

Biztonságkritikus robotrepülőgép földi állomásának kialakítása

Önálló laboratórium zárójegyzőkönyv

2012/13 II. félév

Böjti Paszkál(BH0R0N)
III. évf, mérnök informatikus szakos hallgató
BSc Beágyazott információs rendszerek szakirány

Konzulens
Bartha Tamás

2013. május 20.

Tartalomjegyzék

1. Motiváció	2
2. Bevezetés	3
3. UAV	3
3.1. UCAV	3
3.2. Csoportosítás	3
3.3. Felhasználási területek	4
3.4. Hosszú repülés	4
4. Földi állomás	5
4.1. Hordozható	5

4.2. Komplex	5
4.3. Kompatibilitás	6
4.4. Megoldások	7
4.4.1. MIT GCS	7
4.4.2. Rodan	8
4.4.3. HappyKillmore	9
4.4.4. Arducopter	10
4.4.5. Viking	10
4.4.6. NI LabWindows/CVI	10
5. Repülőgép kialakítása	11
6. Saját implementáció	13
6.1. RS232	13
6.2. Sebesség	15
6.3. Repülési irány	16
6.4. GUI	17
7. Hibadetektálás	18
7.1. Beragadás	18
7.2. Nagy változás	18
7.3. Nagy eltérés a két adat között	19
8. .NET vs UNIX	19

1. Motiváció

Félév során lehetőségem volt a SZTAKI Irányítástechnikai Kutató Laboratóriumában egy UAV légi jármű fejlesztésében részt venni. Az itt kidolgozott szabályozó algoritmusok gyakorlatba való átültetésére egy pilóta nélküli járművet hoztak létre, mely a biztonságos üzemeltetés miatt, illetve az estelegesen előfordulható hibák ellen redundáns hardware elemekkel védekezik. A feladatom e repülő a földi állomásának a kialakítása volt, mely a redundánsan küldött rádiójelek feldolgozására és megfelelő megjelenítésre használálandó. Az összehasonlításból kiderül, eddig a földi állomásokra nem volt jellemző a redundancia. Kutatómunkám során összegyűjtöttem az UAV-k fejlődésének történetét, a piacra elérhető programokat, összehasonlítottam a megoldásokat, majd ezekből ötletet merítve elkezdtem a program megírását. Konzultációk során finomhangolásokkal a megrendelő igényeihez formáltam az elkepzelt megvalósítást.

2. Bevezetés

Napjainkban egyre nagyobb teret hódít a pilóta nélküli légi járművek alkalmazása. Az 1960-as években a hadszíntéren jelentek meg először, ahol megfigyelésre, felderítésre, olyan feladatokra használták, ahol kockázatos lett volna emberi életet veszélyeztetni. Az utóbbi években praktikussága, alacsony üzemeltetési költségei miatt más területeken is hasznosnak bizonyult ez a technológia, pl. geológia mintázatok kutatása, mely az emberi perspektívából nehezen észlelhető, tűzoltósági alakulatok koordinálása, otthoni hobby felhasználás.

3. UAV

[1] A pilóta nélküli légi jármű gondolata egészen a XX. század elejére nyúlik vissza, mikor az I. világháborúban egy olyan távirányítású repülőt alkottak, mely robbanószerrel a fedélzeten a célpontba csapódva okozott kárt. [2] Első nagyobb amerikai UAV projekt 1959-ben kezdődött, mikor aggódtak az ellenséges területre berepülő pilóták életéért. Később a vietnámi háborúban több mint 3000 küldetésben vett részt ilyen repülő és minden össze 554 veszett oda. A technológiai korlátok miatt a fő funkcionálisája video felvétel készítése egy meghatározott útvonalon (általában egyenes vonal, körökkel kiegészítve) repülve, majd a bázisra való visszaérkezés. A rádiótechnika fejlődése miatt egyre összetettebb feladatok elvégzésére lettek képesek. A nagyobb átviteli sebességek köszönhetően valós időben, monitoron keresztül kezelheti az operátor a távirányítású repülőt. A mai UAV-k több üzemmódot is támogatnak, egyik az előbb említett távirányítás, másik a fedélzeti intelligenciára hagyatkozó. Mind a hagyományos repülőiparban, mind ebben az érőben, szükséges és célszerű az emberi terhelés csökkentése, utasszállító gépek esetében is a robotpilóta elvégez minden olyan korrekciót, melyet azelőtt a pilóta folyamatos figyelésével, koncentrációjával lehetett elérni. A modern integrációnak köszönhetően, olyan fejlett feldolgozóegységgel dolgozhatjuk fel az adatokat, melyeknek nem jelentős a fogyasztása, nem foglalnak sok helyet. A szenzorokból érkező információkra, az algoritmusoknak köszönhetően úgy tud reagálni, hogy az nem veszélyezteti a repülő levegőben maradását. Ám hiába a fejlett hardware, a valóban automatikus üzemeltetés még mindig távoli cél, emberi beavatkozás mindenkor kelleni fog le-, felszállásoknál, illetve olyan helyzetekben melyre nincs előre felkészítve az intelligenciája.

