Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Szélessávú Hírközlés és Villamosságtan Tanszék

Szoftverrádió alapú Smog-1 elsődleges földi állomás vevő fejlesztése és optimalizálása

Diplomaterv

Készítette
Várdai Attila

Konzulensek
Dudás Levente
Vágó Péter

2016. május 22.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Várdai Attila*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot/ diplomatervet (nem kívánt törlendõ) meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerzõ(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhetõ elektronikus formában, a munka teljes szövegét pedig az egyetem belsõ hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetõvé.

Budapest, 2016. május 22.

Várdai Attila
hallgató

Kivonat

Abstract

Bevezető

1. fejezet

Smog-1 műhold fedélzeti kommunikációs rendszere

2. fejezet

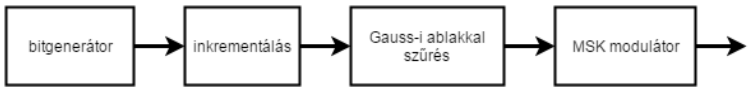
Analóg és digitális modulációs eljárások gyakorlati szempontból

3. fejezet

Szimulációs program Labview környezetben

3.1. Az adó megvalósítása

Az adó megvalósítása Labview környezetben történik, ahol az egyes funkcionális részegységek külön alprogramokban realizálódnak, ezzel a logikai elkülönítést és a kód átláthatóságát lehet biztosítani. Az adó felépítését a 3.1. ábra szemlélteti.

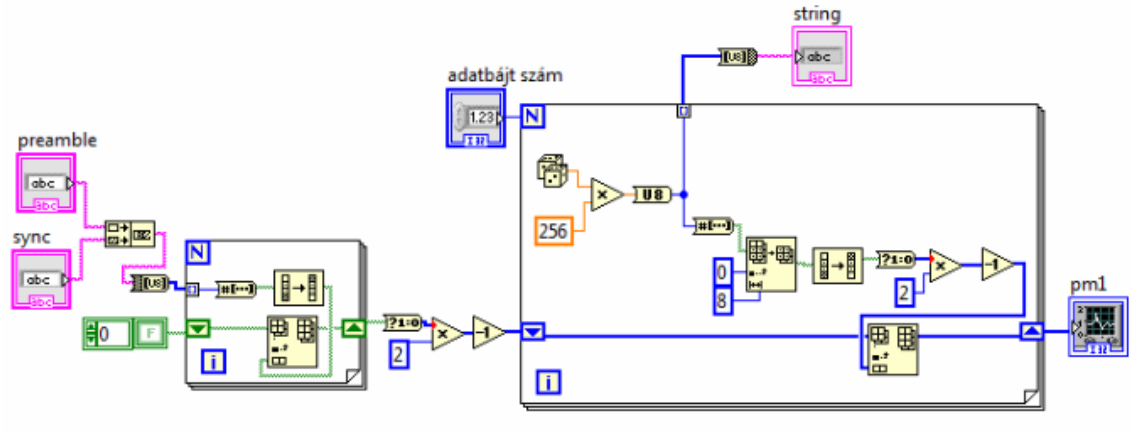


3.1. ábra. *A megvalósított adó blokkdiagramja.*

3.1.1. Bitgenerátor

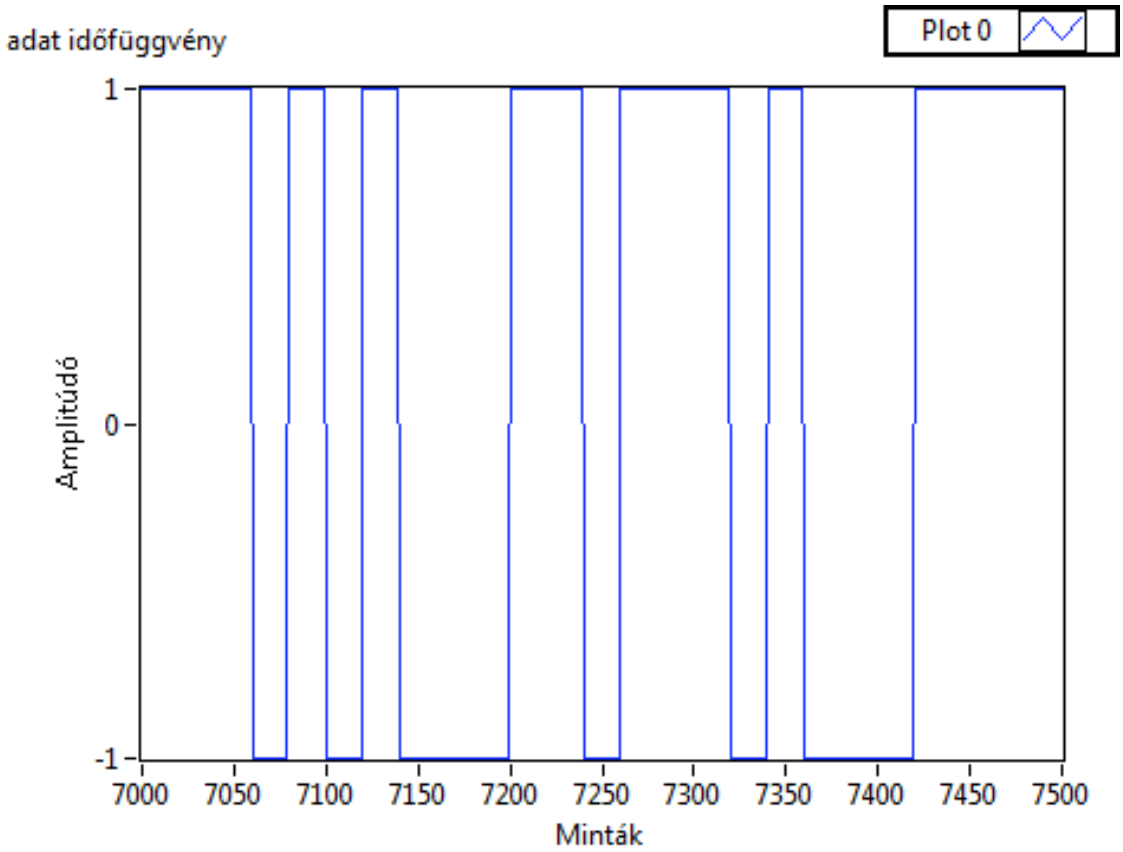
Az adó szimulálásának egyik fő eleme a bitgenerátor, melynek feladata a álvéletlen bitek, avagy az adat generálása a vevő számára. Az adatok továbbítása strukturáltan történik, csomagok (Package) formájában. A csomagok elején a frekvencia kompenzációhoz használt bevezető (Preamble) bitek találhatók, opcionálisan beállítható hosszúságú nulla-egy bitsorozat. A következő két bájt (un. sync word) tartalmazza a szinkronizációs biteket, amelyek segítségével a csomagok detektálása történik vevő oldalon. Alap beállításban hexa 2DD4 a szinkronizációs szó, melynek több előnyös tulajdonsága is van. A szinkronizációs biteket már a tiszta, hasznos adat követi.

