VME-STIFTUNG
Osnabrück - Emsland

ARDUINO - CODEREFERENZ

VONRAVENSBERG SCHULE

PROGRAMM-STRUKTUR

Grundstruktur für ein Programm (Sketch)

```
void setup () {
  // Anweisungen, die nur ein Mal zu Beginn laufen
}

void loop () {
  // Anweisungen, die immer wieder durchlaufen
}
```

Falls programmübergreifende **Variablen** festgelegt oder **Bibliotheken** geladen werden sollen, kann dies vor der **Funktion** void () geschehen.

VARIABLEN, ARRAYS & DATENTYPEN

Datentypen

```
byte          // Zahl zwischen 0 und 255
int           // Ganze Zahl zwischen -32.768 und 32.767
unsigned int  // Zahl zwischen 0 und 65.535
long          // Ganze Zahl zwischen -2.147.483.647und 2.147.483.647
float         // für große Dezimalzahl
char          // Werte von ASCII-Zeichen
boolean       // TRUE oder FALSE
```

String

```
char string1[x]                  // String mit x+1 Positionen
char string2[4] = {‚a' , ‚b' , ‚c'} // String mit Zeichen a, b, c und 0-Ende
char string3[ ] = „arduino“      //  Zeichenkette in String speichern
```

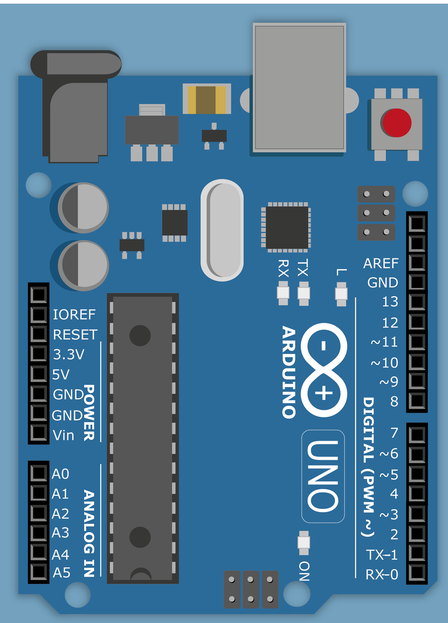
Arrays

```
int array1[x]                  // Array mit x+1 Positionen
int array2[] = {8, 9, 11, 10}  // Array anlegen und mit Werten füllen
x = array2[0]   // Abfrage des Wertes 8 an Position 0 des Arrays array2
array2[3]=23    // Wert 23 an Position 4 des Arrays speichern
```

KONTROLLSTRUKTUREN

```
if (x > 5) { … } else { … }
while (x < 10) { … }
for (int i = 0 ; i < 10 ; i++) { … }
do { … } while (x < 10)
break    // Beende den loop-Teil
```

```
switch (x) {
  case 8:   // Wenn x = 8, dann …
    …
    break;
  default:  // Für alle anderen Fälle
    ….
  }
```



Hier geht‘s zur ausführlichen Code-Referenz auf arduino.cc

OPERATOREN

Arithmetik

```
=    // Zuordnung z.B. y = 3
+    // Addition
-    // Subtraktion
*    // Multiplikation
/    // Division
%    // Modulo (Rest einer Division)
```

Vergleichsoperatoren

```
==   // gleich groß
!=   // nicht gleich groß
<    // kleiner als
>    // größer als
<=   // kleiner oder gleich
>=   // größer oder gleich
```

Logische Operatoren (Boolean)

```
&&   // logisches ‚und'
||   // logisches ‚oder'
!    // logisches ‚nicht'
```

Zusammengesetzte Operatoren

```
x++  // erhöht x um 1
x--  // vermindert x um 1
```

FUNKTIONEN

Pin Input/Output

Digital I/O – Pins 0-13 (sowie A0-A5)

```
PinMode (Pin, INPUT/OUTPUT/INPUT_PULLUP)  // Setzt den Pin-Modus
digitalRead (Pin)  // Lies den Pin-Status aus und speichert HIGH oder LOW
digitalWrite (Pin, HIGH/LOW)  // Schaltet Pin auf 5V (HIGH) oder 0V (LOW)
```

Analog Input – Pins A0 – A5

```
analogRead (Pin)   // Eingehende Spannung als Wert zwischen 0 und 1023
```

PWM Output – Pins 3 5 6 9 10 11

```
analogWrite (Pin, Wert) // Wert zwischen 0 und 255
```

Erweiterte I/O

```
tone (PWM-Pin, Frequenz)          // Frequenzangabe in Hertz
tone (PWM-Pin, Frequenz, Zeit)    // Ton mit Dauer in Millisekunden
noTone (Pin)                      // Stopp der Ausgabe einer tone-Funktion
```

Zeit

```
delay (Zeit)   // Verzögert den Programmablauf – Angabe in Millisekunden
millis ()      // Zählt die Zeit ab Programmbeginn in Millisekunden
```

Mathematik

```
max (x,y)   // Gibt den höheren Wert der Zahlen x und y zurück
min (x,y)   // Gibt den kleineren Wert der Zahlen x und y zurück
```

ARDUINO-BIBLIOTHEKEN

Bibliotheken werden über den Befehl **#include <Bibliothekname>** eingebunden

Serial // Kommunikation mit PC oder via RX/TX – Einbindung nicht nötig

```
begin (Geschwindigkeit)    // Start der Verbindung – Geschw. bis 115200
end ()                     // Stopp der Verbindung
print (Daten bzw. „Text“)  // fortlaufende Ausgabe auf seriellem Monitor
println (Daten bzw. „Text“) // Ausgabe auf  seriellem Monitor mit Absatz
```

Servo.h // Steuerung eines Servo-Motors

```
attach (Pin)      // Servo an den angegebenen Pin anbinden
write (Winkel)    // Servo auf den angegebenen Winkel fahren
read ()           // Servo-Stellung auslesen
```

Wire.h // I²C-Kommunikation über SDA und SCL

```
begin ()                          // Initialisierung des I²C-Busses als Master
requestFrom (Adresse, Anzahl)     // Daten vom Busteilnehmer anfordern
beginTransmission (Adresse)       // Datenübertragung an gewählte Adresse
write (Wert/String)               // Daten werden übermittelt
endTransmission ()                // Ende der Datenübertragung
```