Project Name:	Valiquad
Document Title:	Requirements Hardware
Autor(s):	Manuel Stübler
Created on:	23.10.10
Last modified:	04.12.10
Version:	v1.5
Status:	| | in progress | | presented |x| closed
Located at:	/docs/se/req-hw.odt

Revision History:

#	Date	Version	Section	Description	Author	Status
1	23.10.10	v1.0	all	First creation	Manuel Stübler	closed
2	29.10.10	V1.1	3	Added requirements	Manuel Stübler	closed
3	01.11.10	V1.2	3	Added requirements	Manuel Stübler	closed
4	02.11.10	V1.3	3	Added requirements	Manuel Stübler	closed
5	05.11.10	V1.4	3	Added requirements	Manuel Stübler	closed
6	04.12.10	V1.5	3	Added requirements	Manuel Stübler	closed

Table of Contents

List of Figures

List of Tables

1 Introduction

This document describes and lists the requirements for the lower layers of the Valiquad project. Those layers include the communication protocol that is used between the analyzing software (Valiquad) and the helicopter (Quadrocopter), the data synchronization and storage within the data model of the analyzing software and also the software that runs on the micro-controller of the helicopter to control the sensors and the parameters for the controllers.

2 Brief Description

The requirements are described and stated within a unified table, shown below.

ID	Unique identification for the requirement
Title	Short title
Author (Date)	Name of the author (date of creation)
Description	The description of the requirement in detail
Test	A short explanation on how this requirement can be tested
Solution	An abstract idea on how the requirement can be implemented
Links	References to other requirements

Table 1: Reference Requirement

3 Requirements

ID	R1
Title	Communication Protocol
Author (Date)	Manuel Stübler (23.10.2010)
Description	The analyzing software shall communicate with the Quadrocopter using a standard radio-frequency technology. This technology is given by the currently mounted communication interface on the helicopter, which is an XBee module that uses the ZigBee communication standard. Another XBee module is attached to the host through a serial RS-232 communication port.
Test	While sending a packet from one instance to another and returning the same packet to the sender, the system shall check the sent and received packet for equality.
Solution	ZigBee allows to send a stream of bytes from one instance to another via a standardized radio-frequency protocol. By creating an appropriate application layer protocol that uses a small header and a following body, this stream of bytes is put together within a packet. Such a packet is transmitted over the communication interface and interpreted by the receiver at once. For the communication with the serial port, Java provides the 'Java Communications API' (javax.comm), which will be used by the analyzing software to communicate with the attached XBee module.
Links	None

Table 2: R1 Communication Protocol

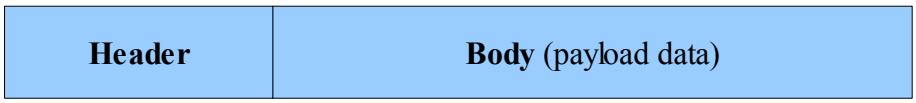

Figure 1: Structure of a physical packet (application layer)

ID	R1.1
Title	Data Integrity
Author (Date)	Manuel Stübler (23.10.2010)
Description	As the packets that are sent over the air can be accidentally modified by disturbance such as interference and other physical treatments, those packets have to be secured by a check sum. This check sum must guarantee a hamming-distance of at least 2, better 4.
Test	Send a packet with a wrong check sum and check if the receiver will detect the error.
Solution	A hamming-distance of 2 can be achieved by a simple parity check. For a hamming-distance of 4, calculating a CRC value might be a good choice. If there is a possibility to calculate CRC values for given packets with some kind of hardware-acceleration, this method shall be used instead of calculating the check-sum in a software-based fashion.
Links	R1

Table 3: R1.1 Data Integrity

ID	R1.2
Title	Data Synchronization
Author (Date)	Manuel Stübler (23.10.2010)
Description	By using the communication protocol described in R1, the analyzing software shall synchronize its database (data model) to the measured data on the helicopter. This synchronization shall implement a mechanism that allows soft real-time constraints. The data that is transmitted has to arrive at the analyzing software within a given time slot to ensure the usefulness of the sensor data. If the data arrives at a later time it still has some, but only little usage and is therefore also stored in the database.
Test	Sending loads of packets and calculating the average latency of the arriving packets. This latency has to be under a defined threshold.
Solution	The sensor values shall be send periodically one after another. The transmission rate must adhere to the constraints of the communication medium and the highest possible sample rate for the sensors.
Links	R1

