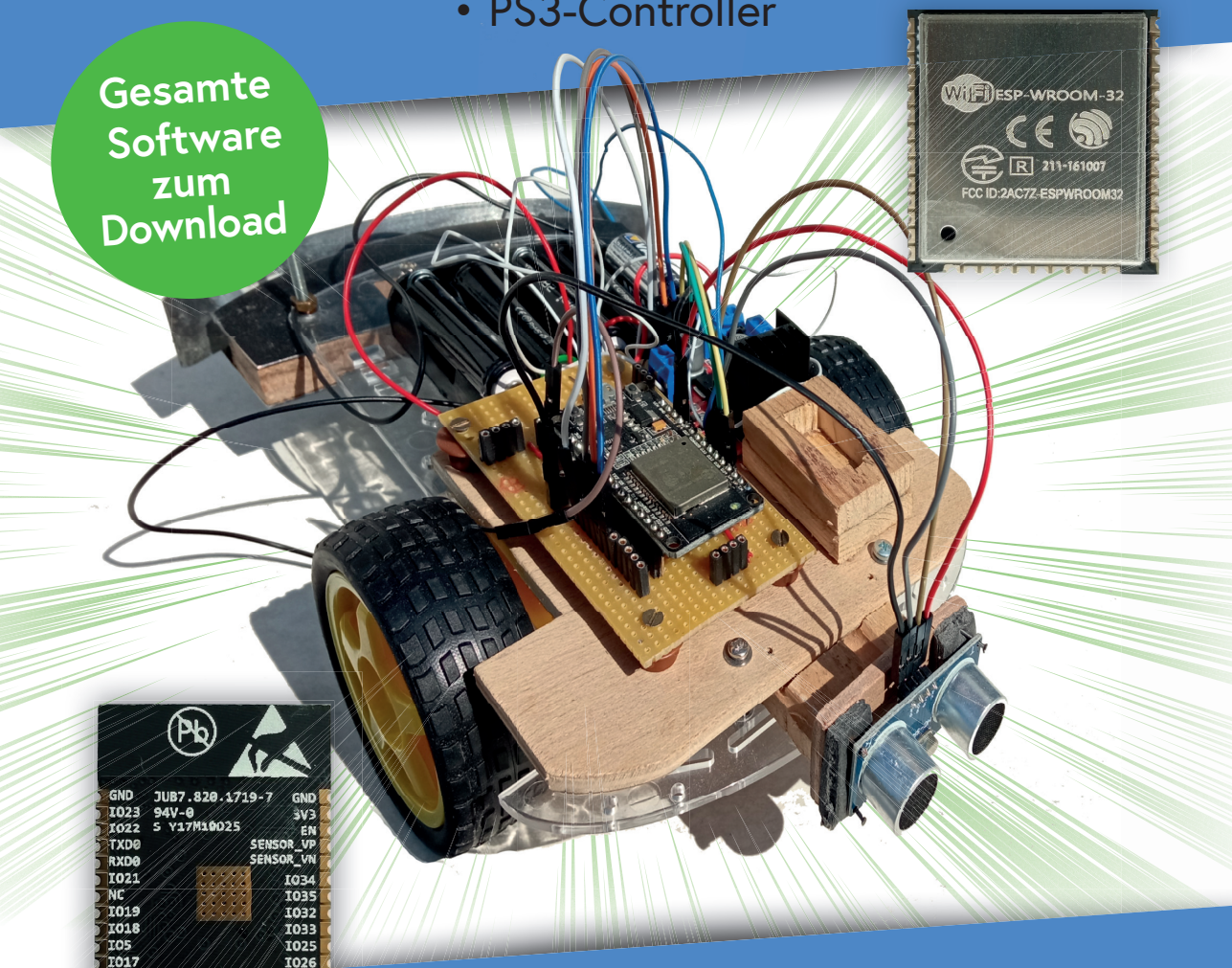
elektor books

ESP32 steuert Roboterauto

- Open-Source-Code mit Arduino IDE und PlatformIO
- Autonomes Fahren: GPS, Accelerometer, Gyroskop
- PS3-Controller

Gesamte Software zum Download



Udo Brandes

elektor
design > share > earn

ESP32 steuert Roboterauto

Open-Source-Code mit Arduino IDE und PlatformIO
Autonomes Fahren: GPS, Accelerometer, Gyroskop
PS3-Controller

www.elektor.de/20277

Udo Brandes

elektor
design › share › sell

1. Auflage 2022

Umschlaggestaltung: Elektor, Aachen

Korrekturlesen: Andreas Riedenauer
Satz und Aufmachung: D-Vision, Julian van den Berg | Oss (NL)
Druck: Ipskamp Printing, Enschede, Niederlande

ISBN 978-3-89576-521-6
Ebook 978-3-89576-522-3

Elektor-Verlag GmbH, Aachen
www.elektor.de

Hinweis: Alle Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt. Trotzdem sind Fehler nicht gänzlich auszuschließen. Der Autor sieht sich deshalb in der Pflicht darauf hinzuweisen, dass er weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen kann.

Die in dem Buch angegebenen Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. können auch ohne besondere Kennzeichnung Marken sein und als solche den gesetzlichen Regelungen unterliegen.

Für die Mitteilung etwaiger Fehler ist der Autor jedoch dankbar. Sie erreichen ihn über den Verlag. Internetadressen oder Versionsnummern stellen den bei Bucherstellung verfügbaren Informationsstand dar. Der Autor übernimmt keine Verantwortung oder Haftung für Veränderungen, die sich aus von ihm nicht zu vertretenden Umständen ergeben. Zum Download angebotene Dateien und Informationen dienen ausschließlich privaten Zwecken.

Alle Software finden Sie zum Download im Elektor Store.
Der Link ist www.elektor.de/20277.

