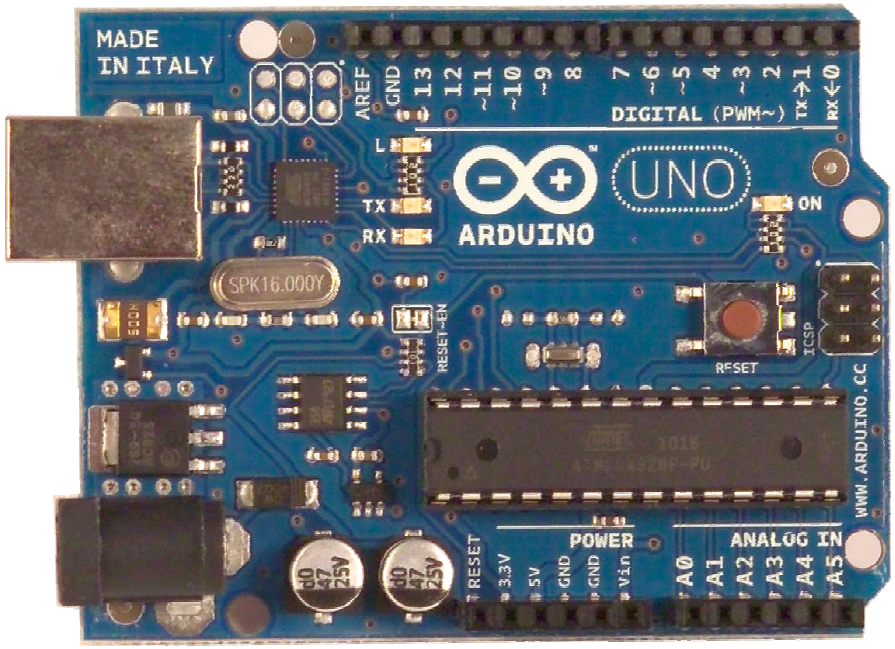
Arduino

Anleitung



Inhaltsverzeichnis

Vorbemerkung

Zum Aufbau dieser Anleitung:

- Nach einer kurzen Vorstellung der Hard- und Software werden die neuen elektronischen Bauelemente und Befehle schrittweise eingeführt.
  So besteht der „praktische" Teil immer aus dem Aufbau:
  - Neue Bauelemente
  - Neue Befehle
  - Aufgabe
  - „weitere" Aufgaben

- Das Erlernen der Programmiersprache stützt sich dabei auf die zahlreichen Beispielprogramme (Sketches), die bereits hinterlegt sind.
  Danach soll aktiv mit den neuen Befehlen umgegangen werden, um weitere Aufgaben zu lösen. Dabei werden früher erlernte Befehle immer wieder aufgegriffen.

- Nach diesem Kurs sollten die Schülerinnen und Schüler in der Lage sein, selbständig ein Projekt mit dem Arduino durchzuführen.

Links

- **Arduino Homepage:**
  http://www.arduino.cc/

- **Tutorials:**
  http://www.kriwanek.de/arduino/wie-beginnen.html
  http://www.arduino-tutorial.de/einfuhrung/
  http://popovic.info/html/arduino/arduinoUno_1.html
  http://www.netzmafia.de/skripten/hardware/Arduino/Arduino_Programmierhandbuch.pdf (Arduino Handbuch)

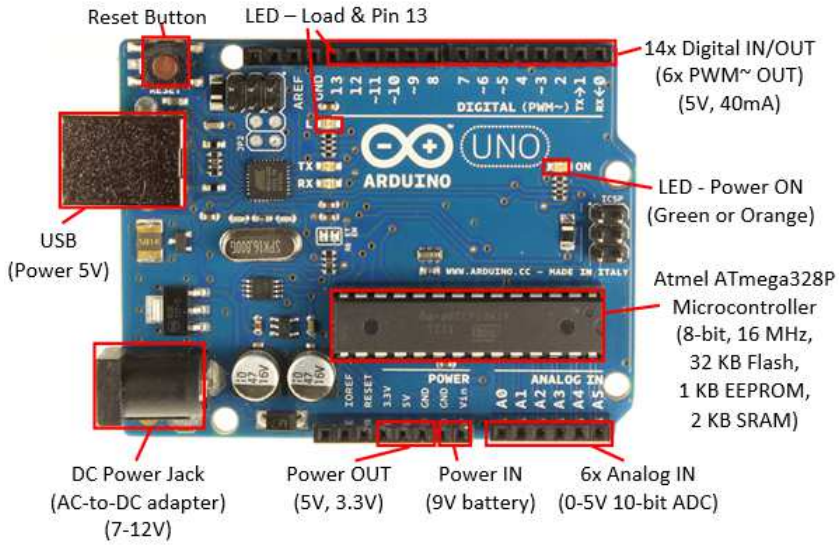
- **Video-Tutorials:**
  http://www.arduino-tutorial.de/category/videoworkshop/
  http://mr.intermediadesign.de/tag/arduino/
  http://maxtechtv.de/category/arduino/beginner/

- **Projekte:**
  http://www.maxpie.de/Projekt_arduino.htm

TEIL 1:
Hardware und Software

Aufbau Arduinoboard

Die Hardware



Quelle: https://www3.ntu.edu.sg/home/ehchua/programming/arduino/images/ArduinoUno.png

Grundprinzip eines Mikrocontrollers

Um Leuchtdioden leuchten zu lassen oder Motoren zu steuern, erzeugt ein Mikrocontroller an seinen **Pins** (Anschlüssen) verschiedene Spannungen.

- **Digitale Pins:**
  Sie haben entweder den Zustand HIGH oder LOW. Es liegen dann entweder 0V oder 5V an.

- **Analoge Pins:**
  Ihr Spannungswert kann **zwischen** 0V und 5V liegen.

Über die Pins können auch Spannungswerte eingelesen werden. Damit kann man z.B. überprüfen, ob ein Schalter geöffnet ist oder geschlossen, ob viel Licht auf eine Photozelle fällt oder wenig.

Auch hier gibt es die Unterscheidung zwischen digitalen Pins (Entweder HIGH oder LOW) und den analogen Pins (Spannungswerte zwischen 0V und 5V).

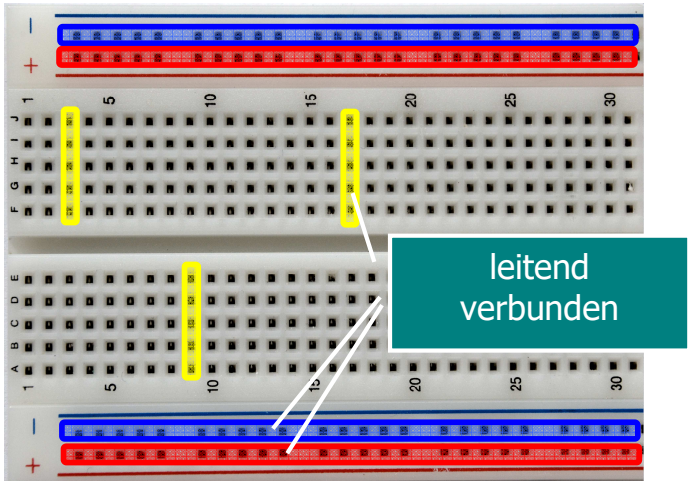
Steckplatine

Auf der Steckplatine werden die Leuchtdioden, Widerstände und weitere elektronische Bauteile gesteckt und so miteinander verbunden.

Dabei gilt:

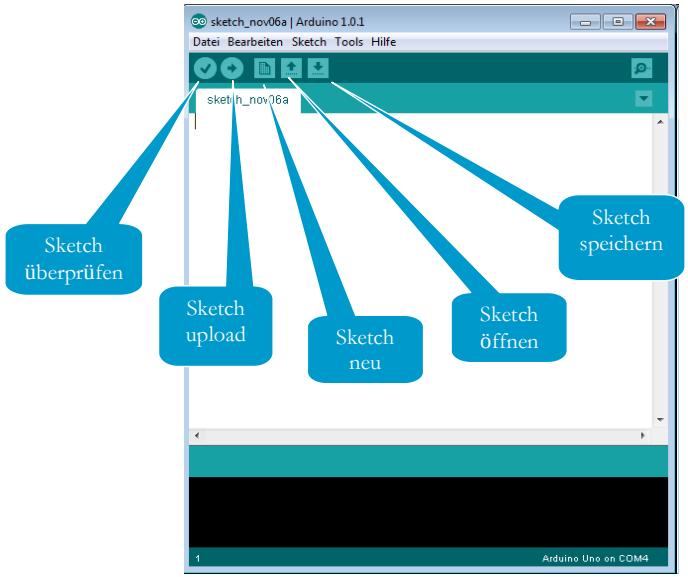
- Die seitlichen Kontakte (blau bzw. rot umrandet) sind über die ganze Länge leitend miteinander verbunden.
  Sie dienen der „Spannungsversorgung" der Schaltung.

- Zusätzlich sind die 5er-Reihen (gelb umrandet) leitend miteinander verbunden.

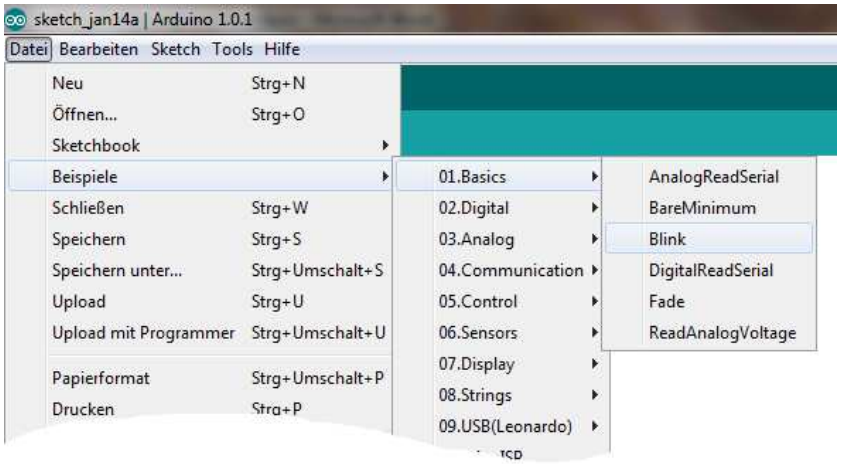


Editor

***Die* Software**



Beispielprogramme (Sketches)



Struktur eines Sketchs

Ein Programm wird allgemein mit „**Sketch**" bezeichnet.

Alle Sketches haben den gleichen Aufbau:

```
void setup()
{
  anweisungen;
}
```

Dieser Programmteil wird **einmal** durchlaufen!

(Setzen von Pinmode oder Start der seriellen Kommunikation)

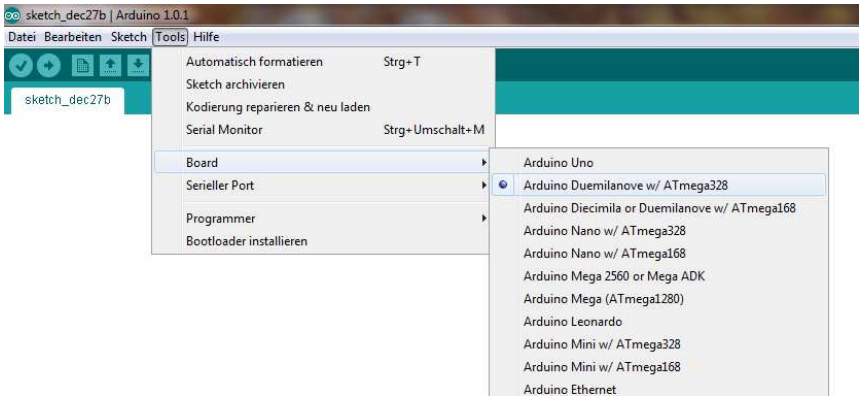
```
void loop()
{
  anweisungen;
}
```

Dieser Programmteil wird **unendlich oft** durchlaufen!

Hier befindet sich das eigentliche Programm,

Einstellungen

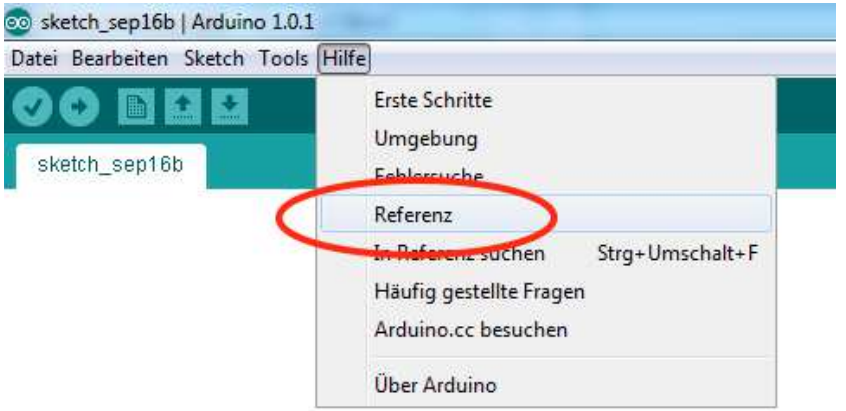
Auswahl des Arduino-Boards



Auswahl des seriellen Ports:

Normalerweise wird der Arduino über den seriellen Port mit der höchsten Nummer verbunden. Ansonsten bitte im Gerätemanager (Systemsteuerung) nachschauen

Hilfe



In dem Untermenü „Hilfe" wird man zu vielen hilfreichen Internetseiten geleitet.

So bekommt man unter „Refenrenz" eine Liste aller Befehle, die man beim Programmieren verwenden kann, mit Erklärung.

