LICHTWELLENRADAR

In dieser Lektion verwendet ihr euer Arduino Board und ein Gerät namens Fototransistor, um elektromagnetische Wellen - insbesondere sichtbares Licht - zu untersuchen. Ihr werdet einige der Anwendungen zur Erkennung und Messung von Licht wie Kommunikations- und Radartechnologien untersuchen.

VORAUSSICHTLICHE ZEIT **90-135 min**

SIE WERDEN ETWAS DARÜBER LERNEN

FOTOTRANSISTOREN, ÜBERTRAGUNGSSYSTEMEN , EMPFANG, FUNKTIONEN , SERIELLEN MONITOR ZU VERWENDEN, ELEKTROMAGNETISCHE ENERGIE

Übersicht

In Lektion 8 habt ihr das Arduino UNO R3 Board benutzt, um einige der Eigenschaften von Schall zu erforschen und wie sich Schall in Wellen ausbreitet. Durch Wellen kann Energie von Ort zu Ort übertragen werden. Aber nicht alle Wellen sind wie Schallwellen. In dieser Lektion verwendet ihr euer Arduino UNO R3 Board und ein Gerät namens Fototransistor, um elektromagnetische Wellen - insbesondere sichtbares Licht - zu untersuchen. Ihr werdet einige der Anwendungen zur Erkennung und Messung von Licht wie Kommunikations- und Radartechnologien untersuchen.

TEACHER NOTES

In dieser Lektion verwenden die Schülerinnen und Schüler einen Fototransistor als Sensor, um die Lichtintensität zu messen. Die Lektion beginnt damit, dass die Schülerinnen und Schüler einen grundlegenden analogen Lichtsensorschaltkreis aufbauen. Sie verwenden den Sensor zur Messung des Umgebungslichts im Raum und untersuchen, wie der vom Arduino UNO R3 Board ausgegebene Analogwert mit der Helligkeit des Lichts im Raum zusammenhängt. Anschließend untersuchen die Schülerinnen und Schüler, wie das Licht in Glasfaseroptik verwendet wird, um große Mengen an

Informationen mit hoher Geschwindigkeit zu übertragen. Obwohl die Schülerinnen und Schüler bei dieser Übung kein Glasfaserkabel verwenden, können sie dennoch das Grundprinzip erkennen, wie ein Fotosender Informationen über Lichtwellen an einen Fotoempfänger sendet, der die Informationen erfasst und interpretiert.

Später in der Lektion benutzen die Schülerinnen und Schüler ihren Lichtsensor als Radar, der die Lichtintensität des Raumes abbildet. Sie schließen ihren Fototransistor an einen Servo an, der manuell über Befehle gesteuert wird, die über den seriellen Monitor eingegeben werden. Dann kodieren die Schülerinnen und Schüler das Radar so, dass es den Raum automatisch durchsucht und dabei Daten sowohl an den seriellen Monitor als auch an den seriellen Plotter ausgibt.

ZEITAUFWAND FÜR DIE LEKTION

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

Übersicht und Fachbegriffe	2 minuten
Lichtsensor	
Benötigte Materialien	1 minute
Fototransistoren	4 minuten
Aufbau der Schaltung	4 minuten
Code-Erstellung - Licht messen	10 minuten
Code-Erstellung - Lichtwellenkommunikation	13 minutesn
Serielle Eingänge	5 minuten
Blickpunkt Erfindungen	4 minuten
Lichtwellenradar	
Benötigte Materialien	1 minute
Aufbau der Schaltung	3 minuten
Code-Erstellung - Manueller Lichttaster	20 minuten
Lichtintensitätsexperiment	8 minuten
Testen und ändern	

Ändern des Codes - Lichtwellenradar	15 minuten
Gesamte Lektion	**90 minuten**

Wenn Sie diese Lektion in zwei Unterrichtsabschnitten abschließen, sollten die Schülerinnen und Schüler am Ende des ersten Unterrichtsabschnitts mit dem Teil "Aufbau der Schaltung" des Abschnitts "Lichtwellenradar" beginnen.

LERNZIELE

- Bestimmen, wie ein Fototransistor Licht detektiert.
- Eingabedaten von einem analogen Lichtsensor interpretieren.
- Aufgerufene Funktionen erstellen und verwenden, um sich wiederholende Befehle im Code auszuführen.
- Anwendungen des Sendens und Empfangens wie Faseroptik und Radar erforschen.
- Einen Schaltkreis durch Eingabe von Informationen über den seriellen Monitor steuern.
- Die Eigenschaften von Licht und anderen Wellenarten im elektromagnetischen Spektrum untersuchen.
- Ein Diagramm erstellen und analysieren, um signifikante Datenpunkte zu bestimmen.

Fachbegriffe

- **ASCII-Code -** ein Code, der allen Groß- und Kleinbuchstaben des englischen Alphabets, den Zahlen 0 bis 9 und einigen Sonderzeichen eine Zahl zuordnet, die als Datenbyte dargestellt werden kann
- **Puffer -** ein Datenspeicherort, der Daten für eine kurze Zeit speichert, während sie von einem Ort zum anderen bewegt werden
- **Aufgerufene Funktion -** vom Hauptprogramm getrennte Codezeilen, die eine bestimmte Aufgabe ausführen; aufgerufene Funktionen können jederzeit innerhalb des Hauptprogramms ausgeführt werden, wann immer diese Aufgabe erledigt werden muss
- **elektromagnetisches Spektrum -** der Bereich all der verschiedenen Arten von elektromagnetischen Wellen (einschließlich Licht), die sich durch ihre Wellenlängen unterscheiden
- **elektromagnetische Welle -** eine Welle, die Energie durch die Veränderung elektrischer und magnetischer Felder überträgt; elektromagnetische Wellen brauchen kein Medium, durch das sie sich bewegen, und haben einen großen Wellenlängenbereich
- **Faseroptik -** die Verwendung von Licht zur Übertragung von Informationen durch dünne Fasern oder Kabel aus Glas oder Kunststoff
- **Fototransistor -** ein elektronisches Bauelement, das Lichtenergie in elektrische Energie umwandelt
- **Transversalwelle -** eine Welle, die im rechten Winkel zur Bewegungsrichtung schwingt oder vibriert
- **Wellenlänge -** der Abstand zwischen zwei aufeinander folgenden Wellenscheitelpunkten; die Wellenlänge bestimmt die Eigenschaften einer elektromagnetischen Welle

TEACHER NOTES

Es gibt für die Schülerinnen und Schüler viele Übungsmöglichkeiten zu den Fachbegriffen. 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.

Lichtsensor

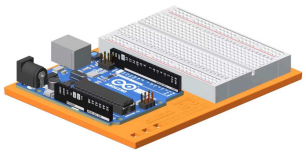
Einer der heute vielleicht am häufigsten verwendeten Sensoren ist ein Lichtsensor. Mobiltelefone haben Lichtsensoren, die von der Kamera verwendet werden, um zu wissen, ob der Blitz beim Fotografieren eingeschaltet werden soll. Straßen-, Auto- und Landschaftsbeleuchtungen haben Sensoren, die in der Dämmerung das Licht einschalten. Sicherheitssysteme verwenden oft Lichtsensoren, um die Sicherheitseinstellungen bei Nacht zu ändern. Natürlich gibt es mehrere verschiedene Arten von Lichtsensoren. Einige erkennen Helligkeit, während andere Farbe erkennen. Verschiedene Arten von Sensoren erkennen auch verschiedene Arten von Licht, wie Ultraviolett-, Infrarot- und Röntgenstrahlen.

In dieser Übung konstruiert ihr einen einfachen analogen Lichtsensor, der die Intensität bzw. Helligkeit des sichtbaren Lichts im Raum misst. Ihr werdet auch untersuchen, wie eine ähnliche Technologie für die Kommunikation und die Übertragung von Informationen über große Entfernungen eingesetzt wird.

TEACHER NOTES

Diese Übung konzentriert sich auf die Verwendung des Arduino UNO R3 Boards und des Fototransistors als Lichtsensor zur Messung der Lichtintensität im Raum. Nach Abschluss dieser Übung lernen die Schülerinnen und Schüler, wie sich Licht als Welle ausbreitet und wie sichtbares Licht Teil des elektromagnetischen Spektrums ist. Sie lernen auch, wie die Wellenlänge Arten von elektromagnetischen Wellen unterscheidet. Mit Hilfe ihrer Schaltung simulieren die Schülerinnen und Schüler, wie Licht zur Übertragung von Informationen über Glasfaseroptik verwendet wird.

Benötigte Materialien



1 ARDUINO PROJEKT

1 FOTOTRANSISTOR

1 10K Ω WIDERSTAND

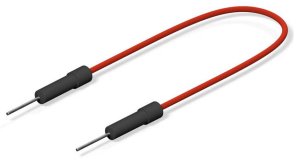
BOARD

2 STECKER-BUCHSEN-STECKVERBINDER

1 BLAUE LED

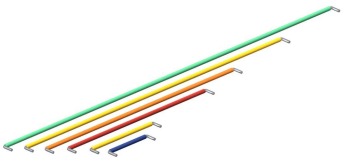
1 220 Ω-WIDERSTAND

1 SCHWARZE STROMKABEL



1 ROT STROMKABEL

1 USB-KABEL

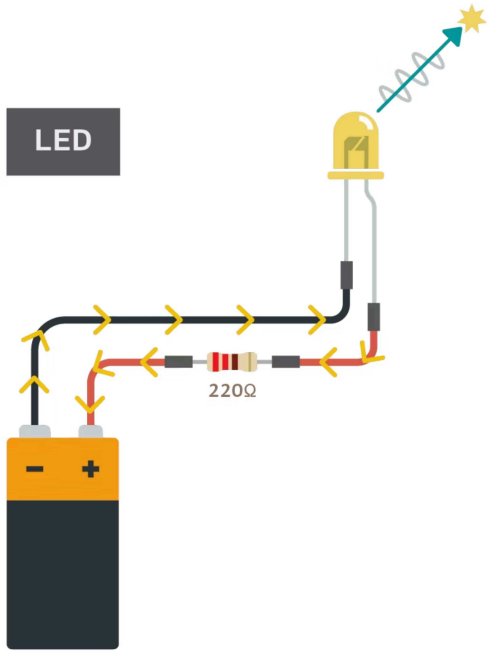


1 STECKVERBINDER

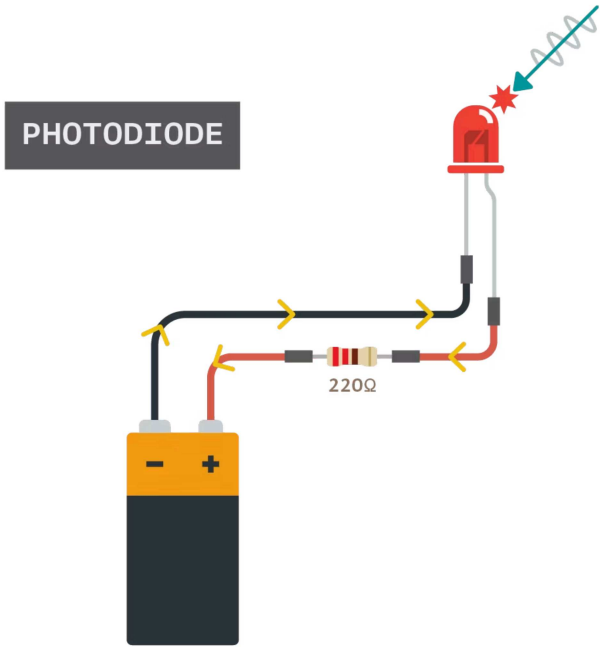
Fototransistoren

Sucht einen Fototransistor. Der Fototransistor in eurem Bausatz sieht wie eine weiße (klare) LED aus, die oben nicht rund, sondern flach ist.

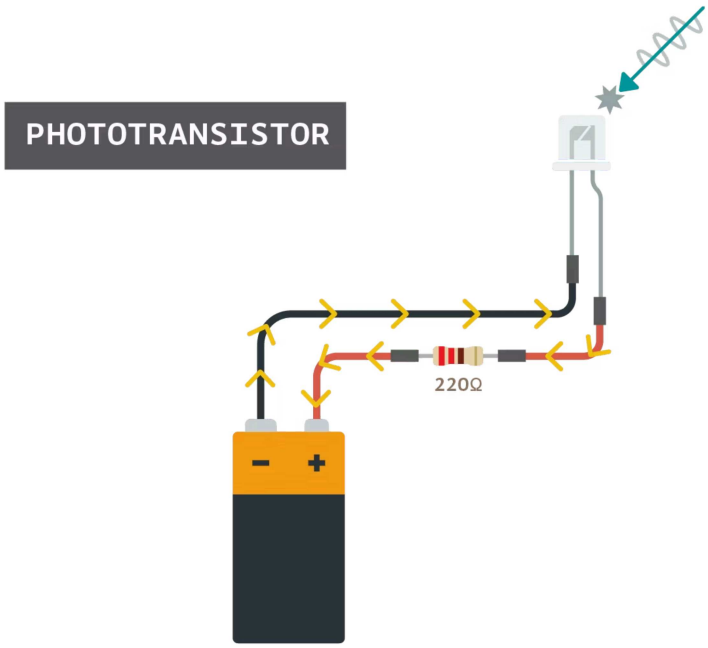
Ein **Fototransistor** ist ein elektronisches Bauteil, das Lichtenergie in elektrische Energie umwandelt. In gewisser Weise ist er das Gegenteil einer LED. Eine LED wandelt elektrische Energie in Lichtenergie um. Wenn Elektronen durch die LED fließen, werden Lichtteilchen, Photonen genannt, emittiert, und es wird Licht erzeugt.



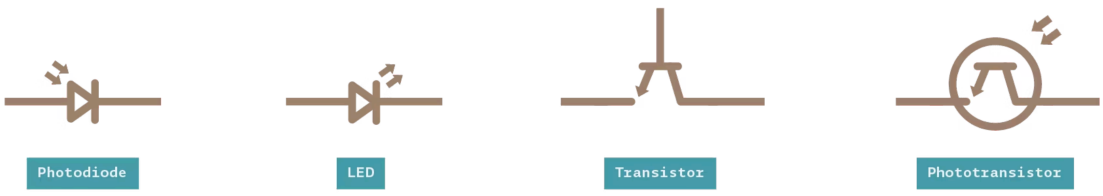
Wenn dieser Prozess umgekehrt wird, wird die LED zu einer Fotodiode und kann zur Lichtdetektion verwendet werden. Wenn Licht auf die Fotodiode trifft, wird Lichtenergie in elektrische Energie umgewandelt, und es wird elektrischer Strom erzeugt.



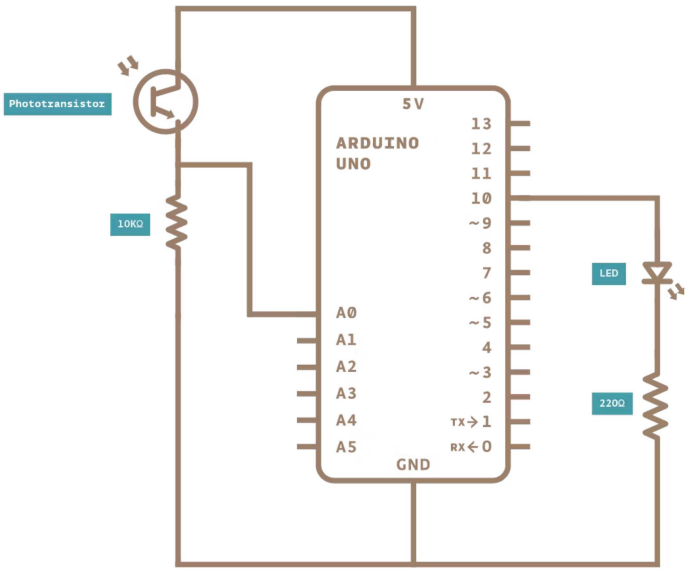
Das Problem ist, dass der von einer Fotodiode erzeugte Strom klein und nicht sehr nützlich ist. Wenn man jedoch die Fotodiode mit einem Transistor kombiniert, kann der Transistor den Strom verstärken, was ihn für die Schaltung nützlich macht.

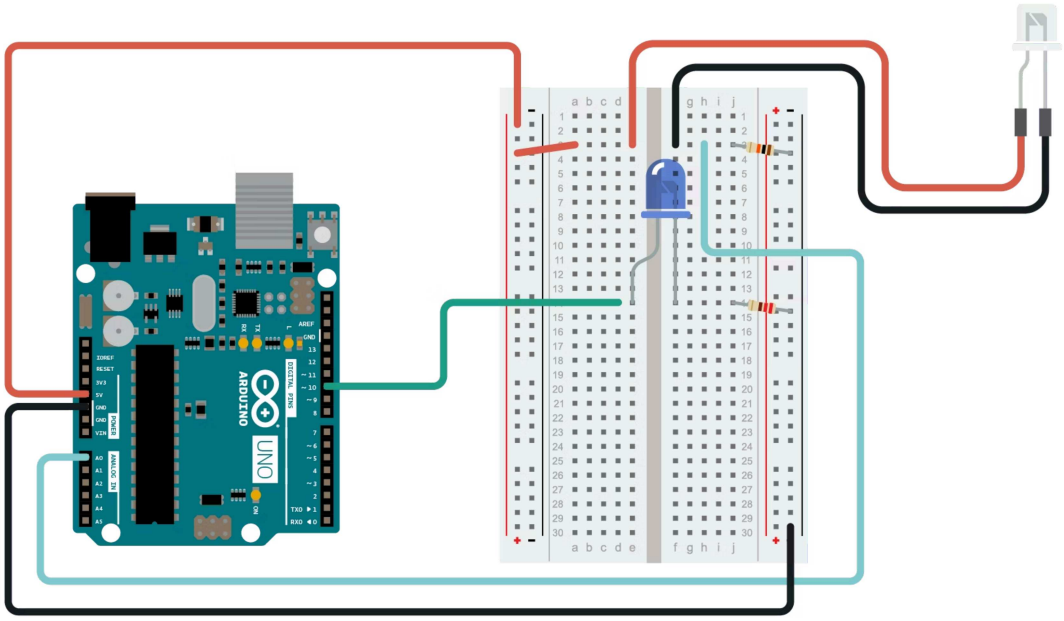


Schaut euch die Schaltzeichen für diese elektrischen Komponenten an. Eine Fotodiode ist ähnlich wie eine LED, außer dass die Pfeile, die Licht darstellen, vertauscht sind. Dies zeigt an, dass Licht eher ein Input als ein Output ist. Wenn ihr die Symbole für eine Fotodiode und einen Transistor kombiniert, dann habt ihr das Symbol für einen Fototransistor.



Schaltplan





Aufbau der Schaltung

Verwendet den Schaltplan und das Verdrahtungsschema oder die folgende Diashow, um die Schaltung aufzubauen.

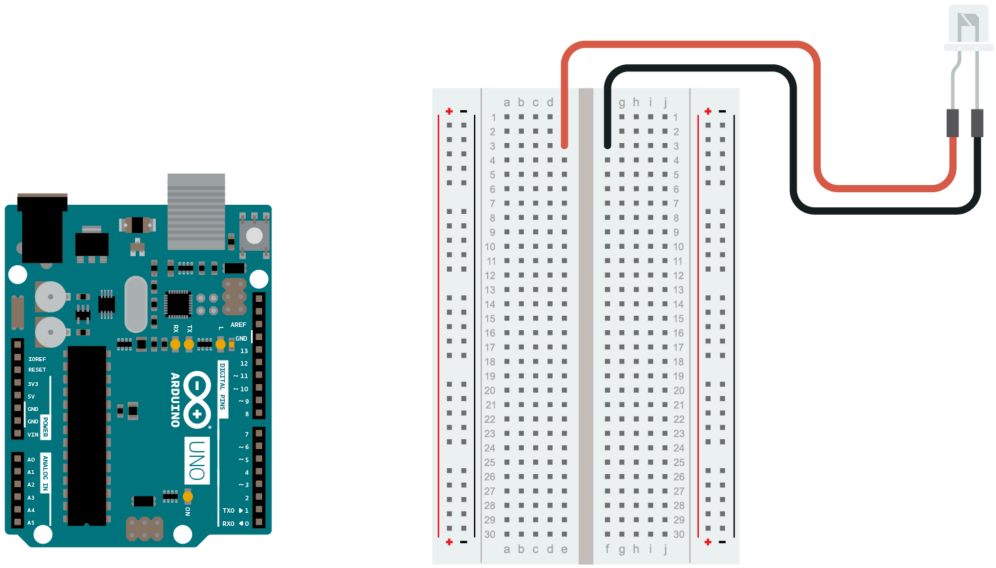
Step 1

Verbindet die Anschlüsse des Fototransistors mit den Stecker-Buchsen-Steckverbindern. Beachtet, dass der Fototransistor eine Anode (langer Anschluss)

und eine Kathode (kurzer Anschluss) hat. Schließt den roten Steckverbinder an die Anode und den schwarzen Steckverbinder an die Kathode an. Die Stecker-Buchsen-Steckverbinder sehen den Stromzuführungen sehr ähnlich. Stellt sicher, dass ihr die richtigen Drähte verwendet.

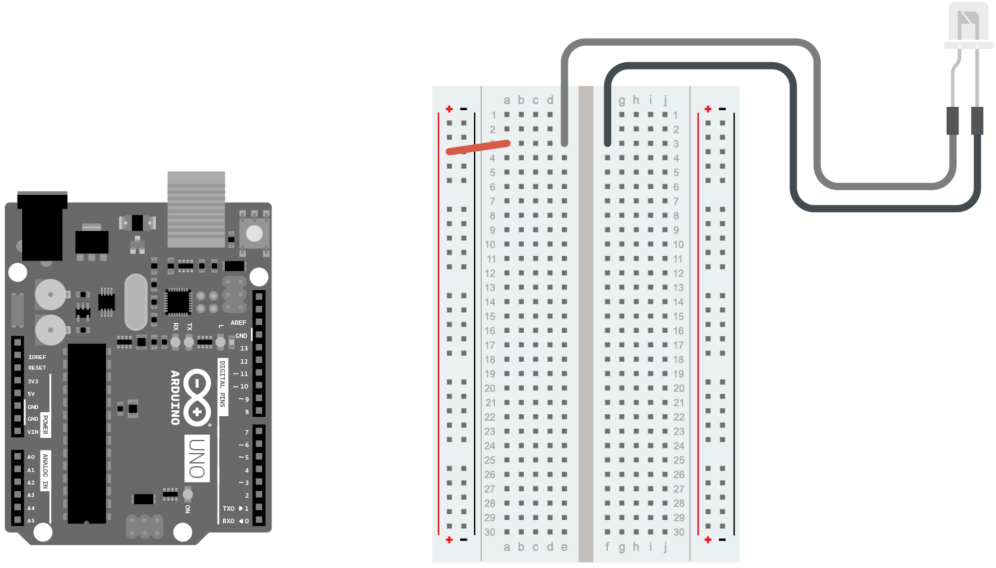
Step 2

Verbindet das andere Ende der Stecker-Buchsen-Steckverbinder mit dem Steckbrett. Der mit der Anode des Fototransistors verbundene Steckverbinder sollte sich auf der linken Seite der Lücke im Steckbrett befinden. Der Steckverbinder, der mit der Kathode des Fototransistors verbunden ist, sollte sich auf der rechten Seite des Spalts befinden.



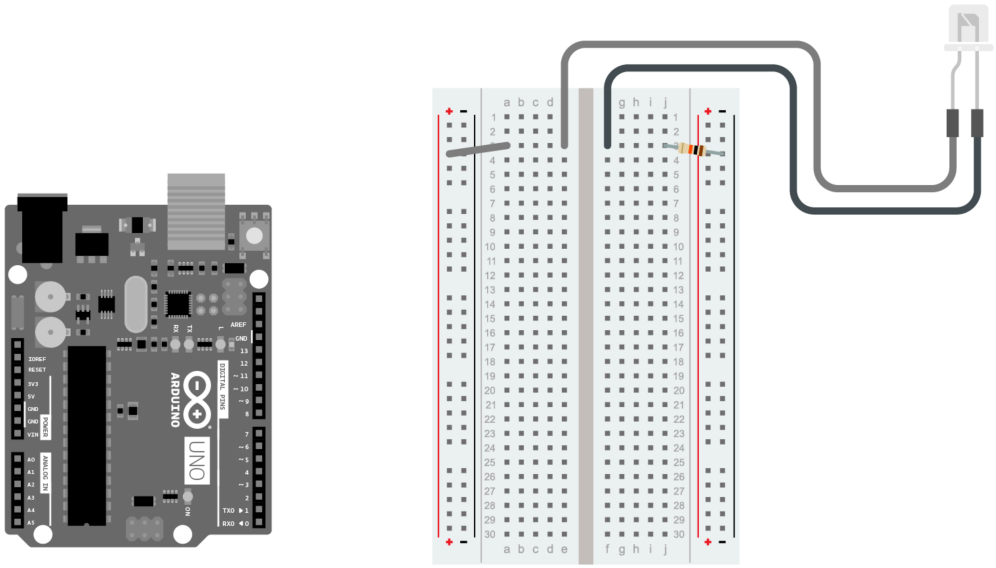
Step 3

Verwendet einen kurzen Steckverbinder, um den Anodendraht vom Fototransistor mit dem positiven Anschluss auf der linken Seite des Steckbretts zu verbinden.



