Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Egyed-Izsák Katalin

SPEKTRUM MONITORING ÉS MODULÁCIÓ ANALIZÁTOR ESZKÖZ FEJLESZTÉSE

TANSZÉKI TÉMAVEZETŐ

Dr. Elek Kálmán

BONN TÉMAVEZETŐ

Mikó Gyula

BUDAPEST, 2017

Tartalomjegyzék

1 Bevezetés

A téma keretében olyan eljárások kifejlesztése volt a cél, melyek a korszerű vezeték nélküli távközlési szabványok szerint folytatott kommunikáció felismerésére, valamint alapvető rádiós paramétereinek automatikus meghatározására alkalmasak. A megvalósított spektrumanalizátor segítségével szeretnénk majd a környezetünkben előforduló jelek spektrumát monitorozni, majd a képernyőn ezeket megjelenítve becslést adni arra, milyen jelről is lehet szó. Ez a különböző szabványok ismeretében lehetséges, más az alakja ugyanis egy LTE, egy WiFi vagy például egy Bluetooth jelnek.

Az eszköz sikeres megvalósítása a továbbiakban drónok működési frekvenciájának kiszűrésére is alkalmas lehet. Miért is van erre szükség? A repülőterek környékén megjelenő drónok nagyon veszélyesek a légi közlekedésre, ezáltal felbukkanásuk bizonytalan ideig tartó szolgáltatás kiesést jelent, lévén, hogy a repülőgépek addig nem szállhatnak fel, míg meg nem találták és el nem távolították ezt a drónt a helyszínről. Ezen zavaró eszközök felderítése igen fontos lenne tehát.

A feladat megvalósítása Linux alapú környezetben történt, programozási nyelvként pedig Pythont használtam. Mivel mindkét környezet számomra eddig teljesen ismeretlen volt, így a félév első része ezen környezetek megismerésével zajlott. Fontos szempont volt még, hogy a szoftverfejlesztés során csak nyílt forráskódú szoftverek felhasználása volt megengedett.

Beszámolómban először a feladat megoldásához szükséges elméleti hátteret fogom ismertetni. Ezután bemutatom, hogyan is épül fel általánosságban egy szoftverrádió, majd rátérek az általam is használt hardver felépítésére. Ezután szót ejtek a téma kidolgozása során használt szoftverekre, majd pedig vázolom, mit is sikerült önállóan véghezvinnem a félév során.

2 Elméleti áttekintés

Jelfeldolgozás során a jel időtartománybeli leírása helyett sokkal célszerűbb annak frekvenciatartománybeli képéből kiindulni. Ugyan az időtartománybeli jel használata egyszerűbb és szemléletesebb (például oszcilloszkóp képernyőjén megjelenő jel esetén), ám valós jelek esetén sajnos nagyon nehezen tudjuk visszafejteni az eredeti jelet. Amikor egy rendszer bemenő jelre adott válaszát keressük, időtartományban konvolúcióval juthatunk el a megoldáshoz. Frekvenciatartományban azonban jóval egyszerűbb dolgunk van, ezért is térek át most erre. A spektrumképből könnyen meg lehet állapítani, eredetileg milyen jelről is lehetett szó, e képből több hasznos információ nyerhető. Sajnos azonban előfordul, hogy a véges megfigyelési időablak következtében a számítás eredményeként olyan frekvenciára is kerül spektrum komponens, ami az eredeti jelben nem szerepelt. Ez a jelenség a spektrumszivárgás vagy aliasing, ami igen megnehezítheti az eredeti jel dekódolását.

2.1 DFT

A DFT (Discrete Fourier Transform), vagyis a véges sorozatok Fourier transzformációja.

Tekintsük az N elemből álló $x(nT)$ diszkrét idejű sorozatot ($n = 0,1,2,\ldots(N\text{-}1)$)!

