SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM
Marosvásárhelyi Kar

SLAM módszerek vizsgálata és megvalósítása

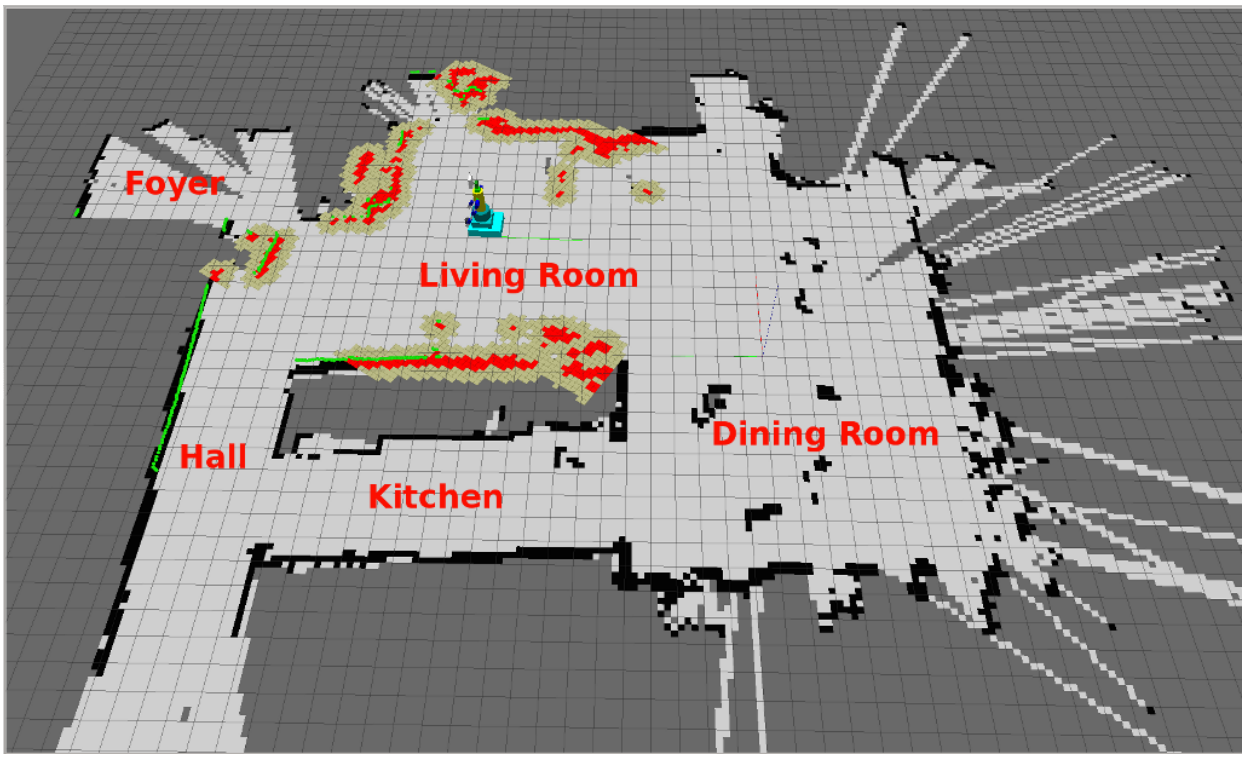
Szerző: Krizbai Csaba, Automatizálás szak III. év.
Vezető tanár: dr. Márton Lőrinc, Fehér Áron

Kulcsszavak: Lidar szenzor, Raspberry Pi,SLAM rendszer, helymeghatározás

Összefoglaló

Az ember érzékei, látás, tapintás, szaglás szoros kapcsolatban állnak a tudásával. Ezért számára egy idegen hely feltérképezése és egy mentális térkép kialakítása igencsak triviális feladat. Alapvetően egy robot valamint egy ember fő navigációs feladatai hasonlóak. Az ember érzékszerveit különböző szenzorok, valamint a gondolkodását egy vagy több mikrokontroller jellképezheti. A XXI. században az autonóm, önvezető rendszerek jelentős változáson, fejlődésen mentek keresztül és egyre nagyobb népszerűségnek örvendhetnek. A modern, precízebb rendszerek álltalában több típusú érzékelőt használnak a helymeghatározásban (Például: GPS, lézerradar, radar, stb.). Fő célja ezeknek az eszközöknek, emberi hibák kiküszöbölése, valamint a hétköznapok tehermentesítése.

