41.8 LUA-Grundlagen - Tabelle, if und Albernheit

Autor: Goetz
Quelle: Mein EEP-Forum

In diesem Kapitel möchte ich eine erste, einfache Anwendung von Lua auf einer Anlage zeigen. Ich werde mich dabei auf den Schattenbahnhof aus der Downloadbase beziehen. Aber ebensogut könnt ihr auch eine eigene Testanlage für diesen Zweck erstellen. Nehmt nur bitte nicht eure Lieblingsanlage, weil dort sonst einiges durcheinander geraten wird. Denn was ich jetzt erkläre ist keine sinnvolle Steuerung, sondern nur eine alberne Spielerei die ein wenig mehr verdeutlicht, was bestimmte Dinge in Lua tun.

Zunächst möchte ich eine einfache Tabelle vorstellen.

Auf meiner Anlage stehen im Schattenbahnhof fünf Ausfahrsignale mit den Nummern 29, 30, 31, 32 und 33. Diese Nummerierung ist nicht besonders praktisch. Und bei gewachsenen Anlagen kann sie noch viel chaotischer sein.

Um mir die Programmierung bequemer zu machen, möchte ich die Signale neu nummerieren. Und zwar als Ausfahrsignal 1 bis 5.

Dazu erstelle ich eine Tabelle mit dem Namen **AusfahrSignal**. Das ist im Grunde nichts anderes als eine Variable mit mehreren Speicherplätzen. Diese Speicherplätze können entweder durchnummeriert oder mit Namen versehen werden.

Damit die Variable **AusfahrSignal** mehrere Speicherplätze bekommt, muss ich sie so definieren:

```
AusfahrSignal = {}
```

Hinter dem Gleichheitszeichen stehen geschweifte Klammern. Die finde ich auf meiner Tastatur, indem ich die rechte AltGr Taste gedückt halte und die 7 bzw. 0 drücke.

Eine Besonderheit bei Lua ist, dass diese Tabellen automatisch mitwachsen. Die Variable hat immer so viele Speicherplätze, wie ich benötige. Deshalb muss ich beim Anlegen einer Tabelle keine Größe vorgeben.

Um einen dieser Speicherplätze anzusprechen, benötige ich die eckigen Klammern. Ich schreibe zum Beispiel

```
AusfahrSignal[1] = 29
```

um im Platz 1 der Variablen **AusfahrSignal** den Wert 29 abzuspeichern.

Und mit der gleichen Methode hole ich mir auch den Inhalt aus diesem Speicherplatz

```
print(AusfahrSignal[1])
```

gibt jetzt den Wert 29 aus, weil ich den zuvor unter der Nummer 1 abgelegt habe.

Wenn ich schon beim Erstellen einer Tabelle weiß, welche Informationen ich darin aufbewahren will, dann kann ich sie gleich bei der Definition in die geschweiften Klammern schreiben:

```
AusfahrSignal = {29,30,31,32,33}
```

Wichtig ist, dass die einzelnen Werte mit einem Komma getrennt sind, damit sie der Reihe nach den Speicherplätzen zugewiesen werden.

Mit dieser Tabelle habe ich nun die Möglichkeit meine Ausfahrsignale als **AusfahrSignal**[1] bis **AusfahrSignal**[5] anzusprechen. Was in diesem kleinen Beispiel noch wie übertriebener Aufwand scheinen mag, das erweist sich später sehr schnell als nützlich.

Zeit für ein kleines Experiment. Ich möchte, dass die Ausfahrsignale reihum zuerst auf Halt, dann auf Fahrt, dann wieder auf Halt schalten und so weiter.

Zunächst lösche ich das Ausgabefenster:
clearlog()

dann schreibe ich etwas rein, damit ich sehen kann, dass mein Skript abgearbeitet wird.

print("Lua winkt")

jetzt definiere ich meine Variablen:

TaktGeber = 0
AusfahrSignal = {29,30,31,32,33}
Halt = 2
Fahrt = 1

Weil ich die Variablen außerhalb einer Funktion definiere, sind sie überall im Skript bekannt und verfügbar. Es sind globale Variablen.

