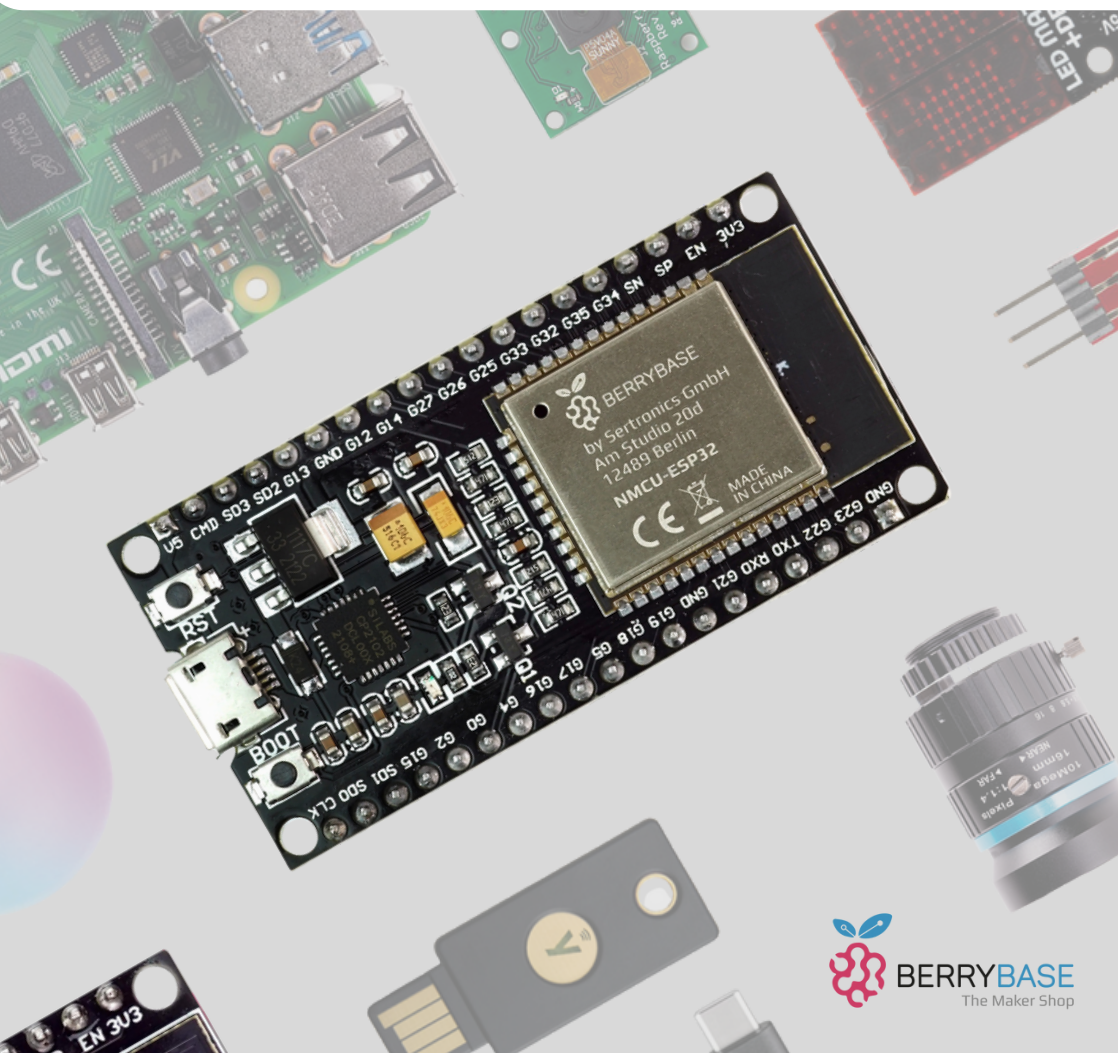
GET YOUR STUFF RUNNING

THE JUMPSTART GUIDE FOR **ESP32 NODEMCU**



BERRYBASE
The Maker Shop

Seite

Hallo und herzlich willkommen zu diesem Guide über das ESP32 NodeMCU Module.

Wir bei der BerryBase freuen uns, dich bei deinem Einstieg in die Welt der Mikrocontroller begleiten zu dürfen. BerryBase ist ein Maker Shop für jeden, mit einer großen Auswahl an Mikrocontrollern und deren Zubehör oder Ressourcen wie diesem Guide, um dir den idealen Start in dein Technik-Projekt zu ermöglichen.

In diesem Guide sollen dir die Grundlagen der Arbeit mit dem ESP32 NodeMCU Module nähergebracht werden. Also keine Scheu: Es ist kein weiteres Wissen vorausgesetzt und der Start beginnt bei Null. Von Grundsätzlichen Informationen über konkretere Schaltpläne, bis hin zu Tipps und Tricks wie die weiter Reise aussehen soll, erfährst du alles hier. Auch fängst du hier direkt mit deinem ersten Projekt an. Aber keine Sorge es ist absolut anfängerfreundlich. Außerdem heißt es nicht umsonst: “Probieren geht über Studieren”.

