1 Einleitung

»Kompliziertes kompliziert zu sagen ist einfach.
Nur Einfaches einfach zu sagen ist kompliziert.«

KARL-HEINZ KARIUS

Laut statista.com [57] wurden im Jahre 2021 weltweit 31,2 Milliarden Mikrocontroller produziert, was rund vier neuen Mikrocontrollern pro Erdenbürger entspricht. Damit sind diese Kleinstcomputer mittlerweile auch in Gegenständen des Alltags verbaut (»eingebettet«, *embedded*), in denen wir sie nicht vermuten. Oft werden Consumer-Produkte, Haushalts- und Kommunikationsgeräte, die Mikrocontroller enthalten, als »smart« bezeichnet. Vom üppig ausgestatteten Smartphone bis zur stark ressourcenbeschränkten Smart Card ist ein Mikroprozessor informationsverarbeitender Kern des Gerätes. All diese Geräte benötigen eine weitestgehend fehlerfreie Software, um mitunter ohne Update-Möglichkeit viele Jahre reibungslos zu funktionieren. Um dies zu gewährleisten, ist ein solides Grundverständnis von Aufbau und Arbeitsweise der Embedded Systeme unerlässlich.

Da die Komplexität durch wachsende Applikationsgröße und Internetanbindung steigt, wird auch die Programmentwicklung umfangreicher und komplexer. Diese IoT(»Internet of Things«)-Geräte besitzen embedded Betriebssysteme, deren Tasks miteinander kommunizieren. Auch dieses Zusammenspiel muss gut durchdacht sein, um performante Software ohne Deadlocks zu designen.

1.1 Ziel des Buchs

Ein Einstieg in die Entwicklung eingebetteter Systeme (in der Folge »Embedded Systeme« genannt) bringt verschiedene Hürden mit sich. Oft ist eine Programmiersprache bekannt, doch die technischen Details machen die Lernkurve steil und den Weg zum Ziel steinig.

Nimmt man ein Buch über die C-Programmiersprache zur Hand, enthält dieses keine Details zur Programmierung von Mikrocontrollern. Versucht man hingegen, die Datenblätter, Family Guides und Reference Manuals zu verwenden, machen die technischen Details den Einstieg schwer. Die Beispielprogramme und Application Notes auf den Webseiten der Hersteller implementieren Lösungen für sehr spezielle Probleme, was eine Umsetzung einer Lösung zu einem anders gearteten Problem schwierig gestaltet.

Insgesamt lässt sich sagen, dass eine Entwicklung oder Anpassung der Software ohne fundiertes Basiswissen aufwendiger und fehleranfälliger als mit den entsprechenden Grundlagen ist.

Ziel dieses Buches ist, der Leserin bzw. dem Leser die Grundlagen anschaulich und fundiert zu vermitteln. Aufgrund der Größe des Gebiets werden einige Bereiche nur gestreift, punktuell werden Themen aber in die Tiefe verfolgt.

1.2 Struktur des Buches

Die Struktur des Buches mag etwas ungewohnt erscheinen. Statt eines strukturierten, aufzählenden Aufbaus wurde ein beispielorientierter erzählerischer Ansatz gewählt. Anhand von Beispielen wird die embedded Welt durchwandert und erklärt.

Ein detaillierter Index am Ende des Buches dient dem Nachschlagen einzelner Themen. Der Inhalt ist drei Teilen zugeordnet, die schichtweise aufeinander aufbauen, aber jeweils für sich separat gelesen werden können, wie Abb. 1–1 zeigt.

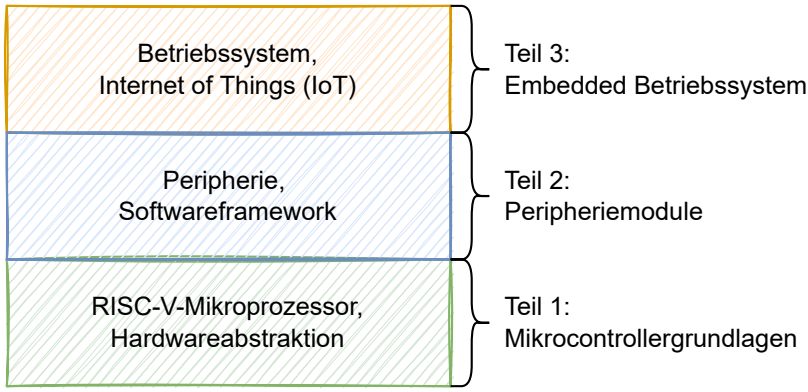

Abb. 1–1
Das Buch ist in drei Teile gegliedert.

