

Projekt LEDitGrow

LED Beleuchtung mit Farbwechsel – Einsatz in der Algen-Kultivierung

Welche Lichtfarbe mögen Algen am liebsten? Um das herauszufinden, haben wir ein programmierbares leuchtendes Plexiglas für Algen konstruiert. In unserer Forschung können wir so die optimale Lichtfarbe für die Algenkultivierung in Bioreaktoren definieren, um möglichst viele Algen in kurzer Zeit zu produzieren.

In unserem Mitmach-Experiment bauen wir die zugrunde liegende LED-Beleuchtung nach. Diese kann mit nach Hause genommen und für die Beleuchtung von Pflanzen eingesetzt werden.

Viel Spaß beim ausprobieren!

Beutelinhalt

Folgendes solltet Ihr in euren To-Go-Beuteln finden:

1. ESP32-WROOM-32 Mikrokontroller inkl. Steckbrett
2. LED – RING
3. QR-Code für die Anleitung

Lest euch die **To-Do-Liste**, **Bauteilbeschreibung** und das **Programm LEDitGrow** durch und versucht, die Zeilen nachzuvollziehen. Der Text ist in folgender Reihenfolge gebaut:

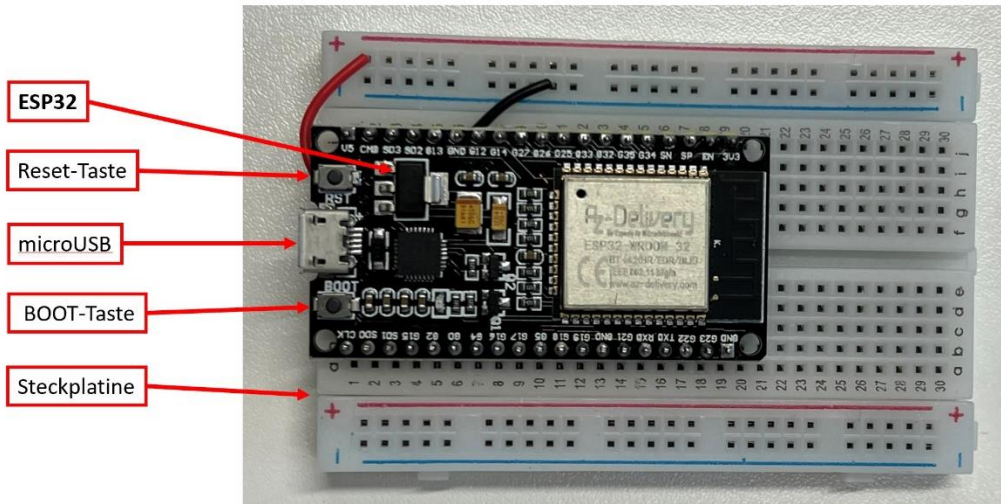
1. To-Do-Liste
2. Bauteilbeschreibung
3. Programm LEDitGrow

Bauteilbeschreibung

ESP32-WROOM-32 Mikrokontroller

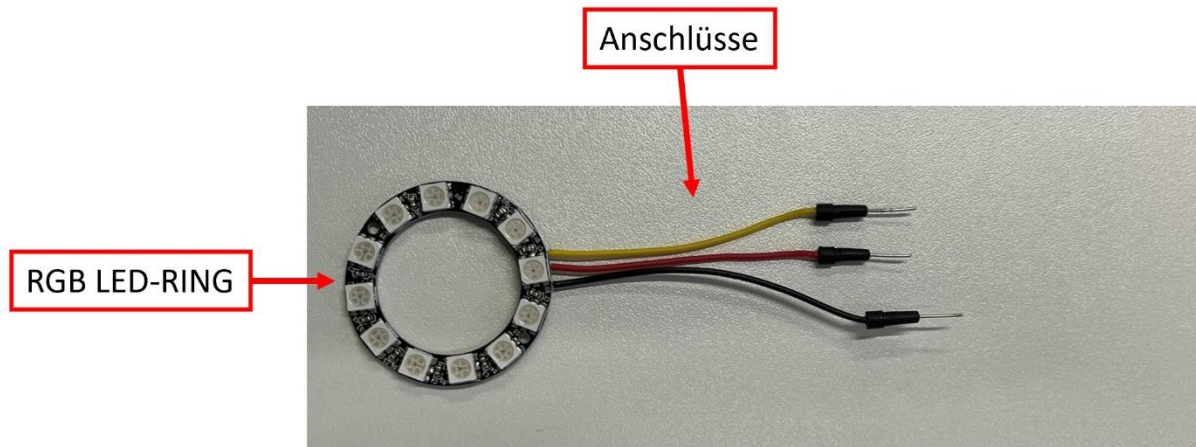
Das ESP32 Dev Kit C V2 ist ein Entwicklungs-Board, das um den **ESP32** WROOM32 Chip herum entwickelt wurde. Es enthält einen Spannungsregler, einen USB Programmierschaltkreis für den ESP32-Chip, einen **microUSB-Anschluss** der eine USB Kommunikation ermöglicht sowie einige andere Funktionen.

Das Entwicklungs-Board ist speziell für die Arbeit auf der **Steckplatine** konzipiert.