A bitgenerátor nevű programrésznek (3.2. ábra), mint önálló alprogramnak, három bemenete van, a Preamble és a sync. word, amelyek szöveges (string) formátumban kerülnek bekérdezésre a bemeneten. A harmadik paraméter egy egész típusú szám, amin keresztül a hasznos adat bájtok számát lehet megadni. A program elején beolvasott szinkronizációs szó és Preamble átkonvertálása történik 0-1 sorozattá, majd a biteknek energiatartalmat is adunk a $2X - 1$ művelettel. Az így kapott -1,+1 sorozathoz fűzzük hozzá a véletlenül kreált adott darabszámú bájtokat, miután azokból is -1,+1 sorozat készült. A bitgenerátor alprogramnak két kimenete van, egyik az energiatartalommal rendelkező adatcsomag



3.2. ábra. *A bitgenerátor programrész kódja*

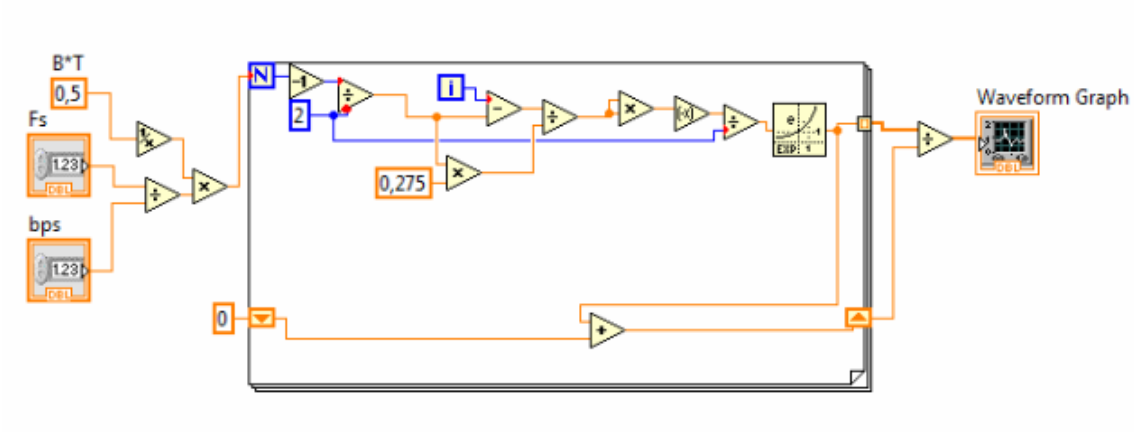
(3.3. ábra), a másik a hasznos adat hexa string formátumban, ami később az adott és vett csomagok összehasonlítását egyszerűsíti meg.



3.3. ábra. *A bitgenerátor kimeneti jelalakja inkrementálva.*

3.1.2. Gauss-i ablak

A GMSK modulátor rövidítésében a "G", mint Gaussian arra utal, hogy Gauss-i lekerekített ablakkal szűrjük a biteket mielőtt a modulátor bemenetére jutnának. Ezért kell egy Gauss window nevezetű alprogram is (3.4. ábra.).



3.4. ábra. *A Gauss-i ablak programrész kódja.*

A bemenetei az f_s mintavételi frekvencia és az adatsebesség (*bps* nevű változó). Egész pontosan a két változó hányadosára van szüksége a programnak, ami azt jelenti, hogy egy adott bit hányszor van mintavételezve. BT jelen esetben egy konstans érték ami azt mondja meg, hány bitre visszamenőleg rendelkezik a rendszer memóriával. Ha 0,5 az értéke, akkor nem csak az aktuális, hanem az előző bit értékét is figyelembe veszi a rendszer, ezért csak akkor éri el a Gauss-i ablakkal szűrt adatfüggvény a +/- 1 értéket ha egymás után több logikai nullás, vagy egyes követi egymást a bitsorozatban (3.5. ábra.). A program a következő képlet alapján állítja elő a Gauss-i ablakot: [1]

$$v[i] = \frac{i - \frac{N-1}{2}}{0,275\frac{N-1}{2}} \qquad i = 0 \ldots N-1 \tag{3.1}$$

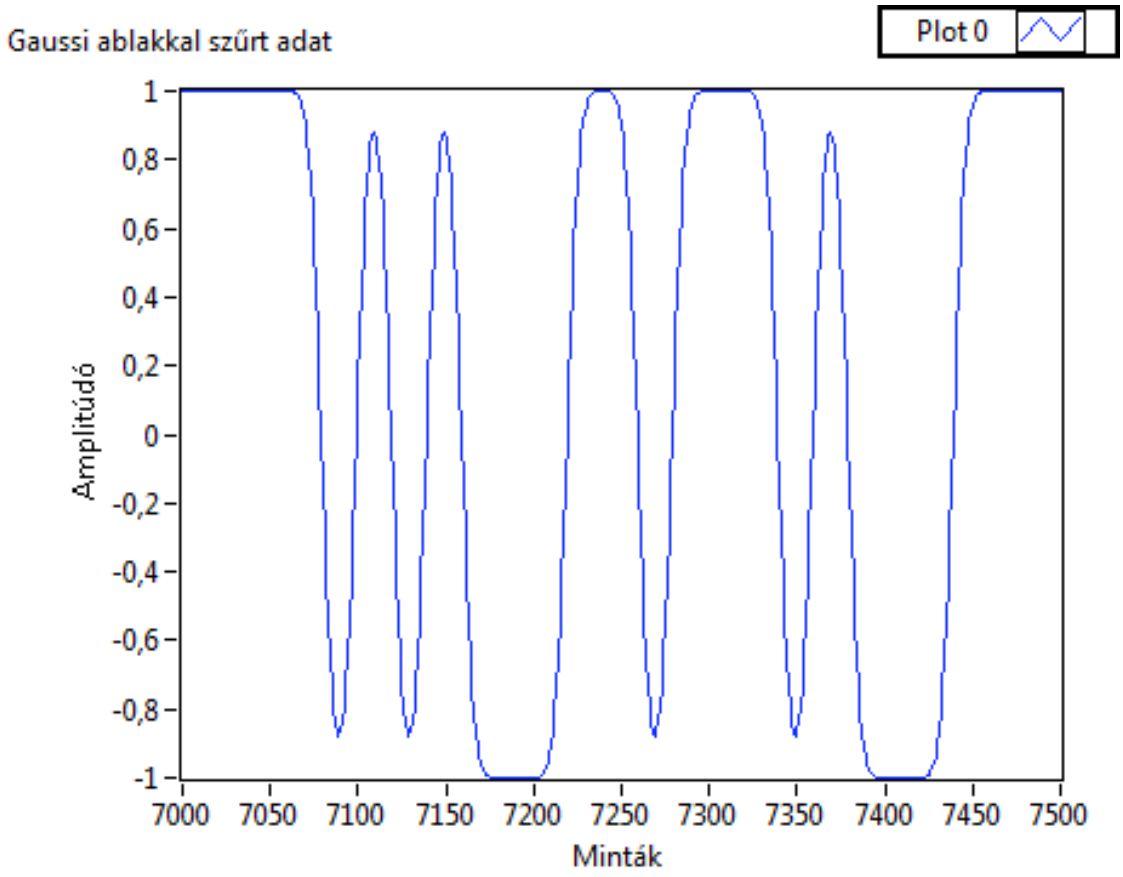
$$w[i] = e^{(\frac{-v[i]}{2})^2} \qquad i = 0 \ldots N-1 \tag{3.2}$$

$$w_n[i] = \frac{w[i]}{\sum |w[i]|} \qquad i = 0 \ldots N-1 \tag{3.3}$$

Ahol a 0,275 mint konstans szorzótényező, a Gauss-i ablak alakjáért felelős, ha ennél kisebb szám, akkor az ablak elkeskenyedik, szélei felől közeledve egyre több helyen vesz fel nulla értéket, míg ha nagyobb, akkor szélesebb lesz, és nem feltétlenül nulla értéket vesz fel a szélein, pedig pont ezért van sávszélesség csökkentő hatása a Gauss-i ablakkal való szűrésnek, mert az egyes bitek közti átmenet nem ugrással történik, hiszen az időtartományban gyors fel- vagy lefutás nagy spektrális tartalommal jár.

A jelenséget a 3.6. és 3.7. ábrák illusztrálják, látható, hogy a Gauss-i ablakkal szűrt adat kimenete spektrálisan hatékonyabb a négyszög ablakozást használó módszerhez képest. A 3.6. ábra. spektrumképe a bitgenerátor inkrementált kimenetének (3.3. ábra.).

Alap esetben $f_s = 250000$, $bps = 12500$, ezért egy bit 20-szor van mintavételezve. Mivel $BT = 0,5$, azaz két bit hosszú a Gauss-i ablak, így a kimeneten 40 minta hosszú ablakot kapunk. Látható, hogy a 0,275-ös konstans használatával a szűrő szélein lévő minták nulla értékűek, biztosítva a megfelelő spektrális hatékonyságot.



