



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Méréstechnika és Információs Rendszerek Tanszék

# Megbízható kommunikációs kapcsolattal rendelkező földi irányító állomás fejlesztése UAV-hoz

SZAKDOLGOZAT

*Készítette*  
Böjti Paszkál

*Konzulensek*  
Vörös András, Dr. Bartha Tamás

2013. november 25.

# Tartalomjegyzék

<b>Kivonat</b>	<b>4</b>
<b>Abstract</b>	<b>5</b>
<b>Bevezető</b>	<b>6</b>
<b>1. Előzmények</b>	<b>8</b>
1.1. Motiváció . . . . .	8
1.2. Repülőgép felépítése . . . . .	8
1.3. Hibatűrés . . . . .	9
1.4. Vezeték nélküli modem . . . . .	10
1.5. Földi állomások . . . . .	10
1.5.1. ArduPilot . . . . .	11
1.5.2. Paparazzi . . . . .	14
1.5.3. MicroPilot Horizon . . . . .	15
1.5.4. OpenPilot . . . . .	17
1.5.5. QGroundControl . . . . .	17
1.5.6. HappyKillmore . . . . .	17
1.5.7. Viking . . . . .	17
<b>2. Tervezés</b>	<b>19</b>
2.1. Kommunikáció . . . . .	19
2.2. Adatok fogadása . . . . .	19
2.2.1. Protokoll . . . . .	19
2.3. Adatok küldése . . . . .	20
2.3.1. Protokoll . . . . .	20
2.4. Grafikus felület . . . . .	21
2.4.1. Főképernyő . . . . .	21
2.4.2. Tervezés képernyő . . . . .	22
2.4.3. Diagnosztikai képernyő . . . . .	22
<b>3. Megvalósítás</b>	<b>23</b>
3.1. .NET . . . . .	23
3.2. Főképernyő . . . . .	23

3.3. Tervezés képernyő	24
3.4. Diagnosztikai képernyő	24
3.5. Soros port	24
<b>4. Értékelés</b>	<b>27</b>
<b>5. Összefoglalás</b>	<b>28</b>
<b>Függelék</b>	<b>30</b>
F.1. Függelék1	30

## HALLGATÓI NYILATKOZAT

Alulírott *Böjti Paszkál*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (Böjti Paszkál, Megbízható kommunikációs kapcsolattal rendelkező földi irányító állomás fejlesztése UAV-hoz, angol és magyar nyelvű tartalmi kivonat, 2013, Vörös András, Dr. Bartha Tamás) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hállózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedélyteljes titkosított diplomatervezet esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2013. november 25.

---

*Böjti Paszkál*  
hallgató

# Kivonat

Napjainkban egyre nagyobb teret hódít a pilóta nélküli légi járművek alkalmazása. Az 1960-as években a hadszíntéren jelentek meg először, ahol megfigyelésre, felderítésre és 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.

Az MTA-SZTAKI Rendszer és Irányításelméleti Kutatólaboratóriumában kidolgozott szabályozó algoritmusok gyakorlatba való átültetésére egy pilóta nélküli járművet hoztak létre, mely a biztonságos üzemeltetés mellett, illetve az esetlegesen előfordulható hibák ellen redundáns hardware elemekkel védekezik. Szakdolgozatom keretében ennek a repülőnek a földi állomását dolgoztam ki, mely a redundánsan küldött rádiójelek feldolgozására és megfelelő megjelenítésre használandó. A földi személyzet mozgó térkép alapú vizualizáció láthatja az aktív útvonalpontokat és a gép útvonalát, a diagnosztikai adatokat és esetleges hibákat egy másik nézetben áttekintheti. Lehetőség nyílik repülési terv meghatározására és feltöltésére a repülőre, melynek fordulópontjait követi.

# **Abstract**

Nowadays..

# Bevezető

[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. Később a vietnámi háborúban több mint 3000 küldetésben vett részt ilyen repülő és minden összes 554 veszett oda. A technológiai korlátok miatt a fő funkcionálisága videó felvétel készítése egy meghatározott útvonalon (általában egyenes vonal, körökkel kiegészítve) repülve, majd a bázisra való visszaérkezés. A rádiotechnika 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 csökkenése, 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 képes 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 fog olyan helyzetekben melyre nincs előre felkészítve az intelligenciája. Ilyen esetekben létfontosságú, hogy az operátor lássa a gép aktuális pozíóját, diagnosztikai adatait.

Feladatom egy olyan grafikus felhasználói felülettel ellátott földi irányító állomás kifejlesztése egy biztonságkritikus robotrepülőhöz, ami képes redundáns kommunikációs csatornán küldött adatok kezelésére és megjelenítésére. Ehhez szükséges megoldanom a kapcsolatot biztosító modem jeleinek vételét. Mivel hibatűrő kialakítása révén redundánsan szerelt, így a párhuzamos csatornákon érkező adatok egymástól való eltérésének a kijelzése is megvalósítandó. Az eltérésnek több oka is lehetséges, előfordulhat, hogy az egyik meghibásodásából következően nem küld semmilyen értéket vagy amit küld az teljesen hibás. Továbbá, ez a program a földön tartózkodó személyzet kiszolgálására készül, így ami a legfontosabb, hogy ők a repülővel kapcsolatos információkat könnyen értelmezhessék. Így szükséges megoldani, hogy a kialakítandó felület felhasználóbarát legyen, ehhez több nézeti oldal szükséges:

- Egy áttekintő képernyő, mely a legfontosabb adatokat jeleníti meg a repülővel kapcsolatban ( pozíció, sebesség, irány)

- Egy tervező modul, amin a lerepülendő útvonalhoz tartozó fordulópontok kijelölése lehetséges
- Egy diagnosztikai nézet, melyen az alacsonyabb prioritású adatok tekinthetők át

# 1. fejezet

## Előzmények

### 1.1. Motiváció

Az MTA SZTAKI Rendszer és Irányításelméleti Kutatólaboratóriumában folyó kutatások eredményének demonstrálására szükség volt egy kézzelfogható eszköz megalkotására. A szabályozó algoritmusok működésének bemutatásának az egyik leglátványosabb módja egy repülőgép irányítása. Egy stabil repülési tulajdonságokkal bíró repülő levegőben tartása sem triviális, szükség van a kormányszervek harmonikus mozgatására, a tolóerő szabályzására. Ezen túlmenően ha feladatakkal látjuk el, pl. fordulópontokat követve feltérképezni az alatta lévő területet, már számolni kell a széllel, mely eltérítheti az útvonaláról, felszálló légáramlatokkal, melyek ellen gyors reagálással kell válaszolnia. Mivel egy teljesen felszerelt repülő összeállítása, felprogramozása nagy szakértelmet, sok időt, energiát igényel és nem utolsó sorban anyagi ráfordítást, egy esetleges meghibásodás jelentős kárt okozna. Ezen okok miatt felmerült az igény a repülő megbízhatóságának növelésére, így a most folyamatban lévő „Nagy megbízhatóságú pilóta nélküli légi jármű projekt” keretében egy olyan avionikai rendszer fejlesztése is folyik, amelyben cél minden egyes repülőgép alrendszer meghibásodásának diagnosztizálása, a diagnosztikai információkat felhasználva a repülőgép átkonfigurálása.

