Beschreibung Fahrzeug und ucboard

PS Echtzeitsysteme
Rev. 0+ (Entwurf)

TECHNISCHE
UNIVERSITÄT
DARMSTADT

PS Echtzeitsysteme
FG Echtzeitsysteme
FG Regelungstechnik und Mechatronik

Inhaltsverzeichnis

Teil I.
Anwenden

1 Fahrzeuge

1.1 Fahrgestelle

1.2 Aktorik

1.2.1 Lenkung

Die Lenkung ist als Achsschenkellenkung 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_{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.

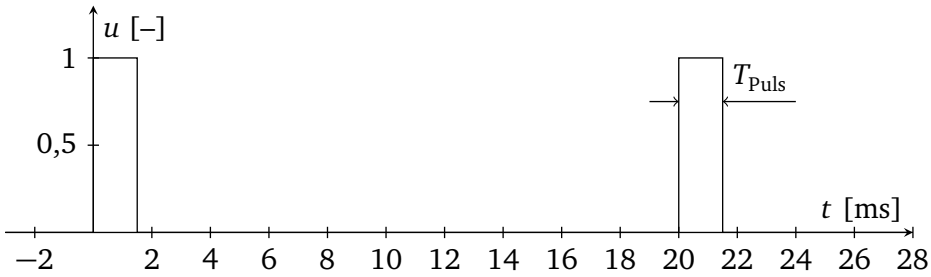


Abbildung 1.1.: Ansteuerung Servo

Das ucboard übernimmt die Ansteuerung des Servos. Dabei wird von außen ein Sollwert zwischen −1 000 und 1 000 vorgeben. Dieser Bereich wird auf eine Impulsbreite von 1 bis 2 ms umgerechnet. Die Auflösung der gestellten Impulsbreite beträgt dabei 1 µs. (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 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 kalibiriert, dass die Impulsbreiten 1 ms und 2 ms die Grenzwerte und 1,5 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 Δt_{Puls} die Abweichung der Pulsbreite von 1,5 ms ist,

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

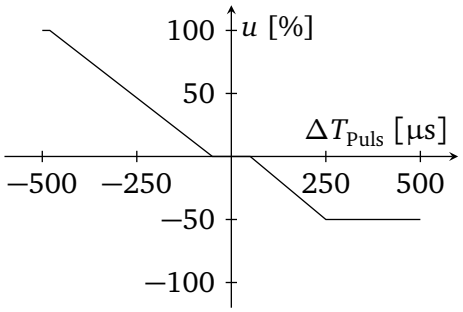
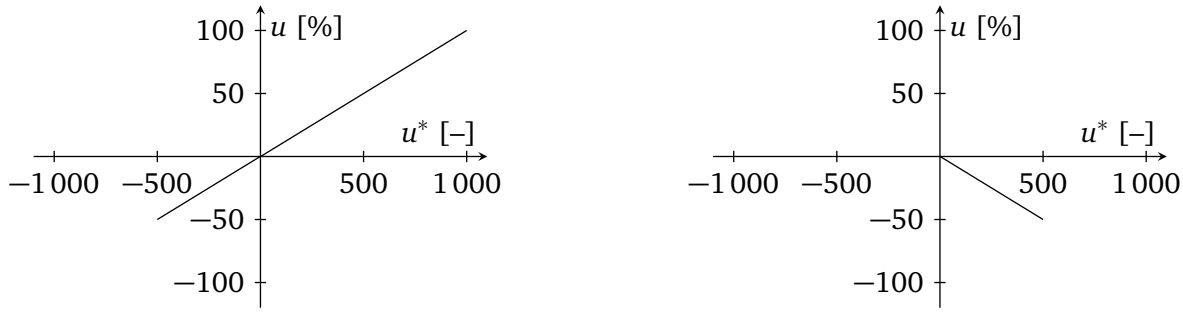


Abbildung 1.2.: Kennlinie Fahrtenregler

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

- Negative ΔT_{Puls}, d. h. kleinere Pulsbreiten als die Neutralstellung (außerhalb der Toleranzzone), führen immer zur Vorwärtsfahrt.
- Wenn von aus einer Vorwärtsbewegung zu positiven ΔT_{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 ΔT_{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 ΔT_{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 ΔT_{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.



