

# Entwurfsdokumentation der Studienarbeit 4 „Spionat“

---

Version 1.0 (Release)

HTW Chur

Telekommunikation und Elektrotechnik

Studienarbeit 4

Ausgabe durch Dozenten: Ende März 2008

Abgabe durch Studierende: Mitte Mai 2008

Autor: Giorgio Crameri

## 1 Zusammenfassung

In diesem Papier wird erklärt, wie der Spionat softwaremässig entworfen wurde. Das System des Spionats, welcher auch als Drohne bezeichnet wird, besteht aus der in der Anforderungsdokumentation beschriebenen WLAN-Kamera, die unabhängig vom Gefährt mit dem steuernden Laptop kommuniziert. Für die Kamera wurde keine Software implementiert. Aus diesem Grund wird darauf nicht weiter eingegangen.

Hingegen ist, um das Vehikel zu steuern, ein umfassendes Gui implementiert, mit welchem sowohl die Geschwindigkeit wie auch die Fahrrichtungen vorgegeben werden können. Das Gui wird in Java erstellt, die Software auf dem Mikrokontroller des Spionats in C. Durch Java werden die C-Funktionen angesprochen. Die Kommunikation basiert auf einer virtuellen RS-232 Schnittstelle. Die Befehle werden „verpackt“ und als TCP/IP-Pakete per WLAN verschickt.

---

## 2 Inhaltsverzeichnis

1	Zusammenfassung .....	2
2	Inhaltsverzeichnis .....	3
3	Einführung .....	5
4	Grobdesign .....	6
4.1	Architektur der Software .....	6
4.2	Schnittstellen der Units .....	7
5	Detaildesign .....	9
5.1	Algorithmen.....	9
5.2	Design der Units .....	10
5.3	Unit-Spezifikationen .....	11
5.3.1	Keyboard .....	11
5.3.2	Gui.....	11
5.3.3	Velocity .....	12
5.3.4	Ugv .....	12
5.3.5	Bend .....	12
5.3.6	SerialWriter .....	12
5.3.7	SerialReader .....	12
5.3.8	Connection .....	12
6	Version und Revision History .....	13
7	Glossar .....	13
8	Literaturverzeichnis.....	13
9	Abbildungsverzeichnis .....	14
10	Anhang.....	15
10.1	Sektion 1: Serielle Kommunikation .....	15
10.1.1	Protokoll.....	15
10.1.2	Aufbau .....	15

---

10.1.3	Erkennungszeichen .....	15
10.1.4	Zieladresse (hex) .....	15
10.1.5	Adressenbeschreibung .....	16

### 3 Einführung

Im folgenden Text wird auf das Grobdesign und das Detaildesign des Programmes eingegangen, wobei ausschliesslich der Java-Code berücksichtigt wird. Die Schnittstelle zu den C-Funktionen, d.h. mit welchen Parametern sie aufgerufen werden können, wird im Anhang Absatz 10.1 erklärt. Im Grobdesign Absatz 4 werden allgemeine Verläufe und die Strukturen der Software dargestellt. Im Detaildesign Absatz 5 die Algorithmen und Klassen wie auch die Verweise auf die einzelnen Funktionalitäten. Details zu den Methoden entnimmt man den Kommentaren im Code.

## 4 Grobdesign

### 4.1 Architektur der Software

Eine gute Architektur der Software ist beim Projekt „Spionat“ unerlässlich, da es äusserst komplex ist. Diverse Details auf verschiedenen Ebenen, die einwandfrei zusammenarbeiten sollen, müssen geklärt werden. Wie in Abbildung 1 dargestellt, wird auf dem Laptop die graphische Steueroberfläche des Spionats laufen. Sie ist in Java implementiert. Die Befehle zum Fahrzeug und seine Rückmeldungen werden durch diese Schnittstelle übergeben bzw. dargestellt. In der Grafik wird dies mit den Blockdiagrammen aufgezeigt. Gleichzeitig wird im Internet Explorer das Bild der Kamera, welche sich auf dem Fahrzeug befindet, dargestellt. So kann man die Drohne fernsteuern und gleichzeitig ihre Umgebung und ihre Bewegungen überwachen.

