



**Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Automatika Intézet**



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

KERÉKPÁR SEBESSÉGMÉRŐ

Szerző:

Lénárd Tibor András
villamosmérnöki szak, III.
évfolyam

Konzulens:

Lamár Krisztián
főiskolai adjunktus

Budapest, 2011

Tartalomjegyzék

I. Előszó	3
II. A mérés elve	4
III. A hardver	5
Mikrovezérlő	5
Kijelző	5
Energia ellátás	5
IV. A szoftver	6
A megszakításkezelő (ISR)	6
A számítás menete	7
LCD kijelző menürendszere	7
Futási szintek	8
Töltés vezérlés	8
V. Összehasonlítás	9
VI. További fejlesztési ötletek (v2)	9
VII. Irodalomjegyzék	10
VIII. Melléklet	11
1. Kapcsolási rajz - MCU & LCD	11
2. Kapcsolási rajz – Kezelőszervek	12
3. Kapcsolási rajz – Akkumlátor töltés	13
4. Nyákterv	14
5. Futási szintek	15
6. Fotó a kész áramkorról	16

I. Előszó

A dolgozat témája egy kerékpár sebességének mérésére képes áramkör, elsősorban a kerékre szerelt jeladó, szenzor jelére támaszkodva.

Az áramkör elkészítésének ötletét az vetette fel, hogy az előző gyári sebességmérő óráam megsemmisült, de nem voltam megelégedve a kereskedelemben kapható sebességmérők által nyújtott teljesítménnyel, extrákkal.

A tervezéskor szerepet játszott az iskolában tanult programozási alapismeretek felhasználásra való törekvés, ezen készség öntanulása, bővítése. Az alkatrész kiválasztáskor nagy részt már korábban beszerzett, vagy újra felhasználható komponenseket szerettem volna használni, ezzel is minimalizálva az alkatrészkiadásokat.

A kezdeti terveket, ötleteket, sajnos csak kompromisszumokkal sikerült megvalósítani, sajnos a készülék geometriai méreteibe nem fért bele a gps modul és a hozzá tartozó antenna. A készülék fejlesztése sem zárult még le szoftveres oldalról, a menürendszer és az adatgyűjtés kidolgozása, bővítése folyik.

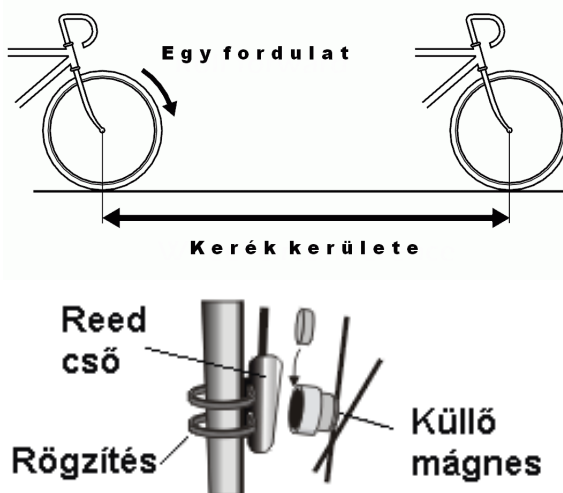
A következőkben szeretném bemutatni az általam készített sebességmérő készülék mérés elvét, az áramköri felépítését, szoftvermegoldásait, összehasonlítást egy gyári sebességmérő órával és a további fejlesztési ötleteket.

II. A mérés elve

Hajdanán, mikor megjelentek az első kerékpáros sebességmérő készülékek, az első tengelyről vett fordulatszámmal arányos jelet egy spirál szálon vezették fel a mérő szerkezetig. Ez a mód hasonló az akkori gépjárművek sebességmérő szerkezetéhez.

Az elektronika miniaturizálásával megjelentek az elektronikus készülékek, ma ezeknek létezik nagyon sok fajtája.

A mérési módszer a következő, a kerékpár külső gumijának futófelületét le kell mérni (ki kell számolni), és ezt az adatot el kell tárolni az órában. A kerék egyik küllőjére felszerelünk egy mágnes, mely a villán elhelyezett reed csöves érzékelőt aktiválja. A reed szenzor jele kábelen (vagy drágább óráknál vezeték nélkül) jut el az óráig.



