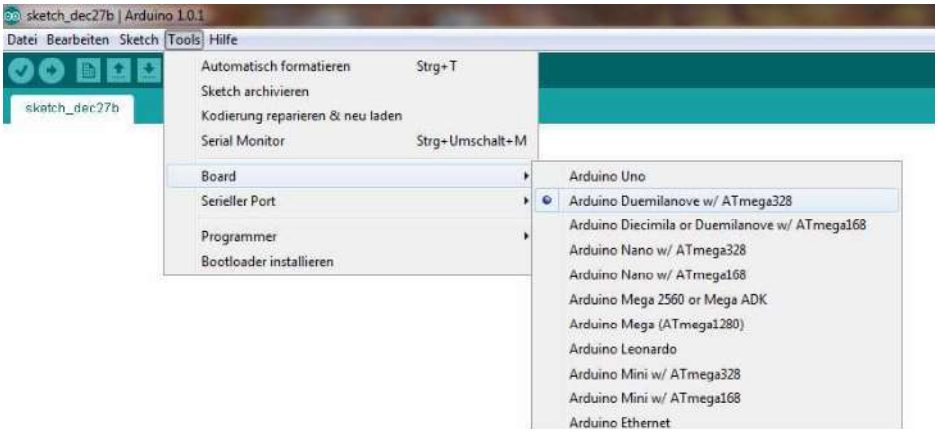
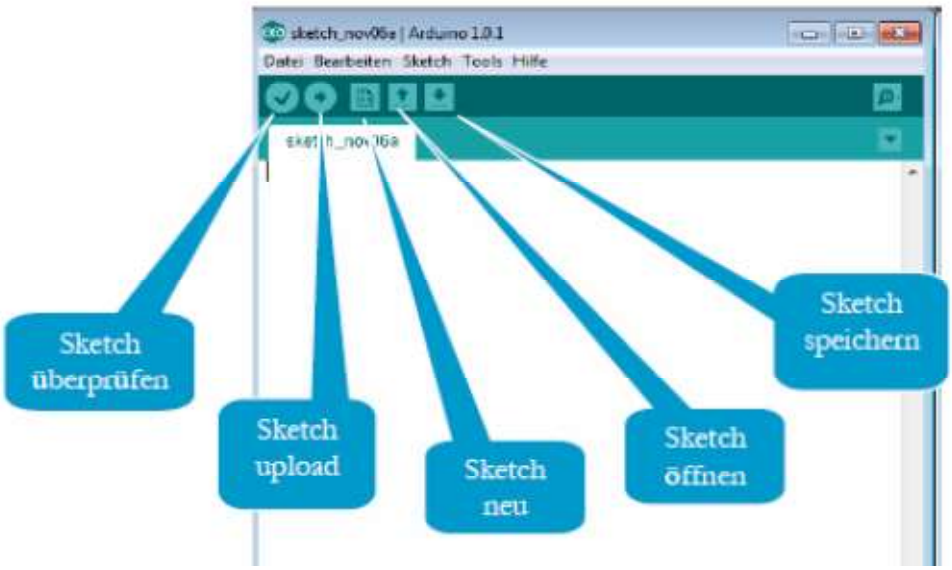
EINFÜHRUNG

Die Wahl des richtigen Arduino (Arduino Uno)



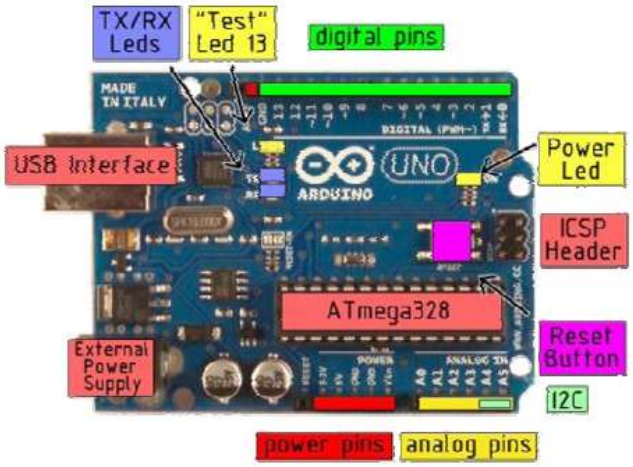
Auswahl des seriellen Ports:

Die Software IDE



Kapitel A0

Der Grundaufbau eines Mikrocontrollers:



Ein Mikrocontroller ist sinngemäß ein kleiner Computer. Er verfügt über eine Recheneinheit, über verschiedene Speicher und Schnittstellen. Insbesondere sind viele seiner Anschlüsse (Pins) zum Steuern elektronischer Schaltungen vorgesehen. Wir unterscheiden digitale und analoge Pins.

