Embedded Systems

Detour: Number Systems

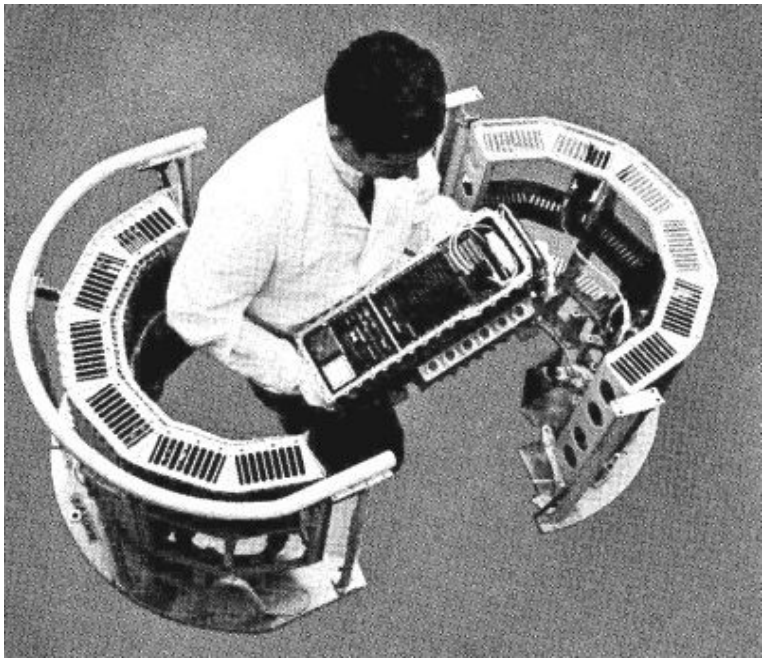
- Humans work in Base Ten (Decimal)
  - Each digit is 0-9, each place is 10^place
- Computers work in Base Two (Binary)
  - Each digit is 0 or 1, each place is 2^place
  - Most devices use “Two's Compliment” which allows for signs with only the two symbols.
  - Decimal/fractional numbers are usually represented as Floating Point Numbers (“Floats”) that use several number fields
- Often Hexadecimal (Hex – Base 16, symbols 0-F) or Octal (Base 8, symbols 0-7) are used as a compromise

Embedded Systems

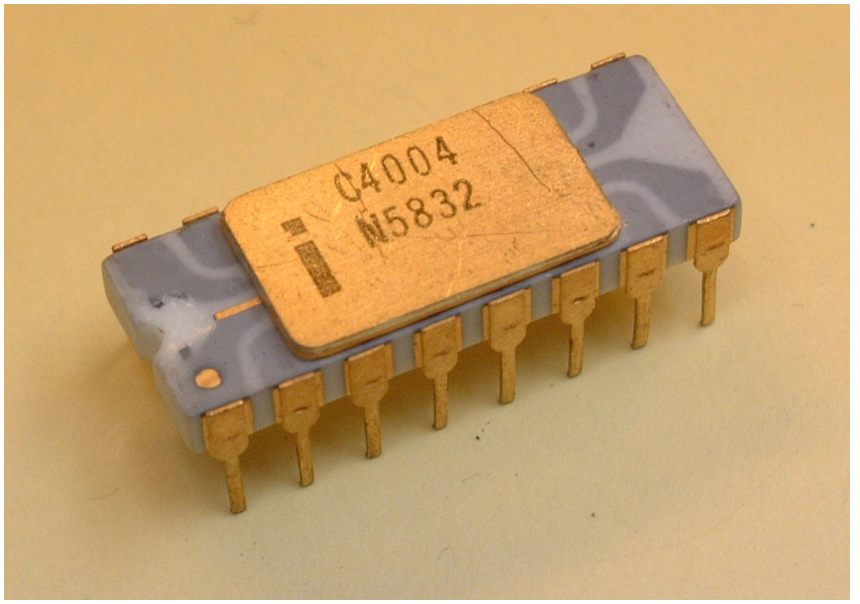
- Computers, customized for a specific task
- Range in size and complexity from flashlights to airplanes.
- Most often, Microcontrollers
  - Fixed Memory, RAM, CPU and I/O on one chip.
- Other flavors:
  - ASIC
  - FPGA/Programmable Logic
  - DSP
  - Single Board Computers
  - Special Function Computers

History

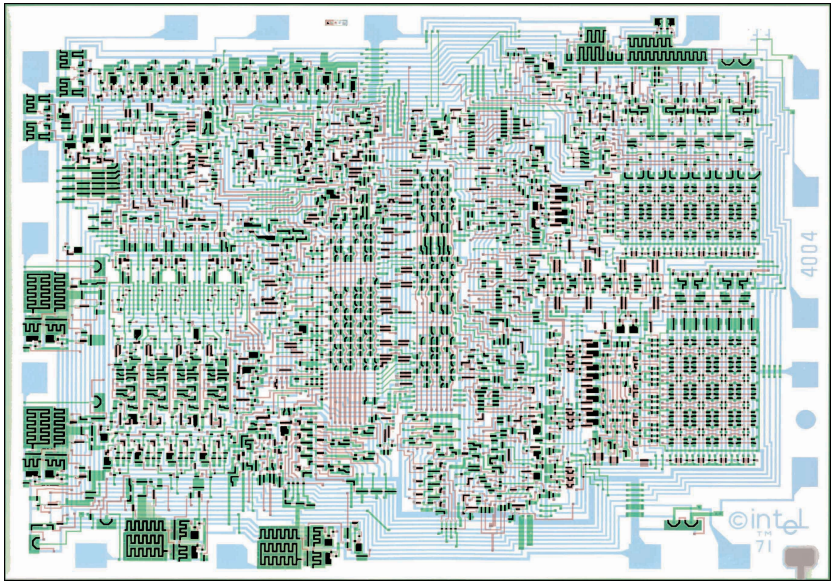
- D17 Computer (Minuteman Missiles) in 1961
- Apollo Guidance Computer in 1966
- Intel 4004 Single-Chip CPU in 1971
- TI TMS 1000 in 1971-1974
  - Powers the Speak-and-Spell
- Atmel and Microchip introduce programmable models in 1993.



D17



4004 Package



4004 Mask

TMS1000

Anatomy

- Attached to something to Monitor and/or Control
- Less user-modifiable
- Usually less powerful
- Many embedded computers per recognizable computer.
  - Including several INSIDE the computer.
- SCADA – **S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition

Microcontroller Families

- Order of 20 common families in circulation
  - Many more obscure designs around.
- Many are closely related to or directly descended from “Full” computers.
- 8-Bit micros make up about half the CPUs sold every year.

