
PIO-Programmierung mit dem Raspberry Pi Pico

Praxisnahe Projekte, Profi-Tricks und
eigene Anwendungen



Hans-Joachim Seeger

● © 2026: Elektor Verlag GmbH, Aachen.

1. Auflage 2026

● Alle Rechte vorbehalten.

Die in diesem Buch veröffentlichten Beiträge, insbesondere alle Aufsätze und Artikel sowie alle Entwürfe, Pläne, Zeichnungen und Illustrationen sind urheberrechtlich geschützt. Ihre auch auszugsweise Vervielfältigung und Verbreitung ist grundsätzlich nur mit vorheriger schriftlicher Zustimmung des Herausgebers gestattet.

Die Informationen im vorliegenden Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht. Die in diesem Buch erwähnten Soft- und Hardwarebezeichnungen können auch dann eingetragene Warenzeichen sein, wenn darauf nicht besonders hingewiesen wird. Sie gehören dem jeweiligen Warenzeicheninhaber und unterliegen gesetzlichen Bestimmungen.

Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Verlag, Herausgeber und Autor können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Für die Mitteilung eventueller Fehler sind Verlag und Autor dankbar.

● Erklärung

Autor und Verlag haben sich nach bestem Wissen und Gewissen bemüht, die Richtigkeit der in diesem Buch enthaltenen Informationen sicherzustellen. Sie übernehmen jedoch keine Haftung gegenüber Dritten für Verluste oder Schäden, die durch Fehler oder Auslassungen in diesem Buch entstehen, unabhängig davon, ob diese Fehler oder Auslassungen auf Fahrlässigkeit, Unglücksfälle oder andere Ursachen zurückzuführen sind

● **978-3-89576-736-4** Print
978-3-89576-737-1 eBook
978-3-89576-738-8 ePub

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

Elektor Verlag GmbH, Aachen
www.elektor.de

Elektor ist die weltweit wichtigste Quelle für technische Informationen und Elektronik-Produkte für Maker, Ingenieure und Elektronik-Entwickler und für Firmen, die diese Fachleute beschäftigen. Das internationale Team von Elektor entwickelt Tag für Tag hochwertige Inhalte für Entwickler und DIY-Elektroniker, die über verschiedene Medien (Magazine, Videos, digitale Medien sowie Social Media) in zahlreichen Sprachen verbreitet werden. **www.elektor.de**

Inhaltsverzeichnis

Warnhinweise	11
Programmdownload	12
Einführung	13
Teil 1 • Grundlagen	15
1.1. Einführung	15
1.1.1. Einleitung	15
1.1.2. Warum PIO so besonders ist	15
1.1.3. Warum ist PIO kaum bekannt	15
1.1.4. Ziele des Buches	16
1.1.5. Was am Ende möglich ist	16
1.1.6. Aufbau des Buches	16
1.1.7. Zielgruppe	17
1.1.8. Benötigte Kenntnisse	17
1.2. Was ist PIO?	17
1.2.1. Die Idee hinter Programmable I/O (PIO)	17
1.2.2. Aufbau der PIO-Hardware	17
1.2.3. Wie PIO vom RP2040 gesteuert wird	19
1.2.4. PIO versus CPU: Wer macht was?	19
1.3. Die PIO-Sprache und ihre Befehle	19
1.3.1. Was ist die PIO-Sprache	19
1.3.2. Grundprinzip	20
1.3.3. Die wichtigsten PIO-Befehle	20
1.3.4. Wichtige Begriffe (Überblick)	24
1.4. Erste praktische Schritte	24
1.4.1. Grundlagen	24
1.4.2. Softwarevoraussetzungen	25
1.4.3. Aufbau der PIO-Hardware	25
1.4.4. Arbeitsweise im Überblick	25
1.4.5. Was PIO ist und wie es arbeitet	26
1.4.6. Aufbau der PIO-Einheit	26

