University of Tartu

Faculty of Science and Technology

Institute of Technology

Kristo Allaje

Communications subsystem hardware and software development for the ESTCube-2 nanosatellite

Master thesis (30 ECTS)
Robotics and Computer Engineering

Supervisors:

Janis Dalbins, M.Sc.Eng.
Indrek Sünter, PhD.

Tartu 2021

Abstract/Resümee

Communications subsystem hardware and software development for the ESTCube-2 nanosatellite One of the most crucial components of satellites is their communications subsystem. Without a functioning radio link, it would be challenging to receive telemetry and payload data from the satellite and send telecommands to it from the ground. ESTCube-2 is a 3U CubeSat from the Estonian Student Satellite Foundation that is expected to launch in 2022. The mission of ESTCube-2 is to test various payloads in LEO. The primary payload being the plasma brake, similar to the Electric Solar Wind Sail (E-Sail) experiment on ESTCube-1. Due to the critical nature of the satellite communications system, it is essential to start with thorough testing early to reach high reliability by the launch. The goals for this master thesis are to test ESTCube-2 communications subsystem hardware and software, and to create an engineering model, to resolve any issues discovered.

CERCS: T120 Systems engineering, computer technology; T170 Electronics; T320 Space technology

Keywords: CubeSat, ESTCube-2, radio communication

ESTCube-2 nanosatelliidi raadioside alamsüsteemi riistvara ja tarkvara arendus Üks kõige olulisemaid satelliidi komponente on sidesüsteem. Ilma toimiva raadioühenduseta oleks keeruline satelliidilt telemeetria ja eksperimentide andmeid vastu võtta ning maapinnalt käske tagasi saata. ESTCube-2 on 3-ühikuline kuupsatelliit Eesti Tudengisatelliidi Sihtasutuselt, mis peaks orbiidile jõudma 2022. aastal, et läbi viia eksperimente madalal Maa orbiidil. ESTCube-2 peamine eksperiment on plasmapidur, mis sarnaneb ESTCube-1 pardal olnud elektrilise päikesepurjega. Satelliitsidesüsteem on süsteemikriitiline ning selle töökindluse tagamiseks tuleks võimalikult varakult alustada põhjaliku testimisega. Selle magistritöö eesmärk on testida ESTCube-2 kommunikatsiooni alamsüsteemi riist- ja tarkvara ning luua insenerimudel, et avastatud probleemid kõrvaldada.

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia; T170 Elektroonika; T320 Kosmosetehnoloogia

Märksõnad: kuupsatelliit, ESTCube-2, raadioside

Contents

Abstract/Resümee **2**

List of figures **5**

Acronyms **8**

1 Introduction **11**

2 Other CubeSat missions **12**
2.1 ESTCube-1 12
2.2 TTU100 and TTU101 14
2.3 Aalto-1 16
2.4 ASTERIA 17

3 ESTCube-2 communications subsystem prototypes **19**
3.1 Hardware 19
3.2 Revision 3 software 24

4 Thesis scope **26**

5 Revision 3 prototype functional testing **27**
5.1 MSP430 hardware abstraction layer 27
5.2 FreeRTOS 29
5.3 C/C++ compiler toolchain 30
5.4 Firmware architecture 30
5.5 Drivers and applications 32

6 Revision 4 prototype hardware verification **35**
6.1 Hardware overview 35
6.2 Power management 36
6.3 Primary communications system 41
6.4 Secondary communications system 43
6.5 UHF transmission and reception chain 45
6.6 External connectors 47

7 Design of the engineering model **49**
7.1 Power management 49
7.2 Primary communications system 50
7.3 Secondary communications system 51
7.4 UHF transmission and reception chain 53

7.5 External connectors 55

8 Conclusion 57

References 58

Appendix A Revision 4, issues discovered 63

Appendix B Engineering model, printed circuit board 65

Appendix C Tools used during the thesis 73

Non-exclusive license 75

List of figures

1 High-level overview of ESTCube-1 communications system and its connections between internal components. The subsystem connects to the satellite's power bus and two communication buses. 13
2 High-level overview of TTU-100/TTU-101 communications system and its connections between internal components [26]. . 15
3 A block diagram of digital, RF and power interfaces for the Aalto-1 satellite [28]. 16
4 A block diagram of ASTERIA's telecommunication system [33]. 17
5 Block diagram illustrating main hardware components of ESTCube-2 communications subsystem revision 3. 20
6 A partially assembled COM revision 3 circuit board. The PCOM microcontroller along with its peripherals and UHF RF chain has been assembled. KCOM section has been left unsoldered on this particular circuit board. 21
7 A block diagram of the main hardware components of the communications subsystems in revision 4. 22
8 Revision 4 after an extensive hardware review that discovered significant number of crucial design mistakes. The list of problems can be found in appendix A. 23
9 A block diagram of the primary communications system software architecture presented in Erik Amor's thesis [36]. 25
10 A diagram detailing the created layered software architecture. 31
11 Oscilloscope screenshot, showing the bitstream output by the transceiver in blue. The data sampling interrupt of the communications subsystem microcontroller is shown in red. 33
12 Power lines present on the revision 4 prototype. 37
13 A diagram showing the main internal components of the *RF power amplification* block from figure 12. 37
14 A diagram showing the subcomponents of the *IC 3V3* block from figure 12. 38
15 The top portion of the image shows one of the revision 4 current sense amplifiers that were tasked with measuring current consumption on different power lines. The bottom half shows the suggested wiring diagram by the manufacturer for the LT6105 current sense amplifier [53]. 40
16 A block diagram of the components of the primary communications system on the revision 4 prototype. 41
17 The TCVCXO switch present in the revision 4 prototype. . . . 43

18 A block diagram of the most important components making up the secondary communications system on revision 4. 44
19 A block diagram showing the main components of the ultra high frequency radio chain on the revision 4. 45
20 RF track impedance mismatches between passive components and unnecessarily long sections of microstrip in the revision 4 UHF chain. 47
21 An image of the ESTCube-2 bus along with the bus connectors that can be seen on the left. 48
22 An overview of the most important components in the engineering model and how they interact with each other. 49
23 A block diagram of the 3.3 V power distribution on the engineering model. 50
24 A block diagram of the components making up the primary communications system on the engineering model. 51
25 A block diagram of the components making up the secondary communications system on the engineering model. 52
26 A picture of the UHF transmission path on the revision 4 (left) and the engineering model (right) circuit board. 53
27 A picture of the SCOM UHF transmission path (outlined in red) on the revision 4 (left) and engineering model (right) circuit boards. 54
28 The remove before flight connector that will be used on ESTCube-2. 55
29 Engineering model serial connections for additional debugging capability. 56
30 Top side of the engineering model printed circuit board. This side houses all of the components that are related to transmission and reception on radio frequencies along with the secondary communications subsystem. 65
31 Bottom side of the engineering model printed circuit board. This side houses the primary communications subsystem along with the DC/DC switching regulator. 66
32 The top layer houses all the RF components, shielding gaskets and the secondary communications subsystem. 67
33 The second layer from the top was allocated for ground. It is meant to provide shielding to the components on the bottom layer of the printed circuit board and serves as a current return path for the integrated circuits. 68

34 The third layer from the top was allocated for distributing power to various integrated circuits. 69
35 The fourth layer from the top was used for digital signal routing between various components. 70
36 The fifth layer from the top, similar to the forth, was also used for digital signal routing between various integrated circuits. . 71
37 The sixth layer from the top, houses the primary communications subsystem along with components that might be susceptible RF interference, like the DC/DC switching regulator. 72

Acronyms

API Application Programming Interface. The API defines the way, e.g. function calls, that two software layers have to use for interacting with one another. 27, 28, 32

AX.25 Amateur X.25. A popular data link layer protocol used by radio amateurs in packet radio. It's a popular communication protocol on educational CubeSat missions that transmit on radio amateur bands. 14, 16, 24, 32

CDHS Command and Data Handling System. A system that handles commands sent to, and received by the satellite, including payloads. 12, 13, 18

COM Communications. A system that handles radio communication from and to the satellite. 13, 21, 22, 26, 37–39, 41, 50, 52, 57, 63

EPS Electrical Power System. The system which supplies the spacecraft bus and payloads with electrical power. 13, 15, 20, 35, 37, 41, 49

FPGA Field-Programmable Gate Array. An integrated circuit based on configurable logic blocks. 14

FRAM Ferroelectric Random Access Memory. Non-volatile random access memory that makes use of the electrical polarisation characteristic of ferroelectric materials. 21, 24, 29, 31, 34, 41–43, 63

GCC GNU Compiler Collection. GNU compiler collection is an optimising compiler produced by the GNU project. 30

HAL Hardware Abstraction Layer. The HAL provides the developers a convenient way of interacting with the underlying embedded system hardware while hiding the hardware specific complexities. 27, 28, 31, 32, 57

ICP Internal Communication Protocol. ICP is a communication protocol developed by the ESTCube team for subsystem to subsystem communications onboard the ESTCube-2 satellite. 21, 24, 32, 36, 41, 52, 56

KCOM Kill-Communications. A secondary subsystem that was housed on the revision 3 prototype along with the primary communications subsystem. It was responsible for very high-frequency band radio reception and disabling of the RF power amplifier. 20–22

LDO Low-DropOut voltage regulator. Low dropout voltage regulators are used to output a lower voltage from a higher voltage input by turning excess power into heat. 35, 37–39, 41, 49, 63

LEO Low Earth Orbit. LEO is defined as a distance of 80 to 2000 km from the Earth's surface. 11, 17

OQPSK Offset Quadrature Phase Shift Keying. OQPSK is a variant of phase shift keying where two bits are modulated at once using one of four different values of phase. 14, 15, 18

PCOM Primary Communications. The primary communications system on ESTCube-2 is responsible for the day to day radio frequency communication between the ground station and the satellite. 20–22, 24, 37, 39, 42, 43, 45, 52, 55, 56, 63

PLL Phase Locked Loop. PLL is a feedback system that is used to generate an output signal in phase with the input. 15

QSPI Quad Serial Peripheral Interface. QSPI is a modified version of SPI that has four data lines allowing a system to send 4 data bits per clock cycle. 41, 42, 63

RF Radio Frequencies. Radio frequencies are used for wireless communications using electromagnetic radio waves. 11, 15, 20, 36, 43, 46, 50, 52, 53

S band S band. IEEE designation for frequencies from 2 to 4 GHz. 17

SAW Surface Acoustic Wave filter. A surface acoustic wave filter converts electrical energy into acoustic or mechanical energy using a piezoelectric material. 44, 45, 63

SCOM Secondary Communications. A secondary subsystem housed on the same printed circuit board as the primary communications system on ESTCube-2. It mainly serves as a backup in case of primary communications failure. 22, 37, 42, 44, 45, 52, 53, 55, 56, 63

SDR Software Defined Radio. A radio system where the signal processing is done in software instead of using traditional analogue hardware components. 11, 18

SPI Serial Peripheral Interface. SPI is a synchronous serial communication protocol that enables communication over short distance in embedded devices. 15, 21, 31, 41–43, 50, 63

SRAM Static Random Access Memory. Volatile random access memory that uses flip-flops to store information. 15, 29

TCVCXO Temperature Compensated Voltage Controlled Crystal Oscillator. 35, 41, 42, 50, 53, 63

UART Universal asynchronous receiver/transmitter. A device that enables communication through an asynchronous serial communication link. 12, 21

UHF Ultra High Frequency. The UHF band spans from 300 MHz to 3 GHz. 11, 12, 14–17, 20, 22, 36, 41, 44–46, 52, 53

VHF Very High Frequency. The very high frequency band spans from 30 to 300 MHz. 11, 12, 20, 36, 44, 45, 52

X band X band. IEEE designation for frequencies from 8 to 12 GHz. 14, 15

1 Introduction

Over the years, one satellite standard, the CubeSat, has become more and more popular [1]. These are small satellites made of 10x10x10 cm units, with a single unit weighing up to 1.33 kg [2]. The rise in popularity has been driven by advances in miniaturisation, decreasing costs for deploying a spacecraft to low Earth orbit (LEO) and favourable conditions in LEO, inside the protection of the Earth's magnetic field [3, 4]. Due to this, utilising them for testing cutting edge technology is a popular choice as the costs associated with failure are smaller than with larger conventional satellites [5].

The most common way satellites communicate with Earth is using radio frequency (RF) in the range of 30 MHz - 30 GHz [6]. Radio transmitters in the very high frequency (VHF) and ultra high frequency (UHF) bands have been successfully demonstrated on multiple missions over the years and have the highest technology readiness level [7]. However, due to decreasing costs associated with sending spacecraft to orbit, an ever-increasing number of satellites share the same radio spectrum. The increasing number of users on the VHF and UHF bands is one reason why spacecraft developers are looking for alternative methods of communication. Many satellite developers implement communications on higher frequencies [8], use novel methods like optical [9, 10, 11] or software defined radio (SDR) based communications [12, 13, 14], build antennas with innovative designs [15, 16] or relay data inside a satellite swarm [17].

This master's thesis focuses on the communications subsystem of ESTCube-2, a 3 unit CubeSat from the Estonian Student Satellite Foundation. The thesis aims to test ESTCube-2 communications subsystem prototype hardware and software and to create an engineering model, resolving any discovered issues in the prototype. This thesis gives the reader an overview of how other educational missions have designed their communications systems in chapter 2, covers the work done by others on the ESTCube-2 communications in chapter 3, and describes the work done during this thesis in chapters 5, 6 and 7.

2 Other CubeSat missions

The following section provides a brief overview of other educational CubeSat missions, specifically focusing on their communications subsystem for teleoperation from Earth. At the end of the chapter, a CubeSat mission from NASA is described to provide insight into how CubeSats are being used professionally.

2.1 ESTCube-1

ESTCube-1, the first Estonian satellite, was launched from the Guiana Space Centre on the 7th of May 2013. It was a 1U CubeSat with a mission to test spacecraft spin-up with magnetic torquers, tether deployment, and E-Sail testing in the plasma brake configuration [18, 19]. Unfortunately, the objective of reeling out the E-sail tether was unsuccessful due to a fault in the tether reeling mechanism. However, the team managed to create a working satellite, spin the spacecraft to 841 deg s^{-1} for tether deployment, test experimental cold cathode electron emitters (semi-successful), acquire photos of Estonia and capture hundreds of photos of the Earth. Finally, the mission provided opportunities to work on space-related topics for the participating youths as it was an educational mission at its core. [19, 20]

Communications with ESTCube-1 was half-duplex, and took place on the VHF band (145.xxx MHz) for uplink at 1200 bits s^{-1}. The downlink was on the UHF band (437.505 MHz) at 9600 bits s^{-1}[21] and the satellite transmitted a continuous wave beacon on 437.250 MHz. The satellite used two monopole antennas, one for uplink and the other for downlink, because of their ease of manufacture and their omnidirectional radiation pattern. The main requirements of the ESTCube-1 communications system were: 1) receive telecommands from the ground station, 2) forward telecommands to command and data handling system (CDHS), 3) transmit telemetry data to the ground station, 3) turn off all transmitters on telecommand, 4) compile telemetry data about the communications subsystem and forward it to the CDHS.

The primary uplink was meant for receiving telecommands and new firmware updates from the mission ground station. Depending on the packet header, received packets were either handled on the communications subsystem or forwarded to the CDHS through a universal asynchronous receiver/transmitter (UART) connection. The communications subsystem compiled telemetry about itself and transmitted it to CDHS on request.

Finally, every spacecraft transmitter transmitting on amateur radio frequencies must be capable of being turned off quickly by a telecommand [22]. This requirement came from the International Amateur Radio Union as ESTCube-1 was transmitting on amateur frequencies. When requested, communications (COM) turned off its transmitter, disabled power to its RF amplifier, and forwarded the same telecommand through CDHS to the electrical power system (EPS) microcontroller to disable beacon transmission. If the CDHS microcontroller didn't respond, COM would forward the command directly to the EPS microcontroller.

All this was achieved with an MSP430F2418 microcontroller and two external ADF7021 transceivers. The transmitted continuous-wave beacon was created by EPS through a controllable oscillator. A diagram about the ESTCube-1 communications system was taken from the projects internal wiki and can be seen in figure 1.

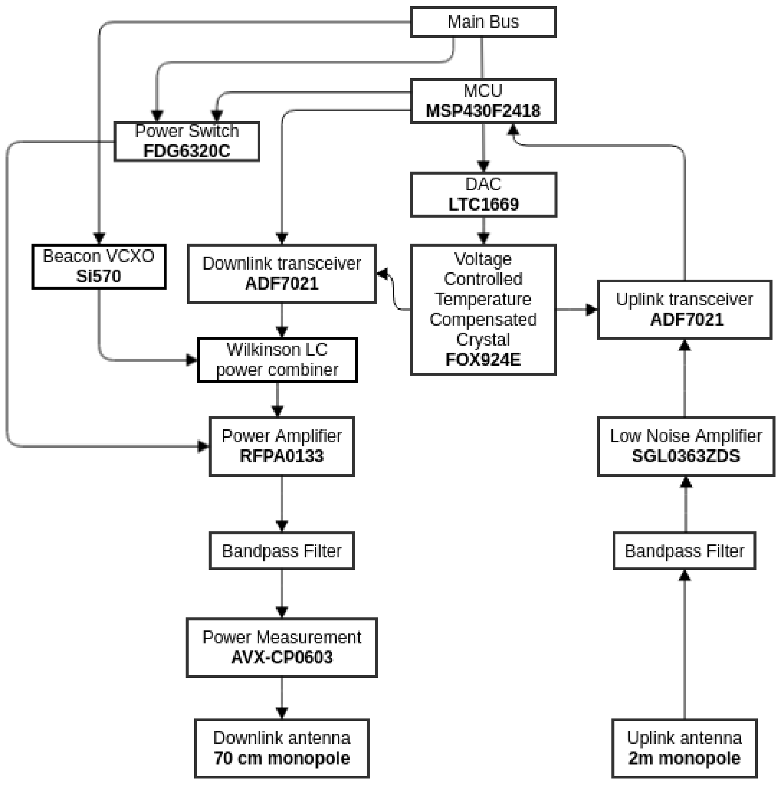


Figure 1: High-level overview of ESTCube-1 communications system and its connections between internal components. The subsystem connects to the satellite's power bus and two communication buses.

2.2 TTU100 and TTU101

The project to build a 1 unit CubeSat began in 2014 in Tallinna Tehnikaülikool. However, due to additional funding from AS Datel, it became a possibility to develop two identical 1 unit CubeSats. The first of them, called TTU101, commonly referred to as Koit, was launched on the 5th of July 2019 onboard Soyuz from the Vostochny Cosmodrome, the second called TTU100, also referred to as Hämarik, was launched on 3rd of September 2020 on board a Vega rocket from the Guiana Space Centre. The main objectives for both missions are Earth observation and technology demonstration of their cameras, image processing logic and their X band transmitter. In addition, there are experiments aboard to test field-programmable gate array (FPGA) fault tolerance to random bit changes and light-emitting diodes to test optical communication methods. As of the time of writing this thesis, both missions are still ongoing, but both satellites have issues regarding the reliability of their communications with Earth. [23]

Communications with both satellites is half-duplex and takes place on the 70 cm UHF band, with up- and downlink for both satellites at 435.450 MHz [23, 24, 25]. The communication with both spacecrafts takes place using amateur X.25 (AX.25) frames with a data rate of 9600 bits s^{-1}. Additionally, there is offset quadrature phase shift keying (OQPSK)) downlink capability on both satellites, 10.460 GHz for Koit and 10.465 GHz for Hämarik, that can provide, in theory, data rates up to 20 Mbits s^{-1}. The X band transmitter is meant for downlinking large amounts of data, specifically meant for downlinking images.

