DIMMER-SCHALTER

In vielen Wohnungen und Bürogebäuden werden die Lichter über einen Dimmerknopf gesteuert, mit dem Sie die Beleuchtung auf eine für die Umwelt angenehme Einstellung dimmen können. In dieser Lektion untersucht ihr analoge Signale und wie sie zur Steuerung einer Schaltung verwendet werden können. Ihr erstellt einen LED-Schaltkreis, der die LEDs dimmt, wenn ihr an einem als Potentiometer bezeichneten Gerät dreht.

VORAUSSICHTLICHE ZEIT **90-135 min**

SIE WERDEN ETWAS DARÜBER LERNEN

BEDINGTE ANWEISUNG, SERIELLE KOMMUNIKATION, POTENTIOMETER, VARIABLE, ANALYSE VON DATEN, SPEICHERUNG VON INFORMATIONEN

Übersicht

Die meisten Fahrzeuge verfügen über Knöpfe am Armaturenbrett oder Bedienfeld, mit denen sich die Lautstärke des Radios, die Geschwindigkeit der Wischerblätter, die Temperatur im Fahrzeuginneren und die Drehgeschwindigkeit der Ventilatoren, die die Luft durch die Lüftungsöffnungen drücken, regeln lassen. In vielen Wohnungen und Bürogebäuden werden die Lichter über einen Dimmerknopf gesteuert, mit dem Sie die Beleuchtung auf eine für die Umwelt angenehme Einstellung dimmen können.

In dieser Lektion untersucht ihr analoge Signale und wie sie zur Steuerung einer Schaltung verwendet werden können. Ihr erstellt einen LED-Schaltkreis, der die LEDs dimmt, wenn ihr an einem als Potentiometer bezeichneten Gerät dreht. Ihr werdet auch einen Sketch in der Arduino-Software (IDE) schreiben, der analoge Ein- und Ausgänge verwendet, um die Helligkeit der LEDs einzustellen. Während ihr den Programmcode für euren Sketch schreibt, werdet ihr mit neuen Programmierkonzepten vertraut gemacht, wie z.B. der Verwendung von Variablen zum Speichern von Informationen, der Verwendung der seriellen Kommunikation,

um Informationen von dem Arduino Board zurück zum Computer zu senden, und der Verwendung komplexerer Bedingungen, um bestimmte Verhaltensweisen in bestimmten Situationen zu erzeugen. Schließlich werdet ihr die Schaltung so modifizieren, dass sie als LED-Array fungiert, bei dem die LEDs auf der Grundlage des analogen Wertes des Potentiometers nacheinander aufleuchten.

TEACHER NOTES

In dieser Lektion werden die Schülerinnen und Schüler mit Potentiometern vertraut gemacht und erfahren, wie diese zur manuellen Steuerung eines Stromkreises verwendet werden können. Die Schülerinnen und Schüler bauen eine LED-Schaltung, bei der das Arduino UNO R3 Board die Helligkeit der LEDs je nach Stellung eines Potentiometers steuert. Während die Schülerinnen und Schüler ihren Stromkreis codieren, werden sie mit Konzepten wie Variablen, bedingten Anweisungen, Lesen und Verwenden analoger Eingangssignale und serieller Kommunikation vertraut gemacht. Nachdem ihr Stromkreis und ihr Sketch fertiggestellt sind, verwenden die Schülerinnen und Schüler ein Multimeter, um weiter zu untersuchen, wie das Potentiometer die LEDs in der Schaltung steuert.

ZEITAUFWAND FÜR DIE LEKTION

Übersicht und Fachbegriffe	3 minuten
Testen des Potentiometers	5 minuten
Dimmer-Stromkreis	
Benötigte Materialien	2 minuten
Aufbau der Schaltung	15 minuten
Code-Erstellung - Ausblenden der LEDs	18 minuten
Code-Erstellung - Serielle Kommunikation	10 minuten
Code-Erstellung - Blinkender Indikator	15 minuten
Testen und ändern	20 minuten
Aufräumen	2 minuten
Gesamte Lektion	**90 minuten**

Wenn Sie diese Lektion in zwei Unterrichtsstunden absolvieren, sollten die Schülerinnen und Schüler den Abschnitt Code-Erstellung - Serielle

Kommunikation bis zum Ende der ersten Unterrichtsstunde abgeschlossen haben..

LERNZIELE

- Zwischen digitalen und analogen Signalen unterscheiden.
- Untersuchen, wie Potentiometer Spannung und Widerstand beeinflussen.
- Einen dimmbaren Lichtkreis konstruieren und damit experimentieren.
- Programmiervariablen zum Speichern von Informationen verwenden.
- Die Entscheidungsfindung bei der Codierung untersuchen.
- Bedingte Anweisungen in Code schreiben, um bestimmte Situationen zu prüfen und zu kontrollieren.
- Die serielle Kommunikation zur Analyse von Daten und Informationen innerhalb eines Programms verwenden.
- Code debuggen, der Fehler enthält.

Fachbegriffe

- **Baudrate -** bei serieller Kommunikation die Anzahl der Informationsbits, die pro Sekunde zwischen dem Computer und dem Arduino UNO R3 Board übertragen werden
- **Bit -** eine Binärziffer (entweder 1 oder 0); die kleinste Dateneinheit für elektronische Geräte
- **Byte -** eine Gruppe von acht Bits
- **Potentiometer -** ein elektrisches Gerät mit variablem Widerstand, das häufig zur Steuerung der Spannung in einer Schaltung verwendet wird
- **Umfang -** definiert, wo eine Variable innerhalb eines Computerprogramms oder eines Arduino Sketches verwendet werden kann
- **serieller Monitor -** ein Werkzeug in der Arduino IDE, mit dem Informationen vom Arduino UNO R3 Board anzeigt werden können
- **Variable -** in der Programmierung ein Container mit einem Namen, der zur Speicherung von Informationen oder Daten verwendet wird

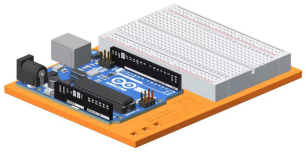
TEACHER NOTES

Es gibt viele Übungsmöglichkeiten zu den Fachbegriffen für die Schülerinnen und Schüler. Allerdings ist die Zeit dafür im 90-minütigen Rahmen für diese Lektion nicht berücksichtigt. Diese Übungen können vielleicht in den normalen Unterricht einfließen oder in Eigenarbeit erledigt werden.

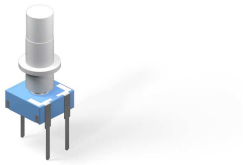
Testen des Potentiometers

In dieser Übung werdet ihr ein Multimeter verwenden, um zu untersuchen, wie ein Potentiometer als variabler Widerstand wirkt.

Benötigte Materialien



1 ARDUINO PROJEKT BOARD



1 POTENTIOMETER

1 MULTIMETER MIT PRÜFLEITUNGEN

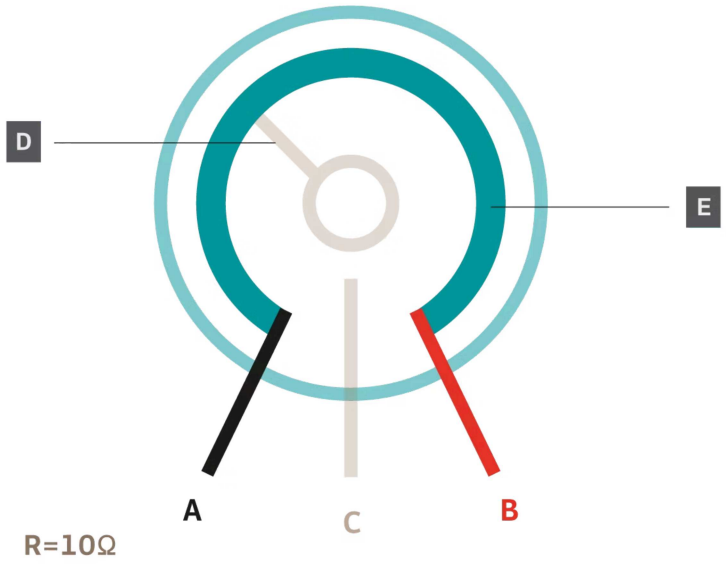
1) Findet ein Potentiometer. Das Potentiometer hat zwei Stifte auf der einen Seite und einen einzelnen Stift auf der anderen. Eventuell müsst ihr einen weißen Knopf an den Stift des Potentiometers anbringen.

Ein **Potentiometer** ist eine Art variabler Widerstand. Mit anderen Worten kann es so verändert werden, dass es keinen Widerstand oder einen sehr großen Widerstand hat. Die Potentiometer in eurem Bausatz haben einen Bereich von 0 bis 10.000 Ohm.

FURTHER NOTES

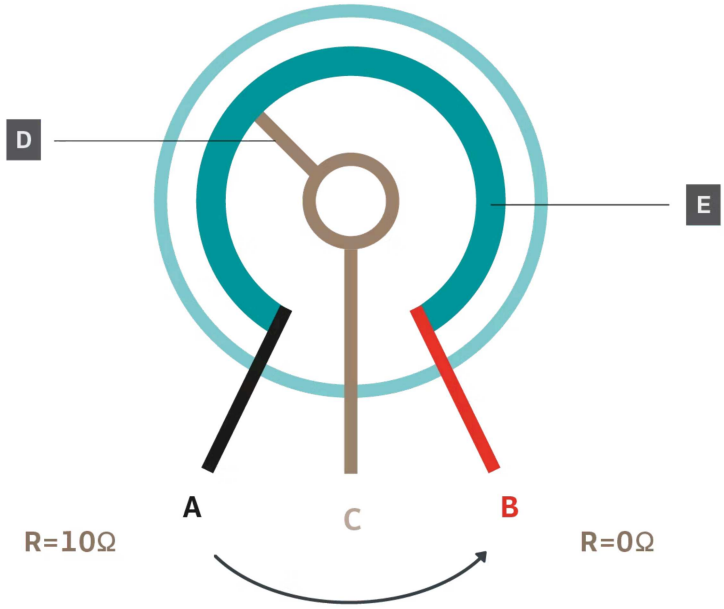
Die meisten Potentiometer sind mit ihrem Bereich gekennzeichnet. Die Potentiometer im Bausatz sind als 10K-Potentiometer gekennzeichnet, d.h. 10 Kiloohm oder 10.000 Ohm.

Im Inneren eines Potentiometers befindet sich ein Widerstandsmaterial und ein Wischer, der an diesem Material entlang gleitet. Jedes Ende des Widerstandsmaterials ist mit einem Stift oder einer Klemme verbunden. In der folgenden Abbildung wären dies die Pins A und B. Der Widerstand zwischen den Pins A und B ist festgelegt und ist der maximale Widerstandswert, den ein Potentiometer der Schaltung hinzufügen kann. Für die Potentiometer in eurem Bausatz beträgt der maximale Widerstand 10 Kiloohm.



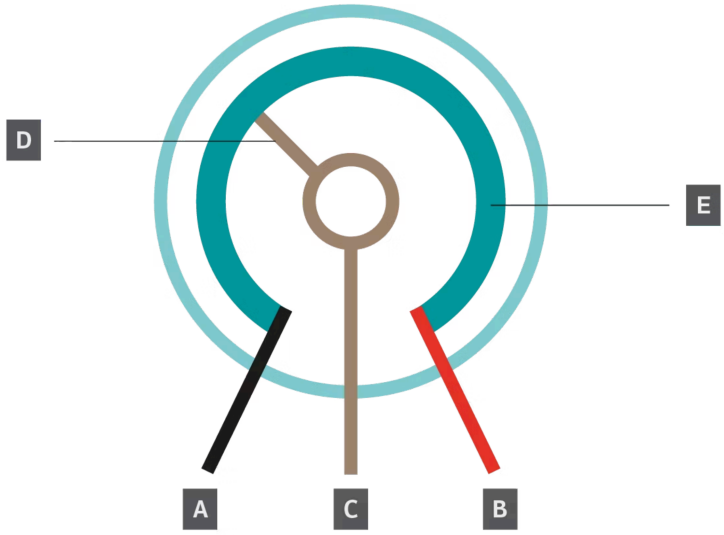
D) Wischer E) Widerstandsstreifen

Ein dritter Pin (Pin C in der Abbildung) ist mit einem Schleifer verbunden. Der Widerstand durch den Wischer, oder Pin C, hängt von der Position des Wischers beim Kontakt mit dem Widerstandsmaterial ab. Je grösser die Fläche des Widerstandsmaterials, die der Strom durchfließen muss, bevor er das Potentiometer durch den Schleifer verlässt, desto höher ist der Widerstand durch Pin C.



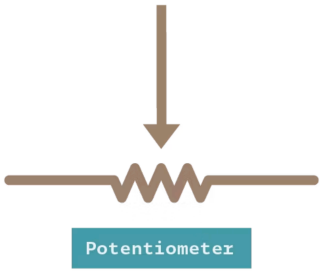
D) Wischer E) Widerstandsstreifen

Potentiometer ändern also den Widerstand in der Schaltung. Mikrocontroller lesen jedoch nicht den Widerstand, sondern die Spannung. In Lektion 3 habt ihr einen digitalen Pin verwendet, um die Spannung entweder als HIGH oder LOW zu lesen. Wie bringt man also ein Potentiometer dazu, die Spannung in einem Stromkreis zu beeinflussen? Indem man das Potentiometer an eine Stromquelle und an Ground anschließt. Der Wischer-Pin wird üblicherweise mit einem analogen Pin auf dem Arduino Board verbunden.

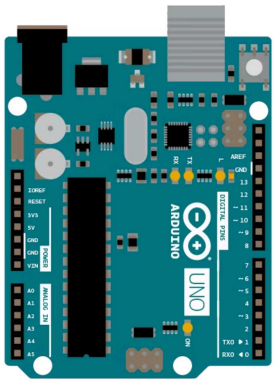


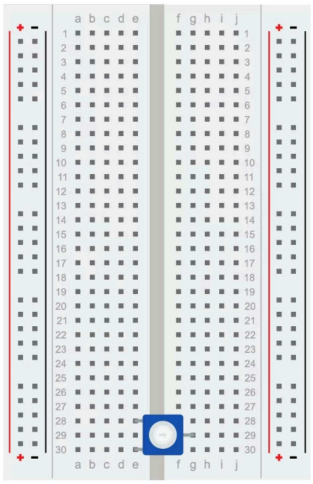
A) An Ground B) An Strom C) An analogen Pin D) Scheibenwischer E) Widerstandsstreifen

Das Ohm'sche Gesetz besagt, dass es eine gegensätzliche Beziehung zwischen Stromstärke und Widerstand gibt. Wenn der eine nach oben geht, geht der andere nach unten. Mit anderen Worten, wenn der Widerstand eines Potentiometers zunimmt, nimmt die Stromstärke ab. Der Mikrocontroller misst die Spannung über einen analogen Eingang und kann, je nach Programm, die Reaktion der Schaltung verändern. In Schaltplänen werden Potentiometer normalerweise durch ein Symbol dargestellt, das wie ein Widerstand aussieht, mit einem Pfeil, der in die Mitte des Widerstands zeigt.



2) Befestigt ein Potentiometer am Steckbrett. Die Seite mit den zwei Stiften sollte in die Löcher 28e und 30e gesteckt werden. Der andere Stift sollte in Loch 29g stecken.