3.5. ábra. *Az inkrementált kimenete a bitgenerátornak, Gauss-i ablakkal szűrve.*

3.1.3. MSK modulator programrész

Ez a programrész önmagában egy MSK modulátor, a Bemenetére érkező adat már Gaussi ablakkal szűrt (3.5. ábra). További bemenetei a vivőfrekvencia (fv), a mintavételi frekvencia (fs), illetve az adatsebesség (datarate). Továbbiakban a frekvencia moduláció realizálódik a programrészben:

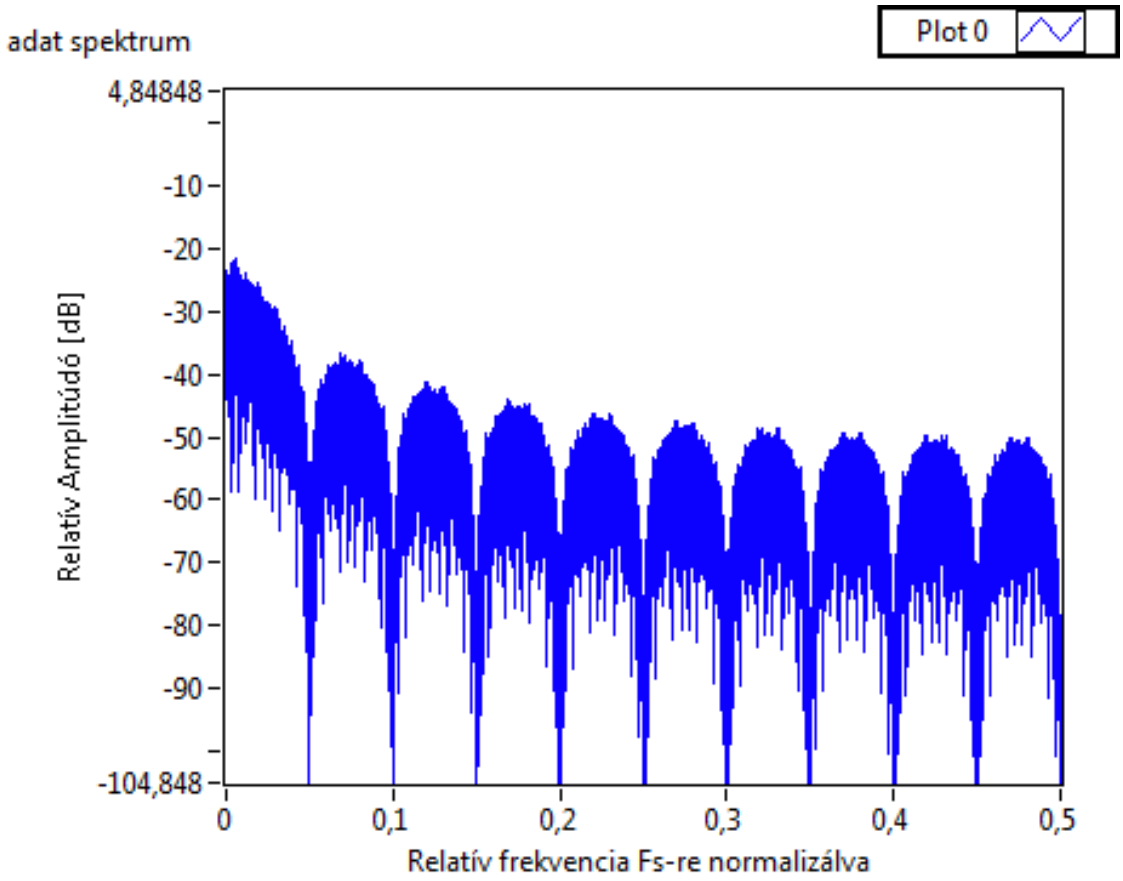
$$S_{FM}(t) = U_v \cos(\omega_v t + 2\pi k_{FM} \int_0^t S_m(\tau) d\tau) \tag{3.4}$$

$$k_{FM} = \frac{dev}{f_s} \tag{3.5}$$

$$dev = \frac{datarate}{4} \tag{3.6}$$

$$S_{FM}[i] = U_v \cos(2\pi f_v \frac{i}{f_s} + 2\pi \frac{datarate}{4 f_s} \sum_{i=0}^{N-1} S_m[i]) \tag{3.7}$$

Ahol $S_m[i]$ a Gauss-i ablakkal szűrt adattömb elemeit jelenti. A programrész kimenete komplex kell legyen, hiszen az adó és a vevő közt nem koherens az átvitel, ha csak valósban működne, vételi oldalon előállhatna olyan helyzet, hogy nem veszünk semmit, mert a fázisviszonyok pont úgy alakulnak. A programrész kimenetén tehát a koszinuszos valós rész



3.6. ábra. *A bitgenerátor kimenetének spektrumképe.*

illetve a szinuszos képzetes rész kerül átadásra egy komplex tömb formájában. Természetesen a komplex kimenetet még $\sqrt{2}$-vel szorozni, kell hogy a teljesítmény egységnyi legyen, ez azért lényeges, hogy a jel/zaj viszonyt meg lehessen határozni additív fehér Gauss-i zajos csatorna esetén.

Ha a kimenet valós részének spektrumát jelenítjük meg, bizonyos bemeneti paraméterekkel, illetve zajjal terhelve, akkor a következő spektrumábrához jutunk 3.9. ábra.

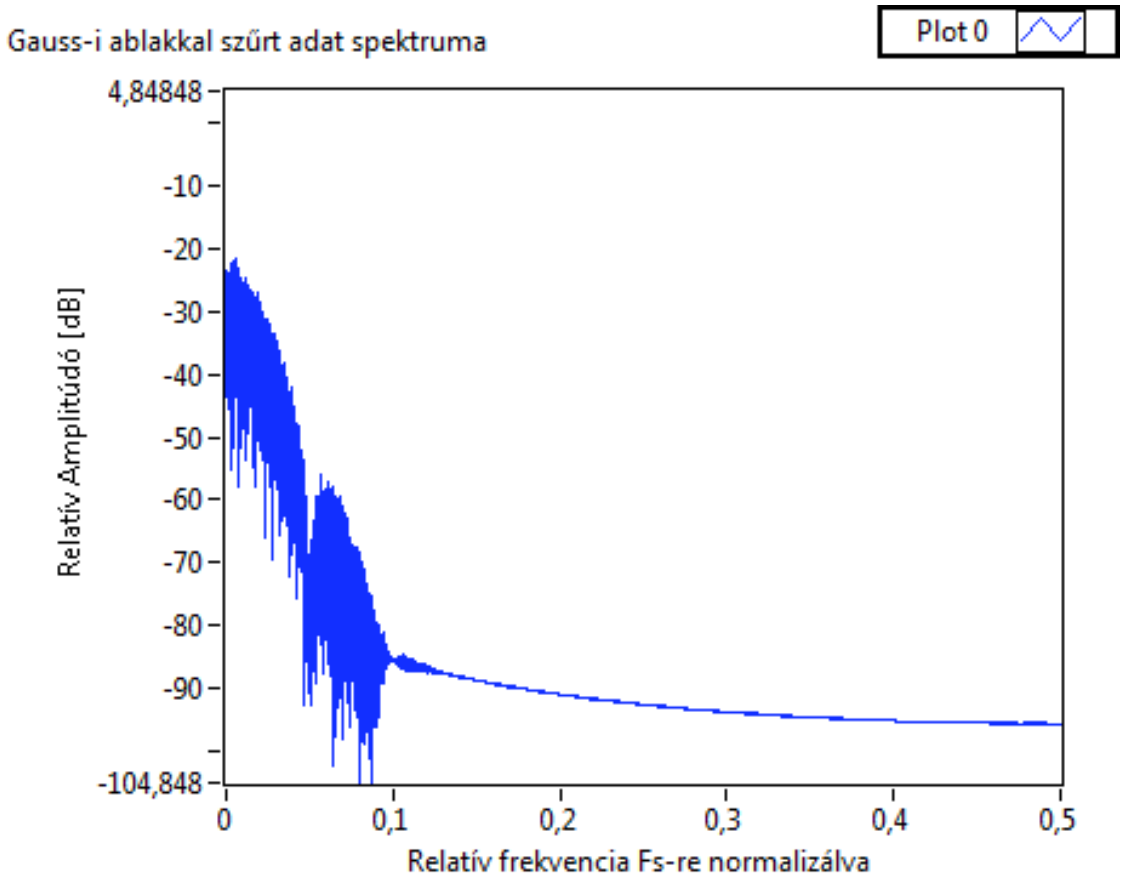
3.1.4. Additív fehér Gauss-i zajos csatorna szimulálása

A szimuláció alapcélja, hogy megvizsgáljuk az egyes implementált vevők hogyan viselkednek additív fehér Gauss-i zajos (AWGN) csatorna jelenlétében. Ez a csatorna jól szimulálja a Smog-1 műhold és a vevő közötti csatornát, hiszen a közvetlen rálátás biztosított a műholdra, a többutas terjedés pedig nem jellemző.

