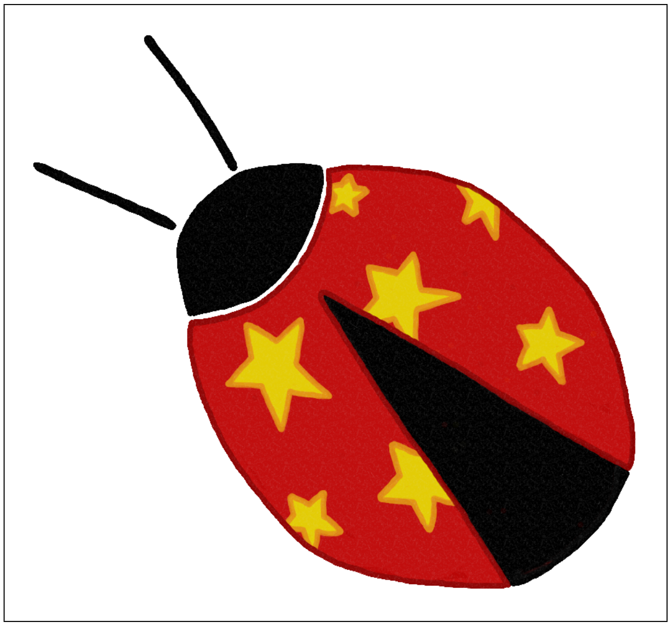
European CanSat Competition 2018



Team Starbugs

Bremen, Germany

Lennard Anders

Joscha Knobloch

Swantje Knüwer

Marius Kriworuschenko

Gianluca Müggenburg

Nicolas Nierentz

Contents

List of Figures

List of Tables

1 Introduction

1.1 Team organisation and roles

Lennard Anders

- **Field of work:** structure, recovery system
- **Expected workload:** much
- **Hours at school:** 0h
- **Hours after school:** 10h per week
- **Interests:** CAD, hardware engineering

Joscha Knobloch

- **Field of work:** Head of hardware design, PCB, schematics, software satellite
- **Expected workload:** much
- **Hours at school:** 0h
- **Hours after school:** 15 per week
- **Interests:** electrical engineering, hardware/software interaction

Swantje Knüwer

- **Field of work:** software satellite, physics
- **Expected workload:** much
- **Hours at school:** 0h
- **Hours after school:** 12h per week
- **Interests:** mechanical engineering, physics

Marius Kriworuschenko

- **Field of work:** financing, outreach program, PCB, planing
- **Expected workload:** much
- **Hours at school:** 0h
- **Hours after school:** 12h per week
- **Interests:** automatization, electrical engineering

Gianluca Müggenburg

- **Field of work:** ground station, blog
- **Expected workload:** much
- **Hours at school:** 0h
- **Hours after school:** 10h per week
- **Interests:** IT, Software Development, UI-Design

Nicolas Nierentz

- **Field of work:** ground station, analysis tool
- **Expected workload:** much
- **Hours at school:** 0h
- **Hours after school:** 10h per week
- **Interests:** IT, Software Development

1.2 Mission objectives

The idea for the secondary mission came from the problem faced by satellites orbiting close to earth: they can only communicate with the ground station by radio for a limited amount of time because the transmission gets blocked by the earth itself.
In order to avoid this problem, so-called geostationary satellites are used to establish a radio relay system (in addition to their regular function).
The secondary mission consists of three goals. The first goal is to establish a connection between two satellites. For that a satellite will be simulated on the ground which sends out data. Then the satellite in the air collects this data and forwards it back to the ground station. Another goal is to create a very robust and fault tolerant system for the satellite. Lastly it should be build as a very modular system which allows to swap out the secondary mission to be able to adapt to other missions.
The mission is considered successful if the ground station receives data from the satellite. If this shouldn't be the case than the mission is seen as failed.

Modularity Our satellite consists of two modules. The primary module contains all the necessary parts which are required to reach the prerequisite of the competition as well as parts that were determined to be useful in all space-missions.
All design decisions were made with the modularity and reusability in mind. This allows to easily swap out the secondary module. The primary module can be used for a variety of missions. The mission specific functionalities then can be added via the secondary module.

In our case the secondary module contains everything needed for our secondary mission as well as batteries which allow for a higher runtime and another SD-Card for a second data backup.

2 CanSat Description

2.1 Mission overview

The CanSat was designed and built so that it can be launched with a rocket up to a height of 1000 metres above the ground. The fall velocity is designed to be 9m/s. It contains the needed technology in order to accomplish the above mentioned goals. Therefore it does the following:

- measure temperature
- measure air pressure
- measure humidity
- measure orientation/acceleration
- measure the battery voltages
- receive GPS information
- receive data from the simulated satellite of the ground station
- send all the acquired data to the ground station
- save all data on both micro SD-Cards

The sensors get polled at a rate of 20Hz for a duration of 31.6 hours with full batteries.

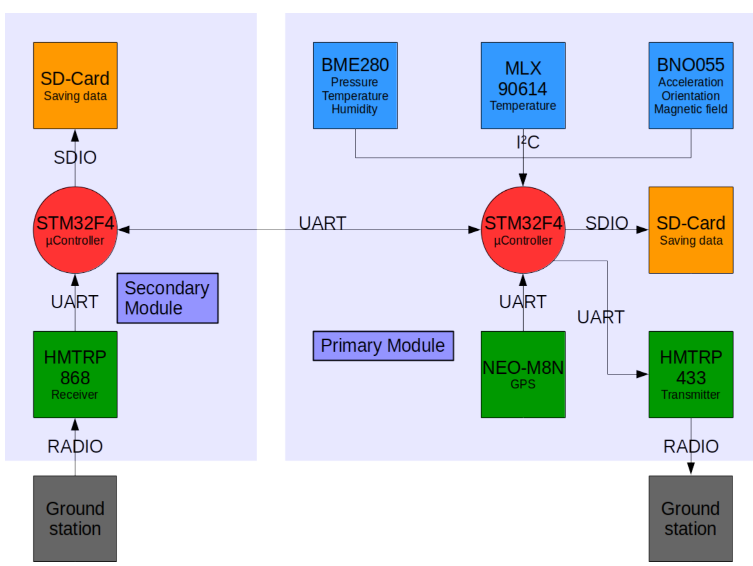


Figure 1: Block diagram

2.2 Mechanical/structural design

The structural design of our satellite has gone through many iterations. But for the sake of this documentation only the parts of our satellite that changed since the German competition will be mentioned.

2.2.1 General design

Since the German competition the casing for the satellite has shrunk from 67mm to 66/65mm in diameter and 116mm to 114mm in height. This is due to different guidelines compared to those in the German competition. The guidelines changed from 66 by 115 millimetres with 2mm tolerance to the same dimensions without the tolerance. Since space on the PCBs was already tight, it was the goal to keep using 60mm diameter PCBs, the thickness of the walls got reduced from three millimetres to two or two and a half millimetres which is noticeably thinner. In order to counteract this loss of stability the material of the casing got changed from PLA to carbon fibre reinforced PLA.

Figure 2: General design

There is also a 3D-printed plate between both missions.
It not only enhances the structural strength of the satellite since it protects the walls from flexing to much but also adds additional protection against batteries or other components from moving freely inside the satellite in case the mount of a component fails.

2.2.2 Mechanical integration with other components

This part of the design also has gone through some major changes.

Parachute Firstly the mounting mechanism of the parachute was changed from simple knots holding the parachute to a 3D-Printed mechanism where the strings are held by the casing itself and are fastened by a screw.

Figure 3: Parachute

PCB Secondly the PCBs are no longer held by sandwiching the two halves of our satellite but rather by screws being screwed directly into the plastic. This has the advantage that the satellite can be easily assembled / disassembled. Previously at the German competition the satellite took quite some time to get assembled and this was one critique point where the satellite could have been improved. Since the GPS-module is not directly soldered to the main PCB there are four standoffs for it so it does not rattle inside of the casing.

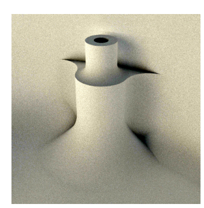


Figure 4: PCB

Temperature Sensor Thirdly the hole for the temperature sensor was made much smaller and is above the PCB. The reason for this design change is that previously there needed to be a cut out of a section of our PCB to get past the hole in the old design. Now the sensor is on the very edge of the PCB so the limited space can be used effectively.

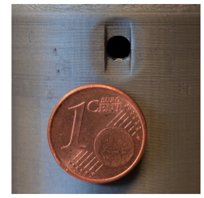


Figure 5: Sensor

Access to switches Fourthly there are two additional holes in the casing. These are located at the exact position where the switches are on the PCB so the satellite can be switched on or off even when it is closed. Not being able to do this also was one of the major critiques at the last competition.

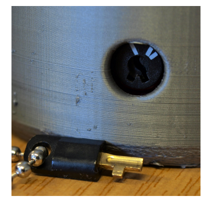


Figure 6: Switches

Effective usage of Space Finally a lot of batteries could be fit inside of the satellite by calculating in advance how thick the PCBs with all their components are going to be and then put the mounting holes for them at the very top and bottom of the casing. Because of this there is now space to put up to 42 watt hours of batteries mounted by Velcro straps inside the satellite.

Figure 7: Batteries

2.3 Electrical design

The decision was made for a clean split between the two missions. Both modules can be used freely without the other one. Never the less a interface between those modules is implemented. The power supply solution for both modules is unified. In case of a malfunction of a battery or the connection between the modules the other one is capable to supply the module with power. The list for the technical requirements can be found in section 6 Requirements.

2.3.1 Power supply

In order to accomplish the following requirements a power supply solution was developed which is derived from the main goals.

- 3.3 V output
- low-noise and stable supply for the analog sensors

- minimum current handling capability of 500 mA
- high efficiency
- small footprint
- variable extra battery capacity within the secondary module
- Fault-tolerance in case of a defective cable or battery
- reusability and simple recharging

Regulator The integration of a self-built step-down switch-mode power supply was chosen because one-chip solutions are mostly linear regulators with a poor efficiency compared to switched ones. This also results in a smaller footprint on the PCB because a regulator with external components is smaller than a pre-built module.

In the end the Texas instruments TPS-565201 was chosen due to its good efficiency. It can also handle the required current with ease. Furthermore it has a small but still solder-able package as well as good documentation for choosing the external components and designing a good layout.

The design consists of multiple analog MEMS-Sensors. Testing was done and the decision was made to avoid using a separate linear regulator due to the fact that the results with our self-built regulator were good. This is probably due to the relatively big inductor, generous amount of filtering and decoupling. A separate linear regulator would have also added a lot to the footprint of our power supply solution.

A picture of the Power supply can be found in the appendix.

Battery The battery-technology of choice is lithium-polymer as it has a very good capacity to size/weight ratio and is very robust. It is also available in a lot of packages.

For the regulator to work normally it needs at least 0.83V more than the output voltage it is set to. As a normal lithium-polymer battery will drop below this threshold in the 3.3V system 2 cells were connected in series.

Battery-management and -charging For battery management and charging, batteries with a protection circuitry already built into the package were bought. This protection circuitry handles overcharging, over-discharging, too fast charging/discharging and short circuit.
The charging circuitry is not included into the satellite. A rather simple charging station, to put the easily removable batteries in, was developed. This saves space in the satellite and allows the use of a large charger that can charge all batteries simultaneously. Another advantage it provides is being able to quickly switch batteries and get the satellite up and running again fast when the batteries are empty.

Coupling both modules batteries onto a single rail Actions were taken in order to allow the secondary module to have no battery at all or as much as the developer anticipates. Redundancy in case of a damaged cable is also provided. Therefore both of the batteries were coupled into the regulators input-rail, of which we have one per module, with a shotkey-diode. This allows the batteries to have different capacities and still get discharged at the same rate. The modules can also be safely connected even if the battery voltages are different as only the one with more voltage gets discharged until both are at the same level. No current can flow between the batteries. In terms of fault-tolerance and redundancy this has the advantage of allowing one battery or the cable that attaches the battery to the system to fail. Then the left over battery supplies the whole system with power. In terms of power even the connection cable between the modules is allowed to fail. This gives the system three connections, one of which may fail, therefore making it a good redundant system.
In case of our secondary module we settled for the extreme of having a lot of battery capacity added to the system. The other extreme of having no batteries at all in the secondary module can also be easily achieved. Therefore the voltage of 3.3V is available on the mission-interconnect-connector. If another voltage is required it can be regulated from the battery voltage which is also present on the connector. This way systems with a minimum power consumption or a very short flight can be built even without a regulator or battery.

Battery voltage readout To read back the current battery voltage the MCUs integrated ADC is used. As the battery voltage is too high for the ADC it is lowered by a simple voltage divider.

Switching the satellite on and off For switching the satellite on and off a key-switch was build into the design. It switches off the batteries. This means that the switching on one of the batteries turns on both modules if they are connected. In this case only the switched on battery will be drained. More details on the switch can be found in subsubsection 2.3.3 Parts.

Drawbacks of this design

- A fraction of the capacity is lost within the shotkey-diode. It is approximately 5% of the batteries capacity.
- If a secondary mission needs a lot of power and the cable of the secondary missions battery fails the current might not be handled. In this case the current handling capabilities of the primary modules battery is the bottleneck, limiting the system to a maximum of 500mA in total on the 7.4V rail.
  This can be overcome by switching to batteries with a higher output current rating. The next bottleneck in the chain would then be the connector, which limits the secondary module to a maximum of 1A on the 7.4V rail. Any higher load might damage the connector.

Battery capacity / battery duration To get an estimate of the resulting duration, the battery capacity of the satellite was determined at first.

Module	Capacity
Primary	2 batteries in series at 1100mAh
	7.4V — 1.1Ah
Primary	3 times 2 batteries in series at 1850mAh
	7.4V — 5.55Ah
Sum	6.65Ah

Table 1: Battery capacity

After that the expected power draw of the system was determined. In Table 4 Modules primary module and Table 5 Modules secondary module all components are listed with the expected current draw according to the data-sheets. In addition to the current draw of all the components the efficiency of the regulator as well as the voltage drop over the diode needs to be factored in.
Tests showed that the Regulator always has a better efficiency than 90%. The diode has a voltage drop of 0.4V. The base data with which the power duration was calculated:

- current draw of the primary module: 243.4 mA
- current draw of the secondary module: 158.1 mA
- battery voltage: 7.4V
- battery capacity: 6.65Ah
- the regulators efficiency: >90%
- the voltage drop of the diode: 0.4V

The total power duration is calculated the following:

Voltage on the regulator input and therefore after the diode:

$$7.4V - 0.4V = 7V$$

Total energy at the regulators input:

$$7V * 6.65Ah = 46.55Wh$$

Energy after the regulator:

$$46.55Wh * 0.9 = 41.895Wh$$

Capacity on the 3.3v rail:

$$41.895Wh/3.3V \approx 12.69Ah$$

Total power duration:

$$12.69Ah/401.5mA \approx 31.6hours$$

power consumption The total power consumption that needs to be listed within the requirements table is calculated as follows:

$$(0.4015A * 3.3V)/0.9 \approx 1.472W$$

2.3.2 General electrical design and PCB layout

The decision was again to only use a single PCB per Module which sits horizontally in the satellite. This has the advantage of using the available space as efficiently as possible. No Space is wasted on board-to-board interconnects. However as it is only a single board space is at a premium and tight integration is key when designing around this requirement.

Therefore special care was taken to make every single aspect of the satellite as small as possible within our possibilities. All of the components were carefully selected to provide a small footprint, high reliability and a small power draw.

PCBs As the primary modules PCB contains a lot of connections and provides the base for a couple of noise critical circuits a 4-layer board was chosen. This has the advantage of providing a more stable ground and power transport as they have their own discrete layers. Also the extra space is used to route the traces more tightly and reduce inductance in the paths.
Pictures of the primary modules PCB can be in the appendix:
Primary top rendering
Primary bottom rendering
Primary top
Primary bottom
Secondary top rendering
Secondary bottom rendering
Secondary top
Secondary bottom

Assembly Due to the integration of sensors with a very small footprint a hand soldering approach was not suitable anymore. Therefore our sponsor Straschu Holding GmbH handles the PCB production and assembly, while we provide all the necessary data and parts.

Integrated Antenna on the PCBs Part of the primary modules antenna and the whole secondary modules antenna is printed on the PCB itself. This saves space in the module and increases the antennas accuracy. Further details on the antennas can be found in subsection 2.4 Antennas.

Connectors and cables Cables were only used if absolutely necessary, because they were classified as potential weak points in the design. To standardise the connectors, only a single connector type should be used in the satellite. To minimise the possibility of plugging a connector into the wrong socket, different numbers of contacts were used.

The selection process contained ordering a couple of connectors from Molex and JST to see which one is best suited. The JST-GH connector was chosen because of its small size, high current handling capabilities and availability in all needed number of contacts.
Two batteries always share a single 4-contact connector. The 5-contact connector is used for the mission interconnect cable. The 6-contact connector is used for the debug interface.

The final cables are professionally manufactured by Straschu Holding GmbH. This was especially helpful, as the crimp contacts are very small and only easily crimp-able with a very expensive crimping tool.

2.3.3 Parts

STM32F413RH As the Microprocessor[1] is the heart of the project, a lot of time was spent on research and part-selection. The following requirements were set at the beginning:

Figure 8: STM32F413RH

- operation at 3.3V
- small but solderable package
- a good development environment with modern debugging-support.
  As the Arduino-environment was formerly used, a lot more flexibility was needed.
- flexibility in terms of IO and digital interfaces. At least the following is required
  - 1 I2C-Bus
  - 1 SPI-Bus

[1] [11] *STM32F413RH Datasheet*

  - 4 Uart-Interfaces
  - 1 ADC-channel

- good documentation and support. At best widely used and well understood by a lot of people on forums.
- a built in RTC
- built in support for CRC32 generation
- good DMA support

In the end, every big manufacturer of Arm MCUs would have had a chip within their lineup that ticks all the boxes for the listed requirements. The STM32F413 by ST-micro was chosen because they manufacture mainly in Europa which leads to good availability and is most widely used.

