SCHEIBENWISCHER

In dieser Lektion arbeitet ihr weiter mit einem Servo, um eine Schaltung zu erstellen, die einen Scheibenwischer simuliert. Ihr fügt Steuerelemente zu eurer Schaltung hinzu, die programmiert werden, damit euer Scheibenwischer zwischen verschiedenen Modi wechseln kann. Während ihr eure Schaltung programmiert, lernt ihr neue Konzepte wie verschachtelte Bedingungen, Switch-Case-Strukturen und Schleifen kennen.

Übersicht

Habt ihr jemals das Sprichwort gehört: "Die Notwendigkeit ist die Mutter der Erfindung"? Es bedeutet, dass, wenn ein Bedürfnis oder Problem auftaucht, jemand auf eine Idee kommt, wie man diesem begegnen oder es lösen kann. Das war sicherlich der Fall bei der Erfindung, die ihr in dieser Lektion bauen und programmieren werdet - dem Scheibenwischer. Es mag wie ein einfaches Gerät erscheinen, aber denkt an die unzähligen Leben, die diese eine Erfindung wahrscheinlich durch die Verhinderung von Autounfällen gerettet hat. Aber in dieser Lektion geht es nicht nur um eine weitere Erfindung, es geht darum, wie Menschen mit dieser Erfindung umgehen.

In dieser Lektion arbeitet ihr weiter mit einem Servo, um eine Schaltung zu erstellen, die einen Scheibenwischer simuliert. Ihr fügt Steuerelemente zu eurer Schaltung hinzu, die programmiert werden, damit euer Scheibenwischer zwischen verschiedenen Modi wechseln kann. Während ihr eure Schaltung programmiert, lernt ihr neue Konzepte wie verschachtelte Bedingungen, Switch-Case-Strukturen und Schleifen kennen. Jede Übung wird sich darauf konzentrieren, neue Funktionen zu euren Scheibenwischern hinzuzufügen, so dass sich eure Scheibenwischer am Ende ein- und ausschalten, die Geschwindigkeit ändern und die Windschutzscheibe waschen können.

TEACHER NOTES

In dieser Lektion werden keine neuen elektrischen Komponenten vorgestellt. Die Schülerinnen und Schüler arbeiten weiterhin mit einem Servomotor, Tastschalter und Potentiometer. Da keine neuen elektronischen Komponenten vorgestellt werden, liegt der Schwerpunkt dieser Lektion auf der Programmierung. Es werden mehrere neue Programmierkonzepte vorgestellt, wie z. B. die bedingte Switch-Case-Struktur, verschachtelte Bedingungselemente, while-Schleifen und for-Schleifen. Die Schülerinnen und Schüler erstellen eine Scheibenwischerschaltung, die mit einem Tastschalter zwischen vier verschiedenen Modi umschaltet - aus, ein,

intervallgeschaltet und Waschmodus. Das Potentiometer wird zur Einstellung der Geschwindigkeit des intervallgeschalteten Modus verwendet.

Damit die Schülerinnen und Schüler ihren Scheibenwischer herstellen können, müssen Sie ihnen einige zusätzliche Materialien zur Verfügung stellen: * Pappreste * Scheren * Klebeband (durchsichtiges Klebeband funktioniert gut)

Wenn ihr das kreuzförmige Servohorn noch nicht mit den Servos verbunden habt, müsst ihr dies vor Beginn dieser Lektion tun. Jedes Kit enthält einen Servo, der mit einer kleinen Tasche mit Servohörnern geliefert wird. Sucht das Servohorn, das wie ein Kreuz aussieht, und setzt es auf die Abtriebswelle des Servos. Befestigt die Schraube mit einem kleinen Kreuzschlitzschraubendreher durch das Loch im Servohorn am Servo.

Servohornbefestigung

Wenn ihr das kreuzförmige Servohorn noch nicht mit den Servos verbunden habt, müsst ihr dies vor Beginn dieser Lektion tun. Jedes Kit enthält einen Servo, der mit einer kleinen Tasche mit Servohörnern geliefert wird. Sucht das Servohorn, das wie ein Kreuz aussieht, und setzt es auf die Abtriebswelle des Servos. Befestigt die Schraube mit einem kleinen Kreuzschlitzschraubendreher durch das Loch im Servohorn am Servo.

Zeitaufwand für die Lektion

Die hier aufgeführten Zeiten sind nur ein Richtwert und müssen möglicherweise angepasst werden, während Sie das Verständnis überprüfen und anpassen.

Übersicht und Fachbegriffe	3 Minuten
Blickpunkt Erfindungen	4 minutMinutenes
Basic-Scheibenwischer	
Benötigte Materialien	2 Minuten
Aufbau der Schaltung	10 Minuten
Hinzufügen des Wischerblatts	5 Minuten
Code-Erstellung - Scheibenwischer ein/aus	13 Minuten
Code-Erstellung - Wischerbewegung	10 Minuten
Code-Erstellung - Intervallgeschalteter Wischer	13 Minuten
Testen und ändern	
Modifizieren des Codes - Verbesserte Funktionalität	15 Minuten

Ändern der Schaltung - Waschmodus	3 Minuten
Ändern des Codes - Waschmodus	12 Minuten
Gesamte Lektion	**90 Minuten**

Wenn Sie diese Lektion in zwei Unterrichtsstunden absolvieren, sollten die Schülerinnen und Schüler den Abschnitt Code-Erstellung - Wischerbewegung bis zum Ende der ersten Unterrichtsstunde abschließen.

Lernziele

- Die Untersuchung der Beziehung zwischen elektrischen und magnetischen Feldern fortsetzen.
- Die Bedingungsstruktur des Switch-Case untersuchen.
- Zwischen verschiedenen Arten von Schleifen im Code unterscheiden.
- Schleifen (for und while) im Code konstruieren, um sich wiederholende Befehle zu verarbeiten.
- Den Timer des Arduino UNO R3 Boards verwenden, um die Zeit zu messen.
- Pseudocodeschritte in eine logische Reihenfolge bringen.
- Den Code zur Verbesserung seiner Benutzerfreundlichkeit und Anwändbarkeit überarbeiten.

Fachbegriffe

- **Erfordernis** - die Fähigkeit eines Objekts, seinen Gebrauch durch seine physikalischen Eigenschaften anzuzeigen
- **Schrittweite** - etwas um eine bestimmten Anzahl zu erhöhen, was zu einem Betrag führt, der höher ist, als er vorher war
- **Initialisierung** - ein Befehl, der eine Variable erstellt und ihren Anfangswert setzt
- **Schleife** - eine Programmierfunktion, die eine Reihe von Befehlen wiederholt
- **Verschachtelte Bedingung** - ein Bedingung, die innerhalb einer anderen Bedingung platziert wird; verschachtelte Bedingungen bieten die Möglichkeit, ein Ergebnis zu erzeugen, das auf einer Reihe von Entscheidungen beruht.

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.

Blickpunkt Erfindungen

Eine erfolgreiche Erfindung entwerfen

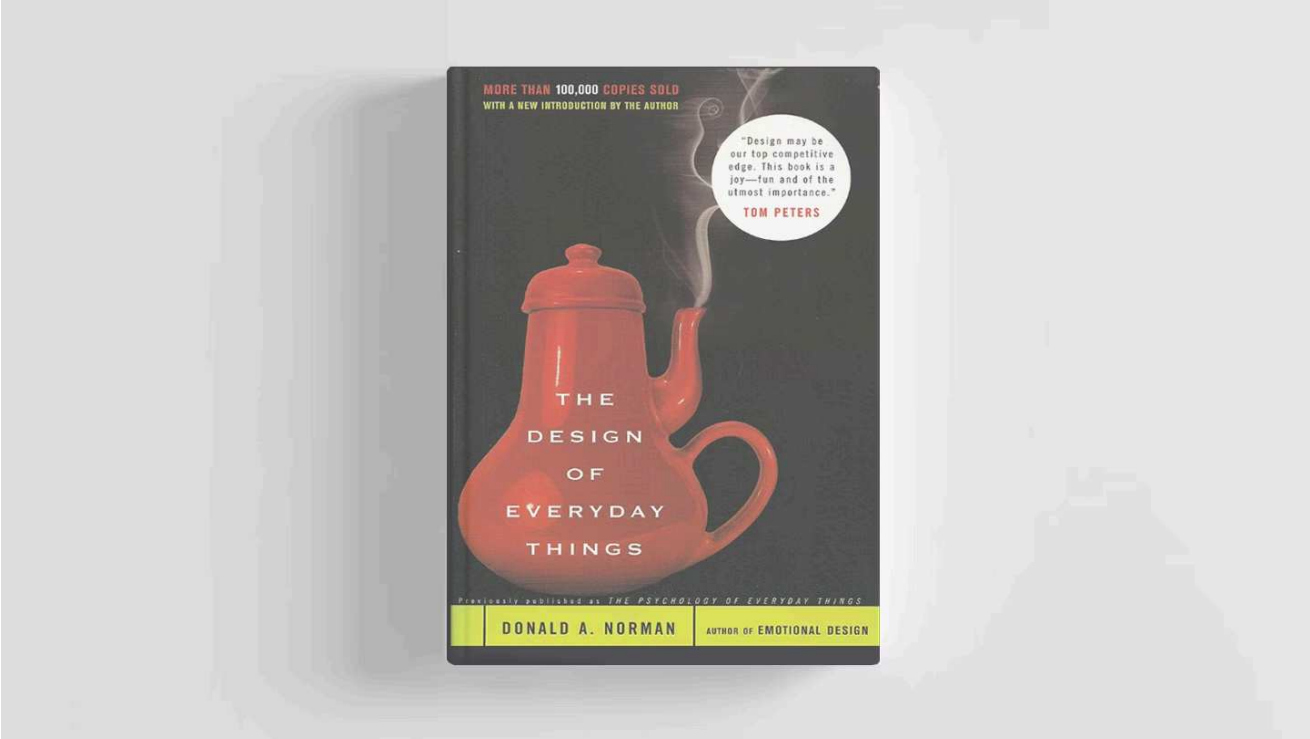
Automobile waren ein erstaunlicher technologischer Durchbruch im späten 19. Jahrhundert. Aber bei etwas schlechtem Wetter waren sie fast nutzlos. Und warum? Bei einem Regenschauer funktionierten die Maschinen einwandfrei. Das Problem war, dass der Regen auf der Windschutzscheibe die Sicht für den Benutzer fast unmöglich machte.

Mary Anderson is credited with inventing the first usable windshield wiper design in 1903. When Anderson's wipers were turned on, they continuously sprang back and forth. This was good for a heavy rain. But during a light rain, they were a little *too* good. They cleared water so quickly they ended up screeching back and forth over a dry windshield, annoying drivers.

Erst 1963 wurde der moderne intervallgeschaltete Scheibenwischer von Robert Kearns entwickelt. Kearns ließ sich vom menschlichen Auge inspirieren, das nicht kontinuierlich, sondern in Intervallen blinzelt. Die Fahrer konnten nun das Timing der Scheibenwischer so einstellen, dass sie viel weniger störend waren. Die Funktionalität wurde nicht verbessert, wohl aber die Benutzerfreundlichkeit.

Dies zeigt, dass es bei der Entwicklung einer erfolgreichen Technologie nicht nur um das Verständnis von Mechanik, Elektronik und Code geht. Es geht auch darum, die Art und Weise zu verstehen, wie Menschen mit neuer Technologie umgehen werden.

Donald Norman, der Autor des Buches The Design of Everyday Things, gilt als Pionier auf dem Gebiet des Usability Engineering. Norman erkannte, dass es für jede Technologie viele Möglichkeiten gibt, wie ein Benutzer sie theoretisch nutzen könnte; man könnte mit einem neuen Smartphone Wurfspiele spielen oder mit ihm im Dreck wühlen. Aber wie Benutzer es tatsächlich nutzen werden, hängt vom Design ab. Erfordernis ist die Fähigkeit eines Objekts, seine Nutzung durch seine physikalischen Eigenschaften anzuzeigen. Theoretisch könnte man eine Computermaus auf den Boden legen und mit den Füßen bedienen, aber das Design ist für eine menschliche Hand optimiert.



Übrigens hat der Erfinder der Computermaus, Douglas Engelbart, viel darüber nachgedacht, wie Menschen mit Werkzeugen interagieren. Er hielt seine Ideen zu diesem Thema für wichtiger als alle seine Erfindungen.

In den 1950er Jahren erkannte Engelbart, dass sich der technologische Fortschritt beschleunigte. Dies würde unweigerlich zu raschen Veränderungen in unserer Welt führen. Menschliche Gesellschaften brauchen jedoch Zeit, um sich dem Wandel anzupassen. Engelbart befürchtete, dass der Wandel zu schnell erfolgen und ernsthafte Probleme verursachen würde. Er glaubte, die Lösung lag in der Zusammenarbeit, um Veränderungen vorauszusehen und sich auf sie vorzubereiten. Durch Zusammenarbeit könnten wir unser Fachwissen kombinieren und unsere kollektive Intelligenz erhöhen. Leisten wir dabei gute Arbeit? Das ist immer noch eine ungelöste Frage.

Hinweis: 1968 hielt Douglas Engelbart eine Live-Demonstration mit dem Titel "Die Mutter aller Demos", in der er viele Merkmale moderner Computer zeigte, darunter die Verwendung von Fenstern, einer Computermaus, Grafiken, Videokonferenzen, Textverarbeitung und Hypertext. Ihr könnt euch "Die Mutter aller Demos" unter dem untenstehenden Link ansehen:

https://dougengelbart.org/content/view/209/

In unserer modernen Ära hat vielleicht kein technologischer Innovateur mehr Veränderungen in unserer Gesellschaft bewirkt als Steve Jobs. Zusammen mit Steve

Wozniak hat Jobs die Computerfirma Apple gegründet.

Apple produzierte den ersten erschwinglichen Heimcomputer, den Apple II. Später brachte es den iMac auf den Markt, dem viele den Verdienst zuschreiben, das Web für die Öffentlichkeit benutzerfreundlich zu machen. Noch später brachte Apple das iPhone auf den Markt. Obwohl es bereits andere Smartphones gab, war das iPhone das erste Touch-basierte Gerät, das für Computer und das Surfen im Internet optimiert war. Seitdem haben wir unsere Telefone nicht mehr auf die gleiche Weise benutzt.

Einige glauben, dass Jobs' Beitrag zur Gesellschaft überbewertet wird. Wozniak selbst weist darauf hin, dass Jobs kein brillanter Programmierer oder Hardware-Designer war. Aber das geht am Thema vorbei. Jobs' Vermächtnis findet sich nicht in einem bestimmten technologischen Durchbruch. Sein Vermächtnis ist eine starke Vision davon, wie die Menschen mit ihrer Technologie umgehen können. Indem er diese Beziehung veränderte, veränderte er das Gesicht der Welt. In der nächsten Übung werdet ihr ein weiteres Gerät bauen und codieren, das die Welt verändert hat - den Scheibenwischer.

Basic-Scheibenwischer

Robert Kearns wird die Ehre zuteil, den intervallgeschalteten Scheibenwischer erfunden zu haben. Als Ford, Chrysler und andere Autofirmen versuchten, sein Wischerdesign in ihren Autos zu verwenden, klagte Kearns wegen Patentverletzung. Mit anderen Worten, Kearns behauptete, dass diese Autofirmen seine Idee ohne Genehmigung verwendet und seine Idee kopiert hätten, um ihren eigenen intervallgeschalteten Scheibenwischer herzustellen. Die Autohersteller behaupteten, es handele sich nicht um eine Patentverletzung, da eine Erfindung angeblich neue Teile haben solle und Kearns' Wischersystem nur eine Kombination von gemeinsamen Teilen sei. Mehrere US-Gerichte, darunter der Oberste Gerichtshof der Vereinigten Staaten, stellten sich jedoch auf die Seite von Kearns und erklärten, dass Erfindungen Baugruppen aus alten Teilen sein können. Kearns gewann seinen Fall und wurde von den Autofirmen bezahlt.

In dieser Übung baut und programmiert ihr euren eigenen Basic-Scheibenwischer. Genau wie bei der Erfindung von Kearns werdet ihr keine neuen elektronischen Komponenten oder Teile in eurem Schaltkreis verwenden. Ihr solltet bereits mit allen Teilen vertraut sein. Wenn ihr sie jedoch auf eine andere Art und Weise als bisher zusammensetzt, könnt ihr eine neue Erfindung erschaffen.

FURTHER NOTES

Ihr beginnt mit dem Aufbau eines grundlegenden Scheibenwischerkreises. Ein Standardservo erzeugt die Wischerbewegung, während ein Druckknopf und ein Potentiometer zur Steuerung des Wischers verwendet werden.

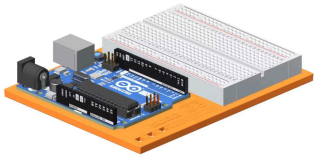
TEACHER NOTES

Klassendiskussion

Erwägen Sie eine Klassendiskussion darüber, ob die Schülerinnen und Schüler den Entscheidungen der US-Gerichte darüber zustimmen oder nicht, dass neue Baugruppen aus alten Teilen neue Erfindungen sind.

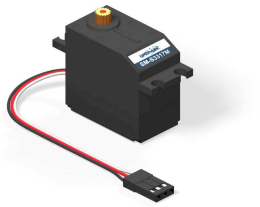
- Können die Schülerinnen und Schüler andere Erfindungen auflisten, die neue Baugruppen aus alten Teilen sind?
- Welche anderen Kriterien sollte es geben, damit eine Vorrichtung als Erfindung betrachtet werden kann?
- Wie sollten Menschen dafür bestraft werden, dass sie die Erfindung eines Anderen ohne Erlaubnis benutzen oder kopieren?

