AMPEL

In dieser Lektion, werdet ihr ein Programme zur Steuerung der LEDs in der Schaltung schreiben, einschließlich eines Programms zur Steuerung einer Ampel. Wenn das Ampelprogramm fertig ist und läuft, werdet ihr ein Multimeter verwenden, um elektrische Messungen an der Schaltung vorzunehmen.

VORAUSSICHTLICHE ZEIT **90-135 min**

SIE WERDEN ETWAS DARÜBER LERNEN

PSEUDOCODE, ARDUINO IDE , ANALOGEN KOMPONENTEN, DIGITALE KOMPONENTEN, OHMSCHES GESETZ, WIDERSTANDSWERT ZU MESSEN, EIN EXPERIMENT DURCHFÜHREN, CODE DEBUGGEN, SPANNUNG MESSEN

Übersicht

Woran denkst du, wenn du an Kommunikation denkst? Wie kommunizieren Menschen miteinander? Gewiss, zur Kommunikation gehören Sprechen und Zuhören, aber es gibt auch andere Formen der Kommunikation. Zum Beispiel werden Lichter und LEDs oft als Anzeige an Maschinen verwendet, um den Zustand der Maschine zu kommunizieren. Fahrzeuge haben Leuchten, die über einen niedrigen Benzinstand, einen niedrigen Öldruck, eine schwache Batterie und über Motorprobleme informieren. Sogar Ampeln kommunizieren, wann es erlaubt ist, eine Kreuzung zu überqueren. All diese und viele andere Beispiele erfordern eine präzise Steuerung des Stromflusses in einem Stromkreislauf. Diese Steuerung wird häufig von einem Computerprogramm durchgeführt, das den Ampeln mitteilt, wann sie ein- oder ausgeschaltet werden sollen.

In dieser Lektion werdet ihr untersuchen, wie Schaltkreise von Mikrocontrollern wie dem auf deinem Arduino UNO R3 Board betrieben werden. Genauer gesagt werdet ihr den Unterschied zwischen digitalen und analogen Geräten untersuchen und wie diese Geräte als Eingänge zum Sammeln von Informationen oder als Ausgänge zur Ausführung einer Aufgabe verwendet werden können. Ihr werdet außerdem die Arduino Programmierumgebung und ihre Kommunikation mit dem Arduino UNO R3-Board kennenlernen. Nachdem ihr eine Schaltung mit LEDs konstruiert habt, werdet ihr ein Programme zur Steuerung der LEDs in der Schaltung schreiben, einschließlich eines

Programms zur Steuerung einer Ampel. Wenn das Ampelprogramm fertig ist und läuft, werdet ihr ein Multimeter verwenden, um elektrische Messungen an der Schaltung vorzunehmen.

TEACHER NOTES

In dieser Lektion werden die Schülerinnen und Schüler in die Arduino Software (IDE) eingeführt und verwenden diese, um Code zu schreiben, der auf das Arduino UNO R3 Board hochgeladen wird. Die Lektion beginnt mit einer grundlegenden Einführung zum Arduino UNO R3 Board, zur Arduino Software (IDE), und zur Verwendung von Pseudocode, um den Beginn eines neuen Programms zu entwickeln (in der Arduino IDE als Sketch bezeichnet). Die Schülerinnen und Schüler bauen eine Ampelschaltung, schreiben einen Sketch, der die Funktionsweise der Schaltung steuert, laden den Sketch auf das Board hoch und debuggen ihren Code, bis die Schaltung korrekt funktioniert. Wenn die Schaltung fertig ist, benutzen die Schülerinnen und Schüler ihr Multimeter, um zu bestimmen, wie das Board die Schaltung steuert, und verwenden das Ohmsche Gesetz, um die Widerstandswerte für die LEDs in der Schaltung zu berechnen.

ZEIT FÜR GESAMTE LEKTION

Die hier aufgeführten Zeiten sind nur Richtwerte und müssen möglicherweise der jeweiligen Lernsituation angepasst werden.

Übersicht und Fachbegriffe	3 minuten
Blickpunkt Erfindungen	5 minuten
Einfache Ampelschaltung	
Benötigte Materialien	2 minuten
Aufbau der Schaltung	13 minuten
Code Erstellung - LEDs zum Leuchten bringen	18 minuten
Code-Erstellung - Blinken der LEDs	10 minuten
Code-Erstellung - Programmieren einer Ampelanlage	10 minuten
Ampel mit Fußgängertaste	
Benötigte Materialien	1 minute
Aufbau der Schaltung	4 minuten
Code-Erstellung - Programmierung der Taste	15 minuten

Testen und ändern	8 minuten
Aufräumen	1 minute
Gesamte Lektion	**90 minuten**

Wenn Sie diese Lektion in zwei Unterrichtsstunden abschließen wollen, sollten die Schülerinnen und Schüler den Abschnitt "Code-Erstellung - Blinken der LEDs" zum Ende der ersten Stunde beendet haben.

LERNZIELE

- Unterschiede zwischen digitalen und analogen Komponenten erkennen
- Bestandteile des Arduino UNO R3 Boards untersuchen
- Eine Ampelschaltung konstruieren und damit experimentieren
- Einen Pseudocode schreiben, der beschreibt, wie eine Ampel funktionieren soll
- Den Pseudocode in Computercode übersetzen und ein Programm zur Steuerung einer Ampel schreiben
- Code debuggen, der Fehler enthält
- Spannung, Strom und Widerstand in einer Schaltung messen
- Das Ohm'sche Gesetz zur Berechnung des Widerstands in einer Schaltung verwenden

Fachbegriffe

- **Algorithmus -** eine Reihe von Schritten oder Regeln, die zur Lösung eines Problems befolgt werden sollten.
- **Vergleichsoperator -** ein mathematisches Symbol, das zum Vergleich zweier Werte verwendet wird; Vergleichsoperatoren umfassen gleich (==), nicht gleich (!=), größer als (>), kleiner als (<), größer oder gleich (>=) und kleiner oder gleich (<=).
- **Kompilieren -** der Prozess, den ein Computer durchführt, um eine Programmiersprache in Maschinencode zu übersetzen, den ein Computer oder ein anderes digitales Gerät verstehen kann.
- **Bedingte Anweisung -** eine Aussage in der Programmierung, die zwei Werte vergleicht, um zu sehen, ob eine bestimmte Bedingung wahr ist. Wenn die Bedingung wahr ist, führt sie einen bestimmten Befehl oder eine bestimmte Gruppe von Befehlen aus.
- **debuggen -** Fehler in einem Programm zu identifizieren und zu beheben.
- **Optimierung -** Prozess der Verbesserung eines Programms, um den Code effizienter zu machen.
- **Pseudocode -** eine informelle Beschreibung dessen, was ein Programm tun sollte, geschrieben in normaler Alltagssprache.
- **Pull-Down-Widerstand -** ein Widerstand, der den Standardwert eines digitalen Pins auf LOW (niedrig) setzt, indem er die Spannung auf Null oder nahe Null absenkt. Er wird zwischen den digitalen Pin und Ground geschaltet.
- **Schwellenwert -** ein Betrag, der erreicht werden muss, damit eine bestimmte Bedingung erfüllt wird oder eine Aktion stattfindet.
- **verifizieren -** Code auswerten, um festzustellen, ob er lesbar ist und von einem Computer kompiliert werden kann.

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.

Blickpunkt Erfindungen

Geschichte der Ampel

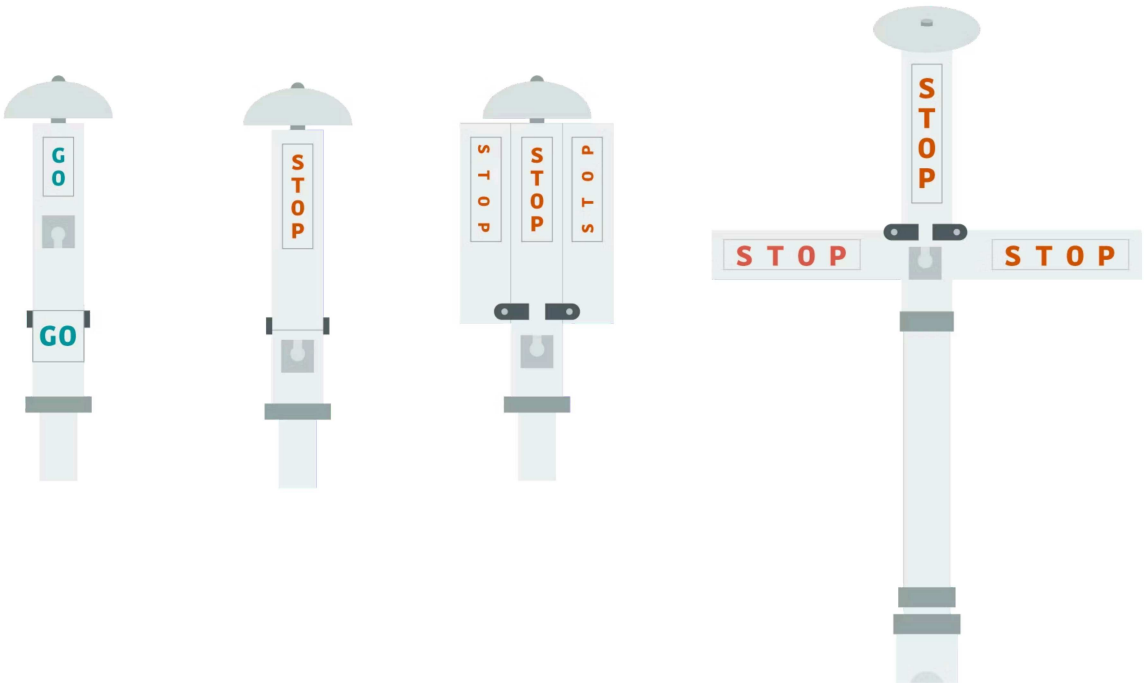
Erfinder, Programmierer und Ingenieure lieben es, wenn der Inspiration freien Lauf gelassen wird. Aber selbst brillante Ideen können selten gleich beim ersten Mal in die Tat umgesetzt werden. Ihr erinnert euch vielleicht daran, dass das Team von Thomas Edison mehr als 1.000 Prototypen der Glühbirne entwickelte, bevor sich ihre Idee eines Glühfadens in einem Glaskasten als wirklich erfolgreich erwies. Schwierigkeiten und Sachen, die übersehen wurden, müssen verbessert werden, bevor eine Idee wie gedacht funktioniert. Diese Konstruktionsfehler werden Bugs genannt.

Ein tragisches Beispiel für einen Fehler findet sich in der Geschichte der Ampel.

Selbst in der Ära der Pferdekutschen war es notwendig, den Verkehr zu lenken. Ein britischer Eisenbahnmanager namens John Peake Knight schuf 1868 das erste Straßenverkehrssignal. Im Gegensatz zu modernen Verkehrssignalen wurde das Gerät von einem Polizeibeamten bedient, der das Signal manuell änderte. Er benutzte Gaslichter, um die Worte auf der Beschilderung zu beleuchten. Leider verursachte ein Gasleck nur wenige Wochen nach der Installation des Geräts eine Explosion, die den Beamten schwer verletzte.

Jahrzehntelang galten die Geräte als zu gefährlich, um sie zu benutzen. Aber die Idee war zu vielversprechend, um in Vergessenheit zu geraten, und spätere Erfinder schufen daraus Variationen.

Im Jahr 1910 entwarf ein Amerikaner namens Ernest Sirrine die erste automatisch gesteuerte Ampel. Zwei Schilder, auf denen abwechselnd "Stopp" und "Weiter" angezeigt wurden. Zwei Jahre später ersetzte ein Entwurf von Lester Farnsworth Wire die Schilder durch rote und grüne Lichter und brachte uns unserer modernen Ampel näher. Zahlreiche Erfinder schufen Variationen dieses Konzepts. Dazu gehörte die Hinzufügung eines gelben "Vorsicht"-Lichts, um Unfälle zu vermeiden, die durch den plötzlichen Wechsel von Grün auf Rot verursacht werden. 1923 schuf Garrett Morgan, ein prominenter afroamerikanischer Erfinder in Cleveland, Ohio, ein Signal, das die zeitliche Abstimmung der Signale für einen reibungsloseren Verkehrsfluss verbesserte. (Morgan spielte auch eine Rolle bei der Erfindung der Gasmaske).



Unzählige weitere Verbesserungen wurden im Laufe der Jahre bei der Gestaltung von Verkehrssignalen vorgenommen. Das Verbessern von Designfehlern kann oft eine große Quelle für neue Erfindungen sein.

Aber woher kam der Begriff Bug überhaupt? Aus seinen Briefen geht hervor, dass er von Thomas Edison stammt. Eine etwas spannendere Geschichte über den Begriff dreht sich jedoch um Navy Konteradmiral Grace Hopper.

Grace Hopper wurde während des Zweiten Weltkriegs in die Marine aufgenommen. Im Laufe ihrer Karriere wurde sie eine der ersten Informatikerinnen der Welt. Sie arbeitete an einem frühen elektromechanischen Computer namens Mark I, der Informationen für militärische Operationen berechnete. Nach dem Krieg arbeitete sie auch an dem Nachfolger des Computers, Mark II.

Eines Tages führte eine Fehlfunktion bei Mark II dazu, dass die Operation abgebrochen wurde. Hoppers Team untersuchte diese und fand eine Motte in einem der elektrischen Relais. Das Team klebte die Motte auf eine Seite in ihrem technischen Logbuch, wo sie immer noch im Smithsonian National Museum of American History zu sehen ist. Dieses Ereignis war ein buchstäbliches "Debuggen" bzw. Fehlersuchen.

Hopper war einer der ersten Wissenschaftler, der voraussah, dass es möglich war, eine Programmiersprache mit englischen Wörtern zu schaffen. Folglich entwickelte sie den ersten Compiler, ein System, das menschliche Sprache in Maschinensprache umwandelte. Als 1959 die Computersprache COBOL geschaffen wurde, war Hopper eine Beraterin. Sie starb 1992 aber Informatiker bezeichnen sie immer noch liebevoll mit ihren Spitznamen "Amazing Grace" und "Grandma COBOL".

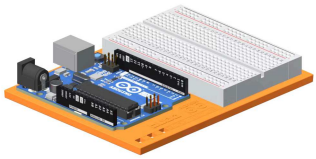
Grundlegende Ampelschaltung

In dieser ersten Aktivität verdrahtet ihr eine einfache Ampelschaltung mit drei LEDs. Nachdem die Schaltung aufgebaut ist, schreibt ihr ein Programm und ladet es auf das Arduino UNO R3 Board, um die Ampel zu steuern.

TEACHER NOTES

In dieser Aktivität bauen die Schülerinnen und Schüler mit Hilfe von LEDs und Widerständen eine Schaltung, die einer Ampel ähnelt. Jede LED wird durch einen anderen Pin auf dem Arduino UNO R3 Board gesteuert.

Benötigte Materialien



1 ARDUINO PROJEKT BOARD

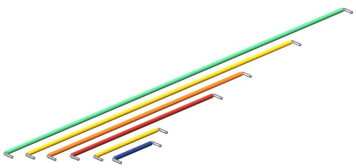
3 220 Ω-WIDERSTAND

1 ROTE LED

1 GELBE LED

1 GRÜNE LED

1 SCHWARZE STROMKABEL



1 STECKVERBINDER **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

Stromkabel: Bei den Stromkabeln handelt es sich um die langen roten und schwarzen Steckverbinder, die an jedem Ende eine rechteckige Stiftbefestigung haben.

ERFAHREN SIE MEHR: Arten von Geräten

VERSTÄNDNIS-ÜBERPRÜFUNG

Classify the following devices as digital or analog:

- Ein/Aus-Lichtschalter
- Lautstärkeregler
- Der Blinker eines Autos
- Rauchmelder
- Scheibenwischersteuerung
- Multimeter

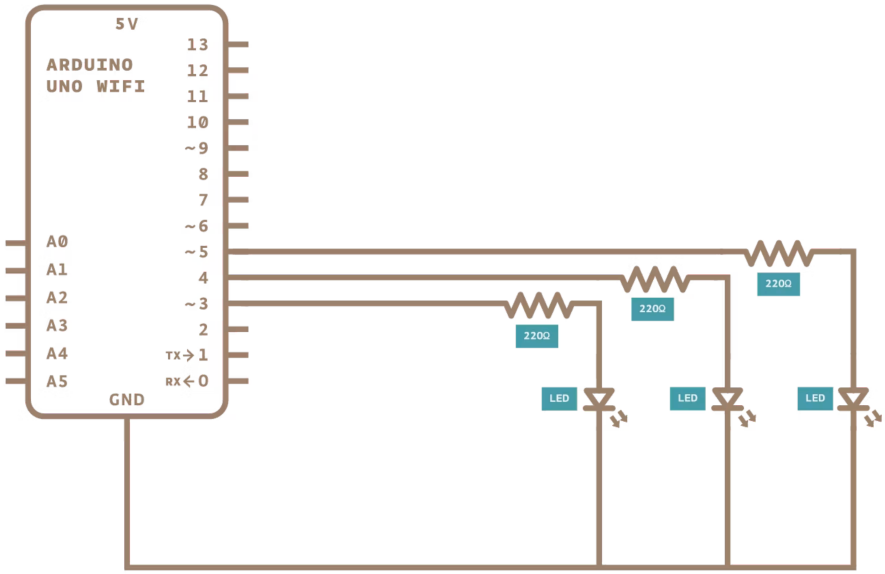
FURTHER NOTES

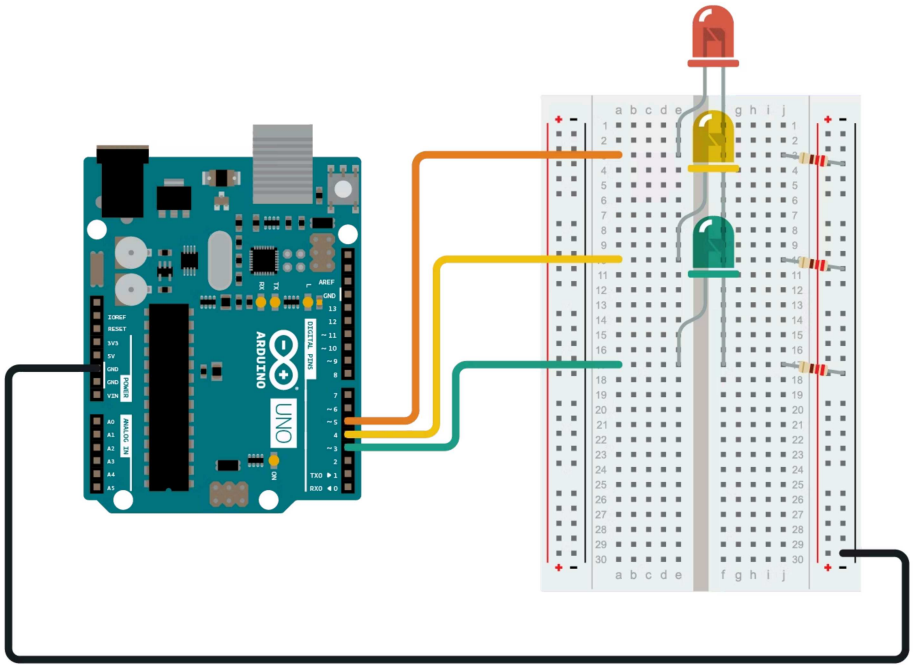
VERSTÄNDNIS-ÜBERPRÜFUNG ANTWORTEN

- Ein/Aus-Lichtschalter - digital
- Lautstärkeregler - analog
- Der Blinker eines Autos - digital
- Rauchmelder - die meisten sind digital
- Scheibenwischersteuerung - kann analog oder digital sein