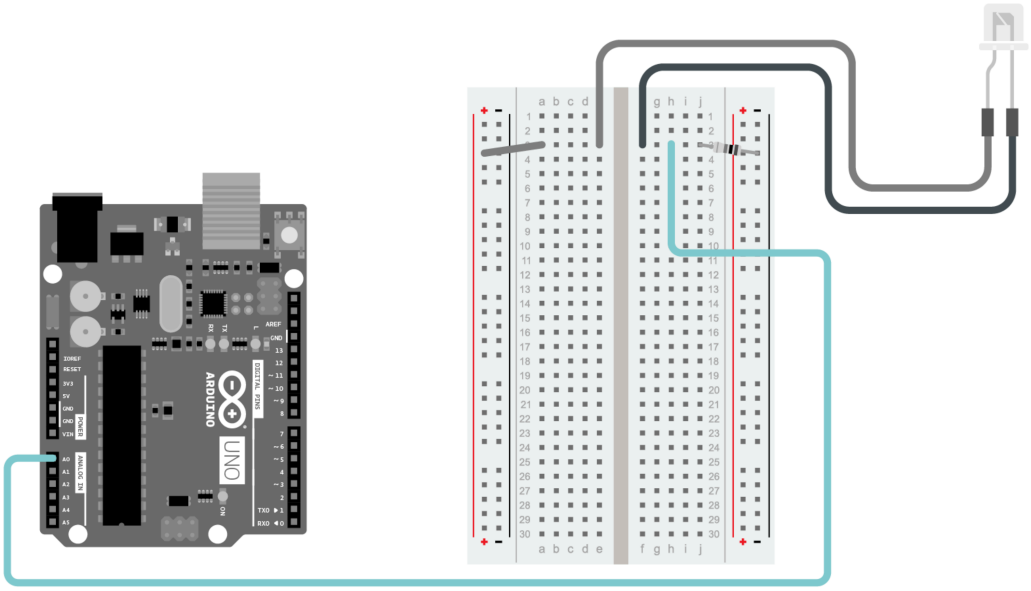
Step 4

Verwendet einen 10K-Ohm-Widerstand, um den Kathodendraht vom Fototransistor mit dem negativen Anschluss auf der rechten Seite des Steckbretts zu verbinden.



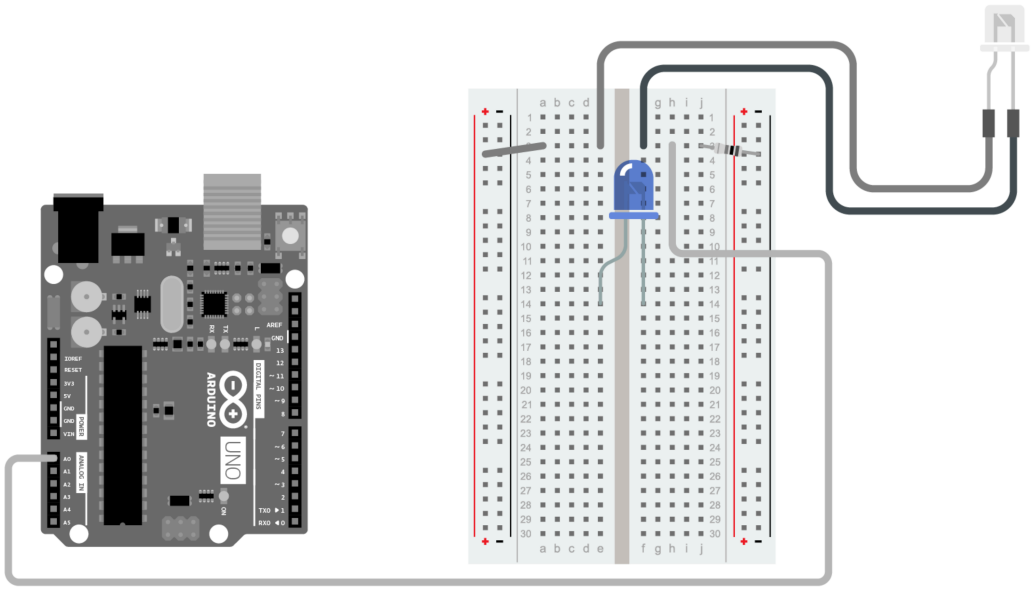
Step 5

Befestigt einen langen Steckverbinder auf dem Steckbrett zwischen dem Kathodendraht des Fototransistors und dem 10K-Ohm-Widerstand. Verbindet das andere Ende des Steckverbinders mit dem Analogpin A0 auf dem Arduino UNO R3 Board. Auf diese Weise könnt ihr den Fototransistor als Lichtsensor verwenden.



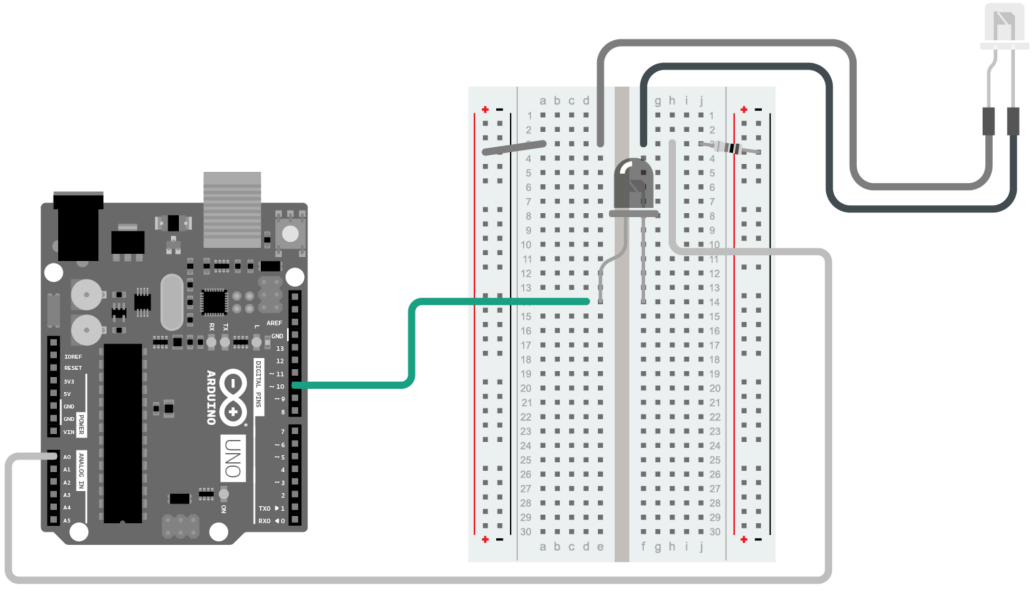
Step 6

Bringt eine blaue LED in der Mitte des Steckbretts an, sodass die LED die Lücke des Steckbretts überspannt. Achtet darauf, dass sich die Anode der LED auf der linken Seite der Lücke und die Kathode auf der rechten Seite der Lücke befindet.



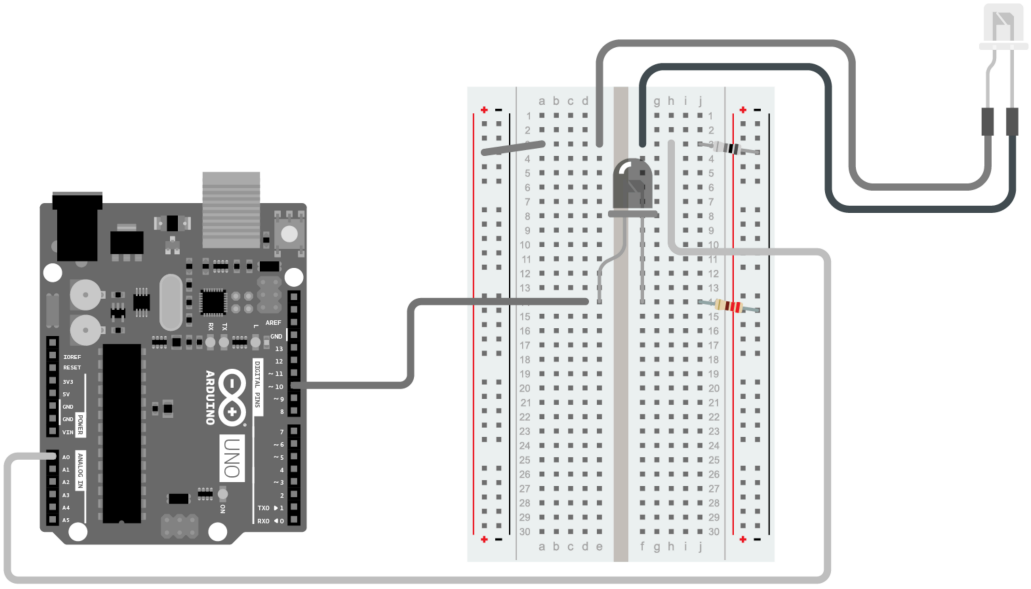
Step 7

Verbindet die Anode der LED mit einem langen Steckverbinder mit Pin 10 auf dem Arduino UNO R3 Board.



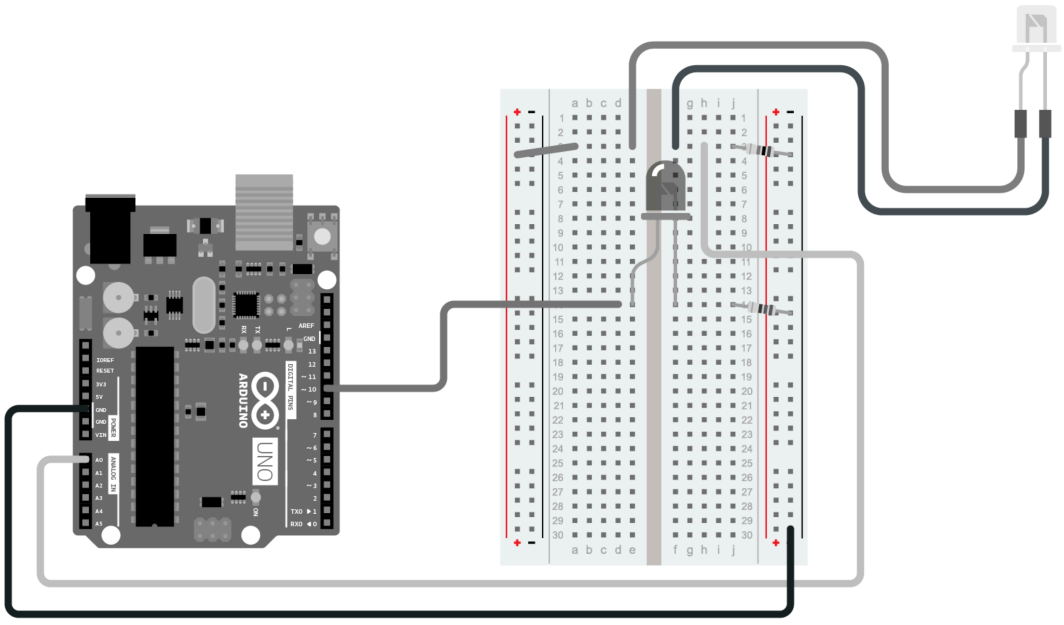
Step 8

Verwendet einen 220-Ohm-Widerstand, um die Kathode der LED mit dem Minuspol auf der rechten Seite des Steckbretts zu verbinden.



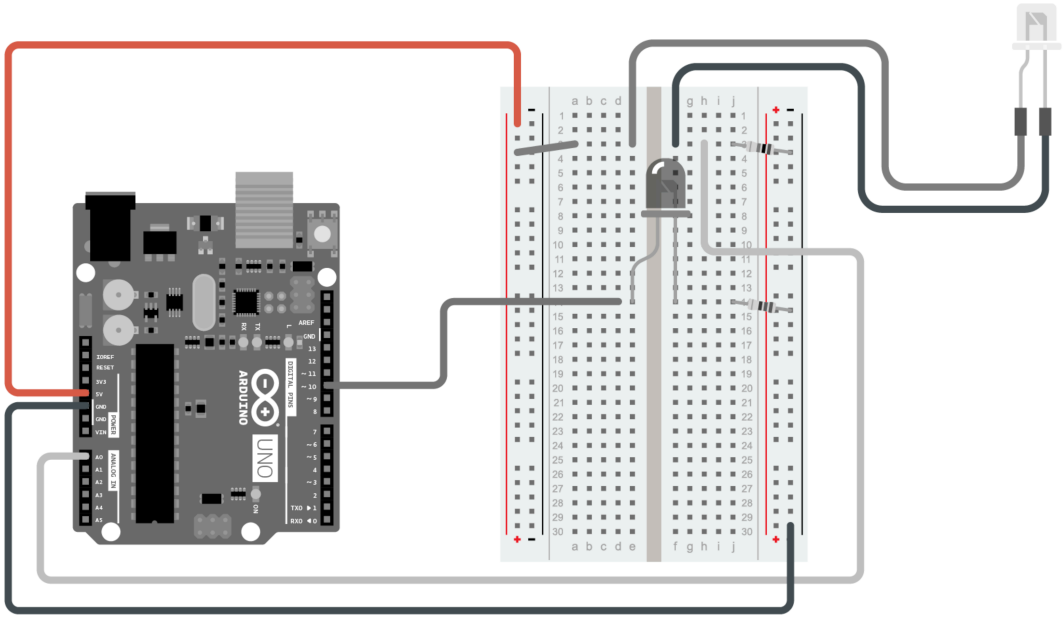
Step 9

Verwendet das schwarze Stromkabel, um den Minuspol auf der rechten Seite des Steckbretts mit einem Groundpin auf dem Arduino UNO R3 Board zu verbinden.



Step 10

Schließt das rote Stromkabel vom 5V-Pin am Arduino UNO R3 an den positiven Anschluss auf der linken Seite des Steckbretts an.



Code-Erstellung - Licht messen

Es ist an der Zeit, den Code zu schreiben, der die Schaltung steuert. Dieser erste Sketch ermöglicht es euch, den Fototransistor als Lichtsensor zu verwenden, um die Helligkeit bzw. Intensität des Lichts im Raum zu erfassen. Im Moment konzentriert ihr euch nur auf die Erstellung des Codes für den Lichtsensor. Später werdet ihr euch darauf konzentrieren, den Code zur Steuerung der LED hinzuzufügen und die LED als Lichtquelle zu verwenden.

TEACHER NOTES

Bei dieser Programmierungsübung erstellen die Schülerinnen und Schüler einen grundlegenden Sketch, der die Intensität im Raum als Analogwert des Fototransistors anzeigt. In dieser Übung werden keine neuen Kodierungskonzepte vorgestellt.

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.

3) Speichert den Sketch unter einem neuen Namen. Wählt im Dateimenü die Option **Speichern unter...** Benennt die Datei "Lektion9_V1_", gefolgt von euren Initialen. Klickt auf **Speichern**.

4) Wie immer ist eines der ersten Dinge, die zu tun sind, wenn man einen neuen Sketch für eine Schaltung beginnt, die Benennung der auf dem Arduino UNO R3 Board verwendeten Pins. Im Moment konzentriert ihr euch nur auf den Lichtsensor (Fototransistor). Der Lichtsensor sollte an den analogen Pin A0 auf dem Arduino UNO R3 Board angeschlossen werden. Erstellt zu Beginn eures Sketches eine Konstante und benennt diesen Pin **sensorPin**.

Hinweis: Ihr könnt den LED-Pin vorerst ignorieren. Ihr werdet ihn in der nächsten Übung hinzufügen.

5) Erstellt vor der Funktion **void setup()** eine Variable namens **lightAmount**. Später werdet ihr diese Variable verwenden, um den analogen Wert vom Lichtsensor zu speichern.

6) Geht zur Funktion **void setup()** eures Sketches. Gebt die Lichtintensitätsmessung vom Fototransistor auf dem seriellen Monitor aus. Erstellt einen Befehl, um den seriellen Monitor mit einer Baudrate von 9.600 zu starten.

7) Der erste Befehl in der Funktion **void loop()** sollte daraus bestehen, eine Messung der Lichtintensität vom Lichtsensor zu erhalten. Dazu ist ein **analogRead**-Befehl für den Lichtsensor erforderlich, der an den **sensorPin** angeschlossen wird. Speichert diese Messung in der Variablen **lightAmount**.

8) Nach dem Speichern der Messung müsst ihr Befehle hinzufügen, um die Lichtintensitätsmessung auf dem seriellen Monitor auszudrucken. Druckt zunächst ein Datenetikett für die Messung aus. Druckt dann die Variable **lightAmount** auf dem seriellen Monitor aus.

> **Hinweis:** Stellt sicher, dass ihr einen **Serial.println()**-Befehl als letzten Druckbefehl verwendet, so dass jede Lichtablesung auf eine neue Zeile im seriellen Monitor gedruckt wird.

9) Fügt am Ende der Funktion **void loop()** eine Verzögerung von 100 Millisekunden hinzu, um die Schleife zu verlangsamen und den seriellen Monitor davon abzuhalten, Daten so schnell zu drucken.

10) Verifiziert euren Code und debugged ihn, falls erforderlich.

11) Schließt das Arduino UNO R3 Board mit dem USB-Kabel an den Computer an.

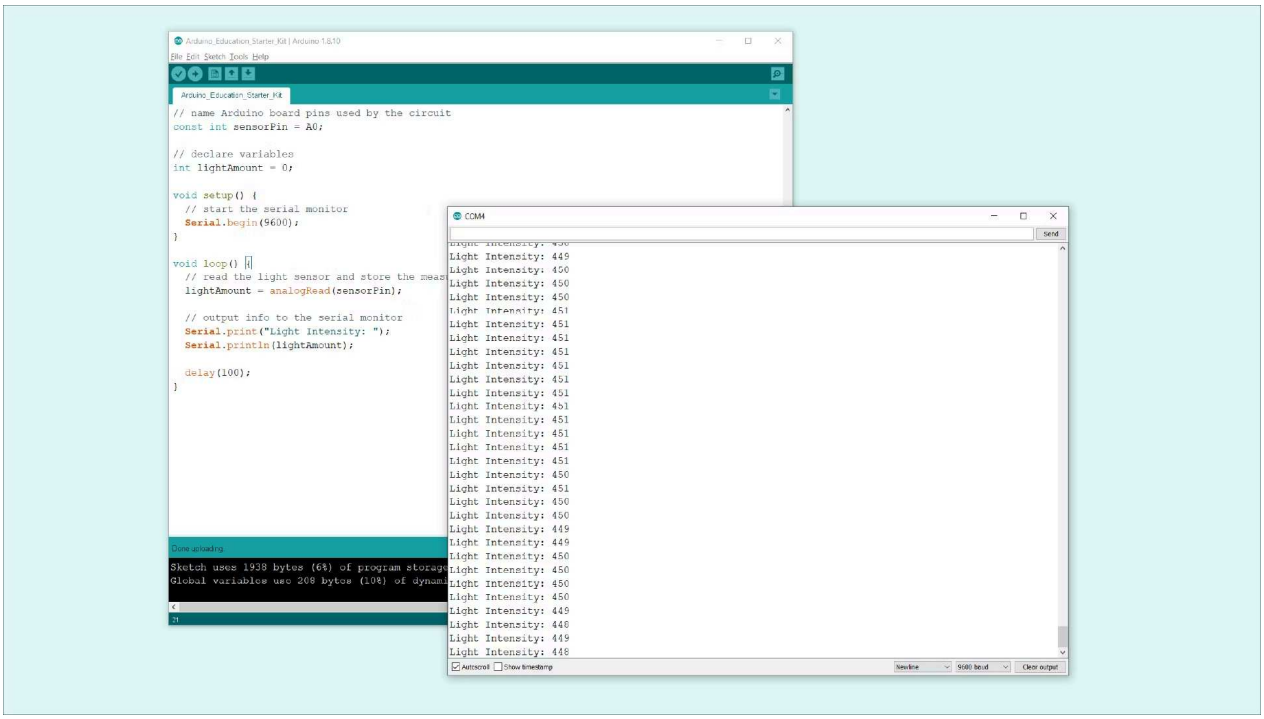
12) Geht in der Arduino IDE zum **Tools**-Menü und bewegt dann die Maus zur Menüoption **Port**. Wählt den Port aus, der zu eurem Arduino UNO R3 Board gehört.

13) Klickt auf die Schaltfläche **Hochladen**, um den Sketch auf euer Arduino UNO R3 Board hochzuladen.

14) Startet den seriellen Monitor, damit ihr die Messung des Lichtsensors beobachten könnt.

15) Bewegt den Lichtsensor herum und beobachtet die Messung auf dem seriellen Monitor. Deckt den Lichtsensor mit der Hand ab. Ihr solltet sehen, dass die Messung

sehr niedrig wird. Richtet den Lichtsensor auf ein Licht im Raum. Der Lichtsensor sollte einen hohen Messwert ausgeben.



16) Nachdem ihr euch vergewissert habt, dass euer Lichtsensor korrekt funktioniert, klickt ihr auf **Speichern**, um euren Sketch zu speichern.

ERFAHREN SIE MEHR: **Elektromagnetisches Spektrum**

Code-Erstellung - Lichtwellenkommunikation

Eine der größten Anwendungen elektromagnetischer Wellen ist die Kommunikation. Fernseher, Radios, Mobiltelefone, Computer, Bluetooth-Geräte und ferngesteuerte Roboter und Spielzeuge sind nur einige Beispiele für Geräte, die Informationen über Funkwellen senden und empfangen. Aber es sind nicht nur Radiowellen, die zur Kommunikation genutzt werden. Die **Glasfasertechnologie** nutzt Licht, um große Informationsmengen über große Entfernungen mit sehr hoher Geschwindigkeit zu übertragen.

Im Gegensatz zu Radiowellen, die sich in der Luft und sogar im Weltraum ausbreiten können, erfordert Glasfaseroptik die Verwendung eines Kabels, das aus winzigen Glas- oder Kunststoffsträngen besteht. Ein Sender wandelt und kodiert ein elektrisches Signal in digitale Impulse intensiven Laserlichts um. Das Licht wandert durch das optische Kabel zu einem Empfänger am anderen Ende des Kabels. Der Empfänger detektiert, dekodiert und wandelt die Lichtimpulse wieder in elektrische Signale um.

In dieser Übung werdet ihr einen LED- und Lichtsensor verwenden, um das Senden von Informationen durch Lichtwellen zu simulieren.

TEACHER NOTES

Bei dieser Programmierungsübung modifizieren die Schülerinnen und Schüler ihren vorherigen Sketch so, dass die blaue LED blinkt und als Lichtquelle fungiert. Während sie ihren Lichtsensor auf die LED richten, beobachten die Schülerinnen und Schüler, wie der Lichtsensor auf das blinkende Licht reagiert. Sie beziehen dieses Verhalten auf die Glasfasertechnologie und das Senden und Empfangen von Informationen durch Lichtwellen. Die Schülerinnen und Schüler werden in diese Kodierungskonzepte eingeführt:

- **Aufgerufene Funktion –** Ein Code-Abschnitt, der eine bestimmte Aufgabe abschließt und vom Hauptteil des Programms aus mehrfach aufgerufen werden kann

Die Schülerinnen und Schüler wiederholen diese Konzepte:

- **millis() –** Gibt an, wie viele Millisekunden seit dem letzten Zurücksetzen des Arduino verstrichen sind

1) Falls erforderlich, startet ihr die Arduino IDE und öffnet euren Sketch zu Lektion9_V1.

2) Speichert die Skizze unter einem neuen Namen. Wählt im Menü Datei die Option **Speichern unter...** Benennte die Datei "Lektion9_V2_", gefolgt von euren Initialen. Klickt auf **Speichern**.

3) Die LED auf euren Steckbrett sollte mit dem digitalen Pin 10 auf dem Arduino UNO R3 Board verbunden werden. Nennt zu Beginn eures Sketches Pin 10 **LEDPin**.