A sebesség kiszámítás innen már egyszerű, megmérhetjük a két impulzus közötti időt, mely megegyezik a kerék körfordulásának idejével, ebből a kerék kerületének ismeretében számítható a pillanatnyi sebesség.

Amennyiben az úthosszt is szeretnénk rögzíteni, minden impulzus hatására a kerék kerületet hozzá kell adni az eddig megtett úthosszhoz.

A pedál-fordulatszám szenzor nem különbözik a sebesség jeladó szenzortól, de itt a pedál körfordulási idejéből egy percre számított fordulatszámot jelenítünk meg az kijelzőn.

A ma csúcstechnológiának számító órák már GPS segítségével jelzik ki a sebességet, akár fizikai szenzor nélkül. Távlati fejlesztési terv a jelen dolgozatban kifejtett sebességmérőhöz külső GPS készüléket kapcsolni, s a két forrásból származó adatokat párhuzamosan megjeleníteni, elemezni.

III. A hardver

A következőkben bemutatásra kerülnek a készüléket alkotó áramköri elemek. A teljes kapcsolási rajz a mellékletben megtalálható.(1-2-3. számú melléklet)

-Mikrovezérlő

Az áramkör egy PIC 18F2550 típusú 8 bites mikrovezérlőre épül, két oszcillátort tartalmaz, egy fő (20 Mhz) és egy segédoszcillátort (32768 Hz). A mikrovezérlő és perifériáinak órajele a fő oszcillátorról jön, a segédoszcillátort a valós idő alap (RTC) használja fel az óra 1 másodperces léptetéséhez.[1]

A felhasznált perifériák a következők: 4db időzítő modul, 1db TTL UART, 4db bemenet az analóg – digitális átalakítóhoz. A szenzorok jelét két digitális bemenet fogadja, melyek megszakítást generálnak, ha a szenzorokon lefutó él keletkezik.

-Kijelző

A mért értékek kijelzése egy Philips PCD8544 lcd kijelzővel [2] valósul meg, a kezeléshez 4 nyomógomb áll rendelkezésre.

A kijelző 84x48 pixel megjelenítésére képes, ezeket az általam használt módban 84 oszlopra és 6 sorra ossza fel. A kijelző ic teljesen integrált, bele van építve az lcd kijelzőbe, így mindössze 8 érintkezőn keresztül valósul meg a teljes működés.

A kijelző SPI buszon keresztül vezérelhető, működési tápfeszültsége 3,3V, ezen a feszültségen a működés közben felvett árama nagyságrendileg 90uA. A kedvező fogyasztási adatok miatt a kijelzőt nem kapcsolja ki az áramkör, hanem megjelenít rajta egy órát amikor nincs sebességmérés.

-Energia ellátás

A kijelző mellett további fogyasztást jelent a mikrovezérlő, mérés közben nagyjából 8mA áramot fogyaszt, ezen kívül az lcd kijelző 4db fehér ledből háttérvilágítást kapott, ennek fogyasztása kb 50mA. A nagy energia igény miatt elemes táplálás nem jöhetett szóba, így akkumulátort választottam.

Az áramkör energia ellátása egy Li-ion akkumulátorról történik, ennek feszültsége 3,6V-4,2V között változhat, ezért szükségessé vált egy stabilizátor ic beépítése. Ennek kiválasztásakor fontos volt, hogy kevés feszültség essen rajta, mert csak így biztosítható, hogy merülő akkumulátor esetén is stabil maradjon a tápfeszültség. Ezért ULDO (ultra-low-drop-out) típust választottam. Ennek legfőbb jellemzője a nagyon kicsi feszültségesés, a gyártó adatlapja szerint 1mA

fogyasztásnál 7mV, 150mA maximális fogyasztásnál is csupán 300mV.
[3]

Az akkumulátor új problémát vet fel, meg kell oldani a töltését. Az akkumulátor töltését egy belső kapcsoló üzemű „step-down” átalakító végzi, ennek a vezérlő jelét a mikrovezérlő állítja elő a CCP1 hardveres pwm modul segítségével.[1] Az akkumulátor töltéséhez rendelkezni kell több adattal, a pillanatnyi töltőárammal, a pillanatnyi akkumulátor feszültséggel, és ennek megfelelően kell szabályozni a töltést a pwm jel kitöltésének változtatásával. A szoftveres megvalósításnál bővebben kifejtem az akkumulátor töltésére vonatkozó információkat.

