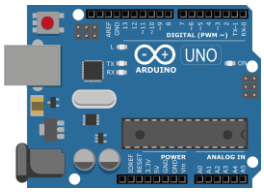
Planung eines

Schüler-Workshops

mit dem Arduino Uno

Patrick Binkert



Schüler-Workshop mit dem Arduino Uno

Inhaltsverzeichnis

1. fortlaufende To-Do-Liste 1

2. Bauteilwunschliste 2

3. Inventur 2

4. konzeptioneller Aufbau 3

4.1 Allgemein 3

4.2 Stationssammlung 4

4.3 Zuordnung der Stationen zu Modulen 6

4.4 grafische Darstellung des Workshops 7

4.5 Bauteile für das freie Experimentieren 8

4.6 Ablaufplan 8

5. relevantes Vorwissen 9

5.1 Vorwissen aus der Informatik gemäß sächs. Lehrplan 9

5.2 Vorwissen aus der Physik gemäß sächs. Lehrplan 9

5.3 Vorwissen aus Werken und TC gemäß sächs. Lehrplan 9

6. IDEs 10

6.1 Arduino IDE 10

6.2 Ardublock 11

6.3 Ardublockly 12

6.4 open-roberta 13

Patrick Binkert

7. Einführung 14

7.1 Zeitfenster 14

7.2 Was muss / sollte die Einführung alles enthalten? 14

7.3 Lernziele 15

7.4 Begrüßung 16

7.5 Vorstellung Arduino 16

7.6 elektrotechnische Grundlagen 17

7.7 IDE - Oberfläche zeigen und Programmstruktur 17

7.8 Experimentierablauf, Verhaltensweisen 17

8. nullte Station – das „Hallo Welt“ des Physical Computing 19

8.1 Lernziele (elektrotechnisch / physikalisch) 19

8.2 Lernzeile (informatisch) 19

8.3 Aufgabenstellung 20

8.4 Aufbau und Inhalte 20

8.5 Materialien 21

8.6 Aufbau der Schaltung 21

8.7 Programmcode 22

9. erste Station – „Ist es heiß hier?“ 24

9.1 Lernziele (elektrotechnisch / physikalisch) 24

9.2 Lernziele (informatisch) 24

9.3 Aufgabenstellung 25

9.4 Zusatzaufgabe 25

9.5 Aufbau und Inhalte 25

9.6 Materialien 26

9.7 Aufbau der Schaltung 26

9.8 Programmcode 27

10. zweite Station – „Auf die schiefe Bahn geraten!“ 30

10.1 Lernziele (elektrotechnisch / physikalisch) 30

10.2 Lernziele (informatisch) 30

10.3 Aufgabenstellungen 30

10.4 Aufbau und Inhalte 31

10.5 Materialien 31

10.6 Aufbau der Schaltung 32

10.7 Programmcode 33

11. dritte Station – „It's the survival of the fittest!“ 35

11.1 Lernziele (elektrotechnisch / physikalisch) 35

11.2 Lernziele (informatisch) 35

11.3 Aufgabenstellung 35

11.4 Zusatzaufgabe 36

11.5 Aufbau und Inhalte 36

11.6 Materialien 37

11.7 Aufbau der Schaltung 38

11.8 Programmcode 38

12. vierte Station – „1+1=10“ 42

12.1 Lernziele (elektrotechnisch / physikalisch) 42

12.2 Lernziele (informatisch) 42

12.3 Aufgabenstellung 43

12.4 Zusatzaufgabe 43

12.5 Aufbau und Inhalte 43

12.6 Materialien 44

12.7 Aufbau der Schaltung 44

12.8 Programmcode 45

13. fünfte Station – „Halt Stop – jetzt fahr ich!“ 47

13.1 Lernziele (elektrotechnisch / physikalisch) 47

13.2 Lernziele (informatisch) 47

13.3 Aufgabenstellung 47

13.4 Zusatzaufgabe 48

13.5 Aufbau und Inhalte 48

13.6 Materialien 49

13.7 Aufbau der Schaltung 49

13.8 Programmcode 50

14. sechste Station – „Du siehst rot? – Ich sehe nur (255,0,0)!“ 52

14.1 Lernziele (elektrotechnisch / physikalisch) 52

14.2 Lernziele (informatisch) 52

14.3 Aufgabenstellung 53

14.4 Zusatzaufgabe 54

14.5 Aufbau und Inhalte 54

14.6 Materialien 55

14.7 Aufbau der Schaltung 55

14.8 Programmcode 57

15. siebte Station – „smartCar - smartParking“ 62

15.1 Lernziele (elektrotechnisch / physikalisch) 62

15.2 Lernziele (informatisch) 62

15.3 Aufgabenstellung 63

15.4 Zusatzaufgabe 63

15.5 Aufbau und Inhalte 63

15.6 Materialien 65

15.7 Aufbau der Schaltung 65

15.8 Programmcode 66

16. achte Station – „Viva la Wetter“ 72

16.1 Lernziele (elektrotechnisch / physikalisch) 72

16.2 Lernziele (informatisch) 72

16.3 Aufgabenstellung 72

16.4 Zusatzaufgabe 73

16.5 Aufbau und Inhalte 73

16.6 Materialien 74

16.7 Aufbau der Schaltung 75

17. Programmcode 76

17.1 Teil 1 – DTH11-Sensor 76

17.2 Teil 2 – Ergänzung um Luftdrucksensor 77

17.3 Teil 3 – Ergänzung um einen LCD-Bildschirm 78

17.4 Teil 4 – Zusatzaufgabe – Hinzufügen eines Lichtsensors 80

18. neunte Station – „Wieder mal viel Lärm um nichts!“ 83

19. zehnte Station – „Ganz schön dicke Luft hier… „ 84

20. elfte Station – „Da geht mir glatt ein Licht auf!“ 85

20.1 Lernziele (elektrotechnisch / physikalisch) 85

20.2 Lernziele (informatisch) 85

20.3 Aufgabenstellung 85

20.4 Zusatzaufgabe 86

20.5 Aufbau und Inhalte 86

20.6 Materialien 87

20.7 Aufbau der Schaltung 87

20.8 Programmcode 88

1. fortlaufende To-Do-Liste

Status	Beschreibung
✔	Übersichtsdokument erstellen
✔	modularisierte System planen → Verortung der Inhalte und Ziele
✔	Inhalte der Module selektieren und strukturieren
✔	Lehrplan Inhalte der Informatik, der Physik und von Werken, sowie TC
✔	Stationsübersicht
✔	IDEs anschauen und vergleichen
✔	**Inhalte der Einführung**
✔	**konzeptioneller Aufbau an Aachen anpassen**
✔	**Inhalte der Einführung**
✔	**Konzeption nullte, erste und zweite Station**
✗	**Konzeption der anderen Stationen**

2. Bauteilwunschliste

Status	Bauteil
✗	Lagesensor
✗	Farbsensor
✗	RFID Kit
✗	7 Segment Anzeige
✗	HK-A5 Laser PM2.5/10

3. Inventur

Bauteil	Menge
Potentiometer $10\ k\Omega$	13
LCD I2C Display	16
Ultraschallsensor	12
Joystick	11
Farbsensor	12
Fotowiderstand	12
Wassersensor	12
Feuchtigkeitssensor	2
7 Segment Anzeige	12
vierstellige 7-Segment-ANzeige	12
Zeitmodule (real time clock)	12
Servomotor	12

4. konzeptioneller Aufbau

4.1 Allgemein

- Entwicklung eines Workshops für Schülerinnen und Schüler für 5-6 h mit gleichzeitiger Möglichkeit der Integration in den Schulunterricht für Lehrerinnen und Lehrer
- Erstellung einer Stationssammlung (ca. 10 Stationen)
- eine Station wird dabei für die Schülerinnen und Schüler zwingend verpflichtend sein, da dort die elementaren Grundlagen gelegt werden
- **im Workshop Szenario** können die Schülerinnen und Schüler nach der ersten Station frei wählen wie sie weiter verfahren – um eine bestmögliche Differenzierung zu gewährleisten wird über eine Abbildung (eine Art Tacho von leicht bis schwer) die Schwierigkeit deutlich gemacht, diese wird weiterhin unterschieden in:
    - Schwierigkeit bei der Algorithmierung
    - Schwierigkeit hinsichtlich der Menge an Code
    - Schwierigkeit hinsichtlich des elektronischen Aufbaus
- **für den Einsatz im schulischen Kontext** werden die Stationen zu Modulen zusammengefasst, um den Lehrerinnen und Lehrern eine Aufteilung auf die jeweiligen Unterrichtsstunden (90 min) zu erleichtern
- demnach ergibt sich eine modulare, frei kombinierbare und von der zeitlichen aufeinander Reihung der Module abhängige Struktur
- **in beiden Kontexten** steht allem voran eine Einführung in den Arduino, Grundlagen der Elektronik (lediglich kleiner Abriss → hauptsächlich Widerstände, LEDs und Betriebsspannung interessant), Einführung in die ausgewählte IDE und die Bearbeitung einer nullten Station
- alle Materialien werden als OER zur Verfügung gestellt

4.2 Stationssammlung

Station	Titel	Inhalte
0	Das „Hallo Welt“ des Physical Computing	LEDs, Taster, Vorwiderstand, Breadboard, …
1	Ist es heiß hier?	serieller Monitor, analoger Input
2	Auf die schiefe Bahn geraten!	Aufbau Lagesensor → Bedingung für den Stromfluss (geschlossener Stromkreis), akustische und optische Ausgabe
3	It's the survival of the fittest!	Reaktionsspiel → Zufallsfunktion, Tastereingaben, Ausgabe eines Signal welches den schnelleren Spieler deutlich identifizierbar macht **Zusatz**: Zeitausgabe über seriellen Monitor
4	1+1=10	Binäraddierer über Pins, wenn Stromfluss – dann 1 – sonst 0, das „7.999 Problem“, Ausgabe über seriellen Monitor oder über LCD-Screen
5	Halt Stop – jetzt fahr ich!	Ampelphasen aufmalen, Realisierung einer Ampelschaltung **Zusatz**: Taster und Fußgängerampel
6	Du siehst rot? – Ich sehe nur (255,0,0)!	RGB-Farbmodell, zunächst RGB-LED ansteuern, dann Farbsensor einbinden, verschieden farbige Blätter „abscannen“ **Variante 1:** Ausgabe der Farbe mit RGB-LED

		Variante 2: Halbkreis mit verschiedenen Farben → Ansteuerung eines Pfeils mit Hilfe eines Servomotors zum jeweiligen Farbsegment **Zusatz**: weitere Farben ergänzen
7	smartCar - smartParking	Infrarotsensor, Ultraschallsensor, LEDs→ beide Vorstellen und die SuS entscheiden mit was sie dieses Projekt umsetzen wollen, Ausgabe über seriellen Monitor **Zusatz:** LEDs als optische Warnung, sowie Piepton als akustische Warnung
8	Viva la Wetter	Luftfeuchtesensor, Temperatursensor, Luftdrucksensor, Ausgabe in einer ersten Stufe über seriellen Monitor, dann über LCD-Screen **Zusatz**: Einbinden eines Lichtstärkesensor, Ausgabe in der Einheit Lux
9	Wieder mal viel Lärm um nichts!	Lärmometer, Mikrofon, Servomotor, Halbkreis mit Emojis → leise = freundlich, laut = böse
10	Ganz schön dicke Luft hier…	Einbinden des Feinstaubsensors, Ausgabe über LCD-Screen, Warnung bei Gesundheitsgefahr
11	Da geht mir glatt ein Licht auf!	IR-Photodiode, IR-LED, wenn Lichtschranke unterbrochen lasse LED aufleuchten, 3 Schranken aufbauen – mit Spielzeugauto dazwischenfahren – immer nur LED leuchten lassen wo das Auto **Zusatz**: Geschwindigkeit berechnen

4.3 Zuordnung der Stationen zu Modulen

→ Ziel ist es, die Module so zu gestalten bzw. zu kombinieren, dass damit ein gängiger 90-Minuten Unterrichtsblock optimal gefüllt ist

Modul	Stationen
Einführung	- theoretischer Input - nullte Station (für alle) - **eine frei wählbare Station:** o Station 1 - „Ist es heiß hier?“ o Station 2 - „Auf die schiefe Bahn geraten.“
Einstiegsaufgaben	- frei wählbare Stationen - mindestens zwei umsetzen - **Stationen:** o Station 3 - „It's the survival of the fittest!“ o Station 4 - „1+1=10“ o Station 5 – „Halt Stop – jetzt fahr ich!“ o Station 6 - „Du siehst rot? – Ich sehe nur (255,0,0)!“ (zwei verschiedene Varianten)
weiterführende Aufgaben	- frei wählbare Stationen - mindestens zwei umsetzen - **Stationen:** o Station 7 - „smartCar - smartParking“ o Station 8 – „Viva la Wetter“ o Station 9 – „Wieder mal viel Lärm um nichts!“ o Station 10 – „Ganz schön dicke Luft hier…“ o Station 11 – „Da geht mir glatt ein Licht auf!“
freies Experimentieren	- nur verbleibende Bauteilübersichten herausgeben → Zeit für eigene Projekte / Ideen

4.4 grafische Darstellung des Workshops

Einführung

theoretischer Input (Arduino, Microcontroller)	Microcontroller und Elektronik	nullte Übung (LEDs und Taster)

Einstiegsaufgaben

frei wählbare Stationen, mindestens zwei müssen bearbeitet / umgesetzt werden	Reaktionsspiel, Binäraddierer, Ampelschaltung, Farbsensor

weiterführende Aufgaben

frei wählbare Stationen, mindestens zwei müssen bearbeitet / umgesetzt werden	Lärmometer, Feinstaubmessstation, Laternenbeleuchtung, Wetterstation, Einparkhilfe

freies Experimentieren

eigene Ideen und Projekte	zusätzliche Bauteilübersichten

4.5 Bauteile für das freie Experimentieren

- Schrittmotor
- Servomotor
- Feuchtigkeitssensor
- Infrarotsensor und Fernbedienung
- RFID Kit
- 7 Segment Anzeige

4.6 Ablaufplan

coming soon

5. relevantes Vorwissen

5.1 Vorwissen aus der Informatik gemäß sächs. Lehrplan

- **Gymnasium:**
  - **Klasse 7:** EVA-Prinzip, Informatiksysteme, Gerätertreiber, OAM-Modell
- **Oberschule:**
  - **Klasse 7:** Computeraufbau (Hardware, Prozessor, Bus, Speicher, ...), EVA-Prinzip, OAM-Modell

5.2 Vorwissen aus der Physik gemäß sächs. Lehrplan

- **Gymnasium:**
  - **Klasse 6:** Gefahren beim Umgang mit Strom, Leiter und Nichtleiter, verzweigte und unverzweigte Stromkreise, Aufbau von Stromkreisen
  - **Klasse 7:** Kennen der physikalischen Größen Spannung und Stromstärke, Gesetze der Spannung und Stromstärke, Messen von Spannung und Stromstärke
- **Oberschule:**
  - **Klasse 6:** Gefahren beim Umgang mit Strom, Leiter und Nichtleiter, verzweigte und unverzweigte Stromkreise, Aufbau von Stromkreisen
  - **Klasse 7:** Kennen elektrische Stromstärke, Stromstärkemessung, Kennen elektrische Spannung, Spannungsmessung, Gesetze für Stromstärke und Spannung, elektrische Leistung, elektrische Energie

5.3 Vorwissen aus Werken und TC gemäß sächs. Lehrplan

- **TC in Oberschule und Gymnasium:**
  - **Klasse 5/6:** EVA-Prinzip, Herstellen der Systembereitschaft
- **Werken:**
  - **Klasse 3:** Umgang mit elektrischem Strom, Bauteile, Schaltplan
  - **Klasse 4:** Begegnung mit Robotern und Automaten, EVA-Prinzip, Programmieren eines einfachen Ablaufs

6. IDEs

6.1 Arduino IDE

Die Arduino IDE wird von Arduino selbst bereitgestellt und bildet die Schnittstelle zwischen Code und Microcontroller. Die Software selbst ist dabei OpenSource und steht für alle gängigen Plattformen zur Verfügung. Die Oberfläche ist sehr übersichtlich gestaltetet und unterteilt sich in eine Menüführung am oberen Rand, dem Quellcodebereich in der Mitte und einem Bereich für Fehler- und Systemmeldungen am unteren Rand. Weiterhin integriert sind der serielle Monitor bzw. Plotter und eine Bibliotheksverwaltung. Außerdem sind bereits zahlreiche Beispielsketches mit enthalten, was vor allem Neueinsteigern eine große Hilfestellung bietet. Die Übertragung erfolgt nach Auswahl des richtigen Boards und der Auswahl des richtigen Ports ganz bequem über Buttons, direkt über dem Quellcodebereich.

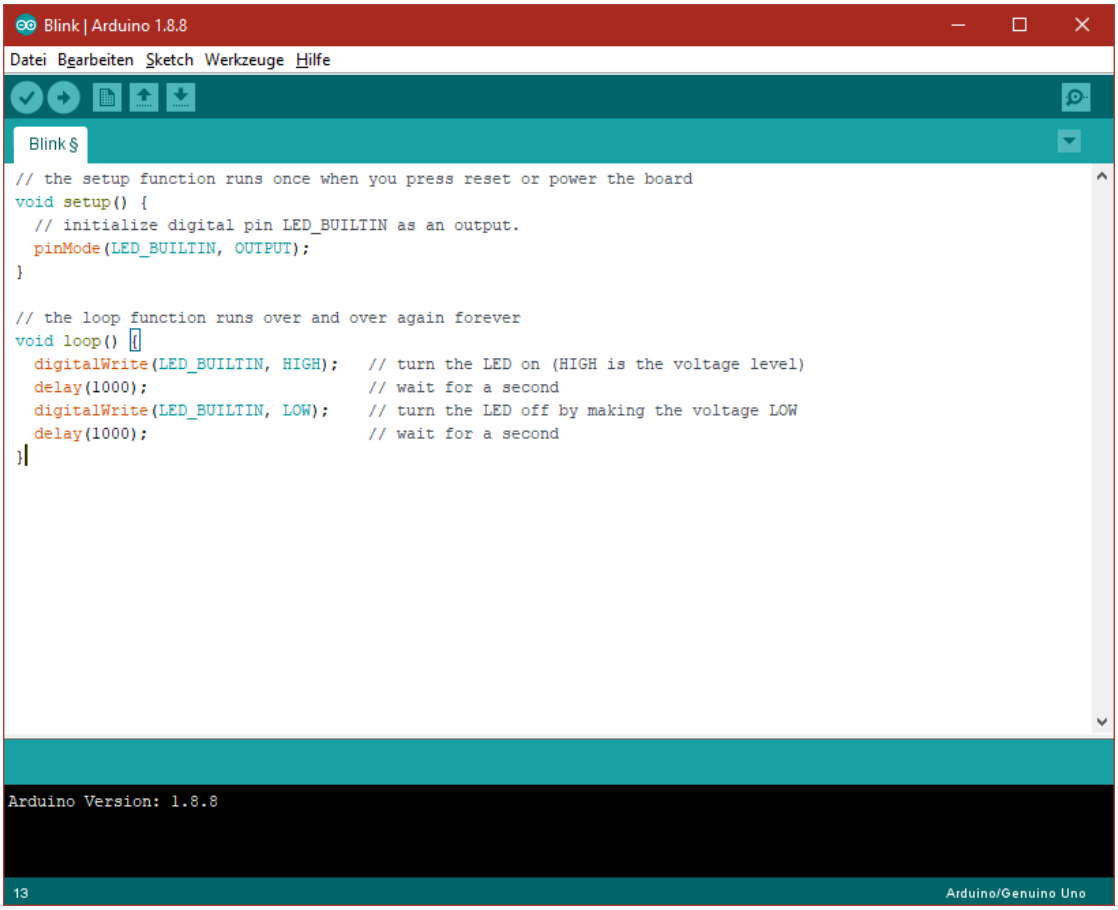


Abbildung 1: Arduino IDE

6.2 Ardublock

Die Programmierung mit Hilfe von Ardublock setzt auf das grafische Programmieren durch das Platzieren von Blöcken, ähnlich wie beim Calliope oder dem Lego Mindstorm. Das Programm ist als Javadatei verfügbar und läuft in der dazugehörigen Java Laufzeitumgebung. Allerdings läuft dieses Programm nicht selbstständig, so wird hier die Arduino IDE benötigt um das Programm letztendlich auf den Microcontroller zu übertragen, da Ardublock lediglich als Erweiterung in die Arduino DIE angebunden wird. Der Funktionsumfang ist äußert vielfältig und integriert unter der Oberfläche bereits zahlreiche Bauteile, wodurch die Arbeit mit den dazugehörigen Bibliotheken zunächst nicht notwendig beziehungsweise stark minimiert wird. Allerdings wird bei der Blockprogrammierung die Visualisierung des Quellcodes vernachlässigt, dieser kann erst anschließend in der Arduino IDE nach erfolgreicher Übertragung betrachtet werden. Ardublock ist als Java-Programm System unabhängig und läuft somit auf allen gängigen Plattformen, insofern die dazugehörige Laufzeitumgebung installiert ist. Die Installation der Erweiterung ist auf der Website nachvollziehbar beschrieben und setzt lediglich Kenntnisse im Umgang mit dem Explorer voraus.

