Inhaltsverzeichnis

1. Vorwort und didaktische Überlegungen

Aller Anfang ist schwer - jedoch nicht mit dieser Anleitung für Arduino. Sie soll als Grundlage zum Erlernen der Arduinoplattform dienen und Anfängern einen didaktisch fundierten, einfachen, interessanten und eng geleiteten Einstieg in die Arduinothematik geben. Der Kompetenzerwerb ist durch dieses Heft in diversen Bereichen des Mikrocontrollings so breit gefächert, dass der Leser befähigt wird, sich selbstständig in fortgeschrittene Themengebiete einzuarbeiten. Dazu gehört etwa das Erlernen weiterer Programmiermöglichkeiten oder die Verwendung weiterer Module.
Dieses Werk ist in einem Zeitraum von fast zehn Jahren parallel zu einer Technikfortbildung für Lehrkräfte und dem Unterrichtseinsatz mit Schülern verschiedener Altersklassen entstanden. Mit den Jahren wurde es mehrfach evaluiert, aktualisiert und erweitert. Entstanden ist ein Arbeitsheft, das unterrichts- oder kursbegleitend eingesetzt werden kann.

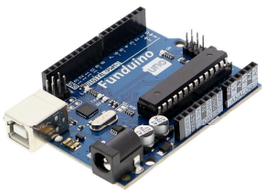
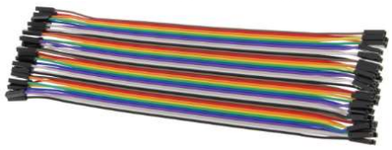
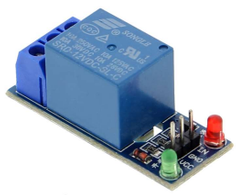
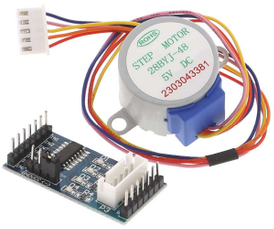
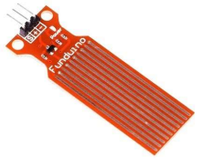
Einsatz im Unterricht

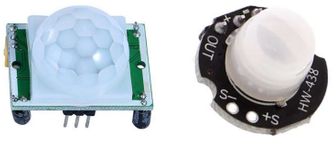
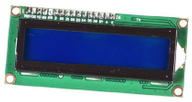
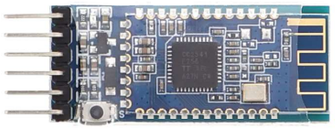
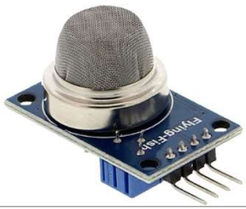
Die **didaktische Konzeption** dieser Anleitung basiert darauf, dass die Lernenden weitgehend **selbstständig** mit der Anleitung und den entsprechenden Elektronikbauteilen **arbeiten**. Die Kursleitung gibt für jede Anleitung die notwendigen Arbeitsmaterialien aus und steht den Lernenden danach ausschließlich beratend zur Seite, beispielsweise um einzelne Teilnehmer oder Kleingruppen zu **fördern**. Gerade in der Anfangsphase kann es passieren, dass die Lernenden Fehler in der Elektronik, der Programmiersyntax oder der Programmlogik nicht selber finden oder erkennen können. Ein Gefühl für **Problemlösungen** dieser Art wird sich bei den Lernenden jedoch rasch entwickeln. Wichtig ist in dieser Hinsicht, dass man vor den praktischen Übungen die **theoretische Einführung** liest, um bei den späteren Praxisaufgaben nicht an fehlendem Grundwissen zu scheitern.

Durch die **freie Arbeitsweise** erreicht man eine optimale und **effektive Lernzeit**, da **jeder in seinem eigenen Tempo** arbeiten und lernen kann. Weiterhin entwickelt sich mit der Erarbeitung der einzelnen Anleitungen schnell eine projektähnliche Gruppendynamik, da zwar jeder für sich arbeitet, aber dennoch alle das gleiche Ziel verfolgen. Das gegenseitige Erklären und Helfen **unterstützt den Lernprozess** aller Beteiligten und **fördert die Teamfähigkeit**.
In vielen Arbeitsphasen ergeben sich **Differenzierungsmöglichkeiten**, indem man Schüler dazu anleitet, die genannten Beispiele zu erweitern oder die eigene Idee **kreativ** in die Lösung der Aufgabe einfließen zu lassen.

1.1 Vorbereitung und Materialbeschaffung

Für die Übungen in diesem Heft werden folgende Materialien benötigt

01. Mikrocontroller (UNO MEGA /NANO) 	02. Breadboard 	03. Breadboardkabel Stecker / Stecker 
04. Breadboardkabel Fassung / Fassung 	05. Breadboardkabel Stecker / Fassung 	06. Servo MG90S oder SG90 
07. je 20 LEDS (grün, rot, gelb, blau, weiß) 	08. RGB-LED 	09. Infrarotsender und -empfänger 
10. Vierstellige 7-Segment Anzeige 	11. Relaiskarte 	12. Infrarotfernbedienung 
13. Schrittmotor mit Treiberplatine 	14. Tropfensensor 	15. Feuchtigkeitssensor 
16. RFID-KIT 	17. je 20 Widerstände (100,200,300,1K,10K Ohm) 	18. Ultraschallsensor 

19. Bluetooth Modul HC-05 	20. Neigungssensor 	21. Bewegungsmelder HC-SR501 oder SR602 
22. Temperatursensor TMP36 	23. Fotowiderstand 	24. Drehpotentiometer 
25. Taster (groß) 	26. Taster (klein) 	27. Piezospeaker 
28. I²C LCD 	29. Diode 	30. Lautsprecher 
31. Ampelmodul 	32. WS2812 LED Modul mit 8 LEDs 	33. Bluetoothmodul (Alternative zu HC-05) 
34. NTC-Temperatursensor 	35. 5V Lüfter oder DC-Elektromotor 	36. TIP120 Transistor 
37. Lichtschranke 	38. MQ135 Gassensor 	

1.2 Was ist Arduino?

Arduino ist eine Open-Source-Elektronik-Prototyping-Plattform für flexible, einfach zu bedienende Hardware und Software im Bereich Mikrocontrolling. Es ist geeignet, um in kurzer Zeit spannende und spektakuläre Projekte zu verwirklichen. Viele davon lassen sich unter dem Begriff „Arduino“ beispielsweise bei YouTube finden. Es wird vor allem von Künstlern, Designern, Tüftlern und Bastlern verwendet, um kreative Ideen zu verwirklichen. Aber auch in Schulen, Hochschulen und Universitäten wird die Arduino Entwicklungsumgebung zunehmend eingesetzt, um Lernenden einen kreativen und spannenden, aber vor allem auch einfachen Zugang zum Thema „Mikrocontrolling“ zu ermöglichen. Auch Themengebiete wie „Automatisierungstechnik“, „Robotik“ etc. lassen sich mit der Arduino Entwicklungsumgebung erarbeiten.

