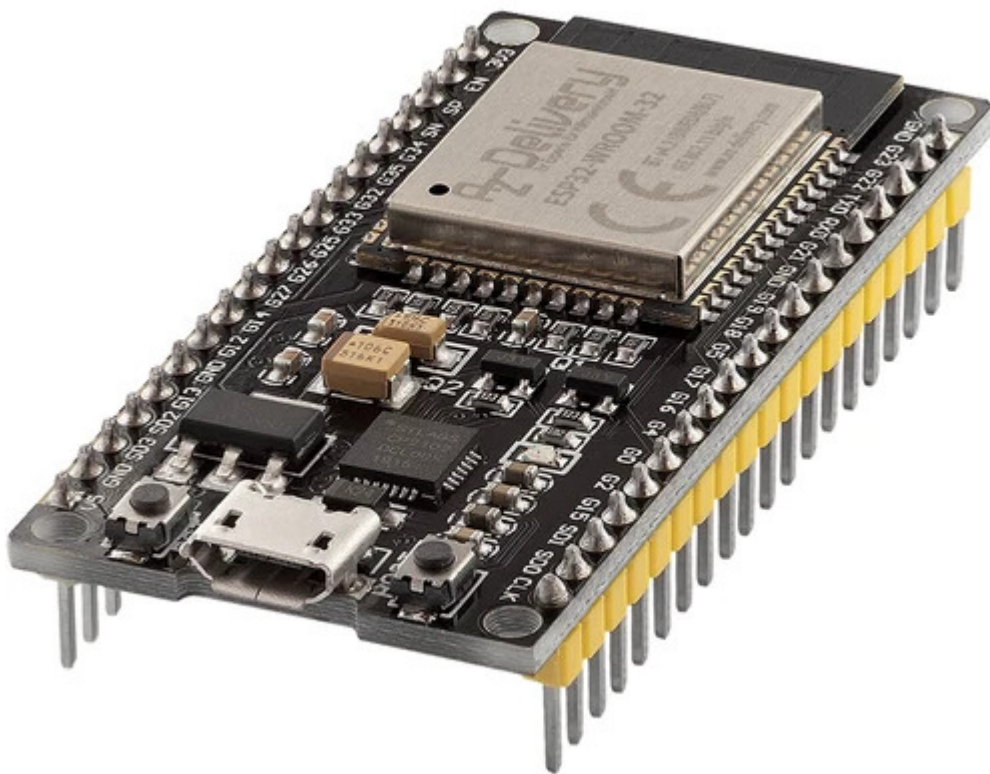


## Willkommen!

Vielen Dank, dass sie sich für unser *ESP-32 Dev Kit C V2* von AZ- Delivery entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

**Viel Spaß!**





## Inhaltsverzeichnis

Einführung.....	3
Technische Daten.....	4
ESP32 Dev Kit C V2.....	5
Pinbelegung.....	6
Pinbeschreibung.....	7
Pins der kapazitiven Touch-Sensoren.....	8
Analog-zu-Digital-Wandler Pins.....	9
Digital-zu-Analog-Wandler Pins.....	9
Echtzeituhr GPIO-Pins.....	10
PWM (Pulsweitenmodulation) Pins.....	11
I2C-Schnittstelle Pins.....	11
SPI-Schnittstelle Pins.....	12
Strapping Pins.....	12
Pins HIGH beim Booten.....	13
Freigabe (EN).....	13
USB zu Seriell Kommunikation.....	14
WiFi Kommunikation.....	15
Bluetooth Kommunikation.....	16
Andere Funktionen.....	18
Wie man die Arduino IDE einrichtet.....	19
Zusätzliche Einrichtung.....	23
ESP32 Dev Kit C V2 Anschlussbeispiel.....	27
Sketch-Beispiel.....	28



## Einführung

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 und einige andere Funktionen.

Für die Applikationsentwicklung hat man die Wahl zwischen der Arduino IDE oder der ESP-IDF (Native Plattform). Meistens wird die Arduino-IDE aufgrund ihrer Benutzerfreundlichkeit und Kompatibilität gewählt. Die Anwender-Community ist sehr aktiv und unterstützt Plattformen wie das ESP32.

Das ESP32 Dev Kit C V2 wird mit einer vorinstallierten Firmware geliefert, die es erlaubt, mit Interpretersprache zu arbeiten und Befehle über die serielle Schnittstelle (CP2102-Chip) zu senden. Die ESP32-Boards sind eine der meistgenutzten Plattformen für Internet of Things (IoT)-Projekte.

Das ESP32 Dev Kit C V2 Board ist speziell für die Arbeit auf dem Breadboard konzipiert. Es verfügt über einen Spannungsregler, der es ermöglicht, direkt vom USB-Port zu speisen. Die Input/Output Pins arbeiten mit 3,3V. Der CP2102-Chip ist für die USB-zu-Seriell-Kommunikation zuständig.

## Technische Daten

Stromversorgungsspannung (USB)	5V
Eingangs-/Ausgangsspannung	3.3V
Benötigter Betriebsstrom	min. 500mA
SoC	ESP32-WROOM 32
CPU	Xtensa® single-dual-core 32-bit LX6
Taktfrequenzbereich	80MHz / 240MHz
RAM	512kB
Externer Flash-Speicher	4MB
I/O Pins	34
ADC Kanäle	18
ADC Auflösung	12-bit
DAC Kanäle	2
DAC Auflösung	8-bit
Schnittstellen	SPI, I2C, I2S, CAN, UART
Wi-Fi Protokolle	802.11 b/g/n (802.11n bis zu 150 Mbps)
Wi-Fi Frequenz	2.4 GHz - 2.5 GHz
Bluetooth	V4.2 - BLE und Classic Bluetooth
Drahtlose Antenne	PCB
Dimensionen	56x28x13mm(2.2x1.1x0.5in)



