UNIVERSITY OF TARTU

Faculty of Science and Technology

Institute of Technology

Jürgen Laks

# DEVELOPING HARDWARE FOR THE ESTCUBE-2 STAR TRACKER

Bachelor's Thesis (12 ECTS)

Computer Engineering curriculum

Supervisor:

Erik Ilbis, MSc

Tartu 2019

# Abstract


## Developing hardware for the ESTCube-2 star tracker

Star trackers are used on satellites for accurate attitude determination. This thesis focuses on developing and testing the hardware for the ESTCube-2 star tracker prototype. The workflow consisted of selecting suitable components, creating a schematic and a PCB layout, assembling and testing the hardware. The created hardware can take images with an image sensor and transmit the image data to a computer. Some software was written to ease the testing of the hardware. The software allows users to control power buses, update the FPGA configuration, configure the image sensor and view the captured images from a personal computer. The created hardware serves as a basis for future software development and the creation of the flight model of the ESTCube-2 star tracker.



**CERCS:** T111 Imaging, image processing; T120 Systems engineering, computer technology; T170 Electronics; T320 Space technology

**Keywords:** electronics, star tracker, image sensor, FPGA



## ESTCube-2 tähejälgija riistvaraarendus

Tähejälgijaid kasutatakse satelliitide täpse asendi määramiseks. Käesolev bakalaureusetöö kirjeldab ESTCube-2 tähejälgija prototüübi riistvaraarendust ning selle testimist. Töö käigus valiti sobivad komponendid, koostati elektriskeem ja trükkplaadi disain, koostati ja testiti tähejälgija riistvara. Loodud seade on võimeline tegema pildianduriga pilte ja väljastama pildiinfot arvutisse. Riistvara testimiseks loodud tarkvara laseb arvutist lülitada tähejälgijal sisse/välja toitepingeid, uuendada FPGA konfiguratsiooni, konfigureerida pildiandurit ning laseb vaadata pildianduri tehtud pilte. Loodud riistvara on aluseks tulevasele tarkvaraarendusele ja tähejälgija lennumudeli valmistamisele.



**CERCS:** T111 Pilditehnika; T120 Süsteemitehnoloogia, arvutitehnoloogia; T170 Elektroonika; T320 Kosmosetehnoloogia

**Keywords:** elektroonika, tähejälgija, pildiandur, FPGA

# Contents

# List of figures



# List of tables

# Acronyms and abbreviations

CCD – Charge Coupled Device

CMOS – Complementary Metal-Oxide-Semiconductor

OBCS – On Board Computer Subsystem

EPS – Electrical Power Subsystem

I2C – Inter-Integrated Circuit

UART – Universal Asynchronous Receiver-Transmitter

SPI – Serial Peripheral Interface

FPGA – Field Programmable Gate Array

BGA – Ball Grid Array

QFP – Quad Flat Package

EQFP – Plastic Enhanced Quad Flat Pack

EPCS – Enhanced Programmable Configuration with SPI

FITS – Flexible Image Transport System

FRAM – Ferroelectric Random-Access Memory

SRAM – Static Random-Access Memory

SDRAM – Synchronous Dynamic Random-Access Memory

# 1 Introduction

The task of determining the attitude of a satellite is crucial for any space mission. While gyroscopes can be used for getting relative attitude data, they are not good for determining the absolute determination of a spacecraft. Star trackers, also known as star sensors, are high-accuracy 3-axis attitude determination sensors which are intended to be the most accurate attitude determination systems on a satellite [1] [2]. Star trackers work by capturing an image of stars in the sky using an image sensor and try to match the stars on the captured image against a database of stars [2] [3]. Once a match is found, the attitude of a spacecraft can be determined with a high accuracy. This paper focuses on the development and testing of the hardware of the star tracker, which will be used on ESTCube-2.

ESTCube-2 is a nanosatellite developed mostly by the students of University of Tartu and students that join the ESTCube program from all over the world. The satellite builds on the successful mission of ESTCube-1 [4]. ESTCube-2 is a 3-unit cube satellite which aims to further test Coulomb drag propulsion [5]. The satellite is also designed to be able to test an Earth observation payload and a laser communication payload. These payloads are supported by high speed communications and a cold gas thruster payload. [6] The Earth observation cameras are modified versions of ESEO cameras [7]. Pointing the satellite accurately is required for taking images of the Earth and this is the main reason ESTCube-2 needs a star tracker.

# 2 Prior work

Historically star trackers have been used in space missions for over 30 years. ASTROS, a star tracker made in 1985, weighed 41 kg and had a power consumption of 43 W. [8] Another star tracker, HDOS'HD-1003 created by Goodrich in 1996 was a major leap in progress because it only weighed approximately 3.2 kg [9].

Today, star trackers have shrunk in size and weight and use less power. The commercially available NST-3 Nano Star tracker, for example, weighs under 165 grams and has a significantly lower power consumption [10].

A star tracker has been developed in the KTH Royal Institute of Technology. This star tracker has an FPGA that reads and processes image data. The FPGA then outputs the location and intensities of each star to the main processor. [11]

Commercially available star trackers include the VST-41M star tracker which is developed by VECTRONIC Aerospace. It has a field of view of 14x14 degrees, a resolution of 512x512 pixels and has an update rate of 4 Hz. Its maximum power consumption is rated at 2.5 W. [12]

The successor of the aforementioned star tracker is the VST-68M star tracker. Its peak power consumption of 3 W is slightly higher than its predecessor, but its update rate is higher (5 Hz) and the sensor's resolution has increased to 1024x1024 pixels. While the field of view of the star tracker has stayed the same (14x14 degrees), the higher sensor resolution guarantees at least 16 visible stars independent of the current attitude as opposed to the 10 visible stars of its predecessor. [12] [13]

Another commercially available star tracker is the MAI-SS Space Sextant. Its peak power consumption is approximately 2.3 W and the average power consumption is approximately 1.5 W. Its update rate is 4 Hz and the maximum tracking rate is over 2 °/sec. [14]

One more commercially available star tracker is the KU Leuven star tracker. It has an update rate of 10 Hz and a nominal power consumption of under 1 W. [15]

The ST200 developed by Hyperion Technologies is one of the smallest star trackers made. It weighs only 42 grams. Its update rate is 5 Hz and its nominal power consumption is 0.65 W. [16]

# 3 The hardware

## 3.1 Requirements for the hardware

The hardware created for the ESTCube-2 star tracker must meet the following requirements:

- the image sensor must have a resolution of at least 1024x1024
- the hardware must have at least 80 Mbits of storage space for image data and the database of stars (assuming three uncompressed pictures at 1024x1024 pixels each + 32 Mbits for the database of stars)
- the hardware must have at least 3.6 Mbits of storage space for the FPGA's configuration (assuming the worst case – 35 % configuration compression ratio) [17]
- the FPGA configuration memory must be directly writable from the MCU
- the hardware must have one or more temperature sensors placed near the processors and memory devices
- the hardware must have a data processing device capable of processing the image sensor's data
- the hardware must include a microcontroller capable of
    - handling communication with other subsystems via UART
    - communicating with the FPGA via SPI
    - configuring the CMOS sensor via I2C
    - reading analog voltages with a built-in ADC
- the hardware must include a USB interface to all UARTs for debugging
- the hardware must include some LEDs for debugging that can be manually disconnected
- the hardware must have CMOS sensor headers for different sensor modules
- the PCB must have multiple test points and headers for programming processors
- it must be possible to power the hardware from a single USB cable
- all systems must be powered from 3.3 V input
- power for memory devices, processors and sensors must be switchable from the MCU

## 3.2 General architecture of the proposed star tracker hardware

Each star tracker needs an imaging sensor which is a CMOS sensor that takes images of the sky. Its output is read by an FPGA (Field Programmable Gate Array) that processes the image data

stream and extracts the star information from the image. The extracted star information consisting of star positions and sizes is then saved to an SDRAM (Synchronous Dynamic Random Access Memory). The FPGA is used because of its capability of doing parallel database lookups at high speeds. The database of stars, from which matches to detected stars are searched from, is stored in NOR FLASH memory. An additional FRAM (Ferromagnetic Random Access Memory) is used for storing the configuration for the FPGA, which is needed to boot the FPGA. After the FPGA has finished processing, it sends the computation result (the determined attitude of the spacecraft) to an STM microcontroller that in turn can forward the data to the OBCS (On Board Computer System). The entire data flow is graphically represented in Figure 1.

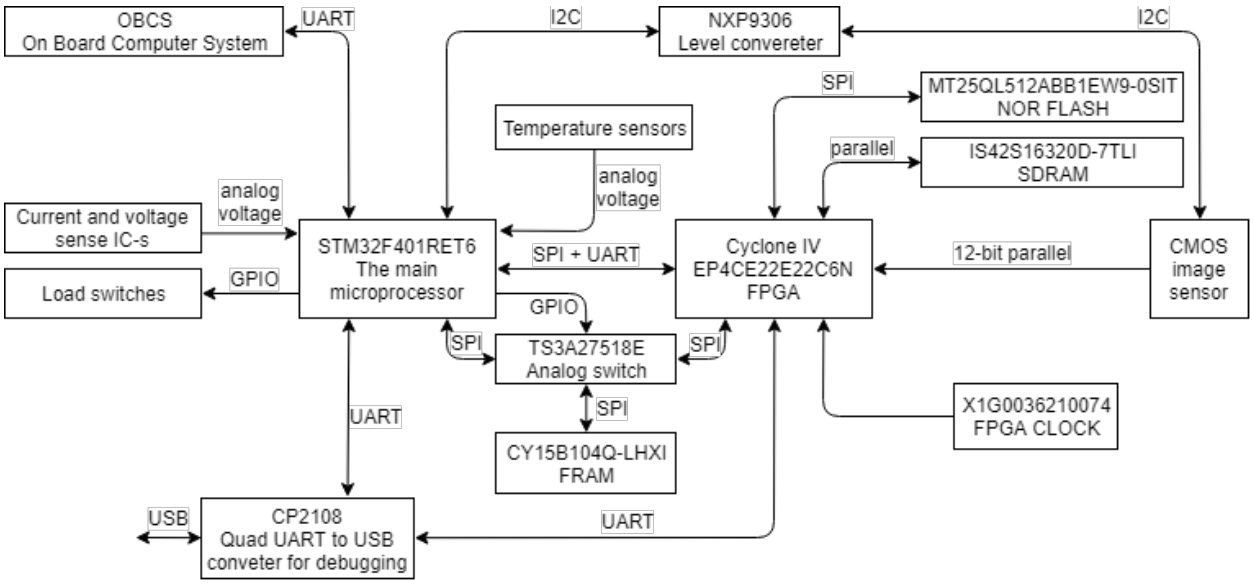


Figure 1 Data flow between components

### 3.2.1 The CMOS sensor

The CMOS sensor used in the star tracker is MT9P031I12STM manufactured by ON Semiconductor. It is a monochrome sensor that has a resolution of 2592x1944 pixels at 14 frames per second or 640x480 pixels at up to 53 frames per second. It has a maximum data rate of 96 Mp/s at a clock rate of 96 MHz. [18]

This image sensor is responsible for taking pictures of stars in the sky and clocking the image data out to the FPGA over a 12-bit parallel bus. It is also connected to the microcontroller over an I2C bus for configuration.

