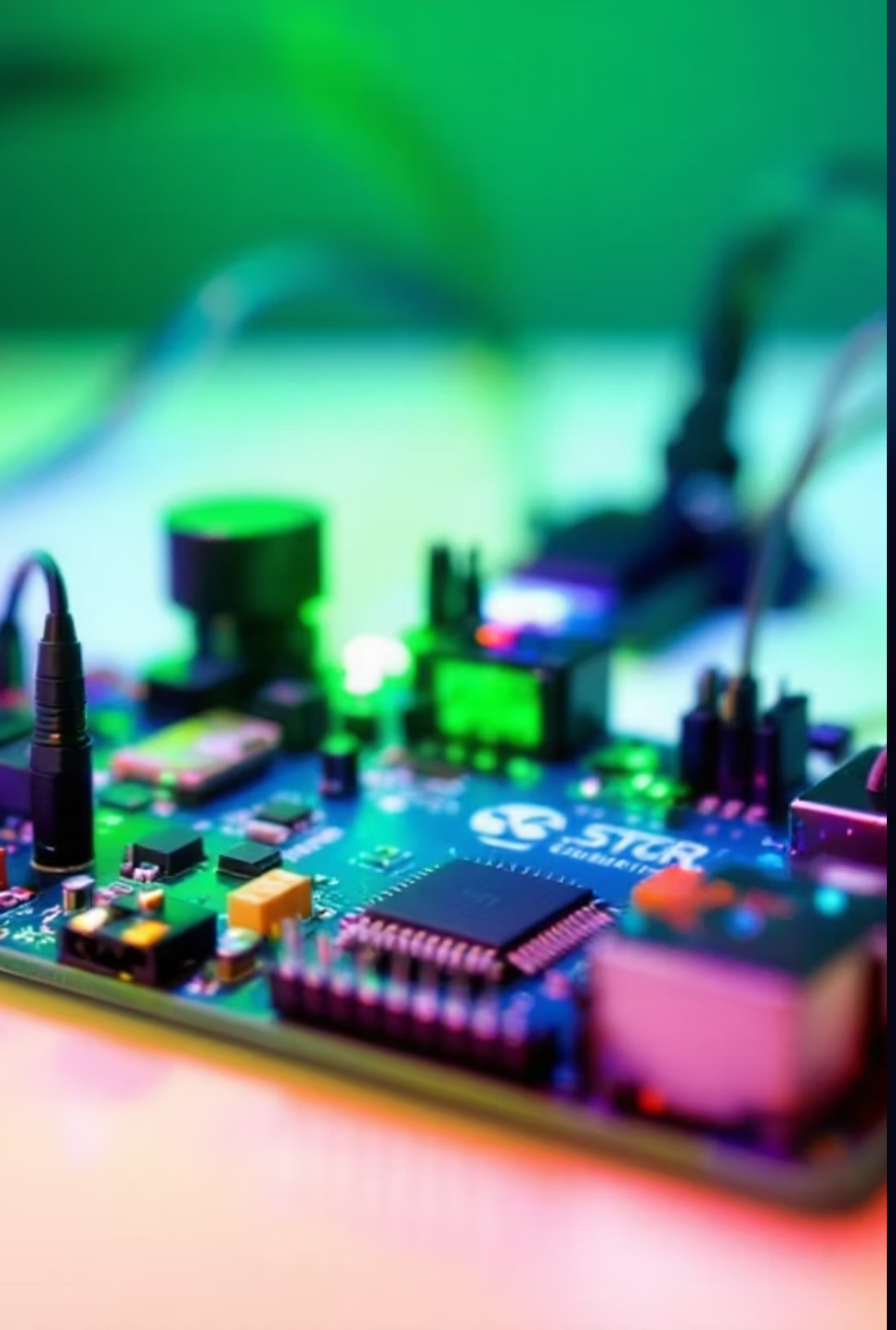


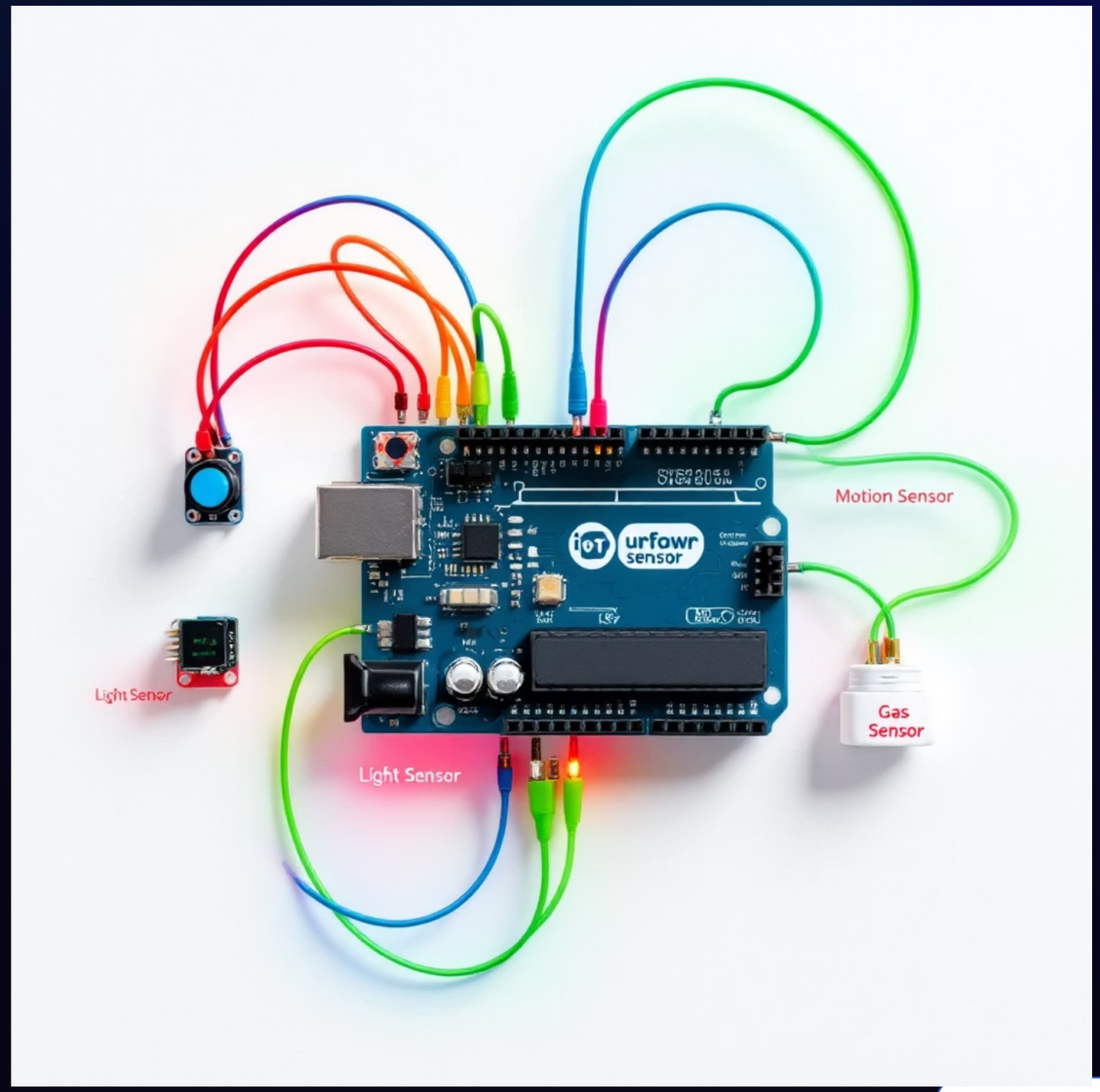
Zephyr Workshop - Sensor Subsystem

A comprehensive introduction to Zephyr's sensor abstraction layer, featuring practical implementation with the TMP102 temperature sensor.

