

Lehren mit dem All

Smarthome im Weltall Die Zukunft des Wohnens zwischen den Planeten

Programmierung eines Smarthomes

Was ist ein Smarthome?

- smartes, intelligentes Haus
- Automatisierung von Funktionen wie z.B. Beleuchtung, Heizung, Lüftung und Sicherheit
- Vernetzung von Systemen und Geräten
- Ziele: Komfort, Energieeffizienz, Sicherheit und Personalisierung im Alltag

- Sensoren z.B.:
 - Bewegungsmelder
 - Helligkeitssensoren
 - Temperatursensoren
 - Rauchmelder
 - CO₂-Sensoren
 - Überwachungskameras
 - eigene Wetterstation
- Aktoren z.B.:
 - automatisierte Rollläden, Markisen, Garagentore
 - elektrisch schaltbare Steckdosen
 - automatisierte Belüftung
 - Alarmanlage
 - automatisierte Fenster und Türen

- Steuerungssysteme
 - zentrale Steuerung mittels Tablets
 - Sprachassistenten wie z.B. Alexa oder Google Assistant
 - Apps, auch per Fernzugriff möglich

- Integration von Photovoltaikanlagen
- effizientes und nachhaltiges Energiemanagement
- Künstliche Intelligenz und das Erlernen persönlicher Gewohnheiten
- Integration von E-Autos
- Vernetzung von immer mehr Geräten, z.B. Kühlschränke, die Lebensmittel bestellen

- hohe Anschaffungskosten
- Abhängigkeit von der Technologie und möglichen Ausfällen
- Komplexität der Installation und Bedienung
- Datenschutzbedenken
- Sicherheitsbedenken

- Smarthome-Technologien machen unser Leben komfortabler, effizienter und sicherer
- sind bei uns aber auf der Erde nicht zwingend notwendig
- im Weltall sind wir aber darauf angewiesen
 - extreme Bedingungen: Temperaturen, Vakuum, Strahlung, keine Atmosphäre
 - kein wirklicher Lebensraum
 - Ressourcenmanagement überlebenswichtig, kein natürlicher Sauerstoff, kein Wasser
 - Automatisierung zwingend notwendig!

NASA-Projekt: Artemis Base Camp

- Basis am Mond-Südpol
- langfristige Präsenz auf dem Mond & Vorbereitung auf Mars-Missionen
- Behausung für bis zu vier Astronauten
- smarte Technologien und Automatisierungen nötig

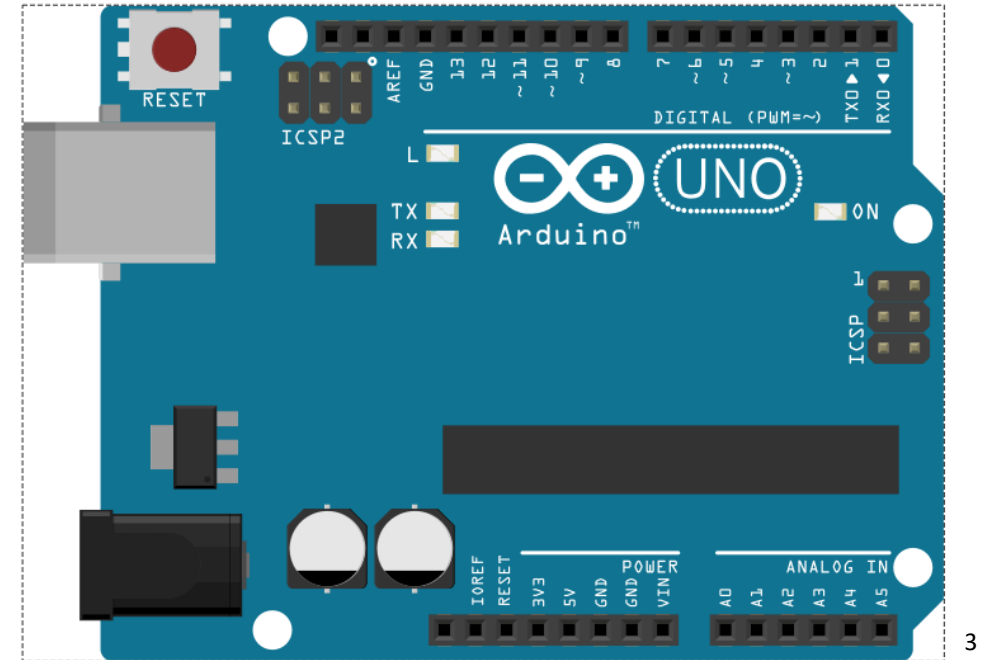
Nötige Automatisierungen auf Raumstationen:

- Klimaregulierung, da Lüften nicht möglich
- Sauerstoffaufbereitung
- Wasseraufbereitung
- Effizientes Energiemanagement für alle Anlagen
- Nahrungsmittelversorgung und Pflanzenanbau

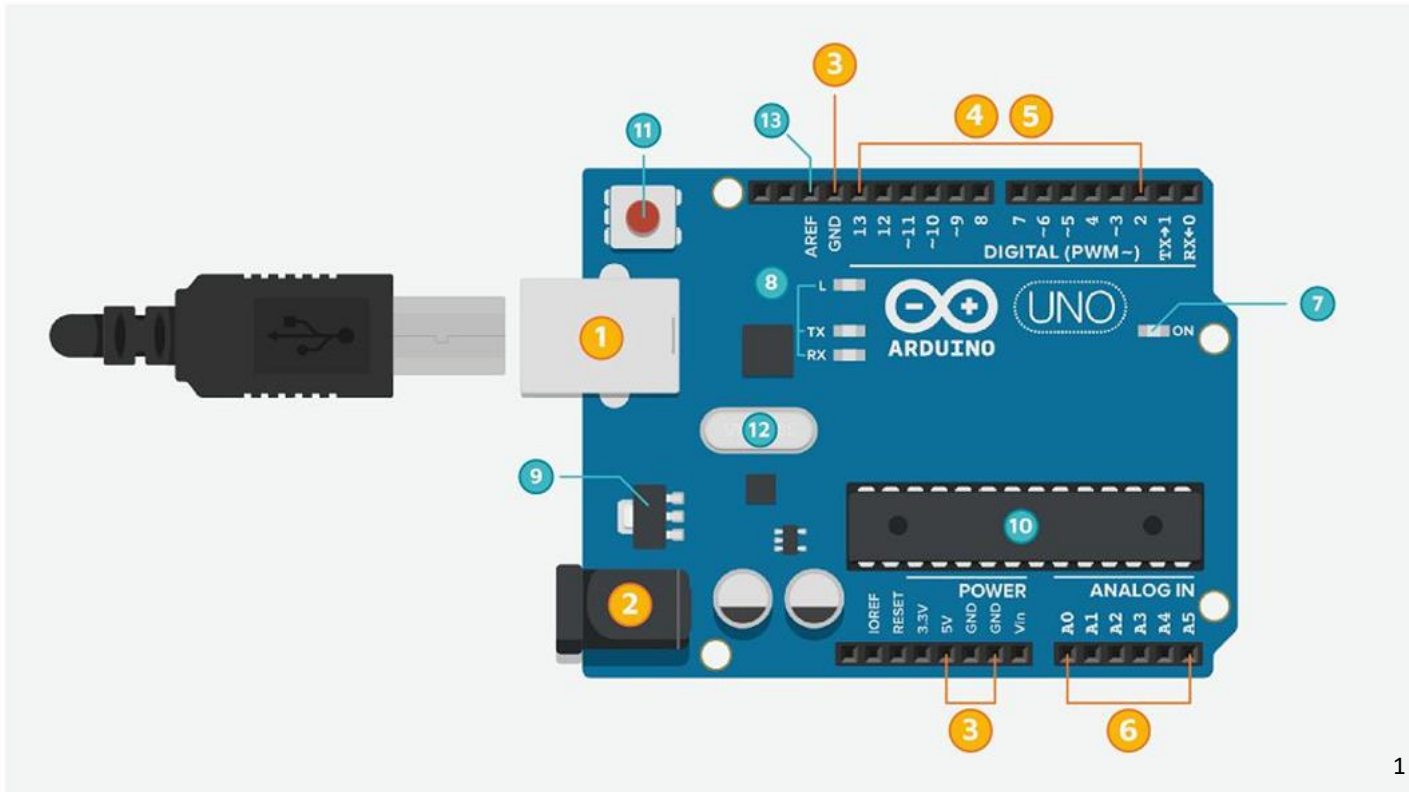
Was ist Programmieren?

- Befehle (Code) für Computer oder Roboter
- Sensoren & Aktoren
- Entwicklung von Spielen, Webseiten, selbstfahrenden Autos, usw.
- verschiedene Programmiersprachen mit unterschiedlicher Grammatik

- Ein Microcontroller, der aus verschiedenen Komponenten besteht
- Kein richtiger Computer, da keine Grafik-, Audio- und Netzwerkausgaben auf der Platine vorgesehen sind
- Liest Eingabesignale und leitet daraus anhand der Programmierung Ausgabesignale ab

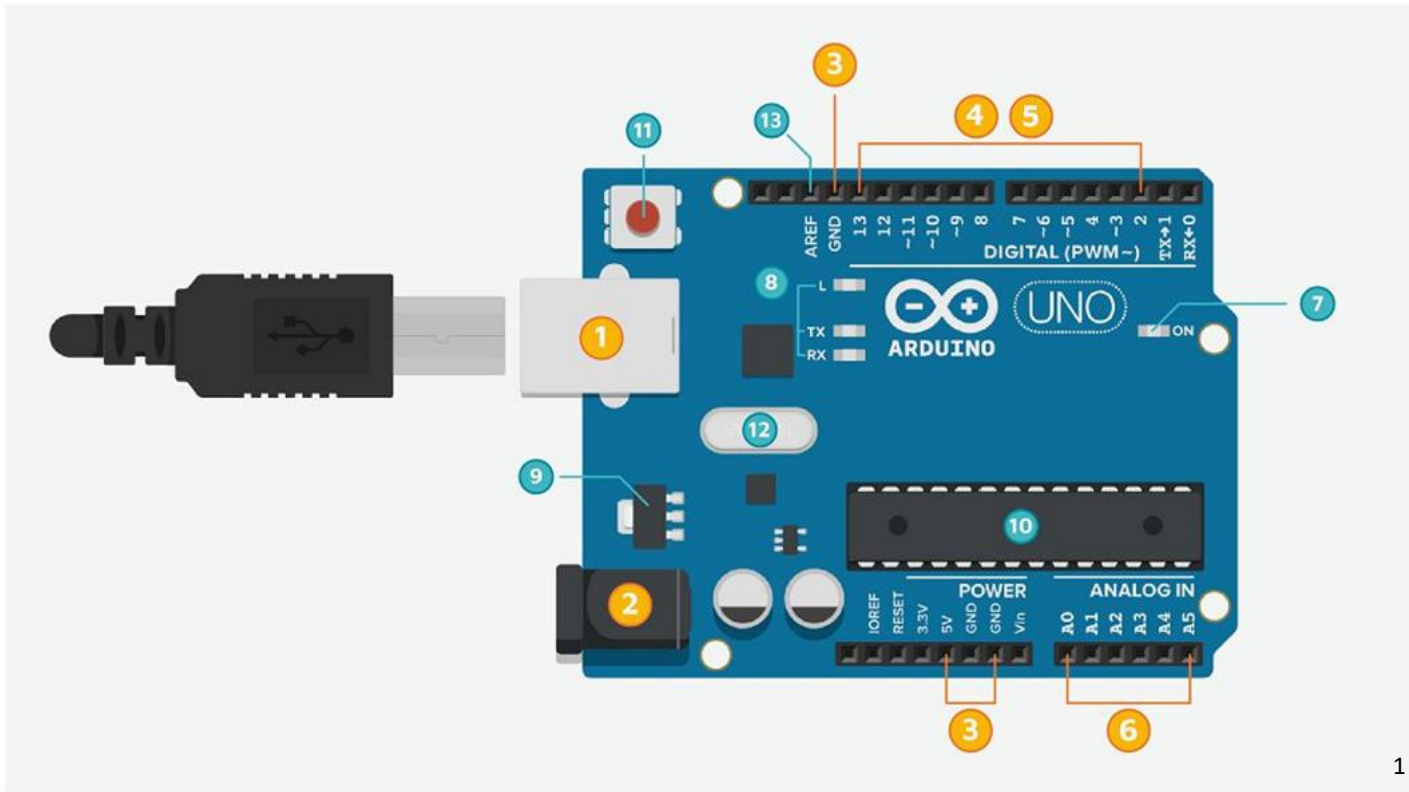


3



Gelb (relevante Elemente):