8-Bit

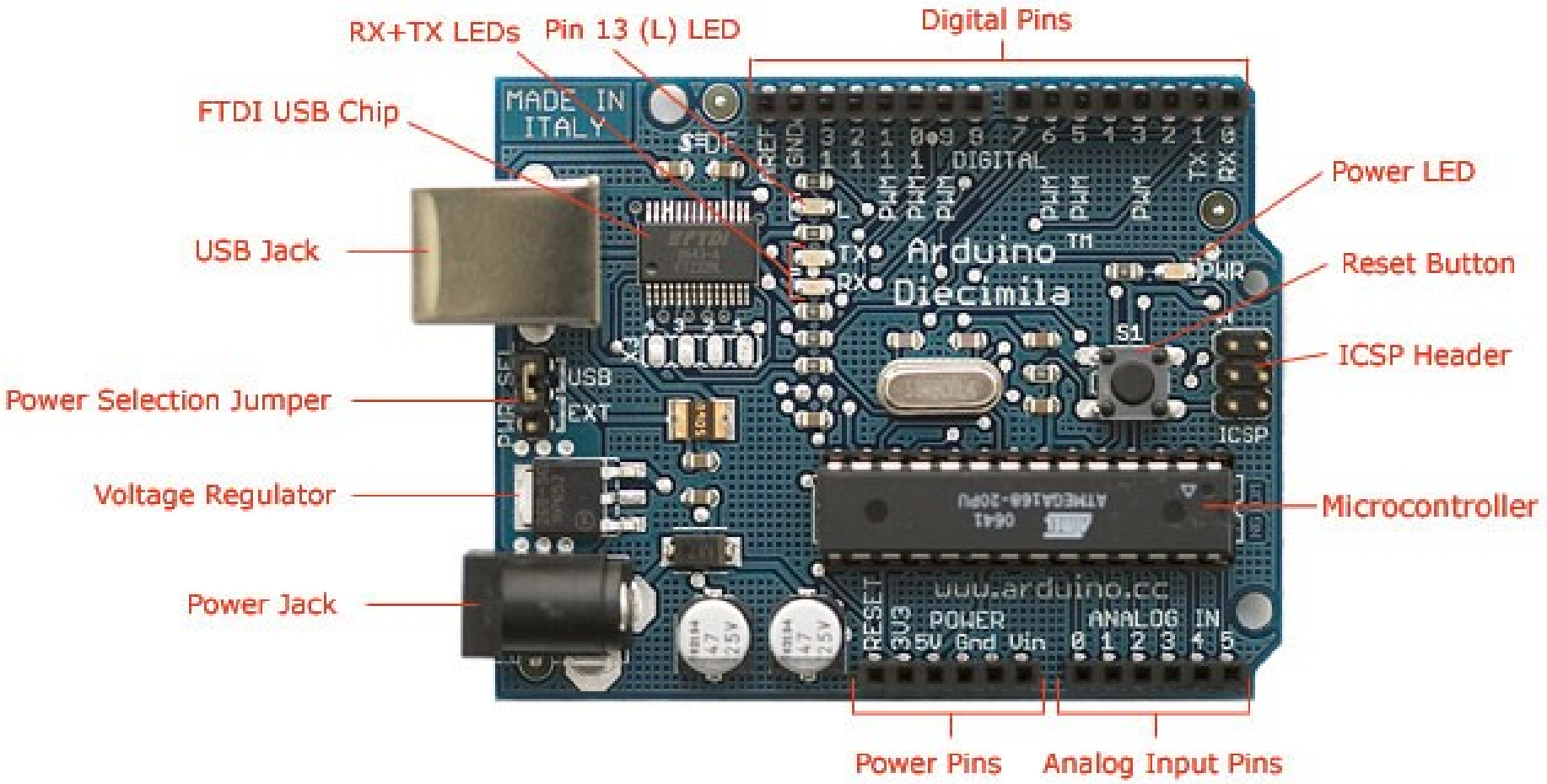
- 8051 – Since 1980, but direct descendent of Intel's MCS-48 from 1976.
- PIC - “Peripheral Interface Controller” ubiquitous, around since 1975.
- 68HC11 – uC cousin of the Motorola 6800 from 1985.
- Z80 – Fancier Intel 8080, same parent as x86 PCs.
- AVR – Atmel's line from 1996, descendent of Norweigan college students' design.

16- and 32- Bit

- MSP430 – TI 16-bit design, from the 1990s, like a PDP-11.
- ARM – Designed for PCs in 1987, good for mobile, licensed to everyone.
- ColdFire – Motorola 68K's embedded variant.
- X86 Embedded Boards

Arduino

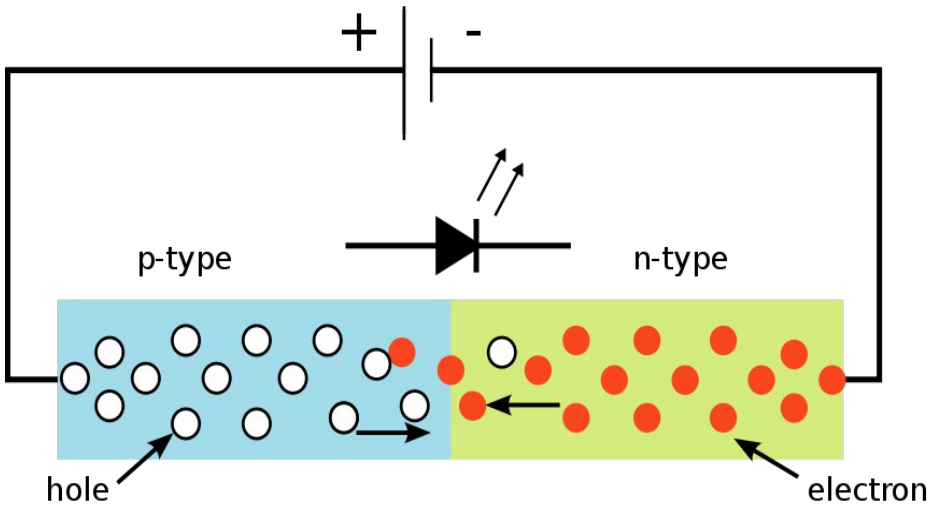
- Project started in Ivrea, Italy in 2005
  - Based on earlier Processing and Wiring projects.
- Effort to make uCs accessible to hobbyists, artists, and other non-engineers.
- Based on an AVR ATMega8 family part
- C++ like language, Java IDE
- Great for rapid prototyping

