Dokumentation zum Übertragungsprotokoll RoboKIDE → Roboter

Einleitung

RoboKIDE ist eine GUI/IDE zur grafischen/visuellen Programmierung von einfachen Robotern. RoboKIDE ist in Java implementiert. Der Roboter soll das Programm empfangen und dann interpretieren/ausführen. Das Protokoll ist absichtlich nicht allein für das Fangenspiel von c't-Bots konzipiert, sondern allgemein gehalten.

Das Protokoll

Anfragen (Enquiries)

Mit Anfragen fordert man vom Roboter Informationen über ihn an. So kann z.B. die Größe seines Speichers abgefragt werden, um dann in RoboKIDE eine Fehlermeldung zu erzeugen, wenn das Programm, das übertragen werden soll, zu groß ist...

Der komplette Datenaustausch wird mit ASCII-Zeichen realisiert, d.h. z.B., dass Integer-Variablen nicht binär übertragen werden, sondern zuerst in einen String umgewandelt werden.

RoboKIDE schickt folgendes zum Roboter:

`ENQ c`

`ENQ` ist ein ASCII-Steuerzeichen, steht für Enquiry und hat den Zahlenwert 5 dezimal (0x05 hexadezimal). `c` ist ein druckbares ASCII-Zeichen, dass die Art der Anfrage spezifiziert. Empfohlene Werte für `c`: `'a' .. 'z', 'A' .. 'Z', '0' .. '9'`.

Der Roboter antwortet einfach direkt mit der gewünschten Information. Die Anzahl der ASCII-Zeichen in der Antwort muss je nach dem festgelegt werden. Falls er die Information nicht bereitstellen kann, braucht er *kein* NAK (Negative Acknowledge) oder sonst etwas zurückschicken, sondern antwortet einfach nicht!

RoboKIDE wartet eine gewisse kurze Zeit (timeout) auf eine Antwort – falls keine kommt, weiß RoboKIDE, dass die gewünschte Information nicht vorhanden ist.

Folgende Anfragen sind bis jetzt festgelegt:

- Anfrage für **maximal erlaubte Anzahl an Steuerelementen pro *Programm*** (was Programm ist: siehe weiter unten, im Kapitel Übertragung)
  Enquiry: ENQ a (a legt fest, dass es sich eben um genau diese Anfrage handelt)
  Answer: 99 (ein oder zwei ASCII-Zeichen; müssen Ziffern sein, d.h. '0'..'9')

Übertragung

Die Übertragung ist einfach das, was RoboKIDE über RS232 zum Roboter schickt.

Die Übertragung ist so aufgebaut:

SOH	*Programmdatenblock*	[weitere *Programmdatenblöcke*]	EOT

Jakob Schöttl <jakob.schoettl@hm.edu>

Das in eckigen Klammern ([ ]) ist optional. Es können also auch mehr als ein *Programmdatenblock* zwischen SOH und EOT eingebaut werden.

Ein *Programmdatenblock* umfasst alles nötige, um ein Programm (z.B. das für die Fängerrolle) zu übertragen.
Ein *Programmdatenblock* ist so aufgebaut:

Programmnummer	STX	*Programmdaten*	ETX

- *Programmnummer*: Ein ASCII-Zeichen (Byte) das die eindeutige Nummer des Programms angibt; reserviert: '0' für „Fänger“ und '1' für „Wegläufer“. Das Zeichen muss eine Ziffer von 0 bis 9 sein (ASCII-Code: 48 bis 57 dezimal).
- *Programmdaten*: Ein Text (CSV-Format) der ein Programm für den c't-Bot darstellt, siehe weiter unten.

ASCII-Steuerzeichen:

Dez.	Hex.	Abk.	Beschreibung
1	0x01	SOH	Start of Heading
2	0x02	STX	Start of Text
3	0x03	ETX	End of Text
4	0x04	EOT	End of Transmission

Tabelle aller ASCII-Steuerzeichen, siehe
http://de.wikipedia.org/wiki/Steuerzeichen#Tabelle_der_Steuerzeichen_in_ASCII

Die *Programmdaten* sind die reinen Daten, die ein Programm für den Roboter darstellen und werden in folgender Form übertragen:

- CSV-Format, mit Semikolon (;) getrennt.
- Vor und nach dem Semikolen sind keine Leerzeichen.
- Rein Text (ASCII), d.h. z.B. Integer werden als String übertragen, nicht binär.
- Zeilen werden mit folgendem Zeilentrennzeichen getrennt: Line Feed '\n' (Unix-/Linux-Standard), ASCII-Code: LF, dezimal 10, hexadezimal A.

