Internet of Things

Module 2: Devices

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

You Need to Build…

- Flyover drone to analyze crops
- Generator defect detector
- Intelligent smartwatch
- Shopper gaze monitor

How do you build IoT things?

- IoT is a synthesis of hardware and software
  - Software implementation plus a collection of attached circuits
- Not enough to understand Arduino programming
  - Good designers know the system they are building on
  - Underlying algorithms, protocols, architectures
  - External interfaces, APIs, components

Roadmap

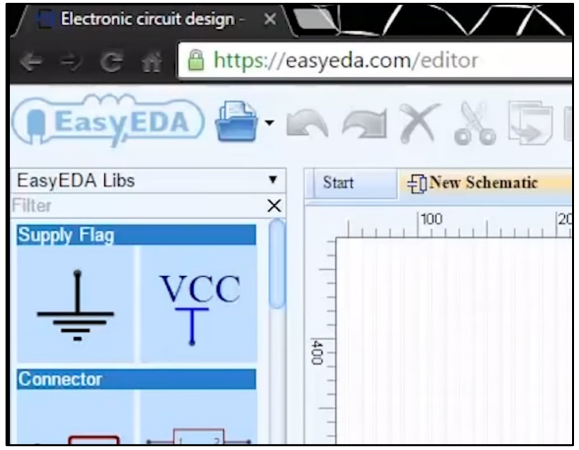
1. IoT Hardware
   - Components, sensors/actuators, reference circuits, integrated circuits
   - Memory, microcontrollers, storage
2. IoT Platforms
   - Arduino, Raspberry PI, Amazon FreeRTOS, VxWorks, RIOT
   - Designs, architectures, and tradeoffs
3. Deep Dive: Arduino Programming
   - Development workflow, language, API, coding examples
4. Case Studies
   - Deployment scenarios: cars, aircraft, buildings, retail

IoT Hardware

We Are Going through a Hardware Revolution

- Low-cost, componentized hardware enables rapid prototyping without advanced electrical engineering knowledge

Componentized Circuits



Cloud-Based Fabrication

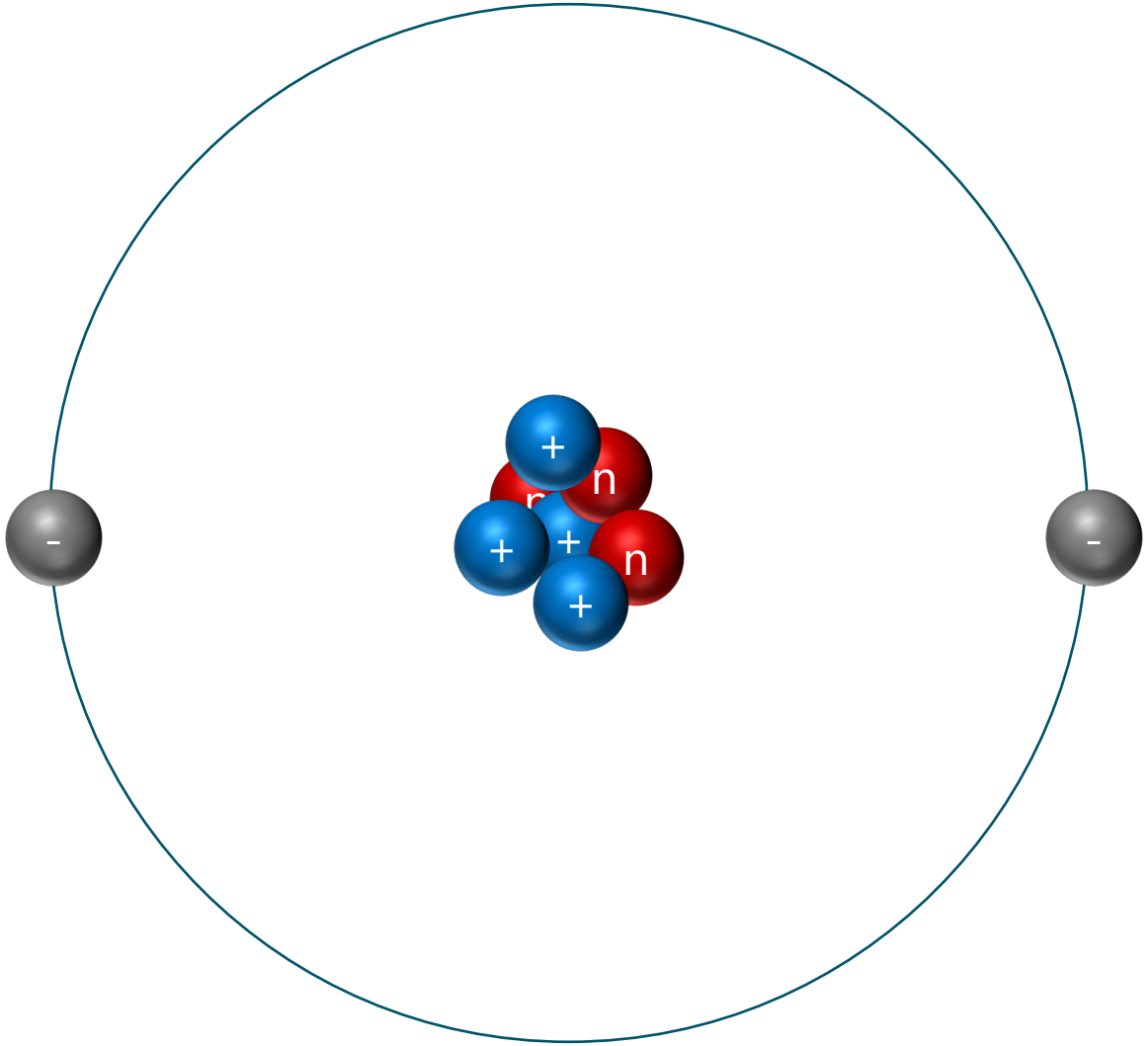
Open-Source Schematics

- This lecture: What You Need to Know to Build IoT Hardware Systems
  - Electricity Fundamentals, IoT Circuits, Microcontroller Platforms


REV:C01

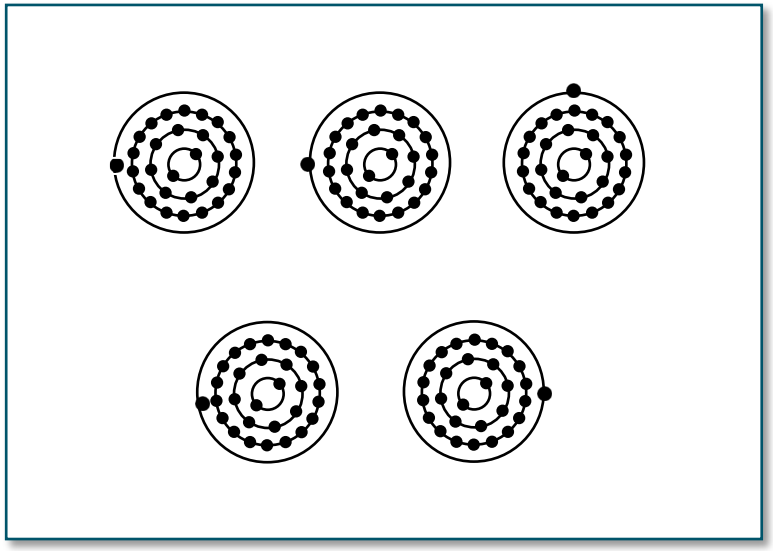
Background: Atoms

- Made of protons, neutrons, and electrons
- In materials, nucleus held in fixed position
- Some electrons are also fixed
- But some electrons may be free to "move"

