ECE 471 – Embedded Systems
Lecture 31

Vince Weaver

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

vincent.weaver@maine.edu

2 December 2019

Announcements

- HW#10 was posted, an energy assignment
- Posting of project presentation order
  Lots of people want to go Friday
- Final is on Monday, December 16th at 9:30AM in Barrows 123
- Remember to do course evaluations

HW#9 – Code

- C code review
- Do note, it's an LED display not LCD
- Error checking. Be sure not to segfault if dev not there!
- Following a spec?
  - Corner cases
  - should shutdown_display() clear the display? Maybe, makes it harder to grade.
  - Single-digit temps, be sure to remove leading zeros
  - is Zero negative?

- Rounding
- Do you need a . after a three digit temp?
- Left/right justified for single digit
- Reporting error! Must be sure display not printing invalid info! (door on walk-in oven. If it goes from 70F to 1000F (off scale) between readings, don't want it to stay at 70F, you want ERR or HOT or some way to notify something is wrong) More realistically, probe wire broke, should it just report last reading? Or maybe go blank?

HW#9 – Questions

- List an *example* of poorly written embedded code.
- Why write good code?
  Cut-and-pasting, good practice, among other reasons.
- Why is touch useful? force make to rebuild
  For some reason a number of assignments somehow didn't have updated time in the tar file.
- 2038 problem
  Time in Linux is seconds since 1-1-1970. Not a problem 64-bit machines, but overflows in 2038 for 32-bit. Can

avoid with a 64-bit system or else a specially patched Linux system
* discuss y2k problem ** worst problem year 19100 on websites

- ctime – last status (metadata) change (originally create time) things like permissions change, ownership change, rename
  mtime – last modified
  atime – last access
- In stat syscall. `stat` command. Why atime bad?

noatime, relatime

- utime() used by touch. Cannot change ctime, set to current time
- why not believe timestamp? maybe could look at ctime. also set clock back if own machine.
  HW assignment at Cornell

Ethernet

- Old, complicated standard, whole way up to 100GBps
- Modern form is often RJ-45, twisted pairs
- Power over ethernet (only Pi3B+ support)
- PiB models have 10/100 Mbps, Pi3B+ has “gigabit”
- Connected to on-board USB hub

HDMI

- High-Definition Multimedia Interface (2003)
- Compatible with DVI (if no copy protection used)
- Video, audio (up to 8 channels), CEC (consumer electronics control), ethernet
- No support for captions
- DDC – i2c bus, used for EDID (getting device info) and HDCP (copy protection)

- TDMS – transition minimized differential signaling
  Video, then during scan line breaks, audio, etc

- CEC – control up to 15 devices with one remote control
  (one wire serial bus)

- Various versions, various fees

Other video ports

- NTSC/composite – Pi has (hidden on later models)
- VGA (analog) – hard to get on Pi
- DSI connector – for touchscreen
- DVI
- Thunderbolt
- Displayport

- USB?

Wireless / Wifi

- Wireless ethernet
- 2.4GHz or 5GHz

CANbus

- Automotive. Introduced by BOSCH, 1983
- One of OBD-II protocols
- differential, 2 wires, 1MBps important things like engine control
- single wire, slower cheaper, hvac, radio, airbags

CANbus Protocol

- id, length code, up to 8 bytes of data id (usually 11 or 29 bits) type and who is sending it. Also priority (lower is higher) length is 4 bits. some always send 8 and pad with zeros
- Type is inferred from id. Can be things like engine RPM, etc
- DBC database has the ids and values. ASCII text database, hard to get legally.

- Dominant/Recessive. Message with lowest ID wins arbitration.
- CAN-FD – extended version with larger sizes

CANbus Linux

- Can4linux – `open("/dev/can0"); read(); write();` External project?
- SocketCAN – contributed by Volkswagen. In kernel. Uses socket interface. /Documentation/networking/can.txt

CANbus on Pi

- Not by default
- Can get SPI to CANbus adapters

ISA Bus

- Introduced with IBM-PC in 1981
- 8-bit (4.77MHz) then 16-bit (8MHz)
- +/-5V, +/-12V, 8 data, 20 address, DMA, IRQ
- Replaced by VLB (more pins, extra header), EISA (double pins in same connector), MCA micro-channel (different proprietary from IBM)

- Not enumerable at first, set jumpers. Later “Plug-n-Play”

LPC Bus

- Low-pin-count bus
- Intel, 1998, try to get rid of ISA
- Things like PS/2, Serial ports, floppy, etc.
  Still used for TPM Trusted Computing nonsense
- Replace 16-bit 8.33MHz parallel bus with 4-bit wide 33.3MHz bus. Only 7 wires. Easier to route than 72

“Conventional” PCI Bus

- Peripheral Component Interconnect
- Enumerable
- 1993, intel
- 62-pins, parallel, 133MB/s
- Extended with 32 or 64-bit versions, 33 or 66MHz, 3.3 or 5V. All slight differences in connectors to support all that.

