Hands-on Zephyr Project Workshop

Navigating Low Power IoT Development with Practical Examples

Jonas Remmert

Table of Contents

Introduction

Introduction

- **Jonas Remmert**
- SMIGHT GmbH
- Developer for Low Power IoT products
- **Experience:** RTOS (FreeRTOS, Zephyr), NXP MCUX, hardware development

Workshop Goals

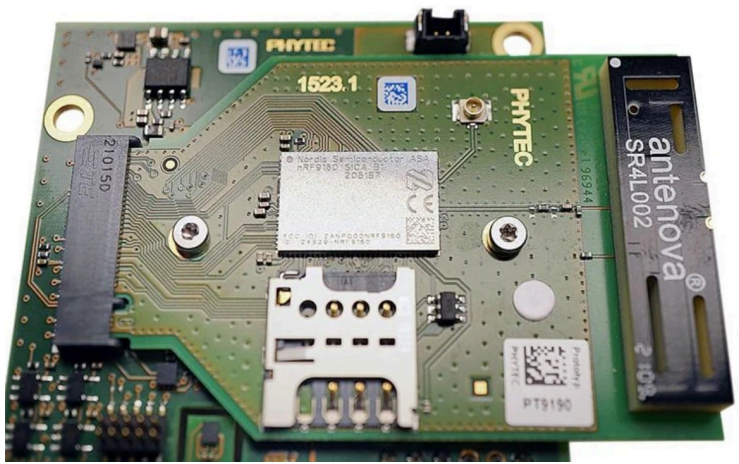
- Combine theory with hands-on practice
- Introduction to Zephyr RTOS
- Setting up the SDK
  - On a Local Machine
  - GitHub Codespaces
- Exploring key features
- Hands-on development

Audience Check

- What are your goals for the Workshop?
- Have you used an RTOS before?
- Your experience with Zephyr?

Areas of Work

- **Hardware:** Customer-specific hardware solutions, focus on low-power embedded systems
- **nRF9160 SoC:** Development of IoT applications using the Nordic nRF9160 SoC for both hardware and firmware
- **Zephyr:** Contributor to the Zephyr Project
- **Example:** Pump Monitor, developed in collaboration with **BeST Berliner Sensortechnik GmbH**



Pump Monitor for BeST Berliner Sensortechnik GmbH

Open Source and Vendor-Neutral Governance

- Governed by the Linux Foundation
- Vendor-neutrality: fairness and interoperability
- Technical Steering Committee (TSC) and Working Groups (WG)
- Security and tooling (e.g. SBOM) as integral part
- **Alternative to vendor SDKs**
- **Safety certification ongoing**

The Linux Foundation logo

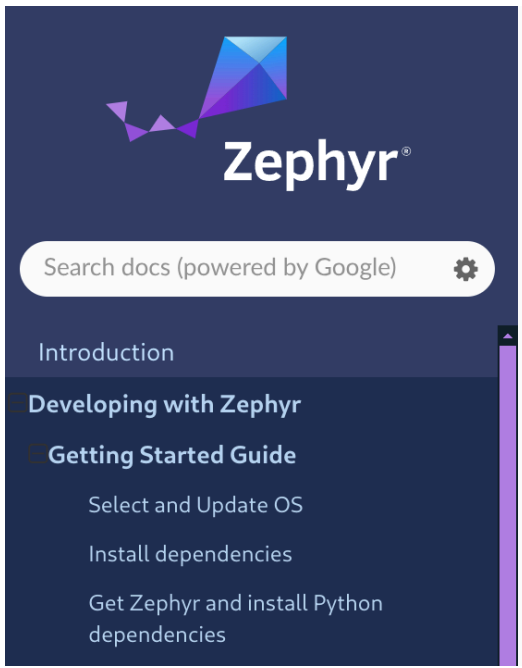
Zephyr for Products

Examples

- Overview: zephyrproject.org/products-running-zephyr
- Wildlife Tracking and Protection (OpenCollar)
- Wind Turbines (Vestas)
- Irrigation (Gardena)
- Hearing Aid (Oticon)
- Wastewater Pump Monitoring (BeST Berliner Sensortechnik, German Railways - DB)

Development Setup

Getting Started Guide



Getting Started Guide

Follow this guide to:

- Set up a command-line Zephyr development environment on Ubuntu, macOS, or Windows (instructions for other Linux distributions are discussed in Install Linux Host Dependencies)
- Get the source code
- Build, flash, and run a sample application

Zephyr Getting Started Guide[1]

[1] docs.zephyrproject.org/latest/getting_started/index.html

Setting up a Local Development Environment

Linux, macOS and Windows[1] are supported!

Overview of Components

Packages

- git
- Cmake
- Python ...

Zephyr SDK

- Toolchains for different architectures

Python Tools

- west, requirements.txt

IDE

- Text editor, IDE (e.g. VSCode)

Repositories

- Zephyr
- Modules via west

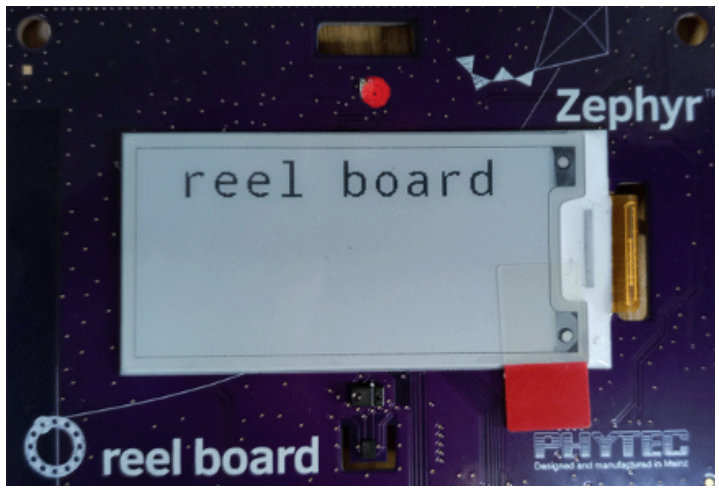
[1]Windows is supported, but will complicate development for Embedded Systems (e.g. no onboard package manager, often inconsistent Python environment setups).