3.1. UCAV

[3] Katonai felhasználásban elengedhetetlen a fedélzeti fegyverzet, így ezekre a járművekre is felszerelésre került. Az ilyen drónok az estek többségében operátori irányítás alatt állnak, így emberi felelőssége van a fegyver használata során.

3.2. Csoportosítás

Amerikai légierő ezeket a csoportokat tartja nyilván

- Kézi indítású: 600 m magasság, 2 km hatótávolság

- Közeli: 1500 m magasság, 10 km hatótávolság
- MALE(Medium Altitude,Long Endurance) 9 km magasság, 200 km hatótávolság
- HALE(High Altitude,Long Endurance) 9+ km magasság, 200+ km hatótávolság

3.3. Felhasználási területek

Célpont kiképzésre használandó, légi fenyegesként ellenséges célpont

Felderítő harctéren nagyobb látóteret biztosít

Támadó fegyverzettel ellátott, lég-föld, lég-lég rakétákkal csapások mérése

Szállító raktérrel ellátott, nehezen megközelíthető csapatok segítésére

Kísérleti későbbi UAV-k kifejlesztéséhez

Civil hétköznapi embereknek szórakozás céljából

Érzékelés különböző érzékelőkkel ellátva képes kémiai anyagok jelenlétéit érzékelni, infrakamerákkal és radarral éjjel is látható az operátor számára a környezet

Megfigyelés egyik legolcsóbb megfigyelési mód az UAV alkalmazása, kisebb üzemben tartási költsége van, mint egy repülőnek, pilóta bérénél. Pl. vadak vonulásának követése

Geológiai felfedezés mágneses érzékelőkkel a Föld mágnesesség változásának méreteivel következtetni lehet a föld alatt található ércekre.

Mentés természeti katasztrófa következtében bajbajutottak segítésére, mentőcsapatok megfelelő irányba küldésére használatos

Erdőtűz detektálás folyamatosan a megfigyelt terület felett repülve, infravörös kameralval a legkisebb tűz érzékelhető. Szél és egyéb meteorológiai adatokkal szolgáltatva a tűzoltók munkáját segíti

3.4. Hosszú repülés

Mivel az UAV-k nem rendelkeznek fedélzetű személyzettel, így a levegőben töltött idejük kizártlag az energiaforrásuk szűkössége határozza meg. Elektromos meghajtású gépek esetén, megfelelő méretű napelem alkalmazása esetén akár végtelenségig levegőben maradhatnak. A minél hosszabb fennmaradás előnyös lehet taktika bevetések során, ahol megspórolható az oda-visszaút megtételéből származó kiesés. Leghosszabb repült idő 2010. július. 9 – 23 között QinetiQ Zephyr Solar Electric érte el 336 óra 22 perccel (2 hét)

4. Földi állomás¹

Egy UAV vezetéséhez elengedhetetlen egy bázis, ahonnan a földi személyzet irányítja, monitorozhatja a repülést. Általában több funkciót lát el:

- Küldetés tervezés: útvonal meghatározása, illetve a célpontok kijelölése
- Adatok megjelenítése: megfelelő módon kijelezni a repülőgép állapotát, esetleges hibáit.

4.1. Hordozható

Egy egységbe építették az akkumulátort, kijelzőt, kezelőszerveket, hordozhatósága miatt ez az elterjedtebb a kézzel indítható UAV-k esetében.



1. ábra. Hordozható GC

4.2. Komplex

Általában közel a hadszíntérhez üzemeltetik a repülőket, így a személyzetet, illetve a felszereléseket páncélozott járművel szállítják, védik. Pl. az MQ-1 Predator GCS egy utánfutóban helyezkedik el, mely szünetmentes tápegységet biztosít a pilóta és segédszemélyzeti, adatelemző/rögzítő és radar munkaállomásoknak. A kommunikációt UHF

¹GC(S):Ground Control System

és VHF frekvencián bonyolítják közvetlen rálátás esetén, azon kívül műhold közbeiktatásával.



2. ábra. Utánfutós megoldás

4.3. Kompatibilitás

[4] Ahogy jöttek az újabb gépek, problémát okozott, hogy mindegyik földi állomása küllőnbözött egymástól, így 2008-ban szorgalmaztak egy univerzális GCS architektúrát, mely minden addigi és jövőbeli katonai UAV-vel ugyanúgy tud kommunikálni, adatokat szolgáltatni. Így született a Raytheon Common Ground Control System, melynek fejlesztéséhez videojátékokból merítettek ötleteket. Csökkenteni próbálták a kiképzési időt, ne kelljen több hónapos kurzusokon részt venni, hanem úgy, mint egy videojáték-nál, kisebb gyakorlás után már tudja az ember, mit hol kell keresni, illetve úgy érezze, hogy valóban ő vezeti a repülőt.



