ECE 471 – Embedded Systems
Lecture 12

Vince Weaver

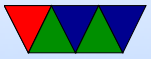
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

28 September 2020

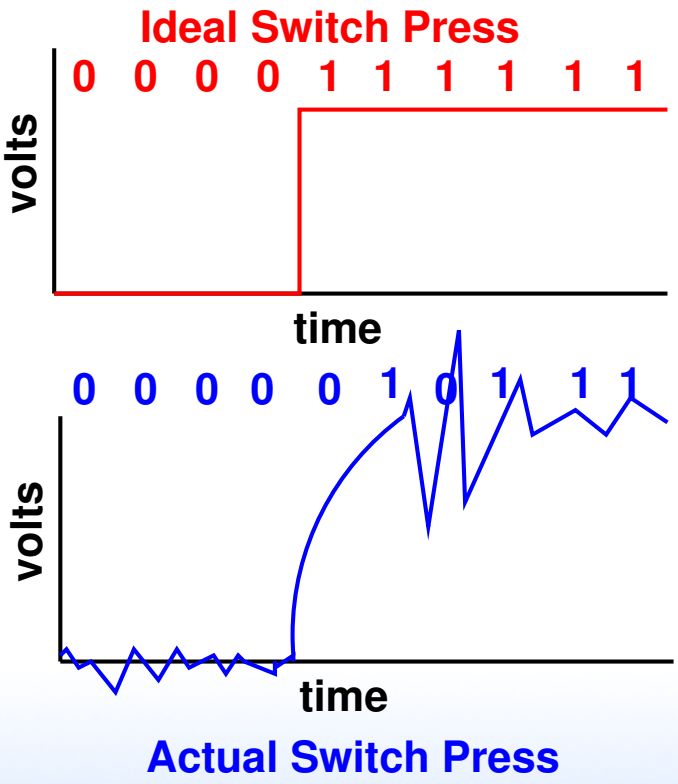
Announcements

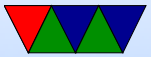
- How is HW#4 going?



Debouncing (from last time)

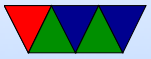
- Noisy switches, have to debounce





Debouncing!

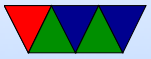
- Can you fix in hardware? Capacitors?
- Can you fix in software? No built-in debounce like on STM32L
- Algorithms
  - Wait until you get X consecutive values before changing
  - Get new value, wait short time and check again



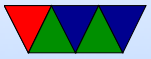
Coding Directly for the Hardware

One way of developing embedded systems is coding to the raw hardware, as you did with the STM Discovery Boards in ECE271.

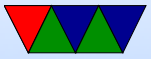
- Compile code
- Prepare for upload (hexbin?)
- Upload into FLASH
- Boots to offset



- Setup, flat memory (usually), stack at top, code near bottom, IRQ vectors
- Handle Interrupts
- Must do I/O directly (no drivers)
  Although if lucky, can find existing code.
- **Code is specific to the hardware you are on**

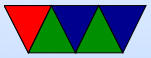


Instead, one can use an Operating System

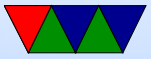


Why Use an Operating System?

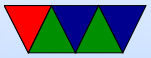
- Provides Layers of Abstraction
  - Abstract hardware: hide hardware differences. same hardware interface for classes of hardware (things like video cameras, disks, keyboards, etc) despite differing implementation details
  - Abstract software: with VM get linear address space, same system calls on all systems
- Other benefits:
  - Multi-tasking / Multi-user



  - Security, permissions (Linus dial out onto /dev/hda)
  - Common code in kernel and libraries, no need to re-invent
  - Handle complex low-level tasks (interrupts, DMA, task-switching)
- Abstraction has a cost
  - Higher overhead (speed)
  - Higher overhead (memory)
  - Unknown timing
- What about other things?
  - Easy to code for? Provide examples

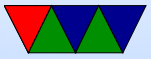


- Nice GUI interface? Sometimes



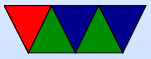
What's included with an OS

- kernel / drivers (syscall barrier) – Linux definition
- also system libraries – Solaris definition
- low-level utils / software / GUI – Windows definition
  Web Browser included?
- Linux usually makes distinction between the OS Kernel and distribution. OSX/Windows usually doesn't.



Bypassing Linux to hit hardware directly

- Linux does not support things like pullups, but people have written code that will poke the relevant bits directly.

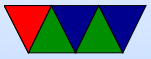


Bypassing Linux for speed

http://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/

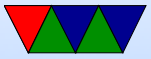
Trying to generate fastest GPIO square wave.

shell	gpio util	40Hz
shell	sysfs	2.8kHz
Python	WiringPi	28kHz
Python	RPi.GPIO	70kHz
C	sysfs (vmw)	400kHz
C	WiringPi	4.6MHz
C	libbcm2835	5.4MHz
C	Rpi Foundation “Native”	22MHz



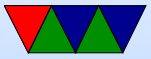
Operating Systems Types

- Monolithic kernel – everything in one big address space. Something goes wrong, lose it all. Faster
- Microkernel – separate parts that communicate by message passing. can restart independently. Slower.
- Microkernels were supposed to take over the world. Didn't happen. (GNU Hurd?)
- Famous Torvalds (Linux) vs Tannenbaum (Minix) flamewar



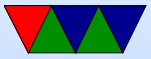
Common Desktop/Server Operating Systems

- Windows
- OSX
- Linux
- FreeBSD / NetBSD / OpenBSD
- UNIX (Irix/Solaris/AIX/etc.)
- BeOS/Haiku



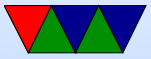
Embedded Operating Systems

- Microsoft WinCE, Windows Mobile
- Linux / Android
- VXworks – realtime OS, used on many space probes
- Apple iOS
- QNX – realtime microkernel UNIX-like OS, owned by Blackberry now
- Cisco iOS
- ThreadX – found in Pi GPU



Embedded Linux Distributions

- linaro – consortium that work on ARM software
- openwrt – small distro initially designed for wireless routers
- yocto – Linux Foundation sponsored embedded distro
- maemo – embedded distro originally by Nokia (obsolete)
- MeeGo – continuation of maemo, also obsolete



- Tizen – Follow up on MeeGo, by Samsung and Intel
- Ängstrom – Merger of various projects
- And many others. It's very easy to put together a Linux distribution