

# Beschreibung Fahrzeug und ucboard

**PS Echtzeitsysteme**

Rev. 0+ (Entwurf), Stand 09.11.2017



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

PS Echtzeitsysteme  
FG Echtzeitsysteme  
FG Regelungstechnik und Mechatronik




---

# Inhaltsverzeichnis

<b>I. Anwenden</b>	<b>3</b>
<b>1. Fahrzeuge</b>	<b>4</b>
1.1. Fahrgestelle . . . . .	4
1.2. Aktorik . . . . .	4
1.2.1. Lenkung . . . . .	4
1.2.2. Fahrmotor (Fahrtenregler) . . . . .	5
1.3. Sensorik . . . . .	7
1.3.1. Ultraschall . . . . .	7
1.3.2. Hall-Sensor . . . . .	7
1.3.3. IMU . . . . .	8
1.3.4. Magnetometer . . . . .	8
<b>2. Kommunikation</b>	<b>11</b>
2.1. Prinzip . . . . .	11
2.2. Hardware . . . . .	12
2.3. Befehle . . . . .	13
2.3.1. Übersicht . . . . .	13
2.3.2. Beschreibung . . . . .	13
<b>II. Entwickeln und Anpassen</b>	<b>26</b>
<b>3. Git-Repository</b>	<b>27</b>
3.1. Ort . . . . .	27
3.2. Inhalt . . . . .	27
3.3. Firmware-Releases . . . . .	27
<b>4. Flashen</b>	<b>29</b>
4.1. Linux . . . . .	29
4.1.1. Vorbereitung . . . . .	29
4.1.2. Flashen . . . . .	29
4.2. Windows . . . . .	30
4.2.1. Vorbereitung . . . . .	30
4.2.2. Flashen . . . . .	30
<b>5. Programmierung</b>	<b>31</b>
5.1. Einrichtung Entwicklungsumgebung . . . . .	31
5.2. Übersicht über Sourcecode . . . . .	32
5.3. Belegung der Ressourcen . . . . .	33
<b>6. Schaltungsentwurf</b>	<b>34</b>
6.1. Beschreibung . . . . .	34

---

6.2. Auslegung . . . . .	34
6.3. Verbesserungen für weitere Versionen . . . . .	34
<b>7. Aufbau Fahrzeug</b>	<b>35</b>
7.1. Erstinbetriebnahme . . . . .	35



---

# **Teil I.**

# **Anwenden**

---

# 1 Fahrzeuge

## 1.1 Fahrgestelle

**Tabelle 1.1.:** Technische Daten

Radstand	258 mm
Spurweite	155 mm
Masse	kg

## 1.2 Aktorik

### 1.2.1 Lenkung

Die Lenkung ist als Achsschenkelenkung ausgeführt und wird über ein normales Modellbauservo angetrieben.

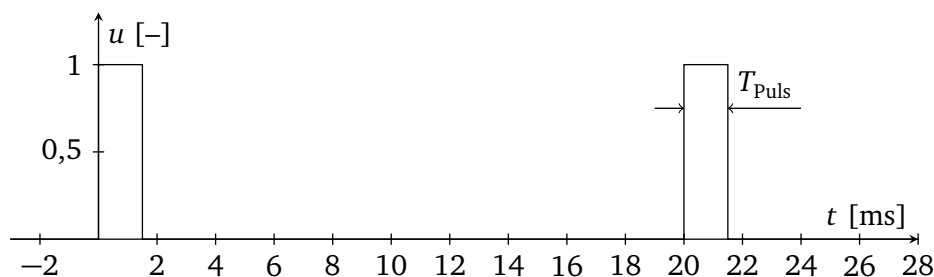
Das Servo stellt (über einen integrierten P-Regler) einen Sollwinkel, welcher sich über eine Mechanik auf einen Lenkwinkel der Räder überträgt.

Die Kennlinie „Servowinkel - Lenkwinkel“ ist nicht dokumentiert.

Die Ansteuerung des Servos, d. h. die Vorgabe eines Soll-Servowinkels, erfolgt über ein PWM-Signal. Dieses hat eine Frequenz von 50 Hz (wobei diese nicht sonderlich wesentlich ist) mit Impulsbreiten  $T_{\text{puls}}$  zwischen 1 ms und 2 ms (die wesentlich sind).

Dabei bedeutet eine Impulsbreite von 1,5 ms die Neutralstellung des Servos, welches (näherungsweise) einer Geradeausstellung der Räder entsprechen sollte. Ein Impulsbreite von 2 ms bedeutet eine maximale Auslenkung des Servos gegen den Uhrzeigersinn (bei Blick auf die Servowelle) was hier einer Lenkbewegung nach rechts entspricht. Eine Impulsbreite von 1 ms entspricht damit einer maximalen Lenkbewegung nach links.

In Abbildung 1.1 ist das Ansteuersignal für den Servo dargestellt.



**Abbildung 1.1.:** Ansteuerung Servo

Das ucboard übernimmt die Ansteuerung des Servos. Dabei wird von außen ein Sollwert zwischen  $-1000$  und  $1000$  vorgegeben. Dieser Bereich wird auf eine Impulsbreite von 1 bis 2 ms umgerechnet.

Die Auflösung der gestellten Impulsbreite beträgt dabei  $1\text{ }\mu\text{s}$ . (Damit ist die effektive Auflösung der Stellgröße 2.)

Es ist Folgendes zu beachten:

- Die Servos können aufgrund der Lenkmechanik nicht ihren vollen Stellbereich ausnutzen. Dies hört man, wenn ein Sollwert von 1 000 oder  $-1\,000$  vorgegeben wird. Es sollte auf Dauer vermieden werden, Servowinkel zu steuern, die nicht erreichbar sind.
- Lenken im Stillstand ist wie bei einem normalen Auto schwerer als in der Fahrt. Es kann je nach Achslast sein, dass das Servo einen gewünschten Lenkwinkel im Stillstand nicht erreicht.
- Die Lenkung besitzt ein gewisses Spiel. (Mit der IMU kann aber die Gierrate bestimmt werden, und damit kann bei bekannter Fahrgeschwindigkeit wiederum auf den Lenkwinkel geschlossen werden. Somit könnte der Lenkwinkel unterlagert geregelt werden.)

---

### 1.2.2 Fahrmotor (Fahrtenregler)

---

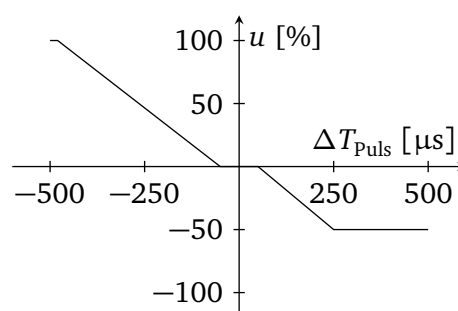
Die Fahrzeuge verfügen über Gleichstrommotoren, die über einen Tamiya Fahrtenregler angesteuert werden. Dabei ist bei den Chassis 1 bis 5 ein Tamiya TEU-101BK und bei den Chassis 6 und 7 ein Tamiya TEU-104BK verbaut. (Im Wesentlichen unterscheiden diese sich dadurch, dass bei den TEU-104BK ein Batterieschutz implementiert ist, der jedoch hier deaktiviert ist.)

Die Fahrtenregler werden wie die Lenkung durch ein „Servo-PWM-Signal“ angesteuert. D. h. eine Pulsbreite von  $1,5\text{ ms}$  entspricht „aus“, kürzere Pulsbreiten einer Vorwärtsfahrt und negative Impulsbreiten einer Rückwärtsfahrt bzw. Bremsen.

Die „Endanschläge“ (sowie die Neutralstellung) sind dabei kalibrierbar. Hier ist es so kalibriert, dass die Impulsbreiten  $1\text{ ms}$  und  $2\text{ ms}$  die Grenzwerte und  $1,5\text{ ms}$  die Neutralstellung darstellen.

Da der Fahrtenregler intern gewisse Toleranzzonen berücksichtigt sowie in der Rückwärtsfahrt nur die halbe Stellgröße verwendet, ergibt sich die in Abbildung 1.2 gezeigte Kennlinie, wobei  $\Delta t_{\text{Puls}}$  die Abweichung der Pulsbreite von  $1,5\text{ ms}$  ist,

$$T_{\text{Puls}} = 1,5\text{ ms} + \Delta T_{\text{Puls}}.$$



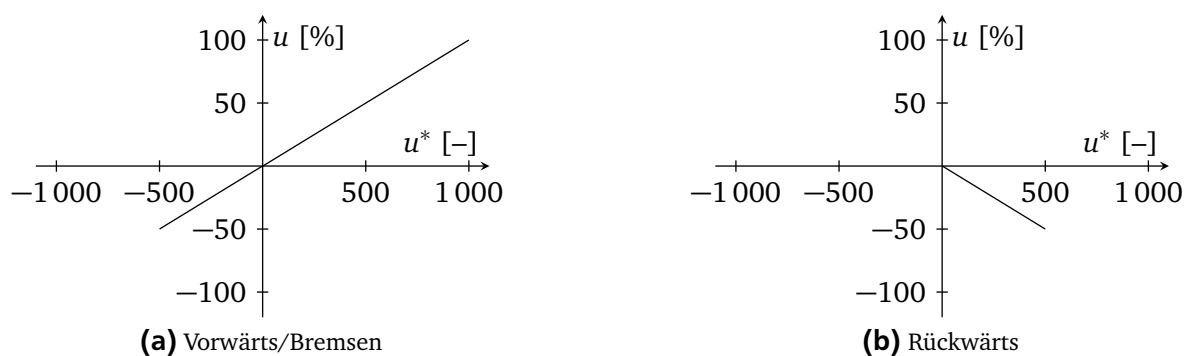
**Abbildung 1.2.:** Kennlinie Fahrtenregler

Aufgrund des eigentlichen Einsatzzweckes der Fahrtenregler weisen diese folgendes Verhalten auf:

- Negative  $\Delta T_{\text{Puls}}$ , d. h. kleinere Pulsbreiten als die Neutralstellung (außerhalb der Toleranzzone), führen immer zur Vorwärtsfahrt.
- Wenn von aus einer Vorwärtsbewegung zu positiven  $\Delta T_{\text{Puls}}$  gewechselt wird, dann erfolgt eine Bremsung. Diese wird aber nicht über den Stillstand hinaus ausgeführt, d. h. der Fahrtenregler stellt sicher, dass keine Rückwärtsbewegung eintritt.

