X-Copter Team

Exposee

Development a platform to test several sensors or processes for a autonomic aircraft

Interim Report

X-Copter



Table of content

[Project Description 4](#_Toc432066803)

[Project members 5](#_Toc432066804)

[Functional and non-function Requirements 6](#_Toc432066805)

[Analysis of customer needs 8](#_Toc432066806)

[System architecture 9](#_Toc432066807)

[Project documentation 10](#_Toc432066808)

[Project Management (Scrum) 10](#_Toc432066809)

[Product Owner 10](#_Toc432066810)

[Development Team 10](#_Toc432066811)

[Scrum Master 10](#_Toc432066812)

[Sprint 1 11](#_Toc432066813)

[Choice of the RC – Controller and the receiver 14](#_Toc432066814)

[The requirements for the RC- Controller and the receiver are: 14](#_Toc432066815)

[A selection of companies, they produce RC- Controller: 14](#_Toc432066816)

[Major properties of the RC- Controller: 14](#_Toc432066817)

[Power supply 17](#_Toc432066818)

[Get the circuit plan 17](#_Toc432066819)

[Check the board 17](#_Toc432066820)

[Kinect and USB Host Controller Documentation 18](#_Toc432066821)

[Initial Situation 18](#_Toc432066822)

[Requirements for the USB-Controller 18](#_Toc432066823)

[Common USB-Controller Packages 18](#_Toc432066824)

[Controllers that come into question 19](#_Toc432066825)

[Conclusion 20](#_Toc432066826)

[Charging Batteries for the X-Copter 21](#_Toc432066827)

[Charging Batteries: 21](#_Toc432066828)

[Commercial Flight control 23](#_Toc432066829)

[Sprint 2 25](#_Toc432066830)

[First Test flight 26](#_Toc432066831)

[Preparation 26](#_Toc432066832)

[Attempt 1 26](#_Toc432066833)

[Attempt 2 26](#_Toc432066834)

[Attempt 3 27](#_Toc432066835)

[Attempt 4 27](#_Toc432066836)

[Conclusion 27](#_Toc432066837)

[Sprint 3 28](#_Toc432066838)

[Domain model 28](#_Toc432066839)

[Definition of Graupners HoTT- SUMD- Signal 29](#_Toc432066840)

[Why SUMD 29](#_Toc432066841)

[Definition 30](#_Toc432066842)

[Time Requirements 30](#_Toc432066843)

[Structure of a HoTT- SUMD frame 30](#_Toc432066844)

[SUMD\_Header section description 30](#_Toc432066845)

[SUMD\_Data section description 30](#_Toc432066846)

[SUMD\_CRC section description 31](#_Toc432066847)

[Channel data interpretation 31](#_Toc432066848)

[Implementation of the SUMD Parsing 32](#_Toc432066849)

[Saving raw SUMD-Frame Bytes from the UART 32](#_Toc432066850)

[Interpreting the received SUMD-Frame 32](#_Toc432066851)

[UART Driver 32](#_Toc432066852)

[SUMD-Frame-high 33](#_Toc432066853)

[SUMD-Frame-low 33](#_Toc432066854)

[Second Flight Test 35](#_Toc432066855)

[Organisation 35](#_Toc432066856)

[Changes at the X-Copter since the last Flight test 35](#_Toc432066857)

[Attempt 1 35](#_Toc432066858)

[Conclusion 35](#_Toc432066859)

[PID Regulator 36](#_Toc432066860)

[The Proportional Part 36](#_Toc432066861)

[The Integral Part 36](#_Toc432066862)

[The Differential Part 37](#_Toc432066863)

[Lessons Learned 39](#_Toc432066864)

[Scrum 39](#_Toc432066865)

[Long term calendar management 39](#_Toc432066866)

[Quick task can not be made quick 39](#_Toc432066867)

[Literaturverzeichnis 41](#_Toc432066868)

# Project Description

Today multi-copter is more and more common. They’re not only being used for fun, but also in a lot of other areas. Photographers can take pictures that were not easily possible before and film makers are using them to produce stunning shots. The police and the army are using them to observe areas of interest or events like football games and demonstrations. (Team, 2015)

Our client though has something else in mind. The goal is to produce a completely autonomous system which can either navigate itself through an area and build a map out of the data received through its sensors. (Team, 2015)

The previous team developed a model and some software for the X-Copter. Our goal is now to continue to develop more Software and test the flight ability of the X-Copter. To get the data while the X-Copter flight in the air we should develop a monitoring station.

# Project members

* Jan Goller
  + Team Linux and power supply
* Thomas Weber
  + Team Flight Controller
* Alexander Ott
  + Team Flight Controller
* Florian Schneider
  + Team Flight Controller
* Daniel Maurus
  + Team Linux and USB-Controller
* Stefan Gabor
  + Team Linux and USB-Controller
* Jochen Hoeft
  + Team Flight Controller and pilot
* Lukas Öfner
  + Scrum Master
  + Communicate between customer and Teams
* Benjamin

# Functional and non-function Requirements

NF1: Name: Flight test with the commercial flight controller

Description:

Rationale:

Priority:

NF2: Name: USB-Controller for the Kinect-Cameras

Description:

Rationale:

Priority:

NF3: Name: Monitoring station

Description:

Rationale:

Priority:

NF4: Name: Own flight controller

Description:

Rationale:

Priority:

NF5: Name: Stable power supply for the X-Copter

Description:

Rationale:

Priority:

NF6: Name: Find a remote

Description:

Rationale:

Priority:

NF7: Name: Autonomic flight

Description:

Rationale:

Priority:

NF8: Name:

Description:

Rationale:

Priority:

NF9: Name:

Description:

Rationale:

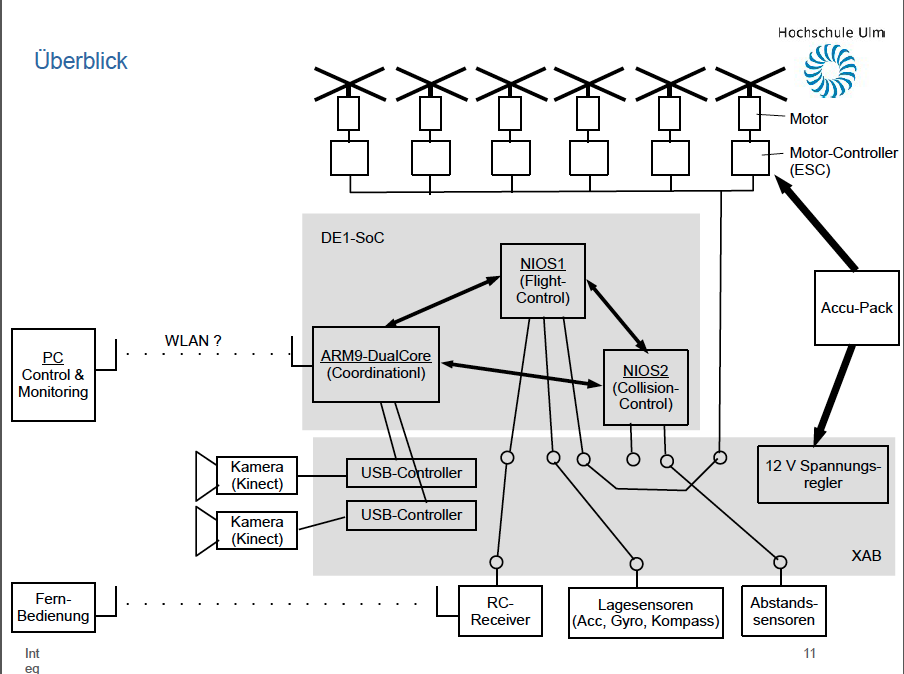
Priority:

# Analysis of customer needs

# System architecture

The system architecture is documented in the final report of the team Bumblebee. (Team, 2015) This is a quick overview for our project. The DE1-SoC board is the main part of the X-Copter. The board contain a ARM9-DualCore CPU and two NIOS1 CPUs. The extension board is for all Sensors and the power supply. The cameras are connected to the extension board.

Our RC-Receiver is also connected to the extension board.



# Project documentation

## Project Management (Scrum)

## 

Scrum is an iterative and incremental agile software development methodology for managing product development. We had a lecture of Project Management by Dr. Balsen. He owns a small developing company. He used Scrum for years. Dr. Balsen show us all the positive and negative aspects of Scrum and teach us how to handle Scrum. A key principle of scrum is its agile switch of customer changes during a project. And an other main reason for Scrum is the step by step developing.

There are several roles in Scrum:

### Product Owner

The product owner is the voice of the customer. He writes customer-centric items typical, the user stories, rank them and prioritizes them. In your project the product owner is same with the customer.

### Development Team

The development team is self-organizing in Scrum. A team is made up of 3-9 persons. The team is responsible for the progress of the project. Each team had their own tasks.

In each task the actual work is described by the product owner and scrum master. If the development team finish some task the continuing with the next open task.

### Scrum Master

The scrum master coaching the team with the scrum principles. He is responsible to remove impediments of the development team. The scrum master facilitation team events like the daily scrum or other meetings. He acts as buffer between the team and the customer.

# Sprint 1

The first Sprint starts at the 13. April and ends on the 11. June. We finished a lot of tasks and reached almost our goals completely. You can see it in the Burndown Chart a picture is below. The last part where the curve doesn’t fit to the nominal value was the issue with the USB-Controller. The Problems are documented in the Impediment Backlog. All Sprint tasks are documented exact below.

1. Select the remote:

* costumer pitch about the remote
* comprasion of remotes
* price inquiry
* order the remote

1. Stable battery

* get the [circuit](http://www.dict.cc/englisch-deutsch/circuit.html) [diagram](http://www.dict.cc/englisch-deutsch/diagram.html)
* check the existing board
* build the circuit on a prototyp board

1. [Charge](http://www.dict.cc/englisch-deutsch/charge.html) [the](http://www.dict.cc/englisch-deutsch/the.html) [battery](http://www.dict.cc/englisch-deutsch/battery.html)

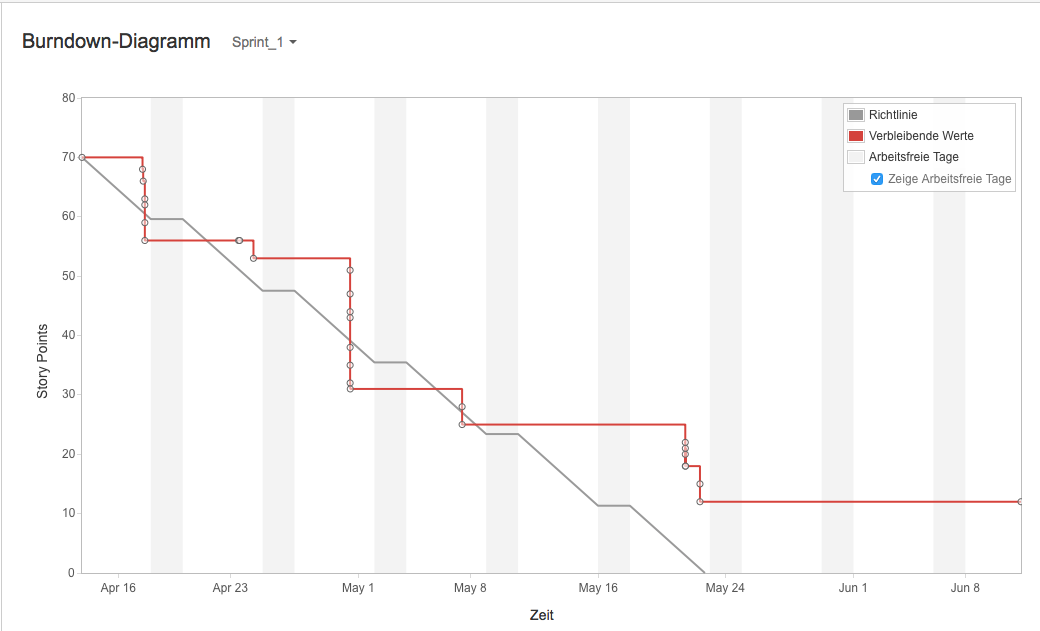
* programming the charger
* extern power supply

1. USB-Controller

* comprasion of USB-Controller
* proof requirements
* find a new USB-Controller

1. Commercial flight controller

* get the Software and install it
* feature list
* configure the [cruise](http://www.dict.cc/englisch-deutsch/cruise.html) [control](http://www.dict.cc/englisch-deutsch/control.html)
* connect the crusie controll to the rotors
* connect the crusie controll to the flight controller
* connect the flight controller to the X-Copter
* get a connection to the remote
* configurate the flight controller
* configurate the remote



## Choice of the RC – Controller and the receiver

### The requirements for the RC- Controller and the receiver are:

* Both have to use the sum signal
* Both have to provide 4 channels minimum, better up to 6 – 8 channels
* The costs have to be less than 350€ for both of them
* The RC- Controller have to be configurable easily

### A selection of companies, they produce RC- Controller:

1. Graupner
2. Futaba
3. Spektrum
4. Modelcraft

We decided to work with a Graupner RC- Controller!