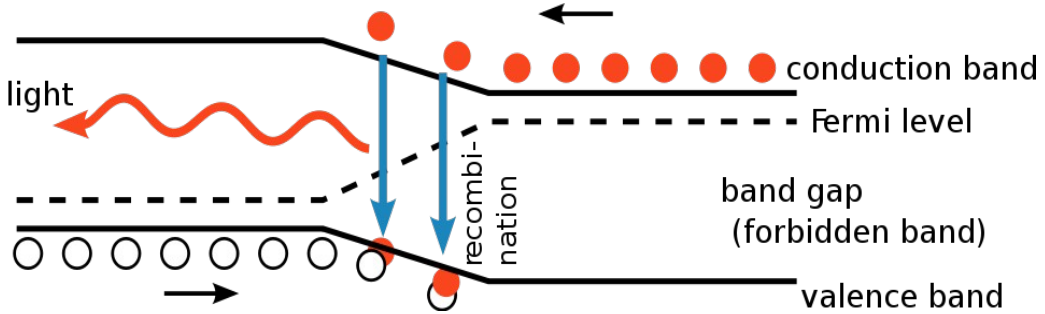


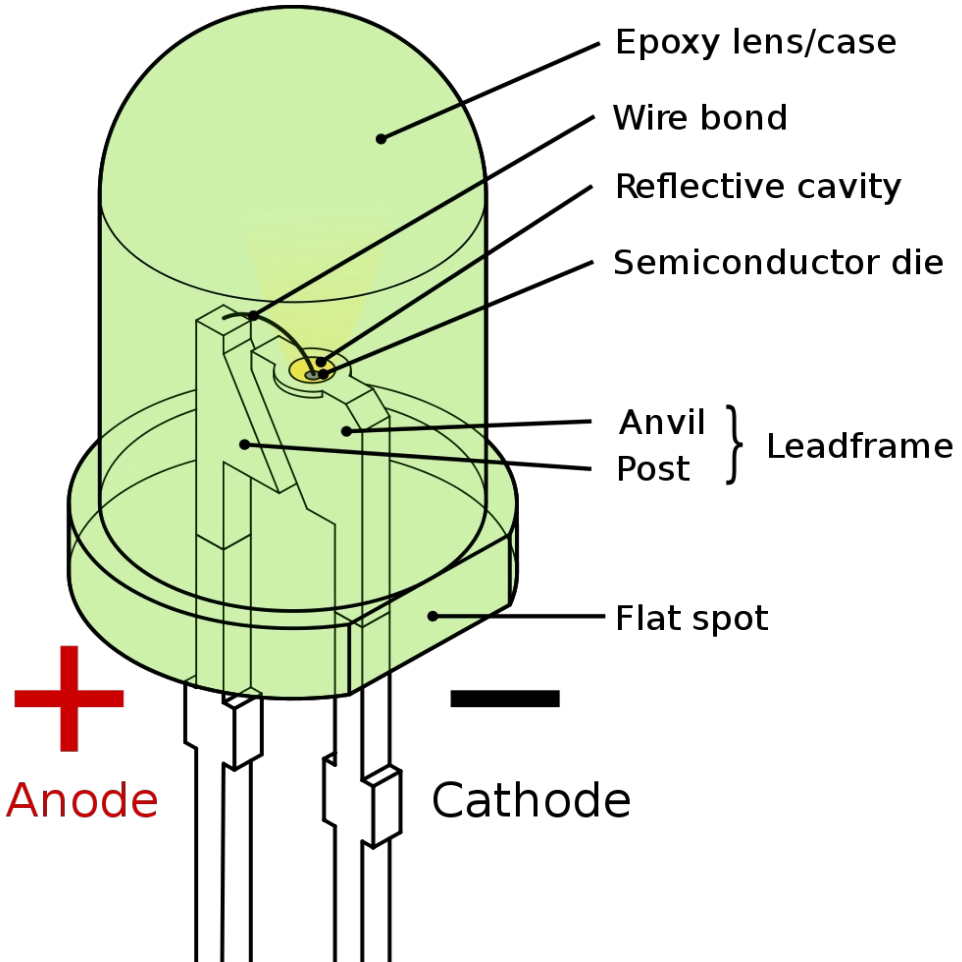
Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.

LEDs

- Light Emitting Diode
- Requires a Current Limiting Resistor



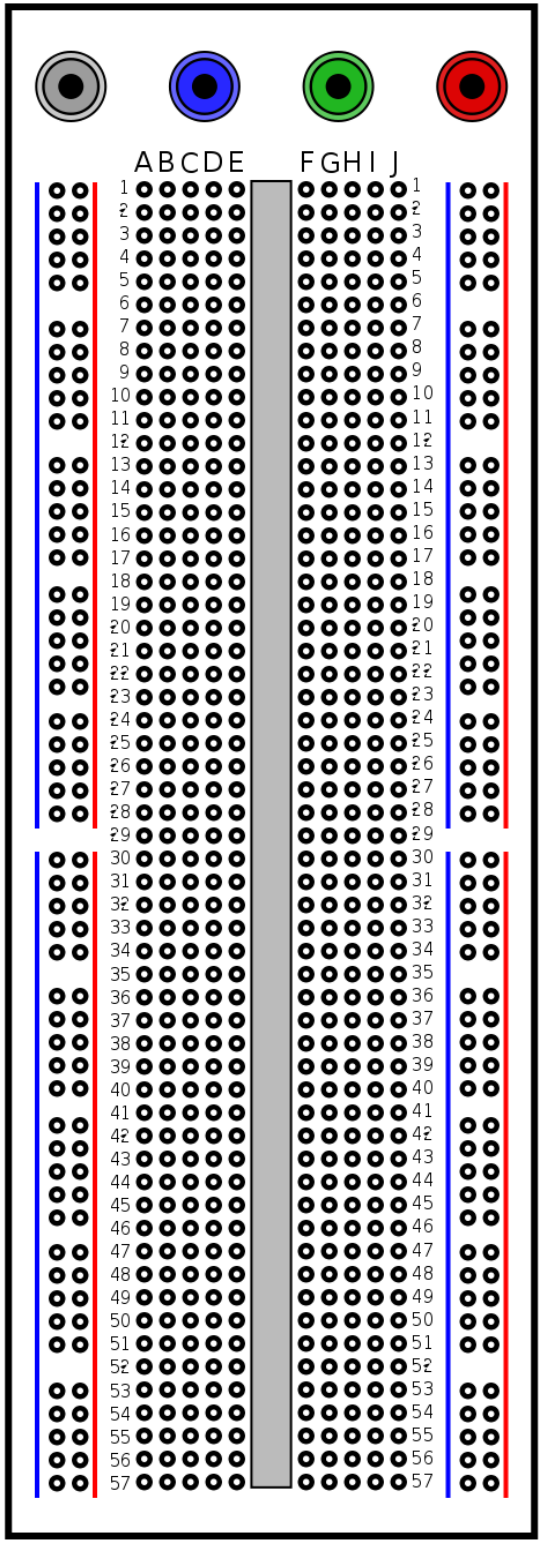




Embedded Development Two
Putting Parts Together

Breadboard

- Each numbered row is connected internally, up to the middle separator.
- Each marked Bus Column is connected internally
  - Sometimes spit in the middle
- DIP ICs straddle the middle
- Good for prototyping, bad for reliability and sensitive signals.



Breadboard Best Practices

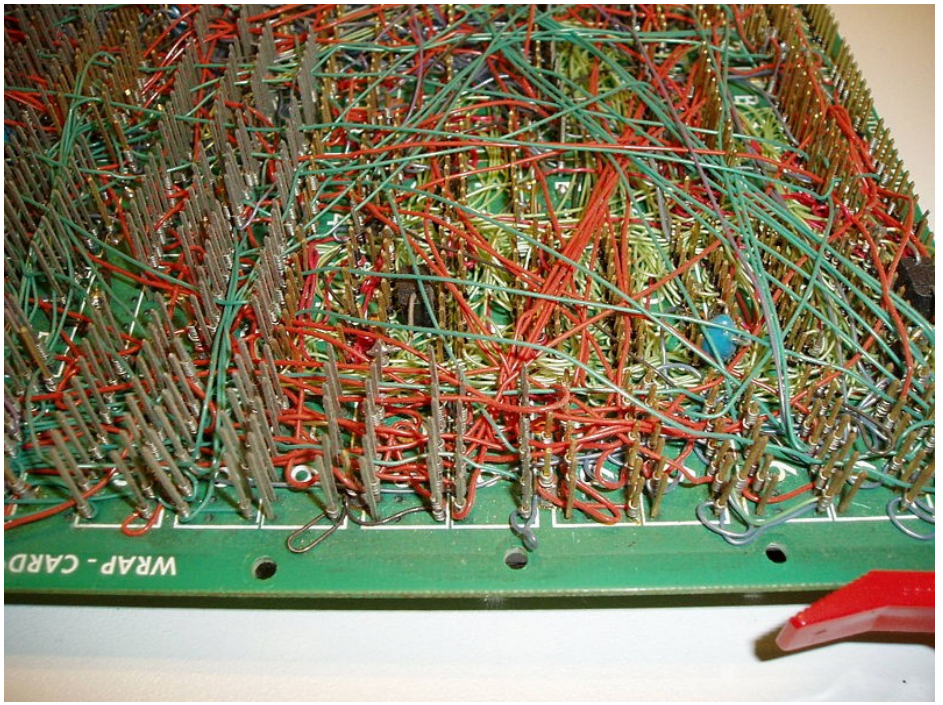
- Neatness Counts. A lot.
- Color Code wires
- Use wires of approximately the correct length
- Start with ICs – always orient them the same way
  - Then add power and ground
  - Then add internal connections
  - Then add chip-to-chip connections
  - Then everything else
- Tape flags and other labels are your friend