TEIL 2:
Grundkenntnisse in der Programmierung

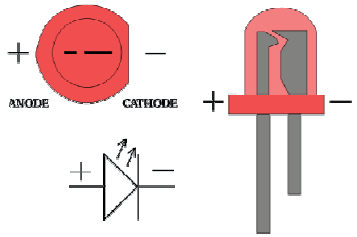
Blinkende LED

Neue Bauelemente: LED

Neue Befehle: pinMode, digitalWrite, delay

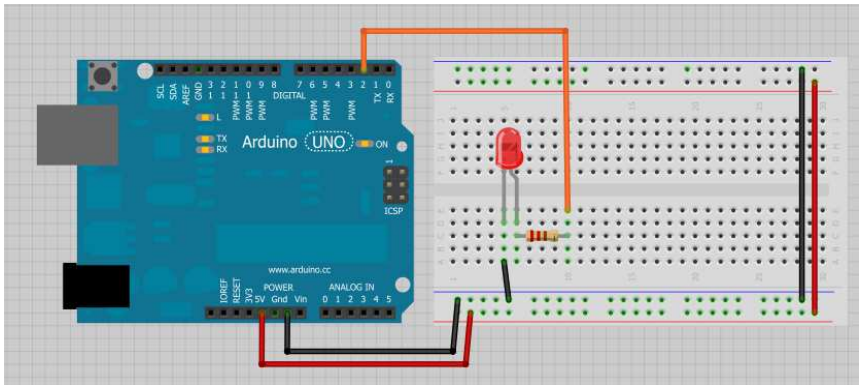
Ziel: Eine LED auf der Steckplatine blinken lassen.

INFO: LED



Aufgabe A1.1:

- Baue die Schaltung (s. Abb.) auf.
  Verwende einen Vorwiderstand von R = 220 Ω.
- Übernehme den Sketch (A011_Blink) in den Editor.
- Speichere den Sketch unter „A011_Blink" ab.
- Übertrage den Sketch auf den Arduino.
- Beobachte die eingebaute LED.



Aufgabe A1.2 :

- Ändere den Sketch „A011_Blink" so ab, dass die LED 3 Sekunden lang leuchtet und 1 Sekunde lang dunkel bleibt.
- Speichere den neuen Sketch im Ordner „Sketches"
  unter „A012_Blink2_(xx)" ab. xx = Dein Name.

Zum Ausprobieren:

- Auf dem Arduinoboard befindet sich an Pin 13 eine eingebaute LED (inklusive Vorwiderstand). Stecke eine LED direkt in Pin13 und den benachbarten GND-Eingang. Achte auf die Polung der LED und ändere den Sketch entsprechend ab.

Kapitel 1

A011_Blink

```
void setup() {
  pinMode(2, OUTPUT);       //Pin2 wird als
}                           //Ausgabe-Pin deklariert

void loop() {
  digitalWrite(2, HIGH);    //LED an Pin2 leuchtet auf
  delay(1000);              //1000 Milisekunden Pause
  digitalWrite(2, LOW);     //LED an Pin2 erlischt
  delay(1000);              //1000 Milisekunden Pause
}
```

Neue Befehle:

- **pinMode(2, OUTPUT)**

  Pin2 als **OUTPUT** deklariert. D.h. er kann die Werte HIGH (5V) und LOW (0V) annehmen und damit eine Leuchtdiode ansteuern.

- **digitalWrite(2, HIGH)**
  Pin2 wird **HIGH** (5V) gesetzt.

  digitalWrite(2, LOW)
  Pin2 wird **LOW** (0V) gesetzt.

- **delay(1000)**
  Bis zur Verarbeitung des nächsten Befehls wartet der Arduino 1000 ms.

Kommentare:

- Einzeilige Kommentare fangen mit // an.
- Blockkommentare fangen mit /* an und enden mit */ und umfassen mehrere Zeilen.

Einrücken:

- Durch „Einrücken" in den Befehlszeilen wird der Sketch übersichtlicher gestaltet. So wird z.B. jede Zeile die zu „void setup" gehört um 2 Zeichen eingerückt.

Lauflicht/Ampel

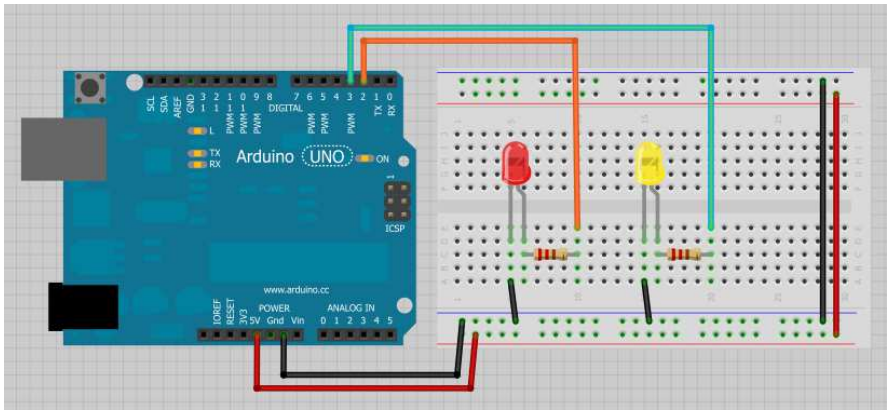
Neue Bauelemente: -

Neue Befehle: -

Ziel: Mehrere LEDs blinken lassen.

Aufgabe A2.1 :

- Ergänze die Schaltung aus Kapitel 1 um eine gelbe LED an Pin3.
- Übernimm den Sketch „A021_Blink_2LED".
- Übertrage den Sketch auf den Arduino.



Aufgabe A2.2:

- Ändere den Sketch so ab, dass die beiden LEDs abwechselnd blinken.
- Ändere auch die Blinkfrequenz.

Kapitel 2

A021_Blink_2LED

```
void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
}

void loop() {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  delay(1000);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  delay(1000);
}
```

Die Pins 2 und 3 werden als OUTPUT-Pins festgelegt.

Die LEDs werden im Sketch nacheinander angeschalten.

Da die Befehle sehr schnell nacheinander ausgeführt werden, nimmt das Auge keine zeitliche Verzögerung war.

Aufgabe A2.3: Lauflicht

- Baue ein Lauflicht aus einer roten, gelben und grünen LED auf, die nacheinander je eine halbe Sekunde aufleuchten sollen.
- Schreibe den Sketch.
- Speichere den Sketch unter dem Namen „A023_Lauflicht_(xx)" ab.
- Zeichne den Schaltplan.

Aufgabe A2.4: Ampel

- Verwende die rote, gelbe und grüne LED aus Aufgabe A2.3.
- Schreibe einen Sketch so, dass die LEDs entsprechend einer Verkehrsampel an und ausgehen.
  (Die Ampel soll mit „Grün" starten, auf „Rot" gehen und anschließend wieder auf „Grün").
- Speichere den Sketch unter dem Namen „A024_Ampel_(xx)" ab.

Variablen

Neue Bauelemente: -

Neue Befehle: int

Ziel: Verwendung von Variablen.

Eine Variable ist eine frei wählbare Bezeichnung (**Achtung: keine Umlaute!**), die mit einem Wert belegt werden kann:

```
int ledRot = 2;
```

In diesem Beispiel wird die Variable „**ledRot**“ als **Integer**-Variable definiert und ihr wird gleichzeitig der **Wert 2** zugeordnet. An jeder Stelle des Sketchs hat diese Variable nun den Wert 2.

Bemerkung: Je nachdem, welchen Wertebereich die Variable umfassen soll, muss ihr Datentyp entsprechend festgelegt werden (siehe rechte Seite).
Je „kleiner“ der Datentyp, desto weniger Speicherplatz wird benötigt.

A011_Blink

```
void setup() {
  pinMode(2, OUTPUT);
}

void loop() {
  digitalWrite(2, HIGH);
  delay(1000);
  digitalWrite(2, LOW);
  delay(1000);
}
```

A031_Blink_Variable

```
int ledRot = 2;

void setup() {
  pinMode(ledRot, OUTPUT);
}

void loop() {
  digitalWrite(ledRot, HIGH);
  delay(1000);
  digitalWrite(ledRot, LOW);
  delay(1000);
}
```

Die Verwendung von Variablen hat folgende Vorteile:

- Wird eine LED an einem anderen Pin angeschlossen, muss nur eine Zeile geändert werden.
- Der Sketch ist einfacher zu lesen und zu verstehen, vor allem wenn mehrere LEDs oder andere Bauteile (Motoren…) verwendet werden.

Kapitel 3

Info Datentypen:

- **byte**

  Byte speichert einen 8-bit numerischen, ganzzahligen Wert ohne Dezimalkomma. Der Wert kann zwischen 0 und 255 sein.

  ```
  byte y = 180; // deklariert y als einen 'byte' Datentyp
  ```

- **int**

  Integer ist der meist verwendete Datentyp für die Speicherung von ganzzahligen Werten ohne Dezimalkomma. Sein Wert hat 16 Bit und reicht von -32.767 bis 32.768.

  ```
  int x = 1500; // deklariert y als einen 'integer' Datentyp
  ```

  Bemerkung: Integer Variablen werden bei Überschreiten der Limits 'überrollen'. Zum Beispiel wenn x = 32767 und eine Anweisung addiert 1 zu x, x = x + 1 oder x++, wird 'x' dabei 'überrollen' und den Wert -32,768 annehmen.

- **long**

  Datentyp für lange Integer mit erweiterer Größe, ohne Dezimalkomma, gespeichert in einem 32-bit Wert in einem Spektrum von -2,147,483,648 bis 2,147,483,647.

  ```
  long z = 9000; // deklariert z als einen 'long' Datentyp
  ```

- **float**

  Ein Datentyp für Fliesskomma Werte oder Nummern mit Nachkommastelle. Fliesskomma Nummern haben eine bessere Auflösung als Integer und werden als 32-bit Wert mit einem Spektrum von -3.4028235E+38 bis 3.4028235E+38.

  ```
  float pi = 3.14; // deklariert pi als einen 'float' Datentyp
  ```

  Bemerkung: Flieskomma zahlen sind nicht präzise und führen möglicherweise zu merkwürdigen Resultaten wenn sie verglichen werden. Außerdem sind Fliesskomma berechnungen viel langsamer als mit Integer Datentypen. Berechnungen mit Fliesskomma Werten sollten nach Möglichkeit vermieden werden.

Aufgabe A3.1:

- Führe bei den Sketches für das Lauflicht und die Ampel Variablen ein.
- Speichere die Sketches unter dem Namen „A031_Lauflicht_mit_Variable_(xx)“ und „A032_Ampel_mit_Variable_(xx)“ ab.

Lautsprecher

Neue Bauelemente: Lautsprecher

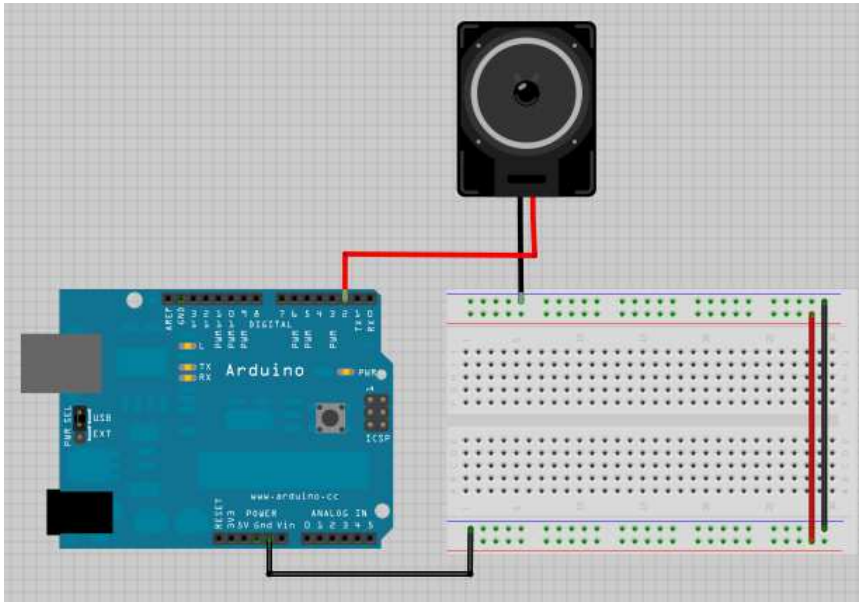
Neue Befehle: tone, noTone

Ziel: Töne und Melodien auf einem Lautsprecher erzeugen.

Aufgabe A4.1:

- Schließe an Pin2 einen Lautsprecher an.
- Übernehme den Sketch „A041_Lautsprecher" und übertrage ihn auf den Arduino.

Die 3 LEDs (rot, gelb und grün) können auf der Steckplatine bleiben. Sie werden später für die Ampel noch mal benötigt.



Aufgabe A4.2

- Erzeuge eine eigene kleine Melodie.
- Speichere den Sketch unter „A042_Lautsprecher_Melodie_(xx)"

Kapitel 4

A041_Lautsprecher

```
int lautPin = 2;          //Lautsprecher an Pin2
int note_a = 440;

void setup() {}

void loop() {
  tone(lautPin,note_a);   //Ton mit der Frequenz 440 Hz

  delay(2000);            //Die Länge des Tones beträgt 2000 ms

  noTone(lautPin);        //Ton wird ausgeschalten

  delay(1000);            //1000 ms Pause
}
```

Neue Befehle:

- **tone(pin, Frequenz)**

  Es wird ein Ton mit der in Hz angegebenen Frequenz erzeugt.

  Die Länge des Tones wird durch die anschließende Pause bestimmt.

- **noTone(pin)** schaltet den Lautsprecher bzw. den Ton aus.

Für Musikalische:

Melodien lassen sich natürlich leichter erstellen, wenn man die Note, z.B. „c" oder „g", und deren Länge „1/4" oder „1/8", direkt eingeben kann:

Aufgabe A4.3:

- Öffne den Sketch „A043_Lautsprecher"
- Starte den Sketch.
- Erzeuge eine eigene Melodie.
- Speichere den Sketch unter „A043_Lautsprecher_(xx)"

Eine Tabelle mit Noten und den entsprechenden Frequenzen findest du im Anhang

Unterprogramme I

Kapitel 5

Neue Bauelemente: -

Neue Befehle: void (eigentlich kein neuer Befehl!)

Ziel: Sketches einfacher und übersichtlicher gestalten.

Sketches lassen sich einfacher und übersichtlicher gestalten, wenn immer wieder auftretende Programmteile in Unterprogrammen „ausgelagert" werden. Diese können dann mit nur einem Befehl an einer beliebigen Stelle aufgerufen werden.

Die beiden Programmteile `void setup()` und `void loop()` bilden die bekannte Grundstruktur eines Sketches.

Entsprechend lassen sich eigene Unterprogramme erstellen.