## **ESP32 Dev Kit C V2**

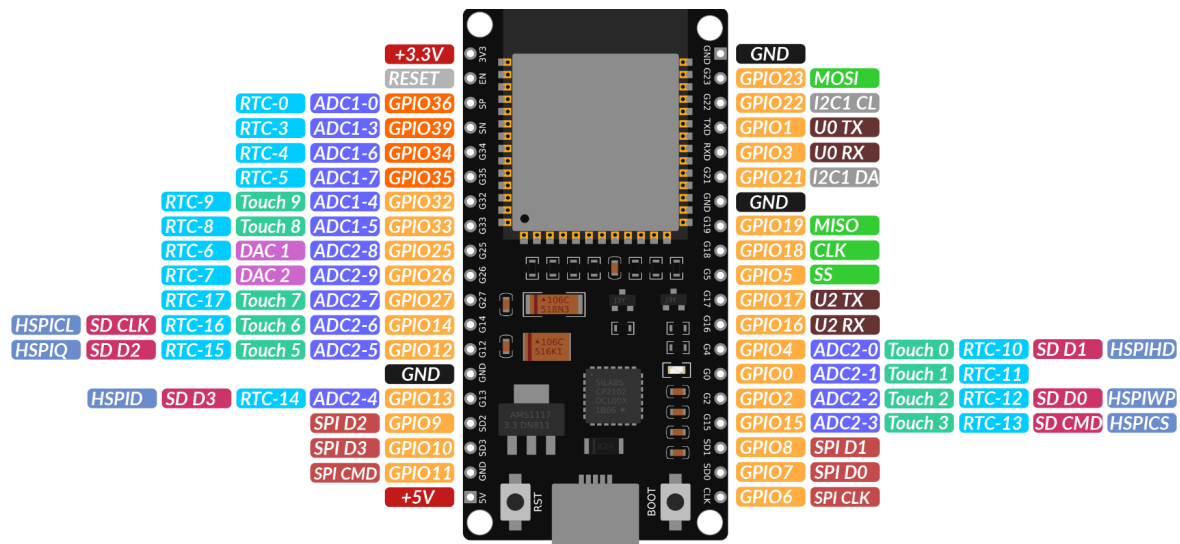
Die ESP32 WROOM-32-Serie der Wi-Fi-Chips wird von Espressif Systems hergestellt. Der ESP32 WROOM-32 ist ein preiswertes Wi-Fi-Modul, das sich für DIY-Projekte im Bereich Internet of Things (IoT) eignet. Dieses Modul kommt mit vielen GPIOs und unterstützt eine Vielzahl von Protokollen wie SPI, I2C, I2S, UART und mehr. Das Beste daran ist, dass es mit einem drahtlosen Netzwerk ausgestattet ist, was es von anderen Mikrocontrollern wie dem Mikrocontroller unterscheidet. Das bedeutet, dass er Geräte über Wi-Fi und Bluetooth® fernsteuern und überwachen kann.

Der ESP32 WROOM 32 ist ein System-on-Chip (SoC), das einen 32-Bit Tensilica-Mikrocontroller, digitale Standard-Peripherieschnittstellen, Antennenschalter, RF-Balun, Leistungsverstärker, rauscharme Empfangsverstärker, Filter und Power-Management-Module in einem kleinen Gehäuse kombiniert. Er bietet 2,4GHz Wi-Fi (802.11 b/g/n, unterstützt Geschwindigkeiten bis zu 150MB/s), BLE und klassische Bluetooth®-Funkkommunikation, 34 I/O-Pins, I2C- und I2S-Schnittstellen, ADC (Analog-Digital-Wandlung), DAC (Digital-Analog-Wandlung), eine SPI-Schnittstelle, UART auf dedizierten Pins und PWM (Pulsweitenmodulation).

Der Prozessorkern von Espressif , LX6 genannt, basiert auf dem Xtensa® Dual-Core 32-Bit LX6 Prozessor-Controller und läuft im Frequenzbereich von 80-240MHz. Er verfügt über ein 448kB großes Boot-ROM, 520kB On-Chip-SRAM und 4MB externen Flash-Speicher, auf den über die SPI-Schnittstelle zugegriffen werden kann.

## Pinbelegung

Das ESP32 Dev Kit C V2 hat 38 Pins. Die Pinbelegung ist wie folgt:



- Orange Digital In/Out ports (all support PWM)
- Blue Digital Input ports
- Purple Analog Input 12 bits, 0 to 3.3V
- Green Analog Output 8 bits, 0 - 3.3V
- Light blue Capacitive Touch Sensor ports
- Pink I/O -pins from RTC ultra low power processor, usable in deep sleep mode
- Red SD card interface
- Dark red SPI bus for Flash-memory, do not use

The following pins show the default assignment. All these signals can be changed to any In/Out port. This applies also to UART0 and UART1, which cannot be accessed in the default assignment.

- Grey I2C bus (Wire)
- Green VSPI bus
- Brown Serial interfaces
- Blue HSPI bus

Eine detaillierte Beschreibung der Pinbelegung und der I/O-Fähigkeiten entnehmen Sie dem Datenblatt, das Sie unter dem folgenden [link](#) finden können.

**Hinweis:** Der absolute Maximalstrom, der pro GPIO gezogen werden darf, beträgt 40 mA gemäß dem Abschnitt "Empfohlene Betriebsbedingungen" im ESP32-Datenblatt.



## Pinbeschreibung

Genau wie ein normales Mikrocontroller-Board verfügt das ESP32 Dev Kit C V2 über digitale Ein-/Ausgangs-Pins (GPIO Pins - General Purpose Input/Output Pins). Diese digitalen Eingänge/Ausgänge arbeiten mit 3,3V.

**Es darf keine 5V Spannung an die ESP32 Chip Pins angeschlossen werden!**

Die Pins sind nicht 5V-tolerant, das Anlegen von mehr als 3,3V an einem Pin wird den Chip beschädigen.