Einleitung

Mikrocontroller haben sich zu einem erschwinglichen und weit verbreiteten Bauteil entwickelt. Die Vorreiterrolle fällt sicherlich Arduino zu. Dabei bezeichnet der Begriff nicht allein die Hardware, die aus einem einfachen Ein-/Ausgabe-Board mit einem Mikrocontroller sowie analogen und digitalen Ein- und Ausgängen besteht. Vielmehr bezieht sich der Name ebenso auf eine Entwicklungsumgebung, die auch technisch weniger Versierten den Zugang zur Programmierung und zu Mikrocontrollern erleichtert. Hard- und Software sind quelloffen.

Diese Pionierarbeit von Arduino haben sich andere Hersteller von Hardware zunutze gemacht und für ihre Produkte hinsichtlich der Softwareentwicklung einen Zugang zur Arduino-Plattform geöffnet. Hierzu zählt auch die Firma Espressif mit ihrer ***ESP32***-Serie. Mikrocontroller dieser Baureihe zeichnen sich durch eine Vielzahl implementierter Funktionen aus, die bei einem Arduino konventioneller Prägung mit einem Atmel-AVR-Mikrocontroller aus der megaAVR-Serie, erst mit s.g. ***Shields*** (das sind aufsteckbare Erweiterungen) nachgerüstet werden müssen. Prominentes Beispiel sind hier die WiFi-Funktionalitäten.

Nun sind in Büchern, auf Internetseiten oder in Foren sicherlich allgemeine Informationen für die Realisierung eines Projektes "Roboterautos mit dem ESP32" zu finden. In der Regel handelt es sich dabei aber nur um Ausführungen zu einem Teilaspekt, die denn inhaltlich oder funktional nicht aufeinander abgestimmt sind. So ist nicht nur die Beschaffung der benötigten Informationen mühselig und zeitaufwändig; sie ist vielmehr auch außerordentlich fehlerträchtig. Dies kann in der Folge der Freude am eigenen Projekt doch sehr abträglich sein und sogar dazu führen, dass es nach einem vielversprechenden Start frustriert aufgegeben wird.

Ansatzpunkt dieses Buches ist, diese Lücke zu schließen. Es geht auf verschiedene Möglichkeiten eines Chassis ein, vermittelt nötige technische und elektronische Kenntnisse und führt schrittweise von einer einfachen Motorsteuerung zu einem komplexen sensor- und sprachgesteuerten Roboterauto. Außerdem werden interessante Randbereiche aufgegriffen, die so in den meisten Internetdarstellungen nicht enthalten sind, so z.B. Einbindung von speziellen Funktionen der ESP-IDF API.

Dabei wird versucht, schrittweise vom Einfachen zum Schwierigen vorzugehen und komplexere Aufgaben in kleine Teile zu zerlegen, die sich dann zu einem größeren Ganzen fügen. Im Vordergrund stehen realisierbare und nachvollziehbare Lösungsvorschläge, die Sie in Teilen auch als optionales Modul übernehmen oder darauf verzichten können. Deshalb ist ist es sicherlich nicht notwendig, die Kapitel sequenziell abzuarbeiten. Aber vielleicht stockt der Umsetzungsfluss an der einen oder anderen Stelle, sodass die Projektarbeit wieder Fahrt aufnehmen kann, wenn Informationen aus vorhergehenden Abschnitten vor Augen sind.

Neben den textlichen Ausführungen spielen Abbildungen eine tragende Rolle. Getreu dem Motto, ein Bild sagt mehr als tausend Worte, lässt sich mit Bildern – trotz des Platzes die sie nun einmal einnehmen – die zu vermittelnde Botschaft einfach besser darstellen.

Für wen das Buch gedacht ist
Das Buch richtet sich an alle,

- die neugierig sind und sich gerne mit den Möglichkeiten von Mikrocontroller im Zusammenspiel mit dem Internet und dem PC,
- die zusammenhängenden Informationen zum Thema Robotautos suchen
- die mit einfachen Mitteln schnell loslegen möchten,
- die aber auch etwas weiterführende und vertiefende Anregungen, Vorschläge und praktische Anleitungen suchen
- die auf funktionierenden, aussagekräftigen und ausführlichen Schritt für Schritt Erläuterungen Wert legen
- die in komprimierter Form Tipps für eigene Entwicklungen suchen
- Freude am eigenen Entwickeln und Realisieren haben

Voraussetzungen
Im Prinzip reicht es für viele der im Buch dargestellten Beispiele aus, wenn man über ein Entwicklungs-Board mit dem ESP32, einem geeigneten Chassis, einen Internetzugang und einen PC verfügt. Programmierkenntnisse und Erfahrungen in der Installation von Software sind nicht erforderlich, da jeder einzelne Schritt zur Realisierung des Projektes detailliert beschrieben und mit Screenshots visuell unterlegt ist.

Der Inhalt dieses Buches
Bei der Reihenfolge der Kapitel wurde versucht beginnend bei der Darstellung von grundlegenden Informationen über die Lösung einfacher Aufgaben zu etwas anspruchsvolleren Techniken zu führen.

- der Mikrocontroller ESP32
  Hier wird der Mikrocontroller ESP32 beginnend bei den technischen Daten, dem Entwicklungsboard sowie der Inbetriebnahme vorgestellt.
- die Software erstellen
  In diesem Kapitel geht es um die Entwicklung der Software mit Hilfe vom Platform-IO
- die Stromversorgung
- Kern sind hier die Herausforderungen der Stromversorgung
- rund um die Hardware
  Thema ist das Chassis des Roboterautos und alles um Ihre Werkstatt.

- das Chassis
  Dieser Bereich befasst sich mit verschiedenen Möglichkeiten eines Chassis.

- der Gleichstrommotor
  Das Herz des Roboterautos ist Gegenstand dieses Kapitels.

- kabellose Steuerung über WiFi
  Hier werden verschiedene Szenarien des Einsatzes des WiFi-Moduls vorgestellt.

- mit Sensoren Hindernisse erkennen
  Dieses Kapitel befasst sich mit zwei Sensoren und Ausweichmanövern bei Hindernissen.

- eine eigene Roboterauto-App
  Gegenstand ist eine eigene App für Ihr Roboterauto.

