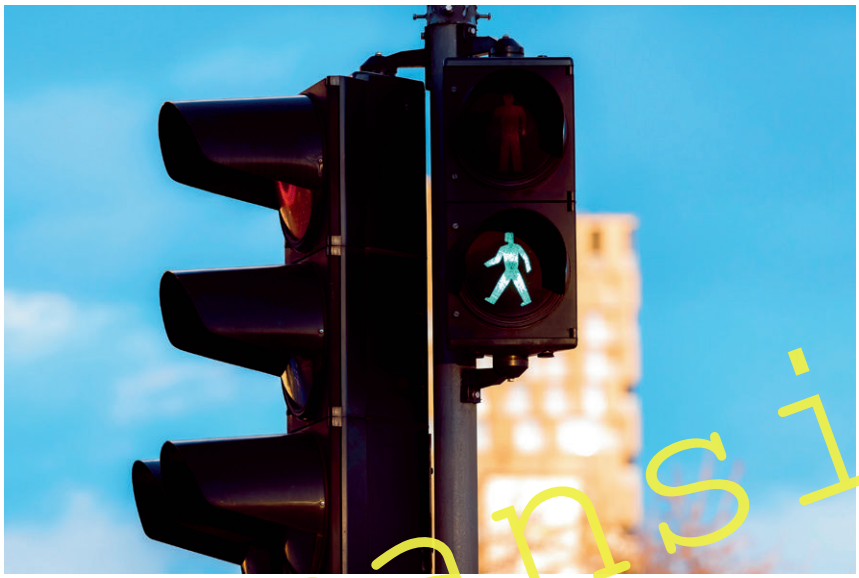
I.D.35

Elektrizitätslehre und Magnetismus

Vom Blinklicht zur Ampelschaltung – Programmieren mit dem Mikrocontroller Arduino

Thomas Rosenthal, Esslingen am Neckar
Illustrationen von: Dr. Wolfgang Zettlmeier



© Spyderskidoo/iStock/Getty Images Plus

In vielen technischen Geräten sind heutzutage Mikrocontroller verbaut. Ihre Schüler sollen dies am Beispiel der Programmierung einer Ampelschaltung an einer Kreuzung erkennen. Dabei können sie bei der Autofahrerampel eine Tag-und-Nacht-Situation simulieren, während bei der Fußgängerampel per Taster die Farbe Grün, begleitet von einem Ton, angefordert werden kann. Dabei lernen sie auch elektronische Bauteile und deren Zusammenführung zu elektronischen Schaltungen kennen.

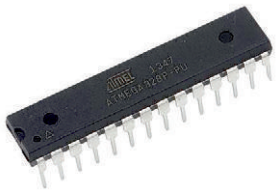
KOMPETENZPROFIL

Klassenstufe/Lernjahr:	9/10
Dauer:	10 Unterrichtsstunden
Kompetenzen:	1. Fachwissen: Mikrocontroller kennenlernen; 2. Erkenntnisgewinnung: einzelne Bauteile programmieren und in einem Projekt zusammenführen; 3. Methodik: elektronische Schaltungen stecken und Schaltpläne erstellen
Thematische Bereiche:	elektronische Schaltungen, Programmierung
Medien:	Arbeitsblätter, Farbfolie, Programmcodes
Zusatzmaterialien:	Lösungen und Erweiterungsmöglichkeiten zur Differenzierung

Sachanalyse

Zum Mikrocontroller im Allgemeinen und zum Arduino im Besonderen

Als Mikrocontroller (auch µController, µC) bezeichnet man **Halbleiterchips**, die neben einem Prozessor zugleich auch Peripheriefunktionen besitzen. In vielen Fällen befindet sich auch der Arbeits- und Programmspeicher teilweise oder komplett auf demselben Chip. Ein Mikrocontroller ist ein Ein-Chip-Computersystem und hat deshalb eher eine geringe Leistungsfähigkeit. Er muss oft auch nur bestimmte, wenige Prozesse ausführen. Diese funktionieren nach dem EVA-Prinzip: Eingabe – Verarbeitung – Ausgabe:



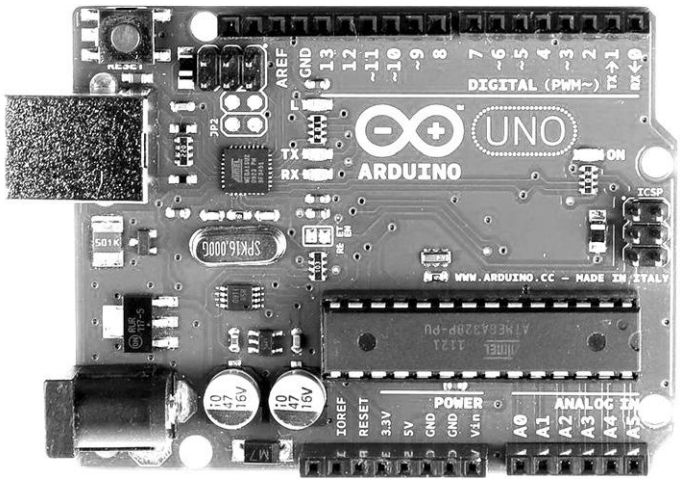
Quelle: de.rs-online.com
© Arduino www.arduino.cc

1. **Eingabe**: Damit eine Datenverarbeitung überhaupt stattfinden kann, müssen zunächst einmal Daten vorhanden sein. Diese lassen sich über Tastatur, Maus, Gamepad, Scanner, Mikrofon oder Webcam in das Computersystem eingeben.
2. **Verarbeitung**: Nach der Eingabe von Daten in ein Computersystem kann die Recheneinheit (CPU, Prozessor, Controller) darauf zugreifen. Die CPU, die aus Speicher, Steuer- und Rechenwerk besteht, berechnet aus der Dateneingabe die Datenausgabe. Für die nötige Berechnung oder zur späteren Aufbewahrung werden die Daten (zwischen-)gespeichert. Die gängigsten Speicher sind: Festplatte, SSD, Arbeitsspeicher (RAM), ROM, CD, DVD, SD-Karte oder USB-Stick.
3. **Ausgabe**: Damit die berechneten Daten nun zur Verfügung stehen, müssen sie in einer bestimmten – der jeweils gewünschten – Form wieder ausgegeben werden. Dies erfolgt am häufigsten durch Bildschirm, Drucker, Lautsprecher oder Beamer.

