FRANKFURT UNIVERSITY OF APPLIED SCIENCES

PROJECT

Introduction

Prof. Dr. Oliver Hahm

2024-10-21

AGENDA

- About
- Organizational
- Internet of Things
- Software for low-end IoT Devices
- Technical Insights on RIOT
- RIOT Community

ABOUT

ABOUT

PROF. DR. OLIVER HAHM

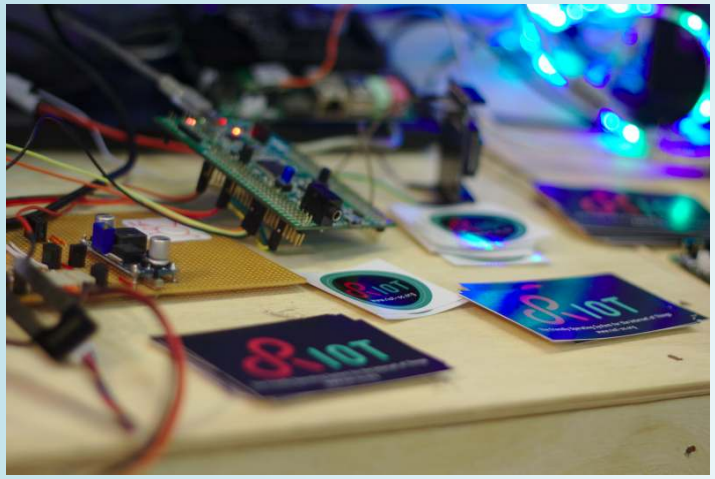
FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- Study of Computer Science at Freie Universität Berlin
- Software Developer for ScatterWeb and Zühlke Engineering
- Research on IoT and Operating Systems

Contact

E-mail: oliver.hahm@fb2.fra-uas.de
Appointments: via e-mail, room 1-212

JOIN THE RIOT!



RIOT is the friendly operating system for the IoT!

You're interested in ...

- ...programming the IoT?
- ...collaborate with hundreds of people from all over the world?
- ...contribute to a big FLOSS project?

Get in touch and do some hacking at the *All RIOT* event at the university! Usually every second Wednesday at 2pm in room 1-237.
First meeting: November 06, 2024.
All information on https://allriot.dahahm.de

WHAT ABOUT YOU?

- What is your motivation for this course?
- What do you think about the Internet of Things?

ORGANIZATIONAL

WHY?

- Let’s pretend you are an IT service provider
- I am your customer
- You have to…
  - collect the requirements,
  - survey the solution space,
  - propose a viable system architecture,
  - implement a walking skeleton,
  - develop an MVP[1],
  - document your project, and
  - work as a team.

Think about the software development model you want to use!

1. Minimum Viable Product

WHAT?

- Develop an application for a constrained IoT device
- Enable remote access to the application via an Instant Messenger (WhatsApp, Telegram, Signal…)
- Remote access should allow for…
  - Reading sensor values
  - Change settings
- Remote access requires…
  - *IPv6* connectivity
  - a *border router*
  - potentially an Internet service as a gateway

HOW?

- Team work (two students per group)
  - ↪ But grading is individually
- Each team work on a common code base
- git is used as version control system
- Develop the software
  - Create a firmware based on RIOT (https://riot-os.org)
  - To run additional services you can use AWS (https://www.awsacademy.com)
    ⇒ Please send me an email to get an invitation
- Write documentation about your project
- Run (and evaluate) your code on RIOT native and on real hardware
- Present your work

WHEN?

- November 25, 2024: Submission and presentation of your architecture
- January 20, 2025: Present your walking skeleton (incl. demo)
- February 10, 2025: Presentation
  - Give a short presentation on your work (live demo?)
- February 21, 2025: Submission
  - Final version of the code is in the repository
  - You have granted access to me
  - Send me your documentation

REQUIRED PRIOR KNOWLEDGE

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- For successful participation the knowledge from multiple courses is required, e.g., ...
  - Software Engineering
  - Computer Networks
  - Operating Systems
  - Embedded/Real-Time Systems
  - Distributed Systems

EVALUATION

Which aspects of your work are going to be evaluated?

- Your *implementation* (50%)
  - Functionality (20%)
  - Creativity (10%)
  - System and code architecture (10%)
  - Code quality (5%)
  - Infrastructure (5%)
- The *documentation* (25%)
  - Inline code documentation (5%)
  - Final How-To (20%)
- Your *presentations* (25%)
  - Architecture presentation (5%)
  - Walking Skeleton presentation (10%)
  - The final presentation (10%)

GRADING SYSTEM

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Definition of the Grades

- 1.0
  An excellence performance. It is awarded if the work evaluated is outstanding, flawless and near perfection. It exceeds the expectations and is particularly witty.
- 2.0
  A good performance. The work evaluated meets the expectations and fulfills the requirements well. It may contain some minor or formal errors.
- 3.0
  A satisfying performance. The work evaluated meets most of the expectations and fulfills the basic requirements. It contains some clear errors that should be corrected.
- 4.0
  A sufficient performance to pass the examination. The work evaluated fulfills the bare minimum but significantly more. It contains several clear errors that must be corrected.
- 5.0
  An insufficient performance. The work evaluated does not even fulfill the basic requirements and is not enough to pass the exam. It may also be awarded in case of cheating or plagiarism.

HOW TO GET AN 1.0?

- Basic functional requirements are fulfilled
- One of the following additional requirements are addressed
  - enhanced security, e.g.,...
    - end-to-end encryption
    - link layer security
    - authentication
  - additional features, e.g., support...
    - multiple IMs
    - multiple radio technologies
    - IPv4 and IPv6
    - additional sensors or actuators
  - improved efficiency, e.g.,...
    - reduced energy consumption
    - reduced memory footprint
    - no or only minimal cloud services required

RECOMMENDED PROCEDURE

- Prepare your workstation
- Research on tools and protocols
- Familiarize yourself w/ RIOT and AWS
- Design your architecture
- Build a firmware w/ basic connectivity
  - Use shell commands for interaction
- Implement a walking skeleton
- Refine and extend
  - Automate setup

Continuously document everything!