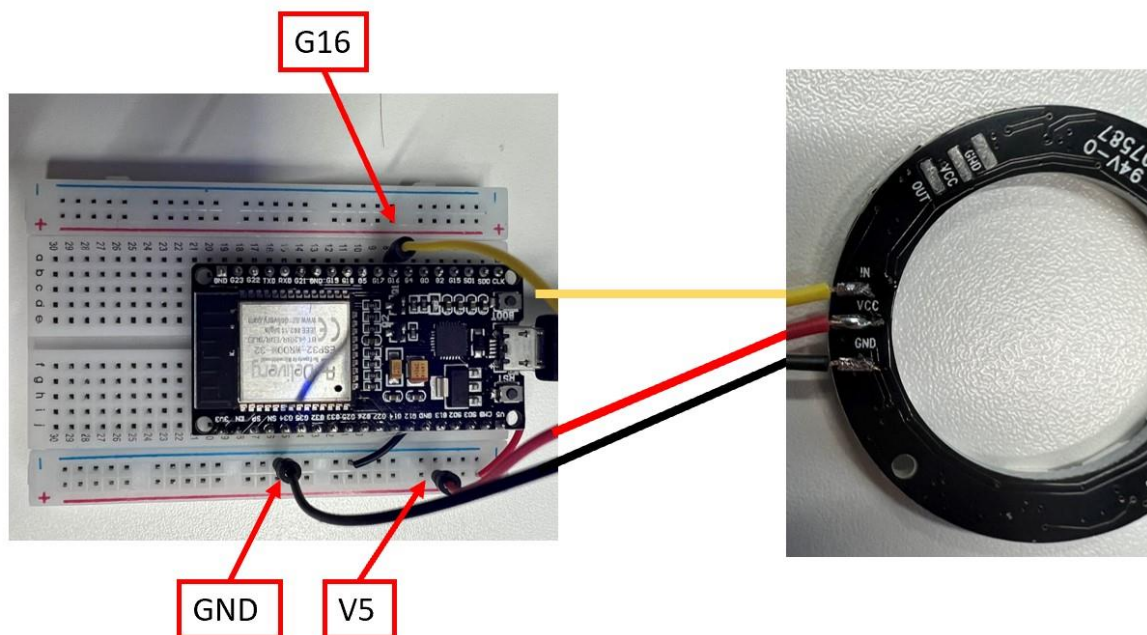


RGB LED RING (WS2812B)

Die LEDs sind adressierbare RGB-LEDs, die in Reihe geschaltet und mit 3 Kabeln verdrahtet sind.



Die Verkabelungen des RGB-LED-Rings mit dem ESP32 ist simpel. Man muss über den 5V Pin an der Steckplatine das VCC-Kabel (rot) anlegen, GND-Kabel (schwarz) zu GND und das IN-Kabel (gelb) zu G16. Das nächste Bild zeigt die Verkabelung und muss genauso gesteckt werden.



Anschließend verbindet ihr den ESP32 über mircoUSB an Laptop oder PC.

To-do-Liste

Bevor der Code zusammen mit dem ESP32 verwendet werden kann, muss die Software Arduino IDE installiert werden

Projekt „ARDUINO-IDE“ installieren und anlegen

1. Installiert die aktuelle Arduino IDE-Software von arduino.cc/en/Main/Software

Stellt vor Beginn dieses Installationsvorgangs sicher, dass die derzeit aktuelle Software ARDUINO IDE auf dem Computer installiert ist. Fall dies nicht der Fall ist, deinstalliere es und installiere es erneut. Andernfalls funktioniert es möglicherweise nicht.

Folgende Anweisungen helfen nach Installation, um das ESP32-Board in Arduino IDE verwenden zu können.

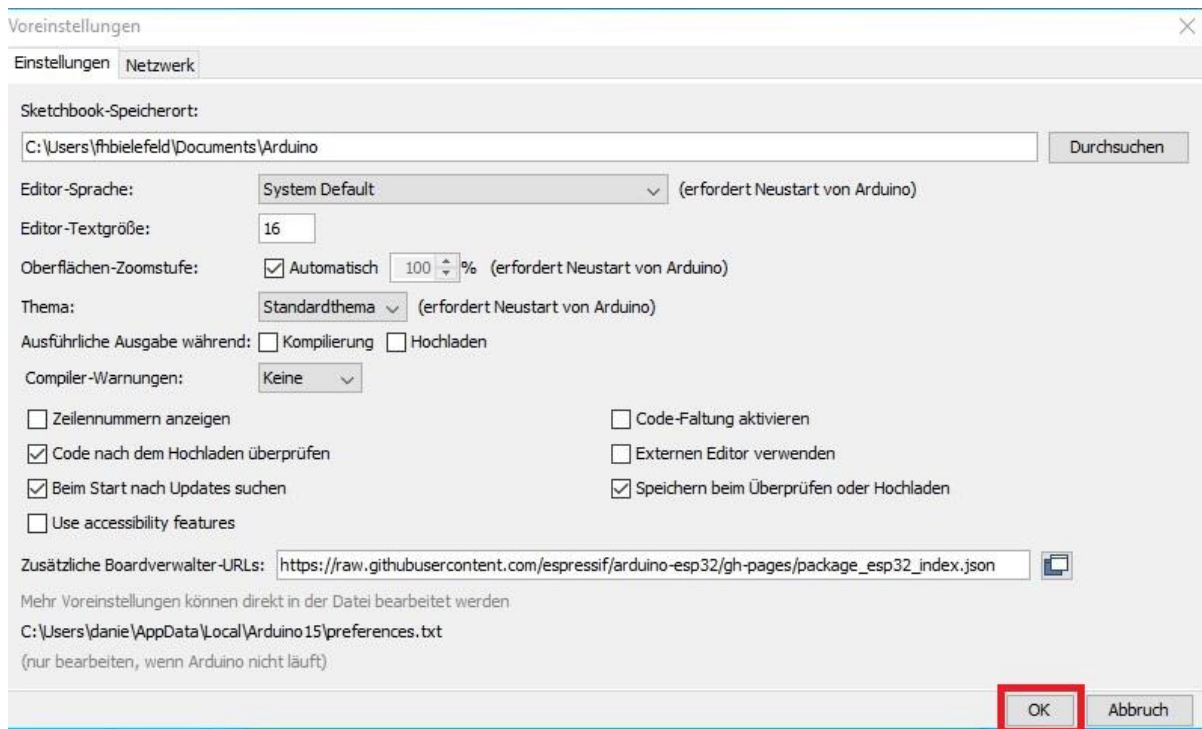
2. Geht nun in die Arduino IDE Software zu **Datei > Voreinstellungen**