A CMOS image sensor was chosen as opposed to a CCD image sensor, because CMOS sensors have a lower power consumption and a higher radiation tolerance, both of which are essential to space applications. [19]

The CMOS sensor used for the star tracker does not feature a true global shutter. It instead features a mode called “global reset release“ where the exposure of all pixels is started at the same time. The amount of light each pixel is exposed to is dependent of the time between the start of the exposure and the time the value is read from the pixel. Since all of the pixels have to be read one at a time, the first pixels will have lower exposure and the last pixels read from the sensor will have higher exposure. [18] This effect can be minimized by either making the image resolution smaller (i.e. reading in less pixels), raising the data clock speed (so the process of reading data in would take less time) or by making the exposure time longer (so that the process of reading the data in would take marginally less time than exposure). The clock rate used in the star tracker is 77.625MHz. At this clock speed it would take approximately 13.5 milliseconds to read out a 1024x1024 image. For comparison the expected exposure time needed is over 100 milliseconds.

The MT9P031I12STM was chosen mainly because the ESTCube-2 team has used this image sensor for ESEO cameras [7]. Because of this, the ESTCube team had a better overview of the steps that must be taken in order to get the image sensor working. The team also had test software and development boards at hand that eased the development of the star tracker considerably. Since this sensor has been used in other star trackers as well, it was a safer choice than a camera with no space heritage as it is proven to work in space applications [11].

### 3.2.2 The FPGA and memory devices

The FPGA used in the star tracker is Cyclone IV EP4CE22 manufactured by Intel (previously Altera). Its main purpose is to read the image from the CMOS sensor, do image processing to extract the locations of the stars from the image and finally match the detected stars against the database of known stars.

The main FPGA selection criteria were a large number of logic elements and a non-BGA (Ball Grid Array) package. The non-BGA package was needed to make soldering the PCB easier. The solder joints of integrated circuits (ICs) in QFP (Quad Flat Package) packages are also more robust than the ones of ICs in BGA packages. Checking BGA soldering would require X-ray imaging, for which we did not have the required machinery available. The chosen 144-pin EQFP package

was relatively easy to solder and the soldering inspection could be done using a standard microscope.

The number of logic elements needed for implementing the star detection and matching algorithms was not known, since the algorithms were not developed nor synthesized at the time of the FPGA selection. Instead we first chose the suitable package types and then filtered the remaining FPGAs by the number of logic elements they contained. The chosen EP4CE22 has 22320 logic elements [20].

Since the FPGA does not contain any non-volatile memory, it must be accompanied by a device called Enhanced Programmable Configuration with SPI (EPCS) that stores the configuration for the FPGA [21]. The EPCS device is a flash memory device that can store configuration data that is used for configuring the FPGA after it is powered on [21].

The problem with Altera's EPCS devices is that these are essentially FLASH memory devices [21], which are prone to corruption in space. A better option would be to use FRAM based memory, which is much more resistant to radiation [22]. The chosen FRAM device was the CY15B104Q, which has a similar instruction set to EPCS devices [21] [23].

The FPGA has two additional memory devices connected to it. IS42S16320D-7TLI is an SDRAM chip with a memory capacity of 512 Mb that is used when fast storage is required [24]. This will be needed for downloading the image from the sensor, for example, as FLASH storage is not fast enough for the high data rate coming from the sensor. An additional FLASH chip (MT25QL256ABA1EW9-0SIT) with a capacity of 256 Mb is used for long-term storage of data such as dark frames and the database of known stars. [25]

### 3.2.3 The microcontroller

The STM32F401RET6 is the central part of the star tracker. It is a microcontroller that handles power sequencing, sensor configuration, FPGA configuration upgrades, temperature sensing and communications with OBCS.

This microcontroller is a 32-bit ARM Cortex CPU. It has 512 Kbytes of flash memory and 96 Kbytes of SRAM. It has 16 ADC channels, each of which has a resolution of 12 bits. The operating voltage of the microcontroller is between 1.7 and 3.6 volts. [26]

This microcontroller is connected to OBCS over UART, to the FPGA over SPI, to the FRAM over SPI and to the image sensor over I2C interface. The temperature sensors are connected directly to the analog inputs of the STM microcontroller.

### 3.2.4 The NXP9306 level converter

The logic level for the image sensor is 2.8 volts and the logic level of the MCU is 3.3 volts. A level converter is needed between them to translate the voltages to appropriate levels. The NXP9306 was chosen to translate the I2C interface voltage levels as it was successfully used in ESTCube-1 [27]. It also has an enable pin that is controlled by the MCU.

### 3.2.5 The SM72480SDE temperature sensors

A SM72480SDE temperature sensor is close to the FPGA and SDRAM and its task is to measure their temperatures. It is a temperature sensor that outputs an analog voltage dependent on the temperature. The choice to use analog temperature sensors came from the design of ESTCube-1, where I2C temperature sensors were used. The sensors on ESTCube-1 would sometimes fail and would therefore block the whole I2C bus [4].

The CMOS sensor module that was developed for ESEO and is now used for developing the star tracker has an integrated SM72408SDE temperature sensor. This sensor is connected to the main MCU in order to sense the temperature of the CMOS sensor.

### 3.2.6 The ADR3425 voltage reference

Voltage sensing, current sensing and temperature sensing is all done by measuring analog voltage levels. A high accuracy voltage reference is needed in order to take accurate measurements. The ADR3425 is a 2.5-volt high accuracy voltage reference that is connected to the Vref pin of the MCU.

## 3.3 Power supply

The supply voltage coming from EPS (Electrical Power System) is 3.3 volts. For normal operation the FPGA needs additional 1.2 V (at approximately 1.5 A) and 2.5 V (at approximately 100 mA) rails and the image sensor needs additional 1.8 V (under 300 mA) and 2.8 V rails (under 150 mA). This is achieved using multiple switching buck-converters and LDOs (Low Dropout Voltage Regulators). As the system also needs to be powered from USB while testing, it needs a buck converter that creates 3.3 V from the 5 V USB input. Most of these voltage regulators can be turned

on and off by load switches that are placed before the voltage regulators. A block-diagram of the power architecture can be seen on Figure 2.

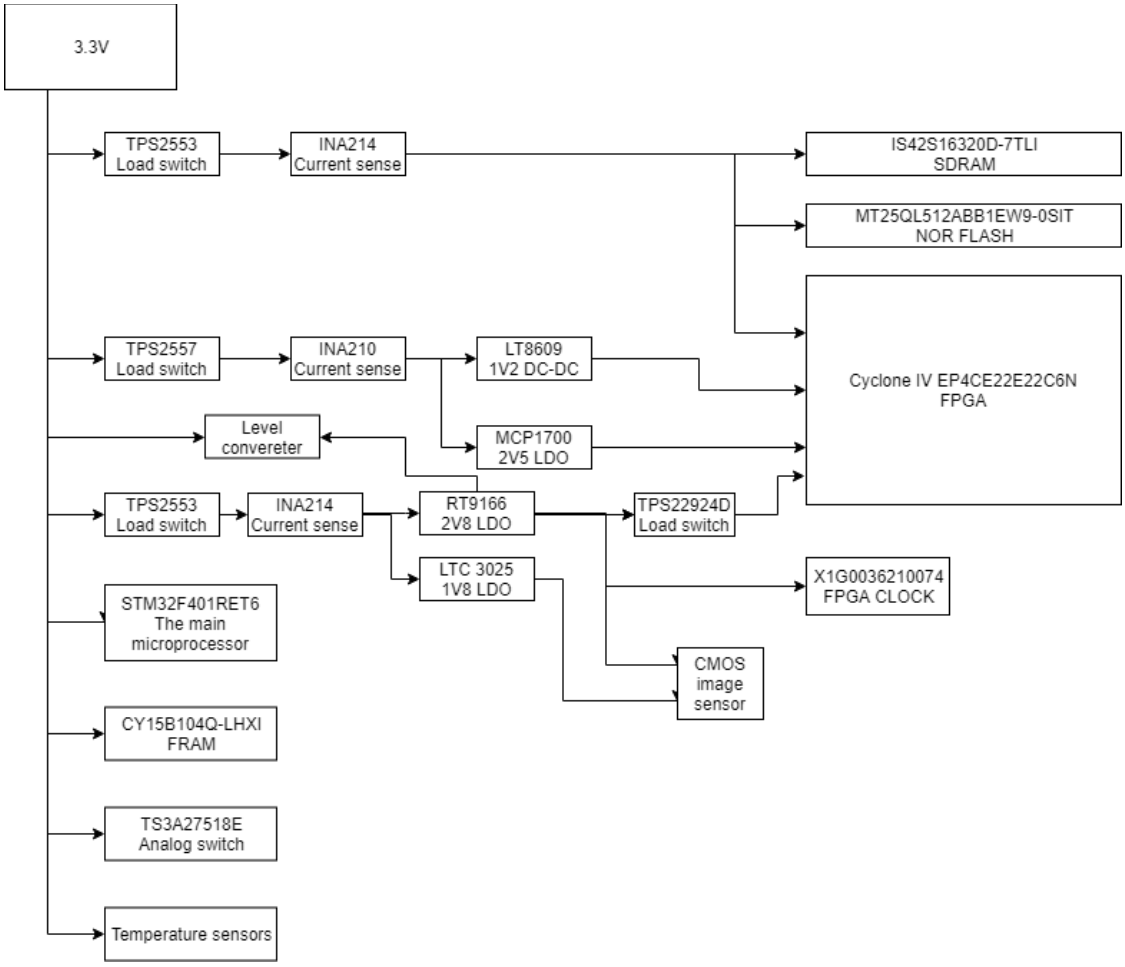


Figure 2 The power architecture of the start tracker of ESTCube-2

## 3.4 Additions for ease of debugging

The created star tracker hardware is just the first revision. This means that a lot of emphasis was put on making the process of debugging the hardware as simple as possible.

### 3.4.1 CP2108 quad USB to UART converter

Having a UART interface is convenient when developing software. It allows the developer to print debug output to a device that can print it out. To make it possible to view the debug logs on a PC, a USB to UART converter is needed.

This USB to UART converter has 4 UART peripherals. The converter creates four virtual COM ports once connected to a computer, one corresponding to each of its UART peripherals. One of these UART peripherals is connected to the MCU and one of them is connected to the FPGA.

Four LEDs were connected to the USB to UART converter to indicate if information is being sent or received by the FPGA or MCU.

### 3.4.2 Multiple CMOS sensor connectors

The star tracker PCB has two connectors for connecting the CMOS sensors: a flat-flex connector and a pin header.

The flight model of the star tracker will only have a flat-flex connector. It takes substantially less room on the PCB than its pin header counterpart. This flat-flex connector is needed on the star tracker prototype for testing the CMOS sensor modules that will be used in the flight model of the star tracker.

ESEO (European Student Earth Orbiter) is a satellite that uses the same CMOS sensors onboard as the star tracker of ESTCube-2. Their prototype CMOS sensor module connector was a 2 by 15 pin header. The pin header connector for the star tracker prototype was chosen so that it would be possible to use ESEO's camera modules, that are tested on ESEO's hardware, with the star tracker. The pin headers also made it much easier to debug with a logic analyzer as connecting probes to pin headers is much simpler than to a flat-flex connector.

### 3.4.3 Connectors for firmware updates

The STM microcontroller and the FPGA have their own JTAG connectors for debugging purposes. The FPGA also has a header for Altera USB-Blaster. The purpose of this header was to upload FPGA configurations to the FPGA during debugging. As this could also be done using the JTAG connector, the USB-Blaster header was never used during development.

### 3.4.4 Other additions

Four LEDs were added to the star tracker for debugging purposes. Three of them were connected to the MCU and one to the FPGA.

If testing the sensor in a dark environment was needed, the LEDs should all be off to decrease light pollution in the test environment. While a software solution is possible, a hardware solution is

much easier to use while testing the hardware. A pin header with a jumper was added next to the LEDs that would allow disconnecting the LEDs.