Egy olyan autonóm robot megvalósítása volt a cél amely képes feltérképezni, szimulálni a környezetét. Majd az általa meghatározott helyszínen önállóan eljutni egyik pontból a másikba. A 0. ábrán már létező, háztartási robot feltérképezett képe látható.



0. ábra. Létező háztartási robot térképe

https://www.sifsof.com/clinical-apps/simultaneous-localization-and-mapping-slam/

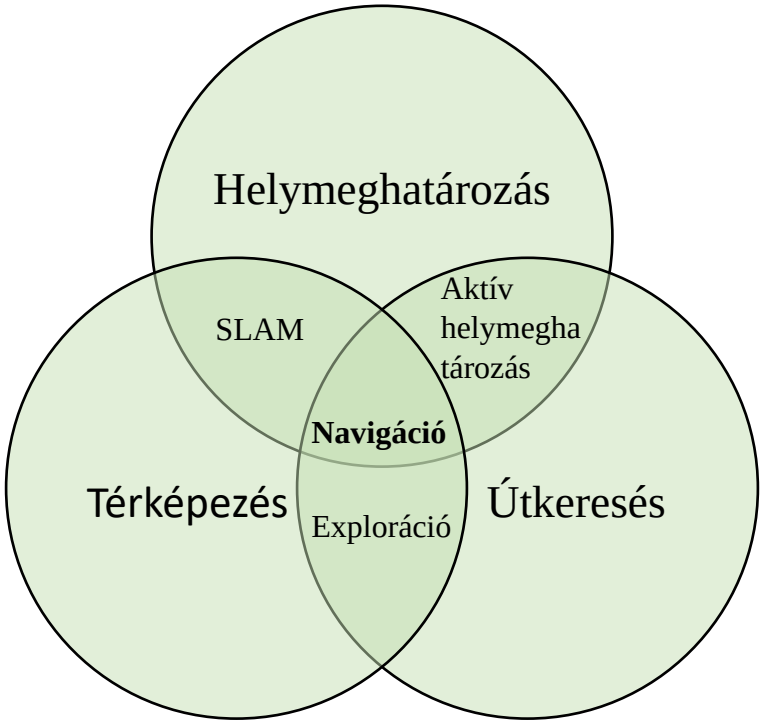
Tartalomjegyzék

Tartalom

Bevezető

Célom megtervezni és megvalósítani egy olyan robotot amely képes számára ismeretlen teremben közlekedni. A robot navigálása előtt szükségünk van, hogy hol is tartózkodik és mi van körülötte. Ahhoz, hogy tudjuk a pozícióját, ismernünk kell a környezetét . Viszont, hogy megismerje a környezetét ismernünk kell a robot pozícióját. Ez az úgynevezett tyúk – tojás probléma. Ezt a problémát a SLAM (*Simultaneous localization and mapping / egyidejű helyzetmeghatározás és térképezés*) próbálja megoldani. Ez a módszer az aktuális pozíciót az elmozdulás alapján számolja, majd a begyűjtött mérések alapján pontosítja.

Az emberi és robot navigáció között sok különbség fedezhető fel, de az alapvető problémák, feladatok megegyeznek. Ez a probléma három részfeladatra bontható fel, amelyek részben átfedésben vannak egymással. (1. ábra)



1. ábra. A navigáció három alkotóeleme és azok kapcsolatai

Manapság a GPS legnépszerűbb helyzetmeghatározó eszköz. Viszont számunkra nem koordinátára van szükségünk, hanem egy bizonyos helyszínen belüli pozícióra. A