Die GPIO Pins 34 bis 39 sind GPIOs - nur Eingangspins. Diese Pins haben keine internen Pull-ups oder Pull-down-Widerstände. Sie können nicht als Ausgänge verwendet werden, verwenden Sie diese Pins also nur als Eingänge: GPIO 34, GPIO 35, GPIO 36, GPIO 39

Auf dem ESP-WROOM-32 Chip befindet sich ein integrierter SPI-Flash. Die Pins GPIO6 bis GPIO 11 sind in bestimmten ESP32-Entwicklungsboards freigelegt. Diese Pins sind mit dem integrierten SPI-Flash auf dem Chip verbunden und werden für andere Anwendungen nicht empfohlen.

GPIO 6 (SCK/CLK), GPIO 7 (SDO/SD0), GPIO 8 (SDI/SD1), GPIO 9 (SHD/SD2), GPIO 10 (SWP/SD3), GPIO 11 (CSC/CMD).



## **Kapazitive Touchsensor-Pins**

Der ESP32 hat 10 interne kapazitive Touch-Sensoren. Die kapazitiven Touch Pins können auch verwendet werden, um den ESP32 aus dem Tiefschlaf aufzuwecken. Diese internen Touch-Sensoren sind mit diesen GPIOs verbunden: T0 (GPIO 4), T1 (GPIO 0), T2 (GPIO 2), T3 (GPIO 15), T4 (GPIO 13), T5(GPIO 12), T6 (GPIO 14), T7 (GPIO 27), T8 (GPIO 33), T9 (GPIO 32).



## **Analog-zu-Digital-Wandler Pins**

Der ESP32 hat 18x12 Bit ADC (Analog-Digital-Wandler) Eingangskanäle (während der ESP8266 nur 1x 10 Bit ADC hat). Dies sind die GPIOs, die als ADC und entsprechende Kanäle verwendet werden können:

ADC1\_CH0 (GPIO 36), ADC1\_CH1 (GPIO 37), ADC1\_CH2 (GPIO 38),  
ADC1\_CH3 (GPIO 39), ADC1\_CH4 (GPIO 32), ADC1\_CH5 (GPIO 33),  
ADC1\_CH6 (GPIO 34), ADC1\_CH7 (GPIO 35), ADC2\_CH0 (GPIO 4),  
ADC2\_CH1 (GPIO 0), ADC2\_CH2 (GPIO 2), ADC2\_CH3 (GPIO 15),  
ADC2\_CH4 (GPIO 13), ADC2\_CH5 (GPIO 12), ADC2\_CH6 (GPIO 14),  
ADC2\_CH7 (GPIO 27), ADC2\_CH8 (GPIO 25), ADC2\_CH9 (GPIO 26).

## **Digital-zu-Analog-Wandler Pins**

Es gibt 2 x 8 Bits DAC (Digital-zu-Analog-Wandler) Kanäle auf dem ESP32, um digitale Signale in analoge Spannungssignalausgänge zu konvertieren.

Dies sind die DAC-Kanäle:

DAC1 (GPIO25), DAC2 (GPIO26).



## **Echtzeituhr GPIO-Pins**

Es gibt eine RTC ( Echtzeituhr ) GPIO Unterstützung auf dem ESP32. Die GPIOs, die zum RTC-Subsystem mit geringem Stromverbrauch geroutet werden, können verwendet werden, wenn sich der ESP32 im Tiefschlaf befindet. Diese RTC-GPIOs können verwendet werden, um den ESP32 aus dem Tiefschlaf aufzuwecken, wenn der Ultra Low Power (ULP) Co-Prozessor läuft. Die folgenden GPIOs können als externe Aufweckquelle verwendet werden: RTC\_GPIO0 (GPIO36), RTC\_GPIO3 (GPIO39), RTC\_GPIO4 (GPIO34), RTC\_GPIO5 (GPIO35), RTC\_GPIO6 (GPIO25), RTC\_GPIO7 (GPIO26), RTC\_GPIO8 (GPIO33), RTC\_GPIO9 (GPIO32), RTC\_GPIO10 (GPIO4), RTC\_GPIO11 (GPIO0), RTC\_GPIO12 (GPIO2), RTC\_GPIO13 (GPIO15), RTC\_GPIO14 (GPIO13), RTC\_GPIO15 (GPIO12), RTC\_GPIO16 (GPIO14), RTC\_GPIO17 (GPIO27).



## **PWM (Pulsweitenmodulation) Pins**

Der ESP32 LED-PWM-Controller hat 16 unabhängige Kanäle, die so konfiguriert werden können, dass sie PWM-Signale mit unterschiedlichen Eigenschaften erzeugen. Alle Pins, die als Ausgänge fungieren können, können als PWM-Pins verwendet werden (GPIOs 34 bis 39 können keine PWM erzeugen). Um ein PWM-Signal einzustellen, müssen Sie folgende Parameter im Code definieren: Frequenz des Signals, Tastverhältnis, PWM-Kanal, GPIO, an dem das Signal ausgegeben werden soll.

## **Die I2C-Schnittstellen Pins**

Der ESP32 hat zwei I2C-Kanäle und jeder Pin kann als SDA oder SCL eingestellt werden. Wenn Sie den ESP32 mit der Arduino IDE verwenden, sind die Standard I2C Pins:

GPIO 21 (SDA), GPIO 22 (SCL). der Ultra Low Power (ULP) Co-Prozessor läuft. Die folgenden GPIOs können als externe Aufweckquelle verwendet werden:

## SPI interface Pins

Standardmäßig ist die Pinbelegung für SPI Pins:

SPI	MOSI	MISO	CLK	CS
VSPI	GPIO 23	GPIO 19	GPIO 18	GPIO 5
HSPI	GPIO 13	GPIO 12	GPIO 14	GPIO 15

## Strapping Pins

Folgende Pins werden verwendet, um den ESP32 in den Bootloader- oder Flash-Modus zu versetzen:

GPIO 0, GPIO 2, GPIO 4, GPIO 5 (muss beim Booten HIGH sein), GPIO 12 (muss beim Booten LOW sein), GPIO 15 (muss beim Booten HIGH sein).

