

TDK dolgozat

Faragó Attila

2021

Neumann János Egyetem
GAMF Műszaki és Informatikai Kar
Informatika tanszék

A gravitációs hullámok adatainak strukturált beolvasása és megjelenítése LabVIEW környezetben

Szerző:

Faragó Attila
(3. évfolyam)

Témavezető:

Dobján Tibor
(főiskolai tanársegéd)

2021. november

Tartalomjegyzék

1. CÉLOK	4
2. A GRAVITÁCIÓS HULLÁMOK	5
2.1. MI AZ A GRAVITÁCIÓS HULLÁM?	5
2.2. A HULLÁMOK ÉRZÉKELÉSE – A MICHELSON-INTERFEROMÉTER	5
2.3. A FABRY-PÉROT REZONÁTOR	7
3. A GRAVITÁCIÓS HULLÁMOK ÉRZÉKELÉSÉRE ALKALMAS INTERFEROMÉTEREK	8
3.1. LIGO (LASER INTERFEROMETRY GRAVITATIONAL WAVE OBSERVATORY)	8
3.1.1. A korai (Initial) LIGO	8
3.1.2. A fejlesztett (Advanced) LIGO	8
3.2. VIRGO	9
3.2.1. A korai (Initial) Virgo	9
3.2.2. A fejlesztett (Advanced) Virgo	10
3.3. KAGRA (KAMIOKA GRAVITY AND GRAVITATIONAL WAVES)	10
4. AZ OBSZERVATÓRIUMOK MEGFIGYELÉSI IDŐSZAKAI	11
5. HDF FÁJLOK	12
5.1. HDF4	12
5.2. HDF5	12
5.2.1. Alkalmazási területek	13
5.2.2. A HDF5 szerkezeti felépítése	14
5.3. HDF VIEW	14
6. A BEOLVASÁST VÉGZŐ FŐPROGRAM FELÉPÍTÉSE	15
6.1. SZOFTVERKÖRNYEZET	15
6.2. HDF5 ADATSTRUKTÚRA KIOLVASÁSA	15
6.2.1. A letöltött HDF5 fájl megnyitása a szoftverkörnyezetben	15
6.2.2. A csoportobjektumok rekurzív listázása az objektumok típusa szerint	16
6.2.3. A HDF5 adatállományának megnyitása	17
6.3. ADATKLASZTER LÉTREHOZÁSA AZ ADATTÍPUSOK KINYERÉSE CÉLJÁBÓL	18
6.4. A METAADATOK KINYERÉSÉHEZ HASZNÁLT STRUKTÚRÁK	20
6.4.1. A „meta/Description” esete	21
6.4.2. A „meta/DescriptionURL” esete	22
6.4.3. A „meta/Detector” esete	22
6.4.4. A „meta/Duration” esete	22
6.4.5. A „meta/GPSstart” esete	23
6.4.6. A „meta/Observatory” esete	23
6.4.7. A „meta/Type” esete	23
6.4.8. A „meta/UTCstart” esete és az időbélyeg konverziós „subVI”	24
6.5. A „QUALITY” KÖNYVTÁR ADATAINAK ÉS MASZKJAINAK KINYERÉSE ÉS AZ ADATOK JELENTÉSÉNEK MAGYARÁZATA	26
6.5.1. A „quality/injections/InjDescriptions” esete	26
6.5.2. A „quality/injections/InjShortnames” esete	26
6.5.3. A „quality/injections/Injmask” esete	27
6.5.4. A „quality/simple/DQDescriptions” esete	30
6.5.5. A „quality/simple/DQShortnames” esete	31
6.5.6. A „quality/simple/DQmask” esete	32
6.6. A „STRAIN/STRAIN” NYÚLÁS IDŐJEL ADATOK KINYERÉSE	33
6.7. AZ ADATTÍPUS MEGÁLLAPÍTÁSA AZ ADATKÉSZLET ELEMEINÉL	34
6.8. A KIOLVASÁST VÉGZŐ FŐPROGRAM BEZÁRÁSA	36
7. JELFELDOLGOZÓ ÉS MÉRÉSTECHNIKAI MELLÉKPROGRAMOK, PROGRAMOPTIMALIZÁLÁS	37

7.1.	PROGRAMBEHÍVÓ VI BEMUTATÁSA.....	37
7.2.	A „STRAIN” ADATOK „WAVEFORM GRAPH”-JÁNAK FORMÁZÁSÁRA ÉS FELCÍMKÉZÉSÉRE SZOLGÁLÓ „SUBVI” BEMUTATÁSA.....	37
7.3.	A „STRAIN” ADATOK RÉSZLEGES KIOLVASÁSA ÉS ÁBRÁZOLÁSA	39
7.4.	SPEKTROGRAM ALKALMAZÁSA A HULLÁMOK ÁBRÁZOLÁSÁRA.....	43
7.5.	PROGRAMOPTIMALIZÁLÁS.....	44
8.	KONKLÚZIÓ.....	46
9.	FELHASZNÁLT IRODALOM	47

1. Célok

Az iparban és a tudományban is elengedhetetlen olyan segédprogramok használata, amelyek megkönnyítik az adatok rendszerezését, feldolgozását és az adatok további felhasználása során a kalkulációt.

Munkám célja a National Instruments által fejlesztett LabVIEW programmal történő adatfeldolgozás. A TDK referátum írása során elkészíték egy a gravitációs hullámok kutatásával foglalkozó obszervatórium, a LIGO által gyűjtött adatbank feldolgozását elvégző programcsomagot.

Az elkészült fő- és mellékprogramoktól az az elvárás, hogy egy bárki által hozzáférhető és letölthető adatfeldolgozó programot alkossanak, amelyeket minden olyan ember képes használni, aki már minimális, felhasználói szinten alkalmazza a LabVIEW programot. Céлом, hogy az általam készített programcsomag alkalmazásával kikerülhető legyen a HDF5 (Hierarchical Data Format – hierarchikus adatformátum) formátum kezeléséhez szükséges ismeretek elsajátítása, ezáltal a tudományos életben kezdő, vagy azon témákat mindössze hobbi célból űző emberek is hozzáférhessenek és könnyebben beleláthassanak a gravitációs hullámokat tároló tudományos adatbankokba.

Munkám célzottan a LIGO csillagászati obszervatórium által gyűjtött adatokra fókuszál, és annak megismerését szolgálja, jellegét tekintve tulajdonképpen egy HDF5 formátumú adathalmazt konvertál, így később nyílt felhasználhatósága révén kifejezetten, de nem kizárólag, csillagászati, meteorológiai kutatási munkákhoz is alkalmazható. A programcsomag megalkotását követően optimalizálási műveleteket is végzek, ami a stabilitást növeli és a futtatási időket csökkenti.

A dolgozatot a tudományos háttér feldolgozásával kezdem, megismertetem a gravitációs hullámokat, az őket kimutatni képes interferométereket és azok fizikai elvét. A TDK munkát a HDF fájlok bemutatásával folytatom, kitérve az alkalmazási területekre és a szerkezeti felépítésre. A dolgozat legfőbb része a program bemutatása lesz. Ezekben a fejezetekben pontról pontra ismertetem a program felépítését, a használt parancsokat és a kapott eredményeket. A TDK munkát egy olyan fejezettel zárom, amiben a főprogram bemutatását követően, mellékprogramokat készítek, melyekben olyan példaalkalmazásokat szemléltetek, amikkel demonstrálom, hogy a kiolvasott adatokkal további jelfeldolgozó és mérés technikai munkák végezhetők.

2. A gravitációs hullámok

2.1. Mi az a gravitációs hullám?

A gravitációs hullámok témakörét a speciális relativitáselmélet tárgyalja. Albert Einstein, 1915-ben publikált művében, az általános relativitáselméletben vizsgálta a térbeli és időbeli távolságok mérésének szabályát arra a speciális esetre vonatkozóan, ha gyorsuló megfigyelők végeznek mérést.

Einstein felfedezte, hogy a gravitációs mezőkben mozgó megfigyelők mindig ugyanabban a gravitációs mezőben és mindig azonos mértékben, illetve módon gyorsulók. Ezek tudatában kijelenthető, hogy az idő és a távolság mérésének szabályait a gravitációs mezőkben a mezőkhöz kell rendelni, a 4 dimenziós téridő helyfüggő görbületeként. Einstein úgy értelmezte, hogy a gravitációs mezőben mozgó testekre a gravitáció a téridő görbültégeként, nem pedig erőként hat. Ez a görbültés képes eltéríteni, illetve befolyásolni a testek mozgását.

A téridő görbültését a testek alakítják ki, de a téridő görbülete befolyásolja a testek mozgását. Fontos felfedezés, hogy hatással lehetünk a téridő görbültésére, amennyiben testeket mozgatunk. Ez a változó görbültés ki is mérhető, ehhez más testek mozgásának megfigyelése szükséges.

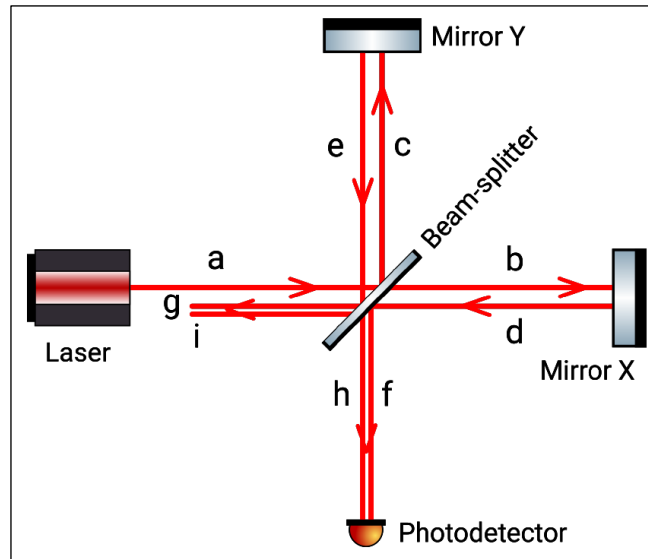
A gravitációs hullámok tehát a téridőnek hullámszerűen ingadozó és tovaterjedő fodoródásai. 1916-ban a világhírű tudós azt is megjósolta, hogy elmélete alapján ezeknek a hullámoknak létezniük kell. Ugyan bizonyítani még nem tudta, de előre megadta a hullámok pontos tulajdonságait. Ezek közé a tulajdonságok köz tartozik, hogy a hullámok transzverzálisak, fénysebességgel terjednek, amplitúdójuk a forrásuktól távolodva egyenes arányban csökken. [4]

2.2. A hullámok érzékelése – a Michelson-interferométer

Albert Michelson, amerikai fizikus 1881-ben Berlinben épített egy interferométert, amelyet ma Michelson-interferométer néven ismer a tudomány. Az első ilyen készülék célja a relatív fénysebesség mérése volt, két egymásra merőleges irányú fényúton.

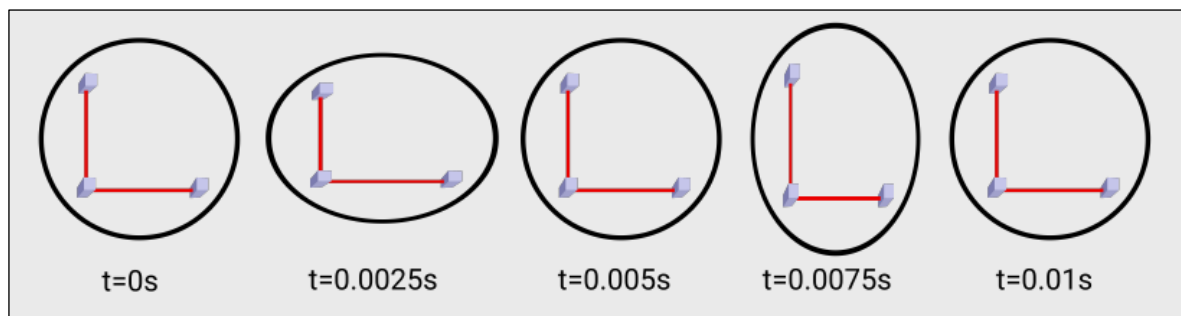
Az interferométer (1. ábra) két tükörből (Mirror X és Mirror Y) és egy félig fényáteresztő tükörből, a nyalábosztóból (beam-splitter) áll, melyet a tükrök közé helyeznek. Az elhelyezés speciális, mivel olyan, hogy a nyalábosztóra kibocsátott fénynyaláb egyik fele a Mirror X-re, tehát a mögötte lévő tükörrre esik, míg a másik fele visszaverődik a Mirror Y-

ra, tehát a másik oldalon elhelyezett tükörrre. A nyalábosztó hátoldalán kilépő két párhuzamos fénysugár ezáltal egymással interferál.



1. ábra: A Michelson-interferométer elvi ábrája [5]

Hogyan befolyásolja a gravitációs hullám az interferométert? A gravitációs hullám ciklusával szinkronban az interferométer egyik karja kissé hosszabb lesz, míg a másik kar kissé rövidebbé válik. Ez a különbség a két kar fényhullámának kicsi, de egyértelműen észlelhető fáziseltolódását okozza egymáshoz képest. A 2. ábra a Michelson-interferométer gravitációs hullám által erősen eltúlzott alakváltozását mutatja. Az öt részkép a gravitációs hullám teljes ciklusát mutatja be.



2. ábra: A Michelson-interferométer gravitációs hullám miatti összenyomódása és megnyúlása [5]

Az interferométer érzékenységének javításában három alapvető tervezési tényező vesz részt:

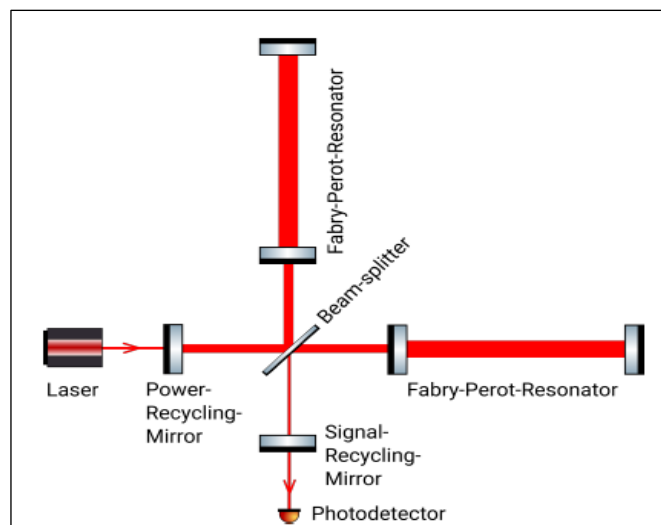
1. a karok hossza (Az első tényező, a kar hossza azért jelentős, mert minél hosszabb az interferométer karja, annál erősebb a gravitációs hullámjel.)

2. a tükrök által keltett belső zaj mennyisége (A zajnak két forrása lehet: az egyik a szeizmikus szigetelés jóságából fakadó, a másik a függesztett tükrök apró mozgásaiból származó zaj, amelyeket az őket alkotó anyagok hőingadozásai generálnak. Minél kisebb a zaj, annál jobbak a körülmények.)
3. a fény mennyisége (A harmadik tényező, a fény azért fontos, mert a több fény megnöveli az interferométer-karok sugarai közötti fáziskülönbség mérési pontosságát.)

2.3. A Fabry-Pérot rezonátor

A Fabry-Pérot rezonátort (3. ábra) 1897-ben Charles Fabry és Alfred Pérot francia fizikusok fejlesztették ki. Két, részben áttetsző tükörből áll, amelyek egymással párhuzamosan vannak elrendezve, pontosan beállított távolságra.

Fabry és Pérot eredetileg a különböző hullámhosszú fények elválasztására fejlesztették ki a készüléket, de közben felfedezték, hogy a konstruktív interferencia a rezonátorban keringő fény erejét is növeli. A fényerő növekedése nem sérti az energiamegmaradás törvényét, hiszen a fény felhalmozásán, illetve tárolásán alapul.



3. ábra: Fabry-Pérot rezonátorokkal továbbfejlesztett Michelson-interferométer. [5]

A gravitációs hullámok detektálására optimalizált Michelson-interferométerben (3. ábra) a Fabry-Pérot rezonátoroknak két fontos funkciója van. Az egyik, a tényleges karhosszabbítás és a gravitációs hullámjel optimális kivonása. Az interferométer fénymennyiségének növelésére legfeljebb négy ilyen rezonátorelrendezést alkalmaznak. A második fontos funkció, hogy a Fabry-Pérot rezonátorokat úgynevezett üzemmód-tisztítóként használják, amelyek optikailag tisztítják a fénysugarakat geometriai és időbeli szűréssel.

3. A gravitációs hullámok érzékelésére alkalmas interferométerek

3.1. LIGO (Laser Interferometry Gravitational Wave Observatory)

3.1.1. A korai (Initial) LIGO

A LIGO projektet (Laser Interferometry Gravitational Wave Observatory) 1984-ben indult el Rainer Weiss, Kip Thorne és tudós kollégáik segítségével. A projekt beindulását a kaliforniai és a massachusetts-i technológiai intézetek, a Caltech és az MIT támogatták.

Az újonnan javasolt nagyobb méretű interferométer-technológia bizonytalanságának köszönhetően a tudósok azt javasolták, hogy az első interferométer, a LIGO I, vagyis a korai (Initial) LIGO technikailag stabilabban, de régebbi elven működjön, ezért a tükröket egyszerű inga fázisban használták. Ekkor már Európában a többszörös inga elvén dolgoztak, így borítékolható volt a kisebb esély a gravitációs hullámok észlelésére. Ez keltette életre később a fejlettebb (Advanced) LIGO létrehozását.

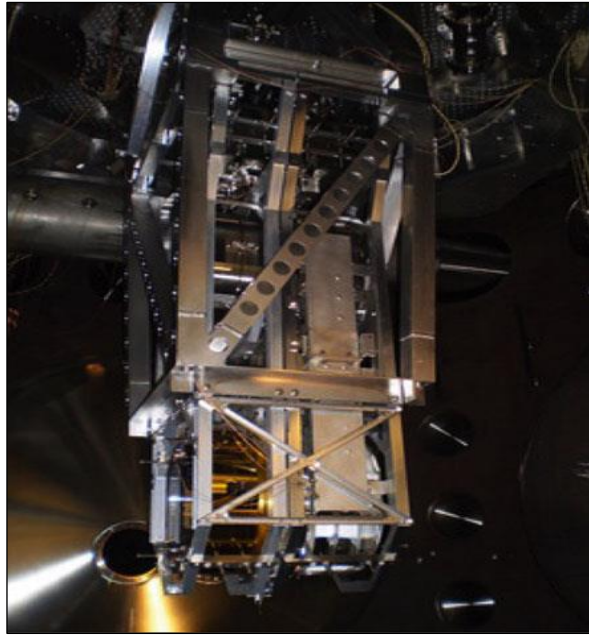
A LIGO I épületeinek és vákuumrendszereinek építése 1994-ben kezdődött. Alapját két, négy kilométer hosszú, rozsdamentes acélcső adja, ami egyben a maga közel 2,5 mérföldes hosszával a világ legnagyobb vákuumrendszerének tekinthető. A csövek átmérője 1,2 méter. A munkálatok csak 1998-ban fejeződtek be. A korai interferométer már tartalmazta a Fabry-Pérot rezonátorokkal továbbfejlesztett Michelson-interferométert, ami a korábban részletezett módon kevesebb fogyasztással, nagyobb energiák mozgósítására és tárolására volt képes.

