Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Hollós Ádám

MÉRŐRENDSZER GNSS VEVŐK TULAJDONSÁGAINAK VIZSGÁLATÁRA

KONZULENS

Dr. Kovácsházy Tamás

BUDAPEST, 2018

Tartalomjegyzék

Összefoglaló

Az elosztott mérő- és irányítórendszerek terjedésével (például Kiber-fizikai rendszerek, Ipar 4.0) egyre gyakrabban szükséges egy elosztott rendszer komponensei közt a szinkron mintavétel és beavatkozás biztosítása, amelyhez szükséges a rendszer óráinak, beleértve a mintavételi és beavatkozási folyamat vezérlőket is, szinkronizálása. A legtöbb esetben nem elég csupán a lokális szinkronizálás, hanem a rendszert egy globális időhöz, többnyire GNSS (Global Navigation Satellite System, pl. GPS) által szolgáltatott referencia időhöz szükséges szinkronizálnunk GNSS vevők felhasználásával, hogy ezzel lehetővé váljon térben is távol elhelyezkedő rendszerekben az egyidejű mintavétel (pl. energiaelosztó rendszerek (smart grid), autonóm járművek együttműködése, stb.) és globális idő alapú időbélyegek rögzítése és továbbítása is.

Ezekben a rendszerekben a mesteróra az a komponens, amely meghatározza a globális időt (pl. GNSS vevő segítségével), majd többnyire az IEEE 1588 (Precision Time Protocol) felhasználásával továbbítja azt a lokális kommunikációs hálózatban. Az ilyen eszközök tulajdonságait ezért alapvetően korlátozza a bennük alkalmazott GNSS vevő pontossága és megbízhatósága. Sajnos napjainkban nincs az oktatásban és kutatásban használható kedvező árú és elfogadható tulajdonságú, akár open hardware és szoftver komponensekből építkező, mesteróra. Egy ilyen kifejlesztése során viszont a fentiek szerint alapvető kérdés az alkalmazott GNSS vevő tulajdonságainak ismerete, vizsgálata, ugyanis az adatlapok ilyen tekintetben kevés információt tartalmaznak. Dolgozatom témája ennek megfelelően a potenciálisan alkalmazható GNSS vevők óraszinkronizáció szempontjából jelentős tulajdonságainak vizsgálata, melyhez kialakítottam egy egyedi mérőrendszert. Ez magában foglalja egy teszt NyÁK megtervezését és összeszerelését, egy PC-s adatgyűjtő program megírását, MATLAB-os szkriptek kidolgozását az adatok kiértékeléséhez, és az adatgyűjtő rendszer validálását mely TI-os fejlesztőkártyák által szimulált GNSS vevők segítségével történt. Az így kialakított mérőrendszerrel méréseket is végeztem a Quectel L86 GNSS vevőn, mely mérések eredményeit a dolgozatomban ismertetem.

Abstract

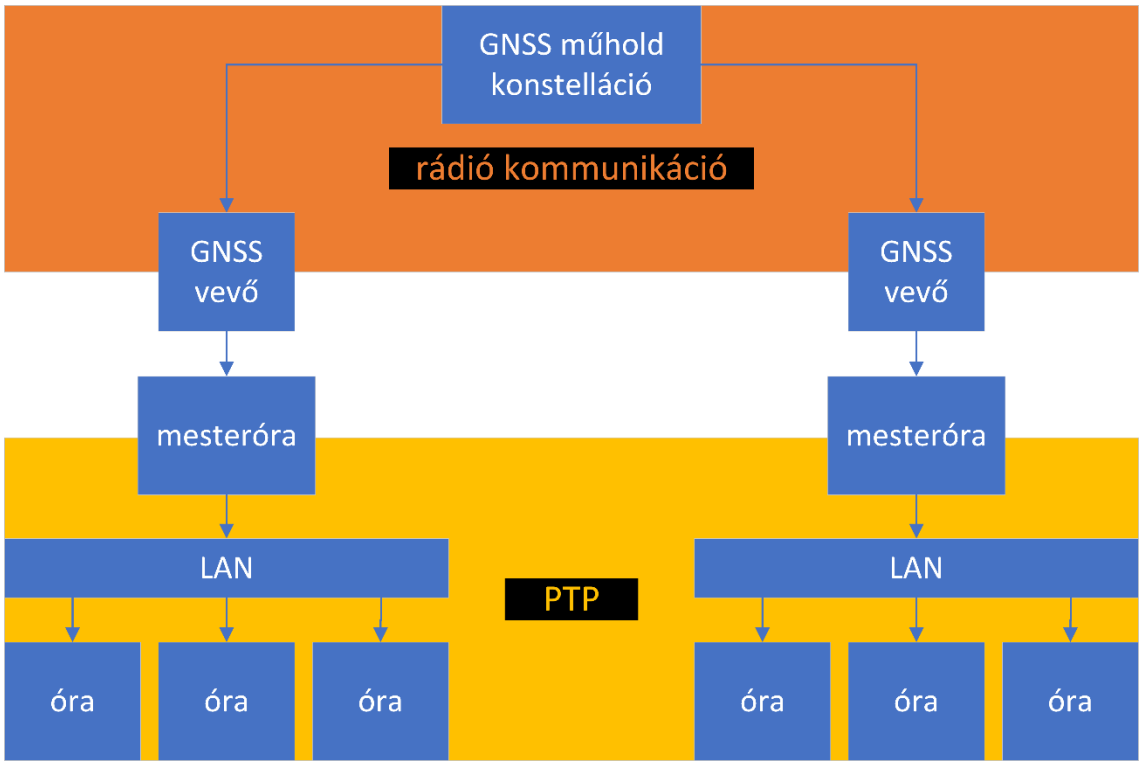
With the continuing spread of distributed measurement and control systems (e.g. Cyber-Physical Systems, Industry 4.0) there is an increased need to implement synchronized sampling and actuation over the components of a distributed system. To achieve this goal, the clocks of the system must be synchronized, including those of the sampling and actuation controllers. In most cases local synchronization alone is not enough; It is also needed to keep in sync with a global reference time in order, for example, to be able to implement timestamps based on global time and simultaneous sampling across system components that are spatially wide spread. (Think of smart grid, which is an electricity supply network paradigm, or cooperation between autonomous vehicles, and so on and so forth.) This global reference time mostly means one which is provided by a GNSS (Global Navigation Satellite System) service, such as GPS.

In these systems it is the master clock which is responsible for attaining (e.g. via a GNSS receiver) and keeping the global reference time. It can then pass it on in the local communication network, usually using IEEE 1588 (Precision Time Protocol). It follows, that thus, the characteristics of such systems are heavily limited by the accuracy and reliability of the used GNSS receiver. Regrettably, there aren't currently any master clocks available for educational or academic research purposes, be it built from open hardware and software components, which would have the desired attributes and a friendly price tag. On the other hand, developing just such a device is possible, given that we know the characteristics of the GNSS receiver that is considered to be used. However, the datasheets provided by the manufacturers are lacking in the necessary and accurate information, so examination and careful evaluation of the receivers are needed. My paper thus is about the examination of potentially promising GNSS receivers for which I developed a custom measurement system. This includes the design of a test PCB, the writing of a PC data collecting program, the formulation of MATLAB scripts which process the collected data, and the validation of the data collecting system via GNSS receivers simulated by TI evaluation boards. I also conducted measurements with the system on the Quectel L86 GNSS receiver, the results of which are presented in the paper.

1 Bevezetés

Manapság egyre nagyobb hangsúly helyeződik az iparban az elosztott rendszerekre; gondoljunk csak az Ipar 4.0 víziójára vagy a dolgok internetére (IoT). Ezek a Kiber-fizikai rendszerek (CPS) több problémát is felvetnek; például szükség lehet a rendszerkomponensek közt egyetemesen értelmezett időbélyegzésre, szinkronizált beavatkozásra vagy egyidejű mintavételezésre. Ahhoz, hogy ezen dolgok bármelyike megvalósítható legyen szükség van egy egyetemes rendszeridőre, ami csak úgy valósítható meg, hogy a rendszerkomponensek óráit egymáshoz szinkronizáljuk (és szinkronban tartjuk). Erre a problémára kínál megoldást többek közt az általában UDP/IP szállítási-hálózati protokoll felett megvalósított Network Time Protocol (NTP) (WAN és LAN) és Precision Time Protocol (PTP, IEEE 1588) (csak LAN), illetve az általában gyártósori automatizálásnál használatos EtherCAT busz Distributed Clocks (DC) protokollja (csak LAN). Az NTP-vel ~100 ms pontosságú, PTP-vel és DC-vel ≪1 µs pontosságú szinkronizáció érhető el [1] [2] [3]. Mivel az NTP pontossága sokszor nem elég, viszont globális, azaz LAN-ok közti szinkronizáció is szükséges, ezért gyakran az egyes helyi hálózattokat valamely Global Navigation Satellite System-hez (GNSS) szinkronizáljuk; használható például az amerikai Global Positioning System (GPS) vagy az orosz Глобальная навигационная спутниковая система (GLONASS). Ilyenkor a műholdas adást egy GNSS vevő veszi, amely meghatározza a pontos időt majd közli azt a LAN mesterórájával ami végül végrehajtja a helyi órák szinkronizálását, többnyire PTP-vel. A GPS például ≤40 ns-os pontossággal képes megmondani az időt [4]. Egy ilyen rendszert ábrázol az 1.1-es ábra.

Ahogy az az 1.1-es ábrából is kitűnik a szinkronizáció sikerességében kritikus feladat hárul a GNSS vevőkre és az egyes vevők és a mesterórák közti kapcsolatra. A vevő-mesteróra kapcsolat lehetséges megvalósításairól részletesen írtam egy konferencia cikkben ([5]) mely [6] folytatásának vehető; a GNSS vevő általában egy PPS jelet szolgáltat mely nanoszekundumos pontossággal az adott másodperc kezdetekor fut fel, a mesteróra ezt figyelve és a soros vonalon közölt idő és dátum információk alapján állapítja meg az időt. Ezen dolgozatom fókusza magának a GNSS vevő tulajdonságainak a vizsgálata, illetve az e célból készített mérőrendszerem és a vele végzett első mérések bemutatása.



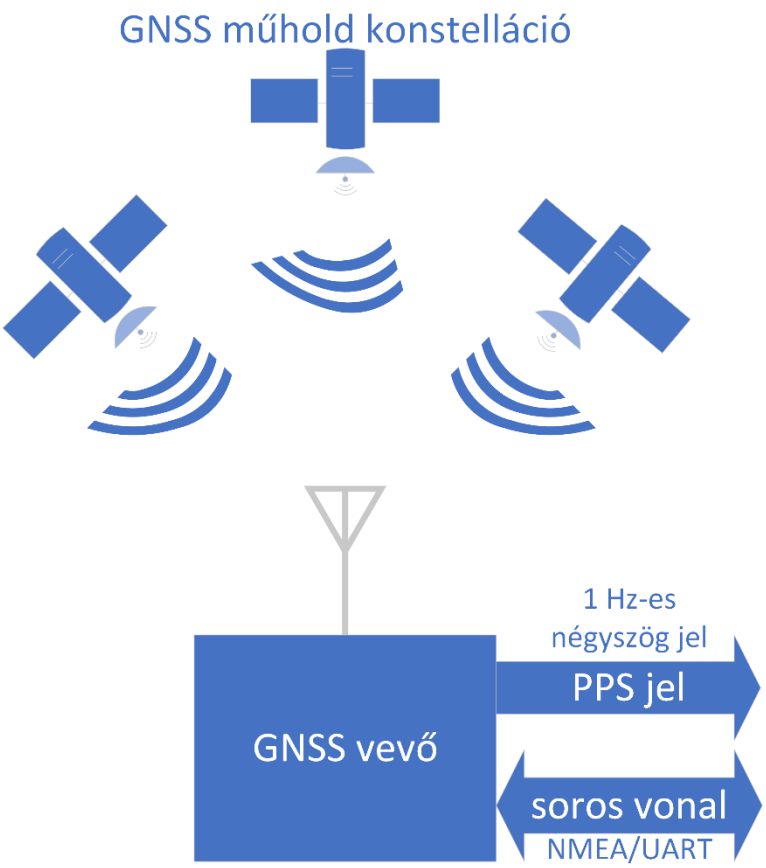
1.1. ábra: PTP-s helyi hálózatok szinkronizálása GNSS-el

