Arduino: Einfacher Einstieg Elektronik und Mikrocontroller

Frank Bartels 2014/2016 (frank@toppoint.de)

- Mikrocontroller Hardware

  * Mikrocontroller = kompletter Computer auf einem Chip
    Hier: ATmega328 - 32KB Flash, 2KB RAM, 16MHz
                      1 serielle Schnittstelle + diverse I/O Pins

  * Prorammierung über spezielle Schnittstelle von außen
    Hier: über serielle Schnittstelle via Arduino-Bootloader
          (alternative ICSP falls Bootloader defekt)

  * I/O-Pins (Input/Output)

    # Digitaler Eingang / Ausgang (0:0V, 1:5V)
      Hier: D0..D13 (D13 verbunden mit on-Board LED)

    # Analoger Eingang (0V..5V meßbar)
      Hier: 8 Pins A0...A7

    # Digitaler Ausgang als PWM Ausgang (Pulsweitenmoduliertes Rechtecksignal)
      20%=1/5: -____-____-____ 80%=4/5: ----_----_----_
      Hier: 6 Pins: D 3 D5 D6 D9 D10 D11
            Werte 0:0% bis 255:100%

    # Spezialfunktionen für einige Pins,
      z.B. serielle Schnittstelle (Zeichen senden/empfangen über einen Draht)
           (TX(senden)(D1) / RX(empfangen)(D0))

- Programmierung mit Arduino-Bibliothek

  * Organisatorisches:

    Verbindung Computer[USB] <--1--> USB-Seriell-Wandler <--2--> Arduino z.B. ProMini
    Verbindung 2 besteht aus 4-5 Drähten: Masse, (5V), RX, TX, DTR
      (5V weglassen, wenn der Arduino anders mit Strom versorgt wird)

    Software: Linux/Debian-Typ: 1. apt-get install arduino
      2. Dann ggf. noch den User in die Gruppe "dialout" tun und neustarten (ausloggen)
      3. (nicht notwendig, aber besser, das jetzt auch zu tun: von **arduino.org**
         herunterladen, hier auch gute Alternative ohne Java: Version 0.0.5alpha)