Ennek a sorozatnak a spektruma (DFT-je):

$$X_k = DFT\{x(nt)\} = \mathbf{F}\{x(nT)\} = \sum_{n=0}^{N-1} x_n e^{-j2\pi\frac{kn}{N}} \quad k = 0,1,2,\ldots,(N-1) \qquad (2.1.)$$

Az inverz transzformáció:

$$x_n = IDFT\{X_k\} = \mathbf{F}^{-1}\{X_k\} = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{j2\pi\frac{kn}{N}} \quad n = 0,1,2,\ldots,(N-1) \qquad (2.2.)$$

Jelek spektrumának becslésére a DFT-n alapuló eljárások a leggyakoriabbak. Egy megvalósítható algoritmus természetesen véges számú mintából számol. Ha N-et 2-nek egész számú hatványaként választjuk meg (pl.: 2^{10}=1024), akkor egy igen hatékony számítási algoritmust használhatunk a (2.1.)-nek a kiszámításához. Ez az eljárás az u.n. gyors Fourier transzformáció (Fast Fourier Transform FFT). [1]

2.2 FFT

Azaz gyors Fourier transzformáció (Fast Fourier Transform, FFT).

Az FFT a diszkrét idejű Fourier transzformáció (DFT) kiszámításának egy hatékony algoritmusa. Ha f_n egy (általában komplex értékű) idősorozat, ahol n=0,1,2,…(N-1), akkor ennek DFT-je, (azaz a spektrum):

$$F_k = \sum_{n=0}^{N-1} f_n e^{-j\frac{2\pi}{N}nk} = \sum_{n=0}^{N-1} f_n W_N^{nk} \qquad k = 0,1,2,\ldots,(N\text{-}1) \qquad (2.3.)$$

Ahol: $W_N = e^{-j\frac{2\pi}{N}}$ (2.4.)

Az algoritmus akkor lesz hatékony, ha N-et a 2-nek egész számú hatványaként választjuk meg.

A (2.3.)-szerinti összegzést bontsuk fel két részre, mely részekben a páros ($2n$), ill. a páratlan ($2n$+1) sorszámú mintákat használjuk csak fel:

$$F_k = \sum_{n=0}^{N-1} f_n W_N^{nk} = \sum_{n=0}^{\frac{N}{2}-1} f_{2n} W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} f_{2n+1} W_N^{(2n+1)k} =$$

$$F_k = \sum_{n=0}^{(N/2)-1} g_n W_{(N/2)}^{nk} + W_N^k \sum_{n=0}^{(N/2)-1} h_n W_{(N/2)}^{nk} = G_k + W_N^k H_k \qquad (2.5.)$$

A (2.5.)-ban G_k és H_k mennyiségeket felfoghatjuk úgy, mint az N/2 hosszúságú g_n és h_n sorozatok DFT-jét, ahol:

$$g_n = f_{2n} \qquad h_n = f_{2n+1} \qquad n = 0,1,2,\ldots,(N/2\text{-}1) \qquad (2.6.)$$

A G_k és H_k kiszámításánál a k index a k =(0,1,2,…,(N/2-1)) értékeket veheti fel, míg a (2.3.) kiszámításához az indexnek a k =(0,1,2,…,(N-1)) tartományon kell futnia. Ezért az (N/2) ill. az ennél nagyobb index értékekre a (2.3.)-et írjuk fel az alábbi alakban:

$$F_{k+N/2} = \sum_{n=0}^{\frac{N}{2}-1} g_n W_{(N/2)}^{n(k+\frac{N}{2})} + W_N^{k+\frac{N}{2}} \sum_{n=0}^{\frac{N}{2}-1} h_n W_{(N/2)}^{n(k+\frac{N}{2})} \qquad (2.7.)$$

ahol: k =(0,1,2,…,(N/2-1))

A (2.7.) összefüggésben szereplő tényezők egyszerűbben is kifejezhetők:

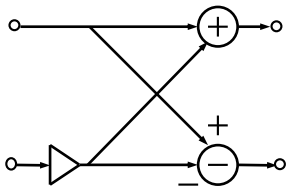
$$W_N^{k+N/2} = e^{-j\frac{2\pi}{N}(k+\frac{N}{2})} = e^{-j\frac{2\pi}{N}k} e^{-j\pi} = -e^{-j\frac{2\pi}{N}k} = -W_N^k \qquad (2.8.)$$

$$W_{(N/2)}^{k+N/2} = e^{-j\frac{2\pi}{N}2(k+\frac{N}{2})} = e^{-j\frac{2\pi}{N}2k} e^{-j2\pi} = e^{-j\frac{2\pi}{N}2k} = W_{(N/2)}^k \qquad (2.9)$$