Die meisten Entwicklungsboards setzen die Pins in den richtigen Zustand für den Flash- oder Boot-Modus. Wenn einige Peripheriegeräte mit den Strapping Pins verbunden sind und die IDE nicht in der Lage ist, den Code hochzuladen oder den ESP32 zu flashen, kann es daran liegen, dass diese Peripheriegeräte den ESP32 daran hindern, in den richtigen Modus zu gelangen. Nach dem Zurücksetzen, Flashen oder Booten funktionieren diese Pins wie erwartet. Es gibt eine Dokumentationsanleitung zur Auswahl des Bootmodus unter folgendem [link](#). Weitere und umfangreichere Erklärungen sind nicht in diesem eBook vorhanden, daher verweisen wir auf das Datenblatt.



## **Pins HIGH beim Booten**

Einige GPIOs ändern ihren Zustand auf HIGH oder geben PWM-Signale beim Booten oder Reset aus. Das bedeutet, dass wenn Ausgänge an diese GPIOs angeschlossen werden, dies zu unerwarteten Ergebnissen führen kann, wenn der ESP32 zurückgesetzt oder gebootet wird.

GPIO 1, GPIO 3, GPIO 5, GPIO 6 bis GPIO 11 (verbunden mit dem integrierten SPI-Flash-Speicher des ESP32 - nicht zur Verwendung empfohlen), GPIO 14, GPIO 15.

## **Enable (EN)**

Enable (EN) ist der Freigabe-Pin des 3,3V-Reglers. Er ist in einem Pulled-up-Status und muss mit Masse verbunden werden, um den 3,3-V-Regler zu deaktivieren. Das bedeutet, dass dieser Pin z.B. mit einem Druckknopf verbunden werden kann, um Ihren ESP32 neu zu starten.



## **USB zu serieller Kommunikation**

Das ESP32 Dev Kit C V2 hat einen microUSB-Anschluss. Er ist um den CP21202-Chip von Silicon Laboratories herum aufgebaut, der eine serielle USB-zu-UART-Kommunikation ermöglicht. Der Chip hat die Funktion eines virtuellen COM-Ports (VCP), der in PC-Anwendungen als COM-Port erscheint. Die CP2102 UART-Schnittstelle implementiert alle RS-232-Signale, einschließlich der Steuer- und Handshaking-Signale, so dass die bestehende System-Firmware nicht geändert werden muss. Um den ESP32 nutzen zu können, muss der Treiber installiert werden.

## WiFi-Kommunikation

Das ESP32 Dev Kit C V2 hat eine integrierte Wi-Fi-Kommunikationsschnittstelle und kann in drei verschiedenen Modi arbeiten: Wi-Fi-Station, Wi-Fi-Access-Point und beides gleichzeitig. Es unterstützt die folgenden Funktionen:

- 802.11b und 802.11g Datenübertragungsraten
- 802.11n MCS0-7 in den Bandbreiten 20MHz und 40MHz
- 802.11n MCS32
- 802.11n 0.4µs guard-interval
- Datenübertragung bis zu 150 Mbps
- Empfangs-STBC 2x1
- Bis zu 20 dBm Sendeleistung
- Einstellbare Sendeleistung
- Antennen-Diversity und -Auswahl (über Software verwaltete Hardware)

## Bluetooth-Kommunikation

Das ESP32 Dev Kit C V2 verfügt über ein integriertes Bluetooth-Radio und unterstützt folgende Features:

- Klasse-1-, Klasse-2- und Klasse-3-Sendeausgangsleistungen und über 30 dB dynamischer Regelbereich
- $\pi/4$  DQPSK und 8 DPSK Modulation
- Hohe Leistung bei der NZIF-Empfangsempfindlichkeit mit über 98 dB Dynamikbereich
- Klasse-1-Betrieb ohne externe PA
- Interner SRAM ermöglicht Datenübertragung mit voller Geschwindigkeit, gemischte Sprach- und Datenübertragung und vollen Piconet-Betrieb
- Logik für eine Vorwärtsfehlerkorrektur, eine Header-Fehlerkontrolle, eine Zugriffscode-Korrelation, CRC, Demodulation, Erzeugung von Verschlüsselungsbitströmen, Whitening und Sendeimpulsformung
- ACL, SCO, eSCO und AFH
- A-law,  $\mu$ -law und CVSD digitaler Audio-CODEC in der PCM-Schnittstelle
- SBC-Audio-CODEC
- Power-Management für Low-Power-Anwendungen
- SMP mit 128-bit AES



Desweiteren unterstützt das Bluetooth Radio die folgenden Protokolle der Kommunikationsschnittstellen:

- UART HCI-Schnittstelle, bis zu 4 Mbps
- SDIO / SPI HCI-Schnittstelle
- I2C-Schnittstelle
- PCM / I2S Audio-Schnittstelle

## Andere Funktionen

Der ESP32-WROOM 32D-Chip hat einen integrierten Hall-Effekt-Sensor, der Änderungen des Magnetfeldes in seiner Umgebung erkennt.

