Workshop
Arduino-Programmierung #2

Variablen, Funktionen, Vergleiche, Arrays, Morsen

Joachim Baur

E-Mail: post@joachimbaur.de

ZTL-Alias: @joachimbaur

Download für diesen Workshop: www.joachimbaur.de/WS2.zip

Variablen

- Im Code mit Namen bezeichnete „Behälter“, die Werte eines bestimmten Typs enthalten

Typ	**Speicherbedarf**	**Wertebereich**
byte	8 Bit	0 … 255
int	16 Bit	-32768 … 32767
float	32 Bit	Kommazahlen (zB 3.1415)
unsigned long	32 Bit	0 … 4 294 967 295
bool	1 Bit	true/false (entspricht 1 oder 0)

- Anlegen einer Variablen durch Angabe von Typ + Name + (optional) Wert
- Name: Nur Buchstaben, Zahlen und Unterstrich sind erlaubt
- Zuweisen/Ändern eines Wertes durch „=“

```
int zaehlerEins;        // Variable zaehlerEins enthält jetzt
den (Standard-)Wert 0
byte mein_alter = 54;   // Variable mein_alter hat den (Start-
)Wert 54
```

https://www.arduino.cc/reference/en/#variables

Zahlensysteme

- `int a = 10;` `// Dezimalsystem, Ganzzahl`
- `int c = 0x0A;` `// Hexadezimalsystem (0…9 + A…F)`
- `int d = 0b1010;` `// Binärsystem, 1010 = dez. 10 = hex. 0A`

Es ist egal, in welchem Zahlensystem ein Wert einer Variablen zugewiesen wird!

Farbe:	Rot
RGB-Werte:	255, 0, 0
HTML/CSS:	"#FF0000" bzw. kurz "#F00"

```
int rotWert = 255;
int rotWert = 0xFF;
int rot = 0xFF0000;    // besser lesbar als: int rot = 16711680
```

https://bin-dez-hex-umrechner.de/

Vergleichoperatoren

==	ist gleich wie
>	ist größer als
>=	ist größer als oder gleich wie
<	ist kleiner als
<=	ist kleiner als oder gleich wie
!=	ist ungleich

Beispiele:

1 == 2	Ergebnis: false, da 1 nicht gleich 2 ist
4 >= 4	Ergebnis: true, da 4 größer oder gleich 4 ist

ACHTUNG:

```
a = 1  // Zuweisung: Variable a wird auf den Wert 1 gesetzt
a == 2 // Vergleich: der Wert von a wird mit 2 verglichen
```

Vergleichsfunktion „if“ (+ „else“)

```
if ( a == 7 ) ...;  // Code-Zeile, die ausgeführt wird,
          // wenn der Wert der Variablen a genau 7 ist

if ( a < 10 ) {
   // Code-Block, der ausgeführt wird,
   // wenn die Variable a einen Wert kleiner als 10 enthält
} else {
   // Code-Block, der ausgeführt wird,
   // wenn die Variable a einen Wert größer oder gleich 10 enthält
}


if ( a < 10 ) {
   ... a von 0 bis 9
} else if ( a < 100 ) {
   ... a von 10 bis 99 (nicht von 0 bis 99!)
} else {
   ... a ist 100 o0der größer
}
```

Logische Operatoren

- Logisches „Und“: &&
- Logisches „Oder“: ||
- Logisches „Nicht“: ! (gleich: ==, ungleich: !=)

```
if ( ( a == 2) || ( a == 7 ) ) {
    ... // wenn a gleich 2 oder a gleich 7 ist
}

if ( ( a < 0 ) && ( b < 0 ) && ( c < 0 ) ) {
    ... // wenn a, b und c alle kleiner Null sind
}

if ( a != 2 && a != 3 ) {
    ... // wenn a ungleich 2 und a ungleich 3 ist
}

if ( a == b ) {
    ... // Vergleich zweier Variablen
}
```

Blink ohne delay()

Sketch: 01_BlinkMillis.ino

```
// GPIO Pin, an den die LED angeschlossen ist
// LED_BUILTIN = vordefinierter Variablenwert, Pin 13 bei Arduino Mega
const int ledPin = LED_BUILTIN;

// LOW und HIGH sind ebenfalls vordefinierte Werte (LOW = 0, HIGH = 1)
int ledSpannung = LOW;

// Zeit in Millisekunden, zu der der nächste Blink-Wechsel stattfinden soll
unsigned long naechsterLedWechsel = 0;

// Dauer in Millisekunden zwischen An/Aus-Wechsel
const int blinkDauer = 1000;

void setup() {
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, ledSpannung);
}
```

const Variable wird später nicht mehr verändert, kann daher im Arduino anders (platzsparender) gespeichert werden