3. ábra. UGCS

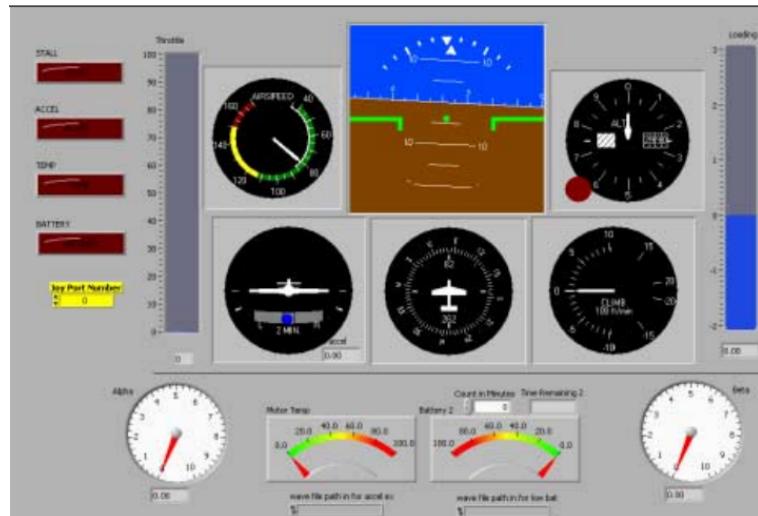
4.4. Megoldások

Számos megoldás született a földi állomás GUI²-jának kialakítására. Elsődleges követelmény, hogy az operátor mindenkor a legfontosabb információkat láthassa, ehhez szoftverergonómiaiak kell megtervezni a műszerek, adatok elrendezését. Az alábbiakban összehasonlításra kerülnek a megjelenítési felületek.

4.4.1. MIT GCS

[5]Elsődleges megfontolás az volt, hogy a műszereket az értékeket linerisan jelenítsék meg, mivel az avionikában ez a kérdés már felmerült, így a valós műszerek alapján a lineárisat választották. Másodlagos kérdés a műszerek formája, itt megvalósítható lenne a csupán digitális értékek kiírása, de szintén a repülésből merítve a kör alakú egységeket választottak

²Graphics User Interface, grafikus megjelenítés



4. ábra. Alap összeállítás

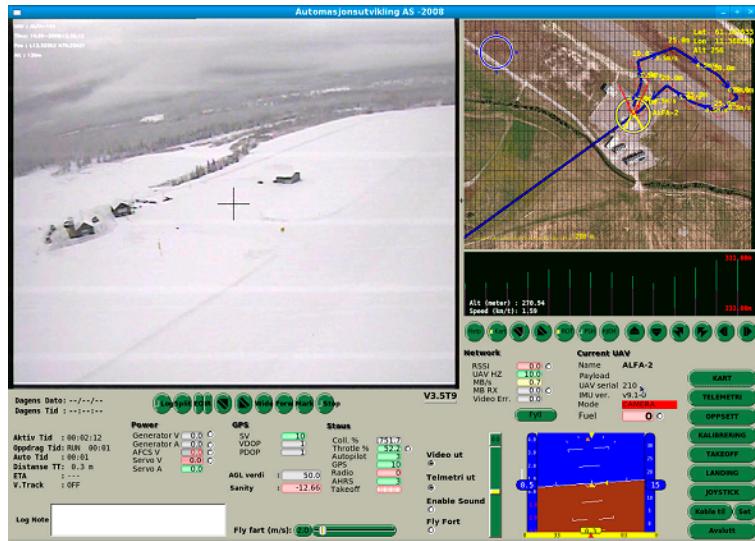
4.4.2. Rodan

[6] Földi, légi, vízi járműveket is támogat, TCP/IP-n keresztül kommunikál, így videó átküldésére is van sávszélesség. SIL³ és HIL⁴ támogatás.

A GUI itt már összetettebb, a kamera képe foglalja el a képernyő nagyobbik részét, mellette egy térképen láthatjuk a jármű eddigi útvonalát. Egyetlen vizuális műszer a műhorizont, a többi érték csak számadattal van jelezve.

³Software In the Loop

⁴Hardware In the Loop



5. ábra. Rodan

4.4.3. HappyKillmore

[7] Ez a GUI az ArduPlane nevű nyílt forráskódú, háziépítésű UAV-hoz készült. Itt is a térkép nézet dominál, oldalt mozgathatjuk számunkra megfelelő elrendezésbe a műszereket. Több különböző oldal közül választhatunk, ha pl. a bejövő adatokra vagyunk kiváncsiak vagy a soros port port számát szeretnénk beállítani. Lehetőségünk van exportálni is az adatokat.