1.1 Motiváció

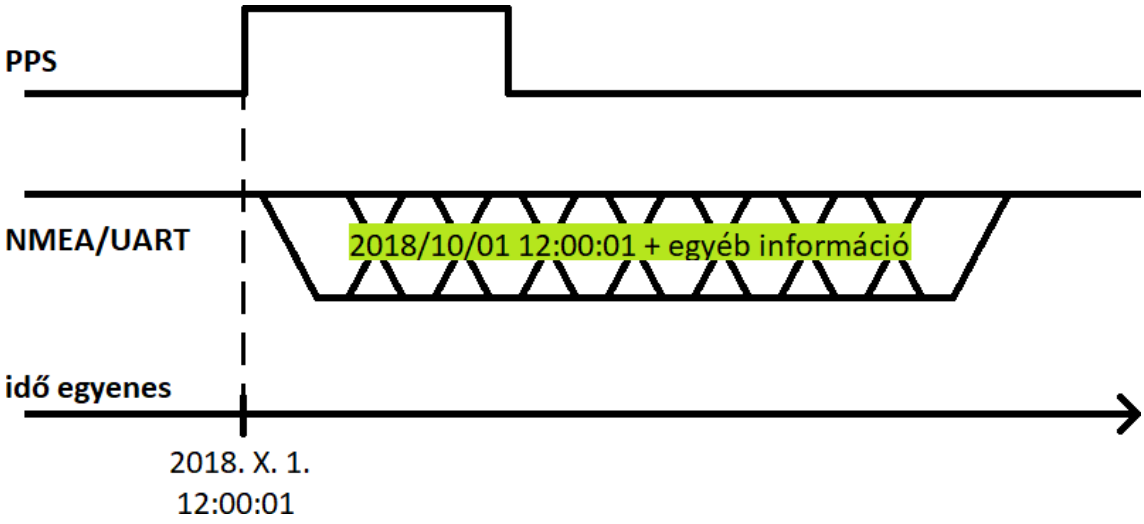
Kutatásom célja egy olyan alacsony bekerülési költségű open-source rendszer kifejlesztése mely, ha az 1.1-es ábrát nézzük, akkor magában foglalja a GNSS vevőt és a mesterórát és egy Ethernet alapú IEEE 1588 2008 (PTP) helyi hálózat mesterórájaként használható. A rendszer célközönsége elsősorban a PTP-vel foglalkozni kívánó kutatók vagy a PTP-t oktatni kívánó tanárok, akik elől így elgördülne egy jelentős belépési korlát. Ezért fontos, hogy a GNSS vevő is minél olcsóbb legyen; szerencsére az utóbbi időben számos a célnak elvileg (adatlap alapján) megfelelő eszköz vált elérhetővé a piacon. Azonban a vevő és mesteróra közti kapcsolat kiépítésére irányuló munkám során kiderült, hogy az adatlapokban foglaltak és a valóság sajnos esetenként jelentősen eltér [5] [7]. Ezért, hogy a vevők tulajdonságait és azok különféle paraméterektől való függését vizsgálni lehessen egy mérőrendszer és mérési eljárás kidolgozása vált szükségessé, aminek célja, hogy objektív módon ki tudjam választani a legolcsóbb megfelelő vevőt. E mérőrendszer és eljárás dolgozatom témája.

2 GNSS vevők bemutatása

Igen sok fajta GNSS vevő érhető el a piacon. Szinte mindenre létezik specializált vevő, mely kihasználja az adott felhasználási mód specialitásait. Jelen kutatás szempontjából az úgynevezett stationary timing GNSS vevők az érdekesek. Ezek azt használják ki, hogy a vevő pozíciója ismert és nem változik; így a vevőnek nem szükséges folyton újra számolnia a pozíciót, elég csak a pontos időt meghatároznia. Ebben ráadásul segíti az is, hogy tudja, hogy mi a valós pozíciója és így pl. el tudja dobálni azokat a műholdakat, amik valamilyen oknál fogva téves adatokat adnak (pl. többutas terjedésből, vagy az ionoszféra zavaraiból következően) [8]. A fix pozíció kezdeti meghatározásra is többféle lehetőség létezik: meg lehet például adni a vevőnek, hogy mi a pontos pozíciója, de olyan vevő is létezik mely indításkor körülbelül egy óráig statikus helymeghatározás üzemmódban működik és így határozza meg átlagolással a pozícióját, majd ezután vált át timing üzemmódba [8]. Kialakítást tekintve is többféle vevő létezik: van olyan vevő melyre egy külső antennát kell csatlakoztatni, de létezik olyan is amely a vevővel egybeépített patchantennával rendelkezik [8] [9]. A vevők abban is különböznek, hogy melyik GNSS adását képesek venni. Léteznek olyanok, melyek például csak a GPS műholdakat veszik, de vannak olyanok is melye egyszerre többfajta műhold rendszer műholdjait képesek használni, pl. GPS + GLONASS [8] [9]. A funkcionalitást tekintve a legtöbb GNSS vevő a 2.1-es ábra szerinti felépítést követi: a vevő veszi a műholdak rádió jeleit, ebből előállítja a pozíció-, dátum- és időinformációkat melyeket egyéb hasznos adatokkal együt NMEA üzentetek formájában egy UART alapú soros vonalon tesz elérhetővé. A vevő továbbá előállít egy pulse-per-second (PPS) jelet, mely a precíz időzítési információt hordozza; a PPS jel másodpercenként 1 darab felfutó (konfigurációtól függően esetleg lefutó) élet tartalmaz, mely adott pontossággal pontosan a soros vonalon közvetlen a felfutó él után küldött NMEA üzenetblokkban közült másodperc végekor fut fel, ld. 2.2-es ábra.



2.1. ábra: átlagos GNSS vevő funkcionális diagramja



2.2. ábra: GNSS vevő által történő pontos idő közlésének diagramja

2.1 NMEA protokoll

Az NMEA protokoll a GNSS vevők által általánosan használt ASCII alapú kommunikációs protokoll. Az adatkapcsolati réteget jellemzően UART valósítja meg. Az üzenetek $-el (dollár jelel) kezdődnek majd ezután az üzenet típusát megadó karakter sor

áll. Az üzenetek mezőkből álnak, melyek egymástól ,-vel (vesszővel) vannak elválasztva. Az üzenet típusa után közvetlenül az első mező jön, az üzenet típusától vesszővel elválasztva. Az utolsó mező után egy * (csillag) karakter áll, miután a dollárjel utáni és a csillag előtti karakterek XOR-jából képzett 8 bit-es ellenőrző összeg áll 16-os (HEX) számrendszerben ábrázolva (2 karakter). Az üzenetet végül a \r\n (Windows-os új sor) karaktersor zárja. Tehát, a fizikai réteg tökéletlenségei miatt az üzenetekben keletkező hibák nagy többsége észrevehető. [10] [11] [9] [8]

Az NMEA protokoll segítségével lehet a vevőnek a működést befolyásoló parancsokat küldeni, például, megadni a pontos (fix) pozíciót, újraindítást előidézni, stb. Információt is így lehet kérni a vevőtől, de a GNSS vevő magától is küld periodikusan üzeneteket, melyek tartalmazzák a felhasználó számára szükséges információkat. Így, a pozíció-, dátum- és időinformáción kívül, a vevő állapotáról képet adó egyéb fontos adatokat is tartalmaznak ezek az üzenetek. Általában az is beállítható, hogy mely üzenetek szerepeljenek ebben a periodikusan (másodpercenként) elküldött üzenetblokkban.

Ilyen, a jelen kutatásom szempontjából fontos adatok például, hogy melyik műholdakat látja, mely műholdakat használja a vevő. Az egyes műholdakat mekkora carrier-to-noise-density ratio[1]-val (C/N_0-al) veszi. Hány műholdat lát. Hány műholdat használ. Mekkorák a megoldás bizonytalanságáról képet adó érzékenységi értékek: Position Dilution of Precision (PDOP), Horizontal Dilution of Precision (HDOP), Vertical Dilution of Precision (VDOP), esetleg Time Dilution of Precision (TDOP). Illetve, hogy sikerült-e egyáltalán a vevőnek kitalálnia a pozíciót és az időt.

2.2 Firmware

A GNSS vevő firmware-e is fontos része a vevőnek. Feladata, az egyenletek időre és pozícióra való megoldásán túl, (többek közt) a használt műholdak okos összeválogatása is. Ez azt jelenti, hogy az összes vehető műhold közül ki kell választania azokat, amelyek jelei alapján megoldva az egyenleteket, a megoldás érzékenysége (vagyis a Dilution of Precision értékek) a legkisebbek lesznek (ez a műholdak egymáshoz

[1] carrier-to-noise-density ratio (C/N_0): a vivő hullám teljesítménye osztva a zaj teljesítménysűrűségével, mértékegysége dBHz [21]

képesti helyzetétől függ). Továbbá azokat a műholdakat szabad csak felhasználnia, amelyek jelei nagy valószínűséggel egyenesen jutnak el a vevőhöz, tehát például azokat a műholdakat, amelyek jelei valószínűleg több utas terjedéssel vagy visszaverődéssel jutnak el a vevőhöz, nem szabad felhasználnia. Triviális tehát, hogy a firmware-nek is jelentős szerepe van abban, hogy végül milyen lesz a GNSS vevő teljesítménye.

2.3 Antenna kialakítás

A GNSS vevőkkel sok fajta antenna konfiguráció használható. Van amelyik csak külső aktív antennával működik, van amelyik működik passzív antennával is, van olyan is mely egybe van építve egy patchantennával, tehát a tervezőnek már csak megfelelően kell kialakítania az eszköz körül a NyÁK-ot és az antennával sem kell vesződnie. Az aktív antennás megoldás előnye, hogy a GNSS vevő távol is elhelyezkedhet az antennától, mivel az antenna felerősíti, esetleg le is keveri a GNSS jelet középfrekvenciára, ami által még tovább is növelhető a vevő és az antenna közti kábelhossz. Hátránya azonban a magas ár és az antenna viszonylag nagy mérete. A passzív antenna szerelhető akár a NyÁK-ra is (pl. patchantenna) távolra viszont nem tehető, mert a GNSS által használt magas frekvenciákon (GPS L1 esetén pl. 1575,42 MHz a vivő [9]) a legjobb kábelek vesztesége is jelentős. További hátránya, hogy a NyÁK-on a tervező mérnöknek kel kialakítani a nagyfrekvenciás vonalakat, melyek nem feltétlenül esnek egybe egy beágyazott rendszerekkel foglalkozó szakember szakterületével. A vevő + patchantenna egybeépítve a legkényelmesebb, hiszen kompakt és itt csak a megfelelő földiköltést kell biztosítani az eszköz körül. Ezért én is egy ilyen, összeépített, vevőt használtam az [5]-ben és [7]-ben részletezett kutatásom során. Azonban, mint ahogy az [5]-ből kiderül, az, hogy hogyan is kell ezt a földkitöltést kialakítani a vevő körül sajnos nem olyan triviális, mint azt elsőre az ember gondolná. Ahogy azt az említett cikkben is írtam, habár betartottam az adatlap utasításait az antenna nyeresége mégsem lett megfelelő és így működés képtelen volt a vevő, mert nem volt elég nagy a látott műholdak C/N_0-ja. Mint ahogy az a későbbi irodalom kutatásom során kiderült, az adatlap által ajánlott földkitöltés nagysága messze elmarad a valószínűsíthetően szükséges mérettől [9] [12] [13]. [12] és [13] szerint a földkitöltésnek vélelmezhetően körülbelül egy 9,5 cm élhosszúságú négyzetnek kellene lennie legalább, ha a GPS L1 adását kívánjuk venni [5]. Azonban [12] szerint nincsenek még szilárd kutatási eredmények, konvenciók a kérdéssel kapcsolatban és én is ezt tapasztaltam. Ezért, a földkitöltés elégséges nagyságát kísérleti

alapon szükséges megállapítani. Egy dolog biztos, hogy minél nagyobb a földkitöltés, annál jobb az antenna nyeresége [13] [14].

3 A mérőrendszer

3.1 Motiváció

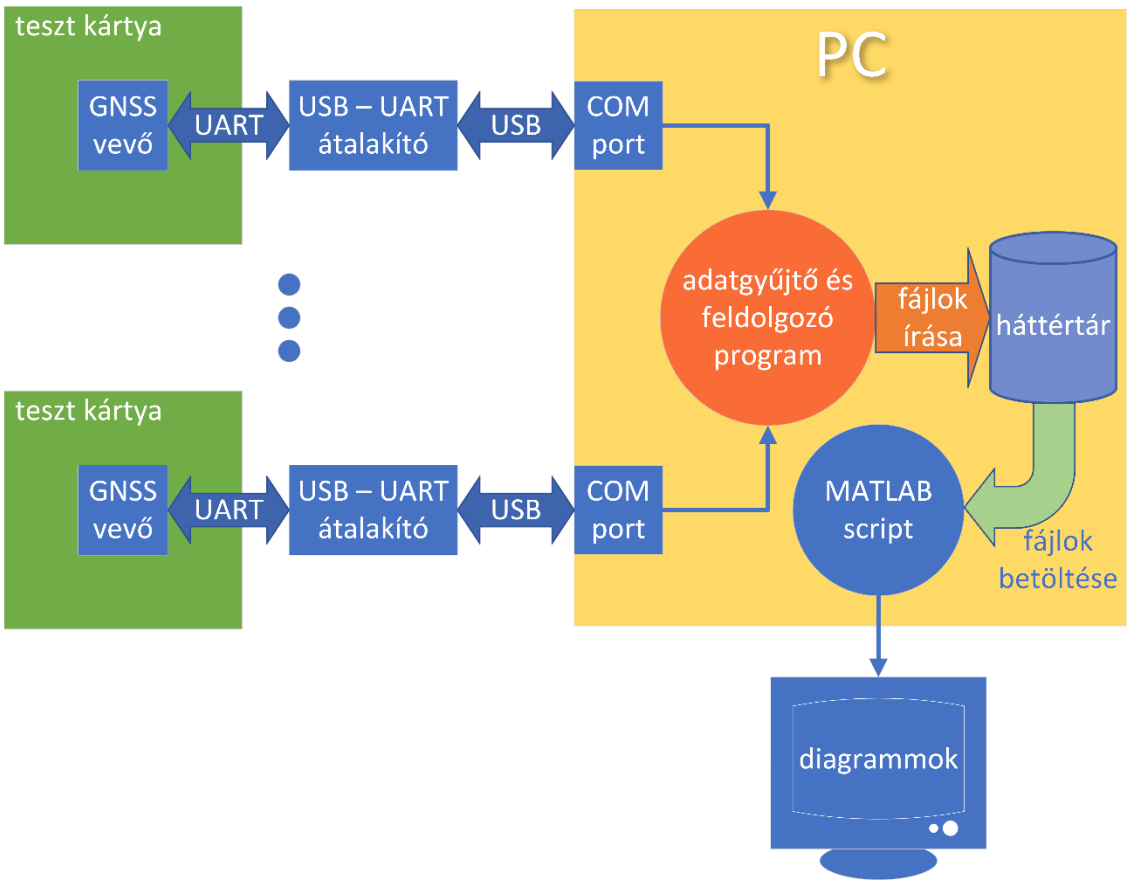
A rendszerrel az elsődleges célom az, hogy az [5]-ben és [7]-ben tárgyalt kutatásomban használt beépített patchantennás Quectel L86 GPS + GLONASS vevő köré kialakítandó földkitöltés szükséges nagyságát meghatározzam (ahogy az előzőekben is írtam, ez [5]-ben tapasztaltak nyomán szükséges [12] és [13] alapján); ha lehetséges viszonylag kis méretű földkitöltéssel megbízható működésre bírni a vevőt, akkor az azt jelenti, hogy az e dolgozat bevezetésében és [6]-ben, [5]-ben és [7]-ben tárgyalt rendszer (ld. 1.1-es ábra) kialakítható volna ezzel az igen kompakt és olcsó vevőnek a segítségével. Sőt, eredményeim alapján mások is használhatnák a vevőt hasonló kutatásaikban.

