MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

AUTOMATIZÁLÁSI ÉS
KOMMUNIKÁCIÓ-TECHNOLÓGIAI TANSZÉK

ARM mikrovezérlő alapú robotjármű fejlesztése

Projekt II.

Konzulens:
Dr. Vásárhelyi József
egyetemi docens

Készítette: Tompa Tamás
Neptun-kód: LJQHYK

Miskolc, 2013. december

Tartalomjegyzék

1. A feladat

A feladat egy robotjármű és a működtetéséhez szükséges számítógép oldali alkalmazás fejlesztése, amely segítségével az távvezérelhető. A robot lelke egy ARM mikorvezérlő alapú fejlesztő kártya, melyhez csatlakoznak a különböző perifériák, pl.: a vezetéknélküli kommunikáció megvalósításáért felelős bluetooth modul, a távolságszenzor, a motor/motorok működtetéséhez szükséges végfokok, a kamera, LED-ek, stb.

A feladat megvalósítását két részre, a robot tervezésére, megvalósítására, illetve a robot vezérlését megvalósító számítógépes szoftver implementálásra bontom.

2. A robot tervezése, megvalósítása

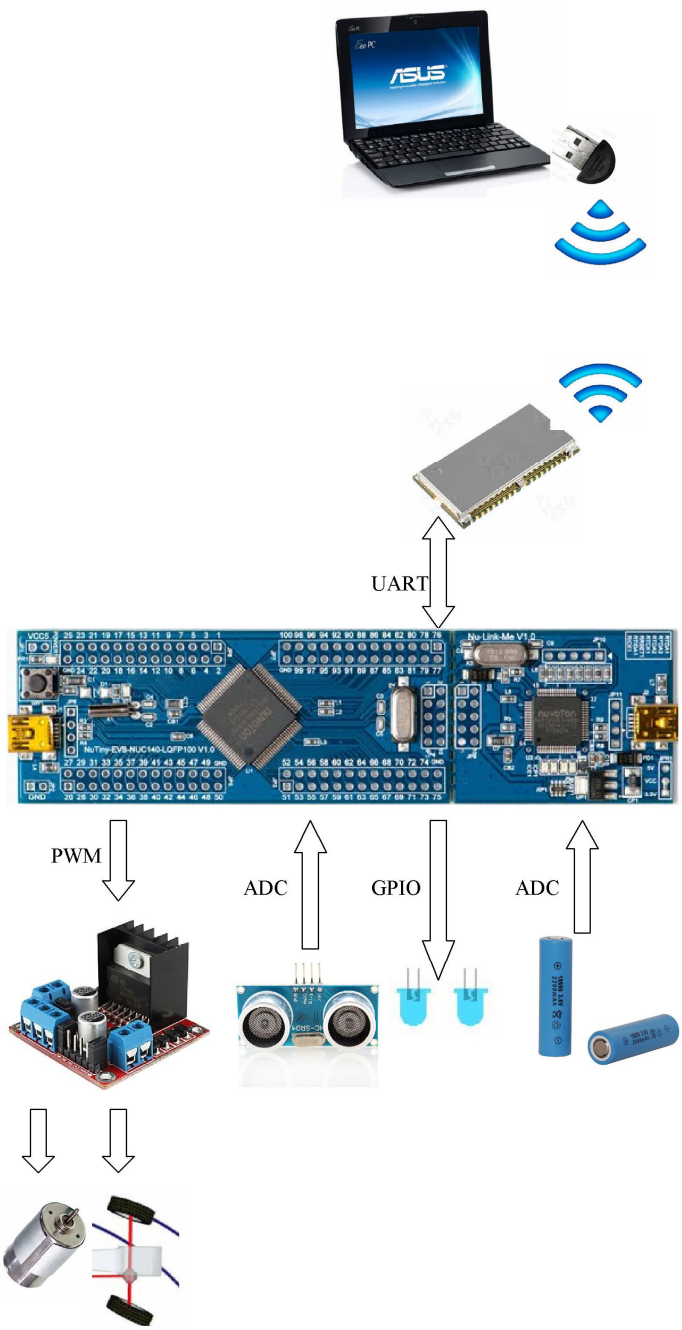
Ebben a fejezetben a robotjármű felépítését, moduljait illetve a robotjárművet vezérlő beágyazott rendszer tervezésének, megvalósításának menetét ismertetem.

2.1 A robot váza

A robotjármű alapja egy távirányítós Nikko játékautó, melynek a vezérlő része eltávolításra került, csak az autó váza és a meghajtáshoz szükséges motor/motorok maradtak meg. Ezen autó egy darab kefés DC motorral rendelkezik, mely a hátsó kerekek hajtásáért felelős, az első kerekek kormányzásához pedig elektromágneses mechanikát alkalmaz.

2.2 A teljes rendszer blokkvázlata

A következő ábra a rendszer, többek között a robot felépítésének blokkvázlatát szemlélteti:



1. ábra A rendszer blokkvázlata

A számítógép egy adott programozási nyelven implementált alkalmazás segítségével vezérli a robotot. A vezérlőszoftver egy USB-s bluetooth modul segítségével továbbítja a robot vezérléséhez szükséges utasításkereteket.
A robot lelke, azaz az ARM mikorvezérlő egy bluetooth modul segítségével fogadja a számítógép által elküldött utasításkereteket, amely következtében majd vezérli a járművet, azaz kezeli a működtetéshez szükséges megfelelő perifériát.

A rendszer elemei:

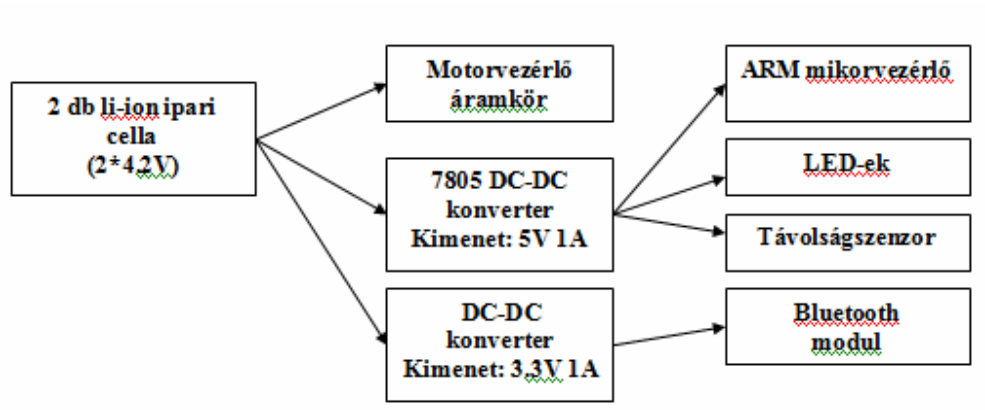
- Számítógép
    - USB-s bluetooth modul
    - Vezérlőszoftver (Java alkalmazás)
- Robotjármű
    - Nuvoton NUC140 ARM alapú fejlesztőkészlet
    - BTM-222 bluetooth modul
    - Motorvezérlő elektronika (h-híd)
    - Távolságszenzor
    - LED-ek (a jármű elején)
    - Tápellátást biztosító akkumulátorok
    - Különböző tápfeszültségeket előállító áramkörök (DC-DC konverterek)

2.3 A Tápellátás

A robot működéséhez szükséges tápfeszültséget 2 darab sorosan kapcsolt li-ion ipari cella biztosítja, a cellák feltöltött állapotban egyenként 4.2V-ot biztosítanak és 1800mAh kapacitásúak. A két darab ipari cella 8V-9V közötti feszültséget biztosít a robot számára.

