Feladatkiírás

Számításigényes jelfeldolgozási problémák esetén a mért jel kiértékelése nem mindig történhet a mérés helyén, mivel nem feltétlenül áll a szükséges erőforrás rendelkezésre. Ilyen feladatra jó példa egy multiharmonikus jel paramétereinek meghatározása. Ez nemlineáris optimalizálási feladat, ahol a pontos kiértékeléshez általában hosszabb mérések feldolgozására van szükség.

A feladat során egy olyan mérésadatgyűjtő rendszer tervezése a cél, mely képes az adatokat egyszerűbb jelfeldolgozás után a PC felé továbbítani, USB-n keresztül. A mérésadatgyűjtő legyen konfigurálható a PC-ről, de egyszerűbb beállításokat magán az eszközökön is lehessen elvégezni. Demonstrálja a működést egy elosztott jelfeldolgozó eljárás implementálásával, ahol a mérésadatgyűjtő előfeldolgozást végez, majd a számításigényesebb eljárások a PC-n kerülnek végrehajtásra.

A hallgató feladatának a következőkre kell kiterjednie:

- Mutasson be lehetőségeket a multiharmonikus jelek paramétereinek becslésére, hasonlítsa össze ezek statisztikai tulajdonságait és erőforrásigényét a szakirodalom alapján.
- Implementáljon egy mérésadatgyűjtő szoftvert, mely képes a mért jel előzetes feldolgozására.
- Implementálja a PC-n a magas szintű jelfeldolgozási eljárásokat.
- Végezzen mérést, és értelmezze a mérési eredményeket.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Mérésadatgyűjtő- és feldolgozó egység tervezése és implementációja periodikus jelek elosztott feldolgozásához

Diplomatervezés I. dokumentáció

Készítette:
Nagy Mihály Péter
villamosmérnök MSc
szakos hallgató

Témavezető:
Dr. Pálfi Vilmos
egyetemi adjunktus

Budapest
2019

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott Nagy Mihály Péter, szigorló hallgató kijelentem, hogy ezt a diplomatervet 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 (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített 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 hozzáférhetővé.

Kelt: Budapest, 2019. 12. 22.

...…….

Nagy Mihály Péter

Tartalmi összefoglaló

A diplomaterv témája egy univerzális mérésadatgyűjtő műszer megtervezése és megvalósítása, mely képes feszültségjelek szimultán mérésére és előállítására, a mért adatokon digitális jelfeldolgozási eljárások elvégzésére, szűrésre, statisztikák, spektrum, harmonikus paraméterek számítására. A műszert saját grafikus felületű kijelzőről kell tudni vezérelni, ahol lehetőség van a mérés és jelgenerálás paraméterezésére, a mért jel idő- és frekvenciatartománybeli képének kirajzolására, a statisztikák és becsült paraméterek megjelenítésére. A műszernek USB kapcsolaton keresztül vezérelhetőnek kell lennie PC-ről is, mely a nagyobb számítási kapacitás révén képes átvenni a jelfeldolgozás feladatát a műszertől. A műszer központi egysége a 32 bites ARM maggal rendelkező STM32F407 mikrokontroller.

Az Önálló laboratórium korábbi félévei során sikerült megvalósítani a teljeskörűen paraméterezhető mérést és jelgenerálást a műszeren. Az aktuális félév során sikerült javítani a műszer teljesítményén, energia- és memória felhasználásán, valamint sikerült megvalósítani egy paraméterbecslő eljárást a mérendő periodikus jelek harmonikusaira. A befejező félév feladata a tervezett funkciók befejezése, grafikus vezérlőfelület elkészítése a PC-re, demonstrációs jelfeldolgozási alkalmazások írása, illetve önálló panel tervezése a műszernek.

Abstract

The task of the thesis is to design and implement a universal data acquisition instrument, which is capable of simultaneously measuring and generating voltage signals, executing digital signal processing tasks on the measured data, apply filtering, calculate statistics, spectre, approximate harmonic parameter values. The instrument is to be controlled from a programmable LCD display with graphical touch-screen, where it is possible to setup measurement and signal generation, view graphs of the measured signal in time- and frequency domain, and view statistics and approximated parameters on the incoming signal. The instrument should also be controllable from a PC through USB connection, which, due to its higher compute capabilities, able to undertake the digital signal processing tasks instead of the instrument. The central unit of the instrument is an STM32F407 microcontroller with 32 bit ARM core.

During the previous semesters, an instrument was implemented with fully parametrizable measurement and signal generation. In this semester, I was able to improve the performance and lower the energy- and memory needs of the system, and also implement a parameter approximation algorithm on the harmonics of the input signal. For the final semester, the task is to finish every remaining feature implementation, create a graphical user interface for the PC (similar to the LCD display), code some demonstration examples, and design and implement a custom board for the instrument.

Bevezetés

A diplomamunka keretében egy univerzális mérésadatgyűjtő műszer megtervezése és megvalósítása a cél, mely képes periodikus jelek digitális feldolgozására. Feladatai:

- egy vagy több bemeneti feszültségjel szimultán mérése, a mért adatokon:
  - FIR szűrés alkalmazása
  - időbeli statisztikák (átlag, RMS, szórás) számítása
  - spektrum számítása
  - harmonikusok paramétereinek számítása
- egy vagy több kimeneti feszültségjel előállítása
- paraméterezhető/vezérelhető saját érintőképernyős kijelzőről, UART kapcsolaton át
- paraméterezhető/vezérelhető PC-s alkalmazásból, USB kapcsolaton át
  - ekkor a műszer jelfeldolgozási feladatait a PC képes átvenni, a műszernek elég a nyers mérési/jelgenerálási adatokat küldenie/fogadnia

A műszer központi egysége egy 32 bites, ARM Cortex-M4F maggal rendelkező STM32F407-es mikrovezérlő. Összetett ADC/DAC és USB perifériái, valamint FPU-val rendelkező és nagy számítási kapacitású processzora révén kiválóan alkalmas a feladatra.
A műszer kijelzője egy Nextion NX4024T032 intelligens kijelző, melyre egy saját fejlesztőkörnyezetből tetszőleges megjelenésű és funkciójú grafikus felület készíthető.
PC-s vezérlő alkalmazást lehetséges MATLAB-ban vagy LabView-ban készíteni, melyek önmagukban tartalmaznak komplex jelfeldolgozási lehetőségeket, hosszú távú cél viszont egy saját grafikus felületű alkalmazás elkészítése.

Az Önálló laboratórium keretében már elkezdtem a műszer kifejlesztését, annak témáját folytatom. A korábbi munka során az alábbi eredményeket sikerült elérni:

- paraméterezhető mérés
  - 1/2/3 bemeneti csatorna szimultán mérése
  - felbontás 12 bit
  - mintavételezési frekvencia: max. 2,4MHz
  - előre definiált paraméterű FIR szűrők közti választás
- paraméterezhető jelgenerálás
  - 1 vagy 2 kimeneti csatorna
  - egyszerű szinusz jel generálása, tetszőleges paraméterekkel

- energiafelhasználás és memóriakihasználás szempontjából hatékony MCU program
- grafikus vezérlő program a kijelzőn
- PC-s vezérlés szöveges terminálból parancsokkal, vagy MATLAB programból

Az elért eredmények kifejtését ezen dokumentum tartalmazza.

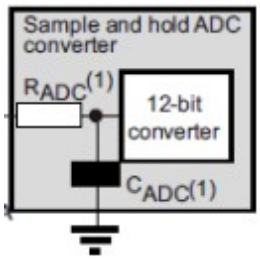
Ebben a félévben az elsődleges cél a mérendő jel harmonikusainak paraméterbecslésének megvalósítása egy szabványos eljárással (ez az Önálló laboratórium során nem került implementálásra), valamint a mérés és jelgenerálás teljesítményének javítása, felhasználhatóságának bővítése az STM32F407 által kínált lehetőségek alapján. A projekt végcélja egy kellően gyors, pontos, széleskörű funkcionalitású és energiatakarékos, fizikailag önálló műszer elkészítése, mely kijelzőről és PC-ről egyaránt, sokféle üzemmódban vezérelhető, és hatékonyan felhasználható komplex jelfeldolgozó eljárásokhoz.

1. Mérés és jelfeldolgozás folyamatának elmélete

1.1 Mérés és digitalizálás szukcesszív approximációs ADC-vel [1]

Egy mikrokontrollerben található analóg-digitális átalakító, mint digitális eszköz, csak időben diszkrét módon, kvantált időpontokban történő mintavétellel képes mérni és számszerűsíteni egy időben folytonosan változó, analóg jelet. A bemeneti lábon található jelről rendszerint f_s mintavételezési frekvenciával veszünk mintát, és egyszerre N db mérési mintát kérünk le. A mintavételt általában egy időzítő periféria indítja hardveresen.

A mintavétel kezdetekor egy kapcsoló rákapcsolja a bemenő jelet a mintavevő- és tartó áramkör bemenetére. Ez az áramkör egy egyszerű RC-körként modellezhető, ahol az R bemenő ellenálláson át, a C mintavevő kondenzátor adott T_s mintavételi idő alatt feltöltődik a bemenő feszültség bizonyos hányadára. Ezt a hányadot a kör $\tau = RC$ időállandója befolyásolja.