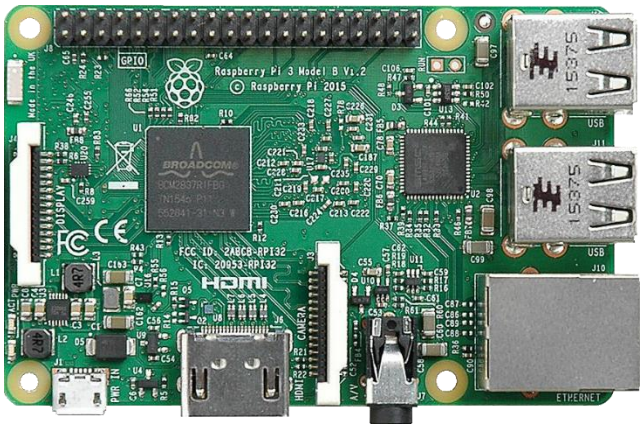
pozíció meghatározására valamint a navigáláshoz szükségünk van egy pontos helyszínrajzra, egy térképre. Ezt a térképet különböző szenzorok által szolgáltatott pontfelhők segítségével építhessük fel. Egy ilyen kigenerált térkép segítségével tudunk navigálni, vagyis utat keresni a jelenlegi pozíciónktól a kívánt helyszínig. Ezt útkeresésnek nevezzük. A robot mozgás közben is képes kell legyen akadályokat felismerni kikerülni, mindezt a lehető legoptimálisabban, ezt explorációnak nevezzük.

Szerkezeti felépítés

A robot adatfeldolgozó egységét egy Raspberry Pi 3 -as egykártyás számítógép látja el. Azért erre eset a választás, mert elegendő számításikapacitással rendelkezik, valamint a méretei is kedvezőek. A térkép kirajzoláshoz szükséges adatokat egy Hokuyu URG Lidar szenzortól gyűjti be a Raspberry Pi.

Rasperry Pi

A Rasperry Pi (2. ábra) egy bankkártya méretű kis egykártyás számítógép, amelyet az Egyesült Királyságban fejlesztettek oktatási célokra. Most már világszerte népszerű vált és nem csak oktatási célokra használják. A számítógépre többféle többnyire Linux alapú operációs rendszert választhatunk. Például: Raspbian, Pidora, Minibian, ROS, stb… Ez a termék a Broadcom cég BCM2837B0 processzora köré épül, ami egy 64 bites ARM Cortex-A53 1,4GHz-es processzor. Alapvetően 1,4GHz működik a CPU, viszont ha eléri a 70 °C, akkor teljesítmény csökkenés történik, ugyanis az órajel automatikusan 1,2GHz alá esik. Ez a kis számítógép vezeték nélküli Lan-t és 4.2-es Bluethooth kommunikációra is képes. Valamint rendelkezik két darab USB 2.0 csatlakozóval, egy HDMI videókimenettel és egy Gigabit Ethernet csatlakozóval. A Pi 3-as nagyobb teljesítményigénnyel működik, mint a korábbi társai, szóval egy 2,5 Amperes micro USB-s tápegység szükséges a működtetéséhez. A Raspberry Pi 3 jelenlegi piaci ára körülbelül 35$.



2. ábra. Raspberry Pi 3 egykártyás számítógép

Hokuyo Urg

Egy Hokuyo Urg 04LX-UG01 Lidar (3. ábra) szenzort használtam a robot környezetének feltérképezéséhez. A Lidar az ultraibolya lézerszkennelés segítségével határozza meg a környezetében lévő tárgyak közelségét. A szenzor mindössze 160g és viszonylag kis energiafelvétellel rendelkezik, a gyártó szerint 2.5W, ezért is tökéletes választás robotokra. Korlátolt a látószöge, összesen 240° tud feltérképezni, valamint 56000 milliméter a maximális látótávolsága. Ezek az értékek mellett kellő pontosan tud mérni, ±30milliméter a tévedhet. A betáplálási feszültsége 5VDC. Egy teljes feltérképezés 100ms alatt történik meg. Ennek a szenzornak az a jelenlegi ára 1,000$.