- Um ausgehend von einer Vorwärtsfahrt Rückwärts zu fahren, muss der Fahrtenregler einmal mit positivem  $\Delta T_{\text{Puls}}$  angesteuert werden (Bremsen), dann muss der Fahrtenregler mit einer Pulsbreite von 1,5 ms angesteuert werden. Damit ist die Rückwärtsfahrt „freigeschaltet“. Wenn jetzt wieder ein positives  $\Delta T_{\text{Puls}}$  aufgebracht wird, dann erfolgt eine Rückwärtsfahrt. (Die Einzelschritte dieser Sequenz sind mindestens für 100 ms zu halten. Damit hat sich bisher immer eine sichere Umschaltung ergeben.)

Das ucboard bietet zur Ansteuerung zwei Möglichkeiten. Zum einen kann der Fahrtenregler „direkt“ betrieben werden, d. h. man dann den Wert für  $\Delta T_{\text{Puls}}$  in Mikrosekunden direkt vorgeben. Die zweite Möglichkeit ist die der „gemanagten“ Ansteuerung. Hierbei wird dem ucboard mitgeteilt, welche Fahrtrichtung gewählt werden soll und welche Stellgröße dabei verwendet werden soll. Dabei meint ein positiver Wert immer die gewählte Bewegungsrichtung, bei Vorwärtsfahrt bedeutet ein negativer Wert eine Bremsung. Dabei bildet bei Vorwärtsfahrt der Wertebereich von 1 bis 1 000 den Stellgrößenbereich von 0 bis 100 % ab, bei Rückwärtsfahrt wird der Wertebereich von 1 bis 500 auf 0 bis –50 % abgebildet. Ein Wert von 0 bedeutet Neutralstellung. Es ergeben sich damit die Kennlinien aus Abbildung 1.3.



**Abbildung 1.3.: Kennlinien zur „gemanagten“ Ansteuerung über ucboard**

Die für die Umschaltung von Vorwärts- auf Rückwärtsfahrt notwendige Bestimmung des Ist-Zustandes des Fahrtenreglers erfolgt dabei über einen Zustandsautomaten, der die Zustandswechsel des Fahrtenreglers nachbildet. Für den normalen Betrieb sollte dies sicher funktionieren. Fehler können nur dann auftreten, wenn

- sehr schnell zwischen Vor- und Rückwärtsfahrt gewechselt wird (weniger als 200 ms zwischen aufeinanderfolgenden Wechsel) oder
- im Direktmodus Werte gestellt werden, die an den Rändern des Toleranzbereichs für die Neutralstellung liegen.

Erfolgt eine Vorwärtsfahrt (länger als 100 ms) stimmen die Zustände wieder überein.

Die Fahrtenregler sind hier so angeschlossen, dass *keine* der 5V-Spannungsversorgungsleitungen angeschlossen ist, sondern die Spannung direkt aus dem Fahrakku nimmt. Die Verbindung zum Fahrakku kann über das ucboard und den drvbatswitch geschaltet werden.

Die notwendige Massenverbindung als Bezug für das PWM-Signal wird über die Verbindung des ucboards mit dem drvbatswitch hergestellt. Dieses muss daher immer angeschlossen sein. (Ansonsten muss (nur!) die Masseleitung des nicht angeschlossenen Spannungsversorgungskabel des Fahrtenreglers an einen Massepin des ucboards angeschlossen werden.)

## Problemlösung

- Schalter des Fahrtenreglers am Chassis auf „ON“?



---

## 1.3 Sensorik

---

### 1.3.1 Ultraschall

---

Es sind drei Ultraschallsensoren des Typs SRF08 verbaut, jeweils einer nach vorne, nach links und nach rechts.

Es wird die Zeit  $T_{\text{echo}}$  vom Aussenden des Ultraschallimpulses bis zum Empfang des ersten Echos gemessen.<sup>1</sup>

Der gemessene Abstand  $d$  entspricht der halben Signallaufstrecke  $c \cdot T_{\text{echo}}$ , wobei für die Schallgeschwindigkeit  $c = 343,2 \text{ m/s}$  angenommen wird. Es ergibt sich damit

$$d = \frac{1}{2} \cdot c \cdot T_{\text{echo}} = 171,6 \frac{\text{m}}{\text{s}} \cdot T_{\text{echo}}.$$

Der Sensor gibt die Zeiten  $T_{\text{echo}}$  in Mikrosekunden zurück, wobei die Auflösung  $4 \mu\text{s}$  ist. Damit ergibt sich eine Distanzauflösung von ca.  $0,7 \text{ mm}$ . Die Messwerte werden vom ucboard in  $0,1 \text{ mm}$  zurückgegeben.

#### Optionen: Messbereich und Verstärkung

Siehe Datenblatt des Sensors.

---

### 1.3.2 Hall-Sensor

---

Mit dem Hall-Sensor (Typ HAL 503) kann die Drehzahl des hinteren linken Rades gemessen werden. Dazu sind auf der Felge des Rades über den Umfang acht Magnete verteilt, die von der Polarität her immer wechselweise angeordnet sind.

Der genannte Sensor besitzt zwei stabile Zustände. Wird ein positiver magnetischer Pol in die Nähe gebracht, so wechselt er in den Zustand 1, bei einem negativen magnetischen Pol in den Zustand 0. Liegt kein Feld vor, dann wird der alte Zustand gehalten.

Somit kann durch Messen der Zeit zwischen zwei Zustandswechseln die Zeitdauer für eine achteil Umdrehung bestimmt werden.

Auf dem ucboard wird die Zeit zwischen den Impulsen auf eine Mikrosekunde genau bestimmt. Die Ausgabe der Messwerte erfolgt dann in  $0,1 \text{ ms}$ . Daneben wird auch die Summe dieser Zeiten über acht Zustandswechsel, also einer Radumdrehung, bestimmt und in einer Genauigkeit von  $1 \text{ ms}$  ausgegeben. Zuletzt erfolgt auch die Ausgabe der Anzahl an gezählten Zustandswechseln in Modulo 256, wenn extern selber mitgezählt werden soll. (Letzteres erlaubt auch die Überprüfung, ob ggf. Messdaten verloren gegangen sind.)

Im Gegensatz zu einem „klassischen“ Encoder kann die Drehrichtung nicht festgestellt werden!

- Der Abstand zwischen Rad und Aufbau darf nicht zu groß sein. Ist das Fahrzeug beispielsweise aufgebockt, so werden in der Regel keine Impulse mehr gezählt.<sup>2</sup>

---

<sup>1</sup> Genau genommen werden auch noch möglicherweise auftretende weitere Echos gemessen. Die dazugehörigen Zeiten stellt der Sensor in weiteren Registern zur Verfügung. Aktuell werden diese jedoch nicht ausgelesen. Möglicherweise könnte man mit einer Auswertung dieser Daten Fehlmessungen reduzieren.

<sup>2</sup> Vom Hersteller gibt es noch einen Hall-Sensor aus der gleichen Baureihe (HAL 502), der jedoch etwas empfindlicher (die notwendige Flussdichte beträgt ca. ein Drittel im Vergleich zum HAL 503) ist. Sollten häufig Probleme mit übersprungenen Impulsen auftreten, wäre dies eine mögliche Lösung.

---

### 1.3.3 IMU

---

Auf der hinteren rechten Ecke des ucboards befindet sich ein Beschleunigungs- und Drehratensensor (IMU – Inertial Measurement Unit) des Typs MPU-9250A.

Die  $x$ -Achse zeigt nach vorne, die  $y$ -Achse nach links und die  $z$ -Achse nach oben.

