

# **ECE 471 – Embedded Systems**

## **Lecture 33**

Vince Weaver

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

`vincent.weaver@maine.edu`

30 November 2022

# Announcements

- HW#10 was assigned
- Presentations are starting
- Feel free to return borrowed hardware.
- Tentative presentation schedule was sent out



# Final Preview

- Monday, December 12th, 12:15pm - 2:15pm
- Is cumulative for whole class, but concentrates on material from latter half of class. \*No assembly language\*
- Know the definitions of an embedded system and be able to say if a certain machine meets them.
- Know hard/soft/firm realtime



- Know the benefits/downsides of an operating system
- Security/Code Quality – mostly be aware of what things can go wrong if you are not careful when coding
- Embedded busses – know the relative tradeoffs between i2c, spi, and 1-wire. Mostly speed, distance, number of devices
- Power/Performance like HW10
- Give you some C code from one of the homeworks, comment it



# HW#9 – Review



# 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!
- Heisenbug if not init buffer[0]
- How do you convert from float to decimal?
  - 45.9 print as 45.8?

Floating point math is a pain! What do you get if you do `int result=10*(45.9-45);`? 9.000000? Print more digits 8.999999999999999857891452847979963 for

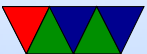


fp to int conversion just drops the floating point part,  
doesn't round

- 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
  - sig-figs for -1.0
  - right justified?
  - is Zero negative?
  - Rounding



- Do you need a . after a three digit temp?
- Left/right justified for single digit
- Oddly placed minus sign
- 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

- Test inputs: try to have one in each case. Also might be nice to check each digit 0..9 to make sure those are all printing well.
- 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 on 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

Recent microsoft problem, virus signatures, YYMMDD0001, when year 2022 doesn't fit in 32-bit number. Oops.

- 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



# Gave Sample Presentation of Chiptune Player



# Other busses

- Beyond this we didn't cover in class this year



# 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. g



- 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



# 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 3

- 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)
- Connector often blue





# USB C

- 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 4

- 2019
- Requires USB-C cable
- Can carry displayport, PCIe
- Support in Linux 5.6



# USB 1.1 and 2.0 Signaling

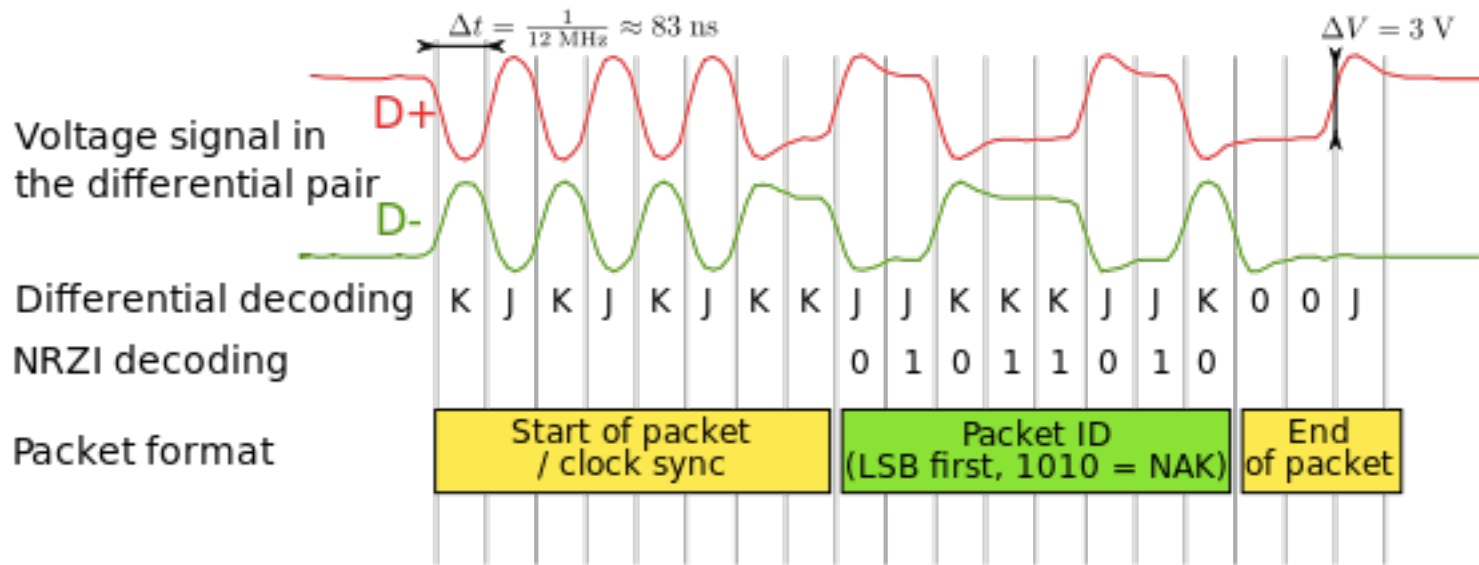
- Differential signaling, twisted pair, 90Ohm impedance
- Low+Full = 0V low, 3.3V high, not terminated
- High = 0V low, 400mV high, terminated with resistor
- Device Detection
  - Host, 15k pulldown pulls data lines to 0 (nothing connected, SE0)
  - USB device pulls line high with 1.5k which overpowers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



# USB 3.0 Signalling

- 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

- Pi4 has USB 3.0
- 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