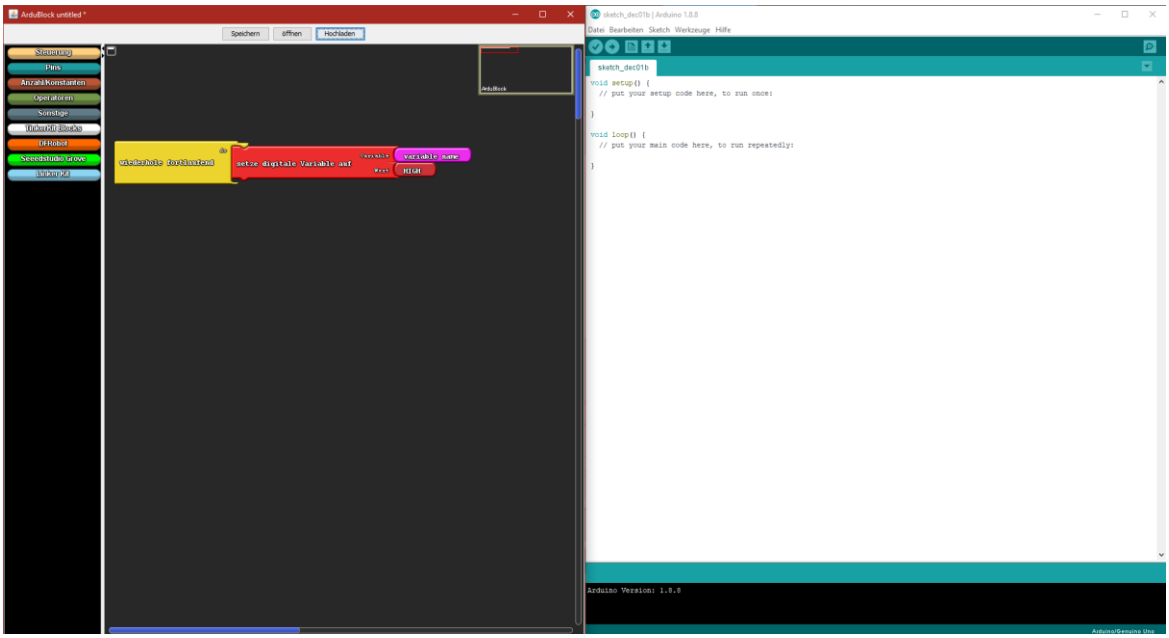


Abbildung 2: Ardublock und Arduino IDE

6.3 Ardublockly

Ardublockly ermöglicht die Programmierung des Arduinos über das Platzieren von Blöcken, ähnlich wie beim Calliope oder dem Lego Mindstorm. Um diesen Editor im Webbrowser nutzen zu können, muss zuvor das dazugehörige Programm lokal auf dem Computer ausgeführt werden. Neben den Grundlagen, wie den algorithmischen Grundstrukturen, Variablen und Funktionen, ist es weiterhin möglich direkt die Tone-Funktion zu nutzen, sowie den Schrittmotor anzusteuern. Allerdings wird die Arbeit mit Bibliotheken völlig aus und vor gelassen. Neben den Blöcken wird einem auf der rechten Seite der Quellcode in Arduino üblicher Syntax und Semantik angezeigt. Die Übertragungen und das Debugging erfolgen dennoch über die Standard Arduino IDE, welche allerdings nur als Hintergrundprozess laufen muss. Die Oberfläche ist leider nur für wenige Sprachen verfügbar, wobei deutsch nicht unterstützt wird. Die Anfrage für die deutsche Übersetzung ist allerdings bereits seit Dezember 2018 geplant und wir zeitnah umgesetzt. Zum Starten des lokalen Prozesses muss lediglich ein Python Skript ausgeführt werden, die restliche Interaktion erfolgt mit der Weboberfläche, dadurch ist auch dieser Editor für alle gängigen Plattformen verfügbar.

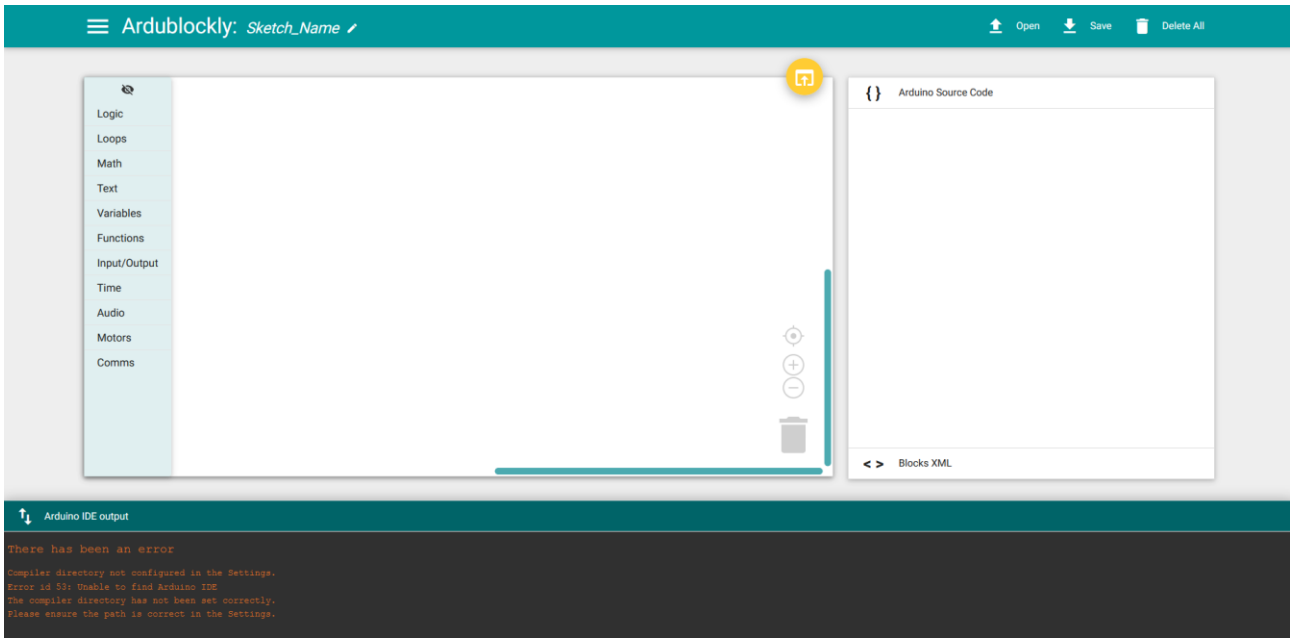


Abbildung 3: Ardublockly

6.4 open-roberta

Das open-roberta lab vereint in seinem Editor, viele verschiedene Physical Computing Plattformen. Seit neustem gehört auch der Arduino Uno dazu, auch wenn die Programmierung dessen noch in der Betaphase ist. Die Programmierung funktioniert auch hier mittels Blöcken, welche auf einer Fläche frei kombiniert werden können, insofern sie denn zusammenpassen. Die Oberfläche teilt sich auf in ein Kontextmenü auf der rechten Seite mit den notwendigen Blöcken, einer Fläche zum Platzieren der Blöcke und dem Bereich, in dem der Quellcode angezeigt wird. Hierbei kann dieser lediglich gespeichert werden und muss nachträglich erst noch auf den Arduino übertragen werden. Der Befehlsumfang umfasst auch hier alle grundlegenden Strukturen, allerdings ist positiv anzumerken, dass zusätzlich zum Servomotor bereits der LCD-Screen über den I²C-Port integriert. Zur Übertragung des Programms ist schlussendlich wieder die Arduino IDE notwendig. Dadurch das open-roberta im Webbrowser läuft, ist auch dieser Editor auf allen Plattformen verfügbar. Ein wesentliches Manko stellt die fehlerhafte Beschriftung am oberen Rand dar, wo die Rede von einer „Roboterkonfiguration“ ist. Dies weckt bei Schülerinnen und Schüler ein falsches Erwartungsbild und sollte zuvor deutlich kommuniziert werden.

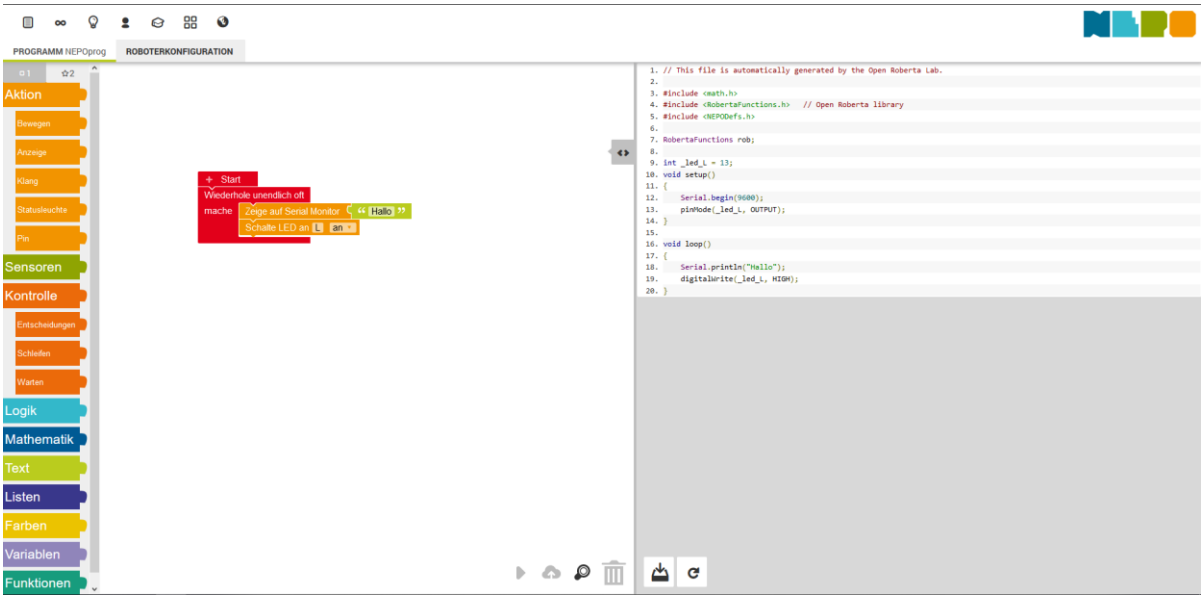


Abbildung 4: open-roberta lab im Bereich Arduino-Programmierung

7. Einführung

7.1 Zeitfenster

- die theoretische Einführung sollte zwischen 30 und 40 Minuten dauern und diese Zeit nicht wesentlich überschreiten
- nach ca. 15 Minuten sollten die Schülerinnen und Schüler das erste Mal etwas „selbst tun“ da sonst die Aufmerksamkeit nicht mehr gegeben ist
- im Anschluss werden die Arduinos und Bauteile ausgeteilt
- danach erfolgt direkt die nullte Übung
- anschließend eine Pause

7.2 Was muss / sollte die Einführung alles enthalten?

- Vorstellung der Betreuenden
- Fahrplan vorstellen (Grundlagen – nullte Übung – Stationsarbeit)
- Motivation und Einführung mit Leitfragen → Stichwortsammlung mit den Schülerinnen und Schülern erarbeiten, vielleicht Tools wie *Classflow*, *Etherpad* oder *Wordle* benutzen:
    - Was ist eigentlich programmieren?
    - Welche aus dem Alltag bekannten Geräte können programmierbar sein?
    - Was benötigt ein Gegenstand um ihn zu Programmieren? → EVA-Prinzip, Prozessor (vgl. mit Gehirn), Energie (Nahrung vs. Elektrizität), Interaktion mit der Umwelt (Sinne vs. Sensoren, Gliedmaßen vs. Aktoren)
- Was ist ein Mikrocontroller? → Halbleiterchips mit Prozessor und Peripheriegeräten, häufig in embedded systems zu finden
- Microcontroller im Alltag? → Telefon, Fahrkartenautomat, Kaffeemaschine, Wecker, Radio, Autos, Digitalkamera, Drohnen, … → LearningApp (Zuordnungsspiel) mit Microcontroller vs. ohne Microcontroller

- Vorstellung Arduino
- Grundlagen Elektronik (lediglich kleiner Abriss → hauptsächlich Widerstände und Betriebsspannung interessant)
- IDE (Arduino IDE oder Ardublockly oder Ardublock oder open-roberta)
- prinzipieller Programmaufbau → `void setup()` und `void loop()`
- Erläuterung der Module beziehungsweise Stationen, sowie der Arbeitsweise
- Experimentierregeln nennen
- nullte Übung

7.3 Lernziele

- Die SuS können Geräte aus ihrem Alltag benennen, in denen Mikrocontroller verbaut sind.
- Die SuS können die Aufgabe eines Mikrocontrollers in einem Gerät aus ihrer Lebenswelt nach dem EVA-Prinzip erläutern.
- Die SuS nutzen ihr Grundlagenwissen aus dem Informatikunterricht, genauer dem EVA-Prinzip, zur Beschreibung der Funktionsweise eines Microcontrollers an.
- Die SuS reaktivieren Grundlagenwissen aus der Physik als Basis für den Umgang mit den elektronischen Bauteilen.
- Die SuS kennen den Aufbau des Arduino Boards und benennen elementare Bauteile und deren Funktion.
- Die SuS benennen Ein- und Ausgabekomponenten des Arduino Unos.
- Die SuS sind in der Lage Schaltungen mit Ein- und Ausgabegeräten zu entwerfen und den Programmcode zur Ansteuerung der Komponenten umzusetzen.
- Die SuS gewinnen einen Überblick über die Arduino IDE und erläutern die Unterscheidung im Programmcode zwischen `void setup()` und `void loop()`.
- Die SuS wenden die *loop*-Methode, für eine wiederkehrende Anweisungsfolge und die *setup*-Methode für einmalige Einstellungen, an.

7.4 Begrüßung

- Vorstellung der Betreuenden
- Was haben wir heute vor? → Fahrplan für heute (Grundlagen – nullte Übung – Stationsarbeit)
- Wir möchten natürlich auch wissen wer ihr seid! → Namensschilder
- Ablauf des Workshops erläutern
- Stations- und Materialienübersicht
- Plakataushänge mit …
    - Befehlsübersichten
    - Programm IDE
    - Verhaltensregeln
    - Aufbau Arduino
    - Grundlagen der Elektronik

7.5 Vorstellung Arduino

- Arduino Uno als eines von vielen Boards (→ am Ende des Workshops darüber informieren, welche Boards es noch gibt, um interessierten Schülerinnen und Schülern die Möglichkeit zu geben sich selbstständig damit auseinanderzusetzen)
- **Ausgaben** über: LEDs, Motoren, Lautsprecher, LCD-Anzeigen, …
- **Eingaben** über: Taster, Schalter, Spannungen und Widerstände, Sensoren (Ultraschall, Infrarot, Temperatur, Beschleunigungen, Luftfeuchte, …)
- Anschlüsse am Arduino (Stromversorgung über USB oder Netzteil, digitaler I/O-Pins, analoge Output-Pins, 5V, Ground, …)
- Reset-Button
- Prozessor (ATmega328) auf dem Arduino

7.6 elektrotechnische Grundlagen

- Betriebsspannung des Arduino
- geschlossene Stromkreise, als Bedingung für den Stromfluss
- Funktionsweise bzw. Aufbau des Breadboards
- anhand der LED → Vorwiderstand, Polarität (langes / kurzes Beinchen)
- Taster → PULL-UP-Schaltung oder mit extra Widerstand
- analoge, digitale und PCM Pins

7.7 IDE - Oberfläche zeigen und Programmstruktur

- **Datei**: Speichern, Öffnen, Beispiele
- **Werkzeuge**: Board, Port, serieller Monitor, serieller Plotter
- **Buttons**: Überprüfen und Hochladen
- **Statusfenster** (Übersetzungsvorgang, Übertragungsstatus, Fehler)
- **prinzipieller Programmaufbau** → void setup() und void loop()
- Demonstration einer Fehlermeldung mit einem simplen Programm (LED an) bei nicht korrekt eingestelltem Port und falsch ausgewählten Board
- Demonstration einer Fehlermeldung bei falscher Syntax → zum Beispiel:

  $exit\ status\ 1$
  $expected\ ';'\ before\ 'digitalWrite'$

7.8 Experimentierablauf, Verhaltensweisen

- Verweis auf Ablaufplan (Pausenzeiten, Arbeitsphase usw.)
- Beginn mit der nullten Übung für alle
- anschließend …
  - **… im Workshop-Szenario**: freie Stationswahl → nochmals alle Stationen zeigen
  - **… im Unterrichtsszenario**: Module erklären, dazugehörige Stationen benennen, Anzahl umzusetzender Stationen benennen

- **Experimentierregeln:**
  - kein Essen und Trinken an den Experimentierplätzen
  - angemessenes Verhalten → nicht herumrennen, nicht auf den Tischen sitzen, … (dies sollte bereits aus dem Schulalltag bekannt sein und nicht überbetont werden)
  - sorgfältiger Umgang mit den Materialien (Drucksachen genauso wie Technik)
  - Umbau der Schaltungen nur im stromlosen Zustand
  - bei defekten Bauteilen umgehend Bescheid sagen
  - Pins nicht verbiegen
  - bei Unklarheiten nicht zögern zu fragen
  - …

8. nullte Station – das „Hallo Welt“ des Physical Computing

8.1 Lernziele (elektrotechnisch / physikalisch)

- Die Schülerinnen und Schüler kennen den Aufbau des Breadboards und nutzen die äußeren Anschlussleisten zur dauerhaften Verbindung mit dem 5V und GND Pins.
- Die Schülerinnen und Schüler erklären, das LEDs nur einen sehr geringen Innenwiderstand haben und gleichzeitig nur wenig Strom vertragen, somit ist in Reihe zur LED ein Vorwiderstand zu schalten.
- Die Schülerinnen und Schüler sind in der Lage an einer LED die richtige Seite für die jeweilige Polarität zu identifizieren, konkret können sie die lange Seite $(+)$ und kurze Seite $(-)$ zuordnen.
- Die SuS benutzen die Grundlagen der Programmierung auf dem Arbeitsblatt zur Entwicklung des eigenen Code, durch Kombination der Codeschnipsel.
- Die SuS beschreiben die Schaltung für einen Taster, abstrahieren diesen und setzen die Schaltung mit realen Bauteilen um.

8.2 Lernzeile (informatisch)

- Die Schülerinnen und Schüler erkennen die Funktionsweise einer einfachen Verzweigung und wenden diese zur Lösung eines von Bedingungen abhängigen Problems an.
- Die Schülerinnen und Schüler unterscheiden im Programmcode drei Unterbereiche, welche die Initialisierung, $setup$- und $loop$-Methode betrifft.
- Die Schülerinnen und Schüler erkennen, dass der Programmcode innerhalb der $setup$-Methode nur einmal und Programmcode innerhalb der $loop$-Methode dauerhaft ausgeführt wird.
- Die Schülerinnen und Schüler kennen Funktionen und rufe diese mit den korrekten Parametern auf, beispielsweise die $delay$- und $pinMode$-Funktion.

8.3 Aufgabenstellung

Hallo Welt! Schon seit geraumer Zeit ist dies häufig das erste was Programmiererinnen und Programmierer in einer neuen Programmiersprache ausprobieren. Beim Physical Computing – die Programmierung physischer Systeme – ist dieses „Hallo Welt“ das Ansteuern einer LED. An dieser Station sollst du deshalb zunächst eine LED zum Leuchten bringen! Wenn du das geschafft hast, erweiterst du das Projekt so, dass die LED dauerhaft blinkt. Da eine blinkende LED auf Dauer nervt und es nicht ökologisch wertvoll ist eine Lampe dauerhaft leuchten zulassen, sollst du zum Abschluss einen Taster nutzen, um die LED ein- und auszuschalten.