A robot egyes perifériái pl.: a mikorvezérlő, a motorvezérlő elektronika, a bluetooth modul, stb., megfelelő működéséhez más-más tápfeszültségek szükségesek, melyeket DC-DC konverter állít elő. A mikrovezérlő üzemszerű működéséhez szükséges tápfeszültség 2.5V-4.5V között, a BTM-222 bluetooth modul üzemszerű működéséhez szükséges tápfeszültség pedig a 3.0V-3.6V-os tartományban van. A motorvezérlő elektronika megfelelő működéséhez 5V-12V tápfeszültség szükséges, így az közvetlenül csatlakozik az akkumulátorokra. A robot további perifériái számára szükséges 5 illetve 3.3V-os feszültséget DC-DC konverter IC-k állítják elő.

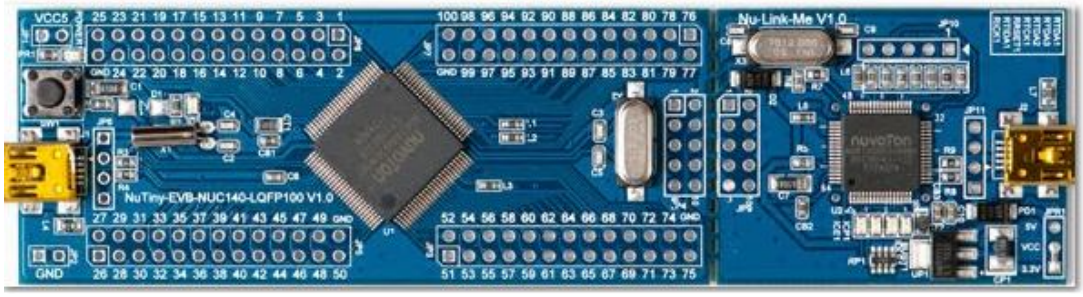
A tápellátás blokkvázlata az alábbi 2. ábrán látható:



2. ábra A tápellátás blokkvázlata

2.4 A mikorvezérlő

A robot lelke egy ARM mikorvezérlőt tartalmazó fejlesztő készlet (Nuvoton NUC140 V3ECN), amely a mikorvezérlő programozásához szükséges „égető” egységet is tartalmazza. Ezen fejlesztőkészlet előnye, hogy minden, a mikrovezérlő működtetéséhez és programozásához szükséges áramkört egy nyomtatott áramköri lapkán tartalmazza. A fejlesztőkészlet az alábbi 3. ábrán látható:



3. ábra A fejlesztőkészlet

A felhasznált perifériákat a következő táblázat tartalmazza:

Pin	Funkció
10	2 darab LED vezérlése
38	UART RX
39	UART TX
74	ADC, távolságszenzor
77	ADC, akkumulátor állapot
63	PWM, első kerék
65	PWM, hátsó motor
71	Hátsó motor forgásirány1
72	Hátsó motor forgásirány2
73	Első kerék forgásirány1
61	Első kerék forgásirány2

4. ábra A felhasznált perifériákat tartalmazó táblázat

2.5 Kommunikációs modul illesztése

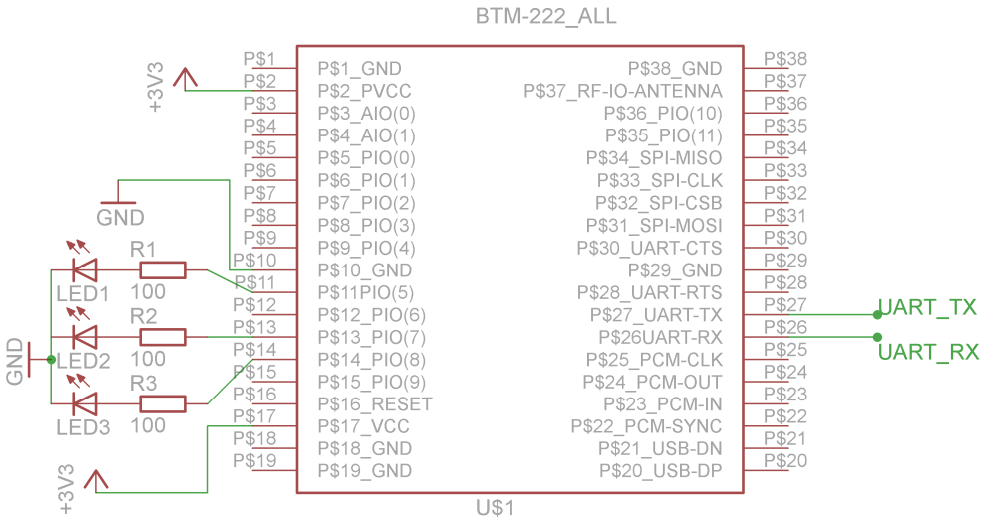
A robotautó és a számítógépszoftver közötti vezeték nélküli kommunikációs kapcsolat megvalósításáért egy usb-s bluetooth stick és a BMT-222 típusú bluetooth modul felelős, amely az 5. ábrán látható:

5. ábra A BTM-222 típusú bluetooth modul

Ezen modul class 1-es típusú, azaz hatótávolsága kb. 100 méter. Paramétereit konfigurálni AT utasítások segítségével lehetséges soros vonalon keresztül egy terminál program segítségével. Konfigurálásra jelen esetben volt szükség, mert az SPP-t (Serial Port Profile) miden további beállítás nélkül tudja kezelni.

A modul 3.3V-os megtáplálást igényel, egyéb más feszültségforrás esetén célszerű egy stabilizátor IC-t alkalmazni, mely képes adott feszültségtartományból 3.3V-os tápfeszültséget előállítani.

Ezen eszköz beépített antennával nem rendelkezik, így azt külön szükséges hozzá illeszteni. Antennának egy 3 cm hosszúságú vezetékdarab alkalmazása ajánlott, de akár erre a célra kialakított wifi antennát is alkalmazhatunk, ebben az esetben viszont nagy hangsúlyt szükséges fektetni az antennavezeték árnyékolására. A következő ábra a modul bekötési rajzát szemlélteti:

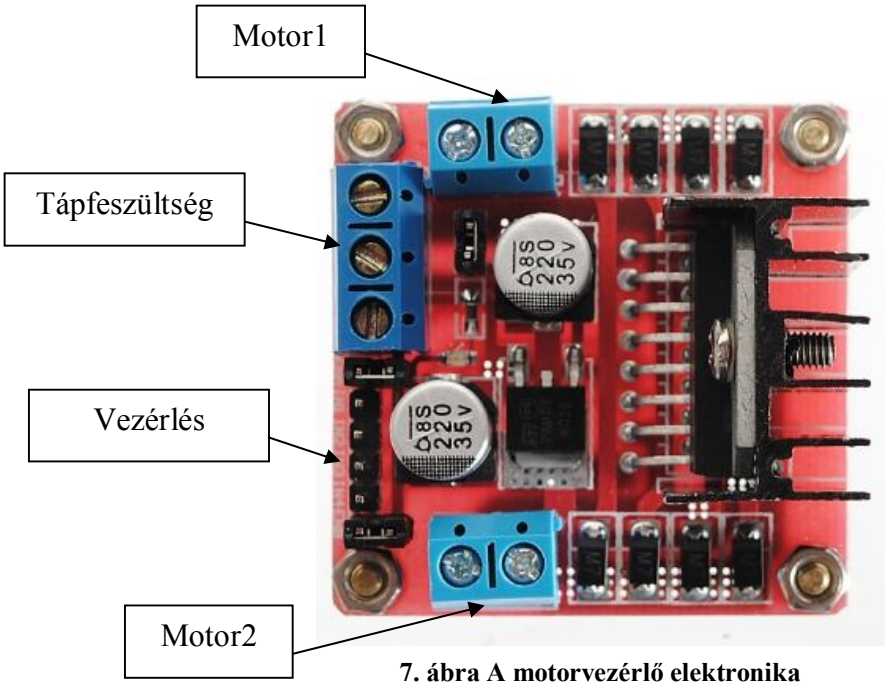


6. ábra A bluetooth modul illesztési rajza

Az ábrán található 3 darab led (LED1, LED2, LED3) funkciója a tápfeszültség, a kommunikációs kapcsolat és az adatküldés/adatfogadás állapotának jelzése.

2.6 Motorvezérlő áramkör illesztése

Ezen áramkör szolgál segítségül a robot hátsó motorjának vezérlésében illetve az első kerekek kormányzásának megvalósításában. A robot egy kefés dc motort tartalmaz, amely a hátsó kerekek meghajtásáért felelős, ezen motor működéséhez kb. 2-3A áramot igényel, így az közvetlenül meg csatlakoztatható a mikorvezérlőhöz. Az illesztést illetve a motor sebesség és a forgásirány vezérlését ezen motorvezérlő áramkör valósítja meg.
Az általam választott végfok, az Arduino-hoz fejlesztett, külön modulként vásárolható L298N kettős h-híd, amely fényképe a 7. ábrán látható.



7. ábra A motorvezérlő elektronika

Ezen elektronika 2 db kefés dc motor vagy 1 db lépetető motor meghajtására képes. Működtetéséhez 5V-35V közötti tápfeszültség szükséges, motoronként 2A áram leadására képes, teljesítménye 25W. Külön pozitívum, hogy rendelkezik 5V-os kimenettel, így szükség esetén a robot 5V-os tápfeszültséget igénylő perifériái számára is megfelelő áramforrást biztosít.

Vezérlése 6 db „tüske” segítségével lehetséges, melyek elnevezése fentről-lefelé:

- ENA: „A” motor engedélyezése
- IN1: „A” motor forgásirány/fék
- IN2: „A” motor forgásirány/fék
- IN3: „B” motor forgásirány/fék
- IN4: „B” motor forgásirány/fék
- ENB: „B” motor engedélyezése

Alapvetően kétféle vezérlésre biztosít lehetőséget:

- Motorok kikapcs/bekapcs
- PWM

Ha az ENA vagy az ENB tüske össze van kötve a felette elhelyezkedő tüskével egy jumper segítségével, akkor a motorok „bekapcsolva/kikapcsolva” elven működtethetőek, azaz nem adható meg számukra sebesség, hanem a híd teljes fordulatszámmal fogja őket meghajtani. Ha a jumpereket eltávolítjuk az ENA vagy ENB lábakról, akkor valósítható meg a híd vezérlése pwm jel segítségével, ebben az esetben a pwm jelet ezen tüskékre szükséges kapcsolni. Jelen esetben a pwm-es vezérlési módot használtam.

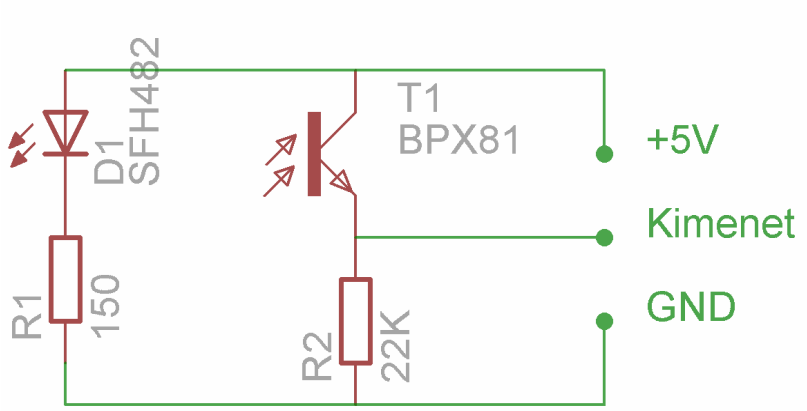
Az egyes motorok forgásiránya az IN1, IN2, IN3, IN4 lábak segítségével vezérelhető, a következő módon:

Beállítás	Funkció
IN1 5V; IN2 GND	„A” motor előre
IN1 GND; IN2 5V	„A” motor hátra
IN3 5V; IN4 GND	„B” motor előre
IN3 GND; IN4 5V	„B” motor hátra
IN1 GND; IN2 GND	„A” motor kikapcsolva
IN3 GND; IN4 GND	„B” motor kikapcsolva
IN1 5V; IN2 5V	„A” motor fékez
IN3 5V; IN4 5V	„B” motor fékez

8. ábra A motorvezérlő áramkör vezérlése

2.7 Távolságszenzor illesztése

Egy kezdetleges távolságszenzor került felhelyezése a robot elejére, melynek célja a robot előtt elhelyezkedő esetleges akadályok érzékelése. A szenzor kapcsolási rajza a 8. ábrán látható.



9. ábra A távolságszenzor kapcsolási rajza

A kapcsolás működése: a fototranzisztor bázisárama a rá eső infravörös fény intenzitásával arányos, amely meghatározza az emitter-kollektor áramot. Ez a változó erősségű áram kerül átalakításra feszültségváltozássá a tranzisztorral sorba kötött 22kOhm-s ellenálláson keresztül. A szenzorra eső infravörös fény intenzitásától függően a szenzor kimentén egy 0 és 5V közötti analóg feszültségjelet szolgáltat. A detektorra visszaverődött infravörös fény mennyisége függ az akadály távolságától illetve annak anyagától. A szenzor a mikorvezérlő 74-es lábára, azaz a 3-as ADC csatornára került illesztése.

Mivel ezen szenzor működését nagymértékben befolyásolják a külső fényviszonyok (napfény, neon lámpák fénye, stb.) illetve érzékelési tartománya kb. 1-10 cm között van, ezért a későbbiekben lecserélésre fog kerülni, egy ultrahangos távolságszenzor fogja felváltani, amely fényképe a 9. ábrán látható.

10. ábra Az ultrahangos távolságszenzor

Ezen szenzor vezérlése annyiban különbözik a 8. ábrán látható szenzorétól, hogy működtetéséhez a „Trig” nevezetű lábára amely 10µs-onként digitális jel küldése szükséges, melynek hatására az „Echo” lábán küldi az általa mért távolság értékét. Ezen szenzor mérési tartománya 2-300 cm.