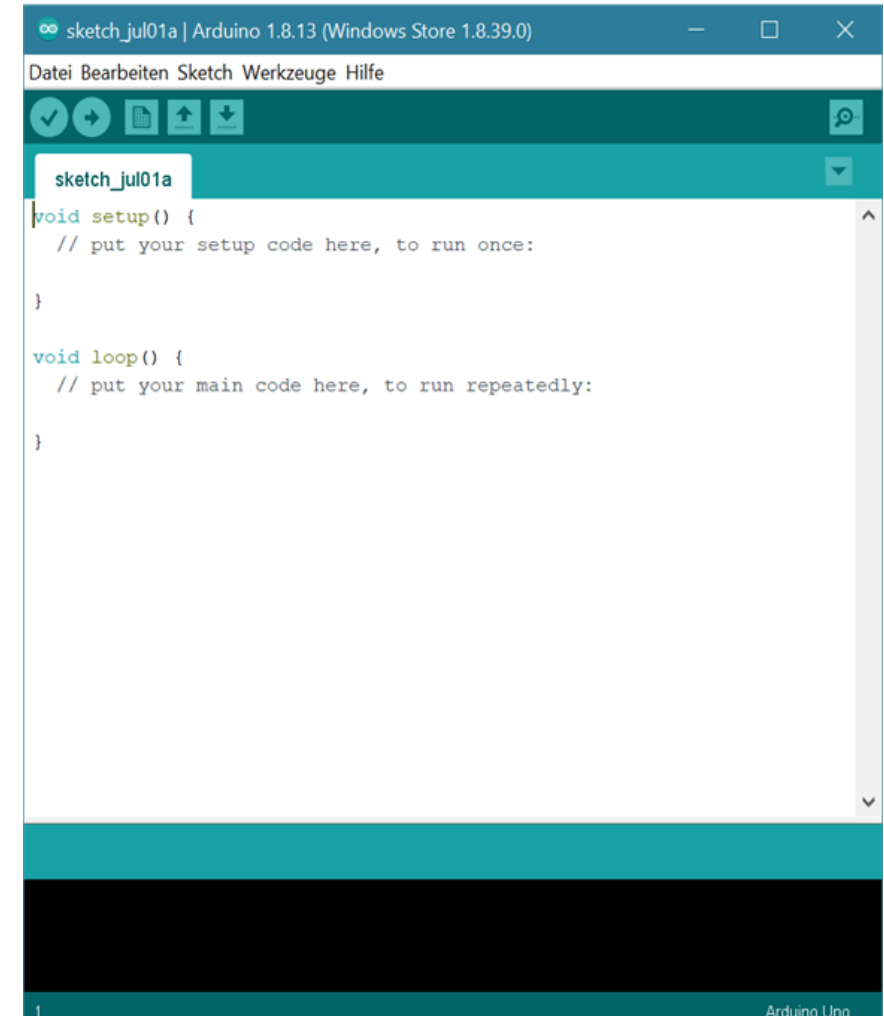
- 1 USB-Anschluss
- 2 Stromanschluss
- 3 Stromversorgung für externe Schaltungen
- 4 "Digital"-Pins (2-13)
- 5 Mit ~ "Digital"-Pins mit Pulsweitenmodulation (3,5,6,10,11)
- 6 "Analog"-Pins (A0-A5)



Türkis (Systemelemente):

- 7 ISP-Schnittstelle und Power-LED
- 8 Kommunikations-LEDs
- 9 Spannungsregler
- 10 Atmel328 (Microcontroller)
- 11 Reset-Button
- 12 Quarz (Frequenzgeber)

- Der Arduino wird mit Hilfe eines Computers programmiert.
- Das Programm heißt Arduino IDE.
- Die Software ist in der Lage unterschiedliche Boards anzusteuern.
- Das Programm unterteilt sich in verschiedene Bereiche.



2

Befehlszeile

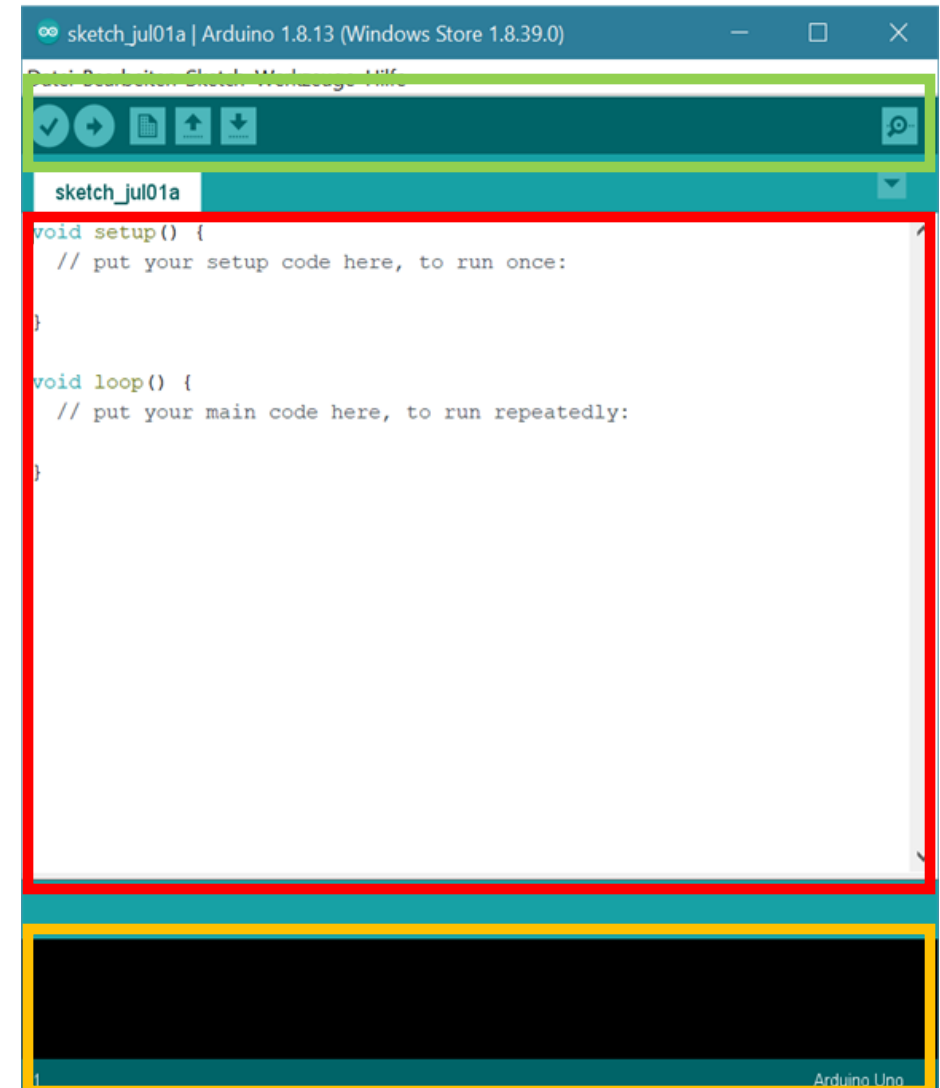
Wichtige Programmfunktionen
(Speichern, Laden, Code
übertragen, etc.)

Programmierbereich

`void setup()` und `void loop()`

Fehler-/Informationsbereich

für Fehler und Benachrichtigungen
beim Kompilieren



Das Programm besteht aus zwei

Funktionen:

- `setup` –Methode: einmalige Ausführung
- `loop` –Methode: endlose Wiederholung
- Die Befehle werden immer innerhalb der geschweiften Klammern geschrieben

```
void setup(){  
    // Anweisung  
}  
  
void loop(){  
    // Anweisung  
}
```

2

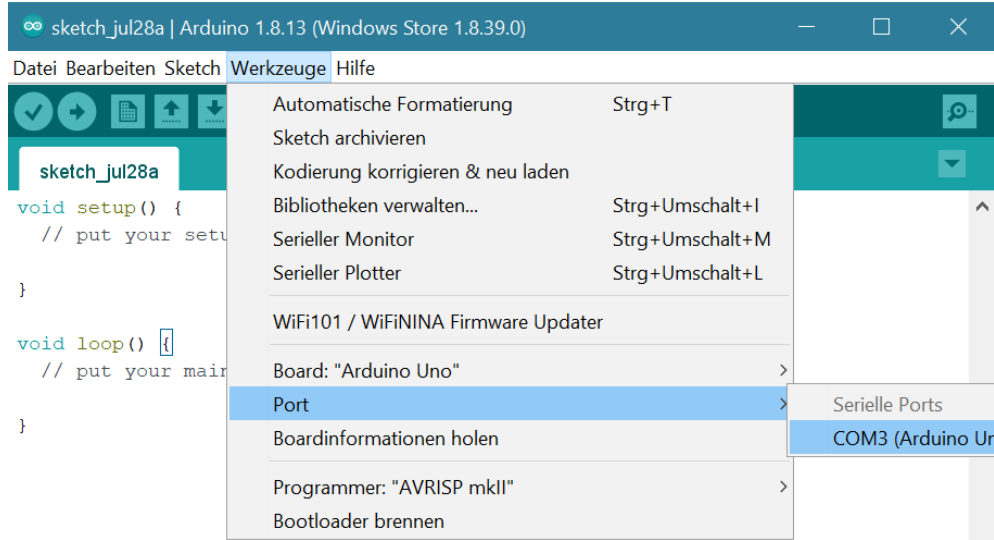
- Ein Arduino wird in C und/oder C++ programmiert.

Wichtig bei der Programmierung:

- Das Programm wird zeilenweise abgearbeitet.
- Pro Zeile ein Befehl.
- Der Befehl wird in der Regel mit einem ";"
(Semikolon) beendet.

Welcher Port?

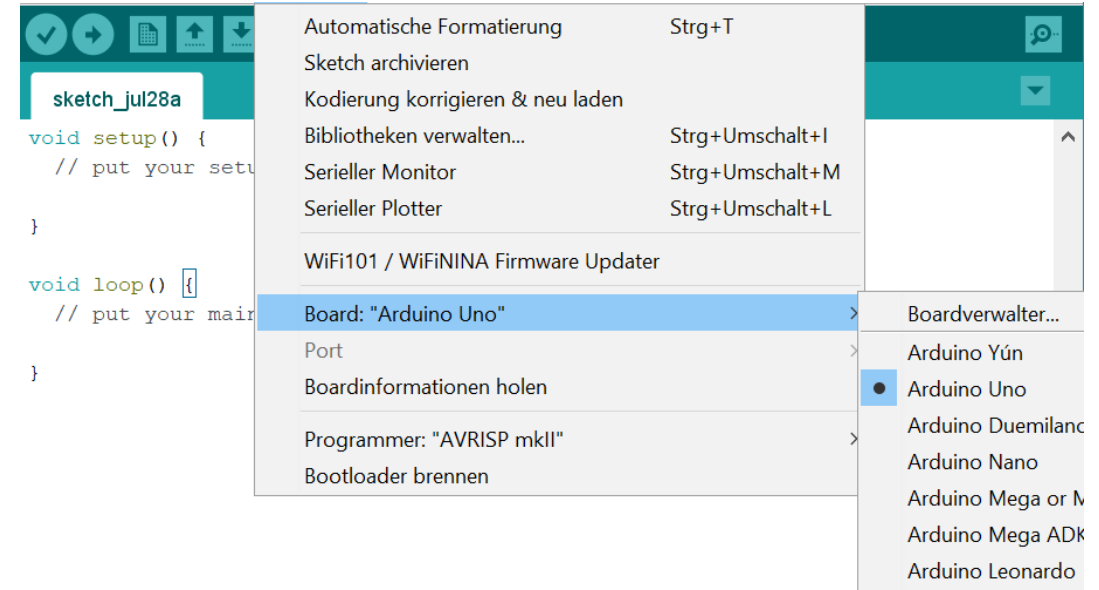
- Geht auf den Reiter „Werkzeuge“
- Klickt auf den Punkt „Port“ und wählt „Arduino Uno“ aus.
- Der COM kann unterschiedlich sein!



2

Welches Board?

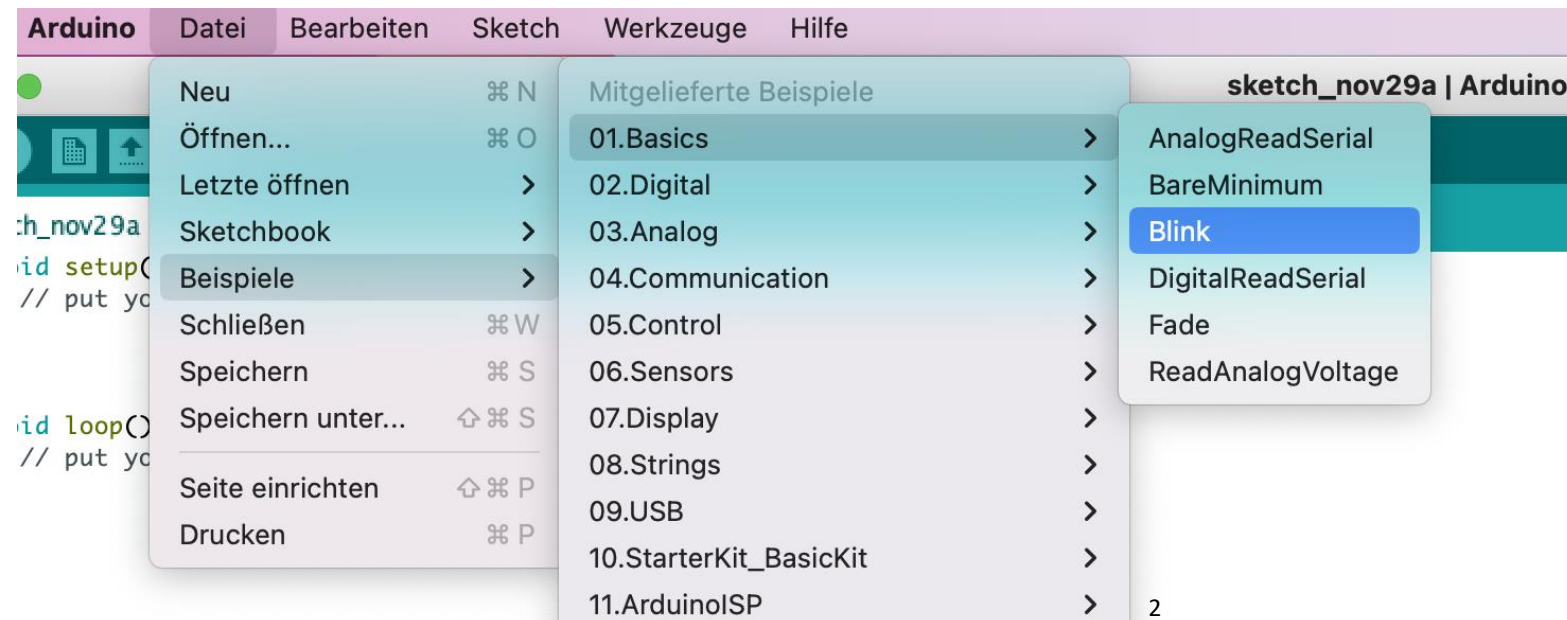
- Geht auf den Reiter „Werkzeuge“
- Klickt auf den Punkt „Board:..." und wählt „Arduino Uno“ aus