1.3 Entwicklung – Arduino, Funduino, Genuino

Die Geschichte von Arduino begann im Jahr 2005, als die beiden „Tüftler“ Massimo Banzi und David Cuartielles ihr erstes Mikrocontroller Board entwickelten und der Programmierer David Mellis die zugehörige Syntax schuf, welche auf den Programmiersprachen C++, C und Assembler beruht. Das Projekt untersteht einer Creative-Commons-Lizenz. Somit wurde Arduino weitgehend zu einer Open-Source Plattform, wodurch die Verbreitung und Entwicklung stark forciert wurde.
Im Laufe der Jahre gerieten die Gründer der Arduinoplattform (Arduino LLC) und die Produzenten der offiziellen Arduinoboards (Arduino S.r.l) in Streit, da beide Gruppen den Markennamen "Arduino" für sich beanspruchten. Zum damaligen Zeitpunkt konnte nicht eindeutig geklärt werden, welche Partei der rechtmäßige Inhaber der Marke sei - ein Rechtsstreit mit weitreichenden Folgen entstand. Die Onlineplattform spaltete sich in mehrere Internetpräsenzen auf (www.arduino.org, www.arduino.cc), beide Streitparteien vertrieben ihre eigenen Mikrocontroller, jedoch beide unter dem gleichen Markennamen „Arduino“. Im Jahr 2015 wurde vom Gründungsmitglied M.Banzi die Marke „Genuino“ vorgestellt. Genuino war fortan die Bezeichnung für Mikrocontroller Boards, die außerhalb der Vereinigten Staaten von Amerika verkauft werden sollten. Erst auf der World Maker Faire 2016 verkündeten Arduino LLC und Arduino S.r.l. den Zusammenschluss der streitenden Parteien unter einer neu gegründeten Arduino Holding. Während der Jahre entstanden aufgrund der Unsicherheit bzgl. der Namensrechte viele neue Marken und Namen anderer Hersteller von arduinokompatiblen Mikrocontrollerboards. Auf Basis der quelloffenen „Open Source“ Grundlage sind die Boards zwar in den meisten Fällen technisch baugleich, dürfen jedoch nicht mit dem Namen „Arduino“ versehen werden. Diese Boards werden dann als „Arduino-Clon“ oder „Arduino-kompatibel“ bezeichnet. Als Beispiel ist in dem namensrechtlich-problematischen Zeitraum die deutsche Firma Funduino GmbH entstanden. Die Firma hat bereits in der Entstehungsphase im Jahr 2010 umfangreiche Unterrichtsmaterialien und Lernsets für den schulischen Bereich entwickelt. Da es nicht möglich war, eine konstruktive Zusammenarbeit mit Arduino herzustellen, produziert die Firma seither eigene arduinokompatible Mikrocontrollerboards.

1.3 Entwicklung – Arduino, Funduino, Genuino

Heute stellt der Begriff Arduino die Definition für flexible, einfach zu bedienende Hard- und Software im Bereich Mikrocontrolling dar. Arduino ist für die Verwirklichung von spektakulären Projekten prädestiniert. Aufgrund der geringen Anschaffungskosten, der praxisnahen Entwicklungsumgebung und dem daraus resultierenden didaktischen Mehrwert, sowie den nahezu unendlichen Kombinationsmöglichkeiten der unzähligen Sensoren und Aktoren findet der Themenbereich Mikrocontrolling mehr und mehr Einzug in den globalen Bildungsbereich.

2. Hardware und Software

Der Begriff Arduino wird im allgemeinen Wortgebrauch gleichermaßen für die verschiedenen „Arduinoboards“ (also die Hardware) als auch für die Programmierumgebung (Software) verwendet.

2.1 Hardware

Neben dem Mikrocontroller, Sensoren und Aktoren benötigt man als Basis für schnelle und flexible Versuchsaufbauten Steckkabel in Verbindung mit einem Breadboard. Dadurch erspart man sich zeitraubende Lötarbeiten. Des Weiteren eignen sich Leuchtdioden sehr gut, um die Signalausgabe des Boards zu überprüfen.

2.1.1 Der Mikrocontroller

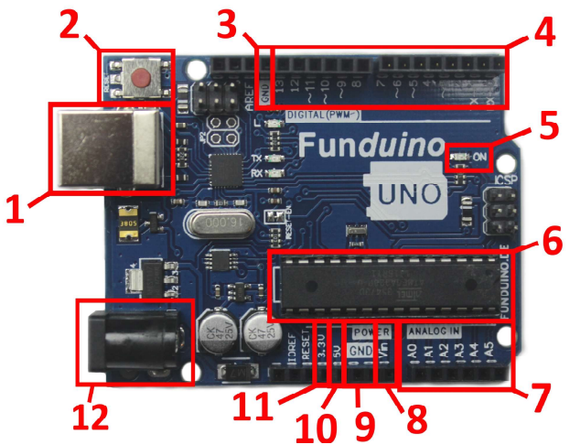
Der „Arduino“ ist ein sogenanntes Mikrocontroller-Board (im weiteren Verlauf auch „Board“ genannt). Es handelt sich dabei um eine Leiterplatte (PCB – *„Printed circuit board“*) mit einer Vielzahl von Elektronikbauteilen rund um den eigentlichen Mikrocontroller-Chip. Am Rand des Boards befinden sich viele Steckplätze (Pins genannt), an denen man die unterschiedlichsten Module wie Sensoren und Aktoren anschließen kann. Dazu gehören: Schalter, LEDs, Ultraschallsensoren, Temperatursensoren, Drehregler, Displays, Motoren, Servos usw.

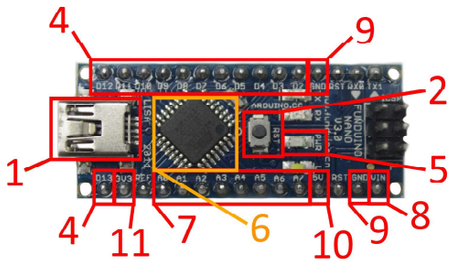
Es gibt sehr viele verschiedene Versionen von Mikrocontrollerboards, die mit der Arduino-Software verwendet werden können. Dazu gehören sowohl viele verschiedene große und kleine Boards mit der offiziellen „Arduino“ Bezeichnung als auch eine Vielzahl von häufig günstigeren Arduino-kompatiblen Boards. Die Boards unterscheiden sich nur in kleinen Details, wie die Menge der digitalen Pins oder der Speicherkapazität.

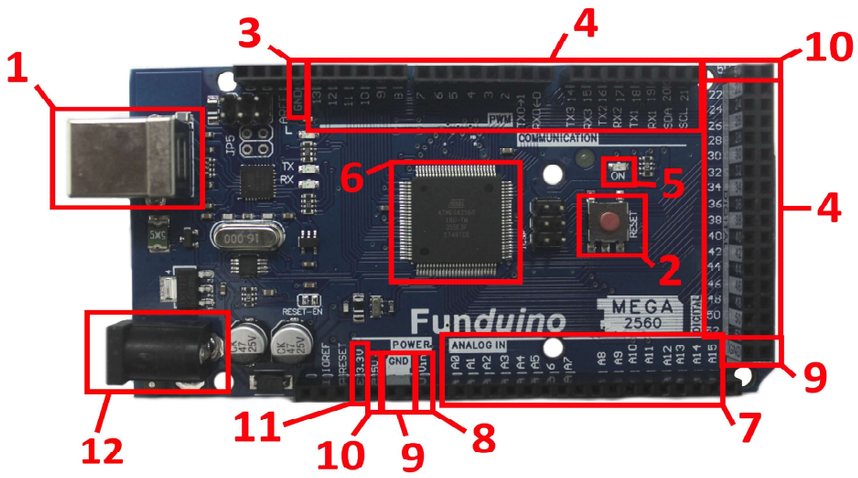
Diese Anleitung wurde mit einem Arduino-kompatiblem UNO-Board der Marke Funduino erstellt. Dennoch kann jeder beliebige Arduino-kompatibler Controller mit dieser Anleitung verwendet werden.