    Wenn fertig, "arduino" starten und immer folgende Schritte beachten:
    1. Tools / Serieller Port -> den richtigen aussuchen
         Achtung: die Einstellung kann manchmal verloren gehen!
                  bitte gelegentlich nachprüfen, wenn was nicht geht.
    2. Tools / Board -> Arduino Pro Mini (5V 16MHz) mit ATmega328
         Das richtige Board auswählen
    2.b. Bei manchen Arduino-Umgebungen:
         ggf. noch den Chip auf dem Board auswählen idR Atmega328
    3. Programm eingeben oder Beispielprogramm laden (Datei/Beispiele/...)
         Zur Programmierung siehe weitere Kapitel
         (Programm speichern (speichern unter) nicht vergessen)
    4. Die Hardware aufbauen natürlich, denn die Verdrahtungen müssen zum
         Programm passen (richtige Zuordnung der Pins).
    5. Strg-U oder Knopf mit Pfeil nach rechts klicken
         dies kompiliert (Ctrl-R/Häkchen ist nur kompilieren)
         und lädt auch gleich hoch (also auf das Arduino-Board)
         Achtung: Falls man was falsch eingegeben hat, erscheint
                  eine Fehlermeldung unter dem Programmtext.
                  Meist interressiert einen erst mal der erste Fehler,
                  also muß man das kleine Fensterchen unten hochscrollen.
                  Man kann das Arduino-Fenster mit der Maus vergrößern!
    6. Ggf. serielle Konsole (Tools/Serial Monitor) starten (Taste Ctrl-Shift-M).
         Da muß man ggf. die Baudrate anpassen (siehe Parameter beim setup() s.u.).

```
* Aufbau eines Arduino-Programms (Programmiersprache: C (bzw. auch C++ möglich))

  // 1. Ggf. Bibliotheken einbinden (Menü: Sketch/Library importieren)
  #include <was_ich_will.h>

  // 2. Defines (Konstanten) und globale Variablen deklarieren
  #define BspPin 2  // Ersetzt im Folgenden alle "BspPin" durch "2"
  int counter=0;    // counter ist nun eine Variable für ganze Zahlen (-32K..32K)
  #define Pins 4
  int led_pin[Pins]={3,5,6,9}; // vier PWM Pins (zB led_pin[0] ist 3)

  // 3. Setup (Initialisierung)
  void setup()
  { // Meine setup-Befehle hierher, wird einmal beiom Start(Reset) ausgeführt
  }

  // 4. Loop (gewünschte Aktivität in Schleife programmieren)
  void loop()
  { // Befehle hierher, werden immer wiederholt, bis Strom weg oder Reset
  }

* Digital Output
          #define BspPin 2            // oder D2 schreiben
  setup:  pinMode(BspPin, OUTPUT);
  use:    digitalWrite(BspPin, LOW);  // 0V (0=LOW)
          digitalWrite(BspPin, HIGH); // 5V (1=HIGH)

* Digital Input
          #define BspPin 2            // oder D2
  setup:  pinMode(BspPin, INPUT);
  use:    int val = digitalRead(BspPin);
          if (val==HIGH) ...          // Falls 1 (ca. >2V) ...
          if (val) ...                // dasselbe wie oben
          if (val==LOW) ...           // Falls 0 (ca. <1V) ...
          if (!val) ...               // dasselbe wie oben
   - Spezialfunktion: digital input with pull-up resistor -
  info:   dann reicht es zum Anschluß eines Tasters, den Taster
          anzuschließen (gegen Masse; pull up gegen 5V intern)
          #define BspPin 2            // oder D2
  setup:  pinMode(BspPin, INPUT);
          digitalWrite(BspPin, HIGH); // Ja, wirklich write auf input pin
                                      // aktiviert 50KOhm Widerstand gegen 5V
  Alternatives setup (seit Arduino 1.0.1 idR auch möglich) in einer Zeile:
          pinMode(BspPin, INPUT_PULLUP);
  use:    (genau wie beim normalen digitalen Eingang)
          (LOW=0='Taster gedrückt'  HIGH=1='Taster NICHT gedrückt')

* PWM Output (quasi analog output)
  info:   980Hz Signal, Angabe in 256 Stufen (0..255)
          #define BspPin 3 // Pin mit "PWM" Beschriftung
  setup:  pinMode(BspPin, OUTPUT);
  use:    analogWrite(BspPin, 127); // 127/255 ca.50% __--__--__--_
          analogWrite(BspPin, 1);   // 1/255  ca.0.4% __|___|___|__

* Analog Input
          #define BspPin A0
  setup:  // (keine Vorbereitung des Pins erforderlich)
  use:    int val = analogRead(BspPin); // liefert 0(0V)..1023(5V)
          Serial.println(val);          //
          Serial.println(5.0*(val/1023.0));  // in V (0.0 .. 5.0)
          analogWrite(AndererBspPin, val/4);  // Ausgabe auf PWM Pin

* Serielle Schnittstelle (Ausgaben und Eingaben auf dem Terminal)
  setup:  Serial.begin(9600);
  use:    Serial.print(1234);  // int (Zahl -32K..+32K), long (-2M..2M)
          Serial.print(1.234); // float (Fließkommazahl), double (genauer)
          Serial.print("Hello ");   // (char *) (Text)
          Serial.println("World!"); // Text und am Ende ein Zeilenvorschub

* Nützliche Helfer
          delay(100);           // warte 100 ms (1ms = 1/1000 s)
          delayMicroseconds(2); // warte 2 us (1Mikrosekunde = 1/1000000 s)
          tone(pin, frequency, duration);  // frequency 31..[Hz], duration [ms]
          obiges entspricht: tone(pin, frequency); delay(duration); noTone(pin);
```

- Teile, allgemeine

    * Arduino ProMini (das Arduino-Board, siehe Plan!)
    * Falls Board = Arduino ProMini: USB-Seriell-Wandler (als Programmieradapter)

    * Z.B. die unten erwähnten Teile verdrahten, siehe Anleitung
    * Breadboard (großes Steckbrett zum Verdrahten, siehe Plan!)
    * Diverse Kabel verschiedener Art und Länge zum verdrahten
      Enden sind Pins ("männlich") oder Buchsen ("weiblich")
    * 9V Clip (für Batterien)
      Siehe Arduino-Board-Beschreibung zum Anschluß!
    * Bitte Digital-Level beachten:
        Logisch 0 = LOW = GND(Masse)   = 0V
        Logisch 1 = HIGH = VCC(Betriebsspannung) = 5V ((ggf. 3,3V !))