2

Erstes Beispielprogramm

- Geht auf den Reiter „Datei“
- Klickt auf den Punkt „Beispiele“, „01.Basics“ und wählt „Blink“ aus
- Überprüft es und ladet es auf den Arduino



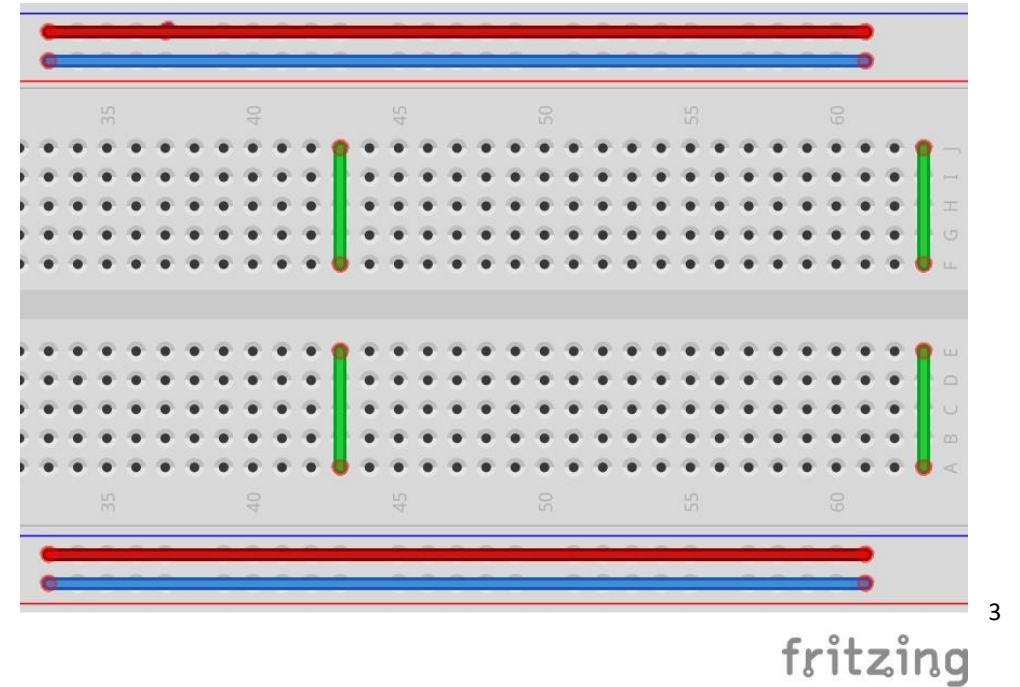
2

Blink §

```
1
2
3 // the setup function runs once when you press reset or power the board
4 void setup() {
5     // initialize digital pin LED_BUILTIN as an output.
6     pinMode(LED_BUILTIN, OUTPUT);
7 }
8
9 // the loop function runs over and over again forever
10 void loop() {
11     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
12     delay(1000); // wait for a second
13     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
14     delay(1000); // wait for a second
15 }
16
17
18
```

Breadboard oder Steckbrett

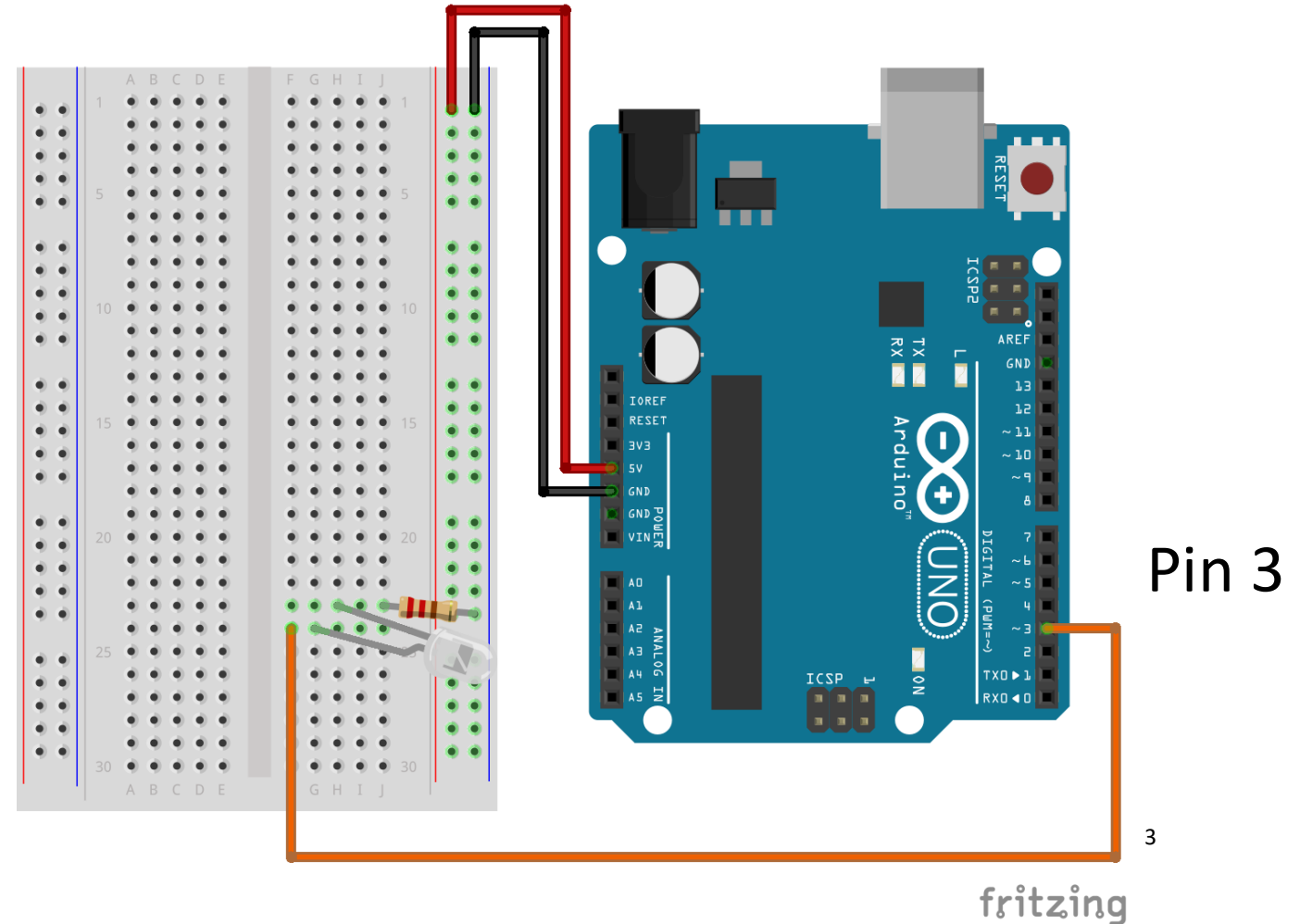
- die oberen & unteren Reihen sind miteinander verbunden
- die fünf Kontakte in einer Spalte sind miteinander verbunden

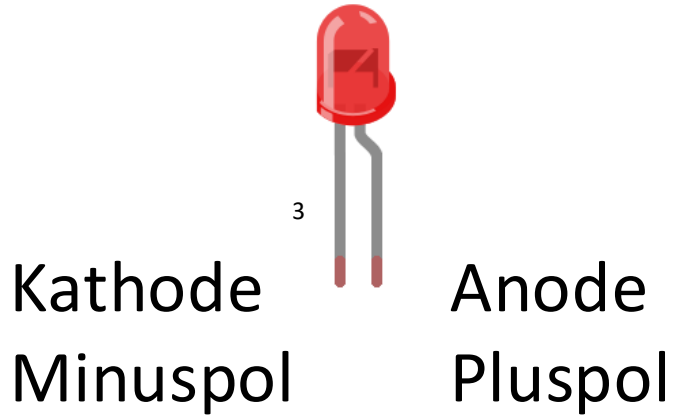


3

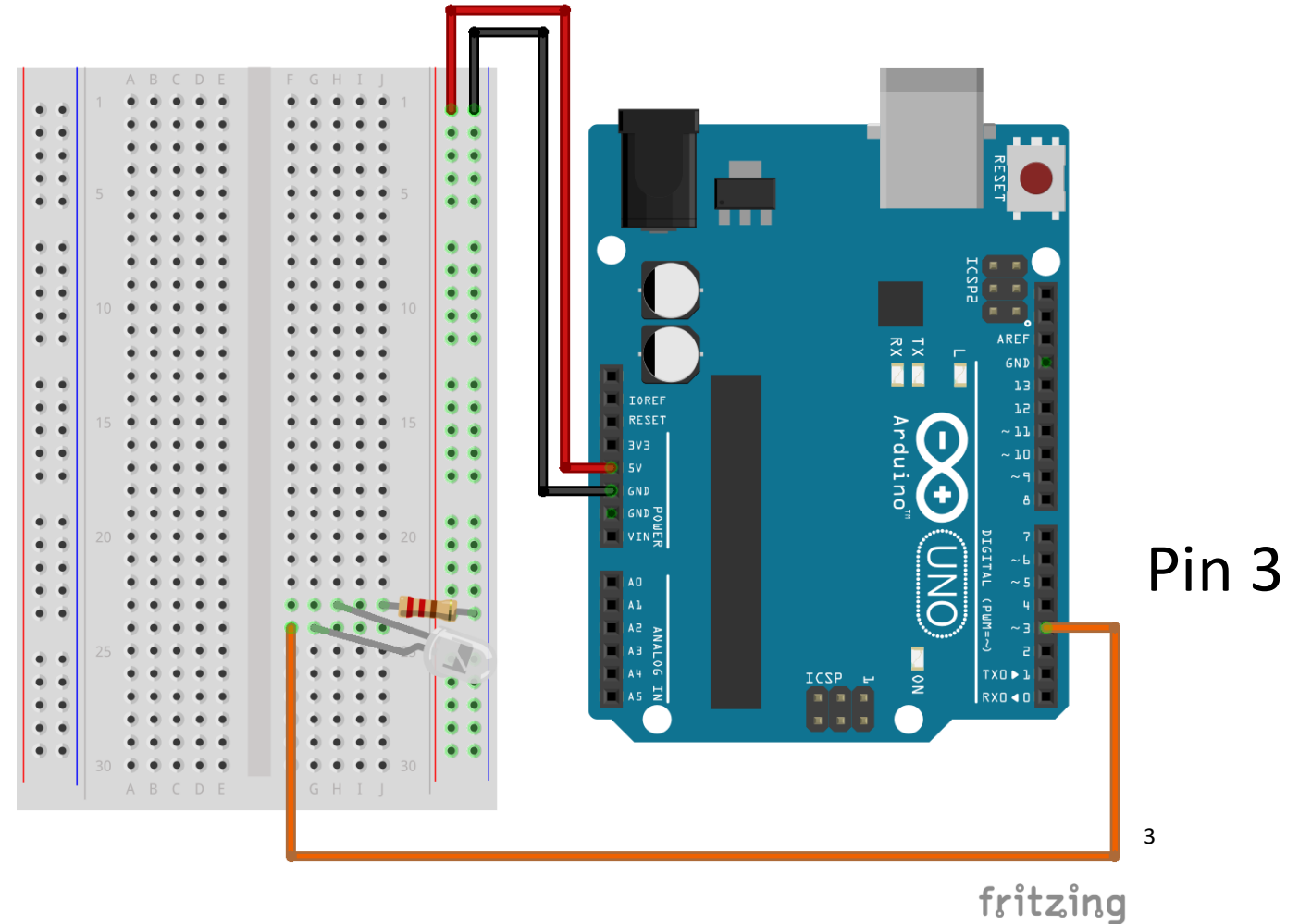
Eine LED soll unsere Lampe im Smarthome sein.

- Baue die nachfolgende Schaltung zunächst nach und ändere das Blink-Programm so ab, dass jetzt die LED auf dem Steckbrett blinkt.

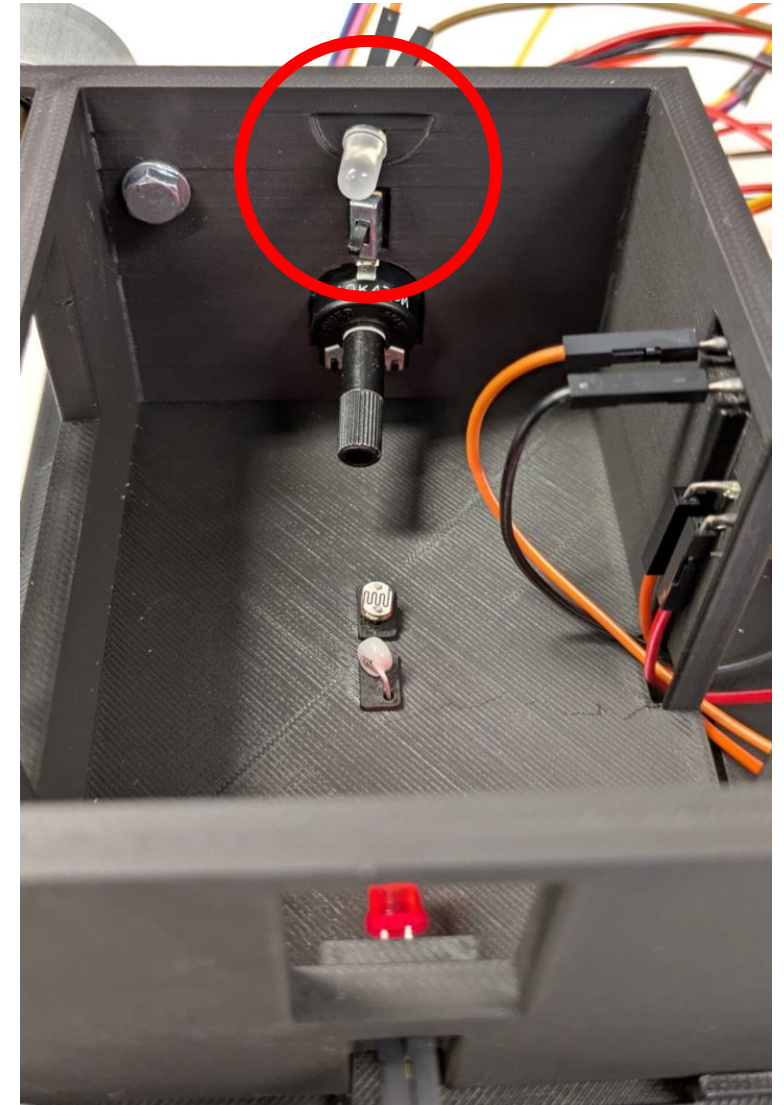




- Schließe die LED an den digitalen Pin 3 an
- Widerstand ist zum Schutz der LED



- Verbaue die LED nun im Smarthome. An der Wand gegenüber vom kleinen Fenster findest du den passenden Platz.
- Die Beinchen kannst du nach außen führen und von außen die Kabel anklemmen.
- Der Widerstand muss auf dem Steckbrett verbleiben.



