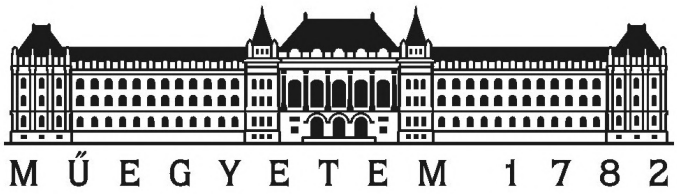


Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Szélessávú Hírközlés és Villamosságtan Tanszék

S-sávú adó fejlesztése 3-PQ méretű diákműholdhoz

Diploma

Miklós Barnabás

2022

HALLGATÓI NYILATKOZAT

Alulírott Miklós Barnabás, szigorló hallgató kijelentem, hogy ezt a Diploma dolgozatot 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, 2022. december 17.

Miklós Barnabás

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
SZÉLESSÁVÚ HÍRKÖZLÉS ÉS VILLAMOSSÁGTAN
TANSZÉK

Diplomatervezés

feladat

Miklós Barnabás

szigorló villamosmérnök jelölt részére, melynek címe

S-sávú adó fejlesztése 3-PQ méretű diákműholdhoz

- Ismerje meg a Mikrohullámú Távérzékelés Laboratóriumban fejlesztett 1-PocketQube (5x5x5cm) osztályú SMOG műholdak és 2-PQ méretű ATL-1 műhold felépítését és működését.
- Tervezzen a következő 3-PQ méretű (5x5x15cm) diákműholdhoz - MRC-100 - illeszkedő műholdfedélzeti nagy sebességű digitális adatkapcsolati rádió adót a következő paraméterekkel:
  - Tápfeszültség: +3,3V
  - Működési frekvencia tartomány: 2245-2290 MHz
  - Adóteljesítmény: legalább 26 dBm (400 mW)
  - Moduláció: QPSK/GMSK
  - Adatsebesség: 10 kbit/s ... 1 Mbit/s (kódolatlan)
  - Digitális adatkapcsolat: full-duplex UART
- Tervezze meg az áramkör kapcsolási rajzát - KiCad.
- Tervezze meg az áramkörhöz tartozó nyomtatott áramkört - KiCad.
- Tervezzen az S-sávú adóhoz illeszkedő, az MRC-100 egyik kisebbik oldallemezén megvalósítható patch antennát (min. 4dBi nyereség): szimuláció, megvalósítás, minősítő mérés.
- Ültesse be, élessze fel, mérje be az áramkört.
- Laboratóriumi és terepi mérésekkel (mérési jegyzőkönyvek) igazolja az áramkörének működését (SDR vevő alkalmazásával).

Irodalom:

- `https://www.silabs.com/support/resources.p-wireless_zigbee-and-thread_efr32mg24-series-2-socs`
- `https://iopscience.iop.org/article/10.1088/1742-6596/2114/1/012029`

Tanszéki konzulens: Dr. Dudás Levente

Tartalomjegyzék

Kivonat

A magyarországi műholdfejlesztések folytatásaként fejlesztett 3 PocketQube osztályú, potenciálisan 6. magyar műhold, az 5x5x15cm-es MRC-100 előreláthatóan 2023 májusában startol majd egy Falcon-9 hordozórakéta segítségével. A start dátuma miatt a műhold hívójele HA100MRC, neve MRC-100, ugyanis ekkor lesz 100 éves a Műegyetemi Rádió Club, ahol a műhold fejlesztése megvalósul.

A műhold fedélzetén helyet kap az általam fejlesztett S-sávú adó is. Ennek oka, hogy a műholdfedélzeti hasznos terhekből adódóan, rövid idő alatt viszonylag sok adat fog keletkezni, és ezeket a földi állomások felett áthaladva nagyjából 100 kbit/s - 1Mbit/s adatátviteli sebességgel kell lesugározni, amelyre a normál UHF sávú telemetria-telekommand rádiólink alkalmatlan, többek között a rádióamatőr sávszélesség korlátok miatt.

A diplomamunkámban leírom a rendszerrel szemben támasztott követelményeket (energiaviszonyok, modulációs mód megválasztás, környezeti paraméterek stb.), az áramköri és nyomtatott huzalozású lemez tervezést, a prototípus panelek élesztésének, bemérésének és validálásának egyes lépéseit. Emellett még az adóantenna tervezésének és megvalósításának a folyamatát is tartalmazza a munka.

A leírás érinti nemcsak a műholdfedélzeti adót, hanem a földi állomás oldalon található vevőt is, amely az esetünkben szoftverrádiós alapokon nyugszik.

Az elsődleges földi S-sávú vevőállomás a BME Etetőn található automatizált, távvezérelt és autonóm energiaellátással rendelkező, jelenleg a SMOG-1-et vevő és vezérlő állomás lesz, amely rendelkezik a megfelelő 3 ill. 4,5 m-es forgásparaboloid apertúra antennákkal, a hozzájuk tartozó X-Y forgatóval, amellyel a LEO pályás MRC-100 megfelelően követhető.

Abstract

As part of Hungarian satellite developments, a 3 PocketQube class, 5x5x15 cm sized MRC-100 satellite will be launched into space on a Falcon-9 rocket in May, 2023.

This device is to become the sixth Hungarian satellite. Its callsign is HA100MRC, its name is MRC-100. The device has been named after the start date which coincides with the centenary of the Radio Club of the Budapest University of Technology and Economics in which these developments have taken place.

On-board the satellite, the downlink communication will be handled by my transmitter circuit. This is necessary because the useful payload of the satellite generates a large amount of data in a short period of time. To communicate this, downlink data speeds of 100 kbit/s-1 Mbit/s are required. These speeds can not be acquired by a normal UHF band telemetry-telecommand radio link, mostly because of the constraints of the radio amateur band.

In my graduate thesis, I discuss the requirements of the system (energy ratios, choosing the right modulation, environmental parameters etc. . .), the schematic and circuit board design, the assembly, measurements and the validation process as well as every step of the transmitter antenna design and validation process.

The thesis includes a write up about the software defined radio based terrestrial receiver station as well.

The primary terrestrial receiver station will be located on BME Etető. The receiver, also the receiver and controller for SMOG-1, is automated, remote controlled and has an autonomous power supply system. It also has a 3 m and 4,5 m paraboloid aperture antenna with an X-Y rotator, which is excellent for following the LEO orbit of the MRC100.

1. fejezet

Áttekintés

2022 december 2.-án adtuk át Magyarország potenciálisan 6. műholdját, mely a 3PQ (5x5x15cm) méretű MRC-100 lesz. A fedélzeten az általam fejlesztett S-sávú adó fogja a Föld felé sugározni a műholdon keletkezett adatokat. Mivel rövid idő alatt nagy mennyiségű adat fog keletkezni, szükség van 100kbit/s-1Mbit/s adatátviteli sebességre.

Ezt a sebességet, melyet az előző műholdakon használt UHF sávú adó nem tudott elérni (főleg a rádióamatőr sávval járó korlátozások miatt), egy robusztus rádiókapcsolattal tervezem elérni. Ez a kapcsolat a műhold fedélzetén lévő relatív nagy teljesítményű, jó hatásfokú és kisméretű S- sávú adóáramkörből és a BME Etetőn lévő automatizált forgásparaboloid antennából fog állni.

Az adó áramkör mellé még tartozik egy, a műhold oldallemezére szerelhető patch antenna is, melynek segítségével stabil kapcsolatot lehet létesíteni a LEO pályán keringő űreszközzel.

A fejlesztés folyamatában két modulációt került szóba: QPSK vagy MSK. E két módszer között minimális különbség van, viszont rengeteg közös tulajdonsággal rendelkeznek. Mindkettő hatékony sávkihasználással rendelkezik, és kifejezetten ellenáll az űr-Föld csatornán fellépő fading jelenségeknek.

Az adó felbocsátás előtt széleskörű tesztelésnek lesz kitéve, melyek szimulálják a felbocsátás és az űr szélsőséges körülményeit.

2. fejezet

Egyetemi műholdak

A Mikrohullámú Távérzékelés Laboratóriumban, e dokumentum írásának időpontja előtt, már készült négy műhold: MASAT-1, SMOG-P, ATL-1 és SMOG-1. Mindegyik sikeresen teljesítette a küldetését.

Ebből a négy űreszközből három a PocketQube méret kategóriába tartozott. Az ATL-1 műholdat az ATL Kft. és a H-ION Kft. együttműködésével fejlesztette a labor, ez az eszköz egy kifejezetten az űrben elvégzendő, speciális akkumulátor hőszigetelő anyagot vizsgáló, kísérletet tartalmazott.

A MASAT-1, SMOG-1 és SMOG-P műholdak fejlesztése teljes mértékben a Budapesti Műszaki Egyetemen történt. Fő küldetésük a Földről űrbe kisugárzott gigahertz alatti spektrumban az elektromágneses sugárzás mérése volt. Emellett még az űrben fellépő különböző ionizáló sugárzásokat is vizsgálták.

Mindhárom műhold Low Earth Orbit pályán került felbocsátásra, és becsült élettartamukat túlteljesítették. A MASAT-1, SMOG-P és az ATL-1 deorbitálódtak, vagyis a légkörbe visszatéréskor elégtek működő műholdként. A SMOG-1 pedig e dokumentum írásának időpontjában még kommunikál, habár ez a kommunikáció nem teljes, mivel a napelemek degradálódása miatt már csak rövid intervallumokban képes működni.

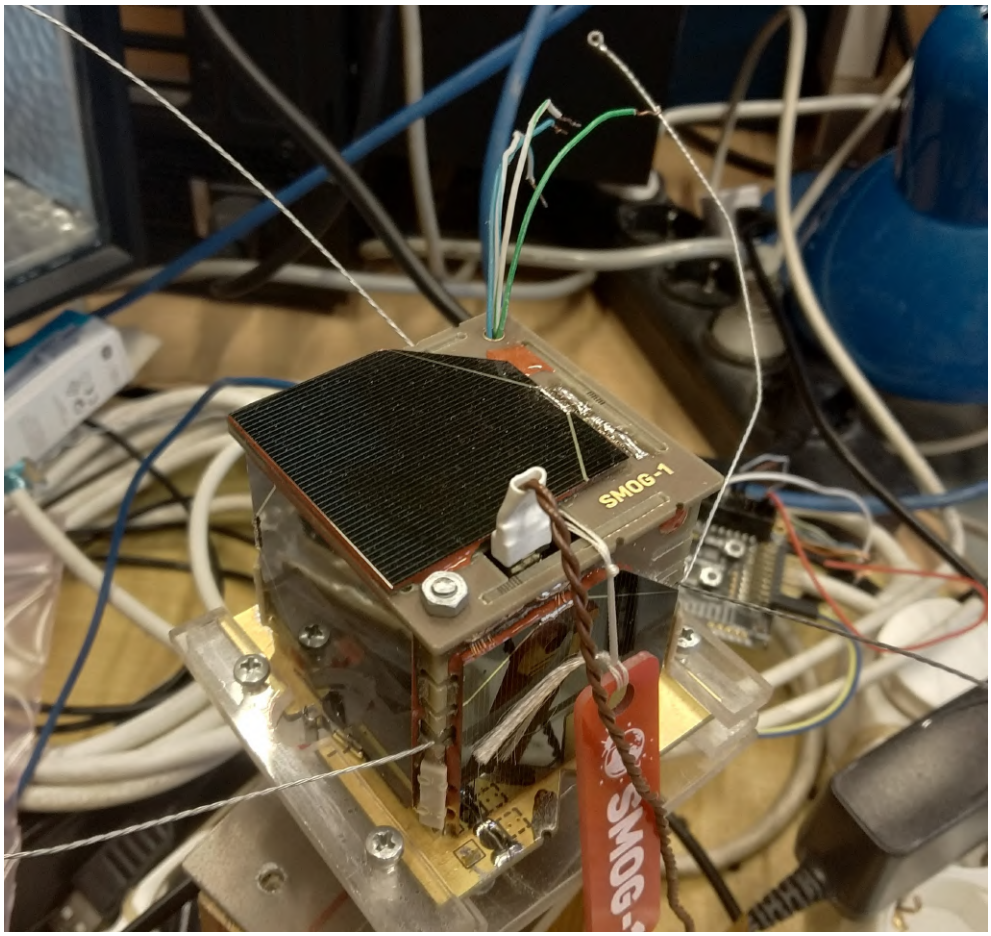
2.1. A rádiós kapcsolat

Az előző űreszközökön, amelyeket a laboratóriumban építettek, a kommunikációs feladatot egy saját fejlesztésű keskeny sávú adó-vevő modul látta el.

A "downlink", azaz ebben az esetben Föld felé irányuló, és az "uplink", tehát Föld felől érkező, kapcsolatot egy UHF sávon üzemelő, 20 dBm adóteljesítményű rádió szolgáltatta. Az alapja egy SiliconLabs EZRadioPRO2 adó-vevő IC-volt. Ennek a maximális kódolatlan sávszélessége 12.5 kbps, ezt a sávban fellépő korlátozások is limitálták. Mind az ATL-1, SMOG-P és SMOG-1 ezt a rádiós interfészt használta, és egy monopól antennával sugározták a Föld felé az adatokat.

Ez a rendszer az MRC-100 fedélzetén szintén helyet kap, és továbbra is végzi a Földdel való kétirányú kommunikációt: az érkező parancsok vételét és a telemetria küldését. Viszont az adatok leküldéséhez már nem lesz elég az általa szolgáltatott sávszélesség, mivel az új műholdon sokkal több mérési adat fog keletkezni. A két nagy sávszélességű spektrum analizátor által gyűjtött jelerősség adatokat és a kamera által készített képeket egy

10 perces áthaladás alatt kell lesugározni. Ezért szükség van egy nagy sebességű downlink rádióadóra. Ennek az adó áramkörnek a fejlesztését írja le ez a dokumentum.



2.1. ábra. A SMOG-1 műhold, 1PQ méret (5x5x5cm)

3. fejezet

A rádióadó

Egy ilyen adóáramkör tervezésének szigorú követelmények szabnak határokat. A pontos tervezési paraméterek 2021 áprilisában derültek ki, ez vezetett némi találgatáshoz és komplikációkhoz a tervezés és megvalósítás folyamatában.

Az 1PQ kontúrba (nettó 4x4cm) bele kell, hogy férjen kettő darab adó, tehát egy redundáns egység is. Ekkora méretű műholdnál limitáció a napelemek által termelt energiamennyiség, ezért az eszköz fogyasztására határozott kitételek voltak megszabva: 3,3 voltos energiabuszról, maximum 1 A fogyasztással kell, hogy működjön. Erről a feszültségről pedig akkora sugárzott adóteljesítményre van szükség, hogy az 1Mbit/s kódolatlan adatátviteli sebességet stabilan tudja tartani a kapcsolat.

Összefoglalva, az STX (az S-sávú adóra a csapatban használt rövidítés: STX, azaz S-band Transmitter) elvárt paraméterei:

- 2245-2290 MHz, S-sáv. 2022 novemberben pontosult 2267,5 MHz-re
- Alacsony fogyasztás(< 1 A)
- PQ panel (körülbelül nettó 4x4 cm)
- Működjön a műhold 3,3 V szabályozott energiabuszáról
- Hatékony sávkihasználás, 100kbit/s - 1Mbit/s kódolatlan adatsebesség
- Kimeneti teljesítmény: körülbelül 26 dBm
- Sugárzott teljesítmény: 30 dBm EIRP

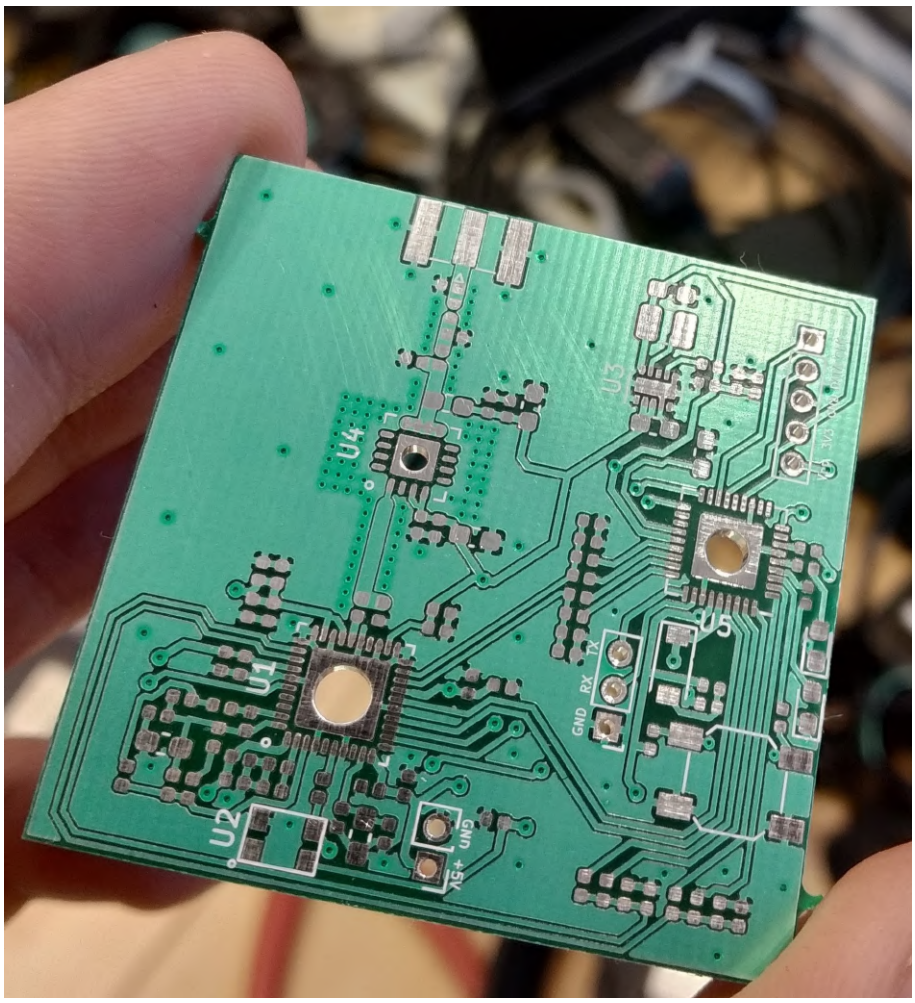
  A következő szekciókban röviden, a teljesség igénye nélkül azokat az alkatrészeket mutatom be, amelyek szóba jöhettek az adó megvalósítása során.

3.1. Különálló szintézeres áramkör

Korábbi félévben többféle megközelítés lett kipróbálva [2]. Az egyik egy különálló szintézerrel és mikrovezérlővel rendelkező sokrétű eszköz. Széles frekvenciasávot lefed (2100-2600 MHz [12]) és képes többféle QAM moduláció megvalósítására (ebből csak QPSK lett tesztelve). Célja a szintézer IC tesztelése: megfelel-e egy műholdfedélzeti adó követeléseinek?

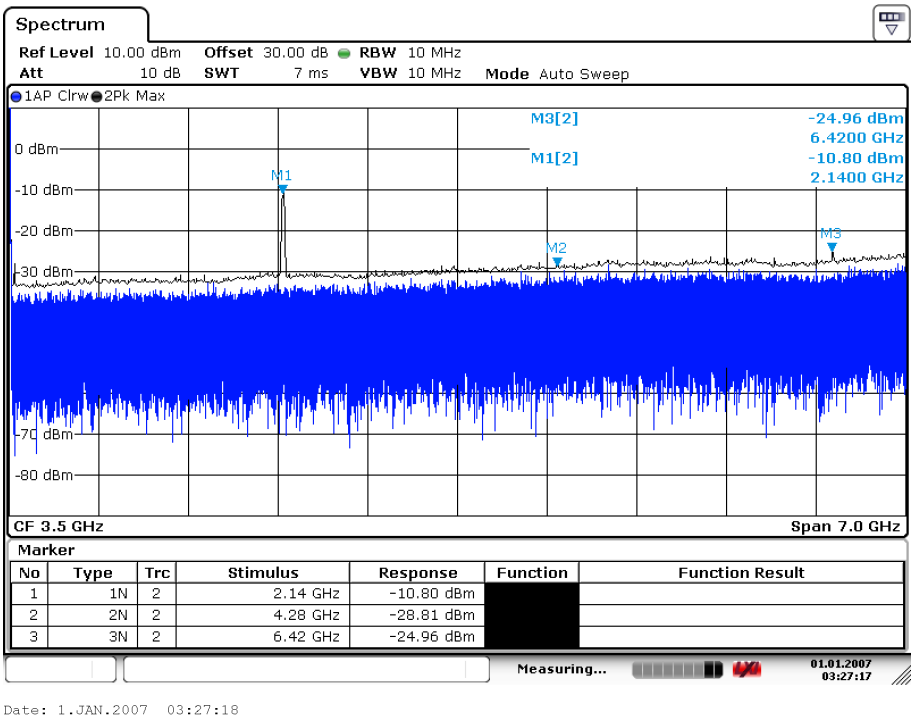
A szintézer IC egy ADRF6703 típusú eszköz volt az Analog Devices kínálatából. Az alapsávi jelet egy PIC32MM0064GPL036 alacsony fogyasztású mikrovezérlő állítja elő. A szintézer bemeneteinek differenciális I és Q jelre van szükségük. Ezt nyolc GPIO láb és egy feszültségbeállító ellenállás hálózat segítségével állítja elő a mikrovezérlő.

A tárgyalt áramkör legyártva 3.1. ábrán látható.



3.1. ábra. A különálló szintézeres áramkör megvalósítása nyomtatott huzalozású lemezen

Az adó, bár kifejezetten szélessávban működik és sokféle modulációra képes, 5 V tápfeszültségről működik hivatalosan. Tesztelve lett 3.3V tápfeszültségről is, de nem hozta a megfelelő eredményeket. Magas fogyasztás mellett (200mA) nem produkált különösebben jó teljesítményszinteket (-10dBm), lásd 3.2. ábra.

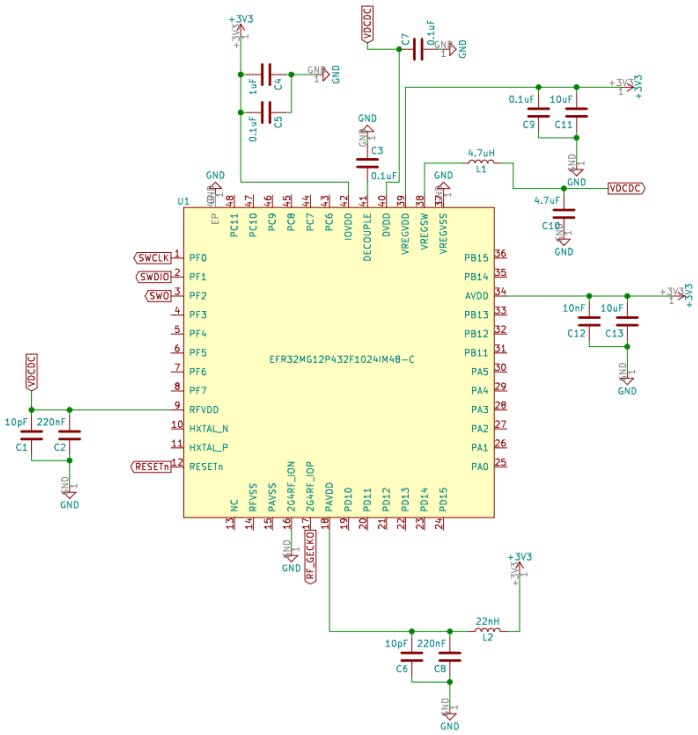


3.2. ábra. Az ADRF6703 szintézer kimeneti teljesítménye 3.3V tápfeszültségen [2]

3.2. Integrált szintézeres áramkör

A tervezett frekvencia, melyen a műhold kommunikálni fog, 2 GHz és 2,4 GHz közé esett a pontos specifikációk előtt. A 2,4 GHz-es ISM sávokban zajló kommunikáció viszont jelentős interferenciát okozott volna a csatornán. Ezért minél távolabb kerül az aktuális csatorna a jelentős zajteljesítménnyel telített sávoktól, annál hatékonyabb lesz a kommunikáció. Viszont azok a kereskedelmi forgalomban lévő szintézer IC-k, amelyek hivatalosan lefedik ezt a 2,4 GHz alatti frekvenciasávot, és képesek saját rádiós protokoll megvalósítására (nem pedig előre kódolt protokollokra mint a BLE, Zigbee stb.), nem túl magas hatásfokúak, és általában 5 V (vagy több) tápfeszültségről üzemelnek.

Azok az IC-k, melyek képesek 3,3 voltról üzemelni és magas hatásfokúak is, mind az ISM sávokat lefedő adó-vevő eszközök. Ezekben a chipekben a rádiós kommunikáció teljes hardveres része egy szilíciumra van integrálva (alapsávi jelgenerálás, felkeverés és erősítés plusz szűrők). Mivel az ISM sávokban üzemelő IC-kre nagy a piac, ezért olcsók és optimalizáltak. De az említett eszközök paraméterei között nem található meg általában az, hogy el tudják-e érni a 2270 MHz-es frekvenciát (amelyen az adó üzemelni fog). Ennek több oka is lehet. Első, hogy fizikailag nem képes a szintézer lemenni eddig a frekvenciáig. A második, le tud menni, de nem lehet elérni ezt a beállítást csak speciális eszközökkel. A harmadik, le tud menni és egyszerű is elérni ezt a beállítást, csak nem írták bele az adatlapba, mivel az átlag felhasználó úgyis csak az ISM sávban fogja használni stb. A harmadik opció a legelőnyösebb. Emiatt sok időt töltöttem a különböző chipgyártók ügyfélszolgálataival való levelezéssel, hogy megtaláljam, melyik ilyen eszköz képes mégis lefedni a megfelelő frekvenciasávot.



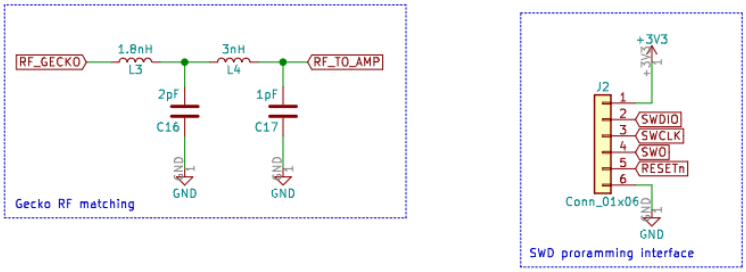
3.3. ábra. EFR32MG12 adó-vevő integrált áramkör, 20 dBm adóteljesítményre tervezett kapcsolásban

3.2.1. SiliconLabs EFR32 Series 1

Egy eszköz, amelyet még korábbi félévekben találtam a Silicon Labs EFR32MG12 chip volt. Ennek a szintézere 2360 MHz-ig volt képes jelet előállítani(adatlapon nem jegyzett), ekkor még nem volt tisztázva a pontos adófrekvencia ezért terveztem ezzel az eszközzel egy kapcsolást (3.3. ábra).

Ez a chip elvileg 20 dBm adóteljesítményre képes [15]. A megvalósított kapcsolás erre a teljesítményre optimalizált, emellett igénybe veszi az IC belső DC-DC konverterét, amely minimálisan több alkatrészt jelent, viszont hatékonyabb működést [16].

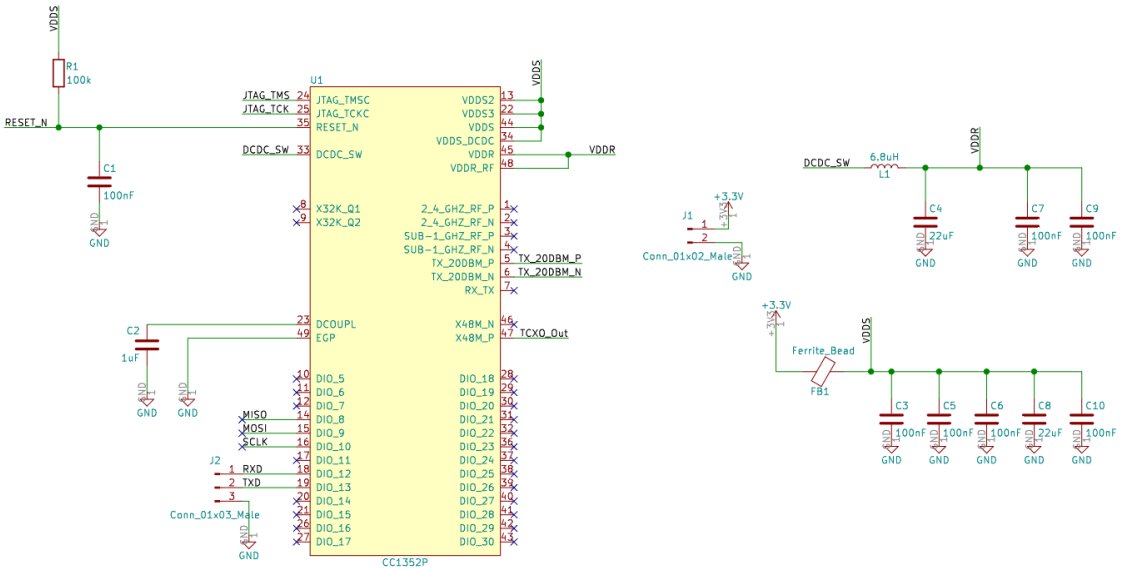
A kapcsolásban az illesztés is az említett adóteljesítményre történt [17]. Sajnos ez az IC nem tudja kezelni a végleges frekvenciasávot, ezért nem lett használva, és a kapcsolás is befejezetlen maradt, amikor pontosították a rádió paramétereit.



3.4. ábra. EFR32MG12 illesztő áramköre és programozó interfésze

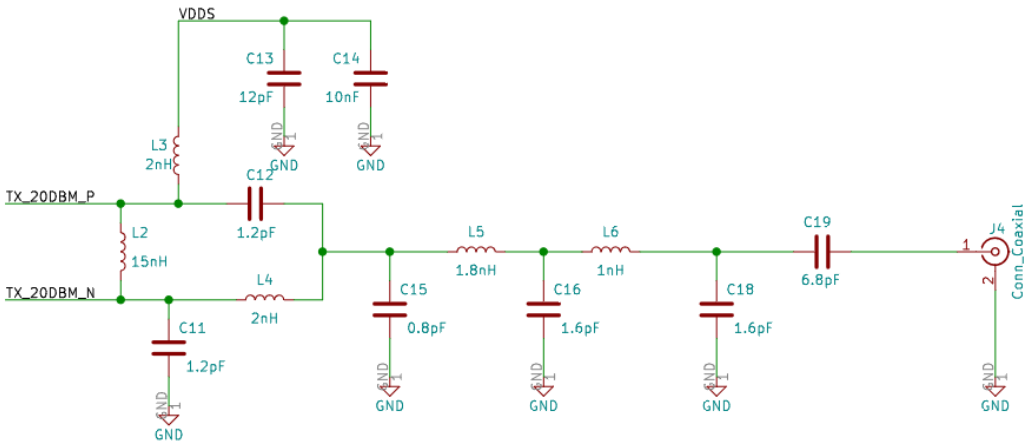
3.2.2. Texas Instruments rádiós IC

Miután fixálódott a működési frekvenciasáv, újra elkezdtem keresni olyan integrált áramkört, amely megfelel az elvárásoknak. Mivel még távolabb került a frekvencia az ISM sávoktól nehezebb feladat lett találni ilyen IC-t. De ennek ellenére megtaláltam a Texas Instruments katalógusában a CC1352P típusú chipet, melynek a szintézere a vevőszolgálat szerint képes 2153 MHz-ig lemenni, és lehet rajta implementálni saját rádiós protokollokat is. Ezért ehhez az eszközhöz terveztem egy kapcsolást. A CC1352P is 20dBm kimenő

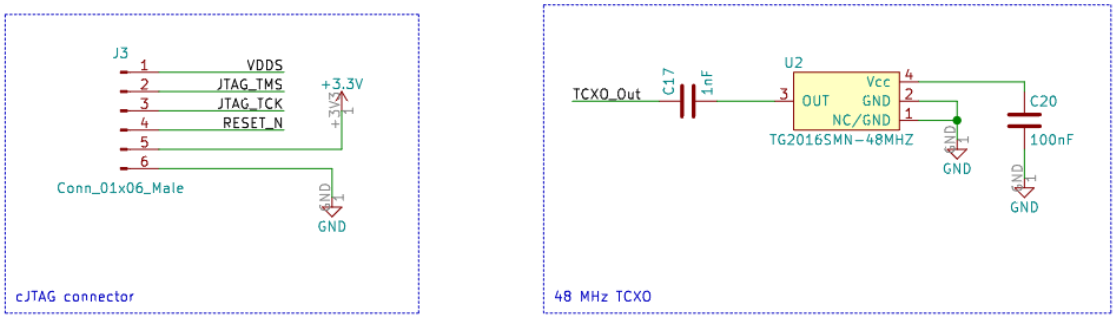


3.5. ábra. A CC1352P adó-vevő integrált áramkör alapvető kapcsolása

teljesítményt tud 2,4 GHz-en, ezt tudja 3 V és 85 mA fogyasztás mellett [18]. Amely hogyha pontos adat, akkor kitűnő hatásfokot jelent. A megvalósított kapcsolás erre a teljesítményre optimalizált, emellett igénybe veszi az IC belső DC-DC konverterét amely minimálisan több alkatrészt jelent, viszont hatékonyabb működést [19]. A kapcsolásban az illesztés is az említett adóteljesítményre történt [20]. A hőmérséklet eltolódásból adódó frekvencia- és fázishiba csökkentésének érdekében egy TCXO (Temperature Compensated Oscillator) szolgáltatja az órajelet a rádiómodulnak.