Table 4: R1.2 Data Synchronization

ID	R1.3
Title	Packet Loss
Author (Date)	Manuel Stübler (23.10.2010)
Description	To fulfill the requirement R1.2 a mechanism for the handling of erroneous packets must be introduces. If the check sum reveals that an error occurred while transmitting a packet with sensor data from the helicopter to the analyzing software, it has to be discarded. This is necessary to not slow down the whole process of data transmission. Especially there will be no resending of an invalid packet, as this would cause the complete system to stumble. There will also be no acknowledge if a packet was successfully received by the host. Packets that are send the other way round by the analyzing software to the helicopter will be acknowledged by the receiver. If the acknowledge is not received by the sender within a defined time-slot, the analyzing software will resend the message. If the message was not acknowledged after the second retry, this means after sending it for the third time altogether, the analyzing software will assume the helicopter to be currently unavailable.
Test	Send erroneous packets from the helicopter to the analyzing software and check whether those packets are discarded without initiating any further steps. Shut down the helicopter and try to send a message from the analyzing software to the helicopter, check if the software will assume the helicopter to be unavailable after sending a packet for the third time.
Solution	Delete erroneous packets that are received from the helicopter within the analyzing software and do not initiate any kind of a resend. Define a special acknowledge packet that will be sent from the helicopter to the analyzing software if a packet was received successfully. If the acknowledgment is invalid, the host discards this packet and acts as if he didn't receive a packet at all.
Links	R1.2

Table 5: R1.3 Packet Loss

ID	R1.4
Title	Data Throughput
Author (Date)	Manuel Stübler (23.10.2010)
Description	The communication protocol must guarantee, that the measured data of all available sensors can be transmitted in (soft) real-time with a sample rate of at least one value per 100ms per sensor. The sensor values shall be put together in a so-called bundle, this bundle holds the values of all sensors for the same time-stamp. If necessary, several successive bundles can be sent within one packet, to ensure an adequate throughput. However the maximum latency, not regarding the time for transmitting a packet, must not exceed 1 second. This means the value of a sensor must be send within 1 second after it has been captured.
Test	Analyze the traffic of the system and check if the given constraints can be fulfilled. The oldest measured value of a sensor within a packet must not be older than 1 second when the packet is sent by the helicopter.
Solution	Regarding the highest possible sample rate of one value per 10ms per sensor, a maximum number of 100 successive bundles can be put together in one packet. A good choice might be 10 bundles per packet, which means packets will be sent every 100ms and hold up to 10 values per sensor.
Links	R1.2

Table 6: R1.4 Data Throuhput

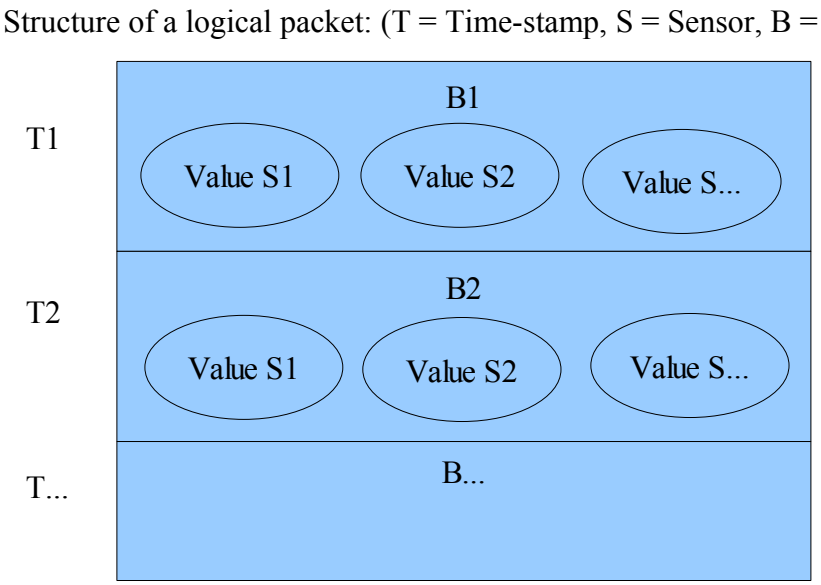

Figure 2: Structure of a logical packet

ID	R1.5
Title	Flight Control
Author (Date)	Manuel Stübler (05.11.2010)
Description	The calculation and transmission of packets must not block the flight control longer than allowed. It has to be guaranteed that the time used for the protocol handling does not exceed the time-slice that is given to the task.
Test	Calculate the time used for the protocol handling and check if it is smaller than the time-slice that is assigned to the task.
Solution	Keep the calculation time as short as possible, e.g. by calculating the check-sum for a packet in hardware, if possible.
Links	R1

