Implementing a Secure Firmware Update Mechanism for IoT Devices

(Zephyr Real-time Operating System)

Submitted by: Sujata Mahar

Assigned by: Spinnaker Analytics

Deadline: 05th August, 2025

Project Overview

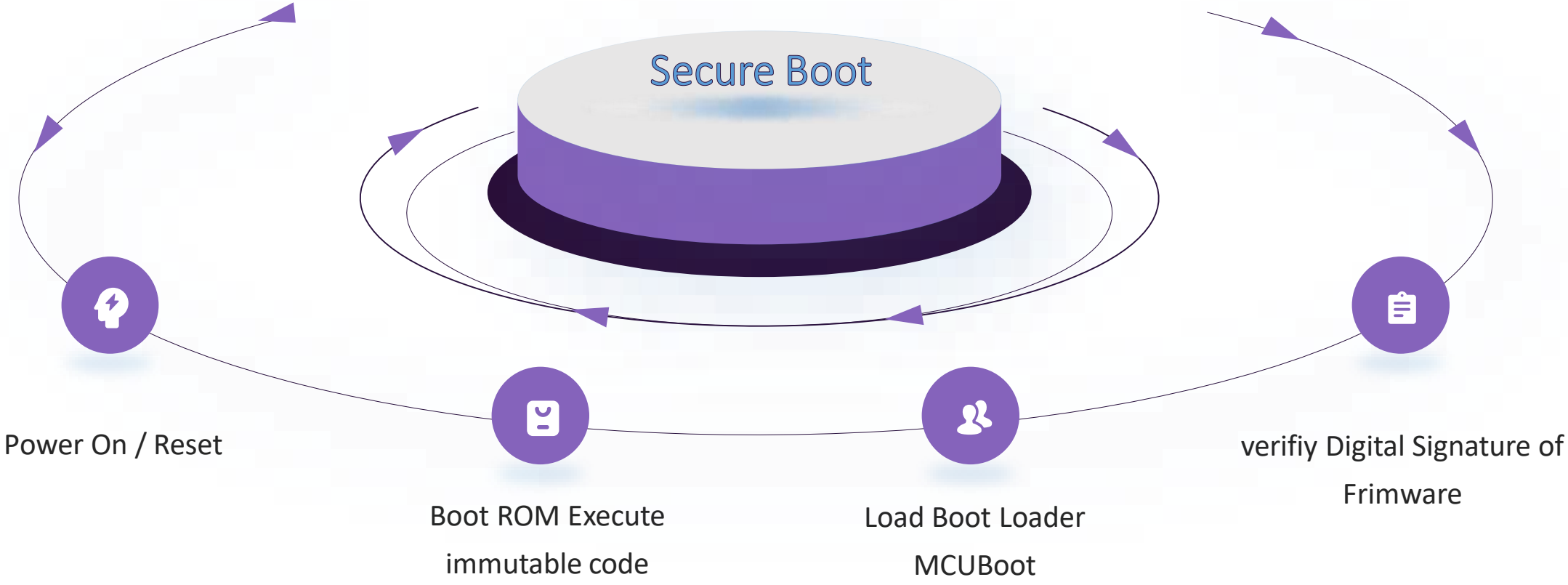
An IoT solutions provider wants to ensure that firmware updates for their devices are safe and trustworthy. The goal is to protect the devices from unauthorized or tampered firmware by using secure boot, code signing, and update verification mechanisms.

In this project, I demonstrate how to set up a secure firmware update process using the Zephyr Real-Time Operating System (RTOS) in a virtualized environment on VMware. The implementation includes configuring secure boot using **MCUBoot**, signing the firmware to verify its authenticity, performing update integrity checks before installation, and enabling secure communication (TLS) during the firmware distribution process all tested and validated within a VMware-based Ubuntu system.

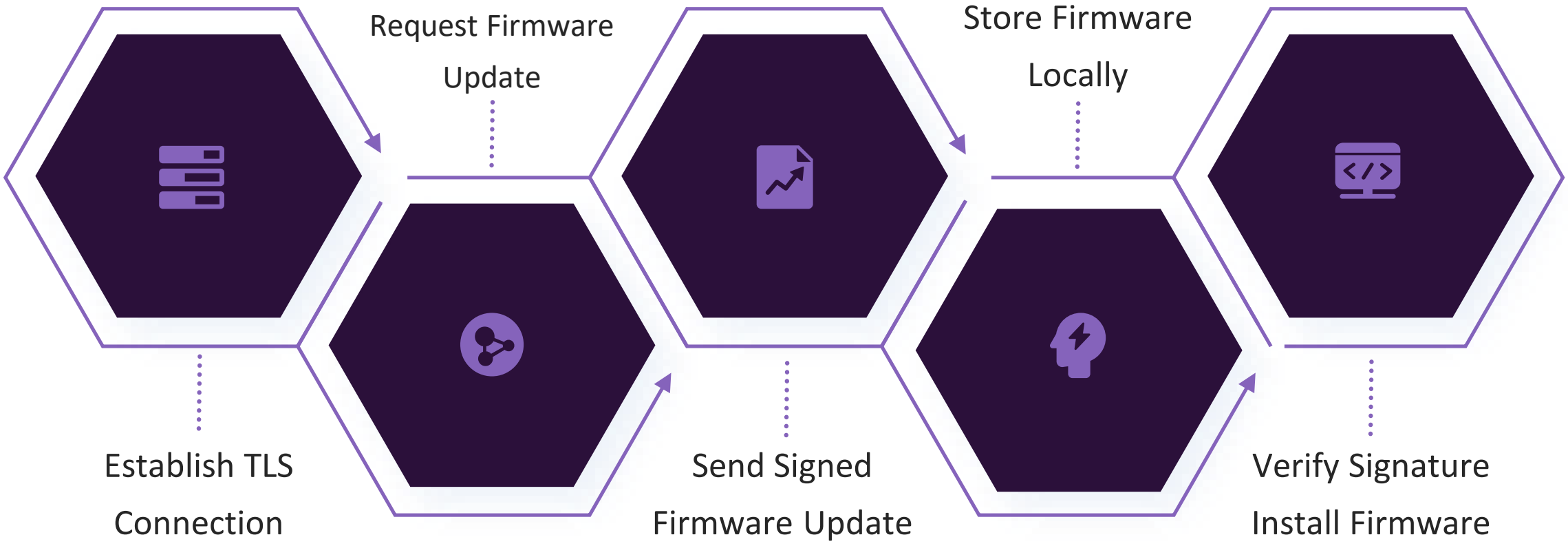
Design & Architecture:

- **Firmware Signing:** The firmware is signed with a private key to ensure it’s genuine and hasn’t been tampered with.
- **Secure Boot:** The device only runs firmware if the digital signature is verified by the bootloader (MCUBoot).
- **Update Server:** A Python-based HTTPS server delivers signed firmware updates securely.
- **TLS Communication:** Devices use encrypted TLS connections to safely download firmware without risk of interception.

Flow Diagrams



Sequence Diagram of Firmware Update Process via TLS



Environment Setup and Tool Justification

In this project, VMware Workstation was used to set up a virtual Ubuntu environment, simulating the secure update server required for the firmware update process. A Python-based HTTPS server was implemented inside this virtual machine to securely host signed firmware files. The server was configured with TLS to ensure encrypted communication between the server and IoT devices.

All cryptographic operations, such as firmware signing using RSA/ECC, key generation, and signature verification, were performed inside the VMware-hosted Ubuntu system. This setup enabled realistic testing of the server-side update delivery flow without the need for physical hardware.

Since a physical IoT board or QEMU-based emulation was not available during testing, secure boot functionality on the device side could not be directly demonstrated. However, the complete backend infrastructure — including firmware signing, update hosting, and TLS-based delivery — was successfully implemented and tested in a controlled virtual environment.

🚀 Practical Setup Location

- **Practical setup done using** : VMware Workstation.
- **Operating System used**: Ubuntu 20.04 LTS (64-bit).
- **Python-based HTTPS server** created for firmware distribution.
- **TLS (SSL) encryption** used for secure communication.
- **Firmware signing and verification** done using OpenSSL.
- **No physical IoT device used** — testing done in virtual environment.
- Device emulation (like QEMU) planned for future extension.

Tools Required

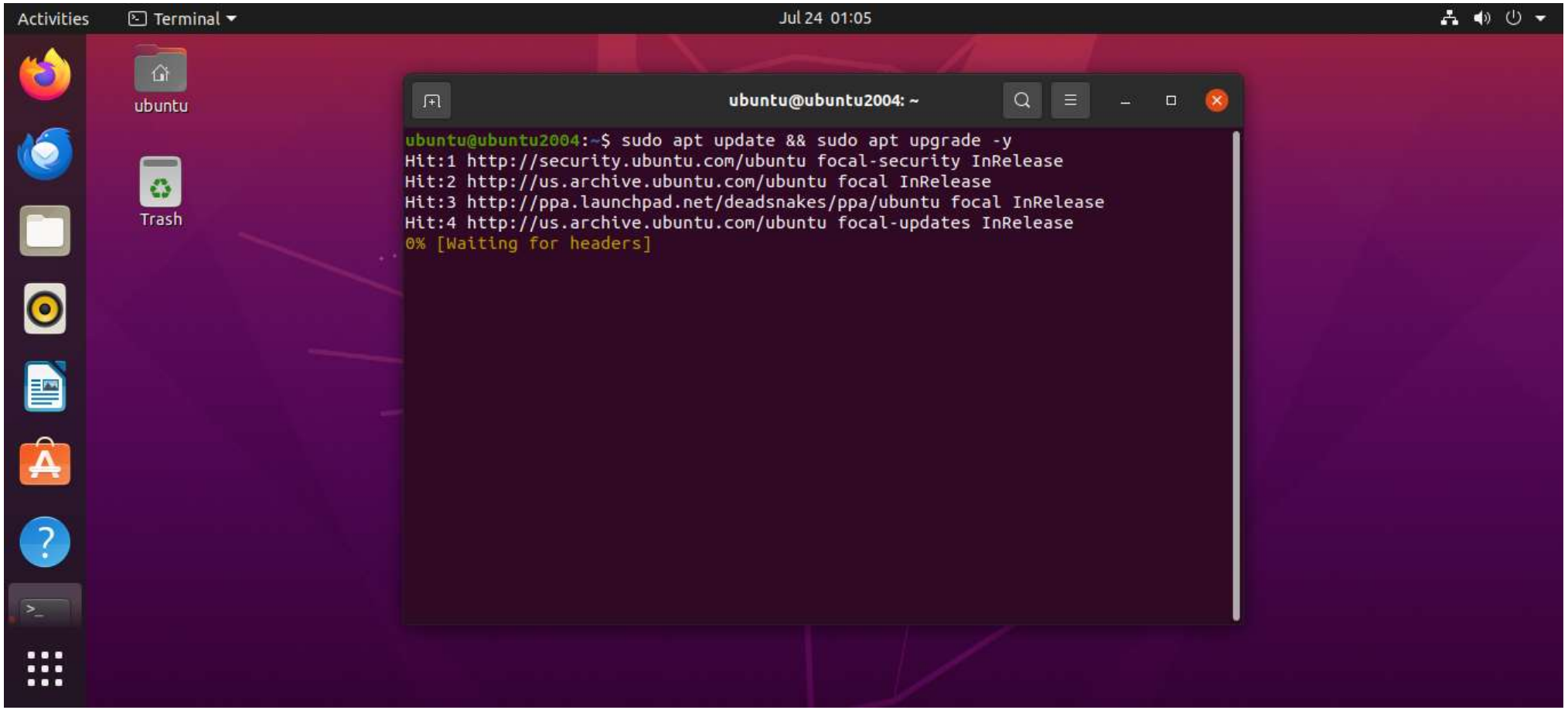
- **VMware Workstation** – For running the Ubuntu virtual machine.
- **Ubuntu 20.04 LTS** – The main OS environment for development and testing.
- **Python 3.x** – Used to build the local HTTPS update server.
- **OpenSSL** – For generating private/public keys and signing firmware.
- **MCUBoot** – Bootloader to verify firmware signature (optional in your setup).
- **Zephyr SDK** – If using MCUBoot and Zephyr OS for secure boot (optional).
- **Curl / Wget** – For testing firmware downloads over TLS.
- **Text Editor (VS Code / nano)** – For editing Python code and configs.