Integrating the MCU was relatively easy. It needs more external components to work but is very flexible as oftentimes functions can be used on different pins to reduce routing problems. Especially for using the RTC, a number of external components is needed but compared to a independent RTC-chip the integrated one still has a smaller footprint.

The LQFP64 package was chosen because it has the lowest pin-count and is easily hand solderable. The maximum frequency of 100MHz leaves a lot of room for complicated algorithms.

GPS-NEO-M8 Since a lot has to be considered for the development of a GPS module and the necessary time is not available, the GPS[2] module was not developed by ourselves. The decision was then based on a GPS module from Drotek which has the Neo-M8 chip installed. With the simultaneous reception of up to 4 GNSS (GPS + GLONASS + BEIDOU + Galileo) a relatively accurate position determination is possible and still small enough to find its place in the satellite. Since the first GPS module with pin header was no longer available, it was necessary to select the one that could be connected to the board by cable, although the goal was to use as few cables as possible. In order to securely fit the GPS module, it is screwed to the housing and connected to the board with a JST-GH plug so that it can't slip out. The cable itself is explained in more detail in section 2.3.2 Connectors and cables.

Figure 9: GPS-NEO-M8

SD-Card The micro-SD-Card is used to save the measured data on the satellite. Data is written to it using the SDIO-Protocol in 1-Bit mode.

[2] [12] *UBLOX NEO M8 Datasheet*

HMTRP-433 and HMTRP-868 Due to the secondary mission two frequencies are required so data can be received on the secondary module and then transmitted trough the primary module to our ground station. For that we use the frequencies 434 MHz on our primary module to transmit the data and 868 MHz to receive the data from our simulated satellite. Since the development of a self-made transceiver isn't easy and a lot of experience is needed we didn't designed our own transceiver and chose the HM-TRP[3] product line which are easy to configure. Both of the transceiver use frequency shift keying (FSK). The antennas for each transceiver are explained in subsection 2.4 Antennas.

Figure 10: HMTRP

Buzzer To find the CanSat easier and to have a possibility of debugging, a buzzer[4] was installed in the satellite. It should be small but powerful so it can still be heard. The decision fell on a audio magnetic buzzer which can be operated with a voltage of 3.3V and is up to 90db loud.

Keyswitch For switching on and off we have decided on a key switch[5]. Due to its design this does not necessarily require the key to be switched but should nevertheless not be turned by any force that occurs. The force required for this would have to be rotating and quite large. To operate the switch a hole has been installed in the housing so that the circuit boards do not have to be removed before.

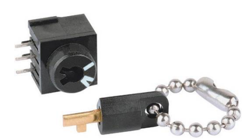


Figure 11: Keyswitch

2.3.4 Sensors

The idea of integrating the sensors on the board instead of a breakout board came up after the German competition when the sensors were reevaluated. For this purpose these should be Bosch sensors as far as possible as experience has already been gained in the German competition. Furthermore these should not only be as precise as possible so that its data can be evaluated better but also as small as possible so that space on the board can be saved as already mentioned in the general electrical design.

BME280 The BME280[6] from Bosch Sensortec is an environmental sensor in which air pressure, humidity and temperature sensor are incorporated. The wiring of the BME280 is kept very simple and can therefore be kept very small. Since the sensor has a built-in temperature and air pressure sensor the altitude can be calculated by the sensors data alone. But as the temperature sensor is somewhat inertial due to the temperature-dependent resistor which must first adapt to the ambient temperature. That's the reason why there is another infrared temperature sensor on our PCB.

Figure 12: BME280

[3] [2] *HMTRP Datasheet*
[4] [6] *AUDIO MAGNETIC XDCR 2-4V TH*
[5] [5] *Miniatur-Schlüsselschalter*
[6] [7] *BME280 Datasheet*

BNO055 The BNO055[7] from Bosch Sensortec is a small sensor integrating a triaxial 14-bit accelerometer, a triaxial 16-bit gyroscope with a range of ±2000 degrees per second, a triaxial geomagnetic sensor and a 32-bit microcontroller running the company's BSX3.0 FusionLib software. Due to the accuracy of the sensor the position and motion of the satellite can be accurately determined and the built-in microprocessor helps calculate the data so that the calculated data can be read directly from the I^2C bus. Furthermore, the sensor itself is much smaller than other system-on-board solutions and can save a lot of space on the board. Thus the sensor could be placed in the middle of the satellite to achieve an even higher accuracy. Since the sensor is so small we can't solder it by hand which is why Straschu Holding GmbH helps us by machine assembly. The sensor is also powered by a voltage of 3.3V.

Figure 13: BNO055

MLX90614 The MLX90614[8] from Melexis is an infrared temperature sensor. Advantages of an infrared temperature sensor are the very fast and accurate determination of the temperature of an object. To measure the temperature accurately a metal plate surrounded by air is required to which the temperature sensor points. This sensor is not from Bosch as they do not have an infrared temperature sensor but most requirements are still fulfilled by the sensor. The sensor can be powered by 3.3V and measured data can be read by using the I^2C bus.

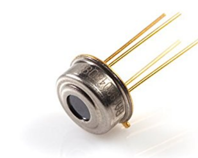


Figure 14: MLX90614

An overview about current consumption and used interfaces can be found in Table 4 Modules primary module and Table 5 Modules secondary module.

2.4 Antennas

Both modules are equipped with there own antennas as they operate at different frequencies.
The primary module operates at 434 MHz and uses a ground plane antenna that is upside down. The Antenna itself is located within the parachutes ropes. The counter-pole is printed on the PCB. This is further described in the PCB section.
The secondary module operates at 868MHz. For it a halo antenna is used. A halo antenna is an omnidirectional lambda half dipole antenna that easily fits in the satellite. The halo antenna is integrated on the PCB as well.

2.5 Software design

During the German CanSat Competition the Arduino development environment was used together with Arduino libraries and the programming language C++. Many Arduino libraries are very poorly programmed and need a lot of time.

[7] [8] *BNO055 Datasheet*
[8] [4] *MLX90614 Datasheet*

To improve the performance of the CanSat the decision was made to use a different micro controller. The advantages of the hardware are mentioned in section 2.3.3 STM32F413RH. The program for the European CanSat Competition was developed in C using CubeMX and System Workbench for STM32. The program uses the STM HAL (Hardware Abstraction Layer) library as well as Bosch sensortec libraries for the BME280[9] and BNO055[10]. Accessing the interfaces by using the functions written for this directly and the use of libraries written by the producer of a sensor improves the performance of the CanSat.

2.5.1 Program flow

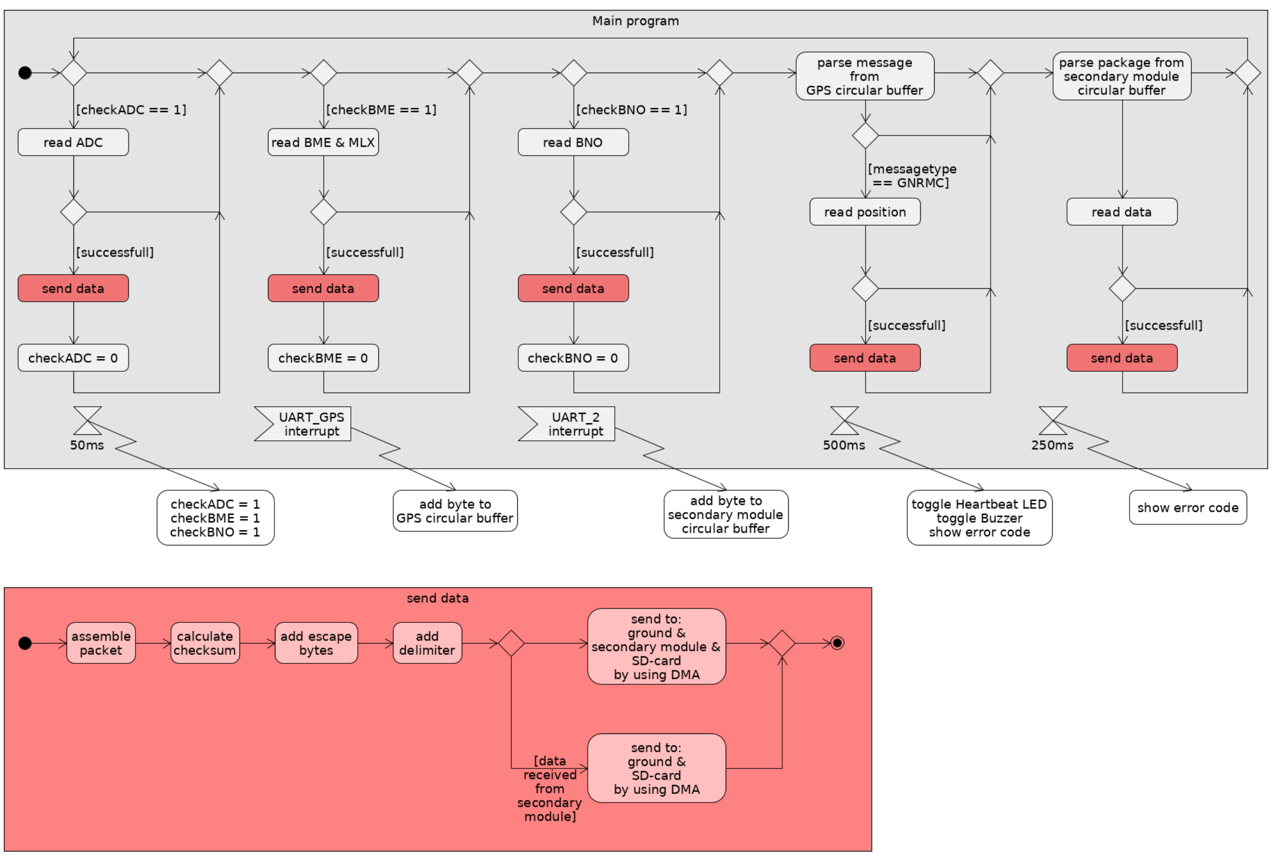


Figure 15: Activity diagram primary module

The program starts with initialising the STM32F413RH itself and the periphery. Afterwards it tests the connection to the UART devices as well as the I^2C sensors and initialises the sensors. If errors occur they will be stored in variables to be later shown by LEDs. After finishing the initialisation the timer interrupts and UART

[9] [9] *BME280 Driver*
[10] [10] *BNO055 Driver*

interrupts are started.

Timer Three different timers are used by the program. The interrupt of the first timer will occur every 50ms to set the variables which show the main program to read data from the ADC and the different I^2C sensors. The other timers are used to show error codes (Table 8 Error codes primary module) and to toggle the heartbeat LED. An interrupt occurs every 500ms respectively 250ms for slow or fast flashing.

UART receive interrupts If a UART receive interrupt occurs the received byte is written to the circular buffer responsible for the corresponding UART interface.

Voltage If *checkADC* is set to 1 the program reads a value from the ADC. The ADC value shows the voltage at this pin proportional from the operating voltage as a proportion of the maximum possible return value (4096). The battery voltage can be calculated by determining the voltage at the ADC pin and removing the changes made to the voltage due to the use of a voltage divider.

$$V_{\%} = adc\_value/4096.0$$

$$V_{Pin} = P\% * 3.3$$

$$V_{Battery} = V_{P}in * 3.14$$

BME280/MLX90614 If *checkBME* is set to 1 the program reads temperature, humidity and air pressure from the BME280 as well as the temperature measured by infrared from the MLX90614. If reading the data was successful it is sent to the ground station, the secondary module and written to the SD-Card.

BNO055 If *checkBNO* is set to 1 the program reads orientation and acceleration from the BNO055. If the readout was successful velocity and distance are calculated from the measured acceleration.

$$v_{direction}:\ v_1 = v_0 + (a_1 * \Delta t)$$

$$s_{direction}:\ s_1 = s_0 + (v_0 * \Delta t) + (0,5 * a_1 * \Delta t^2)$$

The data is send to the ground station, the secondary module and is written to the SD-Card.

GPS If bytes are available in the GPS circular buffer, the program parses GPS messages out of the data and then reads the position from the GNRMC messages. The position is sent to the ground station, the secondary module and is written to the SD-Card.

Mission-Interconnect If bytes are available in the secondary module circular buffer the program parses packages from the data. It disassembles the packages and send the data included to the ground station and writes it to the SD-Card. The data isn't sent to the secondary module to prevent loops.

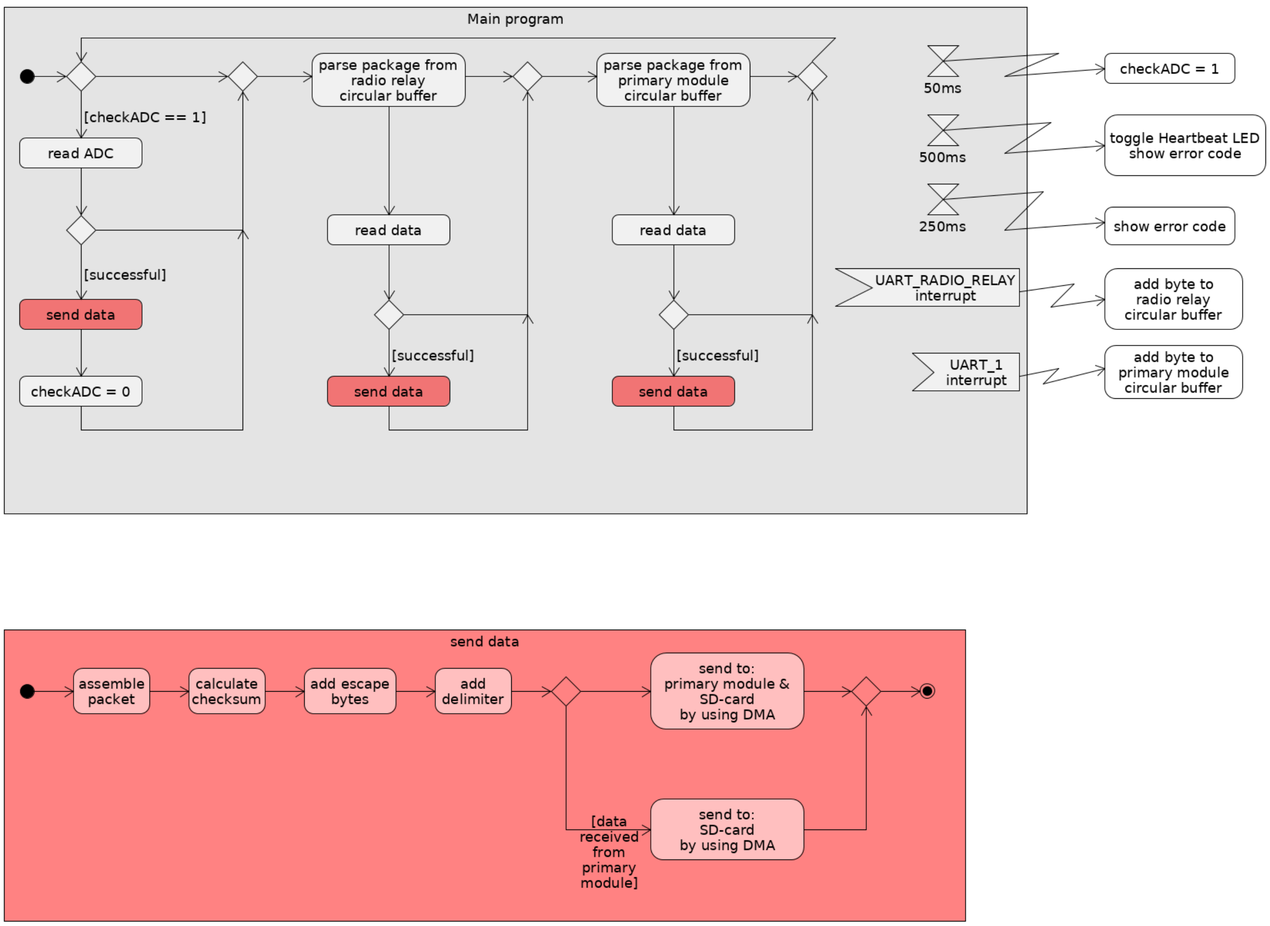


Figure 16: Activity diagram secondary module

The program starts with initialising the STM32F413RH itself and the periphery. Afterwards it tests the connection to the UART devices. If errors occur they will be stored in variables to be later shown by LEDs. After finishing the initialisation the timer interrupts and UART interrupts are started.

Timer Three different timers are used by the program. The interrupt of the first timer will occur every 50ms to set the variables which show the main program to read data from the ADC. The other timers are used to show error codes (Table 9 Error codes secondary module) and to toggle the heartbeat LED. An interrupt occurs every 500ms respectively 250ms for slow or fast flashing.

UART receive interrupts If an UART receive interrupt occurs the received byte is written to the circular buffer responsible for the corresponding UART interface.

Mission-Interconnect If bytes are available in the secondary module circular buffer the program parses packages from the data. It disassembles the packages and send the data included to the ground station and writes it to the SD-Card. The data isn't sent to the secondary module to prevent loops.

Voltage If *checkADC* is set to 1 the program reads a value from the ADC. The ADC value shows the voltage at this pin proportional from the operating voltage as a proportion of the maximum possible return value (4096). The battery voltage can be calculated by determining the voltage at the ADC pin and removing the changes made to the voltage due to the use of a voltage divider. The calculation can be found in Voltage.

Ground If bytes are available in the ground circular buffer, the program parses packages from the data. It disassembles the packages and send the data to the primary module and write it to the SD-Card.

