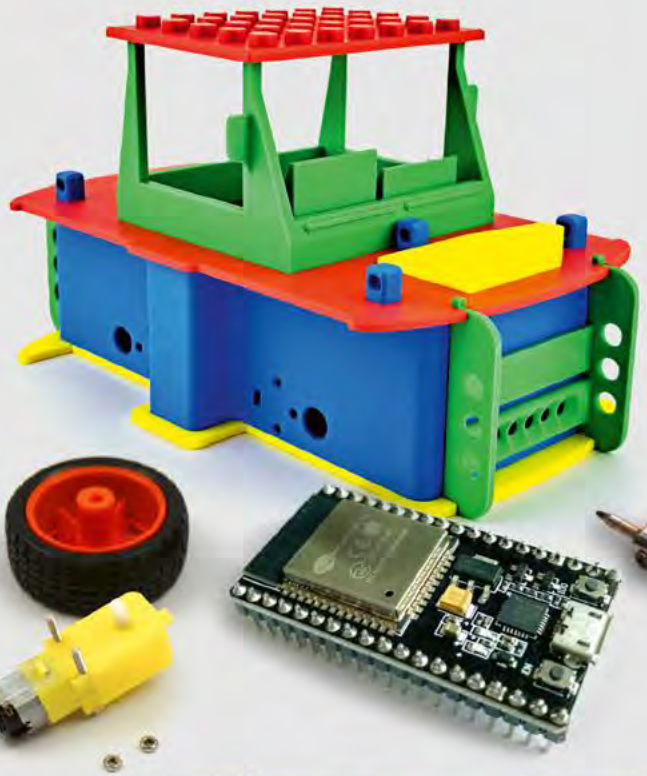
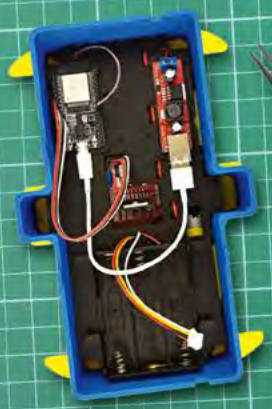
Ingmar Stapel

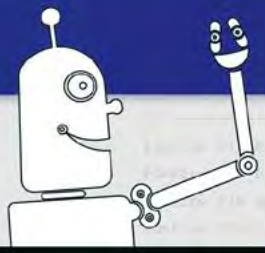
Empfohlen von Make:





Roboter-Autos mit dem ESP32

Bauen, programmieren, steuern



- Vom Roboterbau über die KI-gestützte Programmierung bis zum Training eines selbstfahrenden Modells
- Ferngesteuerte und autonome Modelle selbst bauen
- Ohne Vorwissen einsteigen und direkt losfahren

Alle Codebeispiele und Vorlagen zum Download

Rheinwerk Computing

Vorwort

Mit diesem Buch hältst du nicht nur eine Anleitung zum Bau eines Roboter-Autos in deinen Händen, sondern eine Einladung zu einem dreiteiligen Abenteuer des Entdeckens, Programmierens und Erschaffens. Ganz nach dem Motto »Learning by Doing« wirst du dir mit diesem Buch die spannenden Themen der Elektronik, der künstlichen Intelligenz und der Robotik spielerisch erarbeiten.

Im **ersten Teil** des Buches lernst du einige Grundlagen wie zum Beispiel zum Thema Löten oder der Spannungsversorgung im Roboter-Auto. Mit diesem Basiswissen ausgestattet wird dir der Bau deines ferngesteuerten Roboter-Autos leicht von der Hand gehen. Danach folgt der **zweite Teil,** der etwas theoretischer ist und dir mit der einzigartigen »Prompt-Schule« das Wissen an die Hand gibt, wie du im Dialog mit modernen KI-Assistenten dein Roboter-Auto programmierst – eine Fähigkeit, die dir weit über das Roboter-Auto-Projekt hinaus helfen wird. Im **dritten Teil** verwandeln wir zusammen dein Roboter-Auto schließlich in einen intelligenten OpenBot, der ein modernes Smartphone als Gehirn nutzt, um zum Beispiel mit selbst trainierten KI-Modellen das autonome Fahren zu lernen. Du lernst auch das Roboter-Auto visuell mit einer Block-Programmiersprache zu programmieren und bringst so deinem Roboter-Auto weitere KI-Fähigkeiten bei, abseits vom autonomen Fahren durch einfaches Kombinieren der verschiedenen Coding-Blöcke.

Zugegeben, bei einem Do-it-yourself-Projekt dieser Komplexität wird nicht immer alles auf Anhieb fehlerfrei funktionieren. Aber gerade das gemeinsame Tüfteln, das Suchen nach Lösungen und das Erfolgserlebnis, wenn das Roboter-Auto die ersten selbstständigen Runden im Zimmer dreht, sind unglaublich lohnend und machen enorm viel Spaß!

Die Idee zu diesem Buch und dem Roboter-Auto entstand aus einem ganz persönlichen Projekt: dem gemeinsamen Bau von Roboter-Autos mit meiner Tochter. Es war wunderbar zu sehen, wie ihre Neugier wuchs, wie sie mithalf, Bauteile sortierte und die ersten Kabelverbindungen im Roboter-Auto anschloss. Aus diesem Vater-Tochter-Projekt, das uns viele schöne gemeinsame Stunden bescherte, reifte die Idee, diese Erfahrung zu teilen.

So wird dieses Roboter-Auto auch Teil eines besonderen Projekttages in der 2. Klasse meiner Tochter sein. Die Vorstellung, wie die Kinderaugen leuchten werden, wenn sie ihren eigenen, selbstgebauten Roboter zum Leben erwecken, ist eine wunderbare Motivation. Dieses Buch soll auch anderen Eltern, Lehrern oder Jugendlichen die Möglichkeit geben, solche unvergesslichen Momente des Lernens und Schaffens zu erleben.

In diesem Buch habe ich all mein Wissen und meine Erfahrungen zusammengetragen, um dich auf dieser spannenden Reise zu begleiten: vom physischen Bau des Roboters über die revolutionäre Art des Programmierens im Dialog mit KI-Assistenten bis hin zur Krönung – der Erschaffung eines autonomen Fahrzeugs mit eigenem, selbst trainiertem KI-Modell.

Wenn du also bereit bist, dich auf diese spannende Entdeckungsreise zu begeben, dann leg los! Bau nicht nur ein Roboter-Auto, sondern schaffe deine ganz persönliche Erfolgsgeschichte – vielleicht ja auch gemeinsam mit der nächsten Generation von Tüftlern und Makern, die du beim Bau ihres eigenen Roboter-Autos begleitest, so wie ich mit meiner Tochter zusammen.

Viel Spaß beim Bauen, Programmieren und Entdecken!

Herzlichen Dank

An allererster Stelle gilt mein tiefster Dank meiner Frau und meiner wunderbaren Tochter. Ohne eure Geduld, euer Verständnis und eure tatkräftige Unterstützung wäre auch dieses Buch nicht entstanden. Es gab viele Stunden, in denen ich am Roboter tüftelte, an den Texten und Bildern arbeitete – und besonders viel Zeit in die Gestaltung des 3D-Modells investierte.

Besonderer Dank geht an meine Tochter, die nicht nur geduldig meine »Bastelzeiten« hingenommen hat, sondern dieses Buch auch maßgeblich mitgestaltet hat. Ihre Ideen sind direkt in das 3D-Design des Roboter-Autos eingeflossen, und die Begeisterung, die der 3D-Drucker bei uns beiden ausgelöst hat oder wenn unser KI-Modell einen Banana-Dance ausführte, war eine tolle Motivation, immer weiter an diesem Buch zu schreiben. Dieses gemeinsame Erschaffen, Ausprobieren, Annehmen von Herausforderungen und die pure Freude am fahrenden Roboter-Auto haben dieses Projekt so besonders gemacht. Du warst meine wichtigste Co-Autorin und Testerin auf dieser Reise!

Die Idee, dieses Projekt als Projekttag in deiner 2. Klasse umzusetzen, war eine wunderbare Inspiration und hat gezeigt, wie viel Begeisterung in jungen Köpfen steckt, wenn Technik greifbar und zum Leben erweckt wird. Ein großes Dankeschön geht an deine Klassenlehrerin J. M. und die Schulleitung für diese tolle Möglichkeit.

Ein herzliches Dankeschön geht auch an Jörg Rippel, der mich mit seinem Fachwissen und erstklassigen Humor beim Schreiben des Buches unterstützt hat. Nicht zuletzt möchte ich mich bei allen Makern, Tüftlern und DIY-Enthusiasten wie dem OpenBot-Projekt bedanken, die durch ihre Projekte und Veröffentlichungen eine unendliche Quelle der Inspiration darstellen.

Vielen Dank euch allen!

TEIL I

Baue dein eigenes Roboter-Auto

Kapitel 1
Dein Roboter-Auto zum Selberbau

Vielleicht stellst du dir die Frage: Warum soll ich überhaupt ein Roboter-Auto selbst bauen? Es gibt diese doch fix und fertig zu kaufen! In diesem Kapitel versuche ich, eine Antwort zu geben, warum es so wichtig ist, selbst etwas zu erschaffen und weshalb der Bau eines eigenen Roboter-Autos sehr spannend und zugleich lehrreich sein kann. Auch gehe ich darauf ein, für wen alles dieses Projekt gedacht ist.

Jetzt tauchen wir tiefer in dein persönliches Roboter-Auto-Projekt ein und klären, warum dieses Projekt aus meiner Sicht so lohnenswert ist. Dieses Buch nimmt dich über seine verschiedenen Kapitel mit auf eine spannende Entdeckungsreise, bei der du am Ende dein ganz eigenes Roboter-Auto gebaut und programmiert hast – und dabei richtig viel gelernt hast über Technik, moderne KI-Assistenten und mit Sicherheit auch über dich selbst! In den folgenden drei Kapiteln erfährst du, was dieses Roboter-Auto-Projekt so besonders macht und was dich in diesem Buch erwartet.

1.1 Warum überhaupt ein Roboter-Auto bauen?

Du fragst dich vielleicht, warum du dir die Mühe machen solltest, ein Roboter-Auto selbst zu bauen, wenn doch das Angebot in Onlineshops so groß ist, dass du dich kaum für ein Modell entscheiden kannst. Die Angebote sind vielversprechend, die Modelle toll designt und überbieten sich mit Funktionen und technischen Raffinessen.

Es macht aber einen riesigen Unterschied, etwas selbst zu erschaffen oder einfach nur zu kaufen! Wenn du dein eigenes Roboter-Auto baust, lernst du dabei nicht nur, wie es funktioniert, sondern auch, wie man Probleme löst, logisch denkt und eigene Ideen in die Tat umsetzt.

Stell dir vor, dein Roboter-Auto fährt zum ersten Mal los – ein Moment voller Stolz und Zufriedenheit, wie ihn dir der profane Klick auf einen Sofort-kaufen-Button nicht geben

kann! Du hast die Elektronik verkabelt, den Code geschrieben und jedes Teil zusammengefügt. Du hast mit Sicherheit Höhen und Tiefen erlebt und eben nicht aufgegeben, sondern dich mit dem Projekt Roboter-Auto auseinandergesetzt. Das ist viel befriedigender, als auf den Paketboten zu warten und ein fertiges Spielzeug auszupacken, oder? Du wirst verstehen, was unter der Haube passiert und wie all die Bauteile zusammenarbeiten, die in Summe dein Roboter-Auto ausmachen. Du bist auch in der Lage, dein Roboter-Auto zu reparieren oder anzupassen. Dieses Wissen und die Erfahrungen sind Gold wert, egal ob du später Ingenieurin, Programmiererin oder ein kreativ denkender Kopf wirst.

Selbermachen macht unglaublich viel Spaß, du lernst dabei eine Menge und hältst am Ende etwas Einzigartiges in den Händen, das für dich einen viel größeren Wert hat als ein Artikel in einem Warenkorb.

So ein Selbermachprojekt zeichnet sich durch Folgendes aus:

- **Es ist spannend:** Vom ersten elektronischen Bauteil bis zur ersten selbstständigen Fahrt – jeder Schritt ist ein kleines Erfolgserlebnis.
- **Du lernst dabei:** Ganz nebenbei lernst du viel über Elektronik, verstehst, wie Motoren und Sensoren funktionieren und wie man einem Mikrocontroller beibringt, was er tun soll.
- **Du programmierst auf eine neue Art:** Statt selbst Code zu schreiben, besuchst du unsere »Prompt-Schule«. Dort lernst du, wie du einem KI-Assistenten die richtigen Anweisungen gibst, damit er die Programmierarbeit für dich erledigt.
- **Dein Roboter ist einzigartig:** Weil du ihn selbst gebaut und vielleicht sogar schon mit 3D-gedruckten Teilen individualisiert hast, gibt es ihn so kein zweites Mal.
- **Du verstehst die Technik:** Du bist kein einfacher Benutzer oder Käufer, sondern du weißt genau, was in deinem Roboter steckt und warum es so funktioniert. Das ist ein tolles Gefühl!
- **Es macht Spaß, Probleme zu lösen:** Manchmal funktioniert etwas nicht sofort. Das ist normal! Dann ist Detektivarbeit gefragt. Die Suche nach Fehlern und das Finden von Lösungen sind super Lernmomente, die dir auch Durchhaltevermögen beibringen.
- **Du erlebst eine Verwandlung:** Dein Roboter-Auto wird sich von einem ferngesteuerten Roboter-Auto in einen intelligenten OpenBot verwandeln, der ein Smartphone als Gehirn nutzt und mit selbst trainierten KI-Modellen autonom fährt.