- Servo und Lichtsensor
  Dieses Kapitel zeigt den Einsatz von Servo und Lichtsensor für autonomes Fahren.

- GPS
  Zeichnen Sie mit einem GPS-Empfänger GPS-Daten auf.

- Accelerometer / Gyroskop
  An dieser Stelle geht es um den Einstieg in die Lagebestimmung mit Accelerometer und Gyroskop.

- PS3-Controller
  Steuern Sie den ESP32 mit einer PS3-Konsole.

- Roboterauto-Kamera
  Hier geht es um den Einsatz einer Kamera für das Roboterauto.

Besondere Schriftarten
Verschiedene Objekte werden in von einer im üblichen Text abweichenden Schriftart dargestellt:

- ***Begriffe*** sind in blauer Kursivschrift fett hervorgehoben
- `Code-Zeilen` sind in nicht proportionaler Schrift dargestellt
- Dateipfade und -namen besitzen eine graue Schriftfarbe
- `Konsolenbefehle` sind rot
- Tastaturbefehle wie **Strg+V** sind grün
- unter einem ***Screenelement*** ist eine mit der Maus auswählbare Bildschirmschaltfläche zu verstehen

Noch zwei dringende Bitten vorab
Manche Dinge lassen sich mit englischen Begriffen besser beschreiben als mit deutschen. Sehen Sie mir bitte das Pendeln zwischen den beiden Sprachen und ein gewisses Denglisch nach.
Wichtiger noch ist, dass die Verhältnisse rund den Raspberry Pi und seine Software einer unglaublich rasanten Entwicklung unterliegen. Dinge, die gestern noch nicht möglich waren, bereiten heute vielleicht keine Schwierigkeiten mehr oder funktionieren morgen in einem völlig anderen Zusammenhang. Verdammen sie deshalb nicht den Autor, sondern geben Sie bitte entsprechende Hinweise. damit künftige Auflagen des Buches entsprechend aktualisiert werden können.

Vielen herzlichen Dank

Udo Brandes

Inhaltsverzeichnis

Kapitel 1 • Der Mikrocontroller ESP32

Dieses Kapitel ist dem elektronischen Herzstück des Roboterautos gewidmet, dem Mikrocontroller. Dabei geht es um einige wichtige Begriffsdefinitionen, die technischen Daten des ESP32 und die Erstinbetriebnahme an einem PC.

1.1 Einige Begriffe

Mikrocontroller, auch µController, µCs, MCUs (Microcontroller Units) sind Halbleiterbauteile (Chips), die einen Prozessor oder CPU (Central Processing Unit) und zusätzliche Speicher sowie Peripherie enthalten. Peripherie steht dabei für Funktionen wie z. B. I²C-Schnittstelle (Inter-Integrated Circuit), SPI-Schnittstelle (Serial Peripheral Interface), PWM-Ausgänge (Puls-Weiten-Modulation) u.v.a.m., die früher eigene Bauteile auf einer Platine waren. Der historische Vorläufer waren die Einplatinenrechner.

Ein Mikrocontroller ist also ein Ein-Chip-Computersystem. Falls zusätzliche Komponenten auf demselben Chip enthalten sind, wird es als SoC (System-on-Chip) bezeichnet. Espressif selbst nennt diese Bausteine auch wireless-SoCs.

Ein oder mehrere Halbleiterchips, ggf. mit zusätzlichen Komponenten, bilden ein Modul. Module gibt es in Form kleiner bestückter Platinen, verkapselter Minibaugruppen auf Keramikträger o.ä. als "Hybridbausteine", manchmal in Kunststoff vergossen oder in Metallgehäusen zur Abschirmung. Ist das Modul so weit komplett, dass es nur noch mit Strom versorgt und ggf. über Schnittstellen angeschlossen werden muss, nennt man es auch ein *SoP* (System -on-Package).

Abb. 1: ein ESP-WROOM-32-Modul

1.2 Module und Boards im Vergleich

1.2.1 Die Module der ESP32-Serie

Hersteller der ESP32-Module ist die 2008 in China gegründete Firma Espressif Systems (www.espressif.com), deren erstes Produkt der Mikrocontroller ESP8266 war. Der ESP32 folgte 2016 und glänzt im Vergleich mit einem Arduino und einem ESP8266 mit einem deutlichen Mehr an Leistung und Funktionen (vgl. Tabelle 1).

Eigenschaft	ATmega328P	ESP8266	ESP32
MCU	ATmega328P 8-Bit	Xtensa Single-Core 32-bit L-106	Xtensa Dual-Core (oder Single-Core) 32-bit LX6 Mikroprozessor
802.11 b/g/n WiFi / Bluetooth	nein / nein	ja – HT20 / nein	ja / ja
Typische Frequenz	16 MHz	80 MHz	160 MHz
SRAM	2 kB	160 kB	520 kB
Flash	32 kB	SPI Flash bis zu 16 MB	4 MB integriert
GPIOs	23	17	42
Hardware / Software PWM	- / 6 Kanäle	- / 8 Kanäle	4 / 16 Kanäle
SPI / I²C / I2S / UART	1 / 1 / - / 1	2 / 1 / 2 / 2	4 / 2 / 2 / 3
ADC	10-bit	10-bit	12-bit
Arbeitstemperatur	- 40 °C – 105 °C	- 40 °C – 125 °C	- 40 °C – 125 °C
Spannung	1,8 V – 5,5 V	2,3 V – 3,6 V	2,3 V – 3,6 V

Tabelle 1: Eigenschaften Arduino / ESP8266 vs. ESP32

Aktuell verfügt die ESP32-Familie über drei Serien mit verschiedenen Modulen (vgl. Tabelle 2).