### 1.2. Repülőgép felépítése

A jelenlegi repülő egy saját építésű 3.2 m feszttávolságú modell, melybe diagnosztikai és hibadetekciós célokra egyedi tervezésű komponensek kerültek. A kommerciális célokra szánt szervo motorok nem szolgálnak elég információval a kitérésükről, fogyasztásukról, ezért ezek méréséről gondoskodni kellet.



1.1. ábra

### 1.3. Hibatűrés

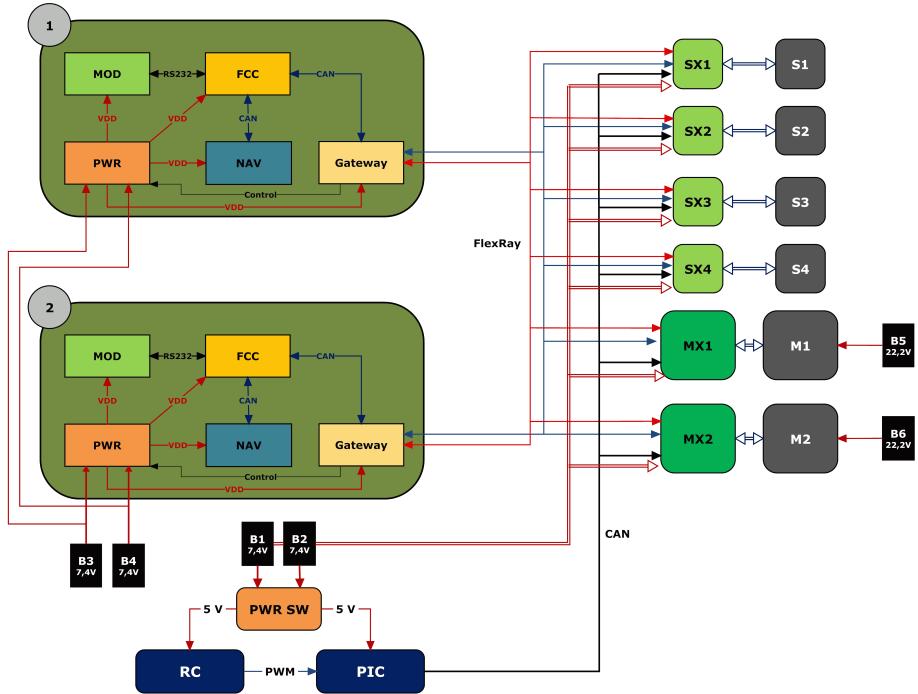
Elsődleges szempont, hogy egy komponens meghibásodása ne okozza a gép vesztét, tehát a rendszerben ne maradjon SPOF<sup>1</sup>. Ezt redundanciával érhetjük el, mely során a repülőgépen duplikáljuk azokat az elemeket, melyek nélkülözhetetlenek a levegőben maradásához:

- motor
- kormányfelületek és ezeket mozgató motorok
- tápellátás
- központi számítógép

A repülő méretéből adódóan térbeli szeparációval nem lehetséges a megbízhatóságot növelni, mint pl. harci repülőgépek fedélzeti számítógépeinél, melyek a találat kockázata miatt elszórva, akár 4–5x-ve vannak.

1.2. ábrán látható a kialakított architektúra. A szenzorokból érkező jelek a duplázott központi feldolgozó egységekbe jutnak, ahol egy–egy Gateway felelős a kommunikáció lebonyolításában. A fogadott jelek feldolgozása<sup>2</sup> után 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. A dolgozat szempontjából 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 gyártónak. A választás [5]XBee-PRO 868 típusú modemre esett, mely alacsony fogyasztása és nagy hatótávolsága miatt ideális egy ilyen környezetbe.

<sup>1</sup>Single Point Of Failure, olyan meghibásodás, mely ha bekövetkezik, az egész rendszer leállásához vezet  
<sup>2</sup>FCC



1.2. ábra

#### 1.4. Vezeték nélküli modem

A kiválasztott modem kétféleképpen képes kommunikálni:

- API csomagküldés
- AT transzparens

**API** módban egy eszköz több eszköztől tud csomagokat venni, ha egy csomag megérkezett a küldőtől a fogadóig, egy ACK<sup>3</sup> üzenettel válaszol, ha ezt nem kapja meg, újraküldi. Egy csomag többek között tartalmazza a küldő és a fogadó címét, lehetőség van broadcast üzenetek küldésére is, melyet minden eszköz megkap. Továbbá az adatintegritás céljából checksum mezőjében összesítve van egy csomag adata.

**AT** mód nem más mint egy vezeték nélküli kapcsolat 2 sorosport közt. Nem csinál mást, mint a soros portján bejövő adatokat egy XBee szabvány szerint rádiójelekké alakítja, melyet a virtuális kapcsolat végpontja fogad és visszaalakítja soros portra. Ez pont-pont kommunikációra hivatott.

<sup>3</sup>Acknowledgement, nyugtázó üzenet



1.3. ábra

## 1.5. Földi állomások

Egy UAV vezetéséhez elengedhetetlen egy bázis, ahonnan a földi személyzet irányítja, monitorozhatja a repülést. Általában több [3] funkciót lát el:

- Küldetés tervezés: útvonal meghatározása, illetve a célpontok kijelölése
- Küldetés végrehajtás: távirányítással vezetve az operátor végrehajtja a feladatot
- Adatok megjelenítése: megfelelő módon kijelezni a repülőgép állapotát, esetleges hibáit

Számos megoldás született földi állomások GUI<sup>4</sup>-jainak kialakítására. Elsődleges követelmény, hogy az operátor mindenkor a legfontosabb információkat láthassa, ehhez szoftverer-gonómiaiag kell megtervezni a műszerek. [4] Kísérleteket folytattak, milyen elrendezésben, hány képernyőn érdemes megjeleníteni az adatokat, úgy, hogy az még ne terhelje túl az operátort. Bebizonyosodott, hogy érdemes több érzékszeren keresztül jelzéseket adni, így pl. nagy prioritású eseménynél a figyelmeztető ablak megjelenését hanghatás is kíséri. Alábbiakban összehasonlításra kerülnek a piacra lévő megjelenítési felületek, megoldások.