1.4.7. Wie arbeitet eine State Machine	26
1.4.8. Datenfluss zwischen CPU und PIO	27
1.4.9. Synchronisation und Timing	27
1.4.10. LED liegt an 3,3 V oder GND	27
1.5. Die Frequenz des PIO	29
1.5.1. Die Frequenz der PIO-State Machine	29
1.5.2. Grenzen der Frequenz	30
1.6. Mein erstes Programm mit PIO.	31
1.6.1. Programm auf dem LED blinken lassen (kein PIO)	31
1.6.2. LED blinkt autonom per PIO	32
1.7. GPIO-IRQ versus PIO-IRQ	34
1.7.1. Zwei völlig unterschiedliche Interrupt-Systeme	34
1.7.2. GPIO-IRQ – CPU reagiert auf Pin-Ereignisse	35
1.7.3. PIO-IRQ – PIO meldet ein Ereignis an die CPU.	35
1.7.4. Kombination beider Systeme.	36
1.8. PIO und Interrupts	37
1.8.1. CPU ↔ PIO-Kommunikation über IRQs.	37
1.8.2. Wie PIO-IRQs im Python-Code ankommen	37
1.8.3. Reaktionszeit: Wie schnell sind PIO-IRQs	38
1.8.4. Synchronisierung zwischen CPU und PIO	38
1.8.5. Synchronisierung über IRQ	40
1.9. Was ist "IRQ Cross Signaling"?	40
1.9.1. Warum braucht man das?	41
1.9.2. Warum ist das so nützlich?	42
1.10. Zusammenarbeit von PIO und CPU	47
1.10.1. Zusammenarbeit von PIO und CPU verstehen	47
1.10.2. Das perfekte Team	47
1.10.3. Vorteile der Aufgabenverteilung.	49
1.10.4. LED blinkt mit Zeiten aus Python	49
1.11. WS2812 – Aufbau, Funktion und Anwendung	53
1.11.1. Was sind WS2812-LEDs?	53
1.11.2. Aufbau der WS2812-LED	53

1.11.3. Funktionsprinzip	54
1.11.4. Ein-Draht-Protokoll.	54
1.11.5. Stromversorgung	54
1.11.6. Anwendungen	55
1.11.7. Vorteile der WS2812	55
1.11.8. Nachteile	56
1.11.9. Warum WS2812 für Einsteiger ideal sind.	56
1.12. PWM und Signalerzeugung mit PIO	56
1.12.1. Grundlagen der PWM	56
1.12.2. PIO: Programmable Input/Output	56
1.12.3. Signalerzeugung mit PIO	57
1.12.4. Anwendungsmöglichkeiten	57
1.12.5. Praktisches Beispiel	57
1.12.6. Zusammenfassung	58
1.13. PIO & DMA: Hochgeschwindigkeits-Datentransfer ohne CPU-Belastung	58
1.13.1. Funktion von DMA (Direct Memory Access)	58
1.13.2. Warum DMA so gut zu PIO passt	58
1.13.3. Wie PIO & DMA zusammenarbeiten	59
1.13.4. Anwendungsbeispiele für PIO + DMA	60
1.14. Checkliste: Optimierung für dein eigenes Projekt	61
1.14.1. CPU / Python-Teil	61
1.14.2. PIO / StateMachine	61
1.14.3. IRQ / Cross-IRQ.	62
1.15. Debugging und Optimierung (Performance, Fehleranalyse und Debugging)	62
1.15.1. Syntax- und Logikfehler im Python-Teil.	62
1.15.2. PIO-Fehler.	63
1.15.3. IRQ-Fehler	63
1.15.4. Speicherfehler	64
1.15.5. Performance messen	64
1.15.6. Tipps für effiziente Programme	65
1.15.7. Zusammenfassung	67
1.16. Fortgeschrittene Anwendungen.	67

