Intro Embedded Operating Systems (Part 2)

1. Design Patterns
   * Modularity
     + Pro
       - Facilitates re-usable code
       - Allows for experimenting with new code
       - Flexibility for application programmers
     + Con
       - Can be complex to know what code is actually being used
       - Module interfaces can suppress useful hardware features
   * Virtualized and Non-Virtualized Resources
     + Pro
       - Direct hardware control reduces uncertainty
       - Flexibility for programmers
     + Con
       - Adds complexity: which to use?
       - Choosing wrong can lead to strange bugs
       - Same or different interfaces?
   * Long Running Operations
     + Pro
       - Some operations are computationally complex
       - Useful to help programmers manage these
     + Con
       - How long is “long”?
       - Overall system timing
   * Event-driven Versus Threading
     + Events
       - Better models hardware
     + Threads
       - Easier to reason about for programmers
     + Difficult to blend them
     + Many ramifications for interface design
2. Debugging
   * Challenges
     + No display
     + Timing overhead
     + Low-level development
     + Toolchain complexity
   * Simple Approaches
     + printf()
     + Turning on/off LED
   * Toolchain Help
     + GDB & JTAG
   * Hardware Approaches
     + Aveksha, Uses JTAG port: <https://engineering.purdue.edu/dcsl/publications/papers/2011/aveksha-sensys2011.pdf>
   * OS Help
     + Crash logs of MCU state
     + Offline analysis tools
     + Debugging state on demand (e.g. button press)
   * Interactive shell
     + Provide inspection capabilities while a device is running
   * Profiling
     + Energy use estimations based on activity
3. Services and Shared Libraries
   * Code update
     + Small bootloader manages booting into the correct image
     + “Golden master” for backups
   * Wireless MAC layers
     + Provide low power send and receive
     + Major challenge is to ensure compatibility between devices
   * Wireless routing
     + Multi-hop
     + Star
     + Flooding
   * Filesystem
     + Nonvolatile storage
     + Variety of abstractions
       - Files
       - DB
       - Append only log
       - Key-value
   * Time synchronization
4. Leveraging Available Hardware
   * DMA
     + Reduce CPU time to transfer buffers between components
   * MPU
     + Provide hardware protection for certain memory regions
   * Watchdog
     + Reset MCU if chip in a bad state