- ◇ Multimeter - meist analog (einige Funktionen wie z.B. Leitfähigkeit können digital sein)

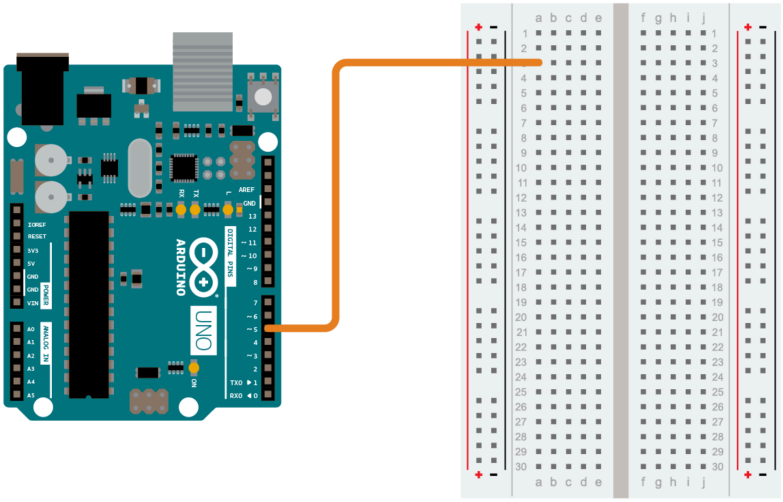
Schaltplan



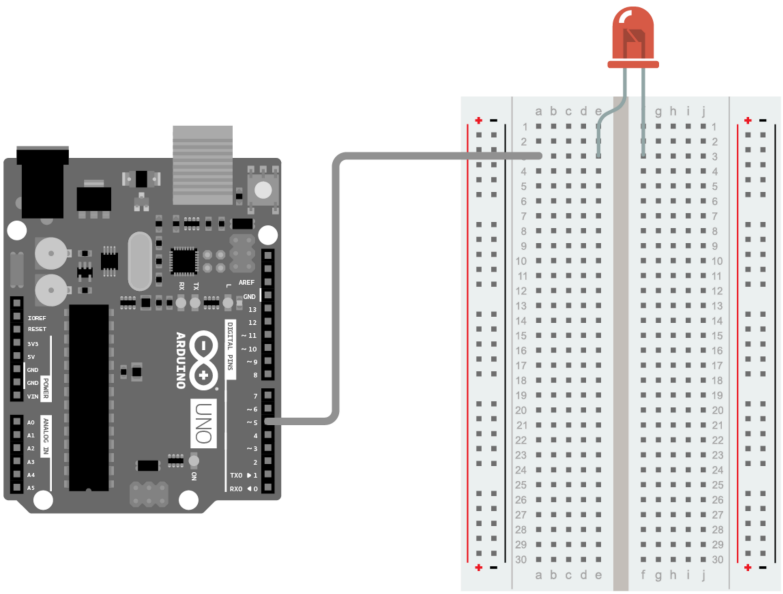


Aufbau der Schaltung

Verwendet den Schaltplan und den Verdrahtungsplan oder die folgende Bilderfolge um die Schaltung aufzubauen. Step 1 Verwendet einen Steckverbinder, um den digitalen Pin 5 auf dem Arduino UNO R3 Board mit Loch 3a auf dem Steckbrett zu verbinden.

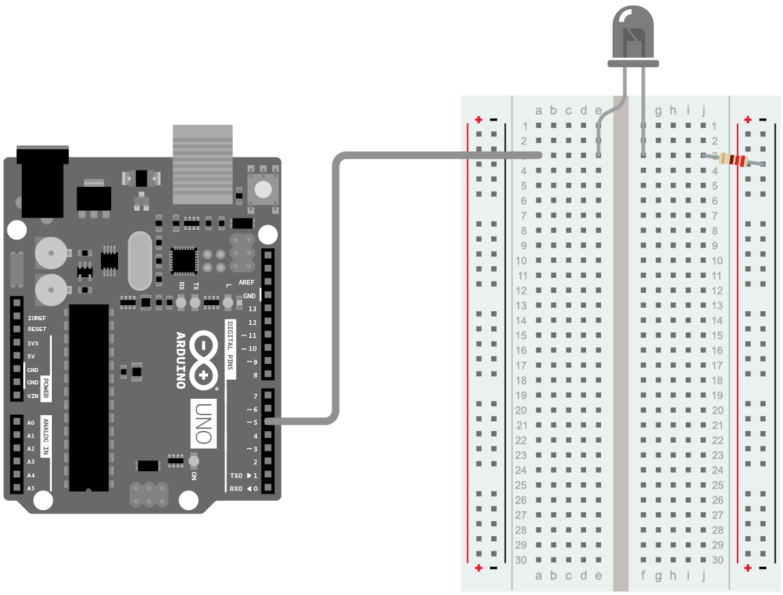


Step 2 Nehmt eine rote LED und steckt sie mit der Anode in Loch 3e. Steckt die Kathode in Loch 3f.

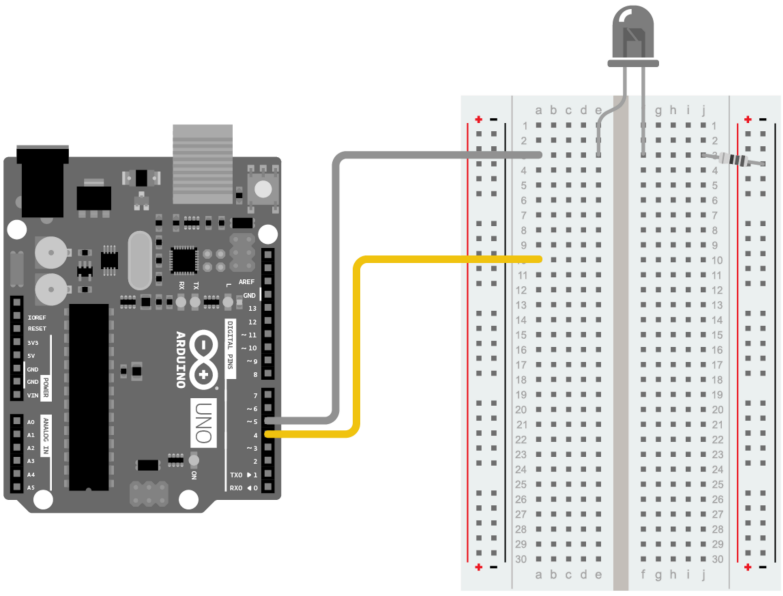


Step 3 Nehmt einen 220-Ohm-Widerstand und steckt einen seiner Anschlussdrähte in Loch 3j. Steckt den anderen Anschlussdraht des Widerstands in die negative Leiste rechts auf

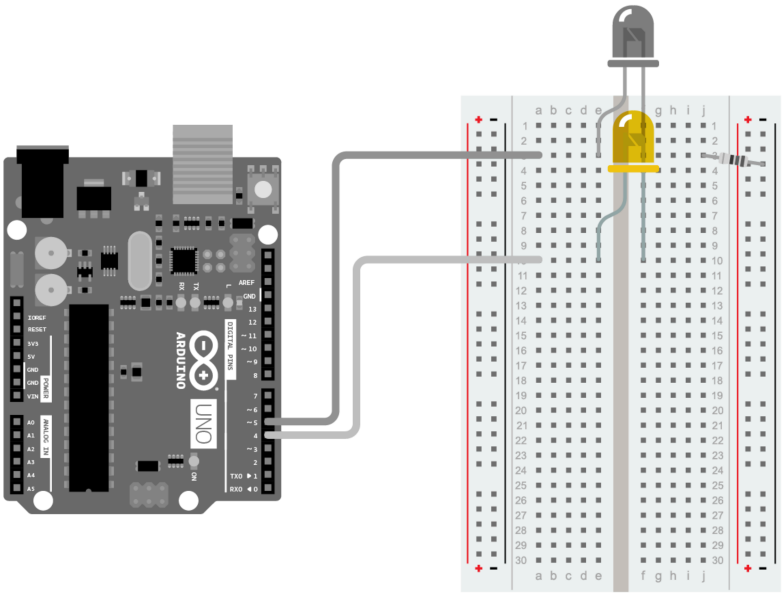
dem Steckbrett.



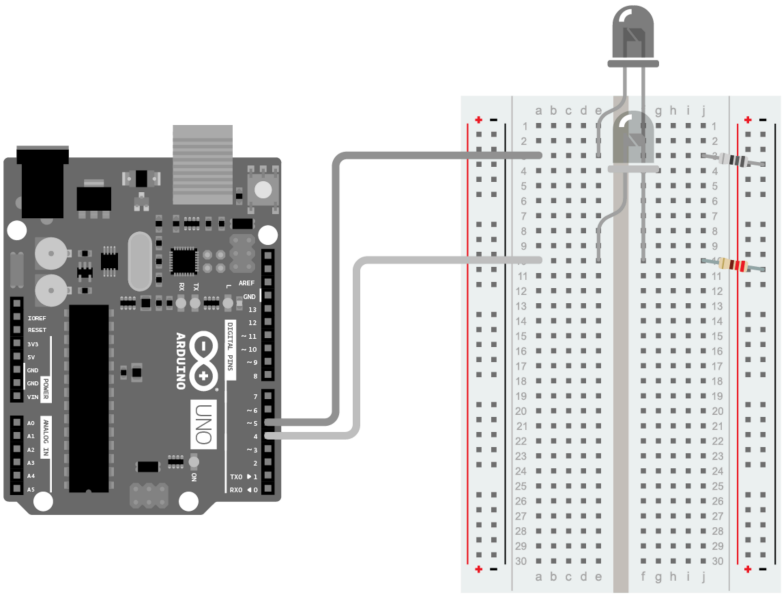
Step 4 Verwendet einen weiteren Steckverbinder, um den digitalen Pin 4 auf dem Arduino UNO R3 Board mit Loch 10a auf dem Steckbrett zu verbinden.



Step 5 Fügt eine gelbe LED dem Steckbrett hinzu. Steckt die Anode in Loch 10e und die Kathode in Loch 10f.

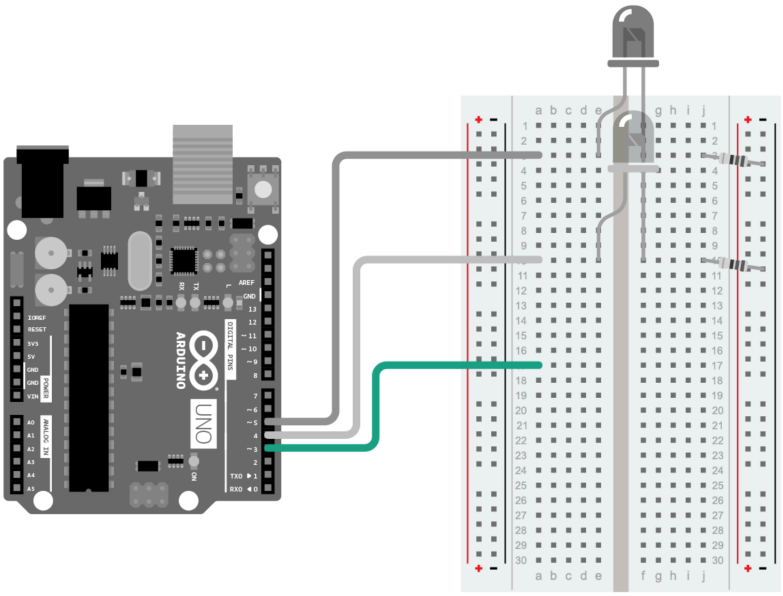


Step 6 Verwendet einen weiteren 220-Ohm-Widerstand, um die gelbe LED mit Ground zu verbinden. Steckt dazu den einen Draht des Widerstandes in Loch 10j und den anderen in die negative Leiste auf der rechten Seite des Steckbretts.

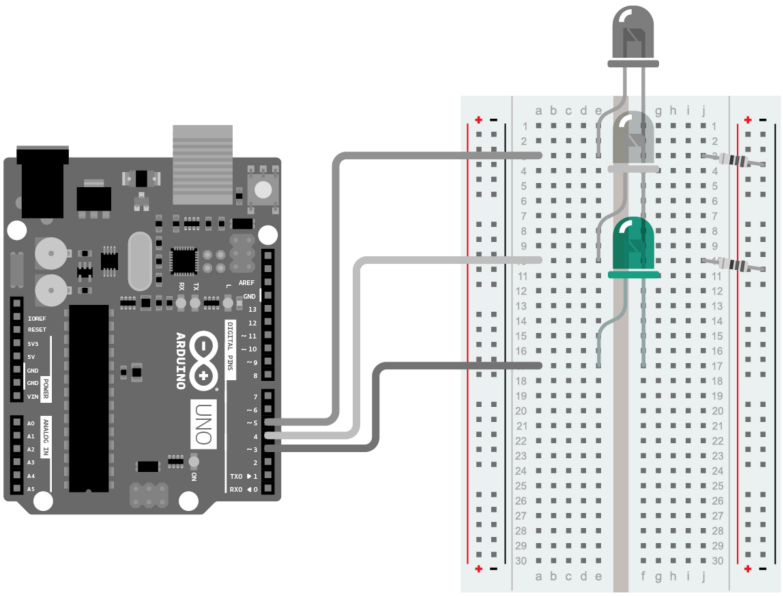


Step 7 Eine grüne LED wird über den digitalen Pin 3 auf dem Arduino UNO R3 Board gesteuert. Verwendet einen Steckverbinder, um den digitalen Pin 3 mit Loch 17a auf dem

Steckbrett zu verbinden.

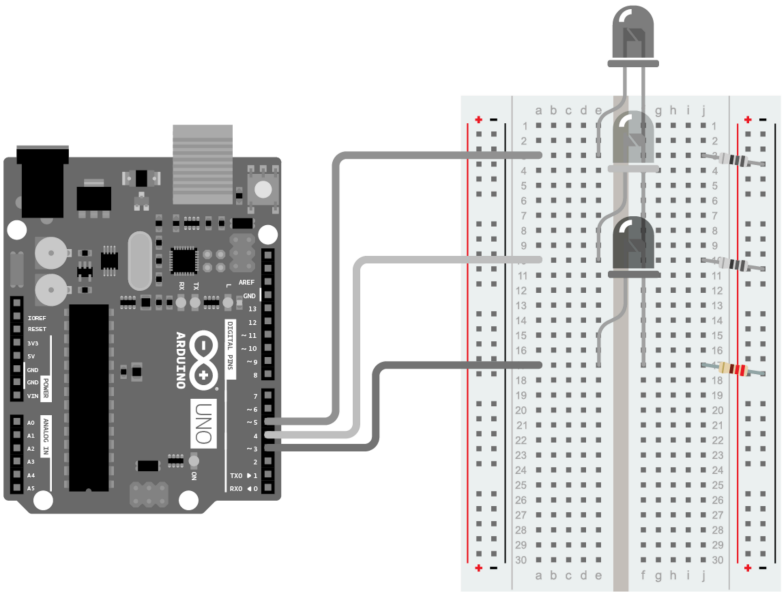


Step 8 Steckt eine grüne LED in Reihe 17 des Steckbretts. Die Anode der LED wird in Loch 17e und die Kathode in Loch 17f gesteckt.

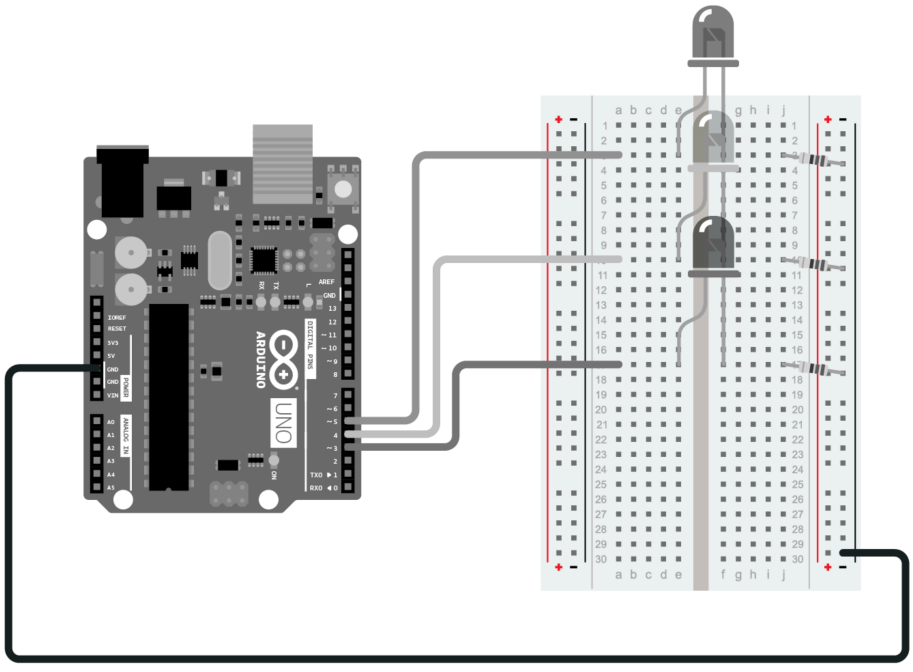


Step 9 Verwendet einen weiteren 220-Ohm-Widerstand, um die grüne LED mit Ground zu verbinden. Ein Draht des Widerstandes wird in Loch 17j gesteckt und der andere in die negative Leiste auf dem Steckbrett. Euer Stromkreis ist fast fertig. Ist euch jedoch aufgefallen, dass der Kreislauf kein geschlossener Stromkreis ist? Es fehlt eine Komponente.

