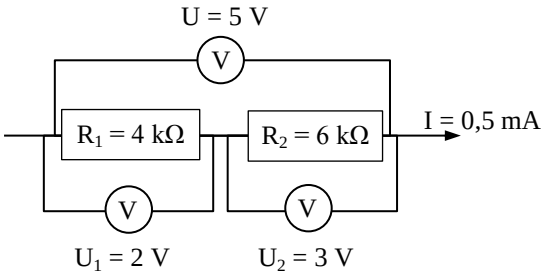
4.4 Potentiometer

4.4.1. Reihenschaltung

Der Strom durch beide Teilwiderstände ist gleich.
Die Spannungen und die Widerstände addieren sich:

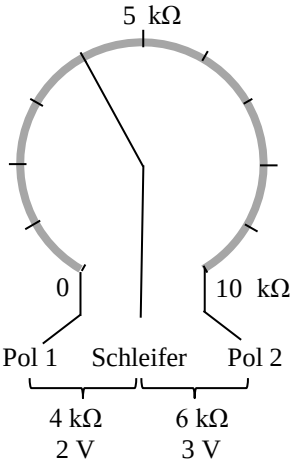
$$\left.\begin{array}{l} U_1 + U_2 = U \\ I_1 = I_2 = I \\ R_1 + R_2 = R \end{array}\right\} \Rightarrow \left.\begin{array}{l} U_1 = R_1 \cdot I \\ U_1 = R_2 \cdot I \end{array}\right\} \Rightarrow \frac{U_1}{U_2} = \frac{R_1}{R_2}$$

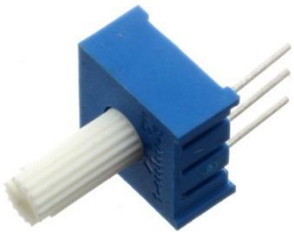


Übungen: Aufgaben zu Potentiometern Nr. 1

4.4.2. Potentiometer

Ein Potentiometer ist ein regelbarer **Spannungsteiler**, dessen kreisförmiger Widerstand von z.B. R = 10 kΩ über einen zeigerförmigen **Schleifer** auf einen beliebigen Wert R_1 verringert werden kann. Im Bild wird zwischen Pol 1 und Schleifer ein Teilwiderstand R_1 = 4 kΩ abgegriffen. Wegen des Ohmschen Gesetzes für die **Reihenschaltung** wird dann die zwischen Pol 1 und Schleifer anliegende Spannung U_1 von z.B. U = 5 V um das gleiche Verhältnis verringert wie der Widerstand:





$$U_1 = R_1 \cdot I = R_1 \cdot \frac{U}{R} = \frac{R_1}{R} \cdot U = \frac{4\,k\Omega}{10\,k\Omega} \cdot 5\,V = 2\,V$$

Die zwischen Pol 2 und Schleifer anliegende Spannung U_2 wird durch den Restwiderstand $R_2 = R - R_1$ bestimmt:

$$U_2 = R_2 \cdot I = \frac{R_2}{R} \cdot U = \frac{6\,k\Omega}{10\,k\Omega} \cdot 5\,V = 3\,V$$

Übungen: Aufgaben zu Potentiometern Nr. 2

4.4.3. Analog-Eingänge

Das Potentiometer ist ein einfacher **Lagesensor**, der die Lage bzw. den Drehwinkel des Knopfes stufenlos (d.h. **analog**) und verhältnistreu (d.h. **proportional**) in einen Widerstand bzw. eine Spannung umsetzt.
Um diese analoge Eingangsspannung elektronisch speichern und verarbeiten zu können, wird sie an den **analogen 10 bit-Eingängen** A0 – A5 des Arduino-Boards durch Vergleich mit einer über 2^{10} = 1024 Stufen anwachsenden Gegenspannung **proportional** einem digitalen Speicherwert (= Zahl der benötigten Spannungsstufen bis zum Ausgleich) zugeordnet. Die minimale Eingangsspannung 0 V ergibt den Wert 0 = $(00000\ 00000)_2$; die maximale Eingangsspannung 5 V entspricht dem Wert 1023 = $(11111\ 11111)_2$ und z.B. 2 V benötigt $\frac{2\,V}{5\,V} \cdot 1023 \approx 409$ Stufen = $(01100\ 11001)_2$. (siehe auch **4.0.1.3 Analog-Eingänge**).