Az AWGN csatorna megvalósítása Labview környezetben Box-Müller algoritmussal történik, mellyel két független véletlen változót lehet előállítani standard normális eloszlással.

$$Z_1 = \sqrt{-2\ln(x_1)}\cos(2\pi x_2) \tag{3.8}$$

$$Z_2 = \sqrt{-2\ln(x_1)}\sin(2\pi x_2) \tag{3.9}$$

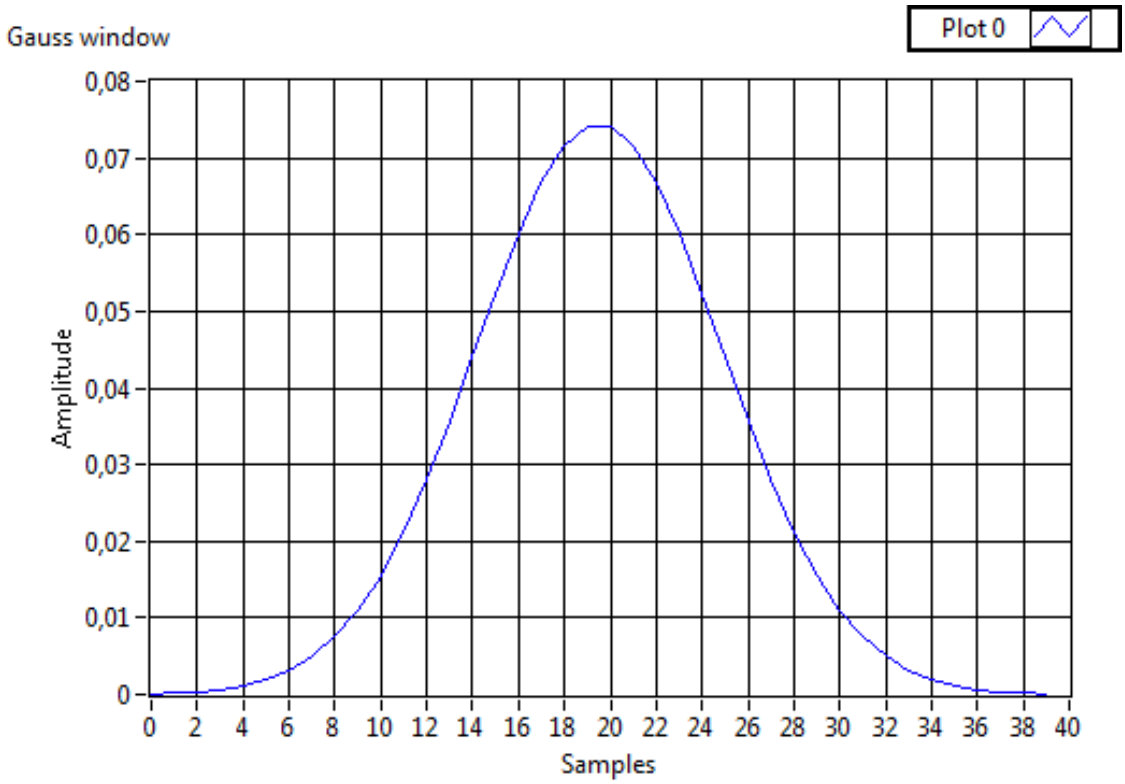


3.7. ábra. *A bitgenerátor kimenetének spektrumképe Gauss-i ablakkal szűrve.*

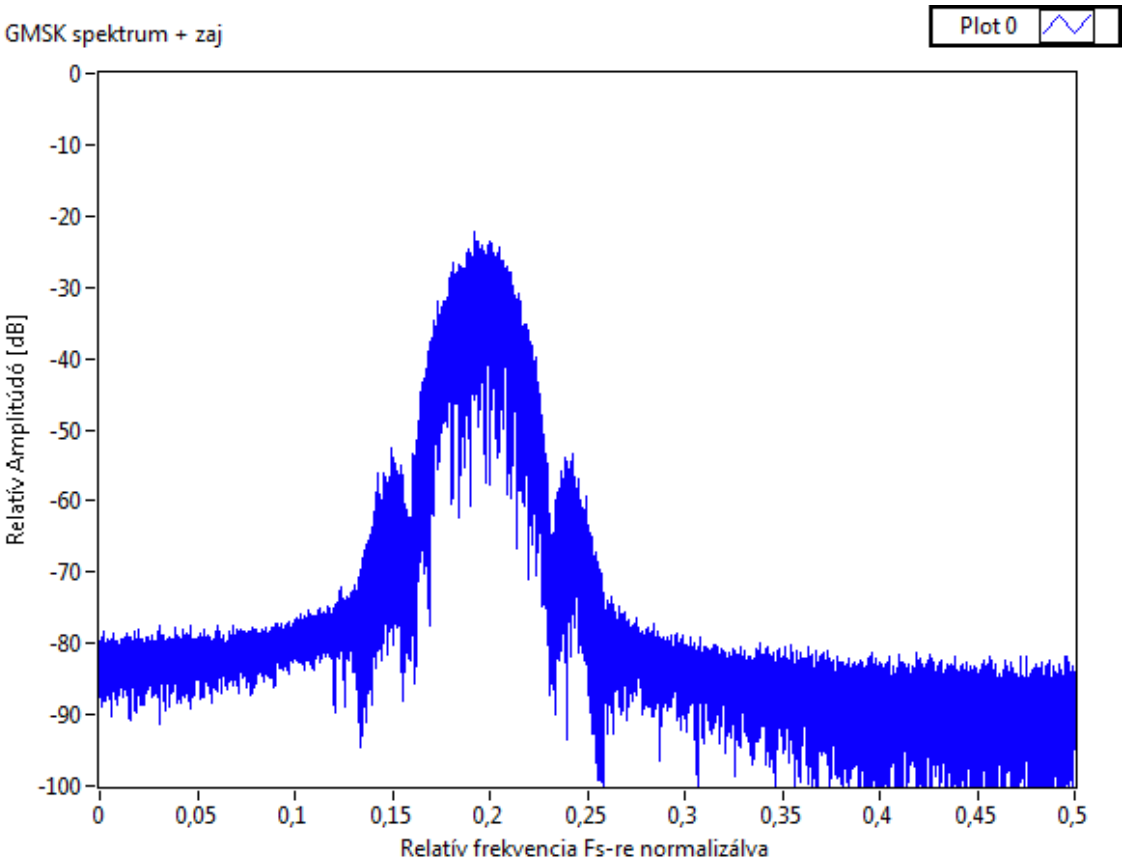
Ahol x_1 és x_2 független véletlen valós szám [0,1] intervallumon. Z_1 változó a komplex zaj valós részét, Z_2 pedig a képzetes részét adja a kimeneti komplex zaj vektorban. A programrésznek egyetlen bemenete van, az f_s mintavételi frekvencia, a program innen tudja milyen hosszú zaj tömböt kell létrehoznia. A főprogramban a jel-zaj viszonyt (SNR) dB-ben lehet megadni így azt decibel skáláról át kell konvertálni, hogy a zaj teljesítményét változtathassuk a jel teljesítményéhez képest.