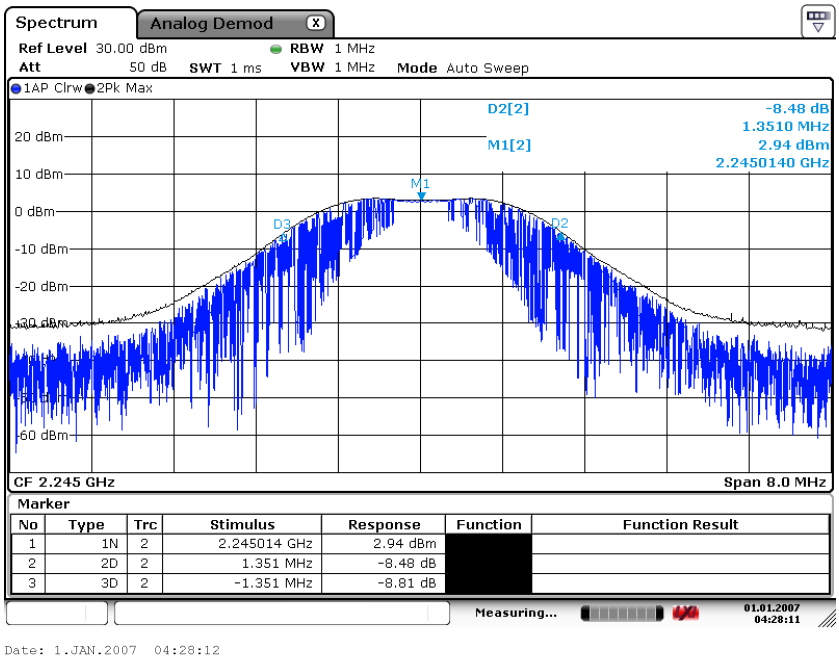


3.6. ábra. A CC1352P adó-vevő IC illesztő kapcsolása 20 dBm teljesítményen



3.7. ábra. A TXCO és a programozó cJTAG interfész

Ez az eszköz csak Gaussian FSK modulációra képes, melyet lehet MSK-ként konfigurálni.



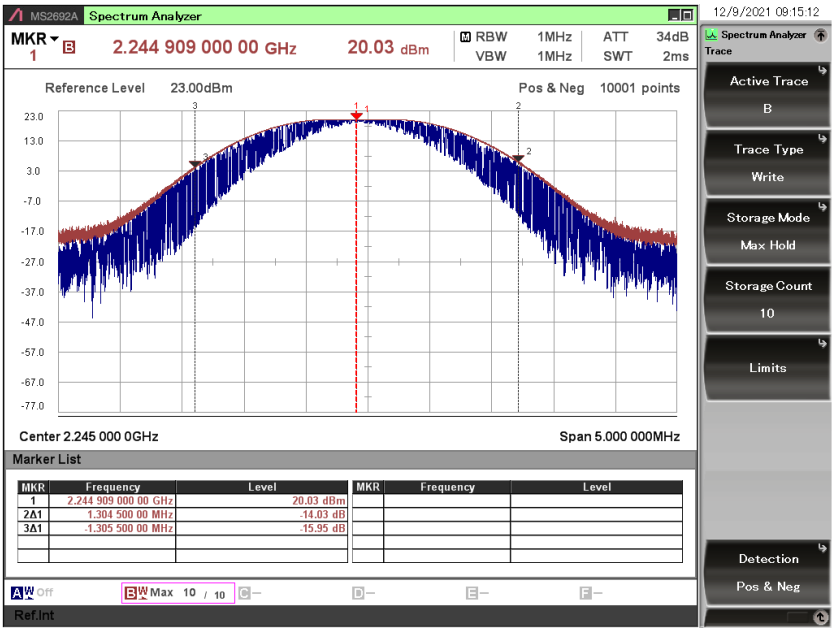
3.8. ábra. A CC1352P kimeneti teljesítménye

Az IC képes volt kiadni a megfelelő frekvenciájú és sávszélességű jelet, de az adó teljesítmény kérdésében nem teljesített az elvárásoknak megfelelően (3.5. ábra). Ezen lehetett volna változtatni a konfigurációval való kísérletezéssel, de kedvező körülményeknek köszönhetően adódott egy másik IC, melynek a konfigurálása sokkal egyenesebben ment, és a paraméterei is kitűnőek voltak.

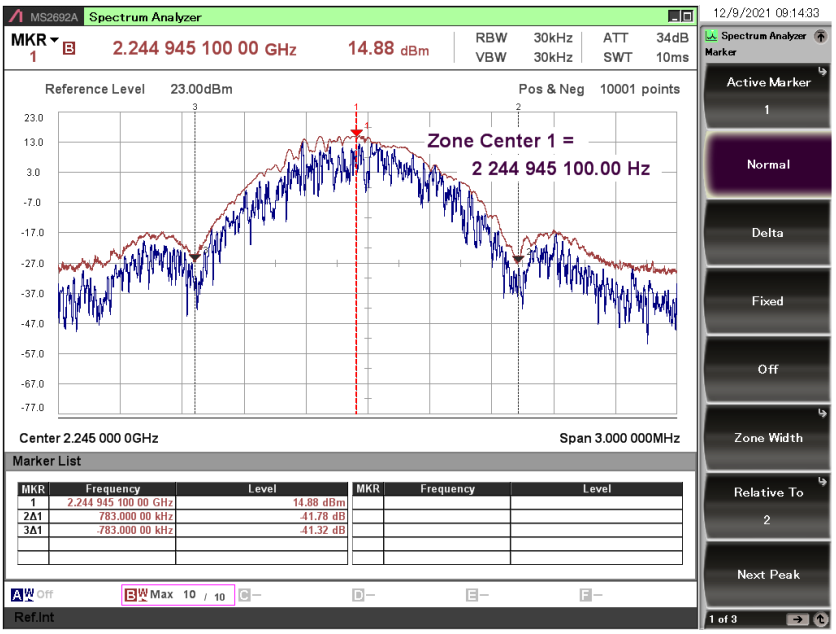
3.2.3. SiliconLabs EFR32 Series 2

A következő chip, mely ki lett próbálva a SiliconLabs EFR32MG24 20dBm-es verziója volt. Ennek a szintézere is képes a 2245-2290 MHz sávot lefedni és szintén konfigurálható GMSK modulációra 2 Mbps sávszélességgel.

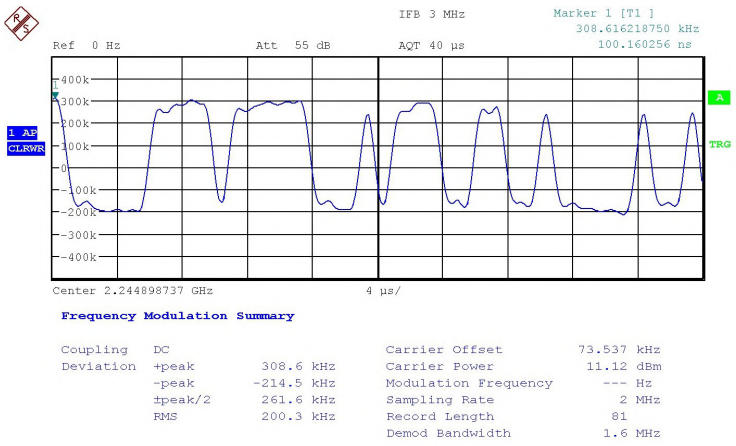
Miután validálva lett egy hivatalos panelen (BRD4187C [9]) vezetett mérés segítségével (3.9. ábra, 3.10. ábra, 3.11. ábra), ezzel az IC-vel lesz megvalósítva a műholdfedélzeti adó áramkör.



3.9. ábra. EFR32MG24 kimeneti teljesítménye



3.10. ábra. EFR32MG24 GMSK modulált kimeneti spectrum



3.11. ábra. EFR32MG24 demodulált alapsávi jelalak

3.3. Végfok erősítők

Az erősítő választást kevésbé limitálta a frekvenciasáv; ezzel szemben a 3 V tápfeszültség kitétel igen. Az elvárt (legalább 27 dBm) teljesítményt elérő IC-k nagy része 5 V vagy magasabb tápfeszültségről üzemel. Ha viszont egy rádióhullámú tranzisztorról van szó, általában nincs megadva az adatlapon, hogy a nominális feszültség alatt miként teljesít az eszköz. Azt majdnem biztosan állítom, hogy nem létezik 3.3 V nominális drain-source feszültségről üzemelő, kisebb, mint 10 dBm erősítésű, 27 dBm kimenő teljesítménnyel rendelkező tranzisztor. Ezért ezeket a paramétereket szükséges kimérni.

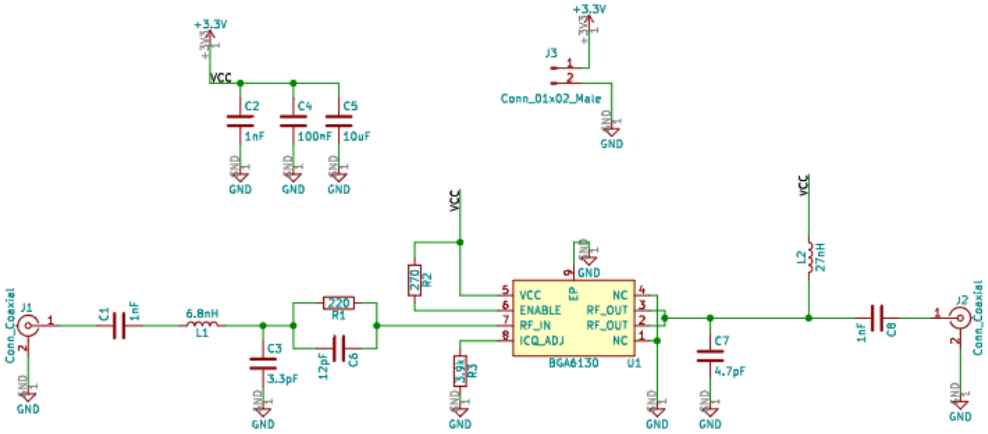
A következő szekciókban méréshez tervezett tesztáramkörök fognak következni. Az ezekhez tartózó mérés úgy fog zajlani, hogy egy megfelelő teljesítményű jelforrás lesz kapcsolva a bemeneti SMA csatlakozóra, például egy nagyfrekvenciás jelgenerátor. A kimeneti SMA csatlakozóra pedig egy spektrumanalizátort kapcsolok, amelyen mérve lesz a kiadott teljesítmény. A tápfeszültséget egy labortáp szolgáltatja.

Ez alól kivételt képez az ADL5606 típusú IC, mely az ADRF6073 szintézer IC-t tartalmazó áramkörrel mértem ki [2].

3.3.1. BGA6130

Ez az integrált erősítő áramkör nem egy diszkrét tranzisztor, és 3 voltról is üzemel. Viszont szükséges a mérése, mivel az elvárt frekvenciához közeli mérési adatokat nem tartalmaz az adatlapja. Ezzel szemben azt állítja, hogy sávszélessége 2700 MHz-ig terjed [22]. Az erősítő adatlap szerint kiemelkedően jó hatásfokkal (55 % Power Added Efficiency, erősítő hozzáadott teljesítményének hatásfoka) büszkélkedik [22]. A mérési áramkör magát az IC-t, a hozzá tartozó impedancia illesztő hálózatot, tápellátást és bemeneti plusz kimeneti SMA csatlakozókat tartalmaz.

Mivel csak a szóródási paraméterekből nem sikerült értékelhető teljesítményillesztést tervezni az erősítőhöz, és a gyártó nem szolgáltatott bármiféle egyéb segítséget ezen a frekvencián való illesztéshez, ennek az eszköznek a használatát hanyagoltam.

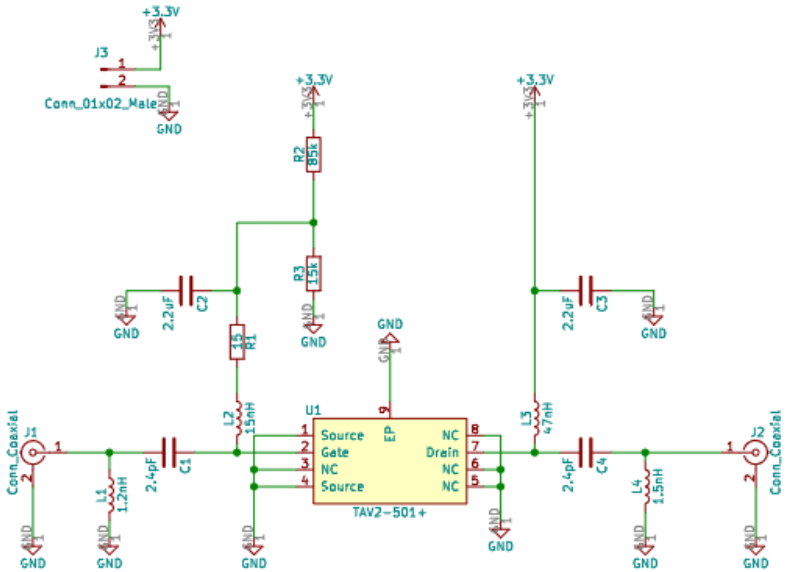


3.12. ábra. BGA6130 erősítő tesztpanel kapcsolás

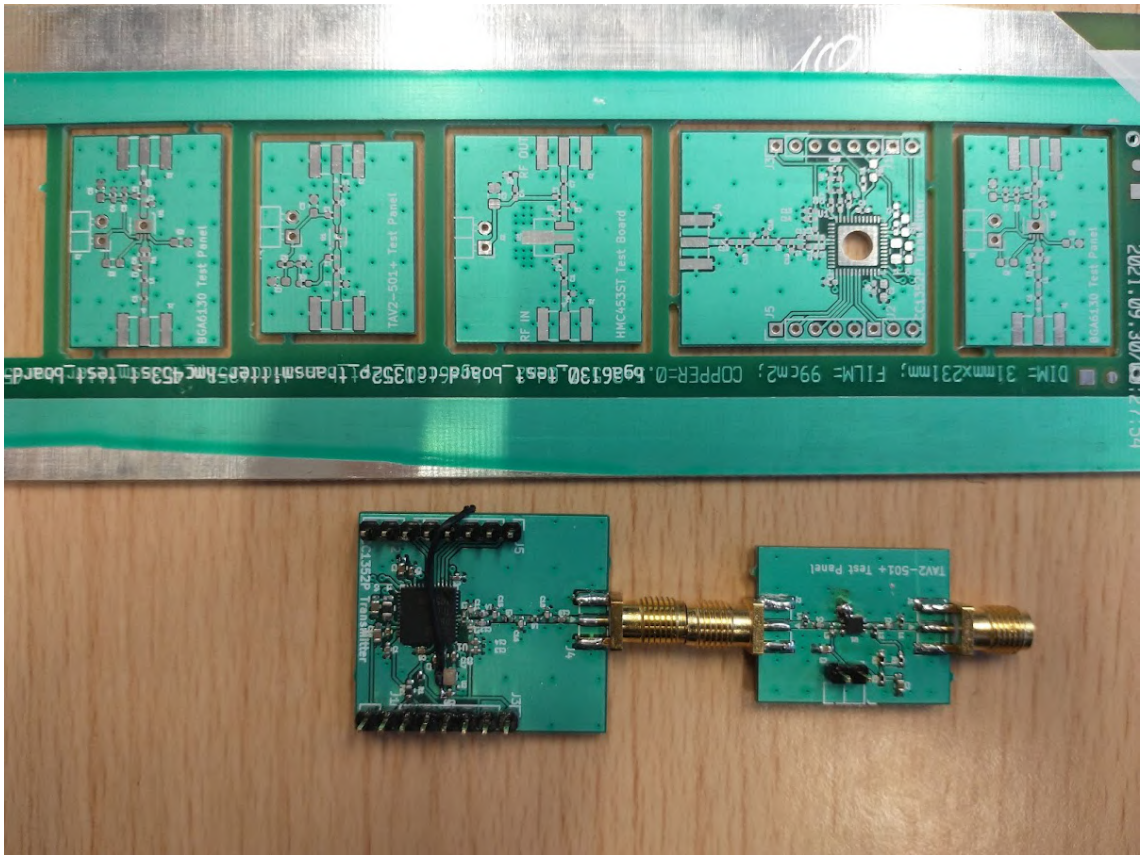
3.3.2. TAV2-501+

Ez a tranzisztor megfelelő kimenő teljesítménnyel és sávszélességgel rendelkezik, viszont csak 4,5 V drain-source feszültségen mért adatai vannak [21]. Ezért, hogy ez is ki legyen mérve 3,3 volton, terveztem hozzá egy erősítő áramkört. Ez tartalmazza a referencia erősítő kapcsolást [21], gate-source feszültség beállító hálózatot, impedancia illesztést és bemeneti, valamint kimeneti SMA csatlakozókat.

Mivel már 4,5 volton kis teljesítményen telítődik az eszköz, és a tokozása sem megfelelő egy hűtéskritikus alkalmazáshoz, a használatát hanyagoltam.



3.13. ábra. TAV2-501+ erősítő tesztpanel kapcsolás

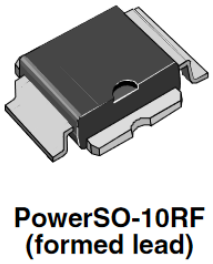


3.14. ábra. Az erősítő és CC1352P tesztpanelek legyártva

3.3.3. PD20010-E

Ez a tranzisztor sokkal nagyobb kimeneti teljesítménnyel és nominális feszültséggel rendelkezik, mint a tervezett [23]. De a MASAT-1 küldetés során is egy ehhez nagyon hasonló, csak más sávszélességű modellel sikereket értek el a laboratóriumban. Ezért alacsonyabb feszültség- és teljesítményszintekkel kell mérni, mivel nagy előnye az előzőekkel szemben, hogy a tokozása sokkal jobban hűthető.

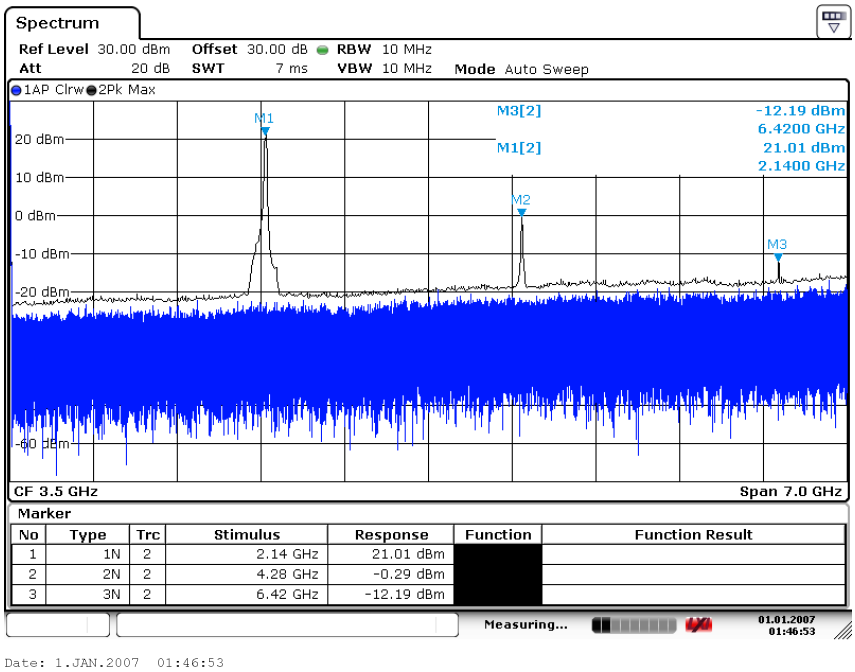
Az eszközzel viszont elérhetőségi problémák léptek fel, tehát a vele való fejlesztés hanyagoltam.



3.15. ábra. A PD20010-E tokozása

3.3.4. ADL5606

Ez az IC 5 V feszültségről üzemel, és előnyös adatlapi paraméterekkel rendelkezett. Viszont mivel könnyen gerjedt, és nem tudta hozni az elvárt teljesítményt 3,3 V feszültségen (3.16. ábra), nem ez lesz a megfelelő erősítő [2].

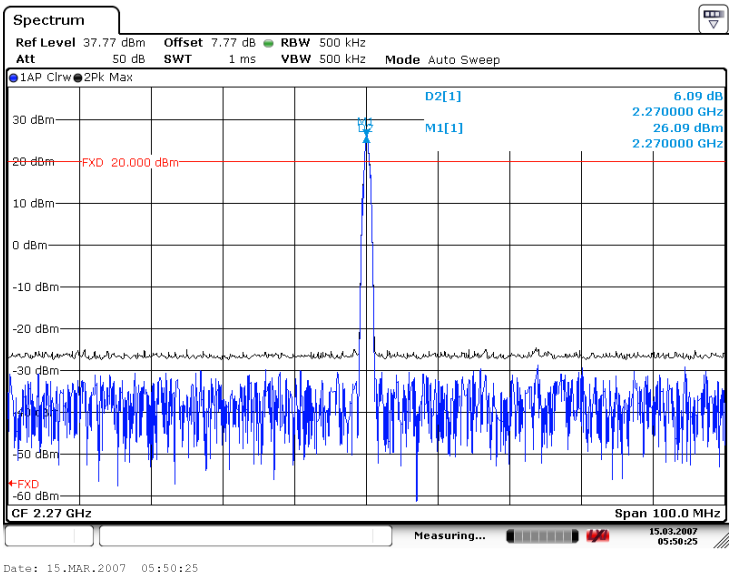


3.16. ábra. ADL5606 spektruma 3.4V tápfeszültségen

3.3.5. HMC453ST

A HMC453ST szintén 5 V feszültségről üzemel, viszont adatlap szerint majdnem képes lefedni a szükséges sávot. Tokozása kifejezetten jól hűthető, ez előnyére válik majd az űrbéli vákuumban.

Ehhez az eszközhöz sikerült hivatalos fejlesztőpanelt beszerezni, és különösebb problémák nélkül validálni a teljesítményét 2270 MHz-en a gyári áramkörrel: 3,3V tápfeszültségen 6,1 dB erősítés 26,1 dBm kimenti teljesítmény(3.17. ábra), amely jobb illesztőhálózattal javítható 27 dBm-re.

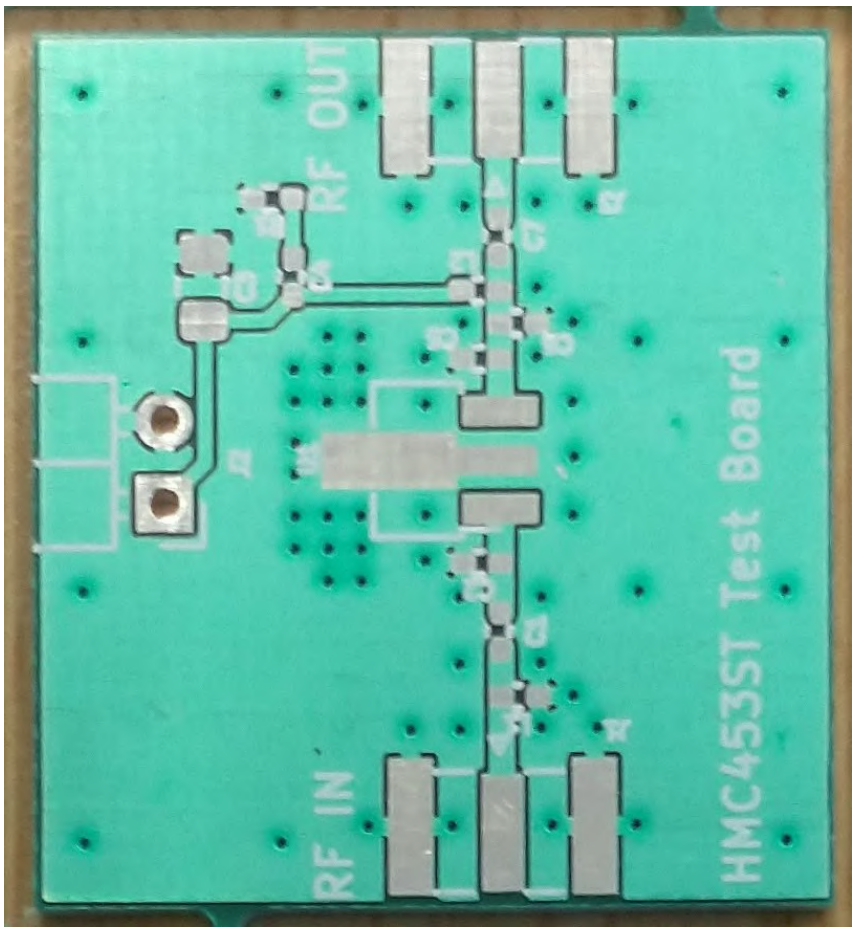


3.17. ábra. HMC453ST 2100MHz hivatalos fejlesztőpanel vezetett mérés 2270 MHz-en

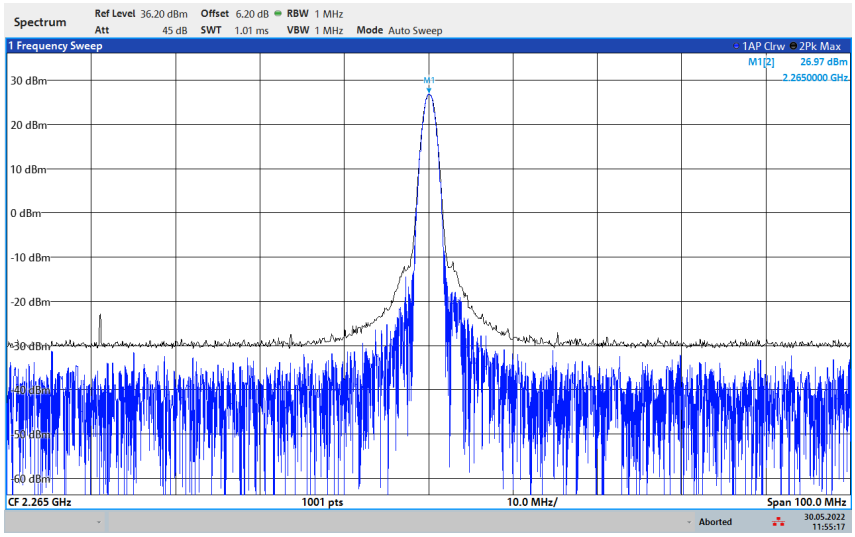
Mivel az eredmények reménykeltőek voltak, saját tervezésű panelen folytatódott az illesztőhálózat finomhangolása (3.17. ábra). A végeredmény 7dB erősítés lett, 27 dBm kimeneti teljesítménnyel, fogyasztása 370 mA/3,3 V, mely 33% PAE (Power Added Efficiency, erősítő hozzáadott teljesítményének hatásfoka, 3.2. egyenlet) értékkel jobb mint az adatlapi 22% PAE érték [6]. Ez valószínűleg annak köszönhető, hogy 3,3 V feszültségen az adatlapinál alacsonyabb teljesítményszinten, de a linearitás határán hatékonyabban működik a tranzisztor, mint a hivatalos 5 V tápfeszültségen.

Az illesztés finomhangolására használtam az AWR Microwave Office mikrohullámú szimulátorprogramot. Először a fejlesztő panel eredményeit próbáltam reprodukálni, tehát olyan lezárást kerestem, ahol az erősítés kiadja a mért értéket, és nagyobb reflexió sincs. Ez sikerült is. Minden eszköznek a gyártótól beszerzett S-paramétereit használtam. A legpontosabb eredményt a kvázi-stacionárius terekkel számoló koplanáris hullámvezető elemekkel kaptam (3.20. ábra, 3.21. ábra). Az illesztőhálózat elemeit úgy hangoltam, hogy a hivatalos kapacitásmodellek mellé párhuzamosan kötöttem egy ideális kondenzátort is, és ennek az értékét állítottam (5.6. ábra, 5.7. ábra).