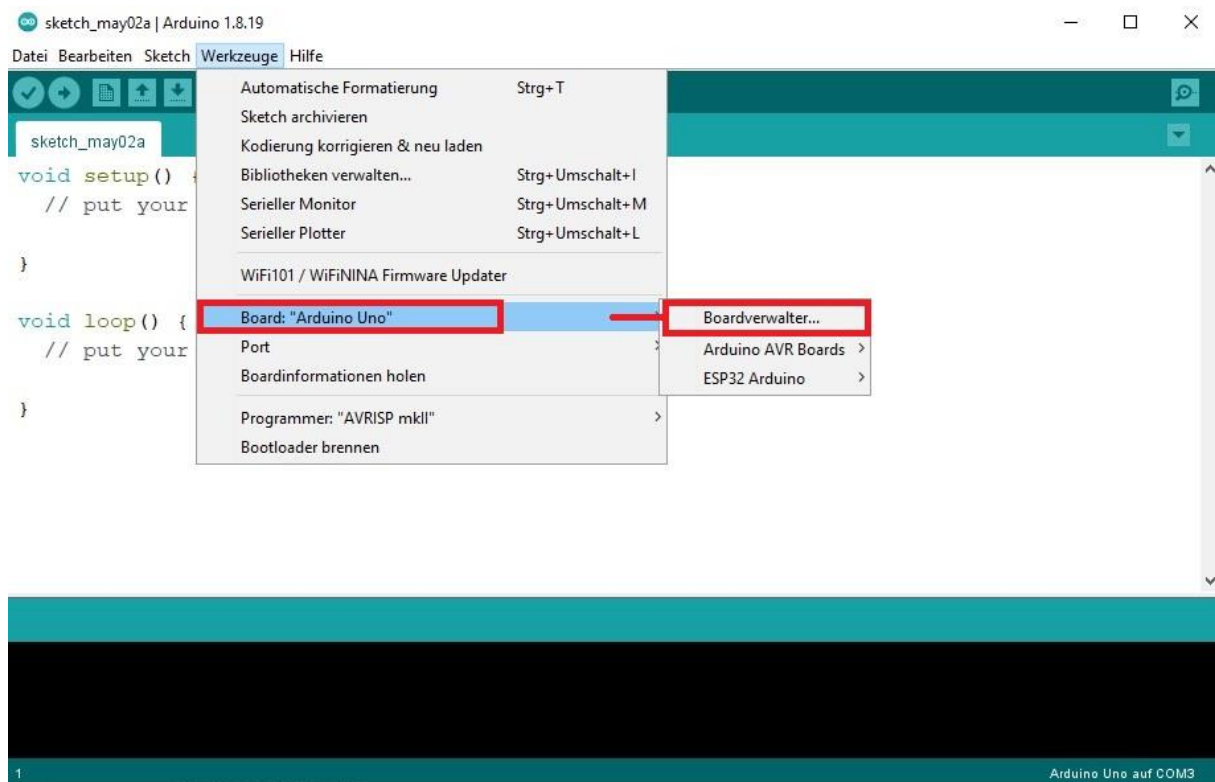
3. Gebt nun folgendes in das Feld „Zusätzliche Boardverwalter-URLs“ ein:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

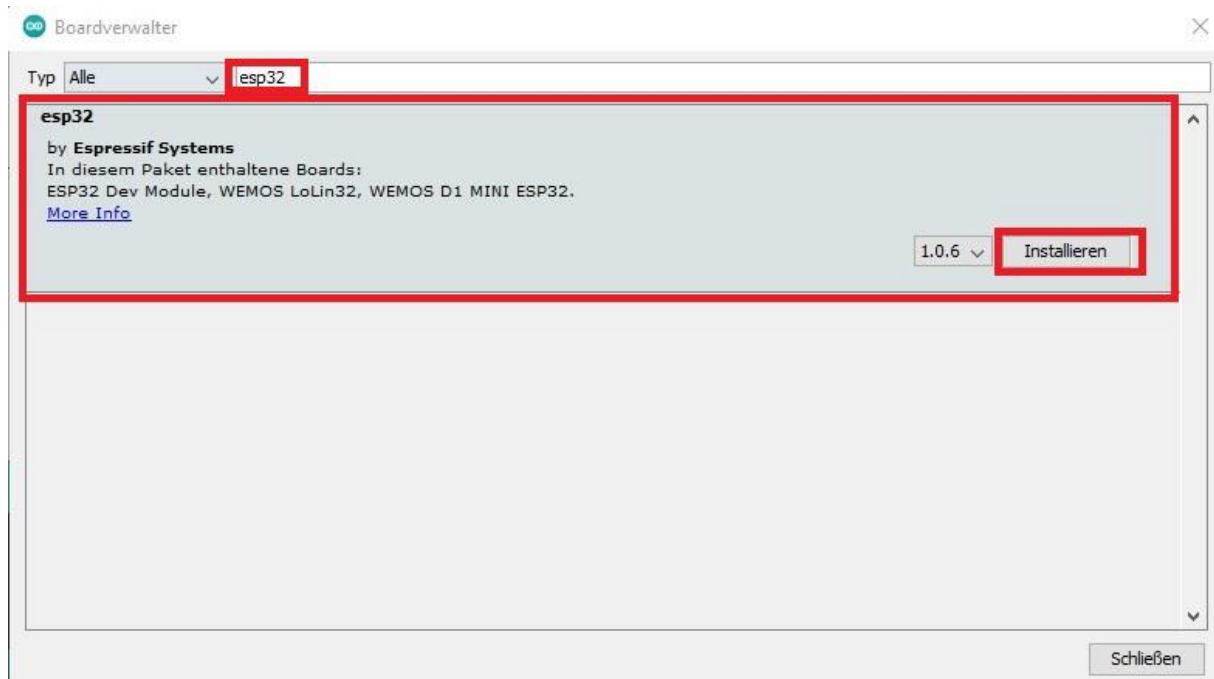
4. Klickt dann auf die Schaltfläche „OK“:



5. Öffnet den Boardverwalter und geht zu **Werkzeuge > Board > Boardverwalter...**