Ohne Unterprogramm:

A051_Lautsprecher

```
int lautsprecher=2;

void setup() {}

void loop() {
  tone(lautsprecher,100);
  delay(1000);
  noTone(lautsprecher);

  tone(lautsprecher,400);
  delay(1000);
  noTone(lautsprecher);

  delay(1000);
}
```

Melodie

Mit Unterprogramm:

A051_Lautsprecher_mit_Unterprogramm

```
int lautsprecher=2;

void setup() {}

void loop() {
  melodie();
  delay(1000);
}

void melodie() {
  tone(lautsprecher,100);
  delay(1000);
  noTone(lautsprecher);

  tone(lautsprecher,400);
  delay(1000);
  noTone(lautsprecher);
}
```

Das Unterprogramm ‚**melodie'** wird mit **einer** Befehlszeile aufgerufen.

Sobald das Unterprogramm durchgelaufen ist, geht es mit der nächsten Befehlszeile weiter.

Aufgabe A5.1:

- Öffne den Sketch „A041_Lautsprecher_Melodie_(xx)"
- Ändere den Sketch so ab, dass der Melodie-Teil in einem Unterprogramm steht.
- Speichere den Sketch unter „A051_Unterprogramm_Melodie_(xx)".

while-Schleife

Neue Bauelemente: -

Neue Befehle: while

Ziel: Den Wert einer Variablen schrittweise ändern.

Mit der **while**-Schleife ist es möglich, einen Block aus Anweisungen wiederholt auszuführen. Die while-Schleife kann z.B. verwendet werden, um den Wert einer Variablen schrittweise zu ändern.

A061_while_LED

```
int z;              // Zählvariable z
int led = 3;

void setup() {
  pinMode(led, OUTPUT);

  z = 1;

  while (z <= 10) {
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000);
    z = z + 1;
  }
}

void loop() {}
```

Startwert

Bedingung

Endwert

Anweisungs-block

Schrittweite

Aufgabe A6.1:

- Schließe eine LED an Pin 3 an.
- Übernehme den Sketch A061_while_LED und übertrage ihn auf den Arduino.
- Ändere den Sketch so ab, dass die LED nur 3 mal blinkt.

A062_Lautsprecher_Signal

```
int lautPin = 2;
int frequenz;

void setup() {}

void loop() {
  frequenz = 400;

  while (frequenz <= 4000) {
    tone(lautPin, frequenz);
    delay(100);
    frequenz = frequenz + 100;
  }
}
```

Startwert

Endwert

Schrittweite

Aufgabe A6.2:

- Schließe einen Lautsprecher an Pin2 an.
- Übernehme den Sketch „A062_Lautsprecher_Signal"
- Übertrage den Sketch auf den Arduino.
- Ändere nacheinander den Startwert, Endwert und die Schrittweite in der for-Schleife und beobachte das Ergebnis.

Aufgabe A6.3:

- Ergänze den bestehenden Sketch um eine zweite while-Schleife, die einen absteigenden Ton erzeugt.
- Speichere den Sketch unter dem Namen „A063_Lautsprecher_Signal_auf_ab_(xx)"
- Übertrage den Sketch auf den Arduino.

Aufgabe A6.4:

- Ändere den Sketch nun so ab, dass die beiden for-Schleifen in einem Unterprogramm „Signal" stehen.
- Speichere den Sketch unter dem Namen „A064_Signal_Unterprogramm_(xx)"

Analoger Ausgang

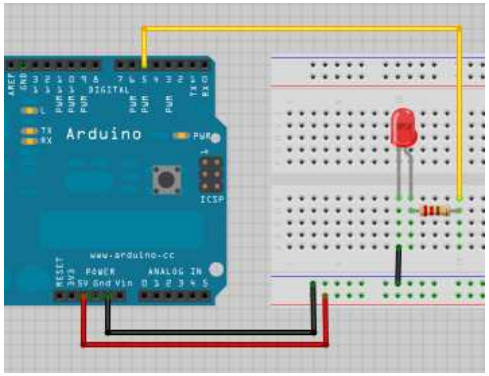
Neue Bauelemente: -

Neue Befehle: analogWrite

Ziel: Eine LED soll ein- und ausfaden, d.h. langsam heller und dunkler werden.

Aufgabe 7.1:

- Baue die Schaltung wie in der Abbildung auf.
- Übernehme den Sketch „A071_LED_Faden"
- Starte den Sketch.



A071_LED_Faden

```
int ledPin = 5;

void setup()  {
   //für analogWrite muss kein Pinmode verwendet werden.
}

void loop()  {

  for(int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
    analogWrite(ledPin, fadeValue);
    delay(30);  //Zeitverzögerung
  }

  for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
    analogWrite(ledPin, fadeValue);
    delay(30);
  }

}
```


Kapitel 7

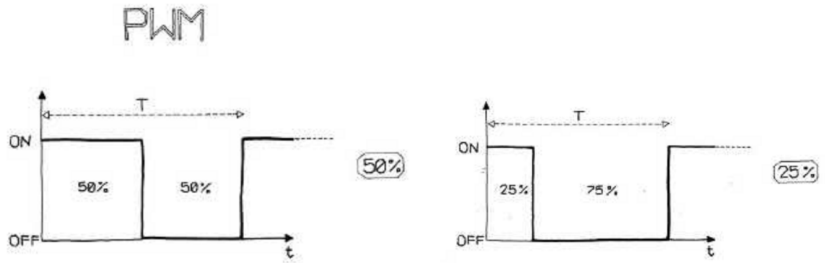
INFO: PWM (PulsWeitenModulation)

Ein digitaler Pin kennt die beiden Zustände HIGH und LOW. Dann liegen an diesem Pin entweder 0V oder 5V an.
Mit dem Befehl **analogWrite(pin, value)** kann man an einigen Pins (siehe Pin-Belegung) auf dem Launchpad eine pseudo-analoge Spannung erzeugen:

Der Befehl **analogWrite(10, 0)** generiert eine gleichmäßige Spannung von 0 Volt an Pin10, der Befehl **analogWrite(10, 255)** generiert eine gleichmäßige Spannung von 5 Volt an Pin10. Für Werte zwischen 0 und 255 wechselt der Pin sehr schnell zwischen 0 und 5 Volt - je höher der Wert, desto länger ist der Pin HIGH (5 Volt).

Der Befehl **analogWrite(10, 127)** führt dazu, dass die Ausgangsspannung zur Hälfte der Zeit auf HIGH steht und zur anderen Hälfte auf LOW. Bei **analogWrite(10, 192)** misst die Spannung am Pin zu einer Viertel der Zeit 0 Volt und zu Dreivierteln die vollen 5 Volt.

Weil dies eine hardwarebasierte Funktion ist, läuft die konstante Welle unabhängig vom Programm bis zur nächsten Änderung des Zustandes per analogWrite (bzw. einem Aufruf von digitalRead oder digitalWrite am selben Pin).



Aufgabe 7.2:

- Ergänze eine zweite LED.
- Ändere den Sketch so ab, dass die LEDs abwechselnd hell und dunkel werden.
- Speichere den Sketch unter dem Namen „A072_2LED_Faden_(xx)" ab.

Aufgabe 7.3:

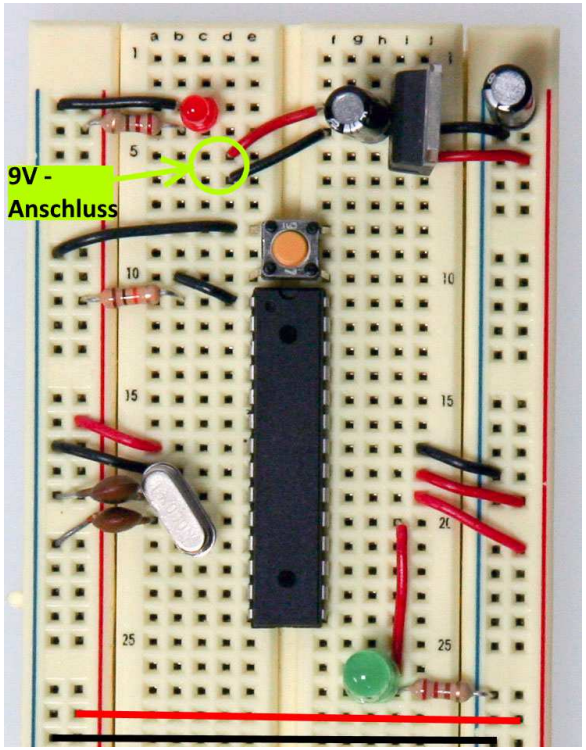
- Ändere den Sketch „A072_2LED_Faden_(xx)" so ab, dass das Ein- und Ausfaden der LEDs durch den Aufruf eines Unterpogramms geschieht.
- Speichere den Sketch unter dem Namen „A073_2LED_Faden_Unterprogramm_(xx)" ab.

Arduino (Standalone)

Standalone

Der Mikrocontroller ***ATmega328*** kann, nachdem er auf dem Board programmiert wurde, auch direkt auf einer Steckplatine verbaut werden. Dazu müssen noch ein Spannungsregler, ein 16 MHz Oszillator sowie einige andere Bauteile und Verbindungen hinzugefügt werden.
Eine gute Anleitung gibt es auf der Arduino-Webseite:

https://www.arduino.cc/en/Main/Standalone



Hinweis:
Bitte die Orientierung des Mikrocontrollers beachten! Die Kerbe zeigt hier nach oben, auf dem Breadbord zeigt die Kerbe weg vom USB-Anschluss.

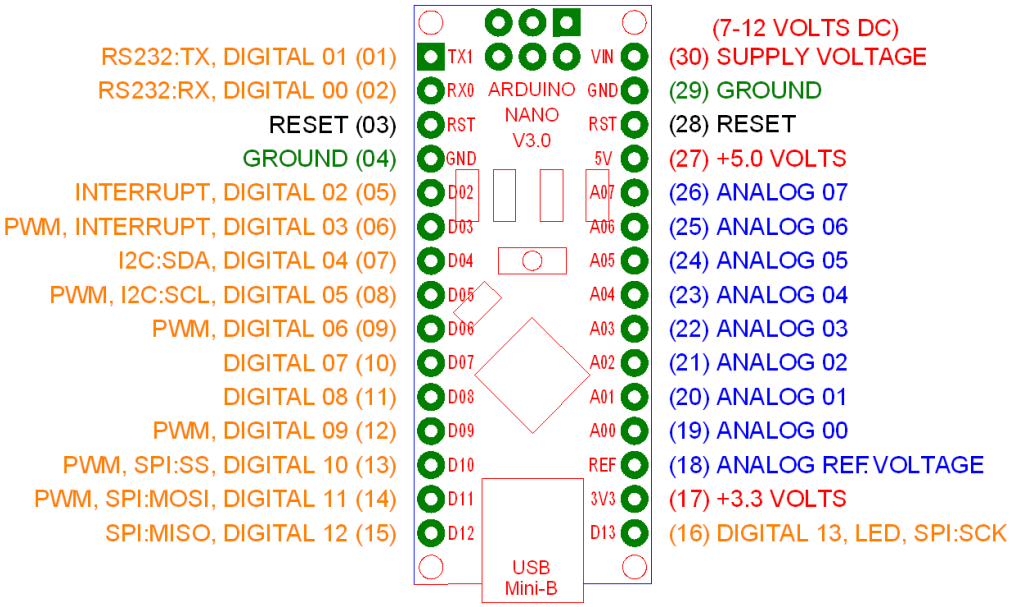
Nano_____ Kapitel 8

Arduino Nano

Eine weitere Möglichkeit, Arduino-Projekte preisgünstig und platzsparend umzusetzen, ist die Verwendung eines Arduino-Nano. Dieser kostet nicht viel mehr als ein einzelner Mikroconteroller *ATmega328.*

Der Arduino Nano kann direkt über ein Mini-USB-Kabel programmiert werden. Damit entfällt das Herausnehmen des Mikrocontrollers und die Gefahr, dass Füßchen abknicken.
Außerdem kann direkt eine 9V-Batterie an Vin angeschlossen werden und ein zusätzlicher Spannungsregler ist nicht erforderlich.

Wichtig:
Oft muss noch ein Treiber für den USB/Seriell-Wandler nach-installiert werden.
In der Regel wird der Nano mit dem Wandlerchip der Firma Winchiphead ausgeliefert.
Der Treiber ist im Internet zu finden.



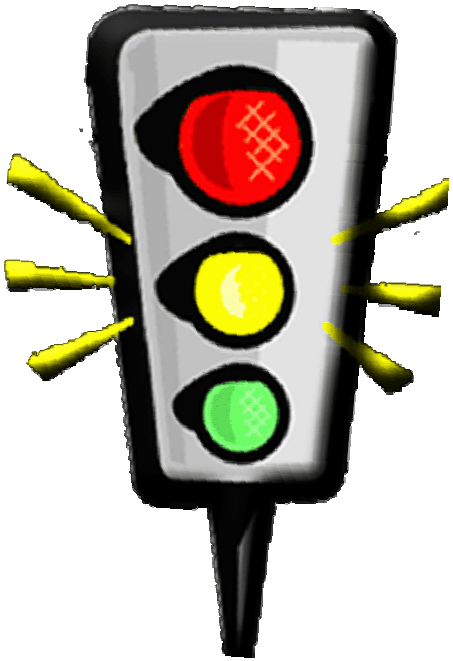
Projekt 1:
Signalampel

Aufgabe:

- Baue eine Ampelschaltung aus einer roten, gelben und grünen LED auf.
- Bei „Rot" soll ein Warnton erklingen.
- Verwende nur den Mikrocontroller bzw. einen Arduino-Nano, nicht das Board!
- Verwende bei der Programmierung Unterprogramme.
- Speichere den Sketch unter „Signalampel_Projekt1_(xx)" ab.

Tipp:

1. Belasse den Mikrocontroller zunächst auf dem Arduino-Board schreibe den Sketch.
2. Erst wenn alles funktioniert kannst du den Mikrocontroller mit auf die Steckplatine packen.



Teil 3:
Serielle Kommunikation
LCD-Display

Serielle Kommunikation

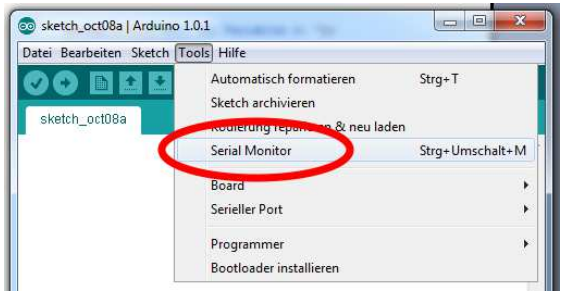
Kapitel 9

Neue Bauelemente: -

Neue Befehle:
- Serial.Begin
- Serial.print, Serial.println

Ziel: Daten vom MSP430 zurück an den Computer senden und in einem Fenster (serieller Monitor) darstellen lassen.

