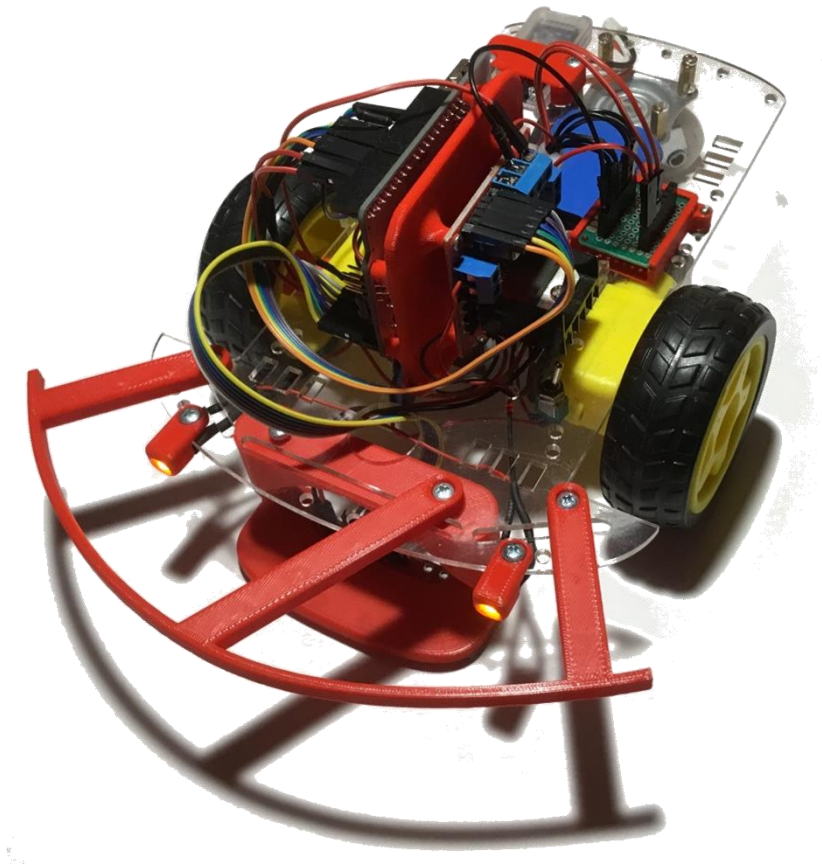
PROJEKT ROBOTERAUTO



In diesem Projekt werden wir ein Roboterauto planen, die dazu benötigten Bauteile mit dem 3D-Drucker erstellen, die Bauteile zusammensetzen und das Auto schlussendlich programmieren.

Lycée Classique de Diekirch

Version 2019
Sven HEYMANS
Thomas SCHEER

Version 2020
Thomas SCHEER

Interessante Internetseiten

https://funduino.de
https://sites.google.com/stonybrook.edu/arduinoble/
https://create.arduino.cc/projecthub/hda-robotics/project-1-2wd-obstacle-avoiding-robot-390ef8
https://www.instructables.com/id/Controlling-servo-motor-using-IR-remote-control/

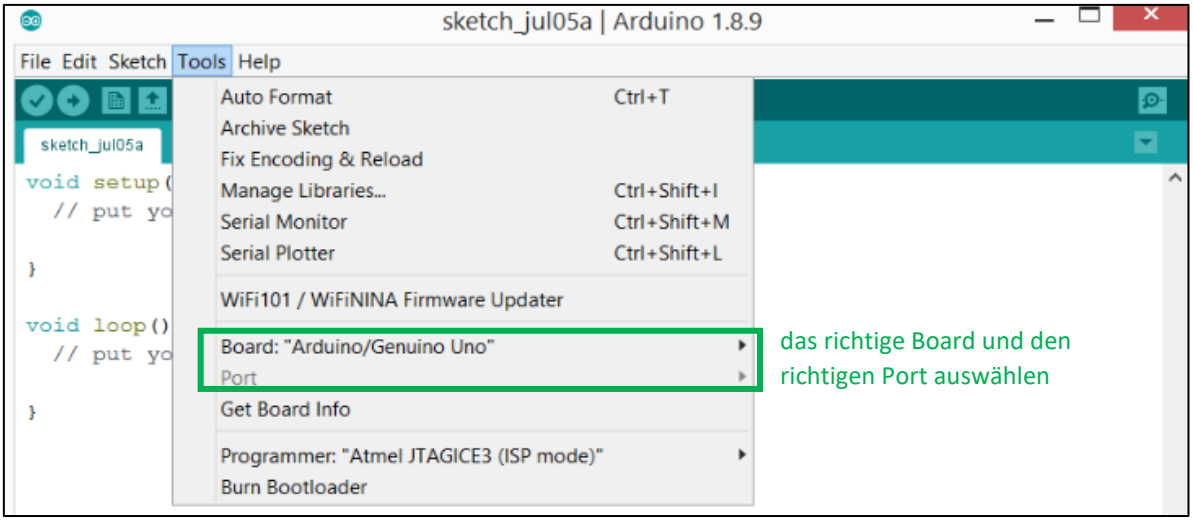
Übersicht der Arbeitsschritte

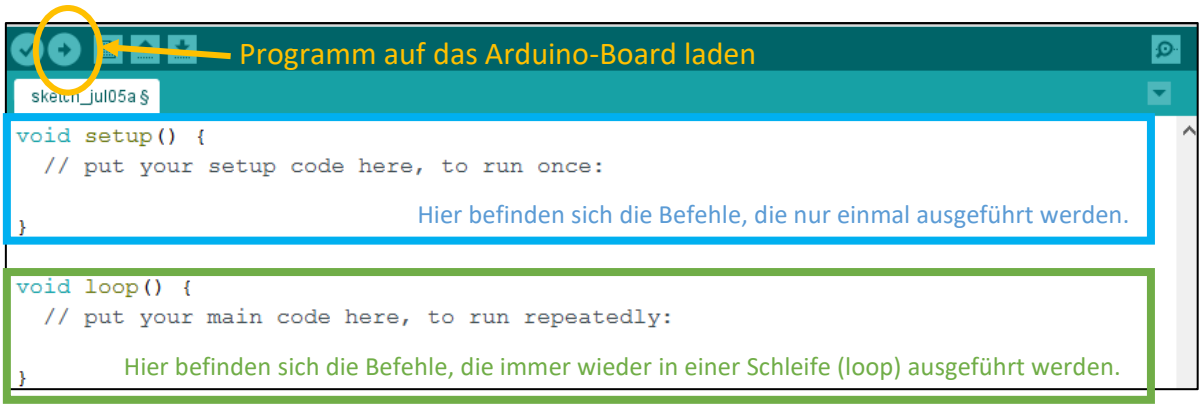
A. Erste Schritte mit dem Arduino

Arduino ist eine Open-Source-Elektronik-Prototyping-Plattform für flexible, einfach zu bedienende Hardware und Software im Bereich Mikrocontrolling. Es ist geeignet, um in kurzer Zeit spektakuläre Projekte zu verwirklichen. Viele davon lassen sich unter dem Begriff „Arduino“ bei Youtube finden. Es wird vor allem von Künstlern, Designern, Tüftlern und Bastlern verwendet, um kreative Ideen zu verwirklichen.

Aber auch in Schulen, Hochschulen und Universitäten wird die Arduinoplattform zunehmend eingesetzt, um Lernenden einen kreativen und spannenden, aber vor allem auch einfachen Zugang zum Thema „Mikrocontrolling“ zu ermöglichen.

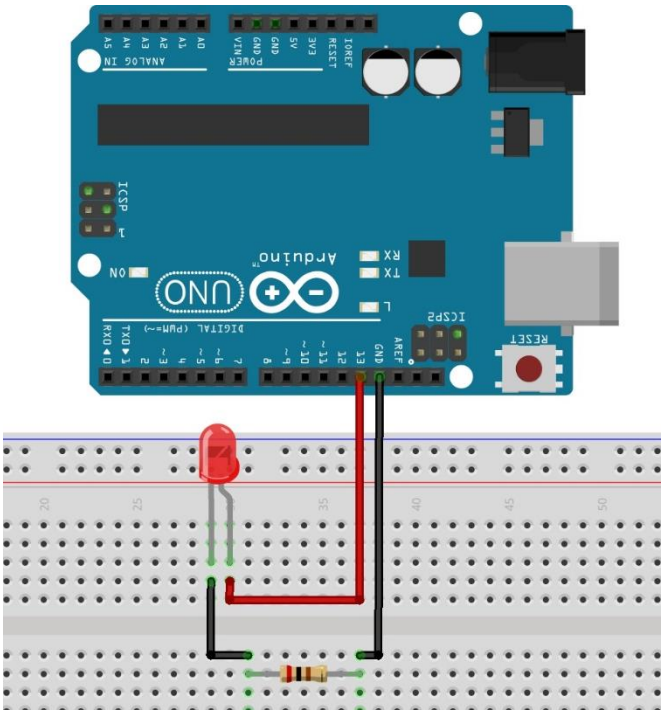
Das folgende Bild zeigt das Fenster, das sich öffnet, wenn man die Arduino-Software startet.





Beispiel 1: *LED ein- und ausschalten*

- Baue den Stromkreis auf dem Bild nach. Der Strom fließt jetzt durch den roten Draht zu der LED und dann über den schwarzen Draht zurück zum Arduino.



Damit die LED nicht zu hell leuchtet (und dadurch beschädigt wird), fügt man einen Widerstand (hier 200 Ohm) hinzu.

Das *längere Beinchen* der LED wird an den roten Draht (+, Pin 13) angeschlossen. Das *kürzere der Beinchen* wird mit dem schwarzen Draht (–, GND) verbunden.