A rendszer további célja, hogy lehetővé tegye különböző GNSS vevők teljesítményének összehasonlítását (a teszt NyÁK-ot külön ki kell alakítani minden különböző vevőhöz), a vevők üzembiztonságának/rendelkezésreállási hányadának meghatározását, az egyes vevők különböző firmware-einek összehasonlítását, egy adott helyszín megfelelőségét GNSS vevő telepítésre való alkalmasság szempontjából és lehetővé tegye a vevő PPS jelében (egy másik mérőrendszerrel) megfigyeltek korrelálását a vételi körülményekhez.

3.2 Felépítés

A mérőrendszer felépítését a 3.1-es ábra mutatja. A GNSS vevők egy-egy teszt kártyán (NyÁK-on) kerülnek elhelyezésre. A vevők soros vonalait (UART Tx és Rx) egy-egy (CP2102-es IC-vel működő) USB-UART átalakító alakítja át egy USB vonallá. Ez az átalakító van hozzákötve egy USB kábellel (ha a kártyák száma nagyobb, mint a PC USB portjainak száma akkor egy USB HUB-on keresztül) az adatgyűjtést és az adatfeldolgozást/elemzést megvalósító PC-hez. A PC-n így, az átalakító dirverének köszönhetően, a GNSS vevők soros portjai COM portonként jelennek meg. A számítógépen futó C# nyelven írt .NET Framework alapú adatgyűjtő és feldolgozó program a .NET Framework szolgáltatásait használva begyűjti a COM portoktól a soros adatfolyamot, amiket rögzít és feldolgoz. Az eredményeket pedig szintén fájlokba írja. Ezután, a mérés végeztével, manuálisan lehet lefuttatni az adatmegjelenítést végző

MATLAB scriptet, mely betölti a C# program által generált eredmény fájlokat, átlagolást/szűrést végez rajtuk, majd grafikusan megjeleníti őket.

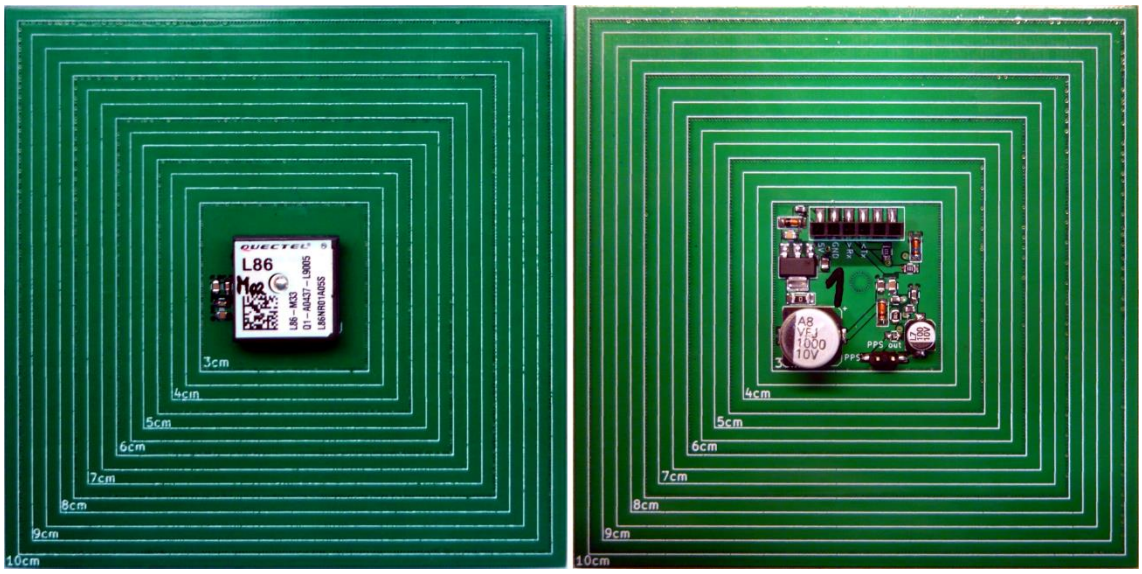


3.1. ábra: a mérő rendszer felépítése

3.3 Teszt kártya

A tesztkártya egy NyÁK amire a tesztelendő GNSS vevő felhelyezhető és amihez a 3.1-es ábra szerinti módon csatlakoztatható az USB-UART átalakító; a kártya feladata, hogy megfelelő tápellátást biztosítson a GNSS vevő számára (a kártya a tápot az átalakítón keresztül kapja), megfelelő földkitöltést biztosítson a vevő antennájához, hozzákösse a vevő soros vonalait az átalakító soros vonalaihoz és, hogy az USB kábelen esetlegesen jelenlévő vezetett EMI-t elszigetelje a vevőtől és annak antennájától. Ahogy azt a Motiváció-ban is írtam, a mérőrendszer egyik célja, hogy mérni lehessen a vevők teljesítményt különböző méretű földkitöltésekkel. Ezért, ahogy a 3.2-es ábra mutatja, minden alkatrészt a kártya közepén helyeztem el egy 3 cm-es négyzeten belül és a szitanyomaton vonalakkal jelöltem, hogy hol kell elvágni a NyÁL-t, hogy adott nagyságú földkitöltést kapjunk. Így a NyÁL (a földkitöltéssel együtt) könnyedén kisebbre vágható. A vevőn és a két-két legkisebb (kapacitású) tápszűrő kerámia kondenzátoron kívül (a

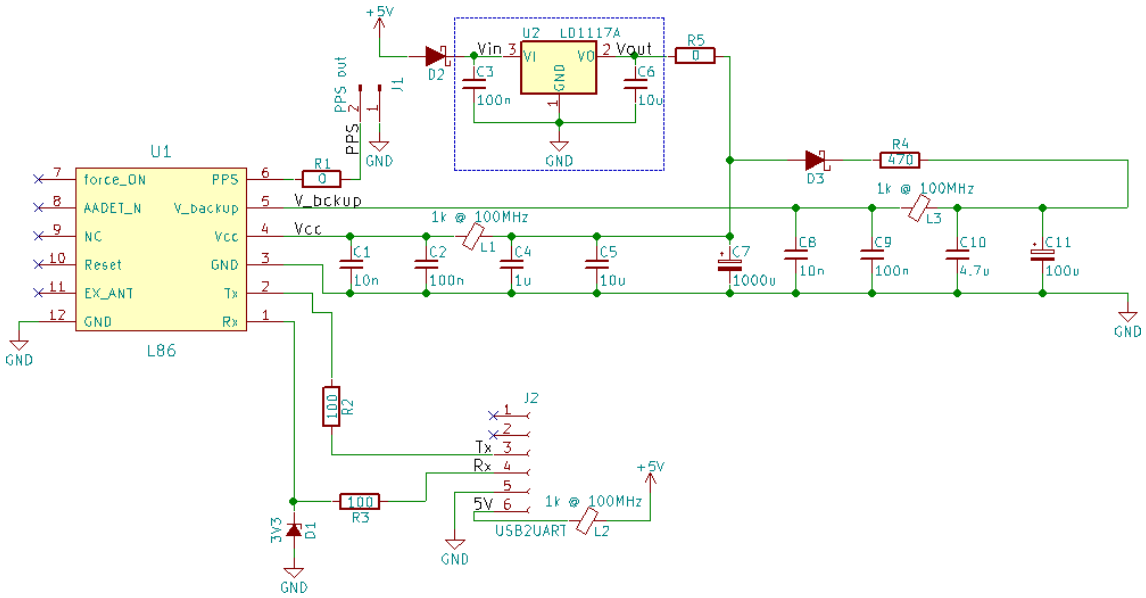
vevőnek két táplába van → az egyiken tartaléktápra van szüksége az eszköznek [9]) minden más alkatrészt a hátoldalon helyeztem el, hogy minél teljesebb lehessen a patchantenna körül a földkitöltés.



3.2. ábra: a vevővel szerelt teszt kártya; balra az eleje (a vevő középen látható), jobbra a hátoldala (az USB UART átalakító csatlakozási pontja a hüvelysor)

A kártya kapcsolási rajza a 3.3-as ábrán látható. A veőnek (U1) két táplába van; az egyik egyeken veszi fel az üzemszerű működéshez szükséges áramot, a másik lábon a vevőnek tartalék tápra van szüksége, melyet jelen esetben egy 100 µF-os kondenzátor biztosít. Erre a tartalékra azért van szükség, hogy a táp hirtelen elvétele esetén a vevő a flash-ét konzisztens állapotra hozva tudjon leállni. Továbbá, ha a tartalék táp később sem megy el, akkor a vevő képes megőrizni bizonyos számára fontos adatokat és hamarabb beméri a pozícióját mikor legközelebb tápot kap. [9] A vevőnek szükséges 3,3 V-os tápot egy LDO lineáris feszültség stabilizátor IC állítja elő az USB-UART átalakító által szolgáltatott 5 V-ból (az átalakító az USB kábelen keresztül kap tápot). Ez egyben kiszűri az USB kábelen és az átalakítón keresztül érkező esetleges vezetett EMI-t is. A vevő soros vonalai 100 Ω-os soros ellenállásokon keresztül csatlakoznak az átalakítóhoz, hogy a jelek élváltásai okán keletkező nagyfrekvenciás zavarok csökkentve legyenek. A vevő Rx lábán egy 3,3 V-os zéner diódát helyeztem el, hogy megvédjem a vevőt az esetlegesen rákapcsolt 5 V-os logikától. A vevő által előállított PPS jel egy PPS-föld tüskesorra (tüske párra) csatlakozik, hogy hozzáférhető legyen később. A PPS jelel is sorba kapcsolható egy ellenállás, de egyelőre ez a kivezetés nincs használva és így az ellenállás is 0 Ω-os. A fordított táprákötéstől egy Schottky dióda véd, ahogy a tartalék tápot biztosító

kondenzátort (C11) is egy Schottky dióda választja le a fő tápvonalról. Az 5 V-os tápcsatlakozó után egy ferrit csip van a nagyfrekvenciás EMI-k ellen és soros ferrit csipek vannak a tápvonalakban is a tápszűrő kondenzátorok mellett, hogy a kerámia kapacitások sajátos frekvencia-impedancia karakterisztikája okán esetlegesen kialakuló RL rezgőkör jósági tényezőjét a problémás frekvenciákon lerontsák.



3.3. ábra: Quectel L86-os GNSS vevőhöz készített teszt kártya kapcsolási rajza; J2 az USB-UART átalakító csatlakozási pontja, J1 tüskesorra ki van vezetve a PPS jel, U1 a vevő

A kártya mérete egy 10 cm oldalhosszúságú négyzet, mivel a szakirodalom alapján, a földkitöltésnek négyzet alakúnak kell lennie, amely oldalhossza az optimális vételi teljesítmény érdekében legalább $\frac{\lambda_0}{2}$-nek[2] kell lennie [12] [13]; esetünkben pedig (GPS L1: 1575.42 MHz) $\lambda_0 = \frac{c}{f} \approx 19\ cm \Rightarrow \frac{\lambda_0}{2} \approx 9{,}5\ cm$ [15]. Így a 10 cm-es oldalhossz a méréshez (kiindulási pontként) ideális.

