A brief overview of ZephyrOS

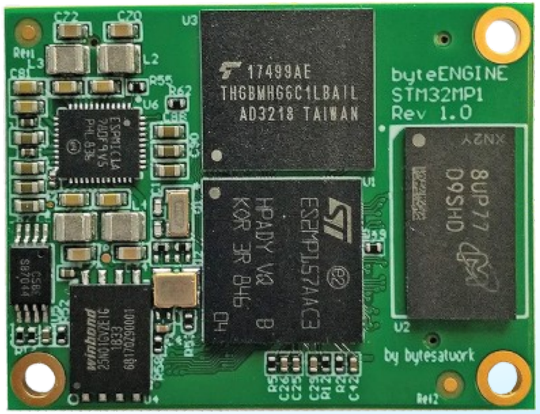
Markus Kappeler
Guy Morand

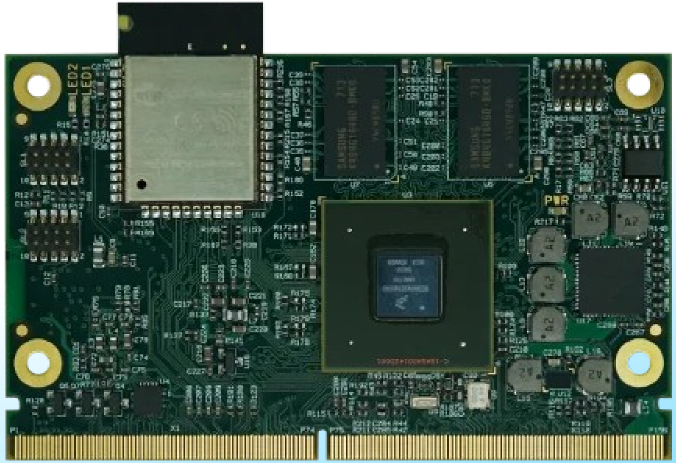
Fribourg Linux Seminar, 1st December 2022

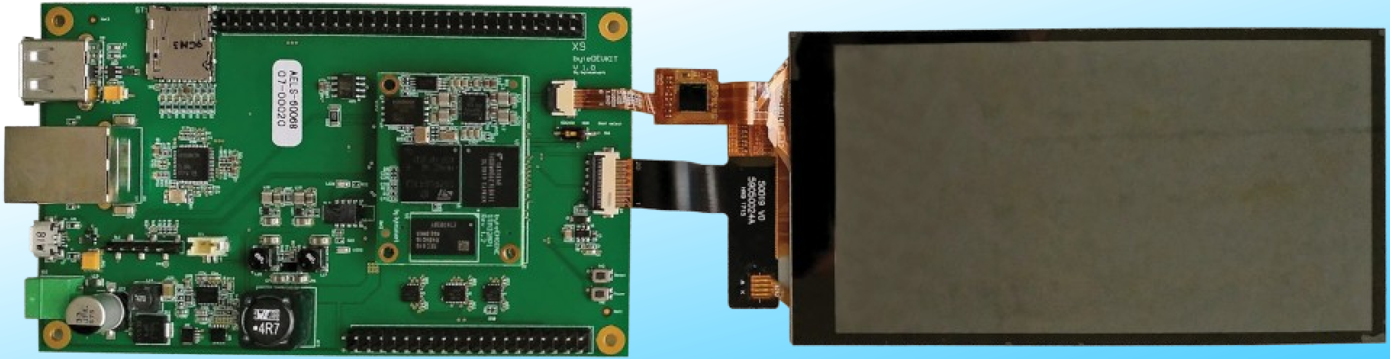
We are bytesatwork

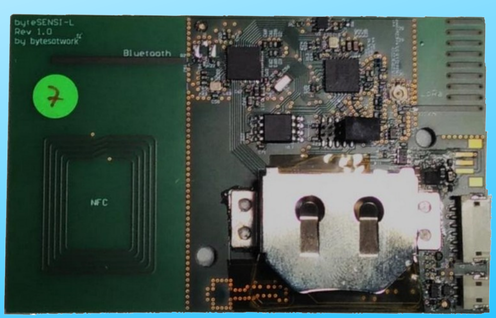
- Markus Kappeler, CEO
- Guy Morand, Software Engineer