Dieses Projekt ist nicht nur Basteln, es ist eine praktische Einführung in MINT-Themen (Mathematik, Informatik, Naturwissenschaft, Technik) auf eine Art, die Spaß macht und motiviert.

1.2 Für wen ist dieses Projekt geeignet?

Dieses Buch und das Roboter-Auto-Projekt an sich sind für alle gedacht, die neugierig sind und gerne etwas Neues in ihrem Leben lernen möchten, von dem sie vielleicht noch keine konkrete Vorstellung haben, was es ist und ob sie es können. Es ist perfekt, wenn du eine Herausforderung suchst, wenn du wissen willst, wie Technik funktioniert und wie man Roboter selbst zum Leben erweckt.

- **Für junge Technik-Entdecker (ab 10 Jahren):** Die Anleitung ist so geschrieben, dass du auch als Kind oder Jugendlicher ab etwa 10 Jahren mit der Unterstützung zum Beispiel deiner Eltern gut mitkommst. Wenn du jünger, aber trotzdem total begeistert bist, kannst du dich unter Anleitung deiner Eltern oder Lehrer an das Projekt heranwagen. Dann klappt es auch für die ganz kleinen Technikfans!
- **Für Roboter-Neulinge:** Du brauchst absolut keine Vorkenntnisse im Programmieren oder in der Elektronik. Wir fangen bei null an und erklären dir alles Schritt für Schritt. Und das Beste: Dank unserer Prompt-Schule musst du kein Programmier-Profi sein, um deinem Roboter-Auto coole Funktionen beizubringen. Du wirst staunen, was du alles mithilfe von KI-Assistenten schaffen kannst!
- **Für Eltern, Lehrer und Betreuer:** Wenn du Kindern oder Jugendlichen einen spannenden und praxisnahen Einblick in die Welt der Technik geben möchtest, ist dieses Projekt genau richtig. Es eignet sich hervorragend als gemeinsames Familienprojekt, für eine Schul-AG oder einfach zum Ausprobieren in der Freizeit.

1.3 Mehr als nur Likes: dein Projekt, deine Zukunft

In unserer heutigen modernen Welt mit ihren Annehmlichkeiten ist es leicht, sich in den unzähligen Angeboten von Social Media und leichten Unterhaltungen zu verlieren. Ein Like hier, ein schneller Kommentar, ein kurzes Short dort – und schon sind Stunden vergangen, obwohl man noch gar nicht alles der unzähligen Freunde und Follower gesehen hat. Das ist zwar manchmal entspannend und zugleich auch stressig, um ja nichts zu verpassen, aber bringt dich das wirklich weiter?

Stell dir vor, du könntest diese Zeit nutzen, um etwas Bleibendes zu schaffen, etwas, was dich schlauer macht, auf das du Jahre später noch stolz sein kannst und von dem du gerne erzählst. Genau darum geht es bei diesem Roboter-Auto-Projekt!

Gerade weil die Schule häufig nicht alle MINT-Themen (Mathematik, Informatik, Naturwissenschaft, Technik) so vermitteln kann, wie es für dich spannend und praxisnah wäre, liegt es an dir, deine eigenen Interessen zu entdecken und zu vertiefen. Dieses Buch bietet dir die Chance, genau das zu tun.

Anstatt passiv Inhalte zu konsumieren, wirst du aktiv gestalten und lernen. Du wirst nicht nur etwas über Elektronik, das Promptschreiben und künstliche Intelligenz erfahren, sondern auch wichtige Fähigkeiten wie Problemlösung, Ausdauer, Konzentration und kreatives Denken trainieren. Diese Fähigkeiten sind in unserer schnelllebigen Welt mit ihren technischen Errungenschaften unglaublich wertvoll geworden – viel wertvoller als das nächste Like oder die neuesten KI-generierten Videos angesagter Vlog Channels.

Nutze deine Zeit klug. Investiere sie in Projekte, die dich inspirieren, die dich fordern und die dir ein echtes anhaltendes Gefühl von Erfolg geben. Dein Roboter-Auto wartet darauf, von dir gebaut, zum Leben erweckt und angefasst zu werden. Es ist deine Chance, nicht nur ein cooles Gerät zu bauen, sondern auch in dich selbst zu investieren und deine Zukunft und vielleicht die Zukunft auch vieler anderer aktiv mitzugestalten.

Das Wichtigste ist deine Lust am Entdecken und Ausprobieren! Egal, ob du allein tüftelst oder gemeinsam mit anderen – dieses Projekt wird dich begeistern und dir zeigen, wie viel Spaß es macht, selbst etwas zu erschaffen.

Bist du bereit, dein eigenes Roboter-Auto zu bauen und zu programmieren? Dann lass uns loslegen!

Kapitel 2

Die richtige Ausrüstung: Werkzeuge für das Roboter-Auto-Projekt

Ein guter Handwerker braucht gutes Werkzeug! Bevor wir die Einzelteile deines Roboters zusammenfügen, werfen wir einen Blick auf die Helferlein, die dir den Bau erleichtern und für feste Verbindungen sorgen. Rüste dich richtig aus, damit dein Roboter-Abenteuer reibungslos startet!

Um aus den vielen Einzelteilen ein funktionierendes Roboter-Auto zu bauen, brauchst du nicht nur die elektronischen Komponenten und ein Chassis, sondern auch ein paar Werkzeuge. Das richtige Werkzeug kann den Zusammenbau erheblich erleichtern, für stabilere Verbindungen sorgen und die Arbeit sicherer machen. In diesem Kapitel stelle ich dir die Werkzeuge vor, die du für dein Roboter-Auto-Projekt benötigst, und erkläre kurz, wofür du sie brauchst. Keine Sorge, es sind keine teuren Spezialwerkzeuge, sondern meist Dinge, die man in einem Elektronik- oder Hobbykeller findet oder günstig besorgen kann, wenn du sie noch nicht hast.

2.1 Warum die richtigen Werkzeuge wichtig sind

Stell dir vor, du willst ein Bild aufhängen, hast aber keinen Hammer für den Nagel. Ähnlich ist es beim Bau eines Elektronikprojekts. Die passenden Werkzeuge helfen dir dabei:

- **Präzise zu arbeiten:** Elektronische Bauteile sind klein. Mit dem richtigen Werkzeug kannst du präzise Verbindungen herstellen und Beschädigungen vermeiden.
- **Zeit zu sparen:** Die Arbeit geht schneller und einfacher von der Hand, wenn du das passende Werkzeug hast.
- **Sichere Verbindungen zu schaffen:** Insbesondere beim Löten sorgen die richtigen Werkzeuge für feste und zuverlässige elektrische Verbindungen.

- **Sicher zu arbeiten:** Einige Werkzeuge, wie zum Beispiel ein Seitenschneider oder Lötkolben, erfordern etwas Vorsicht. Meine Tipps und Hinweise helfen dir, sicher mit ihnen umzugehen.

2.2 Die wichtigsten Werkzeuge für dein Roboter-Auto-Projekt

Hier findest du eine übersichtliche Liste der Werkzeuge, die du für den Bau deines Roboter-Autos benötigen wirst:

- **Seitenschneider:** Du benötigst einen kleinen Seitenschneider, um Drähte, Kabelbinder oder überstehende Pins abzukneifen. Ein feiner Seitenschneider für Elektronikarbeiten ist ideal, da er scharfe Klingen für saubere Schnitte hat.
- **Skalpell:** Sehr hilfreich ist ein richtig scharfes Messer, ein sogenanntes Skalpell, wenn du dein Roboter-Auto-Chassis aus fester Pappe bauen möchtest. Mit so einem scharfen Messer ist der Zuschnitt sehr leicht, und vor allem hast du ein ganz sauberes Schnittkantenbild.
- **Abisolierzange:** Um die Isolierung von Kabeln sauber und einfach zu entfernen, ist eine Abisolierzange sehr hilfreich. Es gibt einfache Modelle oder solche, die sich automatisch auf den Kabeldurchmesser einstellen.
- **Feile:** Es kann immer wieder vorkommen, dass etwas gefeilt werden muss – um vielleicht etwas glatt zu feilen, nachdem es abgeschnitten wurde. Daher solltest du eine kleine flache Metallfeile für dieses Projekt bereitlegen.
- **Schrumpfschlauch:** Wenn du abisolierte Kabel, die du miteinander verbunden hast, wieder sicher isolieren möchtest, dann brauchst du einen Schrumpfschlauch. Das ist ein dünner Schlauch (zum Beispiel mit ca. 4 mm Durchmesser), der sich bei Wärme stark zusammenzieht. So umschließt er die blanke Lötstelle fest und isoliert sie wieder zuverlässig. Achte darauf, dass der Schrumpfschlauch die offene Stelle gut überdeckt, am besten mit jeweils etwa 7 mm Überstand an Anfang und Ende der Lötstelle.
- **Schraubendreher-Set:** Du benötigst kleine Schraubendreher, insbesondere Kreuzschlitz-Schraubendreher (PH0 oder PH1) für die Schrauben des Motortreibers (L298N) und für eventuelle Schrauben im Chassis. Ein Set mit verschiedenen Größen ist immer praktisch.
- **Pinzette:** Eine feine Pinzette hilft dir, kleine Bauteile zu greifen und zu positionieren, insbesondere beim Löten oder beim Einstecken von Jumper-Kabeln an eng zugänglichen Stellen.
- **Heißklebepistole (optional, aber sehr nützlich):** Eine kleine Heißklebepistole ist ideal, um Komponenten schnell und einfach im Chassis zu fixieren oder Kabel sauber zu

verlegen. Der Vorteil von Heißkleber ist, dass er die Teile nicht dauerhaft verbindet und bei Bedarf leicht wieder gelöst werden kann (im Gegensatz zu Sekundenkleber).

- **Lötkolben und Lötzinn:** Viele elektronische Verbindungen, insbesondere an den Pins des ESP32 oder kleineren Modulen, werden gelötet. Ein einfacher Lötkolben mit einstellbarer Temperatur (ca. 30 bis 40 Watt Leistung) und Elektronik-Lötzinn (idealerweise bleifrei) sind unerlässlich. Wähle einen Lötkolben, der gut in der Hand liegt und nicht zu klobig ist.

Wichtiger Sicherheitshinweis

Ein Lötkolben wird sehr heiß! Berühre niemals die Spitze, und arbeite immer auf einer feuerfesten Unterlage. Sorge für gute Belüftung, da beim Löten Dämpfe entstehen. Lasse den Lötkolben niemals unbeaufsichtigt eingeschaltet.

- **»Dritte Hand«, Helferlein mit Klemmen:** Dieses Werkzeug hat meist zwei bewegliche Klemmen und oft eine Lupe. Es ist unglaublich nützlich, um kleine Bauteile oder Kabel festzuhalten, während du lötest. So hast du beide Hände frei für den Lötkolben und das Lötzinn.
- **Sekundenkleber (Gel):** Für das Verkleben der 3D-gedruckten PLA-Teile oder das Fixieren von Kabeln ist ein guter Sekundenkleber (am besten als Gel, da er nicht so schnell verläuft) sehr nützlich. Achte darauf, einen Kleber zu wählen, der für Kunststoffe geeignet ist.
- **UHU Hart oder Kunststoffkleber:** Eine Alternative oder Ergänzung zu Sekundenkleber für das Verkleben der 3D-gedruckten PLA-Teile.
- **Schutzbrille (empfohlen):** Beim Abkneifen von Drähten mit dem Seitenschneider können kleine Drahtstücke wegfliegen. Eine Schutzbrille schützt deine Augen. Auch beim Umgang mit Klebstoffen ist Vorsicht geboten.
- **Klebeband:** Isolierband oder doppelseitiges Klebeband kann beim vorübergehenden Fixieren von Komponenten oder zum Bündeln von Kabeln nützlich sein.
- **Kabelbinder:** Kleine Kabelbinder helfen dir, Kabel ordentlich zu verlegen und zu fixieren, um Kabelsalat zu vermeiden und sicherzustellen, dass sich keine Kabel in den Rädern verfangen.

2.3 Gut verlötet ist halb gewonnen – Grundlagen des Lötens

Für den Bau deines Roboter-Autos wirst du immer wieder Bauteile zum Beispiel mit Kabeln miteinander verbinden müssen – und Löten ist dabei eine wichtige und grundlegende Fähigkeit. Keine Sorge, wenn du noch nie einen Lötkolben in der Hand hattest!