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

· **Analoge Pins:**
Ihr Spannungswert kann jeden Wert **zwischen** 0V und +5V annehmen.

Über die Pins können auch Spannungswerte eingelesen werden. Damit kann man z.B. überprufen, ob ein Schalter geoffnet 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).

Der Grundaufbau der Software:

Ein Programm wird allgemein mit „**Sketch**“ bezeichnet. Alle Sketches haben den gleichen Aufbau:

```
void setup()
{
  anweisungen;
}

void loop()
{
  anweisungen;
}
```

Dieser Programmteil wird **einmal** durchlaufen!

(Setzen von Pinmode oder Start der seriellen Kommunikation)

Dieser Programmteil wird **unendlich oft** durchlaufen!

Hier befindet sich das eigentliche Programm, der so genannte Programmcode.

Ein erster Sketch:

Blink

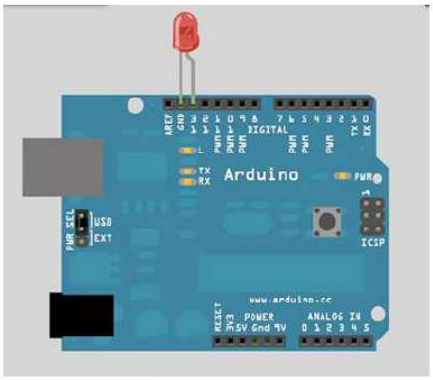
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```



int led = 13; Die Variable led wird mit dem (Integer-) Wert 13 belegt.

pinMode(led, OUTPUT); Da die Variable led mit dem Wert 13 belegt wurde, wird PIN13 als OUTPUT deklariert, d.h. er kann die Werte HIGH (5V) und LOW (0V) annehmen und damit eine Leuchtdiode ansteuern.

digitalWrite(led, HIGH); PIN13 wird HIGH (5V) gesetzt.

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

Aufgabe A0.1:

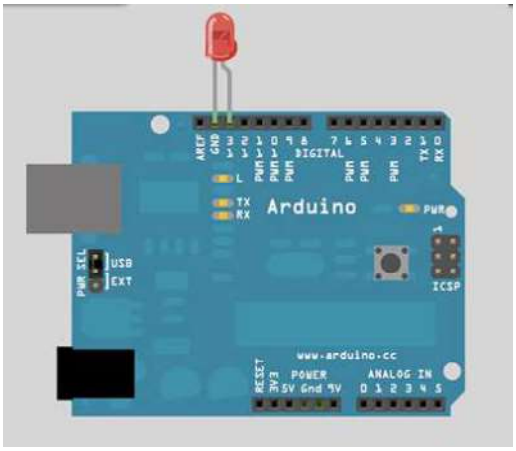
- Ändere den Sketch „Blink“ so ab, dass die LED 3 Sekunden lang leuchtet und 1 Sekunde lang dunkel bleibt.
- Speichere den neuen Sketch im Ordner „Sketches“unter „A01_Blink_(xx)“ ab. xx = Dein Name.

Aufgabe A0.2:

- Ändere den Sketch „Blink“ so ab, dass die LED in einer Sekunde 2x aufblinkt.
- Speichere den neuen Sketch im Ordner „Sketches“unter „A02_Blink_(xx)“ ab. xx = Dein Name.