Könnt ihr herausfinden, was fehlt?

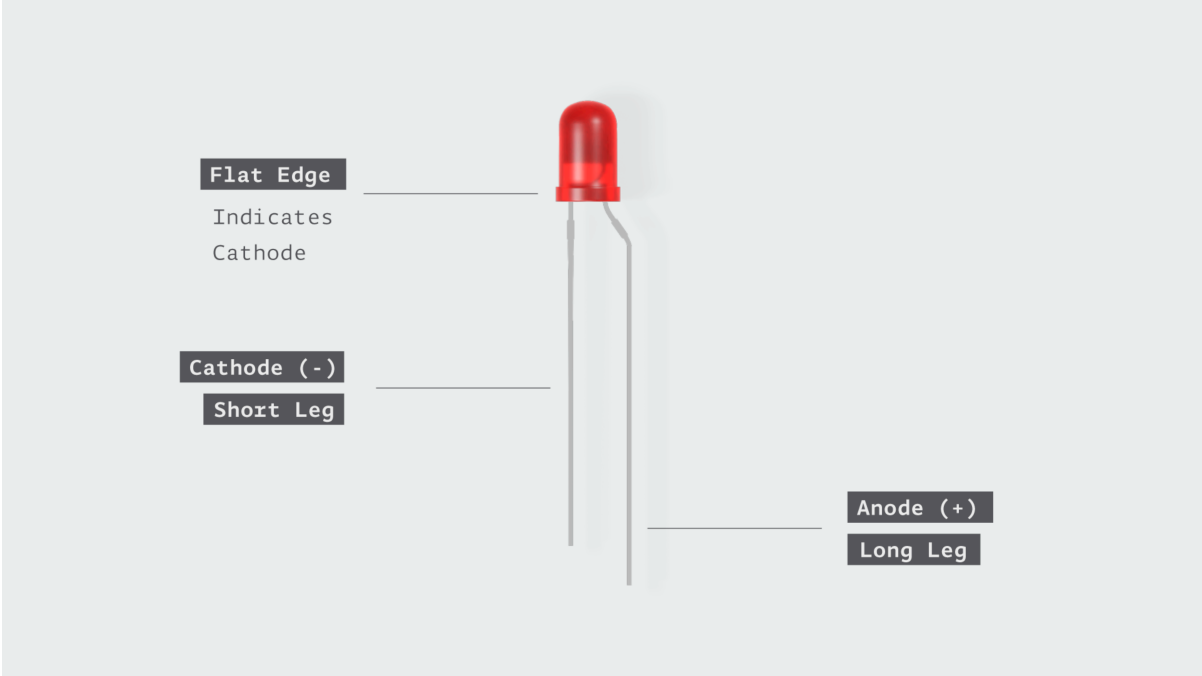


Step 10 Der Stromkreis ist nicht vollständig, weil eure negative Leiste nicht mit dem Ground Pin verbunden ist. Alle drei Zweige des Stromkreises laufen an der negativen Leiste auf der rechten Seite des Steckbretts zusammen. Ihr müsst die Leiste mit dem Ground Pin auf dem Arduino UNO R3 Board verbinden, um den Stromkreis zu vervollständigen. Verwendet das schwarze Stromkabel, um ein beliebiges Loch von der negativen Leiste auf der rechten Seite des Breadboards mit einem der Ground Pin (mit GND gekennzeichnet) auf dem Arduino UNO R3 Board zu verbinden.



FURTHER NOTES

Möglicherweise müsst ihr euch vielleicht daran erinnern, welcher Draht die Anode und welcher Draht die Kathode der LED ist.



Ihr habt nun eine vollständige Parallelschaltung mit drei Zweigen, einen für jede LED-Farbe. Die Stromversorgung für jeden Zweig wird über die digitalen Pins auf dem Arduino UNO R3 Board gesteuert. Diese Pins dienen als Schalter, die den Stromfluss ein- und ausschalten.

FURTHER NOTES

Auf dem Arduino UNO R3 Board sind drei Pins als Masse/Ground (GND) gekennzeichnet. Alle diese Pins funktionieren gleich, und die Schülerinnen und Schüler können ihre Schaltung mit jedem dieser Pins verbinden.

Code-Erstellung - LEDs zum Leuchten bringen

In dieser Aktivität lernt ihr, wie ihr die am Arduino UNO R3 Board angeschlossenen LEDs an- und ausschalten könnt.

FURTHER NOTES

Es kann schwierig sein, den Code von weit entfernten Bildschirmen zu lesen. Unter Datei > Einstellungen die Schriftgröße auf einen Wert zwischen 18 und

24 erhöhen. Dadurch wird es einfacher, auf ihren Bildschirmen aus der Ferne mitzulesen..

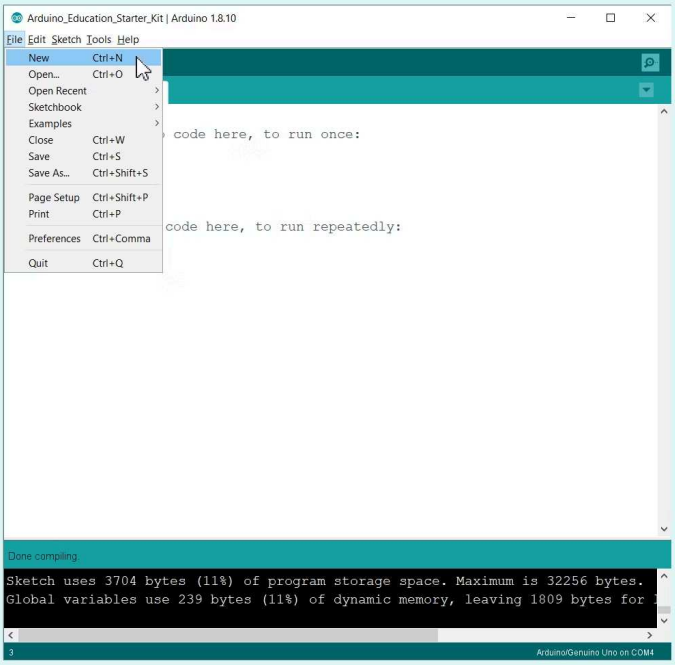
TEACHER NOTES

Bei dieser Aktivität schreiben die Schülerinnen und Schüler ihr erstes Programm mit der Arduino IDE. Sie werden mit diesen Befehlen vertraut gemacht:

- **pinMode()** – legt einen digitalen Pin als und INPUT oder OUTPUT fest
- **digitalWrite()** – sendet ein digitales Signal an den Stromkreis

1) Öffnet die Arduino IDE.

2) Klickt im Menübereich auf Datei und dann auf **Neu**. Dadurch wird ein neuer Sketch in einem neuen Fenster geöffnet. Ihr könnt das andere Sketch Fenster schließen.

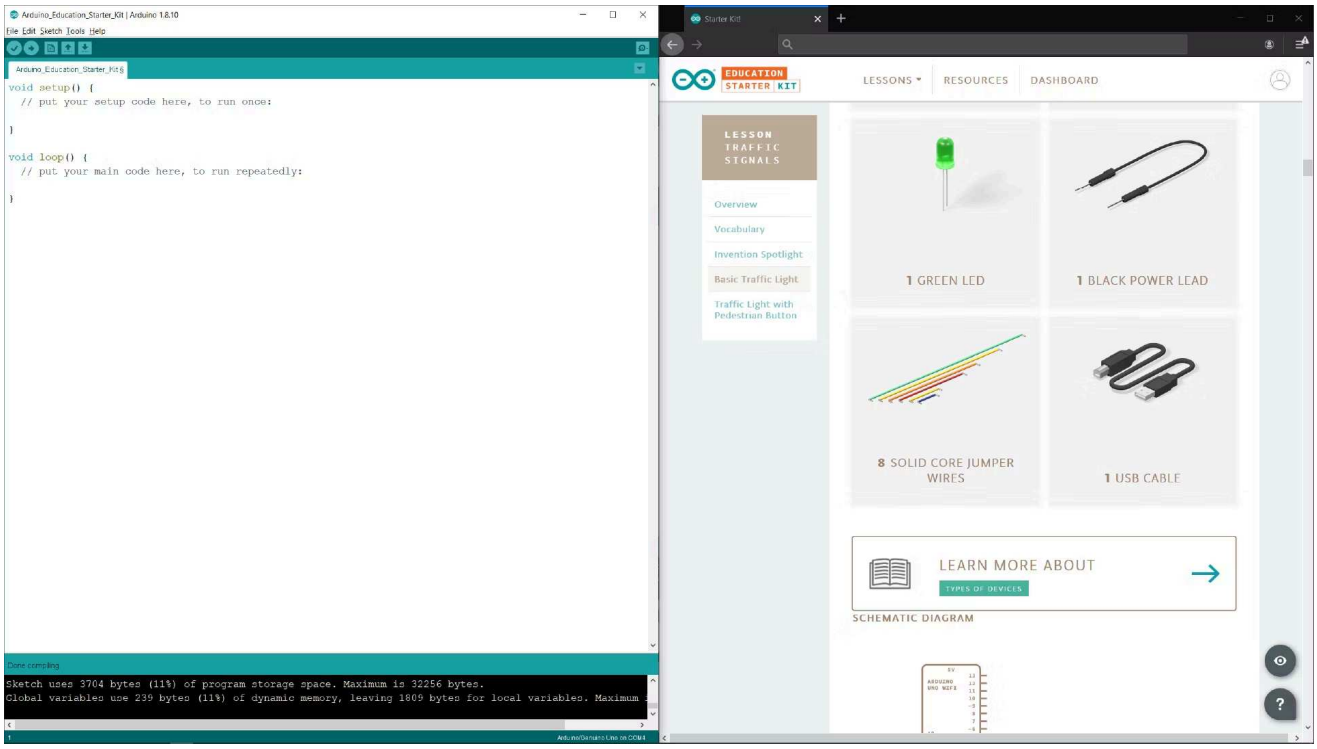


Es gibt zwei verschiedene Möglichkeiten um sich die Arduino IDE und die Anweisungen aus diesem Kurs anzeigen zu lassen. Verwendet die Methode, die für euch am besten geeignet ist.

Methode 1 – Stellt das Arduino IDE Fenster auf Vollbildmodus ein. Wechselt dann zwischen dem IDE Fenster und den Kursanweisungen mithilfe der Taskleiste am unteren Bildschirmrand hin und her.

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 die Sicht möglicherweise erschwert, aber der Vorteil ist, dass ihr nicht hin- und herschalten müsst.



3) Speichert den Sketch unter einem neuen Namen. Wählt im Menübereich Datei und klickt auf **Speichern unter...** Der Speicherort des Sketchs sollte standardmäßig das Arduino Sketchbuch sein. Wenn das nicht der Fall ist, fragt euren Lehrer, wo ihr die Skizze auf eurem Computer speichern sollt. Nennt die Datei "Lektion3_V1_", gefolgt von deinen Initialen und denen deines Partners. Klickt auf **Speichern**.

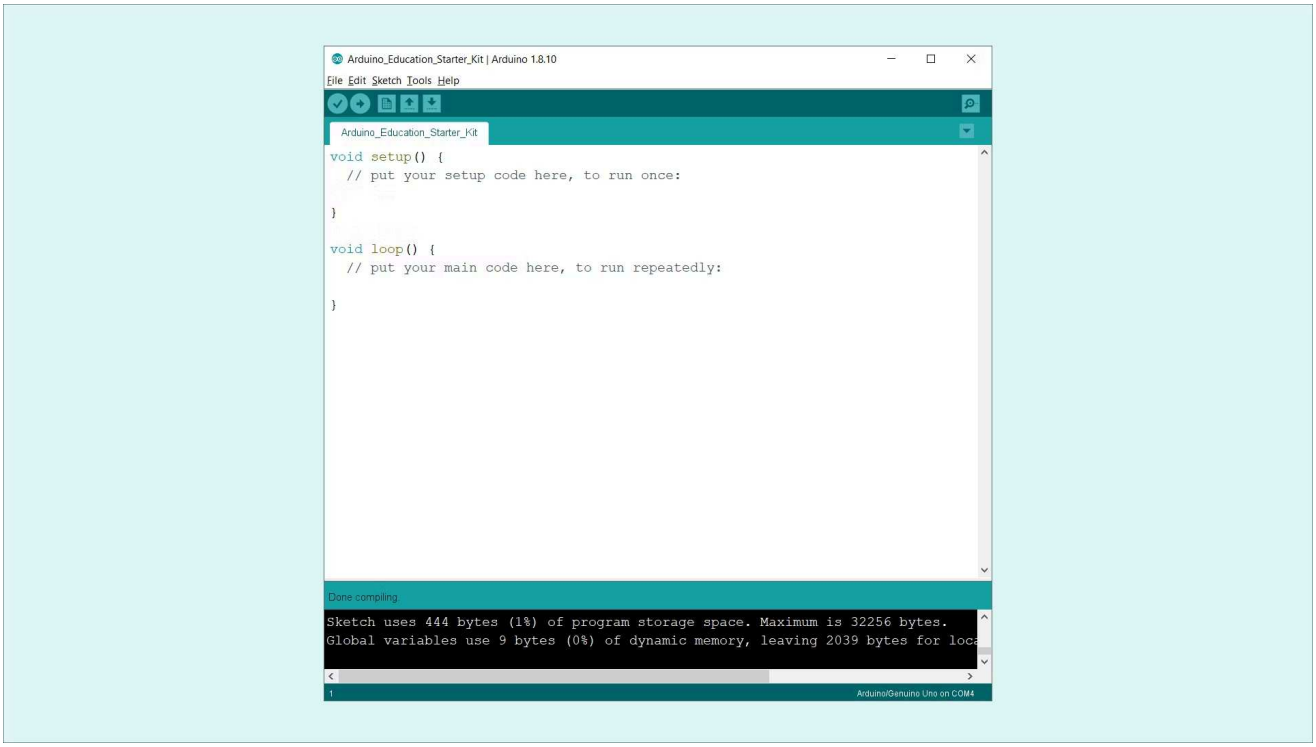
Hinweis: Wenn ihr Sketche speichert, ist es ratsam, größere Änderungen als eine neue Version des Sketchs zu speichern. Auf diese Weise könnt ihr auf eine frühere Version des Sketchs zurückgreifen, wenn im Sketch etwas schief geht und ihr das Problem nicht feststellen könnt. Wenn ihr in jeder Lektion Sketche speichert, erhöht einfach jedes Mal die Versionsnummer (V1), wenn ihr größere Änderungen vornehmt.

FURTHER NOTES

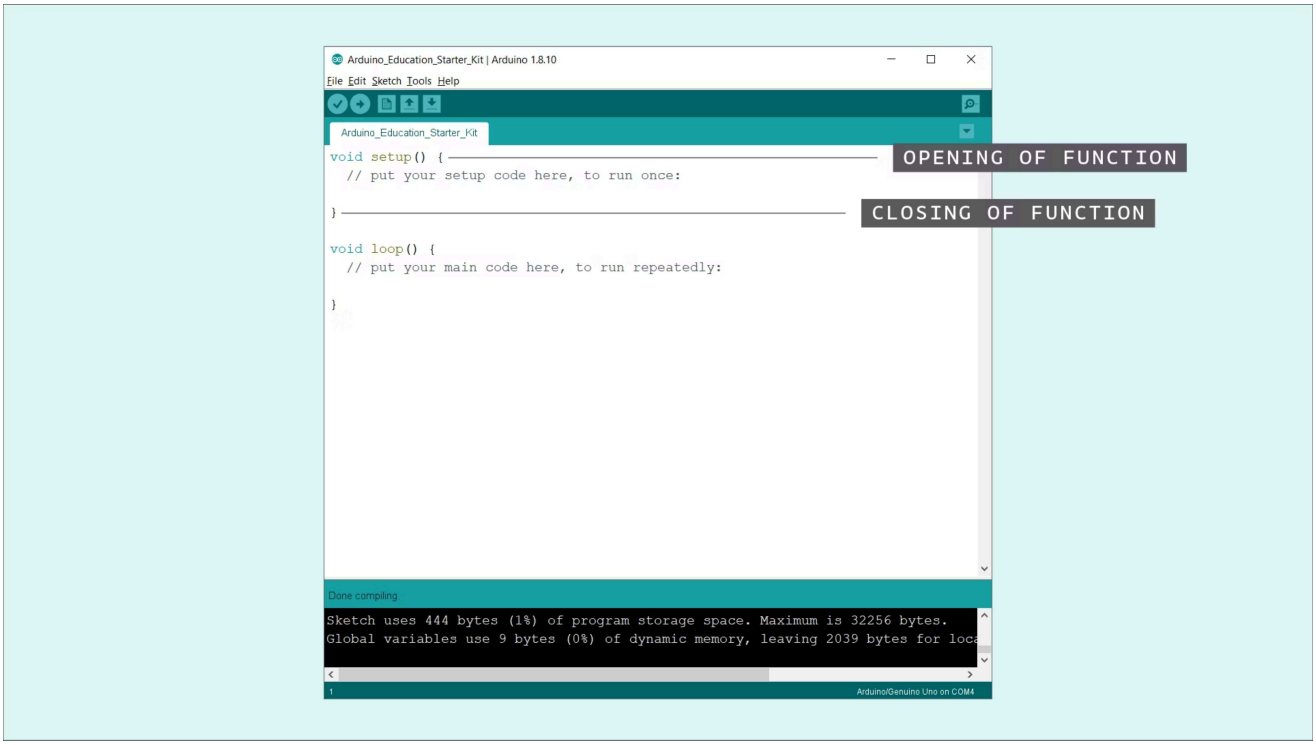
Arduino Sketch-Dateinamen dürfen keine Leerzeichen enthalten. Beim Speichern eines Sketchs werden Leerzeichen automatisch in Unterstriche umgewandelt.

4) Beachtet, dass es in euren neuen Sketches zwei Abschnitte gibt - einen Abschnitt **void setup()** und einen Abschnitt **void loop()**. Diese werden Funktionen genannt. Jeder Arduino Sketch hat diese beiden Funktionen. Die Funktion **void setup()** enthält die Befehle, die ihr zu Beginn eures Programms einmal ausführen möchtet. Die **void loop()**-Funktion enthält die Befehle, die euer Arduino UNO R3 Board kontinuierlich laufen lassen oder ausführen

soll.



