GOPI PUJITH VELPULA

Firmware Developer

+91 7569540248 | velpulagopipujith24@gmail.com

PROFESSIONAL SUMMARY

Firmware Development Engineer with 1+ year of hands-on experience in developing Linux Character Device Drivers and bare-metal firmware on ARM Cortex-M (STM32) platforms. Proficient in register-level programming, peripheral driver development (UART, I2C, SPI), board bring-up, and low-level hardware debugging using GDB, oscilloscopes, and logic analyzers. Experienced in GPIO configuration and interfacing peripherals such as LCDs, LEDs, ADCs, keypad matrices, and PWM, with exposure to both micro-controllers and ARM Cortex-A72 based microprocessor platform

TECHNICAL SKILLS

- **Programming Languages** : C, Embedded C, Python(Basic)
- **Linux Kernel/Driver** : character drivers ,kernel modules ,user-kernel communication ,PROCFS/SFS ,IOCTL
- **Microcontroller/processors** : ARM Cortex-A, Cortex-M4
- **Interfaces / Protocols** : UART, I2C, SPI, GPIO, Interrupts ,Timers
- **Tools & Environments** : GCC, Make files, Keil µ Vision, STMCUBE IDE, Git, Ubuntu/ Linux
- **OS Fundamentals** : Linux Internals, Process management, Memory management, IPC, Multithreading basics, Free RTOS(basics)
- **Debugging Tools** :GDB, DMESG, PRINTK, Logic Analyzer, Oscilloscope, DMM, protocol Analyzer

PROFESSIONAL EXPERIENCE

Designation: Embedded Firmware Developer

WV Automation Pvt. Ltd., Hyderabad *01/2025 – Present*

- Designed and implemented UART, I2C, and SPI communication drivers, ensuring protocol compliance, timing accuracy, and robust signal integrity validation using logic analyzers and oscilloscopes.
- Implemented SYSFS, PROCFS and IOCTL interfaces for structured user–kernel communication, enabling automated test frameworks, runtime configuration, and efficient driver debugging.
- Performed STM32 platform bring-up, including device tree configuration, pin multiplexing (PINCTRL), peripheral clock enablement, and validation of hardware–software integration.
- Debugged low-level hardware and driver issues using DMESG, oscilloscope, and logic analyzer, correlating signal-level behavior with kernel logs and driver execution flow.
- Developed and tested multiple Linux Character Device Drivers, implementing complete file operations (open, read, write, IOCTL) with proper device registration, CDEV management, and error handling.
- Demonstrated strong understanding of Linux internals, including process and thread management, memory management concepts, IPC mechanisms, scheduling basics, and system call flow.
- Developed and optimized bare-metal firmware using register-level programming for low-level hardware control and peripheral driver integration without relying on high-level HAL abstractions.
- Configured and debugged GPIO interfaces using direct register access and bit manipulation techniques to support custom hardware functionality.
- Optimized firmware execution by implementing interrupt-driven and DMA-based data transfers, reducing CPU load and improving real-time system performance and responsiveness.

PROJECT EXPERIENCE

Project 2: Linux character device driver with IOCTL interface for device configuration.

Project Description:

Developed a Linux kernel character device driver with an IOCTL interface to enable dynamic device configuration and control.

Implemented efficient user–kernel communication, supporting custom operations beyond standard read/write mechanisms.

Key Responsibilities:

- Designed and implemented a **Linux character device driver** supporting device interaction through Standard file operations such as `open`, `read`, and `write`.
- Developed and integrated an **IOCTL interface** to enable device-specific control operations from user space.
- Defined and handled **custom IOCTL commands** using `_IO`, `_IOR`, `_IOW`, and `_IOWR` macros for flexible device configuration.
- Implemented **user-kernel data transfer mechanisms** using `copy-to-user` and `copy-from-user` APIs.
- Designed IOCTL commands for **device mode control, status monitoring, and buffer management**.
- Managed **dynamic device configuration**, allowing runtime adjustments without reloading the kernel.
- Ensured proper **resource allocation, initialization, and clean up** in module `INIT` and `exit` routines.
- Implemented robust **error handling and validation** for safe interaction between user space & kernel space
- Debugged and verified driver functionality using **kernel logs(`PRINTK`)** and a custom user-space test Applications.
- Maintained modular, scalable, and readable code following Linux kernel programming best practices.

Project 1:Bare-Metal Firmware Driver Development for I²C and SPI - Industrial monitoring system.

SocArchitecture **: Arm Cortex-M**

Tools **: Keil u vision, Logic analyzers,DSO,Stm32-Cubeid**

TemperatureSensor:TMP102(Texas Instruments)

Project Description:

Developed bare-metal firmware drivers for I²C and SPI communication on an ARM Cortex-M architecture-based industrial monitoring system. The project involved interfacing a temperature sensor and enabling reliable real-time data transfer, focusing on low-level protocol implementation, debugging, and system validation.

Key Responsibilities:

- Developed and implemented **I²C and SPI drivers** from scratch in a bare-metal environment.
- Implemented I²C features including **start, repeated start, stop conditions, ACK/NACK handling**, and read/write operations.
- Enabled **interrupt-driven data transfer mechanisms** for efficient communication.
- Designed and executed **test cases** for I²C master/slave communication under varying load conditions.
- Debugged **protocol violations, transmission errors, and clock glitches** using logic analyzers and digital storage oscilloscopes (DSO).
- Resolved **SPI communication issues**, including slave select (SS) handling and synchronization problems.
- Validated SPI controller functionality across **multiple clock modes (CPOL/CPHA variations)**.
- Integrated temperature sensor data flow from I²C to SPI-based display for **real-time monitoring**.
- Performed **end-to-end functional validation** to ensure reliable data acquisition and display.
- Utilized tools such as **Keil u Vision, STM32CubeIDE, logic analyzers, and oscilloscopes** for development &debug

EDUCATION

Bachelor of Technology (B. Tech), Electronics and Communication Engineering (ECE)
Lakki Reddy Bali Reddy college of engineering (JNTUK) CGPA: **7.31** 2021 – 2025

Intermediate,(MPC) Sreenidhi junior college ,CGPA: **8.68** 2019-2021

Matriculation,(X) ST .Francis E.M High School ,CGPA: **9.5** 2019