3.2. A vevő megvalósítása

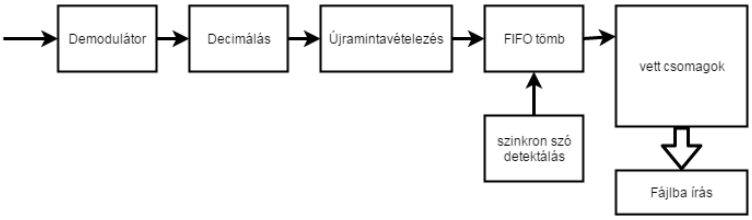
A vevő két fontosabb alprogramból áll, egyik a már demodulált bitekből állítja elő a szöveges hexa string formátumot, amit lehet komparálni a küldött adatsorozattal, másik maga a demodulátor ami a GMSK modulált IQ mintákból biteket állít elő. Demodulátorból több fajta megvalósítására került sor, egyik típus a késleltetős FM demodulátor, másik az illesztett szűrős FM demodulátor. Egy harmadik alprogram számolja a bit hiba arányt (BER) és a csomag hiba arányt (PER), hogy a megvalósított vevő jóságát lehessen behatárolni. A vevő blokkdiagramja az alábbi ábrán látható (**??**. ábra.):



3.8. ábra. *A Gauss-i ablak képe.*



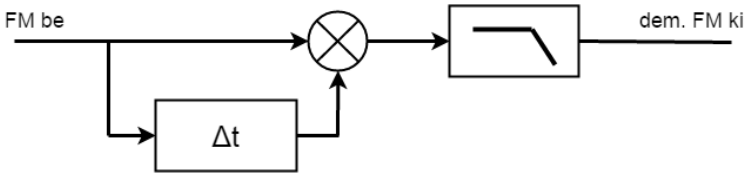
3.9. ábra. *A GMSK moduláció spektrumképe.*



3.10. ábra. *A megvalósított vevő blokkdiagramja.*

3.2.1. Koincidencia (késleltetős) demodulátor

A két fajta demodulátor közül az egyik a koincidencia demodulátor [2], aminek a működése azon alapul, hogy két azonos frekvenciájú szinuszos jel szorzata, függ a köztük lévő fáziskülönbségtől. A bejövő frekvenciamodulált jelet és annak 90°-os fáziskésleltetjét egy szorzóra vezetjük, aluláteresztő szűrőt követően pedig megjelenik a moduláló jellel arányos jel(3.11. ábra).



3.11. ábra. *A késleltetős demodulátor blokkdiagramja.*

A demodulálás után, még itt a demodulátor blokkon belül valósul meg a decimálás és az inkrementálás. Ebben a részben $f_s/datarate$-el csökken a mintavételezés, majd háromszoros inkrementálás történik, hogy a minták alapján többségi döntéssel, ami kemény döntésnek számít, lehessen dönteni az adott bitekre.

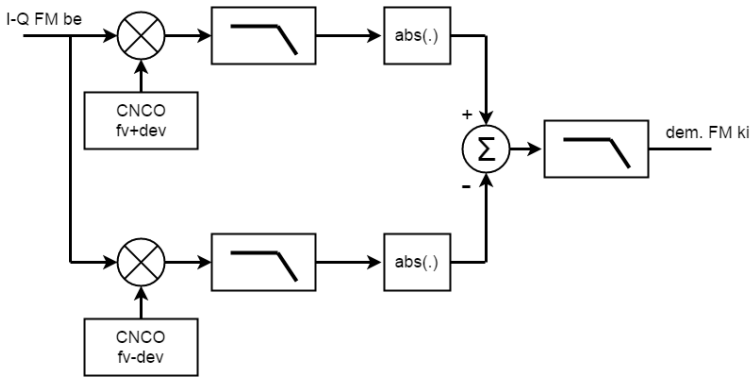
3.2.2. Illesztett szűrős demodulátor

A másik fajta demodulátor az illesztett szűrős demodulátor, amely blokkdiagramja látható a 3.12. ábrán. A bejövő IQ mintákat egyszer lekeverjük $f_v + dev$ azaz a vivőfrekvencia + lökettel, illetve lekeverjük $f_v - dev$ azaz a vivőfrekvencia - lökettel. Mindkét ágon aluláteresztő szűrőt használunk, hogy csak a számunkra hasznos keverési termék maradjon meg, majd abszolút értéket képzünk, és kivonjuk egymásból a két ágat. Végül szintén szűrőfokozat következik.

A szimuláció során nem használjunk vivőfrekvenciát, ami így nulla értékű, a CNCO-t (Complex Numeric Controlled Oscillator) a frekvencialöket értékére kell beállítani a következő módon:

$$\exp(-j2\pi\frac{dev}{f_s}i) \qquad\qquad i = 0 \ldots N \tag{3.10}$$

Ahol N az IQ minták számát jelenti. Következésképp a programrésznek három bemenete van, maguk a demodulálandó IQ minták, f_s mintavételi frekvencia, illetve az adatsebesség (datarate), amiből közvetlenül számolható a frekvencialöket. A CNCO implemetálása után



3.12. ábra. *Az illesztett szűrős demodulátor blokkdiagramja.*

következik az aluláteresztő szűrő ami egy IIR (infinite impulse response) szűrő, egy egyszerű egytárolós RC tagot valósít meg. A szűrő kimenete:

$$y[i] = \frac{1}{1+\frac{f_s}{2\pi f_0}}x[i] + \frac{\frac{f_s}{2\pi f_0}}{1+\frac{f_s}{2\pi f_0}}y[i-1] \tag{3.11}$$

Ahol a szűrő töréspontiﬀrekvenciája f_0, és érdemes úgy beállítani az értékét, hogy 1,2...1,5 -szerese legyen a löketnek, hogy ezen a frekvencián még ne csillapítson. Mind a három esetben ugyanezt a szűrő konstrukciót használtam, ugyanazokkal a paraméterekkel. Ugyancsak itt a demodulátor blokkon belül valósul meg a decimálás és az inkrementálás ugyanúgy, ahogy a késleltetős demodulátor esetében is.

3.2.3. Döntő

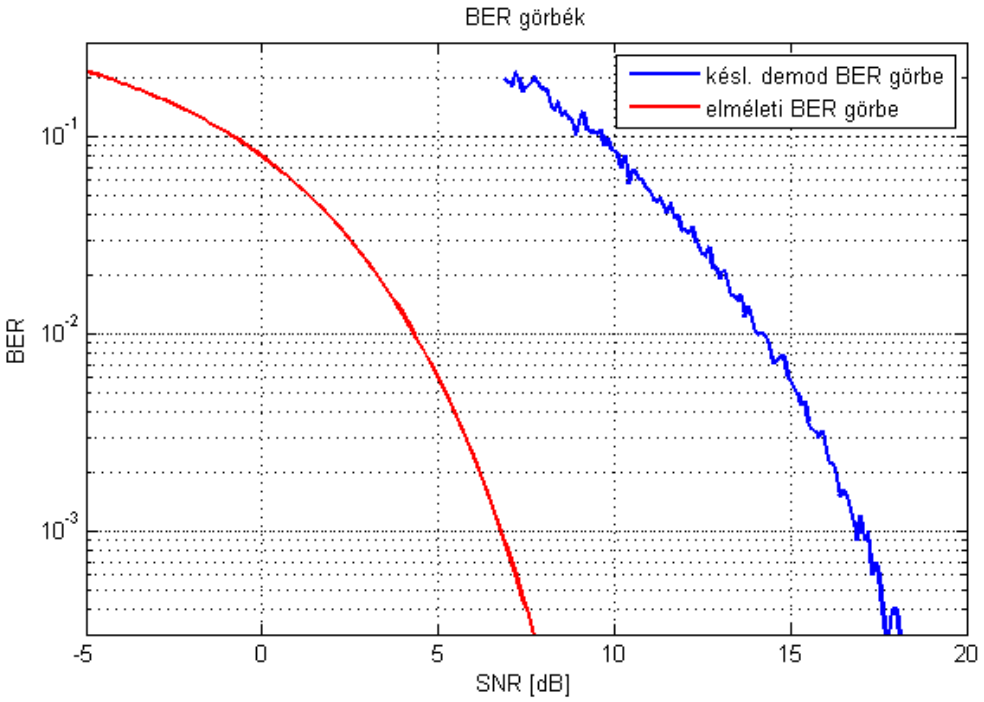
A Döntő az a programrésze a vevőnek, ahol a demodulátorból érkező bitenkénti három mintából kemény döntéssel döntünk valamely bitre. A vevő rátalál az adatcsomagokra, a szinkron szó alapján, majd ha talált egy csomagot azt hexa formátumú stringgé alakítja és egy adattömbben átadja a kimenetre. Kicsit részletesebben, a program állandóan tölt egy tömböt a beérkező +1 ,-1 adatokkal aminek a mérete $3 \cdot (syncbitek) + 3 \cdot 8 \cdot (bajtszam)$, közben a tömb elejét folyamatosan hasonlítja a szinkron szóhoz. Teljes egyezés akkor lenne ha a szorzat összege elérné a szinkron szó hosszát, azaz 2DD4 esetén a 48-at. Persze ez a helyzet csak kiváló jelzaj viszony esetén léphet fel, ezért jóval megengedőbbnek kell lenni, ha a szorzat összege eléri a 36-ot akkor úgy vesszük csomagot detektálunk. Ez esetben elvégezzük a többségi döntést, majd 0, 1 bitsorozatot hozunk létre, majd nyolcasával összefogva a biteket megtörténik a stringgé alakítás.

