MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

RT-Middleware periféria távérzékelés

Tompa Tamás
Villamosmérnök MSc szakos hallgató

Konzulens:
Dr. Kovács Szilveszter
egyetemi docens
Általános Informatikai Tanszék

Miskolc, 2013

Tartalomjegyzék

1. Bevezetés

A tudomány és a technika folyamatos fejlődése révén egyre újabb eszközök válnak életünk részévé, ezen eszközök egyik csoportja a robotok. A robotika napjainkban egyre népszerűbb tudományág, a csúcstechnológia alkalmazása is egyre inkább ebbe az irányba halad.

De mit is értünk robot illetve robotika alatt? Az interneten található számos definíció közül az egyik szerint „*A robotok olyan fizikai ágensek, amelyek a fizikai világ megváltoztatásával oldanak meg feladatokat.*" [1]. Mit is takar ez a definíció? Állítása szerint a robot egy olyan környezetét érzékelő illetve környezetébe beavatkozni tudó gépezet, amely bizonyos feladatok megoldásával megváltoztatja a körülöttünk lévő világot. Tehát egy robotnak konkrét célja, feladata van. Robotika alatt pedig a robotokkal foglalkozó tudományágat értjük.

A robotok alkalmazása számos előnnyel jár, az élet több területén is alkalmazzák őket, elterjedtek például az iparban, az orvostudományban, illetve a jövőben alkalmazandóak, mint szociális segítőtársak. Napjainkban viszont a robotok még terjedtek el a háztartásokban, de már vannak ma is megvásárolható robot eszközök, melyek a háztartásbeli feladatok elvégzését hivatottak segíteni, ilyen például Roomba, a porszívórobot, TurlteBot, a felszolgáló robot, vagy a házilag épített fűnyíró robotok.

Kutatómunkám célja a robotika tématerületének mélyrehatóbb tanulmányozása, egy konkrét robot távvezérlésének megvalósítása céljából. A robot, azaz Ethon-t, a Budapesti Műszaki és Gazdaságtudományi Egyetem fejlesztése, célja egy portás robot megtestesítése, amely adott folyosón bolyongva felismeri a körülötte lévő embereket, felismerés után üdvözli őket, illetve ha akkumulátora merülőben van, akkor megkeresi a töltőegységét. Ethon főként etológiai kutatásra, azaz ember-gép interakció vizsgálatára készült, melyet az Eötvös Lóránd Tudományegyetem tanulmányoz. A Miskolci Egyetem Általános Informatikai tanszékén lévő, a projektben résztvevő kisebb csapat feladata, hogy a fentebb említett funkciókat megvalósító robot rendszert megtervezze, implementálja.

Dolgozatom célja a projekt keretében megvalósított munkám bemutatása, amely során a teljes robotvezérlő rendszer részeként implementálok egy olyan szoftverkomponenst, amely segítségével Ethon távvezérelhető, azaz távolról irányítható, illetve a robot valamennyi funkciója működtethető. Tanulmányozom továbbá a már meglévő robot keretrendszereket, felépítésüket, szükségességüket, illetve vizsgálom előnyeiket, hátrányaikat. Részletezem az

adott robot keretrendszer részeként megvalósított, a robot távvezérlését lehetővé tevő szoftverkomponens tervezését, implementálását, illetve tesztelését.

2. Információgyűjtés

Ebben a fejezetben a megvalósításhoz szükséges elméleti hátteret, többek között a robot célját, felépítését ismertetem valamint tanulmányozom a különböző robotvezérlő rendszereket.

2.1 A robot bemutatása

A robot (teljes nevén Mogi Ethon Rubens, továbbiakban Ethon) célja egy portás robot megtestesítése, amely egy adott területen bolyongva felismeri a körülötte lévő embereket, a felismerés után üdvözli, követi, megjegyzi az illető arcát, felismeri az őt körülvevő akadályokat (akár dinamikus akadályokat is) majd ha akkumulátora merülőben van, akkor megkeresi a töltő egységét.
Ethon-t a Budapesti Műszaki és Gazdaságtudományi Egyetem (BME) tervezte és valósította meg, a Miskolci Egyetem Informatika Tanszékén lévő, a projektben résztvevő kisebb csoport feladata egy, a fentebb említett funkciókat megvalósító robotrendszer implementálása.
A robot fő célja, hogy az Eötvös Lóránt Tudomány Egyetem (ELTE) etológusai segítségével vizsgálják az ember-gép interakciót, azaz, hogy az emberek hogyan reagálnak a környezetükben lévő robotokra.

2.2 A robot technikai leírása

[2]

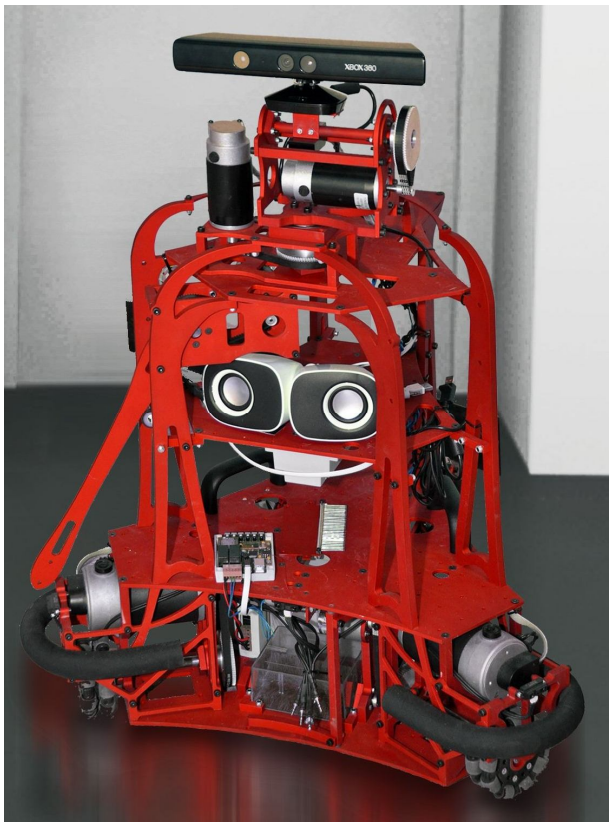
Ethon felépítése két részre különíthető el, ez az alsórész és a felsőrész. Az alsórész a robotalap, amely felépítése állandó, majd e felett helyezkedik el a felépítmény, amely moduláris felépítésű, azaz a későbbiekben még bővülhet különböző feladatokat megvalósító modulokkal.
Az alsórészen található az akkumulátor, a robot működését irányító mikorvezérlő (AVR), a kereket meghajtó, egyenként 300W-os 3 darab kefés DC motor, a motorok üzemeléséhez szükséges végfokok, 6 darab Sharp távolságérzékelő, további elektronikák (step up és step down áramkörök, szervó vezérlő), illetve egy takarító egység, amely egy forgó henger alakú kefe kosztárolóval együtt, és egy labdaütögető egység. Ezen egységeket relével kapcsolgatja a központi elektronika.

Ethon holonómikus hajtással rendelkezik, amely azt jelenti, hogy kerekei kerültén újabb kisebb kerekek (görgők) helyezkednek el. Ennek a hajtásnak az előnye, hogy a robot bármilyen irányban tud mozogni, mert kerekei a forgásiránnyal keresztben is képesek elmozdulni. A motor és a kerekek között szíjáttétel található.
A felső részen 2 darab mikorvezérlő (szintén AVR) helyezkedik el, melyek a robot fejének és karjának a vezérléséért felelősek, itt található a robot fejét megtestesítő Kinect, és itt helyezkedik el a robot karja is. A fej minden irányban mozgatható, a kéz fel-le mozgásra képes. A robot előre meghatározott üzenetekkel (keretekkel) vezérelhető, melyeket különböző soros portokon (alsó rész, fej, kar) szükséges továbbítani.

A robot paramétereit összefoglaló táblázat:

Kerület ütközővel: 757 mm
Kerékátmérő: 101,6 mm
Hasmagasság: 16 mm
Tömeg kb: 40kg
6db távolságérzékelő: 60 fokonként 3db a földtől: 158 mm-re, a középponttól 688/2 mm-re 3db a földtől 201 mm-re, a középponttól 265/2 mm-re
Kar forgástengelye a földtől: 516 mm, a középponttól: 316/2 mm
Kinect magassága a földtől: 805 mm, Kinect függőleges forgástengelye a középponttól 25 mm-re előre Kinect mozgástere: függőleges tengely körül: +/- 90 fok, vízszintes tengely körül a függőlegessel bezárt szög: + - 50 fok
Takarító modul: a robot elejében kb középen, tengelye a robot közepétől 130/2 mm-re takarító tároló térfogata: 325*116 mm^3 összevethető Roomba robottal.

