ECE 471 – Embedded Systems
Lecture 10

Vince Weaver

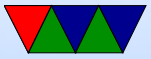
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

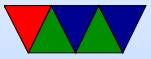
29 September 2016

Announcements

- GPIO permissions: controlled by a daemon called "udev" which is why it takes a split second to be noticed.
- Homework #4 was due.
- Homework #5 will be posted today.
- Hand out i2c displays today. Be careful with them!
- Midterm coming up next week in Thursday class. Probably a video for Tuesday class, more details will

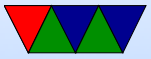


arrive via e-mail.



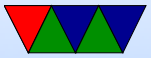
i2c

- Inter-Integrated Circuit, Invented by Philips (now NXP) in 1982
- Broadcom and others for some reason call it “Two Wire Interface”
- Two-wires (4 if you include Vdd and Ground)
- Since 2006, no licensing fees (though do have to pay to reserve number)



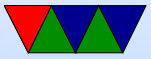
Why is i2c popular?

- Stable standard
- Relatively easy to implement
- Not many wires
- Good enough
- Cheap



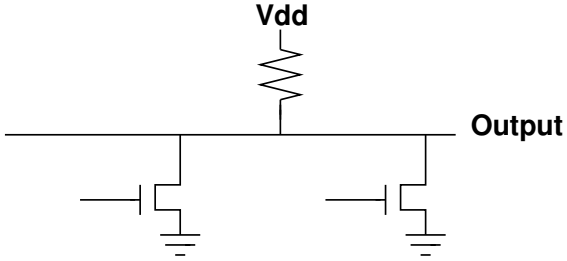
Uses of i2c

- SMbus
- DDC (VGA/HDMI) (video card / monitor communication)
- Configuring SDRAM
- Temp sensor and fan chips on motherboards
- Wii nunchuck

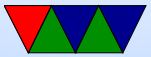


Protocol Overview

- Serial Data Line (SDA) and Serial Clock (SCL), Open Drain, Pulled up by resistors
- Open drain means output can be wired together

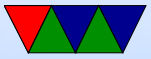


- 7-bit (or 10-bit) address
- Standard=100kbits/s, slow=10kbits/s, fast=400kbits/s



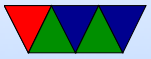
fast plus 1Mbits/s, high 3.4Mbits/s (actual transfers slower due to overhead)

- Length of bus limited to a few meters (400pF)



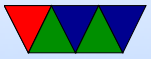
High-level Protocol

- Master (generates clock, init transaction), Slaves (responds)
- Can be multiple masters and slaves
- Master sends start bit, 7-bit address of slave, then read/write bit
- Slave responds with ACK
- Reads and writes are 8 bits of data, followed by 1 ACK



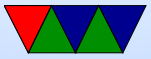
bit

- Send stop bit when done
- Address and Data set Most-significant Bit first



Low-level Protocol

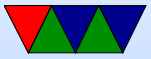
- Busses start out floating high (by pull-up resistors)
- Start bit: SDA transition high-low while SCL high
  Stop bit: SDA transition low-high while SCL high
  All other SDA transitions have to happen while SCL is low.
- To transmit bit, set SCL low, set SDA to value, set SCL high, wait 4us, set SCL low
- After every 8-bits other side sends ACK bit. If 0, more



to come. If 1, we are done (or there is an error)

- Clock stretching: slave can hold SCL low until it is done processing

- Arbitration: masters monitor SDA and won't start unless idle. Deterministic arbitration.
  If tries to send a 1 and notices something else is pulling to zero, then a collision and stops. Low addresses automatically win.