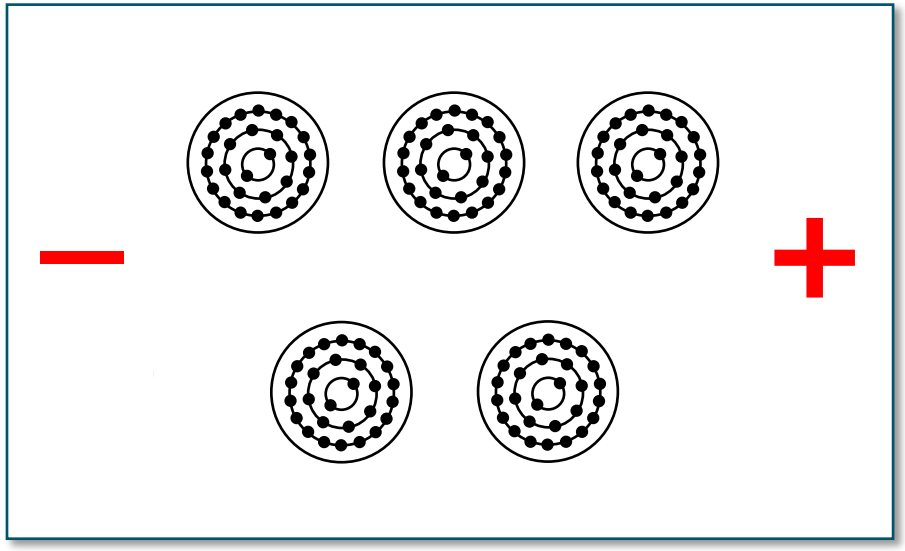


Background: Electrical Current

- Usually free electrons hop around randomly
- However, outside forces can encourage them to flow in a particular direction
  - Magnetic field, charge differential → this is called **current**
  - We can vary properties of current to transmit information (via waves, like dominos, as electron **drift velocities** are very slow)

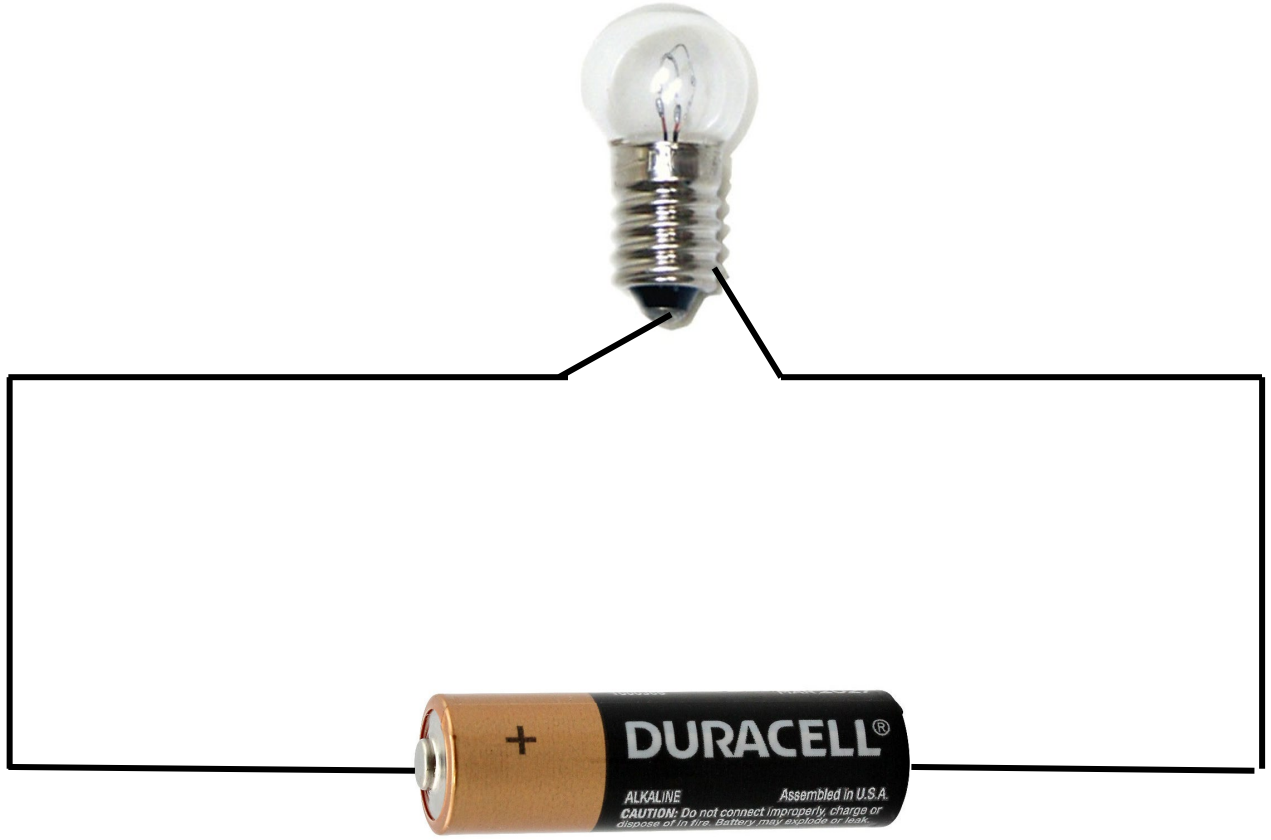


No charge differential



Charge differential

Making an Electrical “Circuit”