Aufgabe A9.1:
- Übernimm den Sketch „A091_serielle_Kommunikation" bzw. „A091_serielle_Kommunikation_2"
- Übertrage den Sketch auf den Arduino und öffne den „seriellen Monitor" (STRG+UMSCHALT+M).
- Was siehst du im seriellen Monitor?



1. Ausgabe von Text:

A091_Serielle_Kommunikation

```
void setup() {
  Serial.begin(9600);
}

void loop() {
    Serial.println("Hello World!");
}
```

Die serielle Kommunikation wird mit einer Übertragungsrate (Baudrate) von 9600 bits pro Sekunde gestartet.

Der Text in Anführungszeichen wird ausgegeben.

2. Ausgabe des Werts einer Variablen:

A091_Serielle_Kommunikation_2

```
int z = 0;   //Variable z hat den Wert 0.

void setup() {
  Serial.begin(9600);
}

void loop() {

    Serial.print("Wert der Variablen z: ");
    Serial.println(z);
    z++;
    delay(500);
}
```

z++ bedeutet, dass z um 1 erhöht wird.

Neue Befehle:

- **Serial.begin(9600)**

  Öffnet den seriellen Port und setzt die Datenrate auf 9600 bps.

- **Serial.print / Serial.println**

  Serial.print("Das ist eine Nachricht ohne Zeilenumbruch");

  Serial.println("Das ist eine Nachricht mit Zeilenumbruch am Ende");

  Serial.print(z); Sendet den Wert der Variable *z*

Aufgabe A9.2:
- Schreibe einen Sketch, mit dem die Werte einer for-Schleife im seriellen Monitor dargestellt werden.
- Speichere den Sketch unter „A092_SerialPrint_for_(xx)"

Serielle Kommunikation

Kapitel 10

Neue Bauelemente: -

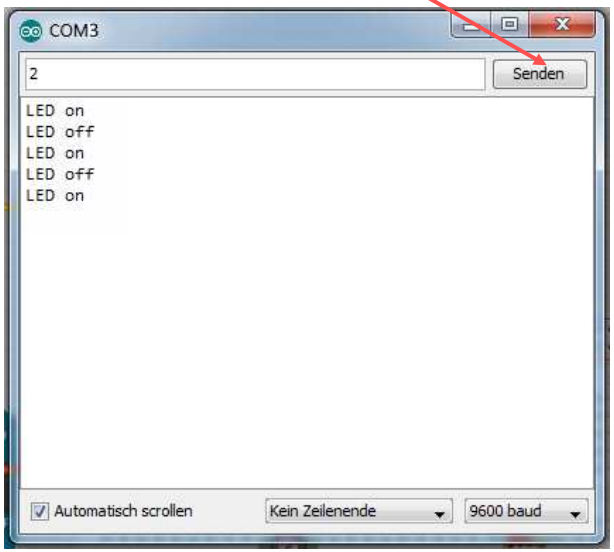
Neue Befehle: Serial.read

Ziel: Daten von der Tastatur einlesen.

Um Programmabläufe zu steuern kann es geschickt sein, Daten von der Tastatur einzulesen. Mit dem folgenden Sketch wird über die Tastatur eine Leuchtdiode an- und ausgeschalten.

Aufgabe A10.1:

- Schließe an Pin13 eine LED an (Kann ohne Vorwiderstand mit GND verbunden werden).
- Übernehme den Sketch „A101_Serial_Read" und übertrage ihn auf den Arduino.
- Öffne den seriellen Monitor.
  Gib in der Kommandozeile
  "1" für LED on
  "2" für LED off
  ein, und drücke auf „Send".
- Beobachte die LED.



A101_Serial_read

```
int led = 13;

void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

void loop () {
  if (Serial.available() >0) {
    int command = Serial.read();
    if (command == '1') {
      digitalWrite(led, HIGH);
      Serial.println("LED on");
    } else if (command == '2') {
      digitalWrite(led, LOW);
      Serial.println("LED off");
    } else {
      Serial.print("Unbekannter Befehl");
      Serial.println(command);
    }
  }
}
```

Neue Befehle:

- **Serial.Read()**

  Zunächst wird überprüft, ob überhaupt Daten (hier: von der Tastatur) anliegen. Dies geschieht mit dem Befehl: **if (Serial.available()>0)**

  Sind Daten verfügbar, dann werden sie mit **Serial.read ()** ausgelesen.

Aufgabe A10.2:

- Starte den Sketch erneut und öffne den seriellen Monitor.
- Gib in der Kommandozeile
  "3", „a" oder „A" ein, und drücke auf „Send".
- Welches Ergebnis liefert der serielle Monitor? Notiere!

LC-Display (I²C)

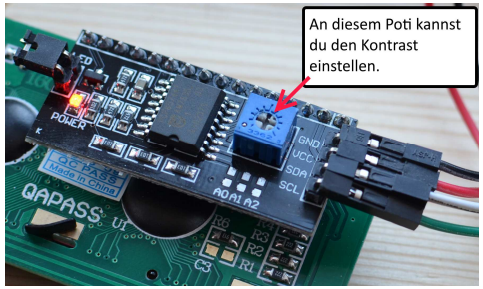
Neue Bauelemente: LCD-Display

Neue Befehle:
- lcd.begin
- lcd.setCursor
- lcd.print

Ziel: Darstellung von Text auf einem LCD-Display.

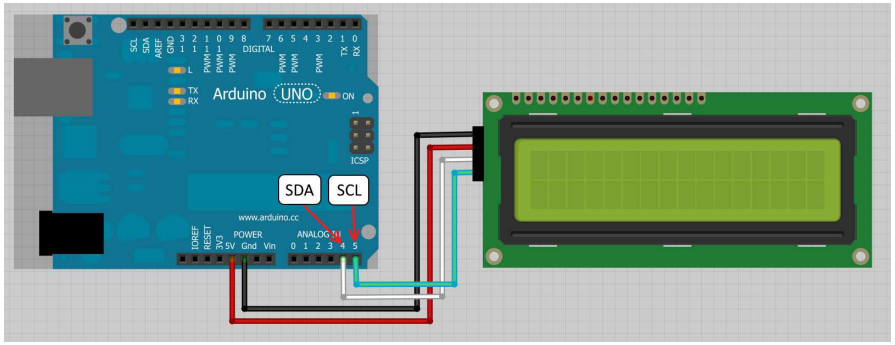
Wird der Arduino in einem Projekt ohne Computer betrieben, muss ein LC-Display eingesetzt werden, um Text oder Variablenwerte darzustellen.

Im Handel sind verschiedene Ausführungen von LC-Displays zu erhalten. Wir verwenden hier ein sogenanntes **I²C-LCD**. Auf der Rückseite des Displays befindet sich ein Zusatzmodul mit 4 Anschlüssen:



- 2 Anschlüsse für die Energieversorgung
- 2 Anschlüsse für die Übertragung der Daten:
  SDA (Datenleitung) und SCL (Taktleitung).

Anschluss:



Kapitel 11

Beispiel 1:

A111_LCD__I2C_

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  lcd.begin();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Hello World!");
}

void loop() {
}
```

Hier werden zwei Bibliotheken eingebunden:
1.) Für den I²C – Datenaustausch allgemein
2.) Für die Verwendung des I²C-LCD

Hier wird die Adresse (**0x27**) und die Größe (16 Spalten und 2 Zeilen) des LCD festgelegt.

Neue Befehle:

- **lcd.begin**
  Der Befehl lcd.begin() startet die Verbindung mit dem LC-Display.

- **lcd.setCursor**
  Der Befehl lcd.setCursor(0, 0) legt Spalte (1. Wert) und Zeile (2. Wert) fest, wo der Text angezeigt werden soll.
  Verändere die Zahl so, dass der Text in der Mitte der unteren Zeile angezeigt wird.

- **lcd.print**
  Der Befehl druckt den in Anführungszeichen angeführten Text, bzw. den Wert einer Variablen. (→Vgl. Serial.print)

Achtung: *Falls das Display nicht funktioniert…*
- Überprüfe noch einmal genau die Verkabelung.
- Es gibt im Handel auch I²C-Module, die die **Adresse 0x3f** haben. Ändere also die Adresse in der 3. Zeile und lade den Sketch erneut hoch.

Beispiel 2:
„Zeit“

A111_LCD__I2C_mit_Zeit

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
int t = 0;

void setup() {
  lcd.begin();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Hello World!");
}

void loop() {
  lcd.setCursor(0, 1);
  t = millis(); // vergangene Millisekunden
                // seit Programmstart
  lcd.print(t / 1000);
}
```

Aufgabe A11.1:

- Schreibe einen Sketch, der in der unteren Zeile ein Sternchen von links nach rechts und zurück laufen lässt. Dabei soll das Sternchen jede Sekunde eine Position vorwärts rücken. Achte darauf dass es nur ein einzelnes Sternchen bleibt!
- Speichere den Sketch unter "A111_LCD_Sternchen_(xy)" ab.

Beispiel 3:
„Zähler“

A111_LCD_I2C_Zaehler

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  lcd.begin();
  lcd.setCursor(0, 0);
  lcd.print("Variable a = ");
}

void loop() {
  lcd.setCursor(13,0);
  lcd.print("  ");

  for (int a = 0; a <= 15; a+=1) {
    lcd.setCursor(13,0);
    lcd.print(a);
    delay(1000);
  }
}
```

Der alte Wert wird hier mit Leerzeichen überschrieben.

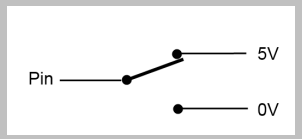
Ansonsten kann es sein, dass „alte Ziffern“ stehen bleiben.

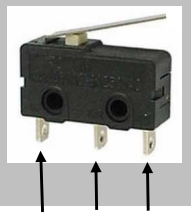
Teil 4:
Digitale und analoge Eingänge

Wechselschalter

Neue Bauelemente: Wechselschalter

Neue Befehle:
- digitalRead
- if (else)

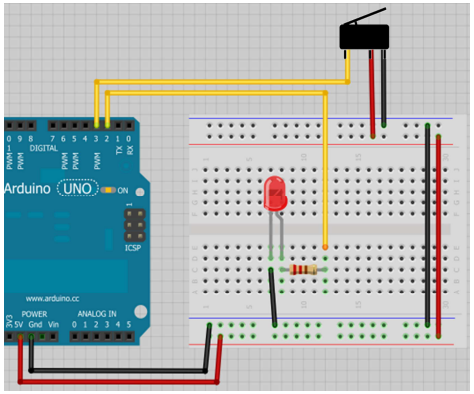




Ziel: Eine LED mit einem Wechselschalter an- und ausschalten.

Wechselschalter

Bei offenem Schalter ist der linke Kontakt mit dem mittleren Kontakt verbunden. Bei gedrücktem Schalter ist der linke Kontakt mit dem rechten Kontakt verbunden.
Damit ist der Zustand von Pin3 **eindeutig** festgelegt.
Es liegen entweder 5V (HIGH) oder 0V (LOW) an.



Tipp: Löte in entsprechenden Farben Drähte an den Wechselschalter an (siehe Schaltung). Dann kommt es zu keinen Kurzschlüssen.

Aufgabe A12.1:
In dieser Aufgabe geht es darum, den Zustand des Schalters auszulesen und das Ergebnis auf dem seriellen Monitor darzustellen.

- Baue die Schaltung wie in der Abbildung auf **(noch ohne LED).**
- Übernehme den Sketch „A121_Wechselschalter_SerialMonitor".
- Übertrage den Sketch.
- Öffne den seriellen Monitor und verändere die Schalterstellung.

Aufgabe A12.2:
In dieser Aufgabe geht es darum, mit einem Wechselschalter eine LED zu steuern.

- Ergänze deine Schaltung mit einer LED (an Pin2) und probiere den Sketch „A122_Wechselschalter_LED" aus.

Kapitel 12

A121_Wechselschalter_SerialMonitor

```
int schalterPin = 3;
int schalterValue;

void setup() {
  Serial.begin(9600);
  pinMode(schalterPin, INPUT);
}

void loop() {
  schalterValue = digitalRead(schalterPin);

  Serial.print(schalterValue);
  delay(1000);
}
```

„Schalterpin" 3 wird als **INPUT**-Pin verwendet

Schalterpin 3 wird mit dem Befehl „digitalRead" ausgelesen.
Das Ergebnis (0 oder 1) wird in der Variablen „schalterValue" gespeichert.

Der Wert der Variablen „schalterValue", also der Zustand des Schalters, wird auf dem seriellen Monitor ausgegeben.

A122_Wechselschalter_LED

```
int ledPin =  2;

int schalterPin = 3;
int schalterValue;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(schalterPin, INPUT);

}

void loop() {
  schalterValue = digitalRead(schalterPin);

  while (schalterValue == 0) {
    digitalWrite(ledPin, HIGH);
    schalterValue = digitalRead(schalterPin);
  }

    digitalWrite(ledPin, LOW);
}
```

Solange der Schalter nicht gedrückt wird werden die beiden Anweisungen „LED an" und „Schalterabfrage" ausgeführt.
Sobald der Schalter gedrückt wird, wird die while-Schleife verlassen.

Aufgabe A12.3:
In dieser Aufgabe geht es darum, mit einem Wechselschalter zwischen einer LED und einem Lautsprecher hin und her zu schalten:

Schließe zusätzlich einen Lautsprecher an Pin 4 an.

A123_Schalter_LED_Lautsprecher