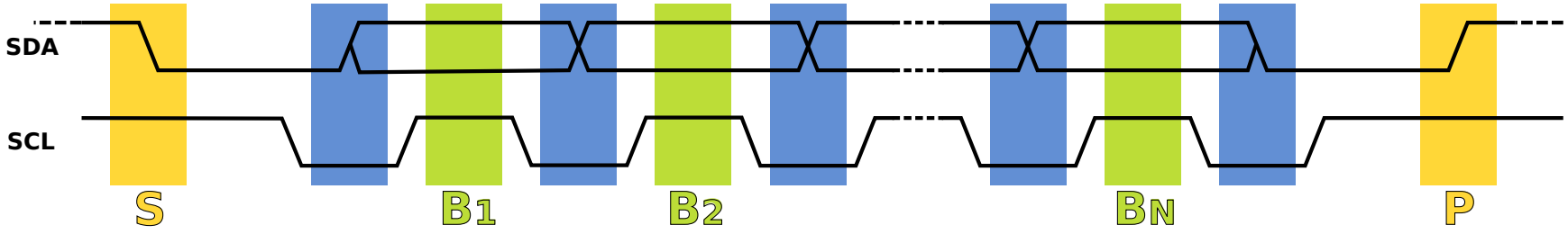
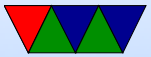


Figure 1: Protocol diagram from Wikipedia

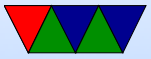


Message Types

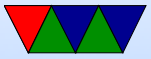
- Master writing to slave:
  Sends start, address, write (0), waits for ACK (low), then sends 8 bits of data, waits for ACK, etc.

- Master reading from slave:
  Sends start, address, read (1), waits for ACK (low), then waits for 8 bits, sends ACK if wants more, otherwise stop if done.

- Combined – can send multiple messages or to multiple



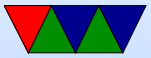
slaves by not sending stop but and instead sending a new start bit



i2c Reserved Addresses

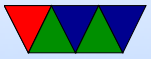
Address	R/W Bit	Description
000 0000	0	General call address
000 0000	1	START byte
000 0001	X	CBUS address
000 0010	X	Reserved for different bus format
000 0011	X	Reserved for future purposes
000 01XX	X	Hs-mode master code
111 10XX	X	10-bit slave addressing
111 11XX	X	Reserved for future purposes

10-bit addresses work by using special address above with first 2 bits, then sending an additional byte with the lower 8 bits.



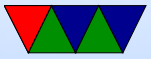
SMbus

- Enhanced i2c bus interface
- Has stricter rules about some signals
- Can do more advanced things, such as have slaves send notifications to master



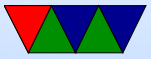
i2c and Rasp-pi

- 2 busses, only one easily accessible on Model B (other on Camera interface).
- `modprobe i2c-bcm2708` and `i2c-dev`
  May also want to edit `/etc/modules` and remove from blacklist `/etc/modprobe.d/raspi-blacklist.conf`
- Also want to install i2c-tools if possible `apt-get i2c-tools`
- i2c port 1 (`/dev/i2c-1`). Used to be i2c-0 on older



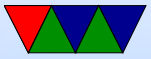
machines. Other boards (beaglebone black) likely different.

- Clock stretching buggy on the rasp-pi
- 3.3V
- default speed is 100kHz. You can change this with the baudrate= module parameter.
- i2c-1 on pins SDA=3, SCL=5
- i2c-0 on the camera interface (pad5)



i2c and Linux

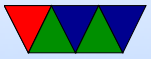
- Like with GPIOs, kernel can drive it, or be exposed to userspace
- i2c-dev module must be installed (and i2c driver)
- Open the device node, `/dev/i2c-1`
- Use ioctls I2C_SLAVE to set the address of the device we wish to talk to.
- Use standard read or write calls to communicate with



the device

- Close the device when done.

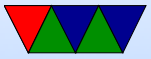
- i2c slave addresses are 7 bits, but when sent the r/w bit is put at end. This can be confusing; some spec sheets will list a slave address as 0xE0/0xE1 (8 bits, including r/w) but Linux exports this as 0x70 (0xE0 shifted right by 1).