A végleges illesztőértékek bizonyos mértékben eltértek a szimuláltaktól. Ezt több dolog okozhatta. Az első, hogy a megadott S-paraméterek 5 V tápfeszültségre voltak megadva, nem pedig 3,3 V-ra. A második pedig, hogy az S-paraméterek a lineáris kisjelű erősítés számolására megfelelőek, nem pedig a teljesítményerősítésére, ahol az eszköz a lineáris működési tartomány határain üzemel. Erre a tartományra már más paraméterek szükségesek. A harmadik pedig, hogy az AWR kvázi-stacionárius terekkel számoló koplanáris

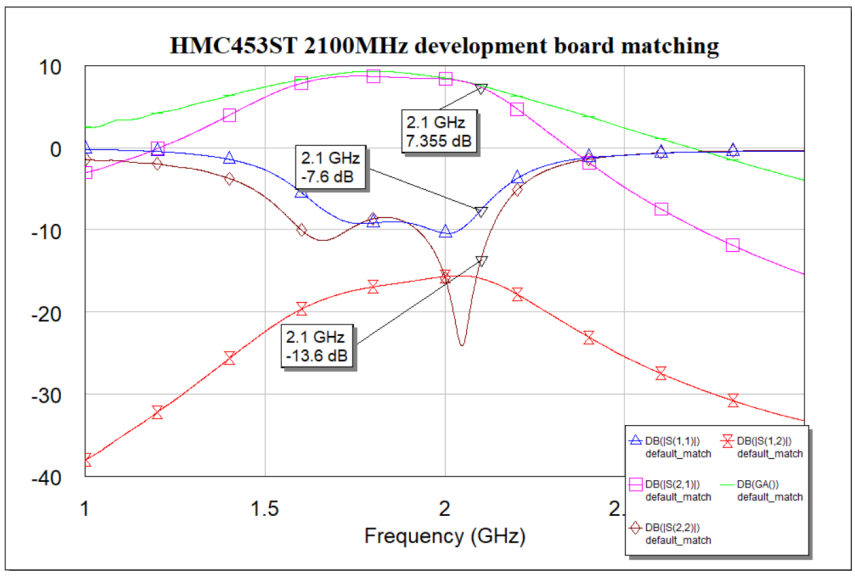


3.18. ábra. HMC453ST saját tervezésű tesztpanel

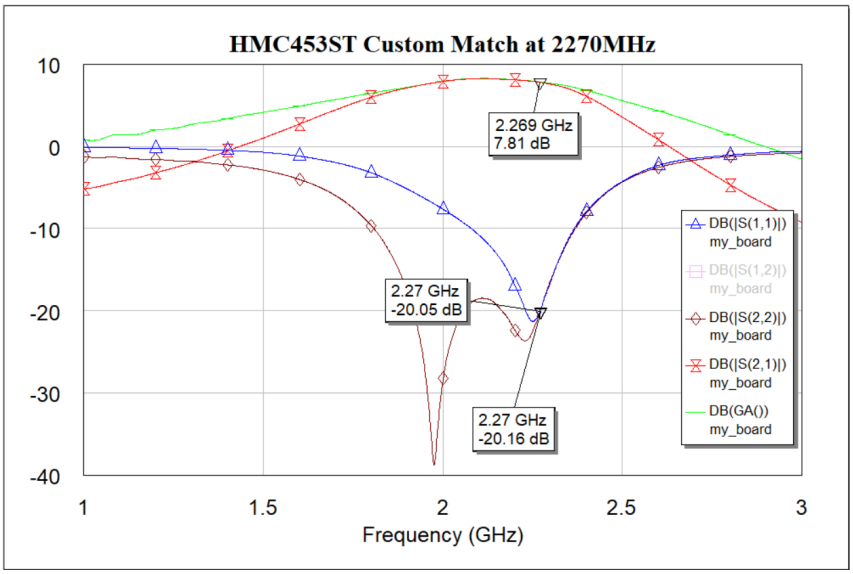


3.19. ábra. HMC453ST kiillesztett kimeneti teljesítménymérés, EFR32MG24 jelforrás

hullámvezető elemei nem voltak elég pontosak, és CST programban végeselem szimulációra lett volna szükség. Tehát a szimulációs módszer további pontosításra szorul, de konzulensi tanácsra a fejlesztés tempója miatt erre nem maradt idő.



3.20. ábra. HMC453ST fejlesztőpanel AWR szimulációja



3.21. ábra. HMC453ST saját tervezésű tesztpanel AWR szimulációja

3.4. Az antenna

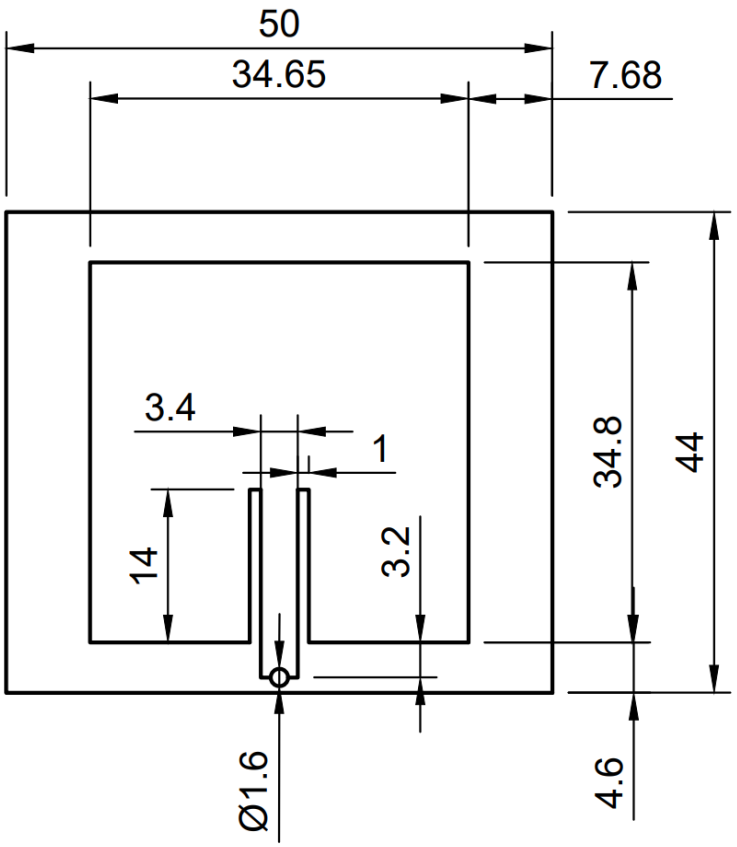
3.4.1. Az antenna tervezése

Egy rádióadónak a kimenetére szükséges egy antenna, hogy megfelelő hatékonysággal tudjon sugározni. Mivel nem elérhetőek S-sávra tervezett, ekkora méretezésű antennák, ezért a feladatom része volt, hogy tervezzek és legyártsak egyet.

A koncepció egy patch antenna, mely az egyik oldalpanelre lesz rögzítve. Ez szigorúan 50x44 milliméterben szabja meg a maximális méretét.

A rezonáns lineáris patch antennát Rogers4003 hordozóra terveztem, mert sokkal kisebb a dielektrikum vesztesége, mint a standard FR4 anyagoknak, és ez volt elérhető számomra.

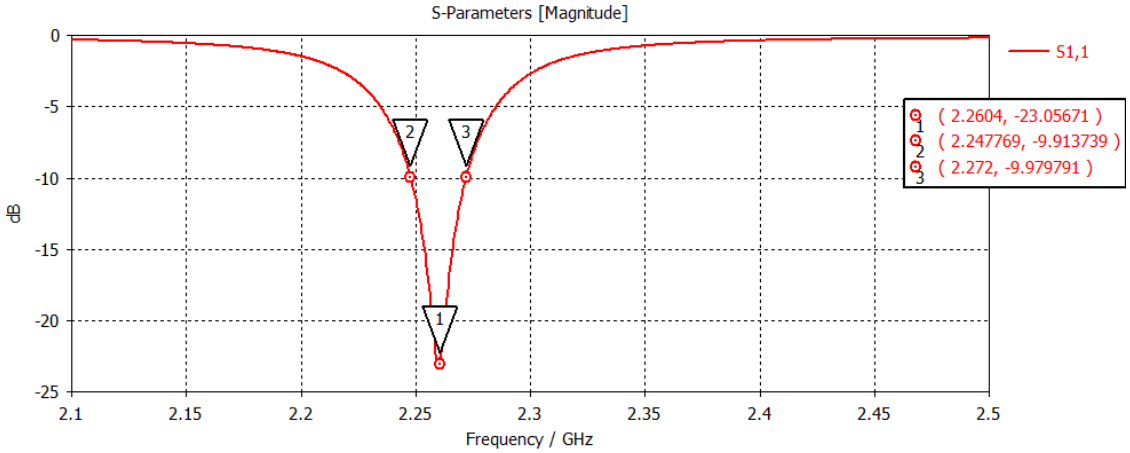
A tervezést először Sonnet szimulátor programban kezdtem, de később a jobb felhasználói kezelőfelület és a pontosabb megoldó miatt CST Studio Suite-tal fejeztem be a



3.22. ábra. Patch antenna műszaki rajz (méretek milliméterben)

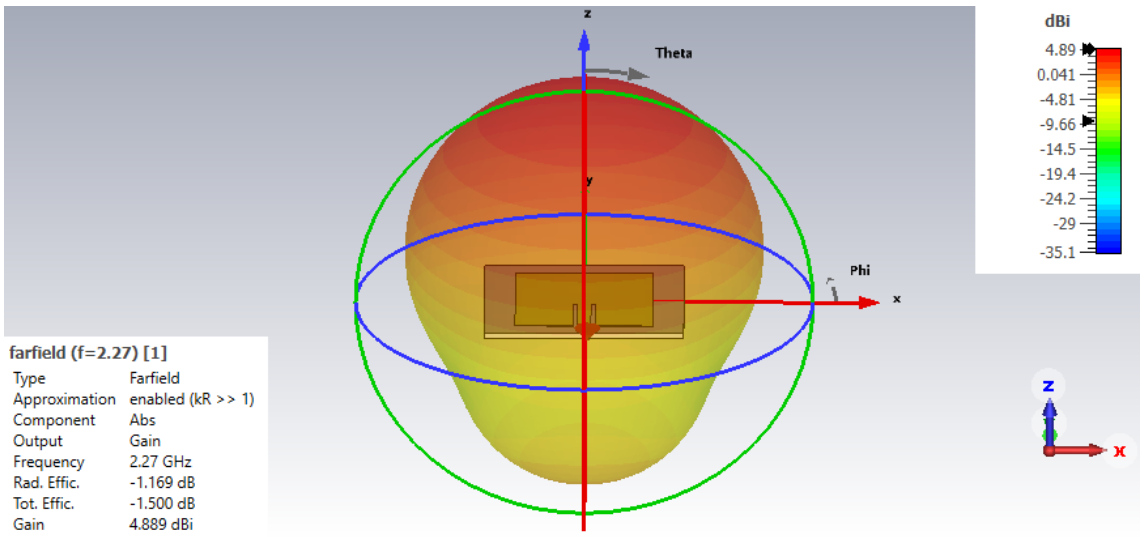
tervezést.

Közelítő értékként ezekben a tanulmányokban vett értékeket használtam [3][4], és utána a paraméterek pontos értékét a CST Optimizer és Parametric Sweep funkciójával számoltam ki.

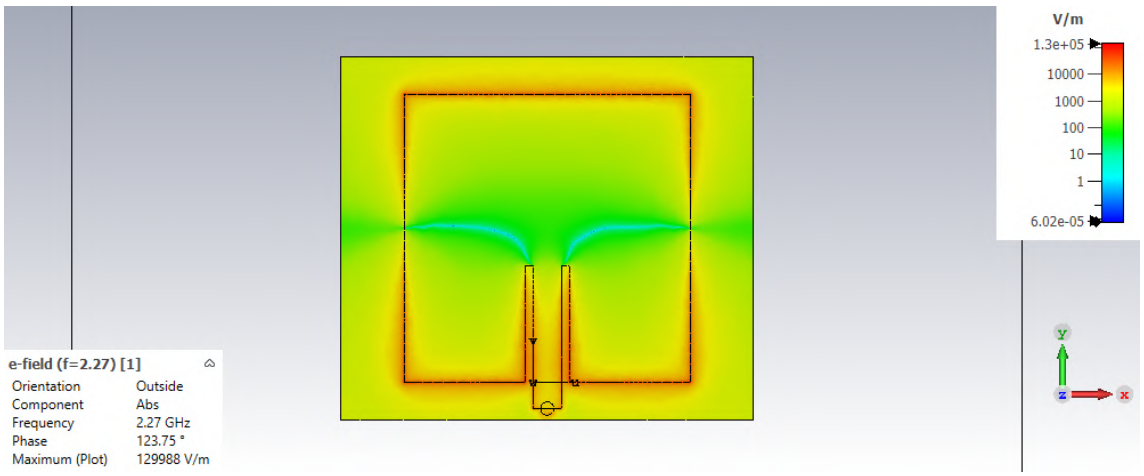


3.23. ábra. A patch antenna S11 paraméter szimulációja

A szimulált értékek tökéletesen lefedik az S-sáv elegendő részét(B=25 MHz). A kis sávszélesség valószínűleg a kis földsíknak köszönhető. Rezonancián <-20 dB leszívás és 4.9 dBi nyereség megfelel az elvárásoknak.



3.24. ábra. A patch antenna távoltér szimulációja

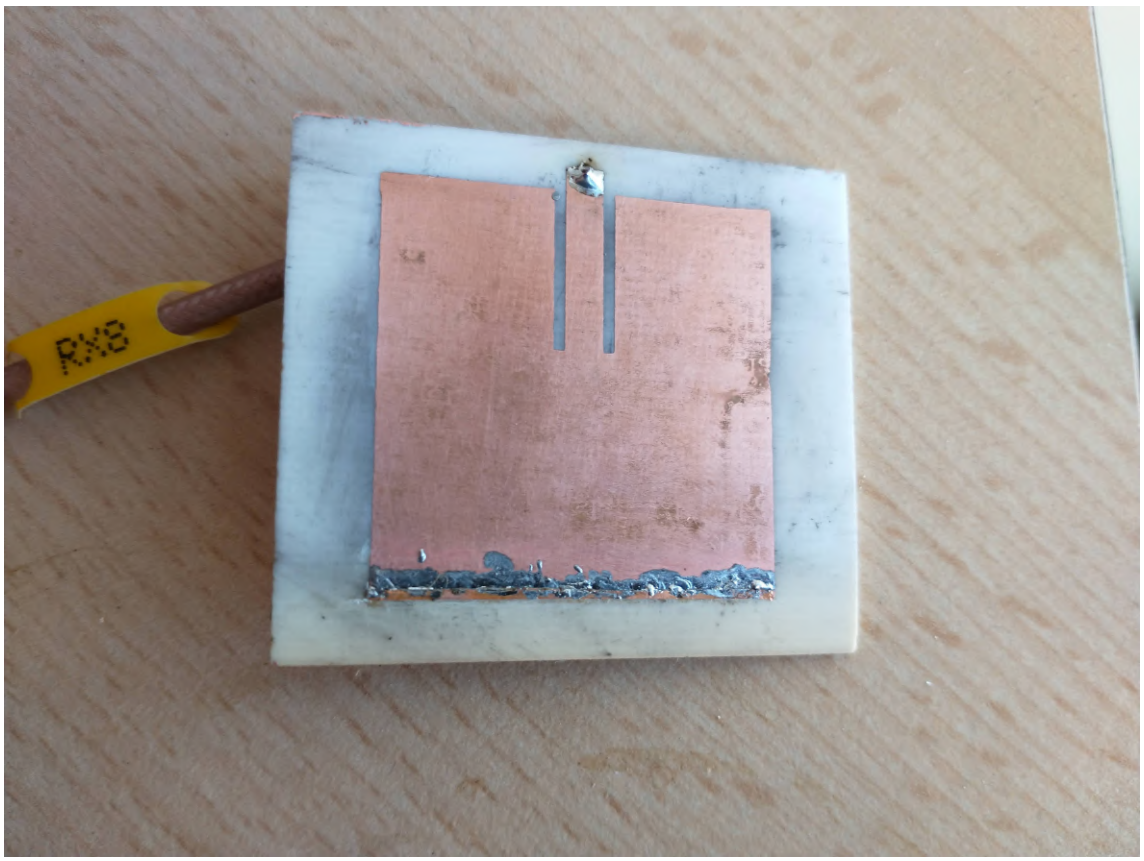


3.25. ábra. A patch antenna elektromos térszimulációjából pillanatkép

3.4.2. Az antenna megvalósítása

Az antennát a laboratóriumban gyártottam le. Először megtisztítottam a hordozót acetonnal, majd a patch kontúrját lézernyomtatóval megfelelő papírra nyomtattam. Ezután a papírról vasalóval ráégettem a hordozóra a kontúrt. A papírt vízzel eltávolítottam, és a hordozóról sósav, hidrogén-peroxid és víz 1:1:1 arányú keverékével lemarattam a felesleges rezet. A tápvonal furat pedig állványos fúrógép használatával készült.

A 3.26. ábrán az első tesztpéldány látható, kalibrációs célból már rá van forrasztva az antennára egy rézszalag, hogy lejjebb vigye a rezonanciafrekvenciáját. Az itt eszközölt változtatásokat a 3.22. ábrán látható műszaki rajz is tükrözi.



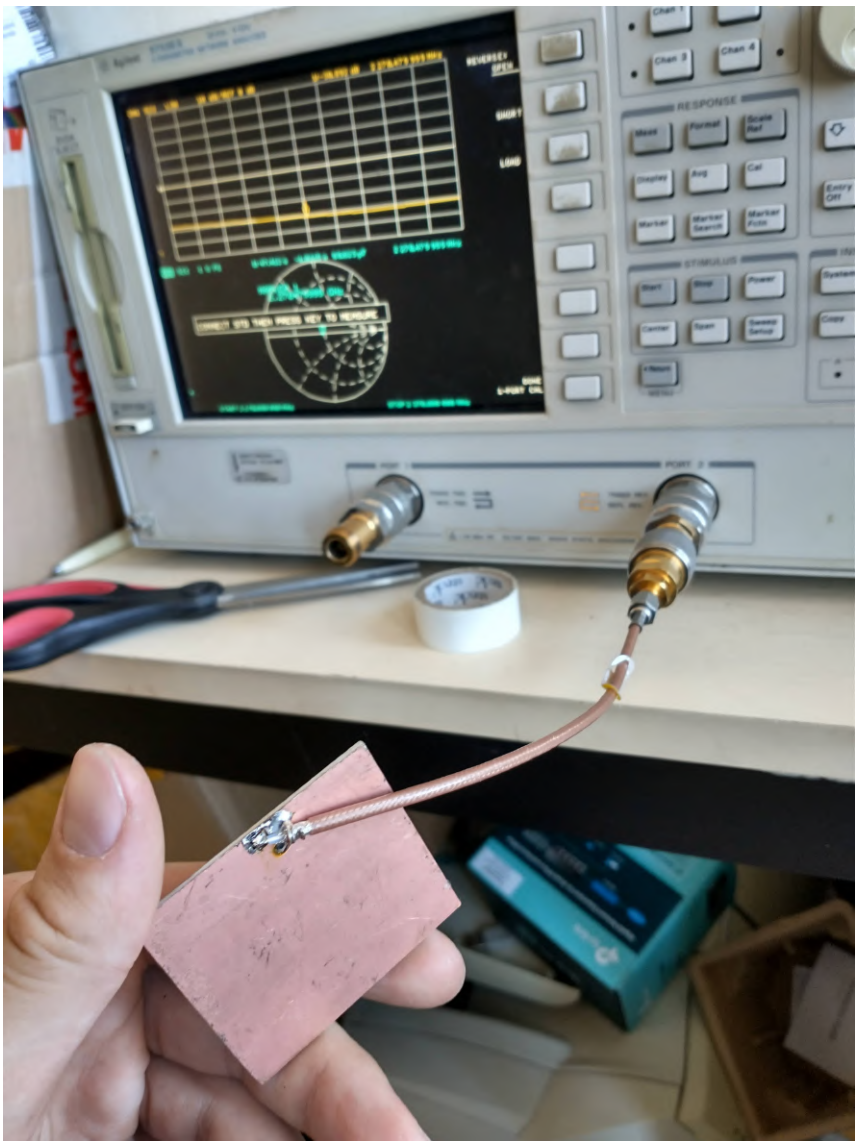
3.26. ábra. A patch antenna legyártva

3.4.3. Az antenna mérése

Az antenna mérése hálózat analizátorral

Az antenna S11 paraméterét először egy Agilent 8753ES hálózat analizátorral vizsgáltam. Ehhez a 3.27. ábrán is látható kábel végére kalibráltam a műszert. A műszer kalibrálásához szükség van egy rövidzárra, szakadásra és 50 Ohm lezárásra. Ezeket a kábel végére forrasztott különböző ellenállásokkal értem el. Ekkora frekvencián még megfelelően szélessávú egy standard 0603 méretű SMD ellenállás, tehát kalibrálásra pont megfelel.

A 3.28. ábrán látszik, hogy a rezonanciafrekvencia a kívánt sávban van, a leszívás mértéke tökéletes. A végeselem szimulációval korrelál.

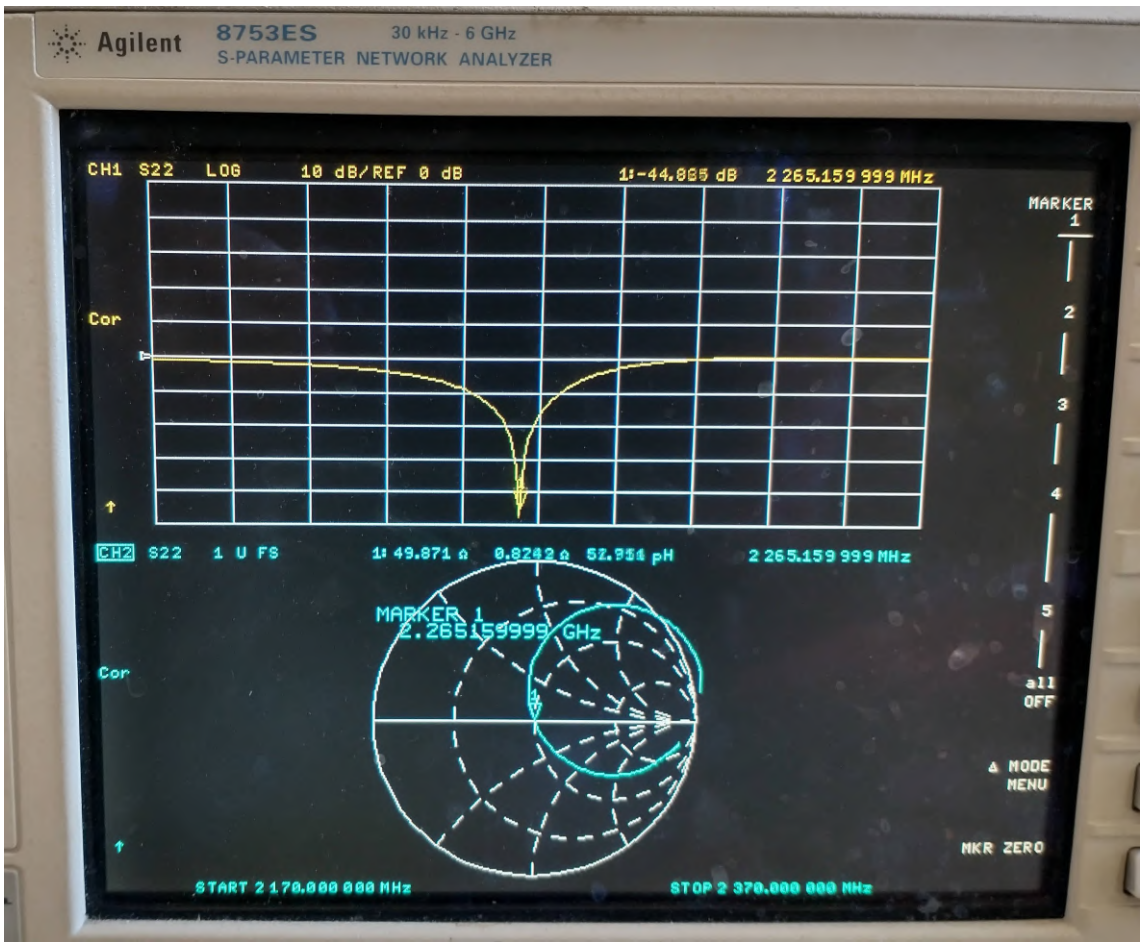


3.27. ábra. VNA mérés

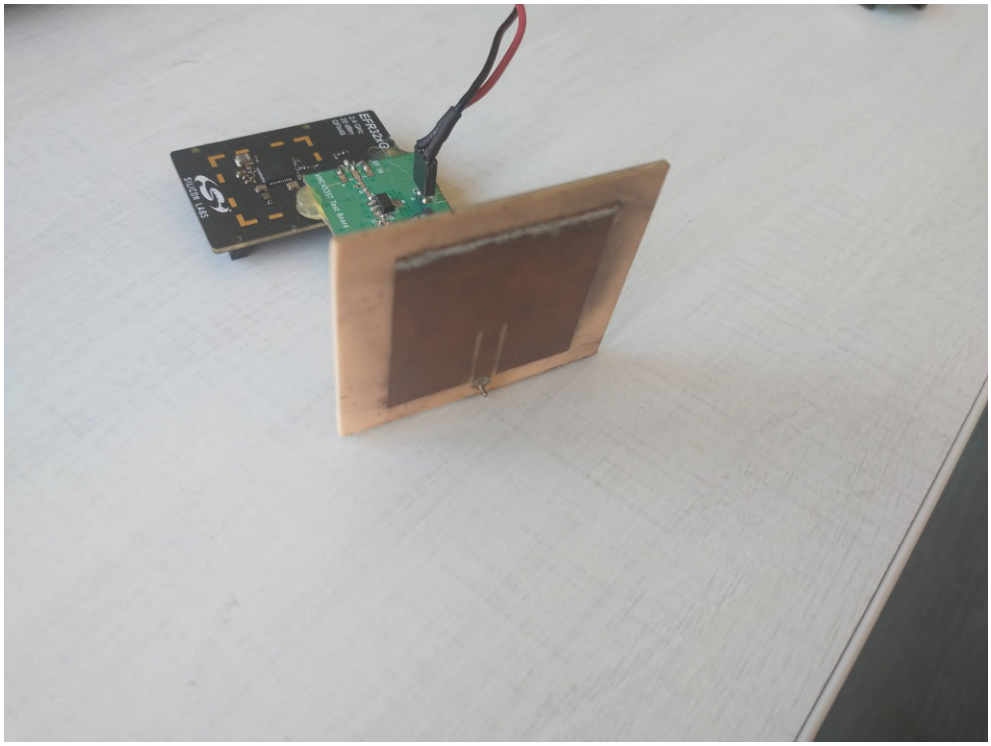
Antenna mérése reflexiómentesített mérőszobában

Az antenna tesztelése úgy zajlott, hogy a fentebb említett BRD4187C fejlesztőpanelre csatlakozott a HMC453ST tesztpanel, melynek a kimenetére csatlakozott az antenna SMA csatlakozóval (3.29. ábra). Ez gyakorlatilag maga a teljes rádió, csak külön modulokból összerakva. Ezt az antennamérő rádiót a 3.30. ábrán látható mérőszobában mértem meg. Az itt látható forgatópad (3.31. ábra) 360 fokban megforgatja az eszközt, és egy tölcsérantennával megméri a maximális teljesítményt, amelyet érzékelt. Ezt a vezérlő szoftver a szoba átvitelére kalibrált adatok alapján átszámolja EIRP(Effective Isotropic Radiated Power) teljesítménybe.

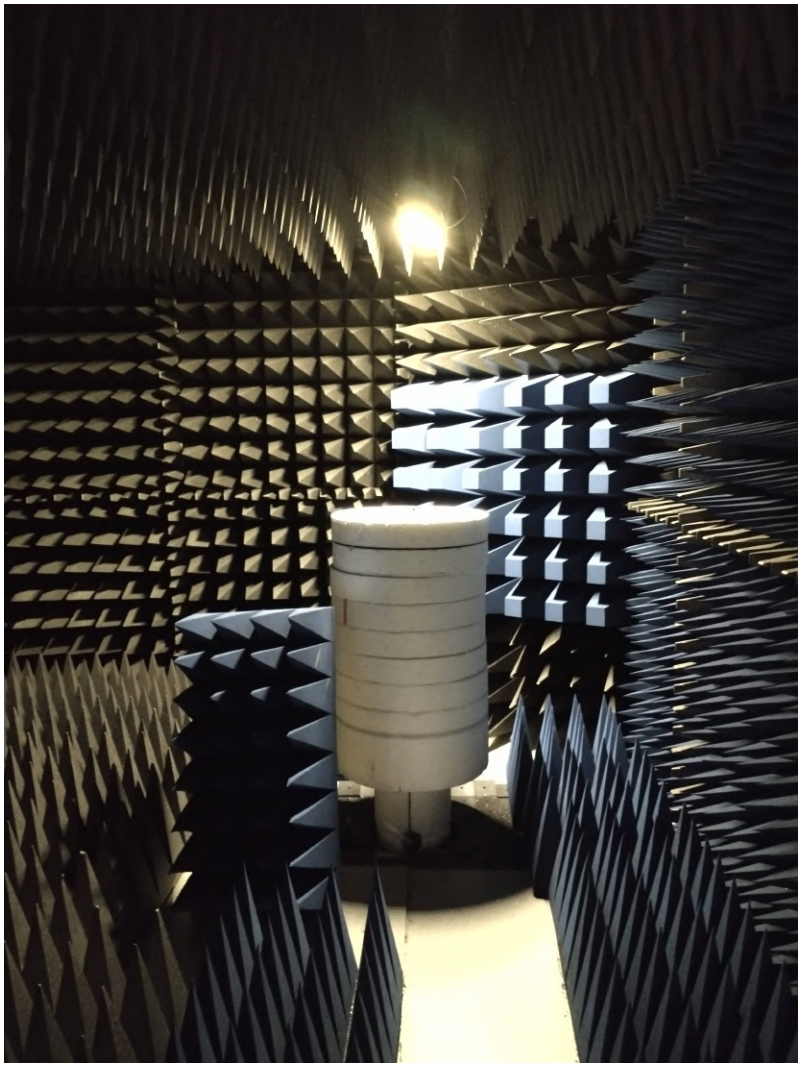
A mérőszobában mért teljesítmény 2270 MHz-en: 30.76 dBm. Ez 3.76 dBi nyereséget jelent. Ez körülbelül 1 dB-vel rosszabb, mint a szimuláció. Viszont végeztem másik sugárzott teljesítménymérést is, amely jobb eredményt adott.



