Zephyr RTOS: From Idea to Integration

Intruducing development with Zephyr

Peter Fecher

February 18, 2026

Table of contents

1. Introduction

2. Let's talk about Zephyr

3. Zephyr's Impact on Development

4. Some Zephyr Perks

5. Summary

Introduction

Who am I?

Peter Fecher

- Embedded Software Engineer
- Working at PHYTEC since 2021
- IoT and Low Power Solutions

- Manufacturer for embedded hardware and software solutions
- Branches in Germany, France, USA, China and India
- About 400 employees worldwide

Figure 1: PHYTEC Headquarters in Mainz, Germany

Our Goal

- Take a look on integration of a real world product using Zephyr
- Look on what Zephyr is and what it is capable of
- Understand why Zephyr simplifies the development process of an embedded system

Our Goal

- Take a look on integration of a real world product using Zephyr
- Look on what Zephyr is and what it is capable of
- Understand why Zephyr simplifies the development process of an embedded system

Our Goal

- Take a look on integration of a real world product using Zephyr
- Look on what Zephyr is and what it is capable of
- Understand why Zephyr simplifies the development process of an embedded system

PHYTEC

Waste Water Monitoring for Trains

In collaboration with BeST Berliner Sensortechnik GmbH

The Problem:

- Trains collect waste water due to toilets, sinks and restaurants
- Waste water is emptied in the depot via vacuum hoses
- Process is done and controlled by a vacuum controlled extraction pistol
- The process is inaccurate and non-transparent

⇒ High potential for failure

Figure 2: DB Train Depot

PHYTEC

Waste Water Monitoring for Trains

In collaboration with BeST Berliner Sensortechnik GmbH

The Problem:

- Trains collect waste water due to toilets, sinks and restaurants
- Waste water is emptied in the depot via vacuum hoses
- Process is done and controlled by a vacuum controlled extraction pistol
- The process is inaccurate and non-transparent

⇒ High potential for failure

Figure 2: DB Train Depot

PHYTEC

Waste Water Monitoring for Trains

In collaboration with BeST Berliner Sensortechnik GmbH

The Problem:

- Trains collect waste water due to toilets, sinks and restaurants
- Waste water is emptied in the depot via vacuum hoses
- Process is done and controlled by a vacuum controlled extraction pistol
- The process is inaccurate and non-transparent

⇒ High potential for failure

Figure 2: DB Train Depot

PHYTEC

Waste Water Monitoring for Trains

In collaboration with BeST Berliner Sensortechnik GmbH

The Problem:

- Trains collect waste water due to toilets, sinks and restaurants
- Waste water is emptied in the depot via vacuum hoses
- Process is done and controlled by a vacuum controlled extraction pistol
- The process is inaccurate and non-transparent

⇒ High potential for failure

Figure 2: DB Train Depot

PHYTEC

Waste Water Monitoring for Trains

In collaboration with BeST Berliner Sensortechnik GmbH

The Problem:

- Trains collect waste water due to toilets, sinks and restaurants
- Waste water is emptied in the depot via vacuum hoses
- Process is done and controlled by a vacuum controlled extraction pistol
- The process is inaccurate and non-transparent

⇒ High potential for failure

Figure 2: DB Train Depot

PHYTEC

Waste Water Monitoring for Trains

In collaboration with BeST Berliner Sensortechnik GmbH

The Problem:

- Trains collect waste water due to toilets, sinks and restaurants
- Waste water is emptied in the depot via vacuum hoses
- Process is done and controlled by a vacuum controlled extraction pistol
- The process is inaccurate and non-transparent

⇒ High potential for failure

Figure 2: DB Train Depot

Waste Water Monitoring for Trains

In collaboration with BeST Berliner Sensortechnik GmbH

The Problem:

- Trains collect waste water due to toilets, sinks and restaurants
- Waste water is emptied in the depot via vacuum hoses
- Process is done and controlled by a vacuum controlled extraction pistol
- The process is inaccurate and non-transparent