Both satellites have two UHF antennas, the primary turnstile antenna created from quarter-wave monopoles and a dipole antenna built into the wings. The X band antenna, which is attached to the bottom of the satellite, consists of four circularly polarised planar antennas. An overview of the most important components in the communications subsystem can be seen in figure 2.[24, 25]

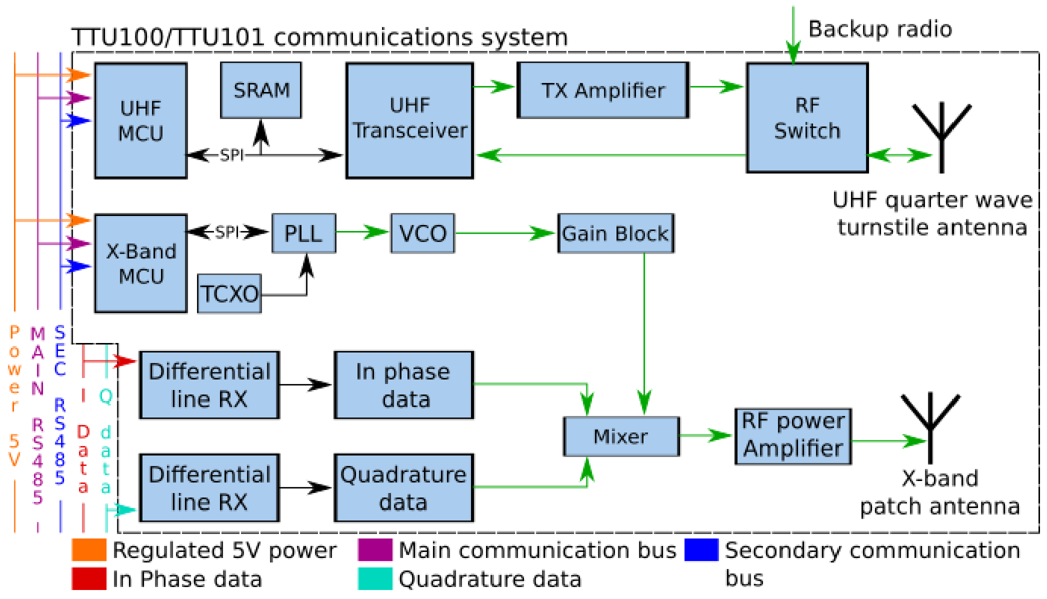


Figure 2: High-level overview of TTU-100/TTU-101 communications system and its connections between internal components [26].

The entire communications system is powered from the 5 V satellite bus, which the communications system converts to 3.3 V for its use. In addition, the bus houses two differential RS485 communication lines that enable communications between different subsystems. The UHF band transmissions are controlled by a PIC18F27J13 microcontroller connected to an external Si4468 transceiver and an external static random access memory (SRAM) memory that's used for packet buffering. Both devices are connected to the UHF microcontroller through an serial peripheral interface (SPI) connection. The transceiver, along with an external control signal from the backup radio, controls the internal state of the RF switch. The external signal enables the EPS to use the main antenna for transmitting the telemetry beacon. [26]

On the X band transmission line, mainly for transmitting images, there is another PIC microcontroller which controls an external phase locked loop (PLL) through an SPI connection. The PLL is meant for creating a carrier frequency for the data that will be sent through two DS90LT012A differential line receivers. The differential line receivers feed the data stream, divided into in-phase and quadrature data beforehand, into a mixer where the OQPSK modulation occurs. After the signal has been modulated, it gets amplified and sent out from the X band patch antenna located at the bottom of the satellite. [26]

2.3 Aalto-1

The development of the Aalto-1, a 3 unit CubeSat, started in 2010 from a student project in Aalto University, Finland. It was meant to be the first Finnish satellite, to be launched in 2015 with a Falcon 9 rocket. However, due to Falcon 9's launch anomalies in 2015 and 2016, the launch was postponed several times until the team found an alternative launch opportunity. Aalto-1 was finally launched on the 23rd of June 2017 onboard an Indian polar satellite launch vehicle. Aalto-1 has three primary payloads: 1) a spectral Earth observation imager, 2) a radiation monitoring device, and 3) an E-Sail module, which at the start of the program, was being developed for ESTCube-1. At the time of writing this thesis, the Aalto-1 mission is still ongoing. However, the E-Sail concept is still to be demonstrated in space. [27, 28, 29]

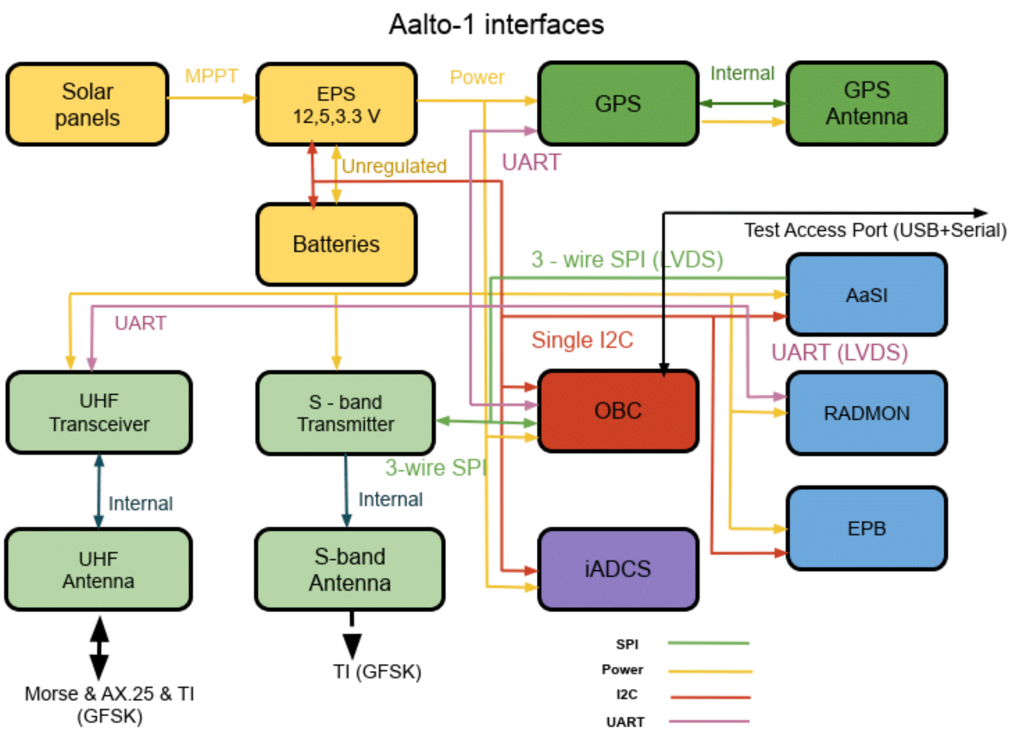


Figure 3: A block diagram of digital, RF and power interfaces for the Aalto-1 satellite [28].

Communication with Aalto-1 is half-duplex and takes place in the UHF band at 437.220 MHz with a data rate of 9600 bits s^{-1} through AX.25 frames. An automatic beacon is transmitted every two minutes on the same frequency, reporting the status of the satellite. The UHF communications system has two cold redundant Texas Instruments CC1125 transceivers and an MSP430FR5729 microcontroller. In addition, there are two dipole antennas, each connected to one of the two available UHF radios. The decision to switch radio systems is made by the on board computer or an external MSP430 based arbiter. An overview of the satellite can be found in the figure 3. [28]

Secondly, the satellite has a custom S band transmitter, which is used for high-speed downlink. It is based on a Texas Instruments CC2500 transceiver along with an MSP430FR5739 microcontroller. The secondary downlink works at 2.402 GHz and uses a custom made circular polarisation patch antenna. [30, 28]

2.4 ASTERIA

ASTERIA (Arcsecond Space Telescope Enabling Research in Astrophysics) was a 6U CubeSat developed by NASA's Jet Propulsion Laboratory. It was launched from the International Space Station to LEO on the 20th of November 2017. The only payload on the satellite was a wide field of view optical telescope with specialised hardware to perform high-precision pointing and thermal control. By February 2018, the ASTERIA spacecraft was able to meet its primary missions objectives, and after that, the mission was extended multiple times. The mission ended in February 2020 due to the loss of contact with the satellite. [31, 32]

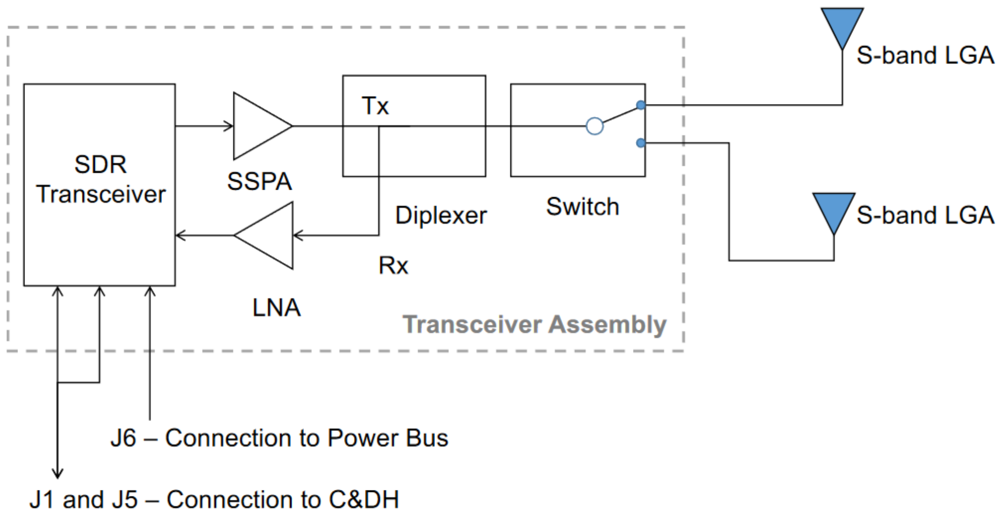


Figure 4: A block diagram of ASTERIA's telecommunication system [33].

The spacecraft featured a full-duplex S band communications with a single uplink and a single downlink channel, 2062.9 MHz and 2271.395 MHz, respectively. The uplink was mainly used for uploading commands and files needed for spacecraft operations. The downlink was used for sending back telemetry and payload data. The spacecraft transmitted using OQPSK with data rates of 10 Kbits s^{-1} in safe mode and 1 Mbits s^{-1} during nominal operations. Reception was implemented using binary phase-shift keying with data rates of 4 Kbits s^{-1} in safe mode and 32 Kbits s^{-1} under nominal conditions. The telecommunication subsystem consisted of a full-duplex SDR transceiver, a switch between the antennas, a diplexer, a low noise amplifier, a solid-state power amplifier and two low gain patch antennas. A block diagram that describes the communications system can be seen in figure 4. [31, 33]

The SDR (CSR-SDR-S/S) connected to the satellite's CDHS module through a single interface that was used for telemetry and data exchange. The entire subsystem was powered through a 28 V rail connected directly to the ASTERIA power system [31, 33]. The two patch antennas were placed on the opposite sides of the spacecraft to lessen possible communication issues during uncontrolled tumbling and maximise coverage. Only one antenna could be active, and the transceiver included an internal switch for antenna selection. The decision of which antenna to use was made by the flight software, which used the spacecraft's pointing system to determine the antenna with the biggest nadir vector. [33]

3 ESTCube-2 communications subsystem prototypes

Since 2015 the ESTCube-2 communications system has been developed by many volunteers. Many of them have volunteered for a semester or a summer, and a few have based their theses on related topics.

- ”UHF Communication System for Cubesatellite” by Jaanus Kalde, 2015 [34]
- ”Cubesat tipa satelīta UHF joslas zemas datu plūsmas komunikāciju apakšsistēmas prototipa izstāde un testēšana” by Janis Dalbins, 2016 [35]
- ”Design and implementation of prototype firmware for ESTCube-2 primary communication subsystem” by Erik Amor, 2018 [36]
- ”Nanosatelīta “ESTCube-2 komunikāciju apakšsistēmas laboratorijas modeļa izstrāde” by Klāvs Reinis Ozols, 2019 [37]

The following chapter is meant to give the reader an overview of the state of progress by the time the author started this master’s thesis.

3.1 Hardware

3.1.1 Revision 3 prototype

The communications subsystem developers had access to two 3rd revision hardware prototypes. The design was based on the work of Janis Dalbins, who defended his master’s thesis on communications subsystem hardware second revision in 2016 [35]. Shortly after the thesis defence, the 3rd revision was created based on the problems found during the testing of revision 2. An overview of the essential components of the communications subsystem revision 3 prototype can be seen in figure 5.

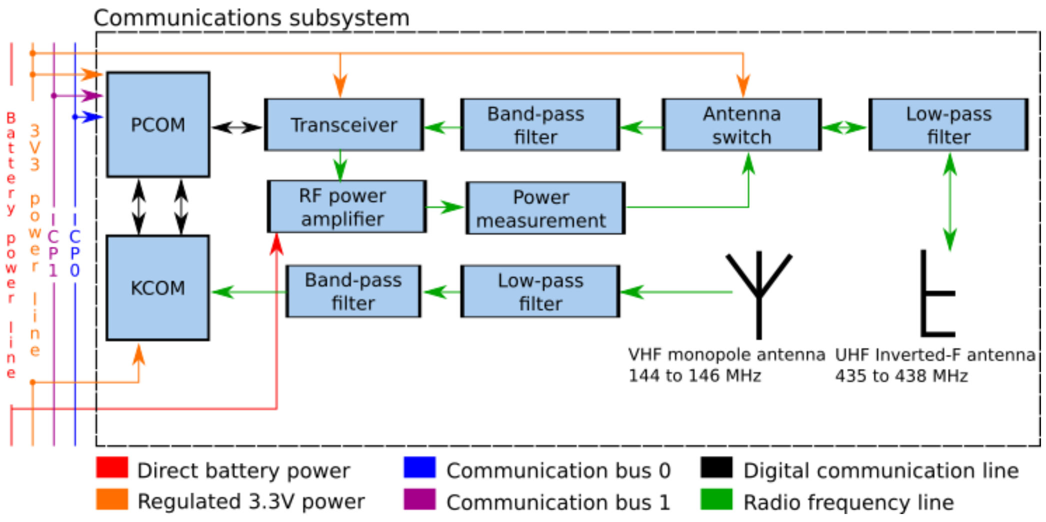


Figure 5: Block diagram illustrating main hardware components of ESTCube-2 communications subsystem revision 3.

ESTCube-2 communications subsystem revision 3 consisted of a primary communications (PCOM) and backup reception subsystem called kill-communications (KCOM) housed on the same printed circuit board. The primary communications system was connected to the satellite bus for direct battery voltage (6.8 to 8.4 V) for amplifying its RF signal. PCOM was also connected to two external communication buses for communicating with other subsystems. All the integrated circuits were designed to be powered from a regulated 3.3 V power rail from the EPS.

PCOM worked by receiving and transmitting on the UHF band and KCOM was meant to only receive on the VHF band. Both communications systems could receive telecommands from the ground station for setting different satellite operating modes. However, only PCOM could respond and fulfil requests to transmit data back to Earth. The primary purpose of KCOM was to disable the satellites transmission ability upon receiving an appropriate telecommand. This requirement comes from the international amateur radio union as ESTCube-2 is expected to transmit on amateur frequencies [22].

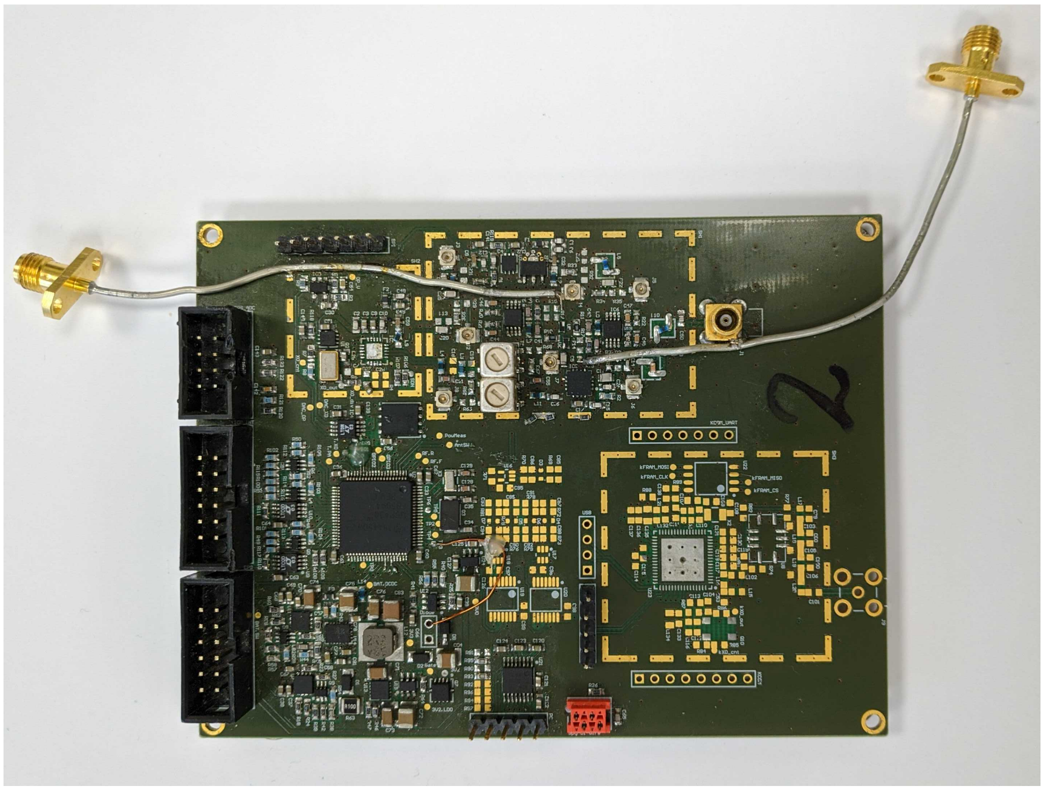


Figure 6: A partially assembled COM revision 3 circuit board. The PCOM microcontroller along with its peripherals and UHF RF chain has been assembled. KCOM section has been left unsoldered on this particular circuit board.

Both PCOM and KCOM had their separate microcontrollers along with external peripherals such as sensors and memories. PCOM consisted of a MSP430FR5994 microcontroller [38], an external Si4463 transceiver [39], an external SPI ferroelectric random access memory (FRAM) and an external SPI 10-bit digital to analog converter. The KCOM consisted of a EZR32WG330F256 microcontroller, that has a transceiver similar to the PCOM built into it, along with an external SPI FRAM [40]. Both COMs communicated with each other by exchanging internal communication protocol (ICP) packets through a UART connection between them. A photo of the revision 3 prototype can be seen in figure 6.

3.1.2 Revision 4 prototype

During the work on the author's master thesis, another COM prototype named revision 4 was created by Klāvs Reinis Ozols based on his bachelor's thesis, defended 2019 [37]. The design was based mainly on revision 3, but there are some noticeable differences between the two iterations. An overview of the communications subsystem revision 4 components can be seen in figure 7.

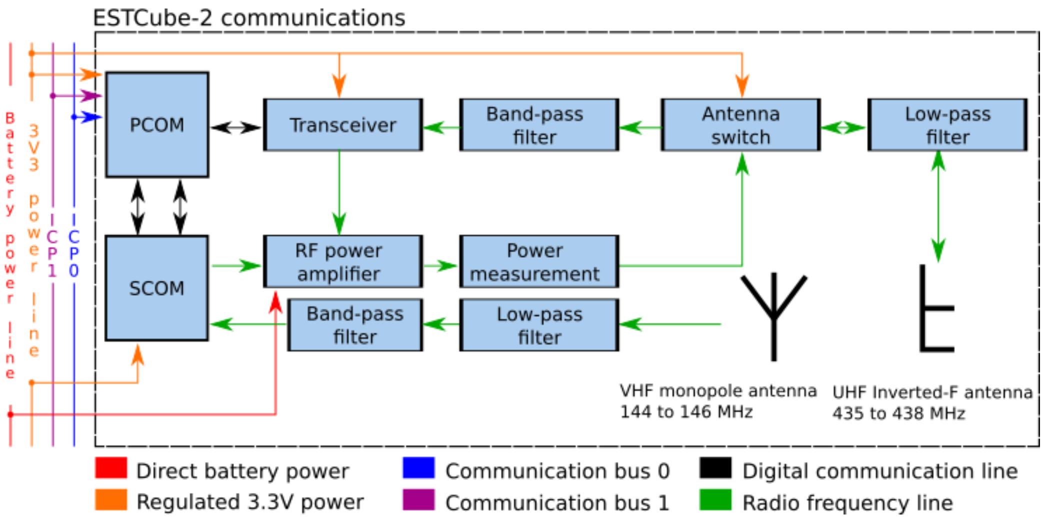


Figure 7: A block diagram of the main hardware components of the communications subsystems in revision 4.

First, the PCOM microcontroller was changed to an STM32L496ZG [41] because it was found that the MSP430 series fell short of the computational requirements, the problem is described further in chapter 5.5. Secondly, the KCOM received the ability to transmit on the UHF band. Due to its ability to transmit, the KCOM was renamed to secondary communications (SCOM). However the transmission chain was still solely controlled by the PCOM and SCOM needed permission in order to transmit on UHF frequencies. Third, the shape used for the printed circuit board was intended to be the same as for the flight model. The prototypes before revision 4 have had a custom rectangular shape to provide easy component placement, but during revision 4, the component placement was made with the final form factor in mind. The prototype can be seen in figure 8.

