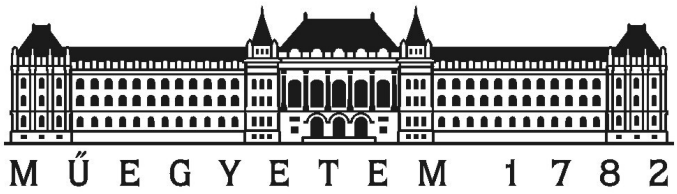


Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Szélessávú Hírközlés és Villamosságtan Tanszék

Püspöki Péter

SMOG-2 elsődleges energiaellátó rendszere

Szakdolgozat

Konzulens

dr. Dudás Levente

Budapest, 2021

HALLGATÓI NYILATKOZAT

Alulírott Püspöki Péter, 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 autentikált 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.

Budapest, 2021. december 8.

Püspöki Péter

Tartalomjegyzék

Kivonat

A SMOG-2 egy PocketQube osztályú 3 PQ méretű ($5 \times 5 \times 15$ cm-es téglatest) műhold, amelyet a Budapesti Műszaki és Gazdaságtudományi Egyetem Szélessávú Hírközlés és Villamosságtan Tanszékén, a Mikrohullámú Távérzékelés Laborban a Műegyetemi Radio Club együttműködésében villamosmérnök hallgatók és oktatók készítenek.

Elsődleges hasznos teherként a fedélzetre kerülő spektrumanalizátor a földi antennák által a világűrbe kisugárzott elektromágneses szennyezést fogja mérni 30 MHz-től 2.6 GHz-ig terjedő frekvenciatartományban.

Az űrben számunkra az egyetlen elérhető megújuló energiaforrás a napfény. A műhold oldalain elhelyezett napelem táblák a Napból érkező sugárzott elektromágneses energiát vezetett elektromos energiává alakítják. A műhold kis mérete korlátozza a rajta elhelyezhető napelemek számát és így a bejövő teljesítményt is. Ahhoz, hogy a cellákból a lehető legtöbb teljesítményt ki tudjuk venni, egy maximális munkapont követő algoritmust alkalmazó tápegységre van szükség, ami tölti a fedélzeten elhelyezett akkumulátorokat.

Dolgozatom célja egy ilyen energia ellátó rendszer kifejlesztése, megépítése és bemérése.

Abstract

SMOG-2 is a PocketQube class 3 PQ size ($5 \times 5 \times 15$ cm cuboid) developed by electrical engineer students and lecturers at the Department of Broadband Infocommunications of the Budapest University of Technology and Economics in the Microwave Remote Sensing Laboratory in cooperation with the Radio Club of Budapest University of Technology and Economics.

The primary payload of the satellite is a spectrum analyzer that will measure the electromagnetic pollution emitted by terrestrial antennas into space in the frequency range of 30 MHz to 2.6 GHz.

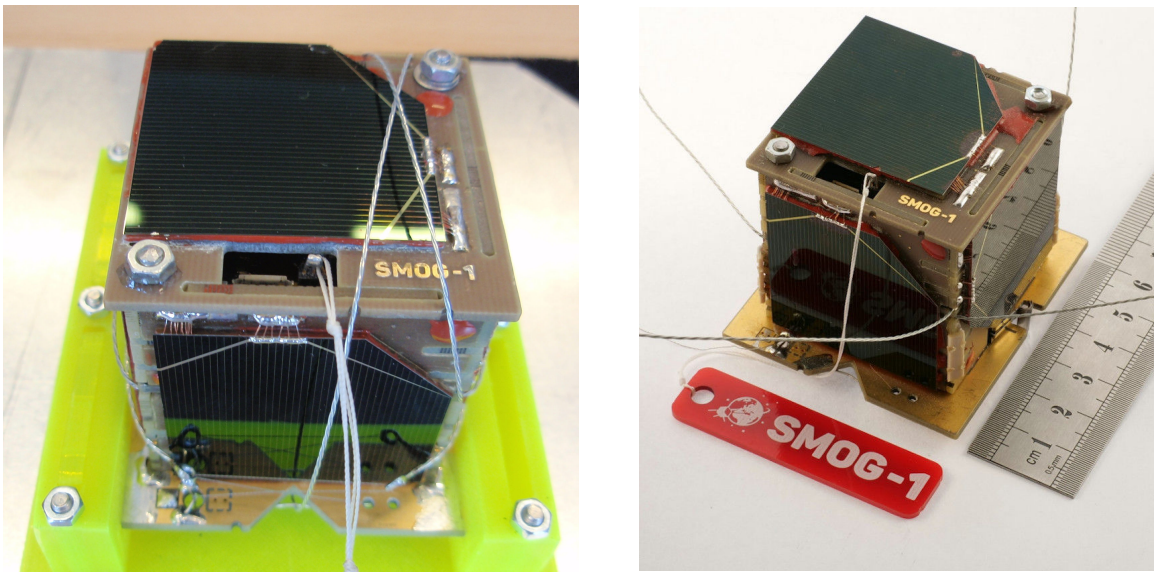
In space the only renewable energy source available to us is sunlight. Solar panels are placed on the sides of the satellite that can convert the radiated electromagnetic energy into conducted electrical energy. The small size of the satellite limits the number of solar panels that can be placed on it and thus also the incoming power. In order to extract as much power as possible from the cells a maximal power point tracking algorithmic power supply is required to be used that can charge the batteries placed onboard.

The aim of my dissertation is to develop, build and measure such an energy supply system.

1. fejezet

Bevezetés

1.1. Előzmények

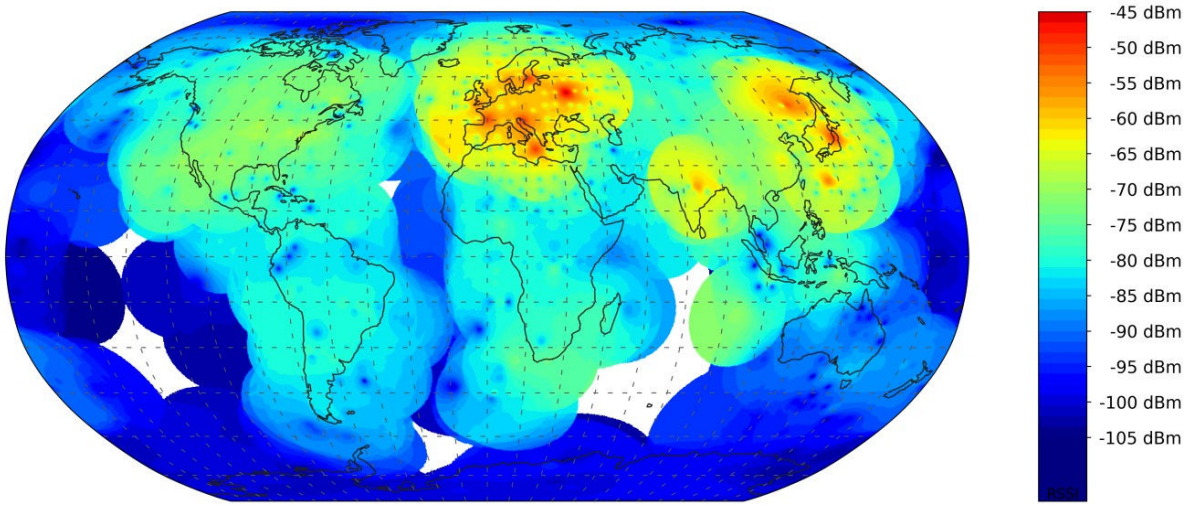


1.1. ábra. SMOG-P és SMOG-1 [1]

A fenti képeken a SMOG-P és a SMOG-1 műholdak láthatóak, amelyek a Budapesti Műszaki és Gazdaságtudományi Egyetemen készültek. (1PQ méretű PocketQube, $5 \times 5 \times 5$ cm)

A digitális földfelszíni TV adók által kisugárzott rádióhullámok nemcsak a földfelszínen lévő vevőkészülékek irányába terjednek, hanem az űrbe is kijutnak. Ez a Föld körül keringő műholdak kommunikációja számára zavaró lehet, másrészt az adó állomások nem megfelelően irányított antennái miatt ez jelentős energia pazarlást jelent.

A SMOG-P és a SMOG-1 műholdak elsődleges küldetése az volt, hogy a fedélzetükön elhelyezett 430–860 MHz-es sávban működő spektrumanalizátor segítségével megmérjék a Föld körüli pályájukon ezt az elektroszmogot. A méréseik alapján elkészült a világon először a Földet körülvevő elektromágneses szennyezettséget ábrázoló térkép, ez látható az 1.2. ábrán.



1.2. ábra. A SMOG-P mérései alapján készített elektroszmog-térkép [2]

Erről a térképről látható, hogy mennyire pazarlóak a földfelszíni műsorszóró és egyéb telekommunikációs szolgáltatók által jelenleg használt rendszerek. A kibocsájtott jelek jelentős része nem a felhasználókhoz jut el hanem a világűrbe, ahol akkora fedőtérerősséget hoznak létre, hogy a műholdvezérlő földi állomásoknak több 10, akár több 100 W-os adóteljesítményre van szükségük.

Egy ilyen térkép segíthet a legtöbb szennyezést kibocsájtó területek meghatározásában.

Az eddigi méréseink alapján a földfelszíni TV adók frekvenciáján jelentős elektroszmog mérhető a Föld körül. Felmerülhet a kérdés, hogy más frekvenciákon is hasonló-e a helyzet. Ennek a megválaszolására készítjük el a SMOG műholdak következő példányát, a SMOG-2-t.

2. fejezet

SMOG-2

2.1. Környezeti viszonyok az űrben

Ahhoz, hogy a műhold az űr szélsőséges körülményei között is megbízhatóan működjön, a létfontosságú alrendszereket redundánsan kell tervezni, hogy egy pont meghibásodás ellen védettek legyenek. Ez azt jelenti, hogy ha bármelyik alkatrész meghibásodik, átmegy szakadásba vagy rövidzárba, a műholdnak akkor is működőképesnek kell maradnia. Az elsődleges energia ellátó rendszer esetében a tartalékolás rendszerszinten valósul meg, egymástól függetlenül működő áramkörök tartoznak minden napelemoldalhoz.

Az űrben a termikus körülmények lényegesen eltérnek a földihez képest. A napból érkező sugárzás jelentősen fel tudja melegíteni a műholdnak az oldalait. Földárnyékban viszont 2,7 Kelvines hőmérsékletű háttérsugárzás mellett nagyon le tud hűlni a műhold. A fedélzeten található alkatrészek nagy része $-40\,^{\circ}\mathrm{C}$-tól akár $85\,^{\circ}\mathrm{C}$-ig is működik, azonban az akkumulátor $0\,^{\circ}\mathrm{C}$ felett használható, így annak a megfelelő hőmérsékleten való tartása kiemelten fontos. A hő terjedését jelentősen befolyásolja az űrben található vákuum, közeg általi hő szállításra itt nincs lehetőség. A hőt termelő alkatrészek hűtése csak hővezetéssel és hősugárzással lehetséges.

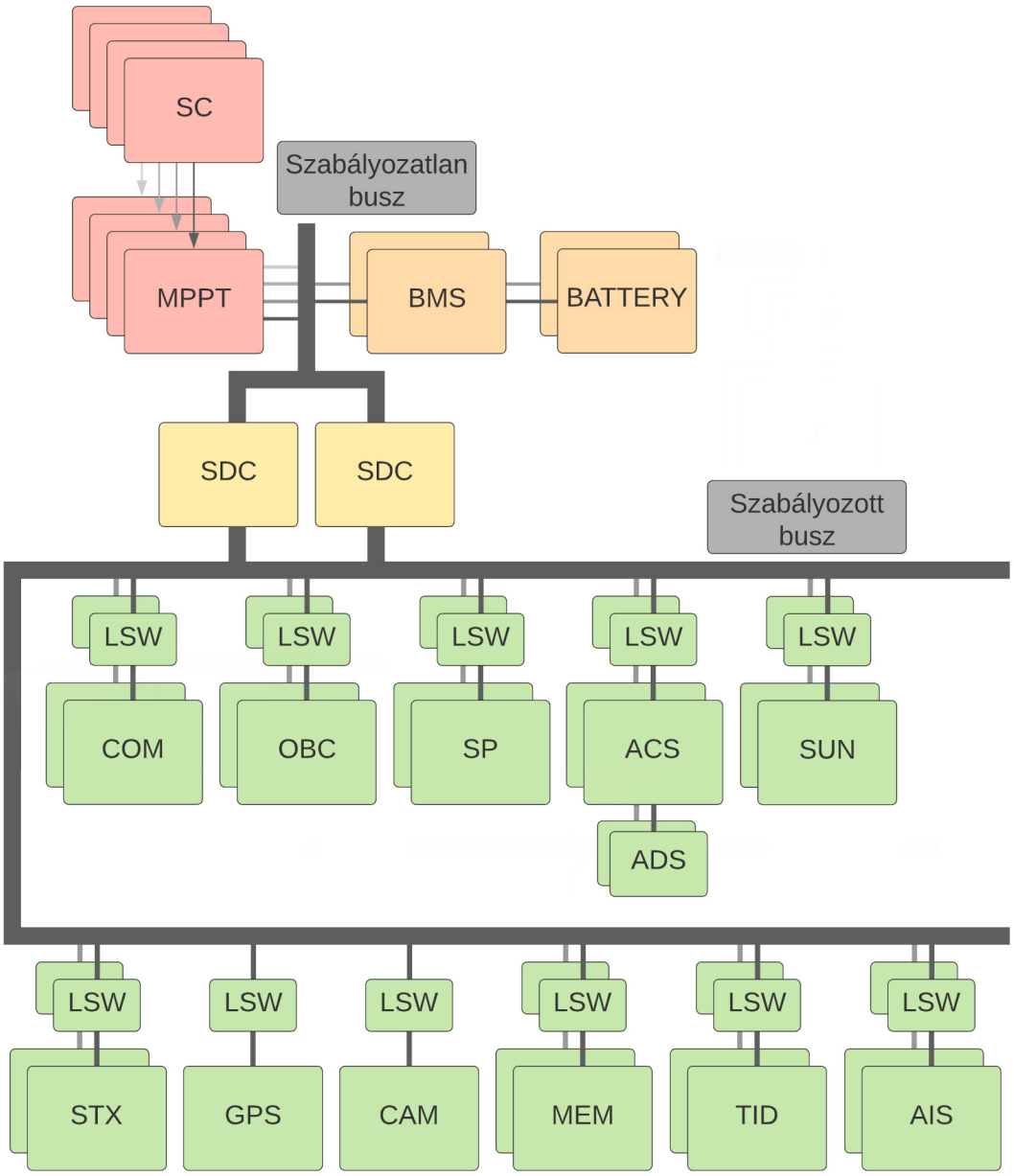
2.2. Mechanikai szerkezet

Minden oldal- és belső lemez 1,6 mm-es FR-4 NYÁK-lemezből készül. Ennek köszönhetően minden felületre kerülhet elektronika.

2.3. Rendszerterv

A műhold rendszerterve a 2.1. ábrán látható. A pirossal jelölt rész az elsődleges energia ellátó rendszer, a narancssárga a energia tároló rendszer, a citromsárga a másodlagos energia ellátó rendszer, a zöld pedig a szabályzott buszról működő alrendszerek.

A napelemből illetve a hozzájuk tartozó MPPT áramkörből négy darab kerül a műholdra, a többi alrendszer körül a legtöbből a redundancia megvalósítása céljából két példány lesz.



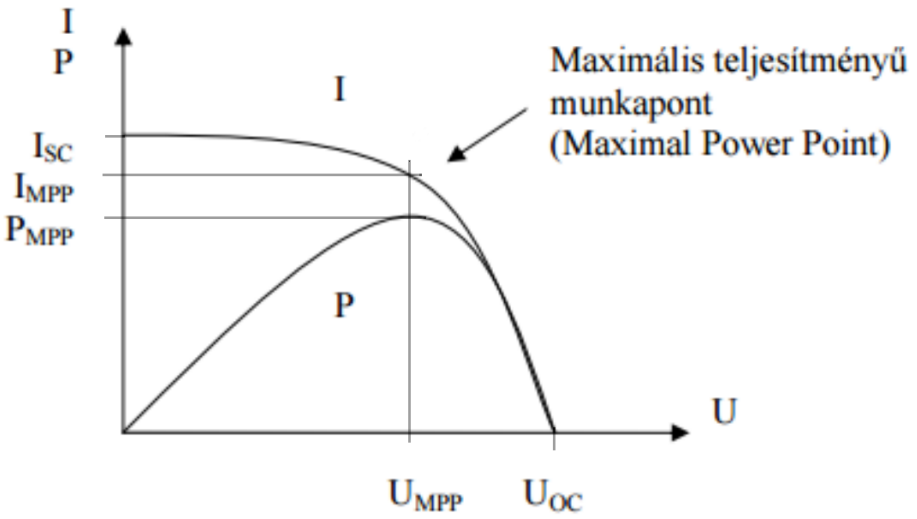
2.1. ábra. A SMOG-2 rendszerterve

- SC: Solar Cell; Napelem cella
- MPPT: Maximal power point tracker; Maximális munkapont követő áramkör
- BMS: Battery Management System; Akkumulátor védő elektronikája
- BATTERY: Akkumulátor
- SDC: Step Dowm Converter; Feszültség csökkentő áramkör
- LSW: Limiter Switch; Túláram védő kapcsoló
- COM: Communication; Kommunikációs egység
- OBC: On Board Computer; Fedélzeti számítógép
- SP: Spectrum Analizer: Spektrumanalizátor

- ACS: Attitude Control System; Helyzet szabályozó rendszer
- ADS: Attitude Determination System; Helyzet meghatározó rendszer
- SUN: Sun sensor; Nap szenzor
- STX: S-band transmitter; S sávú adó
- GPS: Global Positioning System; Globális helymeghatározó rendszer
- CAM: Camera; Kamera
- MEM: Memory; Memória
- TID: Total Ionizing Dose; Totál ionizáló dózis mérő
- AIS: Automatic Identification System; Automatikus azonosító rendszer

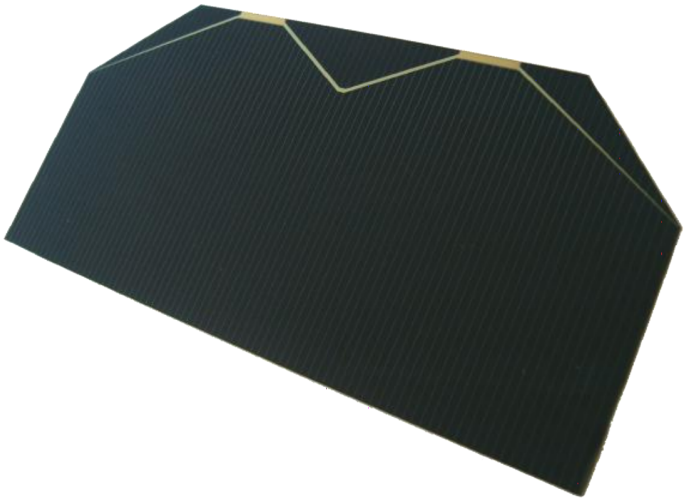
2.4. Napelem cella (SC)

Az ilyen kisméretű műholdak számára az egyetlen lehetséges energiaforrás az űrben a napfény, ebből kell biztosítani a műholdak számára szükséges energiát. A műhold két kisebb 5×5 cm-es oldalát antennák foglalják majd el, napelemek csak a négy nagyobbik oldalon lesznek. Ezeken az oldalakon kettő darab napelem cellának van hely. Minden oldalhoz tartozik egy maximális munkapont követő áramkör, ami biztosítja, hogy a napelemekből a lehető legtöbb teljesítményt fel tudjuk használni.



2.2. ábra. A napelemcella áram-feszültség karakterisztikája [3]

A műholdakon használt napelemcella az Azur Space [4] által gyártott TJ Solar Cell 3G30C - Advanced típusú három rétegű (GaInP/GaAs/Ge) napelemcella [5].