Die Drohne kann sowohl mit der Maus über das graphische Interface gesteuert, als auch mit den Pfeiltasten der Tastatur gesteuert werden. Es gibt dabei nicht nur die Bewegungen links-rechts oder vorwärts-rückwärts sondern auch diagonale, z.B. vorne-links. Daraus ergeben sich acht verschiedene Fahrrichtungen, die gefahren werden können. Dieser Teil der Software greift auf eine gemeinsame Steuerklasse des Gui zu (vgl. dazu Abbildung 2: Klassendiagramm).

Die Kommunikation zwischen Laptop und Spionat findet über ein WLAN Ad-Hoc-Netz statt. Das Java-Interface auf dem Laptop kommuniziert mit einem virtuellen Lantronix-Comport, dem sogenannten Comportdirector. Dieser stellt eine virtuelle serielle Verbindung (RS-232) zum Spionat her.

Beim Spionat lösen die ankommenden Befehle die Ausführung von C-Funktionen aus. Somit kann das Fahrzeug bewegt und überwacht werden. Anfragen haben eine Antwort des Spionats zur Folge, welche die gleiche Route zurück zum Laptop nehmen bis zum Gui, wo sie vom Benutzer wahrgenommen werden können. Auch die Bilder der Kamera werden über das WLAN übertragen, gelangen aber nicht zum Gui, sondern zu einem Webbrowser. Sie kommen über einen separaten Weg, d.h. sie gehen nicht über die virtuelle serielle Schnittstelle und werden nicht von Java-Methoden verarbeitet. Sie werden von einem Active-X-Baustein im Webbrowser empfangen und können durch diesen dargestellt werden. Der Baustein und die ganze Software für die Kamera stammen vom Kamerahersteller.