💡 Practical Steps:

- **1. Install Zephyr Environment**
- **2. Create Project Folder**
- **3. Enable MCUBoot**
- **4. Build & Sign Firmware**
- **5. Run with QEMU**

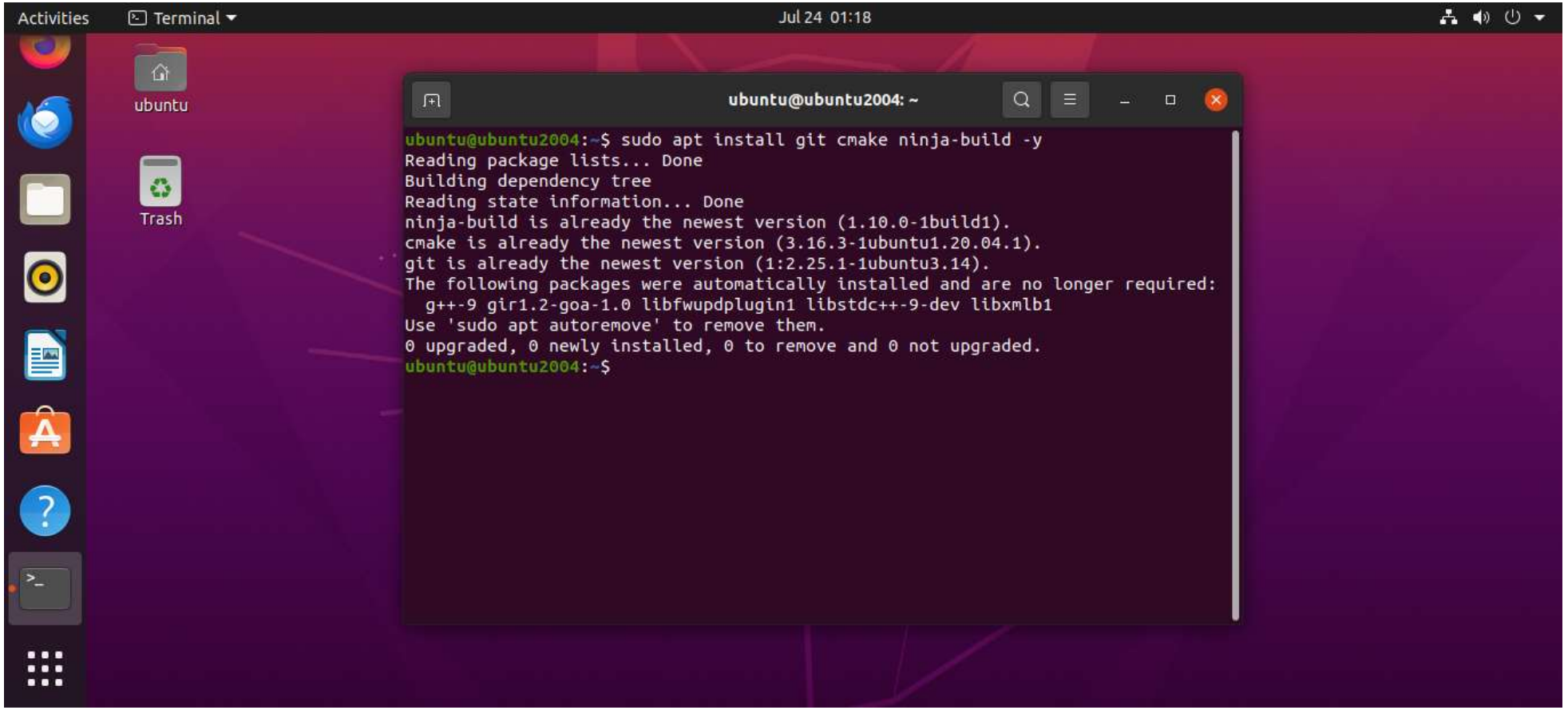
Install Zephyr Environment

$ sudo apt update && sudo apt upgrade -y



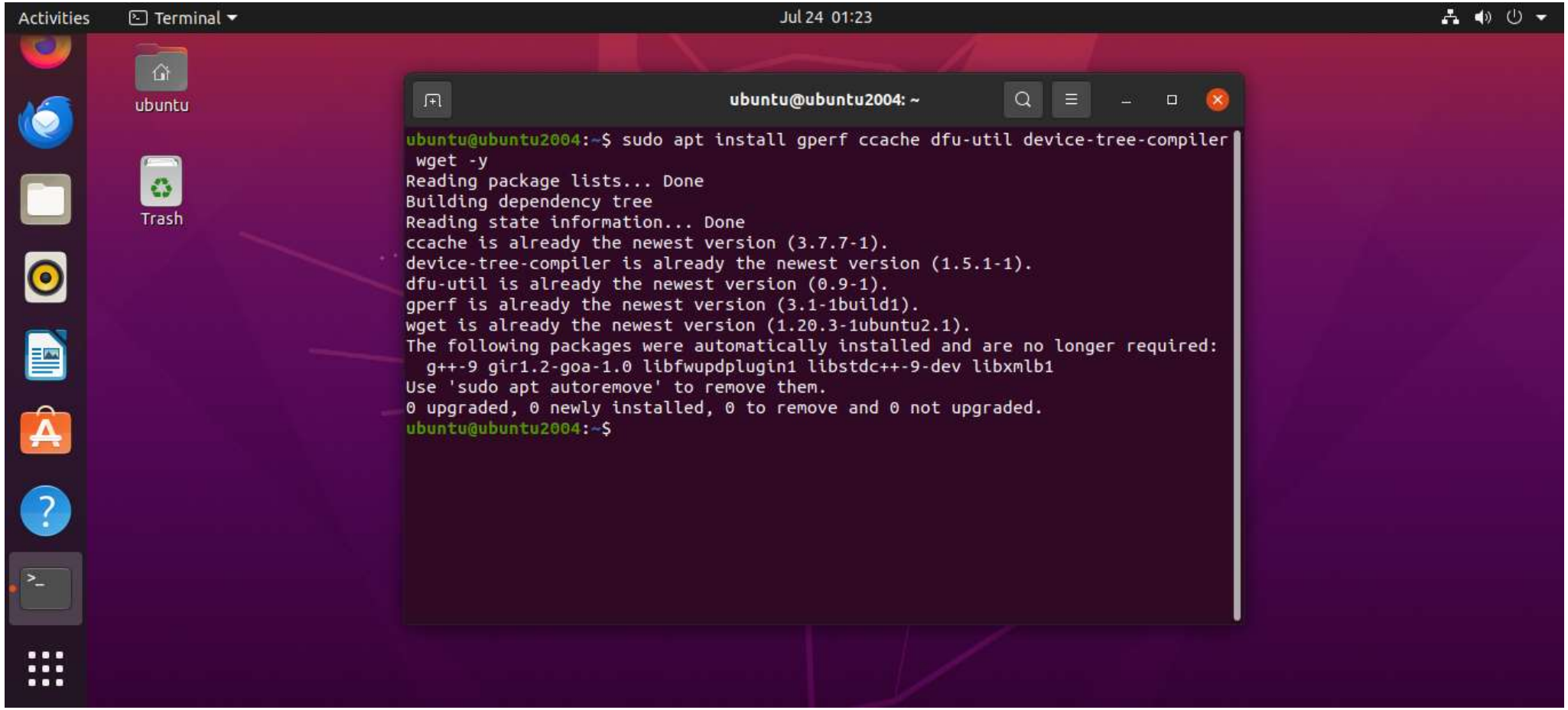
➢ Basic tools

$ sudo apt install git cmake ninja-build -y



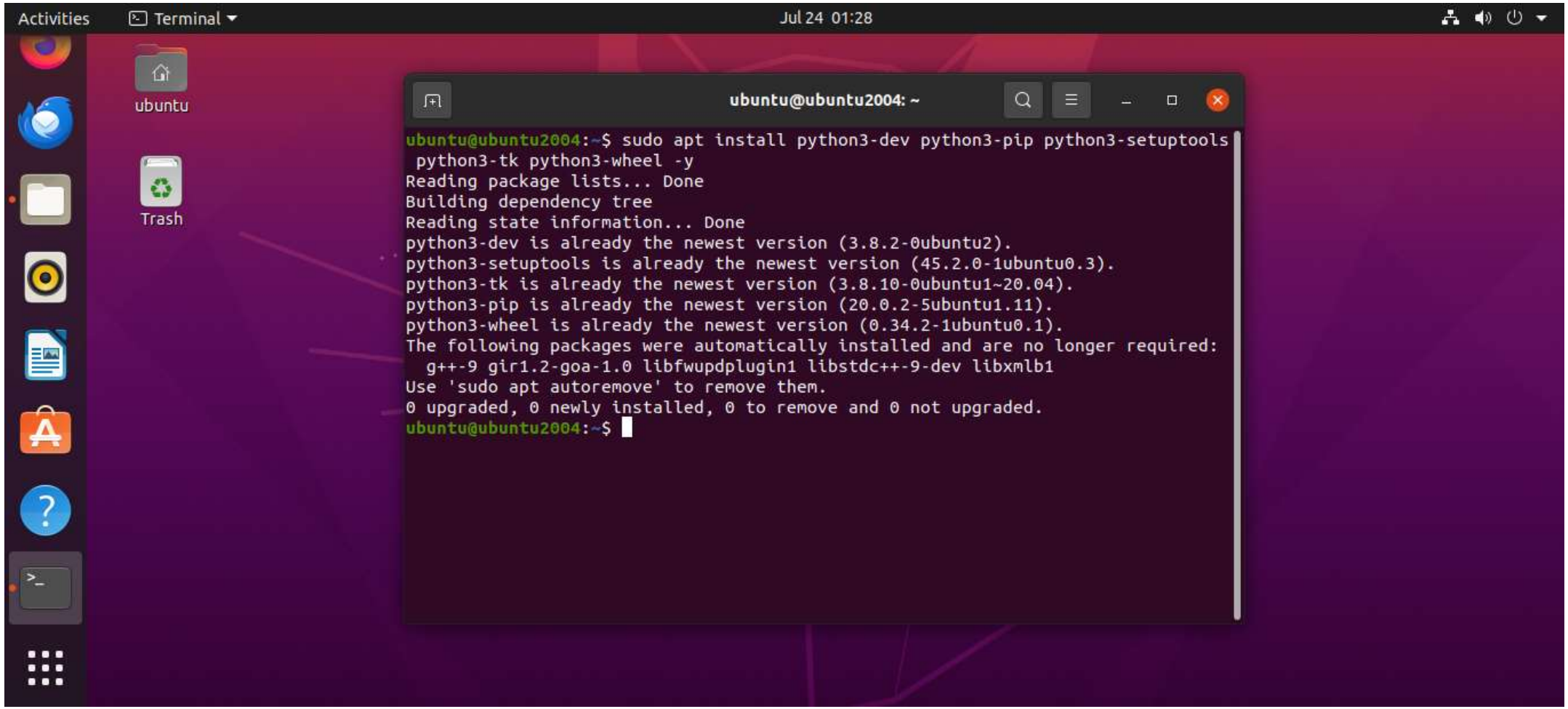
➢ Development tools

$ sudo apt install gperf cache dfu-util device-tree-compiler wget -y



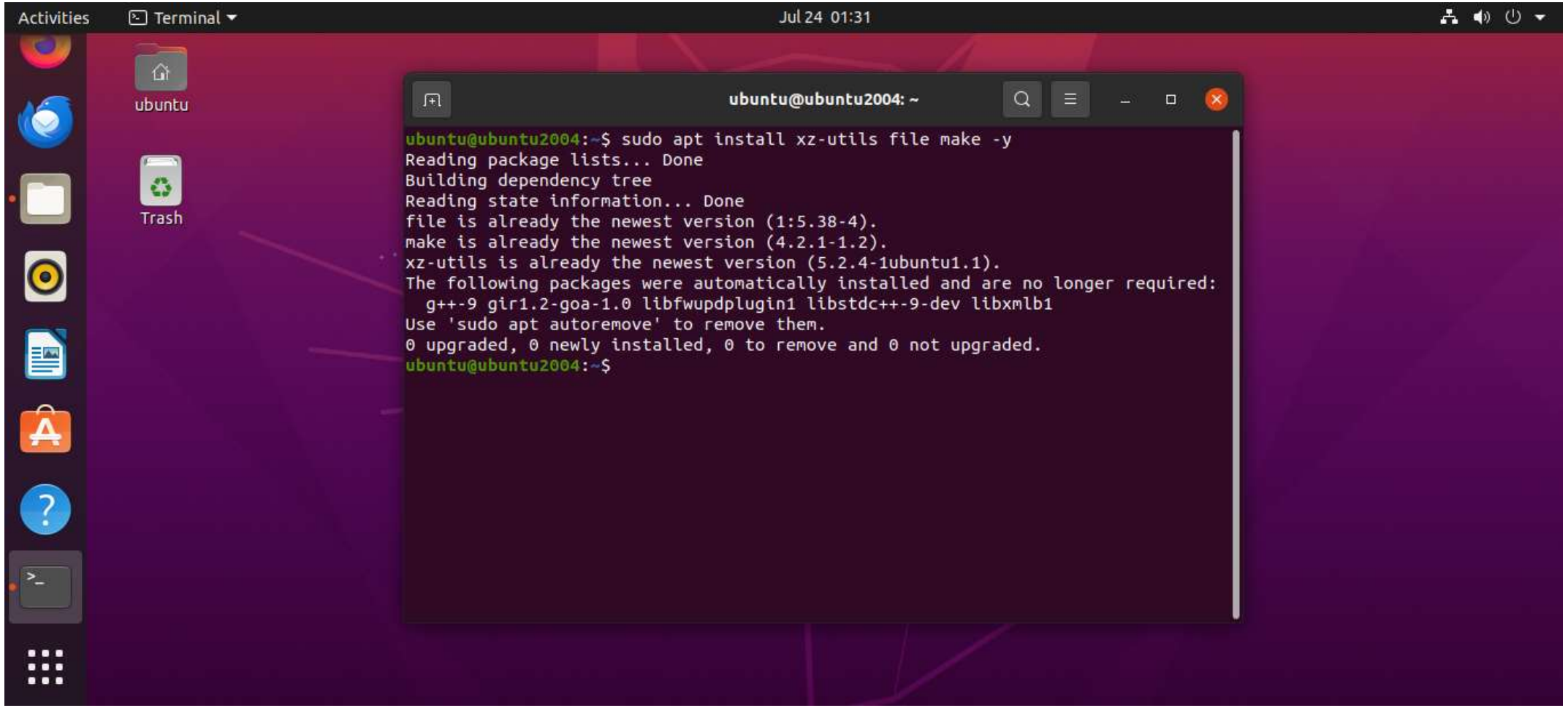
➢ Python related

$ sudo apt install python3-dev python3-pip python3-setuptools python3-tk python3-wheel -y