6. ábra. HK

4.4.4. Arducopter

Az előzőhöz hasonlóan ez is az ArduCopterhez készült. Itt lehetőségünk van térképen előre kijelölni, milyen útvonalat járjon be felszállás után.



7. ábra. Arducopter

4.4.5. Viking

[8] Itt egyértelműen a térkép feletti pozícióban van a hangsúly, jobb oldalt néhány műszer, illetve a fontosabb adatok számokkal



8. ábra. Viking

4.4.6. NI LabWindows/CVI

Az előző Ground control forráskódját átnézve, annak továbbfejlesztése körülményes lenne, így a C# nyelv változatos API-jainak köszönhetően egy új GUI megírása tűnik a

legoptimálisabb megoldásnak. Soros port kezelésére van beépített [10]system.io.ports.serialport osztály, mely megkönnyíti az adatok befogadását.

5. Repülőgép kialakítása

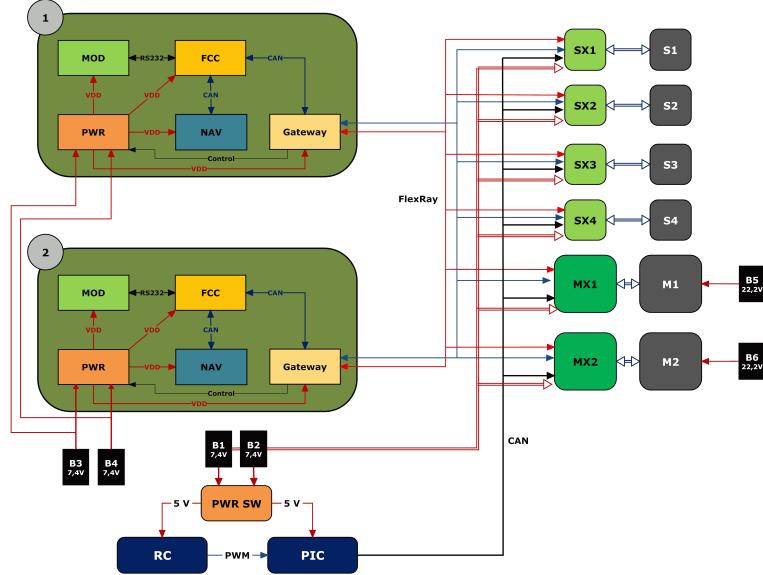
A SZTAKI-nak volt már egy hasonló projektje, mely egy kisebb, boltban beszerezhető modellrepülőre épített saját irányítóegységgel repült. A számos tesztnek köszönhetően felmerült az igény egy olyan, teljesen saját fejlesztésű repülő kialakításra, mely lehetőséget biztosít biztonságkritikus kialakításra, szemben a gyári alkatrészekkel, amelyek semmilyen visszajelzést nem adnak saját állapotukról, mely a szabályzásban jelentős fontossággal bír.



9. ábra.

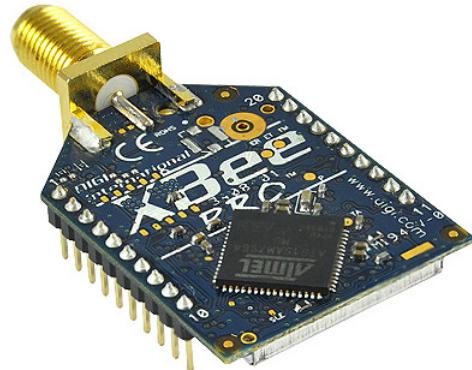
Így az új repülőgépen redundánsan fog szerepelni:

- motor
- akkumulátor
- központi számítógép



10. ábra.

Látható, hogy központi számítógép kettőzött, párhuzamosan működnek, irányítják a servo motorokat. A szenzorokból érkező jelek egy Gateway-en keresztül az FCC-be jutnak, ahol a feldolgozás utána a szabályozó algoritmusok kiadják a megfelelő utasítást a kormányszerveknek. Ilyen utasítás lehet pl. a csűrőlapok adott fokban történő elmozdítása, motor fordulatszámának változtatása. A motorok, úgy mint a központi egység, külön tápellátást kapnak, ezzel is csökkentve a végzetes meghibásodás esélyét. Jelen esetben egyik legfontosabb elem a kommunikációért felelős modem, mely soros porton keresztül kapcsolódik az FCC-hez. A vevő oldalon hasonló modem, szintén soros porton küldi a földi állomásnak a vett jelet, a köztük lévő vezeték nélküli kommunikáció saját szabványa a cégnak. A választás XBee-PRO 868 típusú modemre esett, mely alacsony fogyasztása és nagy hatótávolsága miatt ideális egy ilyen környezetbe.

