ECE 471 – Embedded Systems
Lecture 28

Vince Weaver

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

vincent.weaver@maine.edu

20 November 2019

Announcements

- Don't forget HW#9
- HW grades – will I hit the deadline (firm or soft realtime?)
- Talk on Friday
- Project status reports due Monday (25th)

HW#7 – Code

- You need to set the values to output in the tx buffer. The ones in the classnotes aren't right (more to hw than just cut-and-pasting class notes)
- Also we need to put things in single ended mode. Otherwise it expects every-other input, comparing the diff, instead of one input vs ground. For part 1 possibly worked by accident, but the temp probe would not work this way.
- Some trouble building 10-bit result out of two 8-bit

pieces.

- In general otherwise code was OK.

HW#8 – Questions

- Why need Vdd? To provide enough current for this particular chip needs extra current if you want parasite mode.

  You can try without Vdd but you will always read out 85C.

  Manual suggests MOSFET, but apparently it's possible on Pi if use 4.7k resistor as well as "strong-pullup=y" kernel command line option.
- Because of distance, 1-wire

- shell script
  - `#!/bin/sh` should be first line (magic number)
  - Trouble if edit on windows, why (linefeed vs carriage return)

    shebang description
  - Making executable with chmod
  - Rebooting
  - Default shell, can put other things there, like python or perl, etc, even ARM emulator
  - sh vs bash

Pi/Linux Audio

Digital Audio

- How can you generate audio (which is analog waves) with a digital computer?
- One way is PCM, Pulse Code Modulation, i.e. use a DAC.
  - Sample the sound at a frequency (say 44.1kHz), and take amplitude (16-bit audio, 64k possible values)
  - Why 44.1kHz? Nyquist theorem. Twice sample rate to reproduce properly. 22kHz roughly high end of human hearing.

- A WAV file is basically this, has the samples (8 or 16-bit, stereo or mono) sampled at a regular frequency (often 44.1kHz) to play back, write the values to a DAC at the sample rate.

- What if you don't have a DAC? (The Pi doesn't)
  - Can do PWM, Pulse-Width Modulation
  - By varying the width of pulses can have the average value equal to an intermediate analog value. For example with duty cycle of 50% average value is 1/2 of Vdd
  - Can be "converted" to analog either by a circuit, or

just by the inertia of the coil in a speaker.

- Music can be compressed too, MP3 or lossless AAC. Otherwise many tens of megabytes per song.

PWM GPIO on Pi

- Get around the fact that you can't get good timings w/o real-time OS
- Available on GPIO18 (pin 12)
- Can get 1us timing with PWM
  Software: 100us with Wiring Pi, probably less with GPIO interface.
- Which would you want for hard vs soft realtime?

- Other things can do? Beaglebone black as full programmable real-time unit (PRU)
  200MHz 32-bit processor, own instruction set, can control pins and memory, etc.

Linux Audio

- In the old days audio used to be just open `/dev/dsp` or `/dev/audio`, then `ioctl()`, `read()`, `write()`
- These days there's ALSA (Advanced Linux Sound Architecture)

  The interface assumes you're using the ALSA library, which is a bit more complicated.
  - Handles things like software mixing (if you want to play two sounds at once)
  - Other issues, like playing sound in background

- On top of that is often another abstraction layer, pulse-audio
- A mixer interface to pick volumes, muting
- For quick hack can use `system()` to run a command-line audio player like `aplay`
- Better idea might be to use a library such as SDL-mixer which handles all of this in a portable way with a nice library interface.

Pi Limitations

- Pi interface is just a filter on two of the PWM GPIO outputs
- Also can get audio out over HDMI.
- If you want better, can get i2s hat
- Pi lacks a microphone input, so if want audio in on your pi probably need a USB adapter.

i2s

- PWM audio not that great
- i2s lets you send packets of PWM data directly to a DAC
- At least 3 lines. bit clock, word clock (high=right/low=left stereo), data
- Pi support i2s on header 5

SD/MMC

- MultiMediaCard (MMC) 1997
- Secure Digital (SD) is an extension (1999)
- SDSC (standard capacity), SDHC (high capacity), SDXC (extended capacity), SDIO (I/O)
- Standard/Mini/Micro sizes
- SDHC up to 32GB, SDCX up to 2TB

- Support different amounts of sustained I/O. Class rating 2, 4, 6, 10 (MB/s)
- SDIO – can have I/O like GPS, wireless, camera
- Patents. Need license for making.
- SPI bus mode
- One bit mode – separate command and data channels
- Four-bit mode
- 9 pins (8 pins on micro)

- Initially communicate over 1-bit interface to report sizes, config, etc.
- Starts in 3.3V, can switch to 1.8V
- Write protect notch. Ignored on pi?
- DRM built in, on some boards up to 10% of space to handle digital rights
- Can actually fit full Linux ARM server on a wireless SDIO card

- eMMC = like SD card, but soldered onto board