IV. A szoftver

Az áramkört működtető programot C-ben írtam, a forráskódot a Microchip C18 (v3.30) fordító segítségével fordítottam le. A fejlesztést az MPLAB 8.56 programban végeztem.

A szoftver megírásakor több fontos szempontot tartottam szem előtt, ezek a következők: A méréshez kapcsolódó kritikus rutinok legyenek az ISR (megszakításkezelő) rutinban, de a nagyobb számítással, vagy adatfeldolgozással járó rutinok „flag” változók hatására a main rutinban hívódnak meg. Mindezek mellett azt is szem előtt próbáltam tartani, hogy ne legyen olyan rutin, mely „leblokkolja” a main rutint, tehát a rutinok csak „lefutnak” de utána visszatérnek a main szálhoz.

-A megszakításkezelő (ISR)

A programban több megszakítást használok, ezek a következők:

Timer0: 0,1ms megszakítás generálása az időméréshez (időalap), szoftveres pwm előállítás a háttérvilágításhoz.

Timer1: Real Time Clock - Valós idő alap funkciót lát el, az óra működtetéséhez, és a kijelző frissítésének „flag” változójának 1-be állításához használok fel.

Timer3: A töltés szabályozás analóg konverzióját indítja másodpercenként kb 256-szor.

ADC: Itt történik a tényleges töltésszabályozás

INT0IF & INT1IF: A külső szenzor bemenetek idéznek elő megszakítást a programban.

Az ISR ciklusban használt globális változók folyamatosan változnak, kiértékelésük a main függvényben nem lehetséges, ezért létrehoztam több globális változót, melyekbe csak „átmásolom” a kritikus pillanatban a fontos számértékeket, majd ezeket dolgozom fel a kiértékelés folyamán.

-A számítás menete

A szenzorok által az ISR-ben mért értékek felhasználása a következőképp történik:

A sebesség számító rutin megkapja tized milliszekundumban két szenzor jel lefutó éle közötti időt:

$$\text{sebesség km/h} = \frac{(\text{kerék.kerület.mm}) * 10022 * 3,6 * 10}{1000 * (\text{idő tized msben}) * 10}$$

majd a megfelelő változóban visszaadja a sebességnek megfelelő értékeket ([egész][tized] km/h)

A pedálfordulat számító rutin a következő képlet szerint dolgozik:

$$\text{pedálfordulat/perc} = \frac{60 * 10022}{\text{idő tized msben}}$$

a számítás végeredményét szintén visszaadja egy változóban, amely később kijelzésre kerül a kijelzőre.

-LCD kijelző menürendszere

Az LCD kijelzőre egy függvény írja ki a menüt (lcd_menu();), ennek működése a következőkön alapszik:

Az menürendszer kezeléséhez van egy változó, ami „szinteket” határoz meg és van még egy tömb típusú változó, mely a szinteken belüli lépkedés értékeit tárolja el. Ez akkor fontos, ha egy menüből visszalépünk, ilyenkor a menüválasztás ugyan annál a sornál áll, ahol beléptünk előzőleg. Ezeknek a vizsgálatát és a menüt alkotó szövegek, ikonok, számok kiírását egy hosszas többszörös elágazó függvény valósítja meg.

A menüt 4 nyomógommbal lehet kezelni, ha bármelyik nyomógombot megnyomjuk, akkor lefut az (lcd_button();) függvény, és végrehajtja a menüben a kívánt mozgást. A menüben végezhető mozgások a gomboknak megfelelően a következőek (balról jobbra):

VISSZA; CSÖKKENT; NÖVEL; BELÉP.

A VISSZA egy szinttel feljebb lép a menüszintek között.

A CSÖKKENT és a NÖVEL a szintnek megfelelő változót módosítja, a BELÉP pedig a következő szintre lép.