### 1.5.1. ArduPilot

Az [7] Ardupilot projekt létrejöttének oka, hogy otthoni körülmények között, nem ipari alkatrészekből bárki összeállíthasson egy autonóm járművet, mely az előre betáplált utasításokat végrehajtja. Központi eleme az Arduino cég által készített ATMEL mikroprocesszorra épített platform, mely felhasználóbarátabbá teszi a mikroprocesszor programozását. Magas szintű utasításokkal segíti az eszközzel való barátkozást, nincs szükség assembly szintű tudásra. Erre a platformra hozták létre az ArduPilot programot, képes vezérelni többféle autonóm járművet:

- ArduPlane néven futó változat: robotrepülőgép
- ArduCopter: 1, 3, 4, 6, 8, propelleres helikopter
- Ardurover: 4 kerekű autó

---

<sup>4</sup>Graphics User Interface, grafikus megjelenítés

Sikerességének fő oka, a nyílt forráskód, Arduino panel alacsony ára ( 5 ezer Ft) és a projekt mögött álló lelkes közössége.

Ennek a közösségnak köszönhetően született a Mission Planner nevű szoftver, mely teljes körű támogatást nyújt a járművekkel kapcsolatban.

## Főképernyő

Több nézet segítségével könnyen átlátható a funkcionálitása, repülés szempontjából a főképernyőn a legfontosabb adatok találhatóak. A repülőgép orientációját, sebességét, irányát egy műhorizonton láthatja a felhasználó, ez vizuálisan szemlélteti, hogy hány fokos bedöntéssel repül, milyen állásszöggel emelkedik. Alatta lévő területen előre beállított adatokat jelenít meg, pl. GPS magasság, GPS sebesség, szélirány( valós mágneses irány és a sebesség vektor különbségéből számítható). A legnagyobb területet a térkép foglalja el, melyen az aktuális irány, lerepült útvonal, fordulópontok láthatóak. Megfigyelhető, hogy ez kapja arányaiban a legnagyobb területet.



1.4. ábra

Ez a nézet akkor jöhét jól, mikor olyan meghibásodás történik, melyre nincs felkészítve az irányítóegység és szükséges lehet a kézi üzemmódra váltás. Előfordulhat ilyen esetben, hogy nincs vizuális rálátás az operátor és a repülőgép között, ekkor csak az itt látható műszerek és térkép alapján szükséges irányítania. Szerencsére ez az avionikában már bizonyított, hogy műszerek segítségével, „vakon” is lehetséges repülni, itt azonban ez az eset csak pár percig szükséges.

## Tervező és útvonalfeltöltő

Másik nézet lehetőséget biztosít útvonalpontok kijelölésére, az útvonalpontokhoz egy listából kiválasztható, hogy az adott pont start-, forduló-, végpont és egyéb lehetőségek. Az

útvonalpontok pozícióját egérrel módosíthatjuk, törölhetjük. Bal felső sarokban a kijelölt tervnek hosszát láthatjuk, ez segítséget nyújt, nehogy túlhaladjuk a rádiókapcsolat és a repülő hatósugarát. Az elkészített tervet feltölthetjük a csatlakoztatott eszközre.

Waypoints													
	WP Radius	Loiter Radius	Default Alt	<input type="checkbox"/> Absolute Alt	<input checked="" type="checkbox"/> RTL@def Alt	<input type="checkbox"/> Verify Height	Add Below						
1	TAKEOFF	▼	0	0	0	0	Lat	Long	Alt	Delete	Up	Down	Grad
2	WAYPOINT	▼	0	0	0	0	4,5654736	63,2812500	100	X	Up	Down	0
3	RETURN_TO_LAUNCH	▼	0	0	0	0	4,5654736	63,2812500	100	X	Up	Down	nem szám
4	LAND	▼	0	0	0	0	-25,4829512	3,1640625	100	X	Up	Down	0

1.5. ábra

## Beállítások

Itt lehetőség van a nyelv és mértékegységek kiválasztására, naplázás sűrűségének meghatározására. Egy másik oldalon az aktuális járművet lehet beállítani, mivel más adatok fontosak egy földi és egy légi jármű esetében, továbbá a kommunikációs protokoll is változó. Nem külön oldalon, hanem minden látható a csatlakozás gomb, melynél a soros port és a jelsebesség beállítása után a csatlakozás gombbal csatlakozik a program a portra.



1.6. ábra

## Terminál

Ezen az oldalon az USB-n csatlakoztatott eszközt lehet parancssoron konfigurálni. Az eszköz saját LOG fájljait letölteni.

## Egyéb képernyők

Lehetőség van még az elmentett log fájlok szimulálására, van egy súgó oldal a program használatával kapcsolatban, illetve egy támogatói gomb, mely átirányít egy PayPal oldalra, ahol különböző összeggel támogathatjuk a fejlesztést.

## Összegzés

Felhasználói szempontból barátságos felületet biztosít a különböző nézetekkel, gombok átgondolt elhelyezével. Nem a program hibája, de nincs felkészítve párhuzamos csatornák kezelésére, ez az ArduPilot projekt egyszerűségének köszönhető, mivel nem használ redundanciát. Mivel a repülő, melyhez a programot készítem, nagy megbízhatóságú, így szükséges megoldani a 2 port kezelését és az azokon érkező adatok feldolgozását. Jó ötlet a csatlakozás gomb minden látható elhelyezése, a főképernyón térképen ábrázolni a repülő helyzetét, illetve a műhorizont. Azonban hibadiagnosztikai kijelzés nincs megoldva, melynek szintén fontos a megvalósítása.

### 1.5.2. Paparazzi

[8] Hasonlóan mint az ArduPilot, ez a projekt is a könnyen, otthon összeszerelhető repülő megépítése jegyében született. Nem csak egy nyílt forráskódú robotpilótát ad, hanem komplett „csináld magad” készletet, melyben feszültségszabályzótól kezdve a GPS vevőig minden megkaphatunk. Továbbá a hozzá készült földi állomáshoz antennát, modemét és programot is nyújt.