Az alábbi ábrán a robot fényképe látható:



1. ábra Ethon (Etorobot2-ELTE)

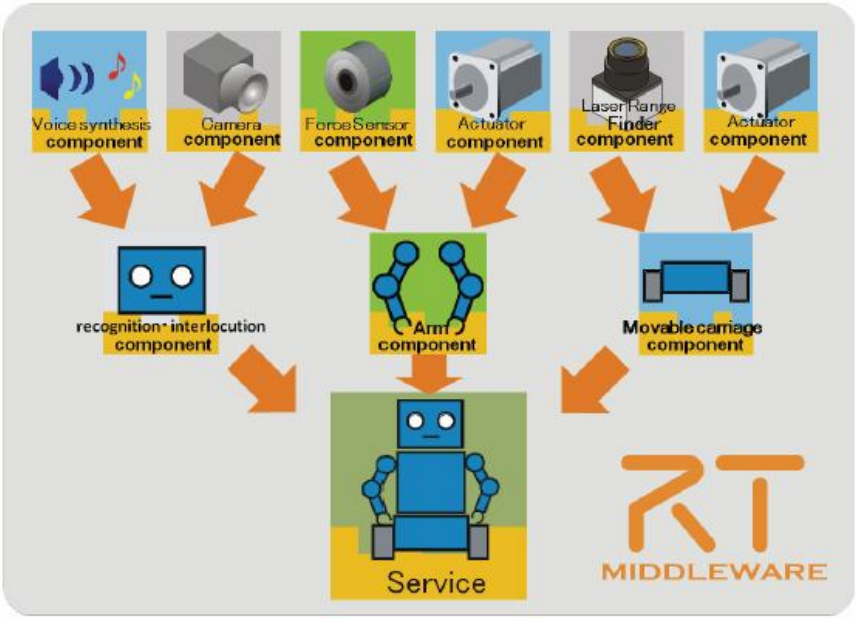
2.3 Robotvezérlő rendszerek

[3]

A robotvezérlő rendszerek célja, hogy egységes felületet biztosítsanak robotok programozására, segítséget nyújtva nagyobb, összetettebb feladatok megvalósításában. Ahogy egyre népszerűbbek lettek a robotokkal kapcsolatos fejlesztések egyre több probléma mutatkozott meg: a robotrendszerek fejlesztése meglehetősen drága, összetett feladat, amely sok időt és specifikus tudást igényel. Ezért terjedtek el a robotvezérlő rendszerek, melyek komponens alapú szemléletükkel lehetőséget biztosítanak ilyen feladatok megvalósításában. A komponens alapú szemlélet szerint minden egyes jól meghatározott funkciót egy adott modul valósít meg, a teljes rendszer pedig több ilyen modul összekapcsolása révén valósul meg.

A robotvezérlő rendszerek főbb funkciói:

- érzékelők, beavatkozók kezelése, integrációja
- vezérlési architektúra létrehozása
- kommunikációs csatornák definiálása
- működés közben adatok gyűjtése
- tesztelés meglévő adatmintákkal



2. ábra Robotvezérlő rendszer lehetséges komponensei (RT-Middleware esetén)

2.3.1 ROS - Robot Operating System

[4][5]

A ROS, egy nyílt forráskódú, amerikai fejlesztésű robot operációs rendszer, amely könyvtárai segítségével lehetővé teszi robot alkalmazások fejlesztését. Előnye, hogy sok beépített funkciót tartalmaz, ilyenek például az arcfelismerés, mozgáskövetés, sztereo látás (több kamera segítségével), robotvezérlés, útvonaltervezés, SLAM, stb. A ROS-ban elékészült alkalmazásokat node-oknak nevezik, melyek közötti kommunikációt maga a ROS valósítja meg. A robotot vezérlő ROS alkalmazás több ilyen komponensből (node-ból) épül fel. Hátrányai közé tartozik, hogy a rendszer felépítése meglehetősen összetett, bonyolult, illetve rendszer telepítése, konfigurálása sem egy pár perces feladat.

2.3.2 OpenRTM-aist

[6][7][8]

Az OpenRTM egy japán fejlesztésű, nyílt forráskódú middleware (köztes réteg) robotrendszer. Ezen middleware az operációs rendszer és az alkalmazás között helyezkedik el, biztosítja a futtató környezetet, futtatható kódot állít elő.

Az OpenRTM lehetőséget biztosít robotok számítógépes hálózaton keresztüli összekapcsolására. Előnyei közé tartozik, hogy mind Linux, mind pedig Windows operációs rendszeren is használható, támogatja a Java, C++ és Python programozási nyelveket, a fejlesztők feladatát pedig magas szintű grafikus eszközökkel segíti. Ezen eszközök segítségével lehetősége van a programozónak megadott paraméterek alapján osztályvázat generálni, illetve az elkészült modulokat egymással összekapcsolni a teljes rendszer működtetése érdekében. Ezen szemlélet szerint az adott feladatot megvalósító rendszer több OpenRTM komponensből áll, melyek közötti kommunikációt maga a keretrendszer valósítja meg, így a komponensek lehetnek azonos hálózati csomópontban vagy különbözőekben is. A távoli komponensek együttműködésére távoli eljáráshívásokat használ.

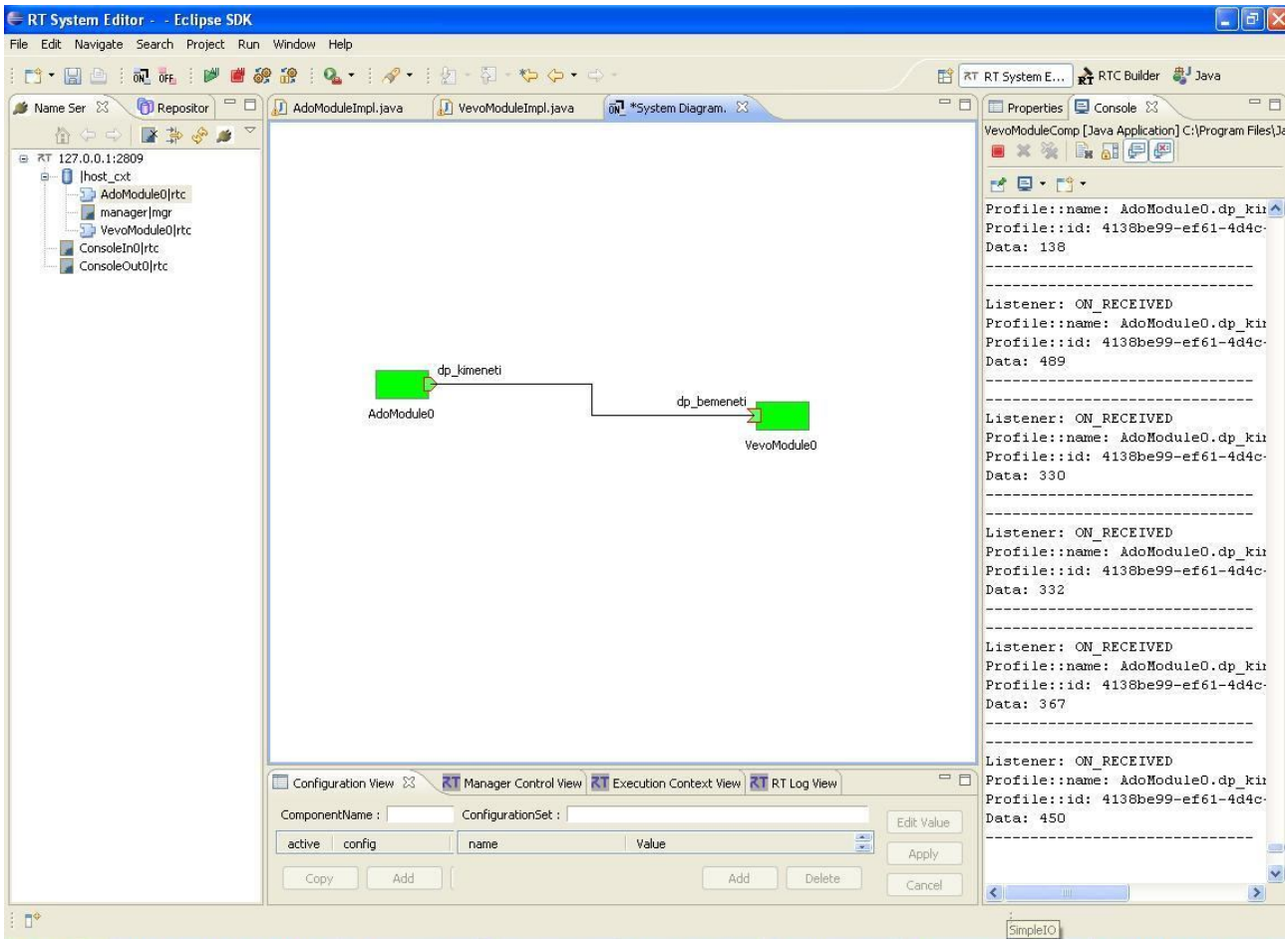
Egy OpenRTM komponens különböző függvényekből, metódusokból épül fel, melyek adott események bekövetkeztekor kerülnek meghívásra. Ilyen függvények pl.: az onInitialize, onStartup, onActivated, onExecute függvények, melyek törzsét a mi feladatunk definiálni az adott feladattól függően. Az onExecute metódusba olyan programkód kerül, amely mindig adott időközönként végrehajtódik, ilyen lehet például a robot távolságérzékelői által szolgáltatott értékek adott időközönkénti lekérdezése.

Néhány segédprogram, amely a fejlesztő feladatát könnyíti:

- *RTC Bulider*: grafikus felületet biztosít komponens létrehozására, a megadott paraméterek alapján projektet, osztályvázat generál. Segítségével megadható a komponens neve, kategóriája, verziója, illetve beállíthatunk adat portokat, szolgáltatás portokat és a használni kívánt programozási nyelvet is. A megadott paraméterek alapján a kiválasztott programozási nyelven ez az eszköz generálja a forráskódot és a szükséges fájlokat, osztályvázakat. Lehetőséget biztosít kódgenerálás utáni paraméter módisításra is, ebben az esetben a módisítások elvégzése után újragenerálja a projekt megfelelő részeit.

- *RT System Editor*: segítségével grafikus felületen irányíthatjuk komponenseink működését, viselkedését, állíthatjuk állapotaikat, illetve összeköthetjük az egyes modulokat egymással. Segítségével grafikus felületen indítható el, állítható le, újraindítható, törölhető az adott modul. Az egyes komponensek állapotainak jelölésére különböző színeket használ, a zöld az aktív állapotot, a piros pedig a hibaállapotot jelöli.

