Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs rendszerek tanszék

ECG TESZTELŐ KÉSZÜLÉK FEJLESZTÉSE

TANSZÉKI KONZULENS

Dr. Benesóczky Zoltán

BUDAPEST, 2015

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Budavári Ruben Pál**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot 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, 2015. 12. 10.

...……

Budavári Ruben Pál

Összefoglaló

Az orvostechnikai eszközök fejlődésével egyre több helyen alkalmaznak különféle vizsgáló, mérő, beavatkozó elektromos készülékeket, amelyek egyre bonyolultabb funkciókat képesek ellátni. Ezek közül az egyik a szív működését, elektromos jeleit vizsgáló EKG készülék. Ezeknek a berendezéseknek a vizsgálatára úgynevezett „EKG teszteret” szokás használni, amivel a szerkezet megfelelő működését ellenőrzik. Ezt a módszert két fő okból szokták alkalmazni. Egyik ezek közül, amikor EKG készüléket fejlesztenek, gyártanak, akkor így tudják megfigyelni, hogy minden funkció megfelelően működik, tudják kalibrálni a készüléket. A másik ok pedig, hogy az EKG berendezéseket meghatározott időközönként felül kell vizsgálni, hogy megfelelnek-e a törvényi előírásnak és ekkor egy tesztelő készülékkel ellenőrizhető a helyes működés.

Szakdolgozatom témája egy ilyen tesztelő eszköz megtervezése és az egyszerűbb funkciókat megvalósító deszkamodell megvalósítása. A Medihead Kft., ahol a dolgozatot készítettem EKG eszközök időszakos felülvizsgálatával is foglalkozik, ilyen helyen pedig van igény akár több teszter berendezésre is.

A dolgozat készítése során először az élettani hátteret vizsgálom meg, majd elkészítem a deszkamodellt, az ebből szerzett tapasztalatok alapján pedig megtervezem a végső készülék hardverét rendszer és kapcsolási rajz szinten és a szoftvert rendszerszinten. A dolgozat írásánál a sorrend fordított, tehát a deszkamodell a teljes funkciókat ellátó készülék után szerepel.

Abstract

With the development of medical tools various tester measuring, actuator devices become widely used, which can serve more and more complex functions. One of these is the examiner of electrical signals and operation of the hearth, the ECG. For the purpose of examining these devices the so-called “ECG tester” is commonly used, to assure the proper operation of the structure. This method is used for two main reasons. One of these is to observe all the functions of the device are working properly and to calibrate it while developing and manufacturing ECG devices. The other reason is that in given amount of time intervals the ECG devices must be inspected if they meet the requirements of law and by these inspections, the proper operation can be verified with a tester device.

The subject of my thesis is designing an ECG tester and build the breadboard version with simple functions. I’ve made my thesis in the Medihead Ltd. This company performs periodical inspections of ECG devices thus requires one or more tester devices.

While creating this paper, first I examined the physiological background, then I created the breadboard, and using the acquired knowledge. I planned the hardware on system and circuit diagram level and the software on system level of the final device. While writing the paper the order is backwards, so the breadboard is featured after the fully functioning device.

1 Bevezetés

Az EKG készülékek tesztelése, fejlesztése, hitelesítése esetén gyakori olyan eszköz alkalmazása, mely az emberi testen mérhető jelnek megfelelő (amplitúdó, időbeni alakulás) jelet képes előállítani, emellett egyéb vizsgálójelek is beállíthatóak ezeken a teszter készülékeken. A cég, ahol a szakdolgozatomat írtam különféle orvosi készülékek időszakos felülvizsgálatával, kalibrálásával is foglalkozik, így majdnem napi szinten kell EKG készülékeket, őrzőmonitorokat felülvizsgálniuk. A szakdolgozat célja egy olyan EKG tesztelő készülék kifejlesztése, ami képes 12 csatornás EKG-k, őrzőmonitorok tesztelésére.

1.1 A készülékről általánosan

A teszter a legáltalánosabb EKG jeleket lesz képes előállítani, különféle percenkénti szívütésszám szerint, emellett bifázisos szinusz, négyszög és háromszög előállítására lesz alkalmas, ez utóbbi performansz jelek célja, hogy így könnyebb detektálni az esetleges hibát az EKG-ben, mert egyértelműbben látszik, ha nem „egyenes” a négyszög vagy ha időben valamilyen oknál fogva torzít a készülék.

Mivel a készülék letárolt adatokból állítja elő a kívánt jelalakot, így egyszerűen új adatok betöltésével más jel is kiadható, így akár a kóros jelek felismerését is lehet majd tesztelni az újabb készülékeken vagy egyéb igények miatt másra is lehet tesztelni. (A kóros jelalakok kiadásának lehetősége nem témája a szakdolgozatnak.)

A készülék kétféle üzemmódban is képes működni, egyrészt önállóan nyomógombok segítségével kiválasztva a jelalakot és annak frekvenciáját; másrészt USB-n keresztül PC-hez kapcsolva is beállíthatóak a megfelelő jelalakok. Az eszköz az éppen aktuális beállítást megjeleníti egy kétsoros LCD kijelzőn.

1.2 Az eszköz használhatósága

Ahogy a bevezetőben is említettem a EKG készülékek és őrzőmonitorok vizsgálata, tesztelése során szükség van egy másik eszközre, ami az ember által keltett EKG jelet előállítja a megfelelő feszültségértékkel, jelalakkal és időfüggvénnyel. Az elvezetéshez való kábelek vizsgálatát (úgynevezett pácienskábel) néha emberekkel végzik és valósan felrakják az illető végtagjára, mellkasára az vezetékeket, de ez ritka és

a 30 szívütés per perc (BPM, Beats Per Minute) valamint 240 BPM között nem lehet így váltogatni. Egy másik ok, hogy különféle kóros EKG alakok is leültethetőek a memóriába, így azok felismerése is tesztelhető lesz. (Ez utóbbi megvalósítása nem célja a szakdolgozatnak)

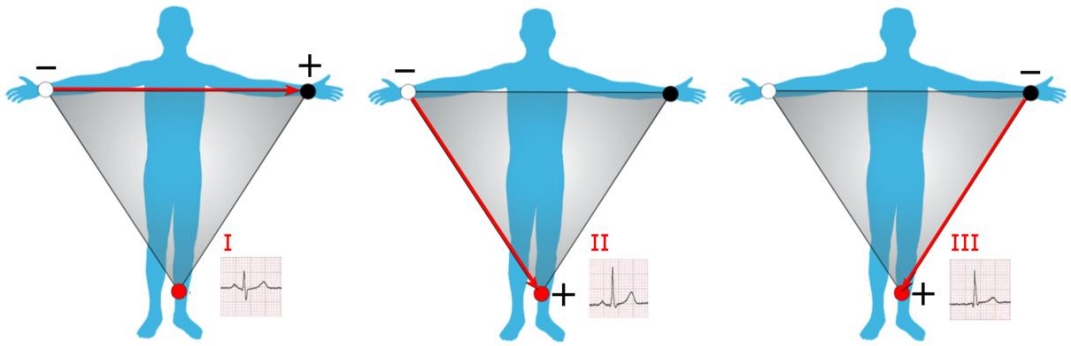
1.3 Az EKG vizsgálatról, az EKG jelről

„A szívműködés megismerését legtöbbször a testfelszínen érzékelhető biopotenciálok (feszültségek) teszik lehetővé, amely az elektrokardiográf (EKG) segítségével válik megismerhetővé.

E vizsgálatok fontosak az emberi szervezet működésének kontrollálásához, a terhelhetőség határainak megfogalmazásához, de különösen fontosak a beteg szervezet állapotának felméréséhez, a terápia meghatározásához, hatásának ellenőrzéséhez.” ([1] 75. oldal)

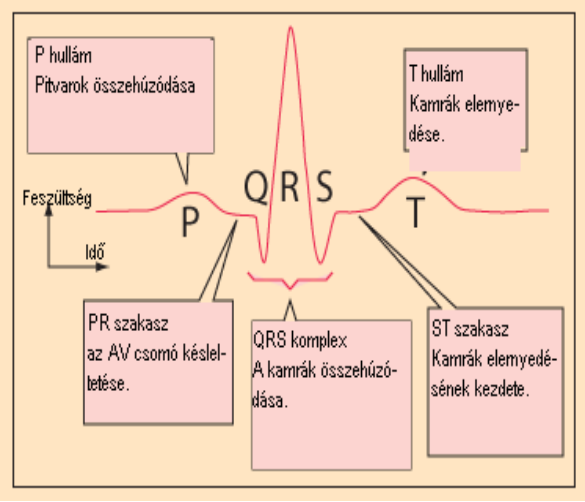
Az emberi szervezetben lévő sejtekben potenciálkülönbségek vannak nyugalmi állapotban. Belül több a negatív töltés, mint kívül. Inger hatására ez a potenciálkülönbség pozitív irányba tolódik, majd visszaáll az eredeti állapotába. Ez előbbi pozitív potenciálkülönbséget akciós potenciálnak nevezzük. A szívbeli összes sejtre nézve és összegezve ezeket a potenciálvektorokat egy úgynevezett integrálvektort kapunk. A szívizomsejtek összehúzódása változik időben és térben is, így az integrálvektor nagysága is iránya is változik. Az integrálvektort nehéz ábrázolni és értelmezni, ezért síkokra (Frontális, vertikális és szagittális) és a síkokon lévő egyenesekre vetítjük ezt, így kapjuk meg a közönséges EKG görbéket. Az elektrokardiográfiát lényegében Willem Einthoven találta fel, aki ezért Nobel-díjat is kapott.

A páciens két kezére (RA, LA) és bal lábára (LF) tett elvezetések között mért feszültség a három legfontosabb jelalak. Az alábbi ábrán láthatjuk ezeket:



1. ábra Einthoven elvezetések

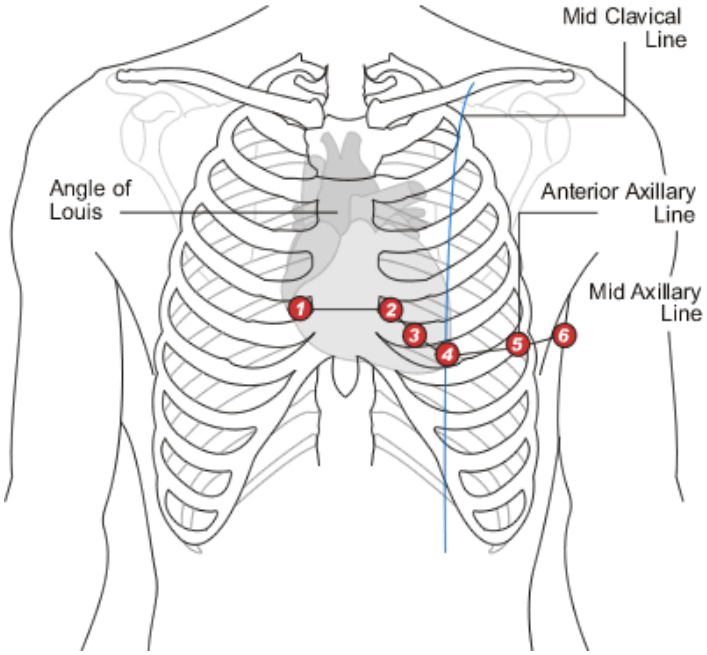
Az I, II és III elvezetéseket Einthoven I, Einthoven II és Einthoven III-nak vagy Einthoven-féle végtagi elvezetéseknek szokás nevezni, a feltalálója után. [2] A végtagi potenciálok európai szabványos jelülése az R (jobb kéz), L (bal kéz), F (bal láb), N (jobb láb); az amerikai pedig: RA, LA, LL, RL ugyanebben a sorrendben, én a szakdolgozat során az amerikai jelölést használom. Ezeknek a potenciálkülönbségeit feleltetjük meg az integrálvektor bizonyos vetületeinek. Bipoláris elvezetéseknek is nevezik őket. Az EKG készülékek ezen pontok közötti feszültségeket mérik és jelenítik meg, rajzolják le.



2. ábra EKG jel

Egy közönséges EKG jel látható balra az ábrán. [3] Legtöbbször az Einthoven II elvezetést szokták ábrázolni. Ezen láthatjuk a szív integrálvektorának alakulását. Ennek a regisztrálásából tudnak az orvosok következtetni betegségesre, elváltozásokra. Például: emelkedett ST szívinfarktusra utal, vagy ha a P hullám bifázisos (negatívba is megy), akkor terheltek a kamrák.

A mellkasra 6 ponton tesznek elvezetéseket (C1, C2 … C6) [3. ábra Mellkasi elvezetések felhelyezésének helye] és ezeknek a potenciálját a végtagi potenciálok átlagához képest mérik: CT=(RA+LA+LF)/3, ezt a számított értéket nevezik „centrál terminálnak” és az így kapott C1-CT, C2-CT … feszültségek lesznek a mellkasi elvezetésekhez tartozó úgynevezett „unipoláris elvezetések”.



3. ábra Mellkasi elvezetések felhelyezésének helye

Van ezen kívül még 3 számított érték, ezek aVR, aVL, aVF, amiket egyik végtagi elvezetés és a másik kettő átlagának különbségeként kapunk, tehát

$aVR=RA-½\cdot(LA+LL)$, így kapunk összesen 12 csatornát. Ehhez összesen 10 elvezetésre van szükség, az eddig említett 3 végtagira és 6 mellkasira, valamint egy árnyékoló vezetékre, amit a jobb lábra szokás tenni.

Az EKG jel (2. ábra EKG jel) egy szívütés során keletkezett jelalakot mutatja, ezek ismétlődnek megfelelő időközönként. Például a 30 BPM azt takarja, hogy 30 szívütés volt egy perc alatt, vagyis 2 másodpercenként volt egy ilyen jelalak, ami ~500ms hosszú, a kettő közötti időben nincs 0.5Hz feletti AC feszültség a mért pontok között.

Az R hullám nagysága általában 1-1,2mV a II. elvezetésen, p-p értéke a jelnek 1,3-1,5mV körül van. Az I elvezetésen 0,2-0,3mV az R hullám, míg a III-on 0,8-0,9mV. Ahogy az 1. ábra Einthoven elvezetések is látszik a II=I+III egyenlet írja le az Einthoven elvezetések közötti összefüggést.

2 Irodalomkutatás

A piacon többféle EKG, őrzőmonitor teszter is van. Különféle szempontok szerint lehet őket osztályozni, összehasonlítani. Egyik alapvető különbség, hogy milyen jeleket lehet rajtuk szimulálni, milyen a frekvenciafelbontásuk, az amplitúdó állítható-e. Másik szempont, hogy milyen egyéb vizsgálójelek állíthatóak be (háromszög, négyszög, szinusz …). Van-e bennük kóros EKG jel, és ha igen, akkor milyenek (például: Kimaradt szívütés, ingerületvezetési rendellenesség, kamrai fibrilláció, pitvari fibrilláció, tachikardia). Van-e más beépített funkció, értem ezalatt a vérnyomás, légzés, hőmérséklet jeleket; ez utóbbi funkciók már nem a sima EKG teszterekre érvényesek, hanem az őrzőmonitor vizsgáló készülékekre. Milyen a kezelőfelülete, akkumulátorról működik vagy hálózati táplálású, mekkora a zaj a kiadott jelen, mennyibe kerül. Alább egy táblázatban összefoglaltam három, ma a piacon lévő készülék tulajdonságait.

Típus	**TechPatient CARDIO(350$) [5]**	**Fluke PS410 [6]**	**Netech MiniSim 1000 [7]**
Tápellátás	9V akkumulátor vagy 9VDC adapter	9V akkumulátor vagy 9VDC adapter	9V akkumulátor vagy 9VDC adapter
Kezelőfelület	Nyomógomok (4 db)	Nyomógomok (6 db)	Nyomógombok (8)
ECG BPM (Beat per minute)	20-tól 300-ig	30, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300	30, 60, 70, 80, 90, 100, 120, 150, 180, 210, 240, 270, 300, 350
Más vizsgáló jelek	Szinusz, négyszög, háromszög, impulzus 1/8 Hz-től 120 Hz-ig DC	Szinusz (0.5, 5, 10, 40, 50, 60 Hz), Négyszög (2, 0.125 Hz), Háromszög (2 Hz), Impulzus (30, 60, 120 BPM; 60 ms aktív)	Szinusz, négyszög, háromszög, impulzus 0.1 - 0.9 Hz (0.1 Hz) 1.0 - 9.0 Hz (1.0 Hz) 10 – 100 Hz (10 Hz) (zárójelben a lépésköz)
Kóros EKG jelek	Nincs	Több mint 40 féle	Több mint 20 féle
Amplitúdó (kettes elvezetés)	0.5, 1, 2, 4 mV	0.5, 1, 2 mV (csak ECG-re állítható)	0.15, 0.3, 0.5, 1.0, 2.0, 3.0, 4.0, 5.0 mV
Kijelző	Kétsoros LCD	Kétsoros LCD	-
Ár	350$	NA	694$

Az újabb EKG teszterekről összegyűjtöttem több tulajdonságot, amiből párat fel is használok a szakdolgozat során. Az alább felsorolt funkciók és lehetőségek is az eszközök céljaihoz igazodnak, van ahol csak egy konkrét típusú eszköz tesztelésére kell egy másik készüléket alkalmazni, ekkor nincs szükség olyan beállítási lehetőségekre, amiket a tesztelendő készülék ki sem használ. Vannak viszont olyan berendezések is, amiket szélesebb körben alkalmaznak, így törekedtek azokat minél több beállítási opcióval elkészíteni. Ilyen például, hogy a memóriából kiolvasott adatokat hány bites DA-val dolgozzuk fel, mert ha a tesztelendő EKG csak 8 bites, akkor nincs szükség jóval nagyobb felbontású DA átalakítóra.

Funkciók:

- USB-n PC-ről jön a beállítás vagy akár maga a kiadandó jel is.
- Galvanikus leválasztás (általában optocsatolóval).
- EKG jel memóriából (EEPROM vagy FLASH), a négyszög, háromszög, szinusz előállítása lehet memóriából (egyszerűbb) vagy külön áramkörrel (pl: XR-2206, akár analóg áramkörrel is), ezzel a kvantálási zaj eltűntethető.
- USB/UART konverzió külön IC-vel. (FT232R USB UART IC)
- LCD, Touch screen
- Hangjelzés
- Állítható kimeneti amplitúdó

A fentebbi listából felhasználtakat a specifikáció tartalmazza.

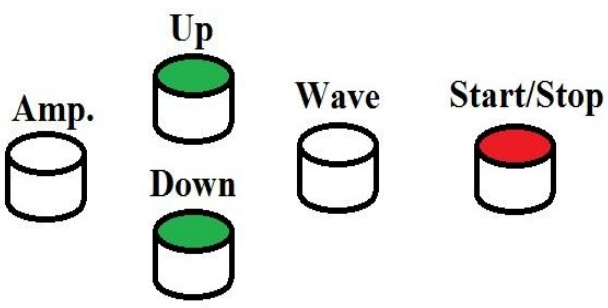
3 Specifikáció

A szakdolgozat specifikációját elemzem és egészítem ki a szükséges részekkel a következő részben.

12 csatornás EKG-ket akarunk tesztelni, ehhez 10 kimeneti pontra van szükségünk a korábban említett okból; ezek: 3 végtagi (R, L, F), 6 mellkasi(C1…C6), árnyékolás (N). A referencia amplitúdó, ami a mi esetünkben 1mV a II. elvezetésre kerülő jelalakhoz tartozik, azaz az R és az F pontok közé lesz kivezetve (F a nagyobb potenciálú). Az R és L kimenet közé is hasonló a jelalak kerül (I. elvezetés), viszont itt már az amplitúdó kisebb, embertől függően 20-30%-a a II. elvezetésnek. Nem mindig szoktak külön jelformát használni a két elvezetéshez, viszont mivel én úgyis külön DA csatornát használok hozzájuk (lentebb részletesen), ezért lehet eltérő hullámformával dolgozni. A III. elvezetést nem kell előállítani, mivel ez az L és F potenciálokból kiadódik. Kiegészítésként a kimeneti amplitúdót is lehet majd állítani, 0.5, 1, 2 mV értékekre. Ez csak az I. és II. elvezetésekre (ebből következően a III-ra is) lesz igaz, a mellkasi elvezetések standard értéket kapnak.

A specifikációban fel van sorolva a beállítható ECG, szinusz, háromszög, négyszög jelalakok frekvenciája, ezeket a deszkamodellben megvalósítottam. Az ECG jeleket ezenfelül, 30 és 240 BPM között tízes lépésközzel lehet állítani. A szinusz (40, 50, 60, 100Hz), a háromszög (2Hz) és a négyszög (0,125 Hz) a specifikációban megfogalmazott marad.