⇒ High potential for failure

Figure 2: DB Train Depot

In collaboration with BeST Berliner Sensortechnik GmbH

The Solution

- A digital monitoring device
- Equipped with pressure and temperature sensors
- LEDs as UI
- Detects train cars via NFC
- LTE-M / DECT NR+ connectivity
- Shall not disturb the normal process, needs to be completely automatic
- No power or data cables

⇒ Capture sensor samples, analyze them on device and send them to DB infrastructure

⇒ More transparency

In collaboration with BeST Berliner Sensortechnik GmbH

The Solution

- A digital monitoring device
- Equipped with pressure and temperature sensors
- LEDs as UI
- Detects train cars via NFC
- LTE-M / DECT NR+ connectivity
- Shall not disturb the normal process, needs to be completely automatic
- No power or data cables

⇒ Capture sensor samples, analyze them on device and send them to DB infrastructure

⇒ More transparency

In collaboration with BeST Berliner Sensortechnik GmbH

The Solution

- A digital monitoring device
- Equipped with pressure and temperature sensors
- LEDs as UI
- Detects train cars via NFC
- LTE-M / DECT NR+ connectivity
- Shall not disturb the normal process, needs to be completely automatic
- No power or data cables

⇒ Capture sensor samples, analyze them on device and send them to DB infrastructure

⇒ More transparency

In collaboration with BeST Berliner Sensortechnik GmbH

The Solution

- A digital monitoring device
- Equipped with pressure and temperature sensors
- LEDs as UI
- Detects train cars via NFC
- LTE-M / DECT NR+ connectivity
- Shall not disturb the normal process, needs to be completely automatic
- No power or data cables

⇒ Capture sensor samples, analyze them on device and send them to DB infrastructure

⇒ More transparency

In collaboration with BeST Berliner Sensortechnik GmbH

The Solution

- A digital monitoring device
- Equipped with pressure and temperature sensors
- LEDs as UI
- Detects train cars via NFC
- LTE-M / DECT NR+ connectivity
- Shall not disturb the normal process, needs to be completely automatic
- No power or data cables

⇒ Capture sensor samples, analyze them on device and send them to DB infrastructure

⇒ More transparency

In collaboration with BeST Berliner Sensortechnik GmbH

The Solution

- A digital monitoring device
- Equipped with pressure and temperature sensors
- LEDs as UI
- Detects train cars via NFC
- LTE-M / DECT NR+ connectivity
- Shall not disturb the normal process, needs to be completely automatic
- No power or data cables

⇒ Capture sensor samples, analyze them on device and send them to DB infrastructure

⇒ More transparency

In collaboration with BeST Berliner Sensortechnik GmbH

The Solution

- A digital monitoring device
- Equipped with pressure and temperature sensors
- LEDs as UI
- Detects train cars via NFC
- LTE-M / DECT NR+ connectivity
- Shall not disturb the normal process, needs to be completely automatic
- No power or data cables

⇒ Capture sensor samples, analyze them on device and send them to DB infrastructure

⇒ More transparency

In collaboration with BeST Berliner Sensortechnik GmbH

The Solution

- A digital monitoring device
- Equipped with pressure and temperature sensors
- LEDs as UI
- Detects train cars via NFC
- LTE-M / DECT NR+ connectivity
- Shall not disturb the normal process, needs to be completely automatic
- No power or data cables

⇒ Capture sensor samples, analyze them on device and send them to DB infrastructure

⇒ More transparency

In collaboration with BeST Berliner Sensortechnik GmbH

The Solution

- A digital monitoring device
- Equipped with pressure and temperature sensors
- LEDs as UI
- Detects train cars via NFC
- LTE-M / DECT NR+ connectivity
- Shall not disturb the normal process, needs to be completely automatic
- No power or data cables

⇒ Capture sensor samples, analyze them on device and send them to DB infrastructure