Beim Arduino handelt es sich um eine aus Soft- und Hardware bestehende **Physical-Computing-Plattform.**

Im Sinne von Open Source sind beide Komponenten quelloffen. Dabei besteht die Hardware aus einem einfachen Eingabe-/Ausgabe(EA)-Board mit einem Mikrocontroller, der analoge und digitale Ein- und Ausgänge hat. Die kostenlose Software als zugehörige Entwicklungsumgebung basiert auf Processing und erleichtert auch technisch weniger Versierten den Zugang zur Programmierung im Allgemeinen und zu Mikrocontrollern im Besonderen.

Die Programmierung selbst erfolgt in einer C bzw. C++ ähnlichen Programmiersprache; umfangreiche Bibliotheken und Beispiele vereinfachen die Programmierung erheblich. Das erste Board wurde im Jahre 2005 von Massimo Banzi und David Cuartielles entwickelt. Der Name „Arduino“ stammt von der Bar in Ivrea (Italien), in der sich einige der Projektgründer oft trafen.



© Arduino www.arduino.cc

Der Name der Bar selbst erinnert an den italienischen König von 1002 bis 1014 namens Arduin von Ivrea. David Mellis entwickelte später die auf C/C++ basierende Programmiersprache dazu. Später wurde alles im Netz veröffentlicht und unter eine Creative-Commons-Lizenz gestellt. Die erste Auflage betrug 200 Stück, von denen 50 Stück an eine Schule gingen.

Zur Ampelsteuerung

Am 10. Dezember 1886 ging die erste Lichtsignalanlage der Welt in Betrieb, aufgestellt auf dem Parliament Square in London. Sie musste mit Gaslicht betrieben werden und explodierte bereits nach kurzer Zeit.
Erst nach der Verbreitung des elektrischen Lichts in den europäischen Großstädten konnte man ab dem Jahre 1912 wieder Lichtsignalanlagen zur Verkehrsregelung verwenden. Die am 5. August 1914 installierte Lichtsignalanlage in Cleveland, USA, ist die erste elektrische Verkehrsampel der Welt. Diese hatte nur zwei Lampen, nämlich rot und grün. Die ersten dreifarbigen Lichtsignalanlagen gab es 1920 in Detroit und New York, während in Europa die erste dreifarbige Lichtsignalanlage im Jahre 1922 in Paris an der Rue de Rivoli/Boulevard de Sébastopol eingerichtet wurde. Mit seinem charakteristischen Verkehrsturm ging in Deutschland die erste Lichtzeichenanlage am Potsdamer Platz in Berlin am 15. Dezember 1924 in Betrieb (vgl. Abbildung).

Bundesarchiv, Bild 102-01702 / CC-BY-SA 3.0

Handbücher zum Arduino

- ***Bartmann, Erik:*** *Mit Arduino die elektronische Welt entdecken, Bombini Verlag, 2017*
  Mit diesem Buch erhält man eine fundierte Einführung in die Grundlagen der Arduino-Programmierung und in die Elektronik zugleich, durch viele alltagsnahe Beispiele illustriert. Außerdem enthält es insgesamt 44 sehr detailliert beschriebene Projekte zum Erproben.
- ***Brühlmann, Thomas:*** *Arduino Praxiseinstieg, mitp Verlags GmbH & Co KG, 2017*
  Alle Komponenten der Hard- und Software werden in diesem Buch ebenso sehr verständlich beschrieben wie die notwendigen elektronischen Grundlagen. Zur praktischen Erprobung dienen zahlreiche und vielseitige Projekte.
- ***Brühlmann, Thomas:*** *Sensoren im Einsatz mit Arduino, mitp Verlags GmbH & Co KG, 2017*
  Für eine ausführliche Beschäftigung mit Sensoren: von Temperatur- über Infrarot-, Farb- und Ultraschallsensoren bis hin zum Einsatz von Kompass, GPS-Modul und Kamera liegt man mit diesem Buch richtig, ergänzt durch zahlreiche Beispielprojekte zum Ausprobieren.
- ***Geddes, Mark:*** *Arduino-Projekte, dpunkt.verlag, 2016*
  25 unterhaltsame und interaktive Projekte werden in diesem Buch vorgestellt. Jedes Projekt enthält präzise Anleitungen und vollständige Programm-Codes.
- ***Kappel, Benjamin:*** *Arduino Elektronik, Programmierung, Basteln, Rheinwerk Verlag, 2016*
  Auch mit diesem Buch erhält man eine fundierte Einführung in die Grundlagen der Arduino-Programmierung und in die Elektronik zugleich, durch viele alltagsnahe Beispiele illustriert. Außerdem enthält es mehrere detailliert beschriebene Projekte zum Erproben.

