Verein zur Förderung der ICT-Berufsbildung

ICT-Schnuppertage

Elektroniker/in

Autor:
Ervin Mazlagić

Auftraggeber:
Freddy Ringier

8. Juli 2013

Inhaltsverzeichnis

1 Aufgaben

Auf den folgenden Seiten finden Sie viele verschiedene Aufgaben die Sie während dem Kurs in Angriff nehmen können. Beachten Sie hierzu bitte die folgenden Hinweise:

1. Sie müssen nicht alle Aufgaben im Kurs lösen.
2. Sie können die Kursunterlagen nach Hause nehmen und dort weiter Programmieren.
3. Die Aufgaben sind so gestellt, dass Sie eine nach der anderen lösen müssen.
4. Sie müssen die Aufgaben nicht alleine lösen sondern können in Teams bearbeiten.

Hier sehen Sie eine Übersicht der Aufgaben:

Aufgabe	Themen	Schwierigkeit
1	Pausen, Ausgaben, absolute Sprünge	einfach
2	Pausen, Ausgaben, absolute Sprünge	einfach
3	Pausen, Ausgaben, absolute Sprünge	einfach
4	Pausen, Ausgaben, relative Sprünge	einfach
5	Eingaben, einfache Verzweigungen	mittel
6	Eingaben, geschachtelte Verzweigungen	mittel
7	Zählerschlaufen, Zeitberechnung	mittel
8	Zählerschlaufen, Theorie zu Variablen	mittel
9	Zählerschlaufen, Variablen	schwer

Tabelle 1: Übersicht der Aufgaben

Wer gerne weitere Aufgabe haben möchte, kann sich beim Instruktor melden.

1.1 Aufgabe 1

Betrachten Sie nochmals das Testprogramm.

Listing 1: Testprogramm

```
'{$STAMP BS1}

start:  HIGH  0         'LED 0 einschalten
        PAUSE 500       'halbe Sekunde warten
        LOW   0         'LED 0 ausschalten
        PAUSE 500       'halbe Sekunde warten
        GOTO  start     'springe zu "start"
```

(a) Versuchen Sie zu zweit oder zu dritt zu verstehen, wie das Programm arbeitet.

(b) Versuchen Sie alleine das Programm langsamer zu machen.

(c) Ändern Sie das Programm so, dass eine andere LED blinkt.

1.2 Aufgabe 2

Betrachten Sie folgendes Programm.

```
'{$STAMP BS1}

start:  HIGH  0         'LED 0 einschalten
        PAUSE 500       'halbe Sekunde warten
        LOW   0         'LED 0 ausschalten

        HIGH  1         'LED 1 einschalten
        PAUSE 500       'halbe Sekunde warten
        LOW   1         'LED 1 ausschalten

        HIGH  2
        PAUSE 500
        LOW   2

        HIGH  3
        PAUSE 500
        LOW   3

        HIGH  4
        PAUSE 500
        LOW   4

        HIGH  5
        PAUSE 500
        LOW   5

        HIGH  6
        PAUSE 500
        LOW   6

        HIGH  7
        PAUSE 500
        LOW   7

        GOTO  start     'springe zu "start"
```

(a) Versuchen Sie zu erraten was das Programm macht.

(b) Laden Sie das Programm auf Ihr `BASIC Stamp` und überprüfen Sie Ihre Vermutung.

(c) Versuchen Sie ein Programm zu schreiben, welches genau umgekehrt läuft.

1.3 Aufgabe 3

Können Sie sich erinnern an die alte Fernsehserie *Knight Rider*? In dieser gab es ein Auto Namens Kit welches einen Computer eingebaut hatte. Dieses war sehr berühmt für sein cooles Frontlicht welches immer hin und her lief.

(a) Versuchen Sie ein Programm zu schreiben, welches die LEDs wie bei Kit hin und her blinken lässt. Als Hilfe haben Sie unten eine Zeichnung die den Ablauf nochmals beschreibt.
Tipp: Sie haben in der Aufgabe 2 bereits etwas ähnliches programmiert.