Benötigte Materialien

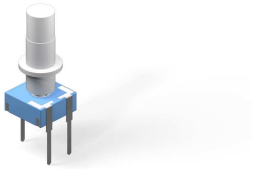


1 ARDUINO PROJEKT BOARD

1 100 MIKROFARAD KONDENSATOR

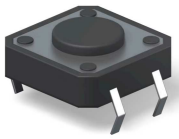


1 SERVO



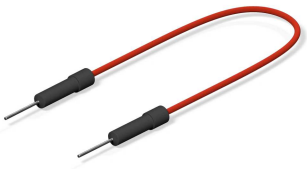
1 POTENTIOMETER

1 10K Ω WIDERSTAND

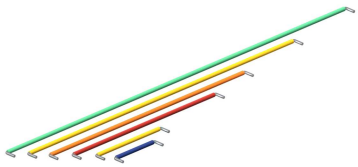


1 TASTSCHALTER

1 SCHWARZE STROMKABEL



1 ROT STROMKABEL



1 STECKVERBINDER

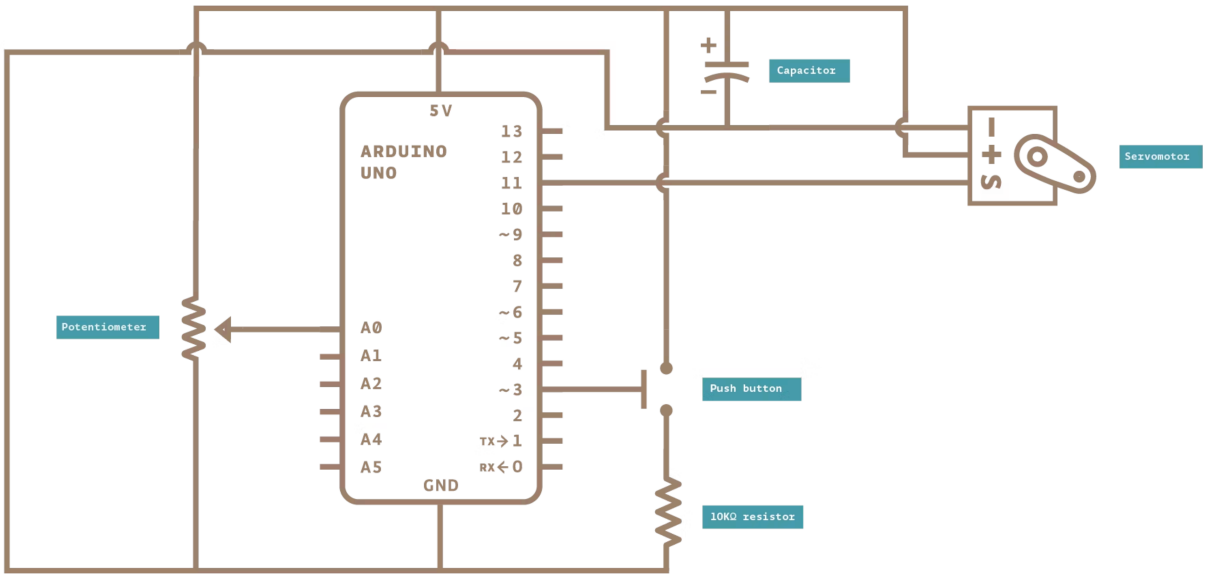
1 USB-KABEL

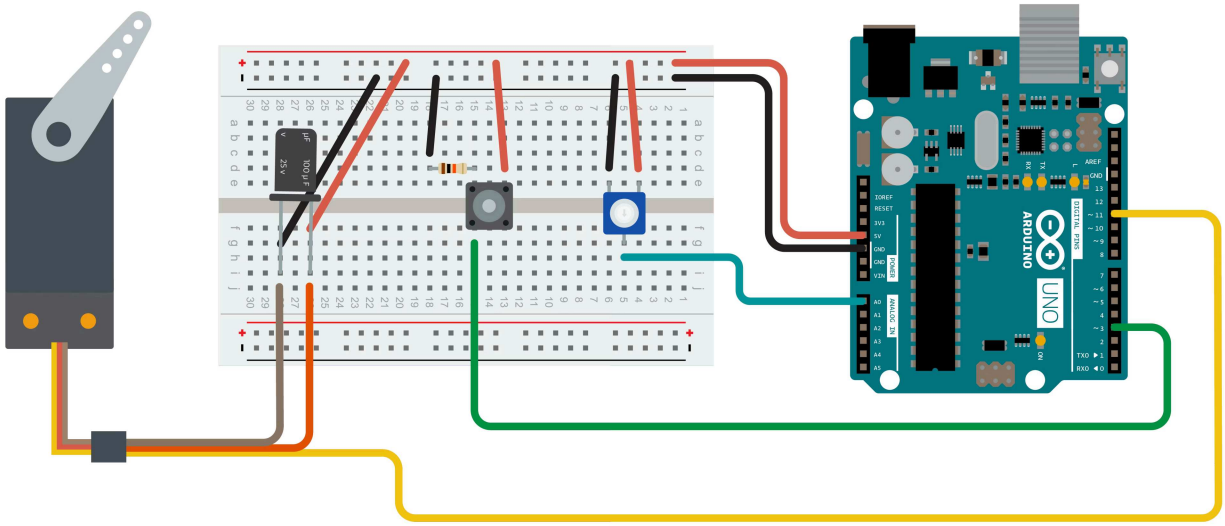
FURTHER NOTES

Die Farbbänder des 10K-Ohm-Widerstandes sind:

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

Schaltplan



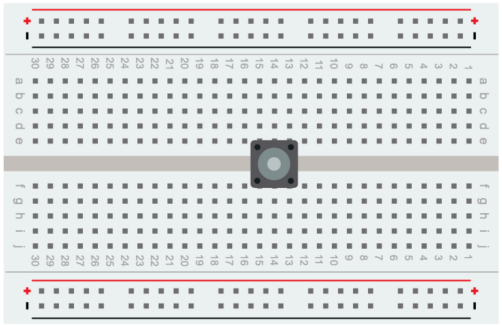


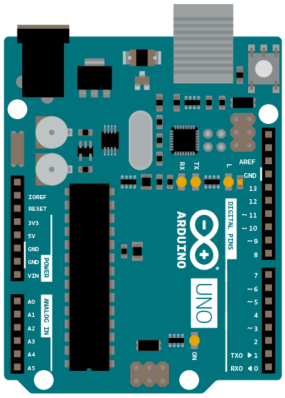
Aufbau der Schaltung

Benutzt das Schaltbild und den Schaltplan oder die folgende Slideshow unten, um die Schaltung aufzubauen.

Step 1

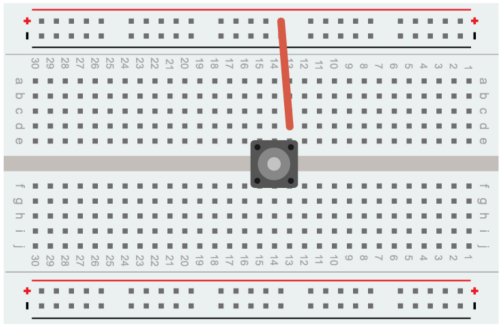
Befestigt einen Tastschalter an der unteren Seite des Steckbretts, so dass er den Spalt in der Mitte des überbrückt.

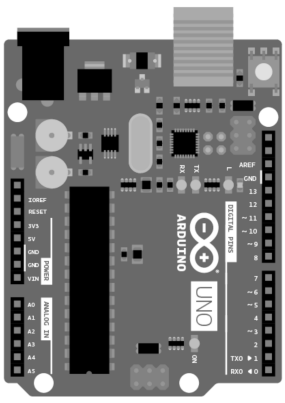




Step 2

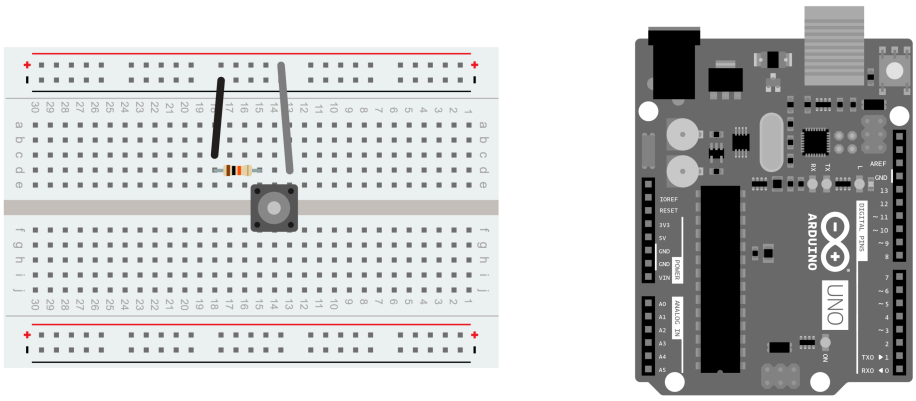
Verbindet den oberen Stift des Tastschalters mit dem Pluspol auf der linken Seite des Steckbretts mit einem kurzen Steckverbinder.





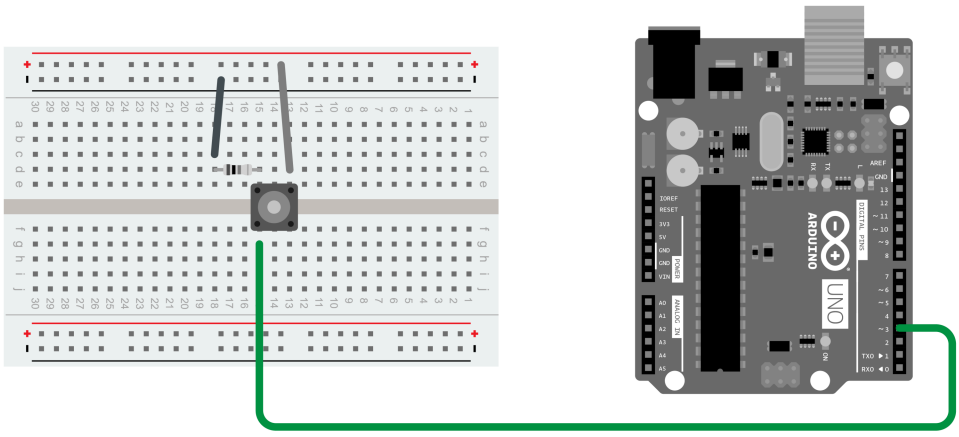
Step 3

Benutzt einen 10K-Ohm-Widerstand, um den unteren Pin des Tastschalters mit dem Minuspol des Steckbretts zu verbinden.



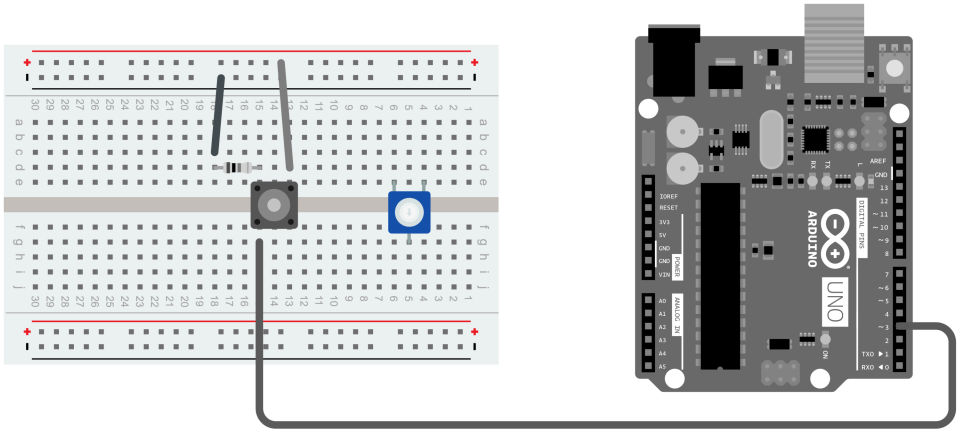
Step 4

Benutzt einen langen Steckverbinder, um den unteren Pin des Tastschalters mit Pin 3 auf dem Steckbrett zu verbinden.



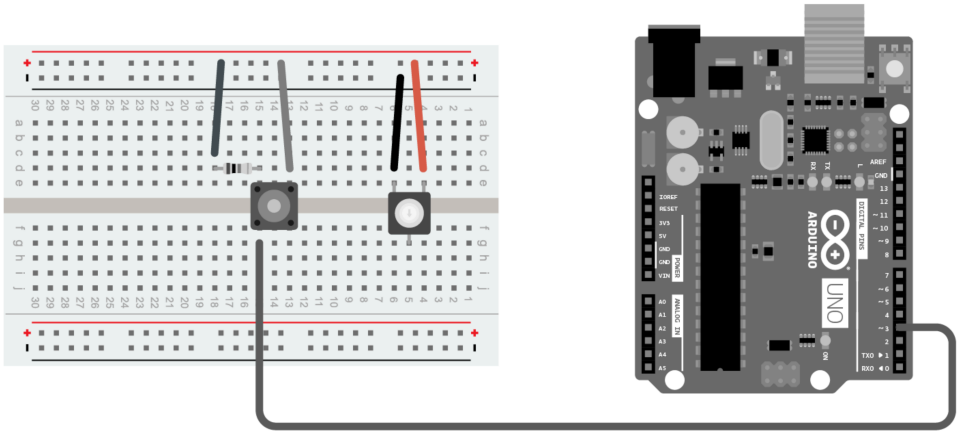
Step 5

Befestigt das Potentiometer am Steckbrett, so dass es den Spalt in der Mitte des Steckbretts überbrückt. Befestigt das Potentiometer so, dass die Power- und Groundpins zum Arduino UNO R3 Board zeigen und der Wischer-Pin vom Arduino UNO R3 Board weg zeigt.



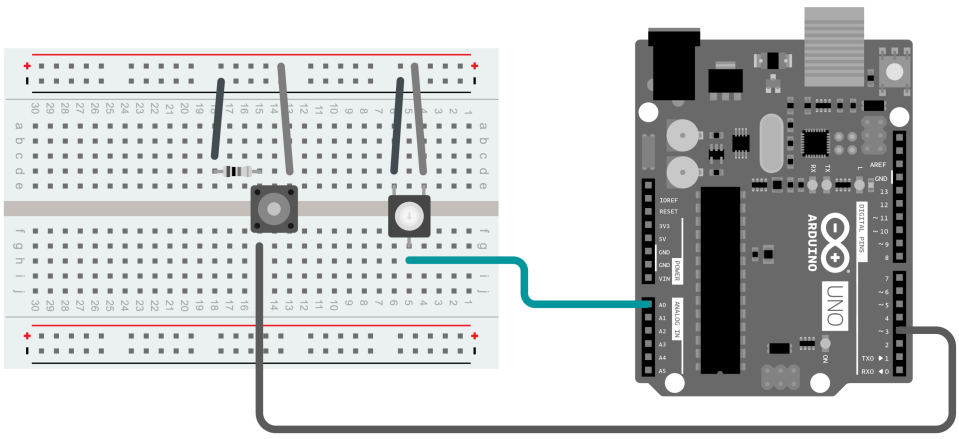
Step 6

Benutzt einen kurzen Steckverbinder, um den Power-Pin des Potentiometers mit dem positiven Pol des Steckbretts zu verbinden. Benutzt einen weiteren kurzen Steckverbinder, um den Groundpin des Potentiometers mit dem Minuspol des Steckbrettes zu verbinden.



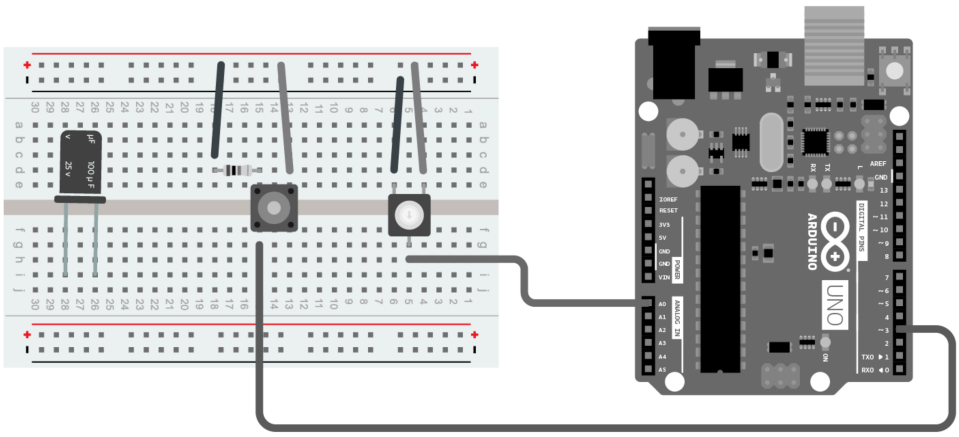
Step 7

Benutzt einen langen Steckverbinder, um den Wischer-Pin des Potentiometers mit dem analogen Pin A0 des Arduino UNO R3 Boards zu verbinden.



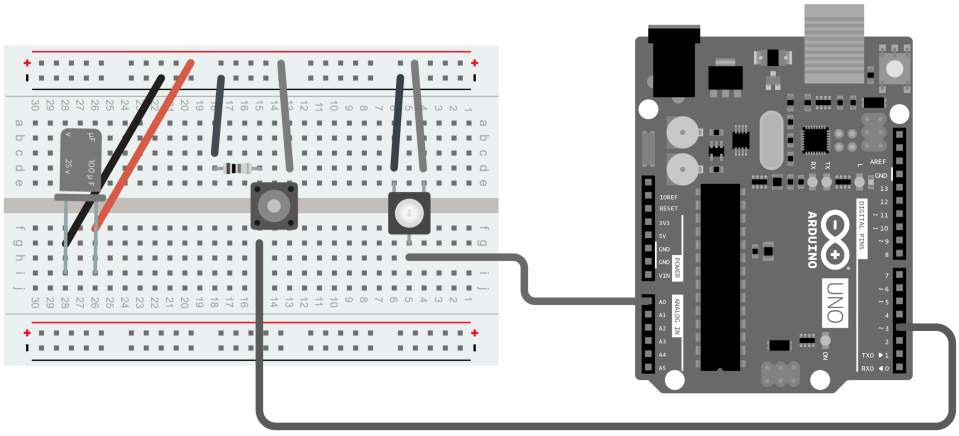
Step 8

Schließt einen 100 µf Kondensator in der Nähe des oberen Teils des Steckbretts an. Schließt den Kondensator so an, dass die Anode von euch weg und die Kathode zu euch hin eingesteckt ist. Erinnert euch daran, dass die Kathode der kurze Draht ist und durch den Streifen an der Seite des Kondensators markiert wird.



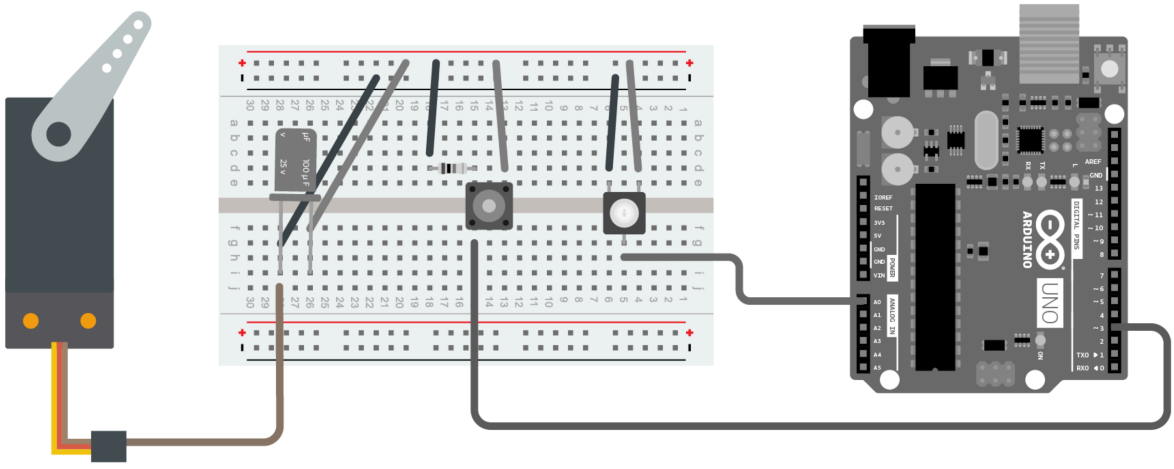
Step 9

Benutzt einen kurzen Steckverbinder, um die Anode des Kondensators mit dem positiven Pol des Steckbretts zu verbinden. Benutzt einen weiteren kurzen Steckverbinder, um die Kathode des Kondensators mit dem Minuspol des Steckbrettes zu verbinden.



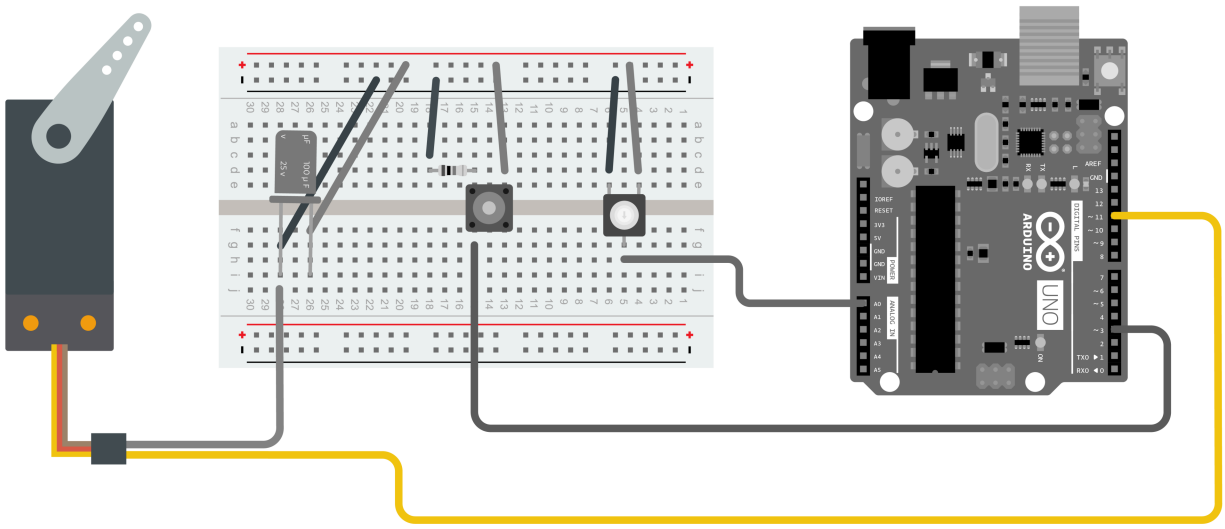
Step 10

Benutzt einen langen Steckverbinder, um das schwarze Kabel von der Buchse des Servos mit der Kathode des Kondensators zu verbinden. Achtet darauf, dass das kreuzförmige Servohorn des Servos angeschlossen und mit der kleinen Schraube gesichert ist.