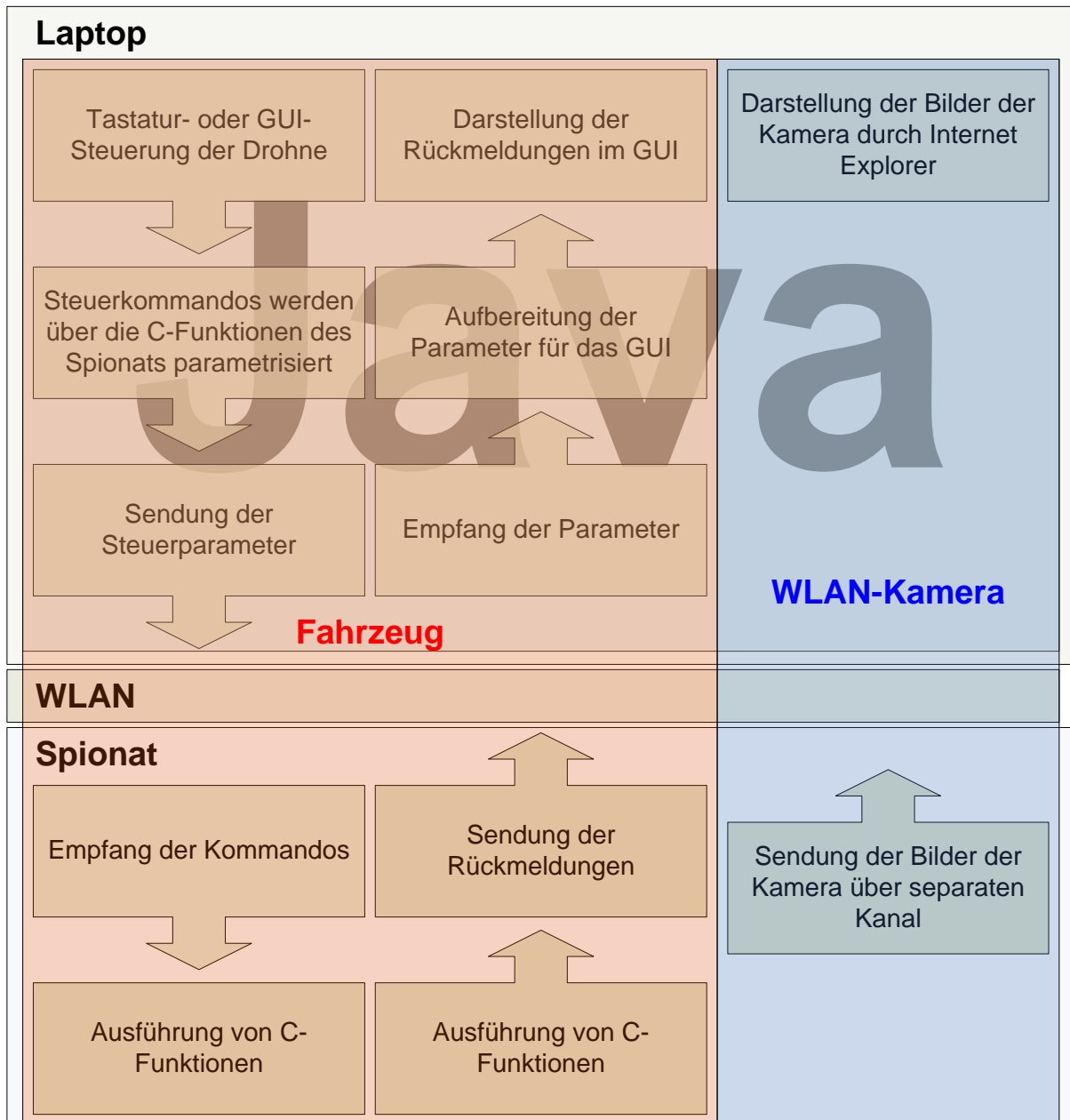


Abbildung 1: Architektur der Software

## 4.2 Schnittstellen der Units

Wie man in Abbildung 2 sieht, gibt es beim Klassendiagramm etliche Schnittstellen. Erwähnenswert ist, dass die Kommunikation über die serielle Schnittstelle mittels ASCII-Codes stattfindet. Die genauere Codierung der Befehle kann im Anhang unter Sektion 1: Serielle Kommunikation gefunden werden. Dort findet man auch, wie der Spionat angesteuert werden kann.

Innerhalb des Java-Code-Komplexes werden die Befehle des Bedieners über Buttons oder die Tastatur in die Befehle für die C-Funktionen übersetzt. Umgekehrt werden darin die ankommenden ASCII-Codes der seriellen Schnittstelle für die visuelle Darstellung aufbereitet.

Wie im Kapitel Architektur der Software Absatz 4.1 erwähnt, läuft die Kommunikation der Kamera über einen separaten Weg. Dieser funktioniert ausschliesslich mit Werkzeugen des Kameraherstellers. Auf diese Werkzeuge wird deshalb nicht näher eingegangen.

## 5 Detaildesign

### 5.1 Algorithmen

Den Ablauf der Algorithmen kann man in Abbildung 1 erkennen. In der Architektur der Software sind Pfeile eingetragen, welche die Richtungen der Steuerflüsse innerhalb des Systems beschreiben. Es gibt nur zwei Wege: zum Spionat hin und von ihm zurück. Der Benutzer kann nur die Geschwindigkeit und die Fahrtrichtung des Spionats via Buttons ändern. Von den Mitgliedern der Klasse Gui wird die untenstehende Instanz der Klasse Spionat mit den treffenden Methoden aufgerufen. Diese sendet die Befehle als ASCII-Code dem Spionat, welcher sie ausführt. Weitere Details darüber findet man im Abschnitt Unit-Spezifikationen und in der Sektion 1 des Anhangs.