Two extra 3-pin pin headers with jumpers were also added close to the bus switch. The first one selected the bus switch's control input source: either the MCU or the second pin header. The second pin header allows selecting whether the control signal of the bus switch is connected to power rail or GND. This was meant for situations where the FPGA needs to communicate with the FRAM while the software for the STM did not exist yet.

Having test pads around the PCB makes the process of debugging PCBs easier and faster. The star tracker PCB has test pads connected to each of the power buses, load switch control signals, CMOS sensor control signals and ADC input signals. There are 5 additional GND test pads.

## 3.5 PCB design

The star tracker prototype was designed in Altium Designer onto a 6-layer PCB. The PCB was later manufactured by Brandner.

### 3.5.1 Layers

The star tracker prototype was designed on a 6-layer PCB. Since we wanted one layer to be dedicated for ground connections and one layer dedicated for power traces, a 4-layer PCB wouldn't have been enough. The layers can be seen in Appendix 1 – Copper layers of the PCB.

Each of the layers was assigned a specific task. Each layer has a GND polygon pour over the full board area. The top (layer #1) and bottom (layer #6) layers were used for connecting the surface mount components that were near each other. Most of the traces were routed on these layers in order to reduce the number of vias used. Layers #3 and #5 were used to route longer traces between components that were far from each other. These layers were mainly used for routing traces to connectors placed at the sides of the PCB. These layers were also used to route shorter tracks in places where the top and bottom layers could not be used. Layer #2 was used solely for low-impedance ground connections. Layer #4 was used for routing power supply traces.

### 3.5.2 Power traces

Power traces were routed mainly on layer #4. A star-shaped design was kept in mind when routing the power traces. A small section of the star tracker PCB with only power traces visible can be

seen on Figure 3 where the violet trace comes from the power source on the right hand side of the image and then splits on the green layer into multiple power traces at a single point.

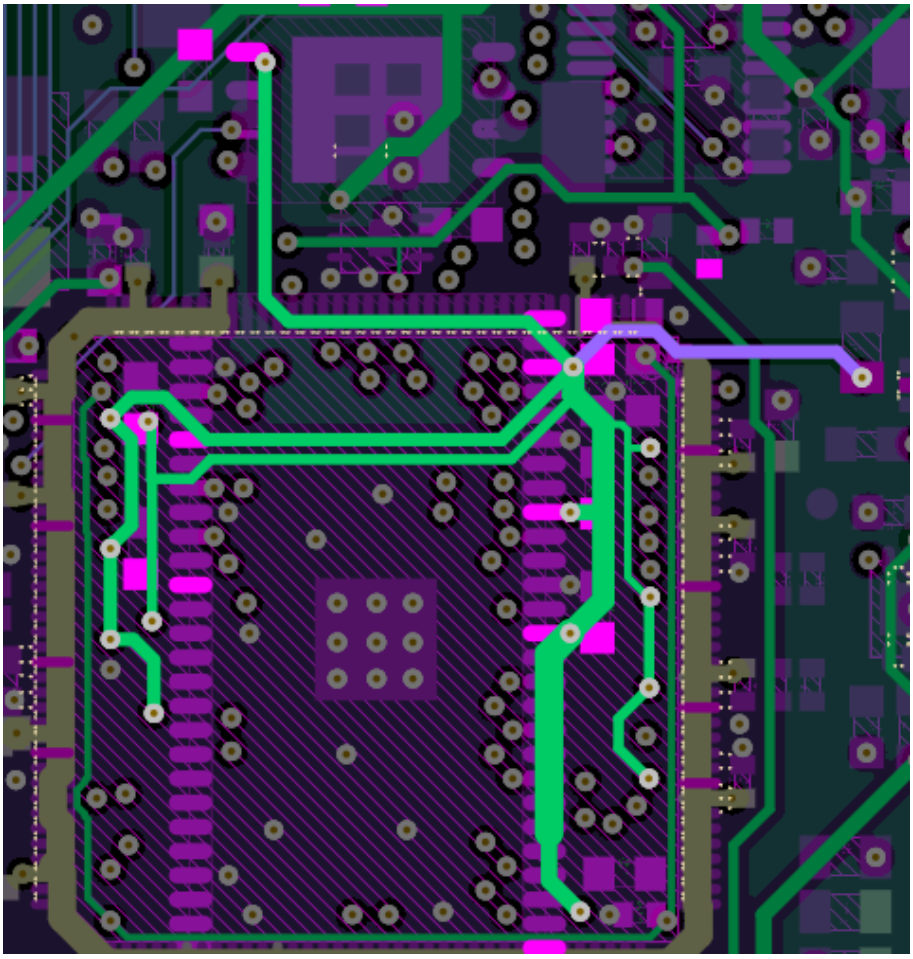

Figure 3 Star-shaped power trace routing

### 3.6 Assembled hardware

Two identical star tracker prototypes were assembled so that multiple embedded software developers could work simultaneously. One of the assembled prototypes can be seen on Figure 4. The assembled prototypes have all components soldered to the PCB (except for some connectors that were not used during testing) and have the camera (image sensor fitted with the lens) installed.

The connector number 1 in Figure 4 is used for connecting the Altera USB Blaster which allows programming the FPGA. The connector number 2 is used for programming the STM microcontroller. Custom cables were made to connect the prototype PCBs to the SEGGER JLink debug probe that was used for programming and debugging the microcontroller. Connector

number 3 is a flat-flex connector that connects the external image sensor module to the star tracker PCB. Connector number 4 serves the exact same purpose as connector number 3 but works with image sensor modules that have pin headers for connectors. Connector number 5 is the JTAG interface to the FPGA which was used for programming the FPGA. Connector number 6 is for connecting the star tracker to the OBCS. This connector carries data over a UART interface. Connector number 7 is for connecting the star tracker to the Electrical Power Subsystem (EPS). The USB connector number 8 is connected to the quad-UART integrated circuit that is used for transmitting data from the MCU and FPGA to a computer.

The switch at the bottom is the main power switch. It switches power between the EPS connector (when set to the right) and the USB connector (when set to the left). The switch on the left controls the BOOT0 pin of the STM. Moving the switch up connects BOOT0 to GND, moving the switch down connects BOOT0 to 3.3 volts.

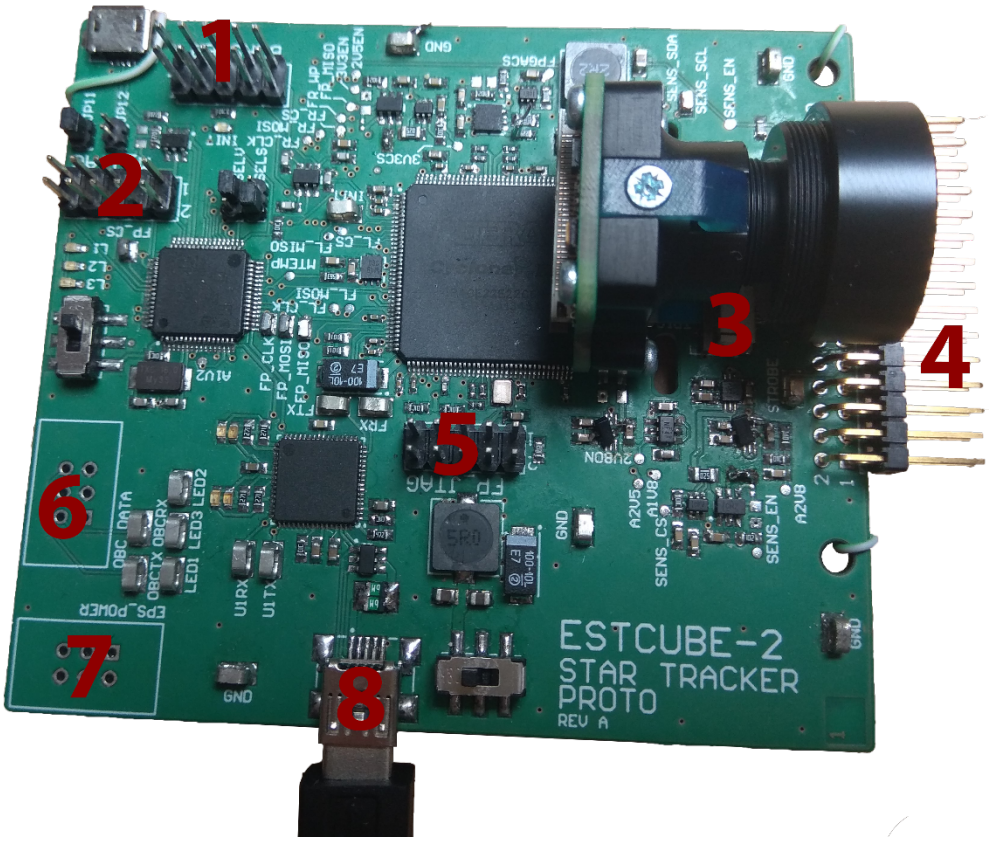


Figure 4 The assembled star tracker prototype

# 4 Software

## 4.1 MCU software requirements

The software of the microcontroller of the ESTCube-2 star tracker must be able to

- control load switches to power devices up/down
- start image capture using the CMOS image sensor
- send the original image to OBC via UART
- configure the image sensor from OBC via UART
- overwrite the FPGA configuration with the one sent from OBC
- telemetry data such as voltages/currents/temperatures must be sent to OBC
- must be able to recover if a thread should go into an infinite loop

## 4.2 Peripheral drivers

In order to control all the devices connected to the microcontroller, the software of the microcontroller of the ESTCube-2 star tracker must have peripheral drivers for

- the CMOS sensor (over I2C)
- ADC (for measuring voltages, currents, temperatures)
- FRAM (over SPI)
- communicating with OBCS (over UART)
- communicating with the FPGA (over SPI)

Most of the peripheral drivers are implemented in the ESTCube Hardware Abstraction Layer library called ECHAL, which is common for all subsystems of ESTCube-2.

As the FRAM driver is not implemented in ECHAL, it was implemented separately. It relies on the SPI driver of ECHAL. Functions for sequential writing were implemented along with reading/writing of memory blocks.

The driver for the CMOS sensor was also not implemented in the ECHAL. Since the same image sensor was used in ESEO cameras [7], the driver was ported from ESEO's software to star tracker software. Porting the library from ESEO's codebase means we can reuse code that has already been properly tested and is reliable therefore saving development time. The ported driver is mostly based on ESEO's software, but now relies on ECHAL for I2C communication.

## 4.3 Main control logic

The main program runs in two (or more if created during runtime) parallel threads. The first thread contains the main state machine loop. This loop handles initial data received from UART. If a state change should be done for a task indicated by the data coming from UART, this thread creates a new thread that handles the task. The task thread then executes the task, resets the state machine's state to the default state and terminates itself. New threads will not be created unless the thread handling the current task is terminated. This way it is not possible to create too many threads at the same time.

The second thread acts as a watchdog timer. It periodically checks if any created task threads are unresponsive. In case a task should go into an infinite loop, the watchdog thread terminates it and resets the state machine's state to default so that the main loop would be able to handle state changes.

