Seamless Power Management on IoT Devices — Lessons from an HVAC Use Case using RIOT

Jürgen Fitschen - SSV Software Systems GmbH

Slides available at:
https://ssv-embedded.github.io/RIOTSummit2020/

Who am I?

Jürgen Fitschen (jue89 on GitHub)
Systems Engineer at SSV Software Systems, Germany
Using RIOT since 2018

What we'll cover ...

Why does good power management *matter*?
How does it *work*?
It's all about *timers*!

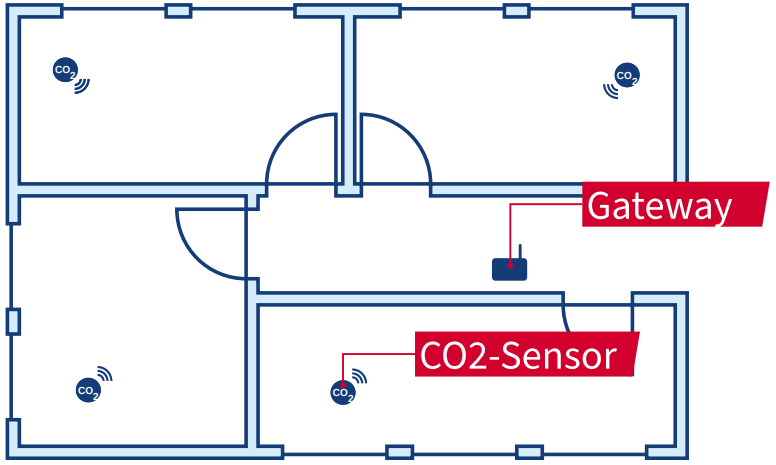
Why does Good Power Management *Matter* to us?

Retrofit Systems are our Passion.

- Goal: enhance efficiency and value of existing systems and environments
- Sensors and actuators must be deployed within already existing systems
- Some retrofit systems require 100+ sensors

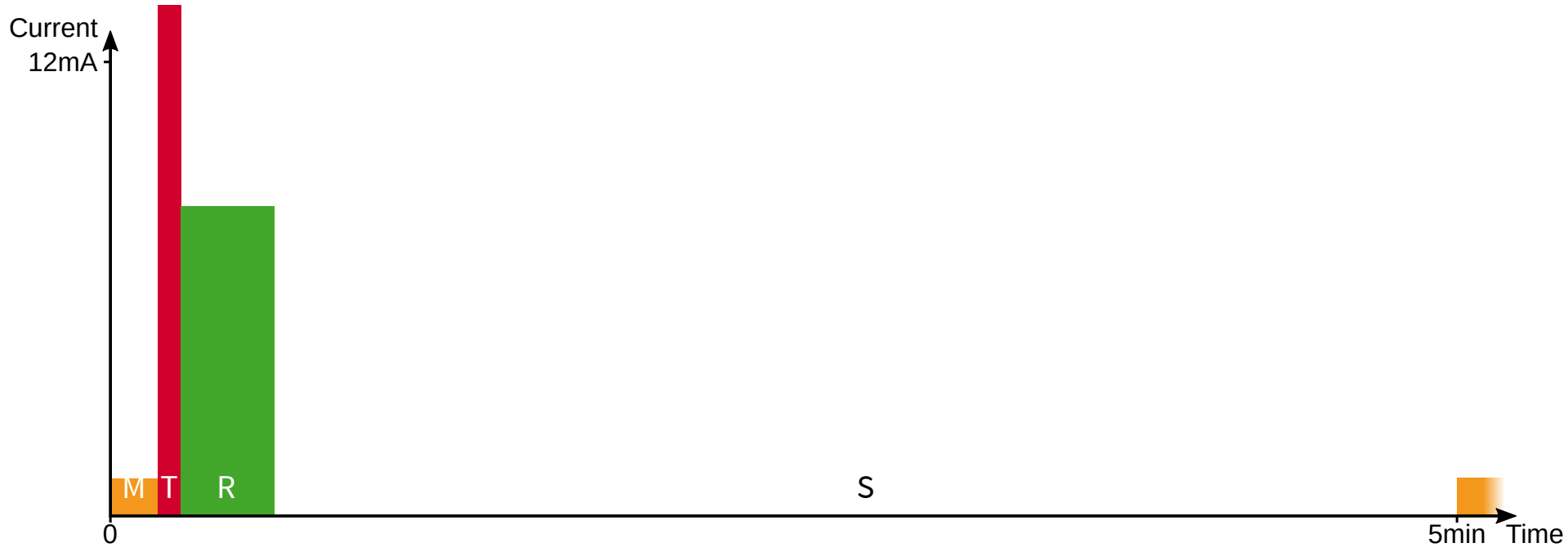
↪ **Battery-powered sensors and actuators are required!**

