Zagreb, Real-Time Systems

RIOT OS & LoRa with RIOT OS

Juraj Vijtiuk · Luka Paulić

May 27, 2019

About us

- Juraj Vijtiuk, Luka Paulić
- Delivering solutions based on Linux, OpenWrt and Yocto
  - Focused on software in network edge and CPEs
- Continuous participation in Open Source projects

RIOT OS

Based on https://aabadie.github.io/riot-course

sartura

Introduction

- Operating system for embedded devices with a focus on IOT
  - Microkernel, real time, multithreaded
  - Support for 8-bit, 16-bit and 32-bit MCUs
  - Unified API, independent of the underlying hardware
  - Mostly POSIX compliant

- Open Source
  - LGPLv2.1 licence
  - `https://github.com/RIOT-OS/RIOT`
  - Active community, started at Inria, FU Berlin and HAW in 2013
- Written in C
- Verbose official documentation

Features

- Real time scheduler
  - Preemptive, tickless O(1) scheduling with priorities
- Support for over 100 boards
  - Native board allows RIOT to run as a process on Linux, BSD and Mac
  - Multiple native instances can be connected via virtual bridges
- Generic API for CPU peripherals
  - Gpio, uart, spi, pwm
  - Identical API for all architectures

- Build time configurable modules
    - Driver support for sensors and actuators can be included at build time
    - System libraries: shell, crypto, xtime
    - Network stacks: LoRaWAN, Sigfox, CoAP, IPv6
    - Support for external packages
- A large number of available example applications

Building a simple RIOT application

- RIOT uses GNU make as the default build system
- All applications use a single Makefile which then includes the main Makefile.include RIOT makefile
- `make flash`, `make term`, `make debug`
- Example:

```
APPLICATION = example
BOARD ?= native
RIOTBASE ?= $(CURDIR)/../../../RIOT
DEVELHELP ?= 1
include $(RIOTBASE)/Makefile.include
```

- `USEMODULE`
  - Support for shell as a module
- `USEPKG`
- `FEATURES_REQUIRED`
- Example:

```
USEMODULE += xtimer shell
USEPKG += semtech-loramac
FEATURES_REQUIRED += periph_gpio
```

Accessing hardware

- Access at 3 levels
  - Board
  - CPU
  - Driver
- Hardware layer separation is reflected in the source code
  - `boards/`, `core/`, `cpu/`, `drivers/`, `pkg/`, `sys/`, `tests/`
- E.g. xtimer

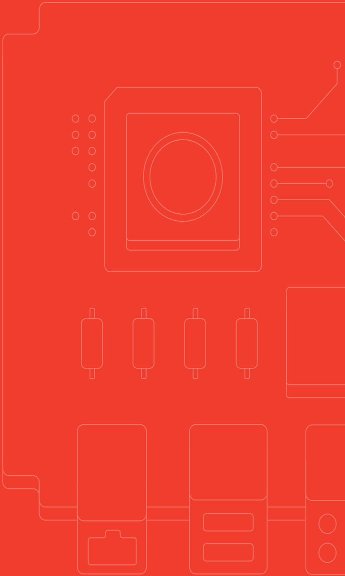
Multithreading

- Tick-less O(1) scheduling
- Highest priority thread runs until finished or interrupted
- Idle thread
  - Power management
- Threads are functions with separate stacks
- Support for synchronization mechanisms and IPC
  - Mutexes, semaphores, other POSIX mechanisms
  - Synchronous & asynchronous messaging

Networking

- Multiple network stacks available
  - IPv6: gnrc, LwIP, emb6
  - BLE, CAN, LoRaWAN
- Sock API
- IPv6 utility library
- POSIX socket API