⇒ More transparency

The Prototype

In collaboration with BeST Berliner Sensortechnik GmbH

- Teensy board
- Pressure sensor
- Some LEDs
- Battery with a standard boost converter
- Stick together with some electrical tape

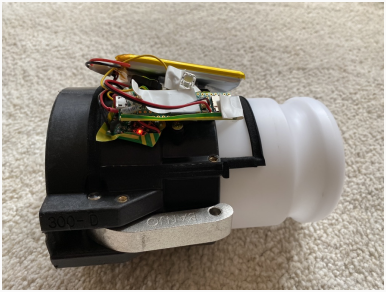


Figure 3: First Prototype of Waste Water Monitoring Device

⇒ Prototype proofed the concept

The Prototype

In collaboration with BeST Berliner Sensortechnik GmbH

- Teensy board
- Pressure sensor
- Some LEDs
- Battery with a standard boost converter
- Stick together with some electrical tape

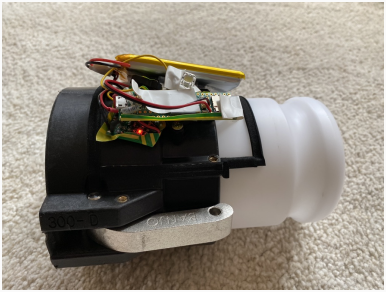


Figure 3: First Prototype of Waste Water Monitoring Device

⇒ Prototype proofed the concept

The Prototype

In collaboration with BeST Berliner Sensortechnik GmbH

- Teensy board
- Pressure sensor
- Some LEDs
- Battery with a standard boost converter
- Stick together with some electrical tape

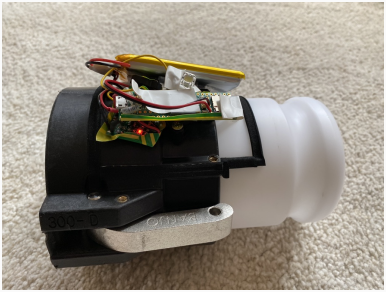


Figure 3: First Prototype of Waste Water Monitoring Device

⇒ Prototype proofed the concept

The Prototype

In collaboration with BeST Berliner Sensortechnik GmbH

- Teensy board
- Pressure sensor
- Some LEDs
- Battery with a standard boost converter
- Stick together with some electrical tape

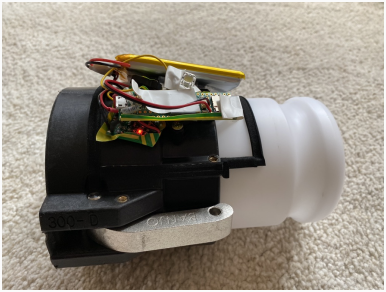


Figure 3: First Prototype of Waste Water Monitoring Device

⇒ Prototype proofed the concept

The Prototype

In collaboration with BeST Berliner Sensortechnik GmbH

- Teensy board
- Pressure sensor
- Some LEDs
- Battery with a standard boost converter
- Stick together with some electrical tape

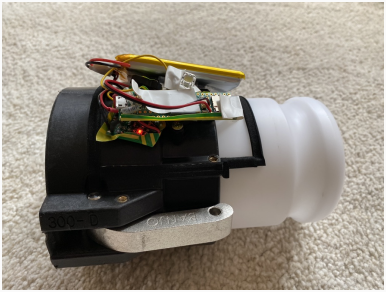


Figure 3: First Prototype of Waste Water Monitoring Device

⇒ Prototype proofed the concept

The Prototype

In collaboration with BeST Berliner Sensortechnik GmbH

- Teensy board
- Pressure sensor
- Some LEDs
- Battery with a standard boost converter
- Stick together with some electrical tape