FURTHER INFORMATION

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

All material regarding this course can be found at
https://teaching.dahahm.de

This includes

- Announcements
- Slides
- Dates

INTERNET OF THINGS

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Three Disruptive Technologies as the Roots of the IoT

Wireless Communication

Low-cost Embedded Systems

The Internet

SMART OBJECT NETWORKING AT INTERNET-SCALE

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Connecting the Physical World with the Internet

- Transforming Things into Smart Objects
- Enabling Interconnected Smart Services

Building & Home Automation
Industrial Automation
Mobile Health
Micro & Nano Satellites

USE CASE REQUIREMENTS

- Interoperability
- Energy Efficiency
- Reliability
- Latency
- Low Cost Factor
- Autonomy
- Security
- Scalability

It ain’t smart if I have to charge it every day.

SOFTWARE FOR LOW-END IOT DEVICES

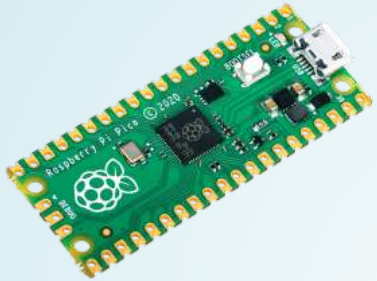
CONSTRAINTS AND REQUIREMENTS

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Low-end IoT Devices: Limited Resources (RFC7228)

Arduino Due

Raspberry Pi Pico



nRF52840 Dongle

- Memory < 1 Mb
- CPU < 100 MHz
- Energy < 10 Wh

+ Use Case Requirements

Software Requirements

- Energy Efficiency
- Sustainability
- Network Connectivity
- Real-time Capabilities
- Small Memory FootprintLow Cost Factor
- Security and Safety
- Support for Heterogeneous Hardware

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

No user interaction

- No GUI required ⇒ No Pseudo-Parallel Execution is required
- Must Operate Autonomously → Must Recover from Errors
- Autoconfiguration is required

Constrained Hardware

- Often no MMU[1] and no FPU[2]
- Typically no Display or Input Devices
- In many cases no Persistent Memory

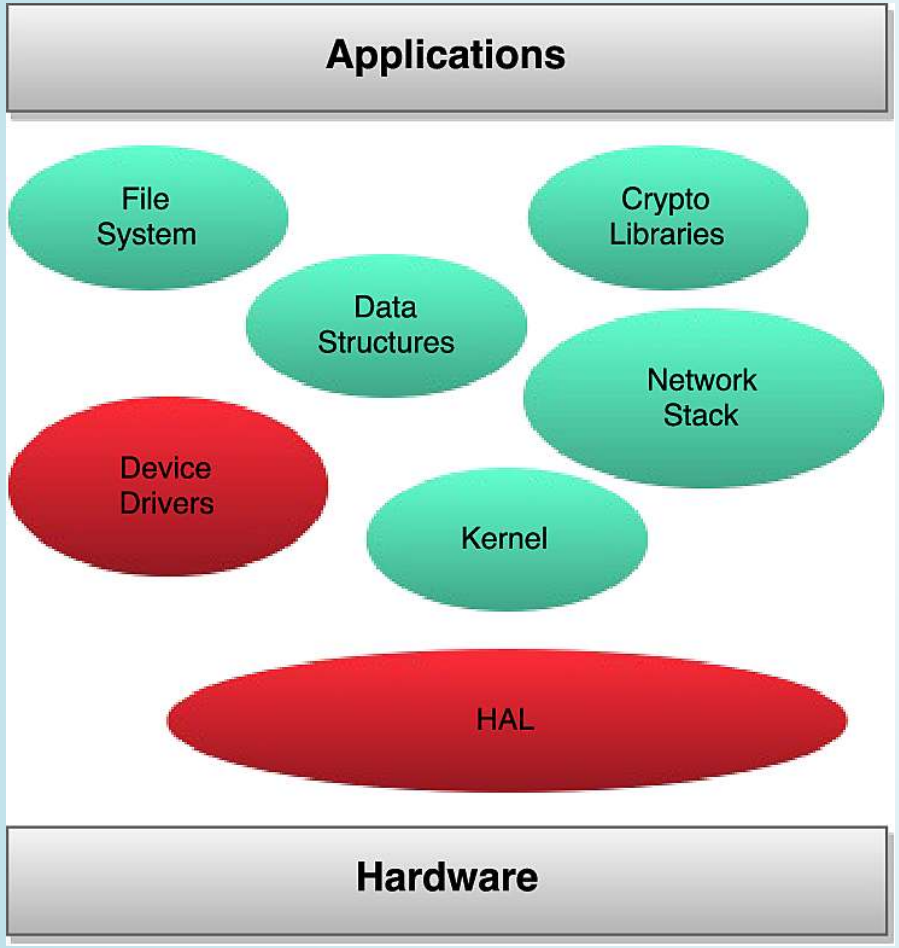
- No Multi-User Support required
- Often only one Application
- Typically no dynamic linking → just one statically linked binary