3. ábra. Hokuyo URG 04LX típusú lidar szenzor

A Hokuyo Urg RS232 szabvány szerint kommunikál a felhasználóval. A szenzor és a gazda(host) között egy előre meghatározott kommunikációs protokollon történik a kommunikáció. A szenzor előre meghatározott utasításokat vár a felhasználótól. Ha megfelelő parancs érkezett a soros porton, akkor azt, a szenzor értelmezi és választ küld vissza. A 4. ábrán látható a felhasználó által küldött üzenet felépítése. Miután ezt az üzenetet a szenzor értelmezte, az 5. ábrán látható protokollon küldi vissza a válaszát.

(HOST → SENSOR)

Command Symbol	Parameter	String Characters (Max. 16 letters)	Line Feed(LF) or Carriage Return(CR) or Both

Command

4. ábra. A gazda utasítása felépítése

(SENSOR → HOST)

Command Symbol	Parameter	String Characters	LF	Status	Sum	LF	Data	Sum	LF	LF

Command Echo

Reply

5. ábra. Szenzor válasza, a gazdának

Összesen 13 típusú parancsot tud értelmezni, ezek közül csak a legfontosabbakat fogom bemutatni. A szenzor üzembelépéskor alapértelmezetten SCIP1.1 rendszeren fut. A Hokuyo URG 04LX család már támogatja a SCIP2.0. szóval az első parancs amit minden indításkor elküldök neki az ez az üzemód váltás. Lehetőségünk van lekérdezni a szenzor paramétereit (szériaszám, gyártó, szenzor típusa, stb…) egy bizonyos VV (Command Symbol) paranccsal. Egy bizonyos MD (Command Symbol) utasítással tudjuk a lidar szenzor által mért távolságokat lekérdezni (6. ábra).

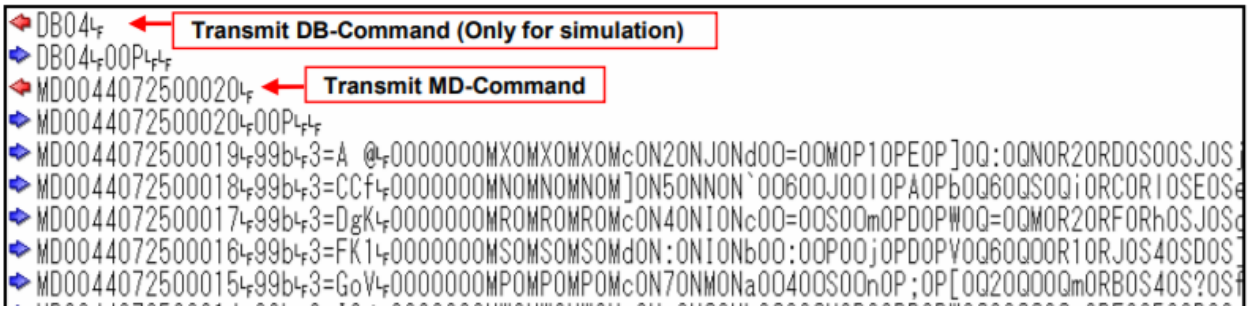
(HOST→ SENSOR)

M (4dH)	D (44H) or S (53H)	Starting Step (4bytes)	End Step (4 bytes)	Cluster Count (2bytes)	Scan Interval (1 byte)
Number of Scans (2 bytes)	String Characters (max 16-letters)	LF (1 byte)			

6. ábra. MD parancs felépítése

Az utasítás felépítéséhez szükségünk van egy kezdő pontra (Starting Step) és egy végső pontra (End Step). Ez a két paraméter a mérés kezdeti, valamint a végpontját jelenti. Jelen esetben ezek az értékek 44 és 725 között lehetnek, mivel 44 pont a 120° és 725 pont a -120° felel meg. A mérési értékeket a felhasználó átlagolhatja (Clouster Count), ez a paraméter 0-tól 99-ig változtatható.

A következő ábrán (7. ábra) látszik, hogy milyen mérési értékeket térít vissza a szenzor.