4) In diesem Sketch werdet ihr den Timer des Arduino für zwei verschiedene Aufgaben verwenden. Die erste Aufgabe besteht darin, die LED für eine bestimmte Zeit ein- und auszuschalten. Die zweite Aufgabe besteht darin, in bestimmten Zeitintervallen Messungen mit dem Lichtsensor durchzuführen. Erstellt zwei Variablen des langen Typs - eine mit der Bezeichnung **timerLED** und eine mit der

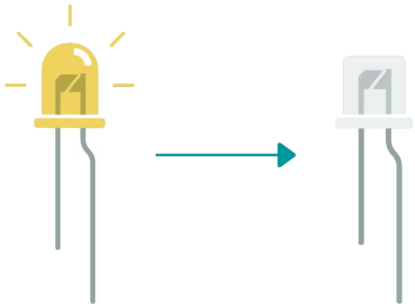
Bezeichnung **timerSensor** -, um die Zeit vom Timer des Arduino zu speichern. Denkt daran, dass Variablen des langen Typs es euch ermöglichen, einen Zeitwert von mehr als ein paar Sekunden zu speichern.

Hinweis: Denkt daran, dass ganzzahlige Variablen von -32.768 bis 32.767 reichen können. Long-Variablen können von -2.147.483.648 bis 2.147.483.647 reichen.

5) Um die LED ein- und auszuschalten, könnt ihr eine Variable verwenden, um den Zustand der LED umzuschalten. Mit anderen Worten, wenn die Variable 0 ist, ist die LED ausgeschaltet. Wenn die Variable gleich 1 ist, ist die LED eingeschaltet. Erstellt eine ganzzahlige Variable namens **toggleLED** und setzt sie gleich 0.

6) Fügt im Abschnitt **void setup()** einen Befehl hinzu, um den **pinMode** des **LEDPin** als Output festzulegen.

7) Im **void loop()**-Teil des Sketches habt ihr zwei Aufgaben. Die erste Aufgabe besteht darin, einen Sender zu erstellen, der das Licht zum blinken bringt. Die zweite Aufgabe besteht darin, einen Empfänger zu erstellen, der das Licht erfasst. Beginnt mit dem Code für den Sender.



In der **void loop()**-Funktion eures Sketches fügt ihr einen Code zur Übertragung eines Signals hinzu, indem ihr eine LED in Vier-Sekunden-Intervallen an- und ausschaltet. Um das Intervall zu erzeugen, wird der Timer von Arduino verwendet. In anderen Lektionen habt ihr eine while-Schleife verwendet, die eine Reihe von Befehlen wiederholt, solange der Timer die festgelegte Zeit noch nicht erreicht hat. Indem ihr Befehle in die while-Schleife einfügt, könnt ihr während des Wartens andere Aufgaben ausführen. Im Folgenden seht ein Beispiel, das ihr in Lektion 7 verwendet habt, in der ihr einen intervallgeschalteten Scheibenwischer erstellt habt. Während ihr auf den Timer zur Bewegung des Scheibenwischers wartet, überwacht die Schleife den Status eines Tastschalters.

In diesem Sketch muss der Empfängerteil der Schaltung jedoch in der Lage sein, Licht zu erkennen und Daten in einem anderen Zeitintervall auszugeben. Anstatt eine while-Schleife für den Sender und eine while-Schleife für den Empfänger zu verwenden, könnt ihr eine einzelne Schleife verwenden, die die Hauptfunktion **void loop()** ist. Innerhalb der Schleife könnt ihr if-Statements verwenden, um die Zeit für jede Aufgabe zu überprüfen.

TEACHER NOTES

Diese Idee, zwei verschiedene Timer-Intervalle innerhalb der **void loop()**-Funktion zu verfolgen, kann für die Schülerinnen und Schüler anfangs schwer zu begreifen sein. Wenn Ihre Schülerinnen und Schüler mit diesem Konzept Schwierigkeiten haben, ermutigen Sie sie, weiterzumachen. Es wird deutlicher werden, wenn sie den Code erstellen und das Verhalten der Schaltung beobachten. Wenn die Übung abgeschlossen ist, sollten Sie den Code vielleicht sogar als Klasse durchsprechen, wenn der Schaltkreis funktioniert, um ein bessers Verständnis für die Logik in diesem Sketch zu erhalten.

Erstellt zu Beginn der Funktion **void loop()** ein if-Statement. Die Bedingung innerhalb des if-Statements sollte Arduinos Timer unter Verwendung des Befehls **millis()** mit der Variablen **timerLED** plus 2.000 Millisekunden vergleichen, indem ein "größer als" oder "gleich"-Zeichen verwendet wird.

Hinweis: Um ein Vier-Sekunden-Intervall zu erzeugen, wird die LED für zwei Sekunden (2.000 Millisekunden) eingeschaltet und für zwei Sekunden ausgeschaltet.

8) Wenn die Bedingung des if-Statements wahr ist (alle zwei Sekunden), muss sich der Zustand der LED ändern. Erinnert euch, dass ihr die Variable **toggleLED** verwendet, um zu verfolgen, ob die LED ein- oder ausgeschaltet ist. Ihr könntet if-Statements verwenden, die besagen, dass wenn die Variable **toggleLED** 0 ist, sie auf 1 gesetzt wird, sonst wenn die Variable **toggleLED** 1 ist, sie auf 0 gesetzt wird. Eine effizientere Methode zur Programmierung desselben Verhaltens ist jedoch die Verwendung des not-Operators.

oder

Der not-Operator steht für boole'sche Werte. Das heißt, er arbeitet mit Werten, die entweder 1 oder 0 sind, wahr oder falsch, hoch oder niedrig. Er wird durch das Ausrufezeichen (!) dargestellt. Wenn dieses Symbol vor eine Variable gesetzt wird, ändert es deren Wert in den entgegengesetzten Wert. Wenn also der Wert Null ist, dann ist nicht Null gleich Eins; wenn der Wert Eins ist, dann ist nicht Eins gleich Null.

> **Hinweis:** Wenn ihr den not-Operator mit einer Variablen verwendet, die einen anderen Wert als 1 oder 0 hat, wird die Variable automatisch auf 0 geändert. Wenn eine Variable z.B. gleich 5 ist, dann ist !5 gleich 0. Wenn eine Variable gleich -10 ist, dann ist !-10 auch 0.

Erstellt innerhalb des if-Statements einen Befehl, der die Variable **toggleLED** mit dem not-Operator auf den entgegengesetzten Wert setzt.

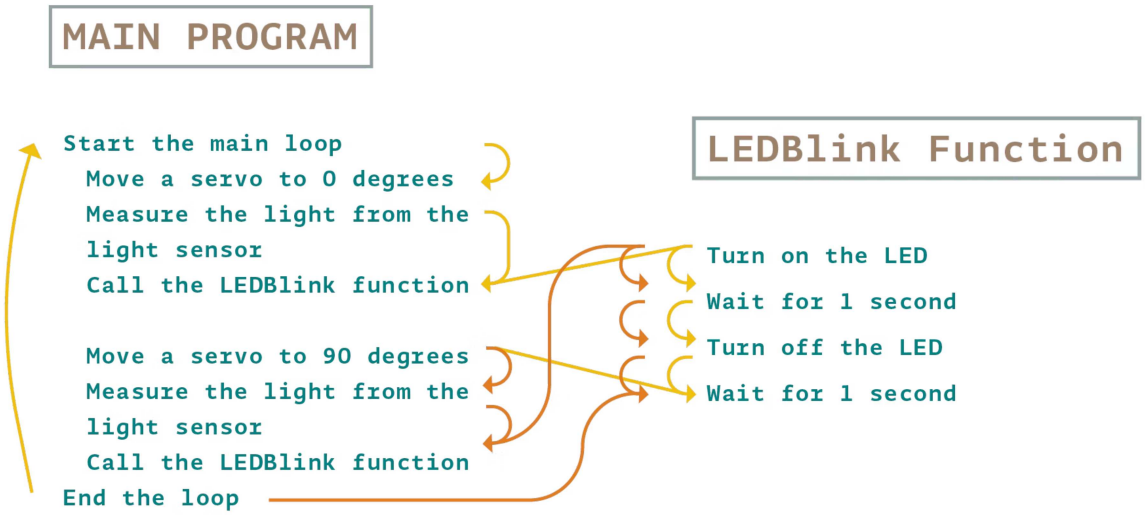
9) Fügt als Nächstes einen Befehl zum Ein- oder Ausschalten der LED basierend auf dem Wert der Variablen **toggleLED** hinzu. Verwendet einen **digitalWrite()**-Befehl, um die Konstante **LEDPin** auf den **toggleLED**-Wert zu setzen.

10) Nachdem die LED entweder ein- oder ausgeschaltet worden ist, muss der Timer zurückgesetzt werden. Fügt einen Befehl hinzu, der die Variable **timerLED** mit Hilfe des Befehls **millis()** auf die aktuelle Zeit des Arduino Timers setzt.

11) Der Senderteil eures Sketches ist komplett. Bevor ihr mit der Arbeit am Empfängerteil des Sketches beginnt, könnt ihr die Senderaufgabe in eine eigene

Funktion umwandeln. Eine **aufgerufene Funktion** ist wie ein Miniprogramm, das eine bestimmte Aufgabe ausführt. Sie kann jederzeit innerhalb des Hauptprogramms ausgeführt werden, wann immer diese Aufgabe erledigt werden muss.

Wenn Funktionen "aufgerufen" werden, verlässt das Programm den Hauptteil des Codes und springt zu der aufgerufenen Funktion und führt die Codezeilen innerhalb der Funktion aus. Wenn die Funktion beendet ist, kehrt das Programm zum Hauptcodeabschnitt zurück und macht dort weiter, wo die Funktion aufgerufen wurde. Seht euch dieses Beispiel mit Pseudocode an.



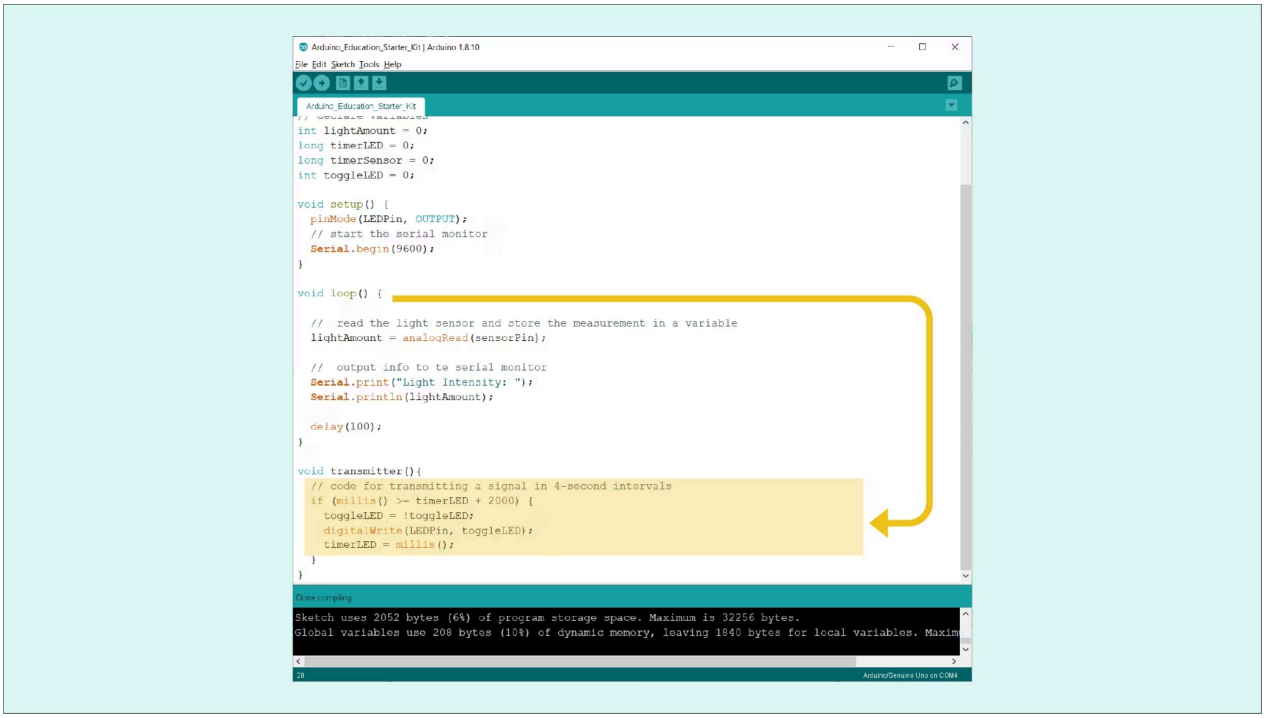
In diesem Beispiel wird die Funktion **LEDBlink** innerhalb der Hauptschleife des Sketches zweimal aufgerufen. Die Verwendung einer Funktion zum Blinken der LED verhindert, dass die gleichen Codezeilen an zwei verschiedenen Stellen geschrieben werden müssen.

Hinweis: Ein weiterer Vorteil der Verwendung von Funktionen besteht darin, dass das Debuggen von Code dadurch viel einfacher wird. Wenn ihr Code mit Hilfe von aufgerufenen Funktionen in kleine Abschnitte zerlegt, könnt ihr eine Funktion nach der anderen aufrufen, um sie zu testen. Auf diese Weise könnt ihr nur diese eine Funktion debuggen, anstatt zu versuchen, den gesamten Code auf einmal zu debuggen.

Um eine aufgerufene Funktion zu erstellen, geht ihr nach der schließenden Klammer für die Funktion **void loop()** ganz nach unten in dem Sketch. Genau wie die Funktionen **void setup()** und **void loop()** beginnen andere aufgerufene Funktionen mit dem Wort *void*, gefolgt vom Namen der Funktion. Die Benennung von Funktionen folgt den gleichen Regeln wie die Benennung von Variablen oder Konstanten. Ihr könnt Funktionen mit allem benennen, was kein Schlüsselwort oder Befehl in der Arduino IDE ist. Befehle für eine Funktion werden in geschweifte Klammern für die Funktion gesetzt. Erstellt unten in dem Sketch eine neue Funktion mit dem Namen **void transmitter()**. Eure neue Funktion sollte wie folgt aussehen:

Hinweis: Zwischen dem Namen der Funktion und der öffnenden Klammer, die die Befehle für die Funktion enthält, befindet sich eine Reige offener und geschlossener Klammern. Bei einigen Funktionen ist es wichtig, Informationen wie Variablen oder andere Werte aus dem Hauptteil des Programms in die Funktion zu übernehmen. Diese Informationen werden verwendet, um die Aufgabe für die Funktion zu erledigen. Bei den Funktionen, die ihr in diesem Kurs schreibt, ist es jedoch nicht erforderlich, Informationen in die Funktion zu übernehmen. Das liegt daran, dass ihr Variablen als globale Variablen erstellt habt, die überall in dem Sketch verwendet werden können, einschließlich aufgerufener Funktionen. Daher sollten in den Funktionen, die ihr erstellt, die Klammern leer sein.

12) Verschiebe das gesamte if-Statement, das die LED zum Blinken bringt, von der **void loop()**-Funktion in die **void transmitter()**-Funktion am unteren Ende eures Sketches. Ihr könnt den Code ausschneiden und einfügen, oder ihr könnt den Code markieren und an die gewünschte Stelle ziehen.



13) Als Nächstes müsst ihr den Befehl hinzufügen, der die Funktion aufruft. Geht zurück zur Funktion **void loop()**. Der Befehl, der eine Funktion aufruft, ist einfach der Name der Funktion, gefolgt von Klammern. Am Anfang der **void loop()**-Funktion fügt ihr einen Befehl zum Aufruf der **transmitter()**-Funktion hinzu.

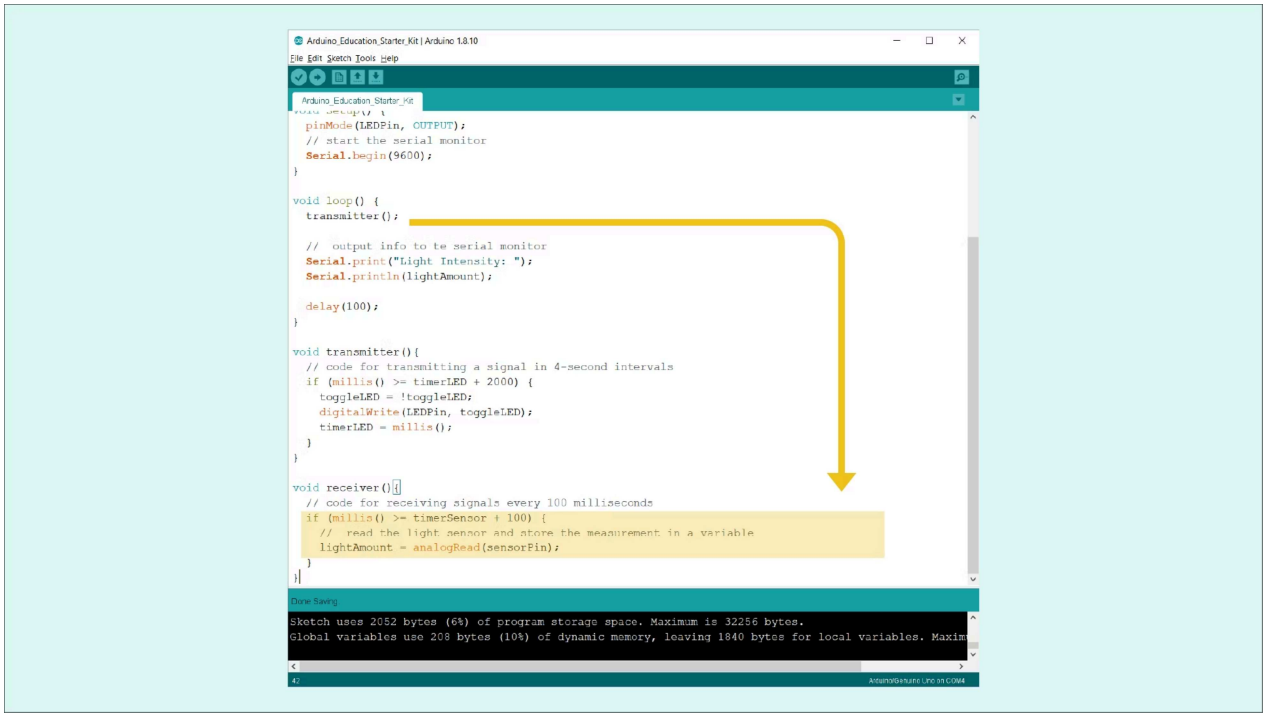
14) Überprüft euren Code und debugged ihn wenn nötig.

15) Nun ist es an der Zeit, sich auf den Empfängerteil des Sketches zu konzentrieren. Ihr könnt eine weitere aufgerufene Funktion zur Überprüfung des Lichtsensors erstellen. Erstellt unten in eurem Sketch unterhalb der **transmitter()**-Funktion eine weitere Funktion namens **receiver()**.

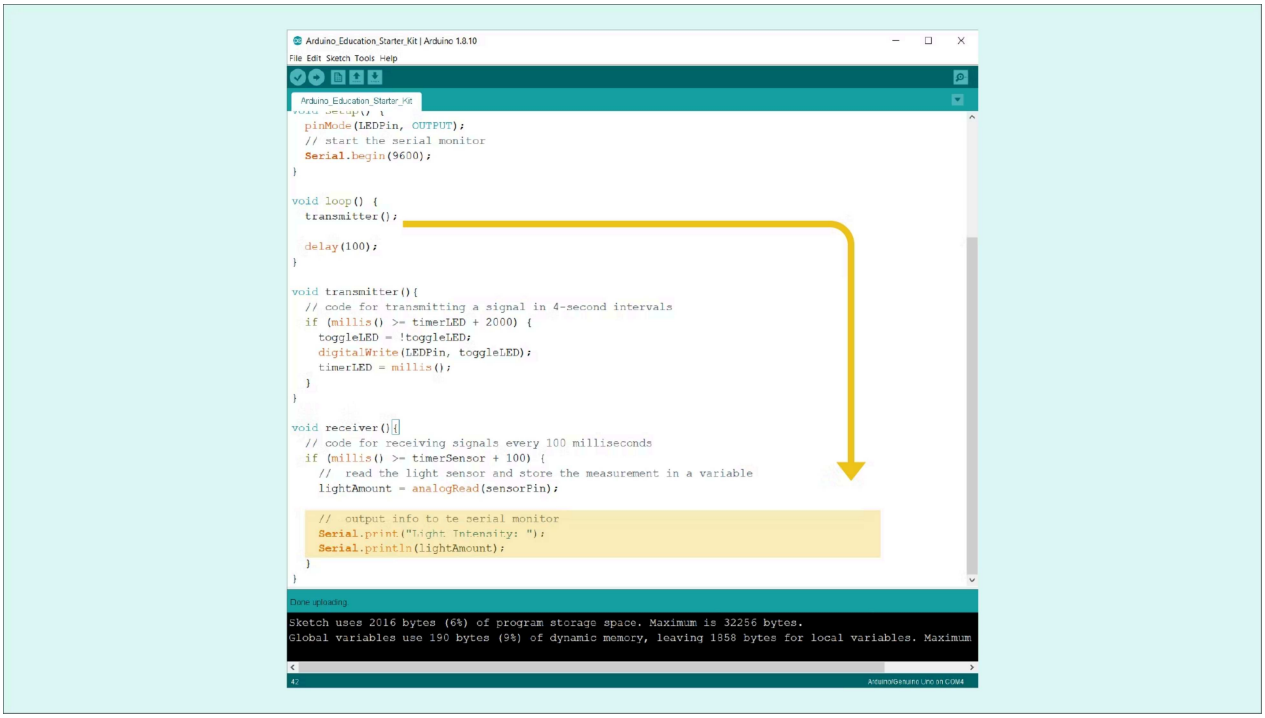
Um zu vermeiden, dass der Empfänger bei jeder Wiederholung der Schleife den Status des Lichtsensors ausgibt, könnt ihr erneut den Timer von Arduino verwenden, um in bestimmten Zeitintervallen eine Lichtmessung durchzuführen. Verwendet für diese Funktion die Variable **timerSensor**, um die Zeit und ein Zeitintervall von 100 Millisekunden zu verfolgen. Erstellt innerhalb der Funktion **void receiver()** ein

weiteres if-Statement. Die Bedingung innerhalb des if-Statements sollte den Arduino Timer unter Verwendung des Befehls **millis()** mit der **timerSensor**-Variablen plus 100 Millisekunden vergleichen, indem das "größer" oder "gleich"-Zechen verwendet wird.

16) Verschiebt den Befehl zum Ablesen des Lichtsensors und zum Speichern des Wertes in der **lightAmount**-Variablen aus der Hauptschleife hinunter in das if-Statement der Empfängerfunktion. Dies veranlasst den Empfänger, alle 100 Millisekunden einen Messwert zu erfassen.



17) Verschiebt auch die Befehle zum Drucken der Variablen **lightAmount** auf den seriellen Monitor hinunter in das if-Statement der Empfängerfunktion.



18) Nachdem der Lichtsensor eine Lichtmessung vorgenommen und diese auf dem seriellen Monitor ausgedruckt hat, muss der Timer zurückgesetzt werden. Fügt am Ende des if-Statements der Funktion **void receiver()** einen Befehl hinzu, der die Variable **timerSensor** mit Hilfe des Befehls **millis()** auf die aktuelle Zeit auf dem Arduino Timer setzt.

19) Die Empfängerfunktion ist nun abgeschlossen. Als Nächstes müsst ihr den Befehl hinzufügen, der die Funktion aufruft. Fügt in der Funktion **void loop()** unterhalb des Befehls, der die Funktion **transmitter()** aufruft, einen Befehl hinzu, der die Funktion **receiver()** aufruft.

20) Schließlich benötigt ihr nun die Verzögerung von 100 Millisekunden in der Funktion **void loop()** nicht mehr, da ihr den Timer zur Steuerung sowohl des Senders als auch des Empfängers verwendet. Löscht die 100-Millisekunden-Verzögerung in der **void loop()**-Funktion.

TEACHER NOTES

VERSTÄNDNIS-ÜBERPRÜFUNG

Stellen Sie den Schülerinnen und Schülern die folgenden Fragen zu den Funktionen:

- **Was ist eine aufgerufene Funktion?**

Antwort: Eine aufgerufene Funktion ist eine Reihe von Befehlen außerhalb des Hauptteils eines Programms, die eine bestimmte Aufgabe ausführen.