Now What?

- Breadboards are expensive, labor-intensive, bulky, and fragile.
- Wire Wrap
  - Out of vogue, point-to-point wired on long pins
- Perfboard
  - Accessible and Universal
  - Not repeatable, error prone
- Printed Circuit Boards
  - Design with CAD software, like gEDA and EAGLE
    - Send away
    - Use copperclad with printer or photo transfer and chemical baths.
    - Or, a mill it
  - Hobby tool called Fritzing

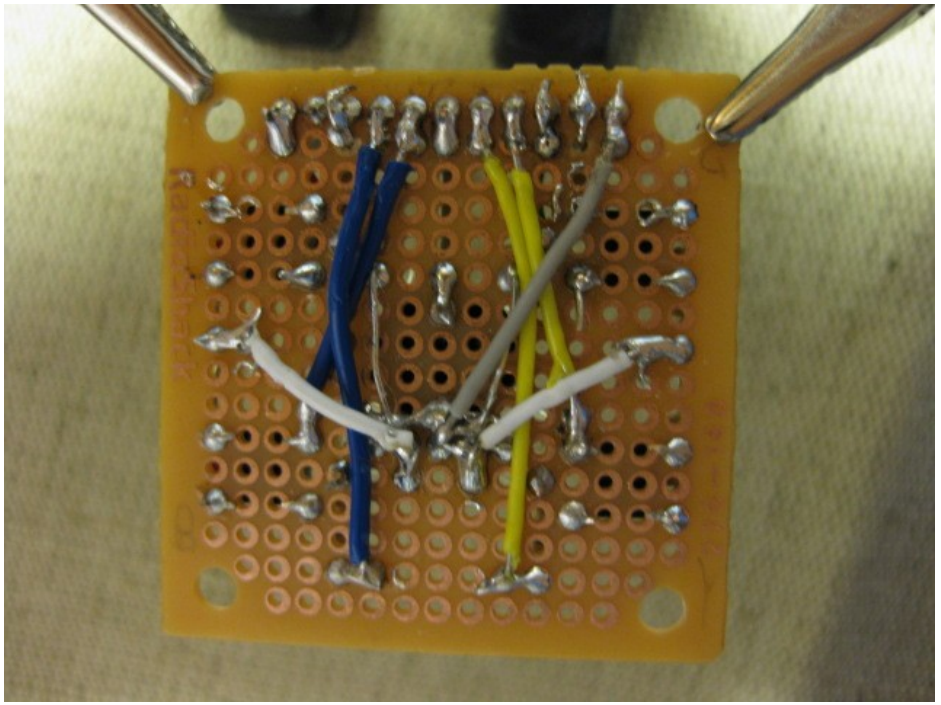
Examples

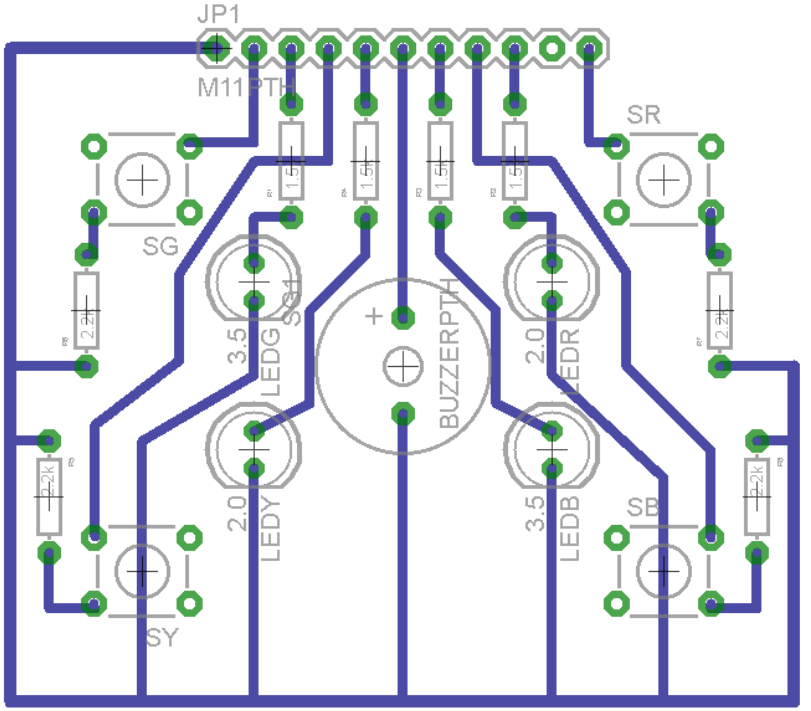
- Wire Wrap



- Household Etching

Perfboard V. PCB



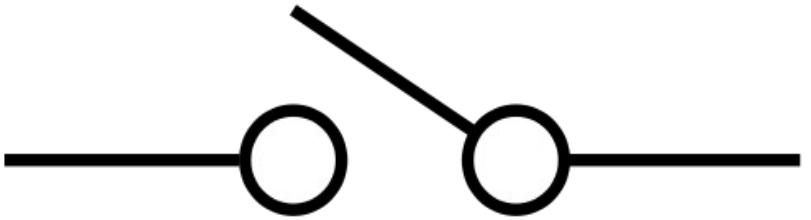


Buttons and Switches

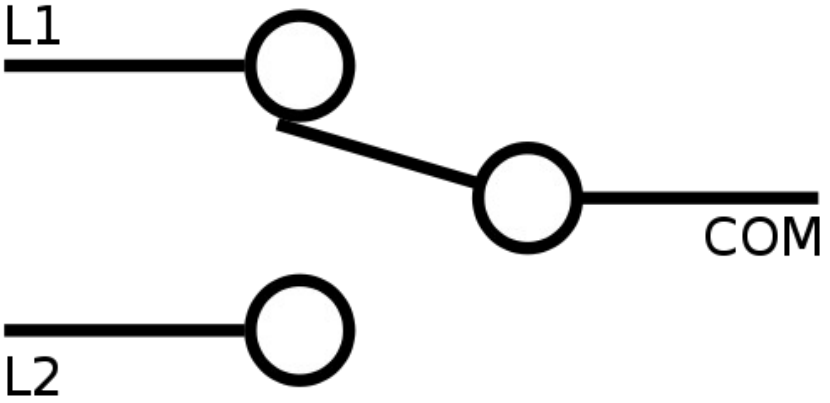
- A simple button just makes or breaks a connection
- Need some kind of reliable binary on/off
  - Unconnected pin = Unknown value
- Pull a signal between Vcc and Gnd
  - Dead zone between 1 and 0 voltage - “Hysteresis”
  - Pullup and Pulldown Resistors
- Many kinds of switch