1. Memory Management Unit
2. Floating Point Unit

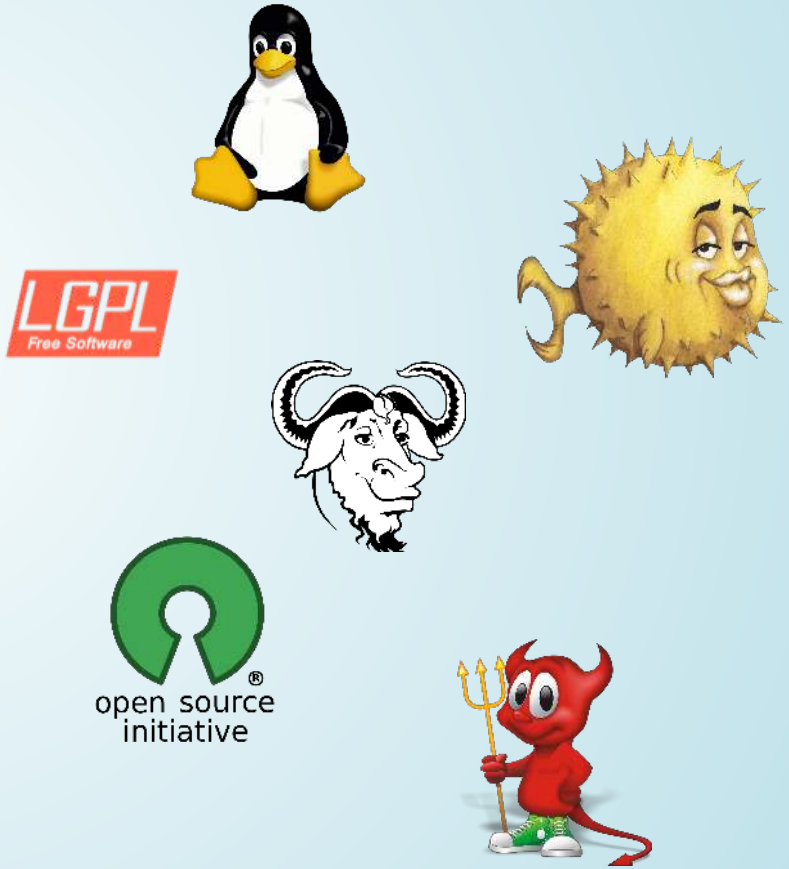
AN OS FOR LOW-END IOT DEVICES

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Unified Software Platform



Open Source



OPERATING SYSTEMS FOR LOW-END IOT DEVICES

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Full-fledged OS

WSN OS

RTOS

Does not fit

- Too Big
- Requires a MMU
- Not Targeted for Real-Time or Low-Energy

Too complicated

- Hard to Learn
- No System Level Compatibility

Too Minimalistic

- No Built in Networking Support
- No Common API

TECHNICAL INSIGHTS ON RIOT

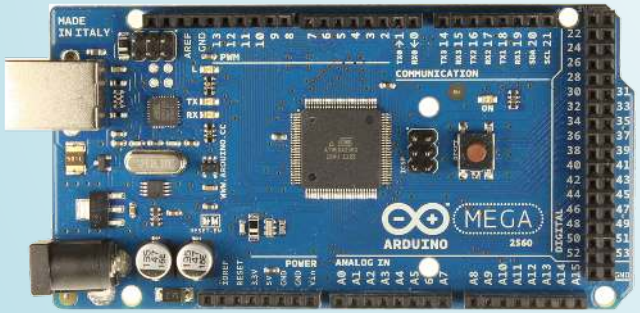
RIOT FACTS

THE FRIENDLY OS FOR THE IOT

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

"If your IoT device cannot run Linux, then use RIOT!"

- RIOT requires only a few kB of RAM/ROM, and a small CPU
- With RIOT, code once & run heterogeneous IoT hardware
-
  - 8bit hardware (e.g. Arduino)
  - 16bit hardware (e.g. MSP430)
  - 32bit hardware (e.g. ARM Cortex-M, x86)



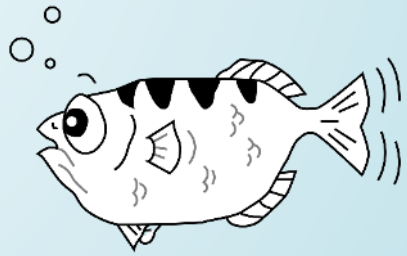
OPEN STANDARDS, OPEN SOURCE

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- Free, open source (LGPLv2.1) operating system for constrained IoT devices
- Write your code in ANSI-C or C++
- Compliant with the most widely used POSIX features like pthreads and sockets
- No IoT hardware needed for development
- Run & debug RIOT as native process in Linux

WIRESHARK

Valgrind



GDB - The GNU Debugger

PROGRAMMING LANGUAGE AND GUIDELINES

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Important Programming Language Properties

- No Overhead
- Full Control over Memory Management
- Direct Access to the Hardware
- Binding to other Languages
- Usability

Why C?

- Ticks all the Boxes
- Stable Specification
- Widely Used $\rightarrow$ Tooling

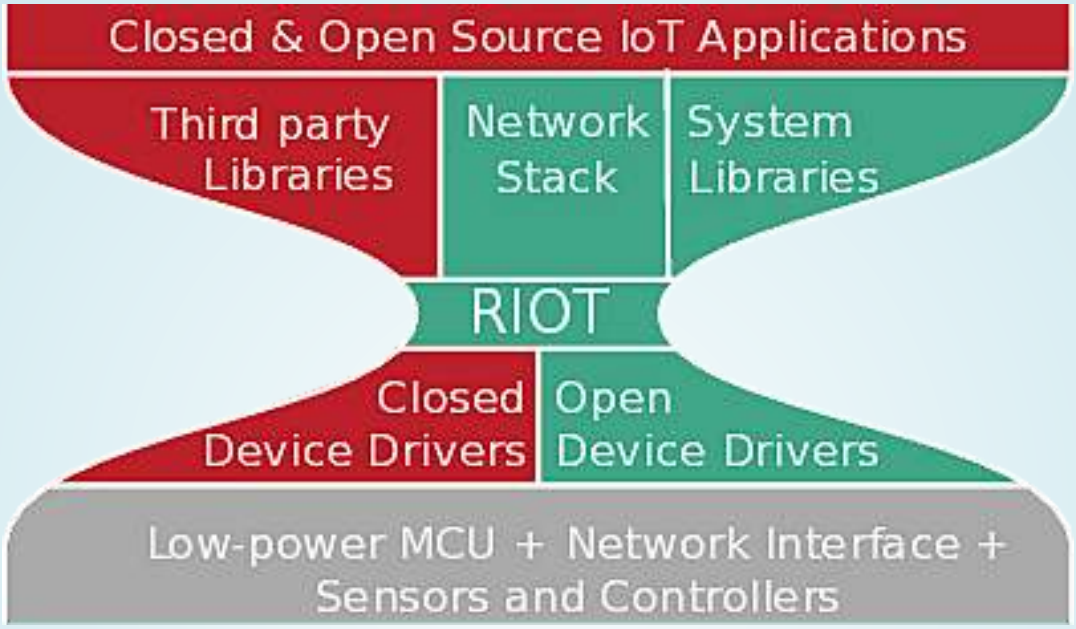
```
140 void thread_yield(void)
141 {
142     unsigned old_state = irq_disable();
143     thread_t *me = thread_get_active();
144
145     if (me->status >= STATUS_ON_RUNQUEUE) {
146         sched_runq_advance(me->priority);
147     }
148     irq_restore(old_state);
149
150     thread_yield_higher();
151 }
```