3.1.2. A fejlesztett (Advanced) LIGO

Az Advanced LIGO projektet az amerikai National Science Foundation (NFS) 2008. áprilisában hagyta jóvá. Ez a két meglévő (Hanford, Livingstone) LIGO interferométer második generációra történő frissítését és egy harmadik (LIGO India) interferométer létrehozását irányozta elő.

Az Initial LIGO-hoz képest az Advanced LIGO számos technikai fejlesztést mutatott be:

- a lézer erősebb és sokkal stabilabb lett, 200 wattos fénykibocsátással;
- az inga felfüggesztése aktív szeizmikus elölszigetelést kapott;
- négyszeres ingákat alkalmaztak a vizsgálati tömegek szeizmikus elszigeteléséhez (4. ábra);
- nagyobbak és nehezebbek lettek a próbatestek (40 kg-osak), melyek üvegszálak felfüggesztést is kaptak elektrosztatikus aktuátorokkal és jel újrahasznosítással.



4. ábra: Az Advanced LIGO négyszeres ingaszerkezete [7]

3.2. Virgo

3.2.1. A korai (Initial) Virgo

Kezdetben a CNRS francia tudományos szervezet nem biztosított forrást Alain Brillet francia, és Adalberto Giazotto olasz tudósoknak egy interferométer létrehozására irányuló projekthez. 1989-ben azonban két másik olasz csoport, a Frascati és a Naples csatlakozott a Virgo projekthez. Új kérelmet nyújtottak be a CNRS-hez és az olasz tudományos szervezethez, az INFN-hez. Mindkét ügynökség végül 1993-ban, illetve 1994-ben hagyta jóvá a projektet. A toscanai Pisa melletti Cascinát választották a Virgo interferométer helyének (5. ábra). A névválasztás azt a reményt fejezte ki, hogy a Virgo interferométerrel meg lehet mérni a Virgo galaxishalmazból származó csillagászati objektumok gravitációs hullámait.

A LIGO-val ellentétben a Virgo nem egy hierarchikus felépítésű projekt, hanem európai és később nemzetközi tudományos együttműködési modellen alapszik. Ennek nyomán alapították meg az Európai Gravitációs Observatórium (EGO) konzorciumot 2000-ben.

A Virgo technikai alapjai egyeztek a LIGO-éval, azzal a különbséggel, hogy a Virgo tudósai a kezdetektől fogva a Giazotto által javasolt, a próbatetek számára kialakított szeizmikus szigetelőrendszerre támaszkodtak, amely hét kaszkád ingából állt, melyek együttesen nyolc méter magas tornyot alkottak. Erről a Super Attenuator (szuper csillapító) néven ismertté vált kialakításról feltételezték, hogy jobban javítja az érzékenységet alacsony frekvenciákon, mint bármelyik más interferométer.



5. ábra: Légi felvétel a Virgo létesítményről [8]

3.2.2. A fejlesztett (Advanced) Virgo

Az Advanced Virgo projektet 2009 végén hagyták jóvá, közel két évvel az Advanced LIGO jóváhagyása után. A továbbfejlesztett Virgo egymásra merőleges, 3 km hosszú karokkal, függesztett tükrökkel, valamint elődjéhez hasonlóan Fabry-Pérot rezonátorral rendelkezik. Annyi technikai változtatást eszközöltek, hogy az Advanced Virgo nem elektrosztatikus aktuátorokkal oldja meg a próbatestre történő erő kifejtést, hanem kis mágneseket ragasztottak a vizsgálati tömegekre, amelyek elektromágneses úton fejtik ki az erőt.

3.3. KAGRA (Kamioka Gravity and gravitational waves)

A KAGRA építését 2010-ben hagyták jóvá. Az új interferométer egy 3 km-es karokkal rendelkező földalatti detektor, amelyet Japánban, Kamioka bányájában építettek fel.

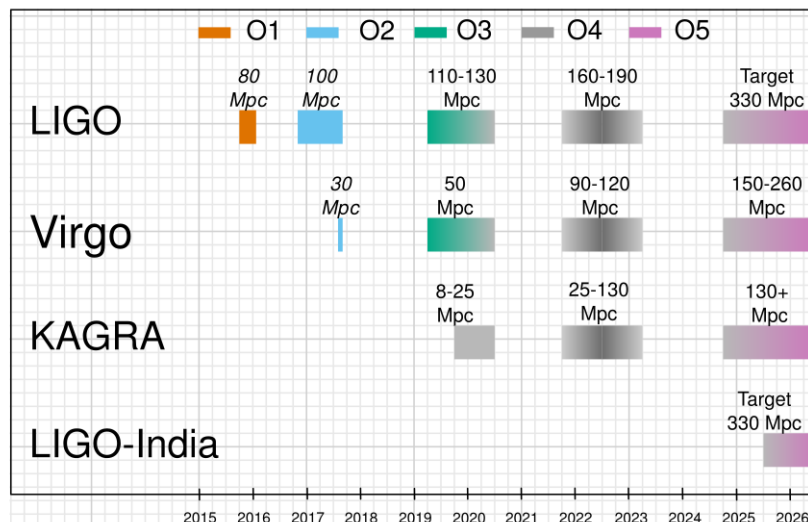
A végső célok között szerepel, hogy kriogén konfigurációban a KAGRA a próbatesteket 20 K-re hűtik le. Ezért a KAGRA előreláthatólag a harmadik generációs interferométerek számára tervezett két fontos új jellemzőjével rendelkezik: földalatti és kriogén. A zafír tükrök termikus zajának hűtéssel történő csökkentését a KAGRA projekt jóváhagyása előtt demonstrálták a CLIO-n, a kamiokai bánya 100 méteres prototípus interferométerén, amelyet a KAGRA technológiák tesztelésére használtak. Ily módon a termikus zaj csökkentése segít a tükrök vezérlésében, és megkönnyíti az interferométer érzékenységének növelését alacsony frekvenciákon. A KAGRA 2016-ban lett először (még korlátozottan) működésbe hozva, majd 2019-re befejezték minden optikai elem telepítését és a kriogén konfiguráció beállítását. Az interferométer a többszörös-inga felfüggesztést is megkapta. A műszaki átadást követően 2020. február 25-én kezdte meg működését obszervatóriumként.

4. Az obszervatóriumok megfigyelési időszakai

Az adatok felleléséhez, letöltéséhez és célorientált megkereséséhez tisztában kell lenni azzal, hogy hol találhatjuk meg, illetve mikortól érhetjük el őket.

Ezek az információk a Gravitational Wave Open Science Center (GWOSC) honlapján [11] érhetők el. Az alábbi felsorolásban összeszedtem az egyes mérési időszakokat. Ezek az adatok, illetve a 6. ábrán látható tervek a 2019-es Covid-19 járvány előtt készültek. Ennek megfelelően várhatóan késéssel kezdődik el az O4 időszak, illetve a KAGRA adatgyűjtése is. Az eddigi mérések időszakai az alábbiak szerint valósultak meg:

- **O3 időszak:** 2019. 04. 01. – 2020.03.27. (Szünet: 2019.10.01. – 10.31.)
Detektorok: H1, L1 és V1
- **O2 időszak:** 2016.11.30 – 2017.08.25.
Detektorok: H1, L1 és V1
- **O1 időszak:** 2015.09.12. – 2016.01.19.
Detektorok: H1 és L1
- **S6 időszak:** 2009.07.07. – 2010.10.20.
Detektorok: H1 és L1
- **S5 időszak:** 2005.11.04 – 2007.11.01.
Detektorok: H1, H2 és L1



6. ábra: A gravitációs hullámokat vizsgáló obszervatóriumok megfigyelési időszakai [12]

5. HDF fájlok

Ahhoz, hogy megérthessük a dolgozat célját, nevezetesen a HDF fájlok ismerete nélküli adatfeldolgozást, meg kell ismerni magát az adatstruktúrát. A HDF (Hierarchical Data Format) fájlokat tudományos adatok, mérések tárolására és közzétételére használják.

A HDF önmagát írja le, lehetővé téve az alkalmazás számára, hogy külső információk nélkül értelmezze a fájl felépítését és tartalmát. Egy HDF fájl tárolhatja a kapcsolódó objektumok keverékét, amelyek csoportként vagy egyedi objektumokként érhetők el. A felhasználók létrehozhatják saját csoportosító struktúrájukat. [2][3]

5.1. HDF4

A jelenleg használt HDF5 elődje korlátozottabban volt képes az adattárolásra, ennek ellenére mindmáig támogatott formátum, melyért a HDF Group nonprofit szervezet felel. A HDF4 támogatja a többdimenziós tömbök, raszter képek és táblázatok megjelenítését. Mindegyik egy speciálisan tömörített adattípust jelent, mely biztosít egy alkalmazásprogramozási interfészt (API-t) az adatok írására, olvasására, valamint az adatok és metaadatok rendszerezésére. [2]

A HDF4 korlátai a következők:

- hiányzik belőle egy átlátható objektum modell, ami a formátum támogatását és fejlesztését is megnehezíti;
- a számos különféle interfész stílus (képek, táblázatok, tömbök) komplex alkalmazásprogramozási interfészt alkotnak;
- többnyire csak előre definiált metaadatok kezelésére képes (kivéve az SD – Scientific Dataset – objektumokat);
- a címzési limithez 32 bites előjeles egészeket alkalmaz, ami maximálisan 2 GB-nyi adatot jelent. Ez tudományos kutatási eredmények rögzítésére és tárolására nem elegendő.

5.2. HDF5

A HDF5 fájlknál csoportokba, alcsoportokba rendezhetők az adatok, melyek között hiperhivatkozásokat hozhatunk létre. Könnyebben elképzelhető, ha egy kisebb könyvtárszerkezethez hasonlítjuk az adatstruktúrát, ahol az adatok szerteszét megtalálhatók.

A HDF5 fájlok belső szerkezete három fő egységből építhető fel:

- a) Adathalmazok, amelyek homogén típusú többdimenziós tömbök;

- b) Csoportok, amelyek olyan tárolószervezetek, amelyek tárolhatnak adathalmazokat és más csoportokat.
- c) Attribútumok, vagyis az adatkészletekhez csatolható metaadatok felhasználó által meghatározott bitjei.

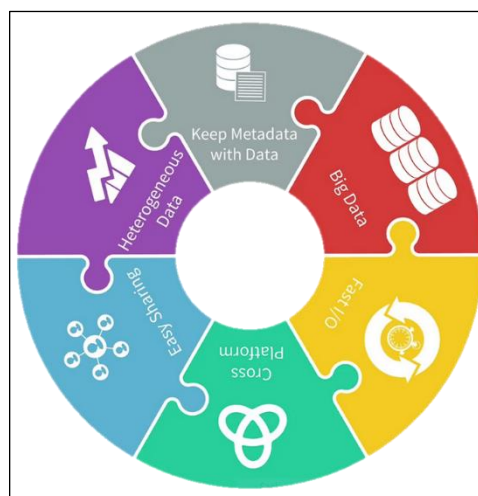
Az adattárolás többdimenziós adattáblákban történik. Ezeknek az alapvető egységeknek a felhasználásával a felhasználók sajátos alkalmazásformátumokat készíthetnek, amelyek az adatokat a problémás tartománynak megfelelő módszerrel rendezik. Nagyon gyakori, hogy a kutatók vagy más szervezetek megállapodnak egy „formátum a formátumban” módszerben, amely meghatározza, hogy a csoportok, adatkészletek és attribútumok milyen elrendezéssel használhatók az információk tárolására.

Mivel a HDF5 képes a platformokon átívelő adatok kezelésére, az adatok más csoportokkal történő megosztása mind a csoportok, az adatkészletek és az attribútumok esetében egyszerűen kezelhető. Mivel a fájlok önleírók, az alkalmazások formátumához még az alkalmazás formátumának ismerete sem szükséges.

Ezek alapján megállapítható, hogy a HDF5 egy ténylegesen hierarchikus, fájlrendszer-szerű adatformátum. [1][2]

5.2.1. Alkalmazási területek

A HDF5 jól működik olyan idősoros adatoknál, mint a részvényárfolyamok, a hálózati megfigyelési adatok és a 3D meteorológiai adatok. A 7. ábra alapján megállapítható, hogy a HDF5 jól használható olyan kutatásokhoz, melyeknél szükséges nem csak az adatok, de a metaadatok tárolása is. Kifejezetten jól alkalmazható az ún. Big Data kutatásokhoz, mely vizsgálat, kutatás során nagy mennyiségű adatot, adathalmazt kell rendezetten tárolni.



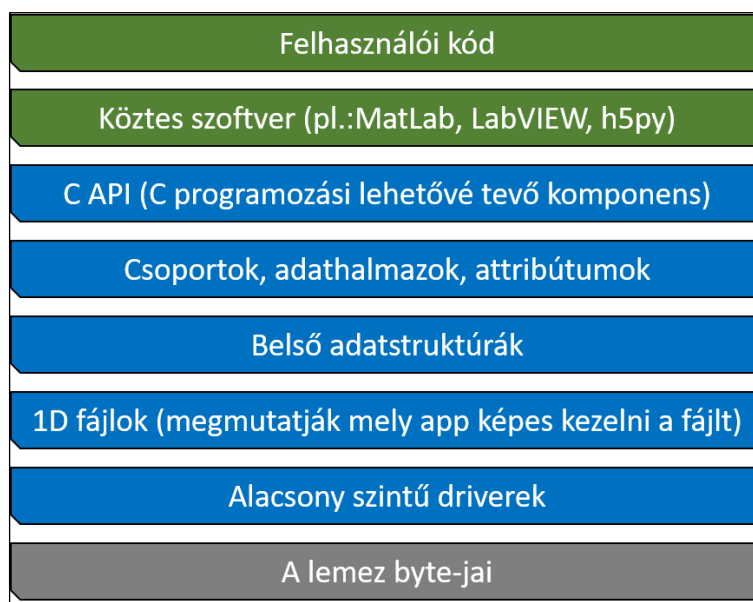
7. ábra: A HDF5 formátum előnyei [3]

Előnye, hogy standardizáltsága révén a tudományos élet által alkalmazott valamennyi ismertebb program képes kezelni, írni, olvasni ezeket az adatokat, kompaktsága pedig könnyen megoszthatóvá és terjeszthetővé teszi. Egyaránt alkalmazzák a fizika, csillagászat, meteorológia, pénzügy stb. területén és az iparban is.

5.2.2. A HDF5 szerkezeti felépítése

A 8. ábrán látható a HDF5 formátum használata során felmerülő különféle logikai rétegek rajza. A HDF5 könyvtár felépítése:

- a kékkel jelölt rész a HDF5 könyvtárban lévő összetevőket,
- a zölddel jelölt rész a „kliens” kódot jelenti, amely behívja a HDF5-öt,
- a szürkével jelölt rész az operációs rendszer által biztosított erőforrásokat jelenti.



8. ábra: A HDF5 könyvtár felépítése [2]

5.3. HDF View

A HDFView egy ingyenes grafikus megjelenítési elven működő fájlböngésző HDF5 fájlhoz, amelyet a HDF Group fejlesztett ki. Ez egy alapvetően egyszerű program, de Java nyelven íródott, így Windows, Linux és Mac rendszereken is elérhető megjelenítő, ami tökéletesen használható arra, hogy egy ismeretlen HDF5 fájlt tanulmányozhassak vele. A HDFView lehetővé teszi az adatkészletek és csoportok attribútumainak vizsgálatát is és szinte minden adattípust támogat, amelyet maga a HDF5 is támogat, egyes változó hosszúságú struktúrák kivételével. [16]

6. A beolvasást végző főprogram felépítése

6.1. Szoftverkörnyezet

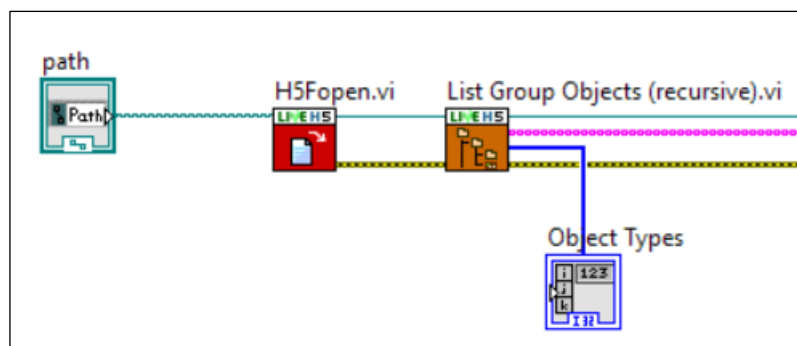
A program megalkotására a LabVIEW 2018 szoftverkörnyezetét használom. A készített VI (Virtual Instrument) megnyitásához szükséges a National Instruments által fejlesztett LabVIEW Professional 2018, vagy a LabVIEW egy régebbi verziója. Mivel a VI a HDF struktúrák kezelését végzi, így egy az ilyen adatállományokkal történő munkát segítő bővítmény telepítése is szükséges. Ez a bővítmény a Live HDF5 v1.1.1.86, vagy ennél régebbi verzió. A VI-ban fellelhető egyes alkalmazási példákhoz az Advanced Signal Processing Toolkit telepítése is feltétel.

6.2. HDF5 adatstruktúra kiolvasása

A gravitációs hullámok vizsgálata során gyűjtött adatokat HDF5 formátumban teszik közzé a gravitációs hullámokat vizsgáló obszervatóriumok. Ezeket a fájlokat a LabVIEW-ban a Live HDF5 segítségével lehet könnyebben kezelni.

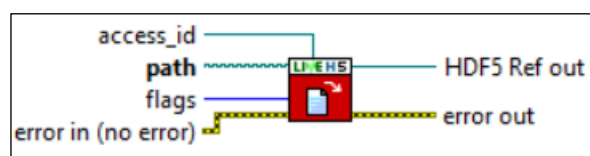
6.2.1. A letöltött HDF5 fájl megnyitása a szoftverkörnyezetben

A megnyitáshoz az 9. ábrán látható módon a „H5Fopen.vi”-t használtam, majd megadtam a bemeneti oldalon az elérési utat (path).



9. ábra: A HDF5 fájl megnyitása a szoftverkörnyezetben

A „H5Fopen.vi” ki- és bemenetei (10. ábra) közül az elérési utat (path), a referenciavezetékét (HDF5 Ref Out) és a hibavezetékét (error out) használtam fel, a többi vezeték felhasználása szükségtelen volt esetemben.