Buttons and Switches (cont'd)

- SPST
  - Single-Pole Single-Throw

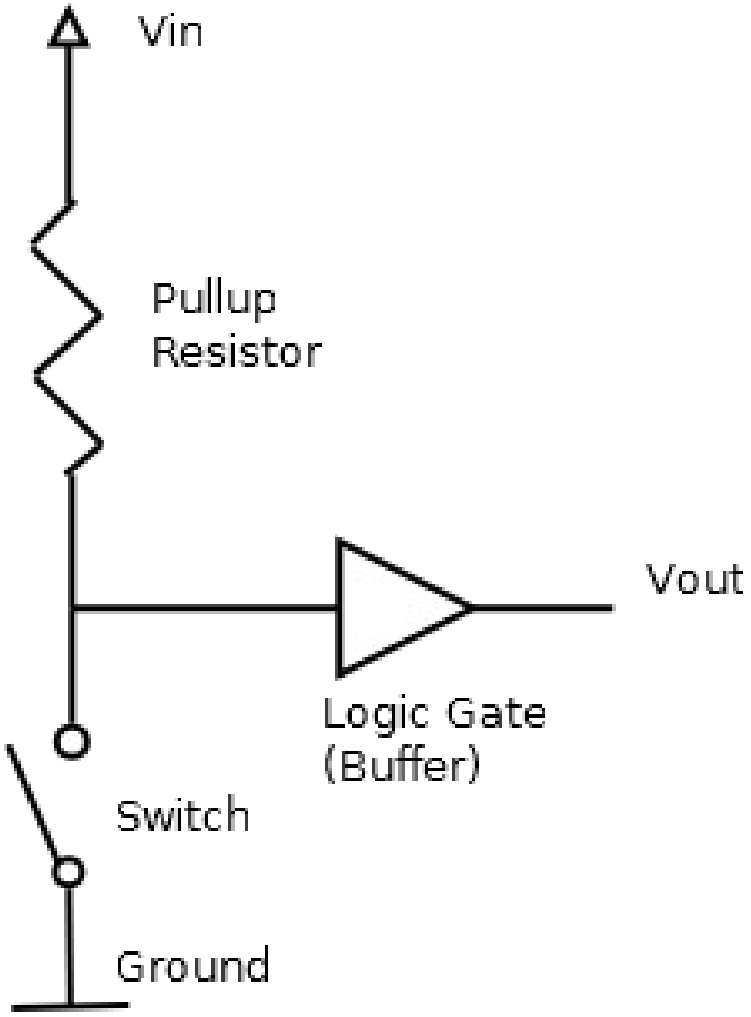


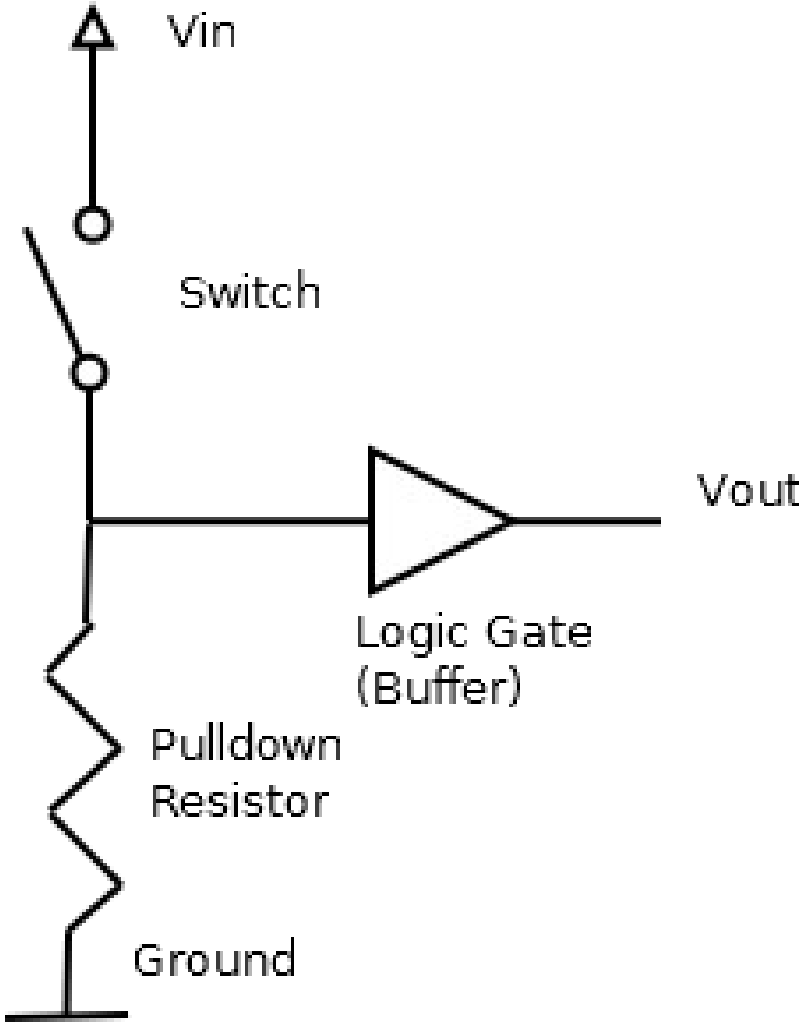
- SPDT
  - Single-Pole, Double Throw



- NC/NO - **N**ormally **C**onnected / **N**ormally **O**pen

Pullup and Pulldown



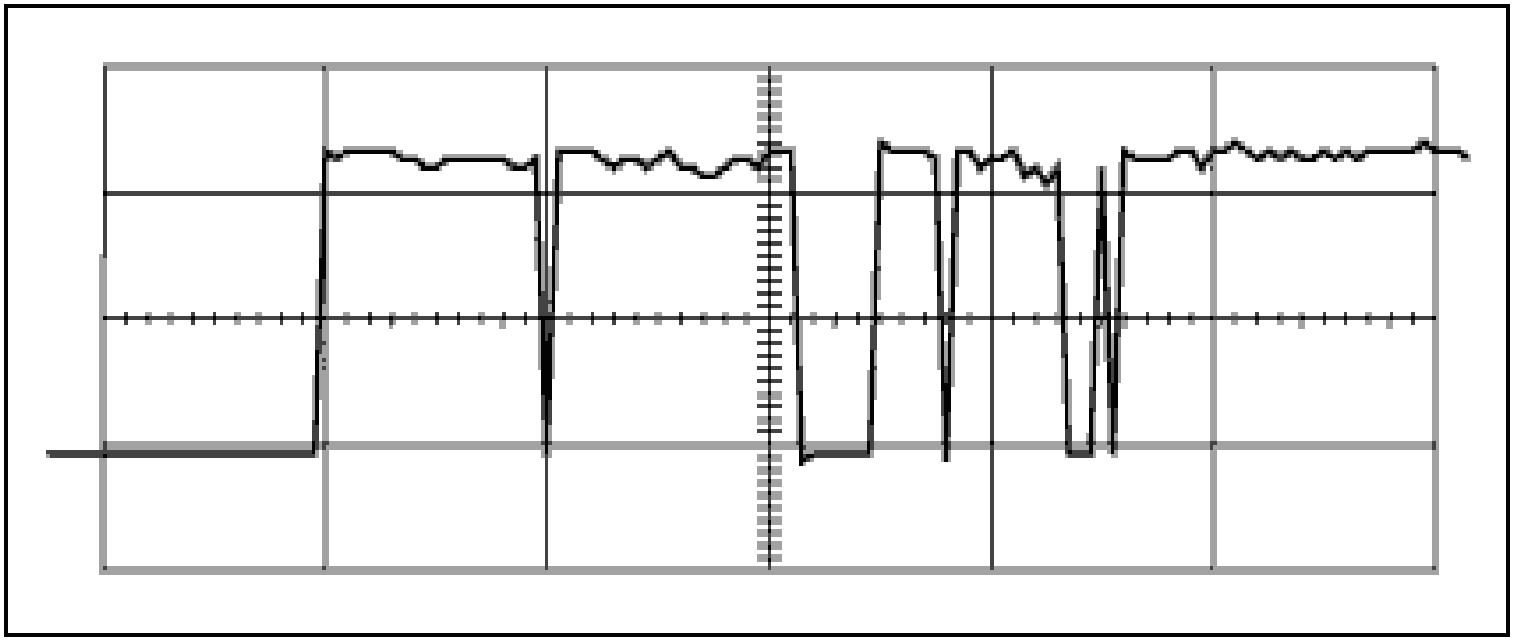


Integrated Pullups

- Many devices, especially microcontrollers, have built-in pullup resistors
- Enable in software when setting up a pin/port
- On an Arduino, the following code will pull up an input pin.

```
pinMode(pin, INPUT);     // set pin to input
digitalWrite(pin, HIGH);  // turn on pullup resistors
```

Switch Bounce



- Flipping a switch or pressing a button doesn't make a single clean transition
  - Read too fast, get the wrong value
  - Extra events on “when the switch changes”

Switch bounce image courtesy of Maxim Semiconductor

Handling Bounce

- Hardware Methods
  - RC Circuit
  - Latch
  - Monostable multivibrator
  - Timer
  - State machine
- Common factor: Require additional parts.

- Software Methods
  - Fixed Delay
  - Timer/Comparator

Debouncing on the Arduino

```
int val;
int val2;
int buttonState;
void loop(){
    val = digitalRead(switchPin);
    delay(10);
    val2 = digitalRead(switchPin);
    if (val == val2) {
        //Act on Input here
```

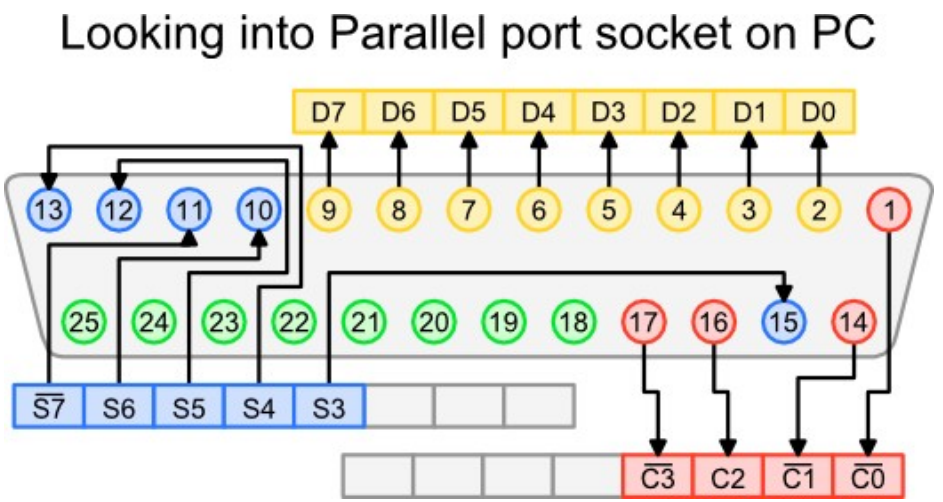
Alternative technique example at:
http://www.arduino.cc/en/Tutorial/Debounce

Communication

- Main Choices:
  - Serial vs. Parallel
    - Serial requires fewer pins
    - Parallel allows more data per action
    - Different decoding requirements
  - Synchronous vs. Asynchronous
    - Asynchronous requires a start/stop symbol
    - Synchronous requires a separate sync signal

Parallel Protocols

- Bundles of Discrete Logic Signals
- One Hot/Encoded symbols
- IEEE 1284 PC Parallel Port
- PCI (not express)