Wichtige Infos zu LEDs:

- Der Strom kann nur in einer Richtung durch die LED fließen. Daher muss sie korrekt angeschlossen werden. Eine LED hat einen längeren und einen kürzeren Kontakt. Der längere Kontakt ist + und der kürzere ist –.
- Eine LED ist für eine bestimmte Spannung ausgelegt. Wird diese Spannung unterschritten, leuchtet die LED weniger hell oder sie bleibt aus. Wird die Spannung jedoch überschritten, brennt die LED sehr schnell durch und wird an den Kontakten sehr heiß (ACHTUNG!).
- Typische Spannungswerte nach LED-Farben: Blau: 3,1V, Weiß: 3,3V, Grün: 3,7V, Gelb: 2,2V, Rot: 2,1V. Die Spannung an den digitalen Ports (PINs) des Boards beträgt 5V. Beim direkten Anschluss an diese Ports gibt jede LED recht schnell den Geist auf (brennt durch). Daher muss ein Widerstand mit in den Stromkreis geschaltet werden. Im Internet gibt es unter dem Suchbegriff „Widerstandsrechner LED“ sehr gute Hilfen dazu.
- Unverbindliche Empfehlung für Widerstände an folgenden LEDs (bei Anschluss an die 5V-PINs des Mikrocontroller-Boards):

LED:	Weiß	Rot	Gelb	Grün	Blau	IR
Widerstand:	100 Ohm	200 Ohm	200 Ohm	100 Ohm	100 Ohm	100 Ohm

- Schreibe das folgende Programm ab und lade es auf dein Arduino.
  (Die Kommentare //…. brauchst du nicht abzuschreiben.)

Eine_blinkende_LED

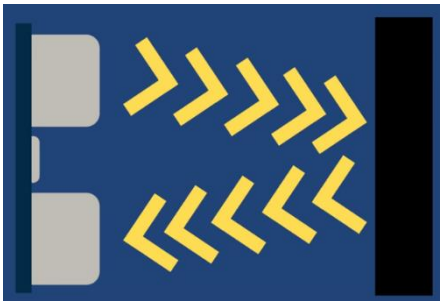
```
int LED_PIN = 13;            // PIN 13 ist mit der LED verbunden


void setup()                 // Hier beginnt das Setup
{                            // Hier beginnt ein Programmabschnitt
pinMode(LED_PIN, OUTPUT);    // Hier sagen wir dem Arduino, dass der PIN "LED_PIN"
                             // (Pin 13) ein Ausgang sein soll
}                            // Hier endet ein Programmabschnitt


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

// Hier am Ende springt das Programm an den Start des Loop-Teils. Also...
// ...schalte die Spannung an PIN 13 ein.
// ... usw... usw... usw...
```

Aufgabe 1: Was musst du ändern, wenn du die LED jetzt an den PIN 9 anschließt?
Aufgabe 2: Füge eine zweite LED zu deinem Aufbau hinzu und verändere das Programm so, dass die LEDs abwechselnd eingeschaltet sind. Speichere das Programm unter dem Namen „LED“.

B. Entfernungen mit Hilfe eines Ultraschallsensors messen

In vielen Projekten ist es notwendig, dass man die Entfernung zwischen dem Roboter und einem Hindernis messen kann. Dazu benutzen wir in der Regel einen Ultraschallsensor (Typ HC-SR04).

Wie funktioniert der Sensor?



Der Ultraschallsensor erhält über den Trigger-PIN ein Signal, dass er die Entfernung messen soll. Dadurch wird eine Ultraschallwelle gesendet, welche dann auf Hindernisse treffen kann. Falls die Welle auf ein Hindernis trifft, wird sie zum Sensor zurückgeworfen und der Echo-PIN sendet ein Signal zum Arduino. Die Entfernungsmessung mithilfe eines Ultraschallsensors funktioniert ganz ähnlich wie die Entfernungsmessung einer Fledermaus.

Die Zeit, die zwischen dem Entsenden und dem Erhalten des Signals vergeht, erlaubt uns die Entfernung zu berechnen:

$$d\ (in\ m)\ = 343{,}2 \cdot t\ (in\ s)$$

Die Schallgeschwindigkeit beträgt in relativ trockener Luft und durchschnittlichem Luftdruck in Bodennähe bei 20°C etwa 343,2 m/s = 34‘320 cm/s = 34,320 cm/ms = 0,034320 cm/µs.

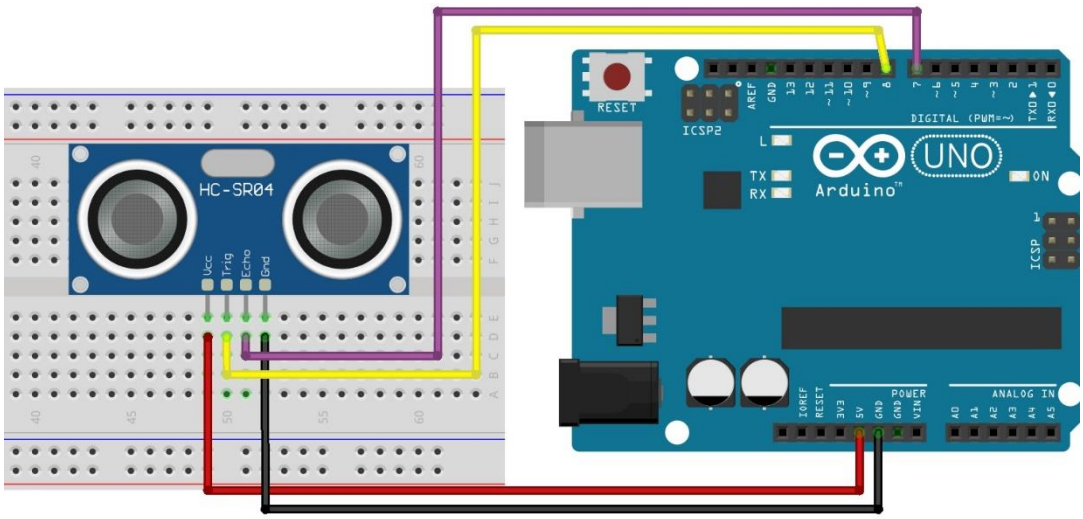
µs: Mikrosekunde (1 Sekunde = 1‘000‘000 µs)

Man teilt zunächst die Zeit durch zwei (weil man ja nur eine Strecke berechnen möchte und nicht die Strecke hin und zurück). Den Wert multipliziert man mit der Schallgeschwindigkeit in der Einheit Zentimeter/Mikrosekunde und erhält dann den Wert in Zentimetern.

$$d\ (in\ cm) = 0{,}03432 \cdot \frac{t}{2}\ (in\ \mu s)$$

Beispiel 2: *Entfernung messen und anzeigen lassen*

- Baue den Stromkreis auf dem Bild nach.



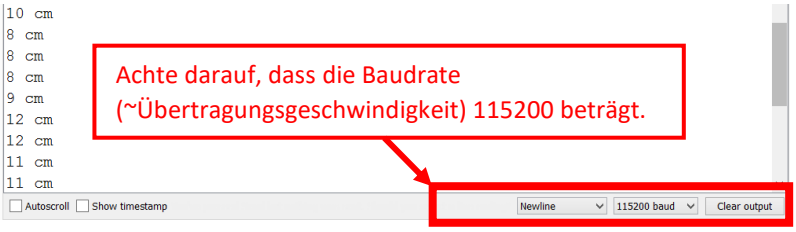
- Lies das folgende Programm durch, schreibe es ab und lade es auf dein Arduino.
  Speichere das Programm unter dem Namen „Ultraschallsensor 1“.

Ultraschallsensor

```
int trigger=8;          // Trigger-Pin des Ultraschallsensors an PIN 8 des Arduino-Boards
int echo=7;             // Echo-PIN des Ultraschallsensors an PIN 7 des Arduino-Boards
long dauer=0;           // Das Wort "dauer" ist jetzt eine Variable, unter der die Zeit gespeichert
                        // wird, die eine Schallwelle bis zur Reflektion und zurück benötigt.
                        // Startwert ist hier 0.
long entfernung=0;      // Das Wort "entfernung" ist jetzt die Variable, unter der die berechnete
                        // Entfernung gespeichert wird.
                        // Anstelle von "int" steht hier vor den beiden Variablen "long".
                        // Das hat den Vorteil, dass eine größere Zahl gespeichert werden kann.
                        // Nachteil: Die Variable benötigt mehr Platz im Speicher.

void setup()
{
Serial.begin (115200);        // Serielle Kommunikation starten, damit man sich später die Werte
                              // am Serial Monitor ansehen kann.
pinMode(trigger, OUTPUT);     // Trigger-Pin ist ein Ausgang
pinMode(echo, INPUT);         // Echo-Pin ist ein Eingang
}

void loop()
{
digitalWrite(trigger, LOW);   // Hier nimmt man die Spannung für kurze Zeit vom Trigger-Pin,
                              // damit man später beim Senden des Trigger-Signals ein
                              // rauschfreies Signal hat.
delay(5);                     // 5 Millisekunden warten
digitalWrite(trigger, HIGH);  // Jetzt sendet man eine Ultraschallwelle los.
delay(10);                    // Dieser "Ton" erklingt für 10 Millisekunden.
digitalWrite(trigger, LOW);   // Dann wird der „Ton“ abgeschaltet.
dauer = pulseIn(echo, HIGH);  // Mit dem Befehl "pulseIn" zählt der Mikrocontroller die
                              // Zeit in Mikrosekunden, bis der Schall zum
                              // Ultraschallsensor zurückkehrt.

entfernung = (dauer/2) * 0.03432;     // Nun berechnet man die Entfernung in Zentimetern.

Serial.print(entfernung);     // Dieser Befehl schreibt die Entfernung in den Serial Monitor
Serial.println(" cm");        // Hinter dem Wert der Entfernung soll auch am Serial Monitor die
                              // Einheit "cm" angegeben werden.
                              // print bedeutet "schreibe etwas und bleibe dann in der gleichen Zeile
                              // println bedeutet "schreibe etwas und gehe dann in die nächste Zeile"
delay(1000);                  // Warte 1'000 ms = 1 Sekunde und mache dann die nächste Messung.
}
```

Variablentyp	Bedeutung	Beschreibung
int	ganze Zahlen	ganze Zahlen (-32‘768 bis 32‘767)
long	ganze Zahlen	(-2 Milliarden bis 2 Milliarden) - gut, wenn man z.B. die abgelaufenen Millisekunden zählen will, da die schon mal über 32‘767 gehen
float	Fließkommazahlen	gebrochene Zahlen
char	Character	Alphanumerische Zeichen (Buchstaben, Zahlen, Sonderzeichen)
array	Variablenfeld	mehrere Werte eines Variablentyps können gespeichert werden

- Öffne den „Serial Monitor“
  (Tools -> Serial Monitor).