Scope (Geltungsbereich):
Variablen, die außerhalb von setup() oder loop() definiert werden, gelten „global“ (überall) im Sketch

unsigned long = unbedingt nötig für Zeitmessung (Millisekunden seit Start des Arduinos)

Sketch: 01_BlinkMillis.ino

```
. . .

void setup() {
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, ledSpannung);
}

void loop() {
    // millis() gibt die Millisekunden seit dem Start des Arduino an
    // zählt einfach hoch, +1000 pro Sekunde bis ca. 4 Milliarden

    if (millis() > naechsterLedWechsel) {
        naechsterLedWechsel = millis() + blinkDauer;

        if (ledSpannung == LOW) {
            // die LED-Spannung ist auf "LOW"
            ledSpannung = HIGH;
        } else {
            // die LED-Spannung ist auf "HIGH"
            ledSpannung = LOW;
        }

        digitalWrite(ledPin, ledSpannung);
    }
}
```

Funktionsaufrufe mit Argumenten

- Funktionen können wie Variablen beliebige Namen haben
- Argumente (Variablen, Werte) werden an Funktionen innerhalb der runden Klammern übergeben
- Die Signatur der Funktion muss die gleichen Variablen-Typen und Variablenanzahl entgegen nehmen wie der Funktionsaufruf sendet

```
void loop() { // Funktion wird hier benannt und implementiert (Code-Block)
}

void setzeLedAusgang( int pinNr, int highOderLow ) {
    digitalWrite( pinNr, highOderLow );
}

setzeLedAusgang( 13, 1 ); // 1 = HIGH (5V), LED an Pin 13 anschalten

const int MEIN_PIN = 4;
setzeLedAusgang( MEIN_PIN, 0 ); // 0 = LOW (0V), LED an Pin 4 ausschalten
```

Rückgabewerte von Funktionen

- Funktionen können Rückgabewerte (Rechenergebnisse z.B.) mit „return“ an den Funktionsaufruf zurückgeben
- Die Funktionssignatur hat an 1. Stelle den Typ des Rückgabewerts

```
int multipliziere( int nummer1, int nummer2 ) {
    int ergebnis = nummer1 * nummer2;
    return ergebnis;
}

int multipliziere( int nummer1, int nummer2 ) {
    return nummer1 * nummer2;
}

int zweiMalDrei = multipliziere( 2, 3 );
// Variable zweiMalDrei enthält das Ergebnis 6

multipliziere( 2, 3 );
// auch erlaubt: Funktion wird ausgeführt, Ergebnis aber verworfen/ignoriert
```

For-Schleifen

- Wenn Funktionsaufrufe oder Berechnungen mehrere Male wiederholt werden sollen, werden u.a. „for“-Schleifen benutzt
- Eine „for“-Schleife benutzt eine Schleifenvariable, um die Durchgänge zu zählen bzw. die Abbruchbedingung zu prüfen
- Die „for“-Schleifen-Definition hat 3 Teile (durch Strichpunkte getrennt)

```
int summe = 0;

for (int i = 0; i < 10; i++ ) {
    summe = summe + 1;
}

// summe ist jetzt 10

// „i++“ ist dasselbe wie „i = i + 1“ bzw. „i += 1“
// es gibt auch i--

// Aufgabe: Schleife, die Startvariable summe = 10 herunterzählt auf 0?
```

Morsen (sort of…)

- Das Programm soll die eingebaute LED blinken lassen
- 1x blinken, dann Pause
- 2x blinken, dann Pause
- 3x blinken, dann Pause
- 4x blinken, dann Pause
- 5x blinken, dann Pause
- Danach das alles wieder von vorne (1x blinken)…

- Wir benutzen dazu Schleifen und Funktionen

Sketch: 02_Morsen.ino

```
5   const int ledPin = LED_BUILTIN; // 13 beim Mega
6
7   void setup() {
8     pinMode( ledPin, OUTPUT);
9     digitalWrite( ledPin, LOW );
10  }
11
12  void loop() {
13    // 5x die Morsen-Schleife ausführen:
14    for (int i=1; i<=5; i++) {
15      blinkeAnzahl( i );
16      delay(500);
17    }
18  }
19
20  void blinkeAnzahl( int anzahl ) {
21    for (int i=0; i<anzahl ; i++ ) {
22      digitalWrite( ledPin, HIGH );
23      delay( 300 );
24      digitalWrite( ledPin, LOW );
25      delay( 300 );
26    }
27  }
```