AUTOMATIZÁLÁSI ÉS
KOMMUNIKÁCIÓ-TECHNOLÓGIAI TANSZÉK

Miskolci Egyetem

Gépészmérnöki és Informatikai Kar

Robottervezés, robotvezérlés

Projekt tervezés I.

Készítette: Tompa Tamás

Neptun-kód: LJQHYK

Témavezető: Dr. Vásárhelyi József

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, hiszen a robotok alkalmazása számos előnnyel jár. Az élet több területén ma is alkalmazzák őket, elterjedtek az iparban, 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, ilyen pl.: Roomba, a porszívórobot, TurlteBot, a felszolgáló robot, vagy egyéb házilag megépített fűnyíró robotok. Tehát a robotikai egy feltörekvőben lévő tudományág, egyre népszerűbb az élet minden területén, így megrendezésre kerülnek különféle robotversenyeket is. Ilyen robotverseny például az évente megrendezendő „Magyarok a Marson" nevezetű verseny is.

A „Magyarok a Marson" verseny célja egy adott feladatot elvégző robot határidőre történő megvalósítása. A versenyfeladat évről-évre változik, idén egy olyan kétkerekű robot építése a feladat, amely a pályán lévő adott területekre mágneseket tud elhelyezni, illetve adott mágnest el tud eltávolítani.

Projektfeladatom célja egy a verseny feltételeinek eleget tevő robot tervezése, megvalósítása, illetve a robotot vezérlő számítógép oldali szoftver implementálása.

2. Robot tervezése, megvalósítása

Ebben a fejezetben a tervezendő robot célját, felépítését, megvalósítását ismertetem.

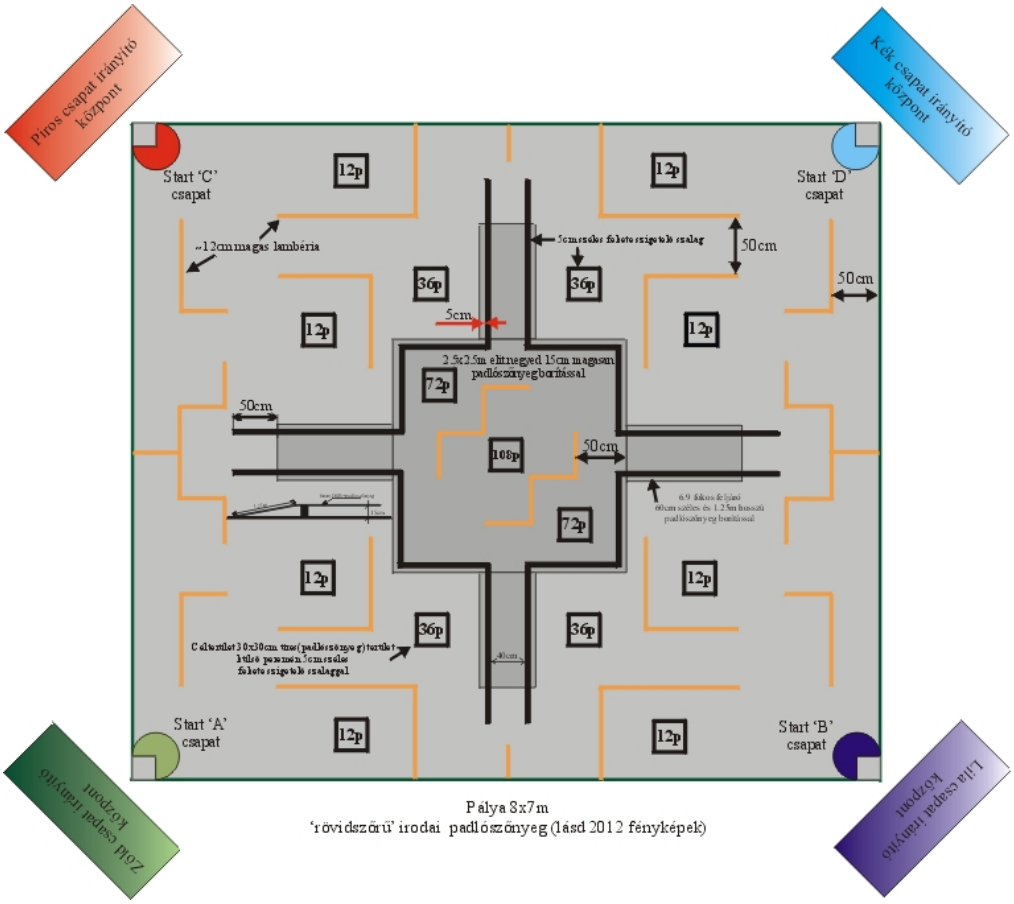
2.1 A tervezendő robot célja, feladata

[1]

A cél egy olyan robot tervezése, és megvalósítása, amely a 2013-as „Magyarok a Marson” nevű verseny követelményrendszerének eleget tesz. Ez alapján a robotnak az alábbi követelményeknek kell megfelelnie:

- csak két kereke lehet, és a két keréken kívül semmilyen része nem érheti a földet
- 4 kg-nál nem lehet nagyobb a tömege
- méretei nem haladhatják meg 25x25x25 cm-ert
- tudnia kell mágneseket elhelyezni a pálya különböző részein

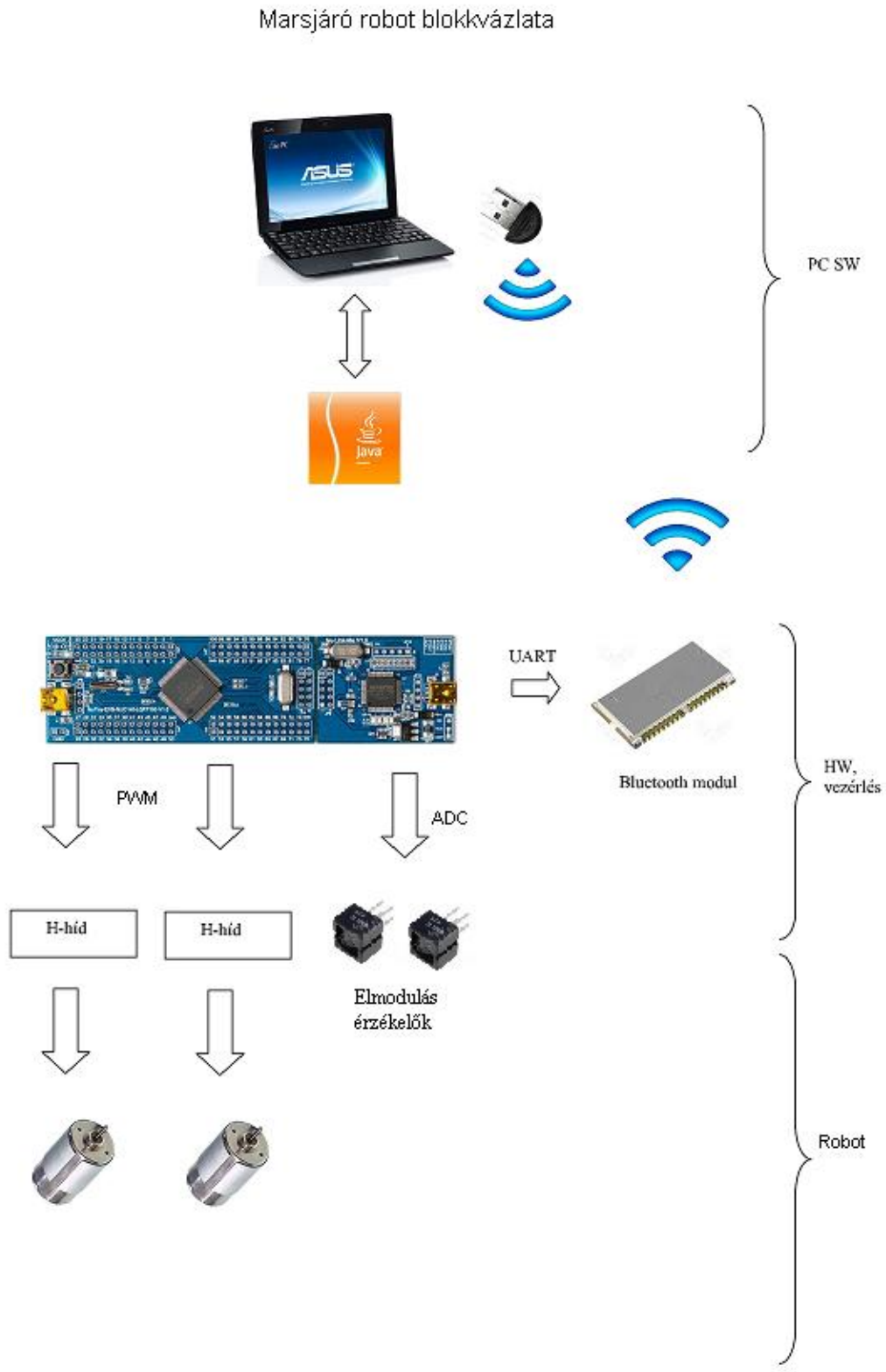
A robot verseny való feladata, hogy a pálya különböző pontot érő területeire mágneseket helyezzen el, illetve az ellenfelek mágnesét kilökje az adott területről. A következő ábra a versenypályát mutatja:



1.ábra A „Magyarok a Marson” verseny pályája

2.2 A rendszer blokkvázlata

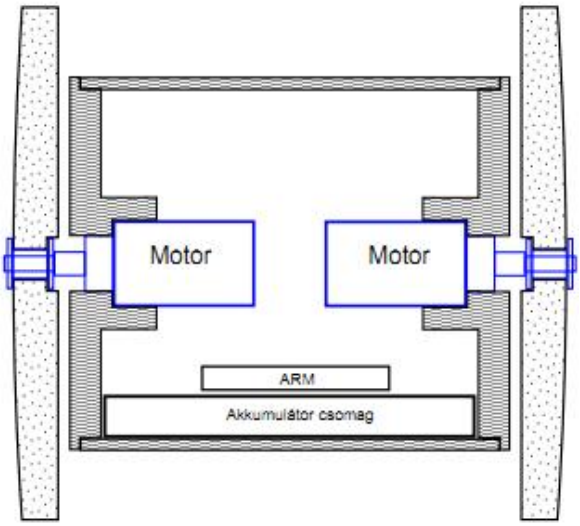
A következő ábra a rendszer felépítését szemlélteti:



2.ábra A rendszer felépítése

Az ábrán látható, hogy a számítógép egy Java programozási nyelven írt alkalmazás segítségével küldi el az irányításhoz szükséges vezérlőinformációkat az usb-s bluetooth adapternek, majd a robot bluetooth modulja fogadja a számítógépről elküldött információt és továbbítja a mikrovezérlőnek. A mikrovezérlő a fogadott adatok alapján pedig 2 különböző PWM jelet generál, amelyeket továbbítja a h-híd-nak. A motorok vezérlése különböző PWM jelek segítségével történik, tehát mindkét motor egymástól függetlenül vezérelhető. Az elmozdulás érzékelők a robot által megtett távolság méréséhez szükségesek.

A robot felépítése egy hengerhez hasonlít leginkább, azaz a robotvázat egy hengerszerű test alkotja, melyhez a két kerék csatlakozik, a robot áramkörei pedig a robot vázat jelentő hengerben foglalnak helyet. A robot mozgása tankszerű, azaz a két kereket meghajtó motor egymástól függetlenül vezérelhető, így biztosítva a jármű adott irányba történő haladását.



3.ábra A robot belső felépítése

2.3 A rendszer részletes felépítése

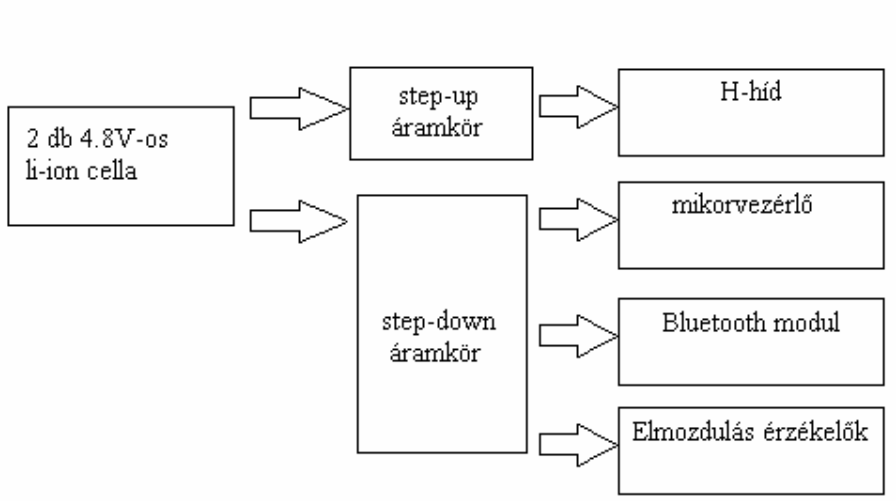
2.3.1 Tápellátás

[2]

A robot egyes moduljai számára szükséges tápfeszültséget két darab sorosan kapcsolt li-ion ipari cella biztosítja, a cellák feszültsége 4.2V egyenként. Az egyes modulok más és más tápfeszültséget igényelnek, a mikrovezérlő üzemszerű működéséhez szükséges tápfeszültség 2.5V-4.5V között, a BTM-222 bluetooth modul üzemszerű működéséhez

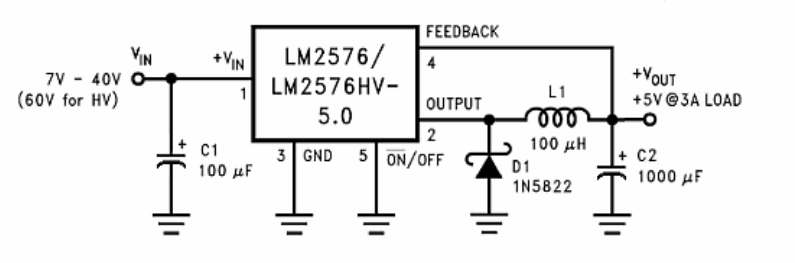
szükséges tápfeszültség pedig a 3.0V-3.6V-os tartományban van. Ettől alacsonyabb tápfeszültségek esetén a helyes működés nem garantálható, túl magas tápfeszültség hatására pedig az adott eszköz tönkre mehet, tehát minden egyes modul számára külön áramkör szükséges, amely a 2*4.2V-ból a számára megfelelő elektromos feszültséget előállítja. Ezt a feladatot DC-DC konverterek valósítják meg. A mikorvezérlő és a bluetooth modul számára szükséges 5V-ot egy step-down konverter, a h-híd működéséhez szükséges 11V-ot pedig egy step-up áramkör állítja elő.

A 4. ábra a tápellátás vázlatát, a 5. ábra a li-ion ipari cellát, az 6. ábra pedig a step-down konverter kapcsolási rajzát szemlélteti.



4.ábra A tápellátás blokkvázlata

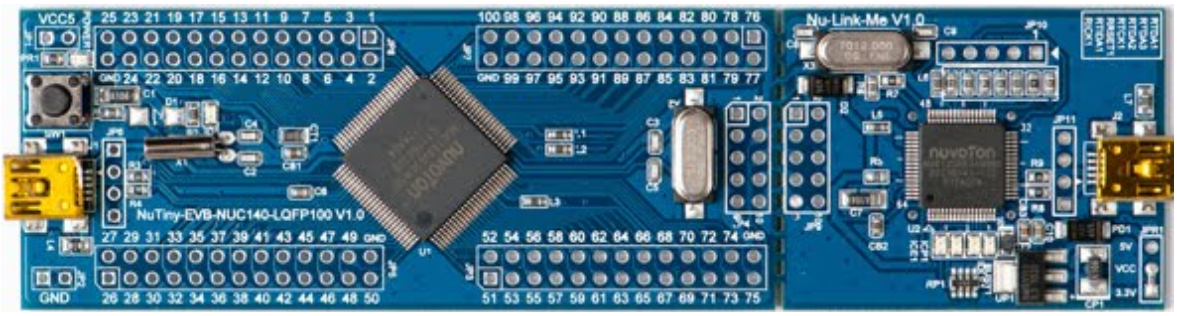
5.ábra A li-ion ipari cella



6.ábra A step-down konverter kapcsolási rajza

2.3.2 A mikorvezérlő

A robot lelkét a Nuvoton NUC140-es kit-en lévő ARM mikorvezérlő alkotja, ez a mikrovezérlőkit tartalmazza a mikrovezérlőt és a felprogramozásához szükséges égetőt egy NYÁK lapkán, előnye, hogy kis mérete ellenére is számos perifériával (uart, 4 darab pwm csatorna, 8 darab adc csatorna, SPI, CAN, I2C) rendelkezik.