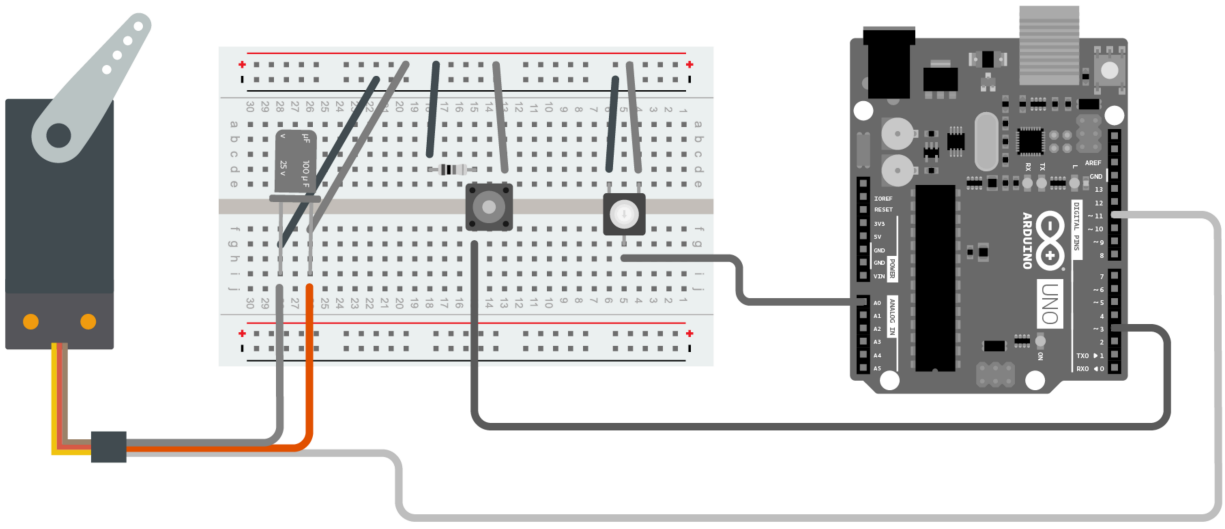
Step 11

Verbindet das weiße Kabel von der Buchse des Servos mit Pin 11 des Arduino UNO R3 Boards. Dies ist der Draht, der das Servo steuert.



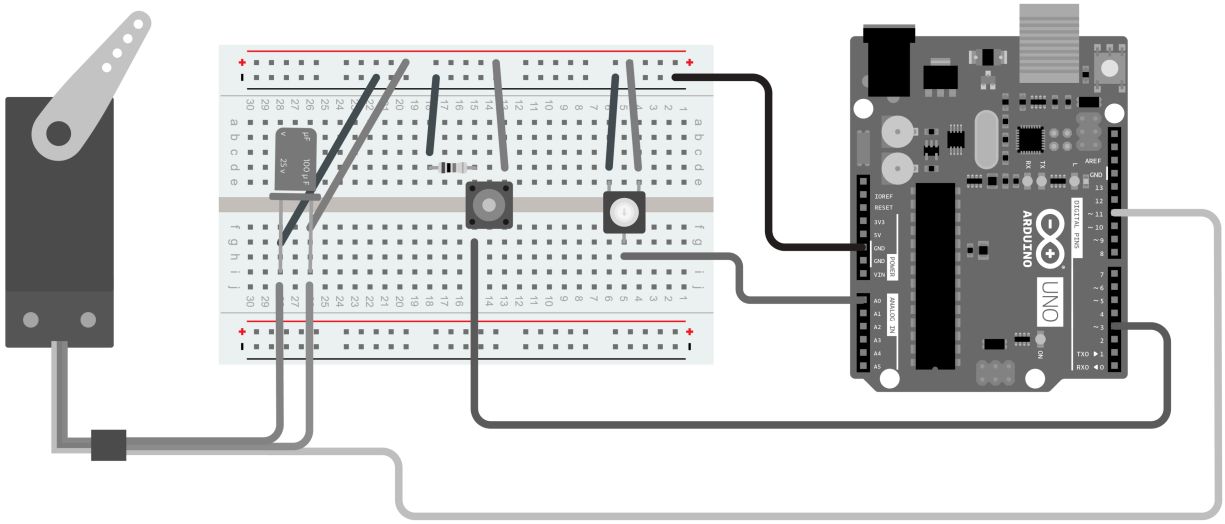
Step 12

Benutzt einen dritten langen Steckverbinder, um den roten Draht von der Buchse des Servos mit der Anode des Kondensators zu verbinden.



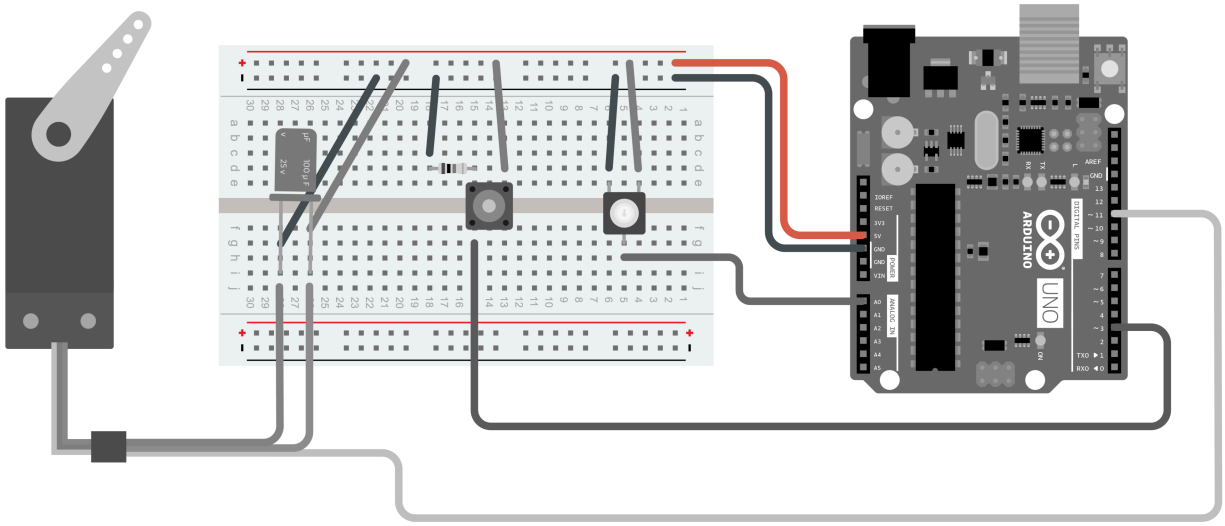
Step 13

Schließt das schwarze Stromkabel an einen Ground-Pin auf dem Arduino UNO R3 Board an. Schließt das andere Ende des Stromkabels an den Minuspol des Steckbretts an. Dadurch werden der Servo und das Potentiometer geerdet.



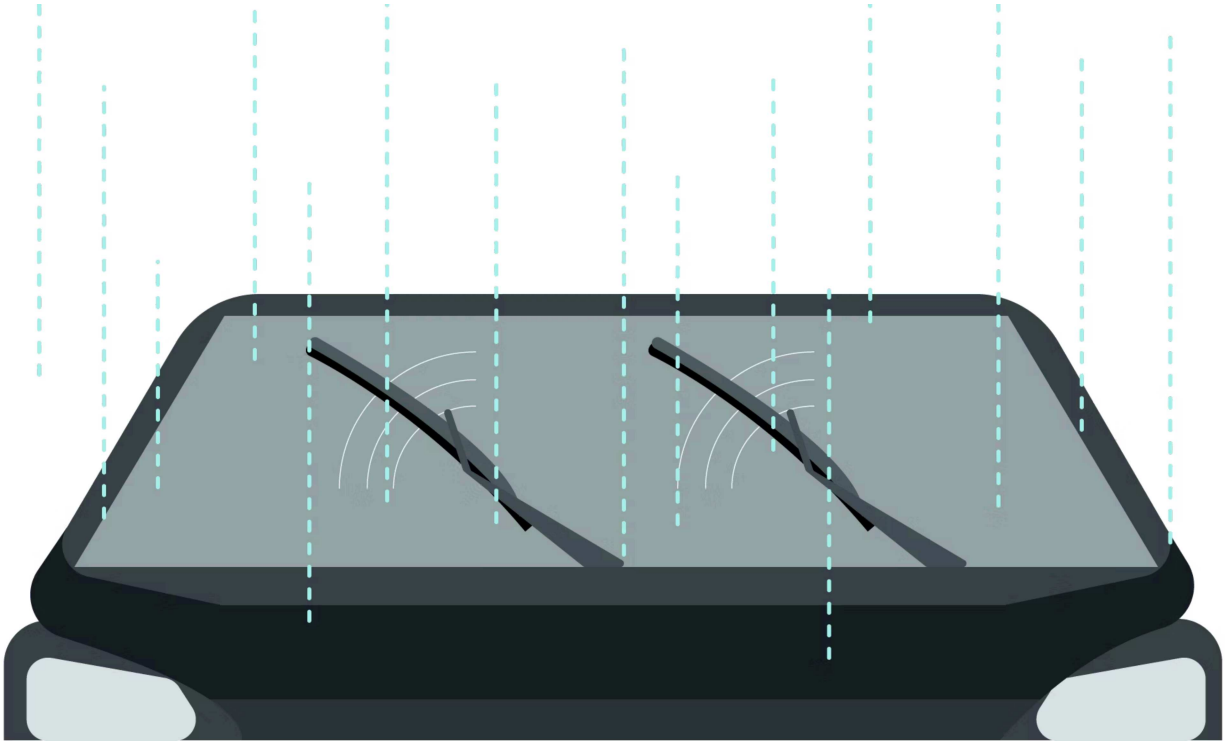
Step 14

Schließt das rote Stromkabel an den 5V-Pin auf dem Arduino UNO R3 Board an. Verbindet das andere Ende des Stromkabels mit dem positiven Anschluss des Steckbretts. Dies versorgt Servo und Potentiometer mit fünf Volt Strom.


100 µF
25 V
ARDUINO
UNO

Hinzufügen des Wischerblatts

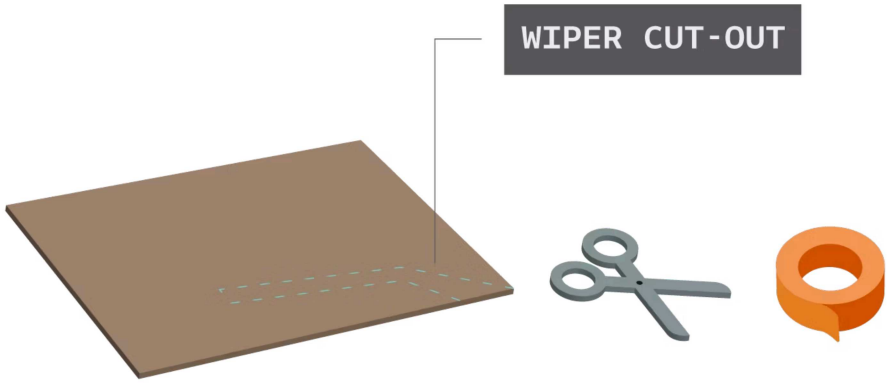
Die Scheibenwischer in den meisten modernen Autos funktionieren ein wenig anders als der Wischer, den ihr herstellt. In den meisten Autos werden beide Scheibenwischer von einem einzigen Motor angetrieben. Durch Hebel und Gestänge bewegen sich die Scheibenwischer vor und zurück. Bei eurem Scheibenwischer befestigt ihr den Wischer direkt am Servo.



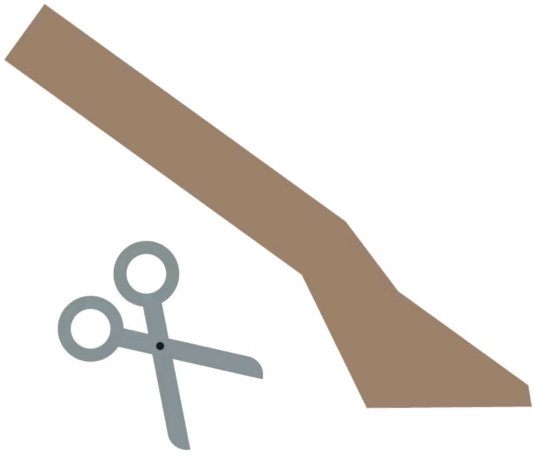
FURTHER NOTES

Für diese Übung brauchen wir Karton, Scheren und Klebeband zur Verfügung stellen.

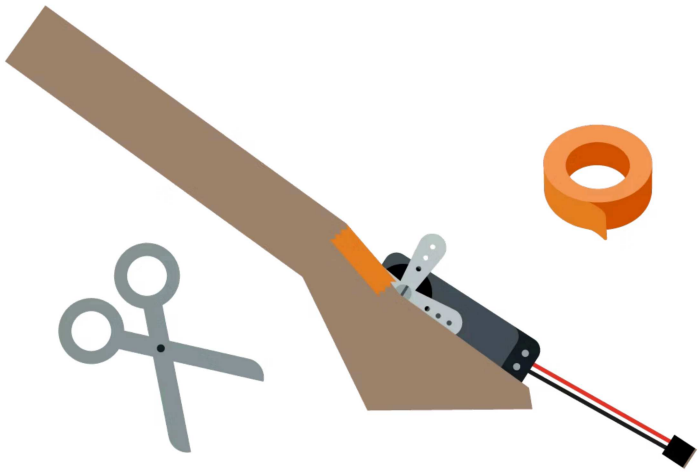
1) Zeichnet die Form eines Scheibenwischers auf ein Stück Pappe. Dieser sollte nicht länger als 7 bis 10 Zentimeter (drei oder vier Zoll) lang sein. Zeichnet euren Scheibenwischer so, dass ihr Platz auf dem Karton spart.



2) Benutzt die Schere, um euren Wischer auszuschneiden.



3) Befestigt euren Scheibenwischer mit ein paar Stücken Klebeband an dem Horn des Servos. Wenn ihr fertig seid, ist euer Scheibenwischer fertig.



Code-Erstellung - Scheibenwischer ein/aus

Bei dieser Übung programmiert ihr euren Scheibenwischer so, dass er sich ein- und ausschaltet, wenn ihr den Knopf in eurer Schaltung drückt.

FURTHER NOTES

In dieser Übung erstellen wir die Grundfunktionalität, mit der der Scheibenwischer zwischen Ein- und Ausschaltmodus hin- und hergeschaltet wird. Wir werden in folgende Programmierkonzepte eingeführt:

- **verschachtelte Bedingungen –** erzeugt Bedingungen innerhalb einer anderen bedingten Anweisung

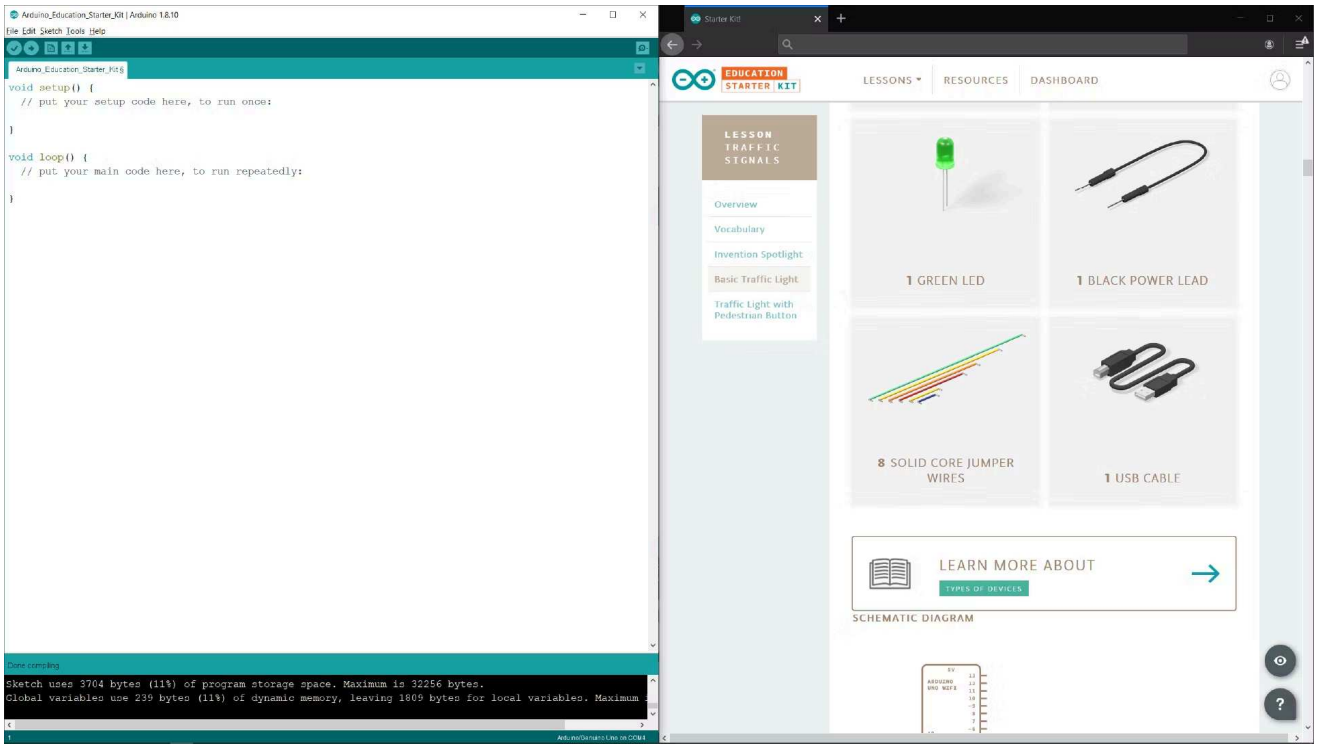
Wir werden diese Konzepte überprüfen:

- **#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

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

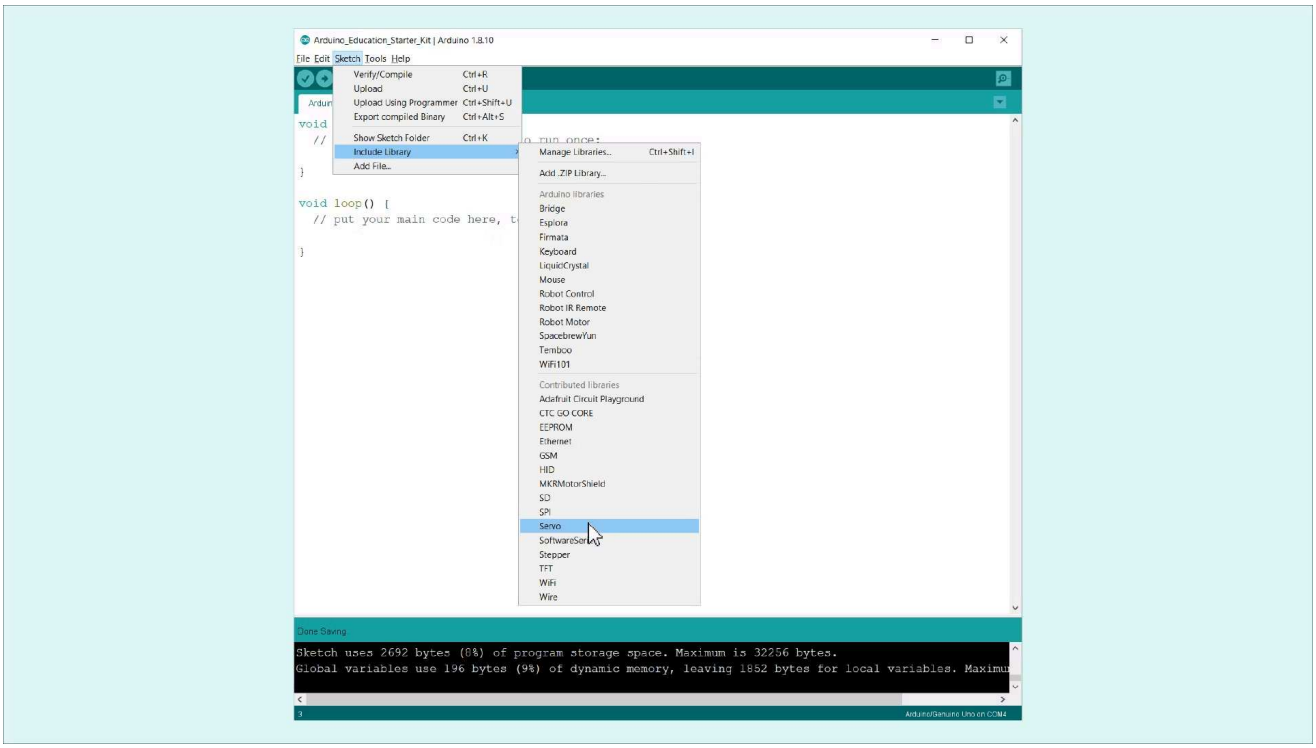
2) Wählt im Dateimenü die Option **Neu**. Dadurch wird ein neuer Sketch in einem neuen Fenster gestartet. Ihr könnt das andere Sketchfenster schließen.