8.4 Aufbau und Inhalte

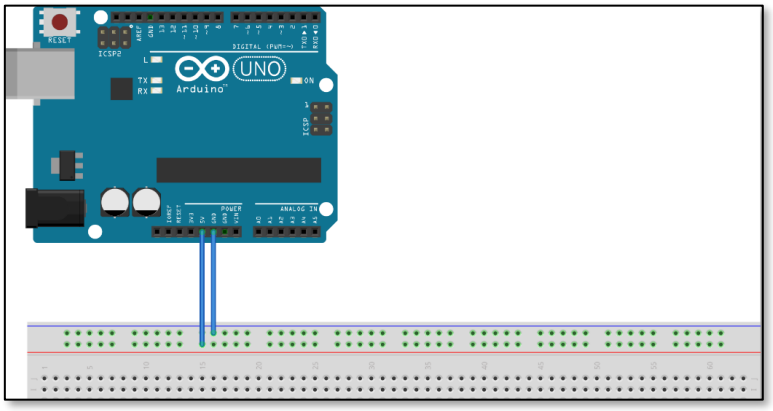
- Information zum Breadboard als Bild (verbundene Pins)
- bebilderte Anleitung zum Verbinden des Arduinos mit dem Computer → Board und Port auswählen
- erster Schritt: $5V$ und GND mit äußeren Anschlussleisten verbinden
- LED: langes / kurzes Bein, Widerstand (Ohm und Farbcodierung)
- Wie programmieren?
    - Grundgerüst ($init$, $setup$, $loop$)
    - jede Anweisung schließt mit Semikolon, jede Klammer muss auch wieder geschlossen werden
    - Variablentypen und deren Deklaration (int, $float$, $string$, $boolean$)
    - Kommentare
- Variable für LED, $pinMode$, $digitalWrite$
- $delay$-Funktion
- Aufbau einer Schaltung mit Taster, $digitalRead$, HIGH / LOW
- einfache Verzweigung

8.5 Materialien

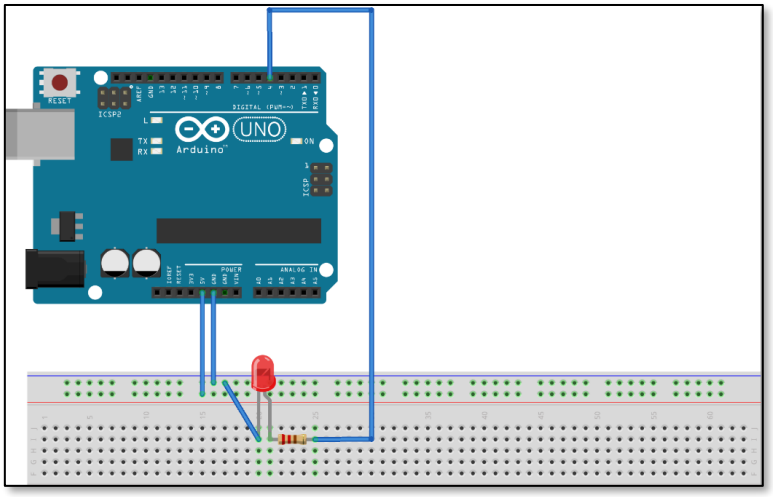
Bauteil	Menge
Arduino Uno	$1x$
Breadboard	$1x$
LED	$1x$
Kabel	$7x$
Taster	$1x$
Widerstand $(220\ \Omega)$	$2x$

8.6 Aufbau der Schaltung

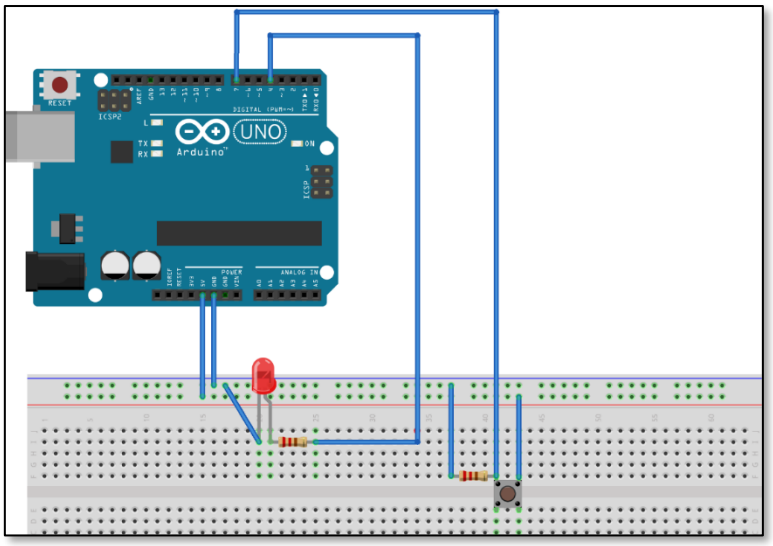
8.6.1 Teil 1 – Verbinden der Anschlussleisten



8.6.2 Teil 2 – LED ansteuern



8.6.3 Teil 3 – LED mit Taster steuern



8.7 Programmcode

8.7.1 Teil 1 – Verbinden der Anschlussleisten

```
int LED=4; //an diesem Pin ist die LED angeschlossen

void setup(){
      pinMode(LED, OUTPUT);
      //konfiguriert den spezifizierten Pin als Ausgabe(-pin)
}

void loop(){
      digitalWrite(LED, HIGH);
      //die Spannung an diesem wird HIGH gesetzt, dass bedeutet auf 5V
}
```

8.7.2 Teil 2

```
int LED=4; //an diesem Pin ist die LED angeschlossen

void setup(){
      pinMode(LED, OUTPUT); //konfiguriert den spezifizierten Pin als Ausgabe(pin)
}

void loop(){
      digitalWrite(LED, HIGH);
      //die Spannung an diesem wird HIGH gesetzt, dass bedeutet auf 5V
      delay(200); //Pause von 200ms
      digitalWrite(LED, LOW);
      //die Spannung an diesem wird LOW gesetzt, dass bedeutet auf 0V
      delay(200); //Pause von 200ms
}
```

8.7.3 Teil 3

```
int LED=4; //an diesem Pin ist die LED angeschlossen
int taster=7; //an diesem Pin ist der Taster angeschlossen
int tasterstatus=0; //der Status des Tasters wird mit 0 (low) initialisiert,
sprich "nicht gedrückt"

void setup(){
      pinMode(LED, OUTPUT);
      //konfiguriert den spezifizierten Pin als Ausgabe(pin)
      pinMode(taster, INPUT);
      //konfiguriert den spezifizierten Pin als Eingabe(pin)
}

void loop(){
      tasterstatus=digitalRead(taster); //Einlesen des Tasterstatus
      if (tasterstatus == HIGH){ //Wenn Taster gedrückt, dann ...
            digitalWrite(LED, HIGH);
            //die Spannung an diesem wird HIGH gesetzt, dass bedeutet auf 5V
      }
      else { // ... sonst ...
            digitalWrite(LED, LOW);
            //die Spannung an diesem wird LOW gesetzt, dass bedeutet auf 0V
      }
}
```

9. erste Station – „Ist es heiß hier?“

9.1 Lernziele (elektrotechnisch / physikalisch)

- Die Schülerinnen und Schüler erkennen anhand des Aufbaus den grundlegenden Aufbau von smarten Geräten und mit dem Abstraktionsschritt vernetzter Geräte die Funktionsweise des IoT. (internet of things)
- Die Schülerinnen und Schüler abstrahieren den Schaltplan und die Belegung der jeweiligen Pins des Temperatursensors und setzen dies in einer realen Schaltung um.
- Die Schülerinnen und Schüler kennen die Verfahrensweise, um ein analoges Signal in sinngebende Werte umzuwandeln.
- Die Schülerinnen und Schüler kennen die Funktionsweise eines Potentiometers.

9.2 Lernziele (informatisch)

- Die Schülerinnen und Schüler erkennen die grundlegende Funktionalität eines Debuggers, welcher in der Arduino IDE durch den seriellen Monitor repräsentiert wird.
- Die Schülerinnen und Schüler nutzen den seriellen Monitor, um die errechneten Ergebnisse sichtbar zu machen.
- Die Schülerinnen und Schüler starten mit Hilfe der $Serial.begin(x)$-Funktion die Übertragung und geben mit der $Serial.print$- und $Serial.println$-Funktion die Daten aus.
- Die Schülerinnen und Schüler interpretieren die übertragenden Daten auf Sinnhaftigkeit und passen bei Bedarf den Programmcode an.
- Die Schülerinnen und Schüler lesen die Daten eines analogen Sensors aus und mappen die Ergebnisse anschließend mit Hilfe der map-Funktion zum Festlegen des Pausenwertes.

9.3 Aufgabenstellung

Jeder empfindet Temperaturen bekanntlich anders, total im Gegensatz zu einem Temperatursensor. Die optimale Zimmertemperatur liegt bei $22°C$. Daher wird es nun einmal Zeit, dass du eine Schaltung realisierst, welche die Temperatur mit dem dazugehörigen Sensor misst. Deinen Aufbau kannst du anschließend durch erwärmen des Sensors mit der Hand oder durch Abkühlen des Sensors mit einem Kühlpack testen.

9.4 Zusatzaufgabe

Du willst selbstverständlich das genauste Ergebnis von allen erreichen. Ergänze dein Projekt daher nun um eine Mittelwertrechnung, indem du zunächst eine beliebige Anzahl an Temperaturen aufnimmst und die Summe der einzelnen Werte durch die Messwertanzahl teilst.

9.5 Aufbau und Inhalte

- Was ist überhaupt ein analoger Sensor?
- Aufbau der Schaltung und beschriftete Pins beim Temperatursensor
- Woher weiß ich nun was für Werte der Sensor misst?
- Erklärung des seriellen Monitors (Befehle, Arduino IDE)
- Die Werte machen keinen Sinn? Das stimmt!
- **Erklärung der Umwandlung:** Nutzung der map-Funktion, Erklärung mit Hilfe eines Zahlenstrahls (Abbildung von 24 Buchstaben auf x bis y Zahlen)
- **Gut zu wissen** → Schrittweite berechnen: Analoge Spannung im Bereich: $0\ bis + 5V$, $10 - Bit -$ A/D-Wandler bedeutet $2^{10} - 1\ Werte = 1023$, Damit ergibt sich folgende Rechnung: $\frac{5V}{1023} = 0.00488\ V = 4.88\ mV$, Sensor misst Temperaturen im Bereich: $0°C$ bis $150°C$
- Modifizierung der Ausgabe mit Sensorwert, umgerechneter Temperatur und Einheit der Temperatur

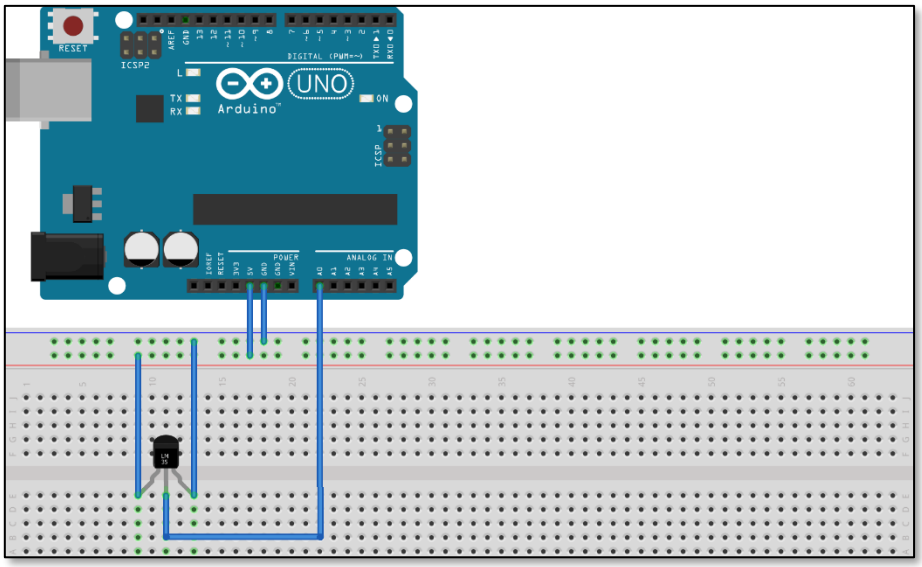
- Hinzufügen des Potentiometer zum Einstellen der Wartezeit zwischen den einzelnen Messungen unter Nutzung der map-Funktion
- Testen der Anordnung
- Aufnahme von bis zu fünf Messwerten, bilden des Mittelwertes, Ausgabe der gemittelten Temperatur

9.6 Materialien

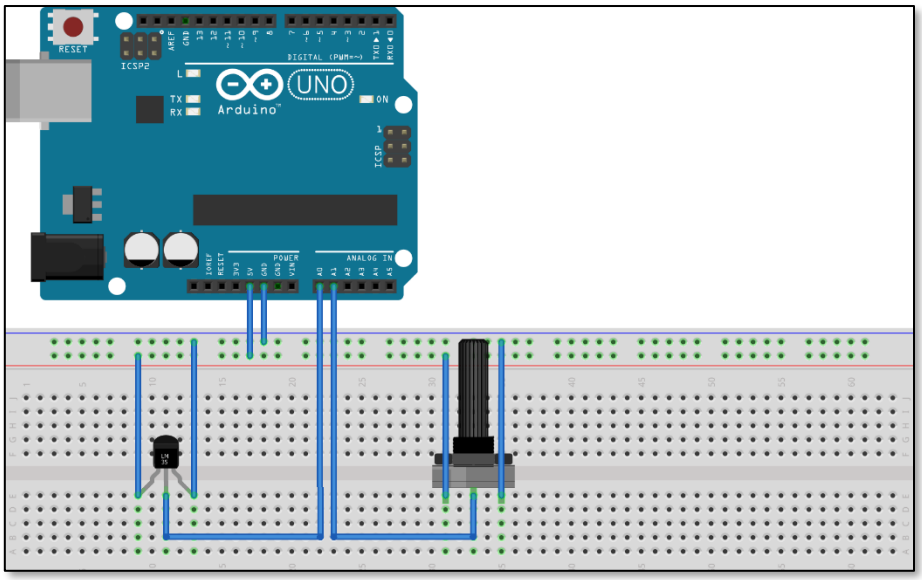
Bauteil	Menge
Arduino Uno	$1x$
Breadboard	$1x$
Potentiometer	$1x$
Temperatursensor LM35	$1x$
Kabel	$8x$

9.7 Aufbau der Schaltung

9.7.1 Teil 1 – Auslesen des analogen Temperatursensors



9.7.2 Teil 2 – Ergänzung eines Potentiometer zur Kontrolle der Wartezeit



9.8 Programmcode

9.8.1 Teil 1 - Auslesen des analogen Temperatursensors

```
int tempsensor = A0;
int sensorwert;
int temperatur = 0;

void setup() {
      Serial.begin(9600); //startet Verbindung zum seriellen Monitor
}

void loop() {
      sensorwert = analogRead(tempsensor); //Auslesen des Temperatursensors
      temperatur = map(sensorwert, 0, 1023,0,100);
      //Verknüpfen der Schrittweite mit dem Messbereich
      Serial.print("Sensorwert: "); //Ausgabe der Messergebnisse
      Serial.print(sensorwert);
      Serial.print(" => Temperatur = ");
      Serial.print(temperatur);
      Serial.println(" °C");
}
```

9.8.2 Teil 2 - Ergänzung eines Potentiometer zur Kontrolle der Wartezeit

```
int tempsensor = A0;
int potentiometerpin = A1;
int wartezeit = 0;
int sensorwert;
int temperatur = 0;

void setup() {
      Serial.begin(9600); //startet Verbindung zum seriellen Monitor
}

void loop() {
      sensorwert = analogRead(tempsensor); //Auslesen des Temperatursensors
      temperatur = map(sensorwert, 0,1023,0,100);
      //Verknüpfen der Schrittweite mit dem Messbereich
      Serial.print("Sensorwert: "); //Ausgabe der Messergebnisse
      Serial.print(sensorwert);
      Serial.print(" => Temperatur = ");
      Serial.print(temperatur);
      Serial.println(" °C");
      sensorwert = analogRead(potentiometerpin); //Auslesen den Potentiometers
      wartezeit = map(sensorwert,0,1023,1000,10000);
      //Verknüpfen der Schrittweite mit dem Wartezeitintervall
      Serial.println(wartezeit); //zur Kontrolle die Wartezeit ausgeben
      delay(wartezeit); //Ausführen der Wartezeit
}
```

9.8.3 Zusatzaufgabe – Durchschnittswert der Temperatur berechnen

```
int tempsensor = A0;
int potentiometerpin = A1;
int wartezeit = 0;
int wert1;
int wert2;
int wert3;
int wert4;
int wert5;
int gemittelterwert;
int sensorwert;
int temperatur = 0;

void setup() {
      Serial.begin(9600); //startet Verbindung zum seriellen Monitor
}
```

```
void loop() {
      sensorwert = analogRead(potentiometerpin); //Auslesen den Potentiometers
      wartezeit = map(sensorwert,0,1023,1000,10000);
      //Verknüpfen der Schrittweite mit dem Wartezeitintervall
      Serial.print("Wartezeit in ms: ");
      Serial.println(wartezeit); //zur Kontrolle die Wartezeit ausgeben

      wert1 = analogRead(tempsensor); //Auslesen des Temperatursensors
      delay(wartezeit); //Ausführen der Wartezeit
      wert2 = analogRead(tempsensor); //Auslesen des Temperatursensors
      delay(wartezeit); //Ausführen der Wartezeit
      wert3 = analogRead(tempsensor); //Auslesen des Temperatursensors
      delay(wartezeit); //Ausführen der Wartezeit
      wert4 = analogRead(tempsensor); //Auslesen des Temperatursensors
      delay(wartezeit); //Ausführen der Wartezeit
      wert5 = analogRead(tempsensor); //Auslesen des Temperatursensors

      gemittelterwert = (wert1+wert2+wert3+wert4+wert5)/5;

      temperatur = map(gemittelterwert,0,1023,0,100);
      //Verknüpfen der Schrittweite mit dem Messbereich

      Serial.print("Sensorwert: "); //Ausgabe der Messergebnisse
      Serial.print(sensorwert);
      Serial.print(" => Temperatur = ");
      Serial.print(temperatur);
      Serial.println(" °C");
}
```

10. zweite Station – „Auf die schiefe Bahn geraten!“

10.1 Lernziele (elektrotechnisch / physikalisch)

- Die Schülerinnen und Schüler erklären anhand des Aufbaus vom Lagesensors, wann es sich um einen geschlossenen Stromkreis handelt.
- Die Schülerinnen und Schüler erkennen, dass es sich zur besseren Erprobung der Funktionalität anbietet, den Lagesensor nicht direkt mit dem Breadboard, sondern über Kabelverbindungen anzuschließen.

10.2 Lernziele (informatisch)

- Die Schülerinnen und Schüler definieren mit der Bedingung für einen geschlossenen Stromkreis und nutzen diese Information, um festzustellen wie die Bedingung für die einfache Verzweigung lauten muss.
- Die Schülerinnen und Schüler erkennen eine Funktion als Programmkonstrukt an, welches nach Eingaben und durch Abarbeitung beliebig vieler Verarbeitungsschritte eine Ausgabe realisiert.
- Die Schülerinnen und Schüler kennen die *tone*-Funktion und die Bedeutung derer Aufrufvariablen.
- Die Schülerinnen und Schüler bestimmen die Eingabevariablen der *tone*-Funktion, auf Basis der zur Verfügung gestellten Tonfolge in der Einheit Hertz.
- Die Schülerinnen und Schüler nutzen die einfache Verzweigung, um zu bestimmen, unter welchen Bedingungen welche LED leuchten soll.

10.3 Aufgabenstellungen

Stell dir vor du fährst mit deinem Fahrrad einen ziemlich steilen Berg hinauf. Ab einem Winkel von $45°$ besteht die Gefahr, dass du nach hinten mit dem Fahrrad umfällst. Das gilt es zu verhindern und du musst dafür das notwendige Projekt umsetzen! Um dies zu realisieren nutzt du einen sogenannten Kippsensor und zur Ausgabe der Warnung einen Lautsprecher, sowie zwei LEDs (grün und rot).