Wir wollen nun durch regelmäßiges Blinken die Anwesenheit von Personen im Haus simulieren.

- Ändere das Programm nun so ab, dass die LED viermal
 - 1 Sek. leuchtet
 - 3 Sek. nicht leuchtet
 - 2 Sek. leuchtet
 - 1 Sek. nicht leuchtet

- Platzhalter
 - Datenspeicherung & -verarbeitung
 - individuelle Bezeichnung
-
- Integer (int): ganze Zahlen
 - Float (float): Fließkommazahlen
 - character (char): einzelne Buchstaben
 - boolean (bool): logische Zustände True (1) oder False (0)

- Variablen werden meist ganz oben im Programm eingeführt, noch vor der Setup-Funktion.
- Man nennt zuerst den Datentypen, z.B. *int*
- Dann den Variablennamen, z.B. *Sensorwert*
- und weist dieser Variablen einen Anfangswert zu, z.B. *0*

Beispiel: *int Sensorwert = 0;*

- Nutze die for-Schleife für Wiederholungen und vereinfache dein Programm

```
for(Start; Test; Fortsetzung) {  
    ...  
    Befehle, die wiederholt werden sollen  
}
```

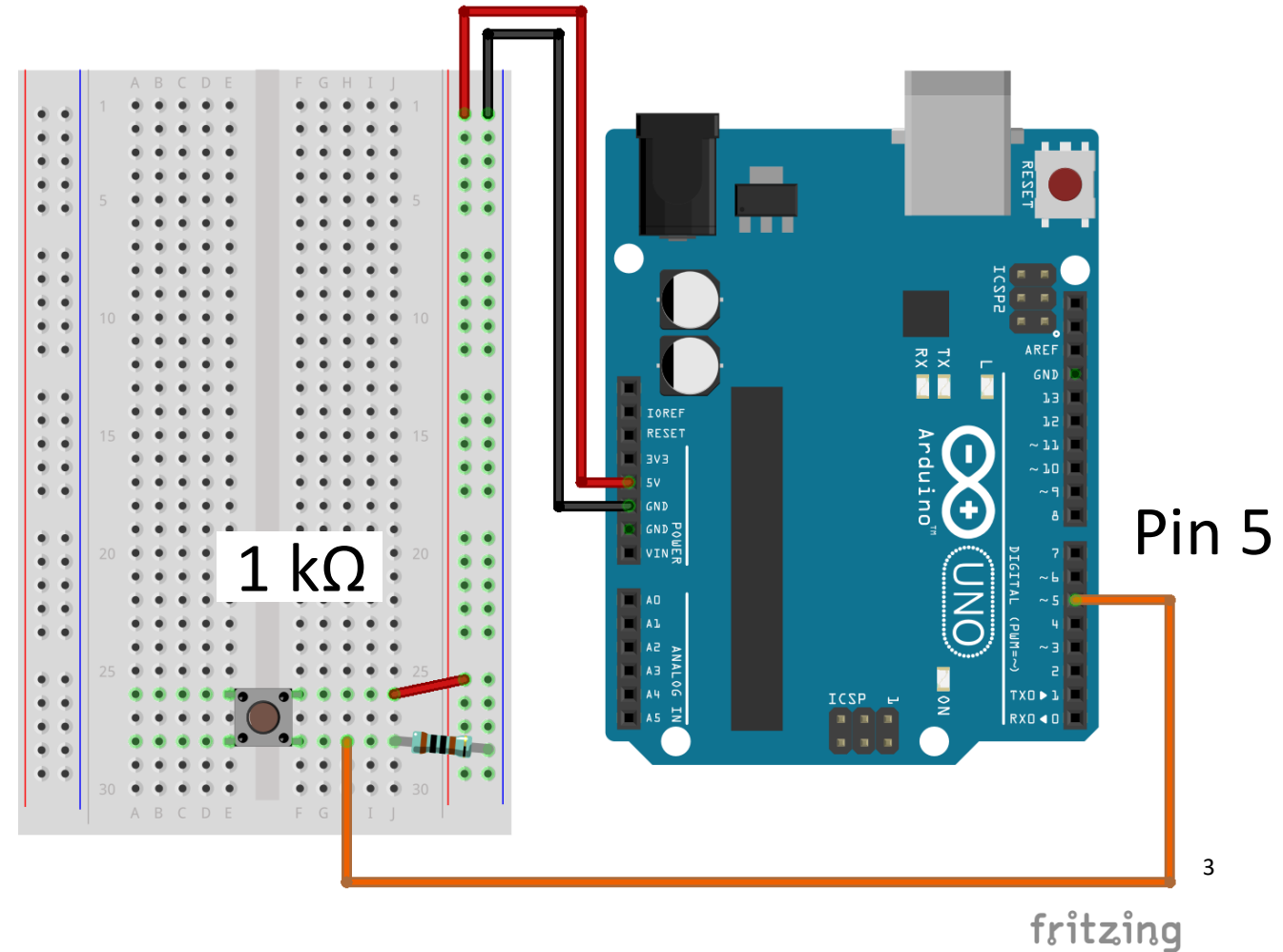
Start: Definition eines Zählers, z.B. *int zaehler = 1*

Test: Definition der Bedingung, z.B. *zaehler < 5*

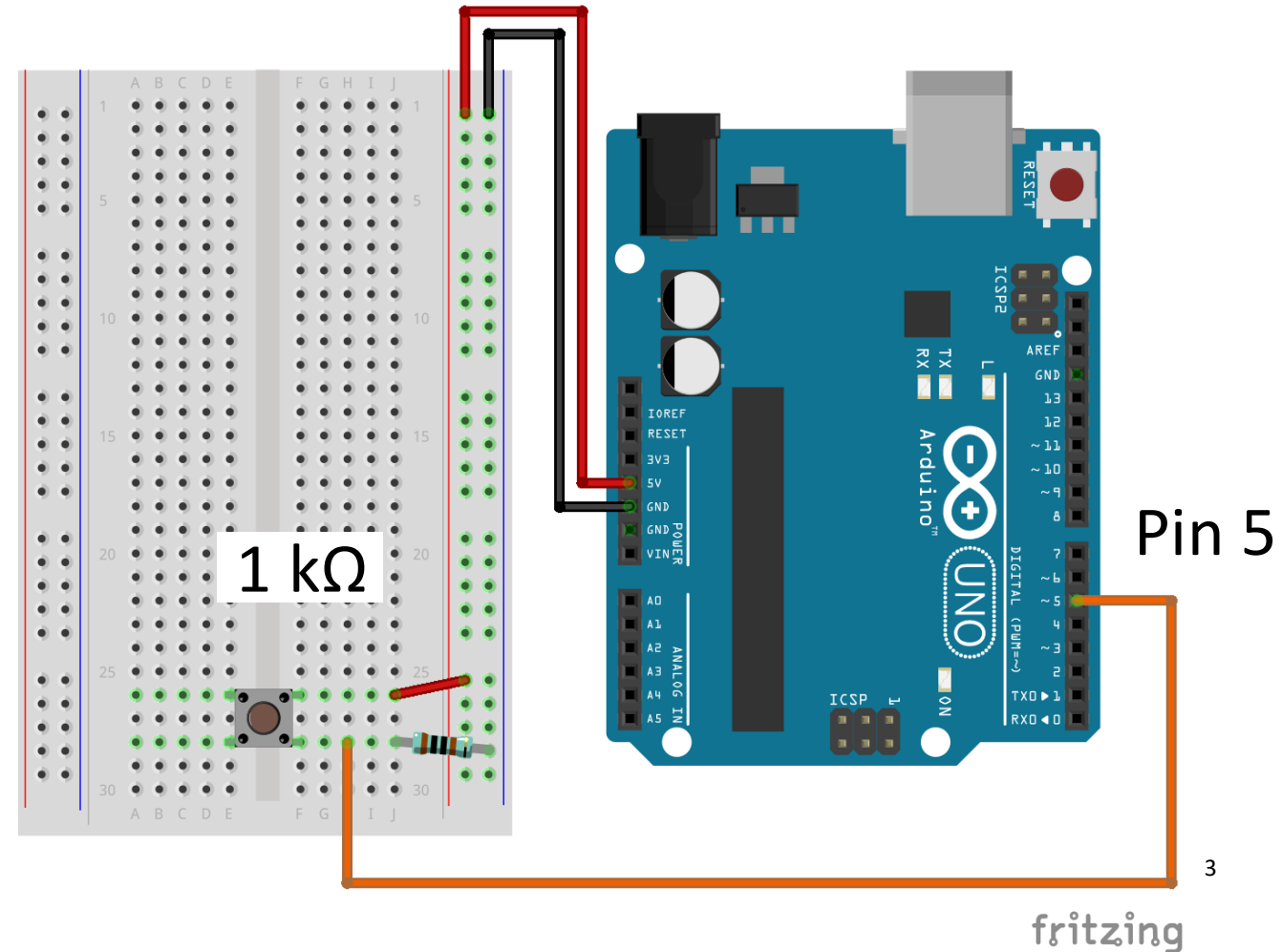
Fortsetzung: Was soll passieren, wenn die Bedingung erfüllt ist?
z.B. *zaehler = zaehler + 1*

Im nächsten Schritt wollen wir eine Klingel einbauen. Dazu brauchen wir einen Taster und einen Buzzer.

- Baue die folgende Tasterschaltung nach und erweitere dein Programm.



- Taster als Eingabe
hat zwei Zustände:
 - gedrückt (1)
 - ungedrückt (0)
- muss als INPUT definiert werden
- Auslesebefehl lautet:
digitalRead(digitaler_Pin)



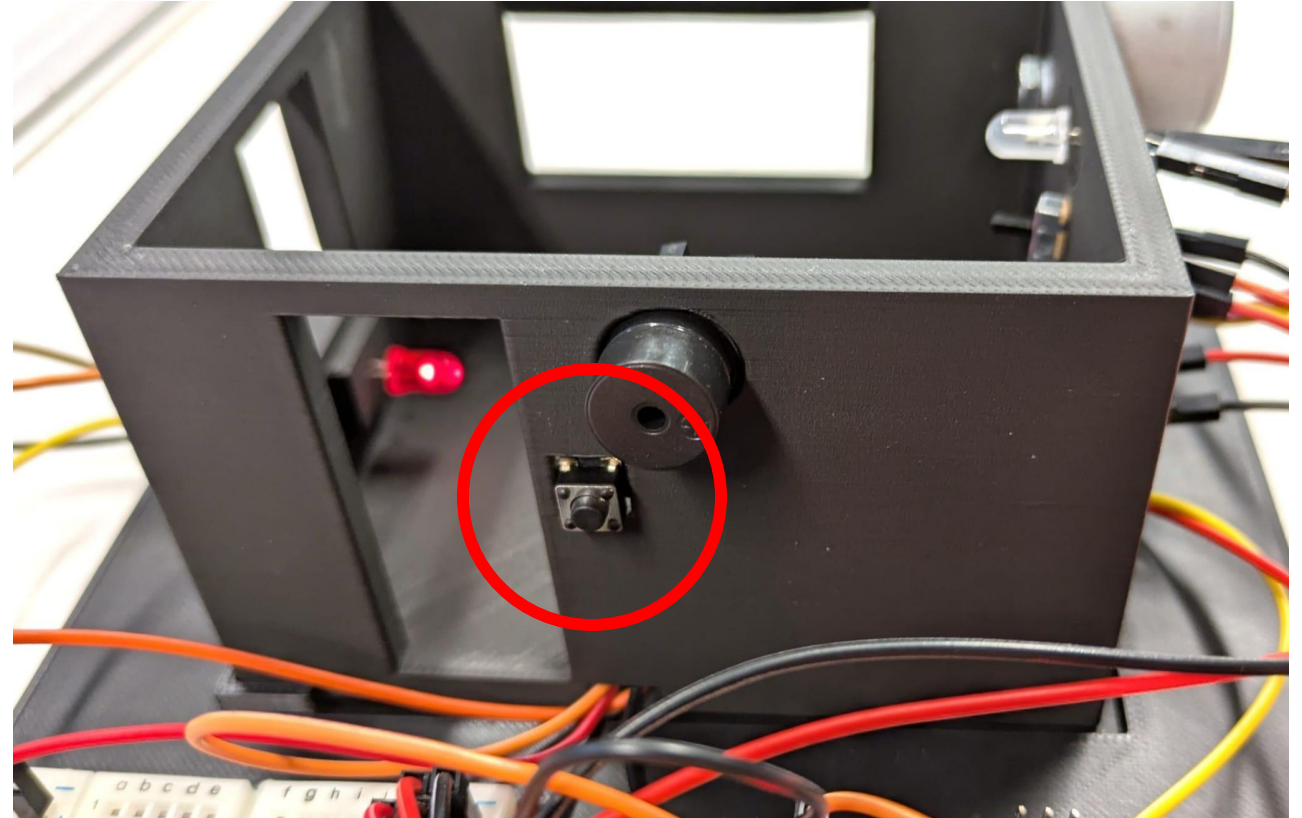
if-Abfrage für Bedingungen

```
if(Bedingung erfüllt) {  
    ... Befehle, die nur dann ausgeführt werden,  
        wenn die Bedingung erfüllt ist  
}  
else {  
    ... Befehle, die nur dann ausgeführt werden sollen,  
        wenn die Bedingung nicht erfüllt ist  
}
```

- Bedingungen mithilfe von Vergleichsoperatoren prüfen
 - Gleichheit: `==`
 - Ungleichheit: `!=`
- Wenn der Taster gedrückt wird, soll die LED leuchten
 - Tasterpin ist INPUT
 - Auslesebefehl: `digitalRead(digitaler_Pin)`
 - Taster drücken liefert 1

```
if(Bedingung erfüllt) {  
    ... Befehle ...  
}  
  
else {  
    ... Befehle ...  
}
```

- Verbaue den Taster nun neben der Tür.
- Du führst die Kabel dieses Mal von innen durch.
- Auch jetzt verbleibt der Widerstand auf dem Steckbrett.