3.2.4. BER illetve PER számítása

A BER alprogramom belül kerül összehasonlításra az adott és vett hexa stringek. Történik egy string boolean konverzió, majd rávizsgálunk a két tömb egyezésére, ezt összegezzük és elosztjuk az elemszámmal, majd a hányadost kivonjuk 1-ből.

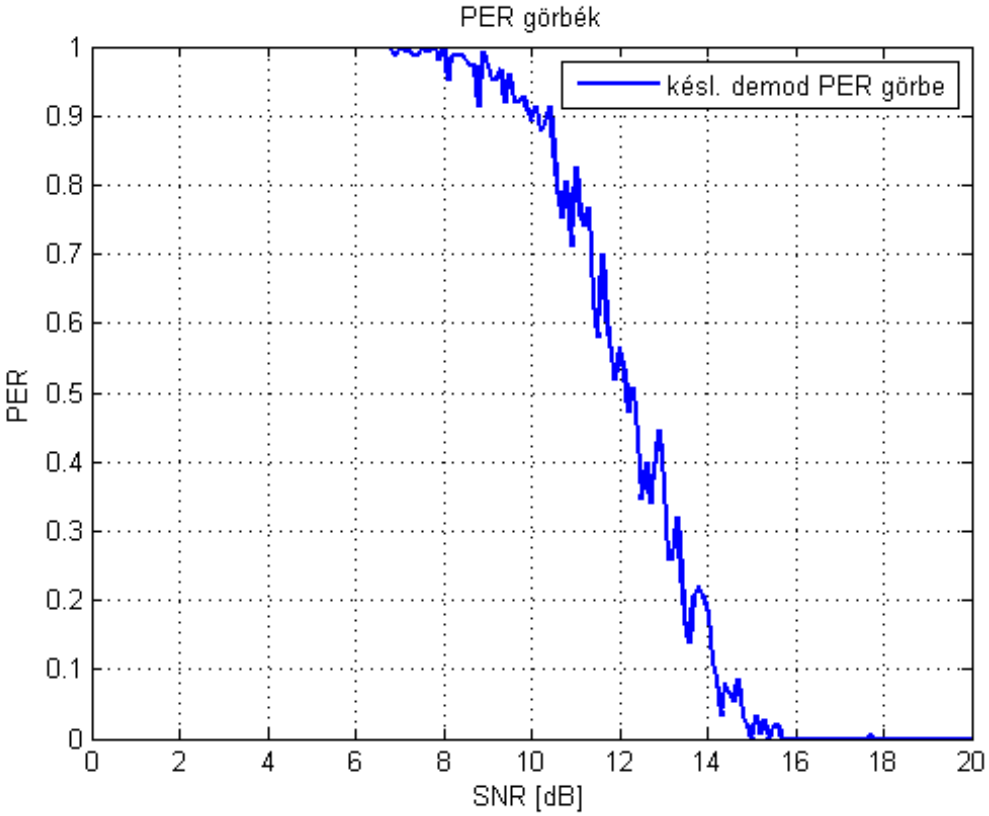
A főprogramrészben az adó úgy van beállítva, hogy egymás után 10-szer adja ugyanazt

a bitsorozatot, de van olyan helyzet, hogy a vevő több csomagot detektál mint amennyit az adó adott. Ez azért lehetséges, mert az adatban is megtalálható a szinkron szó, vagy ahogy hasonló bitsorozat, de ebben az esetben a kapott bithiba arány közel lesz a 0,5-höz, így kiszűrhető. A 0,25-nél nagyobb bithiba arányú csomagokat a szimuláció nem veszi figyelembe. A program BER illetve PER (csomaghiba arányt) számol, a jelzaj viszony függvényében, illetve a névlegestől eltérő frekvencia függvényében is, hogy kiderüljön a vevő, illetve az egyes demodulátor típusok mennyire érzékenyek a frekvencia elhangolódásra.

