🧭Day 1 – Zephyr RTOS Setup & Debugging in VS Code

TechDhaba Embedded Systems Division

🚢Training Objectives

By the end of this session participants will: - Install and configure **Zephyr RTOS** on their systems.
- Build and flash a first Zephyr application.
- Debug applications locally and remotely using **VS Code + OpenOCD**.
- Understand the basics of RTOS tasks, threads, and logging.

🚡Day Plan (6 Hours)

Time	Module	Topics	Output
09:00 – 09:30	Intro to Zephyr	Architecture, kernel, drivers, board ports	Concept overview
09:30 – 10:30	Environment Setup	Install toolchain, SDK, and workspace	Zephyr workspace ready
10:30 – 11:30	First Build – Blinky	Build, flash, test	LED blinking
11:30 – 12:30	VS Code Integration	Extensions, CMake, Cortex-Debug	VS Code builds successfully
12:30 – 13:30	Debugging with OpenOCD	Launch, breakpoints, inspect	Stepping through code
13:30 – 15:00	Lab – Thread & Logging Demo	Multi-threaded LED & log sample	Multi-thread debug working

---

🛖Module 1 – Zephyr RTOS Architecture (Overview)

- Microkernel + monolithic hybrid design.
- Configurable using **Kconfig**.
- Modular build using **CMake + Ninja**.
- Supports drivers, networking, BLE, USB, etc.

Module 2 – Environment Setup

Step 1: Install Dependencies

```
sudo apt update
sudo apt install --yes git cmake ninja-build gperf ccache dfu-util
  device-tree-compiler wget python3-pip python3-venv python3-tk
  openocd udev
```

Step 2: Workspace Creation

```
mkdir ~/zephyrproject && cd ~/zephyrproject
west init zephyrproject
cd zephyrproject
west update
west zephyr-export
pip3 install -r zephyr/scripts/requirements.txt
```

Step 3: Install Zephyr SDK

```
wget https://github.com/zephyrproject-rtos/sdk-ng/releases/download/v0.16.5/
zephyr-sdk-0.16.5_linux-x86_64.tar.xz
tar xf zephyr-sdk-0.16.5_linux-x86_64.tar.xz
sudo mv zephyr-sdk-0.16.5 /opt/zephyr-sdk
/opt/zephyr-sdk/setup.sh
```

Module 3 – First Application: Blinky

Build and Flash

```
cd ~/zephyrproject/zephyr
west build -b nucleo_f429zi samples/basic/blinky
west flash
```

✅LED should start blinking.

To check available boards:

```
west boards
```

🛖Module 4 – VS Code Integration

Required Extensions

- **C/C++ Extension Pack**
- **CMake Tools**
- **Cortex-Debug**
- **Remote SSH** (optional)

`launch.json`

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Debug (OpenOCD)",
      "type": "cortex-debug",
      "request": "launch",
      "servertype": "openocd",
      "cwd": "${workspaceRoot}",
      "executable": "${workspaceRoot}/build/zephyr/zephyr.elf",
      "device": "STM32F429ZI",
      "configFiles": [
        "interface/stlink.cfg",
        "target/stm32f4x.cfg"
      ],
      "runToMain": true
    }
  ]
}
```

Start Debugging

Press **F5** → Cortex Debug session starts → set breakpoints in `main()`.

🚥Module 5 – OpenOCD Debug Commands

Manual debug flow:

```
openocd -f interface/stlink.cfg -f target/stm32f4x.cfg
arm-none-eabi-gdb build/zephyr/zephyr.elf
(gdb) target remote localhost:3333
(gdb) monitor reset halt
```

```
(gdb) break main
(gdb) continue
```

Observe registers, stack frames, and variables.

🛝 Module 6 – Thread & Logging Demo

Application Code (`main.c`)

```c
#include <zephyr.h>
#include <sys/printk.h>
#include <drivers/gpio.h>

#define LED0_NODE DT_ALIAS(led0)
#define SLEEP_TIME_MS 500

const struct gpio_dt_spec led = GPIO_DT_SPEC_GET(LED0_NODE, gpios);

void blink_thread(void)
{
    while (1) {
        gpio_pin_toggle_dt(&led);
        printk("LED toggled!\n");
        k_msleep(SLEEP_TIME_MS);
    }
}

K_THREAD_DEFINE(blink_tid, 1024, blink_thread, NULL, NULL, NULL, 5, 0, 0);

void main(void)
{
    gpio_pin_configure_dt(&led, GPIO_OUTPUT_ACTIVE);
    printk("Zephyr Thread Demo Start\\n");
}
```

Expected Result

Two threads running simultaneously, visible in VS Code debug view (Registers → Threads list).

Module 7 – Remote Workspace Setup

```
ssh user@192.168.x.x
# Clone and build remotely
code --remote ssh-remote+user@192.168.x.x ~/zephyrproject/zephyr
```

Build and debug over SSH — perfect for distributed hardware labs.

Module 8 – Lab Exercises

Task	Description	Outcome
Lab 1	Modify blinky to use two threads	Verify scheduler activity
Lab 2	Add logging using `LOG_INF()`	Observe UART console logs
Lab 3	Debug crash by inserting NULL pointer	Use OpenOCD to analyze fault
Lab 4	Remote build from VS Code SSH	Verify artifact sync & flash remotely

---

🛖 Module 9 – Troubleshooting

Problem	Fix
`west: command not found`	Re-source Zephyr environment: `source zephyr/zephyr-env.sh`
OpenOCD error `target not halted`	Reset board and try again
No LED blink	Check device tree alias `led0`
GDB cannot connect	Verify ports 3333 and 4444 open

🌆 Module 10 – Wrap-Up

- Zephyr RTOS workspace configured successfully.
- Blinky and multi-thread apps built and debugged.
- VS Code ready for remote and local debug flows.

"At TechDhaba, we turn complex embedded debugging into a simple, visual art form."

Appendix – Quick Commands

Command	Purpose
`west build -b <board>`	Build for specific board
`west flash`	Flash firmware
`west debug`	Launch GDB debug session
`west boards`	List supported boards
`arm-none-eabi-gdb <elf>`	Manual debugging
`openocd -f <cfg>`	Start OpenOCD server