2.8 LED-ek illesztése

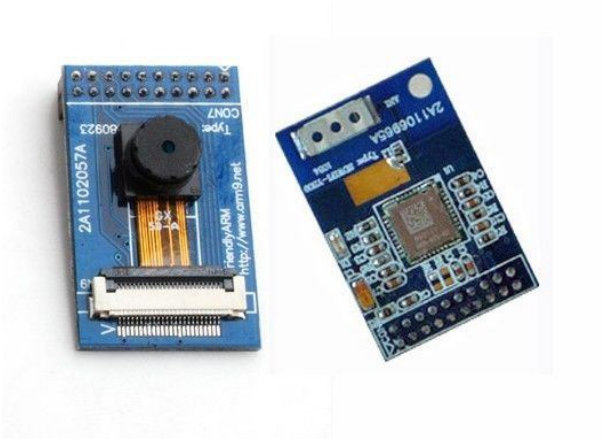
Két darab fehér színű led kapott helyet a robot elején, melyek az adott vezérlőkeret segítségével be/ki kapcsolgathatóak. A két led egymással sorban kapcsolva (előtét ellenállások alkalmazásával) csatlakozik a mikorvezérlő 10-es számú lábára.

2.9 Kamera modul illesztése

A későbbiekben egy kamera modul illesztését tervezem megvalósítani, melynek elsődleges feladata a kamerakép továbbítsa a robot számítógépes vezérlőszoftvere felé, amely majd megjeleníti azt. Ezen feladat megvalósításának szűk keresztmetszete a vezetélnéklüi kommunikáció megvalósításához alkalmazott bluetooth modul (bluetooth kapcsolat) a keskeny sávszélessége miatt.
Két lehetőség kínálkozik, az első, hogy a jelenlegi bluetooth kapcsolaton keresztül továbbítja a kamera a képet, de ekkor a kép felbontását kicsire (pl.: 320*240) állítom. Valószínűleg a legkisebb felbontás esetén is másodperces nagyságrendű idő lesz, míg a mikorvezérlő feldolgozza a kamera modul által küldött adatkeretek és azt továbbítja a bluetooth kapcsolaton keresztül. A másik lehetőség a bluetooth modul lecserélése wifi modulra, vagy egy olyan kamera alkalmazása, amely rendelkezik beépített vezetéknélküli kommunikációs kapcsolatot megvalósító modullal (akár bluetooth, akár wifi).

Ezen érvek alapján két kamera jöhet szóba, melyek fényképe az alábbi ábrákon látható:

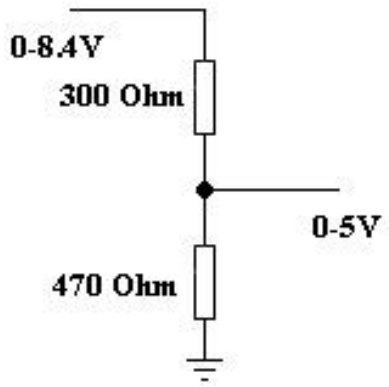
11. ábra RS232 kommunikációra képes kamera



12. ábra Beépített wifi modullal rendelkező kamera

2.10 Akkumulátorok feszültségének mérése

Célja, hogy a robot képes legyen az akkumulátorok feszültségét és ez által azok töltöttségi állapotát mérni. Mivel a 2 darab akkumulátor feltöltött állapotban 8.4V feszültséget biztosít, a mikrovezérlő az ADC csatornáin közvetlenül maximum 5V feszültséget tud mérni, így szükség van egy feszültségosztó alkalmazására, amely a 8.4V-ból 5V-os feszültséget állít elő. A feszültségosztó kimenete így már közvetlenül csatlakoztatható a mikrovezérlő ADC csatornájára, jelen esetben a 77-es lábra, azaz a 6-os ADC csatornára. Az alkalmazott feszültségosztó kapcsolást a következő ábra szemlélteti:



13. ábra Az alkalmazott feszültségosztó kapcsolási rajza

2.11 A vezérlés utasításkeretei

A vezetéknélküli kommunikáció során a robot adott funkciója (pl.: led-ek bekapcsolása, előre haladás, állj, sebesség változtatás stb.) adott utasításkeretek elküldésével működtethető. A keretek „keretkezdet”+”adat”+”keretvég” formátumúak, melyek pontos felépítését a következő táblázat tartalmazza.

Funkció	1.byte	2.byte	3.byte	4.byte	5.byte	6.byte
Lampa	0x01	"l"	127			
Előre	0x02	pwm (0-100)	127			
Hátra	0x03	pwm (0-100)	127			
Balra	0x04	pwm (0-100)	127			
Jobbra	0x05	pwm (0-100)	127			
Állj	0x06	0	127			
Távolságszenzor lekérdezés	0x07	0x07	127			
Távolságszenzor válasz	0x08	1.byte	2.byte	3.byte	4.byte	127
Akku szenzor lekérdezés	0x09	0x09	127			
Akku szenzor valasz	0xAA	1.byte	2.byte	3.byte	4.byte	127

14. ábra A vezérlés utasításkereteinek felépítése

Például a távolságszenzor által mért érték lekérdezéséhez a „0x07 0x07 127” keret küldése szükséges, melyre a robot a „0x08 1.adat byte 2.adat byte 3.adat byte 4.adat byte 127” kerettel válaszol, ahol az 1., 2., 3. és 4. byte az érzékelő által mért adat 4 részre bontva.

A távolságszenzor lekérdező és az akkumulátorok állapotát lekérdező keret is ADC értékeket továbbít, melyek további feldolgozást igényelnek. Ezen feldolgozást a robot számítógépes vezérlőszoftvere valósítja meg, amely során ezen értékeket feszültség értékekké konvertálja.

2.12 A mikorvezérlő szoftvere

A mikorvezérlő szoftvere a CooCox nevezetű fejlesztőkörnyezet segítségével készült C programozási nyelven.
A forráskód logikai több részre osztva, különböző forrásállományokban található, melyek a következők:

Az egyes állományok és funkcióik:

Állomány	Funkció
main.c	Főprogram.
uartComm.c	Bluetooth modulon keresztüli kommunikációt, illetve az adatkeretek küldését/fogadását megvalósító függvényeket tartalmazza.
motorControl.c	A hátsó motor és az első kerekek működtetéséhez szükséges pwm jeleket előállító függvények gyűjteménye.
timer.c	Időzítést megvalósító függvényeket tartalmaz.
lampaControl.c	A robot elején elhelyezkedő led-ek működtetését megvalósító függvények tartalmazza .
Init.c	Az egyes perifériák (pwm, adc) adott paraméterekkel való inicializálást megvalósító állomány.
ADC.c	A távolságszenzor jelét feldolgozó függvények gyűjteménye.

Az uartComm.c állomány függvényei:

Függvény	Funkció
void UART0_INT_HANDLE(uint32_t u32IntStatus)	A küldött utasításkeretek összerakását, a fogadott keretek szétdarabolását illetve ezen keretek alapján adott funkciók végrehajtását (pl.: led-ek bekapcsolása) valósítja meg.
void UART_Init2()	Az UART adott paraméterekkel való inicializálást valósítja meg.