Example Retrofit Setup: Battery-powered Sensors



- Task: Send notifications when the windows should be opened
- Measure CO_2 contentration in every room
- Send the sensor reading to gateway
- Notify occupier upon high sensor readings via e-mail

The CO_2 Sensor's Application Sequence



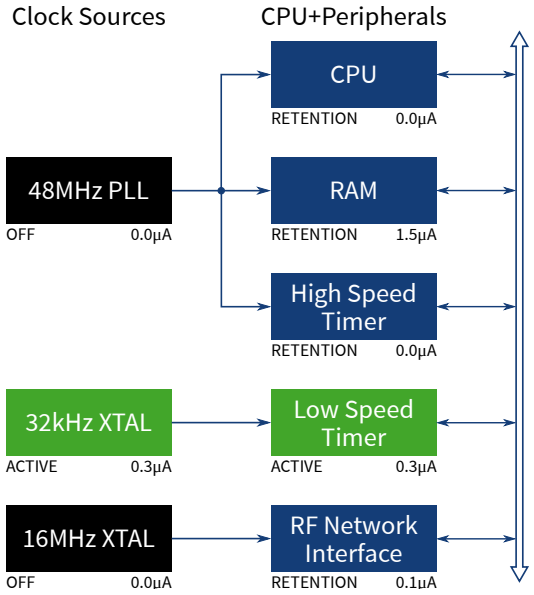
Phase	Duration	Current	Charge
Measure	100 ms	1,000 µA	0.1 mC
TX	50 ms	13,500 µA	0.7 mC
RX	200 ms	8,200 µA	1.6 mC
Sleep	299,650 ms	10 µA	3.0 mC
Sum:	300,000 ms		5.4 mC

Battery Charge: [1]	2,400 mAh = 8,640,000 mC
Battery Self-discharge Current: [2]	1.9 µA
Number of Cycles:	1,444,454
Livetime:	13.74 years

↪ Reduce power consumption during sleep phase!

How does Power Management *Work*?

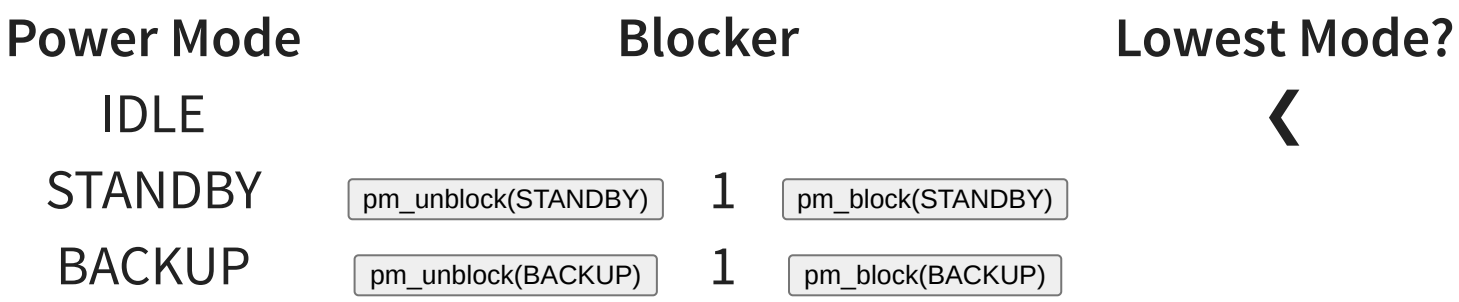
The Internals of the Microcontroller *SAM R30*[3]



SLEEPCFG State	STANDBY
RF Network Interface State	SLEEP
Current Consumption	2.2µA

➥ Set the *SLEEPCFG* register to "STANDBY" and the *RF Network Interface* to "SLEEP" during sleep phase!

RIOT has a Driver for Power Management



- `pm_layered` keeps track of which power mode can be entered
- The idle thread enters the lowest mode
- Someone must tell `pm_layered` which modes are allowed

➥ **For a seamless user experience, drivers must interact with `pm_layered`**

It's all about *Timers*!