Serial Protocols

- Cheaper to implement (Cables and transceivers)
- Less susceptible to interference
  - Crosstalk, Clock Skew
- Speed issue:
  - Only one bit moved at a time
- Complexity Often discussed via Wire Count
  - 9-wire, 5-wire 4-wire, 3-wire, 2-wire and 1-wire common

Common Protocols

- RS232
  - From 1962
  - Compliant designs must handle ±25V
    - Most don't, and run at 3.3V or 5V
- SPI (“Four Wire”)
  - Built in to many uC designs, including Atmega8
- I2C (“Two Wire”)
  - SMBus (computer sensors) is a subset
- 1-Wire
  - Only one wire and a ground connection

RS232

- Option conventions
  - Speed: 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600 and 115200 bit/s
  - (Data/Parity/Stop)
    - Data: Number of data bits per frame, 5,6,7, **8** or 9
    - Parity : **N**one, **O**dd, **E**ven, **M**ark, or **S**pace
    - Stop: Number of sync bits at end of frame (usually 1)
  - Flow Control
    - RTS/CTS, DTR/DSR (using wires)
    - XON/XOFF (escaped Signals)
    - None or Higher Level

I2C/TWI

- Very popular for small, low power board integration
- Up to 112 Devices
  - One master, switchable at any time
- 100Kbit/s low power mode, up to 3.4Mbit/s high speed mode
- Single Ended
  - One Wire signals, one wire carries reference
- Clock Stretching
  - Any slave device can hold the clock until it is ready to respond

Arduino

- Has SPI and RS232 Support
- Communication between the Arduino and PC are via RS232 - 9600 (8/N/1)
  - Bridged from USB with FTDI FT232RL or programmed ATmega8U2
- Serial library is always included
  - Serial.begin(9600); in setup
  - Serial.println(); to write
- Have to #include <SPI.h> for SPI support

Homework

- Write a sketch that correctly counts the number of times a button attached to the Arduino has been pressed, and prints it to the serial monitor.
- Bring a copy of your code to turn in next week.