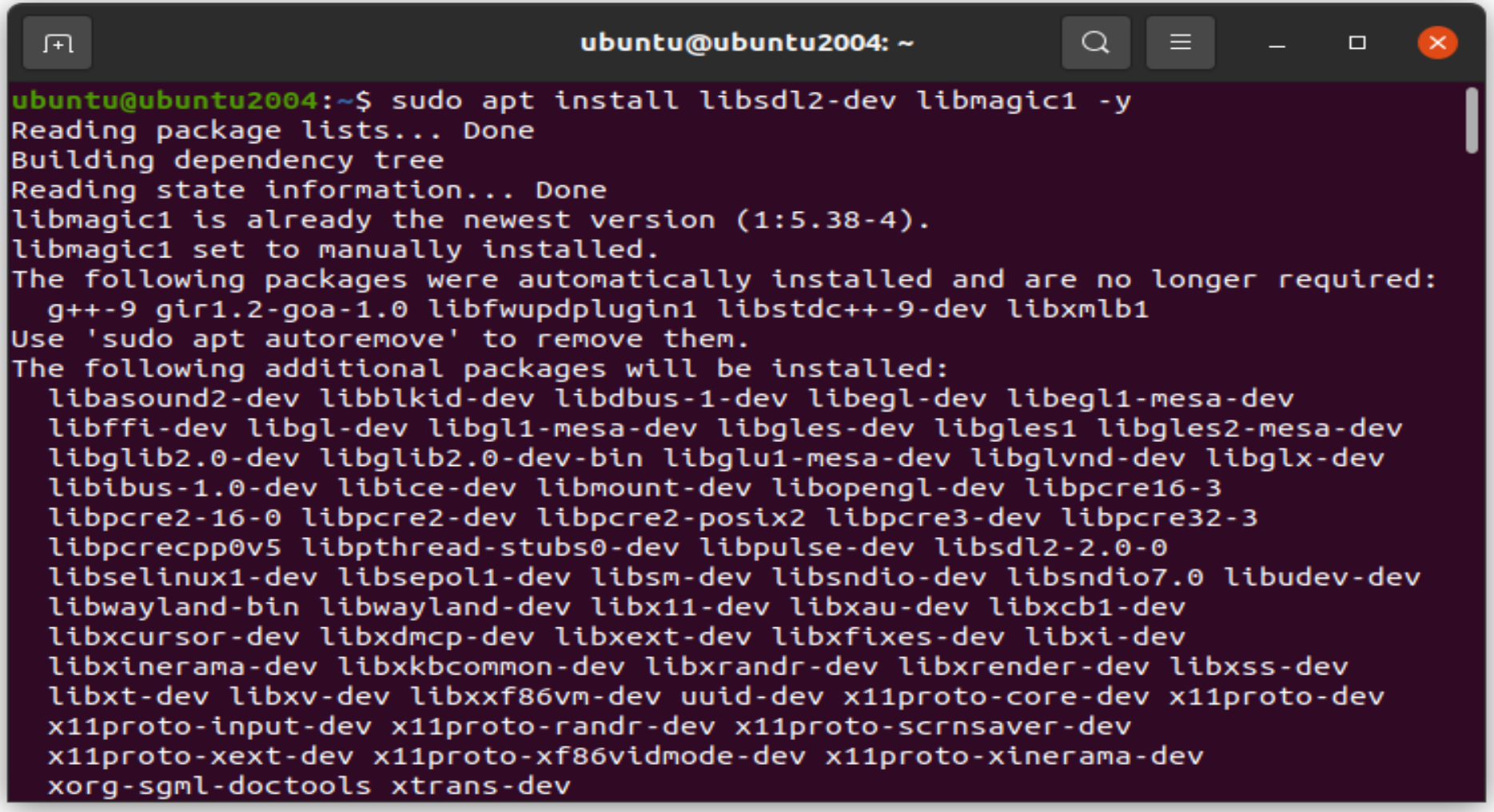
➢ Utils

$ sudo apt install xz-utils file make -y



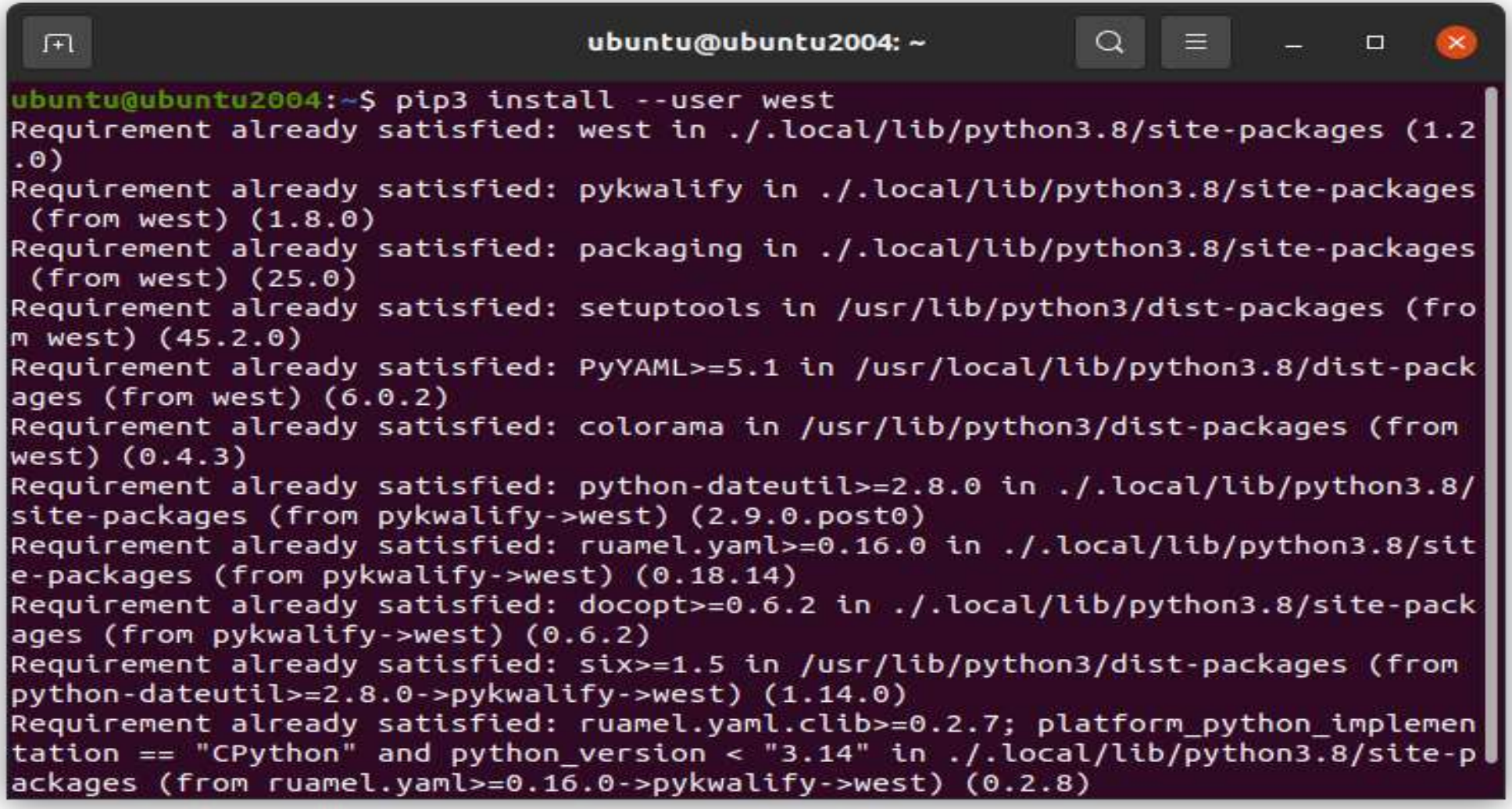
➢ Compilers

$ sudo apt install libsdl2-dev libmagic -y



➢ Install west (Zephyr's build tool)