Die Datenstruktur ist dabei folgende (siehe auch Beispiele):

- In jeder Zeile wird genau ein Steuerelement übertragen.
- In der *ersten* Zeile wird das Start-Steuerelement übertragen, das den Einstiegspunkt ins Programm darstellt.

- In allen weiteren Zeilen werden die übrigen Steuerelemente (in beliebiger Reihenfolge) übermittelt.

Steuerelemente sind die einzelnen Anweisungen, die in der GUI durch farbige Kästchen mit Pfeilen zum Nächsten symbolisiert werden – ihr wisst schon.

Die Daten einer Zeile (eines Steuerelements) – in folgender Reihenfolge:

- **name: String**
  Name des Steuerelements, beschreibt meist den Typ, z.B. „Gegner erkennen“. Für Verarbeitung auf c't-Bot-Seite eigentlich nicht nötig.
- **controlElementUID: Integer**
  Unique Identifier des Steuerelements zur Identifikation in verketteter Liste.
- **typeUID: Integer**
  Typ-Id des Steuerelements (1 bis 3 reserviert für Go, Turn und Recognize).
- **pathCount: Integer**
  Anzahl der Pfade zu möglichen nächsten Steuerelementen (z.B. 2 Pfade bei Recognize d.h. If-Struktur).
- **nextControlElementUID: Integer**
  Id des nächsten Steuerelements. Wenn kein nächstes Steuerelement definiert ist, ist der Text „null“ (String natürlich, kein Nullpointer!).

} so viele UIDs nacheinander wie es Pfade gibt!

- **Paremter.key: String**
  Ein String als Key (Id) der den Parameter identifiziert.
- **Parameter.value**
  Datentyp von value ist nicht festgelegt; hängt aber natürlich mit dem Key zusammen. Wird in jedem Fall als String übermittelt.

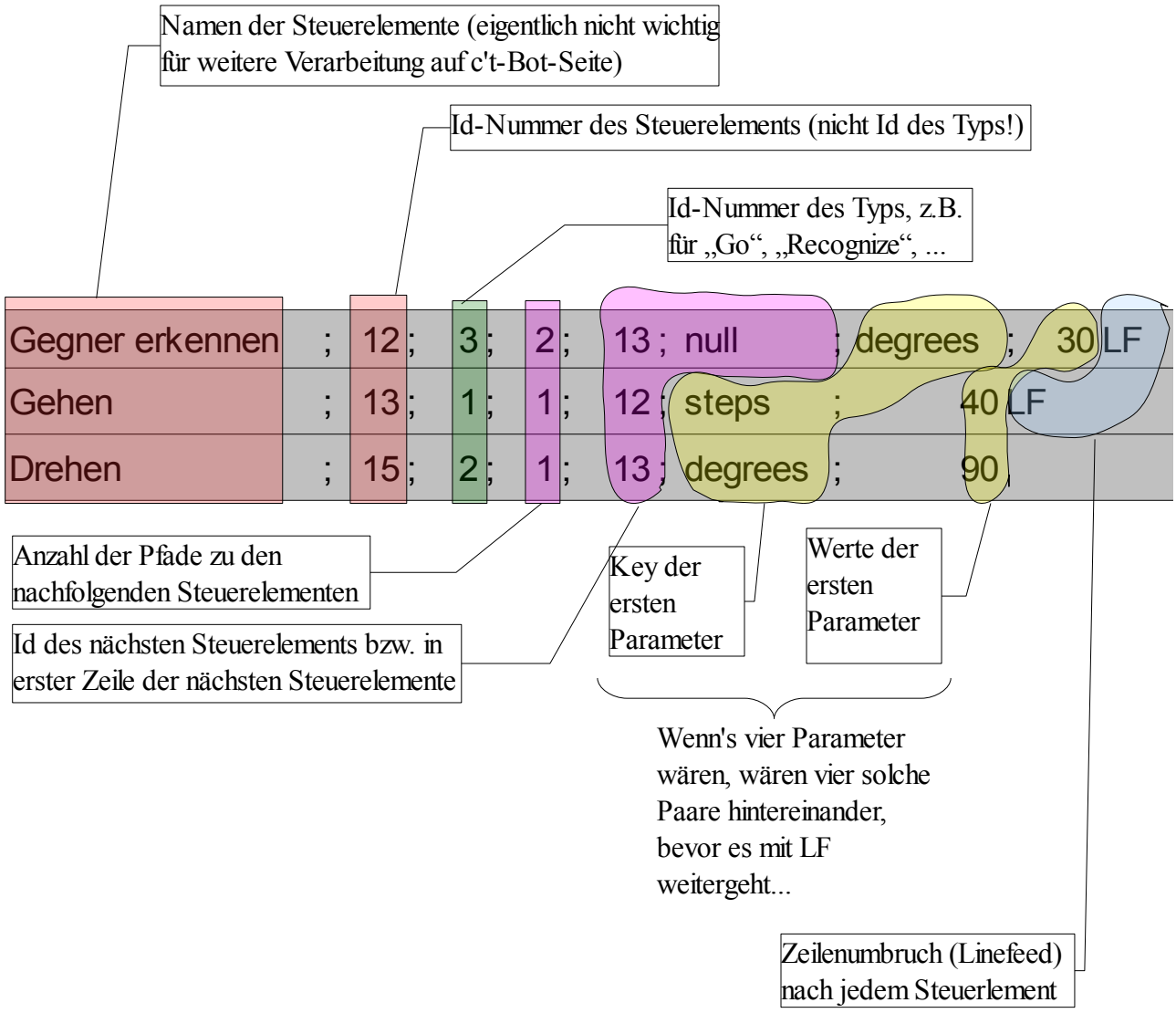
} so viele Parameter nacheinander wie es gibt!