Analog I/O
Datasheets

Digital Devices, Analog World

- Analog = Continuous Time, Continuous Value
- Digital = Discrete Time, Discrete Value
- Microcontrollers, like all modern computers, are digital devices.
- The world is an analog place
- Input: Analog to Digital Converters (ADC)
- Output: Digital to Analog Converters (DAC)

Terminology

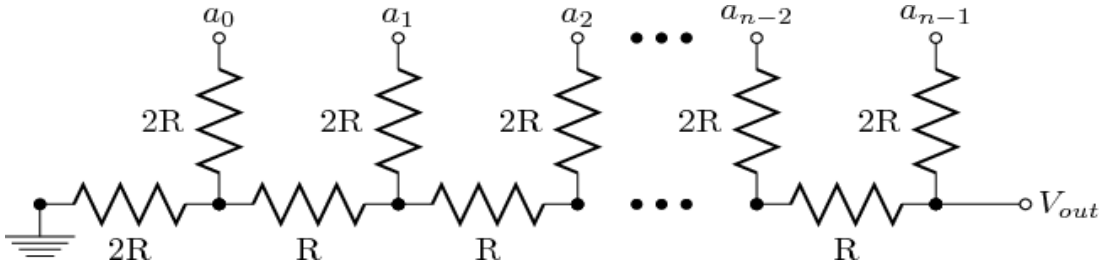
- Range
  - The spectrum of values a device can manage
    - Usually Volts, Usually limited user-configuration
    - Watch limits: -5V to +5V is not the same as 0-10V
- Resolution
  - The number of discrete levels a device can encode
    - Often quoted in Bits
  - Equivalently: The smallest change the device can detect/produce
    - Often quoted in Volts/Div
- Rate
  - How fast/often the signal is sampled
    - Must be twice as fast as the fastest signal to be sampled (Nyquist–Shannon sampling theorem)

Terminology, contd.

- Signal-to-noise
  - The relative size of the desired signal to background signals
    - Often quoted in dB – $10*\log_{10}(\text{value})$
- Linearity
  - Most ADCs designed so each step is the same size
    - Non-Linearity measures the deviation from that ideal
    - Some ADCs are intentionally non-linear

Kinds of DAC

- R-2R Resistor Ladders



- Thermometer-coded DAC
  - Voltage source per output value
  - Turn on the closest match
  - VERY expensive
- Most DAC is accomplished with PWM
  - PWM – Pulse Width Modulation
  - Requires only one pin, and a timer

PWM

- Carrier Frequency
  - Limiting factor: Counter resolution
- Filter to smooth out the pulses
- Many devices require no filtering
  - Lights have Persistence of Vision
  - Motors have Inductance