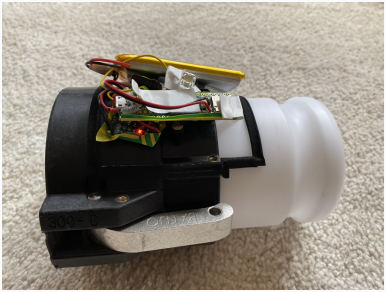


Figure 3: First Prototype of Waste Water Monitoring Device

⇒ Prototype proofed the concept

The Prototype

In collaboration with BeST Berliner Sensortechnik GmbH

- Teensy board
- Pressure sensor
- Some LEDs
- Battery with a standard boost converter
- Stick together with some electrical tape

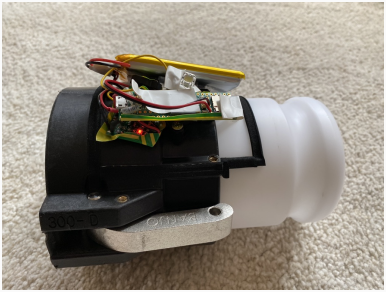


Figure 3: First Prototype of Waste Water Monitoring Device

⇒ Prototype proofed the concept

Let's talk about Zephyr

What is Zephyr?

- More than an RTOS – an IoT ecosystem
- All appliances – from microcontrollers over modems to Application processors
- Often an alternative to vendor SDKs
- Spreading in the industry
- Goal: To be the leading and most versatile IoT ecosystem

Figure 4: Zephyr RTOS Logo

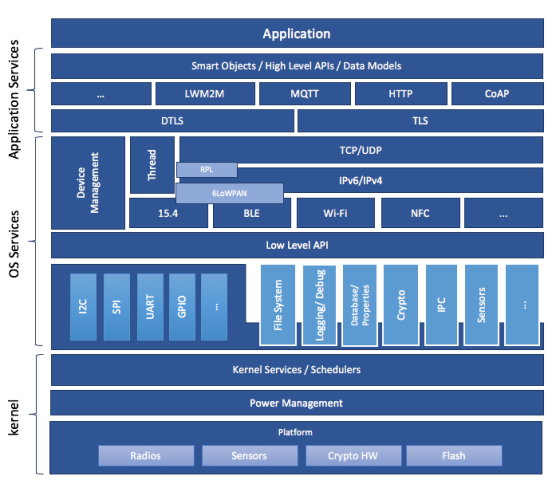


Figure 5: Zephyr System Architecture - https://docs.zephyrproject.org/latest/security/security-overview.html

Central Features of Zephyr

PHYTEC

Core Features

- Cross-platform compatibility
- Device drivers
- BLE, Bluetooth, Wi-Fi, 802.15.4 network stacks
- POSIX API support
- Security features (encryption, secure boot, etc.)
- Abstraction (!)

Developer Features

- Fully open source
- High modularity and scalability
- Extensive documentation
- Active community
- Useful subsystems (shell, logging, etc.)

Four Core Components

- **Modules:** Reusable code components and libraries
- **Kconfig:** Configuration system for the build process
- **CMakeLists:** Build system for compiling and linking the code
- **Device Tree:** Hardware description language

These technologies form the foundation for flexible and portable applications.

Modules

- Structured, reusable functional units
- Examples: kernel, drivers, subsystems, libraries
- Each module has its own configuration options
- Modules can depend on other modules
- Internal or external (e.g. from third-party vendors)

The Kconfig System

- Configuration management system used by Zephyr
- Based on the Linux kernel's Kconfig system
- Enabling/disabling features and setting parameters
- Generates compile-time options
- The `prj.conf` file defines the build configuration of a project
- Interactive configuration via `west build -t menuconfig` is possible

Example prj.conf:

```
CONFIG_LOG=y
CONFIG_LOG_DEFAULT_LEVEL=3
CONFIG_UART_CONSOLE=y
CONFIG_GPIO=y

```

Describes the available hardware

- Proven concept from the Linux kernel
- Central source of hardware information
- Separates hardware description from driver code

```
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>;
        };
};

