Operating Systems

RIOT: An Open OS for an Open IoT

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

January 18, 2022

Agenda

- Internet of Things
- Software for Low-End IoT Devices
- Case Study: RIOT
- RIOT Community

Agenda

- Internet of Things
- Software for Low-End IoT Devices
- Case Study: RIOT
- RIOT Community

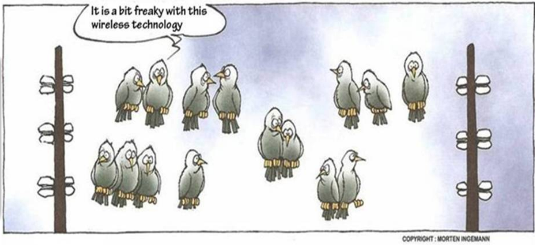
The Evolution of the IoT

Three Disruptive Technologies as the Roots of the IoT

The Evolution of the IoT

Three Disruptive Technologies as the Roots of the IoT

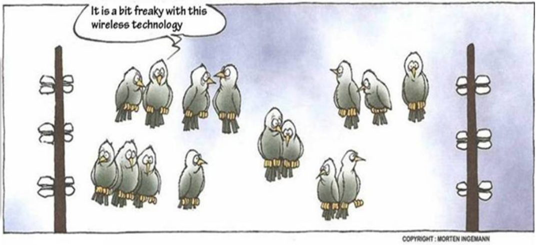
- Wireless Communication

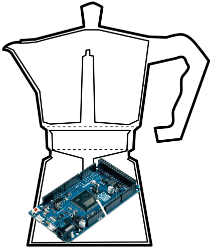


The Evolution of the IoT

Three Disruptive Technologies as the Roots of the IoT

- Wireless Communication
- Low-cost Embedded Systems

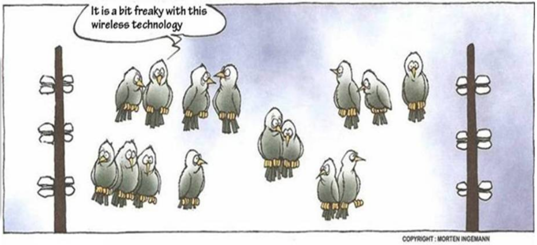


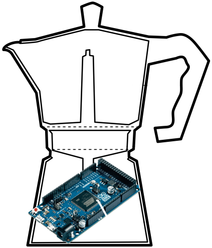


The Evolution of the IoT

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

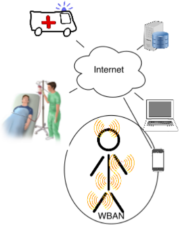
Smart Object Networking at Internet-Scale

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

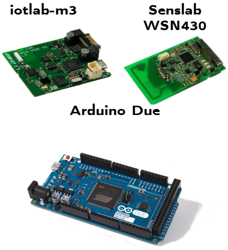
It ain't smart if I have to charge it every day.

Agenda

- Internet of Things
- Software for Low-End IoT Devices
- Case Study: RIOT
- RIOT Community

Requirements for IoT Software

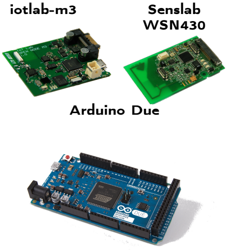
Low-end IoT Devices: Limited Resources (RFC7228)



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

Requirements for IoT Software

Low-end IoT Devices: Limited Resources (RFC7228)



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

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

Embedded Operating Systems

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

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

[1] Memory Management Unit
[2] Floating Point Unit

Embedded Operating Systems

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

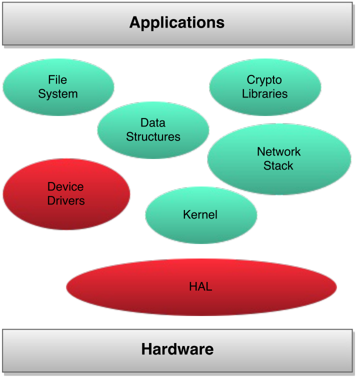
Source: Embedded Lab, https://www.electronics-lab.com/

- 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

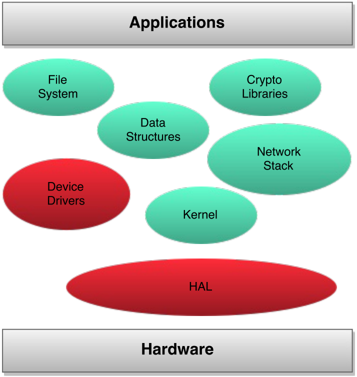
The Need for an OS for Low-end IoT Devices

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

Linux

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

Operating Systems for Low-End IoT Devices: Linux

Full-fledged OS

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

WSN OS

- Hard to Learn
- No System Level Compatibility

Operating Systems for Low-End IoT Devices: Linux

Full-fledged OS

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

WSN OS

- Hard to Learn
- No System Level Compatibility

RTOS

- No Built in Networking Support
- No Common API

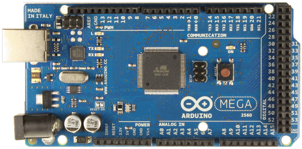
Agenda

- Internet of Things
- Software for Low-End IoT Devices
- Case Study: RIOT
- 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

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?