6. Sucht nach **ESP32** und klickt auf die Schaltfläche „Installieren“ für „**ESP32 by Espressif Systems**“:

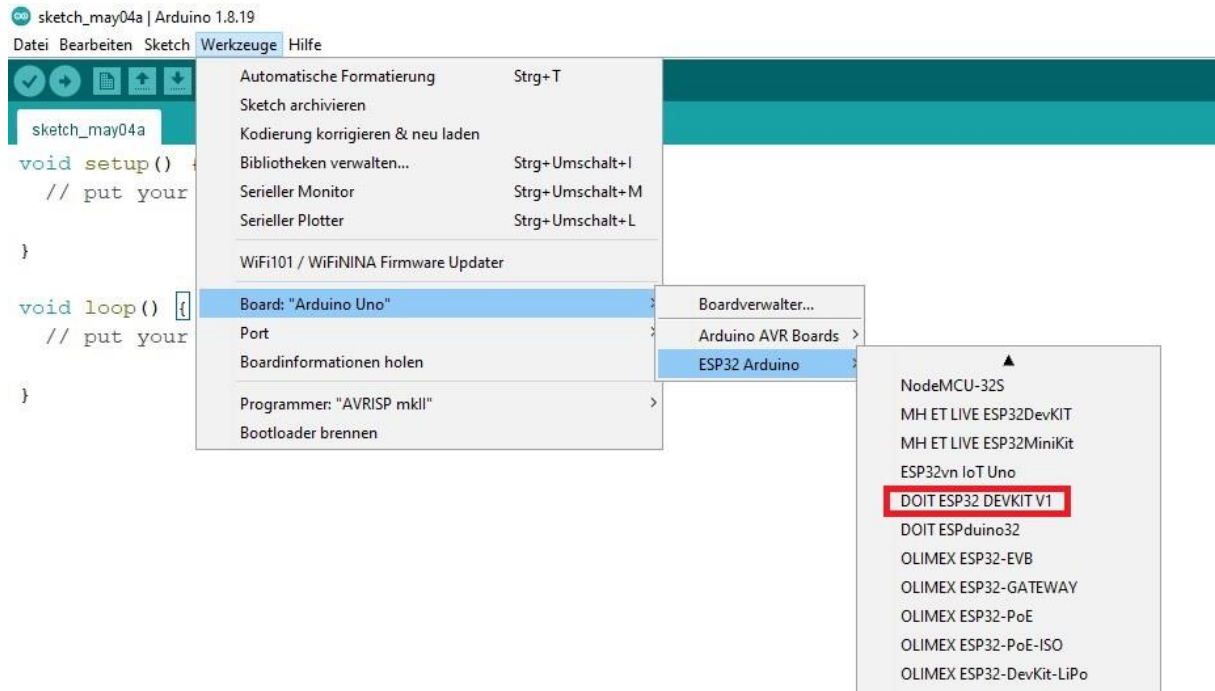


7. Nach wenigen Sekunden ist das Paket installiert.

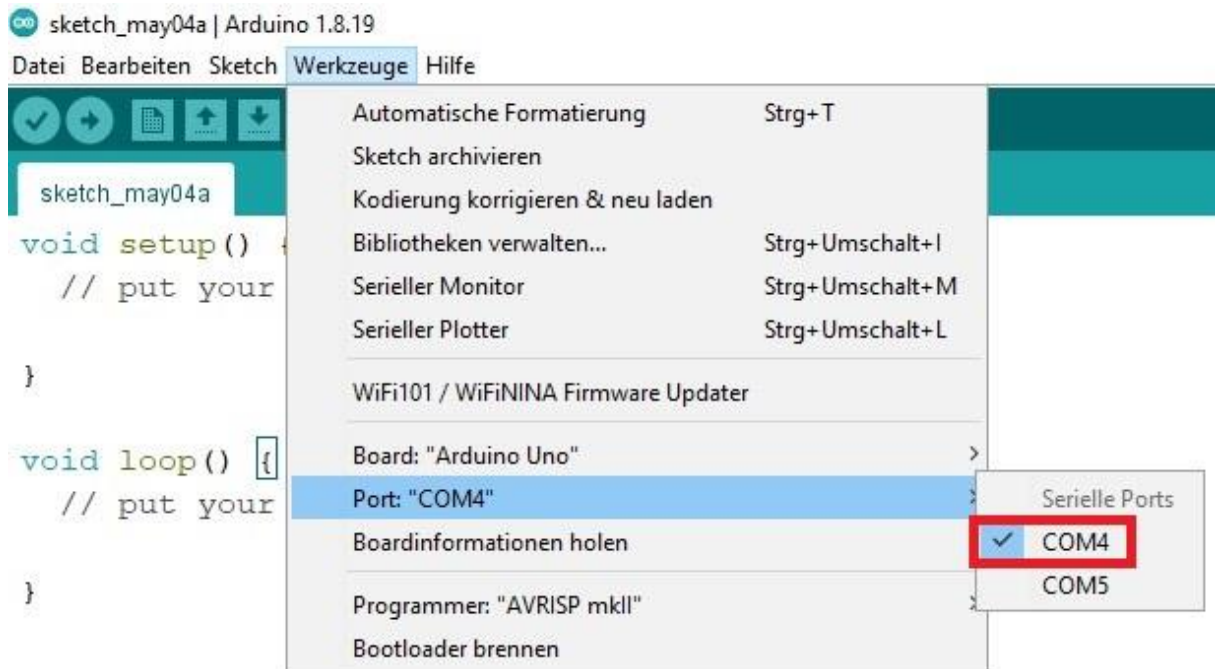
Testen der Installation

Schließt das ESP32-Board nun an den Laptop/PC an. Führt bei geöffneter Arduino IDE die folgenden Schritte aus:

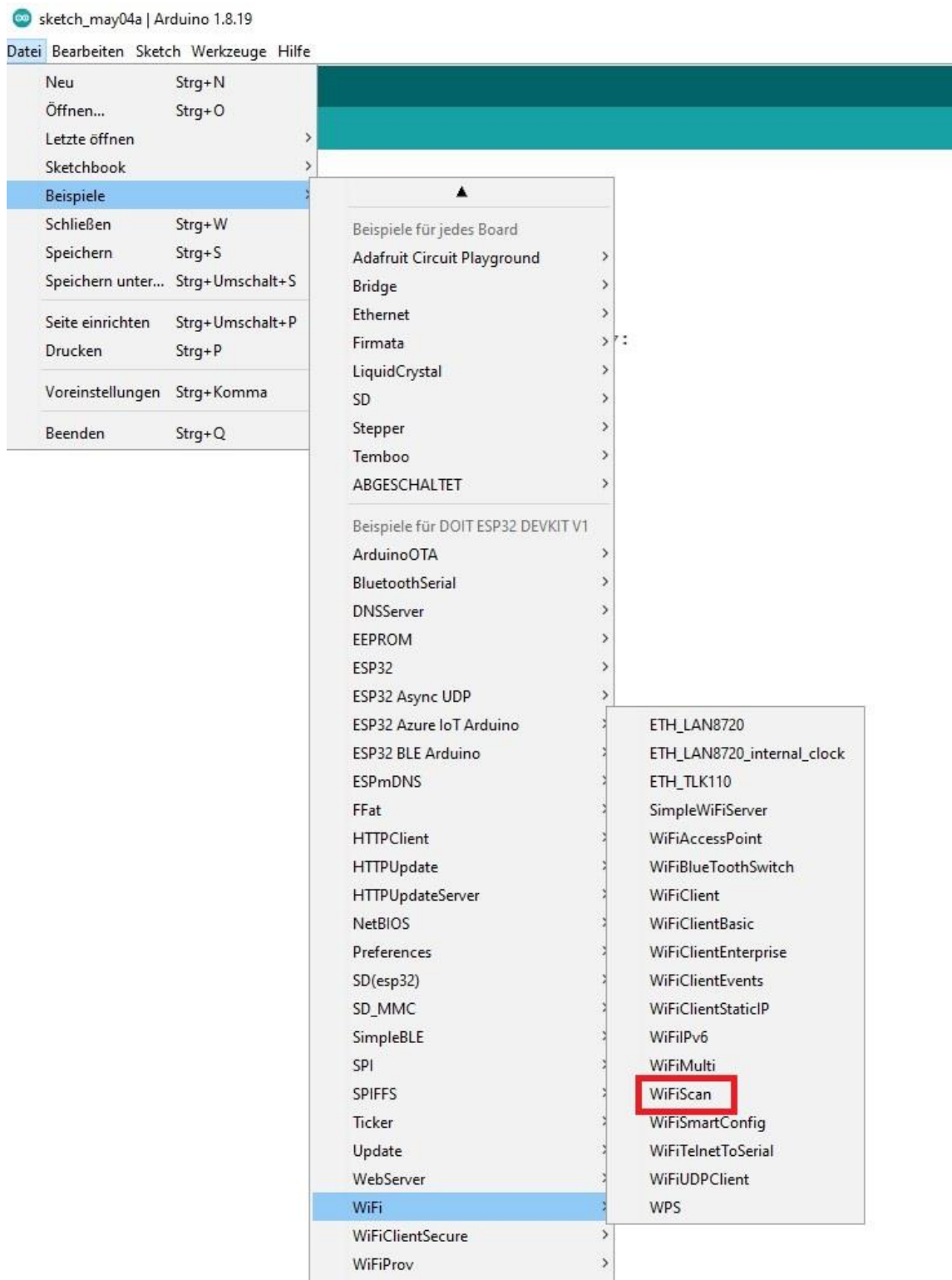
1. Wählt das Board im Menü **Werkzeuge** > **Board** aus (in diesem Fall ist es das **DOIT ESP32 DEVKIT V1**):



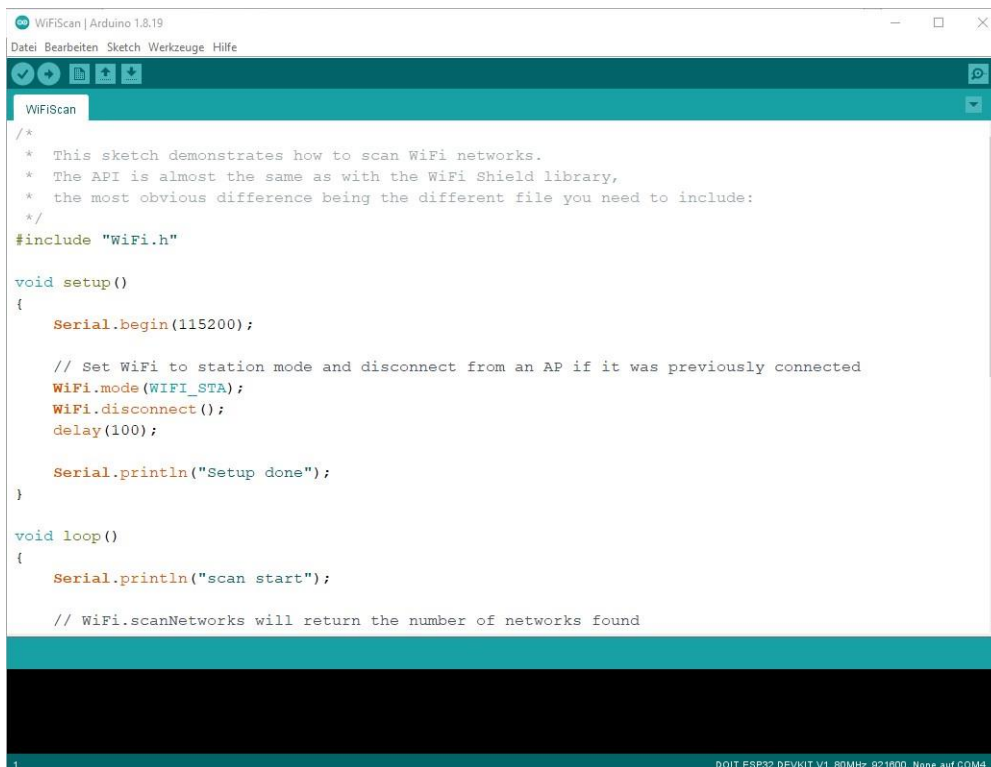
2. Wählt das USB-Port über **Werkzeuge** > **Port** aus:



3. Öffnet das folgende Beispiel unter **Datei > Beispiele > WiFi (ESP32) > WiFiScan**