Programming Guidelines

- Follow a Structured and Procedural Approach
- Keep It Simple, Stupid (KISS)
- No Dynamic Memory Allocation
- Be Resource-aware
- No Macro “Magic”

RIOT ARCHITECTURE

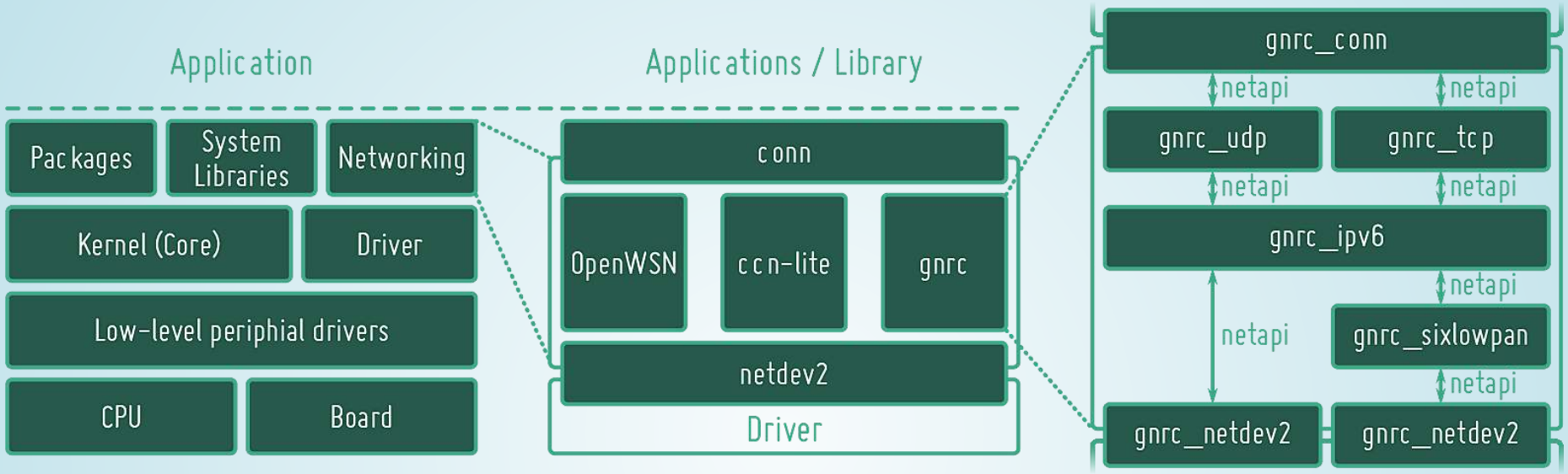
ARCHITECTURAL OVERVIEW



Design Decisions

- Efficient & Flexible Micro-Kernel
- System Level Interoperability
- Networking Interoperability

THE STRUCTURE



CONCEPTS

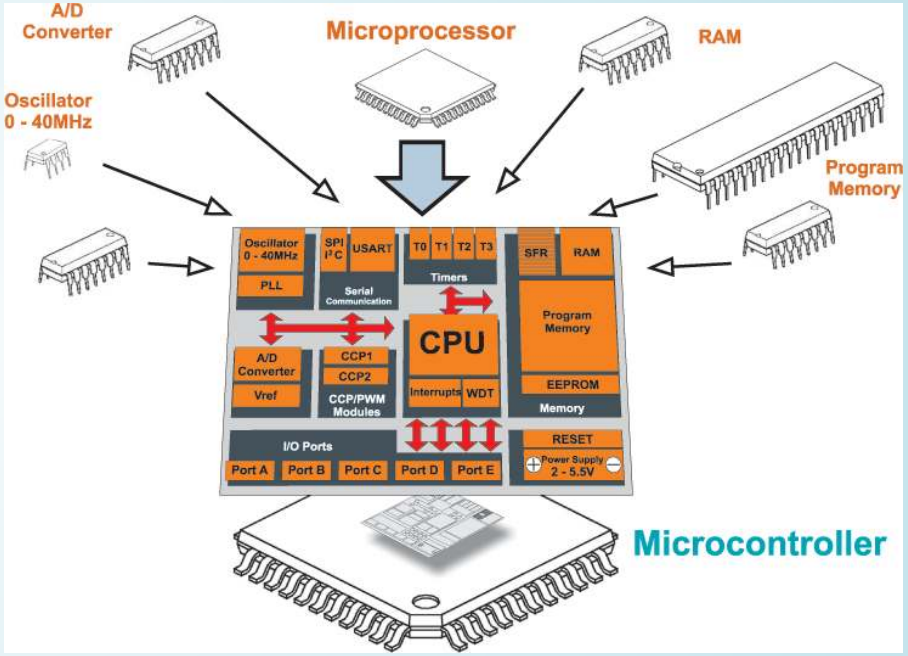
HARDWARE ABSTRACTION LAYER (HAL)

Challenge: Support a Plethora of different Platforms

- Different Processor Architectures (8 bit, 16 bit, 32 bit ...)
- MicrocontrollerPeripherals
- Sensors and Actuators
- Network Devices
- Crypto Devices
- ...