2.1.1 Der Mikrocontroller

Die folgenden drei Boards sind die bekanntesten Mikrocontrollerboards, die mit der Arduinosoftware verwendet werden: UNO, NANO und MEGA



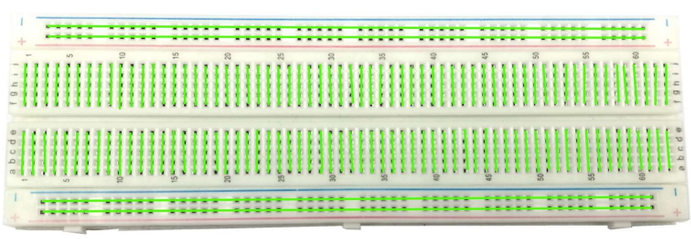




Legende:

1. USB-Anschluss	5. Power-ON Leuchte	9. GND (*Ground* bzw. „-“)
2. Reset-Knopf	6. Mikrocontroller	10. 5V Ausgang vom Spannungsregler
3. GND (*Ground* bzw. „-“)	7. Analoge Eingänge	11. 3,3V Ausgang vom Spannungsregler
4. Digitale Ein- und Ausgänge	8. Ext. Spannungsversorgung per Pin (*Vin*)	12. Externe Spannungsversorgung (*7-12V*)

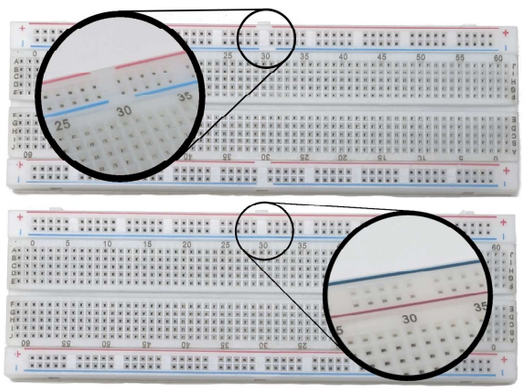
2.1.2 Zubehör - Das Breadboard



Ein Breadboard oder auch „Steckbrett“ ist ein gutes Hilfsmittel, um Schaltungen aufzubauen, ohne löten zu müssen. In einem Breadboard sind immer mehrere

Kontakte miteinander verbunden. Daher können an diesen Stellen viele Kontakte hergestellt werden, ohne dass sie verlötet oder verschraubt werden müssen.

Es gibt sehr viele verschiedene, große, kleine, farbige, transparente Breadboardversionen. Auch die Verdrahtung im Inneren des Breadboards kann unterschiedlich sein. In der Regel sind die Kontakte so miteinander verbunden, wie es auf dem Bild dargestellt ist. Die äußeren Linien sind durchgehend miteinander verbunden und die kleinen Linien im inneren Bereich sind jeweils mit fünf Steckplätzen untereinander verbunden. Die kleinen inneren Segmente eignen sich für detaillierte Aufbauten, während die äußeren Linien in der Regel für die Verteilung der Spannungsversorgung genutzt werden.



Es gibt auch Breadboards, bei denen die äußeren Kontakte in der Mitte noch einmal oder sogar mehrfach unterteilt sind. Daher sollte man vor dem Beginn mit der Arbeit prüfen, wie das verfügbare Breadboard kontaktiert ist. Häufig sind die äußeren Kontakte mit roten und blauen Linien versehen, an denen man erkennen kann, ob die Kontakte in einer durchgehenden Linie miteinander verbunden sind oder nicht.

2.1.3 Zubehör - Leuchtdioden

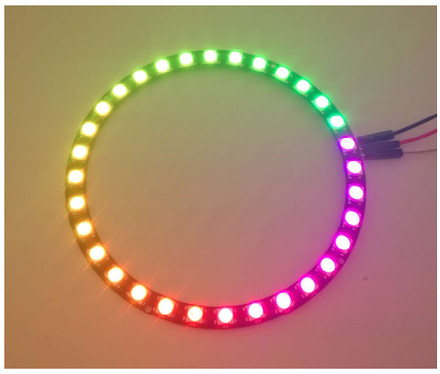
Mit LEDs kann man sehr schnell die Ergebnisse eines Projekts testen. Daher sind sie für nahezu alle Arduinoprojekte nützlich. **Die wichtigsten Informationen über Leuchtdioden:**

- Der Strom kann nur in einer Richtung durch die LED fließen. Daher muss sie richtig angeschlossen werden. Eine LED hat einen längeren und einen kürzeren Kontakt. Der längere Kontakt ist „+“ und der kürzere ist „-“.
- Eine LED ist für eine bestimmte Stromstärke ausgelegt. Wird diese Stromstärke unterschritten, leuchtet die LED weniger hell oder sie bleibt aus. Wird die Stromstärke jedoch überschritten, brennt die LED zügig durch und wird an den Kontakten heiß (Achtung, Verbrennungsgefahr!). Eine zu hohe Stromstärke kann auftreten, wenn die für die LED maximale Spannung überschritten wird. Schließt man bspw. eine LED direkt an den 5V Ausgang an, ist sie umgehend defekt. Daher nutzt man bei der Verwendung von LEDs an Arduinoboards immer einen Vorwiderstand.
- Typische Spannungswerte nach LED Farben: Blau: 3,1V, Weiß: 3,3V, Grün: 3,7V, Gelb: 2,2V, Rot: 2,1V. Die exakte vorgesehene Spannung kann man im Datenblatt zur jeweiligen LED nachlesen.

• Unverbindliche Empfehlung für Widerstände bei Verwendung der folgenden LED-Farben an den 5V Pins des Mikrocontrollers:

LED-Farbe:	Weiß	Rot	Gelb	Grün	Blau	Infrarot-LED
Widerstand:	100 Ohm	200 Ohm	200 Ohm	100 Ohm	100 Ohm	100 Ohm

Intelligente Leuchtdioden - NeoPixel



Neben den Standard-LEDs gibt es noch die "Luxusvariante" in Form von WS2811-, WS2812- und WS2812B-LEDs. Häufig werden diese LEDs auch als NeoPixel bezeichnet. Es handelt sich dabei um farbige (RGB) LEDs mit einem integrierten Chip, der dazu dient, die RGB-LEDs zu kontrollieren. Der Vorteil liegt darin, dass die LED über nur drei statt vier Kontakte angesteuert wird und es können sehr viele LEDs hintereinandergeschaltet werden, ohne dass weitere separate Kabel benötigt werden. Vorgefertigte WS2812-LED-Kombinationen findet man dann häufig in Form von LED-Ringen oder farbige LED-Lichterketten. Auf dem Bild ist ein WS2812-Ring zu sehen, bei dem die LEDs mit nur drei Kabeln in Regenbogenfarben angesteuert werden.