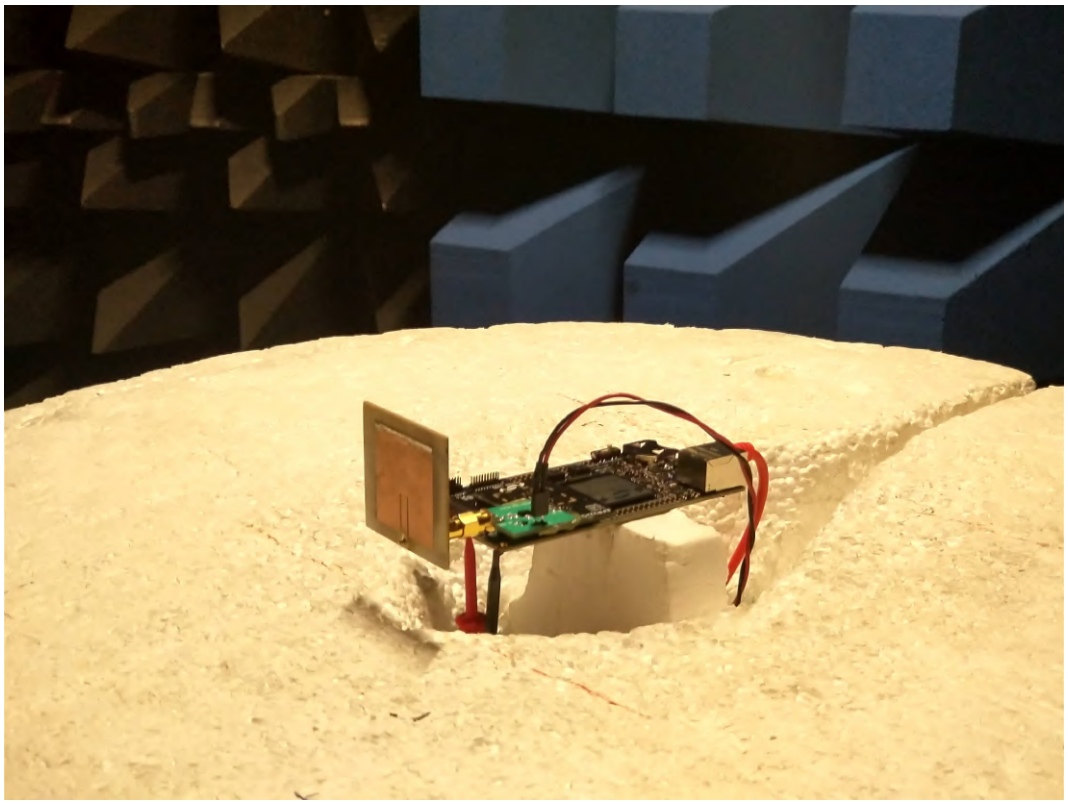
3.28. ábra. VNA mérési eredmény



3.29. ábra. A teljes rádió prototípus, modulokból összerakva



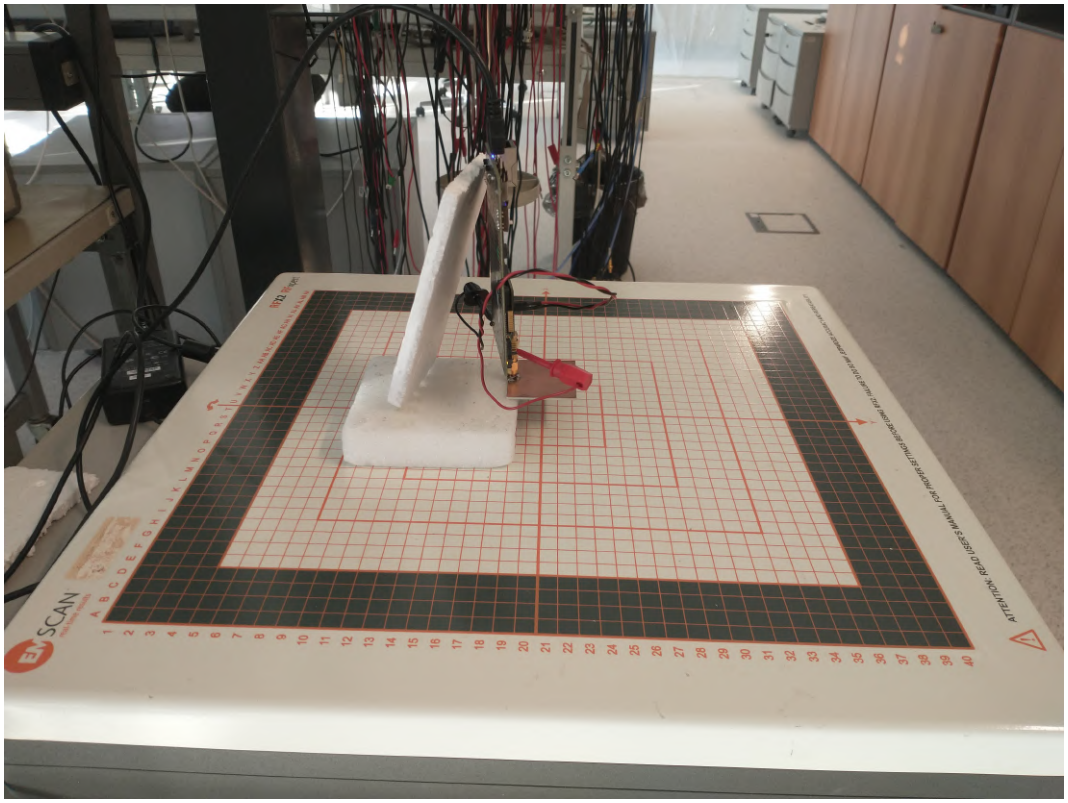
3.30. ábra. Az antenna mérőszoba



3.31. ábra. Az antenna forgatópad, rajta a mérendő eszközzel

Antenna mérése közeltérmérővel

Kaptam hozzáférést egy speciális antennamátrixszal működő közeltérmérő berendezéshez is. Ennek a neve RFExpert, és a 3.32. ábrán látható.



3.32. ábra. A közeltérmérő berendezés, RFExpert

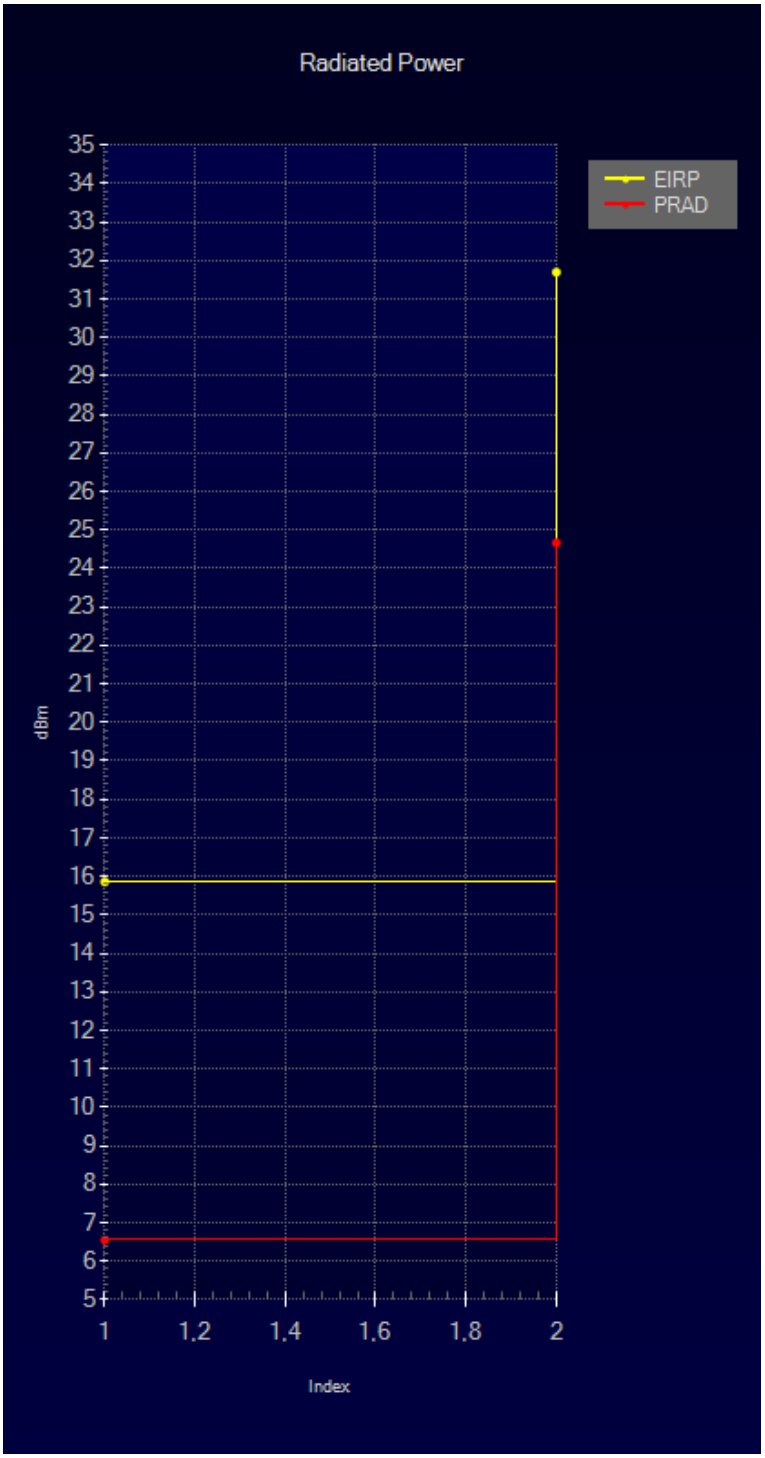
Ezt a műszert úgy kell használni, hogy az antennamátrix felé irányozzuk az antennát (3.32. ábra), megadjuk a sugárzó távolságát a műszervezérlőnek, és elindítjuk a mérést. A műszer megméri minden egyes antenna egységével a sugárzott teljesítményt, és ebből a közeltéri jelerősség mátrixból számol távoltéri paramétereket.

A 3.33. ábrán látszik (index=2), hogy a sugárzott teljesítmény 31.7 dBm EIRP, ami 4.7 dBi nyereséget jelent. Ez szinte egy az egyben megegyezik a szimulált értékkel.

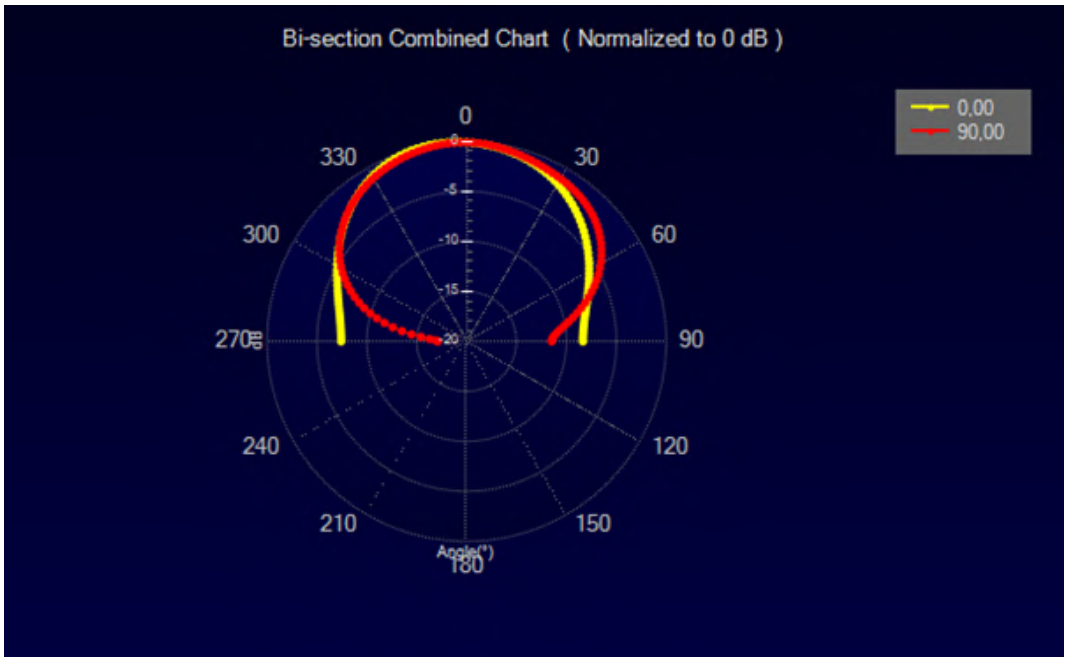
Antennamérések összegzése

Habár volt szükség kalibrálásra, a CST szimuláció kifejezetten pontosan kiadta az antenna rezonanciafrekvenciáját. Ez valószínűleg annak is köszönhető, hogy a Rogers anyagok dielektromos állandója nagyon kontrollált és pontos, nem változik lapról-lapra, mint a különböző FR4 hordozóknál.

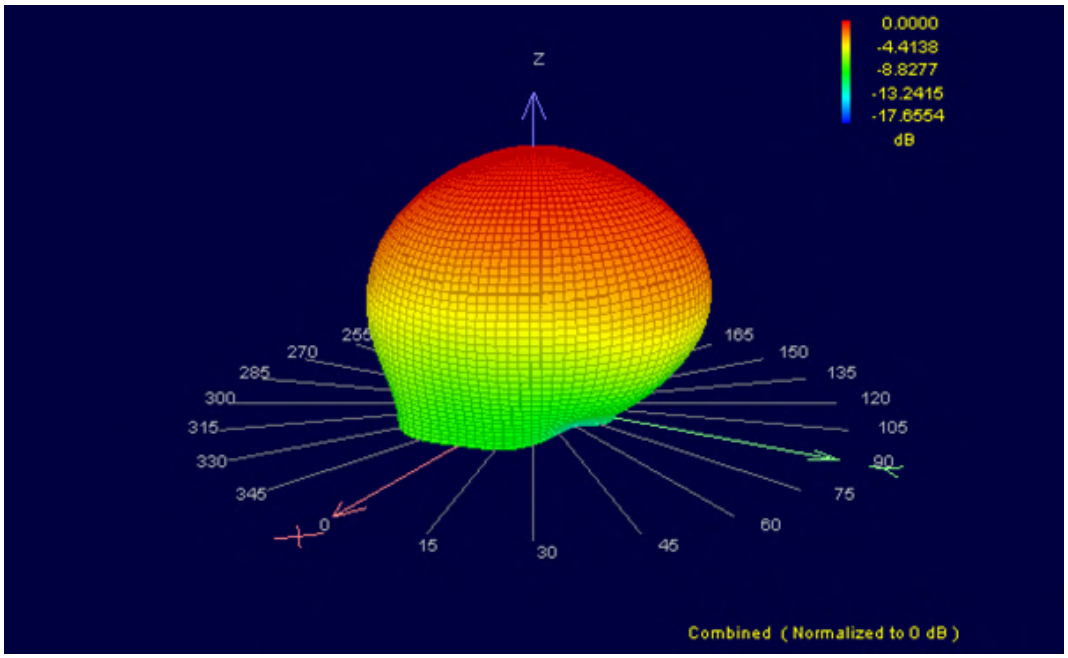
A mért nyereségben való különbség érdekes kérdés, de az biztos, hogy az antenna mérőszoba határaiban benne van az 1 dB eltérés. A közeltérmérő nyereség eredményeit is nehéz validálni, abba is belefér a pontatlanság, mivel nem távoltérmérés történik. Az igazság valahol a kettő között rejlik, valószínűleg 4-4.5 dBi környékén.



3.33. ábra. RFExpert-tel mért EIRP



3.34. ábra. RFExpert-tel mért, 4.8 dBi-re normalizált iránykarakterisztika 2D-ban



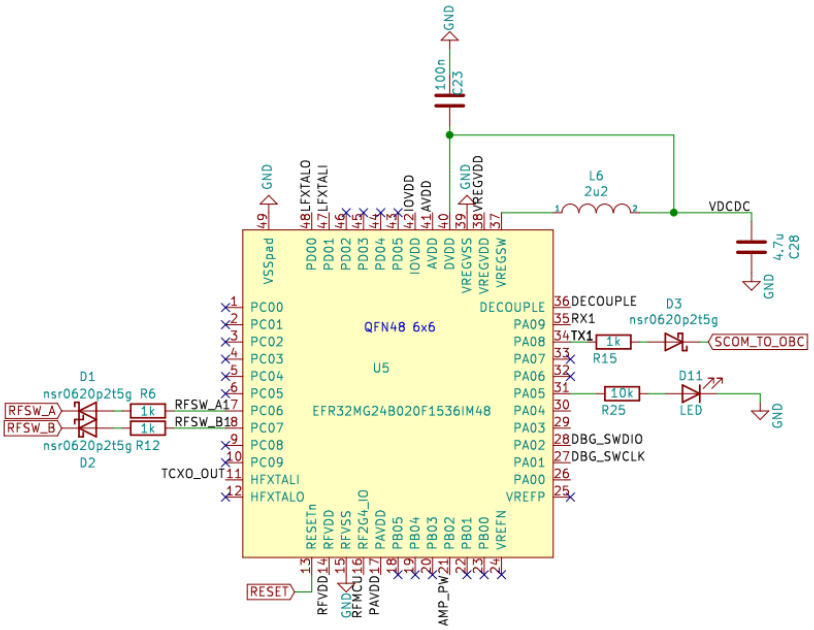
3.35. ábra. RFExpert-tel mért, 4.8 dBi-re normalizált iránykarakterisztika 3D-ban

3.5. Az MRC100 S-sávú adó kapcsolása

Az előbbiekben tárgyalt mérések miatt a választás az EFR32MG24 + HMC453ST szintézer-végfok kombinációra esett. A megtervezett áramkör működését az alábbi szekcióban részletezem. Az egész áramkör kapcsolása: 5.3. ábra és 5.4. ábra .

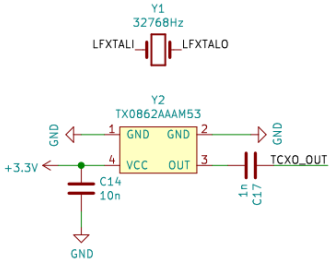
Az RFMCU és perifériák

A rádiós mikrovezérlő úgy van konfigurálva, hogy használja a belső DC-DC konverterét, mely alacsonyabb fogyasztást eredményez (3.36. ábra).



3.36. ábra. EFR32MG24 belső DC-DC konverter használatra konfigurálva

Az alacsony frekvenciájú órajelet egy kvarckristály szolgáltatja, a magasat pedig egy Temperature Compensated Oscillator (TCXO). A TCXO azért fontos, mivel a műhold nagy hőmérsékletingadozásnak lesz kitéve, és nem lenne előnyös, ha (a Doppler-effektus mellé még) jelentős frekvenciahiba jelenne meg a rádiójelen emiatt (3.37. ábra).

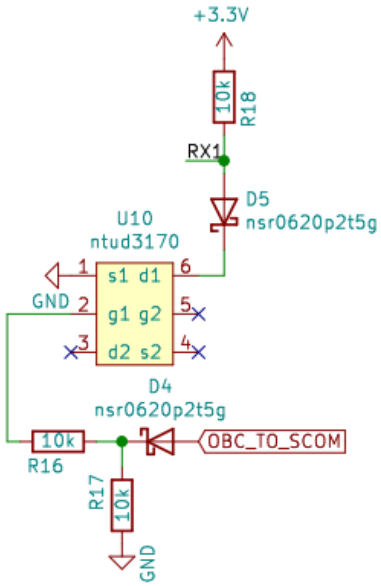


3.37. ábra. Referencia órajelek

A panelen két adóáramkör helyezkedik el (hideg-kapcsolt redundancia céljából). Csak egy soros interfész van a panelra kivezetve, ezért a mikrovezérlők soros portjait le kell vá-

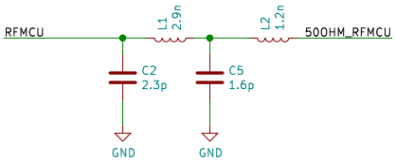
lasztani azért, hogy ne legyenek problémák a közös busz miatt. A TX vonal egy Schottky-diódával (3.36. ábra), az RX vonal pedig egy MOSFET-tel van leválasztva(3.38).

A mikrovezérlő felől a kikapcsolt soros interfész diódája magas impedanciát mutat a bekapcsolt interfész irányába. Az OBC irányából pedig a MOSFET Gate látszik, egy ellenállásosztón keresztül, amely szintén nagy impedanciát mutat, függetlenül attól, hogy a soros busz RX oldala épp ki- vagy bekapcsolva van a túloldalán.



3.38. ábra. RX leválasztó áramkör

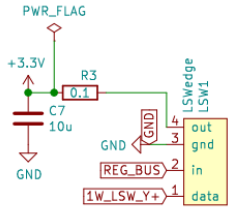
Az RFMCU kimenetén az adatlapi 2.4GHz-es illesztőhálózat (3.39. ábra) található meg [7]. Ez egy viszonylag szélessávú hálózat. Ezt mutatja az is, hogy a 2270 MHz-es mérésnél szintén képes volt a 20 dBm kimeneti teljesítményre, annak ellenére, hogy körülbelül 170 MHz-el eltér a sávközéptől, amelyre az eredeti illesztőhálózat optimalizálva volt [11].



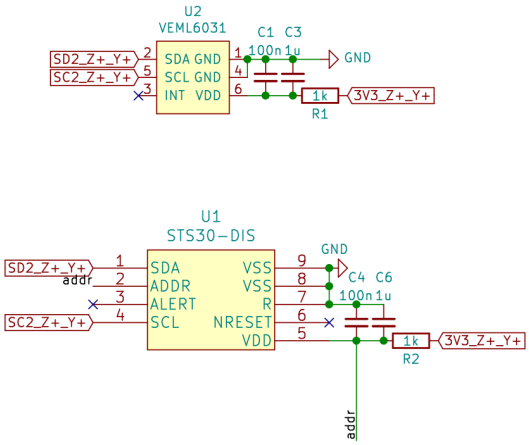
3.39. ábra. RFMCU kimeneti illesztés

A két adóáramkört külön egy-egy áramlimiter kapcsoló (3.40. ábra) köti a műhold energiabuszához, nehogy egy zárlat esetén az STX modul lehúzza az egész buszt földpotenciálra. Ezekkel az OBC (On-Board Computer) tudja monitorozni az adók áramfelvételét és váltani a redundánsra vagy kikapcsolni őket, hogyha rendellenes értéket mér.

A kapcsoláson még megtalálható egy hőmérő (STS30-DIS) és egy fényszenzor (VEML6031) is, amely a műholdas alkalmazásnál a napszenzor szerepét tölti be (3.41. ábra).



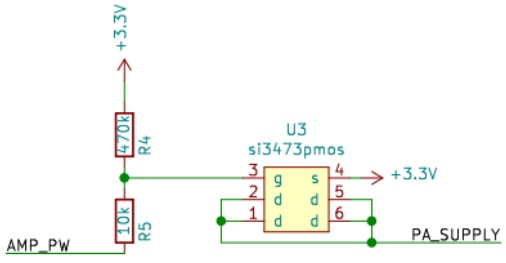
3.40. ábra. Limiter kapcsoló



3.41. ábra. A hőmérő és a napszenzor

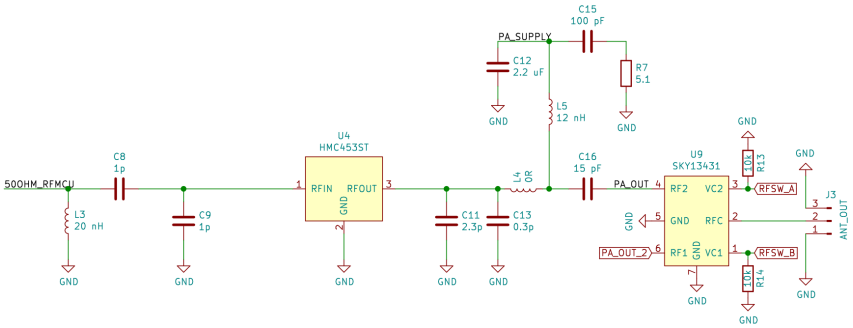
3.5.1. Az erősítő

Az erősítőt egy tranzisztor fogja pontosan akkor kapcsolni, amikor adás történik (3.42. ábra), mivel bemeneti jel nélkül körülbelül 370 mA-t fogyaszt.



3.42. ábra. Az erősítőt kapcsoló tranzisztor

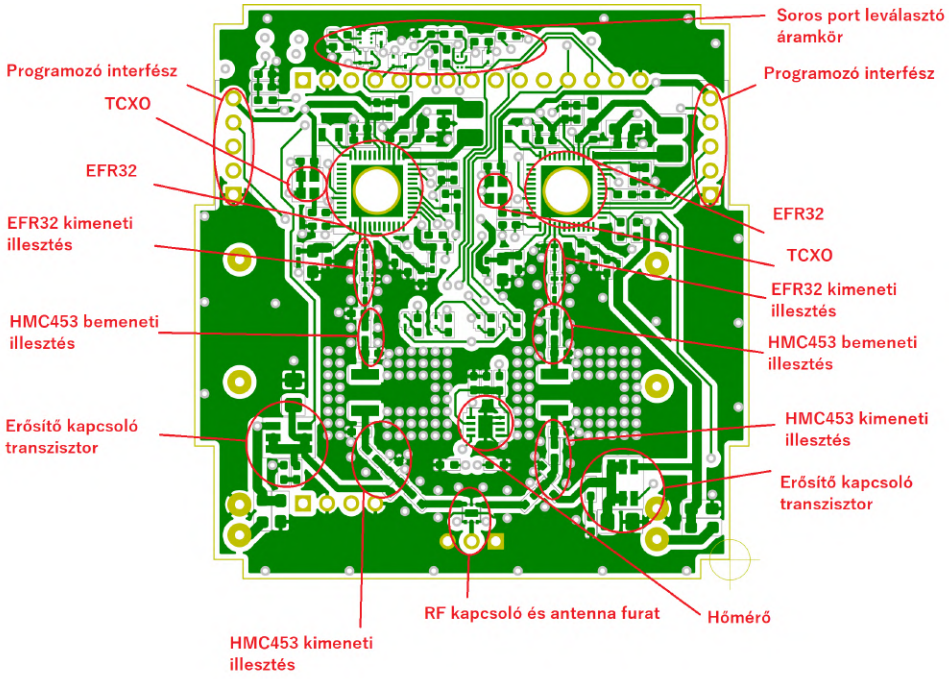
Az erősítő be- és kimenetén pedig az előző szekcióban tárgyalt illesztés található. Mivel hidegkapcsolva van mellette egy teljesen független adóáramkör, de antenna csak egy darab van, szükség van egy RF kapcsolóra is.



3.43. ábra. A végfok illesztés és RF kapcsoló

3.6. Az MRC100 S-sávú adó nyomtatott huzalozású lemezterve

Az áramkör egy négy rétegű FR4 anyagú lemezen valósítottam meg. A terv KiCAD szoftverrel készült.



3.44. ábra. A panel fontosabb részei

3.6.1. RF Tervezési megfontolások

Egy rádiós panel tervezésekor rengeteg irányelvet figyelembe kell venni, hogy a végtermék megfelelő sugárzási paraméterekkel rendelkezzen, és ne keletkezzenek nem kívánatos frekvenciakomponensek (gerjedés, harmonikus sugárzás) [10].

Néhány fontos irányelv, amely alapján a terv készült:

- Az illesztőhálózat alatt nem fut semmiféle jelvezeték (3.48. ábra)

- Az illesztőhálózat alatti rétegen (3.48. ábra) megszakítatlan földkitöltés van
- Az összes szűrőkondenzátor és a TCXO direkt fémezett furatokkal van földre kötve (3.46. ábra)
- Az RFMCU föld lába és az illesztés földje nincs megszakítva
- Az illesztés kondenzátorai direkt fémezett furatokkal vannak földre kötve
- Mivel az illesztőhálózat nem fed le nagy távolságot, ezért a tápvonal vastagsága pont az adott soros illesztő alkatrész vastagságára van méretezve (3.46. ábra)
- Szűrőkondenzátorok lehetőleg minél közelebb vannak az RFMCU megfelelő lábaihoz (3.46. ábra)

Az áramkörön három illesztőhálózat található:

- RFMCU kimeneti illesztés: hivatalos 2.4 GHz-re optimalizált illesztőhálózat 50 Ohm-ra 0201 SMD méretű alkatrészekből összeállítva, mérések alapján kifejezetten szélessávú (3.9. ábra), mivel hozza a 20 dBm kimeneti teljesítményt S-sávban is.
- Végfok erősítő bemeneti és kimeneti illesztés: a 3.3.5 alszekcióban tárgyalt illesztés kétrétegű panelre lett optimalizálva. Viszont a tervezett STX lemez 4 rétegű, tehát teljesen más parazita értékek fognak érvényesülni. Ezért ez az áramkör hangolást igényelt.

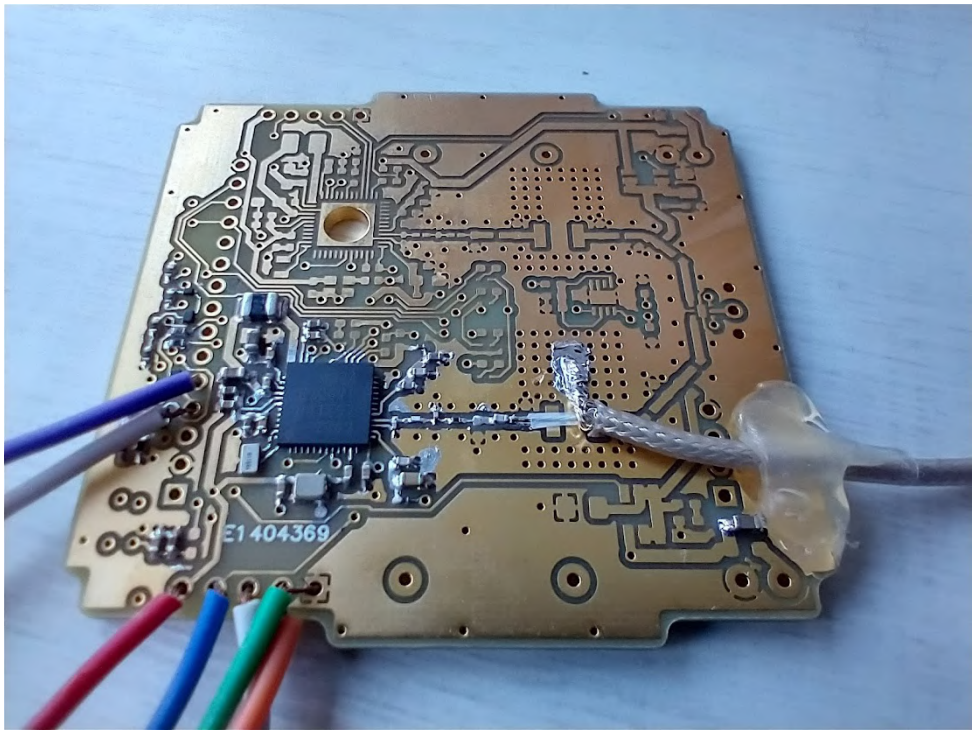
Az illesztőhálózatok hangolása

Az adópanel EFR32 illesztőhálózata a referencia áramkörhöz hasonlóan 0201 méretű alkatrészekből készült. Így eredeti értékekkel is hozta az elvárt kimeneti teljesítményt, mivel a különböző parazita impedanciák megegyeztek az S-sávú adópanelen és a fejlesztőkártyán [9]. Ettől függetlenül egy egyszerűsített hálózatot is kipróbáltam, ahol az L2 tekercset (3.4) rövidre zártam, mert egy induktivitás beiktatási csillapítása pár tized decibel csökkenést okoz a kimeneti teljesítményben. Természetesen a keletkezett T-hálózat ugyanazokat az elméleti ki- és bemeneti impedanciákat hozta, mint az eredeti π-hálózat.

Ez a változtatás nem hozott mérhető különbséget a kimeneti teljesítményben (3.45. ábra), ezért végül az eredetit ültettem be.

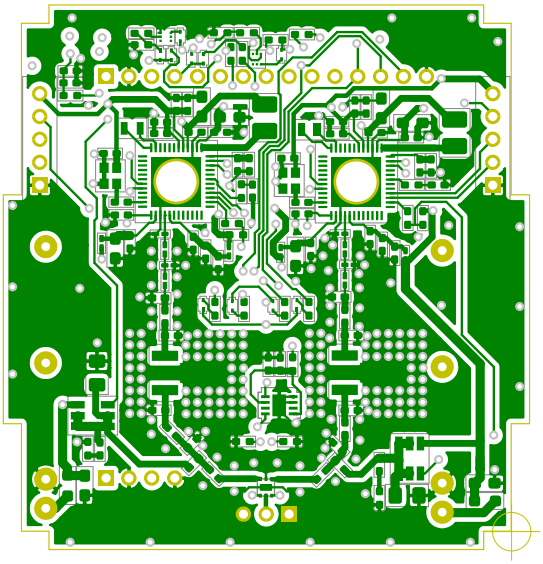
Az erősítő illesztőhálózata viszont hangolásra szorult a rétegszám változása miatt. Miután a 3.3.5 alszekcióban tervezett szimulációs modell nem hozott elég pontos eredményeket, a szimuláció pontosítása helyett empirikusan, iterálva végeztem a hangolást.

Először egy teljesen beültetett panelen, antenna nélkül, vezetett teljesítménymérés alapján illesztettem ki az erősítőt. Ezek után már a felragasztott antennával, szintén vezetetten még egyszer hangoltam az illesztésen. A következő hangolást már az erősítő kimenetét antennára kötve, sugárzott mérésekre alapozva végeztem. Az utolsó iterációt pedig már a műhold vázába helyezett panellel mértem (3.55. ábra), mivel a váz jelenléte is eltolta annyira az antenna impedanciáját, hogy érdemes volt tovább hangolni az illesztést. Természetesen ideálisabb lett volna, hogyha az antenna prototípus fázisában panellal a hátulján és műholdvázzal együtt van már eleve hangolva, viszont ezek még nem álltak rendelkezésre a tervezés és az illesztés korai fázisaiban0.