## 4.4 Communication protocols

The communication with OBC works over UART. In the default state the first byte sent through UART specifies the new state to go in. The protocol is described in Table 1 where bytes marked brown are transmitted out of the star tracker, bytes marked blue are bytes sent to the star tracker.

| First byte | Meaning | Data | Comments |
|---|---|---|---|
| 0x00 | Get board ID | 0xAB | Always returns 0xAB. |
| 0x01 | Byte echo | [byte] [byte] | The board echoes the byte written to it. |
| 0x02 | FRAM overwrite | 0xAC [size]x4 0xAC ([data]x32 0xAC)xN 0xED | Size is the configuration size in bytes and data is the data that will be written to the FRAM. After each 32 data bytes are sent, the star tracker will send an acknowledge byte (0xAC) and 0xED to indicate the end of a successful writing operation. N can be calculated by dividing the total size of the data in bytes by 32 and rounding the result up. The last block of data bytes can be less than 32 bytes if the number of configuration bytes is not exactly divisible by 32. |

| | | | |
|---|---|---|---|
| 0x03 | Read CMOS sensor configuration | [data]x88 | After receiving 0x03, the star tracker returns 88 bytes describing the current configuration of the CMOS sensor. The meaning of each byte (or bit in some cases) can be read from the isens_params_t struct in the CMOS sensor driver software. |
| 0x04 | Write CMOS sensor configuration | 0x58 0xAC [data]x88 0xAC 0xED | 0x58 represents the number of configuration bytes expected by the star tracker. Data consists of 88 bytes describing the current configuration of the CMOS sensor. The meaning of each byte (or bit in some cases) can be read from the isens_params_t struct in the CMOS sensor driver software. After the configuration is received an acknowledge byte is sent. Then the configuration is sent to the image sensor. A 0xED byte is sent after successfully configurating the image sensor. |
| 0x05 | Power on | 0xAC | Turns power supplies for FPGA and the image sensor on |
| 0x06 | Power off | 0xAC | Turns power supplies for FPGA and the image sensor off |

Table 1 Communication protocol used for controlling the star tracker prototype

## 4.5 PC control software

ESTCube-2 Star Tracker Configuration Utility is a PC application written in Java by the author of this thesis. Its purpose is to ease the testing of the star tracker by giving the user a graphical user interface for controlling the electronics.

### 4.5.1 Port scanner

Since the hardware of star tracker implements a quad-UART to USB chip for interfacing with the computer, the user can get confused quickly as to which serial port is connected to which chip on the start tracker proto board. This port scanner (as seen on Figure 5) maps ports to the devices

(STM or FPGA) and configures the serial port fields for other tabs (FPGA configuration updater, CMOS sensor configuration updater etc).

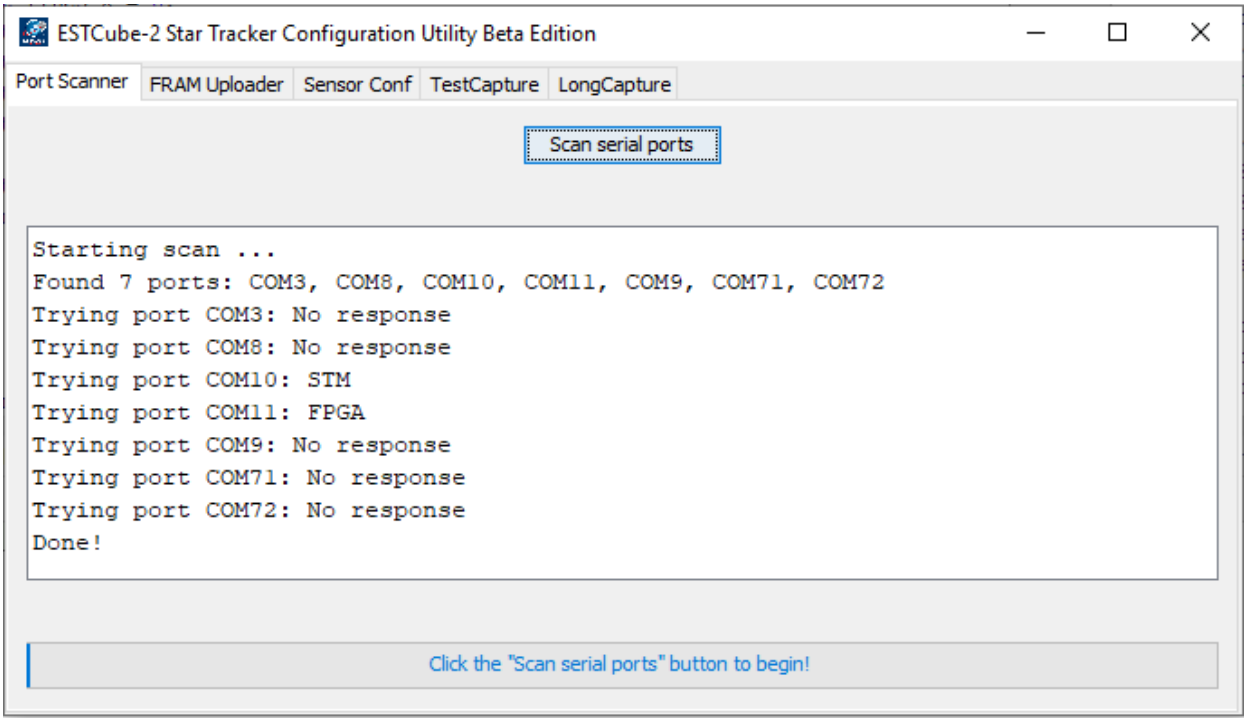


Figure 5 ESTCube-2 Star Tracker Configuration Utility port scanner tab after scanning

### 4.5.2 FPGA configuration updater

This tab (as seen from Figure 6) can be used to upload a bitstream of the FPGA configuration to the FRAM. This is the configuration that the FPGA will boot with. While this is not necessarily needed for the testing or the algorithm development stage, it is certainly needed for storing the configuration that the FPGA will boot to while in flight.

In order to upload the configuration to the FRAM a .rpd file has to be generated. This process is outlined in Appendix 2 – Binary .rpd file generation guide. Once the .rpd file has been generated, the user can click the “Browse” button and browse to the generated .rpd file. After clicking the “Send bitstream” button, the upload process begins.

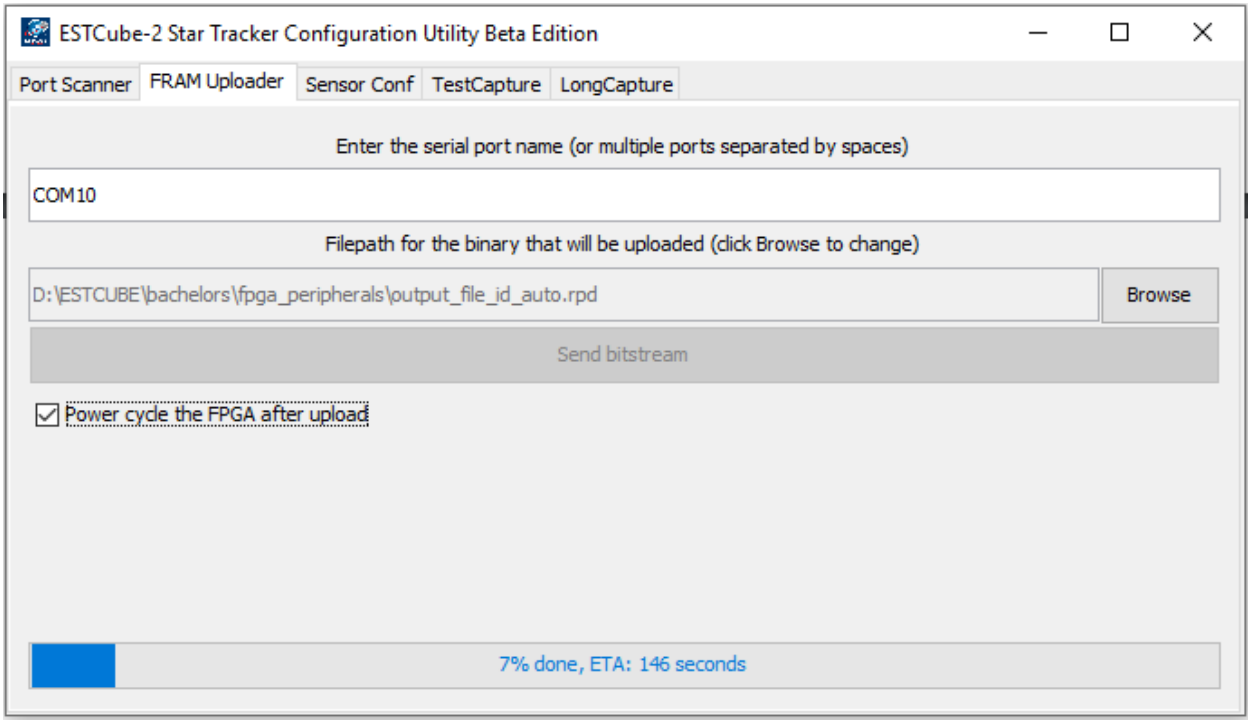


Figure 6 ESTCube-2 Star Tracker Configuration Utility FRAM updater tab

### 4.5.3 CMOS sensor configuration updater

The CMOS sensor configuration tab (as can be seen from Figure 7) is used for configuring internal registers of the MT9P031 image sensor. By clicking on the "Read configuration" button, the current state of the registers is displayed in the text area below the button row. These values can be changed either manually or with the text fields below the text area. When changing the parameters (such as image width) from the text fields, the text area containing the full sensor configuration is changed automatically. By clicking the "Write configuration" button, the configuration in the text area is sent to the star tracker and the sensor is configured. The power for the image sensor can be controlled using the "Power on" and "Power off" buttons.

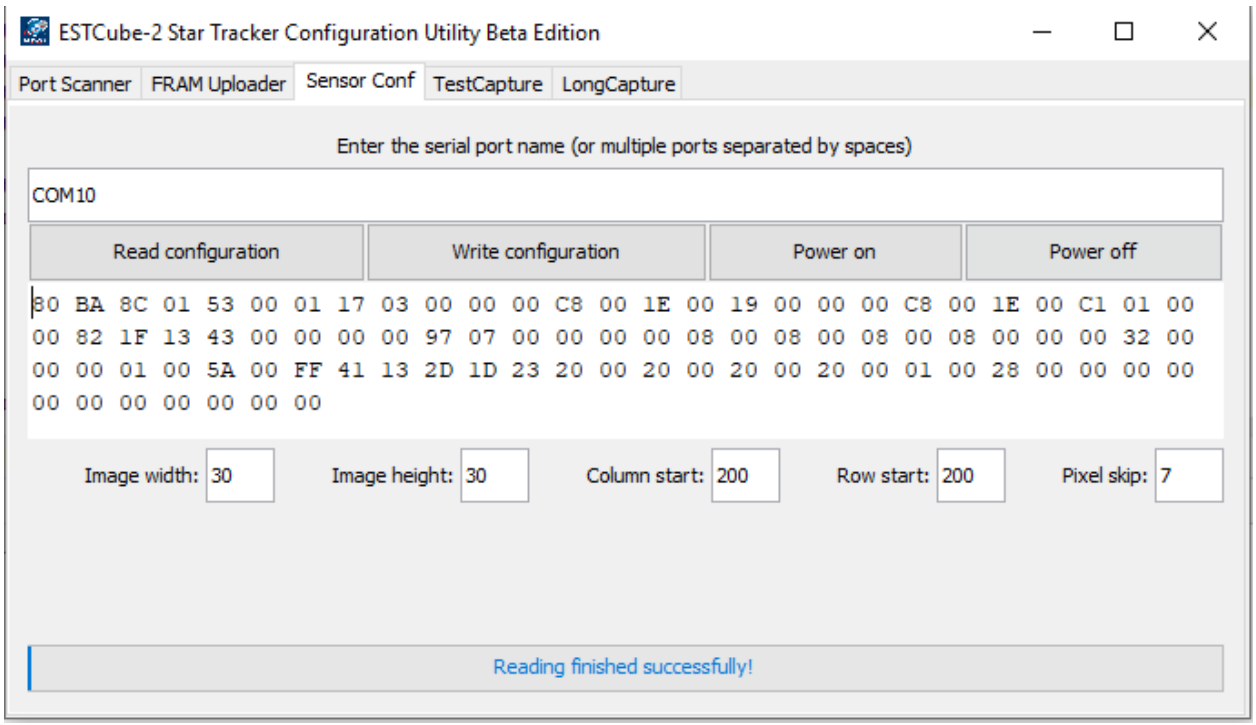


Figure 7 ESTCube-2 Star Tracker Configuration Utility image sensor configuration tab

# 5 Testing

Testing the hardware is important in order to find the faults in the design. Hardware tests also serve as a good basis to software that will be written for the hardware. More software had to be written to test the image sensor than any other sensor or component. Moreover, the testing of the image sensor tested almost every other part of the hardware as well, since the test needed the microcontroller and FPGA to be in a working condition, which in turn, relied on the power distribution circuitry being in a working condition. From now on, the image sensor fitted with the lens is referred to as the camera.

