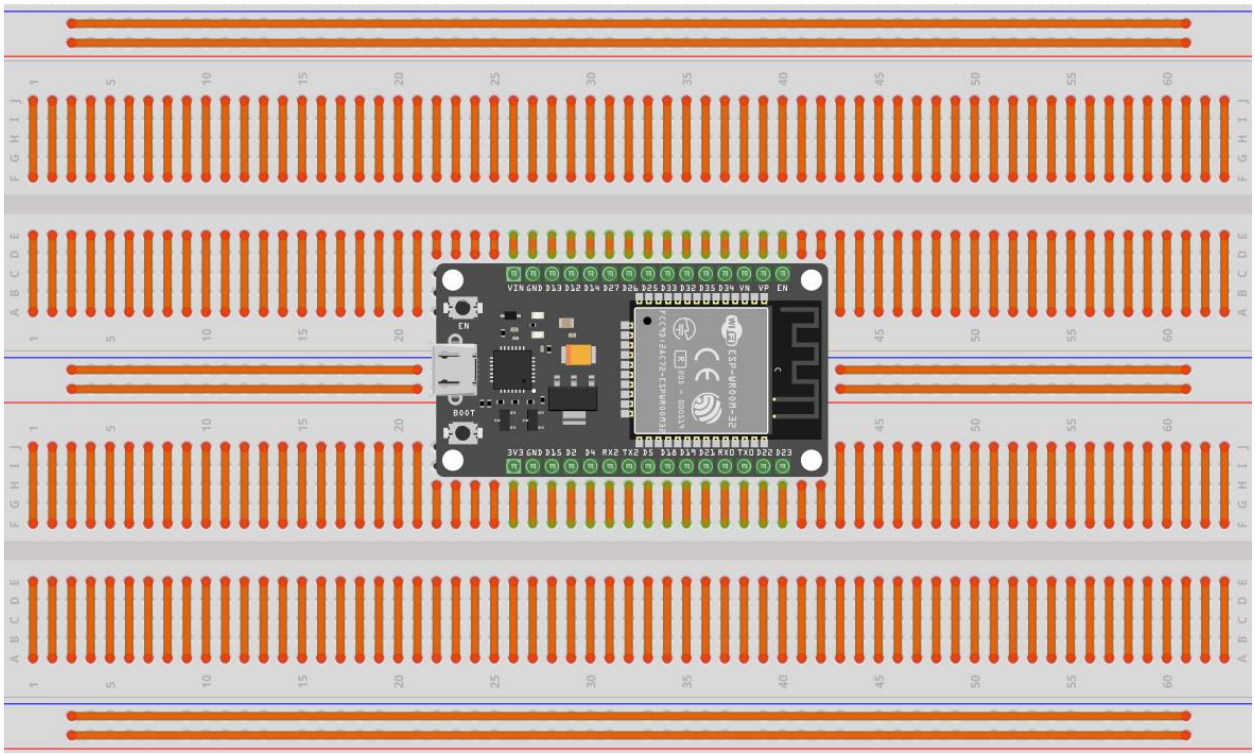
Praktikantenaufgabe “Vier Gewinnt”

Benötigte Materialien:

- ESP32
- Steckbrett
- Jumperkabel
- 220 Ω Widerstände
- LEDs beliebiger Farbe

Aufgabenstellung:

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.