- **Warum sind aufgerufene Funktionen nützlich?**

Antwort: Aufgerufene Funktionen ermöglichen es euch, den Code in kleinere Abschnitte zu zerlegen, die bestimmte Aufgaben ausführen. Auf diese Weise könnt ihr auch jeweils eine Aufgabe oder einen Codeabschnitt debuggen, indem ihr die einzelnen Funktionen nacheinander aufruft. Aufgerufene Funktionen sind auch für Aufgaben nützlich, die in einem Programm mehrfach wiederholt werden müssen. Anstatt die Code-Zeilen mehrfach zu schreiben, könnt ihr den Code in eine Funktion schreiben und die Funktion dann bei Bedarf im Programm aufrufen.

Sie können sich von den Schülerinnen und Schülern auch die Reihenfolge erklären lassen, in der die Befehle aus dem Sketch ausgeführt werden, und wie das Programm zwischen der Funktion **void loop()** und den anderen aufgerufenen Funktionen hin- und herspringt.

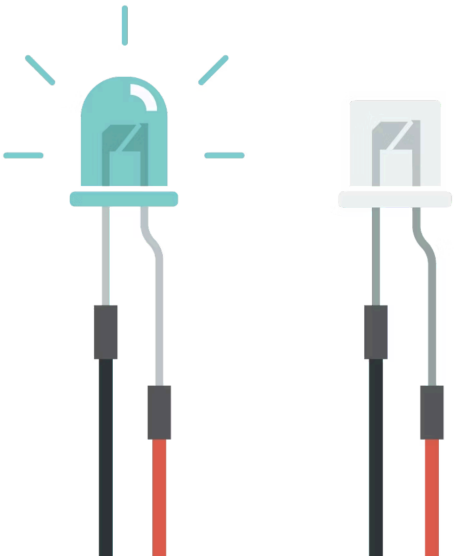
21) Verifiziert euren Code und debugged ihn, falls erforderlich.

22) Schließt das Arduino UNO R3 Board mit dem USB-Kabel an den Computer an.

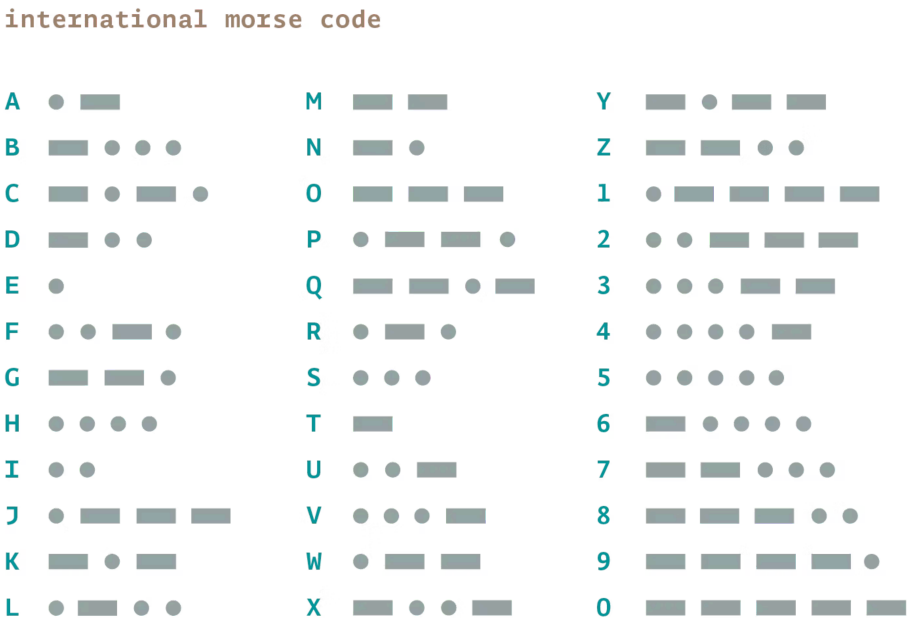
23) Klickt auf die Schaltfläche **Hochladen**, um euren Sketch auf euer Arduino UNO R3 Board hochzuladen.

24) Startet den seriellen Monitor, damit ihr die Messung des Lichtsensors beobachten könnt.

25) Richtet den Lichtsensor auf die blaue LED. Beobachtet den Messwert des seriellen Monitors, während die LED ein- und ausblinkt. Beobachtet, was mit der Lichtintensität der LED geschieht, wenn ihr den Sensor näher zu der LED oder weiter von ihr weg bewegt.



Der Telegrafenapparat war das erste Gerät, das Informationen über große Entfernungen übertragen konnte. Telegrafen schickten Nachrichten über einen leitenden Kupferdraht mit Hilfe von Morsecode. Im Morsealphabet wird jede Zahl und jeder Buchstabe des Alphabets durch eine Reihe von Punkten und Strichen dargestellt. Diese Punkte und Striche konnten elektrisch über den Draht übertragen und am Empfangsende als Audio-Klicks und Pieptöne empfangen werden.

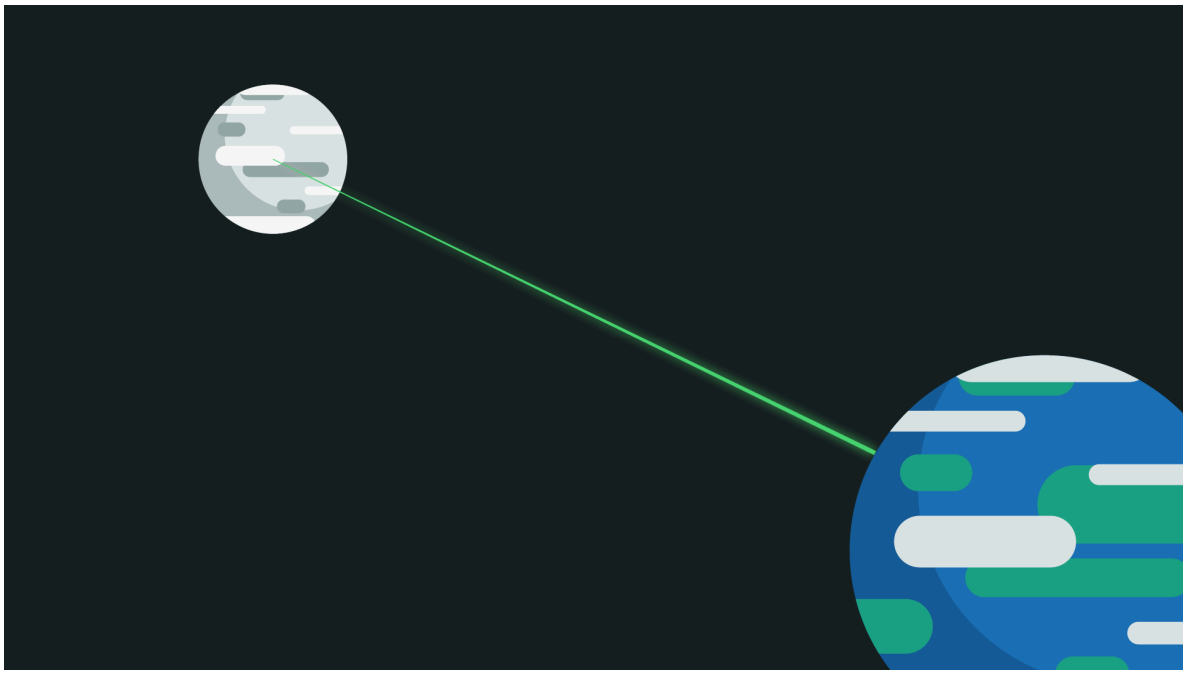


Faseroptik funktioniert nach dem gleichen Grundprinzip wie eine Telegrafenmaschine. Bei der Faseroptik werden die Daten jedoch mit Licht durch ein

Kabel aus winzigen Glassträngen übertragen, anstatt ein elektrisches Signal durch einen Kupferdraht. Durch die Verwendung digitaler Codierung anstelle von Morsezeichen können über Glasfaserkabel Tausende von Telefongesprächen, Webseiten, Videos, Liedern und anderen Medien in sehr kurzer Zeit übertragen werden.

Die Schaltung, die ihr gerade gebaut habt, ähnelt einem Glasfasersystem mit zwei wesentlichen Unterschieden. Erstens verwendet ein faseroptisches System ein Kabel zwischen dem Sender (blaue LED) und dem Empfänger (Lichtsensor), das verhindert, dass das Licht über große Entfernungen gestreut wird. Der andere Hauptunterschied besteht darin, dass der Sender bei der Übertragung großer Datenmengen sehr schnell mit verschiedenen Impulsen blinkt, um Bits kodierter Informationen zu übertragen.

Hinweis: Die NASA verwendet ein ähnliches System wie die Schaltung, die ihr gerade gebaut habt, um die Entfernung zum Mond zu messen. Im Jahr 1969 landeten Neil Armstrong und Buzz Aldrin auf dem Mond. Sie hinterließen etwas mehr als Fußabdrücke im Mondstaub. Sie hinterließen das Lunar Laser Ranging Retroreflector Array - im Grunde genommen ein ausgefallener Spiegel, der auf die Erde gerichtet ist.



Zurück auf der Erde sendet ein starker Laser, der auf den Spiegel gerichtet ist, Lichtimpulse aus, die mit Lichtgeschwindigkeit durch den Weltraum rasen. Das Laserlicht, welches von Teleskopen erfasst wird, wird vom Spiegel reflektiert und kehrt zur Erde zurück. Anhand der Lichtgeschwindigkeit und der Zeit, die der

Laserpuls braucht, um zum Mond und zurück zu gelangen, können Wissenschaftler die Entfernung zum Mond sehr genau bestimmen.

26) Wenn ihr mit der Beobachtung der Schaltung fertig seid, klickt ihr auf **Speichern**, um euren Sketch zu speichern. Behaltet eure Schaltung für die nächste Übung aufgebaut.

Serielle Eingänge

In diesem Kurs habt ihr gesehen, wie Eingabegeräte wie zum Beispiel Tastschalter und Potentiometer zur Steuerung der Schaltung verwendet werden können. Diese physikalischen Steuergeräte ermöglichen die Interaktion zwischen Mensch und Schaltung. Es gibt jedoch noch eine andere Möglichkeit, wie Menschen mit einem Arduino UNO R3 Board interagieren und ihm Anweisungen senden können - den seriellen Monitor.

Ihr habt bereits gesehen, wie der serielle Monitor als Ausgabewerkzeug zur Anzeige von Informationen und Daten verwendet werden kann. Er kann aber auch dazu verwendet werden, Informationen einzugeben und dem Arduino UNO R3 Board mitzuteilen, was zu tun ist. Sehen wir uns einen schnellen und einfachen Beispielsketch an. Ihr könnt eure Schaltung aus der vorherigen Übung für diesen Sketch verwenden.

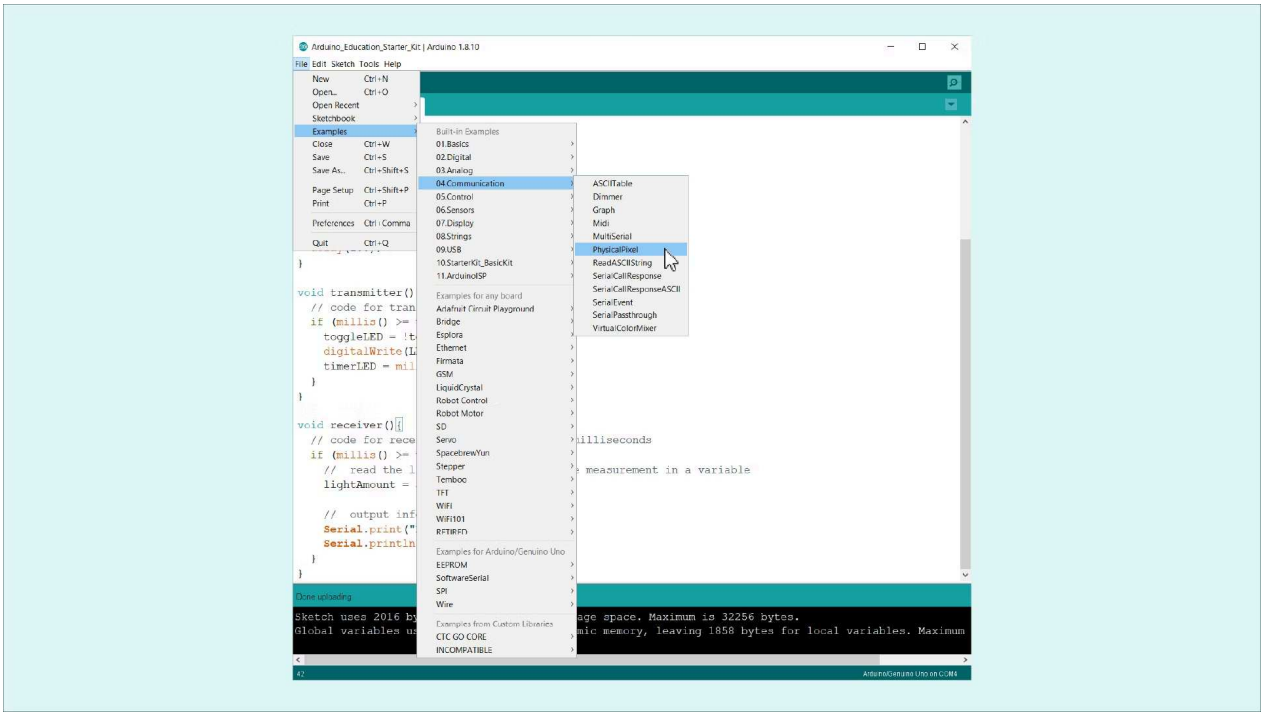
TEACHER NOTES

In dieser Übung untersuchen die Schülerinnen und Schüler einen Beispielsketch, der die eingebaute LED auf ihrem Arduino UNO R3 Board steuert. Sie werden in diese Kodierungskonzepte eingeführt:

- **Serial.available()** – Prüft, ob Informationen über serielle Kommunikation an das Arduino UNO R3 Board gesendet wurden
- **Serial.read()** – Liest den Wert der Information, die an das Arduino UNO R3 Board gesendet wurde, Byte für Byte

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

2) Wählt im Dateimenü den Befehl **Beispiele > 04.Kommunikation > PhysicalPixel.** Der Sketch wird in einem neuen Fenster geöffnet.

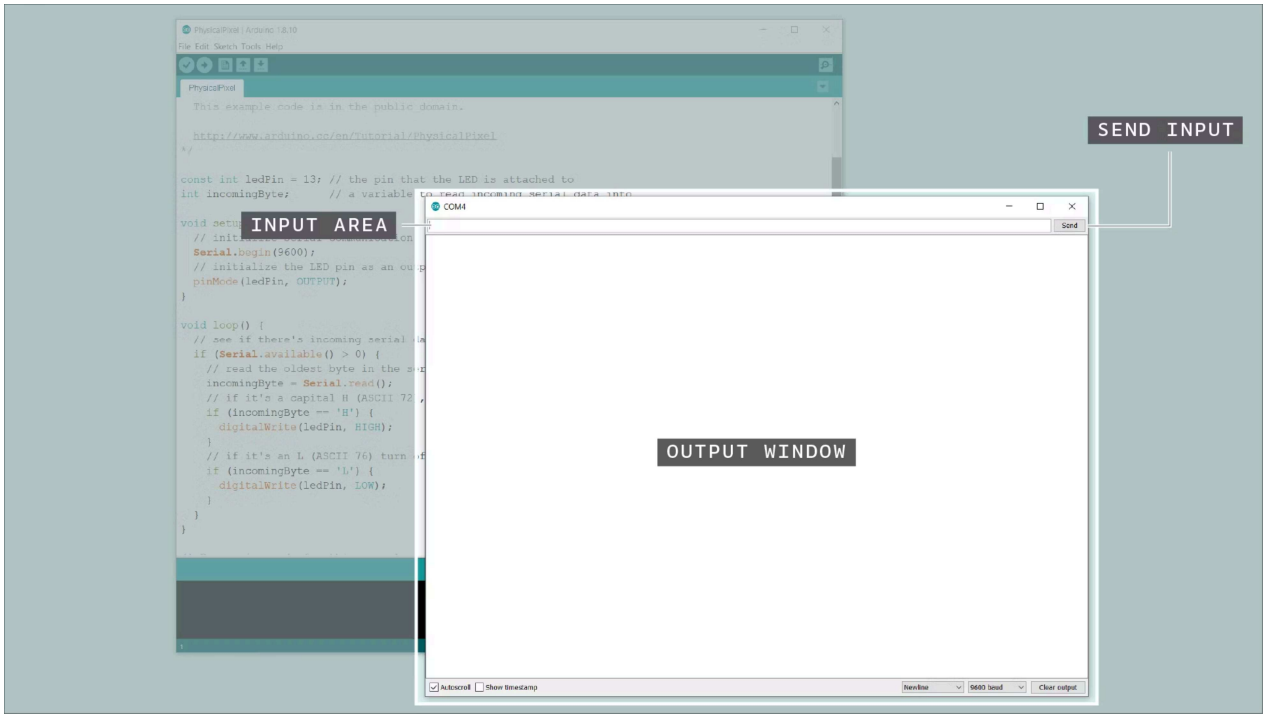


Hinweis: Schließt das andere IDE-Fenster, falls es noch geöffnet ist. Ihr werdet es nicht benötigen.

3) Schließt das USB-Kabel an das Arduino UNO R3 Board an.

4) Ladet den Sketch auf euer Arduino UNO R3 Board hoch.

5) Startet den seriellen Monitor. Bisher habt ihr in diesem Kurs das Ausgabefenster des seriellen Monitors verwendet, um Daten und Informationen aus dem Sketch anzuzeigen. Oberhalb des Ausgabefensters befindet sich jedoch ein Eingabebereich, in dem ihr Informationsbits an das Arduino UNO R3 Board senden könnt.



6) In diesem Beispielsketch könnt ihr die eingebaute LED des Arduino UNO R3 Boards durch Eingabe von Befehlen in den seriellen Monitor ein- und ausschalten. Gebt auf der Eingangsleitung des seriellen Monitors ein großes **H** ein, um die eingebaute LED in den Zustand HIGH (an) zu versetzen. Klicket auf **Senden**, um die Informationen an das Arduino UNO R3 Board zu senden. Die eingebaute LED in der Nähe von Pin 13 sollte sich einschalten.

7) Gebt ein großes **L** in den Eingangsbereich des seriellen Monitors ein, um die eingebaute LED in den LOW (Aus)-Zustand zu versetzen. Klickt auf **Senden**, um den Befehl an das Arduino UNO R3 Board zu senden. Die eingebaute LED sollte nun erlöschen.

8) Wenn Informationen über den seriellen Monitor eingegeben werden, werden sie in einem **Puffer** gespeichert. Ein Puffer ist ein Ort, der Daten für kurze Zeit speichert, während sie von einem Ort zum anderen bewegt werden. Der Speicherpuffer des Arduino ist ein FIFO-Puffer, was für first in, first out steht. Stellt euch den Arduino-Puffer wie die Schlange im Supermarkt oder die Mittagsschlange in der Schule (ohne Abkürzungen) vor. Wer zuerst ankommt, verlässt die Schlange als Erster. Ein Datenpuffer funktioniert auf die gleiche Weise. In welcher Reihenfolge auch immer die Daten eingegeben werden, werden sie vom Arduino in der gleichen Reihenfolge verarbeitet.

Gebt im Serienmonitor ein großes **H** gefolgt von einem großen **L** ein und klickt auf **Senden**.

Ihr konntet wahrscheinlich nichts sehen, weil es so schnell ging, aber das **H** wurde zuerst verarbeitet und schaltete die eingebaute LED ein. Dann wurde das **L** verarbeitet und schaltete die LED wieder aus. Dies geschah so schnell, dass ihr es wahrscheinlich nicht gesehen habt.

Versucht, eine Reihe von **H**s (vielleicht 20) und dann ein *L* einzugeben und auf **Senden** zu klicken. Je nachdem, wie viele **H**s ihr eingebt, seht ihr möglicherweise ein schnelles Blinken der LED, während jedes einzelne **H** verarbeitet wird und die LED aufleuchtet. Zuletzt wird das **L** verarbeitet und schaltet die LED wieder aus.

9) Seht euch den Sketch an, den ihr auf das Arduino UNO R3 Board hochgeladen habt. In der Funktion **void loop()** prüft der Befehl **Serial.available()**, ob Informationen über den seriellen Monitor gesendet wurden. Stellt euch das so vor, als ob ihr eure E-Mails abrufen würdet. Wenn Informationen gesendet wurden, gibt dieser Befehl eine 1 zurück, andernfalls eine 0.

Die Daten werden Byte für Byte aus dem Speicherpuffer verarbeitet. Erinnert euch daran, dass ein Byte eine achtstellige Binärzahl ist, die sich aus Einsen und Nullen zusammensetzt. Jeder Buchstabe (sowohl Groß- als auch Kleinbuchstaben), jede Ziffer (0-9) und viele Sonderzeichen und Tastaturfunktionen können durch ein anderes Byte dargestellt werden. Dies wird als **ASCII-Code** bezeichnet. Zum Beispiel ist der Großbuchstabe H die Zahl 72, die als Byte 1001000 dargestellt wird.

> **Hinweis:** ASCII steht für American Standard Code for Information Interchange.

Decimal	Binary	Char	Decimal	Binary	Char	Decimal	Binary	Char
0	0	[NULL]	43	101011	+	86	1010110	V
1	1	[START OF HEADING]	44	101100	,	87	1010111	W
2	10	[START OF TEXT]	45	101101	-	88	1011000	X
3	11	[END OF TEXT]	46	101110	.	89	1011001	Y
4	100	[END OF TRANSMISSION]	47	101111	/	90	1011010	Z
5	101	[ENQUIRY]	48	110000	0	91	1011011	[
6	110	[ACKNOWLEDGE]	49	110001	1	92	1011100	\
7	111	[BELL]	50	110010	2	93	1011101	]
8	1000	[BACKSPACE]	51	110011	3	94	1011110	^
9	1001	[HORIZONTAL TAB]	52	110100	4	95	1011111	_
10	1010	[LINE FEED]	53	110101	5	96	1100000	`
11	1011	[VERTICAL TAB]	54	110110	6	97	1100001	a
12	1100	[FORM FEED]	55	110111	7	98	1100010	b
13	1101	[CARRIAGE RETURN]	56	111000	8	99	1100011	c
14	1110	[SHIFT OUT]	57	111001	9	100	1100100	d
15	1111	[SHIFT IN]	58	111010	:	101	1100101	e
16	10000	[DATA LINK ESCAPE]	59	111011	;	102	1100110	f
17	10001	[DEVICE CONTROL 1]	60	111100	<	103	1100111	g
18	10010	[DEVICE CONTROL 2]	61	111101	=	104	1101000	h
19	10011	[DEVICE CONTROL 3]	62	111110	>	105	1101001	i
20	10100	[DEVICE CONTROL 4]	63	111111	?	106	1101010	j
21	10101	[NEGATIVE ACKNOWLEDGE]	64	1000000	@	107	1101011	k
22	10110	[SYNCHRONOUS IDLE]	65	1000001	A	108	1101100	l
23	10111	[ENG OF TRANS. BLOCK]	66	1000010	B	109	1101101	m
24	11000	[CANCEL]	67	1000011	C	110	1101110	n
25	11001	[END OF MEDIUM]	68	1000100	D	111	1101111	o
26	11010	[SUBSTITUTE]	69	1000101	E	112	1110000	p
27	11011	[ESCAPE]	70	1000110	F	113	1110001	q
28	11100	[FILE SEPARATOR]	71	1000111	G	114	1110010	r
29	11101	[GROUP SEPARATOR]	72	1001000	H	115	1110011	s
30	11110	[RECORD SEPARATOR]	73	1001001	I	116	1110100	t
31	11111	[UNIT SEPARATOR]	74	1001010	J	117	1110101	u
32	100000	[SPACE]	75	1001011	K	118	1110110	v
33	100001	!	76	1001100	L	119	1110111	w
34	100010	"	77	1001101	M	120	1111000	x
35	100011	#	78	1001110	N	121	1111001	y
36	100100	$	79	1001111	O	122	1111010	z
37	100101	%	80	1010000	P	123	1111011	{
38	100110	&	81	1010001	Q	124	1111100	|
39	100111	'	82	1010010	R	125	1111101	}
40	101000	(	83	1010011	S	126	1111110	~
41	101001	)	84	1010100	T	127	1111111	[DEL]
42	101010	*	85	1010101	U			

Der Befehl **Serial.read()** ruft das letzte vom seriellen Monitor gesendete Byte ab. Dieser Wert wird in der Variablen **incomingByte** gespeichert.

