



The friendly OS for the IoT

Martine S. Lenders, Christian Amsüss, Marian Buschsieweke

2024-08-17, FrOSCon 19

Introduction: What is RIOT?

Developing with RIOT

Workshop

Introduction: What is RIOT?


Developing with RIOT

Workshop

Who are we?


Martine S. Lenders



- RIOT contributor since 2011, maintainer since 2013
-  miri64

Christian Amsüss



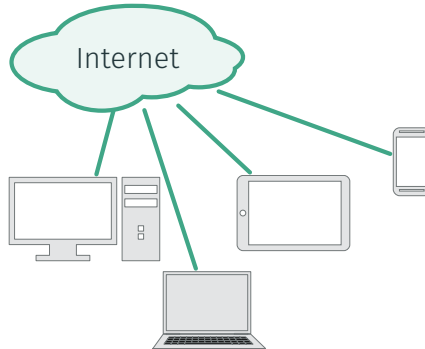
- RIOT contributor since 2016, maintainer since 2020
-  chrysn

Marian Buschsieweke

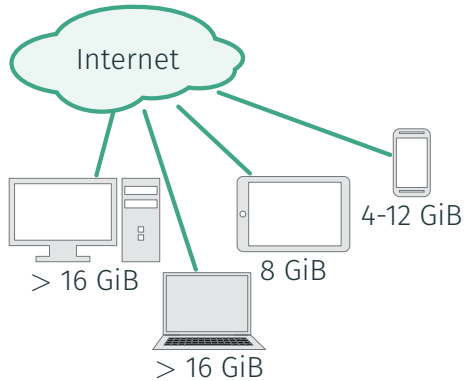


- RIOT contributor & maintainer since 2018
-  maribu

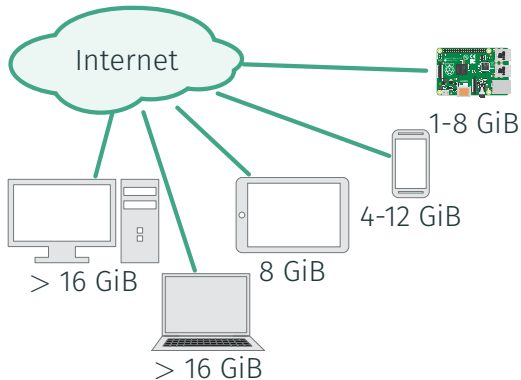
RIOT: An OS for the Constrained IoT



RIOT: An OS for the Constrained IoT

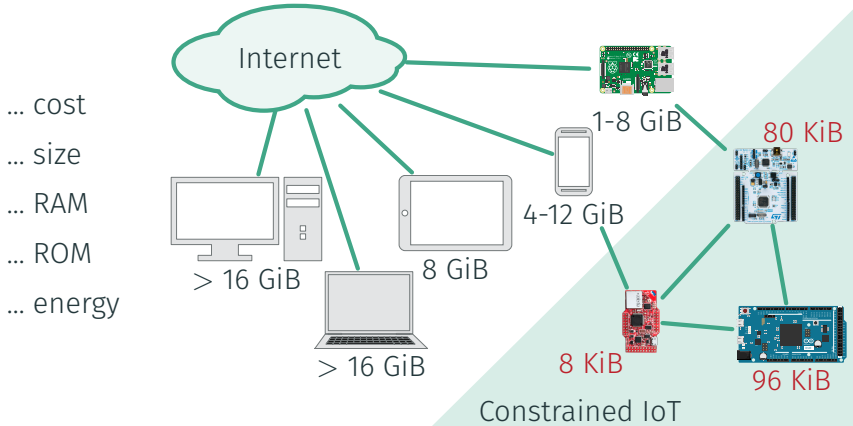


RIOT: An OS for the Constrained IoT



RIOT: An OS for the Constrained IoT

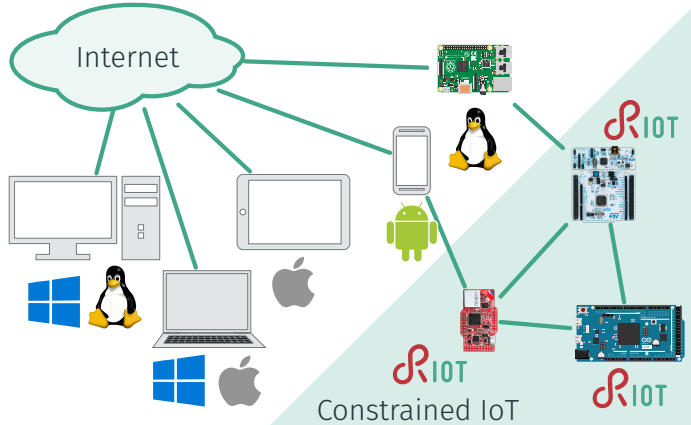
IoT devices are constrained and have orders of magnitude lower ...



RIOT: An OS for the Constrained IoT

IoT devices are constrained and have orders of magnitude lower ...

... cost
... size
... RAM
... ROM
... energy



RIOT: The Friendly OS for the IoT

- Free and Open Source
- Small memory footprint (≈ 1.5 KiB RAM, ≈ 5 KiB ROM)