Daher ist es großartig, dass du dich für dieses Projekt entschieden hast! Lass uns einfach loslegen!

Das ESP32 NodeMCU Module

Das ESP32 NodeMCU Module basiert auf dem beliebten ESP32 Mikrocontroller von Espressif. Zuerst wurde das Module für den Vorgänger ESP8266 entwickelt und dann später auch auf den ESP32 übertragen. Der ESP32 Chip stellt das “Gehirn” des Moduls da und verfügt über genug Leistung für Projekte aller Art. Aufgrund seiner Bauweise und seinem geringen Stromverbrauch eignet sich dieser nicht zuletzt auch für den Batteriebetrieb.

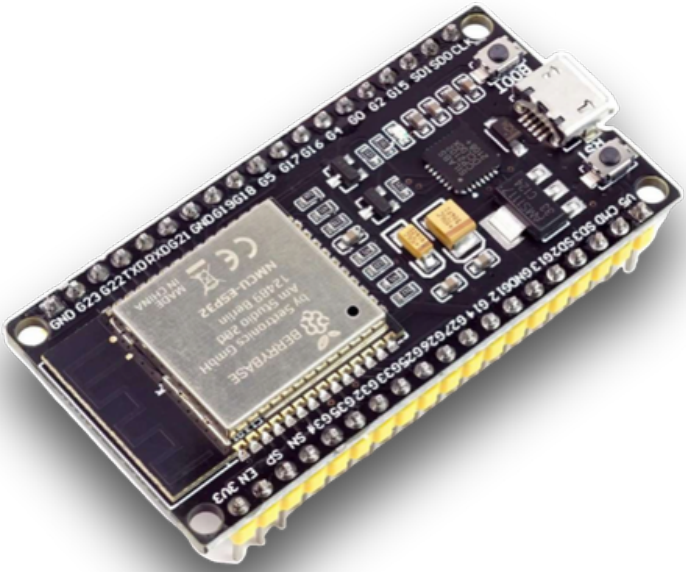


Abb. 1:
Das ESP32
NodeMCU Modul

NodeMCU steht hierbei für die Open Source Firmware und kombiniert “node” und “MCU” (micro-controller unit). Die Einheit ist für die Arbeit mit IoT (Internet of Things) Geräten entwickelt. Mit klassischem Bluetooth, Bluetooth Low Energy (BLE), Wifi und 38 GPIO (General Purpose Input/Output) Pins, die als universelle Ein-/Ausgangsanschlüsse dienen können, kann das Modul mit allen möglichen Smart-Home Geräten kommunizieren. Diese GPIO-Pins ermöglichen die physische Kommunikation und Steuerung zwischen dem Modul und anderen Geräten, indem sie elektrische Signale senden oder empfangen. Die Verbindung kann entweder über eine direkte Verdrahtung zu den GPIO-Pins, wie zum Beispiel bei einer ESP 32 CAM, oder drahtlos über Bluetooth und Wifi hergestellt werden.

Auch steht mit der Firmware Zugriff auf ein API bereit. Mit der API steht ein schnelles und effizientes Interface zur Netzwerkprogrammierung bezüglich des Smart-Home bereit.

Der ESP32 Microcontroller-Chip wurde im Jahr 2016 für die Entwicklung mit SmartHome (bzw. IoT) Geräten vorgestellt und ihm ist seitdem großer Erfolg beschieden. Aus diesem Grund wurden verschiedene Entwicklerboards wie das ESP32 Node MCU Module entwickelt, ESP 32 WROOM, ESP32 Dev Kit C und viel weitere. Alle verfügen über den gleichen ESP32 Chip, aber sie unterscheiden sich in ihren Funktionen (wie auch das NodeMCU API). Auch hat Espressif bereits Nachfolger auf dem Weg gebracht, um mit der Entwicklung der Technik mitzuhalten. So wurden die noch kostengünstigeren und kleineren Chips der C Reihe (z.B. ESP32 C3) vorgestellt, die aber trotzdem eine Vielzahl an Funktionen bereitstellen. Auch die WLAN- und sicherheitsorientierte Variante der S Reihe (z.B. ESP32 S3) befindet sich heute im Programm.

Pinout ESP32 NodeMCU Module

Die Maße vom ESP32 betragen 48mm x 26mm incl. 4 Löchern zum Befestigen an jeder Ecke. Insgesamt verfügt er über 38 Pins.

GPIO: Das ESP32 NodeMCU Module verfügt über 38 GPIO Pins. Diese ermöglichen Input/Output-Signale zum Verbinden verschiedenster Elektronik. Zu beachten ist jedoch, dass einige Pins weitere spezialisierte Funktionen haben oder nur als Eingabepins zur Verfügung stehen.