A kezelő gombokból 5 darab lesz. Három ezek közül a jelalakok kiválasztását biztosítja, tehát az egyikkel (Wave) lehet a formák között váltani, csak egyirányú léptetés lehetséges, a másik kettővel a frekvenciák között lehet lépkedni fel és le (Up, Down). A negyedik gomb az indításért és leállításért felelős (Start/Stop). Ezenfelül az amplitúdó állítását lehet kiválasztani egy gombbal, ekkor az Up és Down gombokkal lehet léptetni.



4. ábra Nyomógombok

A kétsoros LCD kijelzőn az éppen aktív jelalak jellemzői fognak látszani, tehát a hullámforma és a frekvencia. Emellett egy karakter jelzi, hogy éppen beállítás zajlik vagy már a kimeneten megjelent a jelalak.

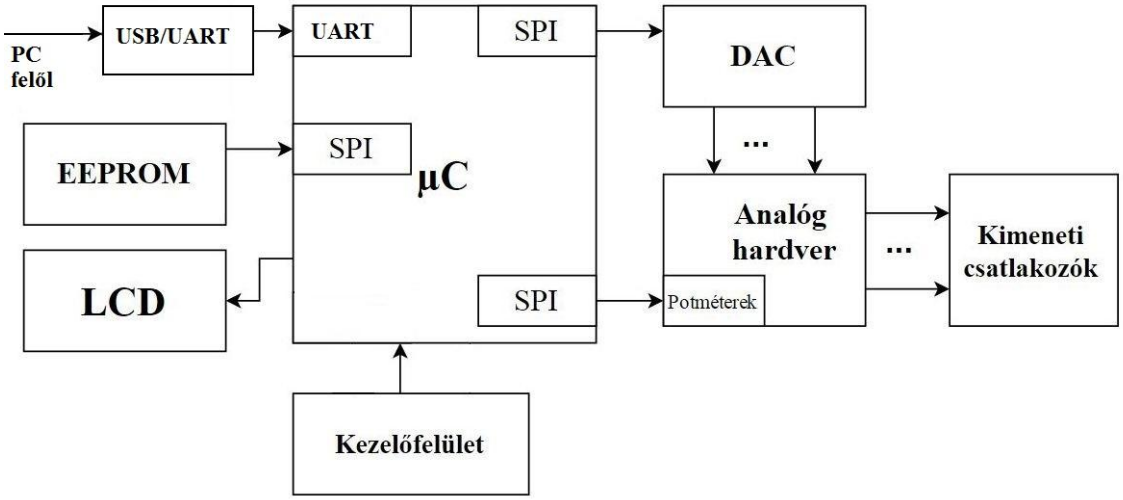
A PC-s kapcsolat USB-n keresztül fog megvalósulni, itt is beállíthatóak lesznek a jelek. A PC-ről jövő beállítás prioritást fog élvezni a nyomógombokkal szemben.

Az eszközt szeretnénk hordozhatóra megcsinálni, így az akkumulátoros/elemes működés, amit használni fogunk.

4 Az eszköz tervezése

A követelmények és specifikáció tisztázása után a következő lépés már lehet az eszköz tervezése. Az eszközünk egy mikrovezérlővel működtetett készülék lesz, ahol a felhasználói beavatkozás értelmezése, a PC-vel való kommunikáció, a jelek előállítása, memóriából való kiolvasása és a jelalakok kimenetre adását is egy PIC gyártmányú mikrokontroller végzi. Az idő szűke miatt, mivel a nyáktervezés, ennek a legyártása, alkatrészek beszerzése nem fért volna bele; így csak egy deszkamodell megvalósítására került sor. Az elkészített eszköz kapcsolása, szoftver felépítése lentebb lesz található, de előbb a végső eszköz részeit nézzük.

4.1 Hardver tervezés



5. ábra Blokkdiagram

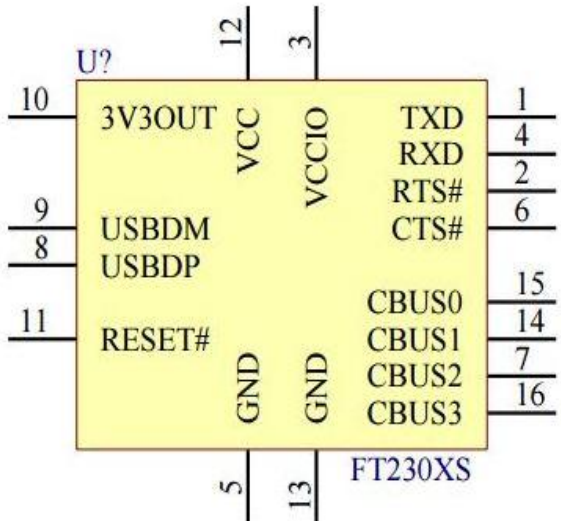
A fentebbi blokkdiagram mutatja, hogy a nagyobb egységek hogyan kapcsolódnak egymáshoz. Részletesebben szétszedve a komponensek:

A központi egység egy PIC mikrokontroller. A cég, ahol a szakdolgozatomat készítettem, már több éve ezt a mikrovezérlő családot használja, valamint ez szerepelt is megkötésként a kiírásban, hogy PIC kontroller egység legyen az eszköz vezérlője. Emellett, a Microchip sokféle funkcióval rendelkező csipeket gyárt, így az igényeknek megfelelően lehet belőlük válogatni. Amint fentebb is látható szükségünk van SPI és UART egységre, ezenfelül 4 gomb, egy 2 soros LCD kijelző, digitális potenciométer (a kimeneti amplitúdó állításához), *chipselect* jelek kezelését is el kell látni, ehhez pedig

szükségünk van megfelelő számú I/O lábra. Számolás: EEPROM 4 láb, LCD, 11 láb, DAC, 5 láb, potenciométerek 4 láb, oszcillátor 2 láb, UART 4 láb, tehát legalább 30 lábú IC-re van szükség. A vizsgálójelek megfelelő időközönkénti kiadásához szükségünk van időzítőre. Számolni kell azzal, hogy a memória olvasása (30µs volt a deszkamodellnél 8 bit olvasására, nekünk pedig minimum 40 bitet kell egyszerre felhozni, számolás lentebb) és mind a 8 DA csatorna írása (min. 12µs programkód futása nélkül, a pontos számolás lentebb a DA résznél) időt vesz el, emellett még a program futása is időbe telik, így biztos, ami biztos alapon legyen legalább 50 MIPS-es az MCU. A Microchip oldalán található selectorral [51] leszűkítettem a kört. Ezután esett a választás a PIC32MX695F512H típusú csipre. A választásban még szerepet játszott, hogy az így is túlméretezett (gyorsaság, lábszám) eszközök közül azt válasszam, ami kevesebb plusz munkát jelent, tehát a 64 lábú eszköz elegendő lesz a feladatra, a legalább 100 lábúakhoz képest.

4.1.1 PC-s kommunikáció / USB

A PC-vel való kommunikáció során USB/UART átalakítót fogunk használni, így a bonyolult USB protokoll működését elfedi nekünk a konverter. Erre a feladatra az FTDI Chip által gyártott FT320X típusú IC-t választottam, ez az eszköz egyszerűen konfigurálható (PC-n keresztül USB-ről) és olcsón beszerezhető ($2).



6. ábra FT2300x lábkiosztás

Az IC bekötésénél a GND lábakat a földre; a V_{CC} lábat az USB csatlakozó 5V-jára (V_{BUS}); a V_{CCIO} és a /RESET lábakat a $3V3_{OUT}$ lábra kötjük (adatlap 9. oldal). Ezzel a megoldással kikapcsoltuk a reset funkciót. Az USBDM és USBDP kivezetéseket az USB csatlakozó D- és D+ lábaihoz vezetjük. A TXD és RXD elvezetéseket a mikrokontrollerünk U5RX és U5TX portlábaihoz vezetjük. Az RTS, CTS és CBUS lábakat nem kötjük be. Az IC egyéb lehetőségeit (receive, transmit buffer, tápfeszültség előállítása) nem használjuk ki.

4.1.2 Nem felejtő memória, EEPROM

A következő komponens a memória kiválasztása, ehhez a cégnél javasolt 25LCxxx EEPROM családból választottam. Ez a család 2.5V - 5.5V táp és magas jel tartományban tud működni, tehát a 3.3V-os PIC-hez egyszerűen illeszthető. A kapacitás meghatározásához meg kell nézni, hogy milyen jeleket akarok eltárolni. Ehhez tudnunk kell, hogy az EKG eszközök mintavételezése mekkora frekvenciával történik. A legtöbb EKG készülék 500 Hz-cel veszi a mintákat, azaz 500 mintát vesz másodpercenként. A mi digitális-analóg átalakítónk 12 bites lesz. Tudjuk, hogy az I. és II. elvezetéshez, valamint a mellkasiakhoz szeretnénk mintákat tárolni, ez utóbbihoz egy jelsorozatot szokás tárolni, tehát a 6 mellkasi elvezetéshez nincs külön-külön minta, hanem ugyanaz a jelalak jelenik meg a kimeneten, így ezekhez elég egy eltárolt jelsorozat. Mindegyik EKG periódusban az aktív rész, tehát ami a P és a T hullám (esetleg U hullám, ezt nem szokás beletenni, de igény szerint ilyen minta is tárolható) között van, bőven 1 másodpercen belüli (250-650ms), tehát a 30 bpm-hez tartozó 2s periódusidőhöz sem tárolunk 2 másodpercnyi mintát, hanem csupán az aktív szakaszt olvassuk a memóriából, a „passzív” részt szoftverből kezeljük. Legrosszabb esetben is ez azt jelenti, hogy

12 [bit]·3[csatorna]·500[minta/másodperc] = 22500 bit.

A szinusz jel esetén a legkisebb frekvencia a 10Hz, ehhez a 100ms-os periódusidőhöz 50 mintára lesz szükségünk, ami *50*12=600 bitet* jelent, mivel mindegyik csatornára ugyanaz a hullám kerül. A háromszög 2 Hz-es, ami azt jelenti, hogy ½ másodperc alatt kell kiadni 250 mintát, ami *250*12=3000 bitet* jelent. A négyszög előállítása szoftverből megoldható, tehát ehhez nem kell memóriát használni. Ez összesen 26100 bitet jelent. Legalább 32Kbit kapacitású adattároló kell. Jobban megvizsgálva a 25LCxxx típusokat, láthatjuk, hogy a kapacitások közötti eltérés nem lassít vagy gyorsít a hozzáférésen, mivel mindegyiknél egy 8 bites parancsszó és egy 16 bites cím szükséges ahhoz, hogy a memória kiadja a benne levő adatot. A nagyobb kapacitású EEPROM-ok ára sem nagyon különbözik, így inkább egy nagyobb kapacitású 256kbit=32Kbyte kapacitású IC-t választottam, ez a 25LC256. Választásomban szerepet játszott, hogy ha esetleg később igény lenne rá, akkor mind a 6 mellkasi elvezetéshez más és más jelalak tárolható; valamint egy másik lehetséges továbbfejlesztés, hogy a számítógépről ne csak beállítások, hanem adatok is érkezhessenek; ezeket letárolva a memóriában, később fel lehet használni EKG

tesztelésre. Az előbbi két szempontot figyelembe véve még hasznát vehetjük a nagyobb tárolási helynek.

Nézzük meg, hogy az adatok kiolvasása mennyi időt fog igénybe venni számunkra! A tárolónak 10MHz a maximális működési frekvenciája. (A 80MHz-es MCU ezt elő tudja állítani). 8 bit „read" utasítás, 16 bit cím, majd az adat olvasása következik. Tudjuk előre, hogy 12 bitesek lesznek az tárolt mintáink; az I-es és II-es elvezetéshez külön-külön tárolunk mintát, mert ezek a valóságban is nagyban különböznek, a 6 mellkasi elvezetéshez viszont csak 1 mintát tárolunk (az életben sem nagyon különbözik), akkor 3·12=36 bitet kell egy ütemben felhoznunk, ezt ki tudjuk olvasni folytonosan, ha előtte a megfelelően megcímeztük a memóriát. – A tárolást meg tudjuk oldani folytonosan, tehát a 36 bitet 5·8 bitként olvassuk fel és ebből az utolsó 4-et eldobjuk, majd a PIC feldolgozza, hogy melyik bit melyik kimenethez tartozik. Ekkor az EEPROM tizede kihasználatlan marad, de ez még belefér. – Ez összesen 8+16+5·8=64 bitet (parancs+cím+adat) jelent. (1/10MHz)·64=6.4µs. Ez bőven megfelel a 100Hz-es szinuszhoz tartozó 200µs-os periódusidőnek, ami a minták kiadásának sűrűségével egyezik (a 200µs számolása a 4.1.4 pontban szerepel). Ha később mind a 6 mellkasi ponthoz külön jelalakot szeretnénk, akkor 8+16+96=120 bit; (1/10MHz)·120=12µs szükséges az olvasáshoz, még ez is belefér az időbe.

Az EEPROM-mal való kommunikációhoz mindössze 4 vezetékre van szükség, ezek az alábbiak:

SI: Serial Data Input, bemeneti vonal

SO: Serial Data Output, kimeneti vonal

SCK: Serial Clock Input, soros órajel bemenet; a PIC lesz a master, tehát az órajelet is majd az szolgáltatja.

/CS: Chip Select Input, IC kiválasztó vonal

Ezek a ki- és bemenetek biztosítottak a mikrokontroller által. Az vonalakat a következőképpen rendeljük egymáshoz:

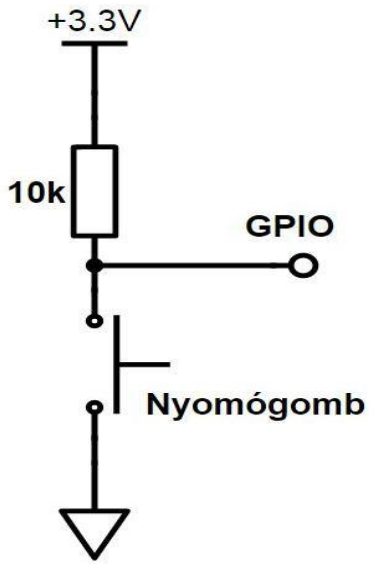
EEPROM	SI	SO	SCK	/CS
MCU	SDOx	SDIx	SCKx	/SSx

Ahol az „x" az SPI egység számát jelöli a mikrovezérlőnkben, mivel 3 ilyen egység is található benne, '2'-vel, '3'-mal és '4'-gyel jelölve. Az hogy melyikre kötjük nem jelent

sem előnyt, sem hátrányt, viszont figyelembe kell venni, hogy a 4-es SPI modul órajel kimenetét már elhasználtuk az UART-hoz, mivel valamelyik SPI modult mindenképp fel kellett használni az UART-hoz. Én a 2-es elnevezésűt választottam az EEPROM-hoz. Az EEPROM többi lábának bekötése: V_{cc} a 3.3V-os tápvonalra megy, V_{ss} a földre, a /HOLD és a /WP lábat is 3.3V-ra kötjük, mert ezeket a funkciókat nem használjuk.

4.1.2 Kezelőfelület

A kezelő felületünk 5 nyomógombból (Amp., Wave, Up, Down, Start/Stop) fog állni. A gombok közönséges GPIO kivezetésre köthetőek. A Start/Stop gomb lesz az, amit figyelni kell tesztelés közben, tehát amikor már a jeleket adjuk ki, hogy jött-e stop jel (ezt szoftverből valósítjuk meg). A nyomógombok pergésmentesítését szintén szoftverből kezeljük. Egy gombhoz tartozó hardver az ábrán látható módon valósul meg.



7. ábra Nyomógomb

4.1.3 LCD kijelző

A megjelenítő eszközünk egy 2 soros karakteres LCD kijelző lesz. Ehhez a kijelzőhöz 16 vezeték csatlakozik. Ebből 2 tápvezeték V_{ss} és V_{dd}, ezek kapják a 0 és a 3.3V-ot. Harmadik a kontraszt beállításért felelős V0; ezt tipikusan egy 10-20kΩ-os potenciométerre kötik (mi 10kΩ-ot használunk), aminek két szélső lába a tápfeszültségen és a földön van; 0.5V körülire állított érték az, aminél jól látszódnak a kijelzett karakterek. Ezt követi, három vezérlőláb. E engedélyező láb, R/W olvasás /írás választó, RS adat/parancs választó, ez utóbbival lehet kiválasztani, hogy parancsot (alacsony) vagy adatot (magas) küldünk a kijelzőnek. Emellett még van 8 adatvonal, tehát összesen 11 vonalat kell bekötnünk a mikrovezérlőnkbe és ezek mind kimenetek lesznek az MCU felől nézve. Háttérvilágításhoz bekötjük még a LED+ lábat a 3.3V-ra egy 330 Ω-os ellenálláson keresztül és a LED- lábat a földre.

4.1.4 Digitális-analóg átalakító

A digitális jelek analóggá alakítását egy MAX5306EUE típusú DA átalakító végzi. SPI protokollon keresztül kommunikál a vezérlőegységgel, 8 darab

feszültségkimenete van, így elegendő a 2 végtagi és a 6 mellkasi kivezetésünkhöz, mivel az R csatlakozó a földponthoz lesz vezetve. A konverter 15MHz-cel tud működni. Egy DA egység írásához 16 bitre van szükség, ebből 4 vezérlőbit és 12 adatbit. Az /LDAC lábát meghajtva az átalakító mind a 8 bemeneti regiszteréből egyszerre írja az adatot a kimeneti regiszterébe, így egyszerre frissíthetőek az adatok. Egy teljes írási ciklus időigénye a következőképpen alakul: $1/15M[Hz]\cdot 16[bit]\cdot 8[csatorna]\approx 8.5\mu s$ Ezzel az idővel bőven benne vagyunk a 2ms-os mintafrissítésben, ami az EKG-k 500Hz-es mintavételéből következik. Ha ehhez hozzávesszük, hogy a szinusz korábban kiszámolt 50 mintájával dolgozunk, amit 10 Hz-hez számoltunk, akkor ez 100 Hz-nél tizedakkora időt jelent, azaz 200µs-ot mintánként, de még ez sem kritikus. Hasonlóan az EKG jelet is sűrűbben kell kiadni, ahogy növeljük a percenkénti szívütésszámot; ebben az esetben viszont nincs 10szeres ugrás, ezért ez sem lesz kritikus. Megnézve a mikrokontroller adatlapját, láthatjuk, hogy ha 80MHz-es kristályoszcillátorról működtetjük, akkor 10MHz-et tudunk beállítani az SPI kommunikáció órajelének; ezért az előbb kiszámolt 8µs helyett 12µs lesz a legkisebb frissítései idő, azonban erről is láthatjuk, hogy nem lesz emiatt időkritikus a rendszer, ehhez hozzájön még az EEPROM olvasása és a programkód futtatása is, ez utóbbiról a megírás és fordítás után lehet pontos időt mondani.

Ez az eszköz 16 lábú; ebből 2 tápfeszültségre kapcsolódik, ami 0V és 3.3V lesz; egy /CS kiválasztó jel, ami a mikrokontrollerbe megy. Egy D_{in} és egy D_{out} vonala van, ezek közül a D_{out}-ot nem használjuk, ezért 10kΩ-mal a földre kötjük, a Din lesz a soros adatfogadó vonal, ez is a mikrovezérlőhöz kapcsolódik. Az SCLK láb is a mikrovezérlőbe megy és a korábban említett /CS és D_{in} lábakkal együtt a hármas SPI egységhez kapcsolódik hasonlóan az EEPROM-hoz.

