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

Rádióátviteli mérések laboratórium 2

10. mérés
ADS-B vétel

Szilágyi Gábor NOMK01

Budapest, 2023. március 24.

1. A feladat

Erre a laboralkalomra a feladat egy C program kiegészítése, ami készen alkalmas arra, hogy valós időben egy szoftverrádióból érkező mintákból dekódolja az ADS-B adásokat. Tanszéki számítógépes terem hiányában konzerv fájlokból kellett kihámoznunk az előre felvett adásokat. A feladathoz nagy segítség a rendelkezésre álló programkód-váz (E függelék), ezt kellett kiegészíteni a minták feldolgozását végző és az eredményeket kiíró résszel.

2. A megoldás lépései

2.1. OOK kódolt bitek előállítása

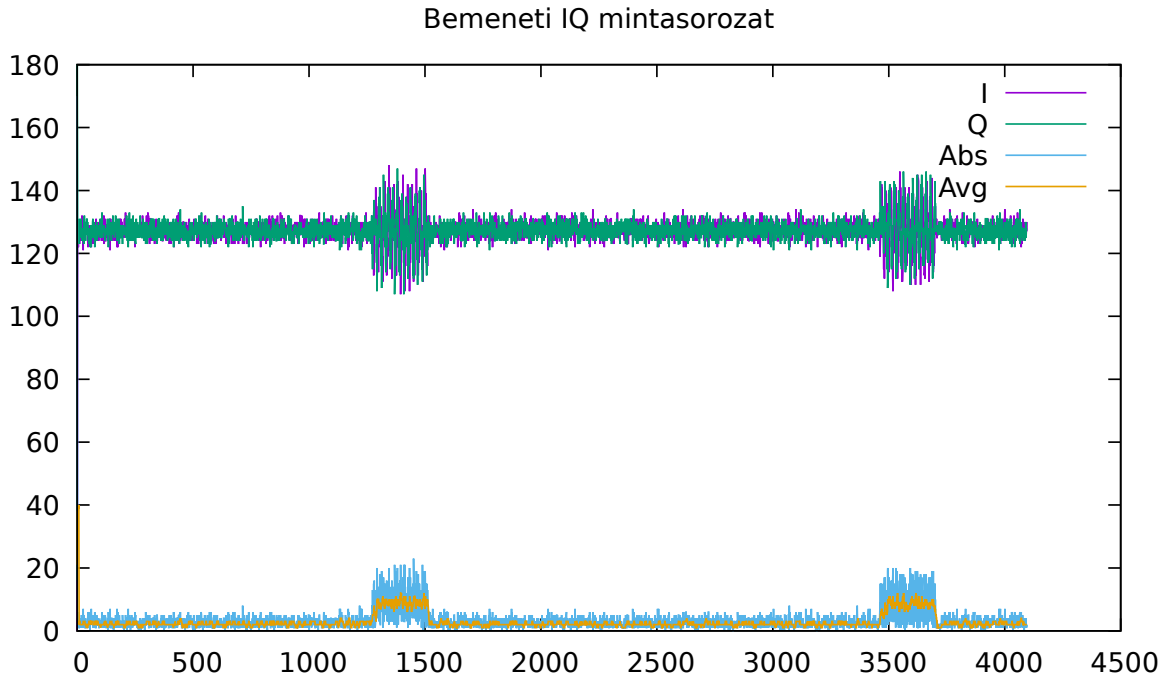
A bejövő I/Q mintákból először amplitúdót számolok. Ehhez egy előre elkészített, 128-as referenciaszinthez igazított look up table áll rendelkezésre, amiből egyszerűen ki kell olvasni az adott I és Q értékekhez tartózó amplitúdót. Erre azért van szükség, mert mintánként egyesével újraszámolni az amplitúdót a gyökvonás miatt túl költséges lenne a megadott mintavételi frekvencia mellett.