Didaktisch-methodisches Konzept

Zur Lerngruppe und den curricularen Vorgaben

Die Einführung in die Programmierung mit dem Arduino empfiehlt sich aus der Erfahrung heraus in der 9./10. Klassenstufe; in naturwissenschaftlich begabten Lerngruppen kann dies durchaus auch schon in der 8. Klassenstufe geschehen. Die guten Schüler können Sie als Unterstützer/Berater in den Unterricht einbinden bzw. ihnen Aufgaben geben, die eine Erweiterung des jeweiligen Projektes darstellen. Nicht alle Aufgaben auf den Arbeitsblättern müssen von allen Schülern in der gegebenen Reihenfolge bearbeitet werden; vielmehr stellen einzelne Aufgaben mögliche **Differenzierungsangebote** dar. Mehr und mehr gewinnt in den MINT-Fächern die Programmierung mit einem Mikrocontroller an Bedeutung. So heißt es beispielsweise im Bildungsplan des Faches Naturwissenschaft und Technik (NwT) in Baden-Württemberg: „Natürliche Vorgänge und technische Prozesse laufen häufig gesteuert oder geregelt ab. Die Schüler lernen die Prinzipien der Steuerung und Regelung kennen und entdecken, dass diese bestimmten Algorithmen folgen." Aber auch die Vermittlung von Grundlagen der Elektronik steht im Vordergrund, es geht einerseits um die Beschreibung der Funktion von Bauteilen elektrischer oder elektronischer Schaltungen und andererseits um die Realisierung elektrischer oder elektronischer Schaltungen.

Methodischer Schwerpunkt der Unterrichtsreihe

Die Ampel dient als Beispiel aus der Lebensumgebung der Schüler, um sie in die Programmierung eines Mikrocontrollers einzuführen. Die Schüler lernen außerdem einzelne elektronische Bauteile und deren Funktionsweise kennen und können diese in einem Projekt zusammenführen. Eine schrittweise Vorgehensweise ist dabei zwingend erforderlich: Nach der Besprechung der theoretischen Grundlagen sollte vor der eigentlichen Programmierung der Aufbau der elektronischen Schaltungen erfolgen, um eine getrennte Fehlersuche zu ermöglichen. *Sämtliche Schaltungen und Programme (Sketches) finden Sie in gesonderten Ordnern auf der beiliegenden **CD-ROM 57**. Bitte beachten Sie dabei die konsequente und nützliche Verzeichnisstruktur.*

Organisatorische Hinweise

Eine gute Organisation ist das A und O im Unterricht: Da ist zunächst einmal die Aufbewahrung: Empfehlenswert sind Kisten, in denen die Arduino-Starterkits mit jeweils innen liegenden Namensschildern aufbewahrt werden, damit das Austeilen zu Beginn einer Stunde zügig funktioniert. In den Boxen sollten neben dem Arduino auch das Steckbrett und das USB-Kabel liegen. Zu klären ist die Frage, ob die Schule USB-Sticks zur Verfügung stellt oder ob jeder Schüler einen im Mäppchen bei sich hat. Die elektronischen Bauteile sollten sortiert in kleinen Boxen gesondert aufbewahrt werden. Um Kabelsalat zu vermeiden, sollten die elektronischen Bauteile – getrennt nach Farbe und Länge – in kleinen, stapelbaren Boxen aufbewahrt werden.

Mögliche Alternativen oder Erweiterungsmöglichkeit

Es gibt natürlich durchaus Alternativen der Einführung in die Arduino-Programmierung mit anderen Projekten wie z. B. den Bau eines Reaktionstesters. Dort könnten Sie ebenfalls die einzelnen Komponenten besprechen. Die Ampelsteuerung ist jedoch ein Beispiel aus der Lebenswelt der Schüler. Erweiterungsmöglichkeiten gibt es viele: In den aufgeführten Handbüchern gibt es zahlreiche, sehr gut beschriebene Projekte mit unterstützenden Anleitungen und fertigen Programmen (Sketches).

Auf einen Blick

1. Stunde (Klassenzimmer)

Thema: **Mikrocontroller im Alltag – Farbfolie**

M 1 (Fo) **Mikrocontroller im Alltag** / Gedankenaustausch über technische Geräte und ihre Funktionsweisen, die sich auf den Einsatz von Mikrocontrollern beziehen.

M 2 (Tx) **Mikrocontroller im Alltag** / Austausch über die Funktionsweise eines Mikrocontrollers an einem praktischen Beispiel und Verallgemeinerung auf das EVA-Prinzip.

Benötigt:
- ☐ OH-Projektor / Beamer / Visualiser / Whiteboard
- ☐ Arduino-Boxen und ggf. USB-Sticks
- ☐ mehrfach ausgedrucktes und entsprechend der Schülerzahl zugeschnittenes Word-Dokument „Arduino – Organisation“

2./3. Stunde (Computerraum)

Thema: **Aufbau und Funktionsweise des Arduino /** Die Arduino-Softwareumgebung

M 3 (Tx) **Aufbau und Funktionsweise des Arduino-Boards** / Kennenlernen und Verständnis für den Aufbau und die Funktionsweise des Arduino-Boards

M 4 (Tx) **Die Arduino-Softwareumgebung – Teil 1** / Kennenlernen und Verständnis für den Aufbau und die Funktionsweise der Arduino-Softwareumgebung

M 5 (Tx) **Die Arduino-Softwareumgebung – Teil 2** / Kennenlernen und Verständnis für den Aufbau und die Funktionsweise der Arduino-Softwareumgebung

M 6 (Tx) **Ein erstes Blinken einer LED** / Kennenlernen und Verständnis für den Aufbau eines Programmcodes (Sketches) in der Arduino-Softwareumgebung

Benötigt:
- ☐ OH-Projektor / Beamer / Visualiser / Whiteboard
- ☐ digitale Fassung von Schaltplan „01_Blinkende_LED“
- ☐ Arduino-Boxen und USB-Sticks
- ☐ 1 rote LED für jeden Schüler

4./5. Stunde (Computerraum)

Thema: **Verwendung eines Steckbrettes und Autofahrerampel** / Verwendung von Variablen und Ampelkreuzung

M 7 (Tx) **Verwendung eines Steckbrettes und Autofahrerampel** / Kennenlernen des Steckbrettes, Schaltung und Programmierung einer Ampelschaltung