10. ábra: A „H5Fopen.vi” felépítése [13]

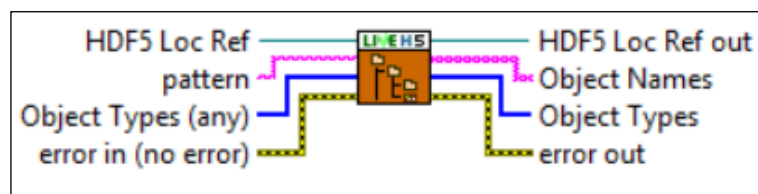
A fájl megnyitásához a Front Panelen található elérési út (path) rubrika melletti mappa ikonra kattintva (11. ábra) meg kell adni az olvasni kívánt fájl helyét. Ennek kitöltése feltétlenül szükséges a VI indítása előtt.



11. ábra: Az elérési út (path) megadása a Front Panelen

6.2.2. A csoportobjektumok rekurzív listázása az objektumok típusa szerint

A 9. ábrán, illetve a 15. ábrán bekötve látható „List Group Objects (recursive).vi” (12. ábra) arra szolgál, hogy ennek segítségével megismerhessem a HDF5 fájl belső könyvtáraiban található objektumok típusát.



12. ábra: A „List Group Objects (recursive).vi” felépítése [13]

A „.vi” bemenetére rákötöttem a referenciavezeték (HDF5 Loc Ref), illetve a hibavezeték (error in). A kimeneten az objektumtípusok (Object Types) megismerése céljából egy a Front Panelre kerülő (13. ábra) tömb megjelenítőt (Array Indicator) hoztam létre. Így sikerült feltárni a HDF5 fájl alkönyvtárainak típusait.

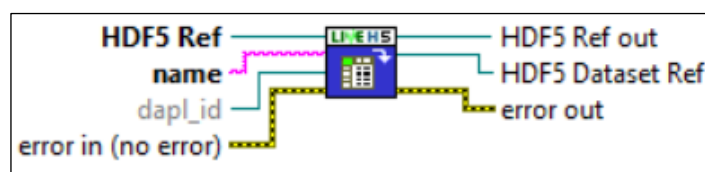
A „Object Names”-re kötött indikátor megmutatta a fentebb feltárt objektumtípusok konkrét nevét, melyek szintén a 13. ábrán láthatók. A „List Group Objects (recursive).vi” segítségével tehát meghatároztam és a felhasználó számára megjelenítettem a HDF5 fájl belső objektumainak nevét és típusát. A típus előrevetíti, hogy az adott objektum milyen adatállományt tartalmaz. Így megállapítható, hogy például a tömb utolsó előtti eleme – a „strain” objektum – csoportfájl (H5G_GROUP), tehát csak közvetve hordoz adatállományt (H5G_DATASET-et). Az adatállományt ezesetben a „strain/Strain” helyen találjuk.

Object Names	Object Types
meta/Description	H5G_GROUP 0
meta/DescriptionURL	H5G_DATASET 1
meta/Detector	H5G_DATASET 1
meta/Duration	H5G_DATASET 1
meta/GPSstart	H5G_DATASET 1
meta/Observatory	H5G_DATASET 1
meta/Type	H5G_DATASET 1
meta/UTCstart	H5G_DATASET 1
quality	H5G_GROUP 0
quality/detail	H5G_GROUP 0
quality/injections	H5G_GROUP 0
quality/injections/InjDescriptions	H5G_DATASET 1
quality/injections/InjShortnames	H5G_DATASET 1
quality/injections/Injmask	H5G_DATASET 1
quality/simple	H5G_GROUP 0
quality/simple/DQDescriptions	H5G_DATASET 1
quality/simple/DQShortnames	H5G_DATASET 1
quality/simple/DQmask	H5G_DATASET 1
strain	H5G_GROUP 0
strain/Strain	H5G_DATASET 1

13. ábra: A HDF5 fájl alkönyvtárainak feltárása objektumtípus kereséssel (Object Types), illetve az objektumok nevének listázása (Object Names)

6.2.3. A HDF5 adatállományának megnyitása

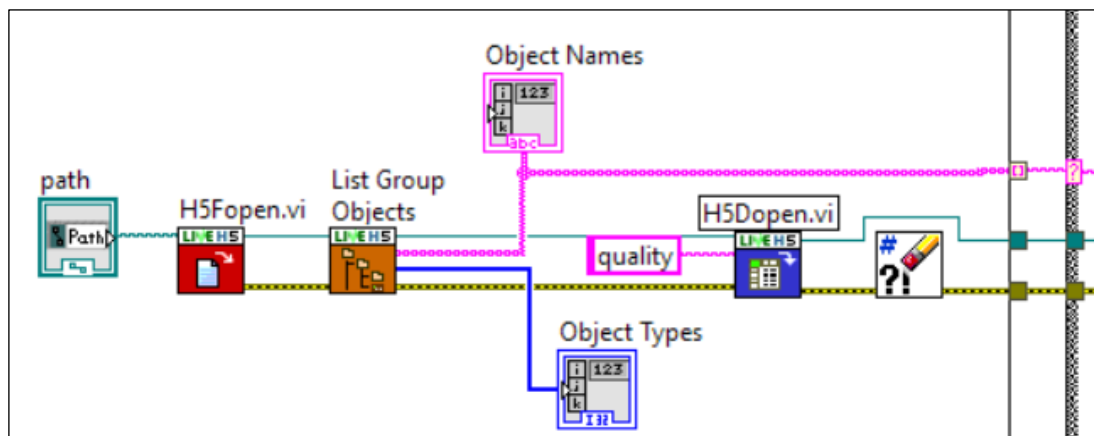
Azáltal, hogy meghatároztam a fentebbi két pont szerint a kiolvasandó adatállományok nevét és típusát, értelmet nyert ezek megnyitása és részletes feltárása a szoftverkörnyezetben. Ehhez a „H5Dopen.vi”-t használtam (14. ábra).



14. ábra: A „H5Dopen.vi” felépítése [13]

A „vi”-ra rákötöttem ezúttal is a referencia-, illetve hibavezetékét. A referenciavezetéseket eztán egy „for” ciklusra vezettem be. A 15. ábrán az eddig ismertetett „vi”-okkal együtt látható a programrészlet. A „name” bemeneten kötelező megadni annak az adathalmaznak a nevét, melyre aztán a „HDF5 Dataset Ref” adathalmaz referenciavezetéken hivatkozni szeretnék. Ezt a funkciót nem használtam ki, mert nem az adathalmaz egy könyvtárát, hanem az egész állományt kívántam megnyitni. Ennek ellenére

a futtatáshoz ez a „vi” megköveteli, hogy a „names” bemenetre legyen kötve egy szöveges konstans (string constant), így erre a „quality” mappát jelöltem ki.

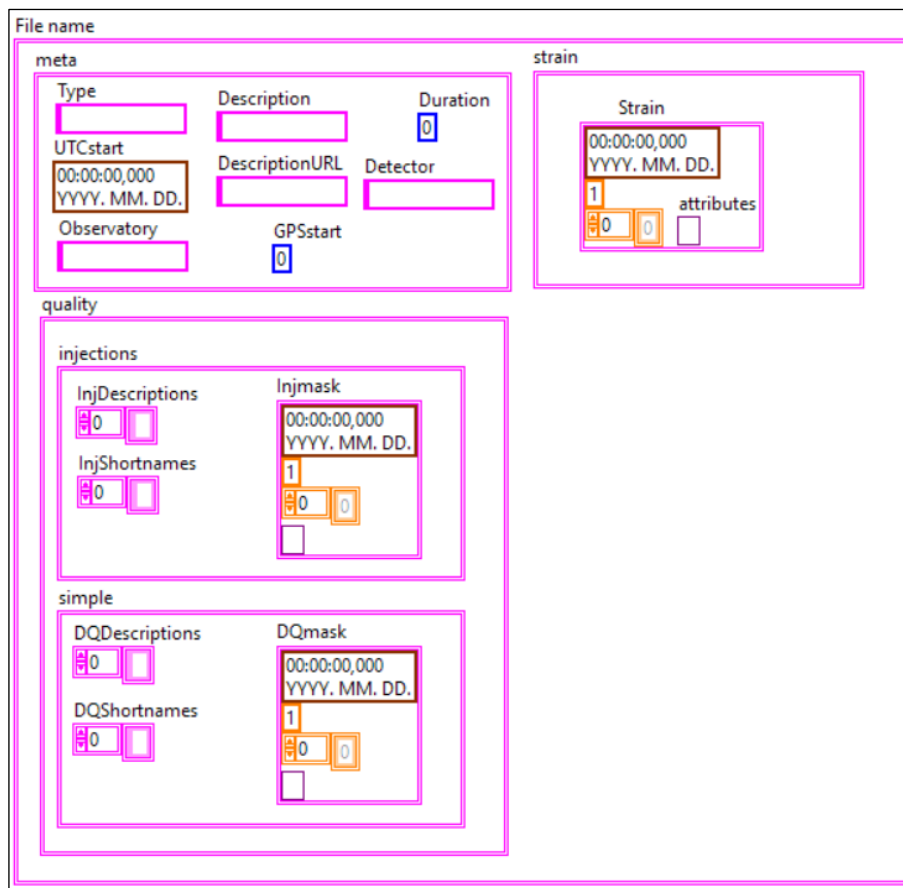


15. ábra: Programrészlet a „H5Dopen.vi”-hoz

6.3. Adatklaszter létrehozása az adattípusok kinyerése céljából

Minden egyes LIGO HDF5 fájl azonos könyvtárstruktúrával rendelkezik, mely szabványosítás megkönnyíti a LabVIEW környezetbe való implementálhatóságot is. Egy klaszterrendszer létrehozásával definiáltam az adattípusokat és azok neveit, melyet a 16. ábra szemléltet.

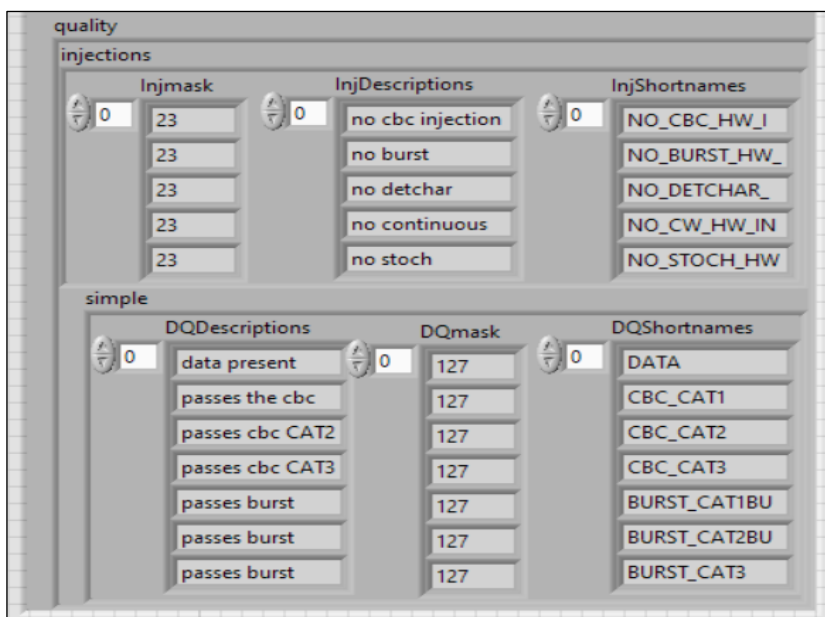
A „meta” klaszter tartalmazza a „Type”, „Description”, „DescriptionURL”, „Detector” és „Observatory” szövegkonstansokat (string constant), melyek sorra az adatállomány típusát, leírását, URL linkjét és a detektor, valamint az obszervatórium azonosítóját jelenítik meg. A „Duration” és a „GPSstart” numerikus konstansok pedig rendre a mintavételezés időtartamát és a mintavételezés kezdetének GPS idejét mutatják meg. A klaszter utolsó adata az „UTCstart”, vagyis a mintavételezés kezdetének ideje, amit időbélyeg formájába átalakítva nyertem ki, ehhez külön „subVI”-t készítettem, amit a későbbiekben fejtek ki.



16. ábra: Az adattípusokat definiáló adatklaszter rendszer a „Block Diagram”-on

A „quality” klaszter négy alklasztert tartalmaz. Az „injections” klaszter tartalmazza az „InjDescriptions” szöveg konstansok tömbjét (array of string constant), mely a mérés speciális körülményeit írja le, például, hogy az egyes injekciók aktívak voltak-e. A „InjShortnames” szöveg tömb ugyanezt az információt kódoltan tartalmazza. Az „Injmask” klaszter négy elemből áll. Tartalmaz egy időbélyeg konstansot, egy numerikus konstansot, egy numerikus konstansokat tartalmazó tömböt és attribútumokat. Utóbbiak a mért adatok leírását, bitszámát, X-Y tengelyek mértékegységét és címkéjét, a mintavételezési frekvenciáját, a nullpontot, illetve az adatrögzítés kezdetének idejét.

A „simple” klaszter a fentebbi analógia szerint ugyanúgy két adatsort és egy alklasztert tartalmaz, a „DQDescriptions” és a „DQShortnames” tömböket, valamint a „DQmask” klasztert. Ezek rendre tartalmazzák, hogy az adatállomány megfelel-e bizonyos sztenderdeknek, illetve ugyanezt az információt kódoltan is kinyerhetjük. Az „injections” és „simple” klaszterek adatai a Front Panelen jelennek meg a 17. ábrán látható módon. Itt olvasható ki, hogy az öt alkalmazott tulajdonság közül melyik volt aktív a vizsgálat időpontjában.

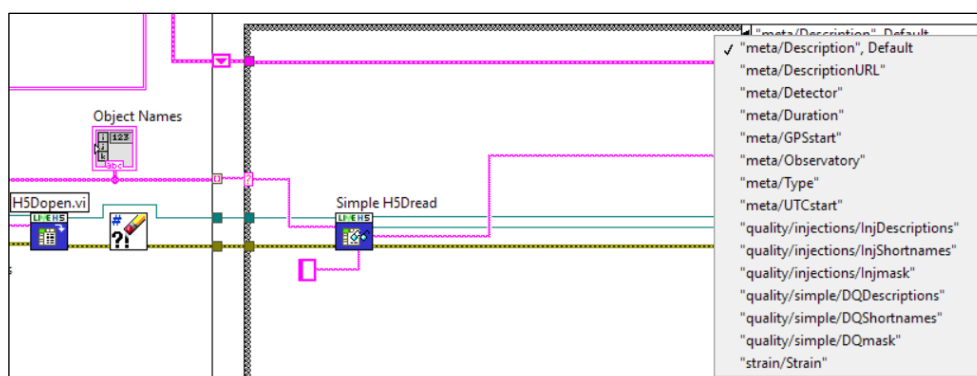


17. ábra: A Front Panelen megjelenített „injections” és „simple” klaszterek és azok adatai

Az utolsó klaszter, ami a legnagyobb adatfolyamot tartalmazza az a „strain” klaszter. Ebben találhatjuk az interferométer által rögzített gravitációs hullámok nyúlás időjelét („strain data”), ami már egy szűrt és mintavételezett jeltömb. A klaszter ebben az esetben is egy időbélyeget, az attribútumokat, egy numerikus konstanst és egy numerikus konstansokból álló tömböt tartalmaz.

6.4. A metaadatok kinyeréséhez használt struktúrák

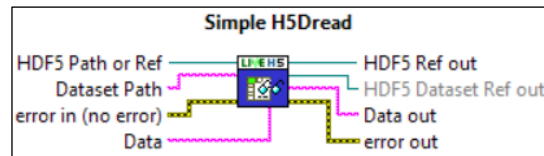
A 18. ábrán látható módon a fentebb kifejtett „vi”-ok mindegyikét egy „for” ciklusba és azon belül egy „case” struktúrába húztam be. Az esetválasztóba (case selector) az objektumok neveit kötöttem (Object Names). Így valamennyi könyvtár egy új esetként jelenik meg.



18. ábra: Az „for” ciklus és a „case structure” alkalmazása

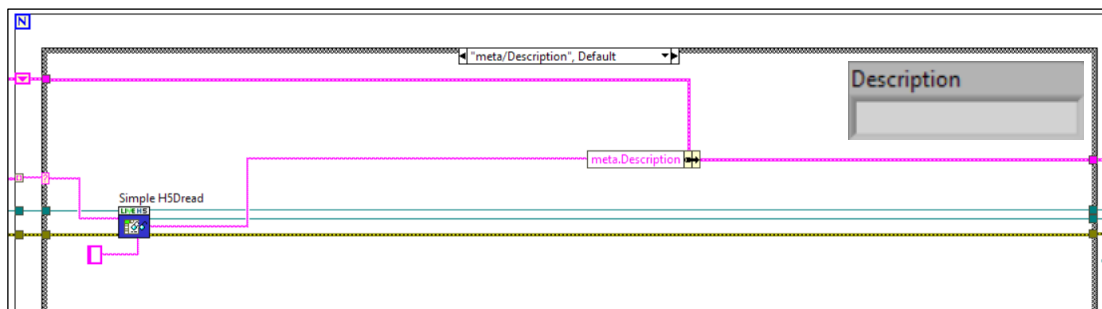
6.4.1. A „meta/Description” esete

A „case selector” első esete a metaadatok leírását olvassa ki (20. ábra). Ehhez a 19. ábrán részletezett „Simple H5Dread.vi”-t használtam.



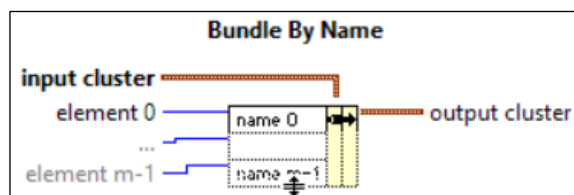
19. ábra: A „Simple H5Dread.vi” felépítése [13]

A referenciavezeték a bemeneti oldalon bekötöttem és a kimeneti oldalon tovább vezettem. Az adathalmaz vezetéket (Dataset path) az esetválasztóból húztam a „SimpleH5Dread.vi”-ba, így ebben az esetben a „meta/Description” adathalmazt olvassa csak ki a „vi”. Az adattípusnál (Data) szöveges konstansot (string constant) definiáltam, hisz a metaadatok leírása szöveges információ, illetve az adatklaszterben is eképpen definiáltam.



20. ábra: A „meta/Description” adathalmaz esete és a Front Panelen megjelenített adat

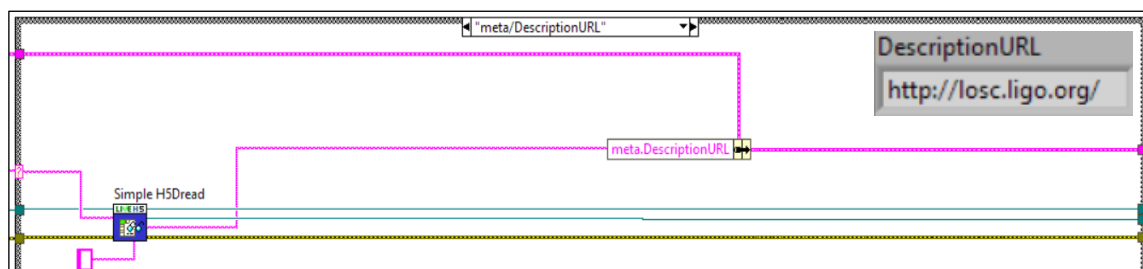
A kimeneti oldalon a kimenő adatot (Data out) egy név szerinti összefűzést végző, ún. „Bundle By Name.vi”-ba rendeztem (21. ábra), ahol ezesetben közvetlenül csak egy elem szerepel, még hozzá „meta/Description” adathalmaz. Ennek az összefűzésnek akkor látható az értelme, amikor már az összes kiolvasható esetben végrehajtottam a fentebbi műveletsort és valamennyi a klaszterkonstansban definiált adat konkrét értékke alakul a „for” ciklusból kimenő adattömbben. Az név szerinti összefűzés (Bundle By Name) bemenetére (input cluster), tehát a definiált klaszterkonstans kerül, kimenetén (output cluster) pedig az egzakt adatokat tartalmazó klaszter fogja elhagyni az „case” struktúrát. Így valamennyi esetenél egyenként, de strukturáltan összefűzve kaphatók meg az adatok.