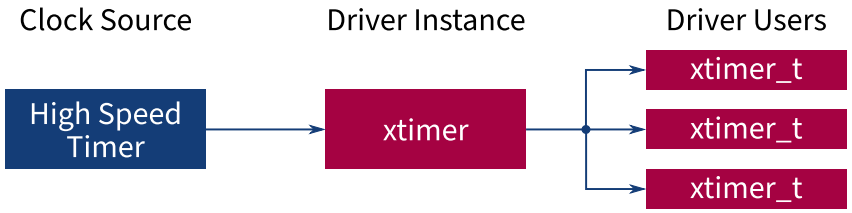
RIOT's Current Default Timer System: `xtimer`

```c
#include "xtimer.h"
#include "timex.h"

static void callback (void * arg) {
        puts((char*) arg);
}

int main (void) {
        /* 1. Run a callback after 3s */
        static xtimer_t cb_timer = {.callback = callback, .arg = "Hello World"};
        xtimer_set(&cb_timer, 3 * US_PER_SEC);

        /* 2. Sleep the current thread for 60s */
        xtimer_sleep(60);
}
```



➥ *xtimer* requires the High Speed Timer to run all the time

➥ STANDBY mode must not be entered at any time

There's an Alternative for the Rescue: `ztimer`

```c
#include "ztimer.h"
#include "timex.h"

static void callback (void * arg) {
        puts((char*) arg);
}

int main (void) {
        /* 1. Run a callback after 3s */
        static ztimer_t cb_timer = {.callback = callback, .arg = "Hello World"};
        ztimer_set(ZTIMER_USEC, &cb_timer, 3 * US_PER_SEC);

        /* 2. Sleep the current thread for 60s */
        ztimer_sleep(ZTIMER_MSEC, 60 * MS_PER_SEC);
}
```

App's Makefile:

```
USEMODULE += ztimer ztimer_usec ztimer_msec ztimer_periph_rtt
USEMODULE += pm_layered
```

samr30-based-board/include/board.h:

```
# Make ZTIMER_USEC block/unblock STANDBY mode
#define CONFIG_ZTIMER_USEC_REQUIRED_PM_MODE PM_SLEEPCFG_SLEEPMOD
# Only block BACKUP mode on startup
#define PM_BLOCKER_INITIAL 0x0001
```

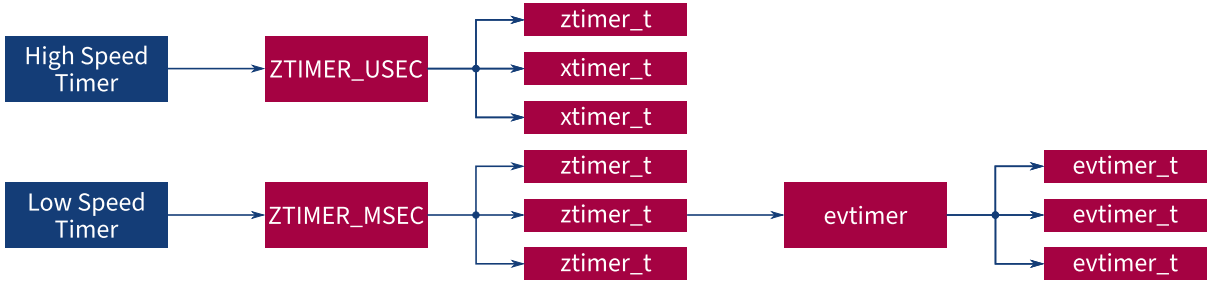
➥ **`ztimer` unblocks STANDBY mode if no `ztimer_t` requires `ZTIMER_USEC` to run**

`xtimer` & `ztimer` can be Friends and Coexist!

```
USEMODULE += ztimer ztimer_usec xtimer xtimer_on_ztimer evtimer
```

➥ `xtimer_on_ztimer` blocks STANDBY mode all the time

```
USEMODULE += ztimer ztimer_usec ztimer_msec ztimer_periph_rtt ztimer_xtimer_compat evtimer evtimer_on_ztimer
# evtimer_on_ztimer hasn't been merged, yet. See Pull Request #13661
```



➥ `ztimer_xtimer_compat` doesn't implement `xtimer_*64()` methods

➥ ztimer is utilized in our use case!

RIOT & Power Management: Status Quo?

Conclusion

RIOT has all important parts for PM inside ...

... but by default they aren't configured for reasonable power saving.

RIOT has three different timer systems ...

... but the *RIOT Developer Memo* could lead to one standard system.
(cf. #12970)

RIOT is heading in the right direction for seamless power management!

References

1. Tadiran Batteries GmbH - Datasheet: SL-860
2. Dittrich, Menachem, Yamin, Adamas - Lithiumbatterien für Funksensornetzwerke
3. Microchip Technology Inc. - SAM R30 Microcontroller