Ajay Garg 98 18 78 25 93 ajaygargnsit@gmail.com

Summary

Anti-Startups / Linux-Kernel / Bare-Metal / Embedded / Device-Side IOT.

Looking for a hands-on systems-software development-role, in a product-based MNC-team, that can provide me long-term career-stability.

Please note that I cannot physically relocate outside Delhi/NCR, till my mother is alive.

Also, kindly note that I am **NOT** interested in the following :

- Hands-on Web / GUI development.
- Hands-on Verilog / VHDL development.
- Hands-on FPGA bitstream-generation

Education

- B.E., Electronics and Communication Engineering, 82.1% (*NSIT, 2004 - 2008*).

Assertions / Strengths :

- Always striving to write portable bare-metal code, when working on the device-side.
- Resilience is a requirement in systems-engineering, and not merely an option.
- Duplicate-code is evil.

Favorite Computer Science topics

- C, Multithreading, Operating Systems, Computer Architecture.

Employment History

Linux Kernel Engineer
(September, 2021 - Present) : Home

Contribution in mainline :
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/log/?qt=grep&q=ajaygargnsit@gmail.com

- [pci, usb] :
  - Device-Drivers for communication over the above buses (details on next page).
- [qemu / kvm, iommu]
  - (***demoable***) Practical understanding of conversions between host/guest virtual/physical addresses, in virtualized/non-virtualized environments.

Systems / Embedded Engineer / Architect
(June, 2008 - June, 2019, one year-gap in-between) : Startups

Multiple Projects, requiring communication / connectivity over following buses / mediums :

- CAN
  - On Linux : via (user-space) SocketCAN.
  - On Bare-Metal Boards : via HALs.

- BLE
  - On nrf52 bare-metal board, via HALs (nrf-SDK).

- SPI / I2C
  - On Linux : via (user-space) /dev-filesystem exposed device “files”.
  - On Bare-Metal Boards : via HALs.

- PCI
  - Communication with BAR-0 config-space of an endpoint (SD-MMC) host-controller, via :
    - (***demoable***) ***kernel-space*** driver, using MMIO.
    - (***demoable***) user-space driver, using mmap()ing of sysfs-exposed resource “file”.

- USB :
  - Sending/Receiving payload raw-bytes between peers, via :
    - (***demoable***) ***kernel-space*** driver on the host-side.
    - (***demoable***) ***kernel-space*** driver on the gadget-side (emulated via configfs).

Multiple Projects, requiring socket-programming :

- Device-Side, Server-Side.
- Protocols :
  - L7 : MQTT, Modbus-TCP, SMPP, Diameter.
  - L6 : WolfSSL, OpenSSL.
  - L3/L4 :
    - Posix on Linux.
    - GPRS / UMTS / LTE on Telit and Simcom modems.
    - LWIP+FreeRTOS for Ethernet on bare-metal.

Instamsg-c :

In one of the product-startups, it was required to connect to company's proprietary MQTT-broker server, in a device/gateway agnostic manner.

So, I bootstrapped the umbrella-framework instamsg-c, which contained about 90% bare-metal C code. The remaining 10% device-specific code was then ported to 14 kinds of gateways. This 10% code majorly contained following APIs (which needed to be ported according to gateway) :

- Connectivity :
  - Wifi (On Linux : via Posix-APIs).
  - Ethernet (On Linux : via Posix-APIs; On bare-metal board via LWIP+FreeRTOS).
  - Cellular - GPRS / UMTS / LTE (On Telit and Simcom modems).

- Persistent-Storage :
  - On Linux : via Posix-APIs.
  - On Bare-Metal Boards (mostly requiring interfacing with external-flash over SPI).

- Time and Watchdog
  - On Linux : viaPthread-APIs.
  - On Bare-Metal Boards (via systick-interrupt and HAL-libraries).

Additionally, Modbus-RTU and Modbus-TCP code was present as part of 90% bare-metal code, while the serial-interface APIs (UART, USART, USB) were part of 10% device-section.