Mit ein wenig Übung und dem richtigen Wissen gelingen dir bald saubere und stabile Lötverbindungen. Dieses Kapitel führt dich in die Grundlagen des Lötens ein und zeigt dir, worauf es ankommt.

2.3.1 Der Lötprozess – eine Anleitung zur perfekten Lötstelle

Beim Löten ist es wichtig, dass du den Lötkolben vor Beginn des eigentlichen Lötvorgangs auf seine Arbeitstemperatur aufheizt. Das dauert meist nur wenige Minuten. Wenn du zum Beispiel zwei Bauelemente oder ein Kabel an eine Platine löten möchtest, ist es empfehlenswert, die zu verbindenden Metallflächen zuerst einzeln dünn mit Lötzinn zu überziehen – diesen Vorgang nennt man *Verzinnen*.

Nach dem Verzinnen führst du die beiden Teile zusammen, erwärmst sie gleichzeitig mit dem Lötkolben und gibst bei Bedarf noch ein klein wenig frisches Lötzinn hinzu. Durch das vorherige Verzinnen fließt das Lot beim verbindenden Lötvorgang wunderschön ineinander. So entsteht eine feste, glänzende und elektrisch gut leitende Lötstelle. Mit dieser Methode vermeidest du auch effektiv sogenannte *kalte Lötstellen*. Das sind fehlerhafte Lötverbindungen, die oft matt aussehen, brüchig sind und den elektrischen Strom nicht zuverlässig oder gar nicht leiten – eine häufige Fehlerquelle bei Elektronikprojekten!

Übung macht den Meister: Leg doch gleich einmal los und übe das Löten! Nimm dir zwei einfache Litzenkabel. Ein Litzenkabel besteht nicht aus einem einzelnen dicken Draht, sondern aus vielen haarfeinen Kupferdrähten, die miteinander verdrillt sind. Das macht das Kabel besonders flexibel und bruchsicher – perfekt für bewegliche Teile in deinem Roboter-Auto.

Isoliere die Enden des Litzenkabels etwa 5 bis 7 mm ab, und verdrille die feinen Kupferdrähte leicht. Verzinne nun jedes Kabelende einzeln: Halte dazu die Lötspitze an das Kabelende und führe etwas Lötzinn zu, bis es schön verläuft und die Drähte umschließt. Anschließend hältst du die beiden verzinnten Kabelenden aneinander, erwärmst sie mit dem Lötkolben und lässt sie sauber miteinander verschmelzen. Nach dem Abkühlen sollte eine stabile Verbindung entstanden sein.

2.3.2 Wichtiges Lötzubehör – was du noch brauchst

Neben einem guten Lötkolben oder einer Lötstation gibt es ein paar unverzichtbare Verbrauchsmaterialien und nützliche Helfer.

Lötzinn und die richtige Temperatur

Es gibt verschiedenes Lötzinn (auch Lot genannt), das für Elektronikarbeiten eingesetzt wird. Die Zusammensetzung des Lötzinns bestimmt dessen Schmelztemperatur und damit auch die Temperatur, die du an deiner Lötstation einstellen solltest.

Kennzeichnung Lötzinn	Schmelztemperatur-bereich	Empfohlene Arbeitstemperatur des Lötkolbens
S-Sn60Pb40	180–190 °C	280–330 °C
S-Pb50Sn50	180–215 °C	290–340 °C
S-Pb60Sn40	180–235 °C	290–340 °C
S-Sn97Ag3	220–225 °C	330–380 °C
S-Sn99Cu1	230 °C	330–380 °C
S-Sn97Cu3	230–250 °C	340–390 °C

Tabelle 2.1 Schmelztemperaturen gängiger Lötzinn-Typen

Für meine Lötarbeiten am Roboter-Auto verwende ich das Lötzinn S-Sn97Ag3 anstatt des Lötzinnes S-Sn60Pb40, das Blei enthält (60 % Zinn, 40 % Blei). Das bleifreie Lötzinn wie S-Sn99Cu1 (Zinn mit Kupferanteil) oder als Alternative S-Sn97Ag3 (Zinn mit Silberanteil) benötigt zwar eine leicht höhere Löttemperatur (meist um 220–230 °C), ist aber für dich und die Umwelt gesünder, da diese Lötzinne kein Blei (Pb) enthalten, das giftig ist.

Vorsicht beim Umgang mit Flussmitteln!

Egal ob das Flussmittel im Lötzinn integriert ist oder du es zusätzlich aufträgst: Sorge immer für eine gute Belüftung deines Arbeitsplatzes. Die beim Löten entstehenden Dämpfe (auch vom Flussmittel) solltest du nicht direkt einatmen! Ein kleiner Ventilator, der den Dampf direkt von dir wegzieht, ist hier sehr nützlich und zu empfehlen. In Industriebetrieben gibt es für den Schutz der Mitarbeiter daher Lötrauchabsaugungen, die die Luft vom Arbeitsplatz gezielt wegführen.

2.3.3 Entlötlitze – wenn mal was danebengeht

Manchmal setzt man zu viel Lötzinn ein oder möchte ein Bauteil wieder von einer Platine entfernen. Hier kommt die Entlötlitze ins Spiel. Das ist ein feines Geflecht aus Kup-

ferdrähten, das mit Flussmittel getränkt sein kann. Du legst die Entlötlitze auf die zu entfernende Lötstelle und drückst mit dem heißen Lötkolben darauf. Das flüssige Lötzinn wird dann durch die Kapillarwirkung von der Entlötlitze aufgesaugt. So kannst du überschüssiges Lot entfernen oder Lötpunkte »freilegen«. Eine andere Möglichkeit zum Entlöten ist eine Entlötpumpe, aber für die typischen Arbeiten am Roboter-Auto ist eine Entlötlitze oft praktischer.

2.3.4 Lötspitzen – das richtige Werkzeug für jede Aufgabe

Die Lötspitze ist der Teil des Lötkolbens, der direkten Kontakt mit der Lötstelle hat und die Wärme überträgt. Lötspitzen sind Verschleißteile und können und müssen von Zeit zu Zeit gewechselt werden. Es gibt sie in vielen verschiedenen Formen und Größen, passend für unterschiedliche Lötaufgaben:

- Für größere Lötstellen (zum Beispiel dicke Kabel für die Spannungsversorgung der Motoren) eignet sich eine breitere, meißelförmige oder flache Lötspitze.
- Für feine Arbeiten an den Pins des ESP32-Mikrocontrollers, an Sensormodulen oder LEDs ist eine kleine meißelförmige Lötspitze ideal. Damit kannst du auch kleinste Lötpunkte präzise und sicher setzen.
- Je feiner die Lötspitze wird, umso mehr Übung brauchst du beim Löten, und so kleine, feine Bauteile werden wir im Roboter-Auto nicht miteinander verbinden.

Abbildung 2.1 Lötstation mit Zubehör

Halte deine Lötspitze immer sauber (abstreifen an einem feuchten Schwamm oder Messingwolle) und gut verzinnt, damit sie die Wärme optimal überträgt.

2.3.5 Praktische Helferlein: die »Dritte Hand«

Beim Löten hat man oft das Problem, dass man eigentlich drei Hände bräuchte: eine für den Lötkolben, eine für das Lötzinn und eine, um die zu verlötenden Bauteile in Position zu halten. Das ist die Aufgabe für die sogenannte *Dritte Hand*. Das ist ein kleiner Ständer mit beweglichen Armen und Krokodilklemmen, der die Bauteile für dich festhält.

Durch die vielfältigen Einstellmöglichkeiten der Arme und Klemmen kannst du Kabel, Platinen oder Bauteile exakt aufeinander ausrichten, sicher fixieren und anschließend sauber miteinander verlöten, ohne dass etwas verrutscht. Oft ist auch eine Lupe integriert, was bei kleinen Bauteilen sehr hilfreich ist.

Ein praktischer Tipp: Die Metallzähne der Krokodilklemmen können manchmal recht scharf sein. Um empfindliche Bauteile oder dünne Kabelisolierungen (vor allem, wenn diese beim Lötvorgang selbst heiß werden) nicht zu beschädigen, kannst du kurze Stücke Schrumpfschlauch über die Zähne der Klemmen ziehen.

Abbildung 2.2 Dritte Hand für Lötarbeiten

2.4 Vorbereitung ist alles

Bevor du mit dem Aufbau des Roboter-Autos beginnst, sorge dafür, dass du alle benötigten Werkzeuge zur Hand hast. Richte dir einen sauberen Arbeitsplatz ein, an dem du genügend Platz hast und der auch gut beleuchtet ist. Wenn du zum ersten Mal lötest, schau dir am besten ein paar Video-Tutorials an oder lass dir von jemandem helfen, der Erfahrung hat. Sorge für eine gute Belüftung, wenn du lötest, denn deine Gesundheit und Sicherheit gehen immer vor!

Mit der richtigen Ausrüstung bist du bestens vorbereitet, um die nächsten Schritte anzugehen und dein Roboter-Auto erfolgreich zusammenzubauen.

TEIL II

Prompte deine Roboter-Auto-Programme

Kapitel 14

Programmieren mit KI-Unterstützung: deine Prompt-Schule

Mache KI zu deinem Programmierpartner! Dieses Kapitel zeigt dir, wie du mit KI-Assistenten und den richtigen »Prompts« erstaunlich einfach Code für dein ESP32-Roboter-Auto erstellst. Entdecke, wie leicht Programmieren mit einem smarten KI-Assistenten sein kann!

In diesem Abschnitt möchte ich dir zeigen, wie du mithilfe von modernen KI-Sprachmodellen, sogenannten Large Language Models (LLMs) wie ChatGPT, Gemini und ähnlichen Diensten, dein Roboter-Auto bzw. deinen ESP32 programmieren kannst.

Kurz gesagt: vom Prompt zum Programm zum fertig programmierten Roboter-Auto.

Diese KI-Werkzeuge können dir als leistungsstarke Assistenten dienen, um Programme für dein Produkt, das Roboter-Auto, zu generieren, Fehler in deinem Programm zu finden oder weitere neue Ideen wie zusätzliche Funktionen zu entwickeln. Zum Zeitpunkt der Erstellung dieses Buches (Mitte 2025) sind viele der großen LLM-Anbieter in der Lage, direkt funktionierende Programme für den ESP32 zu generieren, wobei die Qualität der so erstellten Programme noch stark von deiner Eingabe, also deinem Prompt, abhängt. In den folgenden Abschnitten werde ich dir das Wissen und die Prinzipien beibringen, um gute Prompts zu schreiben.

14.1 Die Kunst des Befehls: Was ist ein Prompt und warum ist »Prompt Engineering« wichtig?

Wenn wir von großen Sprachmodellen, den sogenannten LLMs sprechen, stell sie dir als sehr fortgeschrittene Textgeneratoren vor, die Antworten basierend auf der Eingabe erstellen, die sie von dir erhalten haben. Diese Eingabe wird als *Prompt* bezeichnet – im Grunde ein Stück Text, das du dem Sprachmodell gibst, um seine Antwort zu lenken. Du musst kein Technologieexperte sein, um einen Prompt zu schreiben; das können wirklich alle!

Einen *guten* Prompt zu schreiben, kann jedoch etwas knifflig sein, und du musst etwas Erfahrung sammeln. Mehrere Faktoren beeinflussen, wie gut das Modell deinen Prompt versteht und darauf reagiert. Dazu gehören:

- **Das Modell selbst:** Verschiedene Sprachmodelle können auf denselben Prompt unterschiedlich reagieren.
- **Trainingsdaten:** Die Informationen, aus denen das Modell gelernt hat, werden seine Antworten beeinflussen.
- **Wie du deinen Prompt formulierst:** Die gewählten Worte, Fachbegriffe, die technische Detailtiefe, der Stil und die Struktur deines Prompts spielen alle in Summe eine wichtige Rolle.
- **Kontext:** Ausreichend Hintergrundinformationen helfen dem Modell, bessere Antworten auf deine Frage bzw. auf deine Aufgabenstellung zu geben.

Aufgrund dieser Faktoren erfordert das Erstellen effektiver Prompts oft etwas Ausprobieren und Anpassen – ein Ablauf sich zum Teil wiederholender Schritte, der als *Prompt Engineering* bekannt ist. Wenn dein Prompt nicht klar oder gut strukturiert ist, könnte das Modell vage oder falsche Antworten geben, was es dir erschwert, nützliche Informationen oder eben lauffähigen Programmcode von deinem KI-Assistenten zu erhalten. Ein klar formulierter Prompt bzw. eine Serie von Prompts ist daher oft der Schlüssel zum Erfolg.

14.2 Werkzeuge des Dialogs: wichtige Prompting-Techniken

KI-Assistenten und die dahinterliegenden Large Language Models, kurz LLMs, sind darauf trainiert, Anweisungen zu folgen, und wurden mit riesigen Datenmengen an Text trainiert, sodass sie einen Prompt verstehen und eine Antwort generieren können. Aber LLMs sind nicht perfekt; je klarer deine Prompt-Beschreibung der Aufgabe, desto besser kann das LLM den wahrscheinlichsten nächsten Text aus seinem antrainierten Wissen vorhersagen. Zusätzlich helfen dir spezifische Techniken, die die Funktionsweise von LLMs nutzen, dabei, relevante Ergebnisse zu erzielen. Daher werden wir uns jetzt zusammen in den kommenden Abschnitten verschiedene Prompt-Techniken näher anschauen.