2.3. ábra. TJ Solar Cell 3G30C [4]

A napból elektromágneses sugárzás formájában nagyjából 1360 W teljesítmény érkezik egy négyzetméter felületre. A műhold hatból négy oldallemezein elhelyezkedő napelem táblák mérete 40×80 mm, ezeknek a két sarkán $13,5 \times 13,5$ mm-es letörések találhatóak. A napelemek hatásfoka az űrben 28,5% körül van és a műhold tervezett pályájának a tulajdonságaiból adódóan, a keringési idő 60%-át fogja napon tölteni. Ezen paraméterek ismeretében kiszámítható a körátlagra vonatkoztatott DC bejövő teljesítmény:

$$P_{DC} = 1360 \text{ W} \cdot \frac{(40 \text{ mm} \cdot 80 \text{ mm} - (13,5 \text{ mm} \cdot 13,5 \text{ mm})) \cdot 2}{(1000 \text{ mm} \cdot 1000 \text{ mm}) \cdot 4/6} = 935,74 \text{ mW} \qquad (2.1)$$

A Napból érkező sugárzás egy részét a légkör elnyeli, ezért az űrben tapasztalható 1360 W/m^2 helyett a Föld felszínén nagyjából 1000 W/m^2-rel lehet számolni. Így a bejövő átlag teljesítmény a 2.1. képlet alapján 688,05 mW lesz. A tervezéskor célszerű ezt az értéket használni, hiszen ezzel tudjuk tesztelni a műholdat, illetve ha így is sikerül pozitív energia mérleget elérni, akkor az űrben ennél még kedvezőbb lesz a helyzet.

2.5. Elsődleges energiaellátó rendszer (EPS1)

Elsődleges energiaellátó rendszer feladata a napelemekhez tartozó MPPT áramkör segítségével ellátni a műholdat a működéséhez szükséges energiával. A feszültségszabályzó áramkörei a napelemekről dolgozva a műhold szabályozatlan energiabuszára állítanak elő az akkumulátor töltéséhez megfelelő feszültséget.

2.6. Központi energiaellátó rendszer (EPS2)

A központi energiaellátó rendszer feladata a szabályozatlan feszültségű buszból előállítani egy szabályzott energiabuszt, amiről a többi alrendszer tud működni. Az EPS2 része egy feszültségcsökkentő kapcsolás, az SDC (step down konverter), amely a fedélzeten található alrendszerek részére állít elő 3,3 V-os feszültséget. Ehhez az áramkörhöz tartozik még a PCU (power control unit) vezérlő egysége, amely vezérli az alrendszerek energia ellátását.

2.7. Fedélzeti számítógép (OBC)

A műhold fedélzeti számítógépe felelős a mérési feladatok ütemezéséért, a mérési és telemetria adatok flash memóriában történő tárolásáért, illetve a földi állomástól érkező parancsok végrehajtásáért. Az alkalmazott processzor egy PIC32MK1024GPK064 típusú IC. A fedélzeti számítógép és minden alrendszer között meg van valósítva egy kommunikációs csatorna, ezek nagy része fél-duplex UART protokollt használ, így alrendszerenként egy-egy vezetékre és mikrokontroller lábra van szükség.

2.8. Kommunikációs rendszer (COM), és S-sávú adó (STX)

A kommunikációs rendszer feladata a földi állomással való kapcsolat megvalósítása. A kommunikáció 70 cm-es amatőr sávban történik. Ez az rendszer fogadja a földi állomástól érkező vezérlőparancsokat, illetve küldi a mérési és telemetria adatokat.

A műholdon helyet kap egy S sávban működő adó is, ezzel lényegesen nagyobb adatátviteli sebességet lehet elérni, ami a sok kísérlet által generált adatok letöltését segíti.

2.9. Spektrum analizátor (SP)

A SMOG-2 elsődleges küldetése a SMOG-P és SMOG-1 által már elkezdett elektromágneses szennyezés mérése a fedélzetén elhelyezett, az elődeihez képest kiterjesztett frekvencia tartományban 30 MHz-től 2,6 GHz-ig működő spektrumanalizátor segítségével.

2.10. Egyéb kísérletek

TID

A műholdat érő és az elektronikáját károsító ionizáló sugárzás mérése.

MEM

Szintén a sugárzás hatását vizsgáló kísérlet, egy memóriába ismert adatot írva lehet vizsgálni, hogy az megváltozik-e a műholdat érő sugárzás hatására.

ACS

Aktív elektromágneses helyzet szabályozó rendszer, mellyel a tervek szerint a műhold forgása megállítható lesz, illetve tetszőleges irányba tudjuk majd fordítani a műholdat.

2.11. Földi állomás

A műholddal történő kommunikáció másik oldalán található a földi állomás. Feladata a műholdtól érkező adatok vétele és a vezérlőparancsok küldése.

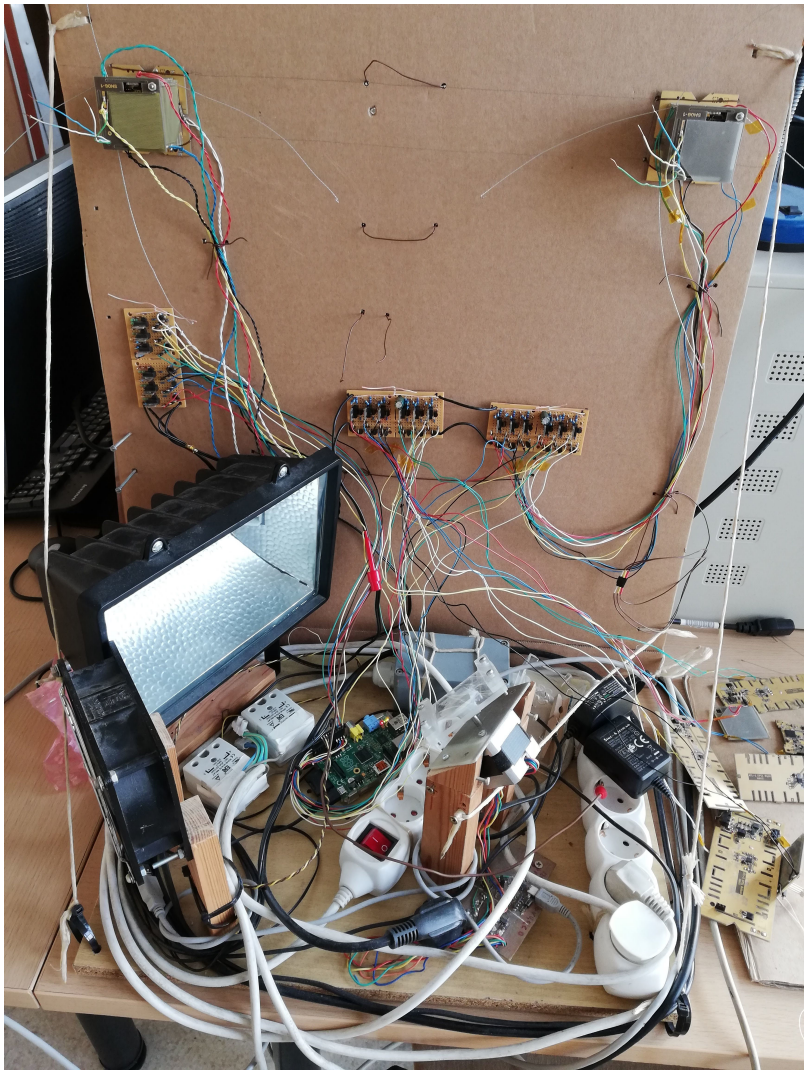
Mivel a műhold antennájának nincs túl nagy nyeresége és a kisugárzott teljesítmény sem lehet tetszőlegesen nagy, ezért a Földön kell készíteni egy nagy nyereségű, kis nyalábélességű antennát, amelyet az áthaladó műhold irányába lehet forgatni. Az elsődleges vezérlőállomásunk az Egyetem E épületének tetején található, itt egy 4,5 méter átmérőjű paraboloid reflektorral ellátott hátrafelé sugárzó helix antennát használunk.

2.4. ábra. Az elsődleges földiállomás antennája

3. fejezet

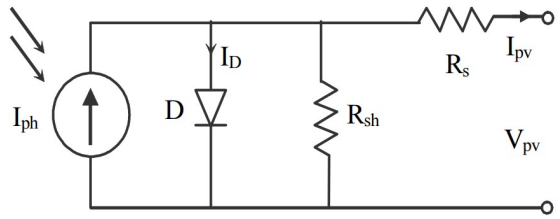
Napelem emulátor

Az energiaellátó rendszer fejlesztése és tesztelése közben szükség van energia forrásra, ez a kész műhold esetében a napelem lesz, viszont ezek használata fejlesztés közben nem célszerű, mivel drágák és rendkívül törékenyek. Másrészről a megvilágításuk is problémákba ütközik, földi körülmények között az űrbeli napfény energiájával és spektrumával megegyező fényt nem lenne egyszerű előállítani.



3.1. ábra. SMOG-P és SMOG-1 teszteléséhez használt napelem emulátor

A 3.1. ábrán látható emulátor tartalmaz egy a napelem cella rövidzárási áramával megegyező áramú tranzisztoros áramforrást és vele párhuzamosan három sorba kötött diódát, melyeknek az együttes nyitófeszültsége a napelem üresjárási feszültségével egyezik meg.



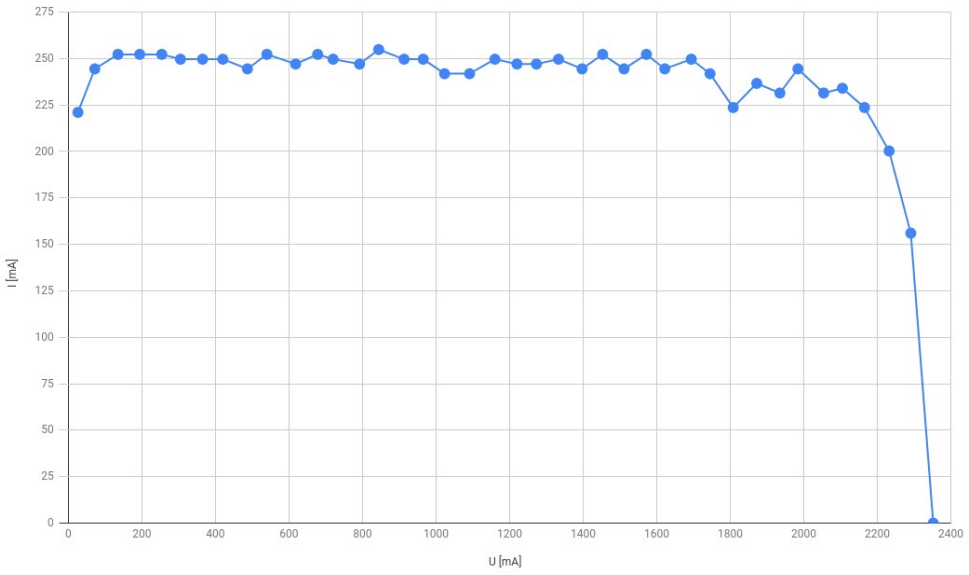
3.2. ábra. Napelem cella ekvivalens áramköri modellje [6]

A napelemcellának az emulálás szempontjából fontos paraméterei az üresjárási feszültsége: $U_{oc} = 2350$ mV, és a rövidzárási árama: $I_{sc} = 505$ mA [5]. A különböző méretű műholdak oldalaira felhelyezhető napelemtáblák paramétereit az alábbi táblázat foglalja össze:

Műhold méret	Napelem cellák elrendezése	I_{sc} [mA]	U_{oc} [mV]
1 PQ ($5 \times 5 \times 5$ cm)	1 cella félbevágva:	252,5	2350
2 PQ ($5 \times 5 \times 10$ cm)	1 teljes cella:	505	2350
3 PQ ($5 \times 5 \times 15$ cm)	2 cella párhuzamosan:	505	4700
	2 cella sorba kötve:	1010	2350

3.1. táblázat. Különböző méretű műholdak napelem elrendezései

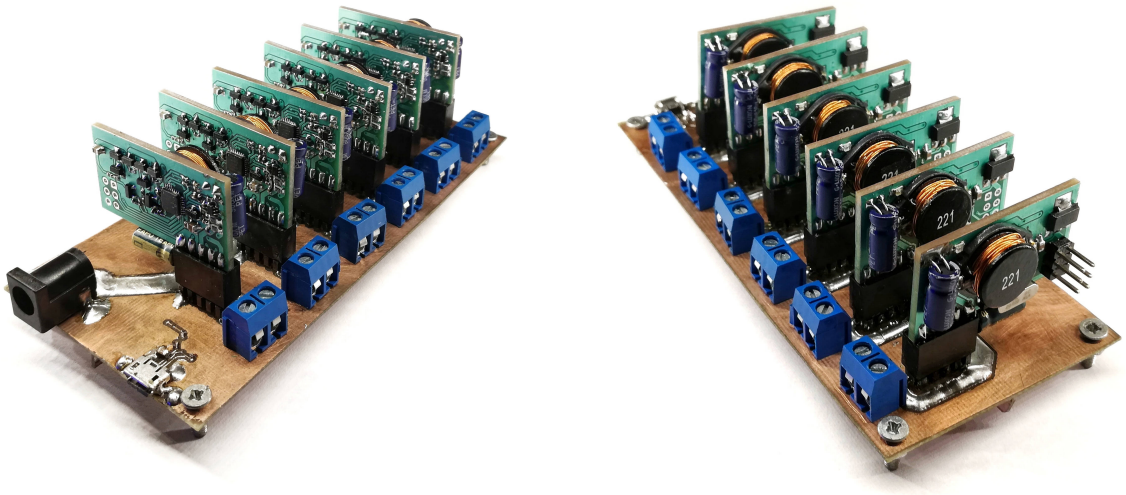
Önállólabor keretében az előző félévben ennek az emulátornak elkészítettem egy továbbfejlesztett változatát. Az általam készített áramkör egy mikrokontroller által vezérelt szinkron buck konverterből áll. Ennek köszönhetően az emulált napelem paramétereit egyszerűen lehet változtatni, így alkalmas több különböző napelem elrendezés emulálására is. Az áramkör kimeneti karakterisztikája a 2.2 ábrán látható.



3.3. ábra. Az emulátor áram-feszültség karakterisztikája

A karakterisztika méréséhez terhelésként egy nagyteljesítményű potmétert használtam, az ellenállásának a változtatás közben mértem az emulátor kimeneti áramát és feszültségét. A mérési eredményből készített karakterisztika a 3.3.ábrán látható.

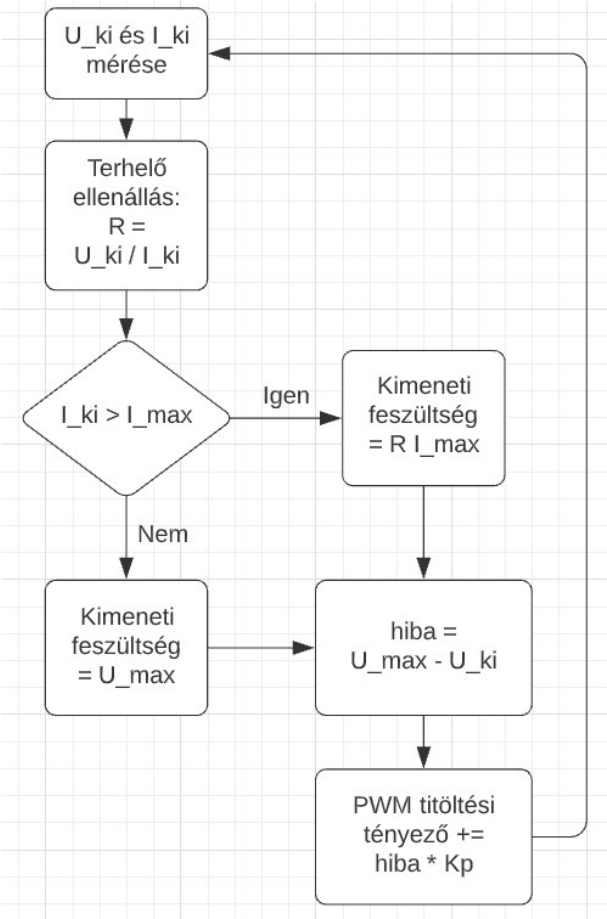
Az emulátor áramkört hat példányban készítettem el, hogy a műhold minden oldalára jusson egy, így a műhold teljes energia ellátását lehet egy időben emulálni.



3.4. ábra. Az emulátor két nézetből

Az előző félévben a hardver elkészítése után már csak arra maradt időm, hogy az áramkör működőképességét igazoló mérések elvégzéséhez szükséges szoftvert megírjam. Ahhoz, hogy az emulátor ténylegesen használható legyen, a mikrovezérlők programját tovább kellett fejlesztenem.

A feszültség szabályzó megvalósítása során először a mikrokontroller egyik időzítőjét állítottam be úgy, hogy az egy milliszekundumonként adjon egy megszakítást, ezzel időzítve az áramkör mérési- és szabályzási ciklusát. A ciklus a kimeneti áram és feszültség mérésével kezdődik, majd ezek ismeretében kiszámítható a kimeneten lévő terhelő ellenállás értéke. Amennyiben a mért kimeneti áram meghaladja a beállított áramkorlátot, az áramkörnek konstans áramú üzemmódban kell tovább működnie. Ezt úgy valósítja meg, hogy a kimeneti feszültséget csökkenti a terhelő ellenállás és az áramkorlát értékének a szorzatára. Ha az áram nem haladja meg a megengedett maximumot, akkor a kimeneti feszültség ismét az emulálni kívánt napelem maximális feszültsége lesz.



3.5. ábra. Az emulátor szabályzójának a folyamatábrája

A PWM kitöltési tényezőjének a meghatározása egy P szabályzóval történik, az elérni kívánt és a mért kimeneti feszültség különbségéből számított hibát egy Kp konstanssal szorzom meg. A konstans értékét tapasztalati úton úgy választottam meg, hogy a feszültség beállása kellően gyors és pontos is legyen.

Ezt követően az új kitöltési tényezőt módosítani kell, hogy megfelelő holtidővel elkerüljük a tranzisztorok egymásba vezetését. Ez annyit jelent, hogy az egyik tranzisztor PWM jelének kitöltési tényezőjét egy lépéssel növelem a másikét pedig eggyel csökkentem.

4. fejezet

SMOG-2 elsődleges energiaellátó rendszere

4.1. Kapcsolóüzemű tápegység topológiák

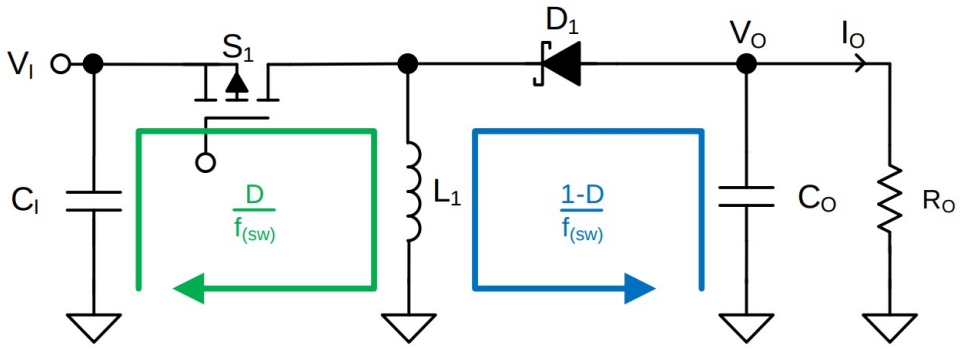
A SMOG-2 négy nagyobbik oldalán két teljes 40 × 80 mm-es napelem cella lesz elhelyezve, ezeket lehet párhuzamosan vagy sorba kötni. Előbbi esetben az áramuk, míg utóbbiban a feszültségük fog összeadódni.

Párhuzamosan kötött napelemek esetén feszültség növelő kapcsolás segítségével kell a két napelem cella maximális 2350 mV-os feszültségéből előállítani a műholdon használt li-ion akkumulátor töltéséhez szükséges legfeljebb 4,2 V-ot.

Ha a napelemeket sorba kötjük, akkor az összeadódó feszültségük akár 4,7 V is lehet, így feszültség csökkentő kapcsolásra lenne szükség, viszont megvilágítástól függően a feszültségük lehet alacsonyabb is mint az akkumulátoré, így ez az elrendezés buck-boost kapcsolás használatát teszi szükségessé.

Ezeket a paramétereket figyelembe véve döntöttem egy invertáló buck-boost kapcsolás használata mellett.