Der Sensor gibt die Messwerte als `int16_t`-Werte aus. Die Umrechnung eines Sensorwertes `val` in eine physikalische Einheit hängt von dem gewählten Messbereich ab. Für Beschleunigungswerte gilt

$$a(val) = \begin{cases} \frac{val}{16384} g & \text{wenn Messbereich } \pm 2 g \\ \frac{val}{8192} g & \text{wenn Messbereich } \pm 4 g \\ \frac{val}{4096} g & \text{wenn Messbereich } \pm 8 g \\ \frac{val}{2048} g & \text{wenn Messbereich } \pm 16 g \end{cases} \quad (1.1)$$

und für Drehratenwerte gilt

$$\omega(val) = \begin{cases} \frac{val}{131} ^\circ/s & \text{wenn Messbereich } \pm 250 ^\circ/s \\ \frac{val}{65,5} ^\circ/s & \text{wenn Messbereich } \pm 500 ^\circ/s \\ \frac{val}{32,8} ^\circ/s & \text{wenn Messbereich } \pm 1000 ^\circ/s \\ \frac{val}{16,4} ^\circ/s & \text{wenn Messbereich } \pm 2000 ^\circ/s. \end{cases} \quad (1.2)$$

Standardmäßig sind die Messbereiche  $\pm 4 g$  und  $\pm 500 ^\circ/s$  eingestellt. Diese können jedoch über entsprechende Befehle geändert werden.

#### Filterung

Die IMU verfügt über digitale Filter, die vor der sensorinternen Unterabtastung angewendet werden. (Der genaue Aufbau der Filter ist nicht dokumentiert. Die Angabe über Bandbreite und Verzögerung („delay“) lassen aber darauf schließen, dass es sich im Wesentlichen um eine Mittelwertbildung handelt.) Die möglichen Einstellungen, die über die ucboard-Befehle vorgenommen werden können, sind in Tabelle 1.2 aufgeführt. Diese Werte sind den Einträgen in der „Register Map“-Dokumentation des MPU-9250 entnommen.

**Hinweis:** Es ist zu beachten, dass die Abtastung (das Abfragen des Sensorwerte) seitens des ucboard unabhängig von der in Tabelle 1.2 bzw. im Sensordatenblatt genannten Abtastszeit immer mit 1 kHz erfolgt! Auch deshalb sollten die in Tabelle 1.2 grau geschriebenen Einstellungen nur zu Informationszwecken verwendet und nicht für den Normalbetrieb vorgesehen werden.

Die Standardeinstellung der Filter ist jeweils 0.

---

### 1.3.4 Magnetometer

---

Das Magnetometer vom Typ AK8963C ist im Gehäuse der IMU integriert. (Prinzipiell empfiehlt es sich, direkt das Datenblatt des AK8963 zu verwenden, und nicht in die entsprechenden Passagen des Datenblattes der IMU zu schauen.)

Die Achsenanordnung des Magnetometers unterscheidet sich von der der IMU bzw. damit auch der Achsenanordnung des Fahrzeugs. So zeigt die  $x$ -Achse zeigt nach links, die  $y$ -Achse nach vorne und die  $z$ -Achse nach unten. **Alle ausgegebenen Werte werden jedoch in Fahrzeug- oder IMU-Koordinaten angegeben!**

Es liegt ca. alle 10 ms neue Messwerte vor. (Ungefähr, da die Taktrate des Mikrocontrollers und die des Sensors leicht abweichen können.)

**Tabelle 1.2.: Filtereinstellungen IMU**

**(a)** Gyroskop

GFILT	Bandbreite [Hz]	Verzögerung [ms]	Interne Sensor- abtastzeit [kHz]
-2	8 800	0,064	32
-1	3 600	0,11	32
0	250	0,97	8
1	184	2,9	1
2	92	3,9	1
3	41	5,9	1
4	20	9,9	1
5	10	17,85	1
6	5	33,38	1
7	3 600	0,17	8

**(b)** Beschleunigungssensor

AFILT	Bandbreite [Hz]	Verzögerung [ms]	Interne Sensor- abtastzeit [kHz]
-1	1 046	0,503	4
0	218,1	1,88	1
1	218,1	1,88	1
2	99	2,88	1
3	44,8	4,88	1
4	21,2	8,87	1
5	10,2	16,83	1
6	5,05	32,48	1
7	420	1,38	1

Die Einstellungen 0 und 1 sind in der Dokumentation tatsächlich identisch angegeben. Das könnte man mal überprüfen.

---

## Korrektur der Sensorempfindlichkeit

Im Magnetometer ist intern für jede Achse ein Korrekturwert für die Sensitivität gespeichert (ASAX, ASAY und ASAZ). Dieser wird beim Starten ausgelesen und zur Korrektur der Messwerte verwendet. Diese Korrektur kann auch abgeschaltet werden (!MAG OPT ~USEASA=0).

Der korrigierte Wert  $H'_{\{x,y,z\}}$  ergibt sich aus dem Sensorwert  $H_{\{x,y,z\}}$  und dem jeweiligen Korrekturfaktor  $ASA\{X,Y,Z\}$  über

$$H'_{\{x,y,z\}} = H_{\{x,y,z\}} \cdot \left( \frac{(ASA\{X,Y,Z\} - 128) \cdot 0,5}{128} + 1 \right).$$

## Kalibrierung des eingebauten Sensors

Eine weitere Kalibrierung des Magnetometers ist für die Zukunft vorgesehen. Aktuell muss dies aber, wenn gewünscht bzw. notwendig, selber auf dem PC realisiert werden.

## Berechnung des Kurswinkels

Unter der Annahme, dass das Fahrzeug eben steht, kann aus dem Messwert der magnetischen Feldstärke in  $x$ - und  $y$ -Richtung der im(!) Uhrzeigersinn gezählte Kurswinkel  $\psi_h$  über

$$\psi_h = \text{atan2}(MY, MX)$$

in Radiant bestimmt werden. Die Funktion  $\text{atan2}$  gibt einen Wert zwischen  $-\pi$  und  $\pi$  zurück. Einen typischen „Kompasswert“ in Grad erhält man daher nach dem Pseudocode

```
heading = atan2(MY, MX) * 180 / pi  
  
if heading < 0  
    heading = 360 + heading  
end
```

---

## 2 Kommunikation

---

### 2.1 Prinzip

---

Die Kommunikation mit dem ucboard ist textbasiert, so dass eine Bedienung über ein einfaches Terminalprogramm möglich ist. Dies ermöglicht ein einfaches Testen und Debuggen. Um dennoch ein einfaches Parsen der Nachrichten zu ermöglichen, besitzen diese ein definiertes Format.

In der Regel sind alle Nachrichten auch einfach lesbar. Um bezüglich der Messdatenerfassung etwas platzeffizienter zu sein, können diese jedoch optional als hex- oder base64-codierte Binärdaten versendet werden.

Ebenso wird in der Regel auf Prüfsummen verzichtet. Lediglich bei Messdaten, die als Binärdaten verschickt werden, kann optional eine CRC16-Prüfsumme angehängt werden. (Bei manchen Befehlen zum ucboard besteht die Möglichkeit, die wesentliche Zahl doppelt zu senden.)

Messdaten und Textnachrichten können vom ucboard ohne Aufforderung versendet werden. Ansonsten reagiert das ucboard auf Befehle, die an dieses geschickt werden. Dabei wird jeder Befehl durch eine Antwort quittiert.

- Die Nachrichten zum ucboard sollen sich möglichst einfach Parsen lassen. Die Nachrichten bestehen aus einem oder mehreren, durch Leerzeichen getrennte Wörter. Der Typ eines Wortes ergibt sich aus dem ersten Zeichen:
  - A - Z und \_ : String
  - ~ : Optionsname
  - 0 - 9 und - : Zahl
- Besteht eine Nachricht aus mehreren Wörtern, so spielt die Anzahl der Leerzeichen zwischen den Wörtern keine Rolle. (Mindestens eines!)

Prinzipiell wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Ein Nachricht beginnt mit einem Startzeichen und endet mit einem Endezeichen. Das Startzeichen kann variieren. Bei Nachrichten zum ucboard ist dieses ! oder ?, bei Nachrichten vom ucboard :, ' und #. Nach dem Startzeichen kann, muss aber kein Leerzeichen folgen. Alle Startzeichen dürfen auch innerhalb der Nachrichten verwendet werden. Dort haben sie keine besondere Bedeutung. (Eine „Aufsynchronisierung“ sollte also anhand des Endezeichens erfolgen.)

Das Endezeichen ist \n (newline) bei Nachrichten zum ucboard und \03 (ETX) bei Nachrichten vom ucboard. Die Motivation dahinter ist, dass damit bei Nachrichten zum ucboard ein einfaches Terminalprogramm verwendet werden kann, bei Nachrichten vom ucboard jedoch auch mehrzeiliger Text sinnvoll dargestellt werden kann.

Einzelne Zeilenumbrüche (leere Nachrichten) sind zu ignorieren. (Diese werden optional nach ETX vom ucboard verwendet, um die Darstellung im Terminprogramm zu verbessern.)

---

#### Zu ucboard

---

- Befehle beginnen mit !

- 
- Abfragen beginnen mit ?
  - Antworten auf Fragen des ucboard (im interaktiven Modus) besitzen keinen speziellen Beginn. (Sie dürfen aber auch mit ! oder ? beginnen.
  - Das Ende einer Nachricht wird ausschließlich durch \n (newline) markiert
  - Optionen beginnen mit ~. Wenn der Option ein Wert zugeordnet wird, dann wird dieser nach einem = angeschlossen. Dabei dürfen um das Gleichheitszeichen herum keine Leerzeichen stehen!
  - Die Reihenfolge der Argumente spielt eine Rolle. Die Position der Optionen spielt keine Rolle. (Bei Verarbeiten im ucboard wird zunächst eine Liste der Argumente und eine Liste der Optionen erstellt. Die Information, ob eine Option am Anfang, zwischen zwei Argumenten oder am Ende stand geht dabei verloren.)