C. Funktionen erstellen und benutzen

Damit auch größere Programme übersichtlich bleiben, versucht man alle Befehle, die mehrfach benutzt werden, in Funktionen zu verpacken.

Beispiel 3: *eine Funktion, die Entfernung bestimmt*

- Benutze denselben Aufbau wie beim letzten Programm „Ultraschallsensor 1“.

Ultraschallsensor_mit_Funktion_1 §

```
int trigger=8;     // Den Wert von "trigger" benutzen wir überall im Programm, deshalb müssen
                   // wir dem Programm bereits hier sagen, dass es die Variable "trigger" gibt.
int echo=7;
long d = 0;        // Die Variable "d" erhält den Wert, den die Funktion "entfernungMessen()" berechnet hat.


void setup()
{
Serial.begin (115200);
pinMode(trigger, OUTPUT);
pinMode(echo, INPUT);
}

long entfernungMessen()      // Das hier ist unsere Funktion
{
long dauer=0;                // Den Wert von "dauer" benutzen wir nur in diesem Block (Funktion), deshalb br
                             // wir auch nur hier dem Programm zu sagen, dass es die Variable "dauer" gibt.
long entfernung=0;
digitalWrite(trigger, LOW);
delay(5);
digitalWrite(trigger, HIGH);
delay(10);
digitalWrite(trigger, LOW);
dauer = pulseIn(echo, HIGH);
entfernung = (dauer/2) * 0.03432;
return entfernung;          // Die Funktion "entfernungMessen()" gibt uns den Wert von "entfernung" zurück.
}
void loop()
{
d = entfernungMessen();    // Hier wird unsere Funktion aufgerufen und die Variable "d" erhält den Wert,
                           // den die Funktion "entfernungMessen()" berechnet hat.
Serial.print(d);
Serial.println(" cm");
delay(1000);
}
```

- Öffne den „Serial Monitor“ und überprüfe, ob das Programm noch immer richtig funktioniert.
- Speichere das Programm unter dem Namen „Ultraschallsensor 2“.

Im nächsten Beispiel gehen wir noch einen Schritt weiter, indem wir auch das Anzeigen der Entfernung in die Funktion einbauen. In dem Fall braucht die Funktion uns den Wert von der Variabel „entfernung“ gar nicht zurückzugeben. Eine Funktion, welche nichts zurückgibt, beginnt man mit dem Begriff „void“.

Beispiel 4: *eine Funktion, die Entfernung bestimmt und anzeigt*

- Benutze denselben Aufbau wie beim letzten Programm „Ultraschallsensor 2“.

Ultraschallsensor_mit_Funktion §

```
int trigger=8;
int echo=7;


void setup()
{
Serial.begin (115200);
pinMode(trigger, OUTPUT);
pinMode(echo, INPUT);
}

void entfernungMessen()     // Das hier ist unsere Funktion
{
long dauer=0;
long entfernung=0;

digitalWrite(trigger, LOW);
delay(5);
digitalWrite(trigger, HIGH);
delay(10);
digitalWrite(trigger, LOW);
dauer = pulseIn(echo, HIGH);
entfernung = (dauer/2) * 0.03432;
Serial.print(entfernung);
Serial.println(" cm");
}

void loop()
{                       // Das Hauptprogramm besteht jetzt nur noch aus zwei Befehlen.
entfernungMessen();     // Hier wird unsere Funktion aufgerufen
delay(1000);
}
```

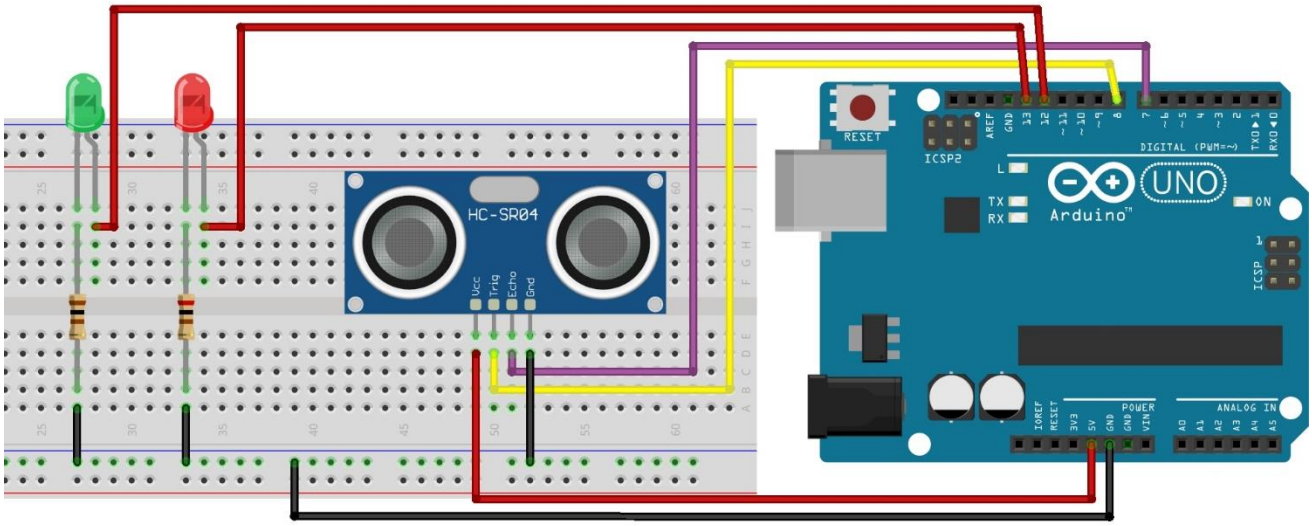
- Öffne den „Serial Monitor“ und überprüfe, ob das Programm noch immer richtig funktioniert.
- Speichere das Programm unter dem Namen „Ultraschallsensor 3“.

D. Bedingungen testen - IF-Anweisungen

In der Robotik werden oft Befehle durchgeführt, nachdem eine Bedingung überprüft wurde. Ein selbstfahrendes Auto misst z.B. ständig die Entfernung zum nächsten Gegenstand und stoppt, falls diese Entfernung zu klein ist. Solche Entscheidungen werden mit dem IF-Befehl getroffen. (if = Englisch für „wenn“)

Beispiel 5: *Entfernung messen und zwei LEDs steuern*

- Das Programm soll die Entfernung messen und eine grüne LED einschalten, wenn die Entfernung größer als 15 cm ist und eine rote LED, wenn die Entfernung kleiner als 15 cm ist. Baue einen Stromkreis, der die benötigten Bauteile enthält.
  Der Vorwiderstand der roten LED beträgt 200 Ohm und der der grünen LED 100 Ohm.



- Verwende den Aufbau und das Programm aus Beispiel 3: „Ultraschallsensor 2“.
  (Die Kommentare //…. brauchst du nicht abzuschreiben.)

Verify

Ultraschallsensor_mit_Funktion_IF_LED

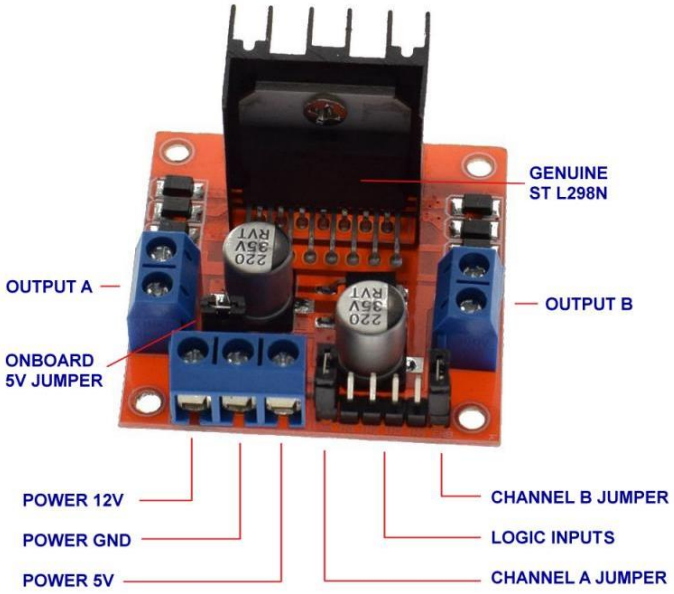
```
int trigger=8;
int echo=7;
long d=0;
int led_rot=13;          // Die rote LED wird an PIN 13 angeschlossen
int led_grun=12;         // Die grüne LED wird an PIN 12 angeschlossen
long d_min=15;           // Minimaler Abstand von 15 cm

void setup()
{
Serial.begin (115200);
pinMode(trigger, OUTPUT);
pinMode(echo, INPUT);
pinMode(led_rot, OUTPUT);    // Hier sagen wir dem Arduino, dass der PIN "led_rot"
                             // (Pin 13) ein Ausgang sein soll
pinMode(led_grun, OUTPUT);   // Hier sagen wir dem Arduino, dass der PIN "led_grun"
}                            // (Pin 12) ein Ausgang sein soll
```

```
long entfernungMessen()
{
long dauer=0;
long entfernung=0;
digitalWrite(trigger, LOW);
delay(5);
digitalWrite(trigger, HIGH);
delay(10);
digitalWrite(trigger, LOW);
dauer = pulseIn(echo, HIGH);
entfernung = (dauer/2) * 0.03432;
return entfernung;
}

void loop()
{
d = entfernungMessen();
Serial.print(d);
Serial.println(" cm");

if (d < d_min)                    // Wenn die Entfernung kleiner als der
{                                 // Mindestabstand "d_min" wird,
  digitalWrite(led_rot, LOW);     // wird die rote LED aus- und
  digitalWrite(led_grun, HIGH);   // die grüne LED eingeschaltet.
}
else                              // else -> andernfalls
{
  digitalWrite(led_rot, HIGH);    // die rote LED wird ein- und
  digitalWrite(led_grun, LOW);    // die grüne LED ausgeschaltet.
}
delay(1000);
}
```

- Öffne den „Serial Monitor“ und überprüfe, ob das Programm noch immer richtig funktioniert.
- Speichere das Programm unter dem Namen „Ultraschallsensor 4“.