Die Befehle für die Funktion **void setup()** oder **void loop ()** müssen zwischen den geschweiften Klammern für diese Funktion eingegeben werden. Die geöffnete geschweifte Klammer - { - zeigt den Beginn der Funktion an, und die geschlossene geschweifte Klammer - } - zeigt das Ende der Funktion an.



Die erste Zeile der Funktion **void setup()** enthält einen Codekommentar. Codekommentare beginnen mit // und werden verwendet, um euch zu helfen, den Überblick darüber zu behalten, was eine Funktion oder Codezeile tut. Codekommentare werden normalerweise in Alltagssprache geschrieben, aber ihr könnt in einen Codekommentar alles einfügen, was ihr

möchtet. Wenn das Arduino UNO R3 Board das Symbol // sieht, ignoriert es den Rest dieser Zeile.

FURTHER NOTES

Die Verwendung von Codekommentaren ist selbst für die erfahrensten Programmierer eine gute Praxis. Sie helfen den Programmierern dabei, :

- festzustellen, wer den Code geschrieben hat und wann.
- den Code in Abschnitte zu organisieren.
- zu verfolgen, was eine Zeile oder ein Codeabschnitt tut.
- Code zu markieren, der fehlerfrei ist.
- es einem anderen Programmierer leicht zu machen, der Logik eines Programms zu folgen.
- Code zu debuggen, der nicht funktioniert.
- Zeilen oder Codeabschnitte einfach ein- oder auszublenden.
- Notizen zu bestimmten Parametern oder Einstellungen innerhalb des Codes zu machen, während der Code modifiziert wird.

Hinweis: Neben den geschweiften Klammern haben auch die Funktionen **void setup()** und **void loop()** Klammern. Die Klammern haben einen anderen Zweck, der in diesem Kurs nicht behandelt wird. Es ist jedoch wichtig, Klammern und geschweifte Klammern nicht zu verwechseln.

5) In eurem Stromkreis werden drei digitale Pins des Arduino UNO R3 Boards verwendet. Jeder der digitalen Pins auf dem Board kann als Output Pin oder als Input Pin eingerichtet werden. Output Pins liefern fünf Volt Spannung. Input Pins lesen aus, ob die Spannung zum Arduino UNO R3 Board zurückkehrt. Für euren Stromkreis müsst ihr die Pins 3, 4 und 5 als Output Pins einrichten.

Der Befehl zum Festlegen, ob ein Pin ein Input oder Output ist, lautet **pinMode(pin, STATE)**. Der *Pin* kann ein beliebiger Pin auf de Arduino UNO R3 Board sein. Der *STATE* ist entweder **INPUT** oder **OUTPUT**. Beachtet die Großschreibung des Befehls und des Status. Damit das Arduino UNO R3 Board den Befehl erkennt, müssen sowohl der Befehl als auch der Status in Großbuchstaben genau so dargestellt werden, wie es hier angezeigt wird.

Beispiele:

```
pinMode (10, INPUT);

pinMode(A1, OUTPUT);
```

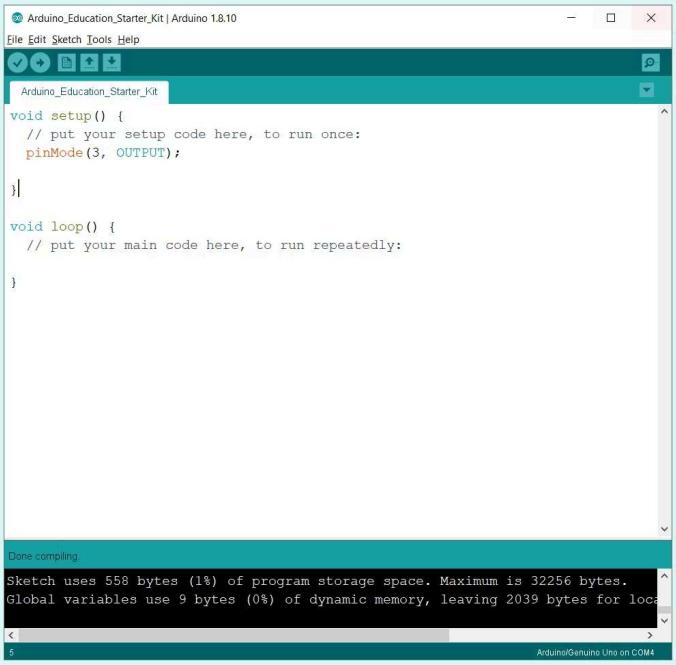
Es ist Zeit, der Funktion **void setup()** ihre erste Codezeile hinzuzufügen. Gebt unter dem ersten Codekommentar den folgenden Befehl ein:

> **Hinweis:** Beachtet, dass der Befehl mit einem Semikolon endet. In der Arduino IDE muss jeder Befehl mit einem Semikolon enden.

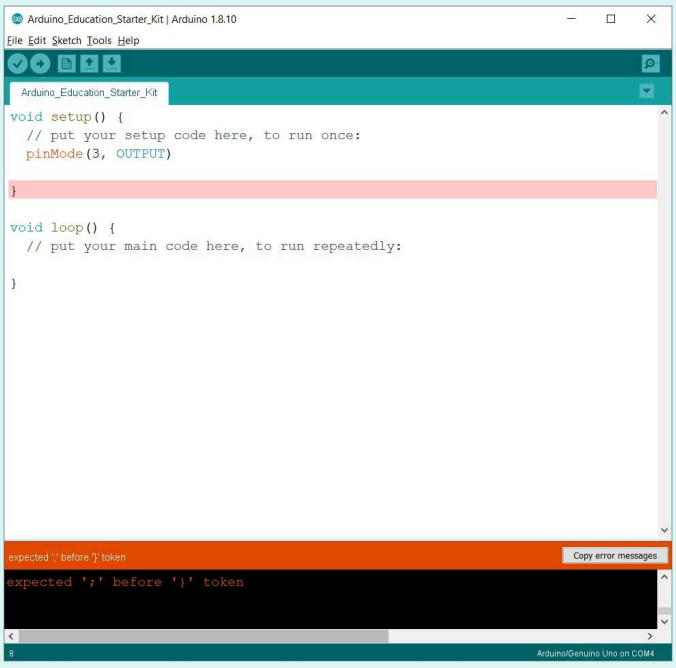
6) Um sicherzustellen, dass ihr den Befehl korrekt eingegeben habt, könnt ihr den Code **überprüfen**. Wenn ihr den Code überprüft, **kompiliert** die Arduino IDE den Code und liest ihn, um sicherzustellen, dass jeder Befehl verstanden werden kann. Wenn der Code kompiliert wird, übersetzt der Computer die Befehle, die ihr auf Englisch geschrieben habt, in eine Sprache, die der Computer verstehen kann.

Um euren Code zu verifizieren, klickt auf die Schaltfläche mit dem Häkchen in der linken oberen Ecke des Fensters.

Wenn euer Code maschinenlesbar ist, erscheint am unteren Bildschirmrand eine Meldung, die anzeigt, dass die Kompilierung des Codes abgeschlossen ist. Der schwarze Bereich am unteren Bildschirmrand zeigt euch auch Informationen über den von eurem Programm belegten Speicherplatz an.



Wenn der Compiler euren Code nicht verstehen kann, erhaltet ihr eine orangefarbene Fehlermeldung, die das Problem erklärt.



FURTHER NOTES

Beim Versuch, Code zu kompilieren, stoppt die Arduino IDE den Prozess, sobald sie einen Fehler in der Syntax findet. Wenn also mehrere Fehler in einem Sketch vorhanden sind, kann immer nur ein Fehler auf einmal identifiziert werden. Es ist

wichtig, dass die Schülerinnen und Schüler sich angewöhnen, den Code oft zu verifizieren.

Wenn euer Code einen Fehler aufweist, müsst ihr euren Code debuggen. **Debuggen** bedeutet, nach Fehlern in der Syntax oder Logik eures Codes zu suchen. Hier sind ein paar Dinge, auf die ihr achten solltet:

- Haben Sie das M im pinMode großgeschrieben?
- Habt ihr beim Tippen von OUTPUT alles in Großbuchstaben geschrieben?
- Habt ihr in eurem pinMode-Befehl sowohl eine geöffnete als auch eine geschlossene Klammer verwendet?
- Habt ihr am Ende des Befehls pinMode ein Semikolon hinzugefügt?
- Habt ihr alles richtig geschrieben? Wenn ihr einen Fehler gemacht haben, korrigiert euren Code und überprüft ihn dann erneut. Fahrt mit dem Debugging eures Codes so lange fort, bis euer Sketch frei von Fehlern ist.

Hinweis: Das Debuggen von Code gehört zum Programmieren dazu. Tatsächlich schätzen einige Leute, dass professionelle Programmierer 25% ihrer Zeit mit dem Schreiben von Code und die restlichen 75% mit dem Debuggen verbringen. Fühlt euch also nicht schlecht, wenn ihr Fehler in eurem Code habt. Das ist ein Teil des Programmierens. Programmierer sind auch gute Problemlöser. Sie können ihre Programmierfehler finden und sie mit funktionierenden Lösungen beheben.

TEACHER NOTES

TIPPS ZU FEHLERMELDUNGEN:

- Wenn Ihre Schülerinnen und Schüler beim Kompilieren eine Fehlermeldung erhalten, hilft die Arduino IDE bei der Diagnose des Problems. Manchmal kann die Interpretation der Fehlermeldung knifflig sein. Je nach Fehler kann es hilfreich sein, mit der Schaltfläche "Fehlermeldungen kopieren" die gesamte Fehlermeldung zu kopieren und dann in ein anderes Dokument einzufügen. Auf diese Weise können Sie die gesamte Fehlermeldung auf einmal sehen.
- Oft wird in der Arduino IDE eine Zeile des Codes hervorgehoben. In vielen Fällen tritt der Fehler direkt vor oder direkt nach dieser hervorgehobenen Zeile auf. Wenn zum Beispiel am Ende eines Befehls ein Semikolon fehlt, hebt die Arduino IDE den Befehl direkt unter der Zeile hervor, in der das Semikolon fehlt.

7) Neben Pin 3 werden in eurem Stromkreis zwei weitere Pins - Pin 4 und 5 - auf dem Arduino UNO R3 Board zur Steuerung von LEDs verwendet. Fügt unter dem ersten Befehl **pinMode()** zwei weitere Befehle hinzu, um mit diesen **pinMode()** Befehlen Pin 4 und 5 als Output Pin festzulegen.

8) Verifiziert euren Code erneut, um ihn auf Fehler zu überprüfen. Debuggt euren Code, wenn ihr einen Fehler gemacht habt. Überprüft noch einmal, ob eure **void setup()**-Funktion wie folgt aussieht:

Hinweis: Überprüft euren Code häufig. Dies wird euch helfen, euren Code klar zu strukturieren, während ihr ihn schreibt.

ERFAHREN SIE MEHR: **Programmier-Syntax**

9) Kommentare zum Code sind hilfreich, um sich daran zu erinnern, was Zeilen oder Abschnitte des Codes bewirken. Ändert den Codekommentar für die **void setup()**-Funktion, damit sie besser beschreibt, was die Codezeilen in diesem Abschnitt tun. Ihr könnten zum Beispiel den Codekommentar ändern um Folgendes zu sagen:

```
// die LED-Pins als Outputs festlegen
```

Wie auch immer ihr den Kommentar ändert, denkt daran, dass alle Codekommentare mit // beginnen müssen. //.

10) Ihr seid nun bereit, zur Funktion **void loop()** überzugehen. Dies ist der Hauptabschnitt des Sketchs oder des Programms. Um eine LED einzuschalten, die an das Arduino UNO R3 Board angeschlossen ist, muss das Board den Stromkreis mit Spannung versorgen. Das Board kann Strom über jeden der digitalen oder analogen Pins senden. Für die digitalen Pins ist **digitalWrite(pin, STATE)** der Befehl, der steuert, ob die Spannung für jeden Pin ein-

oder ausgeschaltet wird. Der *Pin* kann jeder der digitalen Pins (0 bis 13) auf dem Arduino UNO R3 Board sein. Der *STATE* ist entweder **HIGH** (liefert fünf Volt) oder **LOW** (liefert null Volt).

Beispiele:

digitalWrite(10, HIGH); - liefert 5 Volt Strom über Pin 10

digitalWrite(7, LOW); - schaltet den Stromfluss an Pin 7 ab

Schreibt unter dem Kommentar für die Funktion **void loop()** einen Befehl zum Einschalten der grünen LED, die an Pin 3 angeschlossen ist. Vergesst nicht, den Befehl mit einem Semikolon zu beenden.

Hinweis: Die Ressourcen können verwendet werden, um Befehle zu überprüfen und wie sie in der Arduino IDE formatiert werden sollten.

11) Überprüft und debuggt euren Code, falls nötig.

12) Fügt nach dem Semikolon in eurem **digitalWrite()**-Befehl einen Codekommentar hinzu, um zu erklären, was diese Codezeile bewirkt. Eure Codezeile könnte zum Beispiel so aussehen:

13) Fügt **digitalWrite()**-Befehle hinzu, um die an Pin 4 und 5 angeschlossenen gelben und roten LEDs einzuschalten. Fügt Codekommentare am Ende jeder Zeile hinzu. Wenn ihr fertig seid, überprüft und debuggt euren Code.

Hinweis: Beim Programmieren ist Kopieren und Einfügen eine großartige Funktion, um ähnliche Codezeilen zu schreiben. Kopiert eine klar strukturierte Zeile oder einen

Abschnitt des Codes, fügt den Code an der richtigen Stelle ein und ändert die Teile, die geändert werden müssen. Überprüft den Code, wenn ihr fertig seid.

TEACHER NOTES

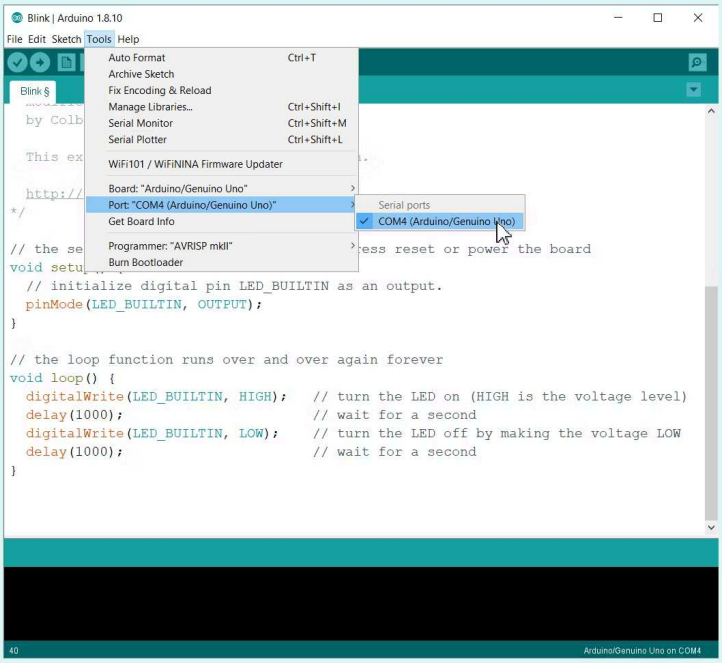
Die Ressourcen enthalten die Beschreibungen und Beispiele für jeden Befehl und jede Funktion, die in diesem Kurs verwendet werden. Lassen Sie die Schülerinnen und Schüler bei Bedarf auf die Ressourcen zugreifen.

14) Nehmt das USB-Kabel, verbindet den flachen USB-Stecker mit dem Computer und das andere Ende mit dem USB-Anschluss auf dem Arduino UNO R3 Board.

Euer Arduino UNO R3 Board sollte ein grünes Licht haben, das anzeigt, dass es eingeschaltet ist.

Hinweis: Wenn das Arduino UNO R3 Board mit Strom versorgt wird, leuchtet möglicherweise auch ein gelbes Licht auf. Dieses Licht ist eine Statusanzeige für Pin 13. ihr könnt dieses Licht vorerst ignorieren.

15) Geht in der Arduino IDE mit dem Mauszeiger im **Werkzeuge**-Menü auf **Port**. Ein Pop-Out-Menü sollte erscheinen, in dem ihr den Port auswählen könnt, über den euer Arduino UNO R3 Board verbunden ist.



FURTHER NOTES

Sobald der Port ausgewählt wurde, sollte die Arduino IDE jedes Mal, wenn das Arduino UNO R3 Board über das USB-Kabel eingesteckt wird, standardmäßig auf diesen Port eingestellt sein. Wenn jedoch ein anderes Arduino Board eingesteckt wird, wird es wahrscheinlich eine andere COM-Nummer haben. Die Schülerinnen und Schüler müssen erneut die Auswahl des richtigen Anschlusses durchlaufen.

Als erstes sollte geprüft werden, ob der richtige Port ausgewählt wurde, falls das Arduino UNO R3 Board und der Computer nicht miteinander zu kommunizieren scheinen.

16) Klickt in der Arduino IDE auf die Schaltfläche **Upload**, um den Code auf das Arduino UNO R3 Board hochzuladen. Die Upload-Schaltfläche ist die Schaltfläche mit einem nach rechts zeigenden Pfeil in der oberen linken Ecke des Fensters.

Der Computer kompiliert den Code und überträgt ihn dann auf das Arduino Board. Während der Übertragung solltet ihr auf dem Arduino UNO R3 Board gelbe Lichter blinken sehen, die anzeigen, dass das Board mit dem Computer kommuniziert.

Wenn die Übertragung abgeschlossen ist, wird der Code ausgeführt, und ihr solltet sehen, dass alle drei LEDs auf dem Steckbrett aufleuchten.

17) Wenn eure LEDs nicht geleuchtet haben, debuggt euren Code und wiederholt Schritt 15. Euer Code sollte ähnlich wie dieser aussehen:

18) Speichert euren Sketch, bevor ihr fortfahrt. Klickt dafür auf **Speichern** oder geht in das Datei-Menü und wählt **Speichern** aus.

TEACHER NOTES

Ermutigen Sie die Schülerinnen und Schüler, Einrückungen zu verwenden, um den Code sauber und ordentlich zu halten. Durch Einrücken lassen sich Codeabschnitte leicht identifizieren. Beispielsweise können die Befehle in der Funktion **void setup()** eingerückt werden, um anzuzeigen, dass sie zu dieser Funktion gehören. Die Einrückung ist besonders dann nützlich, wenn die Schülerinnen und Schüler in späteren Lektionen anfangen, sich mit Bedingungen und verschachtelten Schleifen zu beschäftigen.