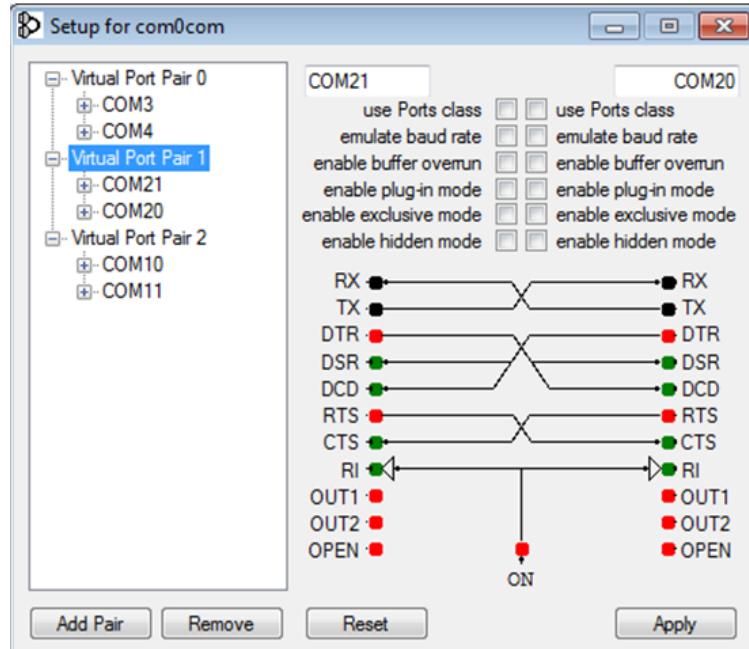


11. ábra.

6. Saját implementáció

6.1. RS232

A modemből érkező adatokat soros porton keresztül fogadja a program, a tesztkörnyezet felállításához HIL adatok szolgáltak. A küldött log fájlokat egy programmal beolvassom és egy [11]null-modem segítségével sorosporton keresztül küldöm a megfelelő portra.



12. ábra.

Beállítottam 2 párt, COM20-COM21 és COM10-COM11 között, a páron keresztül a páratlanon fogadom az üzeneteket.

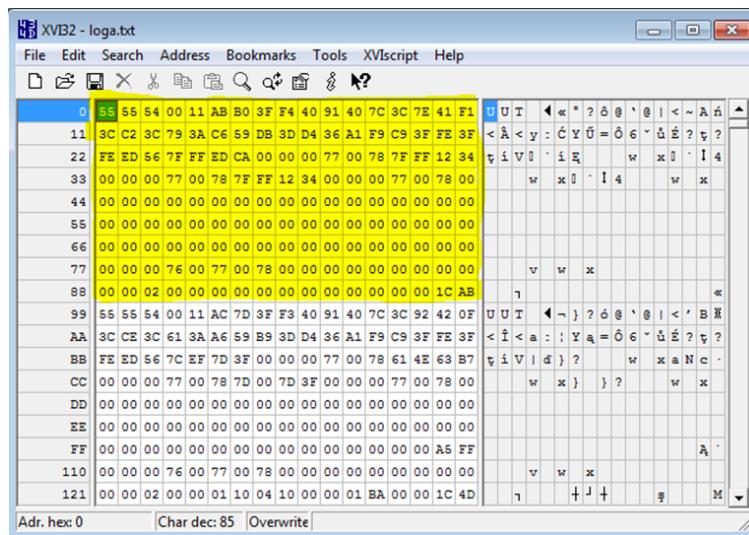
```

Serial ports:
COM1
COM21
COM3
COM10
COM4
COM20
COM11
com10.          sending B0*153 bytes
1*153 bytes
2*153 bytes
3*153 bytes
4*153 bytes
5*153 bytes
6*153 bytes
7*153 bytes
8*153 bytes
9*153 bytes
10*153 bytes
11*153 bytes
12*153 bytes
13*153 bytes
14*153 bytes
15*153 bytes
16*153 bytes

```

13. ábra.

153 bájtos egy csomag, melyet egy UUT 3 bájtos fejléc és egy 2 bájtos checksum zár. A checksum a hasznos bájtok 16 bitre csonkolt összege. minden fogadott csomagnál, a feldolgozás előtt kiszámolom az összeget és ellenőrzöm, az egyezést, a rossz csomagok egyelőre eldobásra kerülnek.



14. ábra.