(b) Vergleichen Sie Ihre Lösung mit der von anderen und besprechen Sie diese zusammen.

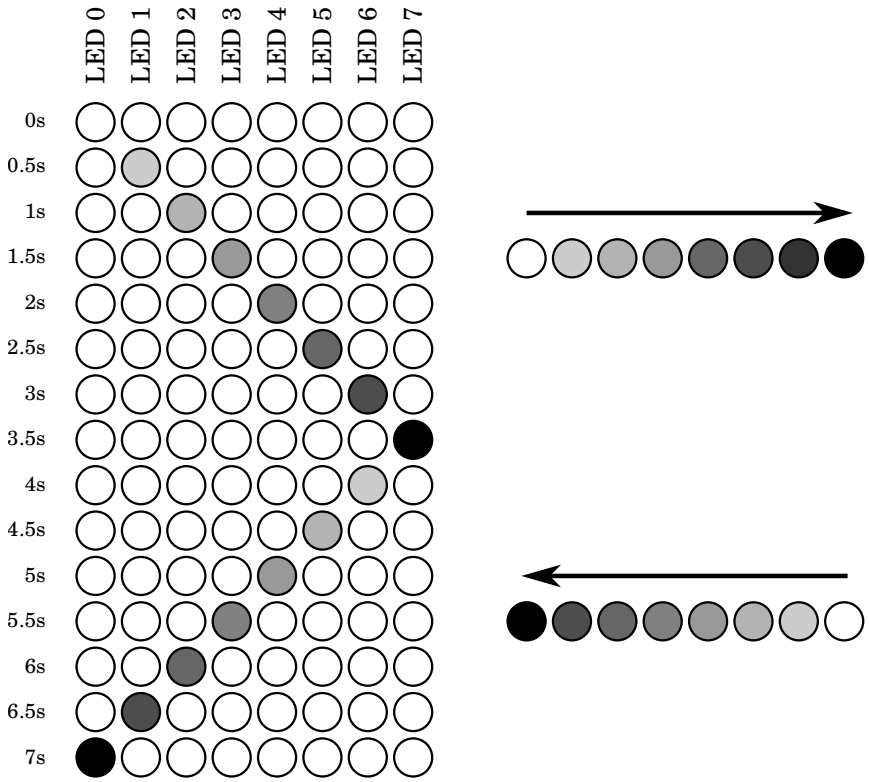

Abbildung 1: Hilfszeichnung zum Knight Rider

1.4 Aufgabe 4

In der Aufgabe 3 haben Sie vielleicht bemerkt, dass sich vieles im Programm wiederholt. Teile die sich in Programmen wiederholen sind wo immer möglich zu vermeiden, denn sie machen den Code schwerfällig. Was das heisst merken Sie bestimmt in der folgenden Aufgabe.

(a) Verändern Sie Ihr Programm aus Aufgabe 3 so, dass es doppelt so schnell wird.

(b) Was ist Ihnen aufgefallen beim lösen der Aufgabe (a)?

(c) Nehmen Sie Ihr Programm aus Aufgabe (4a) und machen Sie folgende Anpassung:

- Ersetzten Sie jede Zeile in der

```
1     PAUSE
```

  vorkommt mit der Zeile

```
1     GOSUB warten
```

- Fügen Sie ganz am Ende des Programms die folgenden zwei Zeilen hinzu:

```
1 warten:  PAUSE 500
2          RETURN
```

(d) Versuchen Sie zu erraten was das Programm nun machen wird.

(e) Laden Sie das Programm auf Ihr `BASIC Stamp` und beobachten Sie was passiert.

(f) Ändern Sie die Geschwindigkeit des Lauflichts so, dass es Ihnen gefällt.