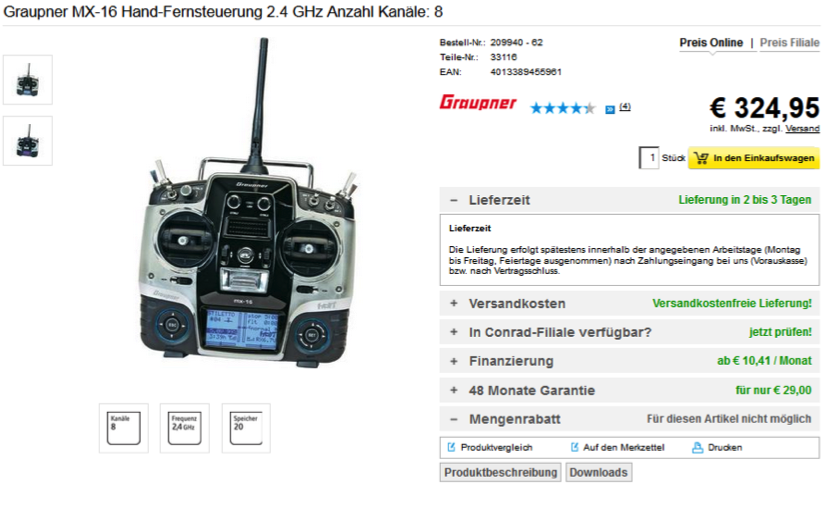
Reasons for Graupner:

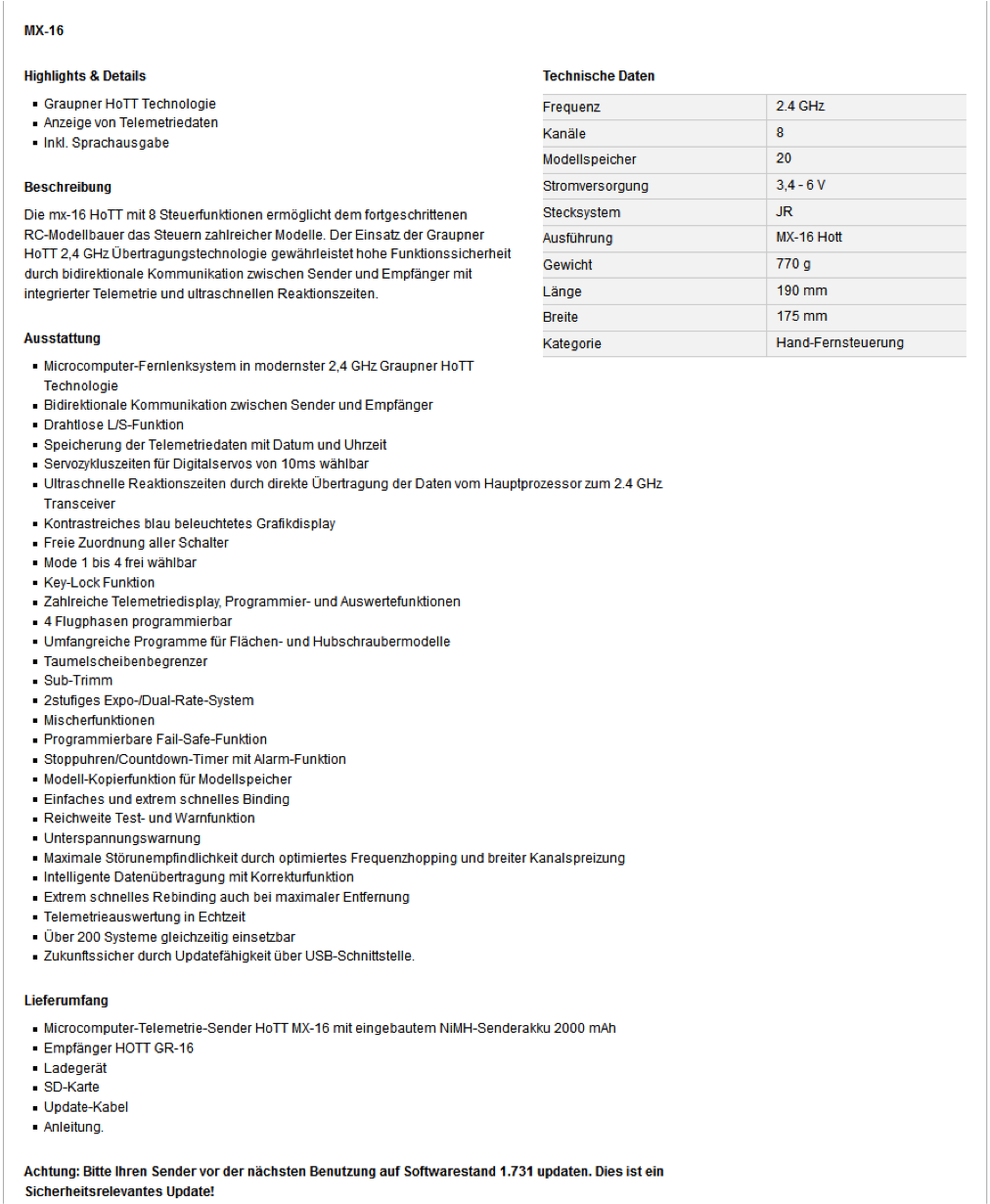
* Graupner is an innovative and leading company in RC- modelling
* Graupner ensures a high quality standart
* Graupner provides lots of datasheets for each product
* Graupner has a big RC- community

Out of the range of Graupner products we selected the “ Graupner MX 16 “:

### Major properties of the RC- Controller:

* 8 channel
* HoTT technology (sum signal, transmit up to 16 channels)
* Bidirectional communication between transmitter and receiver
* Free configurable switches
* Signal range 4 km
* Very fast rebinding





## Power supply

The power supply on the extension board we got from the last project team isn’t working so we have to try to find the error and a solution.