10.4 Aufbau und Inhalte

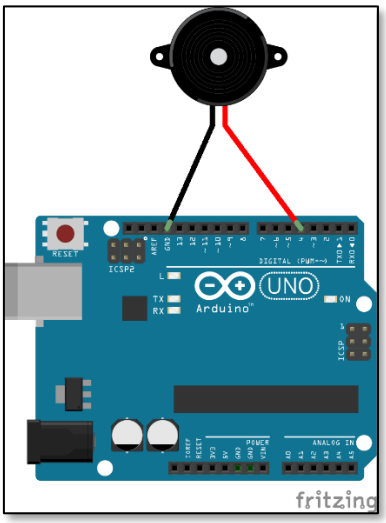
- Aufbau Lagesensor erklären
- Wann handelt es sich um einen geschlossenen Stromkreis?
- Wie kann eingelesen werden, wann es sich um einen geschlossenen Stromkreis handelt? → $digitalRead$
- Lautsprecherbelegung, $tone$-Funktion, $noTone$-Funktion
- Testen des Lautsprecher mit Martinhorn-Tonfolge und / oder „Alle meine Entchen“
- Hinzufügen des Lagesensor → Ausgabe ob offener oder geschlossener Stromkreis über seriellen Monitor
- einfache Verzweigung → wenn Winkel überschritten, dann Ton
- Hinzufügen zweier LEDs (rot und grün): wenn Winkel kleiner – grüne LED, wenn Winkel überschritten – rote LED blinken lassen
- Testen durch Neigen des Breadboards

10.5 Materialien

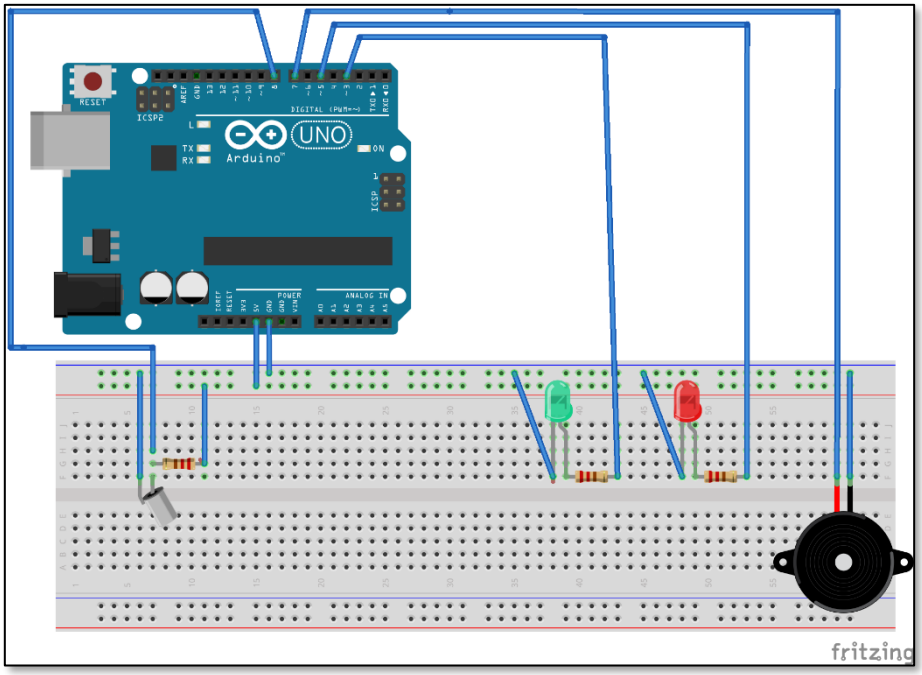
Bauteil	Menge
Arduino Uno	$1x$
Breadboard	$1x$
Lautsprecher	$1x$
Kippsensor (SW-520D)	$1x$
Widerstand ($220\ \Omega$)	$3x$
LED (rot und grün)	$2x$
Kabel	$10x$

10.6 Aufbau der Schaltung

10.6.1 Teil 1 – Testen des Lautsprechers



10.6.2 Teil 2 – Kippsensor mit Warnausgabe



10.7 Programmcode

10.7.1 Teil 1 – Testen des Lautsprechers – Variante mit einem Martinhorn

```
int lautsprecher = 4;

void setup() {
}

void loop() {
    tone(lautsprecher, 622);
    delay(1000);
    tone(lautsprecher, 466);
    delay(1000);
}
```

10.7.2 Teil 1 – Testen des Lautsprechers – Variante mit „Alle meine Entchen“

```
int lautsprecher = 4;

void setup() {
}

void loop() {
      tone(lautsprecher, 262);
      delay(200);
      noTone(lautsprecher);
      delay(200);

      tone(lautsprecher, 294);
      delay(200);
      noTone(lautsprecher);
      delay(200);

      tone(lautsprecher, 330);
      delay(200);
      noTone(lautsprecher);
      delay(200);

      tone(lautsprecher, 349);
      delay(200);
      noTone(lautsprecher);
      delay(200);

      …
}
```

10.7.3 Teil 2 – Kippsensor mit Warnausgabe

```
int neigungssensor = 8;
int neigung = 0;
int lautsprecher = 7;
int rot = 5;
int gruen = 3;

void setup() {
	pinMode(neigungssensor, INPUT);
	pinMode(rot, OUTPUT);
	pinMode(gruen, OUTPUT);
}

void loop() {
	neigung = digitalRead(neigungssensor);
	if(neigung==LOW){
		digitalWrite(gruen, LOW);
		tone(lautsprecher, 444);
		digitalWrite(rot, HIGH);
		delay(200);
		tone(lautsprecher, 666);
		digitalWrite(rot, LOW);
		delay(200);
	}
	else{
		noTone(lautsprecher);
		digitalWrite(gruen, HIGH);
		delay(200);
	}
}
```

11. dritte Station – „It's the survival of the fittest!“

11.1 Lernziele (elektrotechnisch / physikalisch)

- Die Schülerinnen und Schüler wiederholen die Anbindung von LEDs mit Hilfe von Widerständen, sowie die Schaltung von Tastern und des Lautsprechers.
- Die Schülerinnen und Schüler realisieren eine organisierte Schaltung, auch bei einer hohen Anzahl an Bauelementen.

11.2 Lernziele (informatisch)

- Die Schülerinnen und Schüler implementieren einen Zufallszahlgenerator unter Nutzung der Funktionen $random(300)$ und $randomSeed(analogRead(0))$.
- Die Schülerinnen und Schüler erklären anschaulich die Notwendigkeit der $randomSeed$-Funktion.
- Die Schülerinnen und Schüler erkennen elementare Bestandteile des Problemlöseprozesses in der Informatik, sie zerlegen das Projekt in einzelne gut nachvollziehbare Teilbausteine und führen diese abschließend zusammen.
- Die Schülerinnen und Schüler implementieren Funktionen, beziehungsweise Prozeduren, und rufen diese innerhalb des Programms auf.
- Die Schülerinnen und Schüler erkennen, weshalb eine Zeitmessung lediglich mit Hilfe der $delay$-Funktion zu realisieren ist.

11.3 Aufgabenstellung

Der Name ist Programm! – Eine gute Reaktionszeit kann dir in Gefahrensituationen das Leben retten, zum Beispiel bei einer Notbremsung mit deinem Fahrrad, weil ein Fußgänger dich nicht gesehen hat. Du sollst dich allerdings keineswegs in Gefahr begeben, sondern lediglich ein kleines Spiel entwickeln, um die Reaktionszeit zweier Spielerinnen und Spieler zu vergleichen. Schauen wir mal wer der oder die Schnellste ist!

11.4 Zusatzaufgabe

Klasse! Dein Spiel läuft bereits, aber wäre es nicht auch interessant zu erfahren, wie viele Sekunden du bis zum Klicken des Button gebraucht hast? Erweitere dein Projekt daher nun so, dass die Zeit vom Ausschalten der LED bis zum Klicken des Buttons gemessen und ausgegeben wird!

11.5 Aufbau und Inhalte

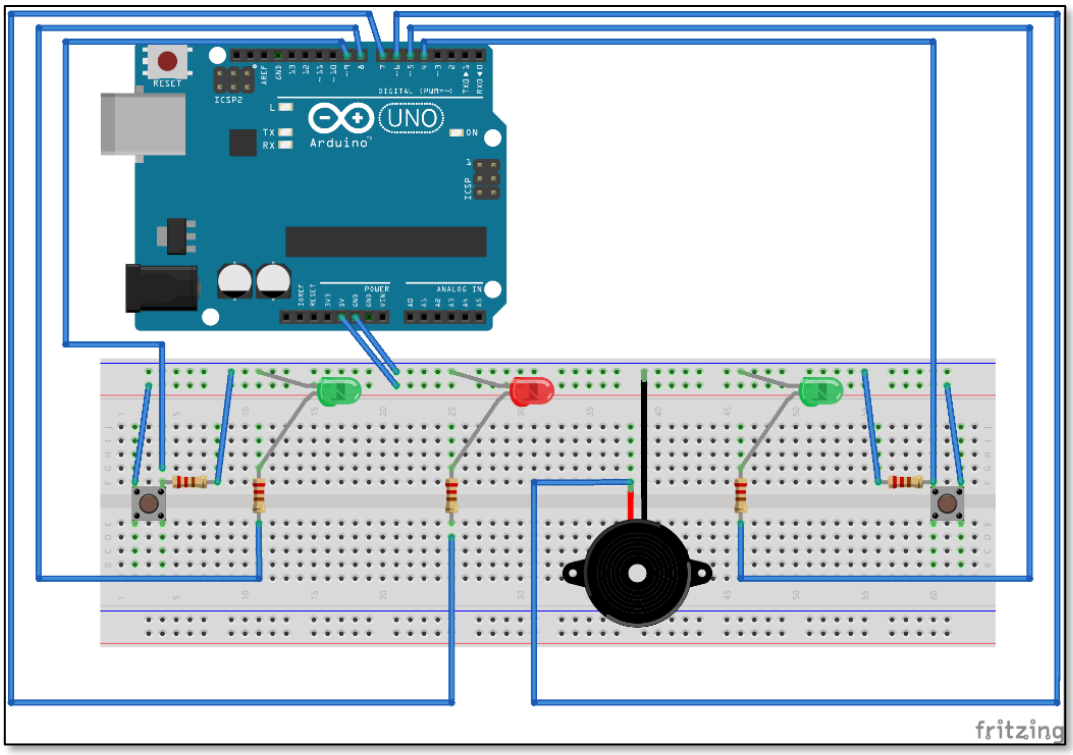
- eine LED soll nach Zeit $t = x$ ausgehen → erste feste Zahl, dann Zufallsfunktion(en): $random(XY)$, $randomSeed(analogRead(XY))$ einführen
- Ausgabe der Zufallszahlen über seriellen Monitor
- LED nach Zeit $t = x$ ausschalten
- **Vorüberlegung**:
    - jeder Spieler / jede Spielerin hat einen Button und eine ihm / ihr zugeordnete grüne LED, weiter soll ein Ton mit einer dem Spieler / der Spielerin zuzuordnenden Tonhöhe ausgegeben werden
    - nur wenn die rote LED ausgegangen ist soll ermittelt werden, welcher Spieler / welche Spielerin zuerst gedrückt hat und die jeweilige LED, sowie der jeweilige Ton angehen
- **Verzweigung** → Was soll beim Drücken des jeweiligen Spielers passieren? → Meldung an den seriellen Monitor, Anschalten der jeweiligen LED, Tonausgabe
- **Probleme:**
    - drückt ein Spieler später, schaltet sich diese LED auch noch an, obwohl das Spiel bereits vorbei ist
    - drückt ein Spieler später, wird die Ausgabe im seriellen Monitor so verändert, dass der spätere Spieler als Gewinner eingetragen wird
    - Was passiert eigentlich, wenn beide Spieler beziehungsweise Spielerinnen gleichzeitig drücken?

- **Problemlösung:**
  - Einfügen einer booleschen Variable, welche überprüft ob bereits ein Spieler oder eine Spielerin gewonnen hat
  - ausgliedern der Ereignisse nach einem Sieg in dazugehörige Funktionen
  - zusätzliche Überprüfung, ob beide Spieler oder Spielerinnen gleichzeitig gedrückt haben
- **Zusatz:**
  - Zeitmessung mit Hilfe einer $delay$-Funktion und einer Wartezeit von $t = 10ms$, sowie der Berechnung $zeit = zeit + 0.01$ (in Sekunden)
  - Ausgabe der Zeit (mit Zuordnung zum Spieler) im seriellen Monitor

11.6 Materialien

Bauteil	Menge
Arduino Uno	$1x$
Breadboard	$1x$
Taster	$2x$
Lautsprecher	$1x$
Kabel	$13x$
Widerstand (220 Ω)	$5x$
LED grün	$2x$
LED rot	$1x$

11.7 Aufbau der Schaltung



11.8 Programmcode

11.8.1 Initialisierung

```
int roteled = 6;
int zufallszahl;
float zeit = 0; //Gleitkommazahl zur Zeitmessung

int buttoneins; //Status des Buttons von Spieler 1
int buttonzwei; //Status des Buttons von Spieler 2

int ledspielereins = 5; //Pin der LED für Spieler 1
int ledspielerzwei = 8; //Pin der LED für Spieler 2

int spielereins = 4; //Pin des Buttons von Spieler 1
int spielerzwei = 9; //Pin des Buttons von Spieler 2

int lautsprecher = 7; //Lautsprecherpin
boolean sieg = false;
```

11.8.2 *setup*-Funktion

```
void setup() {
  Serial.begin(9600); //startet seriellen Monitor
  pinMode(spielereins, INPUT); //legt PIN als INPUT fest
  pinMode(spielerzwei, INPUT); //legt PIN als INPUT fest
  pinMode(ledspielereins, OUTPUT); //legt LED als Output fest
  pinMode(ledspielerzwei, OUTPUT); //legt LED als Output fest
  pinMode(roteled, OUTPUT); //legt LED als Output fest

  digitalWrite(roteled, HIGH); //schaltet LED an
  randomSeed(analogRead(0)); //initialisiert Zufallsgenerator
  zufallszahl = random(200,10000);
      //erzeugen einer Zufallszahl zwischen 200 und 10000
  Serial.print("Zeitdauer bis zum Ausschalten der LED: ");
      //Ausgabe der Zufallszahl im seriellen Monitor
  Serial.print(zufallszahl);
  Serial.println(" Millisekunden.");
  delay(zufallszahl); //Warte den Wert der Zufallszahl in ms
  digitalWrite(roteled, LOW); //schalte LED aus
}
```

11.8.3 Prozedur für den Sieg von Spieler 1

```
void spielereinssieg(){ //Prozedur für den Fall das Spieler 1 gewinnt
  sieg = true; //setzt Wahrheitsvariable auf wahr
  digitalWrite(ledspielereins, HIGH); //schaltet Sieger-LED an

  for (int zaehler=1; zaehler<4; zaehler = zaehler+1){ //Tonfolge
    tone(lautsprecher, 200);
    delay(200);
    tone(lautsprecher, 600);
    delay(200);
  }

  noTone(lautsprecher); //Ausschalten des Lautsprechers
  Serial.print("Sieg für Spieler 1! Zeitdauer: ");
      //Ausgabe des Siegers und der Zeitdauer bis zum Drücken des Buttons
  Serial.print(zeit);
  Serial.println(" Sekunden.");
}
```

11.8.4 Prozedur für den Sieg von Spieler 2

```
void spielerzweisieg(){ //Prozedur für den Fall das Spieler 2 gewinnt
  sieg = true; //setzt Wahrheitsvariable auf wahr
  digitalWrite(ledspielerzwei, HIGH); //schaltet Sieger-LED an

  for (int zaehler=1; zaehler<4; zaehler = zaehler+1){ //Tonfolge
    tone(lautsprecher, 600);
    delay(200);
    tone(lautsprecher, 200);
    delay(200);
  }

  noTone(lautsprecher); //Ausschalten des Lautsprechers
  Serial.print("Sieg für Spieler 2! Zeitdauer: ");
      //Ausgabe des Siegers und der Zeitdauer bis zum Drücken des Buttons
  Serial.print(zeit);
  Serial.println(" Sekunden.");
}
```

11.8.5 Prozedur für den Sieg beider Spieler

```
void beidesieg(){ //Prozedur für den Fall das Spieler 1 und Spieler 1 gewinnt
  sieg = true; //setzt Wahrheitsvariable auf wahr
  digitalWrite(ledspielereins, HIGH); //schaltet Sieger-LED an
  digitalWrite(ledspielerzwei, HIGH); //schaltet Sieger-LED an

  for (int zaehler=1; zaehler<4; zaehler = zaehler+1){ //Tonfolge
    tone(lautsprecher, 400);
    delay(200);
    tone(lautsprecher, 400);
    delay(200);
  }

  noTone(lautsprecher); //Ausschalten des Lautsprechers
  Serial.print("Sieg für beide Spieler! Zeitdauer: ");
      //Ausgabe des Siegers und der Zeitdauer bis zum Drücken des Buttons
  Serial.print(zeit);
  Serial.println(" Sekunden.");
}
```

11.8.6 *loop*-Funktion

```
void loop() {
  zeit = zeit + 0.01; //Addieren von 10 Millisekunden
  delay(10); //10 Millisekunden warten
  buttoneins = digitalRead(spielereins); //Einlesen des Buttonstatus
  buttonzwei = digitalRead(spielerzwei); //Einlesen des Buttonstatus
  if (buttoneins == HIGH && buttonzwei == HIGH && sieg==false) {beidesieg(); }
      //wenn Button gedrückt und sieg Wahrheitsvariable noch auf falsch ist,
        dann rufe Prozedur auf
  if (buttoneins == HIGH && sieg==false) {spielereinssieg(); }
  if (buttonzwei == HIGH && sieg==false) {spielerzweisieg(); }
}
```

12. vierte Station – „1+1=10“

12.1 Lernziele (elektrotechnisch / physikalisch)

- Die Schülerinnen und Schüler realisieren eine Schaltung, bei der unter Nutzung der äußeren Anschlussleisten die Pins mit HIGH ($5\,V$) oder LOW ($0\,V$) belegt werden.
- Die Schülerinnen und Schüler schließen mit Hilfe einer beschrifteten Skizze den LCD-Bildschirm am Arduino an.

12.2 Lernziele (informatisch)

- Die Schülerinnen und Schüler kennen das Dualsystem und erklären die Umwandlung ins Dezimalsystem, sowie dessen Umkehrung.
- Die Schülerinnen und Schüler erläutern anhand eines Algorithmus im Pseudocode die Umwandlung einer Dualzahl in eine Dezimalzahl.
- Die Schülerinnen und Schüler nutzen den Algorithmus zur Umwandlung einer Dualzahl in eine Dezimalzahlen und führen diese für eine vorgegebene Zahl durch.
- Die Schülerinnen und Schüler erkennen an einem Beispiel den Unterschied zwischen den Datentypen int und $double$ und stellen fest, dass eine falsche Wahl bei der Exponentenfunktion problematisch wird.
- Die Schülerinnen und Schüler implementieren eine Zählschleife zum Auslesen der Eingangswerte für die jeweiligen Pins, welche stellvertretend für $1 = High$ und $0 = LOW$ stehen.
- Die Schülerinnen und Schüler lernen das Einbinden und die Nutzung von Bibliotheken kennen, um Informationen auch auf einem LCD-Screen ausgeben zu können.
- Die Schülerinnen und Schüler nutzen die Eigenschaft der setup-Methode zur einmaligen Ausführung des Programmcode, sowie des Rest-Taster des Arduinos zur wiederholten Ausführung.

12.3 Aufgabenstellung

Zugegeben – du weißt, genauso wie wir, dass $1 + 1$ nicht 10 sind. Allerdings ist der Computer vor dir ganz anderer Meinung. Das Ganze nennt sich Binärzahlen – ein Zahlensystem wo es nur Nullen und Einsen gibt. Zunächst verschaffst du dir erstmal einen Überblick wie man Binärzahlen in Dezimalzahlen umwandelt und setzt anschließend ein Programm mit dem Arduino um, was dies für eine vierstellige Binärzahl selbstständig erledigen soll.

12.4 Zusatzaufgabe

Sehr gut – für eine vierstellige Dualzahl funktioniert dein Projekt bereits. Für ein alltagstaugliches Projekt ist es natürlich nicht so toll andauernd den seriellen Monitor nutzen zu müssen, verändere deinen Aufbau daher nun so, dass die Ausgabe über ein LCD-Bildschirm funktioniert.