A program a következő funkciókkal rendelkezik:

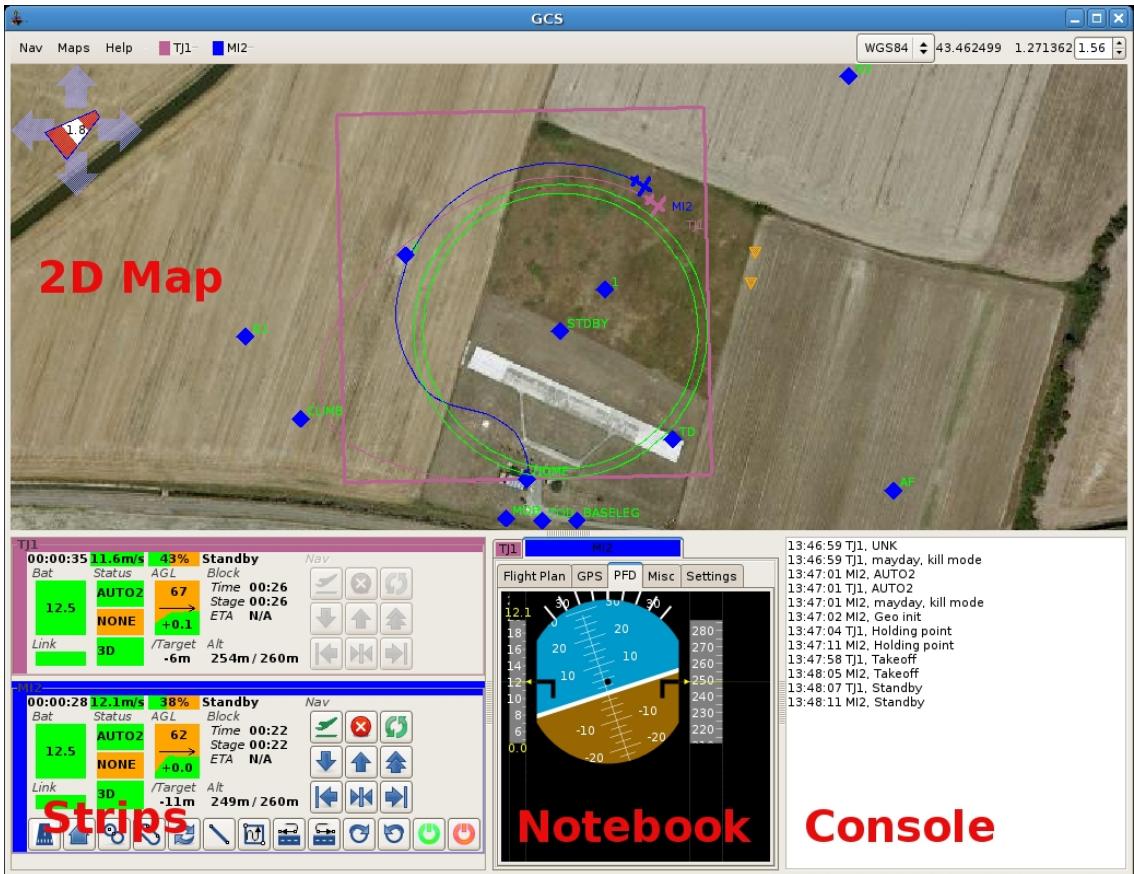
- több platform támogatása (fix- és forgószárny)
- több jármű egyidejű kezelése
- Google Maps, OpenStreetMaps, Microsoft Maps térképes megjelenítés
- küldetés tervezés
- mozgatható iránypontok
- valósidéjű fedélzeti paraméter állítás
- gyorsbillentyű
- hangjelzés
- teljesen átszabható grafikus felület

## Főképernyő

1.7. ábrán látható képernyő nagy százalékát a térkép foglalja el, melyen a lerepült útvonal és a fordulópontok láthatóak. A repülő aktuális helyzete mellett a neve, sebessége és repülési magassága is kijelzve van. Bal oldalt egy információs sávban a következő fontosabb adatok láthatóak: akkumulátor töltöttsége, sebesség, tolóerő, magasság, illetve utasítások: fel-, leszállás, megfigyelés indítása. Jobb felső sarokban a kurzor térképen lévő koordinátája kerül megjelenítésre.

A térképet a kék nyilakra való kattintással és a billentyűzet nyilainak lenyomásával lehet mozgatni, nagyítani az egér görgőjével és a „Page up/down” gombokkal. Az **f** gomb megnyomásával az összes fordulópont látható a képernyőn. A fordulópontokat is ezen a felületen lehet szerkeszteni. A módosítások egy megerősítő üzenet jóváhagyása után töltődnek fel a repülőre, melyre az egy megerősítő válasszal reagál. Újat csak felszállás előtt lehetséges feltölteni.

Középen alul található egy több füllel ellátott rész, melynél kiválasztható, hogy építen egy műhorizontot, a GPS által vett adatokat, az útvonaltervet vagy a beállításokat szeretnénk-e látni.



1.7. ábra

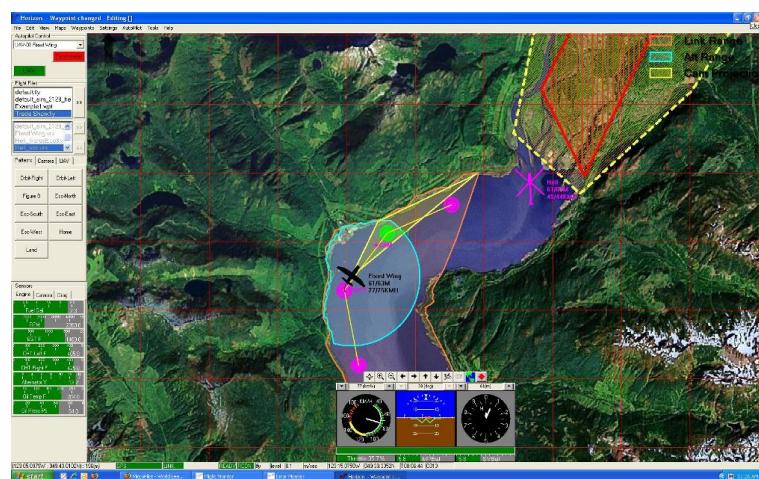
### Összegzés

Itt sem létfontosságú tényező az elemek redundanciája, így nincs felkészítve egy járműből érkező párhuzamos jelek fogadására. Egy képernyő szolgál a navigációra és az útvonal-kijelölésre, mely szerintem nem a legjobb megoldás. Összességében ez is egy nagyon jól használható program, mely teljesen kiszolgálja az UAV-kat melyhez készítették

