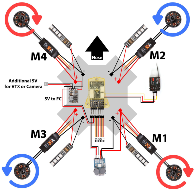
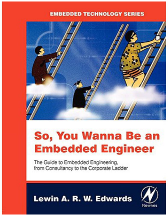


Now what?

Kizito NKURIKIYEYEZU, Ph.D.



[1] Edwards, L. (2014). So You Wanna Be an Embedded Engineer: The Guide to Embedded Engineering, from Consultancy to the Corporate Ladder. Newnes.

Electronics

- DC characteristics of diodes, bipolar transistors, FETs, op-amps, and comparators.
- The ability to read a schematic
- Read and understand datasheets for microprocessors
- Understand basic understanding of how PCB routing can affect signal propagation.
- I/O configurations: open-source, open-drain, full totem-pole, protection diodes
- ESD susceptibility; placement of spark gaps, series resistors, bypassing

Low-level programming

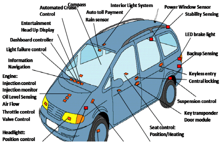
- Computer architecture: arithmetic logic unity, memory system,flip-flops gates, registers, RAM chips, clock based sequential logic, instruction set architecture
- At least understand one assembly language
- Master C (the C build process), volatile, pointers, logical operators and shifting functions and storage classes array, structure, union, data structures, call by reference, function, macros, reading and writing registers, interrupts/polling mode, DMA transfers, code optimization, loop optimization techniques, linked lists, queues, FIFO, hash tables, static and dynamic memory allocations
- Unit testing and mocking frameworks for embedded systems (e.g., CppUTest)
- Understand which C++ are low overhead and which should be avoided in embedded system

Small Embedded Systems)

- common microcontroller peripherals like DMA, timers, ADC, DAC, watchdog, USB, memory, PMW, Memory Protection Unit
- common protocols, like I2C, SPI, UART, USB, DMA, I2S, CAN
- Understand the pro and cons of the common communication protocols and have the ability to write simple device drivers for them

Large embedded systems

- Large software development: verison control, terminal, data structure and algorithms, github, SSH, design patterns, null pointers, static, externs, assertions, memory verification, stl, make
- control theory, PID, digital signal processing, system application specific circuits,
- ensures the availability of system memory, check if the processor's speed



theory, the need to need to limit power lost when running the system continuously

- serial vs parallel I/O, Real Time Clock (RTC), Watchdog Timer

Embedded Linux

- Understanding of advanced topics in operating systems: Synchronization Hardware, deadlocks, logic vs physical memory address spaces, swapping, paging and virtual memory, thrashing, file systems, secondary file system,
- Linux kernel compilation, optimization & booting sequence
- tools and processes that enable the creation of Linux distributions for embedded and IoT software: Buildroot and Yocto, docker, OpenWRT/LEDE
- Software optimizations skills at the System on a Chip (SoC) level, boot