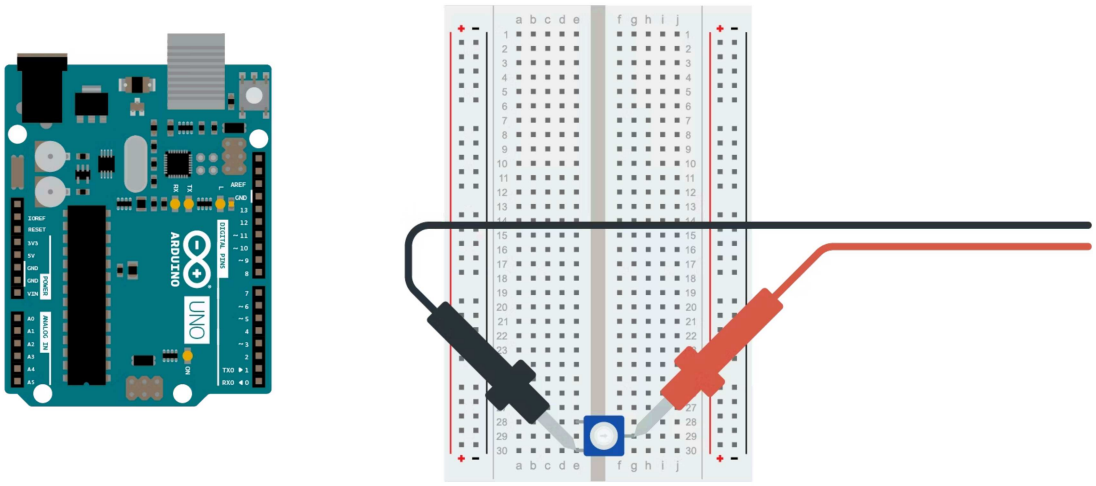


TEACHER NOTES

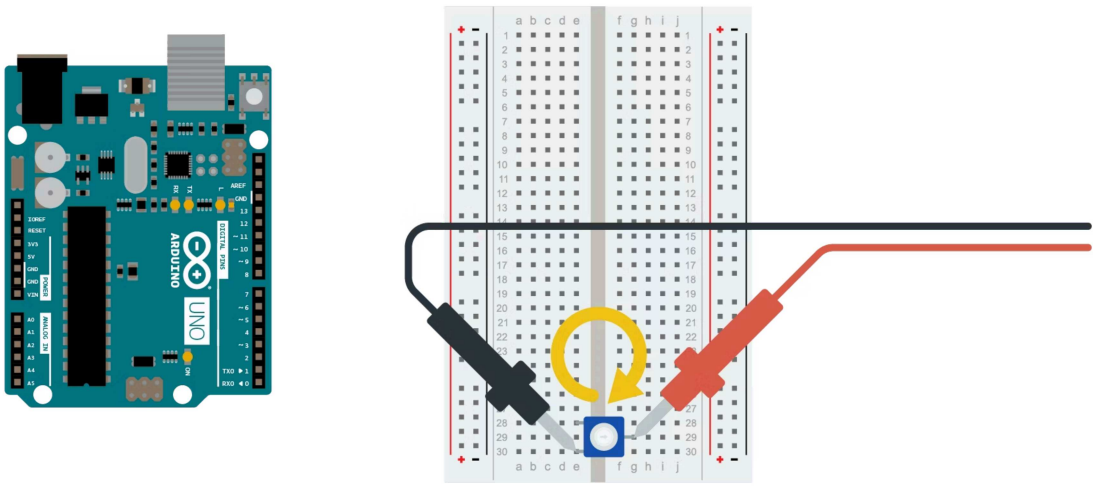
Das Potentiometer sollte nur in einer Ausrichtung auf das Steckbrett passen. Wenn die Stifte des Potentiometers nicht auf das Steckbrett passen, dreht das Potentiometer um 90 Grad.

3) Stellt das Multimeter dazu ein, den Widerstand zu messen. Stellt das Multimeter so ein, dass ein Bereich von 20 Kiloohm (20K) gemessen wird.

4) Verbindet die Prüfleitungen vom Multimeter mit den Anschlüssen des Potentiometers. Berührt mit dem roten Prüfkabel den einzelnen Stift. Dies ist der Wischerstift. Berührt mit dem schwarzen Prüfkabel einen der beiden Stifte auf der anderen Seite des Potentiometers.



5) Dreht langsam den Drehknopf. Beachtet dabei, wie sich der Widerstandswert ändert, wenn der Knopf gedreht wird.



TEACHER NOTES

Schülerinnen und Schüler, die individuell arbeiten, werden Schwierigkeiten haben, diese Übung auszuführen. Möglicherweise müssen Sie sie unterstützen oder sie mit einer Gruppe zusammenarbeiten lassen.

6) Dreht den Knopf ganz in eine Richtung und notiert den Widerstand. Dann dreht ihr den Knopf ganz in die andere Richtung. Wie groß ist der Bereich des Potentiometers?

7) Schaltet das Multimeter aus.

TEACHER NOTES

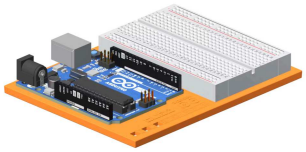
KLASSENDISKUSSION

Nachdem Sie den Bereich des Potentiometers bestimmt haben, rufen Sie ein paar Gruppen dazu auf, die Messwerte für den Bereich ihres Potentiometers bekannt zu geben. Hatte jede Gruppe den gleichen Bereich? Genau wie Widerstände werden auch Potentiometer unterschiedliche Toleranzen haben. Auch wenn diese Potentiometer mit 10 K (10 Kiloohm) bezeichnet sind, wird jedes Potentiometer einen Bereich haben, der 10 Kiloohm etwas übersteigt oder unterschreitet.

Dimmer-Stromkreis

In dieser Übung werdet ihr eine Schaltung bauen und programmieren, die mit Hilfe eines Potentiometers die Helligkeit einer Gruppe blauer LEDs steuert.

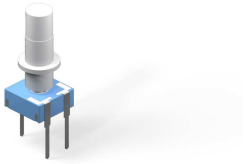
Benötigte Materialien



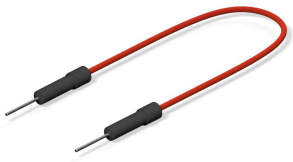
1 ARDUINO PROJEKT BOARD

3 220 Ω-WIDERSTAND

3 BLAUE LED

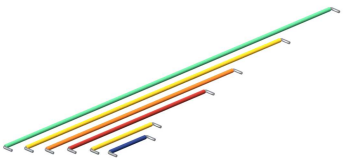


1 POTENTIOMETER



1 ROT STROMKABEL

1 SCHWARZE STROMKABEL



1 STECKVERBINDER

1 MULTIMETER MIT PRÜFLEITUNGEN

1 USB-KABEL

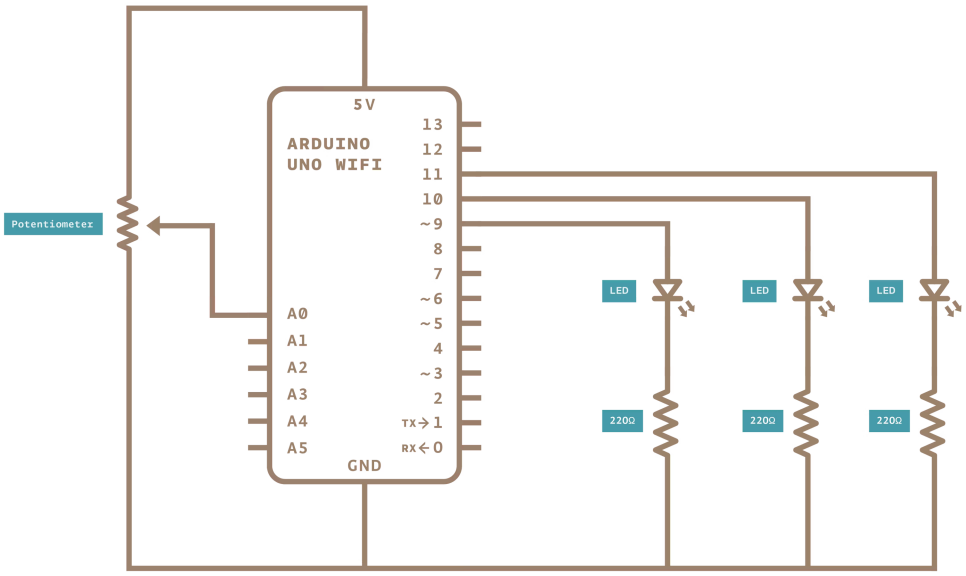
FURTHER NOTES

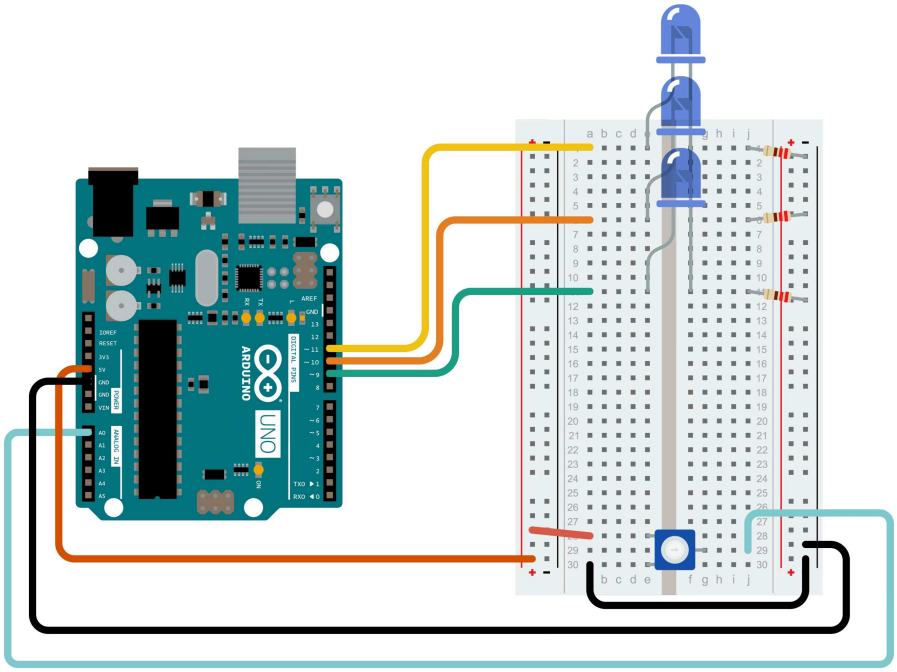
Die Farbbänder des 220-Ohm-Widerstands sind:

- **4 Bänder:** rot, rot, braun, gold
- **5 Bänder:** rot, rot, schwarz, schwarz, gold

Bei den Potentiometern ist der Knopf oft vom Rest des Potentiometergehäuses getrennt. Der Stecker des Knopfes sollte sich leicht in das Loch in der Mitte des Potentiometergehäuses schieben lassen.

Schaltplan

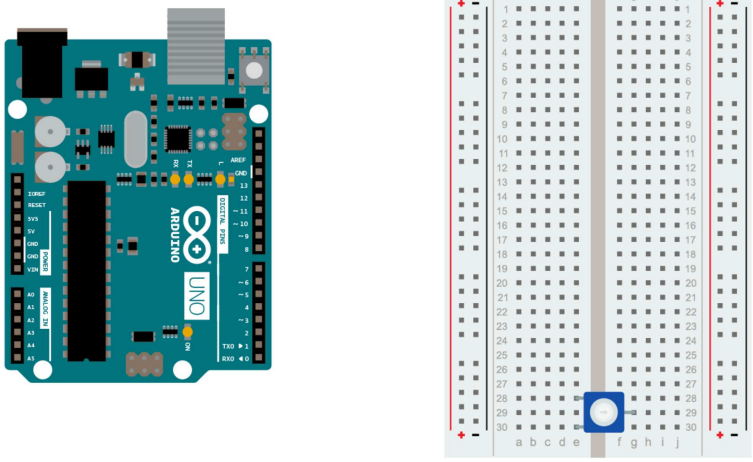




Aufbau der Schaltung

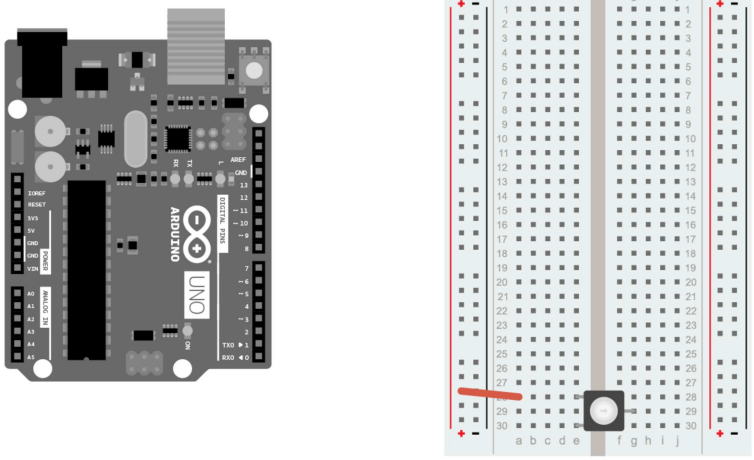
Step 1

Aus dem vorigen Abschnitt sollte das Potentiometer bereits am unteren Ende des Steckbretts angebracht sein.



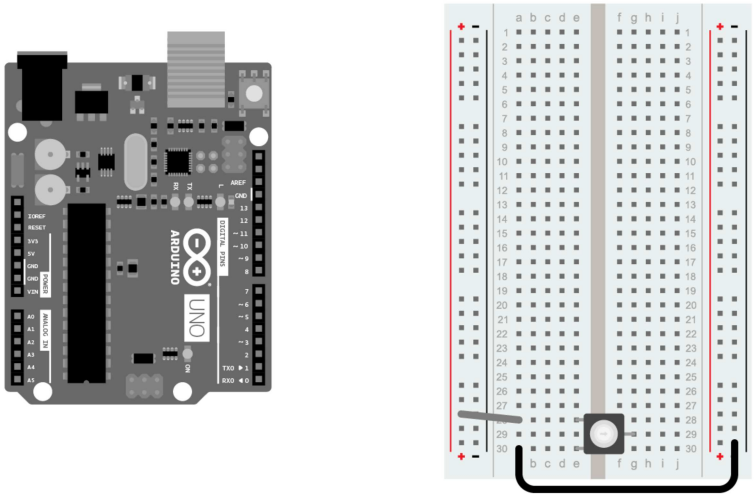
Step 2

Benutzt einen kurzen Steckverbinder, um das Loch 28a mit dem Pluspol auf der linken Seite des Steckbretts zu verbinden. Dadurch wird das Potentiometer mit der Stromquelle verbunden.



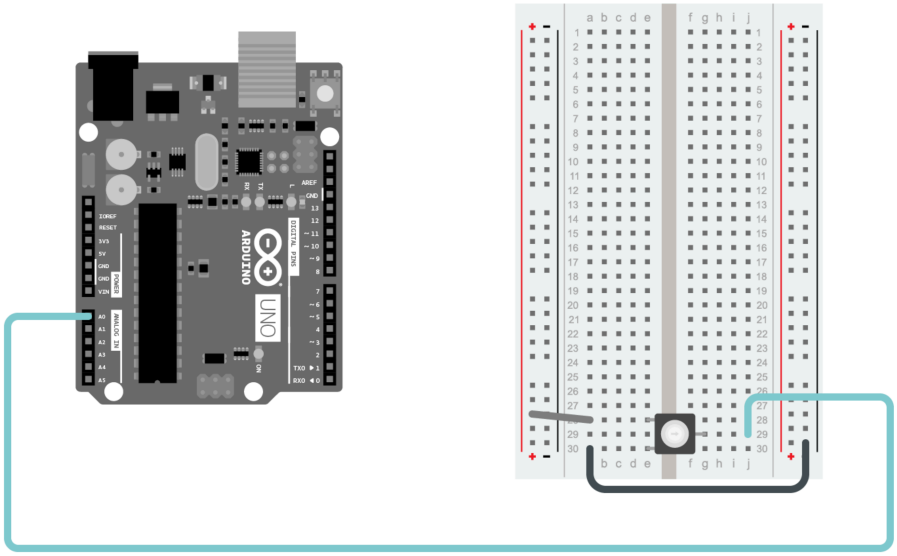
Step 3

Benutzt einen langen Steckverbinder, um Loch 30a mit dem Minuspol auf der rechten Seite des Steckbretts zu verbinden. Dadurch wird das Potentiometer mit Ground verbunden.