We do hardware









bytesatwork

We do software

yocto
PROJECT
Qt
Zephyr™
debian
C++

We are hiring

- **Location:** Winterthur
- **Home office:** max. 2 days / week
- Zephyr-RTOS und embedded Linux (80-100%)
- https://www.bytesatwork.io/jobs/

Agenda

- Zephyr project
- west build system
- Device driver model
- Kconfig
- Demo
- Examples of application
- Advantages / Disadvantage

The Zephyr Project

- Open source real time OS (Apache License V2)
- Supported by the Linux foundation
- Very active and growing community
- Many supported platforms (>400 boards)
- Very well documented

THE LINUX FOUNDATION

Zephyr project members (2022)

Platinum Members

- antmicro
- Baumer
- Google
- intel
- Meta
- NORDIC SEMICONDUCTOR
- NXP
- oticon life-changing technology
- Qualcomm Innovation Center
- T Mobile

Silver Members

- AVSYSTEM
- BayLibre
- FOUNDRIES.IO
- Golioth
- infineon
- Laird CONNECTIVITY
- Linaro
- Memfault
- PARASOFT
- percepio
- SiFive
- SILICON LABS
- SYNOPSYS
- TEXAS INSTRUMENTS
- WNDRVR

Zephyr Ecosystem

Zephyr OS

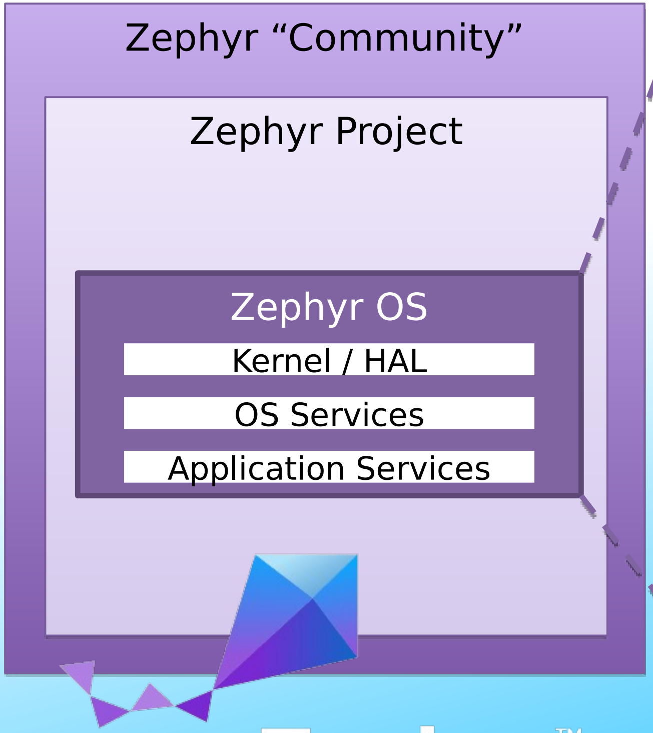
- Kernel und HAL
- OS Services, IPC, Logging, file systems, crypto

Zephyr Project

- SDK
- Middleware
- Device Management
- Bootloader

Zephyr Community

- 3rd Party Module and libraries
- Support for Zephyr in 3rd party projects: Jerryscript, Micropython, Iotivity



Zephyr “Community”

Zephyr Project

Zephyr OS

Kernel / HAL

OS Services

Application Services

Kernel / HAL

- Scheduler
- Kernel objects and services
- low-level architecture and BSP
- Power Management and low level hardware interfaces

OS Services and Low level APIs

- Platform specific driver
- Generic I/O API
- File systems, Logging, Debugging and IPC
- Cryptography Services
- Networking and Connectivity