#### 1.5.3. MicroPilot Horizon

[10] Az 1995 óta működő kanadai [9] MicroPilot a világ egyik legismertebb robotpilóta gyártó cége, 65 országban több mint 750 alkalmazó használja az eszközeiket. Népszerűek az iskolákban, egyetemeken, kutató intézetekben, a gazdaság különböző területein, de a védelmi szféra is szép számmal használ robot légi járművein MicroPilot berendezéseket. Az adatátviteli csatorna ugyanazokat a funkciókat képes biztosítani, mint a programbevitelnél használt kábeles összeköttetés, így ezen keresztül repülés közben is módosítható az útvonal, a repülési paraméterek és az egyéb beállítások. A MicroPilot fedélzeti egysége ezen kívül egy rádió távirányító (RCON) vevőberendezést is fogad, amely lehetőséget teremt a földi adó konzoljáról az irányítás átvételére és botkormányos kézi vezérlésre. Ezt alapvetően a fel és leszállás idejére, illetve az útvonal kritikus szakaszain használják. Amennyiben az RC távirányítóval működő repülőgép veszti el a kapcsolatot, akkor a fedélzeti vevőberendezés FAILSAFE üzemmódra kapcsol és annak beállítása szerint működteti a repülőgépet.

A FAILSAFE vagy az utolsó szervo állást őrzi meg, vagy egy előre programozott legbiztonságosabb földet érért ígérő beállításra ugrik. A hozzá készített földi irányító egység a MicroPilot Horizon nevet viseli. Többek közt a következő tulajdonságok jellemzik:



1.8. ábra

## Biztonság

A térképen kijelölhetők azok a területek, ahova nem szabad berepülni, ha a közelébe kerülne a repülő, figyelmeztetést kap az operátor. Automatikus tengerszint feletti magasság és leszállóhely beállítás, arra az esetre, ha a kapcsolat megszakadna a földi irányítással.

## Operátor központúság

Repülési terv könnyen, kattintással összeállítható és módosítható, minden repülés előtt és közben. POI<sup>5</sup> gombnyomásra lerakható a térképen, a jövőbeli kivizsgáláshoz. Vonalzó eszköz segítségével könnyen végezhet számítást az útvonal hosszával kapcsolatban. Gazdag online súgója gyorsan választ ad a kérdésekre.

## Több eszköz egyidejű kezelése

Több eszközhöz tud egyszerre csatlakozni és kiszolgálni, legyen az repülő vagy helikopter, mindegyikhez külön név rendelhető. Az összes csatlakoztatott jármű között szinkronizálva vannak az útvonalpontok, egy-több és több-több topológia kialakítására is van lehetőség.

## Szimuláció

Már a legelőször kiadásnál rájöttek a szimuláció jelentőségére, mivel ezzel a megoldással a legegyszerűbb a jövőbeli operátorok kiképzése. Lehetőség van minden platform szimulálására SIL módban.

---

<sup>5</sup>Point Of Interest, érdekes hely mely GPS koordinátával megjelölhető

## Kamera

A repülőgépre szerelt kamera képének megjelenítése kulcsfontosságú, mivel a kezelő ezzel láthatja leginkább a repülőgép helyzetét és a megfigyelt célpontot. MPEG4 tömörítés csökkenti a szükséges sávszélességet. Ebben a módban a legfontosabb repüléssel kapcsolatos információk átlátszóan jelennek meg, így nem kell másik képernyőre tekintenie a kezelőnek.

## Figyelmeztetés

Különböző hibák léphetnek fel repülés közben, ezeket egy naplóban tárolja a program. Lehetőség van különböző prioritási szintek meghatározására, hangjelzés hozzárendelésére. Így egy bizonyos szint alatti hibák nem terelik el az operátor figyelmét.

## Telemetriki adatok megjelenítése

15 érzékelő adatait tudja grafikusan ábrázolni, elmenteni, betölteni. Statisztikai analízist biztosít, átlag, minimum, maximum keresésre. Kiugró értékeket pirossal megjelöl, hogy könnyen észrevehető legyen egy jel határon kívüli értéke.

## Összegzés

Ez a program elsősorban távirányításos üzemmód támogatására készült, mely során az operátor irányítja a gépet és ehhez a legtöbb információt szolgáltatja. Így kulcsfontosságú a videókép átvitel támogatása és a legfelhasználóbarátabb megjelenítés.

### 1.5.4. OpenPilot

[11]

### 1.5.5. QGroundControl

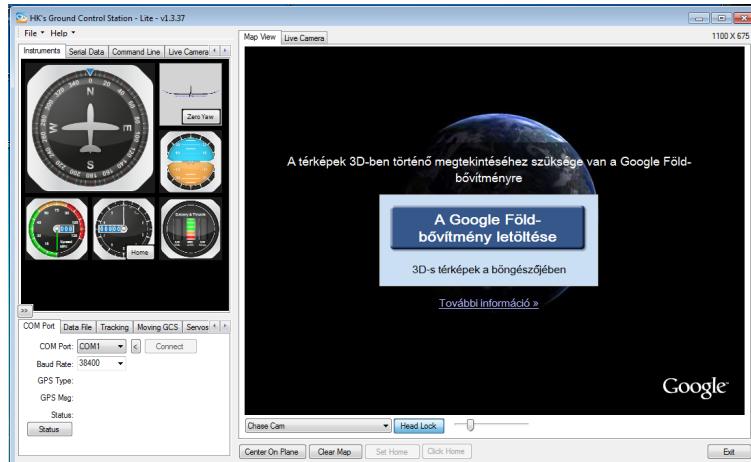
[12]

### 1.5.6. HappyKillmore

[13] 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 kíváncsiak vagy a soros port port számát szeretnénk beállítani. Lehetőségünk van exportálni is az adatokat.

### 1.5.7. Viking

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



1.9. ábra



1.10. ábra. *Viking*

## **2. fejezet**

# **Tervezés**

### **2.1. Kommunikáció**

A kommunikáció csatornánként 2 db modem segítségével történik, a modemek egymás közt vezeték nélkül csatlakoznak, felhasználói oldalon soros portot biztosítanak. Így az elkészítendő programnak elég csak a soros port jeleinek vételével foglalkoznia. A modemelek adatátviteli sebessége változó lehet, így azt a felhasználó egy listából választhatja csatlakozás előtt.

