Relatório Técnico: Configuração do Ambiente Zephyr OS, Ajustes de Ferramentas e Planejamento Futuro

Nicolas Jordani – UFSM

November 27, 2025

1 Introdução

Este relatório descreve detalhadamente o processo de instalação, configuração e correção do ambiente de desenvolvimento baseado no Zephyr OS dentro de uma máquina virtual. Durante o procedimento foram encontrados diversos problemas relacionados às versões do *Python*, do gerenciador de projetos *west*, da árvore de códigos do *Zephyr*, e do particionamento do disco virtual. O documento apresenta as causas desses problemas, as soluções adotadas e os próximos passos planejados para evolução do projeto, incluindo a integração de BLE e comunicação via rádio entre placas.

2 Ambiente Utilizado

- Sistema operacional: Ubuntu 24.04 LTS (Noble)
- Máquina virtual: VirtualBox, com particionamento manual de HDD
- Python: versão final utilizada – 3.10.x
- Zephyr OS: versão utilizada após correções – v4.3.99
- Ferramenta de build: west v1.5.0 (única versão disponível via PyPI para Python 3.10/3.11/3.12)
- Toolchain: Zephyr SDK 0.16.x

3 Problemas Encontrados e Soluções

3.1 Incompatibilidade entre Python 3.12 e o Zephyr

A máquina virtual inicialmente utilizava Python 3.12, versão padrão do Ubuntu 24.04. Entretanto, o Zephyr OS moderno (branch *main*) depende de versões específicas de Python entre 3.8 e 3.10. Além disso, versões mais novas do Python impediam a instalação do *west* atualizado.

Solução: instalar Python 3.10 via repositório de terceiros (Deadsnakes) e criar um ambiente virtual dedicado (venv).

3.2 Versão do West bloqueada em 1.5.0

Mesmo com Python 3.10, o comando `pip install west` sempre instalava a versão 1.5.0. Constatou-se que:

- Não existe versão mais nova do *west* publicada no PyPI.
- O Zephyr main requer west $\geq$ 1.12, impossível de instalar via pip.

Conclusão: a única forma suportada no cenário atual é usar o Zephyr adequado à versão do west disponível.

3.3 Incompatibilidade entre Zephyr main e West 1.5.0

O branch *main* do Zephyr utiliza funcionalidades removidas de versões antigas do west, causando erros como:

- Falta de `runners.yaml`
- Remoção do comando `west run`
- Mudanças no modelo de simulação

Apesar disso, após certos ajustes, o build funcionou usando o alvo `west build -t run`, que passou a ser o método oficial.

3.4 Particionamento da máquina virtual

O espaço originalmente configurado era insuficiente. Isso afetou:

- Instalação do Zephyr SDK
- Clonagem da árvore do Zephyr

- Cache de builds

Correção: expandir ou recriar o disco virtual com tamanho adequado (40 GB para Zephyr + SDK + QEMU).

4 Processo Final Padronizado de Execução

O fluxo de execução final validado é:

1. Ativar o ambiente virtual Python 3.10:

   ```
   source .venv-310/bin/activate
   ```

2. Acessar o workspace:

   ```
   cd /mnt/dados/zephyr
   ```

3. Compilar o exemplo:

   ```
   west build -b qemu_x86 zephyr/samples/hello_world -d build
   ```

4. Executar no QEMU:

   ```
   west build -t run
   ```

5. Encerrar o QEMU:

   ```
   Ctrl + A, X
   ```

Este fluxo foi testado e comprovadamente funcional.

5 Próximos Passos do Projeto

5.1 Implementação de BLE para Acesso ao Log via Celular

O próximo objetivo é integrar Bluetooth Low Energy no projeto, permitindo:

- Visualização de logs da placa via smartphone
- Envio de comandos simples para depuração
- Interação com aplicativos BLE (Android e iOS)

A abordagem recomendada é:

1. Habilitar o subsistema BLE no Zephyr (`CONFIG_BT`, `CONFIG_BT_PERIPHERAL`).
2. Criar um serviço GATT customizado.
3. Expor característica de log.
4. Testar com *nRF Connect* no celular.

5.2 Comunicação Entre Placas por Rádio

Havendo sucesso no BLE, o passo seguinte é integrar comunicação por rádio entre duas placas:

- Celular → BLE → Placa 1
- Placa 1 → Rádio (LoRa, 802.15.4, nRF, etc.) → Placa 2

A meta é adaptar o aplicativo de comunicação por rádio criado por um colega, tornando-o compatível com o fluxo acima.

6 Conclusão

O processo exigiu diversas correções de ambiente devido às mudanças recentes no Zephyr OS e às restrições das versões do *west* disponíveis no PyPI. Entretanto, após ajustes no Python, no particionamento da máquina virtual e no método de execução, o ambiente foi estabilizado com sucesso.

Os próximos passos envolvem avanços funcionais significativos, incluindo integração BLE e comunicação híbrida BLE + rádio, com potencial para construção de uma arquitetura robusta de comunicação entre dispositivos.