Application Services

- High Level APIs
- Standardized data model
- High Level network protocolsl

west build system

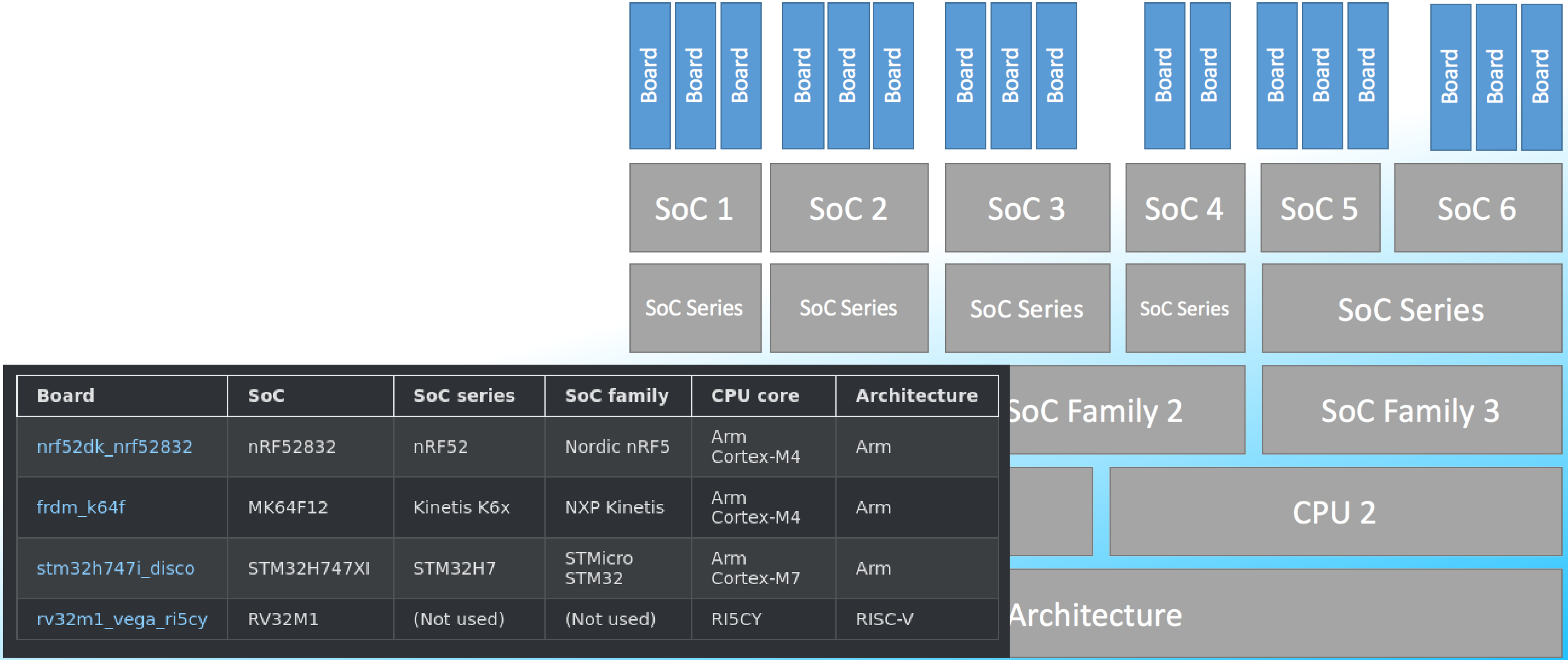
- Python tool to facilitate
  - Project initialization
  - Building
  - flashing
  - Debugging
- Using an external toolchain is also possible
- Windows and MacOS compatible

CMake
GCC
OpenOCD

Typical west workflow

```
west init -m <repository-URL>
west update
source zephyr/zephyr-env.sh
west build -b <board> <application-path>
west flash
west debug
```