2.1.4 Zubehör – Widerstände

Ein elektrischer Widerstand ist ein passives elektrisches Bauelement und wird insbesondere im Bereich des Mikrocontrollings häufig in Stromkreisen integriert. Widerstände werden verwendet, um den elektrischen Strom zu begrenzen. Dadurch können Bauteile im Stromkreis geschützt werden. Typische Leuchtdioden haben z.B. eine vorgesehene Stromstärke von 20mA. Fließt in einem Stromkreis jedoch viel mehr Strom, so kann die Leuchtdiode kaputtgehen.

Widerstände können den elektrischen Strom in einer Schaltung auch aufteilen. Dadurch wird es z.B. möglich, Sensorwerte von Sensoren auszulesen, die in Abhängigkeit des zu messenden Wertes ihre elektrische Leitfähigkeit verändern.

Es gibt sehr viele verschiedene Widerstände, da je nach Anwendungszweck ein individueller Widerstandswert benötigt wird. Die Auswahl des passenden Widerstands ist nicht leicht. Daher wird in den Anleitungen der notwendige Widerstandswert vorgegeben.

2.1.5 Zubehör - Sensoren und Aktoren

Die Möglichkeiten, Sensoren oder Aktoren am Arduino-Mikrocontroller zu verwenden, sind nahezu unerschöpflich. Das liegt daran, dass die Arduino Entwicklungsumgebung kein in sich geschlossenes System ist, bei dem nur vorgefertigte Module verwendet werden können. Ganz im Gegenteil lassen sich unzählige elektronische Module aus dem Elektronikhandel in irgendeiner Weise mit Arduino verwenden, beispielsweise durch das Auslesen digitaler oder analoger Werte. An dieser Stelle folgt beispielhaft eine (winzige) Auswahl an typischen Modulen, die in Kombination mit Arduino Mikrocontrollern verwendet werden können.

(S) = Sensor, (A) = Aktor, (K) = Kombiniertes Modul mit diversen Funktionen

1. Feuchtigkeit (S)
2. Strecke (Ultraschall) (S)
3. Bewegungen (S)
4. Luftfeuchtigkeit (S)
5. Luftdruck (S)
6. Stromstärke (S)
7. GPS-Empfänger (S)
8. Drehreglerposition (S)
9. Temperatur (S)
10. Tastendruck (S)
11. Tropfen (S)
12. Schieberegler (S)
13. Neigung (S)
14. Schallpegel (S)
15. Neigung / Beschleunigung (S)
16. Elektrische Spannung (S)
17. Farberkennung (S)
18. Infrarotsignale (S)
19. Leuchtstärke (S)
20. GSM-Mobilfunk (K)
21. pH-Wert (S)
22. Magnetfelder (S)
23. Flammen / Feuer (S)
24. Herzfrequenz (S)
25. RFID - Funkcode (K)
26. Wasserstand / Pegel (S)
27. Gas (CO, Alkohol etc.) (S)
28. Fingerabdruck (K)
29. Stöße/Erschütterung (S)
30. Mikroschalter (S)
31. Lichtschranken (S)
32. Vibrationen (S)
33. UV-Licht (S)
34. Feinstaub (S)
35. Abstand (Infrarot) (S)

2.2 Software

Die Software, mit welcher der Mikrocontroller programmiert wird, ist Open-Source-Software und kann auf www.arduino.cc kostenlos heruntergeladen werden. In dieser „Arduino-Software“ schreibt man dann kleine Programme, die der Mikrocontroller später ausführen soll. Diese kleinen Programme werden „Sketch“ genannt. Per USB-Kabel werden die fertigen Sketche anschließend auf den Mikrocontroller übertragen. Wie das funktioniert, wird im Themengebiet „Programmieren“ behandelt.

2.2.1 Installation

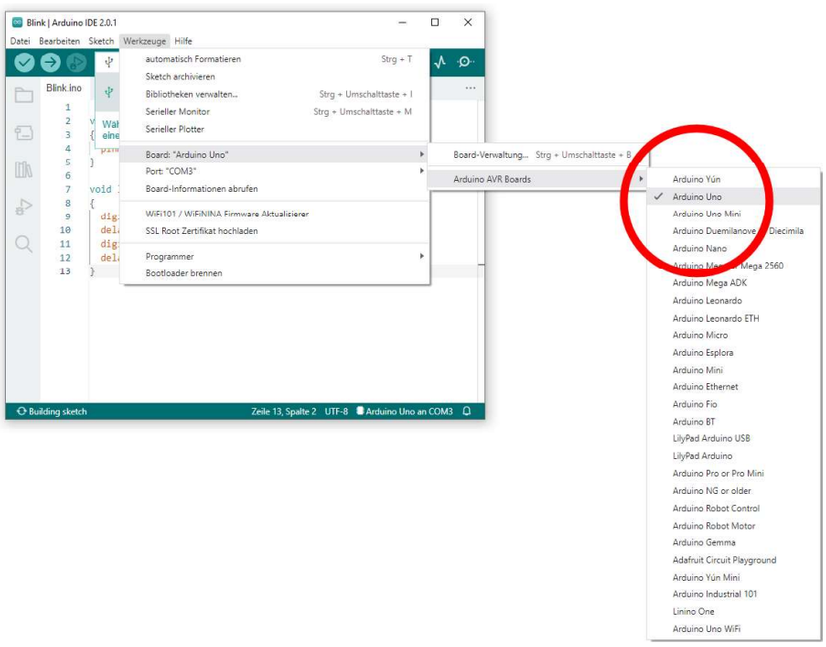
Nun muss nacheinander die Arduino-Software und der USB-Treiber für das Arduinoboard installiert werden.

2.2.2 Einrichtung der Arduinosoftware

Die jeweils aktuellste Version der Arduinosoftware kann auf der Internetseite www.arduino.cc heruntergeladen werden. Nach dem Download der Programmdatei beginnt die Installation. Sollte die Installation nicht automatisch beginnen, muss sie mit einem Doppelklick auf das heruntergeladene Programm gestartet werden. Während dieser Installation sollte noch kein Arduinoboard am Computer angeschlossen sein.
Nach der erfolgreichen Installation öffnet man den Arduino-Softwareordner und startet das Programm mit der Datei „arduino.exe“.

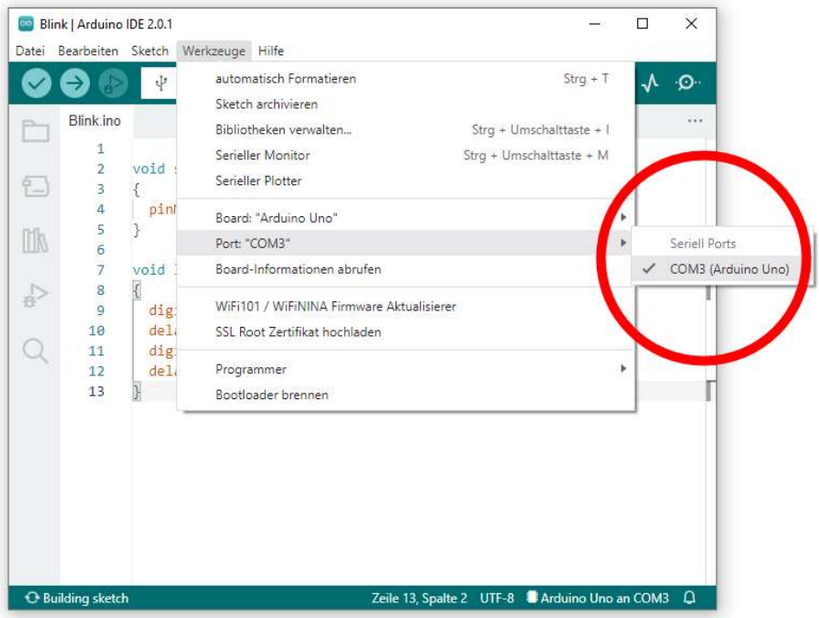
Zwei **wichtige Einstellungen** gibt es in der Software zu beachten, die im Bereich „Werkzeuge“ vorgenommen werden müssen.

a) Es muss das Board ausgewählt werden, das man am Computer anschließen möchte. Das „Funduino Uno“ Board wird hier als „Arduino Uno“ erkannt und das „Funduino MEGA2560“ Board entsprechend als „Arduino MEGA 2560“.

