BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

ÁLTALÁNOS- ÉS FELSŐGEODÉZIA TANSZÉK

KECSKEMÉTI MÁTÉ ISTVÁN

# DIPLOMAMUNKA

GNSS alkalmazása az automatizált monitoring rendszerekben

Konzulensek:

*Dr. Siki Zoltán*
adjunktus

*Bátyi Tamás*
HUNGEOD Kft.

**BUDAPEST, 2019**

MŰEGYETEM 1782

# DIPLOMAMUNKA FELADAT

| | | | |
|---|---|---|---|
| Név: | **Kecskeméti Máté István** | Neptun kód: | **NX84UC** |
| Képzés: | **Földmérő- és Térinformatikai mérnök mesterszak (MSc)** | Szemeszter: | **2019/20/1** |
| Szakirány: | **Földmérő- és Térinformatikai mérnöki specializáció** | Nyilvánt.sz.: | **MSc-SP-F-003-19/20/1** |
| Cím: | **GNSS alkalmazása az automatizált monitoring rendszerekben** | | |

A feladat leírása:

Mozgás- és deformációvizsgálatok esetén a terepi adatgyűjtésben egyre nagyobb szerepet kapnak az automatizált, robot mérőeszközök, szenzorok. A jelölt feladata az Általános és Felsőgeodézia Tanszéken fejlesztett Ulyxes rendszerhez kapcsolódik, ezen belül elsősorban a GNSS vevők adatainak automatizált gyűjtéséhez és elemzéséhez.

Feladat:Készítsen teszt alkalmazást az Ulyxes rendszerben a GNSS vevők alkalmazására. Tegyen javaslatokat a rendszer fejlesztésére a GNSS vevők implementálásával kapcsolatban és vegyen részt a fejlesztések megvalósításában.

Részletes feladatleírás:

1. Szakirodalom alapján az automatizált mozgásvizsgálati rendszerek funkcióinak, eszközeinek bemutatása.
2. Az Ulyxes rendszer általános bemutatása.
3. Teszt környezet kialakítása az Ulyxes rendszerrel, a GNSS vevők adatainak gyűjtésére és elemzésére.
4. A tesztek alapján a tapasztalatok levonása és fejlesztési javaslatok megfogalmazása.
5. Valós körülmények között a továbbfejlesztett rendszerrel mérések végrehajtása.
6. Összehasonlító elemzés a GNSS, robot mérőállomás és digitális szintező és kamera szenzorok alkalmazási lehetőségéről, konklúziók levonása.

A diplomaterv elkészítése során munkanaplót vezetünk, amely rögzíti a konzulens javaslatait és a felhasználható adatokat. A jelöltnek a munka során figyelembe kell vennie a munkanaplóban foglaltakat és azt a diplomatervhez mellékelve be kell adnia a diplomaterv leadásával egyidejűleg.

| Konzulens: | | |
|---|---|---|
| dr. Siki Zoltán | EOAF | 100 % |

| Társ-konzulens(ek): | | |
|---|---|---|
| Bényi László | HUNGEOD Kft. | |
| | | |
| | | |

| | |
|---|---|
| A feladat kiadásának időpontja: | 2019.09.09 |
| A feladat beadásának határideje: | 2019.12.13 |

| | |
|---|---|
| Dr. Barsi Árpád<br>a szakirány részéről | Dr. Dunai László<br>Dékán |

Budapesti Műszaki és Gazdaságtudományi Egyetem
Építőmérnöki Kar
Dékáni Hivatal

1111 Budapest, Műegyetem rkp. 3. K épület I. 28.
Telefon: 463-3531, fax: 463-3530
http://www.epito.bme.hu/

# Konzultációs napló

| Konzultáció időpontja | A konzultáción egyeztetett részfeladatok rövid leírása | Aláírás |
|---|---|---|
| 2019. 09. 05. | Diplomamunka témájának megbeszélése, határidők, források átbeszélése | |
| 2019. 09. 23. | Saját számítógép kompatibilisával kapcsolatos kérdések átbeszélése | |
| 2019. 10. 01. | Bluetooth interfésszel kapcsolatos munkák átbeszélése, megoldása | |
| 2019. 10. 02. | Topcon Hiper II-es GNSS műszer NMEA üzenetek konfigurálásával kapcsolatos munkák átbeszélése | |
| 2019. 10. 15. | Robotplus applikáció működésének átbeszélése | |
| 2019. 10. 17. | Raspberry Pi kamera működésének átbeszélése | |
| 2019. 10. 29. | NMEA üzenetek tesztelése az egyetem udvarán | |
| 2019. 10. 31. | Szabadság híd lehajlásának mérése | |
| 2019. 11. 05. | Raspberry Pi kamera moduljának további tesztelése | |
| 2019. 11. 12. | Hi Target GNSS vevő NMEA kimenet problémáival kapcsolatos konzultáció | |
| 2019. 11. 13. | Hi Target GNSS vevő NMEA kimenet problémáival kapcsolatos konzultáció | |
| 2019. 12. 02. | SQLite adatbázis konfigurálása bemutatása, kamera problémáinak átbeszélése | |
| 2019. 12. 17. | Úszómű mérés feldolgozásának átbeszélése | |
| 2019. 12. 18. | Úszómű mérés GNSS technológiájának feldolgozásával kapcsolatos problémák átbeszélése | |

# Abstract

*Applying GNSS to automated monitoring systems*

Nowadays in the geodesy it is not enough to define an object with 3 coordinates. It is necessary to add one more parameter to this description, the time. Every structure or continent is moving. In the engineering studies we would like to make models about the scale and the direction of the movement.

In the future the automation will get more attention in every part of the engineering sciences, just like in the geodesy. The monotonous tasks will be made by robots or machines instead of human resources. The measuring of deformation and moving can be automated easily because we have to measure the same things every time.

In the beginning of my thesis I will make an overview about the available technologies in the automated monitoring systems. After that i will explain how I joined the works of Ulyxes, which is a monitoring system solution developed by the Department of Geodesy and Surveying at Budapest University of Technology and Economics. The main objective of my thesis is to implement the GNSS technology to this system and make a comparison among the available technologies and to develop them.

# Köszönetnyilvánítás

Köszönettel tartozom elsősorban konzulensemnek, dr. Siki Zoltánnak, aki időt, energiát nem spórolva segédkezett diplomamunkám elkészülésében, emellett minden kérdésemre a lehető legrövidebb idő alatt válaszolva támogatta munkám sikerességét.

Szeretném megköszönni dr. Takács Bencének az Úszómű mérések során nyújtott segítségét, mind a mérések, mind a feldolgozás során.

Továbbá köszönöm szépen a támogatást és segítséget a családomnak és a barátaimnak, akik tanácsaikkal, tudásukkal és tényleges munkájukkal segítették diplomamunkám létrejöttét.

# Nyilatkozat

*Nyilatkozat az önálló munkáról*

Alulírott, Kecskeméti Máté István (NX84UC), a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója, büntetőjogi és fegyelmi felelősségem tudatában kijelentem és sajátkezű aláírásommal igazolom, hogy ezt a diplomamunkát meg nem engedett segítség nélkül, saját magam készítettem, és a diplomamunka feladatomban csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

*Budapest, 2019.*

***szigorló hallgató***

# Tartalomjegyzék

# 1. Bevezetés

Napjainkban a geodéziában nem elég 3 komponens alapján leírnunk egy szerkezet vagy objektum alakjelző pontjait, hanem 4 dimenzióban, az idővel kiegészítve válik az teljessé, ha hosszú távon vizsgáljuk azt. Kis- (épületek, műtárgyak, stb.) és nagy (kontinensek, földrészek, stb.) kiterjedésű alakzatok, objektumok egyaránt végeznek mozgásokat. Ezen mozgások nagyságának és irányának megismerése, azokra modellek felállítása elengedhetetlenné vált a mérnöki és természettudományi életben.

A jelenben és a jövőben hatványozottan igaz, hogy az automatizálás szerepe az élet összes területén növekszik. A monoton feladatokat gépek, robotok végzik már el az emberek helyett, részben vagy teljesen kiváltva azok munkáját. A geodéziában is megjelent erre az igény. A mozgásvizsgálatok könnyen automatizálhatóak, mivel hosszú távon ugyanazokat a feladatokat kell elvégezni.

Diplomamunkám célja bemutatni a jelen pillanatban elérhető technológiákat az automatizált mozgásvizsgálati rendszerekben. Emellett a Budapesti Műszaki és Gazdaságtudományi Egyetem Általános- és Felsőgeodézia tanszékén fejlesztett Ulyxes rendszer munkáiba való bekapcsolódás. A fő irányvonal a GNSS technológia implementálása lenne, ezen felül a már meglévő rendszerek tesztelése és azzal történő összehasonlítása, majd további fejlesztése.

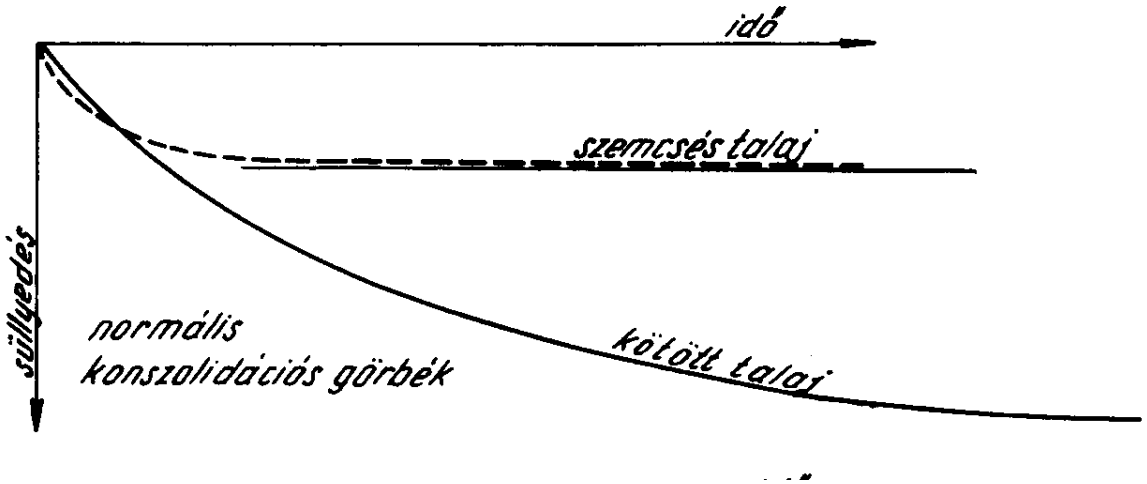
# 2. Napjainkban használt automatizált monitoring rendszerek

## 2.1 Mozgásvizsgálat szerepe az építőmérnöki gyakorlatban és a geotudományokban

A mozgásvizsgálatok során a feladatunk az, hogy meghatározzuk különböző mérnöki létesítmények és objektumok vagy akár egész földterületek, földrészek egyes pontjainak egy adott viszonyítási rendszerben való elmozdulását, valamint az egyes szerkezeti elemek alakváltozását a rájuk ható erők hatására. A különféle deformációk és helyzetváltozások megfigyelése során a célunk sokrétű lehet, azonban főként az elmozdulások előrejelzése és konkrét meghatározása a fő feladat. A következőkben bemutatnám a mozgásvizsgálatok jelentőségét

### *2.1.1 Építőmérnöki gyakorlatban alkalmazott mozgásvizsgálatok [1] [2] [3]*

A süllyedés (Farkas, 2014) az építmények, az alapok függőleges elmozdulása valamely (térben és időben értelmezhető) kezdeti helyzethez képest. Az alapozás megtervezése azt jelenti, hogy igazoljuk, hogy az altalaj nem szenved akkora deformációt, hogy az építményre roncsolóan hat vissza. A süllyedést legnagyobb részét konszolidáció okozza. Ez a folyamat annál lassúbb, minél kisebb a talaj áteresztőképessége és minél nagyobb az összenyomhatósága. A szemcsés és a kötött talajok süllyedésének időbeli lejátszódását az 1. ábra mutatja be.



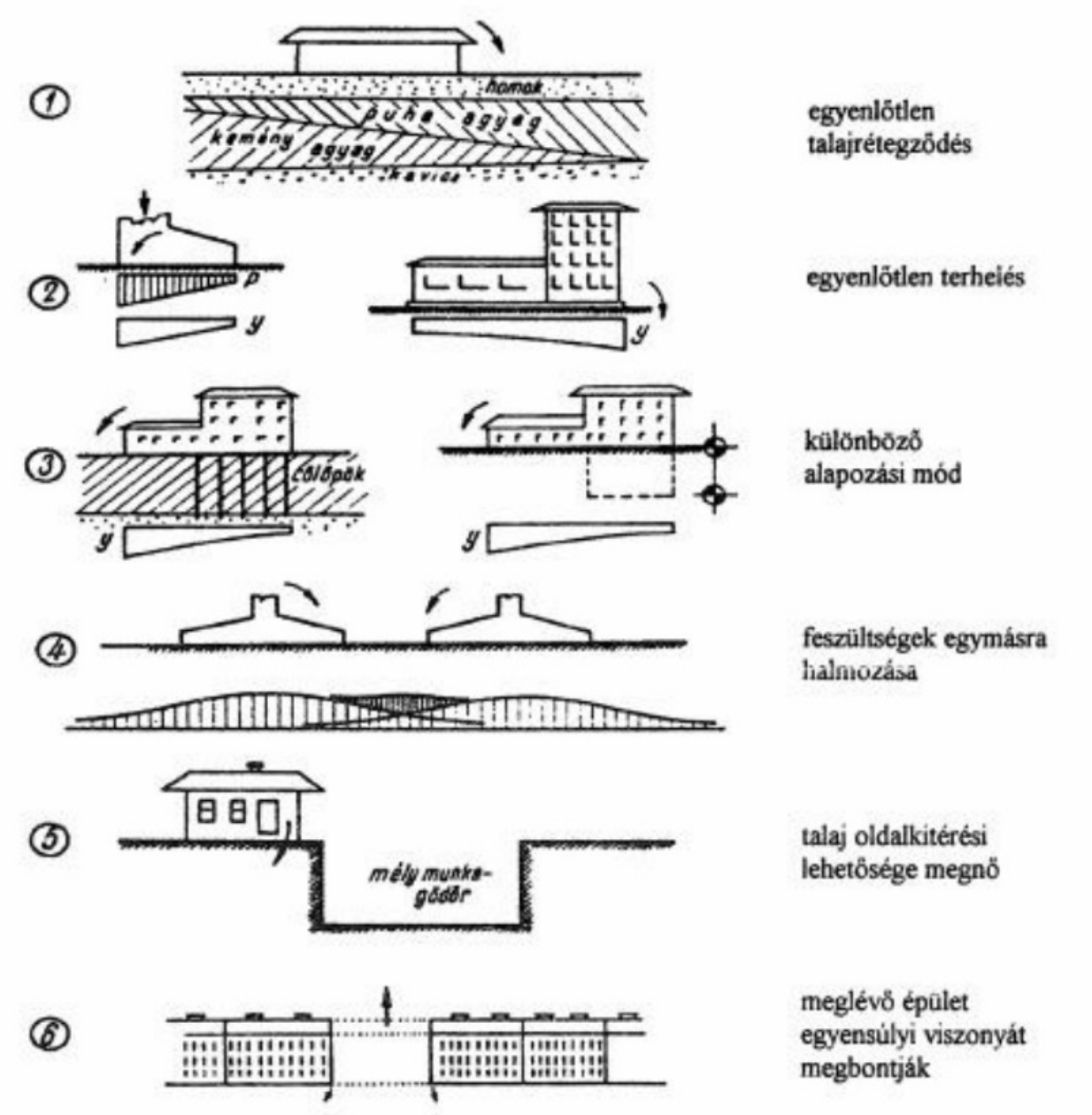

*1. Ábra: Konszolidációs görbék*
*Forrás: Alapozás Előadás jegyzet, 2014 [1]*

Süllyedések okai:

- statikus terhek;

- dinamikus terhek és hatások;
- aláüregelődés (talajvíz kimosó hatása), bánya, pince alagút;
- talajban lévő víz hatása (talajvízszint ingadozás, vízáramlás, roskadás, növekedés, kiszáradás, csőtörés);
- kémiai átalakulások (duzzadás, kioldás), stb.

Az épületkárok döntő részét az egyenlőtlen süllyedés okozza, ezeknek a csoportosítását a 2. ábra mutatja. A repedések a nagyobb süllyedésű hely felé (helyesebben, a mozgás után lejjebb lévő rész felé) emelkednek.




*2. Ábra: Egyenlőtlen süllyedések okai*
*Forrás: Alapozás Előadás jegyzet, 2014 [1]*

Az építőiparban, a legtöbb esetben (Detrekői és Ódor, 1998) magas építmények, lakóházak, hidak, töltések, alagutak, gátak, bányák, erőművek stb. esetén szükséges mozgásvizsgálatot végezni. Ezek a mérések leggyakrabban az építkezések befejeztével kezdődnek és különleges esetben (pl.: veszélyes üzemek) akár több évig tarthatnak az üzemeltetés során. Egy egyszerű családi ház esetén nem merül fel a süllyedésvizsgálat, bár ennek szükségességét is befolyásolhatja a talajminőség. Leggyakrabban több emeletes épületek, hidak esetén jöhet szóba ezen mérések alkalmazása.

Méréseink elvégzése előtt fontos információt kapnunk arról, hogy mekkorák lesznek a várható elmozdulások a megfelelő mérési pontosság megtervezéséhez. Ehhez megfelelő szerkezeti modell

alkalmazása elengedhetetlen, ennek kiválasztása nem a mi feladatunk. Ezen modell alapján határozható meg azon pontok helye, amelyek jól jellemzik a szerkezet mozgását.

Fokozott figyelemmel kell kiválasztanunk az alappontok helyét is. Ezeket célszerű a mozgási zónán kívül, szilárd kőzetbe kell elhelyezni, így elkerülhető, hogy az alappontok mozgása terhelje a vizsgálati pontokat. Biztosításként ezen alappontok közelében célszerű egy helyre több pontot is elhelyezni, így azok ellenőrizhetővé válnak.

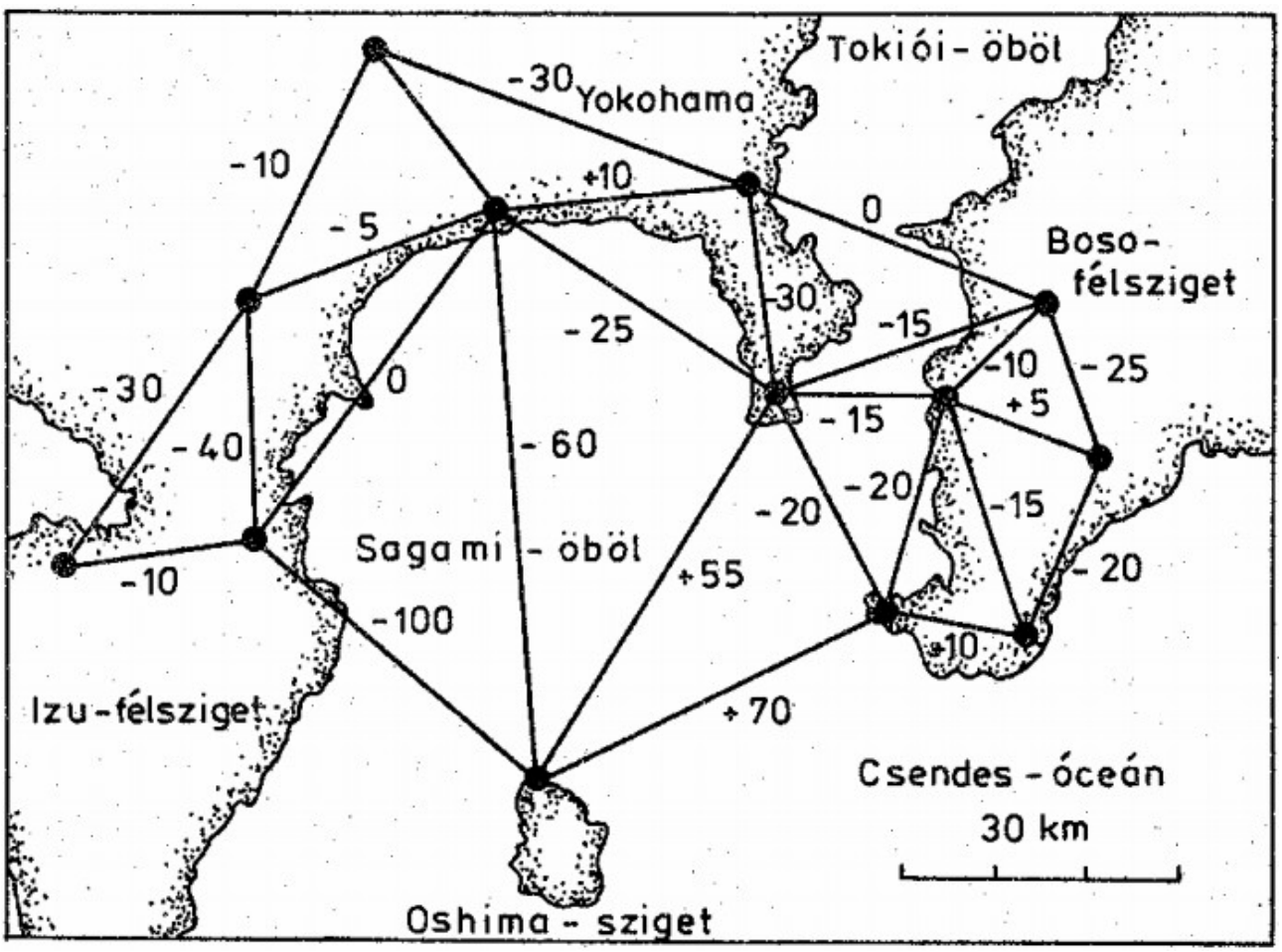
Az előzőekben említett mérési pontosság megválasztásánál nem csak a pontossági követelményeket, hanem a gazdaságosságot is figyelembe kell vennünk. Ha a szerkezeti modellt megfelelően választották ki akkor ebből a mozgás várható lefolyása is jól meghatározható, így a mérési időpontokat is optimálisan ki lehet választani.

Feltételezzük, hogy a mérés ideje alatt a szerkezet mozdulatlan. Ezen körülmény, ha nem biztosított szükséges folyamatos vagy magasabb frekvenciájú mérési technológiát alkalmazni.

Ezek alapján a mozgásvizsgálatok általános sémája a következő: különböző időpontokban a megfelelő technológiával méréseket végzünk az objektumon elhelyezett vizsgálati pontokra, és meghatározzuk azok bizonyos paramétereit, például a koordinátáit. Ha a különböző időpontokban mért adatokból eltérő koordinátákat kapunk, akkor valószínűség számítási módszerekkel meghatározható, hogy a koordinátaeltéréseket ténylegesen mozgás/deformáció okozza vagy csak mérési hiba.

## *2.1.2 Geotudományokban alkalmazott mozgásvizsgálatok [4]*

**A földfelszín vízszintes mozgásainak vizsgálata**




*3. Ábra: Geodéziai alaphálózat pontjai közötti távolságok megváltozása [cm]-ben az 1923-as Kanto-i földrengés során*
*Forrás: ISMÉTELT GEODÉZIAI, GEODINAMIKAI MÉRÉSEK ÉRTELMEZÉSE , 2014 [4]*

A vízszintes irányú mozgások esetén külön kell választani a helyi és a Föld egészére kiterjedő globális mozgásokat. A kisebb területek mozgásvizsgálatának vizsgálatához régebben, nagy szabatosságú vízszintes geodéziai alaphálózatot létesítettek és azokon az alaphálózati méréseket bizonyos időközönként megismételték. A 3. ábrán látható egy hasonló mérés Tokió környékén. Ebben az esetben, ha az ismételt mérések eredményei közötti eltérések mértéke meghaladja a mérési megbízhatóságot, akkor a mérési eredmények geodinamikai értelmezése (a vízszintes elmozdulások megállapítása) viszonylag egyszerű feladat.

Ebben az esetben alkalmazható a GNSS technológia is. A GNSS vevőket mozdulatlannak ítélt helyekre telepítik, ekkora ún. permanens állomásokat hoznak létre. Ezek a már ismert földrajzi koordinátával rendelkező helyek folyamatosan észlelnek, így ha változás következik be valamelyik

koordináta komponensükben, akkor ez kimutatható. Mivel itt az egész földhöz kötött vonatkoztatási rendszerben történik az észlelés, így nagyobb kontinensnyi mozgások is kimutathatóak.

**A földfelszín függőleges mozgásainak vizsgálata**

A jelenlegi függőleges felszínmozgások vizsgálatának (Völgyesi) általánosan alkalmazott geodéziai módszere a nagy kiterjedésű területek esetén az ismételt szabatos magasságmérés. Ezen mérések olyan megbízhatósággal rendelkeznek már, hogy a földfelszínen kijelölt pontok rövidebb idő alatt (10 – 20 év) végbemenő magasság változása már kimutatható.

A vízszintes mozgásokhoz hasonlóan, a süllyedések és emelkedések is kimutathatóak a GNSS technológia segítségével. A választott vonatkoztatási rendszer origójához (geocentrumához) képest határozzuk meg először a térbeli derékszögű koordinátákat. Ezt követően ezt tetszőleges ellipszoid feletti vagy tengerszint feletti magassággá lehet transzformálni. Ugyanúgy ismert koordinátákkal rendelkező ponton történő méréssel lehetséges a mozgások kimutatása.

## 2.2 Méréstechnikai módszerek bemutatása a monitoring rendszerekben

Napjainkban számos megoldás képzelhető el automatizált monitoring rendszerek kivitelezésére. A megfelelő méréstechnika módszer kiválasztásához mérlegelnünk kell a pontossági követelményeket, ebbe beleértendő, hogy mekkora mértékű mozgásról lesz szó és az adott objektum vagy terület külső körülményeit is figyelembe kell vennünk. Nagy prioritása van a gazdaságosságnak is. A következőkben pár, az iparban megvalósult rendszert mutatnék be.

### *2.2.1 Robot mérőállomás automatizált mozgásvizsgálati felhasználása [5]*

Lengyelország fővárosában, Varsóban a kelet-nyugati irányú 2-es metróvonal előkészületei 2010 augusztusában kezdődtek. Az első ütem során, amely 2015 márciusáig tartott, 6,1 kilométert adtak át. Ezalatt az idő alatt volt szükség deformáció és mozgásvizsgálatra a teljes vonalon ugyanis, a városközpont sűrűn beépített és a folyót és az 1-es metróvonalat is keresztezi, így indokolttá vált az épületek és a műtárgy süllyedésének és deformációjának feltételezése.


*4. Ábra: Varsói 2-es metróvonal*
*Forrás: Railway Pro (https://www.railwaypro.com/wp/wp-content/uploads/2018/04/Warsaw-metro.jpg )*

A munkákat az IMG nevű olasz cég végezte el az AGP Metro Poland konzorcium számára. A vállalat tapasztaltnak mondható ezen munkák terén, mivel a Rómában található C metróvonalnál is hasonló feladatokat láttak el. A rendszer maga 10 darab Leica TM30-as mérőállomásból állt, amelyek az építési helyszínen lévő alagútban és az efeletti felszín környezetében lévő épületeken helyeztek el. Az előbbiek a beton alagút falára felhelyezett állványokra, míg utóbbiakat üveg-alumínium „kalitkákban” helyezték el. A minőségbiztosítás érdekében több mint 100, Leica GPR112 típusú referencia prizmát helyeztek ki, míg a mozgásvizsgálati prizmák a környező épületekre létesített Leica GMP104 típusú eszközök voltak. Ekkora terület esetén szükségessé vált meteorológiai szenzor elhelyezése is, mivel az atmoszferikus korrekciók megfelelő számításához elengedhetetlen megfelelő pontossággal ismernünk a hőmérsékletet, a páratartalmat és légnyomást. Mind a robot mérőállomások által, mind a meteorológiai szenzorok által szolgáltatott adatokat a Leica GeoMos szoftvere dolgozta fel. Ezeken felül természetesen felsőrendű szintezést és hagyományos mérőállomások által történő mérések is voltak a helyszínen.

*5. Ábra: Leica TM30*
*Forrás: Leica Geosystems TruStory [5]*

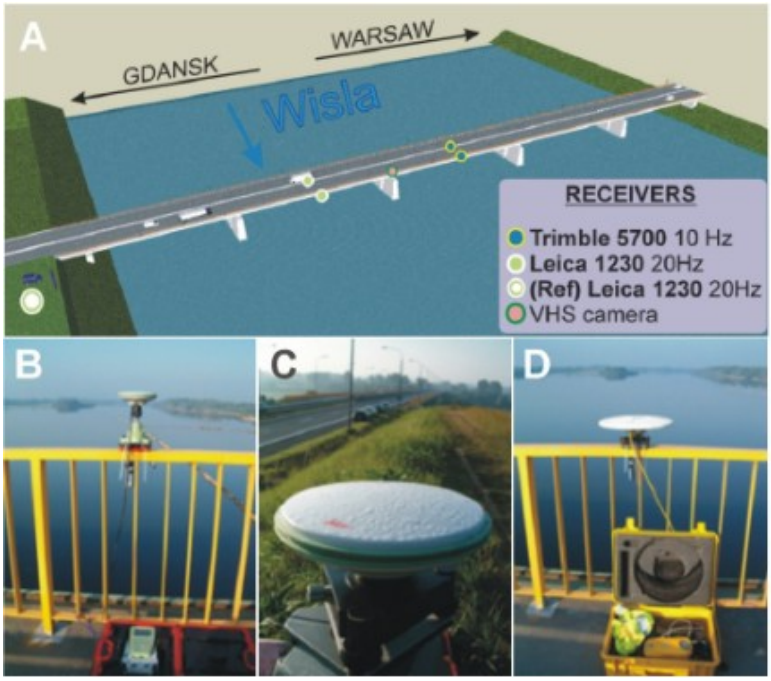
## *2.2.2 GNSS automatizált mozgásvizsgálati felhasználása [6]*

Az előző fejezethez hasonlóan a Varsóban található Katonai Műszaki Egyetem (MUT), Alkalmazott Geomatikai Központja foglalkozott egy kutatás keretében korábban a GNSS technológia felhasználásával az automatizált monitoring rendszerekben.

A GNSS technika megannyi előnyével szemben számos hátránya is van, ha alkalmazni szeretnénk egy automatizált monitoring rendszerben:

- szabad égboltra való rálátás,
- különböző mérési módszerek (legalább két műszer szükséges a méréshez),
- Real Time Kinematic (RTK) üzemmód esetén a pontosság vízszintes 1 cm, míg magsaságilag 3 cm lehet maximum.

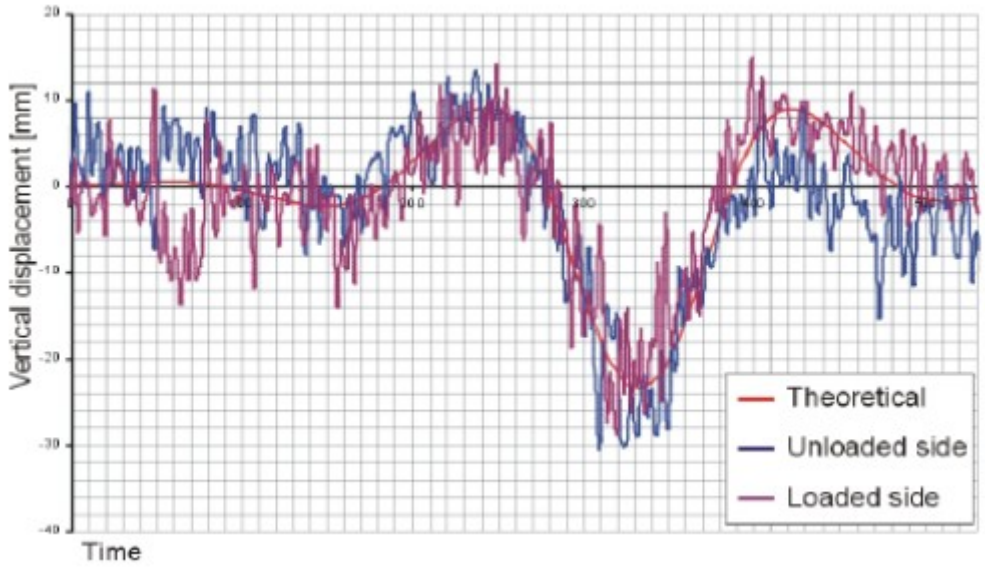
A tesztek során kettő, a Visztula folyó felett átívelő híd próbaterhelése során tesztelték ezen technológiát. Az első műtárgy mérése 2008. Augusztus 24.-én vette kezdetét, Zakroczym közelében. A mérési periódus éjjel 3:00 és 8:00 között volt, ugyanis a legfőbb terhelést a hídon, az éjszaka folyamán közlekedő kamionok és a reggel induló személygépjárművek okozzák.