1.1.1 ábra: ADC bemenetének helyettesítő képe

A projektben használt STM32F407 adatlapja szerint [2] a mikrokontroller ADC-je bemenetén az R bemenő ellenállás legalább 6 kΩ, a C mintavevő kapacitás pedig 4 pF, így a τ időállandó legalább 24 ns, melyet a bemeneti jelforrás impedanciája tovább növelhet. A feltöltődés képlete:

$$U_m = U_{be} \cdot \left(1 - e^{-\frac{T_s}{\tau}}\right) \qquad (1.1.1)$$

Mint látható, minél nagyobb T_s mintavételi időt állítunk be, annál kisebb lesz a tényleges és a mért bemeneti feszültség közti hiba. A mintavételi időt rendszerint az ADC-t vezérlő órajel ciklusszámában megadva állíthatjuk be.

A mintavételezett feszültséget ezután a kapcsoló lekapcsolja a jelforrásról, és megkezdődik a digitalizálási fázis. Az approximációban a feszültségnek megfelelő kód bitenként lesz

meghatározva, egy bit meghatározása 1 ADC órajelciklus időt vesz igénybe. A teljes konverzióhoz szükséges időt (T_c) tehát az ADC felbontása határozza meg.

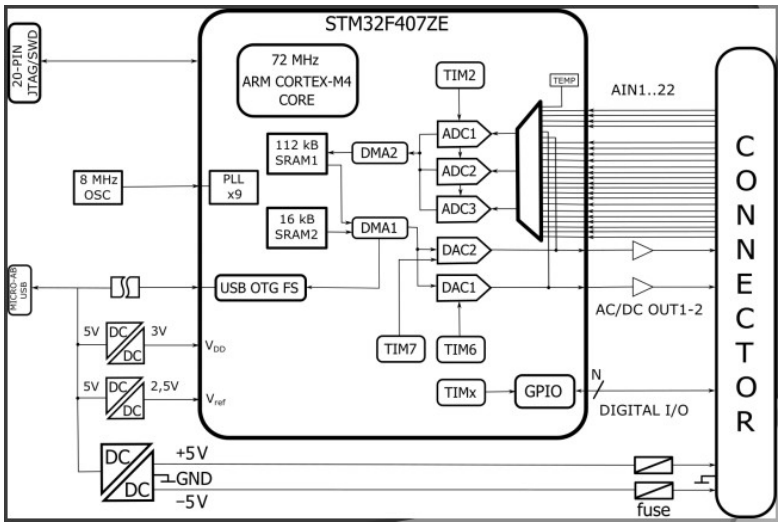
Az ADC leggyorsabb, folyamatos működésénél az elérhető legnagyobb mintavételezési frekvenciával dolgozik. Ezt a mintavételre szánt T_s és a szükséges felbontásnak megfelelő T_c approximációs idő határozza meg, az ADC órajelfrekvenciájának (f_{CLK}) függvényében:

$$f_{smax} = \frac{f_{CLK}}{T_s + T_c} \qquad (1.1.2)$$

1.2 Mért adatok tárolása és kezelése

Az ADC által mért és digitalizált jelsort a mérés végeztével el kell tárolni a mikrokontroller RAM memóriájában. Ez történhet a processzor által, az ADC adatregiszterének szoftveres átmásolásával, például az egyes ADC konverziók végét jelző megszakítások során. Ennél azonban célszerűbb módszer, ha ezt a feladatot a CPU helyett egy adatátvitelre optimalizált célprocesszor, a DMA periféria végzi automatizáltan, a szoftvertől függetlenül. Így a CPU használható közben más feladatra, vagy altatható, ezáltal csökkentve az áramfogyasztást.

A memóriában tárolt mérési adatokon ezután a processzor különféle szoftveres jelfeldolgozást végezhet, melynek végeztével a feldolgozott jelsort soros kapcsolaton át, bitenként továbbítani lehet a vezérlő egységnek (UART periféria használata esetén bájtos csomagokban, USB esetén nagyobb adatkeretekben).



1.2.1 ábra: A műszer tervezett logikai felépítése

1.3 Mért jel statisztikáinak számítása

A mérési adatokra rendszerint különféle statisztikai jellemzőket számítunk, melyek által képet kaphatunk a mért jelről. A minták átlaga/egyszerű középértéke a jel legjellemzőbb értékét reprezentálja:

$$\mu = \frac{1}{N}\sum_{i=0}^{N-1} x_i \qquad (1.3.1)$$

Az effektív középérték azt az időben állandó egyenfeszültséget reprezentálja, melynek teljesítménye egy ohmikus terhelésen megegyezik az időben változó mért jel teljesítményével.

$$\psi = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} \left|x_i\right|^2} \qquad (1.3.2)$$

Az ún. korrigált empirikus szórás azt jellemzi, hogy a jel egyes mért értékei mennyire térnek el az átlagértéktől. Ez a jel ingadozásának, zajának jellemzésére használható:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1} \left(x_i - \mu\right)^2} \qquad (1.3.3)$$

1.4 Mért jel spektrumának számítása [4]

Egy szinusz jel adott időpillanatban felvett értékét egyértelműen meghatározza az alábbi függvény:

$$y(t) = A\sin\left(\omega t + \varphi\right) + C \qquad (1.4.1)$$

Ahol az egyes paraméterek a következőek:

- A a jel amplitúdója, azaz maximális értéke
- ω a jel körfrekvenciája rad/s-ben (frekvenciája Hertz-ben $f = \omega/2\pi$), azaz az egy időegység alatt megtett teljes periódusok száma
- t az adott időpillanat
- φ a jel kezdőfázisa radiánban (fokhoz $180/\pi$-vel kell szorozni), azaz a jel eltolása a függőleges tengely mentén
- C pedig a jel egyenáramú (DC) szintje, átlagértéke, azaz a jel eltolása a vízszintes tengely mentén

A Fourier-tétel szerint az *f* alapfrekvenciájú periodikus jelek felírhatóak adott frekvenciájú, adott amplitúdójú és adott kezdőfázisú szinusz/koszinusz jelek, ún. harmonikusok (*H* darab) és egy *C* nagyságú DC-szint összegeként:

$$y(t)=C+\sum_{h=1}^{H}\left[A_h\cdot\cos(2\pi\, hft)+B_h\cdot\sin(2\pi\, hft)\right] \qquad (1.4.2)$$

A harmonikusok komplex számok, melyek amplitúdója és fázisa az *A* és *B* valós, illetve képzetes együtthatókból számítható: az amplitúdó a komplex szám abszolútértéke, a fázis pedig az argumentuma. A harmonikus frekvenciája az *f* alapfrekvencia egész számú többszöröse.

$$\begin{aligned} A_h&=\sqrt{A_h^2+B_h^2}\\ \varphi_h&=\arctan 2(B_h,A_h)\\ f_h&=h\cdot f \end{aligned} \qquad (1.4.3)$$

A Fourier-transzformáció által megkaphatóak a jelet alkotó harmonikusok. Ez a jel frekvenciatartománybeli reprezentációja, más néven spektruma, mely a jel időbeli változásait jellemzi. Mivel mikrokontroller használata esetén a mérésünk digitális és diszkrét idejű, ezért a Fourier-transzformációt is diszkrét időben kell elvégeznünk *N* db pontra, azaz diszkrét Fourier-transzformációt (DFT-t) kell használnunk. Az *N* mintaszámmal normalizált DFT szerint a spektrum *k*-adik komponense az *x* mért jelből a következőképp számítható:

$$X_k=\frac{1}{N}\sum_{j=0}^{N-1}x_j\cdot e^{-i\cdot 2\pi\cdot\frac{j\cdot k}{N}};k:0..N-1 \qquad (1.4.4)$$

A jel átlagértékét (DC-szintjét) a spektrum első, 0 Hz frekvenciájú komponense adja meg, ez egy valós szám lesz. A Nyquist-Shannon mintavételezési törvény alapján, az f_s mintavételezési frekvenciának legalább a mért jel sávszélességének (legnagyobb felharmonikus frekvenciájának) kétszeresének kell lennie, különben a spektrum nem számítható ki helyesen és a jel nem rekonstruálható. Ennek eredményeképp az f_s /2 frekvenciánál nagyobb komponensek nem hordoznak igazi információt a jelről, a spektrum második fele a spektrum első felének komplex konjugáltja lesz, figyelmen kívül hagyható. Az utolsó értelmezendő komponens a spektrum 1+*N*/2-edik, f_s /2, ún. Nyquist-frekvenciájú komponense lesz, mely a DC-komponenshez hasonlóan valós szám lesz. A két szélső érték (DC- és Nyquist-komponens) közti komponensek komplex számok lesznek. Mivel *N* db mintánk van és f_s frekvenciával mintavételeztünk, így a DFT frekvenciafelbontása $\Delta f = f_s / N$

lesz, vagyis a k-adik komponens frekvenciája $k \cdot \Delta f$. A harmonikus amplitúdóját a komponens abszolútértéke, kezdőfázisát pedig az argumentuma adja. A DFT kiszámítására rendszerint a hatékony gyors Fourier-transzformáció (FFT) rekurzív algoritmust használják. Ez a transzformáció műveletigényét $O(N^2)$-ről $O(N \cdot \log N)$-re csökkenti, de megköveteli, hogy a mintaszám kettő hatványa legyen ($N = 2^n$).

A jelet alkotó harmonikusokat alapvetően tehát négy csoportba sorolhatjuk.

- A DC-komponens már ismertetésre került.
- A periodikus jelek adott időközönként ismétlődnek, ennek frekvenciáját adja a jel alapharmonikusa.
- A jel emellett tartalmazhat olyan komponenseket, melyek frekvenciája az alapfrekvencia egész számú többszöröse, ezek lesznek a felharmonikusok. (Megjegyzés: ha a felharmonikusok nem az alapharmonikus egész számú többszörösei, akkor csak kváziperiodikus jelről beszélhetünk.)
- A DC-komponensen, az alap- és felharmonikusokon kívül minden egyéb jelen lévő komponens zaj lesz.