Ezzel a spektrum felső fele:

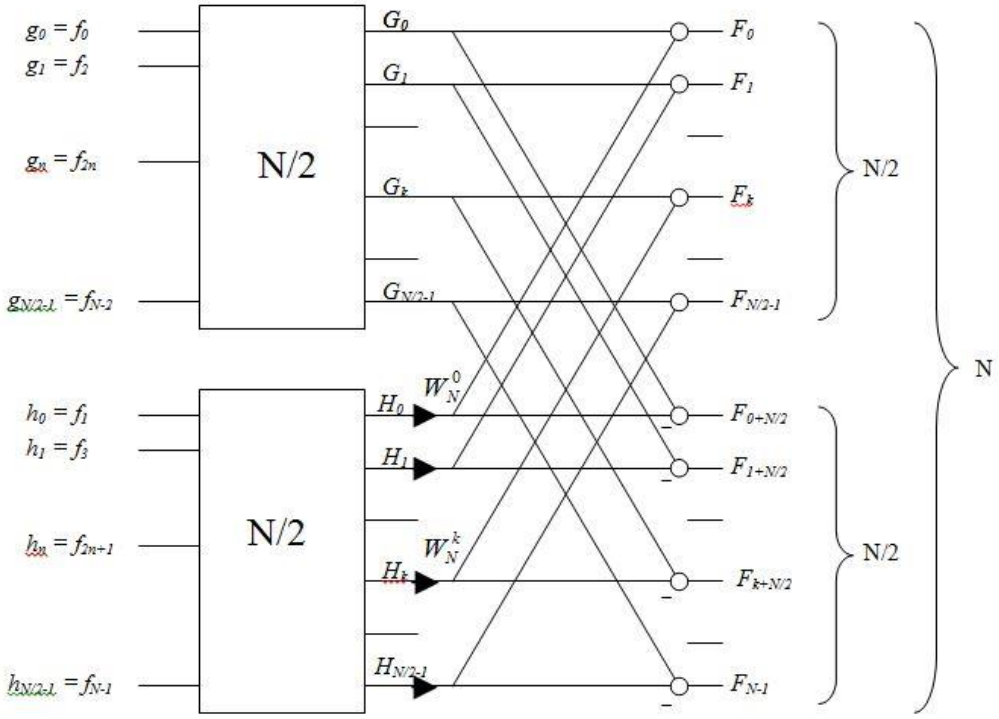
$$F_{k+N/2} = \sum_{n=0}^{\frac{N}{2}-1} g_n W_{N/2}^{nk} - W_N^k \sum_{n=0}^{\frac{N}{2}-1} h_n W_{N/2}^{nk} = G_k - W_N^k H_k \qquad (2.10.)$$

A (2.5.) és a (2.10.) összefüggéseket egy jelfolyam gráfban ábrázolva kapjuk az u.n. "pillangót" (butterfly):



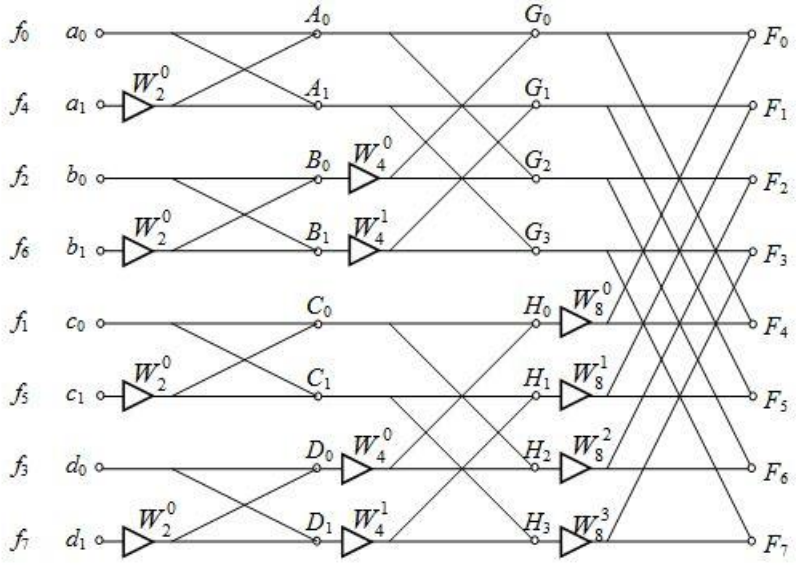
1. ábra A pillangó

Ezzel az átalakítással az *N*-pontos DFT kiszámítását visszavezettük két (*N*/2) fokszámú DFT kiszámítására.



