ECE 471 – Embedded Systems
Lecture 22

Vince Weaver

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

vincent.weaver@maine.edu

30 October 2026

Announcements

- HW7 is due today
- HW8 (1-wire) will be posted
- Project topics due next Friday (6th)
- Please give me advance warning if want to borrow project parts so I can make sure I actually have it.

One-Wire Bus

- From Dallas Semiconductor (bought by Maxim in 2001)
- One data wire plus ground (how do you get power?)
  - Open collector, data line pulled high
  - Devices have capacitor to provide power when data line low "parasite power"
  - Low speed data and power over one wire (you also need ground)

One-Wire Bus Info

- One controller
- Can have many devices
- 16.3kbit/s
- Up to 300m twisted pair (phone or Ethernet wire) [why is twisted better? avoid antenna...]

One-Wire Bus users

- Temp probes
- (old) Apple magsafe connector
- eeproms
- Single bit of GPIO
- Java rings?

One-Wire Protocol

- Each device has unique 64-bit ID:
  - 8-bits of type
  - 48 bit ID
  - 8-bit CRC
- Typically 8-bit command followed by 8-bit data chunks

One-Wire Protocol – Read/Write

Open Collector (BJT equivalent of MOSFET Open Drain)

- Write
  - Write 1 – Controller pull bus low for 1-15us
  - Write 0 – Controller pull bus low for 60-120us
- Read
  - Controller pull bus low for 15us (checks after another 15us).
  - Device does nothing if it's a 1
  - If it's a 0 it pulls the bus low for another 45us

One-Wire Protocol – Reset

- Reset/Presence
  - Controller pulls bus low for 480us.
  - If a device is present it pulls bus low for 60us starting within 60us after the reset pulse.

Enumerating BUS (ROM commands)

- How can you probe all 2^{64} possible addresses? `https://www.maximintegrated.com/en/app-notes/index.mvp/id/187`
- send a READ ROM request, returns 64-bit address. If multiple devices, then AND of all of them. (How do you detect this? Invalid CRC).
- SKIP ROM request sends command to all devices
- MATCH ROM request sends 64-bit address and only matched device responds

Enumerating BUS – continued

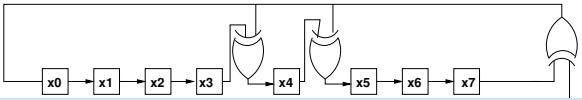
- SEARCH ROM –
  - Read first address bit from all devices on bus. Devices send their bit, followed by complement.

1	1	nothing there
0	1	all devices have 0 there
1	0	all devices have 1 there
0	0	conflict, you will have to probe both ways

  - Then it does a binary search to enumerate all devices on bus.
  - Faster than probing all 2^{64} possible.

Cyclic Redundancy Check (CRC)

- Can detect wrong (flipped) bits in a transmission
- Fast to do in hardware, just a shift register and xor gates (unlike checksum which needs adders)
- 1-wire CRC Can detect all double-bit errors, any overall odd number of errors, any cluster within an 8-bit window
- if CRCs with itself gets 0 at the end, how hardware detects correct address.
- $X^8 + X^5 + X^4 + X^1$
- Fill with zero, shift values in.



Hardware Interface

Possible ways to implement this:

- Use a GPIO and a pull-up resistor
- Use a serial UART. Needs extra circuitry to hook both TXD and RXD to bus
- USB/i2c/network connected
- Dedicated hardware?

Linux Interface

- "w1" driver merged in 3.6 kernel (a while ago now)
  - Driver for various interfaces, including bit-banging over GPIO (w1-gpio)
  - `/sys/bus/w1/devices/22-0000001d84f2/w1_slave`
  - read value and get ASCII dump of transaction
- OWFS – another driver, not in main kernel. Lets you export one-wire devices as a filesystem

One-Wire on Raspberry Pi

- The recommended way to enable things is to use `raspi-config`, interface options, 1-wire, enable on boot (you might need to reboot.

- Otherwise you can manually `sudo modprobe w1-gpio` and `sudo modprobe w1-therm` but on device tree systems (meaning, most recent Raspbian distributions) this might not be enough to make sure GPIO4 gets set up properly.

