Praktikum Linienroboter

Teil 1: Einführungsaufgaben

Version 1, 06.01.2022

Das Ziel dieses Praktikums ist, einen Roboter, der mit Sensoren einer Linie folgen kann, aufzubauen und zu programmieren. Damit das gut klappt, beginnen wir zunächst mit einigen Einführungsaufgaben, um die Grundlagen der Programmierung und einige Bauteile kennenzulernen.

Diese und die kommenden Anleitungen sollen Sie bei Ihrem Praktikum unterstützen. Falls es zwischendurch Fragen geben sollte oder Sie nur anhand der Anleitung nicht weiterkommen, können Sie uns jeder Zeit ansprechen!

Aufgabe 1: "Hello World“-Programm auf Mikrocontroller

Wir beginnen mit dem Mikrocontroller, der den Roboter am Ende steuern wird. Das ist ein ESP32-Entwicklungsboard.

Der eigentliche Mikrocontroller ist das silberne Quadrat in der oberen Hälfte. Der Rest sind bspw. eine WLAN-Antenne und die Stromversorgung des Mikrocontroller. Solche Mikrocontroller können z.B. für die Auswertung von Sensordaten, die Datenaufnahme oder kleinere Berechnungen eingesetzt werden.

Die Metallstifte an der Seite sind die Anschlüsse für verschiedene Schnittstellen, die der Mikrocontroller unterstützt. Wir nennen sie „Pins“. Welcher Metallstift welche Schnittstelle unterstützt unterscheidet sich, je nachdem welches Entwicklungsboard man verwendet. Wenn Sie also selbst zu Hause mit einem ESP32-Board arbeiten wollen, suchen Sie immer die passende Pin-Belegung („Pinout“). Im Praktikum wird vorgegeben, welcher Pin verwendet werden soll, sodass Sie sich darüber keine Gedanken machen müssen. Der ESP32 kann über verschiedene sogenannte Bus-Schnittstellen kommunizieren, die für uns aber nicht relevant sind. Wir nutzen vor allem die GPIOs. Das steht für General Purpose Input Output, also allgemeine Eingabe und Ausgabe. Eingaben sind z.B. Werte, die Sensoren aufnehmen und die wir im Programm verarbeiten, Ausgaben sind Signale, um bspw. die Motoren ein- und auszuschalten.

Wir starten mit einem sogenannten „Hello World“ Programm. Das ist immer ein möglichst einfaches Programm ohne große Funktionalität, mit dem erst einmal nur getestet wird, ob die Verbindung zu dem Gerät bzw. das Gerät selbst funktioniert. Dafür starten Sie die Arduino IDE aus dem Ordner „arduino-1.8.13-windows-portable“. Dann öffnet sich ein leeres Projekt, dass aus zwei Funktionen besteht. Diese beiden Funktionen sind immer der Grund-Bestandteil aller Projeke:

- ***setup():*** Diese Funktion wird beim Starten des ESP32 einmalig ausgeführt und wird genutzt, um die Ein- und Ausgänge den Anschluss-Pins des ESP32 zuzuordnen oder eine Verbindung zu initialisieren.
- ***loop():*** Diese Funktion wird solange das Board eingeschaltet ist wiederholt und zum Abfragen bzw. Beschalten der Ein- und Ausgänge genutzt.

Schreiben Sie zunächst diesen Programm-Code in die Funktionen:

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

void loop() {
   Serial.println("Hello World");
   delay(2000);
}
```

Verbindung zum ESP32 herstellen

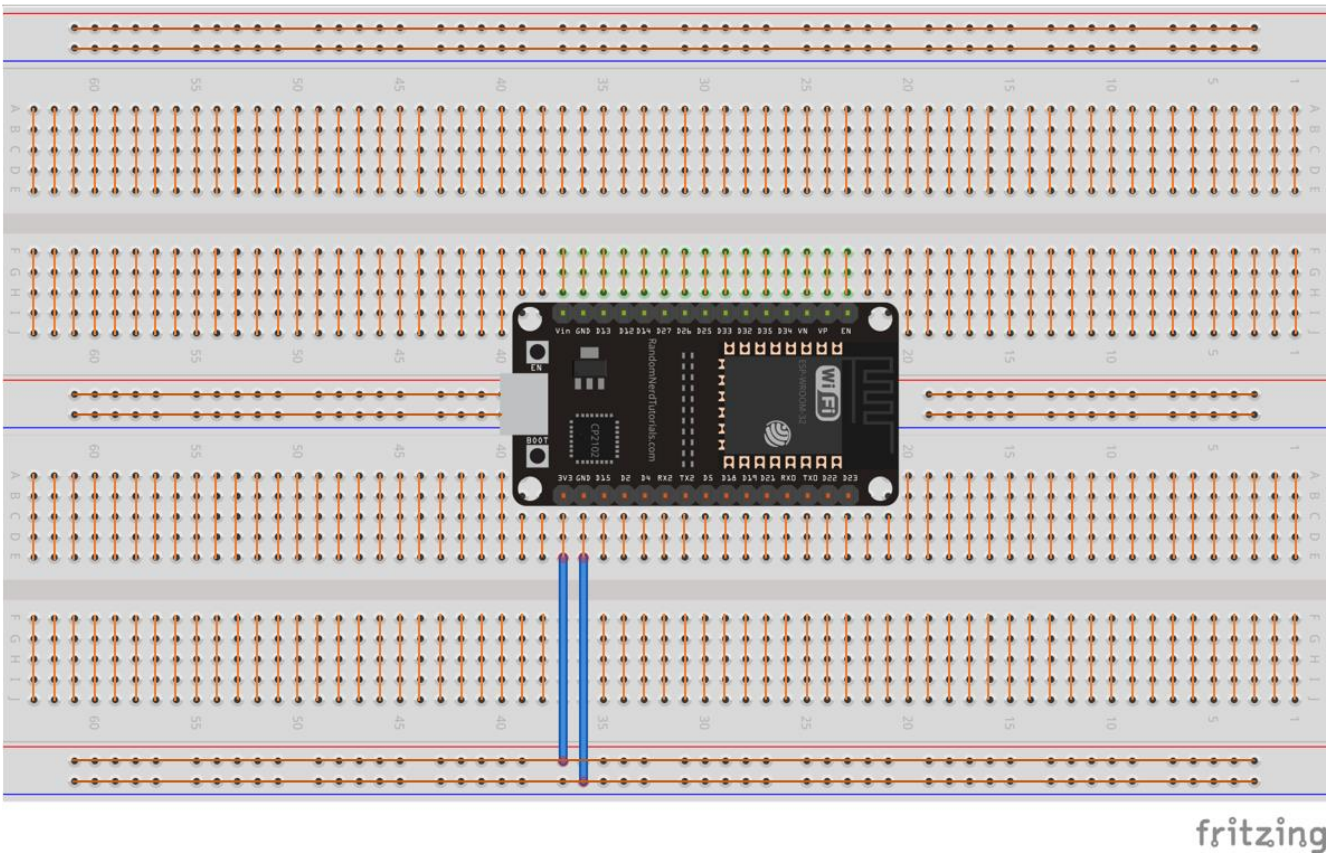
Verbinden Sie den ESP32 mit einem microUSB-Kabel mit Ihrem PC. Das passende ESP32-Board ist in dieser Version der Arduino IDE schon vor-ausgewählt. Stellen Sie als nächstes den Verbindungsport ein. Unter „Werkzeuge > Ports“ werden die möglichen Ports angezeigt. Wenn Sie nicht sicher sind, an welchem Port der ESP32 angeschlossen ist, schauen Sie im Geräte-Manager des PCs. Jetzt können Sie in der Arduino IDE das Programm kompilieren und auf den ESP32 laden. Klicken Sie dafür auf das Pfeil-Symbol.

Wenn das Hochladen abgeschlossen ist, klicken Sie auf „Werkzeuge > Serieller Monitor“. Es öffnet sich ein weiteres Fenster, in dem alle 2 Sekunden der Text „Hello World“ ausgegeben wird. Die Kommunikation zwischen dem ESP32 und dem PC funktioniert also

Dieses neu-geöffnete Fenster ist der Serielle Monitor. Über die Code-Zeile `Serial.begin(9600)` wird eine Verbindung dazu aufgebaut und mit `Serial.println("Hello World")` können dann Text-Zeilen ausgegeben werden. Das können Sie auch später, z.B. für Kontroll-Ausgaben, nutzen.