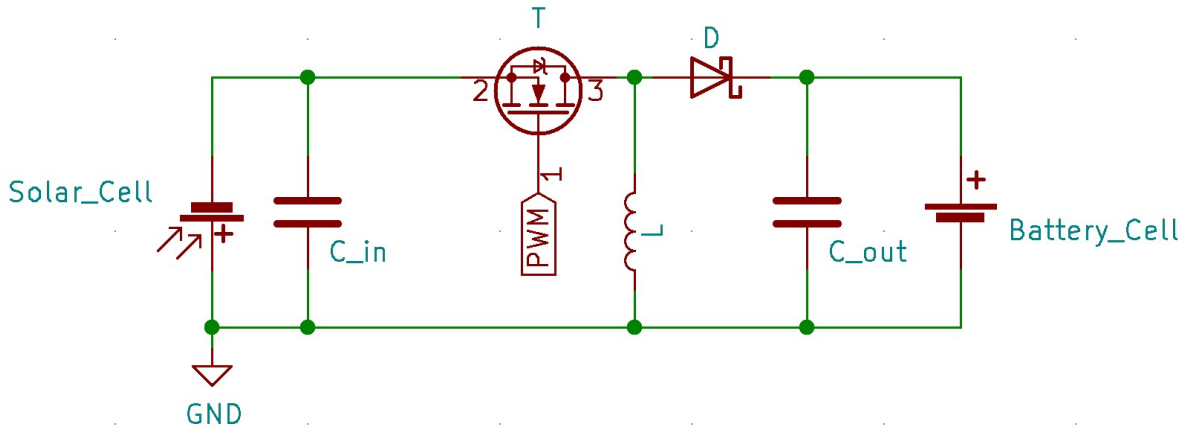
4.2. Invertáló Buck-Boost konverter



4.1. ábra. Az invertáló Buck-Boost konverter egyszerűsített rajza [15]

A megvalósítandó kapcsolóüzemű tápegységnek a feladatai közé tartozik: a napelemből kivenni az adott pillanatban kivehető maximális teljesítményt MPPT algoritmus segítsé-

gével, feszültséget invertálni, több párhuzamosan kötött li-ion akkumulátor cellát tölteni konstans áramú és konstans feszültségű üzemmódban is, az akkumulátor feszültségénél alacsonyabb és magasabb bemeneti feszültségről egyaránt. Olyan integrált áramkört, ami a fent felsorolt összes feltételnek eleget tesz nem találtam, mivel telemetria adatok gyűjtése miatt amúgy is szükséges az áramkörbe egy mikrokontroller, ezért úgy döntöttem, hogy a feszültségszabályzó áramkör vezérlését is a mikrobotrollerre bízom.

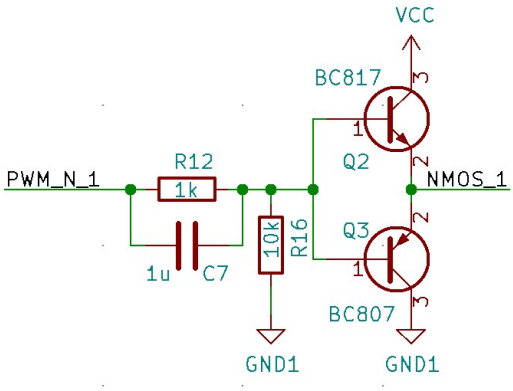


4.2. ábra. A negatív napelemfeszültségből pozitív feszültséget előállító kapcsolás

4.3. Kapcsolási rajz

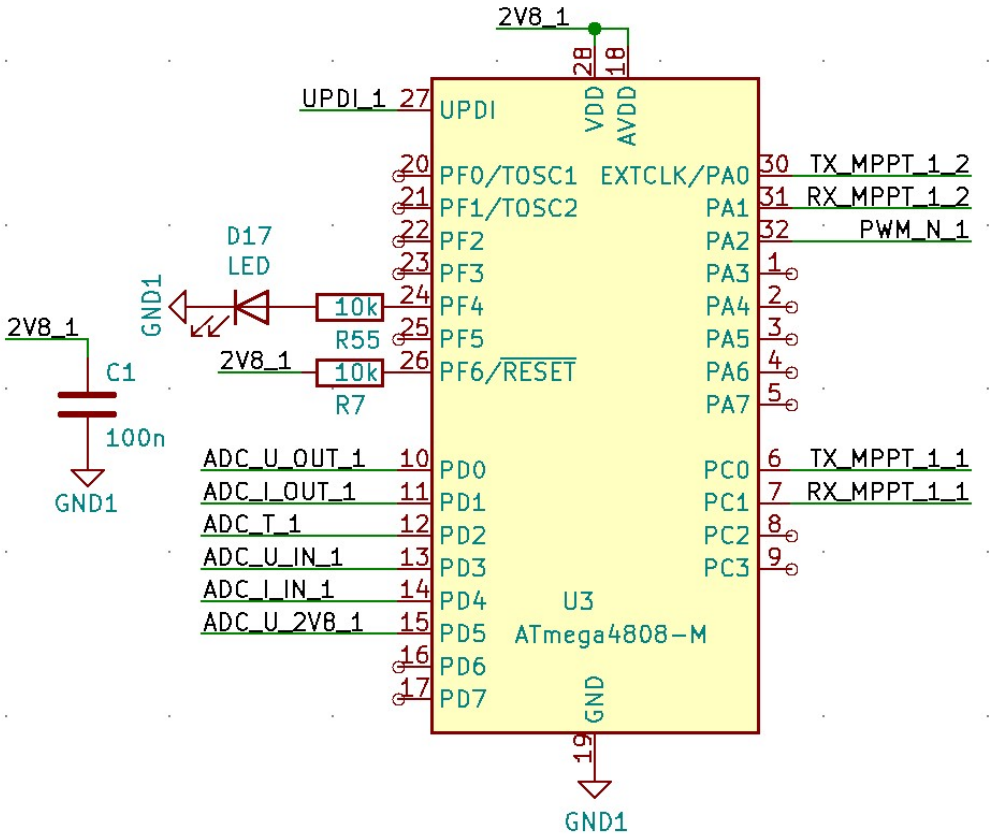
4.3.1. MOSFET vezérlése

A kapcsoló fet vezérlését egy komplementer emitter követő kapcsoláson keresztül valósítom meg, hogy a fet átkapcsolásakor jelentkező akár 60 mA-es rövid áramimpulzusok ne a mikrokontroller kimenetét terheljék.



4.3. ábra. A MOSFET meghajtó áramköre

4.3.2. Mikrokontroller



4.4. ábra. A mikrokontroller

A áramkör megépítéséhez a Microchip Atmega4808 [12] típusú mikrokontrollerét választottam. Ez alkalmas kis fogyasztású akkumulátorról történő működésre, 1,8 V és 5,5 V közötti feszültség tartományban. A 32 lábú QFN tokozás kis mérete ($5 \times 5 \times 0.85$ mm) miatt is előnyös, mivel nem sok hely áll a rendelkezésünkre. Ebben a tokozású verzióban 10 analóg feszültségek digitalizálására használható csatorna áll rendelkezésre, amelyekkel mérni tudom a be- és kimeneti áramokat illetve feszültségeket. Rendelkezik 16 bites időzítővel, ami segítségével PWM jel állítható elő. Továbbá rendelkezik két UART perifériával, ezek segítségével tud majd kommunikálni a fedélzeti számítógépekkel.

4.3.3. Áram és feszültség mérése

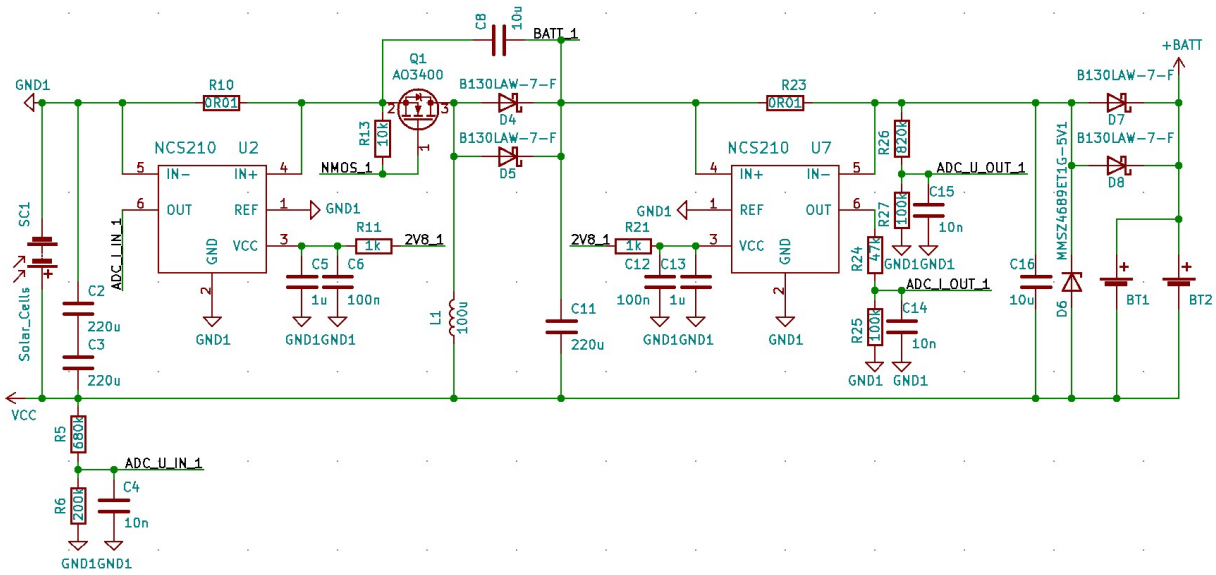
Áram mérésre egyrészt az áramkör bemenetén lévő napelemnél illetve a kimeneti akkumulátor töltő áramnál van szükség. Ezt úgy valósítottam meg, hogy egy ellenállást beiktattam az áram útjába, amin az áram hatására feszültség fog esni. Azért, hogy az ellenálláson ne legyen túl nagy a veszteség, minél kisebbet érdemes választani. Viszont ekkor a feszültségesés is nagyon kicsi lesz amit a mikrokontroller már nem tudna megmérni, ezért egy műveleti erősítőre van szükség, hogy az felerősítse az ellenálláson eső feszültséget. Ehhez egy NCS210 típusú kifejezetten áram mérésre kifejlesztett IC-t választottam, ennek a feszültségerősítése 200, így egy $0,01\ \Omega$-os ellenállást használva a napelem 505 mA-es maximális árama esetén a feszültségesés 505 mA $\times 0,01\ \Omega = 5{,}05$ mV lesz. Ezt a 200-szorosára erősítve a mikrokontroller analóg lábára 5,05 mV $\times$ 200 = 1010 mV fog

jutni, ezt a belső 1100 mV-os referenciával dolgozó analóg digitális átalakítóval már meg tudja mérni.

A be- és kimeneti feszültségeket egy-egy feszültségosztón keresztül kötöttem a mikrokontrollerre. Ezek ellenállásait úgy választottam meg, hogy az előforduló maximális feszültségek esetén is a referencia feszültségnél kicsivel kisebb legyen az átalakítóra jutó feszültség:

$$U_{be} = U_{be_{max}} \cdot \frac{R_1}{R_1 + R_2} = 4700 \text{ mV} \cdot \frac{200 \text{ k}\Omega}{200 \text{ k}\Omega + 680 \text{ k}\Omega} = 1068,18 \text{ mV}$$

$$U_{ki} = U_{ki_{max}} \cdot \frac{R_1}{R_1 + R_2} = (4700 \text{ mV} + 4200 \text{ mV}) \cdot \frac{100 \text{ k}\Omega}{100 \text{ k}\Omega + 820 \text{ k}\Omega} = 967,39 \text{ mV}$$

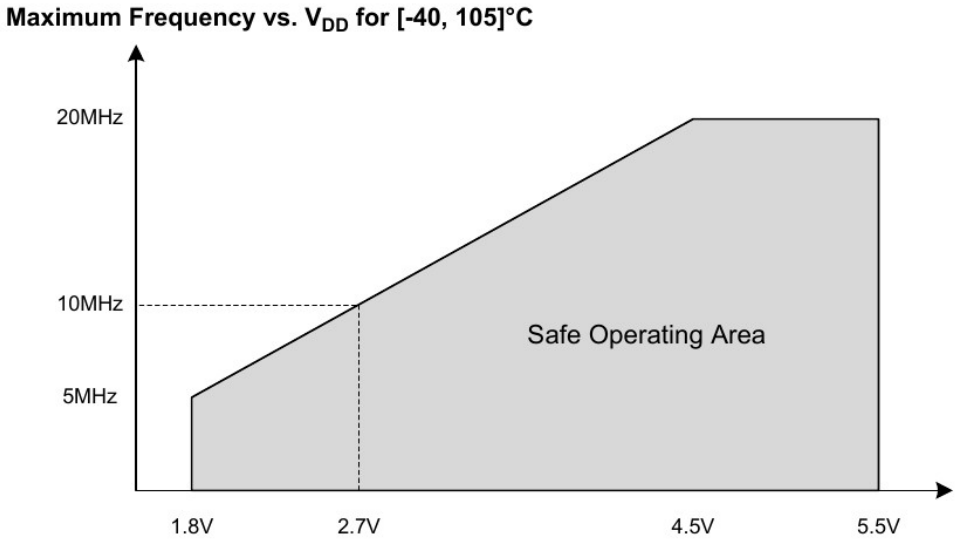


4.5. ábra. A teljes feszültségszabályzó áramkör kapcsolási rajza

Az áramkörömnek elsősorban a műhold fedélzetén lévő akkumulátorokat kell töltenie. Azonban előfordulhat az az eset is, hogy az akkumulátorokat a védőelektronikájuk leválasztotta a szabályozatlan energiabuszról. Ebben az esetben a névleges 4,2 V-os feszültséget továbbra is tartani kell, hogy a műhold működőképes maradjon, viszont így az áramot az alrendszerek aktuális fogyasztása fogja meghatározni.

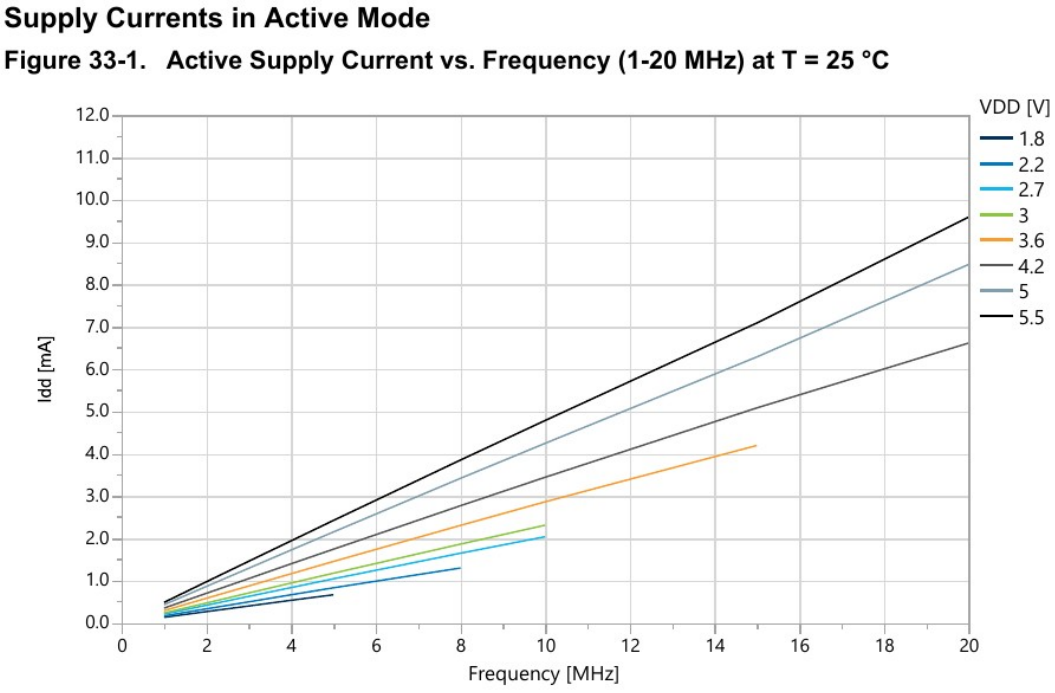
4.3.4. A mikrokontroller tápellátása

A mikrokontroller 1,8 V-tól maximum 5,5 V-ig terjedő feszültségtartományban képes működni, az adatlapjából származó 4.6. ábráról leolvasható, hogy a tápfeszültség függvényében hogyan változik a maximálisan használható órajel frekvenciája.



4.6. ábra. A mikrokontroller maximális órajel frekvenciája a feszültség függvényében [12]

Az órajelet 10 MHz-re választva, 8 bites PWM esetén az 39,0625 kHz frekvenciájú lesz, ami megfelelőnek tűnik. 10 MHz-es órajel használatához legalább 2,7 V-ra van szükség. Az akkumulátor maximális feszültsége 4,2 V lehet, amiről a mikrokontroller még tud működni, viszont a 4.7. árba alapján a fogyasztása egy adott frekvencián a tápfeszültség növekedésével együtt nő. Ennek minimalizálása érdekében a tápfeszültségét 2,7 V körül kell tartani.



4.7. ábra. A mikrokontroller áram felvétele a tápfeszültség és frekvencia függvényében [12]

A mikrokontroller két helyről kaphat energiát: a napelem felől vagy a műhold szabályozatlan energiabuszáról.

Mindkét tápforrásnál használok áramkorlátozó ellenállásokat. Erre akkor van szükség, ha az áramköröm meghibásodik és a normál működés közben elvártnál jelentősen nagyobb

lesz az áram felvétele.

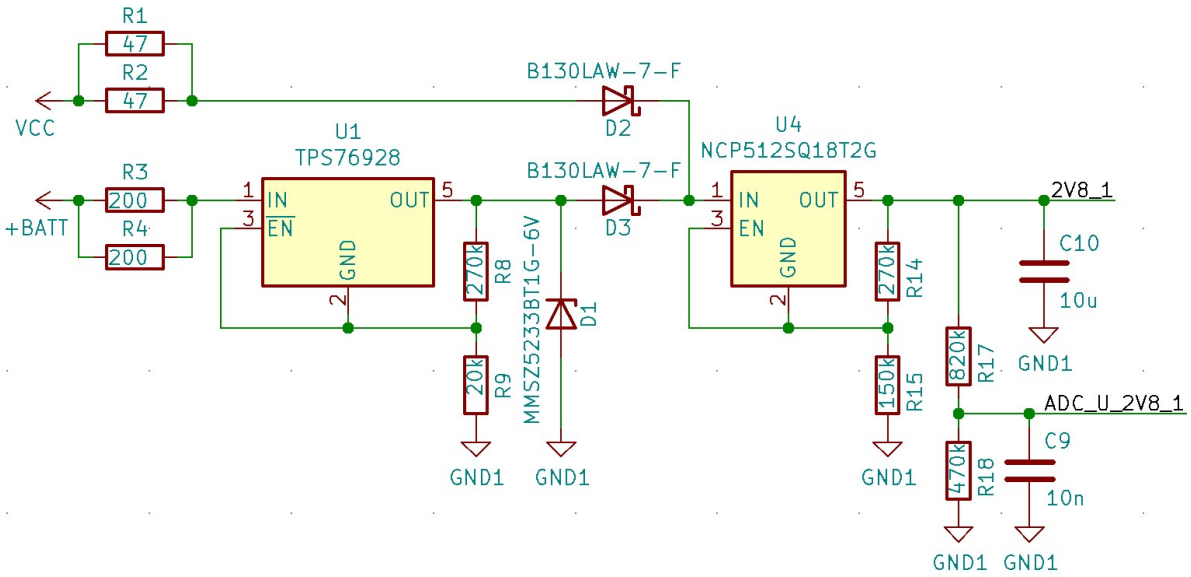
Az akkumulátor felőli korlátozó ellenállásokat úgy méreteztem, hogy rövidzár esetén, teljesen feltöltött 4,2 V-os akkumulátor feszültség hatására is csak 4,2 V/100 Ω = 42 mA áram fog folyni. Ez jelentős veszteség lenne, de a műhold továbbra is működőképes marad. Az ellenállások további növelésével ez az áram csökkenthető lenne, viszont normál működés közben a rajtuk keletkező veszteség nagyobb lenne.

A napelem korlátozó ellenállásait választhattam kisebbre, hiszen egyedül az áramkör használja a hozzá tartozó napelemet. Ha az áramkör meghibásodik, a napelemre az akkumulátorral ellentétben más alrendszernek nem lesz szüksége. Viszont tetszőlegesen kicsire se választhatom az ellenállások értékét, hiszen a rajtuk folyó nagy áram jelentős teljesítmény disszipációhoz vezetne. A ellenállásokat 23,5 Ω-ra választva maximális 4,7 V-os napelem feszültség esetén 200 mA áram folyna, viszont az utánuk lévő áramkorlátozással rendelkező LDO ezt 80 mA-nél nem engedi, hogy több legyen.