Ezeket a lépéseket minden szintnél külön-külön lekódolni rengeteg erőforrás (lassítja a kód fejlesztését) és mindemellett értékes rom területet fogyaszt. Erre azt a megoldást alkalmaztam, hogy egy ciklus megvizsgálja ezeket a feltételeket egy előre meghatározott *konstans táblázat* alapján. Ha vissza/előre kell lépni, kikeresi a

táblázatból, hogy melyik szintre léphet, ha növelni/csökkenteni kell a változó értékét, megkeresi, hogy milyen alsó és felső határok között változtathat, és mennyivel kell léptetnie az értéket. Például az elalvási időzítést 5 másodperces lépésekben állíthatjuk be.

```
const far rom int menu_setting[][6] =
{
  //lim_min, lim_max, léptetés-, léptetés+, prev level, next level
  {0, 1, 1, 1, 0, 1}, //#0 normal display
  {0, 3, 1, 1, 0, 2}, //#1 menu display
  {0, 0, 0, 0, 1, 2}, //#2 jumping level for rows
  {0, 1, 1, 1, 1, 7}, //#3 first row - Daily run
  {0, 1, 0, 0, 1, 1}, //#4 second row - Trip history
  {0, 1, 0, 0, 1, 1}, //#5 third row - Cardio
  {0, 3, 1, 1, 1, 8}, //#6 fourth row - Settings
  {0, 0, 0, 0, 1, 7}, //#7 jumping level for daily delete
  {0, 0, 0, 0, 1, 8}, //#8 jumping level for settings
}
```

- Futási szintek

Az áramkör üzemmódjait egy globális változó mutatja (runlevel), ennek értékei lehetnek 0-1-2-3.

A különböző futási szintek közötti váltáskor az áramkör ki- vagy bekapcsolja az éppen használt időzítő modulokat és perifériákat (5. számú melléklet).

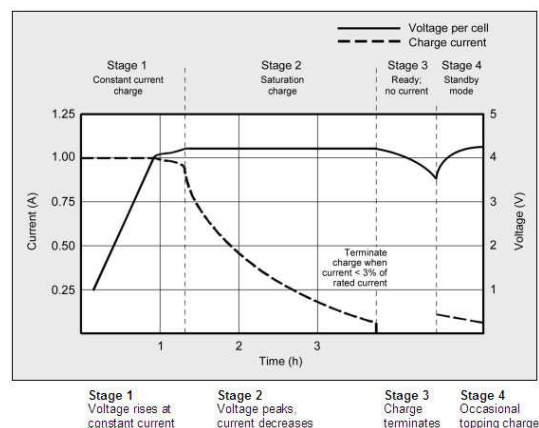
Bekapcsoláskor 0-s szinten indul az áramkör, innen 2-es szintre vált, elindítja az RTC-ért felelős timer1-et, a mérés időalapjául szolgáló timer0-t, engedélyezi a megszakításokat, majd módosítja a futási szintet. Ha letelt a működési idő, és nem történt szenzor esemény, vagy gombnyomás, akkor átkapcsol 1-es szintre, leállítja a timer0-t és végrehajt egy sleep() utasítást, ennek hatására a mikrovezérlő fogyasztása több nagyságrenddel kisebb lesz. Másodpercenként egyszer felébred a processzor, növeli az óra változókat, megvizsgálja a gombok lenyomott helyzetét, leméri a külső töltőfeszültséget, ha egyik se ad igaz választ, akkor újra elalszik.

Amennyiben történt esemény, pl lenyomtak egy gombot, vagy jel érkezett valamelyik szenzorról, akkor azonnal 2-es módba vált az áramkör, amennyiben töltő feszültséget kap az áramkör, úgy 3-as futási szintre vált, inicializálja a pwm modult, a timer2 és timer3 időzítőket. Ezzel megkezdődik a beépített akkumulátor töltése

-Töltés vezérlés

A beépített Li-Ion akkumulátor kapacitása 660mAh. Töltése közben megpróbáltam közelíteni egy ajánlott töltési karakterisztikát [4].