Step 4

Benutzt einen weiteren langen Steckverbinder, um Loch 29j mit Pin A0 auf dem Arduino UNO R3 Board zu verbinden.

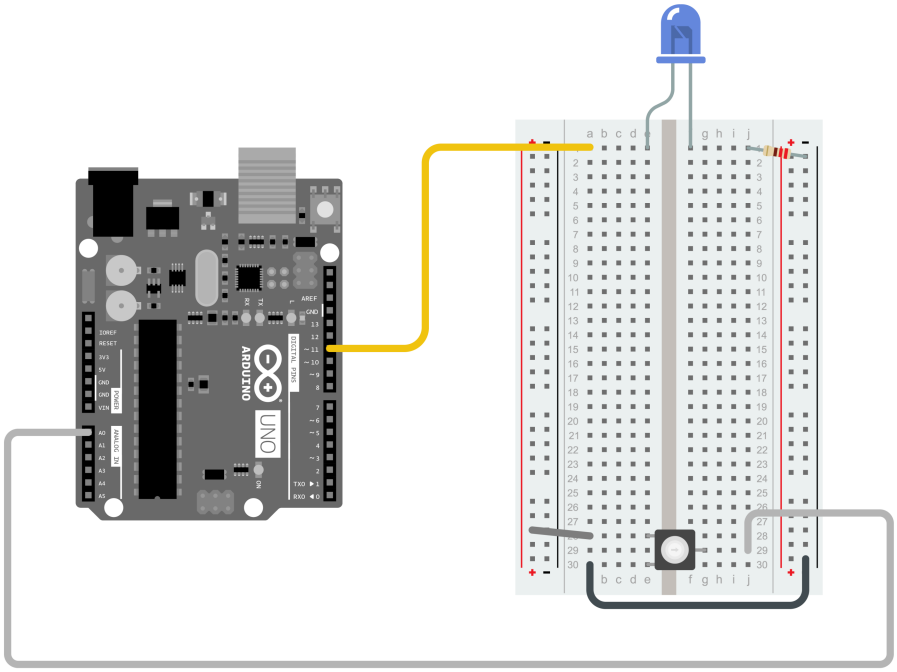


Step 5

Findet eine blaue LED. Steckt die Anode der LED in Loch 1e. Fügt die Kathode in Loch 1f ein.

Benutzt einen 220-Ohm-Widerstand, um die LED mit Ground zu verbinden.

Benutzt einen langen Steckverbinder, um die LED mit Pin 11 auf dem Arduino UNO R3 Board zu verbinden.

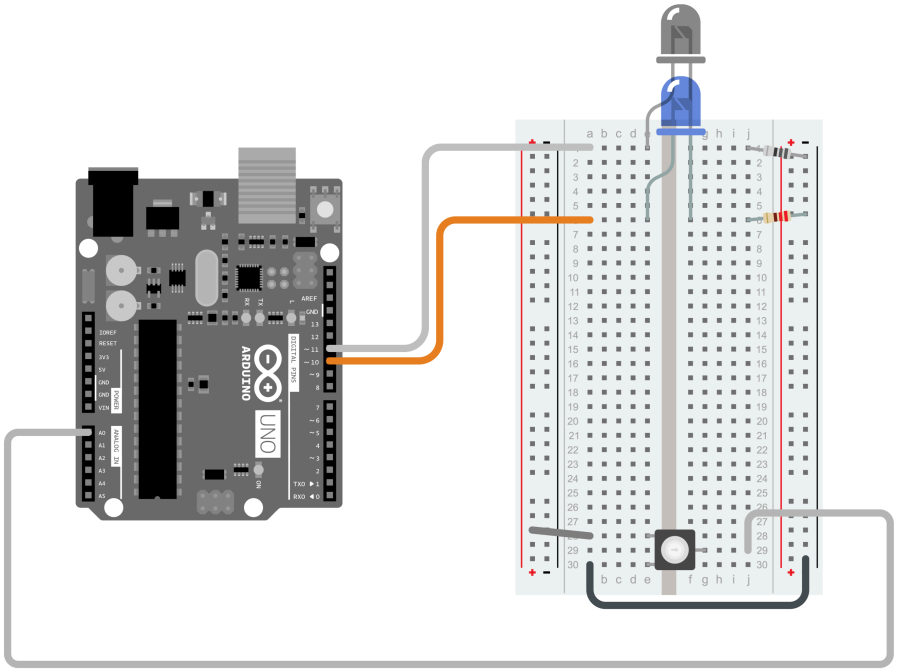


Step 6

Fügt eine weitere blaue LED in die Schaltung in Reihe 6 ein. Fügt die Anode in Loch 6e und die Kathode in Loch 6f ein.

Benutzt einen weiteren 220-Ohm-Widerstand, um die LED mit Ground zu verbinden.

Ihr solltet einen langen Steckverbinder verwenden, um die LED mit Pin 10 auf dem Arduino UNO R3 Board zu verbinden. Die zweite LED ist nun in der Schaltung verdrahtet.

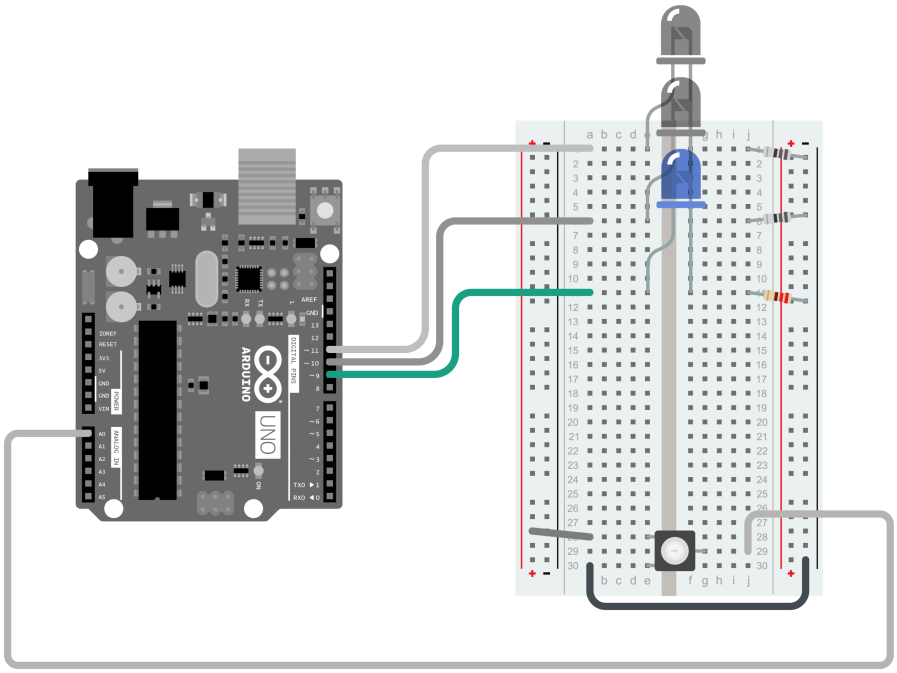


Step 7

Schließt eine dritte blaue LED an die Schaltung in Reihe 11 an. Stellt sicher, dass die Anode in Loch 11e und die Kathode in Loch 11f ist.

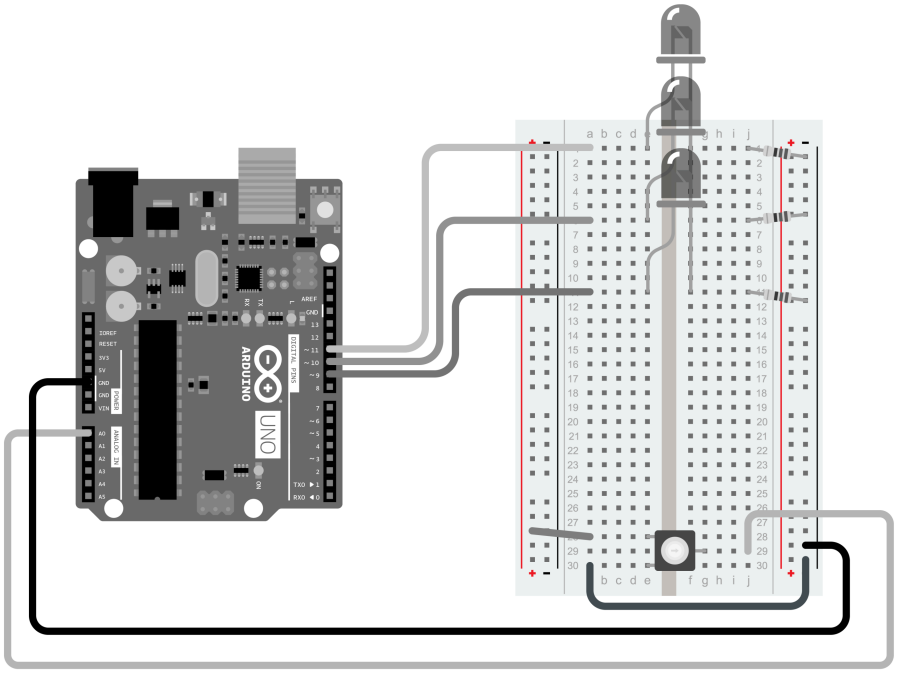
Verbindet die LED durch einen weiteren 220-Ohm-Widerstand mit Ground.

Benutzt einen langen Steckverbinder, um die dritte LED mit Pin 9 auf dem Arduino UNO R3 Board zu verbinden.



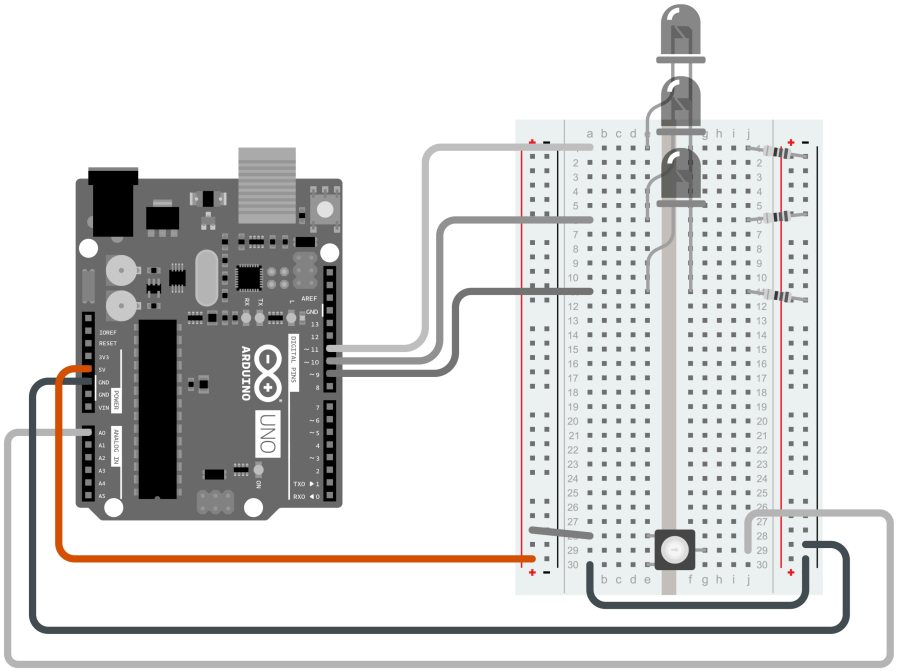
Step 8

Benutzt das schwarze Stromkabel, um den Minuspol auf der rechten Seite des Steckbrettes mit einem Ground Pin auf dem Arduino UNO R3 Board zu verbinden. Denkt daran, dass die Ground Pins mit GND beschriftet sind.



Step 9

Benutzt das rote Stromkabel, um den positiven Pol auf der linken Seite des Steckbrettes mit dem Fünf-Volt-Pin auf dem Arduino UNO R3 Board zu verbinden. Der Fünf-Volt-Pin ist mit "5V" beschriftet. Die Schaltung sollte nun vollständig sein.



TEACHER NOTES

Wenn die Schülerinnen und Schüler diesen Schaltkreis aufbauen, sollen sie genau darauf achten, wie die Komponenten auf dem Steckbrett platziert werden und warum diese Löcher im Vergleich zu anderen funktionieren. Dies wird die letzte Lektion sein, in der spezifische Lochkoordinaten erwähnt werden. Nach Abschluss dieser Aktivität müssen die Schülerinnen und Schüler ein gutes Verständnis dafür haben, wie das Steckbrett funktioniert, damit sie ihr Projekt in Lektion 5 abschließen können.

Code-Erstellung - Ausblenden der LEDs

Hinweis: In dieser Übung programmiert ihr die LEDs in eurem Stromkreis so, dass sie unter Verwendung des Potentiometers als Eingabegerät ein- und ausgeblendet werden. Der Pseudocode für dieses Programm könnte etwa so aussehen:

1) Erstellt Variablen, um Input- und Outputinformationen zu speichern. **2)** Setzt die drei Arduino UNO R3 Pins als Output Pins ein. **3)** Startet die Hauptschleife (main loop).

a) Speichert den Wert des Potentiometers als Variable. b) Bestimmt den Output-Wert zu den LEDs anhand der Potentiometervariablen. c) Stellt die LED-Helligkeit an Hand der Outputvariablen ein.

4) Beendet die Hauptschleife.

FURTHER NOTES

Wir werden in diese Kodierungskonzepte eingeführt:

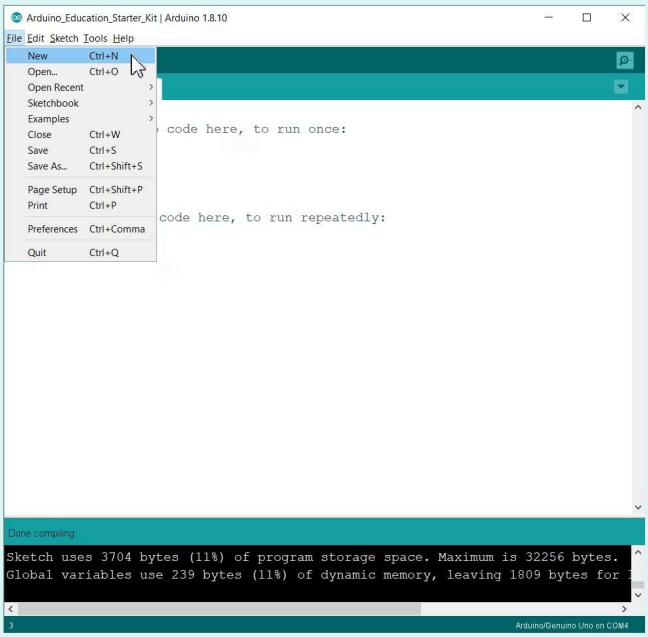
- **Variablendeklaration -** Deklariert den Typ und den Namen einer Variablen
- **Variablenzuweisung -** Weist einer Variablen einen Wert zu
- **analogRead() –** Liest einen analogen Wert aus der Schaltung
- **analogWrite() –** Sendet einen analogen Wert an die Schaltung

Wir werden diese Konzepte überprüfen:

- **pinMode() –** deklariert einen digitalen Pin als einen INPUT oder OUTPUT

1) Startet die Arduino IDE auf eurem Computer oder Gerät.