A következő 3. ábrán a szerkesztő felület látható:



3. ábra Az RT System Editor grafikus szerkesztő felülete

Néhány szó a 3. ábrán látható, általam készített demo-ról:

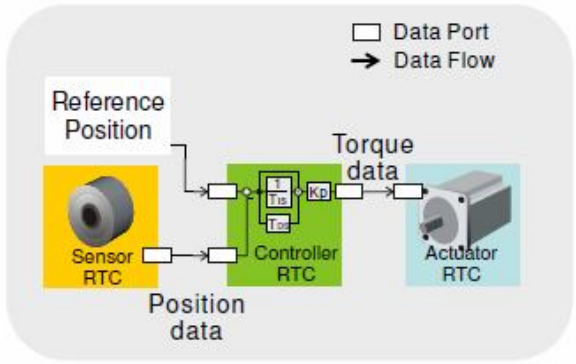
Két komponens látható működés közben (színük zöld --> állapotuk aktív) melyek közötti kommunikációt adatportok valósítják meg. Az egyik komponens az AdoModule0, amely célja, hogy egy véletlen értéket (egész számot 100 és 500 között) generáljon és azt elküldje a VevoModule0-nak. A kommunikáció tehát portok segítségével történik, a dp_kimeneti az adó modul, a dp_bemeneti pedig a vevő modul adatportja. Minkét adatport short típusú, az AdoModule0 a kimeneti portjára kiírja a generált véletlen számot, majd a VevoModule0 a bementi portján fogadja ezt az értéket és kiírja a konzolra. A példa esetében a két komponens helyi gépen fut, de akár a világ két különböző pontján lévő számítógépen is futhatnának.

A komponensek között kommunikáció

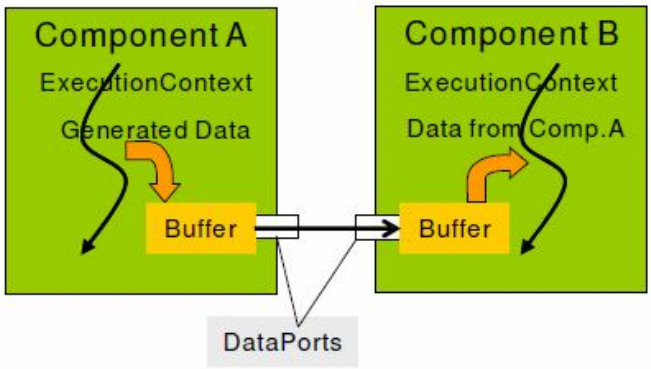
A komponensek közötti kommunikáció megvalósítására a rendszer kétféle lehetőséget, az adat portokat és a szerviz portokat kínálja.

Az adatportok [9]

Az adatportok olyan változókat jelölnek, melyeket egy komponens be- vagy kimenetéül szolgálnak. Ebben az esetében annyi különböző adatportra van szükség, amennyi különböző változó értékét befolyásolni szeretnénk. Az adatportok lehetnek bemeneti és kimeneti portok. A kimeneti adatport egy komponens kimenetéül, a bemeneti adatport pedig egy komponens bemenetéül szolgál. Például ha vezérelnünk kell egy motor fordulatszámát és egy LED ki/bekapcsolását, akkor a felhasználói felület komponensének van két int típusú kimeneti adatport, egy a fordulatszámra (egész szám), egy pedig a led működtetésére (szintén egész szám legyen, 0 mikor kikapcsoljuk, 1 ha bekapcsoljuk). A motor vezérlését és a LED működtetését megvalósító komponensnek pedig kettő, szintén int típusú bemeneti adatportja van. A 4. és 5. ábrák két komponens közötti kommunikáció megvalósítását szemlélteti adatportok segítségével. A két komponens, a szenzor (érzékelő szerv) és a motor (beavatkozó szerv) egy-egy adatporttal, a szenzor kimeneti a motor pedig bemeneti porttal rendelkezik.



4. ábra Példa adatport alkalmazására



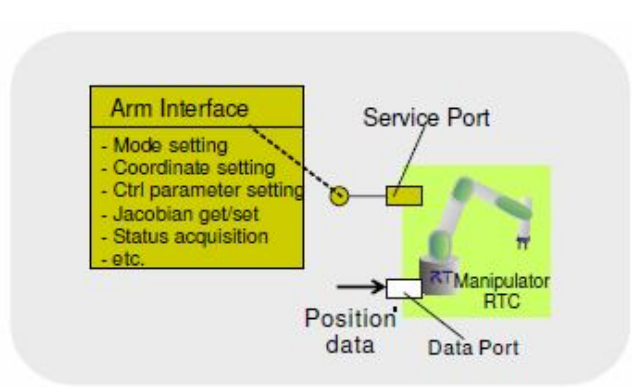
5. ábra Adatport két komponens között

A szervíz portok [9]

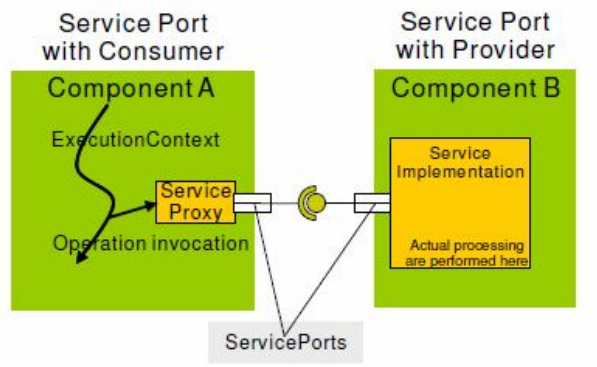
A szerviz portok két típusát, a szolgáltatót és a fogyasztót különböztetjük meg. Mindkét esetben interfészeken keresztül szolgáltatnak funkciókat. Nagy különbség az adatportokhoz képest, hogy itt nem ki- és bemeneti változókat valósítanak meg, hanem a távoli osztály metódusait használhatjuk úgy, mint ha azok helyben lennének. Az OpenRTM interfészek

definiálásra az IDL (Interface Definition Languange)-t használja. Ebben az IDL kiterjesztésű fájlban megadott metódusok lesznek az a metódusok, melyek távolról hívhatók úgy, mint ha azok helyben lennének. Ezen IDL fájl alapján az OpenRTM szerkesztő felülete generálja a szükséges osztályokat, osztályvázakat.

A 6. és 7. ábrák két komponens közötti kommunikáció megvalósítását szemlélteti szervízportok segítségével.



6. ábra Példa szervízport használatára



7. ábra Szervízport két komponens között

3. A megvalósítás

Ebben a fejezetben a robotvezérlő szoftver célját, tervezését, implementálást, felépítését és használatát ismertetem.

3.1 A kitűzött cél

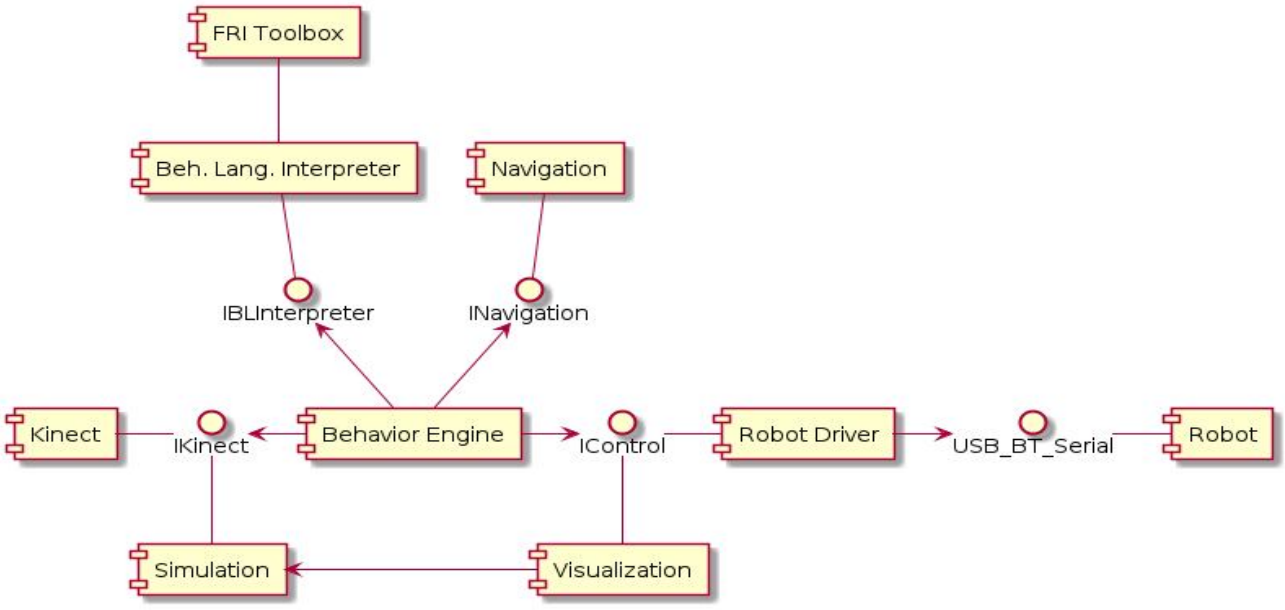
Célom az RT-Middlreware rendszer részeként megvalósított olyan komponensekből álló szoftver tervezése, implenetálása, tesztelése, amely lehetősvé teszi Ethon távirányítását. A megvalósításhoz első lépésben egy, az RT-Middleware-től független, a fentebb említett feladatot megvalósító alklamazást tervezek, melyet majd azt követően alakítok az OpenRTM rendszer részét képező modulokká.

A robot távvezérlése alatt a robot valamennyi funkciójának működtetését, azaz a mozgás vezérlését, a fejének és karjának mozgatását, egyes egységeinek működtetését, a robot által küldött értékek (pl.: pozíció, érzékelők által szolgáltatott távolság érték) feldolgozását értem.