Az akkumulátor töltése (1.szakasz) áramgenerátoros üzemmóddal történik. A szoftver a töltő áramkörrel sorba kapcsolt 0,1ohm nagyságú ellenálláson eső feszültség műveleti erősítővel erősített jelét megméri az ADC



modul segítségével. A mérést másodpercenként 256-szor indítja el, majd a konverzió elkészülte után a megszakítás-kezelő rutinban összehasonlítja a kapott értéket a megfelelő előre definiált konstansokkal. Amennyiben az akkumlátor feszültsége eléri a maximális 4,1V-ot, akkor a szabályzó rutin az áramjel helyett az akkumlátor feszültségét igyekszik konstans értéken tartani (2. szakasz). Ha a töltés vége felé a töltő áram egy beállított érték alá esik (0,05C) akkor a töltés véget ér (3.szakasz).

A szabályzási rutin megszakítás-vezérlőbe történő elhelyezéséhez az ötletet a Microchip Pickit2 áramkörének forráskódja adta [5]

V. Összehasonlítás

<i>Saját sebességmérő</i>	<i>Gyári sebességmérő</i>
gyengébb mechanikai kialakítás	robosztus mechanika
nagyobb méret	kis méret
nagyobb súly	csekély súly
saját igényeknek megfelelő program és adat elemzés	gyártói program
módosítható program	nem módosítható program
árkategóriában extra funkciók	drága extra funkciók, a márkát fizetjük meg

Sikerült az áramkörrel egy hosszabb (142km-es) túrát tenni, kisebb szoftverhiba miatt pontatlanságok jelentkeztek a menet közbeni sebességmérésnél, de a távolság mérése stabilan működött. Azóta a programhiba kijavításra került. Hardveres vagy szoftveres működésképtelenség nem fordult elő a tesztidőszak alatt. A mérési eredmények tárolása és elemzése a jelenlegi szoftverben nem megoldott, de memóriakapacitás rendelkezésre áll az újabb ötletek kivitelezésére.

VI. További fejlesztési ötletek (v2)

Ide tartozik az a már megemlített lehetőség, hogy soros vonalon az áramkörhöz kapcsolhatunk egy külső gps vevőt, majd az adatokat párhuzamosan jelenítjük meg, akár a navigációs adatokkal együtt.

Egyéb fejlesztési lehetőség, egy következő hardver verziónál, a memóriakártyára történő adatrögzítés, esetleg további fizikai szenzormodulok jelének fogadásával együtt (gyorsulásmérő, magasságmérő, erőmérők), vagy a kerékpározó személy fizikai jellemzőinek mérésére kifejlesztett pulzusmérő öv jeleinek fogadása és mentése.

VII. Irodalomjegyzék

[1] Microchip PIC18F2550 adatlap:

<http://ww1.microchip.com/downloads/en/devicedoc/39632d.pdf>

[2] Philips PCD8544 kijelző adatlap:

<http://www.datasheetcatalog.org/datasheet/philips/PCD8544U.pdf>

[3] U-LDO LP2985:

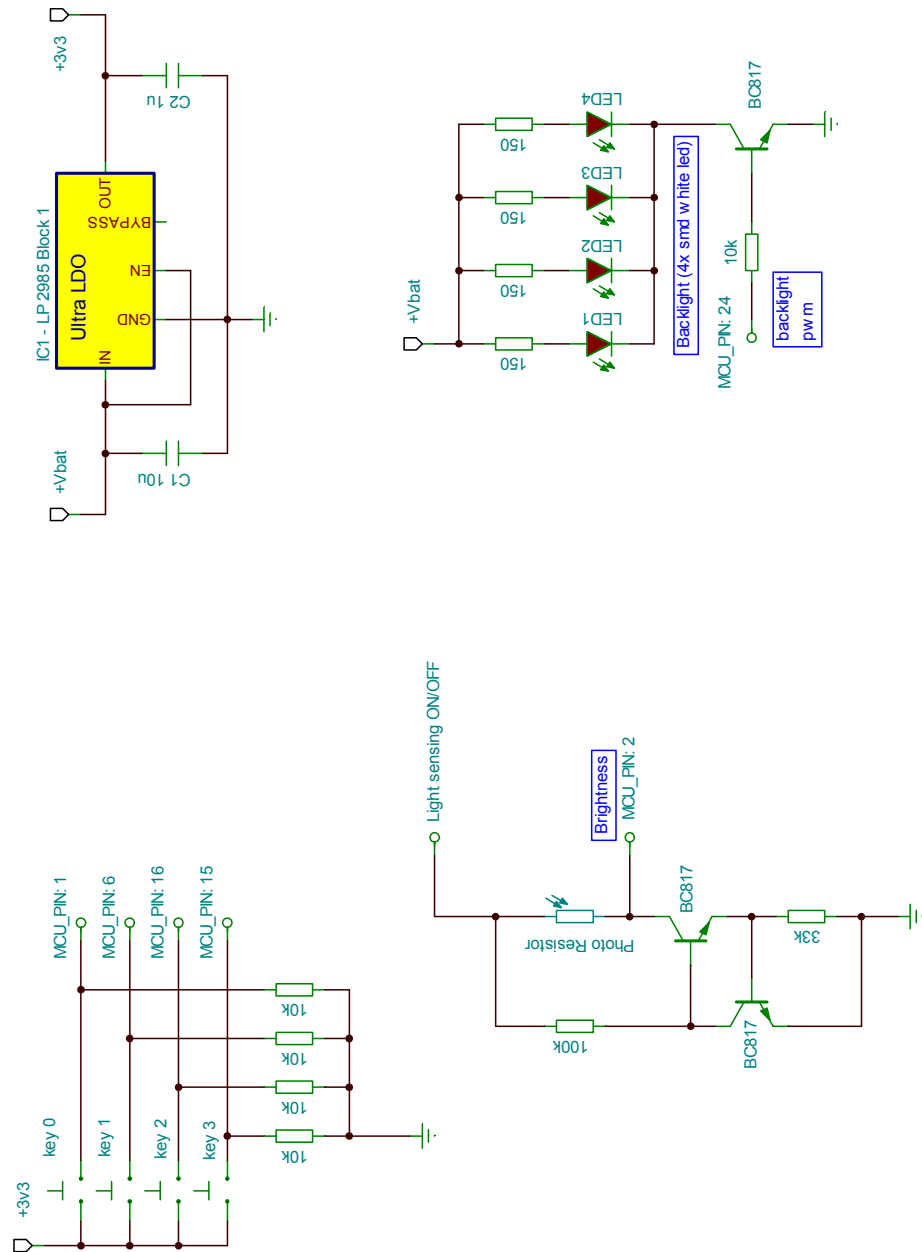
<http://www.national.com/ds/LP/LP2985.pdf>

[4] Charging Lithium-ion (Lithium-ion akkumlátor töltése):