Az amplitúdókból egy FIR szűrővel mozgó átlagot állítok elő, a szűrőablak `FIR_LEN = 8` minta hosszú. Erre a lépésre azért van szükség, hogy különböző vételi teljesítményű jelek esetén is helyesen el lehessen végezni a 0/1 döntést. A FIR szűrő egy egyszerű mozgóátlag, így a késleltetése `FIR_LEN/2 = 4` minta hosszú, tehát ennyivel a legújabbnál régebbi mintára vonatkozó mozgóátlagot állít elő. Az eredeti I/Q minták („I" és „Q"), az amplitúdó („Abs") és a belőle képzett mozgóátlag („Avg") az 1. ábrán láthatóak. A szűrőhöz szükséges késleltetés egy cirkuláris buffer segítségével van megvalósítva, az átlagképzéshez pedig egy akkumulátort használok és az átlag `FIR_LEN`-szeresét tárolom, így itt sincs szükség mintánként egy osztásra, csak egy összeadásra, egy kivonásra és memóriaírásra, ill. -olvasásra.

A mozgóátlagot az OOK modulációnál adaptív döntési szintként lehet használni, ami az ON és OFF teljesítményszintek átlaga közelébe áll be, ha a 0-s és 1-es bitek gyakorisága körülbelül megegyezik. A Manchaster kódolásnál ez automatikusan teljesül. Az aktuálisan feldolgozott (4 ütemmel ez előtti) minta amplitúdóját a legutóbb kiszámolt átlagszinthez komparálom, ez alapján döntök 0-s vagy 1-es bitre. Pontosabban az aktuális LUT-ból kiolvasott amplitúdó `FIR_LEN`-szeresét számolom ki és ezt hasonlítom össze az akkumulátorban tárolt értékkel, ami az átlag `FIR_LEN`-szerese. Így ennél a lépésnél sincs szükség minden ütemben osztás műveletet végezni. Mivel a mintavételi sebesség megegyezik a bitsebsséggel, minden bejövő mintából egy bit áll elő.

2.2. Állapotgép a bitek feldolgozásához

Az aktuális bit segítségével vezérlek egy állapotgépet. Ennek az első feladata az ADS-B adás premble-jének felismerése, amely 16 bit hosszú. Itt egyszerűen egy számlálóval a preamble megadott mintázatával egyező előző biteket számolom. Ha az aktuális bit hibás, akkor nullázom a számlálót és elölről kezdem a keresést. Ha a számláló elért 16-ig, akkor ez azt jelenti, hogy megtaláltam a preamble-t, következik még 112 adatbit, ami a Manchaster-kódolás miatt 224 bitet jelent, így összesen 240-ig kell elszámolni. A Manchaster-dekódoláshoz egyszerűen a kód bitpárjaiból az első bit adja meg az adatbitet (0-tól indexelve így a páros indexű bejövő bitek).



1. ábra. A konzerv fájlból kiolvasott I/Q minták, az amplitúdó és az amplitúdó mozgóátlaga.

Az adatbitek kiírásához 8-asával bájtokba rendezem őket, majd a hexadecimális értékeket kiírom a kimenetre. A megtalált adások elejét új sorral és csillaggal jelzem. A teljes konzervfájlra a kimenet az A. függelékben olvasható. A kimenet első sorának utolsó 4 karaktere (`6186`) megegyezik a bemeneti mintákat tartalmazó fájl nevében megadott részlettel, ebből látszik, hogy a dekódolás helyes. Az ábrák generálásához a Gnuplot programot használtam, aminek az ábrát generáló szkriptje a D. függelékben látható.