A tervezendő és megvalósítandó szoftver célja a fentebb említett funkciók megvalósítása, a Kinect által szolgáltatott kamerakép megjelenítése illetve mindehhez grafikus felhasználói felület biztosítása.

3.2 A teljes vezérlőrendszer felépítése

A 2. ábra a teljes vezérlőrendszer moduljait és a közöttük lévő kapcsolatokat szemlélteti:



8. ábra A rendszer moduljai

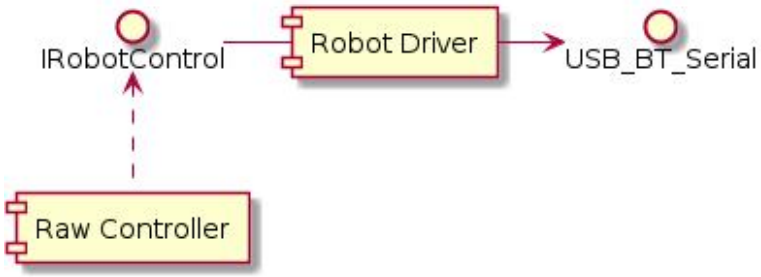
Az egyes modulok és feladatai:

Modul	Funkció
Kinect	Magát a Kinect eszközt szemlélteti.
Simulation	Szimulációt megvalósító komponens.
Visulation	Megjelenítést megvalósító modul.
RobotDriver	A robot közvetlen vezérlést megvalósító komponens, utasításkeretek küldését végzi.
Navigation	Útvonaltervezést, navigálást lehetővé tevő modul.
Behavior Engine	A szituációnak és a belső állapotnak megfelelően dönt arról, hogy milyen cselekvést kell végrehajtania, milyen viselkedéskomponens az éppen aktuális. Ez tárolja a viselkedéshez tartozó belső állapotokat is.

A megvalósítás keretében a teljes rendszer egy adott feladatot megvalósító részét implementálom.

3.3 A megvalósítandó rendszer felépítése

A 9. ábra a távvezérlést megvalósító, az OpenRTM rendszer részeként definiált komponenseket és azok közötti kapcsolatokat szemlélteti:



9. ábra A távvezérlést lehetővé tévő OpenRTM modul felépítése

Modul	Funkció
robotDriver	A robot vezérléséhez szükséges utasításkeretek összeállítását és adott soros portra való küldését végzi.
rawController	Konzolos illetve grafikus felhasználói felületet biztosít melyeken keresztül kényelmesen irányítható a robot. Lehetőséget biztosít a robot vezérlése és a robot valamennyi funkciójának működtetésére

3.4 A robot vezérlése

A robot konkrét, előre meghatározott utasításkeretek segítségével vezérelhető, a keretek felépítését a 10. és a 11. ábra szemlélteti.

Data frame types											
Function	R/W	0. byte (= Addr)	1. byte	2. byte	3. byte	4. byte	5. byte	6. byte	7. byte	8. byte	9. by
status register	R	0x00	status	ancknowledge+ID	*CheckSUM*						
configuration register	R/W	0x01	config	*CheckSUM*							
current position of robot	R/W	0x02	[MSB]	X pos		[LSB]	[MSB]	Y pos		[LSB]	[MSB] D
reference position of robot	R/W	0x03	[MSB]	X pos		[LSB]	[MSB]	Y pos		[LSB]	[MSB] D
max velocity and acceleration	R/W	0x04	Max velocity	lax angular velocit	Max accel	Max angular accel	*CheckSUM*				
distance sensors feedback	R	0x05	Sensor 1.	Sensor 2.	Sensor 3.	Sensor 4.	Sensor 5.	Sensor 6.	*CheckSUM*		
relay outputs	W	0x06	relay 1 | relay 0	*CheckSUM*							
Data request	W	0x07	Addr	*CheckSUM*							
BlueBots reference velocity	W	125	X velocity	Y velocity	Angular velocity	Max velocity	*CheckSUM*				
BlueBots buttons	W	127	1,2 or 3 button	dummy(0)	*CheckSUM*						

Position scales	Data type	Unit	Minimum	Maximum
position unit:	32 bit int	[mm]	-2^31	2^31-1
direction unit:	16 bit int	0...360 deg	0	360
max velocity unit:	8 bit uint	8x [mm/sec]	0	
max ang vel. unit	8 bit uint	2x [deg/sec]	0	
max acceleration unit	8 bit uint	8,2x [mm/sec^2]	0	
max ang accel unit	8 bit uint	4,1x [deg/sec^2]	0	

CheckSUM calculation:
CheckSUM = (1. byte) + (2. byte) +....+ (n-1. byte)

10. ábra Az alsórész utasításkereteinek felépítése

Data frame types						
function	R/W	0. byte (= Addr)	1. byte	2. byte	3. byte	4. byte
status register	R	0x00	status	Acknowledge+ID	CheckSUM	
enable motor 1 & motor 2	R/W	0x01	enable (0/1)	CheckSUM		
motor 1 pos reference	W	0x02	0-255	CheckSUM		
motor 1 speed & acc	W	0x03	speed 0-255	acc 0-255	CheckSUM	
motor 2 pos reference	W	0x04	0-255	CheckSUM		
motor 2 speed & acc	W	0x05	speed 0-255	acc 0-255	CheckSUM	
motor 1 position	R	0x06	limit1)-0-255-(limit2	CheckSUM		
motor 2 position	R	0x07	limit1)-0-255-(limit2	CheckSUM		

Data request:	1. byte	2. byte	3. byte	
	0x08	Addr	CheckSUM	Robot send data frames only upon data request.

	data type	unit
axis 1 position unit:	8 bit uint	0-140°
axis 2 position unit	8 bit uint	0-270°

11. ábra A felsőrész utasításkereteinek felépítése

Az alsórészhez egyetlen soros port tartozik, tehát ezen kereteket arra szükséges küldeni. Vezérlés esetében cím+adatbyte+checksum, regiszter lekérdezés pedig 0x07+regiszter cím+checksum a keretformátum. Regiszter lekérdezés esetén tehát először egy data request keretet szükséges küldeni (adott címmel), melyre azt követően küldi vissza a választ a táblázatban lévő formában. A checksum a keretben lévő bájtok összegét jelenti, ha a checksum helyes, akkor végrehajtja az adott utasítást, ha helytelen akkor nem hajtja végre az utasítást, hanem egy „Frame lost the byte" választ küld.
Például ha a státusz regiszter szeretném lekérdezni, akkor a következő keretet kell kiküldeni a soros portra: 0x07 0x00 0x07. Erre a választ 0x07 status ancknowledge+ID checksum formában adja vissza.

A robot helyváltoztatására kétféle lehetőség kínálkozik:

- „BlueBots"-os vezérlés
- Pozícióvezérlés

BlueBots irányítás a táblázatban található „BlueBots reference velocity" nevezetű keret küldésével valósítható meg. Ez azt jelenti, hogy a robot kap x, y pozíciót, szöggyorsulást, és sebesség referenciát, majd ennek hatására az adott x, y pozícióba, az adott szögelfordulással és sebességgel elmozdul.
Másik lehetőség a referencia és a célpozíció megadása. Ez a táblázatban található „reference posotion of robot" és a „current position of robot" keretek segítségével adható meg. Ezt a működési módot előbb engedélyezni kell, melyhez a 0x01 0x01 checksum utasítás tartozik. Ezt követően az aktuális pozíció megadása szükséges (általában 0), melyhez képest a robot elmozduljon. Koordináta, szögelfordulás, gyorsulás és sebesség megadása után Ethon az adott x, y pozícióba indul, úgy, hogy az éppen x, y koordinátába éréskor fejezi be az elfordulást is, tehát egyszerre halad az adott irányba és közben fordul az adott szögben. Ez a mikorvezérlő programjában található trajektória tervezésnek köszönhetően működik.

A felsőrészhez két sorosport is tartozik, melyek mögött egy/egy darab mikorvezérlő helyezkedik el. Külön mikrovezérlő irányítja tehát a fej és a kéz működését is. A vezérlésükhöz szükséges keretformátum a fentebb található 5. ábrán látható. Elvi felépítése hasonló az alsórész kereteinek felépítésével, csak a címek eltérőek. Például ha a azt szeretném, hogy a kéz adott pozícióba mozduljon, akkor a 0x04 0-255 checksum keret küldése szükséges, ahol a 0-255 (0: legalsó végállás, 255: legfelső végállás) jelöli a pozíciót.

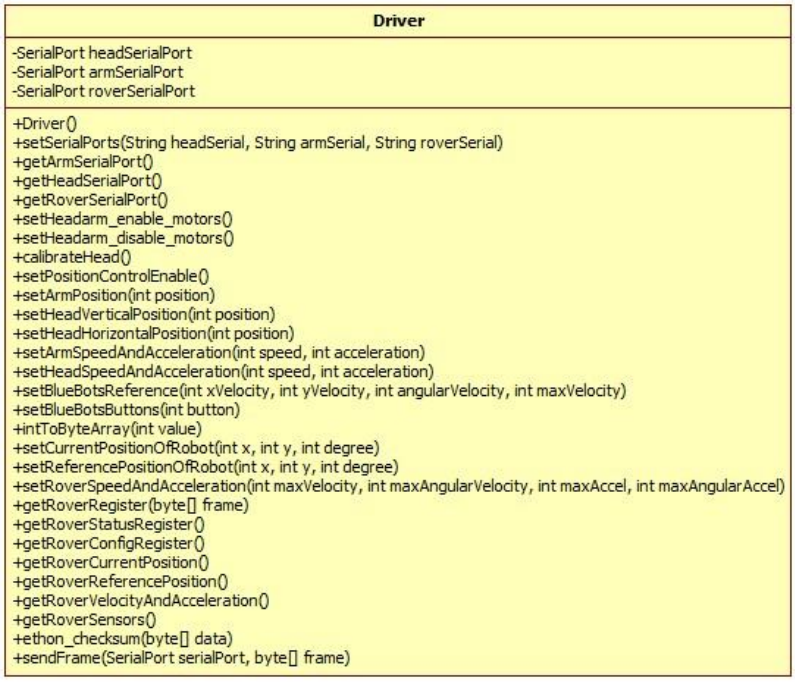
3.5 A robotDriver osztály

A „robotDriver” osztály feladata, hogy magasabb szintű utasításokat (pl.: menj előre 10 métert) átfordítsa a robot nyelvére, azaz a magas szintű utasításnak megfelelő keretet/kereteket összeállítsa és elküldje a robot adott részének (fej, kar, vagy alsórész) szóló soros portra.