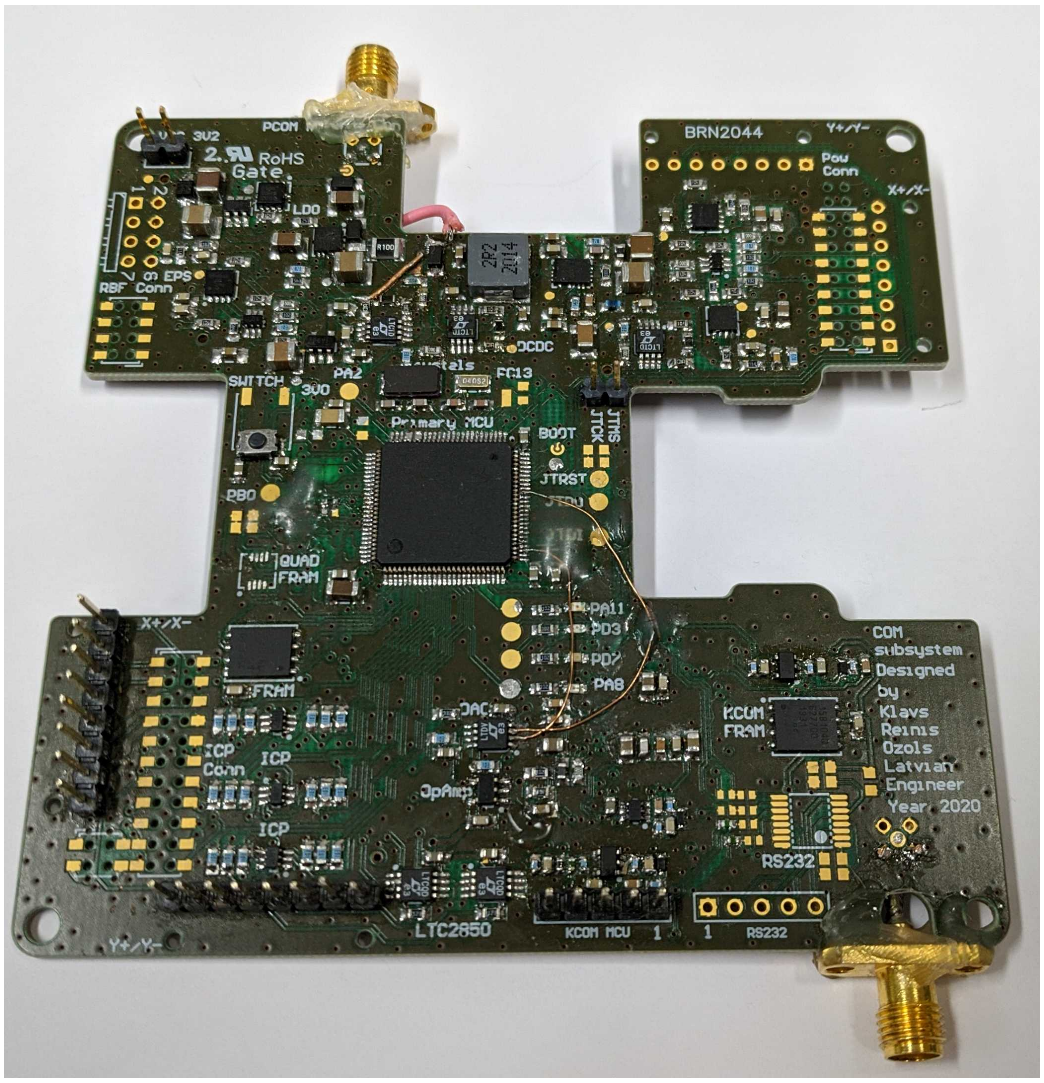
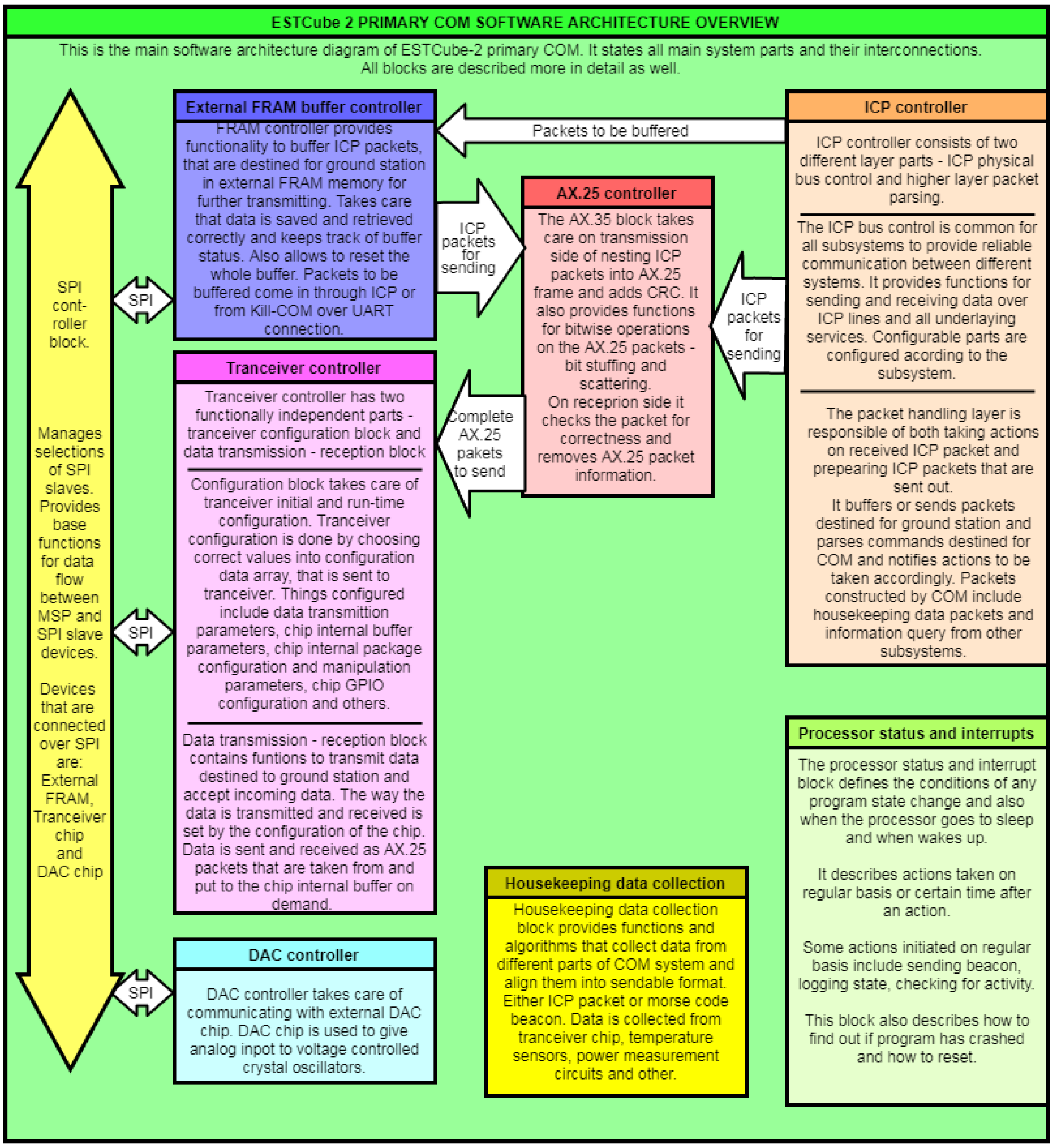


Figure 8: Revision 4 after an extensive hardware review that discovered significant number of crucial design mistakes. The list of problems can be found in appendix A.

3.2 Revision 3 software

The author's work on the ESTCube-2 communications system starts after the work of Erik Amor, who defended their bachelors' thesis on the topic of "Design and implementation of prototype firmware for ESTCube-2 primary communication subsystem"[36]. At the beginning of the thesis, PCOM had a complete system architecture description in text and diagram form, enough firmware for sending and receiving AX.25 packets, along with the capability to queue packets to be transmitted in an external FRAM buffer. There was enough functionality for basic ICP communications, and Amor had written a simple command-line interface to provide an easy way to test all the available modules. Overall the firmware was able to fulfil the goals of a communications system at a bare minimum level. The system architecture at the start of the thesis is shown in figure 9.


ESTCube 2 PRIMARY COM SOFTWARE ARCHITECTURE OVERVIEW
External FRAM buffer controller
Packets to be buffered
ICP controller
ICP packets for sending
AX.25 controller
ICP packets for sending
Tranceiver controller
Complete AX.25 pakets to send
SPI
SPI
SPI
DAC controller
Housekeeping data collection
Processor status and interrupts

This is the main software architecture diagram of ESTCube-2 primary COM. It states all main system parts and their interconnections. All blocks are described more in detail as well.

SPI cont-roller block.

Manages selections of SPI slaves. Provides base functions for data flow between MSP and SPI slave devices.

Devices that are connected over SPI are: External FRAM, Tranceiver chip and DAC chip

FRAM controller provides functionality to buffer ICP packets, that are destined for ground station in external FRAM memory for further transmitting. Takes care that data is saved and retrieved correctly and keeps track of buffer status. Also allows to reset the whole buffer. Packets to be buffered come in through ICP or from Kill-COM over UART connection.

Tranceiver controller has two functionally independent parts - tranceiver configuration block and data transmission - reception block

Configuration block takes care of tranceiver initial and run-time configuration. Tranceiver configuration is done by choosing correct values into configuration data array, that is sent to tranceiver. Things configured include data transmittion parameters, chip internal buffer parameters, chip internal package configuration and manipulation parameters, chip GPIO configuration and others.

Data transmission - reception block contains funtions to transmit data destined to ground station and accept incoming data. The way the data is transmitted and received is set by the configuration of the chip. Data is sent and received as AX.25 packets that are taken from and put to the chip internal buffer on demand.

DAC controller takes care of communicating with external DAC chip. DAC chip is used to give analog inpot to voltage controlled crystal oscillators.

The AX.35 block takes care on transmission side of nesting ICP packets into AX.25 frame and adds CRC. It also provides functions for bitwise operations on the AX.25 packets - bit stuffing and scattering. On receprion side it checks the packet for correctness and removes AX.25 packet information.

Housekeeping data collection block provides functions and algorithms that collect data from different parts of COM system and align them into sendable format. Either ICP packet or morse code beacon. Data is collected from tranceiver chip, temperature sensors, power measurement circuits and other.

ICP controller consists of two different layer parts - ICP physical bus control and higher layer packet parsing.

The ICP bus control is common for all subsystems to provide reliable communication between different systems. It provides functions for sending and receiving data over ICP lines and all underlaying services. Configurable parts are configured acording to the subsystem.

The packet handling layer is responsible of both taking actions on received ICP packet and prepearing ICP packets that are sent out. It buffers or sends packets destined for ground station and parses commands destined for COM and notifies actions to be taken accordingly. Packets constructed by COM include housekeeping data packets and information query from other subsystems.

The processor status and interrupt block defines the conditions of any program state change and also when the processor goes to sleep and when wakes up.

It describes actions taken on regular basis or certain time after an action.

Some actions initiated on regular basis include sending beacon, logging state, checking for activity.

This block also describes how to find out if program has crashed and how to reset.

Figure 9: A block diagram of the primary communications system software architecture presented in Erik Amor's thesis [36].

4 Thesis scope

The communications subsystem is a crucial part of every spacecraft. Thus the system must be thoroughly tested to discover design issues or miscalculations the designers might have made during the development process. Ideally, both the software and hardware would be fault-tolerant, well documented and the system extensively tested in scenarios as close to the real world as possible. Unfortunately, this was still not the case with neither the communications subsystem software nor hardware. The main goal for this master's thesis get the communications subsystem as close to flight ready as possible by doing the following:

- Functional testing of COM revision 3.
- COM revision 4 hardware testing and software development.
- Create an engineering model that resolves the problems found in revision 4.

5 Revision 3 prototype functional testing

The author's goal was to improve Amor's work and finalise the software design for the primary communications subsystem. However, shortly after taking over the software development, the author discovered many issues in the implementation of the software.

5.1 MSP430 hardware abstraction layer

The communications subsystem revision 3, along with several other ESTCube-2 subsystems were using an MSP430 series microcontroller. However, the custom hardware abstraction layer (HAL) for ESTCube MSP430 architecture was exceedingly outdated compared to the developed STM32 version. Due to a lack of developers, the MSP430 HAL had stagnated. At the same time, the custom HAL developed by ESTCube for STM32 microcontrollers was actively being maintained and in use on the on board computer and star tracker subsystems. This created a situation where if MSP430 and STM32 based subsystems were to use an identical sensor and a sensor driver had been written for one of the HALs, then the same code could not be reused for the other microprocessor architecture. This problem caused unnecessary overhead in firmware development, and issues resolved in one version of the sensor driver did not always propagate to the other architecture, causing further problems.

As STM32 HAL, along with accompanying drivers, was more mature and tested, the author decided to rework the MSP430 HAL to mimic the style of the STM32 HAL along with the application programming interface (API) calls it provided to the driver layer. The ESTCube team's requirements for the STM32 API, the requirements the new MSP430 API needed to conform to, are listed below.

1. HAL will not depend on higher Open Systems Interconnection levels of the software, for example, OS [42].
2. HAL functionality will not depend on the circuit board layout.
3. HAL will contain as little internal logic as possible; it only bridges hardware capabilities.
   (a) It may contain simple logic for parameter validation.
4. HAL has to abstract memory, peripheral, direct memory access addresses.

   (a) HAL only exposes pins and ports.
   (b) Pin mapping to specific devices is done on higher levels.

5. Support all hardware functionality that any higher-level needs.
   (a) Higher-level software shall not bypass HAL in any way. If new features are needed, then new functionality shall be added to the HAL.

6. HAL shall have a well-defined API convention.
   (a) All function names begin with `hal_` followed by `<header name>_` followed by `<descriptive name>`.
   (b) Doxygen shall be used to generate documentation from source files.

7. All initialisation functions must take at least a pointer to a configuration structure as an argument. The pointer can be a void pointer.
   (a) This enables better extendability in the future while maintaining most backwards compatibility

8. All configuration structures must include API version as the first argument

At the end of the MSP430 HAL rework, the new HAL had been successfully tested on two different types of MSP430 microcontrollers. MSP430RF5994 [38] for the communications subsystem and MSP430FR5969 [43] for the electrical power subsystem. The MSP430 HAL modules had been updated to meet the described standards above, mostly by rewriting them from scratch. Besides the analogue to digital converter and the watchdog module, the author wrote all modules listed below. The last two were created by a summer intern who was working under the author's supervision.

1. Digital Input/Output
2. Enhanced Universal Serial Communication Interface
   (a) UART mode
   (b) SPI mode
3. Timer module
4. 16-bit cyclic redundancy check module
5. Analog to digital converter
6. Watchdog timer module

5.2 FreeRTOS

The firmware was not using a real-time operating system, making task synchronisation difficult. The code execution was controlled by a `while` loop in the `main` function that went through different checks to see what functions should be called. Data between different functions and interrupt service requests were shared using global variables, making the code more difficult to understand and increased the risk of concurrency issues. Additionally, there was no preemption functionality in the code that would allow a higher priority task to suspend a lower priority task. For complex firmware, real-timeliness requirements are more challenging to meet without a real-time operating system of some sort.

Due to the disadvantages mentioned above of bare metal embedded systems, the author decided to use a real-time operating system in the revision 3 software. To maintain consistency and code reusability between various ESTCube-2 subsystems, FreeRTOS was chosen as other software developers actively used it to develop the on board computer system.

FreeRTOS is a real-time operating system for microcontrollers and small microprocessors developed by Real Time Engineers Ltd [44]. It is free to use and is distributed under the MIT open source license. Additionally, it has low requirements for microcontroller ROM and RAM and many officially supported, and community provided ports, including some for the Texas Instruments MSP430 and MSP430X based microcontrollers. Using FreeRTOS ESTCube software developers for the MSP430 architecture would have access to professionally developed and tested tools like preemptive or co-operative task scheduling, queues, semaphores, mutexes, stack overflow checking, trace recording and run-time collection of statistics on tasks. [44, 45, 46]

Unfortunately, FreeRTOS did not provide a port for the MSP430RF5994 MCU that works with the open-source MSPGCC toolchain. There was an MSPGCC toolchain port for the MSP430F449 that the author tried to adapt [47]. However, the original port proved to be unstable, unexpectedly causing memory issues in the entire system.

Finally, the problem was tracked down to invalid assembly commands in the provided `port.c` file, specifically the `portSAVE_CONTEXT()` and `portRESTORE_CONTEXT()` functions. As MSP430F449 only has 60 KiB flash, 2 KiB SRAM [48] compared to MSP430FR5994 [38] that has 256 KiB FRAM, 8 KiB SRAM, it doesn't need to make use of MSP430X Extended Instruction set. The extended MSP430X instructions give the MSP430X CPU full access to its 20-bit address space, allowing it to address more than 64 KiB of memory. The MSP430F449 port used a different instruction set in stashing and popping MCU registers, creating corrupted task control blocks within the operating system and causing undefined behaviour. After modifying the functions mentioned earlier with the MSP430X Extended Instruction set, the FreeRTOS port became stable, enabling further firmware development. [38, 48, 49]

5.3 C/C++ compiler toolchain

The primary communications subsystem firmware was compiled using the open-source MSP430 GNU compiler collection (GCC) [50]. However, the toolchain originally chosen for MSP430 architecture was far from stable. Over time, as the author added more and more code, the toolchain stability issues became even more apparent as software development progressed. The toolchain was finally deemed unusable after adding the FreeRTOS operating system into the firmware. The code was slow to compile, taking minutes, even for minor changes. Additionally, the accompanying debugger could not step in or out of large FreeRTOS code files, often crashing the entire computers operating system.

This made the author search for alternatives and reconsider using the open-source MSP430 GCC package. There was no documentation as to why previous members chose this toolchain, nor could any one of the senior members remember. It seems likely that team initially chose the open-source option because at the start of the ESTCube-2 project, the recommended development environment for MSP430 microcontrollers, Code Composer Studio, allowed for a maximum compiled binary size of 16 KiB when using the proprietary Texas Instruments MSP430 C/C++ compiler [51].

By the beginning of the thesis, the compiled binary size restriction seemed to have been lifted. The author could not find any information about it from Texas Instruments, only old forum posts [51]. Finally, after continued investigation and testing, a decision was made to switch the entire MSP430 toolchain for all subsystems to the Texas Instruments proprietary C/C++ compiler. The new compiler greatly improved the overall development stability. Porting the existing code and operating system was made easy by the MSP430X FreeRTOS port guide intended for the TI C/C++ Compiler [52].

5.4 Firmware architecture

While testing the revision 3 communications subsystem prototype, the author's secondary goal was to separate the subsystem firmware into logical layers, as seen in figure 10. Each layer can interface with the one directly below it by calling specific exposed functions. This approach would deter from creating a tightly coupled monolithic piece of code that's hard to maintain and debug. Another advantage of this style is that if a subsystem changes its microcontroller, layers above HAL would not be affected. Only the HAL code would have to be rewritten as the registers, and their locations, for working with various peripherals vary from a microcontroller to a microcontroller along with their initialisation logic.

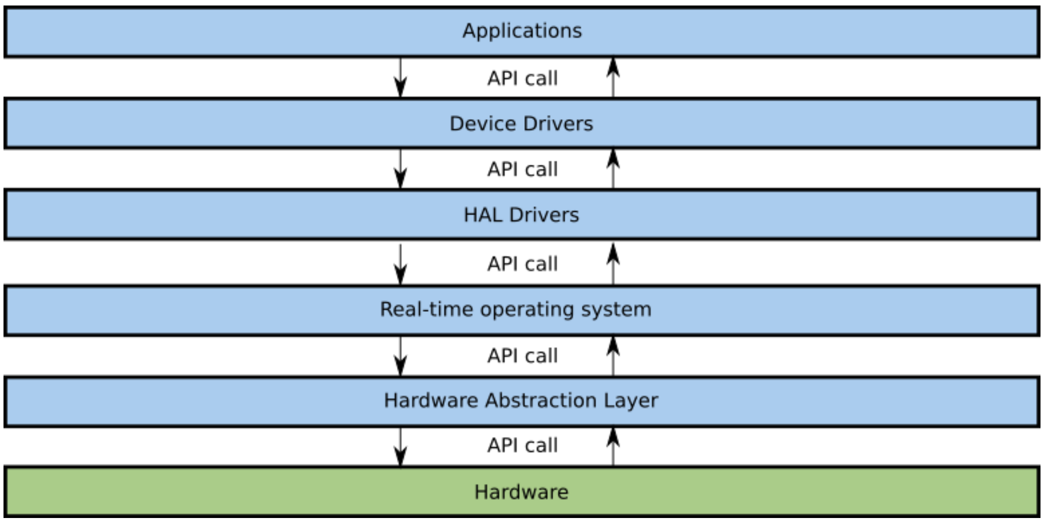


Figure 10: A diagram detailing the created layered software architecture.

