ECE 471 – Embedded Systems
Lecture 30

Vince Weaver

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

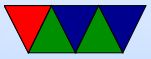
vincent.weaver@maine.edu

22 November 2024

Announcements

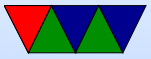
- Don't forget projects
- Don't forget HW#9
- Project status report due 25th (Monday before Thanksgiving)

- Hopefully midterm went OK

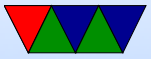


Project Status – Due Monday (25th)

- A one-line statement of your project topic
- A short summary of the progress you've made so far
- List any parts you need that you don't have yet
- List if you're willing to present early (Friday the 6th, Monday 9th or Wednesday 11th vs Friday the 13th) (there will be some bonus for presenting early)
- Identify the problem: this can be brief, the idea is you picked the topic, but can you identify the hardware/ software problem you need to solve to complete it.

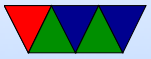


- For example: you're making a line-following robot.
- The hardware problem to be solved would be assembling hardware that can detect a line
- The software problem to be solved is code that can read the sensor, detect the line, and provide feedback to the motors



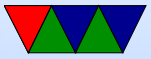
Other I/O You'll find on Embedded Boards

Prioritized for things people might be using for the projects



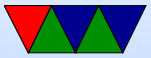
Starting Programs at Boot (old way)

- `init` process starts first
- Traditionally would start various shell scripts under /etc (the name and order of these can vary a lot)
- Currently you can still put things you want to run at start in the `/etc/rc.local` shell script
- At some point this will stop working, due to systemd



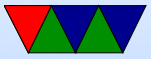
Starting Programs at Boot (new way)

- Used when init is `systemd`
- Need to create a systemd unit file
- TODO: I need to test this and write up a proper example here



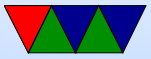
i2c/SPI boards

- If you borrowed a sensor, your best bet is the datasheet
- These are all adafruit breakouts so relatively well documented
- They have sample code, but for arduino or python
- If you run into trouble let me know. Some boards I have sample code for, others might take a bit longer
- Also if device doesn't work let me know. Actually happened last year one of the sensors was broken



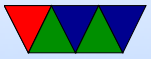
Wii Controllers

- If you want some interesting I/O you can find old controllers for the wii console
- The wiimotes themselves are Bluetooth and in theory can talk to a Pi (I haven't done this yet)
- The various sub-adapters talk i2c (like the nunchuck, or classic controller) and can easily be used with a pi



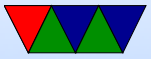
Wii Nunchuck

- Fairly easy to interface (I have breadboard adapters)
- Put onto i2c bus. Device 0x52
- Send handshake to initialize. Use longer one (0xf0/0x55/0xfb/0x00) not the simpler one you might find(0x40/0x00). This works on generic nunchucks and possibly also disables encryption of results.
- To get values, send 0x00, usleep a certain amount, and read 6 bytes. This includes joy-x, joy-x, accelerometer x/y/z and c and z button data. More info can be found



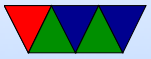
online.

- byte0 = joy-x
- byte1 = joy-y
- byte2 = top 8 bits accelerometer x
- byte3 = top8 acc y
- byte4 = top8 acc z
- byte 5 is bottom 2 z,y,x then button c and z (inverted?)

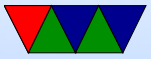


Keyboards

- Keyboards are just a matrix of keys that get scanned, ideally tell you when key pressed and when released
- Even older keyboards with embedded systems with microcontrollers doing the scanning
- Desktop PC systems went PC, XT, PS/2, then USB (also wireless/bluetooth)
- PS/2 is just a serial-like bus, clock/data and you can in theory bitbang it over GPIO but maintaining timing can be tough

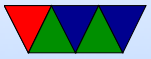


- USB is a lot more complex
- Luckily Linux abstracts all this

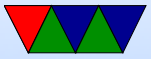


Linux and Keyboard

- Linux assumes "CANONICAL" input mode, i.e. like a teletype. One line at a time, blocking input, wait until enter pressed.
- You can set non-CANONICAL mode, async input, and VMIN of 1 to get reasonable input for a game. Arrow keys are reported as escape sequences (ESCAPE-[-A for up, for example).
- Even lower-level you can access "RAW" mode which gives raw keycode events, etc.

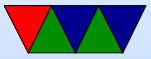


- See the `tcgetattr()` and `tcsetattr()` system calls
- There are libraries like ncurses that abstract this a bit. Also GUI and game libraries (SDL).



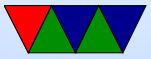
Faking Linux Input Events

- How to insert input events into Linux, i.e. have a software program fake keyboard/mouse/joystick events.
- Linux supports a "uinput" kernel driver that lets you create user input.
- There is some info on a library that makes this easier here: `http://tjjr.fi/sw/libsuinput/`
- It has examples for keyboard and mouse. Joystick should be possible but there's no sample code provided.
- Python wrappers seem to exist too.

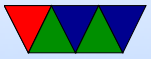


Digital Audio Output

- 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.

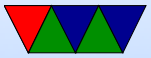


- A WAV file is basically this, has the samples (8 or 16-bit, stereo or mono) sampled at a regular frequency (often 44.1kHz) to play back, write the values to a DAC at the sample rate.



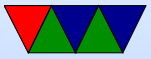
What if no DAC? (Pi has none)

- 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.
- TODO: plot



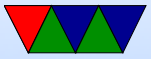
Saving space with audio compression

- This was more relevant before everyone put all their music in the cloud
- Can be tens of megabytes per song.
- Music can be compressed
- Lossy: MP3, ogg vorbis
- lossless AAC, FLAC

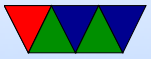


PWM GPIO on Pi

- You can't get good timings w/o real-time OS
- PWM lets you set a continuous cycle with duty cycle on GPIO
- On pi available on GPIO18 (pin 12)
- Can get 1us timing with PWM in Hardware
  Software: 100us Wiring Pi, 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)

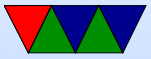


200MHz 32-bit processor, own instruction set, can control pins and memory, etc.



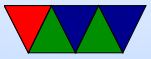
Linux Audio

- 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



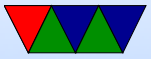
Linux Audio – Libraries

- On top of ALSA is often another abstraction layer
  - pulse-audio – from the same people who made systemd
  - A mixer interface to pick volumes, muting
- More Recently people using pipewire
  aside on DSP to make tiny speakers sound good, intentionally over-drive to sound good but have to model temperature, m1 Linux support has to be careful or you can blow out speakers in software
- JACK interface for low-audio sound between processes



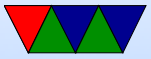
Linux Audio – Programming

- 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.



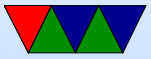
Audio Input

- Pi lacks a microphone or aux input
- if want audio in on your pi probably need a USB soundcard
- Linux can mostly abstract this
- I also have some microphone boards (SPI/i2c?)
- electret vs MEMS microphones
- Can also use ADC board we have, but need extra circuitry to match to the proper input signal



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



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