

IoT Project

Introduction

Prof. Dr. Oliver Hahm
Frankfurt University of Applied Sciences
Faculty 2: Computer Science and Engineering
oliver.hahm@fb2.fra-uas.de
<https://teaching.dahahm.de>

Agenda

- 1** About
- 2** Organizational
- 3** Internet of Things
- 4** Software for low-end IoT Devices
- 5** Technical Insights on RIOT
- 6** RIOT Community

Agenda

- 1 About
- 2 Organizational
- 3 Internet of Things
- 4 Software for low-end IoT Devices
- 5 Technical Insights on RIOT
- 6 RIOT Community

About me



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

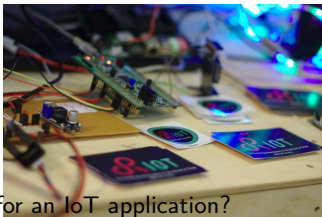
Contact

E-mail: oliver.hahm@fb2.fra-uas.de

Office hours: Tuesdays 13:00 – 14:00, room 1-212

Join the RIOT!

RIOT is the friendly
operating system
for the IoT!



You're interested in ...

- ... realize your own ideas for an IoT application?
- ... collaborate with hundreds of people from all over the world?
- ... contribute to a big FLOSS project?



Get in touch

Get in touch and do some hacking at the **All RIOT** event at the university!

Every two or three weeks in room 1-237.

Or look at <https://riot-os.org/community.html>



What about you?

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

What about you?

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

Agenda

- 1 About
- 2 Organizational**
- 3 Internet of Things
- 4 Software for low-end IoT Devices
- 5 Technical Insights on RIOT
- 6 RIOT Community

What?

- Create a firmware based on RIOT (<https://riot-os.org>)
- The firmware should periodically read sensor data and send it towards an IoT Cloud provider
- Sending data to the cloud requires ...
 - **IPv6** connectivity
 - a **border router**
- Allow remote management and retrieval of sensor data

How?

- Team work (two students per group)
 - \leftrightarrow But grading is individually
- Each team work on a common code base
- git is used as version control system
- Write documentation about your project
- Run (and evaluate) your code on RIOT native and on real hardware
- Present your work

What (and when) to Submit?

- July 14, 2023: Presentation
 - Give a short presentation on your work (live demo?)
- July 28, 2023: Submission
 - Final version of the code is in the repository
 - You have granted access to me
 - Send me your documentation

Evaluation

Which aspects of your work are going to be evaluated?

- Your **implementation** (60%)
 - Functionality (20%)
 - Creativity (10%)
 - System and code architecture (10%)
 - Code quality (10%)
 - Infrastructure (10%)
- The **documentation** (20%)
 - Inline code documentation (5%)
 - Final How-To (15%)
- Your **presentations** (20%)
 - A small midterm presentation (5%)
 - The final presentation (15%)

Grading System

Definition of the Grades

- 1.0 This grade denotes 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 This grade denotes a good performance. The work evaluated meets the expectations and fulfills the requirements well. It may contain some minor or formal errors.
- 3.0 This grade denotes 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 This grade denotes 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 This grade denotes 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.

Further Information

Course page

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

This includes

- Announcements
- Slides
- Dates

campUAS

Enrolment Key:
HahmProject

Additional Sources

Everything related to RIOT can be found at <https://riot-os.org> .

Agenda

- 1 About
- 2 Organizational
- 3 Internet of Things**
- 4 Software for low-end IoT Devices
- 5 Technical Insights on RIOT
- 6 RIOT Community