A tápfeszültséget egy NCP512 típusú LDO segítségével állítom elő. Az LDO választásakor fontos szempont volt, hogy minél kisebb a dropout feszültsége, ez ennek a típusnál 10 mA-es terhelés esetén 25 mV körül van. Másik fontos tulajdonsága, hogy már 1 V-os bemeneti feszültségtől a benne található P csatornás MOSFET egybenyit. Ennek köszönhetően már a névleges 2,8 V-os feszültség alatt is működhet a mikrokontroller, ami alacsony napelemfeszültség esetén előnyös lehet.

Ennek az LDO sorozatnak létezik 2,8 V-os fix kimenetű változata, viszont annak, vagy más hasonló paraméterekkel rendelkező helyettesítő IC-nek a beszerzése az áramkör fejlesztésekor nem volt megoldható belátható időn belül. Ezért az 1,8 V-os verzió használata mellett döntöttem, ennek a kimenetét a GND lábára vissza csatolva oldottam meg a 2,8 V-os feszültség előállítását:

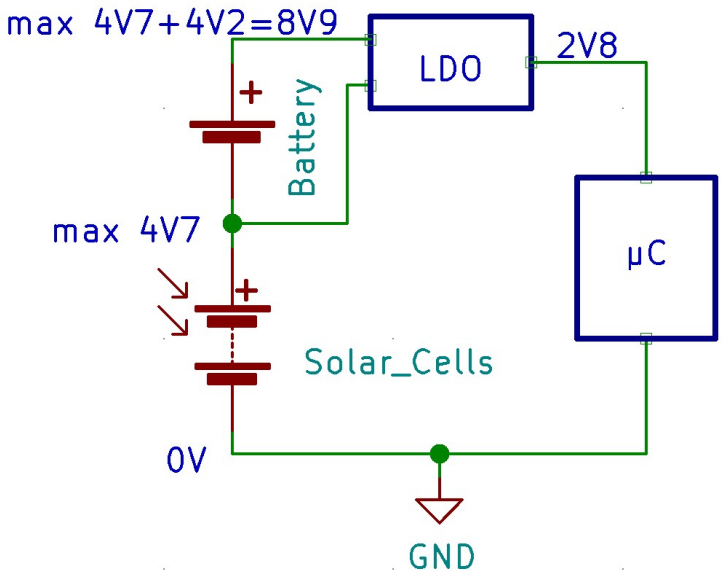
$$U_{out} = U_{ref} \times (1 + R1/R2) = 1,8\ \text{V} \times (1 + 470\ \text{k}\Omega/820\ \text{k}\Omega) = 2,832\ \text{V}$$



4.8. ábra. A mikrokontroller feszültségszabályzó áramköre

Az feszültségfordító kapcsolásból adódóan az akkumulátor feszültsége az áramkörök földpontjához képest a napelem feszültségével magasabban van. Ez a megvilágítástól függően akár 4,2 V + 4,7 V = 8,9 V is lehet. Ez azért okoz problémát mert az NCP512

maximum 6 V-ot visel el a bemenetén. Így az akkumulátorról való működéshez még egy LDO-ra van szükség, ami nagyobb bemeneti feszültségről is tud működni. Erre a célra a megfelelő paraméterekkel rendelkező TPS76928 típusút választottam. A döntést ebben az esetben is jelentősen befolyásolta a beszerezhetőség.



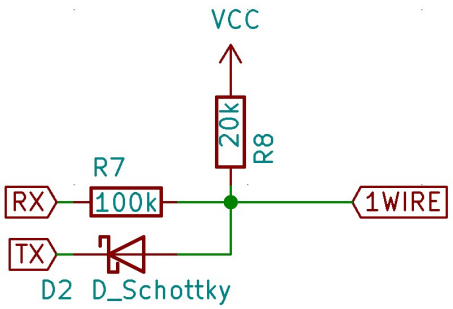
4.9. ábra. A mikrokontroller tápellátásának blokkvázlata

4.3.5. Telemetria adatok gyűjtése

A feszültségszabályzó vezérlése mellett a mikrokontrollernek feladata telemetria adatok gyűjtése is, amelyeket a fedélzeti számítógépnek (OBC) továbbít. A szabályzó működéséhez szükséges mérni a be- és kimeneti feszültségeket és áramokat, így ezek az adatok már rendelkezésre állnak. Ezekből számítható az átalakító hatásfoka is.

Ezek mellett még hasznos telemetria adat a napelemoldalak hőmérséklete, ezért minden oldalpanel belső felén el lesz helyezve egy hőmérséklet érzékelő szenzor.

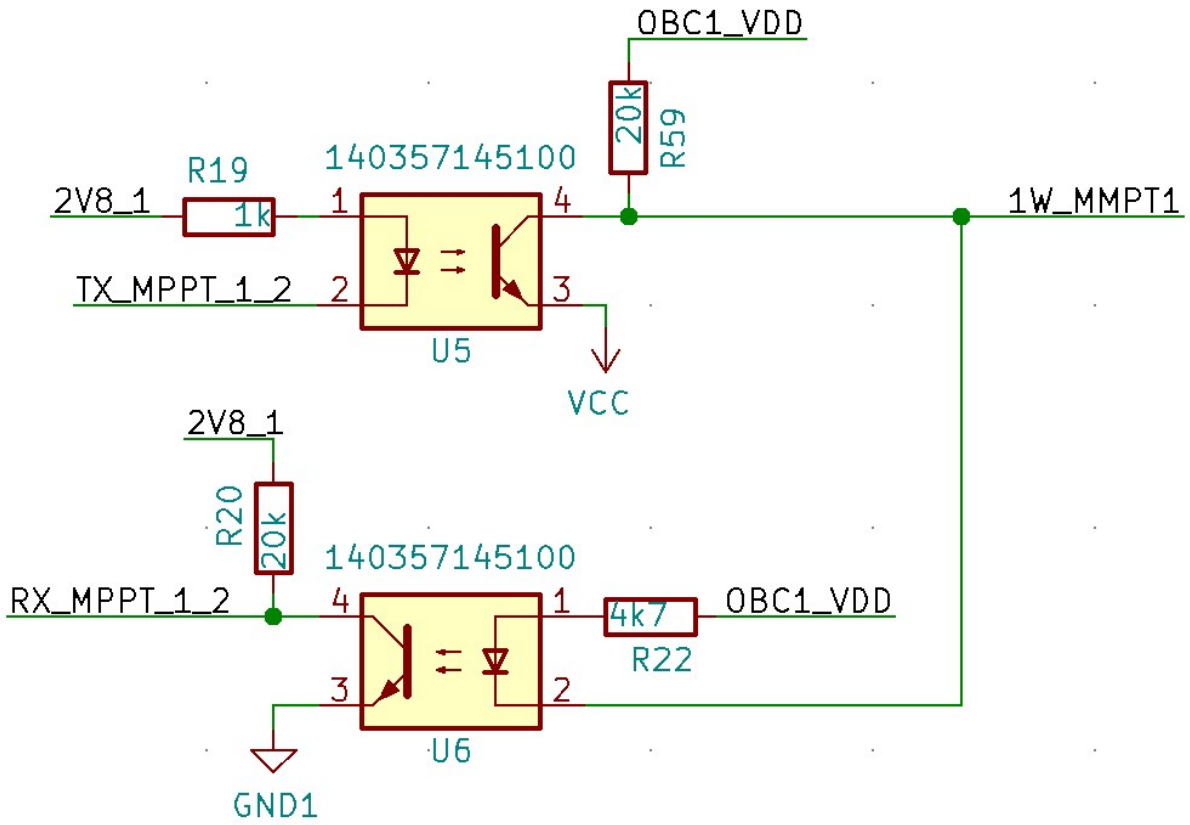
A fedélzeti számítógéppel az alrendszerek fél-duplex UART segítségével kommunikálnak, ezzel megoldható a kétirányú kommunikáció egy vezetéken.



4.10. ábra. A fél-duplex UART-ot megvalósító áramkör

Mivel az én áramköröm és a műhold többi alrendszere különböző feszültség szinten van, a köztük lévő kommunikáció nem oldható meg egyszerűen egy vezetékes kapcsolattal. Erre megoldásként optocsatolók használata mellett döntöttem. A négy áramköröm egy közös

buszon fog kommunikálni az OBC-vel, ami az egyik oldalhoz tartozó áramkört megcímezve küld egy kérést, amire az adott mikrokontroller válaszként elküldi a telemetria adatokat.



4.11. ábra. Az OBC-vel való összeköttetést megvalósító optocsatolók

4.4. Prototípus nyomtatott áramkör

4.4.1. Panelek

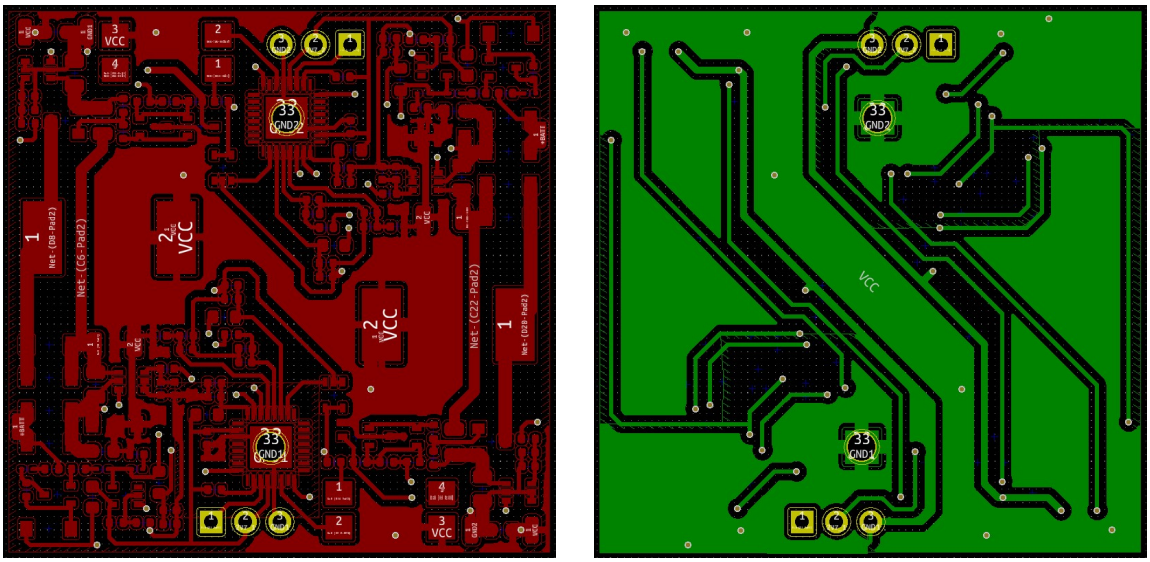
A műhold belsejében található paneleken elhelyezkedő áramkörök számára használható felület nagyjából 40 × 40 mm-es lesz, ekkora méretű paneleken kell elférnie minden alrendszernek. Mivel a műhold négy oldalán lesz napelem, a feszültségszabályzó áramkörömből is négy darabra lesz szükség. Az előzőekhez képest nagyobb méretű műholdban több panelnek van hely, viszont a fedélzetre kerülő kísérletek száma miatt célszerű spórolni a panelek számával ahol lehetséges, ezért a négy áramkörömet maximum két panelen szeretném elhelyezni.

4.4.2. A nyomtatott áramkör tervezése és elkészítése

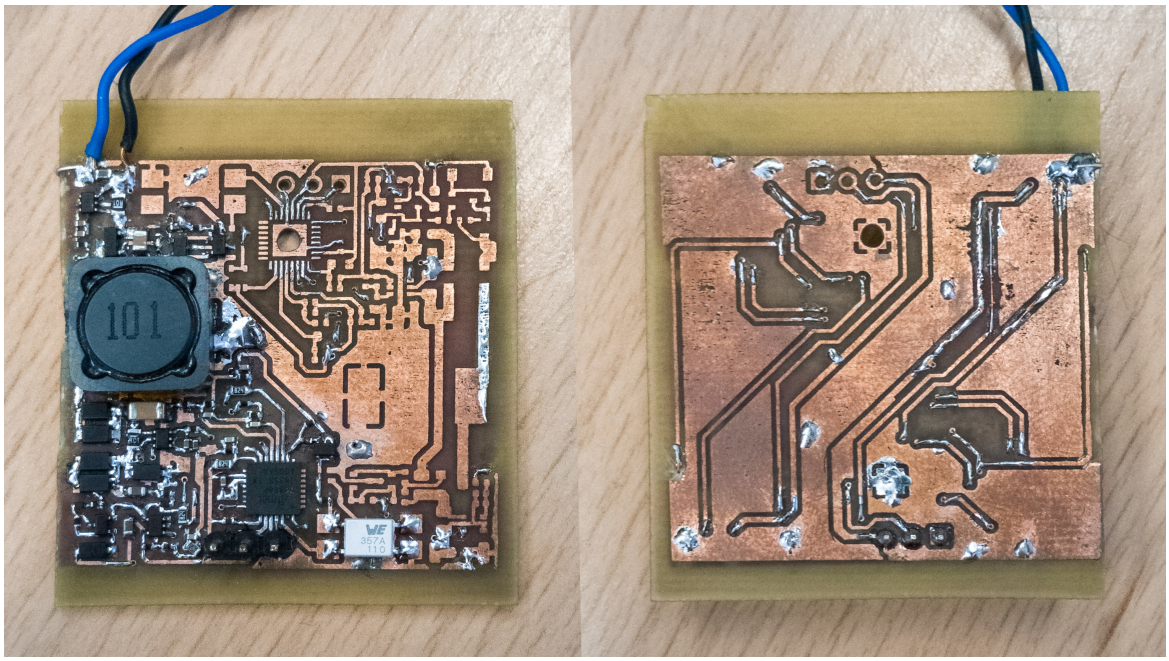
A tervezés során minden áramkört és kapcsolási rajzot a nyílt forráskódú KiCAD tervezőprogramban készítettem. [10]

A prototípus áramkör tervezésekor megpróbáltam azt a véglegeshez minél jobban közelítő méretűre elkészíteni. A gyártatott nyákon négy vezető rétegen tudok majd dolgozni, a

házilag készített prototípusnál csak kettőn, viszont ennek ellenére is sikerült az áramkört két példányban 40 × 40 mm-es hordozó felületen elhelyezni.



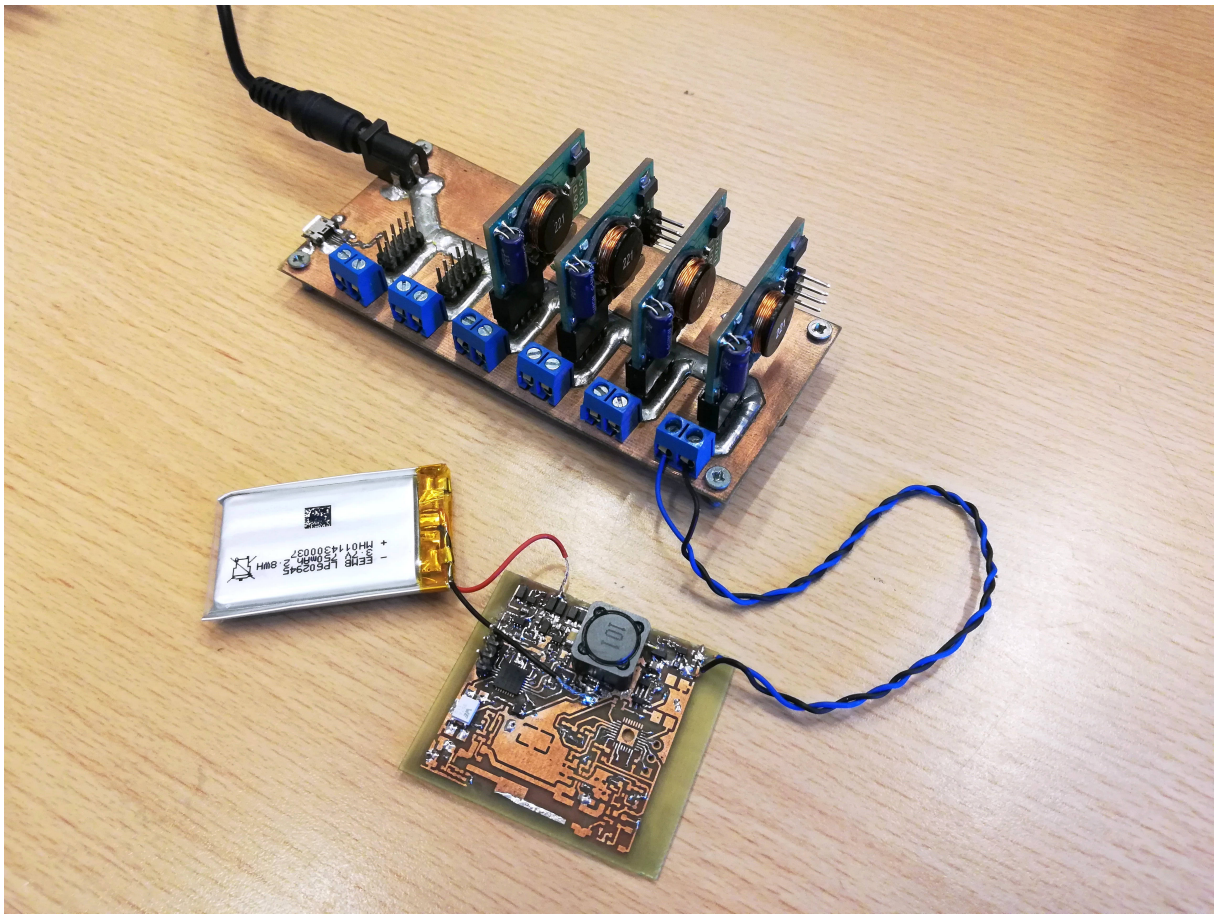
4.12. ábra. A nyomtatott áramkör tervének két oldala



4.13. ábra. Az elkészült áramkör két oldala

Az áramkör felélesztése során először a panelen lévő két feszültség szabályzó áramkörből csak az egyiket ültettem be, a működőképességének teszteléséhez ez is elég. A több áramkör együttes működését a későbbiekben fogom majd vizsgálni.

4.5. Az áramkör mérése



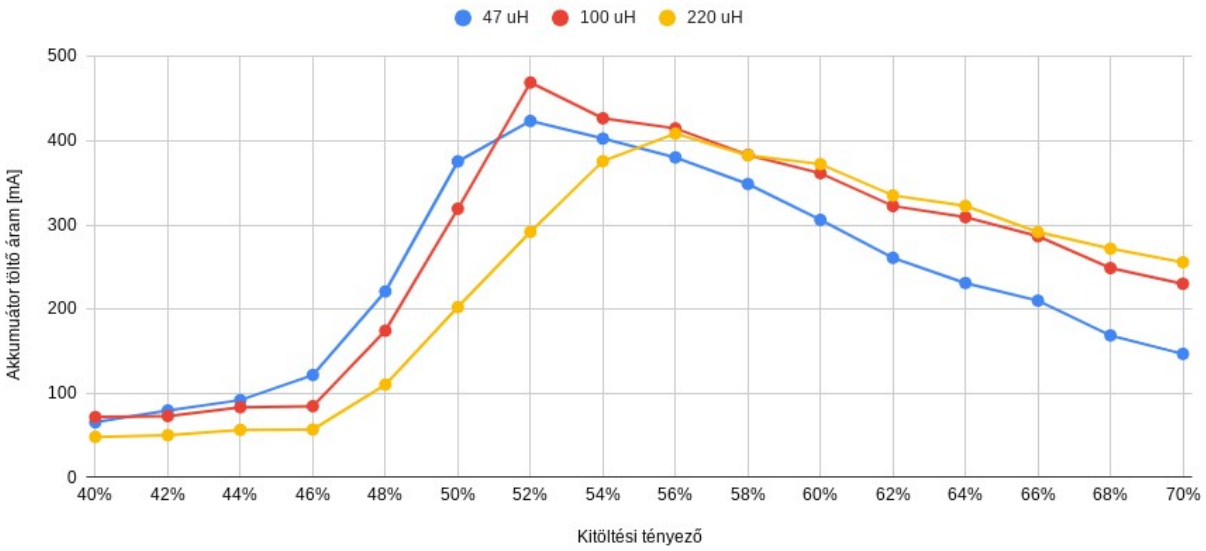
4.14. ábra. Műhold akkumulátor töltése napelem emulátorról

A következő mérések elvégzéséhez egy, a műholdjainkban is használt típusú akkumulátor paramétereivel megegyező akkumulátor cellát kötöttem az áramkör kimenetére. A napelem emulátort két sorba kötött napelemcellának megfelelő paraméterekre állítottam be, a feszültségét 4700 mV-ra, maximális áramát pedig 505 mA-re.