Fogadó oldalon a két sorosport aszinkron ír 1-1 byte tömböt, melyből egy dekódoló függvénytellyel nyerjük ki a sebesség, pozíció, irány, stb. adatokat.

```
public double[] Decode(byte[] array)
```

Ebben a függvényben ellenőrzöm, a checksum-ot, illetve a kezdő UUT bájt hármast. Mivel bájtosával lehet feldolgozni az adatokat, így pl. a 4 bájtos időbélyeget 4 db egymás után jövő bájtból kell összerakni:

```
uint ido = (uint)array[3]<<24 | (uint)array[4]<<16 |
(uint)array[5]<<8 | (uint)array[6] ;
```

Ugyanígy folytatódik az adatok feldolgozása, az előre megadott protokoll szerint.

bájt index	leírás	típus	skálázás	offset
1	start	char(fix 'U')		
2	start	char(fix 'U')		
3	start	char(fix 'T')		
4	idő 1/4	unsigned int	10000	0
5	idő 2/4			
6	idő 3/4			
7	idő 4/4			
...				
26	északi irány 1/2	unsigned short	0x7FFF/400	200
28	keleti irány 1/2	unsigned short	0x7FFF/400	200
30	lefelé irány 1/2	unsigned short	0x7FFF/400	200
...				

Mivel a változások Hamming-távolsága⁵ kicsi lenne, az eredeti számábrázoláson, így skálázással és offset képzéssel megnöveljük. A sebesség adatoknál a visszakódolás: $eredeti = (nyersadat/skalazas) - offset$ képlettel oldható meg.

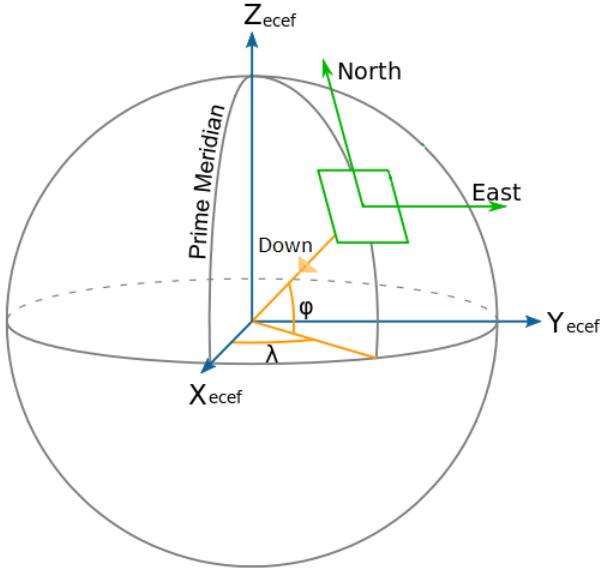
6.2. Sebesség

A sebesség NED⁶ koordinátarendszerben van megadva, mely a repülő középpontjából indul. Mivel kis magasságban repül a repülő, így síknak közelíthetjük a Föld felületét, ez számítások szempontjából előnyös, mivel könnyebb vele dolgozni. A sebesség számítási módja: $\sqrt{V_E^2 + V_N^2}$

Emelkedés: $-V_D$

⁵Bináris számok XOR képzésével kapott 1-esek száma

⁶Nort East Down, Local Tangent Plane, helyi koordináta rendszer



15. ábra. Koordináta rendszer

6.3. Repülési irány

A fedélzeten lévő mágneses iránytű irányszöge és a GPS-ből érkező sebesség vektor a szél következtében eltérő lehet. E kettő adatból későbbiekben a szél iránya is meghatározható. Az valós irány kiszámításához a 4 negyedsíkot külön kellett választani:

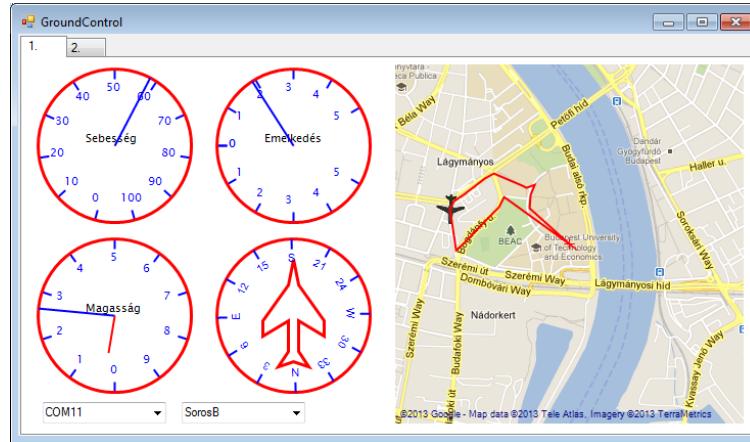
```

if (Ecomp > 0 && Ncomp > 0)
{
heading = Math.Atan(Ecomp * Ncomp);
}
else if (Ecomp > 0 && Ncomp < 0)
{
heading = Math.Atan(Ecomp * Math.Abs(Ncomp)) + 90;
}
else if (Ecomp < 0 && Ncomp < 0)
{
heading = Math.Atan(Math.Abs(Ecomp) * Math.Abs(Ncomp)) + 180;
}
else if (Ecomp < 0 && Ncomp > 0)
{
heading = Math.Atan(Math.Abs(Ecomp) * Ncomp) + 270;
}