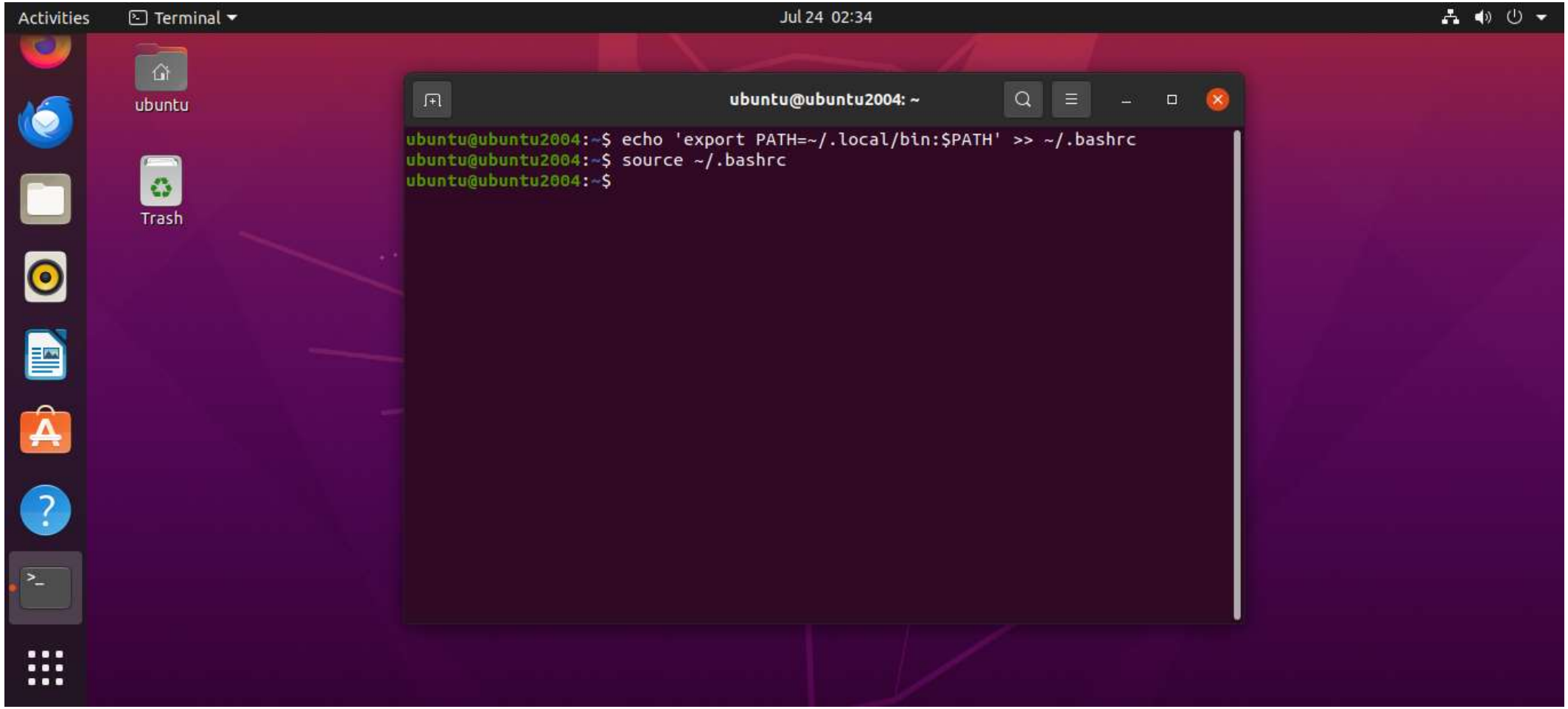
$ pip3 install --user west



➢ Export PATH

$ echo 'export PATH=~/.local/bin:$PATH' >> ~/.bashrc

$ source ~/.bashrc

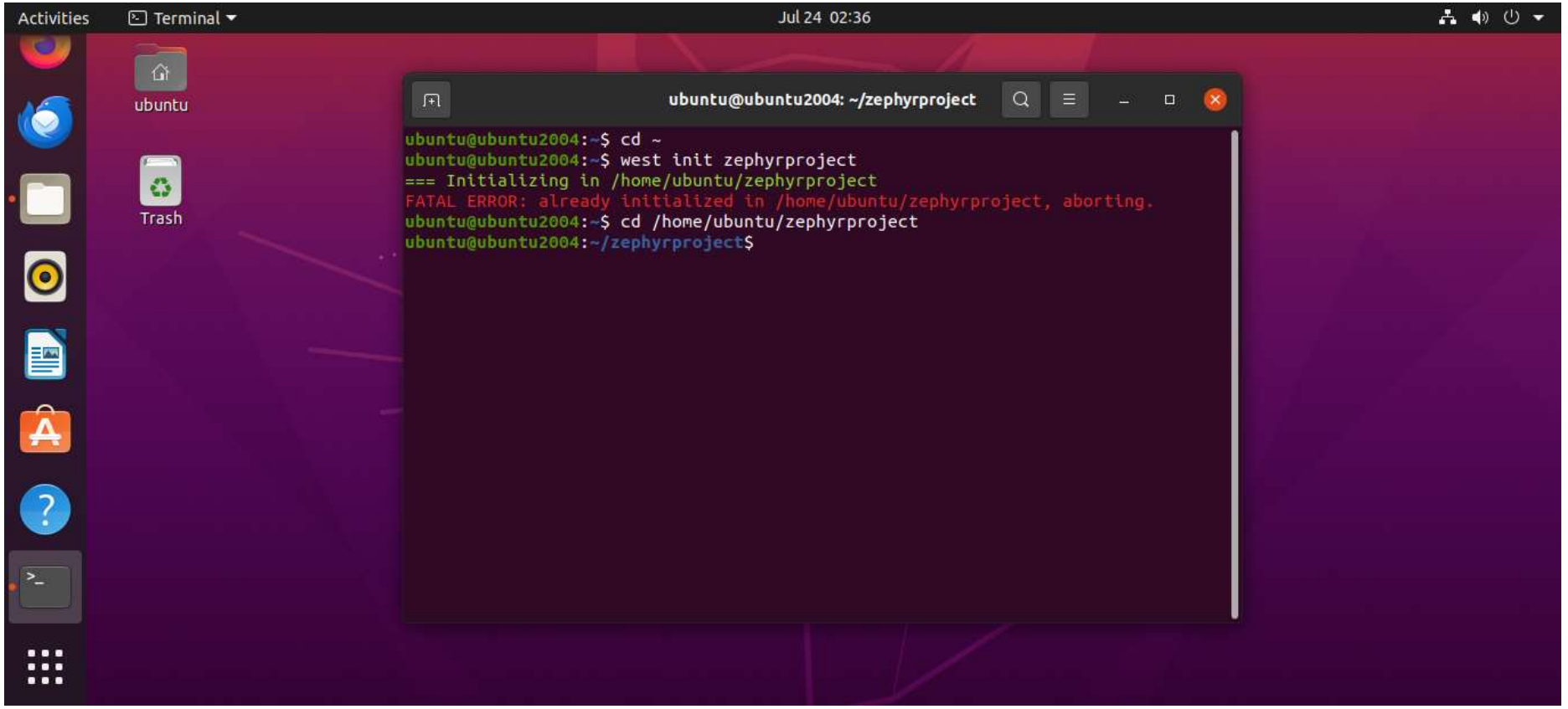


➢ Clone zephyrproject

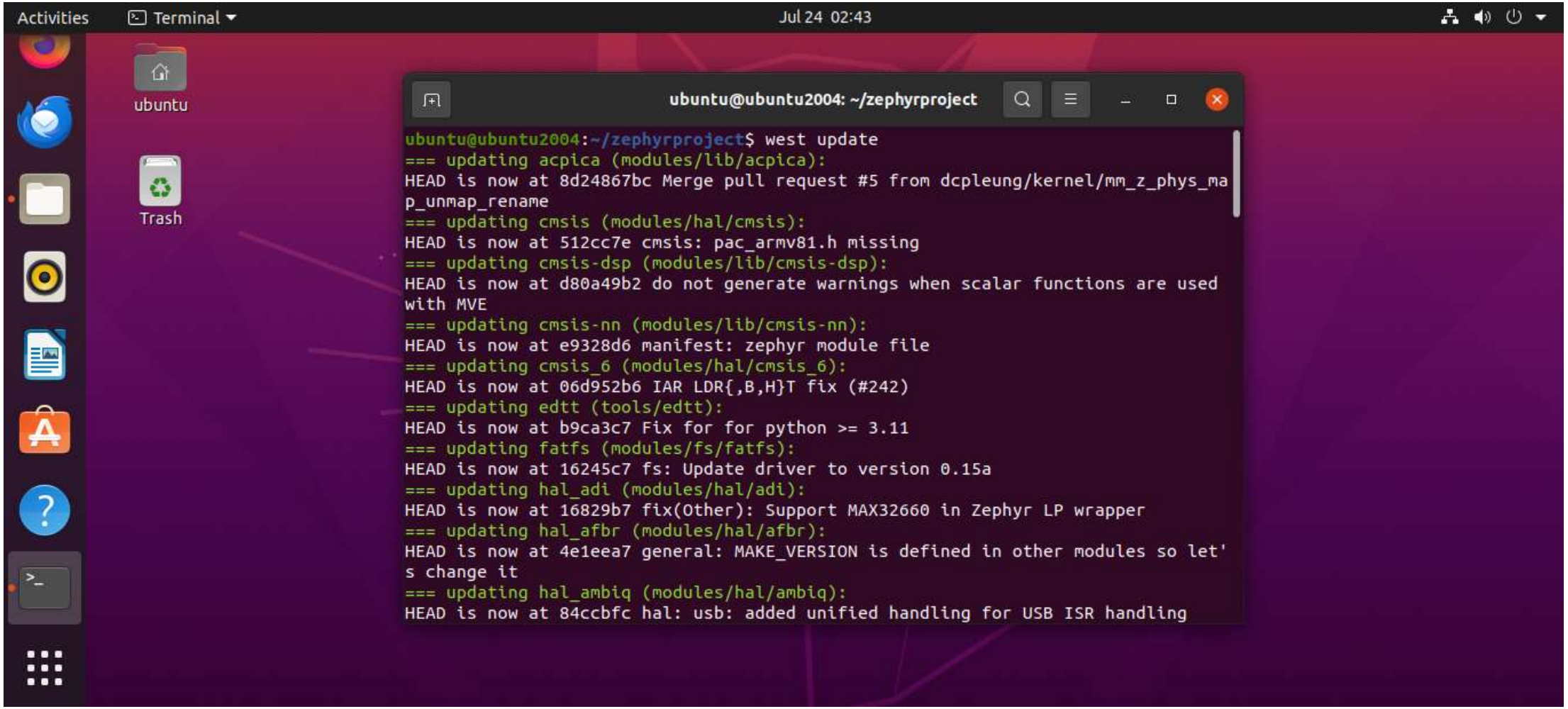
$ west init zephyrproject