7.ábra A mikorvezérlő kit

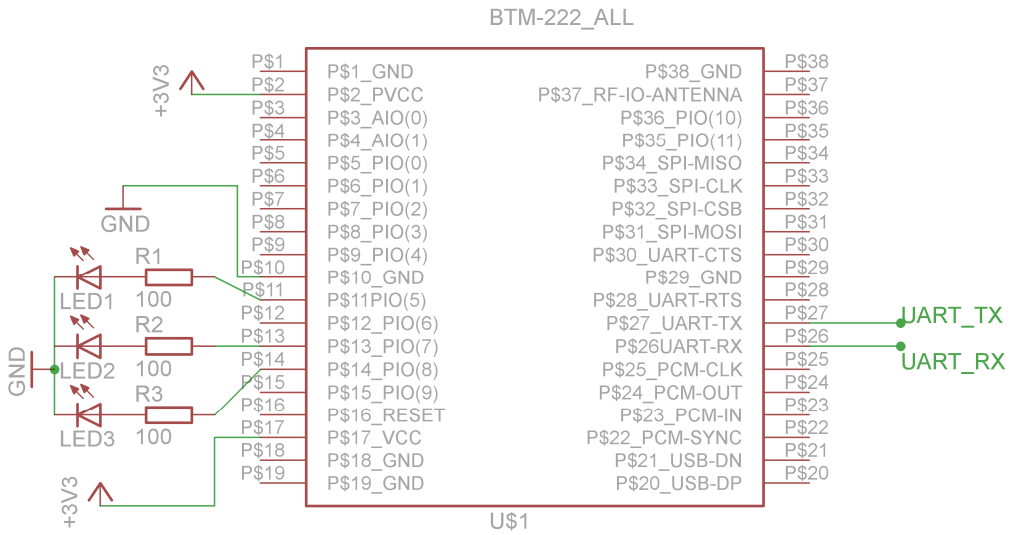
2.3.3 A bluetooth modul

A megvalósítás során BTM-222 típusú modul került alkalmazásra, ez a modul class 1-es típusú, amely azt jelenti, hogy kb. 100 méter a hatótávolsága. Konfigurálni AT utasítások segítségével lehet, de jelen esetben erre nem volt szükség, mert az SPP-t (Serial Port Profile) minden konfigurálás nélkül tudja kezelni. Ha mégis konfigurálásra lenne szükség, akkor azt a PC soros portján keresztül lehet megvalósítani egy terminálprogram segítségével.

A modul 3.3V-os megtáplálást igényel, egyéb más feszültségforrás esetén célszerű egy stabilizátor IC-t alkalmazni, mely képes adott feszültségtartományból 3.3V-os feszültséget előállítani. A modul működéséhez szükséges tápfeszültséget az 5V-ot szolgáltató step-down konverterrel sorosan kapcsolt diódák állítják elő.

Ez az eszköz beépített antennával nem rendelkezik, így erről kell gondoskodni szükséges. Jelen esetben az antennát egy 3 cm hosszúságú vezetékdarab alkotja, de akár WIFI antenna is alkalmazható e célra, csak abban az esetben nagy hangsúlyt kell fektetni az árnyékolásra.

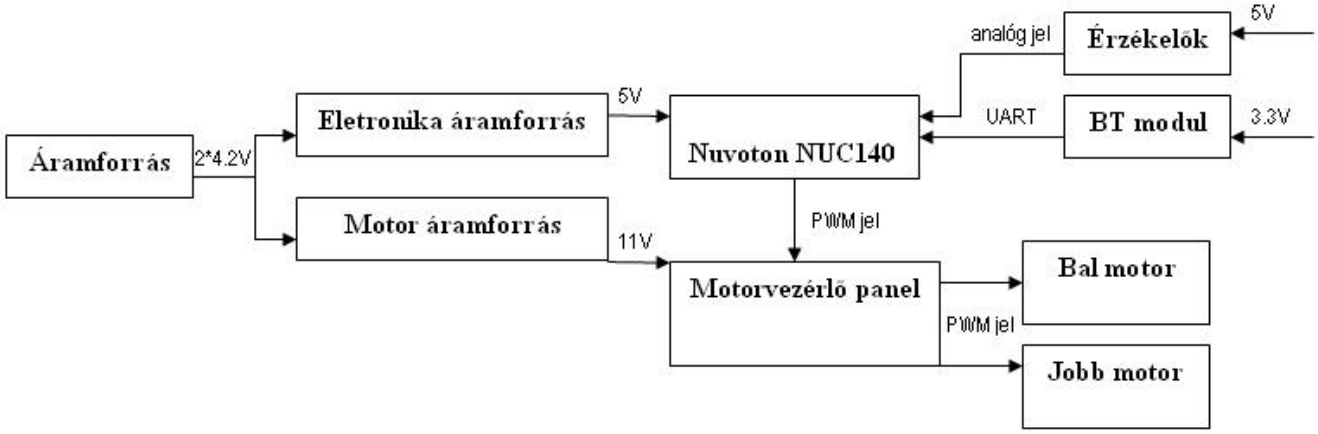
A következő ábra a modul bekötési rajzát szemlélteti:



8. ábra A BTM-222 bluetooth modul bekötési rajza

2.3.4 A robot blokkvázlata

A következő ábrán a robot felépítése látható:



9. ábra A robot blokkvázlata

Az ábrán látható, hogy a fő áramforrásból, amely két darab sorosan kapcsolt li-ion ipari cella, két külön áramkör állítja elő az egyes modulok számára szükséges tápfeszültséget, illetve látható az egyes modulok és a mikorvezérlő közötti kommunikációs kapcsolat.

2.4 Az elkészült robot

A következő 8. ábrán az elkészült robot fényképe látható:

10. ábra Az elkészült robot

3. Alkalmazás tervezése

Ebben a fejezetben a robotot vezérlő számítógép oldali szoftver megvalósításának lépéseit ismertetem.

3.1 A szoftver célja

A feladat egy olyan alkalmazás tervezése és implementálása, amely lehetőséget biztosít az elkészült robot kézi illetve automata vagy részben automata irányítására. Kézi irányítás alatt azt értem, mikor a robot ember által kiadott utasításokat fogad (pl.:menj előre 10cm-t, fordulj jobbra, stb.). Automata vagy részben automata vezérlés esetén, az alkalmazás irányítja a robotot, a robot szenzoraitól érkező jelek alapján.

A kézi vezérlés történet pl. billentyűzet vagy joystick segítségével, automata irányítás egyik lehetséges módja pl. a robot mozgásterét látó a kamera által adott képre történő útvonal rajzolása.

Elvárások az alkalmazással szemben:

- kapcsolat létrehozása a robottal
- különböző irányítási (vezérlési) lehetőségek
- kiadott utasítások késleltetésének megjelenítése
- kamerakép megjelenítése
- paraméter beállítási lehetőségek
- mindehhez grafikus felhasználói felület biztosítása

3.2 Megoldási lehetőségek

[3]

Többféle megoldási lehetőség is kínálkozik. Léteznek különböző robotvezérlő szoftverek, operációs rendszerek, ilyen például a ROS (Robot Operating System) is.

A ROS egy olyan operációs rendszer, amely könyvtárai segítségével lehetővé teszi különböző robot vezérlő szoftverek 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 egymással kommunikálnak, tehát egy robotot vezérlő ROS alkalmazás több kisebb programból épül fel. Hátrányai közé sorolnám, hogy

a rendszer felépítése meglehetősen összetett, bonyolult, illetve rendszer feltelepítése, konfigurálása sem egy egyszerű pár perces feladat.

Különböző programozási nyelveknek (pl.: C++, C#, Java) léteznek a robotk programozását elősegítő könyvtárai, csomagjai, melyek segítségével, szintén megvalósítható a feladat.

Bár a ROS sokrétű, és igen sok lehetőséget biztosít robotok vezérlésre, mélyebb ismeretének hiánya és rendszer bonyolultsága miatt saját alkalmazás fejlesztése mellett döntök, melyet Java programozási nyelven kívánok megvalósítani.

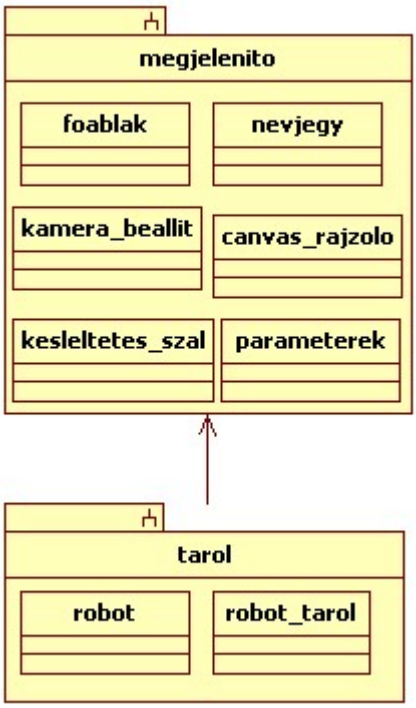
3.3 A rendszer felépítése

A fentebb említett elvárásokból következtetetés a szükséges alrendszerekre:

- Megjelenítési alrendszer
    - grafikus felhasználói felület biztosítása
    - robot irányítását biztosító funkciók megvalósítása
    - kamerakép beöltése
    - utasítások késleltetésének megjelenítése
    - útvonal rajzolása
    - útvonal rajzolás alapján történő vezérléshez szükséges adatok számítása

- Tároló alrendszer
    - robot objektumok létrehozása
    - robot objektumok tárolása

A következő ábra a rendszer felépítését szemlélteti:



11.ábra A rendszer alrendszerei és azok kapcsolata

Megjelenítési alrendszer

Célja a rendszer funkcióinak grafikus felhasználói felületen való megjelenítése. A számára szükséges adatokat maga az alrendszer számolja illetve a vele kapcsolatban álló alrendszer továbbítja. Ilyen számolandó adatok például a rajzolt útvonal hosszának megállapítása cm-ben, a rajzolt útvonal részek közötti közbezárt szögek, illetve az irányok meghatározása.