---

## Von ucboard

---

- Direkte Antworten beginnen mit :
- Auch jeder Schreibbefehl sollte eine kurze Antwort zur Quittierung senden, z.B. :ok\n. Es wäre empfehlenswert, den gesetzten Wert zu wiederholen
- Fehler bei der Abarbeitung von Befehlen beginnen mit :ERR(code), wobei code eine positive Ganzzahl als Fehlercode ist. Optional kann nach einem weiteren Doppelpunkt eine Beschreibung des Fehlers folgen: :ERR(code):Beschreibung
- Textausgaben (Display-Funktion) beginnen mit '
- Fehlermeldungen sind Textausgaben. Diese beginnen mit 'ERR(code), wobei code eine positive Ganzzahl als Fehlercode ist. Optional kann nach einem weiteren Doppelpunkt eine Beschreibung des Fehlers folgen: 'ERR(code):Beschreibung
- Ohne Aufforderung versendete Messdaten beginnen mit # (base64- oder hex-codiert) bzw. ## (lesbarer Text).
- Wenn eine Nutzerinteraktion notwendig ist, dann sollte das letzte Zeichen vor dem Nachrichten-  
dezeichen ein ? oder : sein.
- Alle Nachrichten vom ucboard haben ETX (0x03) als Endzeichen. (Dadurch ist es möglich, eine  
mehrzeilige Ausgabe auf dem Terminalprogramm zu erzeugen.)
  - Über eine – aktuell noch fest eingestellte Option – wird nach jedem ETX automatisch ein  
Zeilenumbruch geschickt. Dies dient der besseren Lesbarkeit. Es wäre jedoch besser, ein Ter-  
minalprogramm zu verwenden (bzw. zu schreiben), welches ETX durch einen Zeilenumbruch  
ersetzt. Damit wäre sowohl der Text besser lesbar als auch der Sparsamkeit Rechnung getra-  
gen.

---

## 2.2 Hardware

---

Das ucboard verfügt über zwei Schnittstellen für die Kommunikation mit dem PC. Zum einen kann über einen USB-Anschluss eine serielle Kommunikation aufgebaut werden und zum anderen kann eine RS232-Schnittstelle verwendet werden. Die RS232-Schnittstelle wird dabei nicht als Standard-D-SUB-Stecker (9-polig) angeboten, sondern als Wannensteckeranschluss. Dieser ist so belegt, dass eine direkte Verbindung zu den Anschlüssen des Onboard-PC über ein Flachbandkabel erfolgen kann.

---

**Tabelle 2.1.: Einstellungen serielle Schnittstellen**

	USB	RS232
Baudrate	921 600	115 200
Datenbits	8	8
Stopbits	1	1
Parity	keine	keine
Hardware-Flowcontrol	Nein	Nein

**ToDo: Testen, welche Baudrate jeweils maximal möglich ist!** Man könnte dies auch über einen Befehl einstellbar machen, und ein Rücksetzen darüber erreichen, dass beim Starten des ucboards z. B. der Taster A betätigt wird. Aber das erscheint mir für diesen Zweck unnötig.

Der dritte, „reine“ UART-Anschluss ist für den Anschluss möglicher Erweiterungen und nicht die Kommunikation mit dem PC vorgesehen.<sup>1</sup>

Die Parameter der Schnittstellen sind in Tabelle 2.1 zusammengefasst.

---

## 2.3 Befehle

---

---

### 2.3.1 Übersicht

---

**Tabelle 2.2.: Übersicht über Hauptbefehle ucboard**

RESET	Neustart ucboard
VER	Abfrage Softwareversion
ID	Abfragen der Fahrzeug-ID
SID	Setzen und Abfragen der Session-ID
TICS	Abfrage ucboard-Zeit (Millisekunden nach (Neu)start)
STEER	Setzen und Abfragen des (Soll-)Lenkwinkels
DRV	Setzen und Abfragen der Fahrgeschwindigkeit
DAQ	Datenerfassung
VOUT	Ein- und Ausschalten 12 V-Ausgang (Kinect)
IMU	Zum Parametrieren der IMU
MAG	Zum Parametrieren des Magnetometers
US	Zum Parametrieren der US-Sensoren
EEPROM	Zum Auslesen des EEPROM-Inhalts

---

**Wichtig:** Momentan werden numerische Eingaben noch nicht an allen Stellen daraufhin überprüft, ob es tatsächlich eine (Ganz)Zahl ist. Dies bedeutet, dass z. B. Eingaben wie 1000., 1000.0 oder auch +1000 für Tausend eine unerwartete Zahl ergibt.

---

### 2.3.2 Beschreibung

---

---

<sup>1</sup> Man könnte auch noch darüber nachdenken, die dritte UART auch für die Kommunikation mit einem PC zu verwenden, da an dieser einfach ein Bluetooth-Adapter angeschlossen werden könnte. Damit wäre es möglich, auch ohne den Onboard-PC das Fahrzeug ferngesteuert zu betreiben.

---

## RESET

---

Führt einen Neustart des ucboards durch.

```
!RESET NOW
```

---

## VER

---

Fragt Versionsstring der ucboard-Software ab.

```
?VER  
:0.1.x
```

---

## ID

---

Zur Abfrage der Fahrzeug-ID, also der Nummer, die über die Dipschalter auf der Platine eingestellt wird.

```
?ID  
:4
```

---

## SID

---

Zum Abfragen und Setzen der Session-ID. Dies ist einfach eine Zahl, die nach einem Neustart des ucboards auf 0 gesetzt wird, und der beliebige uint32-Werte gegeben werden können.

```
?SID  
:0
```

```
!SID 25363  
:25363
```

```
?SID  
:25363
```

---

## STEER

---

Zum Setzen und Abfragen des Soll-Lenkswinkels (Servo-Vorgabe). Der Wert muss eine Ganzzahl zwischen -1 000 und 1 000 sein.

Setzen:

```
!STEER -200  
:-200
```

Optionale Wiederholung des Arguments, um Fehlübertragungen zu detektieren:



---

```
!STEER -200 -200
:-200
```

```
!STEER -200 -201
:ERR(269): Message corrupted! (The two values have to be equal!)
```

Abfragen:

```
?STEER
:-200
```

---

DRV

---

Zum Setzen und Abfragen der Fahrgeschwindigkeit. (Bzw. Motorspannung.)

Es gibt zwei Modi: Die „gemanagte“ und die „direkte“ Ansteuerung. Siehe dazu Abschnitt 1.2.2.

### **Gemanagte Ansteuerung**

Bei der gemanagten Ansteuerung erfolgt die Umschaltung von Vorwärts- und Rückwärtsfahrt automatisch. Zudem wird der gesetzte Wert in den Arbeitsbereich umgerechnet.

Vorwärtsfahrt:

```
!DRV F 300
:F 300
```

Optionale Wiederholung des Arguments, um Fehlübertragungen zu detektieren:

```
!DRV F 300 300
:F 300
```

```
!drv f 300 301
:ERR(269): Message corrupted! (The two values have to be equal!)
```

Es sind Werte von -500 bis 1000 zulässig. Dabei entsprechen negative Werte Bremsen (aber nicht einer Rückwärtsfahrt!).

Die effektive Auflösung ist geringer als der Stellbereich von 1000 es annehmen lässt. Tatsächlich können etwas weniger als 450 unterscheidbare Impulsbreiten, d. h. Sollwerte, vorgegeben werden.

Bremsen

```
!DRV F -500
:F -500
```

Gebremst wird vom Fahrtenregler automatisch nur bis zum Stillstand. Es erfolgt keine Rückwärtsfahrt.

Rückwärtsfahrt

```
!DRV B 300
:B 300
```

---

Es sind Werte von 0 bis 500 zulässig. (Der Stellbereich des Fahrtenreglers ist rückwärts nur halb so groß wie vorwärts.)

Ausschalten des Fahrtenreglers:

```
!DRV OFF
:OFF
```

Der Unterschied zwischen den Werten 0 und OFF liegt darin, dass bei OFF der Fahrtenregler vom Fahrakku getrennt wird, also tatsächlich ausgeschaltet ist. Der Fahrtenregler wird bei Übermittlung des nächsten Sollwertes ungleich OFF automatisch wieder eingeschaltet, jedoch dauert dies einen Moment. (Das Einschalten wird auch mit einem Piepston des Fahrtenreglers begleitet.)

Abfragen:

```
?DRV
:F 300
```

Abfragen der tatsächlichen Impulsbreite (vergleiche „Direkte Ansteuerung“):

```
?DRV D
:D -186
```

### Direkte Ansteuerung

Bei der direkten Ansteuerung wird direkt die Impulsbreite vorgegeben, die an den Fahrtenregler übertragen wird. Dabei wird die Impulsbreite als die Abweichung in Mikrosekunden vom Neutralwert 1,5 ms angegeben. D. h. ein Wert von -500 entspricht der minimalen Impulsbreite (d. h. Maximalwert *vorwärts*) von 1 ms und ein Wert von 500 der maximalen Impulsbreite von 2 ms.

```
!DRV D 200
:D 200
```

### DMS

Die Totmannschaltung (DMS – Dead-man switch) soll insbesondere im Entwicklungsstatus der PC-Software und der Algorithmen verhindern, dass das Fahrzeug bei abgestürztem PC oder abgestürzten ROS-Nodes weiterfährt.