M 8 (Tx) **Verwendung von Variablen und Fußgängerampel** / Kennenlernen der Vereinfachung eines Programmes unter Verwendung von Variablen, Schaltung und Programmierung einer Fußgängerampel

Benötigt:
- ☐ OH-Projektor / Beamer / Visualiser / Whiteboard
- ☐ digitale Vorlage des Word-Dokumentes „Zur Geschichte der Ampel“
- ☐ Kopie oder digitale Vorlage des Word-Dokumentes „AB Widerstände“
- ☐ digitale Fassungen von Schaltplänen „02_Autofahrer-Ampel“ und „04_ Ampelkreuzung“
- ☐ Arduino-Boxen und USB-Sticks
- ☐ 2 rote, 1 gelbe, 2 grüne LEDs, fünf 220 Ω-Widerstände, 11 (5 kürzere und 6 längere) Jumperkabel für jeden Schüler

6./7. Stunde (Computerraum)

Thema: **Töne mit dem Lautsprecher**

M 9 (Tx) **Töne mit dem Lautsprecher abspielen – Teil 1** / Kennenlernen der Funktionsweise eines Lautsprechers und des zugehörigen Befehles

M 10 (Tx) **Töne mit dem Lautsprecher abspielen – Teil 2** / Programmierung von Tönen, einer Melodie und Erweiterung der Fußgängerampel um einen Blindenton

Benötigt:
- ☐ OH-Projektor / Beamer / Visualiser / Whiteboard
- ☐ digitale Vorlage des Word-Dokumentes „Blindenampel“
- ☐ digitale Fassungen von Schaltplänen „05_Lautsprecher“ und „06_Ampelkreuzung_Ton“
- ☐ Arduino-Boxen und USB-Sticks
- ☐ 2 rote, 1 gelbe, 2 grüne LEDs, fünf 220 Ω-Widerstände, 13 (6 kürzere und 7 längere) Jumperkabel, 1 Piezometer für jeden Schüler

8./9./10. Stunde (Computerraum)

Thema: **Die Ampel bei Tag und Nacht / Auf Knopfdruck wird es grün**

M 11 (Tx) **Die Ampel bei Tag und Nacht** / Kennenlernen der Funktionsweise eines lichtabhängigen Widerstandes und der zugehörigen Programmierung

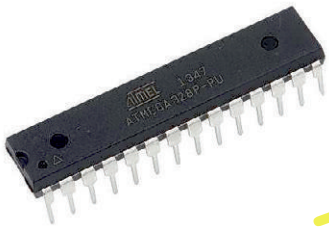
M 12 (Tx) **Auf Knopfdruck wird es grün** / Kennenlernen der Funktionsweise eines Tasters und der gehörigen Programmierung

Benötigt:
- ☐ OH-Projektor / Beamer / Visualiser / Whiteboard
- ☐ Kopie und/oder digitale Vorlagen der Word-Dokumente „AB Serieller Monitor“ und, „AB Spannungsteiler“
- ☐ digitale Fassungen von Schaltplänen „07_Helligkeit_seriell“, „08_Nachtampel“ und „09_Ampelkreuzung_Tag_und_Nacht“
- ☐ digitale Fassungen von Schaltplänen „10_Taster_Unterprogramm“ und „11_Ampelkreuzung_Tag_und_Nacht_Taster“
- ☐ Arduino-Boxen und USB-Sticks
- ☐ 2 rote, 1 gelbe, 2 grüne LEDs, zwei 10–kΩ – und fünf 220–Ω-Widerstände, 20 (10 kürzere und 10 längere) Jumperkabel, 1 Taster, 1 Piezometer für jeden Schüler

M 1 Mikrocontroller im Alltag – Farbfolie

© Martin_Poole / Digital Vision / Getty Images Plus

© grinvalds / iStock / Getty Images Plus



Quelle: de.rs-online.com
© Arduino www.arduino.cc

Mikrocontroller (auch µController, µC) sind Halbleiterchips, die einen Prozessor und sogenannte Peripheriefunktionen enthalten. Ein Mikrocontroller ist ein Ein-Chip-Computersystem.

© Yuri_Arcurs / E+ / Getty Images Plus

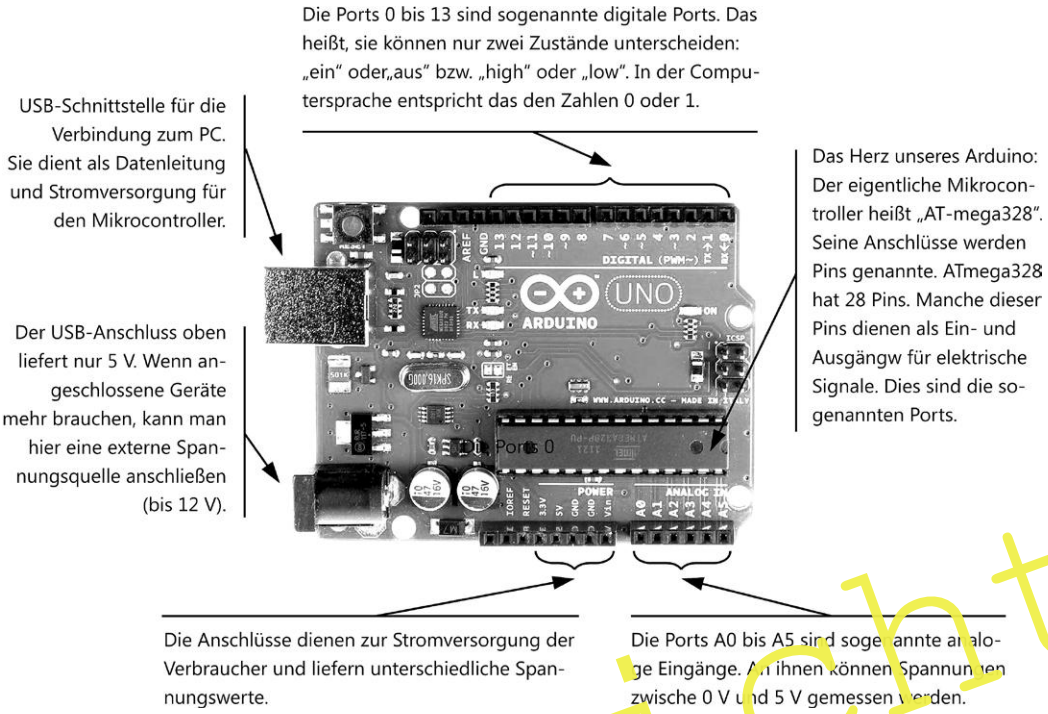
© Guido Mieth / Digital Vision / Getty Images Plus