Goal: Provide a Common API

- Drivers for MCU Core
- Drivers for MCU Peripherals
- Device Drivers
- Timer API



Source: MikroElektronika, https://www.mikroe.com

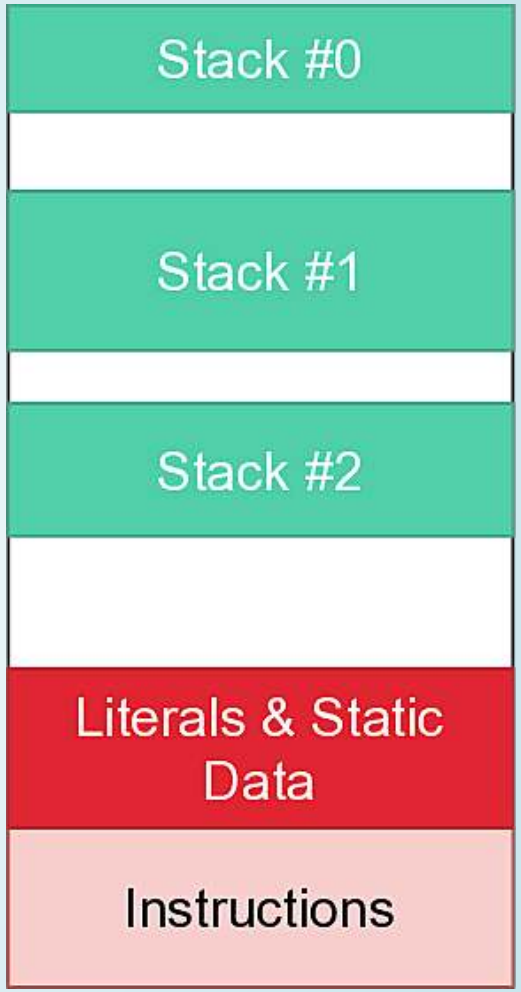
MULTI-THREADING

- Microkernel approach
  → But no Memory Protection
  ⇒ Stack Overflows are possible
- Provides Standard Multi-Threading
- Each Thread contains a (minimal) Thread Control Block (TCB)

Low Memory Usage

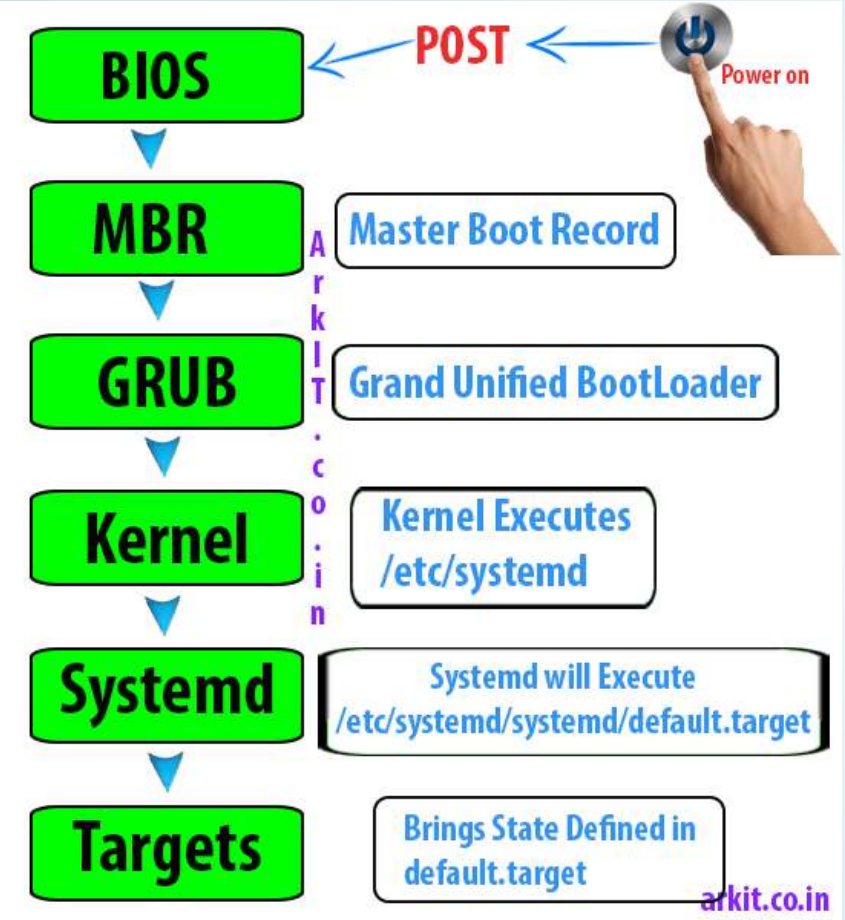
On a Low-end IoT Device (16-bit, 8 MHz):

- Min. TCB: 8 bytes
- Min. Stack Size: 96 bytes
- Up to 16,000 Messages/s
  (≙ 10,000 Packets/s for 802.15.4)