Az implementálás Java nyelven történt, ez a modul tulajdonképpen egy Java osztály, melynek megfelelő metódusait használva (pl.: setArmPosition(int position)) vezérelhető Ethon adott része, illetve lekérdezhetőek a robot által visszaadott értékek (pl.: távolságok, státusz regiszter, pozíciok, stb.) Az osztály adott metódusai tehát adott soros portokra küldenek, illetve soros portról olvasnak és értelmeznek kereteket. Mind az alsórésznek, mind a fejnek és a kéznek szóló utasításkeretek különböznek.

Az osztálybeli soros portok beállításai:

- Baudrate: 115 200
- Adatbitek: 8
- Stop bitek: 1
- Paritás: 0
- Flow control: nincs
- soros portok Driver osztálybeli elnevezései:
  - headSerialPort
  - armSerialPort
  - roverSerialPort



12. ábra A robotot vezérlő "Driver" osztály felépítése

3.6 Az Avatar, mint különálló szoftver

Az Avatar a „Driver” osztály felhasználásával megvalósított, az OpenRTM rendszertől függetlenül használhtó szoftver, amely célja Ethon valamennyi funkciójának grafikus felületen való vezérlése. Ez alatt értendő a robot helyváltoztatása, a fej és a kar mozgatása, a „seprű” és a labdakilövő egység működtetése, a kamerakép megjelenítése, illetve a robot által küldött értékek feldolgozása (pl. érzékelők távolság értékei). Ezen szoftver implementálásához szintén a Java programozási nyelvet választom. A megvalósítás két verzióban történik, az egyik egy „localhost”-os verzió, amelyet Ethon-on futtatva lehet használni, a másik pedig egy hálózati verzió, amely segítségével távvezérelhető a robot. A „localhost”-os változat egyetlen szoftvert jelent, a hálózati változathoz pedig szükséges egy szerver és kliens oldali alkalmazás. A továbbiakban ezen verziók megvalósítását ismertetem.

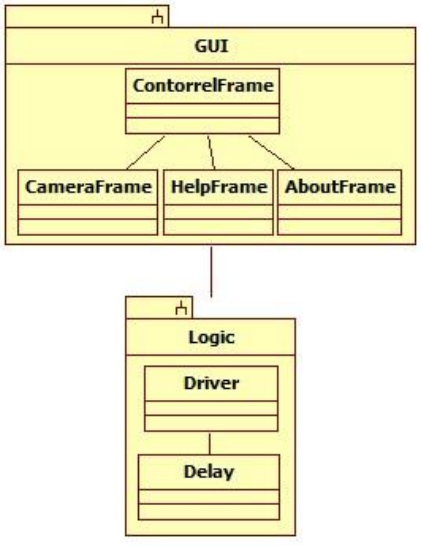
3.6.1 A „localhost”-os verzió

Ezen verzió Ethon számítógépén működtethető, a felhasználói felületen lévő megfelelő paraméterek beállítása után.

Ezen változat funkciói:

- mozgás vezérlése (előre/hátra, jobbra/balra, oldalazás)
- kéz mozgatása (fel/le)
- fej mozgatása (minden irányban)
- takarítóegység és a labdázó egység működtetése
- kamerakép megjelenítése

A következő ábrán ezen változat felépítése látható:



13. ábra A "localhost" verzió feléptése

Logikailag két részre, a felhasználói felületre (GUI) és a vezérlésre (Logic) bontható. A GUI felületén lévő komponensek által kiváltott események alapján történik vezérlés küldése a logikának. A GUI-n kiváltott eseményeket (pl.: előre mozgás, fejmozgás) a logika fordítja a robot nyelvére, azaz Ethon számára értelmezhető parancsokat továbbít a robot mikorvezérlőinek.

A „GUI" osztályai és azok funkciói:

Osztály	**Funkció**
ControllerFrame	A program főablaka, ezen ablakban találhatóak az eseményt kiváltó komponensek (pl.: előre/hátra mozgást vezérlő gombok, fej és kéz működtetés)
CameraFrame	Célja a Kinect által szolgáltatott kamerakép megjelenítése
HelpFrame	A program használatáról nyújt információkat
AboutFrame	A szoftver készítőjéről szolgáltat információt

A „Logic" osztályai és azok funkciói:

Osztály	**Funkció**
Driver	Feladata a felhasználói felületen lévő komponensek által kiváltott események alapján vezérlés küldése a robot számára (keret összeállítás majd továbbítás/olvasás soros portokon)
Delay	Feladata a Driver osztályban megvalósított keret küldés és keret olvasás között eltelt idő beállítása regiszter lekérdezés esetén. (Keret küldése soros porton, ~20ms várakozás, válasz olvasása a soros porton)

A következő ábra a programot szemlélteti működés közben:

14. ábra Az Avatar "localhost"-os verziója

A fej, kar és alsórész sorosport-jainak megadása, valamint a „Serial OK” gombra való kattintás után vezérlehető a robot. A jobb oldali négyzetben mozgatva az egeret az egérmutató mozgásának megfelelően mozog Ethon feje, az egér görgőjével pedig a robot karja mozgatható.

Pozícióváltoztatáshoz a „w”, „s”, „a”, „d” és az „x”, „y” billentyűkkel van lehetőség a sebesség és gyorsulás beállítását követően. A sebesség és a gyorsulás együtt változtatható a „+” és „-” billentyűkkel, alapértelmezett értéke a 120-as skálán 10. A seprű és a labdakilövő egység az „e” és „r” billentyűkkel hozható működésbe, majd ezen billentyűk ismételt megnyomása esetén állíthatóak le.

A kamerakép megjelenítése egy külön - a bal oldalt látható – ablakban történik. Ezen ablak tetszőlegesen átméretezhető, illetve bezárható/előhívható. Ebben az esetben ezen szoftver helyben fut, így a kamerakép késése minimális, alig észrevehető.

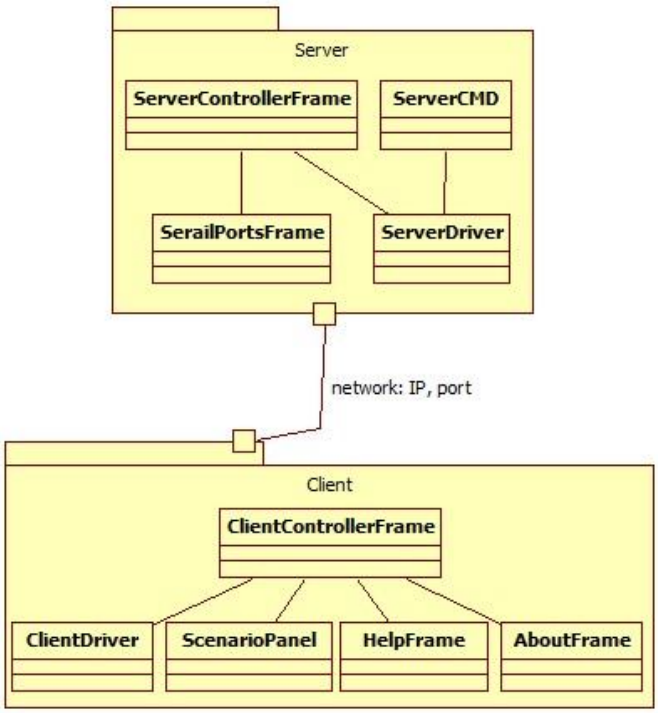
3.6.2 A hálózati verzió

Ezen verzió célja, hogy tényleges megvalósítsa Ethon távvezérlését (nem a helyi gépen!). Fő funkciói megegyeznek a „localhost”-os verzióéval, eltérések legfeljebb a paraméterezésben lehetnek (pl.: nem soros port hanem IP cím és port megadás), kliens-szerver modell használ, mely következtében a kommunikáció a kliens program és a szerver program közötti számítógépes hálózaton történik. A szerver oldali alkalmazás Ethon számítógépén fut, a kliens szoftver azonban egy tetszőleges számítógépen futtatható.

Ezen verzió funkciói:

- mozgás vezérlése (előre/hátra, jobbra/balra, oldalazás)
- kéz mozgatása (fel/le)
- fej mozgatása (minden irányban)
- takarítóegység és a labdázó egység működtetése
- mozgás rögzítés és visszajátszás
- kamerakép átvitele

A következő ábrán ezen változat blokkvázlata látható:



15. ábra A hálózati verzió blokkvázlata

Látható, hogy egy szerver és egy kliens oldali alkalmazást foglal magába, melyek közötti kommunikáció adott IP címen és porton történik. A kliens magasabb szintű szöveges parancsokat továbbít a szerver felé (pl.: setArmPosition 50), melyeket a szerver fogadja, majd átfordítva tovább küldi az adott soros portra.