Nun folgen die Funktionen in beliebiger Reihenfolge:

function EEPMain()
print("EEPMain Aufrufe: ",**TaktGeber**)
gebeHandzeichen()
TaktGeber = **TaktGeber** + 1
return TaktGeber % 20
end

In der Hauptfunktion schreibe ich zunächst den aktuellen Wert von **TaktGeber** ins Ausgabefenster. Für das korrekte Arbeiten von Lua ist das nicht notwendig. Aber so kann ich in meiner Experimentierphase gut beobachten, was passiert.

Dann rufe ich eine Funktion **gebeHandzeichen**() auf, die ich noch definieren muss.

Als nächstes zähle ich zum **TaktGeber** 1 dazu. So kann ich zählen, wie oft die **EEPMain**() Funktion schon aufgerufen wurde.

Und zuletzt gebe ich einen Wert zurück, den ich aus dem **TaktGeber** errechne. Dazu nutze ich den Modulo Operator. Also das % Zeichen. Beim ersten Durchlauf wird mit **return** der Wert 1 zurück gegeben. Denn zu diesem Zeitpunkt habe ich zum **TaktGeber** schon das erste Mal 1 hinzugezählt. Beim zweiten Mal gebe ich dann 2 zurück, beim dritten Mal 3 und so weiter. Und solange **return** in der **EEPMain**() Funktion eine Zahl zurück gibt die größer als 0 ist, wird die Funktion erneut aufgerufen.

Beim zwanzigsten Mal macht der Modulo Operator aus der 20, die in der Variablen **TaktGeber** jetzt gespeichert ist, eine 0. Und wenn **return** den Wert 0 zurück gibt, dann wird **EEPMain**() nicht erneut

aufgerufen. Mit diesem Trick verhindere ich, dass die Signale endlos umschalten.

Zum Schluss zeigt **end** das Ende der Funktion.

Nun zur Funktion, welche die Signale bewegen soll:

```
function gebeHandzeichen()
if TaktGeber%10<5 then
EEPSetSignal(AusfahrSignal[TaktGeber%5+1],Fahrt)
else
EEPSetSignal(AusfahrSignal[TaktGeber%5+1],Halt)
end
end
```

Zunächst benutze ich wieder den Modulo Operator, um aus dem Wert im **TaktGeber** einen anderen Wert zu errechnen. Ich möchte nämlich nur Werte zwischen 0 und 9 erhalten.

Wenn dieser errechnete Wert kleiner als 5 ist, dann schalte ich ein Signal auf **Fahrt**. Andernfalls schalte ich es auf **Halt**.

Diese Unterscheidung treffe ich durch den Befehl **if** ... **then**.
zwischen den Worten **if** und **then** steht die Bedingung, die erfüllt sein muss. In diesem Fall muss der Wert **TaktGeber**%10 kleiner als 5 sein.
In der nächsten Zeile steht dann, was passieren soll, wenn die Bedingung erfüllt ist. Dann folgt ein **else**, zu deutsch "andernfalls".
In der darauffolgenden Zeile steht, was passieren soll wenn die Bedingung nicht erfüllt wurde. Und zuletzt muss ich den **if** ... **then** Teil mit einem **end** abschließen. Sonst würde auch alles, was danach im Skript steht nur dann ausgeführt, wenn die Bedingung nicht erfüllt ist.

Die Funktion selber muss ebenfalls mit **end** abgeschlossen werden. Auch dann, wenn danach im Skript nichts mehr steht!

Folgendes passiert, wenn man EEP mit diesem Skript startet:

Beim ersten Aufruf der **EEPMain**() Funktion steht der **TaktGeber** auf 0
In der Funktion **gebeHandzeichen**() wird geprüft, ob 0%10 kleiner als 5 ist. Die Bedingung ist erfüllt, also wird die nächste Zeile ausgeführt.

In der nächsten Zeile wird mit **EEPSetSignal**() ein Signal geschaltet. Das Signal steht in der Tabelle **AusfahrSignal**. Der Platz, an dem das Signal in der Tabelle zu finden ist, steht in eckigen Klammern. Statt einer Zahl steht hier eine Berechnung. Nämlich **TaktGeber**%5+1. Der **TaktGeber** ist 0, **Taktgeber**%5 ist also ebenfalls 0 und wenn ich dann noch 1 hinzu rechne, dann bekomme ich 1.

In der Tabelle steht unter **AusfahrSignal**[1] der Wert 29.

Also wird Signal 29 umgeschaltet. Und zwar in die Stellung **Fahrt**.