Application layer - These would be the programs that do something useful on the satellite. It may be communications with Earth, satellite attitude estimations, or a task periodically waking up and collecting statistics about the spacecraft health.

Device driver layer - This layer houses logic for controlling specific external devices. When something from the application layer needs to get data from external memory, then this layer knows that the command `0x03` with 3 bytes of 19-bit memory address needs to be sent to the FRAM [40]. All this needs to happen through an SPI connection using the SPI peripheral number 2.

HAL driver layer - A microcontroller has a limited number of peripherals, and many processes might want to use them. A HAL driver is an arbiter between all the tasks, making sure that all requests to use the peripheral are fulfilled in order. Nothing from the layer above can take control of a communication line while another process is using it.

Real-time operating system - An operating system is used for maximising the efficiency of an embedded system. It provides the layers above with task synchronisation and multitasking capabilities while ensuring real-timeliness of the entire system.

Hardware abstraction layer - The hardware abstraction layer hides the hardware-specific details from the higher levels. Software developers working on higher layers do not always need to know what happens when they wish to initialise a microcontroller peripheral. They need it done to start using it for their work. If HALs meant for different microcontrollers have a compatible API, then the layers above can be used interchangeably with different microcontrollers.

Hardware - This represents the physical hardware that all the binary instructions are running on.

5.5 Drivers and applications

After restructuring the software into different layers, making the API calls of layers consistent between the MSP430 and STM32 HAL and drivers, implementing an operating system in the communications subsystem, the benefits of modular design and API consistency between subsystem were revealed. The author could reuse existing device drivers and application code such as ICP communication, command scheduling that he had created for other subsystems. Reuse greatly hastened the development of functional tests that were meant to test multiple modules at a time. As the communications subsystem's primary purpose is to enable the reception and transmission of packets from and to the ground station, this was the first functional test to be written.

For that purpose, a device driver for controlling the external Si4463 transceiver [39] was created along with application software that would control the transmission and reception of packets. As AX.25 frames are not supported by the transceiver, it was operated in direct mode where the received data and reception clock were output in real-time to the transceiver general-purpose input-output pins for the microcontroller to sample [39]. However, unforeseen problems started to appear while receiving AX.25 packets. In particular, there were issues with the decoding of longer packets, whereas packets with almost no payload were received successfully.

A test was conducted where the author sent a contiguous stream of 0's and 1's (`010101010101`) using a programmable synthesiser. The synthesiser was set to 437.505 MHz with a modulating frequency of 4.8 kHz and a deviation of 3 kHz, which equated to frequency shift keying, with a rate of 9600 bits s^{-1}. The output signal from the transceiver was verified using an oscilloscope which showed that the transceiver demodulated all the data correctly.

The application code for receiving data had been reduced to the bare minimum. The microcontroller was programmed to store long sequences of bitstream that originated from the external transceiver, which in turn had demodulated the data from the synthesiser. After analysing the stored data, the author discovered irregularities. Occasionally, the bitstream was 0**11**01 or 0101**00**, hinting at an incorrect sampling of the transceiver output. To validate data sampling interrupts, pin toggles on the rising edge were used. The data sampling interrupts were found to have irregularities and they are shown in figure 11.

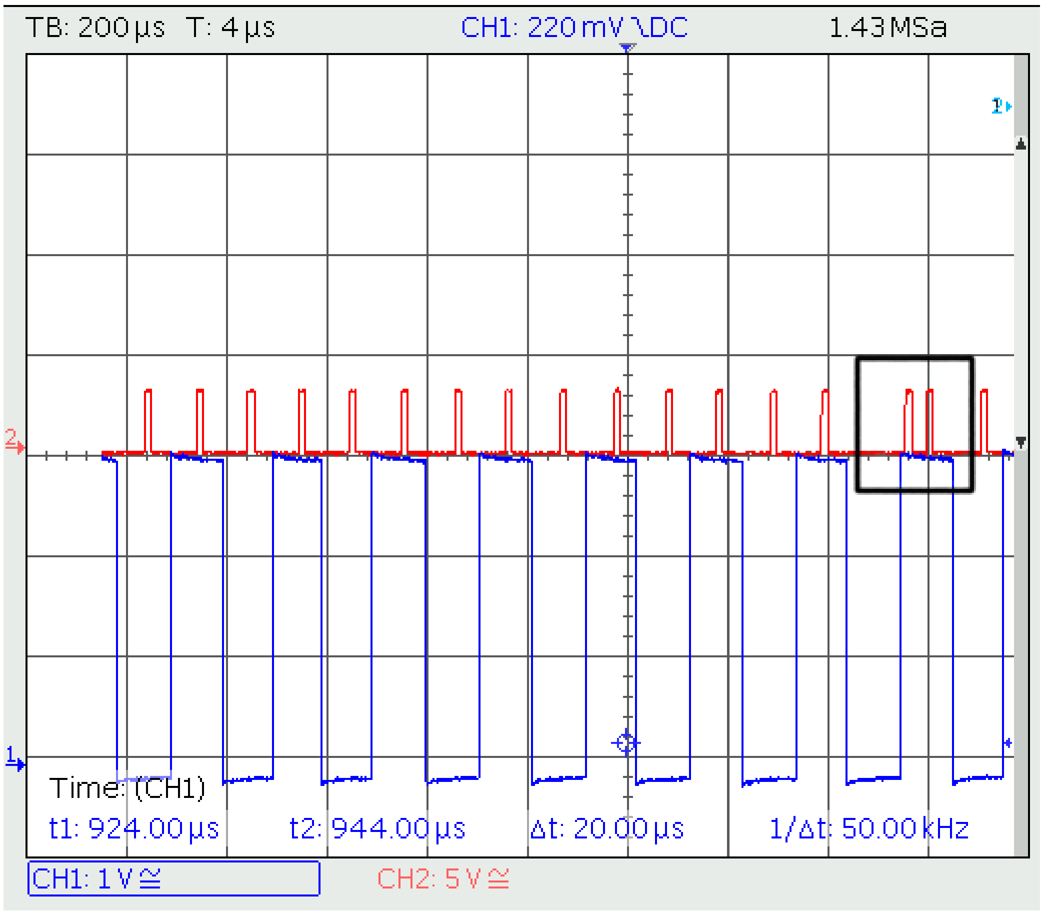


Figure 11: Oscilloscope screenshot, showing the bitstream output by the transceiver in blue. The data sampling interrupt of the communications subsystem microcontroller is shown in red.

The problem was tracked down to the FreeRTOS tick interrupt that had a higher priority than the digital pin interrupt used to sample demodulated data sent by the transceiver [38]. The tick interrupts preempted the data sampling interrupt causing it to be executed after the tick interrupt had finished. While the postponed sampling interrupt was being serviced, another was triggered by the change in the transceiver clock output. All this caused a periodical loss of one bit and the double sampling of the next bit.

It was attempted to raise the microcontroller clock frequency or to reduce the frequency of the tick interrupts. The issues could not be solved by increasing the clock speed because the internal FRAM could only be accessed at a maximum frequency of 8 MHz without introducing wait states [49], Thus nullifying the effect of having a higher clock speed in the first place. By reducing the frequency of tick interrupts, the reception of corrupted packets would have become less frequent, but the issue would have remained.

Lowering the priority of the tick interrupt and making sure that the sampling interrupt takes place on the highest priority digital pins would have just postponed the issue as there were many peripherals with an interrupt priority higher than the highest priority digital pin interrupt. Additionally, all the interrupt priorities had been predefined by the manufacturer, making the creation of a custom interrupt priority order impossible. [49, 38]

6 Revision 4 prototype hardware verification

After discovering that the MSP430 series fell short of the required computational capabilities for the communications subsystem to work properly, the ESTCube team looked for alternatives. A decision was made to abandon the MSP430 microcontrollers in favour of the STM32L4 ultra-low-power series. STM32L4 series offers better performance, more peripherals and memory at a power consumption comparable to the MSP430 microcontrollers [38, 41]. This would mean that ESTCube project software developers have less variety of microcontroller architectures to write code for. Also, they could do it with familiar tools that were already in use.

6.1 Hardware overview

As mentioned previously, the communications subsystem revision 4 was created by Klāvs Reinis Ozols based on his bachelor's thesis, which was defended in 2019 [37]. Due to various reasons, he was unable to test the revision 4 prototype hardware himself, so the author of this thesis performed the hardware verification tests.

During the hardware testing, the entire system was divided into five main blocks divided into sub-blocks. The revision 4 design was tested one functional block at a time, tools used can be seen under appendix C, and documentation about the performed tests, along with results, was written in the ESTCube project's internal wiki, Confluence. The tested functional blocks are as follows:

- Power management
  - Back-up low dropout regulator (LDO) power
  - EPS power
  - DC/DC power
  - Power Amplification rail
  - 3.0 V temperature compensated voltage controlled crystal oscillator (TCVCXO) reference
- Primary communications system
  - Primary COM microcontroller
  - Primary COM micocontroller peripherals
  - External transceiver

  - ICP communication
- Secondary communications system
  - Secondary COM microcontroller
  - Secondary COM microcontroller peripherals
  - VHF reception chain
- UHF transmission and reception chain
- External connectors
  - Bus connectors
  - Side panel connectors

The testing procedure for each functional block was similar. It consisted of 1) soldering the components for the functional block, 2) writing down the order that the components in a functional block should be soldered, along with any issues encountered, 3) developing test procedures for components in the functional block, 4) creating unit tests for the components in the functional block, 5) revising the documentation that was created during the testing of a block.

6.2 Power management

Revision 4 makes use of 3 power lines with different voltages: 1) direct battery voltage for RF power amplification, 2) 3.3 V rail for powering various integrated circuits, and 3) an exact 3.0 V reference for oscillators. Figure 12 illustrates the hierarchy of the power lines.

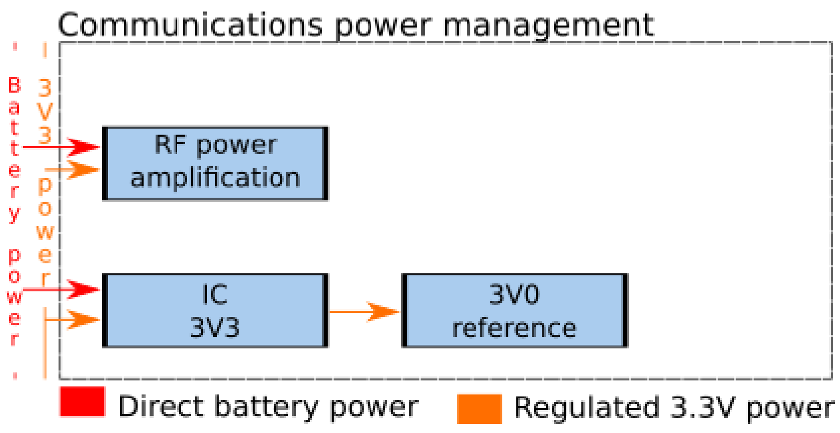


Figure 12: Power lines present on the revision 4 prototype.

The communications subsystem uses battery power when transmitting by directly connecting to the battery power rail and forwarding it to the power amplifier on the circuit board. There is a solid-state switch between the PWR_BAT rail and the power amplifier with current measuring circuitry to measure the power consumption when the satellite is transmitting. The power amplification circuitry can be toggled by PCOM, SCOM and EPS. The additional 3.3 V rail from the EPS is required to power the integrated circuitry controlling the amplification power rail. A simplified overview of the circuitry can be seen in figure 13.

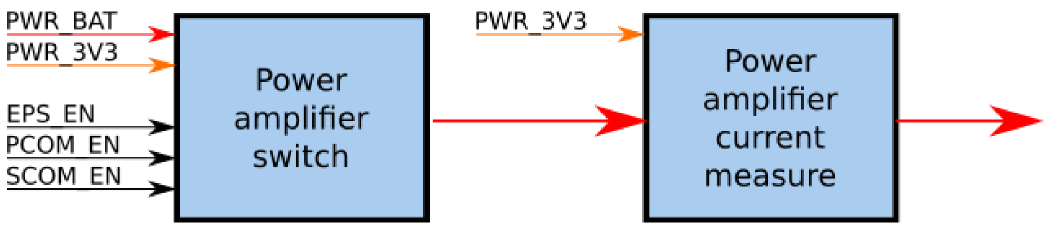


Figure 13: A diagram showing the main internal components of the *RF power amplification* block from figure 12.

The COM_3V3 power rail is used to supply all integrated circuits in the communications subsystem. Under nominal operations, the subsystem is powered by the 3.3 V power rail provided by the EPS along with an LDO that's always enabled. The LDO is meant to serve as a backup, in case of EPS 3.3 V rail failure, and converts the battery power (6.8 to 8.4 V) into 3.3 V usable for COM. The only way to power cycle COM is through EPS when it turns off both the battery and 3V3 rail allocated to the COM subsystem. Additionally, a DC/DC converter is used to provide enough current for the

RF peripherals while transmitting and is typically disabled while the COM subsystem is in reception mode. However, if the need arises, it can be left enabled to provide 3.3 V power to the subsystem. An overview of the 3.3 V power distribution circuitry can be seen in figure 14.

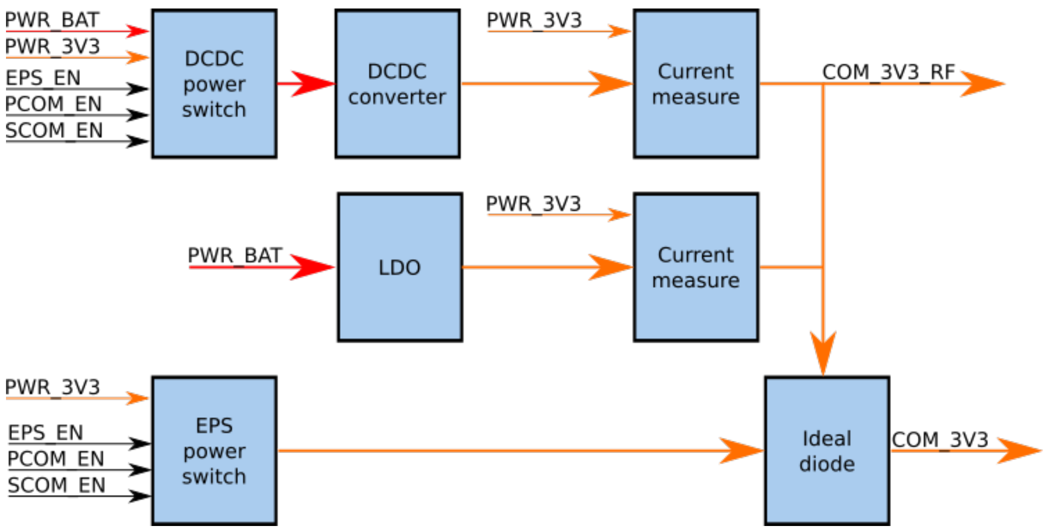


Figure 14: A diagram showing the subcomponents of the *IC 3V3* block from figure 12.

Finally, there is an additional LDO that is meant to provide a precise reference voltage to the onboard digital to analogue converter and voltage-controlled oscillators. The voltage reference is powered from the `COM_3V3` rail, as was shown in figure 12.

6.2.1 Issues discovered

During the power management functional block testing, the author discovered many issues with component placement and routing, mistakes in the schematic design, and fundamental problems stemming from design decisions. The list below is not comprehensive but highlights the most severe issues discovered during the testing of the power management functional block.

- All current sense amplifiers were incorrectly configured.
- LDO current sense output power rail was routed incorrectly.
- `COM_3V3_RF` was left disconnected from the PCOM transceiver.
- There was no way to power cycle the COM subsystem.

The LT6105 current sense amplifiers have an example of a typical configuration shown in their datasheet [53], shown in figure 15. However, the revision 4 prototype did not adhere to the design recommendations in the datasheet, causing current sense amplifiers on all power rails to malfunction. Secondly, the current sense integrated circuits tasked with measuring the current consumption on the LDO power rail had the current sense resistor output rail and the voltage output pin for the PCOM microcontroller swapped with each other on a top layer schematic. This connected the LDO power rail with the analogue to digital converter input of the PCOM microcontroller and connected the VOUT pin of the current sense amplifier to the 3.3 V power rail of the COM subsystem.

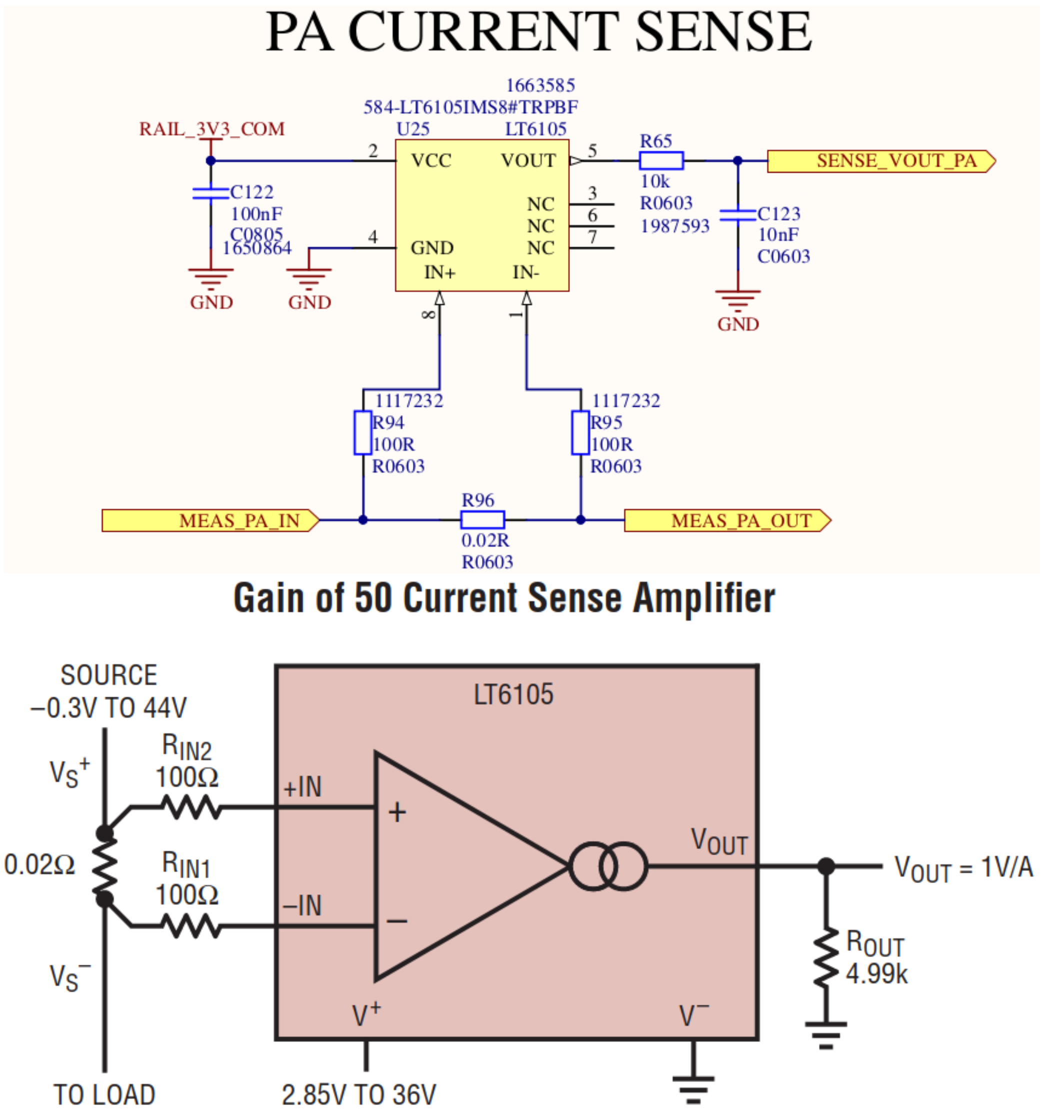


Figure 15: The top portion of the image shows one of the revision 4 current sense amplifiers that were tasked with measuring current consumption on different power lines. The bottom half shows the suggested wiring diagram by the manufacturer for the LT6105 current sense amplifier [53].

Furthermore, after a discussion within the team, the author concluded that keeping the COM EPS and DC/DC switches OFF by default was not a good idea. It was decided to keep these power switches ON by default, and the subsystems controlling the switch would need to actively keep it OFF by asserting a logic HIGH to the power switch's control pin. If one of the controlling subsystems were to fail, then the switch would default to open, ensuring that the needed power is supplied to the communications subsystem. Another benefit that stemmed from that discussion was the discovery that the EPS had no power switches on the battery rails that were forwarded to the COM subsystem. This meant that the power cycling of the COM revision 4 was impossible due to an LDO which was always providing power.