DAC	D_{in}	D_{out}	SCLK	/CS	-
MCU	SDO3	-	SCK3	/SS3	SDI3

A korábban említett /LDAC láb, ami szinkronitásért felelős a mikrovezérlő egy GPIO lábára lesz kötve. A 8 kimenet mindegyike egy-egy műveleti erősítőre lesz kötve, hogy bifázisos jelet kapjunk, az ehhez szükséges kapcsolás a következő:

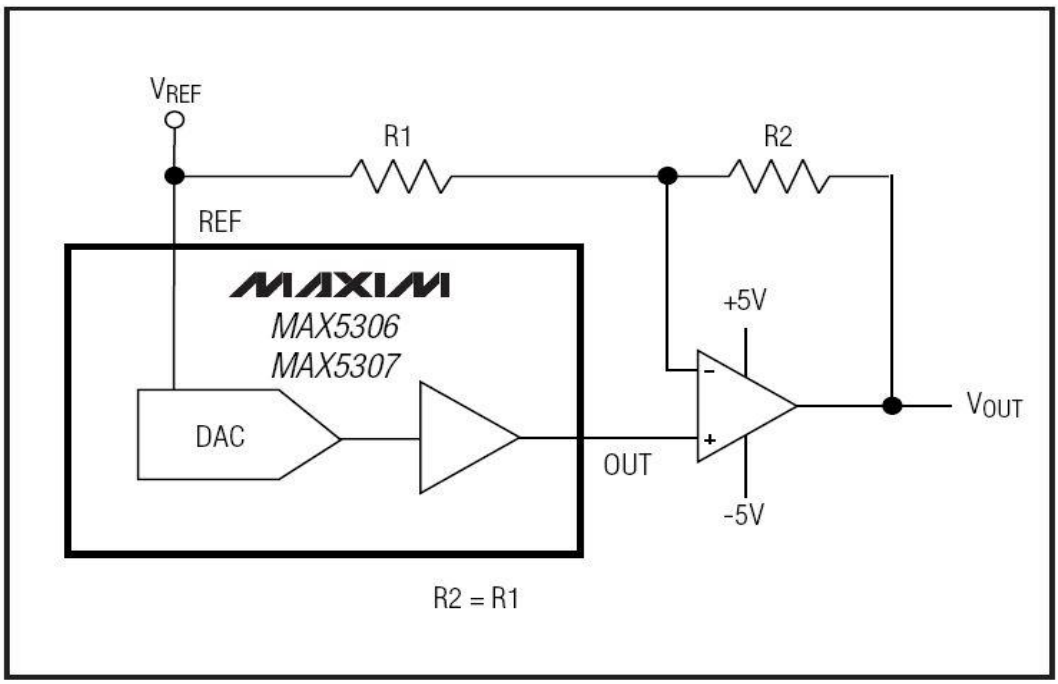
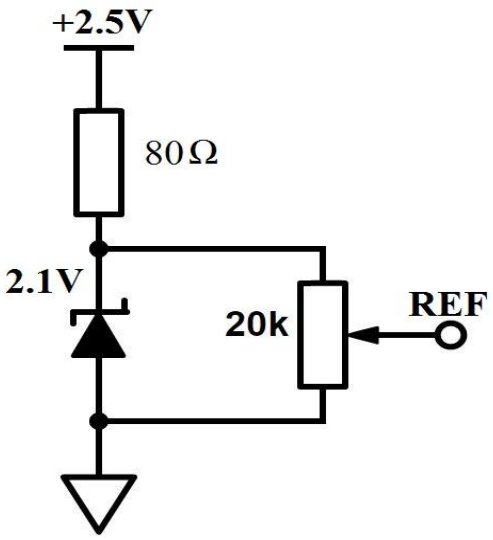


Figure 7. Bipolar Output Circuit

8. ábra DAC bifázisos kapcsolás

Ez a kapcsolás az adatlapban szereplő hivatalos kapcsolás. A mi esetünkben viszont nem ±5V lesz az erősítő tápja, hanem ±2.5V, ennek indoklása és megvalósítása lentebb olvasható. Műveleti erősítőnek 2 darab MCP6054 típusú, csipenként 4 darab erősítőt tartalmazó IC-t választottam. Ezt alacsony feszültségű (V_{dd}-V_{ss}=1.8V – 6.0V) használatra tervezték. Kis zaja miatt választottam ezt az eszközt, néhányszor tíz nV/√Hz-es a bemeneti zaj értéke. R1-et és R2-t 5kΩ-ra választottam, ezzel nem nagyon terhelem az áramkört, viszont nem túl nagy ahhoz, hogy korlátozza a működést.

A REF lábra egy analóg módon előállított referenciafeszültség lesz kötve, a kapcsolás lentebb látható. Egy 2.1V-os Zéner-dióda állítja elő a stabil 2.1V-ot a +2.5V-ból (ez utóbbi előállítását később tárgyalom a tápfeszültségek résznél). A dióda 5mA áram esetén szolgáltat stabil feszültséget. A fölötte levő ellenállásra esik 0.4V és folyik rajta ez az 5mA, ebből kiadódik a 80Ω. A 20kΩ-os potenciométeren folyó 2.1V/20kΩ=105µA nem befolyásolja jelentősen a feszültségviszonyokat. Ezt a potmétert állítjuk úgy, hogy a 2.04-2.05V legyen a REF pont feszültsége, ezt pedig a DA konverterre vezetjük.



9. ábra Referenciafeszültség

4.1.5 Analóg hardver

A következő rész az analóg hardver. Az amplitúdóállító potenciométereket leszámítva itt már nincs digitális eszköz.

Ahhoz, hogy tudjuk milyen jelet akarunk kapni a kimeneten meg kell vizsgálnunk, hogy az emberi szervezet milyen kimenetet produkál, mind feszültség, mind impedancia szempontjából, valamint az EKG készülékeke milyen módon mérnek.

Ha megnézzük egy embernek az impedanciáját, akkor láthatjuk, hogy ez átlagosan 50-100 kΩ-os nagyságrendbe esik. Ez természetesen csak a nagy többség, van ahol ez kisebb, van ahol nagyobb; ez függ az elektródák felhelyezésétől is, az újabb EKG készülékek ezt mérik is, így tudnak jelezni, ha esetleg az elektróda leesik vagy elmozdul a páciensen. Logikusan következik, hogy a végtagi elvezetések között mért impedancia nagyobb, mint a mellkasi elvezetések között mért, de nem térnek el egymástól jelentősen.

Az EKG készülékek belsejében egy ellenálláshálóval találkozunk, erre kapcsolódnak a páciensről jövő elvezetések, ebben MΩ nagyságrendű elemek helyezkednek el és a mérést is ilyen nagyságrendbe eső bemeneti ellenállású eszközökkel végzik. Így a páciensről jövő áram értéke maximum pár nA-es nagyságú. Az elektródákkal sorba szoktak kötni 1-10kΩ ellenállást a páciens kábelben a nagyfrekvenciás vágó zajának csillapítása okán [1]. Az általános feszültségérték, amit az embereken mérni szoktak a

0.5mV és 2mV közé esik. Ennek nagyságából is lehet betegségre, problémára következtetni. Meg kell jegyezni, hogy ez az érték a 0.5Hz és 150Hz közötti tartományra igaz, a modernebb EKG készülékek emiatt tartalmaznak beépített analóg vagy digitális szűrőket, amik a DC szintet levágják; emellett 75Hz vagy 150Hz-es aluláteresztő szűrő is van bennük. A 150Hz egy orvostechnikában általánosan elfogadott érték, az ezen frekvencia alatti komponensek azok, amik értékesek a kiértékelés során, a nagyobb frekvenciájú összetevők már nem hordoznak lényegi információt. Az EKG készülékben, amivel volt lehetőségem dolgozni volt 75Hz-es és 150Hz-es szűrő is és lehetett közöttük váltani a kezelőfelületen keresztül. Ebben alapbeállítás volt a 0.5Hz-es felüláteresztő szűrő, amit nem is lehetett kikapcsolni. Itt jegyezném meg, hogy beállítható volt 50 vagy 60Hz-es lyukszűrő is a hálózati zaj szűrése miatt, ez a funkció az eszközökben általában alapbeállítás, esetleg ahol tudják, hogy melyik piac (pl.: Európai vagy Amerikai) a célja a készüléknek, ott csak az ottani hálózati frekvenciának megfelelő szűrőt építik be.

Ezeket figyelembe véve a tervezésnél láthatjuk, hogy nem kell nagy terhelőárammal számolnunk.

A DA utáni következik mind a 8 vonalra egy RC aluláteresztő szűrő. Azért passzív szűrőt választottam, hogy ezzel is kizárjam az aktív elemek saját zaját. A korábban is említett 150Hz-es törésponti frekvenciához $150Hz \cdot 2 \cdot \pi \approx 942.5\ rad/s$, amihez τ=1.06 ms időállandó tartozik. A műveleti erősítőnél használt 5kΩ itt is megfelelő lesz, ezzel nem is terheljük nagyon a korábbi fokozatot, viszont szolgáltat elegendő áramot a kimenet felé és nem kell hozzá túl kicsi vagy túl nagy kapacitású kondenzátor:

$1.06ms/5k\Omega = 212nF$. Mivel a kereskedelemben 220nF-os kondenzátor kapható, ezért ezzel valósítjuk meg a szűrőt; ehhez $220nF*5k\Omega = 1.1ms$ időállandó és $1/(1.1ms*2*\pi) = 144.7Hz$-es törésponti frekvencia tartozik. Ez még megfelelő a célkitűzésben.

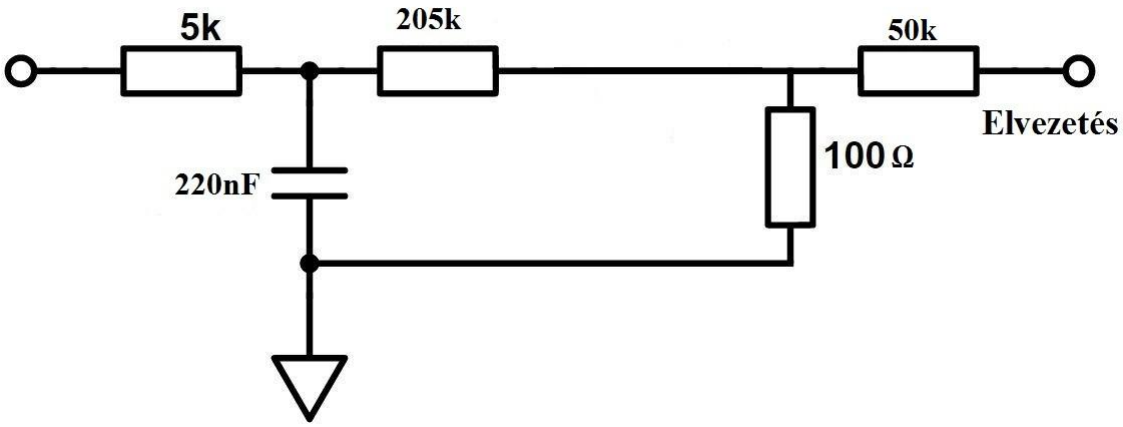
Ezt a szűrt jelet fogjuk a következő lépésben a megfelelő nagyságrendűre osztani. Ehhez közönséges feszültségosztót használok. Az ide érkező jelem ±2.04V-2.05V nagyságú és ezt akarom ±1mV-ra osztani a mellkasi elvezetéseknél, a két végtaginál pedig ±0.5mV, ±1mV és ±2mV nagyságúra. Ez azt jelenti, hogy 2.04V-2.05V és 1mV-ra kell szétosztanom a feszültséget a mellkasiaknál. Ha a kisebbik ellenállás 100 Ω körüli és a nagyobbik párszáz kΩ, akkor azzal nem tévedhetünk. Legrosszabb esetben 0.5mV/100Ω=5μA folyik a kisebbiken, ez 3 nagyságrenddel

nagyobb az EKG-be folyó áramnál, tehát nem fogja elrontani a mérés a beállított értéket. Tudjuk még, hogy a végtagi elvezetés amplitúdóját változtatni akarjuk, ehhez digitális potenciométert használunk, ez kerül a kisebbik ellenállás helyére. A választás az Analog Devices gyártmányú AD8402ARZ1 típusú 1kΩ-os potméterre esett. Megnézzük, hogy ezen milyen érték állítható be, ami 50Ω, 100Ω és 200Ω körüli. 256 értékre lehet állítani a „csúszkát", ebből *1000Ω/256=3.90625Ω* adódik, ezt 13-mal szorozva 50.78125Ω adódik, ami nekünk megfelel. Ekkor 0.5mV csúcsérték esik az ~50Ω-on és 2.04V esik a nagyobb ellenálláson, ebből számolva a nagyobbik ellenállás: *4095*50Ω≈205kΩ*. Ez fix érték lesz, innen kiszámolható a beállítandó érték a ±1 és ±2mV-hoz, ami 100Ω és 200Ω körül kell, hogy legyen, mivel kétszer és négyszer akkora feszültséget várunk. *1mV/(2.04V/205kΩ)=99.5Ω* és *2mV/(2.04V/205kΩ)=199Ω*. Kijött, amire számítottunk, látszik, hogy a nagyságrendbeli különbség miatt elhanyagolható az eltérés. A mellkasi elvezetésekhez is ugyanezt a megoldást választom, csak ott a 100Ω fix érték lesz.

A digitális potenciométerek bekötése a következőképpen történik: A V_{dd} és a D_{GND} a digitális tápvonalakra (3.3V és 0V) kapcsolódik; SDI, CLK és /CS vezetékek a PIC vezérlőbe mennek a 2-es számú SPI egységhez, ahova az EEPROM is csatlakozik. Itt megjegyzem, hogy ekkor a mikrovezérlő 2 egységet hajt meg egy-egy lábával, de ezt bírnia kell az IC-nek. Az egység 10MHz-cel fog működni, mint az EEPROM, ennek oka, hogy így nem kell átparaméterezni az SPI kommunikációt EEPROM használat előtt/után. B1-et, B2-t és A_{GND}-t a földre kötjük; /SHDN és /RS lábakat pedig szintén a 3.3V-ra mivel nem használjuk.

A korábban említett páciens impedancia miatt biztosítanunk kell az ~50kΩ-os kimeneti ellenállást, amit egyszerűen a kimenetekkel sorosan kötve egy-egy 50kΩ-os ellenállással valósítok meg. A másik két végtagi potenciál a jobb kéz és az árnyékolásra szolgáló jobb láb feszültségszintje a rendszerünk GND-jéhez fog csatlakozni, ezekhez is egy-egy 50kΩ-os ellenállást kapcsolunk sorba.

A DA bifázisos kimenetét követő hardver a következőképpen néz ki:



10. ábra Kimeneti hardver

Az L és F kimenetek esetén annyi eltérés van, hogy a 100Ω helyén egy digitális potenciométer helyezkedik el.

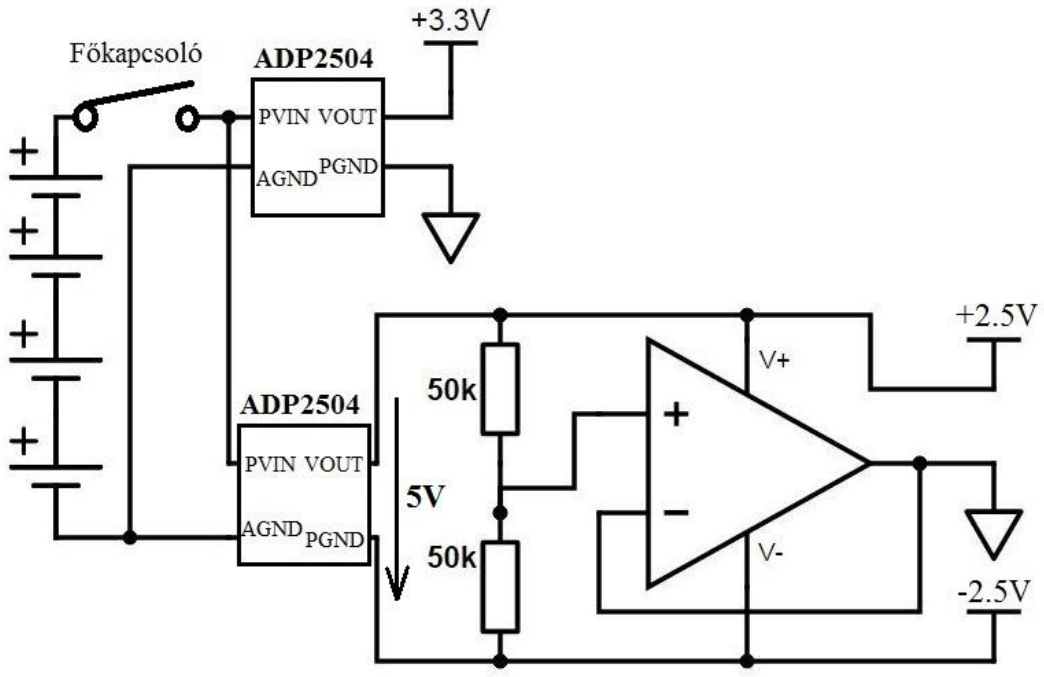
4.1.6 Tápvonalak

A tápfeszültségek előállításának tervezéséhez nézzük milyen feszültségekre van szükségünk és az eszközeinknek milyen áramfelvétele van! Ezt a következő táblázat mutatja:

	MCU	EEPROM	Digitális potenciométer	DA átalakító	LCD
Feszültség [V]	2.3–3.6	2.5–5.5	2.7–5.5	2.7–5.5	3.1–3.5
Maximális áramfelvétel [mA]	300	6	4·2db	33	2.5 (18.5 háttér-világítással)

Összegezve a maximális áramfelvételt kevesebb, mint 400mA-t kapunk, ez csak a digitális részre igaz. A PIC áramfelvételéhez hozzá kell tenni, hogy ez csak impulzusszinten lenne meg, ha kihasználnánk az eszköz lehető legtöbb funkcióját, a mi esetünkben az effektív érték 10mA körül fog maximum mozogni. Az FT230X IC áramfelvételét azért nem számoltam, mivel azt az USB táplálja a PC felől. A feszültségeket megvizsgálva láthatjuk, hogy a 3.3V-os tápfeszültség megfelelő lesz az eszközeinkhez. A korábban említett ±2.5V-os feszültségre azért lesz szükségünk, hogy elő tudjuk állítani a bifázisos jelalakokat. Persze ennél nagyobb vagy kisebb feszültség is megfelelő lenne, nagyobbra viszont nincs szükségünk, mert ez a feszültség elegendő

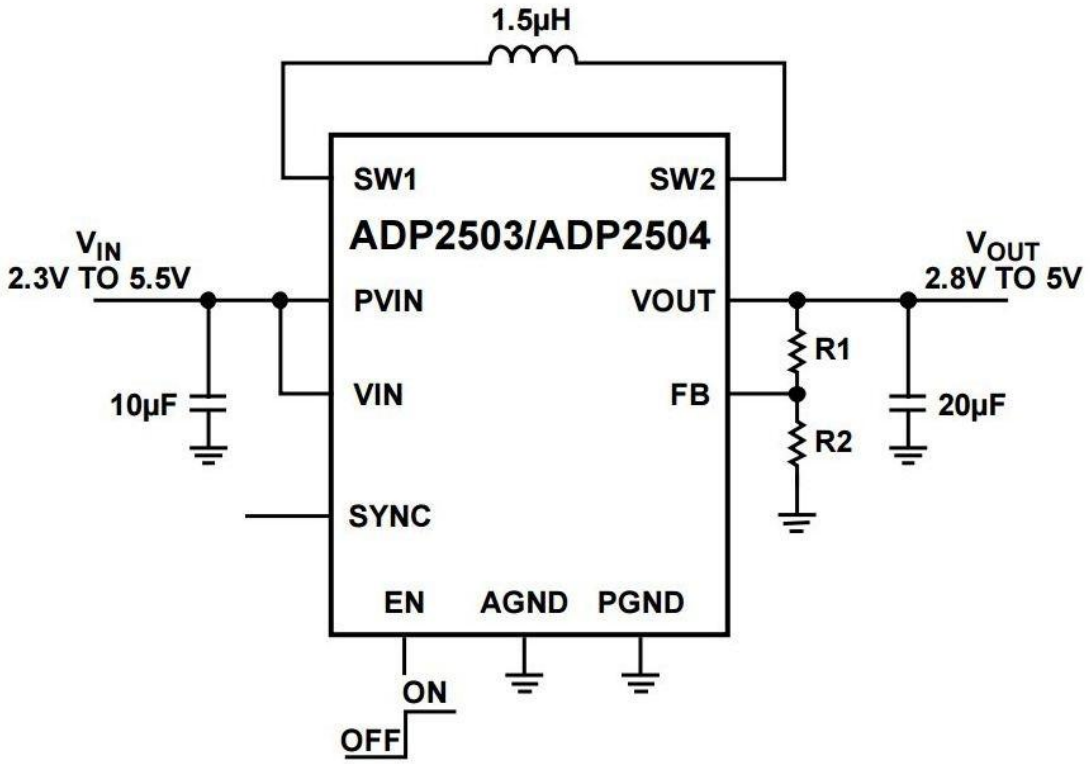
arra, hogy az analóg rész működni tudjon, a műveleti erősítőnk 1.8V egyoldalú tápfeszültségtől már működőképes. Kisebb feszültséget pedig azért nem választottam, hogy a műveleti erősítő saját zaja minél kevésbé rontson a működésen. A specifikációban megemlítettem, hogy hordozható készüléket szeretnénk készíteni, ezért akkumulátorról fog működni az eszközünk. A helytakarékosság miatt a teszterekben használt klasszikus 9V-os elem helyett inkább kisebb AAA típusú akkumulátorokat fogunk használni, ebből négyet sorba kötve 4.8V feszültséget kapunk típustól függően (1.2V-os típusú akkumulátor esetén). Ebből állítjuk elő a 3.3V-ot és a ±2.5V-ot, ehhez két ADP2504 típusú DC/DC átalakítót használunk, az egyikkel 3.3V-ot állítunk elő, a másikkal pedig 5V-ot, aminek a negatív fele lesz a -2.5V és középmegcsapolással létrehozunk egy virtuális földet, ami a 3.3V földjével össze lesz kötve. Értelemszerűen a pozitív fele lesz a +2.5V. Ennek az elrendezésnek a kapcsolása a következő:



11. ábra Tápok, +3.3V ±2.5V

A fentebbi ábrán a konverter bekötése csak blokkszinten néz ki így. A pontos kapcsolás így néz ki: (Adatlapból kivágott kép, viszont az AGND és a PGND nem ugyanaz a

potenciál, tehát az AGND lábat az akkumulátorok negatív felére kell kötni.)