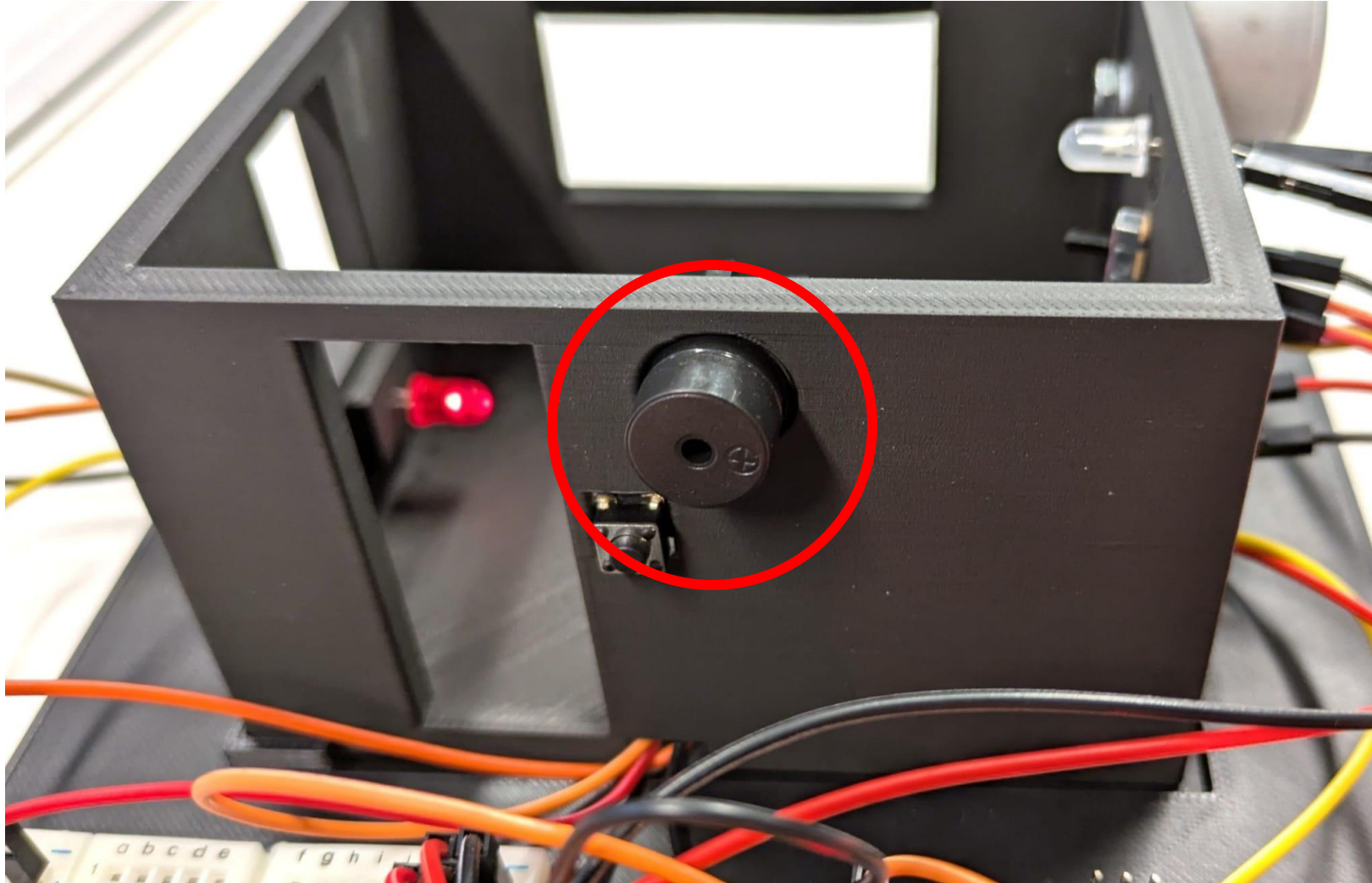


- Baue den Buzzer nun ans Haus und erweitere dein Programm so, dass der Buzzer einen Ton von sich gibt, wenn der Taster gedrückt wird. Nutze folgende Informationen:

Ein Beinchen des Buzzers kommt an den digitalen Pin 6, das andere an GND.

Befehl zur Aktivierung des Buzzers: *tone(digitaler_Pin, Tonhöhe)*

Befehl zur Deaktivierung des Buzzers: *noTone(digitaler_Pin)*



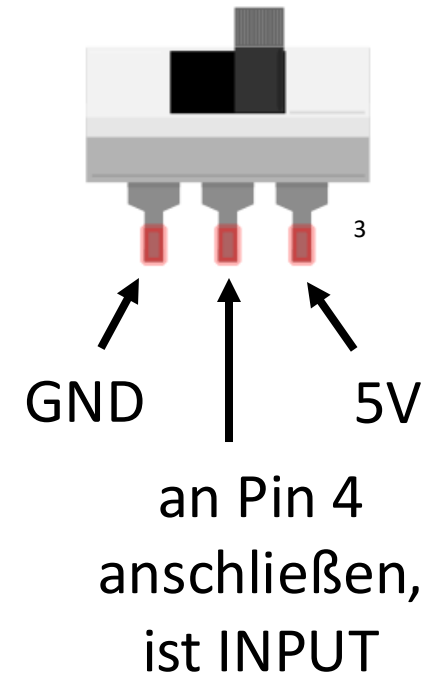
4

Als nächstes wollen wir unsere Lampe, also die LED, per Schalter aktiv an- und ausschalten.

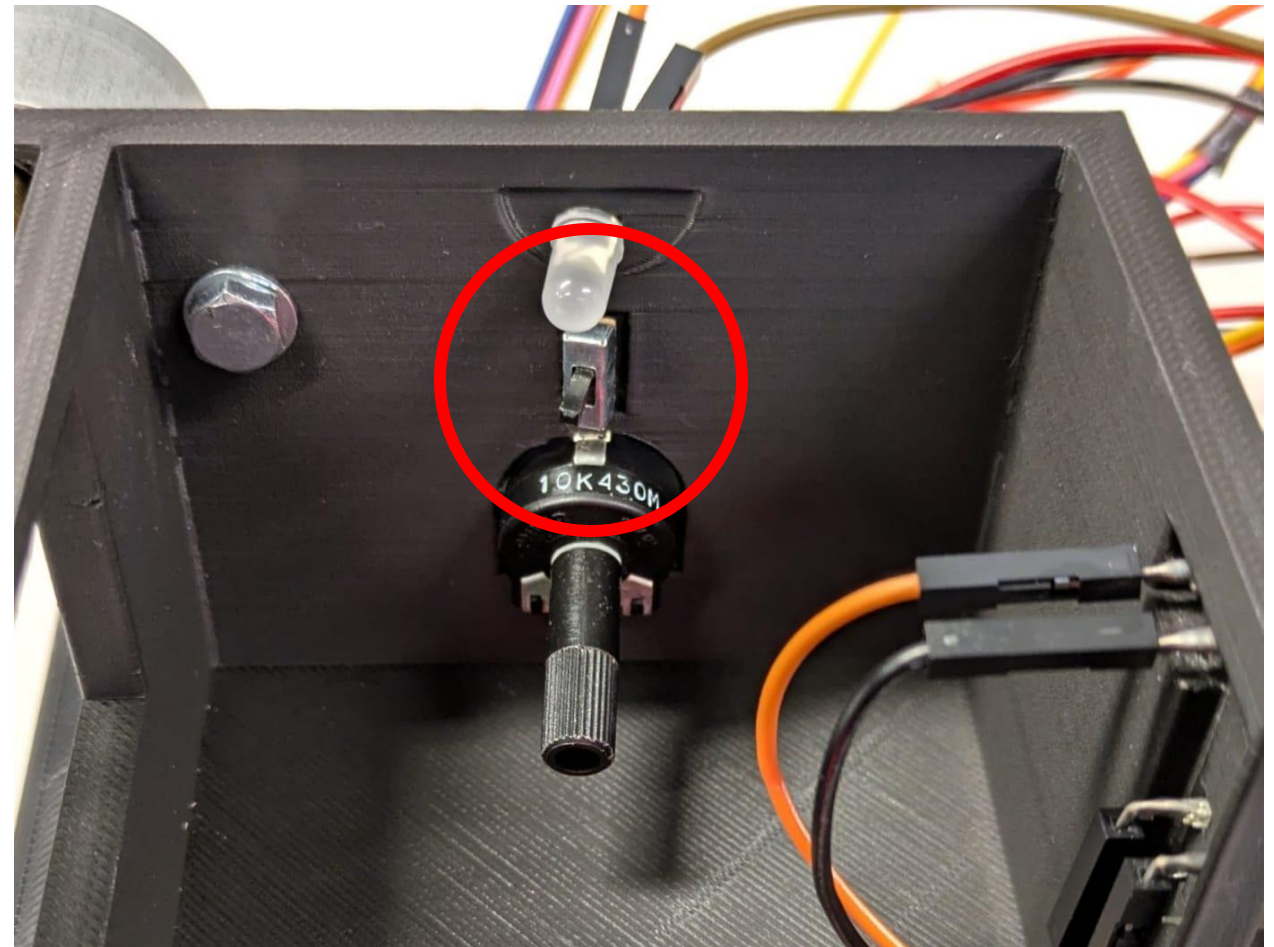
- Schalter verbindet immer zwei der drei Beinchen
- im Beispiel rechts verbindet er das mittlere und rechte Beinchen miteinander

→ `digitalRead(4)` würde eine 1 liefern

- Nutze dies und erweitere dein Programm:
Wenn der Schalter nach rechts geschoben wird, soll die LED leuchten; nach links bedeutet aus.



- Unter der bereits verbauten LED ist Platz für den Schalter.
- Beachte, dass du den Schalter im Haus nur nach oben oder unten schieben kannst.

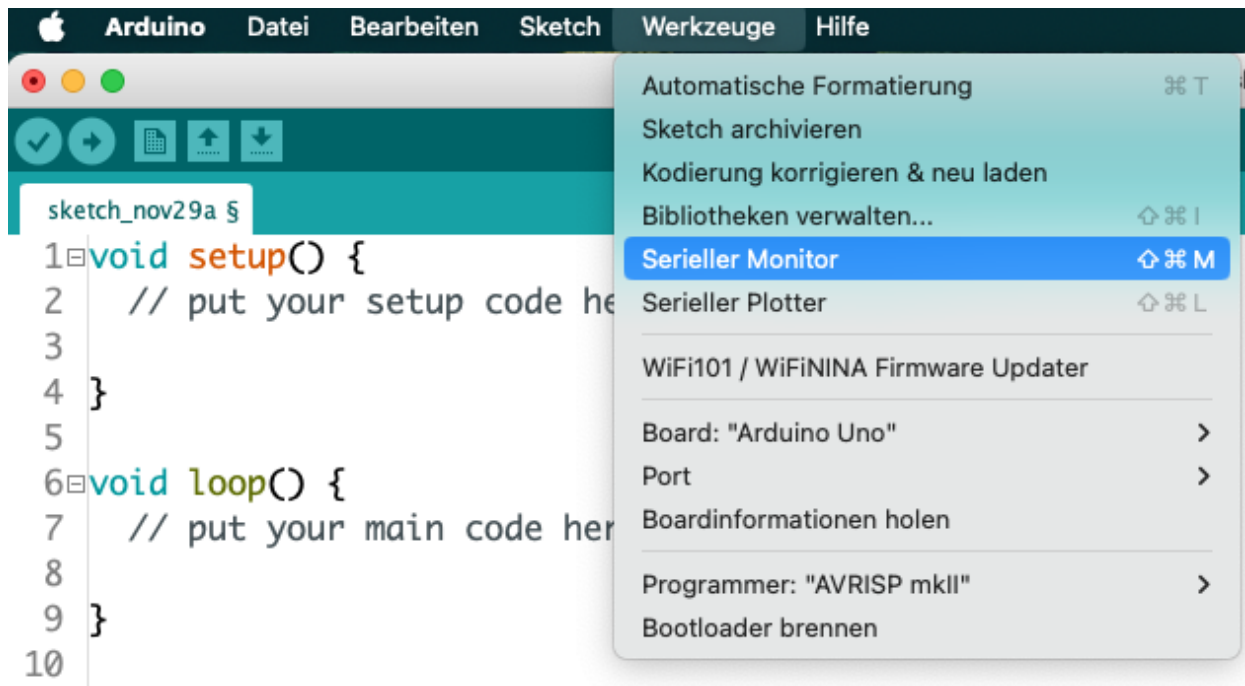


Manchmal ist es sinnvoll, sich Variablenwerte, die Ausgabewerte von Sensoren oder Rechenergebnisse anzeigen zu lassen. Du kannst dir aber auch Text anzeigen lassen. Das geht mit dem seriellen Monitor.