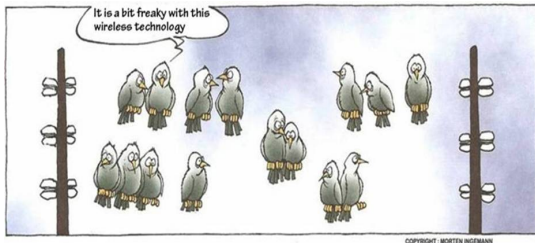
The Evolution of the IoT

Three Disruptive Technologies
as the Roots of the IoT

The Evolution of the IoT

Three Disruptive Technologies as the Roots of the IoT

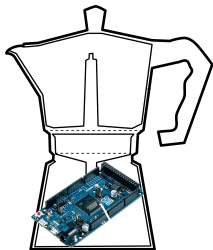
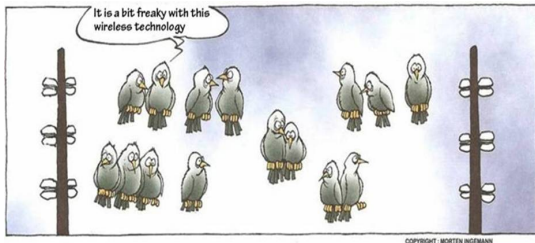
- **Wireless Communication**
- Low-cost Embedded Systems
- The Internet



The Evolution of the IoT

Three Disruptive Technologies as the Roots of the IoT

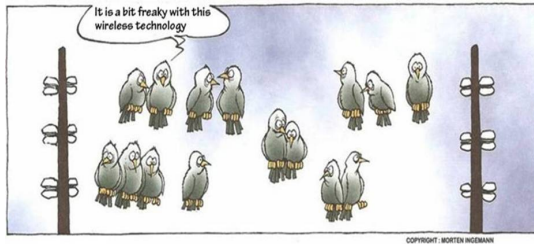
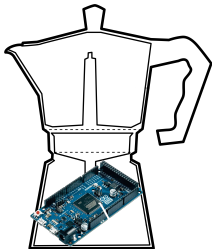
- Wireless Communication
- Low-cost Embedded Systems
- The Internet



The Evolution of the IoT

Three Disruptive Technologies as the Roots of the IoT

- Wireless Communication
- Low-cost Embedded Systems
- The Internet



Smart Object Networking at Internet-Scale

Connecting the Physical World with the Internet

- Transforming Things into Smart Objects
- Enabling Interconnected Smart Services

Smart Object Networking at Internet-Scale

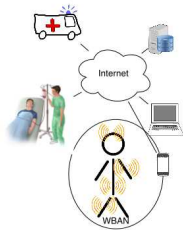
Industrial Automation



Connecting the Physical World with the Internet

- Transforming Things into Smart Objects
- Enabling Interconnected Smart Services

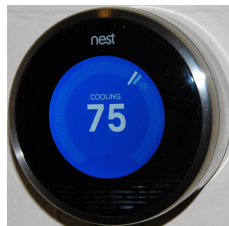
Mobile Health



Micro & Nano Satellites



Building & Home Automation



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

Agenda

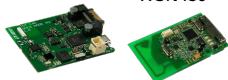
- 1 About
- 2 Organizational
- 3 Internet of Things
- 4 Software for low-end IoT Devices**
- 5 Technical Insights on RIOT
- 6 RIOT Community

Requirements for IoT Software

Low-end IoT Devices: Limited Resources (RFC7228)

iotlab-m3

Senslab
WSN430



Arduino Due



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

Requirements for IoT Software

Low-end IoT Devices: Limited Resources (RFC7228)

iotlab-m3

Senslab
WSN430



Arduino Due



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

+

Use Case Requirements

=

Software Requirements

- Energy Efficiency
- Sustainability
- Network Connectivity
- Real-Time Capabilities
- Small Memory Footprint
- Security and Safety
- Support for Heterogeneous Hardware

Embedded Operating Systems

No User Interaction

- No GUI required \Rightarrow No Pseudo-Parallel Execution is required
- Must Operate Autonomously
 \rightarrow Must Recover from Errors
- Autoconfiguration is required



Source: Embedded Lab, <https://www.electronics-lab.com/>

Embedded Operating Systems

No User Interaction

- No GUI required \Rightarrow No Pseudo-Parallel Execution is required
- Must Operate Autonomously \rightarrow Must Recover from Errors
- Autoconfiguration is required

Constrained Hardware

- Often no MMU¹ and no FPU²
- Typically no Display or Input Devices
- In many cases no Persistent Memory