3.4 PC-s adatgyűjtő és feldolgozó program

Az adatgyűjtő program feladata, hogy a vevők által küldött NMEA üzeneteket az 3.1-es ábra szerinti architektúrában begyűjtse és feldolgozza. A programot, ami a .NET Framework szolgáltatásaira épít, C# nyelven írtam. A program értelmezi az NMEA üzeneteket, összepárosítja az egyes vevőktől érkező blokkokat és az így kapott adatokból

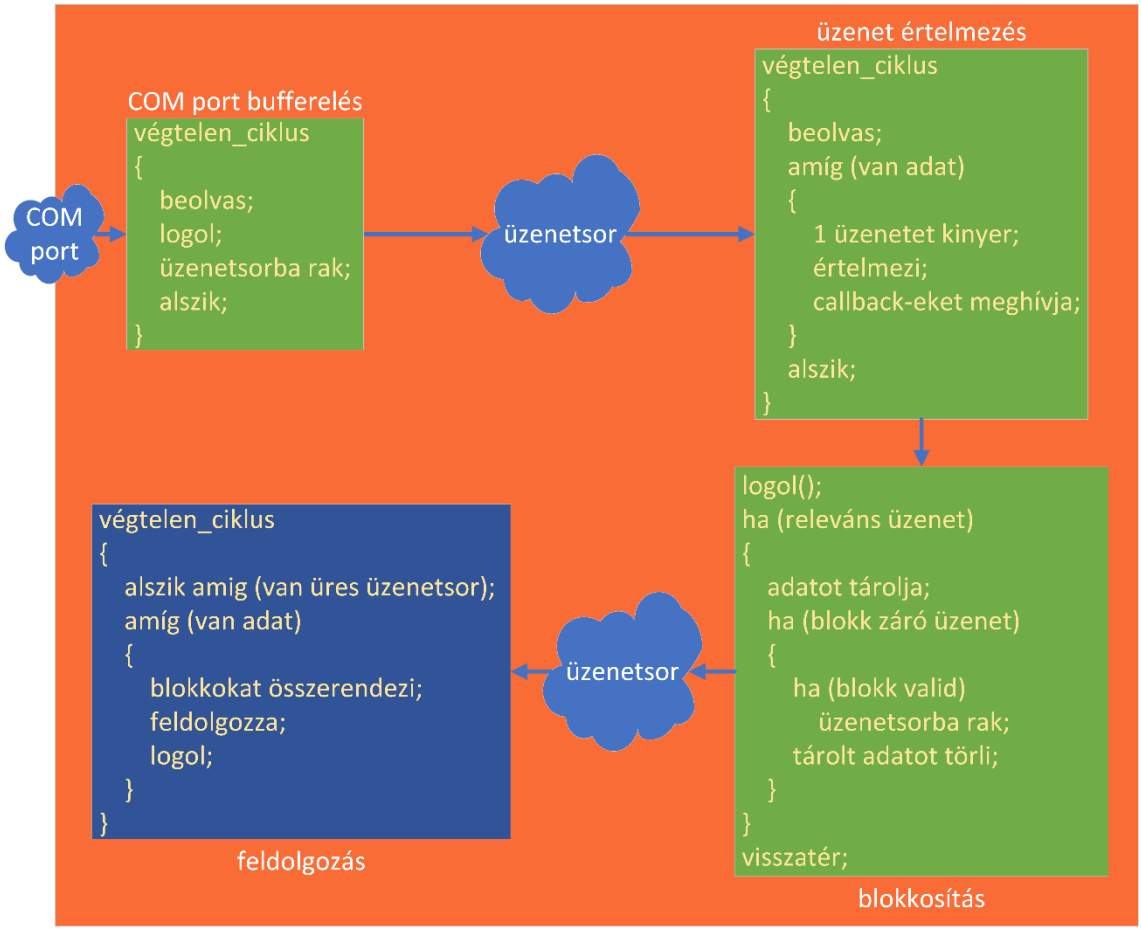
[2] λ_0: a venni kívánt adás szabadtéri hullámhossza

kiszámolja a mérés szempontjából érdekes értékeket; a program bizonyos kapott adatokból egyrész maximumot és átlagot számol, másrészt a $f(ref, x) = \begin{cases} 1, \, ha \, ref = 0 \\ sgn(x - ref) \cdot \left|\frac{x-ref}{x}\right|, \, egyébként \end{cases}$ függvénybe behelyettesítve kiszámolja a vevők adatainak eltérését az egyik, a mérés elején referenciának kijelölt, vevő adataihoz képest. Mind az eredmények, mind az esetleges hibák és természetesen a soros portokról begyűjtött nyers adatok is tárolásra kerülnek a háttértáron szöveges állományként.

A program összesen 9 féle fájlba ír, ezek a következők:

- nyers log: a soros portról begyűjtött karaktereket tartalmazza egy az egyben, minden csatlakoztatott vevőhöz tartozik egy
- felhasznált NMEA üzenetek logja: az azonosított NMEA üzeneteket tartalmazza, minden csatlakoztatott vevőhöz tartozik egy
- Setup.xml: egy XML fájl mely az adott mérési összeállítás lényeges adatait tartalmazza: melyik szoftver verzió végezte a mérést, melyik számítógépen, melyik vevő melyik soros porthoz csatlakozott, melyik vevő melyik sorszámú teszt kártán volt, a teszt kártya fizikai méretei, melyik vevő lett referenciának kijelölve
- hiba fájl: a hibákat és azok természetét tartalmazza, hiba például, ha valamelyik vevőtől nem jön meg, vagy nem jól jön meg az egyik másodperchez tartozó NMEA blokk, vagy ha az adott másodpercben nem volt képes a vevő a pontos idő meghatározására
- heartBeat.log: a program másodpercenként beleír ja a pontos időt és a dátumot; elsősorban debuggolási célokat szolgál
- adat fájl: a begyűjtött adatokat és azokat a számított értékeket tartalmazza, amit az *eltérés fájl* nem, egy sor egy adott másodperchez tartozó adatokat tartalmazza, ez a fájl az összes vevő adtát egyben tartalmazza
- eltérés fájl: ez a fájl tartalmazza a vevők egyes adatainak a referencia vevőhöz képest számolt eltérését, egy sor egy adott másodperchez tartozó adatokat tartalmazza, ez a fájl az összes vevőhöz tartozó adatot egyben tartalmazza

- pozíció fájl: ez a fájl tartalmazza azt, hogy az egyes vevők melyik másodpercben mit mondtak pozícióként (GPS koordinátaként), egy sor egy adott másodperchez tartozó adatokat tartalmazza, ez a fájl az összes vevőhöz tartozó adatot egyben tartalmazza
- log fájl: ez a fájl *az adat fájl*, az *eltérés fájl* és a *pozíció fájl* adatait plusz a felhasznált NMEA üzeneteket és egyéb az NMEA blokkokból kinyert adatokat egyben tartalmazza másodperces bontásban, viszont XML formátumban; ez lehetővé teszi a program könnyű debuggolását



3.4. ábra: a számítógépes adatgyűjtő és feldolgozó program működése

A 3.4-es ábrán látható a program felépítése. Minden vevőhöz tartozó soros portot egy külön OS szál kezel, amely ciklikusan kiolvassa a rendelkezésre álló adatot a portból, át adja azt egy logger osztálynak (mely külön szálon elvégzi a hátértárra írást), belerakja az adatot egy üzenetsorba majd alszik egy ideig. Az üzentsorból az adott vevőhöz tartozó üzenetértelmező folyamat fogyasztja az üzeneteket; amig ki tud rakni egy teljes üzenetet addig egyesével kinyeri és értelmezi őket. Az értelmezett üzenetben foglaltakat egy az

üzenetnek megfelelő osztályba rakja és meghívja a blokkosításért felelős osztály callback függvényét, aminek átadja az üzenetből létrehozott osztályt. A blokkosító osztály felelős azért, hogy az egymás után jövő NMEA üzeneteket a másodperceknek megfelelő blokkokba rendezze (ld. GNSS vevők bemutatása). A callback a neki átadott üzenetet egy logger osztály segítségével nyers formában (NMEA karaktersor) a hátértárra rögzíti. Ezután, ha az üzenet a mérés szempontjából releváns, akkor a lényegi információkat egy, a blokkok adatainak tárolására készített osztályban eltárolja. Ha az épp feldolgozott üzenet a blokk utolsó üzenete volt, akkor leellenőrzi, hogy teljes és konzisztens-e a blokk és ha igen akkor a kitöltött osztályt belehelyezi egy üzenetsorba. Végül, ha az üzenet a blokk végét jelző üzenet volt, akkor új osztály példányt hoz létre a következő blokknak.

A blokkosítók által töltött üzenetsorokat összesen egy darab feldolgozó folyamat fogyasztja. Ez összevárja a különböző vevőktől az azonos másodpercekhez tartozó blokkokat majd feldolgozza őket: kiszámolja, amit ki kell. Az eredményeket a megfelelő logfájlokba beleírja és ha egy másodpercben valamelyik vevő nem tudott adatot szolgáltatni akkor azt a hiba fájlban rögzíti. Ez a szál felelős a mérés elindításáért és leállításáért is (felhasználó utasítására).

A program kimenetét (log fájlok) a következő minden másodperchez és vevőhöz[3] megállapított adatok képzik:

- Position Dilution of Precision
- Horizontal Dilution of Precision
- Vertical Dilution of Precision
- látott műholdak száma
- látott GPS műholdak száma
- látott GPS műholdak listája
- látott GLONASS műholdak száma
- látott GLONASS műholdak listája
- a megoldásban használt műholdak száma

[3] A referenciavevőhöz, értelemszerűen, nem kerülnek kiszámításra az eltérés értékek.

- a megoldásban használt GPS műholdak száma
- a megoldásban használt GPS műholdak listája
- a megoldásban használt GLONASS műholdak száma
- a megoldásban használt GLONASS műholdak listája
- a látott műholdak adatai
    - Pseudorandom Noise (PRN) szám
    - magasság (eleváció)
    - irányszög (azimut)
    - C/N_0
- pozíció
    - szélesség
    - hosszúság
    - tengerszint feletti magaság
- maximális C/N_0
- maximális C/N_0 csak a GPS műholdakat nézve
- maximális C/N_0 csak a GLONASS műholdakat nézve
- átlagos C/N_0
- átlagos C/N_0 csak a GPS műholdakat nézve
- átlagos C/N_0 csak a GLONASS műholdakat nézve
- megoldás minősége
- eltérés értékek: a pozíción, a műhold listán és a műholdak pozíció adatain kívül mindegyik adat összehasonlításra kerül a referencia vevővel, illetve a pozíciókból is kiszámításra kerül a referencia pozíciójához képesti ofszet

3.5 MATLAB-os vizualizálás és átlagolás

A MATLAB szkriptet úgy írtam meg, hogy a konzolból egyetlen függvény meghívásával egy egész mérés feldolgozható. A MATLAB szkripteket tartalmazó

könyvtárat csak a mérések mappáit tartalmazó könyvtárba kell másolni és aztán a setUp szkript egyszeri lefuttatása után a mérésekre lefuttatható a process függvény. Ez a függvény első paraméterként a mérésben használt vevők számát várja, másodikként a mérés nevét (ami alapján a mérés fájljait automatikusan megtalálja a nomenklatúra alapján), azután a mérésben használt vevők nevét a referencia vevővel kezdve és végül az átlagolási ablak hosszát mintaszámban, tehát másodpercben megadva; a program (a pozíción kívül) az adatsorokon a *movmean* függvény segítségével csúszó ablakos átlagolást végez. Ahol az egyik vevőtől valamiért nem ál rendelkezésre adat, oda a használt fájl importáló szkript NaN értéket rak. Ezért a *movmean* függvény az *omitnan* opcióval kerül meghívásra; így a kimaradt adatok nélkül kerülnek kiszámolásra az átlagok.

A process függvény a kiátlagolt adatsorokból grafikonokat készít, amiket PNG-ként és FIG-ként el is ment automatikusan az adott mérés mappáján belül létrehozott Result könyvtárba. A grafikonokon a különböző vevőkhöz különböző színű vonalak/pontok tartoznak; a szkriptek írása közben figyelmet fordítottam arra, hogy a különböző grafikonokon/képeken ezek a színek ne keveredjenek: egy adott vevőnek mindegyik ábrán ugyan az a színe. Azért, hogy az ábrákat ne keljen mindig kézzel teljes képernyőssé tenni a jobb láthatóság érdekében, a szkript egy open-source winowsAPI portolás segítségével ezt automatikusan megteszi [16].