21. ábra: A „Bundle By Name.vi” felépítése [13]

6.4.2. A „meta/DescriptionURL” esete

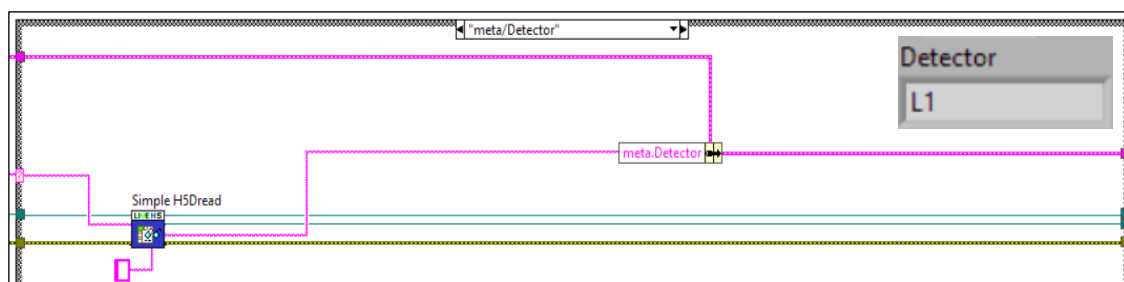
Az esetválasztó második eleme a metaadatok leírásának URL-jét jeleníti meg. Ebben az esetben ugyanazokat a beállításokat végeztem el, mint a 6.4.0-ás fejezetben, hiszen továbbra is szöveges adatot kellett kinyernem. A kinyert adat a „http://losc.ligo.org/” webhelyet közli. A „meta/DescriptionURL”-t a 22. ábra mutatja be.



22. ábra: A „meta/DescriptionURL” adathalmaz esete és a Front Panel adata

6.4.3. A „meta/Detector” esete

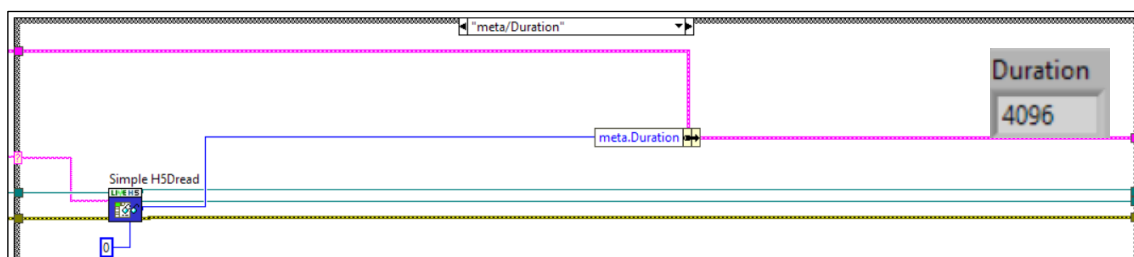
Az esetválasztó harmadik eleme a detektor számát jeleníti meg. Ebben az esetben is ugyanazokat a beállításokat végeztem el, mint a 6.4.0-ás fejezetben, hiszen továbbra is alfanumerikus adatot kellett kinyernem. A „meta/Detector”-t a 23. ábra mutatja be.



23. ábra: A „meta/Detector” adathalmaz esete és a Front Panelen megjelenített adat

6.4.4. A „meta/Duration” esete

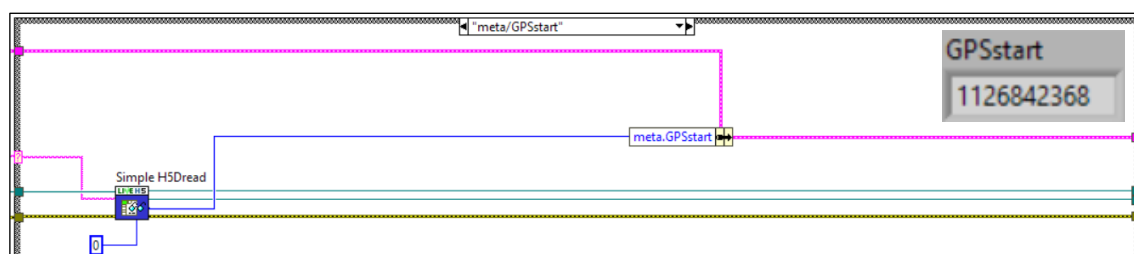
A „case selector” negyedik tagja a mintavételezés idejét jeleníti meg numerikusan. Ennek megfelelően a fentebbi beállítások csak a „SimpleH5Dread.vi” adat (Data) bemenetén változnak. Ebben az esetben egy numerikus konstanst kötöttem ide. A „meta/Duration”-t a 24. ábra mutatja be.



24. ábra: A „meta/Duration” adathalmaz esete és a Front Panelen megjelenített adat

6.4.5. A „meta/GPSstart” esete

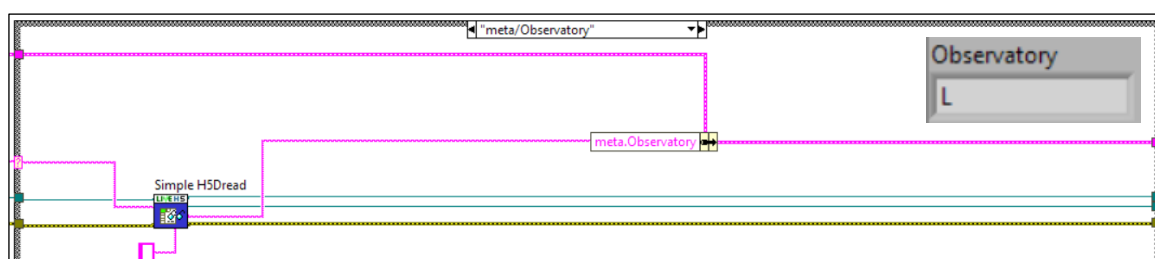
Az esetválasztó ötödik eleme a mintavételezés kezdetének időpontját jeleníti meg a GPS idő formátumában. Ez az eset is ugyanazokat a beállításokat tartalmazza, mint a 6.4.4-es fejezet, mert ezúttal is numerikus adatot kellett kinyernem. A „meta/GPSstart”-ot a 25. ábra mutatja be.



25. ábra: A „meta/GPSstart” adathalmaz esete és a Front Panelen megjelenített adat

6.4.6. A „meta/Observatory” esete

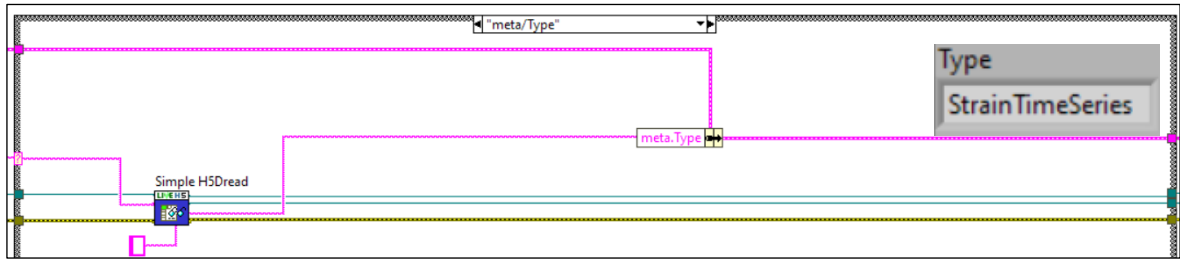
A „case selector” hatodik tagja az obszervatórium jelét mutatja meg. Továbbra is alkalmazom a fentebbi fejezetek konfigurációját. Ebben az esetben a „SimpleH5Dread.vi” adat (Data) bemenetére szöveges konstansot adtam meg. A „meta/Observatory” „block diagram”-on látható struktúráját a 26. ábra mutatja meg.



26. ábra: A „meta/Observatory” adathalmaz esete és a Front Panelen megjelenített adat

6.4.7. A „meta/Type” esete

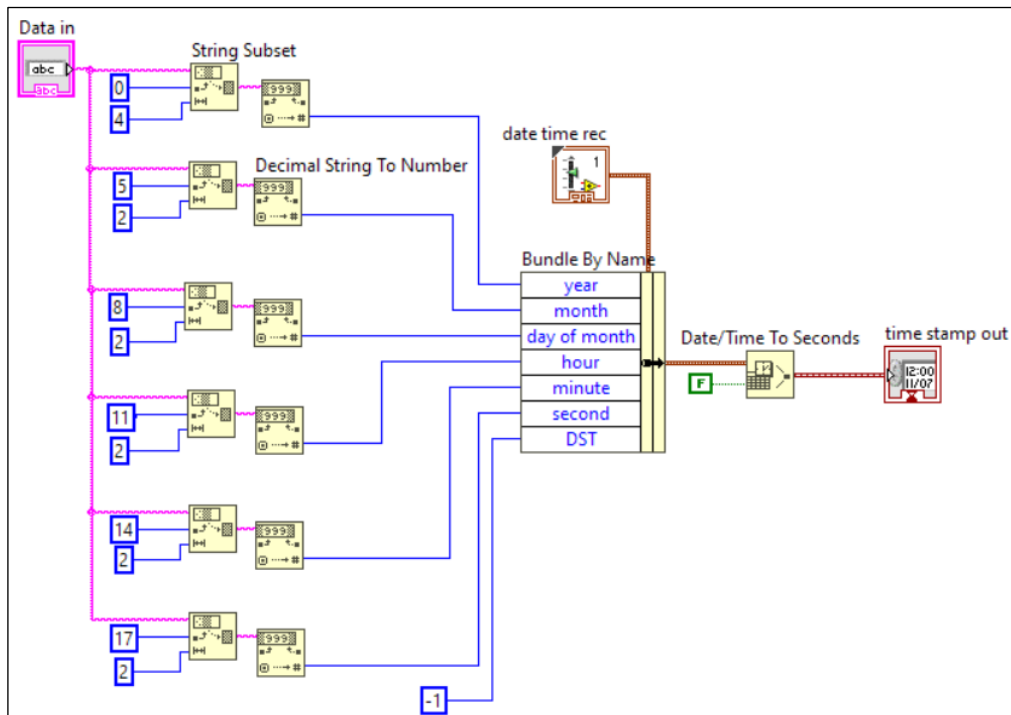
Az esetválasztó hetedik eleme az adat típusának (Type) leírását jeleníti. Ez az eset is ugyanazokat a beállításokat tartalmazza, mint a 6.4.6-os fejezet, mert ezúttal is szöveges adatot kellett kinyernem. A „meta/Type”-ot és a Front Panel értékét a 27. ábra mutatja be.



27. ábra: A „meta/Type” adathalmaz esete és a Front Panelen megjelenített adat

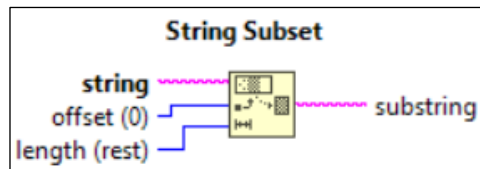
6.4.8. A „meta/UTCstart” esete és az időbélyeg konverziós „subVI”

A „case selector” nyolcadik eleme az egyezményes koordinált világidő (UTC) formátuma szerint adja meg a mintavételezés kezdetét. Továbbra is alkalmazom a fentebbi fejezetek konfigurációját. Ebben az esetben a „SimpleH5Dread.vi” adat (Data) bemenetére szöveges konstansot adtam meg. A „Bundle By Name”-re időbélyeg formátumot készítettem melyet a 28. ábrán keresztül mutatok be.



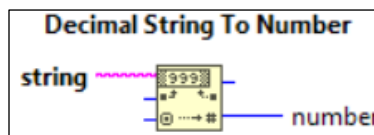
28. ábra: Az időbélyeg konverziót „subVI”-ként hoztam létre

A „Data in” egy szöveges bemenet, ami a „SimpleH5Dread.vi” bemenetével egyezik meg. Előbbi „vi” kiolvassa ugyan az időt, de nem időbélyeg formátumban adja ki az értéket. Ezért az egyes információtrészeket a „String Subset.vi”-jal, a megfelelő indexeléssel nyerem ki a 29. ábrán látható felépítés szerint.



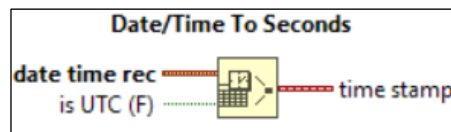
29. ábra: A „String Subset.vi” felépítése [13]

Az így kapott szám még nem numerikus formátumú, ehhez az átalakításhoz a „Decimal String To Number.vi”-t használom, melynek részletei a 30. ábrán láthatók. Ezeket a most már numerikus formátumú adatokat név szerint fűzöm össze egy „date time rec” formátumú klaszterbe.



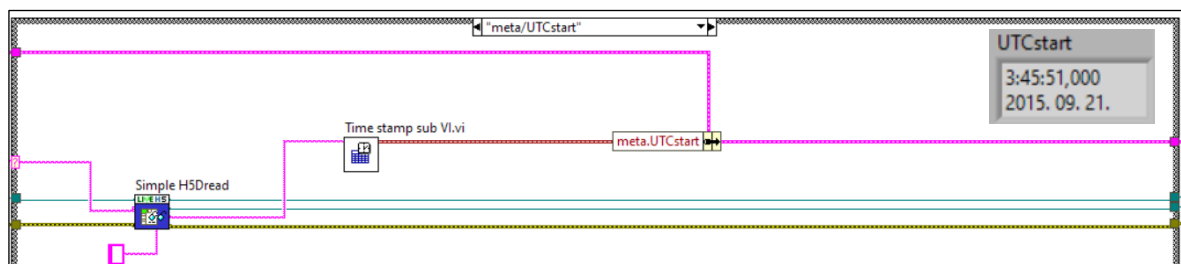
30. ábra: A „Decimal String To Number.vi” felépítése [13]

A „date time rec” formátumot utolsó lépésben időbélyeg formátumává alakítom át a 31. ábrán látható „Date/Time To Seconds.vi” segítségével.



31. ábra: A „Date/Time To Seconds.vi” felépítése [13]

Az így kapott időbélyeg már alkalmas további feladatok elvégzésére, illetve kutatási célú jelfeldolgozó és méréstechnikai megoldásokhoz. A „case” struktúrában elhelyezett „subVI”, és a teljes „meta/UTCstart” eset a 32. ábrán tekinthető meg.

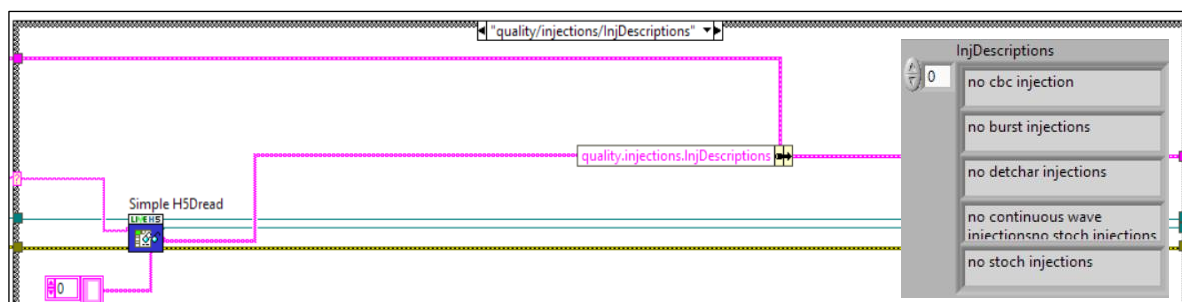


32. ábra: A „meta/UTCstart” adathalmaz esete és a Front Panelen megjelenített adat

6.5. A „quality” könyvtár adatainak és maszkjainak kinyerése és az adatok jelentésének magyarázata

6.5.1. A „quality/injections/InjDescriptions” esete

Ezzel az esettel már egy másik alkönyvtár fájljaihoz értünk, ugyanis a „quality” könyvtár elemei következnek. A konfiguráció a fentebbiekhez hasonló, de itt a „SimpleH5Dread.vi” egy szövegkonstans tömböt (string array) kap, mert ennél az esetnél a tömbben öt elem van. Az injekciók azt jelentik, hogy mesterségesen előállított jelet kevernek rá a mért jelre. Ezeket a rendszer kalibrációjának ellenőrzésére használják.



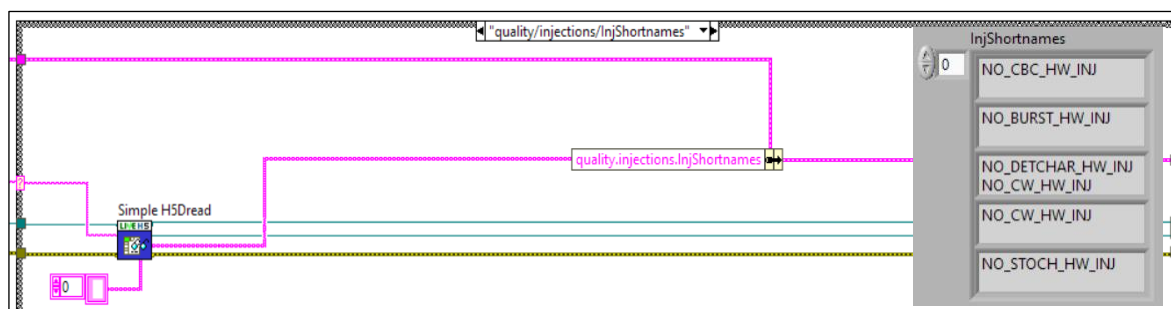
33. ábra: A „quality/injections/InjDescriptions” esete és a Front Panel adata

A kapott elemek jelentései a következők: [14]

- „no cbc injection”: Kompakt bináris koaleszcenciás (CBC) injekciók kerültek hozzáadásra a tervezett vizsgálat során.
- „no burst injections”: „Burst” típusú injekciók jelenlétét mutatja meg.
- „no detach injections”: Megmutatja a detektor karakterisztikus injekcióinak jelenlétét.
- „no continuous wave injections”: Folyamatos hullámú (CW) injekciókat (más néven pulzárinjekciókat) az egész O2 időszak tartalmazott. Minden CW befecskendezés megfelel egy olyan formájú gravitációs hullámjel szimulációjának, amely egy gyorsan forgó, elszigetelt neutroncsillagtól várható (gyakran rádiós pulzusként figyelhető meg).
- „no stoch injections”: Megmutatja, hogy jelen voltak-e sztochasztikus injekciók.