12. ábra DC/DC átalakító

A PV_{IN} lábat a főkapcsolóra kötjük, ahogy a korábbi kapcsolásból kiderül. Az EN és SYNC lábra az analóg földhöz képest kell magas vagy alacsony jelet adni, mindkét lábat a főkapcsolóra kötjük, ezzel engedélyezzük a működést és normál PWM módban használjuk az eszközt. Az R_1 - R_2 osztással tudom beállítani a kimeneti feszültséget, az alábbi egyenlet szerint: $V_{OUT}=(R_1+R_2)/R_2*V_{REF}$, ahol $V_{REF}=0.5V$ és $R_1+R_2\approx400k\Omega$ (adatlap 14. oldal). Ebből adódik, hogy a 3.3V-hoz $R_1=340\Omega$ és $R_2=60\Omega$, az 5V-hoz $R_1=360\Omega$ és $R_2=40\Omega$. Ez a típusú DC/DC átalakító akár 1A-t is képes kiadni magából, de már 3.6V bemeneti és 5V kimeneti feszültség esetén is eléri a 800mA-t. Láttuk, hogy a digitális rész 400mA-nél nem vehet fel sokkal többet, ha még beleszámoljuk a passzív elemek fogyasztását is. Az analóg résznél a műveleti erősítők maximálisan 30mA-t vesznek fel, ami 9*$30mA=270mA$-t jelent a 8 csatornára és a feszültségkövetőre együttesen; a Z-diódán 5mA folyik. A kΩ-os nagyságba eső ellenállásokon is maximum 1-2mA folyhat, így a ±2.5V sem lesz túlterhelve.

4.1.7 Összegzés

A hardver tervezésének végén láthatjuk, hogy melyik részegység milyen feladatot lát el és melyik másik komponenshez kapcsolódik. A mikrokontroller összes lábának bekötése a függelékben található, a korábban definiált szerepű lábakat a hozzájuk kapcsolódó elemhez, IC-hez vezetem; a táplábakat értelemszerűen a 3.3V és a GND vonalra; kap még a PIC egy 80MHz-es kvarc oszcillátort, ami az OSC_1 és OSC_2 lábra kapcsolódik. Az LCD adatvonalaihoz az E portot használom, ami 8 bites és még szabad, a többi vagy nem 8 bites vagy már valamire fel lett használva (SPI, UART). A tápfeszültségek, referenciafeszültség szűrésére 10µF-os kondenzátorokat kötünk az aktív vonal és a föld közé.

4.2 Szoftvertervezés

4.2.1 Áttekintés

A mikrokontrollerünk vezérléséért felelős szoftver rendszerterve következik az alábbi részben. A megtervezendő készülékünkben egyedül a PIC32MX695F512H típusú mikrovezérlő az, ami program futtatására képes; ez fogadja a beállításokat, feldolgozza azokat, ellátja a időzítési feladatokat és vezérli a kimeneteket. Az FTD230X is egy programozható egység, ennek alapbeállítása (adatlap 32-33. oldal) garantálja nekünk a megfelelő működést, így nem kell foglalkoznunk a konfigurálással. Az MCU több feladatot fog ellátni (kijelző írása, jelek kiadása, …) és ennek megfelelően a program is több komponensből fog állni, amik egymással kommunikálnak. A szoftver két részre osztható abból a szempontból, hogy valamelyik külső egységgel való kommunikációt, adatküldést-, fogadást valósít meg vagy valamilyen belső operációt, feldolgozást, időzítést valósít meg. Vegyük először az első csoportba tartozókat:

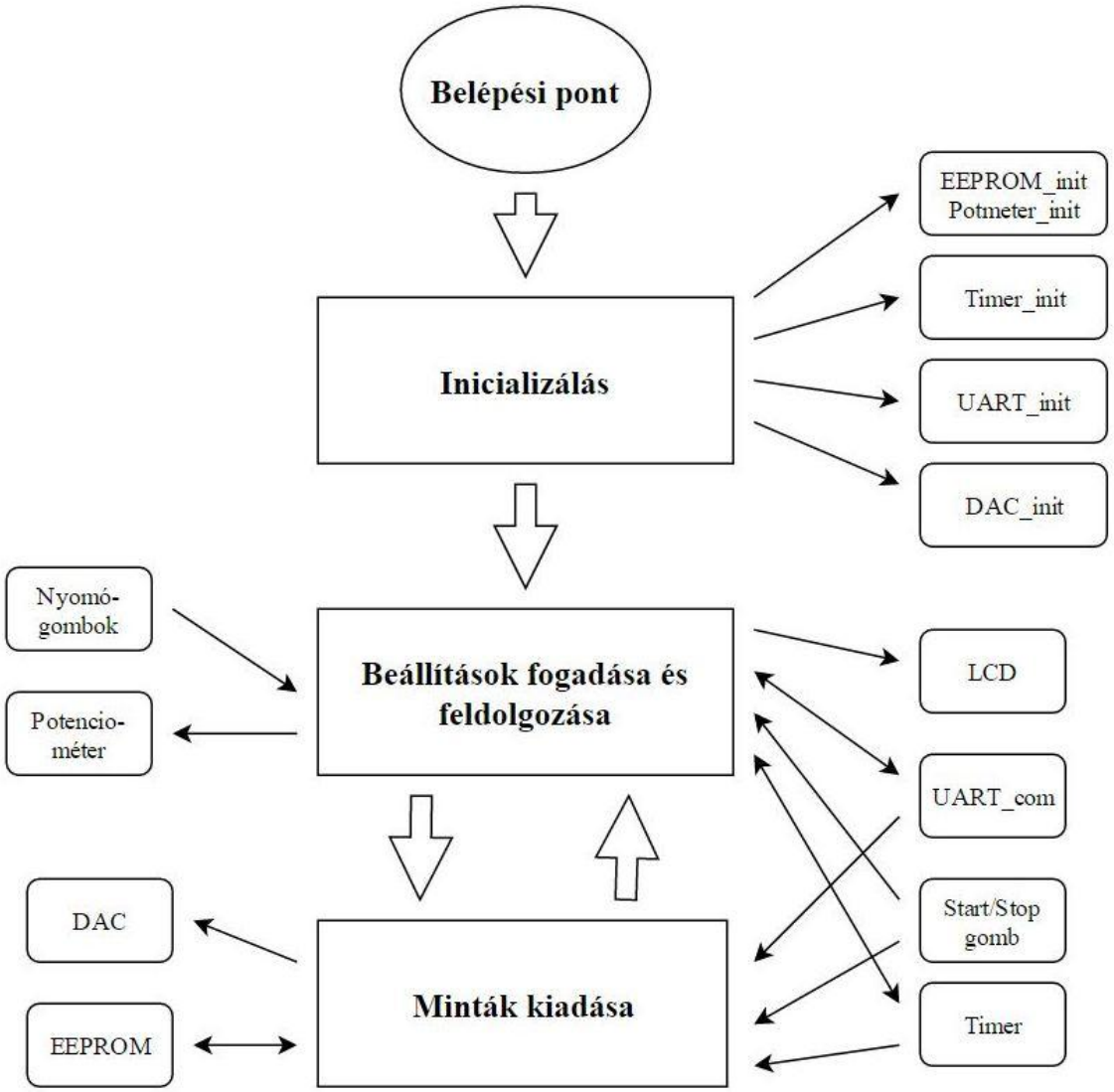
- UART kommunikáció
- EEPROM-ból való adatolvasás
- DAC írása
- Potenciométer állítása
- LCD írása
- Nyomógombok beolvasása

Ezek létének indoklása nem szükséges, mert az eddigi részből következik. A második csoportba tartozó főbb programrészek a következők:

- *Timer időzítés.* Ez lesz az alapja a minták kiadási periódusidejének, valamint a pergésmentesítéshez is timert használunk. Az MCU-ban 5 ilyen egység is található, ebből mi csak egyet fogunk használni, ez fogja ellátni mindkét feladatot.
- *Kapott paraméterek feldolgozása.* A PC-től vagy a nyomógomboktól jött beállítási paraméterek szerint. Ez lesz az egyik legnagyobb része a programunknak, mivel először el kell dönteni, hogy a PC-től vagy a nyomógomboktól jött paramétereket dolgozza fel, ezen beállítások alapján kell választanunk, hogy melyik memóriaterületről fogjuk olvasni az adatainkat (ha az szükséges, mert négyszög jelnél ugye nem kell), milyen sűrűn kell kiadni a mintákat, tehát a timert milyen frekvenciára kell felhúzni. EKG jel esetén jeleznünk kell a programnak, hogy CT potenciált kell számolni (részletesen lentebb).
- *Főprogram.* A kiinduló program, ami a többi programrészt, függvényt meghívja. Először meghívja az inicializáló függvényeket, majd egy végtelen ciklusban várja a beállításokat a perifériák függvényeinek meghívásával és a visszatérési értékük szerint koordinálja a minták kiadását.

A két csoporton felül a korábban említett inicializáló függvények vannak. Ezek azok a függvények, amik a készülék indulásakor először lefutnak és felparaméterezik a megfelelő kommunikációs modulokat, perifériákat; például az SPI kommunikáció sebessége, mintavételezése vagy a timerek alapidőzítésének beállítása zajlik itt.

Az alábbi blokkvázlat mutatja az egységek egymáshoz való kapcsolódását. Itt már látható, hogy a program 3 fő részből áll, az inicializációból, a beállítások fogadásából és a feldolgozásából, minták kiadásából. A nyilak az információáramlás irányát jelzik, a megvalósításuk során a megfelelő függvények végzik a periféria írását vagy az onnan való olvasást.



13. ábra Szoftver blokkvázlat

4.2.2 Fejlesztő környezet és fájlok

A PIC típusú mikrokontrollereket gyártó Microchip cég biztosítja az MPLAB nevű ingyenes fejlesztőkörnyezetet [8], ami fut Windows, Linux és Mac OS operációs rendszerű számítógépeken. Kihasználva ennek az adottságait, a szoftvert C nyelven lehet írni. Két fordítandó fájlból fog állni a projektünk, ezek közül az egyik a *.h* kiterjesztésű header fájl, ami a konfigurációs beállításokat és az oszcillátor frekvenciájának megadását tartalmazza. Itt tudjuk jelezni az MCU felé, hogy 80MHz-es frekvenciájú oszcillátor van az órajelbemenetére kötve, hogy nem használjuk a watchdog, power-up, brown-out reset, code protection funkciókat, hogy a perifériák órajelalapja a rendszer saját órajele, osztás nélkül és hogy HS oszcillátorunk van (tehát a frekvenciája nagyobb, mint 4MHz); a többi beállítás számunkra nem lényeges. A másik egy *.c* kiterjesztésű fájl, amiben a programunk található.

A .c fájl felépítése a következő:

- Először include-oljuk az előbbi .h fájlt, valamint a hivatalosan kiadott plib.h fájlt, ebben olyan függvények, makrók és define-ok vannak, amivel egyszerűen lehet dolgozni. A függvények neve elárulja, hogy mit csinálnak és így átláthatóbb kódot kapunk, például egy port felkonfigurálása, azaz a ki-, bemenetek állítása.

  (pl.: *PORTSetPinsDigitalIn(IOPORT_C, (BIT_1 | BIT_0));*

  C port nullás és egyes bitje legyen digitális bemenet.)

- Utána jönnek a konstansok, ilyen például az EEPROM olvasásakor küldendő 0x03 érték, amit elnevezünk Cmd_read-nek vagy a szinusz jel memóriában való kezdő-, és végcíme Sin_begin és Sin_end névvel. Ezen kívül a portlábakat is itt nevezzük át, például a B port négyes lába lesz a DAC_LDAC, mivel a DA átalakító szinkronizációért felelős lábát ide kötöttük. Ezeket a *#define* paranccsal adjuk meg (pl.: *#define Eeprom_read ((unsigned char)0x03)*). Fontos, hogy megfelelő bitszámúak legyenek a változóink, a példában használt érték 8 bites kell, hogy legyen, mert ezzel tud az EEPROM dolgozni.

- Ezt követi a makrók megadása, ezek kis függvényekként értelmezhetők, segítik a gyorsabb programozást előre definiált funkciójukkal. A /CS jelekhez mind használunk egy-egy makrót, ahol definiáljuk az adott lábhoz a magas és alacsony állapotot is és az ezeknek nevet adunk. Egy példán keresztüli bemutatás:

  #define cs_potmeter_active() (mPORTBClearBits(BIT_5))

  #define cs_potmeter_deactive() (mPORTBSetBits(BIT_5))

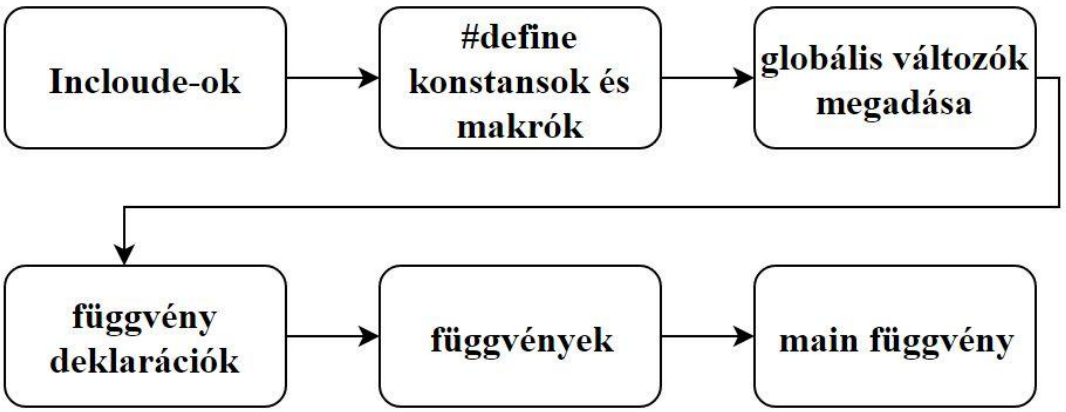
  Ezzel megoldottuk, hogy ha a *cs_potmeter_active();* makrót meghívjuk, akkor kiválasztó jelet kapnak a potenciométerek, amik ugye a B port 5-ös lábára vannak kötve.

- A következő lépés a globális változók megadása, ezeket használjuk fel olyan értékeknek, amiket több függvény is használ, állít. Mindegyik változónk „*unsigned*” típusú lesz, mivel nincs szükségünk előjelre és ezzel a bitműveleteket is könnyen tudjuk kezelni. Ilyen változó például a

timer számlálója, ez az, amit növelünk mindegy egyes timer periódusidőben és összehasonlítjuk a beállításban érkezett értékkel, így kapjuk meg az adott jel mintáinak periódusát.

Az #define-ok és makrók, amire szükségünk lesz a program során a függelékben található.

- A függvények deklarálása a következő lépés, itt csak megadjuk, hogy milyen paraméterrel hívhatóak és a visszatérési értékük milyen típusú (ha van).
- A deklarációk után egyből jönnek a függvények, ahol már az egész funkció megvalósítása meg van írva. Hogy milyen függvényekre van szükség, az lentebb olvasható. Itt található az interrupt függvény is, amit a fordítónak az „*interrupt*” szóval kell kelezni a függvény neve előtt.
- A program utolsó eleme a main függvény, ami a korábban említett inicializálások meghívása után egy végtelen ciklusban vár a beállításokra és megérkezésük után kiadja a megfelelő jelalakot.



14. ábra Szftverstruktúra

4.2.3. Programrészek, függvények

A szoftverünk fentebbi blokkvázlatát jobban részekre bontva egyértelmű lesz, hogy melyik függvénynek mit kell csinálnia, mely másik programrésszel, perifériával kell kommunikálnia.

4.2.3.1 Inicializálás

Nézzük, hogy az indítás utáni inicializáció során mi a dolgunk: Elsőként beállítjuk a lábkiosztásnak megfelelően a használt lábakra, hogy azok digitális portok és bemenetek vagy kimenetek lesznek. Ezt a plib.h-ban található függvényekkel megtehetjük.

A négy külső perifériával (DAC, EEPROM, potenciométerek, UART) való kapcsolathoz fel kell paramétereznünk a kommunikációs protokollt kiszolgáló egységeket, ez a kettő az UART és az SPI.

Az UART-hez a következőket kell beállítanunk: 8 bites adatküldés és adatfogadás; a 'high' szint az aktív, a 'low' az idle állapot; transmit, receive engedélyezés; ha adat érkezett, akkor generáljon megszakítást; 1 stop bit; nincs paritás bit, 9.6kBaud sebesség.

Az adatlap [9] 223-228. oldalain találjuk a vezérlő biteket, amiket be kell állítanunk az előbb specifikált működéshez.

Az SPI kapcsolatnál mindkét esetben be kell állítani, hogy a PIC vezérlő a master; az előosztás az órajelhez, amiből kiadódik a 10MHz mindhárom perifériához. Az EEPROM-hoz és potenciométerekhez 8 bites, digitál-analóg átalakítóhoz 16 bites átvitel a megfelelő;ezért tudjuk megtenni, hogy átparaméterezés nélkül használjuk ugyanazt az SPI egységet a potméterek és az EEPROM esetén. A három adatlapot összevetve az órajel és a bemenet mintavételezése, kimenet írása közötti kapcsolat szempontjából a következő adódik:

	Adat mintavételezése	Adat kiírása
EEPROM	Órajel felfutó élére	Órajel alacsony szintje alatt
Potenciométer	Órajel felfutó élére	-
DAC	Órajel lefutó élére	-

Ezeket a CKE, CKP, SMP bitekkel lehet beállítani mindegyik SPI modulra külön-külön. Az EEPROM-ot és a potmétereket nem fogjuk sosem egyszerre használni, ezért tudjuk őket ugyanazon SPI perifériával kezelni.

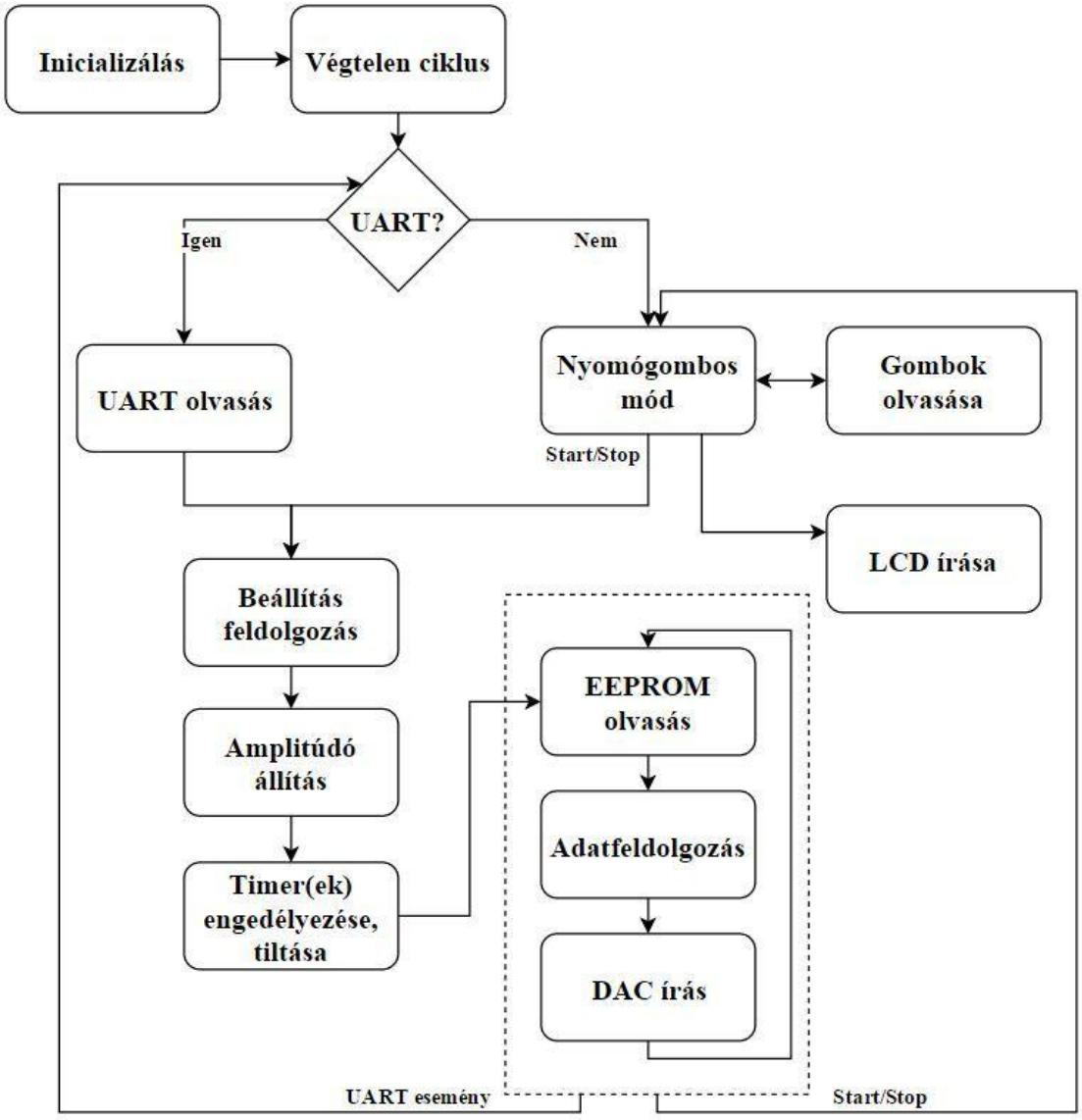
Ezt követi a timerek inicializálása, két időzítőt használunk a pergésmentesítéshez és a minták kiadásának periódusidejéhez. A pergésmentesítés nem

kritikus, 100ms környékén van (célszerű a gomb vásárlása után lemérni, hogy mekkora a prellezés ideje), míg az időkritikus periódusidhöz pontos számítás kell. A 200µs-os idő lesz az, ami a legkisebb olyan periódus, amivel a minták kiadását ütemezni kell, ehhez számoljuk ki, hogy az órajelet mennyivel kell osztani. Ehhez egy 64-es előosztást és egy 249-es összehasonlítást használunk, mert *(1/80MHz)·64·250=200µs*, (0-tól 249-ig számol, az 250 összehasonlítás) ezt az értéket kell majd módosítani, például a 60Hz-es szinusznál, mert ott 333µs-onként kell mintát kiadni (ott 416-ig számolunk) de az összes többihez más érték tartozik, de erre az értékre inicializálunk, aztán ha kell, átállítjuk futás során. Ide a timer2 és a timer3 időzítőt együtt használjuk, így létrehozva egy 32 bites számlálót, azért van erre szükség, mert a 64-es előosztás után nem férne el 16 biten a 4 másodpercig való számolás a négyszög jelhez, de 32 biten már elférünk. *4/((1/80MHz)·64)=5000000*, ami nagyobb, mint 2^{16}=65536, de kisebb, mint $2^{32}\approx 4.3\cdot 10^{9}$. A pergésmentesítéshez 256-os előosztást (timer1) és 31250-ig való számolást alkalmazunk, ez még belefér a 16 bit-be. Mivel mindkettő időzítő megszakítást generálhat, ezért egymás idejét zavarhatnák, de tudjuk, hogy pergésmentesítésre csak beállítás során van szükség, ekkor letiltjuk a periódusidő számlálót és ez igaz visszafelé a minták kiadásának idejére. Ez utóbbi folyamat alatt egyedül a Start/Stop gombot vizsgáljuk, de ehhez nem szükséges pergésmentesítés, mert ha egyszer lenyomást érzékelünk, máris kilépünk a minták kiadásából, elindításkor, pedig várunk 100ms-ot a Start megnyomásának elengedésétől, így nem tud pergés miatt kiugrani a jelkiadási folyamatból.