## 5.1 Configuring the image sensor

The sensor is configured by the main microcontroller over I2C. The configuration of the sensor includes the image size, image location on the sensor, exposure settings and many more parameters. To pass the configuration test, the software has to configure the sensor and then read the configuration back for verification.

Initially the test failed approximately 50 % of the times. The sensor would not acknowledge the data sent over the I2C bus and the test would fail. This was caused by the current-limiting load switch, which had its current limit set too low at around 100 mA. Because of this the sensor would turn off during its peak power consumption. Raising the current limit to 500 mA seemed to have fixed the issue.

## 5.2 Reading the image from the camera

Reading the image from the camera proved to be a very difficult task because most parts of the system were difficult to test separately. The focal distance of the camera used for testing was not known, which made it difficult to focus the camera on fixed patterns against which to check the image output. Since the image sensor clocks data out at speeds multiple times higher than supported by UART, reading the image data was not as easy as transferring the data to UART. Since implementing an SDRAM driver for the FPGA is beyond the scope of this thesis, the data had to be buffered inside the logic elements of the FPGA. This created a storage space problem as only an image of 16x16 pixels could be fitted inside the FPGA. The task of reading a full image had to be broken down into smaller subtasks to verify that each part of the system was working.

### 5.2.1 Reading the first pixel

A goal was set to read a pixel from the image sensor in order to verify that the sensor is working correctly. Then the pixel value was observed while repeatedly shining bright light onto the sensor and covering the sensor alternately. The test was considered passed once the value read from the image sensor increased and decreased according to the amount of light shining at the image sensor.

The software for this test was written in VHDL. The written FPGA configuration implements a UART that listens for two commands. Once the first command is received, the FPGA pulses the TRIGGER pin of the image sensor and starts monitoring the FRAME_VALID and LINE_VALID signals of the image sensor. Once both of the aforementioned signals are high, the value on the 12-bit parallel bus of the image sensor is saved to the FPGA's memory which was constructed of logic elements. Once the second command is received, the FPGA would write the stored 12-bit value to UART as two bytes. A serial terminal emulator was used to send the control commands and receive the sensor values.

### 5.2.2 Reading a matrix of pixels

The next step to reading a full image was to read an array of pixels. For this the image sensor was configured to take images of size 32x32 pixels. The FPGA configuration for the previous test was modified so that instead of storing a single pixel, the values are stored in an array. Also the part of the configuration that previously transmitted the single sensor value to the UART was modified to dump the entire contents of the sensor value array to UART.

Increasing the number of elements of the sensor value array made the compilation time significantly longer. Testing showed that having array lengths of more than 16x16 pixels made that compilation time approximately 30 minutes long, after which it would fail with the reason of not having enough space available in the FPGA. This is the reason why the next steps are based on taking images of 16x16 pixels in size.

As it is difficult to visualize 256 pixels in a terminal emulator, the previously mentioned ESTCube-2 Star Tracker Configuration Utility was updated to include the "TestCapture" tab. In this tab the software automatically opened the correct serial port, sent the trigger command followed by the data dump command, read the image data in, processed it and finally saved it in FITS file format. The image could then be opened using FITSview or any other astronomy related image viewer.

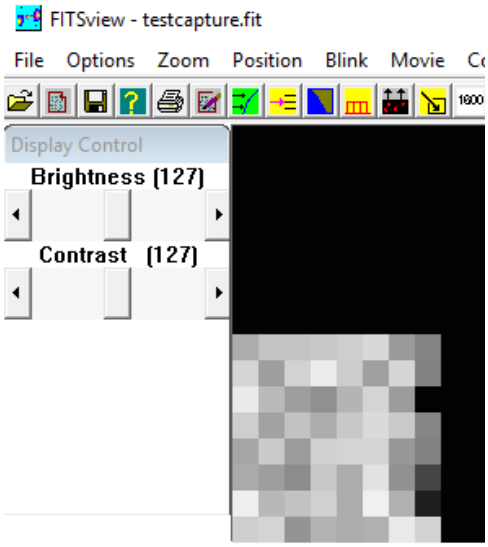


Figure 8 The first 16x16 pixels extracted from the image sensor

### 5.2.3 Reading a larger image

As can be seen from Figure 8 the problem with an image of 16x16 pixels is that it is not possible to check which part of the field of view of the camera this image is taken from, how large of an area the image is covering and if the image is in focus or not. For solving the issue multiple tactics were used.

Firstly, the pixel skip option in the sensor configuration was set to the maximum value of 7. The pixel skip functionality skips a specified number of pixels after reading the intensity of a pixel. This makes it possible to read images with a lower resolution without cropping and thus makes it more likely to get patterns visible on the image.

Another tactic for viewing the bigger picture was to take multiple 16x16 images from different locations on the sensor and stitching them together. For this the ESTCube-2 Star Tracker Configuration Utility was updated to include the "LongCapture" tab which would repeatedly reconfigure the image sensor to take an image at a new location each time and then read the image data from the sensor. Finally, the image was stitched together and shown in the same window in order to remove the time overhead of opening the image with an external image viewer such as FITSview. The results can be seen on Figure 9. It was quite difficult to stitch the images perfectly since the pixel skip parameter in the sensor affected the coordinate system in the sensor as well,

rather than just the number of skipped pixels during reading the image. A failed stitching attempt can be seen on the left image in Figure 9. The center and right pictures have the image stitching working correctly and have pixel skip values of 7 and 2 respectively.

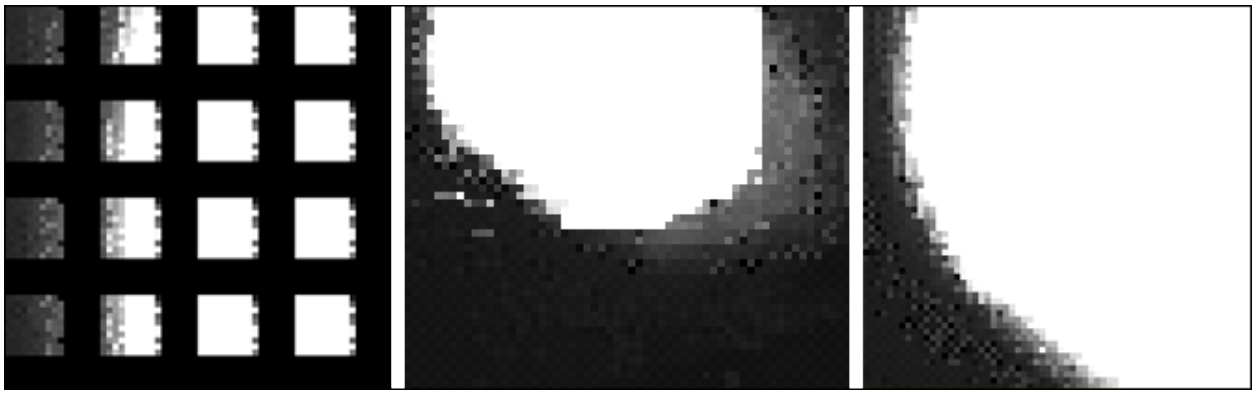

Figure 9 Larger images combined from multiple 16x16 images

In order to find which part of the camera's field of view the image was taken a test setup was created where the camera was positioned in a dark room with a bright 5 mm LED serving as a spotlight facing the camera. Multiple images were taken with the spotlight in different places. In the beginning the spotlight was placed near the camera and slowly moved away from the camera. By observing the image, it was possible to see if the spotlight went closer or farther away from the part of the field of view where the image was taken. Moving the spotlight further and back also made it possible to assess the focal distance of the camera. The images taken during this test can be seen on Figure 9.

### 5.2.4 Taking a real picture

Now that the focal distance was known, a page of text was positioned in front of the camera. Focusing on small text made it possible to precisely measure the focal distance as can be seen from the first two images in Figure 10. The image resolution was then increased by taking an 8 by 8 grid of images each 16x16 pixels in size. Since the reconfiguration of the image sensor took a relatively long time which was only made worse by the communication overhead between the PC, MCU and the FPGA, the capturing process was highly time-consuming and this dictated the limit of the image's resolution. The rightmost image in Figure 10 was taken with a pixel skip value of 1 and had a resolution of 128 by 128 pixels with 12-bit color depth in grayscale.

Figure 10 Images taken of text with different focus and pixel skip parameters

## 5.3 Testing the FPGA

Two tests needed to be done with the FPGA. First, the FPGA had to be programmable over the JTAG interface. This was needed for rapid testing of software. The FPGA also had to be able to read its configuration from the FRAM chip on powerup.

### 5.3.1 JTAG programming

Once the supply voltages for the FPGA were verified, an Altera USB Blaster was connected to the JTAG interface of the FPGA. Then Quartus II was used to generate the FGA configuration. Quartus II included a graphical user interface for programming the FPGA, which successfully uploaded the configuration to the FPGA.

### 5.3.2 Configuration from FRAM

At first there was a mismatch in data endianness in the configuration. Testing showed that the data endianness had to be converted when writing from the output file of Quartus II to FRAM.

After fixing the endianness mismatch, the FPGA still didn't boot. The reason was the missing "Read Silicon ID" opcode of the FRAM chip, since it was not a real EPCS. Fortunately, Altera's Quartus Prime software has the option to "Disable EPCS ID check" when exporting the configuration. When this option was enabled, the FPGA loaded the configuration from the FRAM as it would from an EPCS.

## 5.4 Problems and mistakes

The idea behind making prototypes is to find problems before creating the flight model of hardware. During the testing of the star tracker prototype multiple hardware problems surfaced.

The STM microcontroller wasn't programmable through the JTAG connector. It turns out that the JTAG connector's RESET pin was connected to the microcontroller's JTAG RESET pin while it should have been connected to the RESET pin of the microcontroller. This was fixed with a wire running on the board to the correct pin of the MCU.

The LEDs of the USB to UART converter IC were wired incorrectly. While the UARTs used for communication were UART1 and UART2, the LEDs were wired for UART0 and UART1. While this did not affect the usability of the device, it certainly made debugging more difficult.

The LDOs initially used were advertised as "very low drop-out" since the voltage between the Vin and Vout could be as low as 45 mV for it to work. We discovered during testing that the Vbias pin had to be 1.4 V greater than Vout. Since it was supplied from Vin, the voltage regulator didn't work as a low drop-put voltage regulator. It was then replaced by MCP1700 and RT9166 LDOs. The LTC3026 was still used for generating 1.8 V since here the drop-out would be 1.5 V, which is greater than the required 1.4 V.