6.5.2. A „quality/injections/InjShortnames” esete

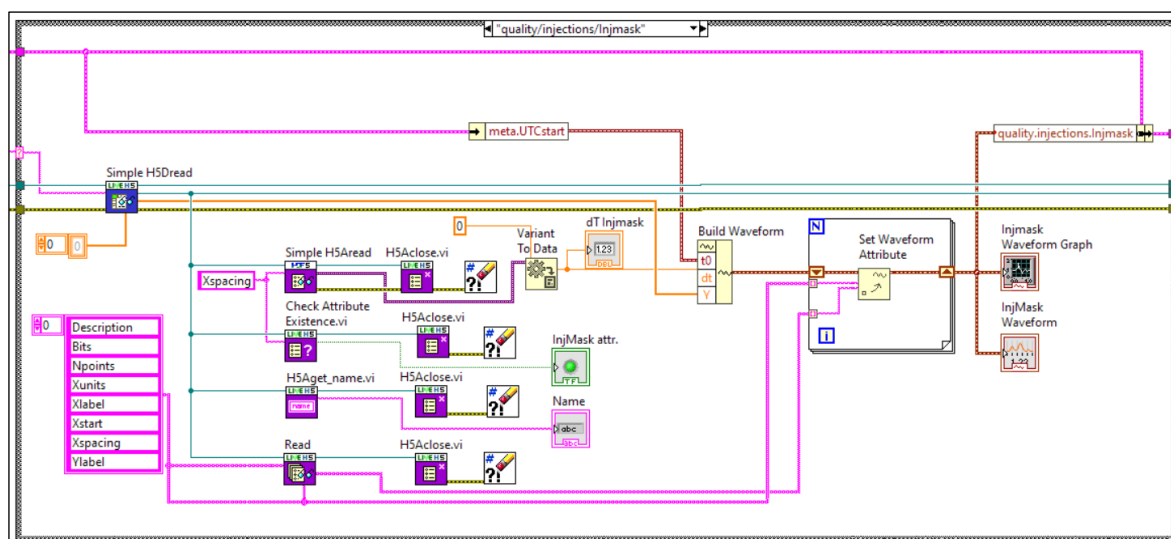
Ez az eset tartalmi szempontból szinte, konfiguráció szempontjából viszont teljesen megegyezik a fentebbi esettel, annyi különbséggel, hogy itt a 6.5.1.-es fejezetben látható tulajdonságok rövidítései láthatók a Front Panelen, amit a 34. ábra szemléltet.



34. ábra: A „quality/injections/InjShortnames” adathalmaz esete és a Front Panelen megjelenített adat

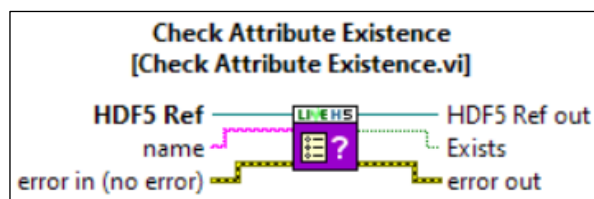
6.5.3. A „quality/injections/Injmask” esete

Az Injmask esete (35. ábra) egy olyan maszkot takar, ami a hagyományos úton történő teljes adatkinyerést nem teszi lehetővé csupán a „Simple H5Dread.vi” alkalmazásával.

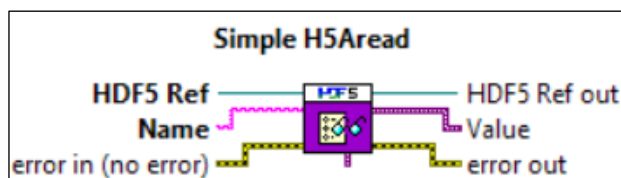


35. ábra: A „quality/injections/Injmask” adathalmaz esete

Ennek az az oka, hogy a maszk attribútumokat is tartalmaz, amiket eltérő módszerrel kell kiolvasnom. Az attribútumok tartalmát ezúttal is először a „HDFView”-val ismertem meg. Először a „Check Attribute Existence.vi”-t (36. ábra) használtam arra, hogy ellenőrizzem, hogy valóban létezik-e a fájlban egy ilyen attribútum. Miután megbizonyosodtam róla, hogy igen, a „Simple H5Aread.vi”-t (37. ábra) alkalmaztam arra, hogy az X tengelyi osztásköz értékét kinyerhessem. Ezt az „Xspacing” adatot azért kell kinyernem, hogy később ábrázolni tudjam az attribútum X tengelyén vett adatait, ehhez szükséges ezt az osztásközt megismernem.

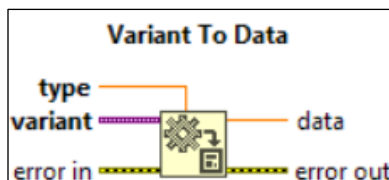


36. ábra: A „Check Attribute Existence.vi” felépítése [13]



37. ábra: A „Simple H5Aread.vi” felépítése [13]

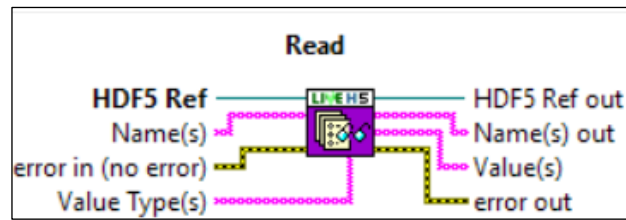
Mivel az attribútum kiolvasása nem konkrét adatot, csak egy „Variant” tömörített információt eredményezett, így ezt a „Variant to Data.vi” (38. ábra) segítségével konvertáltam kézzelfogható adatra. Ehhez a bemeneti oldalra kötöttem a „Simple H5Aread.vi” kimenetét, valamint megadtam, hogy numerikus konstans adatot várok a kimeneten. Az így kapott értékeket egy hullámforma dt értékébe kötöttem, mert a hullámforma X tengelyi osztásköze lesz ez adat. Ezt a „Build Waveform.vi” alkalmazásával értem el.



38. ábra: A „Variant to Data.vi” felépítése [13]

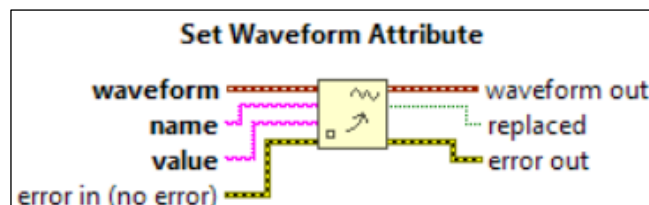
Ez a „vi” tartalmaz még két bemenetet, a t0 és az Y változókat. A t0 értékének helyére a „meta.UTCstart” időbélyeget, vagyis a mintavételezés kezdetének időpontját húztam. Ezt a rózsaszín vezeték adatsomagjából nyertem ki egy „Unbundle by Name.vi”-jal. Az Y bemenetre pedig a közvetlenül a „SimpleH5Dread.vi” által kiolvasott numerikus adattömböt kötöttem.

Az attribútum tartalmaz ezeken kívül még számos adatot, részben olyanokat is, amiket már korábban hagyományosabb úton kinyertem, de ezúttal minden lehetséges információt feltárok, hogy később az algoritmust használóknak rendelkezésre álljon. A „Read.vi” „Name(s)” bemenetére kötöttem egy szövegkonstans tömböt, valamint a „Value Type(s)”, vagyis az értékek típusát definiáló bemenetre is ugyanezt húztam. (39. ábra)



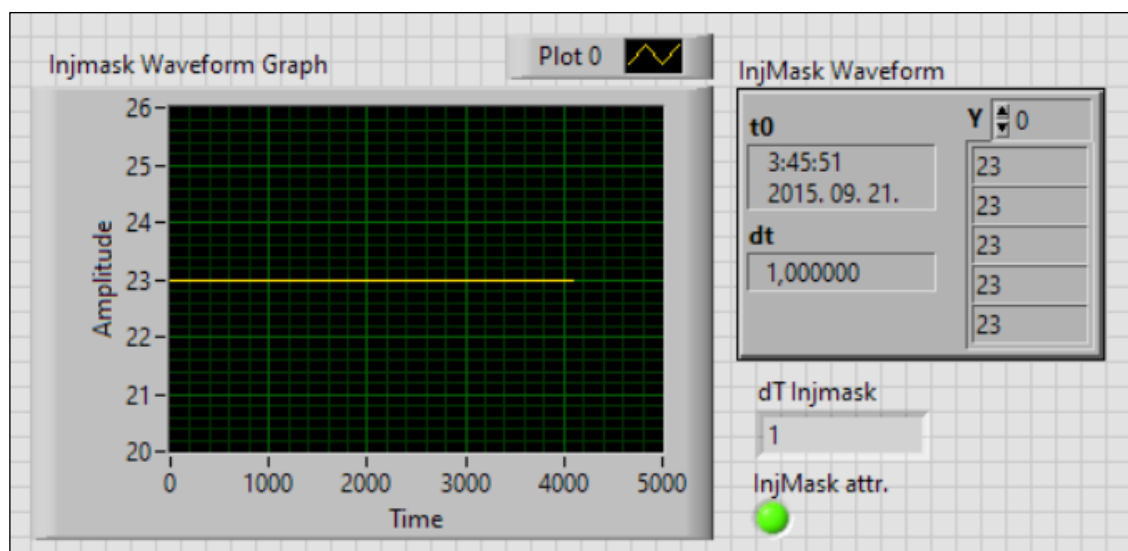
39. ábra: A „Read.vi” felépítése [13]

A „Read.vi” kimenetét egy „for” ciklusra kötöttem, ahol indexelésre állítottam be a ciklusba bemenő adatokat. A „Set Waveform Attribute.vi” (40. ábra) rendszerezni képes az indexelt értékeket a megfelelő típusokkal, és ezekkel feltölti az így generált kimeneten kapott hullámformát.



40. ábra: A „Set Waveform Attribute.vi” felépítése [13]

Utóbbit egy „Shift Register”-rel oldottam meg, szintén a megfelelő indexelés miatt. Az így kapott adathalmazt a „Front Panel”-en a 41. ábrán látható gráf mutatja meg. Ennek a gráfnak az adatait az előző esetekben ismertetett és megindokolt módon egy „Bundle by Name.vi”-ba kötöttem.



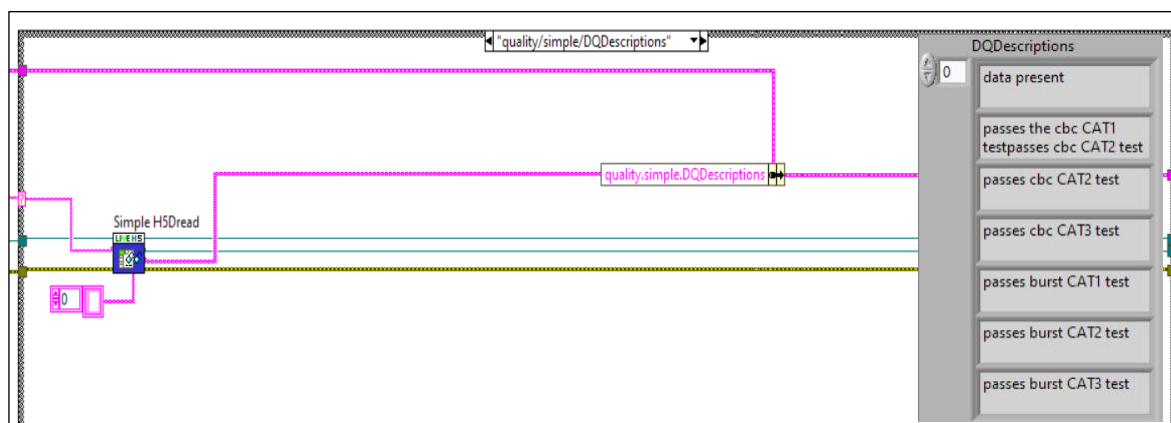
41. ábra: Az „Injmask” attribútum adatiból készített gráf a Front Panelen

A 42. ábra alapján felmerül a kérdés, hogy mit jelentenek az adatok. A konstans 23-as érték ebben a decimális számrendszerben nem sokatmondó. Ha azonban binárisan értelmezem a számot, akkor 5 bitet kapok: $23_{10} = 10111_2$

Az így kapott számjegyek a 6.5.1-es fejezetben bemutatott tulajdonságok állapotát jelzik. Ennek megfelelően ez a konkrét adatsomag, ami 2015. 09. 21-én lett rögzítve, ez a „burst” injekciókon kívül minden más injekció aktivitását mutatják. Természetesen más adatsomagot beolvassva más értéket kaphatunk.

6.5.4. A „quality/simple/DQDescriptions” esete

Ez az eset a DQ maszk elemeinek leírását tartalmazza, megegyező analógia mentén, mint a 6.5.1-es fejezetben.



42. ábra: A „quality/simple/DQDescriptions” esete és a Front Panel adata

Ezúttal is szövegkonstans-tömböket nyertem ki a maszkból, de ebben az esetben 7 tulajdonság jelent meg:

- Data present
- Passes the CBC CAT1 test, passes the CBC CAT2 test
- Passes CBC CAT2 test
- Passes CBC CAT3 test
- Passes burst CAT1 test
- Passes burst CAT2 test
- Passes burst CAT3 test

A fentiek jelentése az alábbi logika szerint alakul: [15]

- Data: Ennek a szintnek a hiánya azt jelzi, hogy a gravitációs hullámok törzsadatai nem nyilvánosak, mert a műszerek nem működtek elfogadható állapotban.

- CAT1: Az adatminőség-ellenőrzés sikertelensége ebben a kategóriában kritikus problémát jelez. Arra utal, hogy egy kulcsfontosságú összetevője a detektornak nem működik az előírt konfigurációnak megfelelően. Mivel ezek az idők jelentős ismert problémát jeleznek, így minden adatelemző csoportban azonosak. A CAT1 (Category 1) sikertelen teljesülése esetén az idők és a megfelelő adatok nem állnak rendelkezésre.
- CAT2: A sikertelenség (nullás bit) ebben a kategóriában azokat az idő adatokat jelzi, amikor ismert, feltérképezett és megértett fizikai kapcsolódás van a gravitációs hullámcsatornához. Ez magában foglalhatja a magas szeizmikus aktivitás idejét is.
- CAT3: Az adatminőség-ellenőrzés sikertelensége ebben a kategóriában azokat az idő adatokat jelzi, amikor a gravitációs hullámcsatornához statisztikai jellegű kapcsolódás van, aminek szabályszerűsége nem teljesen feltérképezett és megértett a tudományos világban.

Ezek alapján már csak annyi különbség adódik a hét tulajdonság között, hogy pontosan milyen injekciót értékeltünk: kompakt bináris koaleszcenciát (CBC), „burst”-öt, folyamatos hullámokat (CW) vagy sztochasztikus injekciókat. A külön-külön történő értékelés oka az, hogy a zajos adatok időszakai eltérő módon befolyásolják az egyes elemzési típusokat.

6.5.5. A „quality/simple/DQShortnames” esete

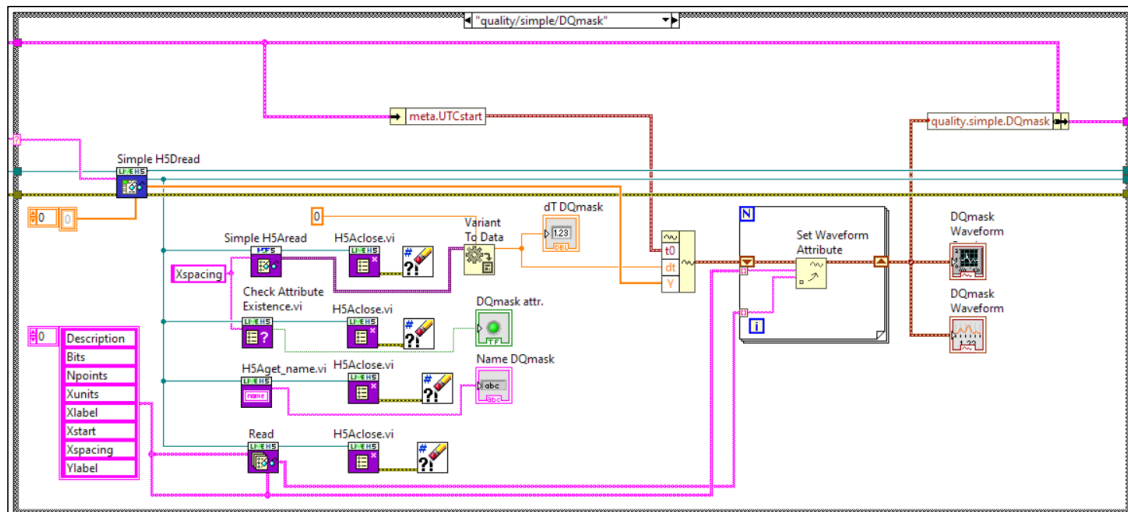
A „DQShortnames” esete ezúttal is az előző fejezet, a „DQDescriptions” rövidítéseit tartalmazza. A 43. ábrán látható a Block Diagramon a program utasításai és a Front Panelen megjelenített adatok.



43. ábra: A „quality/simple/DQShortnames” esete és a Front Panelen megjelenített adat

6.5.6. A „quality/simple/DQmask” esete

A „DQmask” esetének Block Diagramja szinte megegyezően épül fel, mint az „Injmask” programja, ezért nem részletezem újra a program egyes paneleit, a felhasznált „vi”-ok megegyeznek a 6.5.3-as fejezetben ismertetettekkel. A 44. ábra mutatja meg ennek az esetnek a Block Diagramját.