Az LCD-t utolsóként inicializáljuk, ekkor annyit teszünk, hogy a Function Set utasítással (RS=0, R/W=0, DB7-0=’00111000’), beállítunk 8 bites adatátvitelt, 2 soros kijelzést és 5x8 karakterméretet. (LCD adatlap 11-12. oldal)

Az inicializálás fontos lépése még, hogy a perifériákhoz használt órajel alapja a rendszerórajel legyen osztás nélkül, így a 80MHz-cel tudunk dolgozni, ezt is be kell állítanunk. Az utolsó lépés pedig, hogy engedélyezzük a kommunikációkat és a timer interruptokat.

4.2.3.2 Főprogram áttekintés



15. ábra Főprogram folyamatábra

A felparaméterezés után belépünk egy végtelen ciklusba. Ebben először megvizsgáljuk, hogy van-e UART kapcsolat a PC-vel. Ha van, akkor meghívjuk a függvényt, ami fogadja a beállításokat és a kapott paraméterek alapján elkezdjük a minták kiadását, ekkor nem szükséges az LCD kijelzőt működtetni, hiszen a PC-n látható minden konfiguráció. A minták kiadásának időzítését egy feltételvizsgálattal oldjuk meg, ha a timer számlálója elérte a beállított értéket (mivel többféle frekvencia van, ezért más-más periódusidőkkel kell dolgoznunk, pl.: háromszögnél a 2ms, míg 60Hz-es szinusz esetén 333µs a minták kiadási időköze), akkor megvizsgáljuk, hogy melyik állapotban vagyunk (ezt egy switch – case szerkezettel egyszerűen meg tudjuk

valósítani). Az állapotnak megfelelően fordulunk az EEPROM-hoz, majd küldjük tovább az adatot a DA-nak. Ekkor valósítjuk meg azt is, hogy az EKG jelek esetén az aktív szakaszt leszámítva a kimeneti jel nulla legyen, ehhez az ’10_0000_0000’ bitsort kell küldenünk a DA-nak beállítandó értékként, mivel bifázisos jel esetén a full scale érték fele a nulla feszültségszint. Itt kell elvégeznünk a mellkasi elvezetések viszonyítási potenciáljának, a CT pontnak a kiszámítását is. Erről ugye tudjuk, hogy a három végtagi potenciál átlaga. A jobb kéz potenciálját (RA) tekintjük nulla pontnak, a bal kéz (LA) és a bal láb (LF) feszültségszintjeihez tartozó értéket a memóriából olvassuk, ezután az RA, LA és LF értékeket összeadva és harmadolva megkapjuk a CT pontot. Esetünkben ez csak az LA és LF számok összeadását és 3-mal való osztását jelenti. Mivel a mellkasi potenciálokat a CT-hez viszonyítva tároljuk a memóriában, ezért ebből csak le kell vonni az előbb kiszámolt eredményt és ez a 12 bites sorozat kerül kiadásra. Természetesen ezen feladatok megvalósítását függvények látják el. Minden egyes ciklusban megnézzük, hogy jött-e újabb beállítás, ha igen akkor aszerint adjuk ki a jeleket, esetleg leállítjuk a jelek kiadását (előtte kiküldjük a nulla feszültséghez tartozó kódokat); azt is vizsgáljuk, hogy fennáll-e még az UART kapcsolat, ha nem akkor átállunk az eszköz saját felhasználói interfészének használatára.

Másik lehetőség, hogy nincs csatlakoztatva PC a készülékhez, ekkor kiírjuk az LCD-re a „30 BPM ECG” szöveget és várjuk a kezelő beállításait. Ha a Wave gombot nyomta meg, akkor váltunk a jelalakok között (ECG, Sine, Triangle, Square, Break, majd újra ECG), ECG-nél a 30 BPM, szinusznál a 10 Hz az alapbeállítás, háromszög, négyszög és szünet esetén nincs lehetőség frekvenciát választani. Az Up és Down gombok lenyomására a frekvenciák között lehet lépkedni, ez is körbefordul a végértékek után. Minden gombnyomásra frissítjük az LCD-re írt beállítást. Amp. gomb megnyomása után az Up és Down gombok az amplitúdó állítására szolgálnak. Start/Stop gomb megnyomása esetén elindítjuk beállított minta kiadását az UART módnál leírtak szerint, ugyanazokat a függvényeket felhasználva. Ennél a megvalósításnál is vizsgáljuk, hogy közben jött-e létre UART kapcsolat, ha igen, akkor leállítjuk a minták kiadását és onnan várjuk az adatokat.

4.2.3.3 Függvények

Az előző pontban említett funkciókat függvények segítségével fogjuk megvalósítani az átláthatóság kedvéért. Vegyük sorba ezeket! A függvény funkciójának

megnevezése után *dőlt betűvel a fogadott paraméter(ek),* majd aláhúzva a visszatérési érték áll.

- **EEPROM olvasó függvény.**

  Olvasási cím; mennyi bájtot kell felhozni.

  Tömbben a felhozott bájtok.

  A függvény meghívja először a chipselect jelet kiadó makrót, majd kiküldi az SPI vonalon az olvasás parancsot a memóriának, majd megvárja, míg az kimegy, ezt az SPI buffer ürességének vizsgálatával teszi meg. Ezután kiküldi a felső majd az alsó címbájtot, végül nulla értékű bájtokat küld ki, annyit amennyi bájtot fogadni akar, a kiküldött értéknek nincs is jelentősége, de ennek során az EEPROM visszaküldi az adatbájtokat, ezeket pedig letároljuk a tömbünkben. Minden bájt bufferbe történő beírása után megnézzük, hogy kiment-e az adat az előbb leírt módon. Legvégül elvesszük a chipselect jelet az EEPROMtól.

- **DAC írási függvény.**

  Kiküldendő 12 bites értékek és vezérlőjelek egy tömbben; bit, amivel jelezzük, hogy a mellkasi elvezetésekhez külön-külön érték tartozik-e vagy ugyanazzal dolgozzon.

  A függvényhívást követően kiadjuk a /CS jelet a DA átalakítónak. Majd az SPI bufferbe írjuk az LA elvezetés vezérlő 4 bitjét és a minta 12 bitjét, ezt már ilyen formában kaptuk meg a tömbben, ahol a 2 bájtos elemek tartalmazzák a vezérlő jeleket is. Ez azért megvalósítható egy művelettel, mert a DA-hoz 16 bites átvitelt állítottunk az inicializáció során. Megvizsgáljuk, hogy kiment-e az adat az EEPROM olvasó függvényben használt módszerrel. A vezérlőbitek tartalmazzák, hogy melyik DA csatornára kell a kapott 12 bitet küldeni. Ezt a folyamatot végigjátsszuk az LF és a mellkasi csatornákra is. Ezt követően a szinkronizáló LDAC jellel átküldjük a DA bemeneti regisztereiből a kimenetiekbe az adatot. Ezután elvesszük a /CS jelet a DA-tól

- **Adatfeldolgozó függvény.**

  Az EEPROMból felolvasott bájtokat tartalmazó tömb; a tömb mérete, ebből tudjuk, hogy a mellkasi elvezetések külön jelet kapnak-e.

  A DAC számára értelmezhető 16 bites számok tömbben, ahol a felső 4 bit a vezérlést, az alsó 12 az adatot tartalmazza.

Az EEPROM-ból felhozott adatok még nem alkalmasak arra, hogy egyből a DA-ra küldjük őket, először fel kell darabolni és össze kell fűzni a 8 bites adatokat 12 bites étékekké. Ezután ki kell számolni a CT pontot, majd mindegyik érték elé fűzzük a hozzá tartozó DA csatorna kódját, ezt a MAX3506 adatlapjából ki tudjuk nézni, az első csatornához a ’0010’ érték tartozik, a ketteshez a ’0011’ és így tovább. Ezeket az értékeket adjuk vissza egy tömbben.

- **Amplitúdóállító függvény**

  Amplitúdó érték

  Nincs

  A függvényt akkor hívjuk meg, amikor a felhasználó a beállított értékek után megnyomta a Start/Stop gombot vagy ha az USB-n keresztül megérkezett a beállítás. Az SPI buszon keresztül kiküldjük a /CS jelet a potenciométernek, majd a kapott érték alapján a kiválasztó jelet és a 8 bites kódot (’13’ - 50Ω, ’26’ - 100Ω, ’51’ - 200Ω), aszerint, hogy mit állított be a felhasználó. A kiválasztó jel dönt a két potenciométer csatornája között. ’00’ kell az egyikhez, ’01’ pedig a másikhoz. Ez eddig 2+8=10 bit, a maradék 6 bitet nullákkal töltjük fel. Ezután elvesszük a /CS jelet, majd megint kiadjuk, ezzel az elsőnek beállított potméter értéke aktualizálódik. Ugyanezt végigjátsszuk a másikra is, majd elvesszük a /CS jelet.

  Ezt a függvényt csak akkor hívjuk meg, ha új beállítás érkezett, mert akkor lehet csak változtatni az amplitúdón.

- **LCD író függvény**

  Milyen jelalak; milyen frekvencia; milyen amplitúdó; aktív vagy sem.

  Nincs.

  Az LCD-n négy dolgot szeretnénk megjeleníteni, az első, hogy milyen hullámforma az aktuális (ecg, szinusz …), ehhez milyen frekvencia tartozik, már ha van hozzá, mert a szünetjelnél nincs, hogy mekkora a kimeneti jel amplitúdója és hogy éppen aktív állapotban vagyunk, tehát a kimeneten már jelen vannak a minták vagy beállítási (passzív) fázisban vagyunk. Négy globális változóban tároljuk ezeknek az aktuális értékét, így amikor meghívjuk a függvényt nem biztos, hogy az egész LCD-t frissítenie kell, lehet, hogy csak ki kell írnia egy $ jelet, amivel jelezzük, hogy aktív állapotban vagyunk vagy csak

az ECG frekvenciájának kijelzésén kell frissíteni. A felső sort használjuk a hullámforma és az amplitúdó kijelzésére, az alsót pedig a frekvencia és az állapot kijelzésre. A függvény a bemeneti paraméterek alapján megnézi, hogy mi az, amit változtatni kell az aktuális kiíráson, fontos, hogy ha a hullámforma változik, akkor a frekvencia is változik az alapértékre, ez EKG-nél 30 BPM, szinusznál 10 Hz, háromszög esetén 2Hz, négyszögnél 0.125Hz, szünetnél semmi. Csak az első kettő esetben lehet a frekvenciát állítani, de ezt a függvényt már csak előfeldolgozott, értelmes adatokkal hívjuk meg, ezért csak beállítható értéket írhat ki. A bemeneti lehetőségek szerint a következőt kell tennie:

Megvizsgálja, hogy a bejött hullámforma egyezik-e a jelenleg kiírttal, ha nem akkor kiírja a felső sorba az újonnan érkezettet (beírja és a többi karaktert törli a sorból, kivéve az utolsót), és továbbmegy, ha igen, akkor is továbbmegy. Megnézi, hogy milyen frekvencia van éppen kiírva, ha aktuális, akkor továbbmegy, ha nem akkor frissíti, azaz beírja az újat és a sor utolsó karakterét kivéve törli a sort és továbbmegy. Vizsgálja az aktív jelet, ha passzív, akkor az utolsó karaktert törli, ha aktív, akkor $ jelet ír oda. A függvény kezeli, hogy melyik sorba kell írnia.

A globális változók létrehozásakor azt 'szünet' értékűre inicializáljuk, ez nem fog megjelenni, mert az első függvényhívásnál már az ECG 30 BPM értékkel hívjuk meg, hiszen ez az alapbeállítás, innen tud a felhasználó váltani. A változók vizsgálatát egy switch – case szerkezettel oldjuk meg.

- **UART olvasó függvény**

  Olvas vagy vizsgál.

  Három változó, amiben a hullámforma, a frekvencia és az amplitúdó van.

  Az UART olvasó függvényt, akkor hívjuk meg, amikor észleljük, hogy valami történt a vonalon, ezt nekünk egy megszakítás jelzi, másik lehetőség, hogy vizsgálni akarjuk, hogy él-e még a kapcsolat a PC-vel, ezért küldünk a számítógép felé egy '?' karaktert, mire ő 5 másodpercen belül egy 'O' (mint OK) karakterrel kell, hogy válaszoljon. Az *'olvas vagy vizsgál'* bemeneti paraméter határozza meg, hogy milyen műveletet hajtunk végre. Megvizsgáljuk, hogy milyen adatok érkeztek/érkeznek és ennek megfelelően várunk további paraméterre, például frekvenciára vagy amplitúdóra. Ha megérkezett minden, akkor ezeket visszaadjuk a fentebb említett három változóban.

A PC-től a következő struktúrájú beállítást várjuk:

* Egy karakter jelzi, hogy milyen hullámformát szeretnénk. E= ecg, S=szinusz, T=háromszög (triangle), N=négyszög, B=szünet (break). A karakterek lehetnek kicsik is.

* 'E' és 'S' esetén frekvenciaértékre várunk, ez két vagy 3 karaktert jelent; 'enter' jelzi, hogy megérkezett, tehát '10' után ne várjunk még egy nullát, mert ez 10 Hz-et jelent és nem 100-at állítunk be. 'T', 'N' és 'B' esetén csak egy entert várunk. Szünet esetén a következő pontot át is ugorjuk.

* 'E', 'S', 'T' és 'N' esetén várunk egy harmadik karakter, ami az 5, 1 vagy 2 lehet, ezek 0.5mV-ot, 1mV-ot és 2mV-ot jelentenek, ezután entert várunk. Ha enter érkezik az 5, 1, 2 helyett, akkor tudjuk, hogy az 1mV-os értékkel kell dolgoznunk. Ezt a harmadik változó értékében jelezzük.

Ha értelmezhető karaktersorozat jött be, akkor a PC felé visszaküldünk egy '$' jelet, nem érvényes sorozat esetén egy '!' karakterrel jelezzük a hibát és újra elindítjuk a várakozást. A felkiáltójelet bármikor visszaküldhetjük a vizsgálat során és akkor újrakezdjük a várakozást, $ jelet csak a teljesen helyes érték esetén küldünk.

- **Beállításfeldolgozó függvény**

Kettő változóban kapjuk meg a hullámformát és a frekvenciát.

5 érték visszaadása tömbben: 2 cím, 2 timer szorzás és egy mintaszám.

A főprogram ezt a függvényt az UART-tól kapott adatokkal vagy a nyomógombokkal beállított értékekkel hívja meg. A függvény dolga eldönteni, hogy melyik jelalakhoz milyen érték tartozik, amit a timer, memóriaolvasás, … tud értelmezni. Két egymásba ágyazott switch – case szerkezettel kiválasztjuk, hogy melyik jelalakról van szó és aszerint beállítjuk a következőket:

* A jelalak memóriában való tárolásának kezdő és végcíme.

* A timer felszorzásához használt érték pontos beállítása, egyik a korábban említett 200µ-hoz tartozó 250 vagy a 333µs-hoz tartozó 416-os érték, ez minden jelalakra és frekvenciára más és más.

* EKG jel esetén szükséges megadni, hogy mennyi az a minta, amit a memóriából olvasunk és mennyi az az idő, ami a passzív szakaszhoz kell, ez egy számot jelent, amíg elszámolunk a timerrel és közben nulla feszültségszinten állítunk a kimenetre. Előbbire azért van szükség, hogy ha később más mintát

ültetünk le a memóriába, amihez más bájtmennyiség tartozik, attól még ez a függvény használható legyen.

4.2.3.4 Főprogram struktúra

Most már jobban látható, hogy milyen folyamaton megy végig a program és ehhez milyen függvényeket így végigmenni a modellen.

- A végtelen ciklusunk első lépése, hogy megnézzük van-e UART kapcsolat, ha van, akkor a nyomógombokkal nem is foglalkozunk, LCD-t nem kezelünk, ha nincs, akkor pedig ez utóbbiakkal tartjuk a felhasználóval a kapcsolatot. UART esetén várjuk, hogy jöjjön a beállítás (**UART olvasó függvény**) ha megérkezett, akkor a megfelelő paraméterekkel visszatérünk, a főprogram meghívja a **Beállításfeldolgozó függvényt** a kapott értékekkel, majd meghívja az **Amplitúdóállító függvényt.** Engedélyezzük a periódusidő timerünket (timer2 és timer3). Ezután belép egy ciklusba, ami a timer időzítésének megfelelően fut le időközönként és ezen belül vizsgáljuk még 30 másodpercenként, hogy van-e még UART kapcsolat, ha van, akkor megyünk tovább. (UART beállítás esetén itt nem vizsgáljuk a Stop gombot, egyébként igen.) Itt hívjuk meg az **EEPROM olvasó függvényt**, majd a kapott adatokat átadjuk az **Adatfeldolgozó függvénynek**, az itt kiszámolt és feldolgozott bájtokkal pedig meghívjuk a **DAC írási függvényt**, ezt a hármat ismételjük az időzítésnek megfelelően. A timer interrupt növel egy másik számlálót is, amit arra használunk, hogy elszámoljunk vele ~30 másodpercig, ekkor meghívjuk az **UART olvasó függvényt** ('vizsgál' módban) ami megvizsgálja, hogy van-e még kapcsolat, ez azért fontos, hogy ha esetleg megszakadt a PC-s csatorna, akkor át tudjunk váltani nyomógombos kezelésre.
- Ha az UART csatornán érkezett adat vagy megszűnt a kapcsolat, akkor nem adunk ki mintát, egy változóban jelezzük, hogy lépjünk ki az időzítő ciklusból, ezért az a következő vizsgálat során már nem hajtódik végre. Tiltjuk a periódusidő timert. Visszaugrunk a végtelen ciklus elejére, így újra megvizsgáljuk az UART kapcsolatot, ha van, akkor adatok érkezésére várunk, ezeket feldolgozzuk, ha nincs kapcsolat, akkor másik módba lépünk, a nyomógombokról fogjuk a konfigurációt olvasni és használjuk az LCD kijelzőt.
- Nyomógombos kezelés esetén beállítjuk alapértéknek a 30BPM ECG-t és ezt ki is írjuk a kijelzőre, engedélyezzük a timer1-et (pergésmentesítéshez), majd

várunk a felhasználóra. Vizsgáljuk mind az 5 gombot, ezt úgy tesszük, hogy beolvassuk az értékét, ha nem tapasztalunk lenyomást, akkor újra vizsgáljuk, ha tapasztalunk, akkor várunk 100ms-ot és ha utána is lenyomott értéket észlelünk, akkor aszerint cselekszünk. Wave gomb esetén hullámformát, Up és Down gomb esetén pedig frekvenciát váltunk (ha van értelme). Amp. gomb esetén jelezzük egy változóban, hogy az Up és Down gomb állítása az amplitúdó állítást jelzi. Ha újra Wave gombot nyomunk, akkor visszatérünk a hullámforma állításhoz, de még nem léptetünk csak a következő megnyomásakor. Az 5 gomb közül a Start/Stop gomb van priorizálva, tehát addig csináljuk, csak a beállítást, amíg ezt meg nem nyomták és csak akkor megyünk tovább, ha elengedte a felhasználó. Ezután várunk 100ms-ot (itt kevesebb is elég lenne, mert a gomb beállása a kritikusabb, nem a váltás kezdete, de a biztonság és egyszerűség miatt kell a 100ms) és meghívjuk a **Beállításfeldolgozó függvényt**, majd az **Amplitúdóállító függvényt**, tiltjuk a pergésmentesítés timert, engedélyezzük a periódusidő timert és belépünk a fentebb említett ciklusba. Annyi a különbség ez és az UART kapcsolat között, hogy itt vizsgáljuk, minden periódusban, hogy le lett-e nyomva a Stop gomb, ha igen, akkor kiugrunk a ciklusból. A nyomógombkezelés résznél minden lenyomás után állítjuk a státuszt, tehát, hogy melyik jelalak az aktuális. Eszerint meghívjuk az **LCD író függvényt**. A három gomb közül a Wave-nek van elsődleges és az Up-nak másodlagos prioritása, a Down-nak harmadlagos, az Amp-nak pedig negyedleges, tehát ha több gombnyomást érzékelünk egyszerre, akkor eszerint döntjük el, hogy melyik jut érvényre.