Aufgaben

1. Beschreibe Funktionen der einzelnen abgebildeten Geräte.
2. Gibt es technische Gemeinsamkeiten?
3. Kennst du weitere Geräte und kannst du deren Funktionsweise beschreiben?

M 3 Aufbau und Funktionsweise des Arduino-Boards

Nun lernst du das Arduino-Uno-Board, mit dem du in Zukunft arbeiten wirst, genauer kennen.



© *Arduino www.arduino.cc*

Als **Hardware** bezeichnet man im Allgemeinen die mechanischen und elektronischen Bauteile eines Mikrocontrollers bzw. PCs – also im Prinzip alles, was man „in die Hand nehmen“ kann. Der Begriff „hardware“ kommt aus dem Englischen und bedeutet übersetzt ursprünglich „Eisenwaren“. Im englischsprachigen Raum besitzt der Begriff diese Bedeutung auch heute noch. Zusätzlich hat er sich aber weltweit als Überbegriff für Bauteile eines Computersystems verbreitet. Auf der Platine des Arduinos sind neben dem eigentlichen Mikrocontroller noch so manche kleine Bauteile wie zum Beispiel Widerstände, Spannungswandler oder Strombegrenzer verbaut und viele verschiedene Anschlüsse angebracht.

Arduino versus Handy: Bei Computern oder Handys wird oft die Taktfrequenz des Prozessors angegeben. Damit ist die Geschwindigkeit gemeint, mit der Daten verarbeitet werden können. Der Atmega328 des Arduinos besitzt eine Taktfrequenz von 16 MHz und hat insgesamt 32 Kilobyte Speicherplatz. Ein Vergleich mit dem iPhone 8 lässt den Arduino dagegen „ziemlich alt“ aussehen: Es hat einen Prozessor mit 1,8 GHz, also 1800 MHz, Taktfrequenz und 3 GB, also 3.000.000 Kilobyte, Arbeitsspeicher. Beachte jedoch: Beim Flug zum Mond half Neil Armstrong ein Computer mit 4 Kilobyte Arbeitsspeicher und einem 1-MHz-Prozessor.

Aufgaben

1. Beschreibe den Aufbau des Arduino-Boards.
2. Problematisiere die Leistungsfähigkeit eines Mikrocontrollers.

M 5 Die Arduino-Softwareumgebung – Teil 2

Bevor du eine Schaltung nachbaust und vom Arduino steuern lässt, musst du allerdings sicherstellen, dass dein Mikrocontroller richtig mit dem PC kommuniziert. Dazu übertragen wir ein „leeres" Programm vom PC auf den Arduino. Alle Programme haben dabei den gleichen Aufbau.

```
void setup()
{
  // Grundeinstellungen
}

void loop()
{
  // Anweisungen
}
```

Dieser Programmteil wird einmal durchlaufen, Im setup()-Teil werden die Ein- und Ausgänge (und andere Grundeinstellungen) festgelegt.

Dieser Programmteil wird unendlich oft wiederholt (loop – engl.: Schleife). Hier befindet sich das eigentliche Programm.

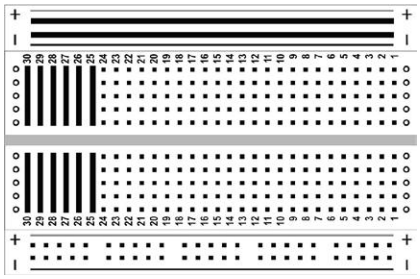
Doch was bedeutet eigentlich das Wort „void"? Void kann man mit Leerstelle oder Loch übersetzen. Das bedeutet in diesem Fall jedoch nicht 0, sondern einfach nichts. Es sagt aus, dass das Programm an dieser Stelle keinen Wert an einen Aufrufer zurückliefert.
Innerhalb der geschweiften Klammern stehen in einem echten, also nicht leeren, Programm später Teile des Programmes. Nützlich gerade bei längeren Programmen können Erklärungen oder Hinweise zu einzelnen Programmzeilen sein: Diese können an den Stellen „// Grundeinstellungen;" und „// Anweisungen;" zwischen „//" und „;" notiert werden. Anschaulich erkennt man Kommentare daran, dass sie vom Programm grau geschrieben werden. Diese Teile werden nicht kompiliert.
Schreibe und kompiliere zunächst das leere Programm von oben und übertrage es anschließend auf den Arduino, indem du auf den Button klickst. Nach erfolgreicher Übertragung meldet das Ereignis-Fenster:

```
Upload abgeschlossen.

Binäre Sketchgröße: 466 Bytes (von einem Maximum von 32.256 Bytes)

8                                                    Arduino Uno on COM3
```