Anstatt Steuerbefehle dem Spionat zu schicken, kann man auch seine Zustände abfragen. Diese Möglichkeit ist im Gui automatisiert, der Benutzer kann sie nicht steuern: Der Batteriezustand wird ständig abgefragt und die Rückmeldungen der Drohne werden auf dem Gui dargestellt. Weitere Details darüber siehe weiter unten.

## 5.2 Design der Units

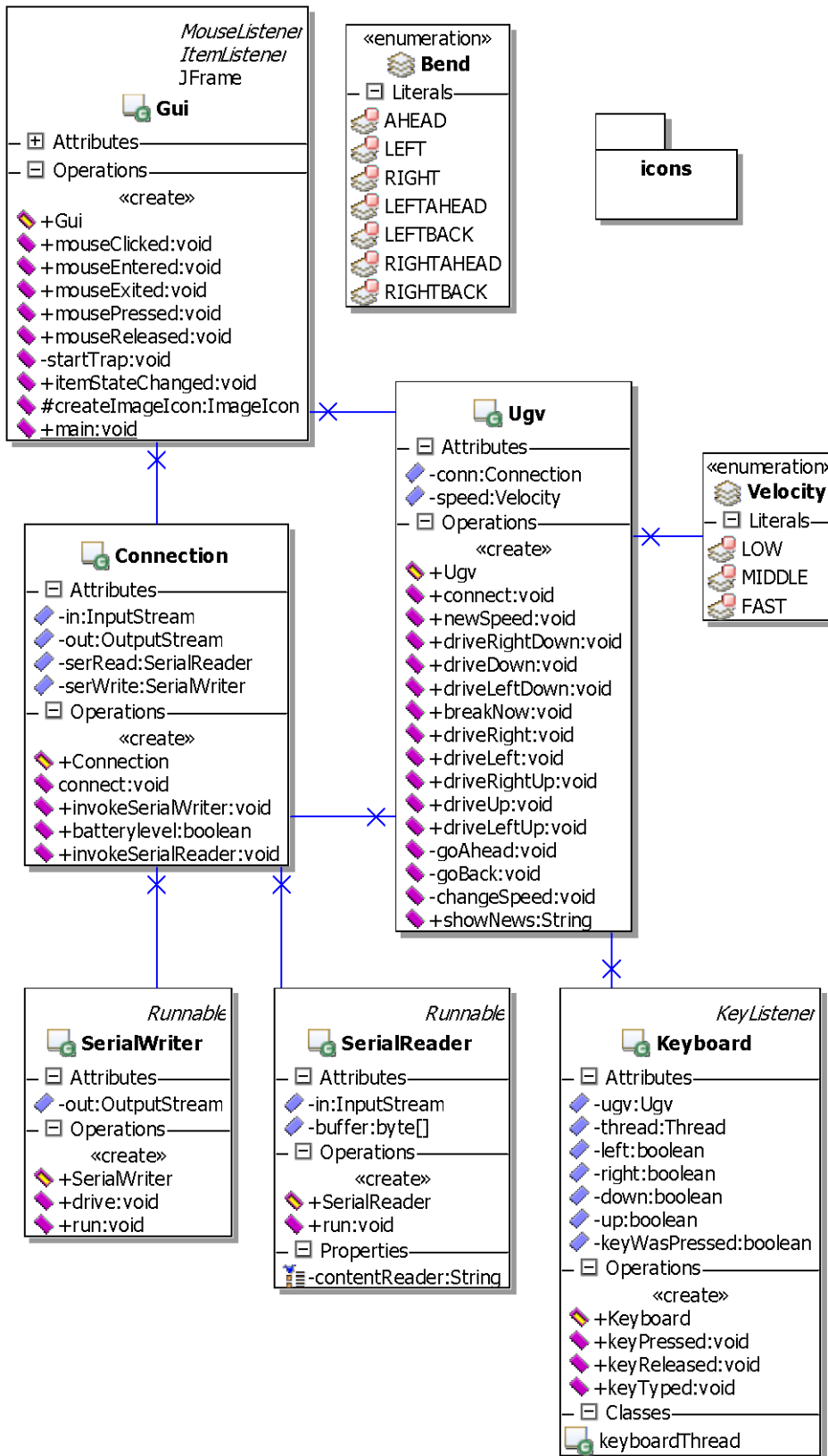


Abbildung 2: Klassendiagramm

Beim Design der Units wird nur auf den Java-Code auf dem Laptop eingegangen. Da die Programmiersprache C nicht objektorientiert ist, kann man daraus kein Klassendiagramm erstellen. Auch Sequenzdiagramme sind überflüssig, da jede C-Funktion genau eine Aktion ausführt, wie z.B. „Fahre mit dem linken Schrittmotor für 10ms vorwärts.“ Einen Überblick über die Tätigkeiten der C-Funktionen bietet Sektion 1: Serielle Kommunikation des Anhangs Absatz 10.1. Dort werden die detaillierten Steuermöglichkeiten der Drohne vollständig erklärt.

Die im Klassendiagramm abgebildeten Units können in drei Gruppen unterteilt werden. Die Gruppe „Benutzerinterface“ beinhaltet die Bausteine, die direkt mit dem Benutzer zu tun haben, also die Klasse „Keyboard“ und die Klasse „Gui“. Durch sie kann der Benutzer weitere Bausteine indirekt benutzen, die seine Befehle näher zu den C-Funktionen des Spionat bringen.

Die Steuerklasse „Ugv“ vermittelt zwischen dem Gui, dem SerialReader und dem SerialWriter. Die Steuerbefehle werden so bearbeitet, dass sie von den C-Funktionen verstanden werden.