*6. Ábra: Műszerek elhelyezése*
*Forrás: GNSS-based multi-sensor system for structural monitoring applications [6]*

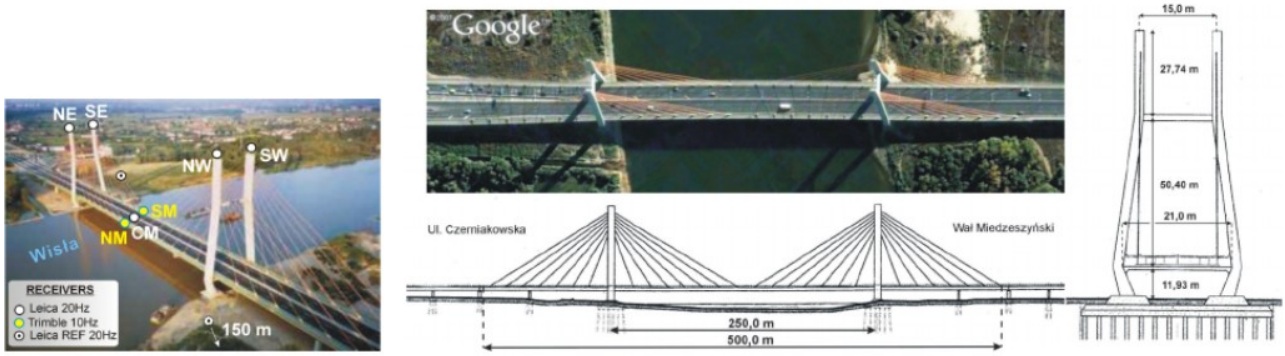
Négy GNSS vevőt alkalmaztak a mérések során, 2 darab Leica 1230-ast és 2 darab Trimble 5700-as modellt, amelyek a 95 méter hosszú műtárgy útjának mindkét oldalán megtalálhatóak voltak. (6. ábra). A műszerek elrendezése lehetővé tette, hogy keresztszelvény mentén történhessen az elemzés

és így a a szerkezet elfordulása is mérhető volt. A 4 GNSS vevőn felül egy referenciaállomás is elhelyezésre került a parton és egy VHS kamera a középső tartószerkezetnél, amely a dinamikus terhelések hatását volt hivatott rögzíteni.




*7. Ábra: Függőleges elmozdulás*
*Forrás: GNSS-based multi-sensor system for structural monitoring applications [6]*

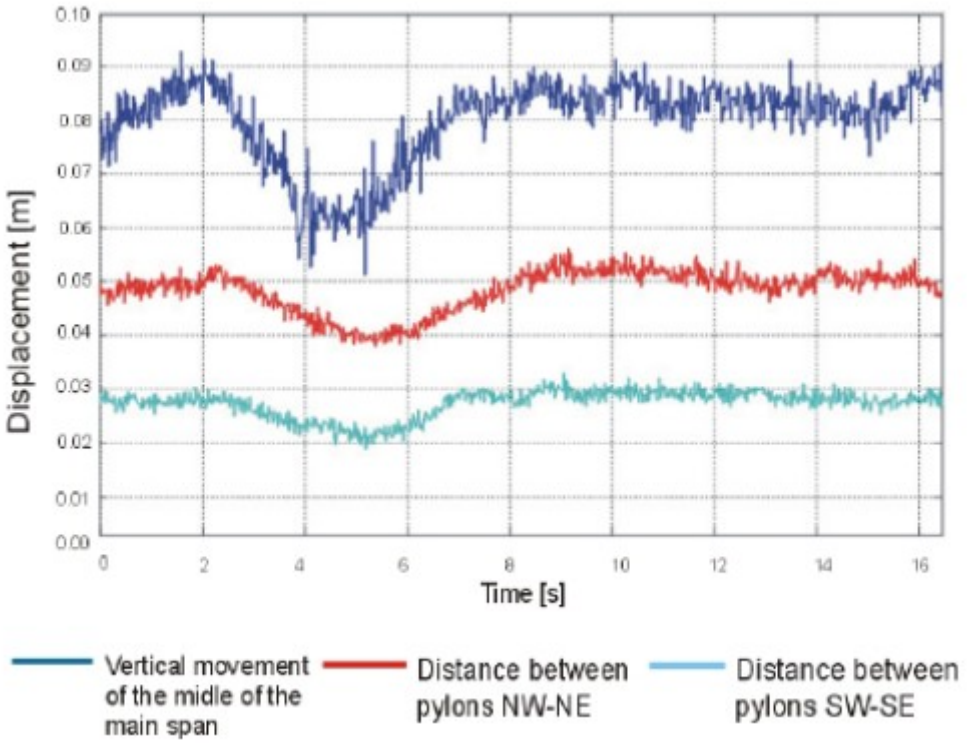
A 7. ábrán látható a Leica 1230-as vevők által 20 Hz-es frekvencián rögzített adatok és az elméleti elmozdulás görbe. Az adatok 300 kN-os terhelés esetén lettek feltüntetve, a tehergépkocsik elhaladása esetén. A lila szín a terhelt, míg a kék terheletlen oldalt jelzi. A piros folyamatos vonal az elméleti elmozdulás görbét jeleníti meg. Jól látható, hogy a mért adatok jól tükrözik az elméleti modellt, így elmondható, hogy a GNSS technológia megbízhatónak mondható ezen mérési folyamatok elvégzésére.




*8. Ábra: Siekierkowski híd próbaterhelése*
*Forrás: GNSS-based multi-sensor system for structural monitoring applications [6]*

Hasonló kísérletet hajtott végre ez a kutatói csapat a Siekierkowski hídon is, amely 250 méter hosszú és négy 90 méter magas pilonjához 28 acél sodrony rögzíti a műtárgyat. Ezt a mérést már 9

darab GNSS vevővel hajtották végre ebből négy a tartóoszlopok tetején (8. ábra: NE, NW, SE, SW), 3 a középső keresztszelvényben a két oldalt és a tengelyen, kettő pedig referenciaállomásként üzemelt. Ezeken felül már 2 darab VHS kamerát alkalmaztak hasonló célokra mint az előzőekben. A fő cél a pilonok viselkedésének meghatározása volt abban az esetben, ha a két pilon közötti rész dinamikus terhelést kapott. Az eredmények a következő ábrán láthatóak:



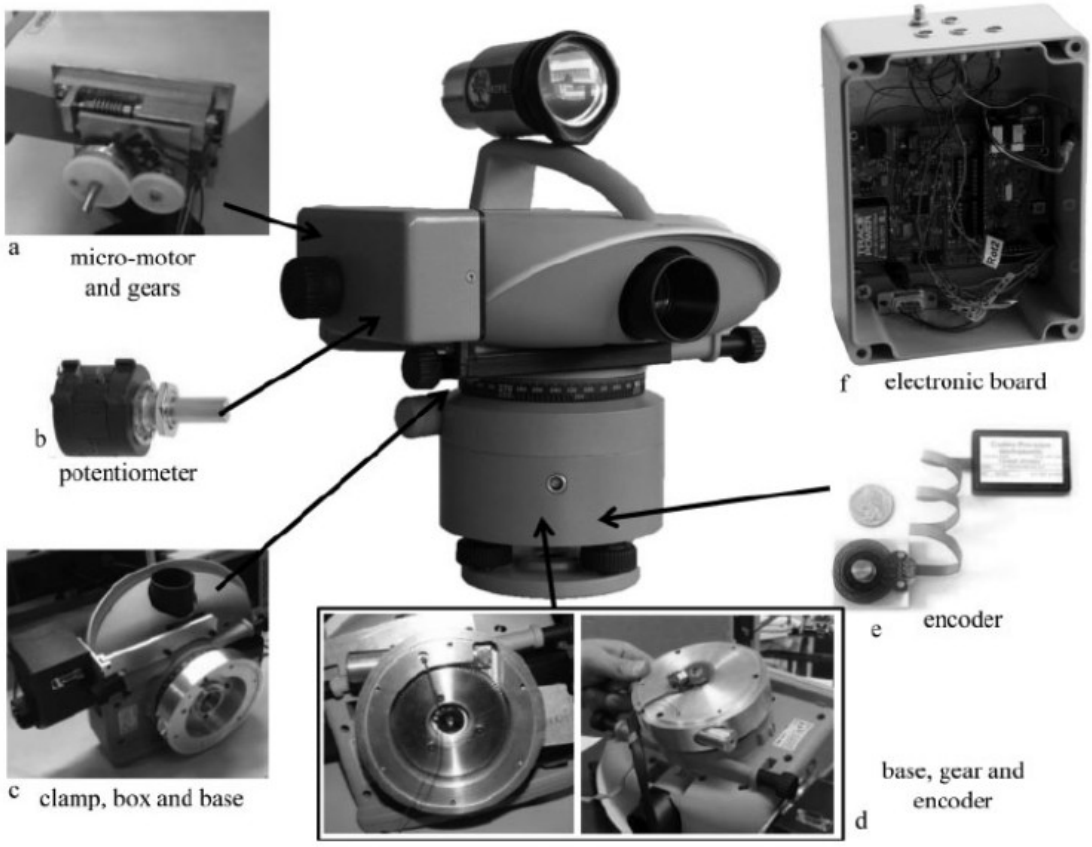

*9. Ábra: Próbaterhelés eredményei*
*Forrás: GNSS-based multi-sensor system for structural monitoring applications [6]*

A függőleges irányú mozgás a középső keresztmetszetben elérte a 2 centiméter is, míg vízszintes irányban látható, hogy 1-1 centiméter volt a relatív mozgás. Ezen értékeket két darab teherautó által keltet dinamikus teher idézte elő, amelyek 60 km/órás sebességgel haladtak el az északi oldalon.

## *2.2.3 Digitális szintező automatizált mozgásvizsgálati felhasználása [7]*

Az előző két esetben tárgyalt mozgásvizsgálatok esetén maximum milliméteres mozgások mutathatóak ki, viszont adódnak helyzetek a mérnöki gyakorlatban, ahol akár tized milliméteres pontosság a követelmény. Ilyenkor a felsőrendű szintezés, mint technológia jöhet szóba, viszont ennek automatizálása nem egyszerű feladat.

Az olasz Ferrarai Egyetemen indult egy kutatás korábban, melyben egy Leica DNA03-as szintező műszert robotizáltak 2005-ben. A következő ábrán láthatjuk a prototípus felépítését:




*10. Ábra: Robotizált Leica DNA03*
*Forrás: Motorised digital levels: development and applications [7]*

a. mikro-motor és fogaskerekek, az automata fókuszáláshoz
b. potenciométer, a megfelelő fókusz beállításhoz
c. műanyag doboz, melyben a fókusz motorja található, az alumínium bilincs ami a műszerhez rögzítését hivatott szolgálni és az új alumínium talp felső része, amely a forgatómotort tartalmazza
d. mikro-motor és fogaskerék az új talpban (bal oldalt) és egy jeladó az alsó részhez rögzítve (jobb oldalt)

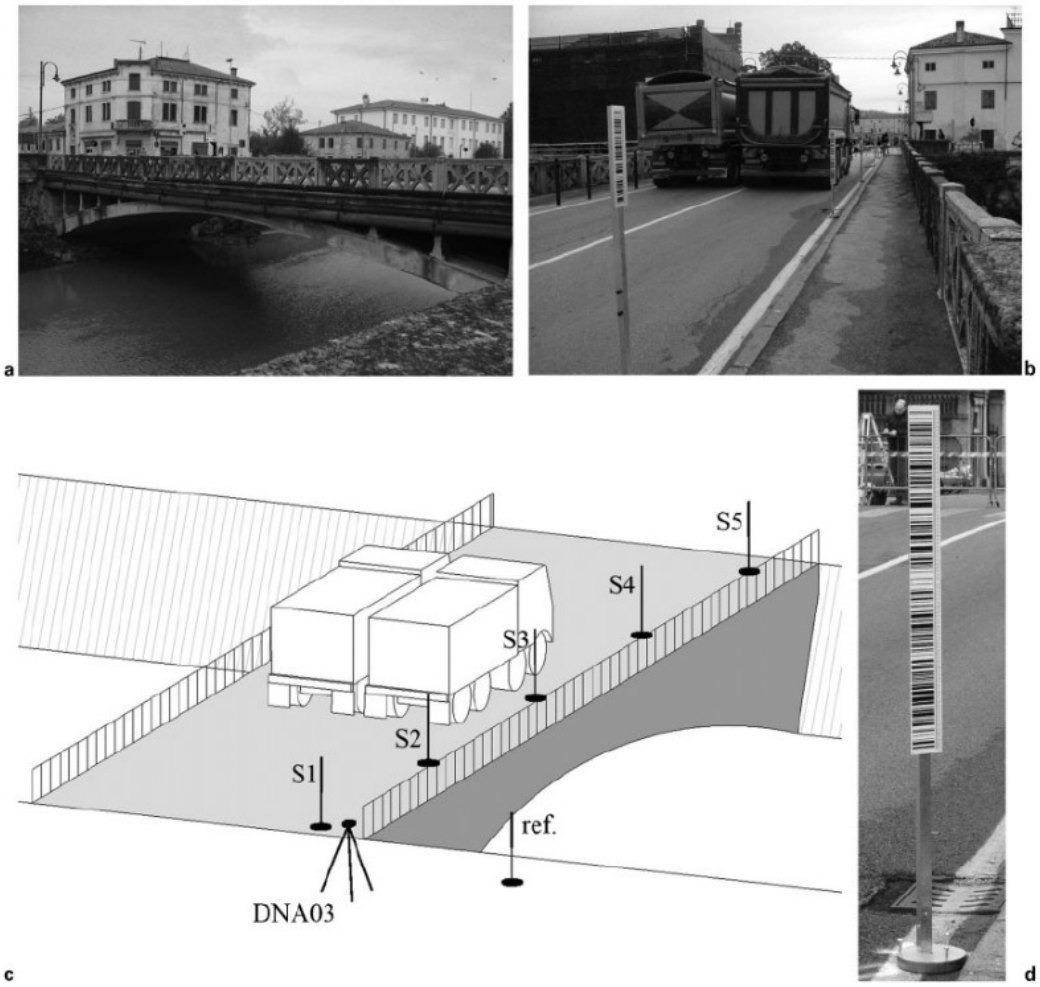
e. jeladó

f. külső alaplap alkatrészei

A két mikro-motor segítségével történik a forgatás és a fókuszálás. Az előbbinél a jeladó az alaplap számára küldi az üzeneteket, míg az utóbbinál a potenciométer teszi azt, itt lehetőség van a manuális fókuszálásra is, a meghosszabbított csavar segítségével. Az alaplap mikroprocesszora soros porton keresztül van a műszerre kötve (RS-232 protokollt használ). A kommunikáció a számítógép és a műszerhez kötött számítógép között bluetoothon vagy mobil interneten keresztül történik. A felhasználói oldalon egy Visual C# alapú szoftveren keresztül történik, ami távirányítási funkciókat, illetve az adatok feldolgozását, megjelenítését biztosítja.

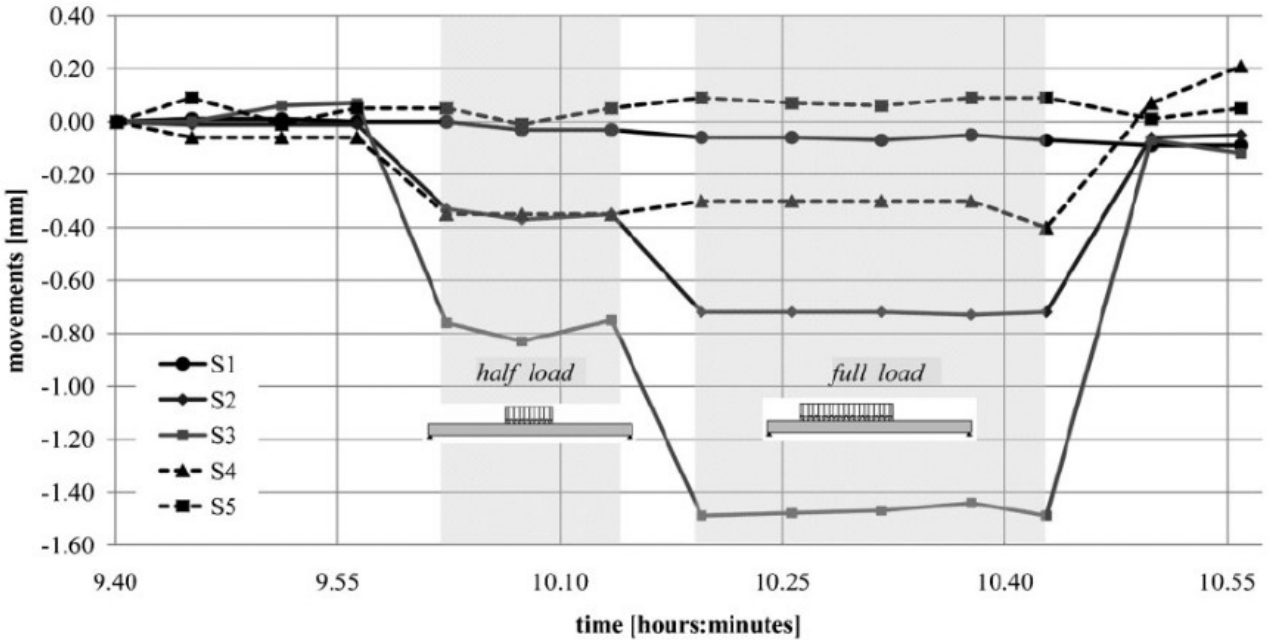
Tesztelésként egy híd statikus próbaterhelésénél használták fel a kész rendszert. Ezen szerkezetek próbaterhelése pár órától akár napokig is terjedhet nagyobb szerkezetek esetén a különböző teherállások száma miatt. Ilyenkor egy ilyen rendszer előnye, hogy csak a mérések elején szükséges telepíteni, majd a végén leszerelni a rendszert.

A közel 10 méter széles és 31 méter hosszú vasbeton szerkezet a 11. a) ábráján látható.




*11. Ábra: Terepi teszt körülmények*
*Forrás: Motorised digital levels: development and applications [7]*

A teherautók kitakarása miatt csak az egyik oldalon voltak elhelyezve egy vonalban a monitoring pontok. A legközelebbi (S1) és a referencia léc 6, míg a legtávolabb lévő (S5) 34,5 méterre volt a műszertől. Az eredmények a 12. ábrán láthatóak. A műszer minden mérés alkalmával először a referencia lécet olvasta le, majd azt követte a monitoring pontok észlelése. Biztosításképpen az egyes teherállásokat kézi módon is végig szintezték, de a legnagyobb eltérés a két technika között 0,3 milliméter volt.




*12. Ábra: Eredmények*
*Forrás: Motorised digital levels: development and applications [7]*

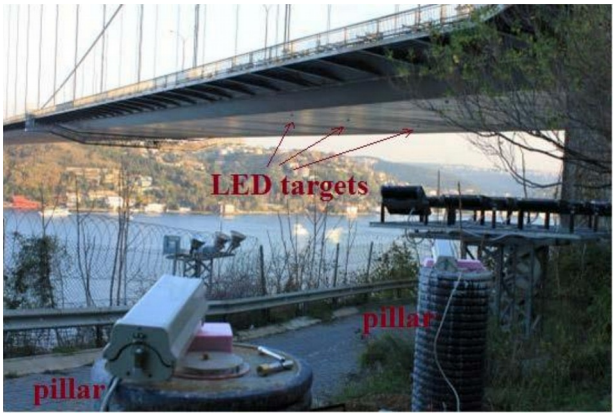
Összességében elmondható, hogy a robotizált szintező műszer megvalósítható, de a motorizálás magas költsége és a felhasználás szűk köre miatt, nem terjedt el a robot mérőállomásokkal ellentétben. Fontos javítás lenne a rászerelt eszközök méretének csökkentése és a forgatás gyorsaságának növelése, de a néhány tíz méterre korlátozott hatótávolságon ez sem segítene.

## *2.2.4 Egyéb szenzorok*

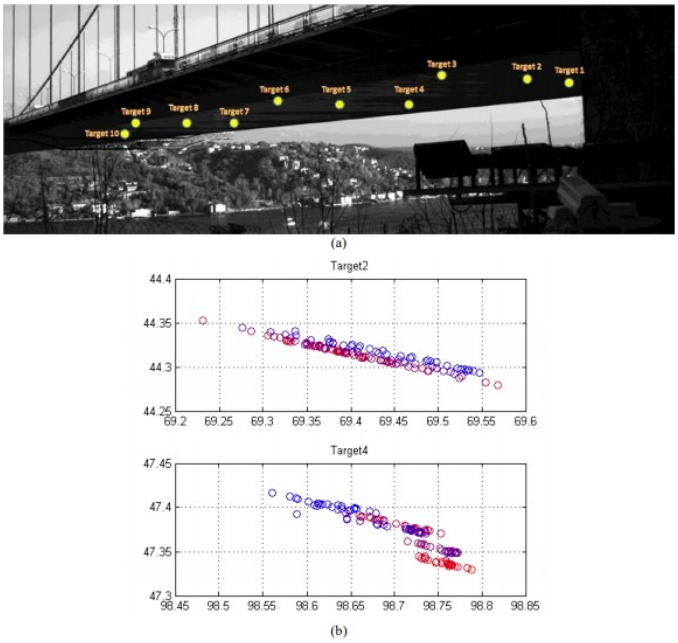
**Kamera rendszerek [8]**

A 2.2.2 fejezetben is említést tettem a két mérőrendszer esetén egy VHS-re rögzítő kameráról. A kamera rendszerek manapság is részét képezhetik egy automatizált mérőrendszernek. Előnye, hogy gyors, dinamikus mozgásokat is képesek rögzíteni és manapság minden elmozdulás adat kész algoritmusokkal, a pontossági igényeknek megfelelően kigyűjthető belőle.

Isztambulban található Isztambuli Műszaki Egyetem és az Isik Egyetem egy projektje alkalmával használtak ilyen automatizált mozgásvizsgálati rendszert a Második Boszporusz-híd lehajlásának vizsgálatához. Ez a híd köti össze Ázsiát és Európát, így elég forgalmasnak mondható. A mérések során 3, Basler A402 típusú kamerával rögzítették 2 napon keresztül az adatokat. Az eszköz felbontása 2352 x 800 pixel volt, de képes lett volna akár 2352 x 1726 pixelben is rögzíteni. Ezen felül az eszközök 12 képkocka/másodpercet használtak ki a 24-ből amire még képes a kamera.




*13. Ábra: Kamerák és a célpontok*
*Forrás: Photogrammetric Deformation Monitoring of the Second Bosphorus Bridge in Istanbul [8]*



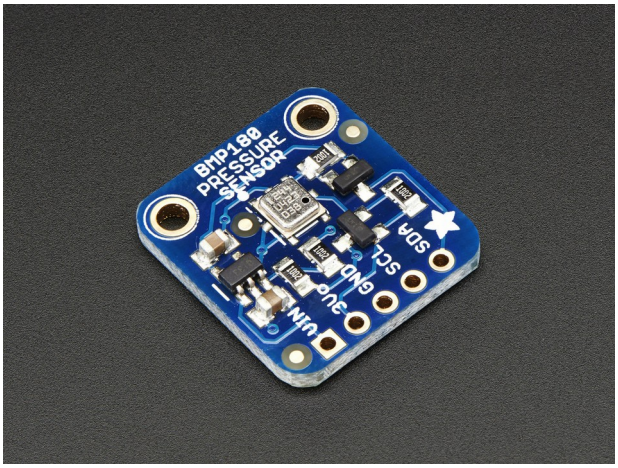

*14. Ábra: a) Target-ek elheyezése b) 2-es és 4-es target elmozdulásai*
*Forrás: Photogrammetric Deformation Monitoring of the Second Bosphorus Bridge in Istanbul [8]*

Tizenkét darab LED célpont jelölte a monitoring pontok helyét, amelyhez állandó áramellátást biztosítottak. A mérések feldolgozásához egy Matlab alapú algoritmust használtak fel, amely automatizáltan képes megtalálni a több mint 35 000 rögzített képkockán a LED-eket kereszt-korrelációval. A 14./a) ábrán láthatóak a monitoring pontok elhelyezkedései és a 2-es és 4-es pont elmozdulásairól egy-egy grafikon. A koordináták elméleti pontossága a ±72.4 mm, ±19.9 mm, és ±13.4 mm rendre, az X, Y, Z tengelyek mentén. Ellenőrzésképpen egy mérőállomás segítségével mérték meg a pontok helyzetét a mérés ideje alatt.

## Meteorológiai szenzorok

Kiegészítő adatokként, ha szükséges a mozgásvizsgálat területi nagysága vagy a mérés időtartama miatt, meteorológiai szenzorok implementálása is szükségessé válhat ezen rendszerekben. Mivel a robot mérőállomások méréseit nagy távolságok esetén nagyban befolyásolja a hőmérséklet és a légnyomás, így ezen adatokat a mérés pillanatában elengedhetetlen ismernünk.

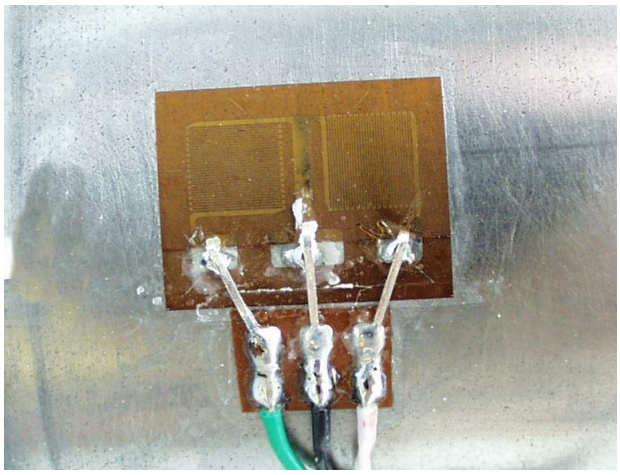
Egy gazdaságos megoldás, ha Raspberry Pi-hez kapható Adafruit BMP180 légnyomás, hőmérséklet és magasságmérésére is alkalmas szenzort szerezzük be. Ezen felül a ma használatos újabb műszerekben megtalálható már az erre a célra használatos szenzorok, de legtöbbször a fizikai adottságoknak köszönhetően csak a légnyomás mérést alkalmazzák. Mivel ezek a műszerben találhatóak, így a hőmérséklet értékekre valószínűleg fals adatokat kapnánk, mert a műszer belsejében ezek a mért értékek eltérhetnek a külső léghőmérséklettől.




*15. Ábra: Adafruit BMP180*
*forrás: adatfruit.com (https://www.adafruit.com/product/1603 )*

## Nyúlásmérő bélyeg [9]

A nyúlásmérő bélyeg a testek megnyúlásának mérésére szolgáló eszköz. Leggyakrabban a a rugalmas elektromos szigetelő fóliából álló fajtáját alkalmazzák, melyre vezető réteget visznek fel a megfelelő alakban. Ezt a mérendő szerkezetre ragasztják, így ha az deformációt szenved vele együtt fog a bélyeg is és így változik az ellenállása. Ez és a megnyúlás egymással arányosak. Leggyakrabban ezt szilárdsági mérések során alkalmazzák. Megfelelő szilárdságú ragasztóval felragasztják közvetlen az objektumra és Wheatstone-híddal mérik legtöbbször az ellenállást a bélyegben. Ez az eszköz is használható automatizált monitoring rendszerek részeként.



*16. Ábra: Nyúlásmérő bélyeg*
*Forrás: Wikipédia [9]*

# 3. Ulyxes nyílt forráskódú monitoring rendszer [10][11]

„Az Ulyxes, egy nyílt forráskódú projekt a robot mérőállomások és egyéb helymeghatározó szenzorok vezérlésére, valamint a mérési eredmények internetes térkép alapú publikálására. A projekt célja egy keretrendszer létrehozása a robot mérőállomások számítógépről történő vezérléséhez és az adatok internetes publikálásához, nem egy bárki által közvetlenül használható alkalmazás, hanem egy keretrendszer létrehozását célozta meg, a fejlesztés során számos további nyílt forráskódú projektet használt fel.”(Idézet az Ulyxes rendszer honlapján szereplő „Projekt célja” részből)

Mostanra a rendszer rengeteg komponenssel bővült a több mint 10 éve történő indulás óta. A rendszerre való igény először a paksi atomerőmű II. blokk lokalizációs torony deformációmérése során jelentkezett 2008-ban. Ezt követően 2010-ben alakult ki az első koncepció amelyben „A cél minimális programozási munkával (meglévő, elérhető ingyenes komponensek felhasználásával), a lehetőségekhez képest minél automatikusabban működő rendszer kialakítása volt.” (Idézet az „Első rendszer koncepció 2010” diasorából)

Az elindult fejlesztésbe 2011-ben Zemkó Szonja, aki BSc-s „Mozgásvizsgálati mérések eredményeinek internetes megjelenítése” című diplomatervével kapcsolódott be. Ugyanebben az évben indul az MSc képzésen az „Alagútmérés, automatizált mérésfeldolgozás” nevű tárgy, amelynek már szerves része az Ulyxes oktatása. 2012-ben elindul a projekt weboldala, ahol megtalálható a forráskód és egy Demo oldal is, ami a webes lekérdezések tesztelését hivatott szolgálni. Egy újabb diploma születik ebben az esztendőben „Robot mérőállomásokra alapozott


*17. Ábra: Ulyxes korábbi alkalmazásai*
*Forrás: Ulyxes rendszer honlapja [10]*

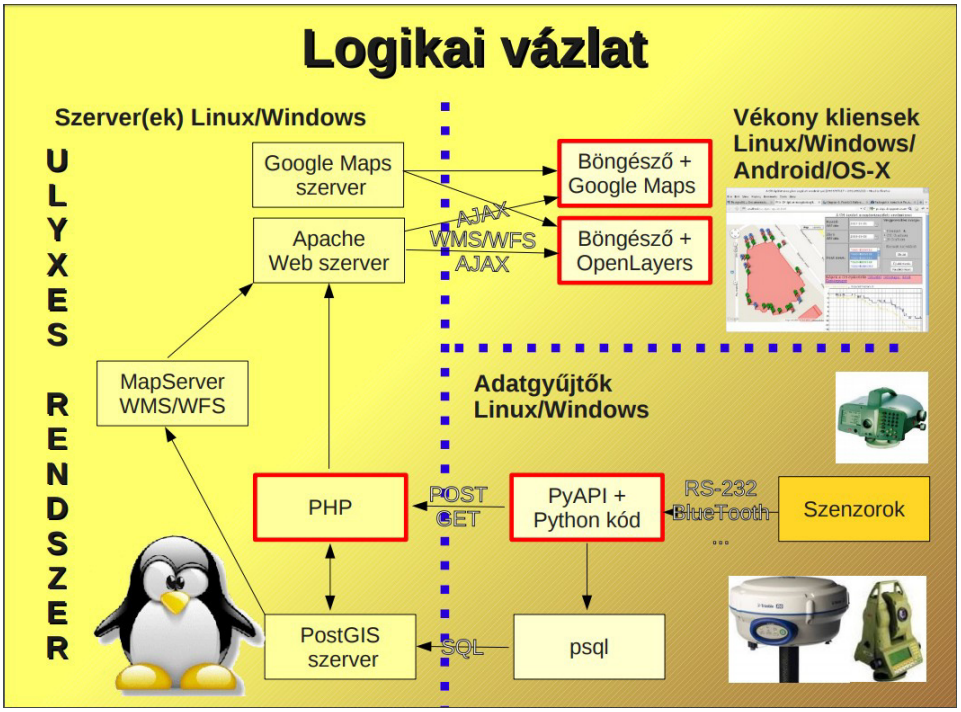
mozgásvizsgálati rendszer továbbfejlesztése és gyakorlati alkalmazása” címmel, készítője: Moka Dániel.

A már használható rendszert a következő években több projektben is felhasználja a tanszék. 2013-ban M0 Hárosi híd próbaterhelése folyamán, majd 2014-ben Szolnok-Szajol vasúti híd hasonló munkáiban és a rá következő esztendőben a Rákóczi híd próbaterhelése során. Eközben 2014-től a Tcl (Tool Command Language) nyelvet felváltja a nagyobb lehetőségeket kínáló Python nyelv. Előnye, hogy sokkal több előre kész függvénykönyvtár van, amely rengeteg időt és energiát spórol meg a fejlesztők számára.

A jelenben sem fejeződött be a projekt fejlesztése, további diplomatervek és TDK dolgozatok születtek a témában. Mostanra a kompatibilis szenzorok száma nagymértékben nőtt, a robot mérőállomásokon túl digitális szintező, GNSS vevő, kamerák, meteorológia szenzor, gyorsulásmérő és Raspberry Pi is implementálásra került.

## 3.1 Rendszer felépítése

### *3.1.1 Logikai vázlat [10] [12] [13] [14]*

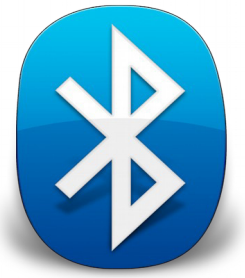



*18. Ábra: Az Ulyxes rendszer felépítése*
*Forrás: Ulyxes rendszer honlapja [10]*

A rendszer vázlatát a 18. ábrán láthatjuk. Ez a 2016-os állapotot tükrözi, de hasonló a helyzet a jelenben is.

Az adatgyűjtés a Szenzorok résznél indul (jobb oldalt alul), ezek az eszközök, műszerek az előző bekezdésben felsoroltak lehetnek. A megfelelő módon megmért, észlelt adatok valamilyen adatátviteli módon tovább haladnak. Ez lehet RS-232 (Recommended Standard 232) vagy a mostanra implementált Bluetooth technológia (lásd továbbiakban). A most említett két módon felül található több megoldás is, ezeket a későbbiekben részletezném.

Az előbbit „1960-ban hagyta jóvá a nemzetközi távközlési bizottság , ez a szabvány az adatátvitelt a DTE (Data Terminal Equipment) berendezés, ami általában számítógép, illetve számítógépes terminál és a DCE (Data Communications Equipment) távközlési, vagy egyéb berendezés (modem, nyomtató, képolvasó, esetünkben műszerek is stb.) között valósítja meg.” (Idézet a Wikipédiáról.) Laikusok számára soros portként ismeretes, manapság a legtöbb számítógép alaplapjáról már eltűnt ez a bemeneti egység.