```
int ledPin =  13;
int lautPin = 4;

int schalterPin = 3;
int schalterValue;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(schalterPin, INPUT);
}

void loop() {
  schalterValue = digitalRead(schalterPin);

  if (schalterValue == HIGH) {
    digitalWrite(ledPin, HIGH);
    noTone(lautPin);
  }
  else {
    digitalWrite(ledPin, LOW);
    tone(lautPin, 400);
  }
}
```

Wenn (**IF**) der Schalter gedrückt wird, geht die LED an und der Lautsprecher aus.

Ansonsten (**ELSE**) geht die LED aus und der Lautsprecher an.

Aufgabe A12.4: „Projekt Ampel“

Wird ein Schalter betätigt, schält zunächst die Ampel für die Autofahrer auf Rot. Danach geht die Ampel für die Fußgänger auf Grün und nach einer gewissen Zeit wieder zurück auf Rot. Danach geht die Autoampel wieder auf Grün.

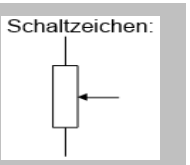
Potentiometer 13

Neue Bauelemente: Potentiometer

Neue Befehle: -

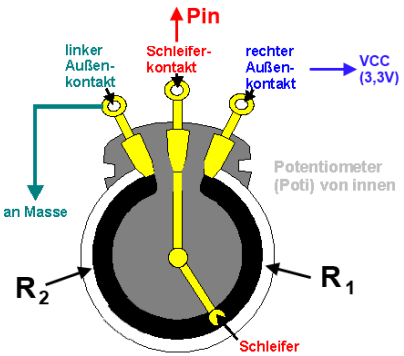
Ziel: Aufbau und Funktionsweise eines Potentiometers.
Prinzip der Spannungsteilung.

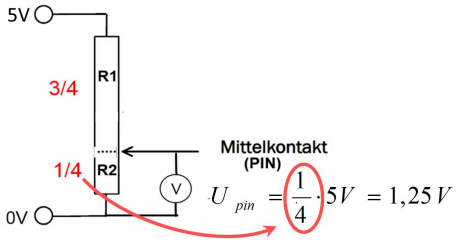
Schaltzeichen:

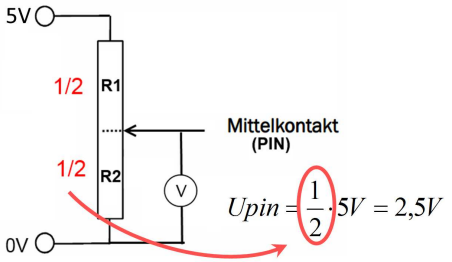


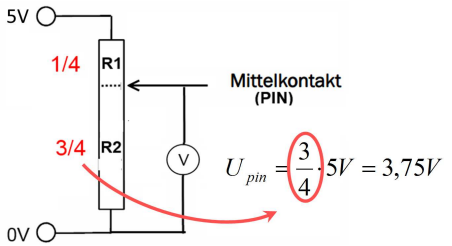
Aufbau eines Potentiometers:

Das Potentiometer besteht aus drei Anschlüssen (s. Abb.). Zwei Anschlüsse an den Enden des Widerstandselements und einem beweglichen Gleitkontakt (auch Schleifer genannt). Der Schleifer bewegt sich durch Drehen des Knopfes auf einer Kohlebahn und teilt den festen Gesamtwiderstand in zwei Teilwiderstände **R_1** und **R_2** auf.









Prinzip der Spannungsteilung:

Auch die Gesamtspannung (hier 5V) teilt sich auf die beiden Widerstände auf.
Am Mittelkontakt (→Pin) liegt dann die Spannung U_{pin} an, die am Widerstand R_2 abfällt:

Allgemein gilt:

$$U_{pin} = \frac{R_2}{R_{ges}} \cdot 5V = \frac{R_2}{R_1 + R_2} \cdot 5V$$

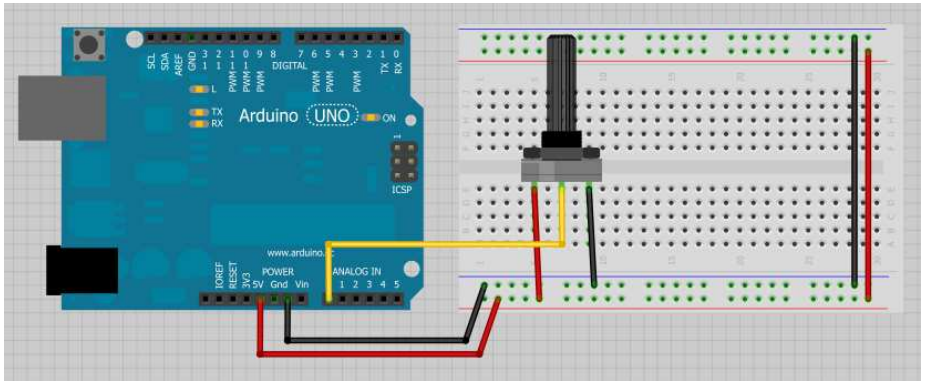
Analoger Eingang

Neue Bauelemente: Potentiometer

Neue Befehle:
- analogRead

Ziel: Ein Potentiometer wird an einem analogen Eingang angeschlossen. Der anliegende Wert wird ausgelesen und über die serielle Verbindung an den Computer zurückgesendet.

Aufgabe A14.1:
- Baue die Schaltung wie in der Abbildung auf.
- Übernehme den Sketch „A141_Poti".
- Starte den Sketch.
- Öffne den „seriellen Monitor" (STRG+UMSCHALT+M).
- Verändere die Stellung des Potentiometers und beobachte die Werte im Fenster des seriellen Monitors. Notiere deine Beobachtungen.
- Zeichne den Schaltplan.



Aufgabe A14.2:
- Ergänze die Schaltung um eine LED.
- Schreibe einen Sketch, so dass die LED blinkt. Die Blinkfrequenz soll mit dem Potentiometer geregelt werden.
- Speichere den Sketch unter dem Namen „A143_Poti_LED_(xx)" ab.

Aufgabe A14.4: „Dimmen"
- Schreibe einen Sketch der es ermöglicht, die LED zu dimmen, d.h. mit dem Potentiometer die Helligkeit der LED einzustellen.
- Speichere den Sketch unter dem Namen „A144_Poti_LED_dimmen_(xx)" ab.

Kapitel 14

A141_Poti

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  delay(1); //delay in between reads for stability
}
```

Neue Befehle:

- **analogRead(pin)**

  Liest den Wert eines festgelegten analogen Pins aus. Die resultierenden Integer Werte haben ein Spektrum von 0 (→0V) bis 1023 (→5V)

  Analoge Pins müssen im Gegensatz zu digitalen nicht zuerst als Eingang oder Ausgang deklariert werden.

Wichtig:

Die Bezeichnung des analogen Pins kann auf zwei Arten erfolgen:

```
analogRead(A0) = analogRead(0)
```

Fotowiderstand LDR

Neue Bauelemente: LDR (**L**ight **D**ependent **R**esistor) Lichtsensor

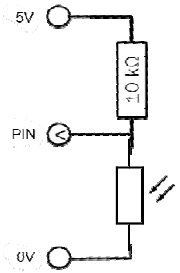
Neue Befehle: -

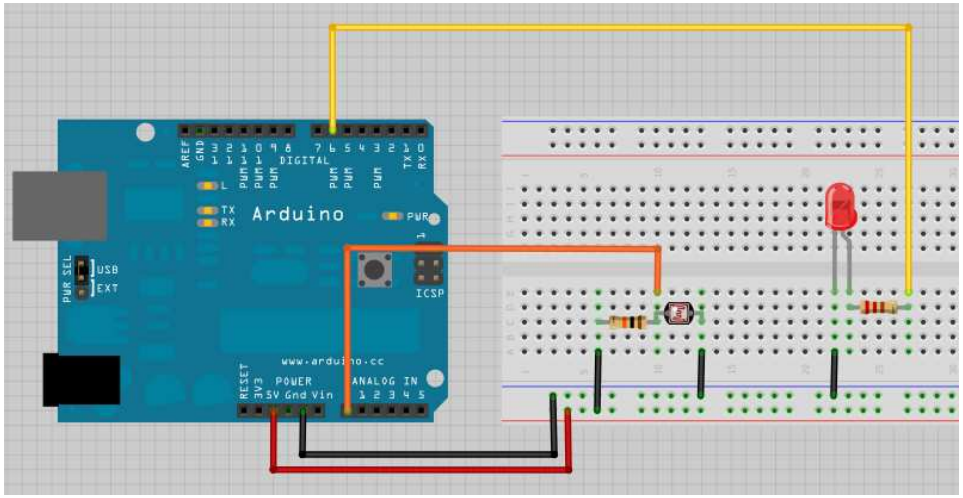
Ziel: Eine LED soll leuchten, wenn ein LDR abgedunkelt wird.

In dieser Schaltung wird das Prinzip des Spannungsteilers (→ Potentiometer) aufgegriffen. Der Spannungsteiler besteht aus zwei separaten Widerständen:

- einem Festwiderstand (hier: 10 kΩ)
- einem Fotowiderstand (LDR), dessen Widerstand mit Lichteinfall abnimmt (und umgekehrt).

Damit ändert sich die Spannung am Mittelkontakt, die an einem analogen Pin ausgelesen werden kann.





Aufgabe 15.1:

- Baue die Schaltung wie in der Abbildung, jedoch ohne LED und Vorwiderstand, auf.
- Verwende den Sketch „A141_Poti".
- Starte den Sketch.
- Öffne den seriellen Monitor.
- Dunkle mit deiner Hand den LDR ab und beobachte dabei die Werte auf dem seriellen Monitor.

Kapitel 15

Info: LDR

Ein Fotowiderstand ist ein Halbleiter, dessen Widerstandswert lichtabhängig ist. Ein LDR besteht aus zwei Kupferkämmen, die auf einer isolierten Unterlage (weiß) aufgebracht sind. Dazwischen liegt das Halbleitermaterial in Form eines gewundenen Bandes (rot). Fällt das Licht (Photonen) auf das lichtempfindliche Halbleitermaterial, dann werden die Elektronen aus ihren Kristallen herausgelöst (Paarbildung). Der LDR wird leitfähiger, das heißt, sein Widerstandswert wird kleiner. Je mehr Licht auf das Bauteil fällt, desto kleiner wird der Widerstand

Aufgabe 15.2:

- Ergänze die Schaltung durch eine rote LED mit Vorwiderstand (Pin 10)
- Ändere den Sketch so ab (Befehl IF und ELSE), dass die LED ab einem bestimmten Schwellenwert anfängt zu leuchten.
- Speichere den Sketch unter dem Namen „A152_LDR_LED_(xx)" ab.
- Zeichne den Schaltplan.

Aufgabe 15.3:

- Füge eine grüne LED mit Vorwiderstand hinzu.
- Ändere den Sketch so ab, dass nur die rote LED oberhalb eines bestimmten Schwellenwerts leuchtet und die grüne LED nur unterhalb des Schwellenwerts leuchtet.
- Speichere den Sketch unter dem Namen „A153_LDR_2LED_(xx)" ab.

Taster

Kapitel 16

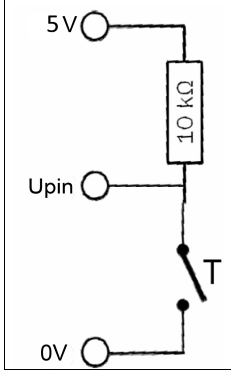
Neue Bauelemente: Taster

Neue Befehle: -

Ziel: Eine LED soll mit einem (einfachen) Taster an- und ausgehen.

Ersetzt man in einem Spannungsteiler den unteren Widerstand durch einen Taster bekommt man folgende beiden Zustände:

Taster geöffnet:



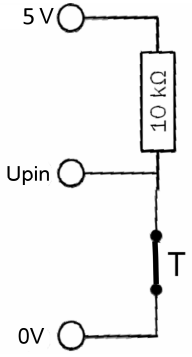
Ein geöffneter Taster entspricht einem **unendlich großen** Widerstand:

$$U_{pin} = \frac{R_2}{R_1 + R_2} \cdot 5V$$

$$= \frac{\infty}{10k\Omega + \infty} \cdot 5V$$

$$= \frac{\infty}{\infty} \cdot 5V = 5V$$

Taster geschlossen:



Ein geschlossener Taster hat **keinen** Widerstand:

$$U_{pin} = \frac{R_2}{R_1 + R_2} \cdot 5V$$

$$= \frac{0\Omega}{10k\Omega + 0\Omega} \cdot 5V$$

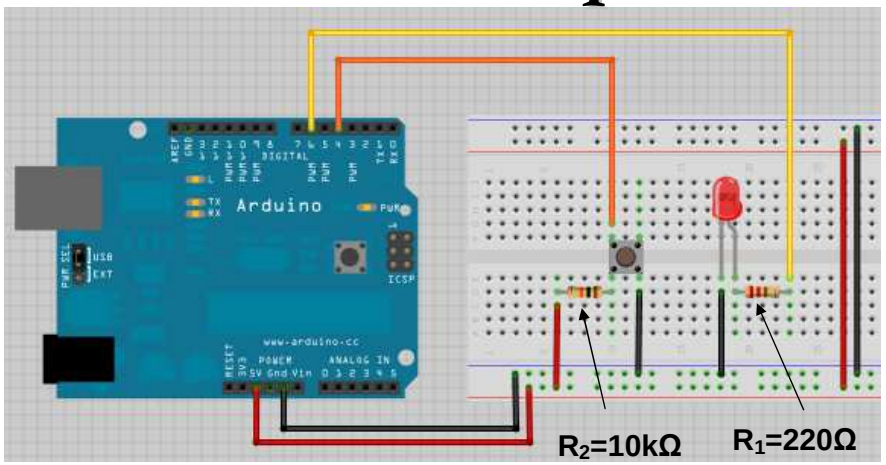
$$= \frac{0\Omega}{10k\Omega} \cdot 5V = 0V$$

Das Schließen und Öffnen des Tasters liefert, genau wie bei einem Wechselschalter, die Zustände HIGH (5V) und LOW (0V).

Durch diese Schaltung kann ein Wechselschalter durch einen Taster ersetzt werden.

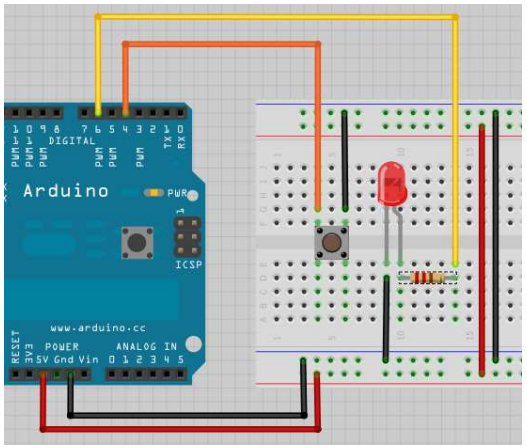
Aufgabe A16.1:

- Baue die Schaltung wie in der Abbildung auf.
- Schreibe einen Sketch, mit dem die LED durch Betätigen des Tasters zum Leuchten gebracht wird. (→ Orientiere dich am Sketch für den Wechselschalter)
- Speichere den Sketch unter „A161_Taster_LED_(xx)"



2. Möglichkeit: Interner Pullup-Widerstand

Mit dem Befehl `pinMode(tasterPin, INPUT_PULLUP);` wird Pin4 **intern** über einen hohen Widerstand mit **HIGH** verbunden.
Wird Pin4 anschließend über einen **äußeren** Taster mit GND (0V) verbunden, befindet sich Pin4 im Zustand **LOW**.
Dadurch kann man sich den äußeren Pullup-Widerstand von 10 kΩ und einige Kabel sparen.



