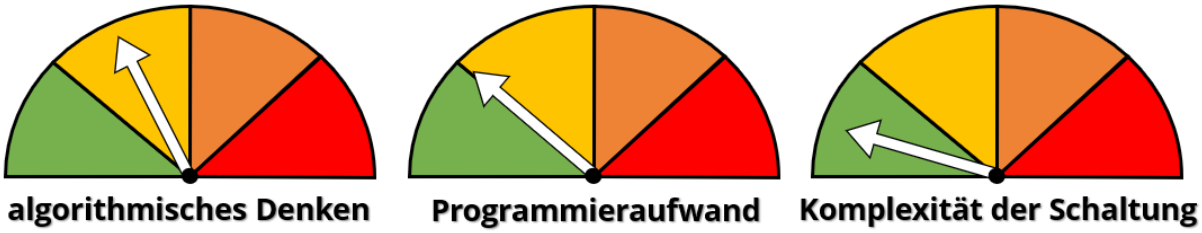
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Station 1 | Ist es heiß hier?



Professur für Didaktik der Informatik

SYLBER Synergetische Lehrerbildung im exzellenten Rahmen

Das Maßnahmenpaket „TUD-Sylber² – Synergetische Lehrerbildung im exzellenten Rahmen“ wird im Rahmen der gemeinsamen „Qualitätsoffensive Lehrerbildung“ von Bund und Ländern aus Mitteln des Bundesministeriums für Bildung und Forschung gefördert.

GEFÖRDERT VOM Bundesministerium für Bildung und Forschung

DIE ZUKUNFT IST JETZT – EIN ERSTER SCHRITT ZUM INTELLIGENTEN HEIM

Schon mal etwas von einem „Smart-Home“ gehört? Die Idee dahinter ist es die Gebäudetechnik zu automatisieren. Beispielhaft kann dies bedeuten, dass bei großer Sonneneinstrahlung die Jalousien allein herunterfahren oder mit fortschreitender Dunkelheit selbstständig das Licht angeht. Wir wollen uns jetzt etwas genauer mit der Temperaturmessung auseinandersetzen – dies könnte zum Beispiel zur Steuerung der Heizungs- oder Klimasystem genutzt werden, um stets die optimale Zimmertemperatur von 22°C im Haus zu haben.

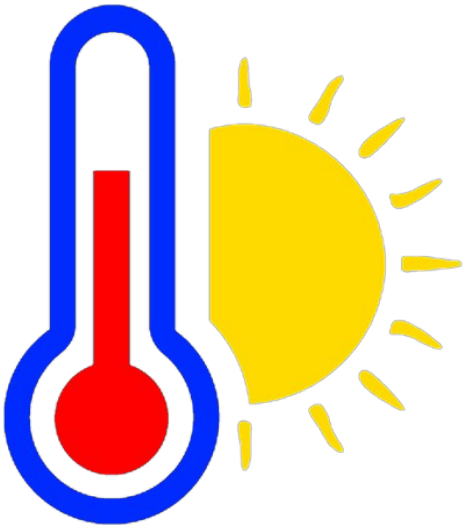
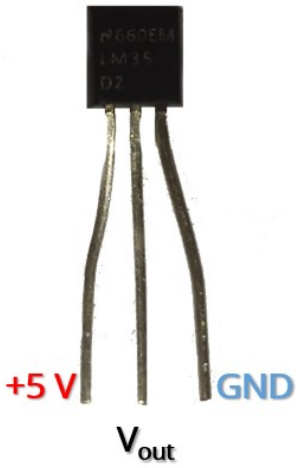


Abbildung 1: Sommertag

Wir wollen allerdings kein anderes Gerät steuern, uns genügt zunächst einmal zu wissen wie warm es gerade in unserem Raum ist. Dazu nutzen wir einen Temperatursensor, welchen du im folgenden näher kennenlernen wirst und geben die Informationen am Computer aus.