A motorControl.c állomány függvényei:

Függvény	Funkció
void setKitoltesiTenyezo(uint8_t kitoltesArg)	A pwm jelek kitöltési tényezőjét beállító függvény.
void forgasIranyHatsoKerek(int forgasIranyArg)	A hátsó motor forgásirányát beállító függvény.
void forgasIranyElsoKerek(int forgasIranyArg)	Az első kerék irányát (balra/jobbra/egyenesen) beállító függvény.
void PWM_Vezerles(int frekvenciaHatso, int kitoltesHatso, int frekvenciaElso, int kitoltesElso)	A hátsó motor fordulatszámát és az első kerék kormányzását megvalósító pwm jeleket (2 db) előállító függvény.
void elore()	A robot előre haladását megvalósító függvény.
void hatra()	A robot tolatását megvalósító függvény.
void jobbra()	A robot jobbra kanyarodását megvalósító függvény.

void balra()	A robot balra kanyarodását megvalósító függvény.
void allj()	Az „állj” funkciót megvalósító függvény, hatására a robot egyhelyben áll.

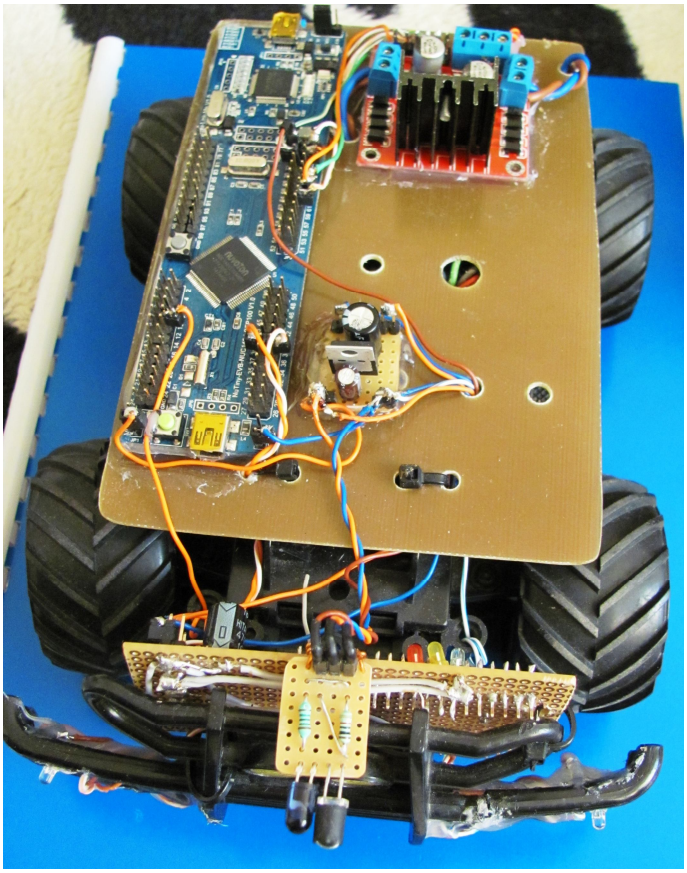
A lampaControl.c állomány függvényei:

Függvény	**Funkció**
void lampa()	A robot elején elhelyezkedő led-ek be- illetve kikapcsolását vezérli.

Az ADC.c állomány függvényei:

Függvény	**Funkció**
void tavolsagSzenzor(void)	Feldolgozza, majd a bluetooth modul felé továbbítja a távolságszenzor értéket.
void batterySzenzor(void)	Feldolgozza, majd a bluetooth modul felé továbbítja az akkumulátorok töltöttségi állapotát.

2.13 Az elkészült robot fényképe



15. ábra Az elkészült robot fényképe

3. A számítógépes vezérlőszoftver

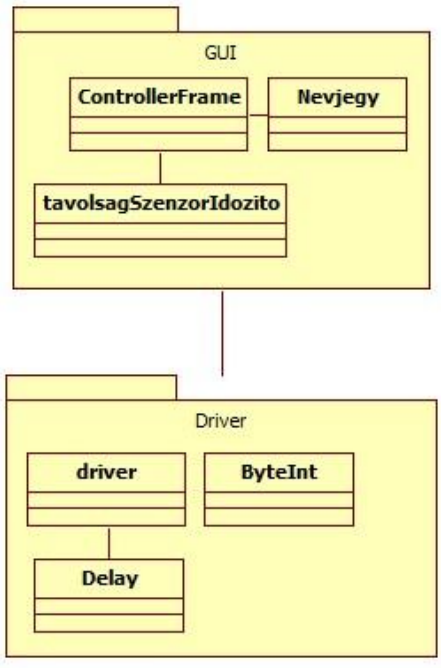
Ebben a fejezetben a robot irányítására lehetőséget biztosító illetve a robot vezérlését megvalósító Java programozási nyelven implementált alkalmazás tervezését, megvalósítását ismertetem.

3.1 A szoftver célja

Az alkalmazás célja, hogy egy grafikus felhasználói felület segítségével lehetőséget biztosítson a robot távvezérlésére, azaz valamennyi funkciójának működtetésére. Ezen szoftver valósítja meg a robot működtetéséhez szükséges utasítás keretek összeállítását, illetve ezen keretek továbbítását egy usb-s bluetooth modul segítségével. Az usb-s bluetooth stick-et soros port üzemmódban használom, így a szoftver a bluetooth modulhoz tartozó sorors portra (1 darab soros port) írja az adott utasításkeretet.

3.2 Az osztályiagram

A következő diagram a szoftver osztályvázát szemlélteti:

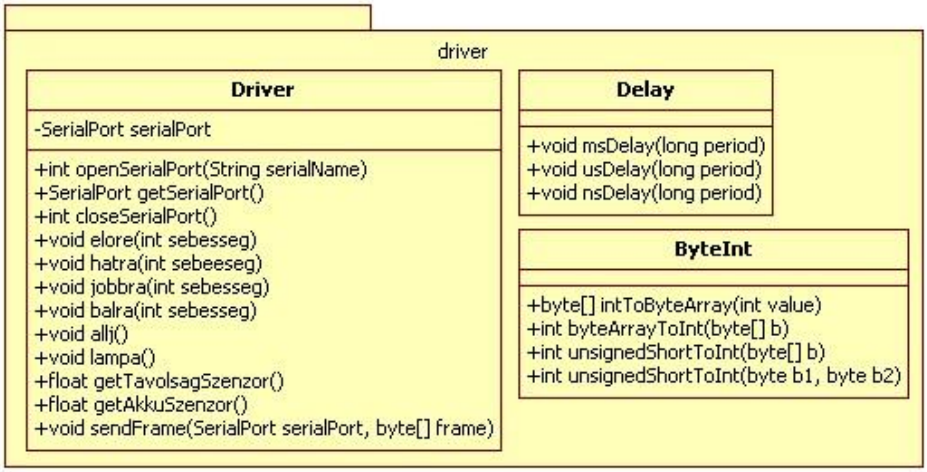


16. ábra A szoftver osztályváza

Az alkalmazás logikai két részre, azaz két csomagra, a „GUI” és a „Driver” csomagra bonható. A Driver csomag feladata a hardver alacsonyszintű vezérlése, azaz a robot működtetéséhez szükséges utasításkeretek összeállítása, sorosportra írása. A GUI csomag feladata a grafikus felhasználói felület megjelenítése. A két modul közötti kapcsolat: a felhasználói felületen kiváltott események alapján hívódik meg a driver osztály megfelelő metódusa (pl.: GUI esemény: előre 50 sebességgel, melynek hatására a Driver osztály elore(50) metódusa hívódik).