Teil I behandelt den Aufbau eines RISC-V-Mikroprozessors, dessen Anbindung an ein Bussystem und die Grundlagen des Zugriffs auf

Peripheriemodule. Grundlagen der Assemblersprache und der hardwarenahen Programmierung in C mit Memory-Mapped I/O, Speicherverwaltung und Performanz werden vermittelt.

Teil II beinhaltet elektrotechnische Grundlagen zum Verständnis einfacher elektronischer Schaltungen. Verschiedene Peripheriemodule zur Ein-/Ausgabe, Kommunikation, Interrupt-Behandlung sowie die Verarbeitung analoger Sensordaten werden anhand der beispielhaften Implementierung eines Pulsoximeters erläutert.

Teil III bettet das Pulsoximeter in den Kontext des IoT ein, indem Daten über verschiedene Internetprotokolle verschickt werden. Die Grundlagen von embedded Betriebssystemen und deren Systemprogrammierung werden anhand des Beispiels verständlich. Eine praktische Betrachtung von Bluetooth LE und der Möglichkeiten des Stromsparens rundet das Kapitel ab.

1.3 Zielpublikum

In erster Linie ist dieses Buch für den Einsatz im einführenden und fortgeschrittenen Unterricht über Embedded Systeme geplant. Es ist von großem Vorteil für das Verständnis, wenn Grundlagen der Informatik und des Programmierens in der Programmiersprache C vorhanden sind. Alternativ sind Grundlagen in einer anderen Programmiersprache sehr anzuraten. Die einzelnen Teile bieten sich an, in separaten Lehrveranstaltungen zu Rechnerarchitektur/-organisation (Teil I), Embedded Programmierung (Teil II), Betriebssystemen (Teil III) und Kommunikationssystemen/IoT (Teil III) Eingang zu finden.

In zweiter Linie richtet sich das Buch an interessierte Leser:innen mit informatischem Background. Entsprechendes Interesse vorausgesetzt profitiert diese Leserschaft vom Inhalt. Die Lesereihenfolge ist beliebig, idealerweise sequenziell vom Anfang zum Ende.

Auch Personen, die bereits im Embedded Systems-Umfeld aktiv sind, sind von diesem Buch angesprochen. Die technischen Details zu Architektur und Programmierung, die hier zusammengetragen sind, vertiefen das vorhandene Wissen. Dieser Leserschaft ist angeraten, das Buch von vorne nach hinten zu lesen und bekannte Teile zu überfliegen. Beim Überspringen könnten wertvolle Details ausgelassen werden.

Für alle Leser:innen dient das Buch als Nachschlagewerk zu den vielfältigen Technologien, die in Embedded Systemen zum Einsatz kommen.

1.4 Gebrauchsanweisung

Es ist möglich, das Buch nur zu lesen. Um ideal zu profitieren, ist anzuraten, die entsprechende embedded Hardware anzuschaffen und die Beispiele im Buch nachzuvollziehen. In diesem Sinne handelt es sich nicht um ein Lese-, sondern ein Arbeitsbuch.

Embedded Hardware Als embedded Hardware findet ein ESP32-C3-Mikrocontroller (siehe Abschnitt 2.2.1), basierend auf einem modernen RISC-V-Prozessor, Anwendung. Die Beispiele erfordern teilweise zusätzliche Komponenten wie eine Steckplatine (siehe Abschnitt 5.3.4) und Bauteile, die auch mit einem kleinen Budget zu beschaffen sind. Quellen zur Materialbeschaffung finden Sie auf der Webseite zum Buch (siehe Anhang A).

Beispielprogramme Die Herausforderungen, die sich bei den ersten Implementierungen im embedded Umfeld ergeben, folgen typischerweise bestimmten Mustern, die sich hauptsächlich aus der Interaktion des Systems mit seiner Umgebung ergeben. Solche Muster werden im Buch zum besseren Verständnis durchgehend in Beispielprogrammen verwendet. Um diese besser nachzuvollziehen, ist der Code der Beispiele online zugänglich (siehe Anhang A) abgelegt. Dies ist vor allem beim großen Pulsoximeterbeispiel wichtig, da im Buch wesentliche Teile, aber nicht die gesamten Sourcen abgedruckt sind.