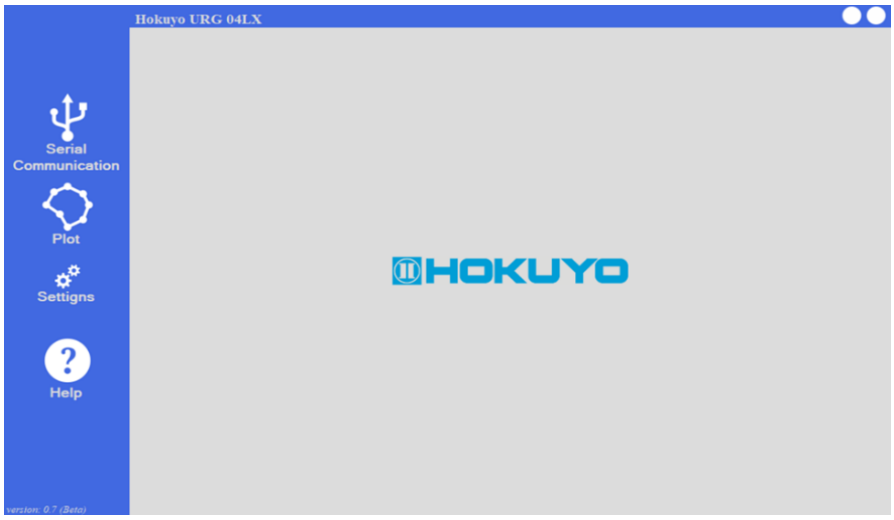


7. ábra. Szenzor válasza az MD parancsra

A Hokoyo által visszatérített távolság értékeit dekódolnunk kell. Erre összesen három fajta dekódolási lehetőségünk van. Én a két karakter (Two-Character Encoding) típusút alkalmaztam.

Távolság mérés

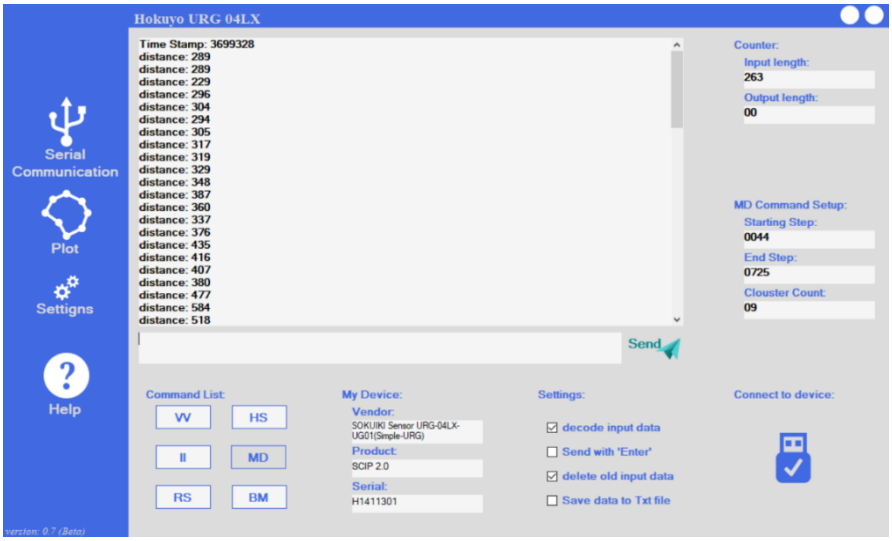
A nyárigyakorlatom során készítettem a fentiekben bemutatott szenzornak egy kezelőfelületet. Ez a szoftver Visual Studio 2019 PRO-ban, C# nyelven készült. Ez a szoftver még egyenlőre Windows operációs rendszer futtatható.



8. ábra. Kezdőfelület

A program indításával ez a kezdőfelület (8. ábra) fogad minket. Bal sávban látható a különböző navigációs gombok. A jobb oldali sarokban látható körök segítségével kiléphetünk vagy tálcára csukhatjuk a programot.

A Serial Communication menüpontban (9. ábra) a felhasználó különböző parancsokat tud küldeni a lidar szenzornak, miután sikeresen létrehozta a soros port kommunikációt, a “Connect to device” gomb segítségével. Ha a kapcsolat sikeresen létrejött, akkor a szoftver kiírja a “My Device” menüpontban a Hokuyo URG szenzor pontos jellemzőit. A felhasználó kézzel is begépelheti a kívánt szöveget, de előre megírt, gyorsgombbal küldhető parancsok is vannak “Command List” menüpontban. Az 9. ábrán látható, ahogyan a felhasználó elküldte az MD parancsot és válaszul megkapta a szenzor által leolvasott távolságokat. Ezek az értékek már dekódolva vannak és centiméterben megjelenítve, mivel a “decode input data” kivan pipálva.