FATAL ERROR: already initialized in /home/ubuntu/zephyrproject, aborting.

$ cd /home/ubuntu/zephyrproject

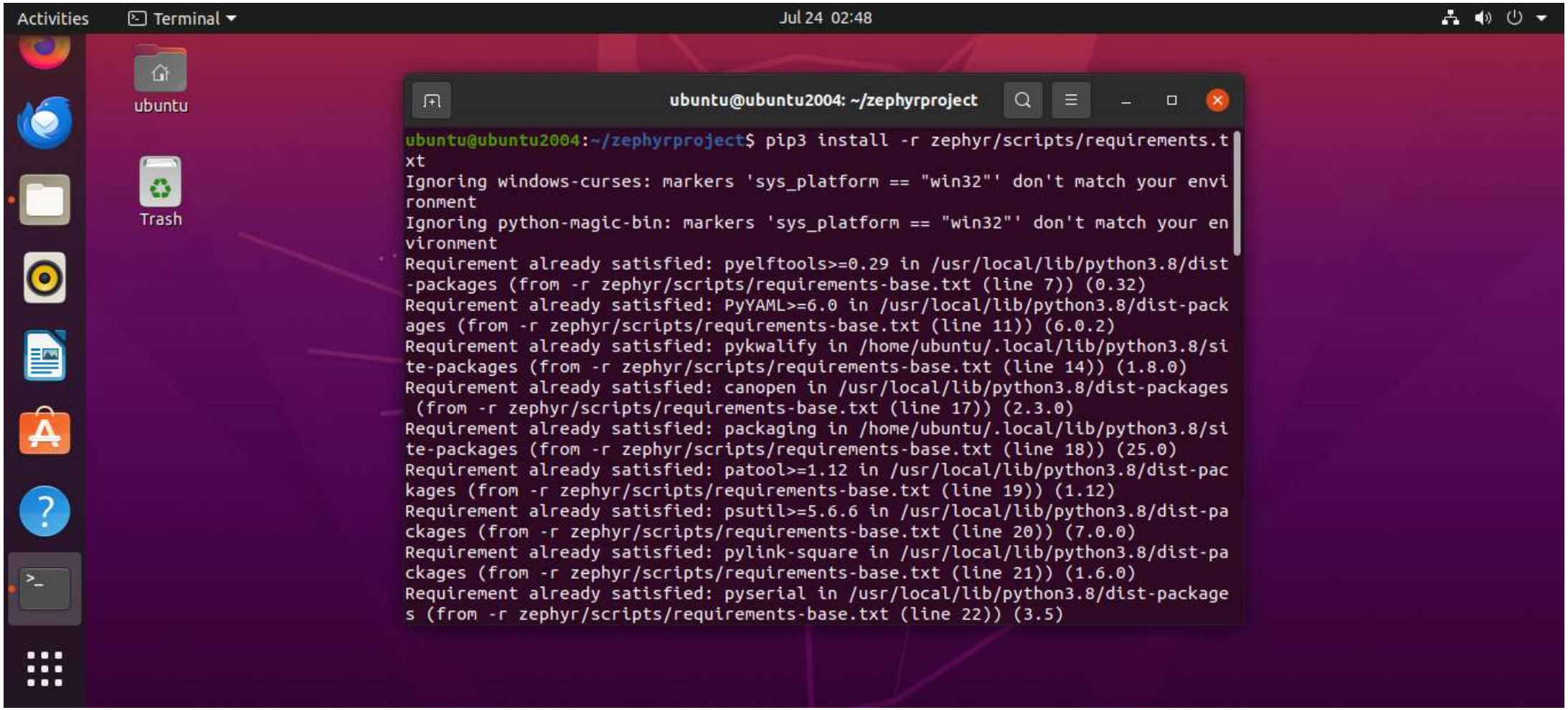


$ cd zephyrproject
$ west update



➢ Install Python dependencies

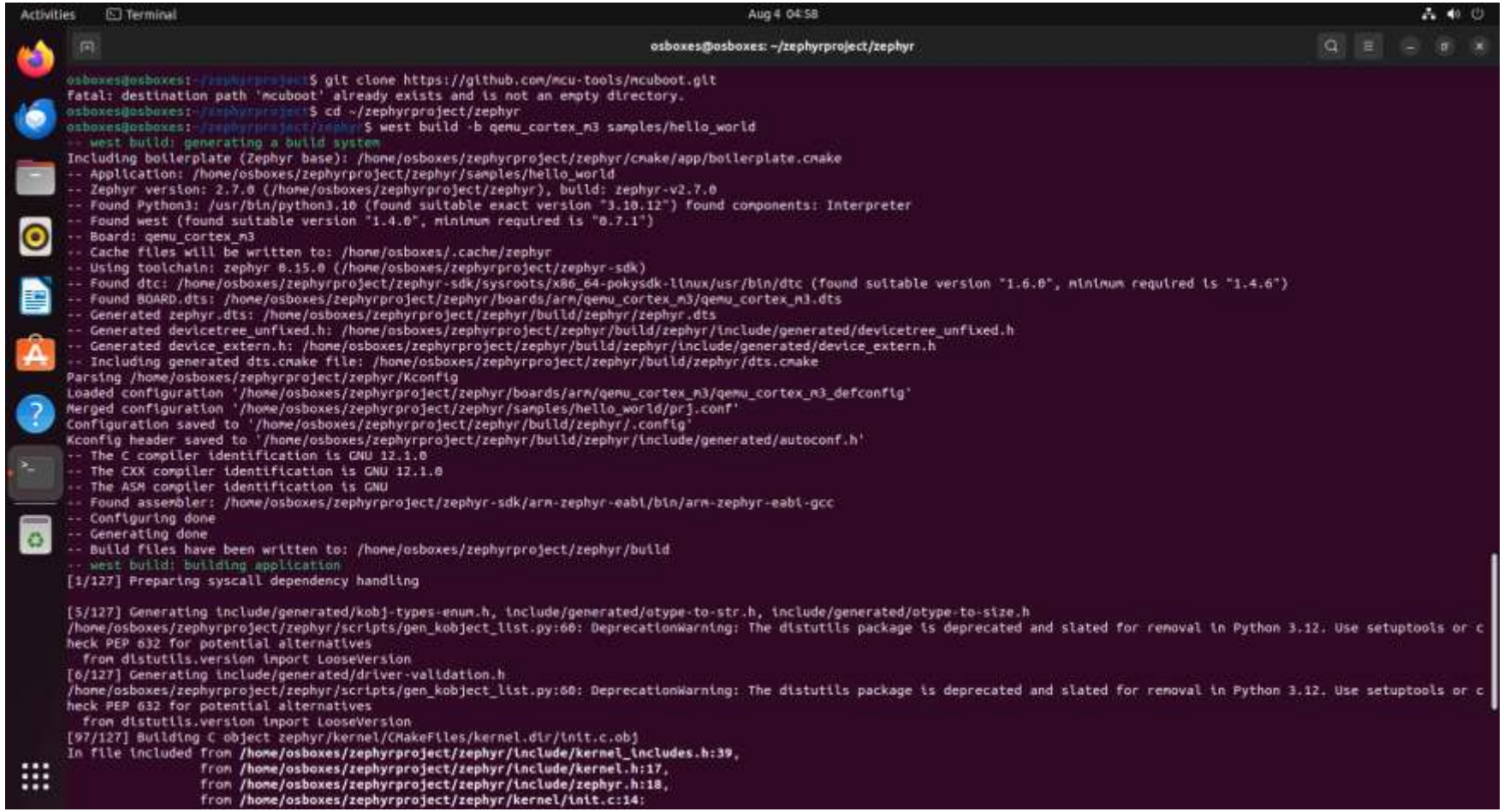
$ pip3 install -r zephyr/scripts/requirements.txt