3) Ihr könnt entweder das Sketch-Fenster maximieren und zwischen ihm und diesen Anweisungen hin- und herschalten, oder ihr könnt beide Fenster nebeneinander anordnen, so dass ihr beide Fenster gleichzeitig sehen könnt. Wählt die Methode, die für euch am Besten funktioniert.



4) Speichert den Sketch unter einem neuen Namen. Wählt im Dateimenü die Option Speichern unter... Der Speicherort des Sketches sollte standardmäßig das Arduino Sketchbook sein. Wenn dies nicht der Fall ist, fragt euren Lehrer, wo ihr den Sketch auf eurem Computer speichern sollt. Benennt die Datei "Lektion7*V1*", gefolgt von euren Initialen. Klickt auf **Speichern**.

5) Da eure Schaltung einen Servomotor als Scheibenwischer verwendet, müsst ihr die **Servo-Bibliothek** in euren Sketch importieren. Geht zum Menü Sketch und klickt auf Bibliothek hinzufügen. Dieses Pop-Out-Menü zeigt euch die Bibliotheken an, die derzeit in der IDE installiert sind. Wählt im Pop-Out-Menü **Servo**.



Die Servo-Bibliothek sollte nun am Anfang eures Sketches enthalten sein.

Erstellt einen Code-Kommentar zur Erläuterung dieses Befehls.

6) Um den Servo zu verwenden, müsst ihr ein Servo-Objekt erstellen. Dadurch könnt ihr die Befehle in der Servobibliothek mit dem an euren Schaltkreis angeschlossenen Servo verwenden. Ihr könnt das Servo-Objekt benennen, wie ihr wollt. In diesem Beispiel wird das Servo-Objekt **myServo** genannt. Gebt nach dem Befehl zum Einfügen der Servobibliothek einen Befehl zum Erstellen und Benennen eures Servoobjekts ein. Fügt dem Befehl einen Code-Kommentar bei.

7) Um den Scheibenwischer ein- und auszuschalten, verwendet eure Schaltung zwei Pins auf dem Arduino UNO R3 Board. Deklariert diese Pins als Konstanten, so dass ihr sie mit Namen und nicht mit der Pin-Nummer bezeichnen könnt. Pin 11 wird den Servo für den Scheibenwischer steuern. Benennt diesen Pin **servoPin**. Pin 3 ist mit dem Tastschalter verbunden, der den Wischer ein- und ausschaltet. Benennt Pin 3 **onButtonPin**. Diese Konstanten-Deklarationen müssen vor der **void setup()**-Funktion stehen.

Hinweis: Bis zu diesem Zeitpunkt wurde ein Beispielcode mit Kommentaren zu fast jeder Zeile gezeigt, die jeden Befehl erklären. Da Programme jedoch immer länger und komplexer werden, kann all dieser zusätzliche Text den Code komplizierter aussehen lassen, als er tatsächlich ist. Um den Code sauber und ordentlich voranzubringen, werden Kommentare zur Erklärung von Code-Abschnitten und nicht jeder Zeile verwendet. Letztendlich liegt es an euch, wie ihr Code-Kommentare verwendet.

8) In früheren Lektionen, in denen ihr Tastschalter verwendet habt, musstet ihr die Taste gedrückt halten, um das Verhalten der Taste zu aktivieren. Mit anderen Worten, die Taste war an, wenn sie gedrückt gehalten wurde, und aus, wenn sie oben war. Dieses Verhalten führt jedoch nicht zu sehr effektiven Scheibenwischern. Könnt ihr euch vorstellen, dass ihr einen Knopf gedrückt halten müsst, während ihr bei Regen über eine nasse Straße fahrt, um die Scheibenwischer zu aktivieren?

Anstatt die Scheibenwischer durch Festhalten des Tastschalters ein- und auszuschalten, könnt ihr eine Variable erstellen, um den Ein-/Aus-Zustand des Servos zu speichern. Dann könnt ihr den Tastschalter verwenden, um den Zustand der Variable umzuschalten. Auf diese Weise schalten sich die Scheibenwischer beim Drücken des Schalters ein und bleiben eingeschaltet, auch wenn der Knopf losgelassen wird. Wenn die Taste dann erneut gedrückt wird, ändert sich der Zustand und die Scheibenwischer schalten sich aus.

Um den Ein-/Aus-Zustand eures Scheibenwischers zu verfolgen, erstellt ihr eine ganzzahlige Variable namens **wiperState**. Zur Steuerung der Scheibenwischer zeigt ein Wert von 0 an, dass die Scheibenwischer ausgeschaltet sind. Ein Wert von 1 zeigt an, dass die Scheibenwischer eingeschaltet sind. Setzt den Wert der Variablen **wiperState** zu Beginn auf 0, oder aus. Eure Variablendeklaration sollte vor der Funktion **void setup()** stehen.

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

10) Ihr seid bereit, zur Funktion void setup() eures Sketches überzugehen. Das erste, was ihr hier tun müsst, ist, eure Arduino-Pins als Ein- oder Ausgänge zu deklarieren. Euer Tastschalter ist ein Eingabegerät. Verwendet den Befehl pinMode(), um die Konstante onButtonPin als Eingang zu deklarieren.

Hinweis: Ihr müsst den Befehl **pinMode()** nicht verwenden, um euren Servo-Pin als Ausgang zu deklarieren. Der Servo-Pin wird automatisch als ein Ausgangs-Pin eingerichtet, wenn ihr das **myServo**-Objekt im nächsten Schritt mit dem Servo-Pin verbindet.

11) Um dem Arduino UNO R3 Board mitzuteilen, an welchem Pin der Servo angeschlossen ist, verwendet ihr den Befehl **servoName.attach(pin**). In diesem Befehl sollte **servoName** so lauten, wie ihr das Objekt im oberen Bereich des Sketches benannt haben. Im gezeigten Beispiel wird es **myServo** genannt. Der Pin sollte **servoPin** heißen, denn so habt ihr den Pin benannt.

Ihr Kommando sollte ähnlich wie dieses aussehen:

12) Fügt einen Befehl hinzu, um den seriellen Monitor mit einer Baudrate von 9600 Baud zu starten. Mit dem seriellen Monitor könnt ihr überwachen, in welchem Zustand sich die Wischer befinden.

13) Als Letztes müsst ihr bei der Funktion **void setup()** noch die Anfangsposition des Servos einstellen. Bei den meisten Autos beginnen die Scheibenwischer von einer unteren Position und bewegen sich nach oben, wenn sie eingeschaltet werden. Für euren Servo sollte dies Null Grad sein. Erinnert euch, dass der Befehl zum Bewegen des Servos **servoName.write(position)** lautet. Hier sollte **servoName** der Name eures Servos sein und position sollte 0 sein. Fügt den Befehl am Ende der Funktion **void setup()** hinzu.

14) Überprüft euren Code und debugged ihn wenn nötig.

15) In der **void loop()**-Funktion des Sketches muss als Erstes der Wischermodus eingestellt werden, je nachdem, ob die Ein-/Aus-Taste gedrückt wird oder nicht. Wenn die Taste gedrückt wird, muss der Wischer in dem Sketch ein- oder ausgeschaltet werden. Wenn die Taste nicht gedrückt wird, sollte der Sketch nichts tun. Ihr könnt dieses Verhalten mit einem Standard-Bedingungstyp erstellen.

Die Bedingung in dem if-Statement muss den Wert des Arduino-Pins lesen, der an den Tastschalter (genannt **onButtonPin**) angeschlossen ist, und ihn mit einem Wert von HIGH vergleichen. Dadurch wird bestimmt, ob der Taster gedrückt ist. Der Befehl **digitalRead(pin)** gibt den HIGH- oder LOW-Wert eines digitalen Pins zurück. Erstellt innerhalb der Funktion **void loop()** eine Standard-Bedingungsanweisung, die wie folgt aussieht:

16) Wenn die Taste gedrückt wird, muss der Sketch den Modus des Scheibenwischers ändern. Erinnert euch, dass die Variable **wiperState** den Modus des Scheibenwischers verfolgt - 0 steht für aus, und 1 für ein. Wenn die Taste gedrückt wird, muss die Variable **wiperState** um 1 erhöht werden. Fügt innerhalb des if-Statements einen Befehl hinzu, der den neuen Wert von wiperState auf den alten Wert von **wiperState** plus 1 setzt.

Hinweis: Das Erhöhen des Wertes einer Variablen um eins ist eine so übliche Praxis bei der Kodierung, dass die Arduino IDE einen verkürzten Befehl für diese Aufgabe hat. Der Befehl Schrittweise erhöht den Wert einer Variablen um eins. Der folgende Befehl macht dasselbe wie der Befehl, den ihr in eurem if-Statement eingegeben habt.

```
wiperState++;
```

17) Der Befehl, den ihr gerade eingegeben habt, dient dazu, den Scheibenwischer von Aus auf Ein zu schalten. Was aber, wenn der Scheibenwischer bereits eingeschaltet ist? Der Sketch muss den Scheibenwischer wieder ausschalten. Mit anderen Worten, die Variable **wiperState** muss auf Null gehen. Die Wischerzustandsvariable erhöht sich jedoch bei jedem Tastendruck um eins. Um dies zu beheben, könnt ihr eine verschachtelte Bedingung

verwenden. Eine verschachtelte Bedingung ist eine Bedingung innerhalb einer anderen Bedingung. Mit anderen Worten, wenn eine Bedingung wahr ist, dann wird überprüft, ob eine andere Bedingung wahr ist.

Habt ihr zum Beispiel jemals einen Prozess wie den unten gezeigten durchlaufen, um zu entscheiden, was an einem Sommertag zu tun ist? Verschachtelte Bedingungen werden verwendet, um eines von beliebig vielen Szenarien zu bestimmen.

WAS IST AN EINEM SOMMERTAG ZU TUN??

Erstellt eine verschachtelte Bedingung, um beim Drücken des Tastschalters zu bestimmen, ob die **wiperState**-Variable größer oder gleich 2 ist; wenn ja, dann setzt die **wiperState**-Variable auf 0 zurück. Dadurch wird die **wiperState**-Variable bei jedem Drücken des Tastschalters zwischen 0 und 1 hin- und hergeschaltet. Eure verschachtelte Bedingung sollte wie folgt aussehen:

Hinweis: In früheren Lektionen, in denen ihr Bedingungen erstellt habt, habt ihr die Bedingung in eine Zeile und die auszuführenden Befehle in die Zeilen nach der Bedingung geschrieben. In diesem Fall steht der auszuführende Befehl in derselben Zeile wie die Bedingung. Beide Formate sind korrekt.

Wenn Code kompiliert wird, achtet die Arduino IDE nicht auf Leerzeichen und Zeilenumbrüche. Beide Beispiele werden auf die gleiche Weise kompiliert. Die erste Methode macht den Code jedoch sauberer und leichter lesbar, wenn mehrere Befehle innerhalb des if-Statements ausgeführt werden sollen. Wenn nur ein Befehl innerhalb des if-Statements ausgeführt werden soll, hält die zweite Methode den Code übersichtlicher und leichter lesbar.

18) Fügt nach eurem Haupt-if-Statement einen Befehl hinzu, um die Variable **wiperState** auf dem seriellen Monitor auszudrucken. Auf diese Weise könnt ihr die Variable visuell sehen und sehen, ob der Wischer ein- oder ausgeschaltet ist. Stellt sicher, dass ihr den **Serial.println()** Befehl benutzt, so dass jede Iteration durch die Schleife in einer neuen Zeile gedruckt wird.

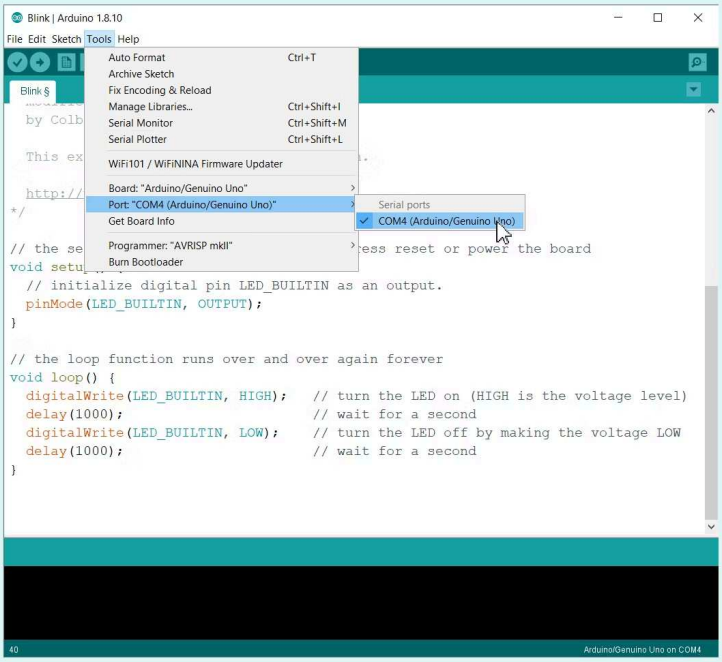
Hinweis: Achtet darauf, diesen Befehl nach der schließenden geschweiften Klammer des Haupt-if-Statements und vor der schließenden geschweiften Klammer der **void loop()**-Funktion hinzuzufügen.

19) Fügt am Ende der Schleife eine kurze Verzögerung von 250 Millisekunden hinzu. Diese Verzögerung gibt euch Zeit, den Tastschalter loszulassen, bevor die nächste Iteration der Schleife beginnt. Ohne diese Verzögerung würde die Schleife so schnell ablaufen, dass ein Tastendruck den Scheibenwischer ein-, dann aus-, dann wieder ein-, dann wieder ausschalten würde und so weiter, bis der Tastschalter losgelassen wird. Diese Verzögerung bestimmt also in gewisser Weise, wie lange der Tastschalter gedrückt werden muss, um die Scheibenwischer ein- oder auszuschalten.

20) Überprüft euren Code und debugged ihn wenn nötig.

21) Verbindet euer Arduino UNO R3 Board über das USB-Kabel mit eurem Computer.

22) 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.

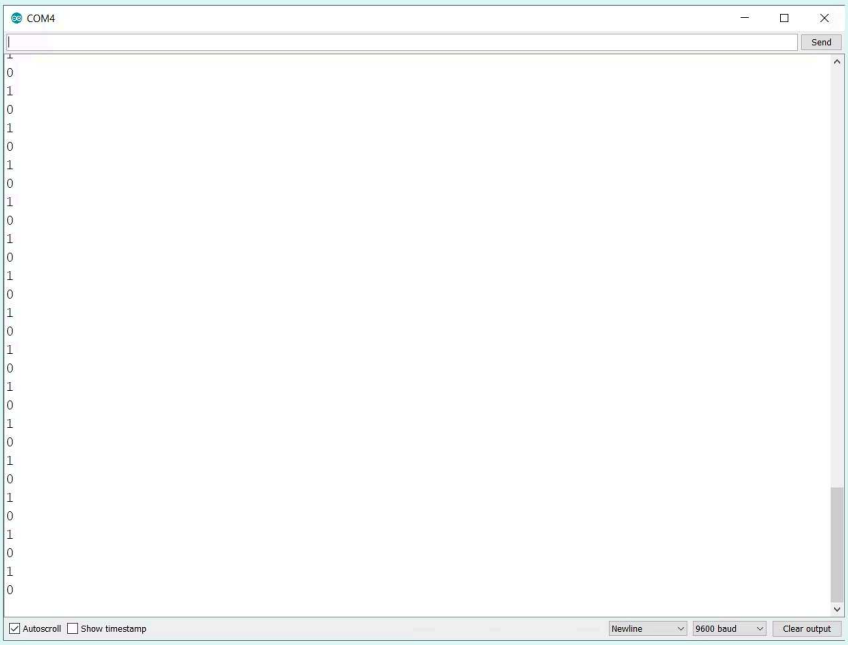


23) Klickt auf den **Upload**-Button, um den Sketch auf euer Arduino UNO R3 Board hochzuladen. Wenn der Upload abgeschlossen ist, beginnt euer Sketch automatisch, zu laufen.

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

25) Beobachtet den seriellen Monitor, während ihr den Ein/Aus-Knopf in eurem Schaltkreis drückt. Ihr solltet sehen, dass die Anzeige zwischen 0 und 1 wechselt. Denkt daran, dass 0 für aus und 1 für an steht. Wenn sich eure **wiperState**-Variable nicht ändert, müsst ihr eure Schaltung und euren Code debuggen. Geht noch einmal die Schaltungsaufbauanleitung durch, um eure Verbindungen zu überprüfen. Vergleicht euren Code mit dem, was hier angezeigt wird

26) Übt, ein Gefühl dafür zu bekommen, wie lange ihr den Tastschalter drücken müsst, um den Zustand des Wischers zu ändern. Was passiert, wenn ihr den Tastschalter zu kurz drückt? Zu lange?



27) Zieht das USB-Kabel von eurem Arduino UNO R3 Board ab, um eure Schaltung auszuschalten.

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

Code-Erstellung - Wischerbewegung

Mit eurem Scheibenwischercode habt ihr einen guten Start hingelegt. Aber auch wenn ihr hoffentlich euren Ein/Aus-Schalter zum Funktionieren gebracht habt, bewegt sich euer Scheibenwischer nicht. In dieser Übung programmiert ihr euren Scheibenwischer-Sketch weiter, sodass sich der Scheibenwischer, wenn er eingeschaltet ist, hin und her dreht. Wenn der Scheibenwischer im ausgeschalteten Zustand ist, hört er auf, sich zu drehen.