What is the Sensor Subsystem?

The Sensor Subsystem in Zephyr RTOS provides:

- A generic interface to interact with various sensor hardware regardless of manufacturer
- Portable application code that works across different sensor implementations
- Unified API for accelerometers, gyroscopes, temperature sensors, and many more



Why Abstraction

Vendor-Agnostic

Switch between sensor hardware without rewriting application logic, enabling easy hardware upgrades

Simplified Architecture

Reduces complexity in the application layer by hiding implementation details

Consistent Pattern

Universal workflow: fetch → get channel → process data

This abstraction is particularly valuable in embedded systems where hardware configurations change frequently during development.

Sensor Driver API Overview

sensor_sample_fetch( )

Initiates a sensor reading operation, gathering fresh data from the hardware

sensor_channel_get()

Retrieves specific channel values from the most recent fetch operation

sensor_attr_set()

Optional: Configure sensor attributes like scale, frequency, and sensitivity

sensor_trigger_set()

Registers interrupt handlers or polling callbacks for sensor events

Zephyr Sensor Channels

Zephyr defines standard channel types that represent different sensor measurements:

SENSOR_CHAN_ACCEL_XYZ

3-axis acceleration data, typically in m/s^2

SENSOR_CHAN_GYRO_XYZ

3-axis angular velocity, typically in rad/s

SENSOR_CHAN_TEMP

Temperature readings in degrees Celsius

SENSOR_CHAN_LIGHT

Ambient light levels, often in lux

SENSOR_CHAN_PRESS

Atmospheric pressure, typically in kilopascals

Each channel type corresponds to a specific type of physical measurement that can be retrieved from compatible sensors.

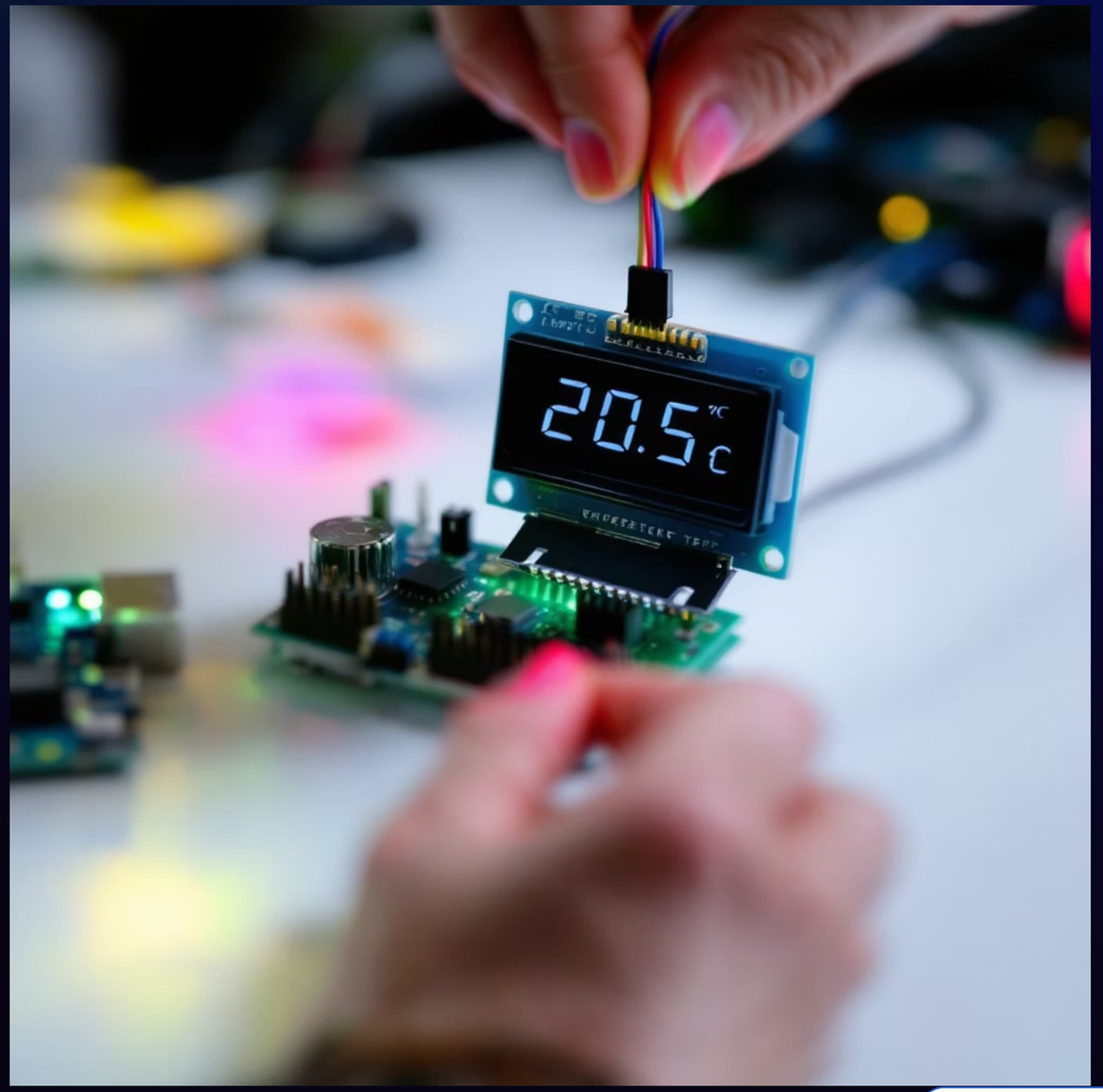
Sensor Value Struct

```
struct sensor_value {
int val1;  /* integer part */
int val2;  /* fractional part in micro units
*/
};
```