- AGP (Accelerated Graphics Card) for graphics cards. 1997. Direct connect to CPU (not shared), multiple channels, faster clock

- PCI-X 1998, extension to 133MHz. Not to be confused with PCI-Express (PCIe)

PCI protocol

- 256B Config space, mapped into CPU address. Small area system can probe, used to setup larger mappings
- Can have on-board ROM that can be executed. Problem when using on non-x86 systems (emulators needed? special [expensive] PowerPC versions?)
- Latency timers keep bus-master from hogging bus
- 4 interrupt lines, can be shared. Level rather than edge-triggered interrupts make sharing easier

•

PCIe

- PCI-express, 2003
- Serial, replaced point-to-point with lanes, packet-based x1, x2, x4, x8 x16, x32
- Compatible with PCI at software level
- Differential Signaling
- External – Thunderbolt

- Serial better due to timing skew
- New x86 audrino quark has PCIe

PCIexpress Mini

- PCIe x1, USB, SMBus, etc
- Smaller card

PCMCIA Bus

- Personal Computer Memory Control International Association
- 16-bit
- Cardbus, 32-bit
- Mostly replaced these days

PC/104 Bus

- Stackable small x86 boards usually
- Run ISA or PCI signals up vertically

VME Bus

- m68k bus but generic enough
- Still found in some embedded systems

Other

- SATA, eSATA, PATA, SCSI (disk drives)
- Firewire
- RapidIO
- Quickpath QPI
- Hypertransport
- Thunderbolt (requested)

- List of competing busses at end of USB wiki article

PWM Linux

- Available on GPIO18 (pin 12) (also GPIO19 on newer)
- Assuming 4.9 kernel or later (sets up proper clock) just edit config.txt and add `dtoverlay=pwm`
- reboot. lsmod should show `pwm_bcm2835` loaded
- `https://www.kernel.org/doc/Documentation/pwm.txt`
- sysfs interface
  - `cat /sys/class/pwm/pwmchip0/npwm`
  - `echo "0" > /sys/class/pwm/pwmchip0/export`
  - `echo "1000" > /sys/class/pwm/pwmchip0/pwm0/period`

  - `echo "500" > /sys/class/pwm/pwmchip0/pwm0/duty_cycle`
  - `echo "1" > /sys/class/pwm/pwmchip0/pwm0/enable`
- Feel free to use wiring-pi or similar

USB

- Universal Serial Bus
- Designed to replace all of the various cables on a PC with one type (keyboard, mouse, printer, serial, SCSI, joystick)
- How successful was that?
- Way more complex than most previous interfaces `http://www.usbmadesimple.co.uk/index.html`

USB 1.0

- 1996
- Low Speed
  - 1.5Mbit/s (keyboard, mouse, etc)
  - Thinner, flexible cable
- Full Speed
  - 12Mbit/s (disk, USB key)
- USB 1.1 – ?

USB Physical Layer

- 2-5m cables
- 4 pins. 5V, GND, D+, D-. Differential signaling (subtract). More resistant to noise.
- Micro connectors have extra pin for on-the-go (says if A end or B end gnd vs v+)
- Unit load, 100mA. Can negotiate up to 500mA (2.5W) (more USB 3.0)
- Up to 127 devices (by using hubs). Up to 6 levels of hubs. Powered vs not.

- Enumeration, vendor and device
- Connectors. A/B. Designed so only one end goes to host. Micro, mini.

USB 2.0

- 2000
- High Speed 480Mbit/s

USB More Recent

- USB 3.0 – 2008 – SuperSpeed 5GBit/s (though hard to hit that) Full Duplex (earlier half duplex)
- USB 3.1 – 2014 – SuperSpeed+ 10Gbit/s
- Backwards compatible, has 5 extra pins next to standard micro with GND, SSTX+/- and SSRX+/- (full duplex)
- USB-C – 2014

  24-pin: 4 power/ground pairs, two differential non-super-speed pairs, four pairs of high-speed data bus, two sideband pins, two pins for cable orientation