Eigenschaft	ESP32-C3	ESP32-S2	ESP32
MCU	32-bit RISC-V single-core Prozessor	Xtensa® single-core 32-bit LX7	ein oder zwei Xtensa® 32-bit LX6 Mikroprozessor(en)
802.11 b/g/n WiFi	2.4 GHz Wi-Fi & Bluetooth LE 5.0	2.4 GHz Wi-Fi	2.4 GHz Wi-Fi & Bluetooth/Bluetooth LE
Frequenz	bis zu 160 MHz	bis zu 240 MHz	80 – 240 MHz

Tabelle 2: Eigenschaften ESP32-Modul-Familie

Einen ausführlichen Beitrag zu ESP32-C3 finden Sie auf YouTube (www.youtube.com/watch?v=VdPsJW6AHqc).

ESP32-S2 verzichtet – anders als die Module der Kernserie ESP32 – auf den zweiten Xtensa LX7-Kern, den Bluetooth-Controller, den CAN-Bus Controller (Car Area Network) und das SD-Karten Interface (SDIO). Dafür hat er einen Controller für USB On-The-Go (USB OTG) eingebaut. Espressif hebt hervor, auch die Controller für kryptografische Funktionen verbessert zu haben. Der Xtensa LX7 läuft weiterhin mit maximal 240 MHz.

Die weiteste Verbreitung und die beste Unterstützung durch Foren hat wohl die ESP32-Serie.

1.2.2 ESP32-Boards

Der ESP32-Chip wird auf einer Vielzahl von Boards verwendet. Der Hersteller Espressif führt selbst einige Module in seinem Programm.

- ESP32-PICO-KIT
- ESP-WROVER-KIT
- ESP32-Dev-KitC
- ESP32-LyraTD-MSC

Einen Überblick über die aktuellen Module und Boards von Espressif finden Sie auf https://docs.espressif.com/projects/esp-idf/en/latest/hw-reference/modules-and-boards.html.

Darüber hinaus wird der ESP32 auch von anderen Herstellern und Distributoren für eigene Boards eingesetzt:

- LoLin32 ESP32 Development Board
- ESP32-OLED Development Board: Dieses Board hat bereits ein OLED-Display.
- ESP32NodeMCU-Board
- u.v.a.m.

1.3 Der ESP32-NodeMCU

Momentan zählt der ESP32-NodeMCU (vgl. Abbildung 2) neben dem ESP32-Dev-KitC V4 zu den gängigsten Entwicklungsboards auf Basis des ESP32. Er ist als Entwicklungsplattform für Einsteiger bestens geeignet; alle ESP32-Pins, die nicht mit besonderen Funktionen belegt sind, wurden herausgeführt und können einfach angeschlossen bzw. verwendet werden. Aus diesem Grund soll es auch Basis für das Roboterauto sein. Selbstverständlich können Sie auch jedes andere Board verwenden. Achten Sie dann auf die Pin-Bezeichnungen.

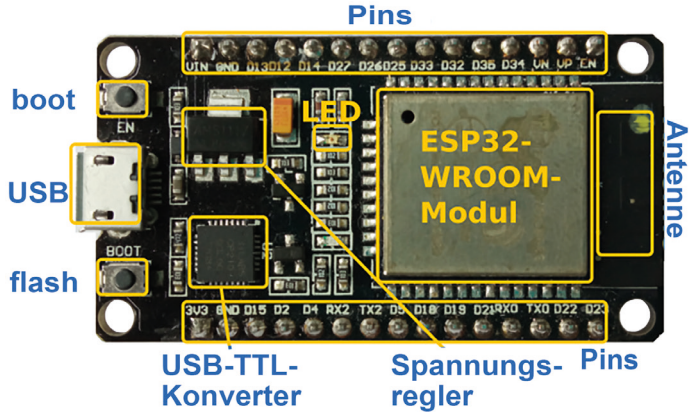


Abb. 2: ESP32-NodeMCU Baugruppen mit elektronischen Komponenten

Der ESP32-NodeMCU kann auf jedes Breadboard gesteckt werden. Mit den Maßen von ca. 53 mm × 25 mm lassen sich komfortabel sehr kompakte Schaltungen realisieren. Nur in den seltensten Fällen werden Sie auf das nackte Modul ausweichen müssen.

ESP32-WROOM-32 Modul

- Antenne
- Header-Pins: Die meisten Pins des ESP-Moduls sind auf die Pin-Header auf der Platine herausgeführt. Es gibt zwei Reihen zu je 19 Pins.
- Reset-Taste: Bei manchen Boards ist diese Taste auch als EN bezeichnet.
- USB-Anschluss: Stromversorgung für das Board und Kommunikationsschnittstelle zwischen einem Computer und dem ESP32-Modul.
- Boot-Taste (Download-Taste): Je nach gewählter Flash-Methode bzw. Entwicklungsumgebung ist die Taste für das Aufspielen der Firmware zu betätigen.
- Häufig beginnt der Flash-Vorgang nach der Programm-Übersetzung selbsttätig. Ist dies nicht der Fall, halten Sie die Boot-Taste gedrückt, bis der Flash-Vorgang eingeleitet ist.
- LED: Sie leuchtet kurz auf, wenn an das Board Spannung angelegt wird. Bei manchen alternativen Boards (z.B. dem ESP32NodeMCU) ist die LED auch mit GPIO02 verbunden.
- USB-UART-Bridge: Verbaut ist eine CP2102 USB to UART Bridge, die Transferraten bis zu 3 Mbps unterstützt.

1.3.1 Die Stromversorgung des ESP32

Das Modul arbeitet mit einer Spannung von typischerweise +3,3 V. Die einzelnen Möglichkeiten für die Stromversorgung sind:

- Micro-USB-Port: Dies wird als hauptsächliche Stromversorgungsoption gesehen, ist aber im Rahmen dieses Projektes nur in der Entwicklungsphase relevant. Da USB normalerweise mit +5 V arbeitet, wird die Spannung durch einen On-Board-Chip auf +3,3 V reduziert. Warnung: Periphere Geräte können nicht ausnahmslos und unbesehen an den 3V3-Ausgang angeschlossen werden, da der On-Board-Chip nur begrenzt belastbar ist.
- 5V/GND-Header-Pins: Dies ist die Stromversorgung im Echtbetrieb des Roboterautos.
- 3V3/GND-Header-Pins

Je nach Einsatz (z. B. intensive WLAN-Nutzung) kann der ESP32 viel Strom ziehen. Laut Datenblatt können dies bis zu 500 mA sein.