4. Ein neuer Sketch öffnet sich in eurer Arduino IDE:



The screenshot shows the Arduino IDE interface with a new sketch named 'WiFiScan' open. The code is as follows:

```
/*
 * This sketch demonstrates how to scan WiFi networks.
 * The API is almost the same as with the WiFi Shield library,
 * the most obvious difference being the different file you need to include:
 */
#include "WiFi.h"

void setup()
{
  Serial.begin(115200);

  // Set WiFi to station mode and disconnect from an AP if it was previously connected
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);

  Serial.println("Setup done");
}

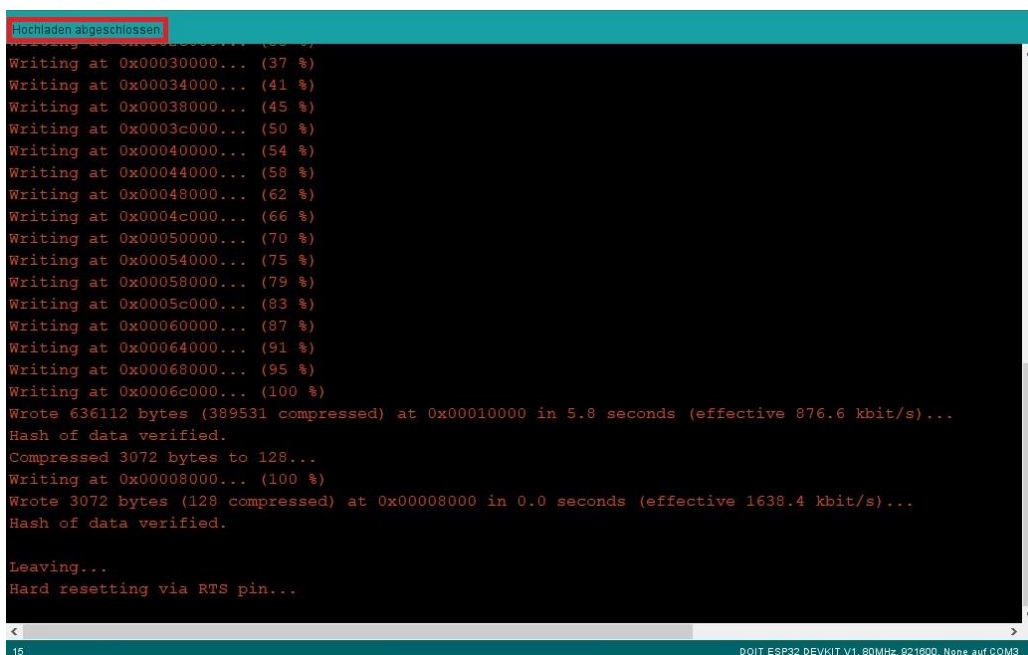
void loop()
{
  Serial.println("scan start");

  // WiFi.scanNetworks will return the number of networks found
```

5. Drückt die **Upload**- Schaltfläche in der Arduino IDE. Wartet einige Sekunden, während der Code kompiliert und auf Ihr Board hochgeladen wird.



6. Wenn alles wie erwartet gelaufen ist, sollte ein „Hochladen abgeschlossen“ zu sehen sein.



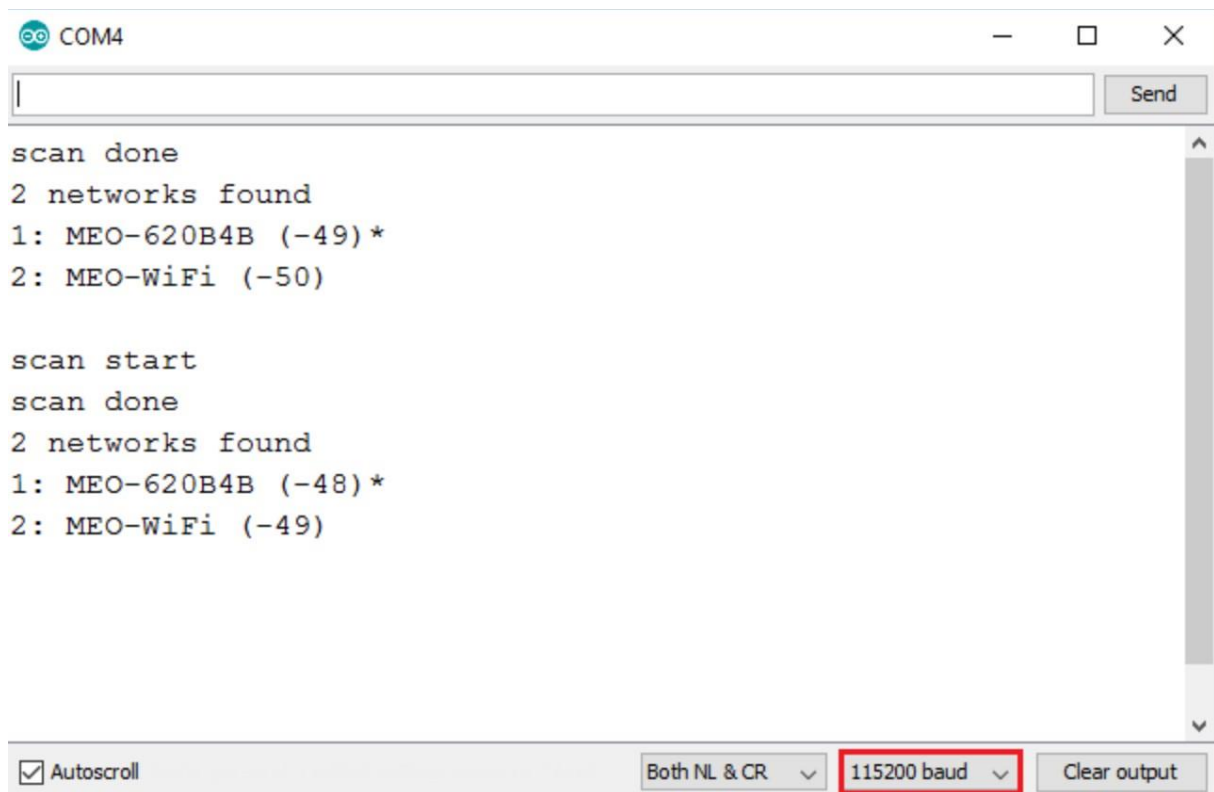
The screenshot shows the Arduino IDE's serial monitor window. The text displayed is as follows:

```
Hochladen abgeschlossen.
Writing at 0x00000000... (37 %)
Writing at 0x00030000... (41 %)
Writing at 0x00038000... (45 %)
Writing at 0x0003c000... (50 %)
Writing at 0x00040000... (54 %)
Writing at 0x00044000... (58 %)
Writing at 0x00048000... (62 %)
Writing at 0x0004c000... (66 %)
Writing at 0x00050000... (70 %)
Writing at 0x00054000... (75 %)
Writing at 0x00058000... (79 %)
Writing at 0x0005c000... (83 %)
Writing at 0x00060000... (87 %)
Writing at 0x00064000... (91 %)
Writing at 0x00068000... (95 %)
Writing at 0x0006c000... (100 %)
Wrote 636112 bytes (389531 compressed) at 0x00010000 in 5.8 seconds (effective 876.6 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 1638.4 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...
```