12.5 Aufbau und Inhalte

- Binärsystem beziehungsweise Dualsystem erläutern (Das Wort „Code“ dual, im ASCII-Code, codieren für einen kleinen WOW-Effekt)
- Beispiel für Dezimal zu Dual
- Beispiel für Dual zu Dezimal
- Algorithmus für Dual zu Dezimal in Worten (Pseudocode)
- Arbeit mit PINS an denen HIGH oder LOW anliegt → $HIGH = 1, LOW = 0$
- Festlegen vom MSB und LSB
- Einführung in $pow(Basis, Exponent)$ - Funktion
- **Diskussion hinsichtlich Auswahl von Datentypen → $7.99\overline{9}$ Problem:** Die Nutzung der Funktion $pow(Basis, Exponent)$ liefert ein Ergebnis des Typs $double$, wenn man dieses Ergebnis in ein Variable des Typs int speichern, so werden die Komma-stellen weggelassen → $2^3 = pow(2,3) = (int)\ 7.99999 = 7$

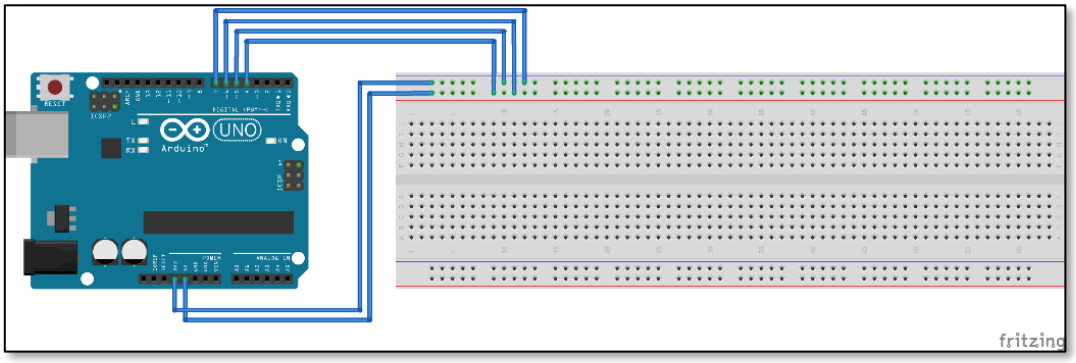
- Einführung in die **Zählschleife** $for\ (Startwert;\ Bedingung;\ Fortsetzung)$
- Auslesen aller Pins und Ausgabe von HIGH und LOW im seriellen Monitor
- Ergänzung der Potenzfunktion und aufaddieren der Zahlenwerte für die Dezimalzahl → Ausgabe im seriellen Monitor
- Einbinden der Bibliothek des LCD-Screens → Testen mit einer Zeichenkette auf einer Zeile und einer Zeichenkette auf zwei verschiedenen Zahlen
- Ausgabe der Eingabe in der ersten Zeile (Binärzahlen)
- Ausgabe des umgerechneten Wertes in der zweiten Zeile (Dezimal)

12.6 Materialien

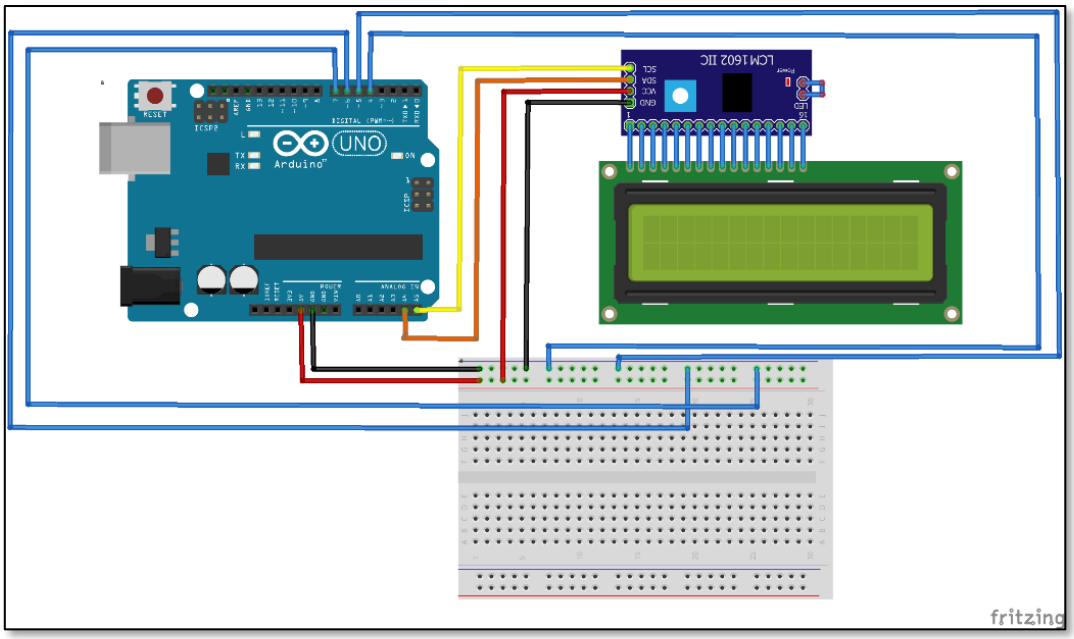
Bauteil	Menge
Arduino Uno	$1x$
Breadboard	$1x$
Kabel	$10x$
LCD-Bildschirm	$1x$

12.7 Aufbau der Schaltung

12.7.1 ohne LCD-Bildschirm



12.7.2 mit LCD-Bildschirm



12.8 Programmcode

```
#include <LiquidCrystal_I2C.h> //bindet LCD-Screen ein
LiquidCrystal_I2C lcd(0x27, 16, 2);

int bin;
float dez = 0;
String eingabe= "";

void setup() {
  lcd.begin(); //startet LCD-Screen
  lcd.clear(); //Startkonfiguration für den seriellen Monitor
  Serial.begin(9600);

  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);

  for (int i = 4; i <= 7; i++) {
      bin = digitalRead(i);
      eingabe = digitalRead(i) + eingabe;
      if (bin != 0) {
          dez = dez + pow(2, i - 4);
      }
  }
```

```
  lcd.setCursor(0,0);
  lcd.print("Eingabe: " + eingabe);
  lcd.setCursor(0,1);
  lcd.print("Ausgabe: ");
  lcd.setCursor(9,1);
  lcd.print(String(dez));

  Serial.print(eingabe);
  Serial.print(" = ");
  Serial.println(dez);
}

void loop() { }
```

13. fünfte Station – „Halt Stop – jetzt fahr ich!“

13.1 Lernziele (elektrotechnisch / physikalisch)

- Die Schülerinnen und Schüler wiederholen die Anbindung von LEDs mit Hilfe von Widerständen, sowie die Schaltung von Tastern und des Lautsprechers.
- Die Schülerinnen und Schüler realisieren eine organisierte Schaltung, auch bei einer hohen Anzahl von Bauelementen.

13.2 Lernziele (informatisch)

- Die Schülerinnen und Schüler abstrahieren die alltägliche Erfahrung beim Kontakt mit Ampeln und rekonstruieren die einzelnen Ampelphasen.
- Die Schülerinnen und Schüler setzen die entwickelte Schaltung für die drei LEDs der normalen „Auto“-Ampel strukturiert, das heißt dass die Pins und die LEDs einer logischen Reihenfolge unterliegen, um.
- Die Schülerinnen und Schüler diskutieren die Umsetzung einer tastergesteuerten Fußgängerampel und beschreiben qualitativ den Zustand der LEDs vor und nach dem Drücken des Tasters.
- Die Schülerinnen und Schüler setzen die tastergesteuerte Fußgängerampel mithilfe einer einfachen Verzweigung um.

13.3 Aufgabenstellung

Ampeln begegnen dir im Alltag ständig. Und zwar egal ob du im Bus, im Auto, mit dem Fahrrad oder auch zu Fuß unterwegs bist. Je nach Größe der Kreuzung kann so eine Ampelschaltung sehr schnell sehr kompliziert werden, eine einfache Ampel kannst du allerdings ganz einfach selbst realisieren.

Überlege zunächst einmal, welche verschiedenen Ampelphasen es überhaupt gibt und zeichne diese in die Vorlage ein. Realisiere anschließend eine einfache Ampel, welche alle Phasen abwechselnd zeigt.

13.4 Zusatzaufgabe

Deine Ampel funktioniert und die Autos, sowie Fahrräder, können nun sicher die Straße passieren? – Sehr gut! Allerdings fehlt noch etwas Wichtiges, denn wir haben noch nicht an die Fußgänger gedacht. Füge zunächst einen Taster zu deiner Schaltung hinzu und überlege, was passiert, wenn dieser gedrückt wird. Setze deine Überlegungen nun im Programmcode um und teste sie!

13.5 Aufbau und Inhalte

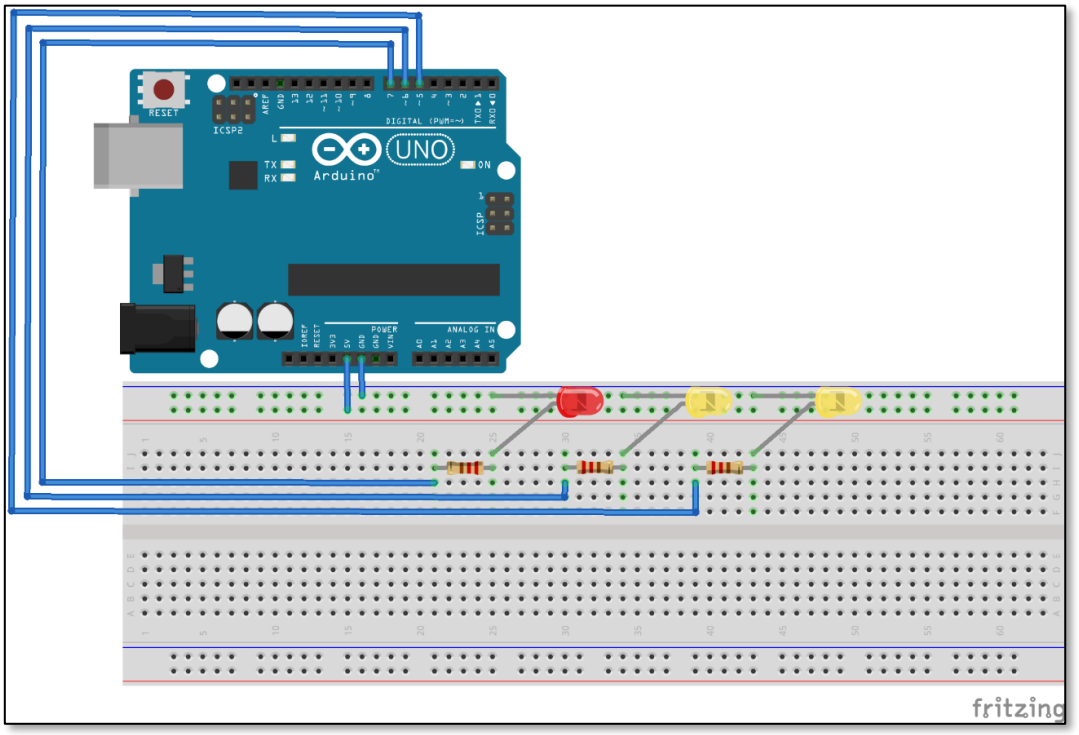
- unter Nutzung einer Vorlage erfolgt die Rekonstruktion der Ampelphasen durch die Schülerinnen und Schüler
- die Schülerinnen und Schüler benennen die Reihenfolge der jeweiligen Phasen und diskutieren die Wahl sinnvoller Pausen zwischen den Wechseln (Grün- und Rotphasen deutlich länger als die Zwischenphasen)
- **schrittweise Realisierung der Schaltung**:
    - zunächst eine LED → über Taster ein- und ausschalten, sowie mithilfe einer $delay$-Funktion steuern, testen
    - hinzufügen zweier weiterer LEDs → alle Ein- und Ausschalten
    - Realisierung der einzelnen Ampelphasen
    - wünschenswert: die Grün- und Rot-Phasen sollten selbstverständlich deutlich länger als die anderen sein
- **Realisierung der Fußgängerampel:**
    - hinzufügen eines Tasters
    - was passiert, wenn der Taster gedrückt wurde? → Wechsel zur Rot-Phase, Fußgängerampel grün
    - wenn Taster nicht gedrückt wurde → dauerhaft grün für Autos und Fußgänger, dauerhaft rot für Fußgänger
    - Umsetzung mit einer einfachen Verzweigung, welche Überprüft ob der Button gedrückt wurde

13.6 Materialien

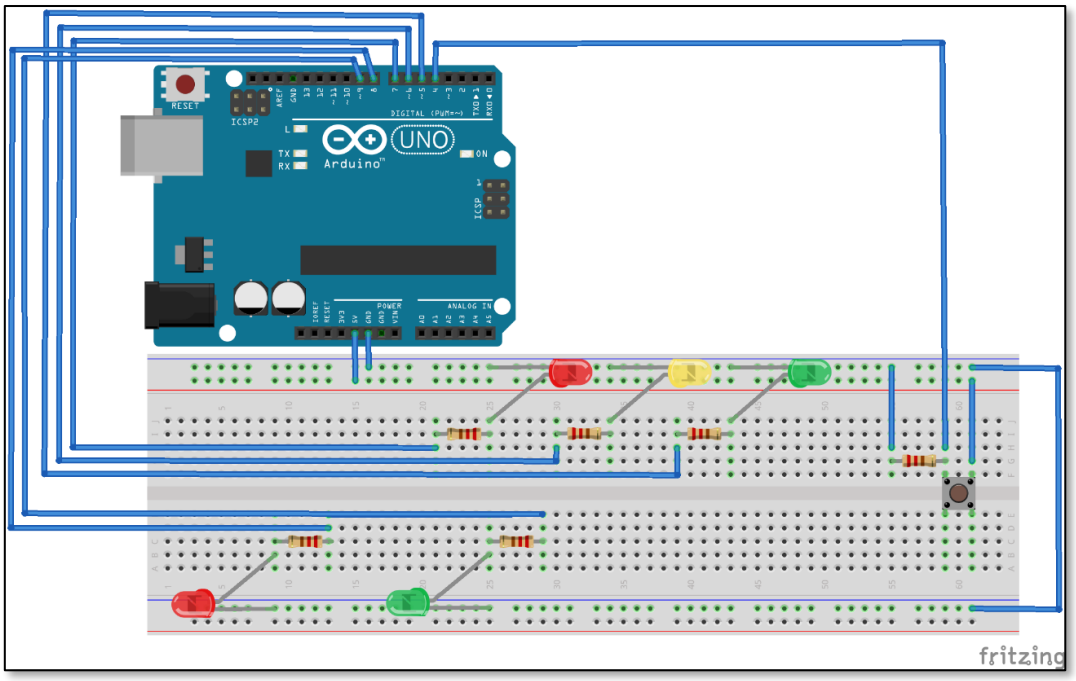
Bauteil	Menge
Arduino Uno	$1x$
Breadboard	$1x$
Widerstand (220 Ω)	$6x$
LED (rot, gelb, grün)	$5x$
Kabel	$12x$
Taster	$1x$

13.7 Aufbau der Schaltung

13.7.1 Teil 1 – „Auto“-Ampel



13.7.2 Teil 2 - komplette Ampelschaltung



13.8 Programmcode

```
int roteLed = 5;
int gelbeLed = 6;
int grueneLed = 7;
int fussgaengerroteLed = 8;
int fussgaengergrueneLed = 9;
int button = 4;

void setup() {
  pinMode(roteLed, OUTPUT);
  pinMode(gelbeLed, OUTPUT);
  pinMode(grueneLed, OUTPUT);
  pinMode(fussgaengerroteLed, OUTPUT);
  pinMode(fussgaengergrueneLed, OUTPUT);
  pinMode(button, INPUT);
}

void loop() {
  digitalWrite(roteLed, HIGH);
  digitalWrite(gelbeLed, LOW);
  digitalWrite(grueneLed, LOW);

  digitalWrite(fussgaengerroteLed, LOW);
  digitalWrite(fussgaengergrueneLed, HIGH);
```

```
  if (digitalRead(button) == HIGH) {
      //kurzes Warten vor dem Umschalten

      delay(2000);

      //Ampel rot, wenn Autos fahren

      digitalWrite(fussgaengerroteLed, HIGH);
      digitalWrite(fussgaengergrueneLed, LOW);

      // Ampelphase 1 & 5 - nur rot

      digitalWrite(roteLed, HIGH);
      digitalWrite(gelbeLed, LOW);
      digitalWrite(grueneLed, LOW);

      delay(15000);

      // Ampelphase 2 - rot und gelb

      digitalWrite(roteLed, HIGH);
      digitalWrite(gelbeLed, HIGH);
      digitalWrite(grueneLed, LOW);

      delay(2000);
      // Ampelphase 3 - nur grün

      digitalWrite(roteLed, LOW);
      digitalWrite(gelbeLed, LOW);
      digitalWrite(grueneLed, HIGH);

      delay(10000);

      // Ampelphase 4 - nur gelb

      digitalWrite(roteLed, LOW);
      digitalWrite(gelbeLed, HIGH);
      digitalWrite(grueneLed, LOW);

      delay(2000);
  }
}
```

14. sechste Station – „Du siehst rot? – Ich sehe nur (255,0,0)!“

14.1 Lernziele (elektrotechnisch / physikalisch)

- Die Schülerinnen und Schüler schließen die RGB-LED mit Hilfe einer Belegungsskizze an und ordnen die jeweiligen Pins den einzelnen Farben (rot, grün, blau) zu.
- Die Schülerinnen und Schüler verbinden den Farbsensor mit dem Arduino und ordnen den Pins die jeweiligen Funktionen zu.

14.2 Lernziele (informatisch)

- Die Schülerinnen und Schüler erlangen ein Überblickswissen über den RGB-Farbraum und können den Kombinationen von Zahlentupel verschiedenen Farben zuordnen.
- Die Schülerinnen und Schüler nutzen das Grundlagenwissen über den RGB-Farbraum zur Ansteuerung einer RGB-LED.
- Die Schülerinnen und Schüler kennen die Funktionsweise des RGB-Farbsensors und implementieren den notwendigen Code, um die Informationen des Farbsensors im seriellen Monitor darzustellen.
- Die Schülerinnen und Schüler verknüpfen die Kenntnisse über die RGB-LED, sowie über den Farbsensor, und realisieren eine Schaltung, welche die Farbe eines eingescannten Blattes mithilfe der RGB-LED sichtbar macht.

oder

Die Schülerinnen und Schüler lernen die Steuerung des Servomotors kennen und bestimmen die Winkel, um die Farben mittels Pfeil auf einer Schablone darzustellen, mit Hilfe der vom Farbsensor ausgegeben Werte.

14.3 Aufgabenstellung

Was wäre die Welt nur ohne Farben? Was für viele von uns eine ziemlich triste Vorstellung wäre, ist für (Farben-)Blinde leider alltägliche Realität. Selbstverständlich könnte man jetzt annehmen, dass es bereits Computerchips gibt, welche diese Fähigkeit ersetzen. Leider ist die Wissenschaft in dieser Hinsicht noch nicht soweit. Du aber darfst dir heute bereits einmal Gedanken machen, wie ein Computer überhaupt Farben erkennen kann.

14.3.1 vertiefende Aufgabenstellung für Variante 1

Deine erste Aufgabe wird es sein, eine RGB-LED zum Leben zu erwecken. Dazu lernst du zunächst was RGB überhaupt bedeutet, um damit dann verschiedene Farben mit nur einer LED darzustellen. In einem zweiten Schritt setzt du dich mit einem Farbsensor auseinander, um verschieden farbige Blätter einzuscannen und die Farbe durch die LED darstellen zulassen.

14.3.2 vertiefende Aufgabenstellung für Variante 2

Deine erste Aufgabe wird es sein, eine RGB-LED zum Leben zu erwecken. Dazu lernst du zunächst was RGB überhaupt bedeutet, um damit dann verschiedene Farben mit nur einer LED darzustellen.

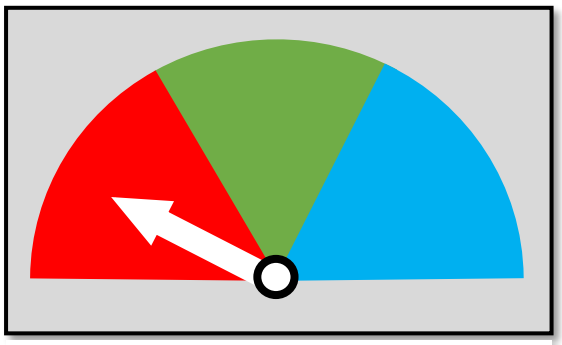


Abbildung 5: Vorlage Farbschablone

In einem zweiten Schritt setzt du dich mit einem Farbsensor auseinander, um verschieden farbige Blätter einzuscannen. Durch Ansteuerung eines Servomotors und der in Abbildung 5 dargestellten Schablone sollen die eingescannten Farben mithilfe des Pfeils dargestellt werden.

14.4 Zusatzaufgabe

Die Farben rot, grün und blau werden bereits richtig eingescannt? Sehr gut! Erweitere nun dein Projekt so, dass auch die Farben orange und gelb erkannt werden können!