### **2.2. Adatok fogadása**

A redundanciából következően külön, külön kell kezelni a 2 párhuzamos csatornán kapott adatokat. Mivel aszinkron módon érkeznek a csomagok, így kettő tároló kell, mely a legutóbb küldötteket tárolja.

#### **2.2.1. Protokoll**

Az adatokat a repülőgép 2 Hz-s frekvenciával küldi, ezek csomagokban érkeznek, melyeknek a felépítése:

bájt index	leírás	típus	skálázás	offset
1	start	char(fix 'U')		
2	start	char(fix 'U')		
3	start	char(fix 'T')		
4	idő	uint32	10000	0
8	magasság	uin16	0x7fff/1000	
...				
TODO	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
...				

**2.1. táblázat.** *Fogadás protokollja*

A skálázás és offset képzés azért szükséges, hogy az adott szélességen (8, 16, 32 bit) minél több biten legyen ábrázolva egy érték, mivel kis változások esetén a Hamming-távolság<sup>1</sup> kicsi lenne az eredeti számábrázoláson. Ahol szükséges, ott a visszakódolás az alábbi formában történik :

$$\text{eredeti} = (\text{nyersadat/skalazás}) - \text{offset}$$

Az értékek megfelelő kiválasztása a minél nagyobb szétszóráshoz szükséges.

### 2.3. Adatok küldése

A repülőgép által lerepülendő feladat útvonalpontjait hasonlóképpen, mint az adatok fogadását, vezeték nélküli csatornán küldjük fel. A feltöltendő adat küldésének protokollja létfontosságú, mivel ha valamilyen hiba kerül a kommunikációba akkor az akár végzetes is lehet. Gondolok itt olyan hibára, hogy egy fordulópont koordinátája úgy kerül feltöltésre, hogy az kiesik a repülő hatósugarából és ezzel nem számolva, lemerül a tápellátást szolgáló akkumulátor. Az ilyen hibák ellen célszerű a feltöltés protokolljába hibadetektálást építeni, hogy ezek a feldolgozás( esetlegesen felszállás ) előtt kiderüljenek ki.

#### 2.3.1. Protokoll

Több megoldás is lehetséges a fordulópontok feltöltésére:

- Rögzített maximális darabszám elküldése egy csomagban
- Változó darabszám esetén egy fordulópont egy csomagban

Az első megoldásban rögzítenénk a fordulópontok maximális számát. Mely azt eredményezné, hogy egy csomagban el lehetne küldeni az egész lerepülendő feladatot. Ha egy pont koordinátájának ábrázolására elég 2\*4 byte, így ha feltételezünk egy MAXPOINT byte-os

---

<sup>1</sup>Bináris számok XOR képzésével kapott 1-esek száma

bájt index	leírás	típus	skálázás	offset
0	start	bájt(fixture 'G')		
1	start	bájt(fixture 'P')		
2	start	bájt(fixture 'S')		
3	pontok száma	bájt		
4	pontok[0].lat	uin32	0x7fff/360	180
...				
8	pontok[0].lon	uin32	0x7fff/360	180
...				
12	pontok[1].lat	uin32	0x7fff/360	180
...				
16	pontok[1].lon	uin32	0x7fff/360	180
...				
78	checksum 1/2	uin16		
79	checksum 2/2	uin16		

## 2.2. táblázat. Küldés protokollja

csomagot, akkor abba beleférne kb. MAXPOINT darab. Egy csomag állna egy fejlécből, a feltöltendő pontok számából, a pontokból, és egy checkszámból.

Másik lehetőségnél N db-t lehetne feltölteni: egy csomag szerkezete: fejléc, küldendő pontok száma, aktuális pont sorszáma, koordinátái, checkszám. Ha a küldendő pontok száma és az aktuális pont sorszáma megegyezik és megérkezett minden csomag akkor ACK-val válaszol ha kész a feltöltés.

Mivel az eddig használt megoldásban a kódba „bele volt égetve” az útvonalterv, mely 5-6 pontot tartalmazott, az első megoldás tűnik kedvezőbbnek. Fogadó oldalon is könnyebb egy ilyen lehetőségre felkészíteni. Ha esetlegesen a jövőben több fordulópontot feltöltésére lesz igény, az is megoldható kis módosítással.

A modem is duplikált, így a 2 soros portra csatlakoztatott modemmel redundánsan küldjük el a csomagot. TODO TODO??? Ha a csomag sértetlenül megérkezett, ACK jelzéssel válaszol.

## 2.4. Grafikus felület

Az előző fejezetben ismertetett grafikus felületekből levonva a következtetéseket, nyilvánvaló, hogy a GUI kialakításában fontos a repülőgép aktuális pozíciójának térképen való mutatása, az repülési állapot könnyen értelmezhető megjelenítése, illetve az esetlegesen előforduló problémák feltűnő jelzése.

### 2.4.1. Főképernyő

A főképernyőn látható lesz a repülőgép aktuális pozíciója és iránya. A pozicionálást segítendő, egy térkép lesz egy repülőgép ikon hátterében. Ez a rész a képernyő kb. 2/3-át fogja elfoglalni. Az oldalsó sávban a „Glass Cockpit” kerül kialakításra, ez egy műhorizont, mely a gép által küldött orientációs adatokból az operátor számára informatívan ábrázolja

a repülési állapotot. Ez a nézet tartalmazza még a gép aktuális sebességét, irányát, melyet egy iránytű segítségével láthat. Valószínűleg ez a képernyő lesz legnagyobb százalékban használva, így a kritikus hibákról itt kell feltűnő értesítést adni.

#### **2.4.2. Tervezés képernyő**

Ezen a képernyőn a felhasználó kijelölheti a lerepülendő útvonalhoz tartozó fordulópontokat, melyet csatlakozás után aszinkron módon feltölthet a repülőre. Mivel a kommunikációs protokoll MAXPOINT pontot enged meg, így ennél többet itt ki sem jelölhet, a lerakott pontok helyét a megszokott Google Maps-hoz hasonló módon hosszan kattintva átrakhatónak kell lennie, illetve köztes pontoknak törölhetőeknek kell lenniük.

#### **2.4.3. Diagnosztikai képernyő**

A 2 porton érkező dekódolt értékek látszódnának 2 oszlopan, mellettük egy hibaérték, mely a különböző hibatípusok hibaszámának összege lenne.