7. Öffnet den Arduino IDE Serial Monitor mit einer Baudrate von 115200:



8. Drückt die **RST-Taste** auf dem ESP32-Board. Es sollten die verfügbaren Netzwerke in der Nähe Ihres ESP32 sehen:

A screenshot of the Arduino IDE Serial Monitor window. The window title is "COM4". It has a text input field at the top with a "Send" button. The output area shows two network scan results. The first scan shows two networks: "1: MEO-620B4B (-49) *" and "2: MEO-WiFi (-50)". The second scan shows two networks: "1: MEO-620B4B (-48) *" and "2: MEO-WiFi (-49)". At the bottom, there is a status bar with a checked "Autoscroll" checkbox, a "Both NL & CR" dropdown menu, a "115200 baud" dropdown menu (highlighted with a red box), and a "Clear output" button.

```
scan done
2 networks found
1: MEO-620B4B (-49) *
2: MEO-WiFi (-50)

scan start
scan done
2 networks found
1: MEO-620B4B (-48) *
2: MEO-WiFi (-49)
```

☒ Autoscroll Both NL & CR 115200 baud Clear output

LEditGrow-Code

Jetzt seid ihr dran! Entwickelt euren eigenen LED-Ring mit dem LEditGrow-Code! Wie ihr das bewerkstelligen könnt, zeigen wir in den nachfolgenden Zeilen. Viel Erfolg!

Programmstruktur

Ein Programm in ARDUINO-Sprache besteht aus mindestens zwei Anweisungsblöcken (auch Methoden oder Funktionen genannt). Der erste Anweisungsblock `void setup ()` wird **einmal** nach dem Anlegen der Betriebsspannung ausgeführt. In dem Setup-Anweisungsblock werden Variablen, Pinmodi sowie Bibliotheken hinterlegt und initialisiert.

Nach Initialisierung des Setups beginnt der zweite Anweisungsblock `void loop ()` (loop=Schleife). Der Anweisungsblock wird nach jedem Durchlauf erneut aufgerufen. In der `loop ()` läuft das Hauptprogramm.

```
void setup() {  
}  
  
void loop() {  
}
```

Variablen

Eine Variable ist ein Container für Werte des Typs der Variable. Variablentypen sind:

Variablentyp	Bedeutung	Beschreibung
byte	Byte	ganze Zahl von 0 bis 255
int	Integer	ganze Zahlen (-32.768 bis 32.767)
long	ganze Zahlen	(-2 Milliarden bis 2 Milliarden)
float	Fließkommazahl	gebrochene Zahlen (3,14 etc.)
boolean	Boolescher Wert	Logischer Wert 0 (low) oder 1 (high)
char	engl: character	Alphanumerische Zeichen (Buchstaben, Zahlen, Sonderzeichen)
array	Variablenfeld	mehrere Werte eines Variablentyps können gespeichert werden

Variablen können **global** (überall verfügbar) oder **lokal** (nur in der jeweiligen Funktion verfügbar) deklariert werden. Das ist unbedingt zu beachten, sonst kann es zu Fehlermeldungen oder unlogischen Programmabläufen kommen.

Funktionen

Funktionen sind Programmanweisungsblöcke. Wiederkehrende Abfolgen von Befehlen können in Funktionen sinnvoll strukturiert werden. Parameter können an Funktionen übergeben und Werte zurückgeliefert werden.

Eine einfache Funktion könnte so aussehen:

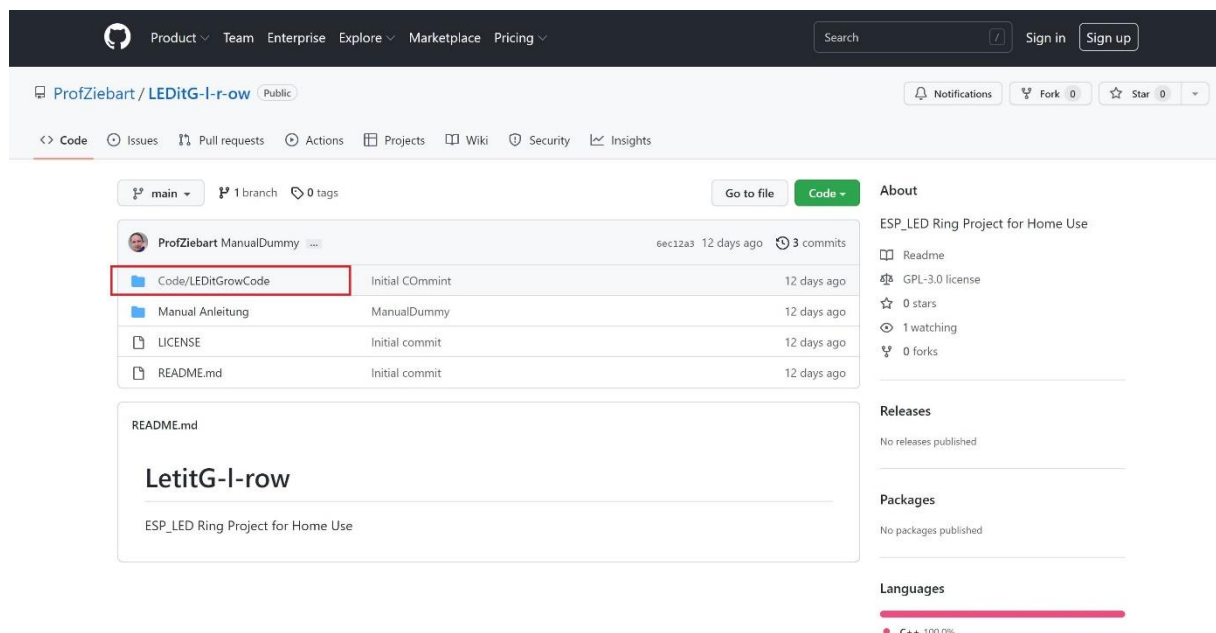
```
void led() {  
  // Anweisungsblock Start  
  digitalWrite(ledPin, HIGH);  
  delay(500);  
  digitalWrite(ledPin, LOW);  
  delay(500);  
  // Anweisungsblock Ende  
}
```