14.5 Aufbau und Inhalte

- RGB-Farbmodell und additive Farbmischung
- Zuordnung von RGB-Tupel und Farbe für die wichtigsten sechs Kombinationen: rot (255,0,0), grün (0,255,0), blau (0,0,255), gelb (100,255,0), türkis (0,255,255), pink (125,0,125)
- Einführung in die Funktionsweise und Belegung der RGB-LED
- Austesten der RGB-LED für die Grundfarben (zyklisch abwechselnd mit Pausen dazwischen)
- Funktionsweise und Belegung des Farbsensor
- Kennenlernen der $pulseIn()$-Funktion
- Ausgabe der Sensorwerte im seriellen Monitor:
    - eingelesene Werte darstellen und dem jeweiligen Farbsensor zuordnen
    - Auswahl der gescannten Farbe durch Vergleich aller Werte → höchster Wert entspricht der Farbe

14.5.1 spezifisch für Variante 1

- Verknüpfung der Ausgabewerte des Sensor mit der RGB-LED
- Testen der Ausgabe mit ausgedruckten farbigen Blättern
- Erweiterung um zwei weitere Farben (gelb und orange)

14.5.2 spezifisch für Variante 2

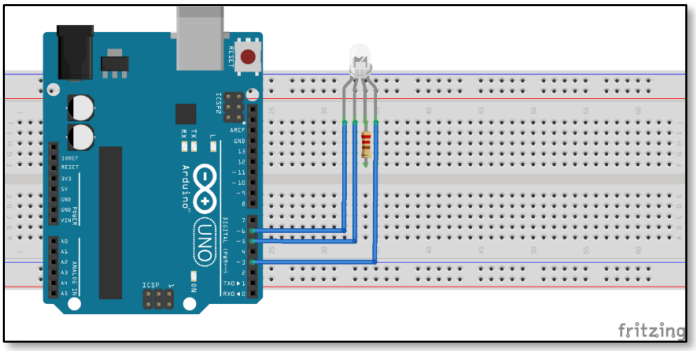
- Realisierung des in Abbildung 5 dargestellten Aufbaus
- Einbinden des Servomotors (unter Nutzung der dazugehörigen Library) → Testen der Steuerung für die Winkel 10°, 45° und 80°
- Zuordnung der Winkel zu den dazugehörigen Farben auf der Schablone und Verknüpfung mit den Ausgabewerten des Farbsensors

14.6 Materialien

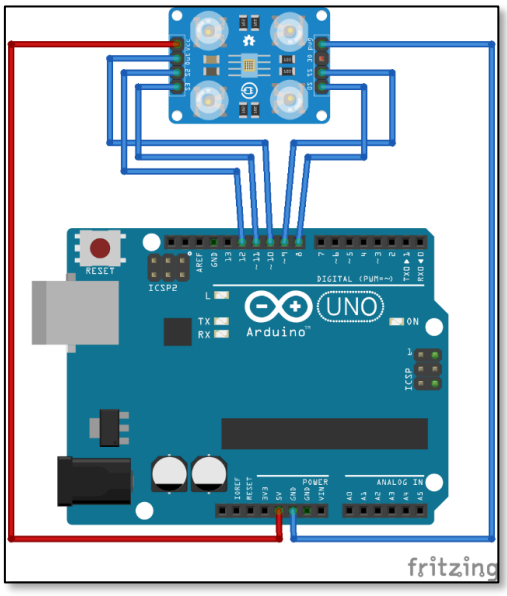
Bauteil	Menge
Arduino Uno	$1x$
Breadboard	$1x$
RGB-LED	$1x$
Kabel	$13x$
Widerstand (220 Ω)	$1x$
Farbsensor TCS230	$1x$

14.7 Aufbau der Schaltung

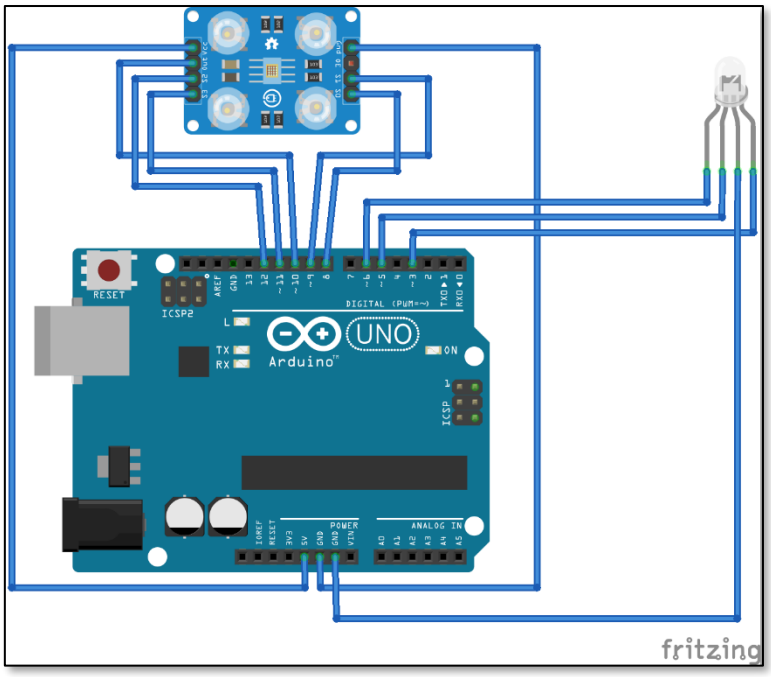
14.7.1 Teil 1 – RGB-LED



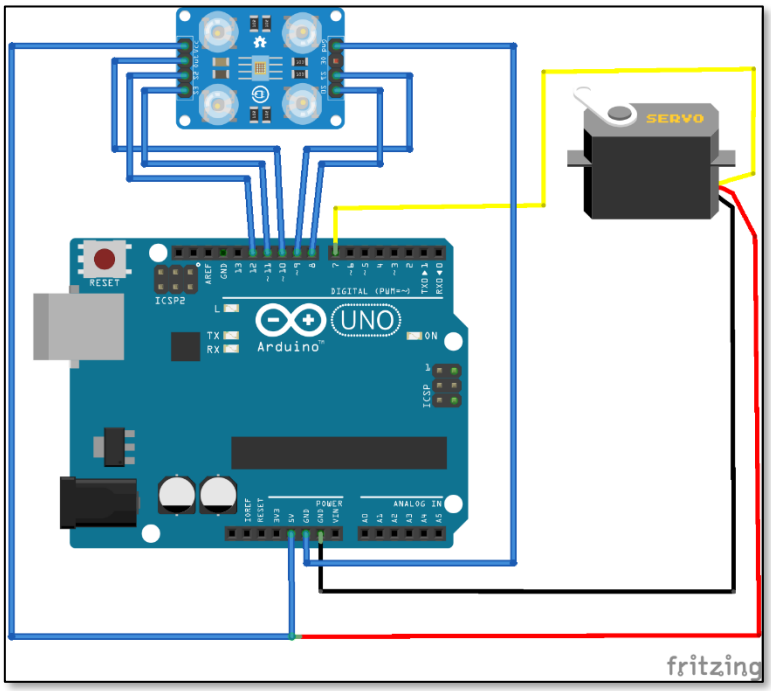
14.7.2 Teil 2 - Farbsensor



14.7.3 Teil 3a – Farbsensor mit RGB-LED



14.7.4 Teil 3b – Farbsensor mit Servomotor



14.8 Programmcode

14.8.1 Teil 1– RGB-LED

```
int LEDblau = 3; // Farbe blau an Pin 3
int LEDrot = 6; // Farbe rot an Pin 5
int LEDgruen = 5; // Farbe gruen an Pin 6

void setup() {
  pinMode(LEDblau, OUTPUT);
  pinMode(LEDgruen, OUTPUT);
  pinMode(LEDrot, OUTPUT);
}

void rgbfarben(int rot, int gruen, int blau){
  analogWrite(LEDrot, rot);
  analogWrite(LEDgruen, gruen);
  analogWrite(LEDblau, blau);
}

void loop() {
  rgbfarben(100,255,0); //gelb
  delay(3000);

  rgbfarben(0,255,255); //türkis
  delay(3000);
  rgbfarben(125,0,125); //lila / pink
  delay(3000);

  rgbfarben(255,0,0); //rot
  delay(3000);

  rgbfarben(0,0,255); //blau
  delay(3000);

  rgbfarben(0,255,0); //grün
  delay(3000);
}
```

14.8.2 Teil 2 – Farbsensor

```
const int s0 = 8;
const int s1 = 9;
const int s2 = 12;
const int s3 = 11;
const int out = 10;
int rot = 0;
int gruen = 0;
int blau = 0;
```

```
void setup(){
  Serial.begin(9600);
  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);
  pinMode(s3, OUTPUT);
  pinMode(out, INPUT);
  digitalWrite(s0, HIGH);
  digitalWrite(s1, HIGH);
}

void loop(){
  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  rot = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  digitalWrite(s3, HIGH);
  blau = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  digitalWrite(s2, HIGH);
  gruen = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);

  Serial.print(" Wert Rot: ");
  Serial.print(rot);
  Serial.print(" Wert Gruen: ");
  Serial.print(gruen);
  Serial.print(" Wert Blau: ");
  Serial.print(blau);




  if (rot < blau && rot < gruen){
    Serial.println(" - (Rote Farbe)");
    delay(1000);
  }
  else if (blau < rot && blau < gruen){
    Serial.println(" - (Blaue Farbe)");
    delay(1000);
  }
  else if (gruen < rot && gruen < blau){
    Serial.println(" - (Gruene Farbe)");
    delay(1000); // pause
  }
  else{
    Serial.println();
  }
}
```

14.8.3 Teil 3a – Farbsensor mit RGB-LED

```
const int s0 = 8;
const int s1 = 9;
const int s2 = 12;
const int s3 = 11;
const int out = 10;
int rot = 0;
int gruen = 0;
int blau = 0;

int LEDblau = 3;
int LEDrot = 6;
int LEDgruen = 5;

void rgbfarben(int rot, int gruen, int blau){
  analogWrite(LEDrot, rot);
  analogWrite(LEDgruen, gruen);
  analogWrite(LEDblau, blau);
}

void setup(){
  Serial.begin(9600);
  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);
  pinMode(s3, OUTPUT);
  pinMode(out, INPUT);
  digitalWrite(s0, HIGH);
  digitalWrite(s1, HIGH);
}

void loop(){
  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  rot = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  digitalWrite(s3, HIGH);
  blau = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  digitalWrite(s2, HIGH);
  gruen = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);

  Serial.print(" Wert Rot: ");
  Serial.print(rot);
  Serial.print(" Wert Gruen: ");
  Serial.print(gruen);
  Serial.print(" Wert Blau: ");
  Serial.print(blau);
```

```
  if (rot < blau && rot < gruen){
    Serial.println(" - (Rote Farbe)");
    rgbfarben(255,0,0);
    delay(1000);
  }
  else if (blau < rot && blau < gruen){
    Serial.println(" - (Blaue Farbe)");
    rgbfarben(0,0,255);
    delay(1000);
  }
  else if (gruen < rot && gruen < blau){
    Serial.println(" - (Gruene Farbe)");
    rgbfarben(0,255,0);
    delay(1000);
  }
  else{
    Serial.println();
  }
  delay(3000);
  rgbfarben(0,0,0);
}
```

14.8.4 Teil 3b – Farbsensor mit Servomotor

```
#include <Servo.h>

Servo meinServo;

const int s0 = 8;
const int s1 = 9;
const int s2 = 12;
const int s3 = 11;
const int out = 10;
int rot = 0;
int gruen = 0;
int blau = 0;

void setup(){
  Serial.begin(9600);
  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);
  pinMode(s3, OUTPUT);
  pinMode(out, INPUT);
  digitalWrite(s0, HIGH);
  digitalWrite(s1, HIGH);

  meinServo.attach(7);
}
```

```
void loop(){
  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  rot = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  digitalWrite(s3, HIGH);
  blau = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  digitalWrite(s2, HIGH);
  gruen = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);

  Serial.print(" Wert Rot: ");
  Serial.print(rot);
  Serial.print(" Wert Gruen: ");
  Serial.print(gruen);
  Serial.print(" Wert Blau: ");
  Serial.print(blau);

  if (rot < blau && rot < gruen){
    Serial.println(" - (Rote Farbe)");
    meinServo.write(150);
    delay(1000);
  }
  else if (blau < rot && blau < gruen){
    Serial.println(" - (Blaue Farbe)");
    meinServo.write(30);
    delay(1000);
  }
  else if (gruen < rot && gruen < blau){
    Serial.println(" - (Gruene Farbe)");
    meinServo.write(90);
    delay(1000);
  }
  else{
    Serial.println();
  }
  delay(3000);

  meinServo.write(0);
}
```

15. siebte Station – „smartCar - smartParking“

15.1 Lernziele (elektrotechnisch / physikalisch)

- Die Schülerinnen und Schüler gewinnen einen Überblick über Sensoren mit dem Zweck der Entfernungsmessung, in diesem konkreten Fall handelt es sich hierbei um einen Infrarot- und um einen Ultraschallsensor.
- Die Schülerinnen und Schüler vergleichen die technischen Spezifikationen zweier Sensoren miteinander und treffen eine begründete Entscheidung für einen Sensor, um einen Rückfahrsensor umzusetzen.

15.2 Lernziele (informatisch)

- Die Schülerinnen und Schüler implementieren den notwendigen Programmcode, um die Daten des Infrarot- oder Ultraschallsensors auszulesen und geben die aufgenommen Werte im seriellen Monitor aus.
- Die Schülerinnen und Schüler erkennen, dass die Daten des Ultraschallsensors nicht mit einem realen Maßeinheitswert in Verbindung zu bringen sind. Aus diesem Grund erhalten die Schülerinnen und Schüler Einblick in die Umrechnung und setzten diese im Programmcode um.
- Die Schülerinnen und Schüler erkennen, dass die Daten des Infrarotsensors nicht mit einem realen Maßeinheitswert in Verbindung zu bringen sind. Aus diesem Grund setzen die Lernenden eine Kalibrierung um, damit Sensorwerte in Maßeinheiten umgerechnet werden können.
- Die Schülerinnen und Schüler bestimmen Schwellwerte für optische und akustische Warneinrichtungen, wie es bei einem richtigen Rückfahrsensor der Fall ist.

15.3 Aufgabenstellung

Rund 40 % aller Verkehrsunfälle passieren beim Parken und Rangieren. Viele neue Autos haben mittlerweile diverse Assistenzsysteme verbaut, allerdings gibt es immer noch genügend ohne. Grund genug, sich einmal das technische Prinzip hinter einem Rückfahrsensor anzuschauen.

Um eine solche Warneinrichtung zu bauen, hast du zwei verschiedene Sensoren zur Verfügung. Schau dir zunächst beide Sensoren genauer an und entscheide dich für einen für beiden, um das Projekt umzusetzen.

15.4 Zusatzaufgabe

Okay, deine Ausgabe am Computer funktioniert bereits. Schauen wir mal wie wir dein Projekt noch besser machen können! Stell dir einmal vor, wir wollen diese Konstruktion in einem realen Fahrzeug verbauen. In diesem Fall kann der Fahrer selbstverständlich nicht die ganze Zeit über auf einen kleinen Monitor schauen. Deshalb wäre es sinnvoll, deinem Projekt eine akustische und optische Warnung hinzuzufügen.

15.5 Aufbau und Inhalte

- Vorstellung und Vergleich des Ultraschall- und des Infrarotsensors

15.5.1 Umsetzung mit dem Infrarotsensor

- Aufbau und Anschluss des Infrarotsensors
- Messwerte mithilfe des seriellen Plotters darstellen
- **Erkenntnis**: Hyperbel, nicht linear, nicht „einfach“ berechenbar
- **Lösung**: Kalibrierung durch Abtastung von Breakpoints, durchnummerieren der Punkte, Ausgabe des Sensorwertes, mit dazugehöriger Nummer im seriellen Monitor → Auftragung in einem Diagramm → es handelt sich um eine Kurve
- **ABER:** Sensorwerte schwanken stark → Berechnung des Mittelwertes
- Ergänzung eines Lautsprechers, dessen Frequenz und Abstand der Töne durch den Abstand gesteuert werden

- Hinzufügen einer RGB-LED, welche in Abhängigkeit des Abstandes grün, gelb oder rot leuchtet beziehungsweise blinkt
- Vorgabe der Berechnungsformel für den Abstand in Zentimetern:

$$abstand = \left(\frac{3027.4}{Sensorwert}\right)^{1.2134}$$

- Vereinfachung der ehemaligen Breakpoints mit berechnetem Wert

15.5.2 Umsetzung mit dem Ultraschallsensor

- Anschlüsse des Ultraschallsensor
- kurzer Funktionsabriss, über das Aussenden einer Schallwelle und dessen Reflexion, Ausgabe einer Zeitdifferenz zwischen Aussenden und Rückkehr der Schallwelle
- Anschluss des Sensors und Ausgabe der Zeitwerte im seriellen Monitor
- Berechnung der Strecke über:

$$v = \frac{s}{t}$$

Dabei ist zu bedenken, dass die Zeit in Mikrosekunden gemessen wird. Um Dimensionstechnisch auf die Einheit Zentimeter zu kommen, muss die Schallgeschwindigkeit $343.2\frac{m}{s}$ in die Einheit $\frac{cm}{\mu s}$ umgerechnet werden. Damit ergibt sich folgende Berechnung für die Strecke:

$$s = v * t = 0.03432\frac{cm}{\mu s} * <\frac{Zeit}{2}> \mu s = < Strecke > cm$$

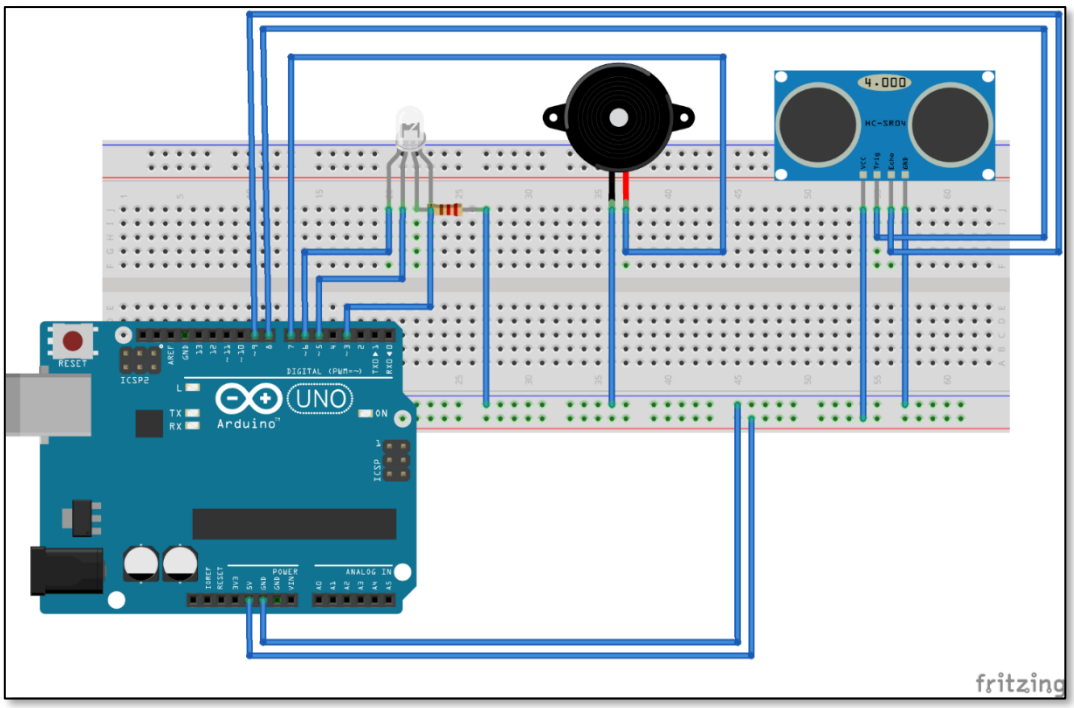
- Ausgabe der Zeit, der Berechnung und der Strecke im seriellen Monitor → Austesten der Zuverlässigkeit der Werte durch Verschieben eines Hindernisses und nachmessen mithilfe eines Gliedermaßstabes
- Ergänzung eines Lautsprechers, dessen Frequenz und Abstand der Töne durch den Abstand gesteuert werden
- hinzufügen einer RGB-LED, welche in Abhängigkeit des Abstandes grün, gelb oder rot leuchtet beziehungsweise blinkt

15.6 Materialien

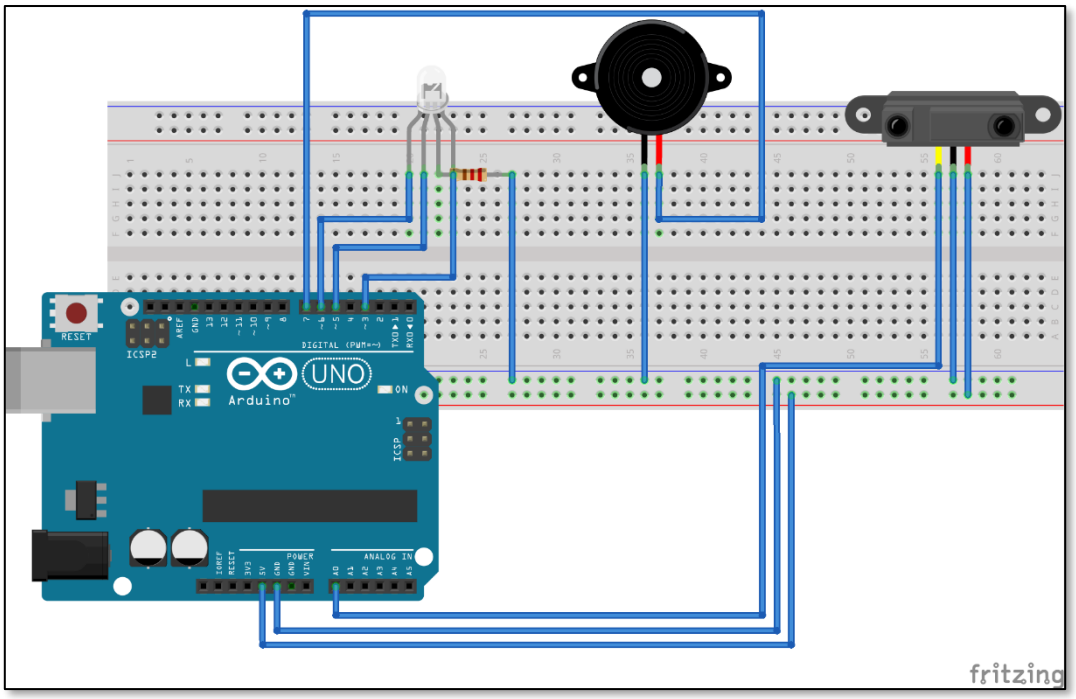
Bauteil	Menge
Arduino Uno	$1x$
Breadboard	$1x$
RGB-LED	$1x$
Kabel	$10x$
Widerstand (220 Ω)	$1x$
Lautsprecher	$1x$
Ultraschallsensor (HC-SR04)	$1x$
Infrarotsensor (Sharp 2Y0A21)	$1x$