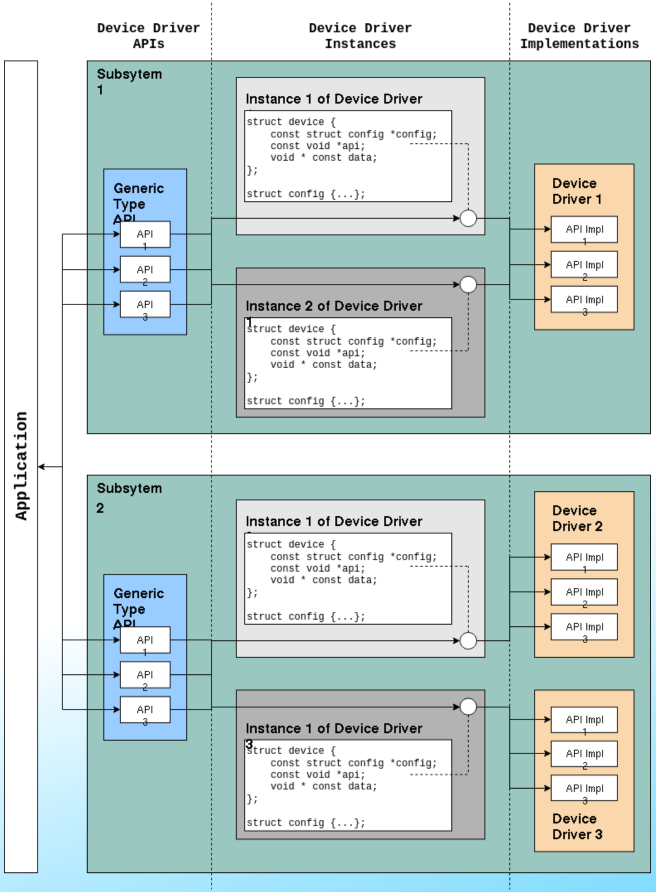
Device driver model



Board	SoC	SoC series	SoC family	CPU core	Architecture
nrf52dk_nrf52832	nRF52832	nRF52	Nordic nRF5	Arm Cortex-M4	Arm
frdm_k64f	MK64F12	Kinetis K6x	NXP Kinetis	Arm Cortex-M4	Arm
stm32h747i_disco	STM32H747XI	STM32H7	STMicro STM32	Arm Cortex-M7	Arm
rv32m1_vega_ri5cy	RV32M1	(Not used)	(Not used)	RI5CY	RISC-V

bytesatwork

Device driver model

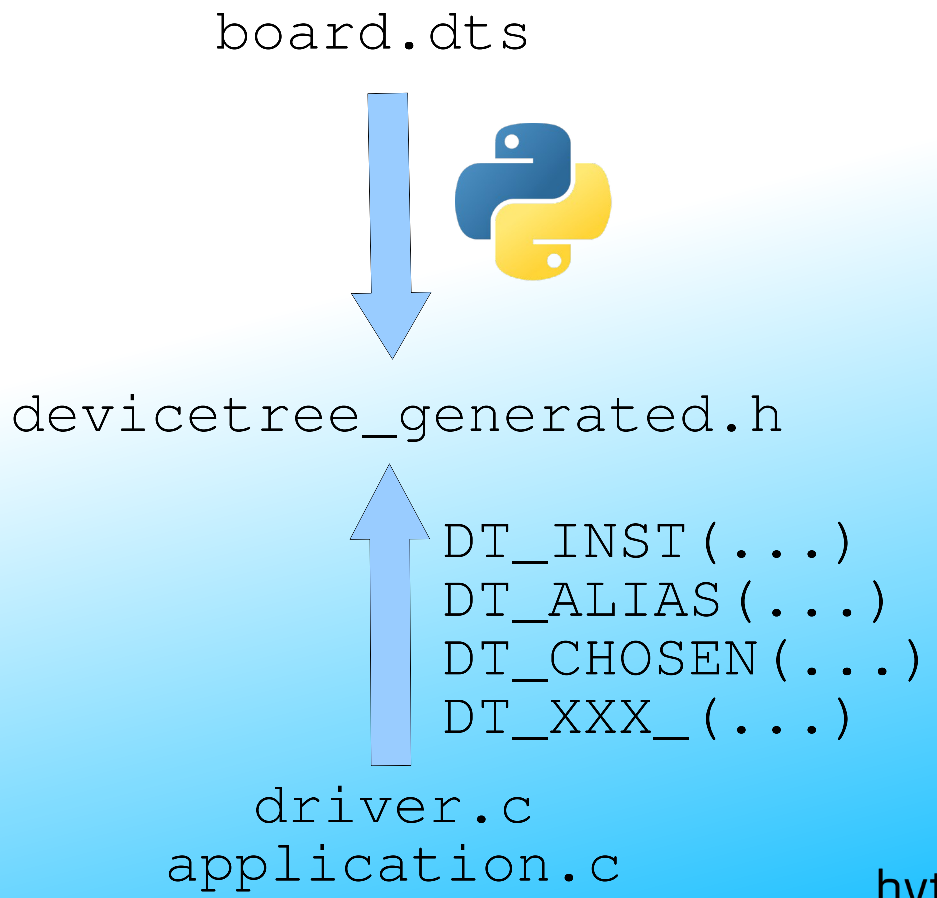


```
struct device {
    const struct config *config;
    const void *api;
    void * const data;
};

struct config {...};
```

```
struct device {
    const struct config *config;
    const void *api;
    void * const data;
};

struct config {...};
```

```
struct device {
    const struct config *config;
    const void *api;
    void * const data;
};

struct config {...};
```

```
struct device {
    const struct config *config;
    const void *api;
    void * const data;
};

struct config {...};
```