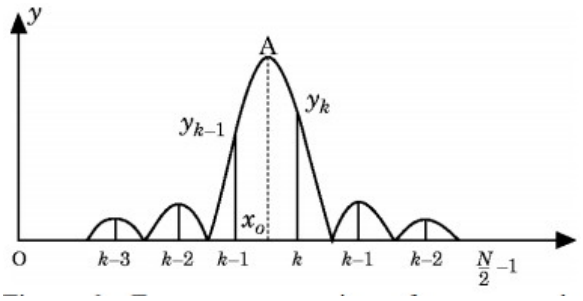
1.5 Mért jel harmonikusainak paraméterbecslése

Egy periodikus jel esetén rendszerint a frekvenciatartománybeli paraméterekre: az alap- és felharmonikusok két együtthatójára, a jel DC-szintjére és alapfrekvenciájára vagyunk kíváncsiak. Az alábbiakban ismertetésre kerül egy olyan szabványos (IEEE-STD-1057) iteratív eljárás, mely az FFT-vel kapott diszkrét spektrumból számított alapfrekvenciából kiindulva, adott pontossággal megbecsüli ezeket a keresett paramétereket.

1.5.1 Alapfrekvencia korrekció [10]

Mielőtt még nekilátnánk a paraméterbecslésnek, a kiindulási spektrum szempontjából figyelembe kell vennünk az ún. spektrálszivárgás jelenségét. Ennek oka, hogy a mérésünk időben diszkrét, ezáltal a spektrum is diszkréten kvantált. Ebből eredően, mikor az alapfrekvenciát, azaz a spektrum csúcsát (DC utáni legalacsonyabb frekvenciájú, legnagyobb amplitúdójú komponens) keressük kiindulási paraméterként, elképzelhető, hogy a csúcsként megtalált diszkrét komponens nem a pontos alapfrekvenciát, azaz a tényleges csúcsot jelöli, mert az valójában a talált csúcs és annak valamelyik (jobb vagy bal) szomszédja között van.

Ennek korrigálásához szükségünk van egy olyan módszerre, amely a talált csúcshoz tartozó „spektrum-domb” súlypontjának megkeresésével pontosabb, interpolált becslést ad az alapfrekvenciára. Annak függvényében, hogy milyen ablakfüggvényt illesztünk erre a dombra, többféle lehetőség is van az interpolációra.



1.5.1.1 ábra: Spektrálszivárgás és a korrekció elve

A legpontosabb becslést a Hanning-ablak használatával érhetjük el, mely a következő:

$$W(n)=a-(1-a)\cdot\cos(2\pi n/N) \qquad (1.5.1.1)$$

A paraméter Hanning ablak esetén $a = 0{,}5$ (Hamming-ablaknál $a = 0{,}54$). Meg kell keresnünk a spektrumban a DC utáni csúcsot, ennek indexe legyen K, amplitúdója Y_k. Ekkor a korrigált csúcsindex $K_{corr} = K + \Delta K$, melyből a ΔK korrekció egy ±1 közötti szám, számítására pedig kétféle képlet van attól függően, hogy a talált csúcs jobb vagy bal szomszédjának nagyobb az amplitúdója.

$$\Delta K = \begin{cases} \dfrac{2Y_{k+1}-Y_k}{Y_{k+1}+Y_k} & (Y_{k+1} \geqslant Y_{k-1}) \\ \dfrac{Y_k-2Y_{k-1}}{Y_k+Y_{k-1}} & (Y_{k+1} < Y_{k-1}). \end{cases}$$

1.5.1.2 ábra: Alapfrekvencia korrekció számítása

A K_{corr} korrigált csúcsindex és a $\Delta f = f_s / N$ frekvenciafelbontás szorzatából végül kiszámítható az $\omega_0 = 2\pi \cdot K_{corr} \cdot \Delta f$ korrigált alapfrekvencia, mely megfelelő kezdeti érték lesz a paraméterbecsléshez.

1.5.2 Monoharmonikus paraméterbecslés [11]

Alapesetben csak az alapharmonikus A és B együtthatóját keressük, a C szinttel és ω alapfrekvenciával együtt. A harmonikus paraméterbecslő eljárásnak alapvetően két típusa van, a három- és a négyparaméteres változat. A háromparaméteres változat az alapfrekvenciát ismertnek és pontosnak veszi, és a három másik paramétert (A, B, C) legkisebb négyzet

módszerrel, egy lineáris egyenletrendszer megoldása által, iteráció nélkül egyetlen lépésben képes meghatározni. Rendszerint azonban még a korrigált alapfrekvenciánk is pontatlan, így erre is meg kell határoznunk egy $\Delta\omega$ korrekciót. Ilyenkor a négyparaméteres eljárást kell használni, melynél ez a frekvenciakorrekció a keresett negyedik paraméter, és, mivel ez a negyedik paraméter nemlineáris, az eljárás iteratív módszert követel meg a becslésre. Ennek ellenére a háromparaméteres eljárás egyetlen lépése jól használható arra, hogy az alapfrekvencia mellett jó kezdeti becslést adjunk az *A,B,C* paraméterekre is. Előzetesen érdemes tehát az $x = [A\ B\ C]^T$ kezdeti paramétervektort megbecsülni az *y* spektrumból az ω kezdeti korrigált alapfrekvencia és a ***D*** segédmátrix segítségével az alábbi módon:

$$\mathbf{D} = \begin{pmatrix} \cos\omega t_1 & \sin\omega t_1 & 1 \\ \cos\omega t_2 & \sin\omega t_2 & 1 \\ \vdots & \vdots & \vdots \\ \cos\omega t_N & \sin\omega t_N & 1 \end{pmatrix}$$

$$\hat{\mathbf{x}} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{y}.$$

1.5.2.1 ábra: Kezdeti A,B,C paraméterek becslése

Most, hogy jó kezdeti becslés adott az *A,B,C* paraméterekre és az ω alapfrekvenciára (a $\Delta\omega$ frekvenciakorrekciót kezdetben vehetjük 0-nak), a keresett négy paramétert oszlopvektorba kell rendeznünk, az *i*-edik iterációban ezen paramétervektor: $x^{(i)} = [A^{(i)}\ B^{(i)}\ C^{(i)}\ \Delta\omega^{(i)}]^T$. Az eljárás lényege, hogy a kezdeti $x^{(0)}$ paramétervektortól és a korrigált kezdeti $\omega^{(0)}$ alapfrekvenciától kezdve, minden iterációban újrabecsüljük az alapfrekvenciát, és a ***D*** segédmátrix segítségével a paramétervektort:

$$\hat{\omega}_i = \hat{\omega}_{i-1} + \Delta\hat{\omega}_{i-1}$$

$$\hat{\mathbf{D}}_i = \begin{pmatrix} \cos\hat{\omega}_i t_1 & \sin\hat{\omega}_i t_1 & 1 & -\hat{A}_{i-1} t_1 \sin\hat{\omega}_i t_1 + \hat{B}_{i-1} t_1 \cos\hat{\omega}_i t_1 \\ \cos\hat{\omega}_i t_2 & \sin\hat{\omega}_i t_2 & 1 & -\hat{A}_{i-1} t_2 \sin\hat{\omega}_i t_2 + \hat{B}_{i-1} t_2 \cos\hat{\omega}_i t_2 \\ \vdots & \vdots & \vdots & \vdots \\ \cos\hat{\omega}_i t_N & \sin\hat{\omega}_i t_N & 1 & -\hat{A}_{i-1} t_N \sin\hat{\omega}_i t_N + \hat{B}_{i-1} t_N \cos\hat{\omega}_i t_N \end{pmatrix}$$

$$\hat{\mathbf{x}}_i = (\hat{\mathbf{D}}_i^T\hat{\mathbf{D}}_i)^{-1}\hat{\mathbf{D}}_i^T\mathbf{y}$$

1.5.2.2 ábra: A keresett paraméterek iteratív becslése

Adott iterációszám után a becslésünk kellően közel konvergál az alapharmonikus paramétereinek valós értékéihez. Ezt az iterációszámot tapasztalati úton kell megtudni, ökölszabályként 10 iteráció valószínűleg elég lesz. Emellett viszont lehetséges a $\Delta\omega$ frekvenciakorrekciós értéket is vizsgálni, mikor kerül egy adott kis ε tolerancia alá, ekkor

ugyanis megállítható az algoritmus. Kevés, zajos adatra, vagy a mintavételezési frekvenciához képest alacsony alapfrekvenciájú jelre azonban előfordulhat, hogy az eljárás nem fog konvergálni a megoldáshoz, és a hibaminimalizálás során lokális minimumban ragad. Ekkor egy lehetséges megoldás, hogy egy 0-1 közti értékű *K* faktorral csökkentjük az alapfrekvencia korrekciót minden iterációban, mely a konvergenciát lassítani, de javítani fogja (ezt a *K* értéket szintén tapasztalati úton lehet meghatározni).