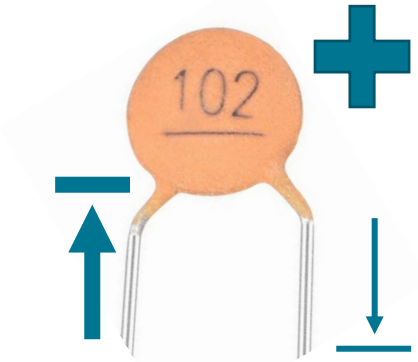
Electrical Current Can Be Manipulated

- It is possible to take current and change it

Example: Resistor

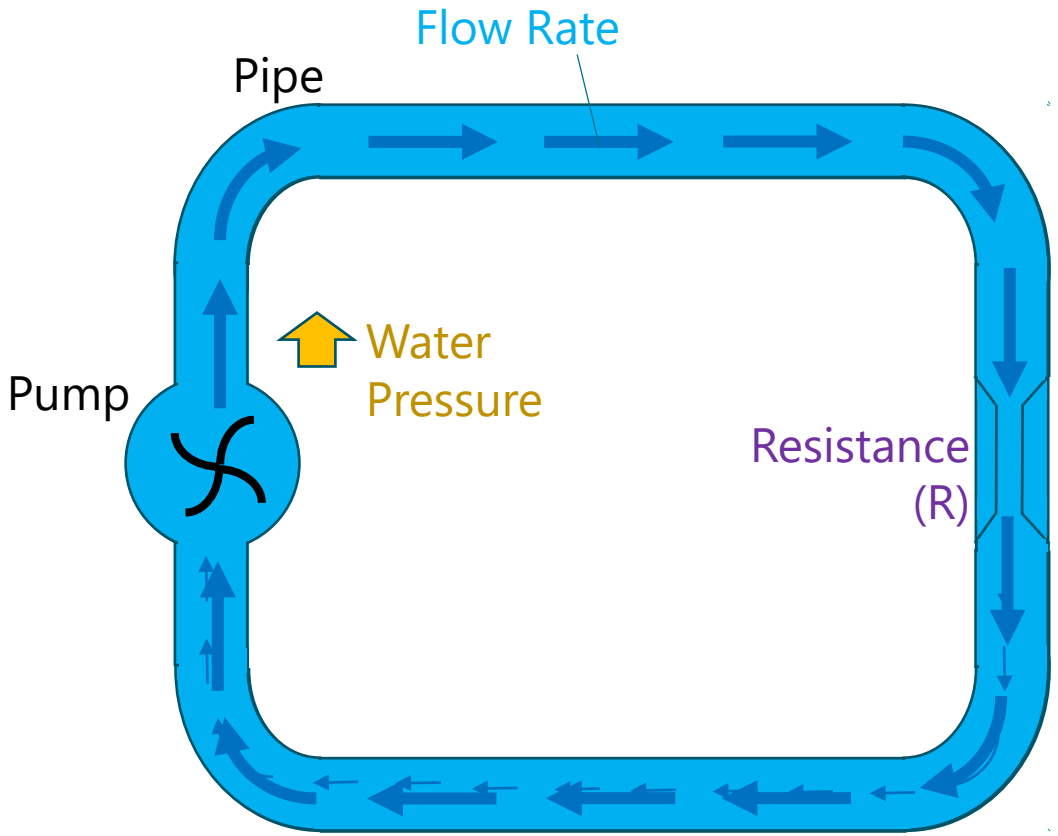
Reduces flow rate of electricity

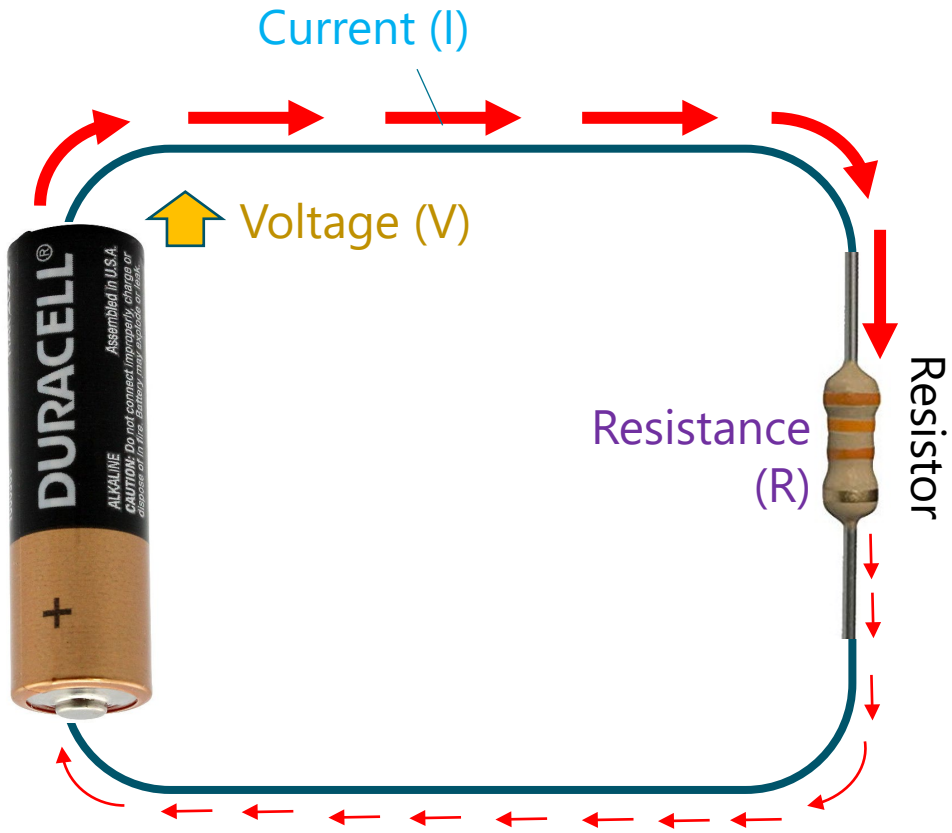
Example: Capacitor



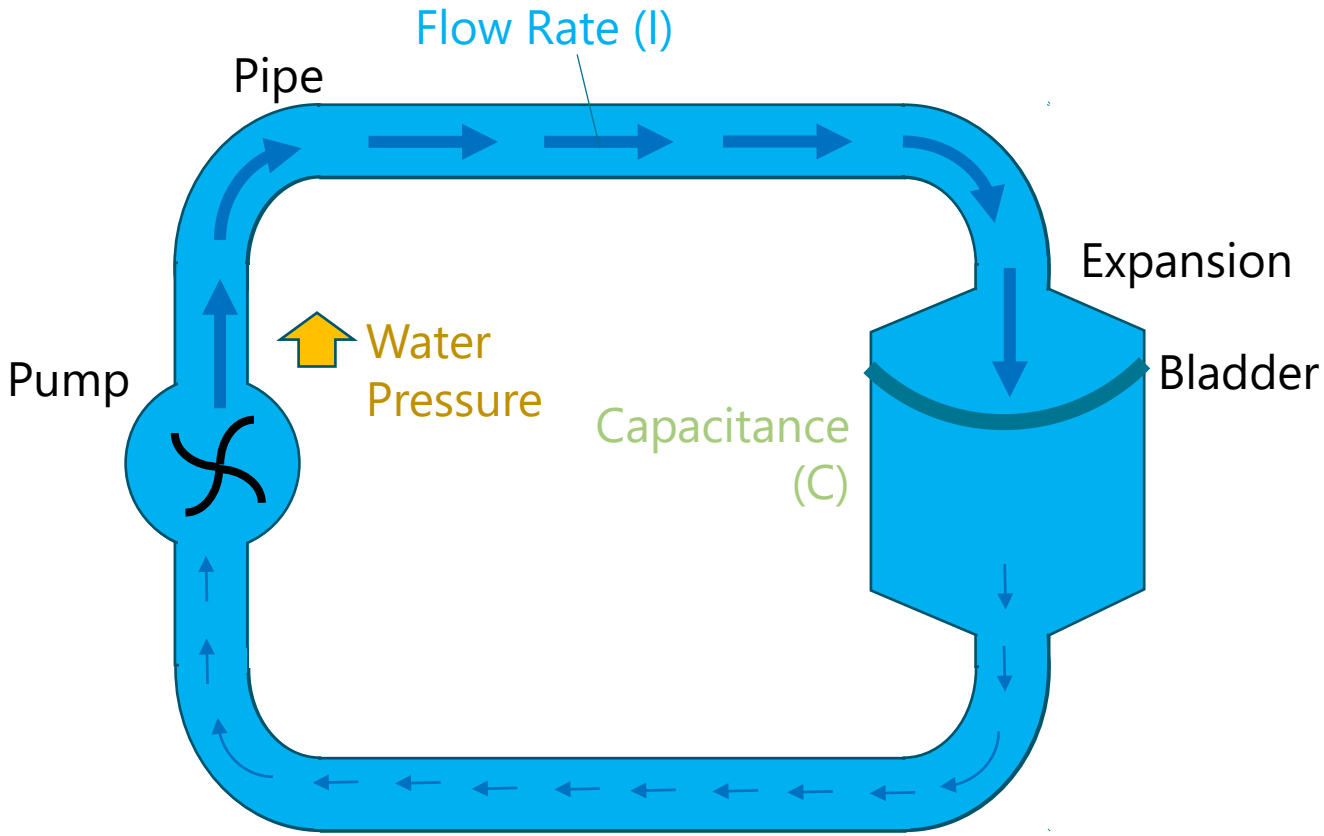
Stores electrical “charge”