```

In Summary

- **Device Tree**: “What hardware do I have?”
- **Kconfig**: “Which features shall be compiled?”
- **CMakeLists**: “How do I build the firmware?”
- **Modules**: Provide the actual implementation

$\Rightarrow$ Together, they generate a firmware image for the specific board

Keeping source in sync

- It is crucial to know which sources are used for the build process
- Zephyr provides tools to keep track of the sources and their versions

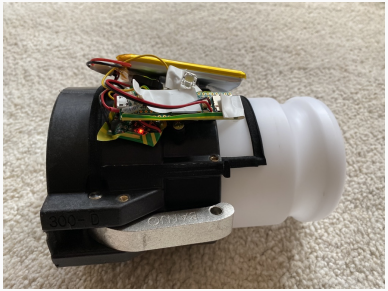
⇒ Zephyr's meta-tool `west`

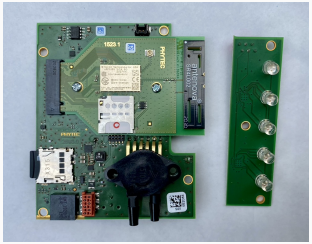
- Inspired by `repo` from Google and `git submodules`
- Managing multiple repositories
- Handles dependencies between source repositories
- Ensures that all sources are in sync
- Provides a unified interface for building, flashing and debugging

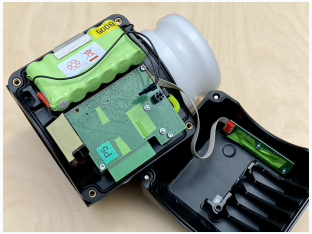
⇒ Consistent and reproducible builds

Zephyr's Impact on Development

From first prototype to series prototype





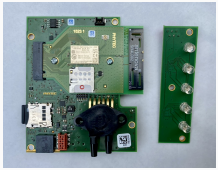


Prototype

- Teensy based (NXP Kinetis)
- Pressure Sensor
- Some LEDs
- Standard boost converter

Series Prototype

- Nordic nRF9160
- SD card reader
- SIM card and antenna for LTE-M
- LEDs via I^2C LED driver
- Accelerometer, temperature sensor
- Hall switch sensor



⇒ Many changes, but Zephyr helps us to integrate them with ease

Such an agile and iteration based process needs a modular structure
$\Rightarrow$ Let's take a look on how we can do this with Zephyr

What needs to be done?

- Change processor architecture (Kinetis $\rightarrow$ nRF91)
- Include LTE-M and SD Card support
- Configure the Sensors
- Add program code

Typical Project Structure

The goal is to have a modular structure, for easy integration of future iterations

⇒ **The typical project structure could look like this:**

```
project—wwm/
|— — boards/
|          +— — wwm1/
|                  |— — board.cmake
|                  |— — board.yaml
|                  |— — wwm1.dts
|                  |— — wwm1_defconfig
|                  +— — Kconfig.wwm1
|— — fw_app/
|      |— — CMakeLists.txt
|      |— — Kconfig
|      |— — prj.conf
|      +— — src/
|             |— — main.c
|             +— — modules/
|                    |— — io/
|                    |      |— — CMakeLists.txt
|                    |      |— — Kconfig
|                    |      +— — io.c
|                    |— — sensors/
|                    |— — led/
|                    +— — lte_mgr/
+— — west.yml
```

⇒ The new hardware needs to be described

This happens in the board definition:

- **wwm1.dts:** describes the hardware and its connections
- **board.yaml:** describes the build configuration for the board
- **Kconfig.wwm1:** describes the software configuration for the board
- **board.cmake:** describes the build process for the board
- **wwm1_defconfig:** describes the default configuration for the board

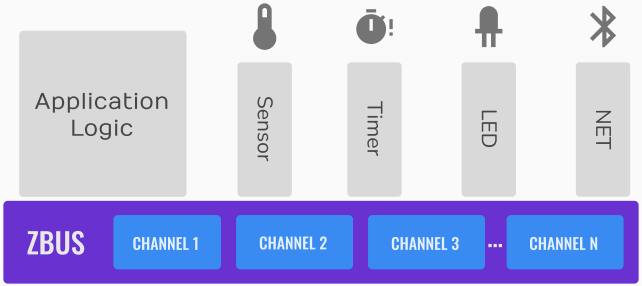
⇒ These files give the build system the information for the build configuration

Keeping Modules modular

- Modules need to be independent of each other but need to work together
- Each module has its own defined purpose
- How to achieve this but still have inter-module communication?