Näheres zur Stromversorgung des Roboterautos finden Sie in Kapitel 3.

1.3.2 Das Pinout

In Abbildung 3 sehen Sie das Pinout des ESP32-Dev-KitC V4, soweit es in diesem Buch von Bedeutung ist.

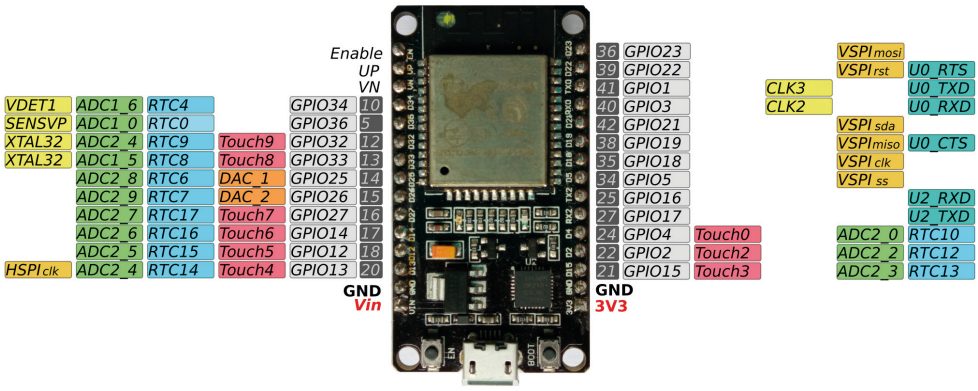


Abb. 3: ESP32-NodeMCU, Pinout

Die einzelnen Pins sind auf dem Board entsprechend bezeichnet. Dabei kann die aufgedruckte Beschriftung je nach Hersteller unterschiedlich sein. So bezeichnen manche Hersteller z. B. den Pin GPIO14 mit G14, manche mit D14 oder auch nur 14.

Darüber hinaus ist das Pinout der verschiedenen Boards nicht normiert. Jeder Hersteller gönnt sich da Freiheiten. Im Extremfall kann sogar bei besonders einfach gehaltenen Klonen der Aufdruck bei einzelnen Pins gänzlich unzutreffend sein, obwohl die eigentliche Zuordnung richtig ist. Deshalb ist es gut investierte Zeit, sich vor jeder Inbetriebnahme mit dem Pinout zu befassen, und grundsätzlich ist natürlich etwas Vorsicht beim Anschluss geboten.

Die GPIO-Pins sind spannungssensitiv. Es kann also zu Schädigungen oder Totalausfall des Mikrocontrollers führen, wenn die an einen Pin angelegte Spannung den Wert von 3,3 V überschreitet.

Abhilfe können Spannungsteiler oder Pegelwandler schaffen. Im einfachsten Fall reichen hierzu zwei Widerstände (vgl. Abbildung 4).

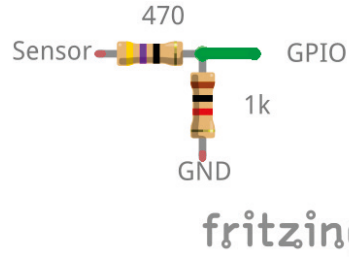


Abb. 4: Spannungsteiler mit zwei Widerständen

Eleganter sind integrierte Pegelwandler (level shifter), die auch bidirektional ausgelegt sein können (vgl. Abbildung 5).

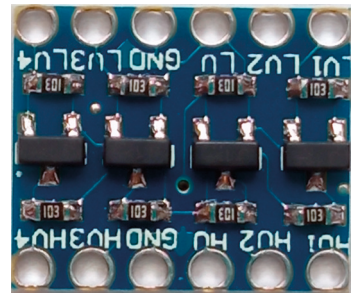


Abb. 5: Levelshifter

Weiter können manche aber nicht alle Pins nur mit maximal 40 mA belastet werden. Eine höhere Belastung kann ebenfalls zur Zerstörung des Boards führen.

1.4 Erste Inbetriebnahme

Verbinden Sie das ESP32-NodeMCU über ein Micro-USB-B-Kabel mit einem PC. Die Betriebszustands-LED sollte einige Male deutlich rot blinken. Es liegt also Spannung an und der On-Board-Spannungsregler arbeitet einwandfrei.

Windows

Windows sollte das Board selbständig erkennen. Dies lässt sich einfach im Geräte-Manager (***Windows → Systemsteuerung → Hardware und Sound → Geräte Manager***) überprüfen (vgl. Abbildung 6).

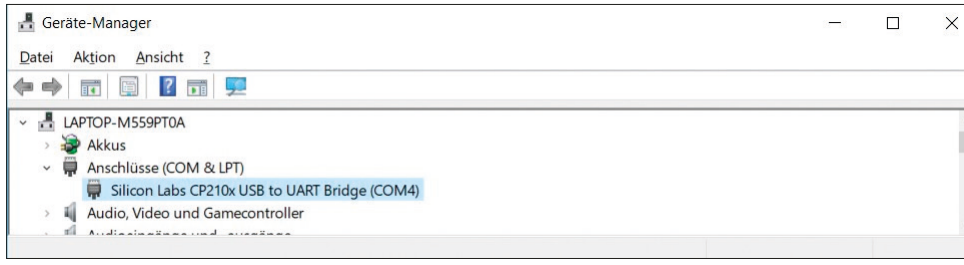


Abb. 6: Windows-Geräte-Manager

Hier ist das Modul an Port COM4 angeschlossen. Die USB zu UART Brücke CP210 wurde erkannt und der nötige Treiber installiert.

Linux

Auch Linux erkennt das Board üblicherweise eigenständig. Die Überprüfung erfolgt in einem Terminal-Fenster mit dem Befehl `lsusb` (vgl. Abbildung 7).