`board.dts`

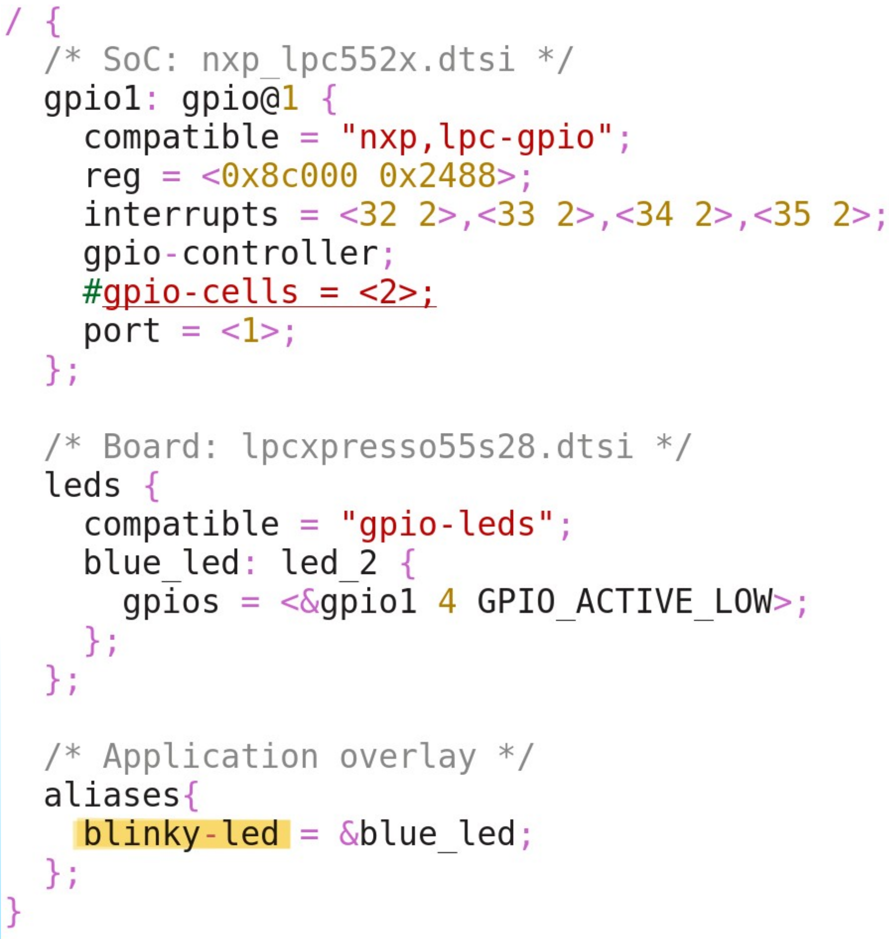
`devicetree_generated.h`

```
DT_INST(...)
DT_ALIAS(...)
DT_CHOSEN(...)
DT_XXX_(...)
```

