Arduino Bastelgruppe OV U13

Die Software

Als erstes sollte man die Arduino Software ARDUINO IDE 1.8.1 von der URL https://www.arduino.cc/en/Main/Software herunterladen. Sie enthält einen Editor zur Engabe und Änderung von Programmen und einen Compiler, welcher den Code für den Arduino erzeugt. Das fertige Programm kann mit einem USB Kabel sofort in den Arduino geladen werden.
Bei allen Arduino Ausdrücken auf Gross- und Kleinschreibung und Zwischen räume achten, sonst gibt's Fehlermeldung! (z.B. unsigned long)
Während der Arbeit öffne man die Refernece Page, wo alle Befehle genau beschrieben sind.

1. Startet die Arduino Software
2. Klickt auf **Hilfe** in der Menuleiste
3. Klickt auf **Referenz**

Euer Webbrowser startet und stellt die in der Arduino Softwarepaket enthaltene Referenzseite dar.

Struktur eines Programms

Variablendeklaration
void setup() { }
void loop() { }

Ein Programm enthält mindestens zwei Grundroutinen. Dabei heisst **void**, dass die Routine kein Ergebnis liefert. Die leeren Klammern besagen, dass es keine Eingabwewerte gibt.

void setup() Hier erfolgen die Intialiserungen
void loop() Die loop läuft ewig ab. Nach jedem Durchlauf werden die Befehle oben fortgesetzt.
Die Befehle der Routinen stehen zwischen den geschw. Klammern { }, jeder Einzelbefehl endet mit ; (Semikolon).

Variable müssen deklariert werden. Vor setup dekalierte Variable sind überall verfügbar („globale Variable"), in einer Routine deklarierte Variable nur innerhalb der Routine („lokale Variable").

Variablentypen

byte	Byte Ganzzahl von 0 .. 255, belegt 1 Byte.
int	Integer Ganzzahl von -32768 .. 32767, belegt 2 Bytes.
unsigned long	Eine Zahl von 0 .. 4,294,967,295 (= 2^32 – 1), 4 Bytes.
int *Wert[Zahl]*	Integer Array mit *Zahl* Plätzen.

Einzelbits der Variablen können auf 0 oder 1 gesetzt werden:

bitSet (Variable, BitNr)	Setzt das Bit in der Variablen auf 1
bitClear (Variable, BitNr)	Setzt das Bit in der Variablen auf 0

Beispiele:
int *I*; | byte *Index*; | unsigned long *Startzeit*; | int *Daten[24]*;
int *Ton[]* = [440, 466, 494, 523, 554]; Voreingestelltes Array mit 5 Plätzen.
bitSet (*I,0*); Setzt Bit 0, das niedrigwertigste Bit von I auf 1.

Zahlen- oder Datenwerte

Zahlenwerte können in folgenden Formaten angegeben werden:

25	Dezimalzahl. KEINE führenden Nullen!! (z.B. 025)
B11010	Binärzahl, beginnt mit **B**. Nur 8 Bits möglich
0x7B	Hexadezimalzahl, beginnt mit **0x**.
HIGH, LOW	Pegel an digitalem Pin.

Ein- und Ausgänge

pinMode(PinNr, Funktion)

OUTPUT bedeutet digitaler Ausgang
INPUT bedeutet Eingang
INPUT_PULLUP aktiviert einen 20KOhm-Widerstand zur +Versorgung, der den offenen Eingang auf HIGH zieht.

Beispiele:

pinMode(8, OUTPUT);	Ausgang.
pinMode(6, INPUT_PULLUP);	Eingang mit 20KOhm-Pullup Widerstand.

Ansprechen der Ein- und Ausgänge

digitalWrite(PinNr, Wert)	legt den Pin auf den Wert LOW oder HIGH.
digitalRead(PinNr)	liefert den Zustand des Pins (LOW oder HIGH)
analogWrite(PinNr, Wert)	Am Pin wird ein PWM-Rechtecksignal abgegeben mit Tastverhältnis Wert/256. Wert = 0 .. 255
analogRead(PinNr)	setzt die Spannung am Pin im Bereich 0 .. 5V in eine Zahl von 0 ..1023 um. Dauert ca. 100 µs.

Ein SonderPin Nr. ist LED_BUILTIN. An ihm hängt die LED auf dem Board.
Nicht alle Pins können analogRead und analogWrite. Ausprobieren.

Beispiele:

digitalWrite(7, LOW);	Setzt Pin 7 auf LOW, Pin 7 muss Ausgang sein!
AnalogWrite(8,128);	Pin 8 gibt eine Rechteckwelle mit einem Gleichspannungsmittelwert von 2,5 V ab.
W = analogRead(9);	W = 0 für 0 V am Pin und 1023 für 5 V am Pin und linear dazwischen. W muss Variablentyp int sein!

Töne

tone(PinNr, *Frequenz*);	Gibt am Pin eine Rechteckwelle der angegebenen *Frequenz* (Typ int) aus. Bereich: 40-65535 Hz.
notone(PinNr);	Beendet Ton an Pin.

Es kann immer nur an einem Pin ein Ton ausgegeben werden. Vor Starten eines „neuen" Tons muss der „alte" Ton mit notone beendet werden.