FURTHER NOTES

Wie man *Einrücken* zu seinem Code hinzufügt:

- Die Tabulatortaste vergrößert den Einzug einer Zeile.
- Umschalttaste+Tab verringert die Einrückung einer Zeile.

Code-Erstellung - Blinken der LEDs

Bisher haben Sie Befehle zum Einschalten der drei LEDs in Ihrer Schaltung über drei digitale Pins auf der Arduino UNO R3-Platine geschrieben. Jetzt, da Ihre LEDs leuchten, ist es an der Zeit, sie zum Blinken zu bringen.

TEACHER NOTES

Bei dieser Aktivität ergänzen die SchülerInnen ihre Skizze und lassen die LEDs blinken. Sie werden mit diesen Befehlen vertraut gemacht:

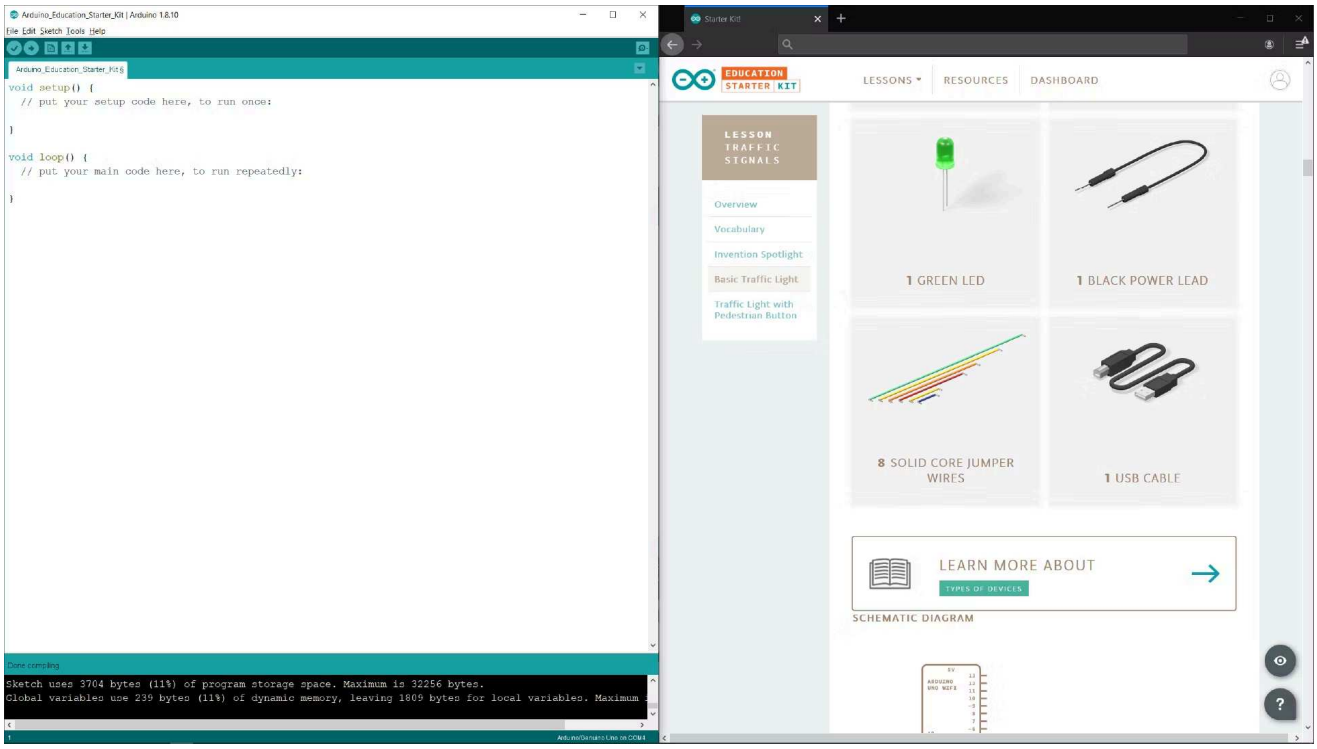
- **delay()** – hält die Skizze/das Programm für eine bestimmte Zeit an

Wenn die Schülerinnen und Schüler diesen Abschnitt ausfüllen, ermutigen Sie sie, weiterhin Kommentare zum Code zu schreiben, um ihren Code zu erklären.

1) Falls erforderlich, startet die Arduino IDE und öffnet euren Sketch zu Lektion3_V1. Schließt das Arduino UNO R3 Board mit dem USB-Kabel an den USB-Anschluss des Computers an.

2) Speichert den Sketch unter einem neuen Namen. Wählt im Menü Datei die Option **Speichern unter...** Der Speicherort des Sketch sollte standardmäßig das Arduino Sketchbuch sein. Wenn dies nicht der Fall ist, fragt euren Lehrer, wo ihr den Sketch auf eurem Computer speichern sollt. Benennt die Datei "Lektion3_V2", gefolgt von euren Initialen und denen eures Partners. Klickt auf **Speichern**.

3) Ordnet euer Lektion3_V2-Sketchfenster und dieses Fenster so an, wie es für euch am besten geeignet ist.



4) Fügt in der **void loop()**-Funktion eurer Skizze drei weitere **digitalWrite()**-Befehle hinzu. Diese Befehle sollten nach den ersten drei Befehlen und vor der schließenden Klammer für die **void loop()**-Funktion hinzugefügt werden. Schreibt je einen Befehl für die Pins 3, 4 und 5.

Damit die LEDs blinken, müsst ihr sie ausschalten. In euren drei neuen **digitalWrite()**-Befehlen sollte der Status LOW statt HIGH sein.

> **Hinweis:** Kopieren und Einfügen kann der beste Freund eines Programmierers sein. Einen Abschnitt sauberen Codes zu kopieren, ihn an einer neuen Stelle einzufügen und seine Parameter zu ändern, kann sowohl schneller als auch sauberer sein. Stellt immer sicher, dass ein Codeabschnitt sauber ist, bevor ihr ihn kopiert und einfügt. Kopiert die ersten drei **digitalWrite()**-Befehle. Fügt eine Kopie dieser Befehle nach den ersten drei Befehlen ein. Ändert den Status von HIGH auf LOW. Ändert eure Codekommentare entsprechend.

Fügt Codekommentare zu euren Befehlen hinzu, die erklären, was jeder Befehl bewirkt. Eure neuen Codezeilen sollten ähnlich wie diese aussehen:

5) Überprüft und debuggt euren Code, falls erforderlich.

6) Klickt auf die Schaltfläche **Hochladen**, um den Sketch auf das Arduino UNO R3 Board zu übertragen. Wenn die Übertragung abgeschlossen ist, wird der Code ausgeführt. Blinken eure Lichter? Höchstwahrscheinlich blinken sie nicht. Besprecht mit eurem Partner, warum ihr glaubt, dass eure Lichter nicht blinken.

7) Syntaxfehler sind Fehler im Code, die der Computer nicht verstehen kann. Häufige Syntaxfehler sind falsch geschriebene Wörter, fehlende Semikolons, fehlende Klammern oder geschweifte Klammern und Schlüsselwörter, die nicht richtig groß geschrieben werden. Die Überprüfungsfunktion in der Arduino IDE hilft bei der Suche nach diesen Fehlern. Allerdings sind nicht alle Fehler Syntaxfehler. Manchmal gibt es auch Logikfehler. Das ist hier der Fall.

Das Problem mit eurem Sketch ist nicht, dass die Lichter nicht blinken - sie blinken. Sie blinken so schnell, dass eure Augen das Blinken nicht sehen können. Codezeilen werden sehr schnell ausgeführt. Tatsächlich könnt ihr mit der Arduino-IDE steuern, wie schnell Befehle auf die Mikrosekunde genau ausgeführt werden. Das ist ein Millionstel einer Sekunde (1/1.000.000).

ERFAHREN SIE MEHR: **Optimierung**

Um euren Code zu reparieren, könnt ihr eine Verzögerung verwenden. Mit einem **delay(time)**-Befehl könnt ihr das Arduino UNO R3 Board anhalten und beliebig lange in Millisekunden warten lassen. In einer Sekunde gibt es 1.000 Millisekunden. Wenn ihr beispielsweise möchtet, dass das Arduino UNO R3 Board zwei Sekunden wartet, bevor es den nächsten Befehl ausführt, würdet ihr folgenden Befehl verwenden:

```
delay(2000);
```

Hinweis: Mit dem Befehl **delayMicroseconds()** könnt ihr eine bestimmte Anzahl von Mikrosekunden festlegen, um das Arduino UNO R3 Board anzuhalten und warten zu lassen. Es gibt 1.000.000 Mikrosekunden in einer Sekunde.

Es ist wichtig zu verstehen, dass das Arduino UNO R3 Board während einer Verzögerung nichts tun kann. Es kann keine anderen Befehle ausführen, bis die Verzögerung vorbei ist. Ein Verzögerungsbefehl veranlasst das Board lediglich, auf die festgelegte Zeit zu warten.

Euer Sketch hat drei Zeilen, die die LEDs einschalten. Fügt nach diesen drei Zeilen einen **delay()**-Befehl hinzu. Stellt die Verzögerungszeit auf eine Sekunde ein. Achtet darauf, am Ende des Befehls ein Semikolon hinzuzufügen.

lIhr müsst auch einen Verzögerungsbefehl am Ende des Sketchs hinzufügen. Ansonsten werden die LEDs, wenn sie erlöschen und der Sketch wieder an den Anfang der Funktion **void loop()** zurückgeleitet wird, wieder eingeschaltet. Fügt einen weiteren **delay()**-Befehl am Ende des Sketchs vor der schließenden Klammer hinzu.

8) Überprüft euren Sketch. Debuggt euren Sketch, falls erforderlich. Euer Sketch sollte ähnlich wie dieser aussehen:

9) Ladet euren Sketch auf das Arduino UNO R3 Board hoch. Wenn ihr fertig seid, sollten eure LEDs im Zwei-Sekunden-Takt blinken.

10) Speichert euren Sketch.

ERFAHREN SIE MEHR: **Pseudocode**

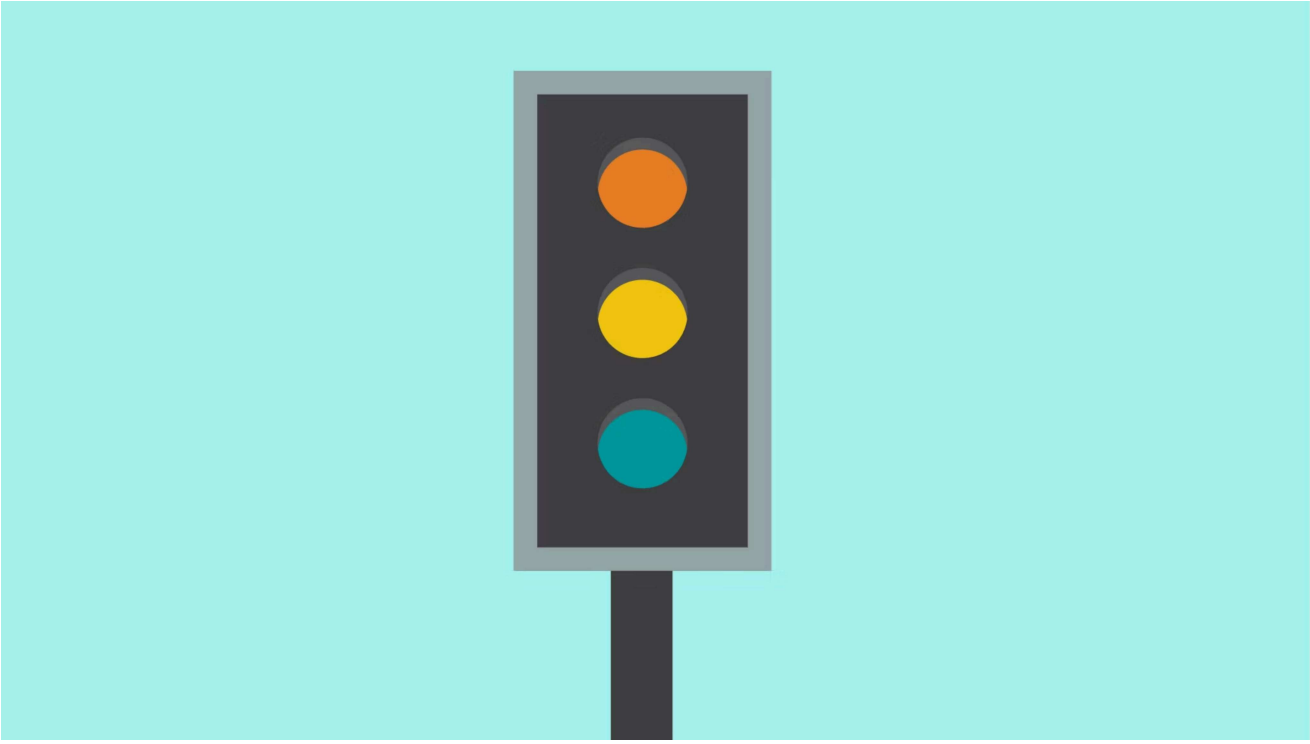
TEACHER NOTES

Erinnern Sie die Schülerinnen und Schüler daran, dass es beim Schreiben von Pseudocode keine richtige oder falsche Antwort gibt. Je detaillierter der Pseudocode jedoch ist, desto einfacher könnte es sein, den Pseudocode in ein funktionierendes Programm zu übertragen.

Codeerstellung - Programmierung einer Ampel

Es ist an der Zeit, eure LEDs zu etwas Nützlichem zu machen. In dieser Aktivität programmiert ihr eure LEDs so, dass sie als Ampel dienen, um den Verkehrsfluss an einer

belebten Kreuzung zu steuern.



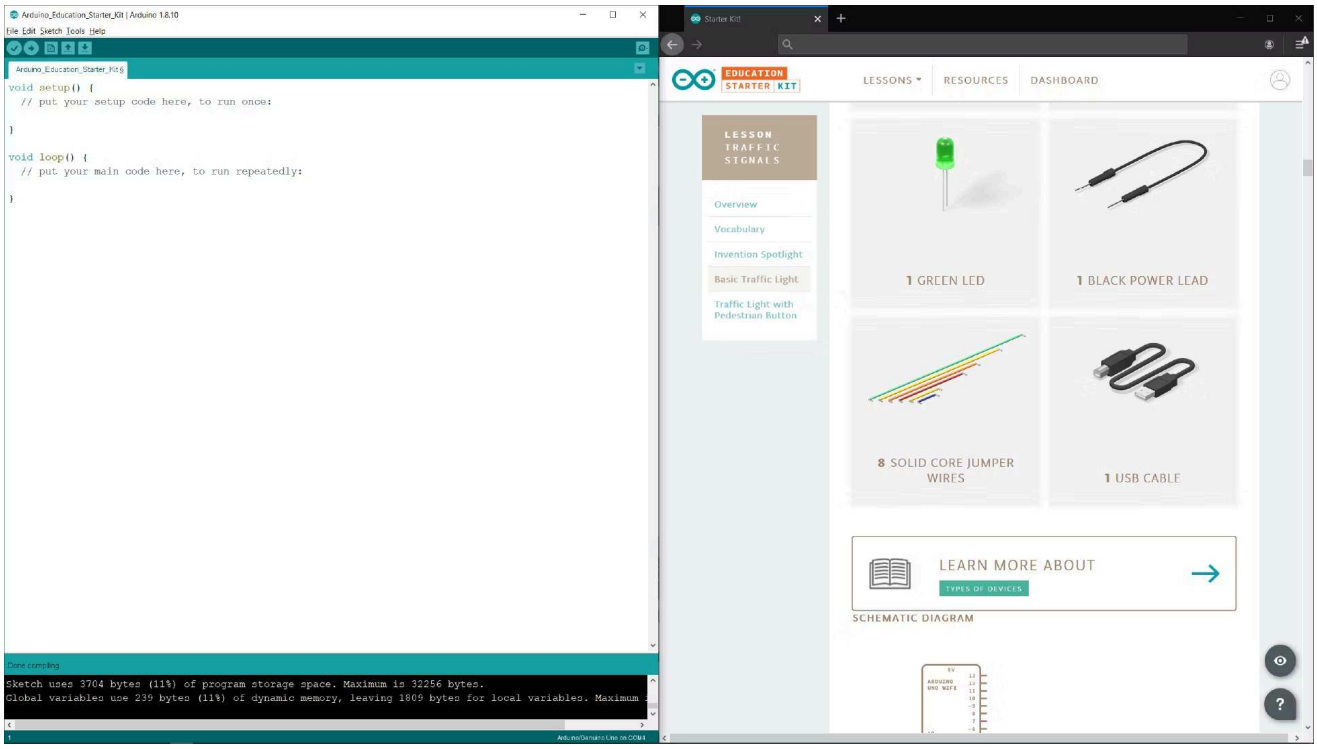
TEACHER NOTES

Bei dieser Aktivität modifizieren die Schülerinnen und Schüler ihr Programm, um mit ihrer LED-Schaltung eine Ampel zu erstellen. Die Aktivität beginnt damit, dass sie Pseudocode für ihr Programm schreiben. Für diese Aktivität benötigen die Schülerinnen und Schüler ihr Arbeitsheft.

1) Falls erforderlich, startet die Arduino IDE und öffnet euren Sketch zu Lektion3_V2. Schließt das Arduino UNO R3 Board mit dem USB-Kabel an den USB-Anschluss des Computers an.

2) Speichert den Sketch unter einem neuen Namen. Wählt im Menü Datei die Option **Speichern unter...** Der Speicherort des Sketch sollte standardmäßig das Arduino Sketchbuch sein. Wenn dies nicht der Fall ist, fragt euren Lehrer, wo ihr den Sketch auf eurem Computer speichern sollt. Benennt die Datei "Lektion3_V3", gefolgt von euren Initialen und denen eures Partners. Klickt auf **Speichern**.

3) Ordnet euer Lektion3_V3-Sketchfenster und dieses Fenster so an, wie es für euch am besten geeignet ist.



4) Bisher habt ihr in diesem Sketch Befehle zum Blinken der drei LEDs geschrieben, die an euer Arduino UNO R3 Board angeschlossen sind. Um diese LEDs so zu modifizieren, dass sie wie eine Ampel funktionieren, ist etwas Nachdenken erforderlich. Nehmt euer Arbeitsheft und schlagt die Seite 9 von Lektion 3 auf.