$$\hat{\omega}^{(i)} = \hat{\omega}^{(i-1)} + K\Delta\hat{\omega}^{(i)}$$

1.5.2.3 ábra: Korrigált frekvencia korrekció

1.5.3 Multiharmonikus paraméterbecslés [12]

Az alapharmonikus becslésénél használt eljárás általánosítható *H* darab harmonikus meghatározására, ha a paramétervektort és a ***D*** mátrixot kiterjesztjük:

$$\hat{\mathbf{x}}^{(i)} = \left[\hat{A}_1^{(i)} \;\; \hat{B}_1^{(i)} \;\; \hat{A}_2^{(i)} \;\; \hat{B}_2^{(i)} \cdots \;\; \hat{A}_H^{(i)} \;\; \hat{B}_H^{(i)} \;\; \hat{C}^{(i)} \;\; \Delta\hat{\omega}^{(i)}\right]^T$$

$$\mathbf{D} = \begin{bmatrix} \cos(\omega t_1) & \sin(\omega t_1) & \cdots & \cos(H\omega t_1) & \sin(H\omega t_1) & 1 \\ \cos(\omega t_2) & \sin(\omega t_2) & \cdots & \cos(H\omega t_2) & \sin(H\omega t_2) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \cos(\omega t_M) & \sin(\omega t_M) & \cdots & \cos(H\omega t_M) & \sin(H\omega t_M) & 1 \end{bmatrix}$$

$$\mathbf{D}^{(i)} = \left[\mathbf{D} \;\middle|\; \begin{array}{l} \sum_{h=1}^{H} -\hat{A}_h^{(i-1)} h t_1 \sin(\hat{\omega}^{(i-1)} h t_1) + \hat{B}_h^{(i-1)} h t_1 \cos(\hat{\omega}^{(i-1)} h t_1) \\ \sum_{h=1}^{H} -\hat{A}_h^{(i-1)} h t_2 \sin(\hat{\omega}^{(i-1)} h t_2) + \hat{B}_h^{(i-1)} h t_2 \cos(\hat{\omega}^{(i-1)} h t_2) \\ \vdots \\ \sum_{h=1}^{H} -\hat{A}_h^{(i-1)} h t_M \sin(\hat{\omega}^{(i-1)} h t_M) + \hat{B}_h^{(i-1)} h t_M \cos(\hat{\omega}^{(i-1)} h t_M) \end{array}\right]$$

1.5.3.1 ábra: Eljárás változóinak kiterjesztése tetszőleges számú harmonikusra

Maga az eljárás a segédváltozók alakjától eltekintve ugyanaz, mint monoharmonikus esetben. Tetszőleges számú harmonikus meghatározható így, azonban ha rendelkezünk a priori ismerettel a mérendő jel alakjáról, bizonyos harmonikusok vizsgálata kihagyható.

1.6 Mért jel FIR szűrése [5]

A mért jelet általában zajtól, illetve zajnak ítélt felharmonikusoktól mentesen szeretnénk vizsgálni. Ez rendszerint azt jelenti, hogy egy adott frekvencia (ún. vágási frekvencia, f_c) feletti spektrum komponenseket ki kell szűrnünk a mérésből egy ún. aluláteresztő szűrővel. Erre számtalan analóg, hardveres megoldás is létezik, melyek közül a legegyszerűbb egy egyszerű RC-áramkör telepítése a bemenő csatornára, mely a vágási frekvenciának megfelelő időállandóval rendelkezik. Ennek a módszernek a hátránya, hogy a több hardverelem miatt növeli az áramkör méretét és költségét, valamint az analóg szűrő növeli a bemenő impedanciát, ami pontatlanabbá teszi a mérést. Jóval költséghatékonyabb a mért jel digitális, szoftveres szűrése. Ebben az esetben feltételezzük, hogy a mintasor zajos, és utólagos matematikai műveletekkel végzünk rajta zajszűrést egy absztrakt, diszkrét időben vett számokkal reprezentált digitális szűrő segítségével.

Az analóg/digitális szűrőket impulzusválaszuk szerint két csoportra oszthatjuk: FIR és IIR szűrőkre. A FIR szűrő egy véges impulzusválaszú rendszer, más szóval a bemenetére adott zérus idejű Dirac-impulzusra a kimenetén olyan választ ad, mely véges időn belül lecseng nullába (szemben az IIR szűrőkkel, melyek belső visszacsatolás hatására végtelen idejű választ generálhatnak). Egy diszkrét idejű, N-ed rendű FIR szűrő impulzusválasza 0-ról indulva N+1 mintán keresztül tart, míg vissza nem tér 0-ba. Szoftveres reprezentációja az impulzusválaszának diszkrét mintáival lehetséges, melyeket a szűrő koefficienseinek (együtthatóinak) is nevezünk. A normalizált FIR szűrő együtthatói ±1 között változnak, összegük 1. A vágási frekvenciát szintén normalizáltan kell megadni 0-1 között, a jel f_s / 2 Nyquist-frekvenciájának arányában. A szűrő rendjét és együtthatóit úgy kell megtervezni, hogy a szűrő a megfelelő frekvenciákat engedje át, tehát az impulzusválaszt a kívánt frekvencia-átviteli karakterisztikához kell igazítani. Az átviteli karakterisztikában vagy oszcilláció lesz a vágási frekvencia környékén, vagy lassú lesz a letörés a vágási frekvenciáig, ezek között kell egyensúlyt találni. Az impulzusválasz általában egy ablakozott függvény.

A mért jelet úgy lehetséges szoftveresen megszűrni a FIR szűrővel, hogy a jelsort konvolváljuk a szűrő impulzusválaszával/együtthatóival, vagyis a mért jelen időtartományban „végigtoljuk” a szűrő függvényt, mindig megszorozva az N legutóbbi jelmintát az aktuális szűrő együtthatóval, és összegezve az eredményt:

$$y[n] = \sum_{i=0}^{N} b_i \cdot x[n-i] \qquad (1.6.1)$$

A jel és a szűrő impulzusválaszának időtartománybeli konvolúciója frekvenciatartományban szorzást eredményez a jel eredeti spektruma és a szűrő átviteli karakterisztikája között. Így a jelből a megfelelő frekvencia feletti komponensek ki lesznek szűrve.

2. A mikrovezérlő felprogramozása

A szoftverfejlesztéshez az alább eszközök lesznek használva:

- STM32F4DISCOVERY fejlesztőpanel: tartalmaz minden szükséges hardverelemet.
- „STM32F4 DSP and standard peripherals library” és „STM32F4 USB Host & Device Stack” függvénykönyvtárak: letölthetőek a www.st.com weboldalról.
- „Clock configuration tool for STM32F40x/41x microcontrollers” rendszerinicializációs kódgenerátor: szintén az ST weboldaláról tölthető le.
- „System Workbench For STM32” fejlesztőkörnyezet: a www.openstm32.org weboldalról tölthető le. Mivel az Eclipse C/C++ szoftveren alapul, így előzetesen szükséges a Java futtatási környezet (JRE vagy JDK) telepítése a számítógép architektúrájának (x86/x64) és az operációs rendszernek megfelelő verzióval. A telepítő a fejlesztőkörnyezet mellett feltelepíti az STLink-V2-1 drivert is a PC-re, mely a panellel történő debug kapcsolat létesítéséhez szükséges a panelre integrált ST-LINKv2 debugger által.
- STM32 VCP driver: az operációs rendszernek és architektúrának (x86/x64) megfelelő verzió a www.st.com weboldalról tölthető le, és a Vezérlőpult Eszközkezelő panelén telepíthető. Ez a driver ahhoz szükséges, hogy az MCU USB kapcsolatát a PC operációs rendszere egy virtuális soros portként ismerje fel.

2.1 Alapbeállítások [1-3]

Az előző félévekben elért eredményekre támaszkodva, a mikrovezérlő alapbeállításait a következőképpen foglalnám össze:

- Tápfeszültség: 3V
- Oszcillátor: 8 MHz-es kvarckristály (HSE)
- Rendszerórajel: 72 MHz (PLL)
- Időzítők órajele: 72 MHz
- ADC/DAC órajel: 36 MHz
- Órajelek kapuzása: mindig csak az aktuálisan használt perifériákhoz
- ADC/DAC referenciafeszültség: 3V (tápfeszültség)
- ADC/DAC felbontás: 12 bit
- ADC időzítő: TIM2 (32 bites)

- ADC DMA: DMA2 0. csatorna
- ADC memória: SRAM1 (108 kB)
- DAC időzítő: TIM6 (DAC1) / TIM7 (DAC2) – mindkettő 16 bites
- DAC DMA: DMA1 5. csatorna (DAC1) / DMA1 6. csatorna (DAC2)
- DAC memória: SRAM2 (2x8 kB)
- GPIO: PA11-PA12 (USB FS D+/D-) digitális I/O MEDIUM sebességgel felhúzás nélkül, PB10-PB11 (USART3) digitális I/O LOW sebességgel felhúzó ellenállással, egyéb lábak analóg I/O
- Kommunikáció: USB Full-Speed device (virtuális COM port-ként) vagy USART3

A beállítások meghatározásánál a fő szempont a kellő gyorsaság és pontosság, valamint az energiatakarékosság volt.

2.2 Jelgenerálás beállítása [1]

A konfigurálási folyamatot a „@D<csatornaszám>” paranccsal lehet indítani, külön-külön a két csatornára. A speciális „@DX” paranccsal kétcsatornás szimultán módot lehetséges használni. Ekkor a két csatornára közös bufferméretet és frekvenciát kell beállítani, de jel egyéb paraméterei különbözőek lehetnek. Ilyenkor egyetlen DMA csatorna van használva az adatátvitelhez, tehát spórolunk a memória sávszélesség használatával.

Első beállítható paraméter a mintaszám, a generált jel az adott csatornán ennyi mintából fog állni, azaz ennyire lesz finom felbontású. 2-től 4096-ig tetszőleges páros érték beállítható. A páros érték a DMA hibák elkerülése végett követelmény, a maximális érték pedig a DAC számára elkülönített összesen 16 kB-os (8-8 kB) SRAM2 memóriakorlátjából adódik [2].