9. ábra. Serial Communication menüpont

A 10. ábrán látható, hogy csatlakozás esetén nem kell kiválasztani azt a kommunikációs portot, hanem az első elérhető portra próbál rácsatlakozni. Ez azért hasznos, mert a program indulásával nem kell beállítani a helyes COM portot. Az alapértelmezett paramétereket a felhasználó tudja változtatni “Settings” menüben.

```
2 references
public static int AutoConnect()
{
    string[] ports = SerialPort.GetPortNames(); //read COM ports
    if(ports.Length > 0)
        if (SerialPortParameters(ports[0], defBaud, defdataBits, defstopBits, defParity) == 1)
            if (OpenSerialPort() == 1)
                return 1;

    return -1;
   //setup def connection value
}
```

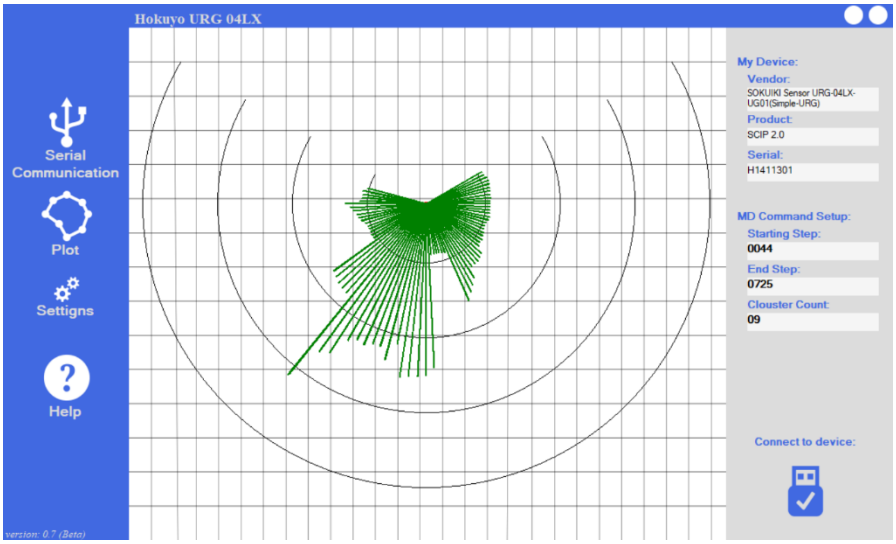
10. ábra. Automatikus serial - port kiválasztás

Az MD parancsra kapott választ dekódolását mutatja be a következő programrészlet (11. ábra). A decodeMD függvényben a beérkező adatokat válogatjuk szét. Minden sor berakok egy üres string tömbe (split_command), hogy majd tudjam őket külön kielemezni. Ezek a lehetnek jó vagy rossz mérések is. Ha a szenzor 00 értéket térített vissza, akkor az nem jó mérés, viszont, ha 99-es értéket akkor a mérés sikeres volt. Hogyha a mérés sikeres volt a “decode” függvény segítségével megtörténik a dekódolás.

```
2 references
public static bool decodeMD(string get_command, ref long time_stamp, ref List<long> distances)
{
    distances.Clear();
    string[] split_command = get_command.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);
    if (split_command[1].StartsWith("00"))
        return true;
    else if (split_command[1].StartsWith("99")) //valid measurments
    {
        time_stamp = decode(split_command[2], 4);
        distance_data(split_command, 3, ref distances);
        //MessageBox.Show("time stamp: " + time_stamp.ToString() + " distance[100] : " + distances[100].ToString());
        return true;
    }
    else return false;
}

2 references
private static long decode(string data, int size, int offset = 0)
{
    long value = 0;

    for (int i = 0; i < size; ++i)
    {
        value <<= 6;
        value |= (long)data[offset + i] - 0x30;  //Hexa 30 = Decimal 48;
    }

    return value;
}
```