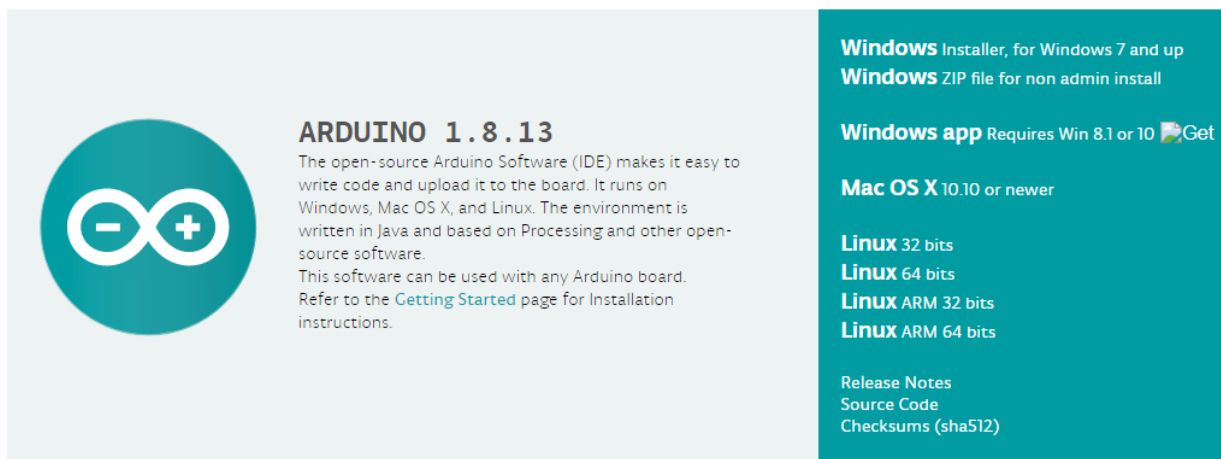
Der Hall-Sensor basiert auf einem N-Carrier-Widerstand. Wenn sich der Chip im Magnetfeld befindet, entwickelt der Hall-Sensor eine kleine Spannung am Widerstand, die direkt vom Analog-Digital-Wandler (ADC) gemessen werden kann, oder er wird durch den extrem rauscharmen analogen Vorverstärker verstärkt und dann vom ADC gemessen.

Der Temperatursensor erzeugt eine Spannung, die mit der Temperatur variiert. Die Spannung wird intern über einen Analog-Digital-Wandler in einen digitalen Code umgewandelt. Der Temperatursensor hat einen Bereich von  $-40^{\circ}\text{C}$  bis  $125^{\circ}\text{C}$ . Da der Offset des Temperatursensors aufgrund von Prozessvariationen von Chip zu Chip variiert und zusammen mit der von der Wi-Fi-Schaltung selbst erzeugten Wärme (die die Messungen beeinflusst), ist der interne Temperatursensor nur für Anwendungen geeignet, die Temperaturänderungen anstelle von absoluten Temperaturen erfassen, und auch für Kalibrierungszwecke. Wenn der Benutzer jedoch den Temperatursensor kalibriert und das Gerät in einer minimal eingeschalteten Anwendung verwendet, könnten die Ergebnisse genau genug sein.

## Wie man die Arduino IDE einrichtet

Falls die Arduino-IDE nicht installiert ist, folgen Sie dem [link](#) und laden Sie die Installationsdatei für das Betriebssystem Ihrer Wahl herunter. Die Arduino IDE Version, die hier verwendet wird ist die **1.8.13**.

### Download the Arduino IDE



**ARDUINO 1.8.13**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

**Windows** Installer, for Windows 7 and up  
**Windows** ZIP file for non admin install

**Windows app** Requires Win 8.1 or 10 [Get](#)

**Mac OS X** 10.10 or newer

**Linux** 32 bits  
**Linux** 64 bits  
**Linux** ARM 32 bits  
**Linux** ARM 64 bits

[Release Notes](#)  
[Source Code](#)  
[Checksums \(sha512\)](#)

Für Windows Benutzer: Doppelklicken Sie auf die heruntergeladene .exe l-Datei und folgen Sie den Anweisungen im Installationsfenster.

# Az-Delivery

Für *Linux* Benutzer, laden Sie eine Datei mit der Erweiterung *.tar.xz* herunter, die extrahiert werden muss. Wenn sie extrahiert ist, gehen Sie in das extrahierte Verzeichnis und öffnen Sie das Terminal in diesem Verzeichnis. Zwei *.sh* Skripte müssen ausgeführt werden, das erste namens *arduino-linux-setup.sh* und das zweite heißt *install.sh*.

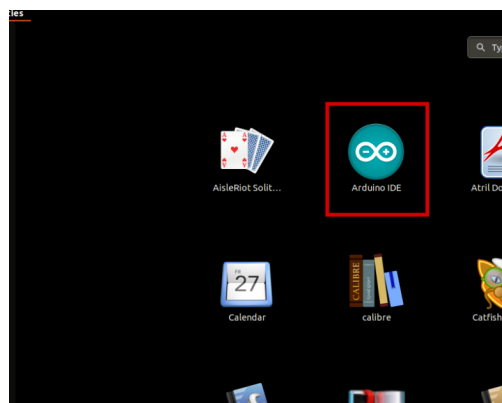
Um das erste Skript im Terminal auszuführen, öffnen Sie das Terminal im extrahierten Ordner und führen Sie den folgenden Befehl aus:

**sh arduino-linux-setup.sh user\_name**

**user\_name** - ist der Name eines Superusers im Linux-Betriebssystem. Ein Passwort für den Superuser muss beim Start des Befehls eingegeben werden. Warten Sie einige Minuten, bis das Skript vollständig abgeschlossen ist.

Das zweite Skript mit der Bezeichnung *install.sh*-Skript muss nach der Installation des ersten Skripts verwendet werden. Führen Sie den folgenden Befehl im Terminal (extrahiertes Verzeichnis) aus: **sh install.sh**

Nach der Installation dieser Skripte gehen Sie zu *All Apps*, wo die *Arduino-IDE* installiert ist.



