6 Star-Wars-Animation streamen

Mit dem ESP32 eine Star-Wars-Animation aus dem Internet streamen? Wie soll das bloß funktionieren? Nun, es geht; allerdings dürfen wir dabei nicht allzu hohe Maßstäbe anlegen (vgl. Abb. 1). Hier zeigen wir zunächst, wie man die Animation bezieht aus dem Internet bezieht und auf dem Terminal ausgibt. Danach wollen wir an einem einfachen Beispiel deutlich machen, wie man selbst die Illusion eines sich bewegenden Objekts in einem Terminal realisieren kann.

Die Animationsdaten werden von einem Telnet-Server in den Niederlanden zur Verfügung gestellt: `towel.blinkenlights.nl`. Man kann sie mit einem Programm ähnlich wie beim Daytime-Client herunter laden (streamen) und anzeigen. Dazu öffnen wir das Programm `wlan_client_time_0.py` und nehmen dort zwei Änderungen vor: Zunächst ersetzen wir

```
server_addr = ('time.fu-berlin.de', 13)
```

durch

```
server_addr = ('towel.blinkenlights.nl', 23) .
```

Ebenso wie beim Daytime-Server braucht der Client hier keine weiteren Informationen an den Server zu senden. Der macht den ganzen Tag nichts anderes, als immer wieder nach dem Verbinden eines Clients die Star-Wars-Animation an eben diesen zu schicken.

Da bei der Animation deutlich mehr als 1024 Zeichen übertragen werden, ersetzen wir noch

```
data = client.recv(1024) # max. Anz. der Bytes
print(str(data,'UTF-8'))
```

durch die Schleife

```
while True:
   data = client.recv(500)
   print(str(data, 'UTF-8'), end='')
```

Auf den zweiten Parameter der `print`-Funktion (nämlich `end=''`) werden wir gleich noch zu sprechen kommen. Hier erst einmal ein Screenshot der Animation, die gleich nach dem Start des Programms im Thonny-Terminal zu sehen ist (s. Abb. 1). Trotz der kruden Grafik gut zu erkennen: C-3PO und R2-D2. Ein Dank an die Entwickler dieser Animation!

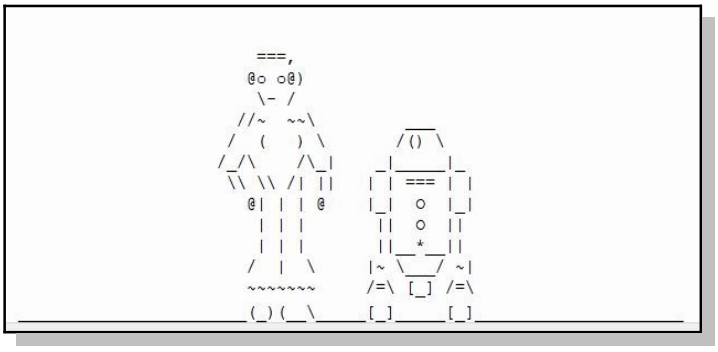

Abb. 1

Hier wird deutlich, dass sich die Einzelbilder der Animation aus einzelnen Textzeichen wie z. B. =, / oder [zusammensetzen. Damit ist klar: Der Telnet-Server sendet die Animation in Form einer (langen) Folge von Zeichen (genauer gesagt: Zeichen- und Steuercodes).

Es bleibt die Frage: Wie wird die Animation erzeugt? Dies wollen wir an einem einfachen Beispiel deutlich machen. Wir wollen einen kleinen "Stab" im Terminal-Fenster drehen lassen. Dazu geben wir den Stab in verschiedenen Lagen rasch hintereinander immer wieder an ein und derselben Stelle im Terminal-Fenster aus (z. B. am Zeilenanfang). Dies erweckt den Eindruck, dass der Stab sich dreht. Nichts anderes geschieht übrigens, wenn wir einen Film anschauen.

Für die Ausgabe im Terminal benutzen wir den `print`-Befehl. Hierbei gibt es ein Problem: Bei der Ausführung von `print('Hallo')` wird z. B. automatisch nach der Ausgabe von 'Hallo' der Cursor an den Anfang der ***nächsten*** Zeile gesetzt. Im Allgemeinen ist das bequem für uns, hier aber ist es störend. Wir können jedoch dafür sorgen, dass beim `print`-Befehl der Cursor direkt hinter der gerade ausgegebenen Zeichenkette stehen bleibt. Dies können wir mit Hilfe des eben schon benutzten `end`-Parameters von `print` erreichen: Durch die Anweisung

```
print('Hallo', end = '<--')
```

wird z. B. die Zeichenkette `'<--'` direkt hinter dem letzten Zeichen der Zeichenkette `'Hallo'` ausgegeben, also `'Hallo<--'`.