Tároló alrendszer

Célja a robot objektum/objektumok létrehozása, az objektumok adatszerkezetben való tárolása.

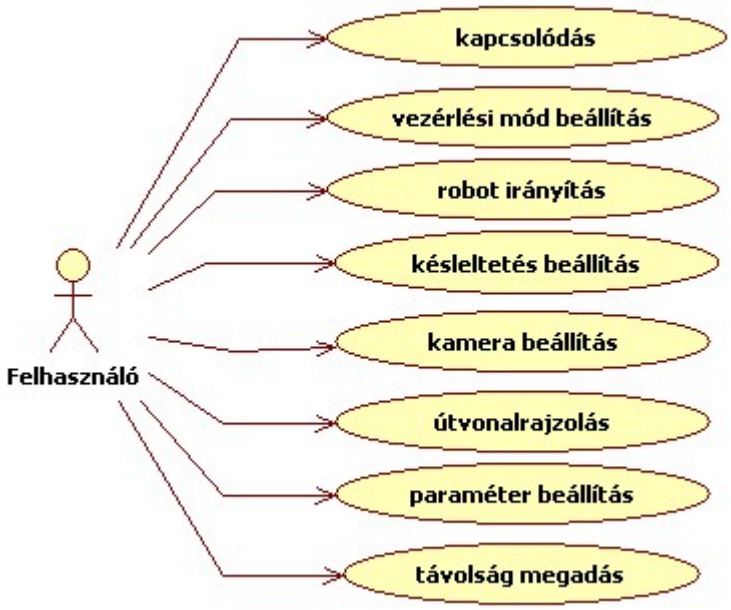
3.4 A rendszer funkciói

[4][5]

Az alkalmazás funkcióit use-case diagram segítségével szemléltetem melynek célja, hogy a szoftverfejlesztés kezdeti fázisában rögzítse a szoftverrel szemben támasztott követelményeket, illetve, hogy komplettem leírja a rendszer funkcióit. A diagram tulajdonsága, hogy szemléletes és könnyen áttekinthető. Elemei az aktorok, use case-ek, és relációk. Az aktorok egy szerepkört reprezentálnak, tulajdonképpen azok a felhasználók, akik a rendszert használják. A use case-ek viselkedési minták, magát a funkciót jelölik, a relációk pedig a use case-ek és az aktorok közötti kapcsolatokat írják le.

A funkciók magyarázatában szereplő prekondíciók azon feltételek felsorolását jelentik, melyeknek teljesülniük kell ahhoz, hogy a use case által jelzett tevékenység elkezdődjön, a post kondíciók pedig azoknak a leírása, hogy milyen állapot következik be a use case végén. A szokásos működés azon eseményeket írja le, melyek a use case „szokásos" körülmények közötti működését jellemzik. A kivételes esetek olyan lehetséges eseményeket jelölnek, amelyek váratlanul következhetnek be.

A következő ábrán a rendszer use-case diagramja látható:



12.ábra a rendszer use-case diagramja

Aktor

Felhasználó: tulajdonképpen az alkalmazást használó személy, alkalmazásbeli jogai nincsennek korlátozva.

Use case-ek

Kapcsolódás: Célja a robottal való kommunikációs kapcsolat létrehozása. Ez jelen esetben bluetooth-on keresztül valósul meg, „serial port profile” (SPP) használatával.

- Prekondíció: a program elindítása, SPP beállítása valamelyik soros port-ra
- Post kondíció: a kommunikációs kapcsolat létrejön
- Szokásos működés: a felhasználó megadja a SPP szolgáltatás során felhasznált soros port-ot, majd megnyomja a csatlakozás gombot
- Kivételes esetek: nem sikerült létrehozni a kapcsolatot, ebben az estben az alkalmazás egy felugró ablak formájában tájékoztatja a felhasználót a keletkezett hibáról.

Vezérlési mód beállítás: Célja a robot irányítási módjának kiválasztása.

- Prekondíció: kapcsolat létrejötte a robottal
- Post kondíció: a kiválasztott vezérlési/irányítási mód beállítódik
- Szokásos működés: a felhasználó által kiválasztott vezérlési mód segítségével változtatja a robot a pozícióját
- Kivételes esetek: ha nem él a kommunikációs kapcsolat a robottal és ezért nem sikerült az utasítás küldés, vagy hogy ha nem sikerült az adott irányítási mód beállítása, akkor arról a program egy felugró ablak segítségével tájékoztatja a felhasználót

Robot irányítása: Célja a robot pozíciójának a megváltoztatása

- Prekondíció: a robottal való kommunikációs kapcsolat létrejötte, illetve az adott vezérlési mód beállítása
- Post kondíció: a robot megváltoztatja a pozícióját
- Szokásos működés: az adott vezérlési mód beállításától függően - amely lehet kézi vagy félautomata – a felhasználó billentyűzet vagy útvonalrajzolás segítségével irányíthatja a robotot

- Kivételes esetek: ha nem él a kommunikációs kapcsolat a robottal és ezért nem sikerült az utasítás elküldése, akkor arról a program egy hibaüzenet segítségével tájékoztatja a felhasználót

Késleltetés beállítása: Célja, hogy a kiadott utasítások – legyen az emberi vagy program által generált – megadott időtartamnyit késsenek.

- Prekondíció: a robottal való kommunikációs kapcsolat létrejötte, illetve az adott vezérlési mód kiválasztása
- Post kondíció: a robotnak kiadott utasítások a felhasználó által beállított időt (másodperben) késnek
- Szokásos működés: a felhasználó egy beviteli mezőben megadja, hogy az utasítások mennyi időt (másodpercet) késsenek, majd egy gomb segítségével nyugtázza a beállított késletetési időt
- Kivételes esetek: ha a felhasználó helytelen késleltetési idő ad meg (például negatív számot, vagy tört számot) akkor arról a program egy hibaüzenet formájában tájékoztat, és újból bekéri a késletetési időt, addig míg a megadott érték helyes nem lesz

Kamera beállítása: Célja, hogy az irányítást elősegítő kamera képét megjelenítse

- Prekondíció: működő kamera és internet kapcsolat
- Post kondíció: a kamera képe megjelenik a program felhasználói felületén
- Szokásos működés: a kamera elérhetőségét egy beviteli mezőben megadja a felhasználó (URL link formájában)
- Kivételes esetek: ha a megadott link formátuma hibás, vagy nem sikerült a kamerához való kapcsolódás, akkor arról a program egy hibaüzenet formájában tájékoztat

Útvonalrajzolás: Félautomata irányítási mód, mikor a megrajzolt útvonal alapján a szoftver utasításlistát küld a robotnak.

- Prekondíció: a robottal való kommunikációs kapcsolat létrejötte, az „automata” vezérlési mód kiválasztása, illetve a kamerával való kapcsolat létrejötte
- Post kondíció: a kameraképre rajzolt útvonal alapján a robot megváltoztatja a pozícióját
- Szokásos működés: a felhasználó a felhasználói felületen lévő kameraképre az egér segítségével megrajzolja a robot útvonalát, majd rákattint az „indít” gombra

- Kivételes esetek: ha a robottal vagy a kamerával való kapcsolat megszakad, illetve, hogy ha az utasítások kiküldése sikertelen, akkor arról a program hibaüzentet küld

Paraméter beállítása: Célja, hogy a kamera által „látott” terület hosszát és magasságát meg lehessen adni cm-ben. Tehát a kamera által „látott” távolságok beállítását jelenti cm-ben, a robot ennek segítségével tudja, hogy milyen hosszú utat kell megtennie a cél eléréséig.