Für die kommenden Aufgaben nutzen Sie ein Steckbrett, auf dem Sie verschiedene Schaltungen aufbauen. Auf einem solchen Steckbrett sind die Verbindungspunkte immer entweder in einer Zeile oder Spalte miteinander

verbunden, das ist in der folgenden Grafik durch die orangen Linien dargestellt. Stecken Sie den ESP32 wie in der Abbildung dargestellt auf das Steckbrett.



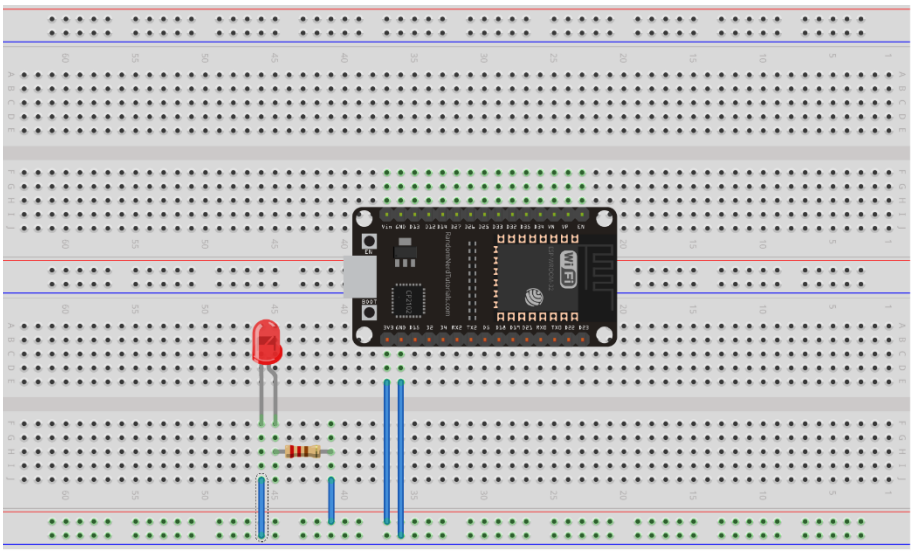
Aufgabe 2: Sperr-/ Durchlassrichtung einer LED

In dieser Aufgabe sollen sie Leuchtdioden (LED) kennenlernen. Wenn durch eine LED Strom fließt, leuchtet sie. Ist der Strom durch die LED zu groß, brennt sie durch. Um das zu verhindern muss immer ein sogenannter Vorwiderstand zwischen der Stromquelle und der LED („vor der LED“) eingebaut werden. Für die folgenden Aufgaben nutzen Sie bitte immer einen 220 Ω Vorwiderstand.

Wenn Sie die LED anschauen, fällt Ihnen auf, dass sie nicht symmetrisch aufgebaut ist, sondern einer der Anschlussdrähte länger ist. Der lange Anschlussdraht ist der Plus-Pol der LED, der kurze der Minus-Pol (an dieser Seite ist auch das Gehäuse der LED abgeflacht, auch daran können Sie die Seiten unterscheiden).