5 Kísérleti modell („deszkamodell”)

5.1 Áttekintés

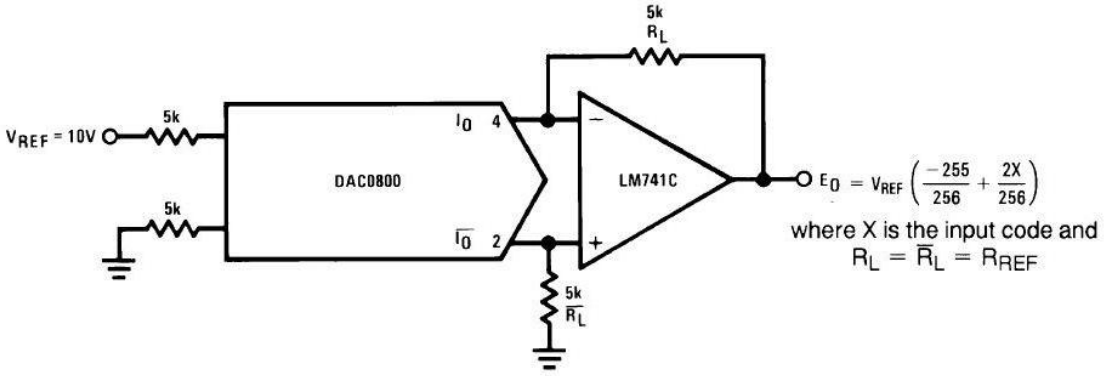
A hardver és szoftverterv végeztével meg is lehetne csinálni a készüléket és megírni hozzá a szoftvert, viszont célszerűbb előtte egy deszkamodellen megnézni az alapvető működést, hogy lássuk, ha esetleg valamin változtatni kell. A szakdolgozat írása során is előbb készítettem el ezt az egyszerűsített változatot és közben gyűjtöttem az információkat a végleges készülékhez, így a tapasztalatokat be tudtam építeni a tervezés során.

A cég, ahol a szakdolgozatot készítettem biztosított számomra egy „PIC16F877 KÍSÉRLETI PANEL”-t [10], amin egy PIC16F874A típusú mikrokontroller volt található. A panelen RS232 csatlakozó található, ezért egy USB/RS232 átalakító kábellel valósítottam meg a PC-s kapcsolatot. A panelen háromféle memóriának van helye, I2C, Microwire és SPI kommunikáció valósítható meg velük. Mivel az SPI gyorsabb, mint az I2C és nem kritikus a vezetékek száma, valamint az SPI ismertebb, mint a Microwire, ezért választottam ezt a protokollt. Ehhez egy 25LC256 típusú EEPROM-ot használok, amit a végső termékbe is terveztem.

Figyelembe véve, hogy dugaszoló próbapanelen fogom a kapcsolást összerakni és ez elérhető számomra, ezért egy DAC0800 típusú párhuzamos, 8 bites, áramkimenetű DA átalakítót fogok használni. A két csatornán való jelkiadást úgy oldom meg, hogy egy jelalak lesz, amit tárolok és ez kerül a kimenetre 1mV nagyságúra osztva, ezután ennek egy továbbosztottja lesz a másik kimenet.

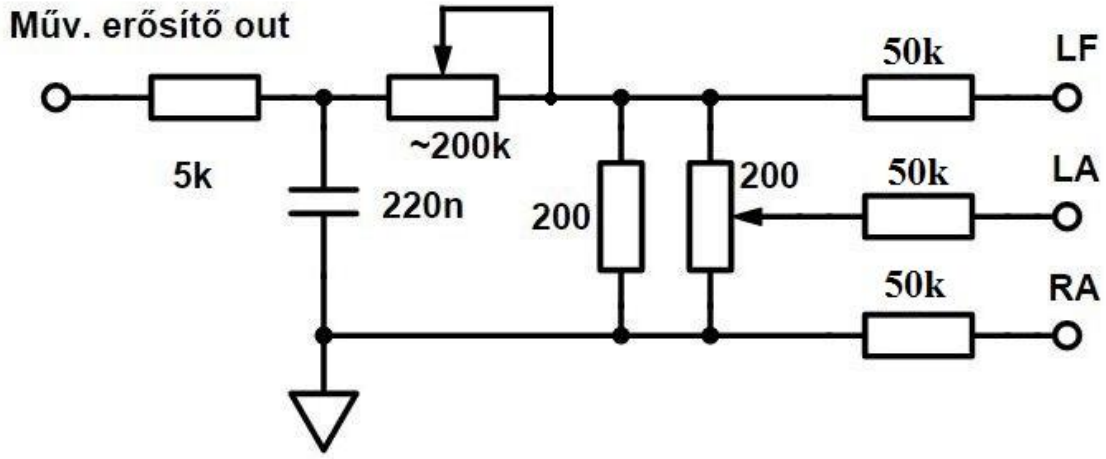
5.2 Hardver tervezése és megvalósítása.

Kihasználva, hogy a fejlesztőpanel 8 bites D portjának minden lábára van egy led kötve, ezt fogom használni a DA átalakító bemenetéhez, így a tesztelés során látszik, hogy mit adunk a kimenetre, ezzel megspóroljuk a debuggolást. A DA lábainak bekötése megtalálható az adatlapban, ha abból egy bifázisos, feszültség kimenetű kapcsolást szeretnénk kapni. A többi láb bekötése az adatlap első oldalán található.



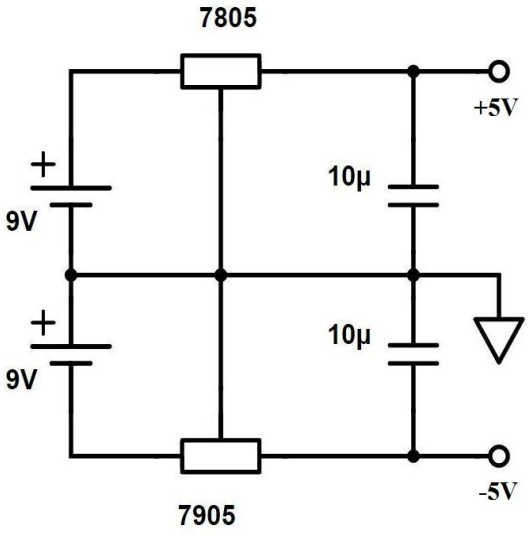
16. ábra Deszkamodell DA kapcsolás

A mi estünkben a referenciafeszültség 2.04-2.05V lesz és egy TL071CN típusú műveleti erősítővel oldjuk meg, hogy bifázisos és feszültség kimenetet kapjunk. Ezután egy 150Hz-es aluláteresztő szűrő és egy leosztás következik a fenti tervhez hasonlóan, azzal az eltéréssel, hogy a 100Ω-os feszültségosztó ellenállás helyett egy 200Ω-os ellenállást és egy 200Ω-os potenciométert kötöttem párhuzamosan, így az eredő ellenállás 100Ω lett. A felső pontjukról vezettem ki az LF kimenetet és a 200Ω-os potmétert pedig úgy állítottam, hogy ott a középső megcsapoláson ~0.3mV legyen, ez lett az LA. A 200Ω-os potmétert, azért választottam (lehetett volna egy darab 100Ω-os a két darab 200Ω-os helyett), mert így finomabban lehet hangolni a kimeneti amplitúdót. A specifikáció szerinti 2 csatorna a mi esetünkben 2 elvezetést jelent, azaz 3 potenciált hozunk létre és ezek között mérjük a feszültséget, ebből az egyik a földpont lesz. A feszültségosztó nagyobbik ellenállása egy 200kΩ-os potenciométer, amit nem pontosan gyártottak, ezért kicsit nagyobb lett a teljes ellenállása, így be tudtam állítani rajta az 1mV-hoz szükséges értéket. (A másik két ellenállás sem volt pontos így 203kΩ-ot kellett rajta állítani) Ezután mindegyik kimenethez sorba kötöttem egy-egy 50kΩ-os ellenállást, a páciens impedancia miatt. Ezt az RA kimenetre is megtettem, azaz a GND pontot kivezettem 50kΩ-on keresztül. A kapcsolás a műveleti erősítő kimenetét követően a következőképpen néz ki:

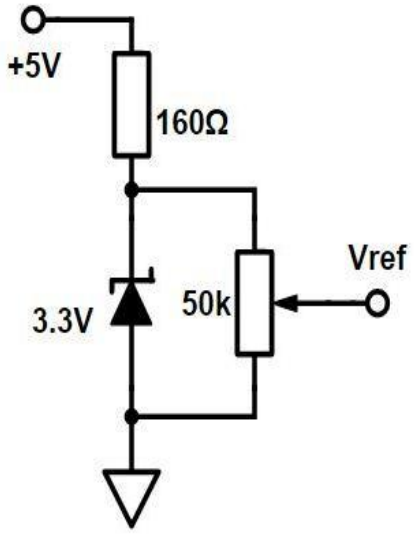


17. ábra Deszkamodell kimenet

A tápfeszültséget itt csak a DA-nak és a műveleti erősítőnek kell adni, mert a fejlesztőpanelnek van saját hálózati adapteres ellátása. A teszterekben klasszikusan használt 9V-os elemre hagyatkoztam én is, ebből kettőt használtam a tápellátáshoz a következőképpen:



19. ábra Deszkamodell tápellátás



18. ábra Deszkamodell referencia

A 7805 és 7905 stabilizátorok +5V és -5V előállítására alkalmasak. A 10µF-os kondenzátorokkal szűrjük ezt a feszültséget. Ez látja el a DA-t és a műveleti erősítőt. A 2.04V - 2.05V referenciát itt is Z-diódával állítjuk elő, itt egy 3.3V-osat használunk, amivel sorba kötünk 160Ω-ot mivel 10-11mA áramnál stabil a dióda feszültsége. Vele párhuzamosan kötve 50kΩ-os ellenállást, azon be tudjuk állítani a megfelelő feszültségszintet úgy, hogy nem húzzuk el a dióda áramát a stabil tartományon kívülre.

A tápfeszültséget azért választottam ±5V-ra, mert a DA elég zajos volt kis tápfeszültség (2-3V) esetén; ezzel együtt kitűnt, hogy a 9V-os elem nagy helyet foglal és az árához képest jóval kisebb a kapacitása, mint például egy AAA típusú elemnek vagy akkumulátornak; ezért a végső tervbe már azt terveztem be.

5.3 Szoftver

5.3.1 Bevezetés

A specifikációban kiírt módon a deszkamodellben nem szükséges az LCD és a nyomógombok kezelése, csak a PC-s kapcsolatot és eszerint a jelek kiadását kell megvalósítani. Ehhez a fent megnevezett fejlesztőkörnyezetet [8] használtam és szintén két fő fájl volt a projektben. Az egyik az *'ecgheader.h'* nevű, amiben a konfigurációs beállítások vannak (Watchdog, Power-up, Brown-out reset, Code protection, Flash memória írás engedélyezése kikapcsolva, HS oszcillátor és 20MHz frekvencia), valamint itt include-oljuk a pic.h és xc.h fájlokat, ezek tartalmazzák a kontrollerspecifikus kódokat.

5.3.2 Felépítés

A szoftverünk struktúrája hasonlóan néz ki a fentebb megtervezetthez, csak sokkal egyszerűbb.

A program indulásakor inicializáljuk a D portot kimenetnek és az A valamint C port megfelelő lábait, ki- és bemeneteknek, az SPI és USART miatt. Az SPI modult is felkonfiguráljuk az EEPROM-hoz. A PIC lesz a master, 5MHz-es órajellel fog a kommunikáció zajlani. Bejövő adat mintavétel a kimenő adatidő végén, átvitel az aktív-idle átmenetnél, idle állapot az órajel alacsony szintjén, és engedélyezzük a kommunikációt. A panelen van egy MAX232 IC, ami elvégzi a szintillesztést az RS232 és az USART modul közötti kommunikációhoz. Az USART átvitel 9.6 kBaudos lesz, 8 bites aszinkron kommunikációhoz állítjuk be a megfelelő biteket és engedélyezzük a kommunikációt. Majd kiolvassuk a 3 szintű USART stacket háromszor, hogy biztos üres legyen. A timer0-t beállítjuk, hogy a rendszerórajelről működjön (ezt alapból néggyel osztja a timer modul), felfutóélre számoljon és túlcsorduláskor interruptot generáljon, előosztást alapból nem állítunk hozzá, egyedül a négyszöghöz kell majd használnunk, mert ott 4 másodpercenként van csak új minta kiadása. Végül engedélyezzük a megszakítást. A megszakítási rutin mindössze annyit csinál, hogy növel egy változót, amit mi fel fogunk használni összehasonlításra.

3 függvényünk van a főprogramon kívül, ezek az alábbiak:

- `unsigned char SPI_com(unsigned char temp_dataout);`

  A függvény a bemeneti paramétert kiküldi az SPI buszon és a beérkezett bájtot pedig visszaadja.

- `unsigned char EEPROM_Read(unsigned int temp_addr);`

  Az EEPROM olvasásához a kapott címet felbontja alsó és felső bájtra, ezután meghívja a fentebbi függvényt, az olvasás paranccsal (0x03), majd a felső és alsó bájttal, aztán egy 0 értékű bájttal és ezalatt visszakapja a megcímzett helyen lévő adatot. Ezt adja vissza.

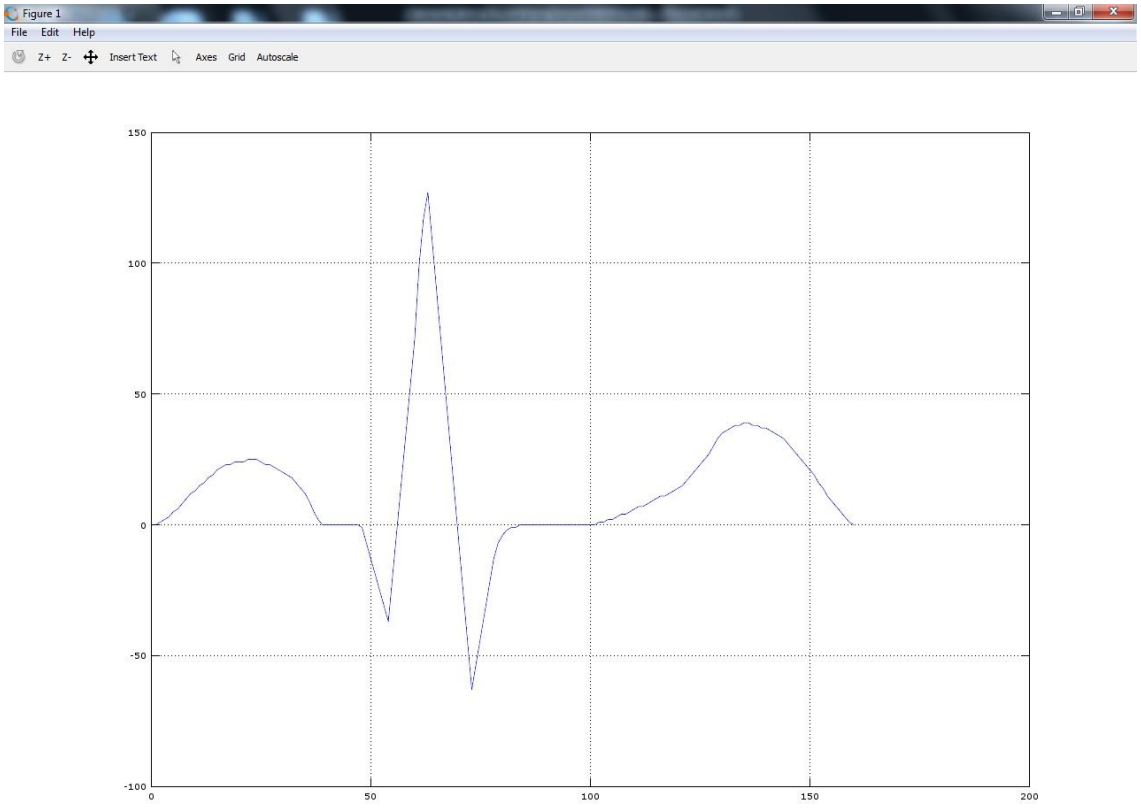
- `void USART_com(void);`

  A függvény megnézi, hogy jött-e adat az USART vonalon, ha nem akkor kiugrik és mást nem is csinál. Ha érkezett adat, akkor megvizsgálja, hogy azon EKG vagy szinusz beállítás jött-e, ha igen, akkor vár a frekvenciára is és miután megkapta. Visszaküld egy ’$’, egy ’CR’ és egy ’LF’ karaktert. Négyszög, háromszög vagy szünetjel esetén is ezeket a karaktereket küldi vissza; ha nem érvényes kód jött, akkor a ’$’ helyett ’!’ jelet küld vissza. Ebben a függvényben valósítjuk meg a fogadott adatok alapján a paraméterek állítását is, a memóriacímeket, a timer számlálót, EKG esetén a passzív szakasz idejét, négyszögnél a timer előosztását paraméterezzük fel.

A főprogramban az inicializáló függvények meghívása után egy végtelen ciklusban várjuk a beállításokat, amíg ezek nem érkeznek meg, addig nulla feszültségszintre állítjuk a kimenetet. Ha fogadtunk adatokat, akkor a felhúzott timer alapján vizsgáljuk a periódusidőnket, ha az összehasonlítás egyezik, akkor felolvassuk a memóriából az adatot vagy elvégezzük a megfelelő műveletet. Ez a művelet lehet a kimenet negálása 4 másodpercenként a négyszögjelnél, a kimenet 0V feszültségre állítása (ez ugye a 0x80-as kód, mert a 0xFF-fel állítjuk 1mV-ra és 0x00-val -1mV-ra az amplitúdót) szünet esetén vagy az EKG jel passzív szakaszában.

A teljes program megtalálható mellékletként a szakdolgozat mellett. Az időzítések számítása pedig a függelékben.

Az EKG jelalakot a PhysioNet honlapjáról töltöttem le [11] az Octave [12] program számára feldolgozható struktúrában, egy átlagos jelet töltöttem le és átírtam egy pár értéket, hogy „kerekebb” legyen a P és T hullám, valamint csúcsosabb az R hullám:



20. ábra ECG_Octave

A szinusz és háromszög jelalakok generálásához a Daycounter honlapot [13] használtam. Az Elnec cég által gyártott Memprog2 típusú programozóval írtam be az EEPROM-ba az adatokat.

A PC oldali használathoz a Putty nevű terminál emulátort használtam.