- Prekondíció: a robottal való kommunikációs kapcsolat létrejötte, illetve az adott vezérlési mód kiválasztása
- Post kondíció: a megadott távolságok beállítódnak
- Szokásos működés: a felhasználó beviteli mezőkben adja meg a távolság értékeket cm-ben
- Kivételes esetek: ha a megadott távolság értékek hibásak (negatív előjelűek), akkor, arról program hibaüzenetet küld és újból bekéri az adatokat

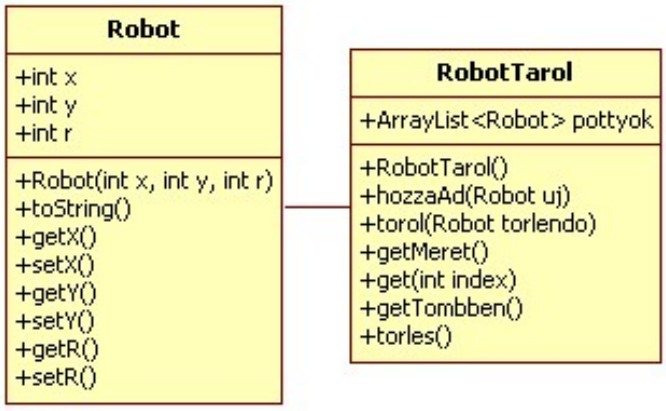
Távolság megadása: Célja annak beállítása, hogy a robot kézi irányítás esetén mennyi utat tegyen meg.

- Prekondíció: robottal való kommunikációs kapcsolat létrejötte és „manual” irányítási mód beállítása
- Post kondíció: a megadott távolságérték alapján a robot adott úthossznyit halad az adott irányba
- Szokásos működés: a távolságértéket egy csúszka segítésével beállítva a robot a beállított úthosszt (távolságot) halad az adott irányba
- Kivételes esetek: ha a megadott távolságértéket nem sikerült beállítani, akkor arról a program hibaüzentet küld

3.5 A rendszer részletesebb felépítése

Ebben az alfejezetben osztálydiagramok segítségével magyarázom az egyes alrendszerekben lévő osztályok működését, céljait.

3.5.1. A tároló alrendszer osztálydiagramja



13. ábra A tároló alrendszer osztálydiagramja

Az alábbi ábrán látható, hogy ez az alrendszer 2 osztályból, a Robot és a RobotTarol osztályokból áll. A Robot osztály 3 adataggal rendelkezik, melyek a robot pozícióját határozzák meg, az x és y adattagok a koordinátarendszer x és y tengelyein felvett értéket, az r adattag, pedig a robotot szimbolizáló kis kör sugarát jelenti. A RobotTarol osztály pottyok adattagja a robot objektumok adatstruktúrában való tárolását valósítja meg.

A Robot osztály fontosabb metódusai:

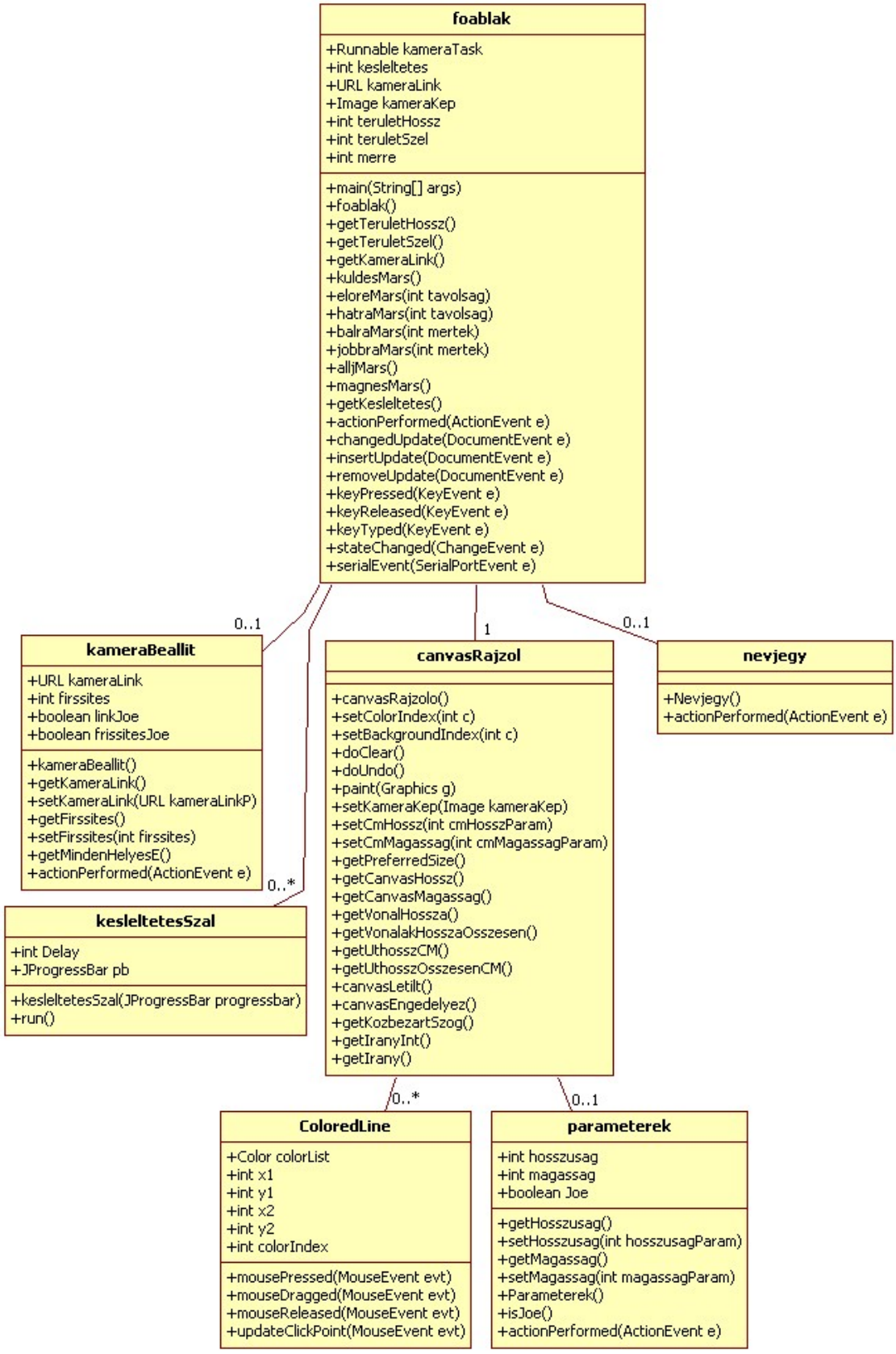
- getX(): az x koordinátát adja vissza int-ként
- setX(): az x koordináta értékét állítja be
- getY(): az y koordinátát adja vissza int-ként
- setY(): az y koordináta értékét állítja be
- getR(): a robotot szimbolizáló pötty sugarát adja vissza int-ként
- setR(): a robotot szimbolizáló pötty sugarát állítja be

A RobotTarol osztály fontosabb metódusai:

- hozzaAd(Robot uj): a paraméterében megkapott robot objektumot hozzáadja az ArrayList-hez

- torol(Robot torlendo): a paraméterében megkapott robot objektumot törli az ArrayList-ből
- getMeret(): az ArrayList méretét adja vissza egy egész számként
- get(int index): az ArrayList adott indexű elemét adja vissza
- getTombben(): Az ArrayList-et konvertálja egy tömbbé
- torles(): az ArrayList-ben lévő összes elemet törli

3.5.2. A megjelenítő alrendszer osztálydiagramja



14. ábra A megjelenítő alrendszer osztálydiagramja

Az ábrán látható, hogy ez az alrendszert 7 osztály alkotja, tulajdonképpen, ez az alrendszer végez minden számítási feladatot.

A foablak osztály jeleníti meg az alkalmazás főablakát, és a benne lévő komponenseket, felhasználói elemeket, gombokat, és ezen komponensek eseményeinek a kezelését.

A kameraBeallit osztály tulajdonképpen a webkamera beállítási ablaka, ahol a webkamera elérhetőségét, és a kamerakép frissítésének időközét lehet beállítani.

A kesleltetesSzal osztály kezeli a program főablakában lévő utasítások késleltetését jelző folyamatjelzőt.

A canvasRajzol osztály, az alkalmazás főablakában megjelenő canvas-t állítja elő. Ebben a canvas-ban töténik az útvonal rajzolása és a kamerakép megjelenítése is. Ez az osztály számítja, a megadott paraméterek alapján, a megrajzolt útvonal adatait (hosszúság, közbezárt szög, irány), melyek alapján generálódnak a robot számára megfelelő utasítások.

A ColoredLine osztály valósítja meg magát az útvonalat, beállítja a rajzolt útvonal színét, illetve kezeli az egér eseményeit.

A paramterek osztály az útvonalrajzoláshoz szükséges paramétereket állítja be. Ilyen paraméterek a webkamera által „látott” távolságok, melyek alapján az útvonalhossz számítódik.