DER TEMPERATURSENSOR

Wir verwenden den Temperatursensor `TMP35`, dieser kann Temperaturen im Bereich von $0 \leq °C \leq 100$ erfassen. Er wird mit Betriebsspannung, sprich mit den `+5 V`, die der Arduino Uno liefert, betrieben. Er gehört zur Kategorie der „analogen Sensoren“. Dies bedeutet, dass wir als Ausgabe vom Sensor einen Spannungswert erhalten, den wir anschließend noch umrechnen müssen, um einen konkreten Temperaturwert zu erhalten – dazu später mehr. Am Sensor selbst seht ihr drei Anschlüsse, welche, wie in der rechtsstehenden Abbildung gezeigt, angeschlossen werden müssen.



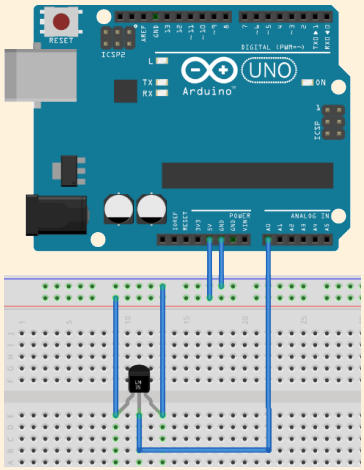
ACHTUNG!

**Auf dem Bild zeigt die flache Seite zu dir –
die andere Seite ist abgerundet!**

AUFGABE 1 – VERBINDEN DES TEMPERATURSENSORS

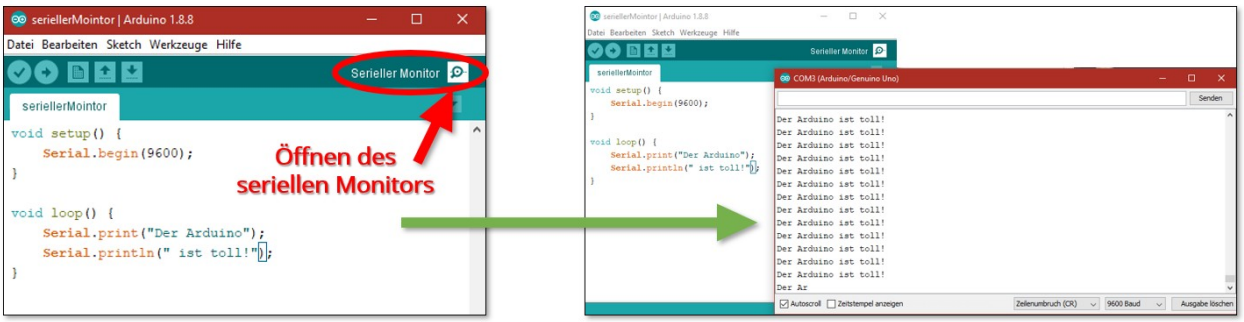
Deine erste Aufgabe ist es, den Temperatursensor mit dem Arduino zu verbinden. Verbinde dazu, wie aus der Einführung bereits bekannt, zunächst die äußeren Anschlussleisten und erst dann den Sensor. Trenne den Arduino vom Computer, wenn du deine Schaltung veränderst.

Außerdem muss ein analoger Sensor selbstverständlich auch mit einem analogen Pin verbunden werden.



Jetzt haben wir den Sensor verbunden. Schauen wir uns also mal an, was wir für Messwerte erhalten. Um die Daten an den Computer zu übertragen nutzen wir den sogenannten **seriellen Monitor**, welcher die Informationen, die über die USB-Verbindung übertragen werden, darstellen kann. Dazu benötigst du folgende Befehle:

Befehl	Wo?	Bedeutung
`Serial.begin(9600)`	`setup()`-Methode	Startet die Verbindung zum seriellen Monitor, wobei die 9600 für die Übertragungsrate steht.
`Serial.print("Der Arduino... ");`	`loop()`-Methode	Ausgabe einer Zeichenkette. Ohne Anstriche können auch Variablen ausgegeben werden.
`Serial.println(" … ist toll!");`	`loop()`-Methode	Gleiche Funktion wie oben, aber mit einem Zeilenumbruch am Ende.



Mit den Befehlen können wir nun eine Übertragung zum Computer realisieren. Nun müssen wir nur noch das Auslesen des analogen Sensors ermöglichen. Dies funktioniert quasi analog wie zum Auslesen des Taster-Status.