Dann gibt es zwei if-Statements, die die **eingehende Byte**-Variable entweder mit einem 'H' oder einem 'L' vergleichen. Beachtet, dass diese Buchstaben von einfachen Anführungszeichen statt normaler Anführungszeichen umgeben sind. Das einfache Anführungszeichen ist eine Möglichkeit, dem Arduino mitzuteilen, dass es die **eingehende Byte**-Variable mit dem Byte-Wert von **H** oder **L** vergleichen soll. Wenn sie mit dem Byte-Wert für **H** übereinstimmt, wird die eingebaute LED eingeschaltet. Wenn sie mit dem Byte-Wert von **L** übereinstimmt, wird die eingebaute LED ausgeschaltet.

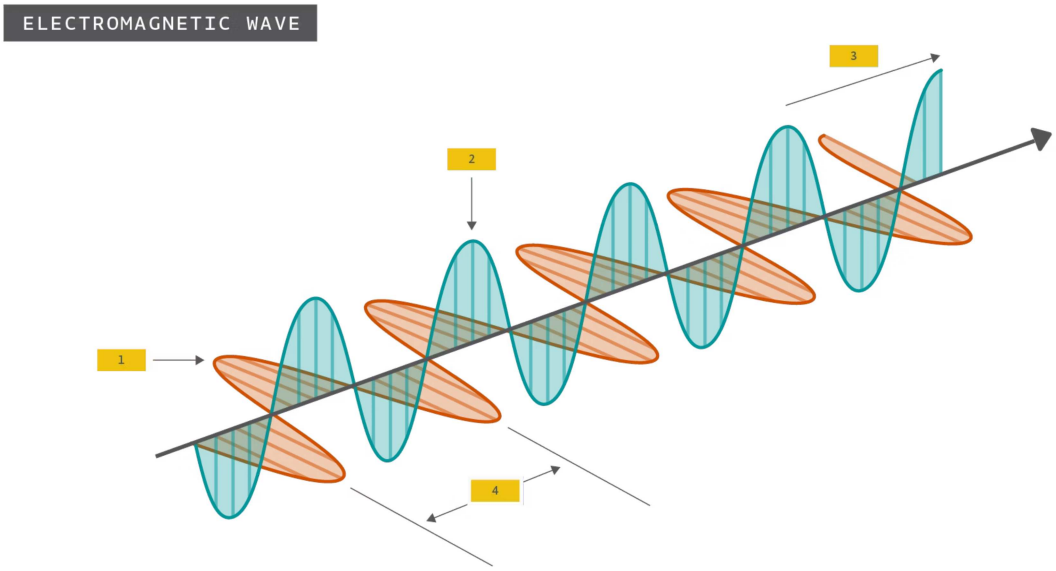
Hinweis: Die Verwendung von einfachen Anführungszeichen ist praktisch, weil es euch davor bewahrt, die Byte-Werte für alle verschiedenen Byte-Werte und ASCII-Codes für verschiedene Zeichen kennen zu müssen.

10) Zieht das USB-Kabel vom Arduino UNO R3 Board ab, um den Schaltkreis abzuschalten. Behaltet euren Schaltkreis für die nächste Übung aufgebaut.

Blickpunkt Erfindungen

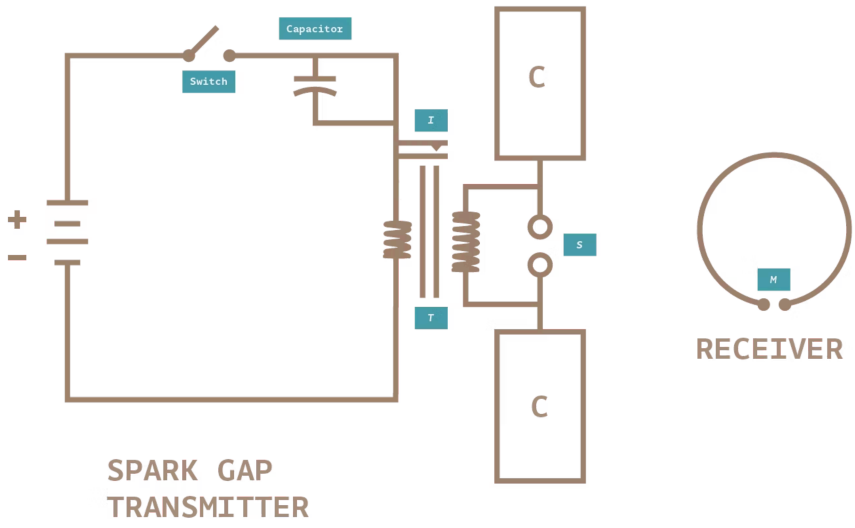
Elektromagnetische Energie

Michael Faraday und anderen Forschern ist es zu verdanken, dass ein neues Bild des Universums in den Blickpunkt gerückt ist. Die Wissenschaftler hatten begonnen zu vermuten, dass die Kräfte der Elektrizität und des Magnetismus (zusammen als elektromagnetische Energie bezeichnet) als unsichtbare Wellen durch den Raum wandern.



1) Magnetisches Feld 2) Elektrisches Feld 3) Ausbreitungsrichtung 4) Wellenlänge

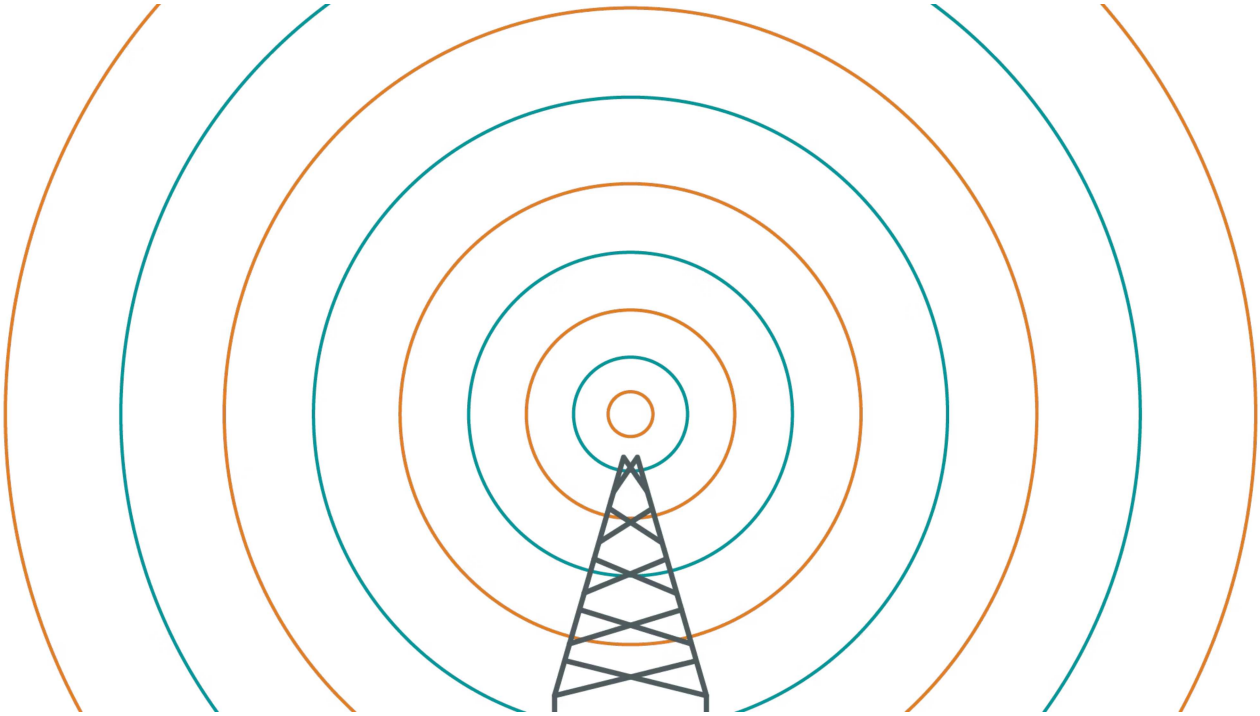
Schon als Student war Heinrich Hertz davon besessen, ein Experiment zu schaffen, um dies zu beweisen. Im Jahr 1887 schuf er einen elektrischen Sender und einen Empfänger. Der Sender erzeugte einen elektrischen Funken, der zwischen zwei stromführenden Drähten übersprang. Als Reaktion darauf erzeugte der Empfänger einen eigenen Funken - obwohl er mehrere Meter entfernt war!



Dieses und andere Experiment von Hertz bewiesen, dass sich elektromagnetische Energie als Wellen durch den Raum bewegt. Dies warf eine interessante Frage auf: Welches Medium durchqueren elektromagnetische Wellen? Schließlich bewegen sich Meereswellen durch Wasser. Schallwellen durchqueren die Luft.

Zu Hertz' Zeit stellten sich die Wissenschaftler ein Medium vor, das sie als leuchtenden Äther bezeichneten. Doch im 20. Jahrhundert bewiesen Experimente, dass es einen solchen Äther nicht gibt. Heute glauben die Wissenschaftler, dass für elektromagnetische Wellen einfach kein Medium erforderlich ist. Seit der Entdeckung von Hertz sind auch andere tiefe Mysterien aufgetaucht. So hat man zum Beispiel herausgefunden, dass die Wellen in manchen Situationen eher als Teilchen wirken. Über die Bedeutung dieser Tatsache wird bis heute heftig debattiert.

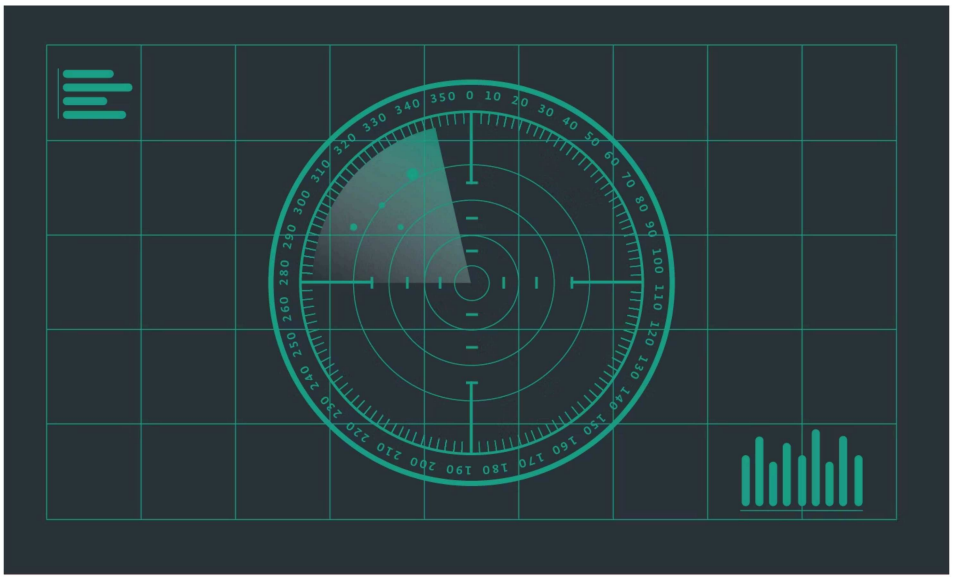
Obwohl spätere Entdeckungen die Situation komplizierter machten, war die Entdeckung von Hertz ein erstaunlicher Durchbruch. Aber wie so oft wurde die Bedeutung damals nicht verstanden - nicht einmal von Hertz selbst. Seine Entdeckung führte schließlich zu Radio, Fernsehen, Radar und mehr. Doch als Hertz nach der Nützlichkeit seiner Entdeckung gefragt wurde, sagte er: "Sie nützt überhaupt nichts". Traurigerweise starb Hertz zu jung, um den Beitrag, den er zur Technologie beisteuerte, zu sehen.



Es brauchte Andere, um die Entdeckung praktisch nutzbar zu machen. Der Erfinder und Geschäftsmann Guglielmo Marconi verdiente sich seinen Platz in der Geschichte, indem er bewies, dass diese Wellen zur Kommunikation genutzt werden konnten. Gegen den Wunsch seines Vaters führte der junge Marconi seine eigenen Experimente durch. Die Entfernungen, über die er Wellen sandte und empfing, wuchsen von einigen Metern bis zu der Entfernung über einen Ozean.

Wie ihr vielleicht schon vermutet habt, war Marconi ein Pionier des Rundfunks. Aber die ersten Investoren, die Interesse an Marconis Übertragungssystemen zeigten, war die Marine. Das Radio bot Schiffen eine bessere Möglichkeit, nach Hilfe zu rufen.

Marconi entwickelte auch eine weitere für die Marine nützliche Technologie: das Radar. Elektromagnetische Wellen prallen von einigen Oberflächen ab. Radarbediener senden einen Impuls aus und erkennen dann, ob die Wellen zurück reflektieren. Dies zeigt an, ob ein Objekt vorhanden ist. Es ist sogar möglich, die Entfernung zum Objekt zu berechnen, indem die Rückkehr zeitlich gesteuert wird.



Während Hertz nicht mehr lebte, um seinen vollen Beitrag zu realisieren, wurde Marconi zu seiner Zeit zum Synonym für Radio. Einige Jahre später erlangte eine andere Innovatorin großen Ruhm, aber nicht für ihre wichtigen Erfindungen; Hedy Lamarr war zwar ein weltberühmter Filmstar, aber nur wenige wussten von ihren Erfindungen. Lamarr entwarf eine Verbesserung der Ampel, schlug Konstruktionsänderungen für Experimentalflugzeuge vor und trug vor allem zur Verbreitung des Wechselsprektrumverfahrens bei.

Während des Zweiten Weltkriegs erfuhr Lamarr, dass amerikanische Torpedos, die von Funkwellen gesteuert wurden, von feindlichen Booten blockiert werden konnten. Die Boote sendeten ein konkurrierendes Funksignal aus, das den Torpedo verwirrte. Sie arbeitete mit einem Pianisten (George Antheil) zusammen, um ein Funkleitsystem zu entwickeln, das von einer Frequenz zur anderen sprang. Das sich bewegende Signal war viel schwieriger zu stören. Sie benutzten eine Notenrolle, um unter den Frequenzen auszuwählen. Folglich bot ihr System 88 Frequenzen an - so wie es 88 Tasten auf einem Klavier gibt.

Das Militär war zunächst nicht an dem Wechselspektrumverfahren interessiert, machte aber Jahrzehnte später davon Gebrauch. Nicht nur das, sondern auch die angewandte Technik wurde zu einem Pionier der Technologie, die heute in der Mobiltelefon- und Wi-Fi-Kommunikation eingesetzt wird.


Wi-Fi

Lichtwellenradar

Neben der Kommunikation werden elektromagnetische Wellen auf vielfältige Weise genutzt. Mikrowellenherde erwärmen Lebensmittel, indem sie Mikrowellen aussenden, die auf die Lebensmittel gerichtet sind. Infrarotwellen werden in ferngesteuerten Geräten zur Steuerung von Fernsehern verwendet. Gammastrahlen werden oft als eine Form der Behandlung zur Abtötung von Krebszellen eingesetzt. Fluglotsen verwenden Radar- und Radiowellen, um die Position von Flugzeugen am Himmel zu kartographieren und den Fluss der ein- und ausfliegenden Flugzeuge zu steuern.

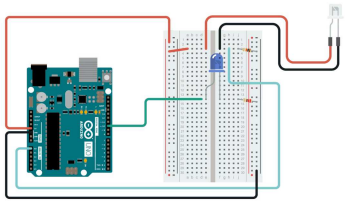
Obwohl die meisten Radarsysteme Funkwellen zur Verfolgung von Objekten verwenden, verwenden einige Systeme, die Lidar genannt werden, Laserlicht. Wie Radarsysteme werden auch Lidarsysteme zur Erstellung von Karten verwendet, die oft die Entfernung zu Objekten anzeigen. Ihr werdet in dieser Übung jedoch keine Laser verwenden; Ihr werdet ein Gerät erstellen und programmieren, das Licht auf ganz ähnliche Weise erkennt wie Radar Funkwellen oder Lidar Laserlicht. Euer Lichtwellenradar wird einen Servo verwenden, um den Fototransistor im Raum auszurichten und die Lichtintensität zu messen.

Hinweis: Lidar steht für Lichterfassung und Entfernungsmessung.

Benötigte Materialien

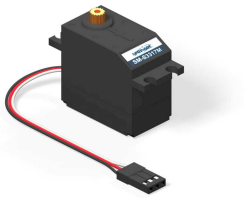
TEACHER NOTES

Wenn es für die Schülerinnen und Schüler an der Zeit ist, die Schaltung für diese Übung zu bauen, müssen Sie ihnen ein kurzes Stück Klebeband zur Verfügung stellen, um den Fototransistor am Servohorn zu befestigen. Die meisten Klebebandtypen werden funktionieren; es wird jedoch empfohlen, ein Klebeband zu verwenden, das leicht entfernt werden kann, ohne klebrige Rückstände auf dem Servo oder Fototransistor zu hinterlassen. Das Klebeband sollte sofort entfernt werden, wenn die Übung abgeschlossen ist.

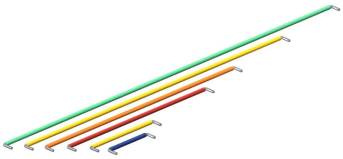


1 LICHTSENSORSCHALTUNG AUS VORHERIGER ÜBUNG

1 100 MIKROFARAD KONDENSATOR



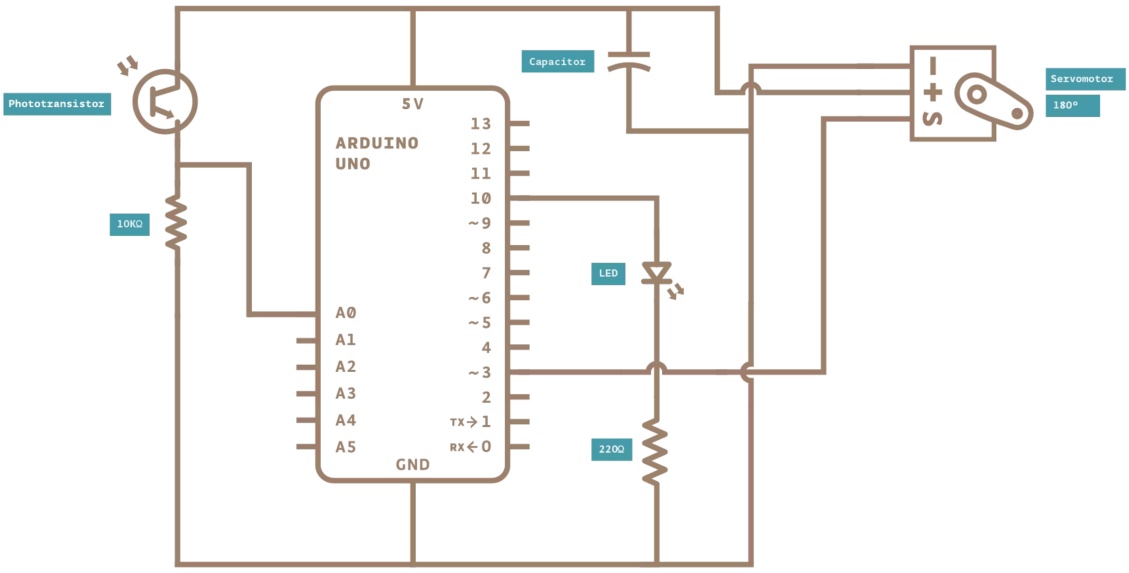
1 SERVO

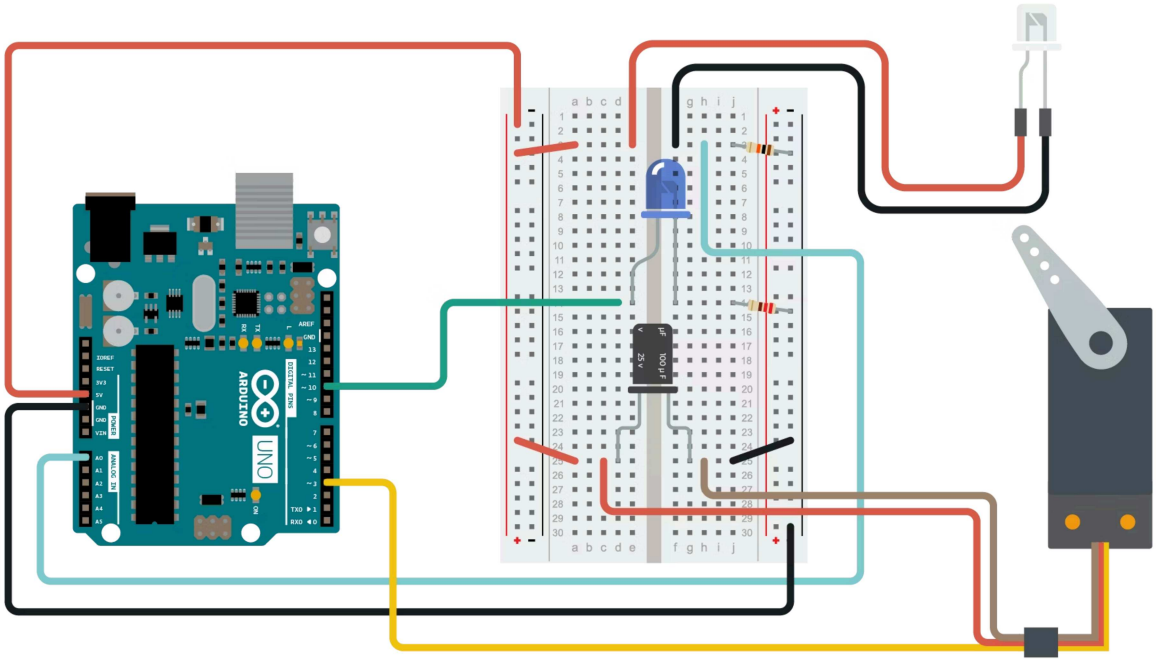


1 STECKVERBINDER

1 USB-KABEL

Schaltplan



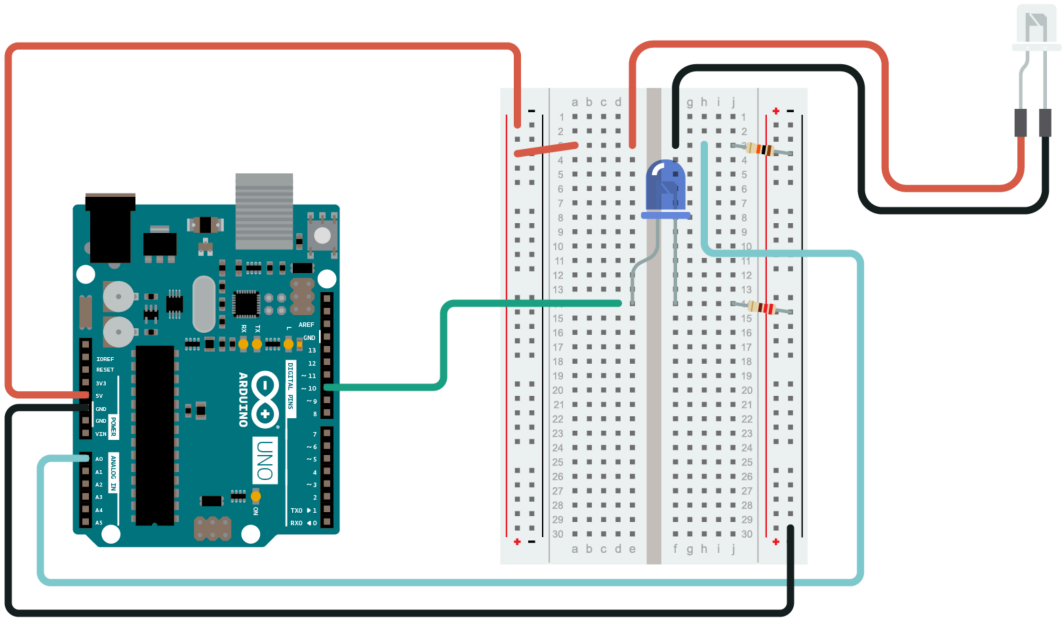


Aufbau der Schaltung

Verwendet den Schaltplan und das Verdrahtungsschema oder die folgende Diashow, um die Schaltung aufzubauen.