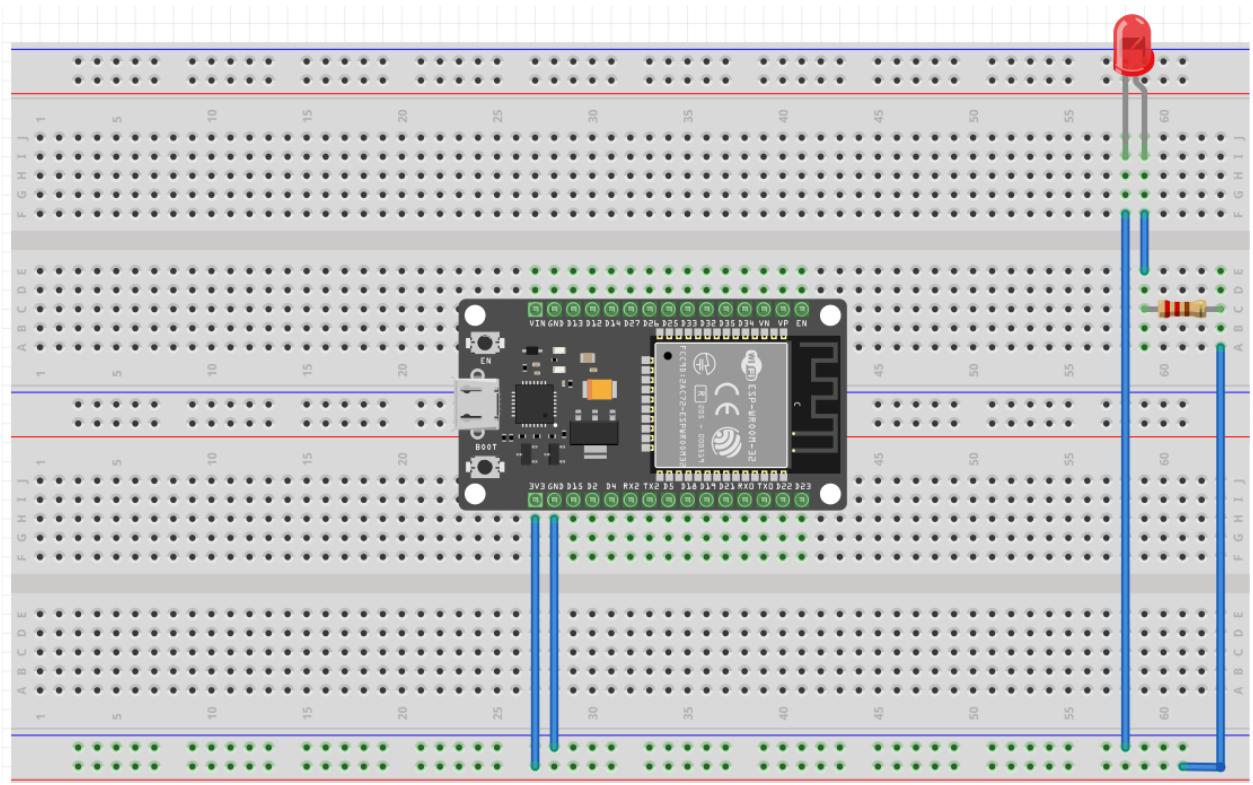


Aufgabe 1: Aufbau einer LED

In dieser Aufgabe sollen Sie Leuchtdioden (LEDs) kennenlernen. Wenn durch eine LED-Strom fließt, leuchtet sie. Ist der Strom 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.

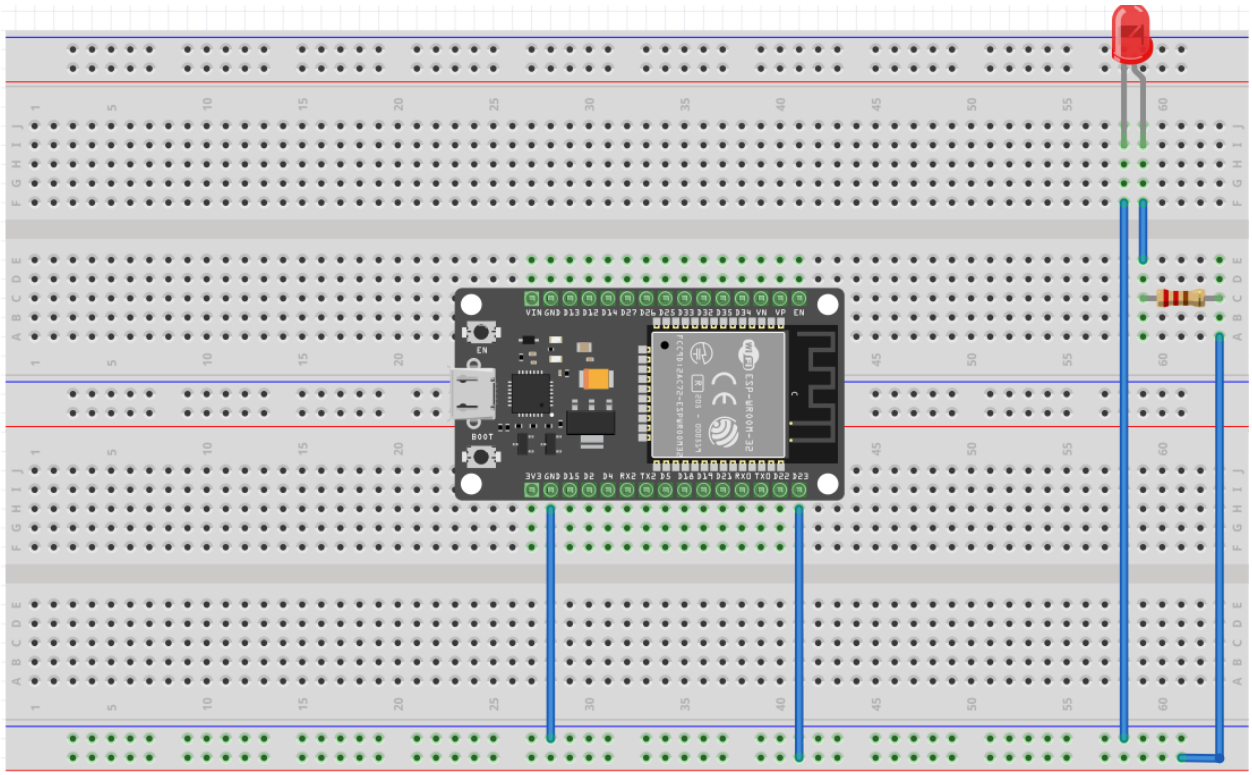
Wenn Sie die LED anschauen, fällt auf, dass sie nicht symmetrisch aufgebaut ist, sondern einer der Anschlussdrähte länger ist. Der lange Anschlussdraht ist der Plus-Pol der LED, der kurze der Minus-Pol (an dieser Seite ist auch das Gehäuse der LED abgeflacht, auch daran können Sie die Seite unterscheiden).