*19. Ábra: Bluetooth logója*
*Forrás: Wikipédia [12]*

Az utóbbi napjainkban is széles körben ismeretes és használt vezeték nélküli kommunikációs szabvány. „A Bluetooth név eredete *Harald Blåtand* dán király nevéből származik, aki nagyon szerette az áfonyát, ezért kék fog lett a neve. Harald arról volt nevezetes, hogy egyesítette a lázongó dán, norvég és svéd törzseket. Ehhez hasonlóan a Bluetooth-t is arra szánták, hogy egyesítsen és összekössön olyan különböző eszközöket, mint a számítógép, a mobiltelefon, vagy a fejhallgató. Lényege, hogy rövid hatótávolságú rádió kapcsolatot hozzon létre ezen eszközök között.” (Idézet a Wikipédiáról)

Ezeken a csatornákon eljutnak az adatgyűjtőkhöz az észlelések, amiből a rendszerben megtalálható még a régebbi Tcl API, és a 2014-ben az előzőt felváltó modernebb nyelv, a Python alapú PyAPI függvénykönyvtár is. Ezek az algoritmusok kommunikálnak a műszerekkel, szenzorokkal, adnak parancsokat, vagy a beérkező adatot teszik értelmezhetővé, feldolgozhatóvá. Ahhoz, hogy komplexebb problémák is megoldhatóak legyenek, az API könyvtárban létrehozott alap parancsok segítségével és felhasználásával a PyAPPS könyvtárban összetettebb feladatokat ellátó minta programok találhatóak. Ilyen például a HorizontalSection applikáció, amely egy vízszintes irányú metszetet készít robot mérőállomás segítségével vagy Section nevű amely hasonlóan az előzőhöz egy metszetet hoz létre egy tetszőleges síkban.


*20. Ábra: PHP logó*
*Forrás Wikipédia [13]*

Ezt követően a már feldolgozott adatok két irányba indulhatnak, ha webes megjelenítés vagy adatbázisba történő közzététel a célunk.

Az első esetben közvetlen PHP utasítások segítségével hajtjuk végre

az adatbázis feltöltését. „A PHP egy általános szerveroldali szkriptnyelv dinamikus weblapok készítésére. Az első szkriptnyelvek egyike, amely külső fájl használata helyett HTML oldalba ágyazható. A kódot a webszerver PHP feldolgozómodulja értelmezi, ezzel dinamikus weboldalakat hozva létre.” (Idézet a Wikipédiáról)

A másik lehetőség a PostGIS szerverre történő adatok közlése. A PostGIS a nyílt forráskódú PostgreSQL relációs adatbázis kezelő alapjaira épül és kiegészíti azt a földrajzi objektumok kezelésével. Ez a program az OGC (Open Geospatial Consortium) szabványait követi. Sok elterjedt kereskedelmi és nyílt forráskódú szoftver alkalmazza adatbázisnak a PostGIS-t, többek között az ArcGIS, a QGIS és az internetes közösségi térkép, az OpenStreetMap is. A PostGIS adatbázisából egy MapServer nevű nyílt (Mapserver.org) forráskódú webes térinformatikai programcsomag jeleníti meg ezeket az adatokat. A program segítségével térképeket állíthatunk elő a különböző formátumú raszteres és vektoros adatokból, azokon lekérdezéseket hajthatunk végre, stb.

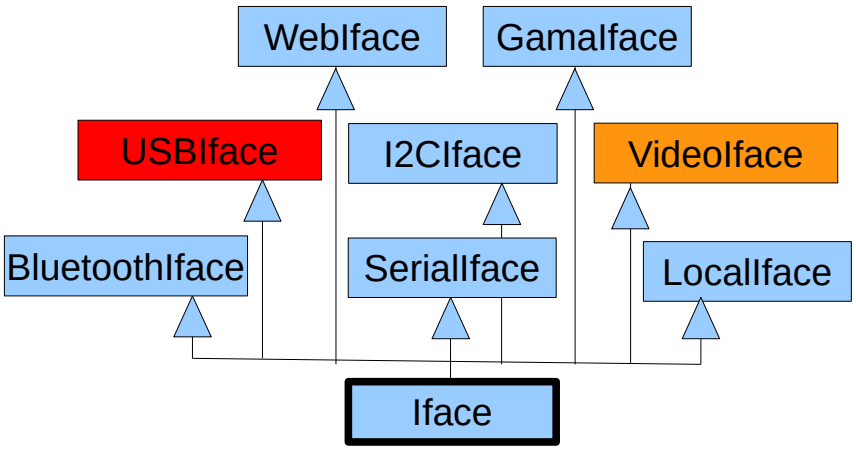
*21. Ábra: PostGIS logó*
*Forrás: Wikipédia [14]*

Az adatok a szerverek oldalán tovább folytatják az útjukat egészen a tanszéken is működtetett Apache Web szerver felé. Ezt követően bármikor, bárhonnan elérhető már az adatbázisunk, lekérdezhetünk vagy módosíthatjuk azokat.

Ezt követően már a megjelenítés van hátra. Jelen pillanatban a böngészőben megjelenített formában vagy a Google Maps vagy OpenLayers-be ágyazva tekinthető meg. Vékony kliensek alkalmazásával (WMS, WFS) csak azok az adatok fognak a képernyőnkön megjelenni, amelyek számunkra szükségesek.

### *3.1.2 Adatgyűjtők felépítése [10] [15]*

A műszer és a számítógép kommunikációjához és az adatok rögzítéséhez az Ulyxes-ben kialakított rendszert a 23. ábra szemlélteti. Egy adott mérőeszköz 2 elengedhetetlen és 2 opcionális egységből áll.




*22. Ábra: Interfész csoport felépítése*
*Forrás: Ulyxes rendszer honlapja [10]*

Elsőként azt kell megválasztanunk, hogy a műszerünk, eszközünk milyen módon kommunikáljon a számítógépünkkel. Jelen pillanatban erre 6 módszer áll rendelkezésre (22. ábra) az Ulyxes rendszerében. A Serial Interface a soros vonalon kommunikál RS-232 szabvány alapján. Manapság

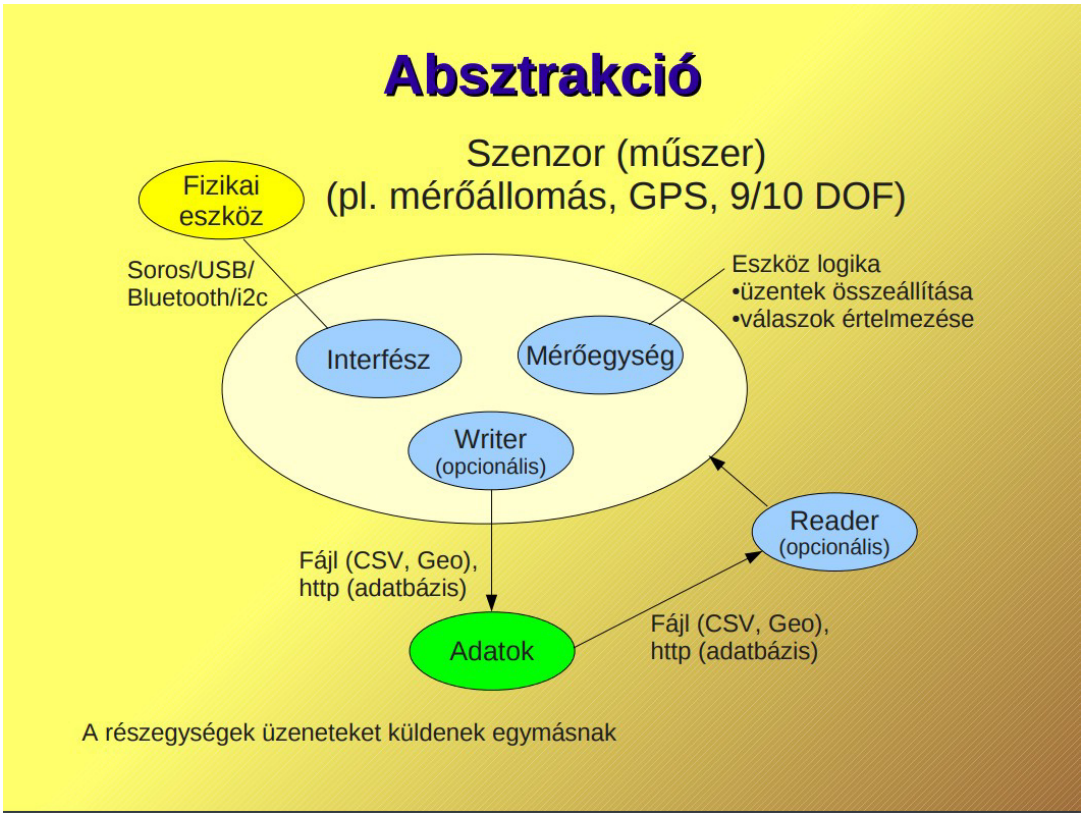
ezen portok nagy része megszűnt a számítógépek alaplapján, viszont kapható USB → RS-232 átalakító a piacon. A legtöbb ma használatos műszer még a mai napig fel van szerelve COM porttal.

A második kommunikációs metódus az I2C interface. Ez a port klasszikusan a legújabb OLED alapú kijelzők, légnyomás érzékelők és gyorsulásmérők által alkalmazott módszer a kommunikációra. Az Ulyxes-ben hőmérsékleti adatokat is képesek vagyunk adatbázisba vagy egyéb módon tárolni, így ha tegyük fel egy Raspberry PI-n (amely rendelkezik I2C interfésszel) keresztül ezeket az értékeket kinyerjük, képesek vagyunk azokat felhasználni.

A harmadik lehetőségünk a helyi fájlok beolvasása és használata a Local interface-n keresztül. Ez főleg tesztelés szempontjából fontos, de ha már meglévő adatok alapján szeretnénk megvalósítani valamit, akkor ez a lehetőség is rendelkezésünkre áll.

A negyedik lehetőség az újonnan implementált BluetoothIface, amely a szabványos bluetooth kommunikáción keresztüli adatközlést hivatott szolgálni.

A WebIface és a GamaIface az utolsó két lehetőségünk a kommunikáció megvalósítására. Az előbbi a HTTP alapú kommunikáció és a JSON (részletek továbbiakban) fájlok kezelésére szolgáló



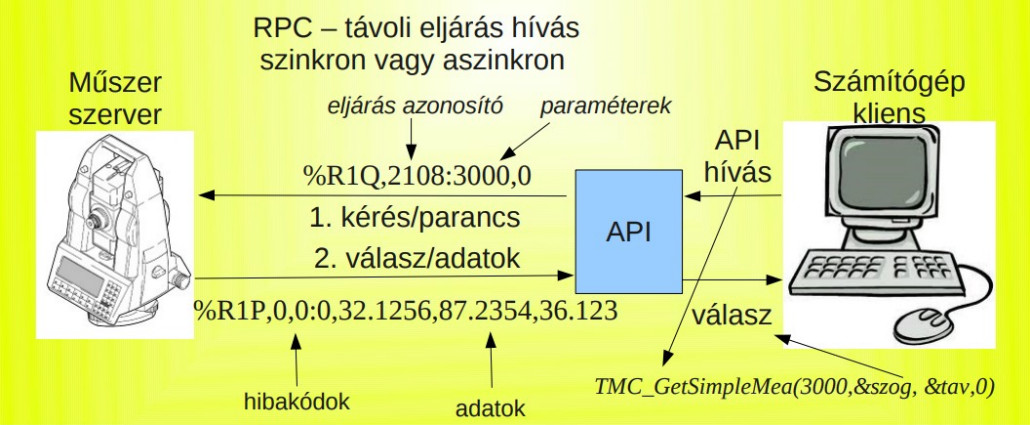

*23. Ábra: Adatgyűjtök logikai vázlata*
*Forrás: Ulyxes rendszer honlapja [10]*

algoritmus, míg az utóbbi a GNU gama szoftverével történő kapcsolat és konverzió kiépítésére szolgál.

Részben kész állapotban megtalálható még a Video interfész is, ugyanis jelen pillanatban egy videó fájlból vagy videó stream-ből csak képkockákat tud kiragadni az algoritmus. Ezeken végezhető el után valamilyen szkripttel az adott elemzés, kiértékelés.

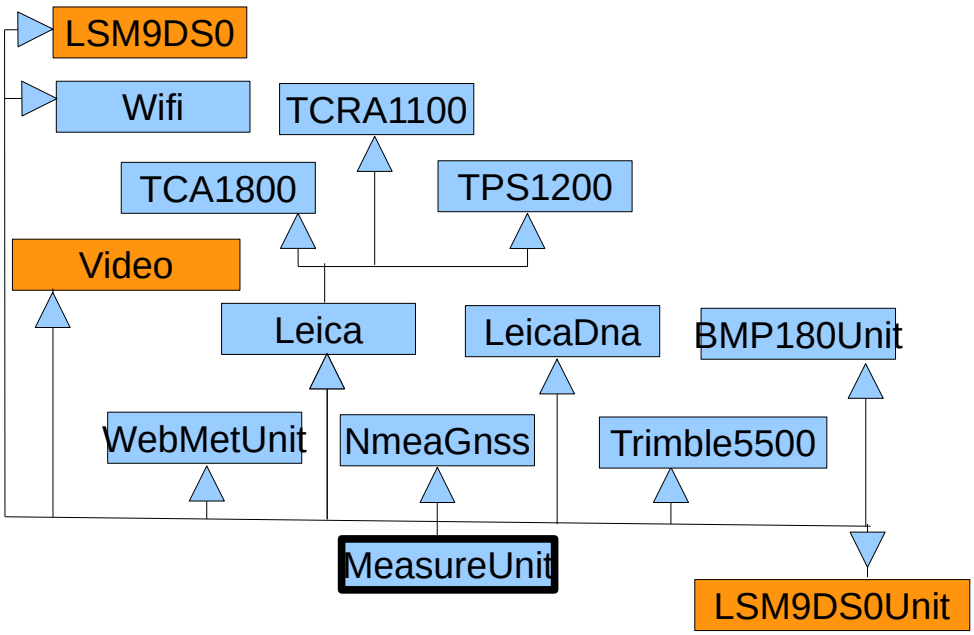
Egy kommunikációs csatorna tervezett állapotban van ez az USB interfész, amely egyszerű USB porton keresztül történő kommunikációra adna lehetőséget.

### *3.1.3 Mérőegységek felépítése [10]*




*24. Ábra: Műszer - számítógép kapcsolat*
*Forrás: Ulyxes rendszer honlapja [10]*

A következőkben a mérőegységeket tárgyalnám ki. Ezek magukat azokat az eszközöket teszik ki amelyekkel a kommunikációt szeretnénk megvalósítani. A létrehozott algoritmusok képesek adatok fogadni és küldeni az eszközök számára, ezen felül értelmezni azokat. Mivel a Python egy objektum-orientált

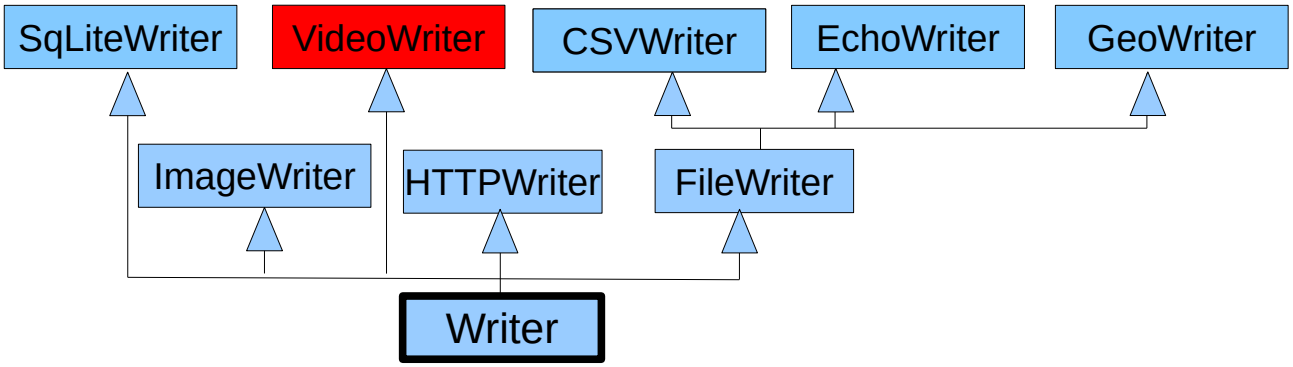



*25. Ábra: Measureunit felépítése*
*Forrás: Ulyxes rendszer honlapja [10]*

nyelv, így a bonyolult parancsokat egyszerűbb magasabb szintű megközelítésből tudjuk végrehajtani. Előnye a metódusnak, hogy műszer csere esetén sem kell módosítani a programon, ha az ugyanazon nyelven értelmezi a parancsokat. Az API könyvtárban található szkriptekek úgy készítették, hogy a bonyolult parancsokat (pl.: %R1Q,2108:3000,0) ne kelljen megjegyezni, csak egyszerűbb, szemléletesebb módon lehessen azokat programozni (pl.: ts.Move(Angle(90, 'DEG'), Angle(85, 'DEG'))). A 24. ábrán látható a bemutatása a kommunikációnak, példának egy Leica mérőállomást mutatnék be.

A meglévő implementált eszközök és műszerek jelen pillanatban a 25. ábrán láthatóak. Ezek között találunk mérőállomásokat, szintezőt és különböző meteorológia adatokat gyűjtő szenzorokat. Ezeken felül megtalálhatóak különböző gyorsulásmérő, wifi, magnetométer és a GNSS vevőkből jövő NMEA üzenetek feldolgozására alkalmas modul is, amely újfent szerves részét képezi a diplomamunkámnak. A számítógépeken lévő vagy külön csatlakoztatható kamerák, webkamerák integrálása is készülő projektek közé tartozik.

### *3.1.4 Writerek és Readerek felépítése [10]*




*26. Ábra: Writerek felépítési*
*Forrás: Ulyxes rendszer honlapja [10]*

Következő részben az opcionális részegységeket elemezném, amelybe a Writerek és a Readerek tartoznak.

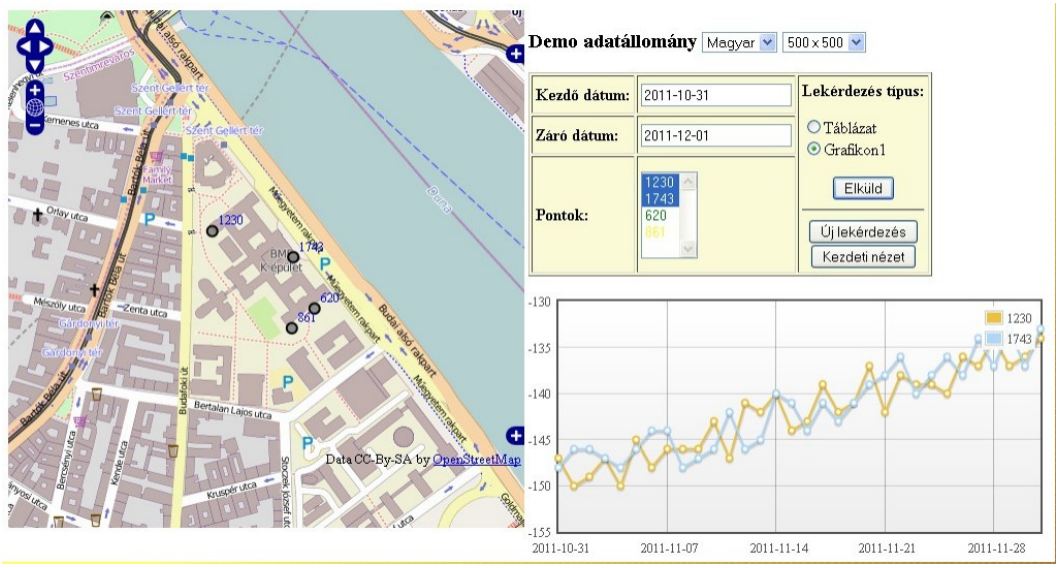
A Writer egység lényege, hogy a már bejövő és/vagy feldolgozott adatokat valamilyen formában tároljuk, megjelenítsük. A legegyszerűbb módja ezek közül az Echo Writer, mely közvetlenül a képernyőre írja ki az eredményeket, ez az újonnan megírt algoritmusok tesztelés során hasznos a fejlesztők számára. Ha a későbbiekben hasznosítani is szeretnénk ezeket a méréseket, akkor már célszerűbb a CSV writer-t vagy a Geo writer-t alkalmaznunk. A különbség a kettő között, hogy


```
{ "log_file": "C:/Users/ASUS/Documents/GitHub/ulyxes-1/data/rp103.log",
  "log_level": 10,
  "station_type": "1200",
  "station_id": "101",
  "station_height": 0.0,
  "station_coo_limit": 0.1,
  "orientation_limit": 0.05,
  "faces": 1,
  "directfaces": 1,
  "fix_list": ["1001", "1002", "1004", "1005"],
  "mon_list": ["1001", "1002", "1003", "1004", "1005"],
  "max_try": 3,
  "delay_try": 0,
  "dir_limit": 0.015,
  "port": "00:12:F3:04:ED:06",
  "coo_rd": "C:/Users/ASUS/Documents/GitHub/ulyxes-1/data/labor1113.coo",
  "coo_wr": "C:/Users/ASUS/Documents/GitHub/ulyxes-1/data/out1.coo",
  "obs_wr": "C:/Users/ASUS/Documents/GitHub/ulyxes-1/data/out1.geo",
  "met_wr": "",
  "inf_wr": "C:/Users/ASUS/Documents/GitHub/ulyxes-1/data/labor_inf1.csv",
  "decimals": 4,
  "gama_path": "C:/GeoEasy/gama-local.exe",
  "stdev_angle": 1,
  "stdev_dist": 1,
  "stdev_dist1": 1.5,
  "dimension": 3,
  "probability": 0.95,
  "blunders": 0
}
```


*27. Ábra: JSON fájl*

a CSV writer egy strukturáltabb, jobban kezelhető állományt hoz létre, amelyet további más szoftverben is felhasználhatunk. A GeoWriter pedig a GeoEasy saját formátumába hozza létre a végeredmény fájlt. A legfontosabb, ha automatizált rendszert szeretnék létrehozni a HTTP writer. Ez a megoldás közvetlen internet kapcsolaton keresztül valamilyen adatbázisba küldi az adatokat. HTTP protokollon át. A megoldás előnye, hogy bárhol, bármikor, bárki számára elérhető lesz a mérési adat. Ezen felül ez a laikusok számára legszemléletesebb módja az eredmények közlésének. A másik nagy prioritást élvező Writer az SqLiteWriter. Ezzel szimultán képesek vagyunk több eszközről jövő adatot egy adatbázis fájlba írni. Az utolsó kimaradt Writer az ImageWriter, mely egy kamera képet képes fájlba menteni. Készülőben van még a VideoWriter, amely mozgóképbe tudja rögzíteni a kimenő információkat.

A Readerek esetén hasonlóan az előzőekhez, több fajta fájlt olvashatunk be a rendszerünkbe, így plusz mérési adatokkal, koordinátákkal, adatbázissal vagy JSON fájlal egészíthetjük ki a rendszerünket. „A JSON (JavaScript Object Notation) egy kis méretű, szöveg alapú szabvány, ember által is olvasható adatcserére. A JavaScript szkriptnyelvből alakult ki egyszerű adatstruktúrák és asszociatív tömbök reprezentálására.” (Idézet a Wikipédiáról) Előnye, ha sok paramétert kell megadnunk egy adott folyamathoz, ezt ezen a fájlon belül megtehetjük könnyen.




*28. Ábra: Kliens megvalósítása*
*Forrás: Ulyxes rendszer honlapja [10]*

Miután a szerverre felkerültek az adatok, azt valamilyen formában szükséges általában megjeleníteni. Ahhoz, hogy bárki számára kódolás nélkül szemléletesen látható legyen a kész végtermék, az Ulyxes-en belül a vékony kliensek alkalmazása lett kialakítva. A 28. ábrán látható ezen alkalmazása, a térképi hátteret az OpenStreetMap szolgáltatja. Internet kapcsolat segítségével ahogy korábban említettem bárki, bárhol elérheti a naprakész adatokat és lekérdezéseket készíthet rajtuk. A pontok helyzete és grafikon is készíthető ezzel módszerrel, így a mozgásvizsgálati pontok szemmel követhetőek. Előnye még szűkíthetjük az időintervallumot és a pontok számát is.

# 4. Teszt környezet kialakítása

Elsődleges feladatom egy olyan teszt környezet kialakítása volt melyben a GNSS vevőkből érkező NMEA üzeneteket valós időben tudjuk feldolgozni. Emellett az összehasonlító elemzés érdekében a többi műszernek is megfelelő környezetet kellett létrehoznom.

## 4.1 NMEA adatok begyűjtése

Az NMEA üzenetek eredetileg egy kombinált elektronikus és adat specifikus kommunikációs mód volt a hajózásban felhasznált navigációs és helymeghatározó eszközöknél, ezek közül a műszerek közül néhányat említenék a teljesség igénye nélkül: szonárok, szélsebesség mérők, visszhangjelzők és GNSS vevők. Az eredeti szabványt az USA-ban lévő Országos Tengerészeti Elektronikai Szövetség hozta létre. 4 verzió van jelenleg használatban, amelyből helymeghatározásra az NMEA 0183-at használjuk. Ezen irányelv egyszerű ASCII formátumú soros kommunikációt használ és azt definiálja hogy az adat hogyan jusson el egy pontból több helyre. Egy példa az NMEA üzenetekre (Wikipédia):

```
$GPGGA,092750.000,5321.6802,N,00630.3372,W,1,8,1.03,61.7,M,55.2,M,,*76
$GPGSA,A,3,10,07,05,02,29,04,08,13,,,,,1.72,1.03,1.38*0A
$GPGSV,3,1,11,10,63,137,17,07,61,098,15,05,59,290,20,08,54,157,30*70
$GPGSV,3,2,11,02,39,223,19,13,28,070,17,26,23,252,,04,14,186,14*79
$GPGSV,3,3,11,29,09,301,24,16,09,020,,36,,,*76
$GPRMC,092750.000,A,5321.6802,N,00630.3372,W,0.02,31.66,280511,,,A*43
$GPGGA,092751.000,5321.6802,N,00630.3371,W,1,8,1.03,61.7,M,55.3,M,,*75
$GPGSA,A,3,10,07,05,02,29,04,08,13,,,,,1.72,1.03,1.38*0A
$GPGSV,3,1,11,10,63,137,17,07,61,098,15,05,59,290,20,08,54,157,30*70
$GPGSV,3,2,11,02,39,223,16,13,28,070,17,26,23,252,,04,14,186,15*77
$GPGSV,3,3,11,29,09,301,24,16,09,020,,36,,,*76
$GPRMC,092751.000,A,5321.6802,N,00630.3371,W,0.06,31.66,280511,,,A*45
```

A „$GP”-t követően kapjuk meg az üzenet fajtáját, számunkra a legfontosabb ezek közül a GGA üzenet, ebben vannak számunkra szükséges adatok:

```
GGA          Global Positioning System Fix Data
092750       UTC idő 09:27:50
5321.6802,N  Szélesség 53°21.6802' N
00630.3372,W Hosszúság 6°30.3372' W
1            Fix quality: 0 = érvénytelen pozíció
                          1 = GPS fix (SPS)
                          2 = DGPS fix
                          3 = PPS fix
                          4 = Real Time Kinematic (RTK fix)
                          5 = Float RTK
                          6 = becsült (dead reckoning)
                          7 = Manuális beviteli mód
                          8 = Szimulációs
08           Műholdak száma
1.03         HDOP
61.7,M       Tengerszint feletti magasság, méterben
55.2,M       Geoidunduláció, méterben
(üres mező)  legutóbbi kapott DGPS üzenet óta eltelt idő, másodperc
(üres mező)  DGPS állomás azonosító száma, ID
*76          checksum adat, mindig „*”-al kezdődik
```

Az Ulyxes rendszerben jelen pillanatban a MeasureUnit-ok között már megtalálható az NmeaGnssUnit, amely GGA üzeneteket képes feldolgozni, emellett a checksum adattal képes ellenőrzést végezni a célból, hogy az átviteli problémák kiszűrhetőek legyenek:

```
    def Result(self, msg, ans):
        """ process the answer from GNSS
            :param msg: MNEA message to get
            :param ans: NMEA message from GNSS unit
            :returns: processed message or None if msg and ans do not match
        """
        res = {}
        if ans[3:len(msg)+3] != msg:
            return None
        # check checksum
        data, cksum = re.split('\*', ans)
        cksum1 = 0
        for s in data[1:]:
            cksum1 ^= ord(s)
        if ('0x' + cksum).lower() != hex(cksum1).lower():
            logging.error(' Checksum error')
            return None
        anslist = ans.split(',')
        if msg == 'GGA':
            # no fix
            if int(anslist[6]) == 0:
                return None
            try:
                hour = int(anslist[1][0:2])
                minute = int(anslist[1][2:4])
                second = int(anslist[1][4:6])
                if len(anslist[1]) > 6:
                    ms = int(float(anslist[1][6:]) * 1000)
                else:
                    ms = 0
                d = date.today()
                res['datetime'] = datetime(d.year, d.month, d.day, hour, minute,
second, ms)
                mul = 1 if anslist[3] == 'N' else -1
                res['latitude'] = Angle(mul * float(anslist[2]), 'NMEA')
                mul = 1 if anslist[5] == 'E' else -1
                res['longitude'] = Angle(mul * float(anslist[4]), 'NMEA')
                res['quality'] = int(anslist[6])
                res['nsat'] = int(anslist[7])
                res['altitude'] = float(anslist[9])
                res['hdop'] = float(anslist[8])
            except:
                logging.error(" invalid nmea sentence: " + ans)
                return None
        return res
```

Ezt a MeasureUnit-ot felhasználva létrehoztam egy szkriptet, amely valós időben soros porton vagy bluetoothon keresztül képes NMEA adatokat fogadni. A megvalósított kód CSV fájlba menti az adatokat:

```
import re
import sys
import msvcrt
import time

sys.path.append('../pyapi')

if __name__ == '__main__':
    # Choose between serial or bluetooth communication
    if len(sys.argv) > 1:
        cm = sys.argv[1]
    else:
        cm = '/dev/ttyUSB0'
    if re.search('^COM[0-9]$', cm) or re.search('^/dev/ttyUSB[0-9]$', cm):
        from serialiface import SerialIface
        iface = SerialIface('test', cm, 19200)
    else:
        from bluetoothiface import BluetoothIface
        iface = BluetoothIface('test', cm, 1)

    from nmeagnssunit import NmeaGnssUnit
    from csvwriter import CsvWriter
    from filewriter import FileWriter
    from gnss import Gnss

    #Making measurement unit
    mu = NmeaGnssUnit()

    #Writer unit creating
    if len(sys.argv) > 2:
        fn = sys.argv[2]
    else:
        fn = 'results.csv'
    wrt = CsvWriter('', 'DEG', '.3f', '%Y-%m-%d %H:%M:%S', ['id', 'latitude',
'longitude', 'altitude', 'datetime'], fn)

    #Get NMEA data
    g = Gnss('', mu, iface, wrt)
    if g.measureIface.state == g.measureIface.IF_OK:
        print('Press any key to finish measuring')
    while g.measureIface.state == g.measureIface.IF_OK:
        g.Measure()
        if msvcrt.kbhit():
            if msvcrt.getwche() == '\r':
                break
        time.sleep(0.1)
```

Az API segítségével egy GNSS eszközt hozhatunk létre az előzőekben tárgyalt három elemből: MeasureUnit, WriterUnit, Interface. Az applikáció konfigurásánál először a portot kell megadnunk amin az adat érkezik, ez lehet a COM port száma vagy a bluetooth eszköz (jelen

esetben a GNSS vevő) MAC címe. A második paraméter helyére a fájl neve kerül megadásra ahova az adatokat menteni szeretnénk.

## 4.2 Bluetooth interface kialakítása

Az Ulyxes egy új kommunikációs módon képes kommunikálni az eszközökkel, műszerekkel amelyekből adatot szeretnénk kinyerni. A bluetooth technológiát a geodéziában is széles körűen használják mérőállomásokban és GNSS vevőkben egyaránt, így fontossá vált egy ilyen jellegű vezeték nélküli kapcsolati mód kialakítása. A 2014-ben történő Python programozási nyelvre való váltás ezt is megkönnyíti, ugyanis a Pythonban kész bluetooth kommunikációs modul áll rendelkezésre, így az Ulyxes-be való beillesztés is zökkenőmentesen történhetett, az új Interface a már meglévőkhöz nagyon hasonlóan objektum-orientáltan készült el. Az elkészült programkódból bemutatnám hogyan olvassa az algoritmus az érkező adatokat:

```
    def GetLine(self):
        """ read a line from bluetooth
        """
        if self.socket is None or self.state != self.IF_OK:
            logging.error(" bluetooth connection not opened or in error state")
            return None
        # read answer till end of message marker
        ans = ''
        w = -1 * len(self.eomRead)
        while ans[w:] != self.eomRead:
            ch = ''
            try:
                ch = (self.socket.recv(1)).decode('ascii')
            except:
                self.state = self.IF_READ
                logging.error(" cannot read bluetooth connection")
                break
            if ch == '':
                # timeout exit loop
                self.state = self.IF_TIMEOUT
                logging.error(" timeout on bluetooth")
                break
            ans += ch
        # remove end of line
        logging.debug(" message got: %s", ans)
        ans = ans.strip(self.eomRead)
        return ans
```

Ha a soros porton történő kommunikáció algoritmusának ezen részét megnéznénk nagyon hasonló parancsokból áll a kód. Az egész bluetooth interfészt elemezve egyedül a port megnyitása folyamán találunk nagy különbséget, ebből is látszik, hogy a hierarchikus programozás során, ha jól megalkotott kódot hozunk létre, azok különbségei elhanyagolhatóak lesznek.

## 4.3 Felhasznált műszerek, eszközök

A teszt környezet kialakítása során több típusú műszert használtam fel ezeket fejteném ki a következőkben. A későbbiekben a teljes teszt végrehajtása során ez tovább bővült.

**Topcon Hiper II [16][17]**

A Topcon Japán cég 1932-es (Wikipédia) alapítása óta gyárt geodéziai célú műszereket. 1970-től emellett van egy másik ágazat amelyben nagy szerepet töltenek be, ez a szemészeti műszerek gyártása. A tanszéken található kettő Topcon Hiper családból származó műszer. A munkáim során a Hiper II-es típusút alkalmaztam, amely 2010-ben forgalomba hozott GNSS vevő.