2.5.2 Data

The data generated by the different sensors and modules is stored on a micro SD-Card, send from each module to the other and to the ground station via radio. Each packet is build up from different information by using the following structure:

$$time, counter, packetnumber, packettype, data, checksum$$

Name	Size (Bit)	
time	32	Time since restart in ms
counter	8	Number of restart
packetnumber	32	Number of packets since restart
packettype	8	Type of data
data	variable	Data
checksum	32	CRC32

Table 2: Protocol

More detailed information about the different packet numbers can be found in Table 10 Packettypes primary module and Table 11 Packettypes secondary module. The packets are separated by a delimiter. If the delimiter occurs in the data it is escaped by an escape byte. If the escape byte occurs in the data it is also escaped.

Primary module During the expected flight time of 111s 250.416KB data will be generated. During the total life span of a battery charge(31.6h) 256.642560MB will be generated.
Detailed information about the amount of data generated by each part of the primary module can be found in Table 6 Amount of data generated by primary module.

Secondary module During the expected flight time of 111s 370.740KB of data will be generated. During the total life span of a battery charge of 31.6h 379.958400MB will be generated.
For more information about the amount of data generated by each part of the secondary module see Table 7 Amount of data generated by secondary module.

2.6 Recovery system

The parachute will be made of stable, airtight fabric. It is connected to the satellite by six strings. The following formula was used to determine the size of the satellite:

$$A = \frac{m * g}{c_w * \frac{1}{2} * \rho * v^2}$$

The determined size of the parachute is $455,3cm^2$:

$$A = \frac{0.3kg * 9.81\frac{m}{s^2}}{1.33 * \frac{1}{2} * 1.2\frac{kg}{m^3} * 9\frac{m}{s}^2} \approx 0.04553m^2 = 455,3cm^2$$

A hexagonal parachute with a hexagonal hole will have the following dimensions:

- Size of the hole: $A = \frac{3}{2} * (2.96cm)^2 * \sqrt{3} \approx 22.77cm^2$
- Size of the parachute including the hole: $A_g = 455,30cm^2 + 22.77cm^2 = 478.07cm^2$
- Radius of the parachute: $r = \sqrt{\frac{478.07cm^2}{\frac{3}{2} * \sqrt{3}}} \approx 13.57cm$

The expected flight time is 111s:

$$t = \frac{s}{v}$$

$$t = \frac{1000m}{9m/s} \approx 111s$$

2.7 Ground support equipment

Our ground support equipment is split into two pieces. The receive client and the transmit client.

2.7.1 Concept

Due to our secondary mission, two frequencies are needed to transmit and receive the data. For transmitting and receiving a Yagi-Antenna is used. Yagi antennas are longitudinal radiators and use radiation-excited elements. A packet loss is displayed on the ground station software to see if the antenna has been correctly aligned. For the European competition, this function has been improved by placing an OLED display on the antenna itself, so that the packet loss is displayed on the antenna itself and the holding person can correct the alignment himself. The ground station is measuring pressure and temperature too, so that the data of the satellite can be compared to the ground stations data. In order to be able to calibrate the radio modules themselves and to display the packet loss, temperature and air pressure on the OLED display, boards with a built-in Atmel microprocessor - the ATmega328p - have been created. This one is very small and is completely sufficient for this purpose. Furthermore, already existing libraries can be used to read out the data of the sensors. It operates at a voltage of 3.3V which the microprocessor receives from the USB to UART converter.

2.7.2 Receive Client

One of the ground segments is a software which displays all the data which is received via radio relay (434MHz). It is written in Python with the help of several libraries. One of which is PyQt5. It is used along side with Numpy for the GUI aspect of the software. For the back-end Numpy is also used to store the data that has been received on a hard drive. This system changed a lot compared to the system used in the German CanSat Competition. A SQLite Database is no longer used and got replaced by Numpy's save and load functions. The use of Numpy in the back-end is convenient because there are no high latency times to convert the stored data back to a format the GUI can display. It already has that format this way. At the same time the advantage of portability which SQLite had still exists because Numpy stores everything in files.

Graphical User Interface The UI of the receive client is based on the UI of the German CanSat Competition. The UI was designed with the two main ideas of modularity and overview, which firstly means that the programmer should be able to adjust the UI simply and quickly to adapt to changes on either the primary or secondary mission. Secondly it means that the team should be able to read the most important data at first glance so that there is no delay in getting critical information.
The First UI Design from the German CanSat Competition can be found in the appendix.
The UI used for the European CanSat Competition is going to have different graphs/plots on the left side of the window and a so called "ToolBox"-element which is going to inherit different tabs with different kinds of information. Although the mainly used tab is going to be the overview which shows the previously mentioned important information. Together with a 3D view of our CanSat which shows the current rotation of the can.
The Final UI Design for the European CanSat Competition can be found in the appendix.
Additionally there will be a status bar at the bottom which shows the current time, the time of the mission since launch, the current packet loss, battery status and the ground temperature and pressure.

Data parsing When the ground segment receives the data from the satellite through a serial connection, it first will be handled by the *serial_controller* class. This class will first parse the received data into a structure which is humanly understandable. In other words the bytes that were transmitted are split into different parts which are defined in the self made protocol (See Table 2 Protocol). Then those byte parts are converted into understandable numbers. Those could be floats or integers depending on the type of the package received. After the data has been parsed it will be then forwarded to the *database_controller* class which as the name suggests manages all the data. The *database_controller* will store the forwarded data in a numpy array in memory as well as on the hard drive as a file.

Data structure The data structure for the ground segment looks as follows. In the back-end more specifically the *database_controller* a dictionary containing numpy arrays for different types to store the data is used. Those types are temperature, pressure, height and many more. If the GUI then wants the dataset for a given type the database_controller can then return the dataset with a complexity of O(1). The *database_controller* also allows to get only the last stored dataset for a given type as well as all the datasets that were stored since the last request. Furthermore, there is also the option to get the last x datasets for a given type with the complexity of O(n). It is even possible to get the datasets from x till y for the given type with a complexity of O(n).

Data calculations Another task the ground segment has is the calculation of values like the height of the satellite, battery percentage and the packet-loss. The GPS also needs to be converted in a different format. Furthermore the temperature needs to be converted from Kelvin to Celsius and from Kelvin to Fahrenheit. In addition to that the total velocity, acceleration and distance has to be calculated.

For the calculation of the height the following formula is used[11]:

$$h = \frac{((\frac{P_0}{P})^{\frac{1}{5.257}} - 1) \times T}{0.0065}$$

It uses the pressure on the ground as well as the pressure and temperature from the satellite. The pressure is given in Pascal and the temperature in Kelvin. This formula originally used the temperature in degrees Celsius so it has been modified to use Kelvin instead.

Since the GPS data which is received from the satellite is in the DDM[12] format it needs to converted it into the DD[13] format so it is easier to communicate with the GoogleMaps API. For this conversion the following formula was used:

$$dd = f(ddm) = \lfloor \frac{ddm}{100} \rfloor + \frac{ddm - (\lfloor \frac{ddm}{100} \rfloor \times 100)}{60}$$

[11] [1] *Height Formula*
[12] Degree Decimal Minutes
[13] [13] *Decimal Degrees*

This formula uses the DDM value as a float to convert it to the corresponding DD value.
An example would be:

$$\begin{aligned}
f(5304.91118) &= \lfloor\frac{5304.91118}{100}\rfloor + \frac{5304.91118 - (\lfloor\frac{5304.91118}{100}\rfloor \times 100)}{60} \\
&= \lfloor 53.0491118\rfloor + \frac{5304.91118 - (\lfloor 53.0491118\rfloor \times 100)}{60} \\
&= 53 + \frac{5304.91118 - 5300}{60} \\
&= 53 + \frac{0.91118}{60} \\
&= 53.081853
\end{aligned}$$

To calculate the current battery charge a table which translates volts to percentages to determine a rough percentage value is used. To smooth those rough numbers linear interpolation between two table values is performed. The table which translates volts to percentages was created by reading values from our batteries over a time period.

The last value the ground segment calculates is the packet loss. This is done by collecting all the packets that have been sent in the same time frame of one second. It can then be calculated using this formula:

$$loss = (1 - ((packetnumber_{last} - packetnumber_{first} + 1)/packets_{total})) * 100$$

This formula uses the last and first packet numbers that were received in the last second to calculate the total packets that **should** have been received in that time frame. An example would be:

First packet number = 250
Last packet number = 255

The total packets that **should** have been received would be *5* but since the first packet also has to count one more has to be added in order to get the correct number of *6* in total.

$$|\{250, 251, 252, 253, 254, 255\}| = 6 \neq 5$$

This number is then divided by the total packets that were received to get the percentage of received packets between zero and one. To get the percentage that were lost the received once are subtracted from it. Lastly it is multiplied with 100 to get a percentage value from 0 to 100.

To convert the temperature following formulas have been used:[14]

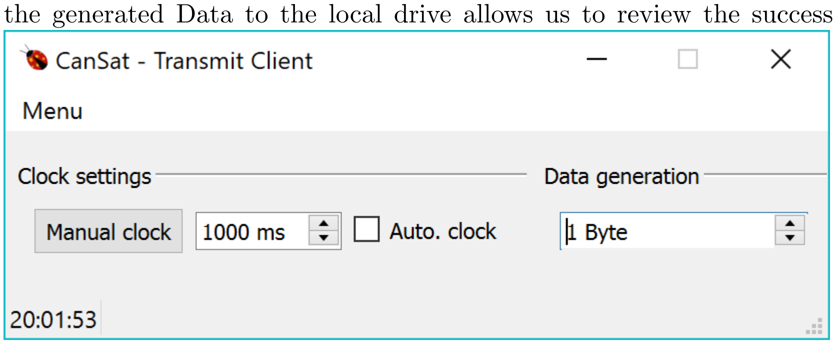
$$T_C = T_K - 273.15$$
$$T_F = (T_K * 1.8) - 459.67$$

To calculate the total velocity, acceleration and distance this formula has been used:

$$x_{total} = \sqrt{x_x^2 * x_y^2 * x_z^2}$$

2.8 Transmit Client

The transmit client is providing an interface for setting the amount of data and the frequency with which it sends to the CanSat. It randomly generates the given amount of bytes, saves them to the local drive and hands over the data to the serial controller which then sends the data via an antenna to the CanSat. Saving the generated Data to the local drive allows us to review the success of the secondary mission post launch.



3 Project planning

3.1 Time schedule of the CanSat preparation

The planning was carried out on the basis of the deadlines and terms specified by the competition. All known deadlines were written into a MS Project document where we planned time frames and deadlines on this basis. In addition, a certain period of time was planned as a buffer for the appointments in order to have sufficient reserves in the event of a time delay. According to these basic rules, the individual areas (preliminary planning, first definition phase, first test phase, development phase, qualification phase, final phase) were planned at the beginning. As the project progressed, especially in the spring of 2018, the planning was repeatedly adjusted to correct some misjudgements in the basic planning, since certain tasks took much less or even a little more time than originally thought.
The planning documentation can be found in the appendix.

[14] [3] *Physik für Ingenieure*

3.2 Resource estimation

3.2.1 Sponsors

Nine sponsors were acquired for the European CanSat Competition 2018, five of them support the project financially with an amount of 3300€.

b.r.m. The first sponsor is the Bremen-based IT service provider b.r.m. It supports the Starbugs team with financial means and provides us with an opportunity to test the satellite at an airfield in Oldenburg-Hatten.

Söffge The second sponsor for the participation in the European CanSat Competition is the family business for cleaning services Söffge in Bremen and Lower Saxony which also supports the project financially in the development of our satellite.

Bremer Rechenzentrum The third sponsor is the nationwide operating billing service provider Bremer Rechenzentrum GmbH which also supports the Starbugs team financially in the development of our satellite. Together with the b.r.m and Söffge they donated 2700€.

Freiplan Ingenieure The participation at the European CanSat Competition is also supported by the Engineering Office for Building Services Freiplan Ingenieure with an amount of 300€.

Straschu Holding GmbH Another sponsor is the company Straschu which helps with manufacturing and assembly of the printed circuit boards.

Technikerschule Bremen The Technikerschule (technical college) Bremen supports the project by professionally printing the satellites shell in 3D.

Hackerspace Bremen The Hackerspace Bremen has supported the project in many ways as far as premises and tools are concerned. Furthermore, the members themselves have been able to support a lot by sharing their experiences with the team.

Conrad The local Conrad Electronic store Bremen also supports the project by providing a few required components for free.

Antares Antares is a German company that develops and designs measuring instruments. The company also supports the project with financial resources.

3.2.2 Expenditures

Some components were ordered twice because they are used in the primary and secondary mission. Furthermore, some components were also damaged and had to be reordered. Some components were also too large and had to be replaced by components with a smaller package. Despite some high Expenses the cost is below the price limit of 500€.

3.2.3 External support

The Hackerspace Bremen proved to be a great helper by making rooms and tools available. However, not only the resources but also the help of the members, especially Oliver Riesner, a university teacher who helped with the microprocessor, were very helpful in the project. Straschu Holding GmbH supported the project by manufacturing the boards and mounting sensors that could not be soldered by hand. They also checked the PCB layout. The company b.r.m. provides a possibility to test our CanSat at the airfield Oldenburg-Hatten. Fenya Haack designed the logo and Linda Müller helped designing the flyer. We would like to thank all the people who have supported us in the course of the project.

3.3 Testing

3.3.1 Ground station

One test that will be done to test the ground station software is to give it some packages to parse and store. After that it will be checked if all the given data was parsed successfully as well as if it was stored correctly. Furthermore tests regarding smoothness, stability and data integrity will be done.

3.3.2 Satellite

Each sensor and module will be tested by using a debugger. For testing the error-codes, wrong I^2C-addresses and UART-interfaces will be used to try to communicate with the modules. The satellite will run for a time of one hour to determine if the program on the satellite runs stable. The current consumption will be measured during the satellite runs to determine how much current is needed by the whole system. A battery test will also be performed to find out for how much time the batteries can supply the satellite.

3.3.3 System

The ground station will be tested with real data sent by the satellite. Both radio links will be tested for their range. A last test in which satellite and ground station are run for long time will be performed.

4 Lessons learnt from the national launch campaign

During the German CanSat Competition we faced a lot of challenges. In the beginning of the national competition we didn't assign a field of work to each team member but during the competition we realised that it's much more effective if everyone becomes an expert for a specific field of work and kept on working in that way during the European CanSat competition.
We found out that the idea of building a CanSat consisting of two modules of which one can be changed was very successful. Because of this we continued working on this and improved the system.
We also learned that planning the CanSat itself before building it needs a lot of time but has the advantage of learning much about the materials, hardware and software used. This can shorten the time needed to build the CanSat and reduce costs because less miss buys are done. For our future life this also has the advantage of a bigger knowledge for further projects.
Lessons we learned in the technical parts of the project are already mentioned there.

5 Public relation

The public relations work is carried out on various multimedia platforms for the CanSat competition. These include Facebook, YouTube, Instagram, Twitter and a website. Often posts are uploaded to these social media platforms, which are explained in more detail on the website. Furthermore, the CanSat competition is explained, the team introduced, the satellite described in detail and the missions explained on the website.

The blog entries should keep readers up to date with the project itself and the development of the CanSat. Especially for the time of the ongoing competition regular posts should be published on all social media platforms.

In addition, an author contract was signed with heise.de to write monthly texts about the project, but despite sending the texts monthly, no single publication was made because the employees had too much to do with other articles.
At the moment a flyer is being worked on which will be placed in different schools and in the Hackerspace to reach even more people with this project. The flyer will explain the project itself, the team and the secondary mission.

5.1 Website

For the German CanSat competition a free WordPress website was created, so that several users can access the website and publish articles. WordPress also supports various designs. To keep these features a own domain was bought and a new WordPress website was created on a webserver from Colorhost. The aim of the website is to bring the satellite and its development closer to the audience. 1,424 visitors with 5,884 views (to date: 10.06.18) were on the website.

Link: https://teamstarbugs.de/?page_id=1083&lang=en

Blog: https://teamstarbugs.de/?page_id=384&lang=en

5.2 Facebook

Another public relations idea was to create a Facebook page. Small contributions are regularly uploaded on this page to make other interested parties familiar with the course of the competition. With the Facebook page 21 subscribers and a 3,039 people were reached. The site itself has been liked 20 times.

Link: https://www.facebook.com/TeamStarbugs/

5.3 Instagram

Since Instagram is becoming more and more important as a social media platform, a Instagram account was created. Under @teamstarbugs small picture and video contributions are uploaded to increase the interest in the competition. 77 subscribers were reached who see our contributions and often also Like them.

Link: https://www.instagram.com/teamstarbugs/

5.4 Twitter

For the European CanSat competition the idea came up to spread the public relations work also on Twitter. However, posts were not uploaded so actively that only 14 followers have been reached so far.

Link: https://twitter.com/TeamStarbugs/

5.5 School and other public relations

Some presentations at our school were held. This includes the German MINT Committee, on whose website we were mentioned.

Link: https://www.mint-ec.de/aktuelles/blog-facebook/landessieger-im-cansat-deutschland-wettbewerb/

But not only the committee also in classes several presentations were held to bring the project closer to the students. One class was so enthusiastic about the project that they will participate as planet@earth in the German Competition 2018.