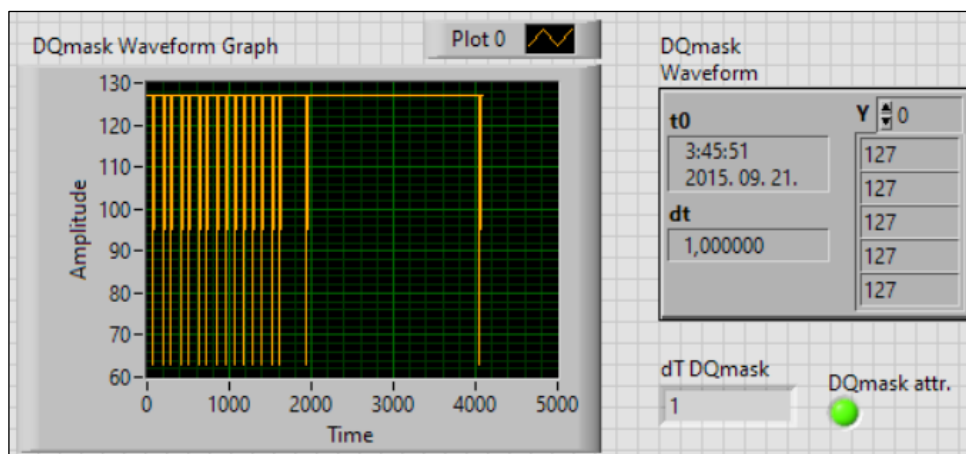


44. ábra: A „quality/simple/DQmask” adathalmaz esete

A VI futtatását követően a Front Panelen a 45. ábrán látható adatokat nyertem ki. A 6.5.4-es fejezetben már említett információkat egy „Waveform Graph” segítségével jelenítettem meg. A grafikonon három szám váltakozása látható, ezek a 127, a 95 és a 63. Az injekciók maszkjánál már megállapítottam, hogy ezeknek a decimális számoknak binárisan kell keresni az értelmét. Az átváltás a következők szerint alakul:

- $127_{10} = 1111\ 111_2$
- $95_{10} = 1011\ 111_2$
- $63_{10} = 0111\ 111_2$

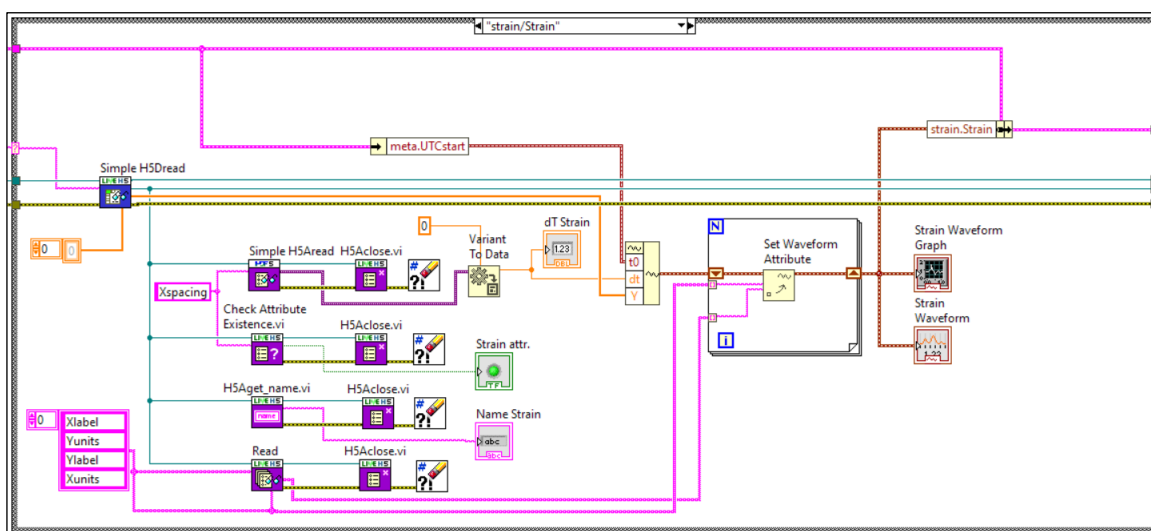
A bináris számok hét bitből állnak, melyek a 6.5.4-es fejezetben kifejtett hét tulajdonság értékére utalnak. A GWOSC adatfájlok minden másodpercéhez tartozik egy maszk érték, amelyeknél a jó adatokat 1-essel jelzik, a meghíúsult időket (tehát a rossz adatokat) pedig nullával jelölik. Az adatok értéke ebben az esetben is azért lett decimálisan tárolva, hogy tárhelyet csökkentsenek a kutatók és ezáltal még tömörebb lehessen az állomány.



45. ábra: A „DQmask” attribútum adatiból készített gráf a Front Panelen

6.6. A „strain/Strain” nyúlás időjel adatok kinyerése

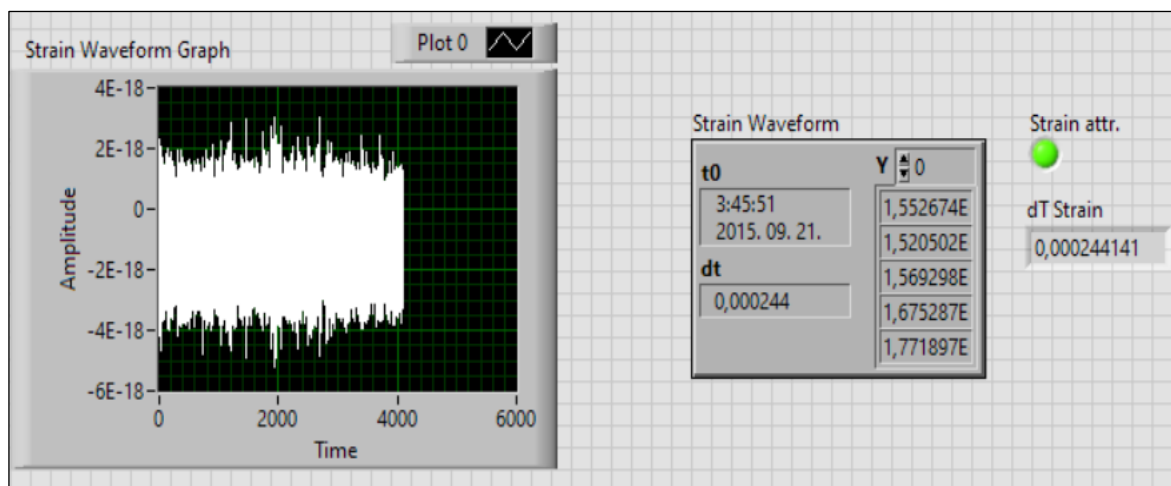
Az adatállomány egyik legfontosabb eleme a „strain” nyúlás időjel állomány, melynek Block Diagramja a 46. ábrán látható. A beolvasás a fentebbi fejezetekben bemutatott maszkokéval megegyező, de a kapott értékek és a generált „Waveform Graph” eltérő.



46. ábra: A „strain/Strain” adathalmaz esete

A 47. ábrán látható az adatsokaság, amit nyúlás időjel esetéből nyertem ki. A beolvasott állomány egy 4 kHz-es adatcsomag, de 16 kHz-es adatokkal is működőképes. Ennek megfelelően az X tengelyen az eltelt időt látjuk, míg az Y tengelyen a hullámjel ingadozása olvasható le, ami igen alacsony értékek mentén, 10^{-18} -adikonos nagyságrendben szerepel.

Az X tengely osztásköze a frekvencia adatból nyerhető ki, 4 kHz esetén ez 0,000244 másodperc nagyságú, ekkora időközönként jelenik meg újabb „strain” komponens a grafikonon.

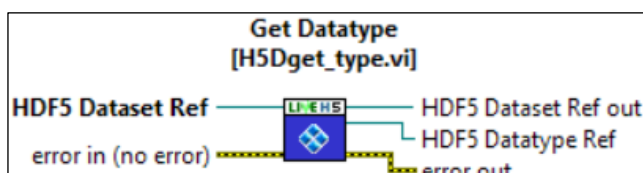


47. ábra: A „strain/Strain” attribútum adatiból készített gráf a Front Panelen

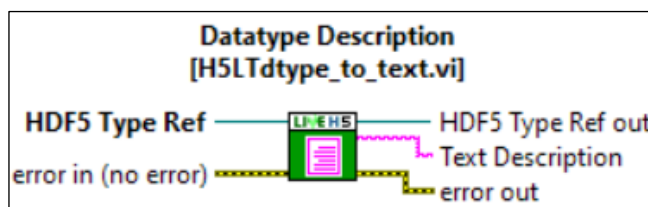
6.7. Az adattípus megállapítása az adatkészlet elemeinél

A klaszterkonstans bemutatásakor a 6.3-as fejezetben megállapítottam, hogy minden LIGO obszervatórium által publikált adat azonos struktúrájú, így elégséges konstansként definiálni az egyes adattípusokat. Ezt a felfedezést az előzte meg, hogy mélyebben kielemeztem a LabVIEW-ban kiolvasott adatok típusait speciális „.vi”-ok segítségével.

Minden a fentebb bemutatott esetről a „SimpleH5Dread.vi” „Dataset Reference” kimenetét kiveztem az esztvlasztóból, majd bekötöttem a „H5Dget_type.vi”-ba (48. ábra). Ez arra szolgál, hogy létrehozza a „HDF5 Datatype Ref” kimenetet, ami több „Datatype”-típusú „.vi” bemenetéhez elengedhetetlen.



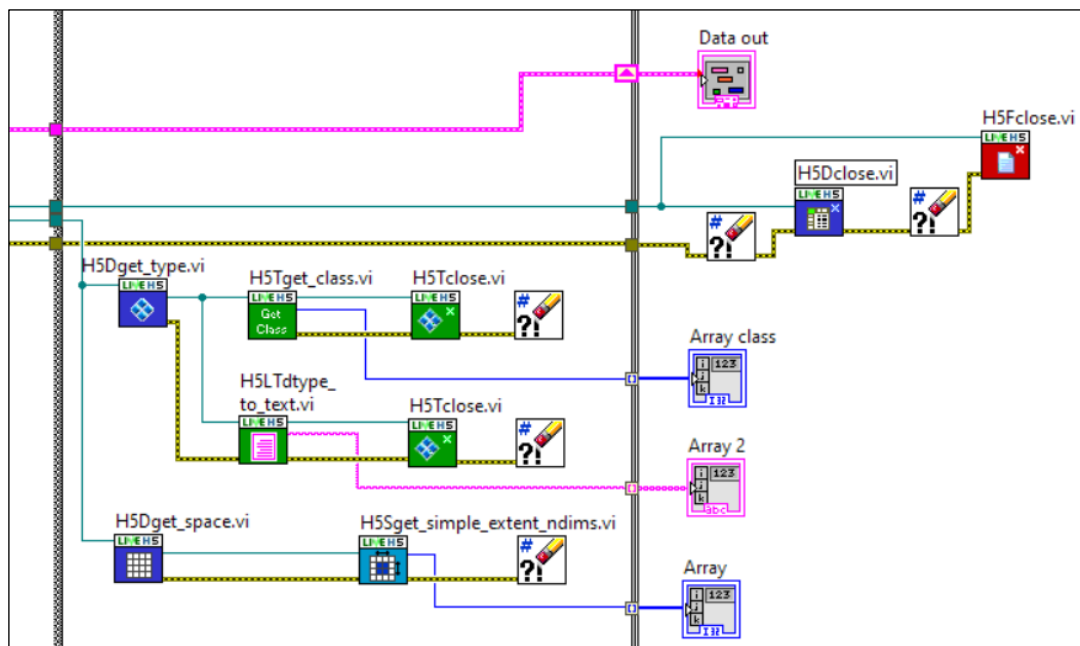
48. ábra: A „H5Dget_type.vi” felépítése [13]



49. ábra: A „H5LTdtype_to_text.vi” felépítése [13]

Az adattípus kinyerése több módszer kombinálásával érhető el. A „H5LTdtype_to_text.vi” (49. ábra) segítségével létrehoztam az „Array 2” szöveges tömböt,

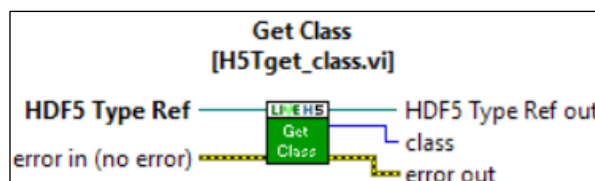
ami az egyes adattípusok (Object Types) formátumát jeleníti meg. A „Datatype” típusú „vi”-ok bekötése a Block Diagramon az 50. ábrán látható.



50. ábra: Az adattípusok az adatkészlet elemeinél a Block Diagramon

Minden „Datatype” típusú (zöld) „vi” egy „H5Tclose.vi”-jal zárható be, ez a hibavezeték és a referenciavezeték bekötésével történik.

A „H5Tget_class.vi”-jal (51. ábra) létrehoztam egy numerikus tömböt (Array class), ami megmutatja az adattípus osztályának azonosítóját. A csoport (H5G_Group) típusú objektumokhoz nem tartozik fájlformátum, így a rendszer „-1”-es jelzéssel közli, hogy üres halmaz az adat. Minden fellelhető formátum a helytakarékoság és a későbbi feldolgozás egyszerűsítése miatt saját osztályba került. A 64 bites egész számok (H5T_STD_I64LE), valamint az előjel nélküli 32 bites, hosszú egész számok (H5T_STD_U32LE) sorszáma a nulla. Ebből megállapítható, hogy bármilyen formátumú is a numerikus érték, nulla értéket ad ki a rendszer. A „strain” adat – ami lebegőpontos numerikus értékeket tartalmaz (H5T_IEEE_F64LE) – az 1-es osztályt kapta. Ezenkívül szöveges adathalmazok találtam (H5T_STRING), amik a 3-as sorszámot kapták.



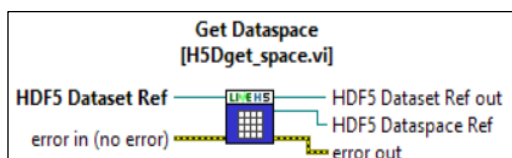
51. ábra: A „H5Tget_class.vi” felépítése [13]

A „H5Dget_space.vi”-t (52. ábra) arra használtam, hogy az 53. ábrán részletezett „H5Sget_simple_extent_ndims.vi”-nak bemenetet tudjak adni. Utóbbi „vi”-ből egy újabb numerikus tömböt kaptam, ami az 51. ábra „Array” oszlopának felel meg.

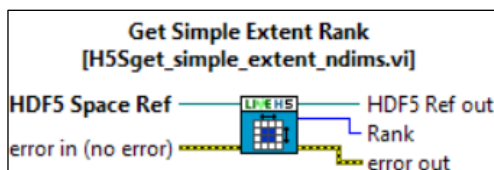
Object Types	Array 2	Array class	Array
H5G_GROUP		-1	0
H5G_DATASET	H5T_STRING {	3	0
H5G_DATASET	H5T_STRING {	3	0
H5G_DATASET	H5T_STRING {	3	0
H5G_DATASET	H5T_STD_I64LE	0	0
H5G_DATASET	H5T_STD_I64LE	0	0
H5G_DATASET	H5T_STRING {	3	0
H5G_DATASET	H5T_STRING {	3	0
H5G_DATASET	H5T_STRING {	3	0
H5G_GROUP		-1	0
H5G_GROUP		-1	0
H5G_GROUP		-1	0
H5G_DATASET	H5T_STRING {	3	1
H5G_DATASET	H5T_STRING {	3	1
H5G_DATASET	H5T_STD_U32LE	0	1
H5G_GROUP		-1	0
H5G_DATASET	H5T_STRING {	3	1
H5G_DATASET	H5T_STRING {	3	1
H5G_DATASET	H5T_STD_U32LE	0	1
H5G_GROUP		-1	0
H5G_DATASET	H5T_IEEE_F64LE	1	1

51. ábra: Az adattípusok az adatkészlet elemeinél a Front Panelen

Ebből az látható, hogy minden olyan könyvtár, ami egy gráfon megjeleníthető adattömböt, maszkot tartalmaz, az 1-es osztályba került, míg a konstans adatokat tartalmazó könyvtárak elemei nullás osztályúak.



52. ábra: A „H5Dget_space.vi” felépítése [13]



53. ábra: A „H5Sget_simple_extent_ndims.vi” felépítése [13]

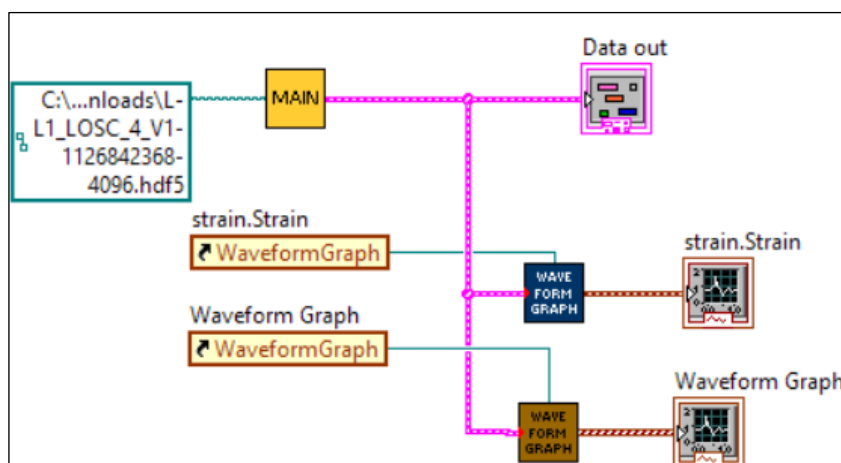
6.8. A kiolvasást végző főprogram bezárása

A „for” ciklusból már csak a kiolvasott adatállományok megjelenítéséért felelős egzakt kimenetek, a HDF kiolvasási lánc referenciavezetéke, illetve a hibavezeték jön ki. Magát az adatállományt a „H5DClose.vi”-ra kötött referencia- és hibavezetékkel zárom be, majd ezeket tovább vezetve a „H5Fclose.vi”-ba, az egész HDF kiolvasáshoz való hozzáférést is megszüntetem. Ennek a műveletgyűttesnek az elemei és bekötése az 50. ábrán látható.

7. Jelfeldolgozó és mérés technikai mellékprogramok, program optimalizálás

7.1. Programbehívó VI bemutatása

Valamennyi alprogramot egy hívó VI segítségével futtatok le, ami lényegében csak az egyes „subVI”-okat tartalmazza. Ez az 54. ábrán látható.



54. ábra: A „subVI”-okat egy hívó VI segítségével futtatom le.

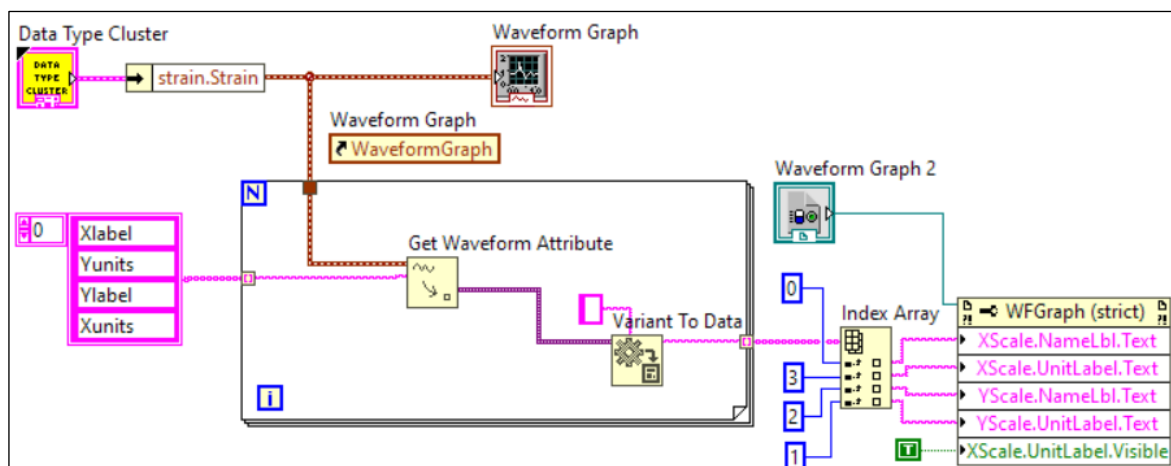
Ez a hívó VI két „subVI”-t, tehát két alprogramot tartalmaz. A kék négyzetben látható „Waveform Graph subVI” arra szolgál, hogy a főprogramban (a 54. ábrán sárga alapon „Main” elnevezéssel látható) kiolvasott nyúlás időjel (strain) adatból generált hullámforma gráf tudományos igényeknek megfelelően felcímkézhető legyen.