Zunächst initialisierst du eine Variable, welche den analogen Pin enthält, an dem du den seriellen Monitor angeschlossen hast.

Beispielhaft kann dies wie folgt aussehen: `int tempsensor = A0;`

Das Auslesen erfolgt dann mit dem Befehl: `analogRead(tempsensor);`

AUFGABE 2 – AUSGABE DER SENSORWERTE REALISIEREN

Kombiniere nun das eben gelernte, um eine Ausgabe der Sensorwerte zu realisieren. **Dazu musst du die folgenden Schritte erledigen → schreibe die Befehle zur Hilfe und Orientierung auf die Striche!**

- initialisiere eine neue Variable für den Temperatursensor:

_____

- lege den `pinMode()` für den Temperatursensor auf Input fest:

_____

- starte die Verbindung zum seriellen Monitor:

_____

- lies den analogen Sensorwert ein (in `loop()`-Methode, da dauerhaft gemessen wird):

_____

- gib den Sensorwert im seriellen Monitor aus:

_____

Teste durch Erwärmen des Sensors mit den Händen die Funktionalität.

TEMPERATURWERTE AUS DEN ANALOGEN WERTEN ERMITTELN

Hmm... die Werte des Sensors machen nicht wirklich Sinn? Das stimmt natürlich. Woran liegt das? Erinnern wir uns daran zurück, was einen analogen Sensor auszeichnet. Dieser gibt Spannungswerte zurück, welche durch den 10-Bit-Analog-Digital-Wandler umgerechnet werden.

ANALOGE SENSOREN UND ANALOGE SIGNALE

Digitale Signale sind dir ja bereits bekannt. Analoge Signale können nicht nur die Werte 0 und 1 annehmen. Dabei handelt es sich um Spannungswerte im Bereich von $0 \leq V \leq 5$. Um die analogen Spannungswerte aufzunehmen besitzt der Arduino einen sogenannten „10-Bit-Analog-Digital-Wandler". Dieser digitalisiert das Eingangssignal, wobei die 10 Bit der Anzahl der möglichen aufzunehmenden Werte entsprechen. In diesem Fall handelt es sich dabei um 2^{10} = 1024 Werte. Der erste Wert entspricht dabei 0 V und der 1024-igste Wert 5 V.

Demnach erhalten wir nur Werte im Bereich von 0 bis 1023. Überlegen wir uns also nun, wie wir aus den bekannten Informationen eine Temperatur konstruieren können. $0\,°C$ entsprechen einer Spannung von $0\,V$ und $100\,°C$ entsprechen $1\,V$. Dies lässt sich daraus konstruieren, dass ein Temperaturanstieg von $1\,°C$ einer Spannungsveränderung von $10\,mV$ entspricht. Die Temperatur kann also wie folgt berechnet werden: $T = 100 * U\,°C$ Das Formzeichen U steht für die Spannung.

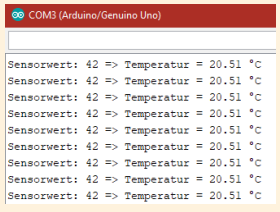
Die Umrechnung des analogen Sensorwertes kann mit einer Verhältnisgleichung realisiert werden: $\frac{Sensorwert}{1024} = \frac{U}{5}$

Verknüpfen wir beide Formeln, ergibt sich folgende Rechenvorschrift für die Temperatur: $T = \frac{5}{1024} * Sensorwert * 100\,°C$

AUFGABE 3 – TEMPERATUR BERECHNEN

Ergänze dein bisheriges Projekt nun um die Berechnung. Beginne damit eine Gleitkommazahl-Variable zu initialisieren und gib ihr zunächst den Wert 0.0. **Warum?** - Das Teilen durch 1024 liefert eine Zahl mit Nachkommastellen und nicht nur eine Ganzzahl. Beachte beim Dividieren, dass du mindestens eine Zahl mit Kommastelle angeben musst, zum Beispiel: $5.0/1024$. Nur so rechnet der Arduino das Ergebnis korrekt aus.

