🧭Setting up Zephyr RTOS Development and Debugging in VS Code

Authored by TechDhaba Embedded Systems Division

> *Empowering engineers to build, debug, and deploy reliable embedded systems using open ecosystems.*

🚃Overview

This comprehensive guide by **TechDhaba** includes **all steps** — from installing Zephyr, running your first application, to setting up **OpenOCD and remote debugging** — designed for both individual developers and enterprise teams.

It concludes with TechDhaba’s expertise in **RTOS training, debugging frameworks, and scalable remote workspace deployments**.

---

1. System Requirements

Hardware

- STM32, nRF52, ESP32, or any Zephyr-supported board.
- Debug probe: ST-Link / J-Link / CMSIS-DAP.
- Reliable internet connection.

Software (Host Machine)

- Ubuntu 22.04 / Windows 11 / macOS (Linux recommended).
- Python 3.8+
- Git
- CMake 3.20+
- VS Code (latest version).
- ARM GCC Toolchain

---

2. Installing Zephyr RTOS

Step 1: Install Prerequisites

```
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: Create a Workspace

```
mkdir ~/zephyrproject
cd ~/zephyrproject
```

Step 3: Clone Zephyr Source

```
west init zephyrproject
cd zephyrproject
west update
west zephyr-export
```

Step 4: Install Python Dependencies

```
pip3 install -r zephyr/scripts/requirements.txt
```

---

3. Installing Zephyr SDK

For **ARM-based boards (e.g., STM32, nRF)**, install the 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
```

Verify installation:

```
echo $ZEPHYR_BASE
```

🛕4. Building Your First Application

Example: **Blinky Application**

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

✅LED should start blinking.

To check available boards:

```
west boards
```

🕋5. Setting Up VS Code

Install Extensions

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

Configure `launch.json` for OpenOCD

```
{
  "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,
      "svdFile": "${workspaceRoot}/zephyr/soc/st/stm32f4/stm32f429.svd"
    }
  ]
}
```

Start debugging → Connect via ST-Link → Press **F5** → View breakpoints, stack trace, and registers live.

---

6. Remote Workspace Setup

Enable **remote Zephyr debugging** for distributed teams:

```
ssh user@192.168.x.x
```

Then open workspace in VS Code using **Remote - SSH** extension.

✅Build, flash, and debug remotely — ideal for shared lab boards (Raspberry Pi, Jetson, or ARM servers).

🚞7. Debugging with OpenOCD

Start manually:

```
openocd -f interface/stlink.cfg -f target/stm32f4x.cfg
```

Then in GDB:

```
arm-none-eabi-gdb build/zephyr/zephyr.elf
(gdb) target remote localhost:3333
(gdb) monitor reset halt
(gdb) continue
```

Use VS Code for GUI-driven debugging — integrates directly with OpenOCD.

🛖8. Common Debug Scenarios

Issue Type	Symptom	Debug Tip
GPIO not toggling	LED not blinking	Inspect ODR register live
Hard fault	CPU halts/reset	Check SP & exception vector
Thread stuck	No context switch	Use Zephyr shell analyzer
ISR misfire	No interrupt trigger	Inspect NVIC enable/pend

🛖9. TechDhaba Expertise

At **TechDhaba**, our embedded division has developed and deployed advanced debugging ecosystems across industries.

✅ **Our Highlights:** - Multi-board Zephyr debugging (remote & local) - Automated GDB and OpenOCD integration - VS Code-based cloud debugging labs for training and enterprise use - Hands-on workshops for engineers (RTOS, Kernel, and Driver Debugging)

We've debugged **critical firmware failures** over remote OpenOCD links — including ISR lockups, scheduler deadlocks, and DMA data corruption — proving the robustness of this setup in real-world enterprise environments.

> "Debugging is not just fixing bugs; it's understanding the system's truth beneath abstraction."

🚢10. Conclusion

With Zephyr, VS Code, and OpenOCD, your workflow becomes **modern, scalable, and production-ready**.

This document is ideal for: - Training engineers in modern RTOS environments. - Deploying standardized embedded DevOps pipelines. - Building a remote debugging infrastructure.

TechDhaba = Simplicity. Scalability. Clarity.

Appendix – Quick Commands

Command	Purpose
`west build -b <board>`	Build for a specific board
`west flash`	Flash firmware

Command	Purpose
`west debug`	Launch GDB debug session
`west boards`	List supported boards
`arm-none-eabi-gdb <elf>`	Manual debugging
`openocd -f <cfg>`	Start OpenOCD server