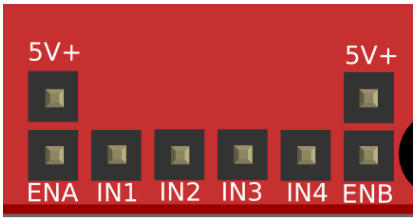
E. Motor mit einem L298N Motortreiber steuern

Wir werden zuerst einen, und danach zwei Motoren mit einem Motortreiber ansteuern. Danach haben wir alles was wir benötigen, um ein erstes Auto, das einem Hindernis ausweicht, zu bauen.



Das Bild links zeigt den Motortreiber L298N. Der Motortreiber wird benutzt, um mit einem oder mehreren Motoren zu kommunizieren (ein-/ausschalten, schneller/langsamer drehen, nach hinten/vorne drehen).

Auf den Seiten befinden sich die Anschlüsse für die Motoren und vorne links (blau) befinden sich die Anschlüsse für die Batterie und die Stromversorgung des Arduinos. Das rote Kabel der Batterie wird bei „Power 12V“ angeschlossen. Das schwarze Kabel der Batterie wird bei „Power GND“ (Ground) angeschlossen. Möchte man das Arduino mithilfe der Batterie versorgen, dann verbindet man den 12V-Anschluss des Motortreibers mit dem „VIN“-PIN des Arduinos und die beiden GND-Anschlüsse miteinander (siehe Bild Seite 11).



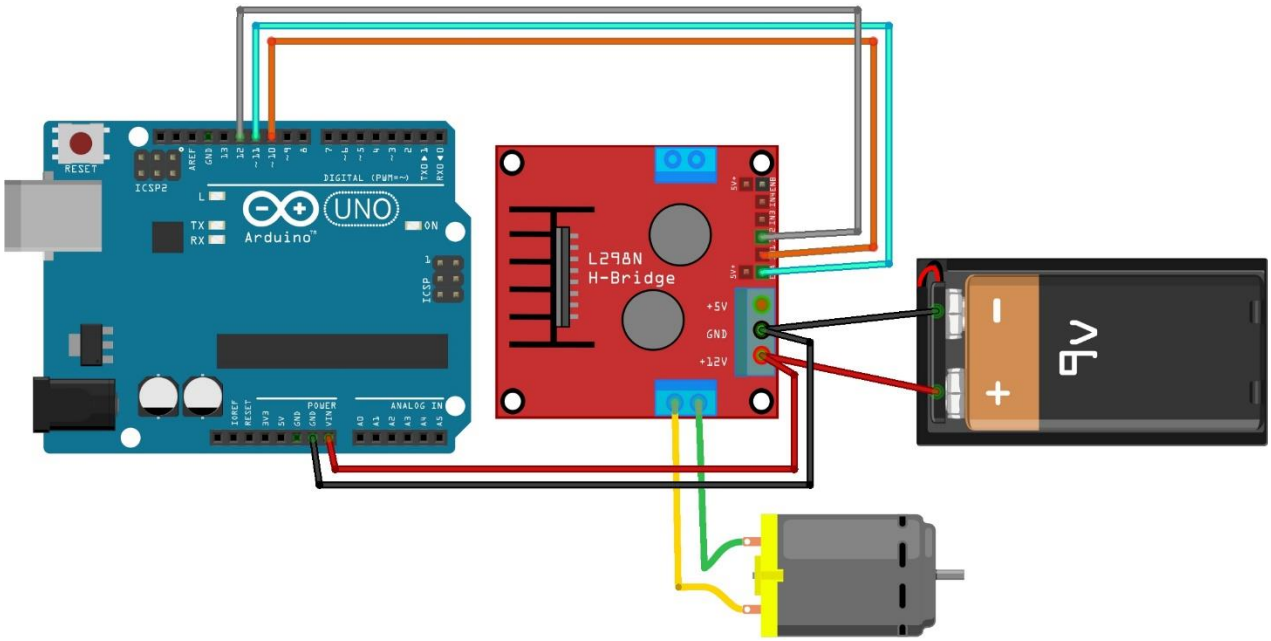
Vorne rechts befinden sich die PINs, die man benutzt, um den Motor zu steuern. Das folgende Bild zeigt eine Vergrößerung dieser PINs. Die PINs *ENA*, *IN1* und *IN2* werden benutzt, um den Motor auf der linken Seite und die PINs *IN3*, *IN4* und *ENB* um den Motor auf der rechten Seite zu steuern.

Die ENA- und ENB-PINs müssen mit einem PWM-PIN (PINs mit dem ~ Symbol) des Arduinos verbunden werden. Über diese PINs wird die Geschwindigkeit des Motors gesteuert. Die anderen PINs können benutzt werden, um die Motoren nach vorne oder nach hinten drehen zu lassen.

Der Arduino ist ein digitaler Mikrocontroller. Er kennt an seinen Ausgängen nur *„5 Volt an“* oder *„5V aus“*. Um die Helligkeit einer LED zu variieren, müsste man die Spannung jedoch variieren können. Zum Beispiel 5V wenn die LED hell leuchtet, 4V wenn sie etwas dunkler leuchtet, usw. DAS GEHT AN DIGITALEN PINS ABER NICHT. Es gibt jedoch eine Alternative. Sie nennt sich **Pulsweitenmodulation** (PWM). Die PWM lässt die 5V Spannung pulsieren. Die Spannung wird also im Millisekundenbereich ein- und ausgeschaltet. Bei einer hohen PWM (Max. 255) liegen die 5V nahezu durchgehend am jeweiligen PIN an. Bei einer geringen PWM (Min. 0) ist das 5V Signal kaum noch vorhanden. Mit dieser PWM kann man bei LEDs einen ähnlichen Effekt erreichen, als würde man die Spannung variieren. Nicht alle digitalen PINs am Board haben die PWM-Funktion. Die PINs an denen die PWM funktioniert sind besonders gekennzeichnet, bspw. durch eine kleine Welle vor der PIN-Nummer (PINs mit dem ~ Symbol).

Beispiel 6: *einen Motor mit einem L298N-Motortreiber steuern*

- Baue den Stromkreis auf dem Bild nach. (ENA -> 11, IN1 -> 10, IN2 ->12)



- Lies das folgende Programm durch, tippe es ab und lade es auf dein Arduino.

H_Bridge_1 §

```
int enA = 11;
int in1 = 10;
int in2 = 12;
int speedMotor_1 = 255; // Je größer dieser Wert ist,
                        // umso schneller dreht sich der Motor
                        // Minimalwert = 0 (Motor dreht gar nicht)
                        // Maximalwert = 255 (Motor dreht mit Maximalgeschwindigkeit)
void setup()
{
 pinMode(enA, OUTPUT);
 pinMode(in1, OUTPUT);
 pinMode(in2, OUTPUT);
 analogWrite(enA, speedMotor_1);  // Hier wird über den PIN 11 (enA)dem Microprozessor
                                  // die Geschwindigkeit (hier 255) übergeben
}

void loop() {
  digitalWrite(in1, LOW);   // Der Motor dreht nach vorn
  digitalWrite(in2, HIGH);
  delay(2000);
  digitalWrite(in1, LOW);   // Der Motor dreht sich nicht
  digitalWrite(in2, LOW);
  delay(2000);
  digitalWrite(in1, HIGH);   // Der Motor dreht nach hinten
  digitalWrite(in2, LOW);
  delay(2000);
}
```

- Speichere das Programm unter dem Namen „Motor 1“.

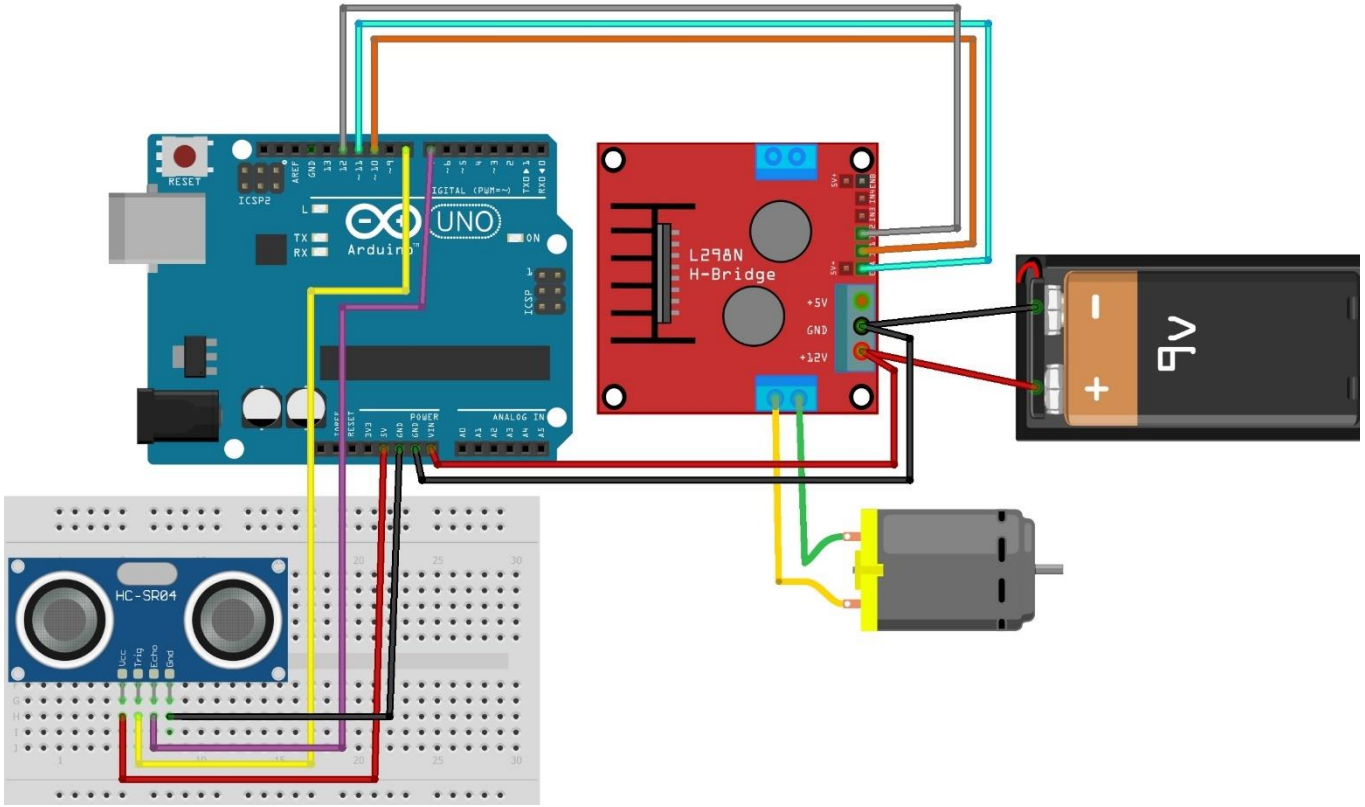
Aufgabe 1: *Ändere das Programm so, dass du eine Funktion schreibst, die den Motor nach vorne drehen lässt, eine andere Funktion, die den Motor nach hinten drehen lässt und eine Funktion, die den Motor stoppt.*