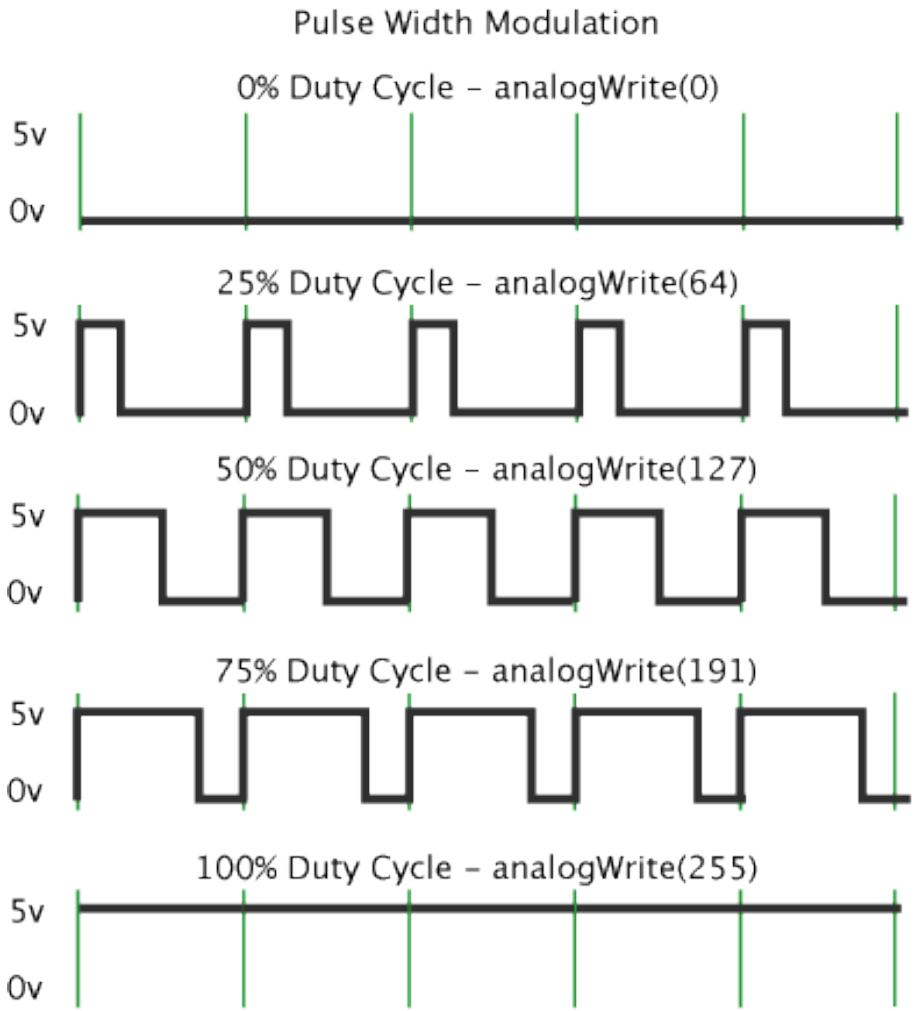


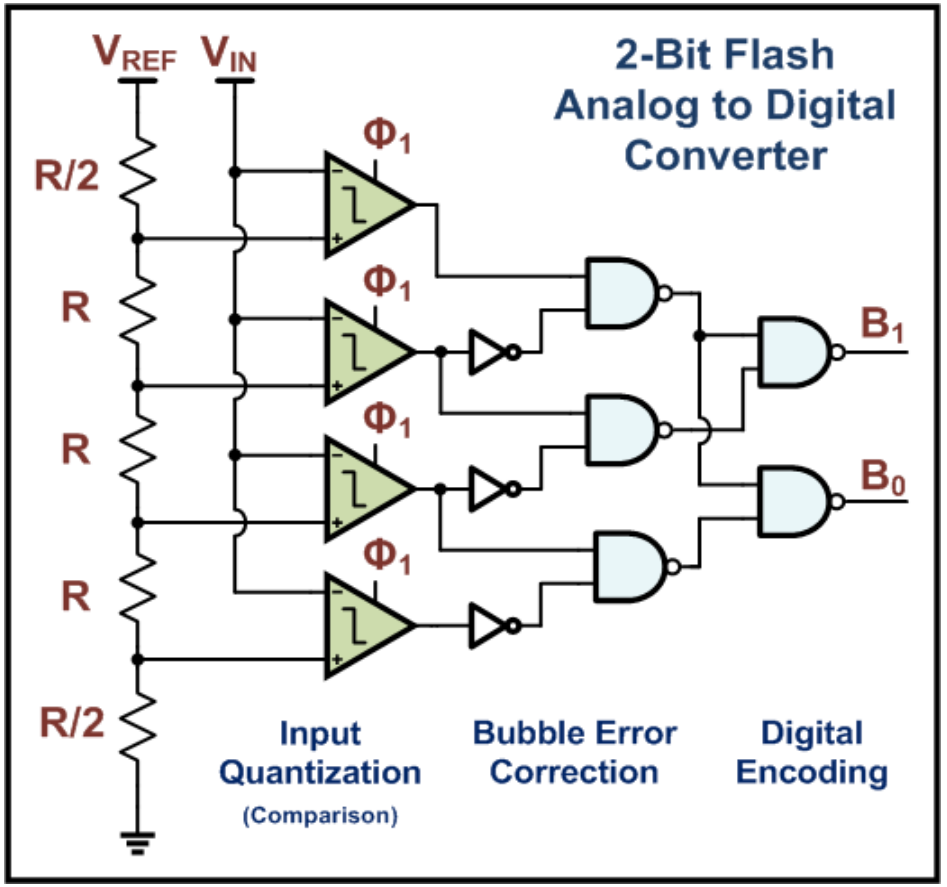
Image courtesy of Arduino.cc

Kinds of ADC

- About a dozen common varieties
- Tradeoff between complexity and Speed
- Many designs use a DAC, and a Comparator to iteratively match the input
- Design goal: Lowest sufficient resolution
- Available as discrete components
- Usually built-in to uCs, included in some sensors.

Flash ADC

- Large bank of comparators
  - Tests against each possible encoded value
  - The closest match is selected
  - $2^N - 1$ comparators for N bits of resolution
- Extremely fast
  - Extremely expensive
  - Hard to manufacture
  - Subject to noise
  - Generally low resolution



Ramp Compare

- Uses a DAC to create a comparison signal
- Single comparator continuously compares input to generated signal
- Repeatedly “ramps” DAC over the range
- Records DAC value when signals match

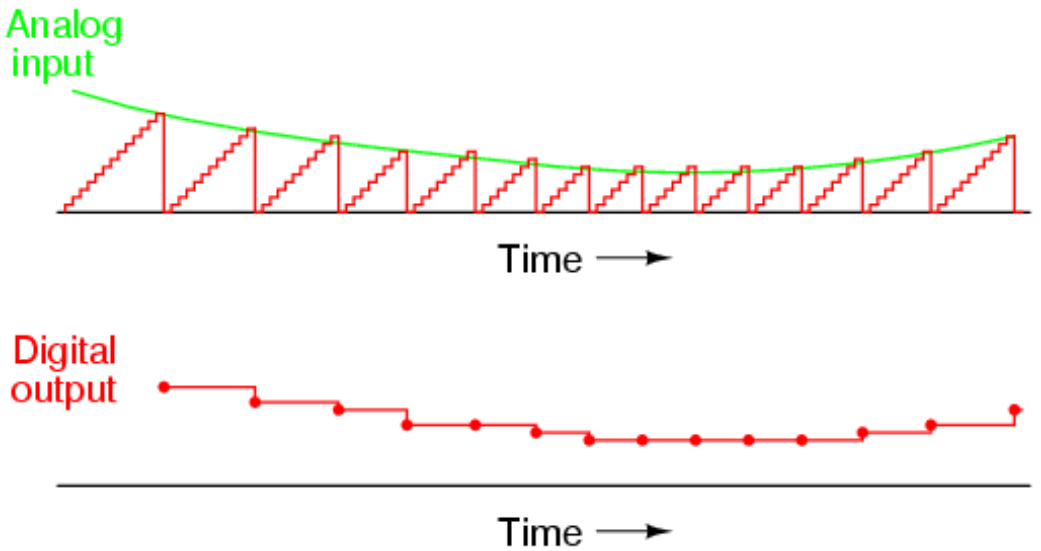
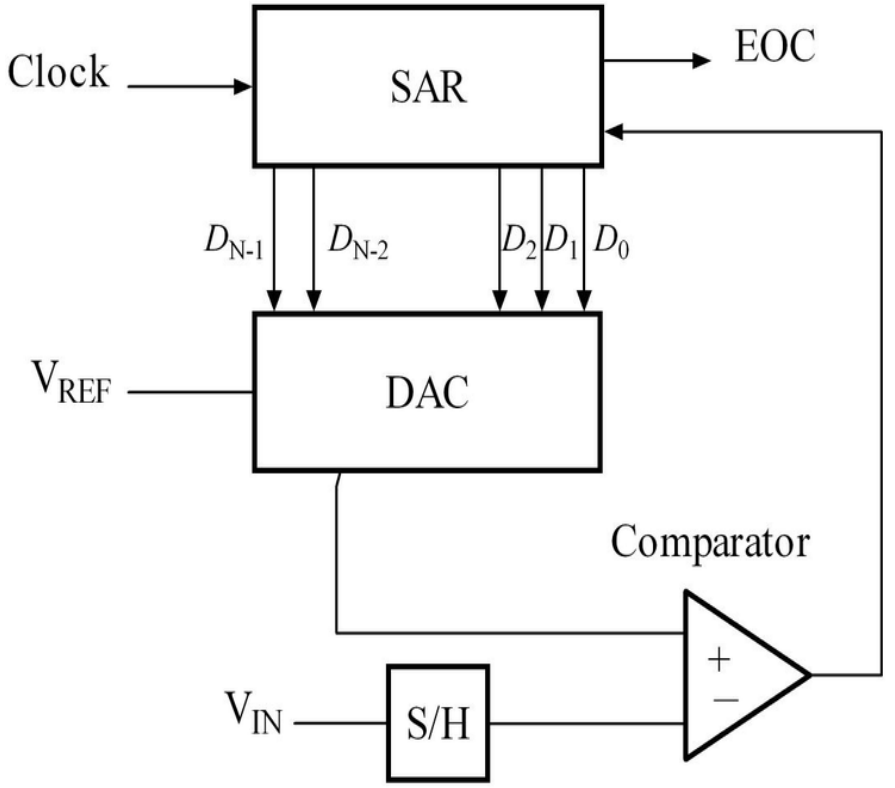


Image from http://www.allaboutcircuits.com/vol_4/chpt_13/5.html

Successive Approximation

- Performs a binary search of the range with a DAC and comparator
  - Set first bit to 1; generate value on DAC; compare.
  - If $V_{in} < V_{dac}$, reset to 0; else, keep bit as 1
  - Repeat for next bit
  - Turn on EOC when match achieved
- Slow, but relatively simple and protected from errors



Arduino

- AnalogRead()
  - Successive Approximation ADC
  - 6 channels, 10-bit resolution, 10kHz
  - Built-in AREF
    - analogReference(type)
    - Type is DEFAULT=5V, INTERNAL=1.1V, EXTERNAL
- AnalogWrite()
  - PWM
  - 490Hz carrier

A final Assignment

- Using the parts and skills from the unit, and anything else you might want to include, build something nifty.
- We have a pool of extra parts available:
  - 7-Segment LED Display
  - RGB (tricolor) LED (Color mixing)
  - Piezo Buzzer (Tiny, tinny, directly drivable speaker)
  - Temperature Sensor

Reading Datasheets

- Key skill, developed by practice
- Formats NOT well standardized, even for similar parts
- Largely about filtering for what you want
- Kind of an art
- Final assignment parts (and some other bits and pieces) as examples

Acknowledgments

- Unless otherwise marked, images used in this presentation are from the Wikimedia Commons