6.3 Primary communications system

The primary communications block consists of an STM32L496ZG microcontroller [41], two LM26LV analog temperature sensors, one SPI and one quad serial peripheral interface (QSPI) FRAM, an external digital to analog converter that controls an external TCVCXO, a UHF transceiver, along with the ICP communications block. A block diagram of the components and their interconnection can be seen in the figure 16.

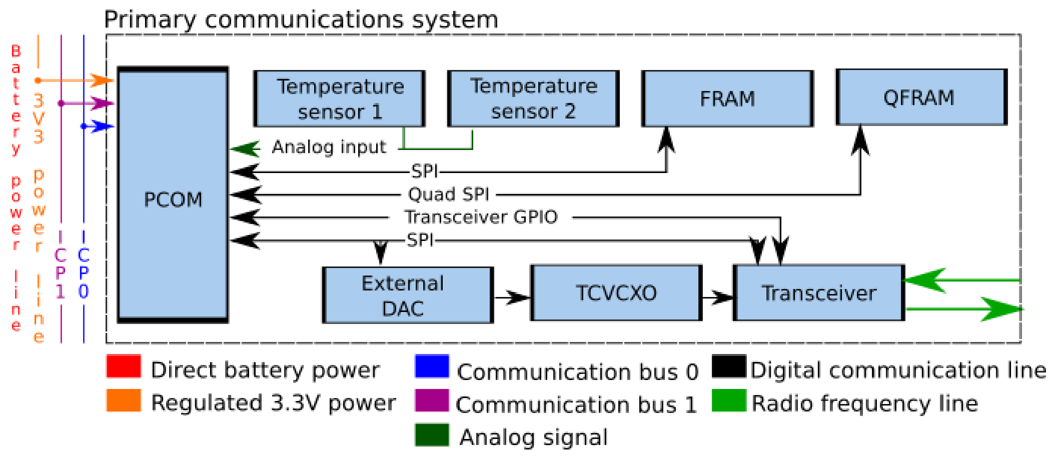


Figure 16: A block diagram of the components of the primary communications system on the revision 4 prototype.

6.3.1 Issues discovered

During the primary communications block testing, several issues were found; some of them were found to be problems across multiple ESTCube-2 subsystems. The list of the most important issues found during testing are as follows:

- Each subsystem used its own programming adaptor nomenclature and pinout.
- The QSPI FRAM footprint was wrong.
- The LM26LV temperature sensor output was routed wrong.
- The TCVCXO switch attenuated the reference signal too much.
- The external transceiver XOUT pin was tied to GND.
- The external transceiver SPI lines were routed wrong.
- The external SPI FRAM memory communication lines were routed incorrectly.
- The external digital to analogue converter was not connected to any SPI lines.
- The reset line between SCOM and PCOM was connected to primary communications system microcontroller JTAG NJTRST, not NRST pin.

First, it was discovered that revision 4 used a different debugger pin layout than the other subsystems due to the lack of a standard for all ESTCube hardware developers to follow. The author coordinated a standard to be used on all subsystems that used the serial wire debug capabilities on the STM32 microcontrollers. Serial wire debug requires fewer pins than a regular JTAG connection and can provide all the needed flashing and debugging capabilities that the ESTCube team needs.

Secondly, there were many problems with SPI signal naming throughout different subsystems. Every developer used a different way of marking the signals on the SPI bus, creating overall confusion. That is the suspected cause for the issues regarding the SPI connection mismatches in revision 4 for both the FRAM memory and the external transceiver. For both integrated circuits, the master in slave out (MISO) and master out slave in (MOSI) data lines were misconnected, with all MISO and all MOSI pins on the SPI lines being connected.

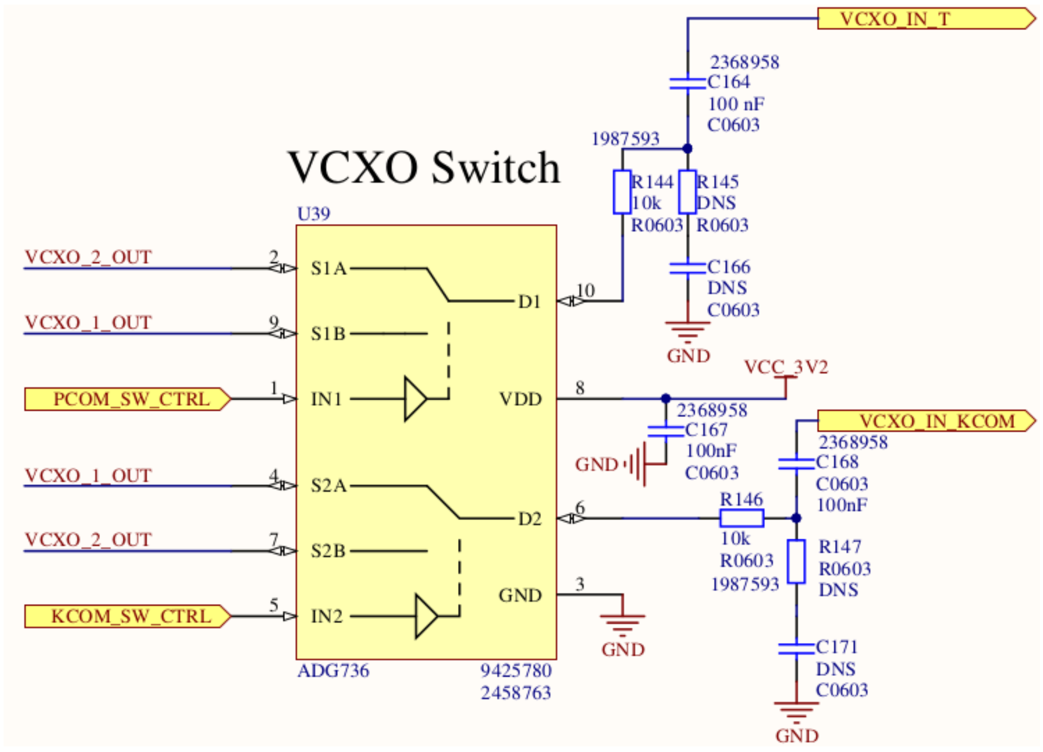


Figure 17: The TCVCXO switch present in the revision 4 prototype.

Finally, the questionable addition of a switch to the outputs of both RF reference oscillators can be seen in the figure 17. The rationale behind this design decision was that if one transceiver reference oscillator failed, the second oscillator on the circuit board could be used for reference for both the primary and secondary communications systems. However, this meant placing both oscillators far from the circuit they were supposed to reference, reducing the overall signal quality. Additionally, the switch itself could not pass a 32 MHz signal without attenuating it too much, making it unusable for both transceivers. Thus it was decided to remove it altogether, as the possible benefit for having two oscillators that could be used for referencing either transceiver was not worth the costs associated with poor component placement options along with degraded signal quality.

6.4 Secondary communications system

The secondary communications system was rather simple, consisting of a EZR32WG330 microcontroller with an inbuilt transceiver, an external SPI FRAM, a functional block for resetting the PCOM and its transceiver, a reference oscillator and a VHF reception chain. The VHF reception chain consists of a high order low-pass filter that shields it from the high powered UHF transmissions and a surface acoustic wave (SAW) band-pass filter for further noise reduction. A simplified representation of the secondary communications system, along with the connections between its components, can be seen in figure 18.

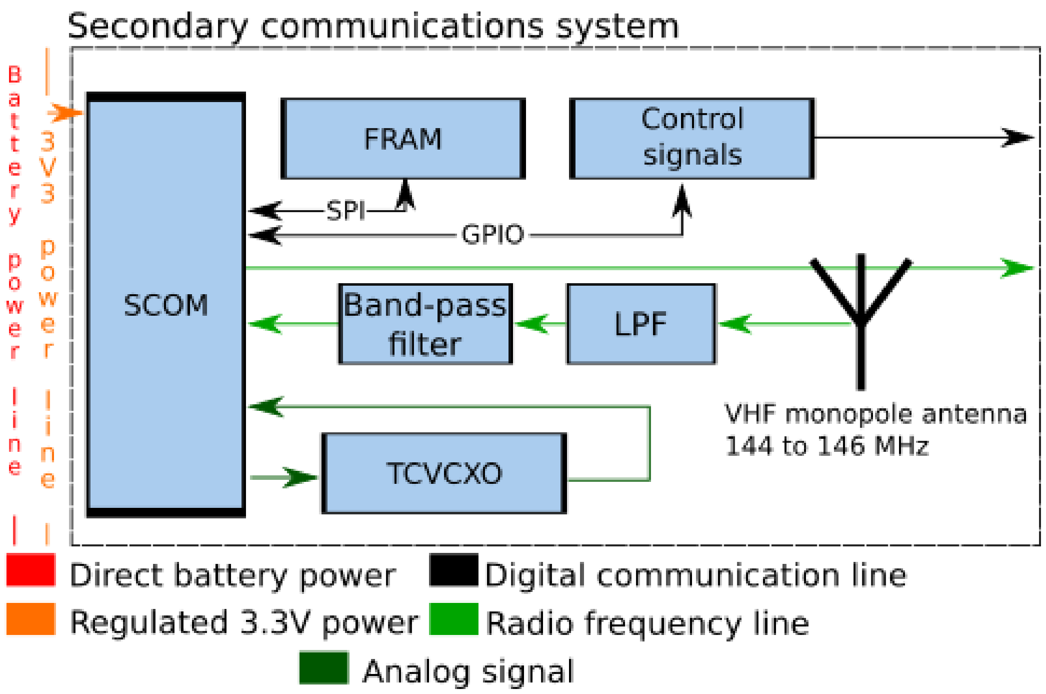


Figure 18: A block diagram of the most important components making up the secondary communications system on revision 4.

6.4.1 Issues discovered

The secondary communications block had a fairly small number of issues due to its simplicity. However, there were still some significant problems found that are listed below:

- SCOM microcontroller used a 8 MHz crystal as its low-frequency external oscillator.
- SCOM SAW filter footprint was mirrored.
- Lack of RF test points on the VHF reception chain.

- Values for SCOM active low-pass filter had not been calculated.
- Most of the component footprints on the SCOM signals block were of the wrong package.

An inappropriate crystal had been selected for the low-frequency oscillator of the SCOM microcontroller. An 8 MHz oscillator was used instead of the required 32.768 kHz oscillator. Secondly, the SAW filter footprint on the VHF reception line had been created with the pinout mirrored, making the testing of the SAW filter infeasible. Additionally, there were not enough test points on the VHF radio reception chain, making it impossible to test the transceiver, SAW filter and low-pass filter components individually. Finally, some functional blocks had component values that had not been calculated, and wrong component footprints were used for the SCOM signals block. A functional block that SCOM uses to send out reset signals to the PCOM microcontroller and transceiver.

6.5 UHF transmission and reception chain

The UHF transmission and reception chain consist of a Wilkinson power divider, used as a combiner, power amplification circuitry, a power measurement block, an antenna switch, a low-pass filter, a SAW filter and an inverted F antenna on the body of the satellite. A flow chart showing the most important components and their interconnections can be seen in the figure 19.

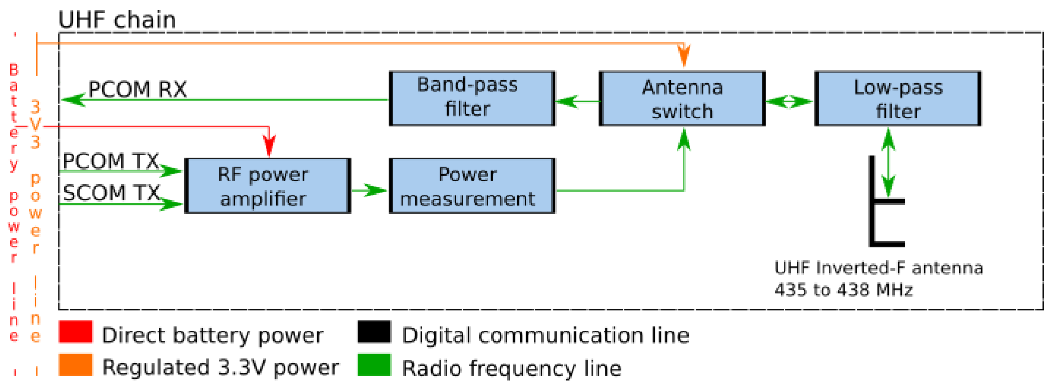


Figure 19: A block diagram showing the main components of the ultra high frequency radio chain on the revision 4.

6.5.1 Issues discovered

As with other functional blocks, many issues were discovered while testing. There were problems with component footprints, the bill of materials, inferior component placing and incorrect passive component values, most of which were minor. The only two significant issues are listed below.

- Operational amplifier power rail used for RF transistor biasing was routed wrong.
- Impedance mismatches on the RF track and long sections of microstrip.

The operational amplifier used for biasing the RF amplification transistor was supplied from the wrong power rail. The operational amplifier was supplied directly from a power rail that was meant to be used for charging the satellites' batteries, not from the designated `PWR_BAT` rail that can be seen in figure 13. Secondly, there were many places on the circuit board where the impedance of the RF track changed when moving from one component to another, along with long sections of microstrip that attenuated the signal even further, shown in figure 20. While it was still possible to receive and transmit through the UHF chain, these problems reduced the overall efficiency of the transmission chain. Furthermore, they deteriorated the ability to detect weak signals on the reception path.

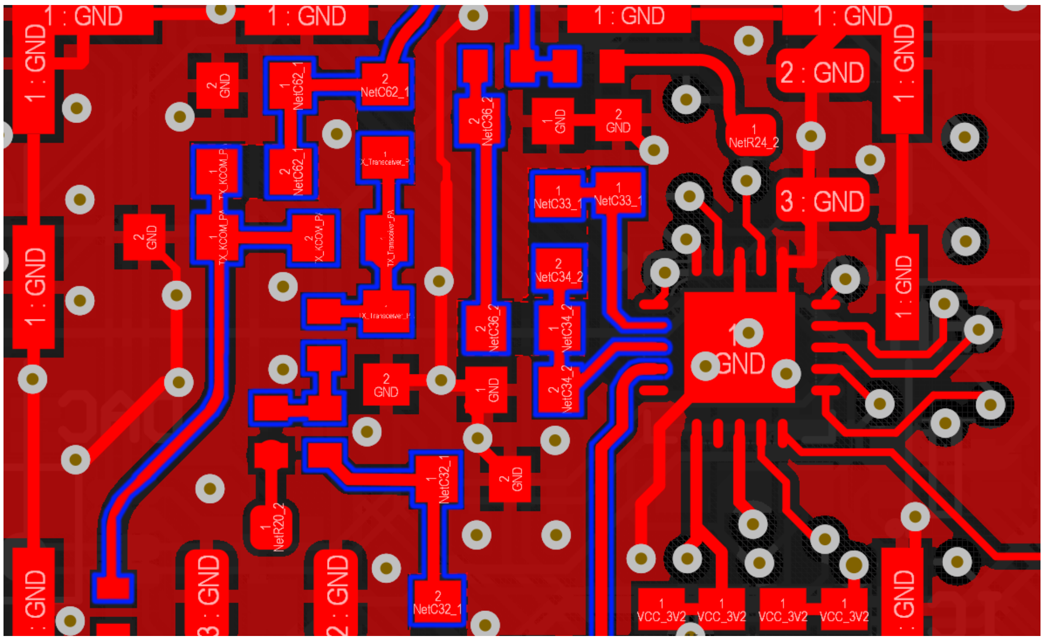


Figure 20: RF track impedance mismatches between passive components and unnecessarily long sections of microstrip in the revision 4 UHF chain.

6.6 External connectors

The communications subsystem connects to other subsystems in the satellite through bus connectors. Bus connectors start from the bottom of the satellite bus, and the signals propagate vertically until the topmost printed circuit board. These connectors are present on every subsystem, and they share the same signal mapping throughout the satellite bus. An image of the ESTCube-2's compact bus and its connectors can be seen in figure 21.

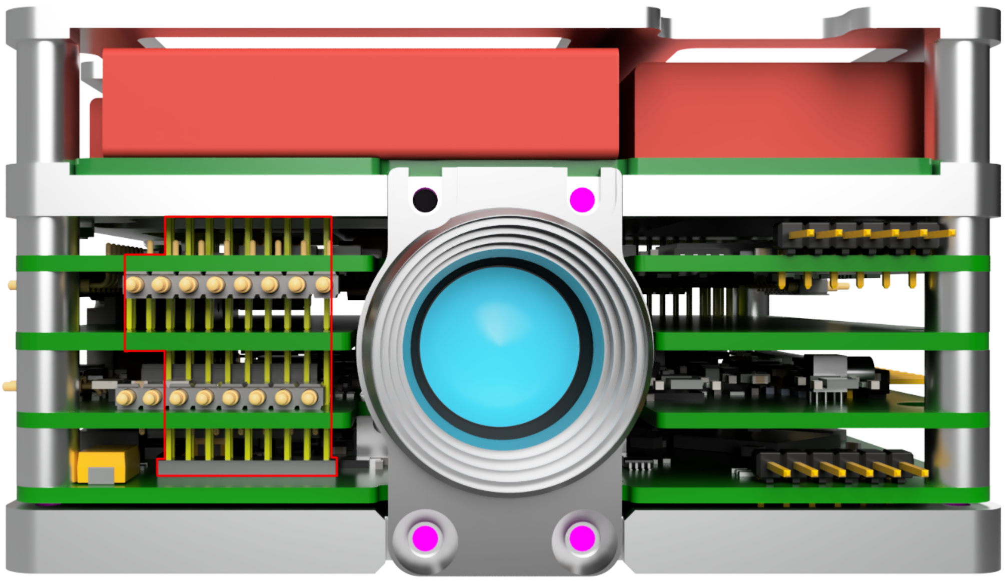


Figure 21: An image of the ESTCube-2 bus along with the bus connectors that can be seen on the left.

6.6.1 Issues discovered

The external connector block consists of connectors between the subsystems and pogo pin connectors for the side panels on the edge of the bus. The most significant issues discovered while testing the external connectors are shown below.

- Bus connector locations on the circuit board were wrong.
- Bus connector signals names were non-standard along with issues in the pinout.

As expected, there were issues with the connectors' location and different signals being on the same pin between multiple subsystems. There was no way to fix these problems other than a redesign of the circuit board as some of the connections had been made in the internal layers of the circuit board. All these problems were noted down and would be fixed in a future revision.

7 Design of the engineering model

Revision 4 was not usable for further development due to the extensive number of issues discovered, listed in appendix A. That is why the focus shifted to creating an engineering model, a replica of the flight model, that would also resolve the issues discovered in the previous prototype. The design of the engineering model is primarily based on revision 4; however, there are some architectural differences. The manufactured printed circuit board can be seen in appendix B.

The following sections are meant to provide an overview of the difference between revision 4 and the engineering model along with the rationale behind the decisions. As with the revision 4, the entire subsystem was separated into different functional blocks, and each block was redesigned one by one. The functional blocks making up the engineering model are similar to revision 4, and the hardware overview can be seen in figure 22.

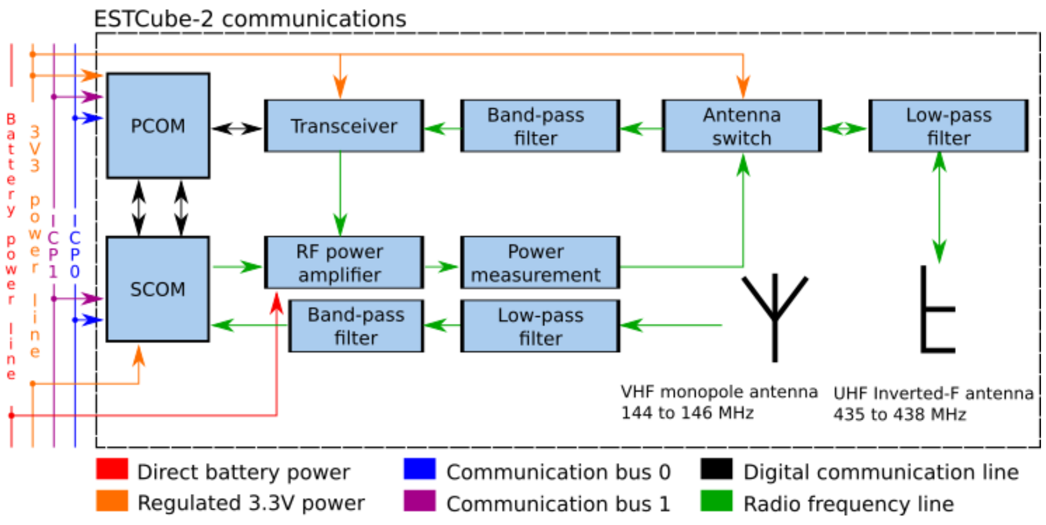


Figure 22: An overview of the most important components in the engineering model and how they interact with each other.