# Az-Delivery

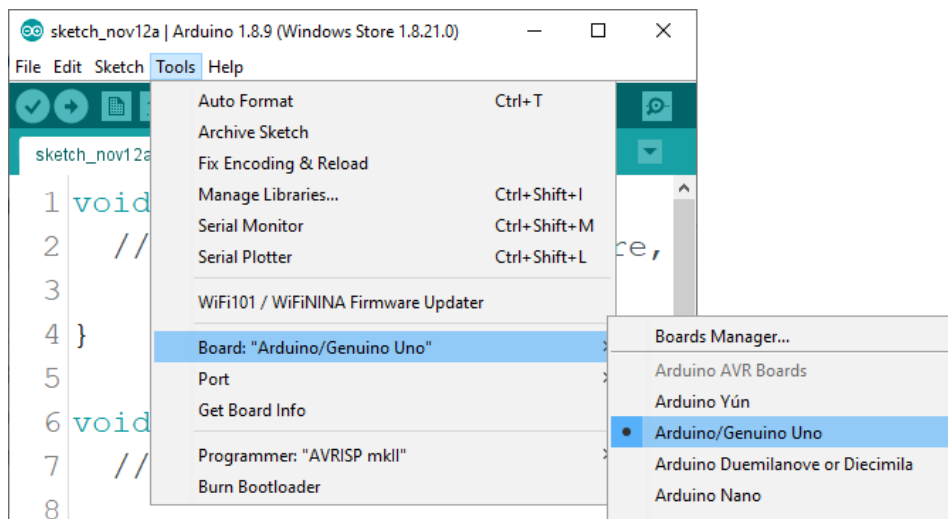
Fast alle Betriebssysteme werden mit einem vorinstallierten Texteditor ausgeliefert (z.B. *Windows* mit *Notepad*, *Linux* Ubuntu mit *Gedit*, *Linux Raspbian* mit *Leafpad* usw.). Alle diese Texteditoren sind für den Zweck des eBooks vollkommen in Ordnung.

Zunächst ist zu prüfen, ob Ihr PC ein Mikrocontroller-Board erkennen kann.

Öffnen Sie die frisch installierte Arduino-IDE, und gehen Sie zu:

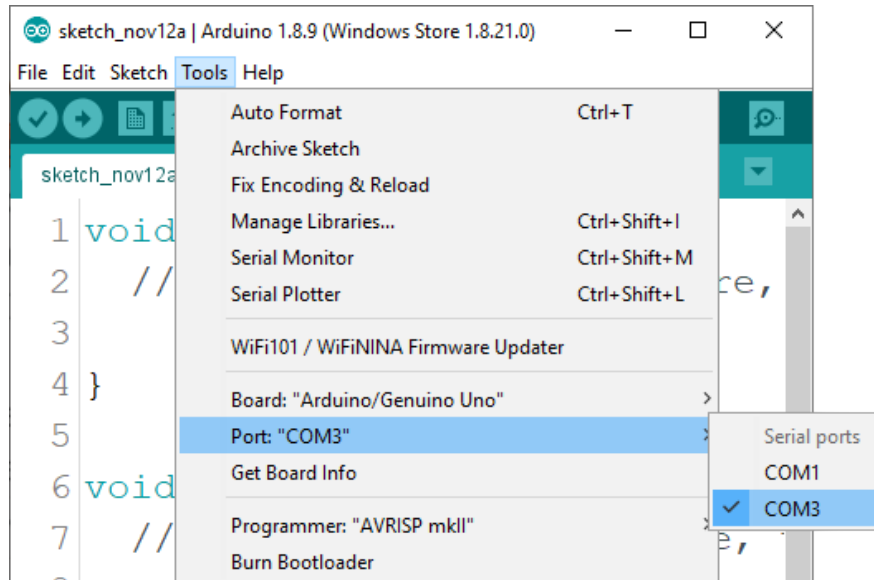
*Tools > Board > {your board name here}*

*{your board name here}* sollte der *Arduino/Genuino Uno* sein, wie es auf dem folgenden Bild zu sehen kann:



Der Port, an den das Mikrocontroller-Board angeschlossen ist, muss ausgewählt werden. Gehe zu: *Tools > Port > {port name goes here}* und wenn das Mikrocontroller-Board an den USB-Port angeschlossen ist, ist der Portname im Drop-down Menü auf dem vorherigen Bild zu sehen.

Wenn die Arduino-IDE unter Windows verwendet wird, lauten die Portnamen wie folgt:



Für *Linux* Benutzer, ist zum Beispiel der Portname `/dev/ttyUSBx`, wobei x für eine ganze Zahl zwischen 0 und 9 steht.