(a) Vorwärts/Bremsen **(b)** Rückwärts

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).
- 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 5 V-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_{echo} vom Aussenden des Ultraschallimpulses bis zum Empfang des ersten Echos gemessen.[1]

Der gemessene Abstand d entspricht der halben Signallaufstrecke $c \cdot T_{\text{echo}}$, wobei für die Schallgeschwindigkeit $c = 343{,}2\,\text{m/s}$ angenommmen 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_{echo} in Mikrosekunden zurück, wobei die Auflösung 4 µs ist. Damit ergibt sich eine Distanzauflösung von ca. 0,7 mm. Die Messwerte werden vom ucboard in 0,1 mm zurückgegeben.

Optionen: Messbereich und Verstärkung

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 achtel 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 ms. Daneben wird auch die Summe dieser Zeiten über acht Zustandswechsel, also einer Radumdrehung, bestimmt und in einer Genauigkeit von 1 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 Regler keine Impulse mehr gezählt.[2]

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. Dieser beinhaltet auch ein Magnetometer, welches jedoch noch nicht angesprochen wird.

[1] 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.

[2] 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.

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

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

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

und für Drehratenwerte gilt

$$\omega(x) = \begin{cases} \frac{x}{131}\,^\circ/\mathrm{s} & \text{wenn Messbereich } \pm 250\,^\circ/\mathrm{s} \\ \frac{x}{65{,}5}\,^\circ/\mathrm{s} & \text{wenn Messbereich } \pm 500\,^\circ/\mathrm{s} \\ \frac{x}{32.8}\,^\circ/\mathrm{s} & \text{wenn Messbereich } \pm 1000\,^\circ/\mathrm{s} \\ \frac{x}{16.4}\,^\circ/\mathrm{s} & \text{wenn Messbereich } \pm 2000\,^\circ/\mathrm{s}\,. \end{cases}$$

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

Filterung

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 quitiert.

- 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 eine 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 Nachrichtendezeichen 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 Terminalprogramm zu verwenden (bzw. zu schreiben), welches ETX durch einen Zeilenumbruch ersetzt. Damit wäre sowohl der Text besser lesbar als auch der Sparsamkeit Rechnung getragen.

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.[1]

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
`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

RESET

Führt einen Neustart des ucboards durch.

[1] 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 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 `int32`-Werte gegeben werden können.

```
?SID
:0

!SID 25363
:25363

?SID
:25363
```

STEER

Zum Setzen und Abfragen des Soll-Lenkwinkels (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
```

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

„Totmannschaltung“ (Dead-man switch)

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
:ok
```

Zum Abschalten der Sicherheitsschaltung dient der Wert `OFF`.

DRV kann auch ohne Parameter aufgerufen werden. Dann dient er nur dazu, die Zeitzählung der Totmannschaltung (bzw. Tot-PC-Schaltung) neu zu starten.

```
!DRV
:ok
```

DAQ

Das Datenerfassungsmodul (DAQ – Data Aquisition) 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 Sensorsignale 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.

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

Tabelle 2.3.: Signale

Signal	Beschreibung	Einheit	Datentyp	Länge
AX	Beschleunigung x-Richtung		int16_t	2
AY	Beschleunigung y-Richtung		int16_t	2
AZ	Beschleunigung z-Richtung		int16_t	2
GX	Drehrate um x-Achse		int16_t	2
GY	Drehrate um y-Achse		int16_t	2
GZ	Drehrate um z-Achse		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

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:

```
?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
```