Temperature Interface w1_therm

- `cd /sys/bus/w1/devices/`
- `ls`
- `cd 28-000005aaf7ed` The serial number will differ (each unique)
- `cat w1-slave`

```
82 01 4b 46 7f ff 0e 10 70 : crc=70 YES
82 01 4b 46 7f ff 0e 10 70 t=24125
```

- Valid if the last value in first line is YES (passes CRC)
- second line has temperature in mili-degrees Celsius

DS18B20

- -55 to 125C
- +/- 0.5C from -10 to 85C
- 9 to 12 bit resolution (configurable)
  Takes longer to convert more bits
- Converts temp in 93ms - 750ms
- Can set alarm (if go over a temp, a high bit set in result)
- small EEPROM can store alarm, config (number of bits) etc

DS18B20 – Low Level Transaction

- Controller resets
- Controller listens for device to see it is present
- Controller sends MATCH ROM (0x55) then sends the 64-bit ID of the device it wants to talk to
- Controller sends a CONVERT T (0x44)
- Controller holds line high during conversion so the device has enough power to do the calculations

DS18B20 – Low Level Transaction (more)

- Controller sends reset
- Controller listens for response
- Controller sends MATCH ROM (0x55) then the 64-bit ID
- Controller sends READ SCRATCHPAD (0xbe)
- Controller reads 8 bytes from device and CRC. If CRC wrong, tries again.

DS18B20 – Decoding the Data

- 9 bytes from device:
  `82 01 4b 46 7f ff 0e 10 70 : crc=70 YES`

  Byte 0/Byte1 = LSB/MSB Temperature = 0x0182
  Byte 2 = TH register (high temp alarm, 8-bit)
  Byte 3 = TL register (low temp alarm, 8-bit)
  Byte 4 = config register 7f = 12 bit
  Byte 5,6,7 reserved (5=0xff, 7=0x10)
  Byte 8 = CRC

DS18B20 – Decoding the Data (more)

- Temperature is signed fixed point...

```
0x0182 = SSSS S654 3210 -1-2-3-4
         0000 0001 1000  0 0 1 0
```

$$2^4 + 2^3 + 2^{-3} = 16 + 8 + \frac{1}{8} = 24.125^oC = 75F$$

Less Convoluted Way of Getting the Temperatu

- This seems like a lot of work just to get the temperature!
- Someone realized this and relatively recently added a "temperature" file to get the value w/o all the work
- Why not use "temperature" file?
- Had to look in linux source code, w1_therm.c
- git log / git bisect
- Added May 11 2020, first appeared Linux 5.8
- Not even the main feature, as an afterthought
- Better fits the sysfs philosophy (one value per file)

Should we Use it?

- Any reason not to use it? What if kernel version too old...
- For now lets still parse things the old way as you might run on a system that's too old to have this new version

C string review

String manipulation is famously horrible in C. There are many ways to get the "YES" and "t=24125" values out of the text file. Any you choose is fine.

This is just a brief review, I will talk about is a bit more Monday

Brief C string review

- If you trust the Linux kernel developers to keep a "stable ABI" then you can just read in the entire line into a string (array of chars) with `fgets()` and index into the string array to look for 'Y'
- Alternately you can use `fscanf()` to read the file. Again you have to trust the format won't change and that the YES always happens the same number of values into the file. One helpful hint, putting a '*' in a conversion (like `%*s` tells scanf to read in the value but ignore it.

C string review 2

- `sscanf()` is like `fscanf()` but reads from a string
- Converting string to decimal or floating point
  `atoi()`, `atof()`, `strtod()`
- Comparing strings. Can you just use ==? NO!
  Be careful using `strcmp()` (or even better,
  `strncmp()`, has unusual return value
  less than, 0 or greater than depending. 0 means match
- If you had a string that was "V=YES" how could you start a string compare starting with the 2nd byte? Pointer math? Use `string+2` or even `&string[2]`.

HW#8 – C string review

- String manipulation is famously horrible in C.
- There are many ways to get the "YES" and "t=24125" values out of the text file for HW#8.
- Any way you choose is fine.