FURTHER NOTES

Bei dieser Übung fügen wir dem Servo Bewegung hinzu, sodass er sich beim Drücken der Taste ein- und ausschaltet. Zu diesem Zweck verwenden wir eine andere Art von bedingter Struktur, die als Switch-Case-Struktur bezeichnet wird.

In dieser Übung werden wir mit diesen Kodierungskonzepten vertraut gemacht:

- **switch(*variable*){}–** initiiert die Switch-Case-Struktur und identifiziert die Variable, die den Fall bestimmt
- **Case –** erzeugt ein Verhalten, das ausgeführt wird, wenn der Fall aufgerufen wird
- **Break –** identifiziert das Ende des Falles

TEACHER NOTES

Die Schülerinnen und Schüler überprüfen diese Konzepte:

- `servoName.write(angle)` – ein Befehl aus der Servo-Bibliothek, der ein Servo-Objekt zu einem bestimmten Winkel bewegt

1) Öffnet gegebenenfalls euren Lektion7_V1-Sketch in der Arduino IDE.

2) Speichert die Skizze unter einem neuen Namen. Wählt im Menü Datei die Option Speichern unter... Benennte die Datei "Lektion7_V2_", gefolgt von euren Initialen. Klickt auf Speichern.

3) Ordnet euer Lektion7_V2-Sketchfenster und dieses Fenster so an, wie es für euch am Besten geeignet ist.

4) Im Moment hat euer Scheibenwischer zwei Modi - an und aus. Der Wischermodus wird durch die Variable wiperState verfolgt. Um das Verhalten des Wischers für jeden Modus zu

programmieren, könntet ihr eine "Wenn-Sonst"-Bedingung verwenden. Der Pseudocode mit einem if-else-Statement könnte etwa so aussehen:

```
1) Wenn der Wischerzustand 0 ist (0 steht für aus), dann:
a. Wird der Servo in die untere Position bewegt.
2) Sonst:
a. Bewegt sich der Servo in die obere Position.
b. Wartet, bis der Scheibenwischer oben ist.
c. Bewegt sich der Servo in die untere Position.
d. Wartet, bis der Scheibenwischer unten ist.
```

Mit einem if-else-Statement könnt ihr jedoch nur zwei Modi programmieren. Später in dieser Lektion werdet ihr dem Wischer weitere Modi hinzufügen. Um das Verhalten für mehr als zwei Modi zu programmieren, könnt ihr eine "Wenn-Sonst wenn"-Anweisung als Typ einer bedingten Struktur verwenden. Der Pseudocode könnte wie folgt aussehen:

```
1) Wenn der Wischerzustand 0 ist (0 steht für aus), dann:
a. Wird der Servo in die untere Position bewegt.
2) Sonst, wenn der Wischerzustand 1 ist (1 für an), dann:
a. Bewegt sich der Servo in die obere Position.
b. Wartet, bis der Scheibenwischer oben ist.
c. Bewegt sich der Servo in die untere Position.
d. Wartet, bis der Scheibenwischer unten ist.
3) Sonst, wenn der Wischerzustand 2 ist (2 für intervallgeschaltet), dann:
a. Verwendet das Potentiometer, um die Verzögerungszeit zwischen den
Wischvorgängen einzustellen.
b. Bewegt sich der Servo in die obere Position.
c. Wartet, bis der Scheibenwischer oben ist.
d. Bewegt sich der Servo in die untere Position.
e. Wartet die festgelegte Zeitspanne ab, die durch den Wert des Potentiometers
bestimmt wird.
```

Eine Wenn-Sonst wenn bedingte Struktur würde für mehrere Modi funktionieren. Ihr könntet so viele weitere if-Statements für so viele Modi hinzufügen, wie ihr programmieren möchtet. Die Arduino IDE hat jedoch eine spezielle Bedingungsstruktur für Situationen wie diese. Mit der Switch-Case-Struktur könnt ihr mehrere Verhaltensweisen programmieren und mit einer einzigen Variablen wählen, welches Verhalten ausgeführt werden soll. Der Pseudocode für eine Switch-Case-Struktur könnte wie folgt aussehen:

```
1)Deklariert einen Switch-Case und identifiziert die Variable, die den Fall
auswählen wird.
1) Deklariert case 0 (aus).
a. Wird der Servo in die untere Position bewegt.
3) Deklariert case 1 (an).
a. Bewegt sich der Servo in die obere Position.
b. Wartet, bis der Scheibenwischer oben ist.
c. Bewegt sich der Servo in die untere Position.
d. Wartet, bis der Scheibenwischer unten ist.
4) Deklariert case 2 (intervallgeschaltet).
a. Verwendet das Potentiometer, um die Verzögerungszeit zwischen den
Wischvorgängen einzustellen.
b. Bewegt sich der Servo in die obere Position.
c. Wartet, bis der Scheibenwischer oben ist.
```

```
d. Bewegt sich der Servo in die untere Position.
e. Wartet die festgelegte Zeitspanne basierend auf dem Wert vom Potentiometer
ab.
```

Es gibt Vor- und Nachteile für die Verwendung einer if-else if-Typstruktur gegenüber einer Switch-Case-Typstruktur. Die folgende Tabelle listet einige dieser Vor- und Nachteile auf.

	Vorteile	Nachteile
if-else if	Kann zwei oder mehr Bedingungen basierend auf verschiedenen Variablen kombinieren Kann sehr komplexe logische Strukturen aufbauen Kann If-Statements innerhalb anderer If-Statements verschachteln Kann auf Wertebereiche testen	Der Code ist weniger sauber. Der Code benötigt mehr Speicherplatz
switch-case	Ist eine einfache Struktur zur Auswahl eines Verhaltens abhängig von einer einzigen Variable. Der Code ist sauber und effizient.	Kann keine komplexen logischen Strukturen basierend auf mehreren Variablen aufbauen Kann nur auf einen einzelnen Wert einer Variablen prüfen (kein Wertebereich)

Eine Switch-Case-Struktur eignet sich hervorragend zur Programmierung des Verhaltens eures Scheibenwischers. Der Befehl **switch(variable){}** startet eine Switch-Case-Struktur. Die Variable im Befehl wird verwendet, um zu wählen, welches Verhalten ausgeführt werden soll. Die Verhaltensweisen bzw. Fälle für das Switch-Case werden innerhalb der Klammern definiert.

Ihr verwendet die Variable **wiperState**, um euer Wischerverhalten einzustellen. Fügt vor der schließenden Klammer eurer **void loop()**-Funktion den switch-Befehl wie dargestellt hinzu.

5) In einer **Switch-Case**-Struktur werden verschiedene Verhaltensweisen als Fälle angelegt. Euer Scheibenwischer hat derzeit zwei Verhaltensweisen, die als Fälle angelegt werden müssen - aus und ein. Um einen Fall zu definieren, verwendet ihr den Wert des Befehls **case**: wobei *value* der Wert der im Switchbefehl definierten Variablen ist. Gebt den Befehl ein, um das Aus-Verhalten des Scheibenwischers wie dargestellt zu definieren.

Hinweis: Beachtet die Verwendung des Doppelpunkts nach dem Fallwert. Der Doppelpunkt zeigt an, dass die Befehle, die auf den Doppelpunkt folgen, das Verhalten für die Groß-/Kleinschreibung festlegen.

6) Wenn sich der Scheibenwischer im Aus-Zustand befindet, sollte der Servo in der unteren Position bleiben. Die Abwärtsposition des Scheibenwischers wird bei Null Grad liegen. Um dieses Verhalten zu erzeugen, fügt ihr den Befehl **servoName.write(0)**; zum Fall 0 hinzu.

Hinweis: Vergesst nicht, den **servoName**-Teil des Befehls in den Namen zu ändern, den ihr eurem Servoobjekt oben in dem Sketch gegeben habt.

7) Das Schlüsselwort **break** wird verwendet, um die Switch-Case-Struktur zu verlassen. Dieses Schlüsselwort wird normalerweise am Ende jedes Falles verwendet. Fügt den Break-Befehl am Ende von Fall 0 vor der schließenden Klammer der Switch-Case-Struktur ein.

8) Als Nächstes müsst ihr den Fall für den eingeschalteten Zustand des Scheibenwischers erstellen. Der eingeschaltete Zustand für den Wischer ist, wenn die Variable **wiperState**

gleich 1 ist. Erstellet den Fall 1 direkt unter dem Break-Befehl für den Fall 0 und vor der schließenden Klammer der Switch-Case-Struktur.

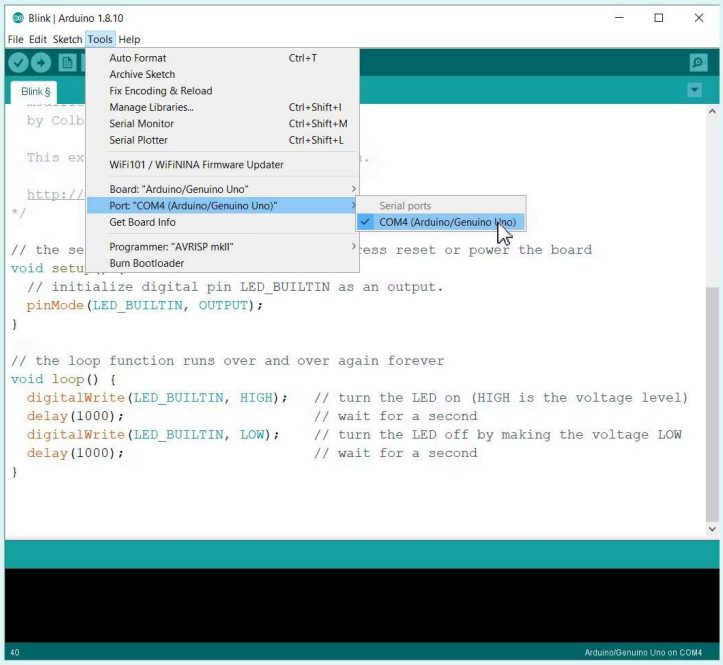
9) Das Verhalten für Fall 1 sollte eine Hin- und Herbewegung des Wischers bewirken. Es dauert etwa eine Sekunde, bis sich der Servo von der unteren Position (0 Grad) in die obere Position (179 Grad) bewegt. Fügt nach jeder Servobewegung Verzögerungen von einer Sekunde hinzu. Euer Code für Fall 1 sollte wie folgt aussehen:

10) Fügt am Ende von Fall 1 vor der schließenden Klammer der Switch-Case-Sruktur einen Unterbrechungsbefehl ein.

11) Überprüft Code und debugged diesen, falls erforderlich. Eure void loop()-Funktion sollte ähnlich wie diese aussehen:

12) Verbindet euer Arduino UNO R3 Board über das USB-Kabel mit eurem Computer.

13) 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.



14) Klickt auf die Schaltfläche Hochladen, um den Sketch auf euer Arduino UNO R3 Board hochzuladen. Wenn der **Upload** abgeschlossen ist, wird euer Sketch automatisch ausgeführt.

15) Drückt den Tastschalter in eurer Schaltung, um den Scheibenwischer ein- und auszuschalten. Im eingeschalteten Zustand solltet ihr sehen, wie sich der Scheibenwischer vor und zurück bewegt. Im ausgeschalteten Zustand sollte der Scheibenwischer in der unteren Position bleiben. Ihr könnt auch den seriellen Monitor starten, um zu beobachten, wie sich die Variable wiperState beim Ein- und Ausschalten des Wischers ändert.

Wenn euer Wischer nicht richtig funktioniert, debugged ihr eure Schaltung und euren Code.

Hinweis: Wenn der Scheibenwischer eingeschaltet ist, kann es schwierig sein, den Scheibenwischer wieder auszuschalten. Wenn ihr die Taste drückt, während sich der Code in einer der Verzögerungen befindet, wird der Tastendruck nicht registriert. Daher müsst ihr den Tastendruck zeitlich auf den Zustand des Scheibenwischers timen, in dem sich der Scheibenwischer im Stillstand befindet. Dies ist der Nachteil der Verwendung von Verzögerungen im Code. Während das Arduino in einer Verzögerung wartet, kann es nichts anderes tun. In der nächsten Übung werdet ihr dieses Problem lösen, indem ihr Schleifen und den internen Timer des Arduino verwendet, um die Verzögerungen zu ersetzen.

16) Nachdem ihr euch vergewissert habt, dass eure Schaltung funktioniert, speichern ihr euren Sketch.

17) Zieht das USB-Kabel von eurem Arduino UNO R3 Board ab, um euren Schaltkreis auszuschalten.

FURTHER NOTES

Verständnis-Überprüfung

Nachdem wir nun sowohl verschachtelte Bedingungen als auch eine Switch-Case-Struktur verwendet haben, fragen wir uns, welche Struktur für diese Situationen besser geeignet wäre:

- Zählen, wie oft eine Taste gedrückt wird, und Aufleuchten einer Anzahl von LEDs, die der Anzahl der Tastendrücke entspricht.
- Bestimmen des analogen Werts eines Potentiometers und Bewegung eines Servos auf 90 Grad, wenn der Wert zwischen 500 und 700 liegt.
- Drucken von verschiedenen Meldungen im seriellen Monitor auf der Grundlage der Lichtintensität von einem Lichtsensor und der Temperatur von einem Temperatursensor.

Es gibt Möglichkeiten, wie ihr Verschachtelte Bedingungsstrukturen funktionieren jedoch in der Regel besser, wenn ihr mehrere bedingte Tests durchführen müsst oder wenn ihr einen Wert mit einem Bereich von Werten vergleichen müsst. Dadurch eignen sich die zweite und dritte Situation besser für verschachtelte bedingte Anweisungen. Switch-Case-Strukturen eignen sich gut für Situationen mit mehreren Verhaltensweisen, die auf einer einzigen Variablen oder einem einzigen Wert basieren. Die erste Situation eignet sich am besten für eine Switch-Case-Kodierungsstruktur.

Code-Erstellung - Intervallgeschalteter Wischer

Neben der Möglichkeit, die Scheibenwischer ein- und auszuschalten, können die meisten Fahrzeuge auch die Geschwindigkeit der Scheibenwischer steuern. Wenn es regnet, könnt ihr die Scheibenwischer bei einem Regenschauer schnell oder bei leichtem Nebel langsam fahren lassen. Dies nennt man die intervallgeschaltete Einstellung. In dieser Übung programmiert ihr eure Schaltung so, dass sie die Geschwindigkeit der Scheibenwischer mit dem Potentiometer steuert.

FURTHER NOTES

Bei dieser Übung fügen wir unserem Scheibenwischerkreis eine intervallgeschaltete Funktion hinzu. Wir fügen unserem Switch-Case-Aufbau einen dritten Fall hinzu, der das Potentiometer zur Steuerung der Geschwindigkeit des Scheibenwischers verwendet.

TEACHER NOTES

Die Schülerinnen und Schüler überprüfen diese Konzepte:

- **switch(*variable*){}** – initiiert die Switch-Case-Struktur und identifiziert die Variable, die den Fall bestimmt
- **case** – erzeugt ein Verhalten, das ausgeführt wird, wenn der Fall aufgerufen wird
- **break** – identifiziert das Ende des Falles
- **map()** – skaliert einen Wert von einem Bereich in einen anderen Bereich

1) Öffnet gegebenenfalls euren Lektion7_V2-Sketch in der Arduino IDE.

2) Speichert die Skizze unter einem neuen Namen. Wählt im Dateimenü die Option Speichern unter... Benennt die Datei "Lektion7_V3_", gefolgt von euren Initialen. Klickt auf **Speichern**.

3) Ordnet euer Lektion7_V3-Sketchfenster und dieses Fenster so an, wie es für euch am Besten geeignet ist.

4) Um das Potentiometer zur Steuerung der Geschwindigkeit der Wischer zu verwenden, müsst ihr dem Arduino mitteilen, an welchem Pin das Potentiometer angeschlossen ist. In

eurer Schaltung sollte das Potentiometer an den analogen Pin A0 angeschlossen werden. Erstellt eine Konstante namens **potPin** und setzt sie gleich A0.

Hinweis: Achtet darauf, dass ihr beim Einstellen des Wertes der Konstante **potPin** den Buchstaben A gefolgt von einer Null verwenden, nicht den Buchstaben O.

5) Ihr müsst auch den analogen Wert des Potentiometers in einer Variablen speichern. Erstellt eine ganzzahlige Variable namens **potValue** und setzt sie auf 0. Diese Variable werdet ihr später im Code verwenden.

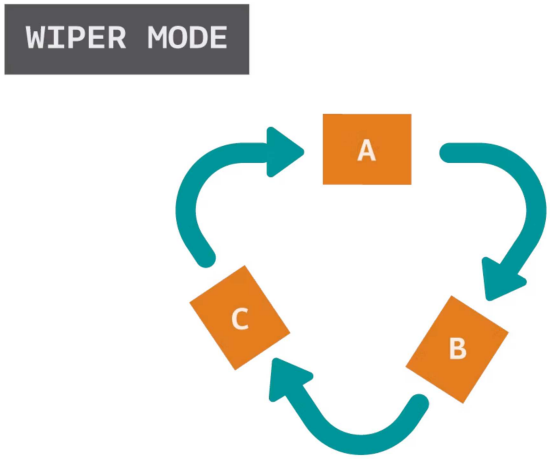
6) Erstellt eine weitere Variable mit dem Namen **delayTime** und setzt sie auf 0. Mit dieser Variable werdet ihr die Zeit zwischen den Servobewegungen auf der Basis des Potentiometers einstellen.

Hinweis: Die Wörter *delay* und *time* sind beides Schlüsselwörter in der Arduino IDE. Ihr könnt diese Wörter nicht als Variablennamen verwenden. Ihr könnt diese Schlüsselwörter jedoch kombinieren, um einen gültigen Namen (**delayTime**) zu erstellen, der kein Schlüsselwort ist.

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

8) Wechselt als Nächstes zur Funktion void loop() eurem Sketch. Das Potentiometer wird verwendet, um die Geschwindigkeit des Wischers im intervallgeschalteten Modus zu steuern. Verwendet einen **analogRead()**-Befehl, um den Analogwert von dem Potentiometer zu lesen. Speichert den Analogwert in der **PotValue**-Variablen, die ihr oben in eurem Sketch erstellt habt. Erinnert euch daran, dass ihr Pin A0 als den konstanten **potPin** benannt habt

9) Der intervallgeschaltete Betrieb des Scheibenwischers wird ebenfalls über die Ein-/Aus-Taste gesteuert. Durch Drücken der Taste wird der Wischermodus von aus auf an, dann auf intervallgeschaltet und wieder zurück auf aus geschaltet. Erinnert euch, dass das Arduino UNO R3 Board die Variable **wiperState** verwendet, um den Wischermodus zu verfolgen



A) EIN: wiperState = 1 B) Intervallgeschaletet: wiperState = 2 C) AUS: wiperState = 0

Am Anfang eurer **void loop()**-Funktion befindet sich ein if-Statement, das abliest, ob die Ein-/Ausschalttaste gedrückt wird. Innerhalb dieses if-Statements befindet sich ein Befehl zur Erhöhung des Wertes der Variablen **wiperState** um 1. Ein verschachteltes if-Statement setzt die Variable wiperState auf 0 zurück, wenn der **wiperState** größer oder gleich 2 ist. Nun, da es einen dritten Modus für den Wischer gibt, muss die Variable **wiperState** zurückgesetzt werden, wenn sie größer oder gleich 3 statt 2 ist. Nehmt diese Änderung in eurem Code vor.