At first we took a look on the EAGLE ® formatted plan of the circuit. It seems to be ok but we decided to generate a new one with a web app by Texas Instruments ® called WEBENCH® System Power Architect to compare them. We used the tool and generated a new circuit plan with the following input parameters:

* V\_in\_max : 25 V
* V\_in\_min : 13 V
* V\_out : 12 V
* I\_out : 5 A

These parameters are given by the maximum and minimum output of the used accumulators (V\_in) we use in the X-Copter and the SoC-Board restrictions which are 12V input voltage and 3.5A output current. We added another 1.5A for provision. After that, we compared our new plan with the plan, the last project used. We figured out that they are the same so there cannot be the problem. The next step was to compare all the components and the voltage control IC. There we saw that the last group, which designed and populated two circuit boards, used different ICs on each of them. So at least one of the boards cannot work. They also told us that when they tried to figure out the error, they probably destroyed some parts. Big error sources are the SMD parts. The problem with them is that they aren’t easily testable. Therefor we bought all the parts in DIP norm and built the circuit on a plugboard to test if the circuit could work in general. The voltage control IC wasn’t available in DIP norm so we mounted it on an adapter to use it on the plug board. After populating the board we tested it with a configurable power supply where we simulated the accumulators. No matter what we changed on the board, the output voltage dropped all the time and wasn’t stable at all. After a discussion on an electronics forum and Mr. Steiper, we found out that it’s not possible to build a stable power supply with our available methods and parts because of the oscillating circuits that the capacitors and coils generate. Because all this cost a lot of time and our missing experience and tools in this subject, we decided to buy a ready-made power supply from an online shop.



## Kinect and USB Host Controller Documentation

### Initial Situation

The customer wants two Kinects to be put on the X-Copter. They are to provide 3D-Image data to map the surrounding locale of the X-Copter. To be able to communicate with two Kinects, two USB Host Controllers are needed. This is because one Kinect needs at least ~21 MB/s data transfer rate for 3D-Images at 640×480 pixels with 30 frames per second, which is too much for one controller to handle. 21 MB/s are divided into ~12MB/s for depth camera and 9 MB/s for color camera [1]. For proper 3D-Image data color- and depth camera have to work at the same time and can not be separated, which strikes out the option to save bandwidth with using only one camera at the time.

Kinect cameras will be connected via USB 2.0 plug to the USB-Controllers. For the Controllers to be able to communicate with the DE1-SOC system, an interface has to be implemented into the existing SOPC for communication between the devices. Real time 3D-Data processing will be the task of another external system with an Intel processor. Our customer stated that on a similar side project of him even an Intel I7 quad core processor is struggling with processing the data. For further information about hardware requirements of Kinect-Systems refer to [2].

### Requirements for the USB-Controller

There are certain cut in stone requirements for the USB-Controller to work with Kinect and to fit in the design of our system:

- Must be available on the market

- Must not exceed the quantity of pins our system is able to offer

- Drivers for Linux have to be available

- Chip has to have outgoing pins to be solderable

- Full High-Speed data transfer rate of 480 MBit/s

- (Should be ULPI compatible if present Waveshare 3300-Transceivers are meant to be used)

### Common USB-Controller Packages

There are three different common USB-Controller Packages that are solderable with the equipment available: QFN (Quad Flat No-leads package), LQFP (Low Profile Quad Flat Package) and TQFP (Thin Quad Flat Package). Information, advantages and disadvantages of these packages can be reviewed at [5]. QFN is harder to solder which is why QFP style packages are the preferred choice.

### Controllers that come into question

Investigation about USB-Controllers lead to a list of four different controllers that will be evaluated further in this document. The first controller is one chosen from Frank Seifert for his Bachelor's Thesis: “Conception and realization of a control computer platform for a quadcopter flying model”[6]. He compared three different solutions for USB-Controller implementation into his system. His selection included the ISP1362BD, its successor the ISP1761BE and a softcore FPGA solution. Implementing the USB-Controller directly into the FPGA fell out of the question because of the high price for an USB-Controller IP-Core (prices circle around 5000€). Open Source IP Cores for USB Host Controllers are few, have a low set of features and are badly documented, which makes them less than optimal for this project. Frank Seifert also crossed out the ISP1761BE because of a higher pin count and no Linux drivers available at the time of writing his Bachelor's Thesis. His research led him to the believe that the ISP1362BD would be best suited for his endeavors.

Further research from our side showed that Linux Drivers are available for the ISP1761BE nowadays, which would make it a suitable choice for the project. Further investigation showed that the successor to the ISP1761BE, the SAF1761BE from NXP Semiconductors, is also available to purchase and supported with Linux drivers. The fourth and last USB Controller mentioned here is the FT313H(L/P) from Future Technology Devices International Ltd.

Cypress is a another company that is also offering a wide array of USB solutions, sadly they don't have USB 2.0 Host Controllers in their repertoire.