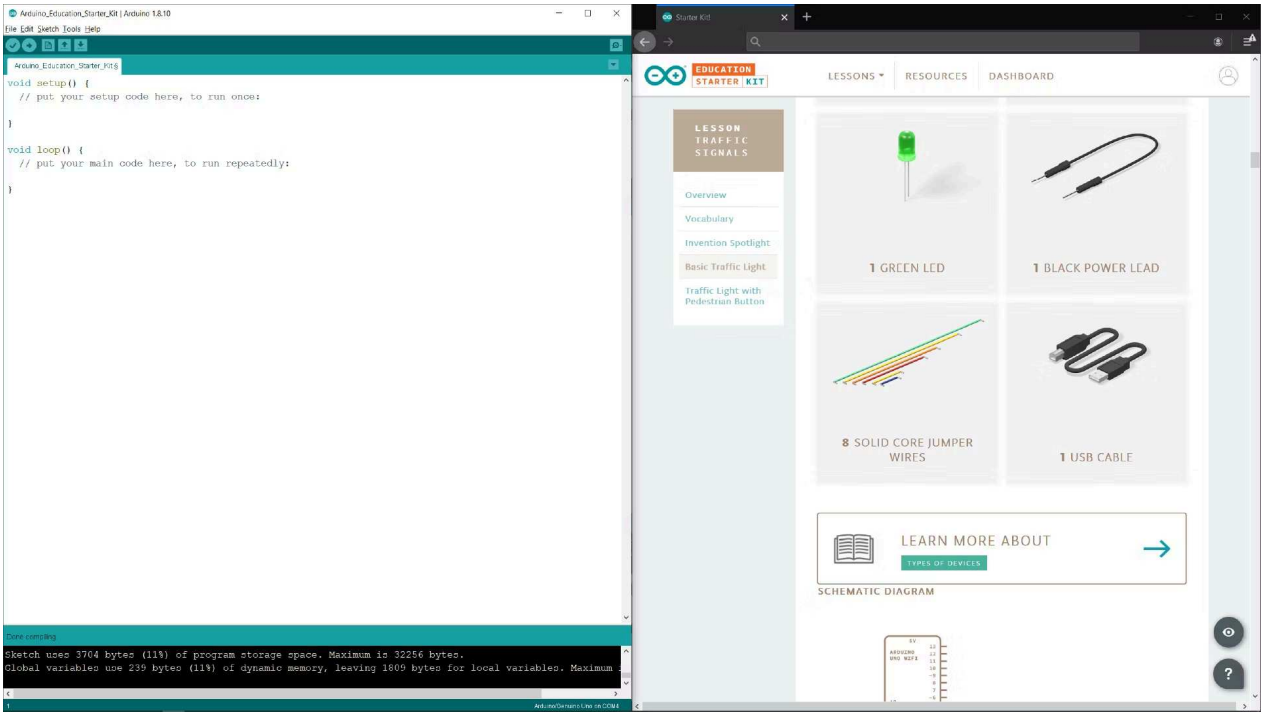
2) Wählt im Dateimenü die Option **Neu**. Dadurch wird ein neuer Sketch in einem neuen Fenster gestartet. Ihr könnt das andere Sketch-Fenster schließen.



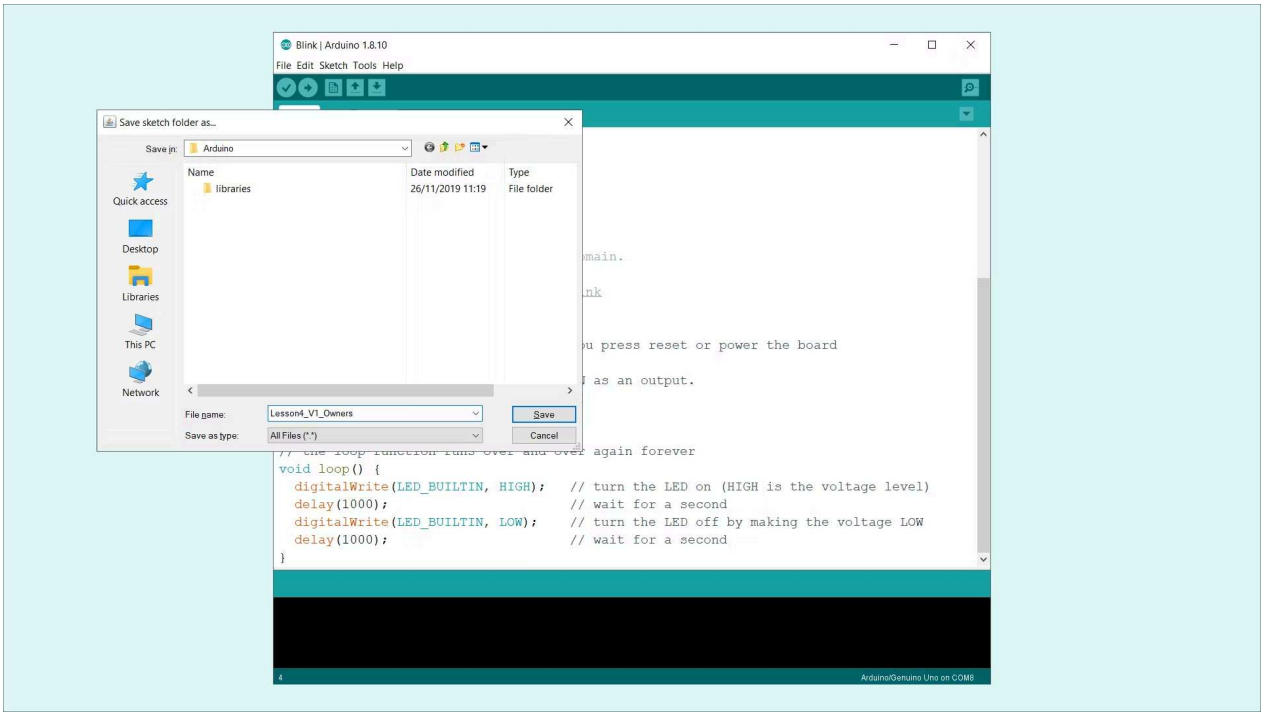
Es gibt zwei Methoden zur Ansicht der Arduino IDE und der Anweisungen in diesem Kurs. Verwendet die Methode, die für euch am besten geeignet ist.

Methode 1 - ihr könnt das Arduino IDE-Fenster auf den vollen Bildschirm maximieren. Dann könnt ihr mit der Taskleiste am unteren Bildschirmrand zwischen dem IDE-Fenster und den Kursanweisungen hin- und herschalten.

Methode 2 - ihr könnt das Arduino IDE-Fenster und die Kursanweisungen nebeneinander anordnen, so dass ihr beides gleichzeitig sehen könnt. Beide Fenster werden kleiner sein, was es möglicherweise schwieriger macht, sie zu sehen, aber der Vorteil ist, dass ihr nicht hin und her schalten müsst.



3) Speichert den Sketch unter einem neuen Namen. Wählt im Dateimenü die Option **Speichern unter...** Der Speicherort des Sketches sollte standardmäßig das Arduino Sketchbook sein. Wenn dies nicht der Fall ist, fragt eure Lehrerin oder euren Lehrer, wo ihr den Sketch auf eurem Computer speichern sollt. Benennt die Datei "Lektion4*V1*", gefolgt von euren Initialen und denen eurer Partner. Klickt auf **Speichern**.



4) Die LEDs in eurer Schaltung sind mit digitalen Pins auf dem Arduino UNO R3 Board verbunden. In Lektion 3 habt ihr gelernt, dass digitale Pins als Input oder Output

verwendet werden können. Die LEDs in eurer Schaltung sind Outputgeräte, so dass die digitalen Pins 9, 10 und 11 als Output Pins deklariert werden müssen. Verwendet in der Funktion **void setup()** den Befehl **pinMode()**, um diese Pins als OUTPUT zu deklarieren.

Das Potentiometer ist ein analoges Eingabegerät und wird an den analogen Pin A0 angeschlossen. Alle analogen Pins am Arduino sind Input Pins. Das bedeutet, dass sie nicht wie die digitalen Pins als INPUT deklariert werden müssen.

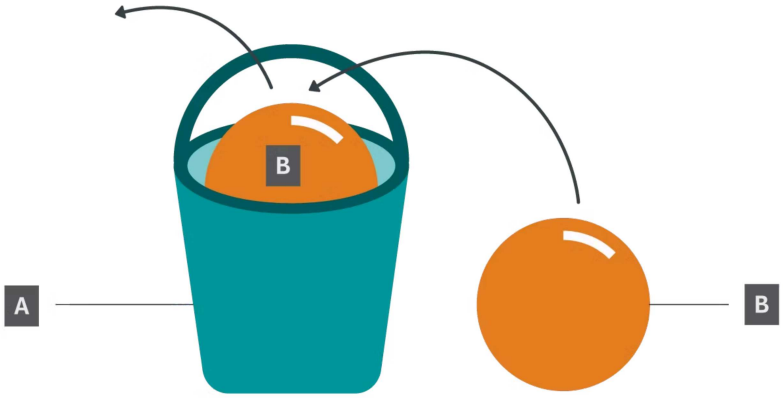
5) Fügt Code-Kommentare zu euren Pin-Deklarationen hinzu. Denkt daran, dass ihr zum Hinzufügen eines Code-Kommentars // eingeben müsst, und alles danach wird beim Kompilieren des Codes ignoriert. Zum Beispiel:

FURTHER NOTES

Das Hinzufügen von Code-Kommentaren ist eine wichtige Fähigkeit beim Programmieren und sollte immer gefördert werden. Es wird speziell in Form bestimmter Schritte in dieser Lektion hinzugefügt. In späteren Lektionen wird jedoch davon ausgegangen, dass die Schülerinnen und Schüler ohne direkte Anweisung selbst Kommentare hinzufügen.

6) Überprüft euren Code und stellt sicher, dass alle Syntaxfehler korrigiert wurden. Eure **void setup()**-Funktion sollte vollständig sein.

7) Um die LEDs mit dem Potentiometer auszublenden, braucht ihr eine Variable, um den Wert des Potentiometers zu speichern. Stellt euch beim Programmieren eine Variable wie einen Eimer vor, der eine Information enthält. Es gibt verschiedene Arten von Variablen. Variablen können zum Beispiel Zahlen, Buchstaben, Wörter, Sätze oder einfach wahr oder falsch sein. Für diese Anleitung werdet ihr nur den Integer-Typ der Variablen verwenden.



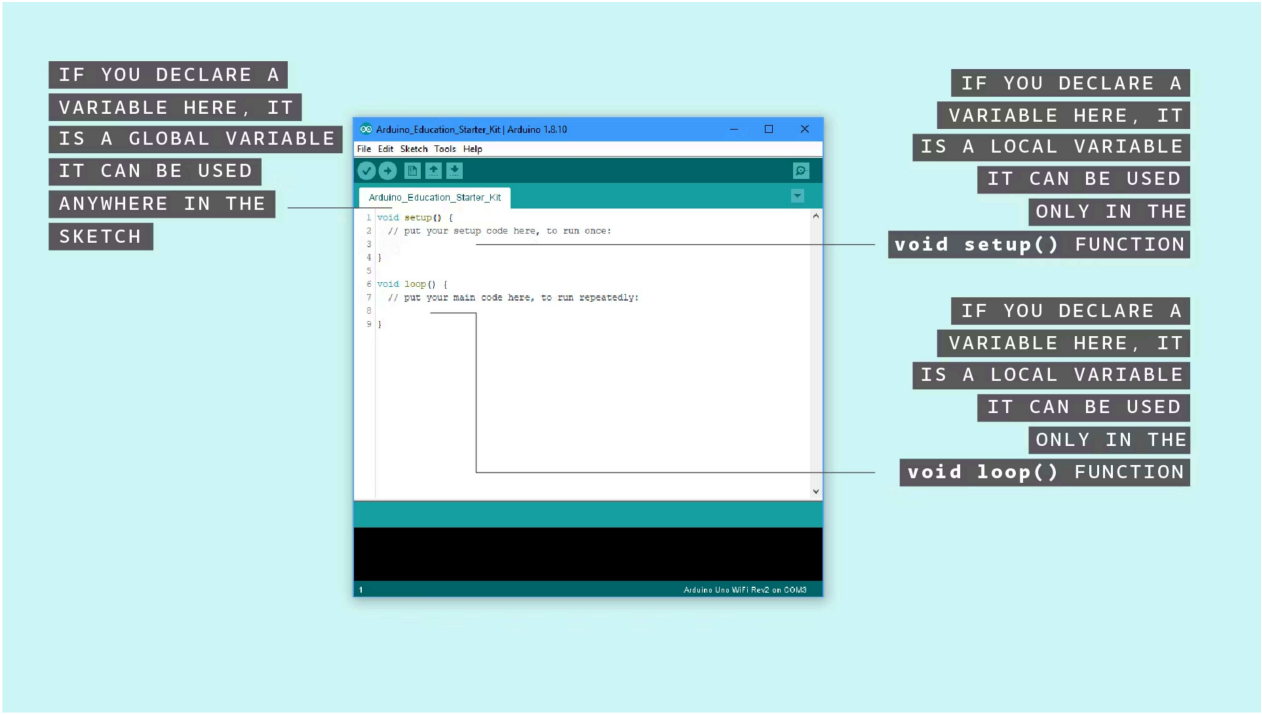
A) Variabel B) Daten

Variablen müssen deklariert werden, bevor sie verwendet werden können. Wenn eine Variable deklariert wird, wird je nach Art der deklarierten Variable ein Speicherplatz zur Aufnahme von Informationen erstellt. Variablen haben auch Namen. Variablennamen können beliebige Buchstaben- und/oder Zahlenkombinationen sein, solange der Name nicht ein Schlüsselwort, ein Befehl oder eine Funktion ist, die bereits in der Arduino IDE verwendet wird. Beispielsweise könnt ihr das Wort INPUT nicht als Variablenname verwenden, da INPUT eine andere Bedeutung hat.

> **Hinweis:** In der Arduino IDE sollten Variablennamen immer schwarz erscheinen. Wenn der Name orange oder blau ist, ist dieses Wort ein Schlüsselwort und kann nicht als Variablenname verwendet werden. Schlüsselwörter sind spezielle Wörter in einer Programmiersprache, die eine bestimmte Bedeutung haben. Beispiele für Schlüsselwörter sind Befehle wie **pinMode()** und Parameter wie **HIGH**.

Der **Geltungsbereich** einer Variable bestimmt, wo die Variable verwendet werden kann, und er hängt davon ab, wo ihr die Variable deklariert. Wenn eine Variable außerhalb aller Funktionen deklariert wird (mit anderen Worten, außerhalb aller geschweiften Klammern), ist sie eine globale Variable und kann überall in der Skizze verwendet werden. Wenn eine Variable innerhalb einer Funktion deklariert wird

(innerhalb geschweifter Klammern), ist sie eine lokale Variable und kann nur in dieser Funktion verwendet werden.



Hinweis: Lokale Variablen können nur innerhalb der Funktion verwendet werden, in der sie deklariert sind. Das heißt, ihr könntet eine Variable mit dem gleichen Namen in verschiedenen Funktionen im Programm deklarieren.

Im Allgemeinen sollten alle eure Sketche für diesen Kurs einem gleichförmigen Muster folgen, bei dem zuerst Variablen deklariert werden, gefolgt von der Funktion **void setup()** und schließlich der Funktion **void loop()**.

Geht vor der Funktion **void setup()** an den Anfang eures Sketches. Hier werdet ihr eure Variable deklarieren, um den Wert des Potentiometers zu speichern. Gebt Folgendes am Anfang eures Sketches ein:

Ihr habt gerade eine Integer-Variable namens **readValue** deklariert und diese Variable auf Null gesetzt. Ihr werdet die Variable etwas später in dem Sketch verwenden.

FURTHER NOTES

Beim Deklarieren einer Variable muss ihr nicht unbedingt ein Wert zugewiesen werden. Schülerinnen und Schüler könnten zum Beispiel die Variable **readValue** so deklarieren und ihr später in dem Sketch einen Wert zuweisen.

8) Erstellt vor der **void setup()**-Funktion eine zweite Integer-Variable namens **writeValue** und setzt sie gleich Null.

Die Variable **writeValue** speichert den Wert, der die Helligkeit der LEDs steuert.