Testing the Local Development Environment

Build and flash the application with `west`

```
cd ~/zephyrproject/zephyr
west build -b reel_board samples/basic/blinky -p
west flash
```

```
cd ~/zephyrproject/zephyr
west build -b native_sim samples/hello_world/ -p
west build -t run

*** Booting Zephyr OS build v4.1.0 ***
Hello World! native_sim/native
```

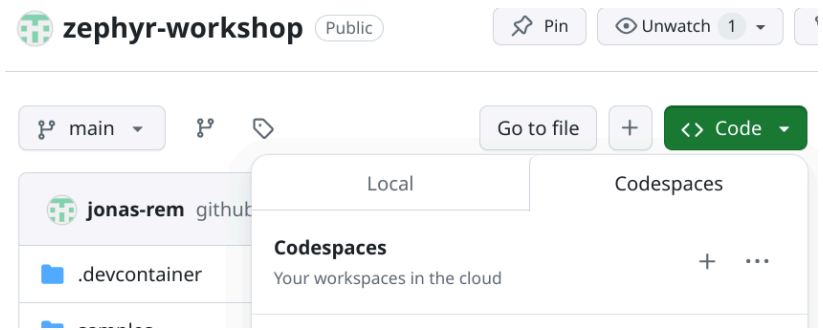


reel board with blinking LED

Starting a Cloud Development Environment

GitHub Codespaces[1]

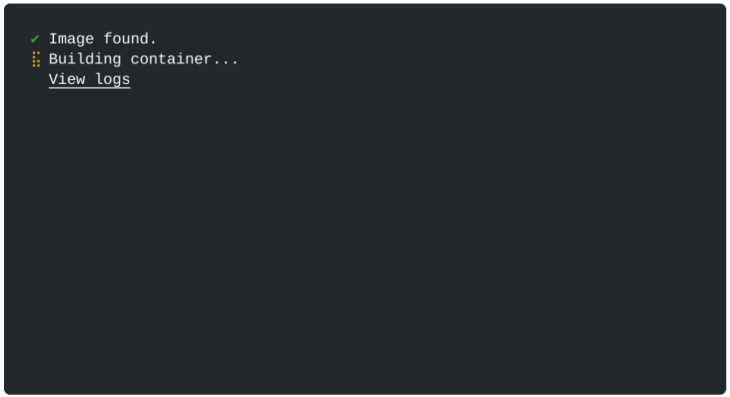
- Cloud hosted development environment based on devcontainers
- Visual Studio Code integration
- Runs on Microsoft Azure cloud
- Configuration individual for each repository



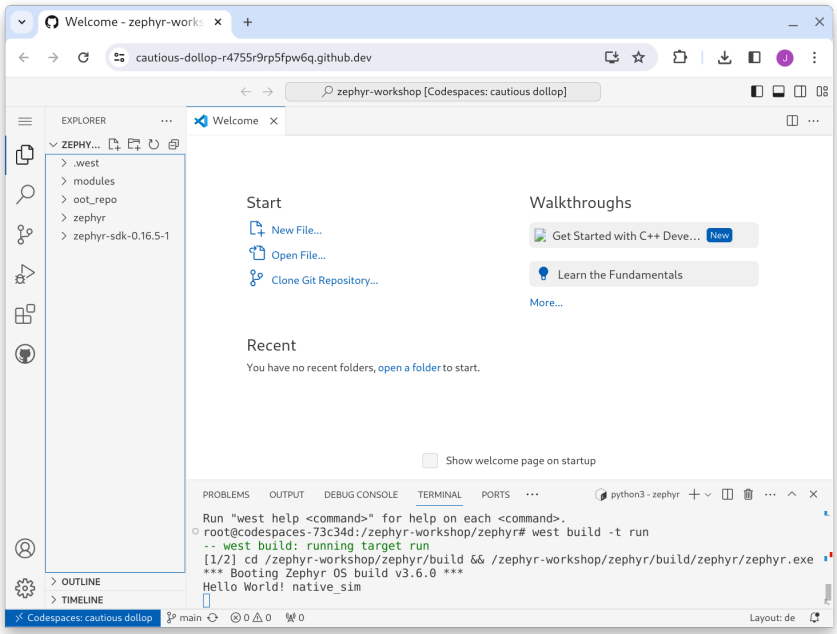
Create a new Codespace

[1]Open Source Alternatives: Gitpod, Eclipse Theia and many more

Active Instance of GitHub Codespace



Codespaces starting..



Codespaces in a Browser Window

Recommendations from Experience

Virtual Machines in combination with embedded hardware can bring their own problems.

Prioritize a local environment over a cloud environment

- Hardware is better accessible
- Better integration of your own tools
- Check vendor tools that can enhance your Zephyr Dev Environment

Hands-on 1 - Codespaces Setup

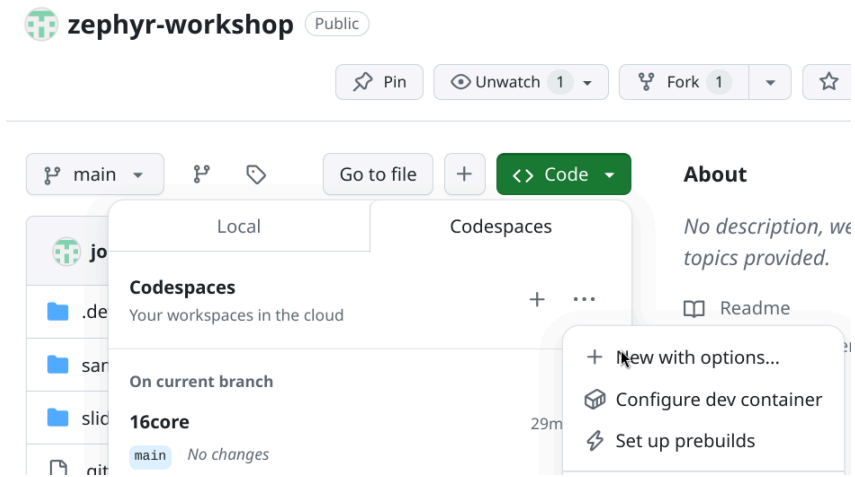
Start your own Codespaces Instance now!

github.com/jonas-rem/zephyr-workshop

Setup will take a few minutes...

Test your setup with the Hello World example:

```
west build -b native_sim zephyr/samples/hello_world -p
west build -t run
```



Setup new Instance

[1] **Recommendation:** Use a 4-core setup instead of the 2-core default.

[2] **Note:** You should have 120 core-hours per month free.