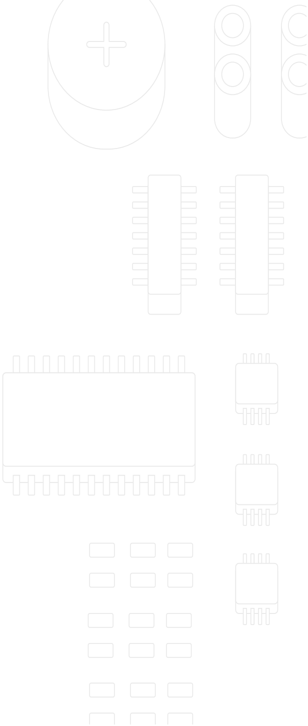
LoRa with RIOT OS



sartura

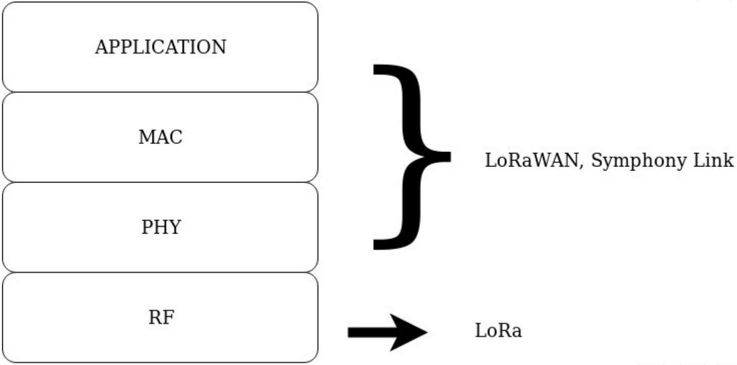
What is LoRa

- LoRa - Long Range
- Acquired by Semtech in 2012.
- CSS modulation
- ISM frequency band



- Pros
    - Long range; >10 km
    - Low power; <= 10 years
- Cons
    - Low data rate; 300 bps - 50 kbps
    - Small data size; ~10 B
    - Regional restrictions
- IoT

LoRa protocol stack



APPLICATION

MAC

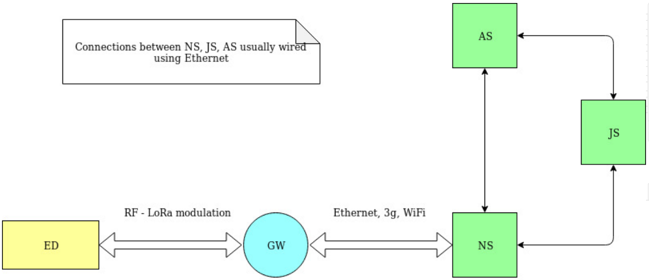
PHY

RF

LoRaWAN, Symphony Link

LoRa

LoRa network architecture



Connections between NS, JS, AS usually wired using Ethernet

AS

JS

ED

RF - LoRa modulation

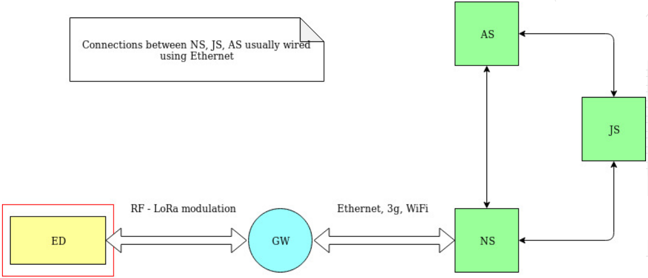
GW

Ethernet, 3g, WiFi

NS

Descripiton:

- ED => end device
- GW => LoRa gateway
- NS => network server
- JS => joint server
- AS => application server



Connections between NS, JS, AS usually wired using Ethernet

AS

JS

ED

RF - LoRa modulation

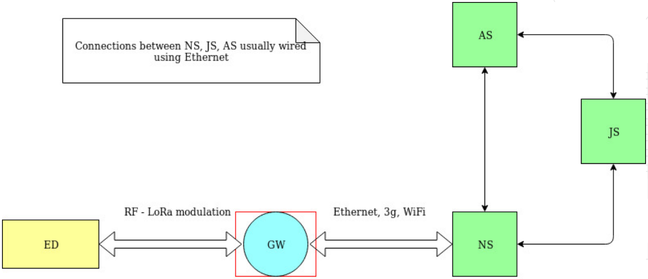
GW

Ethernet, 3g, WiFi

NS

Descripiton:

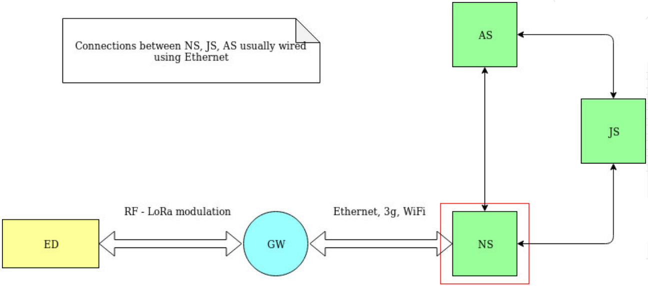
- ED => end device
- GW => LoRa gateway
- NS => network server
- JS => joint server
- AS => application server