Statt der komplizierten Zeile hätt man auch einfach
EEPSetSignal(29,1)
schreiben können. Der Effekt wäre der gleiche.

Weil die Bedingung zwischen **if** und **then** erfüllt war, wird der Teil nach **else** übersprungen. Damit ist das Ende der Funktion erreicht und es geht wieder zurück zur Funktion **EEPMain**(). Dort wird zu **TaktGeber** 1

hinzu gezählt. Dieser neue Wert, also 1, wird dann aus **EEPMain**() zurück gegeben. Das sorgt dafür, dass die **EEPMain**() Funktion kurz darauf erneut aufgerufen wird.

Und dieses Mal ist der Wert im **TaktGeber** 1.
Also ist die Bedingung in der Funktion **gebeHandzeichen**() noch immer wahr.
Das Signal, welches angesprochen wird, ist das Signal, welches in **AusfahrSignal**[2] gespeichert wurde.
Denn **TaktGeber**%5+1 ist 2.

Es wird also Signal 30 auf **Fahrt** gestellt.
Der Rest spielt sich dann genauso ab, wie im ersten Durchlauf.

In der dritten Runde wird Signal 31 auf **Fahrt** gestellt. Dann Signal 32 und in der fünften Runde das Signal 33.

Beim sechsten Durchlauf ist der Wert in **TaktGeber** 5.

Die Bedingung **TaktGeber**<5 ist also nicht mehr erfüllt. Denn **TaktGeber** ist nicht kleiner, sondern gleich 5. Damit wird der Teil ausgeführt, der hinter dem **else** steht.

Die Stelle in der Tabelle, an der die Signalnummer steht, ist **TaktGeber**%5+1. **TaktGeber**%5 ist 0. Also steht die Signalnummer unter **AusfahrSignal**[1]. Das bedeutet, dass im sechsten Durchlauf das Signal 29 auf **Halt** gestellt wird. Im siebten Durchgang wird dann Signal 30 auf **Halt** gestellt. Dann Signal 31, 32 und im zehnten Durchgang das Signal 33.

Im elften Durchlauf wird dann Signal 29 erneut auf **Fahrt** gestellt. Und so weiter, bis nach 20 Durchgängen die **EEPMain**() Funktion nicht mehr aufgerufen wird.

Diese kleine Spielerei hat keinen praktischen Nutzen.
Sie soll nur verdeutlichen, was Variablen, Tabellen, der Modulo Operator und die if... then Verzweigung tun.

Da ich Wert darauf lege mit farbigen Kennzeichnungen **Variablen**, **Befehle** und **Funktionen** deutlich zu unterscheiden, kann ich das Skript nicht als Quellcode darstellen. Und damit habe ich auch nicht die Möglichkeit, Teile des Codes einzurücken. Das ist für die Lesbarkeit aber sehr wichtig. Deshalb ist hier das Skript noch einmal komplett als Quellcode:

Quellcode

```
clearlog()
print("Lua winkt")

TaktGeber=0
AusfahrSignal={29, 30, 31, 32, 33}
Halt=2
Fahrt=1

function EEPMain()
  print("EEPMain Aufrufe: ", TaktGeber)
  gebeHandzeichen()
  TaktGeber=TaktGeber+1
  return TaktGeber % 20
```

```
end

function gebeHandzeichen()
   if TaktGeber%10<5 then
      EEPSetSignal(AusfahrSignal[TaktGeber%5+1],Fahrt)
   else
      EEPSetSignal(AusfahrSignal[TaktGeber%5+1],Halt)
   end
end
```

Durch Einrücken kann man gut sichtbar machen, was zusammen gehört.

Beispielsweise alles, was zwischen
function EEPMain()
und dem zugehörigen
end
steht.

function bildet sozusagen den Anfang des Blocks, **end** sein Ende. Alles, was dazwischen steht, rückt man gleich weit ein. Wie weit genau, das ist jedem selbst überlassen. Ich benutze gerne die Tab Taste dafür, weil sie bequem ist. Manche Programmierer bevorzugen vier Leerzeichen, weil die Tabulator Position nicht fest definiert ist. Im Code Block habe ich die "vier Leerzeichen" Methode verwendet.

Im nächsten Kapitel will ich eine Variante der Einfahrkontrolle für den Schattenbahnhof vorstellen.