A. test.txt (dekódolt adat)

```
*9e996493240f217a3f82ace46186
*904748061051831583582 0efe698
*d04748061051831583582 0eee698
*904748061051831583582 0efe698
*904748061051831587582 0cfe690
*904748061051831583582 0efe698
*4c94c8e1943999527a5e1248982d
*91228688847bdf1364818c19ce94
*904748061051831583582 0efe698
*904748049051831583582 0efe698
*904748061051831503582 0efe698
*904748061051831583582 0efe698
*8c1a553520d924286a1d5430d2c1
*9047480650518315a35820efe698
*904748061451831583582 0efe698
*904748061051931583582 0efe698
*904648061051830583583 0efe688
*904748061055831583582 0efe698
*904548061051831583582 0efe698
*904748061051831583582 0efe698
*9047480610518315a35820efe698
*d065480610518315835820efe698
*904748061051831583582 0cfe698
*904748061051831583582 0efe698
```

B. build

```
#!/bin/bash
gcc adsb.c -lm -o demod
```

C. run

```
#!/bin/sh
cat NOMK01___6186.dat | ./demod > test.txt
#cat NOMK01___6186.dat | ./demod
gnuplot plotter.gp
```

D. plotter.gp

```
set terminal pdfcairo
set output "plot.pdf"
set title "Bemeneti IQ mintasorozat"
set yr [0:180]
plot "test.txt" u 1 w l t "I", "test.txt" u 2 w l t "Q", "test.txt" u 3 w l t "Abs", "
    test.txt" u 4 w l t "Avg"
```

E. adsb.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// ADS-B signal length
```

```
#define PCKT_LEN  240

// Low Pass FIR Filter length
#define FIR_LEN  8

// input data buffer length
#define BUF_SIZE  8192

int main(int argc,char **argv)
{
    // general index variables
    int i,j,k;

    // input data buffer
    unsigned char buffer[BUF_SIZE];
    int read_len, bix;

    // ADS-B preamble pattern
    unsigned char adsb_preamble[16]={1,0,1,0,0,0,0,1,0,1,0,0,0,0,0,0};

    // store previous samples for FIR filtering
    unsigned int fifo[FIR_LEN] = {0};
    unsigned int fptr = 0;

    // Absolute value Look-up Table: I-Q --> ABS(.)
    // input data: unsigned char I and unsigned char Q
    unsigned int iq_to_abs[256][256];

    // "iq_to_abs" array initialization
    for(i=0;i<256;i++) {
        for(j=0;j<256;j++) {
            iq_to_abs[i][j] = (unsigned int)sqrt( (i-128)*(i-128) + (j-128)*(j-128) );
        }
    }

    // variables
    unsigned int abs_val;
    int accumulator = 0;
    unsigned char bit;
    unsigned int stm;
    unsigned char hex;

    i = 0;
    j = 0;

    // main loop
    do {
        read_len = fread(buffer, 1, BUF_SIZE, stdin);   // read data to input buffer
        stm = 0;
        hex = 0;
        for(bix=0; bix<read_len-1; bix+=2) {
            // convert I-Q to magnitude
            abs_val = iq_to_abs[buffer[bix]][buffer[bix+1]];

            // FIR filtering
            accumulator = accumulator-fifo[fptr]+abs_val;
            fifo[fptr] = abs_val;
            fptr = (fptr+1)%FIR_LEN;

            // Decoding
            if(fifo[(fptr-FIR_LEN/2)%FIR_LEN]*FIR_LEN > accumulator) {
                bit = 1;
            }
            else {
                bit = 0;
            }

            // ADS-B packet search and print
            // State machine
            if(stm < 16) {
                if(adsb_preamble[stm] == bit) {
                    stm++;
                }
```

```
                else {
                    stm = 0;
                }
            }
            else {
                if(stm < 240) {
                    if(stm == 16) {
                        printf("\n*");
                    }
                    // Manchaster-decode
                    if(stm>15 && stm%16==15) {
                        printf("%02x", hex);
                    }
                    if(stm%2==0) {
                        hex = hex << 1;
                        if(bit==1) {
                            hex = hex | 1;
                        }
                    }
                    stm++;
                }
                else {
                    stm = 0;
                }
            }
            //printf( "%d\t%d\t%d\t%d\n", buffer[bix], buffer[bix+1], abs_val,
    accumulator/FIR_LEN );
        }

        // uncomment if not testing
        //break;
    } while(read_len>0);
    printf("\n");
    return 0;
}
```