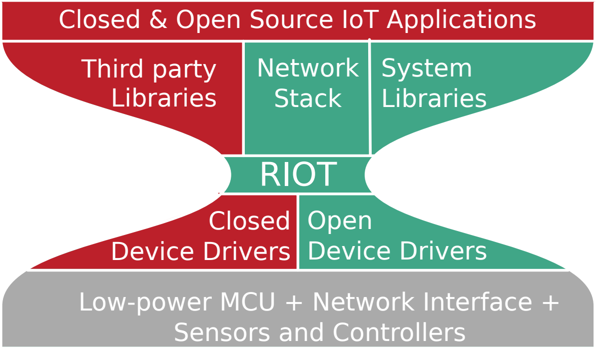
- Ticks all the Boxes
- Stable Specification
- Widely Used $\rightarrow$ Tooling

```c
void thread_yield(void)
{
    unsigned old_state = irq_disable();
    thread_t *me = thread_get_active();

    if (me->status >= STATUS_ON_RUNQUEUE) {
        sched_runq_advance(me->priority);
    }
    irq_restore(old_state);

    thread_yield_higher();
}
```

Programming Guidelines

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

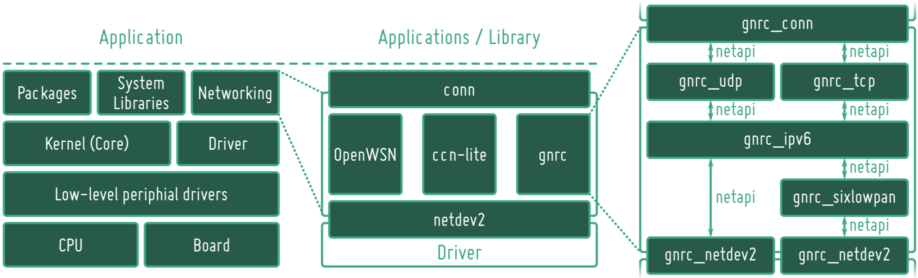
Architectural Overview



Design Decisions

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

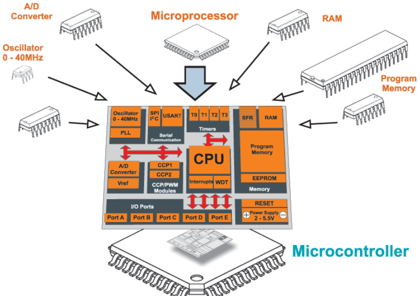
The Structure



Hardware Abstraction Layer (HAL)

Challenge: Support a Plethora of different Platforms

- Different Processor Architectures (8 bit, 16 bit, 32 bit . . .)
- Microcontroller[1] Peripherals
- Sensors and Actuators
- Network Devices
- Crypto Devices
- . . .



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

Goal: Provide a Common API

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

[1] MicroController Unit (MCU)

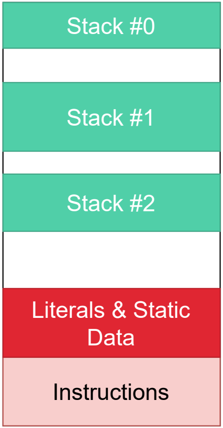
Multi-Threading

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

Low Memory Usage

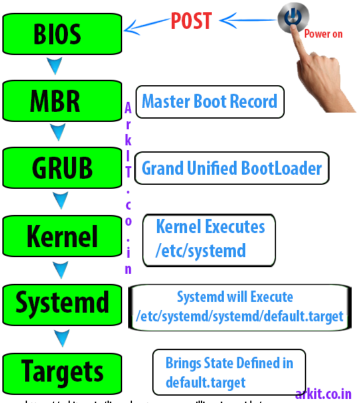
On a Low-end IoT Device (16-bit, 8 MHz):