```

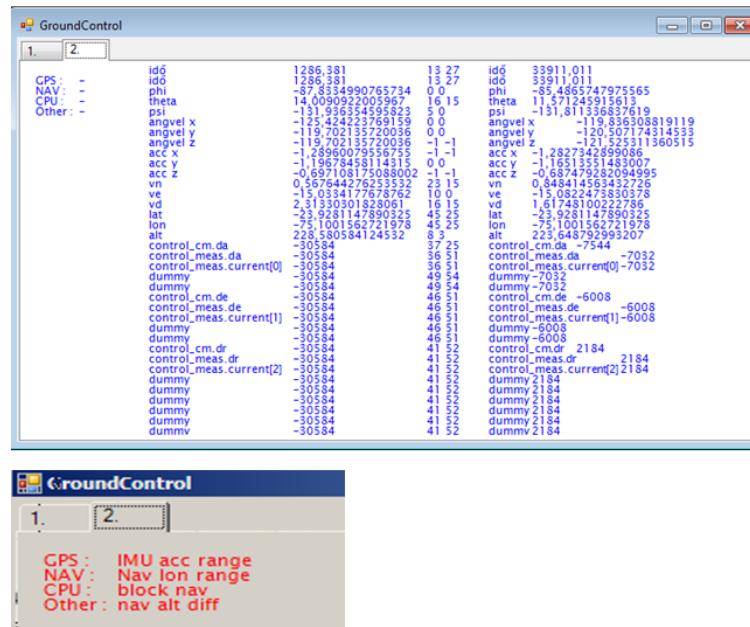
6.4. GUI

A grafikus felület kialakítása során figyelembe kell venni, hogy első ránézésre a legfontosabb adatok látszódjanak. Kettő fül közül az első oldalon a legfontosabb műszerek találhatóak, jobb oldalon egy Google Maps térkép. A térkép egy lokális cache-ből tölti be az előre letöltött térképszelvényeket, így a terepen lehetőség van offline módon is használni ezt a funkciót.



16. ábra.

A második fülön telemetria adatokat figyelhetünk meg, mely jelenleg tesztelés céljából az összes küldött adatot megjeleníti.



17. ábra.

Bal oldalt, a fedélzeti hibát, hibakódja alapján jelzi ki. Első oszlop a küldött telemetria adat neve található, mellette az adat értéke, következő az A és a B adat hibaszámlálója. Jelenleg a szimuláció okozta hibák miatt nagy az eltérés az adatok között, így a későbbi finomhangolás után, a magasságban („alt”) látható 5 m különbségre folyamatosan növekedne a számlálója. Ha egy küszöbérteket elér, akkor piros színnel, illetve feltűnő hibaüzenettel jelezzük a problémát.

7. Hibadetektálás

7.1. Beragadás

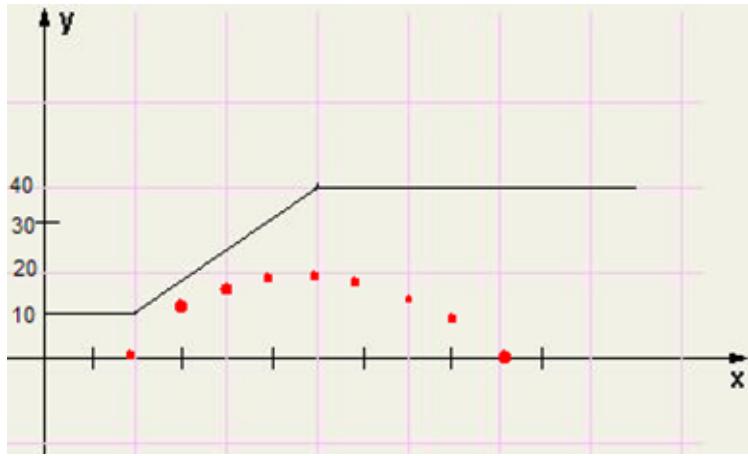
Ez a fajta hiba, valamelyik egység meghibásodását jelzi. Egy ϵ tartományon belül változik a jel egy hibaszámlálót növelünk 5 egységgel.



18. ábra.