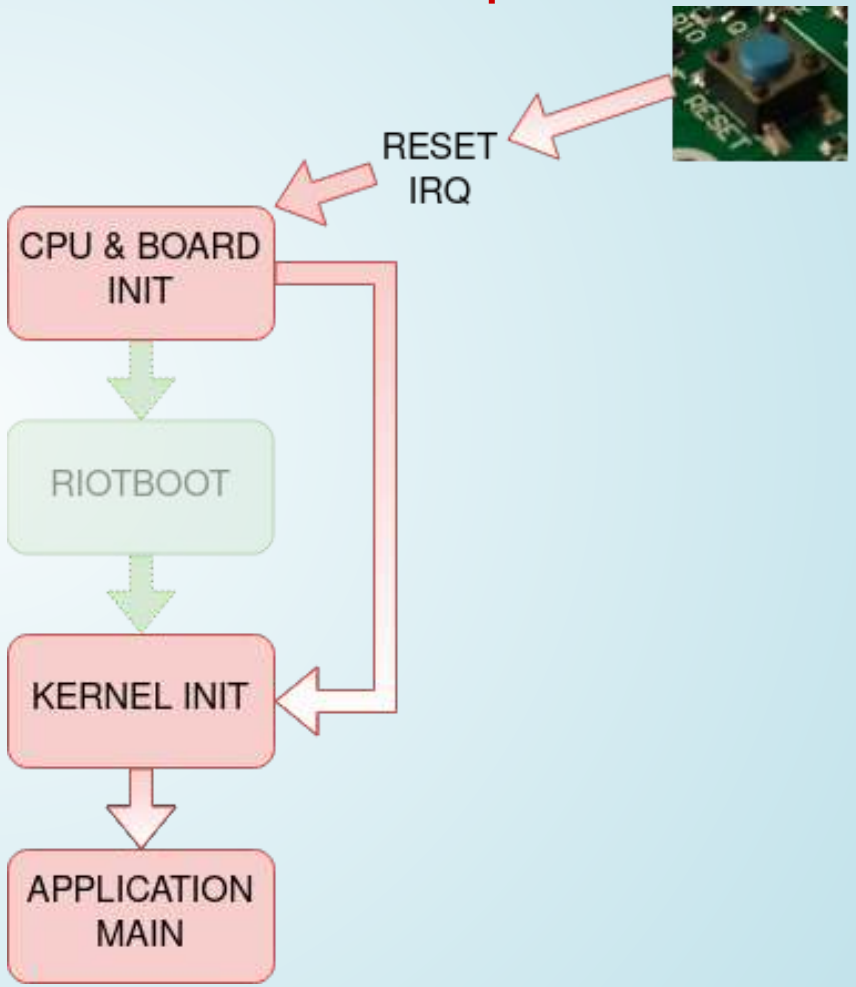
BOOT SEQUENCE

Linux Boot Sequence



Source: https://arkit.co.in/linux-boot-process-millionaire-guide/

RIOT Boot Sequence



SCHEDULING

- Preemptive
- Threads have fixed Priorities
- The Thread in the Run-Queue with the highest Priority will run

A Periodic System Tick requires Timers

- A running Timer prevent the MCU to enter Deep Sleep Modes
- Periodic Wakeup waste Energy if there is nothing to do

Accounting for Real-Time Requirements

- All Data Structures in the Kernel have Static Size ⇒ All Operations are O(1)
- The Behavior of the Kernel is completely deterministic
- Interrupt Handlers are a short as possible