5) Überlegt euch, was passieren muss, damit eure Schaltung wie eine Ampel funktioniert. Schreibt in den dafür vorgesehenen Platz in eurem Arbeitsbuch einen Pseudocode, der beschreibt, wie eure Ampel funktionieren wird. Verwendet Alltagssprache. Die folgenden Fragen könnten euch beim Schreiben eures Pseudocodes helfen:

- Sollten jemals zwei oder mehr Lichter gleichzeitig eingeschaltet sein?
- Welches Licht sollte als erstes, zweites und drittes aufleuchten?
- Was sollen die anderen Lichter machen, wenn ein Licht aufleuchtet?
- Was sollte nach dem Ausschalten der dritten Lampe geschehen?
- Wie lange sollte jedes Licht eingeschaltet bleiben?

6) Übertragt euren Pseudocode auf euren Sketch aus Lektion 3 in die Arduino IDE. Fangt an, den von euch bereits geschriebenen Code zu modifizieren. Fügt dann nach Bedarf neue **digitalWrite()**- und **delay()**-Befehle hinzu. Hier sind ein paar Hinweise, wenn ihr programmiert:

- Eure **void loop()**-Funktion sollte drei Abschnitte haben. Einen für das grüne Licht, einen für das gelbe Licht und einen für das rote Licht.

- Eine Verzögerung sollte jeden Abschnitt trennen
- Nicht alle Verzögerungen sollten gleich lang sein.
- Aktualisiert eure Codekommentare, damit ihr jede Code-Zeile besser versteht.

7) Wenn ihr mit dem Programmieren fertig sind, überprüft euren Code und debuggt ihn gegebenenfalls.

8) Ladet euren Code auf das Arduino UNO R3 Board hoch. Wenn der Sketch vollständig übertragen ist, wird der Code ausgeführt.

9) Beobachtet das Verhalten eurer Ampel. Verhält sie sich wie erwartet? Wenn nicht, geht zurück zu eurem Sketch und modifiziert euren Code. Ladet den Sketch erneut hoch. Modifiziert euren Code weiter und ladet den Sketch erneut hoch, bis die Ampel sich so verhält, wie ihr es in eurem Pseudocode beschrieben habt.

10) Speichert euren Sketch.

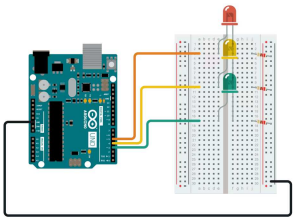
FURTHER NOTES

Die Sketche sollten wie eines dieser Beispiele aussehen. Das zweite Beispiel wurde optimiert, um unnötige Befehle zu entfernen.

Ampel mit Fußgängertaste

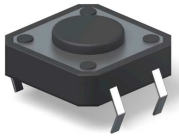
An vielen belebten Kreuzungen haben die Ampeln Fußgängertasten, die die Fußgänger auf dem Gehweg drücken können. Dadurch weiß die Ampel, dass nicht nur Autos, sondern auch Menschen die Straße überqueren müssen. In dieser Aktivität fügt ihr eine Fußgängertaste zu eurer Schaltung hinzu. Dann programmiert ihr die Ampel so, dass sie anders funktioniert, wenn die Taste gedrückt wird.

Benötigte Materialien

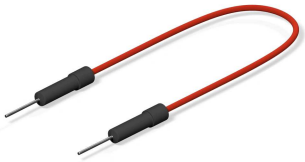


1 ARDUINO PROJEKT BOARD MIT AMPELSCHALTUNG

1 10K Ω WIDERSTAND



1 TASTSCHALTER



1 ROT STROMKABEL

1 STECKVERBINDER

1 USB-KABEL

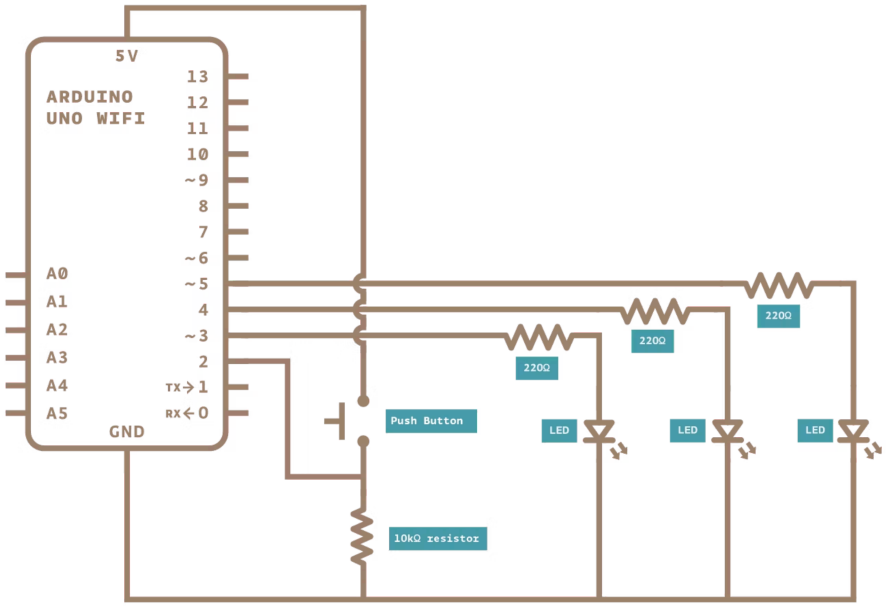
FURTHER NOTES

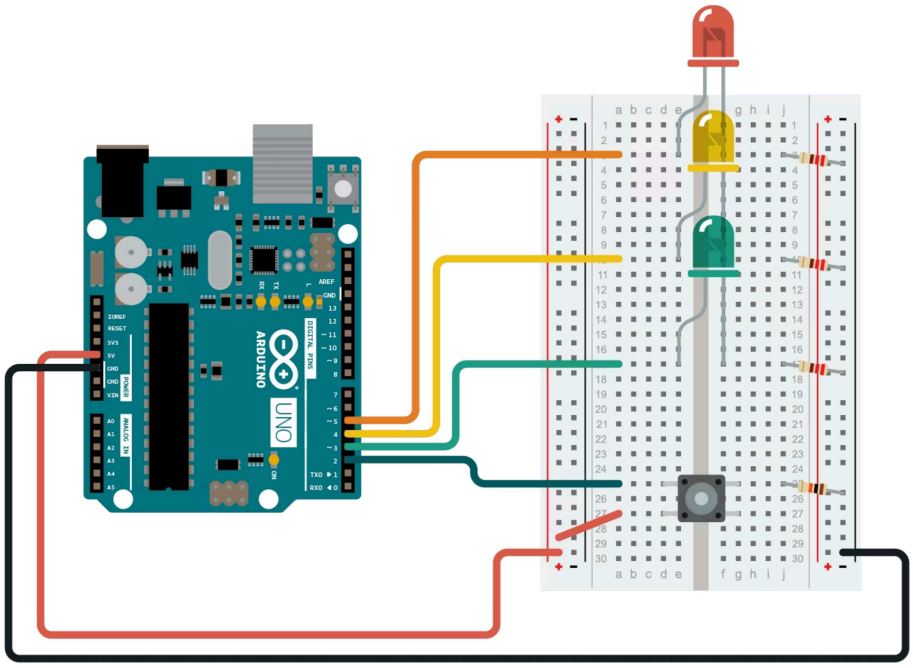
Die Farbbänder des 10K-Ohm-Widerstandes sind:

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

ERFAHREN SIE MEHR: Schalter

Schaltplan



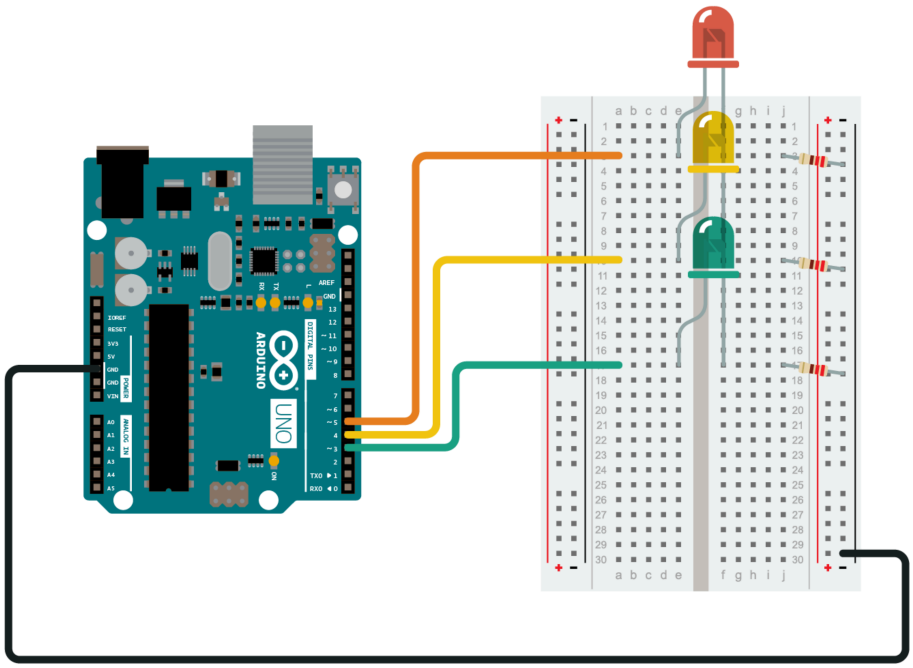


Aufbau der Schaltung

Verwendet den Schaltplan und den Verdrahtungsplan oder die folgende Bilderfolge um die Schaltung aufzubauen.

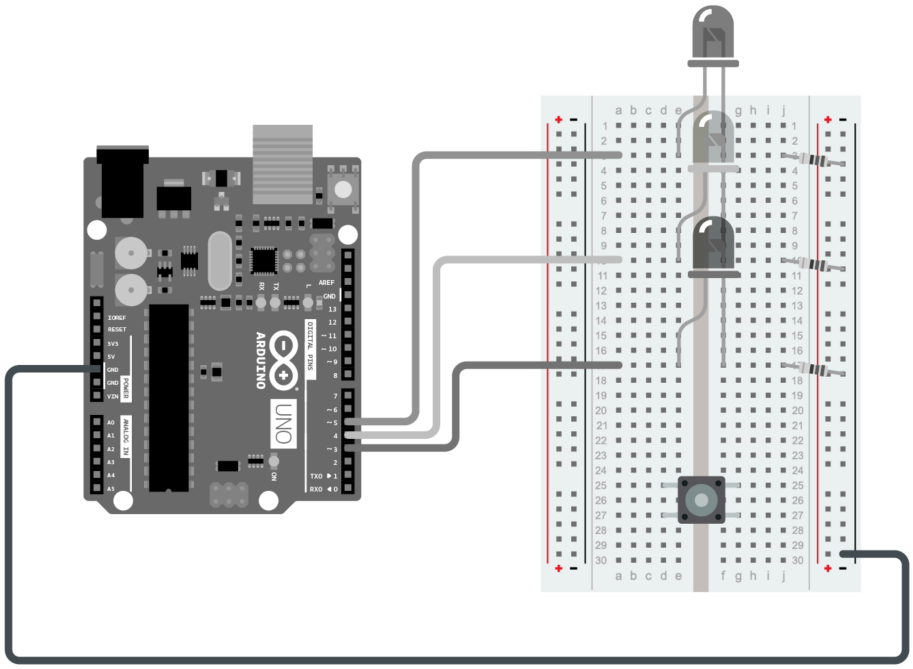
Step 1

Beginnt mit eurer Ampelschaltung aus der vorherigen Aktivität.



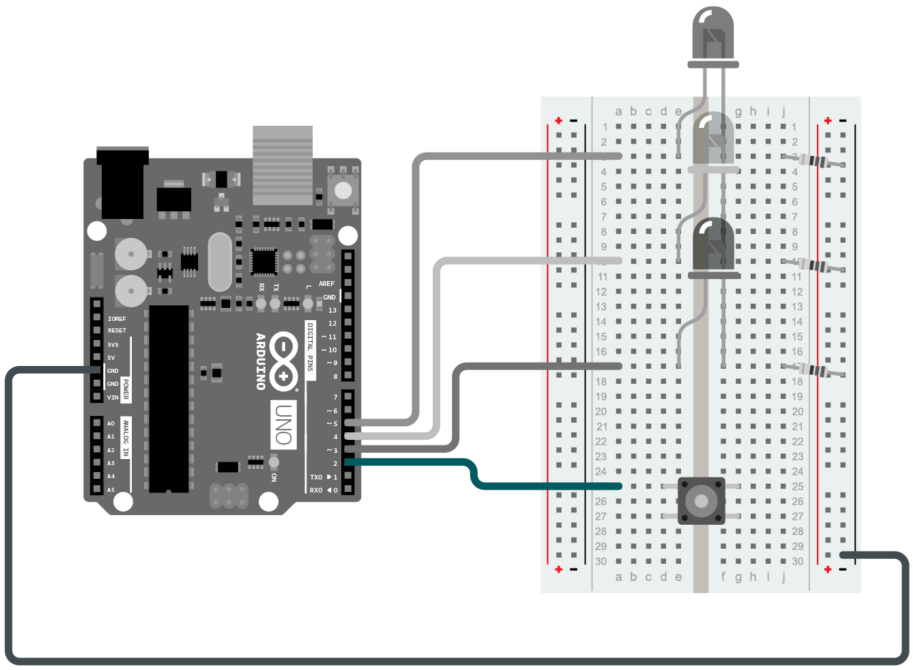
Step 2

Nehmt einen Tastschalter. Befestigt den Tastschalter so am Steckbrett, dass er die Lücke in der Mitte des Steckbretts überbrückt. Die Stifte sollten sich in den Löchern 25e, 25f, 27e und 27f befinden.



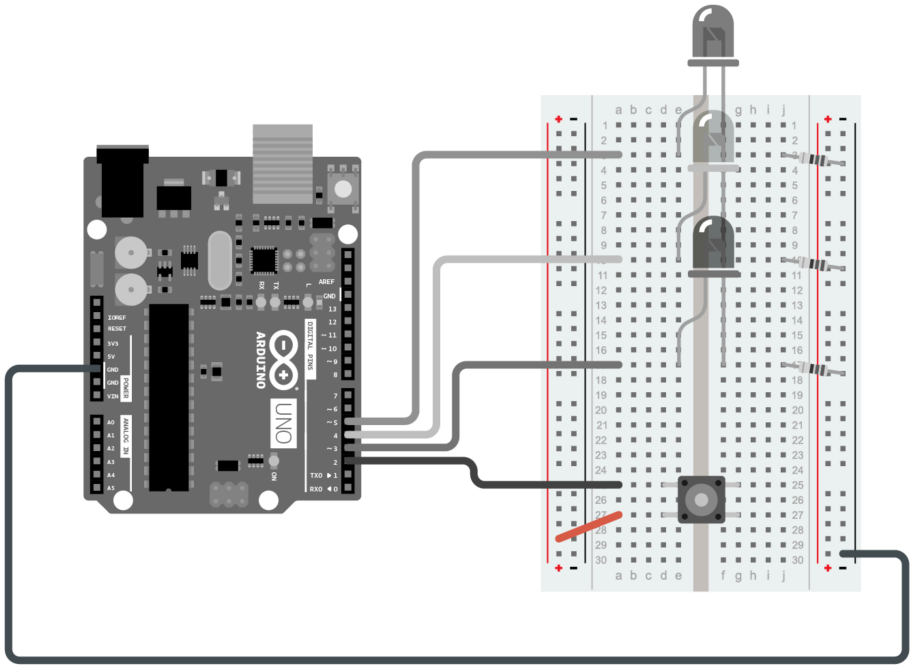
Step 3

Verwendet eine lange Steckverbindung, um den oberen linken Stift des Tastschalters mit dem digitalen Pin 2 auf dem Arduino UNO R3 Board zu verbinden. Führt ein Ende der Steckverbindung in Loch 25a ein. Führt das andere Ende der Steckverbindung in Pin 2 ein.



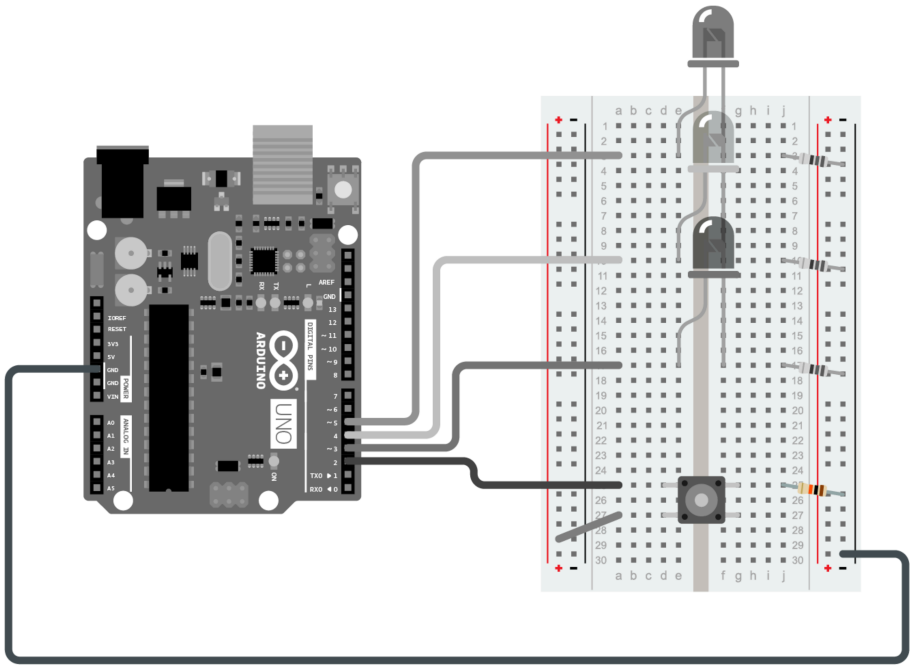
Step 4

Verwendet eine kurze Steckverbindung, um den unteren linken Stift des Tastschalters mit dem positiven Anschluss des Steckbretts zu verbinden. Führt ein Ende der Steckverbindung in Loch 27a ein. Führt das andere Ende in den positiven Anschluss auf der linken Seite des Steckbretts ein.