Connections between NS, JS, AS usually wired using Ethernet

Descripiton:

- ED => end device
- GW => LoRa gateway
- NS => network server
- JS => joint server
- AS => application server



Connections between NS, JS, AS usually wired using Ethernet

AS

JS

RF - LoRa modulation

Ethernet, 3g, WiFi

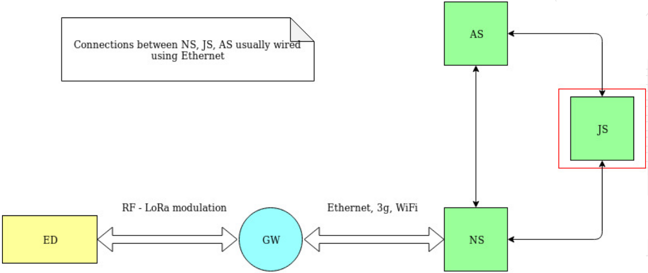
ED

GW

NS

Descripiton:

- ED => end device
- GW => LoRa gateway
- NS => network server
- JS => joint server
- AS => application server



Connections between NS, JS, AS usually wired using Ethernet

ED

RF - LoRa modulation

GW

Ethernet, 3g, WiFi

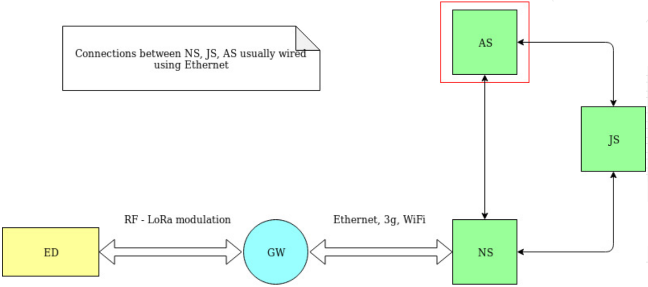
NS

AS

JS

Descripiton:

- ED => end device
- GW => LoRa gateway
- NS => network server
- JS => joint server
- AS => application server



Connections between NS, JS, AS usually wired using Ethernet

RF - LoRa modulation

Ethernet, 3g, WiFi

Descripiton:

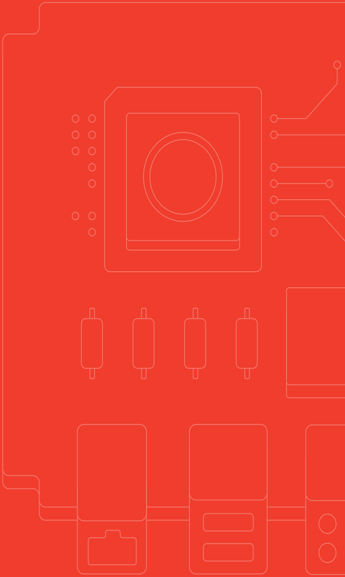
- ED => end device
- GW => LoRa gateway
- NS => network server
- JS => joint server
- AS => application server

LoRa in RIOT OS

- Using Semtech LoRaMAC library
- Makefile example

```
DEVEUI ?= E3632DDB4DE3A955
APPKEY ?= 585F77E5C5FD75833B97FC25F7754391
# Default radio driver is Semtech SX1276 (used by the B-L072Z-LRWAN1 board)
DRIVER ?= sx1276
# Default region is Europe and default band is 868MHz
REGION ?= EU868
# Include the Semtech-loramac package
USEPKG += semtech-loramac
USEMODULE += $(DRIVER)
CFLAGS += -DREGION_$(REGION)
CFLAGS += -DDEVEUI=\"$(DEVEUI)\" -DAPPEUI=\"$(APPEUI)\" -DAPPKEY=\"$(APPKEY)\"
CFLAGS += -DLORAMAC_ACTIVE_REGION=LORAMAC_REGION_$(REGION)
```

Demonstration



RIOT OS & LoRa with RIOT OS

juraj.vijtiuk@sartura.hr · luka.paulic@sartura.hr

info@sartura.hr · www.sartura.hr