Ezután következhet a kívánt jelfrekvencia az adott csatornán. A maximális jelfrekvencia az 1 MHz-es maximum DAC mintavételezési frekvencia [2] mintaszámmal leosztott értéke lesz, de ezt szoftveresen nem fogjuk megkövetelni, hanem egy univerzális 100 kHz-es maximumig minden frekvenciát megengedünk. A kívánt jelfrekvenciából ki kell számítani a szükséges mintavételezési frekvenciát a 72 MHz / (72 MHz / f_s) képlet szerint, ahol f_s a kívánt jelfrekvencia és mintaszám szorzata. Ennek megfelelően kell beírni az ezt biztosító értéket a megfelelő időzítő reload regiszterébe (DAC1 számára TIM6, DAC2 számára TIM7).

Mivel a kimenő csatornákon szinusz jelet fogunk generálni, ezért most várjuk a szinusz további paramétereit: az A amplitúdót és a C DC szintet (kódban, nem feszültségértékben),

valamint a φ kezdőfázist (fokban). A kezdőfázist át kell számítani radiánba $\pi/180$-al történő szorzással, és meg kell határozni a p_s fázis lépésközt, mely a 2π mintaszámmal leosztott értéke lesz. Ezek alapján le tudjuk generálni a jelmintákat a következő képlet szerint:

$$y[n] = A \cdot (\sin(p) + 1)/2 + C \qquad (2.2.1)$$

A p aktuális fázis a φ kezdőfázisról indul és minden jelmintára a p_s fázis lépésközzel növeljük. Az erre kiszámított szinusz ±1 közötti értéket eredményez, ezt tehát el kell tolni 1-el és le kell osztani 2-vel, hogy [0 1] intervallumba eső számot kapjunk, melyet az amplitúdó skáláz és a DC szint tol el. A szinuszt az `arm_sin_f32(phase)` DSP függvénnyel számítjuk [6].

Ezáltal meg is lesz a kívánt szinusz jelminta-sorunk mindkét csatornára. Opcionálisan utolsó paraméterként ki lehet választani, hogy szeretnénk-e használni valamelyik beépített DAC jelgenerátort, ami mintavételenként léptetett háromszög-jelet vagy LFSR-es pszeudozajt kever a generált jelhez (0: egyik sem, 1: háromszög, 2: zaj), és ha igen, milyen amplitúdóval (kódban, 0-4095 között).

A beállított jel generálását a „@G<csatornaszám>” paranccsal lehet indítani, leállítani és a csatornát kikapcsolni a „@S<csatornaszám>” paranccsal lehetséges (szimultán jelgenerálás esetén a csatornaszám helyén csak „X” lehet). Az indításhoz engedélyezni kell a megfelelő DMA csatornát, be kell kapcsolni a megfelelő DAC-t (majd várni ~10µs-ot [2]) és el kell indítani a megfelelő időzítőt, kikapcsolásnál ugyanez kell fordított sorrendben.

2.3 Mérés beállítása [1]

A konfigurálási folyamatot a „@A” paranccsal lehet indítani.

A mérés első beállítható paramétere a mérési mód. Ez egy kódszó, melynek egyes értékei az alábbi módokat reprezentálják:

- **Tripla interleaved mód [0]:** a 3 ADC ugyanazt a csatornát méri folyamatosan, egymás után. A csatornát egyszerre mindig csak egy ADC mintavételezi, fixen 3 órajelnyi ideig, ezután 2 órajelciklus elteltével a következő ADC kezdi a mintavételezést, miközben az előző már konvertál. Vagyis a konverzió párhuzamosan zajlik, a csatorna pedig 36 MHz / (3+2) = 7,2 MHz frekvenciával van mintavételezve! A DMA minden második konverzió végeztével mozgatja az előző 2 eredményt, egy 32 bites értékként a memóriába. Fontos ennél a módnál, hogy ha megvan a kívánt számú

minta és a DMA jelzésére leállítjuk a mérést, újra kell konfigurálni az ADC-t. Mivel a konverzió folyamatos, ennél a módnál nem lesz szükség explicit mintavételezési frekvenciára, és csak egy bemenő csatornát kell megadni.

- **Szimpla mód [1]:** 1 ADC méri a kiválasztott csatornát a kívánt frekvenciával vagy folyamatosan. A DMA minden konverzió végeztével mozgatja a mérés eredményét egy 16 bites értékként a memóriába.
- **Dupla szimultán mód [2]:** 2 ADC szimultán, egymással párhuzamosan mér két külön csatornát, a kívánt frekvenciával vagy folyamatosan. A DMA minden szimultán konverzió végeztével mozgatja a két mérés eredményét egy 32 bites értékként a memóriába.
- **Tripla szimultán mód [3]:** 3 ADC szimultán, egymással párhuzamosan mér három külön csatornát, a kívánt frekvenciával vagy folyamatosan. A DMA minden szimultán konverzió végeztével mozgatja a három mérés eredményét egy-egy 16 bites értékként a memóriába.

A következő beállítható paraméter a mintaszám. Minden használt ADC-nek ennyi mérést kell majd elvégeznie (tripla interleaved módban összesen ennyi mérés kell). A lehetséges értékek: 64, 128, 256, 512, 1024. A kettő hatvány szerinti szám és a szélsőértékek az FFT számítása számára kritériumok [6].

Be lehet még állítani továbbá a mintavételezési frekvenciát (kivéve tripla interleaved mód esetén). A 0 érték jelzi, hogy folyamatos konverziót szeretnénk, vagyis időzítő vezérlése nélkül, minimális mintavételi idővel, a lehető legnagyobb (36 MHz / (12+3) = 2,4 MHz) mintavételezési frekvenciával szeretnénk mérni. Ettől eltérő érték esetén a mérést vezérlő TIM2 időzítő reload regiszterébe (72 MHz / f_s – 1) értéket fogunk írni, mely 72 MHz / (72 MHz / f_s) valós mintavételezési frekvenciát fog biztosítani. Megjegyzendő, hogy az adatlap szerint időzítő által vezérelt működésnél további 2 órajelciklus késleltetés megjelenik [2], így a maximális mintavételi frekvencia ekkor kisebb (36 MHz / (12+3+2) ≈ 2,12 MHz), ezt viszont a szoftverben nem fogjuk megkövetelni. A T_s mintavételi időt a kívánt frekvencia alapján be lehet állítani dinamikusan, kisebb frekvencia esetén nagyobb mintavételi idő beállítása ugyanis pontosabb mérést eredményez, és a két mintavétel közti idő is hasznosan lesz eltöltve. Az egyes mintavételezési frekvenciákhoz használható mintavételi idők:

f_s	T_s
1,33 MHz - 2,4 MHz	3 ciklus
900 kHz – 1,33 MHz	15 ciklus
530 kHz – 900 kHz	28 ciklus
375 kHz – 530 kHz	56 ciklus
290 kHz – 375 kHz	84 ciklus
230 kHz – 290 kHz	112 ciklus
73 kHz – 230 kHz	144 ciklus
1 Hz – 73 kHz	480 ciklus

2.3.1 táblázat: ADC mintavételezési frekvenciasávokhoz rendelt mintavételi idők

A mintavételi frekvencia és a hozzá igazított mintavételi idő beállítása után következik a kívánt bemenő csatornák kiválasztása. Tripla interleaved és szimpla módban 1, dupla szimultán módban 2, tripla szimultán módban 3 bemenő csatorna azonosítót várunk. Minden ADC-re 16 külső csatorna közül választhatunk, az ADC1-nél emellett további 3 belső csatorna közül is, melyek sorban a belső hőmérsékletszenzor, a V_{REF} külső referenciafeszültség és az ún. V_{BAT} tartalék tápellátás feszültsége. Ezt az utolsó három csatornát külön be kell kapcsolni, amennyiben ezek lettek kiválasztva, és egy Delay függvénnyel meg kell várni az adatlapban specifikált [2] felmelegedési idejüket (~10µs). Megjegyzendő, hogy az adatlap a 3 belső csatorna mérésére a maximális mintavételi időt javasolja [2], tehát ezeket a pontos méréshez maximum 72 kHz-el szabadna csak mintavételezni, de ezt a szoftverben nem fogjuk megkövetelni.

Végezetül az utólagos FIR szűrőt kell kiválasztani. A 0-s érték jelzi, hogy nem szeretnénk szűrést, egyéb számokkal pedig a MATLAB-ban előzetesen legenerált, különböző paraméterű konstans szűrőegyüttható-tömbök közül lehet választani. Szűrőtervezés MATLAB-ban a legegyszerűbben a `fir1` függvénnyel történhet, mely egyszerű Hamming-ablakos szűrőt tervez. Paraméterként meg kell adni a szűrő rendjét (együtthatók száma – 1), valamint a vágási frekvenciát normalizáltan, vagyis a Nyquist-frekvencia hányadosaként. Jelenleg az alábbi teszt jellegű szűrök közül lehet választani, ez később bővülni fog:

- 10%-os vágási frekvenciájú, 50 együtthatójú szűrő
- 20%-os vágási frekvenciájú, 50 együtthatójú szűrő
- 5%-os vágási frekvenciájú, 32 együtthatójú szűrő

2.4 Mérés elvégzése, kiértékelés [6]

A beállított mérést a „@M<tartomány>” paranccsal lehetséges futtatni és lekérni.

A mérés lebonyolításának folyamata:

1. engedélyezni kell a DMA csatornát
2. be kell kapcsolni az ADC-(ke)t
3. meg kell várni az adatlapban specifikált felmelegedési időt (~3µs) [2]
4. el kell indítani a mérést
    - időzítő által vezérelt módban el kell indítani a TIM2-t
    - folyamatos módban egy szoftveres flaget kell beállítani
5. a DMA2 megszakítása fogja jelezni, hogy a beállított számú mérés lezajlott és az eredmények elérhetőek a bufferben

A következő lépés az ömlesztett, DMA által hardveresen telepakolt bufferből az egyes ADC-khez tartozó minták szétválogatása [1].