At the Technikerschule Bremen, one of our sponsors, which prints our can a presentation was also held in front of the responsible class.
The Starbugs team was also mentioned on various websites:

- https://www.focus.de/regional/bremen/raumfahrt-bremer-schueler-gewinnen-wettbewerb-mit-mini-satelliten_id_7657898.html
- https://www.dlr.de/dlr/desktopdefault.aspx/tabid-10255/365_read-24461/
- https://www.szut.de/aktuelles/mint-projekte/cansat-2017.html
- https://www.hackerspace-bremen.de/2017/10/02/bremer-team-starbugs-gewinnt-deutschen-cansat-wettbewerb-2017/

6 Requirements

Characteristics	Figure (units)
Height of the CanSat	114 (mm)
Mass of the CanSat	300 (g)
Diameter of the CanSat	65 (mm)
Length of the recovery system	3 (cm)
Flight time scheduled	111 (s)
Calculated descent rate	9 (m/s)
Radio frequency used (1)	434 (MHz)
Radio frequency used (2)	869 (MHz)
Power consumption	1.472 (W)
Total cost	451,67 (€)

Table 3: Requirements

On behalf of the team I confirm that our CanSat complies with all the requirements established for the 2018 European CanSat Competition in the official guidelines.

Marian Kirovaschenko

References

[1] LTD. CASIO COMPUTER CO. *Height Formula*. URL: https://keisan.casio.com/has10/SpecExec.cgi?id=system/2006/1224585971.

[2] HopeRF Electronic. *HMTRP Datasheet*. URL: http://www.hoperf.de/upload/rf_app/HM-TRP.pdf.

[3] Ekbert Hering, Rolf Martin, and Martin Stohrer. *Physik für Ingenieure*. VDI-Verlag GmbH, 1992.

[4] Melexis. *MLX90614 Datasheet*. URL: https://www.sparkfun.com/datasheets/Sensors/Temperature/MLX90614_rev001.pdf.

[5] Mentor. *Miniatur-Schlüsselschalter*. URL: http://www.produktinfo.conrad.com/datenblaetter/175000-199999/183549-da-01-ml-Schluesselschalter_R_R_1859_1031_de_en.pdf.

[6] Inc. PUI Audio. *AUDIO MAGNETIC XDCR 2-4V TH*. URL: http://www.puiaudio.com/pdf/AT-1127-ST-2-R.pdf.

[7] Bosch Sensortec. *BME280 Datasheet*. URL: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME280_DS001-12.pdf.

[8] Bosch Sensortec. *BNO055 Datasheet*. URL: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST_BNO055_DS000_14.pdf.

[9] Bosch sensortec. *BME280 Driver*. URL: https://github.com/BoschSensortec/BME280_driver.

[10] Bosch sensortec. *BNO055 Driver*. URL: https://github.com/BoschSensortec/BNO055_driver.

[11] st. *STM32F413RH Datasheet*. URL: http://www.st.com/content/ccc/resource/technical/document/datasheet/group3/f6/1d/4a/d9/f9/96/43/2a/DM00282249/files/DM00282249.pdf/jcr:content/translations/en.DM00282249.pdf.

[12] u-blox. *UBLOX NEO M8 Datasheet*. URL: https://www.u-blox.com/sites/default/files/NEO-M8-FW3_DataSheet_%5C%28UBX-15031086%5C%29.pdf.

[13] Inc. Wikimedia Foundation. *Decimal Degrees*. URL: https://en.wikipedia.org/wiki/Decimal_degrees.

A Tables

Name	Data	Voltage (V)	Current (mA)	Protocol
STM32F413		3.3	5.6	
BME280	Environment	3.3	1	I^2C
BNO055	IMU	3.3	12.3	I^2C
MLX90614	Temperature (infrared)	3.3	20	I^2C
GPS	Position	3.3	32	UART
SD-Card	Save data	3.3	100	SDIO
HMTRP433	Send data	3.3	50	UART
Power LED	Indicate supply voltage		15	
Heartbeat LED	Indicate running μC		15/2	
Sum			243.4	

Table 4: Modules primary module

Name	Data	Voltage (V)	Current (mA)	Protocol
STM32F413		3.3	5.6	
SD-Card	Save data	3.3	100	SDIO
HMTRP868	Send data	3.3	30	UART
Power LED	Indicate supply voltage		15	
Heartbeat LED	Indicate running μC		15/2	
Sum			158.1	

Table 5: Modules secondary module

Packet type	Frequency	size per package (Byte)	Total size per s (Byte)
GPS	1Hz	36	36
ADC	20Hz	17	340
BME280/MLX90614	20Hz	31	620
BNO055	20Hz	63	1260
Total			2256

Table 6: Amount of data generated by primary module

Packet type	Frequency	size per package (Byte)	Total size per s (Byte)
Radio relay	100Hz	30	3000
ADC	20Hz	17	340
Total			3340

Table 7: Amount of data generated by secondary module

blue	orange	
slow	off	MLX90614 error
slow	slow	BME280 error
slow	fast	BNO055 error
slow	on	I^2C error
off	slow	GPS error
off	fast	Secondary module error
off	on	SD error
on	on	error
off	off	no error

Table 8: Error codes primary module

blue	orange	
slow	off	Primary module error
slow	on	radio error
on	on	error
off	off	no error

Table 9: Error codes secondary module

Number	Binary	Type
0	0000 0000	Environment
1	0000 0001	IMU
2	0000 0010	GPS
3	0000 0011	Akku

Table 10: Packettypes primary module

Number	Binary	Type
16	0001 0000	Radio relay
17	0001 0001	Akku

Table 11: Packettypes secondary module

Name	Amount	Cost per	Costs in Satellite
Diode Schottky 20V 500MA 0805	2	0,32€	0,64€
Super Capacitor 0.22F 5.5V Radial	1	2,07€	2,07€
MLX90614 Temperature Sensor	1	12,14€	12,14€
LED ORANGE CLEAR 0805 SMD	2	0,19€	0,38€
CRYSTAL 32.768KHZ 12.5PF SMD	2	0,63€	1,26€
CONN MICRO SD CARD HINGED TYPE	2	0,91€	1,82€
CONN GH HOUSING 4POS 1.25MM	4	0,10€	0,40€
CONN HEADER GH TOP 4POS 1.25MM	4	0,34€	1,36€
CRYSTAL 8.000 MHZ 18PF SMD	2	0,50€	1,00€
SENSOR PRESSURE HUMIDITY TEMP	1	6,54€	6,54€
CONN GH HOUSING 6POS 1.25MM	2	0,13€	0,26€
CONN HEADER GH TOP 6POS 1.25MM	2	0,43€	0,86€
CONN GH HOUSING 5POS 1.25MM	2	0,13€	0,26€
CONN HEADER GH TOP 5POS 1.25MM	2	0,36€	0,72€
LED BLUE CLEAR 0805 SMD	2	0,22€	0,44€
LED RED DIFFUSED 0805 SMD	2	0,21€	0,42€
LED GREEN DIFFUSED 0805 SMD	2	0,20€	0,40€
Capasitor CER 1.2UF 10V X7R 0805	2	0,40€	0,80€
Didode Schottky 20V 500MA SOD123	2	0,24€	0,48€
Capasitor CER 1UF 16V X7R 0805	2	0,07€	0,14€

Capasitor CER 4.7UF 10V X7R 0805	2	0,43€	0,86€
Coil 470NH 200MA 500 MOHM	2	0,15€	0,30€
Capasitor ALUM POLY 10UF 20% 50V SMD	2	1,17€	2,34€
Microprozessor IC MCU 32BIT 1.5MB FLASH 64LQFP	2	8,16€	16,32€
Capasitor ALUM 47UF 20% 16V SMD	2	0,33€	0,66€
Coil 2.2UH 4.15A 16.5 MOHM	2	0,99€	1,98€
Buzzer Audio magnetic XDCR 2-4V TH	1	0,69€	0,69€
HM-TRP 433 MHz	1	8,50€	8,50€
HM-TRP 868 MHz	1	9,35€	9,35€
GPS UBLOX Neo m8	1	29,15€	29,15€
Murata GRM Ceramikcapasitor X5R, 47uF ±10% / 10 V, SMD	2	0,93€	1,86€
Vishay SMD Resistor, Serie CRCW 10k ±1% 0,125W 0805	2	0,04€	0,08€
Kemet C Ceramikcapasitor X7R, 100nF ±10% / 25 V dc	4	0,03€	0,12€
Ceramikcapasitor 27pF 0603	4	0,06€	0,24€
Crimpconnector GST GH SSHL	54	0,05€	2,70€
Ceramikcapasitor 36pF 0603	2	0,04€	0,08€
DIN 965 TX A2 M2x6 Bolt	5	0,06€	0,30€
DIN 562 A2 M2 Nuts	4	0,18€	0,72€
DIN 7985 TX A2 M1,6x6 Bolt	6	0,19€	1,14€
Keyswitch	2	5,82€	11,64€
BNO055	1	9,51€	9,51€
LP603449 LiPo Akku	2	7,75€	15,50€
LP130450 LiPo Akku	6	13,70€	82,20€
TPS565201DDCR	2	1,52€	3,04€
Rounded costs PCB	2	100,00€	200,00€
Rounded costs shell	1	10,00€	10,00€
Rounded costs parachute	1	10,00€	10,00€
Total Costs			451,67€