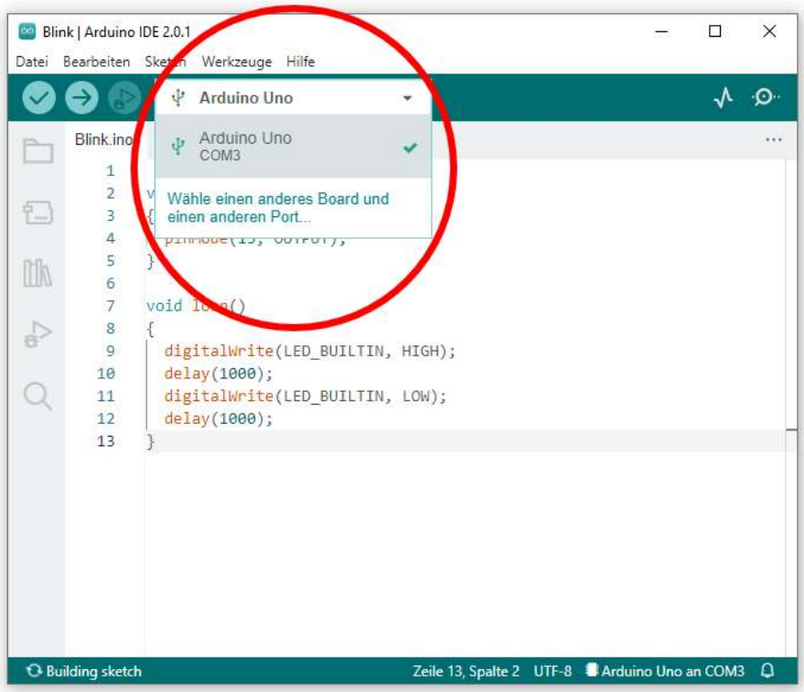


b) Es muss der richtige „Serielle Port“ ausgewählt werden. Das ist wichtig, damit der PC zuordnen kann, an welchem USB-Anschluss das Board angeschlossen ist. Die Auswahl ist jedoch nur möglich, wenn der Treiber richtig installiert wurde und der Mikrocontroller angeschlossen ist.

In den Softwareversionen ab Version V2.0 werden Mikrocontroller und Port häufig automatisch erkannt.



Wenn es nicht eindeutig ist, welcher Port zum jeweils angeschlossenen Mikrocontroller gehört, kann dies mit dem folgenden Ablauf geprüft werden: Ohne dass der Arduino-Mikrocontroller am PC angeschlossen ist, klickt man in der Software in dem Untermenü *Werkzeuge* auf „Port“. Dort werden schon ein oder mehrere Ports zu sehen sein, wie „COM1“, „COM4“, „COM7“ ... Die Anzahl der angezeigten Ports ist dabei unabhängig von der Anzahl der USB-Anschlüsse des verwendeten Computers. Wenn später das Board richtig installiert und angeschlossen ist, wird hier ein weiterer Port angezeigt.



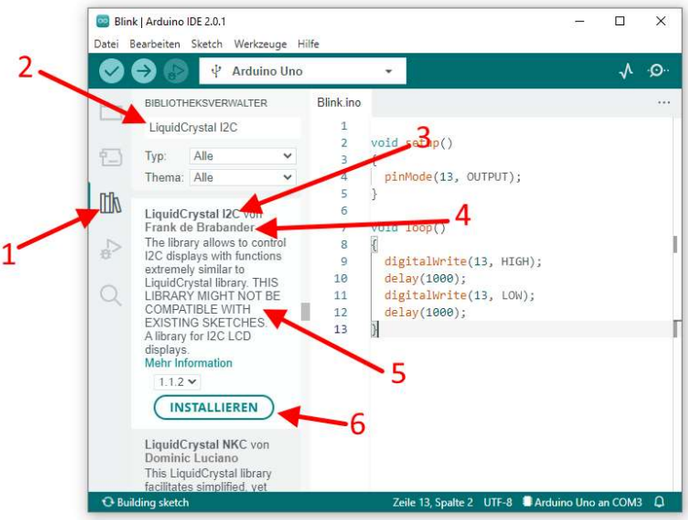
2.2.3 USB-Treiberinstallation

Im Idealfall wird bei der Treiberinstallation von Arduino-Boards oder Arduino-kompatiblen Boards mit originalem ATMEL-Chip (bspw. UNO oder MEGA von Arduino oder Funduino) das Mikrocontrollerboard an den PC angeschlossen und automatisch installiert.
Der Treiber wird jedoch je nach System nicht immer automatisch erkannt und installiert. Man sollte in dem Fall im Verlauf der Installation den Treiber selber auswählen. Er befindet sich in dem Arduino-Programmordner in dem Unterordner „Drivers“.
Kontrolle: In der Windows-Systemsteuerung des Computers findet man den *„Gerätemanager“*. Nach einer erfolgreichen Installation ist das Arduinoboard hier aufgelistet. Wenn die Installation nicht erfolgreich war, ist hier entweder nichts Besonderes zu entdecken oder es ist ein unbekanntes USB-Gerät mit einem gelben Ausrufezeichen vorhanden. In diesem Fall klickt man im Gerätemanager auf das unbekannte Gerät, wählt die Option „Treiber aktualisieren“ und folgt dann den Anweisungen auf dem Bildschirm.

Neben den Arduino-Boards oder Arduino-kompatiblen Boards gibt es mittlerweile sehr viele weitere Mikrocontrollerboards, die mit der Arduino-Entwicklungsumgebung kompatibel sind, jedoch zum Teil auf völlig anderen Mikrocontrollern basieren. Daher sind diese Treiber auch nicht in der Arduinosoftware enthalten und müssen separat installiert werden.
Der bekannteste Chipsatz, der oft in den günstigsten Boards Verwendung findet, ist der „**CH340**“ USB Chipsatz. Dieser wird in einer Vielzahl von UNO-, MEGA-, NANO- und vergleichbaren Mikrocontrollerboards verwendet.
Für die Installation lädt man sich vom Hersteller oder Händler die passenden Treiber herunter. Dabei muss in der Regel darauf geachtet werden, ob man Windows, Mac oder Linux verwendet. Bei der anschließenden Installation werden für gewöhnlich Administrationsrechte benötigt, damit der Treiber ordnungsgemäß installiert werden kann.
Praxistipp: Vor der Arbeit mit dem Mikrocontroller prüfen, ob die Installation von externen Treibern am gewünschten Computer möglich ist. Insbesondere in größeren EDV-Räumen mit Rechtevergaben etc. ist das wichtig, um unerwünschte Verzögerungen zu vermeiden.