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.



Aufgabe 2: 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 der ESP32, den wir programmieren können. Bauen Sie 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 „D23“ bezeichnet. Das ist der Pin mit der Nummer 23. Damit wir nicht im Programm die Nummer verwenden müssen, um die LED ansteuern zu können, „merken“ wir uns den Pin und definieren dafür eine eigne Bezeichnung. Das macht man durch *#define<Platzhalter> <Wert>*, dass man über die *loop()-* und *setup()-Funktion* schreiben. In diesem Fall also: *#define LED 23*

Jetzt müssen wir noch konfigurieren, dass dieser Pin als Ausgang genutzt werden soll. Das muss nur einmal beim Starten des ESP32 ausgeführt werden und wird daher in der *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 Vorwiederstand 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 einen digitalen Wert schreiben auf den Ausgang, genauso heißt auch die Funktion

digitalWrite(LED, HIGH);

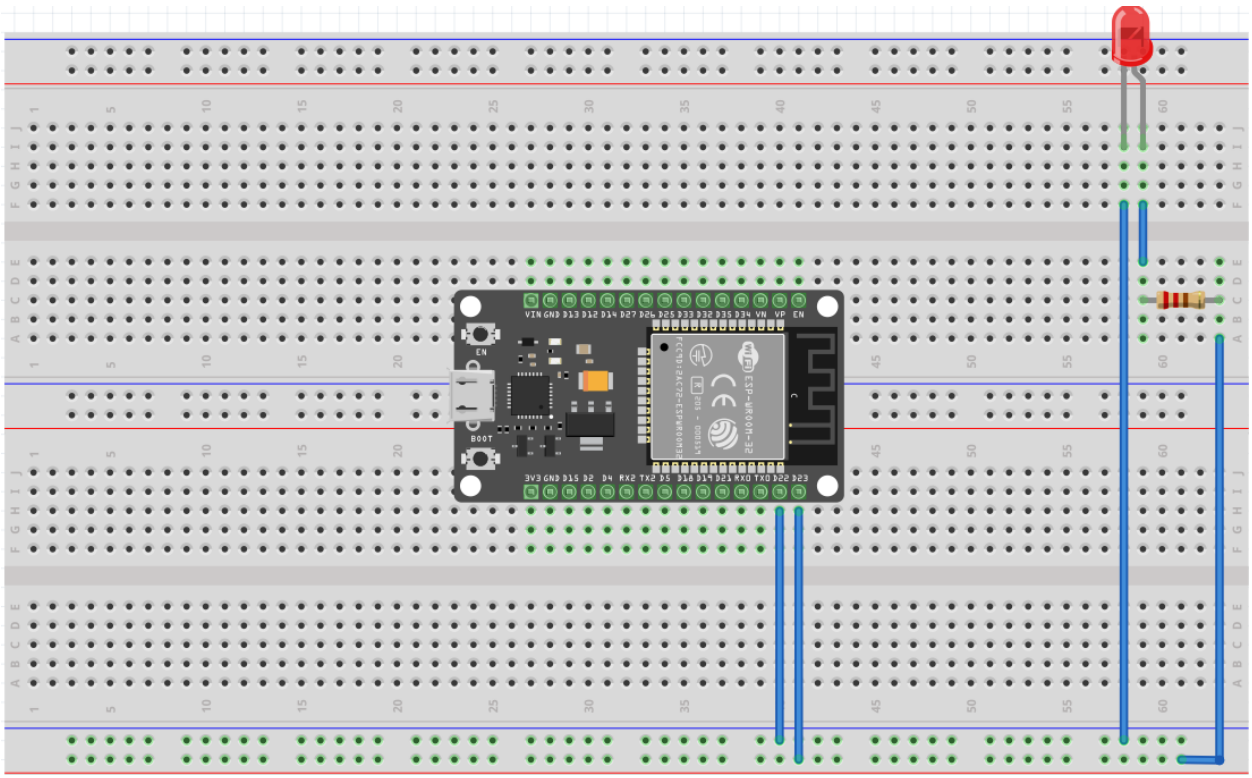
Damit wird die LED eingeschaltet. Um sie wieder auszuschalten, schreibt man *LOW* statt *HIGH*. Eine blinkende LED schaltet immer zwischen an und aus – mit einem zeitlichen Versatz dazwischen. Um eine gewisse Zeit zu warten nutz man