H_Bridge_2 §

```
int enA = 11;
int in1 = 10;
int in2 = 12;
int speedMotor_1 = 255; // Je größer dieser Wert ist,
                        // umso schneller dreht sich der Motor
                        // Minimalwert = 0 (Motor dreht gar nicht)
                        // Maximalwert = 255 (Motor dreht mit Maximalgeschwindigkeit)
void setup()
{
 pinMode(enA, OUTPUT);
 pinMode(in1, OUTPUT);
 pinMode(in2, OUTPUT);
 analogWrite(enA, speedMotor_1);  // Hier wird über den PIN 11 (enA)dem Microprozessor
                                  // die Geschwindigkeit (hier 255) übergeben
}

void stoppen()            // Funktion "stoppen()" -> Motor dreht sich nicht
{
 digitalWrite(in1, LOW);
 digitalWrite(in2, LOW);
}

void nach_vorn()          // Funktion "nach_vorn()" -> Motor dreht nach vorn
{
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
}
void nach_hinten()        // Funktion "nach_hinten()" -> Motor dreht nach hinten
{
 digitalWrite(in1, HIGH);
 digitalWrite(in2, LOW);
}

void loop() {
  nach_vorn();      // Funktion wird aufgerufen
  delay(2000);
  stoppen();        // Funktion wird aufgerufen
  delay(2000);
  nach_hinten();    // Funktion wird aufgerufen
  delay(2000);
}
```

- Speichere das Programm unter dem Namen „Motor 2“.

Aufgabe 2: *Ändere die Schaltung und das Programm so, dass der Motor stoppt, wenn die Entfernung zum nächsten Gegenstand kleiner als ein bestimmter Mindestabstand (20 cm) ist.* Das Programm „Ultraschallsensor 4“ kann dir dabei helfen.



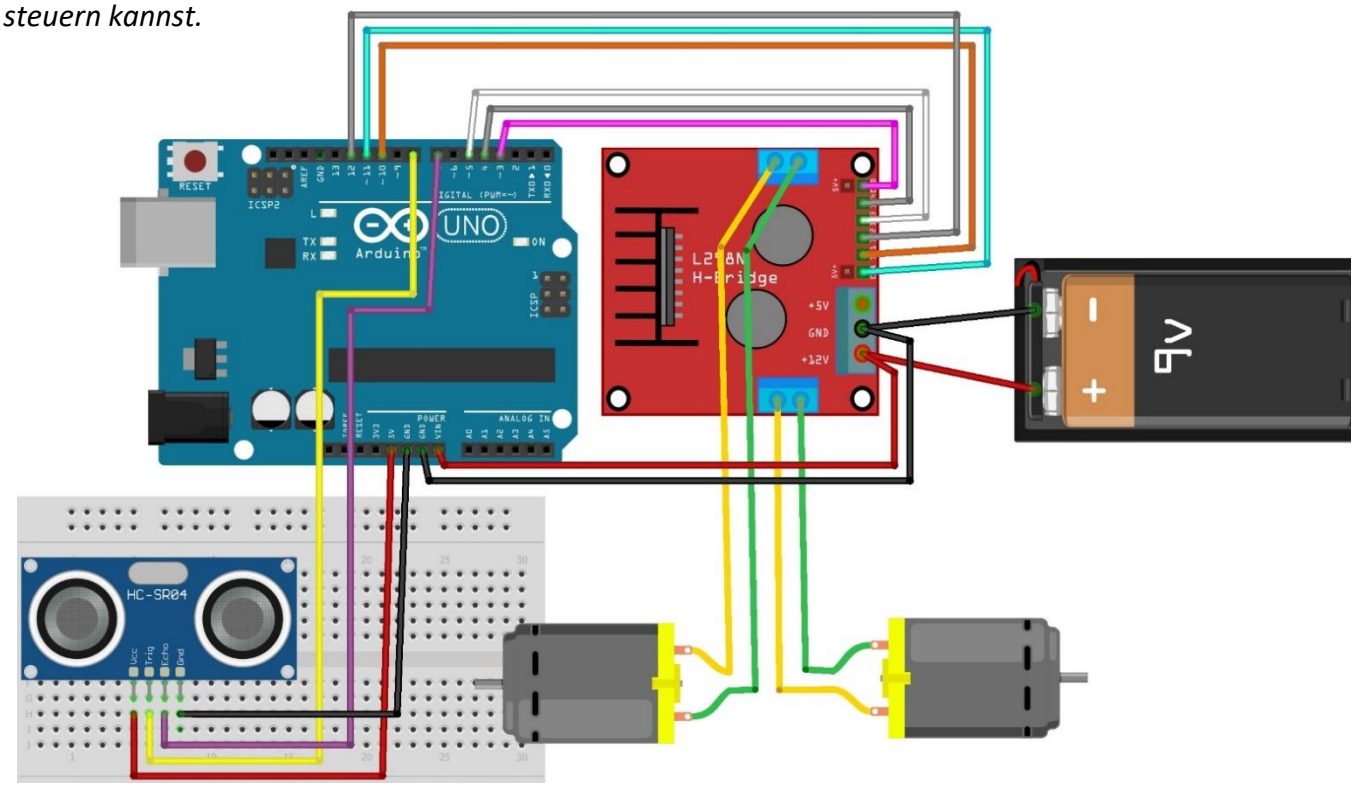
H_Bridge_3 §

```
int trigger=8;
int echo=7;
long d=0;
long d_min=20;

int enA = 11;
int in1 = 10;
int in2 = 12;
int speedMotor_1 = 255;

void setup()
{
 Serial.begin (115200);
 pinMode(trigger, OUTPUT);
 pinMode(echo, INPUT);

 pinMode(enA, OUTPUT);
 pinMode(in1, OUTPUT);
 pinMode(in2, OUTPUT);
 analogWrite(enA, speedMotor_1);
}
```

```
long entfernungMessen()      // Funktion "entfernungMessen()" -> Entfernung wird gemessen
{
  long dauer=0;
  long entfernung=0;
  digitalWrite(trigger, LOW);
  delay(5);
  digitalWrite(trigger, HIGH);
  delay(10);
  digitalWrite(trigger, LOW);
  dauer = pulseIn(echo, HIGH);
  entfernung = (dauer/2) * 0.03432;
  return entfernung;
}

void stoppen()           // Funktion "stoppen()" -> Motor dreht nicht
{
 digitalWrite(in1, LOW);
 digitalWrite(in2, LOW);
}

void nach_vorn()         // Funktion "nach_vorn()" -> Motor dreht nach vorn
{
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
}
void nach_hinten()       // Funktion "nach_hinten()" -> Motor dreht nach hinten
{
 digitalWrite(in1, HIGH);
 digitalWrite(in2, LOW);
}

void loop()                          // Hauptprogramm
{
  d = entfernungMessen();            // Funktion "entfernungMessen()" wird aufgerufen
  Serial.print(d);
  Serial.println(" cm");

  if (d < d_min)                      // Wenn die Entfernung kleiner als der
  {                                   // Mindestabstand "d_min" wird,
    stoppen();                        // wird die Funktion "stoppen()" aufgerufen
  }
  else                                // else -> andernfalls
  {
    nach_vorn();                      // Funktion "nach_vorn()" wird aufgerufen
  }
  delay(1000);
}
```

- Speichere das Programm unter dem Namen „Motor 3“.