Deine Notizen:

Kapitel A1

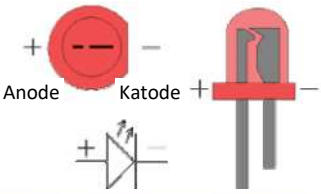


Neue Bauelemente: LED, Steckplatine, Widerstand 220 Ω
Neue Befehle: -
Eine bzw. mehrere LED zum Leuchten bringen

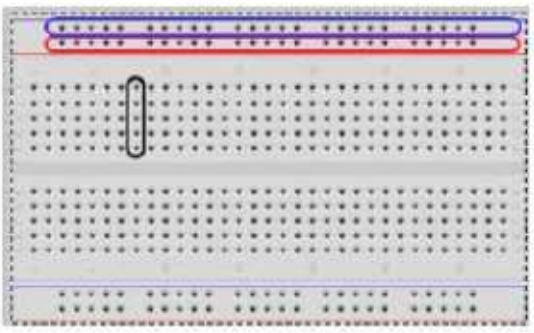
Aufgabe A1.1:

- Schließe, wie im Bild zu sehen, eine rote LED am Pin 13 an. Achte auf die Polung (siehe Info: Katode an GND)
- Öffne im Ordner „Sketches“ den Sketch „A11_Blink“
- Übertrage den Sketch auf den Arduino und beobachte die eingebaute LED.

Info: LED
Auf dem Arduino-Board befindet sich an Pin 13 eine eingebaute LED. Deshalb ist der an allen anderen Pins notwendige Vorwiderstand von 220 Ohm hier nicht notwendig.



Info: Steckplatine
Die seitlichen Kontakte (blau bzw. rot umrandet) sind in der Regel über die ganze Länge leitend miteinander verbunden. Ansonsten sind die 5er-Reihen (schwarz umrandet) leitend miteinander verbunden.



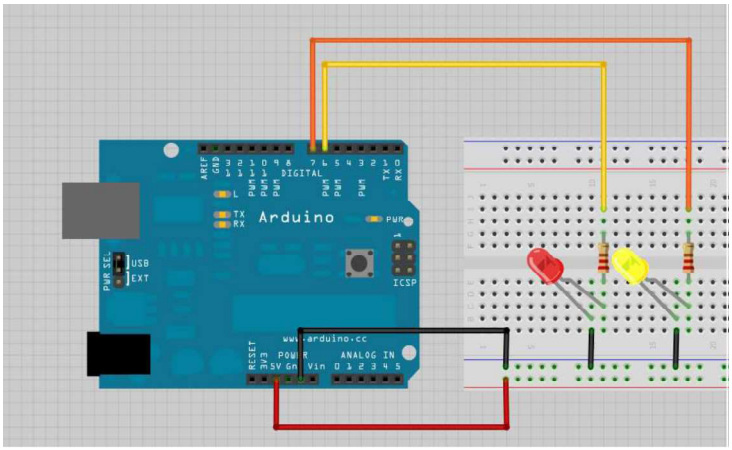
Aufgabe A1.2 :

- Ändere den Sketch „A11_Blink“ so ab, dass die LED 3 Sekunden lang leuchtet und 1 Sekunde lang dunkel bleibt.
- Speichere den neuen Sketch im Ordner „Sketches“
- unter „A12_Blink2_(xx)“ ab. xx = Dein Name.

Aufgabe A1.3 :

- Baue eine Schaltung auf, bei der abwechselnd eine rote und gelbe LED blinkt.
- Verwende dazu die Steckplatine.
- Vorwiderstande nicht vergessen!
- Ändere den Sketch „A11_Blink“ entsprechend ab.
- Speichere den neuen Sketch unter „A13_Blink_2LED_(xx)“ ab.
- Zeichne den Schaltplan.

Steckbeispiel für 2 LEDs:



Aufgabe A1.4:

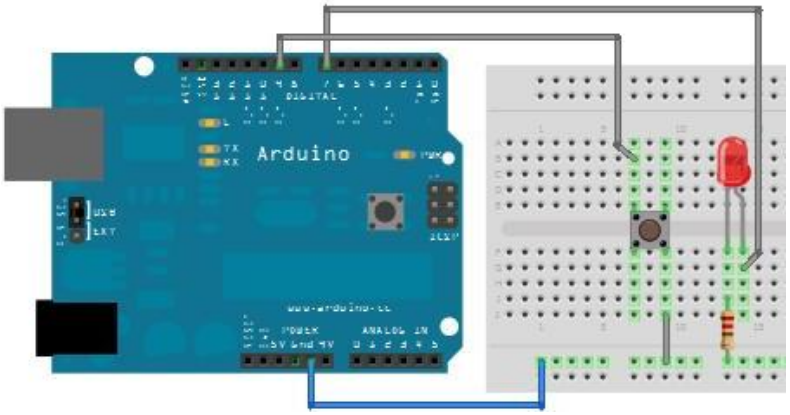
- Baue eine Ampelschaltung aus einer roten, gelben und grünen LED auf.
- Schreibe den Sketch.
- Übertrage das Programm.
- Speichere den Sketch unter dem Namen „A14_Ampel_(xx)“ ab.