Tabelle 2.4.: Fehlerwerte

Datentyp	uint32_t	int32_t	uint16_t	int16_t	uint8_t	int8_t
Keine Daten vorhanden	0xFFFFFFFF	0x7FFFFFFF	0xFFFF	0x7FFF	0xFF	0x7F
Messfehler	0xFFFFFFFE	0x7FFFFFFE	0xFFFE	0x7FFE	0xFE	0x7E
Sensorfehler	0xFFFFFFFD	0x7FFFFFFD	0xFFFD	0x7FFD	0xFD	0x7D
Wert zu groß oder zu klein	0xFFFFFFFC	0x7FFFFFFC	0xFFFC	0x7FFC	0xFC	0x7C
Wert zu groß	0xFFFFFFFC	0x7FFFFFFC	0xFFFC	0x7FFC	0xFC	0x7C
Wert zu klein	0xFFFFFFFB	0x7FFFFFFB	0xFFFB	0x7FFB	0xFB	0x7B

(a) Binärcodierung

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

(b) Textausgabe

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 chname1 [chname2 [chname3 [...]]] [~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


?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 chname` 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.[2]

Eine Messgruppe wird über den Subbefehl `GRP` definiert.

```
!DAQ GRP grpnb chname1 [chname2 [chname3 [...]]] [~option1] [~option2] [...]
```

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

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

- 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

[2] 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“.

- ~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.[3]
- ~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 $T_{\mathrm{s,sig}}$ über die Übertragungsfunktion

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

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

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

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


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

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

    return crc;
}
```

[3] Dies ist z. B. für die US-Sensoren gedacht, deren Daten meist an nacheinander liegenden Zeitschritten ankommen.

- ~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![4]

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

  ```
  ?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.)

[4] 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.)

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
```

Teil II.
Entwickeln und Anpassen

3 Git-Repository

3.1 Ort

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

3.2 Inhalt

`datasheets\`	Datenblätter der Sensoren sowie Datenblätter der Bauteile der Schaltungen
`doc\`	Latex-Dateien um dieses Dokument zu erstellen
`fw_releases\`	(firmware_releases) bin-Dateien der wesentlichen Firmwareversionen
`fw_workspace\`	(firmware_workspace) Workspace der Entwicklungsumgebung, Sourcecode des auf dem Mikrocontroller laufenden Programms
`kicad_drvbatswitch\`	KiCad-Dateien (Schaltplan und Layout) der kleinen Schaltplatine für den den Fahrakku
`kicad_ucboard\`	KiCad-Dateien (Schaltplan und Layout) der ucboard-Platine
`kicadlibs\`	KiCad-Bibliotheken mit Bauteilen für drvbatswitch und ucboard
`mfg\`	Fertigungsdaten (Gerber-Format) für Platinen
`stm32cubemx\`	Projekt für STM32CubeMX (Programm zur Konfiguration des Mikrocontollers)

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_ver`*`VERSIONSNUMMER`*`_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.)

Tabelle 3.1.: Übersicht über Firmware-Versionen

Version	Datum	Kommentar
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.)

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.

4 Flashen

Hinweis: Bisher wird hier nur die Variante des Flashens unter Windows mit dem „STM32 ST-Link Utility“ beschrieben.

4.1 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 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 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 geladen werden. Die Entwicklungsumgebung beinhaltet einen Compiler.

Projekt importieren

Wenn das git-Repostory 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 Mikrocontrollervarianten 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.

5.2 Übersicht über Sourcecode

5.3 Belegung der Ressourcen

6 Schaltungsentwurf

6.1 Beschreibung

6.2 Auslegung

Spannungsteiler mit RC-Glied

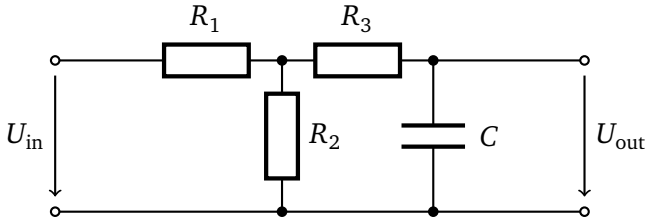


Abbildung 6.1.: Spannungsteiler mit RC-Glied

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.