5.4 Mérések

A hardver megépítését (mellékletben csatolva a képek) és a kontroller felprogramozását követően próbaméréseket végeztem. A rendelkezésemre álló oszcilloszkóp nem tudott pár mV nagyságrendű jeleket mérni, ezért egy EKG készüléket használtam a tesztelésre. 50mm/sec és 10mm/mV-os beállítással mértünk, ez azt jelenti, hogy a függelékben található képeken 2 négyzetoldal jelent 1 mV-ot és 10 négyzetoldal 1 másodpercet, tehát egy négyzetoldal 100ms-ot tesz ki. A tesztméréseknél

használt készülék van egy 0.5Hz-es felülátereszt szűrő és egy 150Hz-es aluláteresztő; az előbbi az oka a négyszögjel csillapodásának. Beállítható rajta 50Hz-es lyukszűrő is, az egyik képen látszik, hogy az 50Hz-es szinuszt ez kiszűrte. A 180 BPM-es és 240 BPM-es jelnél látszik, hogy az amplitúdó 1mV-ról 0.8-0.9mV-ra csökken, ennek oka, hogy ennél a jelnél az R hullám annyira „hegyes", hogy a 150Hz-es szűrő ezt kiszűri; emiatt fontos a végső készülékben több mint 160 mintából álló jelből legyen az EKG tárolva a memóriában, így hosszabb időt kapunk. A Q, R és S hullám csúcsai jobban szét lesznek húzva, ettől még „hegyesek" maradnak.

6 Összefoglalás, kitekintés

A szakdolgozatom során megtervezett EKG teszter készülék azokat a funkciókat látja el, amelyek a legtöbb EKG eszköz időszakos felülvizsgálatához szükségesek. Ezen funkciókon túlmutatóan több továbbfejlesztési lehetőség is szem előtt van, ezek közül emelnék ki néhány fontosabbat. Szélesebb körű használathoz a készüléket fel lehet készíteni kóros EKG jelek kiadására, ezzel az őrzőmonitorok és EKG-k szakértő programjait (interpretereket) lehet tesztelni, hiszen ezek a normálistól eltérő minták felismerésére vannak tervezve. Ehhez kapcsolódóan egy PC-s szoftver megvalósítása is célszerű lehet, amivel különféle jelsorozatok küldhetőek a teszterbe, ami ezeket a memóriájában tárolva később ki tudja adni azokat. Áramköri szempontból érdekesebb lehetőség a vizsgáló jelekhez definiált frekvenciamenetű zaj hozzákeverése, ezzel pedig a vizsgálandó készülékek zajszűrését tudjuk tesztelni. A tápellátás tartósságának elősegítésére ki lehet használni a mikrokontroller sleep üzemmódján és ezzel a fogyasztását csökkenteni.

Ezennel szeretnék köszönetet mondani Zakár Istvánnak és Benesóczky Zoltánnak, akik konzulenseimként tanácsaikkal, javaslataikkal, valamint a folyamatos munka megkövetelésével nagyban segítették a munkámat; valamint a Medihead Kft.-nek, ahol rendelkezésemre bocsátották a tervezéshez szükséges eszközöket.

Irodalomjegyzék

[1] Dió Mihály – Szekrényesi Csaba – Zakár István – Zombory Péter: Biofizika és orvostechnika alapjai, 2013, pp. 75-97

[2] Einthoven elvezetések: https://en.wikipedia.org/wiki/Electrocardiography

[3] EKG jelalak: http://m.cdn.blog.hu/ba/babosi/image/ecg1.gif

[4] Mellkasi elvezetések: http://www.nottingham.ac.uk/nursing/practice/resources/cardiology/function/chest_leads.php

[5] TechPatient CARDIO(350$): http://www.heinstruments.com/ecg-simulator.htm

[6] Fluke PS410 ECG/Arrhythmia Simulator: http://biomedequip.com/index.php?route=product/product&product_id=88

[7] Netech MiniSim 1000 Basic ECG Arrythmia Simulator: http://biomedequip.com/index.php?route=product/product&product_id=94

[8] MPLAB® X, Microchip: http://www.microchip.com/pagehandler/en-us/family/mplabx/

[9] PIC32MX695F512H leírás: http://www.microchip.com/wwwproducts/Devices.aspx?product=PIC32MX695F512H

http://ww1.microchip.com/downloads/en/DeviceDoc/61156H.pdf

[10] PIC16F877 KÍSÉRLETI PANEL: http://www.chipcad.hu/download/fd1.pdf

[11] PhysioNet EKG adatbázis: https://physionet.org/cgi-bin/atm/ATM

[12] Octave program honlapja: https://www.gnu.org/software/octave/

[13] Daycounter sine: http://www.daycounter.com/Calculators/Sine-Generator-Calculator.phtml

Triangle: http://www.daycounter.com/Calculators/Triangle-Wave-Generator-Calculator.phtml

[14] Putty emulátor: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Egyéb források:

[15] Microcontroller selector: http://www.microchip.com/maps/microcontroller.aspx

[16] Blokkdiagramok, folyamatábrák: https://www.draw.io/

[17] FT230X
http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT230X.pdf

[18] Ábrák karjozása: http://www.digikey.com/schemeit/

[19] Zéner-dióda: http://www.nxp.com/documents/data_sheet/NZX_SER.pdf

[20] MCP6054 műveleti erősítő: http://hu.rs-online.com/web/p/muveleti-erosito/6878777/

http://docs-europe.electrocomponents.com/webdocs/0d96/0900766b80d96b07.pdf

[21] Digitális potenciométer: http://hu.rs-online.com/web/p/digitalis-potenciometerek-es-kiegeszito/7097121/

http://docs-europe.electrocomponents.com/webdocs/0de0/0900766b80de04c5.pdf

[22] DC/DC átalakító: http://hu.rs-online.com/web/p/dc-dc-atalakito/7427774/

http://docs-europe.electrocomponents.com/webdocs/0fdf/0900766b80fdfe52.pdf

[23] LCD kijelző: https://www.sparkfun.com/products/9052
https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf

[24] PIC16F874A:
http://www.microchip.com/wwwproducts/Devices.aspx?product=PIC16F874A
http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf

[25] TL071CN műveleti erősítő:
http://www.st.com/web/en/resource/technical/document/datasheet/CD00000488.pdf

[26] DAC0800, DA átalakító: http://www.ti.com/lit/ds/symlink/dac0800.pdf

[27] Memprog2: http://www.elnec.com/products/specialized-programmers/memprog2/

https://www.elnec.com/sw/nnopg_uk.pdf

Függelék

Pin numb	PIC32MX695F512H pin name	Bekötés (NC= not connected, nincs bekötve)
1	ETXEN/PMD5/RE5	LCD DB5
2	ETXD0/PMD6/RE6	LCD DB6
3	ETXD1/PMD7/RE7	LCD DB7
4	SCK2/U6TX//U3RTS/PMA5/CN8/RG6	EEPROM SCK, potméterek CLK
5	SDA4/SDI2/U3RX/PMA4/CN9/RG7	EEPROM SO
6	SCL4/SDO2/U3TX/PMA3/CN10/RG8	EEPROM SI, potméterek SDI
7	/MCLR	NC
8	/SS2/U6RX//U3CTS/PMA2/CN11/RG9	EEPROM /CS
9	VSS	GND
10	VDD	3.3V táp
11	AN5/C1IN+/VBUSON/CN7/RB5	Potméterek /CS
12	AN4/C1IN-/CN6/RB4	DAC /LDAC
13	AN3/C2IN+/CN5/RB3	Amp. gomb
14	AN2/C2IN-/CN4/RB2	LCD E
15	PGEC1/AN1/VREF-/CVREF-/CN3/RB1	LCD R/W
16	PGED1/AN0/VREF+/CVREF+/PMA6/CN2/RB0	LCD RS
17	PGEC2/AN6/OCFA/RB6	Start/Stop gomb
18	PGED2/AN7/RB7	Wave gomb
19	AVDD	3.3V táp
20	AVSS	GND
21	AN8//SS4/U5RX//U2CTS/C1OUT/RB8	UART TX
22	AN9/C2OUT/PMA7/RB9	Up gomb
23	TMS/AN10/CVREFOUT/PMA13/RB10	Down gomb
24	TDO/AN11/PMA12/RB11	NC
25	VSS	GND
26	VDD	3.3V táp
27	TCK/AN12/PMA11/RB12	NC
28	TDI/AN13/PMA10/RB13	NC
29	AN14/SCK4/U5TX//U2RTS/PMALH/PMA1/RB14	NC
30	AN15/EMDC/AEMDC/OCFB/PMALL/PMA0/CN12/RB15	NC
31	SDA5/SDI4/U2RX/PMA9/CN17/RF4	NC
32	SCL5/SDO4/U2TX/PMA8/CN18/RF5	NC
33	USBID/RF3	NC
34	VBUS	NC
35	VUSB3V3	NC
36	D-/RG3	NC
37	D+/RG2	NC
38	VDD	3.3V táp
39	OSC1/CLKI/RC12	Kristályoszcillátor 80MHz, valamint 30pF kondenzátor
40	OSC2/CLKO/RC15	Kristályoszcillátor 80MHz, valamint 30pF kondenzátor

41	VSS	GND
42	RTCC/AERXD1/ETXD3/IC1/INT1/RD8	NC
43	AERXD0/ETXD2//SS3/U4RX//U1CTS/SDA1/IC2/INT2/RD9	DAC /CS
44	ECOL/AECRSDV/SCL1/IC3/PMCS2/PMA15/INT3/RD10	NC
45	ECRS/AEREFCLK/IC4/PMCS1/PMA14/INT4/RD11	NC
46	OC1/INT0/RD0	NC
47	SOSCI/CN1/RC13	NC
48	SOSCO/T1CK/CN0/RC14	NC
49	EMDIO/AEMDIO/SCK3/U4TX//U1RTS/OC2/RD1	DAC SCLK
50	SDA3/SDI3/U1RX/OC3/RD2	földre van kötve, mert nem jön be adat Daból
51	SCL3/SDO3/U1TX/OC4/RD3	DAC Din
52	OC5/IC5/PMWR/CN13/RD4	NC
53	PMRD/CN14/RD5	NC
54	AETXEN/ETXERR/CN15/RD6	NC
55	ETXCLK/AERXERR/CN16/RD7	NC
56	VCAP	10µF, 6V, 1Ω kapacitás pozitív fele, negatív fele GND-n
57	VDD	3.3V táp
58	AETXD1/ERXD3/RF0	NC
59	AETXD0/ERXD2/RF1	NC
60	ERXD1/PMD0/RE0	LCD DB0
61	ERXD0/PMD1/RE1	LCD DB1
62	ERXDV/ECRSDV/PMD2/RE2	LCD DB2
63	ERXCLK/EREFCLK/PMD3/RE3	LCD DB3
64	ERXERR/PMD4/RE4	LCD DB4

Szoftver:

EEPROM:

```
#define Eeprom_read ((unsigned char)0x03) //Olvasás a megadott címtől
#define Eeprom_write ((unsigned char)0x02)//Írás a megadott címtől
#define Eeprom_WRDI ((unsigned char)0x04) //Letiltja az írást
#define Eeprom_WREN ((unsigned char)0x06) //Engedélyezi az írást
#define Eeprom_RDSR ((unsigned char)0x05) //Státuszregiszter olvasása
#define Eeprom_WRSR ((unsigned char)0x01) //Státuszregiszter írása
```

/CS jelek:

```
#define cs_potmeter_active() (mPORTBClearBits(BIT_5))
#define cs_potmeter_deactive() (mPORTBSetBits(BIT_5))
#define cs_dac_active() (mPORTDClearBits(BIT_9))
#define cs_dac_deactive() (mPORTDSetBits(BIT_9))
#define cs_eeprom_active() (mPORTGClearBits(BIT_9))
#define cs_eeprom_deactive() (mPORTGSetBits(BIT_9))
```

Lábak nevei:

```
#define LCD_DB0 PORTEbits.RE0
#define LCD_DB1 PORTEbits.RE1
#define LCD_DB2 PORTEbits.RE2
```

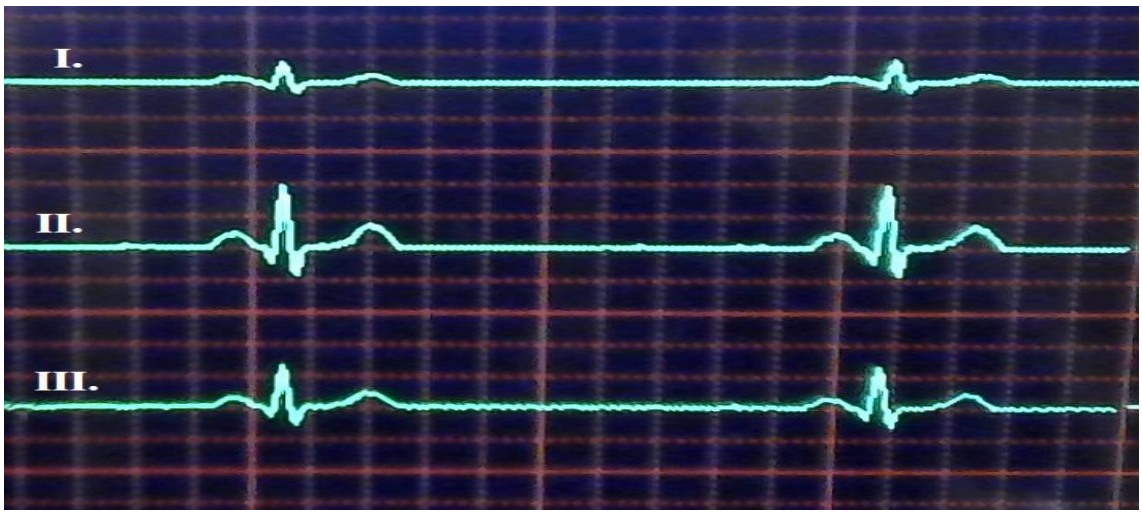
```
#define LCD_DB3 PORTEbits.RE3
#define LCD_DB4 PORTEbits.RE4
#define LCD_DB5 PORTEbits.RE5
#define LCD_DB6 PORTEbits.RE6
#define LCD_DB7 PORTEbits.RE7

#define LCD_RS PORTBbits.RB0
#define LCD_RW PORTBbits.RB1
#define LCD_E PORTBbits.RB2

#define LDAC PORTBbits.RB4

#define But_START_STOP PORTBbits.RB6
#define But_WAVE PORTBbits.RB7
#define But_UP PORTBbits.RB9
#define But_DOWN PORTBbits.RB10
#define But_AMP PORTBbits.RB3
```

EKG minták:
0, 1, 2, 3, 5, 6, 8, 10, 12, 13, 15, 16, 18, 19, 21, 22, 23, 23, 24, 24, 24, 25, 25, 25, 24, 23, 23, 22, 21, 20, 19, 18, 16, 14, 12, 9, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -7, -13, -19, -25, -31, -37, -19, -1, 17, 35, 53, 71, 99, 117, 127, 108, 89, 70, 51, 32, 13, -6, -25, -44, -63, -53, -43, -33, -23, -13, -7, -4, -2, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 2, 3, 4, 4, 5, 6, 7, 7, 8, 9, 10, 11, 11, 12, 13, 14, 15, 17, 19, 21, 23, 25, 27, 30, 33, 35, 36, 37, 38, 38, 39, 39, 38, 38, 37, 37, 36, 35, 34, 33, 31, 29, 27, 25, 23, 21, 19, 16, 14, 11, 9, 7, 5, 3, 1, 0

Deszkamodell osztásszámolás:
Timer alapidő: (1/20MHz)·4·256=51.2µs
EKG jel, 160 mintából áll.
30BPM: T=2000ms. 2ms-onként adunk ki mintát, tehát 320ms aktív, 1680ms passzív szakasz. Timer alapszorzó: 2ms/51.2µs≈39, tehát ezzel még felszorozzuk a timert és így jön ki a 2ms periódusidő. Passzív szakasz: (1680ms/51.2µs)/39≈842.
60 BPM: T=1000ms. 2ms periódusidő, aktív szakasz ua. mint az előbb, passzív szakasz: (680ms/51.2µs)/39≈341.
120 BPM: T=500ms. 2ms periódusidő, aktív szakasz ua. mint az előbb, passzív szakasz: (180ms/51.2µs)/39≈90.
180 BPM: T=333ms. 1ms-onként adunk ki jelet, tehát 160ms-ig aktív. Timer alapszorzó: 1ms/51.2µs≈20. Passzív szakasz: (173ms/51.2µs)/20≈169.
*240BPM:*T=250ms. 1ms periódusidő. aktív szakasz ua. mint az előbb, passzív szakasz: (90ms/51.2µs)/20≈88.
Szinusz jel, az egyszerű időzítés miatt tároltunk 50, 26, 25 és 15 mintából álló szinuszt is, így kerekebben kijöttek az idősítések.
10Hz: T=100ms. 2ms periódusidő a kiadáshoz. Timer szorzó: 39 (fent kiszámolva), 50 mintát adunk ki.
40Hz: T=25ms. 500µs periódusidő. 500µs/51.2µs≈10, ez lesz a szorzó. 50 minta
50Hz: T=20ms. 26 minta. 20ms/26≈769µs periódusidő. 769µs/51.2µs≈15.
60Hz: T=17ms. 25 minta. 17ms/25≈680µs periódusidő. 680µs/51.2µs≈13.
100Hz: T=10ms. 15 minta. 10ms/15≈666µs periódusidő. 666µ/51.2µs≈13
Háromszög jel, 250 mintát tárolunk, ehhez 2ms időzítő periódusidő tartozik, amihez 39-es szorzó, ahogy fent kiszámoltuk

Négyszög jel, mintákat nem tárolunk. Egy 256-os felszorzást kap még az 51.2µs, ebből lesz ≈13ms. 4s/13ms≈305. Ez lesz amivel összehasonlítjuk a timer értéket.