1.5 Aufgabe 5

Betrachten Sie folgendes Programm.

```
'{$STAMP BS1}

INPUT 0                  'Definiere PIN 0 als Eingabe für die Taste 0

start:  IF PIN0 = 1 THEN abc      'Falls die Taste 0 = 1 ist, dann
                                  'springe zu "abc", sonst geh zur
                                  'nächsten Zeile

        LOW  4                    'LED 4 ausschalten
        GOTO start                'springe zu "start"

abc:    HIGH 4                    'LED 4 einschalten
        GOTO start                'springe zu "start"
```

(a) Versuchen Sie zu erraten, was das Programm macht.

(b) Laden Sie das Programm auf Ihr `BASIC Stamp` und experimentieren Sie. Was stellen Sie fest?

1.6 Aufgabe 6

In der Aufgabe 5 haben Sie gelernt wie man eine Eingabe macht und darauf reagieren kann mit einer Ausgabe.

(a) Nehmen Sie das Programm aus Aufgabe 5 als Grundlage und erstellen Sie ein neues Programm, welches für jeden Taster jeweils eine LED leuchten lässt wie es in der Leuchttabelle beschrieben ist. Die LEDs sollen nur so lange leuchten wie die Tasten gedrückt sind.

Eingabe		**Ausgabe**
Taste 0 = 1	→	LED 4 leuchtet
Taste 1 = 1	→	LED 5 leuchtet
Taste 2 = 1	→	LED 6 leuchtet
Taste 3 = 1	→	LED 7 leuchtet

Tabelle 2: Leuchttabelle zu (6a)

(b) Ändern Sie das Programm aus Aufgabe (a) so, dass jeweils zwei Tasten gedrückt werden müssen um eine LED leuchten zu lassen.

Eingabe		**Ausgabe**
Taste 0 = 1 und Taste 1 = 1	→	LED 5 leuchtet
Taste 2 = 1 und Taste 3 = 1	→	LED 7 leuchtet

Tabelle 3: Leuchttabelle zu (6b)

(c) Wie viele Kombinationen von Eingaben gibt es mit vier Tastern?

1.7 Aufgabe 7

Betrachten Sie folgendes Programm.

```
'{$STAMP BS1}

INPUT 0          'Definiere PIN0 als Eingabe für die Taste 0

start:  IF PIN0 = 1 THEN show7  'Falls Taste 1 gedrückt gehe zu show7
        GOTO  start

show7:  FOR B2 = 0 TO 10        'Zähle B2 von 0 bis 10
            HIGH  7
            PAUSE 500
            LOW   7
            PAUSE 500
        NEXT                    'Falls B2<10 springe zum FOR
                                'sonst weiter im Programm
        GOTO start
```

(a) Versuchen Sie zu erraten was das Programm macht.

(b) Tippen Sie das Programm ab und laden es auf Ihr `BASIC Stamp`.

(c) Erweitern Sie das Programm so, dass die LED 7 für 15 Sekunden blinkt, wenn die Taste 0 gedrückt wird. Das Blinken soll dabei einen Takt von 300 Millisekunden haben.

1.8 Aufgabe 8

Betrachten Sie folgendes Programm.

```
'{$STAMP BS1}

INPUT 0                                 'Definiere PIN0 als Eingabe
OUTPUT 4                                'Definiere PIN4 als Ausgabe

SYMBOL Taste0 = PIN0                    'PIN0 soll "Taste0" heissen
SYMBOL LED = B2                         'B2 soll "LED" heissen

LED = 4                                 'Wert 4 wird in LED gespeichert

start:  IF Taste0 = 1 THEN glow         'Falls Taste grdückt, dann "glow"
        GOTO dark                       'springe zu "dark"

glow:   HIGH LED                        'setze LED auf Eins (einschalten)
        GOTO start                      'springe zu "start"

dark:   LOW  LED                        'setze LED auf Null (ausschalten)
        GOTO start                      'springe zu "start"
```

(a) Lesen Sie die Theorie zum Thema Platzhalter im Kapitel 2.4.