A feature that was missing but was never needed during testing was STM boot pins connected to the OBC connector. This would be needed to update the STM firmware by OBC. During testing a slide-switch was used to set the state of the boot pins to required values.

Other minor mistakes consisted of switches missing silk-screen labels, unnecessarily low resistor values for pull-up resistors, wrong resistor packages and moving a larger capacitor away from the flat-flex cable for it to fit better.

## Summary

The aim of this thesis was to develop hardware for the star tracker that will be used onboard ESTCube-2. The architecture of the developed star tracker consists of a CMOS image sensor for taking images of the stars, an FPGA for extracting stars from the image and a microcontroller for controlling power buses, configuring the image sensor and FPGA and mediating data with the attitude determination and control subsystem. All components were chosen with usability in space in mind. The PCB layout was designed for a 6-layer PCB using Altium Designer software.

Special software was written in order to test the hardware. The software for the microcontroller was written in C, VHDL was used for the FPGA and Java was used for the computer software. The created computer software allows updating the FPGA configuration, configuring the image sensor and taking pictures with the image sensor.

The created hardware can control voltage buses, can measure voltage and current of most power buses, has a working FPGA that can be reconfigured through the main microcontroller, has a working image sensor that can take images and transmit the captured image data over a UART bus to a computer where specialized software converts the data to image files.

# Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli arendada vajalik riistvara ESTCube-2 tähejälgija jaoks. Loodud tähejälgija koosneb CMOS pildiandurist, mis võimaldab tähtedest pilte teha, FPGA-st, mis tegeleb pildist tähtede otsimisega, ja mikrokontrolleriga, mis kontrollib toiteliine, konfigureerib pildiandurit ja FPGA-d ning vahendab infot ADCS alamsüsteemiga. Komponentide valimisel arvestati komponentide sobilikkust kosmoses kasutamiseks. Loodud elektroonika on disainitud kuuekihilisele trükkplaadile kasutades Altium Designer tarkvara.

Riistvara testimiseks on kirjutatud vajalik tarkvara. Mikrokontrolleri tarkvara on kirjutatud C keeles, FPGA tarkvara on kirjutatud VHDL keeles ning arvutipoolne tarkvara on kirjutatud Javas. Loodud arvutipoolne tarkvara võimaldab uuendada FPGA konfiguratsiooni, uuendada pildianduri konfiguratsiooni ning võimaldab teha pildianduriga pilte.

Loodud riistvara suudab kontrollida toiteliine, mõõta toiteliinide pinget ja voolu, omab töötavat FPGA-d, mida saab läbi mikrokontrolleri ümber konfigureerida, omab töötavat pildiandurit, mis suudab teha pilte ja neid üle UART liidese arvutisse saata, kus vastav program neid kuvada ja salvestada saab.

# References


[1] M. Marin and H. Bang, "High Update Rate Star Tracker for Gyroless Spacecraft Operation," *2018 5th IEEE International Workshop on Metrology for AeroSpace,* pp. 295-299, 2018.

[2] S. Bendapudi, G. K. Kashyap, M. G. Lithin, M. Subhajit, B. KrishnamPrasad and T. H. Shashikala, "Design of video processor for multi-head star sensor," *2015 2nd International Symposium on Physics and Technology of Sensors (ISPTS),* pp. 59-62, 2015.

[3] X. Wang, X. Wei, Q. Fan, J. Li and G. Wang, "Hardware Implementation of Fast and Robust Star Centroid Extraction With Low Resource Cost," *IEEE Sensors Journal,* vol. 15, no. 9, pp. 4857-4865, 2015.

[4] A. Slavinskis, M. Pajusalu, H. Kuuste, E. Ilbis, T. Eenmäe, I. Sünter, K. Laizans, H. Ehrpais, P. Liias, E. Kulu, J. Viru, J. Kalde, U. Kvell, J. Kütt, K. Zalite, K. Kahn, S. Lätt, J. Envall, P. Toivanen, J. Polkko and P. Janhunen, "ESTCube-1 In-Orbit Experience and Lessons Learned," IEEE A&E SYSTEMS MAGAZINE, Tõravere, 2015.

[5] I. Iakubivskyi, P. Janhunen, J. Praks, V. Allik, K. Bussov, B. Clayhills, J. Dalbins, H. Ehrpais, J. Envall, S. Haslam, E. Ilbis, N. Jovanovic, E. Kilpua, J. Kivastik, J. Laks, P. Laufer and M. Merisalu, "Coulomb drag propulsion experiments of ESTCube-2 and FORESAIL-1 DRAFT," *Acta Astronautica,* 2019.

[6] H. Ehrpais, I. Sünter, E. Ilbis, J. Dalbins, I. Iakubivskyi, E. Kulu, I. Ploom, P. Janhunen, J. Kuusk, J. Šate, R. Trops and A. Slavinskis, "ESTCUBE-2 MISSION AND SATELLITE DESIGN," The 4S Symposium 2016, 2016.

[7] I. Sünter, "DUAL-CAMERA PAYLOAD for ESEO," 2016.

[8] A. R. Eisenman, C. C. Liebe and J. L. Jorgensen, "The New Generation of Autonomous Star Trackers," [Online]. Available: https://trs.jpl.nasa.gov/bitstream/handle/2014/22609/97-1120.pdf. [Accessed 28 03 2019].

[9] L. W. Cassidy, "Space qualification of HDOS' HD-1003 Star Tracker," Space Sciencecraft Control and Tracking in the New Millennium, 1996.

[10] CubeSatShop, "NST-3 Nano Star Tracker," CubeSatShop, [Online]. Available: https://www.cubesatshop.com/product/nst-3-nano-star-tracker/. [Accessed 28 03 2019].

[11] M. Lindh, "Development and Implementation of Star Tracker Electronics," KTH ROYAL INSTITUTE OF TECHNOLOGY, Stockholm, 2014.

[12] VECTRONIC Aerospace, "Star Tracker VST-41M," VECTRONIC Aerospace, [Online]. Available: https://www.vectronic-aerospace.com/space-applications/star-sensor/. [Accessed 03 05 2019].

[13] VECTRONIC Aerospace, "Star Tracker VST-68M," VECTRONIC Aerospace, [Online]. Available: https://www.vectronic-aerospace.com/space-applications/star-tracker-vst-68m/. [Accessed 03 05 2019].

[14] CubeSatShop, "MAI-SS Space Sextant," MAI, [Online]. Available: https://www.cubesatshop.com/product/mai-ss-space-sextant/. [Accessed 03 05 2019].

[15] KU Leuven, Department of Mechanical Engineering, [Online]. Available: https://www.cubesatshop.com/wp-content/uploads/2018/10/KULST-Datasheet.pdf. [Accessed 03 05 2019].

[16] Hyperion Technologies, "ST200 Product description," [Online]. Available: https://hyperiontechnologies.nl/products/st200-star-tracker/. [Accessed 03 05 2019].

[17] Intel, "Configuration and Remote System Upgrades in cyclone IV Devices," 05 2013. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyiv-51008.pdf. [Accessed 03 05 2019].

[18] ONSEMI, "MT9P031 - 1/2.5-Inch 5 Mp CMOS Digital Image Sensor," January 2017. [Online]. Available: http://www.onsemi.com/pub/Collateral/MT9P031-D.PDF. [Accessed 28 October 2018].

[19] O. Saint-Pé, M. Tulet, R. Davancens, F. Larnaudie, B. Vignon, P. Magnan, J. Farré, F. Corbière and P. Martin-Gonthier, "High performances monolithic CMOS detectors for

space applications," [Online]. Available: http://oatao.univ-toulouse.fr/316/1/MartinGonthier_316.pdf. [Accessed 28 03 2019].

[20] Altera, "Cyclone IV Device Handbook Volume 1," 03 2016. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyclone4-handbook.pdf. [Accessed 04 04 2019].

[21] Altera, "Serial Configuration (EPCS) Devices Datasheet," April 2014. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cfg/cyc_c51014.pdf. [Accessed 01 August 2018].

[22] P. M. Campbell and S. D. Wix, "Models for Total-Dose Radiation Effects in Non-Volatile Memory," 2017.

[23] Cypress, "CY15B104Q, 4-Mbit (512 K × 8) Serial (SPI) F-RAM," 3 09 2015. [Online]. Available: http://www.mouser.com/ds/2/100/CY15B104Q%204_Mbit(512K_8)Serial(SPI)_RAM_Datasheet-914618.pdf. [Accessed 07 05 2019].

[24] ISSI, "SDRAM datasheet," 05 12 2015. [Online]. Available: http://www.issi.com/WW/pdf/42-45R-S_86400D-16320D-32160D.pdf. [Accessed 21 02 2019].

[25] Micron Technology, "Micron Serial NOR Flash Memory," Micron Technology, 2014.

[26] ST Microelectronics, "STM32F401xE datasheet," ST Microelectronics.

[27] E. Ilbis, "ESTCUBE-1 ELECTRICAL POWER SYSTEM - DESIGN, IMPLEMENTATION AND TESTING," Tartu, 2013.

# Appendices

## Appendix 1 – Copper layers of the PCB

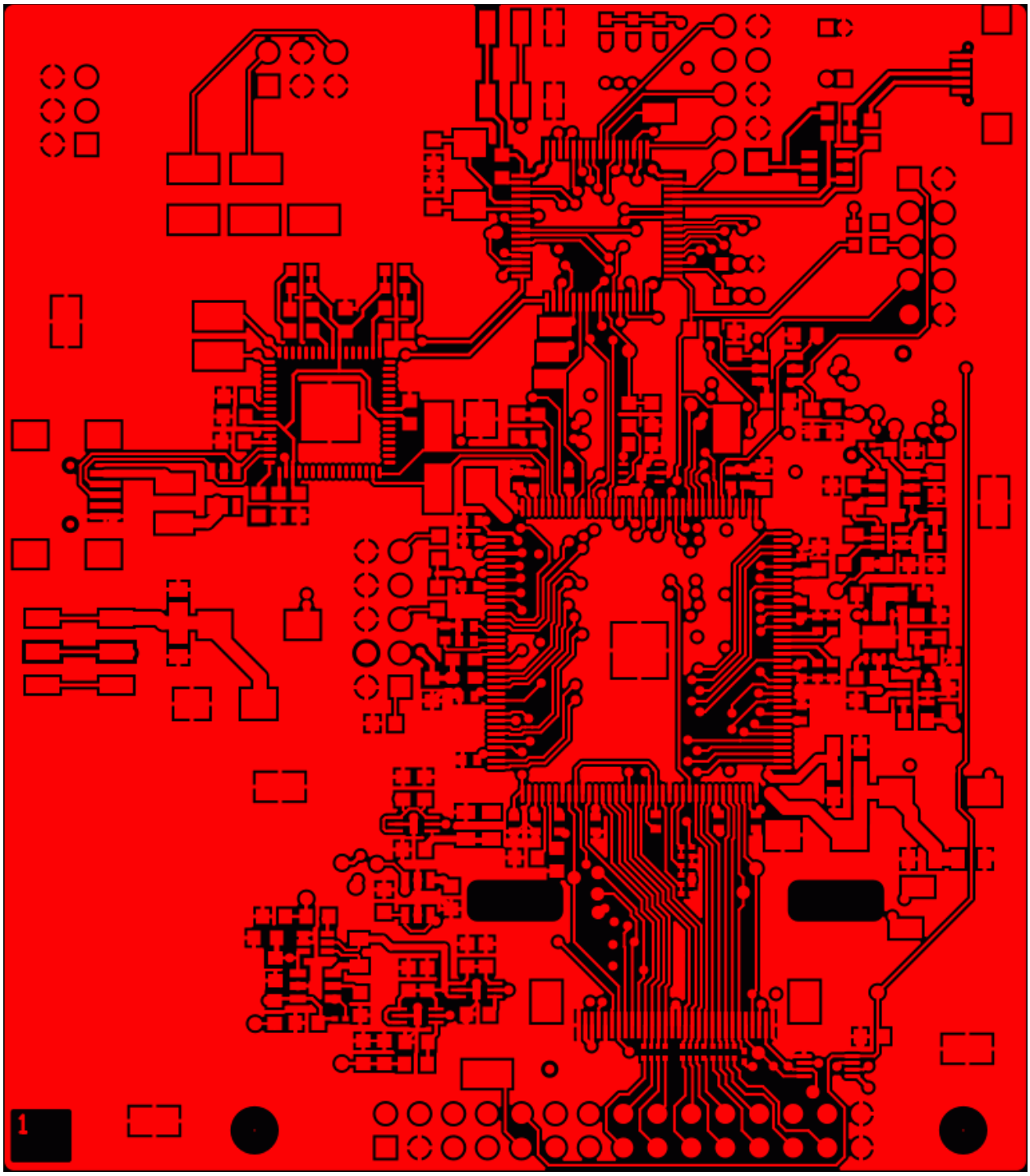


Figure 11 PCB Layer #1 (top)