Lerne direkt die englischen Begriffe

Ich verwende bewusst die englischen Begriffe mit deutscher Erläuterung, da sich diese Begriffe etabliert haben und du dazu im Internet noch weitere Informationen und Beispiele finden kannst.

14.2.1 Allgemeines Prompting / Zero-Shot-Prompting

Ein Zero-Shot-Prompt ist die einfachste Art von Prompt, und wir alle haben diese Art des Promptschreibens sicher bereits gemacht. Er enthält lediglich eine Beschreibung der Aufgabe und eventuell einen Anfangstext, mit dem das LLM starten kann. Diese Eingabe kann alles Mögliche sein: eine Frage, der Anfang einer Geschichte oder eine Anweisung. Der Name *Zero-Shot* bedeutet, dass es keine Beispiele gibt. Für einfache Code-Schnipsel oder Fragen kann das schon ausreichen.

14.2.2 One-Shot- und Few-Shot-Prompting (Prompting mit Beispielen)

Beim Erstellen von Prompts für KI-Modelle ist es oft hilfreich, dem Modell Beispiele zu geben. Diese Beispiele helfen dem LLM zu verstehen, was du als Ergebnis in etwa erwartest, besonders wenn du eine bestimmte Ausgabestruktur oder ein Muster wie einen Programmcode wünschst.

Ein One-Shot-Prompt liefert ein einzelnes Beispiel. Die Idee ist, dass das Modell ein Vorbild hat, das es imitieren kann, um die Aufgabe bestmöglich zu erledigen.

Ein Few-Shot-Prompt liefert dem Modell mehrere Beispiele. Dieser Ansatz zeigt dem Modell ein Muster, dem es folgen soll, was ideal ist für Programmieraufgaben. Mehrere Beispiele erhöhen die Chance, dass das Modell dem gewünschten Muster folgt.

Die Anzahl der Beispiele, die du für Few-Shot-Prompting benötigst, hängt von der Komplexität der Aufgabe, der Qualität der Beispiele und den Fähigkeiten des von dir genutzten KI-Modells bzw. KI-Services ab. Als Faustregel gelten mindestens drei bis fünf Beispiele. Für komplexere Aufgaben oder bei Längenbeschränkungen des Modelleingabefensters musst du eventuell mehr oder weniger Beispiele verwenden. Wähle Beispiele, die für deine Aufgabe relevant, vielfältig, qualitativ hochwertig und gut geschrieben sind. Schon ein kleiner Fehler im Beispiel kann das Modell verwirren und zu unerwünschten Ergebnissen führen. Daher ist auch bei der Verwendung von LLMs die Qualität der Daten bzw. deine Eingabe über den Prompt sehr wichtig, um gute Ergebnisse zurückzuerhalten.

14.2.3 System-, Kontext- und Rollen-Prompting

Wenn wir ein LLM bitten, etwas für uns zu tun – sei es, einen Text zu schreiben, eine E-Mail zu formulieren oder eben Code für unseren Mikrocontroller zu generieren –, müssen wir der KI genau sagen, was wir von ihr wollen. Das ist ein bisschen so, als würden wir einen sehr klugen, aber manchmal etwas unkonventionellen Assistenten anleiten.

Die Techniken System-Prompting, Kontext-Prompting und Rollen-Prompting sind im Grunde unsere Anleitungen für diesen Assistenten. Sie helfen uns, das gewaltige Wissen des LLM so zu lenken, dass es genau die Art von Ergebnis liefert, die wir brauchen. Jede dieser Techniken konzentriert sich auf einen anderen Aspekt der Anleitung:

System-Prompting
Stell dir das System-Prompting als die grundlegenden Regeln und das übergeordnete Ziel vor, die du deinem KI-Assistenten gibst, bevor er überhaupt mit der Arbeit beginnt. Es definiert das »große Ganze« dessen, was das Sprachmodell tun soll. Es ist die Anweisung, die dem Modell seine Hauptaufgabe zuteilt und seinen grundlegenden Charakter bestimmt. Der System-Prompt wird häufig in den Einstellungen des KI-Assistenten hinterlegt, wenn dein Anbieter solch eine Funktion bietet, und ab dann immer angewandt.

- **Beispiel für ein Roboter-Projekt:** Wenn du ein LLM für dein Roboter-Projekt einsetzen möchtest, könnte ein System-Prompt lauten: »Du bist ein Experte für die Programmierung von Arduino-basierten Robotern und hilfst mir beim Schreiben von Code-Beispielen in C++.«
- **Warum ist das wichtig?** Ohne einen klaren System-Prompt könnte das Modell zum Beispiel versuchen, ein Kochrezept zu schreiben, obwohl du Code brauchst, oder es antwortet in einer sehr umgangssprachlichen Weise, obwohl du einen professionellen Stil erwartest. Der System-Prompt stellt sicher, dass die KI von Anfang an auf dem richtigen Weg ist.

Kontext-Prompting
Während der System-Prompt die allgemeinen Regeln festlegt, die ab dann immer gelten, liefert das Kontext-Prompting die spezifischen Details oder Hintergrundinformationen, die für die aktuelle, konkrete Aufgabe relevant sind. Es ist, als würdest du deinem Assistenten einen Bauplan oder eine Skizze für das aktuelle Projekt geben.

- **Beispiel für Code-Generierung:** Du hast das LLM mit dem System-Prompt »Experte für Arduino-Roboter-Code« ausgerüstet. Jetzt möchtest du spezifischen Code. Ein Kontext-Prompt könnte sein: »Ich brauche Arduino-Code, um einen Servomotor zu steuern. Der Servomotor soll zwischen 0 und 180 Grad hin- und herfahren. Bitte verwende die Standard-Servo-Bibliothek.«
- **Weiteres Beispiel mit deinem Roboter-Auto:** Wenn du sagst: »Der linke Joystick des PS5-Controllers soll das Roboter-Auto steuern«, dann gibst du dem Modell Kontext, welche Komponente (linker Joystick) für welche Funktion (Steuerung des Autos) zuständig ist.

- **Warum ist das wichtig?** Der Kontext-Prompt hilft dem Modell, die Nuancen deiner Anfrage zu verstehen und die Antwort genau anzupassen. Er verhindert, dass die KI allgemeine oder falsche Annahmen trifft, indem du ihr alle nötigen Informationen für die aktuelle Aufgabe direkt mitgibst.

Mein persönlicher Favorit: Rollen-Prompting
Mit dem Rollen-Prompting weist du dem Sprachmodell eine klare Rolle, einen bestimmten Charakter oder eine Identität zu, die es annehmen soll. Dadurch weiß die KI genau, was von ihr erwartet wird, und kann ihre Antworten und Aktionen entsprechend anpassen. Für unsere Aufgabe der Programmierung eines ESP32-Mikrocontrollers könnte die Persönlichkeit der KI zum Beispiel wie folgt beschrieben werden: »Du bist ein erfahrener C++-Programmierer für Mikrocontroller in der Entwicklungsumgebung Arduino.« Dies hilft dem KI-Modell, Antworten zu generieren, die mit der zugewiesenen Rolle und dem damit verbundenen Wissen und Verhalten übereinstimmen.

Jetzt haben wir den System- und Kontext-Prompt kennengelernt. Mit dem dritten Prompt-Typ, dem Rollen-Prompt, kann es erhebliche Überschneidungen zwischen diesen drei Arten der Prompts geben. Ein Prompt, der dem System eine Rolle zuweist, kann auch Kontext enthalten. Jede Art dient einem etwas anderen Hauptzweck:

- **System-Prompt:** Definiert die grundlegenden Fähigkeiten und den übergeordneten Zweck des Modells.
- **Kontext-Prompt:** Liefert unmittelbare, aufgabenspezifische Informationen, um die Antwort zu lenken; ist hochspezifisch für die aktuelle, dynamische Aufgabe.
- **Rollen-Prompt:** Gestaltet den Ausgabestil und die »Stimme« des Modells; fügt eine Ebene der Spezifität und Persönlichkeit hinzu.

Diese Unterscheidung bietet einen Rahmen für das Design von Prompts mit klarer Absicht und erleichtert die Analyse, wie jeder Prompt-Typ die Ausgabe des Sprachmodells beeinflusst.

14.2.4 Rollen-Prompting genauer betrachtet

Sehr hilfreich bei der partnerschaftlichen Programmierung mit einem LLM ist Rollen-Prompting. Daher möchte ich darauf etwas genauer eingehen, da ich mit dieser Art der Aufgabenbeschreibung bei der Programmierung sehr gute Erfahrungen gemacht habe.

Rollen-Prompting ist eine Technik, bei der du dem KI-Modell eine bestimmte Rolle zuweist. Das kann helfen, relevantere und informativere Ausgaben zu generieren, da das Modell seine Antworten auf die ihm zugewiesene Rolle zuschneiden kann. Du könntest einem KI-Modell beispielsweise die Rolle eines Bäckers, eines Kindergärtners oder einer

erfahrenen Embedded-Entwicklerin mit Expertenwissen in der Programmiersprache C++ geben. Sobald das KI-Modell eine spezifische Rolle zugewiesen bekommen hat, kannst du ihm rollenspezifische Aufgaben geben.

```
Du bist ein KI-Assistent und Experte für die Programmierung von ESP32-Mikro-
controllern mit der Arduino IDE und der Programmiersprache C++.
Schreibe ein C++-Programm für einen ESP32-Mikrocontroller, das ein Modell-Robo-
ter-Auto steuert. Das Roboter-Auto verwendet eine L298N-H-Brücke als Motortrei-
ber zur Ansteuerung der Gleichstrommotoren im Roboter-Auto. Der ESP32 steuert
den L298N direkt. Das Programm soll die Bluepad32-Bibliothek (https://git-
hub.com/ricardoquesada/bluepad32) nutzen, um einen PS5-Controller mit dem ESP32
zu verbinden. Der PS5-Controller dient als Lenkung für das Roboter-Auto.
Hardware-Pin-Belegung ESP32 zu L298N:
- Motortreiber Eingang IN1 an ESP32 GPIO 13
- Motortreiber Eingang IN2 an ESP32 GPIO 12
- Motortreiber Eingang IN3 an ESP32 GPIO 27
- Motortreiber Eingang IN4 an ESP32 GPIO 33
- Motortreiber Eingang ENA und ENB sind via Jumper auf HIGH gesetzt.
Das Programm soll folgende Funktionalitäten beinhalten:
1. Initialisierung: ESP32 einrichten, Pins für die L298N-H-Brücke (IN1-IN4
als digitale Ausgänge) konfigurieren und die Bluepad32-Bibliothek für die
PS5-Controller-Verbindung initialisieren.
2. Controller-Eingaben verarbeiten: Eingaben vom linken Joystick des PS5-Con-
trollers lesen, um Geschwindigkeit und Richtung zu steuern. Der rechte Joy-
stick wird nicht verwendet.
3. Motorsteuerfunktionen: Implementiere Funktionen (z. B. setMotorSpeed(motor,
speed, direction)), um das Roboter-Auto basierend auf den Joystick-Eingaben zu
bewegen. Geschwindigkeit soll per PWM über die IN1-IN4 gesteuert werden. Len-
kung soll wie bei einem Panzer erfolgen (Tank Steering).
4. Hauptschleife (loop): Kontinuierlich Controller-Eingaben prüfen und die
Motorzustände entsprechend aktualisieren.
Bitte strukturiere den Code mit klaren Kommentaren und organisiere ihn in Funk-
tionen für Initialisierung, Eingabeverarbeitung und Motorsteuerung.
```

Listing 14.1 Beispiel für einen Rollen-Prompt zur Code-Generierung für dein Roboter-Auto

Im Grunde ist es wie mit deinem Roboter-Auto: Wenn du dem Motor ganz klar sagst, was er tun soll (zum Beispiel »fahre vorwärts!«), dann macht er das auch richtig gut. Genauso ist es mit der KI: Gib ihr eine klare Aufgabe, und sage ihr ganz genau, auf welchem Gebiet sie ein spezielles Wissen und Können (ihre »Expertise«) haben soll, und als Ergebnis werden die Code-Vorschläge viel passender und brauchbarer für dein Projekt!

14.2.5 Chain-of-Thought-Prompting (CoT)

Denken in Abschnitten: Wenn dein Programm bereits auf dem ESP32 läuft und du auf unerklärliches Verhalten stößt, kann Chain-of-Thought-Prompting (CoT) eine gute Methode sein, um das LLM zur Problemlösung anzuleiten. CoT-Prompting ist eine Technik zur Verbesserung der Denkfähigkeiten von LLMs, indem man sie dazu anleitet, Zwischenschritte ihrer Überlegungen zu generieren. Dies hilft dem LLM, genauere Antworten zu finden. Du kannst es mit Few-Shot-Prompting kombinieren, um bessere Ergebnisse bei komplexeren Aufgaben zu erzielen, die logisches Denken erfordern, bevor eine Antwort gegeben wird – eine Herausforderung bei einem reinen Zero-Shot-CoT.