2. ábra AzN-pontos DFT visszavezetésekét (N/2)-pontos DFT kiszámítására.

Természetesen a fokszám redukálásos eljárás tovább folytatható, mivel az *N* a 2-nek egész számú hatványa. Az alábbi ábra egy *N*=8 pontos DFT teljes kifejtését mutatja:



3. ábra AzN=8 pontos FFT kiszámításának folyamatábrája

Mint az ábrából látható minden egyes fokozatban $N/2$ darab komplex szorzást, összeadást ill. kivonást kell elvégezni, míg a fokozatok száma $log_2 N$. A műveletek összes száma:

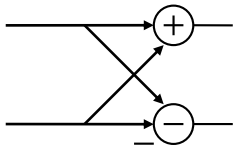
$$sz = \frac{N}{2} \log_2 N \qquad (2.11.)$$

Ha a DFT-t a (2.3.) szerint számítanánk ki, akkor N^2 komplex szorzást kellene elvégeznünk. A megtakarítás nagy N-eknél jelentős, pld. N=1024 esetén az FFT művelet igénye 5120, míg a közvetlen számolással $1024^2 \sim 10^6$ adódik.

További egyszerűsítést jelent a 0-s indexű pillangók számításánál a:

$$W_N^0 = e^{-j0} = 1 \qquad (2.12.)$$

azonosság használata. [2]



4. ábra Azelső fokozat pillangói

3 Szoftverrádió

A szoftverrádió (Software-defined Radio - SDR) alatt olyan rádiókommunikációs eszközt értünk, amely szoftveresen, akár programozhatóan valósít meg egy PC-n vagy beágyazott rendszer segítségével olyan fizikai rétegbeli RF funkciókat, melyeket egyébként hardveresen szoktak hagyományos implementálni. Ilyen funkciók lehetnek az erősítők, szűrők, modulátorok/demodulátorok, detektorok, kódolók/dekódolók stb. [3] Lényege, hogy az analóg jelet, ami a zavarokra sokkal érzékenyebb, minél előbb digitálissá alakítsuk. A cél természetesen az lenne, hogy a digitális fokozatok a lehető legközelebb kerüljenek az antennához, leváltva ezzel a kevésbé zavartűrő analóg fokozatokat.

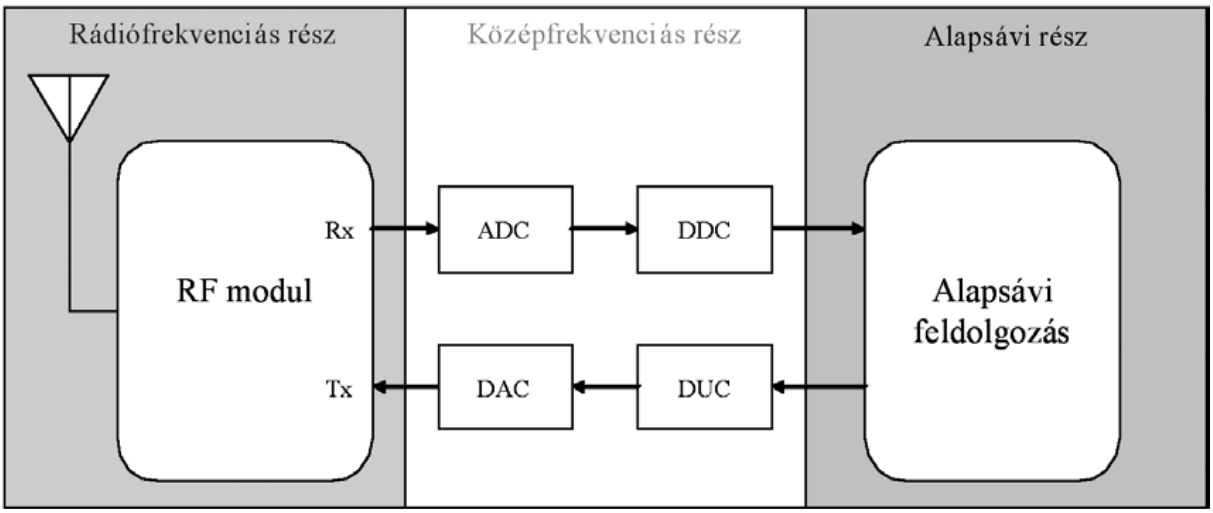
3.1 A szoftverrádió általános felépítése