SerialReader und SerialWriter bilden die äussersten Klassen gegenüber dem Spionat. SerialReader empfängt den seriellen InputStream und SerialWriter erzeugt den OutputStream.

## 5.3 Unit-Spezifikationen

Hier werden die einzelnen Units im Detail erklärt.

### 5.3.1 Keyboard

Mit der Klasse „Keyboard“ kann man die Drohne anhand der vier Pfeiltasten up, down, left und right steuern. Erwähnenswert ist, dass – wenn gleichzeitig up und left gedrückt werden – das Vehikel eine Linkskurve macht. „Keyboard“ ruft die Fahrmethoden der Klasse Ugv je nach Richtung auf.

### 5.3.2 Gui

Mit dem Gui kann man den Spionat mit der Maus steuern: Es gibt einen Button für jede Richtung. Zusätzlich gibt es Radio-Buttons. Damit kann man die Geschwindigkeit zwischen low, middle und fast einstellen. Im Gui kann auch das Bild der Kamera des Spionats angesehen werden. Weiter gibt es - wie in der Anforderungsdokumentation (1) gefordert wird - im Gui einen Thread, welcher in regelmässigen Abständen kontrolliert,

---

ob die Batteriespannung noch reicht. Der Thread dient auch dazu, den Spionat am Fahren zu halten, solange die Verbindung besteht. Falls es einen Verbindungsabbruch gibt, hält das Vehikel nach 2,5 Sekunden an. Das heisst, man muss mindestens alle zwei Sekunden einen Befehl schicken, damit der Spionat merkt, dass er noch Kontakt mit dem Laptop hat. Dies erledigt der Thread, indem er den Batteriezustand abfragt. Im Konstruktor des Gui werden alle anderen Klassen instanziiert.

### **5.3.3 Velocity**

Die Enumerationsklasse „Velocity“ dient zur Festlegung der zu fahrenden Geschwindigkeit. Es gibt drei Ausprägungen: low, middle und fast.