Table 7: R1.5 Flight Control

ID	R1.6
Title	Dynamic approach
Author (Date)	Manuel Stübler (04.12.2010)
Description	The protocol shall be made as dynamic as possible. The parameters that can be set and observed will change over time, but the protocol and the application working with shall be able to handle any future parameters that are added to the system.
Test	Add yet unknown parameters to the system and check if the protocol can handle them without knowing any further things about them.
Solution	To allow this dynamic mechanism to work properly, for example so-called parameter IDs can be introduced. Those IDs must describe such a parameter uniquely and hold all necessary information about it, for example the data type and if it is a read-only parameter or not.
Links	R1

Table 8: R1.5 Dynamic apporach

ID	R2.1
Title	Buffering Target
Author (Date)	Manuel Stübler (29.10.2010)
Description	To temporarily store the sensor values, before they are going to be transmitted, a buffer on the target (the helicopter) might be necessary. This buffer must be big enough to store all collected values from the sensors, before they can be transmitted. The lowest possible rate of data transmission has to be calculated from the available buffer size within the memory on the helicopter.
Test	Set the highest sample rate with the lowest transmission rate and check, whether none of the sensor values will be lost within the depth of bits and bytes.
Solution	Check for the available memory that can be used for such a buffer and calculate backwards the possible sample and transmission rates.
Links	R2.2

Table 9: R2.1 Buffering Target

ID	R2.2
Title	Buffering Host
Author (Date)	Manuel Stübler (01.11.2010)
Description	The data received from the helicopter has to be stored in a database within the analyzing software (the host). Therefore an appropriate data structure is required, to efficiently store all sensor values for a given time-stamp. This database has to be limited to not cause a memory leak. This limit must ensure, that enough values can be stored in the database to correctly analyze the flight dynamics.
Test	Check if the defined data structure stores the values efficiently without having redundancy. Also check for a useful interface to retrieve the data from the data base that matches the requirements of the view and controller objects.
Solution	A possible solution could be a hash-map that uses time-stamps as key values and stores all sensor values with the same time-stamps within one entry in this hash-map. As an indicator for the limit of the database, the time-stamp can be used. For example values that are older than 60 seconds will be discarded.
Links	R2.1, R4

Table 10: R2.2 Buffering Host

ID	R3
Title	State Machine
Author (Date)	Manuel Stübler (29.10.2010)
Description	The software for the helicopter shall be modeled and implemented throughout a state machine, which will be written entirely in C. There is already an existing state machine, which therefore has to be adapted for the new purposes of the protocol and the system.
Test	Run the software and check for the correct states within pre-defined scenarios.
Solution	The state machine will be implemented in C using the given structure of the current software status.
Links	None

Table 11: R3 State Machine

ID	R4
Title	Model-View-Controller
Author (Date)	Manuel Stübler (29.10.2010)
Description	The software for the analyzing tool itself will be written in Java. The central architecture of the system is the Model-View-Controller (MVC). This pattern is also used to define the interface between the hardware-related part of the tool, that is concerned with the data transmission (data model) and the software-related part, which ought to create an adequate user interface (view and controller).
Test	Putting both parts together and checking for the correct behavior of the system.
Solution	As Java is an object oriented programming language, it makes it easy to encapsulate functionality and create usable interfaces to abstract the principles behind an implementation. This mechanism will be used to define the model and therefore the hardware-related part, and on the other hand the view and controller will be defined for the software-related part.
Links	None

Table 12: R4 Model-View-Controller

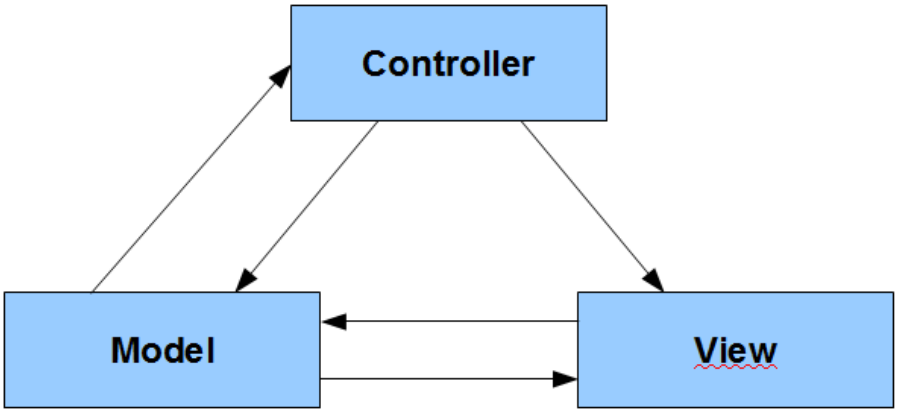

Figure 3: MVC