CoT hat viele Vorteile:

Es ist aufwandsarm, sehr effektiv und funktioniert gut mit handelsüblichen frei zugänglichen LLMs (kein Feintuning nötig). Du erhältst auch Interpretierbarkeit, da du die Denkschritte des LLMs nachvollziehen kannst. Wenn etwas fehlschlägt, kannst du den Fehler möglicherweise identifizieren. CoT scheint auch die Robustheit bei wechselnden LLM-Versionen zu verbessern. Natürlich gibt es auch Nachteile: Die Antwort des LLMs enthält die Denkketten, was mehr Ausgabetoken bedeutet und somit die Vorhersagen teurer und zeitaufwendiger macht.

```
Du bist ein Experte für Roboter, die in C++ für den ESP32 programmiert werden.
Mein Roboter-Auto (L298N direkt an ESP32 GPIOs 13(IN1), 12(IN2), 27(IN3),
33(IN4), ENA und ENB via Jumper auf HIGH gesetzt) dreht nach rechts, wenn ich
nach links lenke, und umgekehrt. Deine Aufgabe ist es, dieses Lenkproblem im
C++-Code zu erklären und zu beheben. Denke bitte in Abschnitten laut nach, be-
vor du die Lösung präsentierst und den korrigierten Codeabschnitt zeigst.
```

Listing 14.2 Beispiel für einen CoT-Prompt zur Fehlerbehebung

14.2.6 Der Aufbau macht den Unterschied: Struktur in Prompts

Wir haben bereits gesehen, wie detailliert ein Prompt sein kann, besonders unser Beispiel im Rollen-Prompting. Ein langer, unstrukturierter Textblock kann eine KI jedoch leicht überfordern oder zu Missverständnissen führen, selbst wenn alle Informationen darin enthalten sind. Eine klare Struktur hilft nicht nur der KI, die Anweisungen präzise zu befolgen, sondern auch dir selbst, den Überblick zu behalten und sicherzustellen, dass du keine wichtigen Details vergisst. Wir dürfen auch nie vergessen, dass KI-Modelle mit unglaublichen Mengen an von Menschen erzeugtem Text trainiert wurden und so die KI-Modelle auch gelernt haben, dass Texte eine gewisse Struktur haben.

Anstatt alle Anweisungen in einem einzigen Fließtext zu schreiben, können wir den Prompt durch logische Blöcke, Einrückungen und Aufzählungszeichen (* oder -) gliedern. Die Verwendung von * und Einrückungen ahmt nach, wie professionelle KI-Systeme im Hintergrund arbeiten, die auch sehr häufig Strukturen der Beschreibungssprache XML oder JSON nutzen. Aber die Verwendung von * und Einrückungen macht es für uns Menschen auch viel einfacher lesbar.

Schauen wir uns unseren ausführlichen Rollen-Prompt aus dem vorherigen Kapitel noch einmal an, diesmal aber in einer klar strukturierten Form.

```
Du bist ein KI-Assistent und Experte für die Programmierung von ESP32-Mikro-
controllern mit der Arduino IDE und der Programmiersprache C++.
** Hauptaufgabe **
  * Schreibe ein C++-Programm zur Steuerung eines Roboter-Autos mit einem
    PS5-Controller.
** Hardware-Spezifikationen **
  * Mikrocontroller: ESP32-Mikrocontroller
  * Motortreiber: L298N-H-Brücke
    * Motortreiber Eingang IN1 an ESP32 GPIO 13
    * Motortreiber Eingang IN2 an ESP32 GPIO 12
    * Motortreiber Eingang IN3 an ESP32 GPIO 27
    * Motortreiber Eingang IN4 an ESP32 GPIO 33
    * Motortreiber Eingänge ENA und ENB sind via Jumper auf HIGH gesetzt.
  * Steuerung: PS5-Controller
    * Verbindung über die Bluepad32-Bibliothek
      (https://github.com/ricardoquesada/bluepad32).
** Angeforderte Programm-Funktionalitäten **
  * 1. Initialisierung:
    * Richte den ESP32 ein.
    * Konfiguriere die Pins für die L298N-H-Brücke (IN1-IN4) als digitale
      Ausgänge.
    * Initialisiere die Bluepad32-Bibliothek für die Verbindung mit dem PS5-
      Controller.
  * 2. Controller-Eingaben verarbeiten:
    * Lies die Eingaben vom linken Joystick des PS5-Controllers.
    * Der rechte Joystick wird nicht verwendet.
  * 3. Motorsteuerfunktionen:
    * Implementiere Funktionen, um das Auto zu bewegen (zum Beispiel
      setMotorSpeed(motor, speed, direction)).
    * Steuere die Geschwindigkeit per PWM direkt über die Pins IN1 bis IN4.
    * Die Lenkung soll wie bei einem Panzer erfolgen (Tank Steering).
```

```
  * 4. Hauptschleife (loop):
    * Prüfe kontinuierlich die Controller-Eingaben.
    * Aktualisiere die Motorzustände entsprechend.
** Anforderungen an den Code **
  * Strukturiere den Code mit klaren Kommentaren.
  * Organisiere den Code in logische Funktionen (Initialisierung,
    Eingabeverarbeitung, Motorsteuerung).
```

Listing 14.3 Beispiel für einen gut strukturierten Prompt

Du siehst sofort den Unterschied. Die Hauptaufgabe ist klar definiert, die Hardware-Spezifikationen sind sauber gruppiert, und die einzelnen Software-Anforderungen sind als übersichtliche Liste aufgeführt. Diese Gliederung zwingt die KI, jeden Punkt systematisch abzuarbeiten, was die Wahrscheinlichkeit für ein qualitativ hochwertiges und korrektes Ergebnis drastisch erhöht.

14.3 Dein perfekter Befehl: einen effektiven Prompt für das ESP32-Roboter-Auto erstellen

Jetzt hast du die Grundlagen kennengelernt und von mir sehr ausführliche Beispiel-Prompts gezeigt bekommen. Du ahnst es sicher schon. Ein gut formulierter Prompt ist der Schlüssel zum Erfolg. Für unser Roboter-Auto sollte dein Prompt eine Kombination aus Kontext- und Rollen-Prompting darstellen und idealerweise folgende Informationen enthalten:

- **Rolle für die KI:** Du bist ein erfahrener Programmierer für ESP32-Mikrocontroller und Robotik-Anwendungen.
- **Verwendete Entwicklungsumgebung (IDE):** Erstelle den Code für die Arduino IDE.
- **Zielhardware:** Das Zielsystem ist ein ESP32 Dev Module.
- **Wichtige angeschlossene Komponenten und deren Verbindung:** Ein L298N-Motortreiber wird direkt vom ESP32 gesteuert.
- **Genaue Pin-Belegung:** Dies ist absolut entscheidend!
  - L298N IN1 an ESP32 GPIO 13
  - L298N IN2 an ESP32 GPIO 12
  - L298N IN3 an ESP32 GPIO 27
  - L298N IN4 an ESP32 GPIO 33
  - L298N ENA und ENB sind via Jumper auf HIGH gesetzt

- **Gewünschte Steuerungslogik:** Eine Lenkung wie bei einer Kettenraupe mit dem linken Joystick eines PS5-Controllers, der via Bluepad32 verbunden ist. Y-Achse für Vor/Zurück (PWM-Geschwindigkeit), X-Achse für Drehung auf der Stelle.
- **Bekannte oder bevorzugte Bibliotheken:** Bluepad32-Bibliothek für PS5-Controller-Anbindung (hier den Link zu Doku/Beispielen einfügen).
- **Beispielcode (Few-Shot/One-Shot):** Füge relevante Code-Ausschnitte aus funktionierenden Beispielen bei (zum Beispiel aus der Bluepad32-Bibliothek, wie die Controller-Daten ausgelesen werden).

Dein fertiger Prompt kann dann zum Beispiel so aussehen:

```
Du bist ein KI-Assistent und Experte für die Programmierung von ESP32-Mikrocontrollern mit der Arduino IDE und der Programmiersprache C++.
** Hauptaufgabe **
* Erstelle ein vollständiges und lauffähiges C++-Programm für die Arduino IDE, um ein Roboter-Auto zu steuern.
** Hardware-Setup **
* Zielsystem: ESP32 Dev Module
* Motorsteuerung: L298N-H-Brücken-Motortreiber
* Controller: PS5-Controller (Verbindung via Bluetooth)
** Pin-Belegung (ESP32 -> L298N) **
* Linker Motor:
  * GPIO 13 -> IN1 (Richtung)
  * GPIO 12 -> IN2 (Richtung)
* Rechter Motor:
  * GPIO 27 -> IN3 (Richtung)
  * GPIO 33 -> IN4 (Richtung)
** Bibliotheken & APIs **
* Benötigte Bibliothek: Bluepad32 (https://github.com/ricardoquesada/bluepad32)
                        für die PS5-Controller-Anbindung.
* Bevorzugte API: Nutze die ESP32 LEDC API für eine präzise PWM-Steuerung der
                  Motorgeschwindigkeit.
** Steuerungslogik (Panzerlenkung) **
* Eingabegerät: Ausschließlich der linke Joystick des PS5-Controllers.
* Deadzone: Implementiere eine Deadzone von ca. 15% für die X- und Y-Achse des
  Joysticks, um unbeabsichtigte Bewegungen in der Mittelstellung zu verhindern.
* Vorwärts-/Rückwärtsfahrt (Y-Achse):
  * Steuert die Grundgeschwindigkeit beider Motoren.
  * Die Geschwindigkeit soll proportional zur vertikalen Auslenkung sein.
```

```
* Drehung auf der Stelle (X-Achse):
  * Nach links: Linker Motor dreht rückwärts, rechter Motor dreht vorwärts.
  * Nach rechts: Linker Motor dreht vorwärts, rechter Motor dreht rückwärts.
  * Die Drehgeschwindigkeit soll proportional zur horizontalen Auslenkung sein.
* Kurvenfahrten (kombinierte X- und Y-Achse):
  * Mische die Eingaben der X- und Y-Achse, um flüssige Kurvenfahrten zu
    ermöglichen.
  * Passe die Geschwindigkeiten des linken und rechten Motors entsprechend an
    (z. B. bei einer Vorwärtskurve nach links dreht der rechte Motor schneller
    als der linke).
** Anforderungen an den Code **
* Struktur:
  * Erstelle einen gut strukturierten und umfassend kommentierten Code.
  * Verwende const int Variablen für alle Pin-Definitionen.
  * Richte die PWM-Kanäle (LEDC) in einer separaten Setup-Funktion ein.
* Funktionen:
  * Implementiere separate, wiederverwendbare Funktionen für die
    Motoransteuerung.
    * Beispiel: void setLeftMotorSpeed(int speed) und
                void setRightMotorSpeed(int speed).
    * Die speed-Werte sollen von -255 (volle Geschwindigkeit rückwärts) bis
      +255 volle Geschwindigkeit vorwärts) reichen und innerhalb der
      Funktionen in PWM-Werte und Richtungslogik umgesetzt werden.
  * Die Hauptschleife (loop()) soll sauber bleiben und hauptsächlich Funktionen
    zur Abfrage des Controllers und zur Ansteuerung der Motoren aufrufen.
```

Listing 14.4 Beispiel-Prompt für das ESP32-Roboter-Auto, angepasst an die direkte L298N-Steuerung

Das war ein sehr ausführlicher Prompt. Ich habe die Erfahrung gemacht, dass LLMs mit Aufgaben und Informationen besser arbeiten können, wenn sie klar strukturiert sind. Je mehr Mühe du dir bei der Formulierung gibst, desto besser werden die Ergebnis sein. Ich speichere daher Prompts, die ein gutes Ergebnis geliefert haben, in einem Dokument und passe sie für neue Aufgabenstellungen an. Dafür nehme ich wieder einen KI-Assistenten, den ich auffordere, anhand dieser Vorlage einen neuen Prompt zu generieren. So bin ich um ein Vielfaches schneller beim Schreiben meiner Prompts, als wenn ich immer wieder bei null anfangen und alles selbst erstellen würde. Genau dieses Vorgehen werden wir dann auch ab Kapitel 16 für unsere Prompts anwenden.

14.4 Der iterative Prozess: vom ersten Entwurf zum fertigen Code

Selbst mit dem besten Prompt ist es nicht sicher, dass ein KI-Service auf Anhieb perfekten, fehlerfreien Code für ein komplexes Entwicklungsprojekt liefert. Die Wahrscheinlichkeit, dass eine komplexe Entwicklungsaufgabe auf Anhieb richtig gelöst wird, liegt 2025 noch bei ca. 78 % (Quelle: The 2025 AI Index Report der Stanford University). Stell dir einen KI-Assistenten als einen extrem schnellen Programmierpartner vor, dessen Arbeit du aber immer überprüfen und mit den richtigen Fragen anleiten und vor allem auch hinterfragen musst.