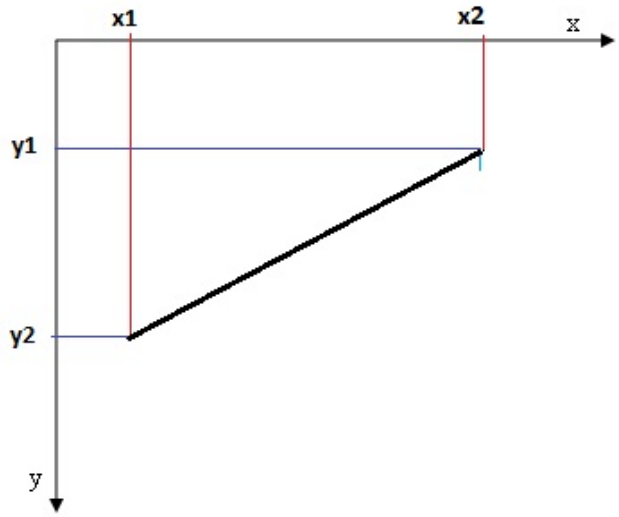
3.5.3 Szükséges matematikai számítások

Matematikai számításokra főként az útvonalrajzolás során van szükség. Szükséges számítási feladatok többek között a robot pozíciójának a meghatározása, a rajzolt útvonal hosszának meghatározása cm-ben, az egyes útvonalszakaszok által közbezárt szögek meghatározása, valamint az egyes útvonalszakaszok irányának (jobbra, balra) megállapítása. Ezen adatok alapján „tudja” a robot, hogy mennyi cm-ert kell haladnia, ezt milyen irányba, illetve, hogy mekkora szögben kell elfordulnia. Ezen számítási feladatokat a számítógép oldali szoftver végzi el és a számított adatokat továbbítja a robot számára.

Úthossz meghatározása

Célja a kameraképre rajzolt útvonalrészek, illetve az egyes útvonalrészekből összetevődő teljes útvonalhossz megállapítása centiméterben. Alapötletem, hogy a kamera által „látott” távolságokat (szélesség, magasság) ismerem pixelben, majd ezen távolságadatok alapján a szoftver egy Pitagorasz-tétel segítségével számítja az útvonal hosszát centiméterben. A kamera által „látott” távolságok a szoftverben egy külön ablakban paraméterezhetőek. A

módszert a következő ábra szemlélteti, ahol a fekete vonal az x,y koordinátarendszerbeli útvonalat szimbolizálja:



15. ábra Az útvonalhossz számítása

Tehát ismerem az útvonal kezdőpontjának koordinátáit (a robot aktuális pozíciója is egyben), ez az ábrán az (x1,y2) koordinátájú pont, illetve ismerem az útvonal végpontjának koordinátáit is, ez az (x2,y1) koordinátájú pont. A Δx és Δy a következőképpen számítható Δx=x2-x1, Δy=y2-y1. Pitagorasz-tétel segítségével az útvonal hossza (pixelben) a következőképpen áll elő: $\Delta x^2+\Delta y^2=$úthossz^2, tehát úthossz$=\sqrt{(\Delta x^2+\Delta y^2)}$. Utolsó lépésként pedig a felhasználó által megadott, a kamera által „látott” távolságok alapján ez a pixelben megkapott távolságérték átszámítódik centiméterre.

Pl.: kezdőpont: $P_1(10,100)$, végpont $P_2(400,5)$

Δx=x2-x1=400-10=300 px

Δy=y2-y1=5-10=-5 px

Úthossz$=\sqrt{(300^2+(-5)^2)}=30.41px\approx$30px

Ha a felhasználó által cm-ben megadott távolságok alapján tudom, hogy egy pixel pl. 2cm-nek felel meg, akkor 30px*2cm=60cm az útvonal hossza.

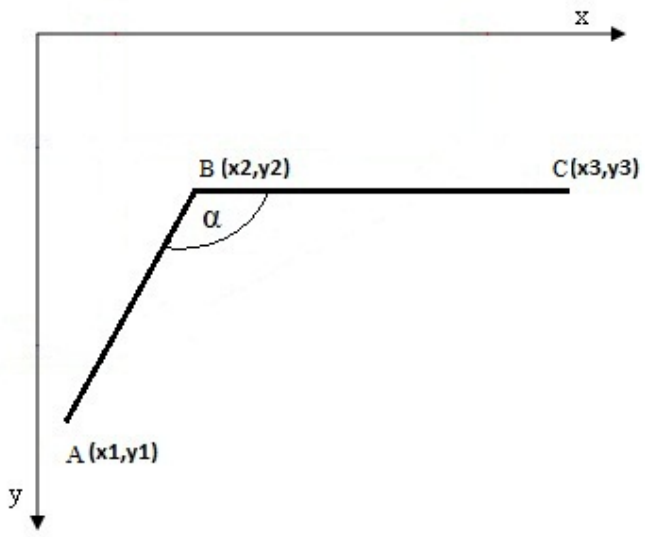
A számítást végző kódrészlet:

```
public double getUthosszCM() {
        double egyPxHosszCM = (double)cmHossz / (double)this.canvasHossz;
        double egyPxMagassagCM = (double)cmMagassag / (double)this.canvasMagassag;

        double xTav = Math.abs((double)this.startX-(double)this.prevX) * egyPxHosszCM;
        double yTav = Math.abs((double)this.startY-(double)this.prevY) * egyPxMagassagCM;
```

```
        double atmeneti = (Math.pow(xTav,2) + Math.pow(yTav,2));
        double hosszCM =  Math.sqrt(atmeneti);

        return hosszCM;
}
```

Közbezárt szög meghatározása
[6]

Feladata az egyes útvonalszakaszok közötti közbezárt szögek megállapítása, amelyek alapján a robot az adott pozícióba elfordul valamennyi fokot. Számítása szintén koordináta-geometria segítségével történik, melyet a következő ábra szemléltet:



16. ábra Közbezárt szög számítása

Az ábrán látható módon ismerem az egyes útvonalszakaszok kezdő- és végpontjának koordinátáit, jelen esetben ezek az A, B és C pontok. Az egyik útvonalszakasz végpontja megegyezik a másik útvonalszakasz kezdőpontjával. A közbezárt szög meghatározása a következő képlet alapján történik:

$$\cos\alpha = \frac{\overrightarrow{BA} * \overrightarrow{BC}}{|\overrightarrow{BA}| * |\overrightarrow{BC}|}$$ ahol A, B, C vektorok

Tehát az első lépés, az ábrán látható A, B és C pontok alapján két vektor előállítása, mely a következőképpen történik:

BA=(x1-x2, y1-y2), BC=(x3-x2, y3-y2).

Következő lépésben a két vektor összeszorzása történik:

$$BA * BC = (X_{BA}*X_{BC})+(Y_{BA}*Y_{BC})$$

Majd az egyes vektorok hosszának a megállapítása:

$$|BA| = \sqrt{(X_{BA}^2 + Y_{BA}^2)}$$

$$|BC| = \sqrt{(X_{BC}^2 + Y_{BC}^2)}$$

A fentebb lévő képletnek megfelelően, így kapott eredmények hányadosának az arccos-át véve, előáll a közbezárt α szög.

Pl.: A(100,10); B(50,30); C(40,60)

BA = (50, -20)

BC = (-10, 30)

|BA| = 2900

|BC| = 1000

BA * BC = (50*(-10))+((-20)*30) = -1100

cosα = -1100/(2900*1000) = 0.00037

α = arccos(0.00037) = 89.7 fok ≈ 90 fok

A számítást végző kódrészlet:

```
public double getKozbezartSzog() {
            if(this.lineCount>2) {
             int x1 = this.startVonal1x;
             int y1 = this.startVonal1y;
             int x2 = this.endVonal1x;
             int y2 = this.endVonal1y;

             int x3 = this.startVonal2x;
             int y3 = this.startVonal2y;
             int x4 = this.endVonal2x;
             int y4 = this.endVonal2y;

             int[] BA = new int[2];
             int[] BC = new int[2];
             BA[0] = x1-x3;
             BA[1] = y1-y3;
             BC[0] = x2-x4;
             BC[1] = y2-y4;

             double BAhossz = Math.sqrt((BA[0]*BA[0])+(BA[1]*BA[1]));
             double BChossz = Math.sqrt((BC[0]*BC[0])+(BC[1]*BC[1]));
```

```
            double felso = (BA[0]*BC[0])+(BA[1]*BC[1]);
            double also = BAhossz*BChossz;

            double reszEredemny = felso/also;

            return (Math.acos(reszEredemny))*180/Math.PI;
           }
           return 0;
 }
```

Irányok meghatározása
[7]

Feladata az egyes útvonalrészek találkozásánál lévő jobbra-balra irányok meghatározása. Szükségességét az indokolja, hogy csupán a közbezárt szögekből nem lehet megállapítani az irányt. Tehát a robotnak nem csak az az információ továbbítódik, hogy milyen szögben forduljon, hanem az is hogy milyen irányban. Pl.: balra 40 fokot, jobbra 60 fokot, a 0 fok jelenti az egyenesen irányt.