Illustration 1: Cypress USB offerings (http://www.cypress.com/fckImages/myresources/USBControllers\_Overviewimg(1).jpg)

Chips with packages that are not solderable with the equipment at our disposal will not go into the equation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **ISP1362BD** | **ISP1761BE** | **SAF1761BE** [3] | **FT313H(L/P)** [4] |
| **Date of production** | Rev. 04  12.2004 | Rev. 01  01.2005 | Rev.02  06.2012 | Ver.1.2  2013 |
| **Package** | LQFP64 | LQFP128 | LQFP128 | 64 LQFP  64 TQFP |
| **Driver for Linux** | yes | yes | yes | yes |
| **Transfer rate** | 96 Mbit/s | 480 Mbit/s | 480 Mbit/s | 480 Mbit/s |
| **RAM Memory** | unkn. | unkn. | unkn. | 48 KB |
| **ULPI compatible** | unkn. | unkn. | unkn. | unkn. |
| **Quantity of I/O Pins** | 27 Pins | 16Bit: 41 Pins  32Bit: 57 Pins | 16Bit: 41Pins  32Bit: 57 Pins | ? |
| **Info** | Discontinued | Discontinued | Available | Available |
| **Pros** | + Frank Seifert implemented this chip in his bachelor project | + speed  + similar to ISP1362BD | + speed  + similar to ISP1761BE | + speed  + 64 PIN package  + UMFT313EV Development Module available |
| **Cons** | - speed (too low for Kinect)  - not available | - not available | - used mainly in automotive systems  - no evaluation board | Unkn. As of time of writing |

### Conclusion

The ISP1362BD is not suitable as an USB Controller for the use with Microsoft Kinects because of a transfer rate of only 96 Mbit/s which is Full Speed USB 2.0. Kinects need at least High Speed USB 2.0 with 480 Mbit/s. Furthermore the controller is not supported anymore and it is almost impossible to obtain those controllers on today's market.

The ISP1761BE does not make the cut either. Although it supports High Speed USB 2.0 and has Linux drivers, it has a larger footprint with its LQFP128 package and is also discontinued. Its successor the SAF1761 which is similar in features is mainly used in automotive systems which means that it is not available in the common consumer market.

Which leads us to the FT313H(L/P) which offers the best characteristics for our endeavors. It is still supported, offers Linux drivers. It has a relatively low footprint, is solderable with the tools at hand and comes in two packages: 64LQFP and64TQFP. It supports High Speed USB 2.0 transfer rates and can also be ordered with a development module.

## Charging Batteries for the X-Copter

**Info:** This little tutorial describes how to charge our batteries for the X-Copter-Project

|  |  |  |
| --- | --- | --- |
|  | **Identifier:** | **Information:** |
| **Hardware:** | **Battery1 (LiPo\_01)** | **Name:** FlightPower Hacker evo 20 4900mAh 6S1P  **Charging:** continuous: 20C, burst: 50C, charging: 1C  **Stored in:** Speicher[31] as LiPo\_01 |
| **Battery2 (LiPo\_02)** | **Name:** XTRON 40C 5000mAh 6S1P  **Charging:** continuous: 40C, burst: 80C, charging: 4C  **Stored in:** Speicher[20] as LiPo\_02 |
| **Battery3 (NiCd\_01)** | **Charging:** loading current: 110mAh  **Stored in:** Speicher[19] as NiCa\_01 |
| **Charging-Station:** | Ultra Duo Plus 60 |

We saved three configurations in the Ultra Duo for our batteries named: Lipo\_01, Lipo\_02 and NiCd\_01

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Battery Overview** | | **LiPo\_01** | **LiPo\_02** | **NiCd\_01** |
| Cell-Count: | | 6 | 6 | 4 |
| **Nominal** | Cell-Voltage | 3.7V | 3.7V |  |
| Total-Voltage | 22.2V | 22.2V |  |
| **Max.** | Cell-Voltage | **4.2**V | **4.2**V |  |
| Total-Voltage | **25**V | **25**V |  |
| **Min.** | Cell-Voltage | 3V | 3V |  |
| Total-Voltage | 18V | 18V |  |
| **Max. Charge Current** | | 4.9 A (1C) | 20 A(4C) |  |

**Hint: The maximum charge power is not above 80 W.**

**Important:** Warnings and security information are found in the documentantion-pdf:

6478\_ULTRA DUO PLUS 60\_de.pdf

You should read that before using the charging station as the information therein will not be in this tutorial.

### Charging Batteries:

**Current charged per charging session** = current \* charging time

Refer to the datasheet of your batteries for information about max. current for charging.

**Standard-charging-current** is 1/10 of the capacity (1.7Ah capacity -> 170mA standard-charging-current)

1. Plug power cable to start charging device
2. Connect charging cables to charging station (red = plus, black = minus)
3. Recommend for LiPo’s: Connect cell adapter to “Balance” board and “Balance” board to charging station
4. We will use the CC/CV charging-mode for our batteries as fast-charging not supported by every type

## Commercial Flight control

In order to test the construction of the XCopter for the first time,

without wasting too much time on developing an own flight control.

We decided to install a commercial flight control. In this case the DJI NAZA V2 was used.

It is a fully developed flight control unit, which was developed to be easily installed in any multi copter system. It comes with an integrated 3-axis gyro sensor and acceleration sensor as well as an external GPS unit. The only items which need to be connected to it are all Electronic speed controllers (ESCs) and a RC-receiver, the gimbal(DJI camera) part is not necessary.

If every Electronic Part is connected properly, the flight controller has to be configured by following these steps:

1. Download the driver and the Assistant installation software in EXE format from www.dji.com.
2. Switch on the transmitter and then power on your autopilot system.
3. Connect your autopilot system and PC via a Micro-USB cable (according to image 1).

Required connections for USB:

* NAZA V2 Power Management Unit at **Exp.**
* V-Sen wire from PMU at **X3**
* USB adapter at **LED**

1. Open the driver installation software and follow the instructions to complete installation.

Important: The NAZA V2 has to be connected to install this driver

1. Run the Assistant installation software and follow the instructions to complete installation.
2. Select the “Basic” option. Please follow step-by-step for your first-time-configuration. Basic configuration is necessary, including Mixer Type, Mounting, RC, and Gain settings.

Further information can be found in the Help text of the NAZA V2 Assistant Software.

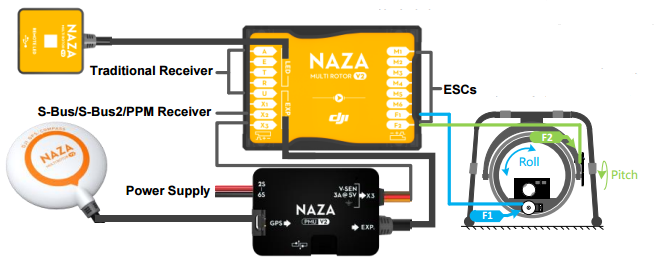


Abbildung 1 NAZ V2

Screenshot from the DJI NAZA V2 quick start guide.