Übungen: Aufgaben zu Potentiometern Nr. 3

4.4.4. Analog-PWM-Ausgänge

Nach der digitalen Speicherung und Verarbeitung muss der (evtl. veränderte) digitale Wert wieder in ein analoges Signal umgewandelt werden, um z.B. eine LED oder einen Motor steuern zu können. Ein einfaches Board wie der Arduino kann eine Ausgangsspannung von z.B. 2 V nicht direkt einstellen und reduziert stattdessen die **Dauer** des 5 V-Spannungspulses mittels 8 bit-**Pulsweitenmodulation** (PWM) **proportional** von jeweils 2^8 = 256 Impulsen auf $\frac{2\,V}{5\,V} \cdot 256 \approx 102$ Impulse. Für die restlichen 256 – 104 = 152 Impulse wird die Spannung abgeschaltet. Der Mittelwert ist dann $\frac{104 \cdot 5\,V + 152 \cdot 0\,V}{256} \approx 2$ V. (siehe auch **4.0.1.2 Analog-PWM-Ausgänge**)

Übungen: Aufgaben zu Potentiometern Nr. 4

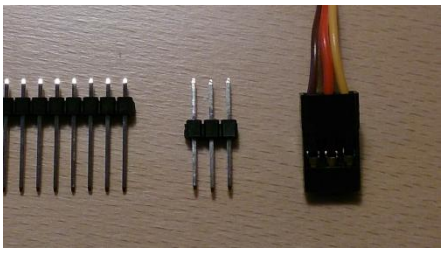
Das infolge des ständigen An- und Ausschaltens resultierende **Flackern** unserer LEDs ist zwar für das menschliche Auge durchaus wahrnehmbar, wird aber im Sehzentrum des Gehirns automatisch in den Mittelwert einer 40 % igen Helligkeit umgedeutet. Das längere Betrachten von nur scheinbar kontinuierlich leuchtenden aber in Wirklichkeit ständig neu sich aufbauenden Bildschirmen z.B. an elektronischen Geräten ist daher viel anstrengender als das Lesen einer Buchseite.

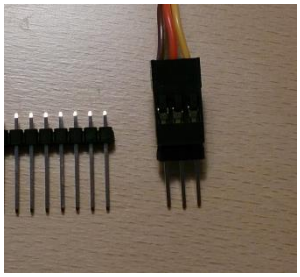
4.4.5. Servosteuerung

Wenn nicht das menschliche Gehirn selbst die eigentliche Digital-Analog-Umwandlung übernimmt, muss der analoge Verbraucher einen entsprechenden Umwandler (**Dekoder**) besitzen. Das häufigste Beispiel sind **Servos** (lat. servus = Diener) für die Steuerung von Autos, Schiffen und Flugzeugen. Es handelt sich um kleine Schritt- oder Linearmotoren, die die erforderlichen kleinen Steuerbewegungen der entsprechenden Ruder und Lenkungen sehr genau und teilweise mit großer Kraft ausführen.

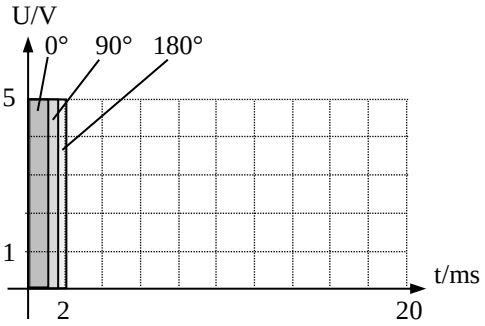
Unser kleiner **Modellbauservo** kann seinen Hebel (Ruderhorn) im Bereich von 0° - 180° einstellen. Er besitzt den gemeinsamen Standardanschluss der beiden Weltmarktführer Graupner und Robbe mit den Farben

- Schwarz (Masse → GND)
- Rot (Pluspol → 5 V)
- Orange oder Gelb oder Weiß (Steuerleitung → PWM).





Um den Stecker mit dem Steckbrett verbinden zu können, benötigt man einen kleinen **Adapter**. Dazu bricht man vorsichtig (evtl. mit Hilfe eines Messers) ein Stück mit drei Pins von der mitgelieferten **Pinleiste** ab, schiebt die drei Pins in der schwarze Plastikhülse ungefähr auf halbe Höhe herunter und steckt das Ganze in den Servostecker.