Aufgabe

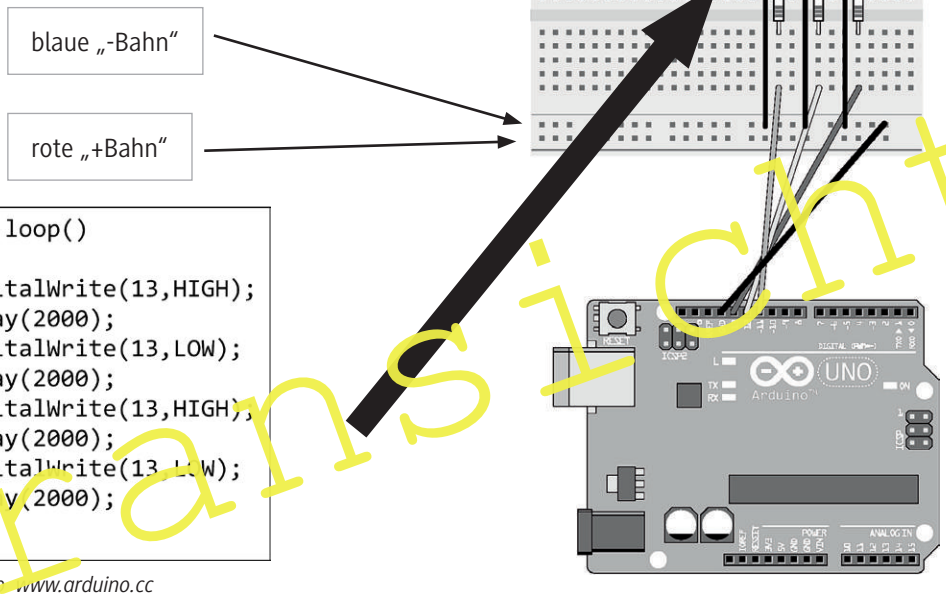
1. Finde heraus, was die Zahl „8" – es kann auch eine andere Zahl dort stehen – unten links im Editor bedeutet, nachdem du das leere Programm hochgeladen bzw. kompiliert hast.

M 7 Verwendung eines Steckbrettes und Autofahrerampel



Für mehrere LEDs und weitere Komponenten reicht der Platz auf dem Arduino-Board nicht mehr aus. Deshalb verwendet man in diesem Fall sog. **Steckplatinen**, auch Breadboards genannt. Auf einer solchen Steckplatine steht wesentlich mehr Platz zur Verfügung, um elektrische Schaltungen aufzubauen. Die Bauteile können dabei einfach – wie der Name schon sagt – in die Steckplatine hineingesteckt werden.

Hinweis: Das längere Beinchen wird jeweils mit dem Widerstand verbunden.




```
void loop()
{
 digitalWrite(13,HIGH);
 delay(2000);
 digitalWrite(13,LOW);
 delay(2000);
 digitalWrite(13,HIGH);
 delay(2000);
 digitalWrite(13,LOW);
 delay(2000);
}
```

© Arduino, www.arduino.cc

Das Besondere an der Steckplatine ist, dass die Steckplätze auf eine bestimme Weise untereinander leitend miteinander verbunden sind. Die unterschiedlichen Verbindungen sind in der obigen Abbildung gut nachzuvollziehen. Dabei ist die Kerbe in der Mitte stets zu überwinden, um einen durchgehenden Stromfluss zu erzielen. Praktisch sind auch die sich auf beiden Seiten befindlichen äußeren roten (+-) und blauen (–) Leiterbahnen.

Aufgaben

1. Informiere dich anhand des Arbeitsblattes „AB Widerstände“ über Widerstände.
2. Baue, wie oben dargestellt, auf der Steckplatine eine Ampelschaltung auf.
3. Erweitere dein Programm der blinkenden LED aus **M 6** zu einer Autofahrerampel. Beachte dabei den korrekten Ablauf der einzelnen Ampelfarben und gib sinnvolle Zeiten ein.
4. Speichere den zugehörigen Sketch unter „02_Autofahrer-Ampel“.
5. Der obige Programmteil lässt sich wesentlich einfacher schreiben. Weißt du wie?
6. Lege in deinem Heft(er) ein Glossar an, in dem du alle Befehle, die du im Laufe der Zeit kennenlernst, aufschreibst. Dieses Blatt darfst du dann ggf. in der Klassenarbeit verwenden.

M 9 Töne mit dem Lautsprecher abspielen – Teil 1

Mit dem Arduino kann man auch einzelne Töne, ja sogar ganze Melodien an einem kleinen Lautsprecher abspielen. Der Lausprecher wird dabei mit einem digitalen Port und GND verbunden. Der zugehörige Befehl beim Arduino heißt tone:

tone(Port, Frequenz, Dauer des Tones in ms);

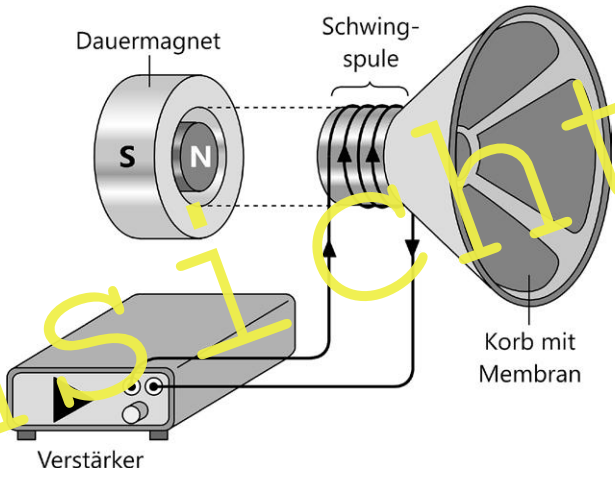
Der Port, an den der Lautsprecher angeschlossen wird, muss nicht extra als OUTPUT definiert werden. Auch dies erledigt der tone-Befehl mit.

```
void loop()
{
 tone(11,440,1000);
 delay(2000);
}
```

Port 11 wird 440 Mal pro Sekunde an- und ausgeschaltet und das ganze für 1000 Millisekunden (1 Sekunde).

Wie funktioniert ein Lautsprecher?