Table 12: Expenses

B Pictures

B.1 Primary Module

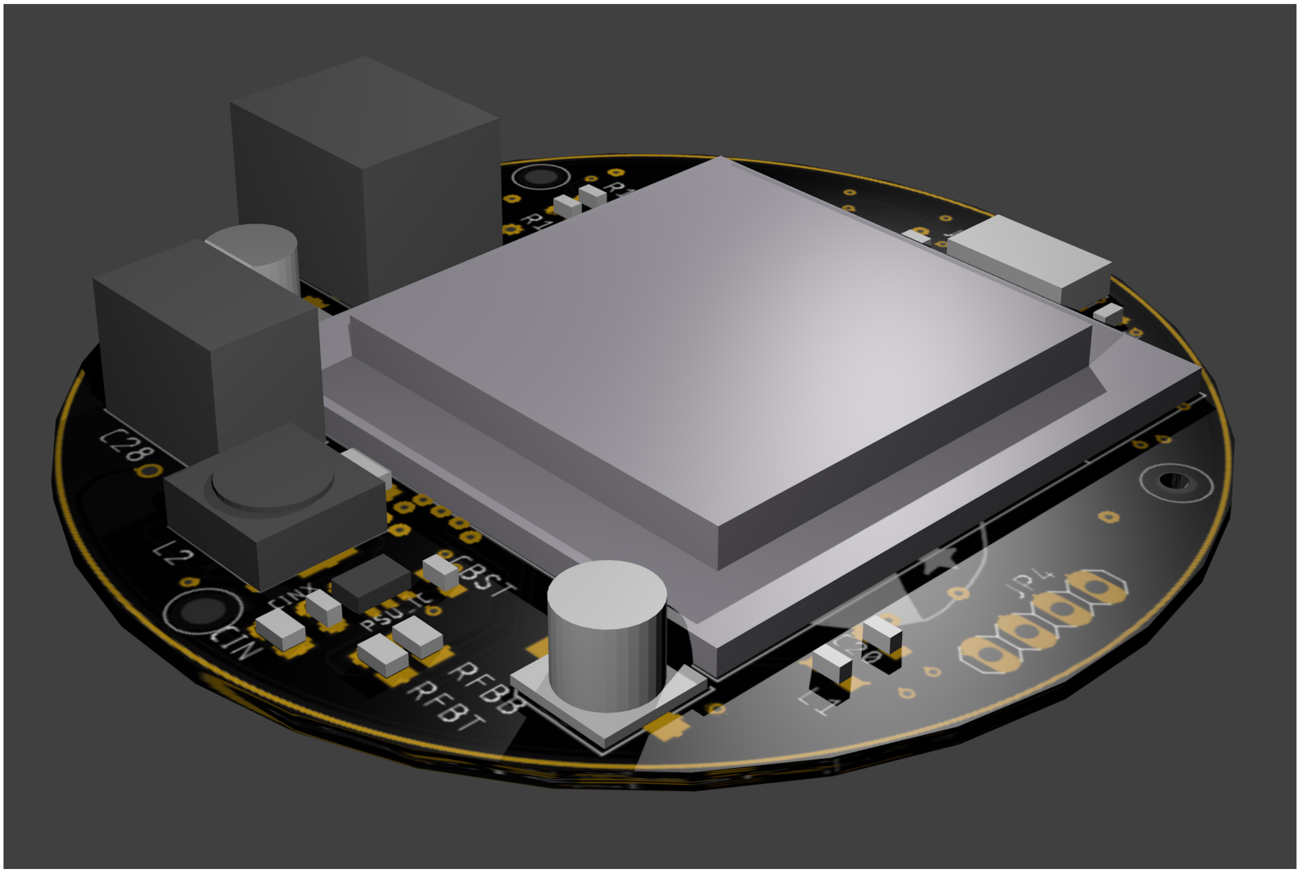


Figure 17: Primary top rendering

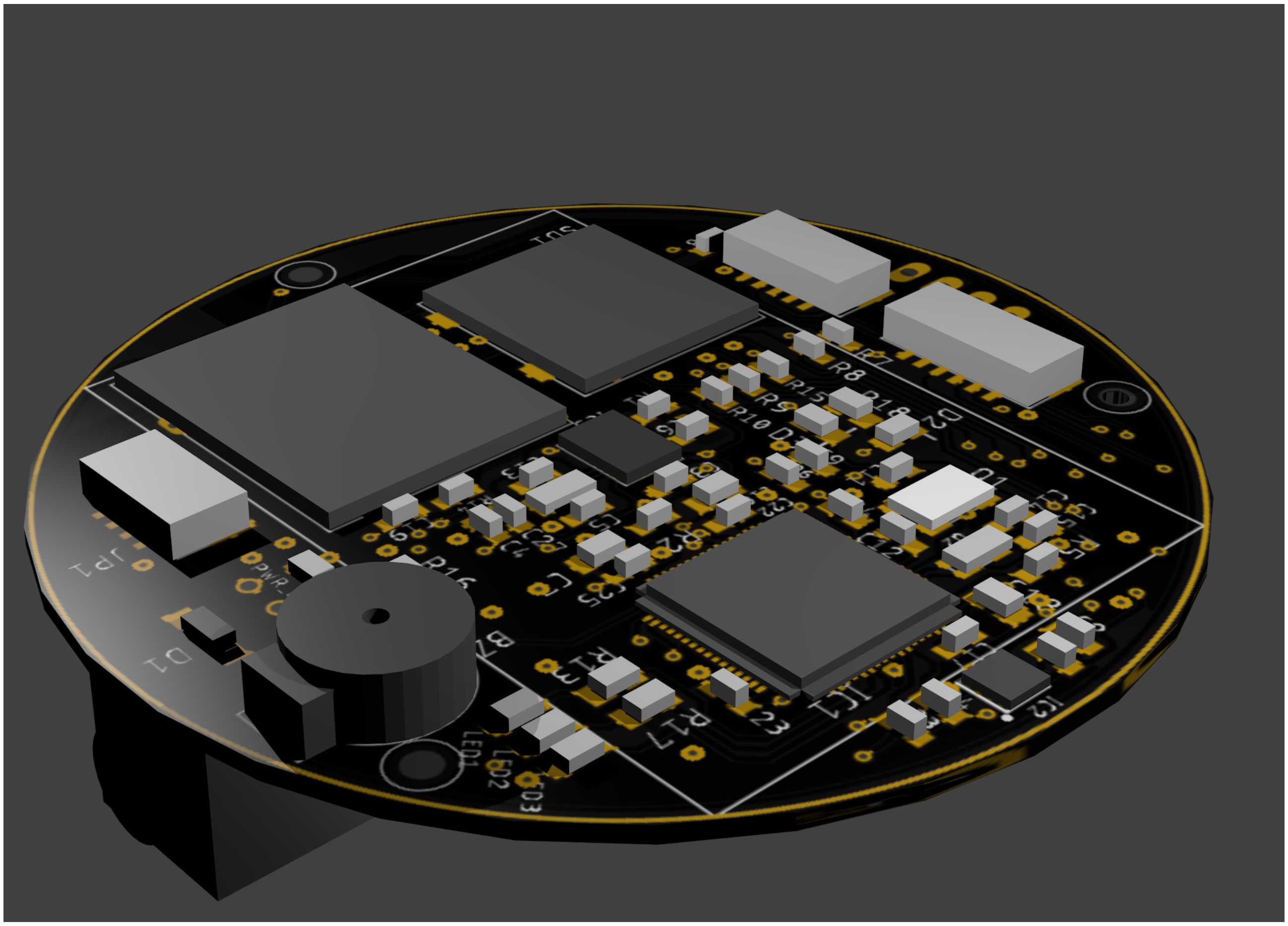


Figure 18: Primary bottom rendering

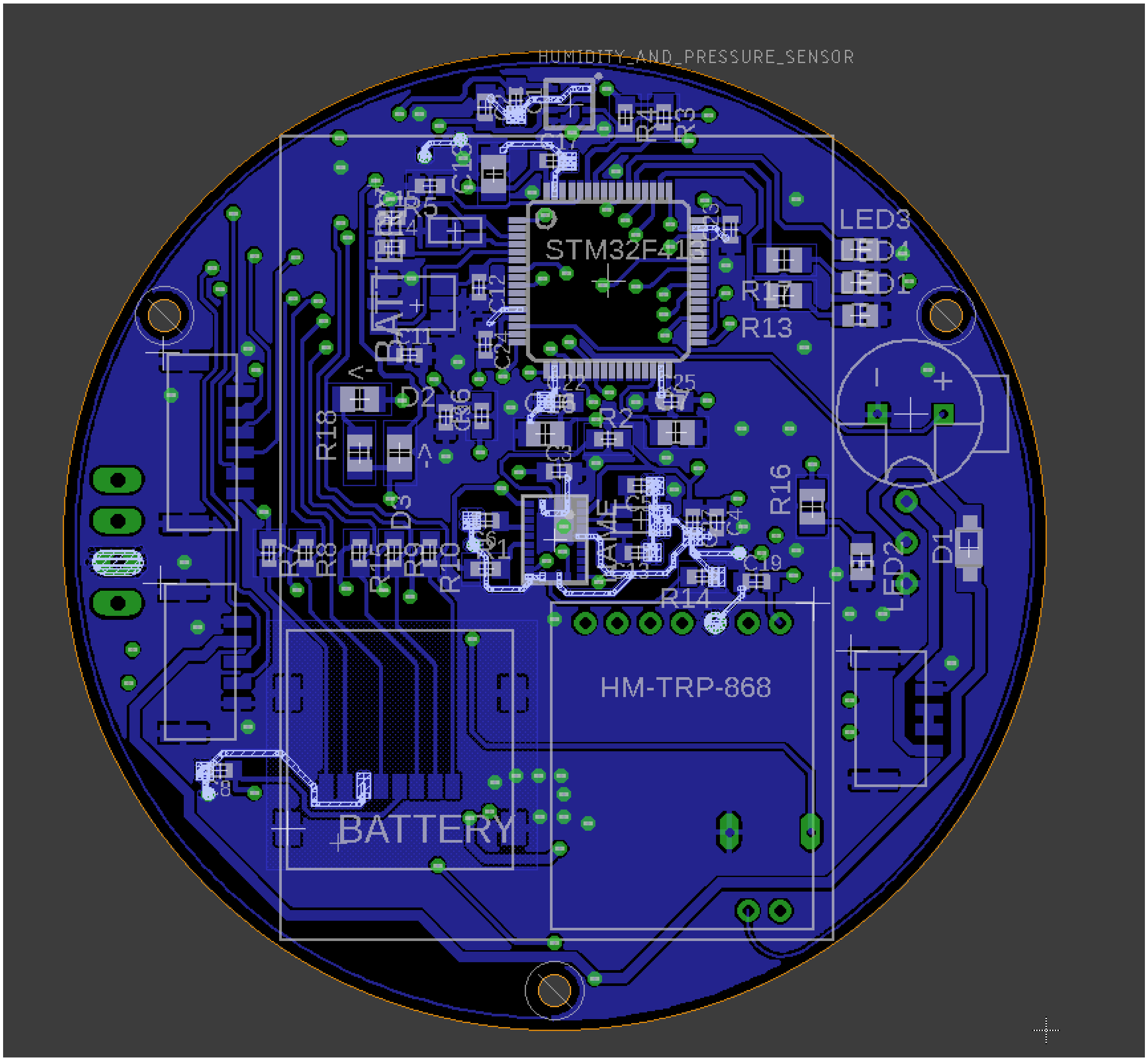


Figure 19: Primary bottom

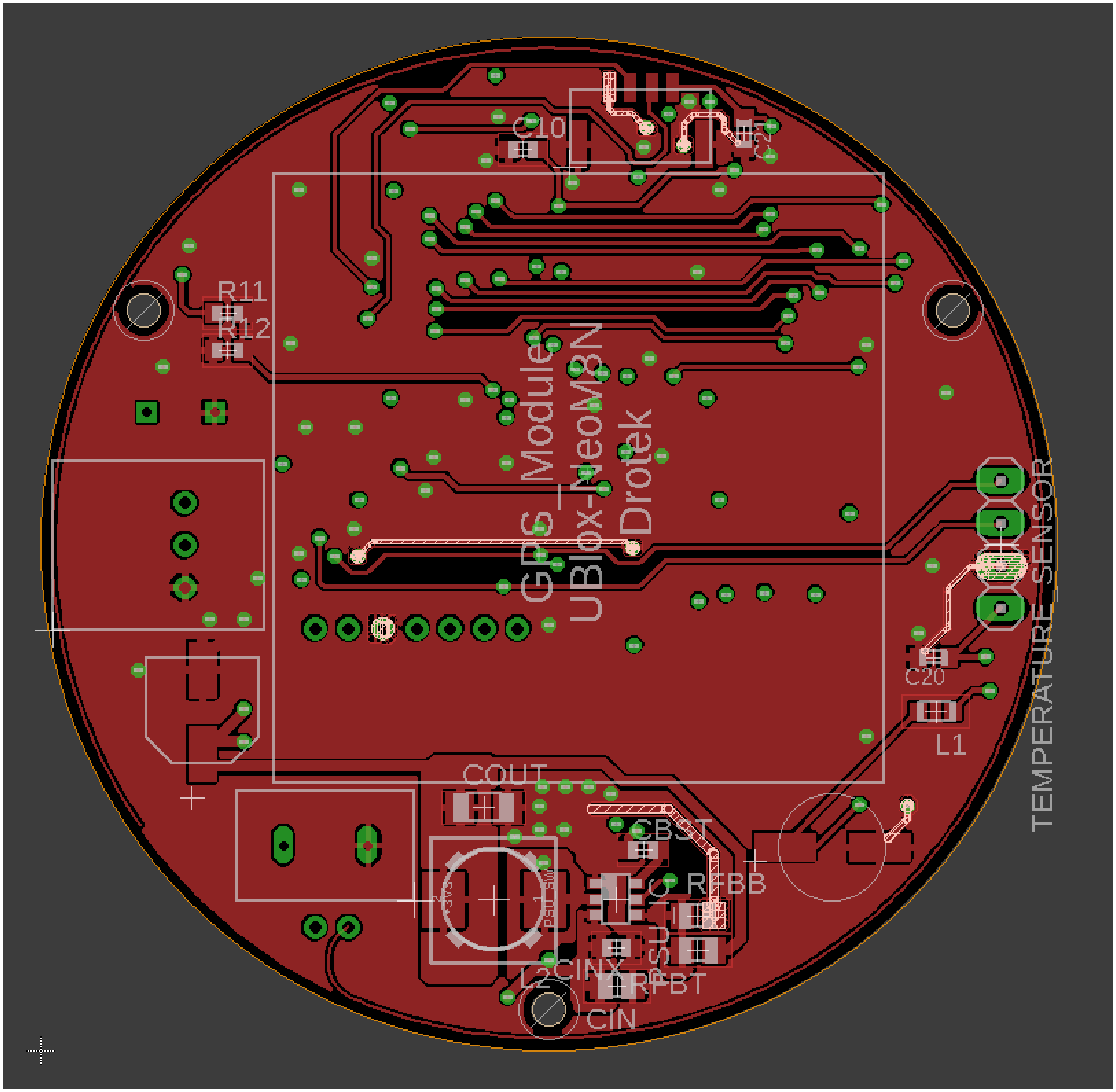


Figure 20: Primary top

B.2 Secondary Module

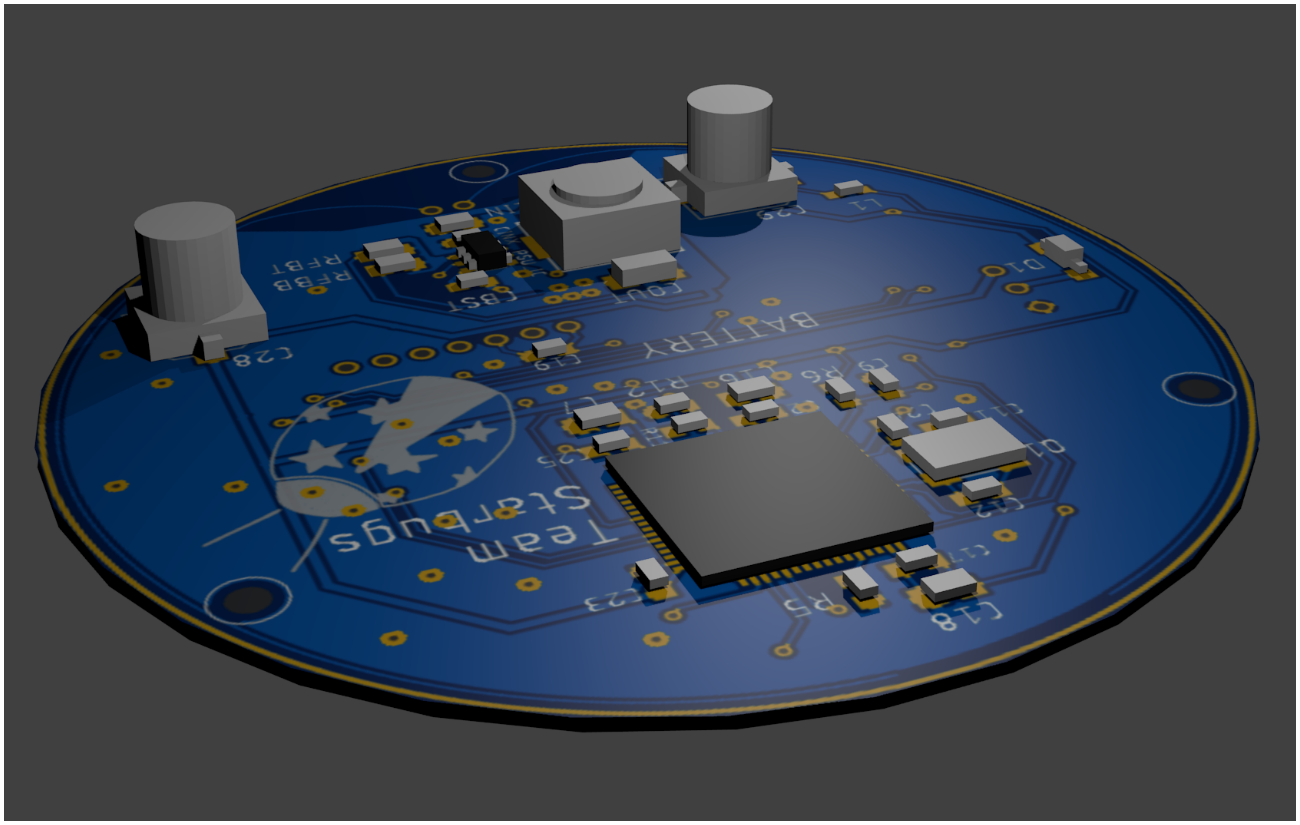


Figure 21: Secondary top rendering

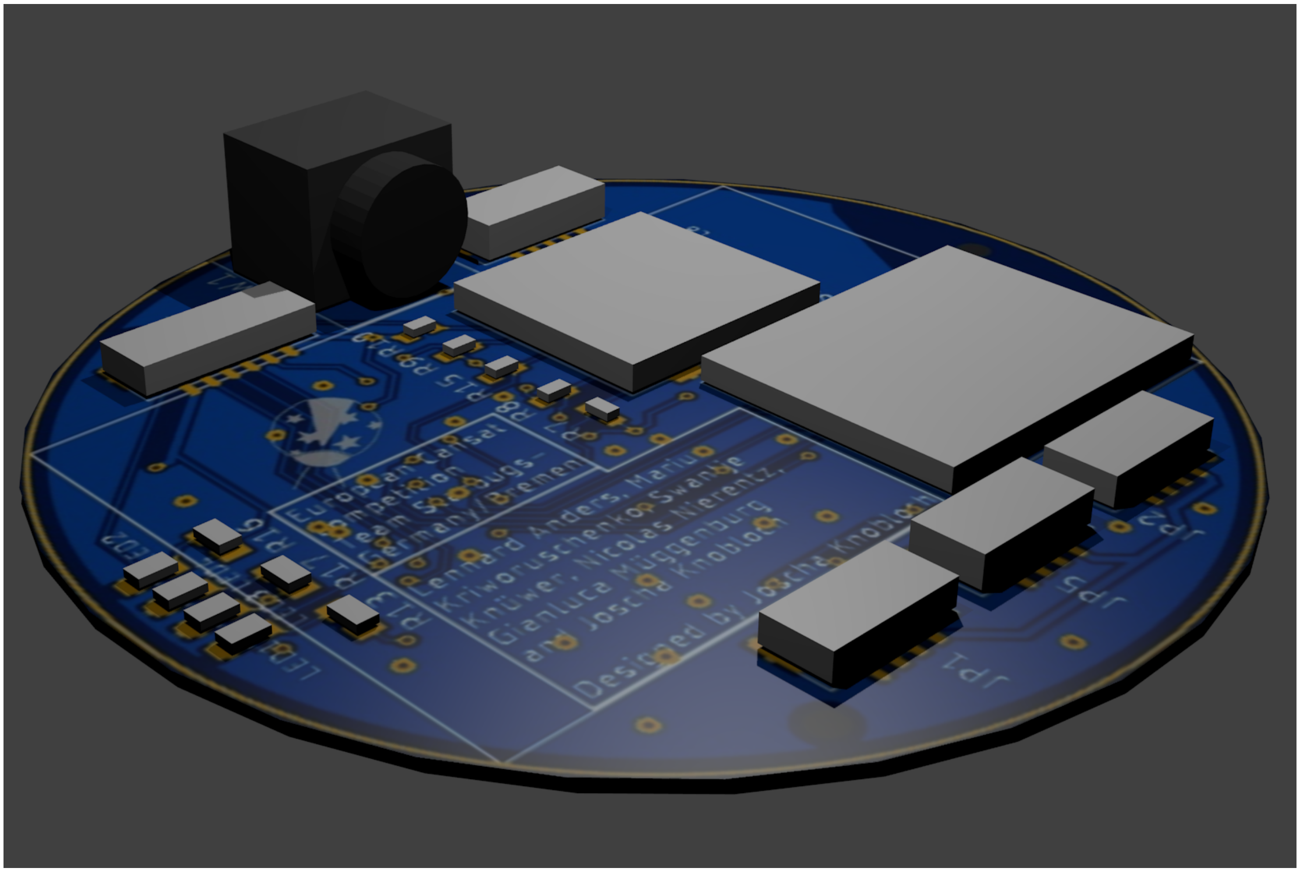


Figure 22: Secondary bottom rendering

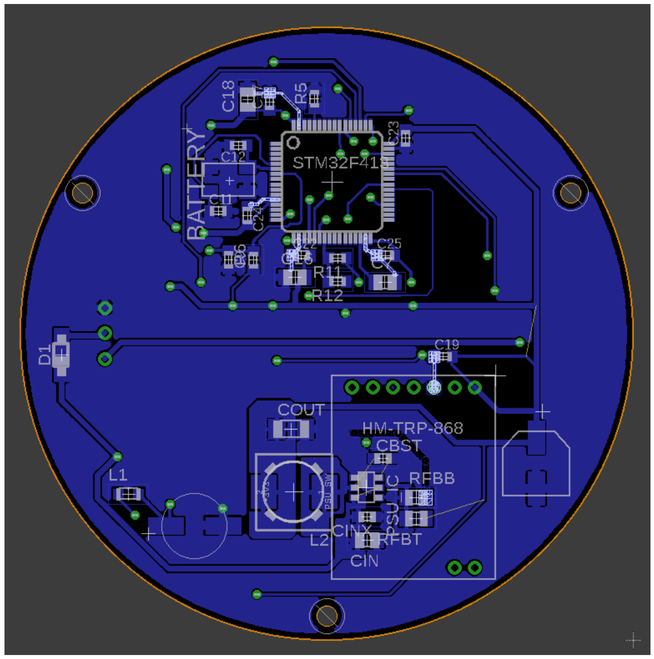


Figure 23: Secondary bottom

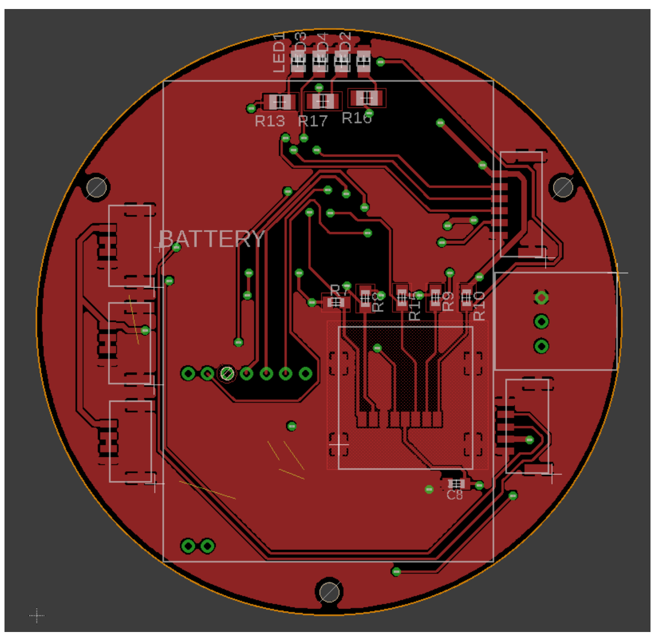


Figure 24: Secondary top

B.3 Power supply

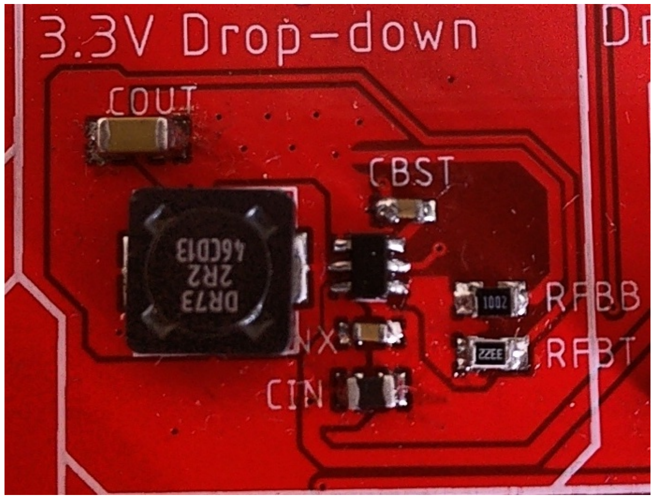


Figure 25: Power supply

B.4 Ground station Receive

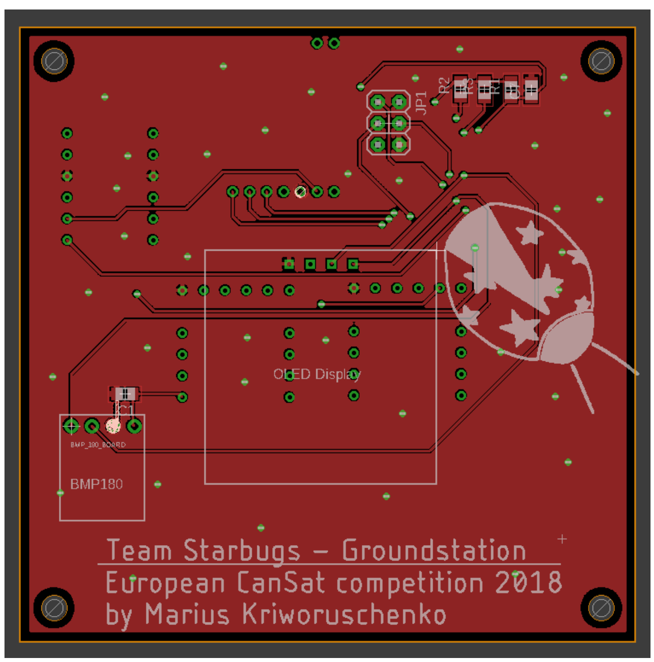


Figure 26: Receive PCB ground station top side

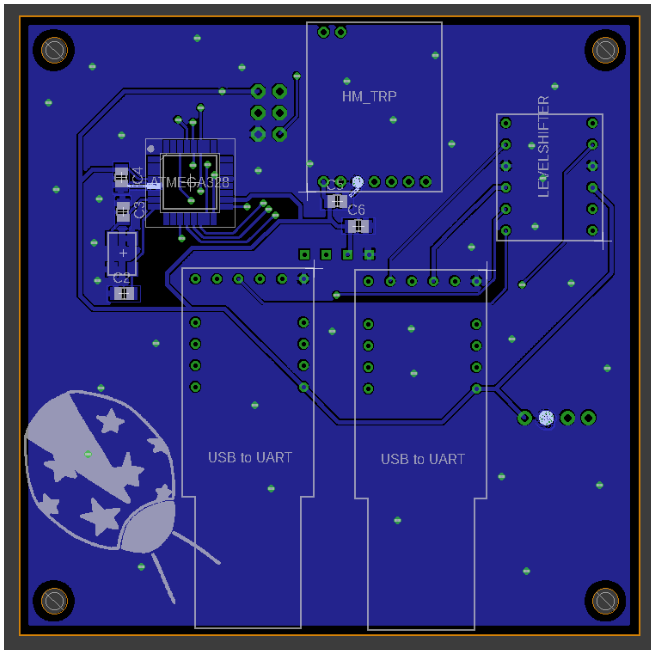


Figure 27: Receive PCB ground station bottom side

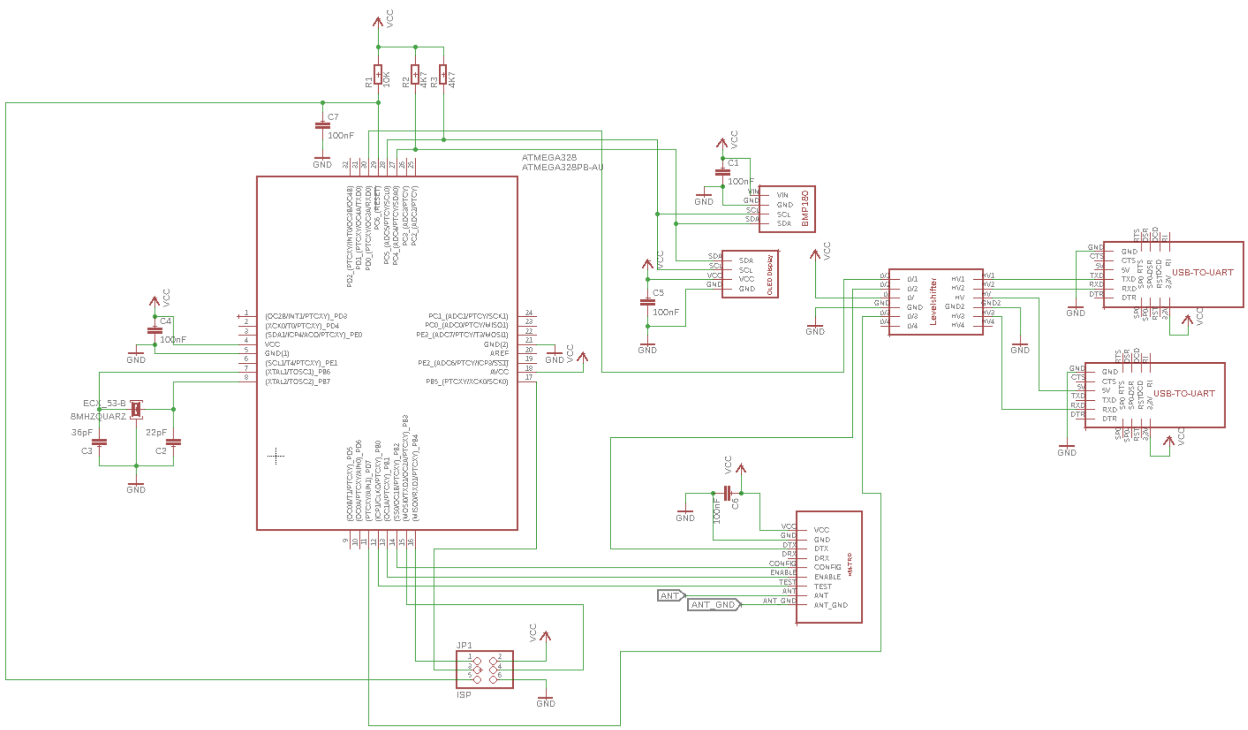


Figure 28: Receive PCB Ground schematic

B.5 Ground station Transmit

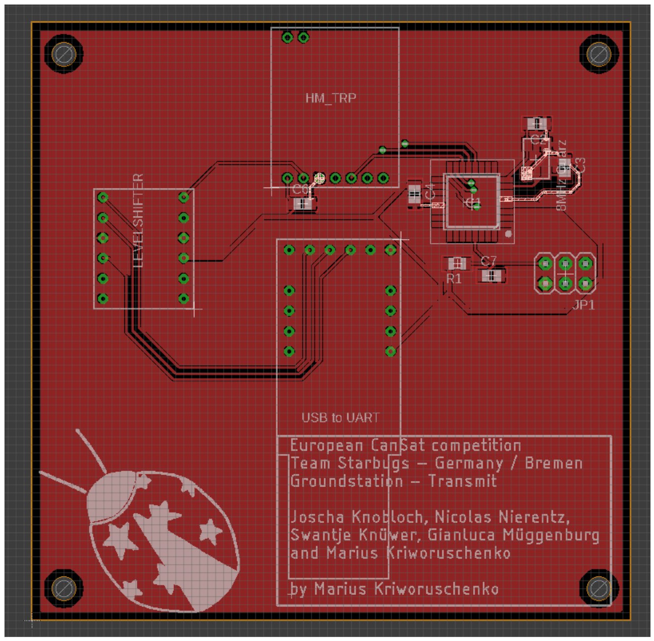


Figure 29: Transmit PCB Ground station top side

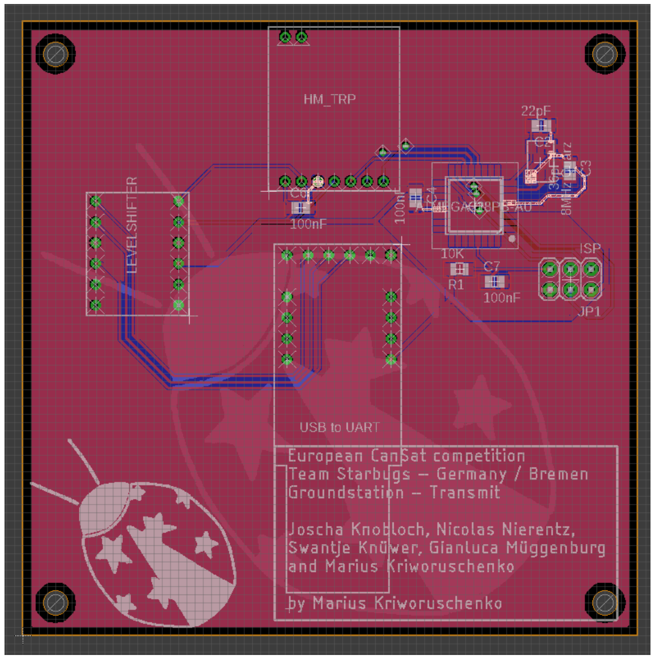


Figure 30: Transmit PCB Ground station bottom side

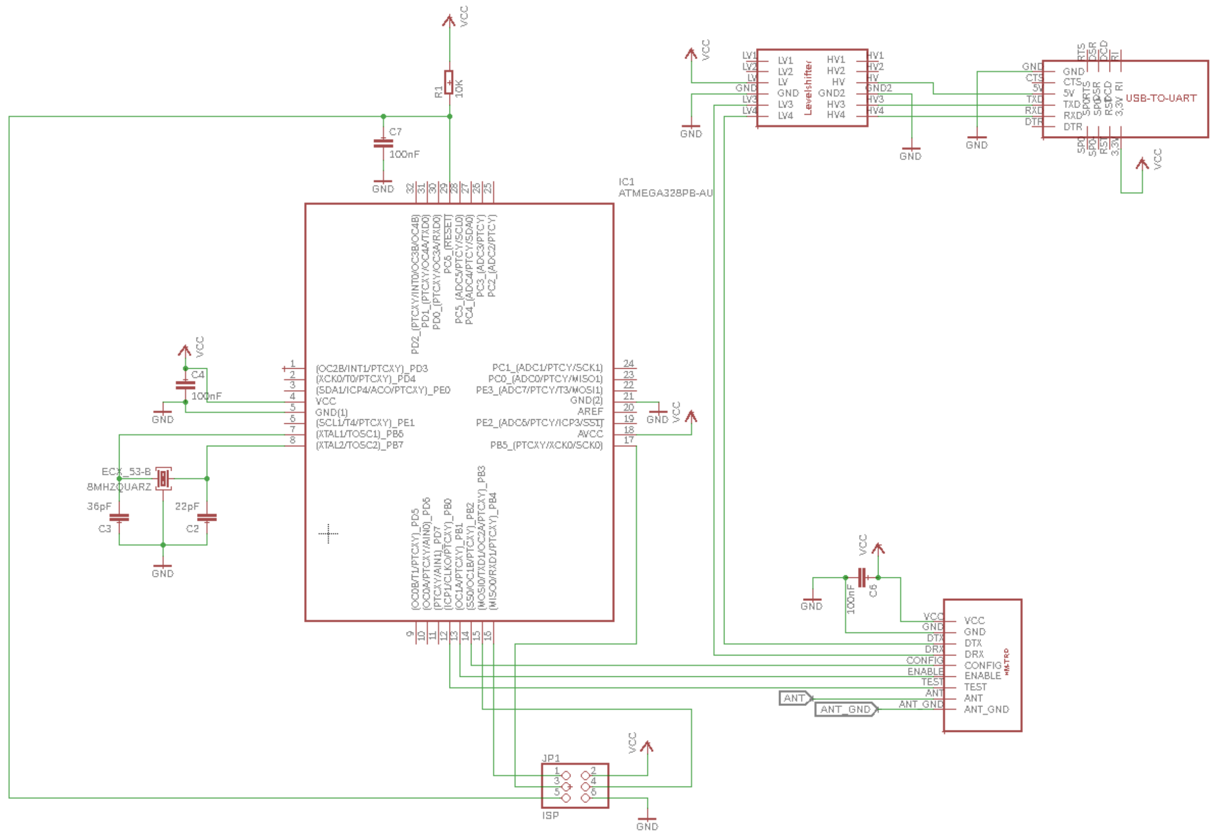


Figure 31: Transmit PCB Ground schematic

B.6 Battery charger

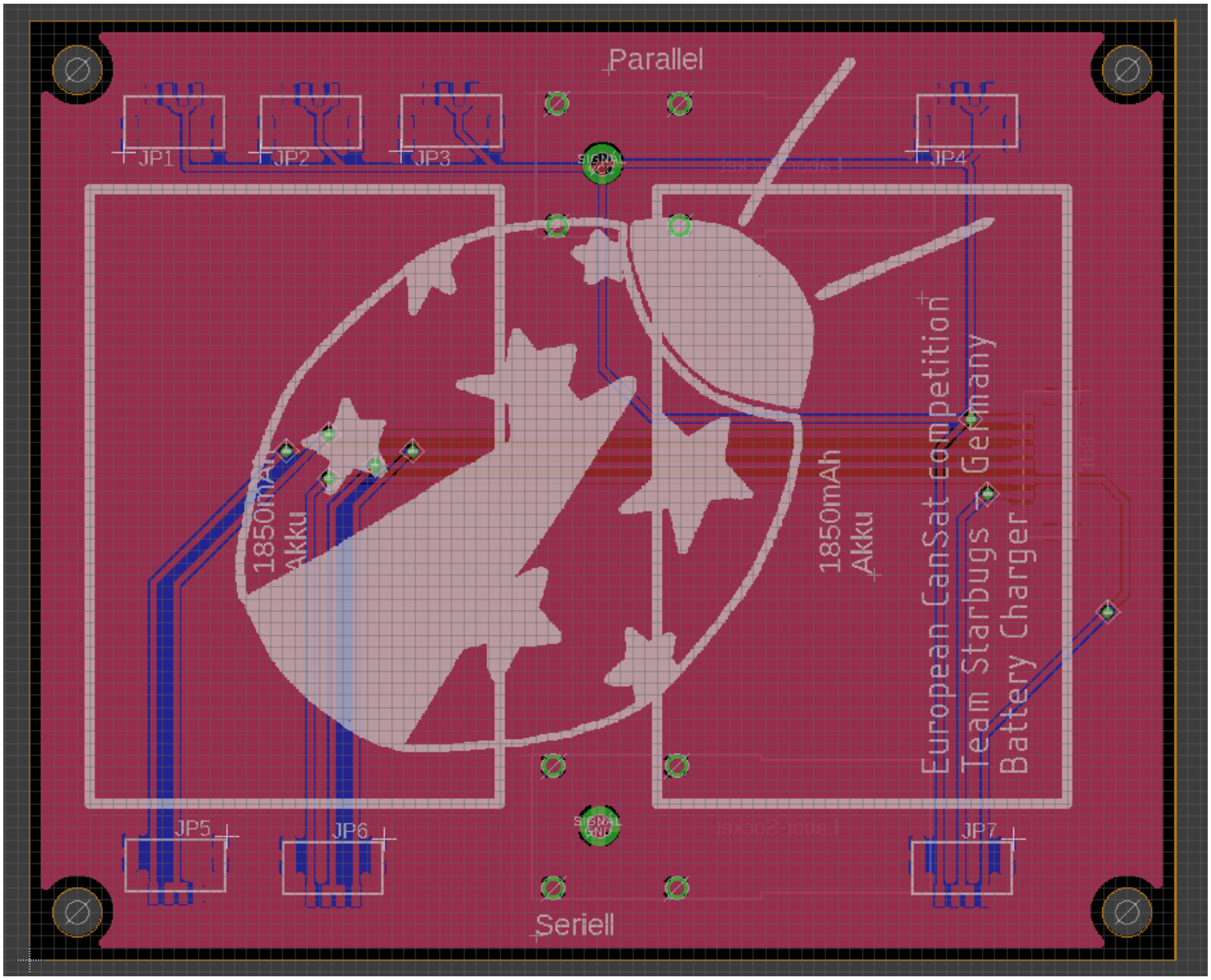


Figure 32: The Battery Charger

B.7 GUI

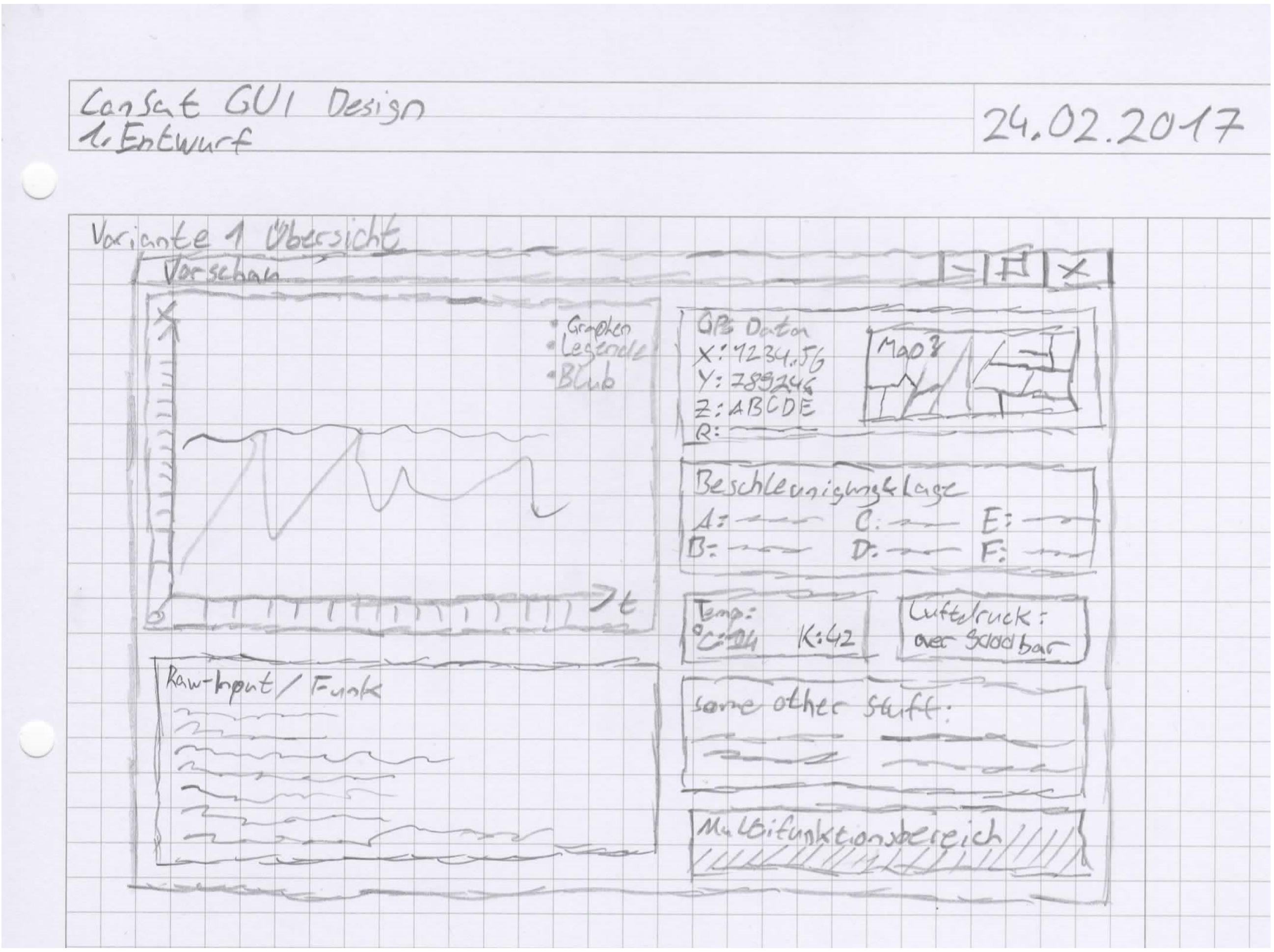


Figure 33: First UI Design from the German CanSat Competition

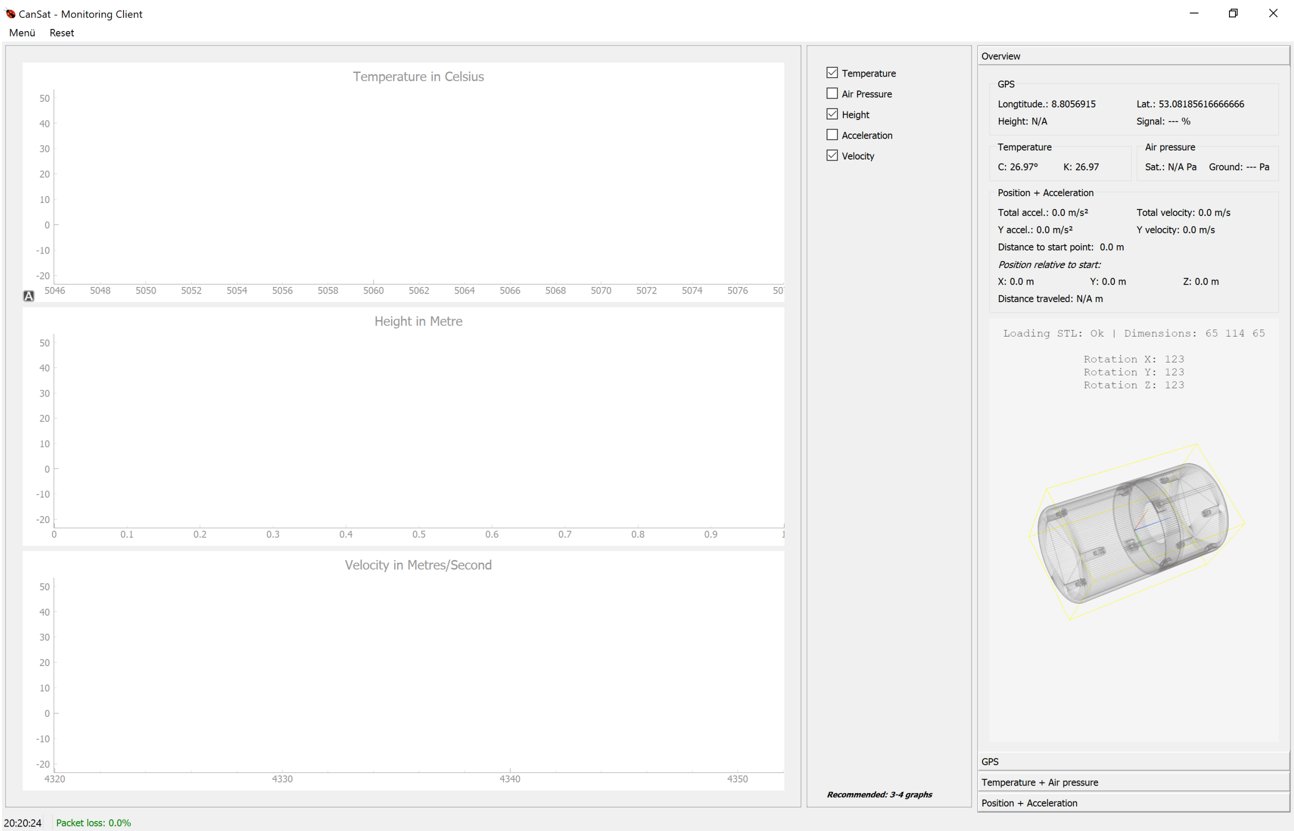


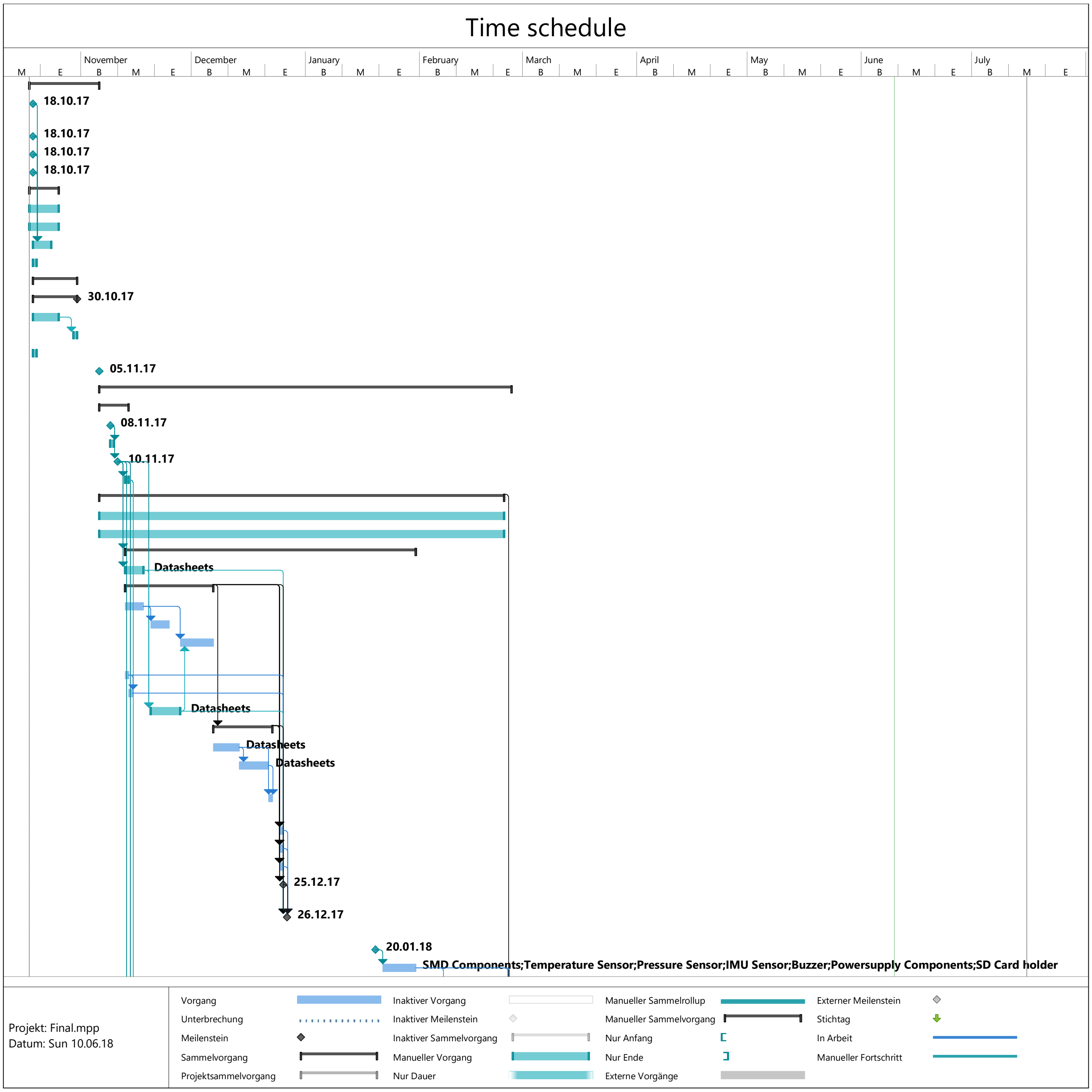
Figure 34: Final UI Design for the European CanSat Competition

Time schedule

ID		Task Mode	Vorgangsname	Duration	Start	Finish	Predecessors	October B
1			**1 Preliminary planning**	**13 dys**	**Wed 18.10.17**	**Sun 05.11.17**		
2			1.1 Problem analysis from the german CanSat competition	1 dy	Wed 18.10.17	Wed 18.10.17		
3			1.2 Definition of goals	1 dy	Wed 18.10.17	Wed 18.10.17		
4			1.3 Perform risk analysis	1 dy	Wed 18.10.17	Wed 18.10.17		
5			1.4 Allocation of the individual areas of responsibility	1 dy	Wed 18.10.17	Wed 18.10.17		