- Hardware abstraction
- Full featured, expendable network stacks

Design principles

Adaptive: High modularity and easy tailoring

Energy efficient: Tickless scheduler

Straight forward: Preemptive multi-threading & powerful IPC

Real-time capable: Deterministic $\mathcal{O}(1)$ scheduling

Reactive: Low latency interrupt handling

From Rapid Prototyping to Deployment

- Build secure, future IoT devices
 - With remote software updates
 - Based on interoperable open standards
- Vendor independent support: TI, NXP, STM, Nordic, Atmel, Silicon Labs, ESP
- Implement once, reuse and reassemble
- Runs on various 8-bit, 16-bit, 32-bit, 64-bit platforms
 - 8-bit: AVR8
 - 16-bit: MSP430
 - 32-bit: ARM7, x86, Cortex-M, RISC-V, ESP
 - 64-bit: ARM64, native64
- Over 270 boards and over 400 sensors and peripherals supported



Supported Connectivity Technologies

Wireless Personal Area Networks (WPANs)

- IEEE 802.15.4(e)
- Bluetooth Low Energy (BLE)
- Near Field Communication (NFC)

Local Area Networks (LANs)

- Ethernet
- Wi-Fi
- CAN Bus

Long-range Technologies

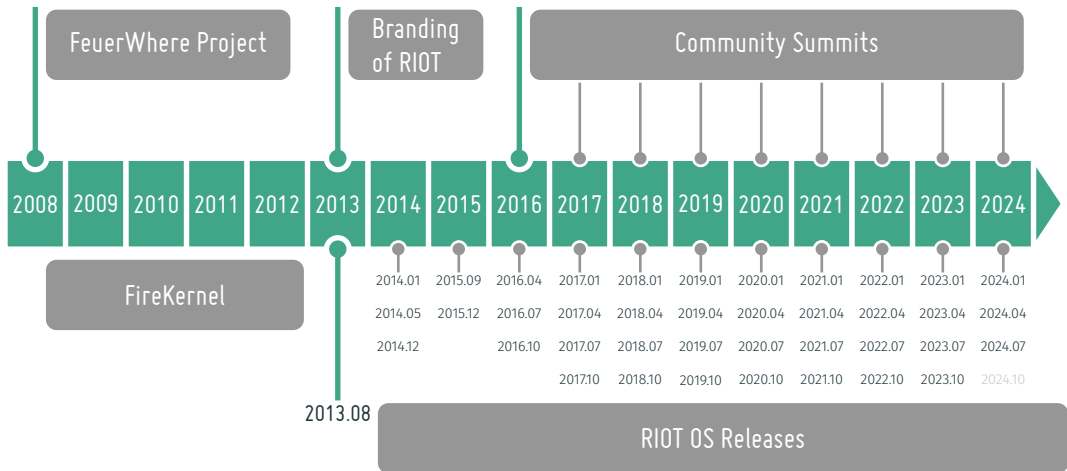
- LoRaWAN

Cellular Technologies (ongoing)