9) Ihr seid bereit, in die Hauptschleife des Programms zu gehen. Das erste, was du in der Funktion **void loop()** tun musst, ist den Wert vom Potentiometer zu bestimmen. Das Potentiometer ist mit dem 5V Power Pin verbunden, so dass es einen Spannungsbereich an Pin A0 zwischen null und fünf Volt erlaubt. Diese Spannung wird dann in einen analogen Wert zwischen 0 und 1.023 umgewandelt. Nachdem die Spannung umgewandelt wurde, können die analogen Werte innerhalb des Programms verwendet werden.

ERFAHREN SIE MEHR: **Bits und Bytes**

Um den Analogwert vom Potentiometer zu bestimmen, könnt ihr den Befehl **analogRead(pin)** verwenden. Dieser Befehl setzt den Wert zwischen 0 und 1.023, abhängig von der Spannung, die von der Schaltung an einem analogen Pin anliegt. Keine Spannung hat einen Analogwert von 0. Volle fünf Volt, die an den Pin angelegt werden, haben einen Wert von 1.023.

Gebt diesen Befehl in die Funktion **void loop()** ein:

Dieser Befehl speichert den Analogwert vom Potentiometer in die Variable **readValue**, die ihr oben in dem Sketch deklariert habt.

Wenn ihr es noch nicht getan habt, dann fügt einen Code-Kommentar in diese Zeile ein, der erklärt, was die Zeile macht. **VERSTÄNDNIS-ÜBERPRÜFUNG**

Was ist der Unterschied zwischen einem analogen Signal und einem digitalen Signal?

Was ist der Unterschied zwischen einem **digitalRead()**-Befehl und einem **analogRead()**-Befehl ist?

FURTHER NOTES

VERSTÄNDNIS-ÜBERPRÜFUNG - ANTWORTEN

Was ist der Unterschied zwischen einem analogen Signal und einem digitalen Signal?

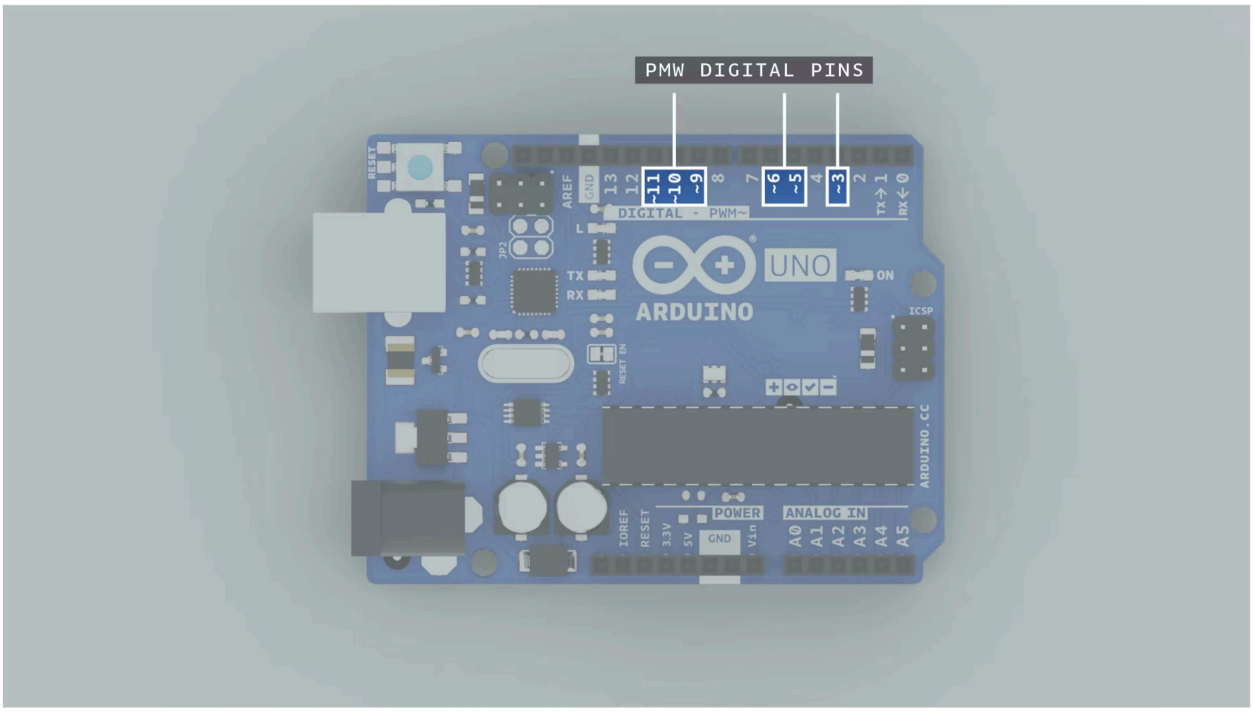
Antwort: Ein analoges Signal hat einen Wertebereich zwischen 0 und 1.023. Ein digitales Signal ist entweder hoch (high) oder niedrig (low), manchmal auch als an oder aus bezeichnet.

Was ist der Unterschied zwischen einem **digitalRead()**-Befehl und einem **analogRead()**-Befehl ist?

Antwort: Ein digitalRead()-Befehl gibt einen Wert zurück, der entweder HIGH oder LOW ist, während ein analogRead()-Befehl einen Wert zwischen 0 und 1.023 zurückgibt.

10) Überprüft euren Code und macht ein Debugging wenn nötig.

11) Die digitalen Pins auf dem Arduino UNO R3 Board können nicht wirklich einen analogen Wert an die Schaltung senden. Denkt daran, dass digitale Pins nur ein- oder ausgeschaltet sein können. Es gibt jedoch eine Möglichkeit, analoge Werte auf digitalen Pins zu simulieren, indem man einen Prozess namens Pulsweitenmodulation (PWM) verwendet. Digitale Pins, die PWM-Funktionalität enthalten, sind mit einem Schnörkelsymbol (~) gekennzeichnet. Aus diesem Grund benutzt ihr die Pins 9, 10 und 11 auf dem Arduino UNO R3 Board.



ERFAHREN SIE MEHR: **Pulsweitenmodulation**

Die digitalen Pins, die PWM zur Simulation von analogen Outputsignalen verwenden, haben eine Auflösung von 8 Bit. Diese unterscheidet sich von dem 10-Bit Analog-Digital-Wandler, der von den analogen Pins verwendet wird. Daher haben PWM-Pins einen Bereich von 256, da 2^8 = 256. Erinnert euch daran, dass acht Bit einem Byte entsprechen. Das bedeutet, dass PWM-Pins nur einen Wert zwischen 0 und 255 an die LEDs senden können.

Hinweis: Weitere Informationen über PWM findet ihr unter **https://www.arduino.cc/en/Tutorial/PWM**

Dies führt zu einem kleinen Problem. Der Inputbereich vom Potentiometer ist 0 bis 1.023, aber der Outputbereich zu den LEDs ist 0 bis 255. Wie könnt ihr den Unterschied in diesen beiden Bereichen auflösen? Die Antwort - ihr könnt einen Befehl schreiben, um den **readValue** (vom Potentiometer) durch 4 zu teilen und diesen Wert als **writeValue** zu speichern. Gebt diesen Befehl als nächste Zeile der Funktion **void loop()** ein:

Wenn anhand dieser Gleichung die Variable **readValue** 1.023 ist, ist die Variable **writeValue** 255.

Hinweis: Auch wenn 1.023 ÷ 4 = 255,75, wird nur der ganzzahlige Teil der Zahl in der Variable gespeichert, da die Variable als Variable vom Typ Integer (int) deklariert wurde.

FURTHER NOTES

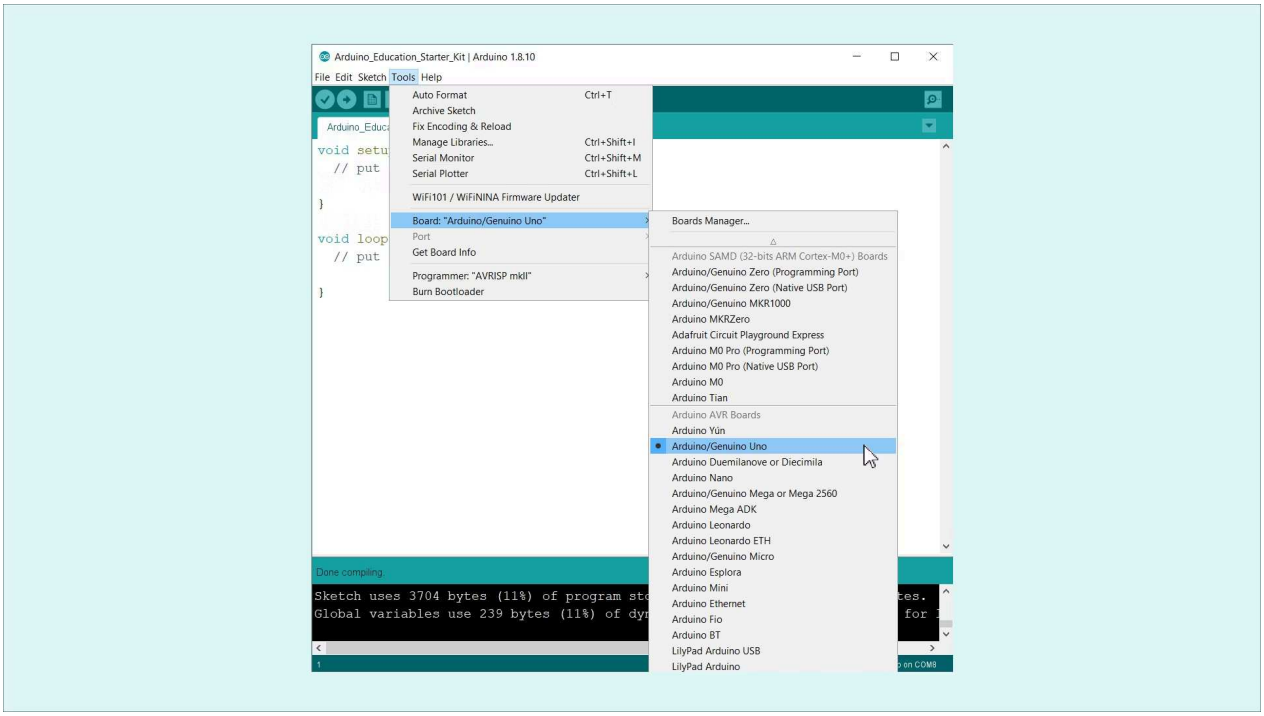
Leerzeichen haben in der IDE von Arduino normalerweise keine signifikante Bedeutung. Wie ihr sie verwenden wollt, ist eure persönliche Präferenz. Zum Beispiel wird jeder dieser Befehle das Arduino UNO R3 Board kompilieren und veranlassen, die gleiche Aufgabe auszuführen:

12) Um die Helligkeit der LEDs einzustellen, benutzt den Befehl **analogWrite(pin, value)**. In diesem Befehl ist pin die digitale Pin-Nummer, die PWM unterstützt und der *value* liegt zwischen 0 und 255. Ihr braucht drei **analogWrite()**-Befehle, einen für jede LED. Der Wert, der an die LED gesendet wird, wird in der Variable **writeValue** gespeichert. Fügt einen **analogWrite()**-Befehl für jede LED in die **void loop()**-Funktion ein. Fügt Code-Kommentare für jede Zeile hinzu. Wenn ihr fertig seid, sollte euer Sketch so ähnlich wie dieser aussehen:

13) Überprüft euren Code und macht ein Debugging wenn nötig.

14) Verbindet das Arduino UNO R3 Board über das USB-Kabel mit dem Computer.

15) Geht in der Arduino IDE zum **Tools**-Menü und bewegt dann die Maus zur Menüoption **Port**. Wählt den Port aus, der zu eurem Arduino Board gehört.



16) Klickt auf die Schaltfläche **Hochladen**, um den Sketch auf euer Board hochzuladen.

17) Nachdem der Sketch hochgeladen wurde, dreht ihr das Potentiometer, um die Helligkeit der LEDs einzustellen. Wenn sich eure LED-Helligkeit nicht ändert, debugged euren Code und wiederholt Schritt 16.

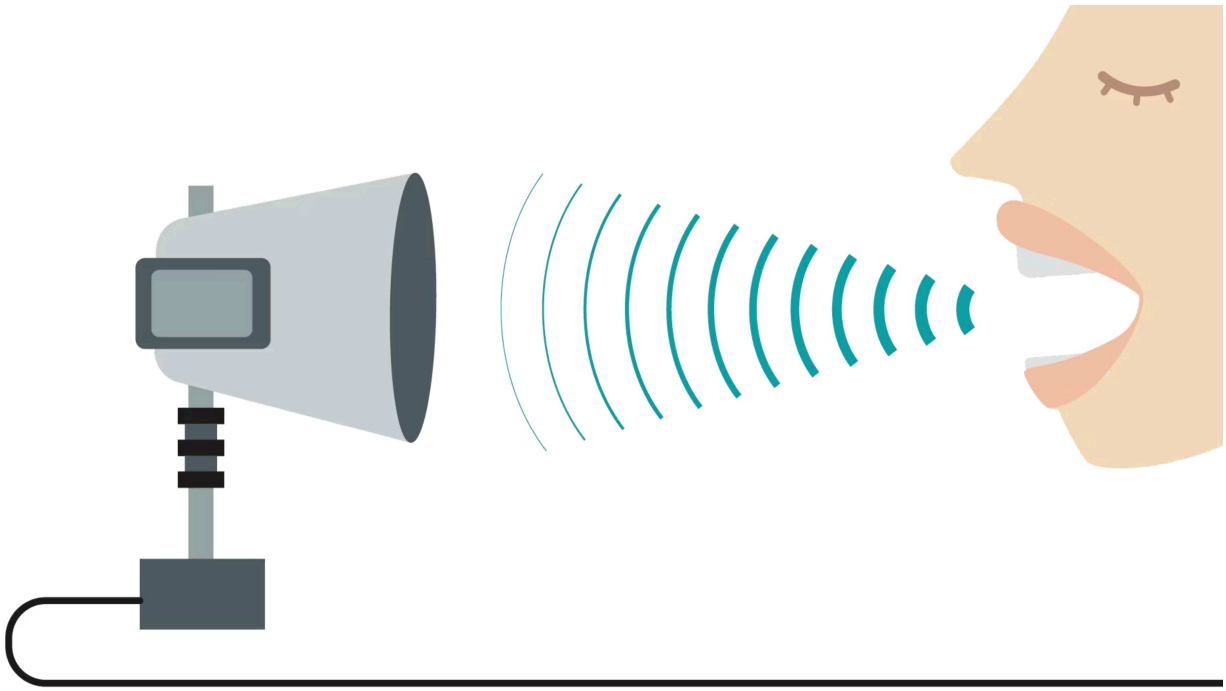
> **Hinweis:** Zum Debuggen ist es oft erforderlich, sowohl den Code als auch die Schaltung auf Fehler zu überprüfen. Wenn sich der Code korrekt kompiliert und richtig zu sein scheint, aber die LEDs sich nicht wie erwartet ändern, liegt vielleicht ein Problem in der Schaltung vor. Prüft, ob alle Komponenten und Steckverbinder gut mit dem Steckbrett verbunden sind.

18) Klickt auf **Speichern**, um euren Sketch zu speichern.