Source: Educación Física, https://efsancristobalcartagena.blogspot.com

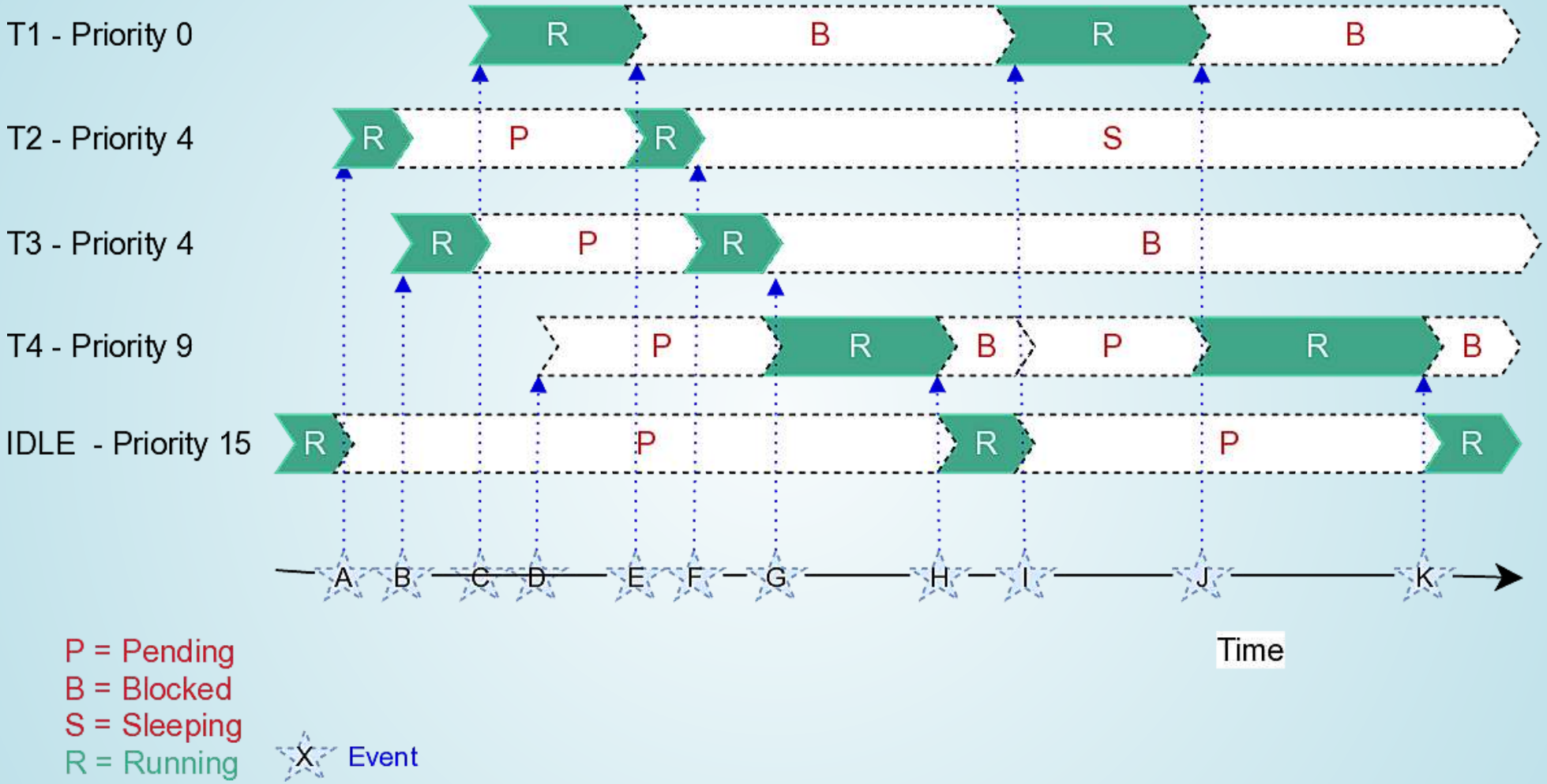
THREAD STATES

- A Thread can have one of the following States:
  - Stopped
  - Sleeping
  - Blocked
  - Running
  - Pending
- The States `Running` and `Pending` indicate that the Thread is on the Run-Queue
  ⇒ The Thread is ready to run

It may be blocked waiting for …

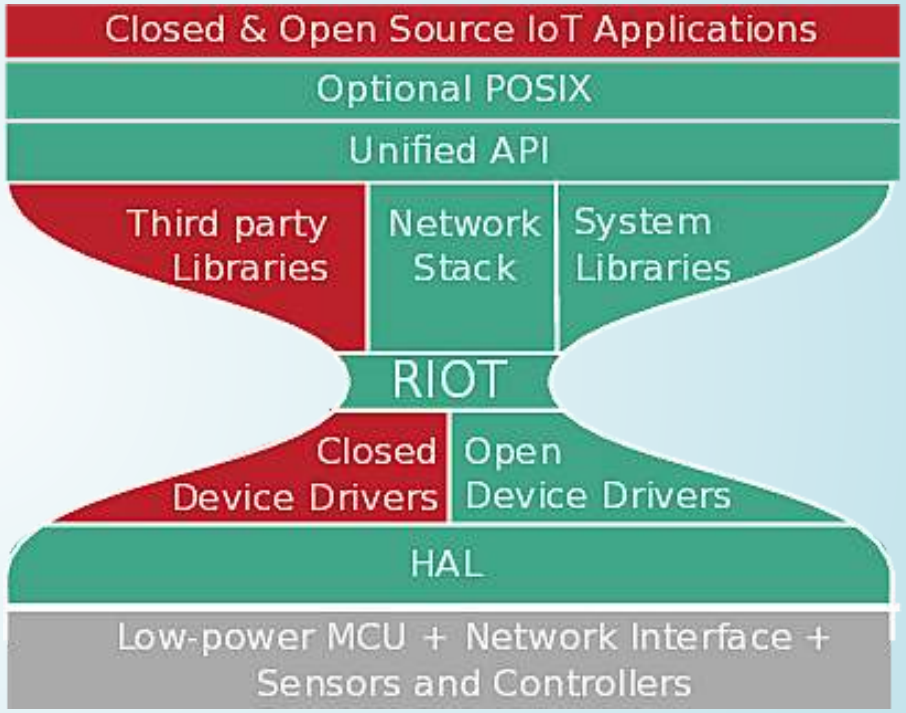
- a mutex
- a message to be received
- a message to be sent
- a response to a previous message
- a thread flag
- an action in its mailbox
- a condition variable

SCHEDULING EXAMPLE



APPLICATION PROGRAMMING INTERFACE (API)

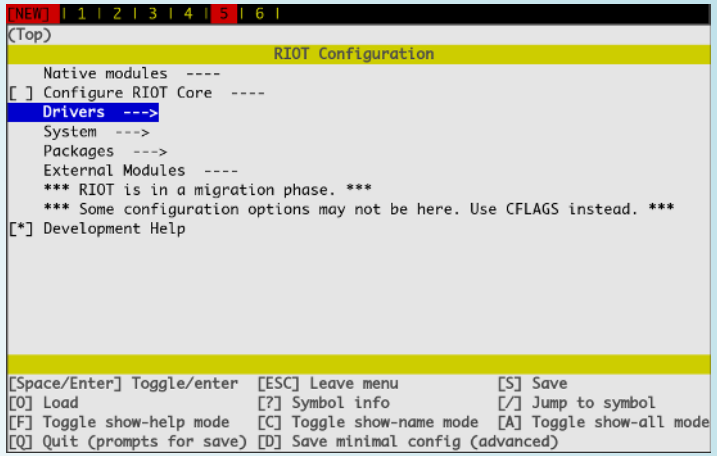
- Application shall be independent from the Hardware
- Portable Operating System Interface (POSIX) provides a common API among OS
- Not well suited for low-power IoT Devices
  - Origins from the 1980's
    $\longrightarrow$ Not very modern
  - Not tailored for constrained Resources
  - $\rightarrow$ But facilitates (initial) porting
- A POSIX-like API for this Class of Devices is missing so far



MODULARITY AND REUSABILITY

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- Specialized Applications require only a Subset of the available Features
- Fine-grained Modularity is required to reduce the Binary Size
- Kernel Features may be disabled (→ Even Multi-Threading is optional)

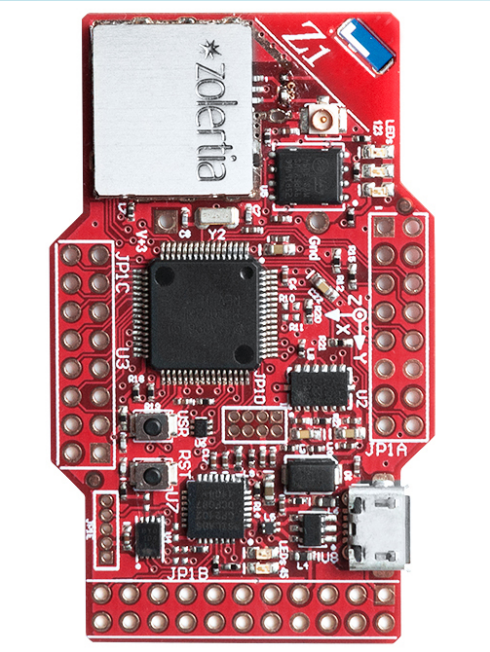


Result: Low Porting Effort

- Emulation support: RIOT as a Process
- Third-Party Development Tools
- Third-Party Library Packages