A mustársárga alapú „Waveform Graph subVI” pedig arra alkalmazható, hogy a nagymennyiségű „strain” adatnak csak egy részét jelenítsem meg a hullámforma gráfon.

7.2. A „strain” adatok „Waveform Graph”-jának formázására és felcímkezésére szolgáló „subVI” bemutatása

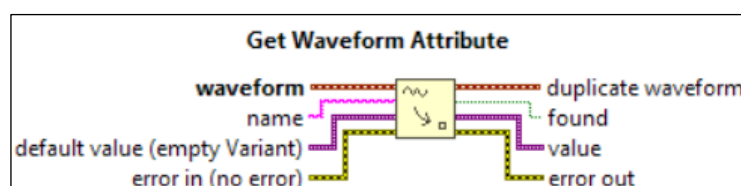
Az 55. ábrán látható program célja, hogy a főprogram (Main) során kapott „Strain Waveform Graph”-ot tudományos igényességgel felcímkézzék. A program ezenkívül képes lenne akár mértékegység hozzárendelésre, vagy bármely más, a nyúlás időjel attribútumában fellelhető, az adatokra jellemző további információ megjelenítésére.

Azért, hogy a későbbiekben hivatkozhasak a 6.3-as fejezetben bemutatott klaszterkonstans adataira, egy „Control” típusú klasztert hoztam létre belőle, így kompakt formában nyerhetők ki belőle az adatok.



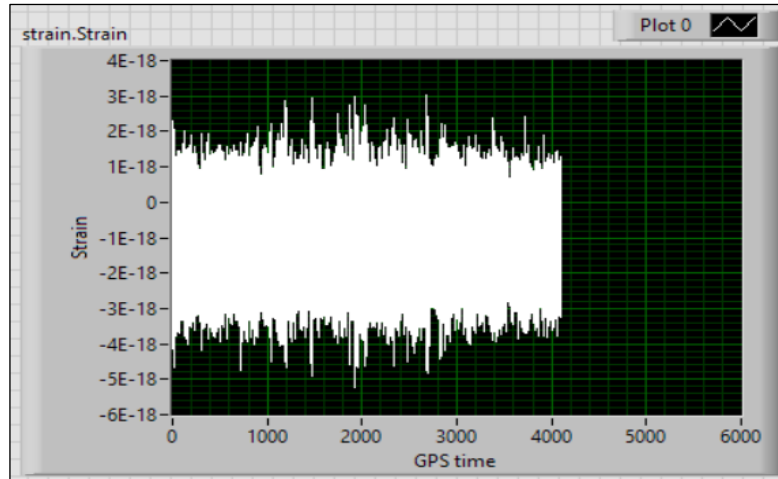
55. ábra: A „strain” adatok „Waveform Graph”-jának formázására és felcímkezésére szolgáló „subVI” Block Diagramja

Ebből a „Cluster Control”-ból a „strain” adatokat szeretném kinyerni, ehhez az „Unbundle by Name.vi”-t használtam. Az így kapott hullámformát egy „for” ciklusba vezetem be. A „Get Waveform Attributes.vi” (56. ábra) egyik bemenetére ezt, másik bemenetére pedig a „strain” attribútumának adataiból álló szövegkonstans tömböket kötöm, ezeket a „for” ciklus indexeli.



56. ábra: A „Get Waveform Attributes.vi” felépítése [13]

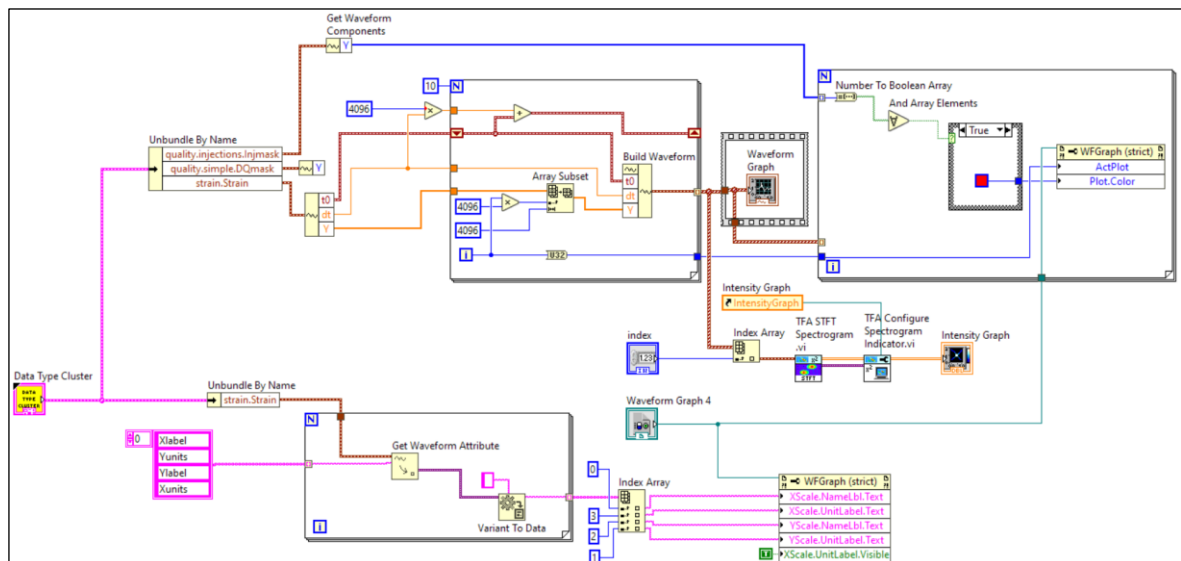
Az így kapott érték formátuma „Variant” típusú, így ezt egy „Variant to Data.vi”-jal szöveges konstanssá konvertálom át, majd megtartva az indexelést kivezetem a ciklusból. Az így kapott tömböket ezután az „Index Array.vi” segítségével logikai sorrend szerint indexelem, és az egyes indexekhez a megfelelő információtartamot csatolom. Így a nullás indexre az „Xlabel”, az elsőre az „Xunits”, a másodikra a „Ylabel”, végül a harmadikra az „Yunits” kerül. Ehhez egy „Property Node”-ot használtam, majd definiáltam a sorrendet egy referenciaként. Ezt a „Property Node” bemenetén zöld vezetékkel kapcsolódó „Waveform Graph 2” jelzi. Végül, hogy mindezt megjelenítsem, egy „Boolean” típust is felvettem a listába, értékét konstans igazra (true) állatva. Ezzel a létrejött az 57. ábrán az attribútum adatainak megfelelő lépésközü és címkéjű „Waveform Graph”. Az Y tengelyen a „Strain” címke, az X tengelyen pedig a „GPS Time” címke látható.



57. ábra: A formázó VI futtatása után megjelenik az X és Y tengelyek címkéje

7.3. A „strain” adatok részleges kiolvasása és ábrázolása

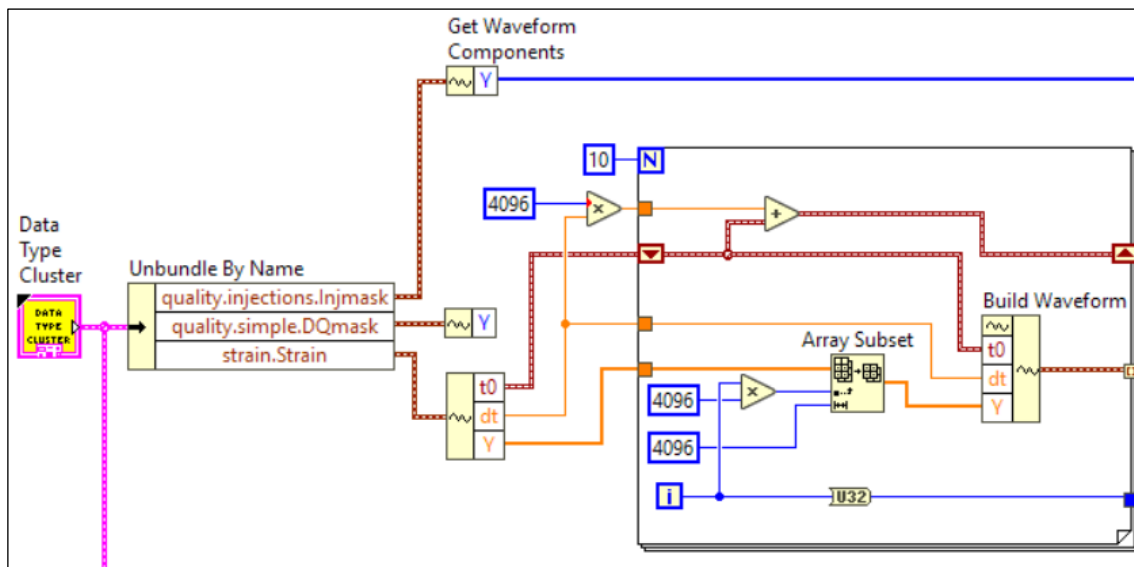
Az összes nyúlás időjel adat kinyerése is hasznos lehet kutatásokhoz, de egy ilyen nagyságú adatsokaságból nehéz az adatsűrűség miatt egyes vizsgálatok elvégzése, ehhez célszerűbb csak néhány elem ábrázolása és kiértékelése. Ezért készítettem egy olyan VI-t (58. ábra), amivel a nyúlás időjel adatok részlegesen jeleníthetők meg.



58. ábra: A nyúlás időjel adatokat részlegesen kiolvasó és ábrázoló VI

A korábban bemutatott „Data Type Cluster”-ből végzem az adatkinyerést, amiből az „Unbundle By Name.vi” segítségével kiválasztom a három hullámforma adatot, amiket a fentebbi fejezet során hoztam létre. Így az „Injmask”, a „DQmask” és a „strain” adatokat veszem ki a klaszterből. A „Get Waveform Components.vi” arra szolgál, hogy visszafejtssem a hullámforma egyes komponenseit: a maszkokból csak a függvény Y értékeit, a nyúlás-időjelből pedig ezen kívül a mintavételezés kezdeti pillanatát (t0), az idő megváltozását (dt)

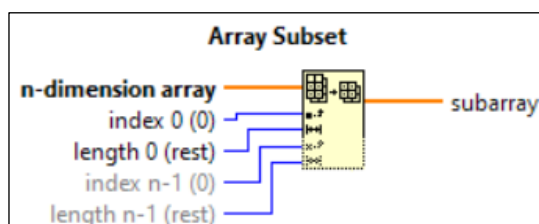
is. A „strain” adatok komponenseit egy „for” ciklusba kötöm (59. ábra), mely ciklus során csak az első tíz beérkező adatot vizsgálom. A „N”-be kötött 10-es érték azt fejezi ki, hogy az első tíz beérkező adat lefuttatása után megáll a ciklusvégrehajtás.



59. ábra: A „strain” adatot részleges beolvasó VI részlete

A t_0 kezdeti időadatot egy „Shift Register”-en keresztül vezetem be a ciklusba. A dt értékét 4096-tal, az egy maszkhoz tartozó adatpontok számával szorzom meg, majd bekötöm ezt is a „for” ciklusba. Ehhez az adathoz hozzáadtam a t_0 kezdeti értéket, majd ezt vezettem a „Shift Register” kimenő csatlakozójába. Ezzel a maszkot másodpercenként egy-egy elemre szedem szét. A létrehozandó új hullámforma t_0 adatát tehát az egyes hullámkomponenseknél a „Shift Register” éppen aktuális értéke adja meg. A dt osztásköz nem változik meg azzal, hogy csak az első tíz elem megjelenítése a cél, ezért ezt változatlanul tovább kötöm a „for” cikluson belüli „Build Waveform.vi”-ba.

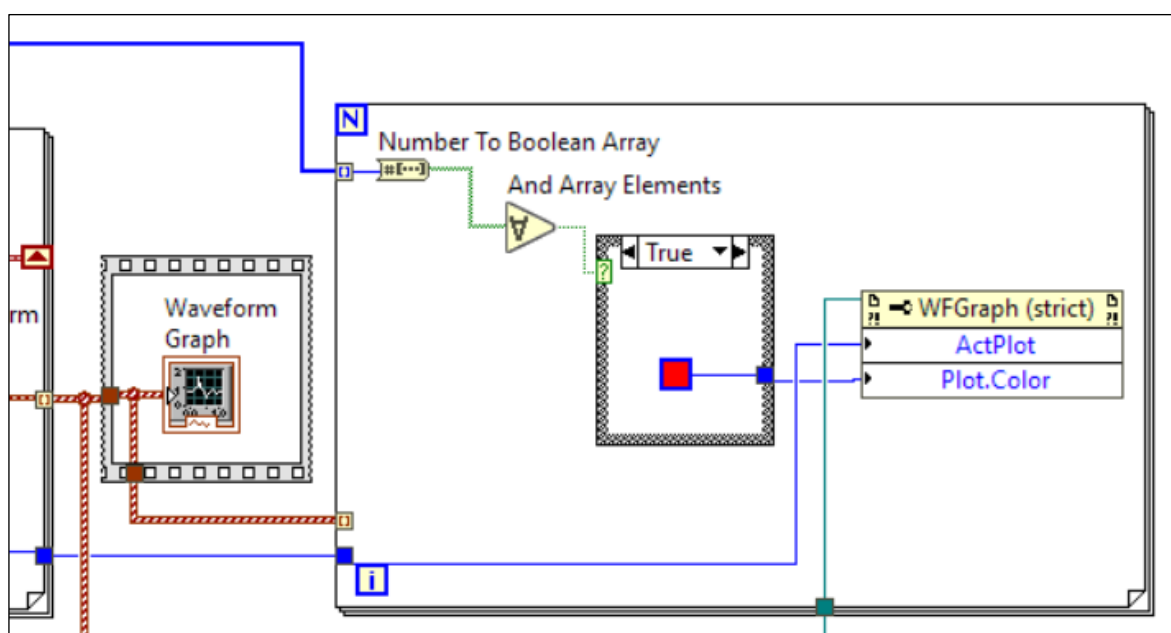
A Y értékeket egy „for” cikluson belüli „Array Subset.vi”-ba kapcsolom. Ennek az 60. ábrán látható „vi”-nak az „index” bemenetén a létrehozni kívánt tömb első elemét definiálom, ami az éppen aktuális iteráció – ami 0-10 közötti értéket vehet fel – 4096-szorosa esetemben. 4096 elemet veszünk ki a nyúlás időjelből, mindezt 4096-os index eltolásokkal, ezért szükséges ezeknek a konstansoknak a felhasználása a tömb létrehozásánál.



60. ábra: Az „Array Subset.vi” felépítése [13]

Az Y érték az új hullámformán tehát a fentebbi módosításokat követően kerül a „Build Waveform.vi”-ba. Az így kapott hullámforma adatai a helyes sorrend betartása miatt a „for” ciklusból kijövet indexelést kapnak.

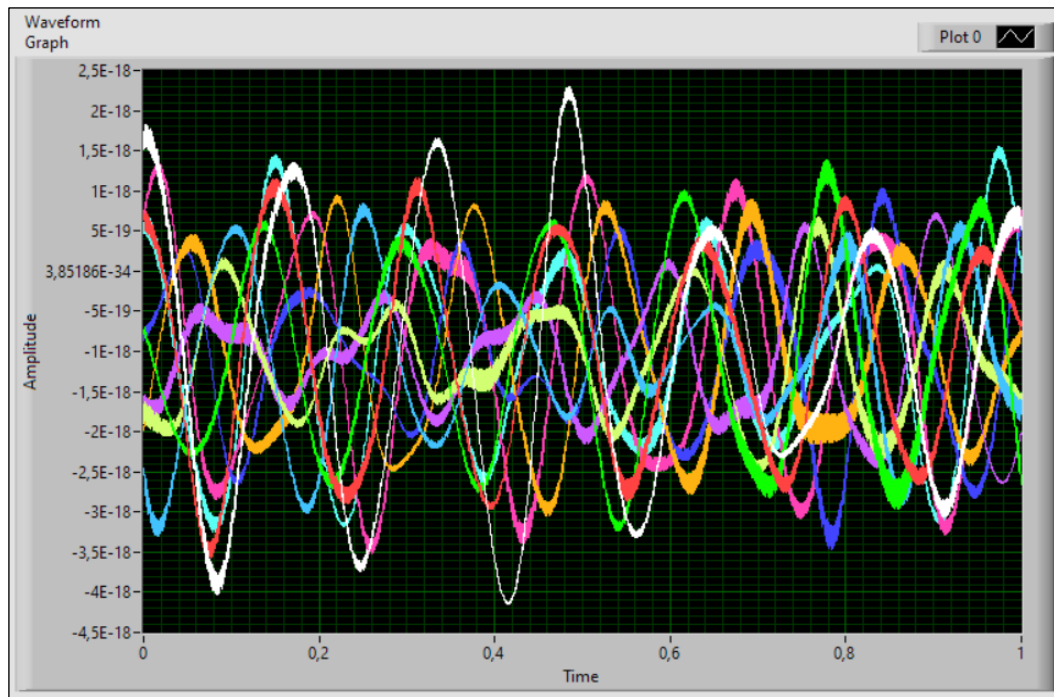
A hullámforma megjelenítését végző „Waveform Graph”-ot egy „Flat Sequence”-be kötöttem. Utóbbi ciklust azért alkalmazom, hogy még az egyes komponensek színezése előtt mindenféleképpen rajzolja meg a LabVIEW a hullámforma elemeit. Máskülönben a színezést végző ciklus egyszerre futna le az ábrázolást végző ciklussal. Ez olyan hibához vezetne, hogy utóbbi nem találna színezhető adatot és a „Waveform Graph” emiatt üres lenne. Tehát a rangsorolt programlefutás miatt vezettem be a „Flat Sequence”-ből a színezést végző „for” ciklusba a hullámforma indexelt vezetéket.