Lautsprecher bestehen aus einem Permanentmagnet und einer Spule, die als Elektromagnet fungiert. Sobald durch die Spule ein Strom fließt, entsteht ein Magnetfeld. Dreht man die Stromflussrichtung um, kehrt sich auch die Richtung des Magnetfeldes um. Je nach Richtung ihres Magnetfeldes wird die Spule vom Permanentmagnet entweder angezogen oder abgestoßen. Ändert man die Stromflussrichtung sehr schnell, schwingt die Spule ständig vor und zurück.

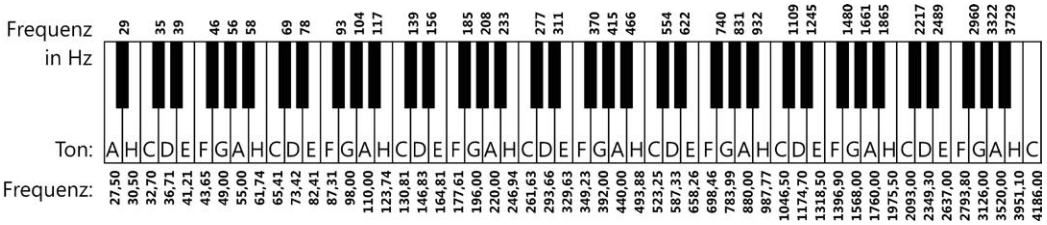


Da die Spule mit einer dehnbaren Membran verbunden ist, wird die Luft vor dem Lautsprecher ebenfalls in Schwingung versetzt. Bei Schwingfrequenzen zwischen etwa 20 Hertz und 20 000 Hertz nehmen wir die Luftschwingungen mithilfe unserer Ohren als Ton wahr.

Da der Arduino nur Gleichstrom ausgeben kann, kann der Strom durch den Lautsprecher immer nur in eine Richtung fließen. Um trotzdem einen Ton zu erhalten, wird der Lautsprecher immer wieder ein- und ausgeschaltet. Weil die gedehnte Membran dann nicht mehr abgestoßen wird, bewegt sie sich von allein in ihre Ausgangsposition zurück. Wird der Stromfluss nun schnell hintereinander an- und abgeschaltet, bewegt sich die Membran ebenfalls schnell vor und zurück. Allerdings nicht ganz so stark, wie wenn man Wechselstrom verwenden würde. Der Ton ist deshalb nicht ganz so laut.

Töne mit dem Lautsprecher abspielen – Teil 2

M 10



Tipp:

Wie du vielleicht schon bemerkt hast, wartet das Programm nach einem *tone*-Befehl nicht, bis der gesamte Ton abgespielt ist, sondern bearbeitet sofort die nächste Programmzeile. Deshalb ist immer ein *delay*-Befehl notwendig, um sicherzustellen, dass ein Ton auch wirklich in der ganzen Länge abgespielt wird. Es gibt auch einen Befehl, um die Tonausgabe abzuschalten: noTone(Port);

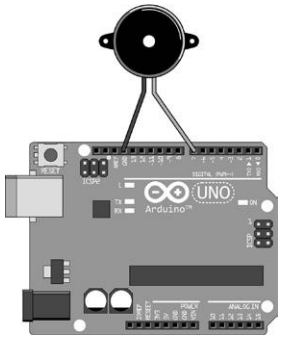
Lebewesen	Hörumfang in Hz
Mensch	16–20 000
Hund	15–50 000
Katze	60–65 000
Delfin	150–150 000
Fledermaus	1000–120 000

```
void setup() {
 pinMode (7,OUTPUT);
}

void loop() {
tone(7,220,1000);
delay(1000);
tone(7,440,1000);
delay(1000);
tone(7,660,1000);
delay(1000);
}
```

Aufgaben

1. Schreibe das Programm ab und speichere den zugehörigen Sketch unter „05_Lautsprecher“.
2. Was wirst du hören, wenn du einen Lautsprecher an Port 11 anschließt und das Programm von **M 9** auf den Arduino überträgst? Überlege zuerst und überprüfe anschließend deine Vermutung.
3. Bestimme den Hörbereich deines Ohres und vergleiche ihn mit dem deines Nachbarn und von einzelnen Lebewesen in der Tabelle.
4. Was ist bei nebenstehendem Programm zu hören?
5. Programmiere eine Melodie deiner Wahl.
6. Ergänze deine Ampelkreuzung um die Ausgabe eines Tones bei der Fußgängerampel, wenn diese Grün anzeigt.
7. Speichere den zugehörigen Sketch unter „06_Ampelkreuzung_Ton“.
8. Ergänze dein Glossar um die *tone*- und *notone*-Befehle.