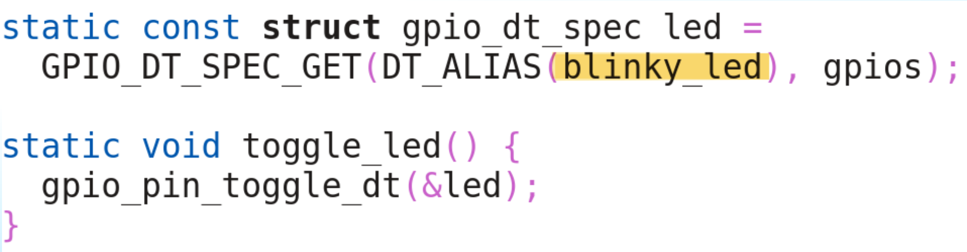
```
driver.c
application.c
```

bytesatwork

Device driver model



```
/ {
  /* SoC: nxp_lpc552x.dtsi */
  gpio1: gpio@1 {
    compatible = "nxp,lpc-gpio";
    reg = <0x8c000 0x2488>;
    interrupts = <32 2>,<33 2>,<34 2>,<35 2>;
    gpio-controller;
    #gpio-cells = <2>;
    port = <1>;
  };

  /* Board: lpcxpresso55s28.dtsi */
  leds {
    compatible = "gpio-leds";
    blue_led: led_2 {
      gpios = <&gpio1 4 GPIO_ACTIVE_LOW>;
    };
  };

  /* Application overlay */
  aliases{
    blinky-led = &blue_led;
  };
}
```



```
static const struct gpio_dt_spec led =
  GPIO_DT_SPEC_GET(DT_ALIAS(blinky_led), gpios);

static void toggle_led() {
  gpio_pin_toggle_dt(&led);
}
```

Kconfig

- Python re implementation of kernel config
- Allows:
  - Enabling features
  - Changing configurations
- Can be overridden in prj.conf

Kconfig: Typical usage

<app>/Kconfig

```
config BLINK_INTERVAL_MS
  int "Blink interval"
  default 100
  help
    Blink interval in milliseconds

source "Kconfig.zephyr"
```

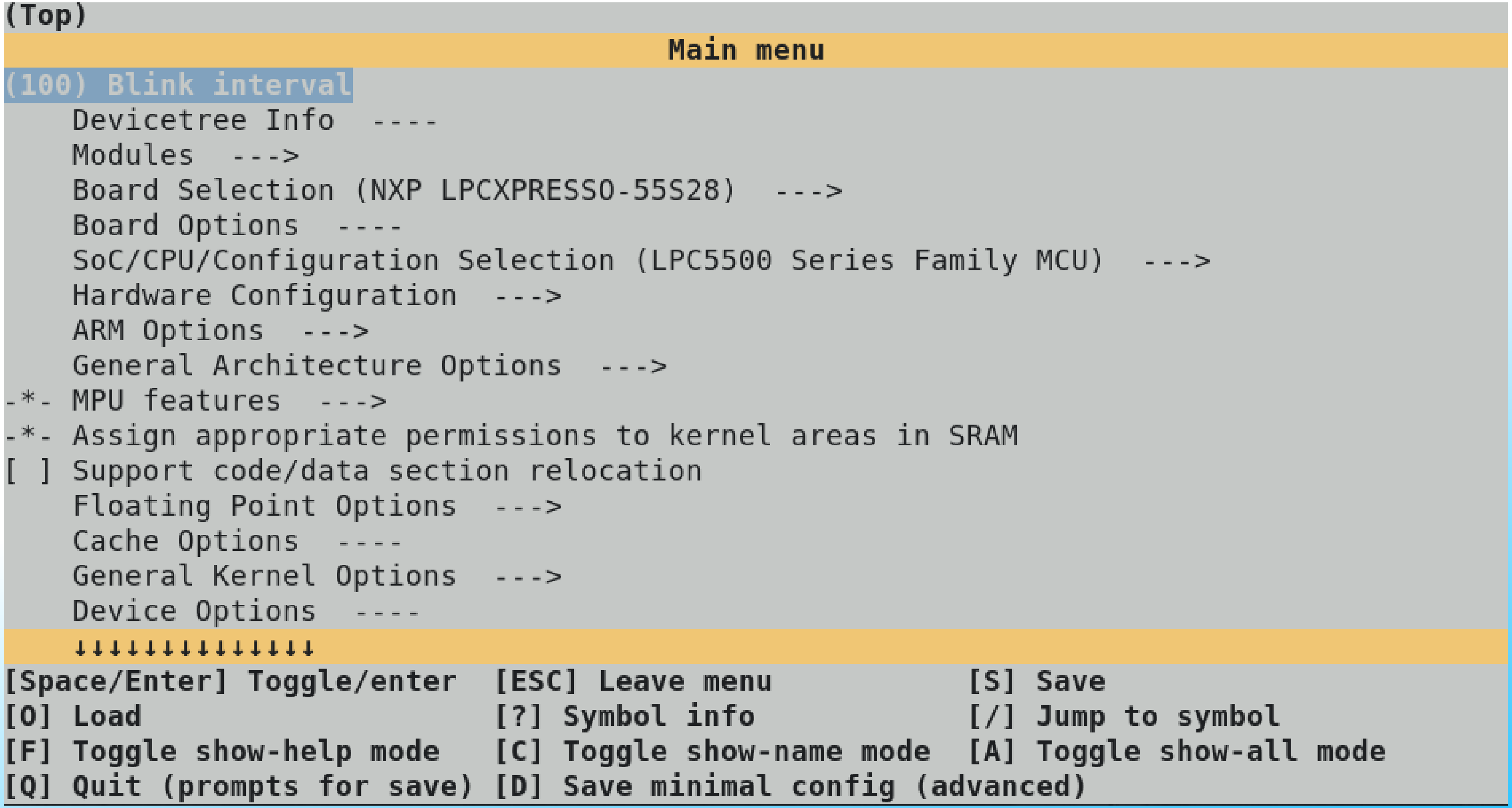
<app>/prj.conf

```
CONFIG_LOG=y
CONFIG_LOG_DEFAULT_LEVEL=4

CONFIG_BLINK_INTERVAL_MS=500
```