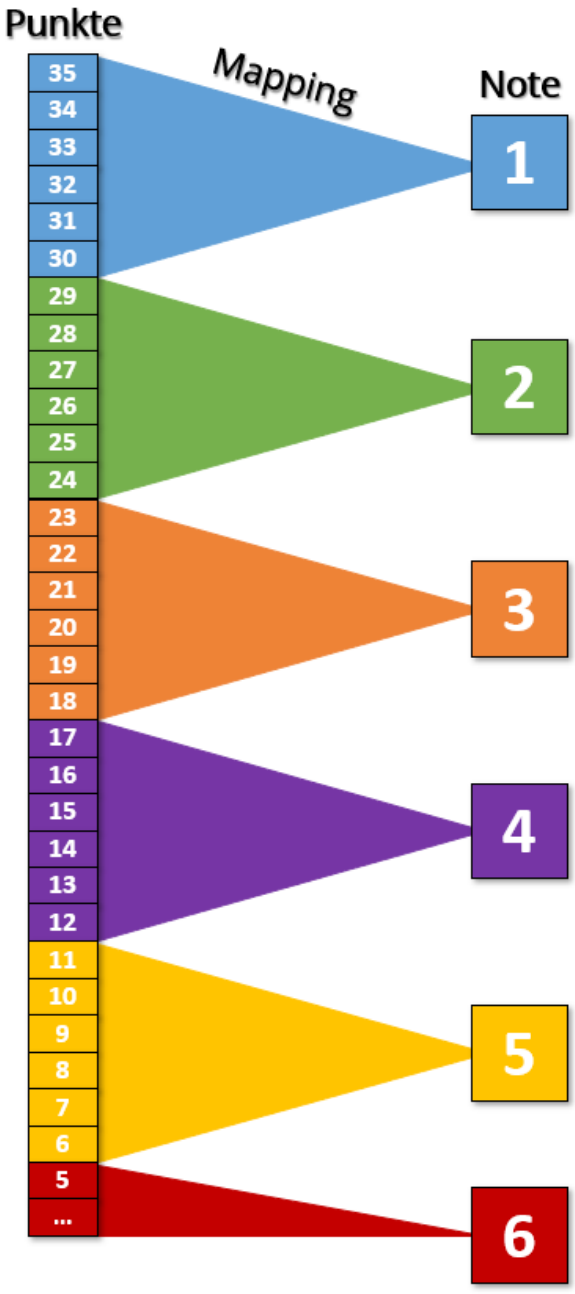
Anschließend folgt deine Berechnung **nach** dem Einlesen des Sensorwertes. Gib dein berechnetes Ergebnis und den Sensorwert im seriellen Monitor aus. Dies sollte etwa so aussehen, wie es in rechtsstehender Abbildung gezeigt ist.



Es ist natürlich nicht der eleganteste Weg für jeden analogen Sensor eine neue Berechnungsvorschrift zu konstruieren. Aus diesem Grund beschäftigen wir uns nun mit dem sogenannten **Mapping**.

DAS MAPPING

Das **Mapping** ist ein sehr wichtiges informatisches Prinzip. Grundsätzlich geht es dabei darum, Werte von einem Bereich auf einen anderen abzubilden. Das ist zu abstrakt? Verständlich. Machen wir es uns deshalb einmal vereinfacht anhand von Punktetabellen und Noten klar. Nehmen wir an, dass bei einem Test maximal 35 Punkte erreicht werden konnten und es dafür die Noten 1 bis 6 gibt, dann könnte eine gleichmäßige Verteilung wie in der Abbildung rechts aussehen.