A rádiófrekvenciás modul az antennából csatolókon keresztül veszi (illetve adja) a rádiófrekvenciás jelet, és azt demodulálja középfrekvenciára. Vevőoldalon az RF modul analóg rádiófrekvenciás erősítést és RF-KF keverést végez, míg adóoldalon először KF jelet RF-ra transzformálja, majd RF tartományban teljesítményerősítést végez.

Az ADC/DAC blokkok analóg-digitális átalakítók. A vevőoldalon analóg jelből digitális jelet, adóoldalon digitális jelből analóg jelet hoznak létre. Ezen blokkok határolják a rendszer analóg és a digitális részeit. A DUC illetve DDC blokkok modulációt végeznek az adó, demodulációt a vevő oldalon (digitális tunereknek is hívjuk őket). Az alapsávi rész ennek megfelelően alapsávi műveletekért felelős: kapcsolatfelvétel, teljesítménykiegyenlítés, frekvenciaugratás, időzítés, korreláció stb. Ebben a részben van megvalósítva az adatkapcsolati protokoll is.

A DDC/DUC blokkok, illetve az alapsávi feldolgozás igen nagy számítási kapacitást igényel, ezért hagyományos digitális eszközökben ASIC eszközöket használnak erre a célra. Az SDR rendszerekben mind az alapsávi feldolgozás, mind a DDC/DUC modulok megvalósítása programozható eszközök felhasználásával történik.

A programozhatóság kiterjeszthető lenne az RF részre is (pl. ha az analóg-digitális konverziót már az antennában megoldanánk). Ilyenkor az RF frekvenciasávok programozható váltását is meg kellene oldani. Ilyen széles frekvenciasávban történő átalakításra még a jelenleg legfejlettebb ADC/DAC eszközök sem alkalmasak. [4]

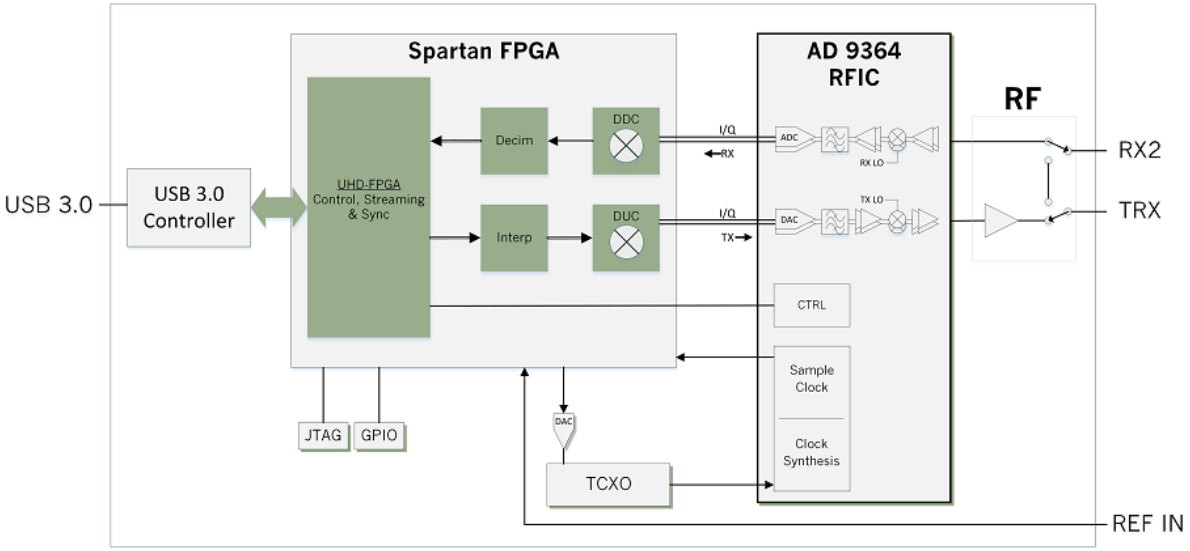


5. ábra szoftverrádió felépítése

4 USRP felépítése

Az általam használt eszköz, az Ettus Research által gyártott USRP B205mini az alábbi tulajdonságokkal rendelkezik:

- 1 TX, 1RX és egy referencia órajel csatlakozó
- 70 MHz-től 6GHz-ig terjedő szélessávú spektrum
- Xilinx Spartan-6 XC6SLX150 FPGA
- Analog Devices AD9364 RFIC adóvevő 56 MHz-es sávszélességgel
- a hoszt számítógéphez USB3.0-s csatlakozón keresztül tud kapcsolódni
- GPIO és JTAG csatlakozó is található rajta
- 10 MHz-s referencia órajellel szinkronizálható



6. ábra USRP B205 mini felépítése [5]

5 Felhasznált szoftverek

5.1 UHD

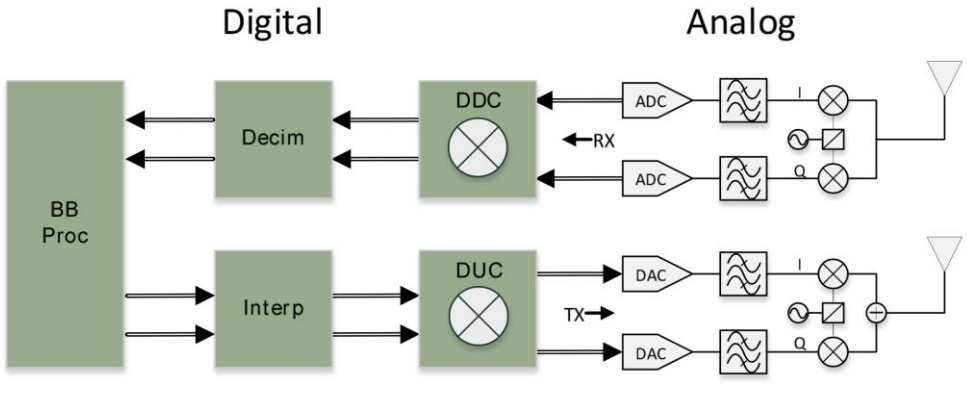
Az USRP Hardware Driver (UHD) egy olyan nyílt forráskódú program, mellyel az összes, USRP-k családjába tartozó eszköz vezérelhető. Az UHD segítségével tudunk az USRP-re különböző jelalakokat küldeni, illetve az USRP-ről vett jelalakokat detektálni. Ezen program alkalmazásával tudjuk például a rádió mintavételi frekvenciáját, középfrekvenciáját, erősítését, stb. is állítani.

Az UHD néhány további hasznos tulajdonsága: az RF frekvencia hangolható, a mintavételezés állítható, illetve tartalmaz IQ jelalak korrekciót is. Ezen funkciók a későbbiekben lesznek hasznosak a feladat szempontjából.

A mintavételi frekvencia állíthatósága például azért lesz majd hasznos számunkra, mert előfordul, hogy egy 100 MHz széles tartományt szeretnénk vételezni, ám a számítógép a valóságban csak egy 20 MHz széles spektrum vételezését teszi lehetővé.

IQ jelalak korrekcióra azért van szükség, mivel valós számokat mintavételezünk.

Emlékeztetőül: $X_k = I_k + jQ_k$, ahol az $[I_k, Q_k]$ számok a digitális modulációs eljárásoknál szokásos, a k-ik csatornában átvinni kívánt szimbólumhoz rendelt konstellációs pont koordinátáit jelentik.



7. ábra IQ jelek [6]

5.2 GNU Radio

Ingyenes, nyílt forráskódú szoftver, melyben a jelfeldolgozás előre kialakított, vagy akár általunk megvalósított blokkok segítségével hajtható végre. A programot önmagában használva különböző folyamatokat szimulálhatunk, ám ha hardvert is teszünk mellé, könnyen kezelhető szoftverrádiót alakíthatunk ki magunknak.