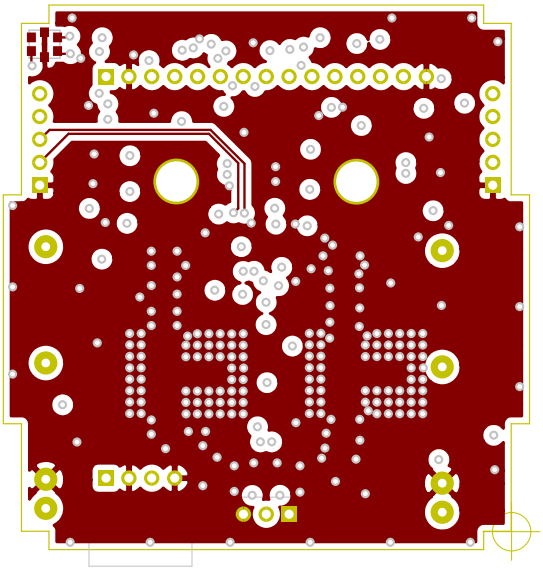


3.45. ábra. Az EFR32 vezetett teljesítménymérése az STX panelen

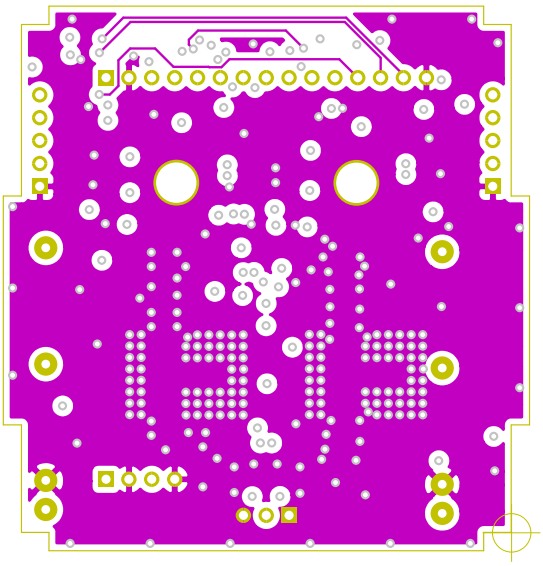
Az erősítő illesztésének hangolása hosszadalmas folyamat volt, amelyet nehezített egy hiba a panelen. Ez pedig a DC leválasztó kondenzátor hiánya volt az RF kapcsoló kimenetén, amely rossz következtetésekhez vezetett az antenna impedanciájával kapcsolatban. A hibát a végső példányokon javítottam.



3.46. ábra. A nyomtatott huzalozású lemez alkatrész oldala



3.47. ábra. A nyomtatott huzalozású lemez antenna felőli oldala



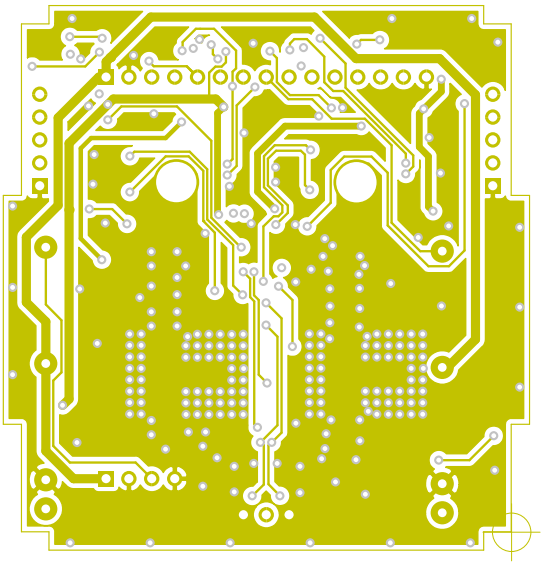
3.48. ábra. A nyomtatott huzalozású lemez alkatrész oldali belső rétege

Termikus megfontolások

Az atmoszféra hiánya miatt az űrben egy eszköz kétféleképpen tudja leadni a termelt hőt.

- Hővezetéssel, tehát átadja a hőenergiát azoknak a szomszéd alkatrészeknek, amelyek vele termikus kontaktusban vannak és hidegebbek.
- Hősugárzással, tehát elektromágneses sugárzásként adja le a hőenergiát, ezen hő-

mérséklettartományon legfőképp az infravörös spektrumban.



3.49. ábra. A nyomtatott huzalozású lemez antenna oldali belső rétege

Az egész műholdon a legnagyobb hőtermelő alkatrész az S-sávú erősítő (3.2. egyenlet). Ez okból is előnyös választás a HMC453ST, amely a tokozása miatt hatékonyan képes hőt leadni. Mivel a hősugárzás nem a leggyorsabb hűtési módszer, nem lehet rá számítani ekkora hőteljesítménynél, tehát hővezetéssel kell eltávolítani a kritikus helyről az energiát. Ezért az erősítő földlába mellett raszteresen elhelyezett fémezett furatok biztosítják a hőelvezetési funkciót (3.46. ábra). Ezek környékén megszakítatlan földréteg van, mely szolgáltatja az elnyelő hőkapacitást az erősítőnek, így az működés közben is üzemi hőmérsékleten tud maradni, nem hevül túl. Ha pedig mégis ez lenne a helyzet, a mellette elhelyezett hőmérő ezt jelezni fogja, és lejjebb vehető a teljesítmény.

$$P_{AE} = \frac{P_{RFin} - P_{RFout}}{P_{DCamp}} = 33\% \tag{3.1}$$

$$P_T = P_{DCamp}(1 - P_{AE}) = 821\,mW \tag{3.2}$$

P_{AE} : Power Added Efficiency, erősítő hozzáadott teljesítményének hatásfoka [%]
P_T : Hőteljesítmény [mW]
P_{RFin} : Erősítő bemenő RF teljesítmény: 100 mW
P_{RFout} : Erősítő kimenő RF teljesítmény: 500 mW
P_{DCamp} : Erősítő teljesítménye: 1221 mW

3.6.2. Szerkezeti megfontolások

Egy űreszköz sok speciális elvárásnak kell, hogy megfeleljen. Lényeges dolog, hogy a rázótesztet károk nélkül viselje el. A rázóteszt az adott rakéta (jelen esetben SpaceX

Falcon-9) teljes profiljával való rázópados tesztet jelenti, ahol az eszköz akár 60 G-s ütéseket is el kell hogy viseljen. Elengedhetetlen, hogy az alkatrészek vákuumban ne engedjenek ki magukból nemkívánatos gáz halmazállapotú anyagokat, azaz ne legyen kigázosodás. Az előbb leírt két kitételt pár irányelv betartásával lehet teljesíteni:

- Speciális forrasztómaszk használata (amely csak minimálisan gázosodik), vagy költségvetés kímélőbb megoldás a forrasztómaszk teljes elhagyása (ez megkönnyíti a tesztelést és a hibakeresést is).
- Olyan alkatrészek választása, melyek stabilan beforraszthatóak (előnyös ilyen szempontból például a QFN tokozás).
- Minimális folyasztószer használata forrasztáskor (a maradványok kigázosodási problémákhoz vezethetnek), izopropil-alkohol mosás összeforrasztás után.
- A hidegforrasztás kerülése, kötések ellenőrzése mikroszkóppal (nem megfelelő kötés elengedhet rázóteszt során), akár röntgennel. A 3.51. ábrán látszik, hogy a kötések fémesen fénylenek, tehát hidegforrasztás nincs a panelen.

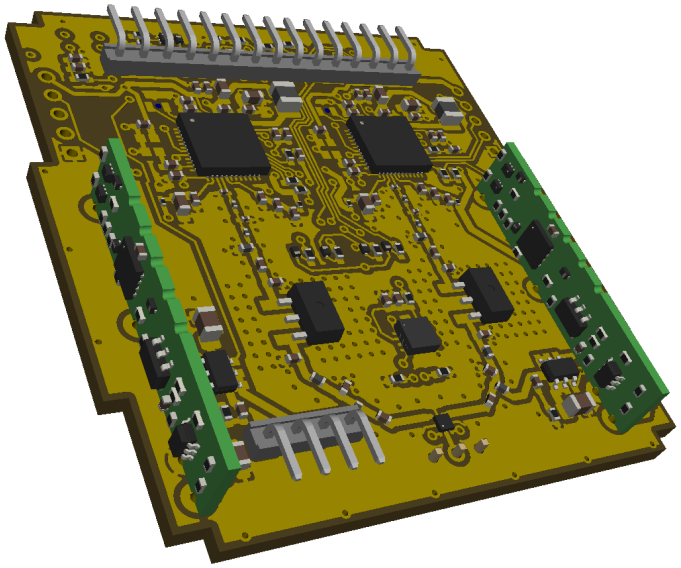
Az utolsó két pont hibátlan betartásához megfelelő forrasztási képességekkel rendelkező egyénre van szükség. Mivel ezt a feladatot én szerettem volna elvégezni, az egyetemen töltött éveim során figyeltem rá, hogy ezeket a képességeket megszerezzem. Így is történt, és a panel átment a rázó és vákuumkamrás teszteken.

Egy PocketQube osztályú műholdban fontos tervezési korlát a méret. Az alkotórészek szorosan illeszkednek egymáshoz, kis toleranciákkal. Emiatt az S-sávú adópanelen is figyelni kellett különböző korlátozásokra.

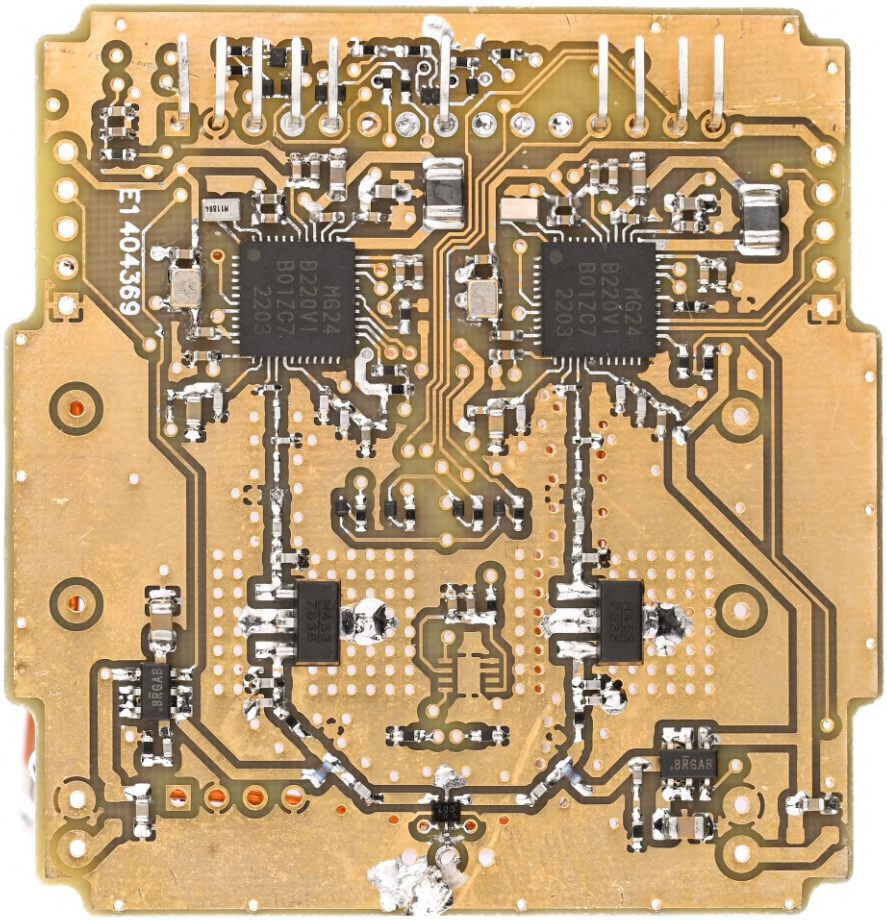
- A panel négy oldalán lévő illeszkedő fülekre nem szabad alkatrészt tenni, se huzalozást a felső rétegére, mivel a forrasztásgátló maszk hiánya miatt ezek rövidre záródhatnak.
- Az áramkör elhelyezkedése miatt a műholdban nem nyúlhat ki egy alkatrész sem a panel síkjából több mint 1,5 milliméterre (kivéve áramlimiter kapcsolók és csatlakozók).
- Minden alkatrészhez kell egy pontos 3D CAD fájl, például step formátumban (3.50. ábra), hogy a végleges modellben legyártás és összeszerelés előtt látható legyen, hogy minden megfelelően illeszkedik.
- Az áramkör körülbelül nettó 4,5 cm Y tengely irányban, ez nagyon szoros korlát, mivel három illesztőhálózat, egy erősítő és egy RFMCU is ezen a hossztengelyen helyezkedik el. Tehát nagyon kompakt tervezés szükséges (a 0201 alkatrészek használatához ez is hozzájárult, nem csak a parazita impedanciák) (3.46. ábra).

3.6.3. Az áramkör élesztése

Az áramkörök összerakása részletről részletre haladt. Mindig csak egy funkcionális részt forrasztottam be és teszteltem, csak azután következhetett a másik.



3.50. ábra. Az S-sávú adó áramkör komplett 3D modellje



3.51. ábra. Az S-sávú adó áramkör kvalifikációs példánya

1. Az RFMCU beültetése és a hozzátartozó tápszűrés, programozó interfész tesztelése: felismeri-e az IC verziót, sorozatszám kiolvasható-e?
2. Feszültségszintek megfelelőek-e? Táplábak feszültsége: 3,3 V, DVDD és VREGSW láb feszültsége: 1,8 V
3. Rádióhoz szükséges alkatrészek beültetése: TCXO és illesztés.
4. Rádió tesztelése vezetett méréssel, modulálatlan vivőjellel (3.45. ábra).
5. Soros leválasztó beforrasztása és tesztelése.
6. Erősítő és a hozzátartozó illesztőhálózatok beültetése és tesztelése.
7. RF kapcsoló beforrasztása és ellenőrzése.
8. Antenna felragasztása és bekötése.

A panel működését teljes mértékben vizsgáló tesztekről a 3.8 szekcióban lesz szó.

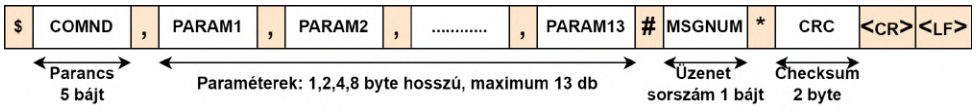
3.7. A rádión futó szoftver

Az eszközön futó szoftver a Simplicity Studio nevű fejlesztőkörnyezetben készült C kód, ARM Cortex-M33 processzorra fordítva. Működését a 3.54 folyamatábra ábrázolja.

3.7.1. Kommunikációs protokoll

Az On-Board Computer-rel (OBC) az eszköz soros porton, a műholdra fejlesztett, egyedi protokollon keresztül kommunikál.

A fejlesztés egy korai fázisában minden modulfejlesztő megkapta a protokoll definíciót, ezzel együtt a függvények pszeudokódját is, és a megvalósítandó parancsokat, amelyeket az OBC-től kaphat. Az összes kommunikációs modul full-duplex UART perifériát kapott, tehát az STX is. Ezen érkeznek az üzenetek, melyeknek formátuma az 3.52. ábrán látható. A parancs és a paraméterek ASCII karakterekből állnak, ivéve némelyik speciális esetet, amelyről később lesz szó.



3.52. ábra. MRC100 soros busz üzenet formátum

Három parancs lett definiálva az STX számára:

- `RQCST`: Ez egy státusz lekérő parancs, melyre a válasz a következőképpen lett definiálva:
  `TECST,S,TEMP,DATCOUNT`.
  - `S`: státusz flag, `R`: READY készen áll a következő parancsra ,`B`: BUSY elfoglalt, `E`:ERROR hiba történt az előző parancs feldolgozása közben

  - `TEMP`: Hőmérséklet paraméter, amely végül nem lett felhasználva, mivel az OBC más modultól kapta az STX hőmérsékletét.
  - `DATCOUNT`: az STX bufferében lévő bájtok száma

- `SBUFF,{MSLIP␣Bináris}`: buffer feltöltő parancs, tartalma a kódolt elküldendő bináris, erre a válasz nyugta üzenet
  - `MSLIP␣Bináris`: MSLIP (Modified SLIP) kódolással átküldött bináris adat. Az MSLIP lényege, hogy a protokollban használt kulcskaraktereket ($,*,# stb.) helyettesítse egy előre definiált, másik karaktersorozattal. Ennek a funkciója, hogy ne zavarják meg ezek a karakterek a kommunikációt. Mivel csak dúsítással lehetséges ez a helyettesítés, több lesz az átküldött adat, mint a bináris (ha minden karaktert helyettesíteni kell, akkor a kétszerese). Ez természetesen a fogadó oldalon dekódolódik, és az eredeti binárist az STX beletölti a bufferébe.

- `SENDB,P,MODU`: adás indító parancs, a bufferben lévő bájtokat kiküldi, erre a válasz nyugta üzenet (nem azt jelenti, hogy el is lett küldve az összes bájt)
  - `P`: rádió teljesítménye, 0-9-ig vehet fel értéket. Értékei: 3.1.táblázat
  - `MODU`: rádió sávszélessége, 0-9-ig vehet fel értéket. Értékei: 3.2.táblázat

3.1. táblázat. P paraméter értékei

P	EIRP
0	21 dBm
1	22 dBm
2	23 dBm
3	24 dBm
4	25 dBm
5	26 dBm
6	27 dBm
7	28 dBm
8	29 dBm
9	30 dBm

3.2. táblázat. MODU paraméter értékei

MODU	Sávszélesség
MOD0	2000 kbps
MOD1	1800 kbps
MOD2	1600 kbps
MOD3	1400 kbps
MOD4	1200 kbps
MOD5	1000 kbps
MOD6	800 kbps
MOD7	400 kbps
MOD8	200 kbps
MOD9	100 kbps

  - `MODU`: A moduláció sávszélességét meghatározó kódszó. Értékei: 3.2.táblázat

A nyugta üzenet formátuma pedig érvényes minden modulra:

- `ACKMS`: az üzenet sorszáma helyén az érkezett parancsüzenet sorszáma található

A kommunikációs függvények az 5.6 fájlban vannak kódolva és a 5.7 fájlban pedig definiálva .

3.7.2. Inicializálás

A program első állapota az inicializálás, itt tölti be a mikrovezérlő a különböző beállításait.

- UART Periféria: beállítja az RX/TX lábak helyét, baudrate-et 500kbps-re konfigurálja, és bekapcsolja a vételi karakterre megszakítást.

- Rádió: a szoftver Silicon Labs által fejlesztett RAIL (Radio Abstraction Layer) API-t (Application Programming Interface) használja. Ez egy absztrakciós réteg, tehát a felhasználónak nem regiszter állításokon keresztül kell elérni a rádiót, hanem különböző magasabb szintű függvényekkel pl. `RAIL_StartTx()`, `RAIL_SetTxFifo()`.

  A modem beállításokat, mint például működési frekvenciák, teljesítmény határértékek, moduláció típus stb. fordítás előtt egy úgynevezett Radio Configurator grafikus interfészen keresztül lehet megválasztani. Ez legenerálja a szükséges rádió regiszter értékeket C kódként, amely belefordul a binárisba, amelyet a mikrokontroller futtat. Inicializáláskor az itt beállított értékeket a RAIL betölti a rádióba.

  A különböző sávszélesség beállításokat külön csatornaként hoztam létre (2000kbps-100kbps, 10 csatorna), az ezeken való váltással lehet választani az sebességek közül. A teljesítményt pedig minden csomag adása előtt lehet állítani.

- GPIO periféria: erősítő vezérlő lábak konfigurálása, ezen kívül mindkét mikrovezérlő (primer és redundáns) ilyenkor a saját vezető ágára állítja az RF kapcsolót.

A inicializáló függvényeket az 5.3 fájlban kódoltam, és az 5.4 fájlban definiáltam. A különböző konfigurációs konstansokat pedig az 5.5 fájlban hoztam létre.

3.7.3. READY állapot

Inicializálás után az eszköz felveszi a READY állapotot, és az UART periféria elkezdi venni az bejövő karaktereket. Ha az előbbiekben megfogalmazott protokoll definíciónak megfelelő üzenet érkezik, akkor az feldolgozásra kerül. Ehhez meg kell felelnie a különböző elválasztó karaktereknek és a CRC értéknek, ha bármi hibás, akkor az STX eldobja az üzenetet. Ekkor nem érkezik válasz az OBC-nek, tehát az tudja, hogy hiba történt a parancs közvetítése során, ezért újra küldi az üzenetet. Ez megtörténik egy meghatározott mennyiségben, és ha továbbra sem válaszol, akkor az OBC tovább lép, ilyen esetben át lehet kapcsolni a redundáns adóra, földi állomásról küldött paranccsal.

A fentebb leírtak szerint három parancsot kaphat ilyenkor az eszköz:

- `RQCST`: A státuszparancsra `TECST` paranccsal válaszol, a státusz flag értéke `R`, és ha már érkezett `SBUFF` parancs akkor a kiküldendő bájtok számát fogja tartalmazni a `DATCOUNT`, hogyha nem, akkor nulla lesz az értéke.

- `SBUFF`: Nyugta válasz után az OBC egy `RQCST` paranccsal lekérdezi a bufferbe érkezett bájtok mennyiségét, és hogyha egyenlő azzal amit küldött, akkor tovább léphet a küldésre.

- `SENDB`: Ha az előbbi pontban leírtak szerint egyezik az adatszám akkor az OBC ezt a parancsot küldi. Ha az STX megkapta, akkor elkezdi az adatfeldolgozást és küldést, tehát átlép BUSY állapotba, közben válaszol az OBC-nek egy nyugtával. Hogyha a buffer üres, akkor ugyanúgy lefut a kiküldést intéző algoritmus, csak amikor érzékeli, hogy nincs mit kiolvasni küldésre, akkor rögtön az adás befejezéséhez ugrik.

Ha minden rendben ment, tehát fel van töltve a buffer (mérete 198000 bájt), és megérkezett egy érvényes `SENDB` utasítás, akkor átlép a program BUSY állapotba, és megkezdődik az adás.

3.7.4. BUSY állapot

A BUSY állapotban az eszköz összerakja a kapott bináris adatokból a csomagokat, és kisugározza őket.

Az OBC egyben küld ki max 198000 bájt adatot, amelyet az STX a bufferébe tölt. Ezeket fel kell bontani 122 bájtos csomagokra, számozni kell őket, és CRC bájtokat, azaz Cyclic Redundancy Check algoritmussal generált ellenőrző adatot kell hozzájuk fűzni.

A CRC bájtok célja, hogy a vett adatokban hibát tudjunk detektálni. Ez úgy lehetséges, hogy az adó számol az elküldendő adat alapján egy értéket, úgynevezett ciklikus kóddal. Ez az érték bele kerül a csomagba, és a vevő oldalon ugyanezzel a ciklikus kóddal számolunk egy értéket, és összehasonlítjuk a kapott és a számolt adatot. Ha a kettő egyezik, akkor valószínűleg hibátlan az adat.

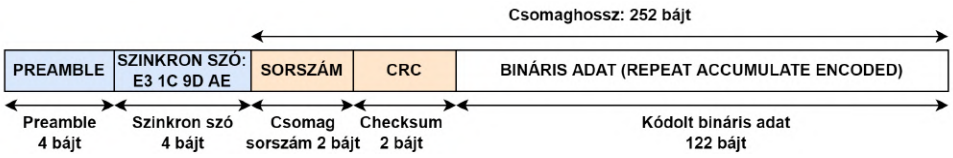
A sorszám célja, hogy ha hiba keletkezik vétel közben egy csomagban, akkor be lehessen azonosítani, hogy pontosan melyikben. Ha sikerül beazonosítani, akkor a Földről utasítani lehet az OBC-t, hogy csak azt a csomagot küldje újra. A sorszám 2 bájt, mivel a bufferben tárolt adat hossza maximum 198000 bájt (1623 csomag), és ez nem ábrázolható 1 bájton, minimum 2 kell hozzá.

Ha a CRC és a sorszám belekerült a csomagba, akkor egy úgynevezett Repeat Accumulate kódolás lesz rátéve, melynek kódaránya 1/2. Tehát a végleges csomaghossz 252 bájt lesz (3.53). Ez egy hibavédő kódolás, melynek célja, hogy rossz jel-zaj viszony mellett is lehessen hibátlan csomagokat venni.

A csomagok struktúrája a 3.53. ábrán látható. Nagyon hasonló szerkezetű csomagokat használt a SMOG-1 és SMOG-P műhold is (kisebb sorszám méret).

A kész csomag betöltődik a RAIL által definiált `TxFifo`-ba, és elindul az adás. Adás közben az STX figyel az OBC-től jövő `RQCST` parancsokra, és mivel BUSY állapotban van, ezért `B` státuszú `TECST` üzenetekkel válaszol, melyekben mindig az aktuális kiküldetlen bájtszám is jelen van. Erre azért van szükség, hogy az OBC tisztában legyen azzal, mikor van vége az adásnak, és amikor ez megtörténik, akkor le tudja kapcsolni az STX áramát. Mivel a szoftver rögtön lekapcsolja az erősítőt, ha kiment az utolsó csomag, ezért ha nem kapcsol ki rögtön az STX, az 8 mA extra fogyasztást jelent (a READY állapot fogyasztása). Viszont egy ekkora méretű műholdnál nagyon fontos az energiagazdálkodás, mivel a készletek végesek. Tehát mindent meg kell tenni a fogyasztás optimalizálása érdekében.

Hogyha bármi hiba történik a BUSY állapotban az adás bármelyik lépésében, akkor az eszköz rögtön ERROR állapotba kerül.



3.53. ábra. STX csomag formátum

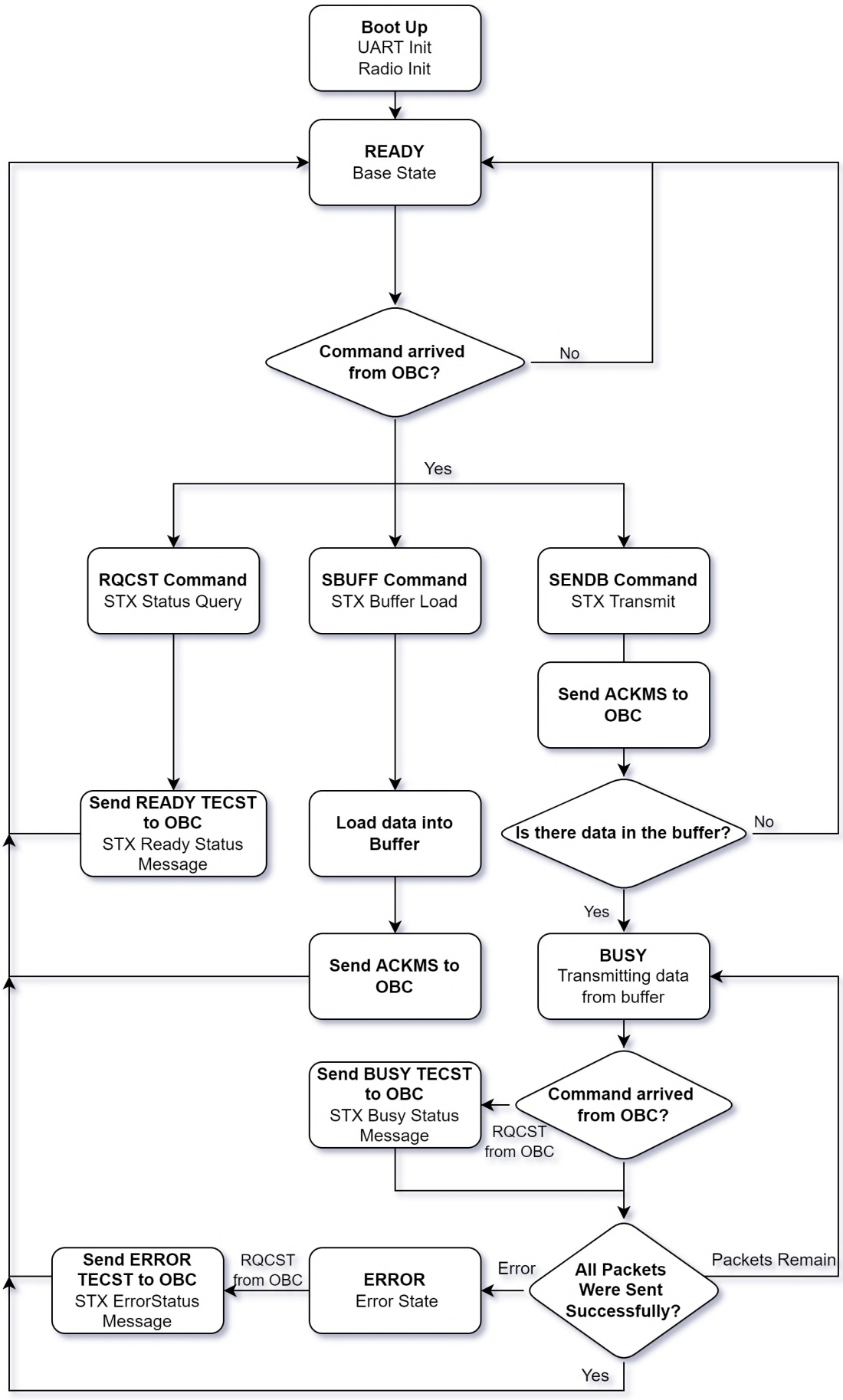
3.7.5. ERROR állapot

Ez az állapot felel a hibák jelzéséért. Ezeket a hibákat a RAIL API generálja bármilyen rádiós probléma fellépésekor. Normális működés esetén soha nem kerülhet az adó ebbe az állapotba. Viszont az űr egy veszélyes környezet az elektronikák számára, és ha például a kozmikus sugárzás az adó bármilyen belső változóját átbillenti, akkor ez okozhat ilyen hibát. Erről pedig értesíteni kell az OBC-t.

Mivel az STX gyakorlatilag egy "slave" periféria, nem kezdeményezhet kommunikációt. Így csak a státusz flag-en keresztül tudja a hibát közvetíteni, de ez nem probléma, mivel a fedélzeti komputer többször is lekérdezi a státuszt adás közben és utána is. Ha hibát kap, akkor újraindítja az adást, és ha ez sokszor előfordul, akkor magát az STX-et is.

Ha ERROR állapotban érkezik egy RQCST parancs, akkor az eszköz visszatér READY állapotba.

A fő állapotgép függvényei az 5.1 fájlban vannak kódolva, és az 5.2 fájlban definiálva.



3.54. ábra. Az adó állapotábrája

3.8. A befejezett panel tesztelése

3.8.1. A teszt szoftver

A kész eszköz teszteléséhez szükség van egy szoftverre, amely soros porton keresztül képes az előző részben leírt parancsok kiküldésére. Tehát gyakorlatilag az OBC kommunikációját emulálja.

Mikrokontrolleren már meg lett valósítva ezeket a parancsokat generáló és feldolgozó kód C nyelven, úgyhogy az egyetlen része, amely nem platformfüggetlen, az a soros porton végzett karakterküldés és -fogadás.

Az OBC emulátor szoftver tehát nagyon hasonló, mint a mikrokontrolleren futó társa, csak más parancsokat generál, és PC platformon van implementálva a soros port kezelés. Egy terminál interfészen keresztül irányítva képes `RQCST,SBUFF,SENDB` utasításokat küldeni, és a válaszok érvényességét hitelesíteni (az OBC emulátor függvényei az 5.8 fáljban vannak megírva, és az 5.8 fáljban definiálva).

A később leírt hardverrádiós és szoftverrádiós tesztekhez ezzel a szoftverrel ellenőriztem az STX működését, amíg az OBC nem készült el teljesen.

A következő részben leírtakhoz egy egyszerűsített szoftvert írtam, amely a tápfeszültség megjelenése után 10 másodperccel elkezd sugározni egy modulálatlan vivőjelet az adott frekvencián. Jelen esetben ez 2267,5 MHz volt.

3.8.2. Sugárzott teljesítménymérés

Ahogy a 3.6.1 alszekcióban is említettem, az utolsó módosításokat az illesztésen sugárzott teljesítményre optimalizáltam.