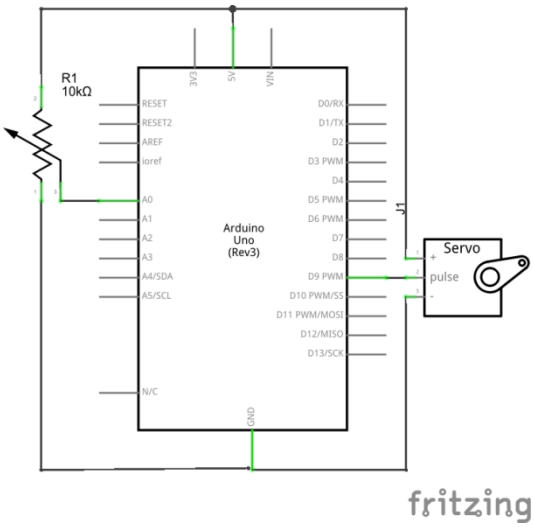


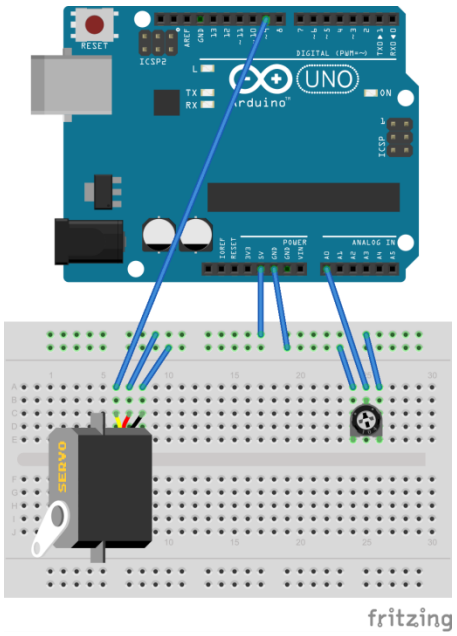
Der Dekoder erwartet in einer Periode von 20 ms eine **Einzelpuls** von 1 – 2 ms Dauer und ordnet dann einer Pulsdauer von T Millisekunden den Winkel $\alpha(T) = (T - 1)\cdot 180°$ zu. Man erhält also

- für T = 1 ms den rechten Anschlag bei 0°,
- für T = 1,5 ms die Neutralstellung bei 90° und
- für T = 2 ms den linken Anschlag bei 180°.

4.4.6. Aufbau

Wir steuern den Servo zunächst direkt mit dem Potentiometer. Um die Wirkung zu verdeutlichen, kann man den Servo mit Doppelklebeband auf der Montageplatte befestigen und einen Stift oder eine kleine Taschenlampe auf das Drehkreuz kleben. Wenn man das Potentiometer durch entsprechende Sensoren ersetzt, kann man den Servo auch dazu bringen, auf Licht, Schall oder Wärme zu reagieren. (siehe 4.4.5.)





4.4.7. Sketch

Um den PWM-Modus des Arduino-Boards auf den PWM-Modus des Servos umzustellen, benötigen wir einige Zusatzbefehle, die mit `#include<servo.h>` zu Beginn des Sketches bereitgestellt werden.
Zunächst wird unser `Servo1` mit der Anweisung `Servo Servo1` als **Objekt** vom Typ `Servo` deklariert. Beachte, dass hier **kein Zuweisungszeichen** steht! Ein solches Objekt ist ein fertiges Unterprogramm, das mit speziellen Befehlen gesteuert wird, welche dann jeweils den vorangestellten Objektnamen enthalten.
Im `setup` wird durch den Befehl `Servo1.attach(9)` in die Lage versetzt, Winkel zwischen 0° und 180° in die oben beschriebenen servofähigen Pulsweiten umzusetzen.
Die Umrechnung des Potentiometerwertes von 0 bis 1023 Stufen auf einen ganzzahligen Winkel von 0° bis 180° macht der Arduino-Compiler in einem Schritt: Potentiometerwert durch 1023 teilen, mit 179 multiplizieren und anschließend **abrunden**.