Dieser Parameter gibt die Zeit in Millisekunden an, innerhalb derer eine neue DRV-Nachricht erhalten sein muss. Wird diese Zeit ohne DRV-Nachricht überschritten, so wird der Motor gestoppt, d. h. der Sollwert auf 0 (aber nicht OFF) gesetzt. Bei der nächsten DRV-Nachricht wird der Motor dann wieder angesteuert.

```
!DRV ~DMS=1000
:~DMS=1000
```

Zum Abschalten der Sicherheitsschaltung dient der Wert 0.

DRV kann auch ohne Parameter aufgerufen werden. Dann dient er nur dazu, die Zeitzählung der Totmannschaltung neu zu starten.

```
!DRV
:ok
```

---

Abfrage der DMS-Abschaltzeit zusammen mit dem aktuellen DRV-Wert:

?DRV ~DMS  
:OFF ~DMS=1000

---

## DAQ

---

Das Datenerfassungsmodul (DAQ – Data Acquisition) dient als allgemeine Schnittstelle zur Übermittlung aller Sensorsignale. Dabei können Einzelabfragen vorgenommen werden (die angefragten Sensorwerte werden direkt als Antwort auf einen entsprechenden Befehl verschickt) als auch Gruppen von Sensorsignalen eingerichtet werden, die dann automatisch vom ucboard an den PC geschickt werden.

Die Sensorsignale werden dabei hier als „Kanäle“ bezeichnet. Die aktuell definierten Kanäle sind in Tabelle 2.3 zusammengefasst.

**Tabelle 2.3.: DAQ-Kanäle (Signale)**

Kanal	Beschreibung	Einheit	Datentyp	Länge
AX	Beschleunigung x-Richtung	→ (1.1)	int16_t	2
AY	Beschleunigung y-Richtung	→ (1.1)	int16_t	2
AZ	Beschleunigung z-Richtung	→ (1.1)	int16_t	2
GX	Drehrate um x-Achse	→ (1.2)	int16_t	2
GY	Drehrate um y-Achse	→ (1.2)	int16_t	2
GZ	Drehrate um z-Achse	→ (1.2)	int16_t	2
MX	Magnetische Flussdichte in x-Achse	0,15 µT	int16_t	2
MY	Magnetische Flussdichte in y-Achse	0,15 µT	int16_t	2
MZ	Magnetische Flussdichte in z-Achse	0,15 µT	int16_t	2
USF	Abstand vorne	0,1 mm	uint16_t	2
USL	Abstand links	0,1 mm	uint16_t	2
USR	Abstand rechts	0,1 mm	uint16_t	2
VSBAT	Spannung Systemakku	mV	uint16_t	2
VDBAT	Spannung Fahrakku	mV	uint16_t	2
HALL_DT	Zeit zwischen den letzten beiden Impulsen des Hallsensors (eine Achtel Radumdrehung)	0,1 ms	uint16_t	2
HALL_DT8	Zeit zwischen den letzten acht Impulsen des Hallsensors (eine Radumdrehung)	1 ms	uint16_t	2
HALL_CNT	Anzahl der Sensorimpulse, Modulo 256 (Ein Sensorwert von „1“ entspricht dabei immer eine ungeraden Zahl)	–	uint8_t	1

Zur Übermittlung von Fehlerwerten sind bestimmte Zahlenwerte oder Textausgaben vorgesehen, die in Tabelle 2.4 aufgeführt sind. Dabei ist „keine Daten vorhanden“ ein Wert, der vom DAQ-Modul selber gesetzt werden, wenn keine gültigen Daten vorliegen. Die einzelnen Sensormodule geben im Fehlerfall entweder „Messfehler“ (bei einzelnen Messfehlern oder Messaussetzern) oder „Sensorfehler“ (ein größeres Sensorproblem, d. h. es ist in diesem Fall mit dem Ausfall aller (weiteren) Messwerten zu rechnen) zurück.

### Allgemeines

Alle aktuell definierten Kanäle können mit ?DAQ CHS abgefragt werden:

**Tabelle 2.4.: Fehlerwerte**

**(a)** Binärcodierung

Datentyp	uint32_t	int32_t	uint16_t	int16_t	uint8_t	int8_t
Keine Daten vorhanden	0xFFFFFFFF	0x7FFFFFFF	0xFFFF	0x7FFF	0xFF	0x7F
Messfehler	0xFFFFFFFFE	0x7FFFFFFE	0xFFFFE	0x7FFE	0xFE	0x7E
Sensorfehler	0xFFFFFFFFD	0x7FFFFFFD	0xFFFFD	0x7FFD	0xFD	0x7D
Wert zu groß oder zu klein	0xFFFFFFFFC	0x7FFFFFFC	0xFFFFC	0x7FFC	0xFC	0x7C
Wert zu groß	0xFFFFFFFFC	0x7FFFFFFC	0xFFFFC	0x7FFC	0xFC	0x7C
Wert zu klein	0xFFFFFFFFB	0x7FFFFFFB	0xFFFFB	0x7FFB	0xFB	0x7B

**(b)** Textausgabe

Keine Daten vorhanden	[---]
Messfehler	[mfault]
Sensorfehler	[fault]
Wert zu groß oder zu klein	[range]
Wert zu groß	[over]
Wert zu klein	[under]

?DAQ CHS

:14

AX	acc. ahead	opt-dep.!	1	I16
AY	acc. left	opt-dep.!	1	I16
[...]				
HALL_DT8	delta time full rev.	1 ms	undef	U16

Dabei beinhalten die Spalten die folgenden Informationen

- Name
- Beschreibung
- Einheit (opt-dep. ! bedeutet, dass die Einheit von den jeweiligen Sensoreinstellungen abhängt.)
- Abtastzeit in Millisekunden (undef bedeutet, dass das entsprechende Sensormodul keine feste Abtastzeit der Messwerte garantiert.)
- Datentyp

Die Informationen zu einem speziellen Kanal können über

?DAQ CH name

abgefragt werden:

?DAQ CH USF

:USF	ultrasonic front distance	0.1 mm	undef	U16
------	---------------------------	--------	-------	-----

### Einzelabfrage von Werten

Zur Abfrage von Einzelwerten dient der Subbefehl GET:

---

```
?DAQ GET cname1 [cname2 [cname3 [...]]] [~AGE] [~TICS]
```

Diesem muss mindestens der Name eines Kanals übergeben werden. Es können aber auch mehrere Kanäle auf einmal abgefragt werden. Die Option `~AGE` gibt zu *jedem* Wert das Alter in Tics (d. h. ms) zwischen dem Zeitpunkt der Datenerfassung und der Datenabfrage zurück. Die Option `~TICS` gibt zu *jedem* Wert den Erfassungszeitpunkt in Tics zurück. Sind beide Optionen gewählt, dann wird immer AGE vor TICS zurückgegeben, unabhängig von der Reihenfolge der Optionen.

Beispiele:

- Einzelabfrage des Sensorwertes des vorderen Ultraschallsensors

```
?DAQ GET USF
:1860
```

- Ausgabe des Alters des Messwertes

```
?DAQ GET ~AGE USF
:1860 68
```

- Abfrage aller Ultraschallwerte, jeweils mit Alter

```
?DAQ GET ~AGE USL USF USR
:487 72 | 1860 75 | 4293 85
```

- Ausgabe der ucboard-Zeit des Messwerts

```
?DAQ GET ~TICS USF
:1860 3059246
```

## Automatische Messgruppen

Es stehen zwanzig parametrierbare Messgruppen zur Verfügung, die zum automatischen Verschicken von Messdaten verwendet werden. In diesen können bis zu zehn verschiedene Kanäle (Sensorwerte) zusammengefasst werden. Neben den normalen Kanälen stehen noch die in Tabelle 2.5 aufgeführten „Sonderkanäle“ zur Verfügung, die Metadaten zu den Datensätzen enthalten. (Diese stehen bei der Einzelabfrage über `?DAQ GET cname` nicht zur Verfügung.)

Es wird je Gruppe immer nur ein Messdatensatz gepuffert. Kann dieser nicht verschickt werden, bis die nächste Abtastung der Gruppe erfolgt ist, so wird dieser verworfen. Dabei werden alle Gruppen gleichberechtigt behandelt, d. h. es erfolgt keine Bevorzugung von Messgruppen mit kleiner Abtastzeit. Damit ist die Wahrscheinlichkeit gering, dass bei langsam abgetasteten Messgruppen Daten verlorengehen, auch wenn viel Bandbreite für Messgruppen mit kleiner Abtastzeit belegt wird.<sup>2</sup>

Eine Messgruppe wird über den Subbefehl GRP definiert.

```
!DAQ GRP grpnb cname1 [cname2 [cname3 [...]]] [~option1] [~option2] [...]
```

Die Gruppennummer `grpnb` muss eine Ganzzahl zwischen 0 und 19 sein.

Die Möglichkeiten der Gruppendifinition wird zunächst an Beispielen demonstriert. Danach werden die Optionen genauer beschrieben.

---

<sup>2</sup> An dieser Stelle muss noch angemerkt werden, dass das Management der zu versendenden Daten im Idle-Task des ucboards erfolgt. D. h. während der Datenerfassung im SysTick-Task werden keine neuen Daten zum Verschicken bereit gemacht, auch wenn der letzte Datensatz abgearbeitet ist. Damit ist der erreichbare Datendurchsatz kleiner als der sich theoretisch aus der Baudrate ergebenden Wert. Dies kann bzw. sollte in Zukunft noch geändert werden. Bisher benötigt der SysTick-Task weniger als 200 µs. Damit würde im ungünstigsten Fall die Datenrate ein Fünftel „verlieren“.