Zeitfunktionen

delay(*Dauer*)	Wartet *Dauer* Millisekunden.
DelayMicroseconds(*Zeit*)	Wartet *Zeit* Mikrosekunden.
millis()	Liefert die Zeit in ms seit letztem Reset. Variable muss vom Typ unsigned long sein.
Micros()	Liefert die Zeit in µs seit letztem Reset. Variable muss vom Typ unsigned long sein. Läuft nach 70 Minuten über auf 0

Abfrage

if (*Wertevergleich*) { } ; Wenn *Wertevergleich* zutrifft, wird der Befehl oder die Befehle zwischen { } ausgeführt.

Wertevergleiche

==	Gleich (NICHT „=“)	!=	Ungleich
<	Kleiner	>	Grösser
<=	Kleiner oder gleich	>=	Grösser oder gleich

Beispiele:

```
if (Wert == 15)  digitalWrite(7, LOW);
```

Inkrement und Dekrement

In Schleifen werden oft Indizes um 1 erhöht oder erniedrigt. Dafüt gibt es Sonderoperatoren:

Variable++	Die *Variable* wird um 1 erhöht, es wird aber der „alte“ Wert als Resultat geliefert.
Variable--	Entsprechend mit Erniedrigung um 1.

Schleifen

Das sind mehrfach durchlaufene Programmteile. Es gibt folgende Mechanismen:

for (I = 0 ; I <12; I++) { Programmteil }	Das Programmteil zwischen den { } wird mit den Werten für I von 0 bis 11 durchlaufen.
while (Abfrage) { Programmteil }	Das Programmteil zwischen den { } wird durchlaufen, solange die Abfrage zutrifft, eventuell also auch garnicht.
do { Programmteil } while (Abfrage);	Das Programmteil zwischen den { } wird durchlaufen, solange die Abfrage zutrifft. Da die Abfrage am Ende erfolgt, erfolgt mindestens 1 Durchlauf.

Beispiel:

3mal das gleiche Funktion mit den 3 Funktionen:

```
for ( I = 0 ; I <12; I++)  { Programmteil }          // Kein ; folgt

I=0;  while(I<12)  { Programmteil  I++ }             // Kein ; folgt

I=0;  do  { Programmteil  I++ }  while(I<12);        // ; muss folgen
```

Programmteil könnte sein (siehe Programm Tonleiter):

```
Frequenz=Ton[I]; tone(12,Frequenz);delay(250);notone(12);
```

Musterprogramm Morseausgabe DL0RA

```
// MorseDL0RA.ino  HHC 17.7.17
// Ausgang: Pin 13
// Datei:   MorseDL0RA.txt

  int Punkt ;                              // Variablendeklaration
  int Strich ;                             // Alles Integer Variable
  int Frequenz = 1000 ;                    // Wertzuweisung erlaubt

void setup()
{
  pinMode (13, OUTPUT);                    // Pin13 = Ausgabe
  Punkt = 100                              // Punktlänge 100 ms
  Strich = 3 * Punkt ;                     // Strich = 3 * Punkt
}
void dit()                                 // Ausgabe Punkt
{
  tone(13, Frequenz);                      // Ton ein
  delay(Punkt);                            // Warte Punktlänge
  noTone(13) ;                             // Ton aus
  delay (Punkt) ;                          // Warte Punktlänge
}
void dah()                                 // Ausgabe Strich
{
  tone(13, Frequenz);                      // Ton ein
  delay(Strich);                           // Warte Strichlänge
  noTone(13) ;                             // Ton aus
  delay (Punkt) ;                          // Warte Punktlänge
}
void Zwi()                                 // Ausgabe Zwischenraum
{   delay (Strich) ; }                     // Warte Strichlänge

void loop()                                // Hier wird DL0RA ausgegeben
{
    dah(); dit(); dit();                   // D
    Zwi();                                 // Zwischenraum
    dit(); dah(); dit(); dit();            // L
    Zwi();
    dah(); dah(); dah(); dah(); dah();     // 0
    Zwi () ;
    dit(); dah(); dit();                   // R
    Zwi () ;
    dit(); dah();                          // A
    Zwi () ;
    delay (1000) ;                         // Warte 1 sec.
}                                          // loop wird endlos wiederholt
```

Musterprogramm Töne einer Oktave

```
  int Spei[] = {440, 466, 494, 523, 554, 587, 622, 659, 698, 740,
                784, 830, 440} ;        // Töne A4 bis A5
  int Frequenz ;
  int I;

void setup()
{  pinMode(13,OUTPUT);  }              // Ausgabe an Pin 13

void loop()
{
  for ( I = 0 ; I <12; I++)            // Schleife I von 0 bis 11
  {
    Frequenz = Spei[I];                // Frequenz mit Index I
    tone (13,Frequenz);                // 500 ms lang ausgeben
    delay (500);
  }
  noTone(13);                          // Ruhe!
  Delay(1000);                         // 1 sec.warten
}
```

Der Compiler hat feste Töne aus der Musik eingebaut:
sioehe dazu im Tutorial bei tone(), Bragg Hagman's notes