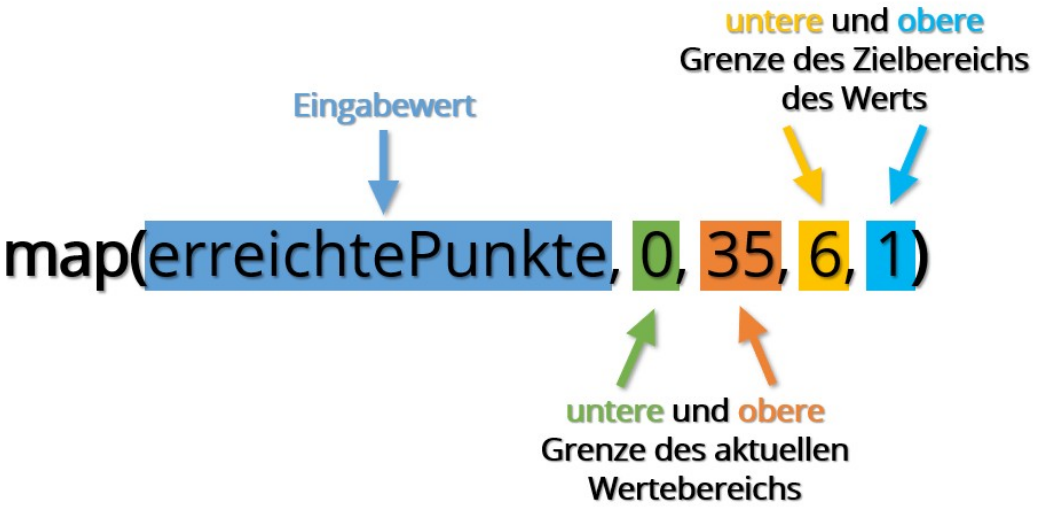
Wir haben 36 verschiedene Punktwerte, denn wir müssen auch null Punkte als mögliche Punktzahl mit einschließen. Diese Punktwerte wollen wir nun gleichmäßig auf sechs Noten verteilen. Dazu müssen wir lediglich die Anzahl möglicher Punkte durch die Anzahl der Noten teilen. Sprich: 36 geteilt durch 6 ergibt 6, was bedeutet das einer Note 6 verschiedene Punktzahlen zugeordnet werden.

DIE MAP-FUNKTION

Damit dies nicht immer händisch berechnet werden muss, gibt es dafür auch einen Befehl, der dies für uns realisiert.

Dabei handelt es sich um die `map()`- Funktion, wobei eine Funktion eine Folge von Berechnungsvorschriften ist, welche übergebene Daten verarbeitet und den berechneten Funktionswert als Ergebnis zurückgibt. Diese Funktion realisiert quasi dasselbe, was wir durch einfache Überlegung und einfache Berechnung mit den Punkten und Noten realisiert haben.

Die Funktion wird dabei mit folgenden Werten aufgerufen:



und wird dabei eine Note zurückgeben. Etwas trickreich ist die obere und untere Grenze des Zielbereichs, da man beim deutschen Notensystem bedenken muss, dass die Note 6 der kleinsten Punktzahl entspricht und damit die untere Grenze repräsentiert.

BERECHNUNG HINTER DER `MAP()`-FUNKTION

Die genaue Berechnung innerhalb der Funktion sieht wie folgt aus:

$$ausgabe = \frac{(eingabe - aktWB_{min}) * (zielWB_{max} - zielWB_{min})}{(aktWB_{max} - aktWB_{min})} + zielWB_{min}$$ WB ... Wertebereich

... eingesetzt mit Werten aus unserem Notenbeispiel:

$$ausgabe = \frac{(eingabe - 0) * (1 - 6)}{(35 - 0)} + 6 = \frac{eingabe * (-5)}{35} + 6$$

... und beispielhaft für eine erreichte Punktzahl von 15 Punkten:

$$ausgabe = \frac{15 * (-5)}{35} + 6 = -2.15 + 6 = 3.86$$ → entspricht **gerundet** der Note 4!

Jetzt kannst du diese Funktion auf jeden beliebigen Sachverhalt anwenden. Du weißt ja schon das der Arduino 1024 Werte digitalisiert, wodurch dein erster Wertebereich immer schon festgelegt ist mit 0 bis 1023. Der Zielwertebereich entspricht dann dem Messbereich des Sensors.

AUFGABE 4 – NUTZEN DER `map()`-FUNKTION

Überlege dir zunächst die Parameter der `map()`-Funktion und trage sie ein:

`map(_____,_____,_____,_____,_____)`

Ändere dein Programm nun so ab, dass der Wert der Ausgabe-Variable mit Hilfe der Funktion berechnet wird.