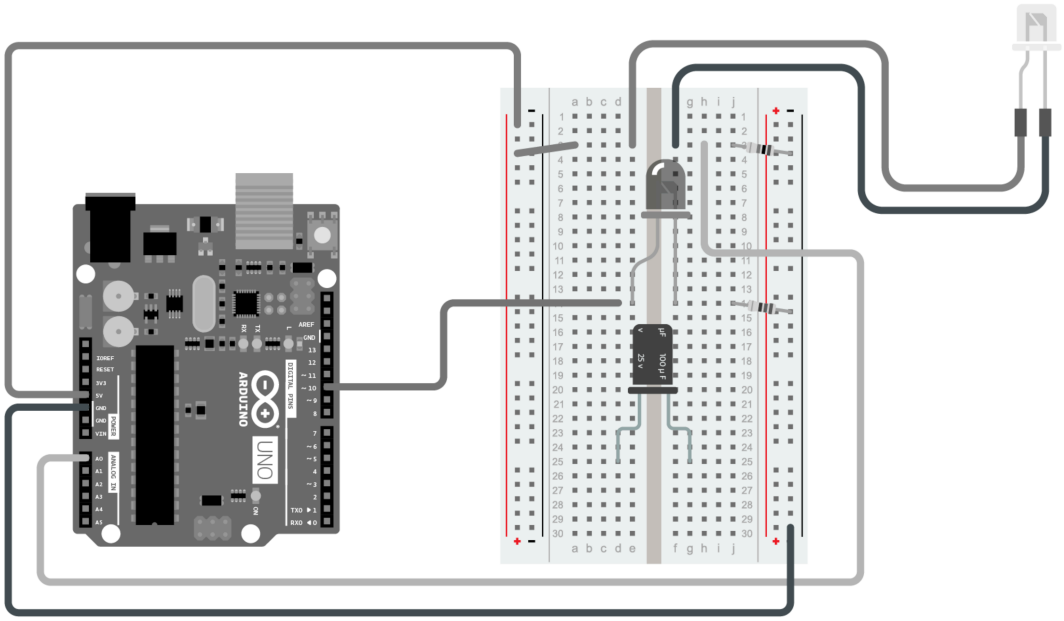
Step 1

Beginnt mit eurer Schaltung aus der Lichtsensor-Übung. Falls erforderlich, geht zurück zur Lichtsensor-Übung und baut den Schaltkreis anhand der Anweisungen im Abschnitt "Aufbau der Schaltung" erneut auf.



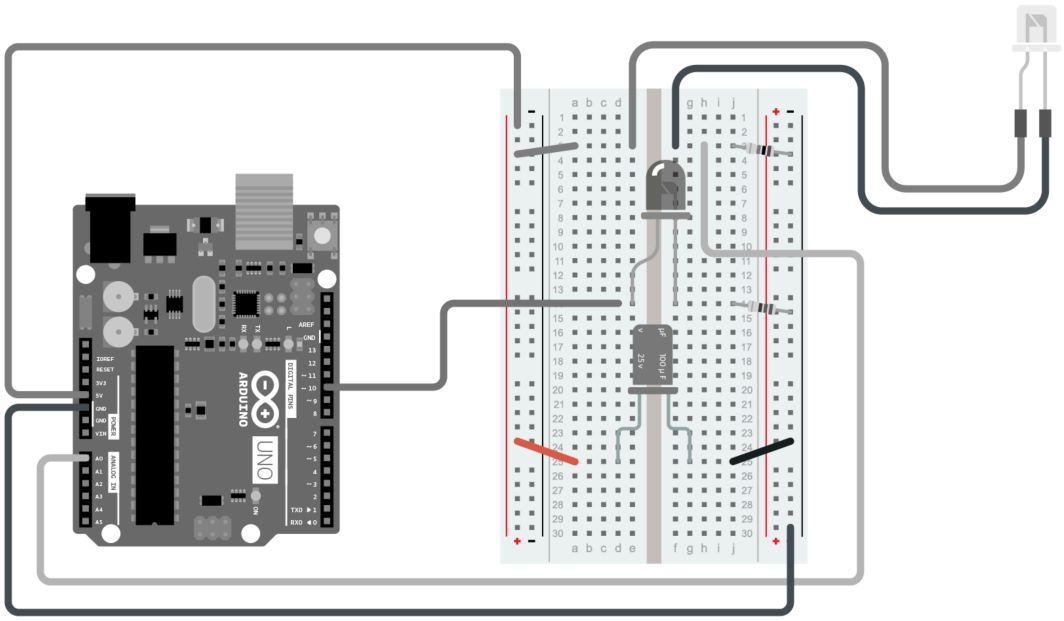
Step 2

Schließt in der Nähe der unteren Seite des Steckbretts einen 100 µf-Kondensator an. Schließt den Kondensator so an, dass sich die Anode auf der linken Seite des Steckbretts und die Kathode auf der rechten Seite des Steckbretts befindet.



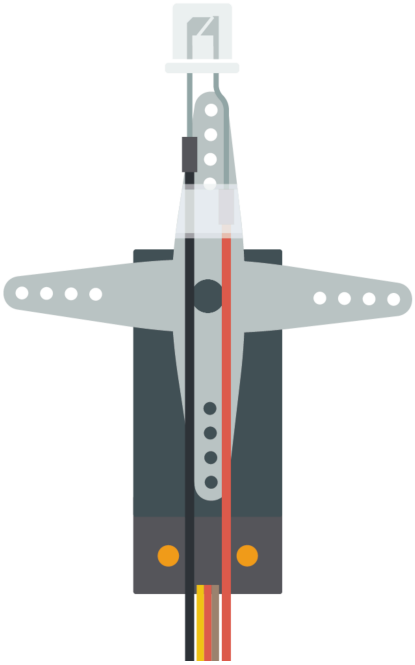
Step 3

Verwendet einen kurzen Steckverbinder, um die Anode des Kondensators mit dem positiven Anschluss auf der linken Seite des Steckbretts zu verbinden. Verwendet einen zweiten kurzen Steckverbinder, um die Kathode mit dem negativen Anschluss auf der rechten Seite des Steckbretts zu verbinden.



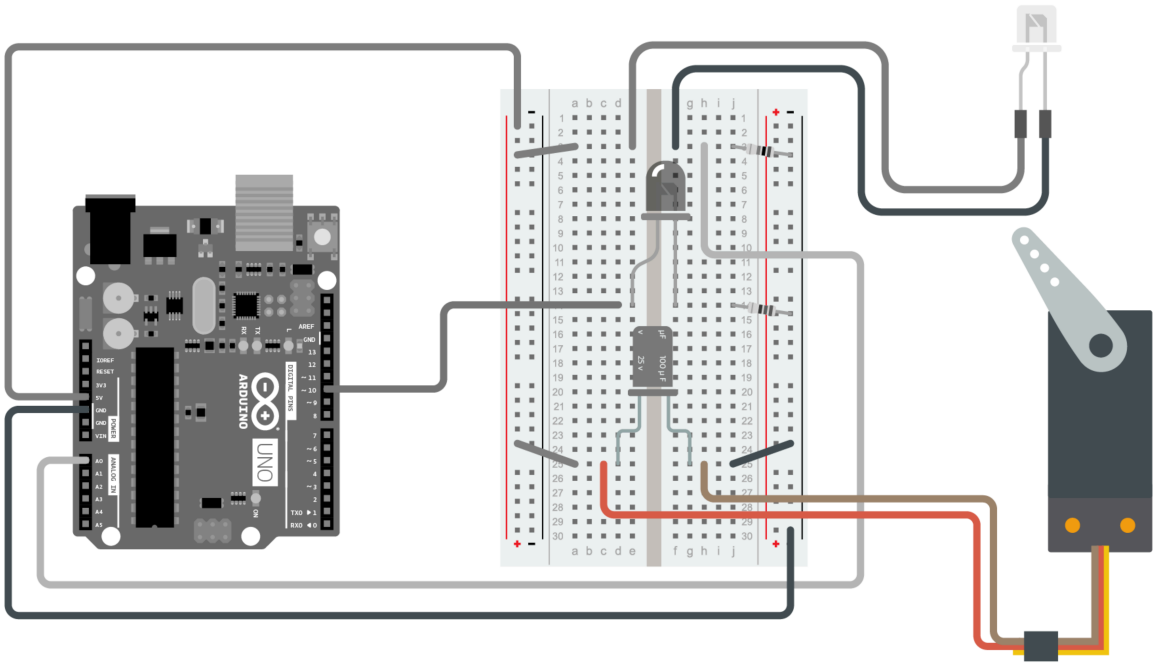
Step 4

Verwendet ein kleines Stück Klebeband, um die am Fototransistor befestigten Stecker-Buchsen-Steckverbinder an das Servohorn des Servos zu kleben. Verlegt die Steckverbinder parallel zu einem der Arme des Servohorns. Stellt sicher, dass die weiblichen Enden der Verbinder knapp über das Ende des Servohorns hinausragen. Wickelt das Klebeband um die Steckverbinder und den Arm des Servohorns. Tipp: Bevor ihr das Klebeband anbringt, faltet ihr ein Ende des Bandes auf sich selbst um, um eine Lasche zu schaffen, die nicht am Servohorn klebt. Dies hilft beim Entfernen des Bandes am Ende der Übung.



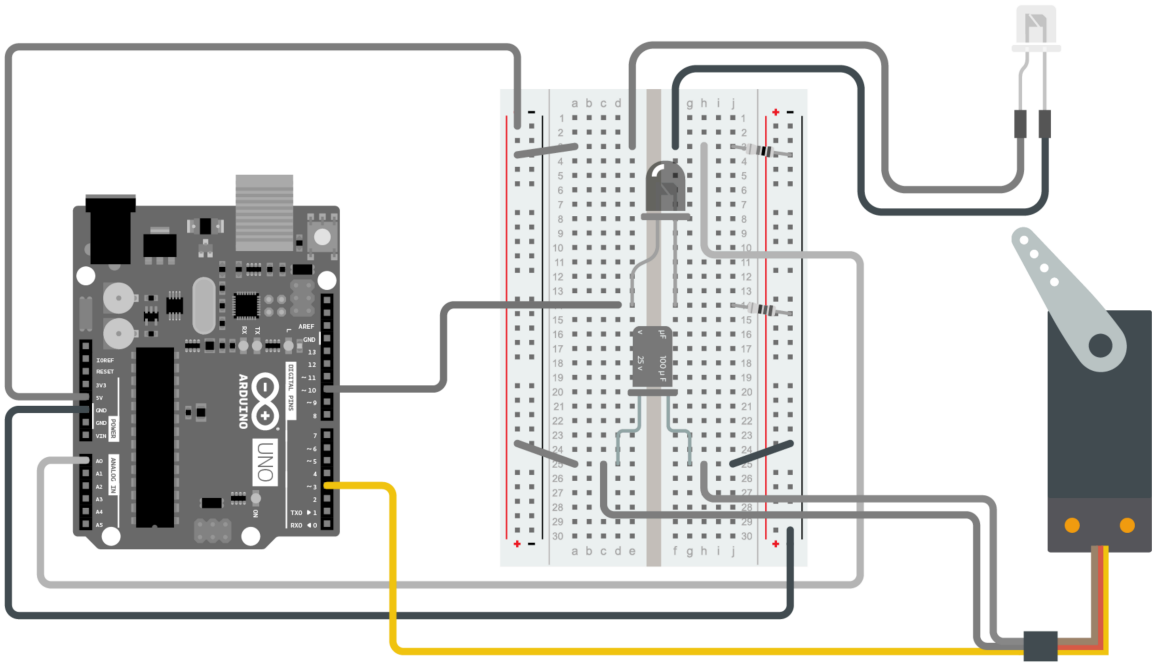
Step 5

Verwendet einen langen Steckverbinder, um den positiven Draht (rot) des Servos mit der Anode des Kondensators zu verbinden. Verwendet einen weiteren langen Steckverbinder, um die negative Leitung (schwarz) des Servos mit der Kathode des Kondensators zu verbinden.



Step 6

Verbindet den weißen Draht vom Servo mit Pin 3 auf dem Arduino UNO R3 Board mit einem weiteren langen Steckverbinder.



Code-Erstellung - Manueller Lichttaster

In dieser Übung programmiert ihr euer Lichtwellenradar so, dass es den Raum abtastet und das Umgebungslicht abbildet. Euer Lichtwellenradarsystem wird ein manueller Scanner sein, der sich bewegt, indem er Befehle von euch über den seriellen Monitor entgegennimmt.

TEACHER NOTES

Bei dieser Übung programmieren die Schülerinnen und Schüler das Arduino so, dass es die Lichtintensität des Raumes manuell abtastet, indem es Befehle vom Benutzer über den seriellen Monitor annimmt. Die Schülerinnen und Schüler erhöhen den Winkel des Servos, indem sie ein Pluszeichen (+) über den seriellen Monitor senden, und verringern den Winkel des Servos, indem sie ein Minuszeichen oder einen Bindestrich (-) über den seriellen Monitor senden. Wenn die Schülerinnen und Schüler diese Übung abschließen, werden sie diese Konzepte wiederholen:

- **Serial.available() –** Prüft, ob Informationen über serielle Kommunikation an das Arduino UNO R3 Board gesendet wurden
- **Serial.read() –** Liest den Wert der Information, die an das Arduino UNO R3 Board gesendet wurde, Byte für Byte
- **#include < library.h> –** Importiert eine installierte Bibliothek in den Sketch

- **Objekterstellung**– Benennt ein Objekt, das mit der Bibliothek verwendet werden soll
- **Konstantendeklarationen –** Deklarieren den Typ und den Namen einer Konstanten
- **servoName.attach() –** Ein Befehl aus der Servo-Bibliothek, der ein Servo-Objekt mit einem physikalischen Pin auf dem Arduino UNO R3 Board verbindet
- **servoName.write(angle) –** Ein Befehl aus der Servo-Bibliothek, der ein Servo-Objekt auf einen bestimmten Winkel bewegt

1) Startet die Arduino IDE und öffnet euren Lektion9_V1-Sketch

TEACHER NOTES

Die Schülerinnen und Schüler sollten mit dem Sketch aus Lektion 9*V1 beginnen, da diese mehr mit dem gemeinsam hat, was sie in diesem Sketch tun werden, als der Sketch aus Lektion 9*V2.

2) Speichert den Sketch unter einem neuen Namen. Aus dem Dateimenü wählt ihr **Speichern unter...** Benennt die Datei mit "Lektion9_V3_", gefolgt von euren Initialen. Klickt auf **Speichern**.

3) Nun, da ein Servo in die Schaltung eingebunden ist, müsst ihr die Servobibliothek in den Sketch importieren. Gebt den Befehl **#include < Servo.h >** ganz am Anfang eures Sketches ein.

4) Erstellt einen Namen für den Arduino-Pin, an dem der Servo befestigt ist. Der Servo sollte an Pin 3 angeschlossen werden. Schreibt im Abschnitt zur Benennung der auf dem Arduino UNO R3 Board verwendeten Pins einen Befehl, um Pin 3 **servoPin** zu nennen.

5) Ihr müsst auch den Pin benennen, an dem die blaue LED angebracht ist. Benennt Pin 10 **LEDPin**.

6) Geht zum Abschnitt für die Deklaration von Variablen. Für diesen Sketch benötigt ihr zwei neue Variablen - eine zum Speichern des Servowinkels und eine zum Speichern der Daten, die vom seriellen Monitor eingegeben werden. Erstellt eine ganzzahlige Variable mit dem Namen **servoAngle** und setzt sie gleich 0. Dadurch wird der Startwert des Servos auf 0 Grad gesetzt. Erstellt eine weitere ganzzahlige Variable namens **inputCommand** und setzt diese gleich 0.

7) Vor der Funktion **void setup()** müsst ihr ein Servo-Objekt erstellen. Benennt das Servo-Objekt **myServo**.

8) Verwendet innerhalb der Funktion **void setup()** den Befehl **pinMode**, um **LEDPin** als Outputpin festzulegen.

9) Ihr müsst auch den Befehl hinzufügen, der das Servo-Objekt mit dem an Pin 3 angeschlossenen Servo verbindet. Schreibt einen **myServo.attach()**-Befehl für den Pin mit der Bezeichnung **servoPin**.

10) Wenn der Sketch läuft, werdet ihr nicht wissen, in welcher Position der Servo startet. Fügt daher einen **myServo.write()**-Befehl hinzu, um das Servo in eine durch die Variable **servoAngle** festgelegte Startposition (0 Grad) zu bewegen. Fügt auch

eine **delay()** von 1.000 Millisekunden hinzu, um den Sketch anzuhalten, während sich das Servo in Position bewegt.

11) Die **void loop()**-Funktion des Sketches wird sich im Vergleich zum V1-Sketch ziemlich stark verändern. Es wird am Einfachsten sein, alles innerhalb der **void loop()**-Funktion zu löschen. Euer Sketch sollte diesem ähnlich sein:

12) Jetzt ist ein guter Zeitpunkt, euren Code zu überprüfen und gegebenenfalls zu debuggen.

13) Es ist an der Zeit, mit dem Neuaufbau der **void loop()**-Funktion zu beginnen. Die Schleife muss damit beginnen, zu überprüfen, ob irgendwelche Daten gesendet und im seriellen Puffer gespeichert wurden. Der Befehl **Serial.available()** prüft, ob Daten im seriellen Puffer gespeichert sind. Wenn Daten im Puffer gespeichert sind, gibt der Befehl eine 1 zurück, wenn der serielle Puffer leer ist, wird eine 0 zurückgegeben.

Der Sketch muss jedoch etwas tun, wenn Daten im seriellen Puffer gespeichert sind. Wenn ihr diesen Befehl in den Vergleichsteil eines if-Statements einfügt, kann der Sketch andere Befehle ausführen, wenn im Puffer Daten gespeichert sind. Erstellt zu Beginn der Funktion **void loop()** ein if-Statement wie dieses:

14) Wenn Daten im seriellen Puffer gespeichert sind, müssen diese Daten gelesen werden. Der Befehl **Serial.read()** betrachtet das erste Byte der Daten im seriellen Puffer. Mit anderen Worten, schaut er auf das erste Zeichen, das in den seriellen Monitor eingegeben wurde, und gibt den Bytewert dieses Zeichens zurück. Wenn der Puffer das erste Byte in der Zeile gelesen hat, löscht er es, und das nächste Byte rückt an den Anfang der Zeile. Erstellt innerhalb des if-Statements einen Befehl, der das erste Byte aus dem seriellen Puffer liest und es in der Variablen **inputCommand** speichert.

15) Ihr werdet das Arduino so programmieren, dass es zwei verschiedene Befehle vom seriellen Monitor akzeptiert - einen Befehl zur Vergrößerung des Servowinkels und einen Befehl zur Verkleinerung des Servowinkels. Ihr könnt beliebige Zeichen auswählen, um diese Befehle darzustellen (**O** für oben und **U*** für unten, **V** für größer und **K** für kleiner oder **R** für rechts und **L** für links). Vorerst werdet ihr ein Pluszeichen (+) verwenden, um den Servowinkel zu vergrößern, und ein Minuszeichen oder einen Bindestrich (-), um den Servowinkel zu verkleinern.

Das Arduino hat zwei verschiedene Verhaltensweisen, je nachdem, welcher Befehl in den seriellen Monitor eingegeben wird (+ oder -). Ihr könnt eine Switch-Case-Funktion verwenden, um jedes Verhalten zu definieren. Erstellt eine Switch-Case-Struktur, indem ihr die Variable **inputCommand** verwendet, um den Fall zu bestimmen. Denkt daran, dass die **inputCommand**-Variable einen Byte-Wert enthält, der ein einzelnes Zeichen darstellt.

16) Der erste Fall der Switch-Case-Struktur sollte für das +-Zeichen stehen, das den Servowinkel vergrößert. Definiert diesen Fall, indem ihr das +-Zeichen in einfache Anführungszeichen setzen, so dass der Byte-Wert des +-Zeichens verwendet wird. Fahrt auch damit fort, den Break-Befehl hinzuzufügen, um den Fall zu beenden.

17) Erstellt einen zweiten Fall für das Zeichen -, um den Servowinkel zu verringern. Auch hier definiert ihr die Groß-/Kleinschreibung, indem ihr das - Zeichen in einfache Anführungszeichen setzt. Fügt auch für diesen Fall den schließenden Breakbefehl hinzu.

18) Für den + Fall sollte der Servowinkel um 10 Grad zunehmen. Der Servowinkel wird in der Variablen **servoAngle** gespeichert. Schreibt innerhalb des + Falls einen Befehl, um die Variable **servoAngle** um 10 Grad zu erhöhen.

19) Aber was passiert, wenn sich der Servo bereits in seinem maximalen Winkel von 180 Grad befindet, wenn der Befehl + in den seriellen Monitor eingegeben wird? Der Servo kann sich nicht über 180 Grad hinaus bewegen. Ihr müsst eine Bedingung schreiben, die besagt, dass wenn die Variable **servoAngle** größer oder gleich 180 ist, der **servoAngle** auf seinen Maximalwinkel gesetzt wird, der 180 beträgt.

20) Ihr könnt auch eine visuelle Anzeige erzeugen, dass der Servo seine Grenze erreicht hat, indem die blaue LED blinkt. Verwendet **digitalWrite()**-Befehle, um die blaue LED einzuschalten, eine halbe Sekunde zu verzögern und dann die blaue LED auszuschalten. Stellt sicher, dass diese Anweisungen innerhalb des if-Statements platziert werden, wenn die Variable **servoAngle** größer oder gleich 180 ist.

21) Der Fall der Vergrößerung des Servowinkels (+) ist nun abgeschlossen. Die Argumente für eine Verkleinerung des Servowinkels werden sehr ähnlich sein. Beginnt mit dem Kopieren des Codes für die Vergrößerung des Servowinkels und fügt ihn in den Fall für die Verkleinerung des Servowinkels ein.

22) Für den Fall, dassihr den Servowinkel verkleinern wollt, müsst ihr einige Änderungen am Code vornehmen. Zunächst muss die Variable **servoAngle** um 10 verringert und nicht erhöht werden. Ändert im Befehl, der die Variable **servoAngle** festlegt, das +-Zeichen in ein --Zeichen.

23) In dem bedingten Statement sollte die Variable **servoAngle** nicht über 0 Grad hinaus abnehmen können. Ändert die Bedingung dahingehend, dass wenn die Variable **servoAngle** kleiner oder gleich 0 ist, der **servoAngle** auf 0 gesetzt wird. Dadurch wird die Variable innerhalb des Bereichs des Servos (0-180 Grad) gehalten.

24) Die Switch-Case-Struktur ist abgeschlossen. Überprüft und debugged euren Code, falls erforderlich.

25) Ihr habt Code geschrieben, um die Variable **servoAngle** zu definieren. Jetzt muss sich der Servo auf diesen Wert bewegen. Fügten Sie nach der schließenden Klammer der Switch-Case-Struktur einen **myServo.write()**-Befehl hinzu, um den Servo auf die Variable **servoAngle** zu bewegen. Fügt dann eine halbe Sekunde Verzögerung hinzu, um dem Servo Zeit zu geben, sich in Position zu bewegen.

Hinweis: Achtet genau darauf, in welche schließenden Klammern ihr diese Befehle setzt. Nach dem Breakbefehl des letzten Falles solltet ihr drei schließende geschweifte Klammern hintereinander haben. Die erste geschweifte Klammer ist für die Switch-Case-Struktur. Platziert die Befehle nach dieser geschweiften Klammer. Die zweite geschweifte Klammer ist für das if-Statement, das den seriellen Puffer mit dem Befehl **Serial.available()** überprüft. Die letzte geschweifte Klammer ist die schließende Klammer für die **void loop()**-Funktion.

26) Nachdem der Servo bewegt wurde, ist es an der Zeit, eine Lichtmessung mit dem Lichtsensor durchzuführen. Verwendet einen **analogRead()**-Befehl, um den an den **SensorPin** angeschlossenen Sensor abzulesen und den Analogwert in der **lightAmount**-Variablen zu speichern.

27) Zuletzt muss der Sketch den Winkel und die Lichtmessung auf dem seriellen Monitor anzeigen. Fügt Befehle zur Anzeige von Datenetiketten, der Variablen **servoAngle** und der Variablen **lightAmount** hinzu. Achtet darauf, beim Drucken der letzten Information einen **Serial.println()**-Befehl zu verwenden, damit der nächste Datensatz in einer neuen Zeile gedruckt wird.

28) Überprüft und debugged euren Code, falls erforderlich.

29) Schließt das Arduino UNO R3 Board mit dem USB-Kabel an den Computer an.

30) Klickt auf die Schaltfläche **Hochladen**, um den Sketch auf euer Arduino UNO R3 Board hochzuladen.