A szerver

A szerver oldali alkalmazás osztályai és azok funkciói:

Osztály	**Funkció**
ServerControllerFrame	Az alkalmazás grafikus felhasználói felülete, segítségével felhasználóbarát módon működtethető a szerver.
ServerCMD	A szerver oldali alkalmazás parancssori változata, grafikus felhasználói felülete nincs, működése parancssorból vezérelhető. (a soros portok fixen beállítottak [egyelőre], távolról is újraindítható!)
SerialPortsFrame	A soros portok beállítását megvalósító felhasználói felület, a 3 darab port (fej, kar rover) megadása után lesz működőképes a szerver .
ServerDriver	Az előzőekben részletesen tárgyalt „Driver” osztály „ServerDriver” néven.

A következő ábrákon a szerver oldali alkalmazás GUI-val ellátott verziója látható működés közben:

16. ábra Soros portok beállítása a szerveren

17. ábra A szerver alkalmazás működés közben

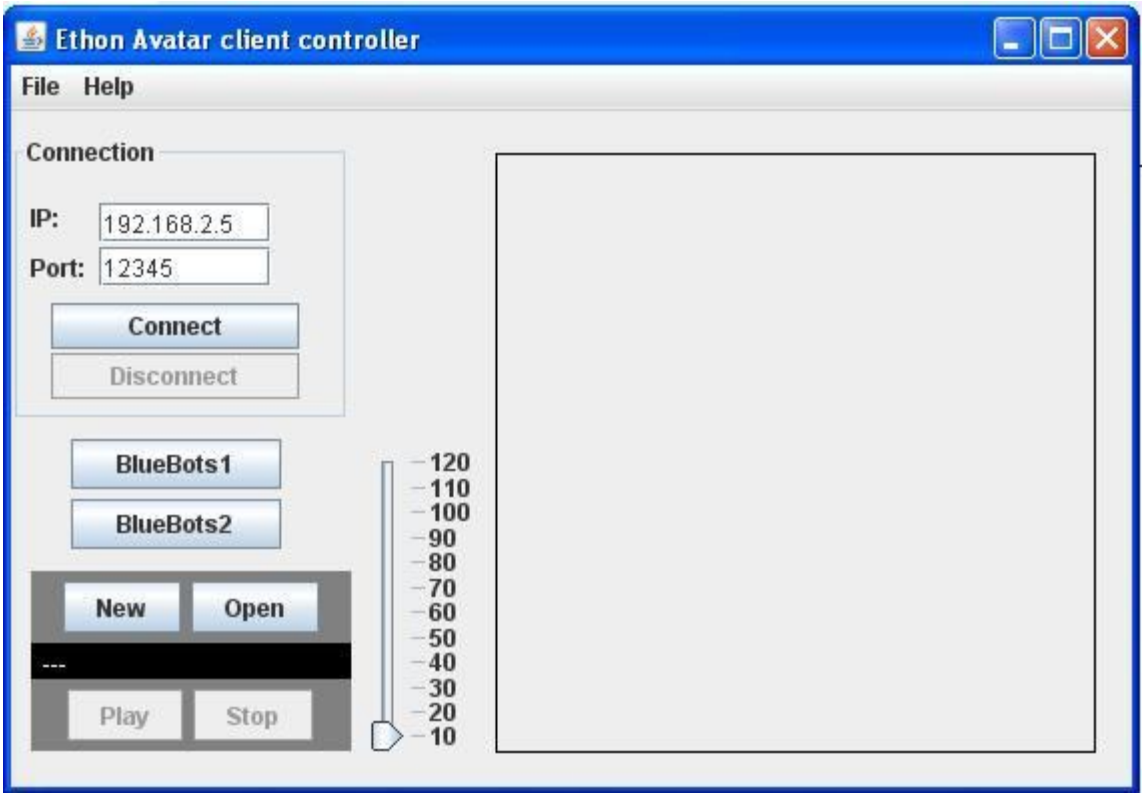
A kliens

A kliens oldali alkalmazás osztályai és azok funkciói:

Osztály	Funkció
ClientControllerFrame	Az alkalmazás grafikus felhasználói felülete, az általa megvalósított GUI komponensek (nyomógombok, billentyűk, canvas) segítségével kényelmesen vezérelhető a robot

ClientDriver	A robot vezérléséhez szükséges magasabb szintű szöveges utasítások küldését valósítja meg hálózaton keresztül. Ezen szöveges utasításokat a szerver oldali alkalmazás fogadja és dolgozza fel.
ScenarioPanel	A robot mozgásának felvételét/visszajátszását megvalósító felület.
HelpFrame	A program használatáról információkat nyújtó ablak.
AboutFrame	A szoftver készítőjéről információkat szolgáltató panel.

A következő képen a kliens szoftver felhasználói felülete látható:



18. ábra A kliens alkalmazás felülete

A bal oldalon lévő beviteli mezőkben van lehetőség a szerver IP címének és portjának megadására, majd a „Connect” gombra való kattintás után jön létre a kapcsolat a robottal. A robot pozíciójának változatására szintén billentyűkkel van lehetőség (w, a, s, d, x, y) ugyanúgy, mint a „localhost”-os verzió esetében. A takarító egység és a labdakilövő egység

az „e” és „r” billentyűkkel működtethető, illetve a „BlueBots1” és a „BlueBots2” gombok is lehetőségre állnak. A robot mozgásának felvételére és visszajátszására a „BlueBots” gombok alatt lévő panel biztosít lehetőséget. A mozgáshoz tartozó utasítások egy szöveges fájlba íródnak, mely elmenthető, illetve betölthető. A 10-120-as számozású csúszkán a robot sebessége állítható, de a „+” és „-” billentyűk segítségével is módosítható. A robot fejének pozíciója (azaz a Kinect helyzete) a jobb oldalt lévő négyzetben változtatható az egér mozgatásának segítségével, a robot karja pedig az egér görgőjével mozgatható (fel-le irányokban).

3.7 Az Avatar, mint OpenRTM komponens

Ezen alfejezet keretében ismertetem a robotvezérlő rendszerektől függetlenül működtethető Avatar szoftver OpenRTM komponenssé alakítását. Az Avatar OpenRTM komponenssé való átalakítását az OpenRTM adta előnyök, többek között a rendszer egyes elemei közötti kommunikáció megvalósítása indokolja, illetve a teljes Ethon vezérlő rendszer is az OpenRTM keretrendszer keretében fog megvalósulni. Az Avatar OpenRTM komponensként való működtetéséhez két modulra, a robotDriver-re és a GUI-ra (rawController) van szükség. A két komponens közötti kommunikáció szerviz porton keresztül valósul majd meg.

3.7.1 A ControllerService IDL fájl

A komponensek közötti kapcsolattartásért a CORBA felelős, ez a kapcsolattartás pedig szerviz portok esetén interfészeken keresztül valósul meg. Az IDL (Interface Definition Language) fájl szolgál az interfész leírására. Ezen fájl hordozható, programozási nyelvtől és operációs rendszertől független. Az IDL támogatja a típusok, konstansok, adatelemek, metódusok, kivételek deklarációját, de nem foglalkozik azzal, hogy a definiált metódusokat hogyan kell implementálni, tehát metódus vázakat tartalmaz, de a metódus törzsek implementálása egy másik állományban történik. Ezen állományt (vagy állományokat) az IDL fordító hozza létra a paraméterében megadott IDL fájl alapján. Az IDL fordítókat az egyes ORB gyártók biztosítják, én jelen esetben a JacORB-t használtam. [10]

A ContorllerService.idl Ethon Java-s driver-ének nyilvános metódusait tartalmazza, tehát ebben a fájlban definiált metódusok távolról hívhatóak, a felhasználói felület e segítségével hívja a driver osztály megfelelő metódusait, miközben a driver komponens egy más hálózati csomópontban fut, mint a felhasználói felület komponense. A driver és a felhasználói felület komponens (a kliens és a szerver) ugyanazt az IDL fájlt használja.

A ConrollerService.idl fájl szerkezete:

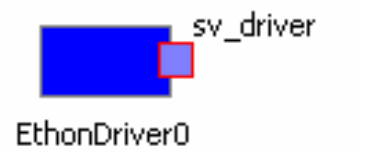
```
module EthonService {
   interface ControllerService {
            void enableMotors();
             void disableMotors();

      void calibrateHead();
      void enablePositionControl();
      void setArmPosition(in short position);
      void setHeadVerticalPosition(in short position);
      void setHeadHorizontalPosition(in short position);
      void setArmSpeedAndAcceleration(in short speed, in short acceleration);
      void setHeadSpeedAndAcceleration(in short speed, in short acceleration);
      void setSpeeds(in short x, in short y, in short angular, in short max);
      void setBlueBotsButtons(in short button);
      void setBlueBotsReference(in short xVelocity, in short yVelocity, in short angularVelocity, in short maxVelocity);
      void setCurrentPositionOfRobot(in short x, in  short y, in short degree);
      void setReferencePositionOfRobot(in short x, in short y, in short degree);
      void setRoverSpeedAndAcceleration(in short maxVelocity, in short maxAngularVelocity, in short maxAccel, in short maxAngularAccel);
   };
};
```

3.7.2 A robotDriver komponens

Ezen komponens feladata megegyezik a 3.5 fejezetben tárgyalt robotDriver osztály feladatával, tulajdonképpen azon osztály felhasználásával létrejött modult jelneti. Célja, hogy a felhasználói felületen kiváltott eseményeket átfordítsa a robot nyelvére, azaz az eseménynek megfelelő üzenetkeretet küldjön a robot megfelelő sorosportjára.

Például: felhasználói felületen kiváltott esemény: takarítóegység működtetése (egy nyomógomb segítségével) --> a driver modul által ezen esemény alapján a megfelelő sorosportra küldött üzenetkeret, amely a takarítóegységet működteti: „127 1 0 checksum”.