7.2. Nagy változás

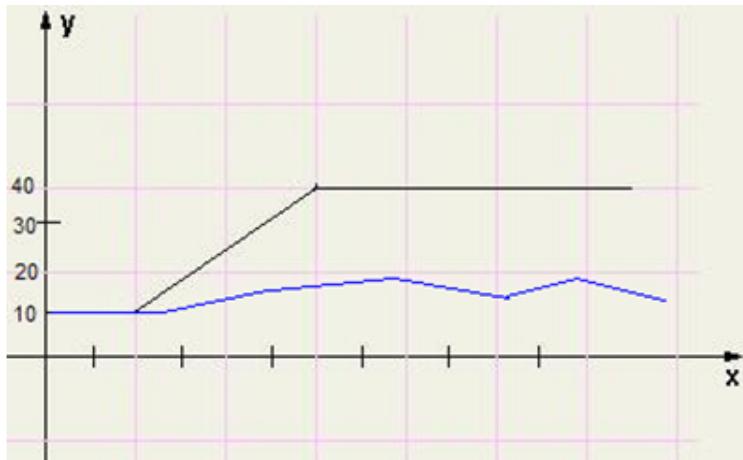
Az eddigi minta lapján, ha 10%-nál nagyobb eltérés található, a számlálója növekszik 2 egységgel. Piros pont jelzi 19 .ábrán, a változás mértékét.



19. ábra.

7.3. Nagy eltérés a két adat között

Ha a két adatsorozat között egy ϵ_2 különbség mutatkozik, növeljük a számlálót, amúgy csökkentjük. Így a gyors változásból adódó hibadetektálást kompenzáljuk, ha A és B együtt változik.



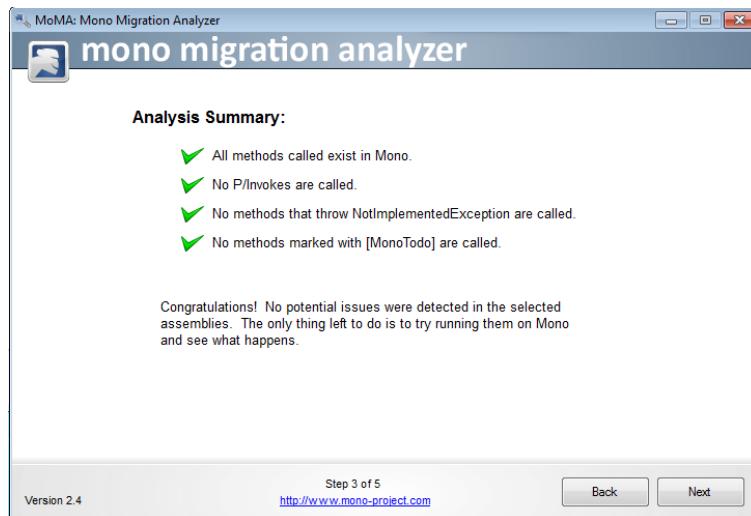
20. ábra.

8. .NET vs UNIX

A .NET alapvetően Microsoft technológia, de pl. a C# könnyű fejleszthetősége miatt lehetőségünk van .NET-ben írt alkalmazások UNIX rendszeren való futtatására is. Erre szolgál segítségünkre a [9]Mono több platformos rendszer, mely megvalósítja a

.NET keretrendszerét és a CLR⁷-t, így ha úgy adódik, hogy a Ground Control állomás platformot vált, az eddig megírt program „portolható” másik operációs rendszerre.

Tesztelhetjük is programjainkat, hogy kompatibilis-e a Monoval:



21. ábra. Mono

Hivatkozások

- [1] http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle, 2013. május 4, 10:00
- [2] http://en.wikipedia.org/wiki/Hewitt-Sperry_Automatic_Airplane, 2013. május 4, 10:00
- [3] http://en.wikipedia.org/wiki/Unmanned_combat_air_vehicle, 2013. május 4, 10:00
- [4] <http://www.defenseindustrydaily.com/uav-ground-control-solutions-06175/>, 2013. május 4, 10:00
- [5] http://api.ning.com/files/ga5AVWy8xTu8cx9rHdzJ73Epc*dzb0uy*UPe0O8wFxogMF6WNRYxzkg2iBk16kNFpYTZe80NgM8kbXOiyxrEwUXIt/GroundControlStation.pdf, 2013. március 19., 15:00
- [6] <http://www.rodiangroup.com/uav-ground-control-station.html>, 2013. március 19., 15:00
- [7] <https://code.google.com/p/ardupilot-mega/wiki/HappyKillmore>, 2013. március 19., 16:00
- [8] http://www.vikingaero.com/uav_ground_control_station.html, 2013. március 19., 16:00

⁷Common Language Runtime

- [9] <http://www.mono-project.com>, 2013. március 19., 17:00
- [10] <http://msdn.microsoft.com/en-us/library/system.io.ports.serialport.aspx>,
2013. március 20, 10:00
- [11] <http://com0com.sourceforge.net/>, 2013. május 4, 10:00