4.5.1. Induktivitás és PWM frekvencia

Először különböző értékű tekercsekkel vizsgáltam az áramkör működését, a PWM kitöltési tényezőjét 50% körül változtatva mértem az akkumulátor töltő áramát.

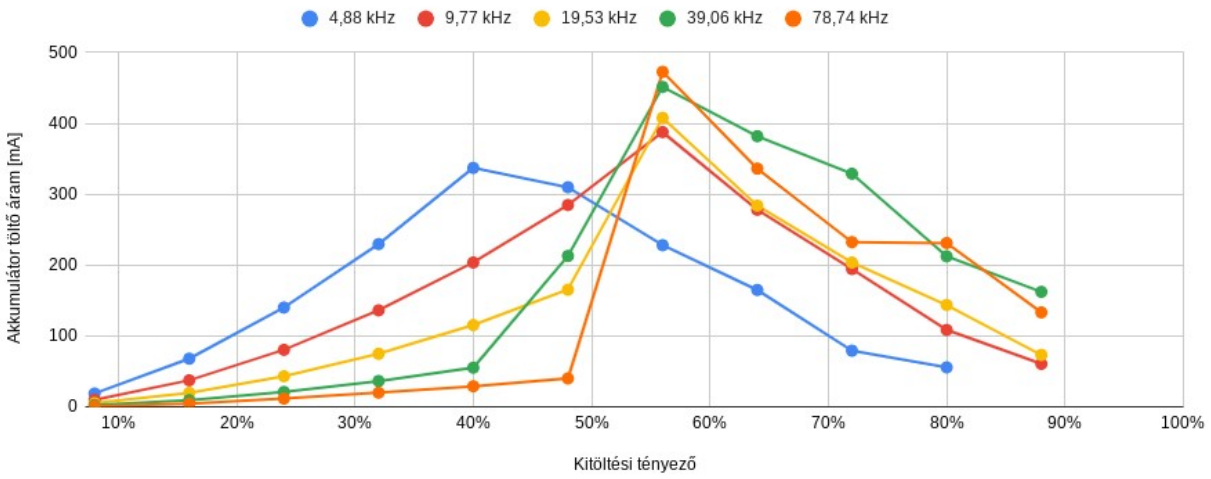
A méréseket egy HP 34401A típusú multiméterrel végeztem.



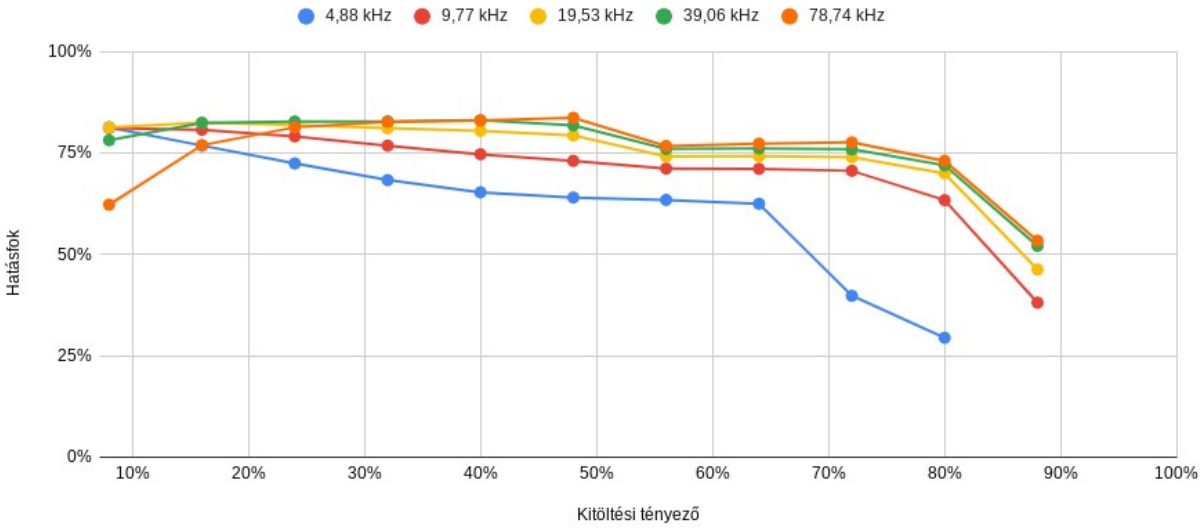
4.15. ábra. Maximális akkumulátor töltő áram különböző tekercsekkel

A mérések alapján a 100 μH induktivitású tekercs használatakor volt a töltőáram maximuma a legnagyobb, ezért ezt választottam.

A tekercs kiválasztása után a PWM frekvenciájának a töltőáram maximumára gyakorolt hatását vizsgáltam. Az alábbi ábrákon látható az akkumulátor árama, illetve az áramkör hatásfoka különböző frekvenciákon.

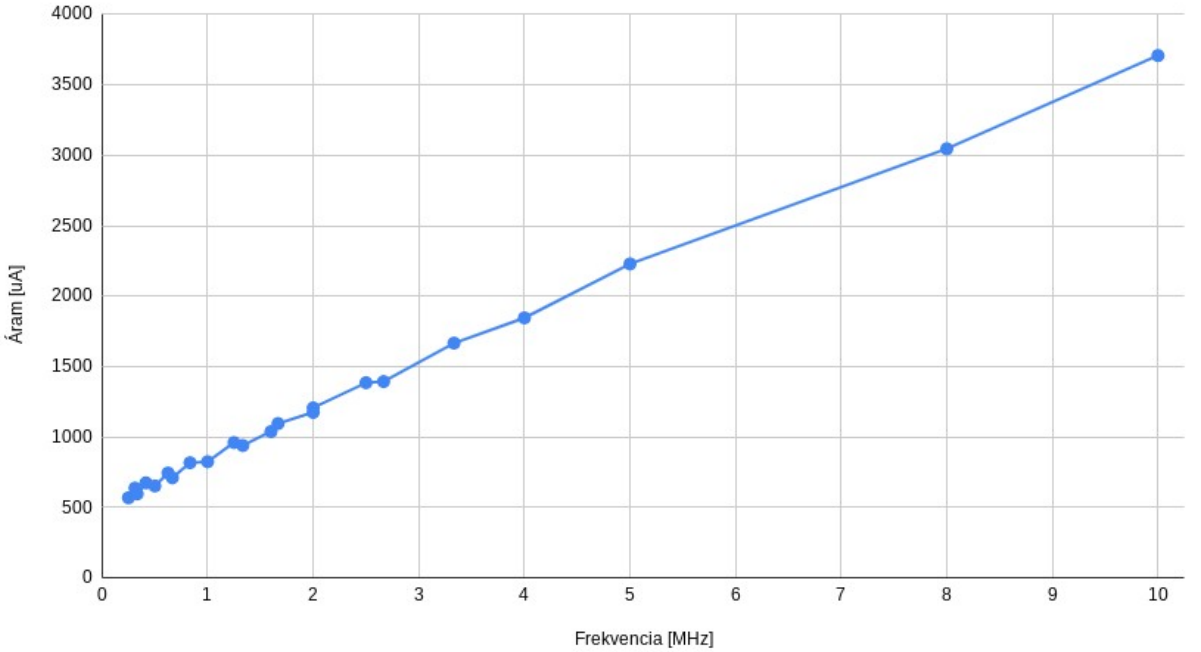


4.16. ábra. Az akkumulátor töltő árama különböző PWM frekvenciákon



4.17. ábra. Az áramkör hatásfoka különböző PWM frekvenciákon

A mérési eredményekből látható, hogy a töltőáram maximuma a frekvencia növelésével együtt nő. Nagyobb PWM frekvencia előállításához a mikrokontrollernek is nagyobb órajel frekvenciára van szüksége ami az áramfelvételét is megnövelné, erről látható egy mérési eredmény a 4.14. ábrán. Mivel a 78,74 kHz és a 39,06 kHz-es frekvencia között nincs túl nagy különbség, így a legnagyobb használható PWM frekvenciának a 39,06 kHz-et választottam.



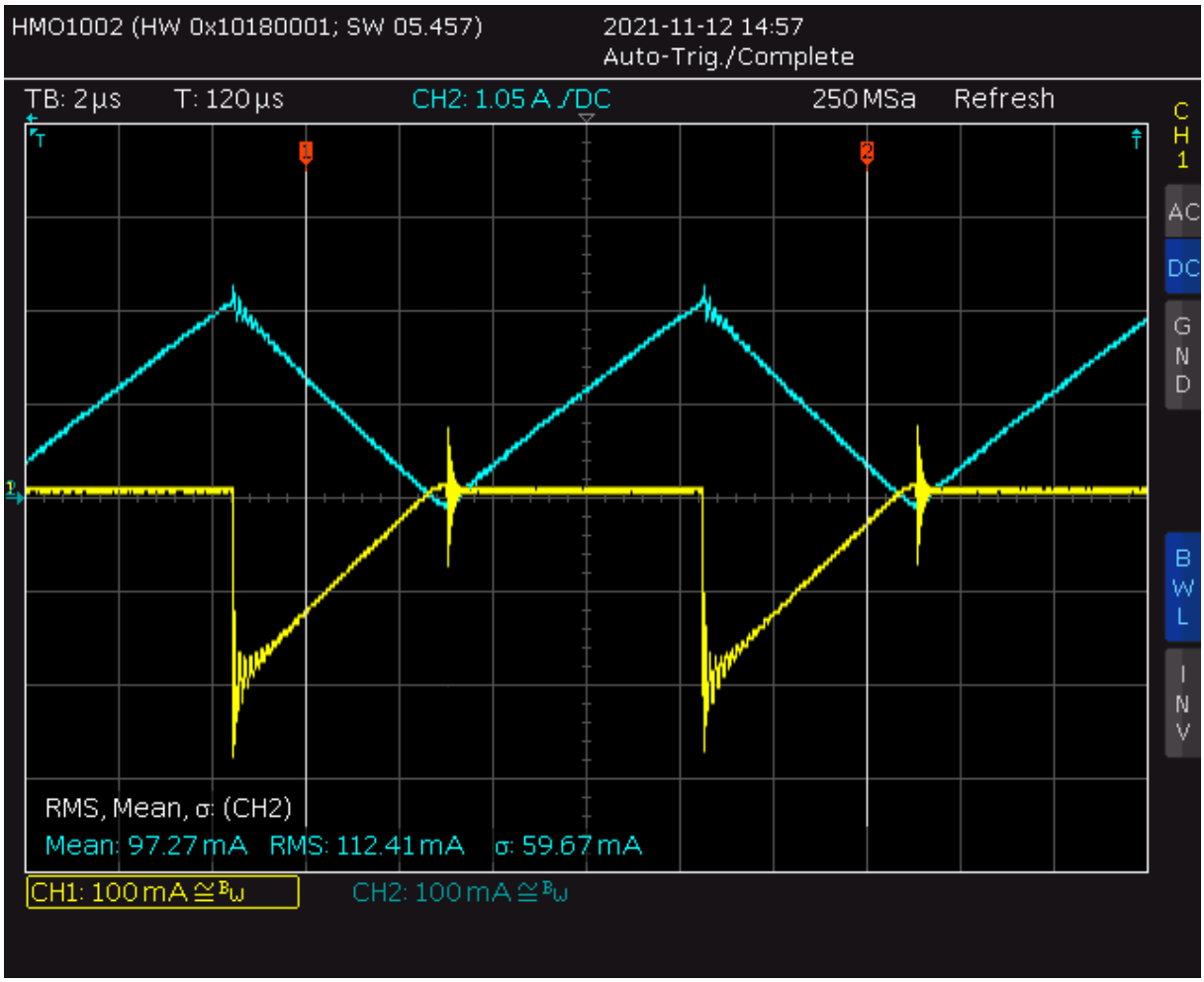
4.18. ábra. A mikrokontroller fogyasztása az órajel frekvenciája függvényében

Az általam mért értékek a 4.7. ábrán látható, a mikrokontroller adatlapjából származó értékeknél valamivel magasabbak, ezt a mikrokontrollerben bekapcsolt perifériák (ADC, időzítő) többlet fogyasztása okozza.

4.5.2. Tekercs és dióda árama

A fejlesztés során felmerült az ötlet, hogy az áramkörben a diódát egy MOSFET-tel helyettesítve az áramkör hatásfoka javítható lenne, hiszen a dióda vesztesége jelentősen korlátozza azt.

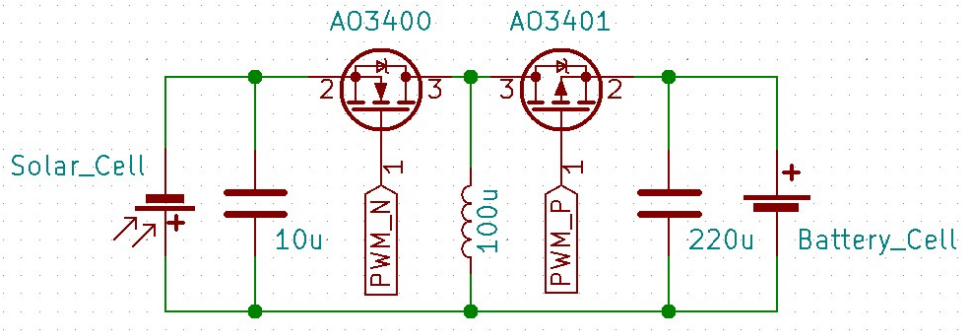
Az alábbi ábrán látható mérés során a tekercsen folyó áramot vizsgáltam. Az oszcilloszkóp egyes (sárga) csatornáján a tekercs, a kettes (kék) csatornáján pedig a dióda árama látható.



4.19. ábra. A tekercs és a dióda árama

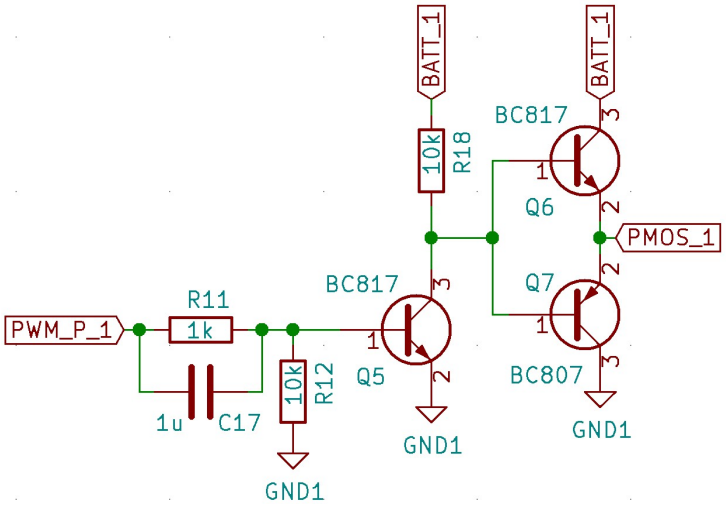
A két MOSFET-es megoldás megvalósításához nagyon precíz vezérlésre lenne szükség, ahogy az a 4.19. ábrán is látható, amint a dióda árama eléri a nullát az megszűnik vezetni. Ha ez nem így történne, akkor az áram elkezdene rajta az ellenkező irányba folyni, ami az áramkör működése szempontjából nagyon hátrányos lenne.

Nagy kimeneti áram esetén a tekercs árama folytonos marad, a nullát nem éri el. Ebben az állapotban a diódát helyettesítő MOSFET vezérlése is egyszerűbb. A dióda helyettesítésére egy P csatornás MOSFET-et választottam. Ennek a source lába a 4.20. ábrán látható módon az áramkör kimenetén van.



4.20. ábra. A feszültségfordító kapcsolás két MOSFET-tel

Ahhoz, hogy a mikrokontroller tudja vezérelni ezt a tranzisztort, az alábbi ábrán látható kapcsolásra van szükség:



4.21. ábra. A P csatornás MOSFET meghajtó áramköre

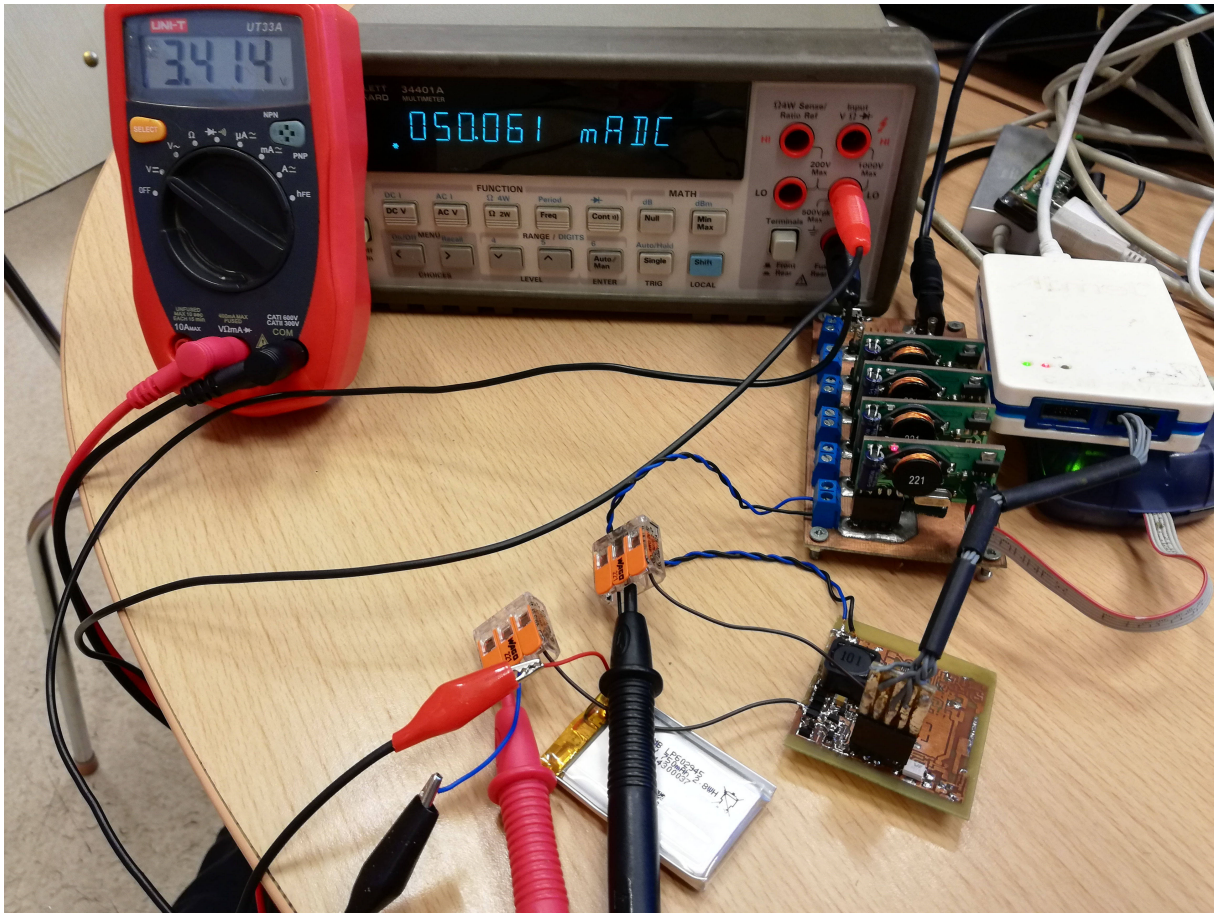
Mivel ez a kapcsolás a vezérlő PWM jelet invertálja, a megfelelő vezérléshez olyan PWM jelre van szükség, aminek akkor van felfutó éle amikor az N csatornás tranzisztort vezérlő jelnek lefutó éle van. Ezeket a jeleket a mikrokontroller egyik időzítőjével és a hozzá tartozó PWM periféria segítségével állítom elő. Ez önmagában nem képes a megfelelő jel előállítására, viszont a mikrokontroller kimeneti regisztere tudja invertálni a P csatornás MOSFET-hez tartozó PWM jelet, így a megfelelő jelalakot elő lehet állítani.

Egy ilyen áramkör elkészítése után különböző áramok mellett mértem a hatásfokot, egyszer a MOSFET-et vezérelve, majd azt nem vezérelve csak a diódát használva. A legnagyobb elérhető áram mellett közel 5%-os hatásfok javulást sikerült elérni, viszont kisebb áram érték esetén nagyobb volt a P csatornás MOSFET-et vezérlő áramkör vesztesége, mint amennyit a dióda helyettesítésével meg lehet spórolni. Mivel ez a megoldás nem hozott egyértelműen javulást, viszont a plusz áramköri részek még tovább bonyolították az teljes áramkört, inkább a diódás megoldás használata mellett döntöttem.