(b) Versuchen Sie mit Hilfe der Theorie aus (a) das Programm zu verstehen.

(c) Laden Sie das Programm auf Ihr `BASIC Stamp` und untersuchen Sie, ob Ihre Annahmen stimmen.

1.9 Aufgabe 9

Betrachten Sie folgendes Programm.

```
'{$STAMP BS1}

SYMBOL LED = B2                 'B2 soll "LED" heissen

LED = 0                         'Wert 0 wird in LED gespeichert

start:  FOR B3 = 0 TO 7         'Zähle B3 von 0 bis 7
            HIGH  LED
            PAUSE 1000
            LOW   LED
            LED = LED + 1       'erhöhe den Wert in LED um 1
        NEXT                    'Falls B3<7: erhöhe um 1, sonst weiter

        LED = 0                 'Wert 0 in LED speichern

        GOTO  start             'springe zu "start"
```

(a) Versuchen Sie zu verstehen was das Programm macht.

(b) Laden Sie das Programm auf Ihr `BASIC Stamp` und untersuchen Sie, ob Ihre Annahmen stimmen.

(c) Entwickeln Sie ein Knight Rider Lauflicht mit Variablen und **`FOR ... NEXT`** Schlaufen wie im Beispielcode.

(d) Erweitern Sie das Programm aus (c) so, dass zwei Knight Rider entgegengesetzt laufen.
Tipp: Wenn zwei Knight Rider entgegenlaufen, so sind immer die folgenden LEDs gleichzeitig ein:

⋮	⋮
LED 0	LED 7
LED 1	LED 6
LED 2	LED 5
LED 3	LED 4
LED 4	LED 3
LED 5	LED 2
LED 6	LED 1
LED 7	LED 0
⋮	⋮

Tabelle 4: Hilfstabelle zum entgegenlaufenden Knight Rider

2 Theorie und Hinweise zu BASIC und BASIC Stamp

2.1 Was ist BASIC?

BASIC steht für *Beginner's All-purpose Symbolic Instruction Code* und bedeutet so viel wie *symbolische Allzweck-Programmiersprache für Anfänger*. Diese Programmiersprache wurde 1964 entwickelt und bis heute ständig erweitert. Das BASIC Stamp verwendet einen neueren Dialekt dieser Programmiersprache. D.h. es ist sehr ähnlich aber nicht genau gleich.

Listing 2: Echter BASIC Code

```
10  INPUT "Geben Sie bitte Ihren Namen ein"; A$
20  PRINT "Guten Tag, "; A$
30  INPUT "Wie viele Sterne möchten Sie?"; S
35  S$ = ""
40  FOR I = 1 TO S
50  S$ = S$ + "*"
55  NEXT I
60  PRINT S$
70  INPUT "Möchten Sie noch mehr Sterne?"; Q$
80  IF LEN(Q$) = 0 THEN GOTO 70
90  L$ = LEFT$(Q$, 1)
100 IF (L$ = "J") OR (L$ = "j") THEN GOTO 30
110 PRINT "Auf Wiedersehen";
120 FOR I = 1 TO 200
130 PRINT A$; " ";
140 NEXT I
150 PRINT
```

Berühmt wurde diese Programmiersprache unter anderem durch den C64, ein Computer aus dem Jahre 1984. Dieser hatte kein Betriebssystem, wie Sie das auf Ihrem Computer kennen und gewohnt sind (Windows, GNU/Linux, MacOS, ...), sondern er hatte einen BASIC-Interpreter. D.h. man konnte "programmieren".

Wer gerne mehr über die Programmiersprache BASIC oder den C64 erfahren möchte, findet auf der freien Enzyklopädie Wikipedia gute Informationen (siehe http://de.wikipedia.org/wiki/BASIC und http://de.wikipedia.org/wiki/C64).

2.2 Entwicklungsumgebung

Die Entwicklungsumgebung, die zur Programmierung des BASIC Stamp benötigt wird, kann auf der Homepage des Herstellers bezogen werden auf

http://www.parallax.com

Auf dieser Seite navigieren Sie zu Downloads > BASIC Stamp Software. Dort finden Sie die Software für Windows, Macintosh und Linux.