### **5.3.4 Ugv**

Die Klasse „Ugv“ enthält die Steuermethoden des Spionat. Für jede Fahrtrichtung und Geschwindigkeit gibt es eine Methode, welche auch die Verbindung zu ihr wieder herstellen.

### **5.3.5 Bend**

Bend ist wieder eine Enumerationsklasse. Sie dient zur Bestimmung der zu fahrenden Kurve. Auf diese Weise kann man die richtigen Einstellungen für die Geschwindigkeit und den Kurvenbogen wählen.

### **5.3.6 SerialWriter**

SerialWriter ist dazu da, um die Befehle an den Spionat zu schicken. Es ergibt einen Outputstream aus dem Laptop.

### **5.3.7 SerialReader**

SerialReader dient zum Empfang der Nachrichten und Rückmeldungen des Spionats. Diese werden dann in der Klasse „Ugv“ weiterverarbeitet und im Gui dargestellt.

### **5.3.8 Connection**

Die Klasse „Connection“ baut eine Verbindung zwischen Laptop und Drohne auf. So kann am bequemsten eine abgebrochene Verbindung wieder hergestellt werden.

Um einen tieferen Einblick in die einzelnen Methoden zu erhalten, sei auf den Source-Code verwiesen. Oberhalb jeder Methode wird erklärt, was diese gerade macht. Tiefer wird in diesem Dokument darauf nicht eingegangen.

Die Steuerungsmöglichkeiten des Spionats beschränken sich in dieser Applikation auf das notwendigste. Selbstverständlich könnten aber noch diverse zusätzliche Einstellungen und Erweiterungen vorgenommen werden. Einen Überblick der Möglichkeiten kann dem Anhang entnommen werden.

## 6 Version und Revision History

Mit dieser Version 1.0 (Release) wird das Code geschrieben. Vorher gab es eine interne Version 0.8, die vom Verantwortlichen Giorgio Crameri erstellt worden war. Diese wurde von allen Projektmitarbeitern korrigiert. Die Änderungen wurden vom Verantwortlichen zusammengetragen und durchgeführt. Gerade vor der Dokumentübergabe an Herrn Professor Bonderer korrigierte eine Übersetzerin die Grammatik und den Styl des Dokumentes. So entstand Version 0.9. Das Feedback auf Version 0.9 wurde von ihm selbst gemacht. In dieser Version 1.0 (Release) ist auch sein Feedback berücksichtigt worden.

## 7 Glossar

*Spionat, Drohne:*

Spionat und Drohne werden synonym verwendet. Damit ist die gesamte Hardware gemeint.

*Vehikel:*

Mit Vehikel ist der Spionat ohne Kamera gemeint, also nur die sich bewegende Einheit mit ihren Steuer- und Kommunikationseinheiten.

## 8 Literaturverzeichnis

1. **Reto, Guadagnini und Sutter, Isabell.** *Anforderungsdokumentation.* 2008.
2. *Sun Developer Network.* [Online] [Zitat vom: 15. 05 2008.]  
<http://java.sun.com/javase/index.jsp>.
3. *Virtual Serial Port.* [Online] [Zitat vom: 15. 05 2008.] <http://www.rxtx.org/>.

Die Quellen werden bei der Implementierung des Codes verwendet und werden deshalb im obigen Text nicht explizit erwähnt.

## **9 Abbildungsverzeichnis**

Abbildung 1: Architektur der Software

Abbildung 2: Klassendiagramm

## 10 Anhang

### 10.1 Sektion 1: Serielle Kommunikation

#### 10.1.1 Protokoll

RS232, 9600 Baud, 8 Bit, 1 Stop, keine Flusssteuerung, keine Parität

Flusssteuerung und Parität sind nicht nötig, da die Kommunikation nicht physikalisch über RS-232 stattfindet, sondern virtuell über TCP/IP. Und TCP beinhaltet sowohl Flusssteuerung wie auch Parität.

#### 10.1.2 Aufbau

Ein Datenpaket besteht immer aus 5 Zeichen im Format EAADD.