A process függvény a következő 11 ábrát állítja elő a gyűjtött adatokból és a számolt relatív eltérésekből (amiket százalékként ábrázol) összesen 37 grafikonnal:

- pozíciók 2D-s szórás diagramja
- pozíciók 2D-s útvonal diagramja
- pozíciók 3D-s szórás diagramja
- pozíciók 3D-s útvonal diagramja
- a vevők pozíció ofszetje a referencia vevőhöz képest
- Dilution of Precision értékekhez tartozó diagrammok
    - PDOP értékek
    - PDOP eltérések
    - VDOP értékek

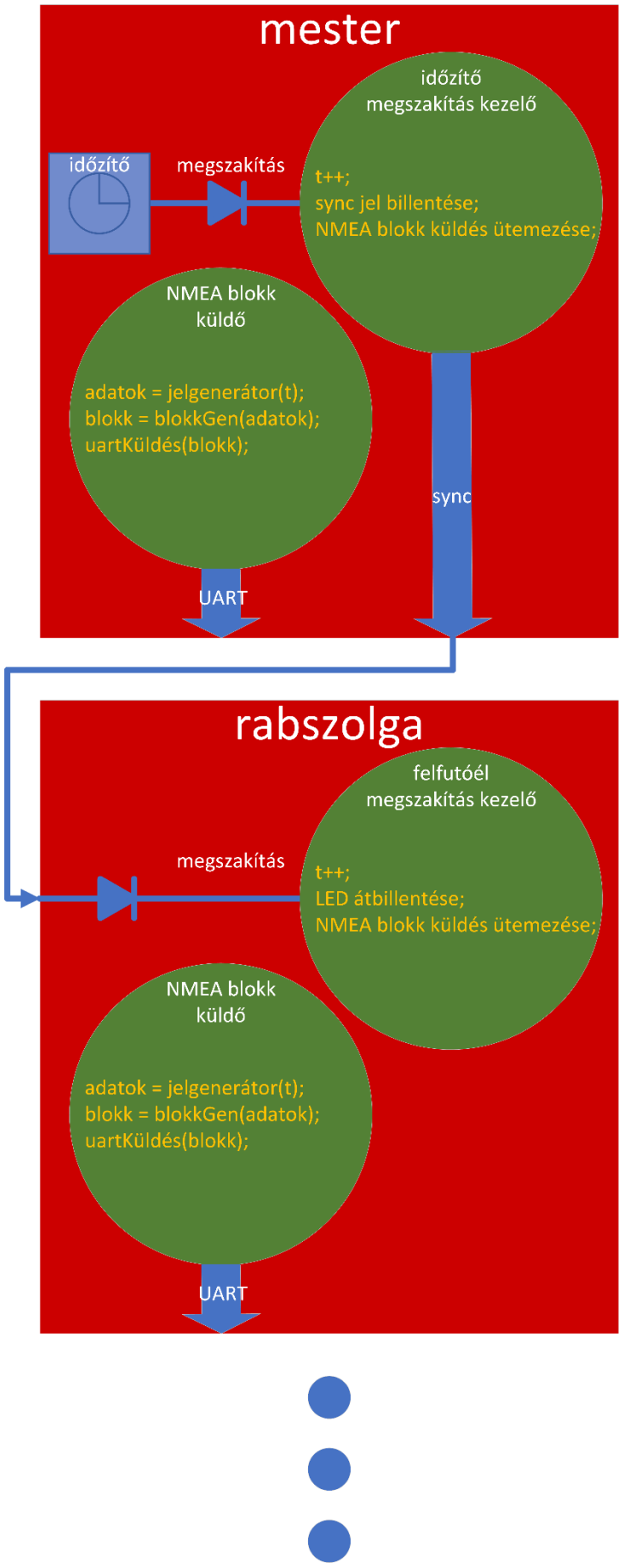
    - VDOP eltérések
    - HDOP értékek
    - HDOP eltérések
- látott és használt műholdak száma
    - látott műholdak száma
    - látott GPS műholdak száma
    - látott GLONASS műholdak száma
    - megoldáshoz használt műholdak száma
    - megoldáshoz használt GPS műholdak száma
    - megoldáshoz használt GLONASS műholdak száma
- látott és használt műholdak számnak eltérései
    - látott műholdak számára nézve
    - látott GPS műholdak számára nézve
    - látott GLONASS műholdak számára nézve
    - megoldáshoz használt műholdak számára nézve
    - megoldáshoz használt GPS műholdak számára nézve
    - megoldáshoz használt GLONASS műholdak számára nézve
- SNR értékek (ezek az L86 vevő esetében a C/N_0 értékeket jelentik)
    - műholdak maximum SNR-ja
    - GPS műholdak maximum SNR-ja
    - GLONASS műholdak maximum SNR-ja
    - műholdak átlagos SNR-ja
    - GPS műholdak átlagos SNR-ja
    - GLONASS műholdak átlagos SNR-ja
- SNR értékek eltérései
    - műholdak maximum SNR-jára nézve

    - GPS műholdak maximum SNR-jára nézve
    - GLONASS műholdak maximum SNR-jára nézve
    - műholdak átlagos SNR-jára nézve
    - GPS műholdak átlagos SNR-jára nézve
    - GLONASS műholdak átlagos SNR-jára nézve
- látott és használt műholdak SNR értékeinek átlagos eltérésé: ezek az értékek úgy kerülnek kiszámításra, hogy minden műholdra kiszámolja a program, hogy az adott vevő által érzékelt SNR mekkora mértékben tér el a referencia vevő által ugyan annál a műholdnál érzékelt értéktől, majd veszi ezen relatív eltérés értékek átlagát
    - látott műholdak átlagos eltérése
    - használt műholdak átlagos eltérése

3.6 Validáció

A mérőrendszer által megvalósított mérőcsatornát természetesen validáltam is. Ezt úgy tettem, hogy Texas Instruments-es fejlesztőkártyákkal (EK-TM4C1294XL) szimuláltam a GNSS vevőkkel szerelt tesztkártyákat és az USB-UART átalakítót (a fejlesztőkártyákon, a debugger áramkörben, van beépített USB-UART átalakító). Így én tudtam kialakítani az NMEA üzenetekből kirajzolódó jelalakokat és amennyire az UART és az USB sávszélessége (alapesetben, valós vevőknél az UART bitsebessége 9600 bps [9] [11], de a szimulációnál ez 115,2 kbps volt) engedte az időt is fel tudtam gyorsítani; végül 10-szeres gyorsítással végeztem a validációt.

A folyamat úgy működött, hogy az épp ellenőrizni kívánt jelnek valamilyen jellegzetes jelalakot választottam, pl. szinusz, háromszög, négyszög és aztán a MATLAB által kirajzolt grafikonon ellenőriztem, hogy alakhűen jött-e át a jel és hogy rendbe van-e a periódus ideje, amplitúdója és ofszetje (DC komponense). Minden jel csatornáját sikerül validálnom.



3.5. ábra: a validációhoz használt szimulációs rendszer

A szimulációs rendszer a 3.5-ös ábrán látható. A rendszer egy mester és több rabszolga egységből/kártyából áll. Mindegyik kártya C nyelven írt TI-RTOS-ra épülő programot futtat. A rendszer működése determinisztikus, azaz reszet után minid ugyan azokat az üzeneteket produkálja. A mester programja úgy működik, hogy egy időzítő adott időközönként megszakítást generál. A megszakítás kezelő függvény pedig egy globális változót inkrementál (t, reszet után természetesen 0-ról indul), egy I/O lábat átbillent (sync jel), majd ütemezi a küldő függvényt egy TI-RTOS rendszerhívással. A küldő függvény meghívja a determinisztikus jelgenerátor függvényt a t paraméterrel, ami visszatér az adott „másodperchez" tartozó jel értékkel (PDOP, HDOP, műholdak, pozíció, stb.) egy struktúra formájában. A függvény végül meghívja ezzel a struktúrával a NMEA blokkot az UART-ra kiküldő függvényt, mely létrehozza és elküldi a struktúrában foglalt értékeknek megfelelő NMEA üzeneteket. A rabszolgák ugyan így működnek, csak a megszakítást az egyik I/O lábuk (amire a mester sync jele van kötve) jelszint változása generálja és egy másik lábukat állítják, amin egy a fejlesztőkártyára gyárilag integrált LED található; így könnyen ellenőrizhető, hogy működik-e a rendszer.

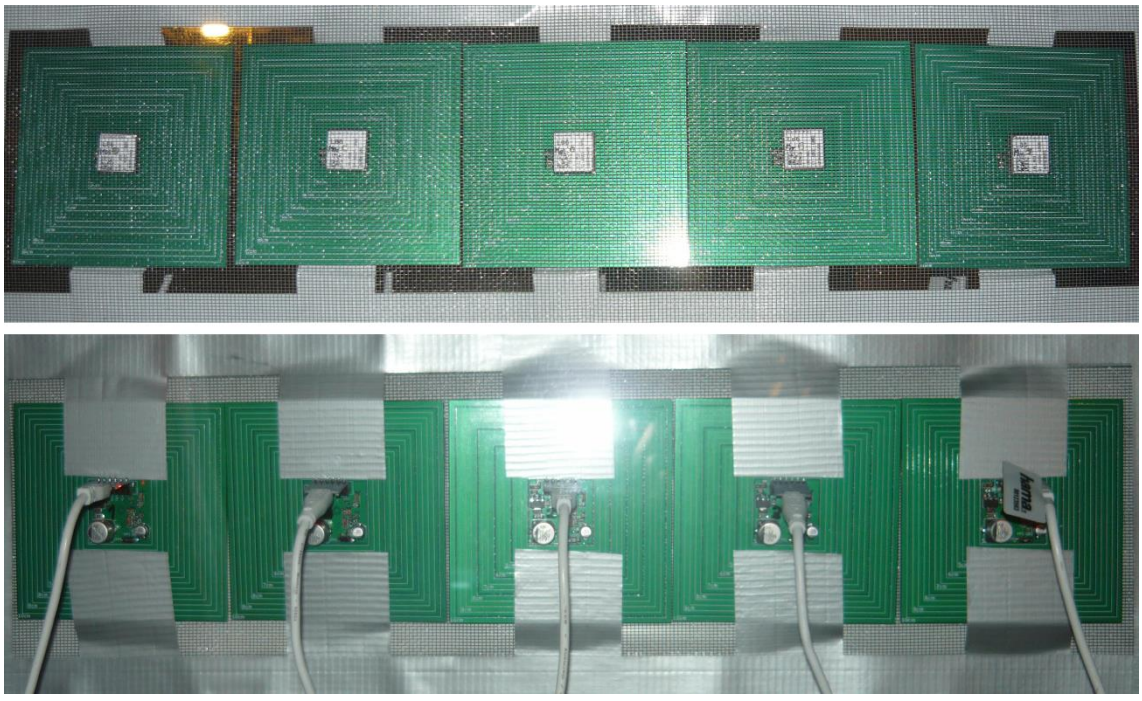
4 Mérési módszer

A mérés a következő kép kerül lebonyolításra; a mért vevők közvetlenül egymás mellé kerülnek kihelyezésre egy jó vételi körülményeket biztosító helyre, így elvileg mindegyik ugyan azt a GNSS adást látja. A mérés kezdetekor mindegyik vevőt hideg indítani kell, hogy ne másítsa meg az eredményt az, hogy az egyes vevők flash-ében más-más kiinduló értékek vannak. Ha nem a vevők firmwareinek az összehasonlítása a cél, hanem maguk a vevők és a földkitöltések összehasonlítása, akkor arra is figyelni kell, hogy a vevők firmwareje azonos legyen (ld. Firmware), hiszen a firmware jelentősen befolyásolhatja egy vevő teljesítményét. Az így összeállított rendszerrel aztán hosszú távú méréseket kell végezni: legalább egy napos, hiszen az ionoszféra a napszakkal változik. [17] [18] [19] [20] Továbbá, a hosszútávú méréssel az különböző vevőkre esetlegesen egyenetlenül ható interferencia jelenségek is kiküszöbölődnek, hiszen ahogy a műholdak vándorolnak, úgy az esetlegesen létrejövő interferencia pontok is mozognak; tehát nem lehet az, hogy a közvetlen egymás mellett lévő vevők teljesítményében észlelt hosszútávú azonos jellegű eltérést interferencia okozza. A hosszú mérés során arra is figyelni kell, hogy a Windows-os PC megfelelően legyen felkonfigurálva és ne induljon újra magától mikor ő épp úgy gondolja; az úgynevezett *Windows Customer Experience Improvement Program*-ot például ajánlatos kikapcsolni, ugyan is hajlamos az éjszaka folyamán újraindítani a gépet. Ráadásul még az újraindítás elött futó fejlesztőkörnyezetet is újra megnyitja abban az állapotban, amiben bezárta ezért a fejlesztő csak annyit tapasztal, hogy a debug módban futó programja misztikusan bezáródott és nem jön rá a turpisságra amíg a sokadik nekifutásra is makacsul (az éjszaka során jelentkező) hiba okát el nem kezdi keresni a Windows rendszer logjában.

A fentiek szerint összeállított mérés lényege, hogy az egyes teszt kártyák teljesítménye egymással összehasonlítható. Így, ha meg szeretnénk határozni, hogy hogyan befolyásolja a vételt a patch antenna földkitöltése, vagy firmware, akkor nincs szükség műhold szimulációra; elég egy-két referencia vevő adataihoz hasonlítani az egy-két vizsgált vevő adatait (amennyiben az adatok azonos mérésből származnak). A földkitöltés hatása vizsgálható például úgy, hogy kiteszünk két referencia vevőt 10 cm-es földkitöltéssel és 3 vevőt kisebb földkitöltéssel. Az e mérés során gyűjtött SNR (L86-os vevő esetén ez C/N_0-t takar) értékeket aztán direktben össze lehet hasonlítani egymással; meg lehet mondani, hogy a kisebb földfelület mennyivel rontotta a vételt.