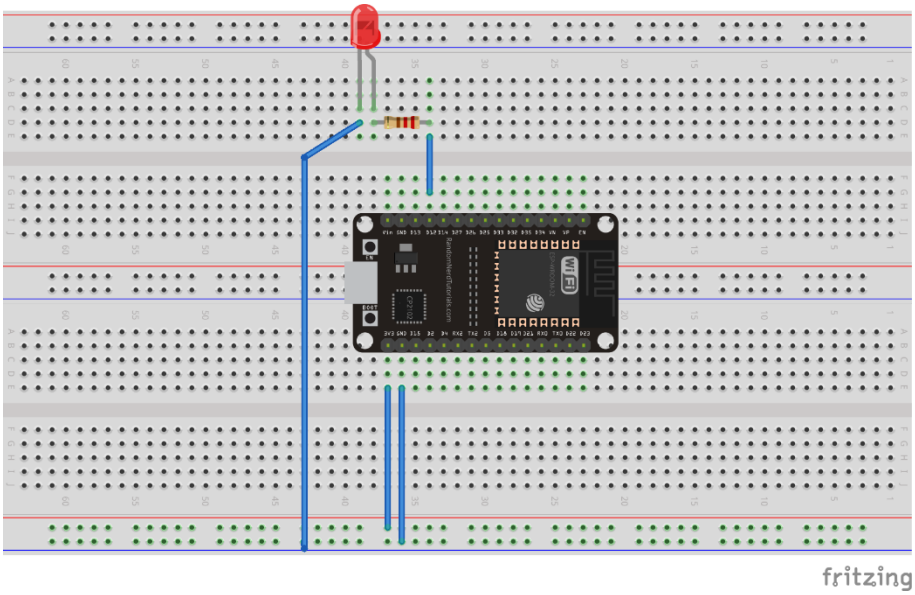
Bauen Sie die Steckbrett-Schaltung der folgenden Abbildung auf. Vom „GND“-Pin des ESP32 gehen Sie mit einer Jumper-Leitung dabei auf die blau markierte Reihe, von dem „3V3“-Pin auf die rot markierte Reihe. Sie können für spätere Schaltungen dann Ground (0V) bzw. 3,3V von diesen Reihen übernehmen.

Leuchtet die LED? Was passiert, wenn Sie sie umdrehen?



Aufgabe 3: LED blinken lassen

Mit dem Aufbau der vorherigen Aufgabe liegt immer Spannung an der LED an und sie leuchtet immer. Wir wollen sie aber selbst ein- und ausschalten können. Dafür nutzen wir einen Ausgangspin des ESP32, den wir programmieren können. Bauen Sie dafür als erstes diese Schaltung auf und öffnen Sie ein neues Projekt in der Arduino IDE:



Der Ausgang, an dem der Widerstand angeschlossen ist, ist mit „D12“ bezeichnet. Das ist der Pin mit der Nummer 12. Damit wir nicht im Programm die Nummer verwenden müssen, um die LED ansteuern zu können, „merken“ wir uns den Pin und definieren dafür eine eigene Bezeichnung. Das macht man durch `#define<Platzhalter> <Wert>`, dass man über die Loop- und Setup-Funktion schreibt. In diesem Fall also:

```
#define LED 12
```

Jetzt müssen wir noch konfigurieren, dass dieser Pin als Ausgang genutzt werden soll. Das muss nur einmal beim Starten des ESP ausgeführt werden und wird daher in die Setup()-Funktion geschrieben. Die Funktion, um einen digitalen Ein- oder Ausgang zu konfigurieren ist `pinMode(<Pin>, <INPUT/OUTPUT>)`. Für alle Stellen im Programm, in denen die Pin Nummer benötigt wird, können wir jetzt den Platzhalter eintragen, also:

```
pinMode(LED, OUTPUT);
```

Damit „weiß“ das Programm, dass dieser Pin, an den die LED über den Vorwiderstand angeschlossen ist, einen Wert High/Low ausgeben soll. Das müssen wir im nächsten Schritt auf diesen Pin schreiben. Das soll immer wieder passieren, muss also in die loop()-Funktion geschrieben werden. Wir wollen ja einen digitalen Wert schreiben auf den Ausgang, genauso heißt auch die Funktion

```
digitalWrite(LED, HIGH);
```

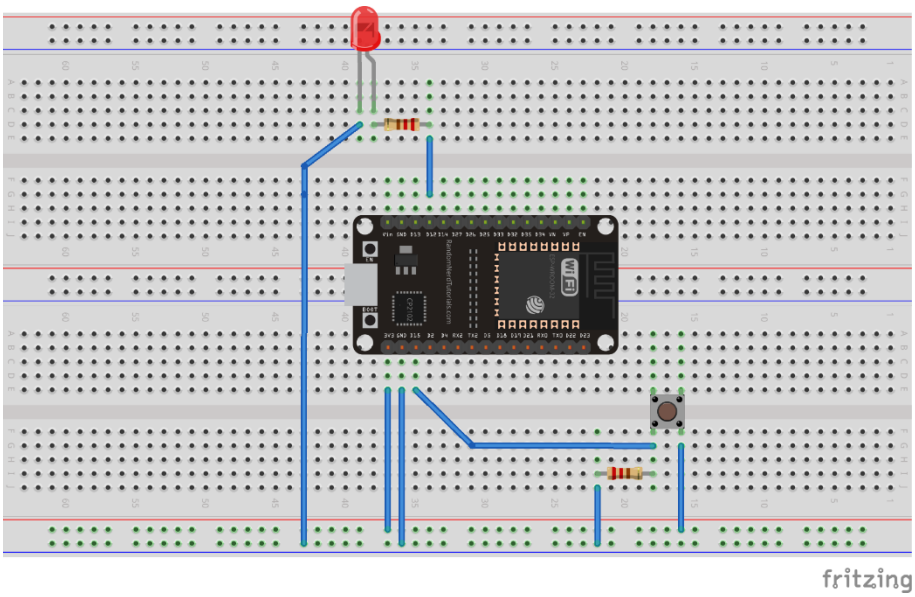
Damit wird die LED eingeschaltet. Um sie wieder auszuschalten, schreibt man `LOW` statt `HIGH`. Eine blinkende LED schaltet immer wieder zwischen an und aus – mit einem zeitlichen Versatz dazwischen. Um eine gewisse Zeit zu warten nutzt man

```
delay(<Zeit in Millisekunden>);
```

Nutzen Sie die Funktionen, die Sie kennengelernt haben und lassen Sie die LED in regelmäßigen Zeitabständen (z.B. 2 Sekunden) blinken.

Aufgabe 4: LED durch Taster steuern

Bisher haben wir einen Ausgang geschrieben. Ebenso wichtig ist es, Eingänge lesen zu können. Das lernen wir als nächstes kennen. Bauen Sie dafür diese Schaltung auf und erweitern Sie ihr Projekt aus der vorherigen Aufgabe.



Die LED ist weiterhin an Pin 12 angeschlossen. Die Konfiguration können wir daher beibehalten. Zusätzlich ist jetzt ein Taster an Pin 15 hinzugekommen. Definieren Sie auch dafür wieder einen Platzhalter und konfigurieren Sie den Eingang.

Jetzt wollen wir lesen, ob der Taster betätigt wird oder nicht. Das wollen wir auch immer wieder machen, das muss also in die loop()-Funktion geschrieben werden. Wir nutzen die Funktion `digitalRead(<Pin>)`. Dafür geben wir an, welchen Pin wir auslesen wollen, die Funktion gibt dann entweder `HIGH` oder `LOW` zurück. Diesen Wert müssen wir also abspeichern und anschließend weiterverwenden. Dafür legen wir eine Variable an. Variablen haben immer einen Typ, einen Namen (Variablen werden immer klein geschrieben) und einen Wert. Dan Anlegen funktioniert nach Folgendem Schema:

```
<Typ> <Name>;
```

Wenn der Wert der Variable am Anfang schon bekannt ist, kann man auch