For example, to represent 20.500000°C:

- val1 = 20 (integer part)
- val2 = 500000 (fractional part)

```
printf("%d.%06d", val.val1, val.val2);
```



This representation allows for precise readings while avoiding floating-point operations, which can be

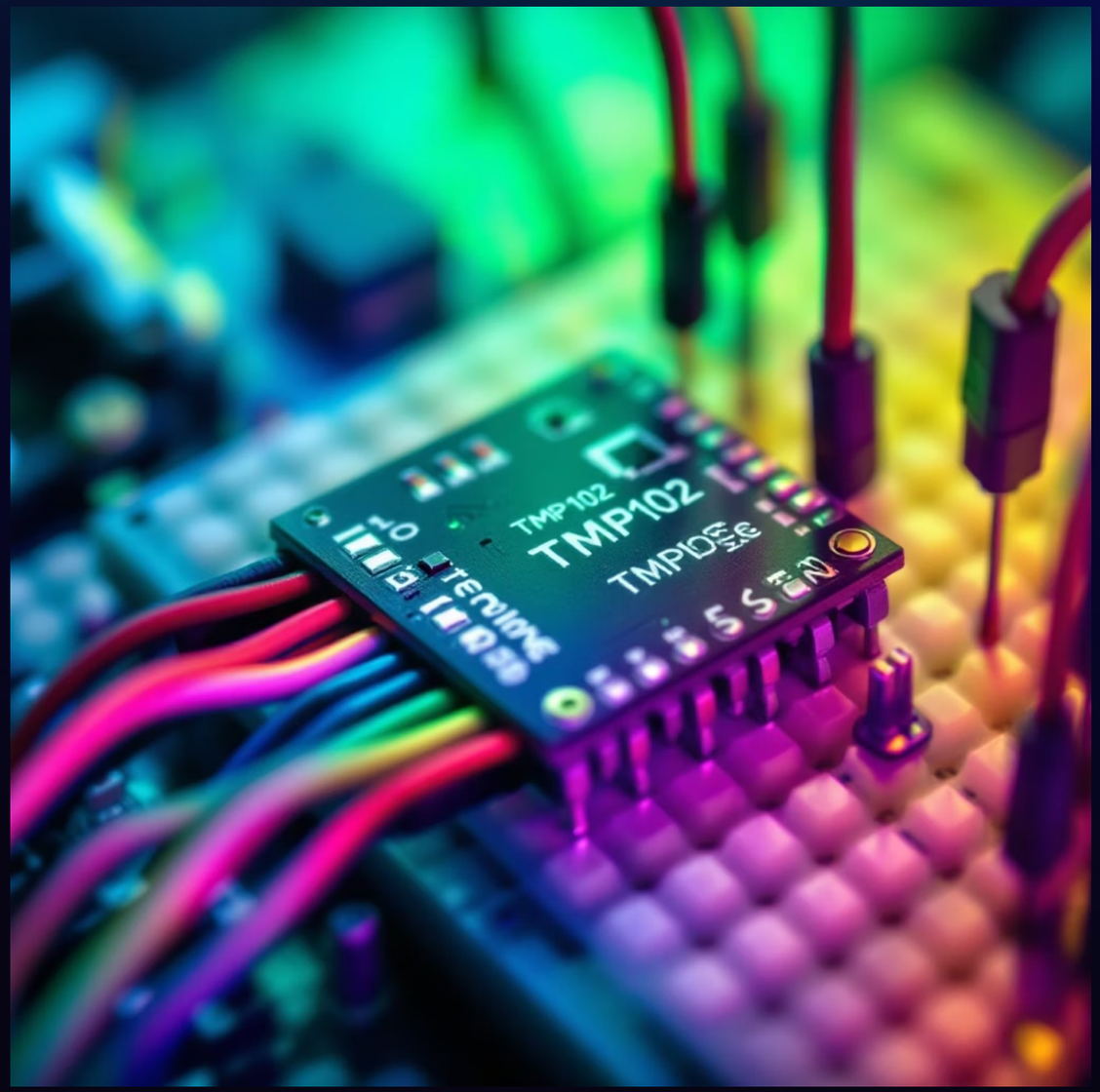
TMP112 Implementation of Sensor API

The TMP112 temperature sensor demonstrates how to implement the Zephyr sensor API:

- Extends the generic sensor driver interface
- Implements required callbacks for initialization
- Provides channel-specific data retrieval
- Translates raw sensor values to standard format

Key files to examine:

- include/zephyr/drivers/sensor.h - API definition
- drivers/sensor/tmp112/tmp112.c - Implementation example



What is Required in DeviceTree?

1

Sensor Description

Sensor must be properly described in the board's .dts file or in an application-specific overlay file

2

Bus Connection

Specify the connection method (typically I2C or SPI) with correct bus parameters

3

Required Properties

Include the "compatible" string to match driver, "reg" for address, and any GPIO interrupt connections

Sample Demo: TMP102

Setup

To prepare the TMP102 demo:

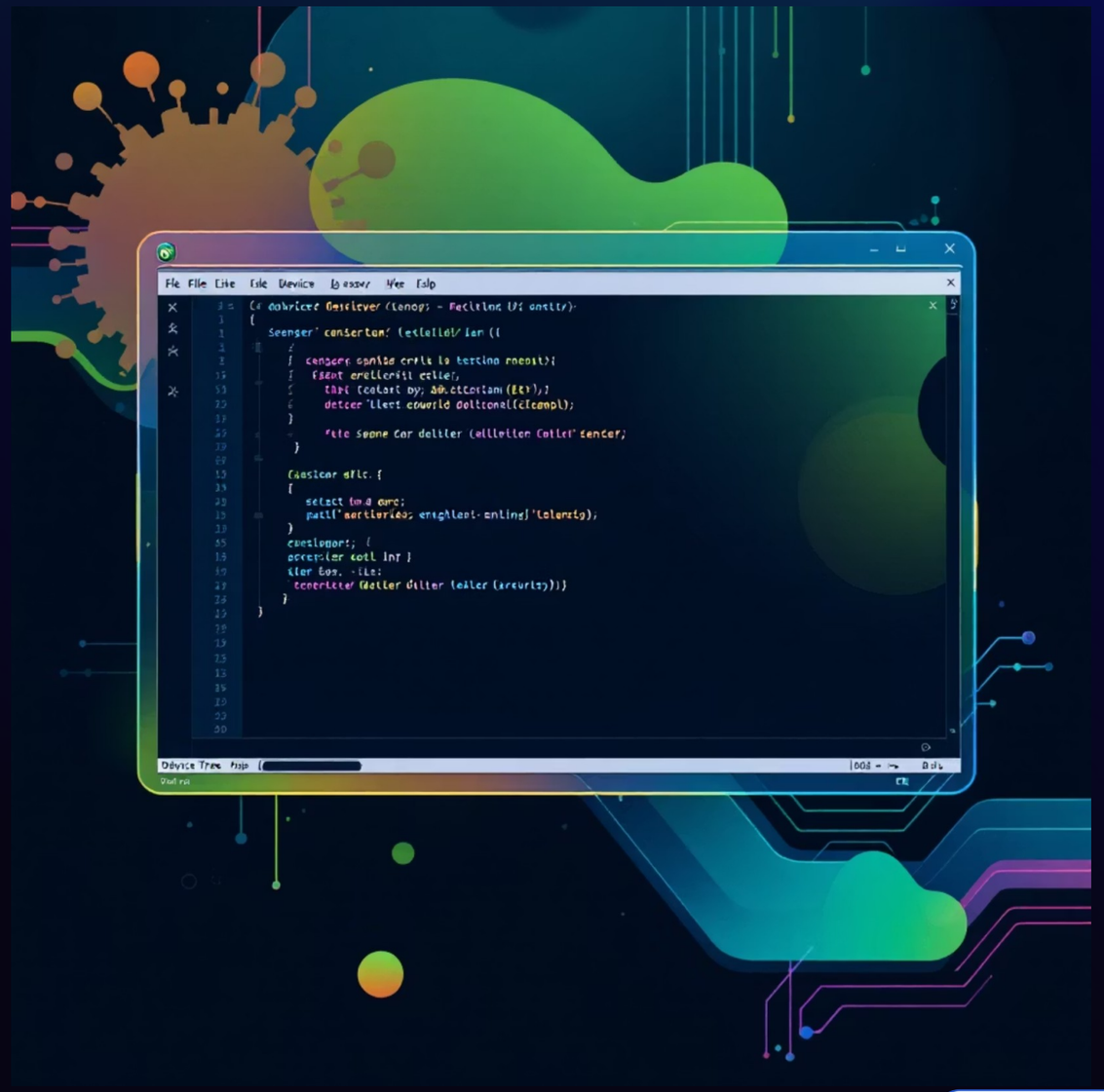
Navigate to the sample directory:

samples/sensor/tmp112

3. Update the arduino_i2c node from
4. Add the TMP103 sensor node with appropriate address

Create an **app.overlay** file for your board nrf52dk_nrf52832.dts

Hint: TMP102 is similar to TMP112 but uses a different I2C slave address. Check frdm_k64f.overlay for reference.



Sample Demo: Building and Testing

Build the Sample

Use the west build command to compile the sample for your target board:

```
west build -p always -b nrf52dk/nrf52832 samples/sensor/tmp112
```

Connect Hardware

Wire the TMP102 sensor to the development board according to the pinout documentation.

Flash and Test

Upload the firmware and verify operation:

```
west flash
```

Monitor the serial output to confirm temperature readings.

Shell Integration

- Zephyr allows registering CLI commands via shell subsystem
- comment out the do_main function
- add to prj.conf

  CONFIG_SHELL=y CONFIG_SENSOR_SHELL=y

- rebuild and flash
- run ‘sensor get tmp112@48 ambient_temp’

zephyr abstraction

- change zephyr board to nrf52833/40 or stm32wl
- rebuild and run