- NB-IoT
- LTE-M

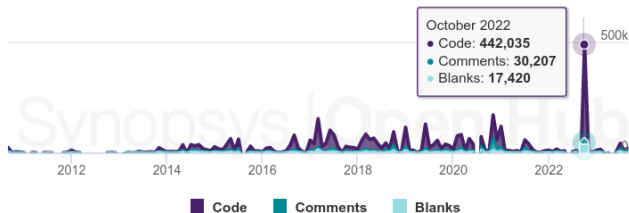


A Bit of RIOT History

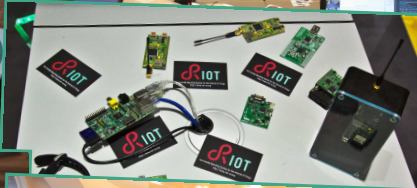
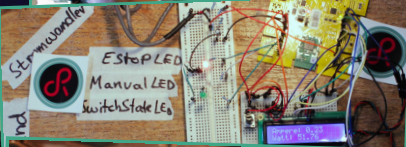
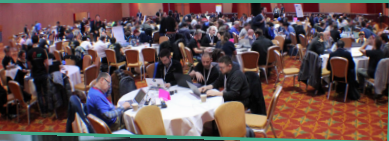


Open Source Community

- RIOT licensed non-viral copyleft (LGPLv2.1)
- Self-organizing community implementing IETF spirit, rough consensus decision-making and running code
- Public development and discussions on GitHub, Discourse forum, and Matrix
- Over 450 contributors around the world, from academia, industry, and hobbyists
- ≈ 2000 forks, $\approx 46,000$ commits, and ≈ 3.4 million lines of code



We care about our community




We care about our community


Meet us in Vienna!


R IOT Summit

September 5 – 7, 2024

<https://summit.riot-os.org/>

**TECHNISCHE
UNIVERSITÄT
DRESDEN**

**Chair of Distributed and
Networked Systems**

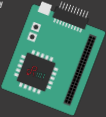
**RIOT**

HACK'n'ACK

Help us develop and maintain the friendly
Operating System for the Internet of Things.

Join the get-together of the RIOT community
to hack, review PRs and meet new people!

- > Become a contributor
- > Learn about
 - > Microcontrollers
 - > Internet of Things
 - > Github
 - > and much more
- > Newbies and experts are welcome




Free Drinks
and Pizza!


EVERY LAST TUESDAY OF THE MONTH @ 5PM

Andreas-Piltzmann-Bau (APB), Room 3105

<https://netd.cs.tu-dresden.de/about/events/riot-hack-n-ack>
<https://www.riot-os.org>



- Monthly get-together of the RIOT community
 - Last Tuesday of the month at 5pm (next: 2024-05-28)
- Hybrid participation (on-site and remote)
- Targets newbies and experts

 <https://forum.riot-os.org/t/3098>

 <https://meet.jit.si/riot-hacknack>

 <https://github.com/netd-tud/WSN-exercises>

Introduction: What is RIOT?