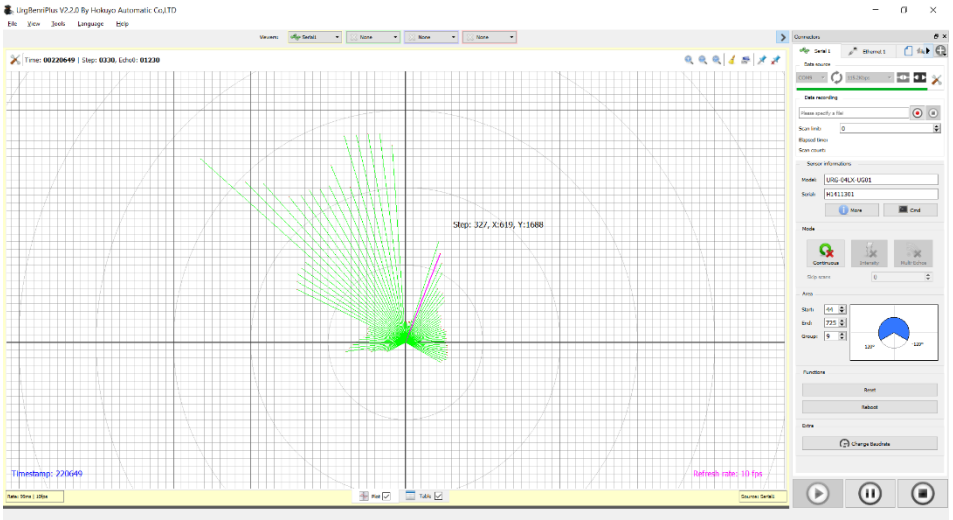
11. ábra. Beérkező mérések dekódolása

A “Plot” menü a (12. ábra) beolvasott távolságokat rajzolja ki. A zöld vonalak a távolságot szimbolizálják. A mérés paramétereit beállíthatjuk az “MD Command Setup” menüpontban.



12. ábra. Plot menü, távolságok kirajzolása

Ezeket a méréseket összehasonlítottam az URG Benri Plus program méréseivel. A 13. ábrán az URG Benri szoftver által kiolvasott értékek látszódnak. A szenzor paraméterei ugyan arra voltak állítva mind két kirajzolás során.



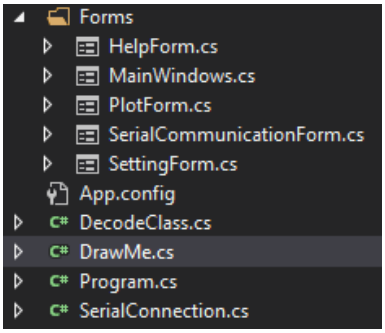
13. ábra. Urg Benri Plus szoftver mérései

```
79          public static void drawMeasurments(ref List<long> distance)
80          {
81
82              int distanceLength = distance.Count;
83              if (distanceLength == 0) return; //exit if distance 0
84              double u = 60;
85              int x, y;
86              int a = 50;
87              int b = 400;
88
89
90              for (int i = 0; i < distanceLength; i++) //draw
91              {
92                  long normalized = (b - a) * (distance[i] - dimin) / (dimax - dimin) + a;
93                  x = centerX + (int)(normalized * Math.Sin(Math.PI * u / 180));
94                  y = centerY - (int)(normalized * Math.Cos(Math.PI * u / 180));
95
96                  mObject.DrawLine(greenPen, new Point(centerX, centerY), new Point(x, y));
97                  //MessageBox.Show("distance: " + distance[i], " normalized: " + normalized); //Just debug
98
99                  u += (int)(240 / distanceLength);
100                 if (u >= 300)
101                     u = 60;
102
103             }
104
105
106         }
107
108
109     }
```

14. ábra. drawMeasurments függvény

A 14. ábrán látható az a függvény ami kirajzolja a beolvasott értékeket. Minden egyes adatcsomag érkezésénél meghívja ezt a függvényt a PlotForm osztályban lévő adatfogadó függvény. A beérkező távolságokat elsősorban normalizálni kell, valamint egy pont köré kell kirajzolni. Ezt a feladatot a 92 - 94 sorban végzi el ez a függvény. A panelra rajzolás a 96. sorban történik meg. A DawLine függvény első paraméter a színt adja meg, a második a kezdőpontot, a harmadik pedig, hogy meddig hozza az egyenest.