Meghatározása a következő képlettel történik:

$$irány = [(X_B-X_A)*(Y_C-Y_A)] – [(Y_B-Y_A)*(X_C-X_A)]$$

Ha az így kapott szám 0, akkor az irány az egyenesen, negatív vagy pozitív szám esetében, pedig a jobbra vagy balra.

A számítást végző kódrészlet:

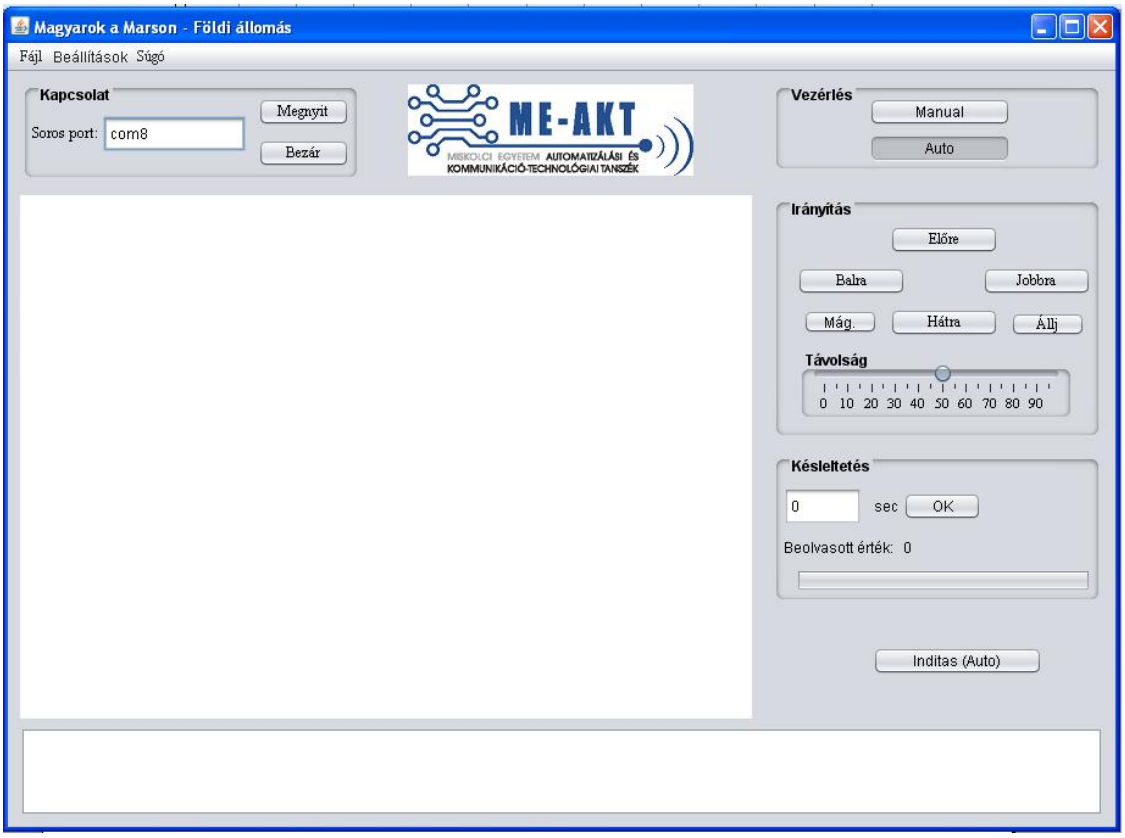
```
public int getIranyInt() {
            int irany = (this.startVonal2x - this.startVonal1x)*(this.endVonal2y - this.startVonal1y) -
            (this.startVonal2y - this.startVonal1y)*(this.endVonal2x - this.startVonal1x);

            /* egyenesen */
            if(irany == 0) {
                    return 0;
            }

            /* balra */
            if(irany < 0) {
                    return 1;
            }

            /* jobbra */
            if(irany > 0) {
                    return 2;
```

```
        }
        return -1;
}
```

3.6 Az alkalmazás használata

Az alkalmazás elindítása után első lépés a robottal való kommunikációs kapcsolat létrehozása. A kommunikáció bluetooth-on történik, tehát első lépésben pl.: a BlueSoleil nevezetű alkalmazás segítségével fel kell deríteni a bluetooth-on látható eszközöket, majd kapcsolódni a robothoz SPP-n keresztül. A következőekben a BlueSoleil által létrehozott soros portot kell megadni a program „Kapcsolat” nevű beviteli mezőjében. Sikereses illetve sikertelen kapcsolódás esetén is a program egy felugró ablak segítéségvel tájékoztatja a felhasználót. A folyamatot a következő ábrák szemléltetik:

17. ábra A robot felderítése, SPP kapcsolat létrehozása



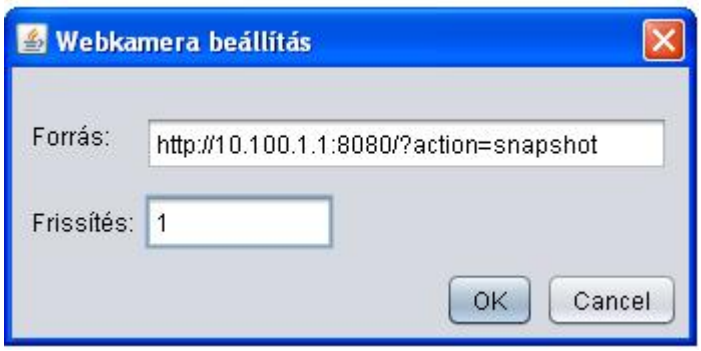
18. ábra Kapcsolódás a robothoz

A következő lépésben az irányítási mód kiválasztása szükséges. Mind „manual” mind „auto” vezérlési mód esetén szükséges az utasítások késleltetési idejének megadása. Ez azt jelenti, hogy pl. 10 másodperces késleltetés esetén a robotnak elküldött utasítások 10 másodperc múlva hajtódnak végre. A késleltetés úgy működik, hogy az első utasítás késik 10 másodpercet, majd a többi utasítás követi az elsőt. Ha például az első utasítás az hogy „menj előre 40 cm-t”, majd pl. 3 másodperc múlva a második utasítás az, hogy „fordulj balra”, akkor az első utasítás késik 10 másodpercet, de a második már őt követi, tehát 3 másodperccel ez első utasítás után fog megérkezni. A kiadott utasítások késleltetését egy folyamatjelző szemlélteti.

„Manual” irányítási mód esetén a robot, a program felületén lévő gombokkal vagy a billentyűzet adott gombjaival irányítható.

„Auto” irányítási mód esetén, szükséges a kamerakép és a paraméterek beállítása. A kamerakép elérhetőségét, és a kamerakép frissítésének időközét az alkalmazás „Beállítások”->”Kamera” menüpontjában lehet megadni. Ha a kamerához való kapcsolódás sikeres volt, akkor a kamera képe megjelenik az alkalmazásban majd a

megadott időközönként újrarajzolódik (azaz frissül). A beállítást a következő ábra szemlélteti:

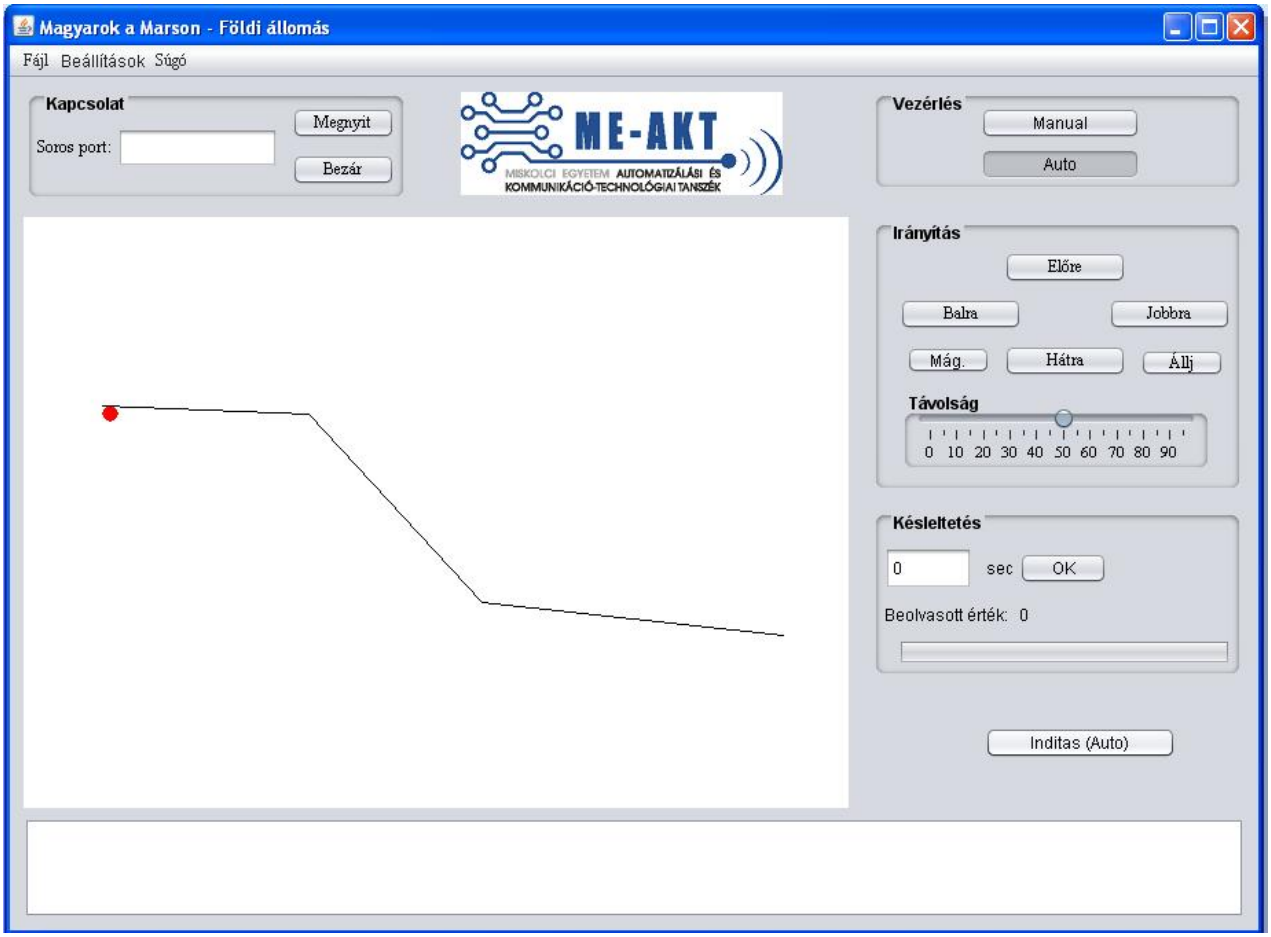


19. ábra A webkamera beállítása

Az útvonal rajzolásához fontos beállítani az úgynevezett paramétereket, ezek tulajdonképpen a kamera által „látott" távolságok megadását jelentik, ezen adatok alapján számítódik a rajzolt útvonal hossza. A következő 12.ábrán látható, hogy a beállított távolságok 7m*6m.

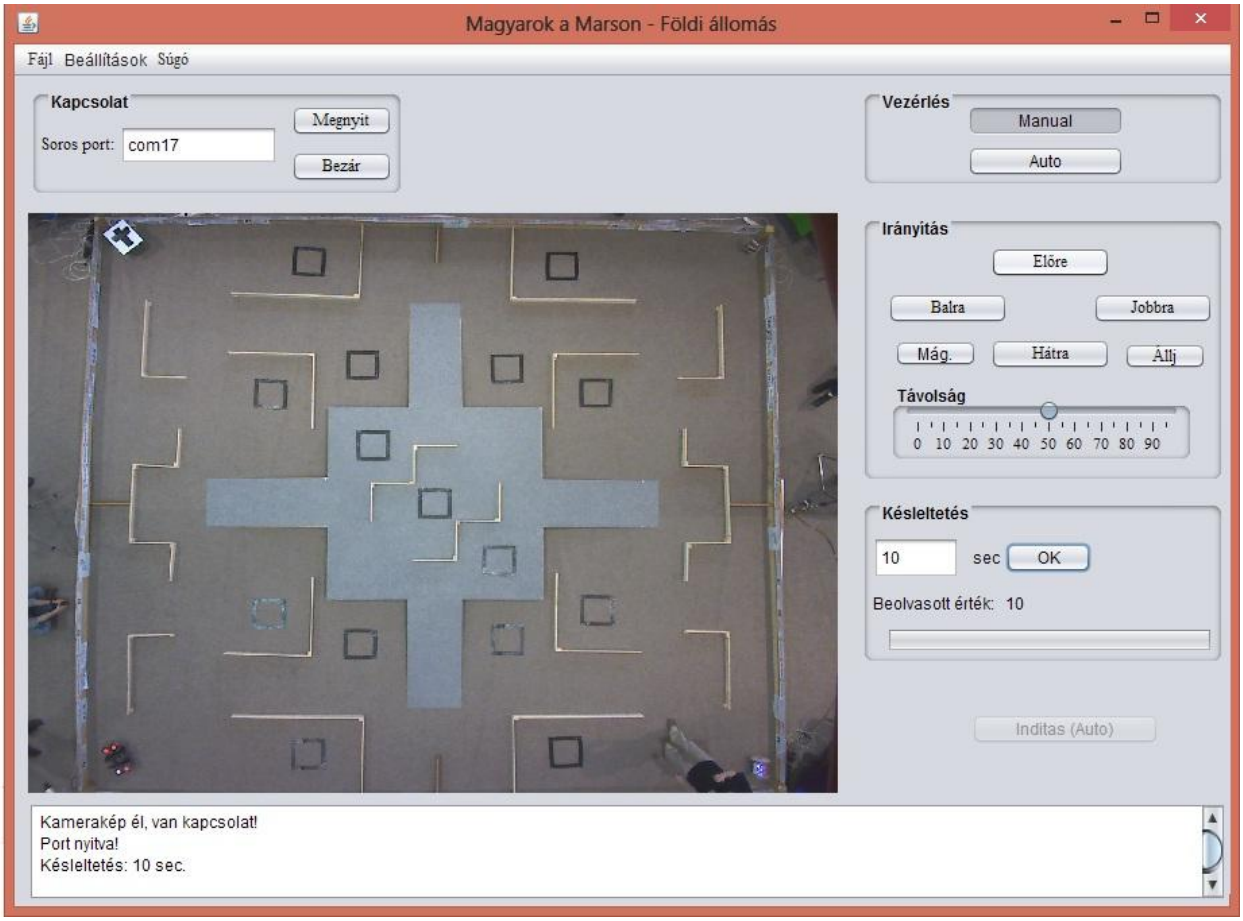
20. ábra A paraméterek beállítása

Az útvonal rajzolása az egér gombjainak segítségével történik. Az egér bal gombjával lehet rajzolni, a jobb gombjával pedig törölni az adott útvonalat. A megrajzolt útvonal alapján generált utasításlista egyszerre küldődik el a robotnak, az „Indítás(Auto)" gomb segítségével. Ezt a következő 13. ábra szemlélteti.



21. ábra Útvonal rajzolása

A következő 14. ábrán pedig a program látható működés közben a „Magyarok a Marson” verseny alatt.



22. ábra Az alkalmazás működése a verseny alatt

4. Összefoglalás

Projektfeladatom megvalósítása során elkészült egy kétkerekű robot és a robotot vezérlő számítógép oldali szoftver. A számítógép oldali szoftver segítségével lehetőség van a robot kézi illetve félautomata irányítására. Kézi üzemmód esetében a program egy „joystick”-ként működik, félautomata üzemmód esetében pedig, egy a robot mozgástere felé helyezett kamera által szolgáltatott kameraképre történő útvonal rajzolásával van lehetőség a robot vezérlésére. Az elkészült robot illetve irányító szoftvere számos továbbfejlesztési lehetőséget tartogat. Ilyen lehet például egy teljesen autonóm vezérlés megvalósítása, amely során a robot saját maga térképezi fel környezetét, majd a feltérképezett környezetet térképként tárolva az irányító szoftverében, saját maga keresheti meg céljait.

A megvalósítás során mélyebben bepillantást nyertem a robotika témakörébe, megismerkedtem a robotika alapjaival, mélyebb ismereteket szereztem az ARM mikorvezélő működéséről, felépítéséről, programozásáról.

Irodalomjegyzék

[1] *http://www.magyarokamarson.hu/index_elemei/kuldetes2013.html*

[2] *http://www.hobby-hour.com/electronics/lm2576-5v-buck-regulator.png*

[3] *http://en.wikipedia.org/wiki/Robot_Operating_System*

[4] *http://dcs.vein.hu/goth/oktatas/he.pdf*

[5] *http://users.iit.uni-miskolc.hu/~mileff/szt/srs.html*

[6] *http://www.vitutor.com/geometry/vec/angle_vectors.html*

[7]*http://stackoverflow.com/questions/3461453/determine-which-side-of-a-line-a-point-lies*

Linkek ellenőrzésének dátuma: 2013.05.05.

Köszönetnyilvánítás

Köszönöm Dr. Vásárhelyi József konzulensemnek, hogy lehetőséget biztosított projektfeladatom megvalósításához, köszönöm segítőkész magatartását, hasznos tanácsait, iránymutatásait.