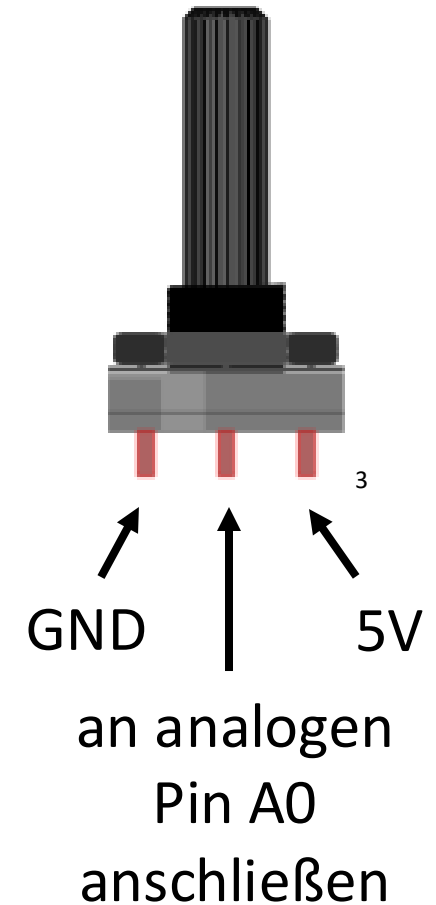
- Füge im *void setup()* folgende Zeile ein:
Serial.begin(9600);
- In der Loop kannst du es so benutzen:
Serial.println("Programmieren macht Spaß");
Serial.println(digitalRead(4));

Text muss immer in Anführungszeichen gesetzt werden, Variablen nicht

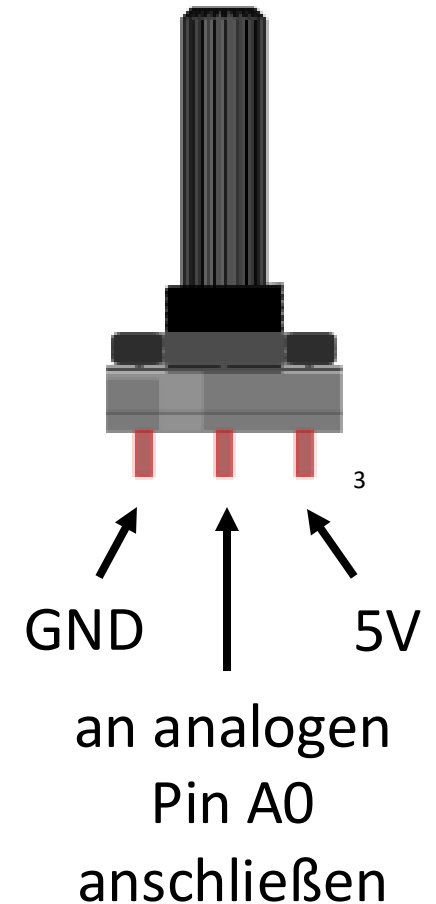
Über den Reiter Werkzeuge kannst du den seriellen Monitor auswählen.
Dort erscheinen deine Ausgaben.



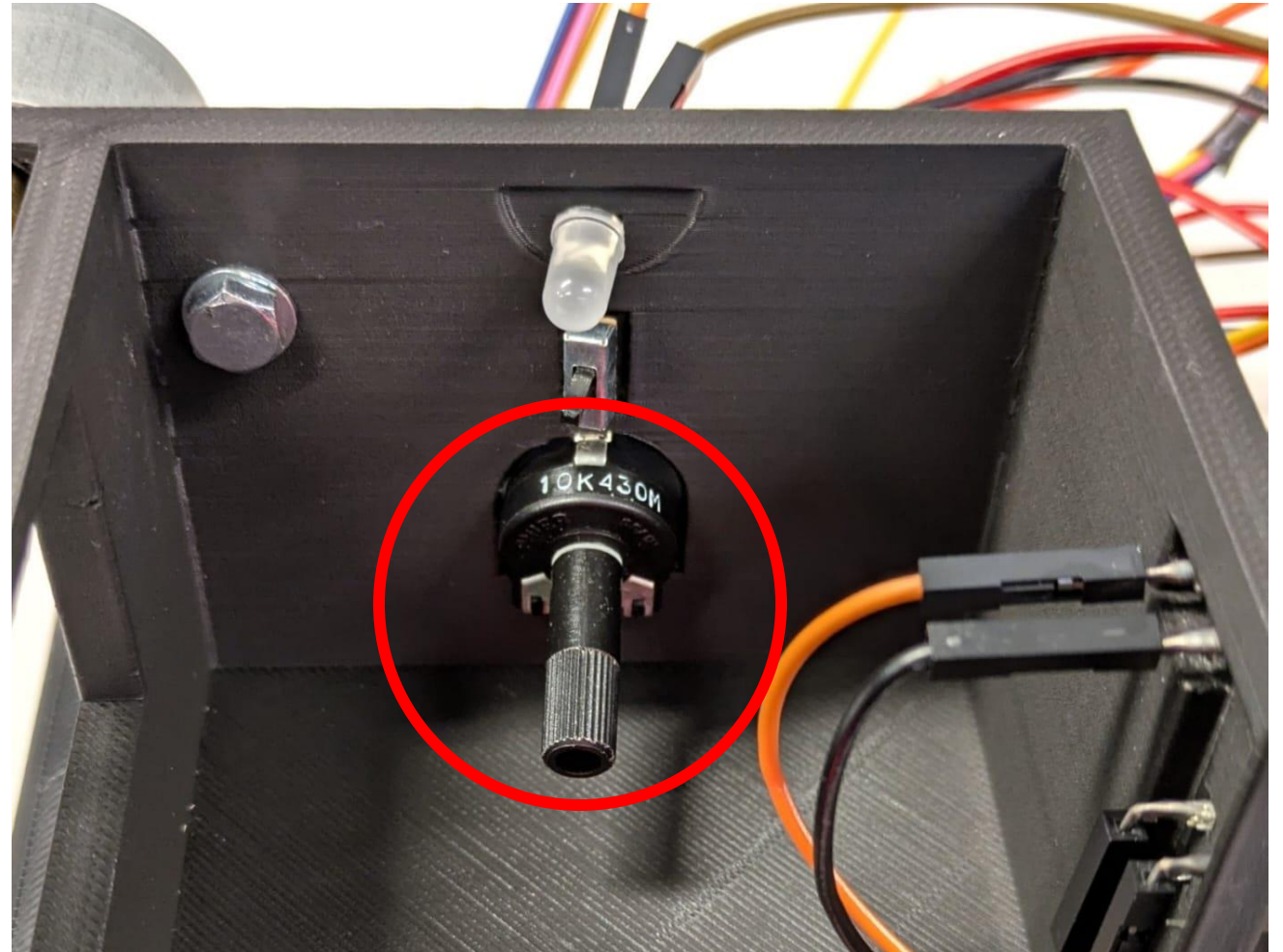
- Unsere Lampe leuchtet sehr hell.
Es wäre toll, wenn wir sie auch dimmen könnten.
- Das geht mit einem Potentiometer.
Das ist ein Widerstand, der mechanisch verändert werden kann.
- Analoge Pins werden meistens dazu benutzt,
Sensorwerte zu lesen und zu verarbeiten,
ihr Wertebereich liegt zwischen 0 und 1023,
- Auslesebefehl lautet:
analogRead(analoger_Pin); hier ist der analoge Pin A0



- Auslesebefehl: `analogRead(analoger_Pin);`
- Wert speichern wir z.B. in:
`Wert_Potentiometer = analogRead(analoger_Pin);`
- Diesen Wert können wir der LED statt dem HIGH-Signal geben, um die Helligkeit zu regeln.
Neuer Befehl: `analogWrite(digitaler_LED-Pin, Helligkeitswert);`
- **Achtung:** Die LED kann aber nur Werte zwischen 0 und 255 bekommen. Analoges Pin liefert Werte zwischen 0 und 1023.

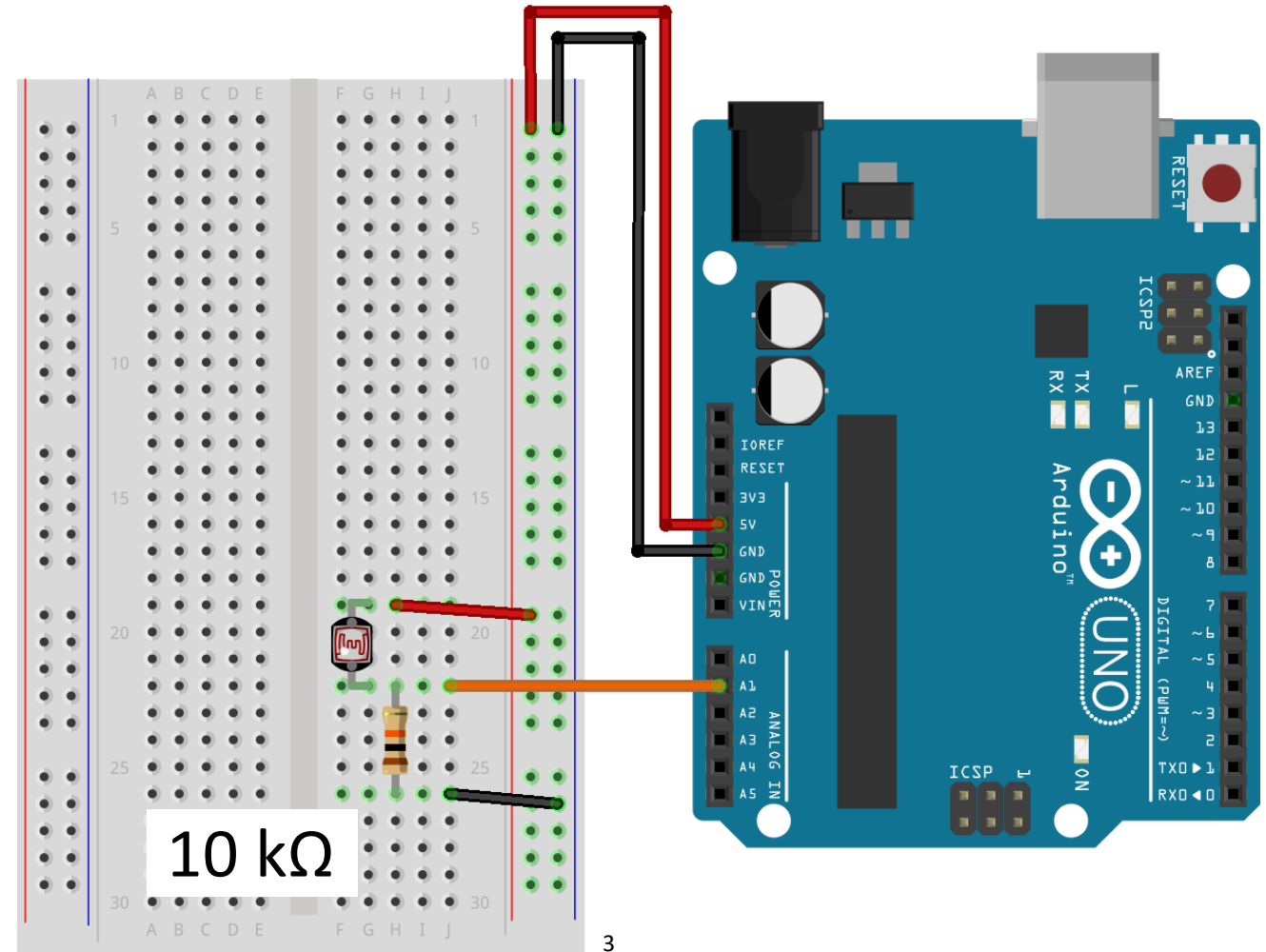


- Installiere das Potentiometer unter dem Schalter.
- Passe das Programm an und dimme die LED nur, wenn der Schalter betätigt wurde.



Mit Hilfe eines Lichtsensors können wir unsere LED auch automatisch anmachen, weil es z.B. zu dunkel geworden ist.

- Baue die Schaltung nach und erweitere dein Programm um die Variable *Wert-Lichtsensor*, in welcher du den Wert des Sensors am analogen Pin A1 speicherst.
- Lass dir den Wert im seriellen Monitor anzeigen.