##### **Hibatípusok**

- beragadás
- túl nagy változás
- túl nagy különbség a 2 vett értéken

Ezek feldolgozására 2 FIFO sort kell alkalmazni, melyek visszamenőleg tárolják a beérkező értékeket. Ez azért szükséges, mivel így a túlságosan kiugró értékeket detektálni lehet, illetve, ha az egész sorban ugyanazok az értékek vannak, gyanús a beragadás esélye. A harmadik esetben sajnos nem lehetséges a „jó” kiválasztása, mivel nem tudjuk, melyik modemből érkezett adat a megfelelő. Ezt csak háromszorozással és többségi szavazással lehetne megoldani.

### 3. fejezet

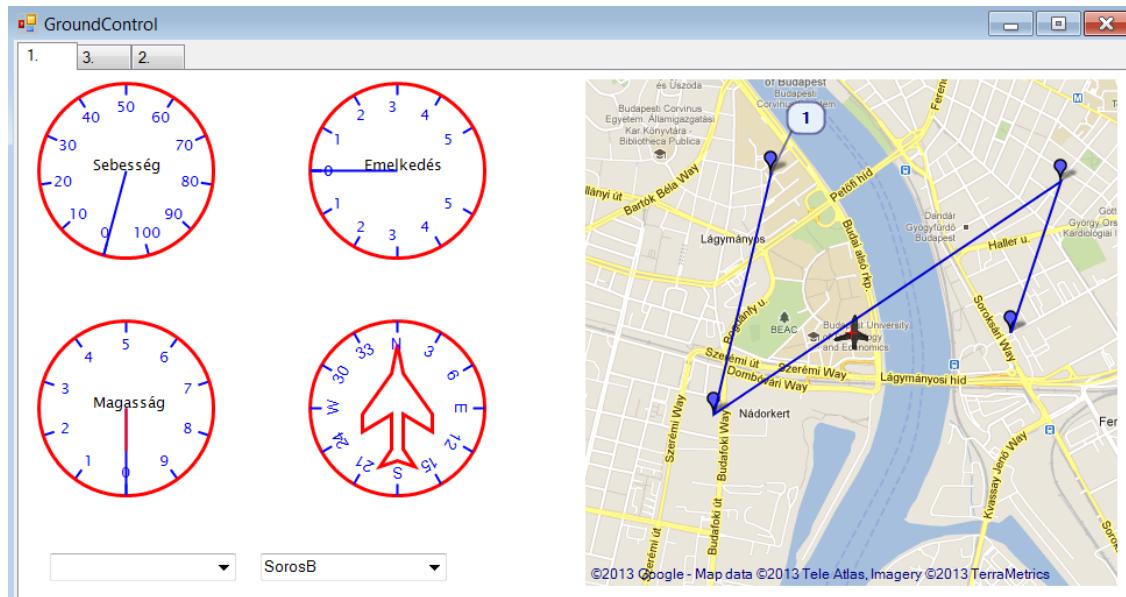
## Megvalósítás

Az előző fejezetekben összegyűjtöttem a megvalósításhoz szükséges információkat, tervezési lépéseket. Ebben a fejezetben a konkrét implementációt fogom bemutatni.

### 3.1. .NET

C# nyelven szükséges a program implementálása, a .NET keretrendszer szerencsére sok API<sup>1</sup>-t biztosít a fejlesztők számára, hogy megkönnyítse és gyorsítsa folyamatot. Így a grafikus megjelenítéshez GDI-t használhatok, illetve a soros porti kommunikációra a SerialPort osztály adta lehetőségek.

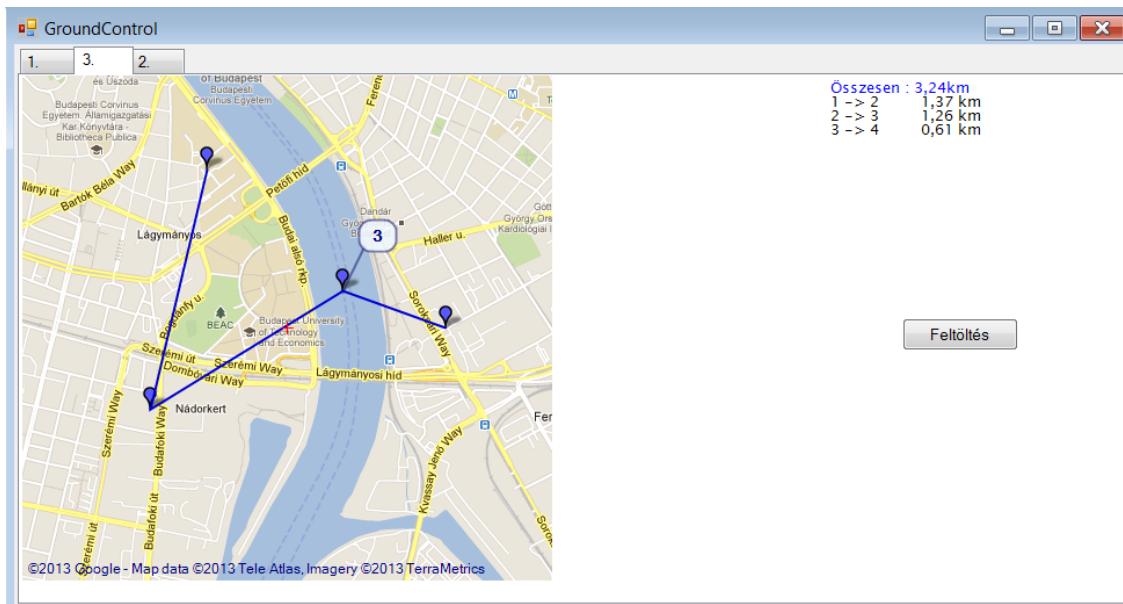
### 3.2. Főképernyő



3.1. ábra

<sup>1</sup>Application Programming Interface, mely előre megírt komponensek használatához biztosít interfész