Kapitel A2

Neue Bauelemente: Taster
Neue Befehle:

- **digitalRead();**
- **if ();**

Ziel: Eine LED mit einem Taster an- und ausschalten, Ampelsteuerung.



Aufgabe A2.1:

- Baue die Schaltung wie in der Abbildung auf, achte auf den Ort des Vorwiderstands 220 Ω für die LED. (Katode am Widerstand)
- Öffne den Sketch „A21_LED_mit_Schalter“.
- Starte den Sketch und verändere die Schalterstellung und beobachte die LED.

Neue Befehle:

- **const –** definiert einen konstanten (unveränderlichen) Wert für das ganze Programm
- **digitalRead(taste) -** Liest den Zustand (HIGH oder LOW) eines Pins aus. (Hier Pin 9)
- **if (else) -** Mit diesem Befehl wird abgefragt, ob eine Aussage wahr ist. Ist sie das nicht, geht das Programm zu dem Teil **else** weiter.

Info: Vergleichende Operatoren für if (else) oder do while

```
x == y     // x ist gleich wie y
x != y     // x ist nicht gleich wie y
x <  y     // x ist weniger als y
x >  y     // x ist mehr als y
x <= y     // x ist weniger oder gleich wie y
x >= y     // x ist größer oder gleich wie y
```

Aufgabe A2.2:

- Schließe eine weitere LED mit Vorwiderstand 220 Ω an.
- Ändere den Sketch so ab, dass bei gedrücktem Schalter LED1 und bei geöffnetem Schalter LED2 leuchtet.
- Speichere den Sketch unter dem Namen „A22_2LED_mit_Taster_(xx)“.
- Zeichne den Schaltplan!

Aufgabe A2.3:

- Schließe einen weiteren Taster an.
- Ändere den Sketch so ab, dass Taster 1 die LED ein- und Taster 2 ausschaltet. Benutze dazu zweimal

  do {}
  while (digitalRead(Taste1) == HIGH);

  als Warteschleife in der nichts passiert bis die Taste betätigt wird.
- Speichere den Sketch unter dem Namen „A23_LED_mit_2Taster_(xx)“.

Aufgabe A2.4:

- Baue den Sketch A2.2 mit 5 LED zur Fußgängerampel aus. Die Tastenbetätigung soll dem Fußgänger Grün geben. Beachte die verschiedenen Ampelphasen. Erstelle auf Papier eine Tabelle mit den 5 LED (Spalten) und den 7 Lichtkombinationen (Zeilen).
- Speichere den Sketch unter dem Namen „A24_Ampel_mit_Taster_(xx)“.

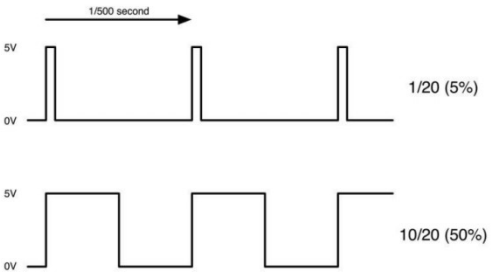
Kapitel A3

Neue Bauelemente: RGB-LED
Neue Befehle:

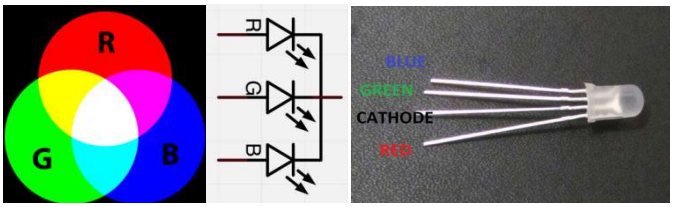
- **analogWrite(digitalPin, wet);**

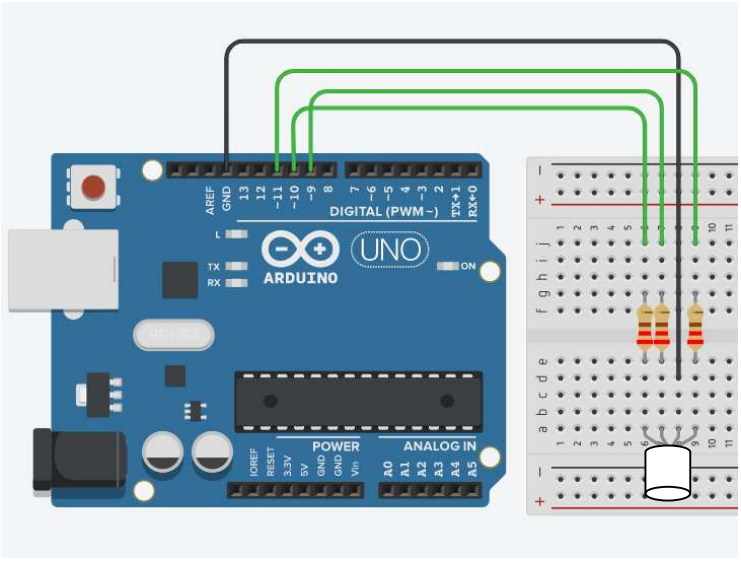
Ziel: eine RGB-LED dimmen
Arbeit mit Variablen

Info: PWM-Ausgänge ~
6 Anschlüsse (3,5,6,9,10 und 11) des Arduino können eine **P**uls-**W**eiten-**M**odulation in 256 Stufen erzeugen. Der Port wird alle 2ms für eine bestimmte Zeit (wert) eingeschaltet. Diese Funktion wird mit **analogWrite(digitalPin,wert);** aufgerufen.



Info: RGB-LED
In einem Plastik-Körper sind drei einzelne LED-Chips verbaut. Sie besitzen eine gemeinsame Kathode. Jede LED benötigt ihren eigenen Vorwiderstand! Das Dimmen der 3 Farben in je 256 Stufen erzeugt bis zu 65535 Nuancen.





Aufgabe A3.1: digitale Ansteuerung

- Baue die Schaltung wie in der Abbildung auf, achte auf den richtigen Anschluss der Kathode und die drei 220 Ohm Vorwiderstände.
- Erstelle einen Sketch „A31_RGB-LED“ analog der Ampelsteuerung. Steuere die 3 Farben mit HIGH und LOW einzeln an. Mische auch die Farben.

Aufgabe A3.2: analoge Ansteuerung

- Ändere den Sketch A31 so ab, dass eine einzelne LED mit **analogWrite();** angesteuert wird. Die LED sollte sich jetzt mit Werten von 0 bis 255 in ihrer Helligkeit dimmen lassen.
- Speichere den Sketch „A32_RGB-LED_analog“ ab.

Aufgabe A3.3: gleitender Farbübergang

- Mit einer Zählschleife kann man die Farbe „faden“. An den Endwerten von 0 und 255 muss der Zähler die Richtung ändern. Nutze dazu die **if ()** -Funktion, um die Schrittweite +/- anzupassen. Du benötigst 2 int-Variablen pro Farbe, eine für den Dimmwert und eine für die Schrittweite/-richtung. Zwischen den Schritten nutzt du **delay(10);**
- Kopiere das Faden für die anderen beiden Farben. Die LED bekommt nun eine Mood-Funktion. Gib jeder Farbe einen anderen Startwert/-richtung.
- Speichere den Sketch unter „A33_RGB-LED_mood“ ab.

Kapitel A4

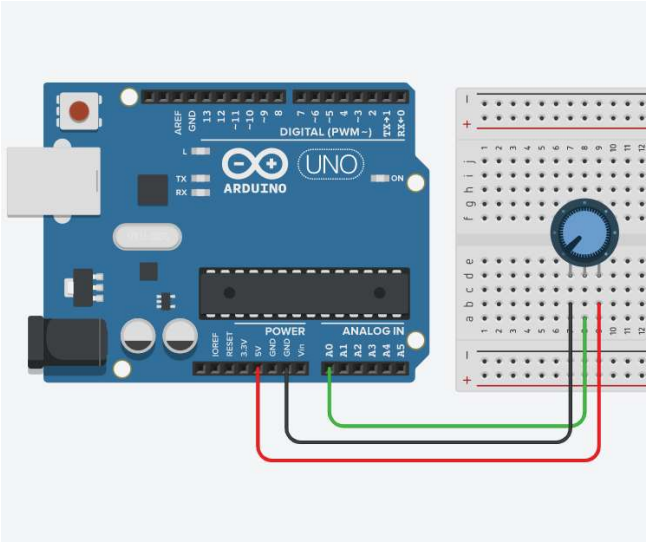
Neue Bauelemente: Potentiometer
Neue Befehle:

- **u_{Int} = analogRead(digitalPin);**
- **Serial.begin(Baudrate);**
- **Serial.println(wert);**

Ziel: eine Spannung messen und eine LED dimmen
Die Serial-Bibliothek nutzen

Info: Serial Bibliothek
Die in der Arduino-IDE bereits enthaltene Bibliothek organisiert eine Datenver-Bindung zwischen Arduino und PC über den USB-Anschluss. Ein serieller Monitor zeigt die ausgetauschten Daten. Die Baudrate (Übertragungsgeschwindigkeit) muss mit **Serial.begin(9600);** im Setup gleich dem seriellen Monitor „9600Bd“ eingestellt sein. Mit **Serial.println();** werden Werte gesendet.

Info: Analog Eingänge
Der Arduino besitzt 6 Eingänge A0..A5, die gleichzeitig analog Spannungen von 0 bis 5 Volt messen können. Er kann bis zu 10.000 Messungen pro Sekunde ausführen. Die Wandlung ergibt Werte von 0 bis 1023 (Integer). Mit der Funktion **analogRead(Port)**; werden die Werte gelesen.



Aufgabe A4.1:

- Baue die Schaltung wie in der Abbildung auf, achte darauf, dass der Mittelanschluss des Potentiometers mit dem Analoganschluss A0 verbunden wird.
- Schreibe einen Sketch, der die Messwerte alle 0,5 sec an den PC sendet
- Stelle am seriellen Monitor rechts unten die Baudrate auf 9600 ein.
- Teste verschiedene Potentiometerwerte und die Grenzen 0 bzw. 1023!
- Teste höhere Übertragungsraten z.B. 115200, in dem du die Werte im Monitor und in **Seriel.begin(115200);** änderst.
- Speichere den Sketch unter „A41_Poti“ ab.

Aufgabe A4.2:

- Ergänze auf dem Breadboard eine einzelne LED mit 220 Ohm Vorwiderstand analog der Experimente der letzten Wochen, die an einem PWM-Port angeschlossen wird.
- Ändere den Sketch A4.1 so ab, dass die LED ihre Helligkeit mit der Potentiometerstellung ändert. Nutze dazu den dir bekannten **analogWrite(Port, Wert);** - Befehl. Beachte, dass der Wert maximal 255 erreichen darf.
- Speichere den Sketch unter „A42_Poti_mit LED“ ab.

Das Einlesen und Auswerten von Sensorwerten (hier Potentiometer) ist für technische Anwendungen wichtigste Grundlage. Ein Schwellwert, z.B. einen bestimmten Spannungswert (Überlast), lässt sich mit dem **if ()** – Befehl abfragen.

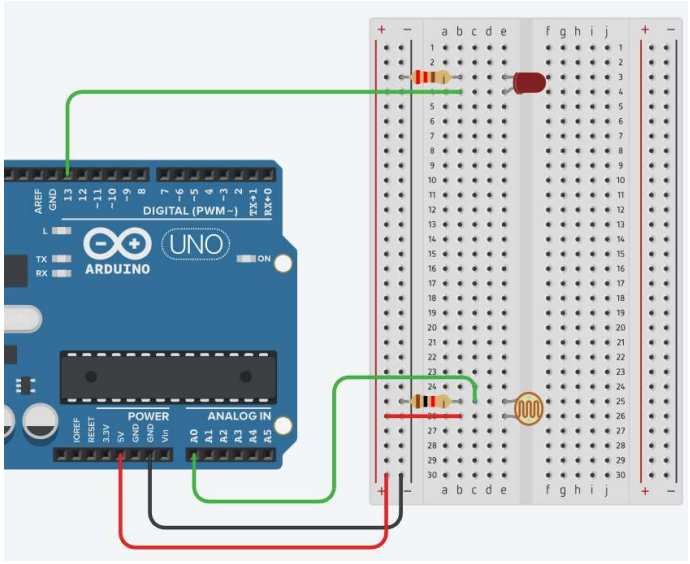
Aufgabe A4.3:

- Die LED soll nicht mehr dimmen, sondern ab der Hälfte des Potentiometers aufleuchten und in der anderen Hälfte erlöschen. Nutze dazu den **if () -** Befehl, um den mit **analogRead()** gelesenen Wert (0..1023) auszuwerten. Definiere selbst eine Spannungsschwelle.
- Speichere den Sketch unter „A43_Poti_Schwelle“ ab.

Zusatz: Lichtband mit 5 LEDs erzeugen

Kapitel A5

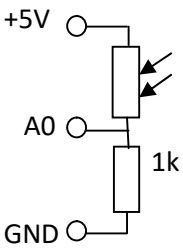
Neue Bauelemente: LDR (Lichtsensor)
Neue Befehle: -
Ziel: Eine LED soll leuchten, wenn ein LDR abgedunkelt wird.



Aufgabe A5.1 LED blinkt abhängig der Helligkeit:

- Baue die Schaltung wie in der Abbildung auf. Beachte den 1kΩ und den 220Ω Widerstand.
- Schreibe einen Sketch „A51_LDR-Sensor“, nutze dazu den Blink-Sketch vom Potentiometer!
- Dunkle mit deiner Hand den LDR ab und beobachte dabei die Rote LED.
- Reduziere die Variable mess um 300 durch Subtraktion. Welche Wirkung hat das Abdunkeln auf die delay()-Funktion? Was bedeutet das für den Widerstandswert des LDR?

Info: LDR
Ein Fotowiderstand ist ein Halbleiter, dessen Widerstandswert lichtabhängig ist. Fällt das Licht (Photonen) auf das lichtempfindliche Halbleitermaterial, dann werden die Elektronen aus ihren Kristallen herausgelöst. Der LDR wird leitfähiger, das heißt, sein Widerstandswert wird kleiner. Je mehr Licht auf das Bauteil fällt, desto kleiner wird der Widerstand. Innerhalb des Spannungsteilers steigt der Spannungswert am Anschluss A0 je mehr Licht auf den LDR fällt. Der Wertebereich liegt bei ca. 100Ω … 100kΩ.



Aufgabe A5.2 LED leuchtet abhängig der Helligkeit auf:

- Ändere den Sketch so ab (wie in Kapitel 4 - mit Befehl if () else, dass die LED ab einem **Schwellwert** von 500 anfängt zu leuchten.
- Speichere den Sketch unter dem Namen „A52_LDR-Sensor“ ab.
- Welche technischen Anwendungen fallen dir ein?

Aufgabe A5.3 Übung:

- Füge eine grüne LED mit Vorwiderstand hinzu.
- Ändere den Sketch so ab, dass nur die rote LED oberhalb eines bestimmten Schwellwerts leuchtet und die grüne LED nur unterhalb des Schwellwerts.
- Speichere den Sketch unter dem Namen „A53_LDR-Sensor“ ab.

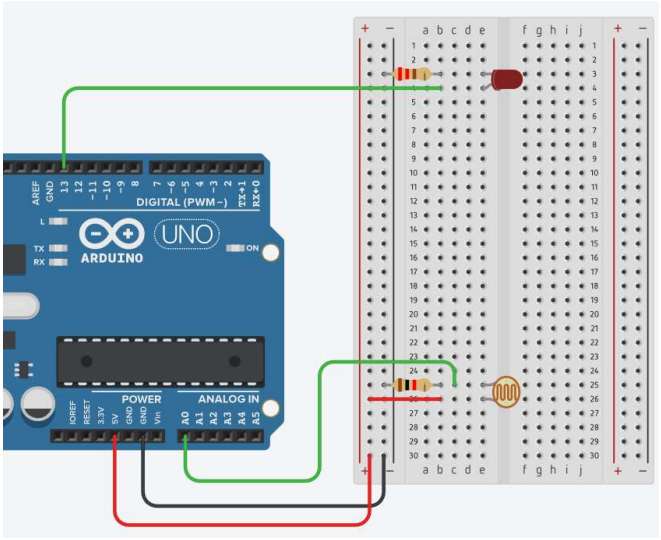
Aufgabe A5.4 Ein-Aus-Lichtschalter mit 2 LDR:

- Ergänze eine 2. LDR-Schaltung für den Anschluss A1!
- Erweitere den Sketch „LDR-Sensor“ um eine zweite Messwerterfassung mess2 und eine Variable für den Schaltzustand der Leuchtdiode!
- Ergänze den Sketch so, dass das Abdunkeln des einen LDR die LED ein (true) und das Abdunkeln des Anderen die LED ausschaltet (false). Gibt es technische Anwendungen dafür?
- Speichere den Sketch unter „A54_LDR-ein-aus“!

```
int mess1 = 0;
int mess2 = 0;
boolean licht = false;
…
Serial.begin (9600);
}
void loop() {
  mess1 = analogRead (A0);
  mess2 = analogRead (A1);
  Serial.println (mess1 + ” “ + mess2);
  …
```

Kapitel A6 Messen mit Arduino

Neue Bauelemente: NTC (Thermistor)
Neue Befehle: map(x,min,max,min2,max2);
Ziel: mit dem Arduino elektrische Größen messen, berechnen und anzeigen



Aufgabe A6.1 Spannungsmessung:

- Baue die Schaltung wie in der Abbildung auf. Beachte den 1kΩ Widerstand!
- Schreibe einen Sketch, in dem die gemessene Spannung alle 100ms an den seriellen Port ausgegeben wird.
- Definiere eine `int`-Variable `u1` und ergänze die Zeile `u1 = map(u1,0,1023,0,5000);` Die Spannung lässt sich nun in mV darstellen.
- Speichere den Sketch unter „A61_U-Messung“ ab!

Info:
Der map(mess,min,max,min2,max2); Befehl passt den Wertebereich einer Variablen an einen neuen Wertebereich an.

Aufgabe A6.2 Widerstandsberechnung:
Der Arduino kann als intelligentes Messgerät verwendet werden. Unser Wissen über den Spannungsteiler mit 2 Widerständen können wir nutzen, um einen unbekannten Widerstandswert mithilfe der Spannungsmessung zu berechnen und direkt auszugeben.

- Definiere eine `float`-Variable r2 und ergänze die Zeile `r2 = (5000.00/u1)*1000 - 1000;` für die Widerstandsberechnung. Unser Festwiderstand (R1) beträgt 1000 Ohm. Wichtig: Die Eingabe von 5000.00 zwingt den Arduino zu einer Fließkomma-Berechnung.
- Teste verschiedene Widerstände aus deinem Baukasten anstelle des LDR! Du kannst auch das Potentiometer mit einem äußeren und dem Mittelanschluss anschließen.
- Speichere den Sketch unter „A62_R-Messung“ ab!

Info:
Die Verhältnisse am Spannungsteiler:
$U_G/U_1 = R_G/R_1 \quad R_G = R_1+R_2$
$R_G=U_G/U_1*R_1$
$R_2=U_G/U_1*R_1 -R_1$

```
// A6 LDR_Mess
int u1 = 0;
float r2 = 0;

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

void loop(){
  u1 = analogRead(A0);
  u1 = map(u1,0,1023,0,5000);
  r2 = (5000.00/u1)*1000 - 1000;
  Serial.print(u1);
  Serial.print(" mV ");
  Serial.print(r2);
  Serial.println(" Ohm");
  delay(100);
}
```

Aufgabe A6.3 Wärmemessung mit NTC (Thermistor):
Neben dem Lichtsensor gibt es auch einen Wärmesensor, mit dem nicht nur die Umgebungstemperatur gemessen, sondern auch z.B. die Temperatur an einem Transistor oder einem IC überwacht werden kann.

- Ersetze den Widerstand durch den NTC!
- Beobachte den Widerstandswert bei Erwärmung zwischen deinen Fingern!
- Versuche eine Berechnungsformel zu finden, die die Anzeige im Bereich 20°C..37°C im seriellen Monitor ermöglicht!
- Wähle eine Schwelltemperatur, an der die LED zu leuchten beginnt!
- Speichere den Sketch unter „A63_T-Messung“ ab!