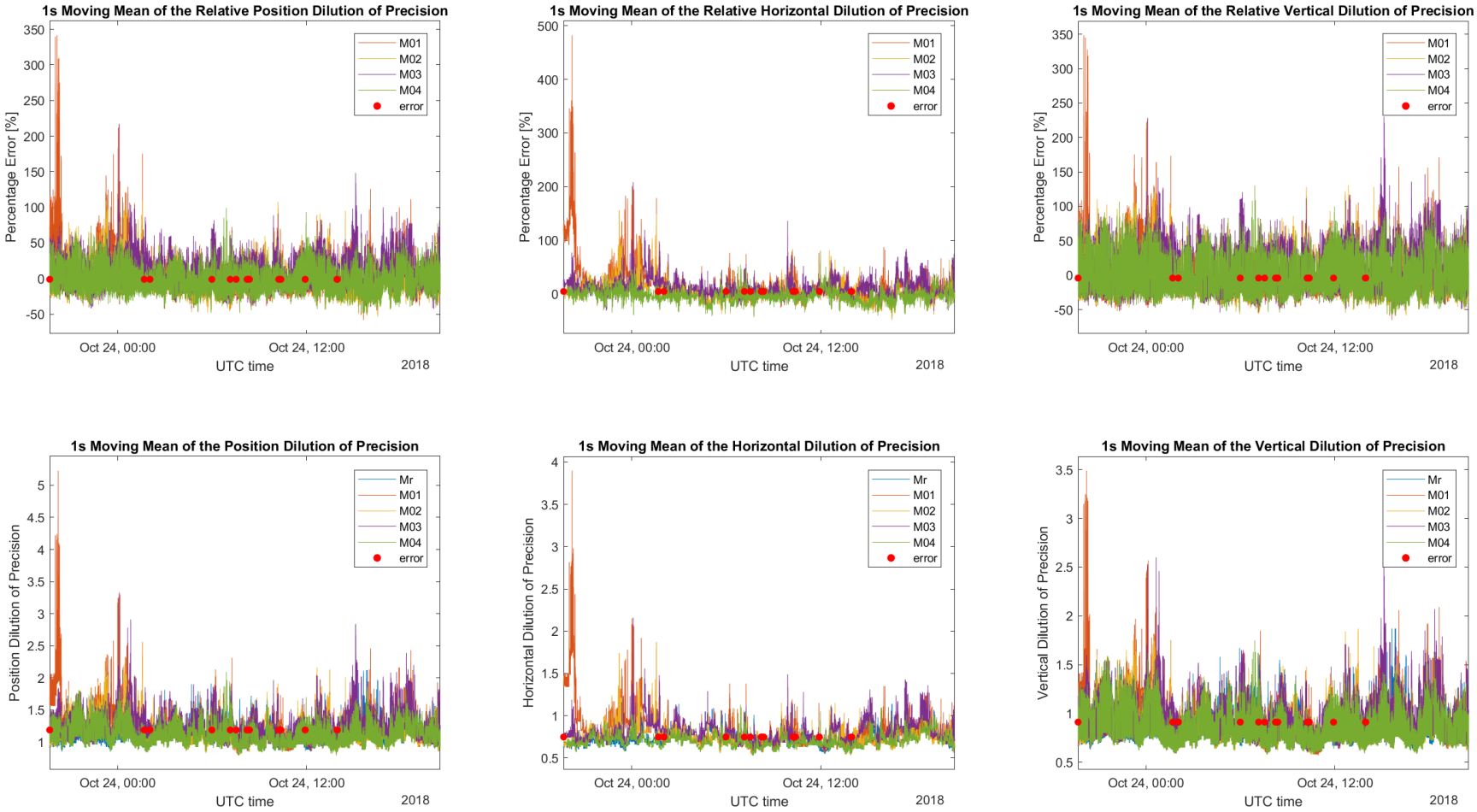
5 Első mérési eredmények

Eddig egy sikeres hosszútávú mérést csináltam. Ennek eredményeit fogom az alábbiakban ismertetni. A mérés hossza ~1 nap (~25 óra) és 5 kártya vett benne részt, melyek mind ugyan azzal a firmware-rel voltak töltve. A kártyákat a 3. emeleti lakásom nyugati tájolású ablakába raktam az 5.1-es ábrán látható módon. A vevőket az ablak üvegszálas szúnyoghálójára ragasztottam fel erős ragasztószalaggal. A mérés megkezdése elött az össze vevőt hideg indítottam, azaz egy tejes reszetet csináltam rajtuk. Ezután vártam egy órát, hogy a vevők beálljanak és a tranziens jelenségek lecsengjenek; így ezek nem zavartak bele a mérésbe. Ezek után indítottam a mérést mely adatgyűjtő egysége egy DELL Latitude E7240 laptop volt. A laptophoz 4 vevő egy USB HUB-on keresztül csatlakozott az 5. vevő pedig direktben az egyik USB porthoz. A mérés UTC idő szerint 2018/10/23 21:39:00-tól, 2018/10/24 22:30:36-ig tartott. A mérésben résztvevő mind az 5 kártya 10 x 10 cm-es volt. A mérés referencia vevőjeként az Mr nevű vevőt választottam.

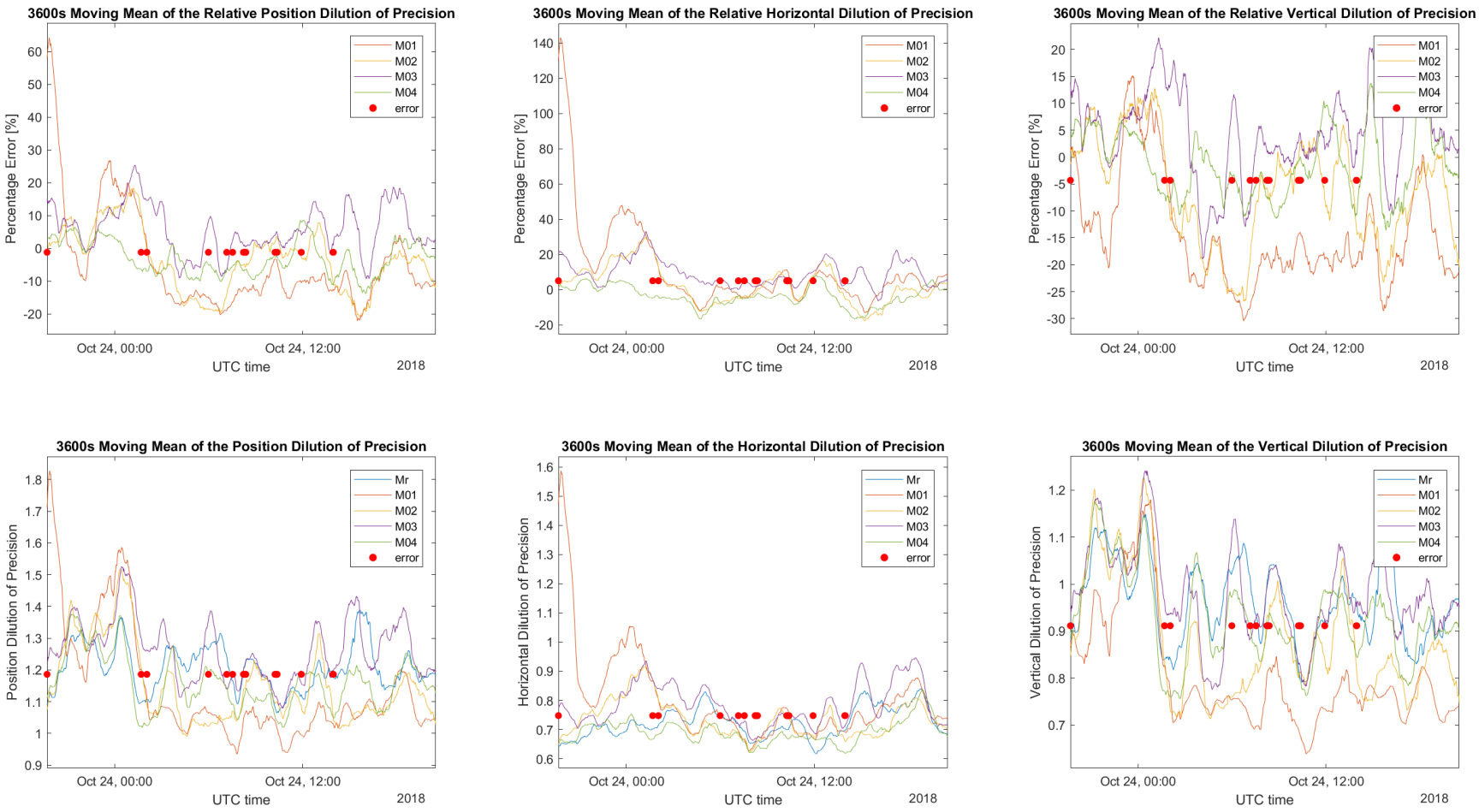


5.1. ábra: a Quectel L86 GNSS vevők a lakásom nyugati tájolású ablakában, a szunyoghálóra duct tape-el felszerelve, az ablak így bezárható volt, és a vevők is védve voltak a leeresztett műanyag redőnnyel (mely a GNSS jelekkel nem interferál)

Ahogy az majd az alábbi, a MATLAB-os szkripttel rajzolt grafikonokon is látszik, viszonylag kevés hiba történt a mérés során, tehát a vizsgált vevők meglehetősen üzembiztosak.



5.2. ábra: a szűretlen Dilution of Precision értékek és a többi vevő az Mr vevőtől való százalékos eltérése



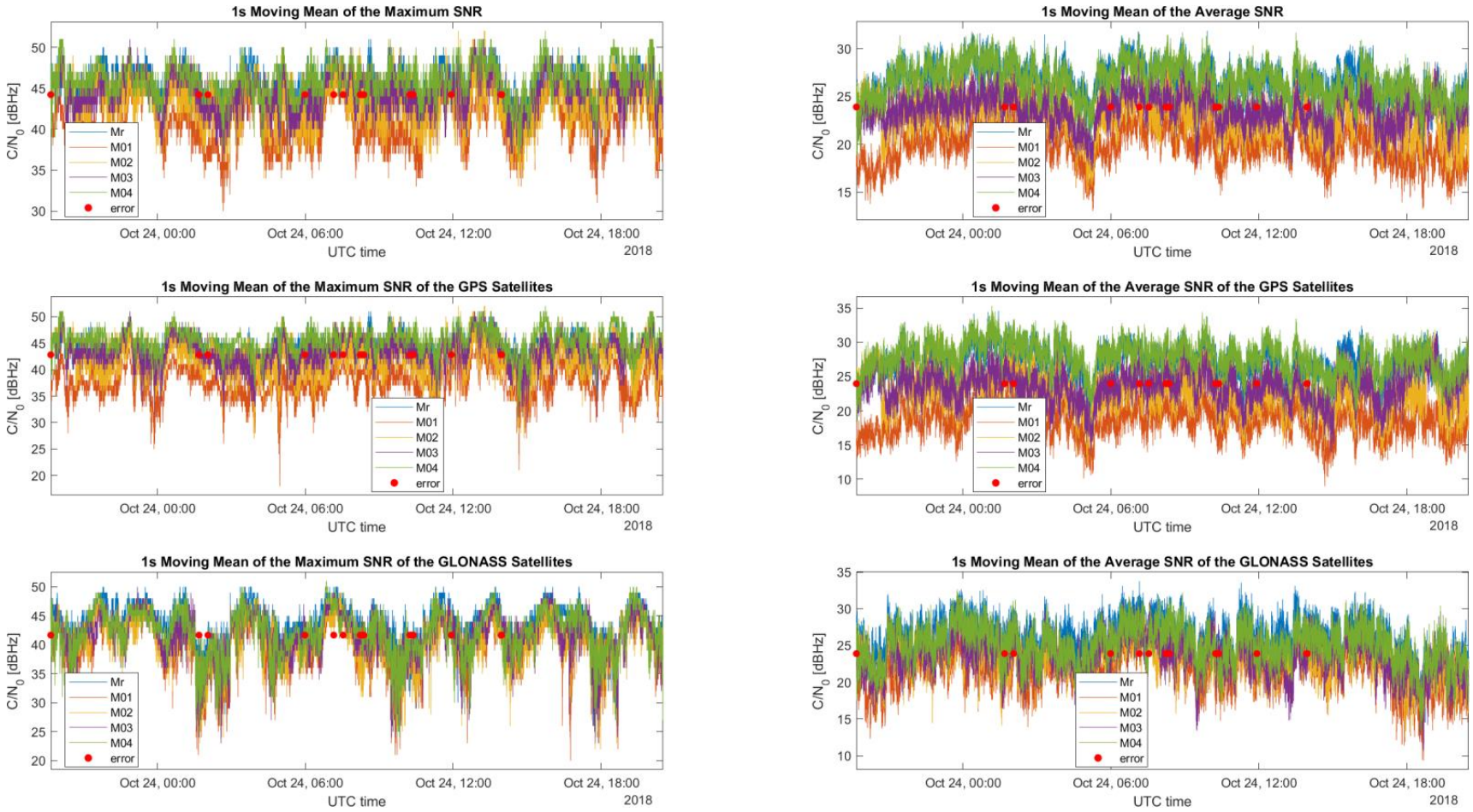
5.3. ábra: az 1 órás (3600 másodperces) mozgóátlaggal szűrt Dilution of Precision értékek és a többi vevő az Mr vevőtől való százalékos eltérése