längeres Beinchen an PIN 7 und kürzeres Beinchen an GND



Glossar

Arduino-Befehl	Beschreibung
Void setup { }	Hier werden Grundeinstellungen (z. B. ob ein Kanal ein In- oder Output ist) vorgenommen. Diese Methode wird nur einmal beim Programmstart ausgeführt.
Void loop { }	Die Methode wird im Gegensatz zu „void setup" ständig wiederholt. Hier wird der eigentliche Programmablauf hineingeschrieben.
pinMode(3,OUTPUT);	Dieser Befehl setzt den digitalen PIN 3 als Ausgang fest.
pinMode(3,INPUT);	Dieser Befehl setzt den digitalen PIN 3 als Eingang fest.
digitalWrite(3 HIGH);	Dieser Befehl „schaltet" den digitalen PIN 3 an.
digitalWrite(3,LOW);	Dieser Befehl „schaltet" den digitalen PIN 3 aus.
delay(2000);	Dieser Befehl erreicht eine Pause in der Programmausführung in Höhe von 2000 Millisekunden, also 2 Sekunden.
int waiting = 5000; delay(waiting);	Diese Befehle definieren eine Variable namens *waiting* und eine Pause in Höhe von 5 Sekunden. Nur bei „int" muss man diese Zahl ändern.
tone(11, 440, 1000);	Port 11 wird 440 Mal pro Sekunde für 1000 Millisekunden (1 Sekunde) an- und ausgeschaltet.
notone(11);	Mit diesem Befehl schaltet man die Tonausgabe am Port 11 ab.
Serial.begin(9600);	Legt die Datenrate in Bit pro Sekunde (Baud) für die serielle Datenübertragung fest.
analogRead(LDR);	Dieser Befehl liest den Wert vom angegebenen analogen Pin, in diesem Fall dem LDR, ein.
Serial.println(Helligkeit);	Dieser Befehl druckt Daten an den seriellen Anschluss als von Menschen lesbarer ASCII-Text, gefolgt von einem Zeilenwechsel.
digitalRead(Taster);	Dieser Befehl liest den Wert vom angegebenen digitalen Pin, in diesem Fall des Tasters, ein.
if (tasterstatus == HIGH) *{* *…* *}* *else* *{* *…* *}*	Erster Teil: Bedingung (wenn / if): Zu Beginn wird geprüft, ob die Bedingung erfüllt ist. Zweiter Teil: Bedingung erfüllt (dann / then): Ist die Bedingung erfüllt, dann führt der Arduino das Programm in den ersten geschwungenen Klammern aus. Dritter Teil: Bedingung nicht erfüllt (ansonsten / else): Ist die Bedingung nicht erfüllt, dann führt der Arduino das Programm in den zweiten geschwungenen Klammern aus.

Hinweise (M 1 und M 2; 1./2. Stunde)

Ziel der ersten beiden Stunde ist es, die Schüler auf den Einsatz von Mikrocontrollern im Allgemeinen neugierig zu machen und in technische Aspekte einzuführen.
Im **Einstieg** zu der Stunde machen sich die Schüler darüber Gedanken, welche einzelnen **Funktionen ausgewählte technische Geräte** haben und was ihnen dabei gemeinsam zu sein scheint. Dazu legen Sie als Lehrkraft zunächst die Folie (**M 1)** auf und lassen in einem **Unterrichtsgespräch** die zugehörigen Aufgaben bearbeiten und verschriftlichen.
In der sich anschließenden **Erarbeitungsphase** geht es in einem **Unterrichtsgespräch** unter Einsatz des Arbeitsblattes **M 2** nun darum, **die technischen Grundlagen der Funktionsweise eines Mikrocontrollers** zunächst am Beispiel eines Autoschlüssels zu besprechen und sie in einem zweiten Schritt mit dem **Eingabe-Verarbeitung-Ausgabe-Prinzip** zu verallgemeinern. Sämtliche Ergebnisse werden anschließend gesammelt und stichwortartig festgehalten.
Den Abschluss der Unterrichtsstunde bildet **Organisatorisches**, um in der nächsten Stunde gleich mit dem Mikrocontroller Arduino starten zu können. Es empfiehlt sich, dass für die kommenden Unterrichtsstunden jeder Schüler seine einzelne Starter-Box hat, in der ein Zettel mit seinem Namen liegen sollte. Jeder Schüler sollte entweder einen USB-Stick in seinem Mäppchen regelmäßig dabeihaben oder einen solchen in der Box ablegen. Auf diesem können die erstellten Programme gespeichert und bearbeitet werden. Zur schnellen Wiedererkennung ist es hilfreich, einen Zettel mit Namen des Schülers und Klasse sichtbar in die Box zu legen, wenn diese Boxen in einem Raum separat gelagert werden. Das Word-Dokument „Arduino – Organisation" bietet eine mögliche Vorlage. Alternativ kann jeder Schüler natürlich seine Box in den entsprechenden Stunden auch immer wie einen Füller oder ein Geodreieck mitbringen.

Erwartungshorizont (M 1)

Aufgabe 1: Waschmaschine: Nach Einfüllen des Autos wird – abhängig von der Art der Wäsche – ein bestimmtes Programm durch Tastendruck ausgewählt. Ein Mikrocontroller verarbeitet diesen Befehl. In der Folge läuft dieses Programm mit einer entsprechenden Dauer, mit einer bestimmten Temperatur und ggf. mit weiteren Extras ab.
Kaffeemaschine: Es sind Wasser und Kaffeebohnen vorhanden bzw. nachzulegen. Man stellt seine Kaffeetasse drunter und wählt eine bestimmte Kaffeeart. Die Kaffeemaschine beginnt zu arbeiten und füllt zum Schluss die Kaffeetasse mit dem gewünschten Produkt.
Mikrocontroller: Auf den Chip eines Mikrocontrollers wird ein zuvor geschriebenes Programm geladen. Die Recheneinheit (CPU, Prozessor, Controller) greift darauf zu, speichert es zwischen und berechnet daraus die Datenausgabe. Anschließend werden bestimmte Befehle ausgeführt.
Tablet / Handy / Notebook: Auf diesen Geräten sind verschiedene Programme bzw. Apps installiert, die ausgewählt werden. In einem Programm bzw. einer App. selbst werden durch Finger- bzw. Tastendruck bestimmte Befehle angefordert, verarbeitet und anschließend umgesetzt.
Smartwatch: Auf solch einer Uhr sind verschiedene Funktionen vorhanden, die über eine Menüsteuerung oder per Tastendruck ausgewählt bzw. angefordert werden. Nach einer Verarbeitung wird der gewünschte Befehl ausgeführt.
Aufgabe 2: In allen Geräten müssen Steuerungen/Minicomputer/Mikrocontroller eingebaut sein.
Aufgabe 3: Individuelle Schülerantworten wie z. B.: eine sich selbsttätig öffnende Schiebetür: Eine Person nähert sich einer sich selbsttätig öffnenden Schiebetür. Tritt ein bestimmter Abstand ein, so öffnet sich die Tür. Ist diese Person durchgegangen bzw. keine weitere Person unterhalb eines bestimmten Abstandes, schließt sie sich wieder.