7.1 Power management

The power management block largely remained, as shown in figure 12. The problems found in revision 4 were fixed, including the removal of the backup LDO on the 3.3 V power rail. The EPS did not have any room on its circuit board for an additional power switch, and the benefit of having a backup LDO was deemed low. Removing the backup LDO enabled power cycling the entire COM subsystem, enabling risk mitigation against single event upsets or latch-ups. The redesigned 3.3 V power rail can be seen in figure 23.

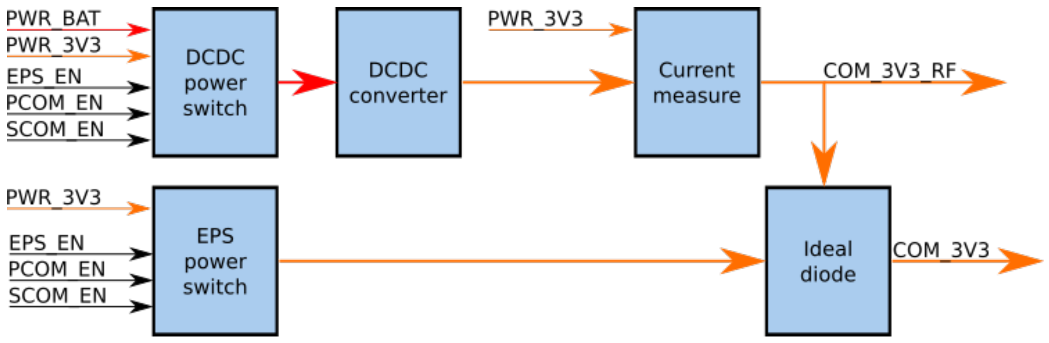


Figure 23: A block diagram of the 3.3 V power distribution on the engineering model.

7.2 Primary communications system

The primary communications system largely remained without changes, except for removing the external digital to analogue converter. The selected STM32L496ZG microcontroller [41] has an internal digital to analogue converter and two output channels, one was connected to the transceivers TCVCXO control input and the other was used for RF transistor gate biasing. The incorrectly connected SPI lines were fixed and the author led an effort that standardised naming conventions throughout the ESTCube-2 subsystems and enforced the new standard. The block diagram of the primary communications subsystem on the engineering model can be seen in figure 24.

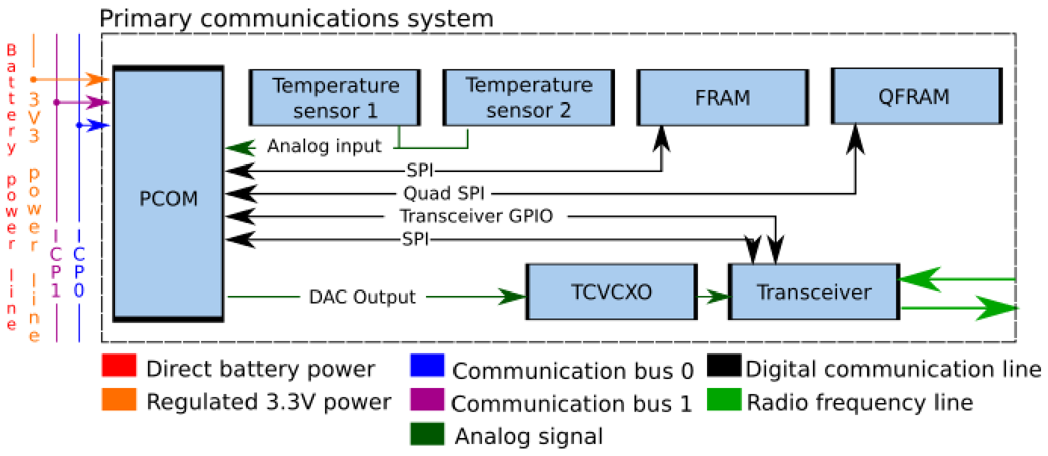


Figure 24: A block diagram of the components making up the primary communications system on the engineering model.

7.3 Secondary communications system

The autonomy of the secondary communications system was increased, making it a separate communications subsystem by itself. The new system architecture diagram can be seen in the figure 25.

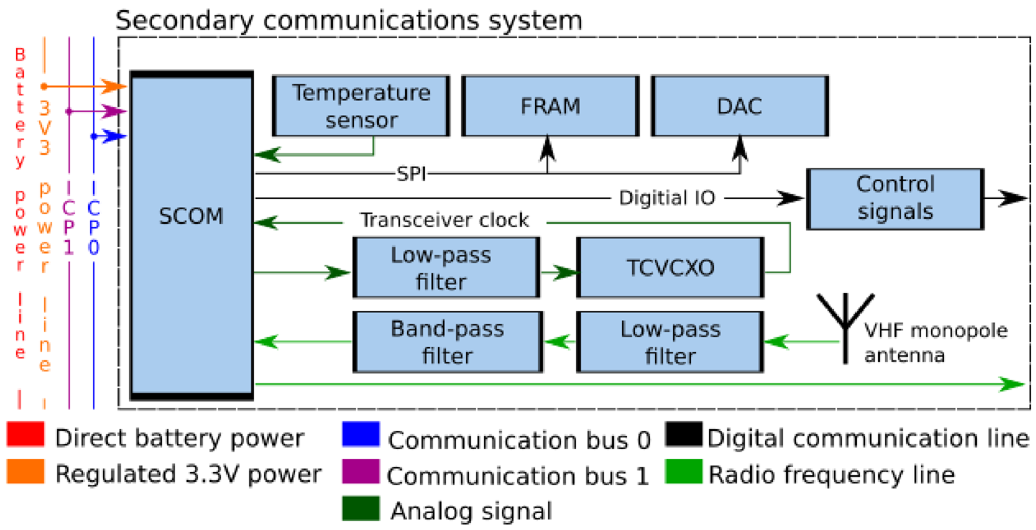


Figure 25: A block diagram of the components making up the secondary communications system on the engineering model.

In revision 4 the secondary COM relied on PCOM in order to transmit on UHF frequencies. SCOM could not control the power amplification transistor nor the RF switch that connected the antenna to either the reception or the transmission chain. If the primary communications subsystem were to fail, then even if the SCOM were still alive and capable of transmitting, it would have been unable to do so because it did not have control of needed components. It was decided to connect the RF switch control input with a simple OR switch that either subsystem could control. Secondly, a voltage adding scheme was designed to input the operational amplifier for biasing the RF transistor. This means that either subsystem can transmit independently on the engineering model, but an arbitration scheme would be needed for sharing the joint resources.

Additionally, the SCOM was connected to the ICP bus directly, instead of it needing to go through PCOM to forward data to the rest of the satellite. This gives the satellite two different subsystems capable of forwarding packets to Earth, hopefully increasing the overall reliability of the satellite. Moreover, connecting SCOM to the satellite main communication bus enables full-duplex communications with the spacecraft. During communication passes, the ground station can uplink data on the VHF band, and the satellite can downlink data on the UHF band at the same time.

Finally, an external digital to analogue converter was added to the SCOM for RF transistor bias control along with an analogue temperature sensor.

The temperature sensor was placed near the external TCVCXO. It will be used for compensating oscillator drift caused by varying temperatures when the satellite passes from the eclipse into the sunlight when in orbit.

7.4 UHF transmission and reception chain

The UHF transmission and reception chain largely remained the same with the most significant changes in component placement. Instead of connecting passive components using microstrip matched to 50 ohms, it was decided to connect most passive components side by side, as shown in the figure 26. This reduces the radio frequency chain sections where the track width, and consequently the impedance, change, hopefully improving the overall efficiency of the RF circuitry.

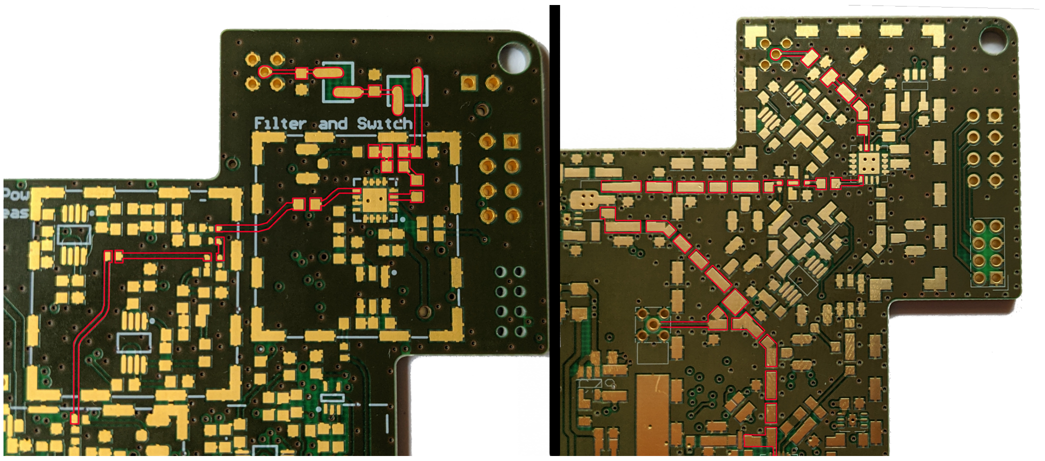


Figure 26: A picture of the UHF transmission path on the revision 4 (left) and the engineering model (right) circuit board.

Another notable change is that custom electromagnetic interference shielding gaskets will be used on the communications subsystem. The unusual shape of the circuit board, shown in figure 8, partly caused the issues with the component layout in revision 4. The commercially available gaskets were not available with any optimal shapes required for the circuit board, thus creating long sections of microstrip where the signal was routed between gaskets.

Finally, an experimental change was made to the UHF transmission chain shown in figure 27. As microstrips are inefficient for transferring RF signals and they are susceptible to interference from the environment, it was decided to use a coaxial cable for connecting SCOM to the UHF transmission chain [54]. It is hoped that by using a coaxial cable, the RF losses could be kept to a minimum, even though it presents additional difficulties during the assembly phase.

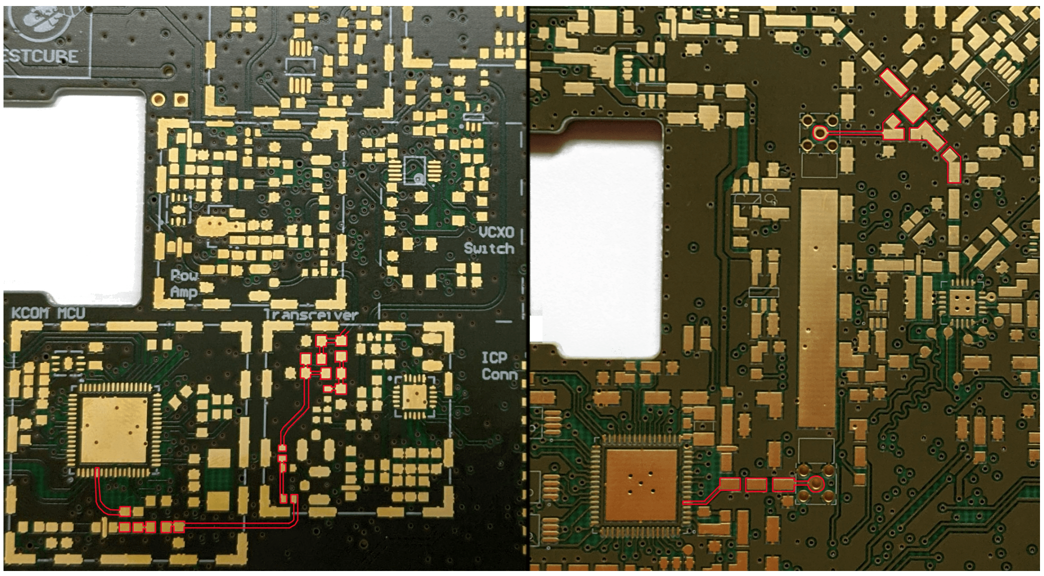


Figure 27: A picture of the SCOM UHF transmission path (outlined in red) on the revision 4 (left) and engineering model (right) circuit boards.

7.5 External connectors

During the engineering model development, more issues were found in the design of all subsystems that had also been overlooked in the previous designs for every subsystem. The biggest problems being that none of the subsystems were connected to the remove before flight connector.

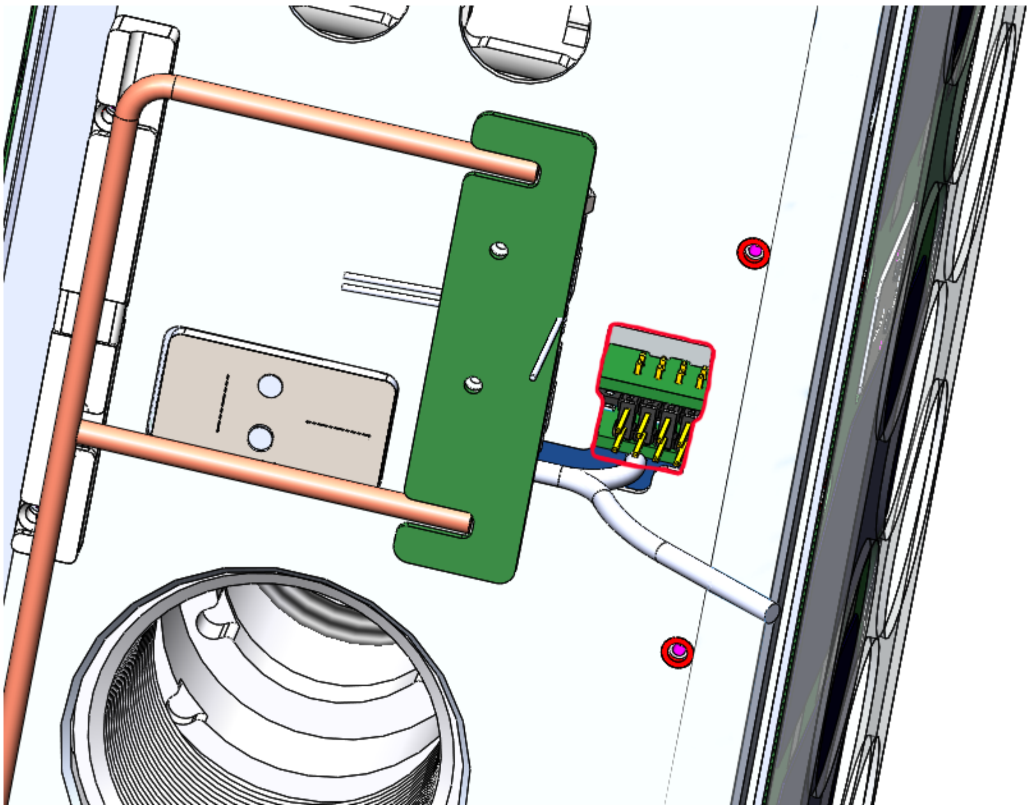
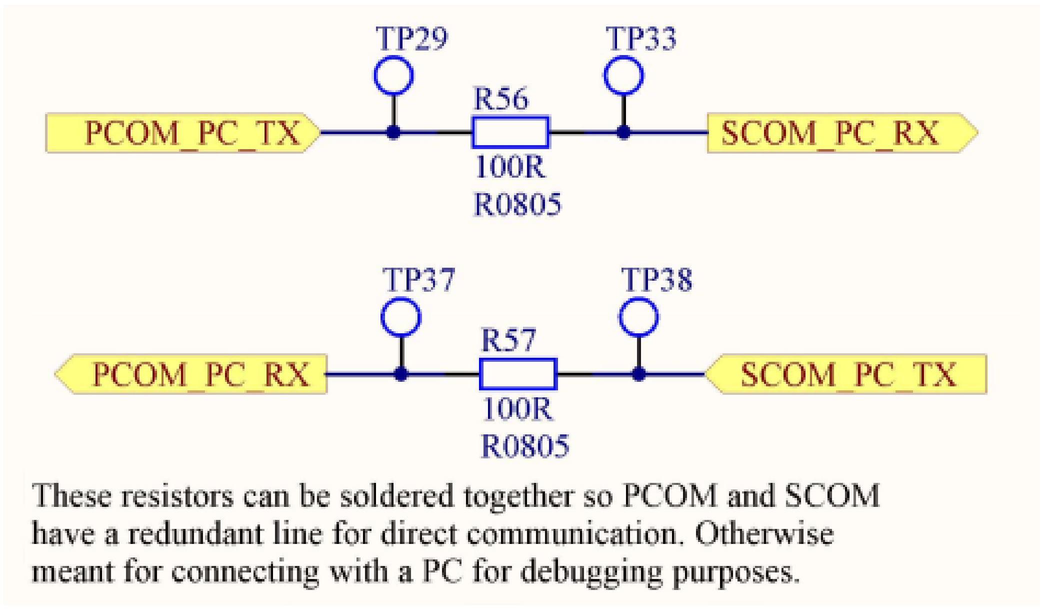


Figure 28: The remove before flight connector that will be used on ESTCube-2.

The remove before flight connector, seen in the figure 28, is used for controlling the satellite once it is fully assembled. By varying the signal lines connected with jumper caps, different combinations of control signals, `0000, 0001, 1101`, can be sent to the satellite. Each subsystem reads the remove before flight signals while booting and switches modes according to the signals present on the pins.

Finally, standardised adaptors were added for programming PCOM and SCOM that conformed to the standard that was decided by the ESTCube-2 team previously. In addition, serial communication lines, figure 29, were routed from both PCOMs to the edge of the prototype's circuit board as an additional debugging tool.



TP29
TP33
R56
PCOM_PC_TX
SCOM_PC_RX
100R
R0805
TP37
TP38
R57
PCOM_PC_RX
SCOM_PC_TX
100R
R0805
These resistors can be soldered together so PCOM and SCOM have a redundant line for direct communication. Otherwise meant for connecting with a PC for debugging purposes.

Figure 29: Engineering model serial connections for additional debugging capability.

This gives the subsystem developers the ability to perform functional tests with the COM subsystem prototype by sending it ICP commands over plain serial using an external serial to USB adapter and a computer. Once the testing is done, the two serial lines could be connected with a resistor, giving PCOM and SCOM an additional direct communication line.

8 Conclusion

This master's thesis aimed to test communications subsystem hardware along with software for the ESTCube-2 nanosatellite. The thesis covers work on two hardware prototypes and describes the creation of the communications subsystem engineering model, a copy of the flight model. The key contributions during this masters thesis were:

- refactoring of the ESTCube HAL for the MSP430 architecture;
- the addition of an embedded operating system to the communications subsystem firmware;
- restructuring of COM software into logical layers;
- discovery of computational problems with the MSP430 microcontroller;
- revision 4 hardware testing and documenting the problems found along with potential fixes;
- the creation of an engineering model.

Additionally, the author standardised signal naming conventions, created a common programming adaptor for ESTCube hardware prototypes, harmonised signal routing and bus connectors locations between ESTCube-2 subsystems and coordinated the use of the remove before flight connector within integrated bus. Creation of standards within the ESTCube project is important as it speeds up the development process and maintains consistency between subsystems.

Finally, due to the extensive hardware problems discovered while testing revision 4, no significant high level firmware development could be done for the new STM32L4 architecture. That is expected to change with the developed engineering model and initial hardware test with it have been performed, further testing is ongoing. However, the continued testing of engineering model hardware and accompanying firmware development is beyond the scope of this thesis.

References

[1] Erik Kulu. *Nanosats Database.* URL: https://www.nanosats.eu/. (accessed: 01.12.2020).

[2] The CubeSat Program. *CubeSat Design Specification REV 13.* Cal Poly SLO. 2015. URL: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf.

[3] National Aeronautics and Space Administration. *Orbit.* URL: https://web.archive.org/web/20131231000203/http://gcmd.nasa.gov/add/ancillaryguide/platforms/orbit.html. (accessed: 02.11.2020).

[4] Harry W. Jones. "The recent large reduction in space launch cost". In: 48th International Conference on Environmental Systems. 2018. URL: https://ntrs.nasa.gov/citations/20200001093.

[5] Armen Poghosyan and Alessandro Golkar. "CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions". In: *Progress in Aerospace Sciences* 88 (2017), pp. 59–83. ISSN: 0376-0421. DOI: https://doi.org/10.1016/j.paerosci.2016.11.002. URL: https://www.sciencedirect.com/science/article/pii/S0376042116300951.

[6] European Space Agency. *Satellite frequency bands.* URL: https://www.esa.int/Applications/Telecommunications_Integrated_Applications/Satellite_frequency_bands. (accessed: 02.11.2020).

[7] Bryan Klofas, Jason Anderson, and Kyle Leveque. "A survey of cubesat communication systems". In: *5th Annual CubeSat Developers' Workshop.* 2008, pp. 1–36. URL: https://www.klofas.com/papers/CommSurvey-Bryan_Klofas.pdf.

[8] *The European table of frequency allocations and applications in the frequency range 8.3 kHz to 3000 GHz.* Tech. rep. Electronic Communications Committee, Nov. 20, 2020. URL: https://docdb.cept.org/document/593.