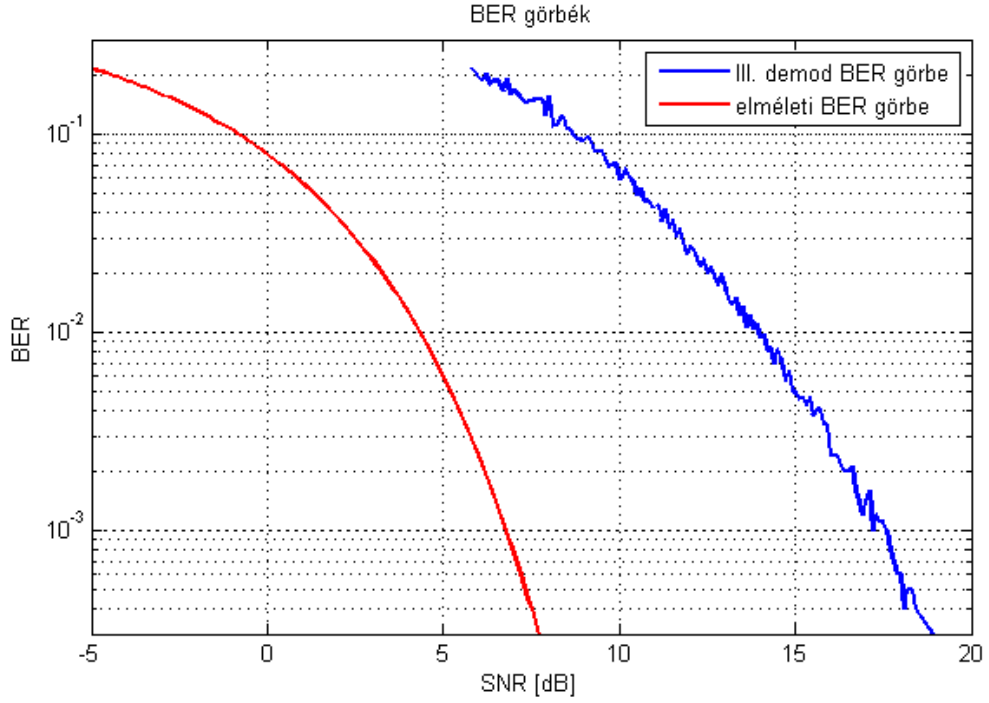


3.13. ábra. *A késleltetős demodulátor BER görbéje, és az elméleti határ.*

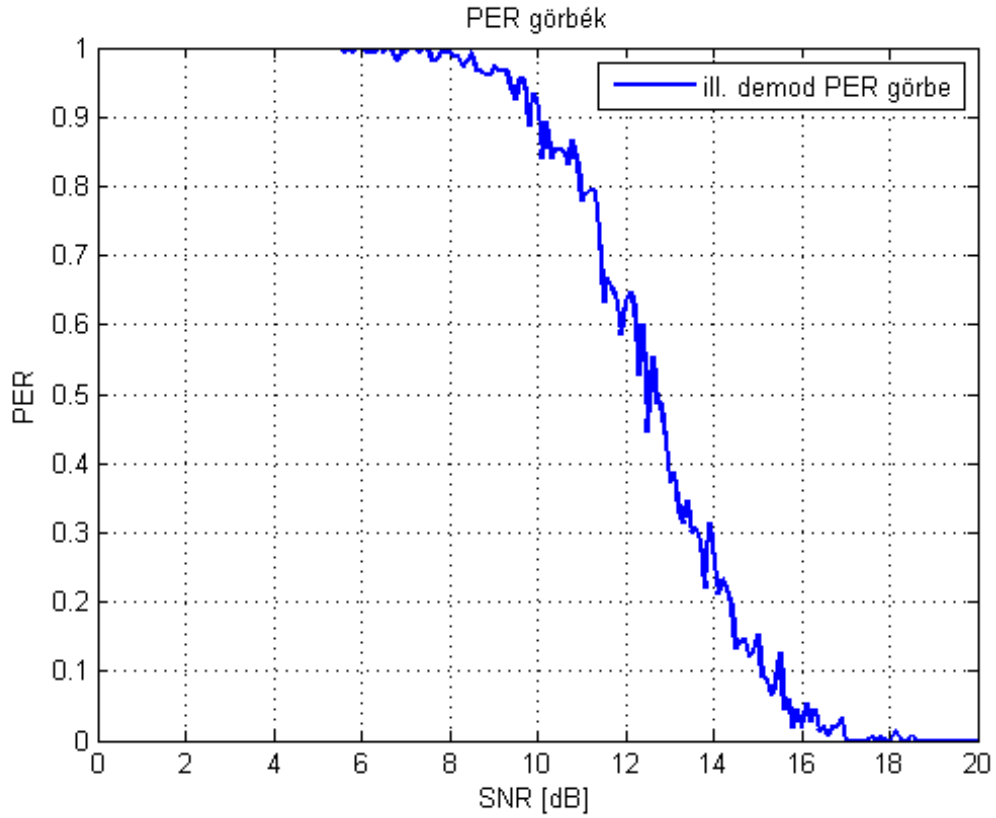
A késleltetős demodulátor bithiba arány görbéje összehasonlítható az elméleti határral (3.13. ábra.), eléggé elmarad tőle, nagyjából 10-12 dB-el, ezen különböző csatornakódolási eljárások tudnak segíteni, hogy közelebb kerüljünk az elméleti határhoz. Külön meg kell jegyeznem a szimulált BER görbe el van tolva 13 dB-el, hiszen a túlmintavételezésből adódóan lesz demodulációs nyeresége a vevőnek, 20-szoros túlmintavételezés esetén ezzel a 13 dB-el korrigálni kell a szimulált görbét. Az ábrák már a korrigált verziót tartalmazzák, mind BER mint PER esetén.



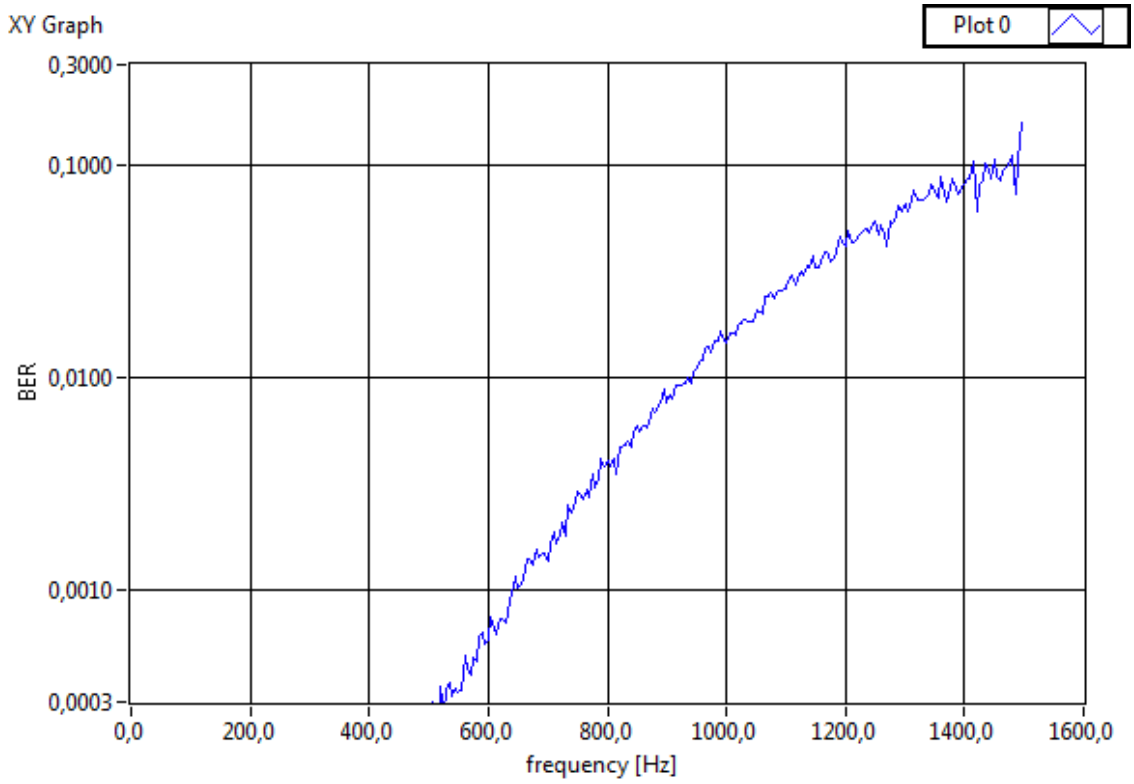
3.14. ábra. *A késleltetős demodulátor PER görbéje.*



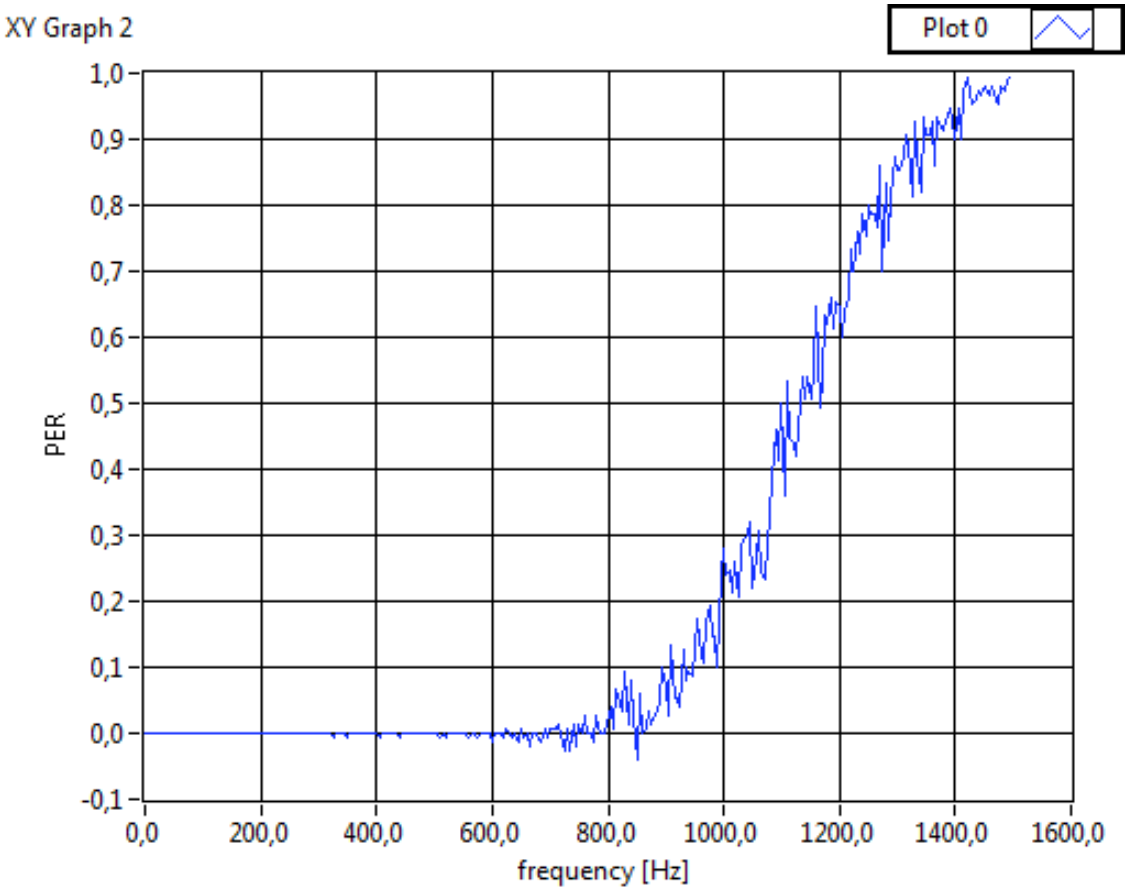
3.15. ábra. *Az illesztett szűrős demodulátor BER görbéje, és az elméleti határ.*