- Gruppe 1 enthält die Werte des Ultraschalls. Es wird gesendet, wenn alle Ultraschallwerte vorliegen, wobei maximal 10 ms nach dem Erfassen des ersten Wertes gewartet wird. Die Daten werden base64-codiert und mit einer CRC16-Prüfsumme verschickt.

!DAQ GRP 1 ~ALL=10 ~ENC=B64 ~CRC \_TIC USL USF USR

(Die Reihenfolge der Kanalnamen (\_TIC, USL, USF, USR) spielt eine Rolle, da die Messwerte ohne Namen in dieser Reihenfolge ausgegeben werden. Die Reihenfolge der Optionen und ob diese am Anfang oder Ende oder auch zwischen den Kanalnamen stehen spielt keine Rolle.)

- Gruppe 2 enthält die Spannungen der beiden Akkus, wobei immer eine Nachricht verschickt wird, sobald eine neue Spannung gemessen ist.

!DAQ GRP 2 ~ANY VSBAT VDBAT

- Gruppe 3 enthält die Beschleunigungswerte in der Ebene und die Gierrate. Die Daten werden alle 10 ms verschickt. Dabei werden alle innerhalb der 10 ms erfassten Daten gemittelt.

!DAQ GRP 3 ~TS=10 ~AVG ~ENC=B64 ~CRC AX AY GZ

Optionen:

- Modus Abtastung
  - ~ALL[=maxwait]: Sendet, wenn zu allen Gruppenkanäle neue Daten vorhanden sind. Wenn maxwait gegeben ist, dann wird nach dem ersten neuen Wert maximal diese Zeit in Millisekunden gewartet, bis die Daten verschickt werden.<sup>3</sup>
  - ~ANY: Sendet, wenn zu einem der Gruppenkanäle ein neues Datum vorhanden ist.
  - ~TS=Ts: Abtastzeit in Millisekunden. (Muss ein ganzes Vielfaches der Abtastzeiten aller Kanäle des Paketes sein.)
    - \* ~AVG[=Ta]: Mittelung über Zeitraum Ta. (Nur im Abtastmodus TS verfügbar.) Wenn die Option AVG verwendet wird, dann müssen alle Gruppenkanäle die gleiche Abtastzeit besitzen. Ta darf dabei maximal Ts sein und muss ein ganzes Vielfaches der Abtastzeit der Gruppenkanäle sein. Wenn kein Ta angegeben ist, dann wird Ta = Ts gesetzt. Diese Mittelung kann damit bezüglich der Sensorabtastzeit  $T_{s,sig}$  über die Übertragungsfunktion

$$G_{avg}(z) = \frac{\frac{1}{n_{avg}} \cdot (z^{n_{avg}-1} + z^{n_{avg}-2} + \dots + 1)}{z^{n_{avg}-1}}$$

mit  $n_{avg} = Ta/T_{s,sig}$  ausgedrückt werden. Über ~TS=Ts wird dann eine Unterabtastung mit dem Faktor  $Ts/T_{s,sig}$  vorgenommen.

- ~SKIP=n: Überspringt n Werte. D. h. bei einem Wert von neun wird jeder zehnte Wert verwendet. Dieses Überspringen wird nach allen anderen Berechnungen und Abfragen durchgeführt. Diese Option dient primär zum Debuggen (um die Menge an ausgegebenem Text zu reduzieren), kann aber auch zur Unterabtastung für Kanäle ohne feste Abtastzeit verwendet werden. (Dabei ist dann aber keine Mittelung möglich.)
- ~ENC=B64|HEX|ASCII: Nachricht wird base64-codiert (ohne Padding), Hex-codiert oder Ascii-codiert. ASCII meint hierbei „lesbaren Text“. Standardwert ist ASCII.
- ~CRC: CRC16-Prüfsumme (Nur wenn ENC = B64 oder HEX)  
Berechnung mit:

<sup>3</sup> Dies ist z. B. für die US-Sensoren gedacht, deren Daten meist an nacheinander liegenden Zeitschritten ankommen.

```

1 // from Antonio Pires, http://stackoverflow.com/questions/10564491/function-to-→
  ←calculate-a-crc16-checksum

  // CRC-16-CCITT (polynom 0x1021)

6 uint16_t crc16(uint8_t* data_p, uint16_t length)
{
    uint8_t x;
    uint16_t crc = 0xFFFF;

11    while (length--)
    {
        x = (crc >> 8) ^ (*data_p++);
        x ^= (x >> 4);
        crc = (crc << 8) ^ ((uint16_t)(x << 12)) ^ ((uint16_t)(x << 5)) ^ ((uint16_t)→
          ←x);
16    }

    return crc;
}

```

- ~AGE und ~TICS: Wie bei Einzelwertabfrage. Damit wird *jedem* Wert das Alter sowie der Erfassungszeitpunkt hinzugefügt. (Wenn beide Optionen gewählt sind, dann immer in der Reihenfolge „Age – Tics“.) Die Werte werden bei der Ascii-Codierung immer innerhalb der senkrechten Striche des Kanals geschrieben. Bei Binärcodierungen werden diese Werte jeweils als 32 bit-Werte jeweils hinter den betreffenden Messwert eingefügt. Diese Optionen dienen lediglich zum Debuggen. Im „Normalbetrieb“ sollten diese aufgrund des großen Platzbedarfs nicht verwendet werden!<sup>4</sup>

**Tabelle 2.5.:** „Sonderkanäle“ mit Metadaten

Signal	Beschreibung	Datentyp	Länge
_CNT	Anzahl der verschickten Datensätze der Gruppe	uint32_t	4
_CNT16	die niederwertigen 16 bits von _CNT	uint16_t	2
_CNT8	die niederwertigen 8 bits von _CNT	uint8_t	1
_TICS	ucboard-Zeit (tics) der Erfassung des ersten Datums	ms uint32_t	4
_TICS16	die niederwertigen 16 bits von _TIC	ms uint16_t	2
_TICS8	die niederwertigen 8 bits von _TIC	ms uint8_t	1
_DTICS	Delta der ucboard-Zeit zwischen ersten und letztem Datum	ms uint32_t	4
_DTICS16	min{_DTICS, 65 535}	ms uint16_t	2
_DTICS8	min{_DTICS, 255}	ms uint8_t	1
_DLY	Zeit zwischen Datensatzerstellung und Versenden, bei 255 gesättigt	ms uint8_t	1

Weitere Funktionen zu Messgruppen:

- Anzeige aller Gruppeninformationen

<sup>4</sup> Das Alter AGE bezieht sich auf den Abtastzeitpunkt des Datensatzes, nicht auf den Zeitpunkt des Versendens. Der Wert ist damit nur bei der Abtastart ALL relevant, da dort die Daten einer Gruppe zu unterschiedlichen Zeitpunkten abgetastet sein können. Die Zeitdifferenz zwischen Abtastzeitpunkt und Versenden kann durch den Sonderkanal \_DLY abgefragt werden. (Somit ist das tatsächliche Alter AGE + \_DLY plus die Zeit zur Datenübermittlung und Verzögerungen innerhalb des Betriebssystem des PCs.)

---

?DAQ GRPS

: [...]

- Anzeigen Informationen zur Gruppe 1

?DAQ GRP 1

: [...]

- Löschen von Gruppe 1

!DAQ GRP 1 ~DELETE

:ok

- Deaktivieren von Gruppe 1

!DAQ GRP 1 ~DEACTIVATE

:ok

- Aktivieren von Gruppe 1

!DAQ GRP 1 ~ACTIVATE

:ok

(Standardmäßig ist eine Gruppe nach der (Neu-)Definition aktiviert. Dies kann unterbunden werden, indem die Option ~DEACTIVATE in die Definition mit aufgenommen wird.)

#### Starten und Anhalten der Erfassung und Übermittlung der Messdaten

- Starten der Datenerfassung

!DAQ START

:started

- Anhalten der Datenerfassung

!DAQ STOP

:stopped

- Abfrage Status der Datenerfassung

?DAQ

:stopped

Die Neudefinition einer Gruppe erfolgt durch einfaches Überschreiben der Gruppendefinition. Dabei ist zu beachten, dass *aktive* Gruppen nur dann überschrieben werden können, wenn die Datenerfassung angehalten ist (!DAQ STOP).

#### Binärdaten

Bei ENC = B64 und HEX werden die Daten als Binärdaten verschickt.

Dabei ist das erste Byte des dekodierten Binärdatensatzes die Messgruppe. Darauf folgen ohne Leer- oder Trennzeichen die einzelnen Messdaten mit der jeweils für den Kanal gültigen Breite.

!daq grp 1 usl usf usr ~all=20

:ok

!daq grp 2 usl usf usr ~all=20 ~enc=hex

:ok

!daq grp 3 usl usf usr ~all=20 ~enc=hex ~crc

:ok

!daq grp 4 usl usf usr ~all=20 ~enc=b64 ~crc

:ok



---

```
!daq start
:started
```

```
##1:7306 | 1887 | 3655
#028A1C5F07470E
#038A1C5F07470ECFA5
#BIocXwdHDou8
```

### Sonstiges

Die Definition einer Messgruppe, die nur aus Sonderkanälen besteht, ist möglich. Dies ist jedoch – wenn überhaupt – nur mit der Abtastart TS sinnvoll, da bei ANY und ALL nie ein Datensatz verschickt werden würde.