15.7 Aufbau der Schaltung

15.7.1 Teil 1 – mit Ultraschallsensor



15.7.2 Teil 2 – mit Infrarotsensor



15.8 Programmcode

15.8.1 Teil 1 – mit Ultraschallsensor

```
int trigger=9;
int echo=8;

long dauer=0;
long entfernung=0;
int pause;
int frequenz;

int LEDblau = 3;
int LEDrot = 6;
int LEDgruen = 5;

void rgbfarben(int rot, int gruen, int blau){
  analogWrite(LEDrot, rot);
  analogWrite(LEDgruen, gruen);
  analogWrite(LEDblau, blau);
}
```

```
void setup(){
  Serial.begin (9600);
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);

  pinMode(LEDblau, OUTPUT);
  pinMode(LEDgruen, OUTPUT);
  pinMode(LEDrot, OUTPUT);
}

void loop(){
  noTone(7);
  digitalWrite(trigger, LOW);
  delay(5);
  digitalWrite(trigger, HIGH);
  delay(10);
  digitalWrite(trigger, LOW);
  dauer = pulseIn(echo, HIGH);
  entfernung = (dauer/2) * 0.03432;

  if (entfernung >= 500 || entfernung <= 0) {
    Serial.println("Kein Messwert");
  }

  else if(entfernung >= 150){
    pause = 500;
    frequenz = 200;
    rgbfarben(0,0,0);
  }

  else if(entfernung < 150 && entfernung >= 50){
    pause = 300;
    frequenz = 400;
    rgbfarben(0,255,0);
  }

  else if(entfernung < 50 && entfernung >= 10){
    pause = 100;
    frequenz = 600;
    rgbfarben(100,255,0);
  }


  else if(entfernung < 10 && entfernung > 0){
    pause = 50;
    frequenz = 800;
    rgbfarben(255,0,0);
  }
```

```
  tone(7,frequenz);
  Serial.print("aktuelle Entfernung zum Ultraschallsensor: ");
  Serial.print(entfernung);
  Serial.println(" cm");
  delay(pause);
  rgbfarben(0,0,0);
}
```

15.8.2 Teil 2 – mit Infrarotsensor (händische Kalibrierung)

```
int Infrarotsensor = A0;

int abstandsWert1;
int abstandsWert2;
int abstandsWert3;
int abstandsWert4;
int abstandsWert5;

int durchschnittAbstandsWert;

int LEDblau = 3;
int LEDrot = 6;
int LEDgruen = 5;

int speacker = 7;
int pause;
int frequenz;

void setup(){
  Serial.begin(9600);
}



void rgbfarben(int rot, int gruen, int blau){
  analogWrite(LEDrot, rot);
  analogWrite(LEDgruen, gruen);
  analogWrite(LEDblau, blau);
}

void loop(){
  abstandsWert1=analogRead(Infrarotsensor);
  delay(20);
  abstandsWert2=analogRead(Infrarotsensor);
  delay(20);
  abstandsWert3=analogRead(Infrarotsensor);
  delay(20);
  abstandsWert4=analogRead(Infrarotsensor);
  delay(20);
  abstandsWert5=analogRead(Infrarotsensor);
```

```
  durchschnittAbstandsWert = (abstandsWert1+abstandsWert2+abstandsWert3+
  abstandsWert4+abstandsWert5)/5;

  Serial.print("Sensorwert: ");
  Serial.println(durchschnittAbstandsWert);

  //Werte des Abstandssensors:
  //10 cm = 478
  //20 cm = 254
  //30 cm = 179
  //40 cm = 143
  //50 cm = 120
  //60 cm = 102
  //70 cm = 91
  //80 cm = 81

  if(durchschnittAbstandsWert <= 60){
    pause = 500;
    frequenz = 200;
    rgbfarben(0,0,0);
  }

  else if(durchschnittAbstandsWert > 60 && durchschnittAbstandsWert <= 120){
    pause = 300;
    frequenz = 400;
    rgbfarben(0,255,0);
  }

  else if(durchschnittAbstandsWert > 120 && durchschnittAbstandsWert <= 254){
    pause = 100;
    frequenz = 600;
    rgbfarben(100,255,0);
  }

  else if(durchschnittAbstandsWert > 254){
    pause = 50;
    frequenz = 800;
    rgbfarben(255,0,0);
  }

  tone(speacker,frequenz);
  delay(pause);
  noTone(speacker);
  rgbfarben(0,0,0);
}
```

15.8.3 Teil 2- mit Infrarotsensor (berechnete Werte)

```
int Infrarotsensor = A0;

int abstandsWert1;
int abstandsWert2;
int abstandsWert3;
int abstandsWert4;
int abstandsWert5;

int durchschnittAbstandsWert;

int abstandBerechnet;

int LEDblau = 3;
int LEDrot = 6;
int LEDgruen = 5;

int speacker = 7;
int pause;
int frequenz;

void setup(){
  Serial.begin(9600);
}

void rgbfarben(int rot, int gruen, int blau){
  analogWrite(LEDrot, rot);
  analogWrite(LEDgruen, gruen);
  analogWrite(LEDblau, blau);
}

void loop(){
  abstandsWert1=analogRead(Infrarotsensor);
  delay(20);
  abstandsWert2=analogRead(Infrarotsensor);
  delay(20);
  abstandsWert3=analogRead(Infrarotsensor);
  delay(20);
  abstandsWert4=analogRead(Infrarotsensor);
  delay(20);
  abstandsWert5=analogRead(Infrarotsensor);

  durchschnittAbstandsWert =(abstandsWert1+abstandsWert2+abstandsWert3+
  abstandsWert4+abstandsWert5)/5;

  abstandBerechnet = pow((3027.4/durchschnittAbstandsWert), 1.2134);

  Serial.print("Sensorwert: ");
  Serial.println(durchschnittAbstandsWert);
```

```
  if(abstandBerechnet >= 100){
    pause = 500;
    frequenz = 200;
    rgbfarben(0,0,0);
  }

  else if(abstandBerechnet < 100 && abstandBerechnet >= 60){
    pause = 300;
    frequenz = 400;
    rgbfarben(0,255,0);
  }

  else if(abstandBerechnet < 60 && abstandBerechnet >= 20){
    pause = 100;
    frequenz = 600;
    rgbfarben(100,255,0);
  }

  else if(abstandBerechnet < 20){
    pause = 50;
    frequenz = 800;
    rgbfarben(255,0,0);
  }

  tone(speacker,frequenz);
  delay(pause);
  noTone(speacker);
  rgbfarben(0,0,0);
}
```

16. achte Station – „Viva la Wetter“

16.1 Lernziele (elektrotechnisch / physikalisch)

- Die Schülerinnen und Schüler schließen die zur Verfügung stehenden Sensoren gemäß einer bebilderten Pinbelegung korrekt an.
- Die Schülerinnen und Schüler erkennen, dass bei Nutzung des seriellen Busses, alle Sensoren in Reihe an die gleichen analogen Pins angeschlossen werden müssen.

16.2 Lernziele (informatisch)

- Die Schülerinnen und Schüler verstehen den grundsätzlichen Aufbau einer Bibliothek.
- Die Schülerinnen und Schüler binden Bibliotheken zunächst innerhalb der Arduino IDE ein, sowie anschließend im Programmcode.
- Die Schülerinnen und Schüler nutzen die durch die Bibliothek zur Verfügung gestellten Funktionen.
- Die Schülerinnen und Schüler erkennen die Funktionsweise des seriellen Datenbusses und begründen die Notwendigkeit einer eindeutigen Adressierung der Bauelemente.
- Die Schülerinnen und Schüler können am Beispiel nachvollziehen, dass mittels Adressangaben auch Operationen mit den vom Sensor bereitgestellten Daten ausgeführt werden können.

16.3 Aufgabenstellung

Wenn man sich im Internet über das aktuelle Wetter erkundigt, erhält man von jeder Website verschiedene Angaben, da überall andere Temperaturen und Vorhersagen stehen. Es wird also Zeit, eine eigene kleine Wetterstation zu bauen, welche die Temperatur, den Luftdruck und die Luftfeuchtigkeit erfasst. Um die Benutzerfreundlichkeit zu verbessern, soll die Ausgabe am Ende über den LCD-Bildschirm realisiert werden.

16.4 Zusatzaufgabe

Wenn du es bis hierhin geschafft hast, ist es eine leichte Sache deine Wetterstation um weitere zu messende Aspekte zu ergänzen. In diesem Fall sollst du zusätzlich einen Sensor zur Messung der Lichtintensität einbinden. Der Sensorwert soll mit der Einheit Lux ebenfalls mit LCD-Bildschirm ausgegeben werden.

16.5 Aufbau und Inhalte

- Einbinden des kombinierten Temperatur- und Luftfeuchtigkeitssensors und der dazugehörigen $< dht.h >$ Bibliothek
- Ausgabe der Temperatur und der relativen Luftfeuchtigkeit im seriellen Monitor
- Ergänzung um eine Umrechnung von Grad Celsius in Grad Fahrenheit
- Erklärung serieller Datenbus, spezifischer der I²C- Bus (Bei mehreren Geräten Reihenschaltung, eindeutige Adressierung der Sensoren in Form einer Hexadezimalzahl) → Verwendung mit Hilfe der Bibliothek $< wire.h >$
- **Die Bibliothek $< wire.h >$:**
    - $Wire.beginTransmission(Adresse);$
    - $Wire.write(Daten);$
    - $Wire.endTransmission();$
    - $Wire.beginTransmission(Adresse);$
    - $Wire.requestFrom(Adresse, X);$
    - $incoming = Wire.read();\ Wire.endTransmission();$
- Einbinden der Bibliotheken $< Adafruit\_Sensor.h >$ und $< Adafruit\_BMP280.h >$
- Auslesen des Sensorwertes mit $luftdruck = bme.readPressure()/100$
- Einbinden der Bibliothek für den LCD-Bildschirm, Instanziieren eines Objektes für den LCD-Bildschirm mit den Parametern $lcd(0x27, 16, 2);$, wobei $0x27$ die hexadezimale Adresse ist, die 16 für die Zeichen pro Zeile steht und die 2 die Anzahl der Zeilen repräsentiert

- **Kennenlernen der folgenden Funktionen:**
  - $lcd.clear();$
  - $lcd.setCursor(0,0);$
  - $lcd.print("Hallo\ Welt!");$
  - $lcd.begin();$
- Definition der Adressen für das Auslesen des Lichtsensors, mit Hilfe des Befehlssatzes der $< wire.h >$ Bibliothek die Daten übertragen, Nutzung des Befehlssatzes gemäß des Datenblattes des Sensors:

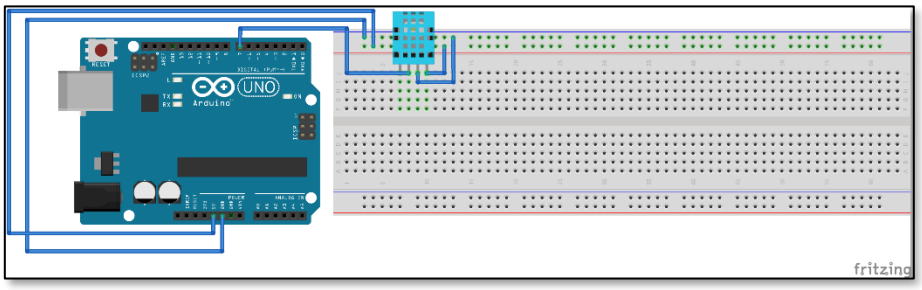
ADDRESS	RESISTER NAME	R/W	REGISTER FUNCTION	RESET VALUE
—	COMMAND	W	Specifies register address	0x00
0x00	CONTROL	R/W	Power on/off and single cycle	0x00
0x01	CONFIG	R/W	Powersave Enable / Integration Time	0x00
0x04	DATALOW	R	ALS Data LOW Register	0x00
0x05	DATAHIGH	R	ALS Data HIGH Register	0x00
0x0A	ID	R	Device ID	ID

16.6 Materialien

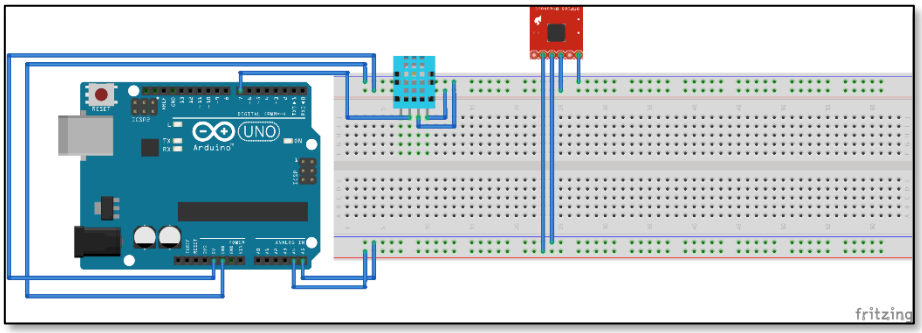
Bauteil	Menge
Arduino Uno	$1x$
Breadboard	$1x$
LCD-Bildschirm	$1x$
Kabel	$19x$
DHT11-Sensor für Luftfeuchtigkeit und Temperatur	$1x$
Luftdrucksensor BMP280	$1x$
Lichtsensor TSL 45315	$1x$

16.7 Aufbau der Schaltung

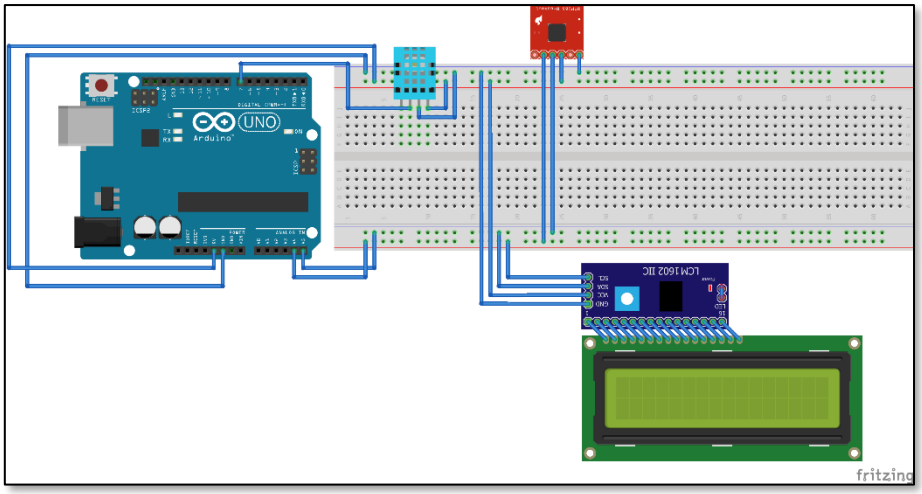
16.7.1 Teil 1 – DHT11-Sensor



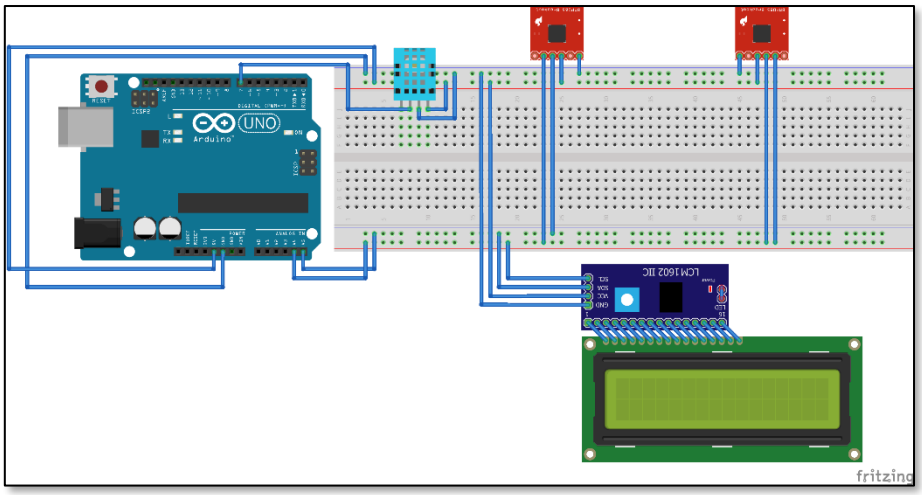
16.7.2 Teil 2 – Ergänzung um Luftdrucksensor



16.7.3 Teil 2 – Ergänzung um einen LCD-Bildschirm



16.7.4 Zeil 4 – Zusatzaufgabe – Hinzufügen eines Lichtsensors



17. Programmcode

17.1 Teil 1 – DTH11-Sensor

```
#include <dht.h>
dht DHT;
#define DHT11_PIN 7

void setup(){
  Serial.begin(9600);
}

void loop(){
  DHT.read11(DHT11_PIN);

  //Temperatur in Grad Celsius
  float temperaturCelsius = DHT.temperature;
  Serial.print("Temperatur = ");
  Serial.print((float) temperaturCelsius);
  Serial.println(" °C");

  //Temperatur in Grad Fahrenheit
  Serial.print("Temperatur = ");
  float temperaturFahrenheit = (DHT.temperature * 1.8)+ 32;
  Serial.print((float)temperaturFahrenheit);
  Serial.println(" °F");

  //relative Luftfeuchtigkeit
  Serial.print("Humidity = ");
  Serial.println(DHT.humidity);
  delay(1000);
}
```

17.2 Teil 2 – Ergänzung um Luftdrucksensor

```
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>

//einbinden der Bibliothek für den seriellen Datenbus
#include <Wire.h>

//Biblitohek für den DHT11 Sensor
#include <dht.h>
dht DHT;
#define DHT11_PIN 7

//Definitionen für den BMP280-Sensor
#define BMP_SCK 13
#define BMP_MISO 12
#define BMP_MOSI 11
#define BMP_CS 10
Adafruit_BMP280 bme;

void setup(){
  Serial.begin(9600);
  // starten des seriellen Datenbussen
  Wire.begin();
  // Beginn der Datenübertragung, mit Adresse des Barometers (wenn Masse auf GND)
  Wire.beginTransmission(0x76);
  // wenn Sensor keine Daten überträgt - Zeige eine Fehlermeldung
  if (!bme.begin()) {
    Serial.println("Überprüfe die Datenbusverbindung, es scheint ein Fehler
vorzuliegen.");
    while(1);
  }
}

void loop(){
  // Auslesen des DHT11-Sensors
  DHT.read11(DHT11_PIN);

  // Temperatur in Grad Celsius
  float temperaturCelsius = DHT.temperature;
  Serial.print("Temperatur = ");
  Serial.print((float) temperaturCelsius);
  Serial.println(" °C");

  Serial.println(" ");

  // Temperatur in Grad Fahrenheit
  Serial.print("Temperatur = ");
  float temperaturFahrenheit = (DHT.temperature * 1.8)+ 32;
  Serial.print((float)temperaturFahrenheit);
  Serial.println(" °F");

  Serial.println(" ");

  // relative Luftfeuchtigkeit in Prozent
  Serial.print("Humidity = ");
  Serial.print(DHT.humidity);
  Serial.println(" %");
```

```
  Serial.println(" ");

  // Luftdruck in hPa
  float luftdruck = bme.readPressure()/100;
  Serial.print("Luftdruck = ");
  Serial.print(luftdruck);
  Serial.println(" hPa");

  Serial.println("-------------------------");

  delay(5000);
}
```

17.3 Teil 3 – Ergänzung um einen LCD-Bildschirm

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
//Hier wird festgelegt um was für einen Display es sich handelt
//In diesem Fall eines mit 16 Zeichen in 2 Zeilen und der HEX-Adresse 0x27

