2. Kommunikation und Synchronisation von Prozessen

2.1 Kommunikation in Rechnernetzen

In jeglicher Kommunikation gibt es (teilweise) ungeschriebene Gesetze und Regeln, die man beachten muss, um verstanden zu werden.

Manche „Regeln“ sind uns so offensichtlich, dass wir nicht darüber nachdenken:

- Wir sprechen die gleiche Sprache.
- Wir lassen einander ausreden
- Wir melden verbal oder non-verbal zurück, ob wir etwas verstanden haben
- …

Protokolle

Für die Kommunikation unter Computern oder IoT Systeme, muss die Kommunikation noch genauer reglementiert werden.

1) Da Computersysteme meist über große Netzwerke verbunden sind, muss klar sein, von wo nach wo übertragen werden muss.

2) Oft überträgt ein Computer an einen anderen mehrere Dinge gleichzeitig, daher muss jedes Paket einen Namen (oft der Dateiname) bekommen und klar sein, wo auf dem Zielrechner sie gespeichert werden soll. (Achtung: Was wenn die Datei bereits existiert?)

Solche Anforderungen an eine Kommunikation können wir mit einer formalen Sprache abbilden.

(cr = Wagenrücklauf → Rücklauf des Schlitten auf neue Zeile, lf → neue Zeile schiebt Papier hoch, Daten werden in Bytes versendet, die mit zwei Ziffern des Hexadezimalsystems dargestellt werden.)

R1: Datentransfer = {Sendung} „ende“ „<cr><lf>“.

R2: Sendung = Übertragungsrichtung Dateiname Inhalt.

R3: Übertragungsrichtung = („schicken“ | „holen“) „<cr><lf>“.

R4: Dateiname = {„a“| „b“ | … | „A“ | … | „0“ | … | „_“ | } „<cr><lf>“.

R5: Inhalt = „*“ „<cr> <lf>“ {Zeile} „*“ „<cr><lf>“.

R6: Zeile = {BinHex BinHex} „<cr><lf>“.

R7: BinHex = „0“ | … | „9“ | „A“ | … | „F“

Eine solche Absprache in Form einer formalen Sprache nennt man **Protokoll**. Im Internet kennt man z.B. das Transmission Control Protokoll /Internet Protocol oder FileTransferProtokoll.

Bsp: Protokoll aus unserer „Übertragungssprache“

Datentransfer = {Sendung} „ende“ „<cr><lf>“.

Sendung = Senden | Empfangen .

Senden =

Partner	Formalsprache	Kommentar
1	„schicken“ <cr><lf>	
2	„verstanden“ <cr><lf>	
1	Dateiname	
2	„verstanden“ <cr><lf>	
1	Inhalt	
2	„verstanden“ <cr><lf>	

Empfangen =

Partner	Formalsprache	Kommentar
1	„holen“ <cr><lf>	
2	„verstanden“ <cr><lf>	
1	Dateiname	
2	„verstanden“ <cr><lf>	
1	Inhalt	
2	„verstanden“ <cr><lf>	

Dateiname = {„a“| „b“ | … | „A“ | … | „0“ | … | „_“ | } „<cr><lf>“.

Inhalt = „*“ „<cr> <lf>“ {Zeile} „*“ „<cr><lf>“.

Zeile = {BinHex BinHex} „<cr><lf>“.

BinHex = „0“ | … | „9“ | „A“ | … | „F“

Das Protokoll ist noch sehr rudimentär. Beispielsweise gibt es noch keine Möglichkeit eine bestehende Datei zu überschreiben oder sich gegenseitig zu antworten. Es handelt sich um ein Protokoll, das nur einen „Master“ als Sender und einen „Slave“ als Empfänger hat.