- Teile für Output (Beispiele)

    * LEDs in rot/grün/gelb/... (kurzer Draht/ bzw. Kante ist Masse/Minuspol)
      Schaltung: GND------(-)LED(+)-----R(470)------Arduino-Digital-Ausgangs-Pin
      (Der Vorwiderstand „R“ ist erforderlich! z.B. 470 Ohm ist für eine LED geeignet)
    * (Piezo-)Lautsprecher
      Schaltung: Dieser kann beim Piezo direkt an einen Output Pin angeschlossen werden
    * Servomotor (*Schaltung*: Pins: GND, 5V(am besten eigene Stromversorgung), Daten)
       (Bibliothek dafür auf http://arduino.cc suchen, falls nicht schon vorhanden)

- Teile für Input (Beispiele)

    * Potis (Dreh-Potentiometer, variabler Widerstand z.B. 10KOhm)
      Schaltung: 0V------Poti------5V
      ==========   ^----Arduino-Analog-Eingangs-Pin, z.B. A0
                Der mittlere Abgriff zum Arduino-Pin hat Spannung zwischen 0V und 5V
    * (Mikro)taster (*Schaltung*: Ein Pin auf GND, der andere an Arduino Pin, zB D2)

    * LDR: Helligkeitssensor (lichtempfindlicher Widerstand) - bräunlich, Glaskörper
      z.B. 1M Ohm bei sehr dunkel, 10K Ohm bei normalem licht, das zum lesen reicht
    * NTC: Temperatursensor analog (sehr kleines gnubbeliges Bauteil mit 2 Drähten)
      10KOhm bei ca.25°C
      Schaltung: GND------NTC(oderLDR)-----+-----R(10KOhm)------5V
      ==========   ^--Arduino-Analog-Eingangs-Pin, z.B. A0
    * Digitaler Temperatur und Feuchte Sensor DHT11 (blaues Kästchen, 4 Anschlüsse)

    * R: Widerstände 10KOhm(?) zB für LDR, NTC   2normale

Beispiele:

```
---------------------------------------------------------------------------------------
const int OnBoardLED = 13;
void setup()                                // Das Programm entspricht Beispiele/Basics/Blink
{ pinMode(OnBoardLED, OUTPUT);
}                                           // Das Einfachste: Eine LED blinken lassen:
void loop()                                 // Folgenden Ablauf ständig Wiederholen:
{ digitalWrite(OnBoardLED, 1);              // LED anmachen (auf 1 oder HIGH setzen)
  delay(100);                               // 100ms (=0,1s) warten
  digitalWrite(OnBoardLED, 0);              // LED ausmachen (auf 0 oder LOW setzen)
  delay(900);                               // 900ms (=0,9s) warten
}
---------------------------------------------------------------------------------------
const int PinLED0 = 5;    // Ein PWM Pin
const int PinLED1 = 6;    // Ein PWM Pin
const int PinPoti = A0;   // Ein Analogeingangs-Pin
void setup()
{ pinMode(PinLED0, OUTPUT);   // PinLED0 sei Ausgang, Bei Analogeingang hingegen nicht nötig
  pinMode(PinLED1, OUTPUT);
}
void loop()
{ int potipos = analogRead(PinPoti)/4; // Poti Position (in Bereich 0..255 gebracht durch /4)
  analogWrite(PinLED0, 255-potipos);   // Wird heller auf der einen Seite des Potis
  analogWrite(PinLED1, potipos);       // Wird heller auf der anderen Seite des Potis
}
---------------------------------------------------------------------------------------
const int LED_N = 4;
const int PinLED[LED_N]={3,5,6,9};    // hier vier PWM Pins ausgewählt
void setup()
{ for (int i=0; i<LED_N; i++) pinMode(PinLED[i], OUTPUT);      // i geht von 0 bis LED_N-1
}
int pos=0;  // Position der aktuellen LED, startet bei der ersten (0); Positionen: 0,1,2,3
void loop()                            // Lauflicht: 1ooo → o1oo → oo1o → ooo1 → Zurück
{ digitalWrite(PinLED[pos],1); delay(500); digitalWrite(PinLED[pos],0); // 0.5s aufleuchten
  pos=pos+1;  if (pos>=LED_N) pos=0;  // weiter; wenn zu weit, dann auf Anfang zurück
}
```