Workspace Application and Hardware Abstraction

Workspace Application

Workspace application or **Out of tree** build[1]

Separate application from Zephyr repository

- Independent version control for app
- Different Licences for app?
- Maintain references to Zephyr, Modules etc. in app repo
- Update Zephyr version independently from app

```
zephyrproject
├── modules
│   └── hal
├── zephyr
│   ├── samples
│   ├── west.yml
└── zephyr-workshop
    ├── west.yml
    └── samples
        └── 01_hello_world
            ├── CMakeLists.txt
            ├── prj.conf
            ├── README.rst
            ├── sample.yaml
            └── src
```

[1] docs.zephyrproject.org/latest/develop/application

[2] github.com/zephyrproject-rtos/example-application

[3] github.com/jonas-rem/zephyr-workshop

Enabled by *west* and Manifest files

Manifests references repositories and modules[1]

```yaml
manifest:
  remotes:
    - name: zephyrproject-rtos
      url-base: https://github.com/zephyrproject-rtos

  projects:
    - name: zephyr
      remote: zephyrproject-rtos
      revision: v3.6.0
      import:
        name-allowlist:
          - cmsis
          - hal_nordic
          - hal_nxp
          - [..]
```

west.yaml manifest file in the zephyr-workshop repository

[1] docs.zephyrproject.org/latest/develop/west/manifest.html

Understanding and Using *west*

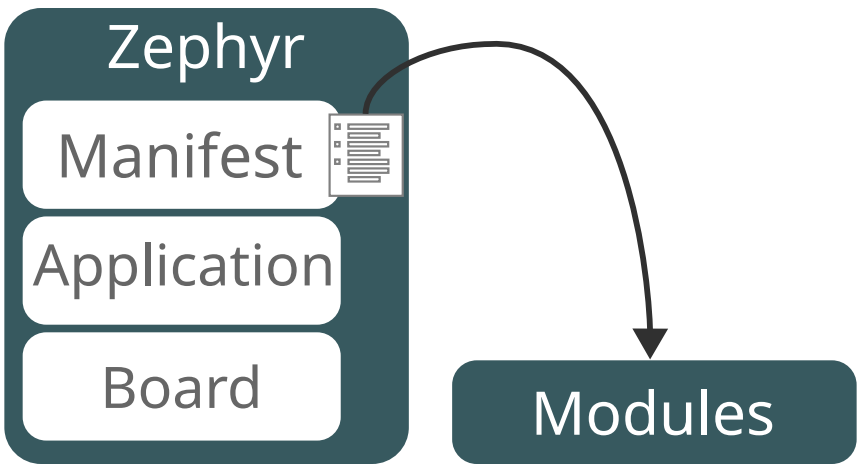
`west` repository management tool, developed by the Zephyr community

- Inspired by Google's Repo tool and git submodules
- Cloning Zephyr repo, dependencies, modules
- Keeping project repositories synchronized
- In addition building, flashing, and debugging support

```
# Navigate to the project root
$ cd zephyrproject

# Update all repositories
$ west update

=== updating zephyr (zephyr):
HEAD is now at 468eb56cf24 [..]
=== updating cmsis (modules/hal/cmsis):
HEAD is now at 4b96cbb [..]
=== updating hal_atmel (modules/hal/atmel):
HEAD is now at aad79bf [..]
```

Application Structure - Use Case I

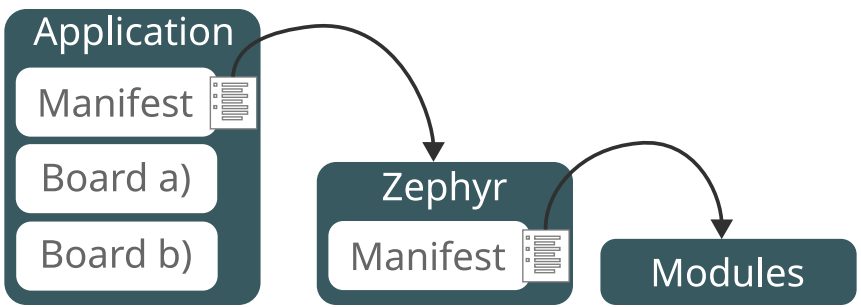
- **Who:** User creates one application for testing
- **What:** One variant of an application
- **Solution:** Make changes inside the Zephyr tree for simplicity[1]



[1]Not recommended for production, use an out-of-tree build instead. This makes it easier to upgrade to more recent Zephyr versions.

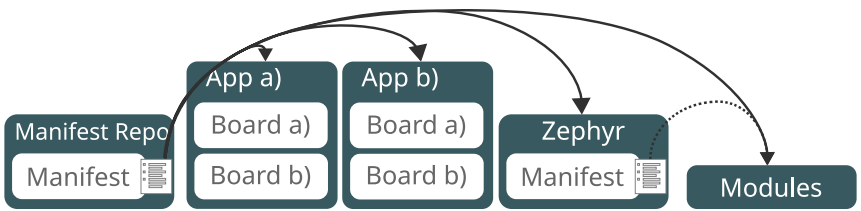
Application Structure - Use Case II

- **Who:** One company developing one product
- **What:** Two variants of the product
- Different sensors, pin assignments, but similar application
- **Solution:** Devicetrees for each variant
  - board_a.dts: `west build -b board_a app`
  - board_b.dts: `west build -b board_b app`



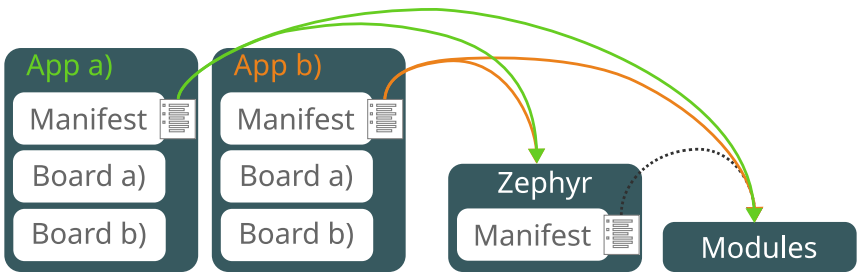
Application Structure - Use Case III

- **Who:** One company developing multiple products
- **What:** Different applications
- **Solution:** Separate applications
- Use same Zephyr version for all applications if you can



Application Structure - Use Case IV

- **Who:** Service provider developing different products for multiple companies
- **What:** Development states and lifecycle for products differ
- **Solution:** Individual manifest files for each product
- Create your own modules, share code inbetween projects
- Quickly setup and reference projects with west



Zephyr Hardware Abstraction

- **Vendor HALs:** Hardware abstraction available from vendors. Abstracted via Zephyr APIs and drivers
- **Devicetree:** Decouples the application from the hardware
- **Architecture:** ARM, RISC-V, x86, ARC, NIOS II, Tensilica, Xtensa
- **Other:** 600+ boards, 180+ sensors

