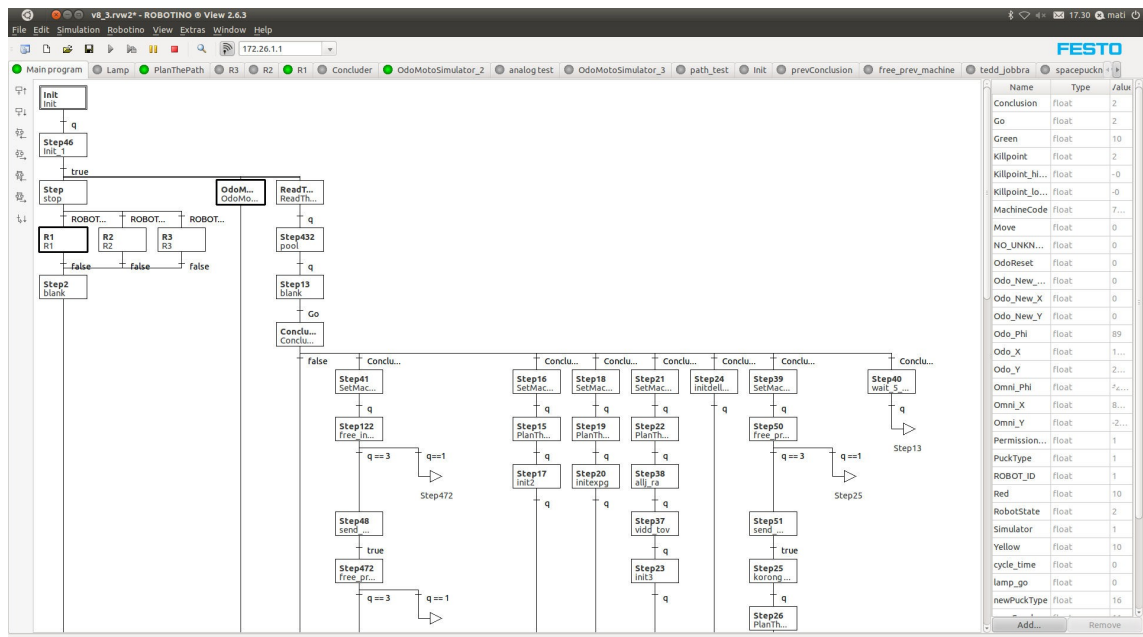
8. Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani mindazoknak akik munkámat segítették és idejüket nem sajnálva támogatták a fejlesztést. Külön kiemelném Megyeri Pétert amiért a tervezésben és ezen dolgozat elkészítésében is mellettem állt. Ötleteivel, meglátásaival iránymutatást adott, valamint Slang Tamást, akinek a diplomamunkája nélkül nem jöhetett volna létre ez a dolgozat.