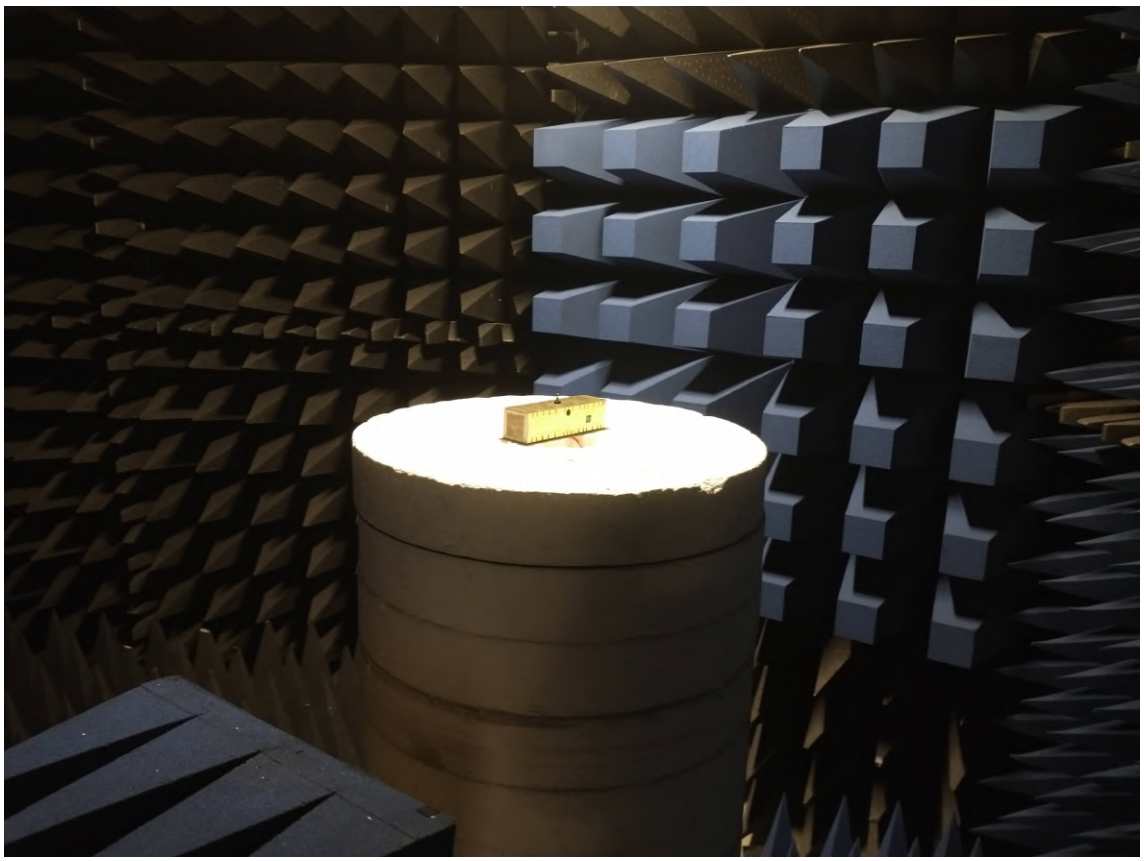
Erre azért volt szükség, mert a napszenzornak kivágott darab a dielektrikumból, maga az STX panel és a műhold váza is mind változtattak az antenna impedanciáján. Ezért az illesztést hozzá kellett méretezni ezekhez az impedanciaváltozásokhoz. Mivel esténként rendelkezésre állt egy reflexiómentesített mérőszoba (részletesebb leírás a 3.4 szekcióban), és a sugárzott teljesítmény a legfontosabb mérendő paraméter egy ilyen rendszernél, nagy mennyiségű mérést végeztem a minden módosítást tartalmazó végleges áramkörön. Ez addig tartott, amíg a specifikációban definiált értékeket el nem értem.

A végső mérés tartalmazta (3.55. ábra):

- A teljes mértékben összeforrasztott panelt
- A műhold külső vázát, belső panelek nélkül
- A panelre ragasztott antennát a napszenzorral
- Egy 3,7 V nominális feszültségen üzemelő lítiumcellát, amelyre egy kapcsoló üzemű tápáramkör volt kapcsolva, így emulálva a műhold szabályzott energiabuszát

A fenti összeállítás célja, hogy az antenna közelterében lévő alkatrészek a legjobban utánozzák a végleges műholdszerkezetet. Például ezért táplálja akkumulátor, nem pedig egy külső labortáp, mivel így a szerkezetből kilógó kábelek nem hoznak váratlan sugárzókat és parazita impedanciákat a mérésbe.

A mérések 30 dBm EIRP értéket hoztak a végső illesztőhálózattal, amely megfelel a specifikációnak. Ez azért alacsonyabb (körülbelül -1 dB), mint az 3.4.3 alszekcióban, mert az RF kapcsoló beiktatási csillapítása körülbelül 0.4 dB, és a napszenzor ront egy kicsit az antenna sugárzási karakterisztikáján.



3.55. ábra. A műhold vázával együtt illesztett adó sugárzott mérése

3.8.3. Szoftverrádiós vételi tesztek

Az adót kétféleképpen lett tesztelve szoftverrádiós vétellel, melynek módjáról a 4.3.2 alszekcióban található részletesebb leírás. A fejlesztés korábbi fázisában, az első funkcionális teszt során csak az OBC-vel lett összecsatlakoztatva, ahol a kommunikációs protokoll implementációja lett ellenőrizve. Miután ez megtörtént, a kimenő csomagok tartalma szoftverrádióval is lett ellenőrizve. A jelforma idő- és frekvenciatartományban is meg lett vizsgálva. Sugárzott teljesítménye validálva lett egy dipól antennával és egy fix távolságon. Az adó áramának értéke az áramlimiter kapcsoló szerint 520 mA, tehát összteljesítménye 1716 mW. A fejlesztés utolsó fázisában, mikor már kész volt az egész műhold, ugyanezen a teszten kellett átmennie az adó repülő példányának (a műholdba beépítve). Az adó megfelelt ezeken a teszteken.

4. fejezet

A rádióvevő

A jel vétele az BME E épület tetején lévő Etető vevőállomás forgásparaboloid antennájával lesz kivitelezve, melyre egy szoftverrádió lesz kapcsolva. Ezen fog történni a demoduláció és a további jelfeldolgozás.

4.1. Link számítás

Egy rádiós kapcsolat tervezésénél fontos a különböző teljesítményviszonyok kiszámolása, hogy a mérnök lássa egyáltalán lehetséges-e az adott paraméterekkel egy működő linket létrehozni. Ez található a következő szekcióban.

$$\lambda_{Sband} = \frac{c}{f_S} = 13,22\,cm \tag{4.1}$$

c : Fénysebesség: 299792458 m/s
f_S : STX frekvenciája: 2267,5 MHz

$$a_{LEO} = 20\log_{10}\frac{4\pi R_{LEO}}{\lambda_S} = 169\,dB \tag{4.2}$$

a_{LEO} : Low Earth Orbit szabadtéri csillapítása [dB]
R_{LEO} : Műholdpálya távolsága: 3000 km
λ_S : Hullámhossz: 13,22 cm

$$G_{RX} = 20\log_{10} k\frac{\pi D}{\lambda_S} = 36\,dBi \tag{4.3}$$

G_{RX} : E épület tetején lévő antenna nyeresége [dBi]
k : Antenna hatásfoka: 70%
D : Antenna átmérője: 3 m

$$P_{RX} = P_{TX} - a_{LEO} + G_{TX} + G_{RX} = -104\,dBm \tag{4.4}$$

P_{RX} : Vett teljesítmény [dBm]
P_{TX} : Adó teljesítmény (különböző csillapításokat beleszámolva): 26 dBm
G_{TX} : Adó antenna nyereség: 4 dBi

$$P_N = 10 \log_{10} \frac{kBT}{1mW} = -111 dBm \tag{4.5}$$

$$SNR = P_{RX} - P_N = 7\, dB \tag{4.6}$$

P_N : Zajteljesítmény [dBm]
k : Boltzman állandó: $1,38e{-}23\frac{J}{K}$
B : Sávszélesség: 2 MHz
T : Antenna zajhőmérséklete: 300 K
SNR : Jel-zaj viszony [dB]

Az 1/2 kódarányú Repeat Accumulate kódolóval védett adatot GMSK moduláció esetén 2 dB SNR jel-zaj viszony mellett kevesebb, mint 0.1% bithiba aránnyal lehet dekódolni. Tehát a csatornán maximális 2 Mbps sávszélességnél is 5 dB tartalék van a rendszerben. Ez 1 Mbps sávszélességnél már 8 dB tartalékot jelent.

4.2. GMSK moduláció

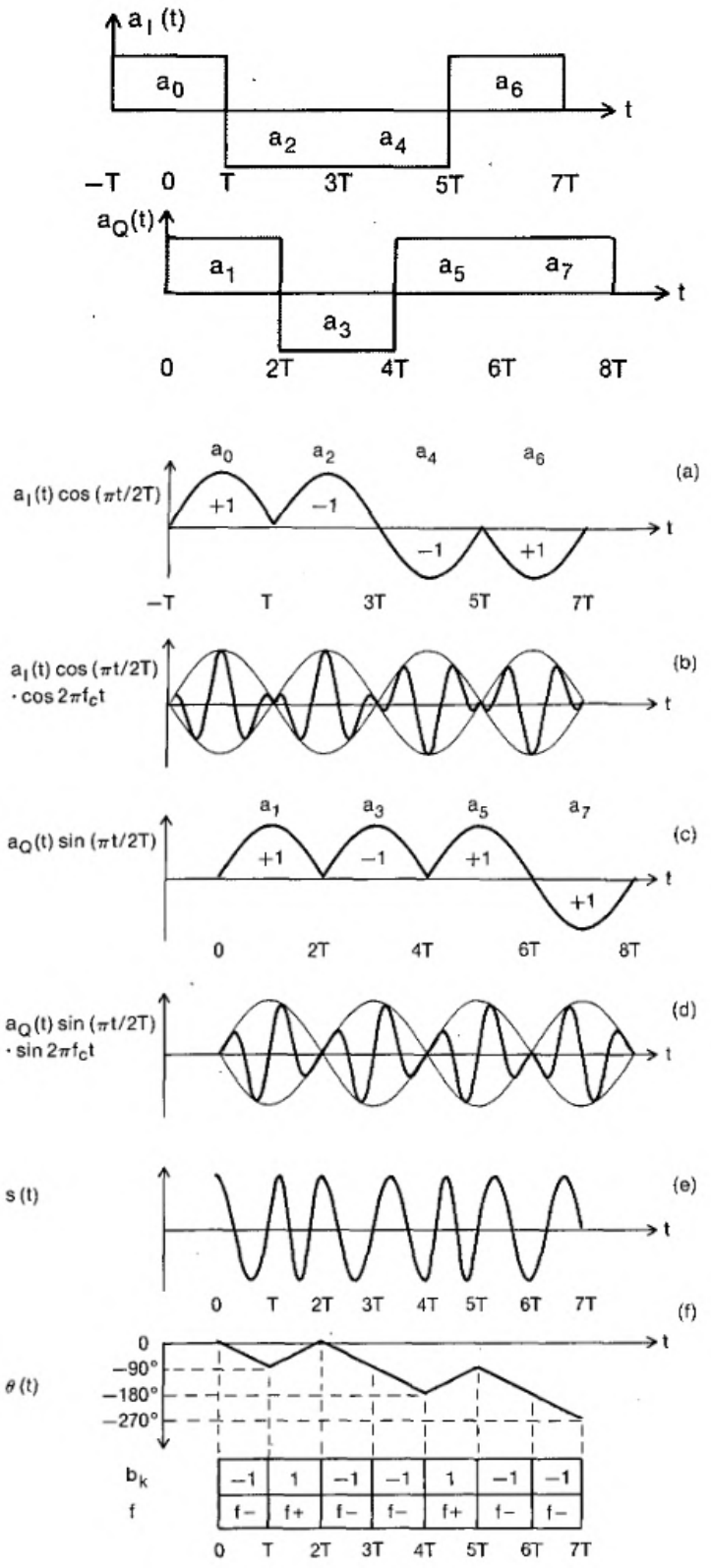
Az MSK moduláció, azaz Minimum Shift Keying, a frekvenciamoduláció egy szélső esete, ahol a modulációs index pont akkora, hogy a szimbólumokat reprezentáló frekvenciák még ortogonálisak maradjanak egymásra.

Más irányból nézve pedig a fázis moduláció egy olyan fázis-folytonos fajtája, ahol az alapsávi impulzusokat szinuszos súlyozással vesszük figyelembe, és a szimbólumok között 90 fokos fáziskülönbség van (4.1. ábra).

Nagyon fontos tulajdonsága, hogy mint minden frekvenciamodulált jel, konstans burkolóval rendelkezik. Ez azt jelenti, hogy a modulált jel információtartalma amplitúdófüggetlen, tehát jól ellenáll a műholdas kommunikációnál fennálló fading jelenségeknek. Ezenkívül még a különböző nem-lineáris torzításokra is kifejezetten érzéketlen, tehát nem probléma, hogyha az erősítést végző tranzisztor olyan fokozatban működik, ahol telítésbe megy.

Az MSK moduláció a konstans burkoló és a fázisfolytonosság miatt jó sávkihasználással rendelkezik. Ha a PSD (Power Spectral Density) értékeiket összehasonlítjuk, akkor jobb, mint a QPSK és az Offset QPSK [5]. A novemberben kiderült specifikációk alapján 2 MHz sávszélesség korlát lesz az S-sávú downlink kommunikációra, ami kimondottan előnyös a küldetés számára.

A GMSK moduláció, azaz Gaussian Minimum Shift Keying, az MSK moduláció egy fajtája, melynek alapsávi jele Gaussi szűrővel lett formálva. Ez csökkenti a sávszélességét és a szimbólumok közti áthallást (ami műholdas csatornán nem olyan lényeges, mivel főleg kétutas terjedés miatt szokott előfordulni).



4.1. ábra. Az MSK jelformája, mint IQ fázis moduláció [5]

4.3. Vevőeszközök

Egy adó fejlesztésének a végső fázisa, hogy az általa sugárzott adatot megpróbáljuk venni.

4.3.1. Hardverrádió

Mivel a szoftver fejlesztését először egy BRD4187C fejlesztő panelen végeztem, célszerű volt a vételt is egy ilyen eszközön implementálni.

A Silicon Labs EFR32 IC családjához jár egy terminál interfésszel rendelkező tesztelő program, a RAILTest [8]. Ez ugyanarra a RAIL API-ra épül, amelyet az STX szoftver használ. Ezzel gyakorlatilag egy rádió összes funkcióját ki lehet próbálni.

Tehát a csomagok vételéhez nem kellett más, mint konfigurálni az előbb említett program rádióját, pont ugyanúgy, ahogy az STX oldalán. A vevő így megszámolja a vett csomagokat, és ez alapján lehet ellenőrizni, hogy az adó megfelelően működik. A bájtok helyességéről a RAIL által biztosított beépített CRC gondoskodott.

4.3.2. Szoftverrádió

A fentebb leírt módszer hátránya, hogy csak azt ellenőrzi, hogy a moduláció és csomaghossz megfelelnek-e. Viszont a bájtok tartalmáról, tehát a csomag belső struktúrájáról nem mond semmit. Csak az derül ki, hogy ugyanazok a bájtok érkeztek meg a vevőbe, amelyek elhagyták az adót. Az olyan kérdésekre, hogy a kódoló implementációja működik e, vagy a sorszám megfelel e a csomagspecifikációnak, nem ad választ. Másik hátránya, hogy a vett adatok soros porton keresztül jönnek a terminál interfészen keresztül, sok redundáns információt tartalmazva. A soros porti terminál interfésze pedig a fejlesztő panelnek nem elég gyors ahhoz, hogy a maximális 2 Mbps sebességű bitfolyamot továbbítsa.

A nagy mértékű flexibilitás mellett a szoftverrádiós vétel az előbb leírt problémákat is megoldja. Mivel a SMOG-1 is GMSK modulációt használt és ugyanezt a kódolást, ezért az ahhoz írt szoftver lett használva a teszteléshez [1]. A vevőeszköz egy Ettus Research B200mini szoftverrádió volt (4.2), az egyik STX-hez gyártott, behangolt prototípus antennával.

4.2. ábra. A B200Mini szoftverrádió

A rádiós kapcsolat le lett tesztelve ezzel az összeállítással, maximum teljesítményen, egy meghatározott bitsorozattal. A kódoló jól működött, a CRC és sorszám bájtok megfelelő helyen lettek elhelyezve a csomagban, és az értékük is specifikáció szerint teljesült.

4.3.3. Hullámpolarizációs problémák

A 3.4 szekcióban leírt antenna polarizációja lineáris, ezért ha a műhold a Z tengelye körül forog, akkor a polarizációs különbség által keletkezett csillapítás több, mint 10 dB is lehet. Természetesen ez akkor fordulhat elő, hogyha a vevőantenna polarizációja is lineáris.

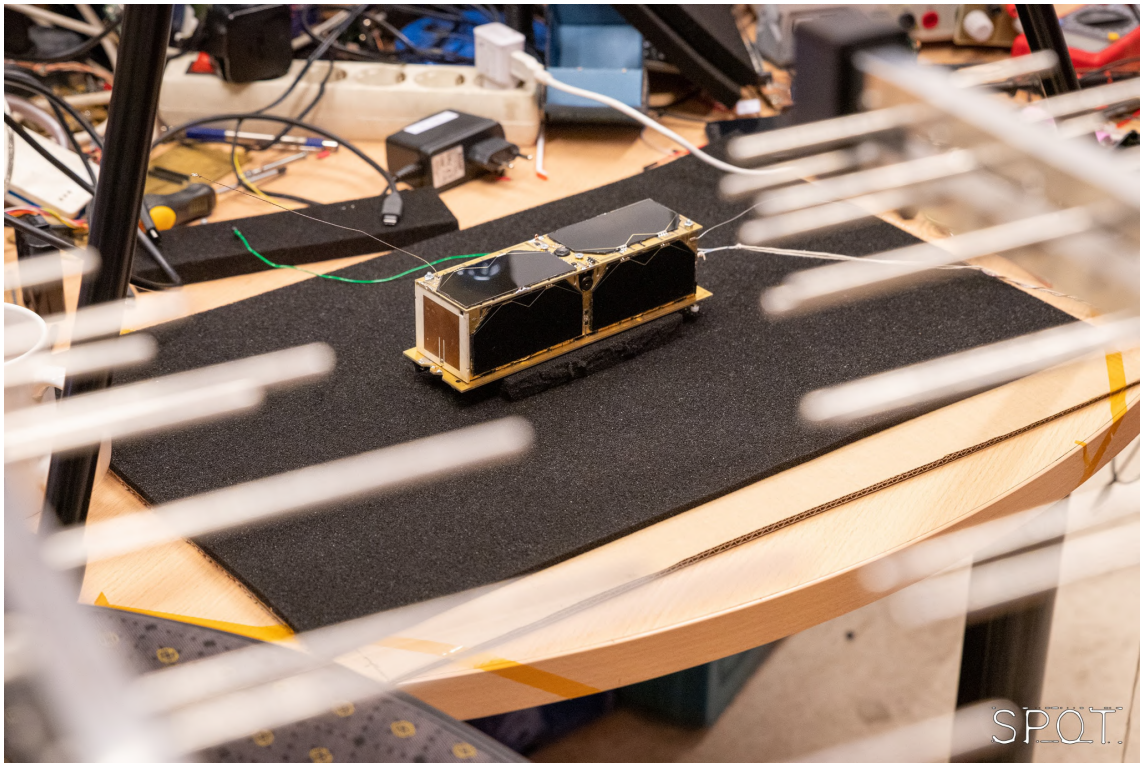
Erre a problémára megoldás lehet egy cirkulárisan polarizált antenna, viszont az E tető vevőállomáson lévő forgásparaboloid antennánál van egy ennél jobb megoldás is: az úgynevezett polarizáció "diversity". Ez ennél az alkalmazásnál azt jelenti, hogy a forgásparaboloid fókuszpontjában el lesz helyezve egy vertikális és egy horizontális polarizációval rendelkező antenna, és ezeknek a komplex mintáiból lesz szoftveresen szimultán kikeverve a vett jel. Így a polarizációkülönbség által behozott veszteség minimalizálható. Ez a funkció még nem lett telepítve a vevőállomáson, és e dokumentum írása közben még fejlesztés alatt áll.

4.3. ábra. A BME Etető vevőállomás

5. fejezet

Összegzés

Az elmúlt félévek során az MT Laborban (és máshol) dolgozó mérnökök és mérnök hallgatók kitartó munkájának eredményeképpen elkészült az MRC100 műhold és 2022 december 2.-án hivatalosan át is adták. December 3-án elindult Skóciába, ahol az Alba Orbital, PocketQube szatellitekre szakosodott cég pályára állító platformjába integrálva megindul az Amerikai Egyesült Államok felé. Itt előreláthatólag 2023 májusában, a SpaceX Falcon-9 rakétájában bocsátják majd fel.



5.1. ábra. A felbocsátásra kész műhold (kredit: Püspök Péter, SPOT)

A műholdon rengeteg új kísérlet és fejlesztés kap helyet (5.2. ábra). Ezek közül kiemelném a rekordméretű aktív mágneses helyzetstabilizáló rendszert, mely jelentős kísérleti fejlesztés a miniatűr műholdak világában.

A helyzetstabilizáló rendszer a Föld mágneses terét használva tekercsekkel tud erőt kifejteni a műholdra. Ezt kihasználva, a sokféle helyzetadatból, amelyet a fedélzeten lévő különböző környezeti szenzorok (gyorsulásmérő, napszenzor, Hall-effekt szenzor stb.)

5.2. ábra. A műholdban helyet kapott modulok

szolgáltatnak, a rendszer stabilizálni tudja a műholdat.

Ez a stabilizálás szükséges a dolgozat által tárgyalt S-sávú adónak is, mivel az antennája irányított. Tehát szükség van arra, hogy a Föld felé nézzen adáskor.

Az S-sávú adó jelenléte az űreszközön előrelépés az eddigi műholdak rádiós kommunikációjához képest. Mivel itt már nincs az UHF sávban jelenlévő sávszélességkorlátozás, a 12,5 kbps helyett 2 Mbps sebességgel tud a Földdel kommunikálni, és a nagyjából 10 perces áthaladás alatt képes lesz leküldeni a 3 GHz-ig üzemelő spektrumanalizátorok mérési adatait, a kamera képeit és a többi mérés által jegyzett adatot.

A rádiós teljesítményre vonatkoztatott hatásfoka kitűnő, és szinte egyedülálló.

$$\eta = \frac{P_{TX}}{P_{DC}} = 29\% \tag{5.1}$$

η : Rádiós teljesítményre vonatkoztatott hatásfok [%]
P_{TX} : Adóteljesítmény (vezetett mérés kimeneti illesztésen): 500 mW
P_{DC} : STX DC teljesítménye (RFMCU és erősítő együtt): 1716 mW

Az előbbi állítást igazolni kívánva, a teljesség igénye nélkül kigyűjtöttem a vele összehasonlítható S-sávú adók közül hármat, és listáztam a rádiós teljesítményre vonatkoztatott hatásfokukat:

- HiSPiCO S-Band Transmitter:
  - Tápfeszültség: 3,3-5 V
  - Maximum teljesítmény : 27 dBm
  - Fogyasztás: 5 Watt
  - Hatásfok: 10%
- Nanoavionics SatLab S-Band Transmitter:
  - Tápfeszültség: 5-40 V
  - Maximum teljesítmény : 30 dBm
  - Fogyasztás: 5,9 Watt
  - Hatásfok: 17%
- DP-CRF-5615 S-Band Transmitter:
  - Tápfeszültség: 3,3-5 V
  - Maximum teljesítmény : 30 dBm
  - Fogyasztás: 6,5 Watt
  - Hatásfok: 15%

5.1. Szerzett tapasztalatok

Az itt dokumentált munkám során rengeteg tapasztalatot szereztem a nagyfrekvenciás rendszerek tervezésében és megvalósításában.

Illesztéseket és antennát terveztem, szimuláltam, építettem és mértem le. Az ezekhez használatos műszerekkel is szereztem mélyreható tapasztalatot, ilyen a hálózatanalizátor, spektrumanalizátor, közeltérmérő antennamátrix és a reflexiómentesített mérőszoba. Megtanultam az ipari standard szimulátorok használatát, mint az AVR Microwave Studio vagy a CST Studio Suite.

A forrasztási képességeimet fejlesztettem miniatűr felületszerelt alkatrészeken és olyan paneleken, amelyeket én terveztem KiCAD szoftverrel. Ezeket a rádiófrekvenciás paneleket teszteltem és hibákat kerestem rajtuk, eközben az ehhez használt műszereken szereztem gyakorlatot.

A megvalósított áramkörökhöz szoftvert fejlesztettem, eközben jelentős tapasztalatokat szereztem a beágyazott C programozás terén.

A fejlesztést sikerült időben befejezni, és diplomamunkám elkészültével jó szívvel írhatom, hogy egy jó paraméterekkel és specifikáció szerint működő eszközt adtam ki a kezem közül.

Köszönetnyilvánítás

Köszönet Dr. Dudás Leventének a rengeteg segítségért, a jó ötletekért, a rászánt időért, tudásért és azért a lehetőségért, hogy amit én építettem, kint járhat a világűrben. Köszönet azért, hogy úgy adott nekem egy fontos témát, hogy nem tudta, képes leszeke megcsinálni, de végig kitartott mellettem. Ez a téma motivált végig a mesterszakos tanulmányaim alatt.

Köszönöm a Mikrohullámú Távérzékelés Laboratóriumban dolgozó kollégáimnak a rengeteg segítséget. Herman Tibornak, Hödl Emilnek és Szüllő Ádámnak azt, hogy minden kérdésemre türelmesen válaszoltak, és feltétel nélkül segítettek. Nagy Dominiknak, hogy mindig ott volt, ha valamilyen problémát meg kellett oldani.

Köszönöm a Silicon Labs Hungary Kft.-nál dolgozó kollégáimnak: Cseh Ádámnak, Vida Zoltánnak, Komancsik Mártonnak, Simon Dánielnek, Szabó Dánielnek, Dr. Zólomy Attilának, Bódi Tamásnak, Varga Péternek és Böröndy Áronnak a sok jó ötletet, türelmes segítséget, és hogy mindig figyeltek a kérdéseimre és kéréseimre.

Irodalomjegyzék

[1] Dr. Dudás Levente: Rádió- és antennarendszerek radaros és műholdas alkalmazásai. Doktori értekezés. Budapesti Műszaki és Gazdaságtudományi Egyetem, Villamosmérnöki és Informatikai Kar, Szélessávú Hírközlés és Villamosságtan Tanszék, 2018.

[2] Miklós Barnabás: Műholdfedélzeti QPSK adó tervezése és megvalósítása. BSC szakdolgozat. Budapesti Műszaki és Gazdaságtudományi Egyetem, Villamosmérnöki és Informatikai Kar, Szélessávú Hírközlés és Villamosságtan Tanszék, 2020.

[3] https://ieeexplore.ieee.org/document/7283837/

[4] Alaa M. Abdulhussein et al.: 2.4 GHz Microstrip Patch Antenna for S-Band Wireless Communications - 2021 J. Phys.: Conf. Ser. 2114 012029

[5] S. Gronemeyer and A. McBride, "MSK and Offset QPSK Modulation," in IEEE Transactions on Communications, vol. 24, no. 8, pp. 809-820, August 1976, doi: 10.1109/TCOM.1976.1093392.