5.1 Dilution of Precision értékek

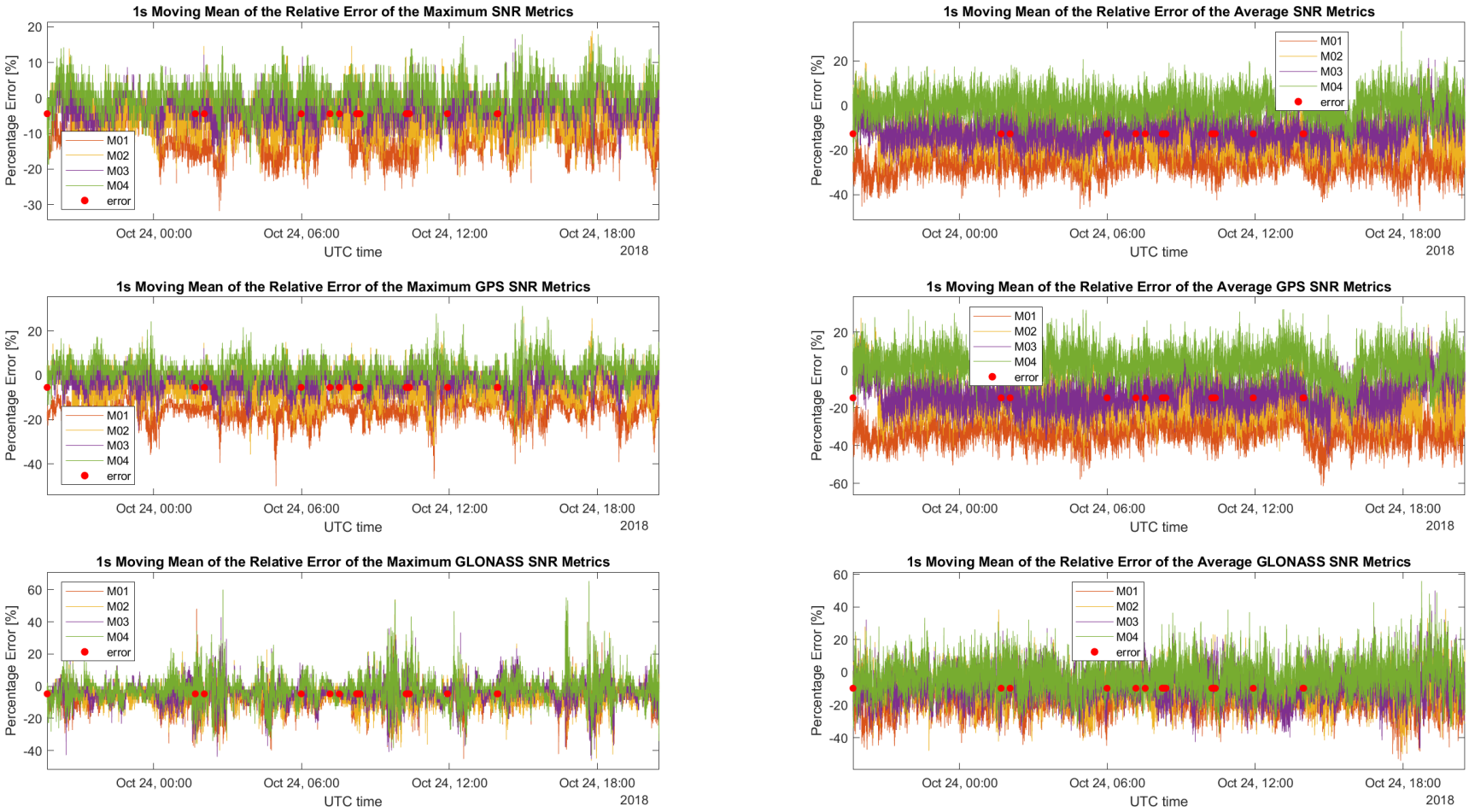
Az 5.2-es ábra és az 5.3-as ábra mutatja Dilution of Precision értékeket. Az ábrákon a piros pöttyök a hibákat jelölik: azokon a helyeken kimarad egy másodper vagy mert valamelyik vevő nem tudott adatot szolgáltatni, vagy mert az üzenet nem jól jött át. Az 5.2-es ábra szűretlenül mutatja az adatokat az 5.3-as ábrán az adatsor meg lett szűrve egy 1 órás ablakszélességű mozgó átlagolással. Látható, hogy az M01-es vevő rosszabbul teljesít, mint a többi és hogy az Mr és az M04 adja a legjobb eredményt. Érdekes viszont, hogy a VDOP értékeket tekintve M01 a legjobb.

5.2 C/N_0 értékek, avagy a vétel minősége

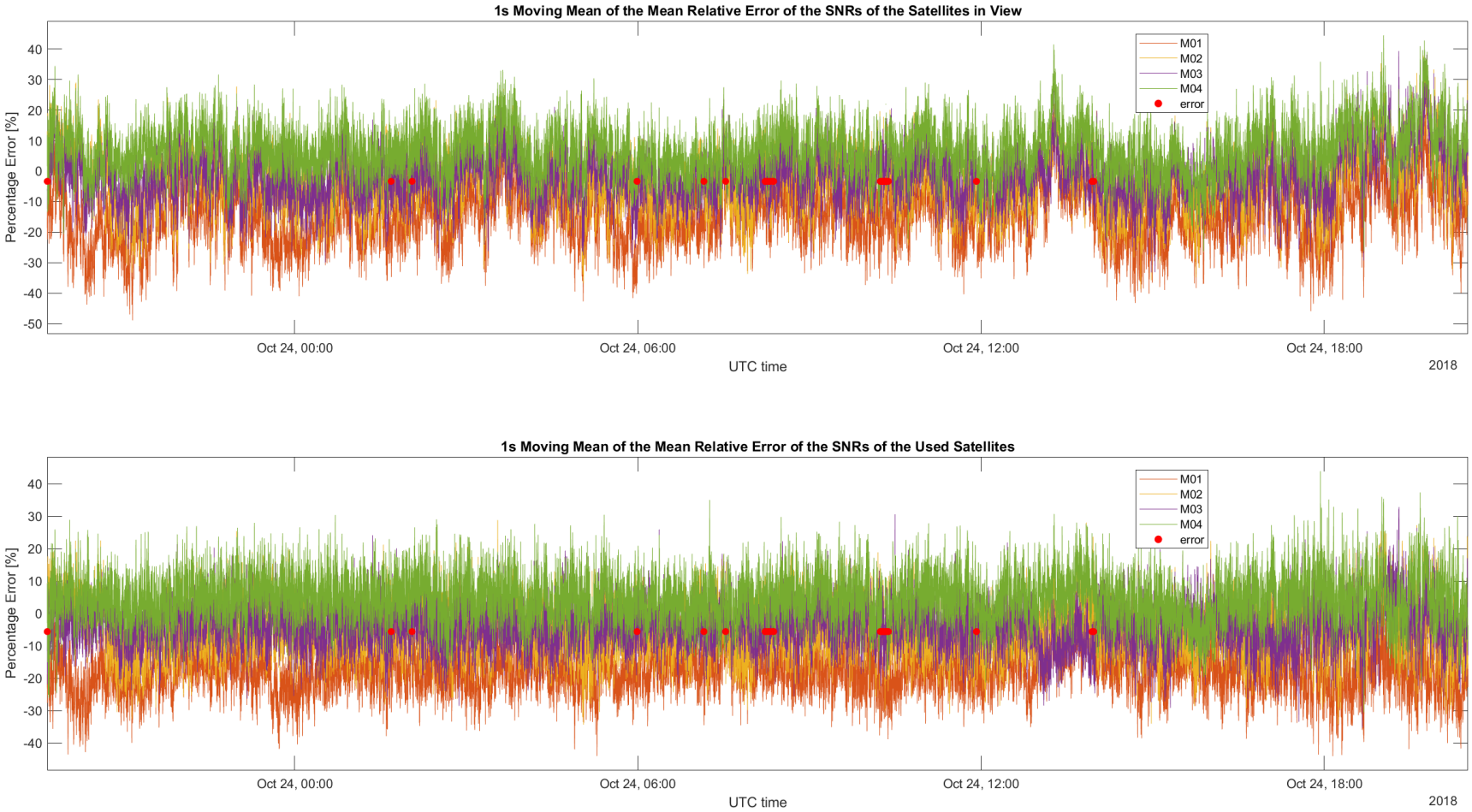
Az 5.4-es ábra, az 5.5-ös ábra, 5.6-os ábra, az 5.7-es ábra, az 5.8-as és az 5.9-es ábra a C/N_0 értékeket és azok eltéréseit mutatja a mérés elején referenciának választott Mr vevő értékeitől. Az ábrákból kitűnik, hogy az Mr és az M04 vevők szignifikánsan jobbak a többi vevőnél, konzisztensen sok decibellel magasabbak a C/N_0 értékeik. Az 5.6-os ábrából és az is 5.9-es ábrából az is látszik, hogy habár az Mr és az M04 vevő egyformának tűnik, a műholdak átlagos SNR értékeit nézve M04 mégiscsak konzisztensen kb. mintegy 5%-al jobb mint Mr. Az is látszik, hogy M02 és M03 sem olyan rosszak; az ő teljesítményük nagyjából megegyezik. M01 viszont végig sokkal rosszabb, mint a 3 másik vevő.



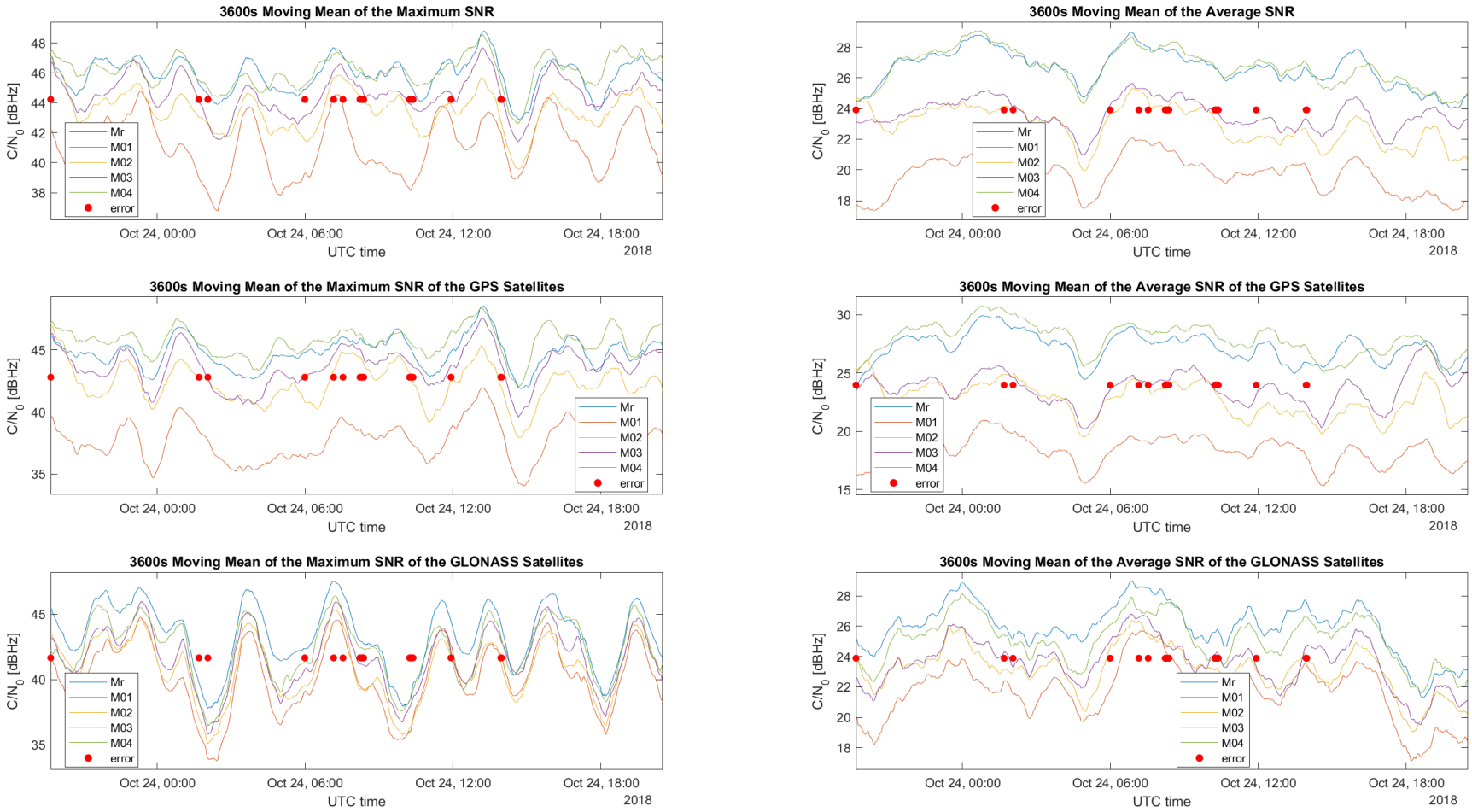
5.4. ábra: a szűretlen C/N_0 értékek dBHz-ben ábrázolva