A162_Taster_INPUT_PULLUP

```
int led = 6;
int tasterPin = 4;

void setup()
{
  pinMode(led, OUTPUT);
  pinMode(tasterPin, INPUT_PULLUP);
}

void loop()
{
 int taster = digitalRead(tasterPin);
 if (taster == HIGH) {
   digitalWrite(2, HIGH);
   }
   else {
   digitalWrite(2, LOW);
   }
 }
```

Rechnen mit Variablen_

Neue Bauelemente: -

Neue Befehle: -

Ziel: Rechenoperationen durchführen.

Wann macht das Rechnen mit Variablen Sinn?

Beispiele:

- Mit dem Befehl analogRead erhält man Werte von 0 bis 1023. Diese können in Spannungswerte von 0V bis 3,3V umgerechnet werden (→A17.1)
- Es wird ein Temperatursensor (Luftdrucksensor etc.) angeschlossen. Auch hier müssen die Werte von 0 bis 1023 auf die entsprechende Temperatur umgerechnet werden.
- Flächenberechnung:

```
int x = 4;
int y = 6;

void setup() {
   Serial.begin(9600);
   delay(4000);
   int flaeche = x*y;
   Serial.print("Die Fläche beträgt: ");
   Serial.println(flaeche);
}

void loop() {}
```

Damit der serielle Monitor rechtzeitig geöffnet werden kann, bevor die Rechnung ausgeführt und dargestellt wird, muss hier ein „delay“ eingefügt werden.

- Anteile in Prozent:

```
int x = 3;
int y = 5;

void setup() {
   Serial.begin(9600);
}

void loop() {
    float prozent = x*100/y;
    Serial.print("Der Anteil in Prozent beträgt: ");
    Serial.println(prozent);
}
```

Hier muss der Datentyp geändert werden, da Dezimalzahlen entstehen.

Kapitel 17

INFO: Rechenoperatoren

Arithmetische Operatoren umfassen Addition, Subtraktion, Multiplikation und Division. Sie geben die Summe, Differenz, das Produkt oder den Quotienten zweier Operatoren zurück.

```
y = y + 3;
x = x - 7;
i = i * 6;
r = r / 5;
```

Die Operation wird unter Beibehaltung der Datentypen durchgeführt. 9 / 4 wird so zum Beispiel zu 2 und nicht 2,25, da 9 und 4 Integer sind und keine Nachkommastellen unterstützen. Dies bedeutet auch, dass die Operation überlaufen kann wenn das Resultat größer ist als der Datentyp zulässt. Wenn die Operanden unterschiedliche Datentypen haben wird der größere Typ verwendet. Hat zum Beispiel eine der Nummern (Operanden) den Datentyp 'float' und der andere 'int', so wir Fliesskomma Mathematik zur Berechnung verwendet.
Bemerkung: Wähle variable Größen die groß genug sind, um die Werte der Ergebnisse zu speichern. Sei dir bewusst, an welcher Stelle die Werte überlaufen und auch was in der Gegenrichtung passiert. z.B. bei (0 - 1) oder (0 - - 32768). Für Berechnungen, die Brüche ergeben sollten, immer 'float' Variablen genutzt werden. Allerdings mit dem Bewusstsein der Nachteile: Großer Speicherbedarf und langsame Geschwindigkeit der Berechnungen.

Gemischte Zuweisungen

Gemischte Zuweisungen kombinieren eine arithmetische Operation mit einer Variablen Zuweisung. Diese werden üblicherweise in Schleifen gefunden, die wir später noch genauer Beschreiben werden. Die gängigsten gemischten Zuweisungen umfassen:

```
x ++        // identisch mit x = x + 1, oder Erhöhung von x um +1
x --        // identisch mit x = x - 1, oder Verminderung von x um -1
x += y      // identisch mit x = x + y, oder Erhöhung von x um +y
x -= y      // identisch mit x = x - y, oder Verminderung von x um -y
x *= y      // identisch mit x = x * y, oder Multiplikation von x mit y
x /= y      // identisch mit x = x / y, oder Division von x mit y
```

Bemerkung:
Zum Beispiel führt x *= 3 zur Verdreifachung des alten Wertes von 'x' und weist der Variablen 'x' des Ergebnis der Kalkulation zu.

Aufgabe A17.1:

- Rechne die ausgelesenen Werte so um, dass sie den Werten der am analogen Eingang anliegenden Spannung entsprechen.
- Ändere den Sketch so ab, dass auf dem seriellen Monitor der Spannungswert ausgegeben wird, wie z.B.: „Die Spannung beträgt: 2,1 V“
- Speichere den Sketch unter dem Namen „A142_Poti_Spannung_(xx)“ ab.

Teil 5
Bewegung

Motor I

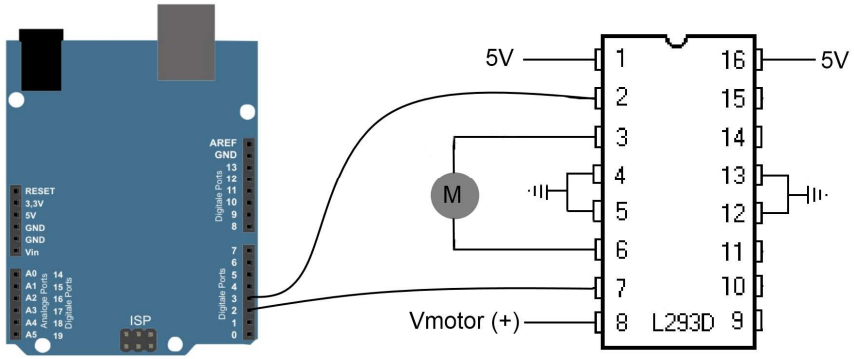
Neue Bauelemente: L293D (Motorsteuerungs IC)

Neue Befehle: -

Ziel: Ein oder zwei Gleichstrommotoren sollen angesteuert werden.

INFO: Motorsteuerung für 1 Motor:

Ein einfacher Elektromotor hat zwei Anschlüsse. Wenn eine Spannung anliegt, dreht sich der Motor in die eine Richtung, kehrt man die Spannung um, dreht sich der Motor in die andere Richtung. Welches Pin man dazu wo am L293D anschließen muss und wie man die Drehrichtung vom Mikrocontroller aus steuern kann, wird hier erklärt:

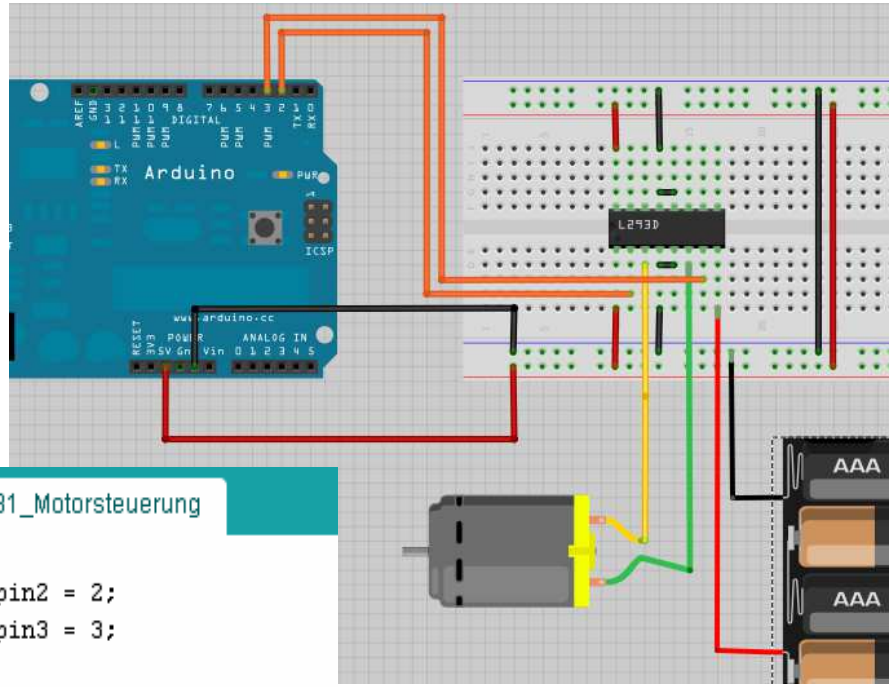


Pinbelegung L293D	Bedeutung und Belegung
4,5,12,13	Erdung (0V)
16	Energieversorgung des IC (5V)
1	Enable-Pin des IC. Ist dieses mit 5V verbunden, sind die Motor-Pins 3 und 6 aktiv.
8	Energieversorgung des Motors. Der Minus-Pol der Energieversorgung muss mit Erde (0V) verbunden sein, also z.B. mit Pin4.
3 und 6	Anschluss des Motors
2 und 7	Anschluss an die Pins 2 und 3 des Arduino. **Pin2: HIGH** **Pin3: LOW** → **Motor dreht sich** Pin2: LOW Pin3: LOW → Motor dreht sich nicht **Pin2: LOW** **Pin3: HIGH** → **Motor dreht sich umgekehrt** Pin2: HIGH Pin3: HIGH → Motor dreht sich nicht

Kapitel 18

Aufgabe 18.1:

- Baue die Schaltung wie in der Abbildung auf.
- Übernehme den Sketch und speichere ihn unter „A181_Motosteuerung" ab.



A081_Motorsteuerung

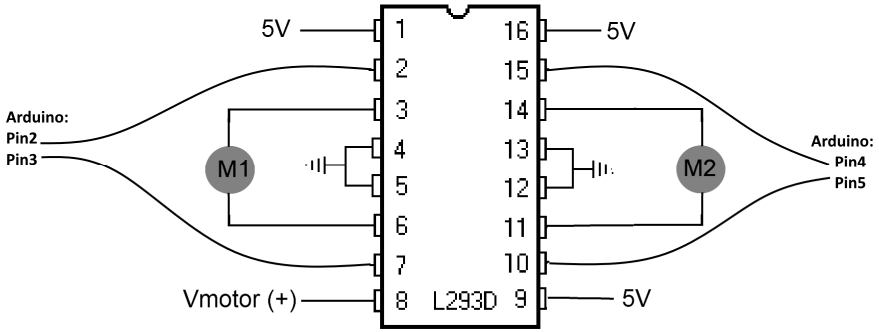
```
int pin2 = 2;
int pin3 = 3;

void setup() {
  pinMode(pin2, OUTPUT);
  pinMode(pin3, OUTPUT);
}

void loop() {
  digitalWrite(pin2, HIGH);  //Motor dreht sich
  digitalWrite(pin3, LOW);
  delay(1000);

  digitalWrite(pin2, LOW);   //Pause
  digitalWrite(pin3, LOW);
  delay(1000);
}
```

Aufgabe 18.2:

- Ändere den Sketch „A181_Motorsteuerung" so ab, dass sich der Motor 3 mal hintereinander (Befehl: for) jeweils 2 Sekunden in eine Richtung und nach einer Pause von 1 Sekunde 2 Sekunden in die andere Richtung dreht. Speichere den Sketch unter dem Namen „A182_Motorsteuerung_(xx)" ab.

Motor II

INFO: Motorsteuerung für 2 Motoren:

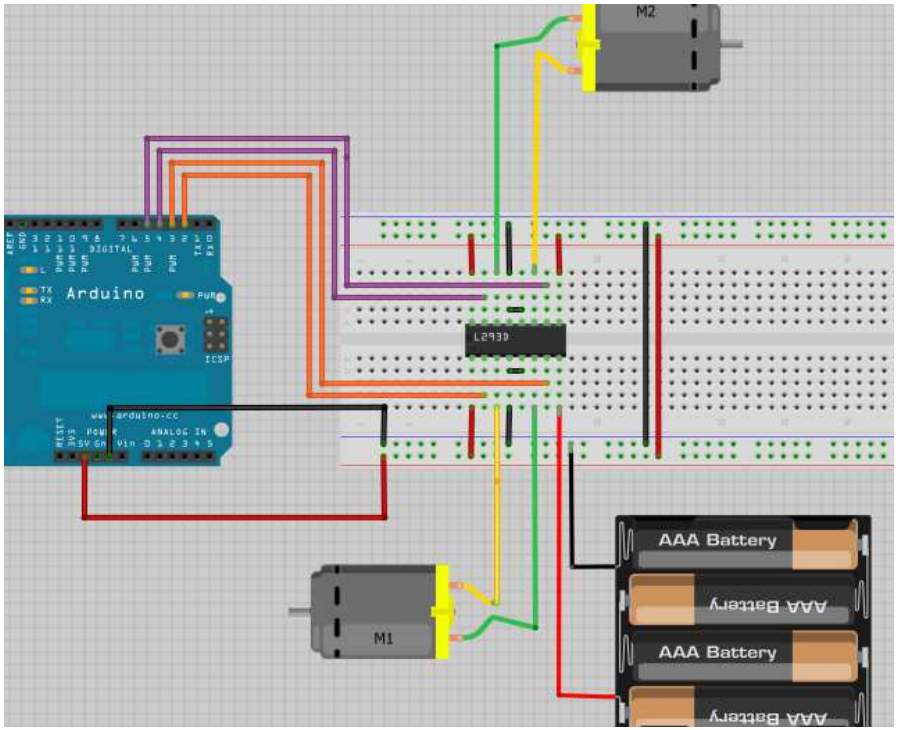


Pinbelegung L293D	Bedeutung und Belegung
4,5,12,13	Erdung (0V)
16	Energieversorgung des IC (5V)
1 und 9	Enable-Pin des IC. Ist dieses mit 5V verbunden, sind die Motor-Pins 3 und 6 bzw. 11 und 14 aktiv.
8	Energieversorgung des Motors. Der Minus-Pol der Energieversorgung muss Erde (0V) verbunden sein (Siehe Schaltung)
3 und 6	Anschluss Motor 1
11 und 14	Anschluss Motor 2
2 und 7	Anschluss an die Pins 2 und 3 des Arduino.
10 und 15	Anschluss an die Pins 4 und 5 des Arduino.