# Sprint 2

1. Commercial flight controller

* Back up the configuration of the controller
* Get a date for the first flight
* Organize a bus for the test flight
* Switch for the RC-Controller
* Build a frame for the landing
* Fuses for the X-Copter

1. Simple flight controller

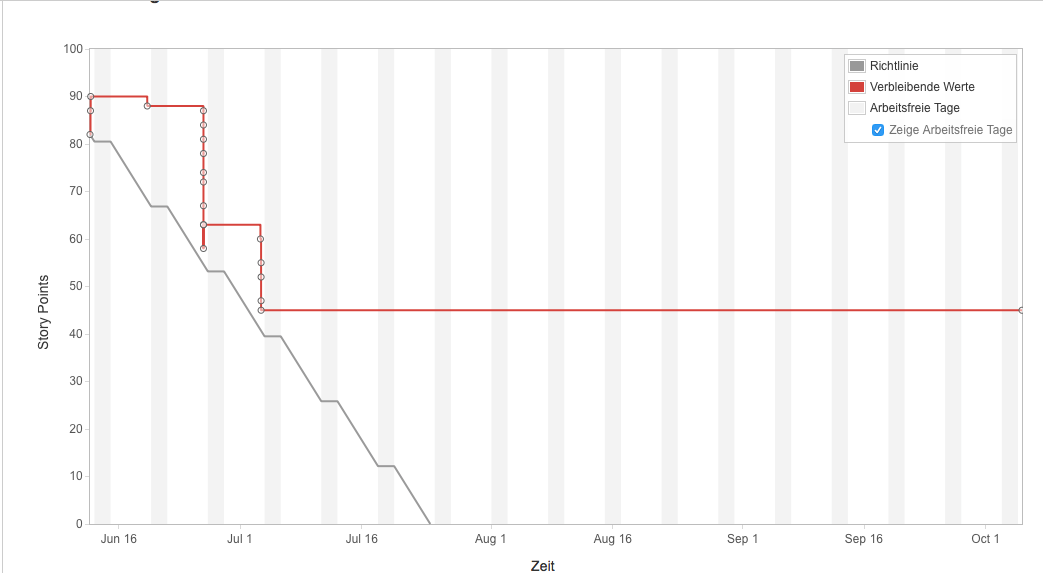
* Get information about the drivers
* Reveal the components of a flight controller
* Get information about PID regulator
* Test the drivers

1. USB-Controller

* Search for a new USB-Controller

1. Day of informatics

* Make a presentation about the X-Copter
* Make a poster for the day of informatics



## First Test flight

### Preparation

### 

At first all Components got attached to the Model.

In addition to the aircraft, a basic equipment of tools and spare parts was taken to the Model flying site.

Jochen Hoeft

### Attempt 1

The first attempt failed. The Motors had the wrong direction of rotation. So there was now upwards boost but a downwards boost.

Error Analysis: Falsely it was assumed that the rotors were mounted incorrectly. So we switched the Rotors which led to upward boost from every rotor.

### Attempt 2

The wrong error analysis led to another fail attempt. From the perspective of the flight controller now each rotor turned the wrong direction. Now the yaw correction worked the wrong direction and self-reinforced the rotation of the X-Copter. The vehicle came immediately out of control.

Error Analysis: The spinning direction of all motors were checked and we noticed the wrong spinning direction. The reason for this was that the flight controller has been mounted rotated by 90∞. The flight controller was turned by 90∞ and remounted. In addition, all rotors were switched back in the correct position.

### Attempt 3

This attempt started well but the X-Copter lost upward boost at one side and crashed.

Error Analysis: It turned out that one rotor loosened. All rotor screws were tightened.

### Attempt 4

The Take-off went well. However, one Motor flew away.

Error Analysis: The rubber vibration damper that were used to mount the motors seemed to be not strong enough.

The tools and spare parts we had were not enough to mount the motors without the vibration damper.

### Conclusion

The X-Copter vehicle should be able to fly.

The rubber vibration damper is not recommended.

Another Test is necessary.

# Sprint 3

## Domain model

In the picture Abbildung 2 Domaine Model you can see the layer orientated structure of our flight controller. From the top to the bottom, each layer uses the software form the layer below. This subjection is shown by the arrows pointing to the bottom. The flight controller acts as an auto stabilizer when the UAV is in the air so that the user don't has to steer against wind gusts and other external influences that disturb the flight.

As you can see in Abbildung 2 Domaine Model and already mentioned, the structure has eight layers:

The very bottom layer is the layer that's closet to the hardware and colored light red in model1.png. In this layer there are the drivers for the general I/O communication on the board such as I²C and PWM.

In the second layer which is colored yellow, the drivers for the sensors and actors the X-Copter uses to communicate with the real world. Here we use four or six motors with rotors depending on the configuration of the X-Copter. In addition, we have an accelerometer, a gyroscope and a compass to detect the three dimensional lay and alignment. To share data between each other, they use the orange layer drivers. To receive the command signals from the remote control a RC driver is also implemented in this layer.

To get useful data from the sensors it has to be filtered first to handle the disturbances. These filters are colored dark green and are in layer three.

In the fourth layer the RC and sensor interpreters are placed and colored blue. The interpreters are used to get human readable and understandable data from the sensors.

One layer above and colored pink, there are the controllers for the sensors, motors and the remote control. Each controller manages the specific device. This includes accessing the driver to get or set the raw data from the drivers. This raw data is then, if its one of the sensors, filtered and interpreted. This all is done automatically to provide an easy access by the controllers to higher layers. The motor controller has

The PIDs are located between the PID\_Controller (orange) and the other controllers (pink) in the sixth layer. Each motor has its own PID regulator that is controlled by the orange colored PID\_Controller.

The PID controller in layer seven collects all necessary data from the controllers in the pink layer five for the PID regulators. This includes the set point from the remote control and the actual value from the sensors. The value the PIDs return is then used by the motor controllers to set the needed speed to let the X-Copter fly stable.

The very top layer is the flight controller that manages the timing and general flow of the application in the main method.