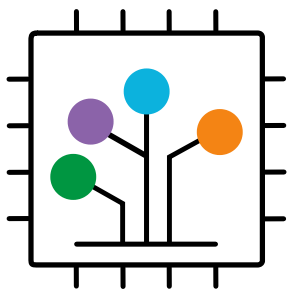
```
zephyrproject/zephyr:~$ ls arch/
arc    CMakeLists.txt  mips   riscv  xtensa
arm    common          nios2  sparc
arm64  Kconfig         posix  x86

zephyrproject:~$ ls modules/hal/
altera        espressif   nordic      silabs
ambiq         ethos_u     nuvoton     st
atmel         gigadevice  nxp         stm32
cirrus-logic  infineon    openisa     telink
[..]

zephyrproject/zephyr:~$ ls boards/
96boards              firefly       native_sim
actinius              gd            rak
...
```

Zephyr Hardware Abstraction - devicetree

Describes the available hardware

- Proven concept used in the Linux kernel
- Single source for hardware information
- Drivers and source are hardware independent

In Zephyr: C header generation at compile time



devicetree Logo

[1] devicetree.org

Zephyr Hardware Abstraction - devicetree

```
arduino_i2c: &i2c0 {
        compatible = "nordic,nrf-twim";
        status = "okay";
        clock-frequency = <I2C_BITRATE_FAST>;
        pinctrl-0 = <&i2c0_default>;
        pinctrl-1 = <&i2c0_sleep>;
        pinctrl-names = "default", "sleep";

        mma8642fc: mma8652fc@1d {
                compatible = "nxp,fxos8700","nxp,mma8652fc";
                reg = <0x1d>;
                int1-gpios = <&gpio0 24 GPIO_ACTIVE_LOW>;
                int2-gpios = <&gpio0 25 GPIO_ACTIVE_LOW>;
        };

        ti_hdc@43 {
                compatible = "ti,hdc","ti,hdc1010";
                reg = <0x43>;
                drdy-gpios = <&gpio0 22 (GPIO_ACTIVE_LOW | GPIO_PULL_UP)>;
        };
};
```

devicetree node for the reel board: `boards/phytec/reel_board/dts/reel_board.dtsi`

Hands-on 2: Let's build Zephyr for the reel board!

Build the `blinky` sample:

```
west build -b reel_board \
  ../zephyr/samples/basic/blinky -p
```

Flash via Drag and Drop:

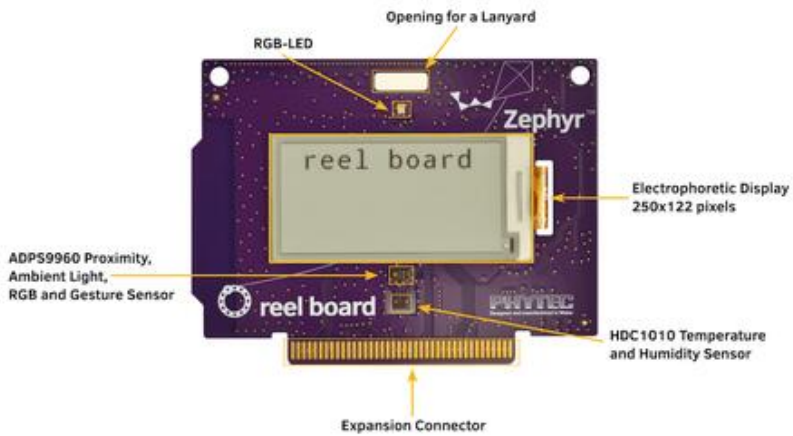
- Download the binary from your Codespace at: `build/zephyr/zephyr.hex`
- Drag and drop the hex file to the mounted reel board mass storage device

Flash via native setup:

```
west flash
```

Attach to the serial console:

```
minicom -D /dev/ttyACM0 -b 115200
```



reel board

Code Examples and Subsystems

Samples in Zephyr

Samples

- Zephyr provides a wide range of samples
- Samples are located in `zephyr/samples/`
- Isolated functionality or feature

Tests

- Tests are located in `zephyr/tests/`
- Isolated test cases for a feature or hardware
- Useful to test e.g. a device driver

Applications

- Application Example: github.com/zephyrproject-rtos/example-application
- ZSWatch - Open Source Smart Watch: github.com/jakkra/ZSWatch

01_hello_world

Description:

- Simple Hello World program[1]

Learn:

- Test setup
- Structure of a Zephyr application

Build and run:

```
west build -b native_sim samples/01_hello_world -p
west build -t run
```

```
samples/01_hello_world/
├── CMakeLists.txt
├── prj.conf
├── README.rst
├── sample.yaml
├── 01_hello_world
└── src
    └── main.c
```

[1] Equivalent in the Zephyr main Repository: zephyr/samples/hello_world.

01_hello_world Configuring the Build System

```
# SPDX-License-Identifier: Apache-2.0

cmake_minimum_required(VERSION 3.20.0)

find_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR_BASE})
project(hello_world)

target_sources(app PRIVATE src/main.c)
```

`samples/01_hello_world/CMakeLists.txt`

01_hello_world Application Source Code

```c
/*
 * Copyright (c) 2012-2014 Wind River Systems, Inc.
 *
 * SPDX-License-Identifier: Apache-2.0
 */

#include <zephyr/kernel.h>

int main(void)
{
    printk("Hello World! %s\n", CONFIG_BOARD);
    return 0;
}
```

`samples/01_hello_world/src/main.c`

01_hello_world Application Build Output

```
west build -b native_sim samples/01_hello_world/ -p
-- Found host-tools: zephyr 0.17.0 (/home/jonas/zephyr-sdk-0.17.0)
-- Found toolchain: zephyr 0.17.0 (/home/jonas/zephyr-sdk-0.17.0)
[..]
Parsing /home/jonas/git/zephyrproject/zephyr/Kconfig
-- The C compiler identification is GNU 12.2.0
-- The CXX compiler identification is GNU 12.2.0
-- The ASM compiler identification is GNU
-- Found assembler: /home/jonas/zephyr-sdk-0.17.0/arm-zephyr-eabi/bin/...
-- Configuring done (3.0s)
-- Generating done (0.1s)
-- Build files have been written to: .../zephyr-workshop/build
-- west build: building application
[1/117] Preparing syscall dependency handling

[2/117] Generating include/generated/zephyr/version.h
-- Zephyr version: 4.1.0 (/home/jonas/git/zephyrproject/zephyr), build: v4.1.0
[117/117] Linking C executable zephyr/zephyr.elf
Memory region         Used Size  Region Size  %age Used
           FLASH:        8094 B       256 KB      3.09%
             RAM:           4 KB        64 KB      6.25%
        IDT_LIST:           0 GB        32 KB      0.00%
```