*29. Ábra: Topcon Hiper II*
*Forrás: xpertsurveyequipment.com [17]*

Tulajdonságai tekintetében egy két frekvenciás GPS+Glonass képes műszerről beszélünk. Amely SIM kártya fogadására is képes és bluetooth kapcsolat létrehozására is alkalmas. Ezek mellett SD vagy SDHC kártya foglalatot is tartalmaz, amelyre statikus mérési adatokat rögzíthetünk. Adatformátumok közül az RTCM, CMR, CMR+, NMEA, TPS képes kezelni és küldeni akár soros akár bluetooth csatornán keresztül maximum 20 Hz-es gyakorisággal.

## Leica TCRP1201+ [11]

Mérőállomások tekintetében a tanszéken található Leica TCRP1201+ típusú műszert használtam fel a munkáim során a monitoring rendszerben történő mérésekre. Előnye ezen típusú eszköznek, hogy SmartStation-ként is képes funkcionálni ami azt jelenti, hogy akár egy GNSS vevőt is szerelhetünk rá vagy közvetlen a mérendő prizmára, ezzel a kombinációval mindkét technológia előnyeit kihasználhatjuk.

*30. Ábra: Leica TCRP1201+*

A műszer szögmérési pontosság 1” és 5” között variálhatóak, a távmérési pontossága prizmára 1 mm + 1,5 ppm, prizma nélküli távmérés esetén 2 mm + 2 ppm. Munkám szempontjából fontos, hogy támogatja az RS-232 szabvány és bluetooth technológiát egyaránt. Továbbá elengedhetetlen tulajdonságai közé tartozik a monitoring rendszerben történő felhasználáshoz az ATR és a szervomotorok megléte. Ezek mellett Powersearch funkció is található benne, amely képes megtalálni a közelében található prizmákat.

## Raspberry Pi [11] [18]

A Raspberry Pi eredetileg egy oktatási célra kifejlesztett kis méretű számítógép. A 2009-ben alapított Raspberry Pi Foundation szervezet hozta létre az Egyesült Királyságban. Támogatóként a Cambridge-i egyetem és a Broadcom cég társult hozzájuk abból a célból, hogy minél szórakoztatóbb módon tanulhassák az informatikát. Az első célok között szerepelt a más eszközök irányítása, a kis méret, alacsony fogyasztás, de emellett megfelelő teljesítménnyel rendelkezzen. Az eredeti programozási nyelve a Scratch volt. (Bánhidi, 2016)

A jelenben a legújabb típus amely a tanszéken is található a Raspberry Pi 4-es verziója. Ez az apró számítógép a ma használatos portok nagy részét hasznosítja. Található rajta USB 3.0, Ethernet port, AUX, USB-C és HDMI csatlakozó is. A legújabb modell akár 4 GB RAM-mal is választható már. Az operációs rendszere a Debian alapú Linux rendszer, a Raspbian. Programozása az előzőekben említett Scratch és a népszerűbb Python nyelv segítségével lehetséges.

*31. Ábra: Raspberry Pi 4*
*Forrás: Wikipédia [18]*

Munkám során az Ulyxes rendszerbe a közel múltban implementált kamera modult alkalmaztam. Az optikai eszköz egy szalagkábelen csatlakozik az alaplapba a Camera CSI-n keresztül. A szenzort egy mérőállomásra erősítve, feltételezve, hogy az mozdulatlan, a rögzített képekből kinyerhetőek az elmozdulási értékek rövid idejű, dinamikus mozgások esetén is.

*32. Ábra: Raspberry Pi kamera modulja*

**Hi-Target V100 [19]**

Egy saját tulajdonú kínai GNSS vevőt is alkalmaztam a tesztjeim során. Mivel ez egy viszonylag új gyártású műszer, így a ma elérhető szinte összes műholdrendszerről jövő adatot képes fogadni (GPS, GLONASS, Galileo, BeiDou, QZSS, stb.).


*33. Ábra: Hi-Target V100*
*Forrás: trends.directindustry.com (https://trends.directindustry.com/hi-target-surveying-instrument-co-ltd/project-161167-145000.html )*

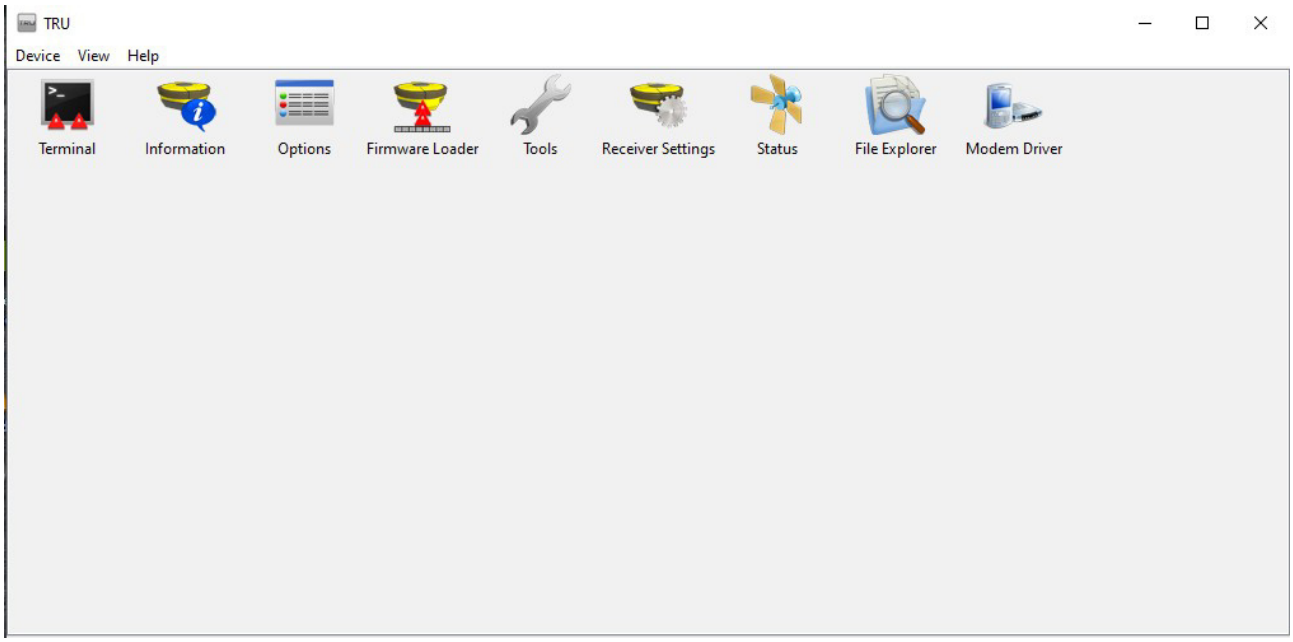
Korszerű bluetooth 4.0, NFC csatlakozás és Micro-USB, valamint soros port is megtalálható a készülékben. Hasonlóan a Topcon vevőjéhez 1 Hz-től 20 Hz-ig képes a pozíciókat a mai legtöbb adatformátumban szolgáltatni (Hi-Target): ASCII NMEA-0183: GSV, AVR, RMC, HDT, VGK, VHD, ROT, GGK, GGA, GSA, ZDA, VTG, GST, PJT, PJK, BPQ, GLL,GRS, GBS. A munkám során ezt a képességét használtam ki legfőképp, hogy az NMEA-0183 szabványt is támogatja. Hátránya a vevőnek, hogy a tanszéki GNSS vevővel ellentétben ebbe SIM kártya nem helyezhető, így mindig szükség van egy vezérlőre vagy egyéb más módra, amely módon megkapja a korrekciókat RTK üzemmódban. SD vagy egyéb más szabványú kártyafoglalat sem kapott helyet a műszerben, viszont egy 8 GB-os belső memória megtalálható benne, így statikus mód esetén a nyers mérési adatokat tudja arra rögzíteni.

# 5. Előzetes tesztelés

## 5.1 GNSS előzetes tesztelése

Az előzetes tesztelés első szakaszában a megírt NMEA üzenetek feldolgozására használt szkriptet teszteltem. Ezt a már előzőekben említett Topcon Hiper II-s műszeren tettem meg. Labor körülmények között a tanszéki komparátor teremben tettem kísérletet az üzenetek kiolvasására. Mivel itt természetesen pozíciót nem kaptam, csak arra volt jó, hogy lássam a megfelelő kommunikáció megvalósult és érkeznek-e egyáltalán adatok valamilyen formában.

A műszer beállítását a Topcon Receiver Utility, ingyenesen elérhető szoftver segítségével tettem meg, amely mind a műszerhez tartozó vezérlőn, mind személyi számítógépen megtalálható. Ha a számítógép USB portját a vevő soros portjával összekötjük, akkor ezen keresztül kommunikálhatunk azzal.




*34. Ábra: Topcon Receiver Utility*

Itt a 34. ábrán látható „Receiver Settings” menüpont alatt beállíthatjuk, hogy az RS-232 szabványon át milyen adatok, milyen gyakorisággal érkezzenek. Ezen felül a kommunikáció konfigurálása (sebesség, paritás, adat bitek) is itt hajtható végre.

A műszer a, b, c, d porton keresztül képes kommunikálni ezek közül kettőn, az „a”-n és a „d”-n állítható NMEA kimenet, a többihez nincs hozzáférésünk. Ha bluetoothon szeretnénk ezeket az adatokat megkapni, azt is itt tehetjük meg, ha egyszerre a kettő konfigurálható kimeneten állítjuk be a formátumot.

Az egyetem udvarán, a K és CH épületek között történt az első éles tesztelés. Itt csak DGPS üzemmódban alkalmaztuk a vevőt, RTK mód használata nem volt létfontosságú a szoftver és a kommunikáció teszteléséhez.

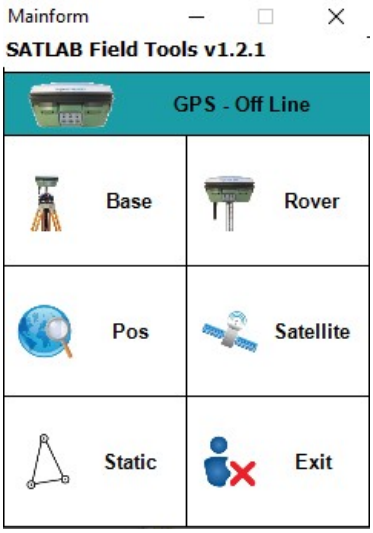
*35. Ábra: K és CH épület közötti mérés*

A 35. ábrán látható a tesztről két kép. A bal oldalin a soros porton történő adatközlés, még a jobb oldalin bluetoothon keresztül történő kommunikációról található fotó. A szkript itt Echo Writer segítségével íratta ki az adatokat a képernyőre, ugyanis nem volt szükségünk a továbbiakban a felállított műszer koordinátáira.

Ha jobban megnézzük a két képet láthatóak a kijelzőn a koordináták, így elmondható, hogy az algoritmus megfelelően működött és az Ulyxes-be alkalmazhatóan implementáltuk. Ezt bármilyen más felhasználásra kisebb módosításokkal (pl.: writer megváltoztatása) alkalmazni lehet.

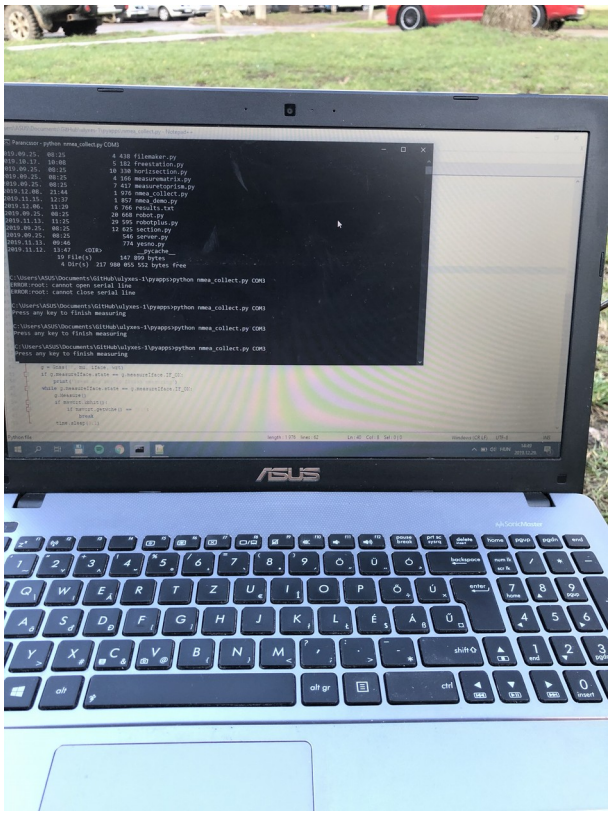
Egy másik műszert is sikerült implementálni és tesztelni a rendszerrel. A Satlab SL500-as GNSS vevőjéből nyertem ki adatokat az *nmea_collect.py* szkript segítségével.

A nehézséget itt is a műszer beállítása jelentette. Ezt a SATLAB Field Tools szoftverével tettem meg a számítógépemről. A




*36. Ábra: SATLAB Field Tools*

programban beállíthattam, hogy a vevő melyik COM portján milyen adat, mekkora sebességgel érkezzen.




*37. Ábra: Adatok fogadása*

*38. Ábra: Terepi mérés*

További nehézséget jelentett, hogy a meglévő Y kábel egyik vége USB, míg a másik RS-232 port volt, de az előbbin nem jött adat, így ennek kiküszöbölésére egy átalakítót alkalmaztam RS-232-ről USB-re. Az eredmények a következő szövegdobozban láthatóak:

```
0;47.96415202766666;21.71206911066667;110.513;134835.000
1;47.964152049999996;21.71206948466667;110.515;134836.000
2;47.96415208416668;21.712069688333337;110.519;134837.000
3;47.96415174566667;21.712068054833335;110.678;134838.000
4;47.96415173583334;21.712068346166667;110.700;134839.000
5;47.964151492000006;21.712067405000003;110.805;134840.000
6;47.96415137983333;21.7120672675;110.890;134841.000
7;47.964151192500005;21.71206663983333;110.994;134842.000
8;47.96415104183334;21.712066363166663;111.093;134843.000
9;47.9641511485;21.712067423666667;111.062;134844.000
10;47.96415125;21.712067939666667;111.027;134845.000
11;47.96415132833334;21.712068140666666;111.001;134846.000
12;47.96415135783333;21.712068184833335;110.974;134847.000
13;47.964151378000004;21.71206818916667;110.949;134848.000
14;47.964151401333325;21.71206820583333;110.920;134849.000
15;47.964151435;21.71206807416667;110.908;134850.000
16;47.964151456;21.712068037999998;110.903;134851.000
17;47.96415146283333;21.712068024166665;110.897;134852.000
```

## 5.2 Robot mérőállomás előzetes tesztelése [11] [15]

Az Ulyxes-ben már készen létrehozott alkalmazás van a mérőállomások automatizált monitoring feladatainak ellátására. Ezt a Robotplus nevű program hajtja végre. Ez a szkript már korábban meghatározott koordinátájú prizmákra képes méréseket végezni, ha az álláspont is ismert. A működése a következőképpen zajlik:

1) Elsőként az alkalmazás a korábban említett JSON fájlból olvassa ki az adatokat. (Részletek a továbbiakban.)
2) Másodjára a program a meteorológiai adatok kéri le, ha ez elérhető. Ezt jelen pillanatban az internetről vagy a 2.2.4-es fejezetben is említett Adatfruit vagy a Sensehat szenzorán keresztül képes.
3) Ezt követően a koordináták beolvasása következik. Itt az álláspont, a FIX és a MON (mozgásvizsgálati) pontok attribútumait tölti be a konfigurációban megadott adatforrásból.
4) A pontok és az álláspont között az összes vízszintes és magassági szög kiszámításra kerül.
5) Ha a műszer látómezejében található prizma (ATR-es műszer elengedhetetlen), akkor azt megmérve a program kitalálja a zenitszög és távolság alapján mely célra történt a mérés. Egyéb esetben, ha nincs prizma, Powersearchet alkalmaz és egy megtalált prizmára végez mérést majd az előzőhöz teljesen hasonló módon végzi el a cél beazonosítását. Ez alapján a közelítő tájolása a műszernek megtörtént.
6) Minden mérés alkalmával új szabadálláspont létesítést végez az alkalmazás, ezzel kiszűrve az álláspont esetleges mozgásának hatását. A FIX pontok észleléseit és koordinátáit GNU Gama szoftverének tovább küldve elkészülnek a kiegyenlített álláspont koordináták és a tájékozási szög. Itt egy durva hiba szűrés is lefut, így kiszűrve a FIX pontok bármilyen nemű elmozdulását vagy félreazonosítását.
7) Az új kiegyenlített értékekkel kiszámítja az új irányszögeket és zenitszögeket a MON pontok felé.
8) Ezt követően végig méri az össze mozgásvizsgálati pontot. Ha egy pontot nem talál akkor beállítástól függően valahányszor újrapróbálja, majd sikertelenség esetén kihagyja azt.
9) A mérési eredményekből kiszámítja a pontok koordinátáit. Ezt beállítástól függően vagy a szerverre vagy egy adatbázis fájlba vagy GEO fájlba menti a nyers mérési adatokkal és az álláspont koordinátáival együtt.

Konfigurálni a korábban említett JSON fájl segítségével lehet, egy példa erre:

```
1  { "log_file": "C:/Users/ASUS/Documents/GitHub/ulyxes-1/data/teszt/tesztlll4.log",
2    "log_level": 10,
3    "station_type": "l200",
4    "station_id": "100",
5    "station_height": 0.0,
6    "station_coo_limit": 0.1,
7    "orientation_limit": 0.05,
8    "faces": 1,
9    "directfaces": 1,
10   "fix_list": ["101", "102", "103", "104"],
11   "mon_list": ["1001", "1002", "1003", "1004", "1005"],
12   "max_try": 3,
13   "delay_try": 0,
14   "dir_limit": 0.015,
15   "port": "COM3",
16   "coo_rd": "C:/Users/ASUS/Documents/GitHub/ulyxes-1/data/teszt/tesztlll4.coo",
17   "coo_wr": "C:/Users/ASUS/Documents/GitHub/ulyxes-1/data/teszt/out.coo",
18   "obs_wr": "C:/Users/ASUS/Documents/GitHub/ulyxes-1/data/teszt/out.geo",
19   "met_wr": "",
20   "inf_wr": "C:/Users/ASUS/Documents/GitHub/ulyxes-1/data/teszt/teszt_inf.csv",
21   "decimals": 4,
22   "gama_path": "C:/GeoEasy/gama-local.exe",
23   "stdev_angle": 1,
24   "stdev_dist": 1,
25   "stdev_distl": 1.5,
26   "dimension": 3,
27   "probability": 0.95,
28   "blunders": 0
29 }
```

*39. Ábra: JSON fájl*

Az 1. sorban a *log_file* pont segítségével adhatjuk meg a log fájlunk mentési útvonalát. A másodikban ennek a szintjét konfigurálhatjuk, az-az megadhatjuk mennyi adat kerüljön ebbe.

A 3. sorban mérőállomásunk típusát adhatjuk meg, természetesen azok közül választva amelyeket implementáltak az Ulyxes-be. A következő pontban álláspontunk számát adjuk meg, ennek koordinátáit a 16. sorban lévő fájlból fogja kiolvasni.

Az 5. sorban a műszermagasságot konfigurálhatjuk, az ezt követő *station_coo_limit* és *orientation_limit* pontban álláspontunk koordinátáinak eltéréseit állíthatjuk be, ha ennél nagyobb eltérés értéket kapunk figyelmeztet a rendszer. A *faces* és *directfaces* pontban a távcsőállások számát adhatjuk meg, az első esetben első távcsőállásban méri meg az összes pontot majd második távcsőállásban visszafele, az utóbbi pedig egy pontot azonnal két távcsőállásban észlel.

A 10. és 11. sorban a prizmák pontszámát adhatjuk meg. Az előbbi a fix pontokat, az-az azokat amelyekből a szabad álláspont létesítés fog megvalósulni, az utóbbi pedig a mozgásvizsgálati pontok számát jelzi.

A *max_try* esetén a maximum próbálkozások számát, míg a *delay_try* számánál a késleltetés idejét adhatjuk meg másodpercben. A port szám a számítógépünk USB portját jelzi amelyen a műszerrel van összekapcsolva.

A *coo_rd, coo_wr, obs_wr* sorokban az útvonalát definiáljuk a fájlok amelyekből vagy kiolvassuk a koordinátákat vagy írjuk. A *met_wr* a meteorológia adatok írására szolgáló fájl útvonalát jelzi. A következő sorban lévő *inf_wr* pedig az infó fájl írásának helyét jelzi.

A decimals pontban az adatok élességét adhatjuk meg. A 22. sorban a GNU Gama kiegyenlítő szoftver útvonalát adhatjuk meg, amely a szabadálláspont kiegyenlítését végzi.

Az *stdev_angle, stdev_dist, stdev_dist1* soraiban a szórását adhatjuk meg a mért irányokhoz (másodpercben), a távolsághoz (milliméterben). Ezt követően a *dimension* pontban a dimenzióját adjuk meg a mérésünknek. Majd a kiegyenlítéskor felhasznált konfidencia-intervallum értékét határozhatjuk meg a *probability* pontban. A *blunders* pedig a durvahiba szűrésre vonatkozó paraméter, 1 = van hibaszűrés, 0 = nincs hibaszűrés.

*40. Ábra: Mini prizmák*

Ha például a 16. sorban található *coo_rd* helyén egy olyan coo fájl elérési helyét írjuk be melyben megtalálható a *fix_list* és *mon_list* pontjai ebben az esetben az automatikusan megméri azokat. Előzetes tesztelésem során ezt megtettem a tanszék komparátor termében. Mini prizmákat alkalmaztam és ebből kitüntettem 4-et amely „fix”-nek vehető és 5-öt véletlenszerűen, amely „mozgásvizsgálati pont”.

Először a *Coomaker* nevű alkalmazás segítségével kézzel megirányozva az összes pontot létrehoztam egy *coo* fájlt amelyben az összes prizmának a koordinátája szerepelt. Itt fontos, hogy a műszer tájékozva legyen. A konfigurációs JSON fájl-t ezt követően hoztam létre. Ha az előzőekben tárgyalt összes paramétert beállítottuk, kezdődhet a mérés.

Ha a műszer érzékeli, hogy egy prizma található a látómezejében a mérés indításakor akkor erre fog történni az első észlelés. Ez alapján az algoritmus megpróbálja kitalálni, hogy melyik prizmát látja a zenitszög és a távolság alapján. Először a fix pontokat mérjük meg, majd azok eredményeit a GNU gama kiegyenlítő szoftvernek átadva egy kiegyenlített koordinátájú szabad álláspont jön létre. Ha ez minden tűréshatárnak megfelel, a szkript tovább ugrik a mozgásvizsgálati pontok mérésére. Amint az észlelések befejeződtek és zavartalanul megmérhető volt minden előállnak a koordináta fájlok és a GeoEasy által olvasható *geo* fájl.

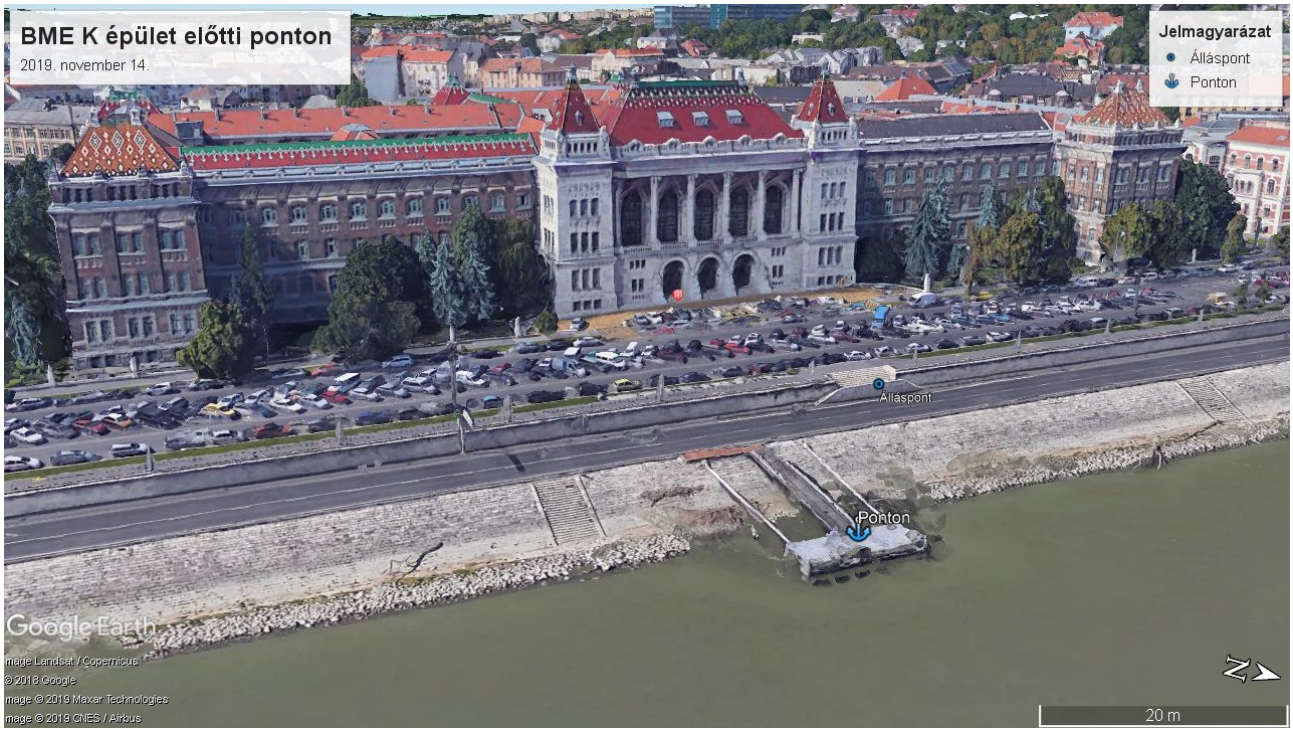
Ezt automatizálni Windows-on az ütemezett feladatok (scheduled tasks) nevű programmal lehet. Megadhatjuk, hogy milyen időközönként melyik programunk fusson le a számítógépünkön. Ha létrehozunk egy batch fájlt a paranccsal mellyel indítottuk a Robotplus-t, akkor a mérések automatikusan meghatározott időpontokban elindulnak.

Mivel az 1200-as műszerben található bluetooth adapter is, így az egész műveletet leteszteltem ezen kommunikációs mód segítségével is. Elmondható, hogy teljesen zavartalanul működött a parancsok és adatok küldése és fogadása a két eszköz között. Egy különbség viszont észrevehető volt a kábeles és a vezeték nélküli megoldás között, ez a sebesség. Míg a műszer és a számítógép lekommunikálta egymás között a parancsokat érezhetően lassabb volt, körülbelül háromszor annyi időt vett igénybe egy mérési sorozat végrehajtása bluetooth kommunikáció segítségével.

Végeredményben a mérőállomás automatizált monitoring rendszer Ulyxes-ben történő felhasználásáról elmondható, hogy teljesen implementált és felhasználható. Terepen kitelepített műszerrel és hozzákötött számítógéppel a mozgások kimutathatóak automatizáltan.

*41. Ábra: Komparátor teremben történő mérés*

## 5.3 Komplex tesztelés




*42. Ábra: November 14. mérés*
*Forrás: Google Earth*

Előzetes tesztelésem során fontos volt, hogy egy komplex tesztet is végrehajtsak, amelyben egyszerre több módon és eszközzel dolgozom szimultán. Ezt egy alkalommal végre is hajtottam az egyetem előtt található pontonon. Választásom azért esett egy pontonra, mert itt a várható mozgások rövid idő intervallumon belül is nagyobbak.

Az előzőekben ismertetett két mérési módszert egy harmadikkal kiegészítettem, a Raspberry Pi kamera moduljával. Ezt a technológiát hatékonynak feltételezhetjük ebben az esetben, ugyanis itt rövid idő alatt (akár másodpercek) történnek a mozgások, így például egy robot mérőállomás által végrehajtott mérés, néhány pont esetén ami körülbelül 1-2 percet vesz igénybe nem feltétlenül tudja kimutatni az ilyen rövid időintervallumú mozgást.

### 5.3.1 Mérés előkészítése


*43. Ábra: Álláspont és a mért pontok közelítő helyei*
*Forrás: Google Maps*

Három technológia alkalmazásával kezdtem el a mérést, a 43. ábrán látható a pontok elrendezése. A helyi hálózatot közel északi tájolásúnak vettem fel és a 100-as ponton létesítettem állást. A 101 és 102 pontok voltak a tájékozó pontok, az-az azok amelyekből a szabad álláspont létesítést elvégeztem. Az 1001, 1002, 1003, 1004, 1005 pontok voltak a mozgásvizsgálati pontok. Az 1001-es számú ponton helyeztem el a GNSS vevőt, egy mini prizmát és egy ArUco jelet.

*44. Ábra: 1001-es pont*

Az ArUco jel hasonló egy QR kódhoz, 4x4 négyzet tesz ki fekete-fehér színben egy jelzést. Ennek detektálására megírt OpenCV alapú algoritmusok elérhetőek. Ez pixel alapon tudja meghatározni az adott kód helyzetét a képen, ezt metrikussá a jel lemérésével tehetjük. Ha tegyük fel egy videót rögzítünk, amelyben másodpercenként 25 képkocka van és jól látható rajta az ArUco jel, akkor minden képkockán meghatározható a helyzete és ebből számítható az elmozdulás.

*45. Ábra: Felállított monitoring rendszer*

Mint az előzőekben már említettem az 1001-es ponton a Hi-Target V100-as vevő üzemelt RTK üzemmódban automatizált NMEA üzenet kimenettel. Ez 1 Hz-es gyakorisággal közölte a pozíciókat. Mivel a műszerláb szerelt prizma, GNSS vevő és ArUco kód egyszerre mozog, így a különböző módszerekkel meghatározott pozíciók összehasonlíthatók.

A következőkben a robot mérőállomást állítottam fel és létrehoztam egy közel északi irányba tájolt koordináta-rendszert, amelyben a műszer koordinátái Y = 100.000 m, X = 100.000 m, Z = 100.000 m voltak. A *Coomaker* program segítségével méréseket végeztem a 7 darab prizmára. Ebből előállt a koordinátajegyzék a kezdő koordinátákkal. Ezt követően a JSON fájl konfigurálása következett a korábban tárgyaltak szerint és így futtatva a Robotplus szkriptet készen állt a mérőállomás által támogatott automatizált monitoring rendszer.

A Raspberry Pi kameráját egy Leica 305 műszerre szereltem fel. Mivel a kis számítógéphez kijelzőt és billentyűzetet nem kapcsoltam hozzá, így SSH (Secure Shell) protokollon keresztül távoli eszközként vezéreltem egy laptopról. Ehhez routert és ethernet kábelt is vittem ki a helyszínre. A WiFi-n történő adatközlés nem bizonyult korábban elég gyorsnak ahhoz, hogy az élességet belehessen állítani a kamera optikájához. Vezeték nélküli kapcsolat esetén körülbelül 15-16

másodperc telt el mire a kijelzőn megjelent az eredmény. Ennek kalibráláshoz egy shell szkriptet alkalmaztam, amely hálózati adatfolyamra töltötte fel az „élő” képet. Abban az esetben ha kábellel volt a rendszer összekötve, ekkor is tapasztalható volt némi késés (kb. 5 másodperc), de ez nem nehezítette meg a munkát. Ezt a VLC Media Player segítségével tudtam megnyitni.

*46. Ábra: GNSS vevő, ArUco jel és mini prizma az 1001-es ponton*

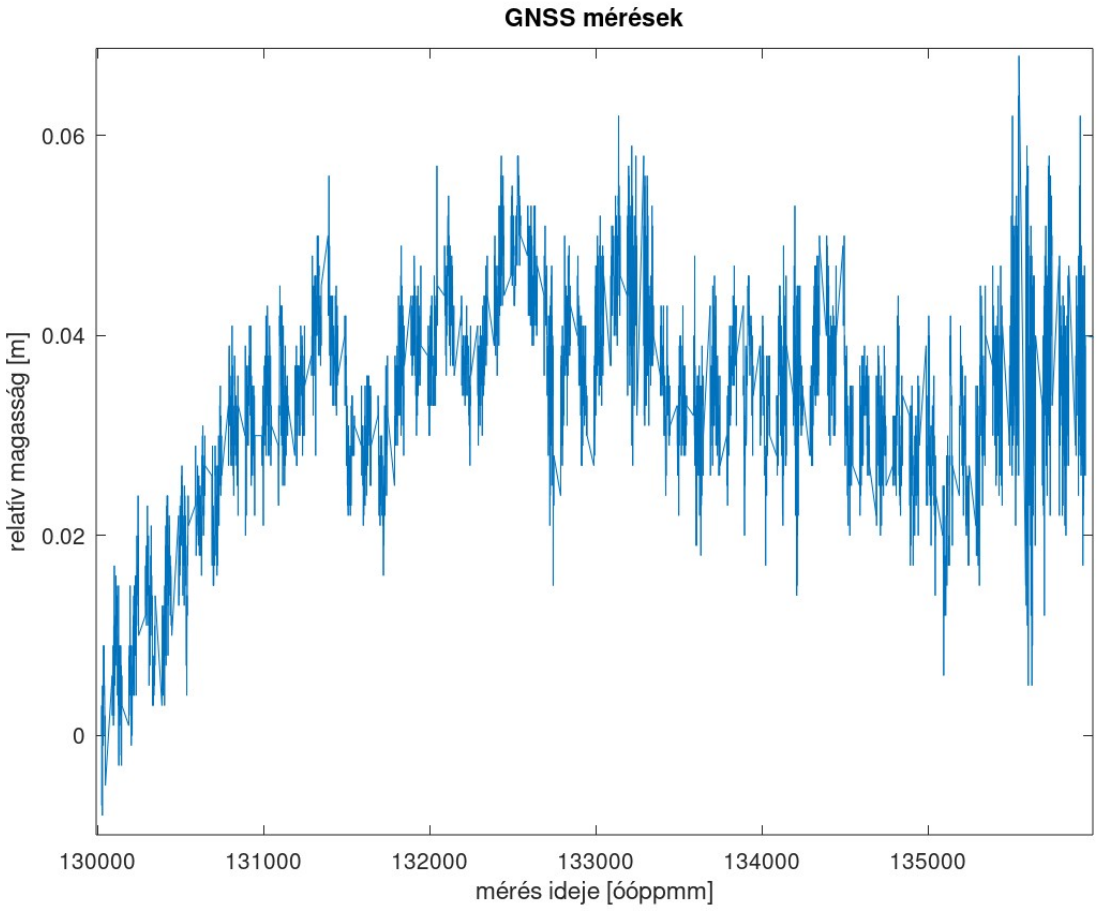
Az élesség beállítását követően, már csak a videó rögzítés elindítása következett, amelyet 1000 x 1000 felbontásban és 25 képkocka/másodperc beállítás mellett indítottam el.