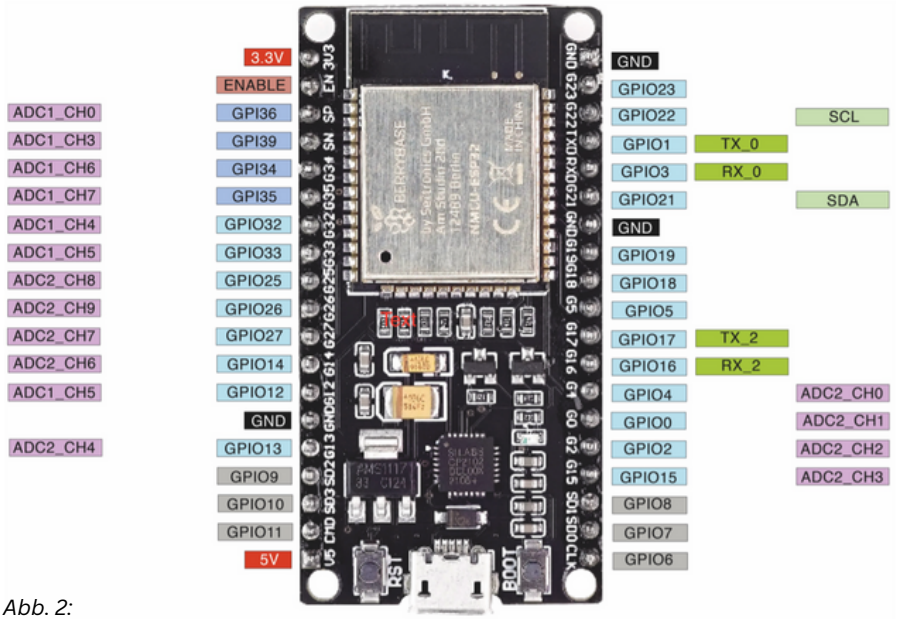


Abb. 2:
PinOut des ESP32 NodeMCU

Power: Strom wird durch den Micro-USB B Stecker oder über Pin 30 (rot VIN) oder Pin 27 (5V; rot) bereitgestellt.

Masse Pins: Die schwarzen Pins (GND) sind Masse Pins für das Anschließen zusätzlicher Elektronik.

Reset: Über den Reset Knopf lässt sich die Schaltung zurücksetzten und der Quellcode wird vom Anfang neugestartet.

Boot: Mit dem “BOOT” Knopf lässt sich der ESP32 manuell in den Flash Mode versetzten. Dies ist notwendig, wenn neue Software auf den ESP32 geladen werden soll.

PWM Pins: Die ESP32 PWM Pins sind mit (~) Symbolen gekennzeichnet. Diese ermöglichen zum Beispiel eine analoge Schaltung über digitale Pins. Weitere Informationen zu PWM findest du bei https://www.exp-tech.de/blog/arduino-tutorial-pulsweitenmodulation-pwm (Arduino Tutorial – Pulsweitenmodulation)

Die beiden **Pins D21 und D22** ermöglichen außerdem die Verbindung mit dem ESP32 NodeMCU über I2C. (SCL/SDA)

Für tiefgründigere Schaltungsinformationen siehe: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (ESP32 Datenblatt/ESP32 Datasheet)

Erste Vorbereitungen

Bevor du nun anfangen kannst, mit dem Arduino zu arbeiten, müssen wir erstmal ein paar Vorbereitungen treffen. Zuerst sind folgende Materialien notwendig:

1. PC/Laptop (Windows/Linux/MacOS)
2. USB Kabel für das ESP32 NodeMCU Module

Als nächstes folgt ein weiterer wichtiger Schritt: Damit der ESP32 weiß, was er machen soll, müssen wir ihm dies mitteilen. Das passiert über die Arduino IDE, welche alle nötigen Funktionen dafür hat. So kannst du den Quellcode mit der IDE schreiben und diesen gleich auf den ESP32 über die eingebaute Upload Funktion übertragen.

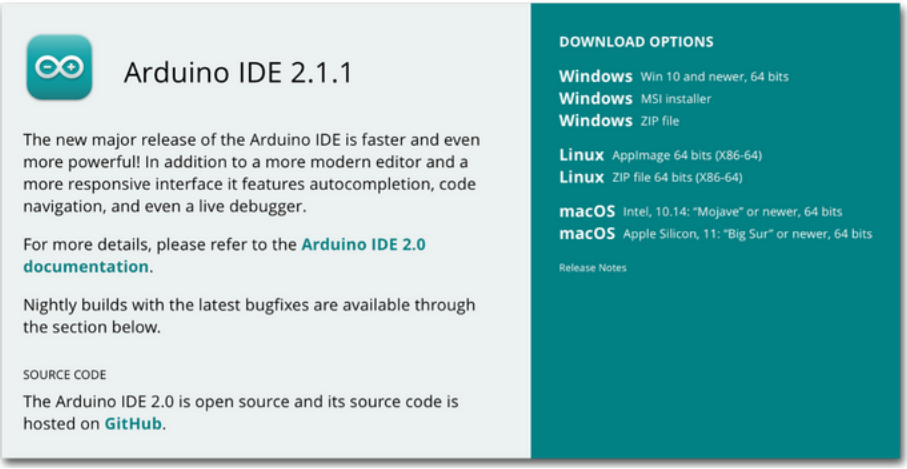


Abb. 3: Die Arduino IDE

Unter diesem Link findest du die Software. Nach dem Aufrufen vom Link, sollte auf der Seite eine Kachel wie diese erscheinen. Die Software heißt “Arduino IDE” und hier in unserem Beispiel ist 2.1.1 die aktuelle Version. Sollte Deine Version eine andere, also eine aktuellere sein, kannst du trotzdem ohne Probleme dem Guide folgen.