4.6. Akkumulátor töltés

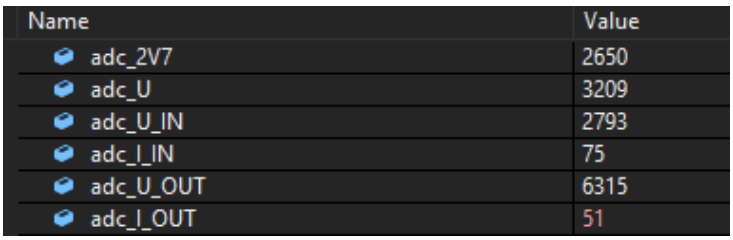
A műhold alrendszerei a napelemekből beérkező energia egy részét rögtön felhasználják, a többit egy lítium-ion akkumulátorban tárolja a rendszer, hogy akkor is tudjon majd működni, amikor földárnyékban nincs napfény. Az akkumulátor töltése az MPPT áramkör feladata. Ezt úgy kell megvalósítani, hogy amíg az akkumulátor nem éri el a maximális 4,2 V-os feszültségét, addig áramgenerátorként, amennyiben ezt a bejövő energia mennyisége lehetővé teszi maximális árammal kell tölteni. Ha elérte a maximálisan megengedett feszültséget, akkor már korlátozni kell a töltőáramkör feszültségét. A műhold négy napelemoldaláról érkező energiát kis veszteségű schottky diódák segítségével lehet összegezni. Ezekre a visszáram elleni védelem miatt van szükség.

Ebben a mérésben az áramkör hatásfokát vizsgáltam, ehhez a napelem emulátor kimeneti feszültségét 3,5 V-ra, az áramkorlátot 80 mA-re állítottam. A 4.22. ábrán látható multiméterekről az akkumulátor feszültsége és töltőárama olvasható le.



4.22. ábra. Az akkumulátor töltéshez használt mérési elrendezés

A 4.23. és a 4.24. ábrákon a mikrokontroller által mért áram és feszültség értékek láthatóak, az áramok mA-ben a feszültségek mV-ban.



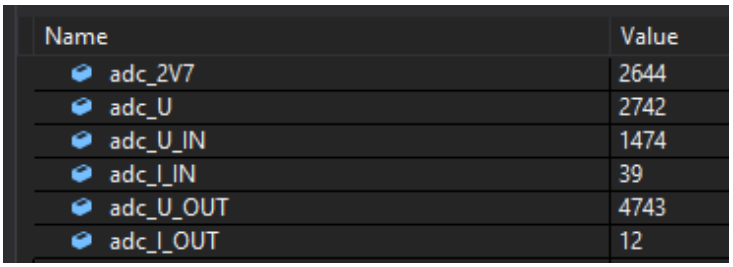
Name	Value
adc_2V7	2650
adc_U	3209
adc_U_IN	2793
adc_I_IN	75
adc_U_OUT	6315
adc_I_OUT	51

4.23. ábra. A mikrokontroller által mért áram és feszültség értékek

A mikrokontroller minden feszültséget a saját földjéhez képest mér, ezért az általa mért kimeneti feszültség az a napelem és az akkumulátor felszültségének az összege lesz. Ebből és a bemeneti napelemfeszültségből kiszámítható az akkumulátor feszültsége, ez jelen esetben 6315 mV - 2793 mV = 3522 mV-ra adódik. Az ábrán látható „adc_U" és „adc_2V7" értékek a saját tápfeszültsége a 2,8 V-os feszültségstabilizátor előtt illetve után.

A mért értékek alapján kiszámítható az áramkör hatásfoka:
$(U_{out} \cdot I_{out})/(U_{in} \cdot I_{in}) = (3522 \text{ mV} \cdot 51 \text{ mA})/(2793 \text{ mV} \cdot 75 \text{ mA}) = 85,75\%$

Ugyanezt a mérést megismételtem 1,5 V-os napelem feszültség esetén is, az alábbi eredményeket kaptam:



Name	Value
adc_2V7	2644
adc_U	2742
adc_U_IN	1474
adc_I_IN	39
adc_U_OUT	4743
adc_I_OUT	12

4.24. ábra. A mikrokontroller által mért áram és feszültség értékek

A mért értékek alapján az áramkör hatásfoka:
$(U_{out} \cdot I_{out})/(U_{in} \cdot I_{in}) = (3269 \text{ mV} \cdot 12 \text{ mA})/(1474 \text{ mV} \cdot 39 \text{ mA}) = 68,24\%$

A mérések alapján elmondható, hogy magasabb napelemfeszültségek esetén a hatásfok is magasabb lesz, viszont amikor az akkumulátor feszültségénél alacsonyabb napelemfeszültségről kell a töltést megvalósítani, a hatásfok el kezd csökkenni.

5. fejezet

Maximális teljesítményű munkapont követés

A napelemekből kivehető teljesítményt befolyásolja, hogy a cellák mekkora terheléssel vannak lezárva, a megvilágítás szögétől és a hőmérséklettől. Ahhoz, hogy a maximális kivehető teljesítményhez hozzájussunk, a napelemeket a maximális teljesítményű munkapontjukban kell működtetni. Ez a munkapont a fent felsorolt paraméterek függvényében változik. A napelemekből kivehető teljesítmény maximalizálásához ezt a változó munkapontot követni kell. Erre ad megoldást a maximális munkapont követő algoritmus (MPPT).

5.1. MPPT algoritmusok

5.1.1. Empirikus alapon működő stratégiák

Amennyiben lassan változnak a környezeti paraméterek, lehet alkalmazni olyan MPPT eljárásokat, melyek arra épülnek, hogy az maximális munkaponti feszültség lineáris összefüggésben van az üresjárási feszültséggel, illetve a maximális munkaponti áram a rövidzárási árammal. Ezt a két mérhető értéket egy-egy meghatározott konstans értékkel szorozva megkapható a keresett munkaponthoz tartozó áram és feszültség értéke. Az algoritmus adott időközönként megvizsgálja ezeket a paramétereket, majd egy a megfelelő munkapontba állítja a napelem terhelését. Ennek az eljárásnak előnye, hogy megvalósítása nagyon egyszerű, még digitális jelfeldolgozás sem szükséges hozzá. Viszont egyszerűsége mellett több problémája is van. Egyrészt az üresjárási feszültség megméréséhez a napelemet le kell választani a műhold tápellátó rendszeréről, így a mérés ideje alatt nem történik teljesítmény felvétel a napelemből. Másik probléma, hogy ha a mintavétel csak ritkán történik a napelem kihasználatlanságának minimalizálása érdekében, akkor a műhold forgásából adódóan folyamatosan változó megvilágítást az algoritmus nem fogja tudni helyesen követni. [11]

5.1.2. Hegymászó stratégiák

Perturb & Observe

Az úgynevezett hegymászó stratégiák az egyik legelterjedtebben használt maximális munkapont követő algoritmusok. A működésének a lényege, hogy az algoritmus beállít

egy munkapontot, majd a terhelést változtatva vizsgálja, hogy a felvett teljesítmény hogyan változik. A változás előjele határozza meg, hogy a következő lépésben a terhelés változtatása milyen irányba történjen. Ezt a folyamatot addig végzi az algoritmus, amíg meg nem találja a teljesítmény első lokális maximumát. Ezután az megtalált munkapont körül oszcillál.

Ennek az algoritmusnak sem túl bonyolult a megvalósítása, a kapcsolóüzemű tápegységet vezérlő PWM jel kitöltési tényezőjét kell perturbálni. A napelem árama és feszültsége egy analóg digitális átalakító segítségével mérhető, így a kivett teljesítmény is kiszámolható. Ennek az algoritmusnak is vannak hátrányai.

Egyik legnagyobb probléma vele, hogy az első lokális maximumnál megáll és ott is marad, amennyiben a munkapont körüli oszcilláció során nem talál egy másik, az előzőnél nagyobb teljesítményű munkapontot. Erre megoldást adhat, hogy időnként a kitöltési tényezőt nagyobb mértékben változtatjuk, annak reményében, hogy a már megtalált lokális maximumból kimozdulva megtaláljuk az abszolút maximumot. Ez történhet akár tetszőleges munkapont áthelyezéssel, vagy a kitöltési tényező közel teljes tartományán történő végig pásztázással is. [11]

Incremental conductance

Az incremental conductance algoritmus azt használja ki, hogy a teljesítménygörbe meredeksége a maximális teljesítményű munkapontban zérus. Két egymást követő minta alapján így egyszerűen eldönthető, hogy a kitöltési tényező változtatása milyen irányba történjen.

Mindkét hegymászó algoritmus esetében a maximális munkapontba történő beállás ideje az ugrások mértékétől függ. Nagyobb lépések használatával a keresett munkapont hamarabb megtalálható, viszont így az arra való beállás pontossága csökken. Kisebb lépések használata esetén pontosabb lesz a beállás, viszont az hosszabb ideig fog tartani. Problémát okoz még az is, hogy ha a megvilágítás folyamatosan változik, akkor maximális munkapont is jelentősen arrébb kerülhet és ennek következtében könnyű elveszíteni a maximális munkapont helyét. Továbbá a munkapont körüli oszcilláció csökkenti a kivehető teljesítményt, mivel nem pontosan a maximális munkapontban működik a napelem. Erre a kompromisszumra viszont szükség van annak érdekében, hogy a környezeti változások hatását követni tudja az algoritmus. [11]

Egyéb stratégiák

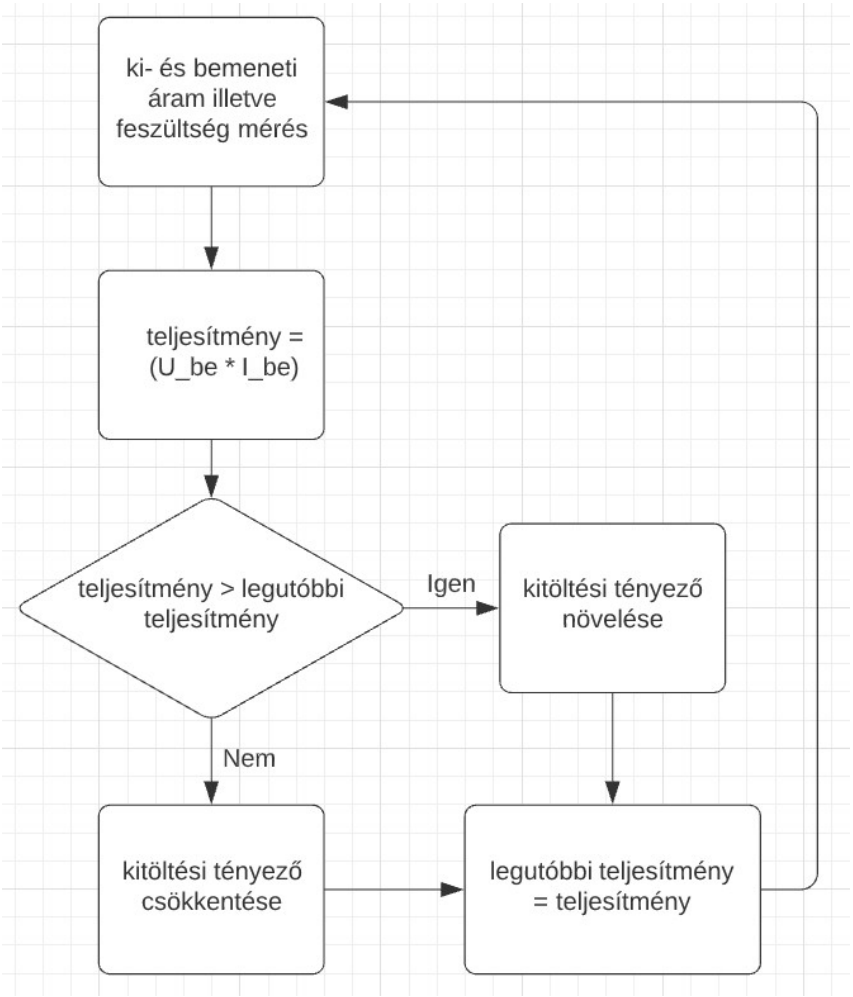
Az irodalomban lehet találni még sok további stratégiát is, például létezik többek között az elmosódott halmazok logikája (fuzzy logic), illetve az árampásztázásra vagy neurális hálózatokra építő megoldások.

A fent ismertetett megoldások közül én a minorkontrollerrel egyszerű megvalósíthatósága miatt a perturb & observe hegymászó stratégia megvalósítása mellett döntöttem.

5.2. MPPT algoritmus megvalósítása

Az MPPT algoritmus a be- és kimeneti feszültségek illetve áramok mérésével kezdődik, majd ezekből kiszámítható a napelemből kivett teljesítmény. Ezután megvizsgálom, hogy

a teljesítmény az előző méréshez képest hogyan változott és ennek függvényében választom meg a kitöltési tényező változtatásának az irányát. A megvalósítandó algoritmusról készítettem egy folyamatábrát, ez látható az 5.1. ábrán.



5.1. ábra. Az MPPT algoritmus folyamatábrája

Az algoritmus megtervezése után implementáltam azt c nyelven:

```
1000 void MPPT()
{
1002     MeasureAdc(); // ki- es bemeneti aram illetve feszultseg meres

1004     inputPower = ((double)inputVoltage * inputCurrent); // napelem
    teljesitmeny
     efficiency = ((double)outputVoltage * outputCurrent) / inputPower; //
    hatasfok
1006
     if (inputPower > lastInputPower)
1008     {
         dutyCycle += increment;
1010     }
     else if (inputPower < lastInputPower)
1012     {
         increment *= -1; // kovetkezo lepes iranya
1014         dutyCycle += increment; // kitoltesi tenyezO valtoztatasa
     }
```

```
1016
        TCA0_SINGLE_CMP2BUF = dutyCycle; // PWM kitoltesi tenyezo beallitasa
1018    lastInputPower = inputPower;
}
```

mpptfunction.c

Ahogy azt az algoritmusok összefoglalása során is írtam, ennek a megoldásnak hátránya, hogy az, ha talál egy lokális maximumot akkor ott megáll és nem feltétlenül jut el a maximális munkapontba. Ennek elkerülésére készítettem el az alább látható függvényt:

```
1000 uint16_t MPPT_Sweep()
{
1002    uint16_t bestDutyCycle = 0;
        uint32_t bestInputPower = 0;
1004    uint32_t inputPower = 0;

1006    for (uint16_t i = top25; i < top80; i++) // 25% - 85%
        {
1008        TCA0_SINGLE_CMP2BUF = i; // kitoltesi tenyezo beallitasa

1010        MeasureAdc(); // ki- es bemeneti aram illetve feszultseg meres
            inputPower = inputVoltage * inputCurrent; // napelem teljesitmeny
1012
            if (inputPower > bestInputPower)
1014        {
                bestInputPower = inputPower;
1016            bestDutyCycle = i;
            }
1018    }

1020    return bestDutyCycle;
}
```

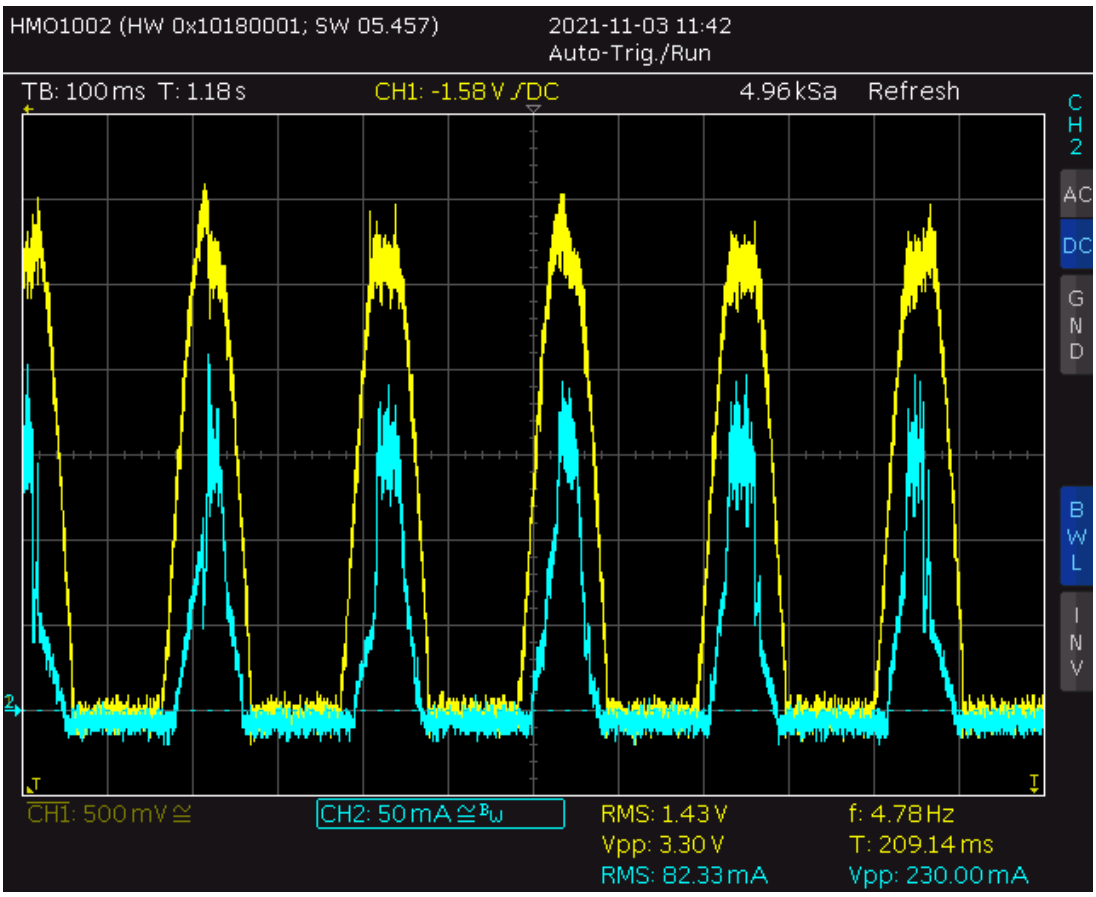
mpptsweepfunction.c

Ez a függvény a kitöltési tényezőt a teljes tartomány 25%-a és 85%-a között változtatva, minden lépésnél megméri a napelem teljesítményét és megkeresi a kitöltési tényezőnek azt az értékét, ahol maximális a teljesítmény.

5.2.1. A műhold forgása

A műhold pályára állításakor, miközben elhagyja a rakéta legfelső fokozatán lévő kidobószerkezetet, a rá ható nem egyenletes erők miatt valamilyen véletlenszerű forgó mozgást kezd el végezni. Ha gyorsan forog a műhold az nem szerencsés, mivel a nem izotróp antenna karakterisztikája miatt ez a forgás fading jelenséget okoz. A napelemek megvilágításának a változási sebességét is a forgás határozza meg, az eddigi műholdjaink közül a SMOG-P-nél mértük a legnagyobb kezdeti forgási sebességet, ez körülbelül 3 Hz volt.

Az MPPT algoritmusnak kellően gyorsnak kell lennie ahhoz, hogy a gyorsan változó megvilágításból adódó maximális munkapont változását követni tudja. Az általam írt algoritmus teszteléséhez az emulátort 5 Hz-el forgó műhold emulálására állítottam be, majd oszcilloszkóppal mértem az akkumulátor töltő áramát. Az 5.2. ábrán látható eredmény alapján az algoritmus tudta követni a maximális munkapontot a gyors forgás ellenére is.



5.2. ábra. A napelem feszültsége (sárga) és az akkumulátor töltőárama (kék)

5.3. Hőmérséklet mérés

Amikor a Nap süti a műhold oldalait, azok fel tudnak melegedni akár 80 °C fölé is, a Föld árnyékában pedig −40 °C-ra is lehűlhetnek. A napelem oldalak hőmérsékletének méréséhez a Texas Instruments által gyártott LM94021 típusú analóg hőmérőt választottam. Ez −50 °C-tól 150 °C-ig képes mérni, így a várható teljes hőmérsékleti tartományban képes lesz méréseket végezni.