1.16.1. Mögliche Projekte / Experimente	67
1.17. Abschlussprojekte und Tests	68
1.17.1. Beispiele für Abschlussprojekte	68
1.17.2. Tipps für Fortgeschrittene und Abschlussprojekte.	69
Teil 2 – Hardware	70
2.1. Verwendete Hardware.	70
2.1.1. Hinweis zu Pico und Pico 2	70
2.1.2. Taster-Beschaltung	70
2.1.3. LED-Beschaltung	71
2.1.4. Verwendete Breadboard und Platinen mit Pico.	72
2.1.5. Platine E/A Modul 1 mit 8x LED und 8x Taster (P113a)	75
2.1.7. Platine mit Servo (P136_1)	77
2.1.8. Platine mit WS2812 (P130).	78
2.1.9. Platine mit Poti (P131)	79
2.1.10. Platine mit SPI-Display (P132a).	80
2.1.11. Platine Eingabe 1 (P119).	81
2.1.12. Schaltung E/A Modul 1 mit 8x LED und 8x Taster	82
2.1.13. Schaltung Encoder	83
2.1.14. Schaltung Servo	83
2.1.15. Schaltung WS2812.	84
2.1.16. Schaltung Poti	84
2.1.17. Schaltung Poti 1.	85
2.1.18. Schaltung SPI-Display	85
2.1.19. Schaltung Eingabe 1	86
Teil 3 – Programme mit PIO.	87
3.1. LED-Programme.	87
3.1.1. LED-Blinken mit nop	87
3.1.2. LED-Blinken mit Schleifen	89
3.1.3. Wechselblinker mit PIO und Schleife.	94
3.1.4. Warnblinker mit PIO und sicherem Ein/Aus	96
3.1.5. Lauflicht mit 8 LEDs	98
3.1.6. Knight Rider-Lauflicht mit 8 LEDs.	100

3.1.7. Kirmes-Lichtsteuerung mit 8 LEDs	103
3.1.8. Karussell-Licht mit 8 LEDs	105
3.1.9. 2 LEDs an 2 Pins – ohne GND und 3,3 V	107
3.2. LED-Taster-Programme	109
3.2.1. Taster schaltet LED	109
3.2.2. Taster toggelt LED	111
3.2.4. Taster schaltet Warnblinker.	115
3.2.5. Taster schaltet Warnblinker.	117
3.2.6. 4 Taster 4 LED schalten mit IRQ und Anzeige Funktion.	118
3.2.7. 4 Taster 4 LED schalten mit Anzeige kurz oder lang	121
3.2.8. LED-blinken mit unterschiedlichen Zeiten	123
3.2.9. 5 Taster mit SPI und Menü Auswahl	126
3.3. Encoder Programm	130
3.3.1. Encoder mit 3 LEDs	130
3.4. Servo Encoder-Programme	135
3.4.1. Servo mit Encoder steuern und Angabe Winkel in Grad	135
3.4.2. Servo mit Poti steuern	138
3.4.3. Servo mit Encoder steuern, Angabe Winkel und SPI-Display	141
3.5. WS2812 – LEDs (NeoPixel)	144
3.5.1. WS2812 1 LED mit Regenbogenfarben ansteuern	144
3.5.2. WS2812 8 LEDs mit Regenbogenfarben ansteuern.	146
3.5.3. WS2812 64 LEDs mit Regenbogenfarben ansteuern.	148
3.5.4. WS2812 64 LEDs mit Lauflicht ansteuern	150
3.5.5. WS2812 64 LEDs mit Zufall.	153
3.5.6. WS2812 64 LEDs mit Muster.	154
3.6. PWM mit PIO	159
3.6.1. PWM PIO-Dimmen mit einer LED	159
3.6.2. PWM PIO-Dimmen mit einer LED auf ab	162
3.6.3. PWM PIO-Dimmen mit vier LED	164
3.6.4. PIO Servo-Tester Automatic mit Anzeige auf der Konsole	167
3.6.5. PIO Servo-Tester Automatic mit SPI-Display	169
3.6.6. PIO 4 Taster 4 LEDs mit CPU IRQ.	172