61. ábra: A színezést végző „for” ciklus a Block Diagramon

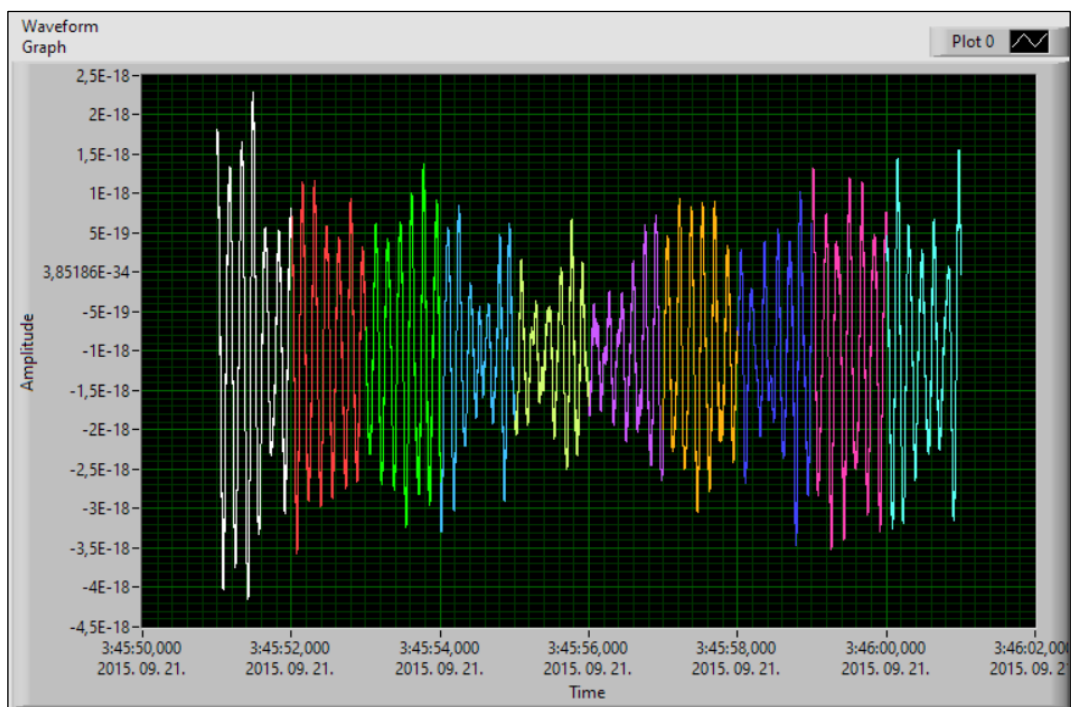
A színezés az „Injmask” Y értékei alapján történik. Miután indexelve bevezettem ezeket egy „for” ciklusba, a számokat egy bináris adattípussá, „Boolean”-né alakítom át a „Number To Boolean Array.vi” segítségével. Az „And Array Elements.vi”-t arra használom, hogy minden iterációnál csak vagy egy igaz vagy egy hamis értéket kaphassak, ami alapján differenciált színek oszthatók ki. A „case” struktúrának a „true” és „false” ágába két „Color Box” konstantt kötöttem, majd ezt bevezettem a „Property Node” függvénygörbét színező „Plot.Color” bemenetére. A nyúlás időjel első tíz elemén az „ActPlot”-tal aktiváltam a színezést, így végső soron a 62. ábrán látható görbét kaptam. A „WFGraph (strict)” kapott egy referenciát is, ez az 58. ábrán látható. Itt megfigyelhető az is, hogy alkalmaztam a 7.2-

es fejezetben bemutatott és az 55. ábrán szemléltetett formázó VI-t is, így a Front Panelen kapott hullámformát fel is címkéztem.



62. ábra: Az színezett görbék időrendi sorrendiség nélkül

Az 62. ábra látványos ugyan, de a hullámok egymáson és nem időrendben szerepelnek, így a Front Panelen a gráfon beállítottam, hogy az „Ignore Time Stamp” parancs ne teljesüljön.

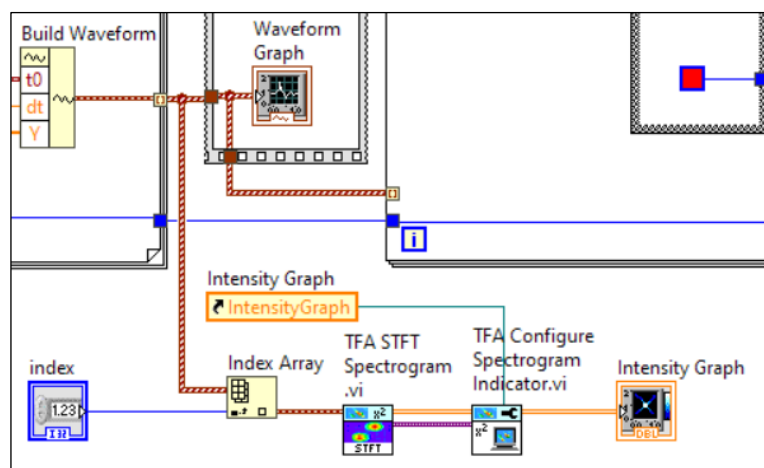


63. ábra: A színezett hullámok az időrendi sorrendiség ábrázolásával

Így kaptam meg az 53. ábrán látható „strain” adat első tíz, színezetten és időrendben megjelenített hullámát.

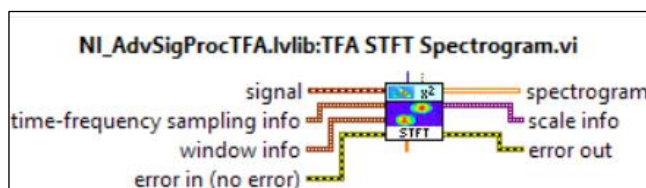
7.4. Spektrogram alkalmazása a hullámok ábrázolására

Az 58. ábrán látható VI tartalmaz még egy spektrogram megjelenítőt is. A 64. ábrán látható, hogy a fentebb is használt hullámformát kötöm be egy tömb indexelő, ún. „Index Array.vi”-ba. Az „index” csatlakozóra pedig egy „control” típusú bemenetet rakok, ezzel egy 0-10 közötti szám választható.

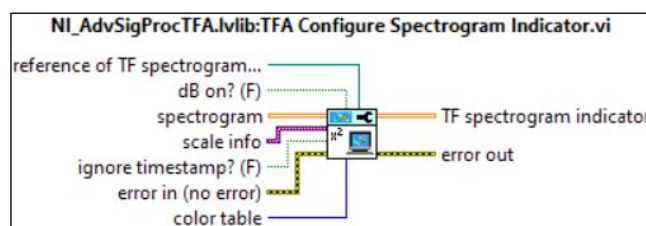


64. ábra: A spektrogramot megjelenítő programrészlet

A „TFA STFT Spectrogram.vi” (65. ábra) egy a dolgozat elején említett bővítmény, az Advanced Signal Processing Toolkit beépített függvénye. A frekvencia analízis (Time Frequency Analysis) rövid idejű Fourier-transzformáció (Short-time Fourier Transformation) spektrogramnak több bemenet is megadható, én csak a függvény jelet kötöttem be.

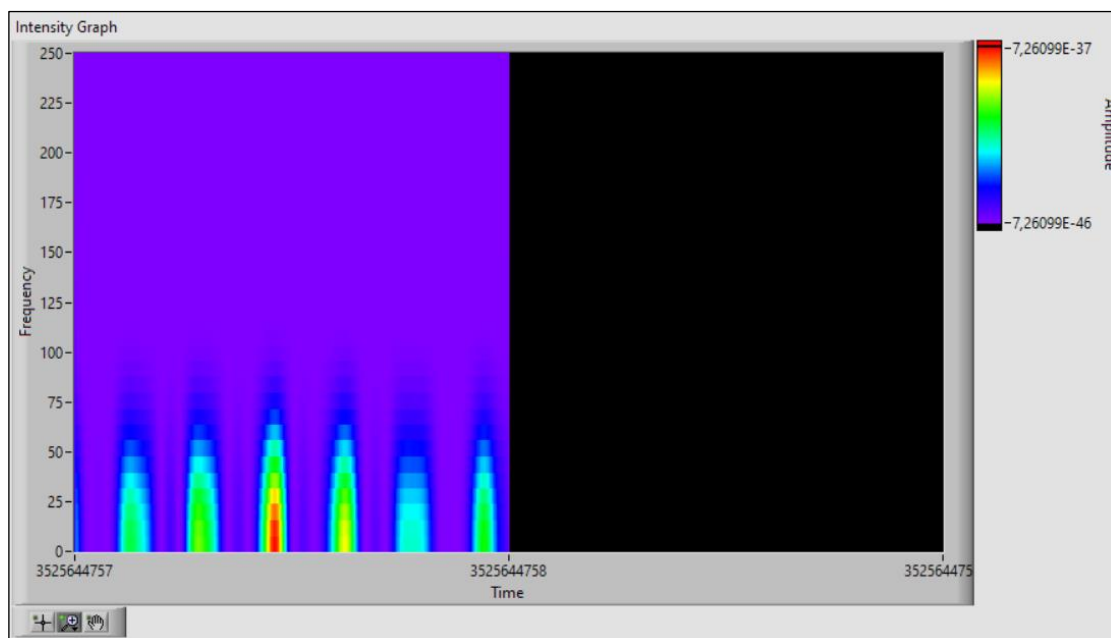


65. ábra: A „TFA STFT Spectrogram.vi” felépítése [13]



66. ábra: A „TFA Configure Spectrogram Indicator.vi” felépítése [13]

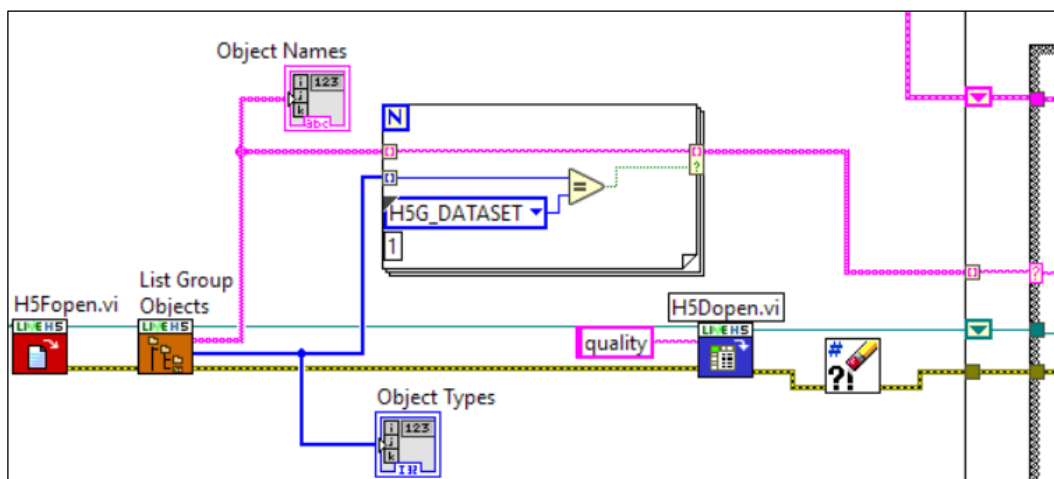
Az ebből kimenő spektrogram jelet egy a „TFA Configure Spectrogram Indicator.vi”-ba kötöttem (66. ábra). Itt is lehetne a frekvencia analízist tovább konfigurálni, de csak a „TFA STFT Spectrogram.vi” kimeneteit és egy „Intensity Graph” referenciát adtam meg bemenetként, majd egy „Intensity Graph” megjelenítőt választottam az adatok ábrázolására a kimeneti oldalon. A 67. ábrán látható a Front Panelen kapott spektrogram.



67. ábra: A Fourier-transzformációból kapott spektrogram

7.5. Programoptimalizálás

A program megírásának végére a futtatási idő jelentősen megnőtt, és előfordult, hogy a számítógép teljesítménye nem bírta a programfuttatást. Ezért szükségessé vált a program egyes elemeinek optimalizálása, illetve egyes hibavezetékek, referenciavezetékek lezárása. Ennek megfelelően az optimalizálást a 68. ábrán látható résszel kezdtem.

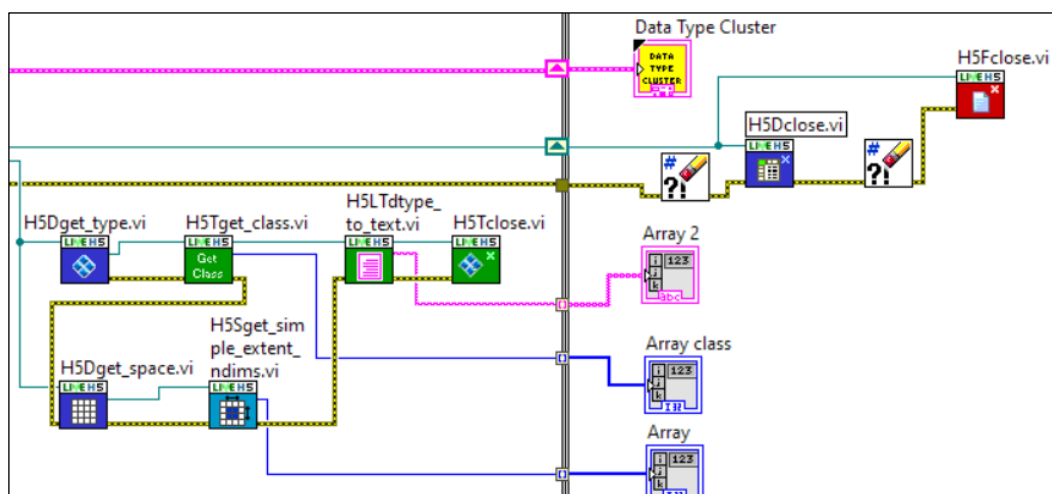


68. ábra: Optimalizálás a „for” ciklus futtatását megelőző szekción

Az esetválasztót is tartalmazó „for” ciklusba nem engedem belépni, csak a „Dataset” típusú adatokat. Ezzel kikerülöm, hogy az üres adatként megjelenő állományok (pl.: H5G_Group) a program futását megakasszák, vagy lelassítsák. Ezért a listázott objektumtípusokat egy olyan (a 68. ábrán kis méretben látható) „for” ciklusba rendeztem, ami csak a „H5G_Dataset” adattípusokat engedi tovább. Így az objektumok nevei (Objects Names) csak feltételesen és indexelve hagyhatják el a ciklus, csak akkor, ha típusuk „Dataset”.

További optimalizációs lehetőséget adott a referenciavezetékek és a klaszterkonstans adatainak „Shift Register”-be kötése. Ez biztosítja, hogy az ezeken a vezetékeken haladó információ is tartja a sorrendiséget, csökkentve a programösszeomlás esélyét.

A 69. ábrán látható a „Shift Register” másik vége, illetve a 6.7-es fejezetben használt „vi”-ok optimalizálása. Utóbbit úgy oldottam meg, hogy a hibavezetékét és a referenciavezetékét konzekvensebben kötöttem be. A hibavezetékek számát egyre redukáltam, ez a vezeték viszont mindegyik „vi”-on keresztülhalad. A „H5Tclose.vi”-ból is csak egyet hagytam, ez viszont valamennyi hibát és referenciát lezár. Ezek a módosítások is lerövidítették a programfuttatást és megnövelték a stabilitást.



69. ábra: Az optimalizálás során a hiba- és referenciavezetékek bekötését is módosítottam

8. Konklúzió

A TDK munka során létrehoztam a célként kitűzött, gravitációs hullámokat beolvasó és az adatok feldolgozására képes programcsomagot. Azt a tudományos eredményt is sikerült elérnem, hogy a programcsomag stabilan futtatható, így publikálható a nyilvánosság számára. A programcsomag fájljai a [17]-es forrásban feltüntetett linken érhetők el és szabadon felhasználhatók.

A dolgozat során sikerült olyan megoldásokat találnom, melyek képessé tették a programot valamennyi, a GWOSC honlapján fellelhető gravitációs hullámok adatait tároló, HDF formátumú fájl kiolvasását. Ezen túlmenően az adatokat nem csak megjeleníteni, de feldolgozni is lehet az egyes VI-ok segítségével, ezzel meghaladtam a HDFView korlátait, ami csak az adatkilvasást tette lehetővé.

Munkám során nem csak a programalkotást valósítottam meg, hanem közérthetően be is mutattam a témához tartozó háttérismereteket. Betekintést adtam a HDF adatformátum struktúrájába, alkalmazásának előnyeibe. A szakirodalmak tanulmányozása alapján vázoltam a gravitációs hullámok megértéséhez szükséges fizikai alapelveket, fogalmakat. Bemutattam ezen hullámok kimutatását végző interferométereket, obszervatóriumokat.

A főprogram megírását követően olyan példaalkalmazásokat készítettem, amik demonstrálták a használhatóságot, illetve megmutatták az olvasók, illetve a programcsomagot a jövőben használók számára, hogy milyen jelfeldolgozó eszköztár hasznosítható a National Instruments LabVIEW szoftverkörnyezetében.

9. Felhasznált irodalom

1. Wikipedia: Hierarchical Data Format, URL: https://en.wikipedia.org/wiki/Hierarchical_Data_Format megtekintés dátuma: 2021. 01. 26.
2. Andrew Collette: Python and HDF5, O'Reilly Media, Inc, 2014, ISBN: 978-1-449-36783-1
3. HDF Group: What is HDF5?, URL: <https://www.hdfgroup.org/solutions/hdf5> megtekintés dátuma: 2021. 01. 26.
4. „Egy felfedezés, ami megváltoztatja eddig ismert világunkat”, Raffai Péter (PhD) információs honlapja, URL: <https://gravitacioshullam.hu/gravitacios-hullamok/mik-a-gravitacios-hullamok> megtekintés dátuma: 2021. 05. 28.
5. Hartmut Grote: Gravitational Waves: A History of Discovery, CRC Press Taylor & Francis Group LLC, 2020, ISBN: 978-0-367-13682-6
6. Jagannath D. Ranade: Application of the Fabry-Perot interferometer for high resolution studies, Royal Holloway College, (University of London) Englefield Green, Surrey, 1950
7. Massimo Bassan (szerk.): Advanced Interferometers and the Search for Gravitational Waves, Springer International Publishing Switzerland, 2014, ISBN 978-3-319-03792-9
8. VIRGO hivatalos weblap: VIRGO in a nutshell, URL: <http://public.virgo-gw.eu/virgo-in-a-nutshell/> megtekintés dátuma: 2021. 07. 28.
9. VIRGO hivatalos weblap: Advanced VIRGO, URL: <http://public.virgo-gw.eu/advanced-virgo/> megtekintés dátuma: 2021. 07. 28.
10. KAGRA hivatalos weblap: KAGRA Gravitational-wave Telescope Starts Observation, URL: <https://gwcenter.icrr.u-tokyo.ac.jp/en/archives/1381> megtekintés dátuma: 2021. 07. 28.
11. Gravitational Wave Open Science Center hivatalos weblap egyes obszervatóriumok által rögzített fájlok eléréhez, URL: <https://www.gw-openscience.org/about/> megtekintés dátuma: 2021. 07. 28.
12. LIGO-Virgo Collaboration: Observing Capabilities, Timeline, URL: <https://emfollow.docs.ligo.org/userguide/capabilities.html> megtekintés dátuma: 2021. 07. 28.
13. LabVIEW Professional 2018. szoftver beépített súgója
14. Gravitational Wave Open Science Center hivatalos weblap, URL: https://www.gw-openscience.org/o2_inj/ Megtekintés dátuma: 2021. 10. 11.
15. Gravitational Wave Open Science Center hivatalos weblap, URL: https://www.gw-openscience.org/o2_details/ Megtekintés dátuma: 2021. 10. 18.
16. HDF Group hivatalos weblap, URL: www.hdfgroup.org/downloads/hdfview Megtekintés dátuma: 2021. 11. 02.
17. Saját forrás: A dolgozathoz tartozó programcsomag letöltési linkje, URL: https://drive.google.com/drive/folders/1p_S74NH4cW9HNbvXEtmETRToRBfVp_Mf?usp=sharing