Aufgabe 3: *Ändere die Schaltung und das Programm so, dass du zwei Motoren mit dem Motortreiber steuern kannst.*



H_Bridge_4 §

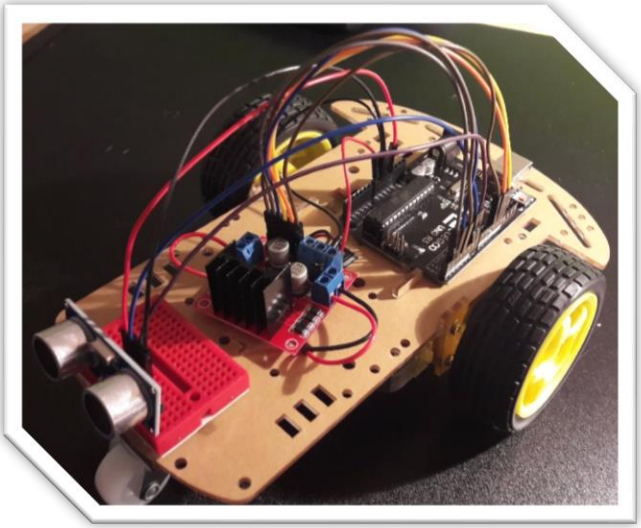
```
int trigger=8;
int echo=7;
long d=0;
long d_min=30;
int enA = 11;
int in1 = 10;
int in2 = 12;
int speedMotor_1 = 150;
int enB = 3;
int in3 = 5;
int in4 = 4;
int speedMotor_2 = 150;

void setup()
{
 Serial.begin (115200);
 pinMode(trigger, OUTPUT);
 pinMode(echo, INPUT);
 pinMode(enA, OUTPUT);
 pinMode(in1, OUTPUT);
 pinMode(in2, OUTPUT);
 analogWrite(enA, speedMotor_1);
 pinMode(enB, OUTPUT);
 pinMode(in3, OUTPUT);
 pinMode(in4, OUTPUT);
 analogWrite(enB, speedMotor_2);
}
```

```
long entfernungMessen()      // Funktion "entfernungMessen()" -> Entfernung wird gemessen
{
  long dauer=0;
  long entfernung=0;
  digitalWrite(trigger, LOW);
  delay(5);
  digitalWrite(trigger, HIGH);
  delay(10);
  digitalWrite(trigger, LOW);
  dauer = pulseIn(echo, HIGH);
  entfernung = (dauer/2) * 0.03432;
  return entfernung;
}

void stoppen()              // Funktion "stoppen()" -> Das Auto fährt nicht
{
 digitalWrite(in1, LOW);
 digitalWrite(in2, LOW);
 digitalWrite(in3, LOW);
 digitalWrite(in4, LOW);
}
void nach_vorn()            // Funktion "nach_vorn()" -> Das Auto fährt nach vorn
{
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
 digitalWrite(in3, LOW);
 digitalWrite(in4, HIGH);
 Serial.println("vorn");
}

void nach_hinten()          // Funktion "nach_hinten()" -> Das Auto fährt nach hinten
{
 digitalWrite(in1, HIGH);
 digitalWrite(in2, LOW);
 digitalWrite(in3, HIGH);
 digitalWrite(in4, LOW);
 Serial.println("hinten");
}

void nach_rechts()          // Funktion "nach_rechts()" -> Das Auto dreht nach rechts
{
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
 digitalWrite(in3, HIGH);
 digitalWrite(in4, LOW);
 Serial.println("rechts");
}
void nach_links()           // Funktion "nach_links()" -> Das Auto dreht nach links
{
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
 digitalWrite(in3, HIGH);
 digitalWrite(in4, LOW);
}
```

```
void loop()                          // Hauptprogramm
{
  d = entfernungMessen();            // Funktion "entfernungMessen()" wird aufgerufen
  Serial.print(d);
  Serial.println(" cm");

  if (d < d_min)                     // Wenn die Entfernung kleiner als der
  {                                  // Mindestabstand "d_min" wird,
    stoppen();                       // wird die Funktion "stoppen()" aufgerufen
    delay(100);
    nach_hinten();
    delay(500);
    nach_rechts();
    delay(500);
  }
  else                               // else -> andernfalls
  {
    nach_vorn();                     // Funktion "nach_vorn()" wird aufgerufen
  }
  delay(100);
}
```

- Speichere das Programm unter dem Namen „Motor 4“.

F. Erstes Roboterauto bauen



Jetzt bist du bereit, dein erstes Roboterauto zu bauen. Das Auto soll über einen Ultraschallsensor die Entfernung zum nächsten Gegenstand messen und sich umdrehen, falls diese Entfernung zu klein ist. Zusätzlich kann man einen Schutz einbauen, um das Auto zu stoppen, falls die Entfernung sehr klein ist (Unfallschutz).

Hilfe: *Wie kann das Roboterauto sich drehen?*

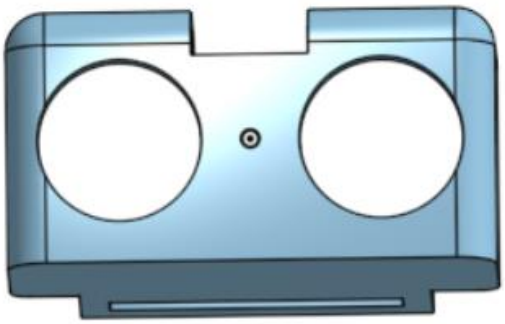
Das Auto dreht nach links, wenn

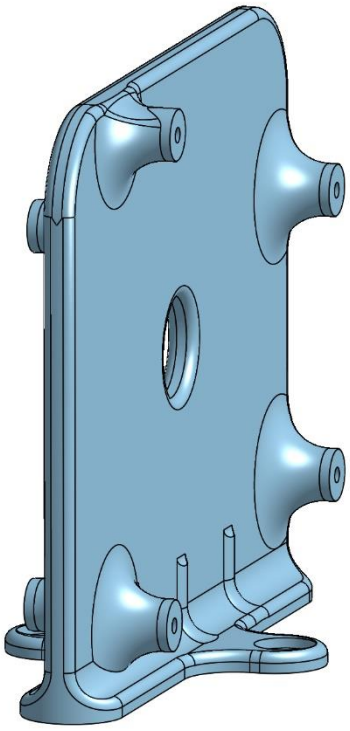
- der linke Motor langsamer dreht als der rechte Motor, oder
- der linke Motor nach hinten dreht und der rechte Motor nach vorne dreht.

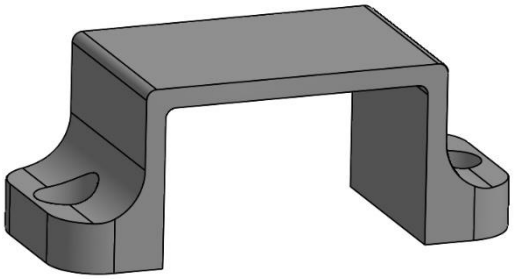
Das Auto dreht nach rechts, wenn

- der rechte Motor langsamer dreht als der linke Motor, oder
- der rechte Motor nach hinten dreht und der linke Motor nach vorne dreht.

An dieser Stelle kannst du auch bereits den 3D-Drucker benutzen, um einige Teile für dein Roboterauto zu drucken. Dies kann z.B. eine Halterung für den Ultraschallsensor sein, eine Befestigung für die 9V-Batterie oder eine Befestigung für den Motortreiber und das Arduino-Board.



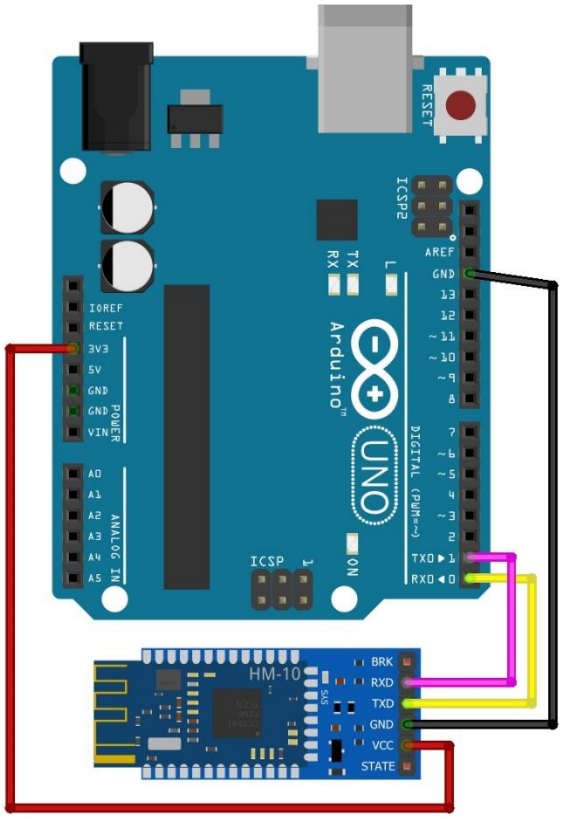




Speichere das Programm unter dem Namen „Roboterauto 1“.

G. Das Auto über Bluetooth mit dem Handy steuern

ArduinoBlue ist eine iOS/Android Anwendung, die es uns erlaubt, das Arduino-Board über Bluetooth zu steuern.
Wir verwenden hier das **HM-10** Bluetooth 4.0 BLE oder das **HM-11** Bluetooth 4.0 BLE Modul.



Beispiel: *Bluetooth-Modul*

- Baue die Schaltung auf dem Bild nach.
- Lies das folgende Programm durch, tippe es ab und lade es auf dein Arduino.