2.2.4 Bibliotheken hinzufügen

Eine Bibliothek (auch *Library* genannt) ist für einige Projekte sinnvoll, da diese die Programmierung vereinfachen kann. Durch die Verwendung einer Bibliothek kann in einem Sketch auf Funktionen zurückgegriffen werden, sodass diese nicht komplett im Sketch ausgeschrieben werden müssen. Im praktischen Teil dieser Anleitung wird mehrfach auf solche Bibliotheken verwiesen. Diese müssen bei Bedarf erst zur Arduinosoftware hinzugefügt werden, damit sie verwendet werden können. Für das Hinzufügen einer Bibliothek zur Arduinosoftware gibt es verschiedene Möglichkeiten. Die einfachste Möglichkeit bietet sich durch einen Klick auf das Büchersymbol (1) am linken Rand der Software mit dem Namen „Bibliotheksverwalter“.



Im oberen Bereich (2) kann über das Suchfeld die gewünschte Library gesucht und betrachtet werden. In den Ergebnissen wird der Name (3) der Bibliothek angezeigt, wie auch der Autor (4). Darunter befindet sich eine Kurzbeschreibung der Bibliothek (5). Mit einem Klick auf den Button „INSTALLIEREN“ (6) wird die Bibliothek heruntergeladen und im Anschluss installiert. Mit der Installation von Programmbibliotheken werden gleichzeitig Beispielsketche zur Arduinosoftware hinzugefügt. Diese Beispiele befinden sich unter „Datei > Beispiele“ und können einen guten Einblick in die einzelnen Funktionen der jeweiligen Bibliothek geben.

Alternativ können die Bibliotheken in der Arduinosoftware im Bereich *„Sketch > Bibliothek einbinden > Bibliotheken verwalten…“* gesucht und eingebunden werden. Es gibt zusätzlich die Möglichkeit eine Bibliothek auf einer externen Seite herunterzuladen und über die „.ZIP Bibliothek hinzufügen…“ Funktion einzubinden.

Wer sich einen detaillierten Überblick über die Funktionen einer Library verschaffen möchte, kann die entsprechenden Dateien auf der Festplatte im Arduinoverzeichnis suchen und dann mit einem Editor oder auch mit der Arduinosoftware öffnen. Eine Library besteht in der Regel aus mindestens zwei Dateien, mit den Dateiendungen „ .h“ und „ .cpp“. Die Datei mit der Endung „ .h“ enthält bzw. beschreibt die Funktionen, während sich in der Datei mit der Endung „ .cpp“ der eigentliche Quellcode befindet.
Es ist auch möglich, selber Libraries zu erstellen. Dies ist jedoch nur für fortgeschrittene Nutzer sinnvoll.

2.3 Alternative Software

Neben der Arduino-Software gibt es weitere Möglichkeiten, Arduinoboards zu programmieren. Diese basieren häufig auf die Programmierung mittels einer grafischen Programmieroberfläche, die insbesondere im Bildungsbereich durch „Scratch“ oder die Programmierung von LEGO Mindstorms bekannt ist.

Bei dieser Art der Programmierung werden alle Elemente im Wesentlichen per Drag-and-drop zusammengebaut. Die Elemente werden durch ihre intuitiv verständliche Darstellung repräsentiert, wie z. B. Programmierbefehle durch Bausteinbilder. Diese Art der Programmierung bewirkt gerade bei Schülern eine zusätzliche Motivation, da das Tippen von Code zunächst häufig als kompliziert erachtet wird. Ein weiterer Vorteil besteht darin, dass sich durch das Wegfallen des „Eintippen“ von Programmcodes auch keine Syntaxfehler einschleichen können. Ein Nachteil besteht jedoch darin, dass das „echte“ Schreiben von Programmcodes auf diesem Wege nicht gelernt werden kann.

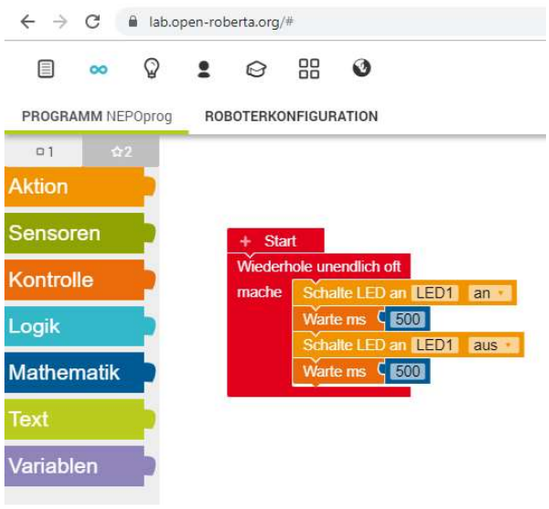
2.3.1 **Open Roberta** (deutsch)

Im **deutschen Sprachraum** ist „Open Roberta“ (https://lab.open-roberta.org) sehr bekannt. Mit Openroberta können nicht nur Arduinoboards, sondern auch viele andere Mikrocontroller und Roboter programmiert werden. Es ist also hardwareübergreifend funktional und dadurch besonders in Bildungseinrichtungen geeignet um mit Schülern das Programmieren zu erlernen.

Für die Programmierung von Arduinoboards gibt es im Openroberta-LAB den Bereich „NEPO4ARDUINO“. Im Openroberta-LAB wird für die USB-Verbindung zwischen Browser und Mikrocontroller ein zusätzliches Programm benötigt. Die Informationen zur Installation sind etwas versteckt. Man findet sie, wenn man sich durch folgenden Menüpfad klickt:

Hilfe – Allgemeine Hilfe – Vorbereitung - NEPO4ARDUINO

Das Bild zeigt ein Programm, mit dem am Arduinoboard eine LED zum Blinken gebracht wird. In diesem Fall wurde als visuelle Programmieroberfläche Openroberta verwendet.



2.3.2 Englischsprachig

Im englischen Sprachraum sind die bekanntesten visuellen Programmierplattformen **Mblock** (http://www.mblock.cc) und **S4A** „Scratch for Arduino“ (http://s4a.cat). S4A ist beliebt, da es dem regulären „Scratch“ sehr ähnlich ist, welches an vielen Schulen bereits im Informatikunterricht etabliert ist.

3. Programmieren

Damit ein Arduino-Mikrocontroller das macht, was der Benutzer verlangt, wird mithilfe der Arduino-Software ein kleines Programm geschrieben. Dieses Programm wird in der Arduino Entwicklungsumgebung als „Sketch“ bezeichnet. Ein fertiger „Sketch“ wird nach der erfolgreichen Kontrolle in der Arduino-Software auf den Speicher des Mikrocontrollers geladen und dort unmittelbar ausgeführt.

3.1 Grundstruktur für einen Sketch

Ein Sketch kann zunächst in drei Bereiche eingeteilt werden, wie hier farblich dargestellt.

```
int LED=9;                                                    Bereich 1
int helligkeit=0;
int x=5;

void setup()                                                  Bereich 2
{
 pinMode(LED, OUTPUT);
}

void loop()                                                   Bereich 3
{
 analogWrite(LED, helligkeit);
 helligkeit=helligkeit + x;
 delay(25);
 if(helligkeit==0 || helligkeit== 255)
 {
 x = -x;
 }
}
```

3.1.1 Bereich 1 - Variablen benennen

Im ersten Bereich werden Elemente des Programms benannt. Zum Beispiel werden dort Variablen festgelegt oder sog. Programmbibliotheken geladen. Dieser Teil ist nicht für jeden Sketch zwingend erforderlich.