Aufgabe 19.1:

- Ändere den Sketch „A181_Motorsteuerung" so ab, dass sich die Motoren zunächst in die gleiche Richtung, in die umgekehrte Richtung und anschließend in verschiedene Richtungen drehen. (Pause 1 Sekunde)
- Erzeuge eine Ausgabe auf den seriellen Monitor, die über die Drehrichtung der einzelnen Motoren informiert.
- Speichere den Sketch unter „A191_Motorsteuerung2_(xx)" ab.
- Zeichne den Schaltplan

Kapitel 19



Aufgabe 19.2:

- Um die Geschwindigkeit der Motoren steuern zu können kann man die beiden „enable Pins" 1 und 9 des L293D an einen PWM-Ausgang des ARDUINO anschließen. Damit lässt sich die Leistung des L293D drosseln, indem man ihn z.B. nur 50% der Zeit aktiviert.
- Erweitere deine Schaltung.
- Ändere den Sketch „A191_Motorsteuerung2_(xx)" entsprechend ab und speichere ihn unter dem Namen „A192_Motorsteuerung2_PWM_(xx)" ab.
- Zeichne den Schaltplan.

Aufgabe 19.3: *Jetzt wird´s richtig kompliziert…*

- Regle die Geschwindigkeit der Motoren mit Hilfe eines Potis.
- Erzeuge eine Ausgabe auf dem seriellen Monitor, die die Leistung der Motoren in Prozent angibt.
- Speichere den Sketch unter dem Namen „A193_Motorsteuerung2_PWM_Poti_(xx)" ab.
- Zeichne den Schaltplan.

Unterprogramme II

Kapitel 20

Neue Bauelemente: -

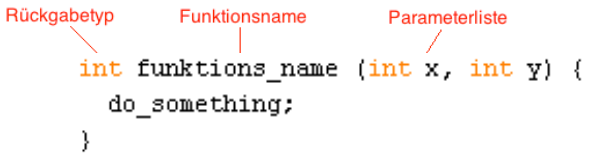
Neue Befehle: Unterprogramme und selbst definierte Funktionen

Ziel: Sketches einfacher und übersichtlicher gestalten.

Sketches lassen sich einfacher und übersichtlicher gestalten, wenn immer wieder auftretende Programmteile in Unterprogrammen „ausgelagert" werden und mit nur einem Befehl aufgerufen werden.

Dabei bilden die beiden Programmteile `void setup()` und `void loop()` die Grundstruktur eines Sketches.
Entsprechend lassen sich weitere Unterprogramme erstellen.

Allgemeine Form eines Unterprogramms:



Rückgabetyp:

Gibt den Datentyp des Werts an, der zurückgegeben wird. (Siehe Beispiel 3)
Wird **kein** Wert zurückgegeben, dann wird hier der Befehl **void** verwendet. (Siehe Beispiel 1 und 2)

Funktionsname:

Frei wählbarer Namen des Unterprogramms bzw. der Funktion.

Paramaterliste

Hier stehen die Werte, die an das Unterprogramm übergeben werden. Die Parameterliste kann auch leer bleiben (Siehe Beispiel 1)

1. Beispiel: Motorsteuerung

A091_Unterprogramme

```
int ml_1 = 2;
int ml_2 = 3;

void setup() {
  pinMode(ml_1, OUTPUT);
  pinMode(ml_2, OUTPUT);
}

void loop() {
  vorwaerts();     //Motor läuft vorwärts
  delay(1000);     //für eine Sekunde

  halt();          //Motor hält an
  delay(1000);     //und wartet eine Sekunde
 }

void vorwaerts () {
  digitalWrite(ml_1, HIGH);  //Motor dreht
  digitalWrite(ml_2, LOW);   //sich vorwärts
}

void halt () {
  digitalWrite(ml_1, LOW);   //Motor hält an
  digitalWrite(ml_2, LOW);
}
```

Das Unterprogramm **‚vorwaerts'** wird aufgerufen.

Unterprogramme II

Neue Bauelemente: -

Neue Befehle: Unterprogramme und selbst definierte Funktionen

Ziel: Sketches einfacher und übersichtlicher gestalten.

2. Beispiel: Motorsteuerung (mit Übergabe von Werten)

A092_Unterprogramme

```
int ml_1 = 2;
int ml_2 = 3;

void setup() {
  pinMode(ml_1, OUTPUT);
  pinMode(ml_2, OUTPUT);
}

void loop() {
  vorwaerts(3000);    //Motor läuft vorwärts für 3000 Millisekunden
  halt(1000);         //Motor hält an und wartet 1000 Millisekunden
}

void vorwaerts (int x) {
  digitalWrite(ml_1, HIGH);  //Motor dreht sich
  digitalWrite(ml_2, LOW);
  delay(x);
}

void halt (int x) {
  digitalWrite(ml_1, LOW);   //Motor hält an
  digitalWrite(ml_2, LOW);
  delay(x);
}
```

‚x' ist eine **lokale** Variable und nur gültig innerhalb des Unterprogramms.

Kapitel 20

3. Beispiel: Multiplikation zweier Werte (mit Rückgabe des Ergebnisses)

A093_Unterprogramme

```
int produkt;            //Die Variable 'produkt' wird als globale
                        //Integervariable festgelegt.
void setup() {
  Serial.begin(9600);
}

void loop() {

  produkt = multipliziere(7,9);
  Serial.println(produkt);

  Serial.println(multipliziere(6,4));
  delay(1000);

}

int multipliziere (int x, int y) {

  int ergebnis = x*y;     //'ergebnis' wird als lokale
                          //Integervariable festgelegt
  return ergebnis;
}
```

Die Funktion „multipliziere" wird aufgerufen und die beiden Werte werden mit übergeben.

Das Ergebnis wird in der Variablen ‚produkt' abgelegt.

Die Funktion „multipliziere" kann auch direkt aufgerufen werden.

Aufgabe A20.1:

- Schreibe den Sketch „A181_Motorsteuerung_(xx)" bzw. „A182_Motorsteuerung_(xx)" neu mit Hilfe von Unterprogrammen.
- Speichere den Sketch unter „A201_Motorsteuerung_Unterprg_(xx)" bzw. „A202_Motorsteuerung_Unterprg_(xx)" ab

Reflexoptokoppler

Neue Bauelemente: Reflexoptokoppler CN70

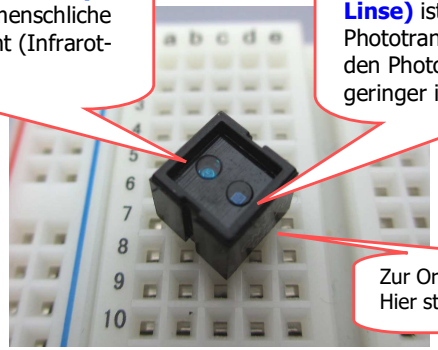
Neue Befehle: -

Ziel: Ein Kurzstrecken-Lichtschranke erstellen.

Jede Lichtschranke besteht aus einem Sender und einem Empfänger. Hier verwenden wir die Kurzstrecken-Reflexlichtschranke CNY70, bei der sich Sender und Empfänger im gleichen Gehäuse befinden - das ausgesendete Licht muss also reflektiert oder zurückgestreut werden, damit es den Empfänger trifft. Dazu genügt ein helles Objekt, das sich in wenigen mm Abstand vor der Reflexlichtschranke befindet.

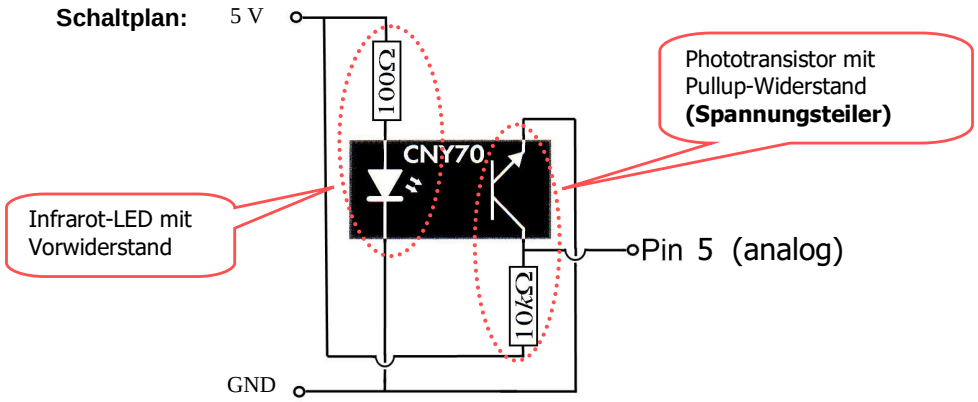
Der **Sender (hellblaue Linse)** ist eine LED, die für das menschliche Auge unsichtbares Licht (Infrarot-Licht) aussendet.

Der **Empfänger (dunkelblaue Linse)** ist ein sogenannter Phototransistor. Je mehr Licht auf den Phototransistor fällt, desto geringer ist sein Widerstand.

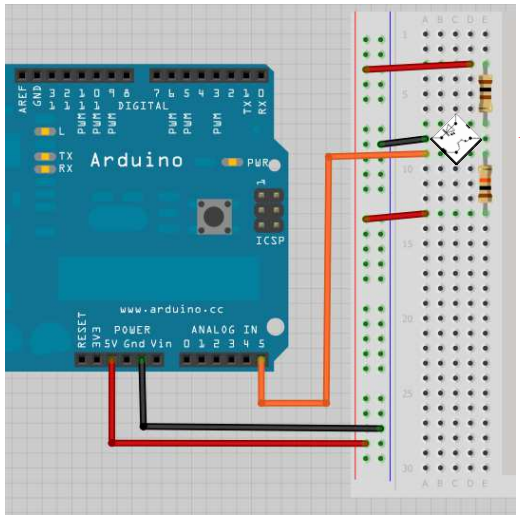


Zur Orientierung:
Hier steht die Beschriftung!

Schaltplan:



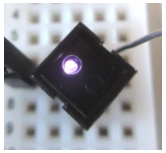
Kapitel 21



Da zwei der vier Anschlüsse mit **GND** verbunden werden müssen, kann man den CNY70 einfach diagonal auf das Steckbrett setzen.

Bemerkung:

Das Infrarot-Licht der sendenden Leuchtdiode ist zwar für das menschliche Auge nicht sichtbar, die meisten Handykameras und einfache Digitalkameras erkennen das Licht aber als weißes Leuchten. So könnt ihr leicht überprüfen, ob die LED auch leuchtet.



Aufgabe A21.1:

- Baue die Schaltung gemäß Schaltplan auf.
- Schreibe einen Sketch, bei dem die Werte vom analogen Pin 5 auf dem seriellen Monitor ausgegeben werden.
- Teste deine Lichtschranke, indem du einen Gegenstand vor den Reflexoptokoppler hälst.
- Speichere den Sketch unter „A211_Reflexoptokoppler_(xx)"

Aufgabe A21.2:

- Verändere den Pullup-Widerstand.
- Wie ändert sich damit die Empfindlichkeit der Reflexoptokopplers? Notiere deine Beobachtungen.

Aufgabe A21.3: (Alarmanlage)

- Ergänze die Schaltung um eine Leuchtdiode und einen Lautsprecher.
- Erweitere den Sketch so, dass ab einem bestimmten Wert die Leuchtdiode anfängt zu blinken und der Lautsprecher ein Warnsignal ausgibt.
- Speichere den Sketch unter „A213_Reflexoptokoppler_Alarm_(xx)"
- Zeichne den Schaltplan.

Servo

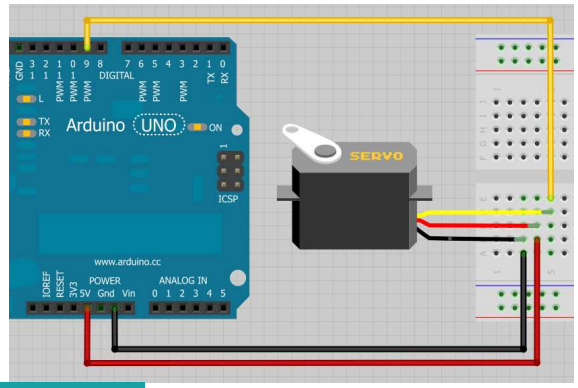
Neue Bauelemente: Servo

Neue Befehle: Servo-Befehle, map

Ziel: Einen Servo ansteuern

Aufgabe A22.1:

- Schließe den Servo an den Arduino an (s. Abb.)
- Übernehme den Sketch „A221_Servo" und starte ihn.



A221_Servo

```
#include <Servo.h>

Servo myservo;

void setup()
{
  myservo.attach(9);
}

void loop()
{
    myservo.write(0);
    delay(1000);

    myservo.write(90);
    delay(1000);
}
```

Die Servo-Library wird eingelesen.

Der angeschlossene Servo bekommt die Bezeichnung „**myservo**".
Diese Bezeichnung ist frei wählbar.

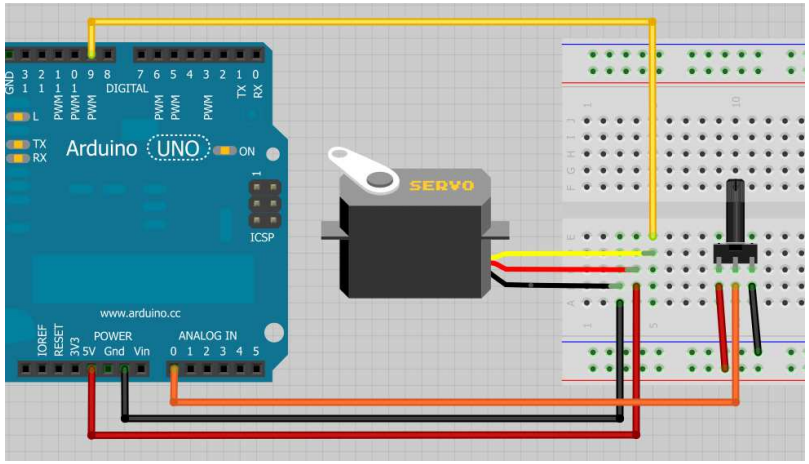
Der Servo wird an Pin 9 angeschlossen.