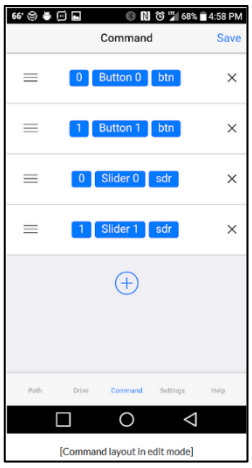
ArduinoBlue_UNO_TX_RX_Auto §

```
#include <SoftwareSerial.h>
#include <ArduinoBlue.h>
int altGaspedal = 49;
int altSteuerung = 49;
int gaspedal;
int steuerung;
int sliderVal;
int button;
int sliderId;
ArduinoBlue phone(Serial);

void setup()
{
    Serial.begin(9600);
    delay(100);
    Serial.println("setup complete");
}

void loop()
{
    button = phone.getButton();
    gaspedal = phone.getThrottle();    // Die Variablen "gaspedal" und "steuerung"
                                       // können einen Wert zwischen 0 und 99 haben
    steuerung = phone.getSteering();   // Wenn die Variablen "gaspedal" und "steuerung"
                                       // einen Wert von 49 (99/2) haben,
                                       // befindet sich der Joystick in der Mitte
    sliderId = phone.getSliderId();    // Id vom bewegten Slider
    sliderVal = phone.getSliderVal();  // Der Wert von "sliderVal" kann zwischen
                                       // 0 und 200 variieren
```

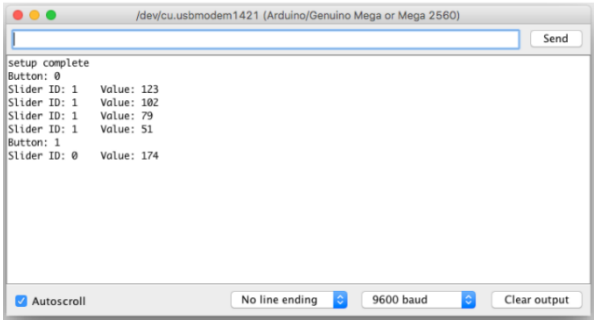
```
    if (button != -1)                   // Wenn ein Button betätigt wird
    {
        Serial.print("Button: ");
        Serial.println(button);
    }

    if (sliderId != -1)                 // Wenn ein Slider bewegt wird
    {
        Serial.print("Slider ID: ");
        Serial.print(sliderId);
        Serial.print("\tValue: ");
        Serial.println(sliderVal);
    }

    if (altGaspedal != gaspedal || altSteuerung != steuerung)  // Wenn der Wert von "gaspedal"
                                                               // oder "steuerung" ändert
    {
        Serial.print("Gaspedal: ");
        Serial.print(gaspedal);
        Serial.print("\tSteuerung: ");
        Serial.println(steuerung);
        altGaspedal = gaspedal;          // Wert von "gaspedal" wird "altGaspedal" übergeben
        altSteuerung = steuerung;        // Wert von "steuerung" wird "altSteuerung" übergeben
    }
}
```

- Speichere anschließend das Programm unter dem Namen „Bluetooth“.

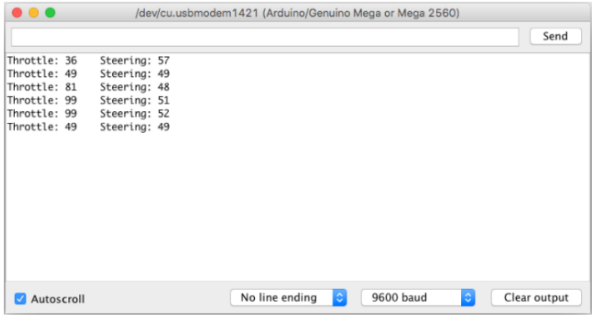
- Erstelle in der App *ArduinoBlue* zwei *Buttons* und zwei *Sliders*.



- Öffne in der Anwendung Arduino den „Serial Monitor“. Wenn du nun einen Button oder Slider in der App betätigst, wird dies im „Serial Monitor“ angezeigt.



- Benutze in der App den Joystick und betrachte im „Serial Monitor“, welche Werte sich ändern.



- Versuche ein Programm zu schreiben, welches dir erlaubt, das Auto über die App zu steuern.

ArduinoBlue_UNO_TX_RX_Auto_2 §

```
#include <SoftwareSerial.h>
#include <ArduinoBlue.h>
float altGaspedal = 49;
float altSteuerung = 49;
float gaspedal;
float steuerung;
int sliderVal;
int button;
int sliderId;
ArduinoBlue phone(Serial);
int enA = 11;
int in1 = 10;
int in2 = 12;
float speedMotor_1 = 0;
int enB = 3;
int in3 = 5;
int in4 = 4;
float speedMotor_2 = 0;

void setup()
{
    Serial.begin(9600);
    delay(100);
    pinMode(enA, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(enB, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);
}
void stoppen()              // Funktion "stoppen()" -> Motor dreht nicht
{
     digitalWrite(in1, LOW);
     digitalWrite(in2, LOW);
     digitalWrite(in3, LOW);
     digitalWrite(in4, LOW);
}

void nach_vorn()
{
      analogWrite(enA, speedMotor_1);
      analogWrite(enB, speedMotor_2);
      digitalWrite(in1, LOW);
      digitalWrite(in2, HIGH);
      digitalWrite(in3, LOW);
      digitalWrite(in4, HIGH);
      Serial.println("vorn");
}
```

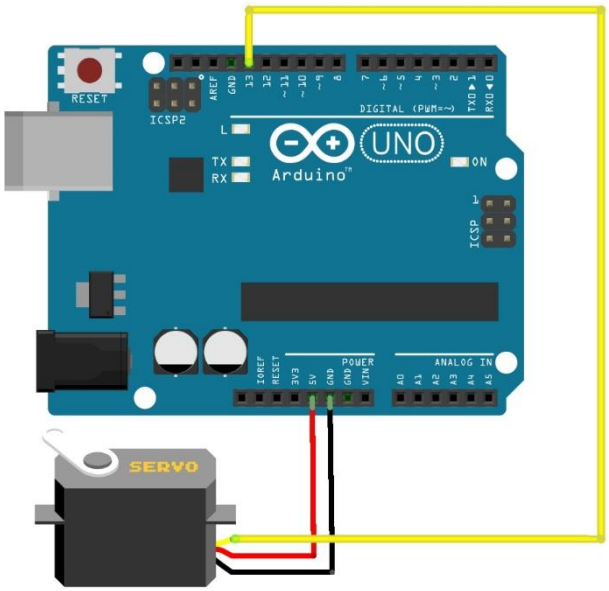
```
void nach_hinten()
{
     analogWrite(enA, speedMotor_1);
     analogWrite(enB, speedMotor_2);
     digitalWrite(in1, HIGH);
     digitalWrite(in2, LOW);
     digitalWrite(in3, HIGH);
     digitalWrite(in4, LOW);
     Serial.println("hinten");
}

void loop()
{
    button = phone.getButton();
    gaspedal = phone.getThrottle();    // Die Variablen "gaspedal" und "steuerung"
                                       // können einen Wert zwischen 0 und 99 habe
    steuerung = phone.getSteering();   // Wenn die Variablen "gaspedal" und
                                       // "steuerung" einen Wert von 49 (99/2) hab
                                       // befindet sich der Joystick in der Mitte
    sliderId = phone.getSliderId();    // Id vom bewegten Slider
    sliderVal = phone.getSliderVal();  // Der Wert von "sliderVal" kann zwischen
                                       // 0 und 200 variieren

    if (button != -1)                  // Wenn ein Button betätigt wird
    {
        Serial.print("Button: ");
        Serial.println(button);
    }
    if (sliderId != -1)                // Wenn ein Slider bewegt wird
    {
        Serial.print("Slider ID: ");
        Serial.print(sliderId);
        Serial.print("\tValue: ");
        Serial.println(sliderVal);
    }

    // Wenn der Wert von "gaspedal" oder "steuerung" ändert
    if (altGaspedal != gaspedal || altSteuerung != steuerung)
    {
        Serial.print("Gaspedal: ");
        Serial.print(gaspedal);
        Serial.print("\tSteuerung: ");
        Serial.println(steuerung);
        altGaspedal = gaspedal;        // Wert von "gaspedal" wird
                                       // "altGaspedal" übergeben
        altSteuerung = steuerung;      // Wert von "steuerung" wird
                                       // "altSteuerung" übergeben

        if (gaspedal > 48 && gaspedal < 52)
        {
            speedMotor_1 = 0;
            speedMotor_2 = 0;
        }
```

```
        if (gaspedal >= 52)
        {
          if (steuerung > 49 && steuerung < 99)
          {
            speedMotor_1 = (gaspedal-50)*5.1;
            speedMotor_2 = (gaspedal-50)*5.1*(1-pow((steuerung-50),2)/2500);
            Serial.print("enA: ");
            Serial.print(speedMotor_1);
            Serial.print("\tenB: ");
            Serial.println(speedMotor_2);
          }

          if (steuerung <= 49 && steuerung >= 1)
          {
            speedMotor_2 = (gaspedal-50)*5.1;
            speedMotor_1 = (gaspedal-50)*5.1*(1-pow((steuerung-50),2)/2500);
            Serial.print("enA: ");
            Serial.print(speedMotor_1);
            Serial.print("\tenB: ");
            Serial.println(speedMotor_2);
          }
          nach_vorn();
        }
        if (gaspedal < 48)
        {
          if (steuerung > 49 && steuerung < 99)
          {
            speedMotor_1 = (50-gaspedal)*5.1;
            speedMotor_2 = (50-gaspedal)*5.1*(1-pow((steuerung-50),2)/2500);
            Serial.print("enA: ");
            Serial.print(speedMotor_1);
            Serial.print("\tenB: ");
            Serial.println(speedMotor_2);
          }

          if (steuerung <= 49 && steuerung >= 1)
          {
            speedMotor_2 = (50-gaspedal)*5.1;
            speedMotor_1 = (50-gaspedal)*5.1*(1-pow((steuerung-50),2)/2500);
            Serial.print("enA: ");
            Serial.print(speedMotor_1);
            Serial.print("\tenB: ");
            Serial.println(speedMotor_2);
          }
          nach_hinten();
        }
    }
}
```

- Speichere anschließend das Programm unter dem Namen „Roboterauto 2“.

H. Ultraschallsensor mit einem Servomotor bewegen

Als nächster Schritt werden wir einen Servomotor benutzen, um den Ultraschallsensor auf dem Roboterauto in unterschiedliche Richtungen „schauen“ (hören) zu lassen. Bei Servomotoren kann man einen genauen Winkel einstellen. Sie werden z.B. auch in Roboterarmen verbaut oder im kleinen Roboter Otto.



Beispiel: *Servomotor steuern*

- Baue die Schaltung auf dem Bild nach.

- Lies das folgende Programm durch, tippe es ab und lade es auf dein Arduino.