3.1.2 Bereich 2 - Setup

Der zweite Bereich wird „Setup“ genannt. Das Setup wird vom Board nur einmal ausgeführt und ist zwingend erforderlich für jeden Sketch, selbst wenn in diesem Bereich keine Einträge erfolgen. Im Setup wird bspw. festgelegt, welcher Pin (Steckplatz für Kabel) am Mikrocontrollerboard ein Ausgang oder ein Eingang ist. Definiert als Ausgang, kann an dem jeweiligen Pin eine Spannung ausgegeben werden (bspw. um an diesem Pin eine LED leuchten zu lassen) und definiert als Eingang kann an dem Pin eine Spannung eingelesen werden (bspw. die Spannungswerte eines Sensors).

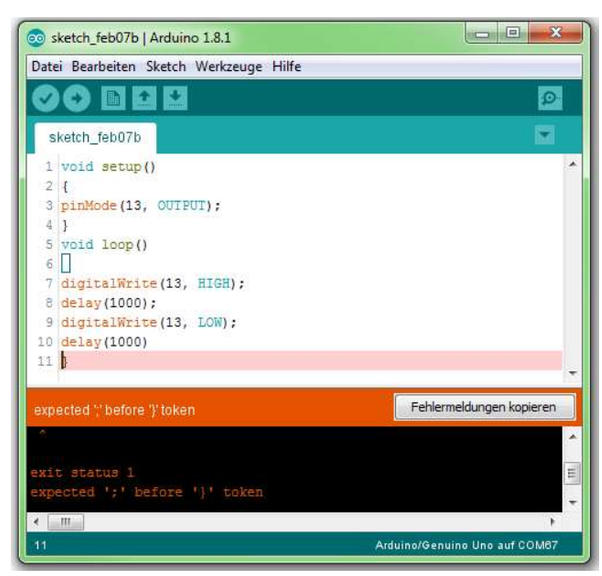
3.1.3 Bereich 3 - Loop

Der Bereich „Loop“ wird vom Board kontinuierlich wiederholt und kann daher auch als Hauptteil des Sketches bezeichnet werden. Der Mikrocontroller verarbeitet den Sketch einmal komplett bis zum Ende und beginnt dann erneut am Anfang des Loop-Abschnitts. Fortgeschrittene Anwender lagern einzelne Programmabschnitte in Unterprogramme aus, die dann vom „Loop“ nur noch aufgerufen werden und ggf. auch Daten zur weiteren Verarbeitung übergeben. Diese Auslagerungen werden in dieser Anleitung nicht weiter behandelt.

3.2 Häufige Fehlerquellen

Bei der Programmierung von Mikrocontrollern können sich an vielen Stellen Fehler einschleichen. Die häufigsten „Anfängerfehler“ während der Arbeit mit der Arduino-Software sind die folgenden beiden.

1) Das Board ist nicht richtig installiert oder es ist ein falsches Board ausgewählt. Beim Hochladen des Sketches wird im unteren Bereich der Software eine Fehlermeldung angezeigt, die in etwa so aussieht wie rechts abgebildet. Im Fehlertext befindet sich dann ein Vermerk „not in sync“.



2) Es gibt einen Fehler im Sketch. Beispielsweise ist ein Wort, eine Variable oder Befehl falsch geschrieben, oder es fehlt nur eine Klammer oder ein Semikolon. Im Beispiel links fehlt die geschweifte Klammer, die den Loop-Teil einleitet. Die Fehlermeldung beginnt dann häufig mit „expected…“. Das bedeutet, dass das Programm etwas erwartet, das noch nicht vorhanden ist.

3.3 Aufbau der Anleitungen

Die Struktur der folgenden Anleitungen ist jeweils sehr ähnlich.

1. Die Anleitungen beginnen mit einer Beschreibung der Aufgabe, die in der Anleitung abgearbeitet werden soll. Außerdem gibt es, falls erforderlich, eine Erklärung zu den verwendeten Bauteilen.
2. Vor dem jeweiligen Sketch befindet sich eine Skizze um die Verkabelung aller Module zu verdeutlichen.
3. Die abgedruckten Sketche in den folgenden Anleitungen bestehen gleichzeitig aus Programmcode und einer Beschreibung, welche die Wirkung des jeweiligen Programmcodes erklärt.

Im **linken Bereich ist der Programmcode** in schwarzer oder farbiger Schrift abgedruckt, während sich **im rechten Bereich** hinter dem Zeichen „//“ in grauer Schrift **die Erklärung** zum Programmcode befindet. Die

Erklärungen in grauer Schrift dürfen mit in in die Arduino-Software eingegeben werden und haben keinen Einfluss auf den Ablauf des Sketches.

Beispiel:

Links: Programmcode in fetter Schrift **Rechts:** Erklärungen zum Programmcode

```
void setup()              //Hier beginnt das Setup.
{                         //Hier beginnt ein Programmabschnitt.
pinMode(13, OUTPUT);      //Pin 13 soll ein Ausgang sein.
}                          //Hier ist ein Programmabschnitt beendet.

void loop()               //Hier beginnt das Hauptprogramm.
{                         //Programmabschnitt beginnt.
digitalWrite(13, HIGH); //Schalte die Spannung an Pin13 ein. (LED an)
delay(1000);              //Warte 1000 Millisekunden (eine Sekunde).
digitalWrite(13, LOW);  //Schalte die Spannung an Pin13 aus.(LED aus)
delay(1000);              //Warte 1000 Millisekunden (eine Sekunde).
}                         //Programmabschnitt beendet.
```

Wer sich nach diesem System durch die Programme arbeitet, wird den Programmcode in kurzer Zeit selber durchschauen und anwenden können. Danach kann man sich selbst mit weiteren Funktionen oder Modulen vertraut machen. Diese Anleitung stellt einen Einstieg in die Arduino Entwicklungsumgebung dar und vermittelt einen Einblick in die Programmiermöglichkeiten. Einen Gesamtliste aller Programmcodes wird auf der Internetseite „www.arduino.cc“ unter dem Punkt „Reference“ aufgeführt und beschrieben.

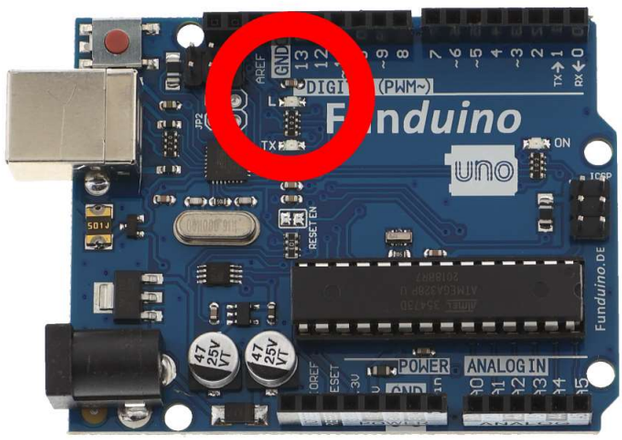
4. Praxisbezogene Anleitungen

Im folgenden Abschnitt wird anhand von praktischen Beispielen mit Bezug zu vorgegebener Hardware das Arbeiten mit der Arduino Entwicklungsumgebung erarbeitet.