3.3 A Driver csomag

A Driver csomag feladata, hogy a robot vezérléséhez szükséges kereteket összeállítsa és adott sorosportra - egyetlen soros port - írja. Ezen feladatok megvalósításához szükséges osztályokat tartalmazza, melyet a következő 17. ábra szemléltet:



17. ábra A Driver csomag felépítése

A csomag a következő osztályokból áll:

- Driver: a vezérléshez szükséges utasításkereteket állítja össze
- Delay: késleltetést megvalósító osztály
- ByteInt: byte és int típusok közötti konvertálást megvalósító osztály

A Driver osztály metódusai és azok funkciói:

Metódus	Funkció
`int openSerialPort(String serialName)`	A paraméterben megadott nevű sorosportot nyitja, visszatérési értéke, hogy a port megnyitása sikeres volt-e vagy sem. (1: sikeres, 0: sikertelen).
`SerialPort getSerialPort()`	Az osztály által használt sorosportot adja vissza.
`int closeSerialPort()`	Sorosport bezárása, visszatérési értéke attól függ, hogy sikerült-e a portot beázárni a vagy sem (1: sikeres, 0: sikertelen).
`void elore(int sebesseg)`	A paraméterben megadott sebességgel előre mozdul a robot (sebesség: 0-100 egész)
`void hatra(int sebesseg)`	A paraméterben megadott sebességgel hátrafelé mozdul a robot (sebesség: 0-100 egész).
`void jobbra(int sebesseg)`	A paraméterben megadott sebességgel jobbra mozdul a robot (sebesség: 0-100 egész).
`void balra(int sebesseg)`	A paraméterben megadott sebességgel balra mozdul a robot (sebesség: 0-100 egész).
`void allj()`	Hatására megáll a robot.
`void lampa()`	A roboton lévő led-ek ki/bekapcsolása.
`float getTavolsagSzenzor()`	A robot távolság érzékelőjének ADC értékét adja vissza egy float típusú változóban.
`float getAkkuSzenzor()`	A robot akkumulátorainak ADC értékét adja vissza egy float típusú változóban.
`void sendFrame(SerialPort serialPort, byte[] frame)`	A paraméterben megadott nevű sorosportra (serialPort), a paraméterben megadott keretet (frame) küldi.

A Delay osztály metódusai és azok funkciói:

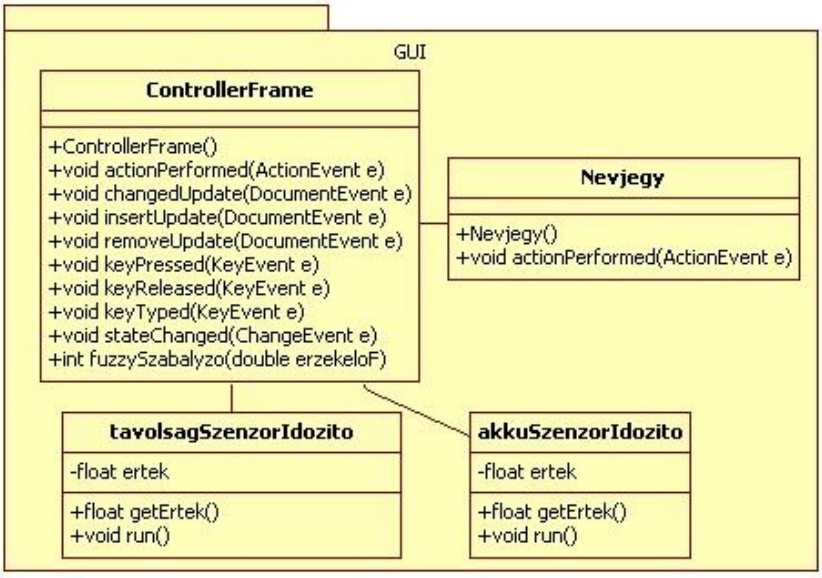
Metódus	Funkció
`void msDelay(long period)`	A paraméterében megadott (period) ms időt késleltet.
`void usDelay(long period)`	A paraméterében megadott (period) µs időt késleltet.
`void nsDelay(long period)`	A paraméterében megadott (period) ns időt késleltet.

A ByteInt osztály metódusai és azok funkciói:

Metódus	Funkció
`byte[] intToByteArray(int value)`	A paraméterében megadott (value) int értéket byte tömbbé konvertálja.
`int byteArrayToInt(byte[] b)`	A paraméterében megadott (b) byte tömböt (pontosabban a tömb elemeit) egy int számmá konvertálja.

3.4 A GUI csomag

A GUI csomag feladata, hogy a robot vezérléséhez egy kényelmesen használható grafikus felhasználói felületet biztosítson. Ezen feladat megvalósításához szükséges osztályokat tartalmazza, melyet a következő 18. ábra szemléltet:



18. ábra A GUI csomag felépítése

A csomag a következő osztályokat tartalmazza:

- ControllerFrame: a grafikus felületet megvalósító osztály
- tavolsagSzenzorIdozito: a robot távolság szenzorának értékét adott időközönként lekérdező osztály. A megadott időközönként távolságszenzor lekérdező keretet küld a robot számra, melyre az visszaadja a távolság érzékelő ADC értékét egy float típusú változóban. Ezen ADC értéket feszültség értékké konvertálja.
- akkuSzenzorIdozito: a robot akkumulátor feszültségmérő szenzorának értékét adott időközönként lekérdező osztály. A megadott időközönként akku feszültség lekérdező keretet küld a robot számra, melyre az visszaadja az akkumulátorok ADC értékét egy float típusú változóban. Ezen ADC értéket feszültség értékké konvertálja.

- Nevjegy: az alkalmazás készítőjéről információkat megjelenítő ablakot megvalósító osztály

A ControllerFrame osztály metódusai és azok funkciói:

Metódus	Funkció
`ControllerFrame()`	Konstruktor.
`void actionPerformed(ActionEvent e)`	A felhasználói felületen lévő nyomógombok eseményeinek kezelését megvalósító metódus.
`void changedUpdate(DocumentEvent e)`	A felhasználói felületen lévő szövegbeviteli mező (sorosport megadása) eseményének kezelését megvalósító metódus.
`void removeUpdate(DocumentEvent e)`	A felhasználói felületen lévő szövegbeviteli mező (sorosport megadása) eseményének kezelését megvalósító metódus.
`void keyPressed(KeyEvent e)`	A robot irányítására lehetőséget biztosító billentyűk billentyű lenyomás eseményét kezelő metódus.
`void keyReleased(KeyEvent e)`	A robot irányítására lehetőséget biztosító billentyűk billentyű elengedés eseményét kezelő metódus. Minden billentyű felengedésre „állj” keret küld a robotnak.
`int fuzzySzabalyzo(double erzekeloF)`	A robot fuzzy sebességszabályzását megvalósító metódus (későbbi fejezetben bővebben).