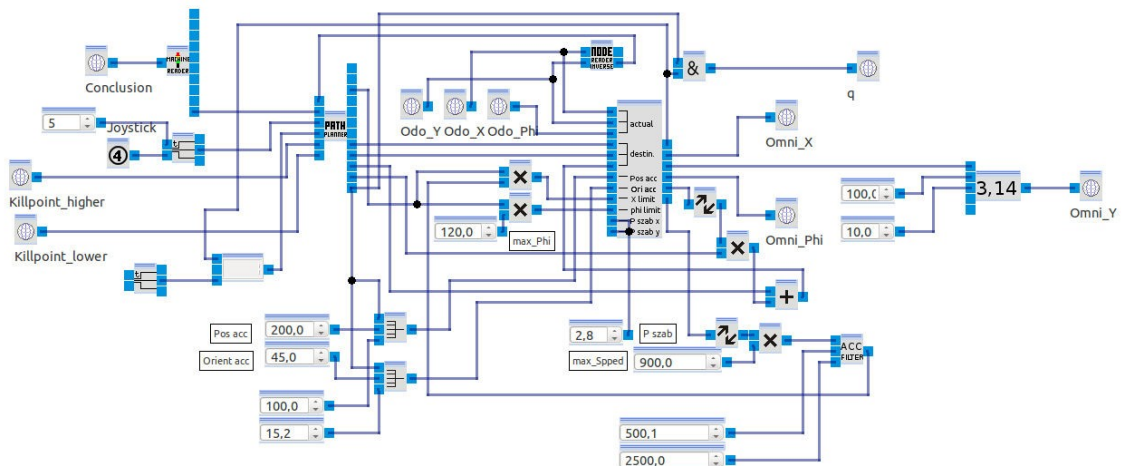
9. Melléklet



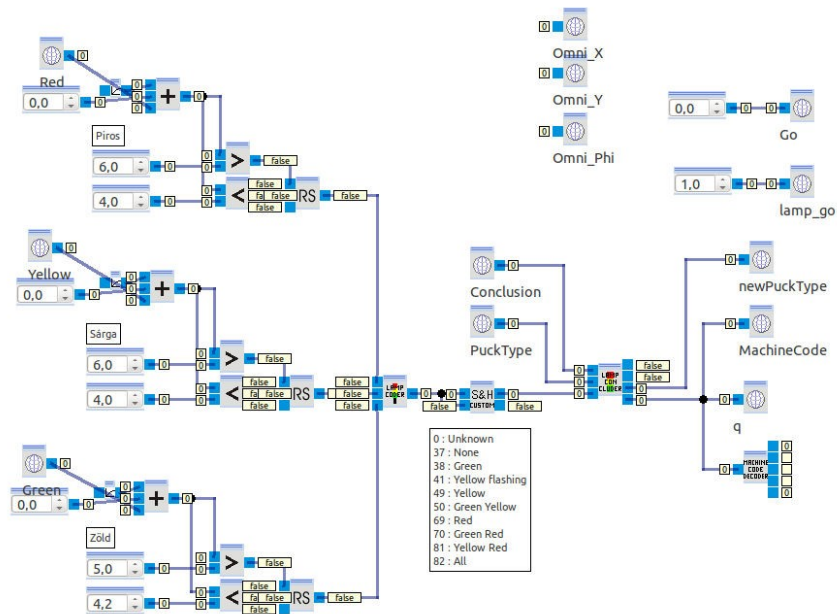
16. ábra. VRE és RM futás közben 1 robottal, gyártás közben



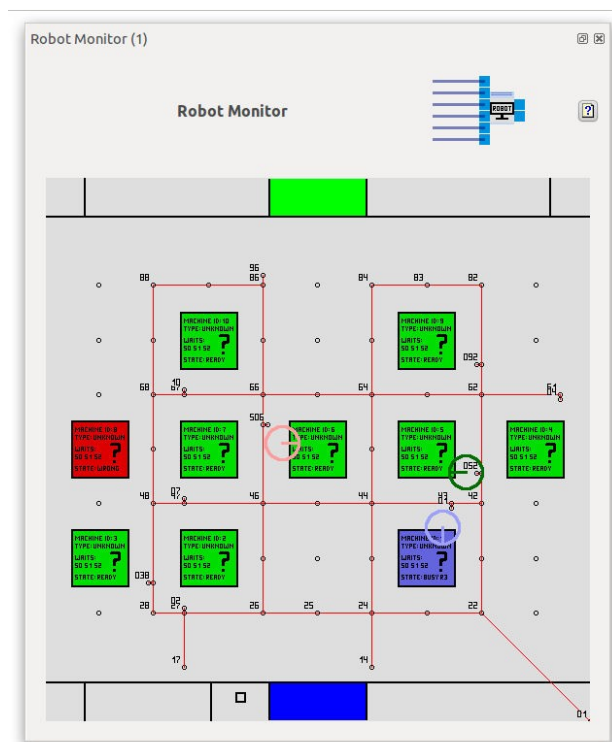
17. ábra. Pollino Main Program RoboCup Logistic League-re párhuzamos és alternatív szálakkal



18. ábra. Útvonal bejáró alprogram Saját blokkokkal



19. ábra. Lámpa szenzor feldolgozó alprogram, Szimulátorhoz



20. ábra. Szimuláció 3 robottal

10. Irodalomjegyzék

- [1] Dr. Donner Csaba: A robottechnika alapjai
- [2] Lantos Béla: Robotok irányítása, Akadémia kiadó Budapest, 1997
- [3] Raul Rojas: Omnidirectional Control, www.fu-fighters.de
- [4] Chuntao Leng, Qixin Cao : Velocity Analysis of Omnidirectional Mobile Robot and System Implementation, Conference publication, Res. Inst. of Robotics, Shanghai Jiao Tong Univ, 2006
- [5] http://doc.openrobotino.org/documentation/OpenRobotinoAPI/1/doc/rec_robotino_com_c/_omni_drive_8h.html
- [6] <http://www.lua.org>
- [7] Raul Rojas and Alexander Gloye Forster: Holonomic Control of a robot with an omnidirectional drive, Künstliche Intelligenz, BöttcherIT Verlag, 2006
- [8] Slang Tamás: Robotino View függvényblokkok programozása, Szakdolgozat MIB II.-177/2007/2011.
- [9] Simon Béla: Fizika középfokon, Shannon iroda, 2003

11. Nyilatkozat

Aláírással igazolom, hogy a Tudományos Diákok Körének dolgozat saját munkám, a felhasznált irodalmat korrekt módon kezeltem és a munkára vonatkozó jogszabályokat betartottam.

Pécs, 2012. november 16.

.....
Malkó Tibor