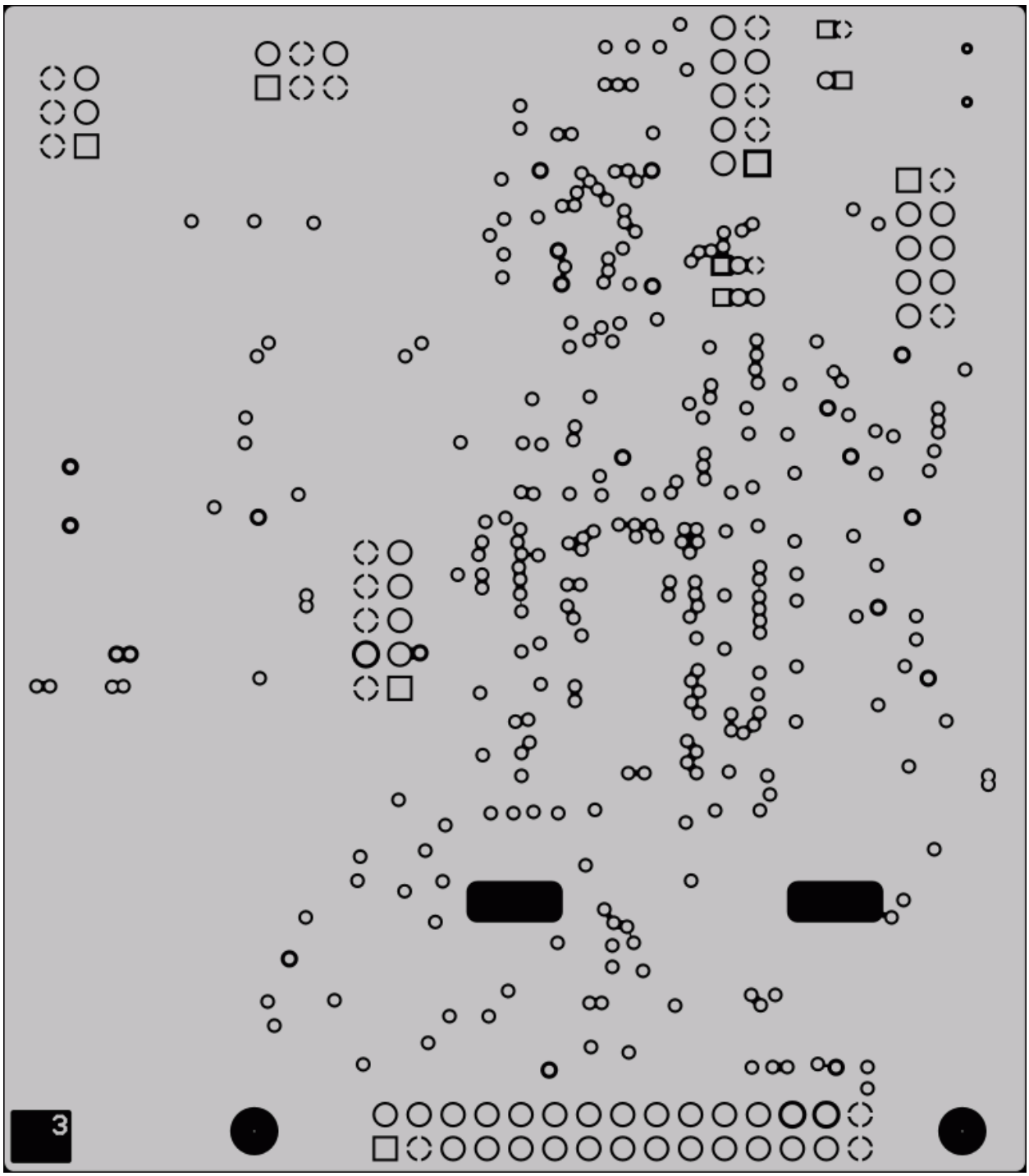


Figure 12 PCB Layer #2

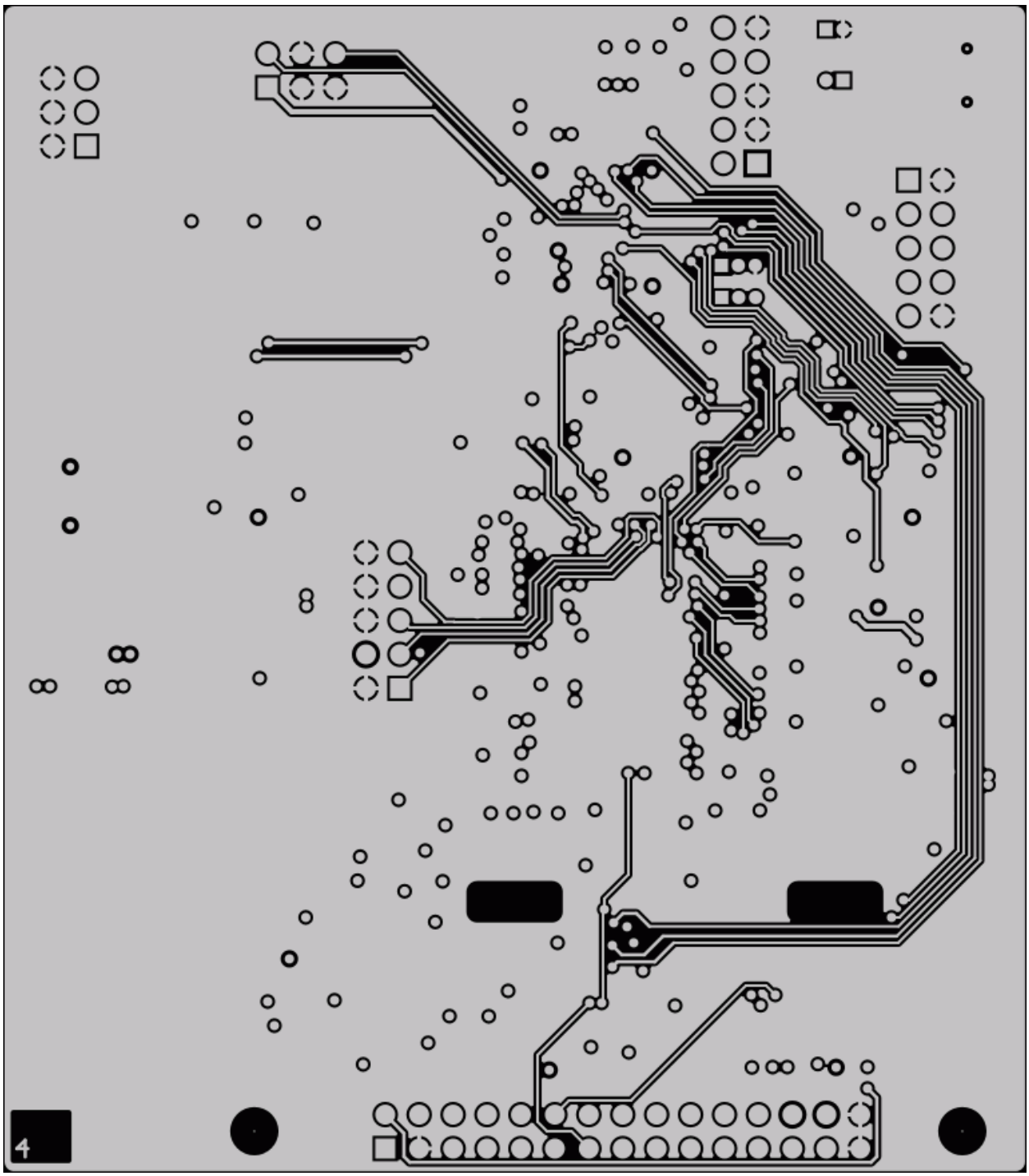


Figure 13 PCB Layer #3

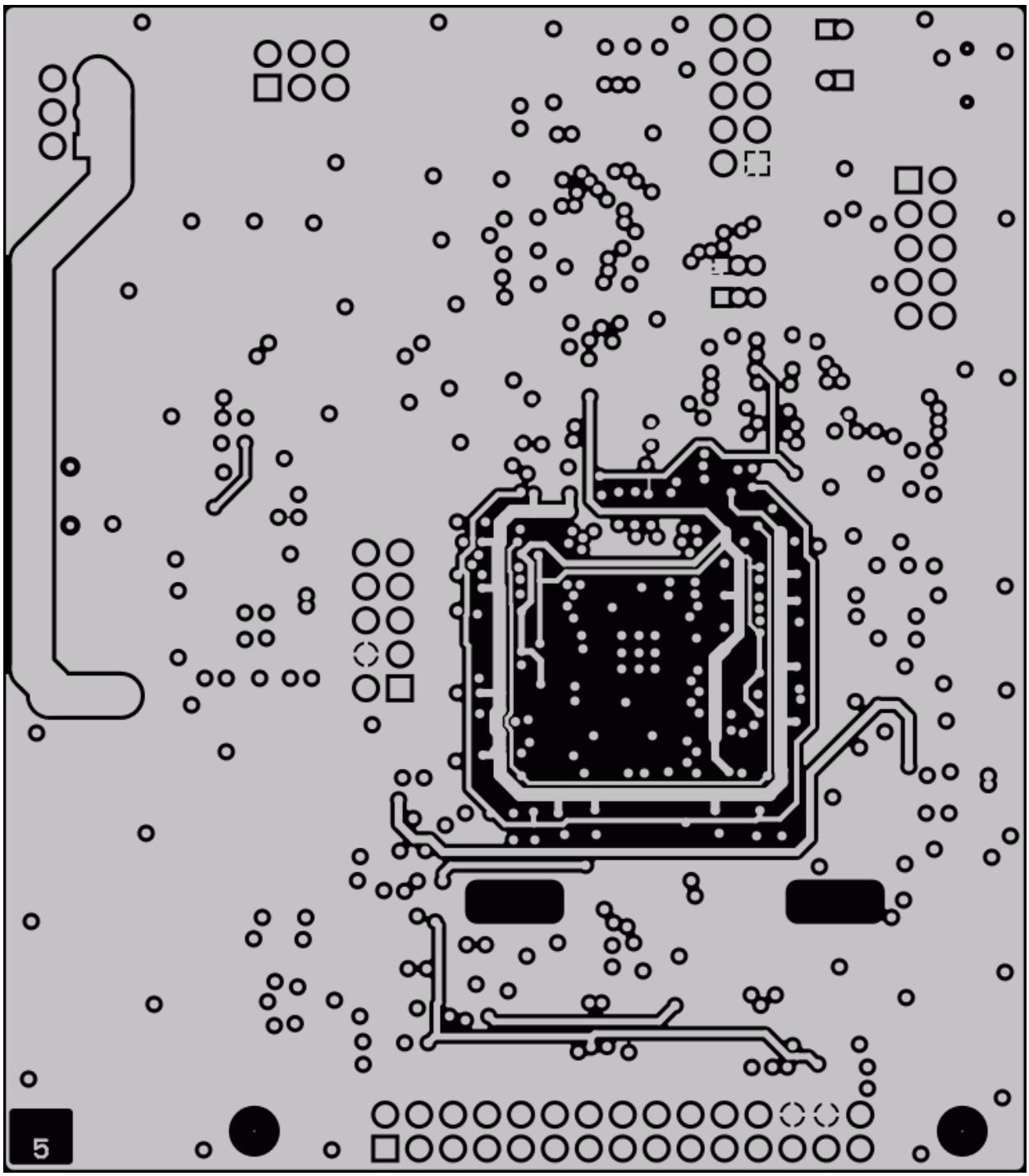


Figure 14 PCB Layer #4

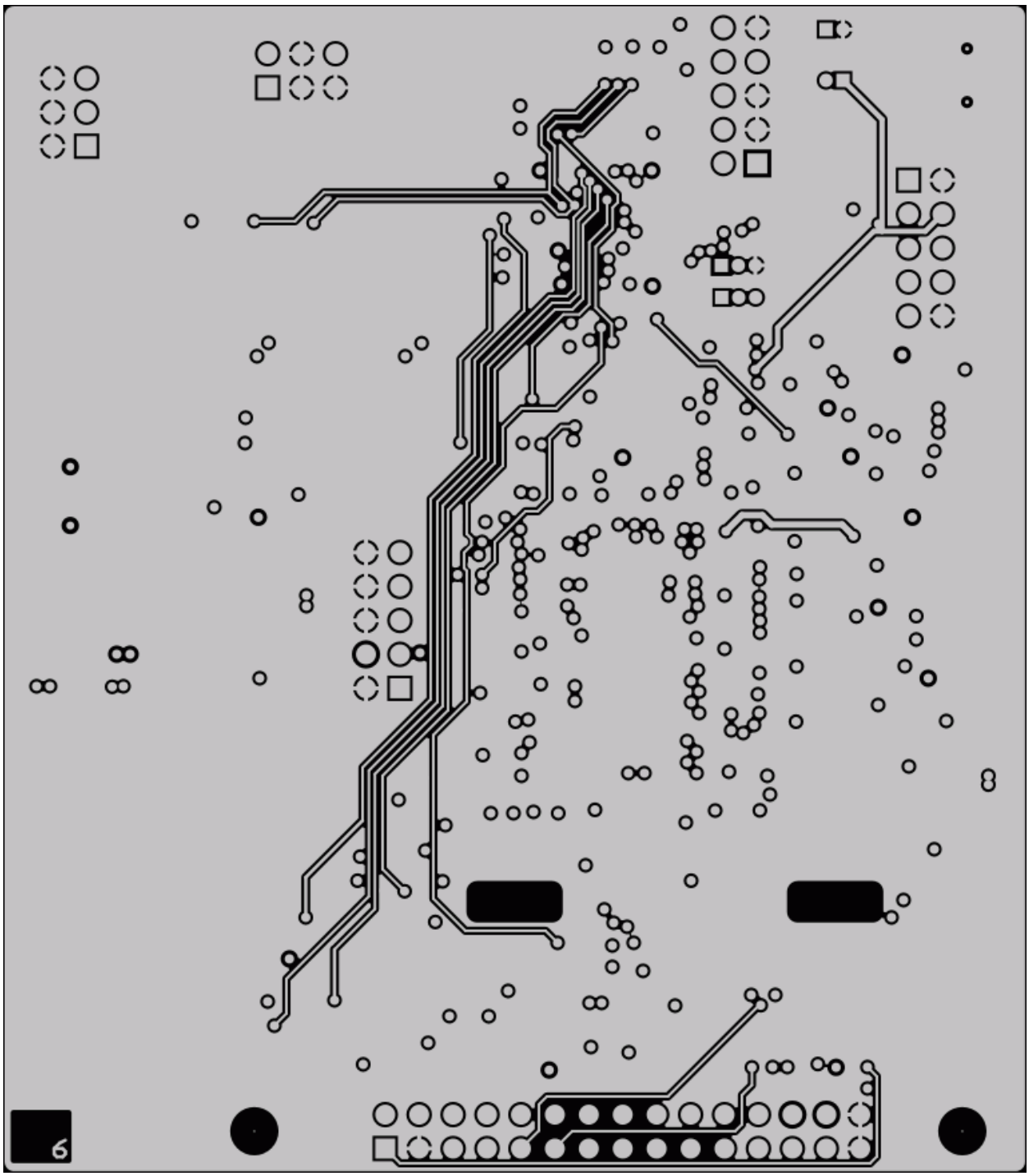


Figure 15 PCB Layer #5

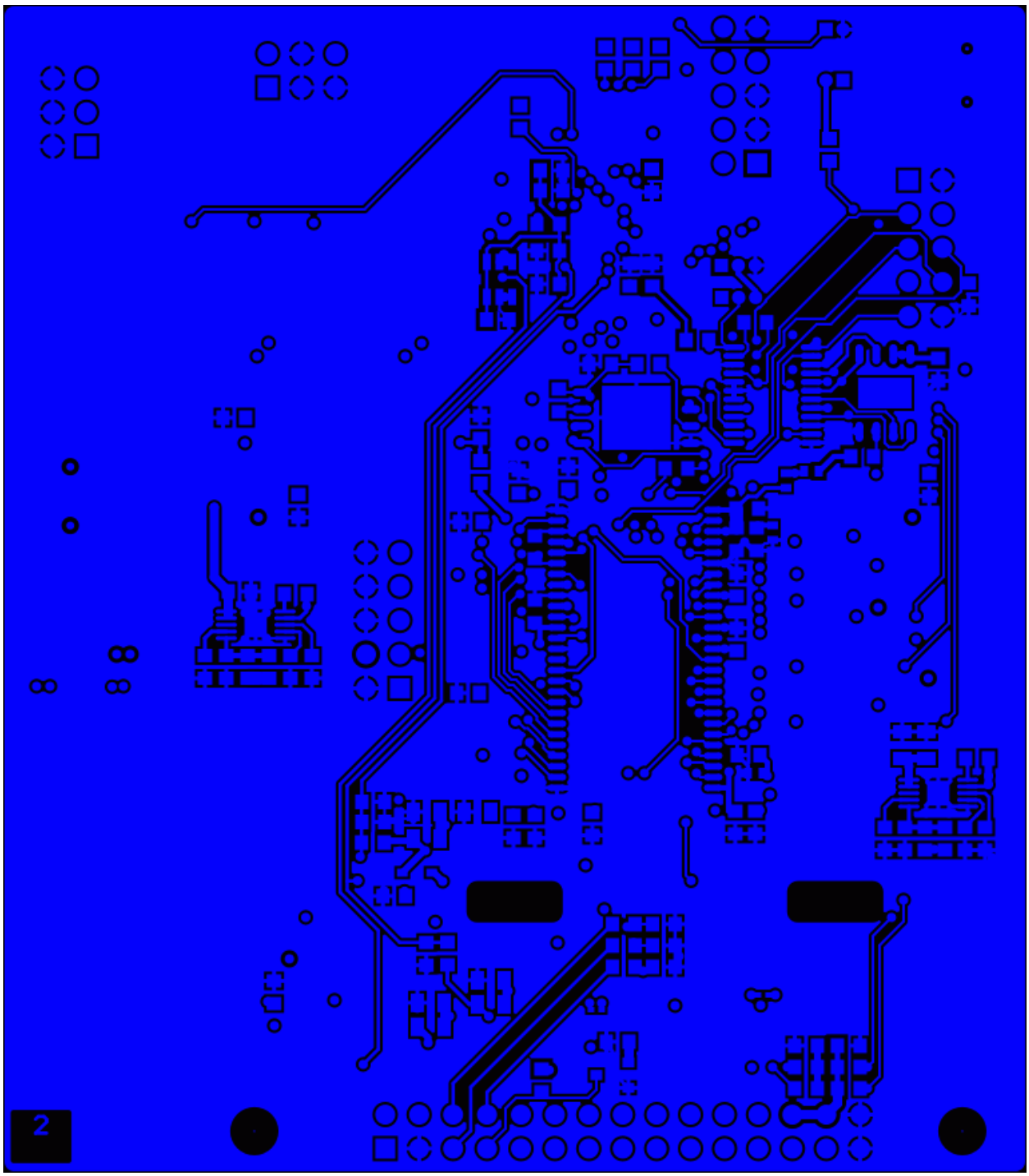


Figure 16 PCB Layer #6 (bottom)

## Appendix 2 – Binary .rpd file generation guide

The process of generating an .rpd file from Quartus II goes as follows:

1. Generate a .pof file by compiling the project
2. From Quartus II choose "File → Convert Programming Files ..."
3. Programming file type → choose "Programmer Object File (.pof)"
4. Configuration device → choose "EPCS4"
5. Mode → choose "Active Serial"
6. Click "Advanced" and from the dialog that opened
    a. Check the "Disable EPCS ID check" checkbox
    b. Uncheck the "Disable AS mode CONF_DONE error check" checkbox
    c. Leave others as is (0, default, default, 1) and click "OK"
7. Optional: check "Create Memory Map File (Generate xxx.map)"
    a. While this is not needed, it will contain the space requirement for the FPGA configuration. The FPGA, based on measurements with a logic analyser, does not read more bytes than it needs. Therefore it is possible to store sensor configuration in the high addresses of the FRAM. Using the .map file it is possible to calculate how much space will be left (for the sensor configuration).
8. Under "Input files to convert" click on "SOF Data" line and from the right side choose "Add File..." and browse to the .sof file generated at step #1