### *5.3.2 Mérés kiértékelése*

**GNSS**

A mérések feldolgozását a GNSS-ből kapott adatokkal kezdtem. Az ebből kinyert koordinátákhoz tartozó időbélyegek, mivel GGA üzenetekből lettek előállítva, UTC időben voltak, ami a helyszínen történt megállapítás alapján +1 óra (ez az érték az időzóna miatt) és -12 másodperc volt a különbség a számítógép órájához képest, amely a mérőállomás adatait rögzítette. Ezt hogy közös időrendszerbe legyenek a méréseim, az adott értékkel eltoltam, így előálltak a mérési eredmények Balti (EOMA) tengerszint feletti magasságai, mivel a vevő a mérés ideje alatt RTK üzemmódban rögzítette az eredményeket az EHT 2014 alapján. Relatív magasságkülönbségek létrehozása

érdekében minden megkapott értékből az elsőt kivontam, ezt vettem a nulla mérésnek. Az adatok feldolgozására létrehoztam egy Octave szkriptet, amely beolvassa, majd a megfelelő számításokat elvégzi az adatokon és megjeleníti azokat. A kapott értékek a következők lettek 13 és 14 óra között:




*47. Ábra*

Mivel az RTK mérési módszerrel a magassági értelemben vett középhiba 8 – 10 milliméterre csökkenthető, így a 3 szigma szabály szerint, 99,7%-os megbízhatósági szint esetén ~2,5 cm-re lehet a pont magasságát meghatározni. Az ábráról levehető, hogy rövid időintervallumokat nézve is nagy volt a szórása a pontok magasságának, de különböző rövid idejű trendek megfigyelhetőek a mérések alatt.

### Robot mérőállomás

A pontonon kihelyezett 5 darab prizma szolgált mozgásvizsgálati pontként és két fix prizma alapján történt a szabad álláspont meghatározás. A kapott adatok COO és GEO kiterjesztéssel álltak rendelkezésemre, ebből egy koordinátalistát hoztam létre a GeoEasy szoftver segítségével. Ennek feldolgozására és megjelenítésére egy újabb Octave szkriptet hoztam létre. Ezt úgy írtam meg, így hogy bármennyi pont automatizált feldolgozására alkalmas legyen.

```
clc; clear all; close all;

%meresi adatok levalogatasa
data = dlmread("out_coo.csv", ",");
points = unique(data(:,1));
pieces_temp = histc(data,points);
pieces = pieces_temp(:,1);

time = dlmread("time.txt");

for k = 1:length(points)
  first = sum(pieces(1:k-1))+1;
  last = sum(pieces(1:k));
  one_point = data(first:last,:);
  figure(k)
  for i = 1:length(one_point)
    if points(k) = one_point(i,1)
      h(i) = i;
      temp(i) = one_point(i,4) - one_point(1,4);
      plot(i,temp(i), 'o-r');
      hold on;
      text(i,temp(i)-0.0005,num2str(time(i)))
      xlabel ("mérés száma");
      ylabel ("magasság [m]");
      cim = sprintf("%d számú pont robot mérőállomással mért magassága",points(k));
      title (cim);
      hold on;
    endif
  endfor
  fit = splinefit(h,temp,15);
  val = ppval (fit, h);
  plot(h, (val - val(1)));
  hold on;
endfor
```

A mérési adatok beolvasását követően csak azok megjelenítését céloztam meg feladatként. Ezt automatizáltan teszi meg az algoritmus minden egyes mért pontra és külön ábrán jeleníti meg azokat. A kód végén pedig egy Spline illesztést hajtok végre egyesével a mozgásvizsgálati prizmákra. Ezt csak a GNSS-el történő összehasonlítás miatt kellett megtennem. Ez az illesztett görbe 15 szakaszból és 16 töréspontból állt.

*48. Ábra: Mozgásvizsgálati prizma*

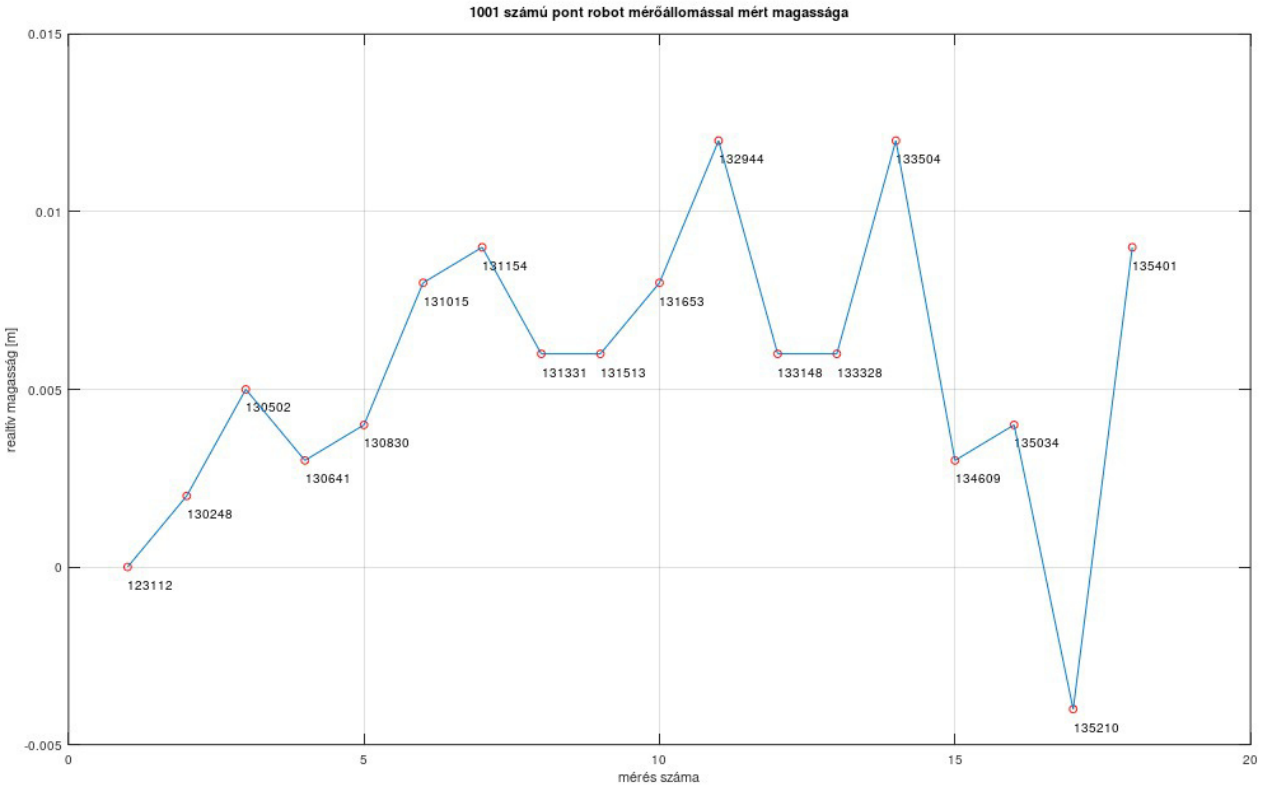
## Kamera

A kamerából kijövő adatok videók voltak. Ezek elemzésekor két problémába ütköztem. Ahhoz, hogy az elmozdulás értékhez időbélyeget is tudjunk illeszteni, a videó rögzítő algoritmus a fájl nevének a felvétel kezdő időpontját adja meg. Ezzel a probléma az volt, hogy a Raspberry Pi-k órája bekapcsolásnál a kikapcsolás időpontját jegyzi meg és onnan folytatja az időszámlálást. Az óraszinkronizálást vagy kézi beállítással vagy az internetre való csatlakoztatással lehet megoldani. Ott helyben ezt a problémát észleltem és a videók rögzítésének időpontját kézzel felírtam.

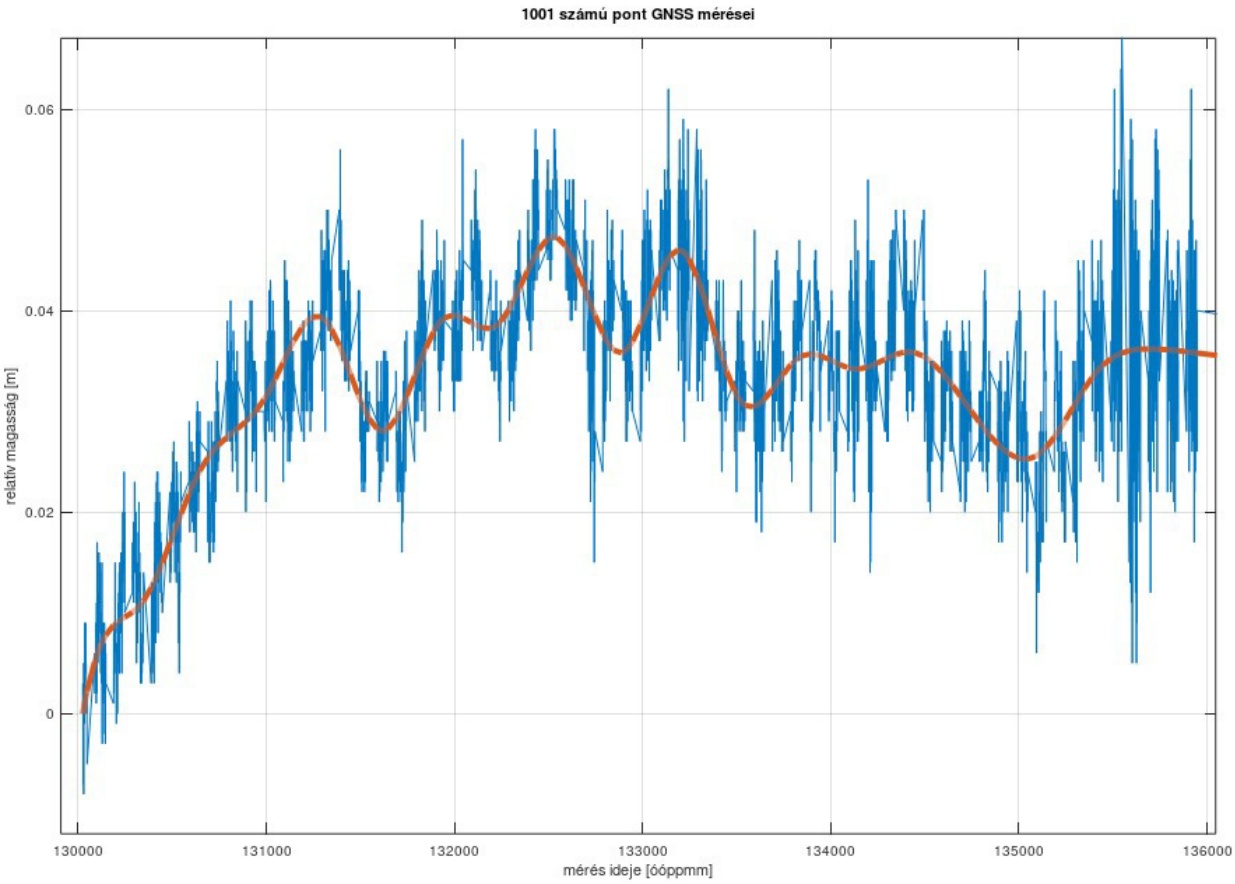
A másik nagyobb probléma, hogy 25 képkocka/másodperc rögzítéséhez a hadrver nem volt elég gyors, a kép ugrált, tehát két egymást követő kép között nem mindig 1/25 másodperc telt el, így az időszinkronizáció sajnos megoldhatatlan volt. Ezt a problémát utólag realizáltam és ezt sajnos nem volt lehetséges javítani, így a továbbiakban nem értékeltem ki a kamera által rögzített videofájlokat ezen mérés alkalmával. Ezen technológia további konfigurálásra szorul, hogy az elkövetkező teszt során alkalmas legyen adatok kinyerésére. A másodpercenként rögzítendő kép számot a PI és az SD kártya sebességéhez kell igazítani.

## GNSS és robot mérőállomás összehasonlítása

Ellenőrzésképpen az 1001-es pontra történő méréseseket feldolgozva a két technológiából hasonló mozgásoknak kellene előzetesen kapnunk. Ehhez a mérési eredményekre egy-egy spline görbét illesztettem, hogy szemléletes legyen a hasonlóság.




*49. Ábra*




*50. Ábra*

A 49. ábrán látható a robot mérőállomással megmért magasságok a hozzátartozó időponttal feliratozva. A következő ábrával összevetve a 49.en csak a második méréstől szükséges az adatok hasonlóságát megnéznünk. Az első szembetűnő különbség, hogy az etalon méréshez viszonyítva a GNSS-ből kinyert adatok akár +6 centiméterre (13:55 környéke) is emelkedtek, de elmondható, hogy itt rövid időn belül nagy volt a magasság szórása. Míg a mérőállomással mért értékeknél a két szélsőérték közötti különbség nem haladta meg a 2 centimétert.

Ha a görbe alakját nézzük meg szemléletesebben kitűnik, hogy nagyon hasonló lefolyása van a kettőnek. 13:11-ig mindkét görbén emelkedés látható, csak a 13:06-kor történt mérésnél volt ~2 milliméteres csökkenés, de ez betudható mérési hibának vagy a hullámzás következtében keletkező kis mozgásnak.

Ezt követően 13:13-kor volt egy enyhe süllyedés ami a GNSS mérések alapján is látható. Majd a mérőállomás mérései alapján újabb növekedés következett, ez már viszont csak eltolva jelentkezik a GNSS méréseiben. Az ezt követő időszakban 14:00-ig a két függvény teljesen hasonló lefutása, de eltolva jelentkeztek a mozgások. Elmondható, hogy a mérőállomás alapján az egymást követő mérések rövid időintervallumot követve valósultak meg és alacsonyak lettek a különbségek az előző-következő között.

## 5.4 Összefoglalás, fejlesztési javaslatok

A GNSS NMEA üzenetekbe való rögzítése RTK üzemmódban megbízhatóan működött, a technológia automatizált monitoring rendszerbeli felhasználására alkalmas. Egy hosszabb mérés során viszont szükséges figyelni az akkumulátor állapotát a vevőben vagy egyszerűen állandó áramellátást biztosítani számára.

A robot mérőállomásról is elmondható, hogy hasonlóan a GNSS-hez teljesen használható volt. Teljesen automatizálttá ütemezett feladat végrehajtásával lehetne, ezt én ezen mérések során nem tettem meg, kézzel indítottam azokat.

A kamera további fejlesztéseket igényel. A Raspberry Pi-k megfelelő időbeállítását vagy minden mérés előtt kézzel szükséges megtenni vagy valamilyen módon hozzáférést kell biztosítani az internethez, így az automatikusan lekérdezi azt. Az ugró képkockák kiküszöbölésére az fps szám csökkentése javasolt 1 vagy 2 képkocka/másodpercre. Mivel a probléma valószínűleg a hardver gyorsaságával (fájl írási sebesség) kapcsolatos, így ezen érték csökkentésével megoldható.

*51. Ábra Monitoring rendszer és a ponton*

# 6. Úszómű mérés

Az előzetes teszteléseket követően élesben, Dunaújvárosban alkalmaztam a rendszert. A Budapesti Műszaki és Gazdaságtudományi Egyetem több tanszékével, többek között a Hidak és Szerkezetek, az Általános és Felsőgeodézia és a Vízépítési és Vízgazdálkodási Tanszék részvételével zajlott a projekt végrehajtása.

Munkánk célja a szerkezet próbaterheléséből keletkező elmozdulások mérése. A mérési eredmények és tapasztalatok a későbbiekben egy hasonló kialakítással rendelkező, de nagyobb méretű szerkezet megvalósításában lesznek hasznosak.

*52. Ábra: Épülő úszómű 2019 októberében*
*Forrás: Takács Bence*

## 6.1 Mérések előkészítése

A kész úszóműre egy acélszerkezetű felépítmény került, amely egy állandó teherként van jelen azon. A különböző teherállások végrehajtásához IBC tartályokat helyeztek a szerkezetre. Ezeket

vízzel töltötték meg ezzel létrehozva a megfelelő nagyságú terhelést. Az én feladatom az ebből keletkező elmozdulások mérése volt különböző technológiák segítségével.

*53. Ábra: Felépítmény és IBC tartályok*
*Forrás: Takács Bence*

A munka több napot vett igénybe, ugyanis ezeket a tartályokat több módon helyezték el. Egy mérési napon egy elrendezést alkalmaztak, de más és más mennyiségű vizet engedtek az IBC tartályokba ezzel szimulálva a különböző teherállásokat. A megbillent szerkezet imitálására a két ellentétes oldalt eltérő mennyiségű feltöltést is alkalmaztak.

A mozgásvizsgálati mérésekhez az előzetes tesztelés során alkalmazott, robot mérőállomást, GNSS technológiát és a kamerát alkalmaztam. Ezen felül a deformáció észleléséhez szintezéssel egészítettük ki a rendszerünket. Ezt nem automatizáltan, hanem hagyományos módon hajtottuk végre.


*54. Ábra: Mozgásvizsgálati pontok*
*Forrás: Takács Bence*

Ahhoz, hogy megfelelően tudjuk a szerkezet mozgásait modellezni mértékadó helyekre szükséges mozgásvizsgálati pontokat helyeznünk. Ezeket a 4 sarokpontra és az oldalak felezőpontjára tettük ki (54. ábra). Mind a 8 ponton megtalálható volt egy állandósított mozgásvizsgálati prizma, amit a robot mérőállomás méréseihez használtam fel. Ezt 2 műszer mérte automatizáltan, az 1-2-3-4-es számú pontokat a Leica 1200-as műszere, míg az 5-6-7-8-at a Leica 1800-as műszere. Fix pontoknak pedig, 4 darab prizmát helyeztünk el a parton, amiből 3-3-at használt fel a műszerek szabad álláspont létesítésére.

A 3-as, 5-ös és 7-es pontokon egy-egy GNSS vevő helyezkedett el, amely a nyers mérési adatok rögzítette. Bázisnak ehhez a méréshez egy kikötőbakon helyeztünk el egy Leica GNSS vevőt, így 4 GNSS vevő dolgozott szimultán.

*55. Ábra: Leica 1200 és a GNSS bázis*

A 3-4-5 pontokon egy-egy ArUco jel lett felragasztva, ezekhez külön kamerák lettek kihelyezve amelyek azokat detektálták. A kamerák egy-egy Leica 305 mérőállomás optikájára voltak rögzítve. A Raspberry Pi-k megfelelő áramellátását powerbankekkel oldottam meg..

*56. Ábra: Kamera konfigurálása*

Elsőként a robot mérőállomásokat konfiguráltam. Ehhez a *coomaker.py* alkalmazással megmértem manuálisan az összes prizmát amelyből előállt a koordinátalista. Két külön számítógéppel vezérelve, azokon külön JSON fájlokkal konfigurálva futott a *robotplus.py* algoritmus, ez egy SQLite adatbázisba rögzítette az összes észlelési adatot.

A GNSS-ek beállítása nem volt különösebben nagy feladat. A 3-as ponton lévő Stonex vevő elindítást követően automatikusan rögzítette a nyers adatokat a saját memóriájába. Az 5-ös és 7-es ponton lévő Topcon Hiper II-es vevők a hozzátartozó vezérlővel konfigurálhatóak és ezek is a saját memóriájukba mentették az észleléseket. A Leica vevőjénél minden mérési napon meghatároztam annak koordinátáját mivel, ezt nem kényszerközpontosan helyeztük el, hanem a kikötőbakon különböző helyeken. Ezt hálózati RTK üzemmódban 120 epocha megmérésével tettem meg. Ez mind EOV, mind ETRS89 koordinátákat rögzített, ezek lettek a bázis koordinátái. Ezt követően automata rögzítést indítottam el, így a RINEX fájlokat is rögzítette.

Ezt követően a kamerák konfigurálása következett. A 3 kamera első alkalommal történt beállítása során a fókuszt is be kellett állítanom. Ehhez egy routert vittünk magunkkal, amire ethernet porton keresztül dugtam rá a Raspberry Pi-ket. Mivel a műszerek a további mérési napokon is közel hasonló távolságra voltak felállítva az ArUco kódoktól, így ezt elég volt az első alkalommal megtennünk.

*57. Ábra: ArUco jel és GNSS*
*Forrás: Takács Bence*

Az idő beállítását először manuálisan a konzol ablakból a *sudo date +%T s '11:14:00'* paranccsal tudtam megtenni. A probléma, hogy ez az algoritmus lassan futott le (kb. 20 másodperc), így a másodperc pontos időbeállítást nem volt könnyű ezzel létrehozni. A későbbiekben a telefonom mobil hozzáférési pontját hozzáadtam a Raspberry Pik konfigurációjához, így azok automatikusan minden mérési nap elején beállították maguknak. Ha azokat konstans áramforrással láttuk el, az idő beállítás probléma megoldódott.

Az ugró képkockák hibáját azzal küszöböltem ki, hogy a kamera nem 25 képkocka/másodperc értékkel rögzített, hanem annál jóval kevesebbel (2 képkocka/másodperc). Ezáltal nem történt kimaradás a videófelvételben és emellett tárolóhelyet is spóroltam a számítógépen.

*58. Ábra: Felállított rendszer*

## 6.2 Mérések végrehajtása

Minden teherállás mérése hasonló forgatókönyv szerint zajlott. Először a Raspberry Pi-k elindításával kezdtem, ezek újra 1000 x 1000 felbontásban, de 2 képkocka/másodperc gyakorisággal vettek fel. Volt olyan alkalom, amikor a teljes napot rögzítették.
A GNSS hasonlóan az előző technológiához volt, hogy egész napokat mért egyfolytában, de akkumulátor kímélés miatt sokszor megállítottuk a teherállások között átállás közben.
Ezt követően a robot mérőállomások *robotplus.py* alkalmazását futtattam, ez automatizáltan körülbelül 2 perc alatt végrehajtotta az összes prizma megmérését.
Majd a szintezés következett. A 3-as számú pontot jelöltük ki kezdőpontnak, így az összes magasságot ehhez képest határoztuk meg. A pontok mérését a Leica DNA03-as műszer közbenső pont funkciójával hajtottam végre, így a 3-as pont volt a hátra és előre mérés is. Egy magasság meghatározás során 5 leolvasás történt és ezeknek a medián értékét vette. Ezt az 59. ábrán látható módon teszteltük, mint látható a képről is 88 mérés terjedelme 2 milliméter volt. Ennek fontossága abban rejlik, hogy az úszómű nem stabil, sok esetben a hullámzás nehezítette a méréseinket, de ennek terjedelme sem haladta meg a 3 milliméteres határértéket. A medián alkalmazásával a kiugró értékeket ki tudtuk szűrni. Két műszerállásból határoztuk meg a pontok magasságát és egy pont (a


*59. Ábra Szintezés tesztelése*

8-as) mindkét műszerállásban szerepelt. Átlagosan elmondható, hogy szintezés záróhibája 0,2 – 0,4 milliméter körül alakult, de nagyobb hullámzás esetén is maximum 2 milliméter volt ez az érték.

Ezt követően egy kontrollmérést indítottam a robot mérőállomásokkal, hogy azok átlagát vegyem vagy a durva hibák (ugyanazon prizma kétszeri mérése, prizma kihagyása) kiszűrhetőek legyenek. A *robotplus.py* program a prizmák megtalálására az irányokat használja fel. Ha a koordináta listában lévő magassághoz képest a pontnak nagyban változik a magassága, akkor ez a műszertől távolabb lévő pontok esetén okozhat két fajta hibát. Vagy teljesen kihagyja az adott mérésből (a beállított tűrést meghaladja) vagy ha a keresés közben a látómezejében egy másik prizma feltűnik azt méri meg helyette.

## 6.3 Mérések kiértékelése

Mivel a komplett projekt 7 mérési napot vett igénybe, így rengeteg adat keletkezett. Feladatom viszont a technológiák használhatósága, összehasonlítása, így csak egy mérési nap eredményeit szemléltetném a következőkben. Ez a december 14-én történt 1 darab teherállást jelenti. Az etalon mérés 11 óra 20 perc és 11 óra 49 között zajlott le, míg a terhelt szerkezet észlelése 13 óra 43 perc és 14 óra 07 perc között volt.

### *6.3.1 Robot mérőállomás*

A robot mérőállomás által mért értékek koordináta rendszerét még az első nap folyamán határoztam meg. Ez a Leica GNSS bázis közelében lévő 1200-as műszer álláspontjának koordinátái a következők voltak: É = 100,0000 m, K = 100,0000 m, M = 100,0000 m. Az úszómű magasságai ez alapján 93-96 méter között helyezkedtek el.

| Mérőállomás | 11:20 – 11:49 |
|---|---|
| 1 | 94,2098 |
| 2 | 93,9093 |
| 3 | 94,0893 |
| 4 | 93,8803 |
| 5 | 93,8620 |
| 6 | 93,9293 |
| 7 | 93,9805 |
| 8 | 95,3639 |

| | 13:43 – 14:07 |
|---|---|
| 1 | 94,1023 |
| 2 | 93,9247 |
| 3 | 94,2319 |
| 4 | 93,9282 |
| 5 | 93,8014 |
| 6 | 93,7470 |
| 7 | 93,6664 |
| 8 | 95,1412 |

| Különbségek | |
|---|---|
| 1 | -0,1076 |
| 2 | 0,0154 |
| 3 | 0,1426 |
| 4 | 0,0480 |
| 5 | -0,0605 |
| 6 | -0,1823 |
| 7 | -0,3141 |
| 8 | -0,2228 |

*60. Ábra: Mérési eredmények [m]*

A 60. ábrán láthatóak a két epocha mérési eredményei és a különbségek méter dimenzióban. Ezek a már átlagolt (szintezés előtt és után mért) eredmények. Az eredményeket egy SQL adatbázisban tároltam, majd táblázatkezelőbe másoltam át. A különbségekből kivehető, hogy a 2-3-4 számú pontoknál volt emelkedés a többi ponton pedig süllyedés.

### *6.3.2 Szintezés*

A szintezést az előző fejezetben tárgyaltak alapján hajtottuk végre. A „null” mérés eredményei a következők lettek (az értékek méterben értendőek):

```
Line  : LINE00026        Method: HE
================================================================================
Date  : 14.12.2019       Time  : 11:33:45
Staff1: --------         Staff2: --------
================================================================================
|    PointID     |Backsight | Intermdt.|Foresight |          | Distance |  Remarks  |
|(Station Result)|          |          |          |(delta H) | (D Bal ) | (Pt Hgt ) |
--------------------------------------------------------------------------------
|Start PtID      |          |          |          |          |          |           |
|3               |          |          |          |          |          |(   0.0000)|
|                |          |          |          |          |          |           |
|3               |   1.44464|          |          |          |     18.56|   --------|
|              4 |          |   1.47582|          |   -0.0312|      7.79|    -0.0312|
|              5 |          |   1.47906|          |   -0.0344|      3.91|    -0.0344|
|              6 |          |   1.43202|          |   +0.0126|     10.73|     0.0126|
|              7 |          |   1.36313|          |   +0.0815|     21.58|     0.0815|
|              8 |          |   1.35684|          |   +0.0878|     23.62|     0.0878|
|3               |          |          |   1.44547|          |     18.56|   --------|
|                |          |          |          |          |          |          1|
|(  1)           |          |          |          |(-0.00083)|(   -0.00)|(  -0.0008)|
|                |          |          |          |          |          |          1|
|3               |   1.50815|          |          |          |     15.24|   --------|
|              2 |          |   1.45716|          |   +0.0510|      4.03|     0.0502|
|              1 |          |   1.38672|          |   +0.1214|      8.06|     0.1206|
|              8 |          |   1.41851|          |   +0.0896|     11.57|     0.0888|
|3               |          |          |   1.50942|          |     15.24|   --------|
|                |          |          |          |          |          |          1|
|(  2)           |          |          |          |(-0.00126)|(   +0.00)|(  -0.0021)|
|                |          |          |          |          |          |          1|
```

Mint látható, az első műszerállásban a 4-5-6-7-8 számú pontokat mértük. A másodikban a 2-1-8 pontokat. Az előbbi esetben a záróhiba -0,83 milliméter lett, az utóbbiban ez emelkedett -1,26 milliméterre (5. oszlop zárójeles értékek). Ellenőrzésképpen a 8-as pontot mindkét állásban megmértük. Az elsőben +0,0878 méterre, míg a másodikban +0,0888 méterre kaptam meg a magasságot (ezt az utolsó oszlopban láthatjuk), ami 1,0 milliméter különbséget jelent.

```
Line  : LINE00027        Method: HE
================================================================================
Date  : 14.12.2019       Time  : 13:51:34
Staff1: --------         Staff2: --------
================================================================================
|    PointID     |Backsight | Intermdt.|Foresight |          | Distance |  Remarks  |
|(Station Result)|          |          |          |(delta H) | (D Bal ) | (Pt Hgt ) |
--------------------------------------------------------------------------------
|Start PtID      |          |          |          |          |          |           |
|3               |          |          |          |          |          |(   0.0000)|
|                |          |          |          |          |          |           |
|3               |  1.25257 |          |          |          |    18.48 |   --------|
|              4 |          |  1.37450 |          |  -0.1219 |     7.67 |   -0.1219 |
|              5 |          |  1.48411 |          |  -0.2315 |     3.97 |   -0.2315 |
|              6 |          |  1.55538 |          |  -0.3028 |    11.05 |   -0.3028 |
|              7 |          |  1.61504 |          |  -0.3625 |    21.91 |   -0.3625 |
|              8 |          |  1.50767 |          |  -0.2551 |    23.91 |   -0.2551 |
|3               |          |          |  1.25293 |          |    18.48 |   --------|
|                |          |          |          |          |          |         1 |
|(  1)           |          |          |          |(-0.00036)|(   +0.00)|(  -0.0004)|
|                |          |          |          |          |          |         1 |
|3               |  1.34997 |          |          |          |    15.20 |   --------|
|              2 |          |  1.42304 |          |  -0.0731 |     3.94 |   -0.0734 |
|              1 |          |  1.48573 |          |  -0.1358 |     8.01 |   -0.1361 |
|              8 |          |  1.60569 |          |  -0.2557 |    11.70 |   -0.2561 |
|3               |          |          |  1.34980 |          |    15.19 |   --------|
|                |          |          |          |          |          |         1 |
|(  2)           |          |          |          |(+0.00017)|(   +0.00)|(  -0.0002)|
|                |          |          |          |          |          |         1 |
```

A megterhelt szerkezet esetén fentebb láthatóak a szintezés eredményei. A záróhibák képe itt még jobban alakult. Az első állásból 0,36 millimétert, a másodikból pedig +0,17 millimétert kaptam. A 8-as pont magasság -0,2551 méter és -0,2561 méter értékeket vettek fel, amiből az előzőhöz hasonlóan 1,0 milliméter volt a különbség.

A szintezést és a robot mérőállomás méréseit is összehasonlítottam táblázatkezelő szoftverben. A különbségek a 61. ábrán láthatók. A szintezésből kapott eredményeket a 2. oszlopban tüntettem fel. Mivel az egész szerkezet is mozgást végzett, ezért, hogy értékelhető eredményeket kapjak a mérőállomás által mért eredményeket a 3-as pont különbségével el kellett tolnom. Ebben az esetben a 2. és 4. oszlop értékeit összehasonlítva

| Különbségek [m] | | | |
|---|---|---|---|
| | Szintezés | Mérőállomás | Eltolt mérőállomás magasságok |
| 1 | 0,2567 | -0,1076 | 0,2502 |
| 2 | 0,1236 | 0,0154 | 0,1272 |
| 3 | 0,0000 | 0,1426 | 0,0000 |
| 4 | 0,0907 | 0,0480 | 0,0947 |
| 5 | 0,1971 | -0,0606 | 0,2032 |
| 6 | 0,3154 | -0,1823 | 0,3249 |
| 7 | 0,4440 | -0,3141 | 0,4567 |
| 8 | 0,3439 | -0,2227 | 0,3654 |

*61. Ábra*

megkaptam a szintezés és a robot mérőállomás által mért értékek különbségét, amely a 62. ábrán látható.