8			**1.7 Create a financial plan**	**6 dys**	**Wed 18.10.17**	**Wed 25.10.17**		
9			1.7.1 Considering sponsors	6 dys	Wed 18.10.17	Wed 25.10.17		
10			1.7.2 Organize travel expenses	6 dys	Wed 18.10.17	Wed 25.10.17		
6			1.5 Create a schedule	3 dys	Thu 19.10.17	Mon 23.10.17	4;3;2;5	
7			1.6 Create documentation	1 dy	Thu 19.10.17	Thu 19.10.17		
11			**1.8 Public relations**	**8 dys**	**Thu 19.10.17**	**Mon 30.10.17**		
12			**1.8.1 Creating a website**	**8 dys**	**Thu 19.10.17**	**Mon 30.10.17**		
13			1.8.1.1 Setting up a Webserver	5 dys	Thu 19.10.17	Wed 25.10.17		
14			1.8.1.2 Buy a domain	1 dy	Mon 30.10.17	Mon 30.10.17	13	
15			1.8.2 Create a Twitter account	1 dy	Thu 19.10.17	Thu 19.10.17		
16			1.9 Completion of preliminary planning	0 dys	Sun 05.11.17	Sun 05.11.17		
17			**2 First definition phase**	**80 dys**	**Mon 06.11.17**	**Mon 26.02.18**		
18			**2.1 Basics**	**6 dys**	**Mon 06.11.17**	**Mon 13.11.17**		
19			2.1.1 Selection of sensors	3 dys	Mon 06.11.17	Wed 08.11.17		
20			2.1.2 Ordering the sensors on a finished development board	1 dy	Thu 09.11.17	Thu 09.11.17	19	
21			2.1.3 Selection of the microprocessor	2 dys	Thu 09.11.17	Fri 10.11.17	19	
22			2.1.4 Ordering the development boards	1 dy	Mon 13.11.17	Mon 13.11.17	21	
51			**2.4 Software ground station**	**80 dys**	**Mon 06.11.17**	**Fri 23.02.18**		
52			2.4.1 Reworking the GUI	80 dys	Mon 06.11.17	Fri 23.02.18		
53			2.4.2 Reworking the database	80 dys	Mon 06.11.17	Fri 23.02.18		
31			**2.3 Create test board**	**57 dys**	**Mon 13.11.17**	**Tue 30.01.18**	21	
32			2.3.1 Search the microprocessor data sheets	5 dys	Mon 13.11.17	Fri 17.11.17	21	
34			**2.3.3 Designing own power supply**	**18 dys**	**Mon 13.11.17**	**Wed 06.12.17**		
35			2.3.3.1 Calculation of requirements	5 dys	Mon 13.11.17	Fri 17.11.17		
36			2.3.3.2 Select and order components	5 dys	Mon 20.11.17	Fri 24.11.17	35	
37			2.3.3.3 Integrate the power supply unit on the test board	7 dys	Tue 28.11.17	Wed 06.12.17	35;33	
38			2.3.4 Selection of the connector	1 dy	Mon 13.11.17	Mon 13.11.17		
39			2.3.5 Ordering of the selected connector	1 dy	Tue 14.11.17	Tue 14.11.17	38	