So kann mit

```
!daq grp 1 _cnt8 _dly ~Ts=10 ~enc=b64
```

ein „Taktgeber“ erzeugt werden, der alle 10 ms einen Wert schickt. Über \_CNT8 kann dabei festgestellt werden, ob Werte übersprungen wurden, und mit \_DLY, ob der aktuelle Wert verzögert wurde.

---

## VOUT

---

Einschalten der Kinect-Spannung

```
!VOUT ON
:ON
```

Abschalten der Kinect-Spannung

```
!VOUT OFF
:OFF
```

Abfragen Zustand

```
?VOUT
:OFF
```

---

## US

---

### Aktivieren und Deaktivieren der US-Sensoren

(Nach einem Neustart des ucboards sind die US-Sensoren aktiv.)

Aktivieren der US-Sensoren

```
!US ON
:ON
```

Deaktivieren der US-Sensoren

---

!US OFF  
:OFF

Abfragen Zustand der US-Sensoren

?US  
:OFF

### Einstellungen US-Sensoren

Es können die beiden Parameter „Range“ und „Gain“ eingestellt werden. (Es wird immer für alle drei US-Sensoren der selbe Wert genommen.) Für die Bedeutung der Parameter und deren Zahlenwerte siehe das Datenblatt des Sensors!

Setzen der Einstellungen:

!US OPT ~RANGE=100 ~GAIN=10  
:~RANGE=100 ~GAIN=10 [4343 mm, 25 ms]

(Es kann auch nur ~RANGE oder ~GAIN gesetzt werden.)

In eckigen Klammern wird die Entsprechung des Range-Parameters in Millimetern und Millisekunden ausgegeben. Die tatsächliche Abtastzeit wird ein paar Millisekunden über diesem Wert liegen, da ein gewisser Overhead vorhanden ist.

Abfragen der aktuellen Einstellungen:

?US OPT  
:~RANGE=100 ~GAIN=10 [4343 mm, 25 ms]

---

### IMU

---

Setzen der Einstellungen:

!IMU OPT ~ARANGE=8 ~AFILT=3 ~GRANGE=1000 ~GFILT=0  
:~ARANGE=8 ~AFILT=3 ~GRANGE=1000 ~GFILT=0 [A: 45 Hz, G: 250 Hz]

(Es brauchen nicht alle Optionen gesetzt zu werden.)

ARANGE gibt den Messbereich des Beschleunigungssensors in g an und kann auf 2, 4, 8 oder 16 gesetzt werden. GRANGE gibt den Messbereich des Gyrometers in Grad pro Sekunde an und kann auf 250, 500, 1000 oder 2000 gesetzt werden. Die möglichen Werte für die Filter sind auf Seite 9 in Tabelle 1.3b (AFILT) bzw. Tabelle 1.3a (GFILT) aufgeführt.

Abfragen der aktuellen Einstellungen:

?IMU OPT  
:~ARANGE=8 ~AFILT=3 ~GRANGE=1000 ~GFILT=0 [A: 45 Hz, G: 250 Hz]

---

### MAG

---

Setzen der Einstellungen:

---

```
!MAG OPT ~USEASA=1
:~USEASA=1
```

USEASA gibt an, ob die im Magnetometer fest gespeicherten Korrekturwerte für die Sensitivität verwendet werden sollen. Dabei bedeutet der Wert 1, dass die Werte verwendet werden und der Wert 0, dass die Werte nicht verwendet werden. Der Wert nach einem Neustart ist 1.

Abfragen der aktuellen Einstellungen:

```
?MAG OPT
:~USEASA=1
```

Abfrage der Korrekturwerte:

```
?MAG ASA
:176 176 165
```

Diese Werte entsprechen der Reihenfolge nach den Korrekturwerten der x-, y- und z-Achse, gezählt in Fahrzeugkoordinaten. (Damit entsprechen diese Werte den Registern ASAY, ASAX und ASAZ des Magnetometers.)

Da es sich bei den Korrekturwerten um fest gespeicherte Werte im Magnetometer handelt, können diese nicht verändert werden. Eine Vorgabe von eigenen Kalibrierungswerten ist für die Zukunft vorgesehen. Momentan muss eine Kalibrierung des eingebauten Sensors, wenn gewünscht, aber noch auf dem PC erfolgen.



---

# **Teil II.**

## **Entwickeln und Anpassen**

---

---

## 3 Git-Repository

---

### 3.1 Ort

---

<https://github.com/tud-pses/ucboard>

---

### 3.2 Inhalt

---

<code>datasheets\</code>	Datenblätter der Sensoren sowie Datenblätter der Bauteile der Schaltungen
<code>doc\</code>	Latex-Dateien um dieses Dokument zu erstellen
<code>fw_releases\</code>	(firmware_releases) bin-Dateien der wesentlichen Firmwareversionen
<code>fw_workspace\</code>	(firmware_workspace) Workspace der Entwicklungsumgebung, Source-code des auf dem Mikrocontroller laufenden Programms
<code>kicad_drvbatswitch\</code>	KiCad-Dateien (Schaltplan und Layout) der kleinen Schaltplatine für den den Fahrakku
<code>kicad_ucboard\</code>	KiCad-Dateien (Schaltplan und Layout) der ucboard-Platine
<code>kicadlibs\</code>	KiCad-Bibliotheken mit Bauteilen für drvbatswitch und ucboard
<code>matlab\</code>	Matlab-Klasse zur direkten Kommunikation mit ucboard, Skripte zur Betrachtung von Sensorwerten
<code>mfg\</code>	Fertigungsdaten (Gerber-Format) für Platinen
<code>stm32cubemx\</code>	Projekt für STM32CubeMX (Programm zur Konfiguration des Mikrocontrollers)
<code>ucterm\</code>	Einfaches in C# geschriebenes Terminalprogramm für Entwicklung am ucboard

---

### 3.3 Firmware-Releases

---

Im Verzeichnis `fw_releases\` befinden sich die Firmware-Versionen (Programme für den Mikrocontroller) zu bestimmten Versionständen. Die bin-Dateien sind dabei nach dem Schema

`pses_ucboard_verVERSIONSNUMMER_BUILDCONF.bin`

benannt, so z. B.

`pses_ucboard_ver0.9.0_Debug.bin`

Zu jeder Version, die in diesem Verzeichnis liegt, sollte im Git-Repository ein Tag der Form

`fwver_VERSIONSNUMMER`

vorhanden sein. Zu dem Beispiel oben gehört also der Tag

`fwver_0.9.0`

Während der Weiterentwicklung bis zu einem Stand, der eine neue Versionsnummer erhält, sollte der Versionsnummer in `version.h` ein „+“ angehängt werden, also z.B. 0.9.0+. (Idealerweise wird das + direkt nach dem Erzeugen der Release-Version angehängt, um zu vermeiden dies zu vergessen.)

Für Weiterentwicklungen innerhalb der Gruppen sollte dem Versionsstring noch der Gruppenname oder eine ähnliche Kennzeichnung hinzugefügt werden.

Tabelle 3.1 gibt eine Übersicht über die Firmware-Versionen.

**Tabelle 3.1.: Übersicht über Firmware-Versionen**

Version	Datum	Kommentar
0.11.1	09.11.2017	– Beseitigung von Bugs bei Peripherieansteuerung
0.11.0	26.11.2016	– Magnetometer in Betrieb genommen
0.10.1	07.11.2016	– Standardeinstellungen geändert: US ist nach Neustart aus, Kinect ist nach Neustart ein
0.10.0	03.11.2016	– Befehl IMU hinzugefügt (Parametrierung IMU) – Interne Änderung im Kommunikationsstack
0.9.1	01.11.2016	– Befehl US hinzugefügt (Ein- und Ausschalten und Parametrierung US-Sensoren) – Beseitigung von Bugs im Kommunikationsstack
0.9.0	25.10.2016	Grundfunktionalität weitgehend vorhanden. (Es fehlen noch die Befehle zum Parametrieren und Kalibrieren der Sensoren und der Treiber für das Magnetometer.)

## Roadmap

**Tabelle 3.2.: Roadmap Firmware**

Version	Kommentar
0.12.0	Verwenden der in IMU abgelegten Offsetwerte für Drehratensensor
0.12.0	Befehle zur Ansteuerung der LEDs und zur Abfrage der Tasten
x	– [intern] Aufteilen Kommunikationsstack in Befehlsbearbeitung und UART-Treiber
x	– [intern] Umstellen der UART-Kommunikation auf DMA
1.0.0	– Befehlsbasierte (= textbasierte) benutzerdefinierte Initialisierung über EEPROM
1.1.0	– Kalibrierungsroutinen
x	– [intern] Code-Review, Gleiche/Ähnliche Funktionen zusammenfassen, ...