Übungsbeispiele Jeder Teil verfügt über theoretische und praktische Übungen. Diese dienen dem Sammeln von Erfahrungen mit der embedded Plattform und dem Üben anhand typischer Problemstellungen. Ein Vergleich mit den bereitgestellten Lösungen hilft bei der Beurteilung der eigenen Ausarbeitung.

Wichtig ist dabei zu beachten, dass eine Musterlösung nicht die einzig sinnvolle Lösung darstellt. Es gibt immer viele Wege zum Ziel, die jeweils ihre eigene Begründung haben. Deshalb werden auf der Webseite zum Buch (siehe A) Kommentare zu den Musterlösungen und alternative Ansätze gesammelt und bereitgestellt.

1.4.1 Konventionen

Im Bereich der Informatik werden oft Fachbegriffe verwendet, die eine sperrige oder ungewohnte deutsche Übersetzung haben. Aus diesem Grund werden die Begriffe bei der ersten Verwendung in »französischen« Anführungszeichen eingeführt und dann direkt verwendet. Eine Vermischung wie »Embedded Systeme« statt »eingebettete

Systeme« oder »embedded systems« ist eine unweigerliche Folge, die dem Autor bitte verziehen wird.

Die abgedruckten Sourcen in der Programmiersprache C wurden der Übersichtlichkeit halber auf den Platzbedarf hin optimiert. Umbrüche langer Zeilen werden mit ↪ dargestellt. Der Programmierstil ist modern und teils ungewöhnlich. So wird ein `int *pX` als `int* pX` dargestellt, um zu zeigen, dass der Pointer zum Typ gehört und nicht zum Namen. Die Benennung von Variablen und Funktionen ist modern in Camel-Case. Wer das unangebracht findet, möge bitte über diese Spitzfindigkeiten hinwegsehen und die eigenen Konventionen weiter verwenden.

Werden im Fließtext Variablen verwendet, werden diese in einer `Terminal-Schriftart` dargestellt. Zur einfachen Darstellung, dass es sich um eine Funktion handelt, wird diese mit runden Klammen, aber ohne Parameterliste, angegeben, beispielsweise `doIt()`.

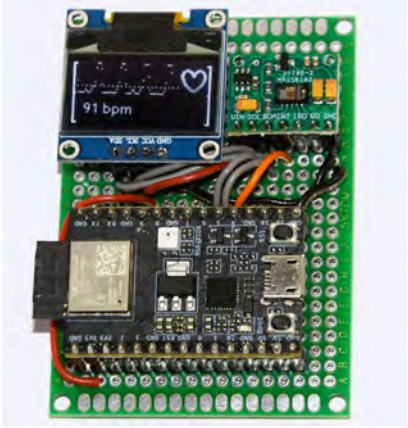

Abb. 1–2
Die Implementierung des Pulsoximeters `poxi` *bildet die Basis der Teile II und III.*

Weiterführendes

Einführende oder weiterführende Themen werden in separaten Kästen untergebracht. Bei Interesse können sie gelesen werden; sie sind allerdings nicht im Fließtext eingebettet.

Die Größenverhältnisse von Kleinsystemen bieten sich für eine kurze Betrachtung an. Die kleinsten Mikrocontroller sind mit Gehäuse kleiner als 2 mm x 2 mm x 0,5 mm. Der in diesem Buch verwendete ESP32-C3-Mikrocontroller hat in etwa die Leistungsfähigkeit eines Intel-Pentium-PC-Prozessors und passt mit seiner Größe 5 mm x 5 mm x 0,85 mm etwa 6½ Mal auf dessen Siliziumfläche - inklusive RAM, Wi-Fi und Bluetooth-Hardware.

Die weiteren Themen sind nicht minder spannend!