5.3 Különböző kiegészítő csomagok Python-hoz

A NumPy és a Scipy olyan nyílt forráskódú kiegészítők a Python programozási nyelvhez, melyeket különböző tudományos számítások elvégzéséhez használhatunk. Ezek segítségével például a többdimenziós tömbök és mátrixok könnyen kezelhetővé válnak. Alkalmazhatók továbbá az FFT algoritmus elvégzésére, vagyis nagyban megkönnyítik a jelfeldolgozási folyamatot.

A Matplotlib egy beépített könyvtár ábrák készítéséhez, matplotlib.pyplot egy függvénnyel tudunk különböző függvényeket egyszerűen kirajzoltatni.

A Tkinter egy grafikus könyvtár, erre az egyszerűen használható és könnyen beállítható mintavételezés megadása során lesz szükség.

6 Elvégzett munka

6.1 Python kód

Első lépésként egy olyan kódot készítettem, amely az USRP segítségével mintavételez, majd FFT-t számol és a spektrumot kirajzolja a számítógép képernyőjére. Szerencsére a beépített `uhd_fft` függvény is ugyanezt valósítja meg, így könnyen össze tudtam hasonlítani a saját kódom által kirajzolt spektrumot az előre megírt kód által kapottal.

Az általam írt kódból néhány sort részletesebben is bemutatnék.

Az USRP-nek az alábbi paraméterek adhatók meg:

```
def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument("-a", "--args", default="", type=str)
    parser.add_argument("-o", "--output-file", type=str, required=True)
    parser.add_argument("-f", "--freq", type=float, required=True)
    parser.add_argument("-r", "--rate", default=1e6, type=float)
    parser.add_argument("-d", "--duration", default=5.0, type=float)
    parser.add_argument("-c", "--channels", default=0, nargs="+", type=int)
    parser.add_argument("-g", "--gain", default=30, type=int)
    return parser.parse_args()
```

A mintavételezés e függvény meghívásával történik:

```
def main():
    args = parse_args()
    usrp = uhd.usrp.MultiUSRP(args.args)
    num_samps = int(np.ceil(args.duration*args.rate))
    if not isinstance(args.channels, list):
        args.channels = [args.channels]
    samps = usrp.recv_num_samps(num_samps, args.freq, args.rate, args.channels,
args.gain)
    with open(args.output_file, 'wb') as f:
        np.save(f, samps)
    usrp.send_waveform(samps,     args.duration,     args.freq,     args.rate,
args.channels, args.gain)
```

A mintavételezés beállításainak megadása:

```
if __name__ == "__main__":
    master = Tkinter.Tk()
    Tkinter.Label(master, text="Frequency").grid(row=0)
    Tkinter.Label(master, text="Sample rate").grid(row=1)
    Tkinter.Label(master, text="Number of sampling").grid(row=2)
```

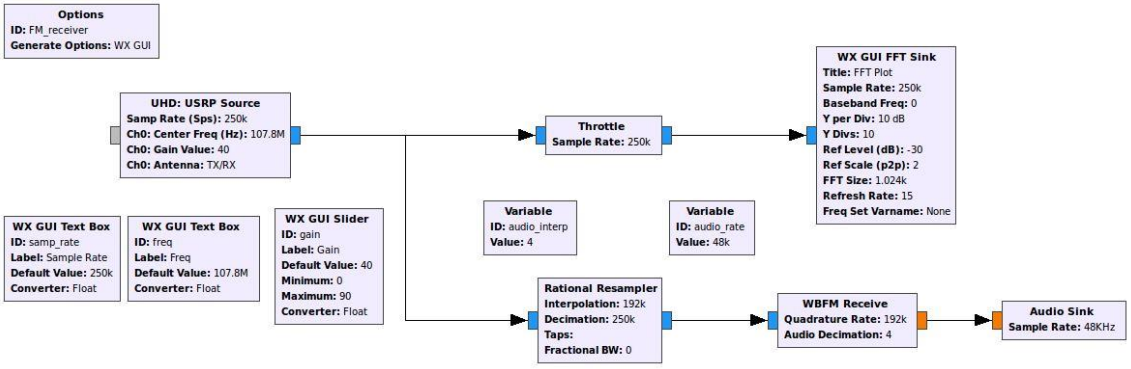
```
e1 = Tkinter.Entry(master)
e2 = Tkinter.Entry(master)
e3 = Tkinter.Entry(master)

e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
e3.grid(row=2, column=1)

e1.insert(Tkinter.END, 107800000)
e2.insert(Tkinter.END, 2e6)
e3.insert(Tkinter.END, 100)
```

6.2 GNU Radio tesztelés

GNU Radio-ban az előre definiált blokkokból megvalósítottam egy FM vevőt, ennek a felépítése látható az alábbi ábrán.