01_hello_world Build Artifacts

Build Location:

- `build/`

Executable Location:

- `build/zephyr/`

Artifacts:

- `zephyr.elf|hex|bin`
- `zephyr.map`
- `autoconf.h` (Kconfig options)
- `devicetree_generated.h` (Generated devicetree header)
- `Kconfig.dts` (devicetree conf)

```
build/
├── app
│   └── libapp.a
├── Kconfig
│   └── Kconfig.dts
└── zephyr
    ├── include
    │   └── generated
    │       └── zephyr
    │           ├── autoconf.h
    │           └── devicetree_generated.h
    ├── zephyr.dts
    ├── zephyr.elf|bin|hex
    ├── zephyr_final.map
```

01_hello_world Sample - Console Output

```
*** Booting Zephyr OS build v4.1.0 ***
Hello World! native_sim
```

02_logging Sample

Description:

- Logging subsystem example[1]

Learn:

- Log levels
- Set loglevel via Kconfig
- Logging backends, e.g. filesystem, BLE
- Log messages printed in own thread when system is idle

Sample:

- Demonstrates logging output

```
CONFIG_LOG=y
```

samples/02_logging/prj.conf

```c
#include <zephyr/logging/log.h>

LOG_MODULE_REGISTER(hello_world, LOG_LEVEL_DBG);

int main(void)
{
    LOG_INF("info string");

    return 0;
}
```

samples/02_logging/src/main.c

[1] Equivalent in the Zephyr main Repository: zephyr/samples/subsys/logging/logger.

02_logging Sample - Console Output

```
*** Booting Zephyr OS build v4.1.0 ***
Hello World! native_sim
[00:00:00.001,691] <err> hello_world: error string
[00:00:00.001,843] <dbg> hello_world: main: debug string
[00:00:00.001,859] <inf> hello_world: info string
[00:00:00.001,874] <dbg> hello_world: main: int8_t 1, uint8_t 2
[00:00:00.001,887] <dbg> hello_world: main: int16_t 16, uint16_t 17
[00:00:00.001,899] <dbg> hello_world: main: int32_t 32, uint32_t 33
[00:00:00.001,921] <dbg> hello_world: main: int64_t 64, uint64_t 65
[00:00:00.001,956] <dbg> hello_world: main: char !
[00:00:00.002,383] <dbg> hello_world: main: s str static str c str
[00:00:00.002,567] <dbg> hello_world: main: d str dynamic str
[00:00:00.002,607] <dbg> hello_world: main: mixed str dynamic str --- ...
[00:00:00.002,640] <dbg> hello_world: main: mixed c/s ! static str ...
[00:00:00.002,653] <dbg> hello_world: main: pointer 0x5c3e
[00:00:00.002,674] <dbg> hello_world: main: HeXdUmP!
                                     48 45 58 44 55 4d 50 21  20 23 40  |HEXDUMP!  #@
```

03_workqueues Sample

Description:

- Workqueue and Timer Example

Learn:

- Workqueue, Timers, Runtime Contexts (IRQ, Thread)
- Queue of work items
- Work items are executed in a thread context
- Timer is used to schedule work items
- System workqueue is enabled by default

Sample:

- Executes a function in different contexts

03_workqueues Sample - Console Output

```
*** Booting Zephyr OS build v4.1.0 ***
Work Item Executed - runtime context:
 Thread Name: main
 Thread Priority: 0

Work Item Executed - runtime context:
 Thread Name: sysworkq
 Thread Priority: -1

Work Item Executed - runtime context:
 Thread Name: my_work_q_thread
 Thread Priority: 5

Timer Expired!!
Work Item Executed - runtime context:
 ISR Context!

Work Item Executed - runtime context:
 Thread Name: sysworkq
 Thread Priority: -1
```

04_shell Sample

Description:

- Shell subsystem example[1]

Learn:

- Command line interface
- Command history, completion, help
- Great for hardware validation, testing and debugging

Sample:

- Provides a basic shell

```
CONFIG_SHELL=y

# Optional features
CONFIG_THREAD_STACK_INFO=y
CONFIG_KERNEL_SHELL=y
CONFIG_THREAD_MONITOR=y
CONFIG_BOOT_BANNER=n
CONFIG_THREAD_NAME=y
CONFIG_DEVICE_SHELL=y
CONFIG_POSIX_CLOCK=y
CONFIG_DATE_SHELL=y
CONFIG_THREAD_RUNTIME_STATS=y
CONFIG_THREAD_RUNTIME_STATS_USE_TIMING_FUNCTIONS=y
CONFIG_STATS=y
CONFIG_STATS_SHELL=y
CONFIG_SENSOR=y
CONFIG_SENSOR_SHELL=y
CONFIG_SENSOR_INFO=y
CONFIG_I2C_SHELL=y
```

samples/04_shell/prj.conf

[1] Equivalent in the Zephyr main Repository: zephyr/samples/subsys/shell/shell_module.

04_shell Sample - Console Output

```
uart:~$
  bypass              clear               date
  demo                device              devmem
  dynamic             help                history
  kernel              log                 log_test
  rem                 resize              retval
  section_cmd         shell               shell_uart_release
  stats               version
uart:~$ kernel thread list
Threads:
*0x20000720 shell_uart
    options: 0x0, priority: 14 timeout: 0
    state: queued, entry: 0x3ed9
    Total execution cycles: 22360 (0 %)
    stack size 2048, unused 932, usage 1116 / 2048 (54 %)

 0x20001288 sysworkq
    options: 0x1, priority: -1 timeout: 0
    state: pending, entry: 0x7169
    Total execution cycles: 163 (0 %)
    stack size 1024, unused 808, usage 216 / 1024 (21 %)
...
```

04_shell Sample - I2C Shell

```
uart:~$ i2c -h
i2c - I2C commands
Subcommands:
  scan          : Scan I2C devices
                  Usage: scan <device>
  recover       : Recover I2C bus
                  Usage: recover <device>
  read          : Read bytes from an I2C device
                  Usage: read <device> <addr> <reg> [<bytes>]
  read_byte     : Read a byte from an I2C device
                  Usage: read_byte <device> <addr> <reg>
  write         : Write bytes to an I2C device
                  Usage: write <device> <addr> <reg> [<byte1>, ...]
  write_byte    : Write a byte to an I2C device
                  Usage: write_byte <device> <addr> <reg> <value>

uart:~$ i2c scan i2c@40003000
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:             -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- 1d -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- 39 -- -- -- -- -- --
40: -- -- -- 43 -- -- -- -- -- -- -- -- -- -- -- --
3 devices found on i2c@40003000
```