2 Hallo, Welt!

»Das Durchschnittliche gibt der Welt ihren Bestand,
das Außergewöhnliche ihren Wert.«

OSCAR WILDE

Seit Kernighan und Ritchie in ihrem Buch *The C Programming Language* [36] gleich zu Beginn ein Programm schrieben, das

```
hello, world
```

auf dem Bildschirm ausgab, verwenden viele Programmierbücher diesen Ansatz. Die Begründung, »der einzige Weg, eine Programmiersprache zu lernen, ist, Programme in ihr zu schreiben«, ist ja durchaus plausibel.
Da dieses Buch unter anderem den Anspruch erhebt, die Programmierung von Embedded Systemen als Fertigkeit zu erlernen, wird dieser beispielgetriebene Ansatz auch hier verfolgt. In diesem Kapitel wird ein erstes C-Programm erstellt, auf eine embedded Plattform mit einem RISC-V-basierten Mikrocontroller aufgespielt und dort gestartet. Der Debugger dient dann zur schrittweisen Programmausführung.

Auf diese Weise werden die einzelnen Komponenten des Entwicklungsflusses verständlich. Begleitend zu dieser praktischen »Implementierung« werden die Entwicklungsumgebung und das »Embedded System«, nämlich der Rechner, auf dem das Programm ausgeführt wird, erläutert. Dieses Embedded System unterscheidet sich doch stark von einem klassischen PC mit Monitor, Tastatur, Maus und grafischer Benutzeroberfläche.

PC, Personal Computer: Heimcomputer oder Arbeitsplatzrechner wie Desktop, Notebook oder Tablet

Embedded System

Unter einem *embedded system* (zu Deutsch: *eingebettetes System*) versteht man ein Computersystem, bestehend aus Hard- und Software, das in einen technischen Kontext eingebettet ist. In diesem Kontext verrichtet es Arbeiten wie Überwachung, Steuerung, Regelung und die weitere Datenverarbeitung. In modernen Systemen nimmt die Kommunikation eine wachsende Rolle ein, was sich in den technischen Modeschlagworten IoT und IIoT (*[Industrial] Internet of Things*: Sensoren und andere Geräte, die [im industriellen Umfeld] vernetzt sind) niederschlägt.

Embedded Systeme treten in ihren Applikationen oft so weit in den Hintergrund, dass sie für Anwender unsichtbar sind oder nicht mehr als Computer wahrgenommen werden. Beispiele sind moderne Haushaltsmaschinen, Unterhaltungsgeräte wie Uhren etc., aber auch Geräte der Kommunikationsinfrastruktur, Industrie und Fahrzeuge vom Automotive-Bereich bis zur Raumfahrt. Aufgrund dieser Unsichtbarkeit, Durchdringung und Allgegenwärtigkeit von Computersystemen stößt man im Umfeld auf die Begriffe *invisible*, *pervasive* und *ubiquitous* Computing.

Da solche Systeme oft mobil sind, keinen Anschluss an das Stromnetz haben, auch extremen Umweltbedingungen ausgesetzt sind und technisch in großen Stückzahlen produziert werden, liegt der Fokus auf kleinen, stromsparenden, robusten und günstigen Komponenten. Dadurch nicht vergleichbar mit üppig ausgestatteten PCs sprechen wir von *ressourcenbeschränkten Systemen*. Diese Beschränkung von Ressourcen wie Arbeitsspeicher, Akkukapazität, Bandbreite und Latenz der Kommunikation, Ein-/Ausgabemöglichkeiten, Antwortzeit und Echtzeitfähigkeit, Kosten etc. wirkt sich direkt auf die gesamte Hard- und Softwareentwicklung aus.

Die Hardware besteht neben Gehäuse und mechanischen Komponenten aus einer Elektronik, die diverse Schnittstellen bereitstellt. Neben LEDs, Displays, Tastern, Joysticks, Segmentanzeigen, Leistungselektronik, Sensoren für Temperatur, Druck, Helligkeit, Beschleunigung und vielem mehr enthält die Hardware als steuernde Komponente einen Mikroprozessor bzw. Mikrocontroller.

2.1 Wahl der Programmiersprache

Für die Programmierung von Embedded Systemen werden verschiedene Programmiersprachen angepriesen und in der Praxis auch verwendet. Sowohl von Skriptsprachen als auch von Sprachen mit einer virtuellen Maschine wird in diesem Buch aus mehreren Gründen Abstand genommen:

Ein wesentliches Ziel dieses Buches ist es, ein grundlagenbasiertes Verständnis des Systems zu vermitteln, weshalb auf die gesamte