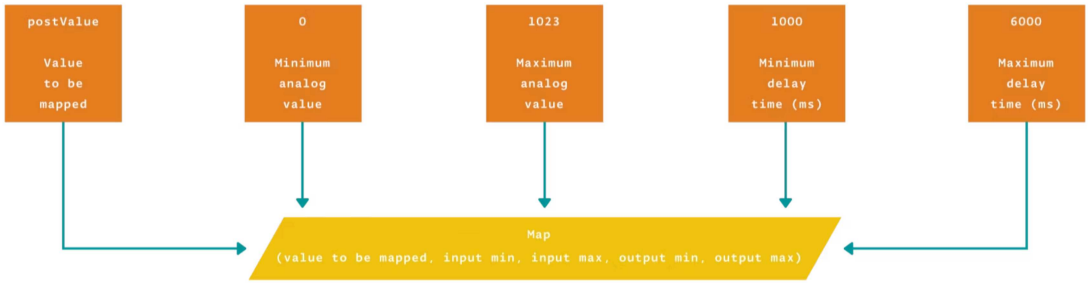
10) Ihr habt zwei neue Variablen für den intervallgeschalteten Modus erstellt, die ihr im seriellen Monitor überwachen könnt - potValue und delayTime. Es ist am einfachsten, den

seriellen Monitor zu lesen, wenn die Informationen jedes Mal, wenn die Schleife durchlaufen wird, auf einer einzigen Zeile gedruckt werden.

- Ändert den Druckbefehl für die Variable wiperState in Serial.print() anstelle von Serial.println().
- Fügt einen Druckbefehl hinzu, um einen Doppelpunkt als Trennzeichen zwischen Werten anzuzeigen.
- Fügt einen Serial.print()-Befehl hinzu, um die Variable potValue zu drucken.
- Fügt einen weiteren Druckbefehl hinzu, um einen Doppelpunkt als Trennzeichen zwischen den Werten anzuzeigen.
- Fügt am Ende einen Serial.println()-Befehl hinzu, um die Variable delayTime zu drucken.
- Ändert den Druckbefehl für die Variable wiperState in **Serial.print()** anstelle von **Serial.println()**.
- Fügt einen Druckbefehl hinzu, um einen Doppelpunkt als Trennzeichen zwischen Werten anzuzeigen.
- Fügt einen **Serial.print()**-Befehl hinzu, um die Variable potValue zu drucken.
- Fügt einen weiteren Druckbefehl hinzu, um einen Doppelpunkt als Trennzeichen zwischen den Werten anzuzeigen.
- Fügt am Ende einen **Serial.println()**-Befehl hinzu, um die Variable **delayTime** zu drucken.

11) Bewegt euch nach unten in den Switch-Case-Abschnitt eures Codes. Ihr habt bereits das Wischerverhalten für Ein und Aus definiert - Fall 0 und 1. Um das Wischerverhalten für den intervallgeschalteten Modus zu definieren, müsst ihr einen weiteren Fall hinzufügen. Fügt unterhalb des Unterbrechungsbefehls für Fall 1 und vor der schließenden Klammer für die Switch-Case-Funktion Fall 2 hinzu.

12) Bei intervallgeschaltetem Betrieb wird die Zeit zwischen den Wischvorgängen mit dem Potentiometer eingestellt. Ihr könnt die Funktion **map()** verwenden, um den Potentiometerbereich von 0 bis 1.023 auf die intervallgeschaltete Zeit zwischen den Wischvorgängen zu skalieren. Da es etwa eine Sekunde dauert, bis sich der Wischer von seiner oberen Position in seine untere Position bewegt, sollte eine Sekunde die vom Potentiometer eingestellte Mindestverzögerungszeit sein. Vorerst könnt ihr sechs Sekunden als maximale Verzögerungszeit verwenden. Erinnert euch, dass die Map-Funktion fünf Parameter hat.




```
map(zu skalierender Wert, Input min, Input max, Output min, Output max)
```

Gebt den Befehl zur Einstellung der Variable **delayTime** auf den skalierten Potentiometerwert wie folgt ein:

FURTHER NOTES

Verständnis-Überprüfung

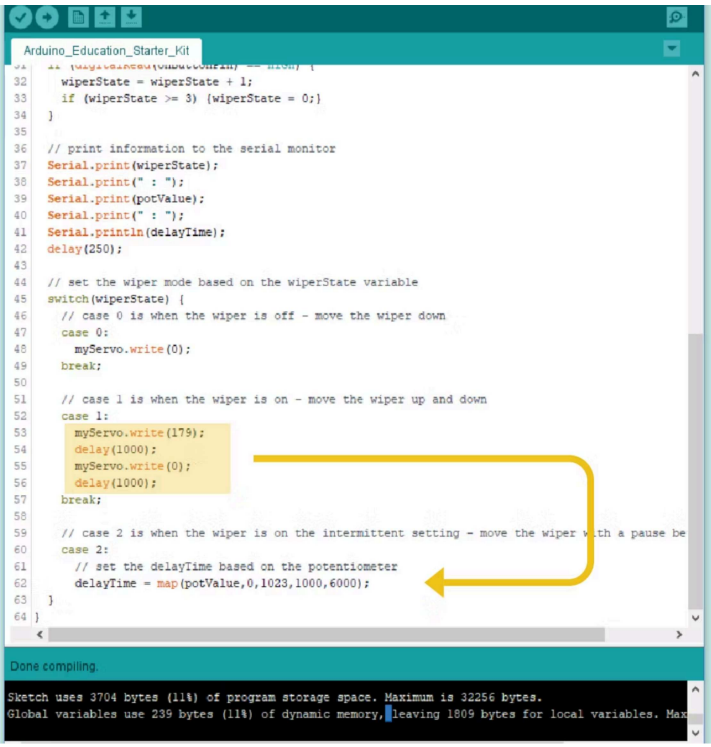
Dies ist die zweite Lektion, in der die **map()**-Funktion verwendet wird, um einen Datenbereich auf einen anderen zu skalieren. Wenn wir jedoch immer noch Schwierigkeiten mit diesem Befehl haben, fragen wir uns, wie die Funktion aussehen würde, wenn sie acht Sekunden als maximale Zeit zwischen den Wischerbewegungen festlegen wollten.

```
map(potValue, 0, 1023, 1000, 8000)
```

Was ist, wenn sie die schnellste Geschwindigkeit auf anderthalb Sekunden statt auf eine Sekunde einstellen müssen?

```
map(potValue, 0, 1023, 1500, 6000)
```

13) Ihr habt bereits Befehle geschrieben, um den Wischer nach oben und wieder nach unten zu bewegen. Kopiert die vier Zeilen aus Fall 1 und fügt sie unter eurem Skalierbefehl ein.



14) Im intervallgeschalteten Modus pausieren die meisten Scheibenwischer in der unteren Position. Ändert in Ieuem letzten Verzögerungsbefehl die Verzögerungszeit von 1.000 Millisekunden auf die Variable **delayTime**, die in eurem Skalierbefehl eingestellt wurde. Dadurch ändert sich die Verzögerung basierend auf dem Wert vom Potentiometer.

15) Beendet Fall 2 mit einem Unterbrechungsbefehl vor der schließenden Klammer der Switch-Case-Funktion.

16) Überprüft euren Code und debugged ihn, falls erforderlich.

17) Verbindet euer Arduino UNO R3 Board über das USB-Kabel mit eurem Computer.

18) Klickt auf den **Upload**-Button, um den Sketch auf euer Arduino UNO R3 Board hochzuladen. Wenn der Upload abgeschlossen ist, wird euer Sketch automatisch ausgeführt.

19) Öffnet den seriellen Monitor, damit ihr Änderungen an deinen Variablen beobachten könnt. Testet euren Code, indem ihr den Wischer zwischen den drei Modi umschaltet. Wenn ihr euch im intervallgeschalteten Modus befindet, stellt ihr das Potentiometer ein, um die intermittierende Geschwindigkeit des Scheibenwischers zu ändern.

Hinweis: Wenn ihr euren Code testet, werdet ihr wahrscheinlich feststellen, dass es schwierig ist, den Wischer dazu zu bringen, den Modus zu wechseln, besonders wenn ihr vom intervallgeschalteten Modus zurück in den Aus-Modus wechselt. Denkt daran, dass das Arduino, während es sich in einer Verzögerung befindet, nicht feststellen kann, ob die Taste gedrückt ist. Um den Modus zu wechseln, müsst ihr das Drücken der Taste so timen, dass es geschieht, wenn das Arduino liest, ob die Taste gedrückt ist. Dies geschieht kurz bevor der Scheibenwischer in seine obere Position fährt. Es folgen ein paar Tipps zum Wechseln des Modus.

- Haltet die Taste länger gedrückt.
- Drückt die Taste kurz bevor der Scheibenwischer nach oben fährt.
- Stellt den Wischer im intervallgeschalteten Modus auf die schnellste Einstellung, um die Verzögerungszeit zu verkürzen.

TEACHER NOTES

Wenn die Schülerinnen und Schüler Schwierigkeiten haben, ihren Wischer zum Wechseln der Modi zu bewegen, könnte dies eher auf die Ineffizienz des Codes als auf einen Syntax- oder Logikfehler im Code zurückzuführen sein. In der nächsten Übung verbessern die Schülerinnen und Schüler ihren Code, um einen besser verwendbaren Wischerkreis zu erstellen.

Hoffentlich habt ihr gesehen, dass euer Scheibenwischer funktioniert. Allerdings ist er nicht sehr effektiv. Dies trifft oft auf neue Erfindungen zu. Ingenieure und Designer müssen ihre Entwürfe und Prototypen modifizieren, um Verbesserungen vorzunehmen, damit ihre

Erfindung brauchbar wird. In der nächsten Übung werdet ihr genau das tun. Ihr werdet euren Scheibenwischer besser nutzbar machen.

20) Zieht das USB-Kabel von eurem Arduino UNO R3 Board ab, um euren Schaltkreis auszuschalten.

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

TEACHER NOTES

Klassendiskussion

Die Gebrauchstauglichkeit ist ein wichtiger Faktor, der Einfluss darauf hat, ob eine Erfindung von der Gesellschaft akzeptiert wird oder nicht. Eine Erfindung kann ein großes Bedürfnis lösen, aber wenn sie nicht praktisch ist, werden die Menschen sie nicht kaufen oder benutzen wollen.

Diskutieren Sie, welche Änderungen am Wischerkreislauf vorgenommen werden könnten, um den Wischer benutzerfreundlicher oder praktischer zu machen.

Testen und ändern

Modifizieren des Codes - Verbesserte Funktionalität

In der vorherigen Übung habt ihr wahrscheinlich ein Problem bei der Möglichkeit bemerkt, die Wischermodi zu ändern. Wenn das Arduino darauf wartet, dass sich der Scheibenwischer bewegt, kann es nicht feststellen, ob die Ein-/Aus-Taste gedrückt ist. Euer Ziel bei dieser Übung ist es, die Reaktionsfähigkeit des Ein-/Ausschalters zu verbessern. Um dies zu erreichen, müsst ihr die Verzögerungsbefehle in eurem Sketch loswerden. Die Verzögerungen erledigen jedoch eine Reihe wichtiger Aufgaben. Sie geben dem Scheibenwischer Zeit, sich in Position zu bewegen, bevor er den nächsten Bewegungsbefehl ausführt. Ohne diese Verzögerungen würde der Scheibenwischer anfangen, sich nach oben zu bewegen, sich dann aber sofort wieder nach unten bewegen - es gäbe kaum eine Bewegung. Verzögerungsbefehle steuern auch die intervallgeschaltete Geschwindigkeit des Wischers. Die Pausen zwischen den Wischerbewegungen sind wichtig. Wie bringt man also das Arduino dazu, eine Pause einzulegen, aber trotzdem noch Dinge zu tun wie z.B. zu prüfen, ob der Ein-/Aus-Schalter gedrückt ist? Ihr könnt den eingebauten Timer des Arduino und eine Schleife verwenden.

FURTHER NOTES

Wir werden unsere Verzögerungsanweisungen durch Schleifen ersetzen, die den internen Timer des Arduino verwenden. Durch die Verwendung von Schleifen kann das Arduino weiterhin den Status der Ein-/Ausschalttaste überprüfen, wodurch die Fähigkeit des Sketches, den Scheibenwischer auszuschalten, verbessert wird. Wir benötigen für diese Übung den Schüler-Arbeitsheft.

TEACHER NOTES

In dieser Übung werden die Schülerinnen und Schüler mit diesen Kodierungskonzepten vertraut gemacht:

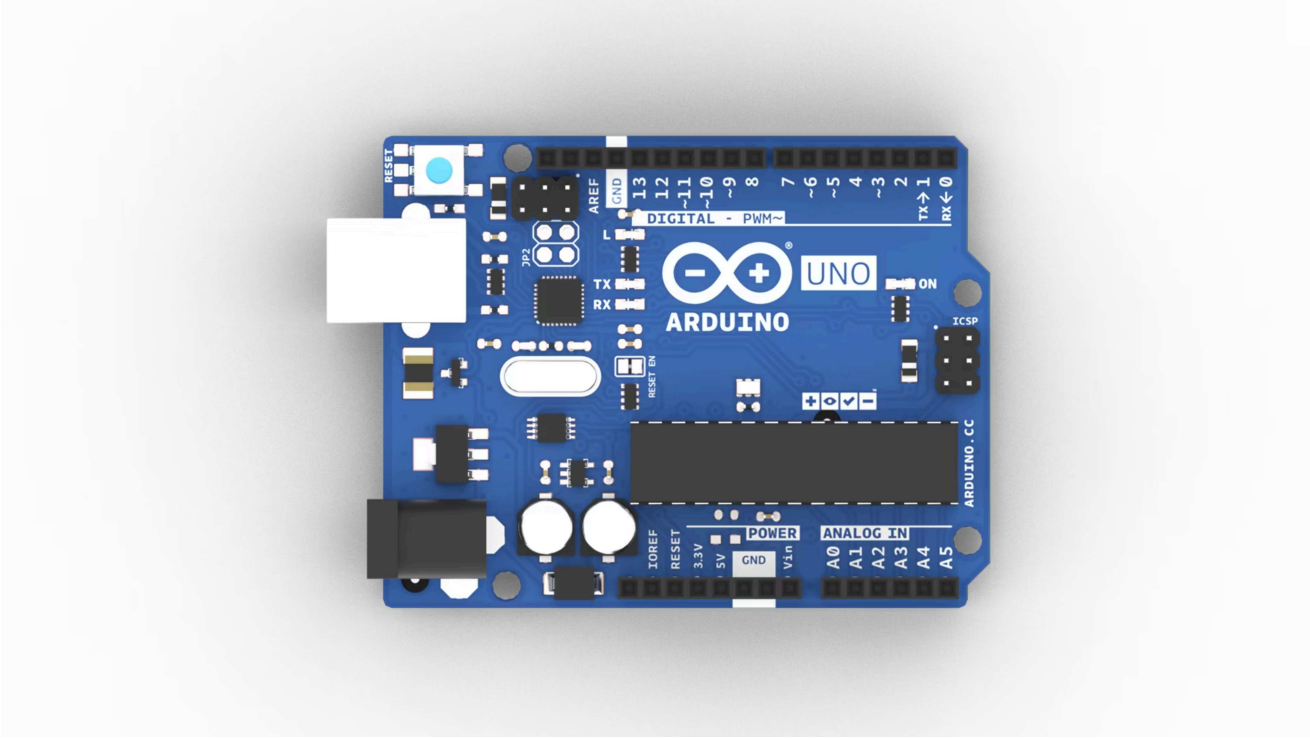
- **millis() –** gibt an, wie viele Millisekunden seit dem letzten Zurücksetzen des Arduino verstrichen sind
- **while(*condition*){} –** erzeugt eine Schleife, die sich wiederholt, solange eine bestimmte Bedingung wahr ist

1) Wenn nötig, öffnet euren Lektion7_V3 Sketch in der Arduino IDE.

2) Speichert den Sketch unter einem neuen Namen. Wählt im Dateimenü die Option Speichern unter... Benennt die Datei "Lektion7_V4_", gefolgt von euren Initialen. Klickt auf Speichern.

3) Ordnet euer Lektion7_V4-Sketchfenster und dieses Fenster so an, wie es für euch am Besten geeignet ist.

4) Euer Arduino UNO R3 Board hat einen eingebauten Timer, der wie eine Stoppuhr funktioniert. Er startet bei jedem Zurücksetzen des Boards bei Null - entweder durch Hochladen eines Sketches von der IDE, durch Aus- und Wiedereinschalten eures Boards mit dem USB-Kabel oder durch Drücken des Reset-Knopfes auf dem Arduino UNO R3 Board.



Um die Zeit vom Timer aus verfolgen zu können, müsst ihr eine neue Variable erstellen. Alle Variablen, die ihr bisher in diesem Kurs verwendet habt, waren ganzzahlige Variablen. Eine Ganzzahlvariable kann eine beliebige nicht-dezimale Zahl zwischen -32.768 und 32.767 sein. Dies ist ein 16-Bit-Bereich (2^{16} = 65.536, wobei die Hälfte des Bereichs negativ und die andere Hälfte positiv ist). Dieser Bereich eignet sich gut für die Lektionen und Übungen, die ihr bisher durchgeführt habt. Der Timer des Arduino zählt die Zeit jedoch in Millisekunden. Wenn ihr eine ganzzahlige Variable verwenden würdet, um den Timer zu verfolgen, könntet ihr die Zeit nur für 32.767 Millisekunden oder 32 Sekunden verfolgen, bevor euch der Platz in der Variable ausgeht.

Anstatt eine Ganzzahlvariable zur Verfolgung des Zeitgebers zu verwenden, könnt ihr eine Variable vom Typ lang verwenden. Long-Variablen können 32 statt 16 Datenbits speichern. Das gibt ihnen einen Bereich von -2.147.483.648 bis 2.147.483.647. Durch die Verwendung

einer langen Variablen zur Verfolgung des Arduino-Timers könnt ihr den Sketch fast 25 Tage lang ausführen, ohne ihn zurückzusetzen.

> **Hinweis:** Eine vorzeichenlose Long-Variable hat den gleichen 32-Bit-Datenbereich wie eine Long-Variable. Bei einer vorzeichenlosen Long-Variablen gibt es jedoch keine negativen Zahlen. Der Bereich ist 0 bis 4.294.967.295. Bei Verwendung einer vorzeichenlosen Long-Variablen könntet ihr die Zeit von Arduino fast 50 Tage lang verfolgen.

Deklariert am Anfang eures Sketches zusammen mit euren anderen Variablen eine Long-Variable mit dem Namen **timerCount** und setzt sie gleich 0.

5) Ihr werdet auch ein paar Variablen benötigen, um den Status des Tastschalters zu verfolgen - ob der Tastschalter gedrückt ist oder nicht. Erstellt eine ganzzahlige Variable namens **buttonState** und setzt sie auf Null. Erstellt eine zweite ganzzahlige Variable namens **lastButtonState** und setzt sie ebenfalls gleich 0

6) Ihr ändert nicht nur die Funktionsweise des Ein-/Aus-Schalters, sondern auch die Art und Weise, wie sich der Servo bewegt. Dadurch wird der Code effizienter. Erstellt eine weitere ganzzahlige Variable namens **servoAngle** und setzt sie gleich 0. Mit dieser Variable werdet ihr die Position des Servos einstellen.

7) Sucht in der Funktion **void loop()** den Code, der Informationen auf dem seriellen Monitor ausgibt. Ein Teil des Problems mit der Wischerschaltung besteht darin, dass das Arduino während der Verzögerungen nichts tun kann. Um dies zu beheben, werdet ihr die meisten **delay()**-Befehle aus dem Sketch entfernen. Löscht den Befehl delay(250), der auf den Code folgt, der Informationen im seriellen Monitor ausgibt.

8) Wechselt in den Switch-Case-Abschnitt eurer **void loop()**-Funktion. Beachtet, dass jeder Fall die Bewegung des Servos mit **myServo.write()**-Befehlen steuert. Fall 1 und Fall 2 sind fast gleich, außer dass in Fall 2 die Verzögerungszeit nach der Abwärtsbewegung des Servos auf dem Potentiometerwert für den intervallgeschalteten Modus basiert. Da das Verhalten in beiden Fällen fast dasselbe ist, könnt ihr die Variable servoAngle und die Variable delayTime verwenden, um die Bewegung des Servos zu steuern. Wenn ihr diese Variablen für jeden Fall einstellt, könnt ihr dann einen **myServo.write()**-Befehl benutzen, der nach der **Switch-Case**-Funktion kommt. Nehmen wir einen Fall nach dem anderen.