Wir benutzen dazu die Funktion `map(x, x0, x1, y0, y1)` = Abrunden[$a \cdot (x - x_0) + y_0$] mit der **Steigung** $a = \frac{y_1 - y_0}{x_1 - x_0}$,

die das Argument x aus dem Definitionsbereich [x_0; x_1] **linear** abbildet in den Wertebereich [y_0; y_1] und anschließend **abrundet**. Wir setzen ein:

x = Potentiometerwert
$x_0 = 0$
$x_1 = 1023$
$y_0 = 0$
$y_1 = 179$.

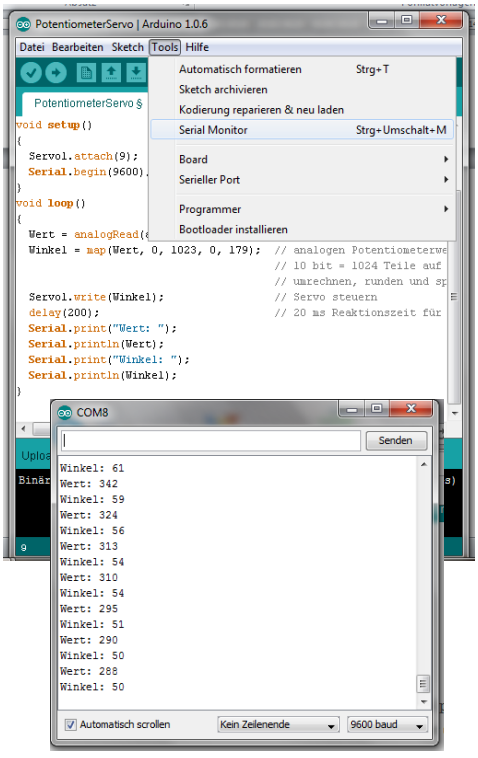
Mit `Servo1.write(Winkel)` kann dem Servo nun die Schwenkposition direkt in Grad mitgeteilt werden.

```
#include <Servo.h>                        // Befehlssatz für Servosteuerung laden
Servo Servo1;                             // Servo1 deklarieren
int analogPin = A0;                       // analoger Eingang auf A0
int Wert;                                 // Variable für Potentiometerwert
int Winkel;                               // Variable für Winkel des Servoarms
void setup()
{
  Servo1.attach(9);                       // Pin 9 auf Servosteuerung umstellen
}
void loop()
{
  Wert = analogRead(analogPin);           // Potentiometerwert einlesen
  Winkel = map(Wert, 0, 1023, 0, 179);    // analogen Potentiometerwert von
                                          // 10 bit = 1024 Teile auf 0 - 179° Winkel
                                          // umrechnen, abrunden und speichern
  Servo1.write(Winkel);                   // Servo steuern
  delay(200);                             // 20 ms Reaktionszeit für Servo
}
```

Übungen: Aufgaben zu Potentiometern Nr. 5

4.4.8. Debugging mit dem Serial Monitor

Laufzeitfehler sind logische, Rechen- oder Einsetzungsfehler z.B. bei der map-Funktion, die werden vom Compiler nicht bemerkt werden, weil formal alles richtig ist. Der fehlerhafte Sketch wird kommentarlos geladen und man sieht nur, dass das Board nicht das tut, was es tun soll. Für die Erkennung und Beseitigung solcher **Bugs** (engl. Käfer, vgl. Sand im Getriebe) kann man sich die Inhalte der Variablen auf dem Computerbildschirm ausgeben lassen. Man reserviert im `setup` mit dem Befehl `Serial.begin(9600);` einen einfachen (**seriellen**) Ausgang mit der Taktung 9600 Hz (**baudrate**) über den usb-Anschluss zurück an den Computer. In der Endlosschleife `loop` fügt man dann einfach den Befehl `Serial.print(Variable);` bzw. `Serial.println(Variable);` hinzu. Das an den print-Befehl angefügte Kürzel ln steht für **line** bzw. neue Zeile: Mit ln erscheinen die Werte untereinander; ohne ln werden sie hintereinander gereiht. In der Arduino-IDE kann man dann die Werte der Variable für jeden Durchlauf der Endlosschleife im **SerialMonitor** verfolgen und auf diese Weise (hoffentlich) Aufschluss über logische Fehler gewinnen.



Übungen: Aufgaben zu Potentiometern Nr. 6