CS49n – Using bits to control atoms

Instructor: Dawson Engler

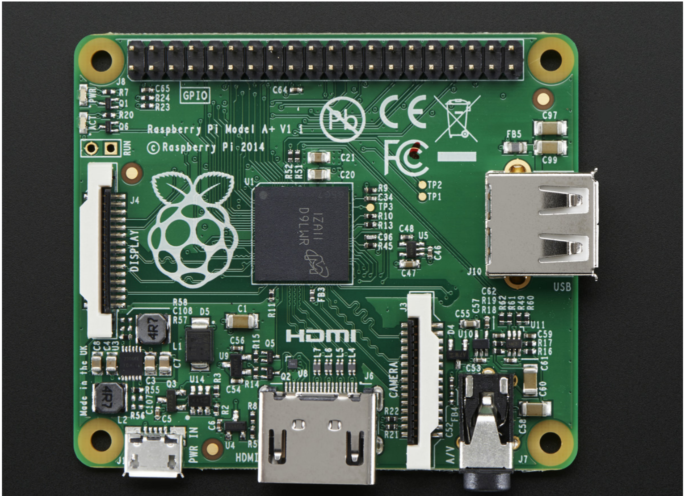
Stanford University

Outline

1. What, Why

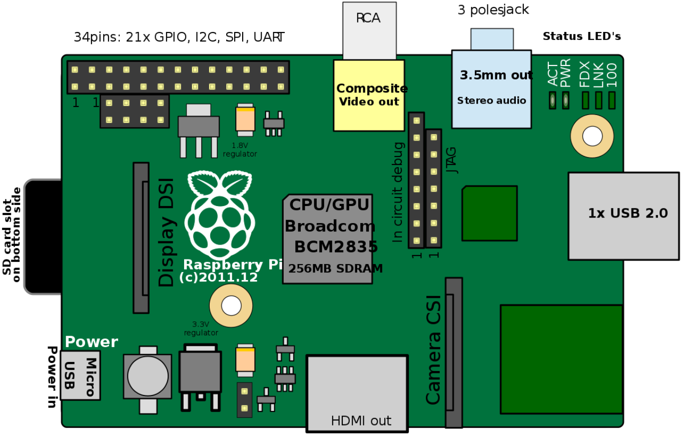
What

- Write small, clean pieces of code to control sensors and do interesting tricks on the raspberry pi A+.
  - `https://en.wikipedia.org/wiki/Raspberry_Pi`



What

- **Write small, clean pieces of code to control sensors and do interesting tricks on the raspberry pi A+.**
  - https://en.wikipedia.org/wiki/Raspberry_Pi



What you will build

- Sonar, IR, hall-effect, accelerometer, microphone.
- Interrupts.
- Network.
- I2C, SPI.
- Bootloader.
- Modern bug finding tricks.
- Final project: open ended. Expense account at sparkfun/adafruit.

Why

- **If you can write this kind of code, you can write pretty much anything.**
- **No abstractions: will understand (a) reality.**
  - Once you get this, easy to delta to other examples.
- **The real world is not a clean, textbook chapter.**
  - Difficult to understand documents.
  - Wrong.
  - Incomplete.
  - Not written to be used.
  - You will learn how to orientate and operate in such a world, without a lot of drama.
  - We chose r/pi A+ because lots of useful blog posts for how to do things. (Later pi's are less helpful).

Why R/pi

- **Most OS classes (cs140) use a fake simulator.**
  - Alot of work. Not that cool at the end.
- **r/pi = real computer for about $20.**
  - HDMI, SD card, memory: can put mouse, display, keyboard, have alot of control.
- **Unlike most machiness: Makes interacting with the real world very easy.**
  - Can build many interesting systems because can use weird hardware.
  - motion sensor, ir sensor, accelerometer, gyroscope, light sensor, etc.
- **Since bare metal: Very easy to build cool tools that are hard otherwise (gprof, eraser).**

You will develop two super-powers

- **Differential debugging. You write code, it doesn't work. Error could be:**
  - The code you wrote;
  - Hardware fault (smoked something);
  - Wiring mistake;
  - Subtle cache issue;
  - Compiler problem;
  - ...
  - You will get good at breaking down problems to isolate.
- **Epsilon-steps.**
  - Engler's theorem: Given a working system W_k and a change c, then as $c \to \epsilon$ then the time T it takes to figure out why $W_n + C$ doesn't work goes to 0 ($T \to 0$).
  - For a fixed amount of IQ, the smaller the step you can take from a working system, the faster you can debug when it doesn't work.