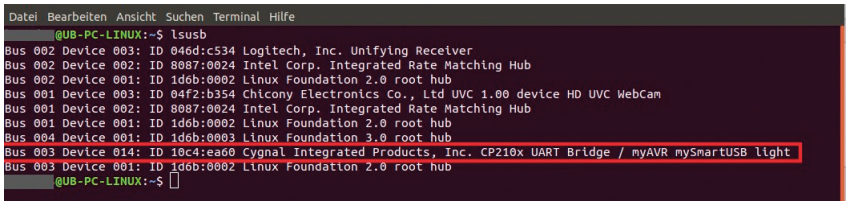


Abb. 7: Linux Terminal lsusb

Kapitel 2 • Die Software entwickeln

Ganz klar, ohne Software kein Roboterauto. Für die Entwicklung der Software sind jedoch einige Dinge nötig. Zu Beginn steht die Entscheidung zugunsten einer Programmiersprache. Für den ESP32 reichen sie von ***MicroPython*** (Python für Mikrocontroller) über ***LUA*** (einer Skriptsprache) bis zu C / C++. Auf letzterem basieren die Beispiele dieses Buchs.

Nun muss aber in C / C++ entwickelter Code übersetzt und mit anderen Modulen zu einem ablauffähigen Programm zusammengesetzt werden. Dies besorgen ***Compiler*** (Übersetzen in einen Maschinen lesbaren Code) und ***Linker*** (Binden der Module zu einem ablauffähigen Programm). Erst danach ist es möglich, den s.g. Maschinencode auf den Mikrocontroller zu flashen.

Die "Umgebung" für all diese Tätigkeiten trägt die Bezeichnung ***Entwicklungsumgebung***. Sie umfasst einen Editor, um den Code zu schreiben, Compiler und Linker sowie Werkzeuge für den Zugriff auf den Programmspeicher des Mikrocontrollers. Wenn diese einzelnen Elemente unter einer einheitlichen Oberfläche zusammengefasst sind, spricht man von einer ***integrierten Entwicklungsumgebung*** (***IDE***, engl. Integrated Development Environment).

Für die Programmierung des ESP32 stehen verschiedene Entwicklungsumgebungen bereit. Die Espressif eigene IDE nennt sich ***ESP-IDF -*** ESP Integrated Development Facicility, vgl. docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/

Hier erfolgt die Programmierung in ***C***. Die Entwicklungsumgebung entfaltet jedoch nur für professionelle und spezielle Anwendungen ihre wahre Leistungsfähigkeit.

Wesentlich eingängiger ist da die ***Arduino-IDE*** (vgl. www.arduino.cc/de). Sie bietet für den Einsteiger beste Möglichkeiten, sich mit der Programmierung von Mikrocontrollern vertraut zu machen. Die Arduino-IDE hat allerdings auch ihre Nachteile. Bösartige Zungen behaupten, sie sei lediglich ein besserer Texteditor und setzen lieber auf Alternative. ***PlatformIO*** (platformio.org/) ist eine solche.

PlatformIO liefert eine Erweiterung für die Open-Source-IDE ***Visual Studio Code*** (VSC); für die Entwicklung zeichnet Microsoft verantwortlich. VSC richtet sich an Entwickler aller Programmiersprachen auf allen Betriebssystemen und erfreut sich vor allem wegen der offenen Architektur seit einigen Jahren immer größerer Beliebtheit. So lassen sich über Erweiterungen Funktionen für unterschiedlichste Programmieraufgaben nachrüsten. Verwechseln Sie jedoch VSC nicht mit Microsoft Visual Studio; dies ist eine proprietäre Software für die Entwicklung von Windows-Software. PlatformIO soll die Grundlage für die Entwicklung der Programme dieses Buchs sein. In einigen Fällen erfolgt jedoch ein Rückgriff auf Bespiele der Arduino-IDE.

Dieses Kapitel stellt Ihnen nach einem Seitenblick auf die Installation von Arduino PlatformIO als mächtige IDE näher vor und geht auf ein paar wesentliche Elemente der Programmierung ein.

2.1 Installation der Arduino-IDE

Diejenigen, die schnell loslegen möchten und einen etwas höheren Installationsaufwand scheuen, beginnen vielleicht mit der Arduino-IDE. Installieren Sie sie wie folgt:

- Wechseln Sie zur Arduino-Download-Seite:

  (www.arduino.cc/en/Main/Software)

- Laden Sie z.B. den Installer für Windows herunter. Führen Sie anschließend den Installer aus.

- Für den ESP32 benötigt auch die Arduino-IDE Boardinformationen bzw. -bibliotheken. Geben Sie zunächst unter

  Einstellungen → zusätzliche Bordverwalter URLs

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

  ein. Wechseln Sie nach ***Werkzeuge → Board → Bordverwalter***, suchen Sie dort nach ESP32 und installieren Sie das angebotene Paket.

2.2 PlatformIO

In diesem Abschnitt geht es darum, PlatformIO zu installieren und als Werkzeug der Programmentwicklung zu nutzen.

2.2.1 VSC Installation

Die folgenden Abschnitte beschreiben die Installation von PlatformIO unter Windows und Linux.

Windows

Folgen Sie für eine Installation unter Windows folgenden Schritten:

Schritt 1: VSC -Installer herunterladen

Laden Sie sich den VSC-Installer von code.visualstudio.com/download herunter.

Schritt 2: VSC-Installer ausführen

Führen Sie den VSC-Installer aus.

Akzeptieren Sie die Lizenzvereinbarung und gehen Sie auf ***weiter***.

Wählen Sie den gewünschten Zielordner aus und dann ***weiter***.

Wählen Sie den gewünschten Startmenü-Ordner aus und klicken Sie auf ***Weiter***. Das Setup erlaubt Ihnen nun, zusätzliche Aufgaben auszuwählen (vgl. Abbildung 1).