[6] https://www.analog.com/media/en/technical-documentation/data-sheets/hmc453st89.pdf

[7] https://www.silabs.com/documents/public/data-sheets/efr32mg24-datasheet.pdf

[8] https://www.silabs.com/documents/public/user-guides/ug409-railtest-users-guide.pdf

[9] https://www.silabs.com/documents/public/reference-manuals/brd4187c-rm.pdf

[10] https://www.silabs.com/documents/public/application-notes/an928.2-efr32-series2-layout-design-guide.pdf

[11] https://www.silabs.com/documents/public/application-notes/an930.2-efr32-series-2.pdf

[12] https://www.analog.com/media/en/technical-documentation/data-sheets/ADRF6703.pdf

[13] http://ww1.microchip.com/downloads/en/DeviceDoc/60001324b.pdf

[14] https://www.analog.com/media/en/technical-documentation/data-sheets/ADL5606.pdf

[15] https://www.silabs.com/documents/public/data-sheets/efr32mg12-datasheet.pdf

[16] https://www.silabs.com/documents/public/application-notes/an0002.1-efr32-efm32-series-1-hardware-design-considerations.pdf

[17] https://www.silabs.com/documents/public/application-notes/an930.1-efr32-series-1.pdf

[18] https://www.ti.com/lit/ds/symlink/cc1352p.pdf?ts=1620844625885

[19] https://www.ti.com/lit/an/swra640e/swra640e.pdf

[20] https://www.ti.com/lit/zip/SWRC363

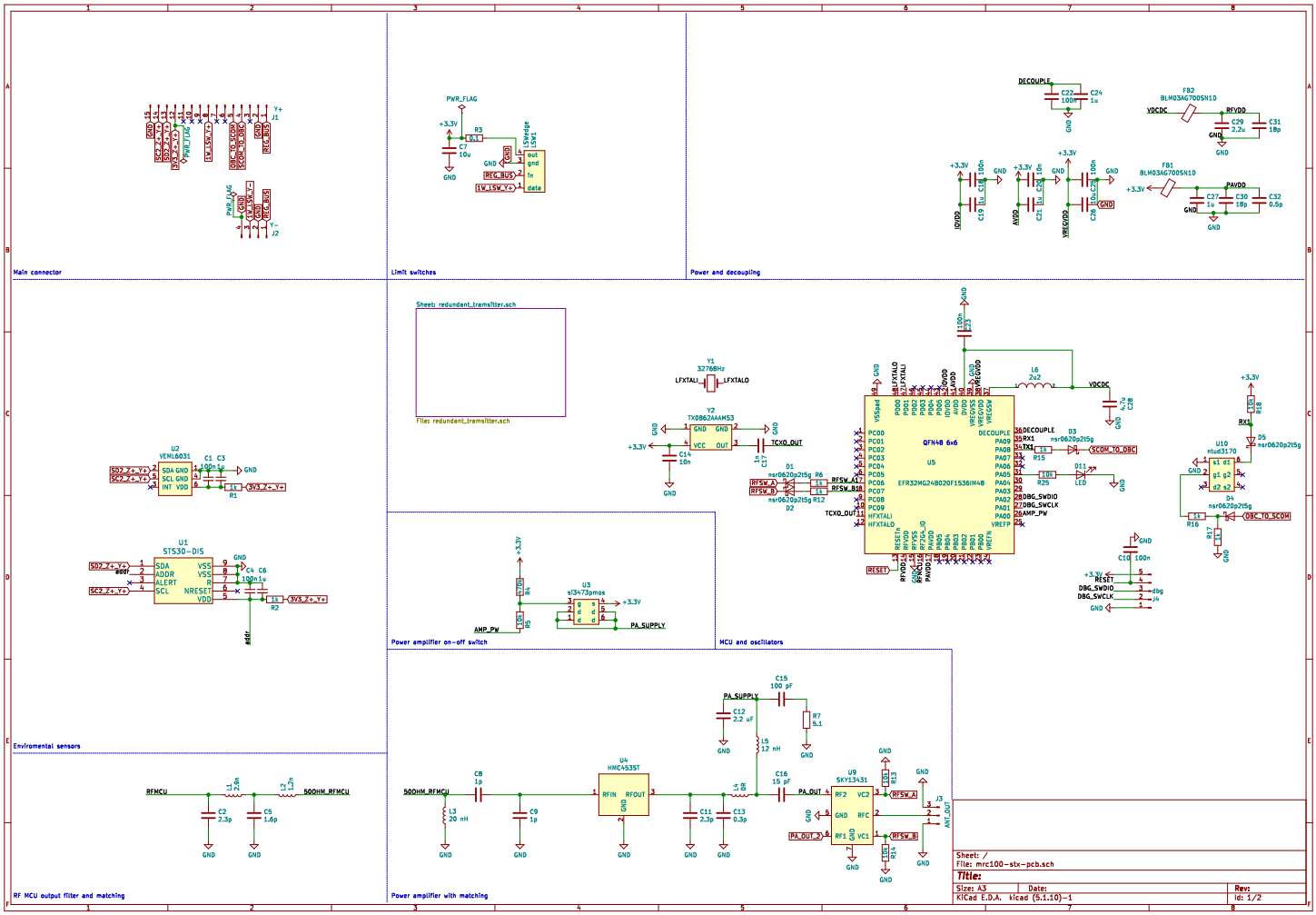
[21] https://www.minicircuits.com/pdfs/TAV2-501+.pdf

[22] https://www.nxp.com/docs/en/data-sheet/BGA6130.pdf

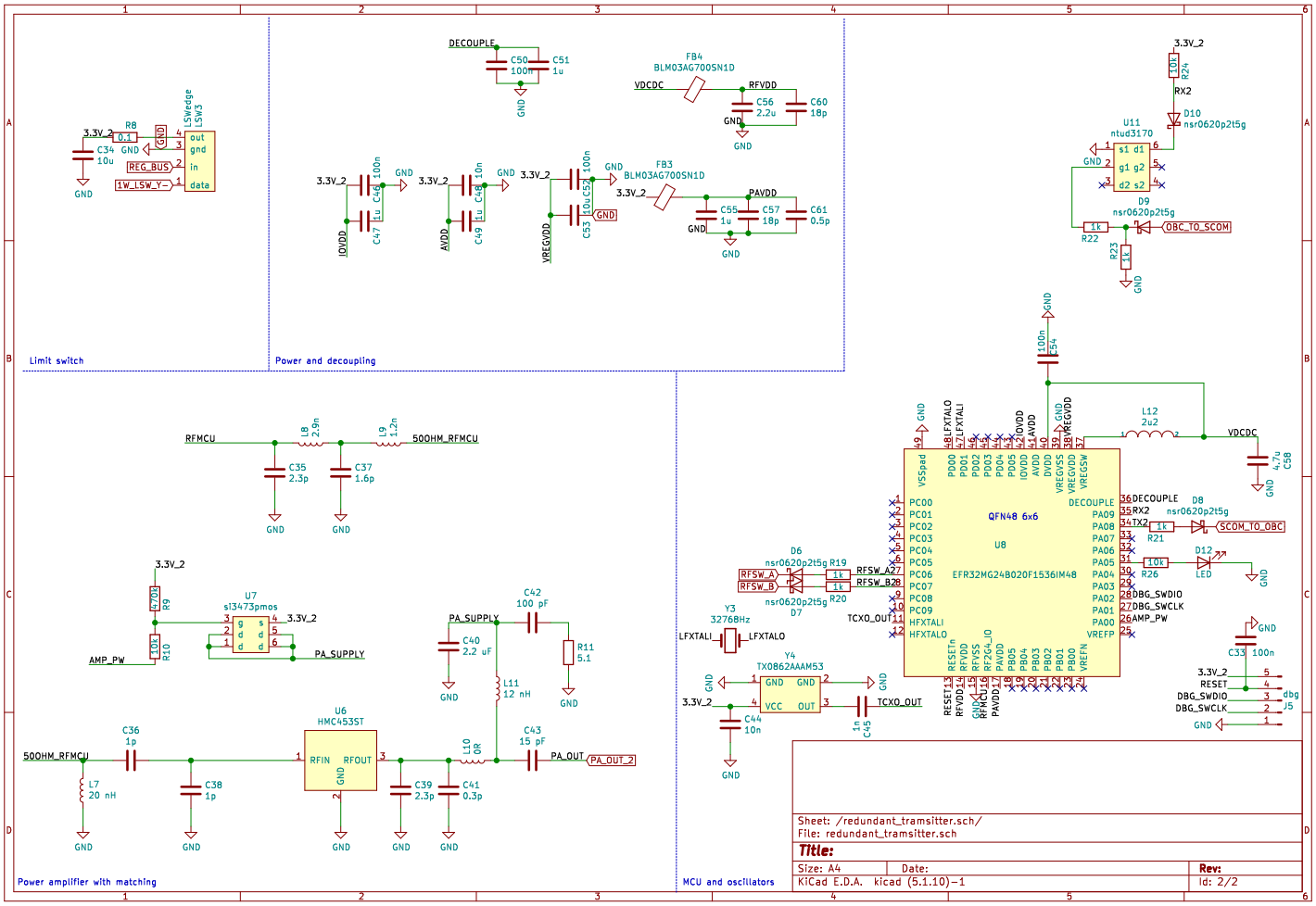
[23] https://www.st.com/resource/en/datasheet/pd20010-e.pdf

Ábrák jegyzéke

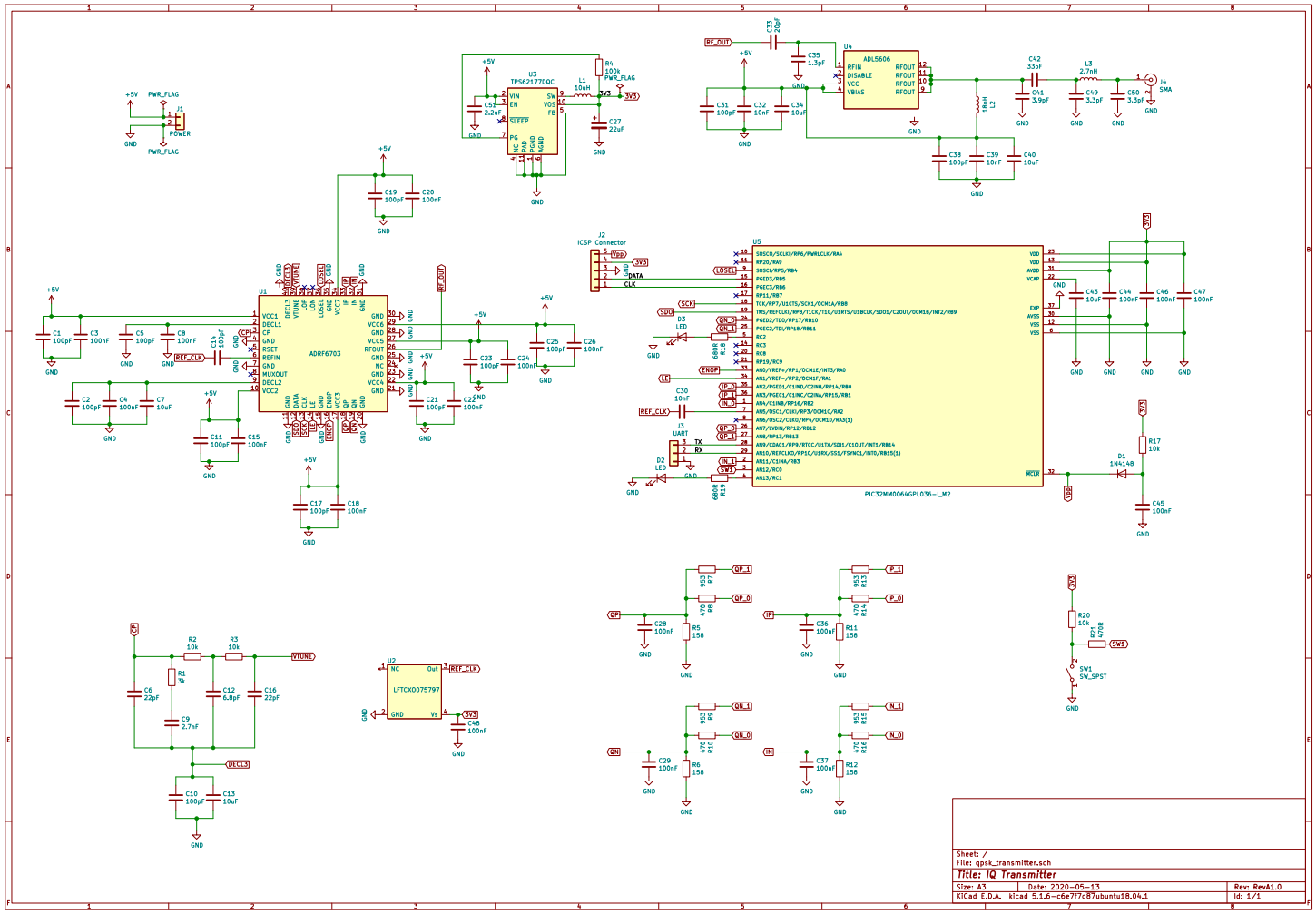
Függelék



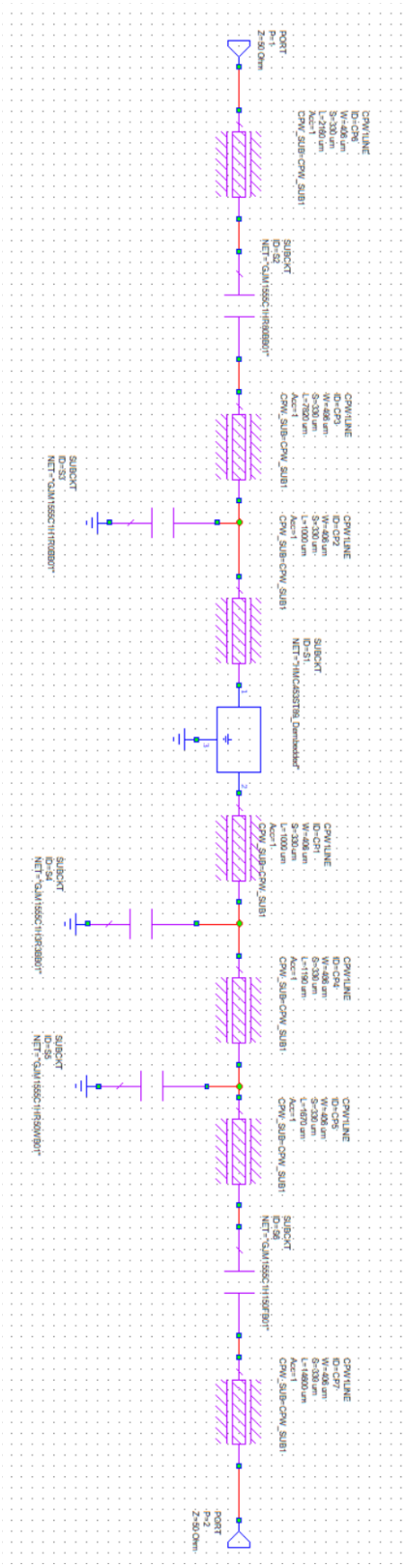
5.3. ábra. MRC100-STX kapcsolási rajz,primer adó



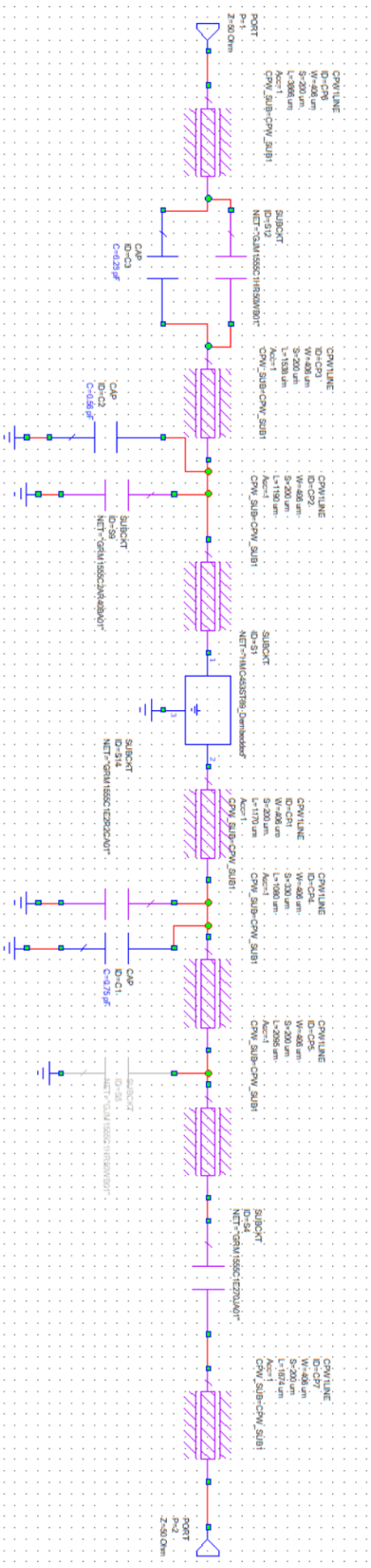
5.4. ábra. MRC100-STX kapcsolási rajz, redundáns adó



5.5. ábra. ADRF6703 adó kapcsolási rajz



5.6. ábra. AWR modell a HMC453st fejlesztőpanelhez



5.7. ábra. AWR modell a HMC453ST saját tervezésű tesztpanelhez

5.1. Listing. app_process.c

```
/***************************************************************************//**
* @file
* @brief app_process.c
*******************************************************************************
* # License
* <b>Copyright 2018 Silicon Laboratories Inc. www.silabs.com</b>
*******************************************************************************
*
* SPDX-License-Identifier: Zlib
*
* The licensor of this software is Silicon Laboratories Inc.
*
* This software is provided 'as-is', without any express or implied
* warranty. In no event will the authors be held liable for any damages
* arising from the use of this software.
*
* Permission is granted to anyone to use this software for any purpose,
* including commercial applications, and to alter it and redistribute it
* freely, subject to the following restrictions:
*
* 1. The origin of this software must not be misrepresented; you must not
* claim that you wrote the original software. If you use this software
* in a product, an acknowledgment in the product documentation would be
* appreciated but is not required.
* 2. Altered source versions must be plainly marked as such, and must not be
* misrepresented as being the original software.
* 3. This notice may not be removed or altered from any source distribution.
*
******************************************************************************/

// -----------------------------------------------------------------------------
// Includes
// -----------------------------------------------------------------------------
#include "sl_component_catalog.h"
#include "rail.h"
#include "em_device.h"
#include "em_chip.h"
#include "em_cmu.h"
#include "em_emu.h"
#include "em_eusart.h"
#include "em_gpio.h"

#include "sl_simple_led.h"
#include "sl_simple_led_instances.h"
#include "string.h"
#include "stdlib.h"
#include "sl_udelay.h"
#include "complex.h"

#define LED_TX sl_led_led0
#define LED_RX sl_led_led1
```

```
#include "mrc_com.h"
#include "app_process.h"
#include "encoder.h"
#include "app_config.h"


#if defined(SL_CATALOG_KERNEL_PRESENT)
#include "app_task_init.h"
#endif

// -------------------------------------------------------------------------
// Macros and Typedefs
// -------------------------------------------------------------------------
#define TX_BUFLEN 128

//format of TECST reply---------------
// $TECST,R(1 byte),TEMP(2*2 bytes),DATCOUNT(4*2 bytes)#
#define COMMA 1
#define STATUS_LEN 1 + COMMA
#define TEMP_LEN 2*2 + COMMA
#define DATCOUNT_LEN 4*2

#define TECST_LEN STATUS_LEN+TEMP_LEN+DATCOUNT_LEN

#define STATUS_I 0
#define TEMP_I STATUS_I+STATUS_LEN
#define DATCOUNT_I TEMP_I+TEMP_LEN
//-------------------------------------

//-------------------------------------


#define MODEM_TABLE_SIZE (10)

typedef enum{
    READY,
    BUSY,
    ERROR
}SCOM_state_t;

typedef enum{
    NONE,
    TX_ERROR,
    OTHER,
}SCOM_error_t;

typedef struct{

    uint8_t modulation_id[4];
    uint16_t channel_number;
}SCOM_modem_t;
```

```
// ------------------------------------------------------------------------------
// Static Function Declarations
// ------------------------------------------------------------------------------



// ------------------------------------------------------------------------------
// Global Variables
// ------------------------------------------------------------------------------

// Current position in buffer
uint32_t inpos = 0;
uint32_t outpos = 0;


volatile uint32_t tx_buf_inpos = 0;
volatile uint32_t tx_buf_outpos = 0;

uint8_t rx_data = 0;
static volatile SCOM_state_t global_state = READY;
static volatile bool command_arrived = false;
static volatile bool send_packet = false;

// ------------------------------------------------------------------------------
// Static Variables
// ------------------------------------------------------------------------------
static bool tx_msg_in_buf = false;
static volatile uint8_t tx_buffer[TX_BUFLEN];
static uint8_t state = 0;
static uint32_t bin_payload_len = 0;
static uint32_t tx_fifo_index = 0;
static uint32_t debug_char_counter = 0;
static RAIL_Status_t rail_status = RAIL_STATUS_NO_ERROR;

static uint8_t buf126[PAYLOAD_LENGTH];

static ra_word_t coded_packet[RA_MAX_CODE_LENGTH];
//static float test_softbits[PAYLOAD_LENGTH*2*16];
static uint16_t packet_counter = 0;

static uint8_t save_workaround[10];

const SCOM_modem_t modem_table[MODEM_TABLE_SIZE] = {

    {.modulation_id = "MOD0", .channel_number = 0, },
    {.modulation_id = "MOD1", .channel_number = 1, },
    {.modulation_id = "MOD2", .channel_number = 2,},
    {.modulation_id = "MOD3", .channel_number = 3,},
    {.modulation_id = "MOD4", .channel_number = 4,},
    {.modulation_id = "MOD5", .channel_number = 5,},
    {.modulation_id = "MOD6", .channel_number = 6,},
```

```
    {.modulation_id = "MOD7", .channel_number = 7,},
    {.modulation_id = "MOD8", .channel_number = 8,},
    {.modulation_id = "MOD9", .channel_number = 9,},

};

static SCOM_modem_t set_modem = modem_table[0];

static const uint8_t power_table[10] = {11,12,13,14,15,16,17,18,19,20};
static const uint8_t power_table_raw[10] = {37,42,49,56,63,70,90,114,137,254};


// -----------------------------------------------------------------------------
// Public Function Definitions
// -----------------------------------------------------------------------------

void EUSART0_RX_IRQHandler(void)
{

    rx_data = EUSART0->RXDATA;
    state = rcvd_stm(state,rx_data);
    if(state ==0xff) command_arrived = true;
    EUSART_IntClear(EUSART0, EUSART_IF_RXFL);
    debug_char_counter++;
    //sl_led_toggle(&LED_RX);

}

void EUSART_tx_loadbuff(uint8_t * message,uint8_t len){

    for(int i = 0; i<len;i++){
        tx_buffer[tx_buf_inpos++] = message[i];
    }
    // Disable receive FIFO level interrupt
    //EUSART_IntEnable(EUSART0, EUSART_IEN_TXFL);
    tx_msg_in_buf = true;
}

void EUSART_tx_process(){

    if(tx_msg_in_buf){
        uint32_t fifo_full = EUSART0->STATUS & EUSART_STATUS_TXFL;
        if(fifo_full){
            if (tx_buf_outpos < tx_buf_inpos)
            {
                EUSART0->TXDATA = tx_buffer[tx_buf_outpos++];
                //sl_led_toggle(&LED_TX);
            }else{
                tx_buf_outpos = tx_buf_inpos = tx_msg_in_buf = 0;
            }
        }
```

```
    }

}
void SCOM_load_modem_setup(RAIL_Handle_t rail_handle, uint8_t modulation_id[4],
    ↪ uint8_t power_index_c){

    //uint8_t power_dbm = 0;
    //uint8_t power_index = abs(power_index_c - '0');
    //if(power_index<10) power_dbm = power_table[power_index];
    //RAIL_SetTxPowerDbm(rail_handle, power_dbm*10);

    uint8_t power_raw = 0;
    uint8_t power_index = abs(power_index_c - '0');
    if(power_index<10) power_raw = power_table_raw[power_index];
    RAIL_SetTxPower(rail_handle,power_raw);

    for(int i =0; i<MODEM_TABLE_SIZE;i++){
        if(!memcmp(modem_table[i].modulation_id,modulation_id,4)){
            memcpy(set_modem.modulation_id,modem_table[i].modulation_id,4);
            set_modem.channel_number = modem_table[i].channel_number;
        }
    }
}

void SCOM_handle_rqcst(SCOM_state_t state){

    uint8_t payload_buffer[TECST_LEN];
    int16_t temp = 0xffff;
    switch(state){
        case READY:
        payload_buffer[STATUS_I] = 'R';
        break;
        case BUSY:
        payload_buffer[STATUS_I] = 'B';
        break;
        case ERROR:
        payload_buffer[STATUS_I] = 'E';
        break;

    }
    payload_buffer[STATUS_I+STATUS_LEN - 1] = ',';
    u16_to_hex_be(&payload_buffer[TEMP_I],temp);
    payload_buffer[TEMP_I+TEMP_LEN - 1] = ',';
    u32_to_hex_be(&payload_buffer[DATCOUNT_I],bin_payload_len);


    create_reply_msg((uint8_t*)"TECST",payload_buffer,TECST_LEN);
    EUSART_tx_loadbuff(reply_msg,reply_msg_len);

}
void SCOM_toggle_amp(bool turn_on){
    static uint8_t current_state = 0;
```

```
    if((turn_on) && (!current_state)){

        GPIO_PinModeSet(AMP_PORT, AMP_PW, gpioModePushPull, 0);
        current_state = 1;
    }else if ((!turn_on) && (current_state)){
        // GPIO_PinModeSet(RFS_PORT, RFSW_A, gpioModePushPull, RFSW_B_PULL);
        // GPIO_PinModeSet(RFS_PORT, RFSW_B, gpioModePushPull, RFSW_A_PULL);
        GPIO_PinModeSet(AMP_PORT, AMP_PW, gpioModePushPull, 1);
        current_state = 0;
    }

}

/**************************************************************************
* Application state machine, called infinitely
**************************************************************************/
void app_process_action(RAIL_Handle_t rail_handle)
{
    (void) rail_handle;

    ////////////////////////////////////////////////////////////////////////
    // Put your application code here! //
    // This is called infinitely. //
    // Do not call blocking functions from here! //
    ////////////////////////////////////////////////////////////////////////
    //
    switch(global_state){
        case READY:
        //SCOM_toggle_amp(false);
        if(command_arrived ){
            command_arrived = false;
            if(!memcmp("RQCST",rcvd_cmd,5)){

                SCOM_handle_rqcst(READY);

            }else if(!memcmp((uint8_t*)"SBUFF",rcvd_cmd,5)) {
                bin_payload_len = rcvd_payload_length;
                create_reply_msg((uint8_t*)"ACKMS",NULL,0);
                EUSART_tx_loadbuff(reply_msg,reply_msg_len);
                memcpy(save_workaround,rcvd_payload,10);
            }else if(!memcmp("SENDB",rcvd_cmd,5)) {

                uint8_t sendb_mod[4];
                uint8_t sendb_power = rcvd_payload[0];
                memcpy(sendb_mod,rcvd_payload+2,4);

                SCOM_load_modem_setup(rail_handle,sendb_mod,sendb_power);
                create_reply_msg((uint8_t*)"ACKMS",NULL,0);
                EUSART_tx_loadbuff(reply_msg,reply_msg_len);
                global_state = BUSY;
                send_packet = true;
                memcpy(rcvd_payload,save_workaround,10);
```

```
            //global_state = READY;
        }else{
            ;
        }
    }
    break;
    case BUSY:

    if(command_arrived){
        command_arrived = false;
        if(!memcmp("RQCST",rcvd_cmd,5)){
            SCOM_handle_rqcst(BUSY);
        }
    }

    if((!send_packet) && (bin_payload_len>0)){
        //current packet tx not finished yet
        break;
    }else if (bin_payload_len == 0){
        SCOM_toggle_amp(false);
        global_state = READY;
        break;
    }



    if(DATA_LEN>=bin_payload_len){ //handling last packet, end of tx
        memset(buf126,0,PAYLOAD_LENGTH);
        memcpy(buf126+PACKET_COUNTER_LEN+CRC_LEN,rcvd_payload+tx_fifo_index,
            ↪ bin_payload_len);

        //RAIL_WriteTxFifo(rail_handle, rcvd_payload+tx_fifo_index,
            ↪ bin_payload_len, false);
        uint8_t fill_zeroes[PAYLOAD_LENGTH];
        memset(fill_zeroes,0,PAYLOAD_LENGTH);

        memcpy(buf126+PACKET_COUNTER_LEN+CRC_LEN+bin_payload_len,fill_zeroes,
            ↪ DATA_LEN - bin_payload_len);
        memcpy((uint16_t*)buf126,&packet_counter,sizeof(packet_counter));
        global_state = READY;
        tx_fifo_index = 0;
        bin_payload_len = 0;
        packet_counter=0;

        //sl_udelay_wait(1000000);//TODO:for debugging purposes, delays the
            ↪ last packet otherwise railtest doesnt prints it
    }else{

        memset(buf126,0,PAYLOAD_LENGTH);
        memcpy(buf126+PACKET_COUNTER_LEN+CRC_LEN,rcvd_payload+tx_fifo_index,
            ↪ DATA_LEN);
        memcpy((uint16_t*)buf126,&packet_counter,sizeof(packet_counter));
```

```
}
//memset(buf126,0xf0,PAYLOAD_LENGTH);
//sl_udelay_wait(210);
SCOM_calculate_crc(buf126,CRC_CALC);
SCOM_encode((ra_word_t *)buf126,coded_packet,PAYLOAD_LENGTH/2);

//memset(coded_packet,0xf0,ra_code_length*2);

// //decode
// memset(buf126,0,PAYLOAD_LENGTH);
// SCOM_decode(coded_packet,test_softbits,(ra_word_t*)buf126);
// volatile crc_code_t crc_result = SCOM_calculate_crc(buf126,CRC_CHECK);
// //----
//RAIL_WriteTxFifo(rail_handle, rcvd_payload+tx_fifo_index,
    ↪ PAYLOAD_LENGTH, false);
RAIL_WriteTxFifo(rail_handle,(uint8_t*) coded_packet, (uint16_t)
    ↪ ra_code_length*2, false);
send_packet=false; // set to false until the transmission is complete,
    ↪ set to true in handler
SCOM_toggle_amp(true);
rail_status = RAIL_StartTx(rail_handle, set_modem.channel_number,
    ↪ RAIL_TX_OPTIONS_DEFAULT, NULL);

if ( rail_status == RAIL_STATUS_NO_ERROR ) {

    if(DATA_LEN>bin_payload_len){
        bin_payload_len = 0;
        tx_fifo_index = 0;
        packet_counter=0;
    }else{
        bin_payload_len -=DATA_LEN;
        tx_fifo_index+=DATA_LEN;
        packet_counter++;
    }

}else{
    sl_led_turn_on(&sl_led_led1);
    RAIL_ResetFifo(rail_handle,true,false);
    global_state = ERROR;
    packet_counter=0;
}
break;
case ERROR:
SCOM_toggle_amp(false);
if(command_arrived){
    command_arrived = false;
    if(!memcmp("RQCST",rcvd_cmd,5)){
        SCOM_handle_rqcst(ERROR);
        global_state = READY;
        send_packet = false;
    }
```

```
        }

        break;
        default:
        SCOM_toggle_amp(false);
        global_state = ERROR;
        break;
    }

    EUSART_tx_process();

    // Enable receive FIFO level interrupt (defaults to one frame)
}

/******************************************************************************
* RAIL callback, called if a RAIL event occurs
******************************************************************************/
void sl_rail_util_on_event(RAIL_Handle_t rail_handle, RAIL_Events_t events)
{
    (void) rail_handle;
    (void) events;

    ////////////////////////////////////////////////////////////////////////////
    // Put your RAIL event handling here! //
    // This is called from ISR context. //
    // Do not call blocking functions from here! //
    ////////////////////////////////////////////////////////////////////////////
    ///
    ///
    ///
    if ( events & RAIL_EVENTS_TX_COMPLETION ) {
        send_packet = true;
        if ( events & RAIL_EVENT_TX_PACKET_SENT ) {
            sl_led_toggle(&sl_led_led0);
            send_packet = true;
            if(bin_payload_len==0){
                SCOM_toggle_amp(false);
            }
        } else {
            RAIL_ResetFifo(rail_handle,true,false);
            SCOM_toggle_amp(false);
            global_state = ERROR;
            sl_led_turn_on(&sl_led_led1);
            packet_counter=0;//all other events in RAIL_EVENTS_TX_COMPLETION are
                ↪ errors
        }
    }


    #if defined(SL_CATALOG_KERNEL_PRESENT)
    app_task_notify();
    #endif
```

```
}
```

5.2. Listing. app_process.h

```
/***************************************************************************//**
* @file
* @brief app_process.h
*******************************************************************************
* # License
* <b>Copyright 2018 Silicon Laboratories Inc. www.silabs.com</b>
*******************************************************************************
*
* SPDX-License-Identifier: Zlib
*
* The licensor of this software is Silicon Laboratories Inc.
*
* This software is provided 'as-is', without any express or implied
* warranty. In no event will the authors be held liable for any damages
* arising from the use of this software.
*
* Permission is granted to anyone to use this software for any purpose,
* including commercial applications, and to alter it and redistribute it
* freely, subject to the following restrictions:
*
* 1. The origin of this software must not be misrepresented; you must not
* claim that you wrote the original software. If you use this software
* in a product, an acknowledgment in the product documentation would be
* appreciated but is not required.
* 2. Altered source versions must be plainly marked as such, and must not be
* misrepresented as being the original software.
* 3. This notice may not be removed or altered from any source distribution.
*
******************************************************************************/
#ifndef APP_PROCESS_H
#define APP_PROCESS_H

// -----------------------------------------------------------------------------
// Includes
// -----------------------------------------------------------------------------
#include "rail.h"
#include "encoder.h"

// -----------------------------------------------------------------------------
// Macros and Typedefs
// -----------------------------------------------------------------------------

// -----------------------------------------------------------------------------
// Global Variables
// -----------------------------------------------------------------------------

// -----------------------------------------------------------------------------
// Public Function Declarations
// -----------------------------------------------------------------------------
```

```
/**************************************************************************//**
* The function is used for Application logic.
*
* @param rail_handle RAIL handle
* @returns None
*
* The function is used for Application logic.
* It is called infinitely.
*****************************************************************************/
void app_process_action(RAIL_Handle_t rail_handle);

#endif // APP_PROCESS_H
```