Hier ist ein typisches Vorgehen:

1. **Prompt senden und ersten Entwurf erhalten:** Schicke deinen detaillierten Prompt an den KI-Chat-Service deiner Wahl.
2. **Prüfen, kompilieren und erste Fehlerbehebung:**
   - Leseverständnis und Logikprüfung: Lies den Code sorgfältig. Entspricht er deinen Anweisungen? Sind die Pins korrekt verwendet? Ist die Logik für die Panzerlenkung nachvollziehbar?
   - Kompilierfehler: Kopiere den Code in die Arduino IDE. Behebe offensichtliche Fehler, oder gib die Fehlermeldungen präzise an die KI zurück, mit der Bitte um Korrektur.
3. **Testen am Roboter und Logikfehler finden:**
   - Lade den Code auf deinen ESP32.
   - Teste die Funktionalität: Reagiert der Roboter korrekt auf den Controller? Drehen die Motoren richtig? Funktioniert die Geschwindigkeitssteuerung?
4. **Feedback an die KI:** Beschreibe beobachtetes Fehlverhalten detailliert, zum Beispiel: `Wenn der linke Joystick leicht nach vorne bewegt wird, fahren die Motoren sofort mit voller Geschwindigkeit. Die PWM-Steuerung scheint nicht proportional zur Joystick-Auslenkung zu sein`, und bitte um Korrektur.
5. **Verfeinern und wiederholen:** Dieser iterative Prozess aus Testen und Feedback ist normal, wird aber häufig immer schneller, da nur noch kleine Anpassungen vorgenommen werden müssen.
6. **Weitere Funktionen hinzufügen:** Sobald die Basissteuerung funktioniert, kannst du die KI bitten, das Programm um zusätzliche Funktionen zu erweitern, wie zum Beispiel die Integration einer Anzeige auf einem OLED-Display für die Geschwindigkeit/Joystick-Werte, Lichteffekte etc.

14.5 Dokumentation deiner Prompts – behalte den Überblick

Besonders bei komplexeren Aufgabenstellungen ist es hilfreich, deine Prompts und die Antworten der KI zu dokumentieren (zum Beispiel in einer Textdatei oder Tabelle). Notiere dir den Prompt, das Ergebnis, beurteile das Ergebnis und deinen Folge-Prompt und so weiter. Das hilft dir, den Überblick zu wahren und deine Prompting-Fähigkeiten zu verbessern.

Ich habe mir so eine persönliche Prompt-Sammlung erstellt. Diese ist sortiert nach Themen von der Code-Generierung bis zur Bild-Generierung. Zu jedem Prompt habe ich zusätzlich notiert, ob ich einen lokalen Service verwendet habe, der bei mir daheim läuft, mit seinem Namen und der Version.

Bei Online-Services habe ich mir ebenfalls den verwendeten Service und dessen Version bzw. LLM-Modell notiert. Denn es ist aktuell so, dass alle paar Wochen ein neuer Service verfügbar wird oder ein etablierter nicht mehr so gut ist wie ein KI-Programmierservice von einem Konkurrenten.

Das ist aufwendig, aber mit der Zeit nimmt die Dokumentation ab, du hast das Prompten gelernt und greifst auf ein paar wenige gute initiale Prompts aus deiner Sammlung zurück. Du wirst ab selbst sehen, dass die Prompts alle immer gleich starten, also das gleiche Grundgerüst haben, und die Aufgabenstellung dann natürlich abweicht. Wobei die Beschreibung der Aufgabenstellung dann auch wiederum immer in jedem Prompt sehr ähnlich ist.

14.6 Realistische Erwartungen und deine Rolle als Entwickler

Die Fähigkeit von LLMs, Code zu generieren, ist ohne Frage beeindruckend und wird rasant besser. Dennoch gilt im Jahr 2025:

- **KI ist ein Werkzeug:** Sie ersetzt nicht dein Verständnis. Du bleibst der Entwickler oder die Entwicklerin und musst den Code prüfen, verstehen und anpassen.
- **Verantwortung:** Du bist für die Sicherheit und korrekte Funktion deines Codes und des Roboters verantwortlich.
- **Präzise Fragen stellen:** Die Qualität der Antwort hängt von deinem Prompt ab. Denke auch an Sicherheitsaspekte wie zum Beispiel: »Was passiert, wenn die Controller-Verbindung abbricht?« Nur wenn du die richtigen Fragen stellen kannst, also ein Verständnis hast von dem, was für die gestellte Aufgabe wichtig ist, kann die KI »guten« Code generieren.

Sieh die KI als einen sehr fortschrittlichen Programmierassistenten an, der aber kein tieferes Verständnis hat, was du erreichen möchtest oder wie genau das Roboter-Auto aussieht und wofür es genutzt wird. Du wirst aber mit der Zeit und vor allem Übung immer besser darin werden, dem KI-Service – ob für die Programmierung oder Bildgenerierung – genau die richtigen Anweisungen zu geben, sodass dir der Service die Antwort generiert, die zu deiner Aufgabenstellung am besten passt.

Du hast dich bis hierher durch viel Theorie gearbeitet, und wir werden jetzt das erlernte Wissen nutzen, um Programme schreiben zu lassen, die es deinem Roboter-Auto ermöglichen, Sensoren zu nutzen. Was wird wohl alles möglich, wenn dein Roboter seine Umgebung noch besser verstehen könnte? Stell dir vor, er könnte Hindernissen von ganz alleine ausweichen, einer geklebten Linie auf dem Boden folgen oder sogar erkennen, wie weit ein Gegenstand entfernt ist. Genau das wird möglich, wenn wir unserem Roboter-Auto »Sinne« verleihen – in Form von Sensoren! Im nächsten Kapitel lassen wir von einem KI-Assistenten deiner Wahl Programme schreiben, die es deinem Roboter-Auto ermöglichen, seine Umwelt wahrzunehmen und intelligent darauf zu reagieren. Sei gespannt!

Inhalt

Materialien zum Buch 18
Vorwort 19

Teil I Baue dein eigenes Roboter-Auto

1 Dein Roboter-Auto zum Selberbau 25

1.1 Warum überhaupt ein Roboter-Auto bauen? 25
1.2 Für wen ist dieses Projekt geeignet? 27
1.3 Mehr als nur Likes: dein Projekt, deine Zukunft 27

2 Die richtige Ausrüstung: Werkzeuge für das Roboter-Auto-Projekt 29

2.1 Warum die richtigen Werkzeuge wichtig sind 29
2.2 Die wichtigsten Werkzeuge für dein Roboter-Auto-Projekt 30
2.3 Gut verlötet ist halb gewonnen – Grundlagen des Lötens 31
2.3.1 Der Lötprozess – eine Anleitung zur perfekten Lötstelle 32
2.3.2 Wichtiges Lötzubehör – was du noch brauchst 32
2.3.3 Entlötlitze – wenn mal was danebengeht 33
2.3.4 Lötspitzen – das richtige Werkzeug für jede Aufgabe 34
2.3.5 Praktische Helferlein: die »Dritte Hand« 35
2.4 Vorbereitung ist alles 36

3 Das Chassis: die Basis deines Roboter-Autos 37

3.1 Präzision aus dem Drucker: das 3D-gedruckte Chassis 38
3.1.1 Was du benötigst 39
3.1.2 Der Fertigungsprozess: vom Entwurf zum gedruckten Chassis 40
3.1.3 Vorbereitung der Montage der Komponenten im Chassis-Design 42
3.1.4 Zusammenfassung 42
3.2 Flexibel und kreativ: das Chassis aus LEGO®-Bausteinen 43
3.2.1 Was du benötigst 43

3.2.2 Aufbau der Getriebemotorhalterung ... 44
3.2.3 Fahrgestell montieren ... 46
3.2.4 Zusammenfassung ... 46
3.3 Günstig und individuell: das Chassis aus Pappe ... 47
3.3.1 Was du benötigst ... 47
3.3.2 Modellzeichnung und Zuschnitt ... 48
3.3.3 Zusammenfassung ... 51
3.4 Deine Wahl, dein Roboter! ... 52

4 3D-Druck-Chassis: So erhältst du das Chassis und andere Teile 53

4.1 3D-Druck verstehen: Schicht für Schicht zum Objekt ... 53
4.2 Die Baupläne: Was ist eine STL-Datei? ... 54
4.3 Dein Druckmaterial: Filament-Arten für dein Roboter-Auto ... 54
4.3.1 Polylactide: der Einsteiger-Star ... 55
4.3.2 Polyethylenterephthalatglycol: der zähe Allrounder ... 55
4.4 Kein 3D-Drucker zuhause? Kein Problem! ... 56
4.5 Dein erster 3D-Druck: wichtige Einstellungen und Tipps ... 56

5 Strom ist die Roboter-Energie – Grundlagen der Elektrizität für dein Roboter-Auto 59

5.1 Elektrische Gesetze und Formeln – das kleine Einmaleins der Elektronik ... 59
5.1.1 Die Reihenschaltung – alles hintereinander ... 60
5.1.2 Die Parallelschaltung – alles nebeneinander ... 61
5.1.3 Das Ohm'sche Gesetz – der Dreiklang der Elektrotechnik ... 63
5.2 Die elektrische Leistung – wie viel Power steckt drin? ... 63
5.2.1 Beispielrechnungen für dein Roboter-Auto ... 63
5.2.2 Beispiel: Reihenschaltung von zwei Motoren ... 64
5.2.3 Beispiel: Parallelschaltung von zwei Motoren ... 65
5.3 Leistungsaufnahme des gesamten Roboter-Autos in einem Beispiel ... 65
5.4 Beispielrechnung für einen LED-Vorwiderstand ... 66

6 Datenübertragung im Detail: der I2C-Bus und der ESP32 69

6.1 Was ist der I2C-Bus? ... 69
6.2 Wie funktioniert I2C? ... 71
6.3 Der I2C-Bus des ESP32 ... 73
6.4 I2C-Hubs: mehr Anschlüsse für Sensoren ... 73
6.5 DIY-I2C-Hub – meine persönliche Empfehlung ... 74
6.6 Die genaue Verwendung des I2C-Busses im Roboter-Auto ... 75
6.7 Level-Shifter (Pegelwandler) – die Spannungsanpassung im I2C-System ... 76

7 Diese Hardware brauchst du für dein Roboter-Auto 77

7.1 Das Gehirn: der ESP32-Mikrocontroller ... 77
7.2 Fortbewegung: die TT-Getriebemotoren mit Rädern ... 78
7.3 Motorsteuerung: L298N-Motortreiber-Modul ... 79
7.4 OLED-Display SSD1306 – das Info-Display ... 81
7.5 Die 18650-Li-Ion-Akkus ... 82
7.5.1 Upcycling: alten Akkus neues Leben einhauchen (nichts für Kinder) ... 83
7.5.2 Wie du 18650-Akkus testest und ihren Zustand prüfst ... 85
7.6 Batteriehalter mit Feinsicherung und Sicherungshalter ... 86
7.6.1 Was ist eine Sicherung? ... 87
7.6.2 Schutz vor Stromspitzen und Kurzschlüssen ... 87
7.7 Die richtige Spannung: der Spannungswandler ... 88
7.8 Beleuchtung: NeoPixel-LEDs ... 89
7.9 Fernsteuerung: Sony PS4/PS5 Gamecontroller ... 91
7.10 Spannung im Blick: die digitale Spannungsanzeige ... 92
7.11 Kleinmaterial: Kabel, Verbinder und mehr ... 94
7.12 Übersicht der benötigten Komponenten ... 96
7.13 Was kommt als Nächstes? ... 98

8 Die Spannungsversorgung des Roboter-Autos 99

8.1 Warum drei 18650-Akkus als Spannungsquelle? 99

8.2 Benötigte Komponenten für die Spannungsversorgung 100

8.3 Aufbau der Spannungsversorgung 100

8.3.1 Wichtige Sicherheitshinweise 101

8.3.2 Erläuterung zur Schalterintegration 102

8.4 Was kommt als Nächstes? 106

9 Das 3D-Druck-Chassis – dein Roboter nimmt Gestalt an 107

9.1 Aufbau des Beispiel-Chassis 109

9.2 Vorbereitung der 3D-Druckteile 110

9.3 Dein 3D-Druck-Roboter-Chassis: die Bauteile im Detail 110

9.3.1 Die Bodenplatte 111

9.3.2 Die Motortreiberhalterung 112

9.3.3 Der Rahmen 112

9.3.4 Der Stoßfänger 113

9.3.5 Der Elektronikträger 114

9.3.6 Der Deckel mit Anbauteilen 115

9.3.7 Das Fahrerhaus mit Dach 116

9.3.8 Die Universalhalterung (optional) 117

9.3.9 Der Nadel- und Luftballonhalter (optional) 117

9.3.10 Der Schiebeschild mit Bolzen (optional) 118

9.4 Was kommt als Nächstes? 120

10 Schritt-für-Schritt-Montage des 3D-gedruckten Roboter-Autos 121

10.1 Fokus auf das 3D-Druck-Beispiel 121

10.2 Start der Roboter-Auto-Verkabelung und -Montage 122

10.3 ESP32 – logische Verkabelung vorbereiten 124

10.4 LED-Daten-Pin vorbereiten 126

10.5 Anschlusskabel der Motoren vorbereiten und anschließen 127

10.6 TT-Getriebemotoren montieren und anschließen ... 128
10.6.1 Motoren in Reihe schalten für reduzierte Geschwindigkeit und erhöhte Lebensdauer ... 129
10.6.2 Motoren parallel schalten für eine hohe Endgeschwindigkeit ... 131
10.6.3 Hinweis zur Polung der Motoren ... 132
10.6.4 Motortreiber einsetzen ... 132
10.7 Spannungswandler befestigen ... 133
10.8 Anschluss des Motortreibers an den ESP32 ... 134
10.9 Verkabelung des Fahrerhauses ... 136
10.9.1 OLED-Display anschließen ... 136
10.9.2 Spannungsanzeige einbauen und anschließen ... 140
10.9.3 Ein-/Ausschalter einbauen und anschließen ... 142
10.10 LED-Verkabelung vorbereiten und einbauen ... 144
10.11 Erster Test der Spannungsversorgung ... 147
10.12 Was kommt als Nächstes? ... 148