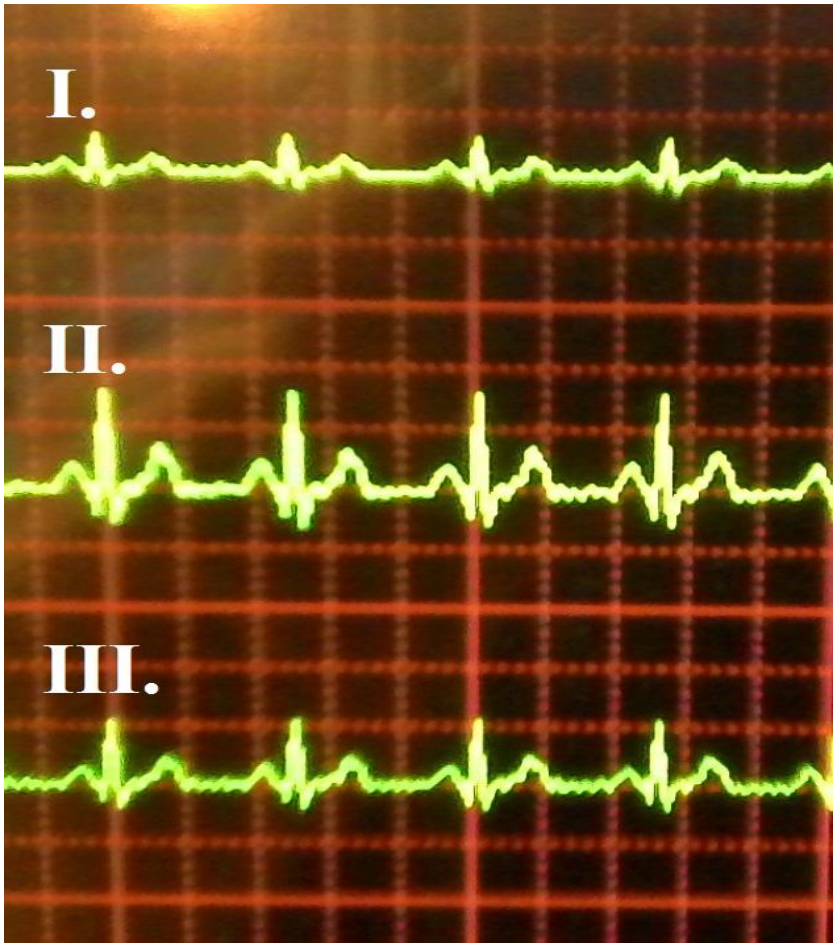
21. ábra 30 BPM



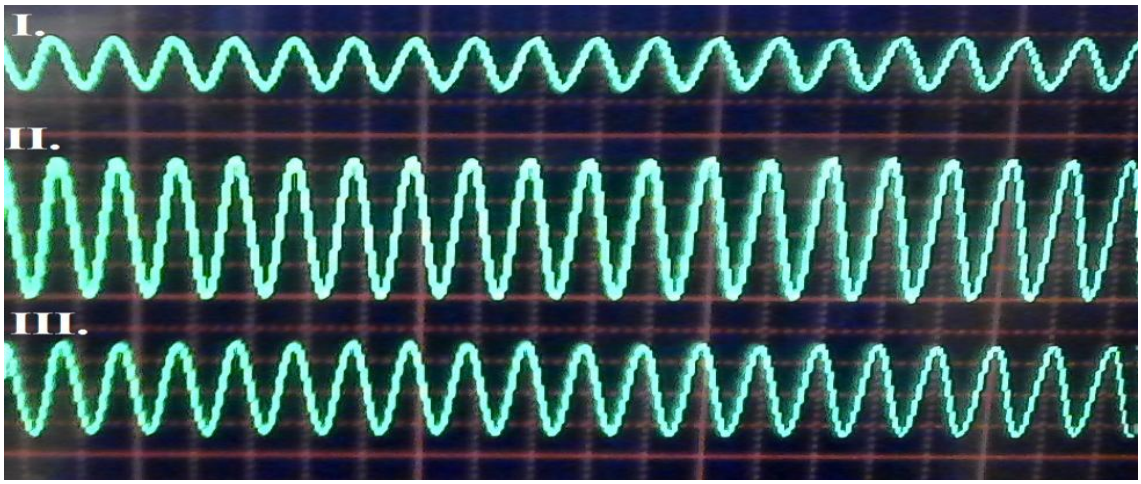
22. ábra 60 BPM

23. ábra 120 BPM

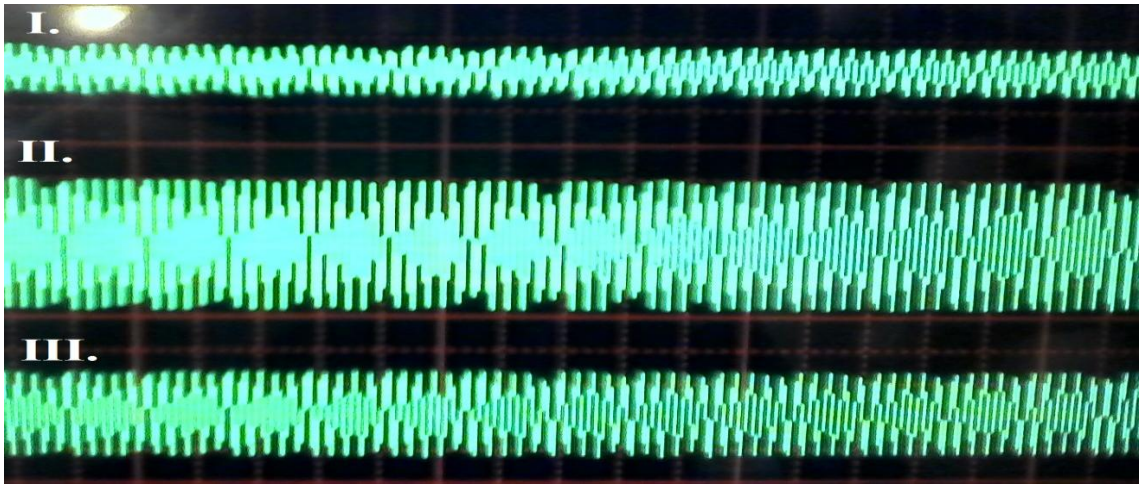
24. ábra 180 BPM



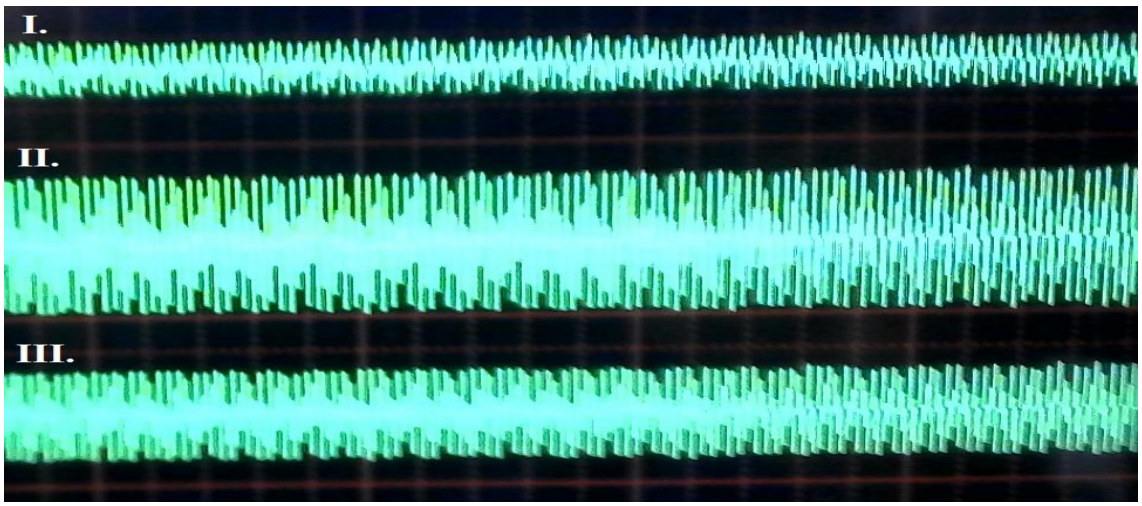
25. ábra 240 BPM



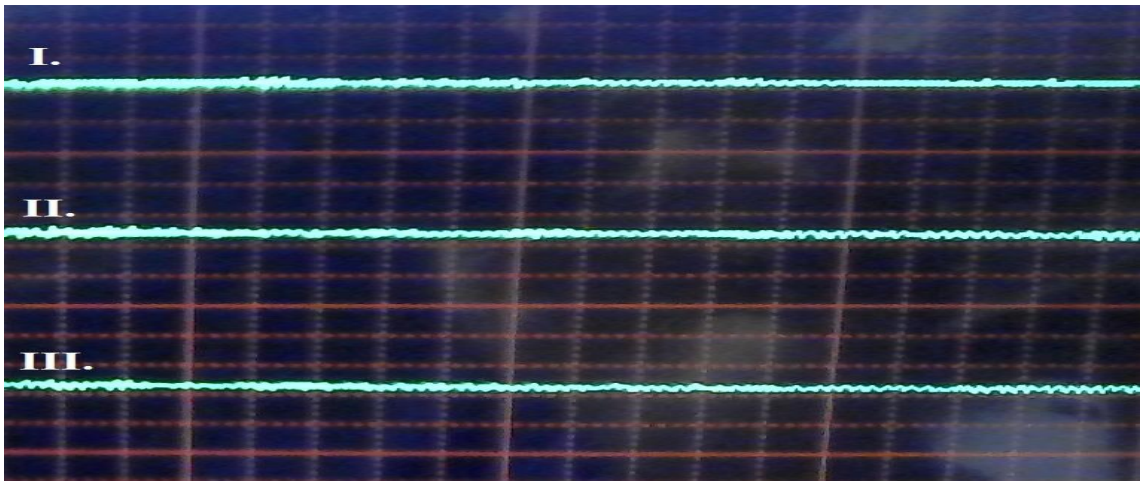
26. ábra 10 Hz szinusz



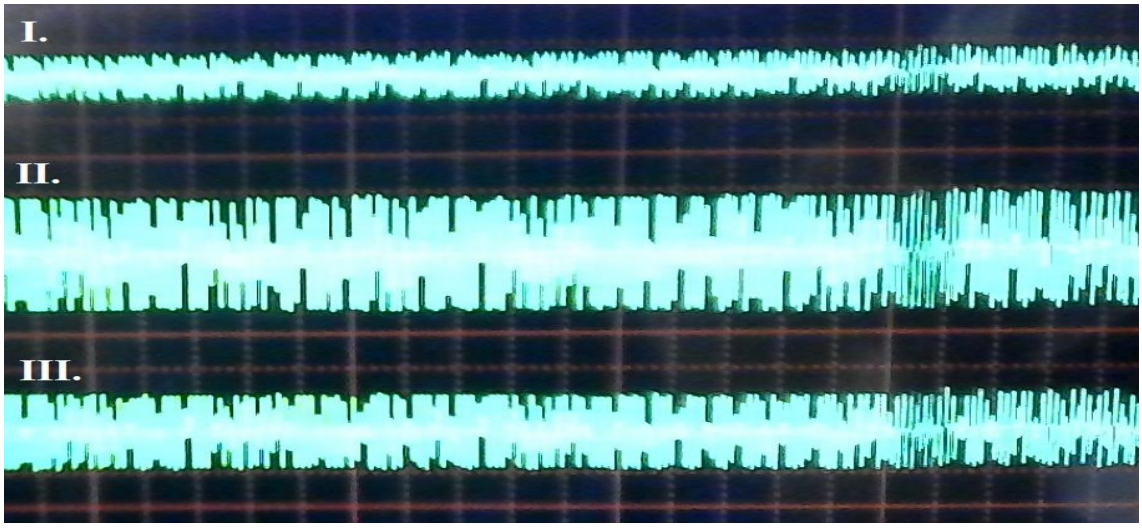
27. ábra 40 Hz szinusz



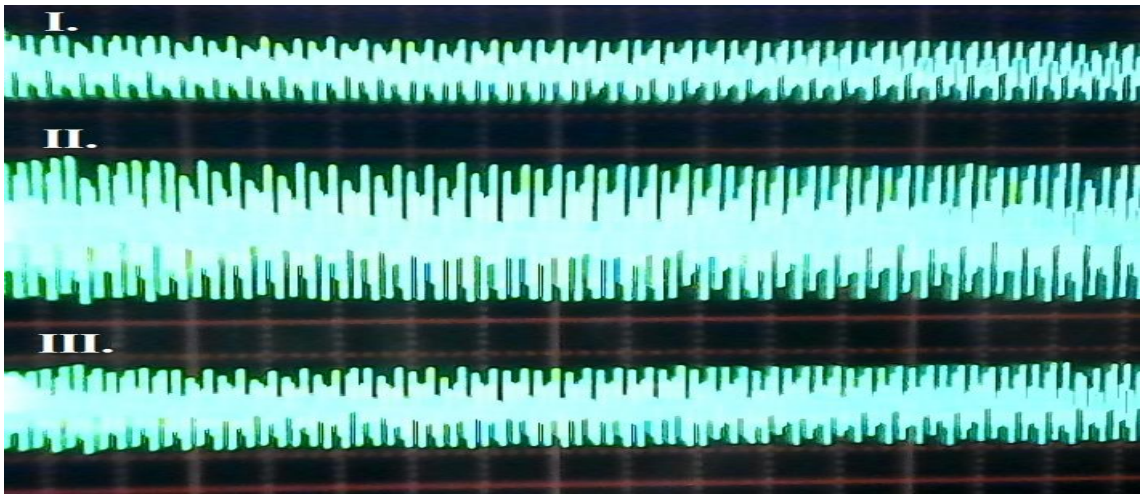
28. ábra 50 Hz szinusz



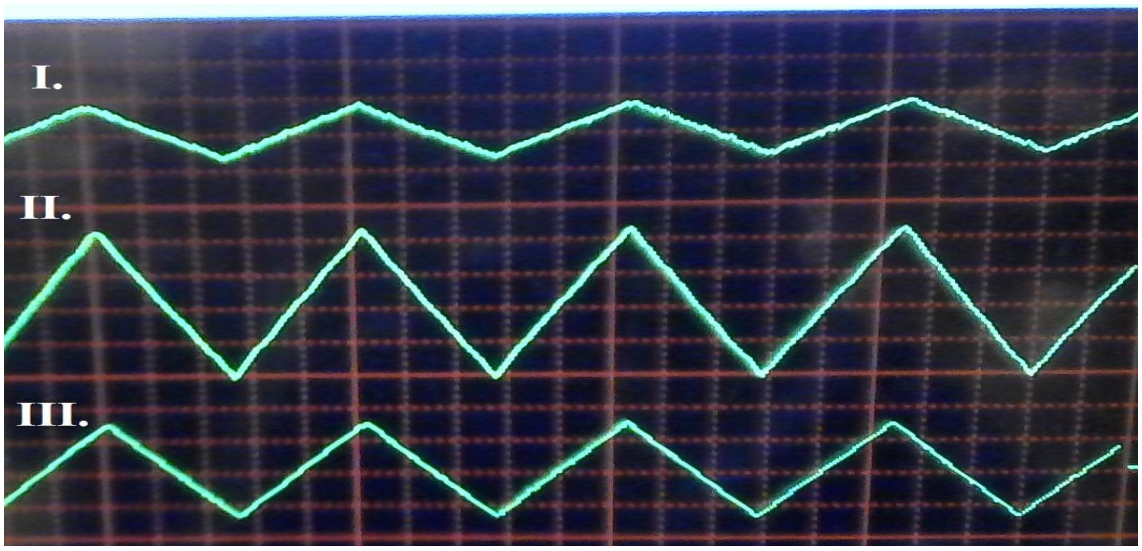
29. ábra 50 Hz szinusz, szűrt



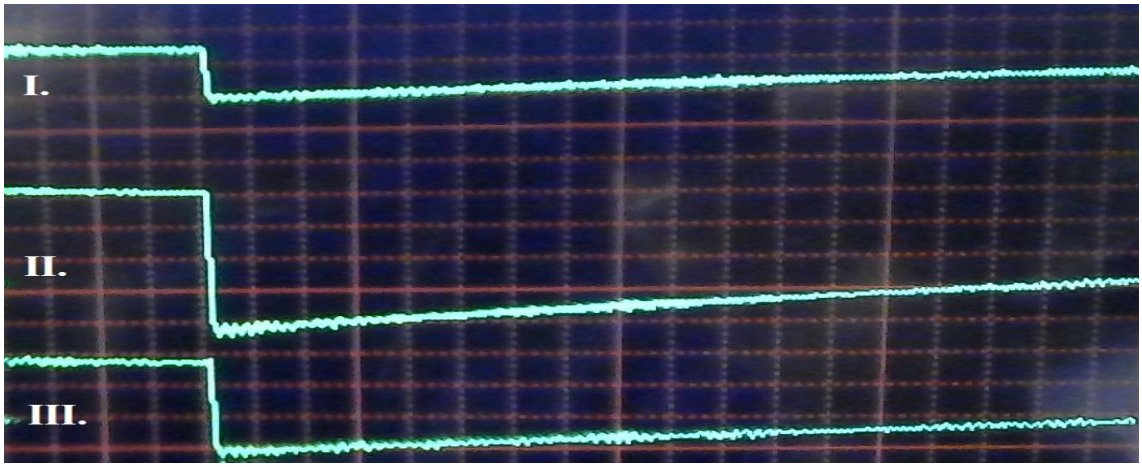
30. ábra 60 Hz szinusz



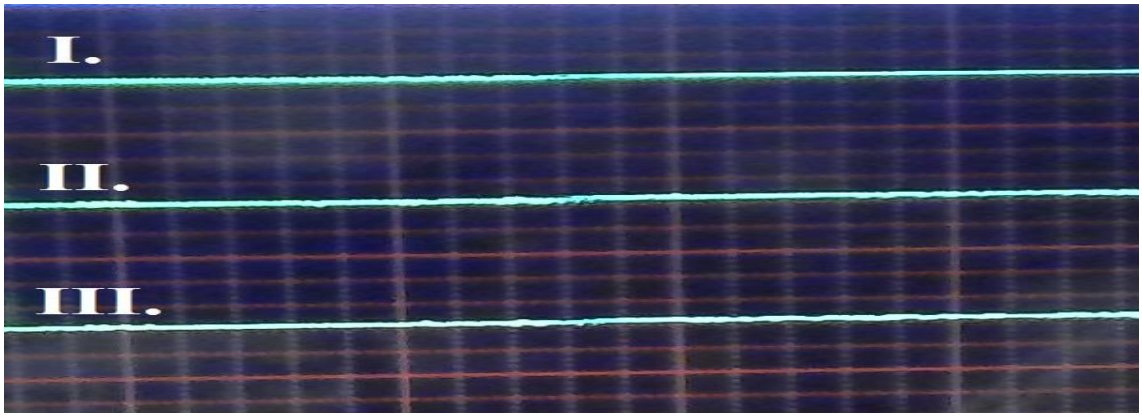
31. ábra 100 Hz szinusz



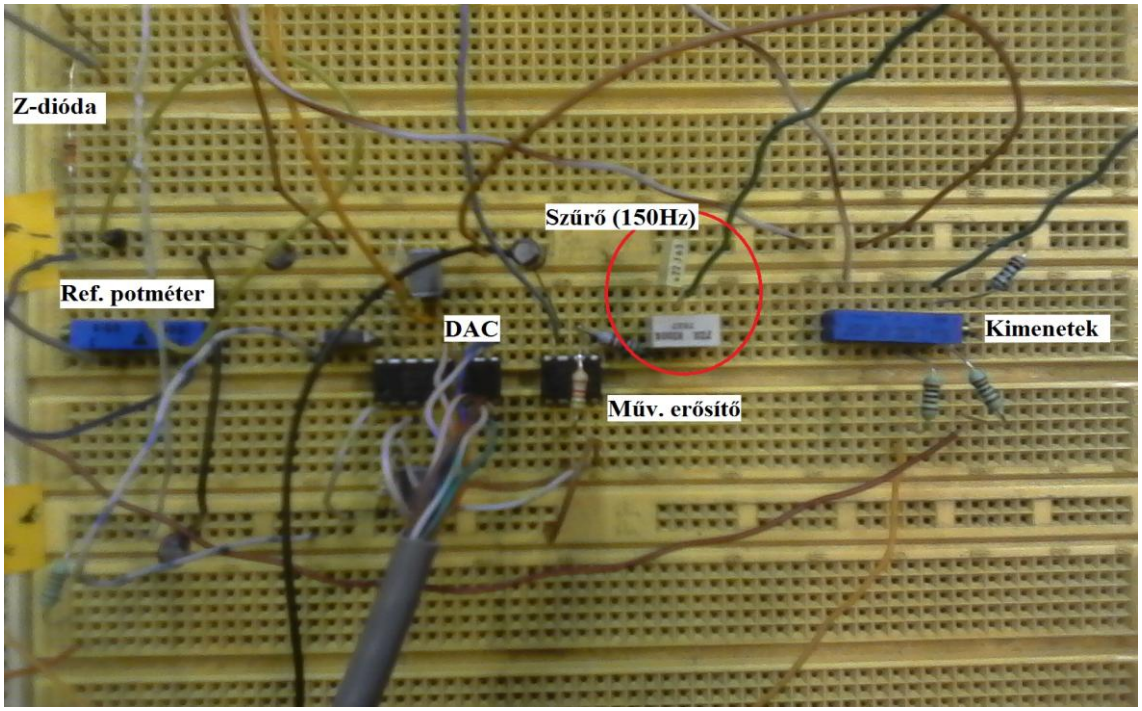
32. ábra Háromszög (2Hz)



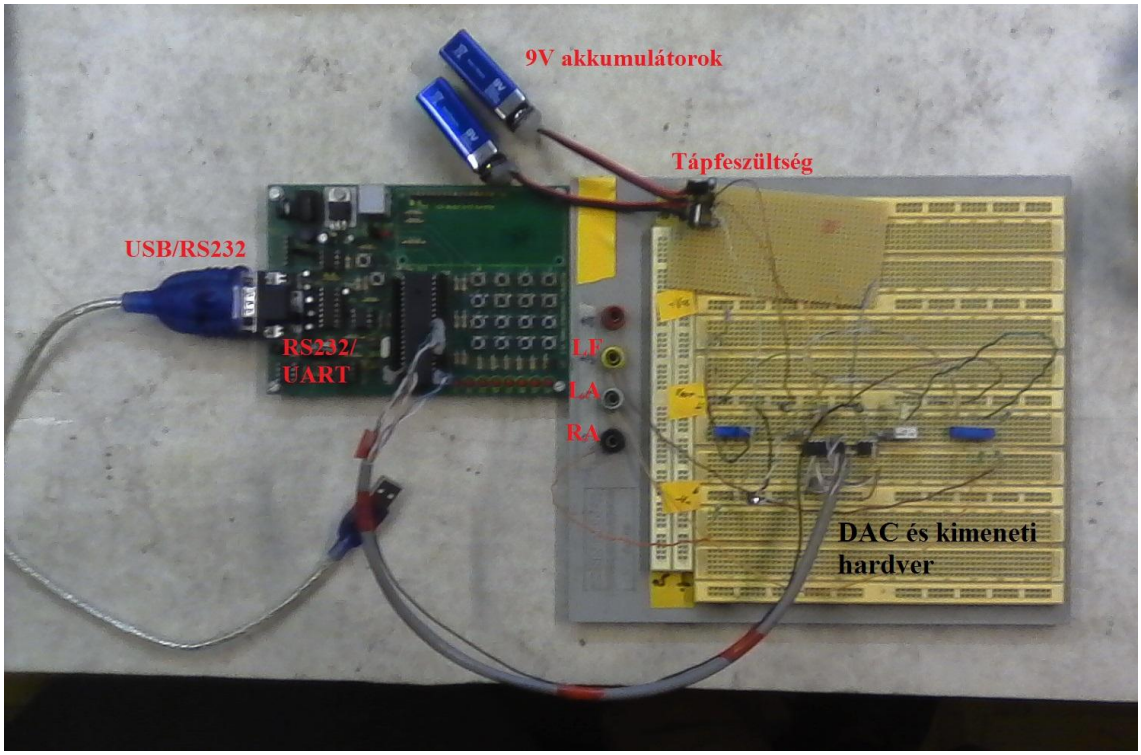
33. ábra Négyszög (0.125 Hz)



34. ábra Szünet



35. ábra Deszkamodell breadboard



36. ábra Deszkamodell teljes