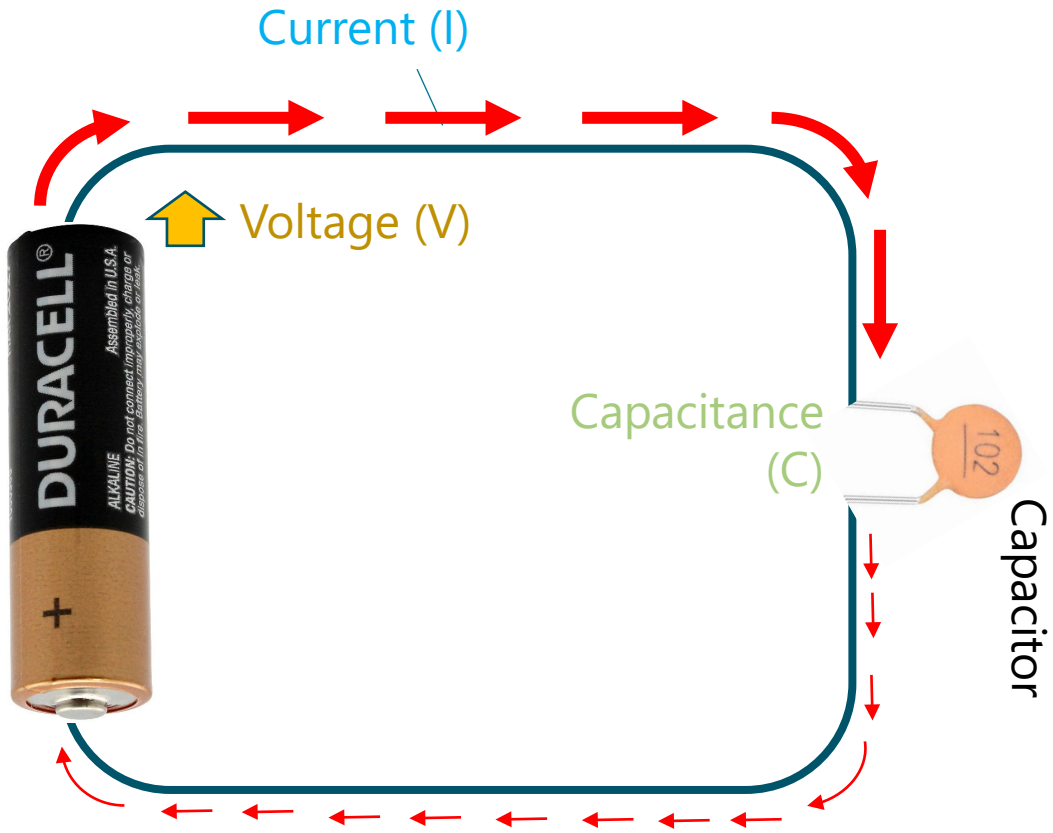
Flowing Electricity Is Like Flowing Water





Water Analogy: Capacitance





Electrical Components

Resistor

Capacitor

Potentiometer

Transistor

Relay

Diode

Photocell

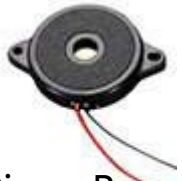
Light-Emitting Diode (LED)

Crystal Oscillator

DC Fan

Speaker

Motor



Piezo Buzzer

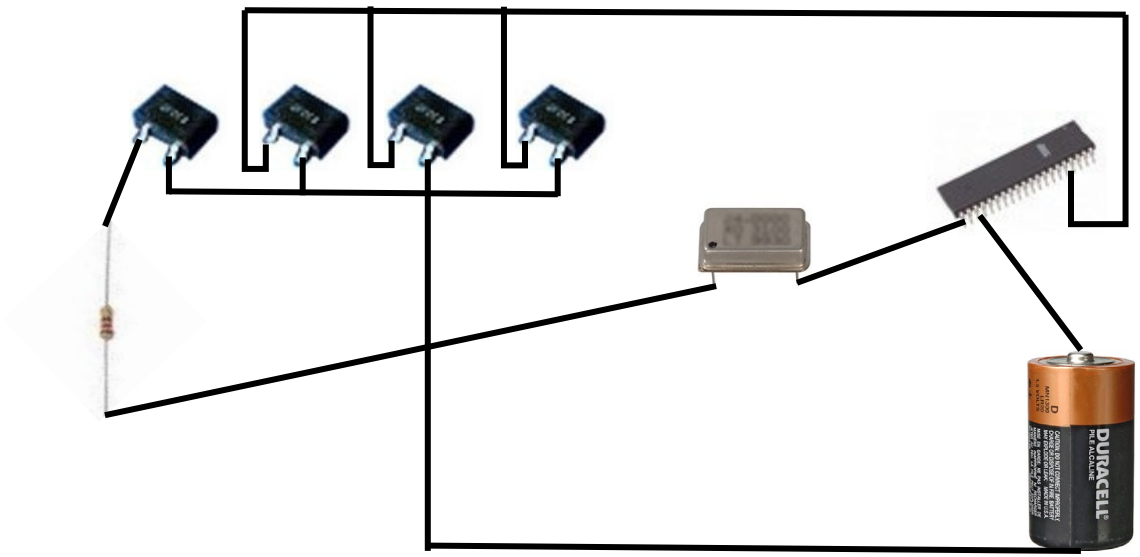
7-Segment Display

Toggle Switch

AA Battery

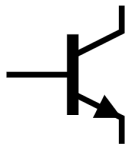
Electrical Ground

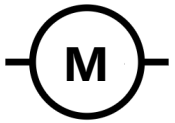
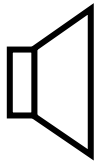
How to draw a circuit?