Hinweis: Wenn ihr eurem Servo-Objekt einen anderen Namen als "**myServo**" gegeben habt, dann sollte der "**myServo**"-Teil des Befehls durch den Namen ersetzt werden, den ihr eurem Servo-Objekt am Anfang eures Sketches gegeben habt.

TEACHER NOTES

Für Schülerinnen und Schüler mag dies zunächst verwirrend sein. Es ist schwer zu erkennen, wie der gesamte Code zusammenarbeitet, wenn Änderungen an bestimmten Teilen vorgenommen werden. Ermutigen Sie die Schülerinnen und Schüler, durchzuhalten und weiter zu machen, wenn sie verloren oder verwirrt scheinen. Wenn der Code fertiggestellt ist, wird er hoffentlich mehr Sinn ergeben. Falls nicht, bieten Sie eine Anleitung an, die hilft, die Logik hinter diesen Änderungen zu erklären.

Entfernet im Abschnitt Fall 0 den Befehl **myServo.write()**. Ihr werdet diesen Befehl später im Sketch hinzufügen. Für den Fall 0 müsst ihr den Wischer nach unten bewegen und ihn dazu bringen, unten zu bleiben. Dies ist die Aus-Position. Fügt einen Befehl hinzu, um die Variable **servoAngle** auf 0 zu setzen.

9) Als Nächstes müsst ihr die Verzögerungszeit einstellen. Wenn der Tastschalter gedrückt wurde, während sich der Scheibenwischer ganz oben befand, sollte es etwa eine Sekunde dauern, bis sich der Scheibenwischer in die untere Position bewegt hat. Fügt einen Befehl hinzu, der die Variable **delayTime** auf 1.000 Millisekunden einstellt

10) Geht zu Fall 1 über, welcher der Einschaltmodus für die Scheibenwischer ist. Genau wie im Fall 0 werdet ihr Variablen verwenden, um das Verhalten des Wischers einzustellen. Löscht die vier Befehle, die sich in Fall 1 befinden.

11) Im eingeschalteten Zustand wechselt der Wischer abwechselnd zwischen den Positionen oben (Winkel 179) und unten (Winkel 0). Wenn der Wischer unten ist, muss er sich nach oben bewegen; wenn der Wischer oben ist, muss er sich nach unten bewegen. Ihr könnt dieses Verhalten mit einem if- else-Statement erzeugen, um die Variable **servoAngle** entweder auf 0 oder 179 Grad zu setzen. Erstellt eine Wenn-Sonst-Bedingung, die wie folgt aussieht:

12) Ihr müsst auch die Verzögerungszeit für den Ein-Modus einstellen. Der Scheibenwischer muss sich ohne Pausen hin- und herbewegen. Da der Scheibenwischer etwa eine Sekunde benötigt, um sich entweder nach oben oder nach unten zu bewegen, fügt ihr einen Befehl hinzu, der die Variable **delayTime** auf 1.000 Millisekunden setzt.

13) Fall 2 ist der intervallgeschaltete Modus. Das Verhalten ist das gleiche wie beim Einschaltmodus, außer dass die Verzögerungszeit, nachdem sich der Wischer nach unten bewegt, auf dem Potentiometerwert basiert. Löscht in Fall 2 die Befehle **myServo.write()** und **delay()**. Löscht nicht den Befehl **map()**, der die Variable **delayTime** einstellt.

14) Ähnlich wie in Fall 1 wechselt der Wischer abwechselnd zwischen den Positionen oben (Winkel 179) und unten (Winkel 0). Wenn der Wischer unten ist, muss er sich nach oben

bewegen; wenn der Wischer oben ist, muss er sich nach unten bewegen. Kopiert euer if-else-Statement aus Fall 1 und fügt es in Fall 2 ein.

15) Der Unterschied zwischen Fall 1 und Fall 2 ist die Verzögerungszeit für den intervallgeschalteten Modus. Wenn der Wischer unten ist (Winkel 0) und sich nach oben bewegt, muss die Variable **delayTime** auf eine Sekunde oder 1.000 Millisekunden eingestellt werden. Fügt in den if-Teil des if-else-Statements einen Befehl ein, um die Variable **delayTime** auf 1.000 Millisekunden einzustellen

Hinweis: Da es nun mehrere Befehle innerhalb des if-Teils der Bedingung gibt, zeigt das Beispiel hier jeden Befehl in einer separaten Zeile.

16) Wenn der Servo oben ist (Winkel 179) und sich nach unten bewegt, muss die Variable **delayTime** entsprechend dem Potentiometer eingestellt werden. Ihr solltet dafür bereits einen **map()**-Befehl haben. Verschiebt den Befehl, der die **delayTime**-Variable auf den abgebildeten Wert des Potentiometers setzt, in den anderen Teil der Bedingung.

Hinweis: Auch hier ist zu beachten, dass die Befehle des anderen Teils der Bedingung auf mehreren Zeilen geschrieben wurden.

17) Eure drei Fälle, die den Wischermodus steuern, sollten nun abgeschlossen sein. Überprüft und debugged euren Code, falls erforderlich.

18) Nach der schließenden Klammer für eure Switch-Case-Funktion müsst ihr den Befehl zum Bewegen des Servos hinzufügen. Verwendet einen **myServo.write()**-Befehl, um den Servo auf die in der Variablen servoAngle gespeicherte Position einzustellen

Hinweis: Wenn ihr eurem Servo-Objekt einen anderen Namen als "myServo" gegeben habt, ersetzt ihr den "**myServo**"-Teil des Befehls durch den Namen, den ihr eurem Servo-Objekt am Anfang eures Sketches gegeben habt.

19) Nachdem dem Servo der Befehl zur Bewegung gegeben wurde, muss der Sketch pausieren, während sich der Servo bewegt. Ihr verwendet den internen Timer des Arduino Boards, um die Pause zu erstellen. Es gibt keine Befehle zum Starten des internen Zeitgebers des Arduino Boards. Er startet, wenn das Board zurückgesetzt oder eingeschaltet wird. Daher könnt ihr den Timer nicht wie eine Stoppuhr verwenden, die von Null an zählt. Stattdessen muss euer Code auf die Zeit schauen, die ab einem bestimmten Punkt verstrichen ist. Um eine Verzögerung zu erzeugen, sollte euer Code den Wert des Zeitgebers in einer Variablen speichern und dann die verstrichene Zeit mit der Zeit vergleichen, die in der Variablen gespeichert wurde.

Der Timer des Arduino Boards zählt in Millisekunden. Der Befehl **millis()** wird verwendet, um den Timer des Arduino Boards abzulesen. Obwohl es keine Parameter gibt, die innerhalb der Klammern stehen, ist es wichtig, die offenen und geschlossenen Klammern einzufügen, um den Befehl abzuschließen.

Erstellt nach eurem Befehl **myServo.write()** einen Befehl zum Speichern der Zeit vom Arduino UNO R3 Board in der Variablen timerCount, die ihr zu Beginn dieser Übung erstellt habt.

20) Bei der Programmierung ist es oft nützlich, bestimmte Codezeilen sich wiederholen zu lassen. **Schleifen** sind Code-Strukturen, die es euch ermöglichen, Befehle eine bestimmte Anzahl von Malen zu wiederholen oder bis eine bestimmte Bedingung erfüllt ist. In der Arduino IDE gibt es mehrere verschiedene Arten von Schleifen, die erstellt werden können.

Loop	Beschreibung	Code-Beispiel

while()	Wiederholt einen Abschnitt des Codes, während eine Bedingung wahr ist. Wenn die Bedingung falsch wird, wird die Schleife nicht weiter laufen.	Dieser Code druckt ununterbrochen "Es ist kalt." in dem seriellen Monitor, während die Temperatur vom Temperatursensor weniger als 0 Grad ist.Wenn die Temperatur 0 Grad oder mehr erreicht, wird die Schleife nicht weiter laufen.
do-while()	Wie eine while-Schleife, die einen Codeabschnitt wiederholt, während eine Bedingung wahr ist. Die Bedingung wird jedoch nicht am Anfang, sondern am Ende der Schleife geprüft. Dadurch wird sichergestellt, dass die Befehle innerhalb der Schleife mindestens einmal ausgeführt werden.	Diese Schleife wird einmal durchlaufen. Danach wird sie sich nur wiederholen und im seriellen Monitor "Es ist heiß." drucken, wenn der Temperatursensor mehr als 100 Grad misst.
for()	Wiederholt einen Abschnitt des Codes eine bestimmte Anzahl von Malen.	Diese Schleife wird 10 Mal durchlaufen. Jedes Mal wird die Schleife von der Variablen i gezählt. Die Anzahl der Messungen und die Temperatur werden beide im seriellen Monitor abgedruckt.

Im Moment werdet ihr euch hauptsächlich auf die while-Schleife konzentrieren. Wie ein if-Statement basiert eine while-Schleife auf einer Bedingung, die wahr ist. Die Bedingung wird in Klammern gesetzt. Wenn die Bedingung wahr ist, werden die Befehle innerhalb der while-Schleife wiederholt. Wenn die Bedingung unwahr wird, wird die Schleife verlassen und der Code setzt am Ende der Schleife fort.

Eure while-Schleife muss eine Schleife basierend auf der Anzahl der Millisekunden in der **delayTime**-Variablen bilden. Damit die Schleife für die Anzahl der Millisekunden in **delayTime** ausgeführt wird, sollte die Bedingung die aktuelle Zeit auf dem Timer mit der Variablen **timerCount** plus delayTime vergleichen. Hier ist ein Beispiel mit Pseudocode.

Pseudocode	Beispiel
Stellt die Variable **delayTime** ein.	**delayTime** beträgt 1.000 Millisekunden
Bewegt den Servo nach oben.	Der Servo bewegt sich nach oben.
Speichert die Zeit auf dem Timer des Arduino als **timerCount**-Variable	Wenn der Timer von Arduino 5 Sekunden lang gelaufen ist, würde die Variable timerCount 5.000 betragen
Wiederholt dies, solange die aktuelle Zeit auf dem Timer kleiner ist als die gespeicherte **timerCount**-Variable plus die delayTime-Variable.	Die Schleife wiederholt sich, bis der Timer des Arduino 6.000 Millisekunden beträgt. Das sind 1.000 Millisekunden (1 Sekunde), nachdem die TimerCount-Variable gespeichert wurde.
Wiederholt Befehle in der Schleife.	Die Befehle werden 1 Sekunde lang wiederholt.
Beenden der Schleife.	Die Schleife endet.

Erstellt nach dem Befehl zum Festlegen der Variable **timerCount** eine while-Schleife, die wie folgt aussieht:

21) Innerhalb der while-Schleife muss der Sketch den Status des Ein-/Aus-Schalters überwachen. Beginnt mit der Eingabe eines **digitalRead()**-Befehls, um den Status der Ein-/Aus-Schaltfläche in der Variablen **buttonState** zu speichern, die ihr oben im Sketch erstellt habt. Erinnert euch, dass der Ein/Aus-Button mit dem **onButtonPin** (Pin 3) auf dem Arduino UNO R3 Board verbunden ist.

22) Als Nächstes muss im Sketch festgestellt werden, ob sich der Zustand der Schaltfläche geändert hat oder nicht. Oben im Sketch habt ihr eine weitere Variable namens **lastButtonState** erstellt. Durch Vergleichen der Variablen **buttonState** und **lastButtonState** kann das Arduino feststellen, ob der Wischer den Modus wechseln soll. Die folgende Tabelle zeigt die vier möglichen Ergebnisse durch jedes Durchlaufen der while-Schleife.

buttonState	lastButtonState	Aktion
0 (Tastschalter ist oben)	0 (Tastschalter war oben)	Der Tastschalter ist nicht gedrückt worden, also sollte nichts passieren.
1 (Tastschalter ist unten)	0 (Tastschalter war oben)	Die Taste wurde gerade erst gedrückt, so dass der Scheibenwischer den Modus wechseln muss.
1 (Tastschalter ist unten)	1 (Tastschalter war unten)	Der Tastschalter wird immer noch gedrückt, also sollte nichts passieren. Dadurch wird verhindert, dass der Scheibenwischer den Modus wechselt, wenn die Taste gedrückt gehalten wird
0 (Tastschalter ist oben)	1 (Tastschalter war unten)	Der Tastschalter ist gerade losgelassen worden, also sollte nichts passieren.

Der Sketch muss nur dann etwas tun, wenn die Variable **buttonState** und die Variable **lastButtonState** unterschiedlich sind. Ihr könnt ein if-Statement verwenden, um diese Bedingung zu überprüfen. In der Arduino IDE bedeutet das Symbol != nicht gleich. Erstellt innerhalb der while-Schleife ein if-Statement wie das hier gezeigte.

23) Innerhalb des if-Statements, das ihr gerade erstellt habt, muss der erste Befehl darin bestehen, die Variable **lastButtonState** auf den gleichen Wert wie die Variable **buttonState** zu setzen. Dies ist wichtig, damit beim nächsten Durchlaufen der while-Schleife derselbe Tastendruck nicht den Wischermodus ändert.

24) Es gibt zwei Möglichkeiten, dass die Variablen **buttonState** und **lastButtonState** nicht identisch sind. Entweder wurde die Taste gerade gedrückt oder die Taste wurde gerade losgelassen. Der Wischermodus sollte sich nur ändern, wenn die Taste gerade gedrückt wurde. Die gute Nachricht ist, dass ihr den Code dafür bereits erstellt habt.

In der Nähe des Anfangs eurer **void loop()**-Funktion solltet ihr ein if-Statement haben, das prüft, ob die Taste gedrückt ist. Verschiebt diese gesamte bedingte Anweisung, einschließlich der darin enthaltenen Befehle, direkt unter den Befehl, wo ihr die Variable **lastButtonState** gleich der Variablen buttonState setzen. Stellt sicher, dass das gesamte if-Statement innerhalb des anderen if-Statements verschachtelt ist.

> **Hinweis:** Wenn ihr den Code vom oberen Rand der Funktion void loop() kopiert und hier eingefügt habt, stellt ihr sicher, dass ihr den Code in der Nähe des Anfangs löscht. Ihr braucht nicht den gleichen Code an beiden Stellen des Sketches.

25) Seht euch die verschachtelte Bedingung an, die ihr gerade verschoben habt. Die Bedingung prüft den Pin auf dem Arduino UNO R3 Board um zu sehen, ob der Tastschalter gedrückt ist. Aber ihr habt bereits einen Befehl geschrieben, dies am Anfang der while-Schleife zu tun. Es besteht keine Notwendigkeit, den Pin erneut zu überprüfen. Stattdessen könnt ihr einfach die Variable **buttonState** verwenden. Ersetzt den Befehl **digitalRead()** in der Bedingung durch die Variable buttonState.

TEACHER NOTES

Den Code in dieser while-Schleife zu verstehen, ist vielleicht die komplizierteste Logik, die die Schülerinnen und Schüler in diesem Kurs erleben werden. Wenn die Schülerinnen und Schüler Schwierigkeiten haben, sollten Sie sich etwas mehr Zeit nehmen, um die am Ende dieser Lektion beschriebene Übung zur Erweiterung der **Schleifenlogik** abzuschließen.

26) Der Code ist fast vollständig. Da sich die while-Schleife sehr schnell wiederholt, ist es möglich, dass das Arduino einen Tastendruck als doppelten Tastendruck registriert. Dies würde dazu führen, dass der Code einen der Wischermodi überspringt. Um dies zu vermeiden, könnt ihr eine kurze Verzögerung innerhalb der verschachtelten Bedingung verwenden. Fügt eine Verzögerung von 50 Millisekunden vor der schließenden Klammer der verschachtelten Bedingung hinzu.

Hinweis: Wenn ihr feststellt, dass euer Code beim Drücken der Taste einen Wischermodus konsequent überspringt, könnt ihr diese Verzögerung auf 40 oder sogar 30 Millisekunden verkürzen

27) VÜberprüft euren Code und debugged ihn, falls erforderlich. Eure gesamte **void loop()**-Funktion sollte ähnlich wie diese aussehen:

28) Verbindet euer Arduino UNO R3 Board über das USB-Kabel mit Ihrem Computer.

29) 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.

30) Klickt auf den Button Hochladen, um den Sketch auf euer Arduino UNO R3 Board hochzuladen. Wenn der Upload abgeschlossen ist, wird euer Sketch automatisch ausgeführt.

31) Drückt die Taste in eurer Schaltung, um die Scheibenwischer zwischen Aus, Ein und intervallgeschaltetem Modus umzuschalten. Startet den seriellen Monitor, um zu beobachten, wie sich die Variablen ändern, wenn ihr den Wischer zwischen den Modi umschaltet. Ihr solltet feststellen, dass es viel einfacher ist, den Scheibenwischer zwischen den Modi umzuschalten, wenn ihr den eingebauten Timer des Arduino UNO R3 Boards verwendet und Variablen zur Verfolgung des Tastenstatus verwendet. Hier sind ein paar Dinge zum Ausprobieren:

- Drückt die Taste zweimal schnell. Wechselt der Scheibenwischer zweimal den Modus?
- Drückt und haltet die Taste gedrückt. Wechselt der Scheibenwischer weiterhin den Modus?
- Drückt die Taste so schnell ihr könnt und lasst sie wieder los. Könnt ihr sie schneller als 50 Millisekunden drücken und loslassen, so dass der Tastendruck nicht registriert wird?

Programmierer verbringen viel Zeit mit der Suche nach Möglichkeiten, den Code zu verbessern und die Geräte funktionaler und benutzerfreundlicher zu gestalten. Deshalb ist es für Programmierer wichtig, gute Problemlöser zu sein. Sie müssen in der Lage sein, Probleme aus verschiedenen Blickwinkeln anzugehen und eine Lösung zu finden, die den Bedürfnissen der Benutzer am besten entspricht.

32) Speichert euren Sketch.

33) Zieht das USB-Kabel von eurem Arduino UNO R3 Board ab, um euren Schaltkreis auszuschalten.

34) Blättert zu Lektion 7 Seite 20 in eurem Arbeitsheft. Vergleicht die Effizienz eures Wischersketches, bei dem Verzögerungen zum Anhalten des Wischers verwendet wurden, mit eurem neuen Sketch, bei dem die Zeitschaltuhr des Arduino UNO R3 Boards verwendet wird. Erklärt in euren eigenen Worten das Problem mit der Verwendung von Verzögerungen und wie dieses Problem mit Hilfe des Arduino-Timers und einer Zeitschleife gelöst wurde.

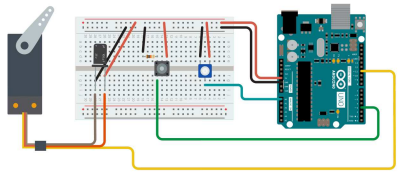
Ändern der Schaltung - Waschmodus

Bislang habt ihr einen Scheibenwischer mit drei Modi gebaut und programmiert - aus, ein und intervallgeschaltet. Die meisten modernen Fahrzeuge verfügen jedoch über einen vierten Wischermodus. Sie verfügen in der Regel über einen Knopf, den ihr drücken könnt, um Scheibenwaschflüssigkeit zum Reinigen der Scheibe zu versprühen. Bei dieser Übung fügt ihr dem Kreislauf eine blaue LED hinzu, die eine Pumpe darstellt, die Waschflüssigkeit ausgibt.

FURTHER NOTES

Fügt ihr die Schaltung und eurem Sketch einen Scheibenwaschmodus hinzu. Dem Kreislauf wird eine blaue LED hinzugefügt, um das Versprühen von Wischerflüssigkeit zu simulieren. Ihr modifizierr den Sketch dann so, dass die Wischer ihre Reinigungsbewegung eine bestimmte Anzahl von Malen mit einer for-Schleife wiederholen.