Blickpunkt Erfindungen

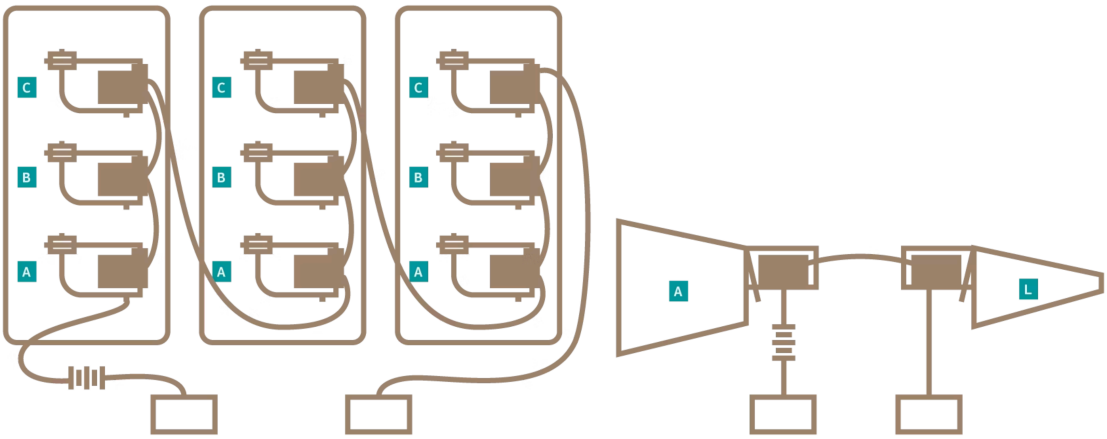
Das Telefon

Heute sind Kommunikationsformen, die der Welt vor wenigen Jahrzehnten noch unbekannt waren, für Menschen selbstverständlich. Die Internet-Revolution, die wir heute erleben, erinnert an vergangene Durchbrüche in der Kommunikationstechnologie, von der Erfindung des Schreibens bis zur Erschaffung der Druckerpresse. Ganz zu schweigen von der Erfindung des Telefons durch Alexander Graham Bell im Jahr 1876.



Schon zu Bells Zeit war es möglich, mit elektrischen Telegrafenmaschinen schriftliche Nachrichten über große Entfernungen über Drähte zu versenden. Diese Maschinen konnten jedoch keine Töne senden. Bell glaubte daran, dass es möglich wäre, Maschinen zu erschaffen, die dies konnten.

Zuerst benutzte er eine Maschine, die einen Ton in ein elektrisches Signal umwandelte. Dieses Signal wurde dann über einen Draht zu einer anderen Maschine übertragen, die einen Satz Metallzungen enthielt. Die Zungen würden vibrieren und den Ton reproduzieren. Der Ton, den er übertragen wollte, war natürlich die menschliche Stimme.



Bell experimentierte mit dieser Idee. Mit der Zeit entstand daraus das erste moderne Telefon. Das Debüt des Telefons ließ viele in der Öffentlichkeit unbeeindruckt. Sie fragten sich, warum sie ihre Stimme senden sollten, wenn sie eine getippte Nachricht schicken konnten. Obwohl das Telefon sowohl das Leben zu Hause als auch im Büro revolutionierte, könnten die modernen Amerikaner ihrer anfänglichen Skepsis wieder zustimmen; der Durchschnittsamerikaner sendet und empfängt heute 94 Sms pro Tag.

In einem großen Teil der Innovationsgeschichte rund um Telefonie und Computer geht es um Effizienz. In den 1930er und 40er Jahren machte sich der Mathematiker Claude Shannon viele Gedanken über die Vereinfachung von Schaltkreisen. Seine Vorstellungskraft war durch die Arbeit eines früheren Mathematikers gefesselt worden, der glaubte, dass sich Logik als mathematische Berechnungen darstellen lässt.

Shannon erkannte, dass elektrische Schalter so angeordnet werden können, dass sie diese Berechnungen, die als Bool'sche Algebra bezeichnet werden, durchführen können. Mit seinen Techniken wurde es möglich, Netzwerke von Telefonleitungen zu vereinfachen. Dies half auch bei der Entwicklung von Computerschaltungen.

Im Laufe der Jahrzehnte wuchs und wuchs die interne Komplexität der Computersysteme. Und mit der Schaffung von Computernetzwerken - d.h. Computern, die miteinander verbunden sind und Informationen austauschen - wuchs auch die externe Komplexität der Computersysteme. In den 1980er Jahren arbeitete ein englischer Wissenschaftler namens Tim Berners-Lee am CERN, einem Physiklabor

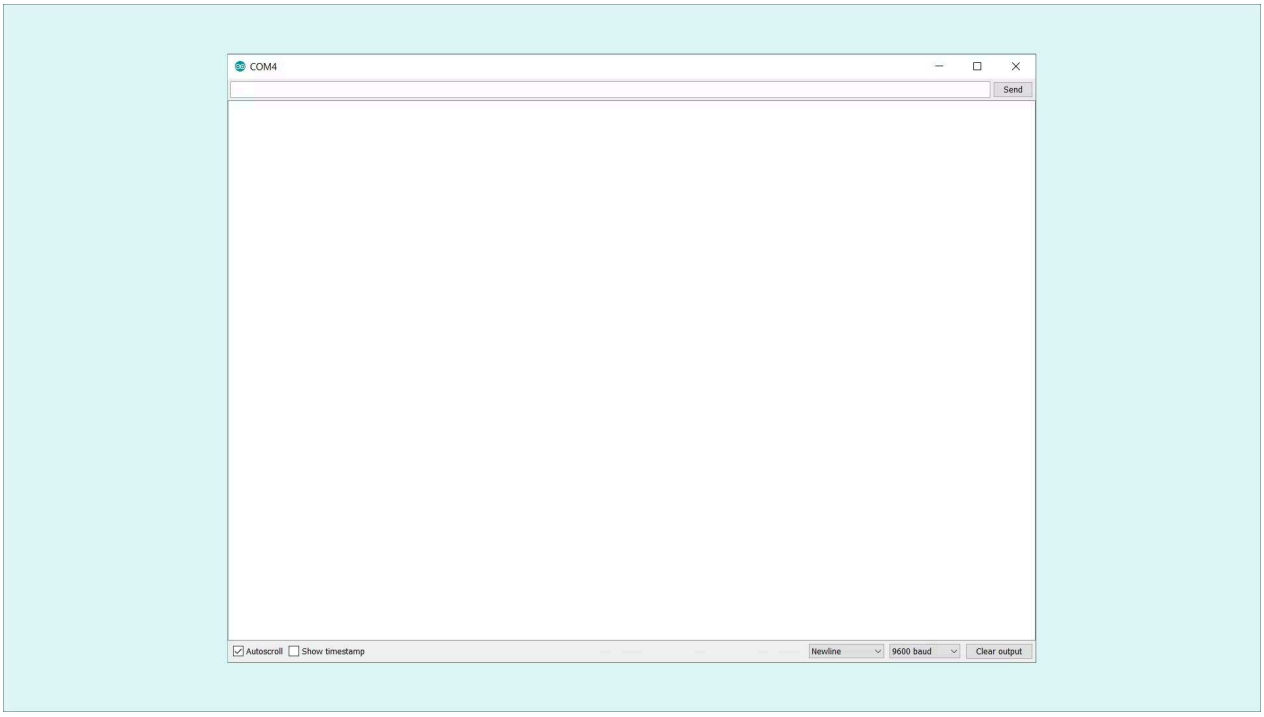
in der Schweiz. CERN hatte ein kleines Problem. Die Wissenschaftler hatten viele Daten durch ihre Forschungen angehäuft, aber sie waren alle in separaten Computersystemen gespeichert. Die Wissenschaftler, die mit einem System arbeiteten, hatten keine einfache Möglichkeit, Informationen mit Wissenschaftlern auszutauschen, die mit einem anderen System arbeiteten.

Mit Hilfe der Hypertext Markup Language (HTML) erstellte Berners-Lee Informationsseiten, die mit anderen Seiten in verzweigten Netzwerken - oder Webs - verlinkt werden konnten. Auf Seiten, die an einem Ort gespeichert waren, konnte auf separaten Computerterminals zugegriffen werden. Die Technologie zur Verbindung von Computern existierte bereits. Dieses Web machte es einfach, die Verbindungen zu nutzen.

Innerhalb weniger Jahre wuchs das Web aus einem Netzwerk, das nur wenige Wissenschaftler nutzten, zu dem weltweiten Phänomen, das es heute ist.

Code-Erstellung - Serielle Kommunikation

Der **serielle Monitor** ist ein nützliches Werkzeug, das in die Arduino IDE eingebaut ist und es dem Arduino Board ermöglicht, direkt mit dem Computer zu kommunizieren. Mit dem seriellen Monitor kann das Board Daten, Text und andere Informationen über den Sketch anzeigen. Er ist besonders praktisch bei der Arbeit mit analogen Signalen. Bei digitalen Signalen könnt ihr sehen, was der Eingabewert ist, indem ihr überprüft, ob eine LED ein- oder ausgeschaltet ist. Ein analoger Eingang kann jedoch 1.024 verschiedene Pegel haben. Man kann nicht genau sagen, welchen Pegel ein analoges Signal hat, indem man einfach nur die Helligkeit einer LED betrachtet. Aber über den seriellen Monitor könnt ihr Informationen sehen, die normalerweise nicht sichtbar sind.



In dieser Übung lernt ihr, Daten und Informationen auf dem seriellen Monitor auszugeben, damit ihr die Lese- und Schreibwerte sehen könnt, mit denen die Helligkeit der LEDs gesteuert werden.

TEACHER NOTES

In dieser Übung fahren die Schülerinnen und Schüler mit ihrer LED-Dimmschaltung und ihrem Sketch fort. Sie fügen Befehle hinzu, die es dem Arduino UNO R3 Board ermöglichen, mit dem Computer zu kommunizieren

und Echtzeitdaten anzuzeigen, wenn sich der Potentiometereingang ändert. Die Schülerinnen und Schüler werden in folgende Kodierungskonzepte eingeführt:

- **Serial.begin() –** Startet den seriellen Monitor
- **Serial.print() und Serial.println() –** Gibt Daten von der Arduino UNO R3-Karte auf den seriellen Monitor des Computers aus

1) Wenn nötig, öffnet euren Sketch aus Lektion4_V1 in der Arduino IDE.

2) Speichert den Sketch unter einem neuen Namen. Wählt im Dateimenü **Speichern unter...** Benennt die Datei "Lektion4*V2*", gefolgt von euren Initialen. Klickt auf **Speichern**.

3) Ihr könnt euer Lesson4_V2 Sketch-Fenster und dieses Fenster so anordnen, wie es euch am Besten gefällt.

4) Um den seriellen Monitor zu benutzen, müsst ihr ein Statement hinzufügen, das dem Arduino UNO R3 Board sagt, dass es den seriellen Monitor starten soll. Diese Anweisung wird normalerweise in dem **void setup()**-Abschnitt des Sketches geschrieben.

Der Befehl zum Starten des seriellen Monitors in einem Sketch lautet **Serial.begin(baud)**, wobei **Baud** die Informationsmenge ist, die zwischen dem Computer und dem Arduino UNO R3 Board pro Sekunde übertragen werden kann. Übliche **Baudraten** sind 9.600 Bit pro Sekunde und 115.200 Bit pro Sekunde.
Gebt in eurer Funktion **void setup()** den Befehl ein:

Hinweis: Beachtet die Groß-/Kleinschreibung in diesem Befehl. Vergesst das Semikolon am Ende nicht.

5) Der Befehl **Serial.print()** wird verwendet, um in Variablen gespeicherte Daten und anderen Text auf dem seriellen Monitor anzuzeigen. Um anzuzeigen, was in einer Variable gespeichert ist, gebt ihr den Namen der Variable in Klammern ein. Um

anderen Text anzuzeigen, setzt ihr den anzuzeigenden Text zwischen Anführungszeichen innerhalb der Klammern.

Um zu beginnen, müsst ihr den Wert anzeigen, der vom Potentiometer auf dem Arduino UNO R3 Board empfangen wird. Denkt daran, dass dieser Wert ein Analogwert ist, der in der Variablen **readValue** gespeichert wird. Gebt am Ende eures Sketches vor der schließenden Klammer der Funktion **void loop()** den Befehl ein:

6) Der serielle Monitor ermöglicht es euch, mehrere Informationen auf der gleichen Zeile zu drucken. Beispielsweise könnt ihr sowohl die Variable **readValue** als auch die Variable **writeValue** auf derselben Zeile drucken. Bevor ihr jedoch den Befehl zum Drucken der in der Variable **writeValue** gespeicherten Daten hinzufügt, ist es ratsam, ein Trennzeichen einzufügen, das die Variablen **readValue** und **writeValue** voneinander trennt. Gebt unter dem ersten **Serial.print()**-Befehl diesen Befehl ein:

Dadurch wird ein Doppelpunkt mit einem Leerzeichen auf beiden Seiten hinzugefügt, nachdem die Variable **readValue** gedruckt wurde. Immer wenn ihr Text, der nicht in einer Variable gespeichert ist, auf den seriellen Monitor drucken möchtet, muss der Text von Anführungszeichen umgeben sein.

7) Ihr müsst auch den Wert ausdrucken, der in der Variable **writeValue** gespeichert ist. Dies ist der Wert, der an die LEDs gesendet wird, um ihre Helligkeit zu steuern. Da dies die letzte Information ist, die im seriellen Monitor ausgedruckt wird, werdet ihr einen anderen Befehl verwenden, um diese Variable anzuzeigen - **Serial.println()**. Vor der schließenden Klammer für die **void loop()**-Funktion gebt ihr den Befehl ein:

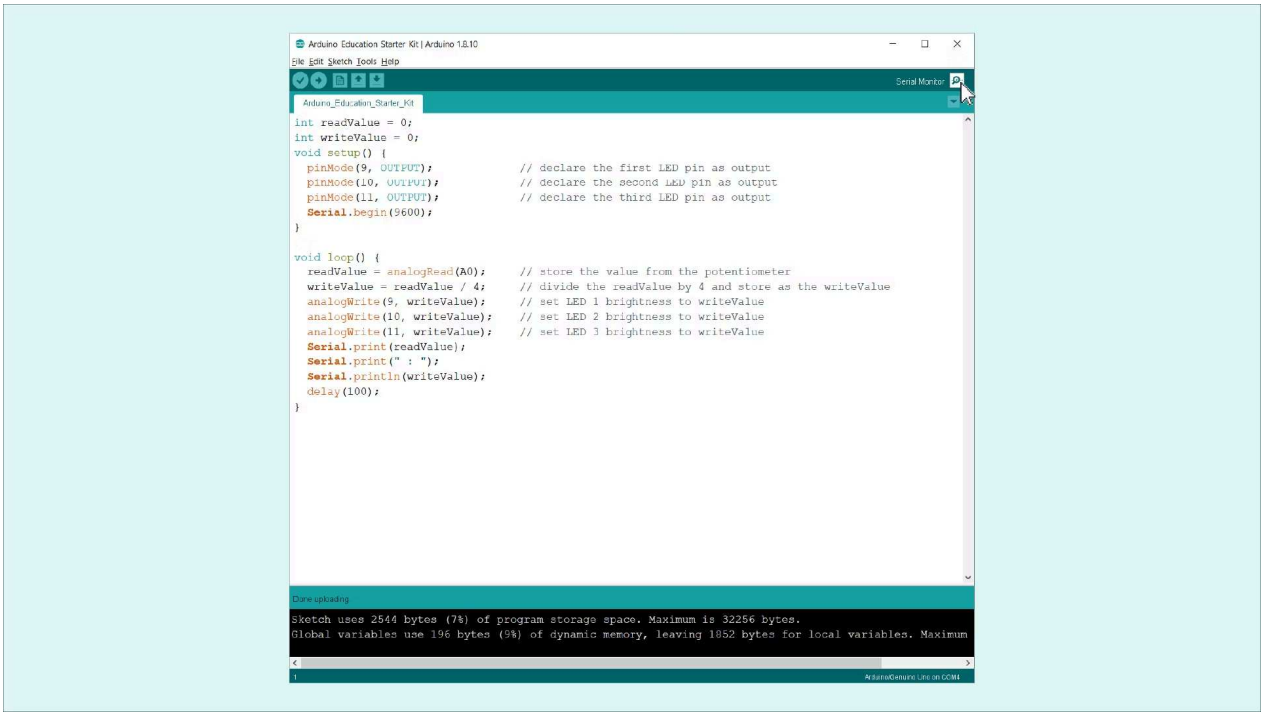
Dieser Befehl ähnelt dem Befehl **Serial.print()** mit dem Unterschied, dass er am Ende einen Zeilenumbruch einfügt. Das bedeutet, dass das nächste, was im seriellen Monitor gedruckt wird, in einer neuen Zeile ausgegeben wird.

