Ankit Khushwaha

+91-7428061540 ankitkhushwaha.dev@gmail.com linkedin github blog

Education

Indian Institute Of Technology Dharwad **Dec. 2022 – Present**
Bachelor of Science-Master of Science, Physics *Dharwad, Karnataka*

Technical Skills

Languages: C, C++, Python

Systems & Tools: Linux, QEMU, U-boot, GDB, GNU Make, Kbuild, Git

Core Skills: Linux Device Driver, Kernel Modules, Kernel Debugging, Device Tree, System Programming

Kernel Subsystems: I2C, Platform/Custom Bus, Character Drivers, Sysfs, Procfs, Timers, Workqueues, Interrupt Handling, Synchronization

Familiar With: Shell Scripting, Computer Architecture, Web Development

Experience

Linux Foundation Mentorship (LFX '25) - Linux Kernel **Sept 2025 - Dec 2025**
Mentee *Upstream Linux Kernel Patches*

- Contributed 10+ patches across Linux kernel selftests in net, mm, tls, mptcp, and user_events subsystems.
- Analyzed syzbot bugs using QEMU and analyzed execution paths with ftrace & gdb to identify root causes.
- Performed memory error analysis using KASAN to detect and fix uninitialized variables and invalid memory usage.
- Collaborated with mentors and upstream contributors through mailing-list discussions, code reviews, & patch revisions.

Acadia Ventures **July 2025 - Aug 2025**
Research and Development Intern *Remote*

- Built a structured testing framework with fixtures to manage inter-test dependencies, streamlining test maintenance and enabling faster debugging cycles.
- Implemented secure JWT-based authentication with role-based access, strengthening API security and control.
- Built notification system using event triggers and async tasks, improving user engagement with timely alerts.

Projects

Linux I2C EEPROM Driver
C, Linux Kernel, I2C, Device Driver

- Developed a Linux **I2C client driver** for an EEPROM device, implementing **probe/remove** lifecycle and device–driver matching via the **I2C ID table**.
- Implemented byte-level and page-aware **read/write** access over the I2C bus using the kernel **I2C transfer APIs**.
- Exposed EEPROM contents to user space through a **sysfs/character device interface** with proper **bounds checking and error handling**.
- Protected concurrent accesses with **mutex-based locking** and handled write-cycle delays and transfer failures for reliable operation.
- Validated the driver with **beaglebone black** and debugged using kernel logs and **dynamic debug**.

Mini SoC Device Model
C, Linux Kernel, Device Model, Makefile

- Built a simulated **SoC sensor hub** as three kernel modules (bus, device, driver) implementing a custom `bus_type` modelled on the Linux **platform bus**.
- Implemented **device/driver matching, probe/remove, and sysfs unbinding**, along with `struct device` lifecycle management .
- Added **Device Tree** support by binding `of_node` to devices with `device_set_node()` and populating and tearing down devices from DT nodes
- Tested on **ARM QEMU** with a cross-compiled kernel and debugged with **gdb** using kernel GDB scripts.

Certifications

The Linux Foundation **Issued Aug 2025**
LFD103: A Beginner's Guide to Linux Kernel Development *Credly*

Udemy **Issued July 2026**
LDD1: Linux Device Driver Programming With Beaglebone Black *Udemy*