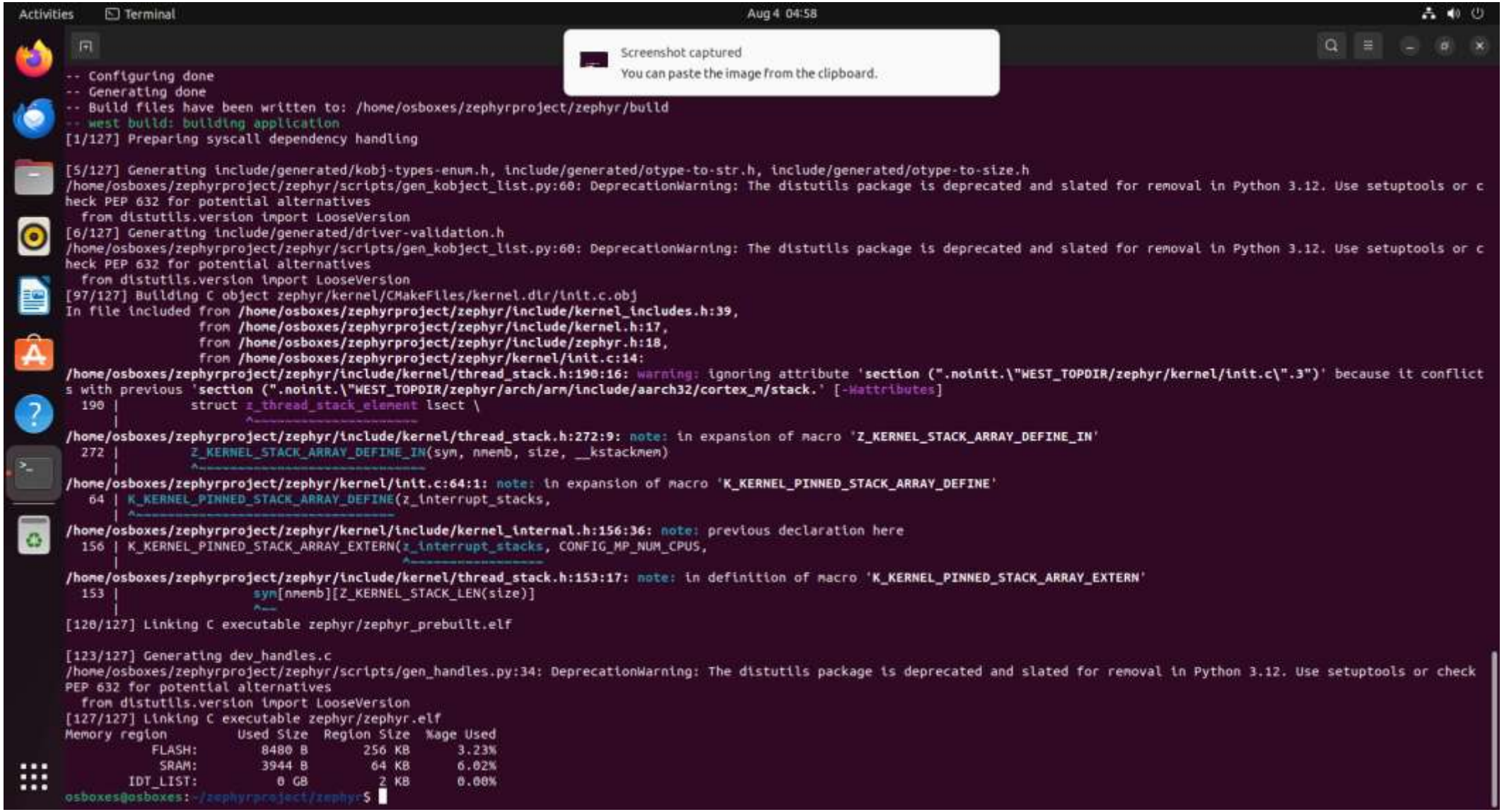
➢ Implement Secure Boot

- MCUBoot Clone

$ git clone https://github.com/mcu-tools/mcuboot.git
$ cd ~/zephyrproject/zephyr
$ west build –b qemu_cortex_m3 sample/hello_world

