JAVA for Enlightened

1. Einstellungen im Programmkopf

   1. Bibliotheken

      Wollt Ihr bestimmte Objekte wie z. B. Motoren benutzen, so müssen geeignete Bibliotheken eingebunden werden

      #include <Servo.h>

   2. Variablen deklarieren

      Wollt ihr eine neue Variable erzeugen (oder in Informatikersprache: deklarieren), schreibt ihr erst den Typ auf, dann den Namen der Variablen und schließt die Anweisung mit einem Semikolon ab:

      Der Arduino muss vorher wissen, ob ihr zum Beispiel Zahlen oder Wörter speichern wollt. So sind ganze Zahlen Variablen vom Typ **int** (engl. *Kurzform für ganze Zahl*).

      z.B.: <typ> <variablenname>; (Texte in <> sind **Platzhalter**, hier muss später euer eigener Text hin)
      int ergebnis;

      Diese Zeile erzeugt eine Variable für ganze Zahlen. Der Variablenname ist „ergebnis“. Java-Arduino unterscheidet zwischen Groß-und Kleinschreibung! „Ergebnis“ ist also nicht gleich „ergebnis“!

      Ihr könnt dieser Variabeln auch direkt beim Start des Programmes einen Wert zuweisen:

      z.B.: **int ergebnis = 0;**

      int<pinname> = <Pin-Anschlussnumer auf dem Board> ordnet dem Anschluss einen Namen zu.
      Servo <servoname> reserviert im Programm einen Namen für den Motor.

2. Anweisungen, die beim Start einmalig ausgeführt werden in der **void setup()** erklärt.

   z.B. :

   pinmode(<Pinname>**,**<Pin-typ>**);** erklärt, ob ein Pin für den Pintyp **INPUT** oder **OUTPUT** gebraucht wird;
   Serial.begin(9600); startet den Monitorausdruck
   <Servo-name>**.attach(**<Pinname>**);** verbindet die Motorsteuerung mit dem entsprechenden Pin.

3. Anweisungen, die wiederholt ausgeführt werden in der **void loop()** erklärt.

   digitalWrite(<Pinname>**,**<Zustand>**);** schaltet die Spannung auf dem Pin: Zustand ein (**HIGH**) bzw. aus (**LOW**);
   Serial.print("Sensorwert: "); druckt das Wort „Sensorwert: „ ohne Zeilenumbruch
   Serial.println(sensorWert); druckt den Inhalt der Variablen Sensorwert mit Zeilenumbruch
   <Servo-name>**.write(Grad);** stellt den Motor auf die Richtung der Gradzahl
   delay(5000); wartet an dieser Stelle 5000 ms
   <Variablenname>=**analogRead(**<Pinname>**);** weist einer zuvor als Int erklärten Variablen den auf dem als Output erklärten Pin gelesenen Wert zu.

   <Variablenname>=<Variablenname>+1; erhöht den Wert der Variablen um 1.

4. Beispiele

```
int ledPin = 8;  //Hier ist die LED an Pin 8 angeschlossen

void setup(){
  pinMode(ledPin,OUTPUT); //Pin 8 soll Ausgang sein
}

void loop(){
  digitalWrite(ledPin,HIGH);  //Einschalten
}
```

5. Bedingungen und bedingte Anweisungen

   1. Bedingte Anweisungen

      if (<Bedingung>)
      {
       <Anweisung1>;
       <Anweisung2>;
      …
       }

   2. <Bedingungen>

      Vergleiche/Abfragen, deren Ergebnis wahr oder falsch sein können. Es können auch elektrische Spannungszustände abgefragt werden.

      if (**digitalRead(**<Pin-Name>**)==**<Zustand>**): untersucht** die Spannung auf dem Pin:
       Wenn der Zustand ein (**HIGH**) ist, wird der folgende Anweisungsblock ausgeführt
       bzw. wenn er aus (**LOW**) ist, wird der folgende Anweisungsblock nicht ausgeführt;
      Achtung: Der PinMode dieses Pins muss dafür vom Typ INPUT sein.

      Vergleichende Operatoren: ==; <, >;

      Beispiel:

      if (altWert = sensorWert)
      {
       richtung=1;
       Serial.println("konstant");
       }

   3. Bedingte Anweisungen mit Alternative

      if (<Bedingung>)
      {
       <Anweisung1>;
       <Anweisung2>;
      …
       }
      else
      {
       <AAnweisung1>;
       <AAnweisung2>;
      …
       }

      Wenn die Bedingung nicht wahr ist, wird der alternative Anweisungsblock ausgeführt.

      Quellen: Projekt Enlightened, der RWTH Aachen, http://schuelerlabor.informatik.rwth-aachen.de/materialien, *Grafiken: InfoSphere*