---

## 4 Flashen

---

### 4.1 Linux

---

---

#### 4.1.1 Vorbereitung

---

STM stellt für den ST-Link/V2-Programmer keine Software für Linux zur Verfügung. Dieser Adapter kann aber dennoch zum Flashen über Linux benutzt werden, wenn das auf GitHub verfügbare Programm `stlink` verwendet wird. Dieses kann wie folgt installiert werden.<sup>1</sup>

1. Für Installation benötigte Pakete installieren

```
sudo apt-get install libusb-1.0-0-dev git unzip libgtk-3-dev
```
2. `stlink` herunterladen (hier Version 1.2.0, ggf. neuere Version nehmen)

```
wget https://github.com/texane/stlink/archive/1.2.0.zip
```
3. Datei entpacken und in das Verzeichnis wechseln

```
# Unzip the file
unzip 1.2.0.zip
cd stlink-1.2.0
```
4. Programm kompilieren

```
mkdir build && cd build
cmake -DCMAKE_BUILD_TYPE=Debug ..
make
```
5. `stlink` installieren

```
sudo make install
```
6. udev-Regeln so einstellen, dass `st-flash` ohne „sudo“ verwendet werden kann

```
cd ..
sudo cp *.rules /etc/udev/rules.d
sudo restart udev
```
7. Programm sollte sich jetzt öffnen lassen

```
stlink-gui
```

---

#### 4.1.2 Flashen

---

1. ST-Link/V2 an Platine anschließen. Fahrzeug einschalten.
2. Öffnen der grafischen Oberfläche von `stlink`

```
stlink-gui
```

---

<sup>1</sup> Vorgehen nach Nicolas Acero.

- 
3. Auf „Connect“-Icon klicken. Es sollten die Daten des Mikrocontrollers angezeigt werden.
  4. Auf „Open File“-Icon klicken und bin-Datei auswählen.
    - Im Repository unter fw\_releases\ liegen die „offiziellen“ Versionen.
  5. Auf „Program“-Icon klicken.
    - Start-Adresse ist 0x08000000. (Sollte voreingestellt sein.)
- 

## 4.2 Windows

---

### 4.2.1 Vorbereitung

---

„STM32 ST-Link Utility“ von der STM-Homepage herunterladen und installieren (beinhaltet auch die Treiber).

**Hinweis:** Die könnte schon problematisch sein, wenn man den ST-Link/V2 zum Debuggen verwendet will, da damit auch ein spezieller Treiber installiert wird, für gbd aber ein generischer Treiber benötigt wird. (Quelle: Forum, quergelesen. Müsste noch geklärt werden.)

---

### 4.2.2 Flashen

---

1. ST-Link/V2 an Platine anschließen. Fahrzeug einschalten.
2. „STM32 ST-Link/V2 Utility“ öffnen.
3. Auf „Connect“-Icon (drittes von links) klicken. Es sollten die Daten des Mikrocontrollers angezeigt werden.
4. Auf „Open File“-Icon (erstes von links) klicken und bin-Datei auswählen.
  - Im Repository unter fw\_releases\ liegen die „offiziellen“ Versionen.
5. Auf „Program Verify“-Icon (sechstes von links) klicken.
  - Start-Adresse ist 0x08000000. (Sollte voreingestellt sein.)



---

# 5 Programmierung

---

## 5.1 Einrichtung Entwicklungsumgebung

---

### System Workbench for STM32

Die verwendete Entwicklungsumgebung „System Workbench for STM32“ basiert auf Eclipse. Die Entwicklungsumgebung kann über [www.openstm32.org](http://www.openstm32.org) geladen werden. Die Entwicklungsumgebung beinhaltet einen Compiler.

### Projekt importieren

Wenn das git-Repository lokal geclont wurde, muss das Projekt einmalig in die Entwicklungsumgebung importiert werden. Dazu wird wie folgt vorgegangen:

1. Starten von „System Workbench for STM32“
2. „File“ → „Switch Workspace“ → „Other...“: Verzeichnis `fw_workspace\` des Repositories auswählen
3. „File“ → „Import“...:
  - „General“ → „Existing Projects into Workspace“
  - „Select root directory“: Verzeichnis `fw_workspace\` des Repositories über „Browse...“ auswählen
  - „Projects“: `pses_ucboard` auswählen
  - „Options“: „Copy projects into workspace“ darf NICHT ausgewählt sein. (Die Daten sind ja schon da, wo sie hingehören.)
  - „Finish“
4. STRG+B sollte Projekt erstellen
5. Falls im „Problems“-Reiter ein Fehler angezeigt wird, dass ein Symbol nicht aufgelöst werden kann:
  - Rechte Maustaste auf Projekt (ersten Eintrag) im „Project Explorer“, → „Properties“
  - „C/C++ General“ → „Indexer“  
(Falls es den Punkt „C/C++ General“ nicht gibt, dann hat man vorher nicht auf das Projekt geklickt.)
  - „Enable project specific settings“ auswählen
  - „Index source files not included in the build“ abwählen

(Die Ursache liegt darin, dass die Headerdateien für viele Mikrokontrollervarianten vorhanden sind, die alle die gleichen Symbole definieren. Letztlich wird nur ein Header eingebunden, aber mit der genannten Option schaut sich Eclipse dennoch alles an und weiß dann nicht, welches Symbol das richtige ist.)

---

## Einrichten des Programmieradapters ST-Link/V2

**Hinweis:** Es müsste möglich sein, den ST-Link/V2 auch zum Debuggen einzurichten. Dies wurde bisher jedoch noch nicht gemacht, sondern er wird lediglich zum Flashen verwendet.

**Hinweis:** Hier wird die Einrichtung unter Windows beschrieben.

1. „STM32 ST-Link Utility“ von der STM-Homepage herunterladen und installieren (beinhaltet auch die Treiber).  
**Hinweis:** Dieser Schritt könnte schon problematisch sein, wenn man den ST-Link/V2 zum Debuggen verwendet will, da damit auch ein spezieller Treiber installiert wird, für gbd aber ein generischer Treiber benötigt wird. (Quelle: Forum, quergelesen. Müsste noch geklärt werden.)
2. Im Verzeichnis `fw_workspace\pses_ucboard` die Datei `stlinkflash_template.bat` in `stlinkflash.bat` kopieren. (Letztere sollte von Git ignoriert werden.)
3. In der Datei `stlinkflash.bat` die Pfade zum ST-Link-Utility anpassen.
4. In der Entwicklungsumgebung:
  - a) „Run“ → „External Tools“ → „External Tools Configurations...“
  - b) Links auf „Program“ klicken, dann oben auf das linke Icon zum Erstellen einer neuen Konfiguration
    - „Name“: „STM flashen“
    - „Location“: Über „Browse Workspace...“ die Datei `stlinkflash.bat` auswählen
    - „Arguments“:  
`"${workspace_loc}/${project_name}/${config_name:${project_name}}/${project_name}.bin"`  
Die doppelten Anführungszeichen müssen miteingegeben werden!
  - c) „Apply“
  - d) „Close“
5. Beim ersten Klicken auf das „External-Tools“-Icon (Play-Symbol mit Werkzeugkoffer) kann dann die Konfiguration „STM flashen“ ausgewählt werden. Danach ist diese automatisch voreingestellt. (Die Batch-Datei verwendet das Kommandozeilenprogramm des „ST-Link-Utilities, um den Chip zu flashen, zu überprüfen und danach zu reseten.)
6. Sollte sich bei Klick auf das Icon eine Message-Box mit dem Fehler „Variable references empty selection: \${project\_name}“ erscheinen, dann muss einfach „in“ eine Datei im Editor-Fenster geklickt werden, so dass der Eingabefokus auf dieser Datei liegt. Wenn dann wieder auf das Icon geklickt wird, sollte es funktionieren.

## Weitere Einstellungen

Standardmäßig speichert die IDE die geänderten Dateien vor einem neuen Build nicht. Um dies zu verändern kann wie folgt vorgegangen werden:

1. „Window“ → „Preferences“
2. In der linken Spalte „General“ → „Workspace“ auswählen.
3. Die Option „Save automatically before build“ auswählen.

---

## 5.2 Übersicht über Sourcecode

---

---

### 5.3 Belegung der Ressourcen

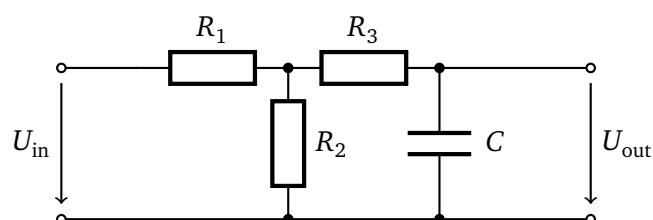
---

# 6 Schaltungsentwurf

## 6.1 Beschreibung

## 6.2 Auslegung

### Spannungsteiler mit RC-Glied



**Abbildung 6.1.:** Spannungsteiler mit RC-Glied

Die Schaltung aus Abbildung 6.1 (ideale Spannungsquelle am Eingang, offene Klemmen am Ausgang) stellt ein  $PT_1$ -Glied mit der stationären Verstärkung

$$\frac{R_2}{R_1 + R_2}$$

und der Zeitkonstante

$$T = C \cdot \left( \frac{R_1 R_2}{R_1 + R_2} + R_3 \right)$$

dar.

## 6.3 Verbesserungen für weitere Versionen

- Hall-Sensor muss nicht über OpAmp geleitet werden. Dafür wäre ein 5 V-toleranter Eingang empfehlenswert.

---

# 7 Aufbau Fahrzeug

---

## 7.1 Erstinbetriebnahme

---

- Bevor der Fahrzeugaufbau auf das Chassis gesetzt wird, muss der Fahrtenregler kalibriert werden.