9. Click on the .sof file you just imported (below "SOF Data") and from the right side choose "Properties"
10. In the dialog that opened check the "Compression" checkbox and click "OK"
11. Click "Generate" and in the "Generated xxx.pof successfully" dialog click "OK"
12. Convert .pof to .rpd
    a. Using the same "Convert Programming Files" window choose "Raw Programming Data File (.rpd)" as the "Programming file type"
    b. Under "Input files to convert" click on "POF Data" line and from the right side choose "Add File..." and browse to the .pof file generated previously
    c. Click "Generate" and in the "Generated xxx.rpd successfully" dialog click "OK" You can now close the Convert Programming Files window (and Quartus II)

## Acknowledgements

I would like to sincerely thank my supervisor Erik Ilbis for assigning me this challenging yet exciting thesis topic, supervising this long-running project and teaching me so many new things in the process.

I would also like to thank the whole ESTCube team for their support, especially Ervin Oro for helping with porting the image sensor driver and Adrian Kirikal for helping writing software for the FPGA.

Finally, I'd like to thank my family for their support.

## Non-exclusive license to reproduce thesis and make thesis public

I, Jürgen Laks,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work from 20/05/2019 until the expiry of the term of copyright,

Developing hardware for the ESTCube-2 star tracker supervised by Erik Ilbis.

2. I am aware of the fact that the author retains the rights specified in p. 1.

3. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Jürgen Laks
20.05.2019