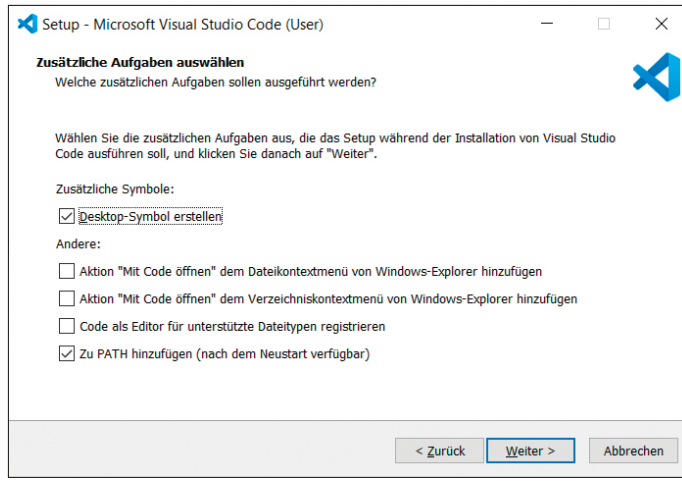


Abb. 1: VSC installieren Windows zusätzliche Aufgaben

Wählen Sie diese Ihren Wünschen entsprechend aus und gehen Sie auf ***Weiter***. Es folgen noch ein paar Bildschirme, die Sie mit ***installieren*** und ***fertigstellen*** quittieren. Bei automatischem Start gelangen Sie zum Startbildschirm von Visual Studio Code (vgl. Abbildung 2).

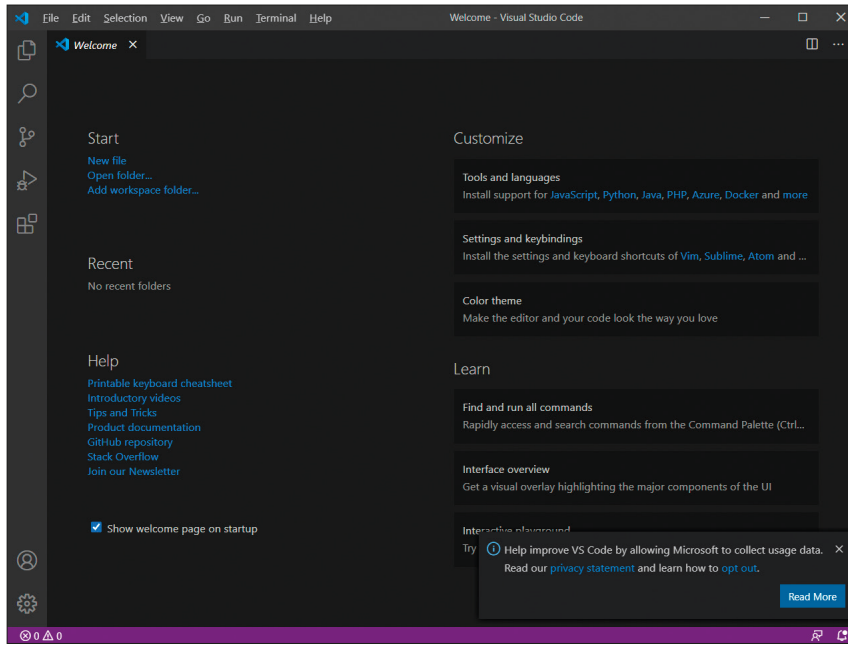


Abb. 2: VSC-Startbildschirm

Schritt 3: Python installieren

Espressif nutzt für Flash- und Ausleseprozesse beim ESP32 ***Python***-Skripte. Die Pakete einer ESP-IDF- oder Arduino-Installation bringen alles Erforderliche mit. Bei einer PlatformIO-Installation müssen Sie selbst für eine Python-Umgebung Sorge tragen.

Besuchen Sie die Python-Download-Seite www.python.org/downloads/ (vgl. Abbildung 3).

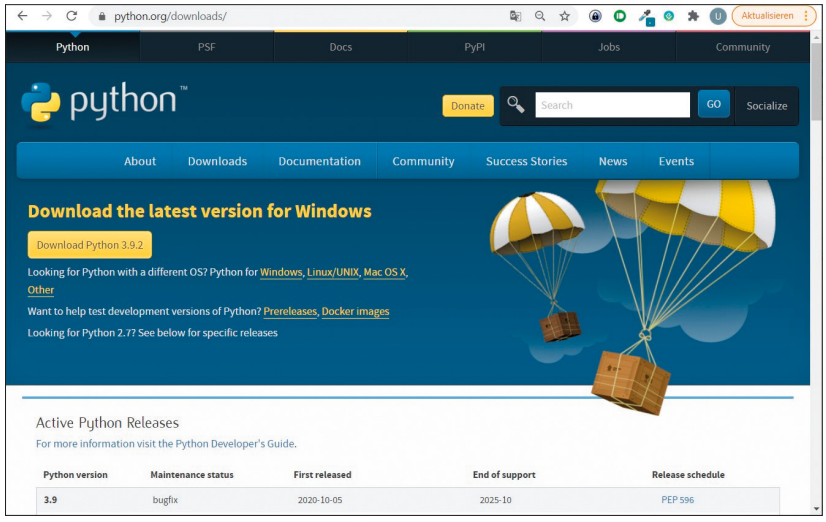


Abb. 3: Python Installer Download-Seite

Klicken Sie auf die Schaltfläche ***DOWNLOAD PYTHON 3.9.2*** (dies ist zurzeit die aktuelle stabile Version), um den Installer für Windows herunter zu laden. Nach einem Doppelklick auf den Installer erscheint das Startfenster für den Installationsvorgang (vgl. Abbildung 4).