A tavolsagSzenzorIdozito osztály metódusai és azok funkciói:

Metódus	Funkció
`float getErtek()`	A távolságszenzor értékét visszaadó metódus.
`void run()`	A TimerTask osztály metódusa, a törzsében definiált funkciókat (távolságszenzor lekérdezése) hajtja végre a megadott időközönként.

Az akkuSzenzorIdozito osztály metódusai és azok funkciói:

Metódus	Funkció
`float getErtek()`	Az akku feszültség szenzor értékét visszaadó metódus.
`void run()`	A TimerTask osztály metódusa, a törzsében definiált funkciókat (akku állapot lekérdezése) hajtja végre a megadott időközönként.

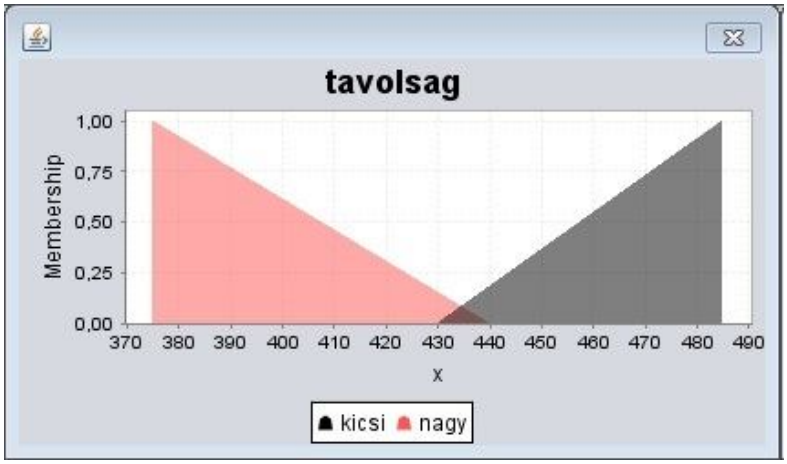
3.5 A fuzzy szabályozó

A szoftverbe beépítésre került egy fuzzy szabályozó, melyet a „jFuzzyLogic_v3.0” Java-s csomag felhasználásával valósítottam meg. Ezen vezérlési mód kiválasztása a „Fuzzy” nevezetű nyomógomb segítségével lehetséges.
A szabályozó célja, hogy a távolságszenzor által küldött értékektől függően, azaz, hogy mekkora a robot előtt lévő szabad terület, szabályozza a hátsó motor fordulat számát, azaz a robot sebességét. Ez alapján, ha a robot előtt nincs akadály, akkor az maximális sebességgel halad, ha viszont akadályhoz közeledik, akkor már az akadály előtt, függve annak a robottól való távolságától csökkenti sebességét, tehát ha már az akadály elég közel kerül hozzá, akkor a robot megáll.

A szabályozót FCL nyelven (fuzzy control language) lehetséges definiálni, mely egy .fcl kiterjesztésű fájlban valósul meg. A Java-s fuzzy csomag ezen fájl beolvasásával valósítja meg a szabályozást. A fájl tartalma a következő:

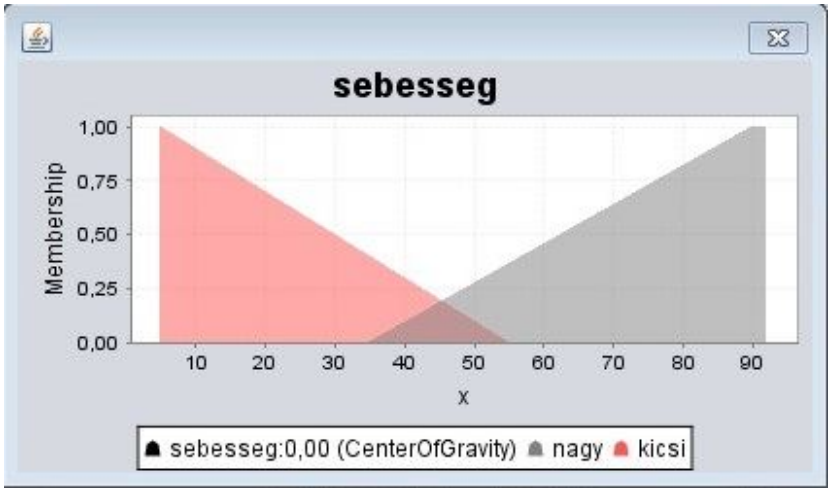
```
FUNCTION_BLOCK robotControl

VAR_INPUT
   tavolsag : REAL;
END_VAR

VAR_OUTPUT
   sebesseg : REAL;
END_VAR



FUZZIFY tavolsag
   TERM nagy := (375, 1) (440, 0) ;
   TERM kicsi := (430, 0) (485, 1);
END_FUZZIFY



DEFUZZIFY sebesseg
   TERM kicsi := (5,1) (55,0);
   TERM nagy := (35,0) (90,1);
   // Use 'Center Of Gravity' defuzzification method
   METHOD : COG;
   // Default value is 0 (if no rule activates defuzzifier)
   DEFAULT := 0;
END_DEFUZZIFY


RULEBLOCK No1
   // Use 'min' for 'and' (also implicit use 'max'
   // for 'or' to fulfill DeMorgan's Law)
   AND : MIN;
   // Use 'min' activation method
   ACT : MIN;
   // Use 'max' accumulation method
   ACCU : MAX;

   RULE 1 : IF tavolsag IS kicsi THEN sebesseg IS kicsi;
   RULE 2 : IF tavolsag IS nagy THEN sebesseg IS nagy;


END_RULEBLOCK

END_FUNCTION_BLOCK
```

A szabályozó két változóval - egy bementi, amely a távolságszenzor értéke és egy kimeneti, amely a hátsó motor sebessége - rendelkezik. Mindkét változó tagságfüggvénye trianguláris, azaz háromszög alakú, melyet a következő 19. és 20. ábrák szemléltetnek:



19. ábra A "tavolsag" tagságfüggvényei

A két háromszög alakú függvény a távolságszenzor által mért értékek „kicsi” és „nagy” tartományba való leképezését valósítja meg, azaz, hogy a szenzor előtt lévő akadály távolsága mekkora mértékű. Ha szenzor által küldött értékek 375-440 közötti tartományban vannak (piros színű függvény), akkor a szenzor előtt lévő akadály távol van, azaz az előtte lévő távolság nagy. A szürke színű háromszög alakú függvény a „kicsi” távolságot definiálja, azaz ha a szenzor által küldött értékek 430-485 közötti tartományban helyezkednek el, akkor a szenzor előtt lévő szabad terület kicsi, tehát előtte akadály található.