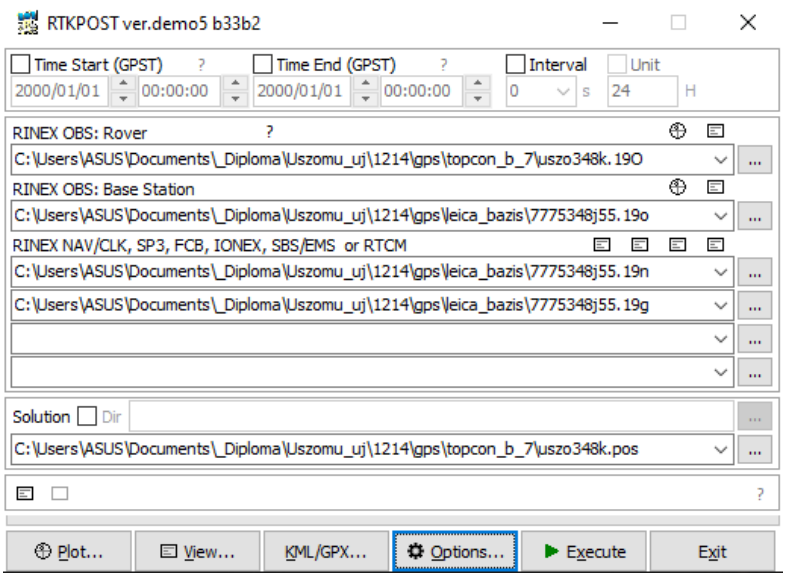
| | Maradék ellentmondás a szintezés és mérőállomás között A 3-as ponthoz viszonyítva [m] |
|---|---|
| 1 | -0,0065 |
| 2 | 0,0036 |
| 3 | 0,0000 |
| 4 | 0,0040 |
| 5 | 0,0061 |
| 6 | 0,0095 |
| 7 | 0,0127 |
| 8 | 0,0215 |

*62. Ábra*

Az értékekből jól látható, hogy az eltérések 1 centiméter alatt voltak 1-től egészen a 6-os pontig. A legnagyobb különbség a 8-as számú ponton volt, ahol 2,15 centiméter ez az érték. A legnagyobb eltérés valószínűleg a szerkezet mozgásából következhetett, ha a Dunán egy hajó jött el gyakran az általuk keltett hullámzás nagy mozgást idézett elő a szerkezeten.

### *6.3.3 GNSS*

A GNSS vevőkből kapott adatok nem teljesen nyers RINEX fájlok voltak. Ez alól kivételt képez a Leica bázisállomásként használt műszere, itt 3.02-es verziójú Receiver Independent Exchange Format (RINEX) adatokat olvastam ki.




*63. Ábra: RTKPOST*

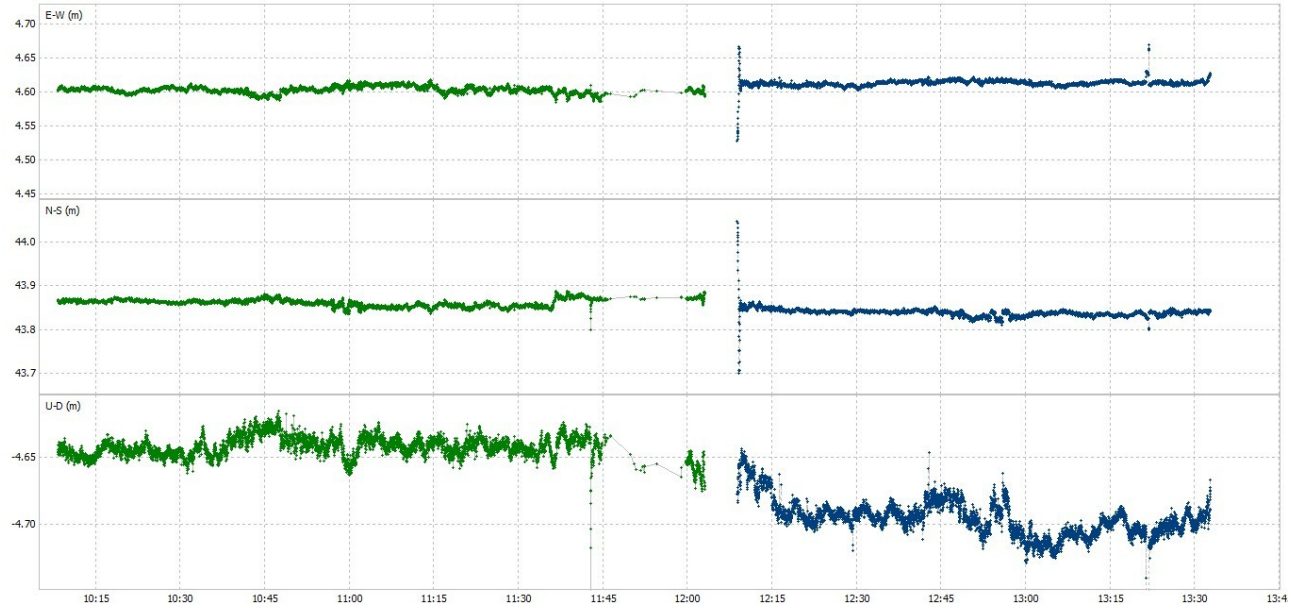
A 3-as ponton lévő Stonex vevő *.STH* kiterjesztésű bináris fájlt szolgáltatott. Ezt egy *ToRinex4* nevű szoftverrel tudtam az általam kívánt formátumúvá alakítani.

Az 5-ös és 7-es ponton lévő Topcon vevők pedig *.TPS* kiterjesztésű fájlba mentették az adataikat. Ezt a Topcon Tools Demo verziójával tudtam RINEX formátumú konvertálni.

Ezt követően az adatokat a nyílt forráskódú RTKLIB Explorer RTKPOST moduljával dolgoztam fel. Ide beolvastam a bázisállomás által mért adatokat, a navigációs fájlokat és a mozgásvizsgálati pontokon a vevők által mért észleléseket. Ezeket különböző beállítások mellett futtattam, hogy a legpontosabb és legtöbb fix pozícióval rendelkező adatsor előálljon.

A 79. oldalon látható a Stonex vevő által a 3-as ponton rögzített adatok grafikonja. A felső a kelet – nyugat, a középső az észak – dél, míg az alsó a függőleges irányú elmozdulásokat hivatott ábrázolni. A vízszintes tengelyen az idő látható GPST-ben, a függőleges tengely pedig az elmozdulás méter egységben. 12:05 és 12:10 között (helyi időben 13:05 és 13:10 között) található egy szakadás az ábrán, ez abból következik, hogy a Leica bázis vevőjében lévő akkumulátor lemerült, így semmilyen pozíció nem nyerhető ki egyik vevőből sem.

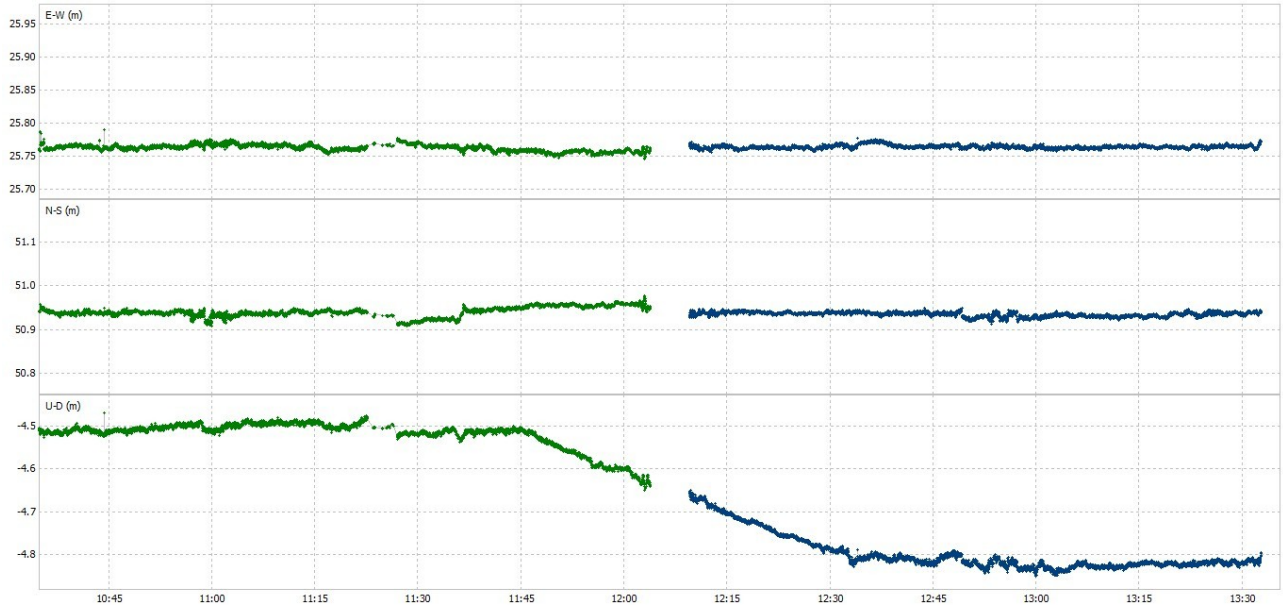
Összehasonlítva a robot mérőállomásokból jövő adatokkal az ott kapott elmozdulás érték erre a pontra 0,1426 méter lett. A feldolgozott adatokat megnézve a kezdeti állapotnál az átlagmagasság 133,875 méter volt, míg az elmozdulást követően ez az érték 134,016 méterre nőtt, így a kettőnek a különbsége 0,141 méter, amely 2,6 milliméteren belül megközelíti a mérőállomás által mért értéket.




*64. Ábra 5-ös pont elmozdulásai (80. oldalon nagyobb méretben is megtekinthető)*

Az 5-ös ponton (64. ábra) nem voltak ekkora elmozdulások mint a másik két esetben, így a GNSS technológia magassági meghatározásának hibaforrásai miatt ezek zajosabbak lettek. A robot mérőállomás mérései alapján a várható elmozdulás -0,0606 méter volt, a 11:45ig (GPSTime) történt mérések magasságának átlaga 133,954 méter volt, míg a 12:30-at követően észlelt eredményeknek 133,898 méter. Ezeknek különbsége -0,056 méter, ami 4,3 milliméterrel tér el a mérőállomáshoz képest.

A 7-es ponton hasonló helyzet állt elő az előző kettőhöz. Itt a 65. ábrán látható értékek is tükrözik a várható, -0,3141 méteres függőleges elmozdulást.




*65. Ábra 7-es pont elmozdulásai (81. oldalon nagyobb méretben is megtekinthető)*

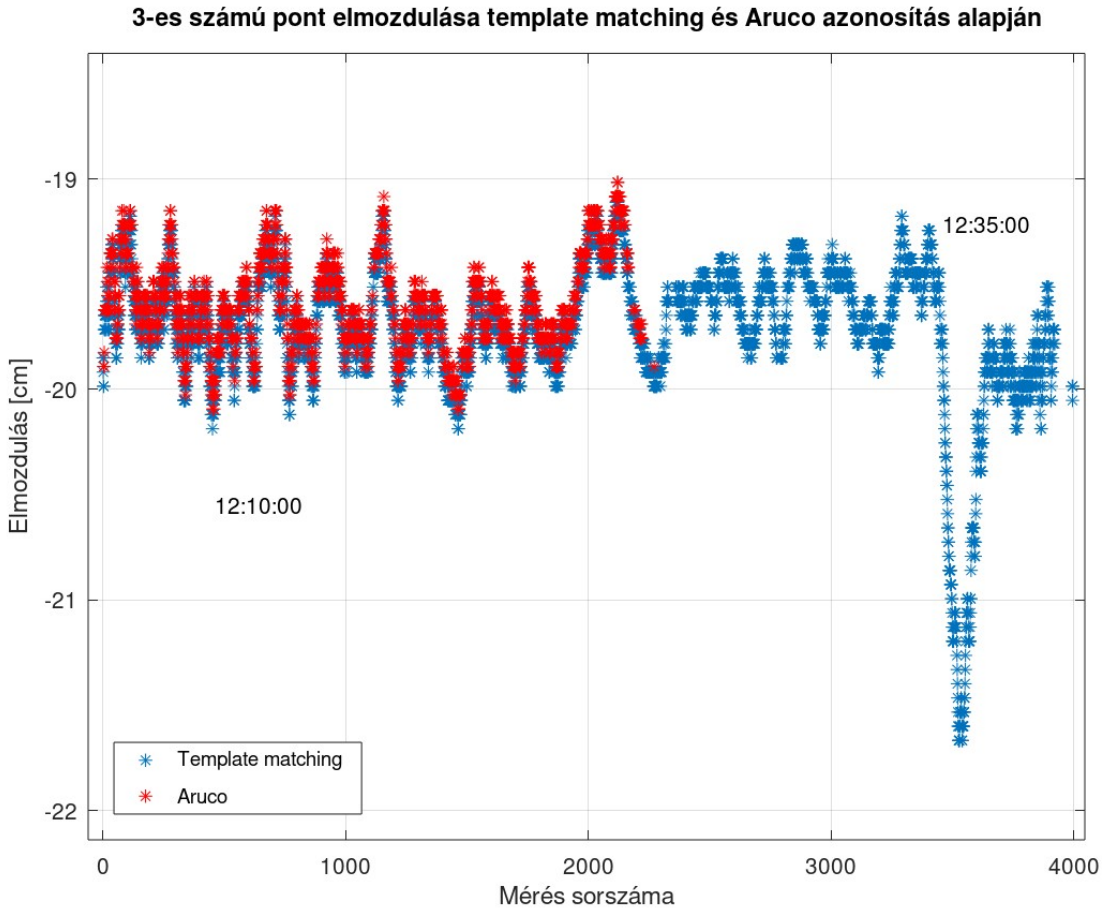
### *6.3.4 Kamera*

A kamerából kapott videókra két már megírt algoritmust futtattam le. Az egyik a már korábban említett ArUco jel kereső szkript, a másik pedig egy „template matching” alapú eljárás és mindkettőről elmondható, hogy OpenCV alapú.

A „template matching” alapú algoritmusnak meg kell adnunk egy mintát, amelyet minden videóban szereplő képkockán megkeres. Ezzel eredményt minden alkalommal kapunk, mivel mindig lesz olyan képkocka részlet amely a legjobban korrelál, míg az ArUco jel keresés nem feltétlen vezet mindig találatra. Az algoritmus futtatásához szükség volt egy mintára amelyet kereshetett, ezt az első képkockából tudtam kiszedni egy python szkript segítségével (*video_fist.py*). A képből csak a keresendő részt meghagyva lefuttattam a *video_correlation.py* szkriptet amelyből előálltak a jel közepének koordinátái, egy időpont (UTC időben) és egy korrelációs együttható, amelyből az illeszkedés megfelelőségére lehet következtetni.

Ezt követően a videó fájlokon a *video_aruco.py* szkriptet is futtattam, amely a az ArUco kódokat kereste a képeken. Ebből időbélyeggel, az ArUco kód sorszámával és a két koordináta komponenssel ellátott fájl készült.

A fájlokat Octave-ba olvastam be és itt különböző számításokat hajtottam rajtuk végre, majd ábrázoltam őket.




*66. Ábra 3-as számú pont elmozdulása*

A 3-as számú pont elmozdulásai a 66. ábrán láthatóak. Az elmozdulásokat először átszámítottam metrikus rendszerbe. Mivel a jelek mérete 72 mm volt, így a videókon lévő pixelszámból (kb. 107 pixel) és a valós méretből felállítva egy arányt megtörténhetett az átszámítás.

A kamera csak az elmozdulások kezdetéig rögzítette az adatokat, így ebből sajnos a várható emelkedésre következtetni nem lehet. Viszont ha az ArUco jel keresés és template matching-et hasonlítjuk össze láthatjuk, hogy erős korreláció van a kettő között, de az adatok száma sokkal kevesebb az előbbi esetben. Az ArUco azonosítás egyértelműen megbízhatóbb, nincs egyetlen kiugró adat sem. A két adatsor lineáris kapcsolatát jellemző mérőszámot is kiszámoltam, ez a korrelációs együttható. Ennek meghatározásához a következő képletet alkalmaztam:

$$r_{x,y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{(n-1) \cdot s_x \cdot s_y} \quad ,$$

ahol:

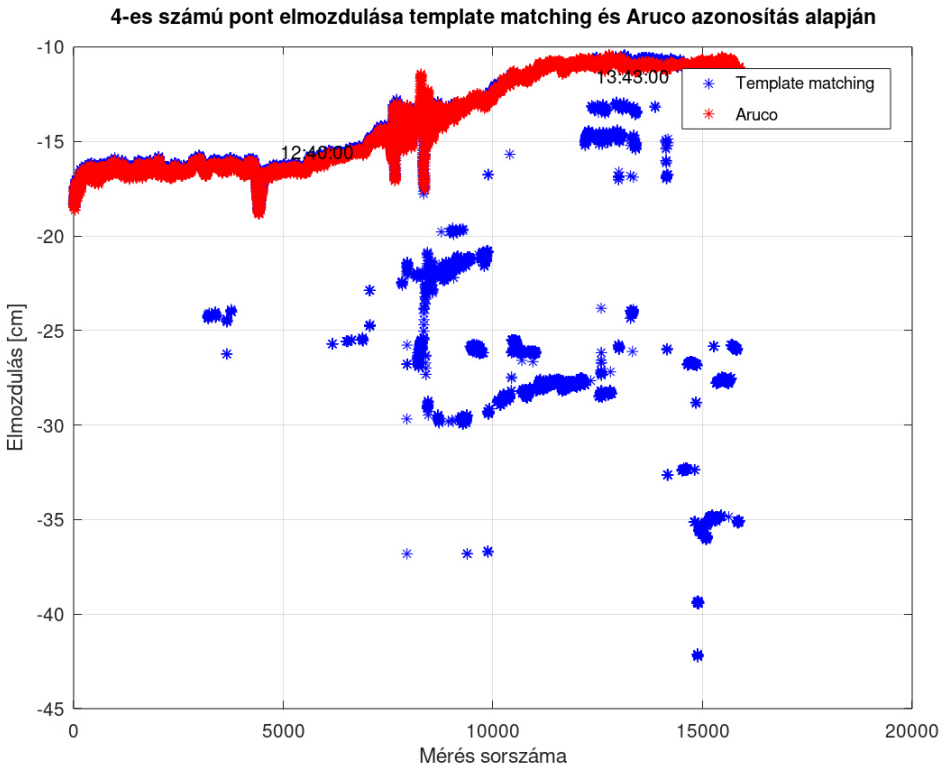
- $r_{x,y}$ = korrelációs együttható az x és y adatsor között,
- n = adatok száma,
- i = az éppen aktuális adat sorszáma,
- $x_i$ = az egyik adatsor éppen aktuális eleme,
- $\bar{x}$ = várható értéke az x adatsornak,
- $y_i$ = a másik adatsor éppen aktuális eleme,
- $\bar{y}$ = várható értéke az y adatsornak,
- $s_x$ és $s_y$ a tapasztalati korrigált szórást jelölik.

A korreláció mindig -1 és 1 közötti értéket vehet fel. Minél közelebb van két adatsor korrelációs együtthatója 1-hez vagy -1-hez, annál szorosabb a köztük lévő lineáris függvénykapcsolat. Ha ez az érték 0 akkor nem beszélhetünk korreláltságról.

A template matching és az ArUco felismerés alapján meghatározott pontok adatsora között a korrelációs együttható értéke:

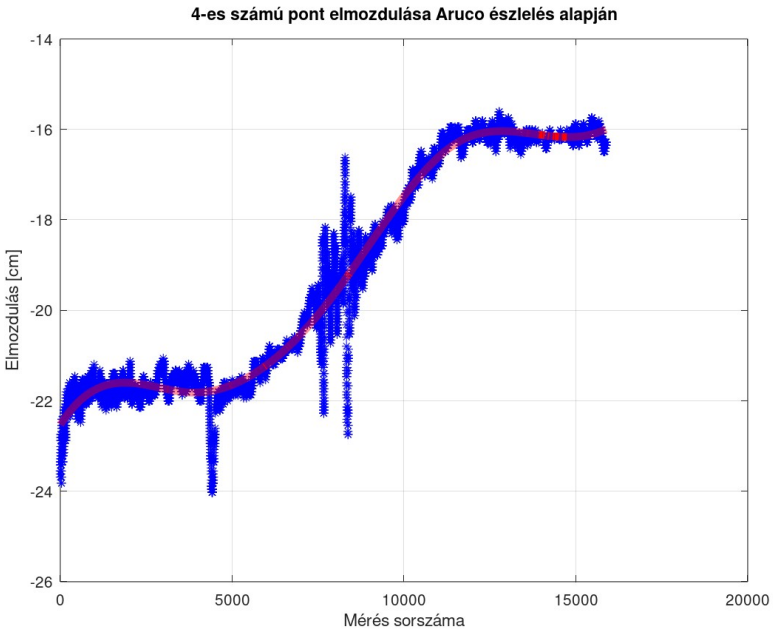
$$r^3_{tA} = 0{,}9898,$$

így elmondható, hogy az ábra és statisztikai adatok alapján is magas a lineáris kapcsolat a két adatsor között.




*67. Ábra 4-es számú pont elmozdulása*

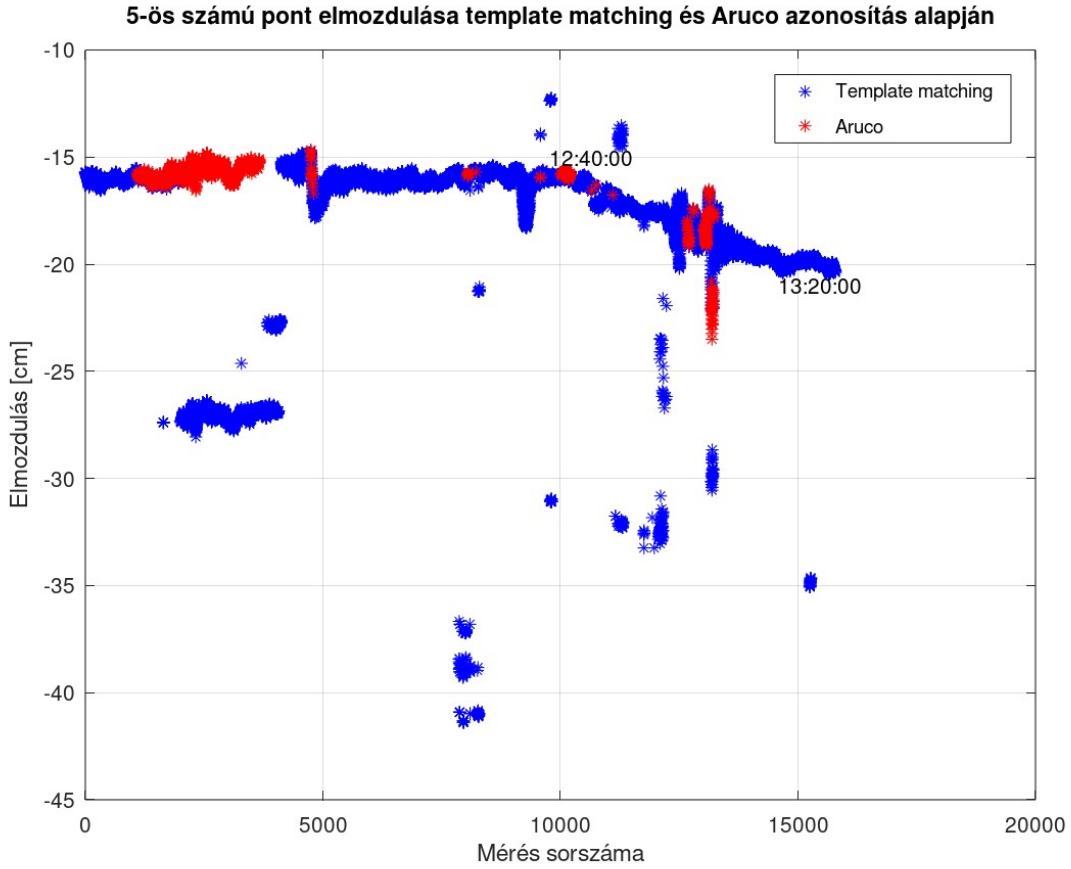
A 4-es pont két algoritmussal kiszámolt elmozdulás értékeit a 67. ábra szemlélteti. Elmondható, hogy a template matching alapján meghatározott jel koordináták jóval zajosabbak lettek. Ezzel szemben ebben az esetben a mérés teljes ideje alatt rendelkezésre állnak adatok az ArUco felismerésből is. A két algoritmus által meghatározott adatsor között a korrelációs együttható: $r^4_{tA} = -0{,}4857$. Ez az érték már jóval alacsonyabb mint az előző. Ez abból adódhat, hogy a template matching által meghatározott pontok sokkal zajosabbak voltak.




*68. Ábra 4-es számú pont elmozdulására illesztett spline görbe*

Az ArUco pontokra, egy spline görbét illesztettem, ezen jól látható az elmozdulás mértéke. Ezt a 68. ábra szemlélteti. Ezen az ábrán az értékek nincsenek hozzáigazítva a template matchingel kiszámolt értékekhez.

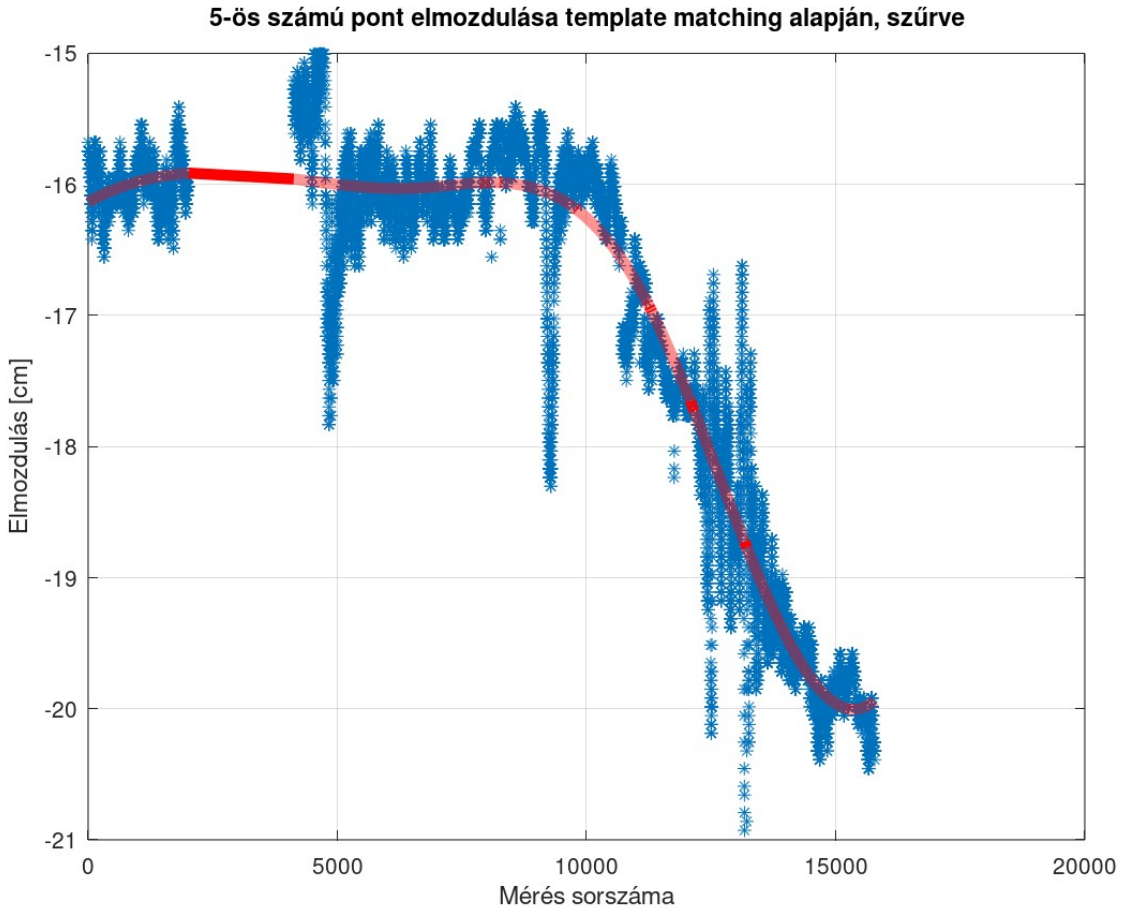
Ahhoz, hogy számszerű értékeket kapjak a még elmozdulás előtti pontokat és a már beállt emelkedett értékeket átlagoltam. A null mérés -16,45 centiméterre, míg a már elmozdult szerkezet -10,76 centiméterre jött ki. Ennek a kettőnek a különbsége 5,69 cm. A mérőállomással meghatározott érték 4,80 cm volt, így a kettő között 0,89 cm-re adódott a különbség.




*69. Ábra 5-ös számú pont elmozdulása*

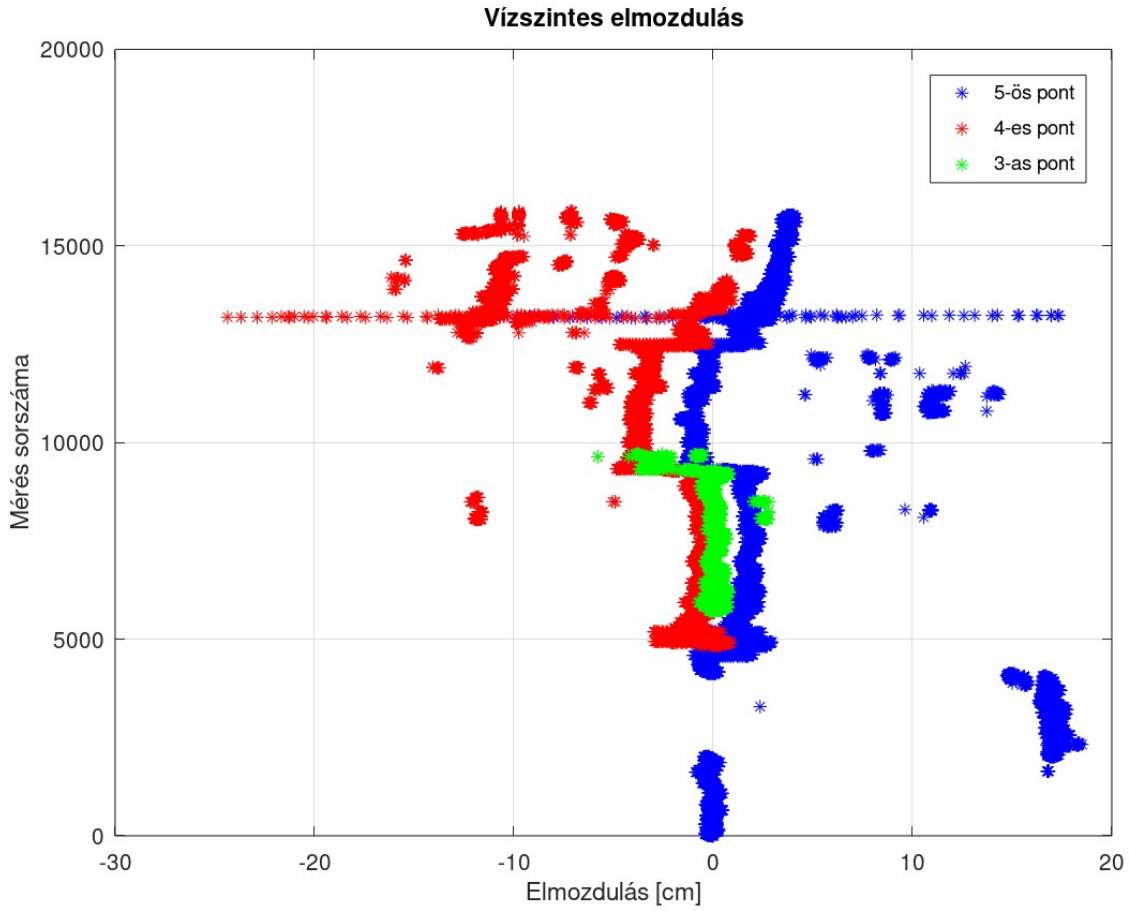
Az 5-ös számú pont két algoritmussal meghatározott pontjai a 69. ábrán láthatóak. Újra előjött az a probléma, hogy nem a teljes mozgásvizsgálat alatt rögzítette a kamera a felvételeket, így csak 13:30-ig találhatóak adatok. Ez elméletben pont az az időpont amelyre a mozgás lejátszódott, de az átlagolásnál a későbbiekben gondot okozhat. Itt elmondható, hogy az előző kettővel ellentétben már alig volt képes ArUco felismerésre az algoritmus, viszont hasonló tendencia szerint sokkal megbízhatóbb eredményeket tudott szolgáltatni. A kettő között kiszámoltam a korrelációs együtthatót ami $r^5_{tA}$ = -0,0953 lett. Ez az alacsony érték abból adódhat, hogy a közös időpontokban

ahol mindkét algoritmus felismerte a jelet, a template matching által meghatározott értékek nagyon zajosak voltak.




*70. Ábra 5-ös számú pont elmozdulására illesztett spline görbe*

A metrikus adatok kiszámolásához így nem illesztettem erre görbét, hanem a template matchingel meghatározott pontokat szűrtem meg és arra tettem ezt, ennek az eredménye a 70. ábrán látható. A leszűrt pontok elejét és végét ezt követően átlagoltam. Erre elsőként -16,09 cm-t kaptam, a mozgást követően pedig -20,18 cm-t. A kettő különbsége 4,08 centiméter, ami a mérőállomással meghatározott 6,06 centiméterhez képest 1,98 centivel kevesebb. Ez a hiba valószínűleg a korábban említett problémából ered, hogy a kamera nem rögzített tovább. Több későbbi adattal jobb eredményt kaphattam volna.




*71. Ábra Vízszintes elmozdulás*

Egy összehasonlítást tettem meg a kamerából kapott adatokkal. A vízszintes irányú mozgás az egész szerkezetre korreláltnak feltételezhető, ha az egy egységként mozdult. Ennek bizonyítására az adatokat időszinkronba hoztam és egy ábrán ábrázoltam őket. Ezt követően kiszámoltam a három adatsor között a korrelációt külön-külön. Az eredmények a 71. ábrán láthatóak.