Diese oben erläuterte Programmstruktur, die Variablen sowie die Funktionen findet ihr ebenfalls in dem LEDitGrow-Code.

Ihr gelangt über den QR-Code oder über <https://github.com/ProfZiebart/LetitG-l-row> zu eurem LEDitGrow-Code den ihr Arduino IDE kopiert.

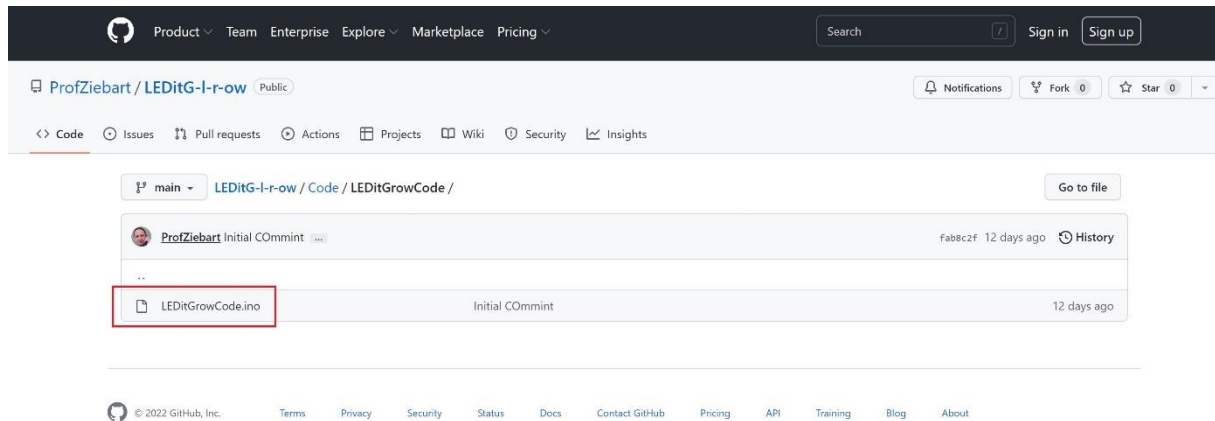
1. QR-Code nutzen oder Link: <https://github.com/ProfZiebart/LetitG-l-row> öffnen

Folgendes Fenster öffnet sich:

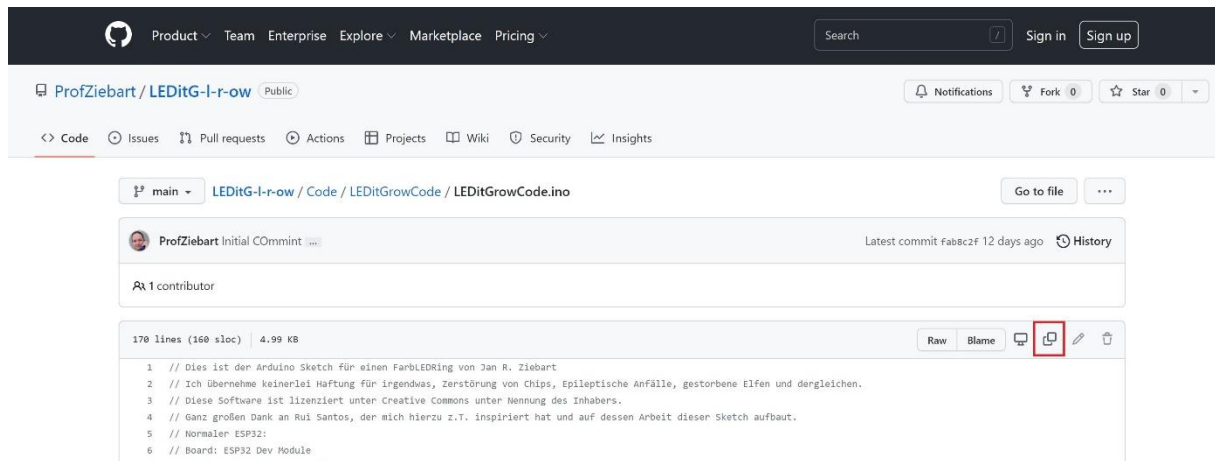


2. Über Code/LEDitGrowCode gelangt ihr zum Programmcode

3. Klickt auf LEDitGrowCode.ino



4. Kopiert euch nun den Code über das Symbol der rechten Seite



5. Öffnet in eurer installierten Arduino IDE Software eine neue Datei: **Datei>Neu** und fügt den kopierten Code ein.
6. Schließt euer fertigt verkabeltes ESP32-Modul mit Steckplatine an euren Laptop/PC über USB an.
7. Drückt die **Upload**-Schaltfläche in der Arduino IDE. Wartet einige Sekunden, während der Code kompiliert und auf das ESP32-Board hochgeladen wird.



8. Der LED-Ring sollte nun wie gewünscht leuchten.