Source: Embedded Lab, <https://www.electronics-lab.com/>

¹Memory Management Unit

²Floating Point Unit

Embedded Operating Systems

No User Interaction

- No GUI required \Rightarrow No Pseudo-Parallel Execution is required
- Must Operate Autonomously \rightarrow Must Recover from Errors
- Autoconfiguration is required

Constrained Hardware

- Often no MMU¹ and no FPU²
- 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 \rightarrow just one statically linked binary

Source: Embedded Lab, <https://www.electronics-lab.com/>

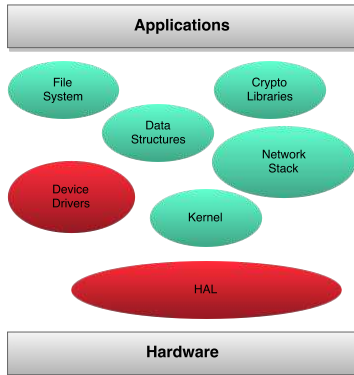
¹Memory Management Unit

Prof. Dr. Oliver Hahn – IoT Project – Introduction – SS 23

²Floating Point Unit

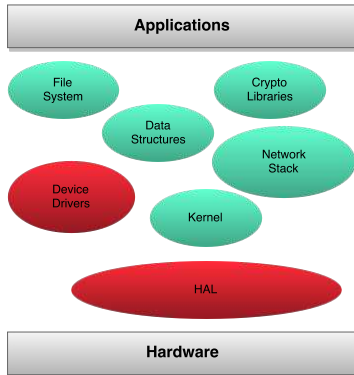
The Need for an OS for Low-end IoT Devices

Unified Software Platform

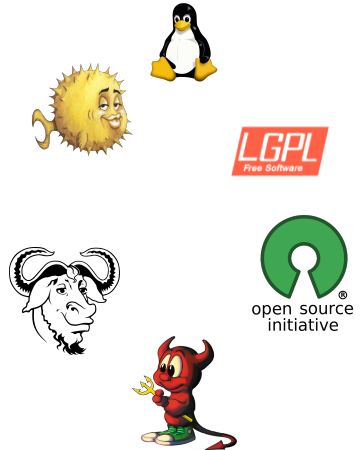


The Need for an OS for Low-end IoT Devices

Unified Software Platform



Open Source



Operating Systems for Low-End IoT Devices: Linux

Full-fledged OS



Does not fit

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

Operating Systems for Low-End IoT Devices: Linux

Full-fledged OS



WSN OS



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

Operating Systems for Low-End IoT Devices: Linux

Full-fledged OS



Does not fit

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

WSN OS



Too Complicated

- Hard to Learn
- No System Level Compatibility

RTOS



Too Minimalistic

- No Built in Networking Support
- No Common API

Agenda

- 1 About
- 2 Organizational
- 3 Internet of Things
- 4 Software for low-end IoT Devices
- 5 Technical Insights on RIOT**
- 6 RIOT Community

The friendly OS for the IoT

"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

- 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



Valgrind



Programming Language and Guidelines

Important Programming Language Properties

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

Why C?