- Need to remember our designs, show to others
- Need to clearly describe what each component is, properties of component, which pins are connected, etc.

Electronic Symbols

Name	Picture	Symbol
Resistor		
Capacitor		
Diode		
Transistor		
Oscillator		

Name	Picture	Symbol
Electric Motor		
Speaker		
Voltage Source		
Switch		
Electrical Ground		

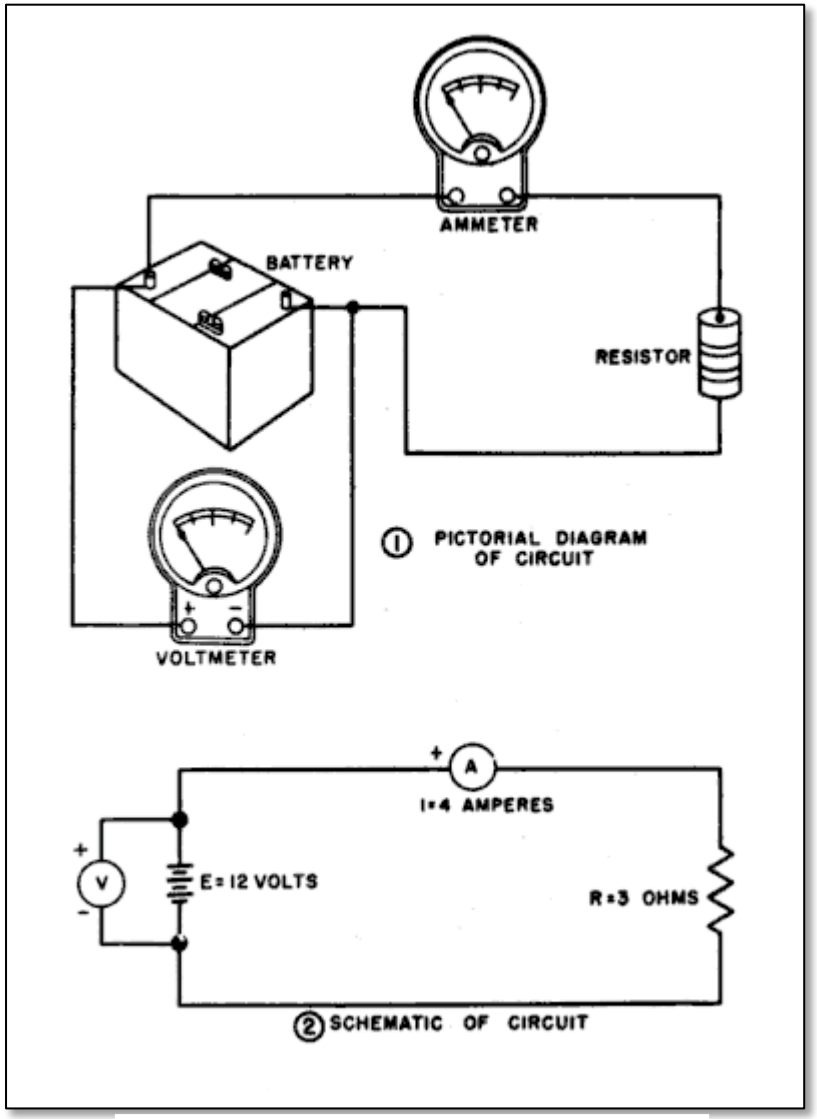


Diagram of a basic Circuit

How to decide which components to use?

1. Think about what you want to build
2. Design a circuit that achieves that objective
3. List components that are required by the circuit and their critical parameters

How to decide which components to use?

4. Choose components with desirable properties
   - Voltage range, tolerance, resistance value, temperature coefficient, wattage, load efficiency, etc.
   - Components that reduce circuit complexity (e.g., built-in ESD protection)
   - Environmental parameters (humidity/pressure/vibration/temperature resilience)

How to decide which components to use?

5. Choose components that fit in the circuit well
   - Manufacturer with good documentation/support/tools, long-term availability, lead-time, cost, fall-back options
   - Mechanical parameters (dimensions, weight, etc.)

6. Prototype before you start production

Use Cases

Suppose you want to build…
something that lights up

- Indicator/status signals
- User interfaces and displays
- Illumination sources
- Light effects

Components that make light

LED (green)

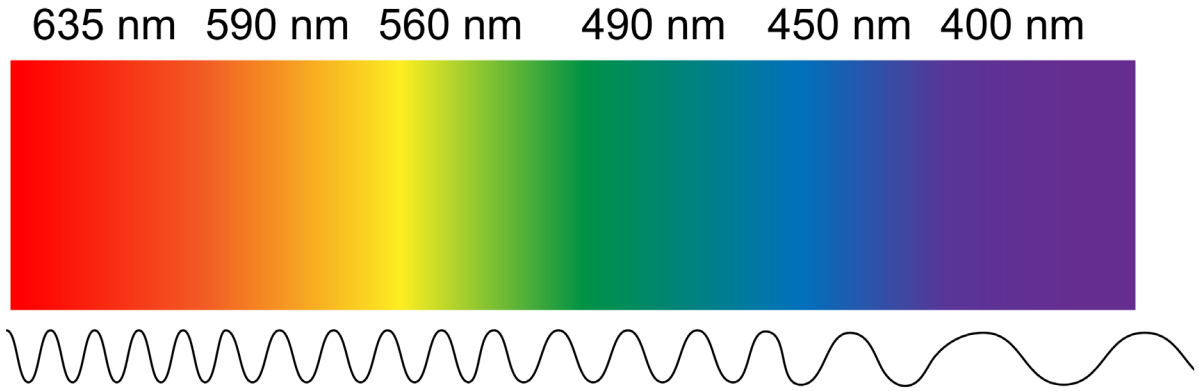
TFT Display

Optical Audio Transmitter

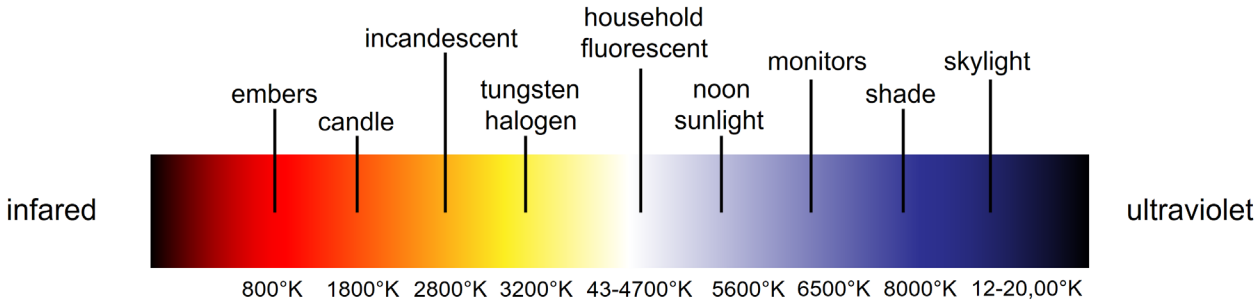
- What they do: Create light
- Key metrics:
  - **Brightness:** how much light is generated (*mcd/nits*)
  - **Viewing angle:** over what angle most of the light is visible (*degrees*)
  - **Color:** color of emitted light (*nm/kelvin*)
  - **Voltage:** voltage needed to "turn on" light (*volts*)
  - **Current:** current needed to "turn on" light (*mA*)