A hőmérő kimeneti feszültsége a −50 °C – +150 °C hőmérsékleti tartományban 1299 mV és 183 mV között közel lineárisan változik. A szenzor adatlapja tartalmaz egy másodfokú képletet, amellyel a nem tökéletesen lineáris karakterisztikát is figyelembe véve kiszámítható a hőmérsékletből a kimeneti feszültségéből:

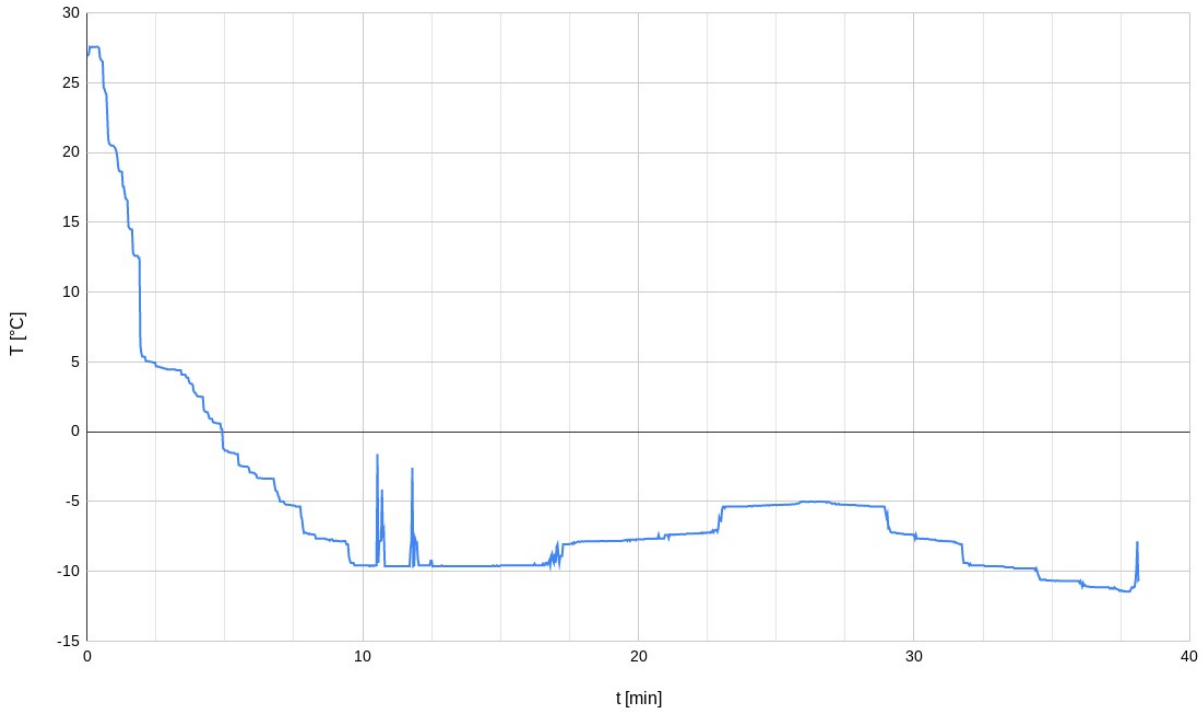
$$V_{TEMP}\ (\mathrm{mV}) = 870,6\ \mathrm{mV} - 5,506\ \frac{\mathrm{mV}}{^\circ\mathrm{C}}(T - 30\,^\circ\mathrm{C}) - 0,00176\ \frac{\mathrm{mV}}{^\circ\mathrm{C}^2}(T - 30\,^\circ\mathrm{C})^2$$

Ezt a képletet átrendezve, a feszültség ismeretében a hőmérsékletet az alábbi egyenlettel kaphatjuk meg:

$$T\ (^\circ\mathrm{C}) = \frac{5,4004\ \frac{\mathrm{mV}}{^\circ\mathrm{C}} - \sqrt{\left(-5,4004\ \frac{\mathrm{mV}}{^\circ\mathrm{C}}\right)^2 - 4 \cdot \left(-0,00176\ \frac{\mathrm{mV}}{^\circ\mathrm{C}^2}\right)(1034,196\ \mathrm{mV} - V_{TEMP})}}{2 \cdot \left(-0,00176\ \frac{\mathrm{mV}}{^\circ\mathrm{C}^2}\right)}$$

A hőmérő tesztelésére végeztem egy kísérletet, az áramkört egy hűtőszekrény fagyasztójába téve, kettő másodpercenként mértem. Az adatokat az optocsatolóval megvalósított egy vezetékes UART-on küldte a mikrokontroller, ahogyan majd az OBC-nek is fogja. A tesztelés során egy FT232-es típusú UART sorosporti átalakító IC használatával az adatokat egy számítógépre juttattam el, ahol egy szöveges fájlba mentettem őket.

A mért hőmérséklet az idő függvényében az 5.3. ábrán látható.

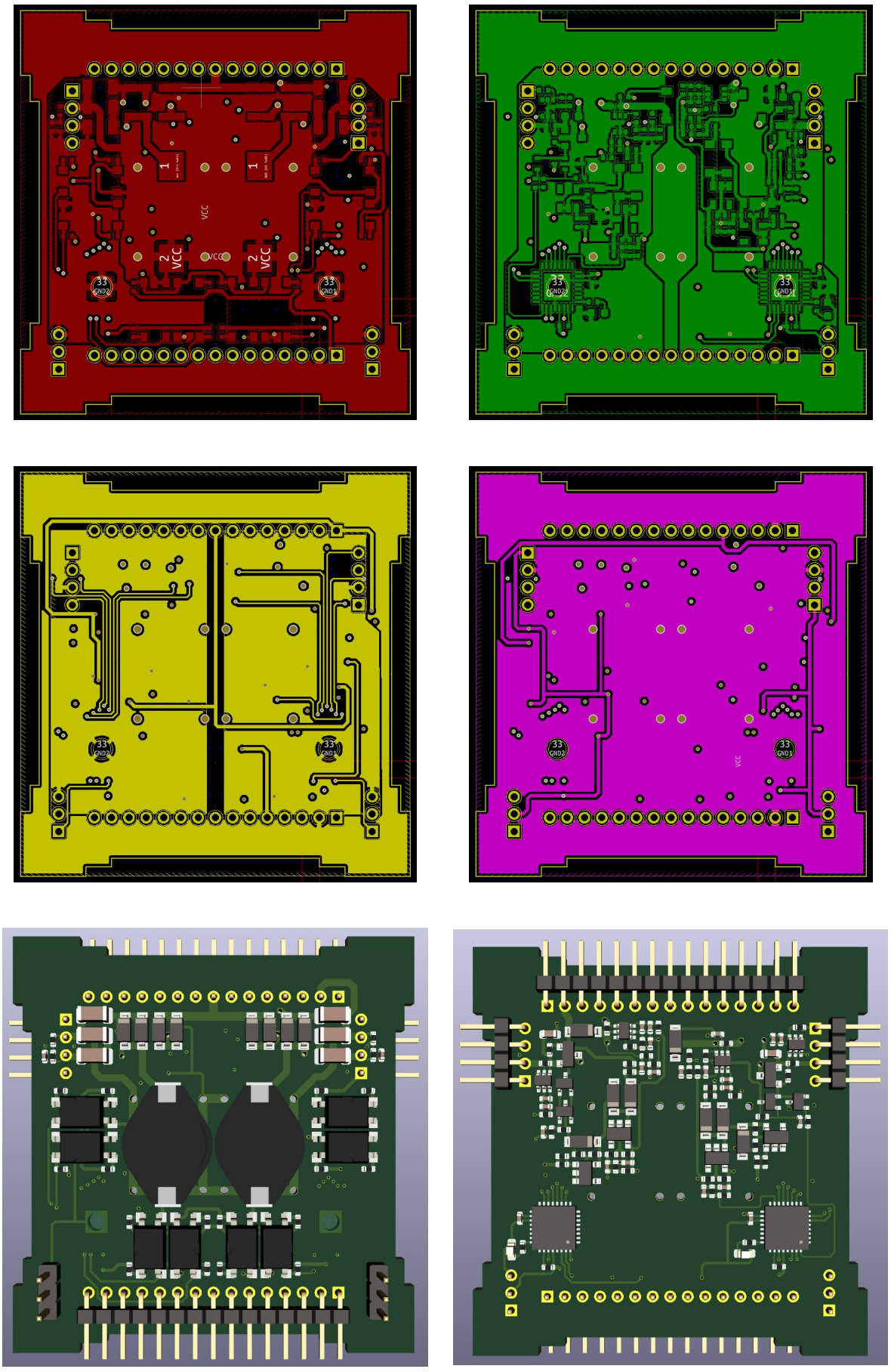


5.3. ábra. A fagyasztóban mért hőmérséklet

5.4. Kvalifikációs példány nyákterv

Közben véglegesítve lett a műhold belső paneljeinek mérete és elkészült a megfelelő alakú kontúr hozzájuk.

A kvalifikációs példány áramköre már nem házilag lesz elkészítve, hanem gyártatott nyákkal fogunk dolgozni. Ez lehetőséget ad kettőnél több réteg használatára. Erre a tervezés során szükségem is volt, mivel a panel két szélén az alrendszerek közötti összeköttetést megvalósító buszcsatlakozó kap helyet, így az áramkörök számára kisebb felület áll rendelkezésre. Az új panelen megtervezett áramkör látható az alábbi ábrákon:



5.4. ábra. A megtervezett áramkör két külső és két belső rétege, illetve a két oldala 3D-s nézetben

6. fejezet

Összefoglalás és folytatás

Munkám során megismerkedtem a Műegyetemen eddig készült műholdak energiaellátó rendszereivel. Megterveztem és elkészítettem a SMOG-2 műhold elsődleges energiaellátó rendszerének a prototípusát, amivel az áramkör működőképessége bizonyítva lett.

További feladat az energia fogyasztás minimalizálása érdekében a mikrokontrollerem fogyasztásának a csökkentése amikor nincs szükség arra, hogy a feszültségszabályzó áramkört működtesse, például amikor földárnyékban van a műhold. Ekkor csak hőmérsékleti adatok gyűjtése lehetséges.

Ki kell dolgozni az OBC-vel való kommunikációt, hogy a telemetria adatokat továbbítani tudjam.

A műhold kvalifikációs példányának a tervezésén kell még tovább dolgozni. Ezt már a végleges műholdban használt panelmérettel és alkatrészekkel fogom elkészíteni, majd ezen további tesztelésekkel kell megvizsgálni a teljes rendszerként való működését, mielőtt a végleges repülő példány megépítését elkezdeném.

Irodalomjegyzék

[1] http://gnd.bme.hu/smog

[2] Markotics Boldizsár, Takács Donát, TDK 2020.

[3] Herman Tibor, A SMOG-1 PocketQube elsődleges energiaellátó rendszere (diplomamunka 2015)

[4] http://www.azurspace.com

[5] http://www.azurspace.com/images/0003429-01-01_DB_3G30C-Advanced.pdf

[6] Zhongfu Zhou, John Macaulay An Emulated PV Source Based on an Unilluminated Solar Panel and DC Power Supply

[7] Géczy Gábor, A SMOG-1 PocketQube másodlagos energiaellátó rendszere (diplomamunka 2016)

[8] https://www.ti.com/lit/ds/symlink/lm94021.pdf

[9] https://www.onsemi.com/pdf/datasheet/ncp512-d.pdf

[10] http://kicad-pcb.org/

[11] David Sanz Morales, Maximum Power Point Tracking Algorithms for Photovoltaic Applications
http://lib.tkk.fi/Dipl/2010/urn100399.pdf

[12] http://www.microchip.com/wwwproducts/en/ATMEGA4808

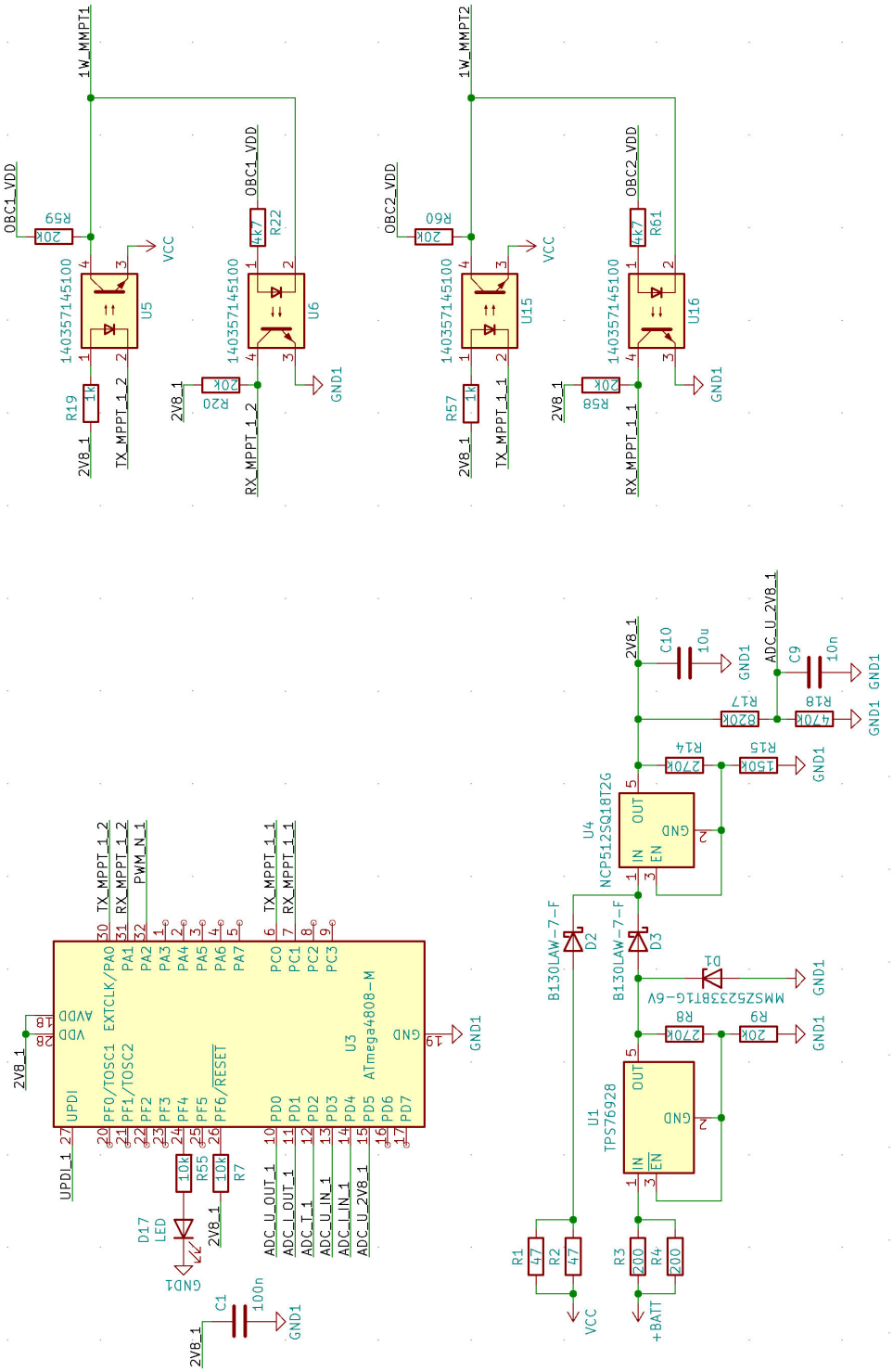
[13] https://www.ti.com/product/INA213

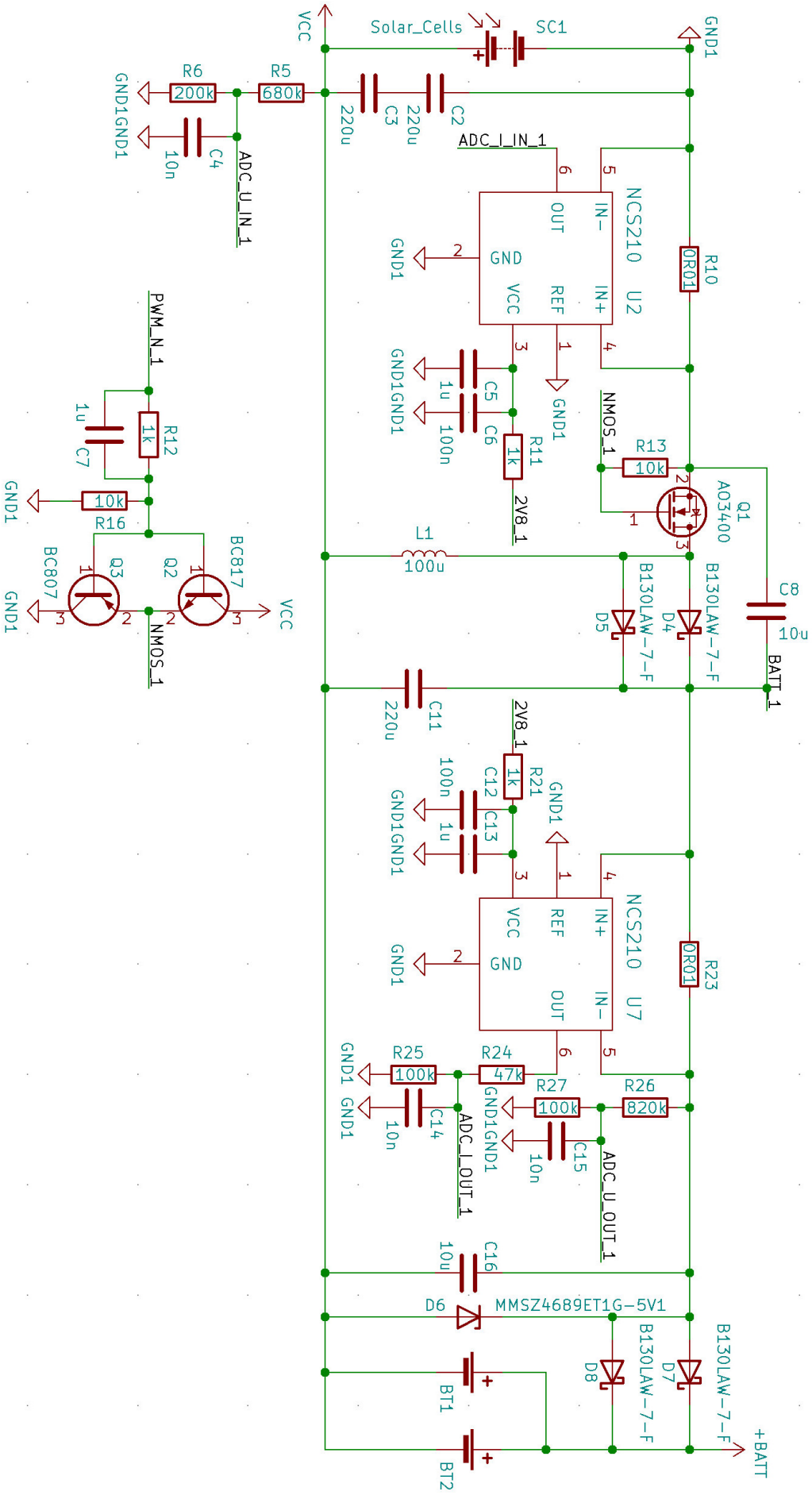
[14] https://www.ti.com/powertopologies

[15] https://www.tij.co.jp/jp/lit/an/slva721a/slva721a.pdf

Ábrák jegyzéke

Függelék


1W_MMPT1
OBC1_VDD
R59
20K
VCC
U5
140357145100
R19
1k
2V8_1
TX_MPPT_1_2
OBC1_VDD
4k7
R22
U6
R20
20K
RX_MPPT_1_2
GND1
1W_MMPT2
OBC2_VDD
R60
R57
R58
R61
U15
U16
TX_MPPT_1_1
RX_MPPT_1_1
U3
ATmega4808-M
UPDI
AVDD
EXTCLK/PA0
PF0/TOSC1
PF1/TOSC2
PF6/RESET
PWM_N_1
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
GND
ADC_U_OUT_1
ADC_I_OUT_1
ADC_T_1
ADC_U_IN_1
ADC_I_IN_1
ADC_U_2V8_1
D17
LED
R55
R7
10K
C1
100n
U4
NCP5122SQ18T2G
IN
EN
OUT
B130LAW-7-F
D2
D3
D1
MMSZ5233BT1G-6V
U1
TPS76928
R8
R9
R14
R15
R17
R18
2K7
150K
820K
470K
C10
10u
C9
10n
ADC_U_2V8_1
R1
R2
R3
R4
47
200
+BATT