20. ábra A "sebesseg" tagságfüggvényei

A 20. ábrán a sebesség tagságfüggvényei láthatóak, amelyek a „kicsi” és a „nagy” sebességeket definiálják. A piros színű függvény a „kicsi” sebességet határozza meg, azaz

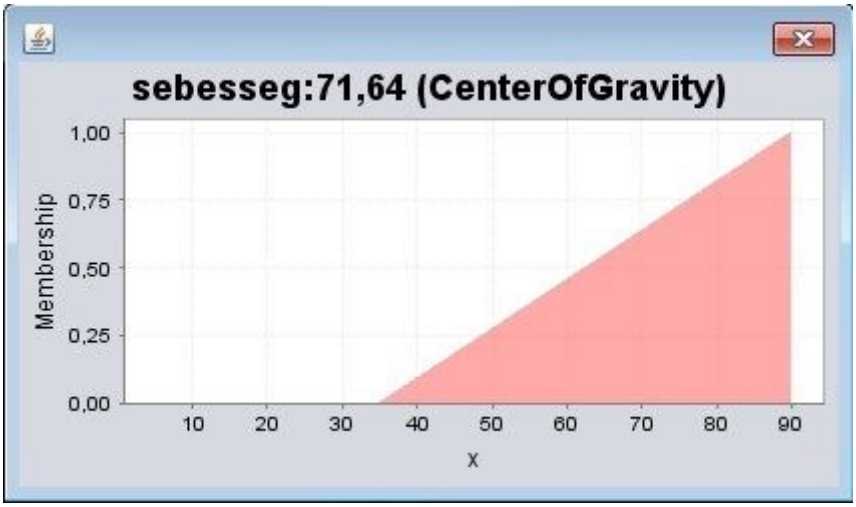
a robot sebessége ebben az esetben kicsi, az elküldött sebességértékek az 5-55 közötti tartományban helyezkednek el. A szürke színű függvény a „nagy” sebességet határozza meg, ekkor a robot sebessége a 35-95 közötti tartományban van, azaz gyorsan halad.

A szabálybázis két szabályból áll, melyek a következők:

IF tavolsag IS kicsi THEN sebesseg IS kicsi

IF tavolsag IS nagy THEN sebesseg IS nagy

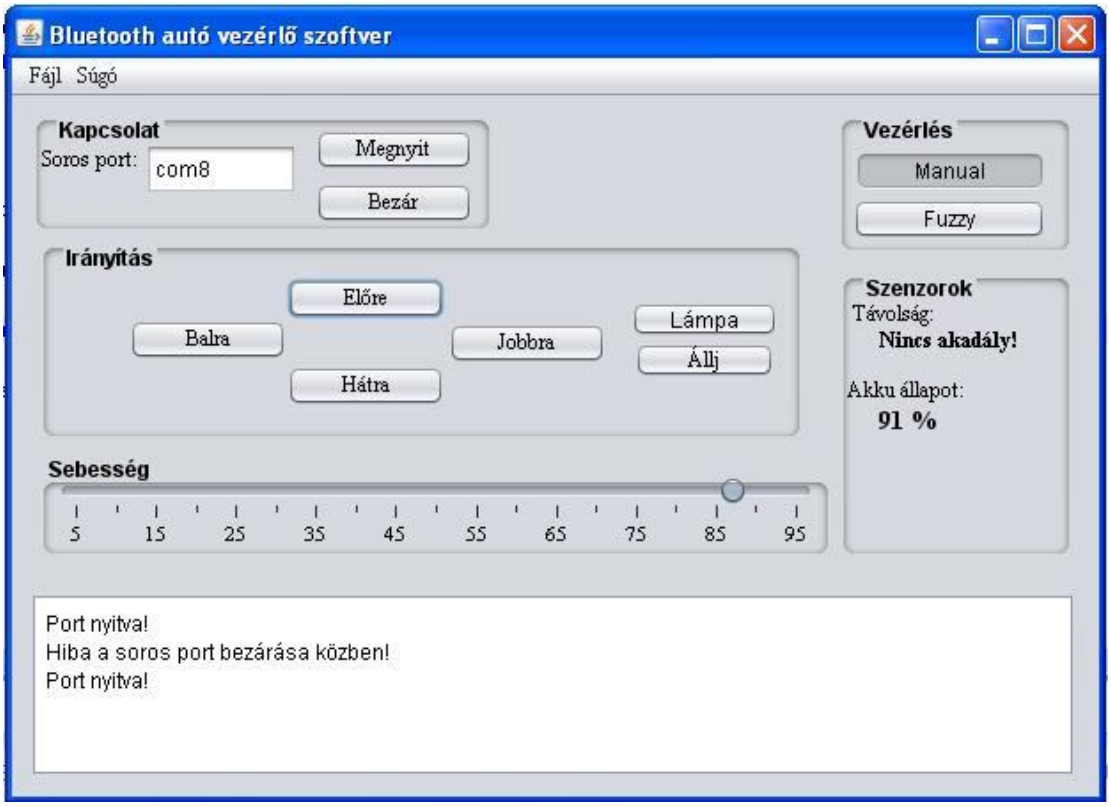
A 21. ábrán a kimeneti változó aktuális értéke (a robot hátsó motorjának sebessége: 71.64; COG módszerrel meghatározva) látható, melyet a távolságszenzor értékei alapján határoz meg a szabályozó.



21. ábra A kimenet aktuális értéke

3.6 A felhasználói felület

A következő ábra a szoftver grafikus felhasználói felületét szemlélteti:



22. ábra Az alkalmazás felülete

A soros kommunikációs kapcsolatot a robottal a „Kapcsolat” nevezetű kis panelablakban lehetséges létrehozni. Ehhez a „Soros port” szövegbeviteli mezőben szükséges megadni azt a soros portot, melyhez az usb-s bluetooth stick kapcsolódott. Ezt követően a „Megnyit” gombra való kattintás után jön létre a kommunikációs kapcsolat a szoftver és a robot között.

A vezérlési mód kiválasztása a „Vezérlés” panelablakában lehetséges, amely lehet manuális illetve fuzzy. Manuális vezérélési módesetén lehetséges a robot nyomgombokkal illetve billentyűgombokkal (w,a,s,d,l, nyilak) történő irányítása. A robot pozícióváltoztatási sebessége a „Sebesség” nevezetű csúszkán állítható.

Fuzzy vezérlés esetén a robot egy adott irányban (pl.: előre) adott sebességgel mozog, amely sebességet a szabályzó határozza meg a távolságszenzor által küldött adatoktól, azaz a távolságtól függően.

4. Összegzés

Projektfeladatom keretében megvalósítottam egy távirányítós játékautó átépítését egy számítógépről vezérelhető robottá. A megvalósítás során megépítettem a robot működtetéséhez szükséges áramköröket, illetve illesztettem őket a robot ARM mikrovezérlő alapú fejlesztő kártyájához. Az egyes áramkörök pl. távolság érzékelő, motorvezérlő végfok, kommunikációs modul, stb. illesztése során programoztam az ARM megfelelő perifériáit, illetve implementáltam a robotot működtető mikorvezérlő szoftvert. A robot távvezérlését egy Java programozási nyelven implementált szoftver valósítja meg, amely a grafikus felhasználói felületének köszönhetően kényelmesen használható funkciókat biztosít a robot irányítására.

A későbbiekben tervezem a robot, illetve az irányító szoftver továbbfejlesztését, a roboton ultrahangos távolságszenzorokat, kamerát tervezek elhelyezni, a Java-s irányító szoftver pedig újabb funkciókkal fog bővülni, pl. kamerakép megjelenítésével, illetve mesterséges intelligenciával fogja felruházni a robotot.