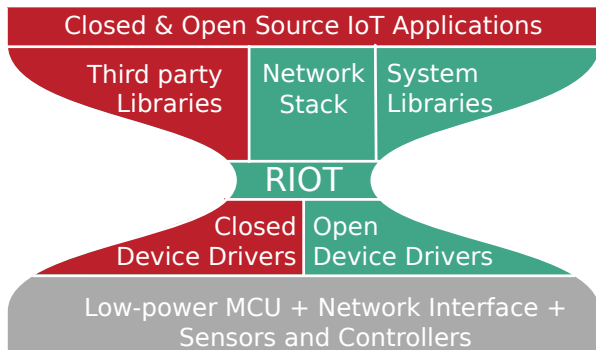
- Ticks all the Boxes
- Stable Specification
- Widely Used → Tooling

```
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”

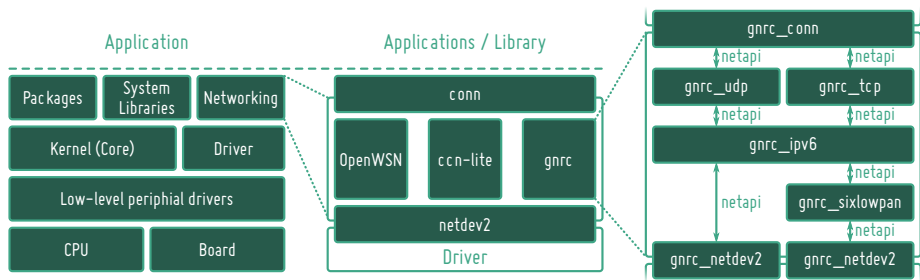
Architectural Overview



Design Decisions

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

The Structure



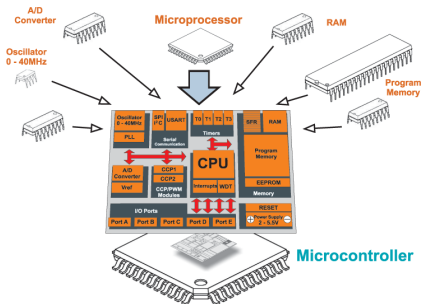
Hardware Abstraction Layer (HAL)

Challenge: Support a Plethora of different Platforms

- Different Processor Architectures (8 bit, 16 bit, 32 bit ...)
- Microcontroller¹Peripherals
- 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>

¹MicroController Unit (MCU)

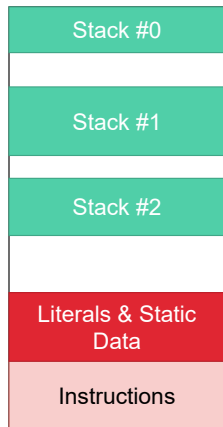
Multi-Threading

- 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

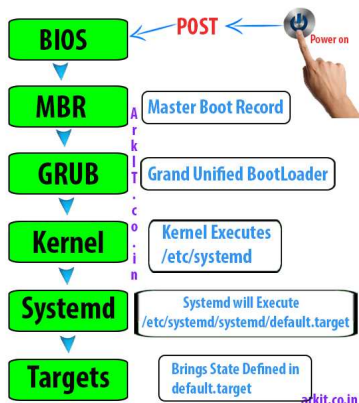
On a Low-end IoT Device (16-bit, 8 MHz):

- Min. TCB: 8 bytes
- Min. Stack Size: 96 bytes
- Up to 16,000 Messages/s
($\hat{=}$ 10,000 Packets/s for 802.15.4)



Boot Sequence

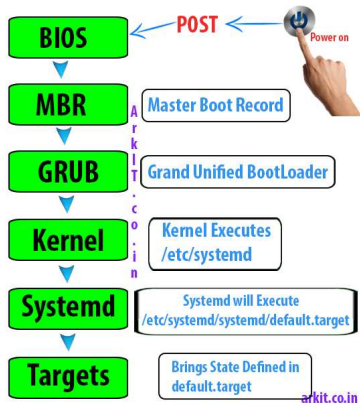
Linux Boot Sequence



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

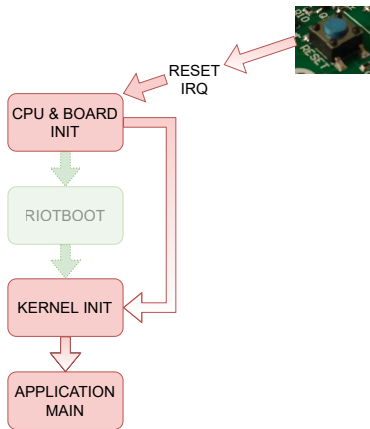
Boot Sequence

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 \Rightarrow All Operations are $O(1)$
- The Behavior of the Kernel is completely deterministic
- Interrupt Handlers are as 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**

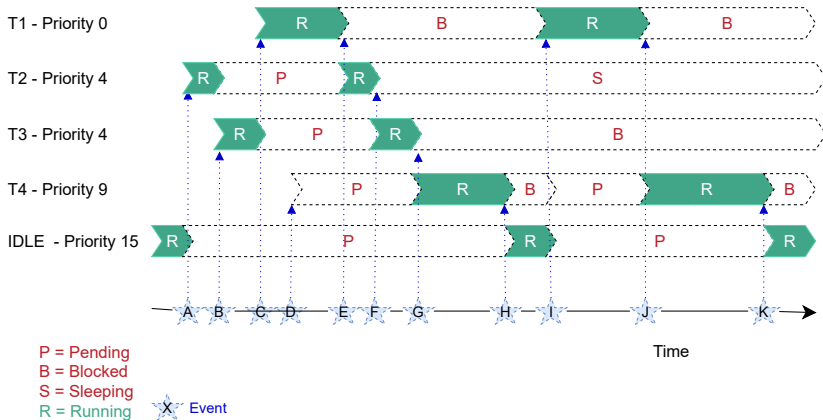
⇒ The Thread is ready to run

It may be blocked waiting for

...

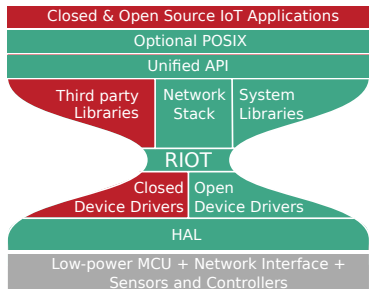
- a mutex
- a message to be received
- a message to be sent
- a response to a previous message
- a thread flag
- an action in its mailbox
- a condition variable

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
→ Not very modern
 - Not tailored for constrained Resources
 - → But facilitates (initial) porting
- A POSIX-like API for this Class of Devices is missing so far



Modularity and Reusability



- 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)