Developing with RIOT

- Getting Started

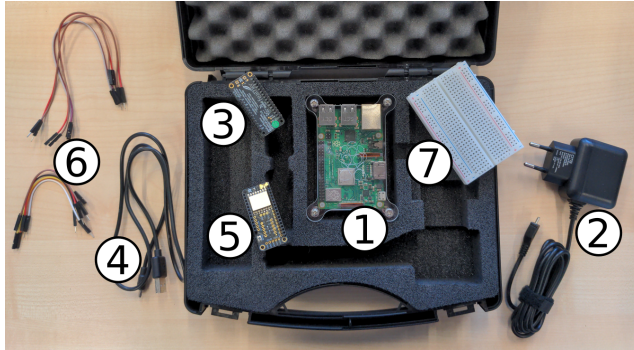
- Tooling

- RIOT Development Concepts

- Firmware Development Flow

Workshop

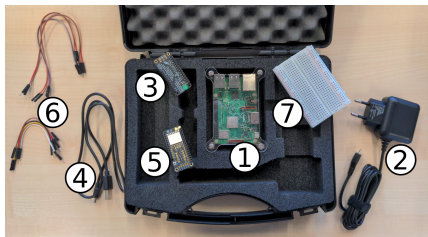
The RIOT Exercise Kit



1. Raspberry Pi: Your fully set-up development environment
2. Power adapter for the Raspberry Pi
3. Adafruit Feather Sense: The IoT device
4. USB cable to connect Feather and Raspberry Pi
5. Adafruit LoRa Radio FeatherWing
6. Jumper cables
7. Breadboard

How to Connect to the IDE

<https://github.com/netd-tud/WSN-exercises/tree/master/00-getting-started>



- Power-on Raspberry Pi using the power adapter
- Connect the Feather Sense (3) to the Raspberry Pi (1) using the USB cable (4)
- With your own computer, connect to the locally provided WiFi network:

SSID: hack_n_ack

Password: ThefriendlyOSfortheIoT

- Open <https://riot-raspi-ffffff.local:8080> (replacing fffffff with the six-digit identifier on the back of your Raspberry Pi), accept self-signed certificate
- Enter credentials

User: pi

Password: riottutorial

- Build system based on GNU Make
- Cross-Compilation
 - GCC, Clang, Dockerized toolchains
- C is our default language
 - Strict coding conventions
 - Module separation
 - Alternatives: Rust, C++, Arduino sketches, (experimental: micropython, JS, Lua)

RIOT Development Concepts

- Static memory
 - Dynamic allocation (e.g., malloc etc.) avoided when possible
- Multithreading
- IPC
 - Message passing
 - Mutex and semaphores
 - Thread-flags
 - Event loops

- Write your own application or use examples (e.g., RIOT/examples)

RIOT Firmware Development Flow

- Write your own application or use examples (e.g., `RIOT/examples`)
- **Cross-compile** for your target architecture, this generates the executable binary
 - The build system gets target device from the **BOARD** environmental variable

RIOT Firmware Development Flow

- Write your own application or use examples (e.g., `RIOT/examples`)
- **Cross-compile** for your target architecture, this generates the executable binary
 - The build system gets target device from the **BOARD** environmental variable
- **Flash** the binary onto your target device

RIOT Firmware Development Flow

- Write your own application or use examples (e.g., `RIOT/examples`)
- **Cross-compile** for your target architecture, this generates the executable binary
 - The build system gets target device from the **BOARD** environmental variable
- **Flash** the binary onto your target device
- **Connect** to your device via a serial connection
 - You can access the stdio of the device

RIOT Firmware Development Flow

- Write your own application or use examples (e.g., RIOT/examples)
- **Cross-compile** for your target architecture, this generates the executable binary
 - The build system gets target device from the **BOARD** environmental variable
- **Flash** the binary onto your target device
- **Connect** to your device via a serial connection
 - You can access the stdio of the device

BOARD=feather-nrf52840-sense make -C examples/hello-world **all flash term**

RIOT Firmware Development Flow

- Write your own application or use examples (e.g., RIOT/examples)

main.c

```
#include <stdio.h>

int main(void)
{
    puts("Hello World!");

    printf(
        "You are running RIOT on a(n) %s board.\n",
        RIOT_BOARD
    );
    printf(
        "This board features a(n) %s MCU.\n",
        RIOT_MCU
    );

    return 0;
}
```

Makefile

```
APPLICATION = "hello-world"
BOARD ?= "feather-nrf52840-sense"
RIOTBASE ?= $(CURDIR)/../RIOT
# USEMODULE += ztimer
DEVELHELP ?= 1
include $(RIOTBASE)/Makefile.include
```