5.3. Listing. app_init.c

```
/**************************************************************************//**
* @file
* @brief app_init.c
*******************************************************************************
* # License
* <b>Copyright 2018 Silicon Laboratories Inc. www.silabs.com</b>
*******************************************************************************
*
* SPDX-License-Identifier: Zlib
*
* The licensor of this software is Silicon Laboratories Inc.
*
* This software is provided 'as-is', without any express or implied
* warranty. In no event will the authors be held liable for any damages
* arising from the use of this software.
*
* Permission is granted to anyone to use this software for any purpose,
* including commercial applications, and to alter it and redistribute it
* freely, subject to the following restrictions:
*
* 1. The origin of this software must not be misrepresented; you must not
* claim that you wrote the original software. If you use this software
* in a product, an acknowledgment in the product documentation would be
* appreciated but is not required.
* 2. Altered source versions must be plainly marked as such, and must not be
* misrepresented as being the original software.
* 3. This notice may not be removed or altered from any source distribution.
*
******************************************************************************/

// -----------------------------------------------------------------------------
// Includes
// -----------------------------------------------------------------------------
#include "sl_rail_util_init.h"
#include "em_device.h"
#include "em_chip.h"
#include "em_cmu.h"
#include "em_emu.h"
```

```c
#include "em_eusart.h"
#include "em_gpio.h"
#include "sl_board_control.h"

#include "sl_simple_led.h"
#include "sl_simple_led_instances.h"

#include "app_config.h"


// -----------------------------------------------------------------------------
// Macros and Typedefs
// -----------------------------------------------------------------------------


static uint8_t rail_tx_fifo[FIFO_LENGTH];

// -----------------------------------------------------------------------------
// Static Function Declarations
// -----------------------------------------------------------------------------
/***************************************************************************//**
* @brief
* CMU initialization
******************************************************************************/
void initCMU(void)
{
    // Enable clock to GPIO and EUSART1
    CMU_ClockEnable(cmuClock_GPIO, true);
    CMU_ClockEnable(cmuClock_EUSART0, true);
}

void initGPIO(void)
{
    // Configure the EUSART TX pin to the board controller as an output
    GPIO_PinModeSet(EUSART0_PORT, EUSART0_TX_PIN, gpioModePushPull, 1);

    // Configure the EUSART RX pin to the board controller as an input
    GPIO_PinModeSet(EUSART0_PORT, EUSART0_RX_PIN, gpioModeInput, 0);

    /*
    * Configure the BCC_ENABLE pin as output and set high. This enables
    * the virtual COM port (VCOM) connection to the board controller and
    * permits serial port traffic over the debug connection to the host
    * PC.
    *
    * To disable the VCOM connection and use the pins on the kit
    * expansion (EXP) header, comment out the following line.
    */
    sl_board_enable_vcom();
}

void initEUSART0(void)
```

```
{
    // Default asynchronous initializer (115.2 Kbps, 8N1, no flow control)
    EUSART_UartInit_TypeDef init = EUSART_UART_INIT_DEFAULT_HF;
    // init.baudrate = 9600;
    // init.oversampling = eusartOVS4;

    // Route EUSART1 TX and RX to the board controller TX and RX pins
    GPIO->EUSARTROUTE[EUSART_NUM(EUSART0)].TXROUTE = (EUSART0_PORT <<
        ↪ _GPIO_EUSART_TXROUTE_PORT_SHIFT)
    | (EUSART0_TX_PIN << _GPIO_EUSART_TXROUTE_PIN_SHIFT);
    GPIO->EUSARTROUTE[EUSART_NUM(EUSART0)].RXROUTE = (EUSART0_PORT <<
        ↪ _GPIO_EUSART_RXROUTE_PORT_SHIFT)
    | (EUSART0_RX_PIN << _GPIO_EUSART_RXROUTE_PIN_SHIFT);

    // Enable RX and TX signals now that they have been routed
    GPIO->EUSARTROUTE[EUSART_NUM(EUSART0)].ROUTEEN = GPIO_EUSART_ROUTEEN_RXPEN |
        ↪  GPIO_EUSART_ROUTEEN_TXPEN;

    // Configure and enable EUSART1 for high-frequency (EM0/1) operation
    EUSART_UartInitHf(EUSART0, &init);
    EUSART_BaudrateSet(EUSART0,0,UART_BAUDRATE);

    // Enable NVIC USART sources
    NVIC_ClearPendingIRQ(EUSART0_RX_IRQn);
    NVIC_EnableIRQ(EUSART0_RX_IRQn);
    // NVIC_ClearPendingIRQ(EUSART0_TX_IRQn);
    // NVIC_EnableIRQ(EUSART0_TX_IRQn);
    EUSART_IntEnable(EUSART0, EUSART_IEN_RXFL);
}
// -------------------------------------------------------------------------
// Global Variables
// -------------------------------------------------------------------------



// -------------------------------------------------------------------------
// Static Variables
// -------------------------------------------------------------------------

// -------------------------------------------------------------------------
// Public Function Definitions
// -------------------------------------------------------------------------


/******************************************************************************
* The function is used for some basic initialization related to the app.
*****************************************************************************/
RAIL_Handle_t app_init(void)
{
    // Get RAIL handle, used later by the application
    RAIL_Handle_t rail_handle = sl_rail_util_get_handle(
        ↪ SL_RAIL_UTIL_HANDLE_INST0);
```

```
    //
        ↪ /////////////////////////////////////////////////////////////////////////
        ↪
    // Put your application init code here! //
    // This is called once during start-up. //
    //
        ↪ /////////////////////////////////////////////////////////////////////////
        ↪
    // Chip errata
    CHIP_Init();

    // Initialize GPIO and EUSART1
    initCMU();
    initGPIO();
    initEUSART0();



    RAIL_ConfigEvents(rail_handle, RAIL_EVENTS_ALL,
    RAIL_EVENTS_TX_COMPLETION | RAIL_EVENTS_RX_COMPLETION);
    RAIL_SetTxFifo(rail_handle,rail_tx_fifo,0,FIFO_LENGTH);

    GPIO_PinModeSet(RFS_PORT, RFSW_A, gpioModePushPull, RFSW_A_PULL);
    GPIO_PinModeSet(RFS_PORT, RFSW_B, gpioModePushPull, RFSW_B_PULL);
    RAIL_SetFixedLength(rail_handle,PAYLOAD_LENGTH*2);

    return rail_handle;
}

// -----------------------------------------------------------------------------
// Static Function Definitions
// -----------------------------------------------------------------------------
```

5.4. Listing. app_init.h

```
/***************************************************************************//**
* @file
* @brief app_init.h
*******************************************************************************
* # License
* <b>Copyright 2018 Silicon Laboratories Inc. www.silabs.com</b>
*******************************************************************************
*
* SPDX-License-Identifier: Zlib
*
* The licensor of this software is Silicon Laboratories Inc.
*
* This software is provided 'as-is', without any express or implied
* warranty. In no event will the authors be held liable for any damages
* arising from the use of this software.
*
* Permission is granted to anyone to use this software for any purpose,
```

```
* including commercial applications, and to alter it and redistribute it
* freely, subject to the following restrictions:
*
* 1. The origin of this software must not be misrepresented; you must not
* claim that you wrote the original software. If you use this software
* in a product, an acknowledgment in the product documentation would be
* appreciated but is not required.
* 2. Altered source versions must be plainly marked as such, and must not be
* misrepresented as being the original software.
* 3. This notice may not be removed or altered from any source distribution.
*
******************************************************************************/
#ifndef APP_INIT_H
#define APP_INIT_H

// -----------------------------------------------------------------------------
// Includes
// -----------------------------------------------------------------------------
#include "rail.h"

// -----------------------------------------------------------------------------
// Macros and Typedefs
// -----------------------------------------------------------------------------

// -----------------------------------------------------------------------------
// Global Variables
// -----------------------------------------------------------------------------

// -----------------------------------------------------------------------------
// Public Function Declarations
// -----------------------------------------------------------------------------
/******************************************************************************//**
* The function is used for application initialization.
*
* @param None
* @returns RAIL_Handle_t RAIL handle
*********************************************************************************/
RAIL_Handle_t app_init(void);

#endif // APP_INIT_H
```

5.5. Listing. app_config.h

```
/*
* app_config.h
*
* Created on: 2022. okt. 10.
* Author: bamiklos
*/

#ifndef APP_CONFIG_H_
#define APP_CONFIG_H_
```

```
#include "em_gpio.h"
#include "sl_simple_led.h"
#include "sl_simple_led_instances.h"

#define PAYLOAD_LENGTH (126)
#define PACKET_COUNTER_LEN (2)
#define CRC_LEN (2)
#define DATA_LEN PAYLOAD_LENGTH - CRC_LEN - PACKET_COUNTER_LEN
#define CODED_PAYLOAD_LENGTH (PAYLOAD_LENGTH*2)


#define EUSART0_TX_PIN 8
#define EUSART0_RX_PIN 9
#define EUSART0_PORT gpioPortA

#ifndef LED_INSTANCE
#define LED_INSTANCE sl_led_led0 //TODO: leds are for debugging
#endif

#define RFS_PORT gpioPortC
#define RFSW_A 6
#define RFSW_B 7
//a 1 b 0, felirat oldal
//a 0 b 1 uj oldal
#define RFSW_A_PULL 0
#define RFSW_B_PULL 1

#define AMP_PW 0
#define AMP_PORT gpioPortA

#define FIFO_LENGTH 4096

#define UART_BAUDRATE 500000 //921600



#endif /* APP_CONFIG_H_ */
```

5.6. Listing. mrc_com.c

```
#include <mrc_com.h>


bool hex_be_to_u8(uint8_t * str, uint8_t * num)
{
    int i;
    uint8_t tmp_char;

    *num = 0;
    if (str == NULL) return false;
    // '0' == 48
    // '9' == 57
```

```
    // 'A' == 65
    // 'F' == 70
    for (i=0; i<2; i++) {
        tmp_char = str[i];
        tmp_char -= '0';
        if (tmp_char > 9) {
            tmp_char -= 7; // 'A' - '0' - 7 == 10
            if (tmp_char > 15) return false;
        }

        *num = *num << 4;
        *num += tmp_char;
    }

    return true;
}

bool hex_be_to_u16(uint8_t * str, uint16_t * num)
{
    int i;
    uint8_t tmp_char;

    *num = 0;
    if (str == NULL) return false;
    // '0' == 48
    // '9' == 57
    // 'A' == 65
    // 'F' == 70
    for (i=0; i<4; i++) {
        tmp_char = str[i];
        tmp_char -= '0';
        if (tmp_char > 9) {
            tmp_char -= 7; // 'A' - '0' - 7 == 10
            if (tmp_char > 15) return false;
        }

        *num = *num << 4;
        *num += tmp_char;
    }

    return true;
}
bool hex_be_to_u32(uint8_t * str, uint32_t * num)
{
    int i;
    uint8_t tmp_char;

    *num = 0;
    if (str == NULL) return false;
    // '0' == 48
    // '9' == 57
    // 'A' == 65
```

```
    // 'F' == 70
    for (i=0; i<8; i++) {
        tmp_char = str[i];
        tmp_char -= '0';
        if (tmp_char > 9) {
            tmp_char -= 7; // 'A' - '0' - 7 == 10
            if (tmp_char > 15) return false;
        }

        *num = *num << 4;
        *num += tmp_char;
    }

    return true;
}

void u8_to_hex_be(uint8_t * str, uint8_t num)
{
    int i;

    if (str == NULL) return;
    // '0' == 48
    // '9' == 57
    // 'A' == 65
    // 'F' == 70
    for (i=1; i>=0; i--) {
        str[i] = (num & 0x0F) + (((num & 0x0F) > 9) ? 55 : 48);
        num >>= 4;
    }
}

void u16_to_hex_be(uint8_t * str, uint16_t num)
{
    int i;

    if (str == NULL) return;
    // '0' == 48
    // '9' == 57
    // 'A' == 65
    // 'F' == 70
    for (i=3; i>=0; i--) {
        str[i] = (num & 0x0F) + (((num & 0x0F) > 9) ? 55 : 48);
        num >>= 4;
    }
}

void u32_to_hex_be(uint8_t * str, uint32_t num)
{
    int i;

    if (str == NULL) return;
    // '0' == 48
```

```
    // '9' == 57
    // 'A' == 65
    // 'F' == 70
    for (i=7; i>=0; i--) {
        str[i] = (num & 0x0F) + (((num & 0x0F) > 9) ? 55 : 48);
        num >>= 4;
    }
}

uint8_t reply_msg[MAX_REPLY_PAYLOAD_LENGTH*2+20];
uint16_t reply_msg_len;

void create_reply_msg(uint8_t * cmd, uint8_t * payload, uint16_t payload_length)
{
    int mpt = 0;
    reply_msg[mpt] = '$';
    mpt++;
    for (int i=0; i<5; i++)
    {
        reply_msg[mpt] = cmd[i];
        mpt++;
    }
    if (payload_length > 0)
    {
        reply_msg[mpt] = ',';
        mpt++;
        mpt += binary_to_mslip(reply_msg + mpt, payload, payload_length);
    }
    reply_msg[mpt] = '#';
    mpt++;
    u8_to_hex_be(reply_msg+mpt, rcvd_serial);
    mpt += 2;
    uint16_t chksum[2];
    fletcher16_init(chksum);
    for (int i=1; i<mpt; i++) fletcher16_update(chksum, reply_msg[i]);
    reply_msg[mpt] = '*';
    mpt++;
    u16_to_hex_be(reply_msg+mpt, fletcher16_get_chksum(chksum));
    mpt += 4;
    reply_msg[mpt] = '\r';
    mpt++;
    reply_msg[mpt] = '\n';
    mpt++;
    reply_msg_len = mpt;
}


uint8_t rcvd_cmd[6];
uint8_t rcvd_payload[MAX_RECEIVE_PAYLOAD_LENGTH+1];
uint32_t rcvd_payload_length;
uint8_t rcvd_serial;
```

```
uint8_t rcvd_stm(uint8_t state, uint8_t chr)
{
    static uint16_t chksum[2];
    static uint8_t tmpstr[4];
    static bool mslip_escape;
    uint8_t tmp_u8;
    uint16_t tmp_u16;

    if (chr == '$') state = 0;

    switch (state) {
        case 0: // wait for dollar
        if (chr != '$') return 0;
        fletcher16_init(chksum);
        mslip_escape = false;
        return 1;
        case 1: // command; only uppercase letters and numbers
        case 2:
        case 3:
        case 4:
        case 5:
        if (chr < '0') return 0;
        // between 9 & A there are: :;<=>?@
        // these are invalid, but dont care here
        if (chr > 'Z') return 0;
        fletcher16_update(chksum, chr);
        rcvd_cmd[state - 1] = chr;
        return (state + 1);
        case 6: // ',' or '#'
        rcvd_cmd[5] = 0;
        fletcher16_update(chksum, chr);
        rcvd_payload_length = 0;
        if (chr == '#') return 8;
        if (chr == ',') return 7;
        return 0;
        case 7: // message body
        fletcher16_update(chksum, chr);
        if (chr == '$') // invalid character
        {
            fletcher16_init(chksum);
            return 0;
        }
        if (chr == '*') return 0; // invalid character
        if (chr == '\r') return 0; // invalid character
        if (chr == '\n') return 0; // invalid character
        if (chr == '#') return 8;
        if (chr == '|')
        {
            mslip_escape = true;
            return 7;
        }
        if (mslip_escape)
```

```
        {
            mslip_escape = false;
            switch (chr)
            {
                case 'S': rcvd_payload[rcvd_payload_length] = '$'; break;
                case '+': rcvd_payload[rcvd_payload_length] = '#'; break;
                case 'X': rcvd_payload[rcvd_payload_length] = '*'; break;
                case 'R': rcvd_payload[rcvd_payload_length] = '\r'; break;
                case 'N': rcvd_payload[rcvd_payload_length] = '\n'; break;
                case '-': rcvd_payload[rcvd_payload_length] = '|'; break;
                default: rcvd_payload[rcvd_payload_length] = 0; break;
            }
        }
        else
        {
            rcvd_payload[rcvd_payload_length] = chr;
        }
        if (rcvd_payload_length >= MAX_RECEIVE_PAYLOAD_LENGTH) // too much data
            ↪ comming
        {
            return 0;
        }
        rcvd_payload_length++;
        return 7; // stay in this state
        case 8: // serialnum1
        fletcher16_update(chksum, chr);
        tmpstr[0] = chr;
        return 9;
        case 9: // serialnum2
        fletcher16_update(chksum, chr);
        tmpstr[1] = chr;
        if (!hex_be_to_u8(tmpstr, &rcvd_serial)) return 0; // failed conversion
        return 10;
        case 10: // asterix
        if (chr != '*') return 0;
        return 11;
        case 11: // checksum
        case 12:
        case 13:
        case 14:
        tmpstr[state-11] = chr;
        return (state + 1);
        case 15: // CR
        if (chr != '\r') return 0;
        // also test the checksum:
        if (!hex_be_to_u16(tmpstr, &tmp_u16)) return 0; // failed conversion
        if (tmp_u16 != fletcher16_get_chksum(chksum)) return 0; // checksum
            ↪ doesnt match
        return 16;
        case 16: // LF
        if (chr != '\n') return 0;
        return 0xFF; // DONE!
```

```
        default:
        return 0;
    }
}
```

5.7. Listing. mrc_com.h

```
#ifndef MRC_COM_H
#define MRC_COM_H

#include <stdbool.h>
#include <stdint.h>
#include <stddef.h>
#include <checksum.h>
#include <mslip.h>
#ifdef __cplusplus
extern "C" {
    #endif

    #define MAX_RECEIVE_PAYLOAD_LENGTH (198000)
    #define MAX_REPLY_PAYLOAD_LENGTH (32)

    bool hex_be_to_u8(uint8_t * str, uint8_t * num);


    bool hex_be_to_u16(uint8_t * str, uint16_t * num);

    bool hex_be_to_u32(uint8_t * str, uint32_t * num);


    void u8_to_hex_be(uint8_t * str, uint8_t num);

    void u16_to_hex_be(uint8_t * str, uint16_t num);

    void u32_to_hex_be(uint8_t * str, uint32_t num);



    extern uint8_t reply_msg[MAX_REPLY_PAYLOAD_LENGTH*2+20];
    extern uint16_t reply_msg_len;

    void create_reply_msg(uint8_t * cmd, uint8_t * payload, uint16_t
        ↪ payload_length);


    extern uint8_t rcvd_cmd[6];
    extern uint8_t rcvd_payload[MAX_RECEIVE_PAYLOAD_LENGTH+1];
    extern uint32_t rcvd_payload_length;
    extern uint8_t rcvd_serial;

    uint8_t rcvd_stm(uint8_t state, uint8_t chr);
```

```
    #ifdef __cplusplus
}
#endif

#endif /* MRC_COM_H */
```

5.8. Listing. obc_emulator.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>

#include "obc_emulator.h"
#include "definitions.h"



#define TEST_SEQ_LENGTH 400000
//#define SEQ_BYTES_LENGTH 400000
//#define TEST_SEQ_LENGTH 160
#define SEQ_BYTES_LENGTH 1500*122
void print_payload(uint8_t * payload_buffer, uint16_t len){
    fprintf(stderr,"\n printing payload: ");
    for(int i = 0; i<len; i++){

        fprintf(stderr, " %.2x,", payload_buffer[i]);

    }
    fprintf(stderr,"\n");
}

int main()
{

    size_t len = 0;

    ssize_t read;
    uint8_t reply[64];
    const char * serial = "COM8";
    uint32_t baud_rate = 500000;///921600;
    HANDLE fd = open_serial_port(serial, baud_rate);
    if(fd<0) return -1;


    uint8_t bin_payload[SEQ_BYTES_LENGTH+1];
    uint8_t mslip_payload[TEST_SEQ_LENGTH+1]; // this is bad, but for testing
        ↪ its okay
    memset(bin_payload,0,SEQ_BYTES_LENGTH+1);
    memset(mslip_payload,0,TEST_SEQ_LENGTH+1);
```

```
rcvd_serial=0;
uint8_t sendb_power = '0';
uint8_t sendb_mod[5] = "MOD1";
uint8_t buf126[126] = { 0xD9, 0xAA, 0x8B, 0xB0, 0x7A, 0x1D, 0xA4, 0xFC, 0x1B
    ↪ , 0x90, 0xB1, 0x20, 0x2E, 0x14, 0x41, 0x66, 0xBE, 0xD9, 0xA5, 0x9C, 0
    ↪ xD1, 0x91, 0x31, 0x7D, 0x4C, 0x75, 0x9D, 0x88, 0xA6, 0xF6, 0x84, 0x7F,
    ↪  0xA0, 0x0F, 0x2F, 0x1A, 0x2C, 0xD3, 0x17, 0x48, 0x63, 0xC8, 0x68, 0
    ↪ x91, 0xDC, 0xA9, 0xF8, 0x9B, 0x82, 0x9D, 0x37, 0x53, 0x2E, 0x68, 0xD0,
    ↪  0x7A, 0xDD, 0x6E, 0x03, 0x83, 0x64, 0x87, 0x03, 0x04, 0x96, 0x32, 0
    ↪ x1F, 0xC3, 0x06, 0x36, 0x0B, 0x69, 0xFE, 0x73, 0xFB, 0xDB, 0x1D, 0xF3,
    ↪  0x76, 0x9F, 0x90, 0xAD, 0xF3, 0xBF, 0x15, 0xC3, 0x39, 0xF2, 0x31, 0
    ↪ x3C, 0x75, 0x95, 0xC3, 0x78, 0x9A, 0x5A, 0xAB, 0xB9, 0x1D, 0xB1, 0xEF,
    ↪  0x28, 0x1A, 0xED, 0x9B, 0x15, 0xC8, 0xB8, 0x08, 0x3E, 0x58, 0x99, 0
    ↪ xEB, 0x4B, 0x58, 0x01, 0x0E, 0x91, 0xF2, 0x40, 0xCE, 0x68, 0xD5, 0x91,
    ↪  0xE0, 0x6F};
//uint8_t buf126[126] = {1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 ,
    ↪ 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 ,
    ↪ 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 ,
    ↪ 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 ,
    ↪ 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67 , 68 ,
    ↪ 69 , 70 , 71 , 72 , 73 , 74 , 75 , 76 , 77 , 78 , 79 , 80 , 81 , 82 ,
    ↪ 83 , 84 , 85 , 86 , 87 , 88 , 89 , 90 , 91 , 92 , 93 , 94 , 95 , 96 ,
    ↪ 97 , 98 , 99 , 100 , 101 , 102 , 103 , 104 , 105 , 106 , 107 , 108 ,
    ↪ 109 , 110 , 111 , 112 , 113 , 114 , 115 , 116 , 117 , 118 , 119 , 120
    ↪ , 121 , 122 , 123 , 124 , 125 , 126 };
uint8_t buf126_2[126]= {0xEB, 0x8B, 0x28, 0x08, 0x3C, 0x17, 0x30, 0x57, 0x05
    ↪ , 0xCC, 0x6C, 0xCD, 0x84, 0x75, 0x0C, 0xDC, 0x0E, 0xF7, 0x27, 0x66, 0
    ↪ xF8, 0x36, 0xF7, 0xEA, 0x76, 0xC5, 0x52, 0x4B, 0x57, 0x33, 0xBB, 0x42,
    ↪  0xBE, 0xE3, 0x4B, 0xFB, 0xFB, 0x7B, 0x52, 0x01, 0x47, 0xBE, 0xCD, 0
    ↪ xCC, 0x33, 0xD9, 0xA8, 0x41, 0xD1, 0xD0, 0xA7, 0xC9, 0x06, 0x9F, 0xB3,
    ↪  0x7C, 0x64, 0x06, 0xC7, 0xBB, 0x39, 0x82, 0xFE, 0xF7, 0x66, 0x49, 0
    ↪ xF2, 0x61, 0xC4, 0x44, 0x61, 0x0C, 0x03, 0x2E, 0xD8, 0x36, 0x08, 0x80,
    ↪  0x78, 0xD9, 0x50, 0x1F, 0xA2, 0x56, 0xBE, 0x55, 0xD2, 0x23, 0x5B, 0
    ↪ x9A, 0xDE, 0x94, 0x1C, 0xDC, 0x8C, 0x82, 0x25, 0x7E, 0xE3, 0xEA, 0xC3,
    ↪  0x44, 0xF6, 0xC6, 0x73, 0xCE, 0xFC, 0x7B, 0x4E, 0x74, 0x54, 0x9F, 0
    ↪ x94, 0xF6, 0xF5, 0x52, 0x4B, 0xC8, 0x75, 0xA7, 0x62, 0x54, 0x3B, 0x7E,
    ↪  0x30, 0xC7};
uint8_t buf237[237] = {0xD9, 0xAA, 0x8B, 0xB0, 0x7A, 0x1D, 0xA4, 0xFC, 0x1B,
    ↪  0x90, 0xB1, 0x20, 0x2E, 0x14, 0x41, 0x66, 0xBE, 0xD9, 0xA5, 0x9C, 0
    ↪ xD1, 0x91, 0x31, 0x7D, 0x4C, 0x75, 0x9D, 0x88, 0xA6, 0xF6, 0x84, 0x7F,
    ↪  0xA0, 0x0F, 0x2F, 0x1A, 0x2C, 0xD3, 0x17, 0x48, 0x63, 0xC8, 0x68, 0
    ↪ x91, 0xDC, 0xA9, 0xF8, 0x9B, 0x82, 0x9D, 0x37, 0x53, 0x2E, 0x68, 0xD0,
    ↪  0x7A, 0xDD, 0x6E, 0x03, 0x83, 0x64, 0x87, 0x03, 0x04, 0x96, 0x32, 0
    ↪ x1F, 0xC3, 0x06, 0x36, 0x0B, 0x69, 0xFE, 0x73, 0xFB, 0xDB, 0x1D, 0xF3,
    ↪  0x76, 0x9F, 0x90, 0xAD, 0xF3, 0xBF, 0x15, 0xC3, 0x39, 0xF2, 0x31, 0
    ↪ x3C, 0x75, 0x95, 0xC3, 0x78, 0x9A, 0x5A, 0xAB, 0xB9, 0x1D, 0xB1, 0xEF,
    ↪  0x28, 0x1A, 0xED, 0x9B, 0x15, 0xC8, 0xB8, 0x08, 0x3E, 0x58, 0x99, 0
    ↪ xEB, 0x4B, 0x58, 0x01, 0x0E, 0x91, 0xF2, 0x40, 0xCE, 0x68, 0xEB, 0x8B,
    ↪  0x28, 0x08, 0x3C, 0x17, 0x30, 0x57, 0x05, 0xCC, 0x6C, 0xCD, 0x84, 0
    ↪ x75, 0x0C, 0xDC, 0x0E, 0xF7, 0x27, 0x66, 0xF8, 0x36, 0xF7, 0xEA, 0x76,
    ↪  0xC5, 0x52, 0x4B, 0x57, 0x33, 0xBB, 0x42, 0xBE, 0xE3, 0x4B, 0xFB, 0
    ↪ xFB, 0x7B, 0x52, 0x01, 0x47, 0xBE, 0xCD, 0xCC, 0x33, 0xD9, 0xA8, 0x41,
```

```
    ↪  0xD1, 0xD0, 0xA7, 0xC9, 0x06, 0x9F, 0xB3, 0x7C, 0x64, 0x06, 0xC7, 0
    ↪ xBB, 0x39, 0x82, 0xFE, 0xF7, 0x66, 0x49, 0xF2, 0x61, 0xC4, 0x44, 0x61,
    ↪  0x0C, 0x03, 0x2E, 0xD8, 0x36, 0x08, 0x80, 0x78, 0xD9, 0x50, 0x1F, 0
    ↪ xA2, 0x56, 0xBE, 0x55, 0xD2, 0x23, 0x5B, 0x9A, 0xDE, 0x94, 0x1C, 0xDC,
    ↪  0x8C, 0x82, 0x25, 0x7E, 0xE3, 0xEA, 0xC3, 0x44, 0xF6, 0xC6, 0x73, 0
    ↪ xCE, 0xFC, 0x7B, 0x4E, 0x74, 0x54, 0x9F, 0x94, 0xF6, 0xF5};
uint32_t mslip_len = 0;

fprintf(stderr,">>␣");
while(true){
    char c = getc(stdin);
    switch (c)
    {
        case 'r':
        fprintf(stderr, "Send␣RQCST␣command!\n");
        send_cmd(fd,CMD_RQCST, NULL,true);
        recv_reply(fd,REP_RQCST,rcvd_payload);
        break;
        case 'b':

        memset(bin_payload,0,SEQ_BYTES_LENGTH+1);
        memset(mslip_payload,0,TEST_SEQ_LENGTH+1);//plus one for the comma,
            ↪ it is part of the payload on uart but not on rf tho
        uint32_t i = 0;
        int j = 0;
        uint8_t packet_counter = 1;

        for(i = 1; i<(SEQ_BYTES_LENGTH+1);i++){
            bin_payload[i] = i%122+1;//0xf0
        }
        bin_payload[0] = ',';
        //print_payload(bin_payload,TEST_SEQ_LENGTH);
        mslip_len = binary_to_mslip(mslip_payload,bin_payload,
            ↪ SEQ_BYTES_LENGTH+1);
        fprintf(stderr, "\n0x%.2x,",bin_payload[i]);
        //print_payload(mslip_payload,TEST_SEQ_LENGTH*2);
        send_cmd(fd,CMD_SBUFF, mslip_payload,false);
        for(int i = SEQ_BYTES_LENGTH-10;i<SEQ_BYTES_LENGTH+1;i++){
            fprintf(stderr, "\n0x%.2x,",bin_payload[i]);
        }
        if (!recv_ack(fd))
        fprintf(stderr, "Invalid␣ACK␣response\n");
        else
        fprintf(stderr, "ACK\n");
        send_cmd(fd,CMD_RQCST, NULL,true);
        recv_reply(fd,REP_RQCST,rcvd_payload);

        break;
        case 't':
        char option = getc(stdin);
        switch(option){
            case '0':
```

```
    strcpy(sendb_mod,"MOD0");
    break;
    case '1':
    strcpy(sendb_mod,"MOD1");
    break;
    case '2':
    strcpy(sendb_mod,"MOD2");
    break;
    case '3':
    strcpy(sendb_mod,"MOD3");
    break;
    case '4':
    strcpy(sendb_mod,"MOD4");
    break;
    case '5':
    strcpy(sendb_mod,"MOD5");
    break;
    case '6':
    strcpy(sendb_mod,"MOD6");
    break;
    case '7':
    strcpy(sendb_mod,"MOD7");
    break;
    case '8':
    strcpy(sendb_mod,"MOD8");
    break;
    case '9':
    strcpy(sendb_mod,"MOD9");
    break;
    default:
    strcpy(sendb_mod,"MOD0");
    break;



}
uint8_t power_option = getc(stdin);
if(power_option>='0' && power_option<='9'){
    sendb_power = power_option;
} else {
    sendb_power = 9;
}
uint8_t sendb_payload[8]; //nullcharacter need to be at the end of
    ↪ payload (wont send it)
sendb_payload[0] = ',';
sendb_payload[1] = sendb_power;
sendb_payload[2] = ',';
memcpy(sendb_payload+3,&sendb_mod,4);
//strcpy(sendb_payload,",FFFF");
send_cmd(fd,CMD_SENDB, sendb_payload,true);

if (!recv_ack(fd))
```

```
            fprintf(stderr, "Invalid␣ACK␣response\n");
            else
            fprintf(stderr, "ACK\n");
            send_cmd(fd,CMD_RQCST, NULL,true);
            recv_reply(fd,REP_RQCST,rcvd_payload);

            break;
            case 'a':
            memcpy(bin_payload+1,buf237,237);
            bin_payload[0] = ',';
            for(int i = 0;i<127;i++){
                //fprintf(stderr," %.2X " ,bin_payload[i]);
            }
            uint32_t mslip_len = binary_to_mslip(mslip_payload,bin_payload,238);
            fprintf(stderr,"␣%d␣mslip␣length\r\n␣" ,mslip_len);
            for(int i = 0;i<mslip_len;i++){
                fprintf(stderr,"␣%.2X␣" ,mslip_payload[i]);
            }
            send_cmd(fd,CMD_SBUFF, mslip_payload,true);

            if (!recv_ack(fd))
            fprintf(stderr, "Invalid␣ACK␣response\n");
            else
            fprintf(stderr, "ACK\n");
            break;
            default:
            break;
        }
        if(c != '\n')
        fprintf(stderr,">>␣");
    }









    CloseHandle(fd);
    return 0;
}
```

5.9. Listing. obc_emulator.h

```
#ifndef OBC_EMULATOR_H
#define OBC_EMULATOR_H


#include <stdint.h>
#include <stdio.h>
#include <stdbool.h>
#include <stddef.h>
#include <string.h>
```

```
#include "uart.h"
#include "mrc_com.h"
#include "checksum.h"
#include "mslip.h"




#endif
```