- Tripla interleaved és szimpla módban a 32 bites bufferelem alsó 16 bitje lesz időben az 1. jelminta, felső 16 bitje a 2. jelminta, és így tovább.
- Dupla szimultán módban az alsó 16 bit az ADC1, a felső 16 bit az ADC2 mintája lesz.
- Tripla szimultán módban az alsó 16 bit az ADC1, a felső 16 bit az ADC2, a következő 32 bitnek az alsó 16 bitje pedig az ADC3 mintája lesz.

Amennyiben be lett állítva FIR szűrő, úgy most annak alkalmazása következik. A CMSIS DSP könyvtár FIR szűréséről azt kell tudni, hogy blokkos feldolgozású [7], vagyis nem egyszerre az egész bemeneti mintasort konvolválja az együtthatókkal, hanem egyszerre csak egy megadott méretű blokkot (valószínűleg futásteljesítmény növelése céljából). A blokkméret most fixen 64-re van állítva az MCU programban. A szűrő együtthatók száma azért volt ennyiben maximalizálva, mert az nem lehet nagyobb a blokkméretnél, hiszen akkor a konvolváló ablak hosszabb lenne a konvolválandó mintasornál. A szűrő előkészítése az `arm_fir_init_f32` DSP függvénnyel történik, melynek paraméterül meg kell adni az együtthatók számát, az együtthatók tömbjét, egy köztes ún. állapot tömböt (melyet csak a függvény használ), és a blokkméretet. A blokkok száma a mérési mintaszám blokkmérettel leosztott értéke lesz, minden használt ADC-re. A szűrőt az `arm_fir_f32` DSP függvénnyel kell alkalmazni az összes használt ADC összes blokkjára. Ezzel készen lesz a szétválogatott, megszűrt jelsorunk. A FIR szűrés forráskódja:

```
arm_fir_init_f32(&S,
(uint16_t)ADC_FIR_numTaps,&ADC_FIR_coeffBuffer[0],&ADC_FIR_stateBuffer[0],
(uint32_t)ADC_FIR_blockSize);

for(i=0; i < ADC_FIR_numBlocks; i++)
{
	arm_fir_f32(&S, &ADC1_buffer_unfiltered[0] + (i * ADC_FIR_blockSize),
&ADC1_buffer[0]	+ (i * ADC_FIR_blockSize), (uint32_t)ADC_FIR_blockSize);

	if (ADC_mode > 1){
		arm_fir_f32(&S, &ADC2_buffer_unfiltered[0] + (i * ADC_FIR_blockSize),
			&ADC2_buffer[0] + (i * ADC_FIR_blockSize),
(uint32_t)ADC_FIR_blockSize);

		if (ADC_mode == 3) arm_fir_f32(&S, &ADC3_buffer_unfiltered[0] + (i *
			ADC_FIR_blockSize), &ADC3_buffer[0] + (i * ADC_FIR_blockSize),
				(uint32_t)ADC_FIR_blockSize);
	}
}
```

Innentől szétválik a mérést kiértékelő folyamat. Amennyiben a lekérni kívánt tartománynak az időtartomány lett beállítva („T”), úgy időtartomány-beli statisztikákat számítunk minden használt ADC jelsorára: átlagot, RMS-t és szórást. Ennek megvalósítása az előző félévekben már megtörtént az `arm_mean_f32`, `arm_rms_f32` és `arm_std_f32` DSP függvények segítségével. A számított statisztikákat feszültségértékbe is konvertálhatjuk, ha az `arm_scale_f32` DSP függvény segítségével a V_{REF} / 4095 hányados ezerszeresével skálázzuk őket. A számítások végén el kell küldeni a vezérlő eszköznek a használt ADC-k időtartománybeli adatsorát, és a kiszámított statisztikákat.

Az „F” paraméterrel a frekvenciatartomány kiértékelését kérjük le, ahol FFT-t kell alkalmazni a jelsor(ok)ra. Az FFT-t az `arm_rfft_fast_init_f32` függvénnyel szükséges előkészíteni, megadva a 2 hatványa szerinti, max. 1024 értékű beállított mintaszámot. Az FFT-t az `arm_rfft_fast_f32` függvénnyel kell alkalmazni az adott ADC jelsorára, és mivel ez normalizált spektrumot számít, ezután skálázni kell a kiszámított komponensvektort a mintaszám reciprokával ($1/N$). A két szélső érték, a DC- és Nyquist-komponens a vektor első két eleme lesz valós számként, ezeket ki kell venni a vektorból külön változókba. A vektor további elemei a köztes, komplex komponensek valós és képzetes részei lesznek felváltva, és mivel alapvetően amplitúdó karakterisztikára leszünk kíváncsiak, így ezekre ki kell számítani az abszolútértéket az `arm_cmplx_mag_f32` függvény használatával. Alkalmazva a leírt műveletsort az összes használt ADC jelsorára, megkapjuk a jelsor(ok) spektrumát, melyet továbbítani kell a vezérlő eszköznek. Az FFT-t végrehajtó kódrészlet (egy ADC-re):

```
fft_status = arm_rfft_fast_init_f32(&fft_inst,ADC_bufferSize);

arm_rfft_fast_f32(&fft_inst,ADC1_buffer,ADC1_fft,0);
arm_scale_f32(ADC1_fft,1.0/(float32_t)ADC_bufferSize,ADC1_fft,ADC_bufferSize);
ADC1_fft_DC_comp = ADC1_fft[0];
ADC1_fft_rad_comp = ADC1_fft[1];
arm_cmplx_mag_f32(ADC1_fft,ADC1_fft_magn,ADC_bufferSize);
ADC1_fft_magn[0] = ADC1_fft_DC_comp;
ADC1_fft_magn[ADC_bufferSize/2] = ADC1_fft_rad_comp;

VCP_Send_Byte_Buffer(ADC1_fft_magn,ADC_bufferSize/2 + 1);
```

Végezetül az alapharmonikus paraméterbecslését kell elvégezni. Ez előbb MATLAB-ban tesztelésre került. Az FFT korrekció MATLAB kódja:

```
function J = IpFFT_Hann(x)
    N = length(x);
    X=abs(fft(x));
     [~,  ind_max]=max(X(3:round(N/2)));  % Az  elso  2  pontot  nem  nezzuk  a  DC  es
oldalhullama miatt
    k = ind_max + 1;
    if (X(k+1) > X(k-1))
        dJ=(2*X(k+1) - X(k))/(X(k+1) + X(k));
    end
    if (X(k+1) <= X(k-1))
        dJ = (X(k) - 2*X(k-1))/(X(k) + X(k-1));
    end
    J = ind_max + dJ;
end
```

A monoharmonikus paraméterbecslés MATLAB kódja:

```
function [A,B,C,J]=LS(x)
N = length(x);
n = (0:N-1).'; % oszlopvektor
% Kezdeti becslo J-re (IpFFT) es A-B-C-re
J = IpFFT_Hann(x);
p_init = [0, 0, 0];
D = [ ...
        cos((2*pi*J*n)/N), ...
        sin((2*pi*J*n)/N), ...
        ones(N, 1), ...
        ];
p_init = pinv(D)*x;
p = [p_init(1), p_init(2), p_init(3), 0];
for ii = 1:10
    A = p(1);
    B = p(2);
    C = p(3);
    J = J + p(4); % Itt p megvaltozast tartalmaz
    % Derivalt matrix
    D = [ ...
        cos((2*pi*J*n)/N), ...
        sin((2*pi*J*n)/N), ...
        ones(N, 1), ...
        (2*B*n*pi.*cos((2*pi*J*n)/N))/N - (2*A*n*pi.*sin((2*pi*J*n)/N))/N ...
        ];
    p = pinv(D)*x;
end
end
```

A *pinv* függvény a pszeudoinverzet jelöli, és megfelel a paramétervektor fentebb leírt szorzójának (az MCU-n nincs ilyen függvény, ott ide sorban le kell írni a mátrixműveleteket). Generálva egy adott paraméterekkel rendelkező, zajos szinusz jelet és tesztelve a paraméterbecslő eljárást arra juthatunk, hogy az algoritmus helyesen működik:

```
>> N = 1024; n = 0:N-1;
>> A = 0.25; B = 0.55; C = -0.3; J = 5.273;
>> Fi = 2*pi*J*n/N; x = C + A*cos(Fi) + B*sin(Fi); x = x + 0.1*randn(size(x));
>> [A_, B_, C_, J_] = LS(x)

A_ =

    0.2474


B_ =

    0.5477


C_ =

   -0.2988


J_ =

    5.2739
```