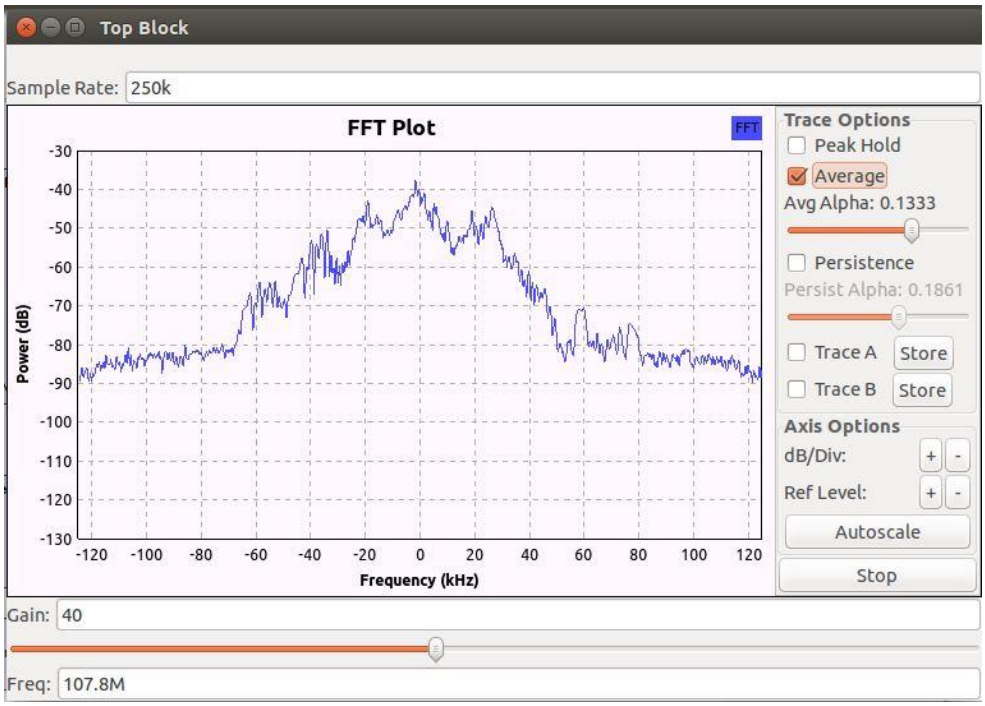
Az USRP-ről beérkező jelet egyrészt FFT-zem és megjelenítem a spektrumot a képernyőn, másrészt szélessávú vevőt alkalmazva a számítógép audio kimenetén is megjelenítem.

Az USRP-n a jó vétel érdekében a mintavételi frekvenciát 250000 Hz-re állítottam, ám ez túlságosan nagy a számítógép audio kimenetén való megjelenítéshez, így szükség van egy Rational Resampler nevű blokkra, mellyel az eredeti mintavétel átállítható.



8. ábra FM-vevő blokkok felépítése

Középfrekvenciának a 107.8 MHz-et, azaz a Kossuth Rádió budapesti adási frekvenciáját választva a lenti képen látható spektrumot kapjuk.



9. ábra Kossuth Rádió spektruma

Irodalomjegyzék

[1] http://szoftverkli.biz/Elek/BME_2017_2/02.Jelek_%C3%A9s_reprezent%C3%A1ci%C3%B3k.pdf

[2] http://szoftverkli.biz/Elek/BME_2017_2/11.Az_FFT_%C3%A9s_jelgener%C3%A1torok.pdf

[3] http://www.mcl.hu/sites/default/files/usrp%20m%C3%A9r%C3%A9si%20%C3%BAtmutat%C3%B3.pdf

[4] http://www.hiradastechnika.hu/data/upload/file/2005/2005_8/HT_0508-8.pdf

[5] https://www.ettus.com/content/files/USRP_B200mini_Data_Sheet.pdf

[6] https://kb.ettus.com/UHD (2017.12.12.)