04_shell Sample - Sensor Shell

```
uart:~$ sensor info
device name: apds9960@39, vendor: Avago Technologies, model: apds9960, ...
device name: mma8652fc@1d, vendor: NXP Semiconductors, model: fxos8700, ...
device name: ti_hdc@43, vendor: Texas Instruments, model: hdc, ...
device name: temp@4000c000, vendor: Nordic Semiconductor, model: nrf-temp, ...

uart:~$ sensor get ti_hdc@43
channel type=13(ambient_temp) index=0 shift=6 num_samples=1
  value=132211120605ns (26.427000)
channel type=16(humidity) index=0 shift=6 num_samples=1
  value=132211120605ns (36.364745)
```

05_sensor Sample

Description:

- TI HDC1010: I2C Temperature and Humidity Sensor[1]

Learn:

- Get Temperature and Humidity e.g. on the reel board via Sensor API

Sample:

- Demonstrates Sensor API

```
sensor_sample_fetch(dev);
sensor_channel_get(dev, SENSOR_CHAN_AMBIENT_TEMP,
                   &temp);
sensor_channel_get(dev, SENSOR_CHAN_HUMIDITY,
                   &humidity);

/* print the result */
printk("Temp = %d.%06d C, RH = %d.%06d %%\n",
       temp.val1, temp.val2,
       humidity.val1, humidity.val2);
```

samples/05_sensor/src/main.c

[1] Equivalent in the Zephyr main Repository: zephyr/samples/sensor/ti_hdc.

05_sensor Sample - Console Output

```
*** Booting Zephyr OS build v4.1.0 ***
Running on arm!
Dev 0x801c name ti_hdc@43 is ready!
Fetching...
Temp = 22.852966 C, RH = 38.793945 %
Fetching...
Temp = 22.842895 C, RH = 38.897705 %
Fetching...
```

06_ble Sample

Description:

- BLE Peripheral device, temperature monitor[1]

Learn:

- BLE peripheral role and advertising
- Health Thermometer Service (HTS)

Sample:

- App to connect: nRF Connect for Mobile (Android, iOS)

```
CONFIG_BT=y
CONFIG_LOG=y
CONFIG_BT_SMP=y
CONFIG_BT_PERIPHERAL=y
CONFIG_BT_DIS=y
CONFIG_BT_DIS_PNP=n
CONFIG_BT_BAS=y
CONFIG_BT_DEVICE_NAME="Zephyr Health Thermometer"
CONFIG_BT_DEVICE_APPEARANCE=768
CONFIG_CBPRINTF_FP_SUPPORT=y
CONFIG_SENSOR_SHELL=y
CONFIG_SENSOR_INFO=y
CONFIG_I2C_SHELL=y
```

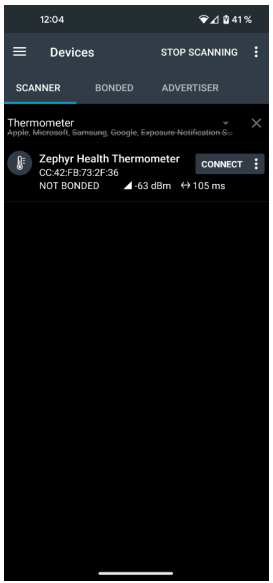
samples/06_ble/prj.conf

[1] Equivalent in the Zephyr main Repository: zephyr/samples/bluetooth/peripheral_ht.

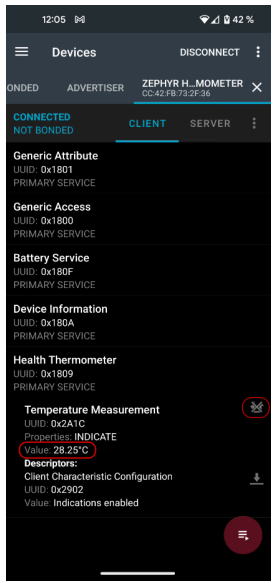
06_ble Sample - Console Output

```
*** Booting Zephyr OS build v4.1.0 ***
[00:00:00.380,645] <inf> bt_hci_core: HW Platform: Nordic Semiconductor (0x0002)
[00:00:00.380,676] <inf> bt_hci_core: HW Variant: nRF52x (0x0002)
[00:00:00.380,706] <inf> bt_hci_core: Firmware: Standard Bluetooth controller ...
[00:00:00.381,347] <inf> bt_hci_core: Identity: D0:6F:6B:78:0C:E8 (random)
[00:00:00.381,378] <inf> bt_hci_core: HCI: version 5.4 (0x0d) revision 0x0000, ...
[00:00:00.381,408] <inf> bt_hci_core: LMP: version 5.4 (0x0d) subver 0xffff
Bluetooth initialized
temp device is 0x28b5c, name is temp@4000c000
Advertising successfully started
Connected
temperature is 24C
temperature is 23.75C
Indication success
Indication complete
```

06_ble Sample - Connect with a Smartphone



Scanning for BLE devices

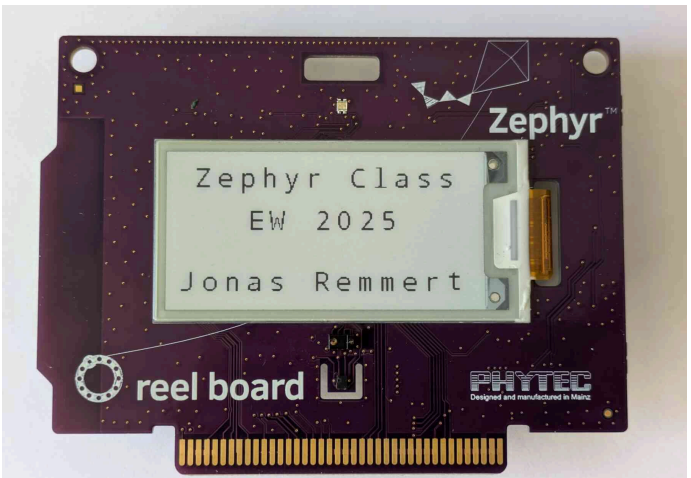


Connected with nRF Connect App

07_display_cfb Sample

Control the passive Display

- **Description:** Character Framebuffer Sample[1]
- **Sample:** Writes text to the display



Updated display on the reel board

[1]Equivalent in the Zephyr main Repository: zephyr/samples/subsys/display/cfb.