Servo_1

```
#include <Servo.h>      // Die Servobibliothek wird aufgerufen. Sie wird benötigt,
                        // damit die Ansteuerung des Servos vereinfacht wird.
Servo servo1;           // Erstellt für das Programm ein Servo mit dem Namen „servo1“

void setup()
{
  servo1.attach(13);    // Das Setup enthält die Information, dass das Servo an
                        // der Steuerleitung (gelb) mit PIN 13 verbunden wird.
}

// Im „loop“ wird über den write-Befehl „servo1.write(Grad)“ das Servo angesteuert.
// Zwischen den einzelnen Positionen gibt es eine Pause,
// damit das Servo genug Zeit hat, die gewünschten Positionen zu erreichen.
void loop()
{
  servo1.write(0);      // Position 1 ansteuern mit dem Winkel 0°
  delay(3000);          // Das Programm stoppt für 3 Sekunden
  servo1.write(90);     // Position 2 ansteuern mit dem Winkel 90°
  delay(3000);          // Das Programm stoppt für 3 Sekunden
  servo1.write(180);    // Position 3 ansteuern mit dem Winkel 180°
  delay(3000);          // Das Programm stoppt für 3 Sekunden
  servo1.write(20);     // Position 4 ansteuern mit dem Winkel 20°
  delay(3000);          // Das Programm stoppt für 3 Sekunden
}
```

- Speichere anschließend das Programm unter dem Namen „Servo 1“.

Beispiel: *Servomotor mit for-Schleife*

- Lies das folgende Programm durch, tippe es ab und lade es auf dein Arduino.

Servo_2 §

```
#include <Servo.h>     // Die Servobibliothek wird aufgerufen. Sie wird benötigt,
                       // damit die Ansteuerung des Servos vereinfacht wird.
Servo servo1;          // Erstellt für das Programm ein Servo mit dem Namen „servo1"

void setup()
{
  Serial.begin(9600);
  servo1.attach(13);   // Das Setup enthält die Information, dass das Servo an
                       // der Steuerleitung (gelb) mit PIN 13 verbunden wird.
}

void loop()
{
  for(int i=0; i<100; i=i+10) // Die Befehle im Innern des For-Blocks werden so lange
                              // wiederholt, wie die Bedingung (i<100) erfüllt ist.
  {
    servo1.write(i);     // Servo auf einen Winkel von i° stellen
    delay(500);          // Das Programm stoppt für 0,5 Sekunden
    Serial.println(i);
  }
}
```

Dieses Programm verändert den Winkel alle 0,5 Sekunden um 10° bis dass der Winkel seinen maximalen Wert von 90° erreicht hat.
Speichere anschließend das Programm unter dem Namen „Servo 2“.

Erklärung: *Die for-Schleife (for-loop)*

Beim ersten Durchgang durch den for-Block hat i den Wert 0. Da dieser Wert kleiner ist als 100, durchläuft das Programm die Befehle im for-Block und stellt den Servomotor auf 0°. Am Ende des ersten Durchgangs wird i um 10 erhöht und man hat also jetzt i = 10. Dieser Wert ist noch immer kleiner als 100 und der for-Block wird erneut durchlaufen. Das passiert solange bis i = 100 beträgt und die Bedingung nicht mehr erfüllt ist. Dann ist die for-Schleife abgeschlossen.

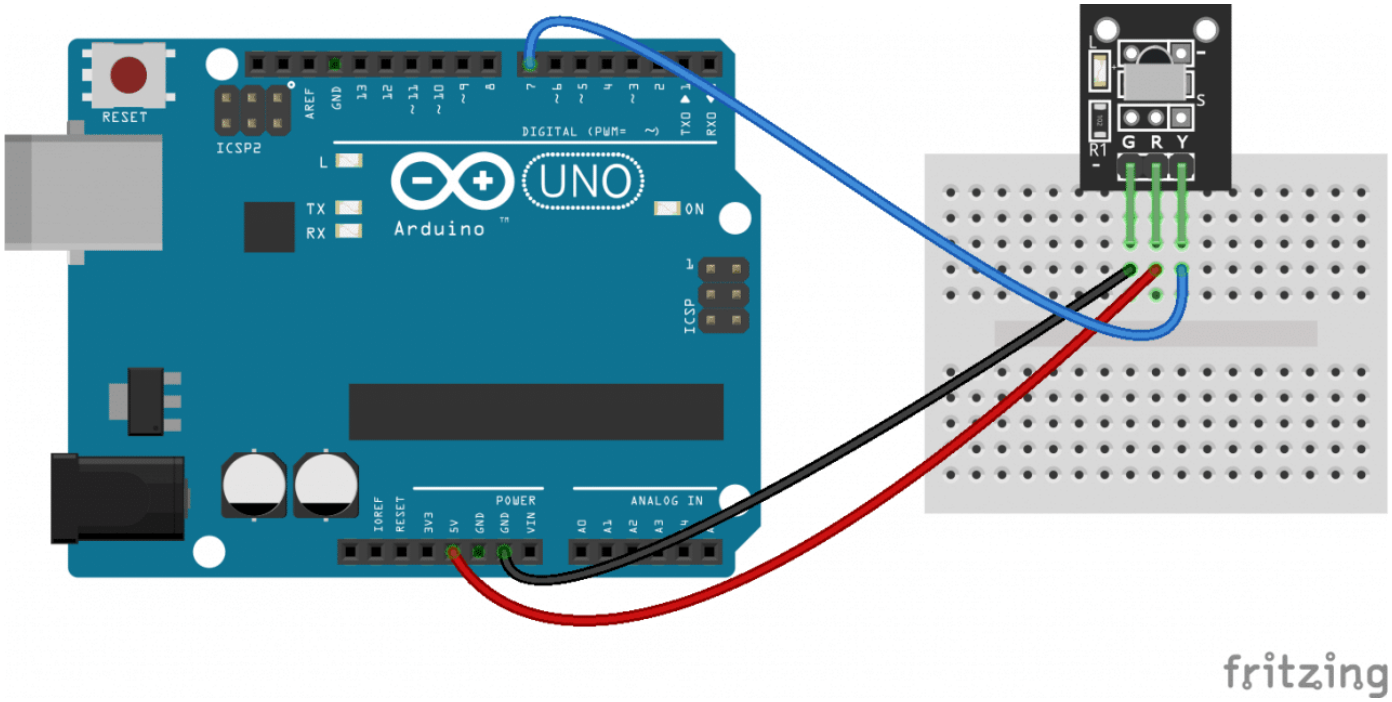
Aufgabe: *Benutze einen Servomotor, um den Ultraschallsensor auf dem Roboterauto von links nach rechts drehen zu lassen.* Speichere anschließend das Programm unter dem Namen „Roboterauto 3“.

I. Mit einer Infrarot-Fernbedienung das Auto steuern

Wir werden zunächst eine LED über die Infrarot-Fernbedienung steuern und uns dann später anschauen, wie man mit dieser Fernbedienung einen Motor steuern kann. Bei der Infrarot-Fernbedienung ist es wichtig, dass sie immer auf den Empfänger gerichtet ist. Anderenfalls können falsche Daten übertragen werden.

Beispiel: *Daten über die Fernbedienung übertragen*

- Baue den Stromkreis auf dem Bild nach.



Bei den Infrarot-Fernbedienungen ist es so, dass wenn man auf die Taste „1“ drückt, nicht eine „1“ an das Arduino gesendet wird. Um herauszufinden, welche Informationen an das Arduino geschickt werden, wenn die verschiedenen Tasten gedrückt werden, kann man folgendes Programm benutzen und eine Tabelle erstellen. Speichere das Programm unter dem Namen „IR-Fernbedienung 1“.

Infrarot_FB

```
IRrecv irrecv(ir_PIN);    // An dieser Stelle wird ein Objekt definiert,
                          // das den Infrarotsensor an Pin 7 ausliest.

decode_results results;   // die Nachrichten werden in "results" gespeichert

void setup(){
  Serial.begin(9600);
  irrecv.enableIRIn();    //Die Übertragung wird gestartet
}

void loop(){
  if (irrecv.decode(&results)) {     //Wenn Daten empfangen wurden,
    Serial.println(results.value, DEC); //werden sie als Dezimalzahl (DEC) im Serial-Monitor angezeigt.
    delay(100);
    irrecv.resume();  //Der Empfänger wird wieder "scharf" gestellt (er empfängt wieder Daten)
  }
}
```

Taste gedrückt	Empfangene Daten	Taste gedrückt	Empfangene Daten
„0“		„8“	
„1“		„9“	
„2“			
„3“			
„4“			
„5“			
„6“			
„7“			

Aufgabe 1: *Baue einen Stromkreis, der es ermöglicht, eine LED über die Fernbedienung ein- und auszuschalten. Schreibe ein passendes Programm.* Speichere das Programm unter dem Namen „IR-Fernbedienung 2“.

Aufgabe 2: *Baue einen Stromkreis, der es ermöglicht, mithilfe der Fernbedienung zwischen 9 LEDs auszuwählen. Schreibe ein passendes Programm.* Speichere das Programm unter dem Namen „IR Fernbedienung 3“.

Aufgabe 3: *Baue einen Stromkreis, der es ermöglicht, einen Motor mithilfe der Fernbedienung ein- und auszuschalten. Schreibe ein passendes Programm.* Speichere das Programm unter dem Namen „IR Fernbedienung 4“.

Aufgabe 4: *Baue einen Stromkreis, der es ermöglicht, einen Motor mithilfe der Fernbedienung schneller oder langsamer drehen zu lassen. Schreibe ein passendes Programm.* Speichere das Programm unter dem Namen „IR Fernbedienung 5“.

(Hilfe: Um die Geschwindigkeit des Motors zu erhöhen, muss man den Wert von speedMotor auf Seite 11 verändern. Dies kann man z.B. machen indem man in einer Funktion folgenden Befehl benutzt: „speedMotor = speedMotor +10;“. Danach muss man daran denken, die neue Geschwindigkeit an den Motor zu schicken mit dem Befehl: analogWrite(enA,speedMotor);)

J. Ferngesteuertes Roboterauto bauen

Jetzt können wir das Roboterauto aus dem Schritt F so anpassen, dass man es über die Infrarot-Fernbedienung steuern kann. Du sollst das Programm so aufbauen, dass der Benutzer zwischen dem Autopiloten aus dem Schritt F und der Fernsteuerung entscheiden kann.
Speichere das Programm unter dem Namen „Roboterauto 4“.