cables can be USB2, USB3, USB3.1, up to 5A(20V=100W) but 3A more common

wrong pullup can cause cable that damages hardware

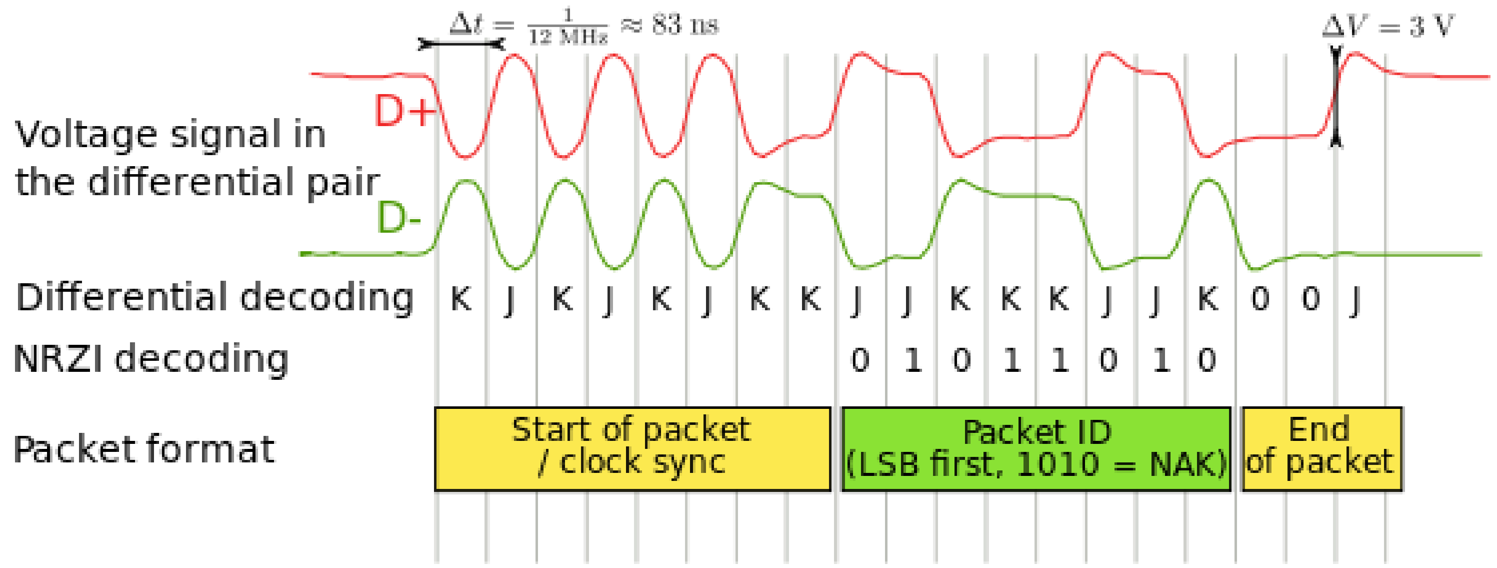
USB 1.1 and 2.0 Signaling

- Differential signaling, twisted pair, 90Ohm impedance
- Low+Full = 0V low, 3.3V high, not terminated
- High = 0V low, 400mV high, terminated with resistor
- Device Detection
  - Host, 15k pulldown pulls data lines to 0 (nothing connected, SE0)
  - USB device pulls line high with 1.5k which overpowers pulldown. Full bandwidth D+ high, low bandwidth D- high

- Some chargers use special resistors across D lines to indicate power they can draw.

- J and K states.
- NRZI line coding – 0 signaled by J to K (switching state). 1 signaled by leaving as is
- Bit stuffing – after six consecutive 1s must include 0
- starts with 8 bit synch – 00000001 which is KJKJKJKK. Data then sent. End marked by 00J.
- Reset by 10ms SE0
- Highspeed uses "chirping" to negotiate speeds, during reset chirps J and K

- Example from Wikipedia CC0:


$\Delta t = \frac{1}{12\text{ MHz}} \approx 83\text{ ns}$
$\Delta V = 3\text{ V}$
Voltage signal in the differential pair
D+
D-
Differential decoding
K J K J K J K K J J K K K J J K 0 0 J
NRZI decoding
0 1 0 1 1 0 1 0
Packet format
Start of packet / clock sync
Packet ID (LSB first, 1010 = NAK)
End of packet

USB 3.0

- SuperSpeed, separate lines, but original lines used to config
- SuperSpeed uses 8b/10b encoding (limits bandwidth), CRC, other features
- SuperSpeed+ uses 128b/132b encoding

Latency

- For Low and Full shortest transaction time is 1ms. Can this be a problem?
  - Low latency gaming keyboards `https://danluu.com/keyboard-latency/`
  - Keyboard latency (scan keys, report keycode)
  - USB latency to system
  - Interrupt latency
  - OS update screen
  - LCD monitors often buffer a few frames

- Old Apple II (1MHz 8-bit) could go from keypress to screen faster than modern keyboards even finish scanning

USB Protocol

- Various packets sent
  Huge list of them
- Checked by CRC
- Low power suspend mode, no more than 2.5mA
- Signal sent to all devices on USB bus, all but addressed one ignores

USB Design

- Each device has endpoint
- isochronous – guaranteed data rate but with some potential data loss (video)
- interrupt – low-latency, like keyboards
- bulk – disk access

USB Linux

- Linux drivers
  - Device classes – HID, audio, etc. One common driver can handle all devices of a class
  - Specific – device driver is board specific and must have a list of all vendor/device IDs that are supported

- libusb
  Allow direct userspace access to USB interface
  Used by low-level things that might not need driver

old cameras (not standardized), custom hardware

USB on Rasp-pi

- USB-OTG – on the go. Allows device to act like a host (so can hook up devices as per normal) or as normal USB device. Decides which based on whether A or B cable plugged in, check ID pin (micro/mini have 5th pin)
  The Pi-B does not support running in gadget mode externally (a hub in the way) and the OTG hardware requires more software support than (it is simpler) than regular USB.

- USB 2.0 (sorta). Cannot supply full power (why? Only 1A power supply typical). Also cannot handle high-bandwidth things like audio cards and USB-cameras well.
- USB-host – standard USB port. Cannot provide high current, so use a powered hub if using anything more than keyboard or mouse