```
<Typ> <Name> = <Wert>;
```

Schreiben.

Die Variable, die wir anlegen muss vom gleichen Typ wie der Rückgabewert der Funktion sein. Zahlenwerte sind meist sogenannte Integer (`int`), `HIGH/LOW`-Werte sind Boolsche Variablen (`bool`).

Wir können die Variable also zum Beispiel so deklarieren

```
bool taster_wert;
```

Um den Wert des Eingangs zu lesen, schreiben wir also

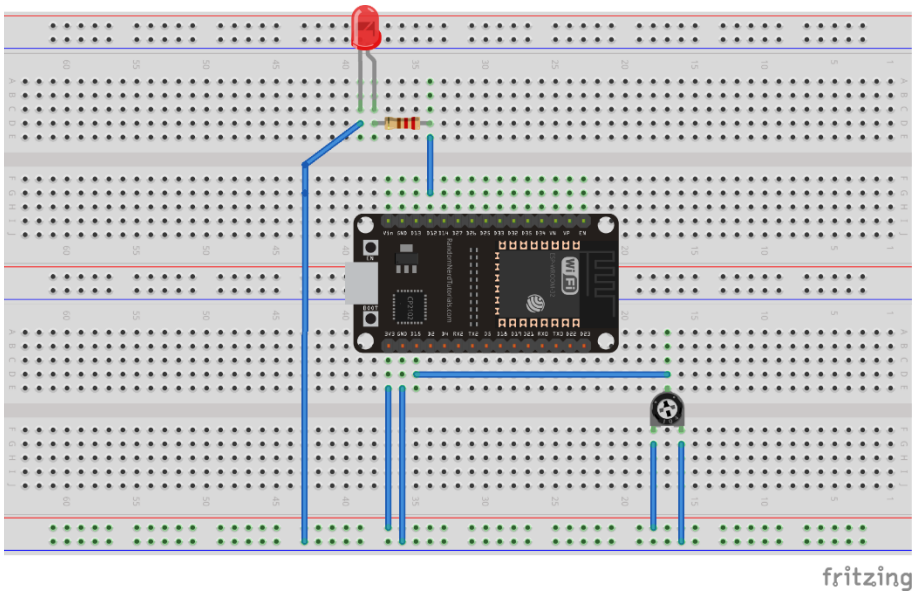
```
taster_wert = digitalRead(<Pin>);
```

Nutzen Sie den Platzhalter, den Sie definiert haben, um den Pin zu lesen. Wie können Sie jetzt den Wert des Tasters mit der LED verbinden?

Aufgabe 5: LED durch Potentiometer dimmen

Für die Programmierung des Roboters brauchen wir später nicht nur digitale Ein- und Ausgänge, sondern auch analoge. Die können nicht nur `HIGH` oder `LOW` sein, sondern mehrere verschiedene (Zahlen-)Werte annehmen.

Um zunächst verschiedene Werte an einem Eingang zu lesen, nutzen wir ein Potentiometer. Das ist ein mechanisch einstellbarer Widerstand. Der Widerstandswert zwischen den beiden äußeren Anschlüssen ist fest. Am mittleren, dritten Anschluss ist der bewegliche Kontakt, dessen Widerstand man durch Drehen verstellen kann. Wenn man an die äußeren Anschlüsse eine Spannung anlegt, erhält man einen einstellbaren Spannungsteiler. Was das bedeutet können wir mit einem Multimeter anschauen. Bauen Sie dafür die abgebildete Schaltung auf. Dann messen Sie mit einem Multimeter zuerst die Spannung zwischen den äußeren Anschlüssen des Potentiometers, die bleibt immer konstant, auch wenn Sie am Potentiometer drehen. Jetzt messen Sie zwischen dem mittleren Anschluss und dem Ground-Anschluss.
Was passiert, wenn Sie am Potentiometer drehen?



Die Gesamtspannung an den äußeren Anschlüssen teilt sich entsprechend des Widerstandsverhältnisses auf. Wenn Sie durch Drehen den Widerstand verändern, verändert sich auch die Spannung, die über dem Teil-Widerstand abfällt. Das können wir auch am Anschluss des ESP32 messen und für unser Programm nutzen.

Die Funktion, die wir dafür nutzen lautet `analogRead(<Pin>)`. Diese liest den analogen Eingang aus, wie bei der vorherigen Aufgabe muss sie den Wert also speichern. Dieser Rückgabewert ist vom Typ `int`.

Jetzt wollen wir durch das Drehen des Potentiometers eine LED mehr oder weniger stark leuchten lassen.
Dafür nutzen wir folgende Funktionen:

```
ledcAttachPin(<Pin>, <Port>);
ledcSetup(<Port>, <Frequenz>, <Auflösung>);
ledcWrite(<Port>, <Wert>);
```

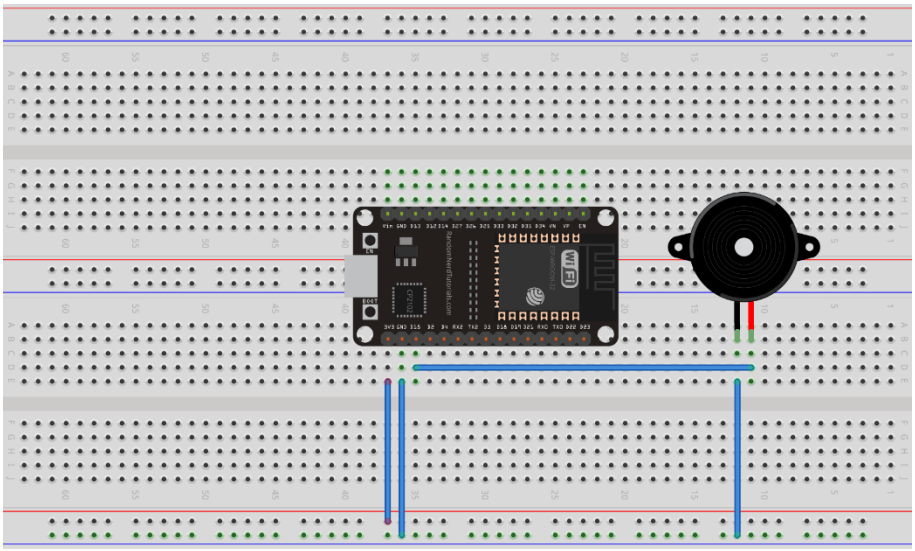
Die ersten beiden Funktionen müssen in der Setup()-Funktion ausgeführt werden, die letzte schreibt den Wert und muss in die Loop()-Funktion. Verwenden Sie für Port „0“, als Frequenz „4000“ und als Auflösung „8“. Probieren Sie den Wert, den Sie am Potentiometer messen zur Ansteuerung der LED zu verwenden.

Aufgabe 6: Beeper ansteuern

Genauso, wie sie in der vorherigen Aufgabe die LED über einen analogen Wert angesteuert haben, können Sie zum Beispiel auch einen „Beeper“, das ist ein kleiner Lautsprecher, ansteuern.

Nutzen Sie die Funktionen, die Sie schon kennengelernt haben. Der Ton verändert sich, wenn Sie eine andere Frequenz nutzen. Probieren Sie verschiedene Frequenzen aus – welcher Ton ist am angenehmsten für Sie? Merken Sie sich diesen Wert, den Beeper brauchen wir später als Hupe für den Roboter!

Tipp: Schreiben Sie `beeper_an = (pow(2, <Auflösung>) - 1);` als Variable in Ihr Programm und nutzen Sie das als <Wert> in der ledcWrite Funktion. Verändern Sie dann nur in verschiedenen Tests die Frequenz.



- Erweiterungen:
    - Frequenz für Beeper in Seriellem Monitor ausgeben
    - LED Schalten nicht tasten

- Cheat Sheet

- Einbinden von Bibliotheken
- `#include<Dateiname>`

- Übersicht der Standardbefehle unter https://www.arduino.cc/reference/de/