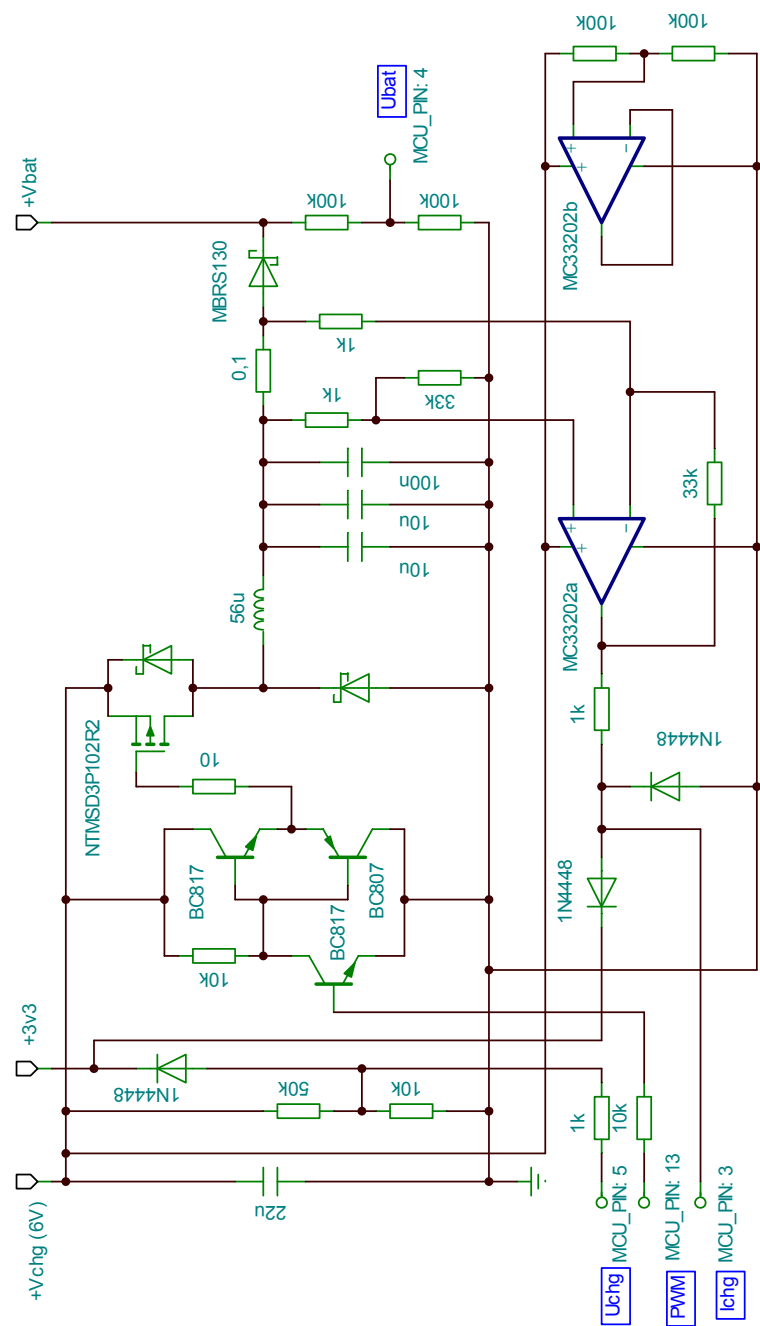
http://batteryuniversity.com/learn/article/charging_lithium_ion_batteries

[5] Microchip Unsupported Source Code V2.32 (Tovább nem támogatott Pickit2 forrás kód) :

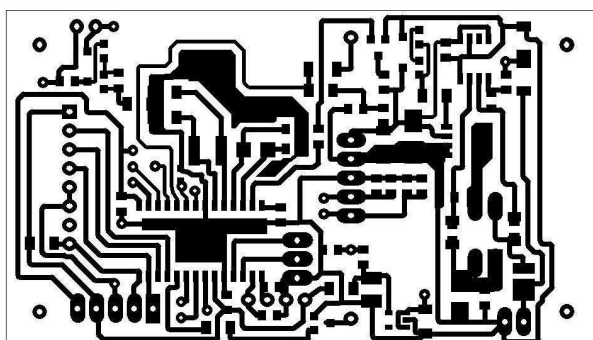
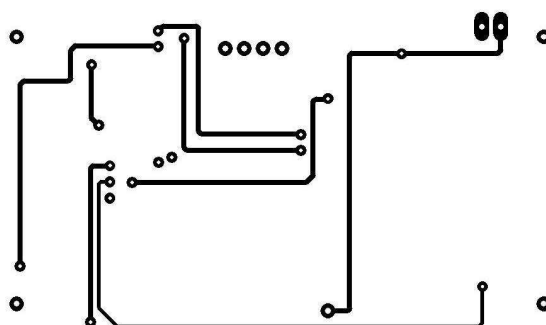
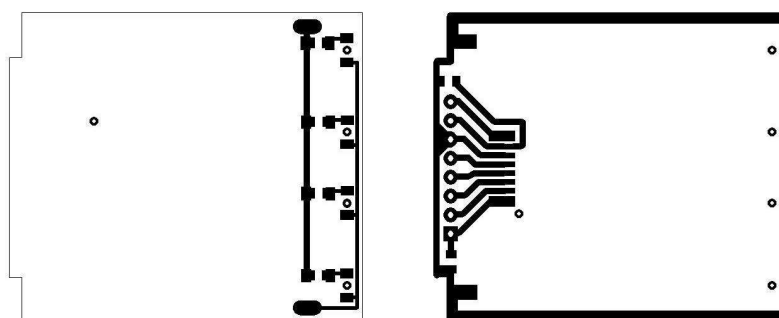
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1960