11 Erste Schritte mit der Arduino-Entwicklungsumgebung 149

11.1 Arduino IDE herunterladen und installieren ... 149
11.2 ESP32-Boards für Arduino IDE einrichten ... 150
11.3 Benötigte Bibliotheken installieren ... 152
11.4 ESP32 mit dem Computer verbinden ... 152
11.5 Test-Sketch hochladen ... 152
11.5.1 Fehlerbehebung (Troubleshooting) ... 153

12 Die komplette Software für dein Roboter-Auto 155

12.1 Voraussetzungen prüfen ... 155
12.2 Roboter-Auto-Programm herunterladen und aufspielen ... 156
12.3 ESP32-Roboter-Auto-Programm testen ... 156
12.4 Ausblick und Erweiterungen ... 158

13 Dein Roboter-Auto in Aktion: Parcours, Spiele und Wettbewerbe 159

13.1 Parcours-Designer: Erschaffe deine eigenen Rennstrecken und Testgelände 160

13.2 Spannende Spielideen und Wettbewerbe für dich und deine Freunde 161

13.2.1 Roboter-Rennen: Wer ist der Champion auf vier Rädern? 161

13.2.2 Präzisionsfahren: die hohe Kunst der Roboter-Steuerung 161

13.2.3 Ballspiele: Anpfiff zum Roboter-Fußball und Kegel-Cup! 162

13.2.4 Abenteuer-Parcours: über Stock und über Stein 162

13.3 Fazit und Ausblick: die nächste Stufe der Roboter-Intelligenz 163

Teil II Prompte deine Roboter-Auto-Programme

14 Programmieren mit KI-Unterstützung: deine Prompt-Schule 167

14.1 Die Kunst des Befehls: Was ist ein Prompt und warum ist »Prompt Engineering« wichtig? 167

14.2 Werkzeuge des Dialogs: wichtige Prompting-Techniken 168

14.2.1 Allgemeines Prompting / Zero-Shot-Prompting 169

14.2.2 One-Shot- und Few-Shot-Prompting (Prompting mit Beispielen) 169

14.2.3 System-, Kontext- und Rollen-Prompting 169

14.2.4 Rollen-Prompting genauer betrachtet 171

14.2.5 Chain-of-Thought-Prompting (CoT) 173

14.2.6 Der Aufbau macht den Unterschied: Struktur in Prompts 173

14.3 Dein perfekter Befehl: einen effektiven Prompt für das ESP32-Roboter-Auto erstellen 175

14.4 Der iterative Prozess: vom ersten Entwurf zum fertigen Code 178

14.5 Dokumentation deiner Prompts – behalte den Überblick 179

14.6 Realistische Erwartungen und deine Rolle als Entwickler 179

15 Sensoren im Überblick 181

15.1 Was sind Sensoren und wozu brauchen wir sie? ... 181

15.2 Der Ultraschallsensor (HC-SR04) – sehen mit Schallwellen ... 183

15.2.1 Wie funktioniert der HC-SR04-Ultraschallsensor? ... 184

15.2.2 Anschluss des HC-SR04 an den ESP32 ... 186

15.3 Digitaler Lichtsensor (GY-302/BH1750) – präzise Helligkeit messen ... 186

15.3.1 Wie funktioniert der digitale Lichtsensor BH1750? ... 187

15.3.2 Anschluss an den ESP32 über den I2C-Bus ... 188

15.4 Infrarotsensoren/Liniensensoren – dem Weg folgen ... 189

15.4.1 Wie funktioniert der Linienfolger-Sensor KY-033? ... 190

15.4.2 Anschluss an den ESP32 und Einsatz mehrerer Sensoren ... 191

15.5 Das Gyroskop (GY-521/MPU-6050) – Drehbewegungen und Lage erfassen ... 193

15.6 Das Kompassmodul – dein Roboter findet die Nordrichtung ... 196

15.6.1 Anschluss an den ESP32 über den I2C-Bus ... 197

15.6.2 Eine kurze Einführung in das Kompassmodul ... 197

15.7 Weitere interessante Sensoren für Roboter-Projekte ... 198

16 Präzises Lichtmanagement – dein Roboter misst Helligkeit in Lux mit dem BH1750 201

16.1 Der digitale Lichtsensor BH1750 – was er kann und warum er praktisch ist ... 201

16.2 Das GY-302-Modul (BH1750) am Roboter montieren und anschließen ... 202

16.2.1 Montage des Moduls ... 202

16.2.2 Anschluss an den ESP32 über I2C ... 202

16.3 Programmieren: Lichtstärke in Lux auslesen und auf dem OLED-Display anzeigen ... 203

16.3.1 Prompt für die Programmierung ... 204

16.3.2 Erläuterung des generierten Codes ... 206

16.3.3 Programm auf den ESP32 aufspielen ... 207

17 Auf der Ideallinie – dein Roboter folgt Spuren mit dem KY-033-Liniensensor 209

17.1 Was bedeutet Linienverfolgung? ... 209

17.2 Den KY-033-Liniensensor am Roboter montieren und anschließen ... 211

17.2.1 Montage der Sensoren ... 211

17.2.2 Anschluss an den ESP32 ... 212

17.2.3 Einstellung der Empfindlichkeit ... 212

17.3 Programmieren: Linienstatus pro Sensor erkennen ... 213

17.3.1 Prompt für die Programmierung ... 213

17.3.2 Erläuterung des generierten Codes ... 216

17.3.3 Programm auf den ESP32 aufspielen ... 217

17.4 Das Programm für die Linienverfolgung in Verbindung mit der Motorsteuerung: Wo ist nur die Linie hin? ... 217

17.4.1 Prompt für die Programmierung ... 218

17.4.2 Erläuterung des generierten Codes ... 222

17.5 Testfahrten und Kalibrierung ... 223

17.6 Problem mit dem Code? ... 224

18 Balanceakt und Drehmomente – dein Roboter versteht Bewegung 225

18.1 Gyroskop und Beschleunigungssensor kurz erklärt ... 225

18.2 Den MPU-6050-Sensor am Roboter-Auto montieren und anschließen ... 226

18.2.1 Montage des Moduls ... 227

18.2.2 Anschluss an den ESP32 ... 227

18.3 Programmieren: Rohdaten von Gyroskop und Beschleunigungssensor auslesen (mit OLED-Anzeige) ... 228

18.3.1 Prompt für die Programmierung ... 228

18.3.2 Erläuterung des generierten Codes ... 231

18.3.3 Programm auf den ESP32 aufspielen ... 232

18.4 Logik: von Rohdaten zu Winkeln – Neigung und Drehung verstehen ... 233

18.4.1 Neigungswinkel (Pitch und Roll) aus Beschleunigungsdaten ... 233

18.4.2 Drehwinkel (Yaw, Pitch, Roll) aus Gyroskopdaten ... 234

18.4.3 Sensorfusion – ganz neue Möglichkeiten durch die Kombination von Sensordaten (Ausblick) ... 234

18.5 Das Programm, um den Roboter um 45 Grad zu drehen ... 234
18.5.1 Prompt für die Programmierung ... 235
18.5.2 Erläuterung des generierten Codes ... 240
18.5.3 Programm auf den ESP32 aufspielen ... 241
18.5.4 Verbesserungen für zukünftiges Prompting ... 242
18.6 Testfahrten bzw. Drehung ... 242

19 Immer nordwärts – dein Roboter-Auto findet die Himmelsrichtung 243

19.1 Was ist ein Magnetometer und wie wird daraus ein Kompass? ... 243
19.2 Das GY-271-Modul am Roboter montieren und anschließen ... 244
19.3 Anschluss an den ESP32 ... 245
19.4 Programmieren: Magnetfeld-Rohdaten und Kompassrichtung auslesen ... 245
19.4.1 Prompt für die Programmierung ... 246
19.4.2 Erläuterung des generierten Codes ... 250
19.4.3 Programm auf den ESP32 aufspielen ... 251
19.5 Erweiterungen und Verfeinerungen ... 252
19.6 Kompasskalibrierung des QMC5883L – ein Muss für präzise Navigation ... 253

20 Dein Roboter fährt allein – autonom mit Ultraschallsensor 255

20.1 Den Ultraschallsensor am Roboter montieren ... 256
20.2 Prompt für die Programmierung ... 256
20.2.1 Erläuterung des generierten Codes ... 257
20.3 Testfahrten und Fehlersuche ... 259

21 Dein Roboter findet seinen Platz in der Welt – Navigation mit dem GPS-Modul 261

21.1 Was ist GPS und wie wird daraus eine Position? ... 262
21.2 Das GY-NEO6MV2-Modul am Roboter montieren ... 262
21.3 Anschluss an den ESP32 ... 263

21.4 Programmieren: GPS-Position und Zeit auslesen ... 263
21.4.1 Prompt für die Programmierung ... 264
21.4.2 Erläuterung des generierten Codes ... 265
21.4.3 Programm auf den ESP32 aufspielen ... 266
21.5 Erweiterungen und Verfeinerungen ... 266

22 Steuerung per Browser – das Web-Interface 269

22.1 Was ist ein Web-Interface und warum ist es nützlich für dein Roboter-Auto? ... 269
22.1.1 Der ESP32 als Access Point oder im Heimnetzwerk ... 270
22.2 Grundlagen: einen einfachen Webserver auf dem ESP32 aufsetzen ... 270
22.3 Prompt für die Programmierung ... 271
22.3.1 Erläuterung des generierten Codes ... 272
22.3.2 Programm auf den ESP32 aufspielen ... 272
22.4 Herausforderungen und Weiterentwicklungsmöglichkeiten ... 274

23 Mehr Bewegung: Servo-Motoren für dein Roboter-Auto 275

23.1 Was sind Servo-Motoren und wie unterscheiden sie sich von den Antriebsmotoren? ... 276
23.1.1 Anwendungsbeispiele für Servos am Roboter-Auto ... 276
23.2 Das PCA9685-Modul und die Ansteuerung von Servos ... 277
23.2.1 Warum nutzen wir das PCA9685-Modul dafür? ... 277
23.2.2 Spannungsversorgung der Servo-Motoren ... 277
23.2.3 Anschluss eines Servo-Motors an das PCA9685 ... 278
23.3 Einbau des Servo-Controllers und Programmierung ... 279
23.3.1 Prompt für die Programmierung ... 280
23.3.2 Generierter Programmcode – drehe den Servo-Motor ... 280
23.3.3 Erläuterung des generierten Codes ... 281
23.3.4 Programm auf den ESP32 aufspielen ... 282