2.4.1 ábra: Paraméterbecslő tesztelése a MATLAB-ban

MCU-n az **arm_max_f32** függvény használható a spektrum maximumának megtalálására, a tömb kezdőindexéhez azonban 2-t hozzá kell adni, hogy a DC és oldalhulláma ne szerepeljen a maximumkeresésben (ezt aztán vissza kell korrigálni). A korrekció kódja az MCU-n:

```
arm_max_f32(ADC1_fft_magn + 2,ADC_bufferSize/2 - 1,&ADC1_fft_magn_max,&ADC1_fft_magn_maxind);
ADC1_fft_magn_maxind += 2;

if (ADC1_fft_magn[ADC1_fft_magn_maxind+1] > ADC1_fft_magn[ADC1_fft_magn_maxind-1])
       ADC1_fft_magn_maxind_corr=(2*ADC1_fft_magn[ADC1_fft_magn_maxind+1]-
       ADC1_fft_magn[ADC1_fft_magn_maxind])/(ADC1_fft_magn[ADC1_fft_magn_maxind+1] +
              ADC1_fft_magn[ADC1_fft_magn_maxind]);
else
       ADC1_fft_magn_maxind_corr=(ADC1_fft_magn[ADC1_fft_magn_maxind] -
       2*ADC1_fft_magn[ADC1_fft_magn_maxind-1])/(ADC1_fft_magn[ADC1_fft_magn_maxind]
+      ADC1_fft_magn[ADC1_fft_magn_maxind-1]);

ADC1_fft_freq = (ADC1_fft_magn_maxind + ADC1_fft_magn_maxind_corr) *
(ADC_samplingFreq / ADC_bufferSize);
```

A paraméterbecsléshez tartozó kód meglehetősen hosszú az MCU-n, mivel a C nyelv nem mátrixok kezelésére optimalizált. A mátrixműveletek (szorzás, inverz, transzponálás) azonban mind adottak függvények formájában a DSP függvénykönyvtárban – a mátrixok struktúraként vannak tárolva, melyekre definiált a szélességük és magasságuk, valamint sorfolytonos

pointer-tömb formájában az értékeik. Emellett vannak függvények vektorműveletekre, trigonometrikus (cos/sin) műveletekre, komplex számokra (abszolútérték-számítás), a standard C math.h könyvtárában pedig megtalálható az *atan2* függvény is a fázisszámításhoz, úgyhogy a MATLAB program logikáját követve az MCU-ra is teljeskörűen implementálható az eljárás.

3. Vezérlés PC-ről [8,9]

Az eszköz programozhatóságának, funkcióinak kipróbálásához különféle MATLAB scripteket írtam. MATLAB-ban a VCOM porton történő kommunikáció ún. Serial objektumon keresztül zajlik úgy, mint hogyha fájlba írnánk/fájlból olvasnánk. Elsőként létre kell hozni az objektumot a Windows által hozzárendelt COM port azonosító megadásával. Azután az objektumra tulajdonságokként definiálni kell a soros port beállításokat: bitráta, adatbitek száma, paritásbit, RTS/CTS stb. Itt ezekkel nem kell foglalkozni, mivel a COM port virtuális, valójában USB kommunikáció zajlik, tehát tetszőleges beállításokkal működni fog. Érdemes viszont az I/O buffert a maximális 4096 byte-os értékre beállítani, hogy lassabb adatfeldolgozás esetén ne történjen túlcsordulás. Emellett a bájtsorrendet big endian-ra kell állítani, mert az MCU oldalon fordított sorrendben fog történni az átküldött bájtok fogadása, így viszont a PC eleve a megfelelő formátumban küldi az adatokat. A port megnyitása az `fopen` függvénnyel történik. Átküldeni bináris adatokat az `fwrite` függvénnyel lehetséges. Alapértelmezés szerint az átküldendő adatot/változót a MATLAB előjel nélküli bájtként, azaz *uint8* típusként értelmezve küldi át (ha tömb, akkor szimpla előjel nélküli bájtsorként értelmezi). A parancsokat például így, karakterenként kell átküldeni, kezdve a „@” karakterrel, majd a parancsazonosítóval, majd a paraméter karakterrel. A mérési és jelgenerálási paramétereket azonban, ha nem 8 bitesek, hanem más típusúak (pl. a mintavételezési frekvencia 32 bites), akkor számukra az `fwrite` függvényben paraméterként meg kell adni az adattípust is (tehát pl. *uint32*). Ezeket a big endian beállítás miatt a PC fordított bájtsorrendben fogja átküldeni, mely így az MCU-nál pont helyesen fog megérkezni. Bináris adatok olvasása a portról az `fread` függvénnyel történik, melynél paraméterként meg kell adni, hogy hány bájtnyi adatot várunk. Ha végeztünk a vezérlési feladattal, a portot az `fclose` függvénnyel kell lezárni, és a Serial objektumot a `delete` függvénnyel kell törölni.

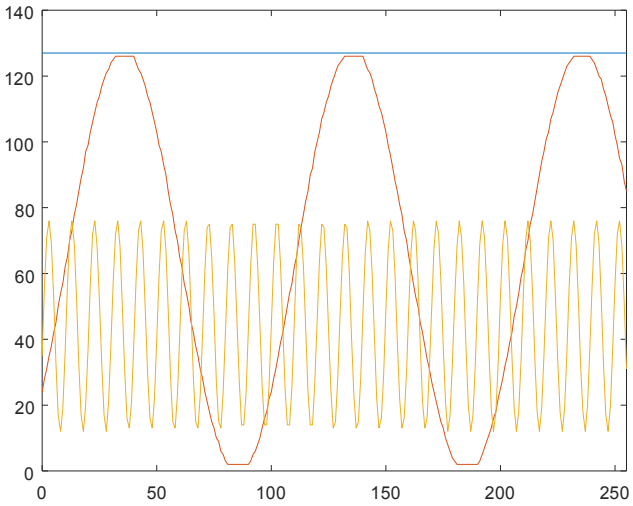
Teszteljük a vezérlést! Mindkét DAC kimenő csatornára beállítok egy-egy generálandó szinusz jelet.

- 1. csatorna: A=4095 (3V), f=1 kHz, φ=0°, C=0, N=256
- 2. csatorna: A=2047 (1,5V), f=10 kHz, φ=+90°, C=409 (0,3V), N=256

A mérésre beállítom, hogy tripla szimultán módú legyen, N=256 mintaszámmal és f_s=100 kHz mintavételezési frekvenciával. Az ADC1 bemenő csatornája a CH1, az ADC2-é a CH2, az ADC3-é pedig a CH3 lesz. A panelen:

- az ADC1 bemenő csatornáját (PA1 láb) a tápfeszültségre (V_{DD} láb) kötöm
- az ADC2 bemenő csatornáját (PA2 láb) a DAC1 kimenő csatornájára (PA4 láb) kötöm
- az ADC3 bemenő csatornáját (PA3 láb) a DAC2 kimenő csatornájára (PA5 láb) kötöm

A három bemenő csatornán tehát sorban egy tápfeszültség közeli DC jelet, egy lassabb frekvenciájú, teljes amplitúdótartományt kitöltő szinusz jelet, és egy gyorsabb, féltartományt kitöltő szinuszjelet várok. FIR szűrést nem kérek. A kapott eredmény a `plot` függvénnyel kirajzolva (a [0 127] kód intervallum a [0V 3V] feszültségtartománynak felel meg):



3.1 ábra: Tripla szimultán mérés tesztelése

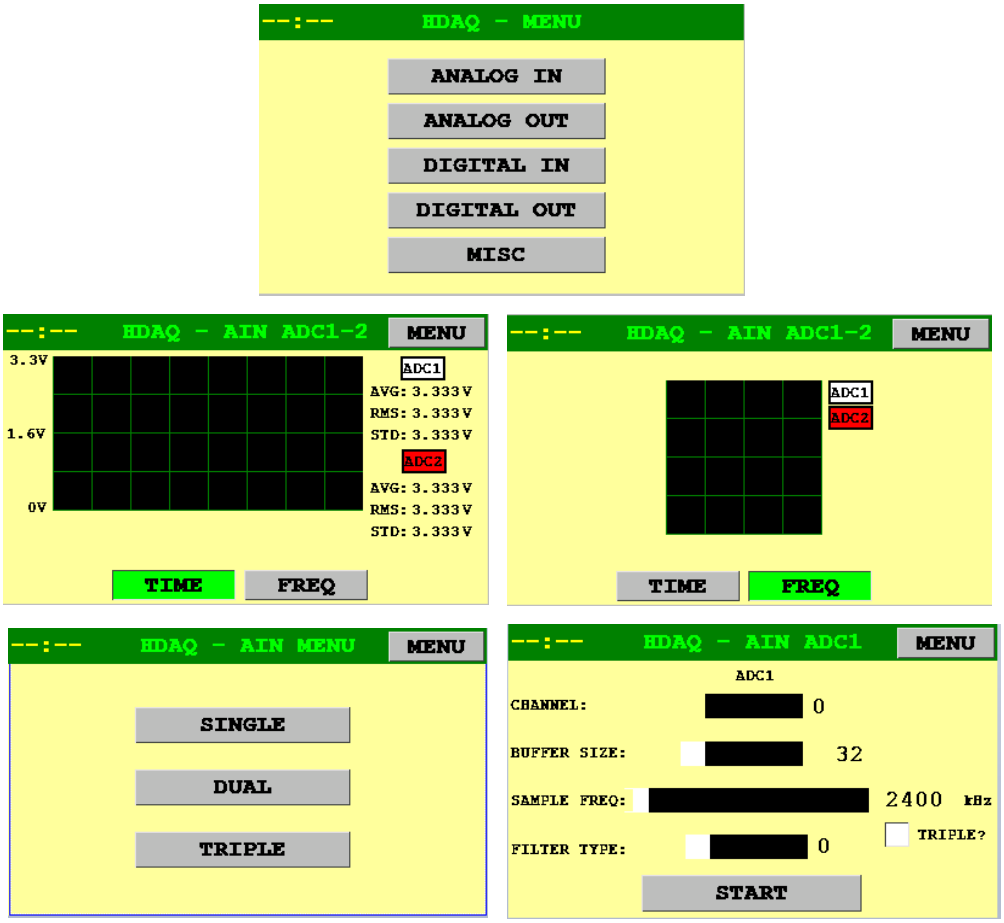
Látható, hogy a gyorsabb jel kb. 10x nagyobb frekvenciával és fele akkora amplitúdóval rendelkezik, mint a lassú, a DC szintek és a fáziseltérés is megfelelő, az N / f_s hányados által kiszámítható időbeli ablakhossz alapján pedig a frekvenciák is helyesek. A jelgenerálás és a mérés is megfelelően működik.