main.c

```
static const int blink_interval_ms = CONFIG_BLINK_INTERVAL_MS;
```

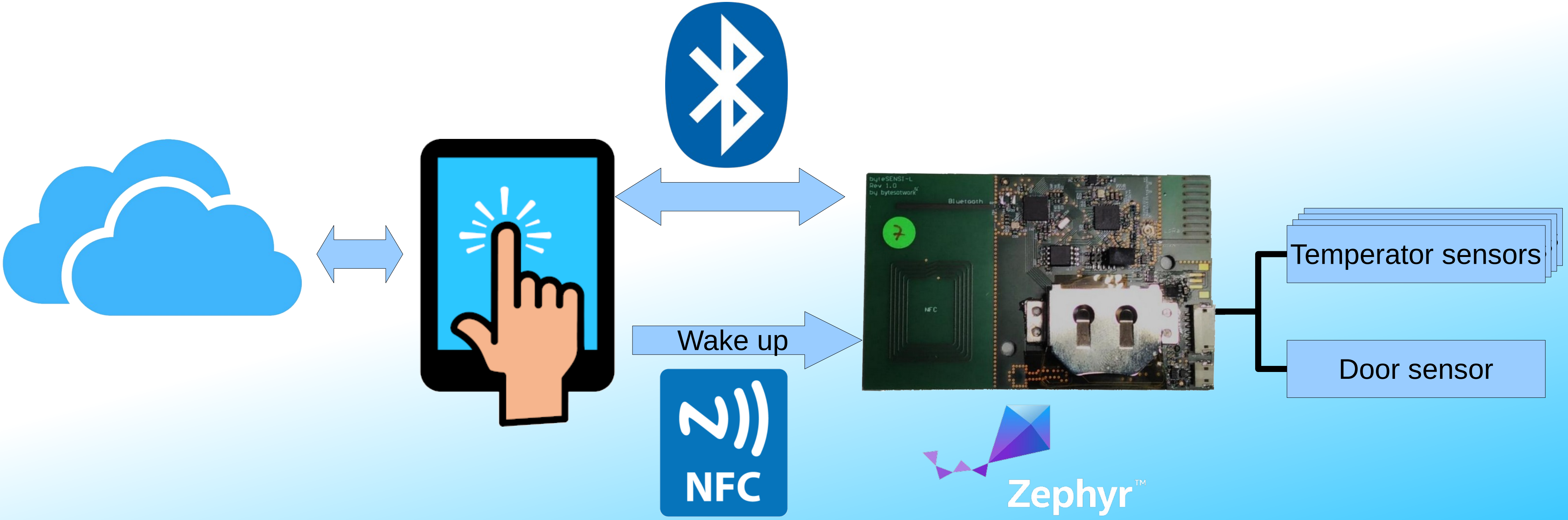
Kconfig: ncurses frontend



```
(Top)
                                    Main menu
(100) Blink interval
    Devicetree Info  ----
    Modules  --->
    Board Selection (NXP LPCXPRESSO-55S28)  --->
    Board Options  ----
    SoC/CPU/Configuration Selection (LPC5500 Series Family MCU)  --->
    Hardware Configuration  --->
    ARM Options  --->
    General Architecture Options  --->
-*- MPU features  --->
-*- Assign appropriate permissions to kernel areas in SRAM
[ ] Support code/data section relocation
    Floating Point Options  --->
    Cache Options  ----
    General Kernel Options  --->
    Device Options  ----
    ↓↓↓↓↓↓↓↓↓↓↓↓↓↓
[Space/Enter] Toggle/enter  [ESC] Leave menu           [S] Save
[O] Load                    [?] Symbol info            [/] 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)
```