RIOT Firmware Development Flow

Cross-compile

```
➔ RIOT git:(master) ! BOARD=pba-d-01-kw2x make -C examples/hello-world/ all
make: Entering directory '/home/user/RIOT/examples/hello-world'
Building application "hello-world" for "pba-d-01-kw2x" with MCU "kinetis".

"make" -C /home/user/RIOT/pkg/cnsis/
"make" -C /home/user/RIOT/boards/common/init
"make" -C /home/user/RIOT/boards/pba-d-01-kw2x
"make" -C /home/user/RIOT/core
"make" -C /home/user/RIOT/core/lib
"make" -C /home/user/RIOT/cpu/kinetis
"make" -C /home/user/RIOT/cpu/cortexm_common
"make" -C /home/user/RIOT/cpu/cortexm_common/periph
"make" -C /home/user/RIOT/cpu/kinetis/periph
"make" -C /home/user/RIOT/drivers
"make" -C /home/user/RIOT/drivers/periph_common
"make" -C /home/user/RIOT/sys
"make" -C /home/user/RIOT/sys/auto_init
"make" -C /home/user/RIOT/sys/div
"make" -C /home/user/RIOT/sys/libc
"make" -C /home/user/RIOT/sys/malloc_thread_safe
"make" -C /home/user/RIOT/sys/newlib_syscalls_default
"make" -C /home/user/RIOT/sys/pn_layered
"make" -C /home/user/RIOT/sys/preprocessor
"make" -C /home/user/RIOT/sys/stdio_uart

text data bss dec hex filename
9832 104 2612 12548 3104 /home/user/RIOT/examples/hello-world/bin/pba-d-01-kw2x/hello-world.elf
make: Leaving directory '/home/user/RIOT/examples/hello-world'
➔ RIOT git:(master) $
```

```
#include <stdio.h>

int main(void)
{
    puts("Hello World!");

    printf(
        "You are running RIOT on a(n) %s board.\n",
        RIOT_BOARD
    );
    printf(
        "This board features a(n) %s MCU.\n",
        RIOT_MCU
    );

    return 0;
}
```

ARM
0010
1100
1010



AVR
1110
0110
0110

RIOT Firmware Development Flow

Flash the binary

```
➔ RIOT git:(master) ➤ BOARD=pba-d-01-kw2x make -C examples/hello-world/ flash-only
make: Entering directory '/home/user/RIOT/examples/hello-world'
/home/user/RIOT/dist/tools/openocd/openocd.sh flash /home/user/RIOT/examples/hello-world/bin/pba-d-01-kw2x/hello-world.elf
## Flashing Target ##
/home/user/RIOT/examples/hello-world/bin/pba-d-01-kw2x/hello-world.elf is not locked.
Open On-Chip Debugger 0.12.0-rc1+dev-00059-g92169e9f5-dirty (2022-10-24-16:23)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html

swd
Info : add flash_bank kinetis kx.pflash
srst_only separate srst_nogate srst_open_drain connect_assert_srst

Info : CMSIS-DAP: SMD supported
Info : CMSIS-DAP: FW Version = 1.0
Info : CMSIS-DAP: Interface Initialised (SMD)
Info : SWCLK/TCK = 0 SWDIO/TMS = 1 TDI = 0 TDO = 0 nTRST = 0 nRESET = 1
Info : Connecting under reset
Info : CMSIS-DAP: Interface ready
Info : clock speed 1000 kHz
Info : SWD DPIDR 0x2ba01477
Info : [kx.cpu] Cortex-M4 r8pl processor detected
Info : [kx.cpu] target has 6 breakpoints, 4 watchpoints
Info : MDM: Chip is unsecured. Continuing.
Info : starting gdb server for kx.cpu on 0
Info : Listening on port 36681 for gdb connections
Info : [kx.cpu] external reset detected

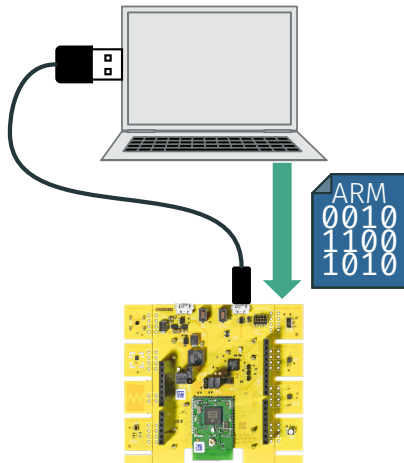
-----
TargetName      Type      Endian TapName      State
-----
0* kx.cpu        cortex_m  little kx.cpu      reset

Info : MDM: Chip is unsecured. Continuing.
[kx.cpu] halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x000009e8 msp: 0x1fff2000
Info : Kinetis MK21DN512xx5 detected: 2 flash blocks
Info : 2 PFlash banks: 512 KiB total
auto erase enabled
wrote 10240 bytes from file /home/user/RIOT/examples/hello-world/bin/pba-d-01-kw2x/hello-world.elf in 0.380103s (26.309 KiB/s)

34 bytes written at address 0x20000000
downloaded 34 bytes in 0.004228s (7.853 KiB/s)

[kx.cpu] halted due to breakpoint, current mode: Thread
xPSR: 0x01000000 pc: 0x20000020 msp: 0x1fff2000
verified 9936 bytes in 0.322125s (30.122 KiB/s)

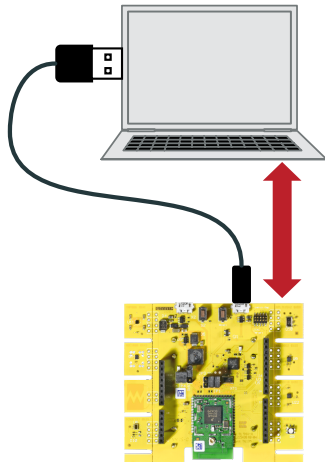
Info : MDM: Chip is unsecured. Continuing.
shutdown command invoked
Done flashing
make: Leaving directory '/home/user/RIOT/examples/hello-world'
➔ RIOT git:(master) ➤
```