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 3: Masse schalten

So wie man einen Pin des ESP32 als digitalen Ausgang nutzen und programmieren kann, um eine Spannung auszugeben, kann man auch einen Pin dafür verwenden, um Masse zu schalten (also 0V ausgeben). Bauen Sie zuerst die folgende Schaltung auf dem Steckbrett nach:



Die Masse liegt nun auf dem Pin „D22“ diesen können Sie sich wieder in einem Platzhalter abspeichern, um diesen im restlichen Programm zu verwenden. Außerdem muss dieser Pin ebenfalls in der setup()-Funktion als Ausgang konfiguriert werden. Im Anschluss können Sie diesen Pin wieder in der loop()-Funktion programmieren. Um Masse zu schalten, wird dieselbe Funktion, wie zum Ein- und Ausschalten der LED verwendet, da es nichts anderes ist, als eine

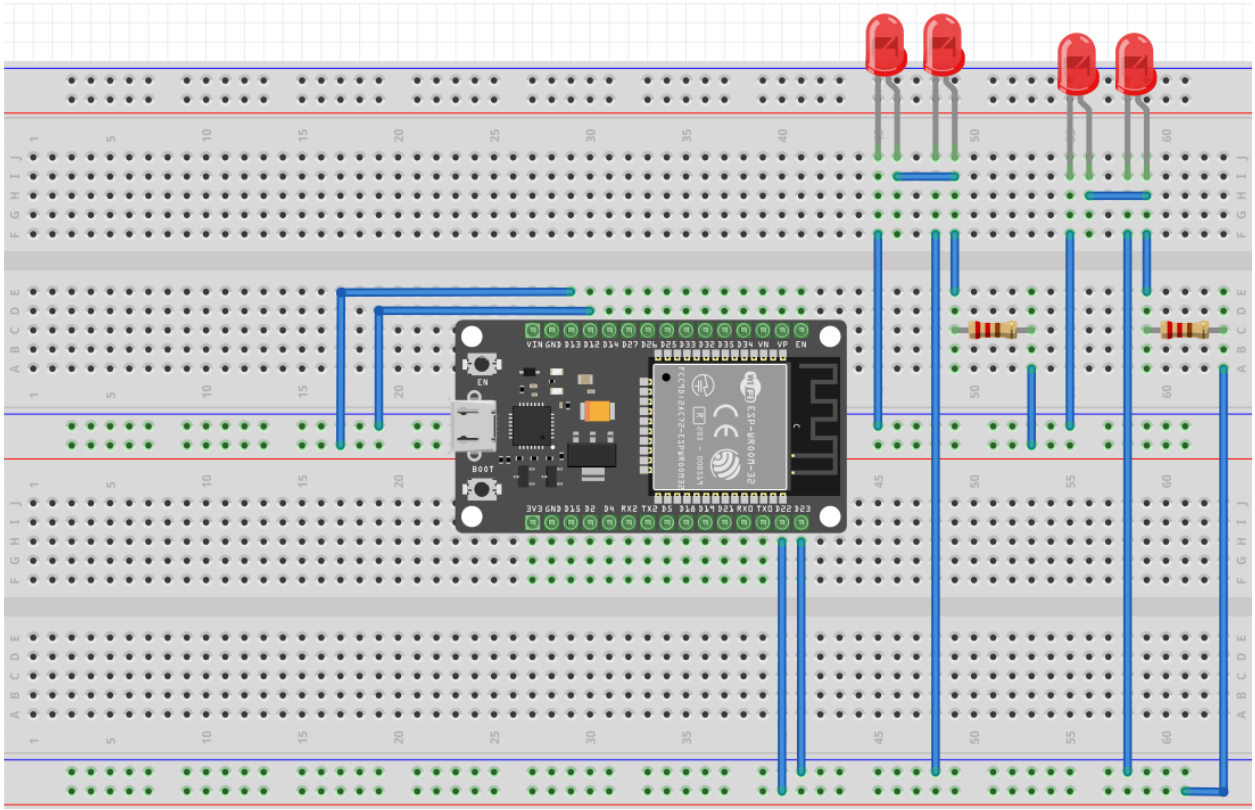
Spannung von 0V (also LED ausschalten) abzugeben. Versuchen Sie die LED wieder zum Blinken zu bekommen, indem Sie über den Pin 23 konstant eine Spannung von 3,3V anlegen und nur über den Masse-Pin 22 die LED steuern.