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



Boot Sequence

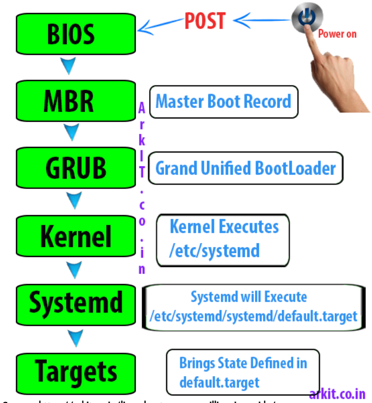
Linux Boot Sequence



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

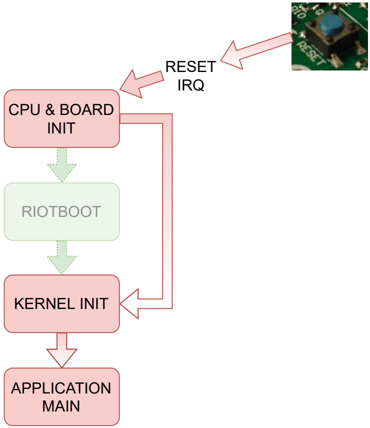
Boot Sequence

Linux Boot Sequence



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

RIOT Boot Sequence

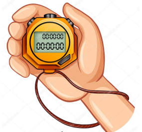


Scheduling

- 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



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

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

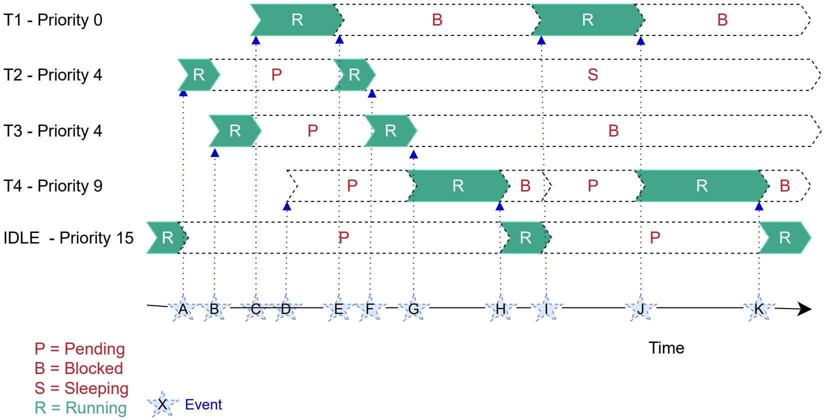
Thread States

- 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
  $\Rightarrow$ The Thread is ready to run

It may be blocked waiting for ...

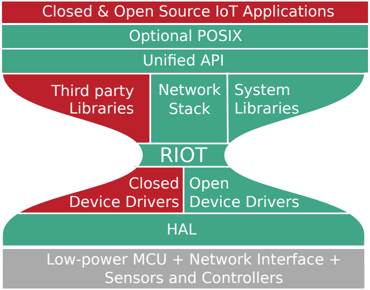
- a mutex
- a message to be received
- a message to be sent
- a response to a previous message
- a thread flag
- an action in its mailbox
- a condition variable

Scheduling example



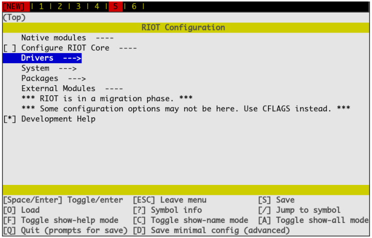
Application Programming Interface (API)

- 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

- 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 ($\rightarrow$ Even Multi-Threading is optional)

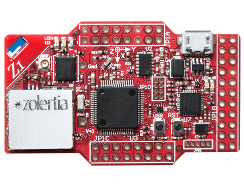


Result: Low Porting Effort

- 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 %
relic	24 lines	<0.1 %

Memory Comparison



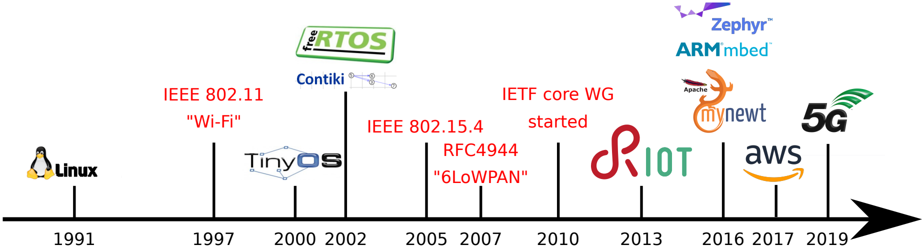
RIOT is as Small as Traditional WSN Operating Systems

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

- Internet of Things
- Software for Low-End IoT Devices
- Case Study: RIOT
- RIOT Community

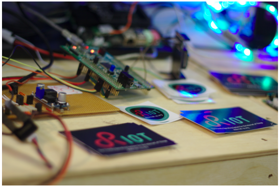
9 Years of RIOT

RIOT Open Source Development

- More Than 38,000 Commits and More Than 14,000 Pull Requests
- Over 1,800 forks on GitHub
- More Than 290 Contributors
- Support for More Than 230 Hardware Platforms
- Over 500 Scientific Publications

Get in touch!

- 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/
- Get together at the yearly RIOT Summit: https://summit.riot-os.org
- 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.*

Source: Pubs and Publications, https://i1.wp.com/www.blogs.hss.ed.ac.uk/pubs-and-publications/files/2016/10/books.jpg?fit=945

www.riot-os.org