4. Vezérlés kijelzőről [13-15]

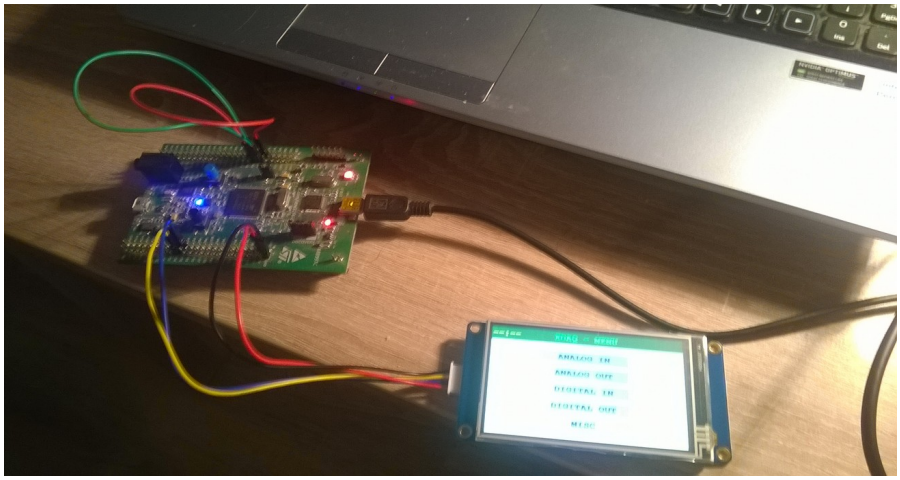
A szoftverfejlesztéshez az alábbi programokat szükséges előzetesen telepíteni:

- „Nextion Editor” fejlesztőkörnyezet: a www.nextion.itead.cc weboldalról tölthető le.
- FTDI VCP driver: az operációs rendszernek és architektúrának (x86/x64) megfelelő verzió a www.ftdichip.com weboldalról tölthető le, és a Vezérlőpult Eszközkezelő panelén telepíthető. Ez a driver ahhoz szükséges, hogy az FTDI-C232HD UART -> USB átalakító kábelt a PC operációs rendszere egy virtuális soros portként ismerje fel, ez szükséges a kijelző csatlakoztatásához a PC-hez, felprogramozás céljából.

A fejlesztőkörnyezetben egyszerűen összeállíthatóak a grafikus felületek, a megfelelő elemek elhelyezésével, és a megfelelő eseményekhez/időzítőkhöz definiált programkódokkal. Az aktuális oldal neve mindig a felső sávban lesz megjelenítve, ahogy itt lesz a főmenübe való visszatérés gombja is. A főmenüben a megfelelő felületre navigálhatunk a megfelelő gombbal: analóg mérésre vagy a jelgenerálás beállítására, illetve a jövőben elkészítendő lehetőségekre. Az analóg mérés felületén válthatunk idő- és frekvenciatartomány-beli képek között, mindkettőn egy közös grafikonon fognak megjelenni a két bemenet mérési pontjai, illetve az időtartományban a statisztikák is ki lesznek írva bemenetenként. A kijelző programkódja az analóg mérés felületén a képet 100 ms-onként frissíti, a „@M” parancs elküldésével. Ez körülbelül elég idő arra, hogy a kezelő valós időben lássa a mért jeleket, de elég idő a statisztikák kiszámításához és a kommunikáció lefolytatásához is.



4.1 ábra: A kijelző képei



4.2 ábra: Az összeállított rendszer

5. Továbbfejlesztés

A kimenő csatornákon egyszerű szinusz jelnél bonyolultabb jel előállítása egyelőre nem lehetséges, de folyamatban van az implementációja felharmonikusok beállíthatóságának (egyszerűen csak egy súlyozott összeg kell a kimenetre), valamint egyszerű DC jel generálhatóságának is. Ennek implementációjáig a multiharmonikus paraméterbecslés az MCU-n nem tesztelhető. A végleges verzióban lesz egy olyan opció is, hogy a PC szimplán leküldi a generálni kívánt mintasort. Ez hasonló lesz a „buta" mérési módhoz, ahol az MCU mindenféle saját feldolgozás nélkül elküldi a mért adatsort ömlesztve a PC-nek. Ezeket a módokat érdemes implementálni, mivel a PC sokkal összetettebb jelformák előállítására és sokkal bonyolultabb jelfeldolgozásra képes, mint az MCU.

Az elkészült műszert ki lehet még egészíteni olyan, mérésadatgyűjtőkre jellemző egyéb funkciókkal, mint a digitális I/O (az analóg mellett), a mérési eredmények tartós tárolása a flash-ben, egyéb statisztikák (pl. jel-zaj viszony) számítása, vagy a valós idejű óra (RTC) használata. A bővített program vezérlését a PC mellett a kijelzőre is implementálni kell, majd a végleges kijelzőprogrammal egyező, vagy legalább hasonló kinézetű grafikus felületet a PC-n is meg kell valósítani, akár MATLAB-ban, akár önálló Windows-os alkalmazásként (például C++ könyvtárak felhasználásával). Végezetül, mikor a szoftver készen van, meg kell tervezni egy önálló áramkört az eszköznek, melyet így a következő félév jelfeldolgozó eljárásaiban hatékonyan használni lehet.

Összefoglalás

A félév során sikerült továbbfejleszteni és további funkcionalitásokkal kibővíteni az Önálló laboratórium során elkezdett munkát:

- Sikeresen meg lett valósítva a harmonikus paraméterbecslő eljárás a mérésnél, egyelőre az alapharmonikusra (a multiharmonikusra történő kiterjesztés elő van készítve).
- A mérésnél sikerült teljeskörűen implementálni a tripla interleaved mérési módot, melyben a három ADC egység összehangoltan mér egyetlen bemeneti csatornát, ezzel akár 7,2MHz mintavételezési frekvencia is elérhető.
- A mérésnél emellett befejezésre került a mintavételi idő adaptív hangolása, mely alacsonyabb mintavételezési frekvenciákon pontosabb mérést biztosít.
- A jelgenerálásnál lehetséges a beépített háromszögjel/zajgenerátor jelét ráültetni a kimeneti jelre, tetszőleges amplitúdóval. Emellett lehetséges szimultán generálni azonos frekvenciájú jelet a két csatornán, memóriatakarékosság végett.

A továbbiakban be kell fejezni az összes tervezett funkció, köztük a multiharmonikus paraméterbecslés implementációját a mikrokontrollerre, majd el kell készíteni a véglegesített vezérlést kijelzőre és PC-re is. Amikor ez megvan, tervezni kell egy önálló áramkört a műszernek, mely tartalmazza:

- a mikrokontrollert és a körülötte lévő hidegítést,
- a kijelzőt és a hozzá kapcsolt UART vonalat,
- a tápellátást biztosító, programozó és PC-s vezérlő USB kapcsolatot,
- egy külső referenciafeszültséget,
- egy oszcillátort,
- valamint a bemeneti és kimeneti vonalakat és azok jelkondicionálását.

Az elkészült műszert aztán a Diplomaterv 2 kurzus keretein belül fel kell használni különféle bonyolultabb, például véletlenszerűséget alkalmazó jelfeldolgozó eljárások megvalósításához és teszteléséhez. A diplomamunkát a műszer kifejlesztéséről és funkcióiról, valamint a tanulmányozott jelfeldolgozó eljárásokról szeretném írni.

Irodalomjegyzék

1. STMicroelectronics: STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs reference manual rev. 12, 2016. (RM0090)

2. STMicroelectronics: STM32F405xx, STM32F407xx datasheet rev. 8, 2016.

3. STMicroelectronics: Discovery kit with STM32F407VG MCU user manual rev. 5, 2016. (UM1472)

4. Wikipedia [hu]: Fourier-transzformáció (https://hu.wikipedia.org/wiki/Fourier-transzformáció)

5. Wikipedia [en]: Finite Impulse Response (https://en.wikipedia.org/wiki/Finite_impulse_response)

6. Keil: CMSIS-DSP Software Library user manual

(http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html)

7. Keil: FIR Lowpass Filter Example (https://www.keil.com/pack/doc/CMSIS/DSP/html/group__FIRLPF.html)

8. Mathworks: Getting Started with Serial I/O
(https://www.mathworks.com/help/matlab/matlab_external/getting-started-with-serial-i-o.html)

9. Mathworks: FIR Filter Design (https://www.mathworks.com/help/signal/ug/fir-filter-design.html)

10. Xie Ming & Ding Kang: Corrections For Frequency, Amplitude And Phase In A Fast Fourier Transform Of A Harmonic Signal – Chonqing University, China, 1996.

11. Peter Händel: Properties of the IEEE-STD-1057 Four-Parameter Sine Wave Fit Algorithm – Uppsala University, Sweden, 2000.

12. Pedro M. Ramos & A. Cruz Serra: Least Squares Multiharmonic Fitting: Convergence Improvements – University of Lisbon, Portugal, 2007.

13. Nextion: NX4024T032 datasheet (https://nextion.itead.cc/resources/datasheets/nx4024t032_011/)

14. Nextion: The Nextion Editor Guide (https://nextion.itead.cc/editor_guide/)

15. Nextion: The Nextion Instruction Set (https://nextion.itead.cc/resources/documents/instruction-set/)