Wähle nun dein Betriebssystem aus, um den Download zu starten.

Folge nun den Anweisungen vom Installer, bis die installierte IDE auf dem Bildschirm erscheint.

Mit dieser Entwicklungsumgebung wirst du nun in Zukunft deine Projekte programmieren können. Die Anwendung umfasst auch weitere Funktionen, wie einen Debugger oder ein Terminal, die im Laufe deiner Projekte sehr praktisch werden können.

Erstes Setup

Vor dir solltest du jetzt die Arduino IDE sehen. Über diese Oberfläche wirst du all deine zukünftigen Projekte programmieren und testen können.

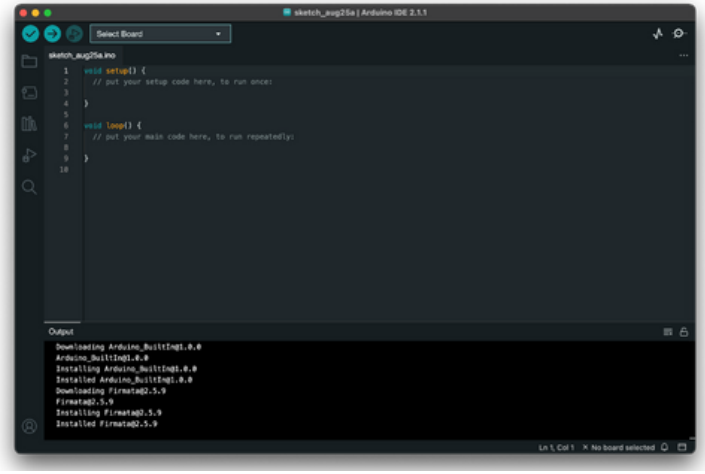


Abb. 4: Arduino IDE im Einsatz

Als erstes verbinde nun den ESP32 mit deinem Rechner über das USB Kabel. Jetzt fehlt nur noch eine kleine Konfiguration in der Software, damit wir loslegen können mit der ersten Inbetriebnahme.

Wähle als Nächstes oben “Select Boards” bzw “Wähle Board” aus. Hier wird das Entwicklerboard festgelegt, mit dem wir arbeiten. Wähle auf der linken Seite also das Board “ESP32 Dev Board” aus und auf der rechten Seite dein ESP32 NodeMCU Module, welches du mit dem PC verbunden hast.

Abb. 5: Arduino IDE Einstieg

Der ESP32 Chip ist nicht von Arduino, daher ist eine zusätzliches Softwareextension notwendig zum Programmieren.

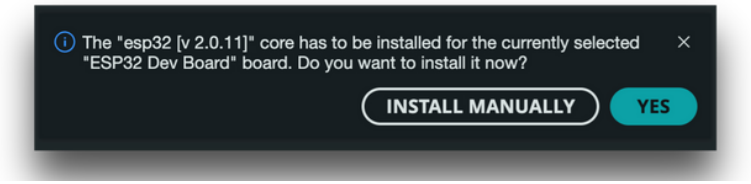


Abb. 6: Warnmeldung. Klicke auf “YES”.

Dafür sollte nun unten rechts eine Abfrage erscheinen, ob du diese installieren möchtest. Dort klicke “Yes” bzw “Ja”, dann wird alles notwendige im Hintergrund automatisch heruntergeladen.

Solltest du die obige Meldung nicht sehen, dann musst du die Softwareextension manuell installieren.

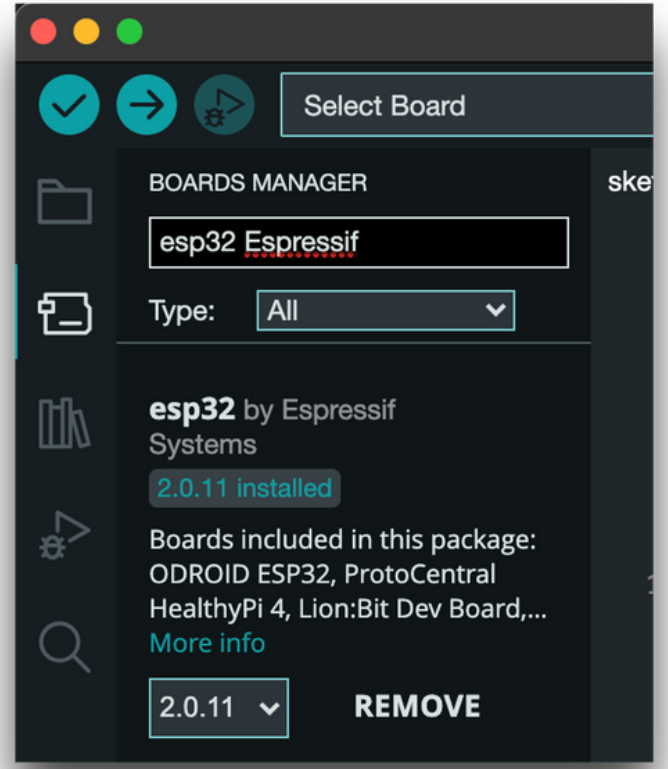


Abb. 7: Screenshot Ardino IDE

Dazu wählst du den “Board Manager” auf der linken Seite aus (weißes Symbol unter dem Dateien-Symbol). Dort gibst du in der Suchleiste “ESP32” ein, denn für diese Boards wollen wir die Extensions laden. Dann installiere die untere Extension (von Espressif selbst). Mit dieser kannst du nun für den ESP32, ESP32-S2, ESP32-S3 und ESP-C3 entwickeln mit der Arduino IDE.