Die Gradzahl „0 Grad" wird auf den Servo übertragen.

Der Servo benötigt Zeit, um die neue Position anzufahren.
(Die Zeit ist abhängig vom Drehwinkel und vom Servotyp). Ausprobieren…

Aufgabe A22.2:

- Ergänze die Schaltung durch ein Potentiometer (10 kΩ)
- Übernehme den Sketch „A222_Servo" und starte ihn.
- Verändere die Stellung des Potentiometers und beachte den Servo.



A222_Servo

```
#include <Servo.h>

Servo myservo;

void setup()
{
  myservo.attach(9);
}

void loop()
{
  int val = analogRead(A0);
  val = map(val, 0, 1023, 0, 179);
  myservo.write(val);
  delay(15);
}
```

Das Potentiometer wird am analogen Pin A0 angeschlossen.

Umrechnung des ausgelesenen Werts auf die 180°-Skala

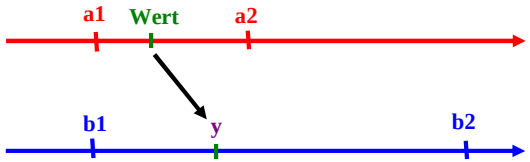
Die Gradzahl wird auf den Servo übertragen

Servo

Neue Befehle:

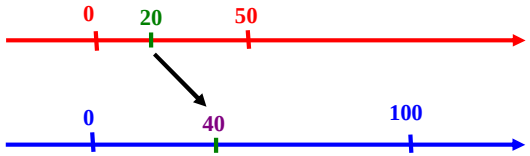
- **y = map(Wert, a1, a2, b1, b2)**

  Rechnet einen Wert von einem Intervall in ein zweites Intervall um:



Beispiel:

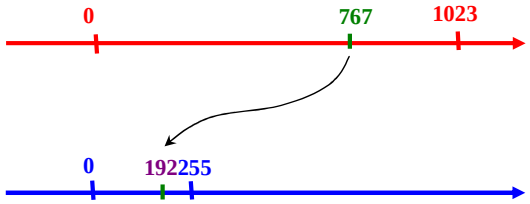
- **y = map(20, 0, 50, 0, 100)**



- Die Umrechnung erfolgt linear und das Ergebnis sind immer ganzzahlige Werte (integer).
  Um Dezimalwerte zu erhalten muss man die Intervalle z.B. verzehnfachen und anschließend das Ergebnis durch 10 dividieren.

- Dieser Befehl ist immer dann sinnvoll, wenn Skalen ineinander umgerechnet werden müssen:
  Mit dem Befehl analogRead() bekommt man Werte zwischen 0…1023. Für analogWrite() werden Werte zwischen 0…255 benötigt:

  y = map(767, 0, 1023, 0, 255) **Ergebnis: y = 192**



Kapitel 22

Aufgabe A22.3:

- Ändere den Sketch so ab, dass sich der Zeiger nur im Bereich 0°-90° bewegt.
- Auf dem seriellen Monitor sollen die Winkelwerte ausgegeben werden.
- Speichere den Sketch unter „A223_Servo_Serial_(xx)" ab.
- Zeichne den Schaltplan.

Aufgabe A22.4:

- Ergänze die Schaltung mit einer grünen und einer roten LED.
- Die grüne LED soll bei der Position „0°" leuchten, die rote LED bei der Position „90°". Bei allen anderen Positionen sollen die LEDs **nicht** leuchten.
- Ändere den Sketch und speichere ihn unter „A224_Servo_LED_(xx)" ab.
- Zeichne den Schaltplan.

Aufgabe A22.5:

- Baue eine Schaltung auf, bei der sich ein Servo zwischen 0° - 179° hin- und herbewegt.
- Schreibe den Sketch und speichere ihn unter „A225_Servo _(xx)" ab.

Aufgabe A22.6:

- Baue eine Schaltung auf, bei der ein Servo (mit Zeiger) die Lichtintensität (LDR) auf einer Helligkeitsskala (dunkel – hell) darstellt.
- Schreibe den Sketch und speichere ihn unter „A226_Servo_LDR_(xx)" ab.
- Zeichne den Schaltplan.

Info: Positionsregelung eines Servos

Bei Modellbauservos wird der Winkel der Ausgangswelle geregelt. Zur Ermittlung des Winkels befindet sich im Servoinnern ein Potentiometer, das mit der Ausgangswelle verbunden ist. Über dieses Potentiometer ermittelt die Servoelektronik den Ist-Winkel der Ausgangswelle. Dieser wird mit dem Soll-Winkel verglichen, der vom Mikrocontroller übermittelt wird. Bei einer Abweichung zwischen Ist- und Soll-Winkel, regelt die Elektronik über den Motor und das Getriebe den Winkel der Ausgangswelle nach.
Die Ausgangswelle kann eine Position zwischen 0° und 180° einnehmen.

Ultraschall

Neues Bauelement: Ultraschallsensor: HC-SR04

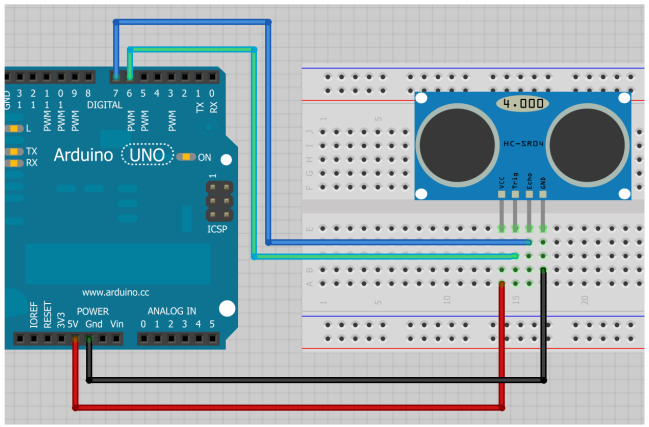
Neue Befehle: delayMicroseconds, pulsIn

Ziel: Ein 40 kHz Schallsignal wird eingesetzt um eine Distanz zu berechnen. Mit Hilfe des Ultraschallsensors können Abstände zu anderen Objekten erfasst werden.

Zielsicher wie eine Fledermaus ihre Beute jagt, so könnt ihr mit Hilfe des Ultraschallsensors Abstände zu anderen Objekten messen. Dies ist immer dann hilfreich, wenn ihr Kollisionen vermeiden oder z.B. eine definierte Strecke in einem Labyrinth zurücklegen müsst.

Aufgabe A23.1:

- Baue die Schaltung wie in der Abbildung auf.
- Übernehme den Sketch „A231_Ultraschall".
- Starte den Sketch und öffne den seriellen Monitor.
- Halte deine Hand vor den Sensor und verändere den Abstand. Beobachte die Abstandswerte im seriellen Monitor.



Aufgabe 23.2:

- Nehmt nun ein flaches Brett als Hindernis und untersucht die minimale Distanz und maximale Reichweite des Ultraschallsensors.
- Verwendet nun andere Hindernisse und bedient euch dazu bei dem ausgelegten Material. Warum haben Nachtfalter, eine Lieblingsbeute der Fledermäuse, ein so "franseliges" und "haariges" Aussehen?

Quelle : Wikipedia-"Eichenspinner (Lasiocampa quercus), ♀"

A231_Ultraschall

```
int echoPin = 7;
int trigPin = 6;
int entfernung;

void setup(){
  pinMode(echoPin,INPUT);
  pinMode(trigPin,OUTPUT);
  Serial.begin(9600);
}

void loop(){
  digitalWrite(trigPin,HIGH);
  delayMicroseconds(1000);
  digitalWrite(trigPin,LOW);
  entfernung = pulseIn(echoPin,HIGH)/58.0;
  Serial.print("Entfernung in cm: ");
  Serial.println(entfernung);
}
```

Hier wird ein Ultraschall-Signal ausgesendet.

Hier wird die Zeit bis zum Eintreffen des Echos gemessen und daraus die Entfernung berechnet.

Aufgabe 23.3: (Für Profis!)

- Der Ultrschallsensor erlaubt nur einen relativ schmalen Blick nach vorne. Kombiniere den Ultraschallsensor mit einem Servo und erweitere so sein Blickfeld auf 180°.
- Speichere den Sketch unter dem Namen „A233_Ultraschall_Servo_(xx)" ab.
- *Quelle: wikimedia-ship radar*

(vgl. Schiffsradar -Abb.rechts. Im Gegensatz zu Ultraschall handelt es sich beim RADAR - Radio Detection and Ranging- um elektromagnetische Wellen im Radiofrequenzbereich)

Teil 6: Anhang

Interrupt 24

Neues Bauelement:

Neue Befehle: attachInterrupt

Ziel: Durch die Verwendung eines Interrupts wird das „normale" Hauptprogramm unterbrochen und ein festgelegtes Unterprogramm aufgerufen. Danach wird das Hauptprogramm an der Stelle fortgesetzt, an der es unterbrochen wurde.

Aufgabe A24.1:

- Baue die Schaltung wie in der Abbildung auf.
  Der Einfachheit halber wird die LED auf dem Arduino-Board verwendet, die intern mit Pin 13 verbunden ist.
- Übernehme den Sketch und übertrage ihn auf den Arduino.
- Betätige den Taster und beobachte die LED auf dem Board.

Die Anweisung lautet allgemein:

attagchInterrupt(Interrupt, Unterprogramm , Modus);

- **Interrupt:**
  Der Arduino UNO hat zwei Interrupt-Möglichkeiten:

  Interrupt 0 an Pin2
  Interrupt 1 an Pin3

- **Unterprogramm:**
  Bezeichnung des Unterprogramms bzw. der Funktion, welche aufgerufen wird.
  Achtung: In diesem Unterprogramm funktionieren die Anweisungen delay() und millis() nicht.

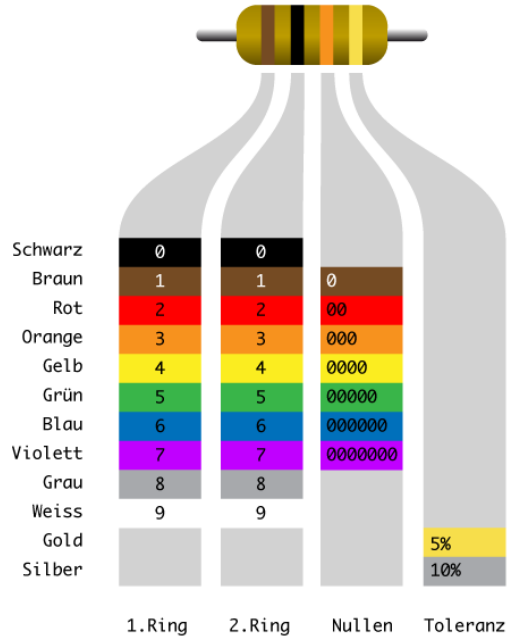
- **Modus:**
  Der Modus gibt an, wann der Interrupt ausgelöst wird.

LOW:	Der Wert des Pins beträgt 0
CHANGE:	Der Wert des Pins ändert sich
RISING:	Der Wert des Pins ändert sich von 0 auf 1
FALLING:	Der Wert des Pins ändert sich von 1 auf 0

interrupt

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(0, blink, FALLING);
}

void loop() {
  delay(1000);
}

void blink() {
  state = !state;
  digitalWrite(ledPin, state);
}
```

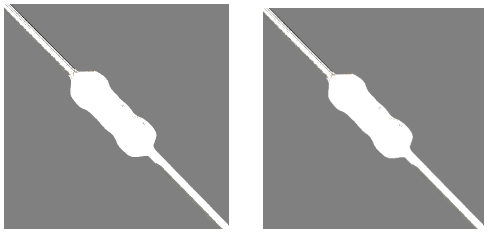
Bestimmung von Widerständen



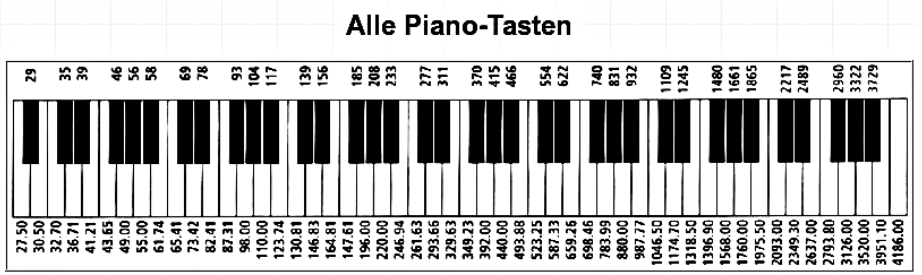
Aufgabe:

1. Bestimme den Wert der beiden Widerstände:

2. Zeichne die Ringe für die Widerstände mit R = 220 Ω und R = 47 kΩ ein.
   (Toleranz 10%)



Frequenztabelle



Frequenzen der gleichstufigen Stimmung – Tabelle											
Oktave	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
C / B#	16,352	32,703	65,406	130,813	**261,626**	523,251	1046,502	2093,005	4186,009	8372,018	16744,036
C# / Db	17,324	34,648	69,296	138,591	277,183	554,365	1108,731	2217,461	4434,922	8869,844	17739,688
D	18,354	36,708	73,416	146,832	293,665	587,330	1174,659	2349,318	4698,636	9397,273	18794,545
D# / Eb	19,445	38,891	77,782	155,563	311,127	622,254	1244,508	2489,016	4978,032	9956,063	19912,127
E / Fb	20,602	41,203	82,407	164,814	329,628	659,255	1318,510	2637,020	5274,041	10548,082	-
F / E#	21,827	43,654	87,307	174,614	349,228	698,456	1396,913	2793,826	5587,652	11175,303	-
F# / Gb	23,125	46,249	92,499	184,997	369,994	739,989	1479,978	2959,955	5919,911	11839,822	-
G	24,500	48,999	97,999	195,998	391,995	783,991	1567,982	3135,963	6271,927	12543,854	-
G# / Ab	25,957	51,913	103,826	207,652	415,305	830,609	1661,219	3322,438	6644,875	13289,750	-
A	27,500	55,000	110,000	220,000	**440,000**	880,000	1760,000	3520,000	7040,000	14080,000	-
A# / Bb	29,135	58,270	116,541	233,082	466,164	932,328	1864,655	3729,310	7458,620	14917,240	-
B / Cb	30,868	61,735	123,471	246,942	493,883	987,767	1975,533	3951,066	7902,133	15804,266	-