Hands-on 3

Test the Samples

Build and run the samples[1]:

```
west build -b native_sim samples/02_logging -p
west build -t run
```

or

```
west build -b reel_board@2 samples/02_logging -p
west flash
```

Task: Update the displayed name on the passive display

```
samples
├── 01_hello_world
├── 02_logging
├── 03_workqueues
├── 04_shell
├── 05_sensor
├── 06_ble
└── 07_display_cfb
```

[1]Note: The sensor and ble samples require a board to run.

Application Development

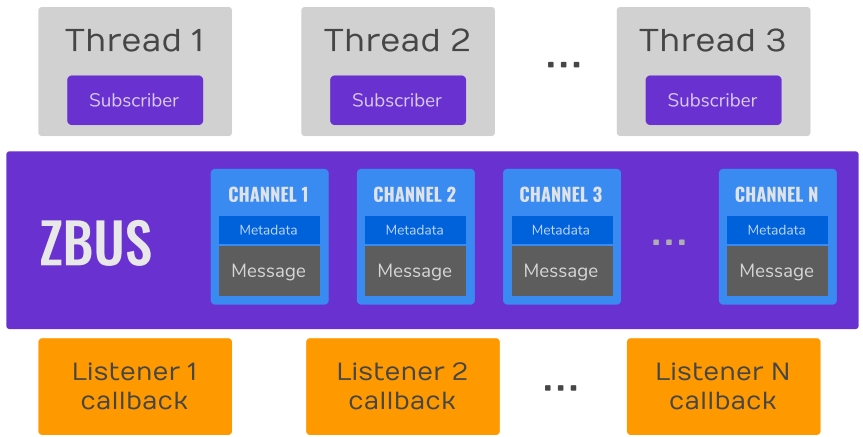
IPC Mechanisms and Zephyr bus (Zbus)

Classic

- Mutexes, Semaphores
- Conditional Variables, Message Queues
- Polling API to wait for any out of multiple conditions

Zbus

- Comparable to D-Bus in Linux
- Many-to-many communication
- Simplifies thread synchronization



Zbus overview

[1] docs.zephyrproject.org/latest/services/zbus

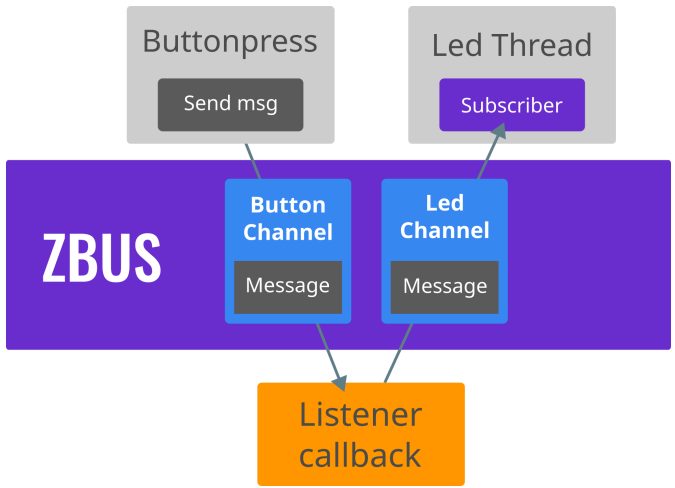
Example Application

Button

- Switch between system states (active, sleep)

LED

- Indicate system state
- Run in own thread for smooth animations



Minimal modular application with zbus

Modular Development

- **General:** Code reuse, maintainability, readability
- **IPC:** Communication e.g. via Zbus
- **Context:** Each module can be controlled independently
- **Testing:** Modules can be tested separately

```
app/
├── CMakeLists.txt
├── Kconfig
├── prj.conf
└── src
    ├── common
    │   ├── CMakeLists.txt
    │   ├── message_channel.c
    │   └── message_channel.h
    ├── main.c
    └── modules
        ├── button
        │   ├── button.c
        │   ├── CMakeLists.txt
        │   └── Kconfig.button
        └── led
            ├── led.c
            ├── CMakeLists.txt
            └── Kconfig.led
```

Testing Modules

Isolated testing of modules

- Add subsystems like `shell`, `ztest` for test cases
- Tests reference modules via CMake
- Interfaces abstracted via Zbus

Example Test Structure:

```
test/
├── button
│   ├── CMakeLists.txt
│   ├── prj.conf
│   └── src/main.c
└── led
    ├── CMakeLists.txt
    └── src/main.c
```

```
# test/button/CMakeLists.txt
target_sources(app PRIVATE src/main.c)
add_subdirectory(../../app/src/common common)
add_subdirectory(../../app/src/modules/button button)
```

```
// test/button/src/main.c
void button_test_msg_cb(const struct zbus_channel *chan) {
    const enum sys_msg *msg = zbus_chan_const_msg(chan);
    if (*msg == SYS_BUTTON_PRESSED) {
        LOG_INF("Button pressed!");
    }
}
ZBUS_LISTENER_DEFINE(button_test, button_test_msg_cb);
ZBUS_CHAN_ADD_OBS(button_ch, button_test, 1);
```

Automated Testing with Twister

Zephyr Test Runner (Twister)

- `west twister` - Automates building and running tests
- Supports multiple platforms (real HW or simulation)
- Integration tests via `sample.yaml`

Run Integration Tests:

```
west twister -T app/ -T test/ --integration
```

Example `sample.yaml`:

```yaml
integration_platforms:
  - reel_board
  - frdm_k64f
  - nrf52840dk/nrf52840
```

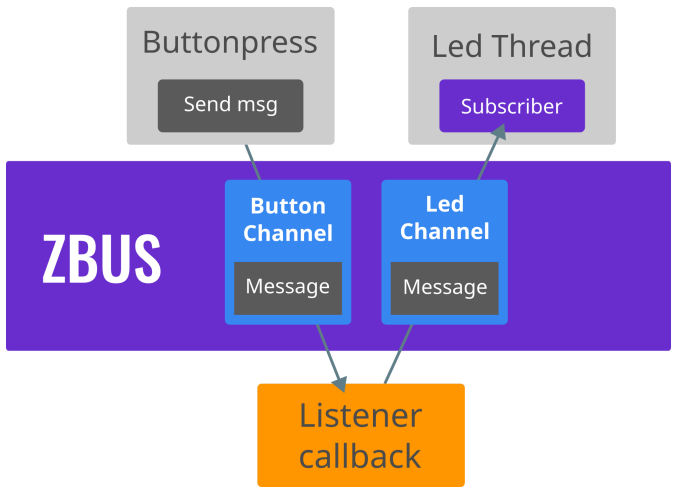
Output:

```
INFO - Total complete: 24/24 100%
INFO - 24 of 24 configurations passed
INFO - Run completed
```