How color is measured

- **Option 1: Wavelength**
  - Color of light as function of electromagnetic wavelength/frequency



- **Option 2: Black body radiation**
  - All objects radiate light based on their temperature
  - Color of light as a function of "blackbody" temperature



More components that make light

Colored LEDs


LED (green)


LED (yellow)

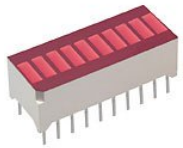

LED (blue)


Red/Green Bi-color LED


Infrared LED


Infrared Phototransistor

Arranged LEDs


Bar Graph Array


7-Segment Display


4-Digit 7-Segment Display

Light Matrices


Flexible RGB LED Matrix


TFT Display


LED Dot Matrix Display

Shaped LEDs

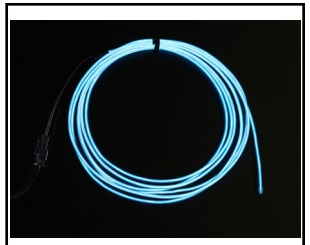

LED (green, flush mount)

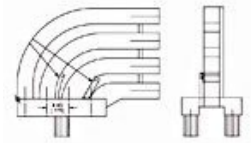

Panel Mount LED

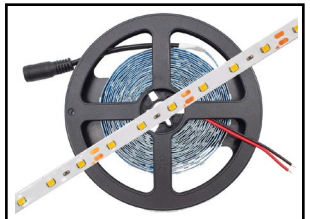

LED Sequin


Backlight Module

Directed Light


Electro-luminescent Wire


Light Pipe


LED Strip

Other Light Tech


Incandescent Bulb


Laser Diode

LCD Display

Light Signaling


Photointerruptor


Optocoupler


Light to Frequency Converter


Optical Audio Transmitter

Suppose you want to build…
something that needs electricity

- Operate for long periods of time without management
- Move around without being plugged in
- Collect energy from environment
- Supply electricity to other devices
- Change voltage from one level to another
- Protect against voltage spikes, electronic noise, and static electricity

Electrical power sources

Batteries

Coin cell battery

Coin cell battery (with leads)

AA battery

AA battery holder

C battery

9V battery snap

9V battery

9V battery holder

USB battery

Plugging in to the Grid

9VDC 2.1mm wall adapter

USB to TTL serial cable

2.1mm connector jack

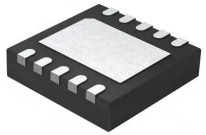
Variable power supply

Energy Harvesting

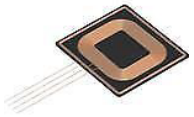
Solar cell [light]

Solar cell [light] (thin film)

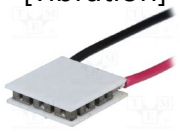
Motor/Generator [motion]



Piezoelectric generator [vibration]



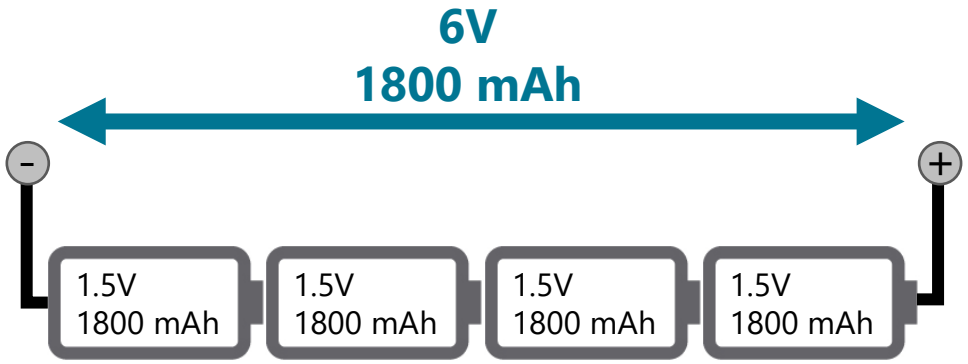
Antenna/ Rectenna [radio waves]



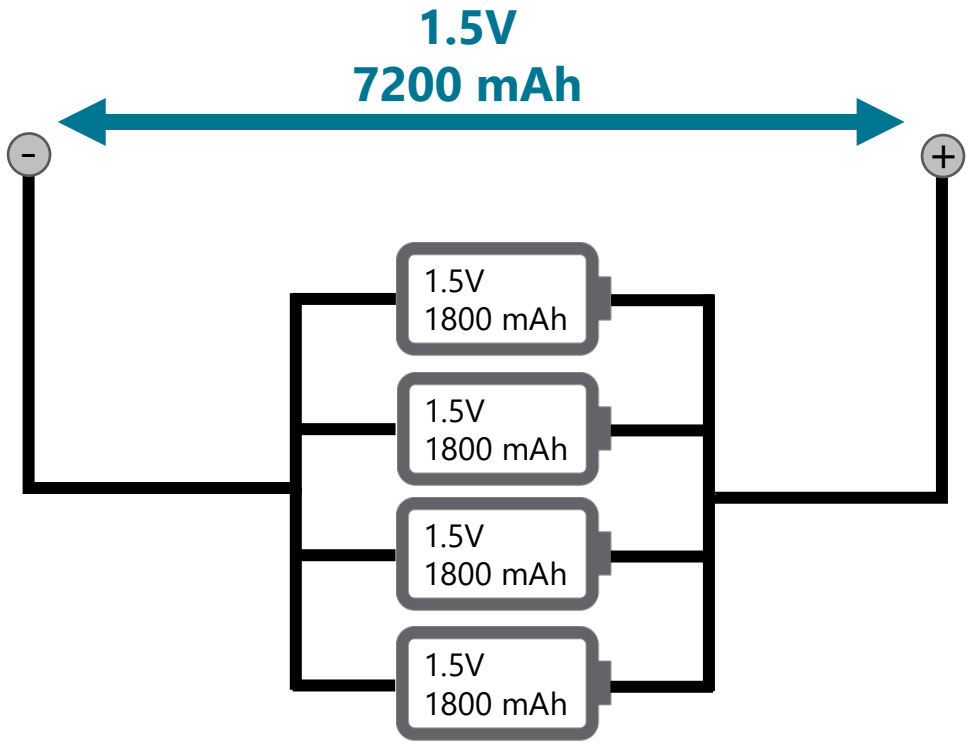
Thermoelectric generator [heat]