VCC
Solar_Cells
SC1
GND1
R6
200k
R5
680k
GND1GND1
10n
C4
220u
C3
220u
C2
ADC_U_IN_1
ADC_I_IN_1
NCS210 U2
OUT
IN-
GND
VCC
REF
IN+
R10
0R01
GND1
1u
C5
100n
C6
R11
1k
2V8_1
NMOS_1
R13
10k
Q1
AO3400
PWM_N_1
1u
C7
R12
1k
10k
R16
BC807
Q3
Q2
BC817
L1
100u
B130LAW-7-F
D5
D4
C8
10u
BATT_1
C11
220u
100n
C12
C13
1u
R21
1k
NCS210 U7
R23
0R01
R25
100k
R24
47k
R27
100k
R26
820k
C14
10n
C15
10n
ADC_I_OUT_1
ADC_U_OUT_1
C16
10u
D6
MMSZ4689ET1G-5V1
D8
D7
BT1
BT2
+BATT

```
1000
#define F_CPU 10000000UL
1002 #include <stdio.h>
#include <stdlib.h>
1004 #include <inttypes.h>
#include <avr/io.h>
1006 #include <avr/xmega.h>
#include <util/delay.h>
1008 #include <stdbool.h>
#include <math.h>
1010 #include <avr/interrupt.h>
#include <util/atomic.h>
1012 #include "macro.h"

1014 uint32_t systemClock = 0;
uint32_t lastClock1ms = 0;
1016 uint32_t lastClock10ms = 0;

1018 #define ADC _delay_us(1); \
            ADC0_COMMAND = 1; /* start conversion // wait for conversion to
    finish */ \
1020             while (ADC0_COMMAND & ADC_STCONV_bm)

1022 double adc_internal_temp = 0.0;
uint16_t adc_T = 0;
1024 uint16_t adc_2V7 = 0;
uint16_t adc_U = 0;
1026 uint16_t inputVoltage = 0;
uint16_t inputCurrent = 0;
1028 uint16_t outputVoltage = 0;
uint16_t outputCurrent = 0;
1030
int8_t sigrow_offset = 0;
1032 uint8_t sigrow_gain = 0;
uint16_t adc_reading = 0; // ADC conversion result
1034
double efficiency = 0.0;
1036
uint16_t maxVoltage = 4200; // mV
1038 uint16_t dutyCycle = 0;
uint16_t bestDutyCycle = 0;
1040
uint16_t lastOutputCurrent = 0;
1042 int16_t increment = 1;

1044 void InitPWM()
{
1046   PORTA_DIR |= PIN2_bm; // pwm output
  PORTA_DIR |= PIN1_bm; // pwm output
1048
  PORTMUX_TCAROUTEA = PORTMUX_TCA0_PORTA_gc;
1050
  TCA0_SINGLE_DBGCTRL |= 1 << 0;
1052
  TCA0_SINGLE_CTRLB = TCA_SINGLE_CMP2EN_bm |
    TCA_SINGLE_WGMODE_SINGLESLOPE_gc; // enable compare channel 2, set
    single-slope PWM mode
1054
  TCA0_SINGLE_EVCTRL &= ~(TCA_SINGLE_CNTEI_bm); // disable event counting
```

```
1056
    uint16_t top = 300;
1058    uint16_t top25 = (uint16_t)(top * 0.25);
    uint16_t top80 = (uint16_t)(top * 0.80);
1060    TCA0_SINGLE_PERBUF = top; // top
    TCA0_SINGLE_CMP2BUF = 0; // compare value
1062
    TCA0_SINGLE_CTRLA = TCA_SINGLE_CLKSEL_DIV1_gc | TCA_SINGLE_ENABLE_bm; //
      set clock source (sys_clk / 1), enable timer
1064 }

1066 void AdcInit()
{
1068    VREF_CTRLA |= 1 << 4; // ADC reference voltage 1V1

1070    ADC0_CTRLA &= ~(1 << RESSEL); // ADC resolution 10 bit
    ADC0_CTRLA |= 1 << ENABLE; // ADC enable
1072
    ADC0_CTRLB = ACC1; // Sample Accumulation Number Select
1074
    ADC0_CTRLC |= 1 << SAMPCAP; // Reduced size of sampling capacitance.
      Recommended above 1 V reference voltages.
1076    ADC0_CTRLC |= 1 << 1; // PRESC = 0x02, DIV8, ADCclk = 1.25 MHz @ F_CPU =
      10 MHz
    // ADC0_CALIB = 1; // 25% Duty Cycle (high 25% and low 75%) must be used
      for ADCclk < 1.5 MHz
1078    ADC0_CALIB = 0; // 50% Duty Cycle must be used if ADCclk > 1.5 MHz;
      ADCclk > 1.5 MHz requires a minimum operating voltage of 2.7 V

1080    ADC0_CTRLD |= 1 << 5; // delay 16 CLK_ADC cycles

1082    ADC0_MUXPOS = 0x00; // external temp sensor
}
1084
void AdcInOut()
1086 {
    ADC0_CTRLB = 2; // 4 results accumulated
1088    ADC0_MUXPOS = 0x03; // U_IN
    ADC;
1090    inputVoltage = ADC0_RES;

1092    ADC0_CTRLB = 4; // 16 results accumulated
    ADC0_MUXPOS = 0x04; // I_IN
1094    ADC;
    inputCurrent = ADC0_RES;
1096    ADC0_CTRLB = 0;

1098    ADC0_CTRLB = 2; // 4 results accumulated
    ADC0_MUXPOS = 0x01; // U_OUT
1100    ADC;
    outputVoltage = ADC0_RES;
1102
    ADC0_CTRLB = 4; // 16 results accumulated
1104    ADC0_MUXPOS = 0x02; // I_OUT
    ADC;
1106    outputCurrent = ADC0_RES;
    ADC0_CTRLB = 0;
1108 }
```

```
1110 void AdcExtTemp()
{
1112   ADC0_MUXPOS = 0x00; // T
   ADC;
1114   adc_T = ADC0_RES;
}
1116
void AdcVoltage()
1118 {
   ADC0_MUXPOS = 0x06; // 2V7
1120   ADC;
   adc_2V7 = ADC0_RES;
1122
   ADC0_MUXPOS = 0x05; // U
1124   ADC;
   adc_U = ADC0_RES ;
1126 }

1128 void MPPT()
{
1130   MeasureAdc(); // ki- es bemeneti aram illetve feszultseg meres

1132   inputPower = ((double)inputVoltage * inputCurrent); // napelem
     teljesitmeny
   efficiency = ((double)outputVoltage * outputCurrent) / inputPower; //
     hatasfok
1134
   if (inputPower > lastInputPower)
1136   {
     dutyCycle += increment;
1138   }
   else if (inputPower < lastInputPower)
1140   {
     increment *= -1; // kovetkezo lepes iranya
1142     dutyCycle += increment; // kitoltesi tenyezo valtoztatasa
   }
1144
  TCA0_SINGLE_CMP2BUF = dutyCycle; // PWM kitoltesi tenyezo beallitasa
1146   lastInputPower = inputPower;
}
1148
uint16_t MPPT_Sweep()
1150 {
   uint16_t bestDutyCycle = 0;
1152   uint32_t bestInputPower = 0;
   uint32_t inputPower = 0;
1154
   for (uint16_t i = top25; i < top80; i++) // 25% - 85%
1156   {
     TCA0_SINGLE_CMP2BUF = i; // kitoltesi tenyezo beallitasa
1158
     MeasureAdc(); // ki- es bemeneti aram illetve feszultseg meres
1160     inputPower = inputVoltage * inputCurrent; // napelem teljesitmeny

1162     if (inputPower > bestInputPower)
     {
1164       bestInputPower = inputPower;
       bestDutyCycle = i;
1166     }
```

```
    }
1168    return bestDutyCycle;
}
1170
int main(void)
1172{
  _PROTECTED_WRITE(CLKCTRL_MCLKCTRLA, CLKCTRL_CLKSEL_OSC20M_gc); // select
    20 MHz oscillator
1174  _PROTECTED_WRITE(CLKCTRL_MCLKCTRLB, 0b00000001); // enable prescaler,
    DIV2 -> f_cpu = 10 MHz

1176  AdcInit();
  InitPWM();
1178
  TCB0_CCMP = 50000; // f_cpu / ccmp = 10 MHz / 10000 = 1 kHz -> 1 ms
1180  TCB0_INTCTRL = 1;
  TCB0_CTRLA = 1; // timer enable
1182
  uint16_t bestOutputCurrent = 0;
1184
  TCA0_SINGLE_CMP2BUF = 0;
1186
  sei();
1188
  while (1)
1190  {
    if (systemClock != lastClock10ms) // true every 100 ms
1192    {
      lastClock10ms = systemClock;
1194    MPPT_Sweep();
    }
1196
    if (systemClock - lastClock1ms >= 10) // true every 1 ms
1198    {
      lastClock1ms = systemClock;
1200    MPPT();
    }
1202  }
}
1204
ISR (TCB0_INT_vect) // Interrupt Service Request for TCA0 Overflow
1206{
  TCB0_INTFLAGS = TCB_CAPT_bm; // Clear the interrupt flag
1208  systemClock++;
}
```

mppt.c

```
1000
#include <avr/io.h>
1002#include <avr/wdt.h>
#include <util/atomic.h>
1004#include <interrupt.h>
#include <stdbool.h>
1006#include <math.h>
#include "main.h"
1008#include <util/delay.h>
#include <avr/eeprom.h>
1010
#define DEADTIME 1
```

```
#define PI 3.1415926535897932

#define CYLCE_TIME 10 // cycle time = CYCLE_TIME * 100 us

#define NMOS OCR1B // nmos pwm output register
#define PMOS OCR1A // pmos pwm output register

uint8_t ID = 0;

void InitADC(void)
{
  DDRC &= ~(1 << 0); // C0 ADC0, output current
  DDRC &= ~(1 << 1); // C1 ADC1, input voltage
                     //    ADC7, output voltage

  ADMUX &= ~(1 << REFS0) & ~(1 << REFS1); // Voltage Reference: AREF,
    Internal Vref turned off
  ADMUX &= ~(1 << MUX0) & ~(1 << MUX1 ) & ~(1 << MUX2) & ~(1 << MUX3); //
    select ADC0, output current

  ADCSRA |= 1 << ADPS0 | 1 << ADPS1 | 1 << ADPS2; // prescaler = 128, ADC
    clock frequency = 125 kHz, single conversion time = 13 ADC clock cycles
    / 125 kHz = 104 us

  ADCSRA |= 1 << ADEN; // enable ADC
}

uint16_t AdcOutputVoltage() // max = 5120 mV
{
  ADMUX |= 1 << MUX0 | 1 << MUX1 | 1 << MUX2; // select ADC7, input voltage

  ADCSRA |= 1 << ADSC; // start Conversion
  while(!(ADCSRA & (1<<ADIF))); // wait until conversion completes
  return ADC;
}

uint16_t AdcInputVoltage() // max = 12890 mV
{
  ADMUX |= 1 << MUX0; // select ADC1, input voltage
  ADMUX &= ~(1 << MUX1) & ~(1 << MUX2);

  ADCSRA |= 1 << ADSC; // start Conversion
  while(!(ADCSRA & (1<<ADIF))); // wait until conversion completes
  return ADC;
}

uint16_t AdcOutputCurrent()
{
  ADMUX &= ~(1 << MUX0) & ~(1 << MUX1) & ~(1 << MUX2); // select ADC0,
    output current

  ADCSRA |= 1 << ADSC; // start Conversion
  while(!(ADCSRA & (1<<ADIF))); // wait until conversion completes
  return ADC;
}

void InitPWM(uint16_t top)
{
  DDRB |= 1 << 1; // B1 output (PWM for p mos)
```

```
1066    DDRB |= 1 << 2; // B2 output (PWM for n mos)
1068    TCCR1A |= 1 << COM1A1; // Clear OC1A/OC1B on Compare Match when
          upcounting. Set OC1A/OC1B on Compare Match when downcounting.
        TCCR1A |= 1 << COM1B1;
1070
        TCCR1A &= ~(1 << WGM10); // PWM, Phase Correct, TOP: ICR1
1072    TCCR1A |= (1 << WGM11);
        TCCR1B &= ~(1 << WGM12);
1074    TCCR1B |= (1 << WGM13);
1076    TCCR1B |= 1 << CS10; // Clock Select: clkIO / 1 (no prescaling)
1078    ICR1 = top; // max 2^16
    }
1080
    static volatile uint32_t counter = 0;
1082
    ISR (TIMER2_COMPA_vect)
1084 {
        counter++;
1086 }
1088 void InitSystemClock(void)
    {
1090    TCCR2A &= ~(1<<COM2A0) & ~(1<<COM2A1); // Normal port operation, OC2A
          disconnected
        TCCR2A &= ~(1<<COM2B0) & ~(1<<COM2B1); // Normal port operation, OC2B
          disconnected
1092
        TCCR2A &= ~(1 << WGM20); // Timer/Counter Mode of Operation: CTC, TOP:
          OCR2A
1094    TCCR2A |= 1 << WGM21;
        TCCR2B &= ~(1 << WGM22);
1096
        TCCR2B &= ~(1 << CS20); // Clock Select: F_CPU / 8 = 2.5 MHz, T = 0.4 us
1098    TCCR2B |= 1 << CS21;
        TCCR2B &= ~(1 << CS22);
1100
        OCR2A = 249; // (249 + 1) * T = 100 us
1102
        TIMSK2 |= 1 << OCIE2A; // Timer/Counter2 Output Compare Match A Interrupt
           Enable
1104 }
1106 uint32_t SystemClockGetValue(void)
    {
1108    uint32_t copy;
        ATOMIC_BLOCK(ATOMIC_FORCEON)
1110    {
          copy = counter;
1112    }
        return copy;
1114 }
1116 int main()
    {
1118    PORTD &= ~(1 << 2); // LED off
        DDRD |= 1 << 2;     // LED output
```

```
1120
    //eeprom_busy_wait();
1122 //eeprom_write_byte((uint8_t*)(0), (uint8_t)ID); // write ID to EEPROM

1124 eeprom_busy_wait();
    ID = eeprom_read_byte((uint8_t*)0); // Read ID from EEPROM
1126
    uint16_t pwmRes = 470; // pwm counter max value
1128
    InitADC();
1130 InitSystemClock();
    InitPWM(pwmRes);
1132
    uint16_t PWMvalue = 0;
1134
    NMOS = 0; // PWMvalue + DEADTIME;
1136 PMOS = 0; // PWMvalue - DEADTIME;

1138 PORTD |= 1 << 2; // LED on
    _delay_ms(50);
1140 PORTD &= ~(1 << 2); // LED off

1142 double Kp = 0.05;

1144 int16_t error = 0;
    double lastError = 0.0;
1146
    double yaw = 0.0 / 95500.0 * CYLCE_TIME; // RPM / 95500.0 * CYLCE_TIME
1148 double pitch = 0.0 / 95500.0 * CYLCE_TIME;
    double roll = 0.0 / 95500.0 * CYLCE_TIME;
1150
    bool isDC = false;
1152 if (yaw == 0.0 && pitch == 0.0 && roll == 0.0) isDC = true; // true if no
      rotation

1154 double yawSum = yaw;
    double pitchSum = pitch;
1156 double rollSum = roll;

1158 double cosYaw = cos(yaw);
    double cosPitch = cos(pitch);
1160 double cosRoll = cos(roll);
    double sinYaw = sin(yaw);
1162 double sinPitch = sin(pitch);
    double sinRoll = sin(roll);
1164
    double cosYawSum = cos(yawSum);
1166 double cosPitchSum = cos(pitchSum);
    double cosRollSum = cos(rollSum);
1168 double sinYawSum = sin(yawSum);
    double sinPitchSum = sin(pitchSum);
1170 double sinRollSum = sin(rollSum);

1172 double R[3][3] = { { cosYaw * cosPitch,  cosYaw * sinPitch * sinRoll -
     sinYaw * cosRoll,  cosYaw * sinPitch * cosRoll + sinYaw * sinRoll },
                       { sinYaw * cosPitch,  sinYaw * sinPitch * sinRoll +
     cosYaw * cosRoll,  sinYaw * sinPitch * cosRoll - cosYaw * sinRoll },
1174                   { -sinPitch,          cosPitch * sinRoll,
                      cosPitch * cosRoll                              } };
```

```
1176    double xyz[3] = { 0.0, 0.0, 0.0 };

1178    switch(ID)
        {
1180      case 1: xyz[0] = 1.0; break;
          case 2: xyz[1] = 1.0; break;
1182      case 3: xyz[2] = 1.0; break;
          case 4: xyz[0] = -1.0; break;
1184      case 5: xyz[1] = -1.0; break;
          case 6: xyz[2] = -1.0; break;
1186    }

1188    double A[3][1] = { { 0.0 },
                           { 0.0 },
1190                       { 0.0 } };

1192    A[0][0] = xyz[0] * R[0][0] + xyz[1] * R[0][1] + xyz[2] * R[0][2];
        A[1][0] = xyz[0] * R[1][0] + xyz[1] * R[1][1] + xyz[2] * R[1][2];
1194    A[2][0] = xyz[0] * R[2][0] + xyz[1] * R[2][1] + xyz[2] * R[2][2];

1196    xyz[0] = A[0][0];
        xyz[1] = A[0][1];
1198    xyz[2] = A[0][2];

1200    // 1 pq 1*40*40  2350 mV 252.5 mA
        // 3 pq 2*40*80  4700 mV 505 mA
1202
        outputVoltageMax = 2000;
1204    outputCurrentMax = 100;

1206    targetVoltage = outputVoltageMax;
        loadResistanceMin = outputVoltageMax / outputCurrentMax;
1208
        bool isCurrentLimit = false;
1210
        sei();
1212
        while(1)
1214    {
          systemClock = SystemClockGetValue();
1216
          if (systemClock - lastClock1ms >= CYLCE_TIME) // CYCLE_TIME = 1 -> 100
        us; CYCLE_TIME = 10 -> 1 ms,
1218      {
            lastClock1ms = systemClock;
1220
            yawSum += yaw;
1222        pitchSum += pitch;
            rollSum += roll;
1224
            cosYawSum = cos(yawSum);
1226        cosPitchSum = cos(pitchSum);
            cosRollSum = cos(rollSum);
1228        sinYawSum = sin(yawSum);
            sinPitchSum = sin(pitchSum);
1230        sinRollSum = sin(rollSum);

1232        R[0][0] = cosYawSum * cosPitchSum;
```

```
    R[0][1] = cosYawSum * sinPitchSum * sinRollSum - sinYawSum *
  cosRollSum;
1234  R[0][2] = cosYawSum * sinPitchSum * cosRollSum + sinYawSum *
  sinRollSum;
    R[1][0] = sinYawSum * cosPitchSum;
1236  R[1][1] = sinPitchSum * sinRollSum + cosYawSum * cosRollSum;
    R[1][2] = sinYawSum * sinPitchSum * cosRollSum - cosYawSum *
  sinRollSum;
1238  R[2][0] = -sinPitchSum;
    R[2][1] = cosPitchSum * sinRollSum;
1240  R[2][2] = cosPitchSum * cosRollSum;

1242  A[0][0] = xyz[0] * R[0][0] + xyz[1] * R[0][1] + xyz[2] * R[0][2];
    A[1][0] = xyz[0] * R[1][0] + xyz[1] * R[1][1] + xyz[2] * R[1][2];
1244  A[2][0] = xyz[0] * R[2][0] + xyz[1] * R[2][1] + xyz[2] * R[2][2];

1246  if (!isDC) targetVoltage = (int16_t)((double)outputVoltageMax * A
  [0][0]);
    if (targetVoltage < 0) targetVoltage = 0;
1248
    outputVoltage = AdcOutputVoltage() * 4.98; // ADC_REF_VOLTAGE / 1024
  * adcOutputVoltage * 2.5133337; 5.065938239
1250  outputCurrent = AdcOutputCurrent() * 1.016889984; // ADC_REF_VOLTAGE
  / 1024 * adcOutputCurrent * 0.534271; 1.076889984

1252  loadResistance = outputVoltage / outputCurrent;

1254  if (outputCurrent > outputCurrentMax) // if current limit reached
    {
1256    if (!isCurrentLimit)
      {
1258      targetVoltage = loadResistance * outputCurrentMax; // new target
  voltage
      }
1260    else isCurrentLimit = true;
    }
1262  else // no current limit
    {
1264    isCurrentLimit = false;
      targetVoltage = (int16_t)((double)outputVoltageMax * A[0][0]);
1266  }

1268  error = targetVoltage - outputVoltage; // calculate output voltage
  error

1270  if (error < 1) PORTD |= 1 << 2; // LED on
    else PORTD &= ~(1 << 2); // LED off
1272
    PWMvalue += (error) * 0.02;
1274
    if ((int16_t)((double)outputVoltageMax * A[0][0]) < 0) PWMvalue = 0;
  // if output value is negative output voltage = 0
1276  if (PWMvalue > pwmRes - DEADTIME) PWMvalue = pwmRes - DEADTIME;
    else if (PWMvalue < DEADTIME) OCR1A = OCR1B = 0;
1278  else
    {
1280    NMOS = PWMvalue + DEADTIME; // n mos
      PMOS = PWMvalue - DEADTIME;
1282  }
```

```
        }
    }
}
```

sce.c