3.16. ábra. *Az illesztett szűrős demodulátor BER görbéje, és az elméleti határ.*



3.17. ábra. *Az illesztett szűrős demodulátor BER görbéje a frekvencia függvényében.*



3.18. ábra. *Az illesztett szűrős demodulátor PER görbéje a frekvencia függvényében.*

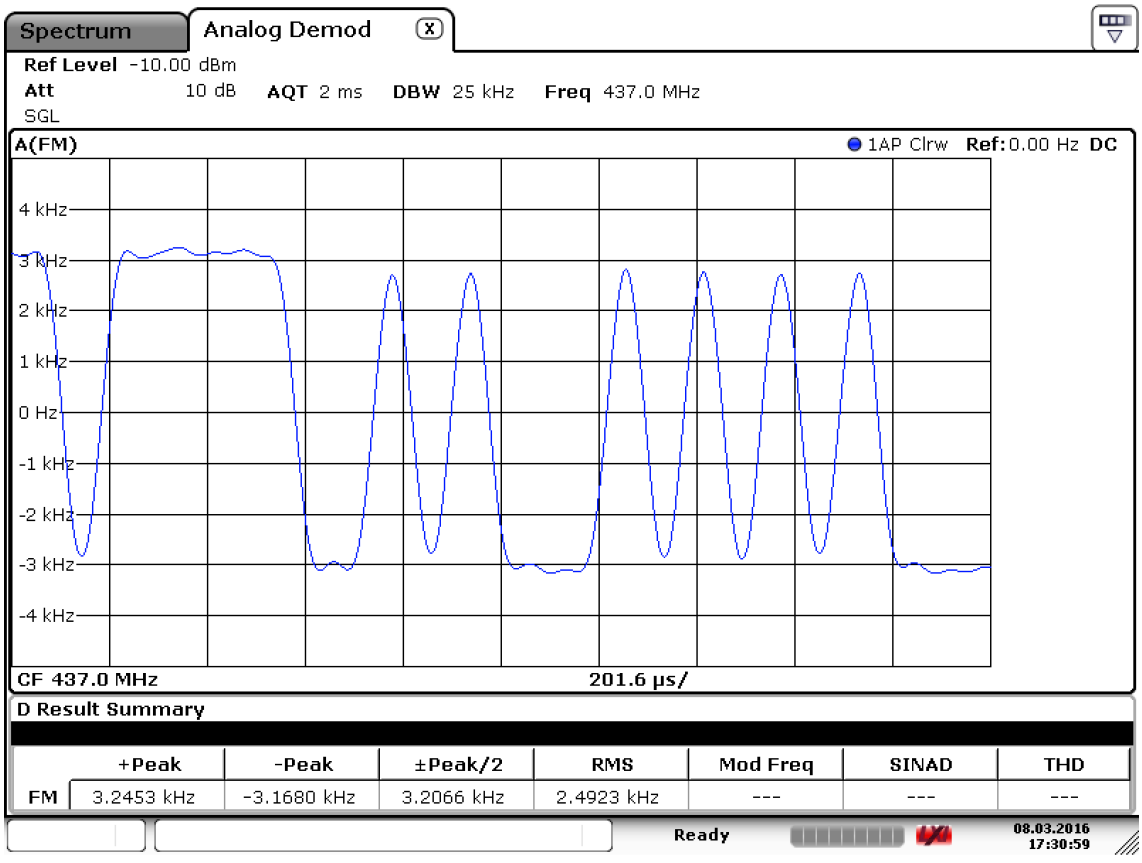
4. fejezet

PXI rendszeren kifejlesztett szoftverrádió

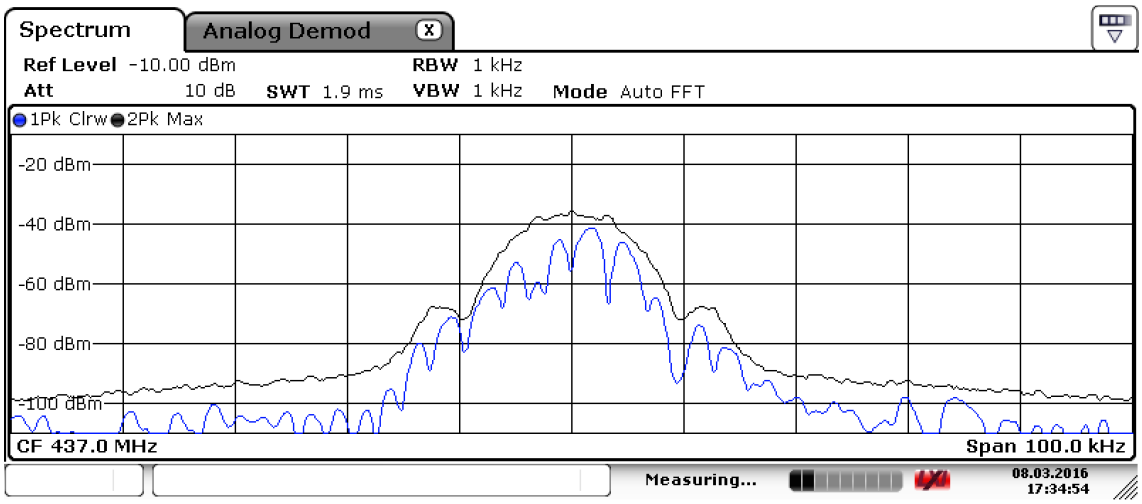
4.1. Az adó

4.1.1. USRP

4.1.2. Mérési eredmények



4.1. ábra. *Az USRP-ről küldött bitek demodulálása a spektrumanalizátorral.*



4.2. ábra. *Az USRP-ről küldött GMSK jel spektruma.*

4.1.3. PXI GEN platform

4.1.4. Mérési eredmények

4.2. A vevő

4.2.1. USRP

4.2.2. PXI 5772 platform

4.2.3. Mérési eredmények

5. fejezet

A műholdpályából adódó anomáliák

5.1. AGC (Automatic Gain Control) megvalósítása

5.2. AFC (Automatic Frequency Control) megvalósítása

5.3. A vevőkészülék optimalizálása a PXI platform beépített FPGA-ja segítségével

Köszönetnyilvánítás

Ábrák jegyzéke

Táblázatok jegyzéke

Irodalomjegyzék

[1] https://en.wikipedia.org/wiki/Window_function Hozzáférve: 2016.05.19

[2] http://staff.kzs.hu/ml/fm_demodul%C3%A1torok.htm Hozzáférve: 2016.05.19