Az adatsorok között a korrelációs együtthatók a következők lettek:

- $r_{3,4} = 0{,}2889$,
- $r_{3,5} = 0{,}6246$,
- $r_{4,5} = -0{,}1115$.

Valószínűleg az adatsorok zajossága miatt jöttek ki ilyen alacsony értékek. A legmagasabb abszolút értékben a 3-as és 5-ös pont között volt. Itt a zajos pontok is hasonló irányba és mértékben tértek el az elméleti helytől.

## 6.4 Összehasonlítás

| Magasság változás [m] | | | | |
|---|---|---|---|---|
| Pontszám | Szintezés | Mérőállomás | GNSS | Kamera |
| 1 | -0,2567 | -0,1076 | - | - |
| 2 | -0,1236 | 0,0154 | - | - |
| 3 | 0,0000 | 0,1426 | 0,1403 | - |
| 4 | -0,0907 | 0,0480 | - | 0,0569 |
| 5 | -0,1971 | -0,0606 | -0,0563 | -0,0408 |
| 6 | -0,3154 | -0,1823 | - | - |
| 7 | -0,4440 | -0,3141 | -0,2944 | - |
| 8 | -0,3439 | -0,2227 | - | - |

*1. Táblázat*

Az 1. táblázatban láthatóak a terheletlen és a terhelt állapot magasságainak különbségei a különböző technológiák által. Ha a mérőállomás által mért eredményeket vesszük etalonnak és ahhoz hasonlítunk, akkor a 2. táblázatban lévő eredmények jönnek ki.

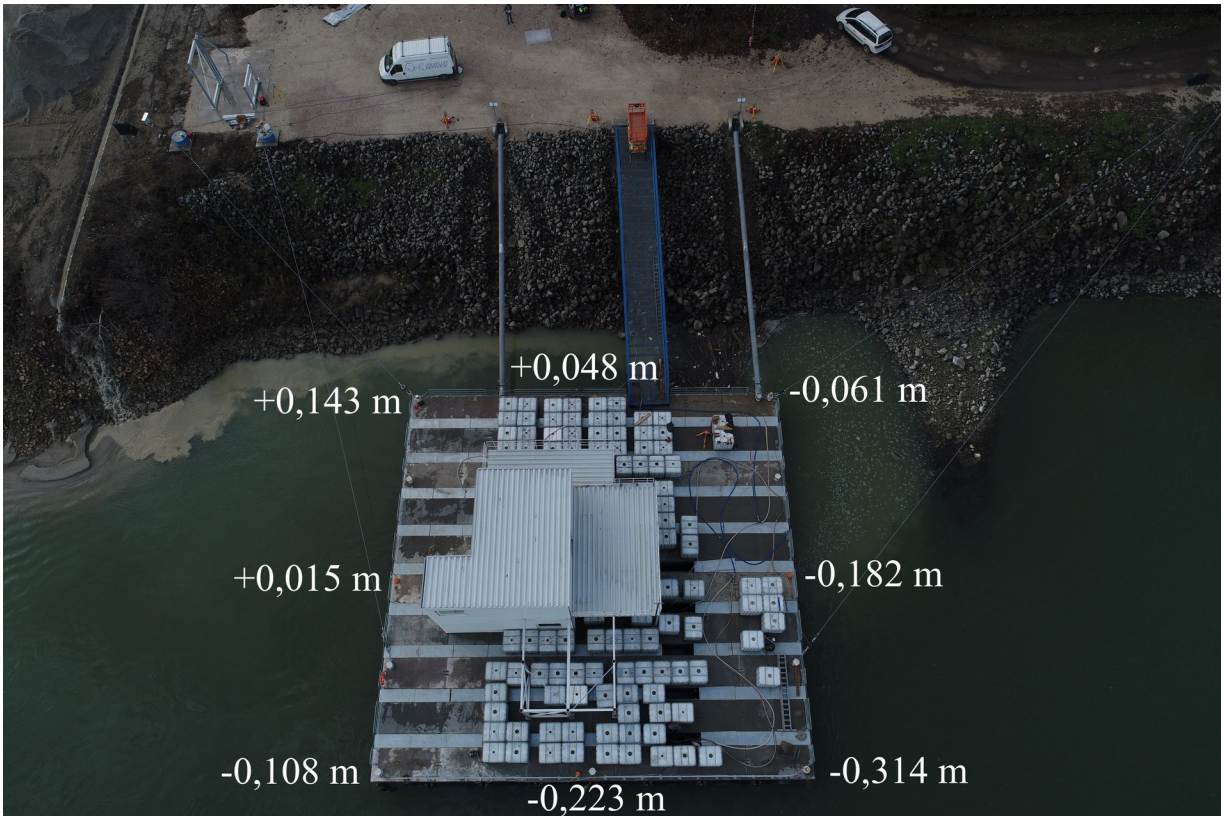
| Pontszám | Szintezés | GNSS | Kamera |
|---|---|---|---|
| 1 | 0,0065 | - | - |
| 2 | -0,0036 | - | - |
| 3 | 0,0000 | -0,0023 | - |
| 4 | -0,0040 | - | 0,0089 |
| 5 | -0,0061 | 0,0043 | 0,0198 |
| 6 | -0,0095 | - | - |
| 7 | -0,0127 | 0,0197 | - |
| 8 | -0,0215 | - | - |

*2. Táblázat: Mérőállomással szembeni eltérések [m]*

Az itt szereplő szintezésnél feltüntetett értékeket a 3-as ponthoz viszonyítottuk és eltoltuk azokat a mérőállomás által mért értékkel (lásd 61. ábra). Ezen értékeket elemezve látható, hogy a legtöbb esetben néhány milliméteres volt az eltérés, de legfeljebb 2,1 centiméter.

A GNSS és a kamera esetében is hasonló a helyzet, mindkettő néhány milliméterre hasonló értékeket adott a mérőállomáséhoz, kivéve 1-1 esetet.

A végső elmozdulás értékek a 72. ábrán láthatóak.




*72. Ábra: Elmozdulások [m]*
*Forrás: Takács Bence*

## 6.5 Konklúzió

Elmondható, hogy a 3 technológia közel hasonló eredményeket hozott. Nagy pontosságú mérésekre csak a robot mérőállomás (csak rövidebb mért távolságokon) és a szintező használható.

A robot mérőállomás által használt *robotplus.py* szkriptben a mérések során egy hiba tűnt fel. Ha a prizmák és a műszer egy vonalban van és az elmozdulások már viszonylag nagyok (szubdeciméteresek), ebben az esetben a műszer pár esetben félre azonosította (összekeverte) vagy teljesen kihagyta a pontok egy részét. Ez abból adódott, hogy a mozgásvizsgálati pontok azonosítása során csak az irányt vette figyelembe, így távolabb lévő pontok esetén nagy tűrés beállítása mellett megmért egy látómezejében lévő pontot. Ez a probléma már utólag (a távolság figyelembe vételéve) javítva lett.

A GNSS technológia megfelelően működött, itt csak az áramellátás megoldására kell figyelni. A mai akkumulátorok még nem képesek egy teljes napon át biztosítani az áramellátást.

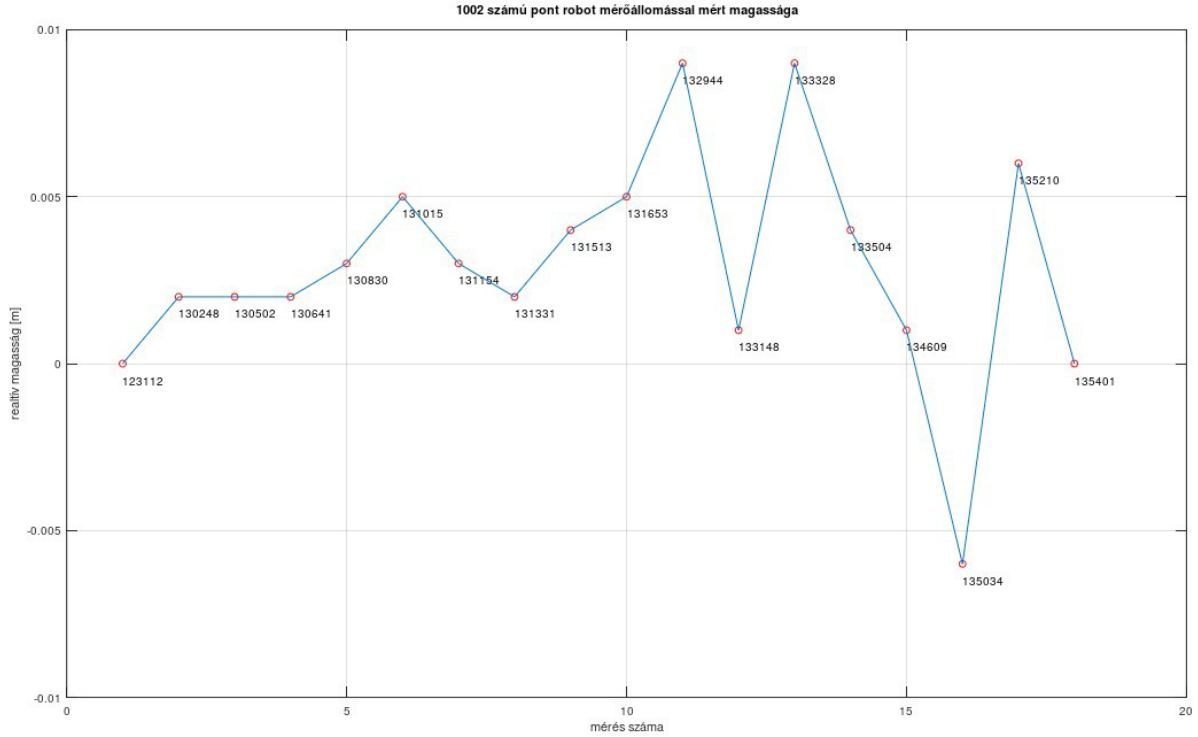
A kamerák használata még nem teljesen kiforrott kültéri, terepi használatra. A vízállóság kérdése a legfontosabb. A kameramodul alaplapja teljesen szabad a kültér felé, így ez könnyen elázhat és a Raspberry Pi-k háza sem szigetelt. Áramellátásról akár egy nagyobb powerbank vagy egy állandó áramforrással ellátott USB port is gondoskodhat. Terepen a megfelelő Wi-Fi lefedettség biztosítása sem egyszerű a kamerák elindítása érdekében, így erről is gondoskodni kell.

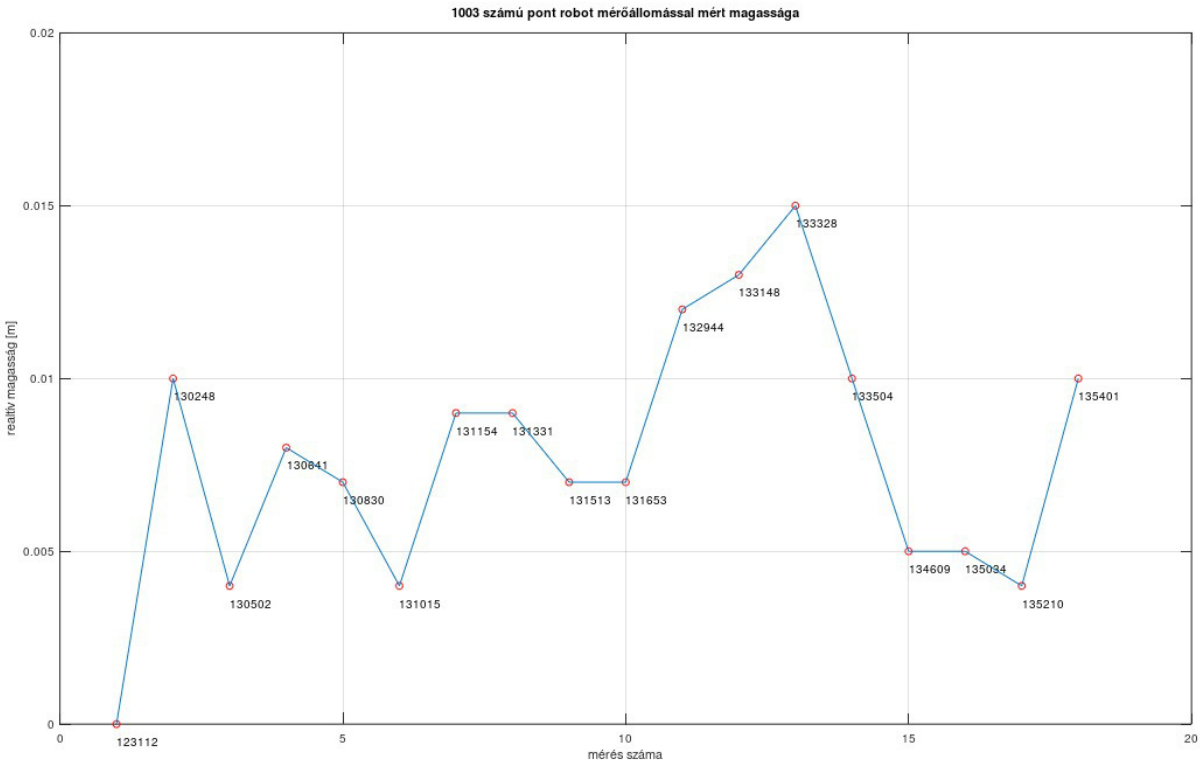
# Irodalomjegyzék

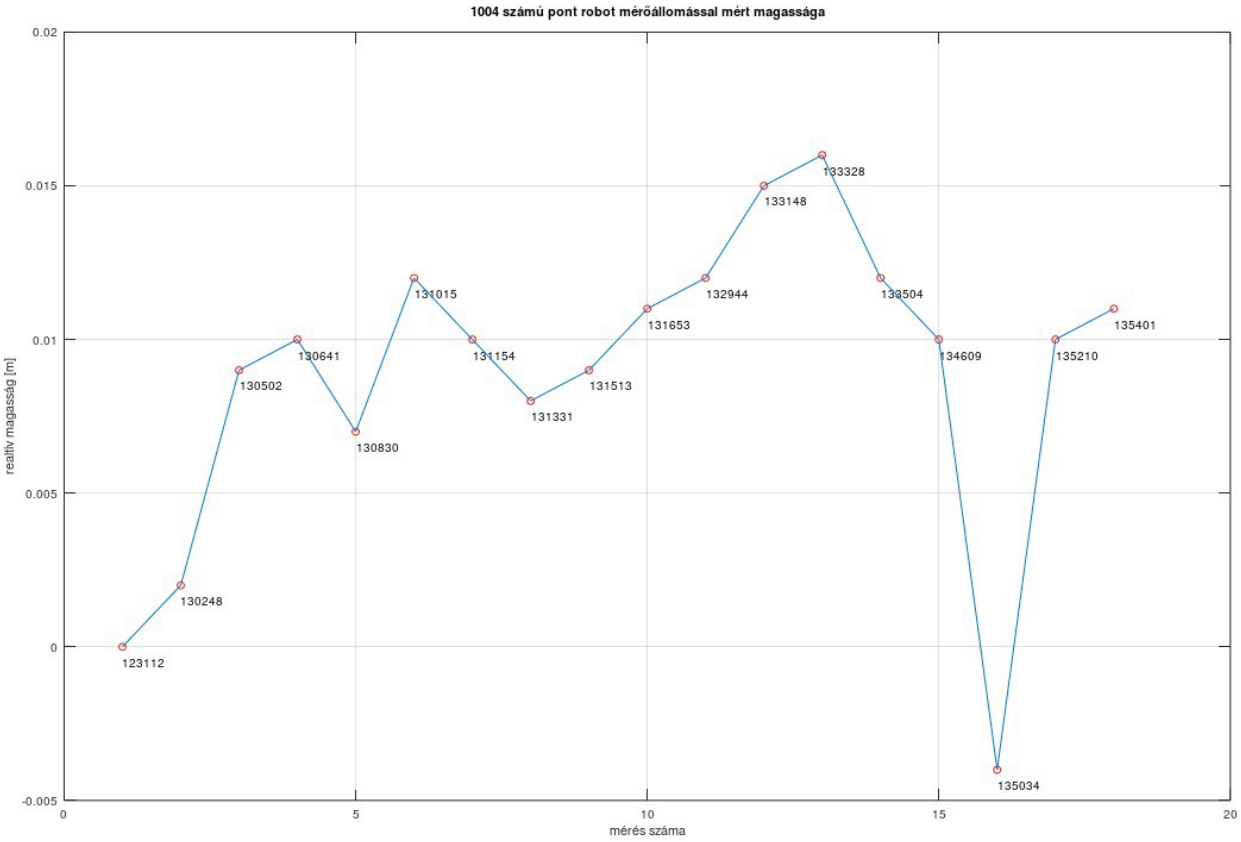
1: Dr. Farkas József, Józsa Vendel, Alapozás Előadás jegyzet, 2014
2: Detrekő Ákos, Ódor Károly, Ipari geodézia I-II, 1998
3: Király Tamás, Automatizált mérőrendszer alkalmazása és továbbfejlesztése,
4: Dr. Völgyesi Lajos, ISMÉTELT GEODÉZIAI, GEODINAMIKAI MÉRÉSEK ÉRTELMEZÉSE , 2014
5: Leica Geosystems, Monitoring Along the Construction Site of the Warsaw Metro Line 2,
6: Janusz Bogusz, Mariusz Figurski, Grzegorz Nykiel, Marcin Szolucha, Maciej Wrona, GNSS-based multi-sensor system for structural monitoring applications, 2012
7: A. Pellegrinelli, A. Furini, M. Bonfe`, P. Russo, Motorised digital levels: development and applications, 2013
8: Özgür AVŞAR, Devrim AKCA, Orhan ALTAN, PHOTOGRAMMETRIC DEFORMATION MONITORING OF THE SECOND BOSPHORUS BRIDGE IN ISTANBUL, 2014
9: Wikipédia, Nyúlásmérő bélyeg, 2019, https://hu.wikipedia.org/wiki/Ny%C3%BAl%C3%A1sm%C3%A9r%C5%91_b%C3%A9lyeg
10: Dr. Siki Zoltán, Ulyxes rendszer honlapja, 2019, http://www.agt.bme.hu/ulyxes/index_hu.html
11: Bánhidi Dávid, Automatizált monitoring rendszer alkalmazása, 2016
12: Wikipédia, Bluetooth, 2019, https://hu.wikipedia.org/wiki/Bluetooth
13: Wikipédia, PHP, 2019, https://hu.wikipedia.org/wiki/PHP
14: Wikipédia, PostGIS, 2019, https://en.wikipedia.org/wiki/PostGIS
15: Siki Zoltán, Takács Bence, Égető Csaba, Ulyxes: an open source project for automation in engineering surveying, 2018
16: Wikipédia, Topcon, 2019, https://en.wikipedia.org/wiki/Topcon
17: Xpert Survey, Topcon Hiper II Dual Frequency GNSS Receiver, 2019, https://www.xpertsurveyequipment.com/topcon-hiper-ii-dual-frequency-gnss-receiver.html
18: Wikipédia, Raspberry Pi, 2019, https://hu.wikipedia.org/wiki/Raspberry_Pi
19: Hi-Target, Hi-Target honlapja, 2019, http://en.hi-target.com.cn/product/detail.aspx?node=101005001&pid=106&catid=3

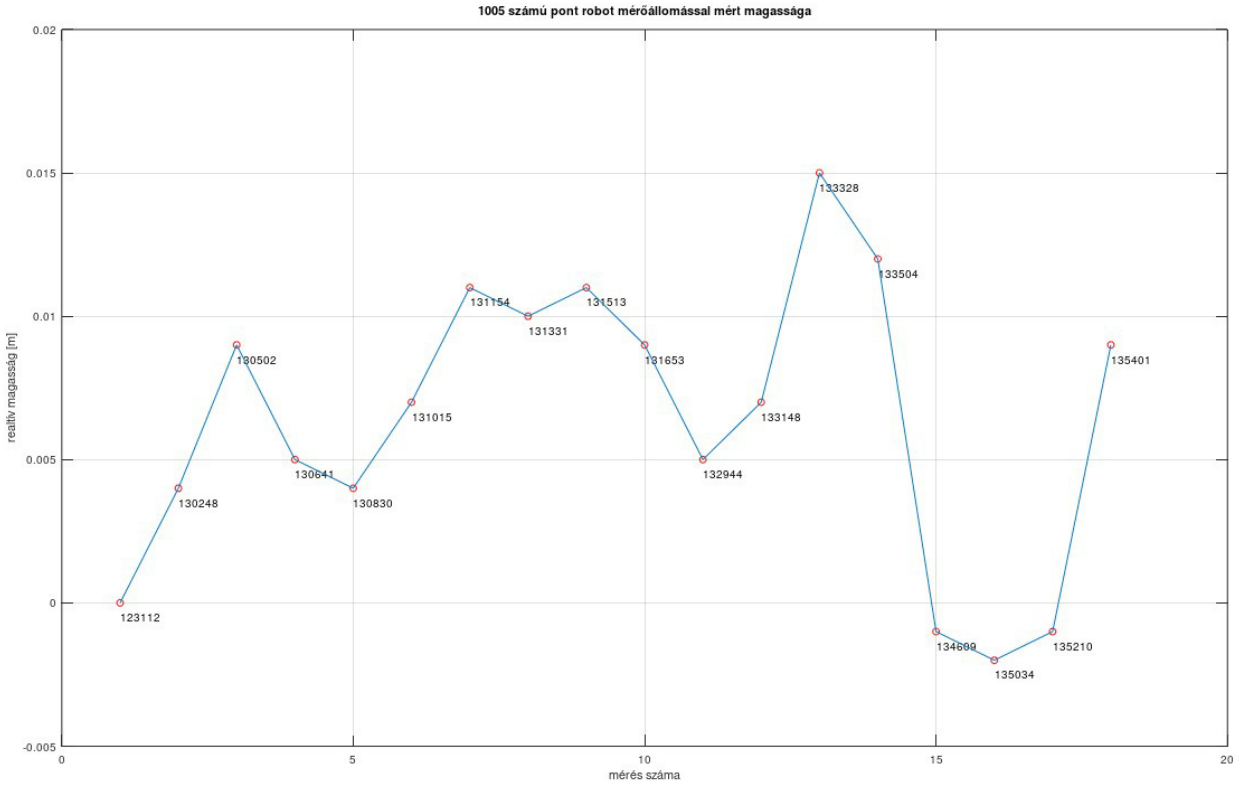
# Mellékletek

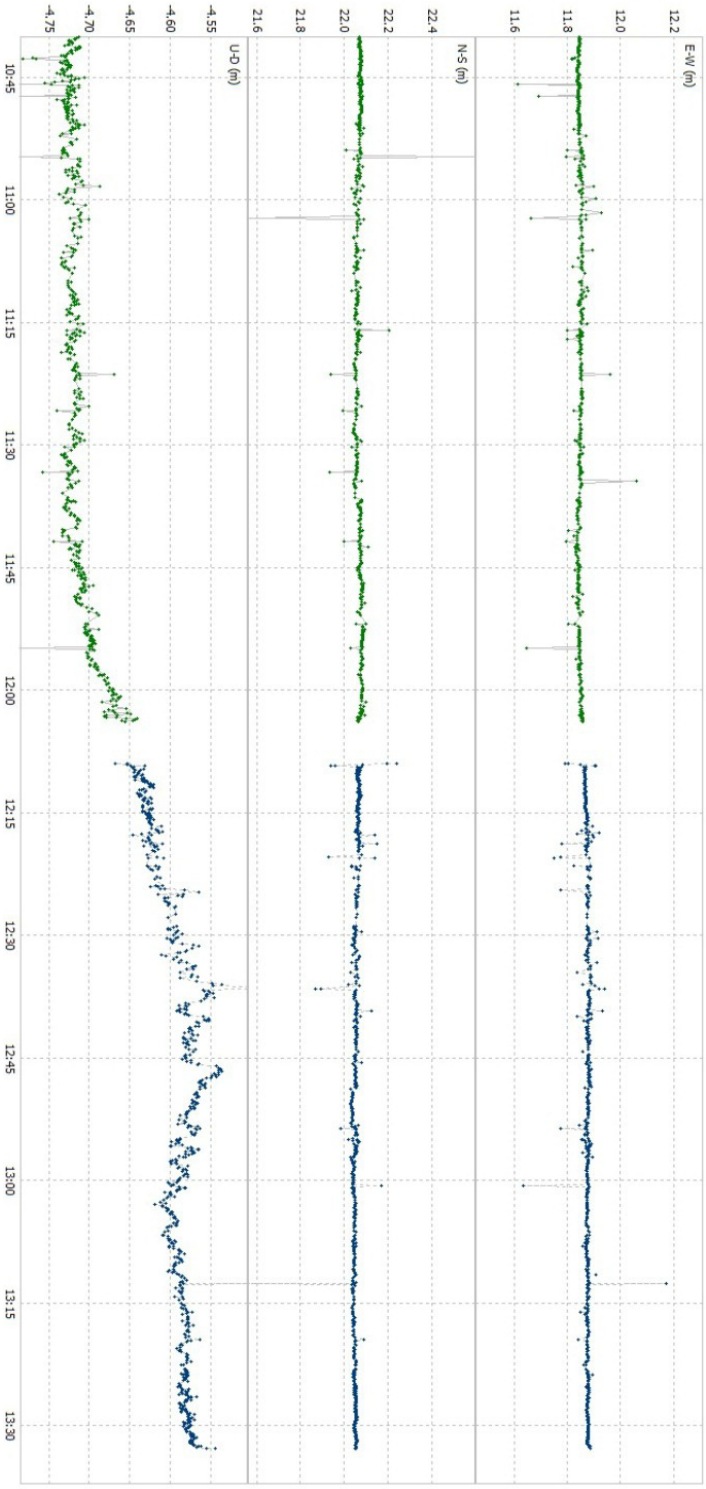
## Ponton mérés kimaradt ábrái



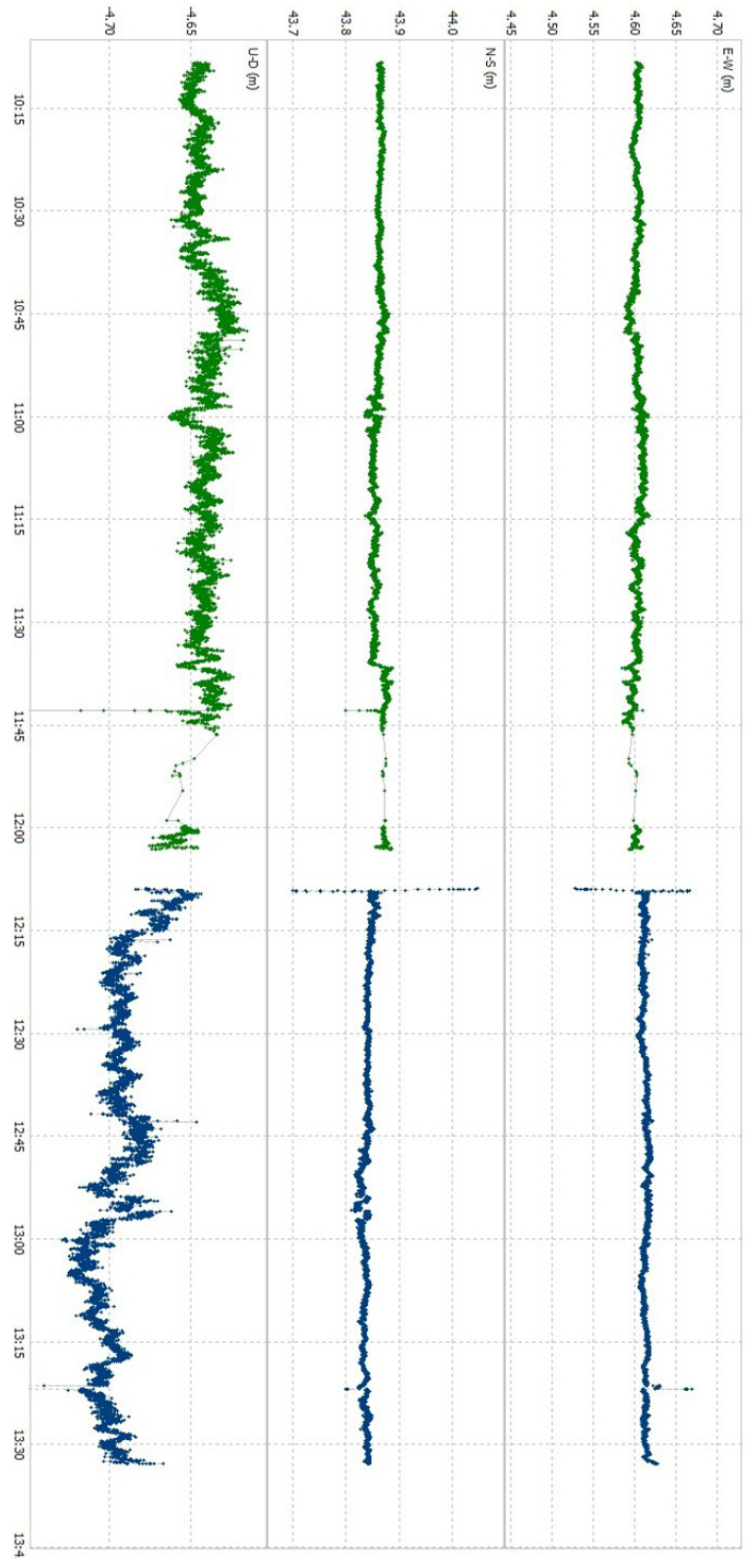




1004 számú pont robot mérőállomással mért magassága
realtív magasság [m]
mérés száma
123112
130248
130502
130641
130830
131015
131154
131331
131513
131653
132944
133148
133328
133504
134609
135034
135210
135401

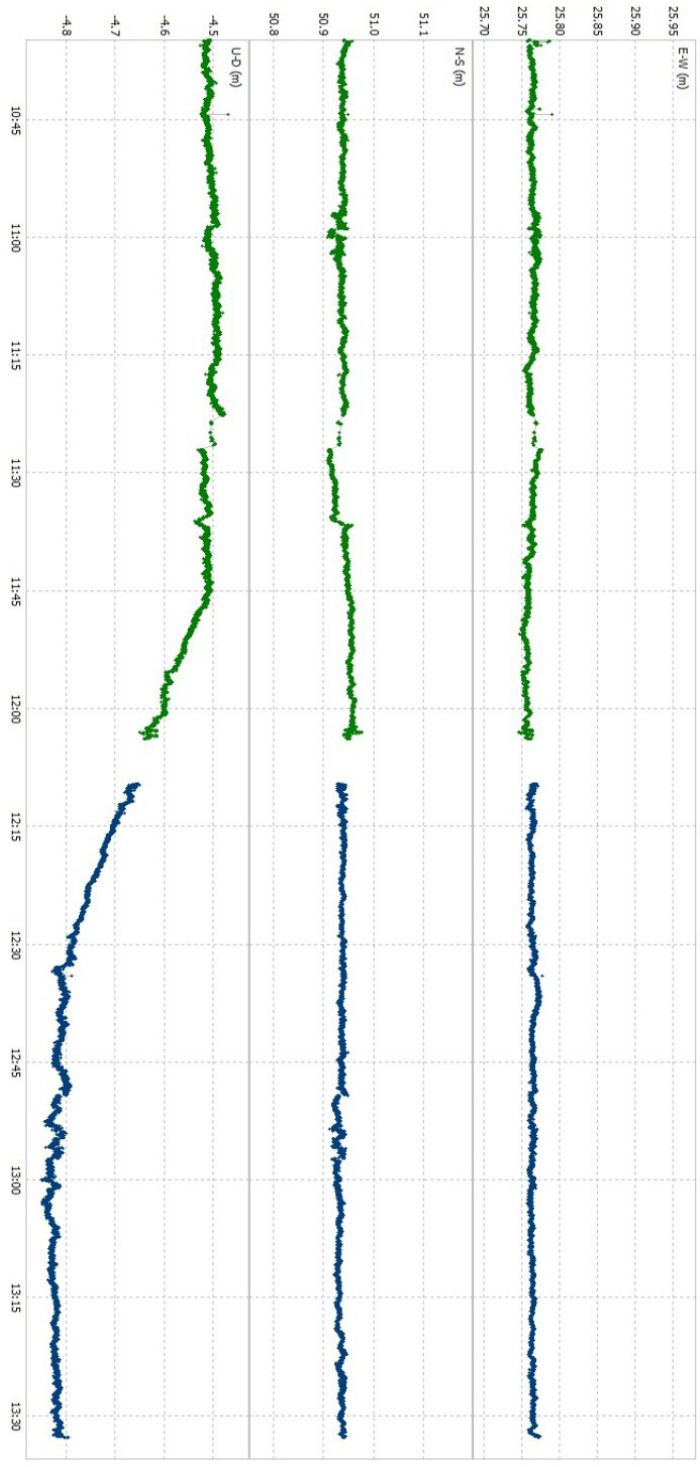

1005 számú pont robot mérőállomással mért magassága
realtív magasság [m]
mérés száma
123112
130248
130502
130641
130830
131015
131154
131331
131513
131653
132944
133148
133328
133504
134609
135034
135210
135401