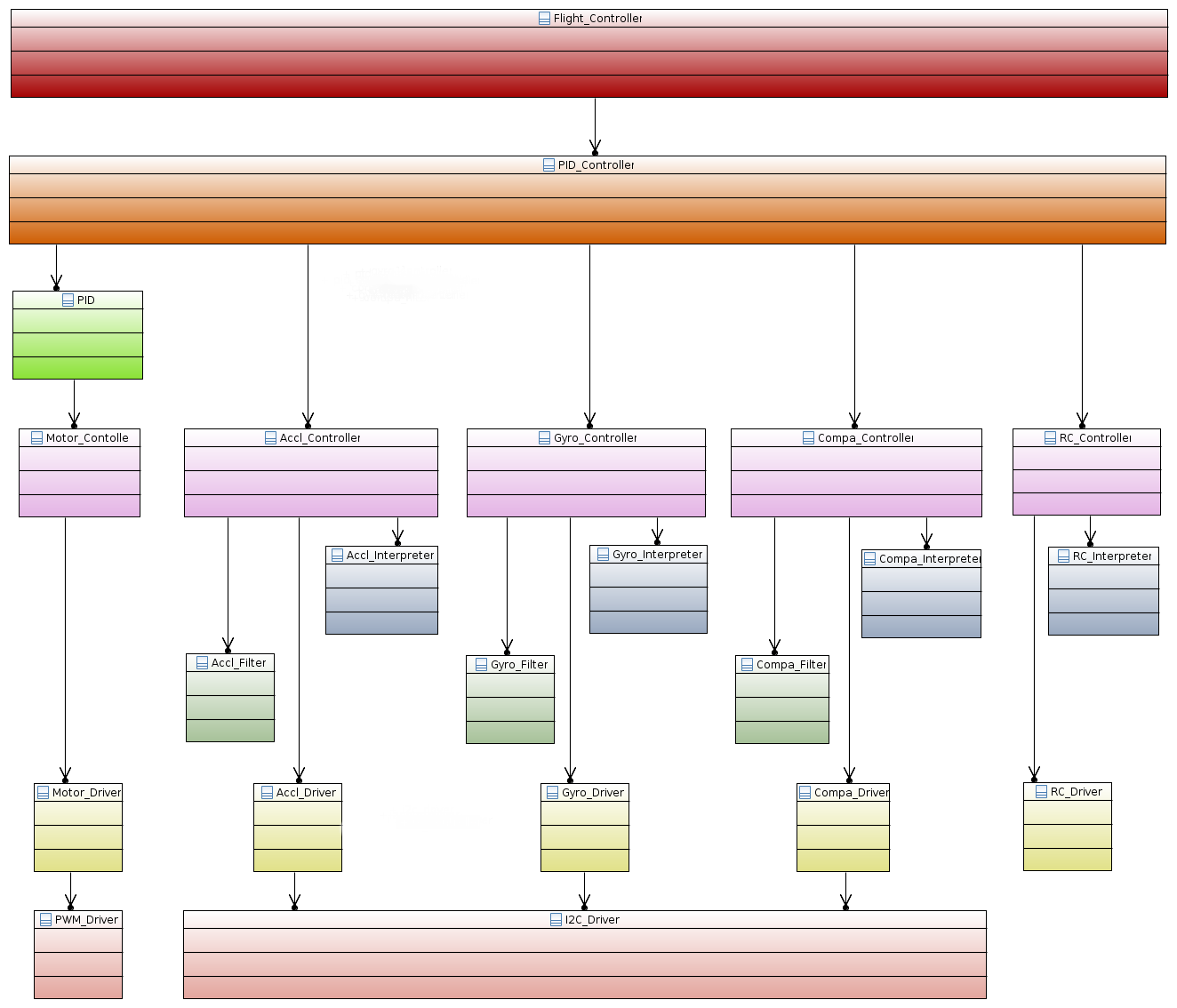


Abbildung 2 Domaine Model

## Definition of Graupners HoTT- SUMD- Signal

### Why SUMD

The “gr 16” receiver supports two different sum signals, “SUMO”- and “SUMD”- Signal.

The SUMO- Signal is an analog sum signal and is equal to a Puls position modulation whereas

the SUMD- Signal is a digital sum signal.

So the big advantage of SUMD is, that it is easy decode able.

### Definition

HoTT SUMD is implemented by a 115200 bit/s serial data stream. The data stream is generated by HoTT receivers. The transmitter generates a data frame at a data rate of 100Hz (10ms). Each data frame consists of a header followed by a data section representing the channel data and is concluded by a CRC checksum.

### Time Requirements

The serial connection needs to be set to 115200 Bit/s, 8 Databits, no Paritybit, 1 Stopbit. Each data frame is sent as a consistent data burst leaving minimal gaps less than 50µs between transmitted data bytes.

### Structure of a HoTT- SUMD frame

A single SUMD data frame comprises of three consecutive sections. SUMD\_Header, SUMD\_Data, SUMD\_CRC.

The SUMD\_Data section contains the channel data in sequential order. The number of channels to be transmitted can be up to 32. Each channel data is represented by a 16 bit word.

### SUMD\_Header section description

***Byte Byte\_Name Byte\_Value***

Byte 0 Vendor\_ID 0xA8

Byte 1 Status 0x01 or 0x81

Byte 2 Number of channels

### SUMD\_Data section description

Byte n\*2+1 High Byte of channel n

Byte n\*2+2 Low Byte of channel n

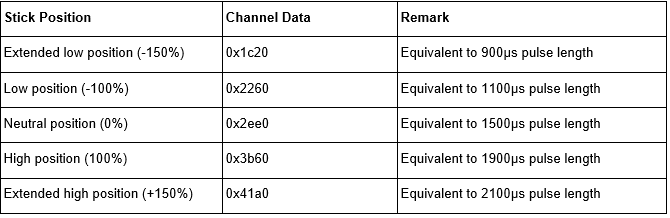
### SUMD\_CRC section description

Byte (N\_Channels+1 )\*2+1 High Byte of CRC

Byte (N\_Channels+1 )\*2+2 Low Byte of CRCderived

### Channel data interpretation

Each channel data is represented by an unsigned 16 Bit Word. The data range is derived from the pulse length for standard servos.



## Implementation of the SUMD Parsing

Since is a serial Format it can be read directly from a UART.