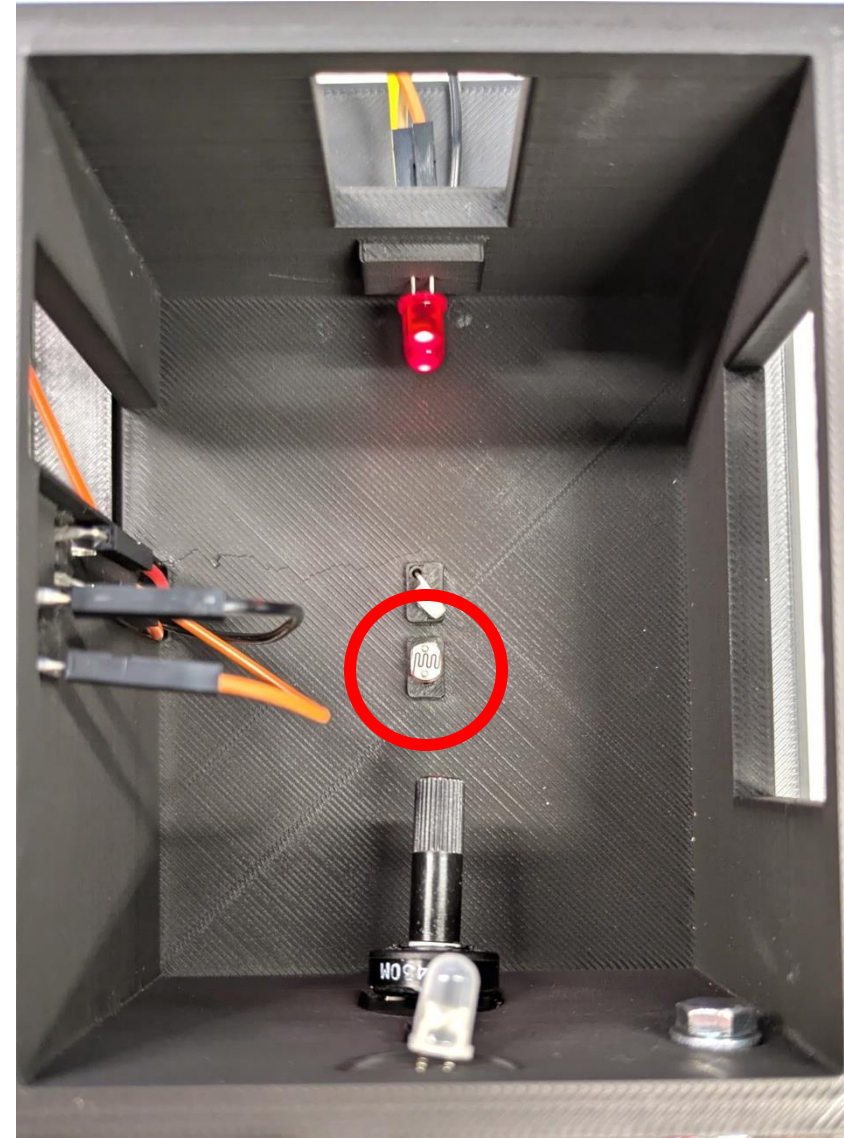
Um den Sensorwert zu integrieren, müssen wir einen Schwellwert definieren. Nimmt der Sensor weniger Licht wahr, soll die LED eingeschaltet werden.

- Erweitere dein Programm um eine Variable *Helligkeitsschwelle* und weise ihr einen sinnvollen Wert zu.
- Erweitere die LED-Abfrage so, dass das Licht angeht, wenn wir den Lichtschalter betätigen **ODER** wenn es zu dunkel ist.

Hinweis: Erweiterung der if-Abfrage auf mehrere Bedingungen:

- UND-Verknüpfung: `if ((Bedingung 1) && (Bedingung 2))`
- ODER-Verknüpfung: `if ((Bedingung 1) || (Bedingung 2))`

- Verbaue den Lichtsensor in der Mitte des Hauses.
- Führe die Beinchen nach unten raus. Du kannst sie dann zur Seite biegen und die Kabel wieder anbringen.

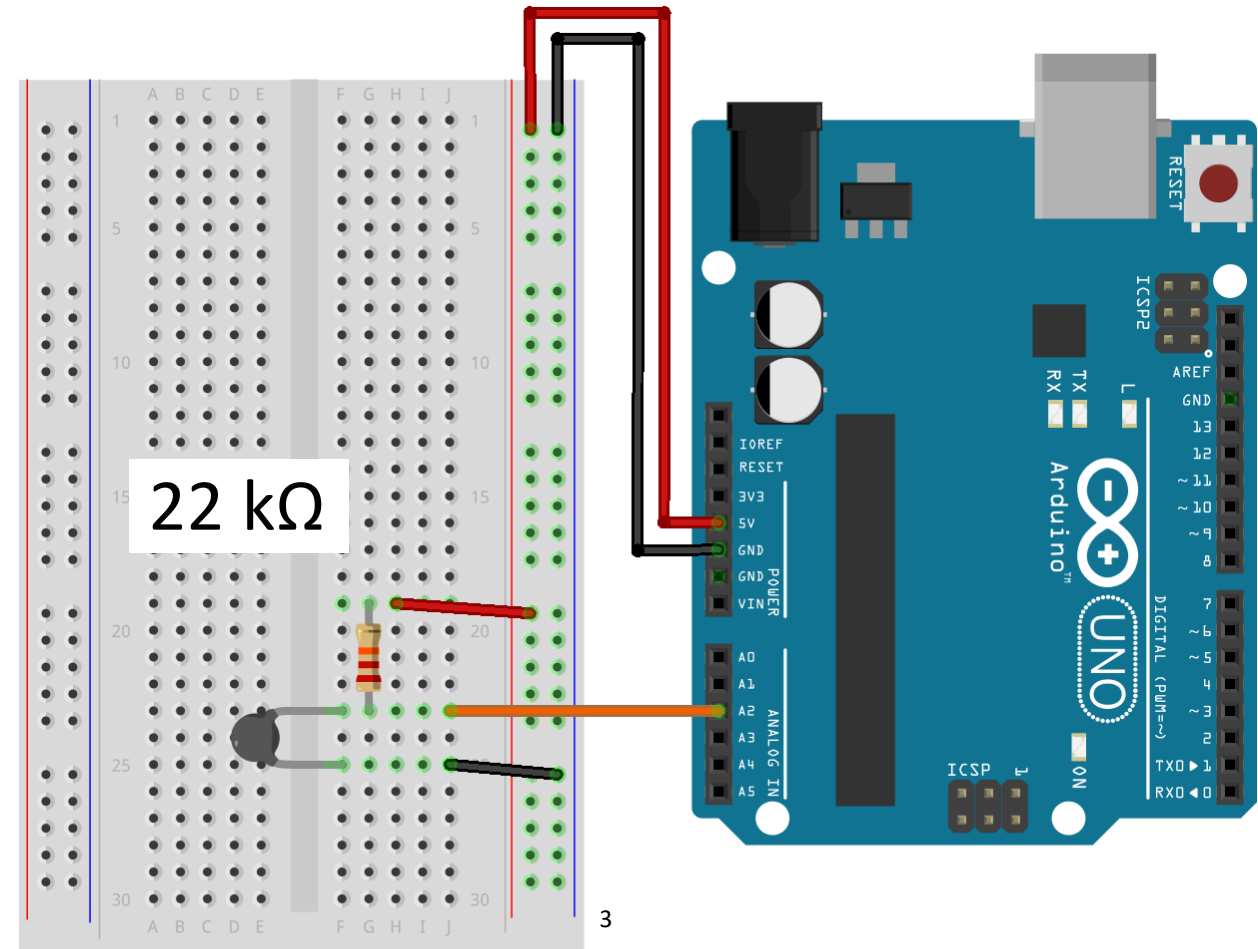


Ein weiteres Element eines Smarthomes ist eine automatische Heizungsanlage, die sich bei niedrigen Temperaturen selbst einschaltet. Wir simulieren sie mit der roten LED.

- Baue eine weitere LED-Schaltung mit der roten LED an den digitalen Pin 7, nutze dazu erneut einen 220Ω Widerstand.
- Du kannst die LED direkt im Haus unter dem Fenster verbauen.
- Teste deine Schaltung.

Wir brauchen einen Temperatursensor, einen sogenannten Thermistor. Das ist ein Widerstand, dessen Widerstandswert von der Temperatur abhängig ist. Wenn die Temperatur ansteigt, nimmt der Widerstandswert ab und umgekehrt.

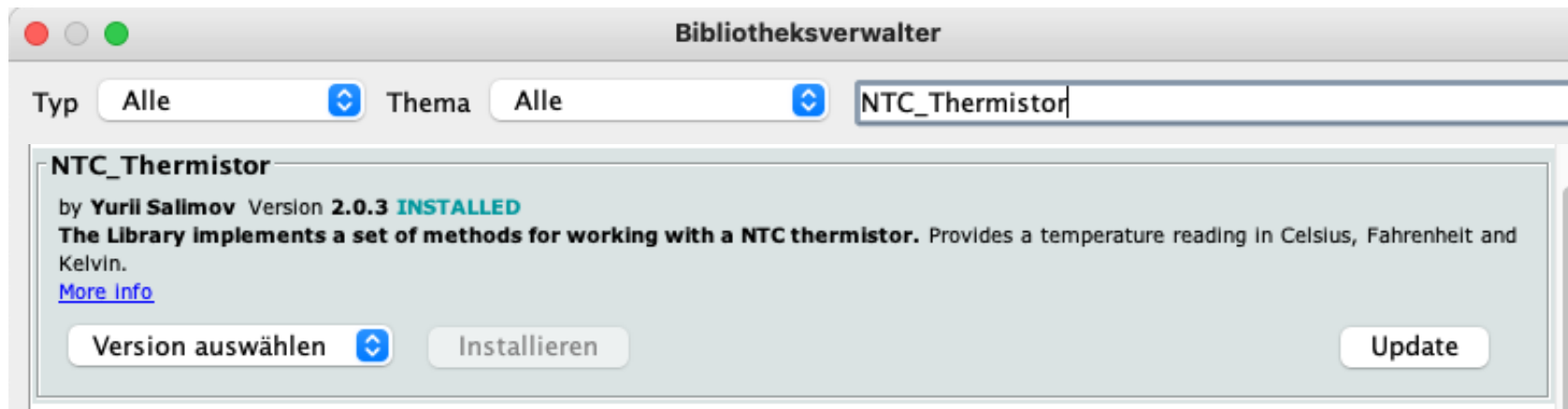
- Baue die nebenstehende Schaltung nach und benutze den analogen Pin A2.



Zur Programmierung wollen wir eine Bibliothek nutzen.

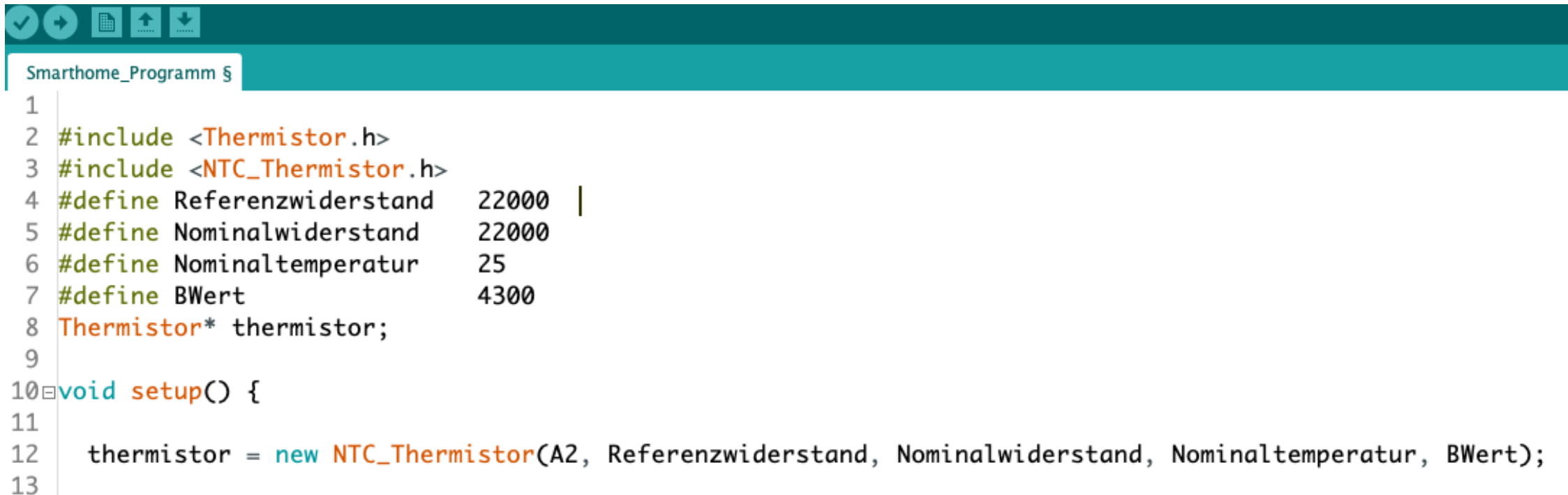
Bibliotheken sind Sammlungen von speziellen Befehlen und Funktionen, die für bestimmte Zwecke geeignet sind.

- Installiere die Bibliothek, indem du über den Reiter *Werkzeuge* die Option *Bibliotheken verwalten...* anklickst.
- Suche nach *NTC_Thermistor* und klicke auf „Installieren“.