31) Startet den seriellen Monitor, damit ihr Befehle an den Servo eingeben und die Winkel- und Lichtmessinformationen beobachten könnt.

32) Steuert den Servo, indem ihr Plus- (+) und Minuszeichen (-) in den Eingabebereich des seriellen Monitors eingebt. Beobachtet, wie sich der Servo in 10-Grad-Schritten bewegt. Was passiert, wenn ihr mehrere Pluszeichen (+++++) oder Minuszeichen (-----) in einer Reihe eingebt? Was passiert, wenn ihr versucht, den Servo unter 0 Grad oder über 180 Grad zu bewegen?

33) Nachdem ihr euch vergewissert habt, dass euer manueller Lichttaster funktioniert, klickt ihr in der IDE auf **Speichern**, um euren Sketch zu speichern. Behaltet eure Schaltung für die nächste Übung aufgebaut.

Lichtintensitätsexperiment

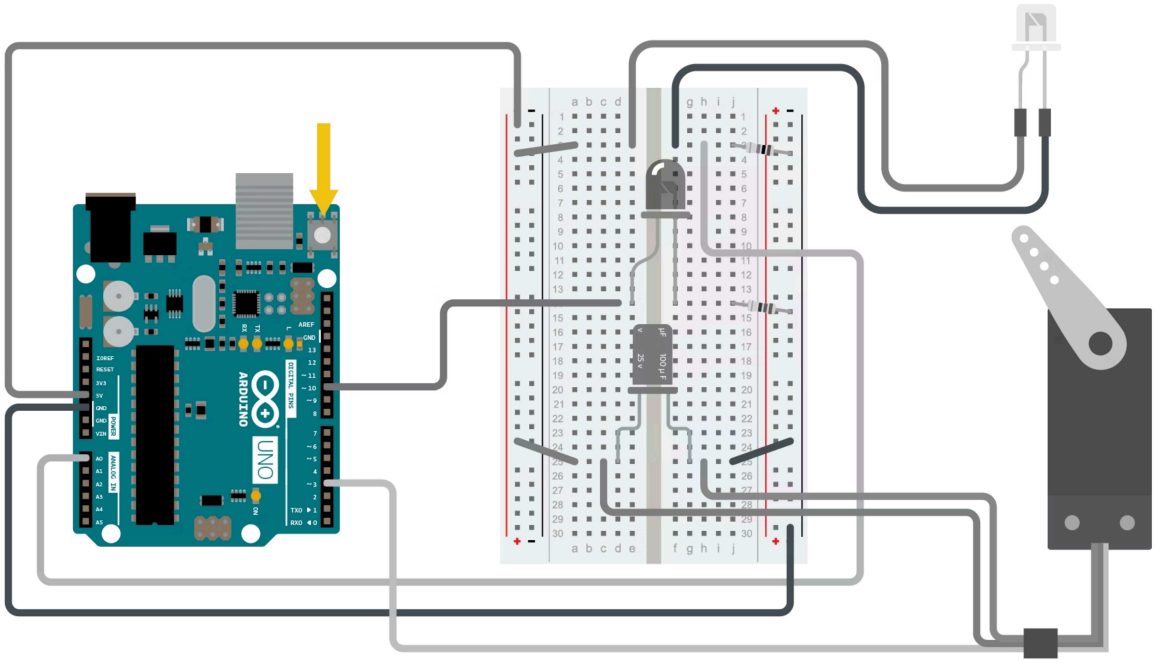
In dieser Übung verwendet ihr euren manuellen Lichtabtastkreis, um die Lichtintensität des Raumes zu messen und abzubilden. Für diese Übung benötigt ihr euer Arbeitsheft.

TEACHER NOTES

Die Schülerinnen und Schüler benötigen für diese Übung ihr Arbeitsheft, Seite 24.

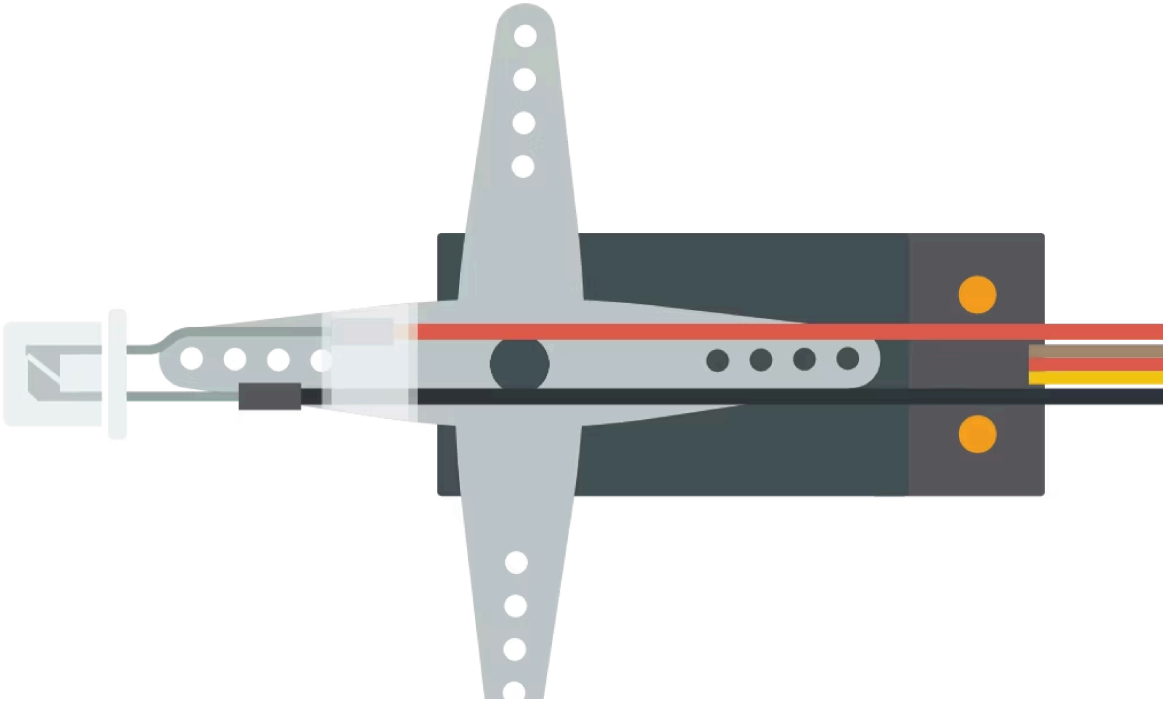
1) Falls erforderlich, schließt ihr euer USB-Kabel an, um euren manuellen Lichtscanner einzuschalten. Der Sketch eures manuellen Lichttasters sollte immer noch auf dem Arduino UNO R3 Board geladen sein.

2) Drückt die RESET-Taste auf eurem Arduino UNO R3, um das Board zurückzusetzen und den Sketch neu zu starten. Dies sollte den Servo in seine Ausgangsposition von 0 Grad bringen.

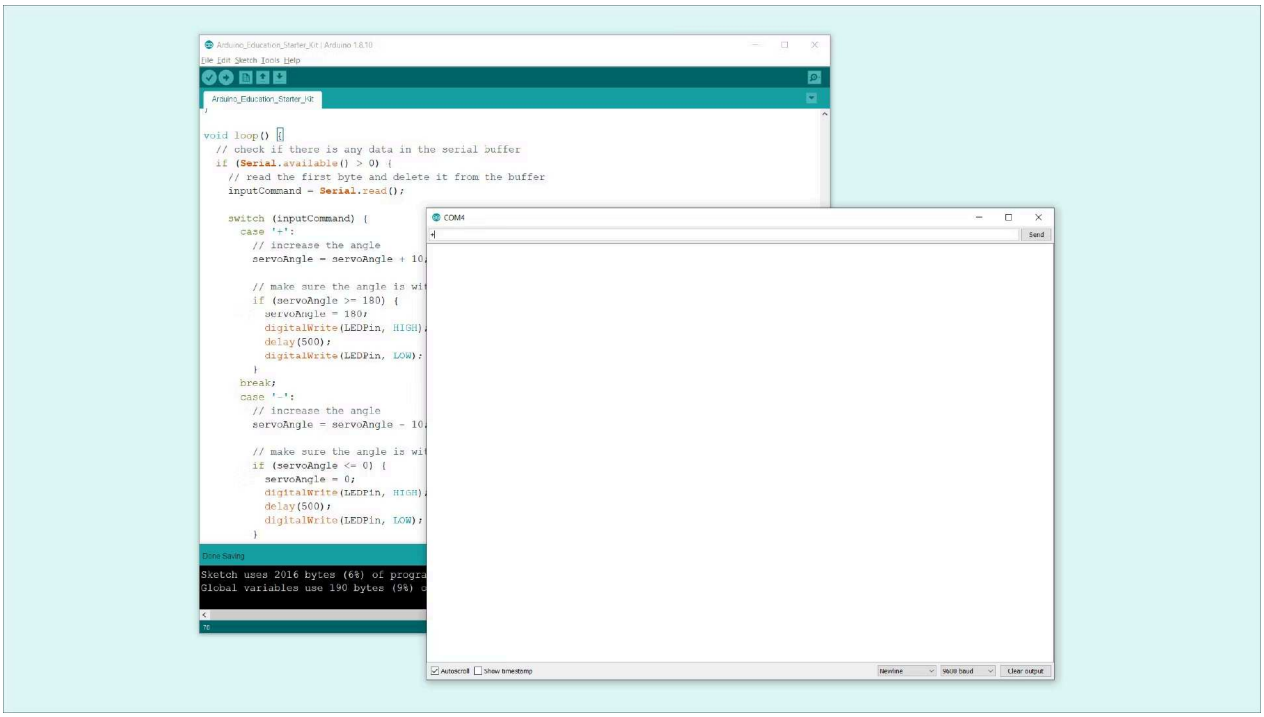


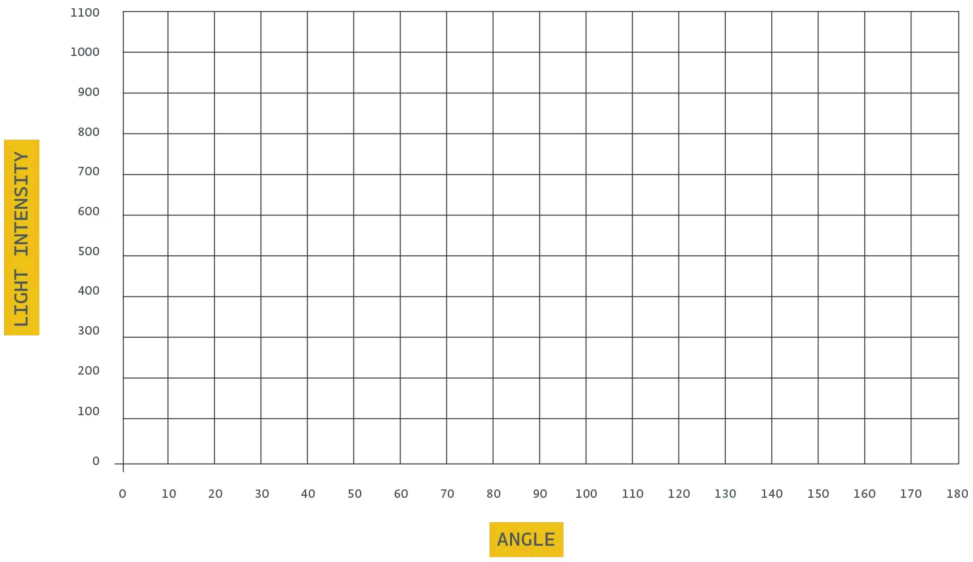
3) Startet den seriellen Monitor, damit ihr Befehle eingeben könnt, um den Winkel des Servos zu ändern und die Lichtintensitätsmessungen zu beobachten.

4) Sucht die Grafik zur Lichtintensität auf der Seite 24 eures Arbeitsheftes, Lektion 9. Ihr füllt diese Grafik aus, während ihr den Raum abtastet und die Lichtintensität unter verschiedenen Winkeln messen. Jede Winkelmessung ist entlang der *x*-Achse des Diagramms aufgeführt. Ihr zeichne den Analogwert, der die Lichtintensität darstellt, entlang der *y*-Achse des Diagramms auf.**5)** Ein Partner sollte den Servo hochhalten, so dass der Lichtsensor seitwärts gerichtet ist. Haltet den Servo so, dass sich der Servo frei drehen kann, ohne dass die Drähte im Weg sind. Es ist wichtig zu versuchen, den Servo für den Rest dieser Übung in dieser Position zu halten, um eine genaue Abtastung zu erhalten. Versucht, den Servo nicht zu bewegen.

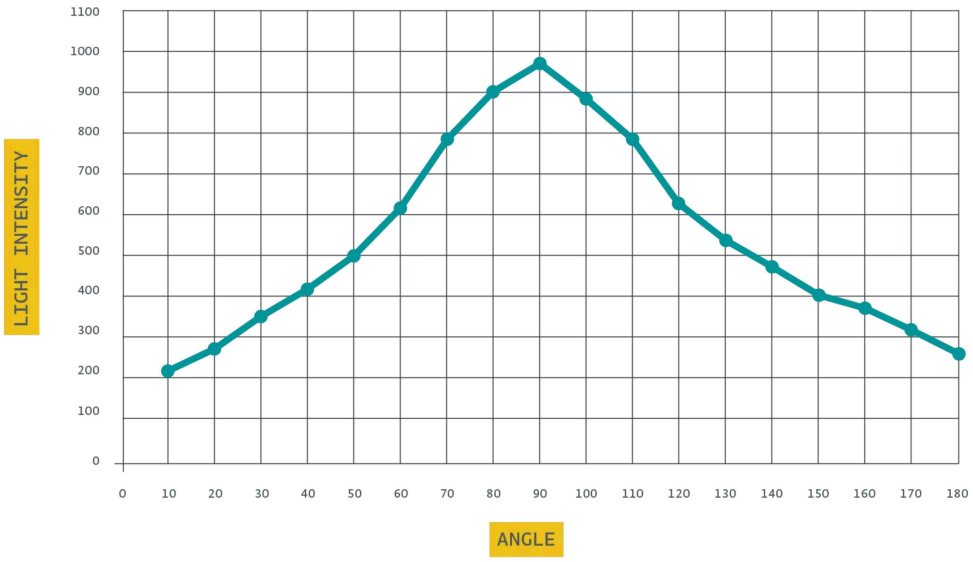


6) Ein anderer Partner sollte ein Pluszeichen (+) in den seriellen Monitor eingeben und auf **Senden** klicken, um den Servowinkel auf 10 Grad zu erhöhen. Lest die Lichtintensität vom seriellen Monitor ab und zeichnet die Intensität in der Grafik für einen Winkel von 10 Grad auf.





7) Wiederholt Schritt 6, um die Lichtintensität bei jedem Winkel, 10 bis 180 Grad, zu messen. Achtet bei den Messungen darauf, dass ihr den Servo so weit wie möglich in der gleichen Position haltet. Zeichnet jede Messung in der Grafik auf.**8)** Verbindet die Punkte, die ihr in eurem Diagramm eingezeichnet haben, mit geraden Linien. Siehe folgende Beispielgrafik.



9) Bestimmt, unter welchem Winkel die maximale Lichtintensität auftritt und wie hoch diese Lichtintensität ist. Notiert diese Werte in dem dafür vorgesehenen Feld in eurem Arbeitsheft.

10) Verwendet den seriellen Monitor, um den Servo in den Winkel zu bewegen, in dem die maximale Lichtintensität auftrat. Verwendet Plus (+) oder Minus (-) Symbole, um den Servo zu bewegen. Beachtet die Richtung, in die der Lichtsensor zeigt.

11) Erklärt in eurem Arbeitsheft, warum die maximale Lichtintensität in diesem Winkel auftrat.

12) Zieht das USB-Kabel ab, um euren Stromkreis abzuschalten. Behaltet euren Schaltkreis für die nächste Übung aufgebaut.

TEACHER NOTES

KLASSENDISKUSSION:

Lassen Sie die Klasse Situationen oder Anwendungen diskutieren, in denen eine Technologie wie ihre Schaltung eingesetzt wird oder nützlich wäre. Einige Beispiele sind hier aufgeführt:

- Solarkraftwerke folgen der Sonne, während sie sich über den Himmel bewegt. Solarreflektoren oder Solarpaneele ändern ihre Position (Winkel) so, dass sie das meiste Sonnenlicht erhalten, je nachdem, wo sich die Sonne am Himmel befindet.
- Die Internationale Raumstation verwendet ein ähnliches System, um der Sonne zu folgen und ihre Solarpaneele so auszurichten, dass sie das meiste Sonnenlicht empfangen.
- Einige Satelliten im Weltraum scannen den Weltraum auf der Suche nach bestimmten Sternenmustern. Sie benutzen diese Muster, um sich in die richtige Richtung zu orientieren.
- Radioteleskope verbringen einen Großteil ihrer Zeit damit, den Kosmos auf der Suche nach verschiedenen Arten elektromagnetischer Strahlung abzutasten. Obwohl sie nicht nach sichtbarem Licht suchen, ist der Prozess, den sie durchlaufen, ähnlich.

Testen und ändern

Manuell gesteuerte Radarsysteme messen und kartografieren bestimmte Bereiche der Reichweite des Radars. Die meisten Radarsysteme scannen jedoch automatisch hin und her. In dieser Übung verwandelt ihr euren manuellen Lichttaster in ein automatisch scannendes Radarsystem.

Ändern des Codes - Lichtwellenradar

Bei dieser Übung wird die gleiche Schaltung wie beim manuellen Lichttaster verwendet. Der Code muss jedoch so modifiziert werden, dass der Servo automatisch

hin und her bewegt wird, um den Raum zu scannen. Für diese Übung benötigt ihr euer Arbeitsheft, Seite 25.

TEACHER NOTES

In dieser Übung wiederholen die Schülerinnen und Schüler diese Kodierungskonzepte:

- **Aufgerufene Funktion –** Ein Code-Abschnitt, der eine bestimmte Aufgabe abschließt und vom Hauptteil des Programms aus mehrfach aufgerufen werden kann
- **for(Initialisierung; Bedingung; Schrittweite){} –** Erzeugt eine Schleife, die eine bestimmte Anzahl von Malen wiederholt wird

Die Schülerinnen und Schüler benötigen für diese Übung ihr Arbeitsheft.

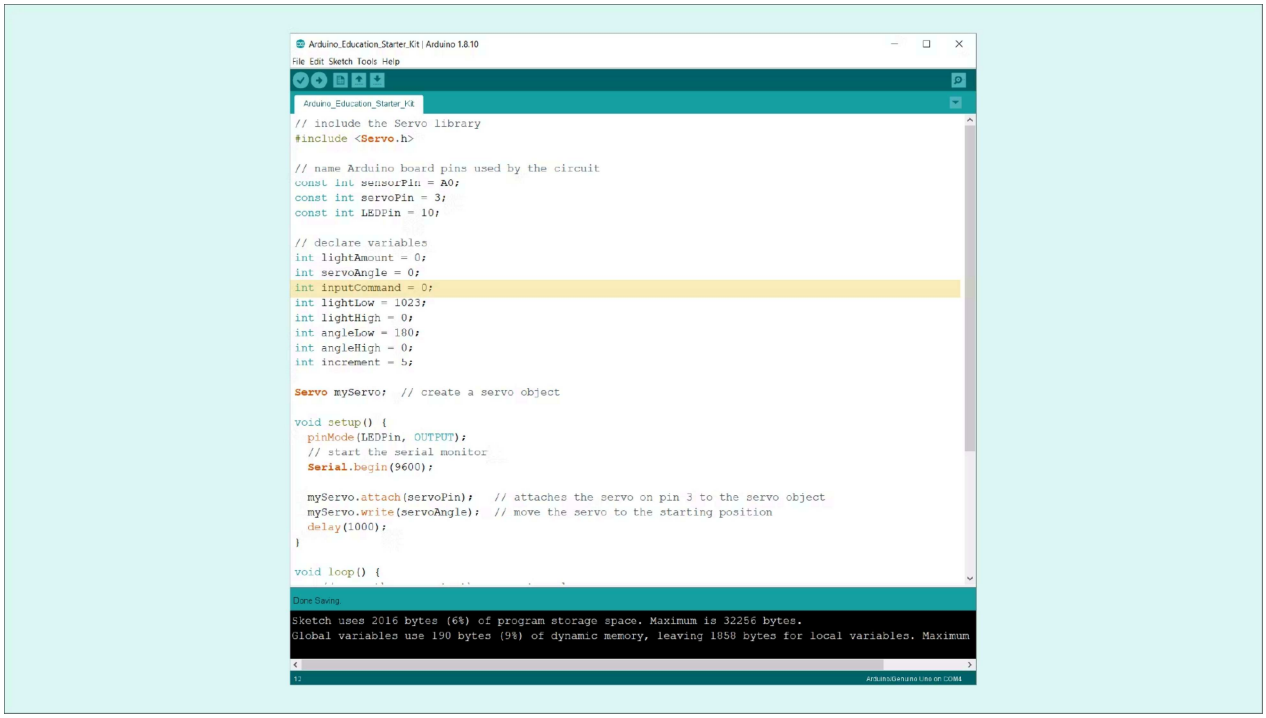
1) Startet die Arduino IDE und öffnet euren Lektion9_V3-Sketch.

2) Speichert die Skizze unter einem neuen Namen. Wählt im Menü Datei die Option **Speichern unter...** Benennte die Datei "Lektion9_V4_", gefolgt von euren Initialen. Klickt auf **Speichern**.

3) Zuerst müsst ihr mehrere neue Variablen erstellen. Oben in dem Sketch, wo ihr eure Variablen deklariert, erstellt ihr die folgenden Variablen und setzt sie auf die gezeigten Werte.

Name der Variablen	Wert	Zweck
lightLow	1023	Verfolgt das niedrigste vom Lichtsensor gemessene Lichtniveau.
lightHigh	0	Verfolgt das höchste vom Lichtsensor gemessene Lichtniveau.
angleLow	180	Verfolgt den Winkel, bei dem die geringste Lichtstärke gemessen wird.
angleHigh	0	Verfolgt den Winkel, in dem die höchste Lichtstärke gemessen wird.
increment	5	Legt den Wert fest, um den sich der Winkel bei jeder Messung erhöht oder verringert.

4) Da ihr keine Befehle über den seriellen Monitor eingeben werdet, benötigt ihr die Variable **inputCommand** nicht. Löscht den Befehl, der die Variable **inputCommand** deklariert.



5) Die Funktion **void setup()** kann so bleiben, wie sie ist, ohne Änderungen. Wechselt zur Funktion **void loop()**. Der Anfangsteil der Schleife wird nicht benötigt, einschließlich der gesamten Switch-Case-Funktion, da ihr keine Befehle über den seriellen Monitor eingeben müsst. Löscht alles in der **void loop()**-Funktion mit Ausnahme der folgenden Befehle:

- Der Befehl **myServo.write(servoAngle)**, der den Servowinkel einstellt
- Der Befehl **delay(500)**, der bewirkt, dass der Sketch wartet, bis der Servo in Position geht
- Der Befehl, der die Variable **lightAmount** unter Benutzung des Befehls **analogRead(sensorPin)** definiert
- Alle **Serial.print**-Befehle

Hinweis: Da sich alle Befehle, die ihr behaltet, innerhalb eines if-Statements befanden, das ihr gelöscht habt, müsst ihr auch eine der beiden schließenden Klammern am Ende des Sketches löschen.

Eure **void loop()**-Funktion sollte wie folgt aussehen:

6) Um des Servo automatisch abtasten zu lassen, könnt ihr eine for-Schleife verwenden, bei der sich der Winkel des Servos mit jedem Durchlaufen der Schleife vergrößert. Erinnert euch daran, dass eine for-Schleife eine Reihe von Befehlen eine bestimmte Anzahl von Malen wiederholt. Ein Zähler wird verwendet, um die Anzahl der Male zu zählen, die der Befehlssatz ausgeführt wurde, und wenn der Zähler beendet ist, endet die Schleife.

Die Struktur einer for-Schleife umfasst:

- Die Initialisierung, die eine Steuervariable für die Schleife erzeugt.
- Die Bedingung, die bestimmt, ob sich die Schleife wiederholen soll.
- Die Schrittweite, die die Steuervariable um einen bestimmten Betrag ändert.

  for (Initialisierung; Bedingung; Schrittweite) { auszuführende Befehle }

Erstellt eine for-Schleife und verwendet die Variable **servoAngle** als Steuervariable:

- Für die Initialisierung setzt ihr **servoAngle** zu Beginn des Scans auf 0.
- Für diese Bedingung sollte die for-Schleife laufen, solange der **servoAngle** kleiner oder gleich 180 ist.
- Für die Schrittweite erhöht ihr die Variable **servoAngle** um die **increment**-variable.