Hands-on 4: Extend the Application

Currently: **standby**, **sleep**. **Task:** Add a third state: **active**. Cycle via button: `standby` -> `sleep` -> `active`

- **Sleep:** LED off
- **Standby:** LED blinking
- **Active:** LED on



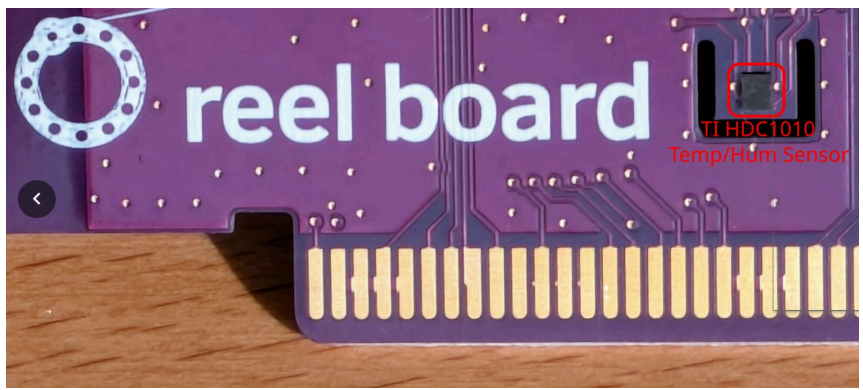
Hands-on 5: BLE Sensor Improvements

`06_ble` measures the temp with the sensor in the nRF52840 die. A higher precision is possible by using the TI HDC1010 sensor.

Change the sample accordingly:

- Confirm that the sensor is working (Sensor Shell)
- Enable the Sensor API
- Change the sensor device and get HDC1010 from DTS
- `SENSOR_CHAN_DIE_TEMP` → `SENSOR_CHAN_AMBIENT_TEMP`

Can you confirm a faster reaction time?



TI HDC1010 Temperature and Humidity Sensor

[1]Hint: Check the sensor sample `05_sensor` for reference. Only a few lines need to be changed.

Summary

Key Points from the Workshop

Ecosystem:

- Many components besides the RTOS itself, e.g. drivers, networking.
- Modular and hardware independent by default!
- Many concepts from Linux makes it easier to get started.

Structure:

- Ready-to-use structure for your application.
- Systems get more complex, avoid developing from scratch.

Open Source:

- Supply chain security.
- No vendor lock-in.
- Supported by large companies.

Community:

- Active, questions and contributions welcomed.

[1]Slides and workshop material available at: github.com/jonas-rem/zephyr-workshop

Questions?

Backup Slides

Debugging: git bisect

- Something worked in the past, but does not work anymore
- **Example:** The lvgl sample worked on the reel_board in v4.0.0, but not with the current main (7fc9c26fb0d)

```
git bisect start
git bisect good v4.0.0
git bisect bad 7fc9c26fb0d
west update && \
west build -b reel_board@2 samples/subsys/display/lvgl/ -p &
west flash
# → After flashing, test console and display to see if the
git bisect good|bad

  ... (repeat binary search for log2(N) steps)
  # log2(10000) ≈ 13.3 → max 14 steps for 10k commits!

git bisect good
1ea35e is the first bad commit
commit 1ea35e237cc0aa26b8a3517144528e9781a56781
Author: ***
Date:   Thu Jan 25 11:09:06 2024 +0100

    west.yml: Update lvgl module to v9.2.0
```

Find the commit that caused a bug with *git bisect*

Hands-On 4: Extension to 3 States - Solution (Test)

```
test/led/src/main.c
@@ -29,7 +29,9 @@ static int cmd_led( ... )
                led_msg(SYS_STANDBY);
                break;
        /* Add your code here */
+       case 2:
+               led_msg(SYS_ACTIVE);
+               break;
        /* */
        default:
                shell_print(sh, "Invalid argument");
@@ -41,9 +43,9 @@ static int cmd_led( ... )
 SHELL_SUBCMD_DICT_SET_CREATE(sub_led_cmds, cmd_led,
        (sys_sleep, 0, "System is sleeping"),
-       (sys_standby, 1, "System is in standby")
+       (sys_standby, 1, "System is in standby"),
        /* Add your code here */
+       (sys_active, 2, "System is active")
        /* */
 );
```

Hands-On 4: Extension to 3 States - app

app/src/common/message_channel.h

```
@@ -20,6 +20,10 @@ enum sys_states {
        /* Add your code here */
+       SYS_ACTIVE,
        /* */
 };
```

app/src/main.c

```
@@ -39,11 +39,14 @@ static void button_msg_cb( ... )
        case SYS_STANDBY:
+               sys_state = SYS_ACTIVE;
+               LOG_INF("System state active");
                break;
        /* Add your code here */
+       case SYS_ACTIVE:
+               sys_state = SYS_SLEEP;
+               LOG_INF("System state sleep");
+               break;
        /* */
```

app/src/modules/led/led.c

```
@@ -75,7 +75,12 @@ static void led_fn(void)
        /* Add your code here */
+       case SYS_ACTIVE:
+               gpio_pin_set_dt(&led, 1);
+               printk("LED on\n");
+               ret = zbus_sub_wait(&led_subscriber, &chan,
+               break;
        /* */
```

Hands-On 5: Change Sensor to HDC1010

```
diff --git a/samples/06_ble/prj.conf b/samples/06_ble/prj.cc
@@ -8,3 +8,4 @@ CONFIG_BT_BAS=y
 CONFIG_BT_DEVICE_NAME="Zephyr Health Thermometer"
 CONFIG_BT_DEVICE_APPEARANCE=768
 CONFIG_CBPRINTF_FP_SUPPORT=y
+CONFIG_SENSOR=y
```

```
diff --git a/samples/06_ble/src/hts.c b/samples/06_ble/src/h
@@ -25,8 +25,10 @@
 #include <zephyr/bluetooth/uuid.h>
 #include <zephyr/bluetooth/gatt.h>

+#include <zephyr/drivers/sensor.h>
+
 #ifdef CONFIG_TEMP_NRF5
-static const struct device *temp_dev = DEVICE_DT_GET_ANY(nc
+static const struct device *temp_dev = DEVICE_DT_GET_ANY(ti
 #else
 static const struct device *temp_dev;
 #endif
@@ -104,7 +106,7 @@ void hts_indicate(void)
-               r = sensor_channel_get(temp_dev, SENSOR_CHAN
+               r = sensor_channel_get(temp_dev, SENSOR_CHAN
                                       &temp_value);
```