#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>

//einbinden der Bibliothek für den seriellen Datenbus
#include <Wire.h>

//Biblitohek für den DHT11 Sensor
#include <dht.h>
dht DHT;
#define DHT11_PIN 7

//Definitionen für den BMP280-Sensor
#define BMP_SCK 13
#define BMP_MISO 12
#define BMP_MOSI 11
#define BMP_CS 10
Adafruit_BMP280 bme;

void setup(){
  lcd.begin(); //Im Setup wird der LCD gestartet
  lcd.backlight(); //Hintergrundbeleuchtung einschalten
  Serial.begin(9600);
  // starten des seriellen Datenbussen
  Wire.begin();
  // Beginn der Datenübertragung, mit Adresse des Barometers (wenn Masse auf GND)
  Wire.beginTransmission(0x76);
  // wenn Sensor keine Daten überträgt - Zeige eine Fehlermeldung
  if (!bme.begin()) {
    Serial.println("Überprüfe die Datenbusverbindung, es scheint ein Fehler
vorzuliegen.");
    while(1);
  }
}
```

```
void loop(){
  // Auslesen des DHT11-Sensors
  DHT.read11(DHT11_PIN);

  // Temperatur in Grad Celsius
  float temperaturCelsius = DHT.temperature;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temperatur:");
  lcd.setCursor(11, 0);
  lcd.print(temperaturCelsius);
  lcd.setCursor(0, 1);
  lcd.print("Grad Celsius");

  delay(2000);

  // Temperatur in Grad Fahrenheit
  float temperaturFahrenheit = (DHT.temperature * 1.8)+ 32;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temperatur: ");
  lcd.setCursor(11, 0);
  lcd.print(temperaturFahrenheit);
  lcd.setCursor(0, 1);
  lcd.print("Grad Fahrenheit");

  delay(2000);

  Serial.println(" ");

  // relative Luftfeuchtigkeit in Prozent
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Luftfeuchte:");
  lcd.setCursor(0, 1);
  lcd.print(DHT.humidity);
  lcd.setCursor(6, 1);
  lcd.print("Prozent");

  delay(2000);

  // Luftdruck in hPa
  float luftdruck = bme.readPressure()/100;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Luftdruck:");
  lcd.setCursor(0, 1);
  lcd.print(luftdruck);
  lcd.setCursor(8, 1);
  lcd.print("hPa");

  delay(2000);
}
```

17.4 Teil 4 – Zusatzaufgabe – Hinzufügen eines Lichtsensors

```
//Lichtsensor
#define I2C_ADDR      0x29
#define REG_CONTROL   0x00
#define REG_CONFIG    0x01
#define REG_DATALOW   0x04
#define REG_DATAHIGH  0x05
#define REG_ID        0x0A

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); //Hier wird festgelegt um was für einen Display
es sich handelt. In diesem Fall eines mit 16 Zeichen in 2 Zeilen und der HEX-
Adresse 0x27. Für ein vierzeiliges I2C-LCD verwendet man den Code
"LiquidCrystal_I2C lcd(0x27, 20, 4)"

#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>

//einbinden der Bibliothek für den seriellen Datenbus
#include <Wire.h>

//Biblitohek für den DHT11 Sensor
#include <dht.h>
dht DHT;
#define DHT11_PIN 7

//Definitionen für den BMP280-Sensor
#define BMP_SCK 13
#define BMP_MISO 12
#define BMP_MOSI 11
#define BMP_CS 10
Adafruit_BMP280 bme;

void setup(){
  lcd.begin(); //Im Setup wird der LCD gestartet
  lcd.backlight(); //Hintergrundbeleuchtung einschalten

  Serial.begin(9600);

  // starten des seriellen Datenbussen
  Wire.begin();

  // Beginn der Datenübertragung, mit Adresse des Barometers (wenn Masse auf GND)
  Wire.beginTransmission(0x76);

  // wenn Sensor keine Daten überträgt - Zeige eine Fehlermeldung
  if (!bme.begin()) {
      Serial.println("Überprüfe die Datenbusverbindung, es scheint ein Fehler
      vorzuliegen.");
    while(1);
  }
```

```
  //Lichtsensor
  Wire.beginTransmission(I2C_ADDR);
  Wire.write(0x80|REG_ID);
  Wire.endTransmission();
  Wire.requestFrom(I2C_ADDR, 1);
  while(Wire.available()){
    unsigned char c = Wire.read();
  }
  Wire.beginTransmission(I2C_ADDR);
  Wire.write(0x80|REG_CONTROL);
  Wire.write(0x03); //power on
  Wire.endTransmission();

  Wire.beginTransmission(I2C_ADDR);
  Wire.write(0x80|REG_CONFIG);
  Wire.write(0x00);
  Wire.endTransmission();
}

void loop(){
  // Auslesen des DHT11-Sensors
  DHT.read11(DHT11_PIN);

  // Temperatur in Grad Celsius
  float temperaturCelsius = DHT.temperature;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temperatur:");
  lcd.setCursor(11, 0);
  lcd.print(temperaturCelsius);
  lcd.setCursor(0, 1);
  lcd.print("Grad Celsius");

  delay(2000);

  // Temperatur in Grad Fahrenheit
  float temperaturFahrenheit = (DHT.temperature * 1.8)+ 32;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temperatur: ");
  lcd.setCursor(11, 0);
  lcd.print(temperaturFahrenheit);
  lcd.setCursor(0, 1);
  lcd.print("Grad Fahrenheit");

  delay(2000);

  Serial.println(" ");

  // relative Luftfeuchtigkeit in Prozent
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Luftfeuchte:");
  lcd.setCursor(0, 1);
  lcd.print(DHT.humidity);
  lcd.setCursor(6, 1);
  lcd.print("Prozent");

  delay(2000);
```

```
  // Luftdruck in hPa
  float luftdruck = bme.readPressure()/100;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Luftdruck:");
  lcd.setCursor(0, 1);
  lcd.print(luftdruck);
  lcd.setCursor(8, 1);
  lcd.print("hPa");

  delay(2000);

  //Lichtsensor
  uint16_t l, h;
  uint32_t lux;
  Wire.beginTransmission(I2C_ADDR); //Datenübertragung beginnen
  Wire.write(0x80|REG_DATALOW); //schreibe ersten Datensatz
  Wire.endTransmission(); //beende Datenübertragung
  Wire.requestFrom(I2C_ADDR, 2);
  l = Wire.read();
  h = Wire.read();
  while(Wire.available()){
    Wire.read();
  }
  lux  = (h<<8) | (l<<0);
  lux *= 1;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Lichtstaerke:");
  lcd.setCursor(0, 1);
  lcd.print(lux);
  lcd.print(" Lux");

  delay(2000);

}
```

18. neunte Station – „Wieder mal viel Lärm um nichts!“

19. zehnte Station – „Ganz schön dicke Luft hier… „

20. elfte Station – „Da geht mir glatt ein Licht auf!“

20.1 Lernziele (elektrotechnisch / physikalisch)

- Die Schülerinnen und Schüler unterscheiden normale LEDs von Infrarot-LEDs.
- Die Schülerinnen und Schüler kennen die Möglichkeit, eine angeschaltete Infrarot-LED mittel Kamera, zum Beispiel die eines Smartphones, zu identifizieren.
- Die Schülerinnen und Schüler kennen die Bedingung für den Stromfluss bei einer Infrarot Photodiode.
- Die Schülerinnen und Schüler realisieren eine organisierte Schaltung, auch bei einer hohen Anzahl an Bauelementen.

20.2 Lernziele (informatisch)

- Die Schülerinnen und Schüler erkennen die Notwendigkeit der Verwendung des Datentyps $float$ für die Division.
- Die Schülerinnen und Schüler bestimmen die korrekten Bedingungen für die Verzweigen, wie zum Beispiel, dass die Variablen für Start- und Endzeit bereits gesetzt wurden und nicht überschrieben werden sollen.
- Die Schülerinnen und Schüler verwenden ineinander geschachtelte Verzweigungen, um Start- und Endzeit zu bestimmen, sowie um die jeweilige LED (beziehungsweise Laterne) an- oder auszuschalten.

20.3 Aufgabenstellung

Der Klimawandel ist allgegenwärtig und vor allem durch Gruppen wie Fridays for Future in den Mittelpunkt des gesellschaftlichen Diskurs gelangt. Jetzt liegt es an uns Möglichkeiten zu entwickeln, um Energie und damit direkt verbunden auch CO_2 einzusparen. Eine, in der Praxis unter Umständen nicht ganz so praktikable Idee, ist hierbei, Straßenlaternen über Lichtschranken zu steuern und diese jeweils nur anzuschalten wenn auch ein Auto, Fußgänger oder Fahrradfahrer daran vorbeifährt. Dies wirst du an dieser Station mit Infrarot-LEDs und -Dioden realisieren.

20.4 Zusatzaufgabe

Sehr gut – das Licht geht nur noch an, wenn die Lichtschranken unterbrochen werden. Da das Grundsystem einmal steht, wäre dies auch eine praktikable Lösung um eine Geschwindigkeitsvormessung vorzunehmen, dies sollst du nun durch Ergänzung einiger kleiner Berechnungen umsetzen.

20.5 Aufbau und Inhalte

- Erklärung des Prinzips einer Infrarot Lichtschranke (IR-Led und IR-Photodiode)
- Was sind IR-Led und IR-Photodioden? Wie funktionieren? Was macht sie besonders? Was ist beim Anschließen zu beachten?
- **Hilfestellung** für die Schülerinnen und Schüler, sowie für die Betreuenden: Die Beinchen beider Bauteile werden mit Isolierband (rot und blau) gekennzeichnet, um ein falsches Anschließen und vor allem die Fehlersuche zu erleichtern
- Testen der Schaltung mit Hilfe eines Lautsprechers (Ton, wenn unterbrochen)
- **Programmidee**: Es soll nur die jeweilige LED (beziehungsweise „die Laterne“) leuchten, an der das Auto gerade vorbeifährt
- Erweiterung der Schaltung um drei weitere Lichtschranken, sowie drei weitere LEDs
- mehrere einfache Verzweigungen zum Programmieren der Funktionalität
- **Zeitmessung / Geschwindigkeitsmessung:**
    - Zeit zwischen unterbrechen der ersten und der dritten Lichtschranke
    - Ausmessen der Distanz zwischen beiden Lichtschranken
    - Berechnung der Geschwindigkeit über:

$$v = \frac{s}{t}$$

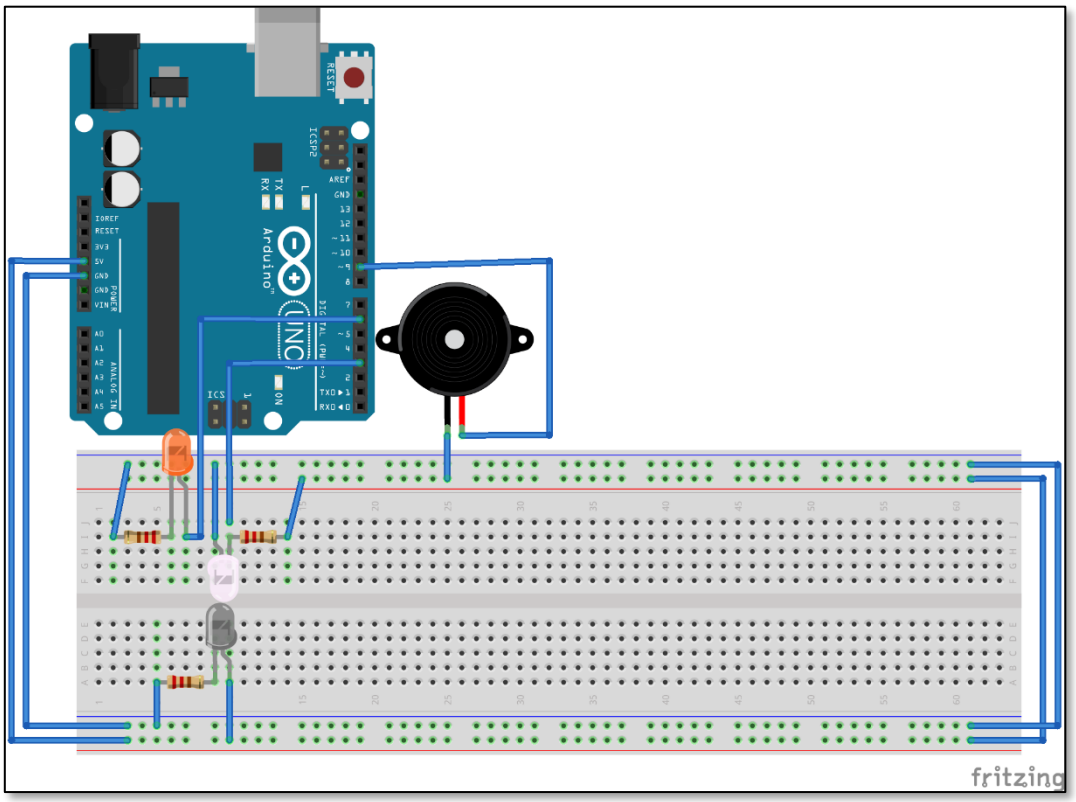
Dabei sind korrekte Einheiten zu beachten. Die Zeit sollte in Sekunden umgerechnet werden und die Distanz in Meter. Die resultierende Einheit $\frac{m}{s}$ ermöglicht einen Vergleich zu anderen bewegten Körpern. Damit kann indirekt die Korrektheit des Messergebnisses überprüft werden.

20.6 Materialien

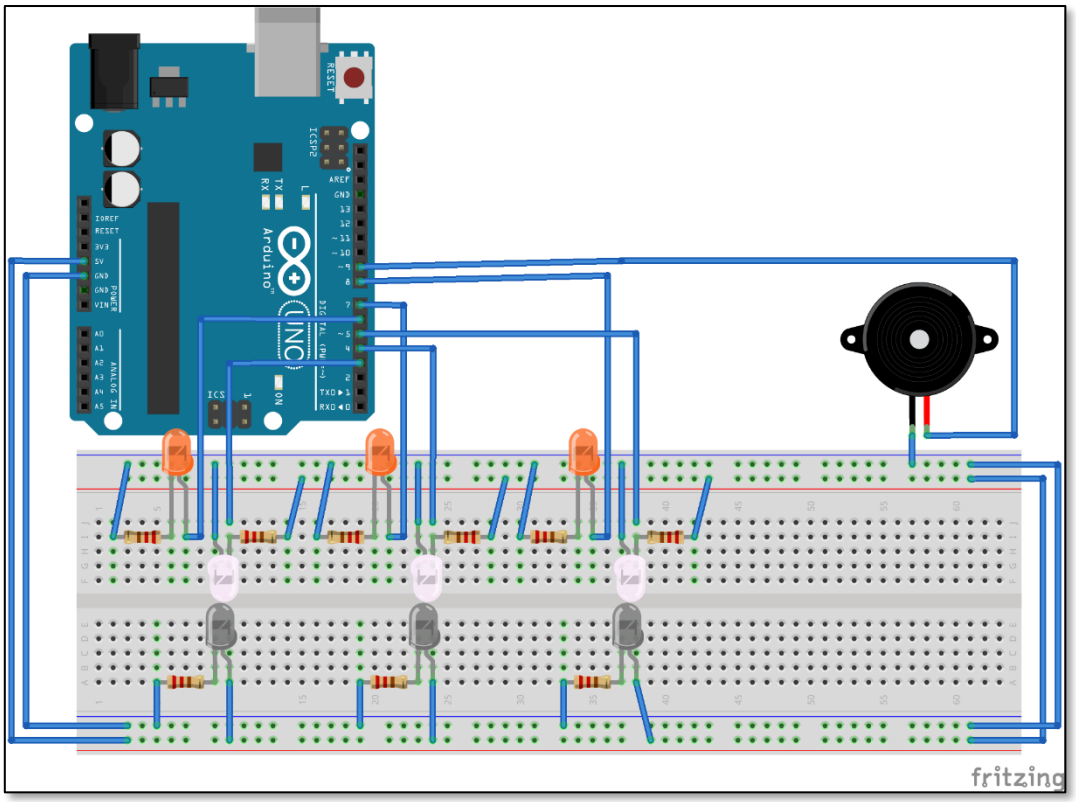
Bauteil	Menge
Arduino Uno	$1x$
Breadboard	$1x$
LED (weiß)	$3x$
Kabel	$ca.\ 15x$
Widerstand (220 Ω)	$6x$
Lautsprecher	$1x$
IR-LED	$3x$
IR Photodiode	$3x$

20.7 Aufbau der Schaltung

20.7.1 Teil 1 – eine Lichtschranke



20.7.2 Teil 2 – drei Lichtschranken



20.8 Programmcode

20.8.1 Teil 1 – eine Lichtschranke

```
int photoDIODEeins = 3;

int speaker = 9;

void setup(){
  Serial.begin(9600);
  pinMode(photoDIODEeins,INPUT);
}

void loop(){
  if(digitalRead(photoDIODEeins)==HIGH){
    tone(speaker,400);
    Serial.println("Lichtschranke unterbrochen! :O");
  }
  else{
    Serial.println("Lichtschranke nicht unterbrochen! :)");
  }
  delay(100);
  noTone(speaker);
}
```

20.8.2 Teil 2 – drei Lichtschranken

```
int photoDIODEeins = 3;
int photoDIODEzwei = 4;
int photoDIODEdrei = 5;
int LEDeins = 6;
int LEDzwei = 7;
int LEDdrei = 8;
int speaker = 9;

void setup(){
  pinMode(photoDIODEeins,INPUT);
  pinMode(photoDIODEzwei,INPUT);
  pinMode(photoDIODEdrei,INPUT);

  pinMode(LEDeins,OUTPUT);
  pinMode(LEDzwei,OUTPUT);
  pinMode(LEDdrei,OUTPUT);
}

void loop(){
  if(digitalRead(photoDIODEeins)==HIGH){
    digitalWrite(LEDeins,HIGH);
    digitalWrite(LEDzwei,LOW);
    digitalWrite(LEDdrei,LOW);
    tone(speaker,400);
  }


  else if(digitalRead(photoDIODEzwei)==HIGH){
    digitalWrite(LEDeins,LOW);
    digitalWrite(LEDzwei,HIGH);
    digitalWrite(LEDdrei,LOW);
    tone(speaker,400);
  }

  else if(digitalRead(photoDIODEdrei)==HIGH){
    digitalWrite(LEDeins,LOW);
    digitalWrite(LEDzwei,LOW);
    digitalWrite(LEDdrei,HIGH);
    tone(speaker,400);
  }
  else{
    digitalWrite(LEDeins,LOW);
    digitalWrite(LEDzwei,LOW);
    digitalWrite(LEDdrei,LOW);
  }

  delay(100);
  noTone(speaker);
```

```
}
```

20.8.3 Teil 3 – Zusatzaufgabe – Geschwindigkeitsberechnung

```
int photoDIODEeins = 3;
int photoDIODEzwei = 4;
int photoDIODEdrei = 5;

int LEDeins = 6;
int LEDzwei = 7;
int LEDdrei = 8;

int startZeit = 0;
int endZeit = 0;
float geschwindigkeit = 0;

int speaker = 9;

void setup(){
  Serial.begin(9600);

  pinMode(photoDIODEeins,INPUT);
  pinMode(photoDIODEzwei,INPUT);
  pinMode(photoDIODEdrei,INPUT);

  pinMode(LEDeins,OUTPUT);
  pinMode(LEDzwei,OUTPUT);
  pinMode(LEDdrei,OUTPUT);
}

void loop(){
  if(digitalRead(photoDIODEeins)==HIGH){
    digitalWrite(LEDeins,HIGH);
    digitalWrite(LEDzwei,LOW);
    digitalWrite(LEDdrei,LOW);

    tone(speaker,400);

    if(startZeit==0){
      startZeit = millis();
      Serial.print("Startzeit: ");
      Serial.println(startZeit);
    }
  }

  else if(digitalRead(photoDIODEzwei)==HIGH){
    digitalWrite(LEDeins,LOW);
    digitalWrite(LEDzwei,HIGH);
    digitalWrite(LEDdrei,LOW);
    tone(speaker,400);
  }
```

```
  else if(digitalRead(photoDIODEdrei)==HIGH && endZeit==0){
    digitalWrite(LEDeins,LOW);
    digitalWrite(LEDzwei,LOW);
    digitalWrite(LEDdrei,HIGH);

    if(endZeit==0){
      endZeit = millis();
      Serial.print("Endzeit: ");
      Serial.println(endZeit);
    }

    tone(speaker,400);

    if(startZeit!=0 && endZeit !=0){
      geschwindigkeit = 0.12 / ((endZeit - startZeit)/1000);
      Serial.print("Geschwindigkeit: ");
      Serial.print(geschwindigkeit);
      Serial.println(" m/s");
    }
  }

  else{
    digitalWrite(LEDeins,LOW);
    digitalWrite(LEDzwei,LOW);
    digitalWrite(LEDdrei,LOW);
  }

  delay(100);
  noTone(speaker);
}
```