Auf der Homepage finden Sie auch weitere Informationen zur Programmiersprache.

2.3 Instruction Set

Wie jede Programmiersprache hat auch BASIC ein sogenanntes Instruction Set. Diese zeigt auf, welche Befehle es in der Programmiersprache gibt und wie man diese einsetzt. Der Hersteller www.parallax.com bietet hierzu eine grosse Dokumentation an. Wir möchten hier nur auflisten, was für uns im Kurs wichtig ist.

Befehl	Beispiel	Beschreibung
END	END	Ende/Bei Reset startet das Programm neu.
FOR ...NEXT	FOR B2 = 0 TO 255 ...NEXT	Zählt von 0 bis 255, erst dann geht es zur Zeile nach dem NEXT.
GOSUB	GOSUB label ...RETURN	GOSUB: Springe zu label. RETURN: Springe zurück zur Zeile nach GOSUB.
GOTO	GOTO label	Springe zu label.
HIGH	HIGH 3	Setze Pin 3 auf logisch 1 (d.h. einschalten).
IF ...THEN	IF B2 = 1 THEN label	Falls B2 gleich 1 ist, dann springe zu label.
INPUT	INPUT 5	Pin 5 als Eingabe definieren.
LOW	LOW 3	Setze Pin 3 auf logisch 0 (d.h. ausschalten).
OUTPUT	OUTPUT 3	Pin 3 als Ausgabe definieren.
PAUSE	PAUSE 100	Macht nichts für 100 Millisekunden.
TOGGLE	TOGGLE 5	Invertiere Pin 5. Invertieren heisst umkehren, d.h. aus 0 wird 1 und aus 1 wird 0.

Wer gerne auch nach dem Kurs Programme schreiben möchte, kann beim Instruktor nach einem grösseren Instruction Set nachfragen.

2.4 Platzhalter

Platzhalter oder Variabeln vereinfachen die Programmierung, denn Sie erlauben es, bestimmten Dingen eigene Namen zu geben usw.

Sie haben in der Aufgabe aus 1.7 gelernt, dass z.B. hinter der Bezeichnung `PIN0` der Taster steht. In Ihrem Programm muss dieser nicht zwingend `PIN0` heissen, denn Sie können einen eigenen Namen für diesen erstellen z.B. `Taste0`.

Um solche Variablen zu erstellen, braucht es Speicher (RAM). Leider gibt es beim `BASIC Stamp` nicht sehr viel davon sondern nur die folgenden Speicherstellen:

Wort	**Byte**	**Bit**
W0	B0	Bit 0–7
	B1	Bit 8–15
W1	B2	Bit 0–7
	B3	Bit 8–15
W2	B4	Bit 0–7
	B5	Bit 8–15
W3	B6	Bit 0–7
	B7	Bit 8–15
W4	B8	Bit 0–7
	B9	Bit 8–15
W5	B10	Bit 0–7
	B11	Bit 8–15

Tabelle 5: RAM-Übersicht für Variablen

Lassen Sie sich nicht einschüchtern oder verwirren durch die obige Tabelle. Diese zeigt lediglich auf, dass das `BASIC Stamp` in Worten speichert, in denen jeweils zwei Byte liegen. Ein solches Byte hat jeweils 8 Bit.

$$\text{ein Wort} \quad = \quad \text{zwei Byte} \quad = \quad 2 \cdot (8 \text{ Bit}) \quad = \quad 2^{16} \quad = \quad 65536$$

Beim arbeiten mit diesem Speicher muss man sich entscheiden, was man verwenden möchte. Benutzt man das Wort `W0` so kann man nicht auch noch die Bytes `B0` und `B1` verwenden, da diesen ja im Wort `W0` enthalten sind. Dies gilt natürlich für alle Speicherplätze.