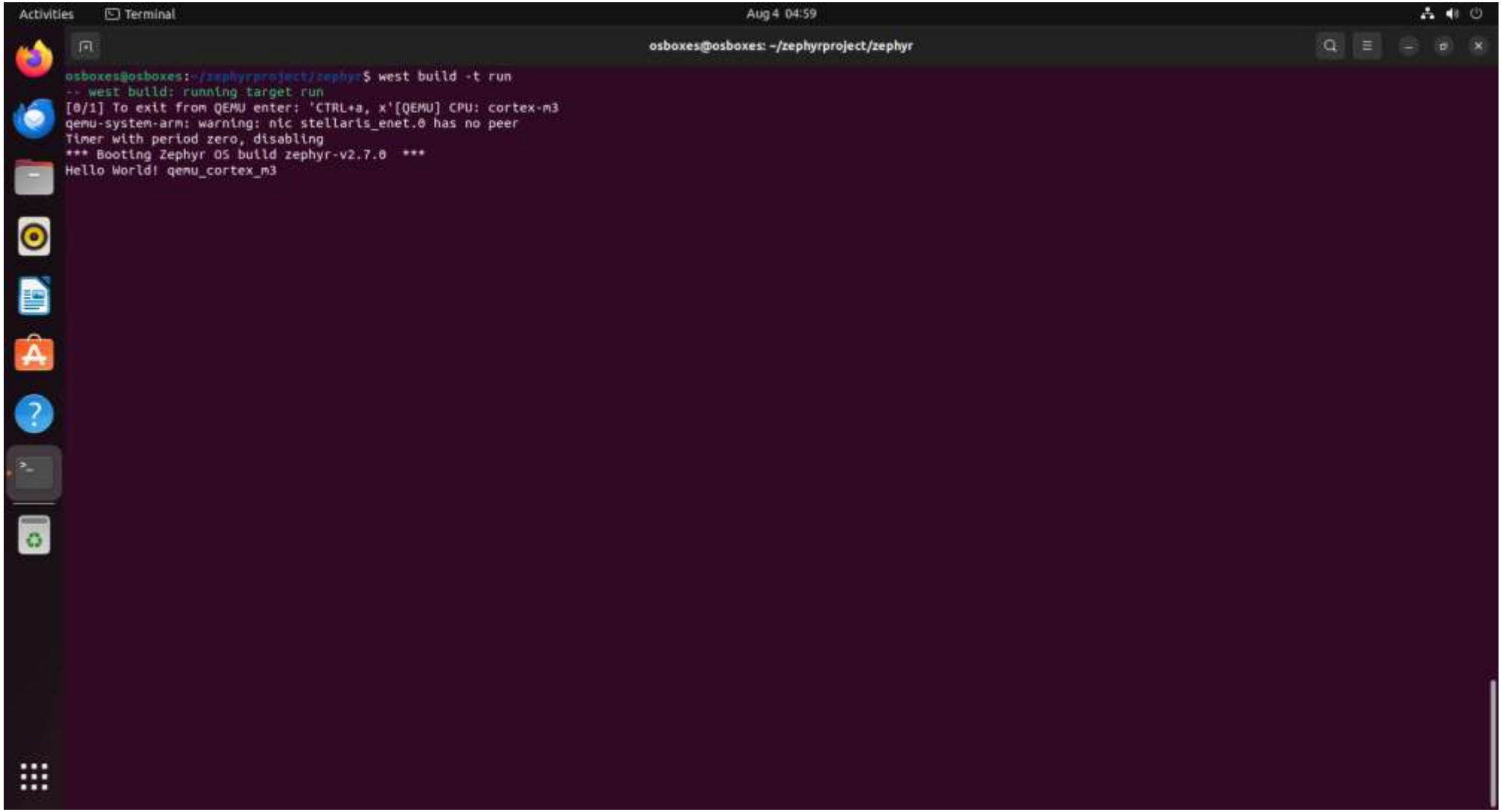


➢ Implement Secure Boot Success



Implement Secure Boot Success Run

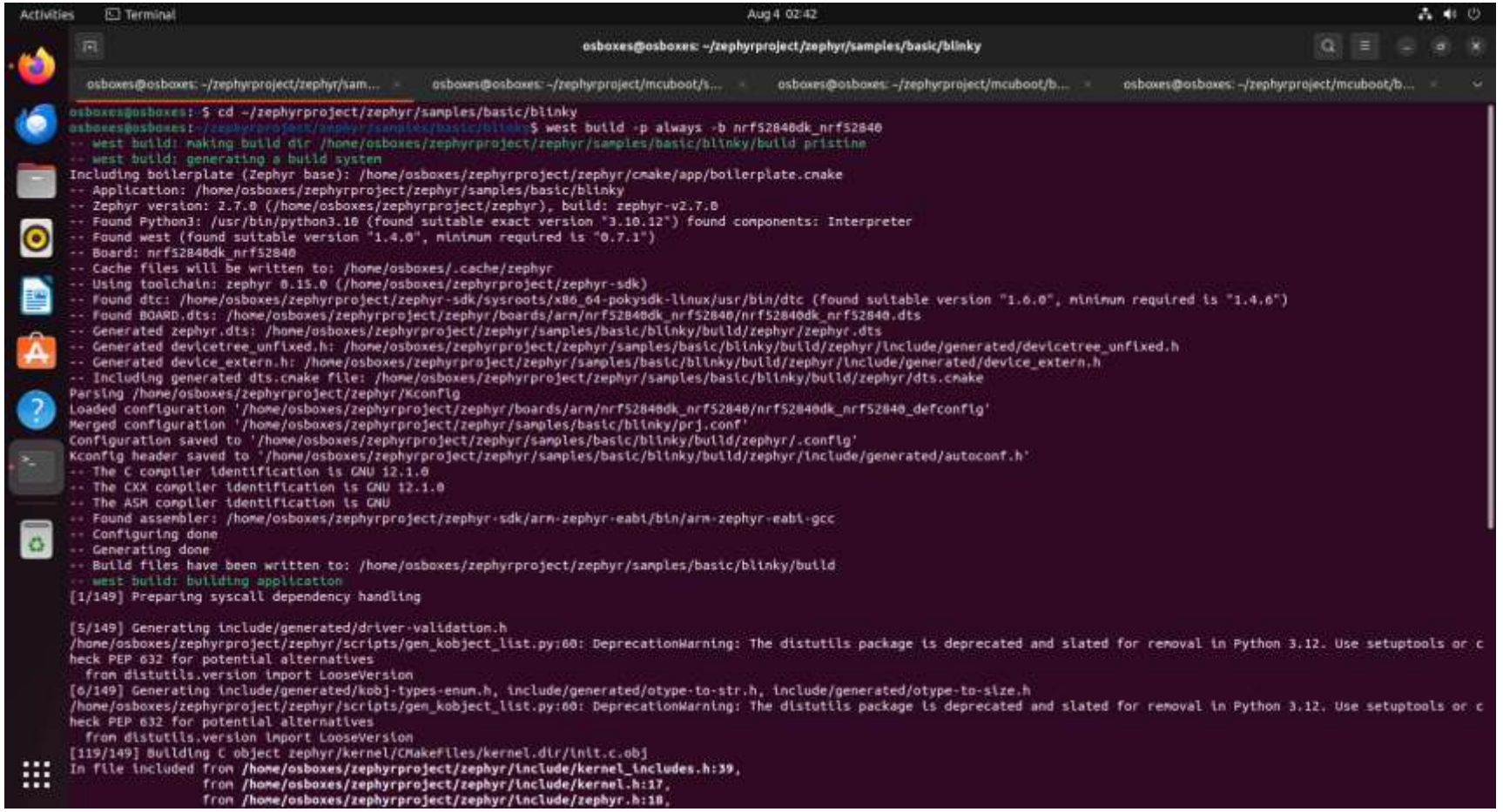
$ west build –t run



➢ MCUboot Build

$ cd ~/zephyrproject/zephyr/samples/basic/blinky

$ west build –p always –b nrf52840dk_nrf52840

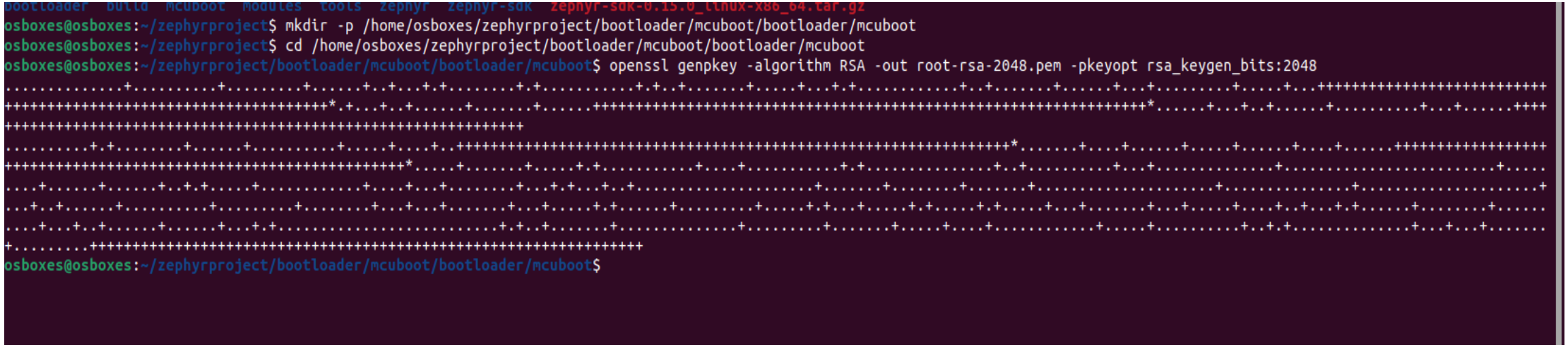


➢ Private Key Generate

$ mkdir –p /zephyroject/bootloader/mcuboot/bootloader/mcuboot

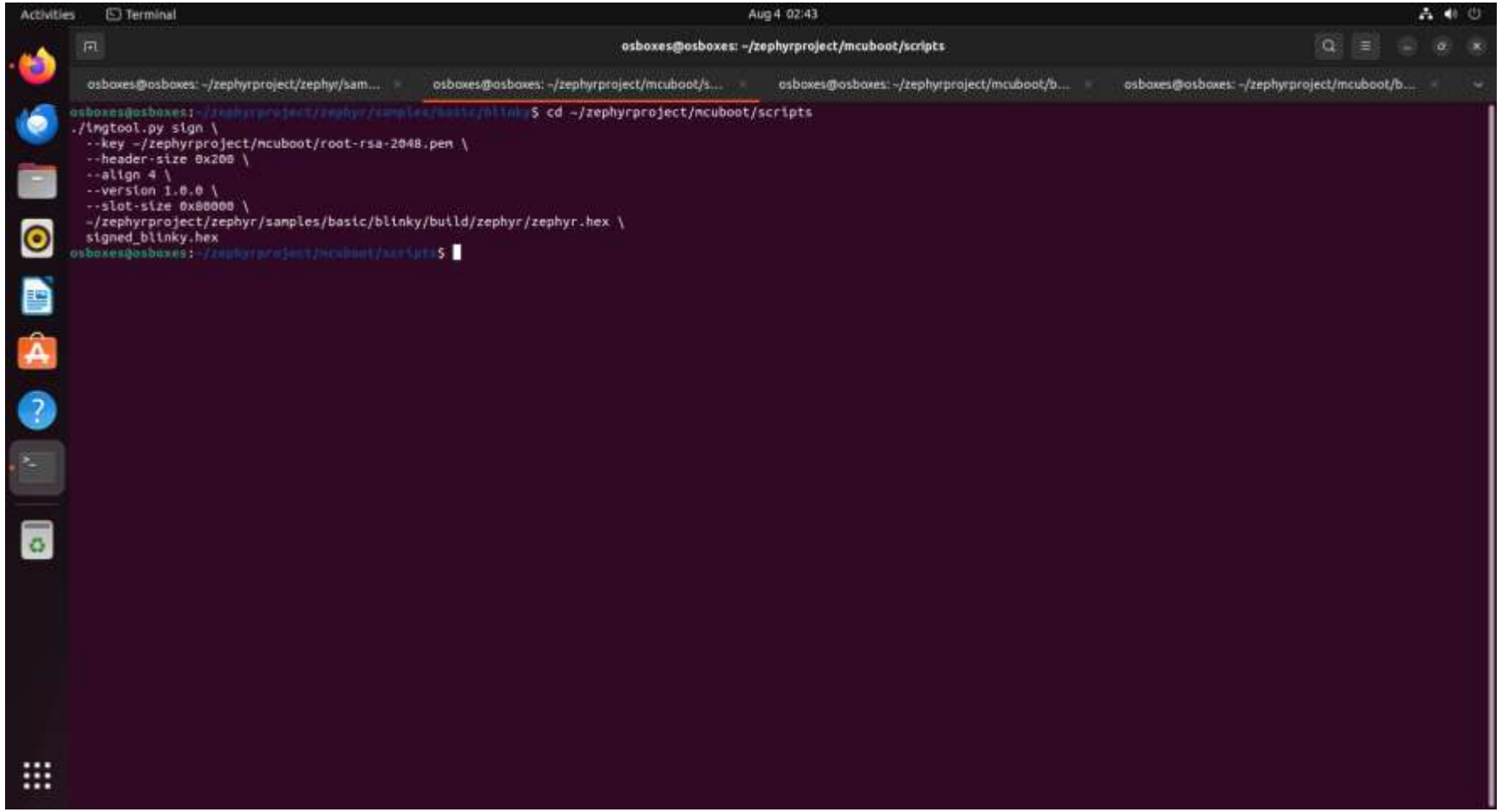
$ cd /zephyroject/bootloader/mcuboot/bootloader/mcuboot

$ openssl genpkey –algorithm RSA –out root-rsa-2848.pem –pkeyopt rsa_keygen_bits:2048