A szoftver menüpontokat vezérlő kódok külön osztályba található. Ezen kívül még négy osztályt hoztam létre amelyeket a soros kommunikációra, az adatok dekodólására, valamint az mérések kirajzolására használok. Így a program könnyebben áttekinthető, fejleszthető és javítható. Ezt a 15. ábrán megtekinthetjük.



15. ábra. A szoftver felépítése

Mérések tárolása

A mérések tárolása fontos része a térkép kirajzolásához. Jelen esetben minden egyes kirajzolás után elmentem a beolvasott távolságokat egy CSV típusú formátumban. Azért eset a választás a CSV típusra mivel ez könnyen áttekinthető és sok platformon probléma nélkül megnyitható. Futás közben egy timer segítségével számoljuk a két mérés között eltelt időt, ez az érték kerül be első oszlopba, a következő két oszlopban a mérés kezdeti, valamint az utolsó mérési szöge kerül be, a maradék oszlopokba a mérési eredmények centiméterben. A CSV állomány a program főkönyvtárában található, a „_MySource” mappában.

A “ShowDataReaded” függvény minden egyes adatcsomag érkezésnél meghívásra kerül. Ha a MD parancsra jött válasz, akkor dekódoljuk a beérkező adatokat, elmentjük a CSV állományba, majd kirajzoljuk a drawMeasurments függvény segítségével. Ha VV parancsra jött válasz, ami azt jelenti, hogy a felhasználó most csatlakozott rá a szenzorra a soros porton, akkor a beérkező szenzor információkat feldaraboljuk és csakis azt jelenítjük meg amire szükségünk van.

```
private void ShowDataReaded(object sender, EventArgs e)
{
    string[] split_command = dataIn.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries); //split input data
    if (split_command.Length == 0) return;
    if (split_command[0].StartsWith("MD"))
    {
        Console.WriteLine("decode THIS");
        DecodeClass.decodeMD(dataIn, ref time_stamp, ref distances);
        SavePlotData(distances); //save to file
        DrawMe.drawMeasurments(ref distances);
        Console.WriteLine("DrawMeasurments -> MD command");
        return;
    }
    else if (split_command[0].StartsWith("VV"))
    {
        vendBox.Text = split_command[3].Remove(0, 5).Remove(split_command[3].Length - 7, 2); //delete first and last carater
        productBox.Text = split_command[5].Remove(0, 5).Remove(split_command[5].Length - 7, 2);
        serialBox.Text = split_command[6].Remove(0, 5).Remove(split_command[6].Length - 7, 2);
    }

}
```

16. ábra. Adatok mentése

Következtetések, jövőbeli tervek

A továbbiakban szeretném tanulmányozni az OpelGL típusú megjelenítést, hogy a jelenlegi ábrázolási módon javítsak. Valamint az elmentet mérések segítségével kitudjak rajzolni egy térképet. Szeretném ezt a szoftver Linux típusú operációs rendszerre is megvalósítani, hogy használni tudjam a Raspberry Pi 3 számítógépen. Szeretném

optimalizálni a számításigényt, hogy majd a későbbiekben minél kevesebb teljesítményfelvétellel üzemelhessen az autonóm robot.

Szakirodalmi hivatkozások

- https://people.inf.elte.hu/lorincz/NIPG_Theses/daroczi-diploma-slam.pdf
- https://www.hokuyo-aut.jp/
- https://regi.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0052_43_autonom_mobil_robotok/lecke3_lap1.scorml
- https://www.researchgate.net/publication/327570150_Simultaneous_Localization_and_Mapping_SLAM_using_RTAB-MAP/fulltext/5b9733de92851c78c4191815/Simultaneous-Localization-and-Mapping-SLAM-using-RTAB-MAP.pdf
- https://github.com/SteveMacenski/slam_toolbox