7) Nach Fertigstellung bewegt die for-Schleife den Servowinkel von 0 bis 180 Grad in Schritten von 5 Grad, wobei bei jedem Schritt Lichtmessungen vorgenommen werden. Um den Servo wieder in seine Ausgangsposition bei 0 Grad zu bringen, könnt ihr eine weitere for-Schleife verwenden. Beginnt mit dem Kopieren der ersten for-Schleife und fügt sie direkt unter der ersten for-Schleife ein.

8) Die zweite für Schleife sollte bewirken, dass der Servo rückwärts arbeitet. In der zweiten für Schleife ändert ihr:

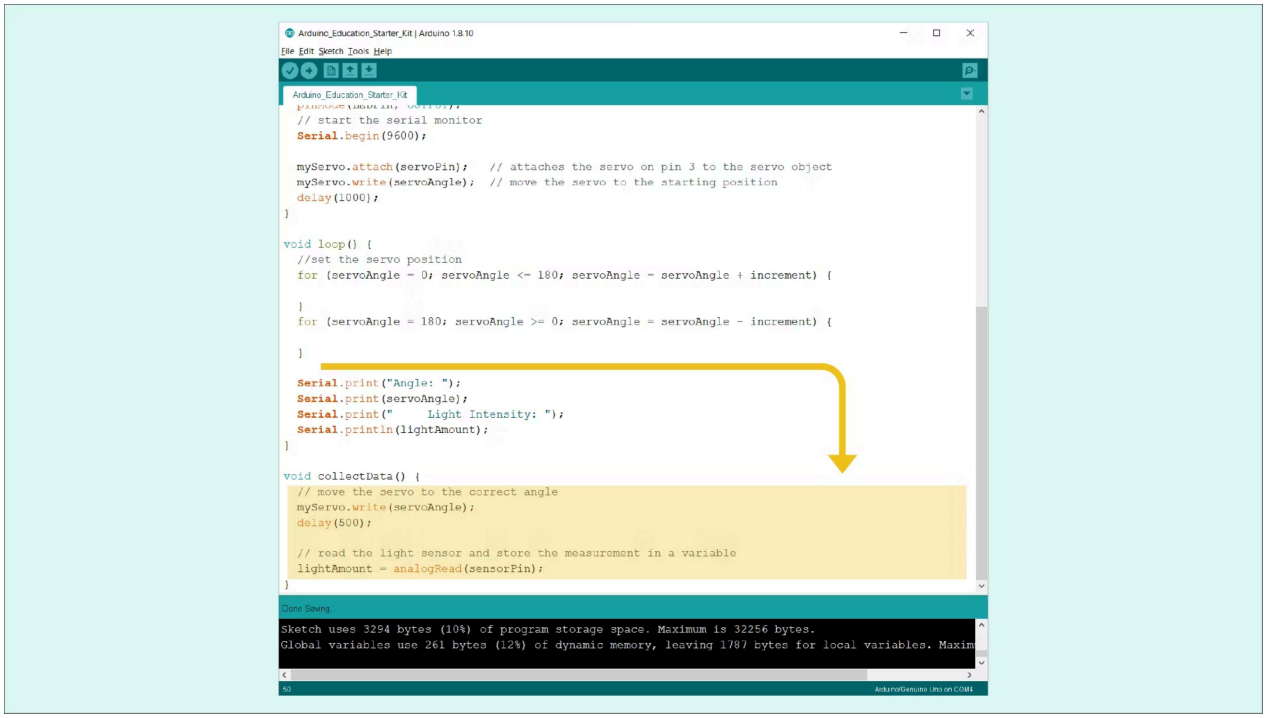
- Die Initialisierung, um die Variable **servoAngle** auf 180 Grad einzustellen.
- Die Bedingung bis zu der die Variable **servoAngle** größer oder gleich 0 ist.
- Die Schrittweite zur Verringerung des **ServoAngle** um die Schrittweitenvariable.

9) Bevor ihr weitergeht, nehmt ihr euch einen Moment Zeit und überprüft euren Sketch. Falls nötig, debugged ihr euren Code.

10) Innerhalb der for-Schleifen werden die gleichen Aufgaben ausgeführt - den Servo bewegen, eine Messung vom Lichtsensor durchführen und die Daten auf dem seriellen Monitor ausgeben. Wenn jedoch dieselben Befehle zweimal ausgeführt werden, einmal in jeder for-Schleife, wird der Code ineffizient. Stattdessen könnt ihr eine aufgerufene Funktion verwenden und die Befehle nur einmal schreiben.

Geht nach der schließenden Klammer der Funktion **void loop()** ganz nach unten in euren Sketch. Als Erstes muss der Servo in Position gebracht und die Daten vom Lichtsensor gesammelt werden. Erstellt eine neue Funktion namens **collectData()**. Denkt daran, dass die Funktion mit dem Schlüsselwort **void** beginnen muss.

11) Verschiebt die Befehle zum Bewegen des Servos (einschließlich der 500 Millisekunden Verzögerung) und zum Speichern der Lichtmessung vom Lichtsensor in der Variablen **lightAmount** aus der **void loop()**-Funktion heraus. Um die Befehle zu verschieben, könnt ihr den Code ausschneiden und einfügen oder ziehen und ablegen.



12) Da das Radar das Licht im Raum abtastet, muss es die höchsten und niedrigsten Lichtmessungen speichern und in welchen Winkeln diese Lichtmessungen stattfinden. Dazu erstellt ihr ein if-Statement, das die Messung des Lichtsensors (gespeichert in der Variable **lightAmount**) mit dem Wert vergleicht, der in der **lightHigh**-Variablen gespeichert ist, die die höchste Lichtmessung verfolgt. Wenn die Variable **lightAmount** größer als die Variable **lightHigh** ist, dann speichert ihr die Variable **lightAmount** als neue Variable **lightHigh**.

Speichert auch die Variable **servoAngle** als neuen Wert für die Variable **angleHigh**, die den Winkel verfolgt, bei dem die höchste Lichtmessung vorgenommen wurde.

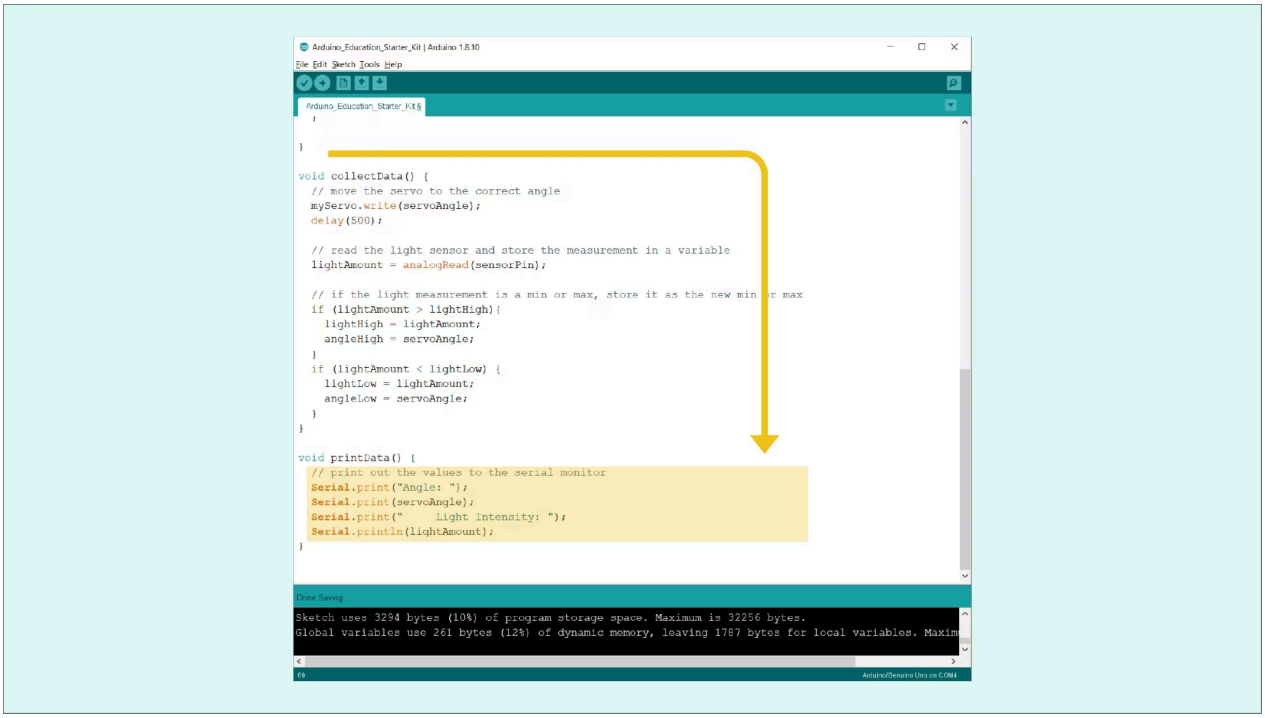
13) Um zu prüfen, ob die niedrigste Lichtmessung vorliegt, erstellt ihr ein zweites if-Statement, das die Messung des Lichtsensors (Variable **lightAmount**) mit dem in der Variablen **lightLow** gespeicherten Wert vergleicht. Wenn die Variable **lightAmount** kleiner als die Variable **lightLow** ist, dann speichert ihr die Variable **lightAmount** als neue Variable **lightLow**. Speichert auch die Variable **servoAngle** als neuen Wert für die Variable **angleLow**, die den Winkel verfolgt, bei dem die niedrigste Lichtmessung vorgenommen wurde.

14) Dies sollte den **collectData()**-Befehl vervollständigen. Zurück in der Funktion **void loop()** müsst ihr die Befehle zum Aufruf dieser Funktion hinzufügen. Erinnert euch daran, dass ihr zum Aufrufen einer Funktion einfach den Namen der Funktion eingebt, gefolgt von offenen und geschlossenen Klammern (und einem Semikolon, um den Befehl zu beenden). Gebt in jeder der for-Schleifen einen Befehl zum Aufruf der Funktion ein.

15) Überprüft euren Code und debugged ihn wenn nötig.

16) Jedes Mal, wenn die beiden for-Schleifen durchlaufen werden, müssen die Daten auf dem seriellen Monitor angezeigt werden. Da die Befehle für die Anzeige der Daten in beiden Schleifen die gleichen sind, könnt ihr eine weitere aufgerufene Funktion verwenden. Erstellt nach der Funktion **collectData()** eine weitere Funktion namens **printData()**. Vergesst nicht, dass sie mit dem Schlüsselwort **void** beginnen muss.

17) Verschiebt die Codezeilen, die die Variablen **servoAngle** und **lightAmount** drucken (einschließlich ihrer Datenbeschriftungen), von der Funktion **void loop()** in die neue Funktion, die ihr gerade erstellt habt.



18) Neben der Variablen **servoAngle** und der Variablen **lightAmount** gibt es mehrere andere Variablen, die angezeigt werden können. Jede dieser Variablen sollte auf derselben Zeile des seriellen Monitors angezeigt werden. Ändert den Befehl **Serial.println()** in einen **Serial.print()**-Befehl, um den Zeilenumbruch zu entfernen.

19) Fügt der Funktion **printData()** Befehle hinzu, die die folgenden Daten zusammen mit ihren Datenetiketten drucken:

- **lightHigh -** Die Variable, die die höchste gemessene Lichtintensität enthält
- **angleHigh -** Die Variable, die den Winkel angibt, bei dem die höchste Lichtintensität auftritt
- **lightLow -** Die Variable, die die niedrigste gemessene Lichtintensität enthält
- **angleLow -** Die Variable, die den Winkel angibt, bei dem die geringste Lichtintensität auftritt

Eure **printData()**-Funktion sollte ähnlich wie diese aussehen:

Hinweis: Stellt sicher, dass ihr einen **Serial.println()**-Befehl als letzten Befehl verwendet, damit der nächste Datensatz in einer separaten Zeile angezeigt wird.

20) Geht zurück zur Funktion **void loop()** und fügt Funktionsaufrufe in jeder for-Schleife hinzu, um die Funktion **printData()** aufzurufen.

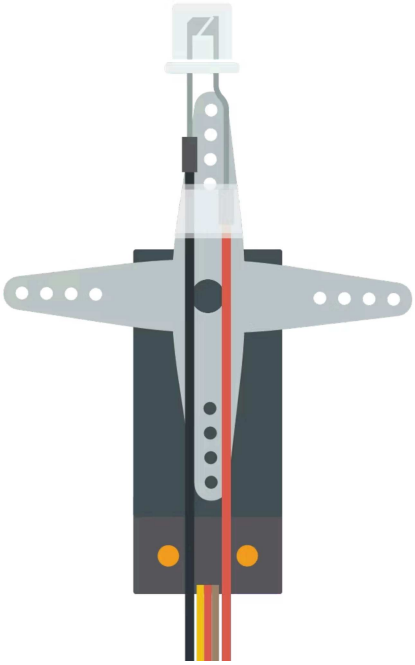
21) Euer Code sollte nun vollständig sein. Überprüft euren Code und debugged ihn gegebenenfalls.

22) Schließt das Arduino UNO R3 Board mit dem USB-Kabel an den Computer an.

23) Klickt auf die Schaltfläche **Hochladen**, um euren Sketch auf euer Arduino UNO R3 Board hochzuladen.

24) Startet den seriellen Monitor, damit ihr die Daten beobachten könnt, während das Lichtwellenradar hin und her bewegt wird.

25) Ein Partner sollte den Servo hochhalten, sodass der Lichtsensor seitwärts ausgerichtet ist. Haltet den Servo so, dass sich das Servo frei drehen kann, ohne dass die Drähte im Weg sind. Versucht, den Servo in dieser Position zu halten.

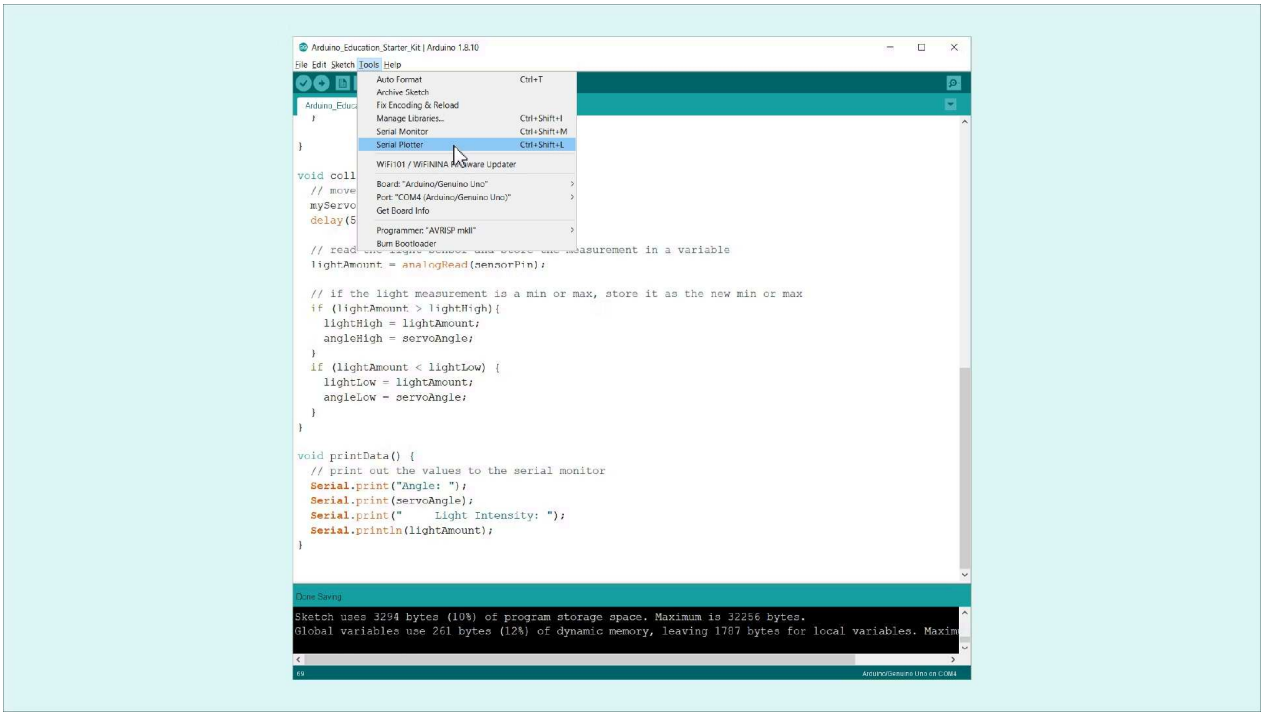


26) Nun, da alles in Position ist, drückt ihr die RESET-Taste auf dem Arduino UNO R3.

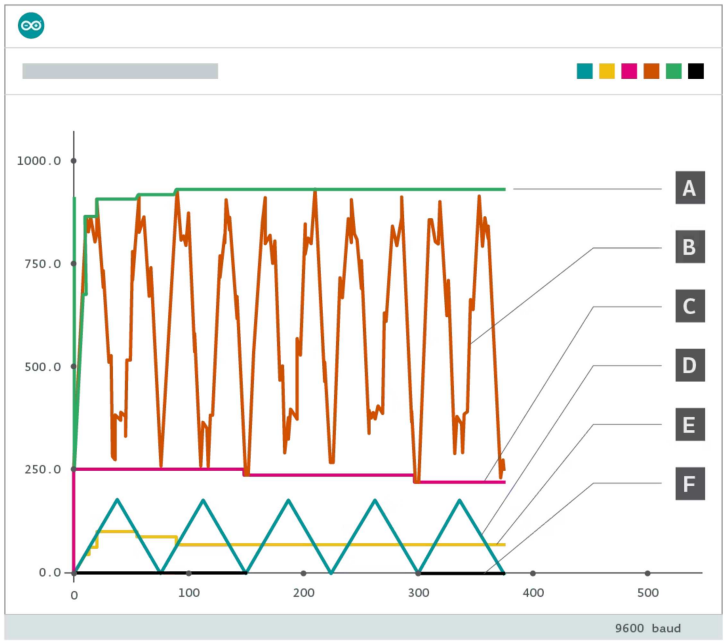
27) Beobachtet, wie das Lichtwellen-Radarsystem einige Abtastungen hin und her ausführt und dabei die Lichtintensität misst.

28) Nachdem das Radar einige Abtastungen durchgeführt hat, notiert ihr den maximalen Lichtintensitätswinkel und die maximale Lichtintensität auf Seite 25 der Lektion 9 eures Arbeitsheftes. Notiert auch den minimalen Lichtintensitätswinkel und die minimale Lichtintensität.

29) Neben dem seriellen Monitor gibt es ein weiteres Werkzeug, das für die Überwachung von Daten aus einem Sketch hilfreich ist. Schließt den seriellen Monitor. Während ihr den Servo immer noch in der gleichen Position haltet und euer Sketch noch läuft, geht ihr in der Arduino IDE zum Menü **Extras** und wählt den **seriellen Plotter**.



30) Während das Radar weiter hin- und herschweift, zeichnet der serielle Plotter die sechs Variablen auf, die über die serielle Kommunikation ausgegeben werden. Jede Linienfarbe stellt eine andere Variable dar. Drei der Linien in der Nähe des unteren Randes des Diagramms verfolgen den Servowinkel, den minimalen Lichtintensitätswinkel und den maximalen Lichtintensitätswinkel. Die Linien liegen zwischen 0 und 180 auf der y-Achse des Diagramms. Die anderen drei Linien verfolgen die Lichtintensität, die minimale Lichtintensität und die maximale Lichtintensität.



A) Maximale Lichtintensität B) Lichtintensität C) Minimale Lichtintensität D) Servowinkel E) Maximaler Lichtintensitätswinkel F) Minimaler Lichtintensitätswinkel

TEACHER NOTES

KLASSENDISKUSSION

Wenn möglich, projizieren Sie eines der Diagramme mit den Daten eines Schülers. Falls dies nicht möglich ist, lassen Sie die Schülerinnen und Schüler ihr eigenes Diagramm betrachten und diskutieren Sie die Ähnlichkeiten und Unterschiede im Raum. Besprechen Sie jede der gezeichneten Linien, was sie darstellen und warum sie so geformt sind, wie sie sind. Hier sind ein paar Diskussionsfragen, die es zu berücksichtigen gilt:

- Warum sieht die Servowinkel-Linie wie ein Bündel von Dreiecken aus?
- Warum ist die Linie der Lichtintensität gezackt, während die Linien der maximalen Lichtintensität und der minimalen Lichtintensität ziemlich glatt sind?
- Unter welchem Winkel tritt die maximale Lichtintensität auf?
- Der minimale Lichtintensitätswinkel kann den Spitzen des Servowinkels oder den Tälern des Servowinkels folgen oder in einigen Fällen zwischen den Spitzen und den Tälern hin und her springen. Erklärt, warum dies geschieht.

31) Während das Radar den Raum weiter abtastet, benutzt ihr ein Buch oder einen anderen Gegenstand, um einen Schatten auf das Radar zu werfen.

Beobachtet die Grafik im Serienplotter, um zu sehen, wie sich die Grafik dadurch verändert.

32) Wenn die Beobachtung der Daten beendet ist, klickt ihr in der Arduino IDE auf **Speichern**, um euren Sketch zu speichern. Schließt alle Fenster der Arduino IDE.

33) Reinigt euren Arbeitsbereich und lagert alle Geräte in einem geeigneten Lagerbereich.

TEACHER NOTES

WEITERE ÜBUNGEN

- **Die blaue LED** – In der letzten Übung wurde eine blaue LED in die Schaltung eingebaut, aber in dem endgültigen Sketch nicht verwendet. Lassen Sie die Schülerinnen und Schüler Ideen sammeln, wie sie die blaue LED in den Sketch einbauen könnten. Lassen Sie sie dann den Code nach Bedarf ändern, um die blaue LED so zu programmieren, dass sie die Aufgabe erfüllt.

- **Code-Bereinigung** – Lassen Sie die Teilnehmer zu einigen der anderen Sketche zurückgehen, die sie im Kurs ausgefüllt haben, und nach Stellen suchen, an denen aufgerufene Funktionen verwendet werden könnten, um den Code effizienter zu machen. Lassen Sie sie eine Kopie jedes Sketches speichern und den Code so ändern, dass er aufgerufene Funktionen enthält.

- **Beispielsketche** – Im Laufe dieses Kurses haben die Schülerinnen und Schüler einige der vielen Beispielsketche untersucht, die durch die IDE Arduino zur Verfügung gestellt wurden. Lassen Sie sie andere Beispielsketche untersuchen, um zu sehen, was mit ihrem Arduino UNO R3 Board sonst noch möglich ist. Die meisten Sketche enthalten Code-Kommentare, die grundlegende Anweisungen zum Einrichten der Schaltung für den Sketch geben. Sie können auch den folgenden Link verwenden, um Schritt-für-Schritt-Anleitungen für die Beispielsketche zu sehen. **https://www.arduino.cc/en/Tutorial/BuiltInExamples**

- **Eingabe eines Strings** – In der Übung Manueller Lichttaster konnten die Schülerinnen und Schüler den Winkel des Servos durch Eingabe von Befehlen in den seriellen Monitor steuern. Der Winkel des Servos wurde durch Plus- und Minuszeichen um 10 Grad verändert. Die Schülerinnen und Schüler könnten sich fragen, ob es möglich ist, den Servo durch Eingabe einer Winkelmessung auf einen bestimmten Winkel einzustellen. Es ist möglich, aber es erfordert einen zusätzlichen Code. Mit dem folgenden Code können Sie das Servo mit den Plus- und Minuszeichen steuern, aber Sie können auch eine Winkelmessung eingeben, indem Sie ein großes M gefolgt von dem Winkelwert und einem Sternchen (*) eingeben, um das Ende der Winkelmessung anzuzeigen. Um das Servo z.B. auf 90 Grad zu bewegen, könnten Sie "M90*" eingeben. Dies wird als Datenstring bezeichnet. Erlauben Sie den Schülerinnen und Schülern, diesen Code und seine Funktionsweise zu erforschen. Sie können diesen Code direkt in einen neuen Sketch in der Arduino IDE kopieren und ihn an die Schülerinnen und Schüler verteilen. Sie sollten in der Lage sein, ihn in ihre Licht-Radar-Schaltung hochzuladen und ihn in Aktion zu sehen.