2

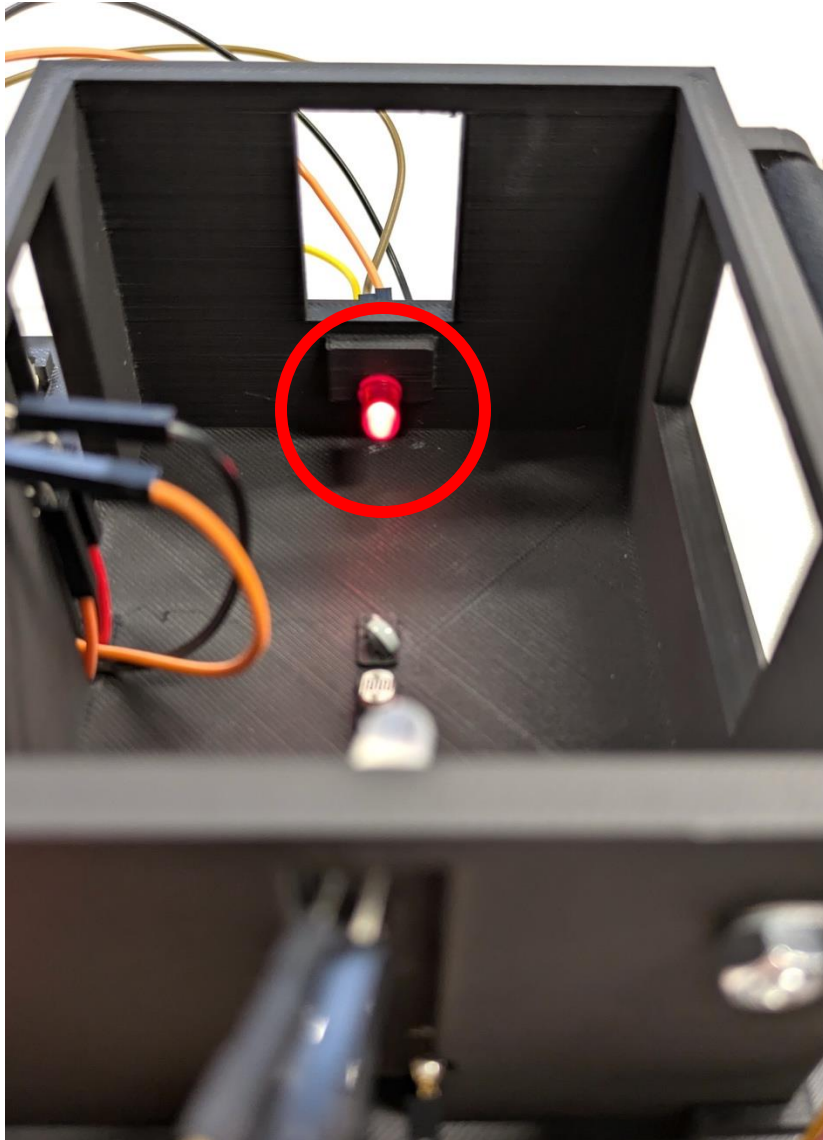
- Um die Bibliothek benutzen und den Temperatursensor programmieren zu können, füge folgende Befehle in dein Programm ganz oben ein:



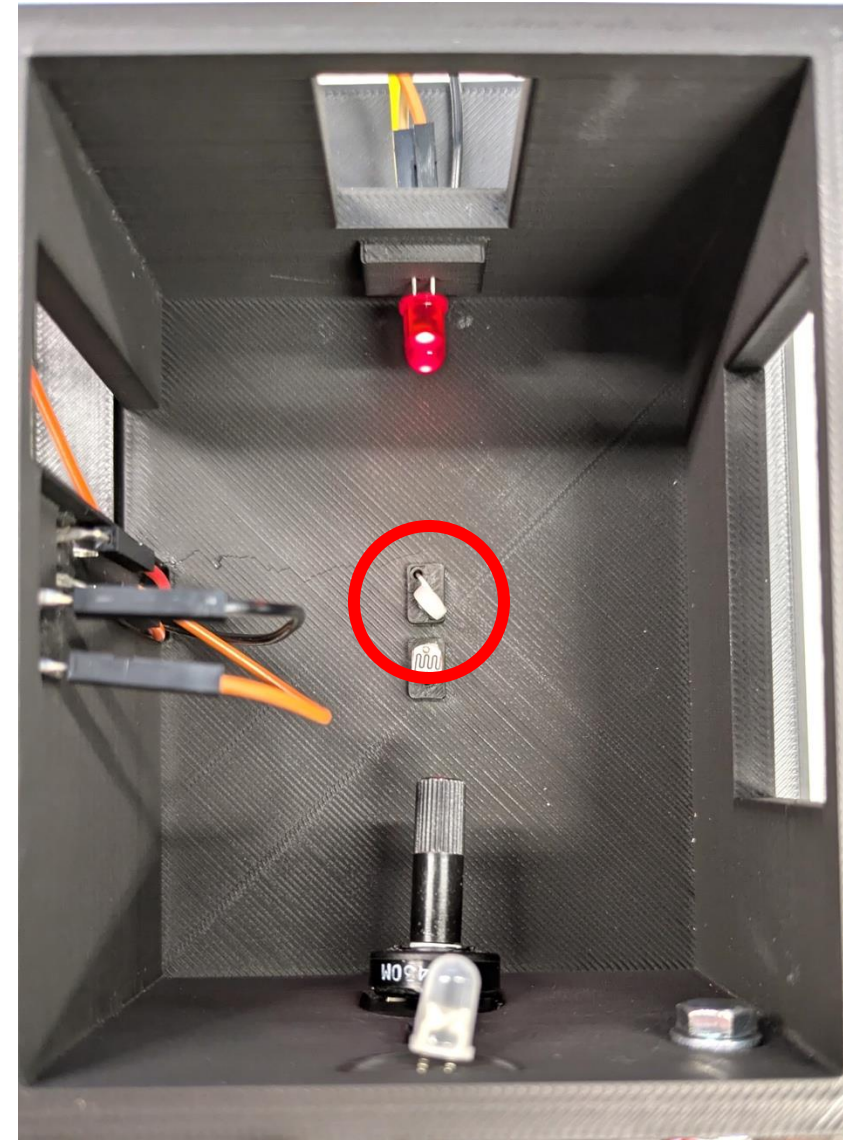
```
1  
2 #include <Thermistor.h>  
3 #include <NTC_Thermistor.h>  
4 #define Referenzwiderstand 22000 |  
5 #define Nominalwiderstand 22000  
6 #define Nominaltemperatur 25  
7 #define BWert 4300  
8 Thermistor* thermistor;  
9  
10 void setup() {  
11  
12     thermistor = new NTC_Thermistor(A2, Referenzwiderstand, Nominalwiderstand, Nominaltemperatur, BWert);  
13
```

2

- Zum Auslesen der Temperatur brauchst du folgenden Befehl:
temperatur = thermistor->readCelsius();
- Definiere dazu die Variable *Temperatur* im Vorfeld als Fließkommazahl.
- Erweitere nun dein Programm so, dass sich die Heizung (rote LED) einschaltet, sobald eine bestimmte Temperatur unterschritten wird.
- Verbaue auch den Temperatursensor mittig im Haus.



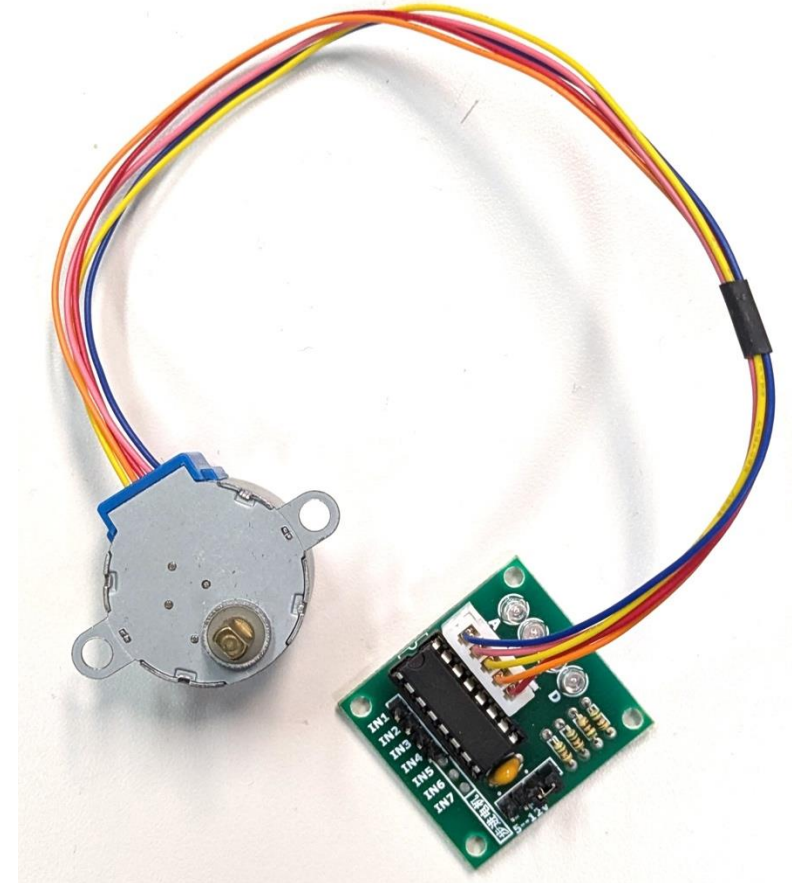
4



4

Zuletzt wollen wir unsere Rollläden in Abhängigkeit des Tageslichts steuern:

- Runter bei Nacht oder starker Sonneneinstrahlung
 - Hoch bei normalem Tageslicht
-
- Verbinde den Motor mit der Treiberplatine, mit der wir den Motor ansteuern können. Das geht über den großen weißen Stecker.



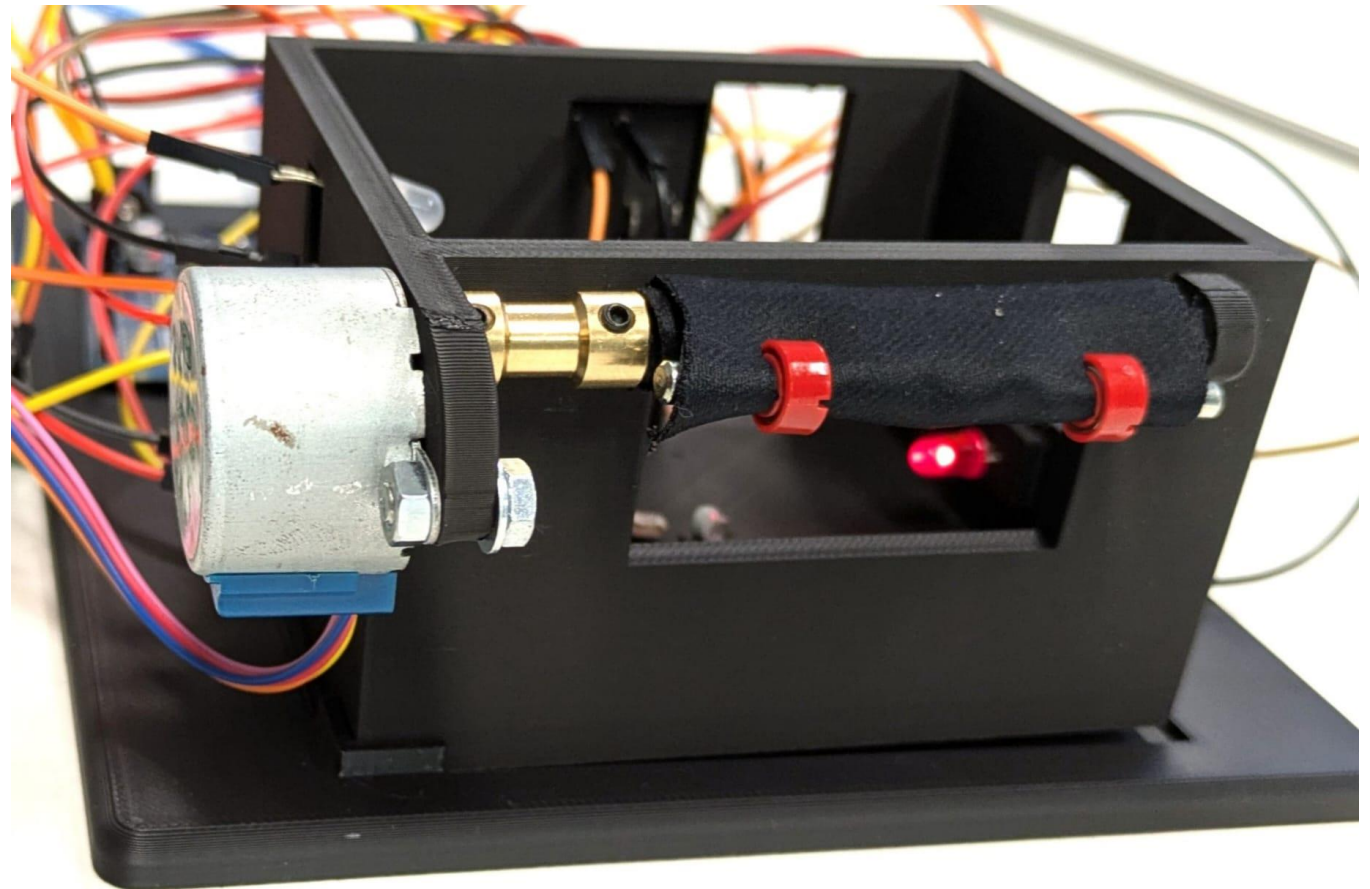
- Schließe nun den Motor über die Treiberplatine an den Arduino an.
Beachte folgende Pinbelegung:

Motortreiber	Arduino
+	5V
-	GND
IN1	PIN 8
IN2	PIN 9
IN3	PIN 10
IN4	PIN 11

- Bibliothek zur Programmierung sollte installiert sein; füge sie ganz oben hinzu: *#include <Stepper.h>*
- Definition neuer Variable: *int Schritte = 2048.*
2048 ist ein Richtwert für den Motor und gibt an, aus wie vielen Einzelschritten sich eine ganze Umdrehung zusammensetzt.
- Definition des Schrittmotors oberhalb der setup-Funktion:
Stepper Schrittmotor(Schritte, 8, 10, 9, 11);
- Geschwindigkeit des Motors in setup-Funktion, Werte von 1 bis 10:
Schrittmotor.setSpeed(Geschwindigkeitswert);

- Mit dem Befehl *Schrittmotor.step(Schritte)*; könnt ihr angeben, wie viele Schritte der Motor machen soll. Ein Minus ändert die Richtung.
- Erweitere nun dein Programm wie folgt:
 - Rollläden sollen automatisch bei Nacht oder zu starker Sonneneinstrahlung (also wenn es zu hell ist) herunterfahren.
Zusätzlich soll das Licht im Wohnzimmer eingeschaltet werden.
 - Rollläden sollen automatisch bei normalem Tageslicht wieder hochgefahren werden und das Licht soll ausgehen.
- *Hinweis:*
Speicher dir die Position der Rollläden ab.

- Du kannst den Motor von außen an das Haus schrauben.
- Nutze die Wellenkupplung, um die Metallwelle als Gardinenstange zu befestigen.
- Mit etwas Stoff und Clips kannst du die Gardine an der Stange anbringen.



4

- 1: eigene Darstellung
- 2: entnommen aus der Arduino IDE, <https://www.arduino.cc/en/software>
- 3: entnommen aus dem Software Fritzing, <https://fritzing.org/download/>
- 4: eigene Aufnahme