```

    [0] [1] [2] [3] [4] [5] [6]
    (Top)
    RIOT Configuration
    Native modules ----
    [ ] Configure RIOT Core ----
    Drivers -->
    System -->
    Packages ----
    External Modules ----
    *** RIOT is in a migration phase. ***
    *** Some configuration options may not be here. Use CFLAGS instead. ***
    [*] Development Help

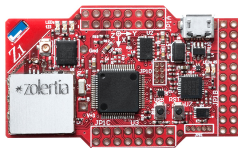
    [Space/Enter] Toggle/enter [ESC] Leave menu [S] Save
    [O] Load [I] Symbol info [J] Jump to symbol
    [F] Toggle show-help mode [C] Toggle show-name mode [A] Toggle show-all mode
    [Q] Quit (prompts for save) [D] Save minimal config (advanced)
  
```

Result: Low Porting Effort

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

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

Memory Comparison



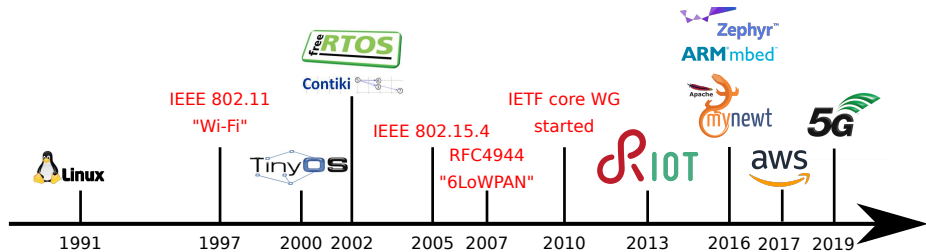
RIOT is as Small as Traditional WSN Operating Systems

Application	ROM	RAM
RIOT 2016.04	52,378	5,618
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.

Review & Perspectives



IoT Software in 2022

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

Agenda

- 1 About
- 2 Organizational
- 3 Internet of Things
- 4 Software for low-end IoT Devices
- 5 Technical Insights on RIOT
- 6 RIOT Community**

10 Years of RIOT!

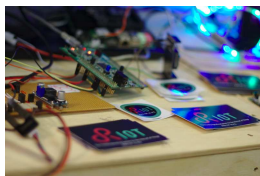
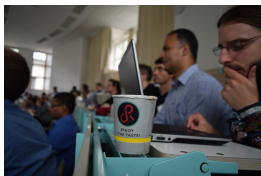
RIOT Open Source Development

- 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!

- Get together at the yearly RIOT Summit:
<https://summit.riot-os.org>
 ↪ **This year at our university!**
- 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/>



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?