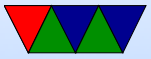
Sample i2c Linux code

For more details on this, see the HW#5 handout.

```
unsigned char buffer[17];
int display_fd;

/* open */
display_fd = open("/dev/i2c-1", O_RDWR);
if (display_fd < 0) fprintf(stderr,"Error!\n");

/* set slave address */
result=ioctl(display_fd, I2C_SLAVE, 0x70);
if (result < 0) fprintf(stderr,"Error!\n");

/* writing */
buffer[0]= HT16K33_REGISTER_SYSTEM_SETUP | 0x01;
```

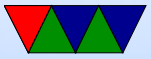


```
if ( (write(display_fd, buffer, 1)) !=1) {
      fprintf(stderr,"Error!\n");
}

/* closing */
close(display_fd);
```



i2c on the Pi – detecting

```
i2cdetect -y -r 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
```



LED Driver Chip

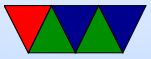
- This is a ht16k33, datasheet available:

  `http://www.adafruit.com/datasheets/ht16K33v110.pdf`

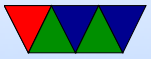
- Supports up to 16x8 LEDs, as well as keypad input. Can dim display, also blink. Common cathode.

  ```
  -|>|- common
  ```

- Works by rapidly scanning all segments fast enough cannot see.

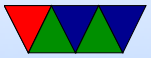


- To set up, write byte commands, high 4 bits command lower 4 bits data.
- To set up full display, write the pointer offset of internal framebuffer, than 16 bytes of on/off data.
- Actual LED hooked up is a BL-Q56D-43UG 4x7 segment Ultra-Green display common cathode.
- How do you set address? (have more than one display hooked up?)

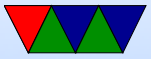


Benefit of OS

- Code is portable across all machines with i2c bus
- Can use same code on Gumstix, Rasp-Pi, Beaglebone, etc.
- Will probably need to change the bus number (It's i2c-3 on gumstix).

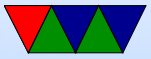


Booting a System



Firmware

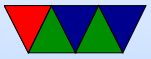
- What is firmware?



Firmware

Provides booting, configuration/setup, sometimes provides rudimentary hardware access routines.
Kernel developers like to complain about firmware authors.
Often mysterious bugs, only tested under Windows, etc.

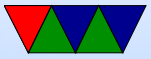
- BIOS – legacy 16-bit interface on x86 machines
- UEFI – Unified Extensible Firmware Interface
  ia64, x86, ARM. From Intel. Replaces BIOS
- OpenFirmware – old macs, SPARC
- LinuxBIOS



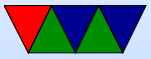
Boot Methods

Firmware can be quite complex.

- Floppy
- Hard-drive (PATA/SATA/SCSI/RAID)
- CD/DVD
- USB
- Network (PXE/tftp)

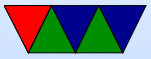


- Flash, SD card
- Tape
- Networked tape
- Paper tape? Front-panel switches?



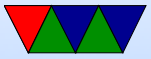
Disk Partitions

- Way to virtually split up disk.
- DOS GPT – old partition type, in MBR. Start/stop sectors, type
- Types: Linux, swap, DOS, etc
- GPT had 4 primary and then more secondary
- Lots of different schemes (each OS has own, Linux supports many). UEFI more flexible, greater than 2TB



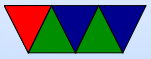
Bootloaders on ARM

- uBoot – Universal Bootloader, for ARM and under embedded systems
- So both BIOS and bootloader like minimal OSes



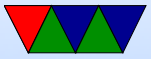
Raspberry Pi Booting

- Unusual
- Small amount of firmware on SoC
- ARM 1176 brought up inactive (in reset)
- Videocore loads first stage from ROM
- This reads `bootcode.bin` from fat partition on SD card into L2 cache. It's actually a RTOS (real time OS in own right "ThreadX")



- This runs on videocard, enables SDRAM, then loads `start.elf`

- This initializes things, the loads and boots Linux `kernel.img`. (also reads some config files there first)



More booting

- Most other ARM devices, ARM chip runs first-stage boot loader (often MLO) and second-stage (uboot)

- FAT partition
  Why FAT? (Simple, Low-memory, Works on most machines, In theory no patents despite MS's best attempts (see exfat))
  The boot firmware (burned into the CPU) is smart enough to mount a FAT partition