[9] T. S. Rose et al. "Optical communications downlink from a 1.5U Cubesat: OCSD program". In: *International Conference on Space Optics — ICSO 2018.* Ed. by Zoran Sodnik, Nikos Karafolas, and Bruno Cugny. Vol. 11180. International Society for Optics and Photonics. SPIE, 2019, pp. 201–212. DOI: 10.1117/12.2535938. URL: https://doi.org/10.1117/12.2535938.

[10] *Modulating Retro-Reflectors: Technology, Link Budgets and Applications.* NASA STI Repository, Sept. 12, 2012. URL: https://ntrs.nasa.gov/citations/20120016693.

[11] Mark Dmytryszyn, Matthew Crook, and Timothy Sands. "Lasers for Satellite Uplinks and Downlinks". In: *Sci* 3.1 (2021). ISSN: 2413-4155. DOI: 10.3390/sci3010004. URL: https://www.mdpi.com/2413-4155/3/1/4.

[12] *Lessons Learned in the First Year Operating Software Defined Radios in Space.* NASA STI Repository, Aug. 4, 2014. URL: https://ntrs.nasa.gov/citations/20150002098.

[13] Mamatha R. Maheshwarappa and Christopher P. Bridges. "Software defined radios for small satellites". In: *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS).* 2014, pp. 172–179. DOI: 10.1109/AHS.2014.6880174.

[14] Emre Baceski et al. "HAVELSAT: A software defined radio experimentation CubeSat". In: *2015 7th International Conference on Recent Advances in Space Technologies (RAST).* 2015, pp. 831–834. DOI: 10.1109/RAST.2015.7208455.

[15] Alper Genc et al. "Active integrated meshed patch antennas for small satellites". In: *Microwave and Optical Technology Letters* 54.7 (2012), pp. 1593–1595. DOI: https://doi.org/10.1002/mop.26906. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mop.26906.

[16] P. Keith Kelly. "A Scalable Deployable High Gain Antenna -DaHGR". In: *Communications.* 2016. URL: https://digitalcommons.usu.edu/smallsat/2016/TS7Communication/6/.

[17] John Hanson et al. "The EDSN Intersatellite Communications Architecture". In: *CubeSat Developers' Workshop.* 2014. URL: https://digitalcommons.usu.edu/smallsat/2014/Workshop/1/.

[18] Jouni Envall et al. "E-sail test payload of ESTCube-1 nanosatellite". In: *Proceedings of the Estonian Academy of Sciences* 63 (Apr. 2014), pp. 210–221. DOI: 10.3176/proc.2014.2S.02.

[19] "ESTCube-1 in-orbit experience and lessons learned". In: *IEEE Aerospace and Electronic Systems Magazine* 30.8 (2015), pp. 12–22. DOI: 10.1109/MAES.2015.150034.

[20] Hendrik Ehrpais et al. "Nanosatellite spin-up using magnetic actuators: ESTCube-1 flight results". In: *Acta Astronautica* 128 (2016), pp. 210–216. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2016.07.032. URL: https://www.sciencedirect.com/science/article/pii/S0094576515302216.

[21] *IARU Amateur Satellite Frequency Coordination.* URL: http://www.amsatuk.me.uk/iaru/finished_detail.php?serialnum=171. (accessed: 10.02.2021).

[22] The International Amateur Radio Union. *Controlling space station transmitters.* URL: https://www.iaru.org/wp-content/uploads/2019/12/controllingsatellites_v27.pdf. (accessed: 10.02.2021).

[23] TalTech Satellites. *Mektrory satelliidiprogramm.* URL: https://satellite.taltech.ee/#/. (accessed: 17.02.2021).

[24] The International Amateur Radio Union. *IARU Amateur Satellite Frequency Coordination.* URL: http://www.amsatuk.me.uk/iaru/finished_detail.php?serialnum=565. (accessed: 17.02.2021).

[25] The International Amateur Radio Union. *IARU Amateur Satellite Frequency Coordination.* URL: http://www.amsatuk.me.uk/iaru/finished_detail.php?serialnum=660. (accessed: 17.02.2021).

[26] I. Müürsepp. private communication. Feb. 2021.

[27] Aalto University. *Aalto-1 was launched into space a year ago – the Otaniemi ground station is already being prepared for the launch of the next satellites.* URL: https://www.aalto.fi/en/news/aalto-1-was-launched-into-space-a-year-ago-the-otaniemi-ground-station-is-already-being. (accessed: 23.02.2021).

[28] J. Praks et al. "Aalto-1, multi-payload CubeSat: Design, integration and launch". In: *Acta Astronautica* (Jan. 2021). ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2020.11.042. URL: http://dx.doi.org/10.1016/j.actaastro.2020.11.042.

[29] Aalto University. *Aalto-1: the Finnish student satellite project.* URL: https://www.aalto.fi/en/spacecraft/aalto-1-the-finnish-student-satellite-project. (accessed: 23.02.2021).

[30] Jaakko Jussila. "Aalto-1 nanosatelliitin S-kaistan lähetin; S-band transmitter for Aalto-1 nanosatellite". en. G2 Pro gradu, diplomityö; master thesis. June 2013, p. 75. URL: http://urn.fi/URN:NBN:fi:aalto-201307127179.

[31] Matthew Smith et al. “On-orbit results and lessons learned from the ASTERIA space telescope mission”. In: *The Year in Review*. 2018. URL: `https://digitalcommons.usu.edu/smallsat/2018/all2018/255/`.

[32] Jet Propulsion Laboratory. *Arcsecond Space Telescope Enabling Research in Astrophysics (ASTERIA)*. URL: `https://www.jpl.nasa.gov/cubesat/missions/asteria.php`. (accessed: 25.02.2021).

[33] Alessandra Babuscia et al. “Arcsecond Space Telescope Enabling Research in Astrophysics (ASTERIA) Telecommunications”. In: *Near Earth Design & Performance Summary Series* (June 2019). URL: `https://descanso.jpl.nasa.gov/NEDSummary/NED-PSS_ASTERIA_FINAL.pdf`.

[34] Jaanus Kalde. “UHF Communication System for Cubesatellite”. MA thesis. University of Tartu, 2015. URL: `https://dspace.ut.ee/handle/10062/50217`.

[35] Janis Dalbins. “Cubesat tipa satelīta UHF joslas zemas datu plūsmas komunikāciju apakšsistēmas prototipa izstāde un testēšana”. MA thesis. Ventspils Augstskola, 2016.

[36] Erik Amor. “Design and implementation of prototype firmware for ESTCube-2 primary communication subsystem”. BA thesis. University of Tartu, 2018. URL: `https://dspace.ut.ee/handle/10062/60289`.

[37] Klāvs Reinis Ozols. “Nanosatelīta “ESTCube-2 komunikāciju apakšsistēmas laboratorijas modeļa izstrāde”. BA thesis. Ventspils Augstskola, 2019.

[38] *MSP430FR599x, MSP430FR596x Mixed-Signal Microcontrollers*. MSP430FR599x, MSP430FR596x. Rev. D. Texas Instruments. Jan. 2021. URL: `https://www.ti.com/product/MSP430FR5994`.

[39] *HIGH-PERFORMANCE, LOW-CURRENT TRANSCEIVER*. Si4464/63/61/60. Rev. 1.3. Silicon Laboratories. URL: `https://www.silabs.com/wireless/proprietary/ezradiopro-sub-ghz-ics/device.si4463`.

[40] *CY15B104Q 4-Mbit (512 K × 8) Serial (SPI) F-RAM*. CY15B104Q. Rev. E. Cypress. Sept. 2014. URL: `https://www.cypress.com/documentation/datasheets/cy15b104q-4-mbit-512-k-8-serial-spi-f-ram-datasheet`.

[41] *Ultra-low-power Arm Cortex-M4 32-bit MCU+FPU, 100 DMIPS, up to 1 MB Flash, 320 KB SRAM, USB OTG FS, audio, external SMPS*. STM32L496xx. Rev. 14. STMicroelectronics. Feb. 2017. URL: `https://www.st.com/en/microcontrollers-microprocessors/stm32l496zg.html`.

[42] J. D. Day and H. Zimmermann. “The OSI reference model”. In: *Proceedings of the IEEE* 71.12 (1983), pp. 1334–1340. DOI: 10.1109/PROC.1983.12775.

[43] *MSP430FR599x, MSP430FR596x Mixed-Signal Microcontrollers*. MSP430FR599x, MSP430FR596x. Rev. G. Texas Instruments. Aug. 2018. URL: https://www.ti.com/product/MSP430FR5969.

[44] Real Time Engineers Ltd. *The FreeRTOS™ Kernel*. URL: https://www.freertos.org/RTOS.html. (accessed: 10.04.2021).

[45] Real Time Engineers Ltd. *Supported Demos*. URL: https://www.freertos.org/a00090.html#TI. (accessed: 10.04.2021).

[46] Richard Barry. *Mastering the FreeRTOS™ Real Time Kernel*. 161204. Real Time Engineers Ltd. 2016.

[47] Real Time Engineers Ltd. *Texas Instruments MSP430 (MSP430F449) RTOS Port*. URL: https://www.freertos.org/portmspgcc.html. (accessed: 10.04.2021).

[48] Texas Instruments. *MSP430F449*. URL: https://www.ti.com/product/MSP430F449. (accessed: 10.04.2021).

[49] *MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family User’s Guide*. MSP430FR58xx, MSP430FR59xx, MSP430FR6xx. Rev. P. Texas Instruments. Oct. 2012. URL: https://www.ti.com/product/MSP430FR5994.

[50] Texas Instruments. *MSP430-GCC-OPENSOURCE*. URL: https://www.ti.com/tool/MSP430-GCC-OPENSOURCE. (accessed: 05.04.2021).

[51] Dev Bhattacharyya. *CC 6.1 maximum of 16KB of code size for MSP432*. Apr. 2015. URL: https://e2e.ti.com/support/microcontrollers/msp430/f/msp-low-power-microcontroller-forum/415496/cc-6-1-maximum-of-16kb-of-code-size-for-msp432. (accessed: 05.04.2021).

[52] Real Time Engineers Ltd. *TI MSP430FR5969 (MSP430X) RTOS Demo*. URL: https://www.freertos.org/MSP430FR5969_Free_RTOS_Demo.html. (accessed: 10.04.2021).

[53] *Precision, Extended Input Range Current Sense Amplifier*. LT6105. Rev. A. Analog Devices. URL: https://www.analog.com/en/products/lt6015.html.

[54] G. Maloratsky. “Using Modified Microstrip Lines to Improve Circuit Performance”. In: *High Frequency Electronics* 10.3 (2011), pp. 36–56. URL: https://www.highfrequencyelectronics.com/Mar11/HFE0311_Maloratsky.pdf.

A Revision 4, issues discovered

The list below shows all the issues discovered while testing the revision 4 prototype.

- All current sense amplifiers were incorrectly configured.
- LDO current sense output power rail was routed incorrectly.
- `COM_3V3_RF` was left disconnected from the PCOM transceiver.
- There was no way to power cycle the COM subsystem.
- Each subsystem used its own programming adaptor nomenclature and pinout.
- The QSPI FRAM footprint was wrong.
- The LM26LV temperature sensor output was routed wrong.
- The TCVCXO switch attenuated the reference signal too much.
- The external transceiver XOUT pin was tied to GND.
- The external transceiver SPI lines were routed wrong.
- The external SPI FRAM memory communications lines were routed incorrectly.
- The external digital to analogue converter was not connected to any SPI lines.
- The reset line between SCOM and PCOM was connected to primary communications system microcontroller JTAG NJTRST, not NRST pin.
- SCOM microcontroller used a 8 MHz crystal as its low-frequency external oscillator.
- SCOM SAW filter footprint was mirrored.
- Lack of RF test points on the VHF reception chain.
- Values for SCOM active low-pass filter had not been calculated.
- Most of the component footprints on the SCOM signals block were of the wrong package.

- Operational amplifier power rail used for RF transistor biasing was routed wrong.
- Impedance mismatches on the RF track and long sections of microstrip.
- Bus connector locations on the circuit board were wrong.
- Bus connector signals names were non-standard along with issues in the pinout.
- Large covered pads did not have vias for outgassing.
- Wrong component was ordered instead of the L31 inductor.
- Wrong passive component values were specified in the schematic.
- Schematic used the outdated nomenclature of Kill-COM when describing secondary COM signals.
- Air core inductor footprint on the UHF line was wrong.
- Debugging connectors did not have silkscreen.
- Wrong type of feed-through capacitor was ordered.

B Engineering model, printed circuit board

Figures 30 and 31 show the manufactured ESTCube-2 communications subsystem engineering model. Figures 32, 32, 33, 34, 35 and 36 show all the layers of the printed circuit board from top to bottom, as seen in Altium Designer 18.

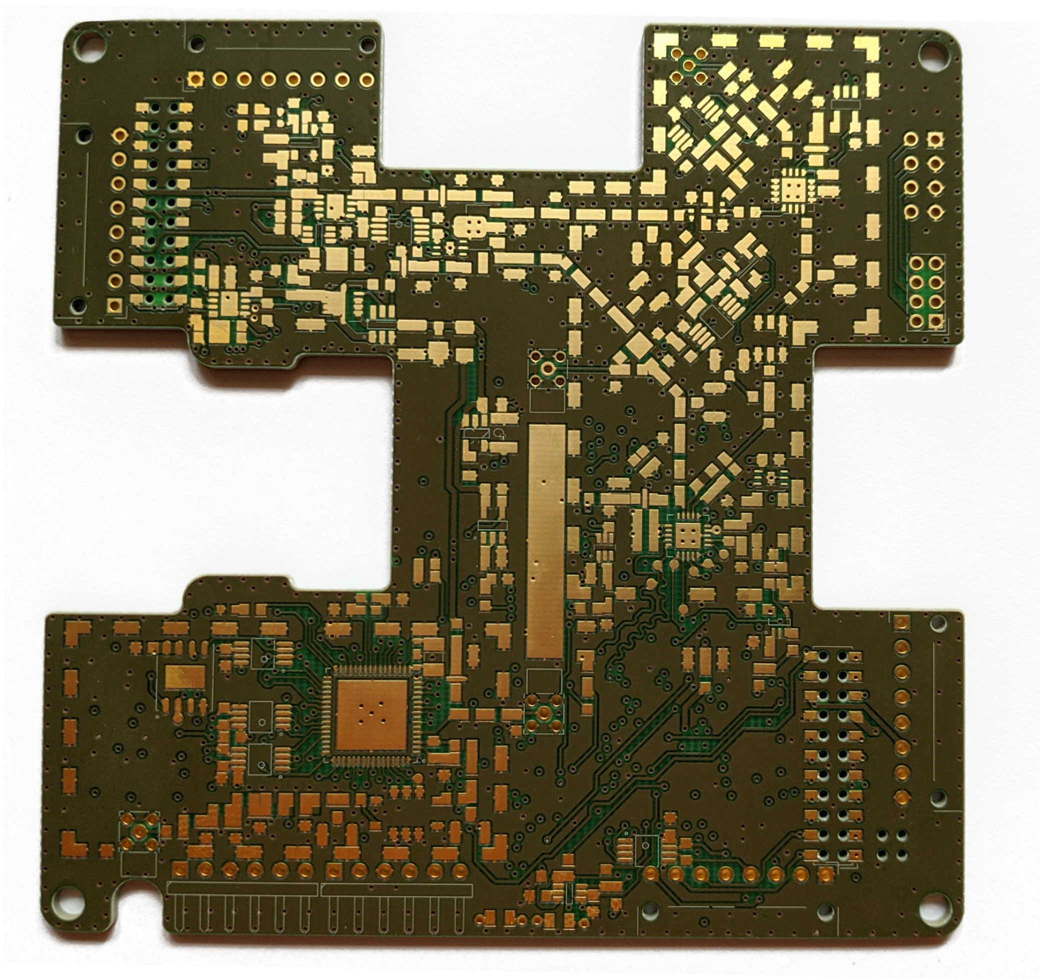


Figure 30: Top side of the engineering model printed circuit board. This side houses all of the components that are related to transmission and reception on radio frequencies along with the secondary communications subsystem.

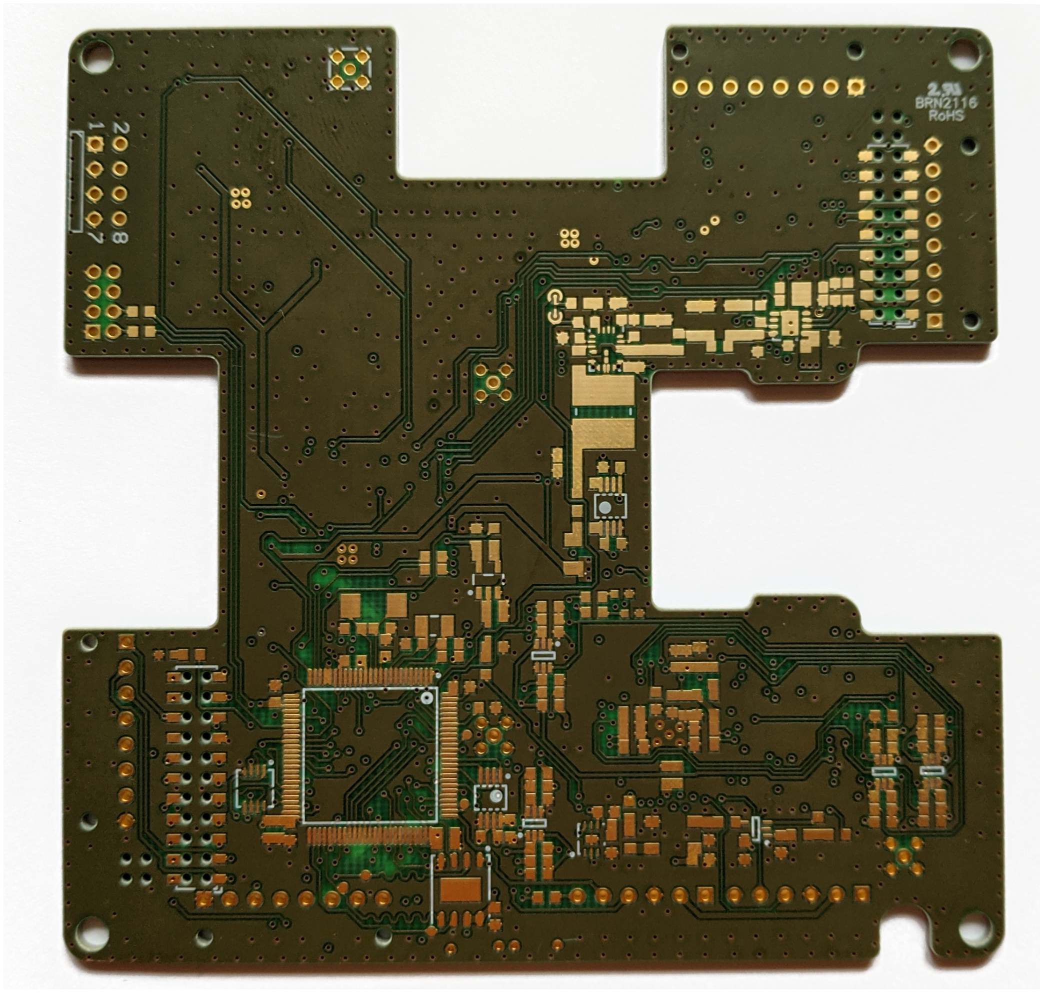


Figure 31: Bottom side of the engineering model printed circuit board. This side houses the primary communications subsystem along with the DC/DC switching regulator.