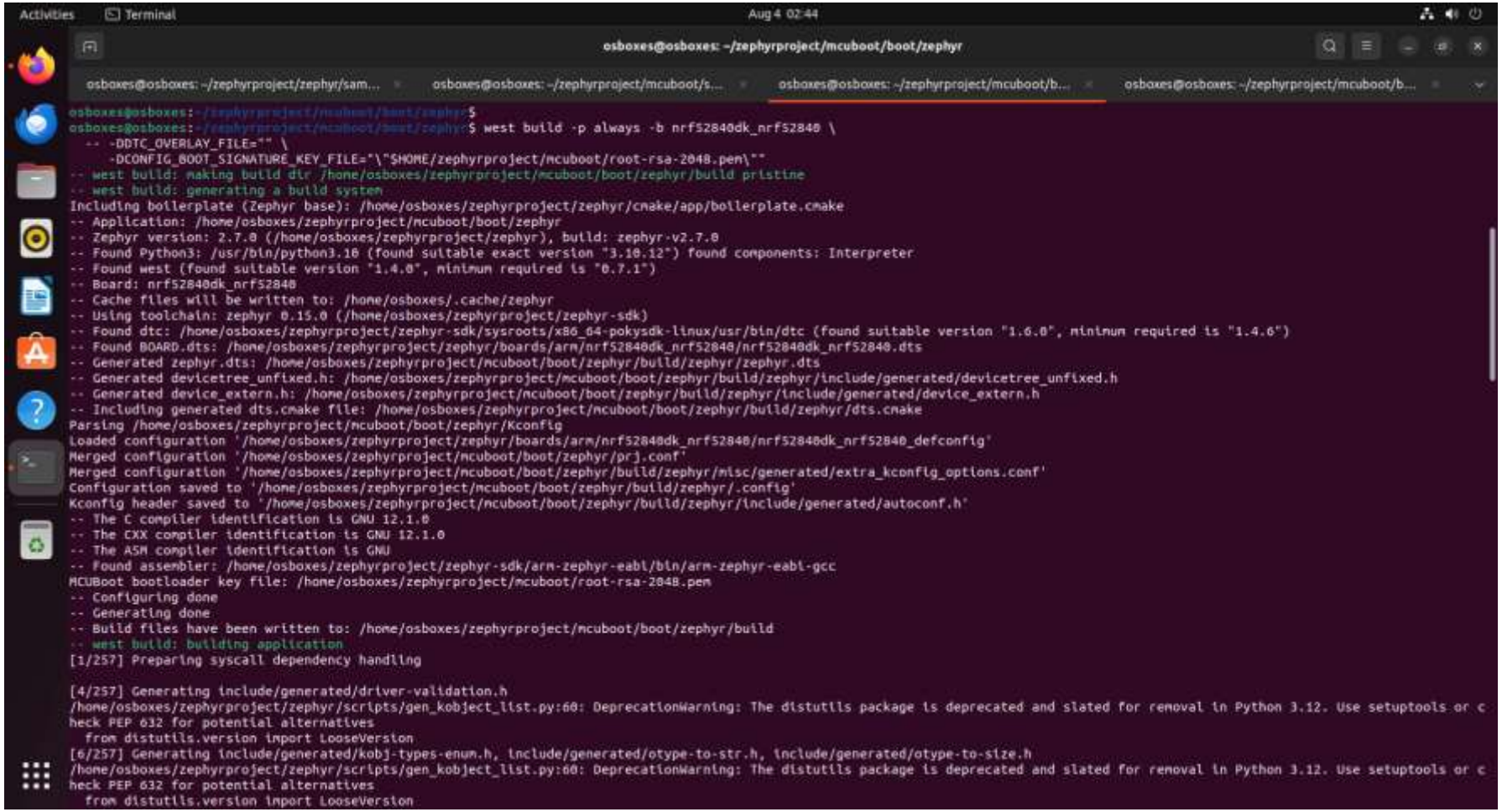
➢ Firmware Sign

$ cd /zephyrproject/mcuboot/scripts

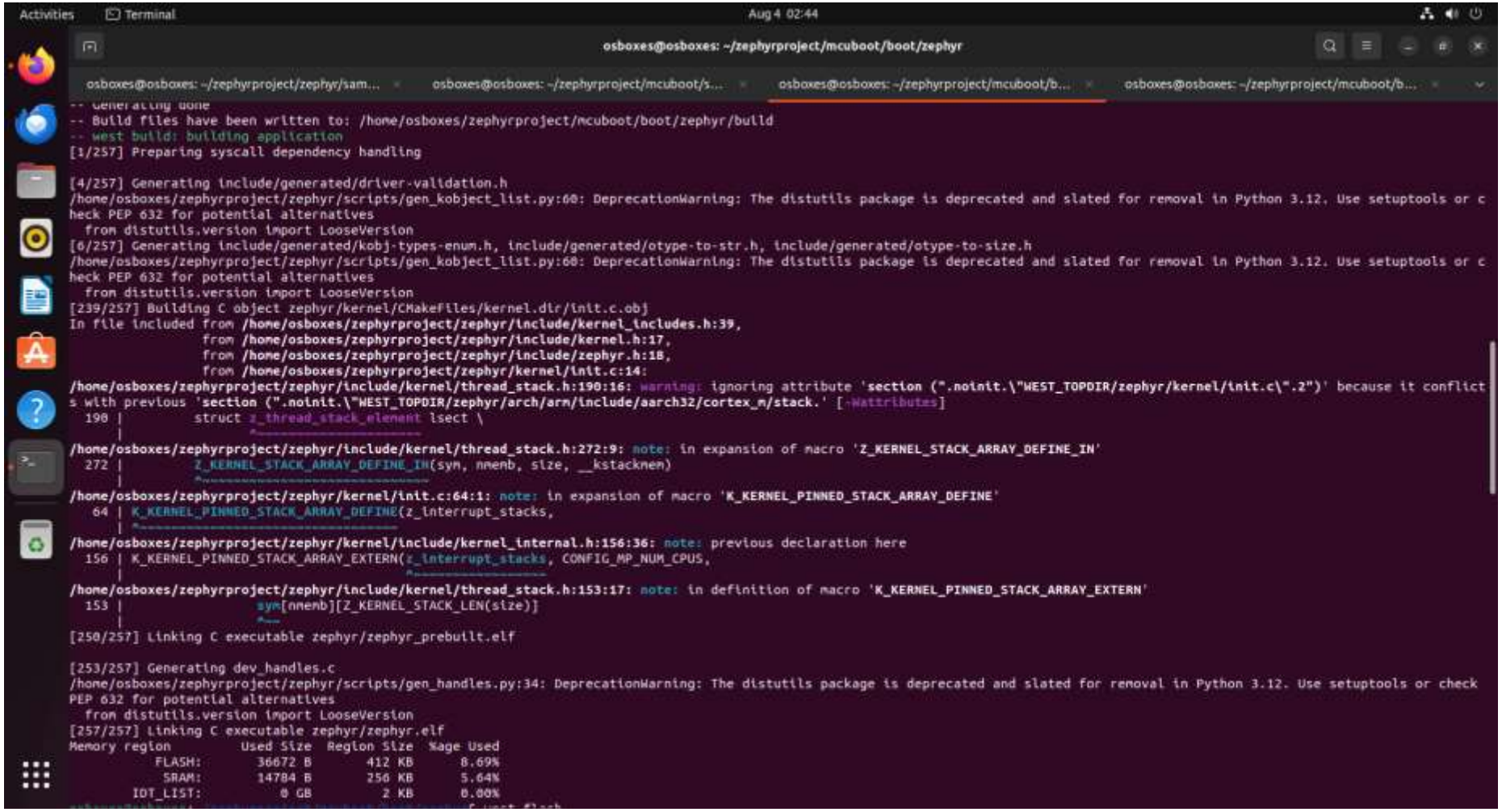


➢ RUN With nzf

$ west build –p always –b nrf52840dk_nrf52840 \



➢ Successful


Activities
Terminal
Aug 4 02:44
osboxes@osboxes: ~/zephyrproject/mcuboot/boot/zephyr
-- Build files have been written to: /home/osboxes/zephyrproject/mcuboot/boot/zephyr/build
-- west build: building application
[1/257] Preparing syscall dependency handling
[4/257] Generating include/generated/driver-validation.h
/home/osboxes/zephyrproject/zephyr/scripts/gen_kobject_list.py:60: DeprecationWarning: The distutils package is deprecated and slated for removal in Python 3.12. Use setuptools or check PEP 632 for potential alternatives
from distutils.version import LooseVersion
[6/257] Generating include/generated/kobj-types-enum.h, include/generated/otype-to-str.h, include/generated/otype-to-size.h
/home/osboxes/zephyrproject/zephyr/scripts/gen_kobject_list.py:60: DeprecationWarning: The distutils package is deprecated and slated for removal in Python 3.12. Use setuptools or check PEP 632 for potential alternatives
from distutils.version import LooseVersion
[239/257] Building C object zephyr/kernel/CMakeFiles/kernel.dir/init.c.obj
In file included from /home/osboxes/zephyrproject/zephyr/include/kernel_includes.h:39,
from /home/osboxes/zephyrproject/zephyr/include/kernel.h:17,
from /home/osboxes/zephyrproject/zephyr/include/zephyr.h:18,
from /home/osboxes/zephyrproject/zephyr/kernel/init.c:14:
/home/osboxes/zephyrproject/zephyr/include/kernel/thread_stack.h:190:16: warning: ignoring attribute 'section (".noinit.\"WEST_TOPDIR/zephyr/kernel/init.c\".2")' because it conflicts with previous 'section (".noinit.\"WEST_TOPDIR/zephyr/arch/arm/include/aarch32/cortex_m/stack.' [-Wattributes]
190 | struct z_thread_stack_element lsect \
/home/osboxes/zephyrproject/zephyr/include/kernel/thread_stack.h:272:9: note: in expansion of macro 'Z_KERNEL_STACK_ARRAY_DEFINE_IN'
272 | Z_KERNEL_STACK_ARRAY_DEFINE_IN(sym, nmemb, size, __kstacknen)
/home/osboxes/zephyrproject/zephyr/kernel/init.c:64:1: note: in expansion of macro 'K_KERNEL_PINNED_STACK_ARRAY_DEFINE'
64 | K_KERNEL_PINNED_STACK_ARRAY_DEFINE(z_interrupt_stacks,
/home/osboxes/zephyrproject/zephyr/kernel/include/kernel_internal.h:156:36: note: previous declaration here
156 | K_KERNEL_PINNED_STACK_ARRAY_EXTERN(z_interrupt_stacks, CONFIG_MP_NUM_CPUS,
/home/osboxes/zephyrproject/zephyr/include/kernel/thread_stack.h:153:17: note: in definition of macro 'K_KERNEL_PINNED_STACK_ARRAY_EXTERN'
153 | sym[nmemb][Z_KERNEL_STACK_LEN(size)]
[250/257] Linking C executable zephyr/zephyr_prebuilt.elf
[253/257] Generating dev_handles.c
/home/osboxes/zephyrproject/zephyr/scripts/gen_handles.py:34: DeprecationWarning: The distutils package is deprecated and slated for removal in Python 3.12. Use setuptools or check PEP 632 for potential alternatives
from distutils.version import LooseVersion
[257/257] Linking C executable zephyr/zephyr.elf
Memory region Used Size Region Size %age Used
FLASH: 36672 B 412 KB 8.69%
SRAM: 14784 B 256 KB 5.64%
IDT_LIST: 0 GB 2 KB 0.00%

Limitations & Future Scope

Limitations:

- Due to the absence of a physical IoT device or full QEMU emulation setup, the secure boot (MCUboot) verification could not be tested end-to-end.
- The project was developed entirely within a VMware Ubuntu environment, and some Zephyr components required newer dependencies that were not compatible with the current system.
- Time constraints and lack of direct mentorship during the internship limited deeper experimentation.

Independent Project Execution

- This project was completed **without any assigned mentor or hands-on guidance.**
- All tools, concepts, and implementation techniques were learned through **self-research, official documentation, and online resources.**
- Despite facing several technical roadblocks, including environment setup errors and the absence of physical IoT hardware, the core project objectives were achieved.
- The experience strengthened my ability to:
- Work independently on complex cybersecurity topics
- Troubleshoot technical problems using community knowledge
- Simulate real-world scenarios in a virtual lab environment

This internship has been a valuable self-driven learning journey in secure firmware mechanisms and IoT security.