Hand crank

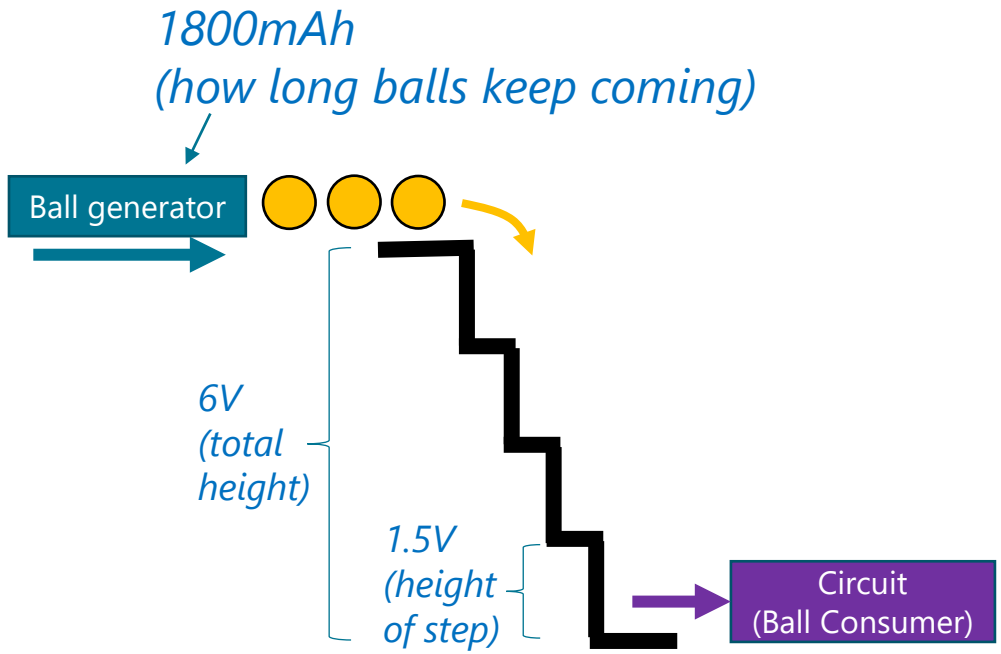
Power: Combining Current Sources

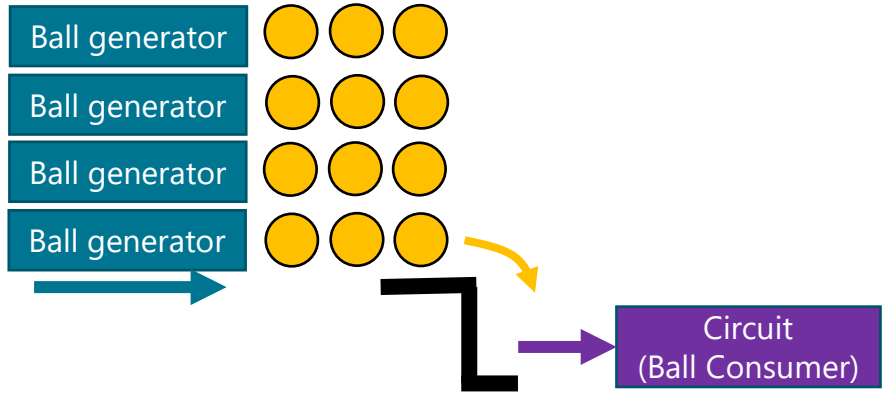


Connected in Series

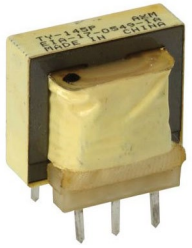


Connected in Parallel

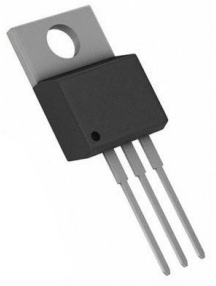




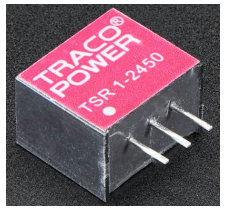
Components that change voltage levels



Transformer
(steps up/down AC voltage)



Voltage regulator
(keeps voltage within range)

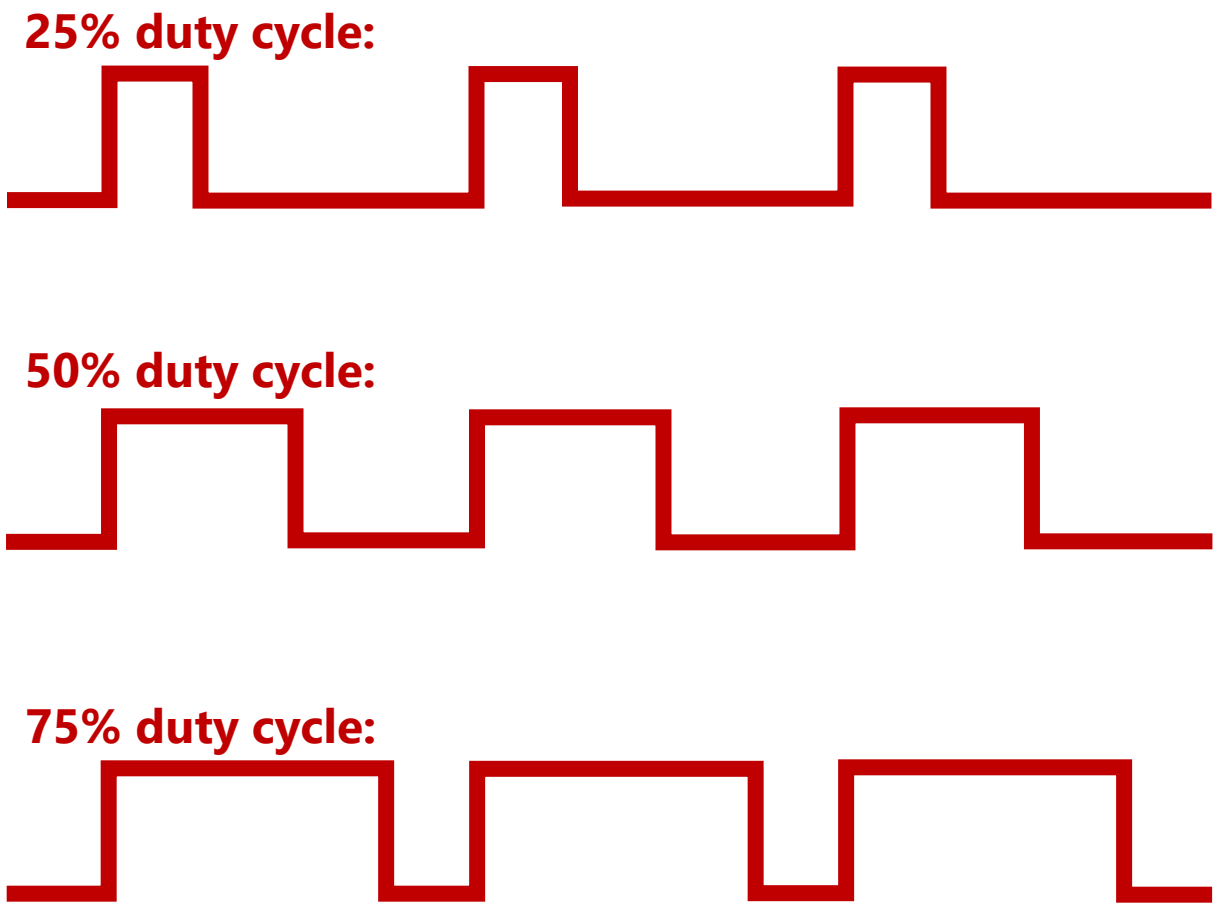


Buck converter
(steps down DC voltage)