4.1 Eine blinkende LED

Aufgabe: Eine Leuchtdiode soll blinken.

Materialbox
1x Arduino-Board



Auf dem Arduino Mikrocontrollerboard ist an Pin 13 bereits eine LED eingebaut (für Testzwecke). Häufig blinkt diese Lampe schon, wenn man ein neues Arduinoboard anschließt, da das Blink-Programm zum Testen des Boards je nach Hersteller bereits vorab installiert ist. Wir werden dieses Blinken jetzt selbst programmieren und die Blinkgeschwindigkeit verändern.

Schaltung:
Die auf dem Board vorhandene LED ist in dem Bild rot eingekreist. Es muss lediglich das Arduinoboard per USB-Kabel mit dem Computer verbunden werden.

Programmbereich 1 wird nicht benötigt.
Programmbereich 2, Setup:
Wir benötigen für diese Aufgabe nur einen Pin des Mikrocontrollerboards, den Pin 13. An Pin13 soll eine Spannung ausgegeben werden, denn die LED soll schließlich leuchten. Daher muss im Setup angegeben werden, dass es sich bei dem Pin13 um einen Ausgang handelt. Die Erklärungen hinter dem doppelten Schrägstrich „//“ in grauer Schrift dürfen mit in die Arduinosoftware eingegeben werden, haben jedoch keinen Einfluss auf den Ablauf des Sketches. Das sogenannte Auskommentieren ist beim Programmieren sehr sinnvoll, um für sich selber Informationen zum Programm, oder einfach nur Geistesblitze zu hinterlassen.

Wir schreiben mitten in das weiße Eingabefeld der Arduinosoftware den folgenden Sketch. Dabei muss lediglich der fett gedruckte Programmcode auf der linken Seite abgetippt werden. Die *„auskommentierten“* Informationen zum Sketch können weggelassen werden.

```
void setup()          //Hier beginnt das Setup.
{                     //Hier beginnt ein Programmabschnitt.
pinMode(13, OUTPUT); //Pin 13 soll ein Ausgang sein.
}                     //Hier ist ein Programmabschnitt beendet.
void loop()           //Hier beginnt das Hauptprogramm.
{                      //Hier beginnt ein Programmabschnitt.
digitalWrite(13, HIGH);//Schalte die Spannung an Pin13 ein (LED an).
delay(1000);           //Warte 1000 Millisekunden (eine Sekunde).
digitalWrite(13, LOW);//Schalte die Spannung an Pin13 aus (LED aus).
delay(1000);           //Warte 1000 Millisekunden (eine Sekunde).
}                      //Programmabschnitt beendet.
```

Nach der letzten geschweiften Klammer im Loop-Teil ist der Sketch abgeschlossen. Wenn der Sketch innerhalb des *„Loop“* vom Mikrocontroller bis zur letzten Klammer abgearbeitet wurde, beginnt der Sketch wieder vorne im Loop-Teil. In diesem Beispiel geht dadurch die LED immer wieder an und aus.

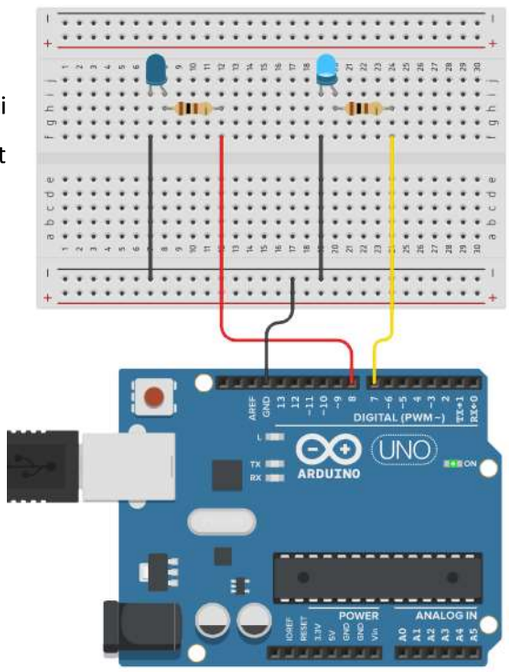
Der Sketch sollte nun exakt so aussehen, wie er auf dem Bild rechts dargestellt ist. Er muss jetzt nur noch auf das Board hochgeladen werden. Das funktioniert mit der rot umkreisten Schaltfläche (oben links in der Software).

Das Programm kann jetzt noch variiert werden. Beispiel: Die LED soll schneller blinken. Dazu verkürzen wir die Wartezeiten (delay) von 1000 Millisekunden auf 200 Millisekunden.

```
void setup()           //Hier beginnt das Setup
{                      //Hier beginnt ein Programmabschnitt.
pinMode(13, OUTPUT);   //Pin 13 soll ein Ausgang sein.
}                      //Hier ist ein Programmabschnitt beendet.
void loop()            //Hier beginnt das Hauptprogramm
{                      //Programmabschnitt beginnt.
digitalWrite(13, HIGH); //Schalte die Spannung an Pin13 ein (LED an).
delay(200);            //Warte 200 Millisekunden
digitalWrite(13, LOW); //Schalte die Spannung an Pin13 aus (LED aus).
delay(200);               //Warte 200 Millisekunden
}                         //Programmabschnitt beendet.
```

Der neue Sketch muss nun wieder auf das Board hochgeladen werden. Bei fehlerfreier Eingabe wird die LED nun schneller blinken.

4.2 Der Wechselblinker



Aufgabe: Zwei Leuchtdioden sollen abwechselnd blinken. Dabei kann die Blinkgeschwindigkeit und der Blinkrhythmus variiert werden.

Materialbox
1x Arduino-Board
2x Leuchtdioden (blau)
2x Widerstand 100 Ohm
1x Breadboard
Einige Steckkabel

Sketch:

```
void setup()
{                    //Wir starten mit dem Setup
pinMode(7, OUTPUT); //Pin 7 ist ein Ausgang.
pinMode(8, OUTPUT); //Pin 8 ist ein Ausgang.
}
void loop()
{                          //Das Hauptprogramm beginnt.
digitalWrite(7, HIGH);     //Schalte die LED an Pin7 an.
delay(1000);               //Warte 1000 Millisekunden.
digitalWrite(7, LOW);      //Schalte die LED an Pin7 aus.
digitalWrite(8, HIGH);     //Schalte die LED an Pin8 ein.
delay(1000);               //Warte 1000 Millisekunden.
digitalWrite(8, LOW);      //Schalte die LED an Pin8 aus.
}                          //Hier am Ende springt das Programm an den Start des Loop-Teils.
Also: schalte die LED an Pin7 an… usw…
```

Je kürzer das „delay“, also die Pause zwischen dem Wechsel der Leucht- und Ruhephasen der LEDs, gewählt wird desto schneller ist der Blinkrhythmus. Der Rhythmus kann dabei so schnell eingestellt werden, dass das menschliche Auge den Wechsel der Leucht- und Ruhephasen nicht mehr erkennen kann. Mit diesem Sketch kann man die Blinkfolge von Feuerwehr- und Polizeiwagen verschiedenster Länder als Modell nachbauen. Programmiere dieses Blaulicht:
Links-Pause-Links-Pause-Links-Pause-Rechts-Pause-Rechts-Pause-Rechts-Pause