1.	Record Nr.	UNINA9910886988003321
	Autore	Eliasz Andrew
	Titolo	Zephyr RTOS Embedded C Programming : Using Embedded RTOS POSIX API / / by Andrew Eliasz
	Pubbl/distr/stampa	Berkeley, CA : , : Apress : , : Imprint : Apress, , 2024
	ISBN	9798868801075
	Edizione	[1st ed. 2024.]
	Descrizione fisica	1 online resource (689 pages)
	Disciplina	005.13
	Soggetti	Programming languages (Electronic computers) Internet of things Programming Language Internet of Things
	Lingua di pubblicazione	Inglese
	Formato	Materiale a stampa
	Livello bibliografico	Monografia
	Nota di contenuto	1. Introduction -- 2. A Review of RTOS Fundamentals -- 3. Zephyr RTOS Application Development Environments and Zephyr Application Building Principles -- 4. Zephyr RTOS Multithreading -- 5. Message Queues, Pipes, Mailboxes and Workqueues -- 6. Using Filesystems in Zephyr Applications -- 7. Developing Zephyr BLE Applications -- 8. Zephyr RTOS and Ethernet, WiFi, and TCP/IP -- 9. Understanding and Working with the Device Tree, in general, and SPI and I2C in particular -- 10. Building Zephyr RTOS Applications Using Renode -- 11. Understanding and Using the Zephyr ZBus in Application Development -- 12. Zephyr Wi-Fi.
	Sommario/riassunto	These days the term Real-Time Operating System (RTOS) is used when referring to an operating system designed for use in embedded microprocessors or controllers. The “Real Time” part refers to the ability to implement applications that can rapidly responding to external events in a deterministic and predictable manner. RTOS-based applications have to meet strict deadline constraints while meeting the requirements of the application. One way of ensuring that urgent operations are handled reliably is to set task priorities on each task and to assign higher priorities to those tasks that need to respond in a more timely manner. Another feature of real-time applications is the

careful design and implementation of the communication and synchronization between the various tasks. The Zephyr RTOS was developed by Wind River Systems, and subsequently open sourced. Its design and implementation are oriented towards the development of time critical IoT (Internet of Things) and IIoT (Industrial Internet of Things) applications, and, consequently it has a rich feature set for building both wireless and wired networking applications. However, with a rich feature set comes a fairly steep learning curve. This book covers the foundations of programming embedded systems applications using Zephyr's Kernel services. After introducing the Zephyr architecture as well as the Zephyr build and configuration processes, the book will focus on multi-tasking and inter-process communication using the Zephyr Kernel Services API. By analogy with embedded Linux programming books, this book will be akin a Linux course that focuses on application development using the Posix API. In this case, however, it will be the Zephyr Kernel Services API that will be the API being used as well as the Posix API features supported by Zephyr. What You'll learn An Overview of the Cortex-M Architecture. Advanced data structures and algorithms programming (linked lists, circular buffers and lists). How to build Zephyr Applications, including setting up a Command Line Zephyr Development Environment on Linux. Task scheduling and pre-emption patterns used in Real Time Operating Systems. Scheduling, Interrupts and Synchronization, including threads, scheduling, and system threads. Overview of Symmetric Multiprocessing (SMP) and Zephyr support for SMP. Memory management, including memory heaps, memory slabs, and memory pools. Who This Book Is For Embedded Systems programmers, IoT and IIoT developers, researchers, BLE application developers (Industrial Control Systems, Smart Sensors, Medical Devices, Smart Watches, Manufacturing, Robotics). Also of use to undergraduate and masters in computer science and digital electronics courses.