Aufgabe 4: Eigene LED-Matrix

Da unser 4-Gewinnt Spielfeld aus 6 Spalten und 5 Reihen besteht, benötigen Sie 30 LEDs. So viele programmierbare Ausgänge besitz der ESP32 allerdings nicht. Aus diesem Grund müssen Sie sich mit einem „Trick“ behelfen. Jede LED einer Zeile Ihrer eigenen LED-Matrix wird mit demselben Pin verbunden, der eine Spannung von 3,3V abgibt, und jede LED einer Spalte wird mit demselben Pin verbunden, über den Sie dann für die Spalte Masse schalten können. Somit brauchen wir nur 11 Pins (6 Spalten + 5 Zeilen) anstelle von 30. Wie steuert man nun einzelne LEDs auf der Matrix an?

Das System funktioniert wie ein Graph mit einer x- und einer y-Achse. Möchte man eine bestimmte LED ansteuern schaltet man Masse auf der passenden Spalte (die x Koordinate) und die Spannung auf der Passenden Zeile (die y Koordinate). Alle anderen Spalten und Zeilen werden nicht angesteuert und aus diesem Grund leuchtet nur die gewünschte LED.

Bauen Sie den folgenden Schaltplan auf dem Steckbrett nach. Dieser Aufbau demonstriert das Prinzip auf einer kleinen 2x2 Matrix. Dabei ist die Zeile 1 (das linke LED-Paar) mit dem Pin „D13“ verbunden und die Zeile 2 mit dem Pin „D22“. Außerdem ist die erste Spalte (die beiden linken LEDs der LED-Paare) mit dem gemeinsamen Pin „D12“ und die zweite Spalte mit dem Pin „D23“ verbunden.



Versuchen Sie die LED, die in der ersten Spalte und in der ersten Zeile liegt, einzuschalten, ohne das eine andere LED leuchtet.

Beim 4-Gewinnt Spiel leuchtet allerdings nicht immer nur eine LED, sondern viele LEDs gleichzeitig. Was Passiert, wenn Sie versuchen, zwei diagonale LEDs anzusteuern (z.B. Zeile 1, Spalte 1 und Zeile 2, Spalte 2)?

Wie Sie gemerkt haben, leuchten alle 4 LEDs auf, was auch logisch ist, wenn man näher darüber nachdenkt. Aus diesem Grund benötigen Sie einen zweiten „Trick“. Steuern Sie nun die LEDs spaltenweise und zeitversetzt an. Dadurch verhindern wir das LEDs zeitgleich leuchten, die nicht zusammen leuchten sollten. Diesen Vorgang wiederholen wir in einer so hohen Frequenz, dass unser Auge nicht wahrnimmt, dass die LEDs ständig ein- und ausgeschaltet werden. Schreiben Sie ein Programm, dass die eine LED einschaltet und für einen kurzen Moment (z.B. 1 Millisekunde) leuchten lässt. Im Anschluss soll diese LED wieder ausgeschaltet werden und die diagonal gegenüberliegende LED soll für einen kurzen Moment leuchten. Durch die Arbeitsweise der *loop()-Funktion* wird dieser Vorgang kontinuierlich wiederholt (Tipp: um das Programm zu Pausieren können Sie wieder die Funktion *delay()* verwenden).

Wenn Sie das Programm richtig geschrieben haben, sollten nun nur die beiden LEDs auf einer Diagonalen zeitgleich leuchten. Dies ist aber, wie Sie nun wissen, nur die halbe Wahrheit. Sie können mit der Verzögerung rumexperimentieren und testen mit welcher Frequenz unser Auge die Täuschung bemerkt.