$\Rightarrow$ The answer is **ZBus**

⇒ **ZBus is a system level efficient publish-subscribe messaging system**



ZBus Architecture Diagram - `https://docs.zephyrproject.org/`

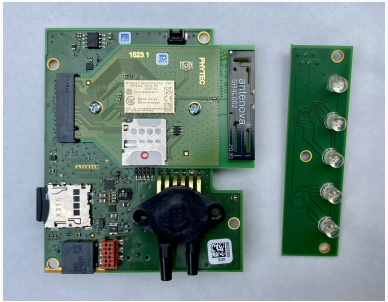
⇒ This allows us to keep modules modular and still communicate effectively

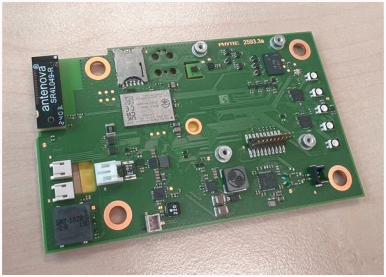
The new hardware needs LTE-M and SD Card support ⇒ This is done via prj.conf (Kconfig):

```
# Enable LTE-M support
CONFIG_LTE_LINK_CONTROL=y
# Networking
CONFIG_NETWORKING=y
CONFIG_NET_IPV4=y
# SD Card support
CONFIG_DISK_ACCESS=y
CONFIG_SDMMC=y

```

From series prototype to final product





The hardware had some changes since the last revision:

- nRF9160 → nRF9161 (DECT NR+ support)
- Some pins changed
- No SD card support any more
- New pressure sensor
- Different LED driver

⇒ The software needs *again* to be adapted to these changes

- Add new board definition for new processor and pinout
- Remove SD card support, depending on board
- Configure new pressure sensor

⇒ Even though the MCU changed, the adaption is no big deal

Some Zephyr Perks

Some Zephyr Perks

Zephyr has some features that can speed up the development process a lot

- The Zephyr Shell
- Emulation and Twister

Zephyr has a built-in Shell

```
*** Booting Zephyr OS build v4.2.0 ***
uart$

```

1. Add needed functionalities via Kconfig
2. Access the targets UART
3. Shell is available after boot of the device

Sounds nice - what can I do with it?

1. Available for various subsystems (e.g. GPIO, Bluetooth, Sensor)
2. Various Backends available (e.g. UART, Bluetooth LE, RPMsg)
3. New commands can be added easily
4. Mostly used for evaluation (e.g. testing a sensor, controlling GPIOs, etc.)
5. Can be used for testing in production

⇒ **This saves a lot of time and work in development and testing**

Twister is a test automation framework for Zephyr

- Run tests on multiple devices in parallel
- Supports various test types (e.g. unit tests, hardware tests, etc.)
- Can be used with emulation (e.g. QEMU) or real hardware
- Provides detailed reports and logs

As Zephyr is designed to be portable, it can be emulated on various platforms

- QEMU or native emulation is possible
- Allows testing and development without physical hardware
- Can be used in combination with Twister for automated testing

⇒ **Together with Twister, this allows programming before choosing the hardware**

Summary

⇒ **Some core concepts of Zephyr are completely new for the embedded world**

- Multi architecture support (and switching between them with ease)
- Modularity and reusability (modules, Kconfig, Device Tree)
- Many ready to use features
- The idea of *first write your program and tests* → *then worry about the hardware*

Thank you for listening!