Pavan Kumar

Embedded Linux & Firmware Engineer

pavannandikanti989@gmail.com 8179564921 Hyderabad, India

Profile

Embedded Linux and Firmware Engineer with 1+ year of hands-on experience in **Embedded Linux driver** development on **ARM Cortex-A72** and **bare-meta**l firmware development on **ARM Cortex-M (STM32)** platforms. Experienced in developing register-level peripheral drivers for **UART, I2C, SPI, GPIO, DMA,** timers, **watchdog timers**, and interrupt handling. Proficient in **Linux character device driver development**, **kernel module development**, ioctl-based **user–kernel communication**, and **Bootloader** concepts. Skilled in designing and implementing user–kernel communication using the **read(), write(), ioctl(), copy_to_user(), and copy_from_user() APIs**. Experienced in **board bring-up**, peripheral initialization, kernel debugging using **printk, dmesg,** and **GDB,** as well as hardware debugging using **logic analyzers**, **oscilloscopes,** and the **J-Link** Debugger. Proficient in **Embedded C**, the **GCC toolchain**, cross-compilation, **Makefiles**, **Git**, Keil µVision, **STM32CubeIDE**, Visual Studio Code, **Ubuntu/Linux**, and **Device Tree (DTS/DTB)** basics. Committed to developing reliable, efficient, and scalable embedded software solutions.

Professional Experience

Firmware Developer, Vim Soft Pvt. Ltd., Hyderabad. 12/2024 – Present

- Worked on **UART, I2C, and SPI communication drivers**, ensuring reliable data transfer and protocol compliance between microcontrollers and peripheral devices.
- Performed **STM32 board bring-up** activities, including clock configuration, peripheral initialization, **GPIO configuration**, pin multiplexing, interrupt configuration, and hardware-software integration.
- Developed **register-level firmware modules** for GPIO, UART, I2C, SPI, timers, DMA, and **watchdog timers** using **Embedded C**.
- Debugged low-level firmware and hardware interface issues using **GDB**, **logic analyzers, oscilloscopes,** and the **J-Link Debugger**, ensuring reliable peripheral operation and protocol validation.
- Configured **interrupt-driven and DMA-based communication** to improve data transfer efficiency and reduce CPU utilization.
- Analyzed hardware schematics, microcontroller datasheets, and reference manuals for peripheral configuration, firmware integration, and **board bring-up**.
- Contributed to the development of **modular and reusable firmware APIs** for peripheral communication, improving code maintainability and reusability.
- Integrated peripherals such as **ADC, EEPROM, LCD**, matrix keypad, 7-segment display, **GPIO**, and timers into the firmware architecture.
- Performed **system-level debugging**, validation, and functional testing during hardware bring-up and firmware integration.
- Developed **Linux character device drivers** implementing **open()**, **read()**, **write()**, **ioctl()**, and **release()** file operations for user–kernel interaction.
- Implemented **user–kernel communication** using **copy_to_user()** and **copy_from_user()** APIs, and debugged **kernel modules** using printk, dmesg, and GDB.

Education

Bachelor of Engineering(B.E) - ECE 2021 – 2025 | Hydearabad

KG Reddy College Of Engineering & Technology

CGPA - 7.14

Skills

Programming Languages
C, Embedded C, Python (Basic)

Interfaces / Protocols
UART, I2C, SPI, GPIO, Interrupts

Linux Kernel / Driver
Character Device Driver, Kernel modules, GPIO, I2C subsystem basics, User-Kernel communication, Device Tree Basics, procfs, sysfs, ioctl

Debugging Tools
GDB, dmesg, printk, Logic Analyzer, Oscilloscope, DMM

Microcontroller / Processors
ARM Cortex-A, Cortex-M4

OS Fundamentals
Linux Internals, Process management, Memory management, IPC, Multithreading basics, RTOS(basics)

Tools & Environments
GCC, Makefiles, Keil µ Vision, STM32cubeIDE, Git, Ubuntu/Linux

Projects

Linux Character Driver Development

- Developed **Linux character device drivers** to enable controlled interaction between user space and kernel space through the **/dev** interface.
- Implemented core driver **file operations**, including **open()**, **read()**, **write()**, **unlocked_ioctl()**, and **release()** using the **struct file_operations** interface.
- Implemented **custom IOCTL commands** using **_IO macros** to support device-specific control operations.
- Implemented **user–kernel** data transfer using **copy_to_user()** and **copy_from_user()** APIs to ensure safe data exchange.
- Performed dynamic **major/minor** number allocation using **alloc_chrdev_region()** and registered character devices using **cdev_init()** and **cdev_add()**.
- Configured dynamic device **node creation** using **class_create()** and **device_create()**, eliminating the need for manual **mknod**.
- Added **kernel logging** using **printk()**, **pr_info()**, and **pr_err()** to trace driver execution, monitor device operations, and assist in debugging.
- Ensured proper **resource allocation and cleanup** during kernel module initialization and removal.
- Debugged and validated driver functionality using **dmesg**, **kernel logs**, and **user-space test applications**.

Bare-Metal Firmware Driver Development (UART, I2C, SPI)

- Developed **bare-metal communication drivers** for **UART, I2C, and SPI** on **ARM Cortex-M (STM32)** microcontrollers using register-level programming in **Embedded C**.
- Configured **UART communication**, including **baud rate generation**, **parity control**, **word length**, and hardware flow control (RTS/CTS).
- Configured **DMA-based UART TX/RX communication** to improve **data throughput** while reducing **CPU utilization**.
- Implemented **polling-, interrupt-, and DMA-driven communication** with **circular buffering** for efficient data handling.
- Implemented **error detection** and **handling** for **UART**, **I2C**, and **SPI** communication.
- Developed **I2C master communication**, including **START/STOP** generation, multi-byte data transfers, **ACK/NACK management**, and address phase handling.
- Developed **SPI master communication** with full-duplex data transfer, configurable **CPOL**, **CPHA**, **clock prescaler**, and **data frame format**.
- Interfaced **sensors, EEPROM, and displays**, and validated communication timing using **logic analyzers** and **oscilloscopes**.
- Debugged firmware using **GDB** and analyzed **peripheral registers** to identify and resolve low-level issues.
- Validated **interrupt handling**, **communication performance**, and firmware functionality for real-time embedded applications.