33			2.3.2 Circuit of the microprocessor	6 dys	Mon 20.11.17	Mon 27.11.17	21;32	
40			**2.3.6 Sensors**	**12 dys**	**Thu 07.12.17**	**Fri 22.12.17**	**34**	
41			2.3.6.1 Search sensor data sheets	5 dys	Thu 07.12.17	Wed 13.12.17		
42			2.3.6.2 Integrating your own circuit on the test board	6 dys	Thu 14.12.17	Thu 21.12.17	41	
43			2.3.6.3 Ordering the components for wiring the sensors	1 dy	Fri 22.12.17	Fri 22.12.17	41;42	
44			2.3.7 Select and order buzzer	1 dy	Mon 25.12.17	Mon 25.12.17	34;40	
45			2.3.8 Selecting and ordering switches	1 dy	Mon 25.12.17	Mon 25.12.17	34;40	
46			2.3.9 Creating a programmer adapter	1 dy	Mon 25.12.17	Mon 25.12.17	34;40	
47			2.3.10 Creating a board for the vertical temperature sensor	1 dy	Mon 25.12.17	Mon 25.12.17	34;40	
48			2.3.11 Ordering the test board, the programmer adapter and the temperature sensor PCB	1 dy	Tue 26.12.17	Tue 26.12.17	32;33;34;38;39;40;44;45;46;47	
49			2.3.12 Arrival of the test board	0 dys	Sat 20.01.18	Sat 20.01.18		
50			2.3.13 Assembling the test board	7 dys	Mon 22.01.18	Tue 30.01.18	49	

Projekt: Final.mpp
Datum: Sun 10.06.18

Vorgang	Inaktiver Vorgang	Manueller Sammelrollup	Externer Meilenstein
Unterbrechung	Inaktiver Meilenstein	Manueller Sammelvorgang	Stichtag
Meilenstein	Inaktiver Sammelvorgang	Nur Anfang	In Arbeit
Sammelvorgang	Manueller Vorgang	Nur Ende	Manueller Fortschritt
Projektsammelvorgang	Nur Dauer	Externe Vorgänge	

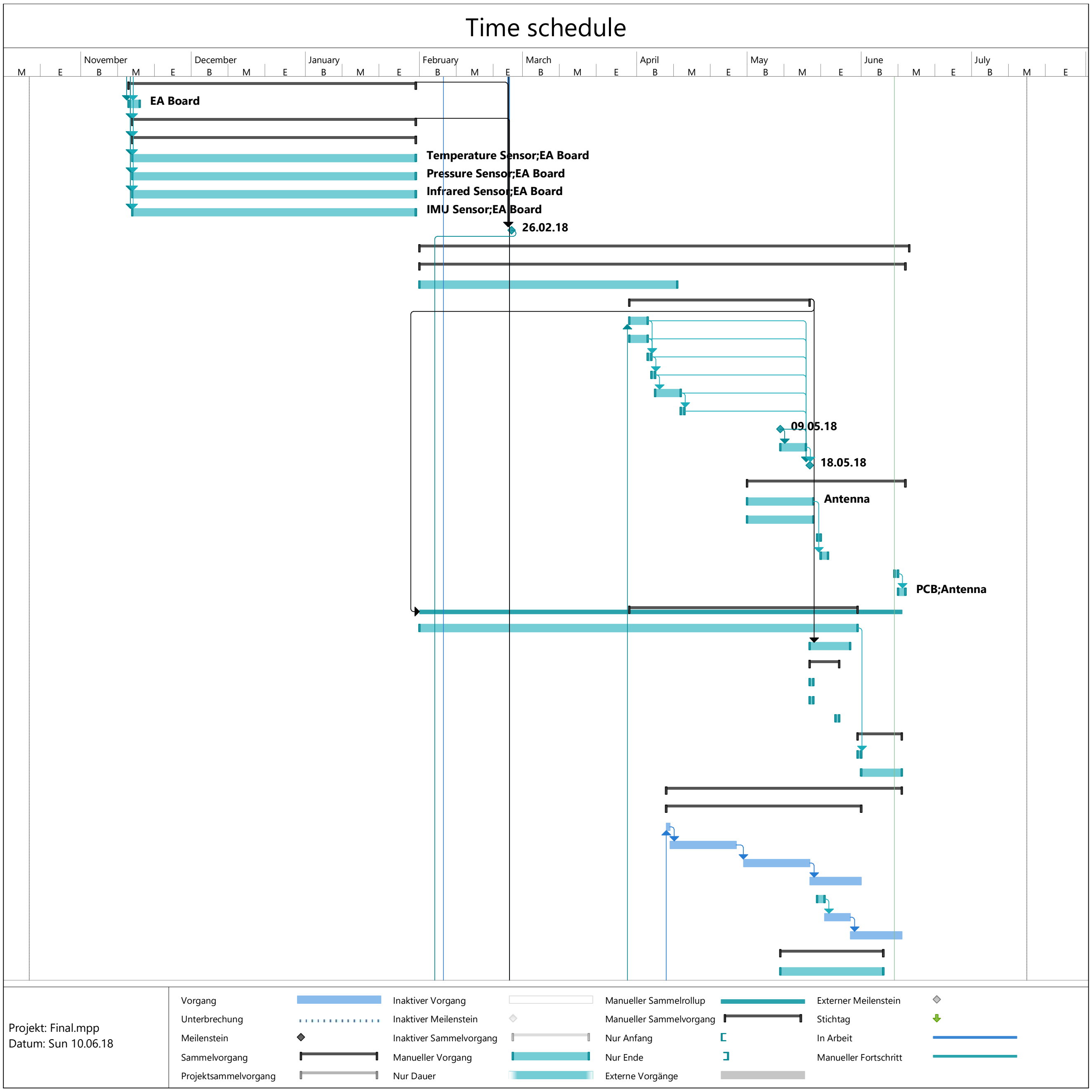

Time schedule
November
December
January
February
March
April
May
June
July
18.10.17
18.10.17
18.10.17
18.10.17
30.10.17
05.11.17
08.11.17
10.11.17
Datasheets
Datasheets
Datasheets
Datasheets
25.12.17
26.12.17
20.01.18
SMD Components;Temperature Sensor;Pressure Sensor;IMU Sensor;Buzzer;Powersupply Components;SD Card holder
Projekt: Final.mpp
Datum: Sun 10.06.18
Vorgang
Unterbrechung
Meilenstein
Sammelvorgang
Projektsammelvorgang
Inaktiver Vorgang
Inaktiver Meilenstein
Inaktiver Sammelvorgang
Manueller Vorgang
Nur Dauer
Manueller Sammelrollup
Manueller Sammelvorgang
Nur Anfang
Nur Ende
Externe Vorgänge
Externer Meilenstein
Stichtag
In Arbeit
Manueller Fortschritt