Teil III Level Up: Das Roboter-Auto fährt autonom

24 Dein Roboter-Auto wird intelligent – lerne das OpenBot Framework kennen 287

24.1 Was du im dritten Teil des Buches lernen wirst ... 288

24.2 Eine neue Dimension der Autonomie – dein Smartphone als Robotergehirn ... 289

24.3 Vorstellung des OpenBot-Projekts ... 289

24.3.1 Die Kernidee – Smartphone als Gehirn, ESP32 als Steuerzentrale ... 290

24.4 Zusätzlich benötigte Komponenten zum bestehenden Roboter-Auto ... 292

25 Das OpenBot-System – Zusammenspiel von Smartphone und ESP32-Mikrocontroller 297

25.1 Die Architektur im Detail ... 297

25.1.1 Smartphone-Seite ... 297

25.1.2 ESP32-Seite der Roboter-Auto-Controller ... 298

25.1.3 Kommunikation über Bluetooth oder USB-Kabel ... 298

25.2 Überblick über die autonomen Fähigkeiten mit OpenBot ... 299

26 Vom Roboter-Auto zum OpenBot: Vorbereitung und Firmware-Installation 301

26.1 Letzte Hardware-Vorbereitung: die Smartphone-Halterung ... 301

26.2 Die OpenBot-Firmware: Vorbereitung und Konfiguration ... 302

26.3 Die Firmware konfigurieren ... 302

26.3.1 Robotertyp festlegen ... 303

26.3.2 GPIO-Pin-Belegung für dein Roboter-Auto anpassen ... 303

26.4 Firmware auf den ESP32 aufspielen ... 306

26.5 Der erste Test ... 307

26.6 Die OpenBot-Smartphone-App – Installation und erste Verbindung ... 307

26.6.1 Installation und Einrichtung der App ... 308

26.6.2 Bluetooth-Verbindung zwischen Smartphone und ESP32 herstellen ... 309

26.7 Die Benutzeroberfläche der OpenBot-App ... 312
26.7.1 Der Diagnose-Modus: der Bildschirm »Robot Info« ... 312
26.7.2 Der Modus »Controller Mapping« ... 314
26.7.3 Der Modus »Freies Fahren / Free Roam« ... 315
26.7.4 Der Modus »Daten sammeln / Data Collection« ... 319
26.8 Die Benutzeroberfläche der OpenBot-App (AI) ... 323
26.8.1 Der Modus »Autopilot« ... 323
26.8.2 Der Modus »Objektverfolgung / Object Tracking« ... 324
26.8.3 Der Modus »Ziel setzen / Set Goal« ... 325
26.8.4 Das Model Management ... 328

27 Datensammlung und KI-Training in der Praxis: eine Schritt-für-Schritt-Anleitung 331

27.1 Trainingsdaten sammeln ... 331
27.2 Vom Rohstoff zum Gehirn – dein eigenes KI-Modell trainieren ... 335
27.2.1 Die Werkstatt einrichten: Vorbereitung der Trainingsumgebung am Windows-PC ... 336
27.2.2 Das Training kopieren, konfigurieren und starten ... 339
27.2.3 Am Mischpult der KI: die Trainingsparameter der Web-App erklärt ... 341
27.2.4 Das Abschlusszeugnis: die Trainingsergebnisse verstehen ... 344
27.3 Das fertige KI-Modell: Export und Übertragung auf das Smartphone ... 346
27.3.1 Schritt 1: Das fertige Modell auf dem PC finden ... 347
27.3.2 Schritt 2: Das Smartphone mit dem PC verbinden ... 348
27.3.3 Schritt 3: Die Modelldatei auf das Smartphone kopieren ... 349
27.3.4 Schritt 4: Die Modelldatei in die OpenBot-App importieren ... 349
27.4 Fazit: Die erste autonome Fahrt mit deinem eigenen Modell ... 350

28 Der visuelle Baukasten – eigene Programme mit OpenBot Playground erstellen 351

28.1 Der OpenBot-Online-Dienst: einfach im Browser loslegen ... 352
28.2 Die magische Verbindung: PC und Roboter koppeln ... 352
28.3 Der Baukasten: Logik und KI mit Blöcken erschaffen ... 352

29 Dein intelligenter Roboter – Zusammenfassung und nächste Abenteuer 355

29.1 Was du gelernt hast: dein Weg zum Roboter-Entwickler 355

29.2 Was jetzt möglich ist: deine Reise beginnt hier 356

30 Die abschließenden Worte: Was du bis jetzt erreicht hast und wie es weitergeht 357

30.1 Was Kinder und auch Erwachsene mitnehmen können 358

30.2 Was du als Nächstes machen kannst 358

30.3 Das ESP32-Roboter-Auto-Projekt in Familie und Schule 359

30.4 Wie ein humanoider Roboter zukünftig mit KI-Unterstützung Probleme lösen könnte – einfach erklärt 359

30.5 Gedanken über die Zukunft des Zusammenlebens mit Large Language Models 361

30.6 Und jetzt? Feiern und Teilen! 364

Index 365

Index

18650-Akkus ... 82, 85
interner Widerstand ... 86
Kapazitätstest ... 85
Sicherheit ... 83
3D-Druck ... 37

A

Abisolierzange ... 30
Accelerometer ... 226
Access Point ... 270
Additive Fertigung ... 54
Adhesion ... 41
Anaconda PowerShell Prompt ... 336
Arbeitstemperatur ... 32
ARCore ... 325
Arduino ... 149
Bibliotheken ... 152
IDE ... 149
Arduino IDE ... 149
Async TCP by ESP32Async ... 273
Autopilot-Modus ... 323

B

Ballspiele ... 162
Batteriehalter ... 86
Battery Management System (BMS) ... 84
Beschleunigungssensor ... 225
BH1750 ... 187, 201
Bit ... 72
Boot-Button ... 154
Brim ... 41

C

CAD-Programm ... 38
Chain-of-Thought ... 173
Chassis
Getriebemotorenhalterung ... 44
ChatGPT ... 167, 361
Checkpoint-Ordner ... 347
Common Collector ... 70
COM-Port ... 153
Computer-Aided Design ... 38
CP210x-Chip ... 153

D

Data Collection ... 319
Daten sammeln ... 319
DC-DC-Wandler ... 88
DC-Motoren ... 80
Drehmoment ... 64
Dritte Hand ... 31, 35
DuPont ... 94

E

Ein-/Ausschalter ... 142
Einbau-Druckschalter ... 97
Elektrische Leistung ... 63
Elektrizitätslehre
Gesamtspannung (U) ... 61
Parallelschaltung ... 62
Reihenschaltung ... 60
Stromstärke (I) ... 60
Widerstand (R) ... 60
Entlötlitze ... 33
Entlötpumpe ... 34
Erdung ... 70
ESP32-Mikrocontroller ... 77
ESP Async WebServer ... 272

F

Fehlerkurve ... 345
Feinsicherung ... 86
Few-Shot-Prompt ... 169
Filament ... 40, 54
Flussmittel ... 33

FreeCAD 39
Free Roam 315
Freies Fahren 315
Fülldichte 41
Fused Deposition Modeling (FDM) 54
Fused Filament Fabrication (FFF) 54
Fusion 360 39

G

Gamecontroller 91
G-Code 40
Gemini 167
General Purpose Input/Output 70
Git 336
GND 70
GPIO 70
GPS-Modul 261
Grove-I2C-Hub 74
GY-302 202
GY-302/BH1750 186
GY-521 ..... 193, 225
GY-NEO6MV2 261
Gyroskop 193, 225

H

Haftung 41
Hard-Iron-Effekt 198
HC-SR04 ..... 183, 255
Heißklebepistole 30
Hexadezimalzahl 188
Hinderniserkennung 258
Human in the loop 360

I

I2C (Inter-Integrated Circuit) 69
I2C-Bus 69–70
I2C-Hub 73
IDE 149
Infill 41
Infrarotsensor 189
Integrated Development Environment 149
Interner Widerstand 129

J

JST-Stecker 95
Jumper-Kabel 94
Female-to-Female 94
Female-to-Male 94
Male-to-Male 94

K

Kabelbinder 31, 96
Kalibrierung 253
Kalte Lötstellen 32
KI-Assistenten 167
KI-Training 331
Klebeband 31
Klemmbausteine 37
Kompass 244
Kompassmodul
Kalibrierung 197
Magnetische Deklination 197
Magnetische Störungen 197
Kontext-Prompting 170
Krokodilklemmen 35
Kupferdraht 96
Kurzschluss 87
KY-033 ..... 189, 211

L

L298N-Motortreiber 80
Large Language Model (LLM) 167, 362
multimodal 360
LEARNING_RATE 341
Lichtsensor 186, 201
Liniensensor 189, 211
Linienverfolgung 209
Lithium-Ionen-Akkus → 18650-Akkus
LLM → Large Language Model (LLM)
Lochstreifenplatine 74
Loss 345
Lötkolben 31
Lötrauchabsaugung 33
Lötspitze 34
Lötstelle 32
Lötzinn 31
Lüsterklemmen 97

M

Magnetometer ... 243
Makerspace ... 40
Master-Slave-Schaltung ... 71
Miniconda ... 336
Motortreiber ... 79
MPU-6050 ... 193, 225–226
Multimeter ... 59

N

NeoPixel ... 89
Nervensystem ... 69
NUM_EPOCHS ... 342
NVIDIA ... 335

O

Objektverfolgung ... 324
Ohm'sche Gesetz ... 63
OLED-Display ... 81, 136
One-Shot-Prompt ... 169
OpenBot ... 287
  Build Tools ... 336
  Checkpoint-Ordner ... 347
  Firmware ... 301
  Playground ... 351
  Smartphone-App ... 288
  Web-App ... 336
Overfitting ... 348

P

Pappe ... 37
Parallelschaltung ... 61, 131
Parcours ... 160
PCA9685-Modul ... 275
Pegelwandler ... 76
Periode ... 80
Physics AI ... 360
Pinzette ... 30
Pmax ... 64
Polung (Motor) ... 132
Polyethylenterephthalatglycol (PETG) ... 55
Polylactide (PLA) ... 55
Präzisionsfahren ... 161
Prompt ... 167
  Chain-of-Thought ... 173
  Few Shot ... 169
  Kontext ... 170
  One-Shot ... 169
  Rolle ... 171
  System ... 169
  Zero-Shot ... 169
Prompt Engineering ... 168
Pulsweite ... 80
Pulsweitenmodulation (PWM) ... 80, 124
PWM-Signal ... 80

R

Reihenschaltung ... 60
RGB-LEDs ... 144
Rollen-Prompting ... 171

S

Schiebeschild ... 162
Schrumpfschlauch ... 30, 96
Schutzbrille ... 31
SCL, Serial Clock Line ... 69
SDA, Serial Data Line ... 69
Seeed Studio ... 74
Seitenschneider ... 30
Sensorfusion ... 194
Serieller Monitor ... 153
Servo-Motoren ... 275
Set Goal ... 325
Sicherungshalter ... 86
Silikonkabel ... 95
Skalpell ... 30
Skirt ... 41
Slicer-Software ... 40
Smartphone-Halterung ... 293
Soft-Iron-Effekt ... 198
Spannungsanzeige ... 92, 140
Spannungswandler ... 88, 133
  Step-Down-Konverter ... 88
SSD1306 ... 81
Step-Down-Konverter ... 88

Roboter-Autos mit dem ESP32

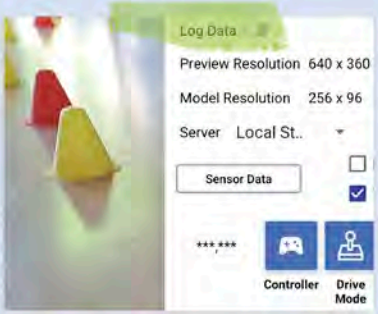
Fahrspaß inklusive!

Für kleine und große Bastler

Wer braucht schon einen Tesla, wenn man ein eigenes E-Auto bauen kann? Dein Flitzer ist zwar etwas kleiner als ein richtiges Auto, aber du kannst ihn selbst programmieren und ihm sogar beibringen, autonom zu fahren. Ingmar Stapel zeigt dir, wie's geht.

Dein neues Auto

Alles richtig verkabeln



Die KI lernt fahren

Dein Roboter-Auto im Selbstbau

Mit einigen Handgriffen und überraschend wenigen Teilen baust du dein eigenes Auto. Der ESP32 steuert alles, die restlichen Teile sind schnell besorgt oder im 3D-Druck erstellt. Wie alles zusammenpasst, zeigen dir detaillierte Schritt-für-Schritt-Anleitungen.

Programmierung mit KI

Programmieren ist nicht schwer, wenn du dir ein bisschen Hilfe holst. Mit künstlicher Intelligenz schreibst du Code, der den Roboter steuert. Das richtige Prompting sorgt dafür, dass die KI das macht, was du willst. So behältst du immer die Kontrolle und verstehst, was passiert.

Autonomes Fahren

Du steuerst dein Auto mit einem Controller – aber wäre es nicht cool, der KI das Lenkrad zu überlassen? Mit OpenBot und einem Smartphone trainierst du dein Auto, damit es ganz von allein fährt.

Getestete Codebeispiele und Vorlagen für den 3D-Druck

Ingmar Stapel baut seit Jahren kleine Roboter, die durch die Wohnung flitzen oder sich im rauen Gelände beweisen. Beispiele findest du unter *www.custom-build-robots.com*.

Auf einen Blick

- Werkzeuge und Teile
- Chassis aus dem 3D-Druck
- Antrieb, Akku und Verkabelung: die Energie für dein Auto
- Das Gehirn: das ESP32 Dev Kit
- Schritt für Schritt: die Montage des Autos
- Programmierung mit KI: richtiges Prompting
- Steuerung mit Codebeispielen
- Sensoren auslesen, Spuren folgen und immer im Gleichgewicht bleiben
- Autonom fahren mit Ultraschallsensor und GPS-Modul
- Dein Auto wird smart: das OpenBot-Framework
- Daten sammeln und KI trainieren

Rheinwerk Computing

Gedruckt in Deutschland
Mineralölfreie Druckfarben
Zertifiziertes Papier

€ 29,90 [D] € 30,80 [A]

Maker
ISBN 978-3-367-11144-2