8) Ihr solltet ein kurzes **delay()** vor der schließenden Klammer hinzufügen. Diese Verzögerung hat mehrere Auswirkungen. Es verlangsamt die Hauptschleife, was die Anzeige auf dem seriellen Monitor besser lesbar macht. Außerdem, wenn ihr viele Informationen an den seriellen Monitor sendet, könntet ihr Daten schneller senden, als der serielle Monitor diese ausdrucken kann. Dies führt dazu, dass sich der Port mit Daten füllt. Eine kurze Verzögerung hilft, um den Informationsfluss zu steuern. Eine Verzögerung von 100 Millisekunden sollte ein guter Ausgangspunkt sein. Ihr könnt diese Verzögerung später bei Bedarf anpassen.

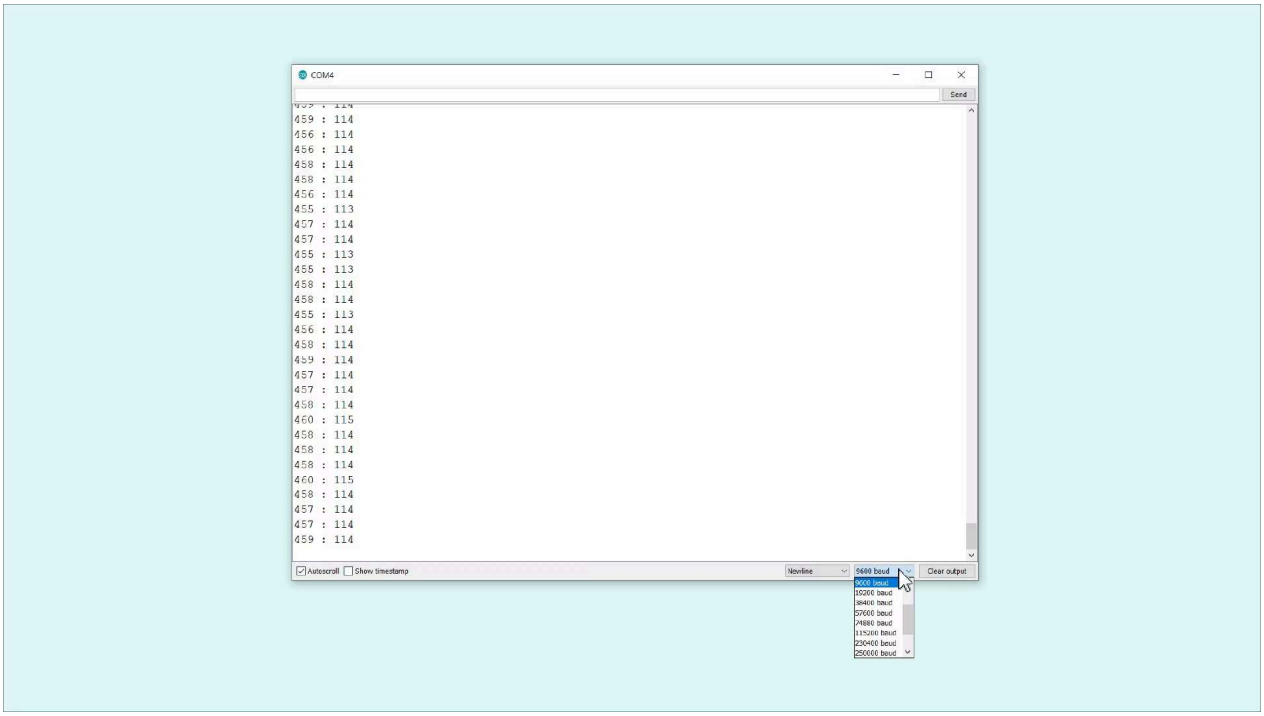
9) Überprüft euren Code und debugged ihn, wenn nötig. Euer Code sollte so ähnlich wie dieser aussehen:

10) Klickt auf den **Upload**-Button, um den Sketch auf euer Arduino UNO R3 Board hochzuladen.

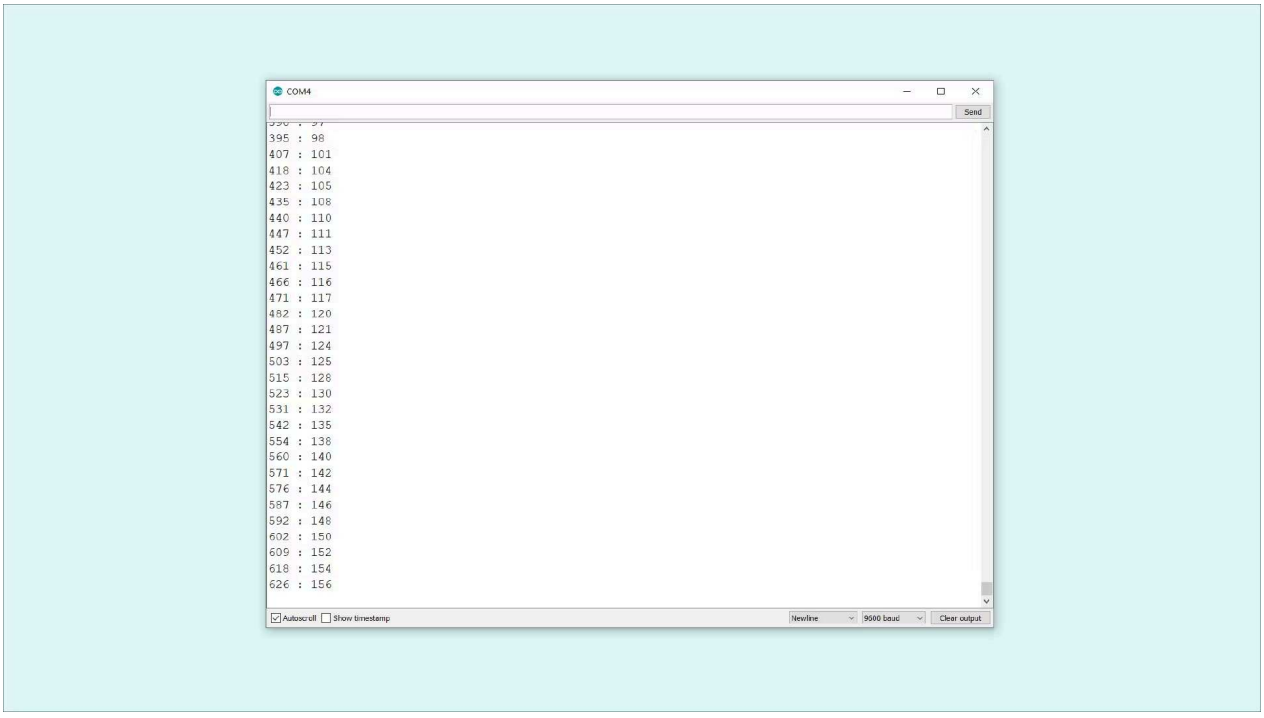
11) Klickt auf den **seriellen Monitor** Button in der oberen rechten Ecke der Arduino IDE. Der serielle Monitor wird in einem neuen Fenster geöffnet.



12) Vergewissert euch unten rechts, dass die Baudrate auf 9600 eingestellt ist. Benutzt das Drop-Down-Menü, um die Baudrate zu ändern, falls nötig. Klickt auf **Output löschen**, um das Fenster des seriellen Monitors zu leeren.



13) Dreht das Potentiometer, um die Helligkeit der LEDs einzustellen. Beachtet, wie sich die im seriellen Monitor angezeigten Werte ändern, wenn ihr am Potentiometer dreht. Denkt daran, dass der erste Wert **readValue** ist und dass dies der analoge Wert vom Potentiometer ist. Der zweite Wert ist **writeValue**, der an die LEDs gesendet wird, um ihre Helligkeit zu regeln.



Schaut euch auch euer Arduino UNO R3 Board an. Ihr solltet die TX LED blinken sehen. Diese LED blinkt jedes Mal, wenn Daten vom Board an den Computer gesendet werden.

14) Klickt auf **Speichern**, um euren Sketch zu speichern.

Code-Erstellung - Blinkender Indikator

Wenn ihr Code schreibt, ist es oft mit dem Ziel, dass bestimmte Dinge in bestimmten Situationen und andere Dinge in anderen Situationen passieren. In dieser Übung werdet ihr lernen, eine bedingte Anweisung zu benutzen, um die LEDs blinken zu lassen, wenn sie ihre maximale Helligkeit erreicht haben.

TEACHER NOTES

In dieser Übung modifizieren die Schülerinnen und Schüler ihren Sketch zum Dimmen der LEDs so, dass die LEDs beim erreichen der vollen Helligkeit blinken. Die Schülerinnen und Schüler erforschen bedingte Anweisungen (Wenn-Anweisungen) und wie bestimmte Befehle in bestimmten Situationen ausgeführt werden können. Die Schülerinnen und Schüler werden in diese Kodierungskonzepte eingeführt:

- **if(Bedingung){} else {} –** wenn-sonst bedingte Struktur

Die Schülerinnen und Schüler werden diese Konzepte wiederholen:

- **if (condition){} –** Grundform einer bedingten Anweisung
- **digitalWrite() –** sendet ein digitales Signal an die Schaltung
- **delay() –** hält den Sketch/das Programm für eine bestimmte Zeit an

1) Wenn nötig, öffnet euren Sketch aus Lektion4_V2 in der Arduino IDE.

2) Speichert den Sketch unter einem neuen Namen. In dem Dateimenü wählt ihr **Speichern unter...** Benennet die Datei "Lektion4*V3*", gefolgt von euren Initialen. Klickt auf **Speichern**.

3) Arrangiert euer Lektion4_V3 Sketch-Fenster und dieses Fenster so, wie es für euch am besten funktioniert.

4) In Lektion 3 habt ihr etwas über bedingte Aussagen gelernt, manchmal auch If-Statements genannt. Diese Aussagen lassen bestimmte Zeilen mit Code laufen, wenn eine Bedingung wahr ist. Erinnert euch daran, dass die Grundform einer bedingten Aussage so aussieht:

```
if (Bedingung) {
```

```
Befehle zur Ausführung, wenn die Bedingung wahr ist }
```

Erinnert euch, dass die Variable **readValue** ein analoger Wert zwischen 0 und 1.023 ist, der durch das Potentiometer bestimmt wird. Die LEDs sind am oberen Ende des Wertebereiches am hellsten. Erstellt eine Bedingung für die LEDs, wenn die Variable **readValue** größer oder gleich 1.000 ist. Tippt die folgende if-Anweisung nach den drei **analogWrite()**-Befehlen in eurer **void loop()**-Funktion und vor den **Serial.print()**-Befehlen ein:

Hinweis: Achtet darauf, dass der Bedingungsteil des if-Statements innerhalb der Klammern steht. Achtet auch darauf, dass das if-Statement sowohl eine öffnende als auch eine schließende Klammer hat. Die schließende Klammer kann in einer anderen Zeile stehen.

5) Ihr solltet dem if-Statement einen Code-Kommentar hinzufügen, um den Zustand zu beschreiben, der getestet wird. Hier ist ein Beispiel:

FURTHER NOTES

Die Schülerinnen und Schüler könnten sich fragen, warum sie 1.000 als Schwellenwert zur Kontrolle des Blinkens und nicht den Maximalwert von 1.023 verwendet haben. Auch wenn das Potentiometer auf 0 bis 10.000 Ohm ausgelegt ist, wird der tatsächliche Widerstandsbereich aufgrund der Toleranz des Potentiometers leicht abweichen. Zum Beispiel könnte er eher bei 10 bis 9.800 Ohm liegen. Genau wie bei Widerständen ist jedes Potentiometer geringfügig anders. Wenn der Schwellenwert für das Blinken auf 1.000 oder höher eingestellt wird, erzeugt es einen Bereich von Analogwerten, in dem das Blinken auftreten wird.

Wenn die Schülerinnen und Schüler ihre LEDs nicht zum Blinken bringen können, lassen Sie sie den seriellen Monitor überprüfen, um zu sehen, wie hoch ihr maximaler Analogwert für ihr Potentiometer ist. Möglicherweise

müssen sie den Schwellenwert in ihrer bedingten Aussage von 1.000 auf 900 herabsetzen.

6) Überprüft und debugged euren Code, wenn nötig.

7) Innerhalb der geschweiften Klammern des if-Statements befindet sich die Stelle, an der ihr die Befehle eingebt, die ausgeführt werden sollen, wenn die Bedingung wahr ist. In eurem Fall möchtet ihr bewirken, dass die LEDs blinken, wenn der Potentiometerwert über 1.000 liegt.

Erinnert euch daran, dass ihr in Lektion 3 den **digitalWrite()**-Befehl benutzt habt, um die LEDs an- und auszuschalten, die mit den Outputpins des Arduino UNO R3 Boards verbunden sind. Hier könnt ihr den gleichen Befehl verwenden, um die LEDs auszuschalten. Fügt in die Klammern eures if-Statements einen **digitalWrite()**-Befehl ein, um die LEDs, die an die Pins 9, 10 und 11 angeschlossen sind, auszuschalten.

Hinweis: Beachtet, wie die Befehle in den geschweiften Klammern des if-Statements eingerückt sind. Die Einrückung hilft euch, die Befehle zu identifizieren, die ausgeführt werden, wenn die Bedingung wahr ist.

Fügt Code-Kommentare zu euren **digitalWrite()**-Befehlen hinzu.

Hinweis: Neben der Verwendung von **digitalWrite(pin, LOW)** zum Ausschalten der LEDs könnt ihr auch den Befehl **analogWrite(pin, 0)** verwenden. Beide Befehle haben genau die gleiche Wirkung auf die LEDs.

8) Damit die LEDs blinken, müssen sie für kurze Zeit ausgeschaltet bleiben, bevor sie wieder eingeschaltet werden. Fügt einen **delay()**-Befehl nach den drei **digitalWrite()**-Befehlen innerhalb der Klammern für das if-Statement hinzu. Stellt die Verzögerung auf 250 Millisekunden (eine Viertelsekunde) ein.

Fügt einen Code-Kommentar für das delay() hinzu.

9) Jetzt, da die LEDs aus sind, müsst ihr Befehle hinzufügen, damit die LEDs wieder leuchten. Benutzt drei weitere **digitalWrite()**-Befehle, um die LEDs wieder anzuschalten.

> **Hinweis:** Kopiert die drei **digitalWrite()**-Befehle, die die LEDs ausschalten, und fügt die kopierten Befehle nach der Verzögerung von 250 Millisekunden ein. Ändert dann für jeden der neuen Befehle den Status von LOW zu HIGH.

Fügt Code-Kommentare für die drei neuen **digitalWrite()**-Befehle hinzu.

10) Um das Blinken zu beenden, müssen die Lichter für eine kurze Zeit an bleiben. Ihr solltet eine weitere Verzögerung von 250 Millisekunden vor der schließenden Klammer des if-Statements hinzufügen. Fügt einen Code-Kommentar für diese Verzögerung hinzu. Dein vollständiges if-Statement sollte ähnlich wie dieses aussehen:

FURTHER NOTES

Es ist möglich, **digitalWrite()** und **analogWrite()** Befehle auf denselben Pins zu verwenden. Was gewählt wird, hängt nur davon ab, wie Sie die Spannung steuern wollen, die durch jeden Pin auf das Arduino UNO R3 Board gesendet wird. Erinnern Sie die Schülerinnen und Schüler daran, dass **analogWrite()** nur auf digitalen Pins funktioniert, die das Symbol für Pulsbreitenmodulation (~) haben.

11) Überprüft und debugged euren Code, falls erforderlich.