A 19. ábrán az EthonDriver0 a komponens nevét, az sv_driver pedig a szervízportját jelöli.



19. ábra A robotDriver komponenst jelölő szimbólum

A komponenst alkotó fontosabb osztályok és funkcióik:

Osztály	Funkció
ControllerServiceSVC_impl	Implementálja az IDL fájlban definiált interfészt, megvalósítja a metódusok törzsét, tulajdonképpen ez maga a driver osztály.
EthonDriverImpl	A komponens onActivated, onExecute, onDeactivated metódusait tartalmazza.
EthonDriverComp	Main metódust tartalmazó osztály, ezzel futtatható a komponens.

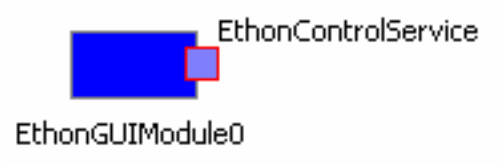
További, a komponenst alkotó osztályok felsorolása:

- ControllerServicem
- ControllerServiceHelper
- ControllerServiceHolder
- ControllerServiceOperations
- ControllerServicePOA
- ControllerServicePOATie
- EthonDriver
- Position
- PositionHelper
- PositionHolder

3.7.3 A rawController komponens

Ezen komponens valósítja a robot vezérlését lehetővé tevő felhasználói felületet és konzolablakot. Két változatban készült el, a konzolos verzióban egyszerű billentyűlenyomások segítségével irányítható a robot, és egy grafikus felhasználói felületet biztosító verzióban, ahol a felületen lévő nyomógombok, vezérlőpanelek segítségével működtethető Ethon valamennyi funkciója.

A 20. ábra a komponens szimbólumát jelöli, ahol az EthonGUIModule0 a komponens neve, az EthonControlService pedig a komponens szervízportja.



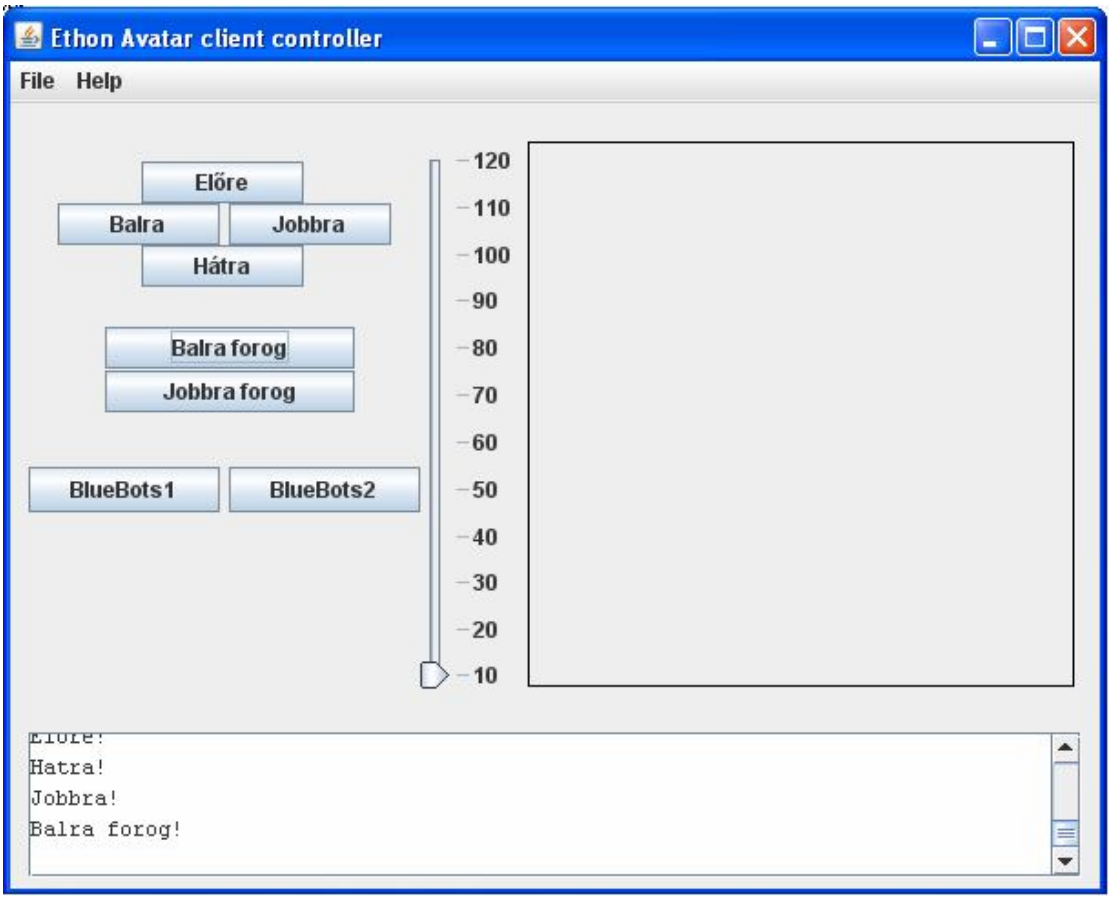
20. ábra A rawController komponenst jelölő szimbólum

A komponenst alkotó fontosabb osztályok és funkcióik:

Osztály	Funkció
EthonGUIModuleImpl	A komponens onActivated, onExecute, onDeactivated metódusait tartalmazza.
EthonGUIModuleComp	Main metódust tartalmazó osztály, ezzel futtatható a komponens.
ClientControllerFrame	A felhasználói felületet implementáló osztály.

További, a komponenst alkotó osztályok felsorolása:

- _ControllerServiceStub
- ControllerService
- ControllerServiceHelper
- ControllerServiceHolder
- ControllerServiceOperations
- ControllerServicePOA
- ControllerServicePOATie
- EthonGUIModule



21. ábra A rawController komponens felhasználói felülete

A felhasználói felület felépítése kisebb-nagyobb eltérésekkel megegyezik az OpenRTM-től függetlenül működtethető Avatar hálózati verziójának felhasználói felületével. Ethon irányítására ugyanazon nyomógombok, billentyűk használhatóak, eltérés, hogy ezen változat kiegészült egy napló ablakkal (a felület alján látható szövegdoboz), melyben a végrehajtott események kerülnek kiírása (pl.: fejmozgatás esetén „Fej mozog!”), illetve már nincs lehetőség a hálózati kapcsolat beállításához szükséges IP cím és port páros megadására, ugyanis ezt már az OpenRTM valósítja meg.

A felhasználói felület által megvalósított funkcióik:

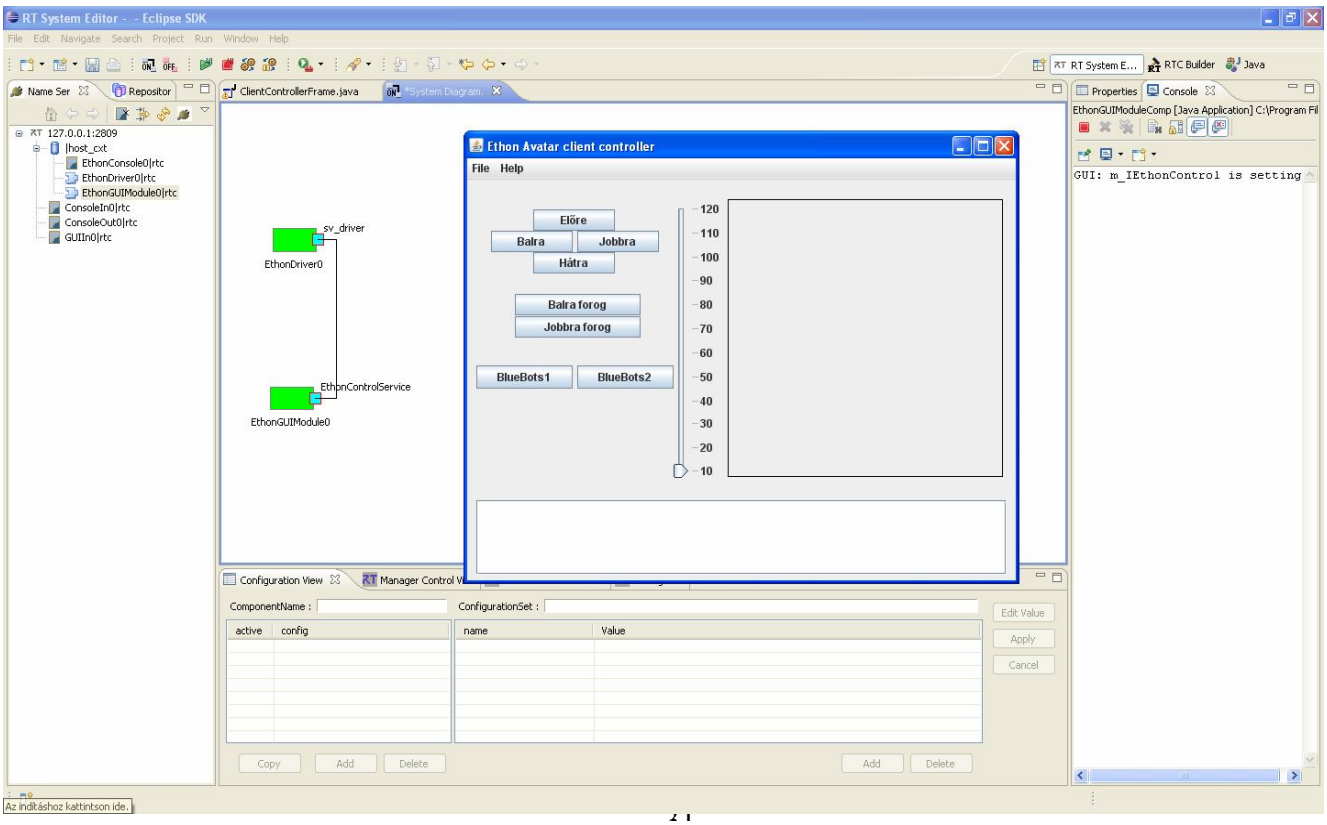
- Robot pozíciójának változtatása (előre, hátra, jobbra, balra, balra forgás, jobbra forgás)
- Pozíció változtatás sebességének állítása (10-120 skálán)
- Takarító egység, labdázó egység működtetése
- Fej (Kinect) mozgatása egér segítségével (jobb oldalt található kis négyzetben)
- Kar mozgatása egérgörgő segítségével (fel/le mozgás)
- Fej kalibrálása (reset)