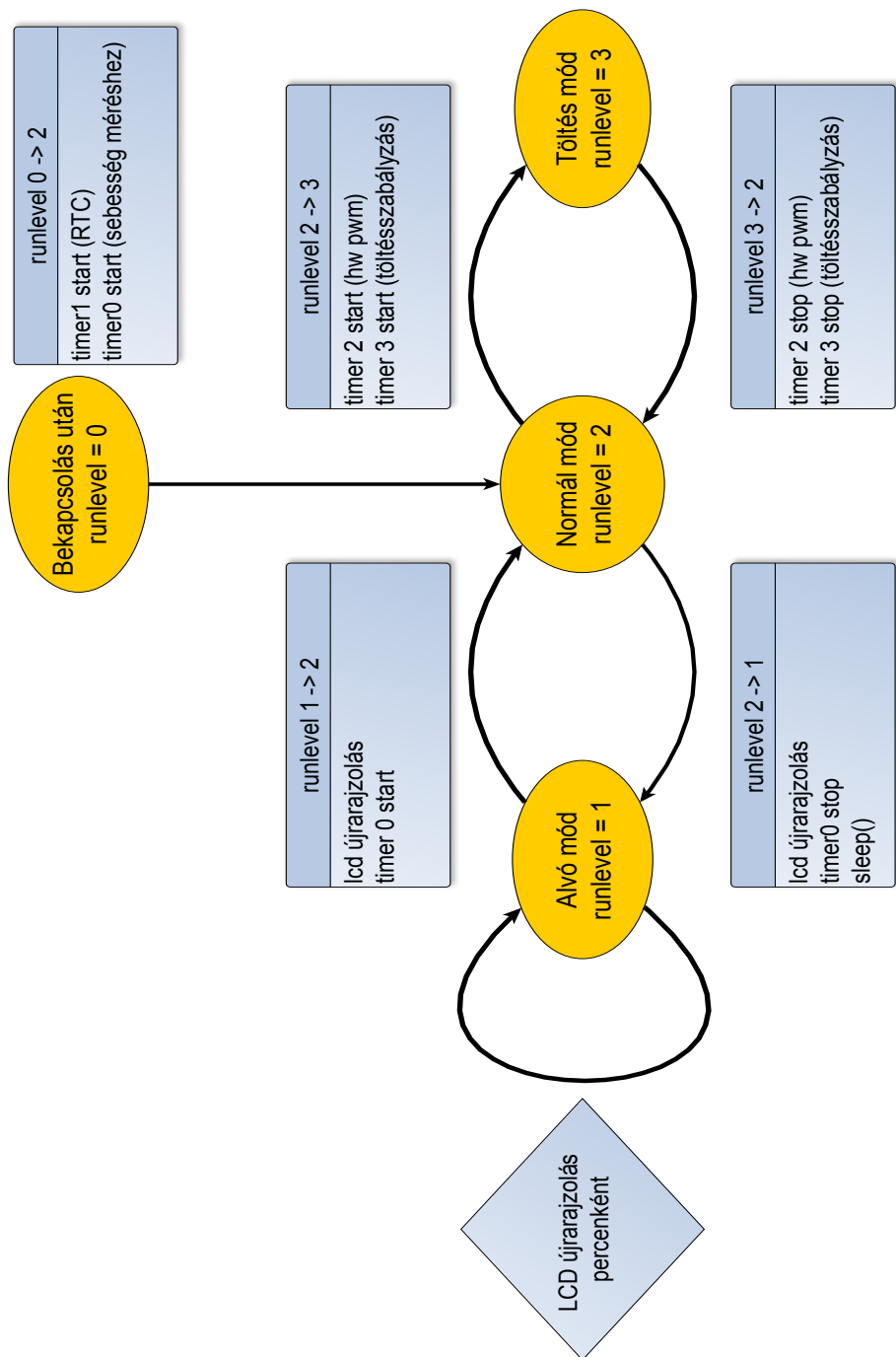
2. számú melléklet – Gombok, háttérfény, ULDO, fényerősség mérő kapcsolási rajza



3. számú melléklet – Töltő áramkör kapcsolási rajza



4. számú melléklet – NYÁK terv



5. számú melléklet – Futási szintek



6. számú melléklet – Fotó az elkészült áramkörről.