3.7. Cross-IRQ zwischen zwei PIO-State Machines	175
3.7.1. PIO-LED mit Taster Cross	175
3.8. Frequenzerzeugung	180
3.8.1. PIO-Rechteckgenerator mit fester Frequenz	180
3.8.2. PIO-Rechteckgenerator mit variabler Duty Cycle	183
3.8.3. PIO-Rechteckgenerator mit Poti.	188
3.8.4. PIO-Rechteckgenerator mit Poti und Display	190
3.8.5. PIO 5-Taster Menü mit Submenü	193
Programmliste	199
Glossar – MicroPython PIO, SPI, I²C, PWM und Displaysteuerung	201
Komponentenliste	204
Literatur	205
Index	206

Warnhinweise

- Die Schaltungen und Boards dürfen ausschließlich mit geprüften, doppelt isolierten Sicherheitsnetzgeräten betrieben werden. Isolationsfehler bei einfachen oder ungeeigneten Netzteilen können zu gefährlichen Berührungsspannungen an Bauteilen oder Leitungen führen.
- Alle Komponenten dürfen nur innerhalb ihrer vorgesehenen Spannungs- und Strombereiche betrieben werden. Falsche Spannungspegel – insbesondere über 3,3 V am Raspberry Pi Pico – können sofortige und dauerhafte Schäden verursachen.
- Der Raspberry Pi Pico sowie angeschlossene Module erzeugen empfindliche CMOS-Signale. Elektrostatische Entladungen (ESD) können Bauteile zerstören. Verwendung einer antistatischen Unterlage und vorsichtiges Arbeiten werden empfohlen.
- Leistungsstarke LEDs, insbesondere Hochleistungs- oder RGB-LEDs, können Blendungen und dauerhafte Augenschäden verursachen. Niemals direkt in helle Lichtquellen schauen.
- Beim Umgang mit LED-Streifen, NeoPixel/WS2812 und Displays kann eine hohe Stromaufnahme entstehen. Die Stromversorgung muss ausreichend dimensioniert sein, um Überlastungen und Überhitzung zu vermeiden.
- Beim Aufbau auf dem Breadboard ist auf sichere und feste Kontaktierung zu achten. Lose oder wackelige Verbindungen können Kurzschlüsse, Fehlfunktionen oder Beschädigungen des Mikrocontrollers verursachen.
- Taster und mechanische Schalter erzeugen Prellen. Unsachgemäße Entprellung kann zu Fehlauflösungen führen, besonders bei programmgesteuerten Menüs oder PIO-Abfragen.
- Bei der Arbeit mit PIO-Programmen können sehr hohe Frequenzen erzeugt werden. Fehlkonfigurationen können zu instabilen Zuständen oder unerwünschtem Verhalten der Peripherie führen.
- Der Autor übernimmt keinerlei Haftung für Schäden, die durch unsachgemäße Nutzung, Fehlverdrahtung, falsche Spannungen oder fehlerhafte Programme entstehen.
- Alle Programme und PIO-Beispiele sind zu Lehr- und Experimentierzwecken gedacht und dürfen nur unter Aufsicht oder mit ausreichendem technischem Verständnis verwendet werden.

- Vor Änderungen an der Verdrahtung oder Bauteilen muss das gesamte System vom Strom getrennt werden, um Kurzschlüsse oder Verletzungen zu vermeiden.

Programmdownload

Die vollständigen Programme aller im Buch verwendeten Projekte können kostenlos von der Elektor-Store Webseite

www.elektor.de

heruntergeladen werden. Sofern ein Programm nicht identisch mit dem im Buch beschrieben ist, sollte die Version aus dem Download verwendet werden, da es sich dabei um die aktuelle Version handelt.

Einführung

Willkommen zu deinem Einstieg in die faszinierende Welt der **PIO-Programmierung** auf dem **Raspberry Pi Pico**!