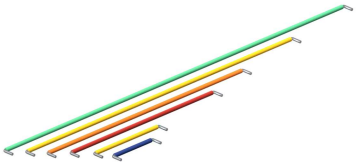
Benötigte Materialien



1 SCHEIBENWISCHERKREIS AUS VORHERIGER ÜBUNG

1 BLAUE LED

1 220 Ω-WIDERSTAND



1 STECKVERBINDER

1 USB-KABEL

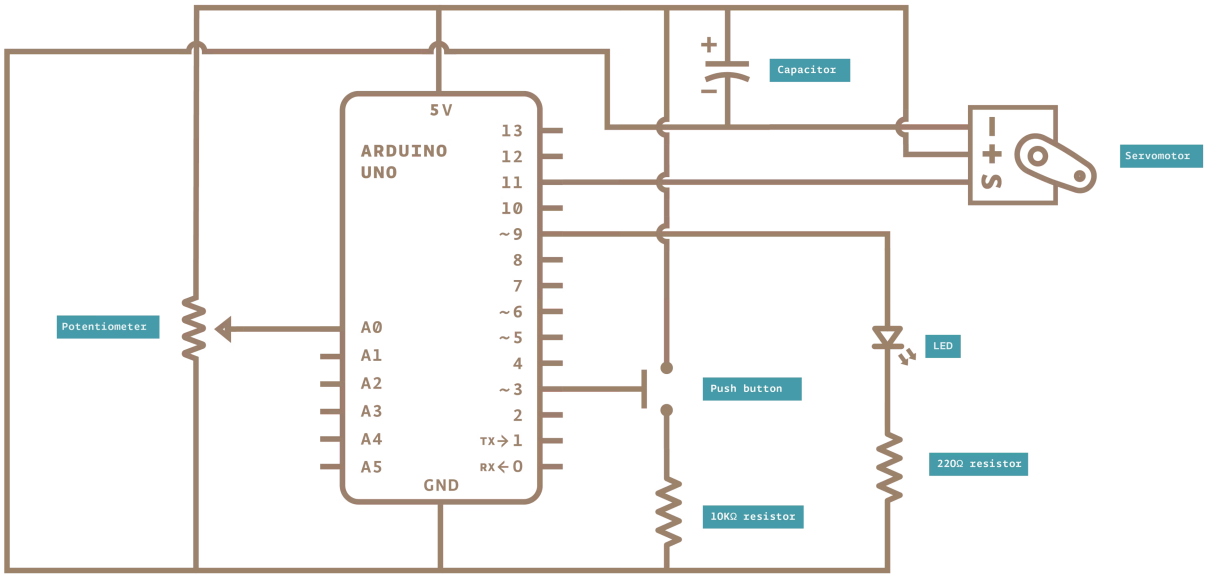
FURTHER NOTES

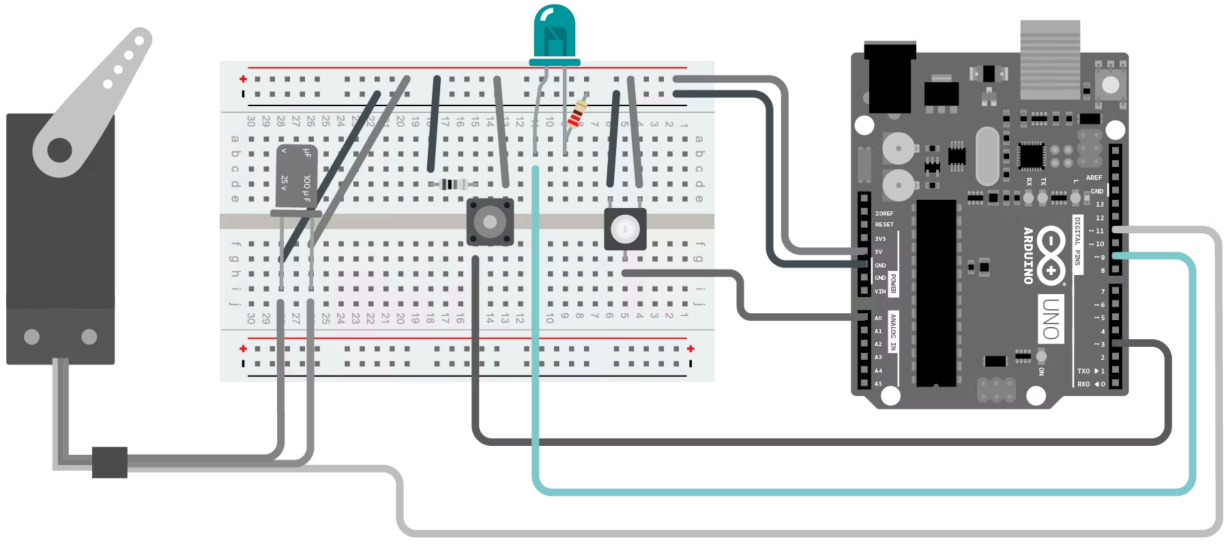
Die Farbbänder des 220-Ohm-Widerstands sind:

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

Schaltplan

Verwendet das Schaltbild und den Schaltplan, um euren Wischerkreis durch Hinzufügen einer blauen LED zu modifizieren. Die Anode der blauen LED sollte mit Pin 9 auf dem Arduino UNO R3 Board verbunden werden. Die Kathode der LED sollte an einen Widerstand mit 220 Ω angeschlossen werden, der die LED mit Ground verbindet.





TEACHER NOTES

Es gibt keine direkte Anweisung, die blaue LED in die Schaltung einzubauen. Inzwischen haben die Schülerinnen und Schüler mehrere Möglichkeiten gehabt,

LEDs von dem Arduino UNO R3 Board aus zu verdrahten und zu steuern. Sie sollten in der Lage sein, auf ihre bisherigen Erfahrungen zurückzugreifen oder den Schaltplan und den Verdrahtungsplan zu verwenden, um die Schaltung zu vervollständigen.

Ändern des Codes - Waschmodus

Nun, da eure Schaltung über eine blaue LED für den Waschmodus verfügt, müsst ihr das Verhalten des Waschmodus programmieren. In dieser Übung modifiziert ihr euren Sketch so, dass er einen vierten Modus enthält, der den Waschmodus eines Scheibenwischers simuliert. Für diese Übung benötigt ihr euer Arbeitsheft.

FURTHER NOTES

Wir fügen unserer Switch-Case-Struktur einen vierten Fall hinzu, der den Scheibenwischer in den Waschmodus versetzt. Für diese Übung benötigen wir das Arbeitsheft.

TEACHER NOTES

Während dieser Übung werden die Schülerinnen und Schüler mit diesem Kodierungskonzept vertraut gemacht:

- **for(*iInitialisierung; Bedingung; Schrittweite*){}** – erzeugt eine Schleife, die eine bestimmte Anzahl von Malen wiederholt

1) Öffnet gegebenenfalls euren Sketch zu Lektion7_V4 in der Arduino IDE.

2) Speichert den Sketch unter einem neuen Namen. Aus dem Dateimen wählt ihr **Speichern unter**... Benennt die Datei "Lektion7*V5*", gefolgt von euren Initialen. Klickt auf Speichern.

3) Ordnet euer Lektion7_V5-Sketchfenster und dieses Fenster so an, wie es für euch am Besten geeignet ist. .

4) Wendet euch an den Abschnitt für Lektion 7 in eurem Arbeitsheft. Denkt über die Funktionsweise eines Scheibenwischers nach, wenn er sich im Waschmodus befindet. Die Waschflüssigkeit muss abgegeben werden, die Scheibenwischer bewegen sich vier oder fünf Mal hin und her, um die Scheibe zu reinigen, und dann schalten sich die Flüssigkeit und die Scheibenwischer automatisch ab. Euer Arbeitsheft enthält die Pseudocodeschritte, die programmiert werden müssen, um den Waschmodus zu eurer Schaltung hinzuzufügen. Die Schritte sind jedoch nicht in der richtigen Reihenfolge. Ordnet die Schritte in der logischsten

Reihenfolge auf der Grundlage eures aktuellen Sketches an. Um die Schritte anzuordnen, nummeriert ihr jeden Schritt von 1 bis 8.

5) In eurem Schaltkreis werdet ihr die blaue LED mit einem digitalen Pin auf dem Arduino UNO R3 Board steuern. Am Anfang eures Sketches deklariert ihr Pin 9 als eine Konstante mit dem Namen **LEDPin**.

6) Fügt in der Funktion **void setup()** einen Befehl hinzu, um den Pin-Modus des **LEDPin** als Ausgang zu deklarieren.

7) Geht zur Funktion **void loop()** in eurem Sketch. Erstellt unter dem Unterbrechungs-Befehl für Fall 2 Fall 3. Dieser Fall wird das Wischverhalten für den Waschmodus definieren. Fahrt fort und fügt auch den Unterbrechungs-Befehl für Fall 3 ein. Ihr fügt die Codezeilen für den Wischermodus zwischen diesen beiden Zeilen ein.

8) Das Erste, was im Waschmodus zu tun ist, ist mit dem Sprühen von Waschflüssigkeit auf die Scheibe zu beginnen. In eurem Schaltkreis werdet ihr dies simulieren, indem ihr die blaue LED einschaltet. Fügt einen **digitalWrite()**-Befehl hinzu, um **LEDPin** auf HIGH zu setzen.

9) Nachdem die Waschflüssigkeit verteilt ist, müsst ihr die Wischer starten. Bei den meisten Fahrzeugen bewegen sich die Wischer drei- bis fünfmal hin und her, um die Scheibe mit der Wischerflüssigkeit zu reinigen. Um diese Bewegung zu erzeugen, könnt ihr eine for-Schleife verwenden. Die for-Schleife wiederholt eine Reihe von Befehlen eine bestimmte Anzahl von Malen. Ein Zähler wird verwendet, um die Anzahl der Male zu zählen, die die Befehle ausgeführt wurde, und wenn der Zähler beendet ist, endet die Schleife. Die Struktur einer for-Schleife sieht wie folgt aus:

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

Die **Initialisierung** erzeugt eine Steuervariable für die Schleife und setzt ihren Wert fest. Sie funktioniert genauso wie die Deklaration einer Variablen und das Festsetzen ihres Wertes. Der Bedingungsteil der for-Schleife ist ein Test, um festzustellen, ob die Schleife sich weiterhin wiederholen soll. Die Bedingung funktioniert genauso wie die Bedingung in einem if-Statement oder einer while-Schleife. Solange die Bedingung wahr ist, werden die Befehle innerhalb der geschweiften Klammern der for-Schleife ausgeführt. Die Schrittweite der Schleife verändert die Steuervariable um einen bestimmten Betrag. Ohne die Schrittweiten-Anweisung würde die Schleife wahrscheinlich kontinuierlich durch laufen, ohne anzuhalten.

Hinweis: Die Buchstaben *i* und *x* werden häufig als Steuervariablen in for-Schleifen verwendet. Ihr könnt jedoch jeden gültigen Variablennamen als Steuervariable verwenden. Ihr könnt auch Variablen verwenden, die außerhalb der for-Schleife initialisiert werden, wie z.B. globale Variablen.

Erstellt nach eurem **digitalWrite()**-Befehl eine for-Schleife. Führt innerhalb der Klammern der for-Schleife Folgendes aus:

- Initialisiert die Variable **i** als Ganzzahlvariable und setzt sie gleich 0
- Bestimmt die Bedingung als **i < 5**. Dadurch wird die Schleife fünf Mal wiederholt
- Stellt die Schrittweite als **i = i + 1** ein. Dadurch wird der Wert von i jedes Mal, wenn die Schleife durchlaufen wird, um eins erhöht.

Eure for-Schleife sollte wie folgt aussehen:

Hinweis: Eine kürzere Version der Schrittweite **i = i + 1** ist einfach **i++** einzugeben. Beide Befehle bewirken dasselbe.

10) Innerhalb der for-Schleife müsst ihr Befehle hinzufügen, um den Servo bei 179 Grad in die obere Position und dann bei 0 Grad zurück in die untere Position zu bewegen. Fügt nach

jedem Befehl Verzögerungsanweisungen ein, um dem Servo Zeit zum Bewegen zu geben, bevor der nächste Befehl ausgeführt wird.

Hinweis: Fall 3 (Waschmodus) unterscheidet sich von den anderen Fällen. Im Waschmodus muss im Sketch nicht wie in den anderen Modi die Ein-/Ausschalttaste überwacht werden. Dies liegt daran, dass der Waschmodus unabhängig davon, ob der Knopf gedrückt wird oder nicht, seinen vollständigen Zyklus abschließen muss, um die Windschutzscheibe zu reinigen. Wenn der Waschmodus dann abgeschlossen ist, wechselt der Wischer automatisch in den Aus-Zustand. Da es nicht notwendig ist, den Status des Ein/Aus-Knopfes zu überprüfen, könnt ihr Verzögerungen nach jeder Servobewegung

11) Wenn die for-Schleife abgeschlossen ist und der Wischer mit der Reinigung der Windschutzscheibe fertig ist, muss die Wischerflüssigkeit abgestellt werden. Fügt einen **digitalWrite()**-Befehl zum Ausschalten der blauen LED hinzu, um das Ausschalten der Wischerflüssigkeit darzustellen. Fügt den Befehl vor dem Unterbrechungs-Befehl für Fall 3 hinzu.

12) Wenn der Wischer mit der Reinigung der Windschutzscheibe fertig ist, sollte der Wischer automatisch ausgeschaltet werden. Fügt vor dem Unterbrechungsbefehl für Fall 3 einen Befehl hinzu, um die Variable **wiperState** für off wieder auf 0 zu setzen.

13) Es gibt eine letzte Codezeile, die ihr ändern müsst. Da ihr jetzt vier statt drei Modi für den Scheibenwischer habt, müsst ihr die Ein-/Ausschalttaste zwischen allen vier Modi umschalten lassen. Geht zum Ende eures Sketches und findet euer verschachteltes if-Statement innerhalb der while-Schleife. Findet das **if-Statement**, das prüft, ob der Wischerzustand größer oder gleich 3 ist, und wenn ja, die Variable **wiperState** auf 0

zurücksetzt. Nachdem ihr nun Fall 3 hinzugefügt habt, müsst ihr die Bedingung dieses if-Statements von 3 auf 4 erhöhen.

.**14)** Überprüft euren Code und debugged ihn wenn nötig.

15) Verbindet euer Arduino UNO R3 Board über das USB-Kabel mit eurem Computer.

16) 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.

17) Klickt auf den Button Hochladen, um den Sketch auf euer Arduino UNO R3 Board hochzuladen. Wenn der Upload abgeschlossen ist, wird euer Sketch automatisch ausgeführt.

18) Startet den seriellen Monitor. Drückte die Taste auf eurer Schaltung, um zwischen Aus, Ein, Intervallschaltung und Waschmodus umzuschalten. Vergewissert euch, dass die blaue LED aufleuchtet, wenn sich der Wischer im Waschmodus befindet.

19) Wenn ihr euren Code und eure Schaltung überprüft haben, speichert ihr euren Sketch.

20) Zieht das USB-Kabel von eurem Arduino UNO R3 Board ab, um euren Schaltkreis auszuschalten.

21) Schließt alle Fenster der Arduino IDE.

22) Säubert euren Arbeitsbereich und lagert alle Geräte in einem geeigneten Lagerbereich.

TEACHER NOTES

Weitere Übungen

- **Schleifenlogik** – In dieser Lektion wurden einige der kompliziertesten Logiken vorgestellt, die in diesem Kurs zu finden sind. Den Schülerinnen und Schülern fällt es möglicherweise schwer, den Code vollständig zu verstehen. Wenn dies der Fall ist, führen Sie diese Übung als eine ganze Klasse durch. Diese Übung wird um die while-Schleife herum geschrieben, die die Schülerinnen und Schüler im Abschnitt "Testen und Ändern" der Lektion erstellen, aber Sie können die gleichen Prinzipien für jeden komplizierten Codeabschnitt verwenden.
  - Wenn möglich, projizieren Sie den Code der while-Schleife so, dass die Schülerinnen und Schüler ihn sehen können, wenn sie die Übung abschließen.
  - Drucken Sie die Namen der einzelnen Variablen, die in der while-Schleife verwendet werden, auf separate Blätter aus. Drucken Sie sie so groß wie

möglich aus, damit sie auch aus der Ferne gut lesbar sind.

- Drucken Sie jeden Befehl der while-Schleife auf einzelne Blätter aus. Fügen Sie den Befehl direkt vor der while-Schleife ein, die die **timerCount**-Variable gleich dem Wert des Arduino-Timers setzt. Drucken Sie sie so groß wie möglich aus.
- Distribute the commands to students and have them form a circle representing the loop.
- Verteilen Sie die Befehle an die Schülerinnen und Schüler und lassen Sie sie einen Kreis bilden, der die Schleife darstellt.
- Geben Sie den Schülern, die innerhalb des Kreises stehen, einen Namen für jede Variable. Jede Variable sollte von einem anderen Schüler verfolgt werden. Vielleicht möchten Sie sogar Werte ausdrucken, die jeder Variablen zugewiesen werden können.
- Gehen Sie als Klasse jeden Befehl der Schleife durch. Wenn jede Iteration durch die Schleife abgeschlossen ist, ändern Sie die Parameter, wie z.B. die Taste, die gedrückt oder losgelassen wird. Erhöhen Sie außerdem die Zeit auf dem Arduino-Timer jedes Mal, wenn Sie die Schleife durchlaufen, um einen bestimmten Betrag (beginnen Sie mit einer Sekunde).

Während die Schülerinnen und Schüler jede Iteration der Schleife wiederholen, lassen Sie sie Verbindungen zu dem herstellen, was in ihren Schaltkreisen vor sich geht. Wenn Sie nicht genügend Schüler haben, können Sie die Variablen und ihre Werte auf eine Pinnwand schreiben, so dass jeder sie sehen kann, anstatt sie den Schülern zuzuweisen, damit sie den Überblick behalten.

- **Waschtaste** – Die Schülerinnen und Schüler könnten inzwischen erkannt haben, dass es nicht sehr praktisch ist, durch den den Waschmodus durchzuschalten, um die Scheibenwischer auszuschalten. Tatsächlich haben die meisten Fahrzeuge keinen einzigen Knopf, mit dem zwischen den Wischermodi umgeschaltet werden kann. Sie haben verschiedene Schalter und Hebel zur Steuerung der Scheibenwischer. Lassen Sie die Schülerinnen und Schüler einen zweiten Knopf zum Wischerkreis hinzufügen. Anstatt die Schülerinnen und Schüler durch verschiedene Modi schalten zu lassen, lassen Sie sie den neuen Knopf programmieren, um den Wischermodus Ihres Wischers zu steuern.

- **Unterbrechungsstifte** – In dieser Lektion nutzten die Schülerinnen und Schüler Verzögerungen, um das Arduino UNO R3 Board anzuhalten, damit sie dem Wischerservo Zeit geben, sich auf seine Position zu bewegen. Dann ersetzten sie die Verzögerungen durch while-Schleifen, mit denen sie den Status des Ein-/Aus-Schalters überprüfen konnten, während sich der Servo in seine Position bewegte. Dadurch wurde die Wischerschaltung viel benutzerfreundlicher. Es gibt jedoch neben der Verwendung des internen Zeitgebers von Arduino noch eine dritte Methode, die hätte verwendet werden können. Die Pins 2 und 3 auf dem Arduino UNO R3 Board sind Interrupt-Pins. Die Schülerinnen und Schüler können den folgenden Link verwenden oder ihre eigenen Links finden, um zu recherchieren, wie Interrupt-Pins funktionieren. Sehen Sie dann, ob sie ihren Sketch so modifizieren können, dass sie Interrupt-Pins anstelle von Verzögerungen oder while-Schleifen zum Umschalten zwischen den Wischermodi verwenden. *
  https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/