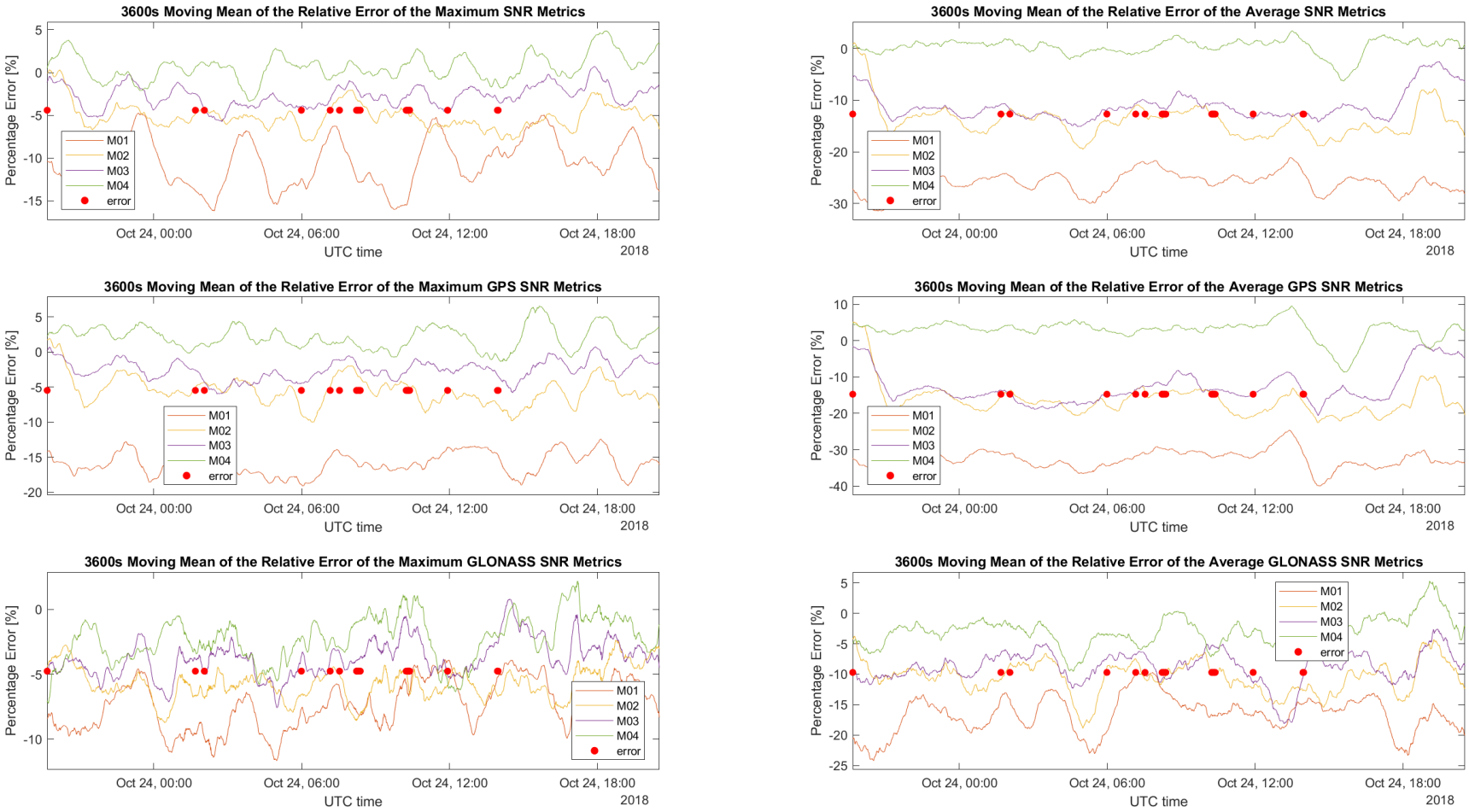
5.5. ábra: C/N_0 értékek Mr-től való szűretlen százalékos eltérése vevőnként



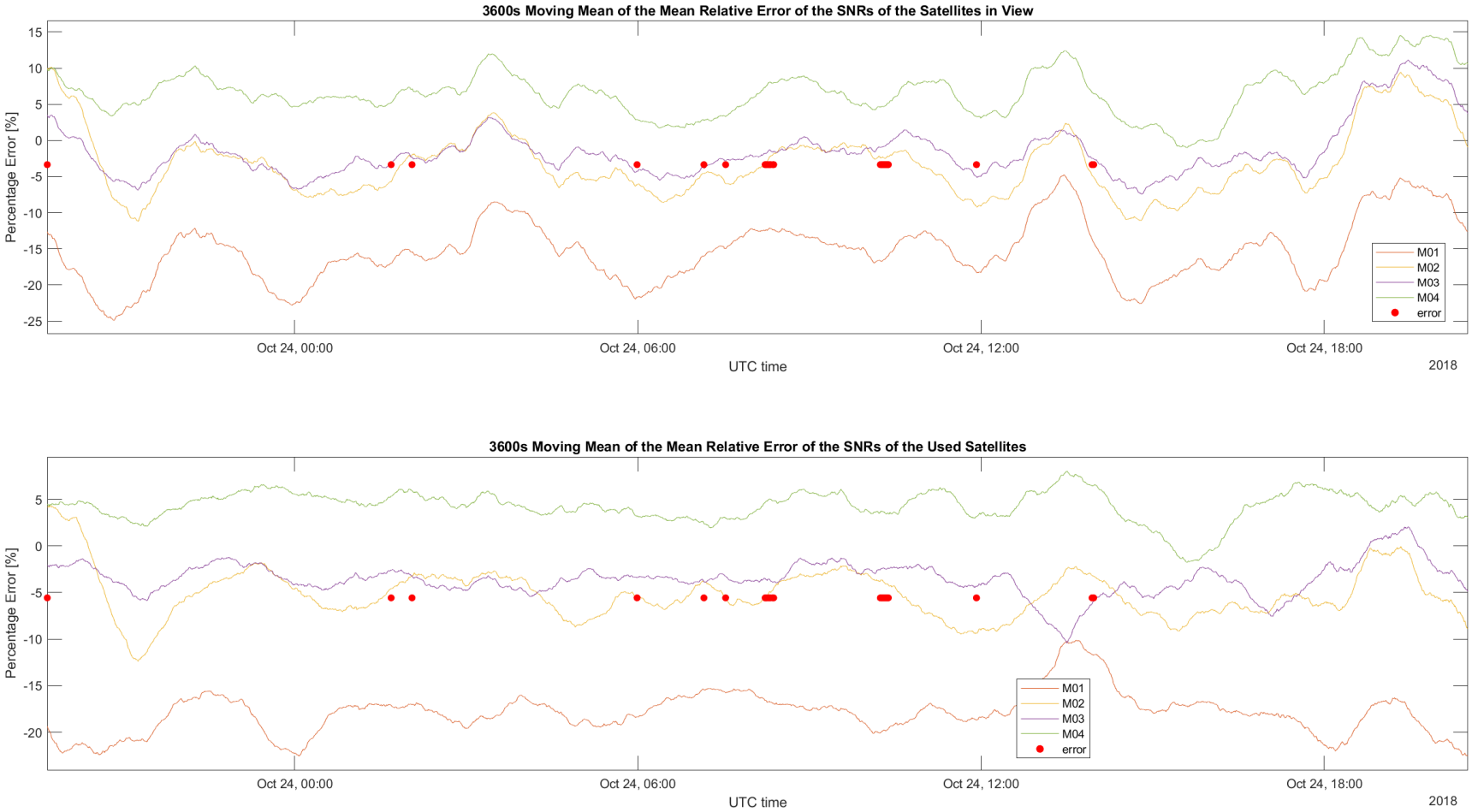
5.6. ábra: C/N_0 értékek Mr értékeitől való szűretlen műholdankénti átlagos százalékos eltérése vevőnként



5.7. ábra: az 1 órás (3600 másodperces) mozgóátlaggal szűrt C/N_0 értékek dBHz-ben ábrázolva



5.8. ábra: C/N₀ értékek Mr-től való az 1 órás (3600 másodperces) mozgóátlaggal szűrt százalékos eltérése vevőnként



5.9. ábra: C/N₀ értékek Mr értékeitől való 1 órás (3600 másodperces) mozgóátlaggal szűrt műholdankénti átlagos százalékos eltérése vevőnként

5.3 Konklúzió

A mérés konklúziójakéként megállapítható, hogy a vevők teljesítménye a teljesen azonos körülmények ellenére is nagyban eltér egymástól. Ez különösen meglepő annak fényben, hogy nem csak hogy ugyan olyan vevőkről van szó melyeken ugyan az a firmware fut, de a vevők valószínűleg ugyan abból a batchből is származnak, mert egymás után jöttek a szalagban, amiben szállították őket. Ez azt jelenti, hogy a földkitöltés hatását vizsgálni kívánó további mérések során figyelembe kell tehát majd venni azt is, hogy a vevők teljesítménye hogyan viszonylott egymáshoz azonos földkitöltés mellett. További méréseket lehetne esetleg végezni úgy is, hogy a vevők sorrendjét összecserélem az ablakban, hogy biztos legyen, az eltérést nem a vevők fizikai elhelyezkedése okozza.

A mérés megerősítéssel szolgált továbbá arra nézve, hogy a földkitöltésnek a 10 cm-es méret jó kiindulási pont, hiszen még a legrosszabb vevő is végig üzembiztosan tudott működni, látta a műholdakat. Mivel a Dilution of Precision értékek végig relatíve kicsik voltak (kivéve az M01-es vevőt), kijelenthető az is, hogy a firmware jól végzi a munkáját: okosan gazdálkodok a műholdakkal, jó műholdakat válogat össze az egyenletek megoldásához.

6 Konklúzió és kitekintés

A dolgozat konklúziójaként megállapítható, hogy a mérőrendszer és a mérési módszer működik; már az első mérés során is izgalmas eredményeket sikerült produkálni melyek direkt módon felhasználhatóak. Ezzel a módszerrel például több GNSS vevő közül ki lehet választani a legjobb teljesítményűt és azt beépíteni a *Motiváció*-ban taglalt rendszerbe. A mérés továbbá rávilágított olyan tényekre melyeket a további mérések során figyelembe kell majd venni, mint például, hogy a vevők maguk sem egyformák még ha elvileg azok is.

A közeljövőben fogok több és hosszabb időtartamú mérést is végezni, továbbá a földkitöltés nagyságának hatását is meg fogom vizsgálni úgy, hogy két-három kártyát meghagyva 10 cm-es referenciának a többit elkezdem egyre kisebre vágni és kiértékelem a teljesítményük referencia kártyákhoz képesti romlását; kikísérletezem, hogy mi az a legkisebb földkitöltés amivel a vevők még jól és megbízhatóan működnek.

Köszönetnyilvánítás

A publikációban bemutatott kutatás az Európai Unió támogatásával, az Európai Regionális Fejlesztési Alap társfinanszírozásával valósult meg (EFOP-3.6.2-16-2017-00013).

Irodalomjegyzék

[1] U. Windl, D. Dalton, M. Martinec és D. R. Worley, „The NTP FAQ and HOWTO,” 21 11 2006. [Online]. Available: http://www.ntp.org/ntpfaq/NTP-s-algo.htm#Q-ACCURATE-CLOCK. [Hozzáférés dátuma: 27 10 2018].

[2] IEEE Instrumentation and Measurement Society, *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,* 2008, pp. 1-2.

[3] EtherCAT Technology Group, „Technical Introduction and Overview,” [Online]. Available: https://www.ethercat.org/en/technology.html#1.5. [Hozzáférés dátuma: 27 10 2018].

[4] National Coordination Office for Space-Based Positioning, Navigation, and Timing, „GPS Accuracy,” 5 12 2017. [Online]. Available: https://www.gps.gov/systems/gps/performance/accuracy/. [Hozzáférés dátuma: 27 10 2018].

[5] Á. Hollós és T. Kovácsházy, „Improved Reference Clock Connection Interface for Prototype IEEE 1588 Master Clocks,” in *2018 19th International Carpathian Control Conference*, Szilvásvárad, 2018.

[6] T. Kovácsházy, "Synchronization performance evaluation of reference clock connection methods for IEEE 1588 master clocks," in *Carpathian Control Conference (ICCC), 2015 16th International*, Szilvásvárad, 2015.

[7] Á. Hollós és T. Kovácsházy, „Low Cost Field Test Measurement Method and Prototype Measurement Device Implementation for Timing Accuracy Evaluation of IEEE 1588 Solutions,” in *2018 Workshop on Metrology for Industry 4.0 and IoT*, Brescia, 2018.

[8] Trimble, *Resolution SMT GPS Timing Module User Guide,* 2009.

[9] Quectel, L86 Hardware Design, 2016.

[10] D. DePriest, „NMEA data," [Online]. Available: https://www.gpsinformation.org/dale/nmea.htm. [Hozzáférés dátuma: 27 10 2018].

[11] Quectel, *L86 GNSS Protocol Specification,* 2017.

[12] K. F. Lee és K.-F. Tong, „Microstrip Patch Antennas," in *Handbook of Antenna Technologies*, Singapore, Springer Nature, 2016, pp. 823-824, 846.

[13] M. T. Nguyen, B. Kim, H. Choo és I. Park, „Effects of Ground Plane Size on a Square Microstrip Patch Antenna Designed on a Low-Permittivity Substrate with an Air Gap," in *2010 International Workshop on Antenna Technology*, Lisbon, 2010.

[14] Z. N. Chen, D. Liu, H. Nakano, X. Qing és T. Zwick, Handbook of Antenna Technologies, Singapore: Springer Nature, 2016.

[15] A. Hudson és R. Nelson, Útban a modern fizikához, Budapest: LSI Oktatóközpont, 1994.

[16] Jan, „File Exchange," MathWorks, 16 4 2018. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/31437-windowapi. [Hozzáférés dátuma: 29 10 2018].

[17] A. B. O. Jensen és C. Mitchell, *GNSS and the Ionosphere*, GPS World, 2011.

[18] L. Liu, W. Wang, B. Ning, O. M. Pirog és V. I. Kurkin, „Solar activity variations of the ionospheric peak electron density," *Journal Of Geophysical Research,* %1. kötet111, %1. szám10.1029/2006JA011598, 2006.

[19] M. B. McElroy, „Ionosphere and magnetosphere," Encyclopædia Britannica, inc., 8 8 2012. [Online]. Available:

https://www.britannica.com/science/ionosphere-and-magnetosphere. [Hozzáférés dátuma: 29 10 2018].

[20] J. V. Sickle, „The Ionospheric Effect," John A. Dutton e-Education Institute, College of Earth and Mineral Sciences, The Pennsylvania State University, [Online]. Available: https://www.e-education.psu.edu/geog862/node/1715. [Hozzáférés dátuma: 29 10 2018].

[21] Broadcom Corporation and Cisco Systems, Inc., *Digital Transmission: Carrier-to-Noise Ratio, Signalto-Noise Ratio, and Modulation Error Ratio*, 2012.