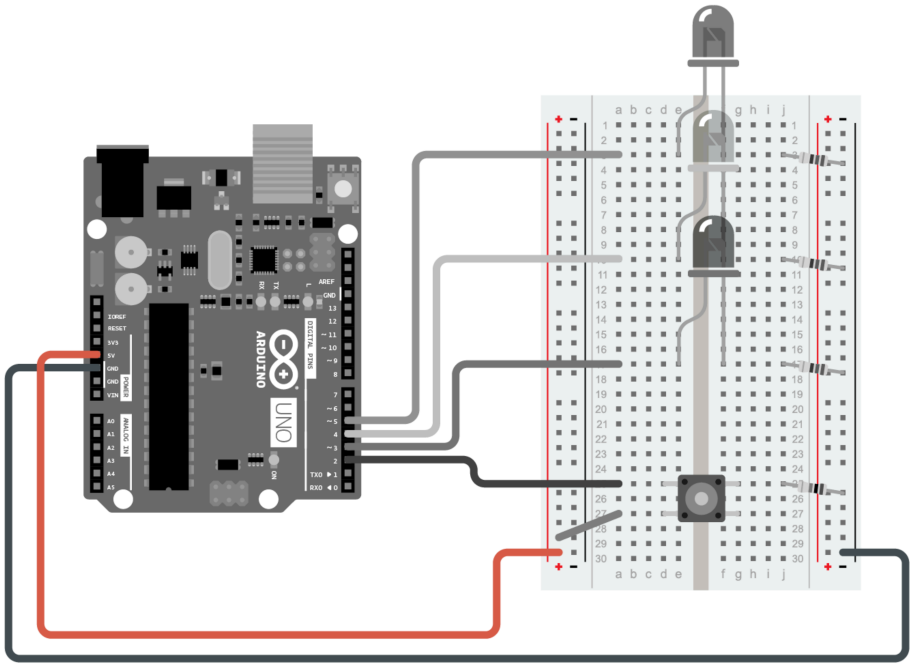
Step 5

Steckt einen 10K-Ohm-Widerstand zwischen den oberen rechten Stift des Tastschalters und Ground an. Steckt ein Ende des Widerstandes in Loch 25j. Steckt das andere Drahtende in die negative Leiste auf der rechten Seite des Steckbretts.



Step 6

Schließlich müsst ihr noch den Tastschalter über die positive Leiste mit Strom versorgen. Verwendet das rote Stromkabel, um den Pluspol auf der linken Seite des Steckbretts mit dem 5V-Pin auf dem Arduino UNO R3 Board zu verbinden.



Ihr wisst bereits, wie der LED-Teil der Schaltung funktioniert. Jede LED dient als Ausgang und wird von einem anderen Pin auf dem Arduino UNO R3 Board gesteuert. Aber der

Tastschalter, der mit Pin 2 auf dem Board verbunden ist, ist anders. Er ist ein Eingabegerät. Pin 2 liest die eingehende Spannung und sendet keine Spannung aus.

Es gibt zwei leitende Pfade, die Strom zu Pin 2 führen können, je nachdem, ob der Taster gedrückt oder losgelassen wird. Wenn der Taster gedrückt ist, liest Pin 2 fünf Volt, wenn Strom durch den Schalter fließt. Wenn der Schalter losgelassen wird, kann kein Strom durch den Schalter fließen; stattdessen geht der leitende Pfad durch den 10K-Ohm-Widerstand, der die Spannung auf Null herunterzieht. Der Abschnitt Pull-Down-Widerstände erläutert die Funktionsweise im Detail.

ERFAHREN SIE MEHR: **Pull-Down-Widerstände**

TEACHER NOTES

Das Verstehen des Tastschalters und der Verwendung des Pull-down-Widerstandes ist eines der am schwersten zu verstehenden Konzepte, die in diesem Kurs behandelt werden. Es ist in Ordnung, wenn die Schülerinnen und Schüler das Konzept zum jetzigen Zeitpunkt noch nicht vollständig verstehen. Es könnte sich zu einem späteren Zeitpunkt in diesem Kurs anbieten, auf dieses Konzept zurückzukommen, wenn die Schülerinnen und Schüler mehr Erfahrung gesammelt haben, damit sie ihr Wissen weiter entwickeln können.

Code-Erstellung - Programmieren des Tastschalters

Bislang habt eure eure Ampel so programmiert, dass die grüne, die gelbe und die rote LED der Reihe nach blinken. Bei dieser Aktivität programmiert ihr eure Fußgängertaste so, dass beim Drücken der Taste die rote und die gelbe LED erlöschen, während die grüne LED blinkt, um anzuzeigen, dass die Fußgänger die Straße überqueren können.

TEACHER NOTES

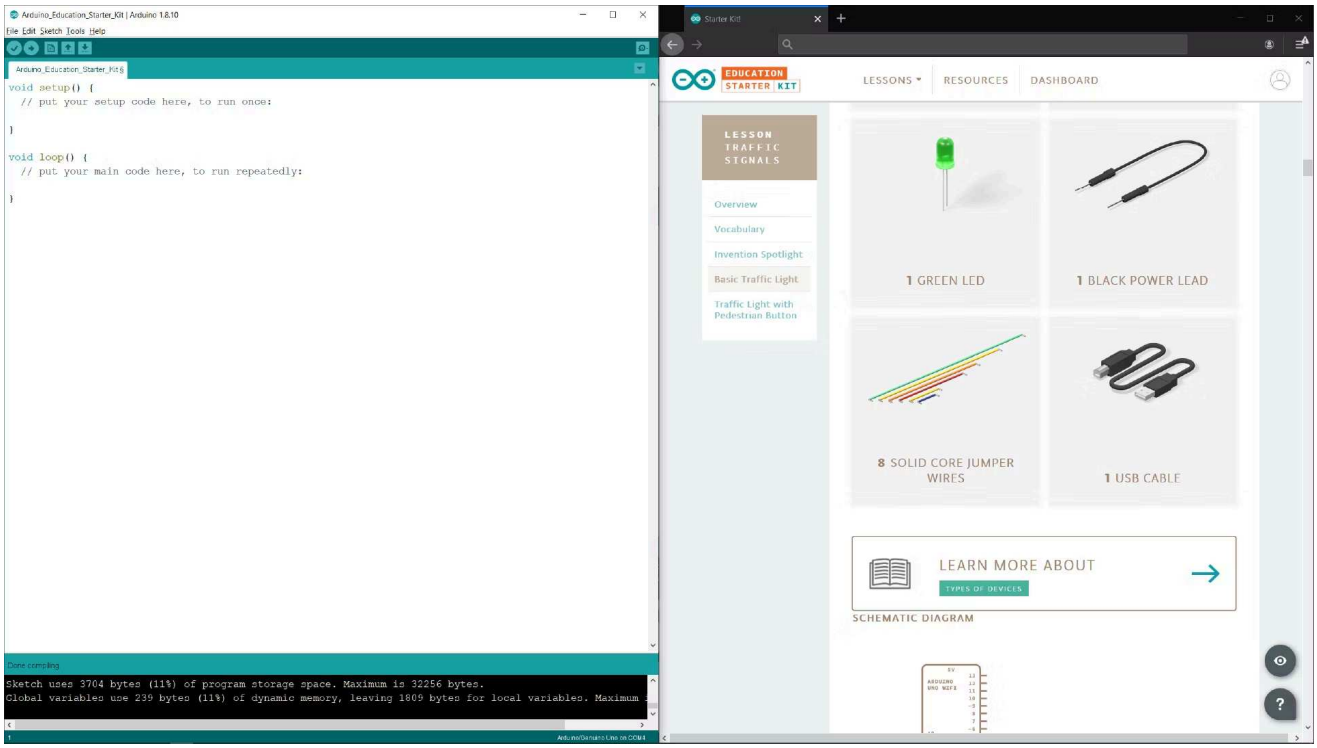
Bei dieser Aktivität programmieren die Schülerinnen und Schüler eine Drucktaste als Input, um das Verhalten der Ampel zu steuern. Sie werden mit folgenden Befehlen vertraut gemacht:

- **if (condition){} –** Grundform einer bedingten Aussage
- **digitalRead() –** Liest einen digitalen Wert aus der Schaltung

1) Falls erforderlich, startet die Arduino IDE und öffnet euren Sketch zu Lektion3_V3. Schließt das Arduino UNO R3 Board mit dem USB-Kabel an den USB-Anschluss des Computers an.

2) Speichert den Sketch unter einem neuen Namen. Wählt im Menü Datei die Option **Speichern unter...** Der Speicherort des Sketch sollte standardmäßig das Arduino Sketchbuch sein. Wenn dies nicht der Fall ist, fragt euren Lehrer, wo ihr den Sketch auf eurem Computer speichern sollt. Benennt die Datei "Lektion3_V4_", gefolgt von euren Initialen und denen eures Partners. Klickt auf **Speichern**.

3) Ordnet euer Lektion3_V4-Sketchfenster und dieses Fenster so an, wie es für euch am besten geeignet ist



4) Euer Arduino UNO R3 Board ist so eingerichtet, dass es den digitalen Pin 2 als Input verwendet. Fügt in der Funktion **void setup()** einen weiteren Befehl **pinMode()** hinzu, der Pin 2 als Input deklariert. Fügt dann einen Codekommentar hinzu, der neuen Befehl erklärt.

Hinweis: Achtet darauf, das Wort **INPUT** ganz in Großbuchstaben einzugeben und sowohl eine öffnende als auch eine schließende Klammer zu haben sowie am Ende des Befehls ein Semikolon einzufügen.

Bedingte Anweisungen

Es gibt zwei verschiedene Verhaltensweisen, die die Ampel anzeigen sollte, je nachdem, ob die Taste gedrückt wird oder nicht. Wenn die Taste gedrückt ist, sollte der Fußgängerübergangsteil des Codes laufen und die grüne LED blinken. Wenn die Taste nicht gedrückt ist, sollte die Ampel normal funktionieren, so wie ihr sie bereits programmiert habt. Um beide Verhaltensweisen zu programmieren, müsst ihr eine neue Code Funktion, eine so genannte bedingte Anweisung, verwenden.

Eine **bedingte Anweisung** wird manchmal als Wenn-dann-Anweisung oder einfach nur als Wenn-Anweisung bezeichnet. Bedingte Anweisungen ermöglichen es euch, bestimmte Codezeilen auszuführen, wenn eine bestimmte Situation oder Bedingung wahr ist. ihr verwendet ständig Bedingungen, um im täglichen Leben Entscheidungen zu treffen. Hier sind einige grundlegende Beispiele:

In der Arduino IDE sehen bedingte Aussagen wie folgt aus:

```
if (Bedingung) {

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

Die Bedingung steht innerhalb der Klammern und verwendet einen **Vergleichsoperator**, um zwei oder mehr Werte zu vergleichen. Bei diesen Werten kann es sich um numerische Werte, Variablen oder Eingaben handeln, die auf das Arduino UNO R3 Board kommen. Die folgende Tabelle ist eine Liste der Vergleichsoperatoren und wie sie im Bedingungsteil einer if-Anweisung verwendet werden.

Vergleichsoperator	Bedeutung	Beispiel
==	ist gleich	if (digitalRead(2) == HIGH) {Etwas t
!=	ungleich	if (digitalRead(5) != LOW) {Etwas tu
<	kleiner als	if (distance < 100) {Etwas tun}
>	größer als	if (count > 5) {Etwas tun}
<=	kleiner oder gleich	if (number) <= minValue) {Etwas tu
>=	größer oder gleich	if (number >= maxValue) {Etwas tu

Hinweis: Beachtet, dass der Vergleich "ist gleich" ein doppeltes Gleichheitszeichen ist. Ein einzelnes Gleichheitszeichen wird verwendet, um einer Variablen einen Wert zuzuweisen. Ein doppeltes Gleichheitszeichen wird verwendet, um zwei Werte zu vergleichen. Mehr über Variablen und die Zuweisung von Werten erfahrt ihr in Lektion 4.

Wenn die beiden Werte in der Bedingung verglichen werden, kann das Ergebnis entweder wahr oder falsch sein. Wenn die Bedingung wahr ist, dann sind die Befehle innerhalb der Klammern abgeschlossen. Wenn die Bedingung falsch ist, dann werden die Befehle innerhalb der geschweiften Klammern übersprungen.

Hinweis: Denkt daran, dass jede Funktion in der Arduino IDE, die eine öffnende Klammer hat, eine schließende Klammer haben muss. Wenn ihr mit bedingten Anweisungen arbeitet, kann es verwirrend sein zu verstehen, welche geschweiften Klammern zu welcher Funktion gehören. Wenn ihr in der Arduino IDE den Cursor direkt hinter eine öffnende oder schließende geschweifte Klammer bewegt, wird das Paar für diese Klammer mit einem Kästchen hervorgehoben.

Beim Programmieren können bedingte Aussagen recht einfach sein, oder sie können komplexe logische Argumente sein, die mehrere Bedingungen und Szenarien beinhalten. Weitere Arten von bedingten Anweisungen werden in Lektion 4 behandelt. Verwendet für

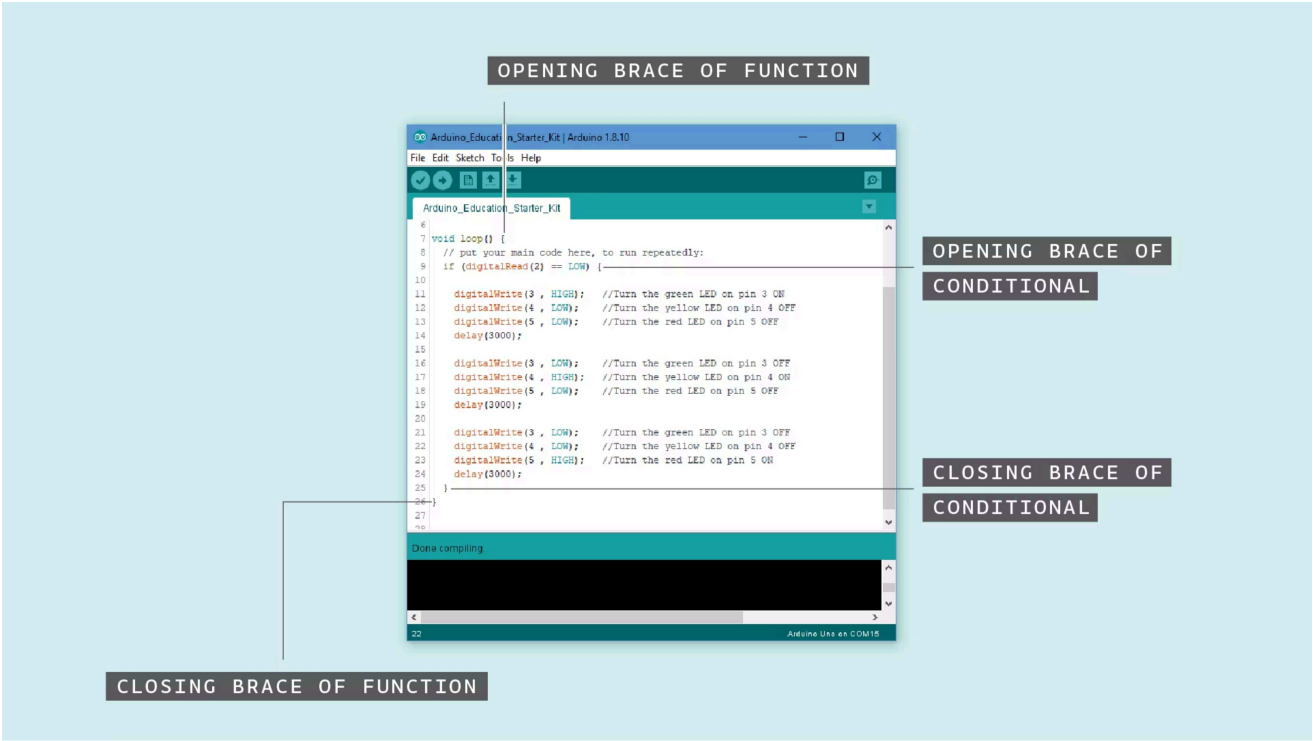
den Anfang die Grundform der if-Anweisung. **5)** Gebt am Anfang der **void loop()**-Funktion die folgende Anweisung ein:

Erinnert euch, dass der Befehl **digitalWrite(pin, value)** verwendet wird, um einen Output Pin entweder auf HIGH oder LOW zu setzen. Dies sind die Befehle, mit denen die LEDs ein- oder ausgeschaltet werden. So wie der Befehl **digitalWrite()** bei Output Pins verwendet wird, wird der Befehl **digitalRead()** bei Input Pins verwendet. Der Befehl **digitalRead(pin)** ist der Befehl, der aussagt, ob ein digitaler Pin einen HIGH- oder LOW-Wert hat.

Erklärung pinMode	Command	Value	Voltage
pinMode(pin,INPUT)	digitalRead(pin)	Gibt HIGH zurück	Liest mehr als 3 Volt
		Gibt LOW zurück	Liest weniger als 1,5 Volt
pinMode(pin,OUTPUT)	digtalWrite(pin, value)	Outputs HIGH	sendet 5 volt
		Outputs LOW	sendet 0 volt

Die soeben erstellte if-Anweisung vergleicht den Status von Pin 2 mit einem Wert von LOW. Erinnert euch daran, dass der LOW-Spannungswert auftritt, wenn die Taste nicht gedrückt ist und der leitende Pfad durch den 10K-Ohm-Widerstand verläuft. Mit anderen Worten, diese bedingte Anweisung besagt, "wenn die Taste nicht gedrückt ist".

6) Als nächstes müsst ihr die Befehle hinzufügen, die ausgeführt werden sollen, wenn Pin 2 LOW ist und die Taste nicht gedrückt wird. Dies sind die Befehle, die ihr bereits erstellt habt und die eure normale Ampel laufen lassen. Ihr könnt diese Befehle entweder ausschneiden und zwischen den geschweiften Klammern für die if-Anweisung einfügen, oder ihr könnt die schließende geschweifte Klammer für die if-Anweisung nach eurer letzten Verzögerung einfach verschieben. Verwendet die Methode, die für euch am besten geeignet ist. Wenn ihr fertig seid, sollte eure **void loop()**-Funktion ähnlich wie diese aussehen:



Hinweis: Beachtet im Code, wie die Befehle für die if-Anweisung eingerückt sind. Die Verwendung von Einrückungen hilft, den Code übersichtlich zu halten und macht deutlich, welche Befehle in einer Funktion ausgeführt werden. Auch wenn es ein paar Sekunden länger dauern kann, verwendet Einrückungen, Zeilenumbrüche und Codekommentare, damit euer Code schön aussieht. Es wird sich auf lange Sicht lohnen.

7) Überprüft euren Code und debugged, falls erforderlich.

FURTHER NOTES

Ein häufiger Syntaxfehler ist es, die Anzahl der benötigten Klammern aus den Augen zu verlieren. Manchmal werden schließende geschweifte Klammern bei Funktionen weggelassen oder es werden zu viele schließende Klammern hinzugefügt. Erinnern Sie die Schülerinnen und Schüler daran, dass in einem Sketch jede öffnende Klammer eine schließende Klammer benötigt.

8) Ihr habt den Teil der Schaltung programmiert, der die Ampel betätigt, wenn die Taste nicht gedrückt wird. Jetzt ist es an der Zeit, den Code zu schreiben, der es einem Fußgänger erlaubt, die Straße zu überqueren, wenn die Taste gedrückt wird. Dazu ist eine zweite bedingte Anweisung erforderlich. Diesmal müsst ihr jedoch den **digitalRead()**-Wert von Pin 2 mit HIGH anstelle von LOW vergleichen. Gebt die folgende bedingte Anweisung am Ende eures Sketchs vor der schließenden Klammer für die Funktion **void loop()** ein.

9) Wenn die Taste gedrückt wird, muss die Ampel den gesamten Verkehr stoppen und den Fußgängern anzeigen, dass es OK ist, die Straße zu überqueren. Zu diesem Zweck schaltet die rote und die gelbe LED aus und lasst die grüne LED blinken. Fügt innerhalb der Klammern eurer zweiten bedingten Anweisung drei **digitalWrite()**-Befehle hinzu:

- Ein Befehl sollte die an Pin 5 angeschlossene rote LED ausschalten.
- Ein Befehl sollte die an Pin 4 angeschlossene gelbe LED ausschalten.
- Ein Befehl sollte die grüne LED, die an Pin 3 angeschlossen ist, einschalten.

10) Eure letzte Aufgabe besteht darin, die grüne LED blinken zu lassen, während die Taste gedrückt wird. Denkt daran, dass die Blinkfrequenz durch eure **delay()**-Anweisungen bestimmt wird. Ihr könnt die Blinkfrequenz für die grüne LED wählen. Fügt vor der schließenden Klammer eurer zweiten if-Anweisung eine Verzögerungsanweisung gefolgt von einer **digitalWrite()**-Anweisung hinzu, um die grüne LED auszuschalten, und fügt dann eine abschließende Verzögerungsanweisung hinzu.

11) Fügt Codekommentare zu euren neuen Befehlen hinzu, die erklären, was jeder einzelne Befehl bewirkt.

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

13) Ladet euren Code auf das Arduino UNO R3 Board hoch. Wenn der Sketch vollständig übertragen ist, wird der Code ausgeführt.

14) Beobachtet das Verhalten eurer Ampel. Drückt die Taste nach unten und wartet, bis die Ampel ihren Zyklus beendet hat. Blinkt das grüne Licht für Fußgänger? Kehrt die Ampel zu ihrem normalen Ampelmuster zurück, wenn die Taste losgelassen wird? Falls nicht, verändert euren Sketch und ladet euren Sketch erneut auf das Arduino UNO R3 Board hoch.

15) Wenn ihr fertig seid, speichert euren Sketch.

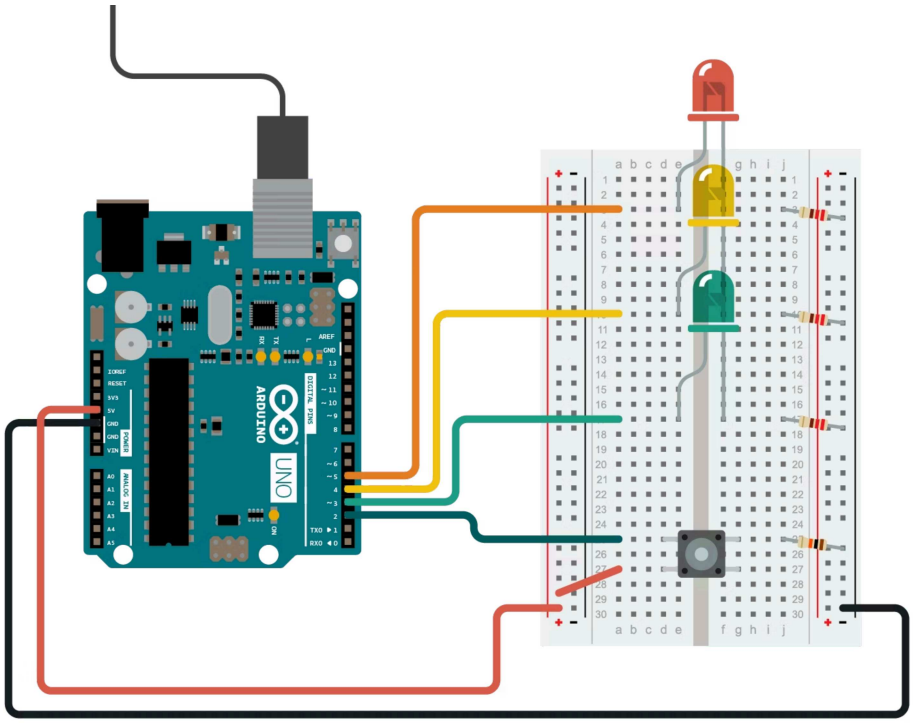
Testen und ändern

In Lektion 2 habt ihr etwas über Serien- und Parallelschaltungen gelernt. Die meisten Schaltungen sind in der Regel nicht 100% in Reihe oder 100% parallel. Die meisten Schaltungen sind eine Kombination aus Serien- und Parallelschaltungen. Tatsächlich enthält die Ampel, die ihr gebaut habt, Elemente sowohl von Serien- als auch von Parallelschaltungen. In dieser Aktivität werdet ihr Teile eurer Schaltung identifizieren, die in Serie und parallel verdrahtet sind. Ihr werdet auch das Multimeter verwenden, um zu untersuchen, wie der Taster die Spannung steuert, die in das Arduino Board gegeben wird. Für diese Aktivität benötigen ihr euer Arbeitsheft.

TEACHER NOTES

Die Schülerinnen und Schüler brauchen für diese Übung ihr Arbeitsheft. Einen Antwortschlüssel für Lektion 3 finden Sie auf Seite 10 im Lehrer-Arbeitsheft.

1) Falls erforderlich, startet die Arduino IDE und öffnet euren Sketch zu Lektion3. Schließt das Arduino UNO R3 Board mit dem USB-Kabel an den USB-Anschluss des Computers an. Stellt sicher, dass die Ampelschaltung wie erwartet funktioniert.

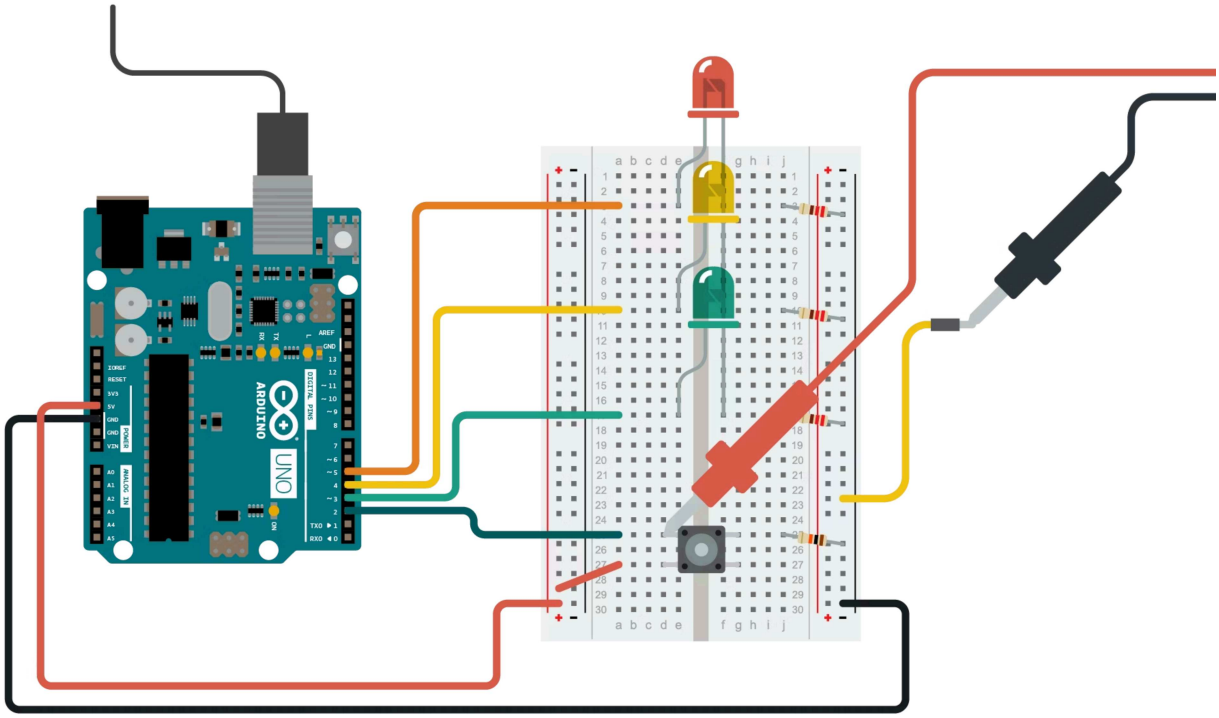


2) Eure Ampel ist eine Kombination aus Reihen- und Parallelschaltungen. Erklärt, welche Komponenten in Reihe geschaltet sind und warum. Erklärt dann, welche Komponenten parallel verdrahtet sind und warum. Notiert eure Antwort in eurem Arbeitsheft.

3) Messt und vergleicht als nächstes die Spannung von Pin 2, wenn der Taster gedrückt oder nicht gedrückt wird. Auf diese Weise könnt ihr sehen, was das Arduino UNO R3 Board sieht. Nehmt euer Multimeter und dreht den Drehknopf auf die Einstellung 20 Volt DC. (20V).

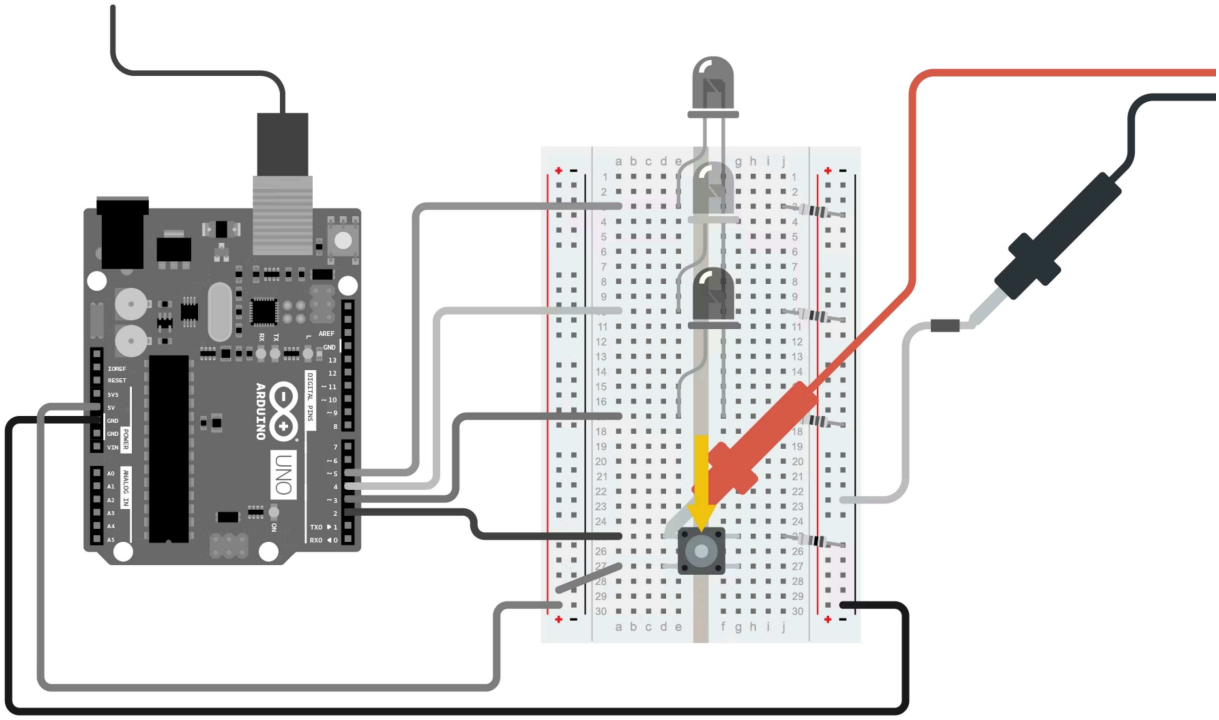
4) Erinnert euch, dass Spannung eine Differenz der potentiellen Energie zwischen zwei Teilen der Schaltung ist. Die Pins auf dem Arduino UNO R3 Board messen immer diese Differenz im Vergleich zum Ground. Um die Spannung von Pin 2 zu ermitteln, müsst ihr das Multimeter mit Ground verbinden. Bringt eine Steckverbindung am Minuspol des Steckbretts an.

5) Wenn sich der Taster in der oberen Position befindet, messt ihr die Spannung zwischen dem Taster und Ground. Berührt das rote Prüfkabel an einem der oberen Stifte des Tastschalters. Berührt das schwarze Prüfkabel mit dem anderen Ende des Steckverbindung, der mit Ground verbunden ist. Zeichnet eure Spannungsmessung als Pin-2-Spannung auf, wenn der Tastschalter oben ist.



6) Ein digitaler Pin hat zwei Zustände, entweder HIGH oder LOW. Ein Pin befindet sich im Zustand HIGH, wenn er mehr als 3 Volt Strom erhält. Ein Pin befindet sich im LOW-Zustand, wenn er weniger als 1,5 Volt empfängt. Tragt in euer Arbeitsheft ein, ob sich der Pin im HIGH oder LOW Zustand befindet, wenn der Tastschalter oben ist.

7) Drückt den Tastschalter, um die Ampel in den Fußgängermodus zu versetzen. Messt bei gedrücktem Tastschalter die Spannung zwischen der Taste und dem Ground. Berührt das rote Prüfkabel an einem der oberen Stifte des Tastschalters. Berührt das schwarze Prüfkabel mit dem anderen Ende der Steckverbindung, der mit Ground verbunden ist. Notiert eure Spannungsmessung in der Spalte "Pin 2 Spannung", wenn der Tastschalter gedrückt ist.



TEACHER NOTES

Schülerinnen und Schüler, die allein arbeiten, werden mit diesem Teil der Aktivität Schwierigkeiten haben.

FURTHER NOTES

Es werden mehr als zwei Hände benötigt, um den Tastschalter zu drücken und gleichzeitig die Spannung zu messen. Seien Sie darauf vorbereitet, den Schülerinnen und Schülern bei Bedarf zu helfen.

8) Notiert in eurem Arbeitsheft, ob der Pin bei gedrückter Taste HIGH oder LOW ist.

9) Seht euch die Daten an, die ihr gerade gesammelt habt. Verwendet diese Daten, um zu erklären, wie die Drucktastenschaltung auf Pin 2 dem Arduino UNO R3 Board entweder als HIGH oder LOW gelesen wird. Fügt unbedingt eine Erklärung hinzu, warum der 10K-Ohm-Widerstand benötigt wird.

10) Räumt euren Arbeitsbereich auf und lagert alle Geräte in einem geeigneten Lagerbereich.

FURTHER NOTES

WEITERE VORSCHLÄGE

- **Diskussion –** Viele moderne Ampeln sind mit Kameras und Sensoren ausgestattet, um festzustellen, wann das Licht wechseln sollte. Führen Sie eine

Klassendiskussion darüber, wie der Sensoreingang in der Ampelschaltung der Schülerinnen und Schüler verwendet werden könnte. Wo würden die Sensoren an den Stromkreis angeschlossen? Wonach würden sie suchen? Unter welchen Bedingungen sollten sich die Ampeln ändern?

- **Verbesserungen an der Ampel –** Lassen Sie die Schülerinnen und Schüler Fehler im Design ihrer Ampel erkennen. Lassen Sie sie dann nach Ideen und Lösungen suchen, um diese Mängel zu beheben. Lassen Sie sie ihre Schaltung oder ihren Code modifizieren, um ihre Lösungen zu implementieren und die Ampel zu verbessern. Ein Fehler in der Schaltung besteht zum Beispiel darin, dass die Fußgängertaste gedrückt bleiben muss, damit die grüne LED blinkt, aber die Fußgänger nicht weiterhin die Taste drücken und gleichzeitig die Straße überqueren können. Wie könnte der Code so geändert werden, dass der Tastendruck gelesen wird, aber auch die Fußgänger die Straße überqueren können? Einige der Lösungen zur Behebung der Mängel an der Ampel werden in Lektion 4 behandelt.
- **Mehrere Ampeln –** An vielen größeren Kreuzungen sind mehrere Ampeln gleichzeitig in Betrieb, so dass die Fahrer eine bessere Sicht haben und wissen, wann sie anhalten und wann sie weiterfahren müssen. Lassen Sie die Schülerinnen und Schüler eine weitere rote, gelbe und grüne LED parallel zu den ursprünglichen LEDs anbringen. Beide roten LEDs sollten gleichzeitig aufleuchten, ebenso wie die beiden gelben LEDs und die beiden grünen LEDs. Lassen Sie die Schülerinnen und Schüler feststellen, was mit der Schaltung passiert, wenn eine zweite Ampel in die Schaltung eingefügt wird.
- **Code Optimierung –** Die Optimierung des Codes ist eine wichtige Fähigkeit beim Programmieren. Sie beschleunigt die Code-Laufzeit und macht den Code effizienter. Lassen Sie die Schülerinnen und Schüler ihren letzten Sketch aus Lektion 3 in der Arduino IDE betrachten. Sie sollten nach Möglichkeiten suchen, ihren Code zu optimieren, indem sie unnötige Befehle entfernen. Wenn zum Beispiel eine LED bereits aus ist, brauchen sie keinen zweiten Befehl, der die LED zum Ausschalten bringt. Lassen sie sie ihren optimierten Code auf ihrer Ampelschaltung testen, um zu bestätigen, dass sie immer noch wie erwartet funktioniert.
- **Pseudocode –** Lassen Sie die Schülerinnen und Schüler Pseudocode für einen Algorithmus schreiben. Lassen Sie sie dann ihren Pseudocode vergleichen und diskutieren. Stellen Sie Fragen wie: "Ist der Code bei allen gleich? Wie viele Möglichkeiten gibt es, eine Aufgabe zu erledigen? Was funktioniert besser: Wenn man mit allgemeinen Anweisungen oder mit detaillierten Anweisungen arbeitet?"

TEACHER NOTES

Mögliche Beispiele für Algorithmen:

- Wie man von zu Hause zur Schule kommt.
- Wie man ein Brot belegt.
- Wie man die Spannung einer Batterie überprüft.