12) Neben der grundlegenden bedingten Anweisung gibt es komplexe Bedingungen, mit denen mehr als eine Bedingung ausgewertet werden können. Eine "if-else"-Bedingung führt einen Befehlssatz aus, wenn die Bedingung wahr ist, und führt den

"else"-Befehlssatz aus, wenn die Bedingung falsch ist. Die Struktur einer if-else-Bedingung sieht wie folgt aus:

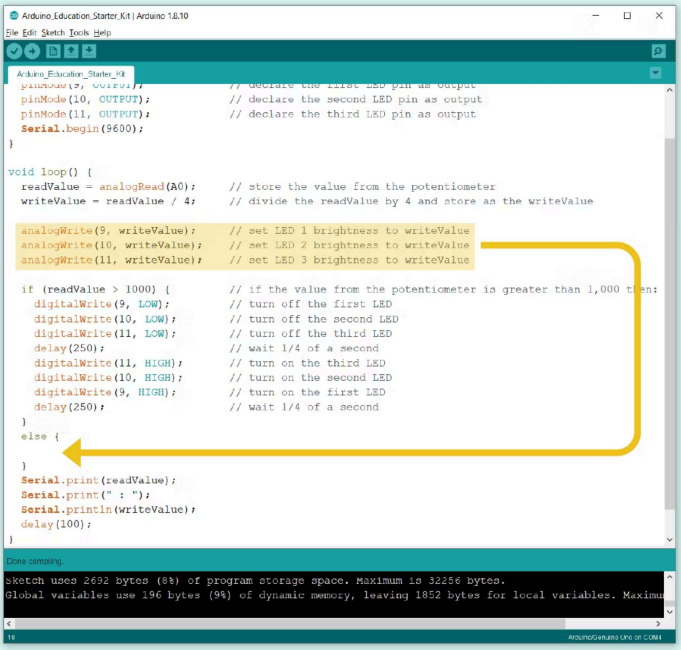
```
if (Bedingung) {
Befehle zur Ausführung, wenn die Bedingung wahr ist
}
else {
Befehle zur Ausführung, wenn die Bedingung falsch ist
}
```

In eurem Sketch sollen die LEDs blinken, wenn die Variable **readValue** größer als 1.000 ist. Ansonsten (else) soll die LED-Helligkeit anhand der Variable **writeValue** eingestellt werden.

Fügt eurem Code nach dem if-Statement einen nachfolgendes **else** hinzu. Achtet darauf, sowohl eine öffnende als auch eine schließende Klammer einzufügen.

13) Ganz oben in eurer **void loop()**-Funktion sind eure drei **analogWrite()**-Anweisungen, die die Helligkeit der LEDs einstellen. Diese Statements sollen nur ausgeführt werden, wenn die Variable **readValue** kleiner oder gleich 1.000 ist. Schneidet diese Statements aus und fügt sie zwischen der öffnenden und

schließenden Klammer des **else {}** Teils eures if-else-Statements ein.



TEACHER NOTES

KLASSENDISKUSSION

Fragen Sie die Schülerinnen und Schüler, ob der **else {}** Teil des Sketches wirklich benötigt wird.

Der Sketch funktioniert auch ohne den **else{}** Abschnitt; er ist jedoch weniger effizient. Wenn der Wert des Potentiometers größer als 1.000 ist, werden die LEDs vor dem if-Statement auf den analogen Wert gesetzt und dann innerhalb des if-Statements auf HIGH und LOW gesetzt. Durch Einbeziehen des **else {}** Teils wird eine Entweder-Oder-Situation geschaffen, in der nur ein Verhalten gezeigt wird (entweder verblassen die LEDs oder sie blinken).

14) Überprüft und debugged euren Code, falls nötig. Euer Sketch sollte so ähnlich wie dieser aussehen:

15) Klickt auf den **Upload**-Button, um den Sketch auf euer Arduino UNO R3 hochzuladen.

16) Klickt auf den **seriellen Monitor** Button in der oberen rechten Ecke der Arduino IDE. Der serielle Monitor wird in einem neuen Fenster geöffnet.

17) Dreht das Potentiometer, um die Helligkeit der LEDs einzustellen. Blinken die LEDs, wenn das Potentiometer ganz nach oben gedreht wird? Wenn nicht, geht zurück zu eurem Sketch und debugged diesen. Wenn ihr fertig seid, ladet euren Sketch erneut auf euer Arduino UNO R3 hoch und versucht es noch einmal.

18) Klickt auf **Speichern**, um euren Sketch zu speichern.

Testen und ändern

LED-Arrays werden oft als Indikatoren verwendet, um den Wert von etwas anzuzeigen. Je mehr LEDs aufleuchten, desto höher ist der Wert. Zum Beispiel benutzen viele Mischpulte LED-Arrays, um die Lautstärke eines Instruments oder Mikrofons anzuzeigen (diese werden Volumeneinheitsmeter oder VU-Meter genannt). Je lauter das Audiosignal, desto mehr LEDs leuchten auf. In dieser Übung ändert ihr euren Sketch mit den blinkenden Indikatoren und programmiert die LEDs so, dass sie als Anzeigefeld fungieren. Für diese Übung benötigt ihr euer Schüler-Arbeitsheft.

TEACHER NOTES

In dieser Übung schreiben die Schülerinnen und Schüler Pseudocode zur Erstellung eines LED-Arrays, das basierend auf dem Wert des Potentiometers jeweils eine LED aufleuchten lässt. Dann ändern sie ihren Code aus der vorherigen Übung, um das Array zu programmieren. Für diese Übung benötigen die Schülerinnen und Schüler ihr Arbeitsheft. Während dieser Übung werden die Schülerinnen und Schüler mit diesen Befehlen vertraut gemacht:

- **if(Bedingung 1){} else if(Bedingung 2){}** – komplexe Bedingungen im if-else if-Format

1) Findet in eurem Arbeitsheft Seite 11 zur Lektion 4.

2) In Lektion 3 habt ihr gelernt, wie man Pseudocode schreibt. Erinnert euch daran, dass Pseudocode wie eine Skizze für ein Programm oder einen Sketch ist und in einer Alltagssprache geschrieben ist, die ihr leicht verstehen könnt. Gleich werdet ihr Pseudocode schreiben, um ein LED-Array zu erstellen, das den Wert des Potentiometers anzeigt. Die LEDs müssen nacheinander aufblenden und aufleuchten,

wenn das Potentiometer aufgedreht wird. Aber bevor ihr euren Pseudocode schreibt, schaut euch die folgenden Fragen an und diskutiert sie mit einem Partner.

- Wie erfährt das Arduino UNO R3 Board den Wert des Potentiometers?
- Wie kann man die Helligkeit der einzelnen LEDs steuern?
- Wie kann man alle LEDs dazu bringen, sich auszuschalten, wenn das Potentiometer ganz heruntergedreht ist?
- Wie kann man eine LED dazu bringen, sich einzuschalten, wenn der Potentiometerwert niedrig ist?
- Wie kann man zwei LEDs dazu bringen, sich einzuschalten, wenn der Potentiometerwert im mittleren Bereich liegt?
- Wie kann man alle drei LEDs dazu bringen, sich einzuschalten, wenn der Potentiometerwert hoch ist?
- Wann solltet ihr **analogWrite()** und wann **digitalWrite()** verwenden?

3) Nun, da ihr euch die obigen Fragen angeschaut habt, schreibt ihr Pseudocode für eure LED-Anzeigenreihe in den dafür vorgesehenen Platz in eurem Arbeitsheft.

TEACHER NOTES

Lösungshinweise für die Fragen in Lektion 4 des Schüler-Arbeitsheftes finden Sie auf Seite 11 im Lehrer-Arbeitsheft.

4) Um euren Pseudocode in einen Sketch für das Arduino UNO R3 Board umzuwandeln, fangt ihr mit dem Sketch aus eurer vorherigen Lektion 4 an. Öffnet euren Lektion4_V3 Sketch in der Arduino IDE.

5) Speichert den Sketch unter einem neuen Namen. Aus dem Dateimenü wählt ihr **Speichern unter...** Benennt die Datei mit "Lektion4*V4*", gefolgt von euren Initialen. Klickt auf **Speichern**.

6) Ihr könnt euer Lesson4_V4 Sketch-Fenster und dieses Fenster so anordnen, wie es euch am besten gefällt.

7) Um jede LED der Reihe nach, basierend auf dem Wert des Potentiometers, einzuschalten, braucht ihr mehrere Bedingungen. Ihr könntet zu diesem Zweck mehrere if-Anweisungen verwenden. Zum Beispiel:

- Wenn der Potentiometerwert weniger als 300 beträgt, schalten sich alle LEDs aus.
- Wenn der Potentiometerwert zwischen 300 und 600 liegt, schaltet sich die erste LED ein.
- Wenn der Potentiometerwert zwischen 600 und 900 liegt, schalten sich die ersten beiden LEDs ein.
- Wenn der Potentiometerwert größer als 900 ist, schalten sich alle LEDs ein.

Diese Verwendung mehrerer bedingter Anweisungen erfordert jedoch in einigen Fällen, dass das Arduino UNO R3 das Potentiometer mit zwei Werten vergleicht. Außerdem wird der Code dadurch ineffizient, da jede einzelne Bedingung jedes Mal durch den Loop getestet werden muss.

Um den Code zu vereinfachen und zu optimieren, können wir eine bedingte Struktur verwenden, die als if-else if-Statement bezeichnet wird. Die Struktur eines if-else if-Statements sieht wie folgt aus:

In einer if-else if-Struktur wird die erste Bedingung getestet. Wenn sie wahr ist, führt sie die mit ihr verbundenen Befehle aus und überspringt alle anderen Bedingungen (selbst wenn einige dieser Bedingungen wahr sind). Wenn die erste Bedingung falsch ist, wird die zweite Bedingung in der Struktur getestet. Wenn die zweite Bedingung wahr ist, führt sie die mit dieser Bedingung verknüpften Befehle aus und überspringt dann die anderen Bedingungen. Wenn sie falsch ist, wird die dritte Bedingung getestet und so weiter. In einigen Fällen ist es möglich, mehrere wahre Bedingungen zu haben. Daher ist die Reihenfolge der Bedingungen wichtig. Nur bei der ersten wahren Bedingung werden die zugehörigen Befehle ausgeführt.

Ändert euern Sketch, indem ihr euer if-Statement wie folgt ändern:

Diese Bedingung wird alle drei LEDs ausschalten, wenn der Wert des Potentiometers kleiner als 300 ist. Ihr könnt **digitalWrite()**-Befehle verwenden, um die LEDs komplett auszuschalten.

8) Die nächste zu testende Bedingung ist, wenn das Potentiometer im unteren bis mittleren Bereich ist. In diesem Bereich soll die erste LED aufleuchten. Verwendet einen Wert von 600 als Schwellenwert für diese Bedingung. Unterhalb eurer if-Anweisung fügt ihr eine else if Bedingung hinzu, die folgendes besagt:

Hier soll die erste LED abhängig vom Wert des Potentiometers ausblenden, also wird **analogWrite()** benutzt, um die erste LED einzuschalten.

9) Überprüft und debugged euren Code, falls erforderlich.

10) Der nächste Bereich ist der mittlere bis hohe Bereich. Ihr stellt den Potentiometer-Schwellenwert für diese Bedingung auf 900 ein. Fügt noch ein weiteres if-Statement hinzu, das so aussieht:

Auch hier wird **analogWrite()** verwendet, um die ersten beiden LEDs basierend auf dem Wert des Potentiometers auszublenden.

11) Ihr beendet die if-else if Bedingungsstruktur mit einem abschließenden else Abschnitt. Diese Sektion enthält die Befehle, die ausgeführt werden, wenn keine der anderen Bedingungen wahr ist. Mit anderen Worten, die **else {}** Befehle werden ausgeführt, wenn **readValue** vom Potentiometer größer oder gleich 900 ist. In diesem Fall ist der Wert hoch, also sollten alle drei LEDs an sein. Euer else-Abschnitt sollte so aussehen:

FURTHER NOTES

Sie sollen vielleicht darauf hinweisen, dass für Potentiometerwerte, die größer oder gleich 300 sind, mehrere Bedingungen der if-else if-Struktur wahr sind. Zum Beispiel, wenn der Wert des Potentiometers 200 ist, dann sind die ersten drei Bedingungen wahr. Allerdings werden nur die Befehle ausgeführt,

die mit der ersten wahren Bedingung verbunden sind. Die anderen wahren Bedingungen werden übersprungen, auch der abschliessende else-Teil der Struktur.

ERFAHREN SIE MEHR: **Bedingte Strukturen**

12) Überprüft und debugged euren Code, falls nötig. Wenn ihr Fehler in eurem Code habt, geht zurück und überprüft ihn anhand des Beispielcodes, der in den Schritten 7-11 gezeigt wird.

FURTHER NOTES

Der Code für diesen Sketch sollte ähnlich wie dieser aussehen:

13) Klickt auf den **Upload**-Button, um den Sketch auf euer Arduino UNO R3 hochzuladen.

14) Klickt auf den **seriellen Monitor** Button in der oberen rechten Ecke der Arduino IDE. Der serielle Monitor wird in einem neuen Fenster geöffnet.

15) Dreht das Potentiometer, um das LED-Array zu kontrollieren. Funktioniert das Array richtig? Wenn nicht, debugged euren Code und ladet euren Sketch erneut hoch.

Hinweis: Zum Debuggen muss oft sowohl der Code als auch die Schaltung auf Fehler überprüft werden. Wenn sich der Code korrekt kompiliert und korrekt zu sein scheint, aber die LEDs sich nicht wie erwartet ändern, liegt vielleicht ein Problem in der Schaltung vor. Überprüft, dass alle Komponenten und Steckverbinder gute Verbindungen mit dem Steckbrett haben.

16) Klickt auf **Speichern**, um euren Sketch zu speichern.

17) Schließt alle Fenster der Arduino IDE.

18) Räumt euren Arbeitsbereich auf und lagert alle Ausrüstungsgegenstände in einem geeigneten Lagerbereich.

FURTHER NOTES

WEITERE ÜBUNGEN

- **Spannungsteiler** – Lassen Sie die Schülerinnen und Schüler das Multimeter verwenden, um den Unterschied zwischen den beiden Anschlüssen zu untersuchen, die sich auf derselben Seite des Potentiometers befinden. Die Schülerinnen und Schüler sollten bemerken, dass die Anschlüsse entgegengesetzt sind. Wenn das Potentiometer ganz auf eine Seite gedreht wird, zeigt ein Anschluss Null Ohm an, während der andere 10 Kiloohm anzeigt. Wenn das Potentiometer ganz in die andere Richtung gedreht wird, tauschen die Werte von null und 10 Kilohm die Anschlüsse. Wenn sich das Potentiometer in der Mitte befindet, sollte der Widerstand an beiden Anschlüssen 10 Kiloohm betragen. Wegen dieser Eigenschaft werden Potentiometer manchmal als Spannungsteiler bezeichnet. Sie teilen die Gesamtspannung für das Potentiometer in zwei verschiedene Spannungen auf, die von verschiedenen Teilen der Schaltung verwendet werden können.

- **Serieller Plotter** – Der serielle Plotter ist ein weiteres nützliches Werkzeug zur Visualisierung von Daten, die vom Arduino UNO R3 Board an den Computer gesendet werden. Mit diesem Werkzeug werden Daten in einem Diagramm dargestellt. Jede Iteration durch die Funktion void loop() erzeugt einen weiteren Satz von Datenpunkten auf dem Diagramm. Lassen Sie die Schülerinnen und Schüler ihren Sketch ausführen. Die Schülerinnen und Schüler sollten das Fenster "Serial Monitor" schließen, da Sie den seriellen Monitor und den seriellen Plotter nicht gleichzeitig laufen lassen können. Wählen Sie in der IDE im Menü Tools die Option Serial Plotter. Lassen Sie die Schülerinnen und Schüler die grafische Kurve betrachten, während sie am Potentiometer drehen und die Helligkeit der LEDs ändern.