E = Erkennungszeichen (alle Zeichen aus 7 Bit ASCII-Code)

AA = Adresse (0-255 hexadezimal / 0-9 und A-F)

DD = Daten (0-255 hexadezimal / 0-9 und A-F)

#### 10.1.3 Erkennungszeichen

„?“: Anfrage (0x3F)

„>“: Antwort (0x3E)

„#“: Befehl (0x23)

#### 10.1.4 Zieladresse (hex)

00: Batteriespannung (read Only RO)

01: Gerätestatus (RO)

02: Sicherheitstimer

03: Sicherheitszeit

04: Motorstatus R

05: Motorstatus L

06: Fahrstatus

07: Schrittregister R

08: Schrittregister L

09: Geschwindigkeit R

0A: Geschwindigkeit L

0B: Zusatzoptionen

## 10.1.5 Adressenbeschreibung

### 10.1.5.1 Batteriespannung (read Only RO) [0]

Gibt die aktuelle Betriebsspannung des Fahrzeuges zurück.

Spannung pro Bit = 38.16mV

Es muss so genau gerechnet werden, da die Auflösung von 256 Schritte es benötigt.

Sonst wird man zu ungenau.

Die Auflösung des AD-Wandlers beträgt  $3.3V / 256 = 12.89mV$ . Der Spannungsteiler hat ein Verhältnis von 2.96. Somit ergibt sich eine maximale Spannung von 9.77V am Eingang. Das dürfte für 6 Batterien mit nominal 1.2V reichen ( $6 * 1.2V = 7.2V$ ).

(Die Kamera schaltet bei ca. 5.8V und die Steuerung bei ca. 4.5V ab.)

### 10.1.5.2 Gerätestatus (RO) [0]

Gibt den aktuellen Status des Gerätes zurück.

Bit 0 = Keine Fahrfreigabe / Sicherheitstimer == 0

Bit 1 = Warnung Batteriespannung (Schwellwert bei 131 entspricht ca. 5V)

### 10.1.5.3 Sicherheitstimer [255]

Der Sicherheitstimer wird jedes Mal, wenn eine Kommunikation stattgefunden hat, auf die Sicherheitszeit gestellt. Der Zähler wird alle 10ms um 1 reduziert. Ist der Zähler bei 0 angelangt, werden die Motoren sofort gestoppt.

### 10.1.5.4 Sicherheitszeit [255]

Die Sicherheitszeit hält die Zeit in einem Mehrfachen von 10ms, welche das Fahrzeug ohne Kommunikation mit dem Master noch bewegen darf (siehe Sicherheitstimer).

### 10.1.5.5 Motorstatus R / Motorstatus L [0]

Hält den Status des betreffenden Motors

Bit 0 = Kontinuierliches Fahren (geschwindigkeitsregisterabhängig)

Bit 1 = Rückwärts

### 10.1.5.6 Fahrstatus [0]

Hier können verschiedene Fahroptionen eingestellt werden.

Bit 0 = Freigabe Schrittfahren (Schritte in Schrittregister werden abgefahren, geschwindigkeitsabhängig)

Bit 1 = Motor permanent unter Spannung („Bremse“)

Bit 2 = Sicherheitszähler inaktiv (nur für Testzwecke verwenden!)

**10.1.5.7 Schrittregister R / Schrittregister L [0]**

Hält die Anzahl Schritte, welche noch zu Fahren sind.

**10.1.5.8 Geschwindigkeit R / Geschwindigkeit L [10]**

Hält die Zeitbasis, bis ein neuer Schritt ausgeführt wird (Zeitbasis: 10ms). Je kleiner die Zeit, desto schneller fährt das Fahrzeug.

**10.1.5.9 Zusatzoptionen [0]**

Bit 1 = Kontinuierliches Hupen

Bit 2 = Piepen

Bit 3 = Warnung (Piepen) bei niedrigem Batteriestand