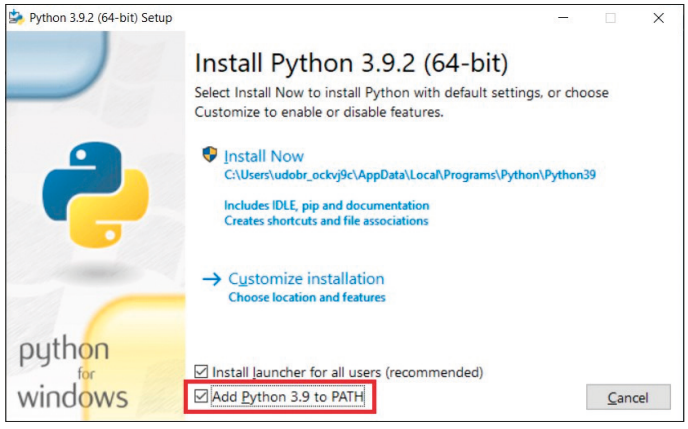


Abb. 4: Python Installer Startfenster

Achten Sie darauf, dass Kontrollkästchen für das Hinzufügen von Python zur Path-Variable zu aktivieren! Klicken Sie dann auf ***Install Now***. Es sollte eine Erfolgsmeldung erscheinen (vgl. Abbildung 5).

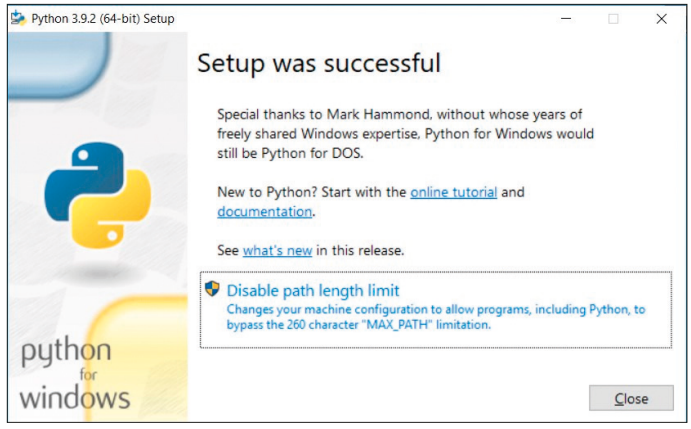


Abb. 5: Python Installer Abschlussmeldung

Der Python-Installer bietet Ihnen unter Windows direkt an, die Beschränkung der PATH-Länge aufzuheben. Das macht die Arbeit mit langen Pfadnamen komfortabler und vermeidet umständliche Abkürzungen. Anschließend können Sie mit ***Close*** den Installer verlassen.

In Ihrem Startmenu haben Sie nun einen Eintrag IDLE (Python-3.10 64bit); mit einem Klick gelangen Sie in die Python-Shell, in der Sie die Programme entwickeln.

Linux
Die Installation unter Linux verläuft prinzipiell ähnlich wie unter Windows. Das Einrichten von Python ist entbehrlich, da auf Linux die Pakete üblicherweise bereits vorhanden sind.

Schritt 1: Pakete herunterladen
Laden Sie die Pakete von code.visualstudio.com/download entsprechend Ihrer Linux-Distribution herunter.

Schritt 2: Pakete installieren
Wechseln Sie in das Download-Verzeichnis. Dort befindet sich z.B. für Ubuntu eine .deb-Datei (z.B. code-1.53.2-1613044664_amd64.deb). Installieren Sie die Pakete mit (vgl. Abbildung 6):

```
sudo apt install code-1.53.2-1613044664_amd64.deb
```

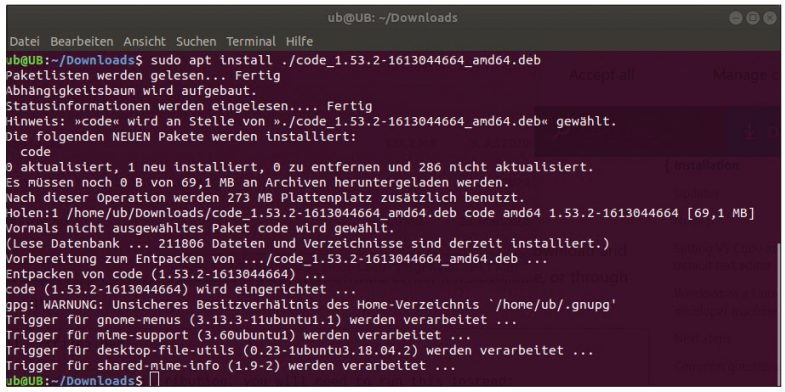


Abb. 6: VSC installieren Linux

In der Aktivitäten-Übersicht von Ubuntu finden Sie ein Icon, um Visual Studio Code zu starten und zum Startbildschirm (vgl. Abbildung 9) zu gelangen.

Jetzt fehlt noch die Erweiterung PlatformIO.

2.2.2 Erweiterung PlatformIO hinzufügen

Hier geht es darum, die Erweiterung PlatformIO VSC hinzuzufügen. Klicken Sie auf das Puzzle-Symbol in der linken Seitenleiste (vgl. Abbildung 7).

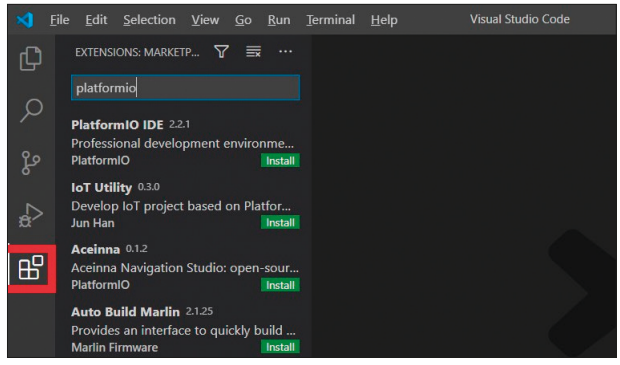


Abb. 7: VSC PlatformIO hinzufügen

Tragen Sie in das Suchfeld *PlatformIO* ein, damit die Erweiterung in der angebotenen Liste erscheint. Klicken Sie auf ***Install***. Der Prozess dauert eine Weile und verlangt nach dem Abschluss einen Restart von VSC. Die Erweiterungen sind nun aufgeführt (vgl. Abbildung 8).