RIOT Firmware Development Flow

Connect to the serial terminal

```
➔ RIOT git:(master) ➤ BOARD=pba-d-01-kw2x make -C examples/hello-world/ term
make: Entering directory '/home/user/RIOT/examples/hello-world'
/home/user/RIOT/dist/tools/pyterm/pyterm -p "/dev/ttyACM0" -b "115200"
Twisted not available, please install it if you want to use pyterm's J50N capabilities
2023-09-08 10:56:00,460 # Connect to serial port /dev/ttyACM0
Welcome to pyterm!
Type '/exit' to exit.
2023-09-08 10:56:03,387 # main(): This is RIOT! (Version: 2023.10-devel-247-ga18256)
2023-09-08 10:56:03,388 # Hello World!
2023-09-08 10:56:03,392 # You are running RIOT on a(n) pba-d-01-kw2x board.
2023-09-08 10:56:03,395 # This board features a(n) kinetis MCU.
2023-09-08 10:56:05,978 # Exiting Pyterm
make: Leaving directory '/home/user/RIOT/examples/hello-world'
➔ RIOT git:(master) $
```



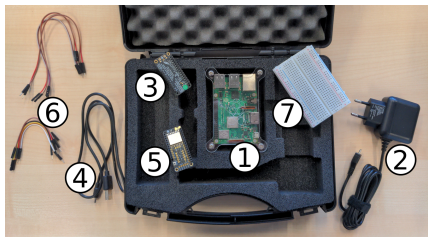
Introduction: What is RIOT?

Developing with RIOT

Workshop

How to Connect to the IDE

<https://github.com/netd-tud/WSN-exercises/tree/master/00-getting-started>



- Power-on Raspberry Pi using the power adapter
- Connect the Feather Sense (3) to the Raspberry Pi (1) using the USB cable (4)
- With your own computer, connect to the locally provided WiFi network:

SSID: `hack_n_ack`

Password: `ThefriendlyOSfortheIoT`

- Open `https://riot-raspi-ffffff.local:8080` (replacing `ffffff` with the six-digit identifier on the back of your Raspberry Pi), accept self-signed certificate
- Enter credentials

User: `pi`

Password: `riottutorial`

We care about our community

Meet us in Vienna!

R IOT Summit

September 5 – 7, 2024

<https://summit.riot-os.org/>