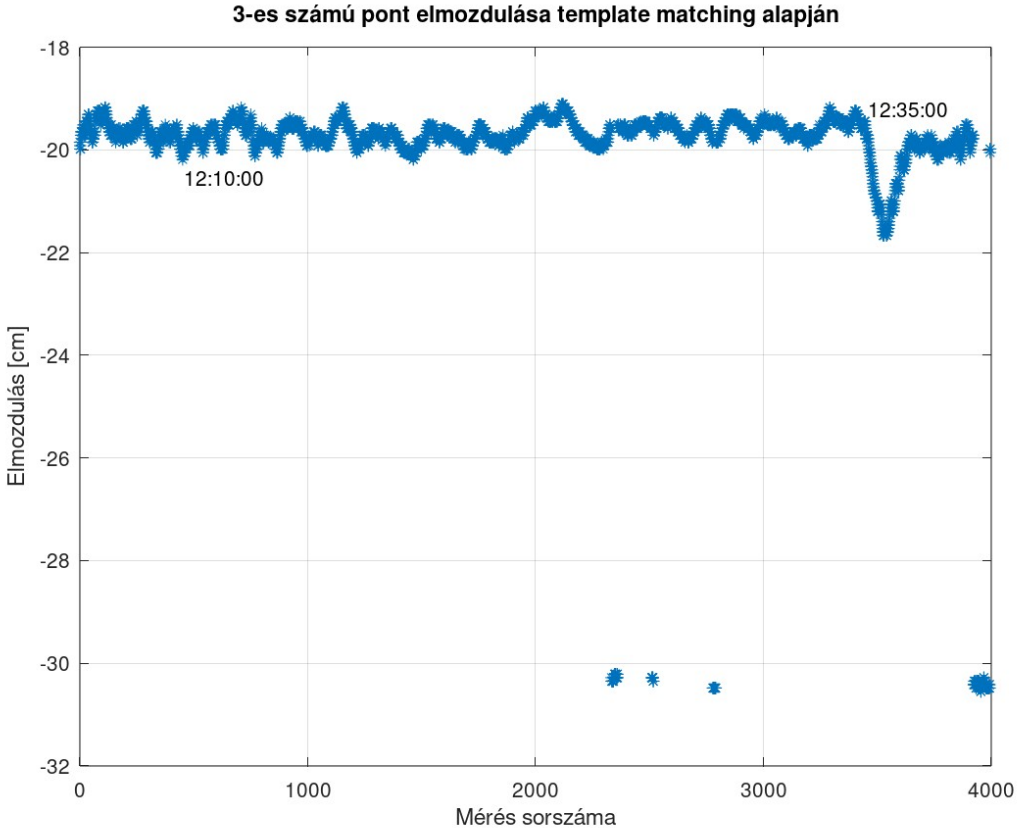

*3-as számú pont elmozdulásai GNSS alapján*

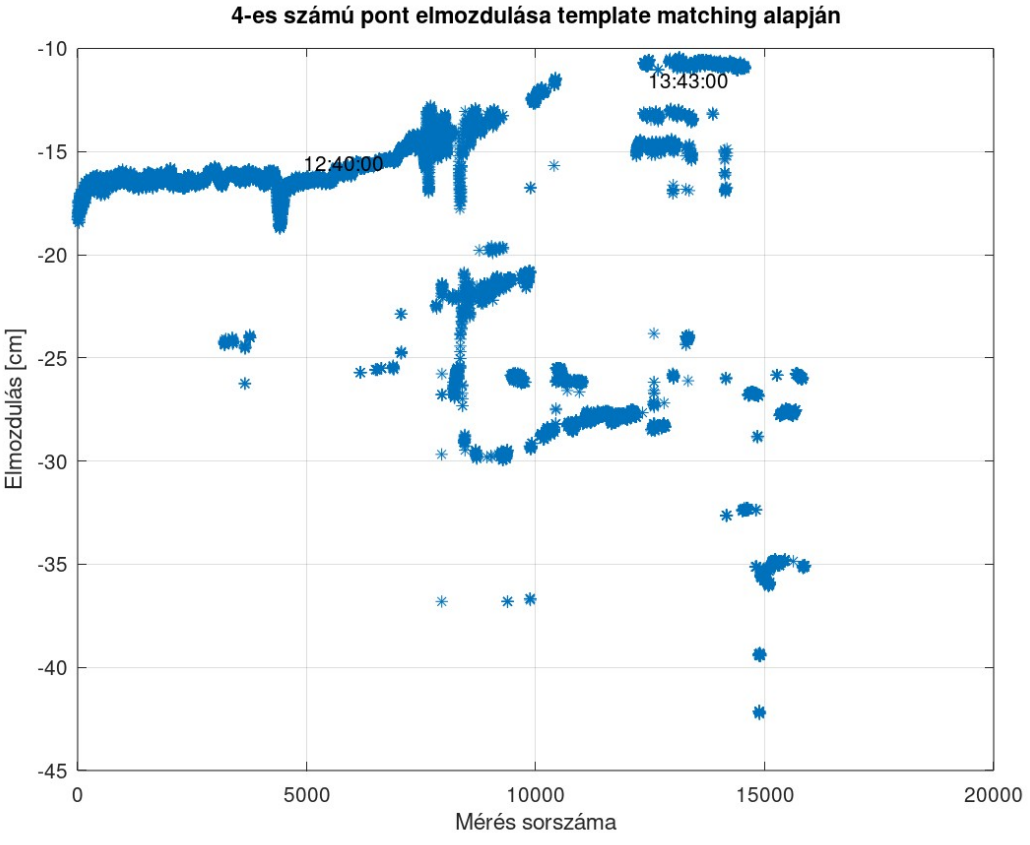


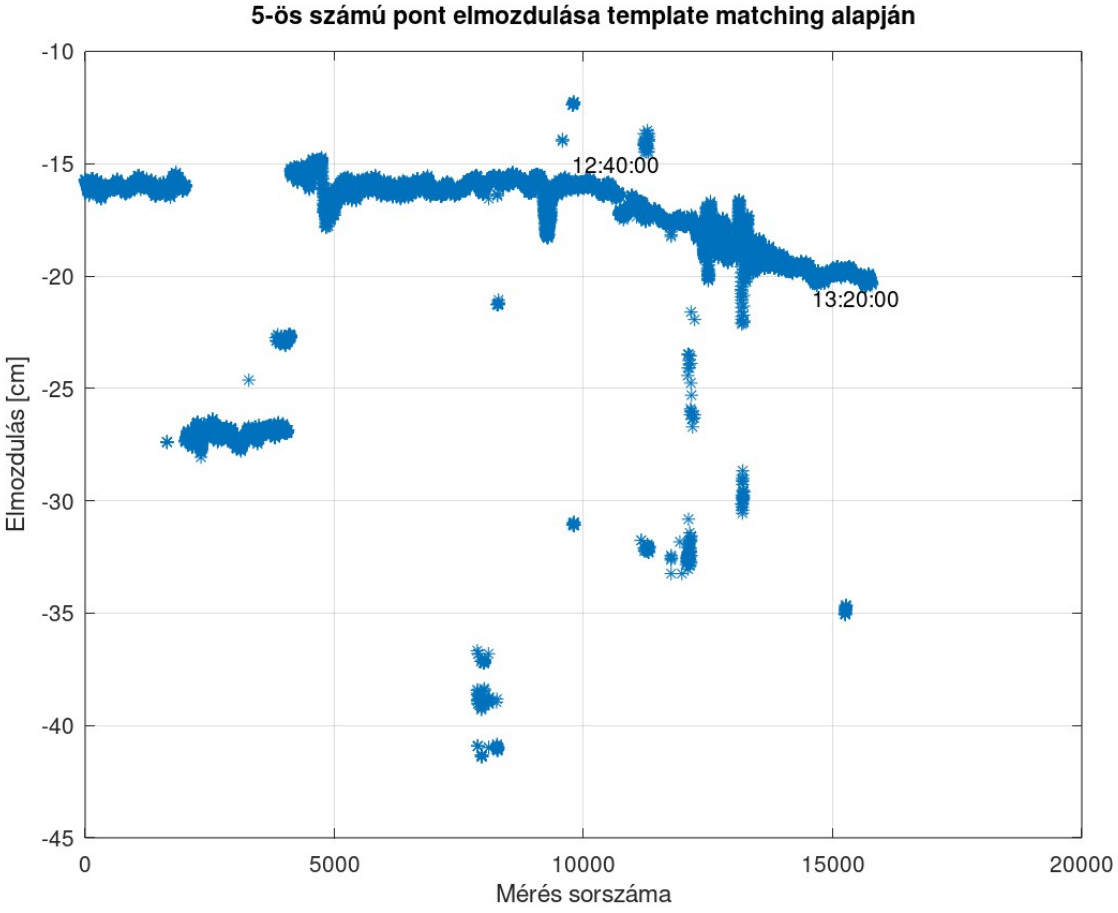

*5-ös számú pont elmozdulásai GNSS alapján*




*7-es számú pont elmozdulásai GNSS alapján*


3-es számú pont elmozdulása template matching alapján
12:10:00
12:35:00
Elmozdulás [cm]
Mérés sorszáma


4-es számú pont elmozdulása template matching alapján
12:40:00
13:43:00
Elmozdulás [cm]
Mérés sorszáma


5-ös számú pont elmozdulása template matching alapján
-10
-15
-20
-25
-30
-35
-40
-45
Elmozdulás [cm]
12:40:00
13:20:00
0
5000
10000
15000
20000
Mérés sorszáma

**Felhasznált Python és Octave kódok**

## nmea_collect.py

```
#!/usr/bin/env python
"""
.. module:: nmea_collect.py

.. moduleauthor:: dr. Zoltan Siki <siki.zoltan@epito.bme.hu>
                  Mate Kecskemeti <kecskemeti.mate3@gmail.com>

GNSS NMEA data collector. It can collect NMEA data from the GNSS receiver and write to
txt file.
Uses bluetooth or serial communication.

    :param argv[1] (port): serial port or bluetooth MAC address, default /dev/ttyUSB0
    :param argv[2] (output file): name of the output file , default results.txt

"""

import re
import sys
import msvcrt
import time

sys.path.append('../pyapi')

if __name__ == '__main__':
    # Choose between serial or bluetooth communication
    if len(sys.argv) > 1:
        cm = sys.argv[1]
    else:
        cm = '/dev/ttyUSB0'
    if re.search('^COM[0-9]$', cm) or re.search('^/dev/ttyUSB[0-9]$', cm):
        from serialiface import SerialIface
        iface = SerialIface('test', cm, 19200)
    else:
        from bluetoothiface import BluetoothIface
        iface = BluetoothIface('test', cm, 1)

    from nmeagnssunit import NmeaGnssUnit
    from csvwriter import CsvWriter
    from filewriter import FileWriter
    from gnss import Gnss

    #Making measurement unit
    mu = NmeaGnssUnit()

    #Writer unit creating
    if len(sys.argv) > 2:
        fn = sys.argv[2]
    else:
        fn = 'results.txt'
    wrt = CsvWriter('', 'DEG', '.3f', '%Y-%m-%d %H:%M:%S', ['id', 'latitude',
'longitude', 'altitude', 'datetime'], fn)
```

```
    #Get NMEA data
    g = Gnss('', mu, iface, wrt)
    if g.measureIface.state == g.measureIface.IF_OK:
        print('Press any key to finish measuring')
    while g.measureIface.state == g.measureIface.IF_OK:
        g.Measure()
        if msvcrt.kbhit():
            if msvcrt.getwche() == '\r':
                break
        time.sleep(0.1)
```

## bluetoothiface.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
.. module:: bluetoothiface.py
   :platform: Unix, Windows
   :synopsis: Ulyxes - an open source project to drive total stations and
       publish observation results. GPL v2.0 license Copyright (C)
       2010-2013 Zoltan Siki <siki.zoltan@epito.bme.hu>.

.. moduleauthor:: Zoltan Siki <siki.zoltan@epito.bme.hu>,
    Kecskeméti Máté <kecskemeti.mate3@gmail.com>
"""

import logging
import bluetooth
import time
from iface import Iface

class BluetoothIface(Iface):
    """ Interface to communicate through bluetooth interfacei as a client.
        This class depends on pybluez.

            :param name: name of bluetooth interface (str)
            :param mac: mac address of server to connect
            :param port: bluetooth port, default 3
            :param eomRead: end of message char from instrument (str), default '\\r\\n'
            :param eomWrite: end of message char from computer (str), default '\\r\\n'
    """

    def __init__(self, name, mac, port=3, timeout=5, eomRead='\r\n',
                 eomWrite='\r\n'):
        """ Constructor for bluetooth client
        """
        super(BluetoothIface, self).__init__(name)
        self.mac = mac
        self.port = port
        self.timeout = timeout
        self.eomRead = eomRead
        self.eomWrite = eomWrite
        self.socket = None
        self.Open()
```

```
def __del__(self):
    """ Destructor for bluetooth client
    """
    self.Close()
    self.socket = None

def Open(self):
    """ Open bluetooth communication
    """
    self.socket = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
    try:
        self.socket.connect((self.mac, self.port))
    except:
        logging.error(" error opening bluetooth connection")
        self.state = self.IF_SOURCE
        self.socket = None

def Close(self):
    """ Close bluetooth communication
    """
    try:
        self.socket.close()
    except:
        pass

def GetLine(self):
    """ read a line from bluetooth
    """
    if self.socket is None or self.state != self.IF_OK:
        logging.error(" bluetooth connection not opened or in error state")
        return None
    # read answer till end of message marker
    ans = ''
    w = -1 * len(self.eomRead)
    while ans[w:] != self.eomRead:
        ch = ''
        try:
            ch = (self.socket.recv(1)).decode('ascii')
        except:
            self.state = self.IF_READ
            logging.error(" cannot read bluetooth connection")
            break
        if ch == '':
            # timeout exit loop
            self.state = self.IF_TIMEOUT
            logging.error(" timeout on bluetooth")
            break
        ans += ch
    # remove end of line
    logging.debug(" message got: %s", ans)
    ans = ans.strip(self.eomRead)
    return ans

def PutLine(self, msg):
```

```
        """ Send message through bluetooth

            :param msg: message to send (str)
            :returns: 0 - on OK, -1 on error or interface is in error state
        """
        # do nothing if interface is in error state
        if self.socket is None or self.state != self.IF_OK:
            logging.error(" bluetooth connection not opened or in error state")
            return -1
        # add CR/LF to message end
        w = -1 * len(self.eomWrite)
        if msg[w:] != self.eomWrite:
            msg += self.eomWrite
        # remove special characters
        msg = msg.encode('ascii', 'ignore')
        # send message to bluetooth interface
        logging.debug(" message sent: %s", msg)
        try:
            self.socket.settimeout(self.timeout)
            self.socket.send(msg)
        except:
            self.state = self.IF_WRITE
            logging.error(" cannot write to bluetooth connection")
            return -1
        return 0

    def Send(self, msg):
        """ send message to bluetooth and wait for answer

            :param msg: message to send, it can be multipart message separated by '|'
(str)
            :returns: answer from instrument (str)
        """
        msglist = msg.split("|")
        res = ''
        #sending
        for m in msglist:
            if self.PutLine(m) == 0:
                time.sleep(5)
                res += self.GetLine() + '|'
        if res.endswith('|'):
            res = res[:-1]
        return res

if __name__ == "__main__":
    a = BluetoothIface('test', '00:12:F3:04:ED:06', 1)
    if a.GetState() == a.IF_OK:
        print(a.Send('%R1Q,2008:1,0'))
        print(a.GetState())
```

uszomu_fuggoleges.m

```
clc; clear all; close all;
pixel = -72/1070;

# point number 3
data3 = dlmread("results_3.txt", ",");
id3 = data3(:,1);
time3 = data3(:,3);
sor3 = data3(:,5) * pixel;

data_aruco3 = dlmread("results_aru3.txt", ",");
id_a3 = abs(data_aruco3(:,1));
time_a3 = abs(data_aruco3(:,4));
sor_a3 = abs(data_aruco3(:,6)) * pixel;

[Ca3, Ia3, I3a] = intersect(time_a3,time3);
oszlopa3_filt = sor_a3(Ia3);
oszlop3a_filt = sor3(I3a);
corra3 = corr(oszlop3a_filt,oszlopa3_filt)

# point number 4
data4 = dlmread("results_4.txt", ",");
id4 = data4(:,1);
time4 = data4(:,3);
sor4 = data4(:,5) * pixel;

k = 1;
for i = 1:length(sor4)
  if sor4(i) < -10 && sor4(i) > -20
    sor4_filt(k,1) = id4(i);
    sor4_filt(k,2) = sor4(i);
    sor4_filt(k,3) = time4(i);
    k += 1;
  endif
end

for i = 1:5500
  atlag_4t(i) = sor4_filt(i,2);
endfor

atlag_4e = mean(atlag_4t)

k = 1;
for j = 8721:length(sor4_filt)
  if sor4_filt(j,2) < -10 && sor4_filt(j,2) > -12
    atl_4(k) = sor4_filt(j,2);
    k += 1;
  endif
endfor

atlag_4v = mean(atl_4)
kul_4 = atlag_4v - atlag_4e

fit = splinefit(sor4_filt(:,1),sor4_filt(:,2),2);
```

```
val4 = ppval(fit, sor4_filt(:,1));

%aruco
data_aruco4 = dlmread("results_aru4.txt", ",");
id_a4 = abs(data_aruco4(:,1));
time_a4 = abs(data_aruco4(:,4));
sor_a4 = abs(data_aruco4(:,6)) * pixel;

fit = splinefit(id_a4(:),sor_a4(:),4);
val4_a = ppval(fit, id_a4(:));

%korrelacio
[Ca4, Ia4, I4a] = intersect(time_a4,time4);
oszlopa4_filt = sor_a4(Ia4);
oszlop4a_filt = sor4(I4a);
corra4 = corr(oszlop4a_filt,oszlopa4_filt)

# point number 5
data5 = dlmread("results_5.txt", ",");
id5 = data5(:,1);
time5 = data5(:,3);
sor5 = data5(:,5) * pixel;

k = 1;
for i = 1:length(sor5)
  if sor5(i) < -15 && sor5(i) > -21
    sor5_filt(k,1) = id5(i);
    sor5_filt(k,2) = sor5(i);
    sor5_filt(k,3) = time5(i);
    k += 1;
  endif
end

for i = 1:9000
  atlag_5t(i) = sor5_filt(i,2);
endfor

atlag_5e = mean(atlag_5t)

k = 1;
for j = 13117:length(sor5_filt)
  atl_5(k) = sor5_filt(j,2);
  k += 1;
endfor

atlag_5v = mean(atl_5)
kul_5 = atlag_5v - atlag_5e

fit = splinefit(sor5_filt(:,1),sor5_filt(:,2),3);
val5 = ppval(fit, sor5_filt(:,1));

data_aruco5 = dlmread("results_aru5.txt", ",");
id_a5 = abs(data_aruco5(:,1));
time_a5 = abs(data_aruco5(:,4));
sor_a5 = abs(data_aruco5(:,6)) * pixel;
```

```
[Ca5, Ia5, I5a] = intersect(time_a5,time5);
oszlopa5_filt = sor_a5(Ia5);
oszlop5a_filt = sor5(I5a);
corra5 = corr(oszlop5a_filt,oszlopa5_filt)

#plot
figure(1)
plot(id5,sor5, '*', "color", "b");
hold on;
title('5-ös számú pont elmozdulása template matching alapján');
xlabel('Mérés sorszáma');
ylabel('Elmozdulás [cm]');
for i = 1:length(time5)
  if time5(i) == 124000
    text(id5(i),sor5(i)+1,'12:40:00')
  elseif time5(i) == 132000
    text(id5(i),sor5(i)-1,'13:20:00')
  endif
end
hold off;

figure(2)
plot(id4,sor4, '*');
hold on;
title('4-es számú pont elmozdulása template matching alapján');
xlabel('Mérés sorszáma');
ylabel('Elmozdulás [cm]');
for i = 1:length(time4)
  if time4(i) == 124000
    text(id4(i),sor4(i)+1,'12:40:00')
  elseif time4(i) == 134245
    text(id4(i),sor4(i)-1,'13:43:00')
  endif
end
hold off;

figure(3)
plot(id3,sor3, '*');
hold on;
title('3-es számú pont elmozdulása template matching alapján');
xlabel('Mérés sorszáma');
ylabel('Elmozdulás [cm]');
for i = 1:length(time3)
  if time3(i) == 123500
    text(id3(i),sor3(i)+0.5,'12:35:00')
  elseif time3(i) == 121000
    text(id3(i),sor3(i)-0.5,'12:10:00')
  endif
end
hold off;

figure(4)
plot(sor5_filt(:,1),sor5_filt(:,2), '*');
hold on;
```

```
title('5-ös számú pont elmozdulása template matching alapján, szűrve');
xlabel('Mérés sorszáma');
ylabel('Elmozdulás [cm]');
plot(sor5_filt(:,1), val5 , "linewidth", 5, "color", "r");
hold off;

figure(5)
plot(id_a4(:), sor_a4(:), '*', "color", "b");
hold on;
title('4-es számú pont elmozdulása Aruco észlelés alapján');
xlabel('Mérés sorszáma');
ylabel('Elmozdulás [cm]');
plot(id_a4(:), val4_a(:) , "linewidth", 5, "color", "r");
hold off;

figure(6)
plot(id5,sor5, '*', "color", "b");
hold on;
plot(id_a5,sor_a5+5.7, '*', "color", "r");
title('5-ös számú pont elmozdulása template matching és Aruco azonosítás alapján');
xlabel('Mérés sorszáma');
ylabel('Elmozdulás [cm]');
for i = 1:length(time5)
  if time5(i) == 124000
    text(id5(i),sor5(i)+1,'12:40:00')
  elseif time5(i) == 132000
    text(id5(i),sor5(i)-1,'13:20:00')
  endif
end
legend( "Template matching","Aruco")
hold off;

figure(7)
plot(id4,sor4, '*', "color", "b");
hold on;
plot(id_a4,sor_a4+5.2, '*', "color", "r");
title('4-es számú pont elmozdulása template matching és Aruco azonosítás alapján');
xlabel('Mérés sorszáma');
ylabel('Elmozdulás [cm]');
for i = 1:length(time4)
  if time4(i) == 124000
    text(id4(i),sor4(i)+1,'12:40:00')
  elseif time4(i) == 134245
    text(id4(i),sor4(i)-1,'13:43:00')
  endif
end
legend( "Template matching","Aruco")
hold off;

figure(8)
plot(id3,sor3, '*');
hold on;
plot(id_a3,sor_a3+4.4, '*', "color", "r");
title('3-es számú pont elmozdulása template matching és Aruco azonosítás alapján');
xlabel('Mérés sorszáma');
```

```
ylabel('Elmozdulás [cm]');
for i = 1:length(time3)
  if time3(i) == 123500
    text(id3(i),sor3(i)+0.5,'12:35:00')
  elseif time3(i) == 121000
    text(id3(i),sor3(i)-0.5,'12:10:00')
  endif
end
legend( "Template matching","Aruco", "location", "southwest")
hold off;
```

## uszomu_vizszintes.m

```
clc; clear all; close all;
pixel = 72/1070;

# point number 3
data3 = dlmread("results_3.txt", ",");
id3 = data3(:,1);
time3 = data3(:,3);
oszlop3 = data3(:,4) * pixel;

# point number 4
data4 = dlmread("results_4.txt", ",");
id4 = data4(:,1);
time4 = data4(:,3);
oszlop4 = data4(:,4) * pixel;

# point number 5
data5 = dlmread("results_5.txt", ",");
id5 = data5(:,1);
time5 = data5(:,3);
oszlop5 = data5(:,4) * pixel;

[C45, I45, I54] = intersect(time4,time5);
[C35, I35, I53] = intersect(time3,time5);
[C34, I34, I43] = intersect(time3,time4);

%Korreláció számítás
oszlop45_filt = oszlop4(I45);
oszlop54_filt = oszlop5(I54);
cor45 = corr(oszlop45_filt,oszlop54_filt)

oszlop35_filt = oszlop3(I35);
oszlop53_filt = oszlop5(I53);
cor35 = corr(oszlop35_filt,oszlop53_filt)

oszlop34_filt = oszlop3(I34);
oszlop43_filt = oszlop4(I43);
cor34 = corr(oszlop34_filt,oszlop43_filt)

#Mátrixok létrehozása
sum(:,3) = oszlop5(:);
```

```
for i = 1:length(I53)
  sum(I53(i),1) = oszlop3(i);
endfor

for i = 1:length(I54)
  sum(I54(i),2) = oszlop4(i);
endfor

k = length(I54);
for j = length(sum):length(oszlop4)
  sum(j,2) = oszlop4(k);
  k += 1;
endfor

for i = 1:length(time4)
  sum(i,4) = i;
endfor

sum(sum == 0) = NaN;

# plot
figure(1)
plot(sum(:,3)-oszlop5(1),sum(:,4),'*', "color", "b");
hold on;
plot(sum(:,2)-oszlop4(1),sum(:,4), '*', "color", "r");
plot(sum(:,1)-oszlop3(1),sum(:,4), '*', "color", "g");
title('Vízszintes elmozdulás');
ylabel('Mérés sorszáma');
xlabel('Elmozdulás [cm]');
legend("5-ös pont","4-es pont","3-as pont")
hold off;
```

## ponton_meroallomas.m

```
clc; clear all; close all;

%meresi adatok levalogatasa
data = dlmread("out_coo.csv", ",");
points = unique(data(:,1));
pieces_temp = histc(data,points);
pieces = pieces_temp(:,1);

time = dlmread("time.txt");

for k = 1:length(points)
  first = sum(pieces(1:k-1))+1;
  last = sum(pieces(1:k));
  one_point = data(first:last,:);
  figure(k)
  for i = 1:length(one_point)
    if points(k) = one_point(i,1)
      h(i) = i;
      temp(i) = one_point(i,4) - one_point(1,4);
      plot(i,temp(i), 'o-r');
      hold on;
```

```
      text(i,temp(i)-0.0005,num2str(time(i)))
      xlabel ("mérés száma");
      ylabel ("realtív magasság [m]");
      cim = sprintf("%d számú pont robot mérőállomással mért magassága",points(k));
      title (cim);
      hold on;
    endif
  endfor
  fit = splinefit(h,temp,15);
  val = ppval (fit, h);
  plot(h, (val - val(1)));
  hold on;
endfor
```

## ponton_gnss.m

```
clc; clear all; close all;

adat = dlmread("nmea_coo_fix.csv", ",");
lat = adat(:,1);
long = adat(:,2);
height = adat(:,3);
time = adat(:,4) + 9988;

for i = 1:length(time)
  idtemp(i) = i;
endfor
id = idtemp';

figure(1)
plot(time, (height - height(1)))
xlabel ("mérés ideje [óóppmm]");
ylabel ("relatív magasság [m]");
title ('1001 számú pont GNSS mérései');
hold on;

fit = splinefit(time,height,20);
val = ppval(fit, time);
plot(time, (val - val(1)) , "linewidth", 4);
```

## ponton_korrelacio.m

```
clc; clear all; close all;

%total station
data = dlmread("robot/out_coo.csv", ",");
points = unique(data(:,1));
pieces_temp = histc(data,points);
pieces = pieces_temp(:,1);
first = sum(pieces(1:1))+1;
last = sum(pieces(1:2));
one_point = data(first:last,:);
time_robot = dlmread("robot/time.txt");
```

```
%gnss
adat = dlmread("gnss/nmea_coo_fix.csv", ",");
height = adat(:,3);
time_gnss = adat(:,4) + 10012;

%process
[C, Ir, Ig] = intersect(time_robot,time_gnss);

corr = corr(height(Ig),one_point(Ir,4))
```

## video_first.py

```
#! /usr/bin/python
# -*- coding: utf-8 -*-
"""
    save first video frame to an image file
"""
import cv2
import sys

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: %s video_file" % sys.argv[0])
        print("    video_file - video to process")
        exit(1)
    # selected method
    if sys.argv[1] in ("0", "1", "2", "3"):
        cap= cv2.videoCapture(int(sys.argv[1])) # open camera stream
        fps = 25    # TODO
    else:
        cap = cv2.VideoCapture(sys.argv[1])     # open video file
        fps = cap.get(cv2.CAP_PROP_FPS)
    if not cap.isOpened():
        print("Error opening video file")
    else:
        # process video
        ret, frame = cap.read() # get first frame
        if ret:
            cv2.imwrite(sys.argv[1] + ".png", frame)
    cap.release()
```

## video_correlation.py

```
#! /usr/bin/python
# -*- coding: utf-8 -*-
"""
    go through video frames to find template and correlation
    video file name can contain date and time like: pi1_YYYYmmdd_HHMMSS.h264
    output sent to standard output in the form:
    line,date/time,row,column,statistic
"""
import cv2
import sys
import datetime
```

```
import re

if __name__ == "__main__":
    if len(sys.argv) < 4:
        print("Usage: %s method_id template video_file" % sys.argv[0])
        print("    method_id - 0/1/2/3
CV_TM_SQDIFF_NORMED/CV_TM_CCORR_NORMED/CV_TM_CCOEFF/CV_TM_CCOEFF_NORMED")
        print("    template  - template image to find in the video frames")
        print("    video_file - video to process")
        print("    fps - frame/sec (optional, default 25)")
        exit(1)
    # selected method
    minv = 2    # index of min value
    maxv = 3    # index of max value
    methods = ((cv2.TM_SQDIFF_NORMED, minv), (cv2.TM_CCORR_NORMED, maxv),
        (cv2.TM_CCOEFF, maxv), (cv2.TM_CCOEFF_NORMED, maxv))
    method = methods[int(sys.argv[1])]
    # open template and convert to grayscale
    templ_gray = cv2.imread(sys.argv[2], cv2.IMREAD_GRAYSCALE)
    if sys.argv[3] in ("0", "1", "2", "3"):
        cap= cv2.videoCapture(int(sys.argv[3])) # open camera stream
        fps = 25    # TODO
        t = datetime.now()
    else:
        fn = sys.argv[3]
        if re.match('([a-zA-Z])*[0-9]_[0-9]{8}_[0-9]{6}', fn):
            l = fn.split('_')
            t = datetime.datetime(int(l[-2][0:4]), int(l[-2][4:6]), int(l[-2][6:8]),
                int(l[-1][0:2]), int(l[-1][2:4]), int(l[-1][4:6]))
            tformat = '%Y-%m-%d %H:%M:%S.%f'
        else:
            t = datetime.datetime(1970, 1, 1, 0, 0, 0)
            tformat = '%H:%M:%S.%f'
        cap = cv2.VideoCapture(fn)     # open video file
        fps = cap.get(cv2.CAP_PROP_FPS)
    if len(sys.argv) > 4:
        fps = int(sys.argv[4])
    if not cap.isOpened():
        print("Error opening video file")
    else:
        # process video
        i = 0   # frame id
        dt = datetime.timedelta(0, 1.0 / fps)
        while True:
            ret, frame = cap.read() # get next frame
            if ret:
                img_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                result = cv2.matchTemplate(img_gray, templ_gray, method[0])
                min_max = cv2.minMaxLoc(result)
                print("{:d},{:s},{:d},{:d},{:f}".format(i,
                    t.strftime(tformat),
                    min_max[method[1]][0], min_max[method[1]][1],
                    result[min_max[method[1]][0]][min_max[method[1]][1]]))
                i += 1
                t = t + dt
```

```
            else:
                break
    cap.release()
```

## video_aruco.py

```
#! /usr/bin/python
# -*- coding: utf-8 -*-
"""
    go through video frames to find ArUco codes (more than one)
    output sent to standard output in the form:
    aruco_id,date/time,row,column
"""
import sys
import re
import datetime
import numpy as np
import pandas as pd
import cv2
from cv2 import aruco

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: %s video_file fps" % sys.argv[0])
        print("    video_file - video to process or id of on-line video source")
        print("    fps - frame/sec (optional, default 25)")
        exit(1)
    # aruco dictionary
    aruco_dict = aruco.Dictionary_get(aruco.DICT_4X4_100)
    parameters =  aruco.DetectorParameters_create()
    if sys.argv[1] in ("0", "1", "2", "3"):
        cap= cv2.videoCapture(int(sys.argv[3])) # open camera stream
        fps = 25    # TODO
        t = datetime.now()
    else:
        fn = sys.argv[1]
        if re.match('([a-zA-Z])*[0-9]_[0-9]{8}_[0-9]{6}', fn):
            l = fn.split('_')
            t = datetime.datetime(int(l[-2][0:4]), int(l[-2][4:6]), int(l[-2][6:8]),
                int(l[-1][0:2]), int(l[-1][2:4]), int(l[-1][4:6]))
            tformat = '%Y-%m-%d %H:%M:%S.%f'
        else:
            t = datetime.datetime(1970, 1, 1, 0, 0, 0)
            tformat = '%H:%M:%S.%f'

        cap = cv2.VideoCapture(fn)     # open video file
        fps = cap.get(cv2.CAP_PROP_FPS)
    if len(sys.argv) > 2:
        fps = int(sys.argv[2])
    if not cap.isOpened():
        print("Error opening video file")
    else:
        # process video
        i1 = 0  # frames
        i2 = 0  # frames with found aruco
```

```
        i3 = 0  # frames without found aruco
        dt = datetime.timedelta(0, 1.0 / fps)
        while True:
            ret, frame = cap.read() # get next frame
            if ret:
                i1 += 1
                img_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                corners, ids, rejectedImgPoints = aruco.detectMarkers(img_gray,
aruco_dict, parameters=parameters)
                if ids is None:
                    i3 += 1
                    continue
                corners2 = np.array([c[0] for c in corners])
                data = pd.DataFrame({"x": corners2[:, :, 0].flatten(),
                    "y": corners2[:, :, 1].flatten()},
                     index=pd.MultiIndex.from_product(
                        [ids.flatten(), ["c{0}".format(i)
                            for i in np.arange(4)+1]], names=["marker", ""]))

                data = data.unstack().swaplevel(0, 1, axis=1).stack()
                # calculate center of markers
                data["m1"] = data[["c1", "c2"]].mean(axis=1)
                data["m2"] = data[["c2", "c3"]].mean(axis=1)
                data["m3"] = data[["c3", "c4"]].mean(axis=1)
                data["m4"] = data[["c4", "c1"]].mean(axis=1)
                data["o"] = data[["m1", "m2", "m3", "m4"]].mean(axis=1)
                # output found markers
                for i in range(ids.size):
                    j = ids[i, 0]
                    x = int(data['o'][j]['x'])
                    y = int(data['o'][j]['y'])
                    print('{:d},{:s},{:d},{:d}'.format(j, t.strftime(tformat), x ,y))
                i2 += 1
                t = t + dt
            else:
                break
    cap.release()
    print(i1, i2, i3)
```