Fehlt dieser `end`-Parameter, dann setzt Micropython automatisch `end = '\n'`. `'\n'` ist eine so genannte **Escape-Sequenz**. Eine weitere Escape-Sequenz ist `'\r'`. Solche Sequenzen werden für nicht sichtbare Steuerzeichen benutzt, in diesem Fall für C**R** (Carrige Return) und LF (Line Feed). Mit CR wird der Cursor im Terminal an den Anfang der **aktuellen** Zeile gesetzt und mit LF an den Anfang der **nächsten** Zeile. Allgemein werden Escape-Sequenzen mit dem \-Zeichen eingeleitet.

Besteht nun der `end`-Parameter aus einer leeren Zeichenkette, werden hinter 'Hallo' weder sichtbare Zeichen, noch die Steuerzeichen CR oder LF ausgegeben: Der Cursor bleibt deswegen direkt hinter dem 'o' von 'Hallo' stehen. Wird eine weitere Zeichenkette mit der `print`-Funktion auf dem Terminal ausgegeben, dann beginnt diese Ausgabe an der aktuellen Cursor-Position, in diesem Fall also unmittelbar hinter dem 'o'.

Unser "rotierende Stab" wird nun durch das folgende Programm (`animation_0.py`) erzeugt:

```
while True:
  print('|', end='\r')
  sleep(0.2)
  print('/', end='\r')
  sleep(0.2)
  print('-', end='\r')
  sleep(0.2)
  print('\\', end='\r')
  sleep(0.2)
```

Die Zeichen | , / , - und \ stellen die verschiedenen Lagen des Stabes dar. (Beachten Sie: Ein einzelnes \-Zeichen leitet eine Escape-Sequenz ein. Um das Zeichen \ selbst auszugeben, benutzt man deswegen \\. (In vielen anderen Programmiersprachen verfährt man hier übrigens auf ähnliche Weise.) Nach der Ausgabe des ersten Stabs wird der Cursor durch den `end`-Parameter '\r' wieder an den Anfang ***derselben*** Zeile geschickt. ***Ohne*** diesen `end`-Parameter würde der nächste Stab ***unter*** den vorhergehenden gezeichnet. So aber erscheint der Stab in seinen verschiedenen Lagen immer an derselben Stelle: Es entsteht die Illusion einer Drehbewegung.

Die bei unserer Star-Wars-Animation benutzte Art und Weise der Darstellung von Bildern wurde übrigens schon vor mehr als hundert Jahren benutzt. Man bezeichnet sie heute als ASCII-Art.

Ein Blick hinter die Kulissen: **Terminal und Steuerzeichen**

Ein Terminal gibt die empfangenen Zeichen nicht einfach nur der Reihe nach auf dem Bildschirm aus. Vielmehr lässt sich z. B. die Position des Cursors steuern. Dies geschieht, indem man dem Terminal neben den **ausdruckbaren Zeichen** auch **nicht darstellbare Steuerzeichen** sendet. Solche Zeichen beschränken ihren Aktionsradius allerdings nicht nur auf das Steuern des Cursor (s. folgende Tabelle). Nicht darstellbare Zeichen können mit Hilfe von Escape-Sequenzen direkt in Zeichenketten eingefügt werden. (Die Escape-Sequenzen sind bei den Beispielen jeweils fett hervorgehoben.):

Zeichenkette	Ausgabe auf dem Terminal	Bemerkung
`'Alles Gute`**`\n`**`Bert'`	`Alles Gute` `Bert`	Neue Zeile (Line Feed)
`'1.`**`\t`**`Anfang'`	`1.  Anfang`	horizontaler TAB
`'Guten`**`\b\b`**`Tag'`	`GutTag`	Backspace (kurz: BS: Cursor um 1 Position nach links)
`'Hallo Bert!`**`\a`**`'`	`Hallo Bert!`	BEL: Bei Fernschreibern wurde hier nach der Ausgabe von 'Hallo Bert' eine Glocke geläutet!
`'\x41\x68'`	Ah	beliebige Zeichen durch ihre Hex-Codes einfügen

Nicht alle Steuercodes werden von sämtlichen Terminals verarbeitet. So können wir z. B. dem Thonny-Terminal kein Glöckchengeläut entlocken. Auch ist die Ausgabe auf den Terminals nicht immer einheitlich: So wird bei einigen Terminals mit `'\n'` tatsächlich der Cursor nur nach unten verschoben und dabei nicht (wie beim Thonny-Terminal und anderen) auch an den Anfang der Zeile gesetzt. Escape-Sequenzen sind nicht auf Strings beschränkt. Wir sind ihnen auch schon bei den Byte-Strings in Kapitel E (S. 13f) begegnet.