Es sollte sich im unteren Teil das Terminal öffnen, wo du sehen kannst, wie die notwendigen Ressourcen heruntergeladen werden.

In der Mitte der Arduino IDE solltest du den Quellcode mit den zwei “Funktionen” sehen. Alles, was nun in den Klammern {} bei “Setup” steht wird einmal ausgeführt, wenn das Programm gestartet wird. Die Funktion “loop” verhält sich schleifenartig. Das heißt der Quellcode in den Klammern {} dieser Funktion wird wie in einer Schleife ausgeführt, bis das Programm endet.

Sobald du deinen Quellcode dann fertig geschrieben hast, kannst du durch das Klicken des “blauen Hakens” überprüfen, ob du Fehler gemacht hast. Sollte aber noch ein Fehler im Quellcode sein, taucht nun unten auf dem Bildschirm eine Fehlermeldung in rot auf.
Falls das jedoch nicht der Fall ist, dann kannst du nun deinen Quellcode auf den Arduino laden. Drücke hierzu einfach auf den Knopf mit dem Pfeil nach rechts: Der Quellcode wird nun auf den Arduino hochgeladen und ausgeführt.

Erste Inbetriebnahme

Nun geht es darum zu testen, ob du alles richtig eingestellt hast. Hierzu wollen wir nun eine LED zum Blinken bringen. So können wir gleich überprüfen, ob alles richtig läuft während du dich gleich mit dem Ablauf vertraut machst. Viele der folgenden Schritte wirst du immer brauchen, sobald du Mikrocontroller aller Art programmierst.

Hierzu benötigst du: 2x Jumper-Kabel, 1x LED und 1x Schichtwiderstand (wie unten auf der Abbildung).

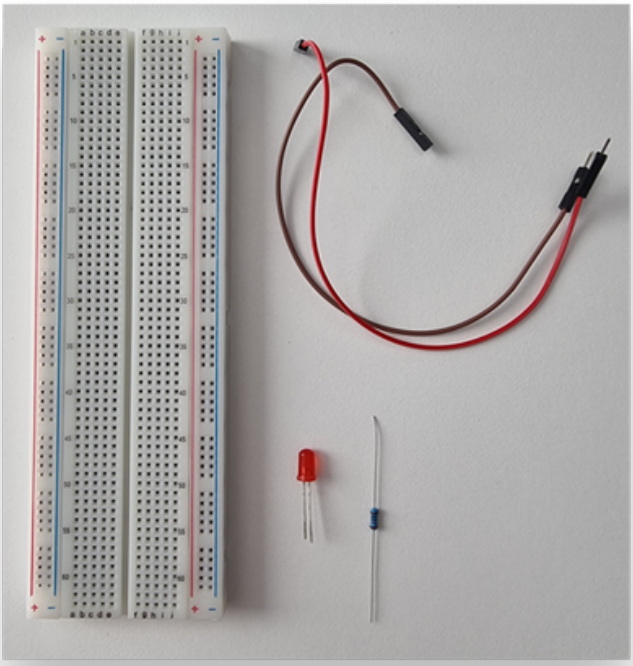
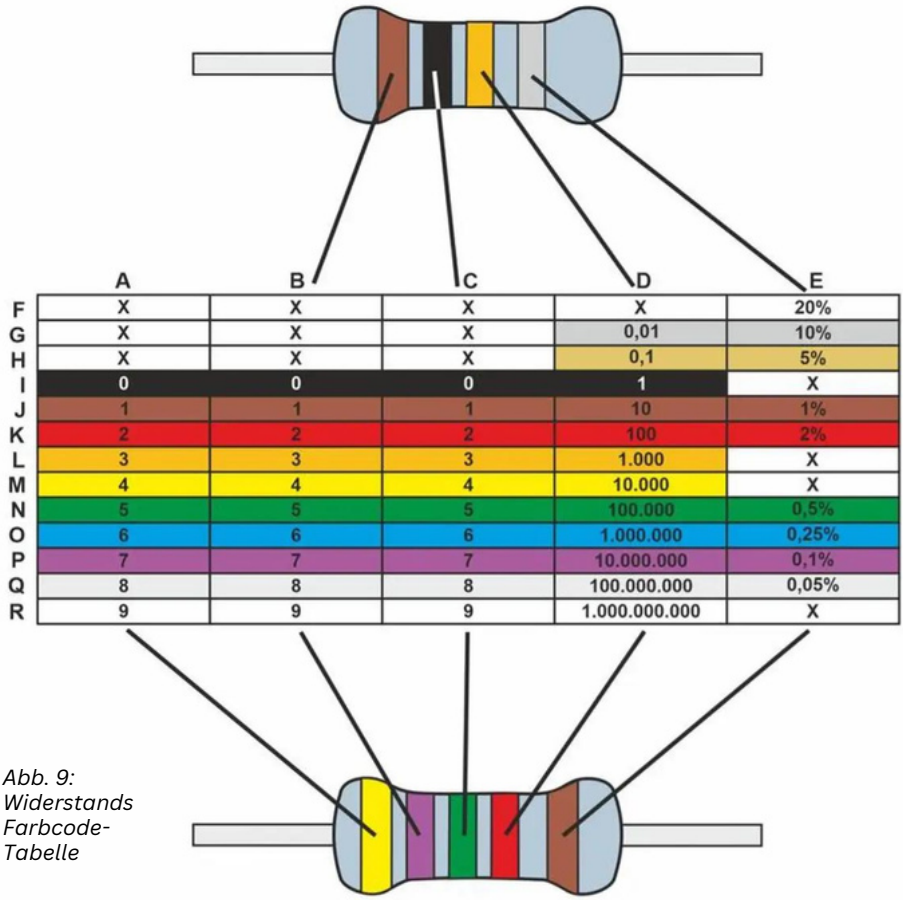