### 3.3. Tervezés képernyő

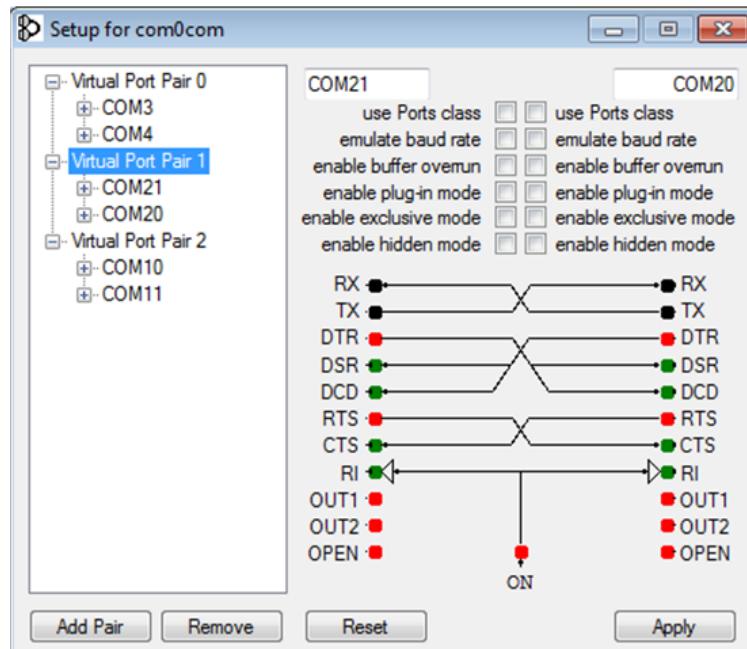


3.2. ábra

### 3.4. Diagnosztikai képernyő

#### 3.5. Soros port

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 [15]null-modem segítségével sorosporton keresztül küldöm a megfelelő portra.



3.3. ábra

Beállítottam 2 pár, COM20-COM21 és COM10-COM11 között, a pároson küldöm, páratlanon fogadom az üzeneteket.

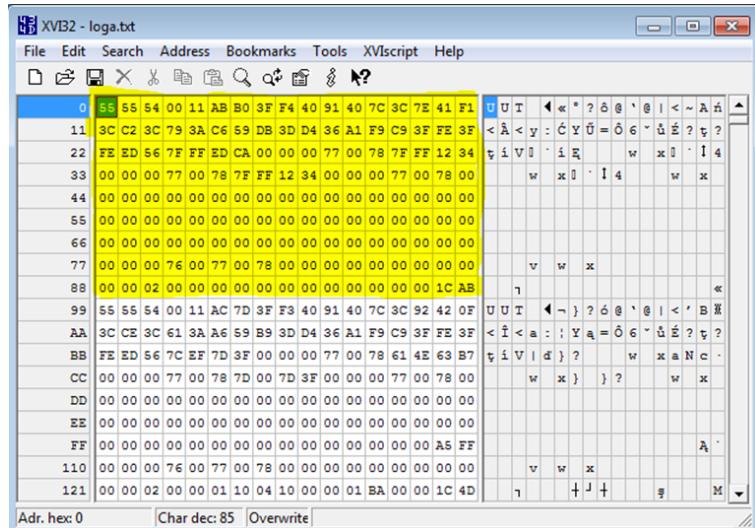
```

file:///C:/Users/Pasquale/Desktop/sorosend/sorosend/bin/Debug/sorosend.EXE
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

```

3.4. 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.



3.5. ábra

Fogadó oldalon a két sorosport aszinkron ír 1-1 byte tömböt, melyből egy dekódoló függvényteljesen 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 hármaszt. 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.

## **4. fejezet**

### **Értékelés**

**Megtervezett műszaki alkotás értékelése, kritikai elemzése, továbbfejlesztési lehetőségek**

## 5. fejezet

# Összefoglalás

### SZUMM

A program megalkotásához nem használok automatikus kódgeneráló és formális módszereket támogató segédeszközöket, így a fejlesztési folyamatba 1000 sorból kb.10 hiba[2] maradhat a rendszerben. Ami odafigyeléssel és teszteléssel, illetve a tervezés során elvégzendő validációval és verifikációval lejjebb vihető. Az implementáció során elkövetett meghibásodásból, pl. egy számláló rosszul megírt növeléséből, ha ráfut a vezérlés akkor hiba keletkezik, mely a felhasználó szempontjából hibajelenséget okoz. A hatásláncot a meghibásodási tényező csökkentésével és hibajelenség kialakulásának megakadályozásával lehet befolyásolni. Előbbit ellenőrzéssel és teszteléssel, utóbbit helyes kivételkezeléssel. Több hibatípus léphet fel, egyik az előre ismert, másik az előre nem ismert. Az előre ismert hibatípusokat optimálisan lehet kezelní a tervezés során, az előre nem ismertek ellen megfelelő rendszerstukatúra kialakítása szükséges. Esetünkben előre ismert hiba lehet

- egy vagy kettő csatorna kiesése
- csomagvesztés
- küldött érték beragadása
- küldött érték túl gyors változása

# Irodalomjegyzék

- [1] <http://www.azom.com/article.aspx?ArticleID=10156>, 2013. november 19., 10:00
- [2] <http://www.inf.mit.bme.hu/sites/default/files/materials/category/kategória/oktatás/bsc-tárgyak/rendszermodellezés/13/Hibamodellezés.pdf>, 2013. november 17., 19:00
- [3] <http://www.uvisionuav.com/portfolio/gcs/>, 2013. november 24., 15:00
- [4] [http://personal.us.es/imaza/papers/journals/maza\\_jint10\\_multimodal/maza\\_jint10\\_multimodal\\_v](http://personal.us.es/imaza/papers/journals/maza_jint10_multimodal/maza_jint10_multimodal_v), 2013. november 24., 15:00
- [5] <http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-pro-868>, 2013. október 29., 14:00
- [6] <http://msdn.microsoft.com/en-us/library/system.io.ports.serialport.aspx>, 2013. március 20., 10:00
- [7] <https://code.google.com/p/ardupilot-mega/wiki/Mission>, 2013. november 24., 15:00
- [8] <http://paparazzi.enac.fr>, 2013. november 24., 15:00
- [9] <http://www.micropilot.com/products-horizonmp.htm>, 2013. november 24., 15:00
- [10] [http://www.szrfk.hu/rtk/kulonszamok/2012\\_cikkek/77\\_Makkay\\_Imre-Papp\\_Timea.pdf](http://www.szrfk.hu/rtk/kulonszamok/2012_cikkek/77_Makkay_Imre-Papp_Timea.pdf), 2013. november 24., 15:00
- [11] <http://wiki.openpilot.org/display/Doc/OpenPilot+Documentation>, 2013. november 24., 15:00
- [12] <http://qgroundcontrol.org/>, 2013. november 25., 15:00
- [13] <https://code.google.com/p/ardupilot-mega/wiki/HappyKillmore>, 2013. március 19., 16:00
- [14] [http://www.vikingaero.com/uav\\_ground\\_control\\_station.html](http://www.vikingaero.com/uav_ground_control_station.html), 2013. március 19., 16:00
- [15] <http://com0com.sourceforge.net/>, 2013. május 4., 10:00

# Függelék

## F.1. Függelék1