Beispiele für Übertragung

Beispiel für Übertragung:

SOH	'0'	STX	<Programmdaten für „Fänger“>	ETX	'1'	STX	<Programmdaten für „Wegläufer“>	ETX	EOT

Beispiel für *Programmdaten*:



Spezialisierung des Protokolls zum Fangenspielen von c't-Bots

Wie oben erwähnt: Das Protokoll ist allgemein gehalten. Am Kinderuni-Tag brauchen wir das Protokoll aber nur zum Fangenspielen!

Deswegen folgende Festlegungen dazu:

Schnittstellen-Parameter

- Baudrate: 9600
- No Parity
- 8 Bits
- 1 Stop-Bit

- Zwischen SOH und EOT gibt es genau zwei *Programmdatenblöcke* – in folgender Reihenfolge:
- *Programmnummer*:
  - '0' für den Fänger
  - '1' für den Wegläufer
- **typeUID** in den *Programmdaten*:
  - 1 für „Go“ (Gehen)
  - 2 für „Turn“ (Drehen)
  - 3 für „Recognize“ (Gegner erkennen)
- Parameter in den *Programmdaten* (nur einer pro Steuerelement!):
  - Go-Steuerelement:

    Key: „steps“ – proportional zur Strecke, die gefahren wird; negativer Wert könnte rückwärts heißen.

    Value: (Integer als String)
  - Turn-Steuerelement:

    Key: „degrees“ – Drehwinkel im Gradmaß; Wert positiv: rechtsrum, negativ: linksrum.

    Value: (Integer als String)
  - Recognize-Steuerelement:

    Key: „degrees“ – Drehwinkel im Gradmaß, um den sich der Roboter maximal dreht, während er versucht dabei einen Gegner zu erkennen. Wenn er einen erkennt, geht es auf dem „ja“-Pfad weiter, wenn er am Ende keinen erkannt hat auf dem „nein“-Pfad. Wert positiv: rechtsrum, negativ: linksrum.

    Value: (Integer als String)

Vorschlag zum Einlesen des Protokolls auf der Roboter-Seite

Das Einlesen sollte mehrstufig realisiert werden. Eine split()-Funktion, die einen String und ein Trennzeichen erwartet und ein Array of String zurückgibt, wäre hilfreich!

1. Zuerst mal die ganze Übertragung (alles zwischen SOH und EOT) in einen String einlesen
2. Diesen String bei jedem ETX trennen (split(); jeder Teilstring stellt dann einen *Programmdatenblock* dar)

3. Bei jedem Teilstring das erste Zeichen als *Programmnummer* einlesen, das STX entfernen → die *Programmdaten* bleiben übrig.
4. *Programmdaten* bei jedem '\n' (LF, Zeilenumbruch) trennen (split(); jeder Teilstring stellt ein Steuerelement dar)
5. Jeden Steuerelement-Teilstring bei jedem ';' trennen (split()) und einzelne Daten zuordnen

Die Programme am besten in entsprechende Arrays eintragen.