DAS POTENTIOMETER

Dabei handelt es sich um ein Bauteil, bei dem man mit Hilfe eines Drehmechanismus den elektrischen Widerstand verändern kann. Je nach Position des Reglers erhalten wir einen Wert zwischen `0V` und `+5V` am mittleren Pin des Potentiometers. Wir können also, identisch zum Temperatursensor, auch hier verschiedene Wertebereiche aufeinander abbilden.

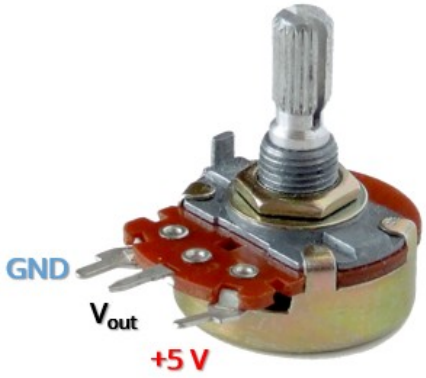


Abbildung 2: Potentiometer

Wozu wollen wir das konkret nutzen? Es ist natürlich alles andere als sinnvoll im Millisekunden-Takt die Temperatur zu messen. Allein wenn man über eine Speicherung der Daten nachdenkt, würden so enorm viele Daten zusammen kommen, obwohl sich die Temperatur natürlich nicht so schnell ändert. Neben dem Einfügen einer Pause zwischen den Messungen mit Hilfe der `delay()`-Funktion, wollen wir zusätzlich die Pausenzeit mit dem Potentiometer variieren.

AUFGABE 5 – PAUSENZEIT UND POTENTIOMETER

Zunächst musst du dir eine Variable für den analogen Sensorwert des Potentiometers anlegen, sowie eine für die Pausenzeit. Anschließend mappst du den Sensorwert auf einen Wertebereich, der prinzipiell von einer bis zu zehn Sekunden (hier kannst du auch gerne etwas variieren) gehen sollte. Trage deine Werte für die `map()`-Funktion unten ein:

`map(_____,_____,_____,_____,_____)`

Lass dir den eingestellten Pausenwert zusätzlich zur Temperatur, als Möglichkeit der Kontrolle, ausgeben.

Jetzt kann man sich natürlich darüber streiten, wie sinnvoll eine Variation der Pausenzeiten bei der Temperaturmessung ist. Ziel des Ganzen war es allerdings, dir einmal zu zeigen wie vielfältig du die `map()`-Funktion anwenden kannst und das diese eine große Bedeutung in der Informatik hat. Aber vielleicht hast du ja eine Idee für eine sinnvolle Anwendung des Potentiometers?

AUFGABE 6 – ANWENDUNG DES POTENTIOMETERS

Überlege dir ein sinnvolles Anwendungsszenario mit dem Potentiometer und schreibe dies in ein bis zwei **kurzen** Stichpunkten auf!

_____

_____

DU HAST NOCH NICHT GENUG VON DIESER STATION?

Kein Problem - Frage die Betreuer einfach nach dem Zusatzblatt! Dort wird es darum gehen, deine Messwerte zu verbessern, in dem du den Durchschnitt von mehreren Temperaturmessungen bildest.

Grafik auf dem Deckblatt: *thirsty, Conmongt, Pixabay License, https://pixabay.com/de/service/license/, https://pixabay.com/de/illustrations/durstig-trinken-wasser-bettler-4294637/*
Abbildung 1: *Heat, Leonardine36, Pixabay License, https://pixabay.com/de/service/license/, https://pixabay.com/de/illustrations/w%C3%A4rme-temperatur-heizung-3092081/*
Abbildung 2: *A potentiometer, lainf, CC BY 2.5, https://creativecommons.org/licenses/by/2.5/, https://de.wikipedia.org/wiki/Potentiometer#/media/Datei:Potentiometer.jpg*
Screenshots: *fritzing electronics made by easy und Arduino IDE 1.8.12 (windows)*
Alle weiteren Grafiken: *Patrick Binkert, EduInf@TUD*