Luckily Altera is offering an RS232 UART IP Core, which can be added to our SoPC using Qsys.

It only requires two additional GPIO Pins, for receiving or transmit serial data.

Reading and Controlling the UART will be Part of the UART Driver.

The UART has to be initiated with the following settings, to receive a SUMD-Frame:

- 115200 Baud

- No Parity

- 1 Stop Bit

Every received Byte has to be interpreted according to the Definition of the SUMD format,

which was described in the previous Section "//TODO Name of section".

Following Steps are executed by the RC interpreted Controller:

### Saving raw SUMD-Frame Bytes from the UART

The SUMD-Controller has to wait for a new SUMD-Frame. A frame starts if the value of a received Byte equals the VendorID. After that, the following Bytes will be saved in an Array.

The size of the Array will be equal to the frame this can be calculated with:

SUMD-Frame length = SUMD Header length + Number of Channels \* 2 + CRC length)

### Interpreting the received SUMD-Frame

According to the SUMD format description, every Byte has its own specific purpose.

The actual received RC-commands are sliced into a High Byte and a Low Byte,

thus it is necessary to unroll both Bytes to a 16 Bit Integer. Every Channel value will be stored in an Array, which is accessible in a struct including all additional Data of the SUMD-Frame.

### UART Driver

This driver will offer functions to initiate and read the RS232 UART IP Core.

It is also possible to check if a new Byte was read. This is highly recommended

if only one Byte will be read from the UART. The driver is divided in a source file "b\_uartriver.c" and a header file "b\_uartriver.h". The UART can be selected with an enumeration, which is defined in the header file of the driver.

### SUMD-Frame-high



Abbildung 3 SUMD-Frame-high

### SUMD-Frame-low



Abbildung 4 SUMD-Frame-Low

## Second Flight Test

### Organisation

The Organisation was similar to to the first flight test. For security reasons we went to a Model flying site. Again the control was taken over by Mr. Hoeft because his Model flying insurance.

### Changes at the X-Copter since the last Flight test

To mount the motors without the rubber vibration damper the mount points had to be modified slightly.

The motor direction of rotation has been checked.

### Attempt 1

Even though the conditions were bad (it was very windy this day) the X-Copter flew very well. The Test was a success.

### Conclusion

In quadrocopter form the X-Copter does only use about 50% of its power, so there is a lot of capacity for payload left.

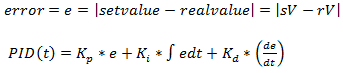
Now it’s proven that the Model is able to fly and we can start to work seriously with our own flight controller.

## PID Regulator

To guarantee a stable hovering UAVs often employ PID regulators and so do we in the X-Copter.

The general functionality works like this: The PID has two input parameters and one return value. The input parameters are the set point and the real value whereby the set point is the value, the to be regulated part (in our case one of the motors), should reach and the real value is the value, the motor has currently. This two values are subtracted and the resulting difference is known as the error. Now each part of the PID regulator manipulates the error respectively and returns it. The acronym PID stands for proportional, integral and differential and means that all of the parts are accumulated into one PID regulator. The factors have to be set individually for each project and setting by hand and trial and error. The mathematical formula is shown in Formel 1 PID-Formula.

Formel 1 PID-Formula



In the following these three parts of the PID regulator will be explained.

## The Proportional Part

This is not a real regular for that matter. It's more a proportional amplifier that enforces the error by the factor. This is the most influencing part of the PID because it reacts very fast. For better understanding here is an example:

Let's say we want to have a motor run at 100 RPM. When we start the P-Regulator and the motor is not running the error will be and it will return to the motor.

## The Integral Part

This is slowest part of a PID because it's an integral meaning it's return value increases or decreases linear over time depending on the error being positive or negative. The bigger the positive errors are the faster the return value increases, the bigger the negative errors are the faster the return value decreases.

For comparison to the P-Part: If the error is 0, the P-Part will return 0 but the I-Part return value won't change. So the I-Part serves mostly to correct the residual error of the P-Part.

## The Differential Part

The D-Part is the most nervous part of the PID because it reacts to temporal changes. So the faster the real value (or sensor data) changes the higher is the return value. This part does not depend on a This is useful for fast response what is important in our system because we want to stabilize it at all the time. Let's say we hover stable and a wind gust is hitting the XCopter. The P-Part will recognize and steer against it in proportion to the error – but this to slow and it would move the UAV. The I-Part won't do anything helpful in this case because the time period is too short. The D-Part will detect a fast and big change of the real value so it will accelerate the motor very fast to correct the error and steer against the wind gust.

Evaluation by the team

In the last three sprints we reached some important milestones like the first successful flight or the successful transmission from the RC Transmitter to the NiosII-processor. On the way to this goals, our team worked very well together and everybody solved his own tasks very carefully. So the team was able to finish a lot of small tasks in a short time. However, at some tasks, we have also lost a lot of time. For example, the integration of the additional USB-controller, the error analysis of the existing power supply and the installation of the Wi-Fi-Driver on the embedded Linux. Another problem was our development methodology - Scrum was a loss of time. We decided to focus on the software development and not on scrum anymore. Thus we should be able to get more work done, in a shorter period of Time. In order to reach our goal.

# Lessons Learned

## Scrum

Scrum is a very good agile Software development tool. But for our team was Scrum a big overhead because of his planning for the future. We needed to much time to build up User stories or tasks. In a big team of 9 people it was always a hard discussion until every person was clear. To maintain our Jira tool a lot of time was wasted. And we decided to continue without Scrum.

## Long term calendar management

Planning of team meetings was very hard because every person had a lot of work to do during the semester. To find a date were all team members had time was impossible. So we decided some important thinks in a smaller group and then its come to communication problems

## Quick task can not be made quick

We thought some task could be done quick. But in a hardware environment there are more then one thing to consider. So we needed more time to finished some tasks. In the future we planning in smaller steps.

Abbildung 1 NAZ V2 21

Abbildung 2 Domaine Model 26

Abbildung 3 SUMD-Frame-high 30

Abbildung 4 SUMD-Frame-Low 30

# Literaturverzeichnis

Team, B. (2015). *Final\_Report\_Bumbelbee .* Ulm: HS-Ulm.