	Diff Size	
Package	**Overall**	**Relative**
libcoap	639 lines	6.3 %
libfixmath	34 lines	0.2 %
lwip	767 lines	1.3 %
micro-ecc	14 lines	0.8 %
relic	24 lines	<0.1 %

MEMORY COMPARISON



RIOT is as small as traditional WSN Operating Systems

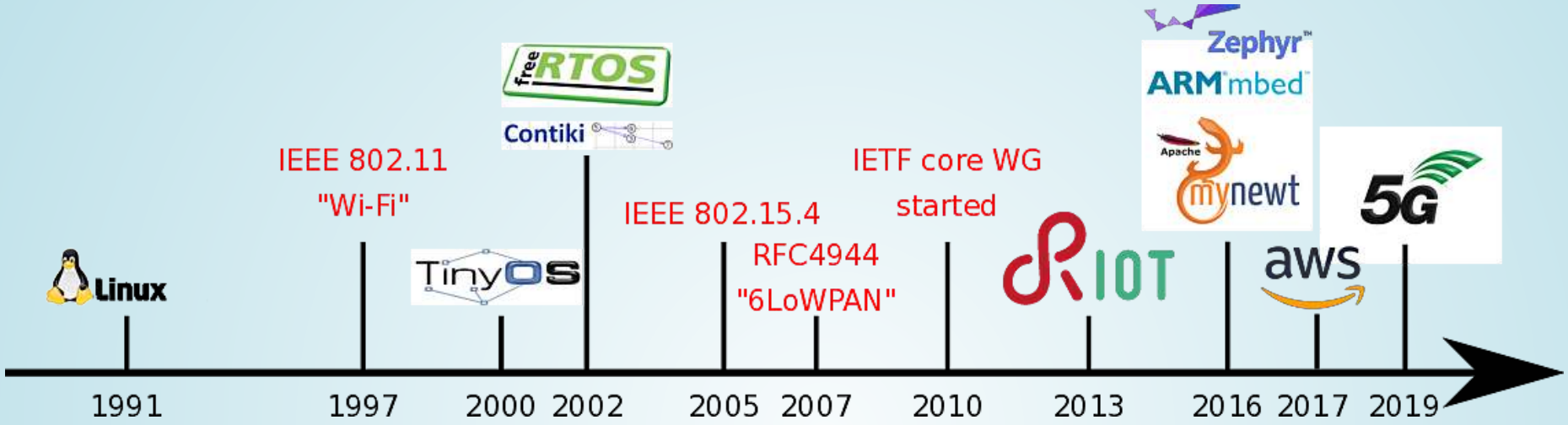
Application	ROM	RAM
RIOT 2024.07	42,341	6,010[1]
Contiki 3.0	51,562	5,530
TinyOS tinyos-main	40,574	6,812

Standard IoT IPv6 Networking Application

Code size comparison [Bytes] between RIOT, Contiki, and TinyOS.

1. Can be reduced for smaller MTUs.

REVIEW & PERSPECTIVES



IoT Software in 2024

- Most popular IoT OS are:
  - RIOT
  - Zephyr
  - AWS FreeRTOS
- RIOT as the Linux for the IoT?
- ongoing challenges: Cloud integration, security, software updates

RIOT COMMUNITY

10 YEARS OF RIOT!

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

RIOT Open Source Development

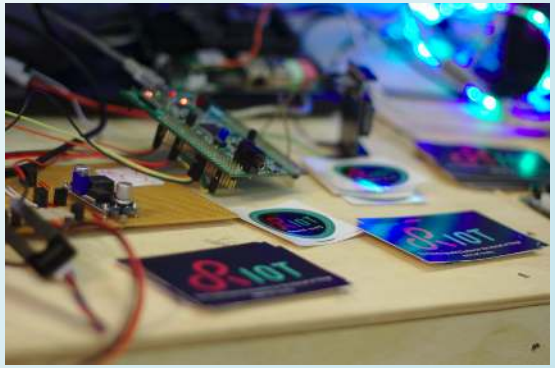
- More Than 43,000 Commits and More Than 16,000 Pull Requests
- Over 1,900 forks on GitHub
- More Than 330 Contributors
- Support for More Than 250 Hardware Platforms
- Over 2,000 Scientific Publications

GET IN TOUCH!

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- Get together at the yearly RIOT Summit:
- News: https://twitter.com/RIOT_OS and https://fosstodon.org/@RIOT_OS
- For Developers and Users: https://forum.riot-os.org
- Support & Discussions on Matrix: https://matrix.to/#/#riot-os:matrix.org
- Get the Source Code and Contribute: https://github.com/RIOT-OS/RIOT
- Show Cases: https://www.hackster.io/riot-os
- Videos on YouTube: https://www.youtube.com/c/RIOT-IoT
- Pics: https://www.flickr.com/people/142412063@N07/
- Getting started with a tutorial on https://riot-os.github.io/riot-course/

GitHub



LITERATURE

- *E. Baccelli et al. “RIOT: An open source operating system for low-end embedded devices in the IoT,” IEEE Internet of Things Journal, December 2018.*
- *O. Hahm, “Enabling Energy Efficient Smart Object Networking at Internet-Scale,” Ecole Polytechnique, December 2016.*
- *O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, “Operating Systems for Low-End Devices in the Internet of Things: a Survey,” IEEE Internet of Things Journal, October 2016.*
- *D. Lacamera, “Embedded Systems Architecture,” O’Reilly, May 2018.*

Any Questions?

LET'S GET STARTED

Go to https://doc.riot-os.org/getting-started.html!