Demo

Temperature logger

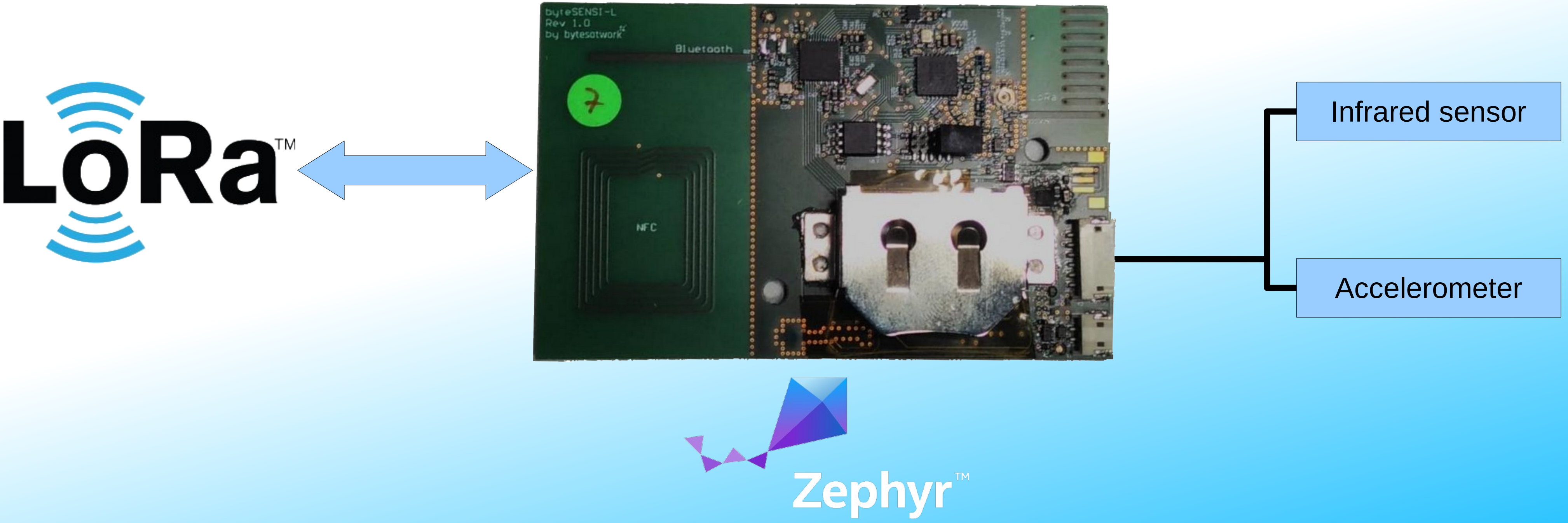


Wake up

Temperator sensors

Door sensor

Manipulation detector



Infrared sensor

Accelerometer

Advantages

- Excellent hardware abstraction
- SDKs runs on Linux, Windows and MacOS
- Independent from hardware manufacturers
- Permissive Apache V2 License
- Supports a lot of SoCs out of the box
- Excellent connectivity (Bluetooth, LoRa, MQTT, ...)
- Power efficient

Disadvantages

- API is still unstable from version to version
- Cryptic compilation errors due to static allocations with device tree macros
- No more web server!
  - CivetWeb support was dropped
- Build system too big and bloated?

Not covered

- Adding new boards / SoC / drivers
- Bootloader
- Real time performances
- Twister test framework
- Profiling and tracing
- Connectivity
- Contributing
- …

Links

- https://www.zephyrproject.org/
- https://docs.zephyrproject.org/
- https://www.bytesatwork.io/
- https://github.com/morandg/zephyr-blinky-advanced

Thanks for your attention

bytesatwork