3.7.4 A rendszer használata

Az elkészült szoftverkomponensek használatához szükséges rendszerösszetevők:

- Oracle JDK (1.6-os verzió!)
- JacORB (névszerver)
- OpenRTM System Editor
- Windows vagy Linux operációs rendszer

Az elkészült komponenseket vagy a .bat (Windows-on) és a .sh (Linux-on) kiterjesztésű fájlokkal, vagy pedig az Eclipse (projekt importálása) segítségével lehetséges futtatni.
A futtatás előtt mindenképp szükséges elindítani a névszervert, jelent esetben a JacORB-t. Ezt az ns –DOAPort=2809 parancssorból kiadott utasítás teszi lehetővé, ahol a 2809 a portszámot jelöli.
A komponensek elindítása után van szükség az OpenRTM System Editor-ára, mely vagy az Eclipse-be ágyazott OpenRTM-ből vagy pedig az OpenRTM feltelepítése által érhető el. Ha a komponensek sikeresen elindultak, akkor a System Editor ablakában kis ikonként jelennek meg. Ezt követően a komponenseket hozzá kell adni a System Editor szerkesztőfelületéhez és a megfelelő portjaikat (jelen esetben 1-1 portot) össze kell kötni egymással. Ezt követően működésük az all activated parancs segítségével aktiválható, sikeres aktiválás esetén a komponensek színe zöldre változik és megjelenik a robot vezérlését lehetővé tevő felhasználói felület.



22. ábra A „robotDriver” és a „rawController” komponens összekapcsolása, működése

3.7.5 A rendszer működésének tesztelése

A rendszer tesztelése alatt az egyes szoftverkomponensek működésének vizsgálatát, a helyes működés ellenőrzését értem. Ez különösen fontos, hiszen a „robotDriver” komponens ad utasításokat robotnak, így szükséges annak ellenőrzése, hogy a kiadott parancsokra Ethon ténylegesen úgy reagál-e, mint amit elvárunk tőle, ha pedig működése helytelen, akkor azt mi okozza. A továbbiakban a tesztelés lépéseit szemléltetem.

A tesztelés információi:

- Használt operációs rendszer: Ubuntu 12.04.
- Alkalmazott fejlesztő- és futtatókörnyezet: Eclipse Ganymede 3.4.2.
- Alkalmazott sorosport kezelő csomag: JSSC 0.9.0
- Java verzió: Oracle JDK 1.6
- RT-Middleware verzió: OpenRTM 1.1
- Névszerver verzió: JacORB 3.3

A „robotDriver” komponens és a „rawContorller” komponens is, helyben Ethon számítógépén futott a teszt alatt.

A „robotDriver” komponens funkcióinak teszteredménye:

Metódus	**Elvárt működés**	**Tényleges működés**
enableMotors()	Fej és kar motorjainak engedélyezése.	Fej és kar motorjai működése engedélyeződött.
disableMotors()	Fej és kar motorjainak letiltása.	Fej és kar motorjai működése letiltott.
calibrateHead()	Fej reset-elése, azaz a robot bekapcsolásakor végbemenő végpozíció megkeresése majd alappozícióba állás.	A fej reset-elt. A parancs hatására sikeresen végrehajtotta a robot bekapcsolásakor lejátszódó fej reset-elési folyamatot.

setArmPosition(short position)	A paraméterben megadott pozícióba állítja a robotkezet.	Hatására a paraméterben megadott pozícióba (egész szám 0-255 között) mozdult a kéz.
setHeadVerticalPosition(short position)	A fej paraméterben megadott függőleges pozícióba áll.	Hatására a fej a paraméterben megadott függőleges pozícióba (egész szám 0-255 között) mozdult.
setHeadHorizontalPosition(short position)	A fej paraméterben megadott vízszintes pozícióba áll.	Hatására a fej a paraméterben megadott vízszintes pozícióba (egész szám 0-255 között) mozdult.
setArmSpeedAndAcceleration(short speed, short acceleration)	A kéz motorjainak sebességét és gyorsulását a paraméterben megadott értékekre állítja.	Hatására a paraméterben megadott értékekre (egész szám 0-255) állt be a kéz sebessége és a gyorsulása.
setHeadSpeedAndAcceleration(short speed, short acceleration)	A fej motorjainak sebességét és gyorsulását a paraméterben megadott értékekre állítja.	Hatására a paraméterben megadott értékekre (egész szám 0-255) állt be a fej sebessége és a gyorsulása.
setBlueBotsButtons(short button)	A takarítóegység vagy a labdázó egység bekapcsol (button: 1,2.)	Hatására vagy a takarítóegység (button:1) vagy a labdázó egység (button:2) bekapcsolt. A utasítás ismételt kiadása esetén az adott egység kikapcsolt.
setBlueBotsReference(short xVelocity, short yVelocity, short angularVelocity, short maxVelocity)	A paraméterben megadott x,y pozícióba a megadott sebességgel és szöggel elindul.	Hatására a robot az adott pozícióba indult a megadott sebességgel és a megadott szöggel.

4. Összegzés

Egy robotvezérlő rendszer tervezésére és megvalósítására alakult a Miskolci Egyetem mérnök informatikus és villamosmérnök hallgatóiból egy néhány fős csapat, hogy egy, a nap 24 órájában üzemelő robotvezérlő rendszert tervezzen és implementáljon. Ezen rendszer célja, hogy a BME által tervezett robotot „életre" keltse, azaz az ELTE által meghatározott feladatokat megvalósítson. Ezen robot funkcióját az ELTE Etológia tanszékén hivatott betölteni.

Mivel ezen rendszer tervezése egy ember számára túl nagy feladat, így azt több részre osztottuk. A felosztás következtében mindenki a saját tématerületével foglalkozik, amely következtében különálló modulok készülnek, melyek közös interfészeket használnak. Dolgozatom keretében a saját munkám került bemutatásra, amely során mélyebb bepillantást nyertem a robotika világába, megismerkedtem a BME által fejlesztett Ethon nevezetű robot felépítésével, illetve a különböző robotvezérlő keretrendszerek felépítésével. Mélyrehatóbban tanulmányoztam az OpenRTM rendszer működését, felépítését, Java programozási nyelven megvalósítottam egy Ethon-t vezérlő az OpenRTM rendszer részként működő illetve egy attól függetlenül használható szoftvert. Ezen szoftver lehetőséget biztosít a robot távvezérlésére, azaz segítségével számítógépes hálózaton keresztül vezérelhető a robot valamennyi funkciója. Az elkészült szoftver két komponensből, a „robotDriver" és a „rawController"-ből áll, a driver üzenetkeretek küldését valósítja meg a robot sorosportjaira, a felhasználói felület (rawController) pedig nyomógombok és vezérlőpanelek segítségével lehetővé teszi Ethon távirányítását. A dolgozat egy pillanatképet mutat a megvalósított szoftver állapotáról, a későbbiekben további fejlesztések kerülnek majd megvalósítására, pl.: a Kinect által szolgáltatott videófolyam továbbítása számítógépes hálózaton keresztül.

A feladatot belátható időn belül nem tudtam volna megvalósítani a projektben résztvevő társaim segítsége nélkül, egyértelműen több ember összehangolt munkáját igénylő feladat a teljes robotvezérlő rendszer megvalósítása. A projekt keretében rengeteg tapasztalatot és értékes útravalót kaptam életem további állomásaira.

Irodalomjegyzék

[1] http://project.mit.bme.hu/mi_almanach/books/aima/ch25s01

[2] Haláchy Nóra, Schmidt Dorottya - Esztétikus markerek robot lokalizációhoz (TDK dolgozat)

[3]http://www.mk.unideb.hu/userdir/vmt2/images/tantargyak/eloadasanyag/intelligens_robot_szerk.pdf

[4] http://en.wikipedia.org/wiki/Robot_Operating_System

[5] http://www.sze.hu/~herno/NGB_IN039_1/docs_ros/ros_02.pdf

[6] http://robotmodell3d.hu/TDK2.pdf

[7] http://openrtm.org/

[8] http://openrtm.org/openrtm/en/content/rtsystemeditor-100

[9] Noriaki Ando, Takashi Suehiro, and Tetsuo Kotoku - A Software Platform for Component Based RT-System Development: OpenRTM-Aist

[10]http://primeranks.net/yeti/University/III%20ev/II%20felev/Osztott%20Rendszerek/Eloadas/07-CORBA.pdf

Hivatkozások ellenőrzésének dátuma: 2013.11.03.

Köszönetnyílvánítás

Köszönöm *Dr. Kovács Szilveszter* konzulensemnek, hogy lehetőséget biztosított dolgozatom elkészítéséhez, köszönöm segítőkész támogatását, hasznos tanácsait. Köszönöm Krizsán Zoltán, Vincze Dávid és Piller Imre számtalan segítségét, illetve, hogy iránymutatásaikkal nagymértékben hozzájárultak szakmai fejlődésemhez.