Time schedule

ID		Task Mode	Vorgangsname	Duration	Start	Finish	Predecessors	October B
23			**2.2 Software satellite**	**56 dys**	**Tue 14.11.17**	**Tue 30.01.18**		
24			2.2.1 Choose RTOS	3 dys	Tue 14.11.17	Thu 16.11.17	21;22	
25			**2.2.2 Sensors**	**55 dys**	**Wed 15.11.17**	**Tue 30.01.18**	21;22	
26			**2.2.2.1 I²C**	**55 dys**	**Wed 15.11.17**	**Tue 30.01.18**	21;22	
27			2.2.2.1.1 Temperature sensor	55 dys	Wed 15.11.17	Tue 30.01.18	21;22	
28			2.2.2.1.2 Pressure sensor	55 dys	Wed 15.11.17	Tue 30.01.18	21;22	
29			2.2.2.1.3 Infrared temperature sensor	55 dys	Wed 15.11.17	Tue 30.01.18	21;22	
30			2.2.2.1.4 Position and acceleration sensor	55 dys	Wed 15.11.17	Tue 30.01.18	21;22	
54			2.5 Completion of the first definition phase	0 dys	Mon 26.02.18	Mon 26.02.18	50;23;51	
70			**4 stage of development**	**95 dys**	**Thu 01.02.18**	**Thu 14.06.18**		
71			**4.1 Create PCBs**	**94 dys**	**Thu 01.02.18**	**Tue 12.06.18**		
72			4.1.1 Search for PCB Manufacturing Sponsors	50 dys	Thu 01.02.18	Wed 11.04.18		
73			**4.1.2 Satellite PCBs**	**35 dys**	**Fri 30.03.18**	**Fri 18.05.18**		
74			4.1.2.1 Intigate power supply	3 dys	Fri 30.03.18	Tue 03.04.18	63	
75			4.1.2.2 Integrating a microprocessor and its circuitry	3 dys	Fri 30.03.18	Tue 03.04.18		
76			4.1.2.3 Integrating sensors	1 dy	Wed 04.04.18	Wed 04.04.18	74;75	
77			4.1.2.4 Integrating controlling LEDs	1 dy	Thu 05.04.18	Thu 05.04.18	76	
78			4.1.2.5 Integrating improvments from the test board	5 dys	Fri 06.04.18	Thu 12.04.18	77	
79			4.1.2.6 Integrating cable connector	1 dy	Fri 13.04.18	Fri 13.04.18	78	
80			4.1.2.7 Visiting Straschu GmbH	1 dy	Wed 09.05.18	Wed 09.05.18		
81			4.1.2.8 Alignments	5 dys	Thu 10.05.18	Wed 16.05.18	80	
82			4.1.2.9 Finishing the PCBs	0 dys	Fri 18.05.18	Fri 18.05.18	74;75;76;77;78;79;80;81	
83			**4.1.3 Ground station PCBs**	**31 dys**	**Tue 01.05.18**	**Tue 12.06.18**		
84			4.1.3.1 Creating and ordering the PCBs	14 dys	Tue 01.05.18	Fri 18.05.18		
85			4.1.3.2 Creating and ordering the battery charger	14 dys	Tue 01.05.18	Fri 18.05.18		
86			4.1.3.3 Ordering components	1 dy	Sun 20.05.18	Sun 20.05.18		
89			4.1.3.6 Writing the software	2 dys	Mon 21.05.18	Tue 22.05.18	84	
87			4.1.3.4 Assembling the PCBs	1 dy	Sun 10.06.18	Sun 10.06.18		
88			4.1.3.5 Connecting the antenna	2 dys	Mon 11.06.18	Tue 12.06.18	87	
90			**4.2 Shell**	**44 dys**	**Fri 30.03.18**	**Wed 30.05.18**	**73**	
96			4.2.3 Developing a concept for the parachute	85 dys	Thu 01.02.18	Wed 30.05.18		
91			4.2.1 Creating the shell	7 dys	Fri 18.05.18	Mon 28.05.18	73	
92			**4.2.2 Consultation with the circuit board concept**	**6 dys**	**Fri 18.05.18**	**Fri 25.05.18**		
94			4.2.2.2 Insert hole for infrared temperature sensor	1 dy	Fri 18.05.18	Fri 18.05.18		
95			4.2.2.3 Insert mounting of the GPS modul	1 dy	Fri 18.05.18	Fri 18.05.18		
93			4.2.2.1 Insert key switch hole	1 dy	Fri 25.05.18	Fri 25.05.18		
97			**4.2.4 Parachute**	**8 dys**	**Thu 31.05.18**	**Mon 11.06.18**		
98			4.2.4.1 Calculation of the parachute	1 dy	Thu 31.05.18	Thu 31.05.18	96	
99			4.2.4.2 Creating the parachute	7 dys	Fri 01.06.18	Mon 11.06.18		
100			**4.3 Software satellite**	**46 dys**	**Mon 09.04.18**	**Mon 11.06.18**		
101			**4.3.1 Other components**	**39 dys**	**Mon 09.04.18**	**Thu 31.05.18**		
102			4.3.1.1 Buzzer	1 dy	Mon 09.04.18	Mon 09.04.18	58	
103			4.3.1.2 SD Card	14 dys	Tue 10.04.18	Fri 27.04.18	102	
104			4.3.1.3 Transceiver	14 dys	Mon 30.04.18	Thu 17.05.18	103	
105			4.3.1.4 GPS	10 dys	Fri 18.05.18	Thu 31.05.18	104	
106			4.3.2 Read out the battery voltage	2 dys	Sun 20.05.18	Mon 21.05.18		
107			4.3.3 Communication between the modules	5 dys	Tue 22.05.18	Mon 28.05.18	106	
108			4.3.4 LEDs error messages	10 dys	Tue 29.05.18	Mon 11.06.18	107	
109			**4.4 Software Groundstation**	**20 dys**	**Thu 10.05.18**	**Wed 06.06.18**		
112			4.4.3 Programming automatic analysis	20 dys	Thu 10.05.18	Wed 06.06.18		

Projekt: Final.mpp
Datum: Sun 10.06.18

Vorgang	Inaktiver Vorgang	Manueller Sammelrollup	Externer Meilenstein
Unterbrechung	Inaktiver Meilenstein	Manueller Sammelvorgang	Stichtag
Meilenstein	Inaktiver Sammelvorgang	Nur Anfang	In Arbeit
Sammelvorgang	Manueller Vorgang	Nur Ende	Manueller Fortschritt
Projektsammelvorgang	Nur Dauer	Externe Vorgänge	


Time schedule
November
December
January
February
March
April
May
June
July
M
E
B
EA Board
Temperature Sensor;EA Board
Pressure Sensor;EA Board
Infrared Sensor;EA Board
IMU Sensor;EA Board
26.02.18
09.05.18
18.05.18
Antenna
PCB;Antenna
Projekt: Final.mpp
Datum: Sun 10.06.18
Vorgang
Unterbrechung
Meilenstein
Sammelvorgang
Projektsammelvorgang
Inaktiver Vorgang
Inaktiver Meilenstein
Inaktiver Sammelvorgang
Manueller Vorgang
Nur Dauer
Manueller Sammelrollup
Manueller Sammelvorgang
Nur Anfang
Nur Ende
Externe Vorgänge
Externer Meilenstein
Stichtag
In Arbeit
Manueller Fortschritt

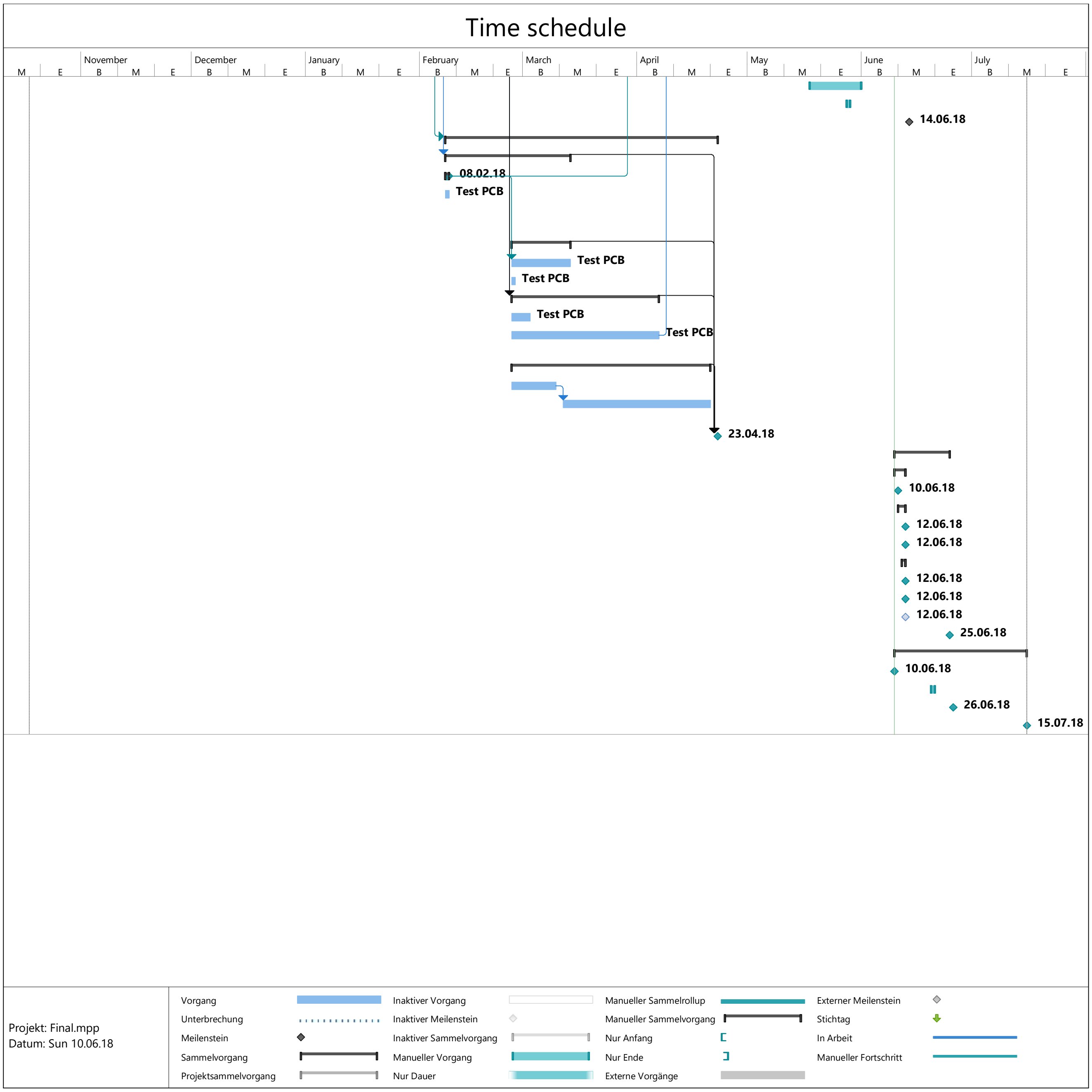
Time schedule

ID		Task Mode	Vorgangsname	Duration	Start	Finish	Predecessors
111			4.4.2 Optimize GUI	10 dys	Fri 18.05.18	Thu 31.05.18	
110			4.4.1 Integrate serial interface	1 dy	Mon 28.05.18	Mon 28.05.18	
113			4.5 Finishing phase of developmenting	0 dys	Thu 14.06.18	Thu 14.06.18	
55			**3 First test phase**	**52 dys**	**Thu 08.02.18**	**Mon 23.04.18**	**54**
62			**3.3 Test board**	**24 dys**	**Thu 08.02.18**	**Tue 13.03.18**	**50**
63			**3.3.1 Power supply**	**1 dy**	**Thu 08.02.18**	**Thu 08.02.18**	
64			3.3.1.1 Functionality testing	1 dy	Thu 08.02.18	Thu 08.02.18	
65			3.3.1.2 Checking the performance in comparison to a finished component	0 dys	Thu 08.02.18	Thu 08.02.18	64
66			**3.3.2 Components**	**12 dys**	**Mon 26.02.18**	**Tue 13.03.18**	
67			3.3.2.1 Testing sensors trough the Software	12 dys	Mon 26.02.18	Tue 13.03.18	63
68			3.3.2.2 Testing the buzzer	1 dy	Mon 26.02.18	Mon 26.02.18	
56			**3.1 Software satellite**	**30 dys**	**Mon 26.02.18**	**Fri 06.04.18**	**25;50**
57			3.1.1 Testing the sensors	5 dys	Mon 26.02.18	Fri 02.03.18	
58			3.1.2 Correct errors / structure the program more efficiently	30 dys	Mon 26.02.18	Fri 06.04.18	
59			**3.2 Software ground station**	**40 dys**	**Mon 26.02.18**	**Fri 20.04.18**	
60			3.2.1 Testing GUI through testdata	10 dys	Mon 26.02.18	Fri 09.03.18	
61			3.2.2 Correct errors / structure the program more efficiently	30 dys	Mon 12.03.18	Fri 20.04.18	60
69			3.4 Finishing testing phase	0 dys	Mon 23.04.18	Mon 23.04.18	66;62;59;56
114			**5 qualification phase**	**10 dys**	**Sun 10.06.18**	**Mon 25.06.18**	
115			**5.1 Testing the individual elements**	**2 dys**	**Sun 10.06.18**	**Tue 12.06.18**	
116			5.1.1 Software ground station	1 dy	Sun 10.06.18	Sun 10.06.18	
117			**5.1.2 PCBs**	**2 dys**	**Mon 11.06.18**	**Tue 12.06.18**	
118			5.1.2.1 Software satellite	2 dys	Mon 11.06.18	Tue 12.06.18	
119			5.1.2.2 Communication between the modules	2 dys	Mon 11.06.18	Tue 12.06.18	
120			**5.2 Testing the entire system**	**1 dy**	**Tue 12.06.18**	**Tue 12.06.18**	
121			5.2.1 Testing the radio path	1 dy	Tue 12.06.18	Tue 12.06.18	
122			5.2.2 Droptest with data measurement	1 dy	Tue 12.06.18	Tue 12.06.18	
123			5.2.3 Drop test at an airfield	1 dy			
124			5.3 Completion of the qualification phase	0 dys	Mon 25.06.18	Mon 25.06.18	
125			**6 Closing phase**	**25 dys**	**Sun 10.06.18**	**Sun 15.07.18**	
127			6.2 Finishing documentation	0 dys	Sun 10.06.18	Sun 10.06.18	
126			6.1 Create presentation frame	1 dy	Wed 20.06.18	Wed 20.06.18	
128			6.3 Flight to Porto	0 dys	Tue 26.06.18	Tue 26.06.18	
129			6.4 Writing the documentation of the competition	12 dys	Sun 01.07.18	Sun 15.07.18	

October
B

Projekt: Final.mpp
Datum: Sun 10.06.18

Vorgang	Inaktiver Vorgang	Manueller Sammelrollup	Externer Meilenstein
Unterbrechung	Inaktiver Meilenstein	Manueller Sammelvorgang	Stichtag
Meilenstein	Inaktiver Sammelvorgang	Nur Anfang	In Arbeit
Sammelvorgang	Manueller Vorgang	Nur Ende	Manueller Fortschritt
Projektsammelvorgang	Nur Dauer	Externe Vorgänge	


Time schedule
November
December
January
February
March
April
May
June
July
M E B M E B M E B M E B M E B M E B M E B M E B M E B M E
14.06.18
08.02.18
Test PCB
Test PCB
Test PCB
Test PCB
Test PCB
23.04.18
10.06.18
12.06.18
12.06.18
12.06.18
12.06.18
12.06.18
25.06.18
10.06.18
26.06.18
15.07.18
Projekt: Final.mpp
Datum: Sun 10.06.18
Vorgang
Unterbrechung
Meilenstein
Sammelvorgang
Projektsammelvorgang
Inaktiver Vorgang
Inaktiver Meilenstein
Inaktiver Sammelvorgang
Manueller Vorgang
Nur Dauer
Manueller Sammelrollup
Manueller Sammelvorgang
Nur Anfang
Nur Ende
Externe Vorgänge
Externer Meilenstein
Stichtag
In Arbeit
Manueller Fortschritt