Abbildung 7: Der Calliope Simulator im MakeCode Editor

Der Wert von Widerständen wird in Ohm (Ω) angegeben. Dieser muss dann passend zur LED gewählt werden. Bei den Starter Kits sind meistens kleine Anleitungen beigelegt, welche dir die richtigen Werte zeigen. Meistens sollte der Widerstandswert zwischen 200Ω-500Ω liegen. Wie viel Ohm (Ω) ein Widerstand hat, kannst du mithilfe der Tabelle in der Grafik und anhand der farbigen Striche auf dem Widerstand erkennen. Benutze den oberen Teil bei vier Strichen auf dem Widerstand und den unteren bei 5 Strichen. Die ersten 3 bzw. 4 Striche sind dafür gedacht, den Wert des Widerstandes zu bestimmen.



	A	B	C	D	E
F	X	X	X	X	20%
G	X	X	X	0,01	10%
H	X	X	X	0,1	5%
I	0	0	0	1	X
J	1	1	1	10	1%
K	2	2	2	100	2%
L	3	3	3	1.000	X
M	4	4	4	10.000	X
N	5	5	5	100.000	0,5%
O	6	6	6	1.000.000	0,25%
P	7	7	7	10.000.000	0,1%
Q	8	8	8	100.000.000	0,05%
R	9	9	9	1.000.000.000	X

Abb. 9: Widerstands Farbcode-Tabelle

Nehme die Werte von (A bei 5 strichen),B und C und multipliziere diesen Wert dann mit D.

Die einzelnen Löcher deines Breadboards sind nach dem folgenden Muster miteinander verbunden (rote Markierungen in Grafik). Die Einkerbung in der Mitte trennt a-e und f-j voneinander.

Die Streifen sind jeweils horizontal miteinander verbunden. Während an den beiden äußeren Seiten Plus (+) und Minus (-) der Versorgungsspannung vertikal miteinander verbunden sind.

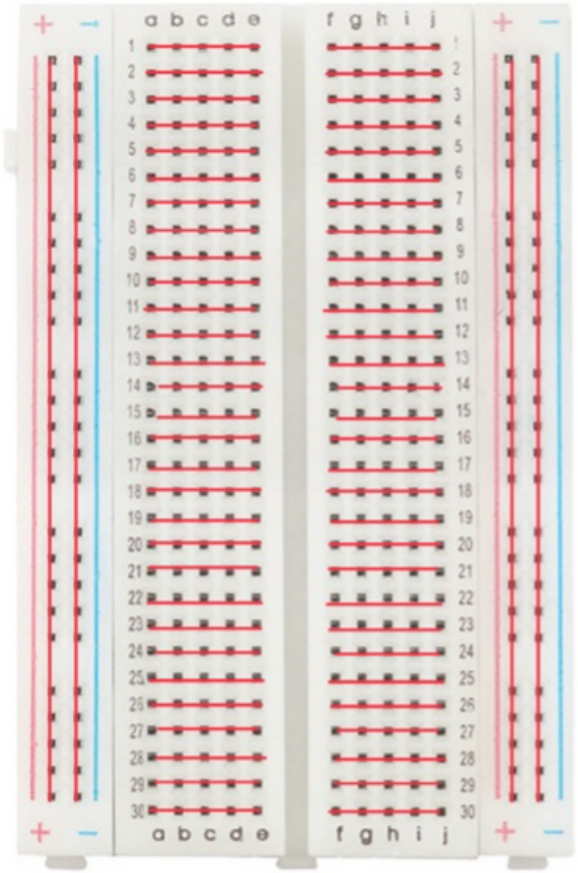


Abb. 10: Aufbau eines Breadboards

Die LED hat eine Kathode und eine Anode, was entscheidend für die Beschaltung ist. Beim falschen Anschließen kann die LED beschädigt werden. Die kürzere Seite der beiden LED-Kontakte ist die Kathode und muss mit dem Ground (GND) verbunden werden. Der Widerstand ist hierbei vor die LED geschaltet, damit diese nicht, aufgrund zu hoher Spannung, durchbrennt.