Dieses Buch richtet sich an alle **Maker, Tüftler und Technikbegeisterten**, die verstehen möchten, was in einem Mikrocontroller wirklich passiert – und die Spaß daran haben, Dinge selbst auszu-probieren. Ganz gleich, ob du schon Erfahrung mit Mikrocontrollern hast oder zum ersten Mal mit dem Pico arbeitest: Hier lernst du Schritt für Schritt, wie du mit **PIO (Programmable I/O)** eigene, flexible und extrem schnelle Hardwarefunktionen programmierst.

PIO ist eines der mächtigsten, aber auch am wenigsten genutzten Werkzeuge des Raspberry Pi Pico. Mit PIO kannst du Aufgaben übernehmen, die sonst nur teure Peripheriebausteine oder komplexe Timer lösen – und das mit wenigen Zeilen Code. Du kannst Signale erzeugen, Protokolle nachbilden, LEDs ansteuern oder sogar ganze Schnittstellen selbst entwickeln.

Ich habe dieses Buch geschrieben, weil viele Maker beim ersten Kontakt mit PIO zurückschrecken: "Assembler? State Machines? Das klingt kompliziert!" Aber keine Sorge – wir nehmen dich an die Hand. Jedes Kapitel erklärt ein Konzept **verständlich**, zeigt dir **den Code**, und führt dich **praktisch** zu deinem eigenen funktionierenden Projekt.

Was dich erwartet

Das Buch ist in vier Teile gegliedert:

1. **Grundlagen** – Du lernst, wie PIO funktioniert, wie man es programmiert und mit MicroPython ansteuert.
2. **Erste Schritte** – Wir beginnen mit einfachen Projekten: LED blinken, Pins steuern, Timing verstehen.
3. **Praktische Projekte** – Du baust reale Anwendungen: UART, SPI, WS2812-LEDs, PWM und Audio.
4. **Fortgeschrittene Themen** – Du lernst, mehrere State Machines zu kombinieren, Interrupts zu nutzen und dein System zu optimieren.

Jedes Kapitel folgt demselben Aufbau:

- Einführung & Theorie
- Schritt-für-Schritt Anleitung mit Code
- Analyse & Verständnis
- Experimente und Erweiterungen

So bleibst du nie bei reiner Theorie stehen, sondern siehst sofort, wie sich dein Wissen in funktionierende Schaltungen umsetzen lässt.

Was du brauchst

Für die meisten Projekte genügt ein **Raspberry Pi Pico**, ein **Breadboard**, einige **LEDs**, **Widerstände**, **Taster** und **Kabel**.

Du kannst die Beispiele wahlweise in **MicroPython** oder **C/C++** umsetzen – wir konzentrieren uns im Buch auf MicroPython, weil es besonders einsteigerfreundlich ist und du sofort Ergebnisse siehst.

Wie du am besten lernst

Am meisten lernst du, wenn du mitmachst.

Tippe die Programme selbst ab, ändere Werte, beobachte das Verhalten und experimentiere. Wenn etwas nicht funktioniert: perfekt! Fehler sind in der Elektronik der beste Lehrer.

In späteren Kapiteln wirst du merken, dass du PIO fast "intuitiv" beherrschst – du wirst deine eigenen Ideen umsetzen können, ganz ohne vorgefertigte Bibliotheken.

Danksagung

Ein Dank geht an die Raspberry Pi Foundation für den offenen und genial konzipierten RP2040, an das **MicroPython-Team**, das uns diese großartige Plattform ermöglicht, und an alle Maker in der Community, die ihre Projekte und Ideen teilen.

Und ein besonderer Dank geht an dich, lieber Leser – dass du dich entschieden hast, nicht nur zu benutzen, sondern zu verstehen.

Viel Spaß beim Lernen, Experimentieren und Tüfteln – und willkommen in der Welt der PIO-Programmierung!

Hans-Joachim Seeger