## **Zusätzliche Einrichtung**

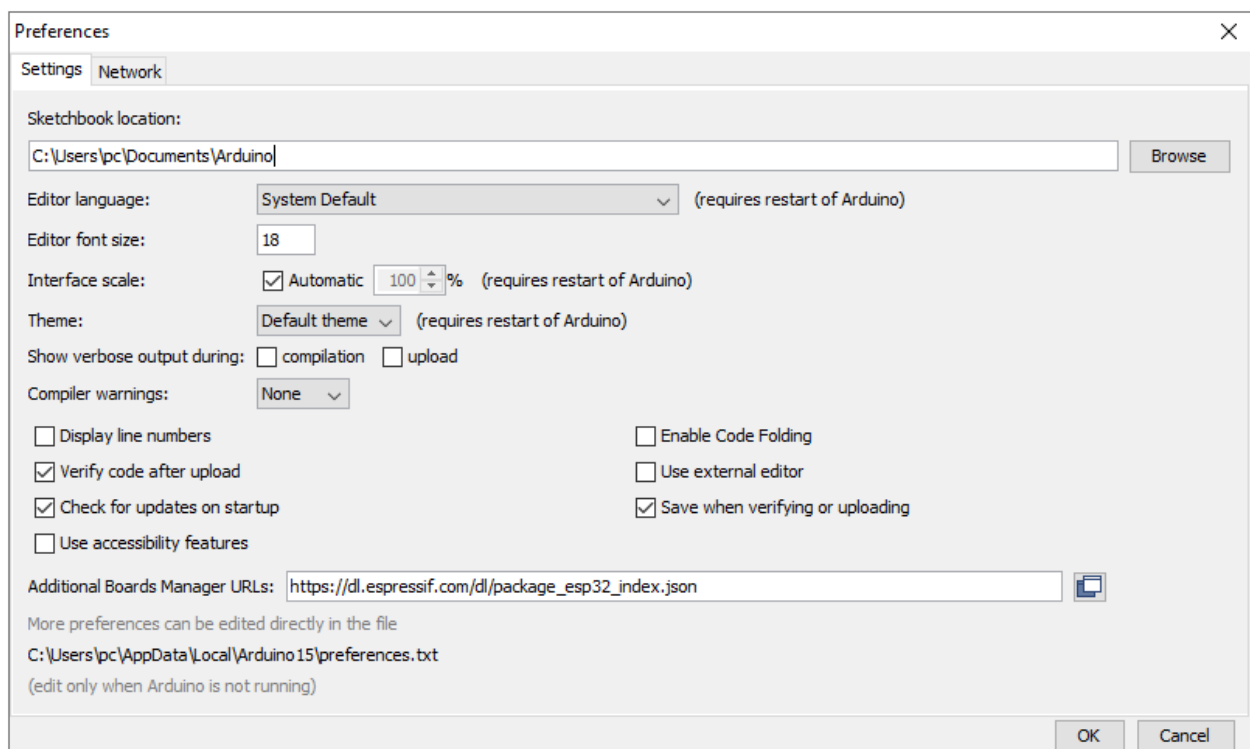
Um das ESP32 Dev Kit C V2 mit der Arduino IDE zu verwenden, folgen Sie diesen einfachen Schritten. Vor der Einstellung der Arduino IDE muss der Treiber für die USB-zu-Seriell-Kommunikation installiert werden. Falls der Treiber nicht automatisch installiert wird, gibt es eine Support-Seite, die die Treiber für Windows/Mac oder Linux enthält und je nachdem ausgewählt werden kann. Die Treiber können unter folgenden [link](#) heruntergeladen werden.

# Az-Delivery

Als Nächstes installieren Sie den Support für die ESP32-Plattform, öffnen Sie die Arduino IDE und gehen Sie zu: *File > Preferences*, und finden Sie das Feld "Additional URLs".

Kopieren Sie folgenden Link:

[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)





# Az-Delivery

Fügen Sie diesen Link in das Feld "Additional URLs" ein. Wenn Sie bereits einen oder mehrere Links in diesem Feld haben, fügen Sie einfach ein Komma nach dem letzten Link ein, fügen Sie den neuen Link nach dem Komma ein und klicken Sie auf die Schaltfläche **OK**. Schließen Sie dann die Arduino-IDE.



Öffnen Sie die Arduino IDE erneut und gehen Sie zu:

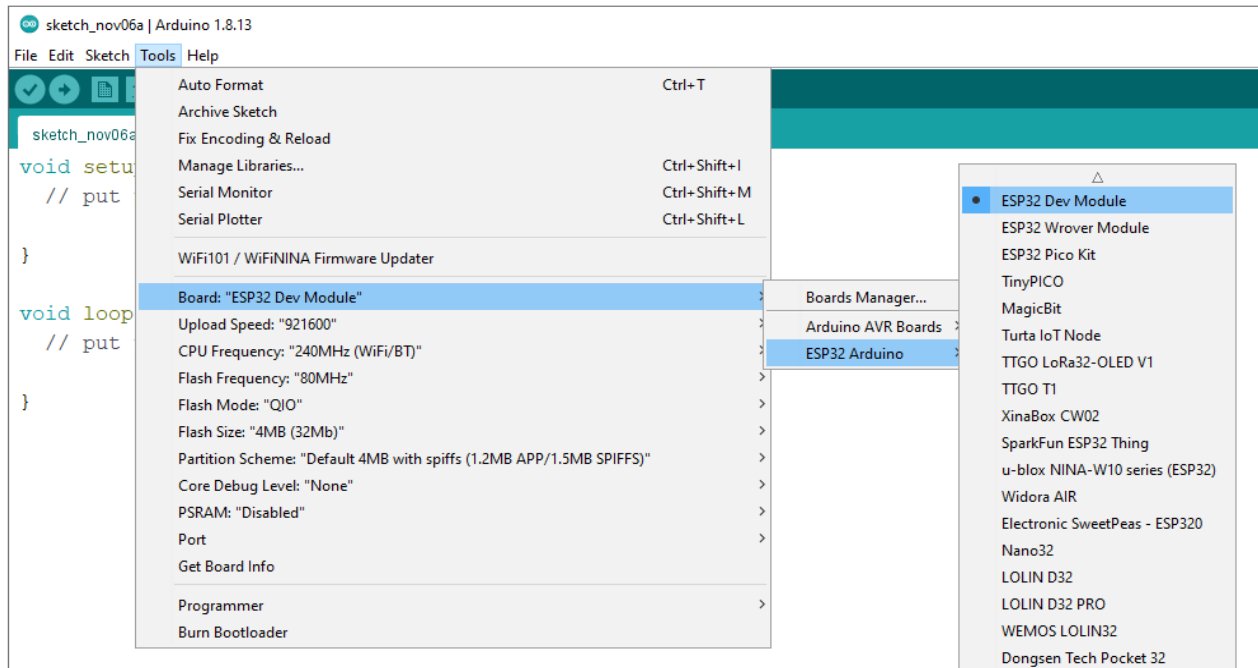
*Tools > Board > Boards Manager*

Es öffnet sich ein neues Fenster, geben Sie *esp32* in das Suchfeld ein und installieren Sie das Board mit dem Namen *esp32* von *Espressif Systems*, wie unten abgebildet:



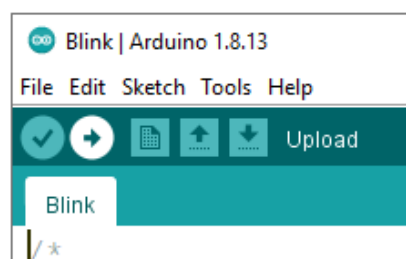
Um das ESP32-Board auszuwählen, gehen Sie zu:

*Tools > Board > ESP32 Arduino > ESP32 Dev Module*



Um einen Sketch auf das ESP32-Board zu laden, wählen Sie zuerst den Port aus, an dem Sie das Board verbunden haben. Gehen Sie zu:

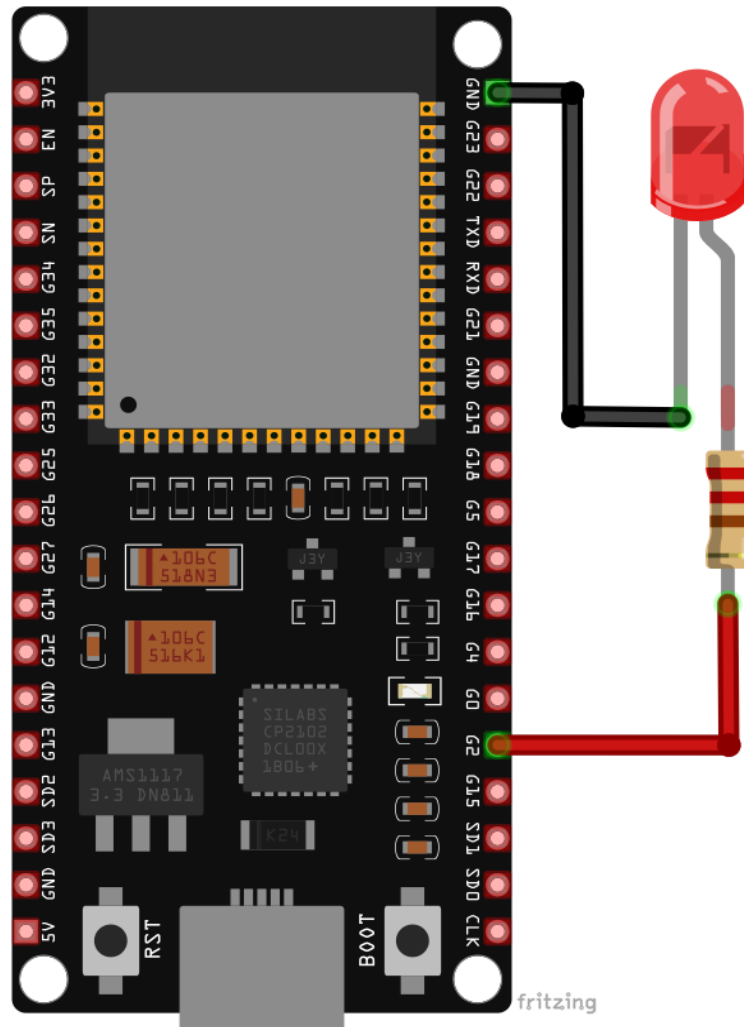
*Tools > Port > {port name}*



Wenn der Upload nicht beim ersten Versuch funktioniert, könnte es helfen den "Boot-Knopf" auf dem Modul während des Uploads zu drücken. Bitte benutzen Sie ein USB 2.0-zertifiziertes Kabel für die Programmierung.

## ESP32 Dev Kit C V2 Anschlussbeispiel

Verbinden Sie das ESP32 Dev Kit C V2 mit einer LED und einem Widerstand, wie unten abgebildet:



ESP32 Dev Kit C V2 pin	LED pin	Drahtfarbe
GPIO2 (pin2)	Anode (+) über Widerstand	<b>Roter Draht</b>
GND	Kathode (-)	<b>Schwarzer Draht</b>

# Az-Delivery

## Sketch-Beispiel

### Blinkende LED

```
int ledPin = 2;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

## PWM - Pulsweitenmodulation

```
#define LEDC_CHANNEL_0 0
#define LEDC_TIMER_13_BIT 13
#define LEDC_BASE_FREQ 5000
#define LED_PIN 2

int brightness = 0;
int fadeAmount = 5;

void ledcAnalogWrite(uint8_t channel, uint32_t value, uint32_t valueMax = 255)
{
    uint32_t duty = (8191 / valueMax) * min(value, valueMax);
    ledcWrite(channel, duty);
}

void setup() {
    ledcSetup(LEDC_CHANNEL_0, LEDC_BASE_FREQ, LEDC_TIMER_13_BIT);
    ledcAttachPin(LED_PIN, LEDC_CHANNEL_0);
}

void loop() {
    ledcAnalogWrite(LEDC_CHANNEL_0, brightness);
    brightness = brightness + fadeAmount;
    if (brightness <= 0 || brightness >= 255) {
        fadeAmount = -fadeAmount;
    }
    delay(30);
}
```

# Az-Delivery

Jetzt sind Sie dran! Entwickeln Sie Ihre eigenen Projekte und Smart- Home Installationen. Wie Sie das bewerkstelligen können, zeigen wir Ihnen unkompliziert und verständlich auf unserem Blog. Dort bieten wir Ihnen Beispielskripte und Tutorials mit interessanten kleinen Projekten an, um schnell in die Welt der Mikroelektronik einzusteigen. Zusätzlich bietet Ihnen auch das Internet unzählige Möglichkeiten, um sich in Sachen Mikroelektronik weiterzubilden.

**Falls Sie nach weiteren Hochwertige Mikroelektronik und Zubehör , sind Sie bei der AZ-Delivery Vertriebs GmbH goldrichtig. Wir bieten Ihnen zahlreiche Anwendungsbeispiele, ausführliche Installationsanleitungen, Ebooks, Bibliotheken und natürlich die Unterstützung unserer technischen Experten.**

<https://az-delivery.de>

Viel Spaß!

Impressum

<https://az-delivery.de/pages/about-us>