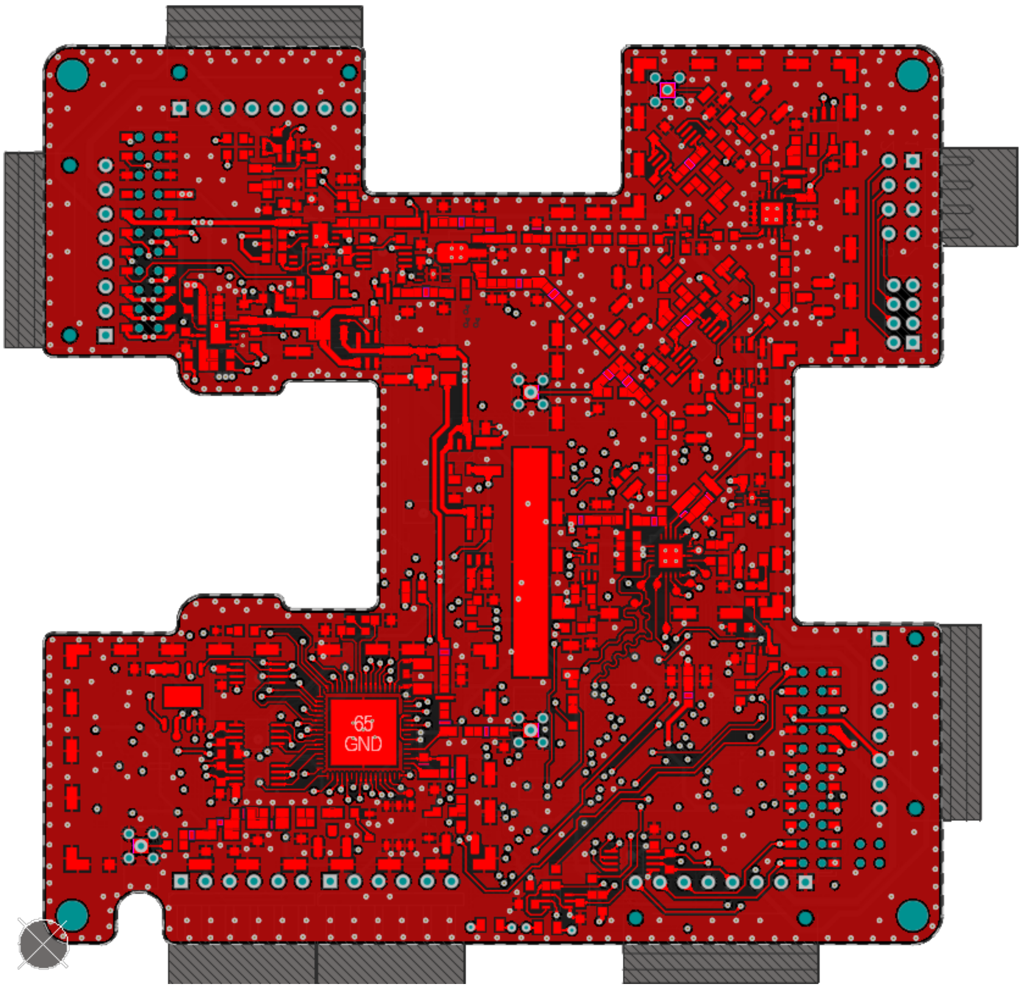


Figure 32: The top layer houses all the RF components, shielding gaskets and the secondary communications subsystem.

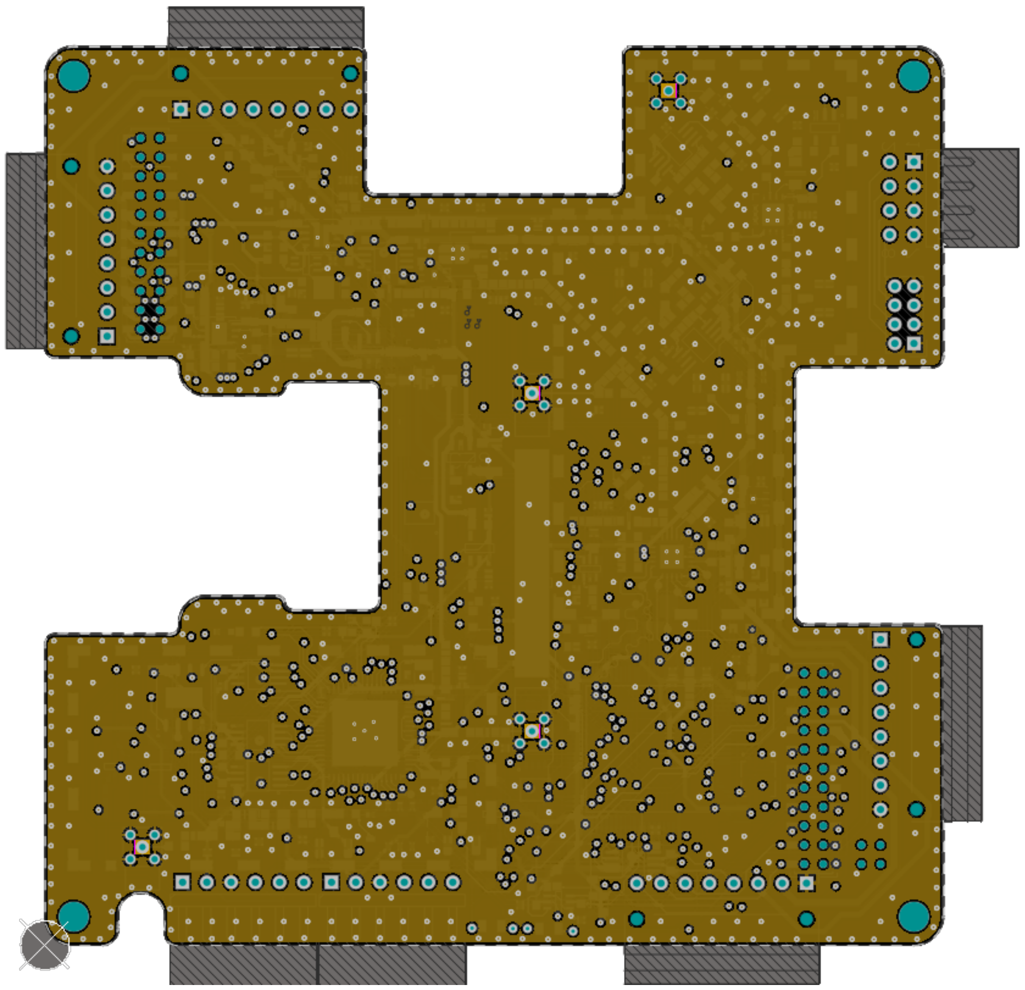


Figure 33: The second layer from the top was allocated for ground. It is meant to provide shielding to the components on the bottom layer of the printed circuit board and serves as a current return path for the integrated circuits.

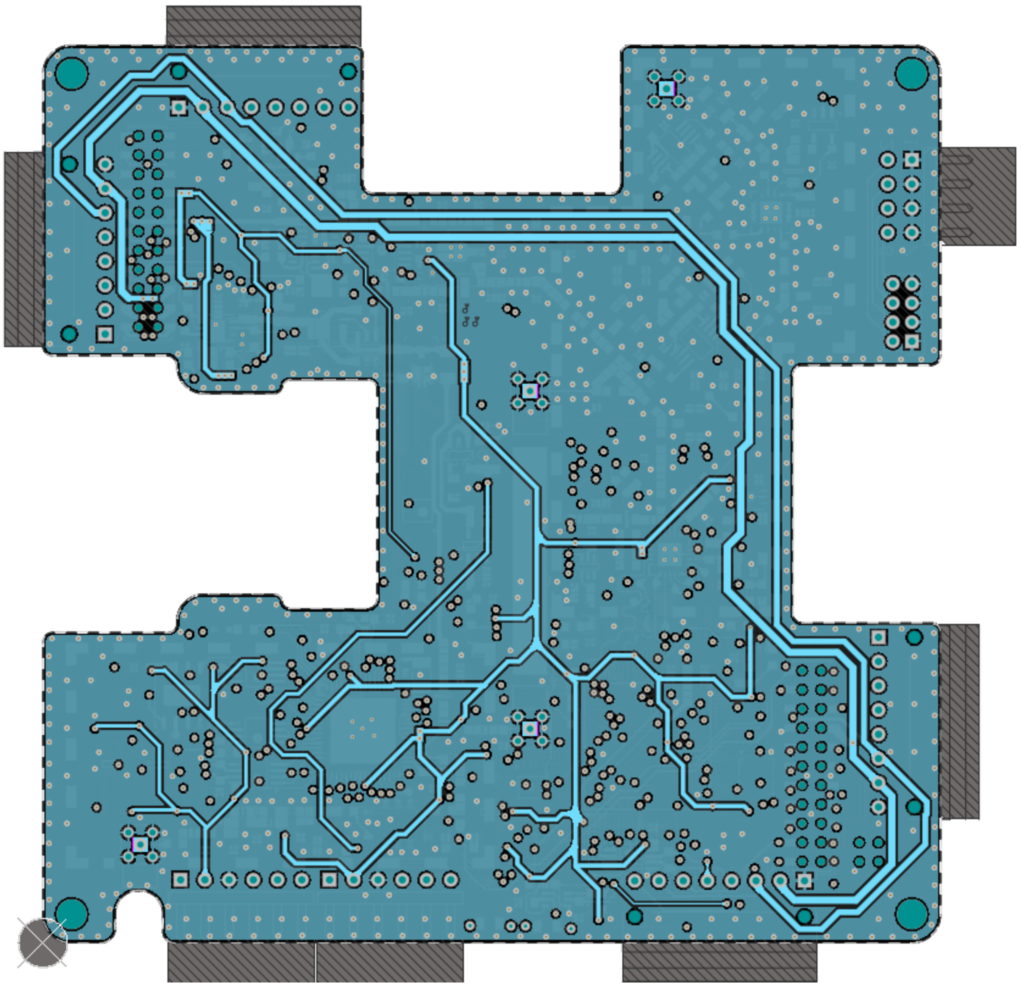


Figure 34: The third layer from the top was allocated for distributing power to various integrated circuits.

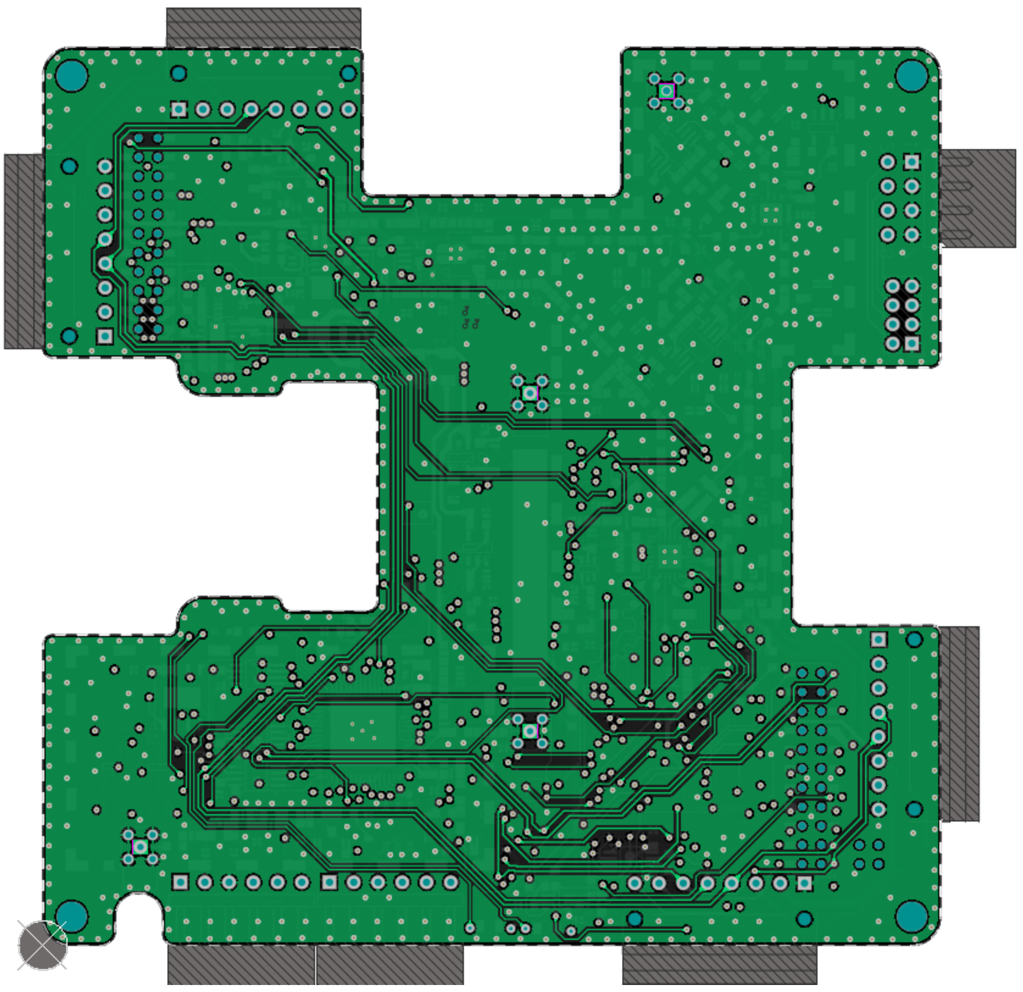


Figure 35: The fourth layer from the top was used for digital signal routing between various components.

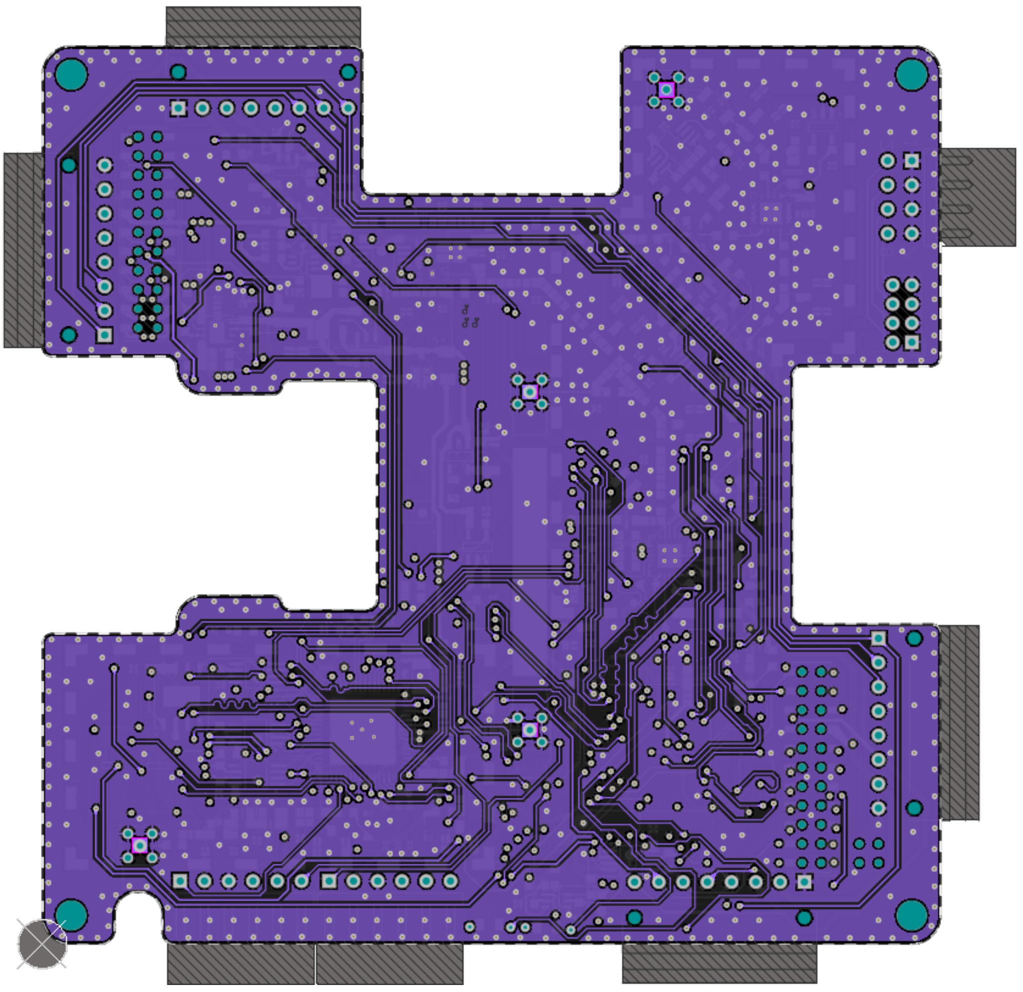


Figure 36: The fifth layer from the top, similar to the forth, was also used for digital signal routing between various integrated circuits.

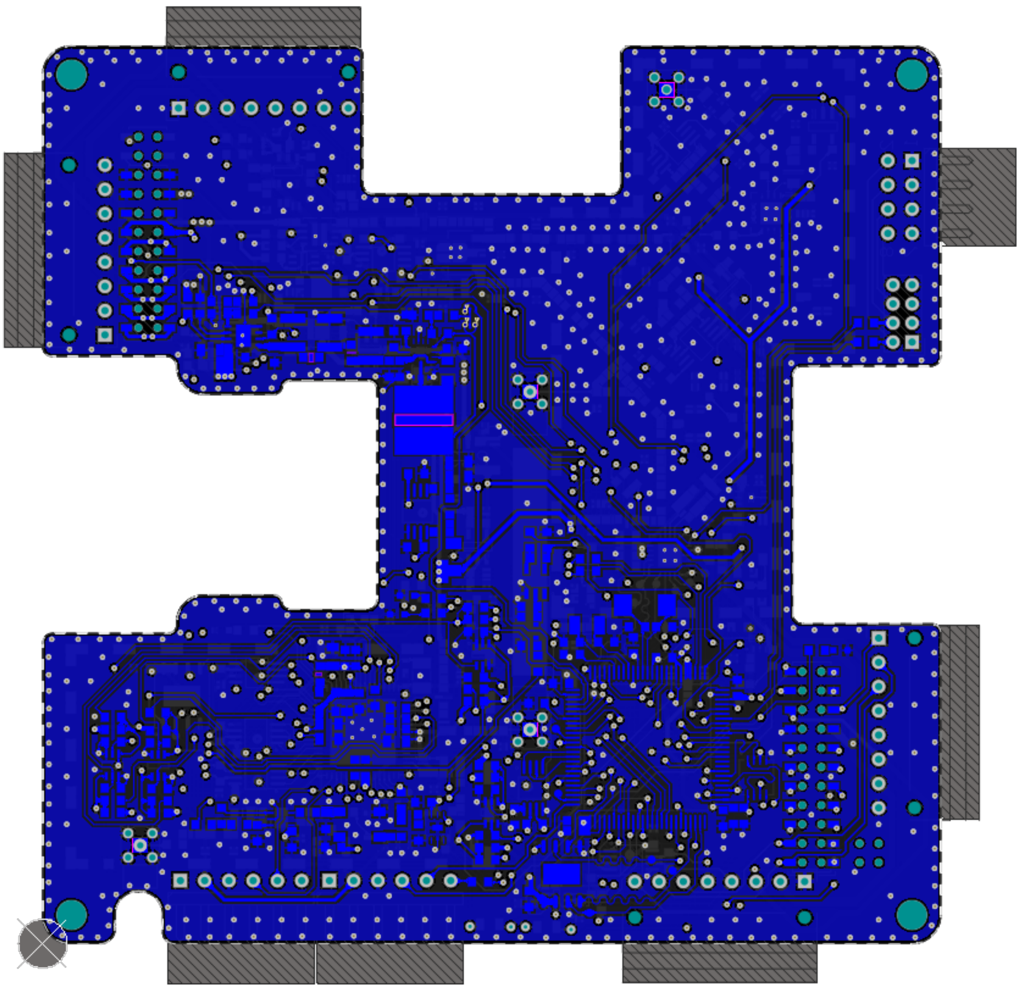


Figure 37: The sixth layer from the top, houses the primary communications subsystem along with components that might be susceptible RF interference, like the DC/DC switching regulator.

C Tools used during the thesis

The list below gives the reader an overview of notable tools used during this master's thesis.

- Code Composer Studio with the Texas Instruments MSP430 C/C++ compiler - An integrated development environment provided by Texas Instruments that can be used for programming their MSP430 series microcontrollers. **Available at:** `https://www.ti.com/tool/CCSTUDIO`

- Open Source toolchain for MSP Microcontrollers - An open-source debugger and a C/C++ compiler that can be used for programming the Texas Instruments MSP430 series microcontrollers. **Available at:** `https://www.ti.com/tool/MSP430-GCC-OPENSOURCE`

- GNU Arm Embedded Toolchain - Open source GNU tools for the ARM processor. **Available at:** `https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm`

- FreeRTOS - An open-source real-time operating system for embedded systems. **Available at:** `https://www.freertos.org/`

- Altium Designer 18 - Electronics computer-aided design tool for schematics and printed circuit board design. **Available at:** `https://www.altium.com/altium-designer/`

- OpenOCD - The Open On-Chip Debugger is an open-source solution for programming, debugging and boundary scanning of embedded systems. **Available at:** `http://openocd.org/`

- Rohde & Schwarz HMS3000 Spectrum Analyser - A spectrum analyser used for the S-parameter measurements of the revision 4 prototype.

- HP4396A Network/Spectrum analyser - A HP4396A network analyser was used for testing the various filters on the revision 4 prototype.

- Rohde & Schwarz HMO1522 digital oscilloscope - The oscilloscope was used for measuring different high-speed analogue signals like clock signals, power supply ripple and data on digital communication lines.

- Saleae Logic 8 - The Saleae Logic 8 USB logic analyser was used for decoding various digital communication protocols like SPI and UART communications.

- Extech 380562 milliohm meter - The milliohm meter was used for measuring the shunt resistors present in the current sense circuitry.