Stecke den Widerstand, die LED und die Kabel wie im Bild in das Breadboard. Wo auf dem Breadboard ist egal, Hauptsache du berücksichtigst die Verkabelung dieses wie zu vor beschrieben. Die beiden Kabel sind farbig makiert zur Unterscheidung. Bei der LED ist die Seite mit dem „Knick“ die Anode. Das schwarze Rechteck mit dem R ist das Symbol für den Schichtwiderstand.

Die dünnen roten Linien sind eine rein visuelle Hilfe, um noch einmal die Verkabelung des Breadboards zu zeigen.

Verbinde nun das braune Kabel mit einem Ground (GND) Pin des ESP32 NodeMCU Modules und das rote Kabel mit dem G4 Pin.

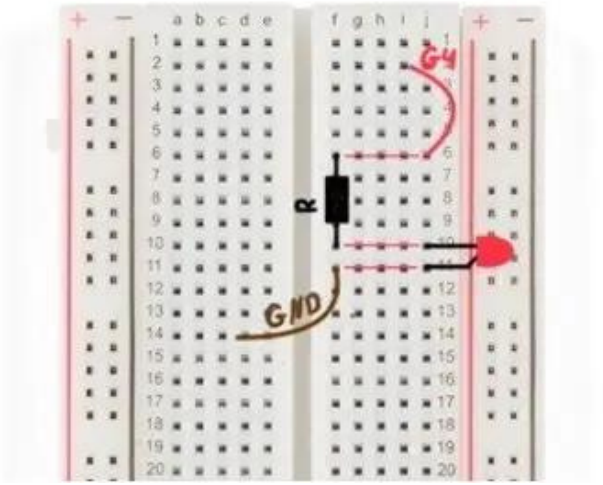


Abb. 11: Planung des Aufbaus

Als Ergebnis sollte die Schaltung ähnlich wie im Bild (Aufbau) aussehen. Wo genau du die Schaltung auf dem Breadboard platzierst ist nicht wichtig. Beachte aber, dass die Kabel und LED richtig verbunden sind. (siehe Breadboard Verkabelung)

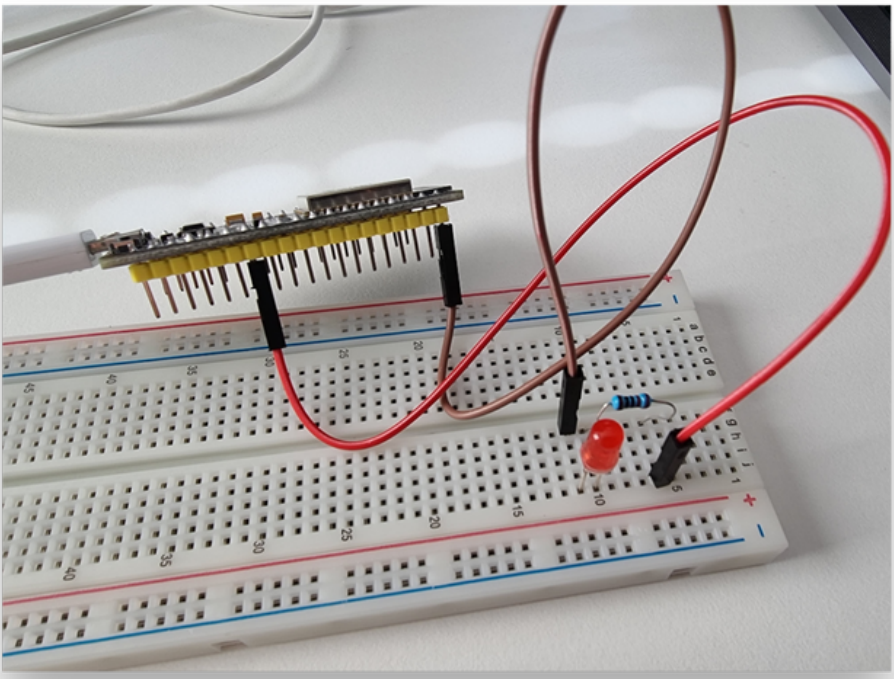


Abb. 12: Unser Versuchsaufbau mit dem ESP32 NodeMCU

Damit die LED nun leuchten kann, müssen wir jetzt die Logik dafür im Quelltext schreiben. Diese sieht dann so aus:

Zuerst definieren wir die LED Nr. 4 und setzten den Modus des Pins auf Ausgabe. Die Loop Funktion wird nun in einer Schleife ausgeführt. In unserem Kontext bedeutet „HIGH“ Strom fließt und „LOW“ bedeutet kein Strom fließt. Jetzt setzten wir den Pin G4, der als LED definiert ist, auf „HIGH“. Danach integrieren wir eine Verzögerung von 2 Sekunden (2.000 Millisekunden), um das Blinken zu erzeugen. Nun setzten wir die LED wieder auf „LOW“. Dieser Vorgang wiederholt sich nun und unsere LED blinkt durchgängig (zusätzliche Kommentare in grau im).

```
#include <Arduino.h>

// definiere welcher Pin genutzt wurde
// (G4 in unserem Fall)
#define LED 4

// wird einmal ausgeführt
void setup() {
  Serial.begin(115200);
  // setzte den Modus auf Ausgabe,
  // damit Strom ausgegeben werden kann um die LED einzuschalten
  pinMode(LED, OUTPUT);
}

// wird in einer Schleife ausgeführt
void loop() {
  // HIGH = Strom
  // schalte den Strom auf Pin G4 ein
  digitalWrite(LED, HIGH);

  // warte 2 Sekunden
  delay(2000);

  // LOW = kein Strom
  // schalte den Strom auf Pin G4 aus
  digitalWrite(LED, LOW);

  // warte 2 Sekunden
  delay(2000);
}
```

Abb. 13: Quellode zum LED blinken lassen

Fehlerbehebung

Funktioniert der Upload nicht?
Stelle sicher, dass du das Node MCU Module richtig verbunden hast und bei der Auswahl vom Board/Port alles stimmt. Checke auch, ob die richtigen Softwareextensions für das Node MCU Module heruntergeladen sind.

Fehler im Quellcode?
Stell sicher, dass du an alle Klammern "{}" und Semikolons ";" gedacht hast. Kontrolliere auch auf Rechtschreibfehler im Quelltext. Vergiss auch nicht „LOW" und „HIGH" richtig zu benutzen.

LED blinkt nicht?
Überprüfe, ob der Quellcode richtig und ohne Fehler auf das Module hochgeladen wurde (ohne Fehlermeldung; – eine LED blinkt meistens am Module während des Hochladens). Überprüfe außerdem, ob die Kabel mit den richtigen Pins verbunden sind: Das müssen GND und G4 sein! Kontrolliere auch, ob die Verkabelung auf dem Breadboard stimmt, d.h. Anode und Kathode der LED richtig verkabelt sind und ein passender Widerstand gewählt wurde und die Komponenten richtig auf dem Breadboard verbunden sind (siehe Breadboard Grafik).

Sollte es trotzdem nicht funktionieren, kannst du dich an Foren für Hilfe wenden. In der nächsten Sektion "Wie geht es nun weiter?" kannst du unter dem Punkt "Funktioniert etwas nicht?" weitere Links zu Hilfeforen finden.

Wie geht es nun weiter?

Du kennst dich jetzt mit den grundlegenden Funktionen vom ESP32 NodeMCU Modul aus und kannst diese an deinen Rechner anschließen. Auch kennst du den grundlegenden Aufbau der Arduino IDE zum Programmieren des Quellcodes. Darüber hinaus hast du dich mit dem Aufbau des ESP32 Node MCU auseinandergesetzt und den Unterschieden zu weiteren Modellen gelernt.

Mit den Grundlagen im Hinterkopf bist du jetzt bereit auf eigene Faust zu experimentieren. Probiere einfach mal all dein Zubehör aus, was du eventuell im Starter-Kit mitbekommen hast. Auf dieser Basis lassen sich dann zusätzliche, komplexere Projekte aufbauen, bei denen nur deine Kreativität Grenzen setzt. Dies könnte zum Beispiel ein Musiksensitiver LED-Strip, ein Lügendetektor oder ein Thermometer mit Anzeige sein. Wenn du dir mehr zutraust, wären auch Projekte wie ein Bewässerungssystem oder eine Wetterstation für dein Smart Home möglich. Auf ein fröhliches Experimentieren!

Nützlich hierfür können auch weitere Ressourcen sein:

ESP32 Espressi: https://www.espressif.com/en/products/socs/esp32 (Englisch)

Tutorials Einsteiger Projekte: https://arduino-de.site/projekte/ (Einsteiger & Deutsch/ für den Arduino verfasst, lassen sich aber auch mit dem NodeMCU Module realisieren)

Tutorials für Projekte: https://www.instructables.com/search/?q=ESP32+&projects=all (hauptsächlich Englisch)

Einstieg programmieren mit C für die Arduino IDE

Sprache Referenz Arduino IDE: https://www.arduino.cc/reference/de/

Einstieg in C: http://www.atlas.uni-wuppertal.de/~kind/Einstieg_in_C.pdf

ESP32 Smart-Home Projekte

Entwickle mit Apple HomeKit: https://github.com/espressif/esp-homekit-sdk (fortgeschritten/Englisch)

Entwickle mit Matter im Smart Home:

https://www.espressif.com/solutions/device-connectivity/esp-matter-solution#sdk-for-matter (fortgeschritten/Englisch)

Funktioniert etwas nicht?

Arduino Help Center: https://support.arduino.cc/hc/en-us (Englisch)

Arduino Forum: https://forum.arduino.cc/t/wie-man-dieses-forum-benutzt-bitte-lesen/902274