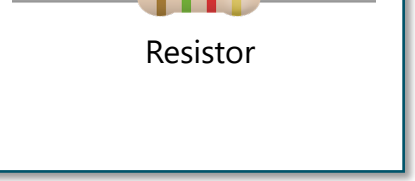
Boost converter
(steps up DC voltage)

Power: Pulse-Width Modulation (PWM)

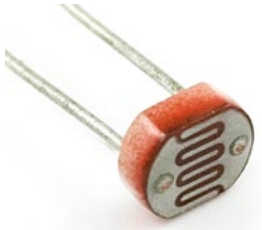
- Suppose you want to increase/ decrease power to a component
  - E.g., vary brightness of LED, speed of motor
- **PWM**: controls power by rapidly turning on/off
  - Cheaper, more efficient electronics than varying voltage



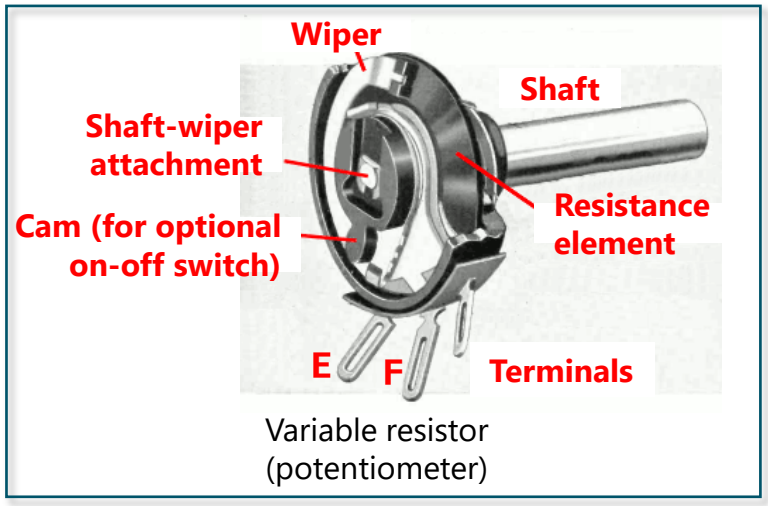
Resistors and Potentiometers



Resistor



Photoresistor



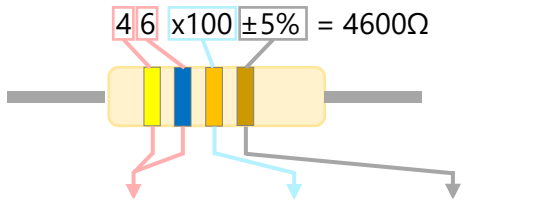
Variable resistor (potentiometer)

What they do: resist the flow of current

Useful for:

- Protecting components from getting too much current
- Increasing and decreasing voltage
- Measuring things (e.g., light intensity)

Resistors and Potentiometers

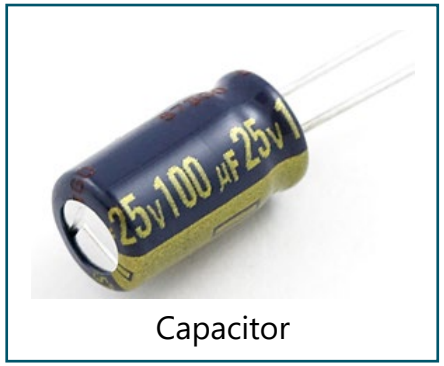


Color	Digit	Multiplier	Tolerance
Black	0	x1	
Brown	1	x10	±1%
Red	2	x100	±2%
Orange	3	x1K	
Yellow	4	x10K	
Green	5	x100K	±0.5%
Blue	6	x1M	±0.25%
Violet	7	x10M	±0.1%
Grey	8	x100M	±0.05%
White	9	x1G	
Gold		x0.1	±5%
Silver		x0.01	±10%

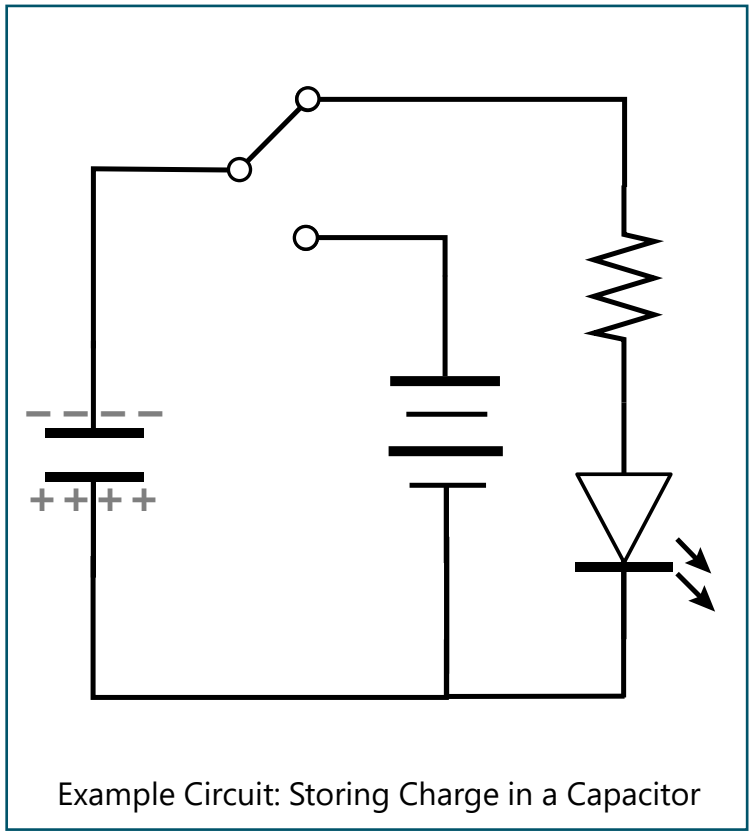
Key metrics:

- **Resistance:** how much they resist current (Ohms)
- **Tolerance:** how accurate their resistance rating is (%)
- **Power rating:** how much current they can handle (Watts)
- **Maximum voltage:** how much voltage difference across their input/output they can handle

Capacitors



Capacitor

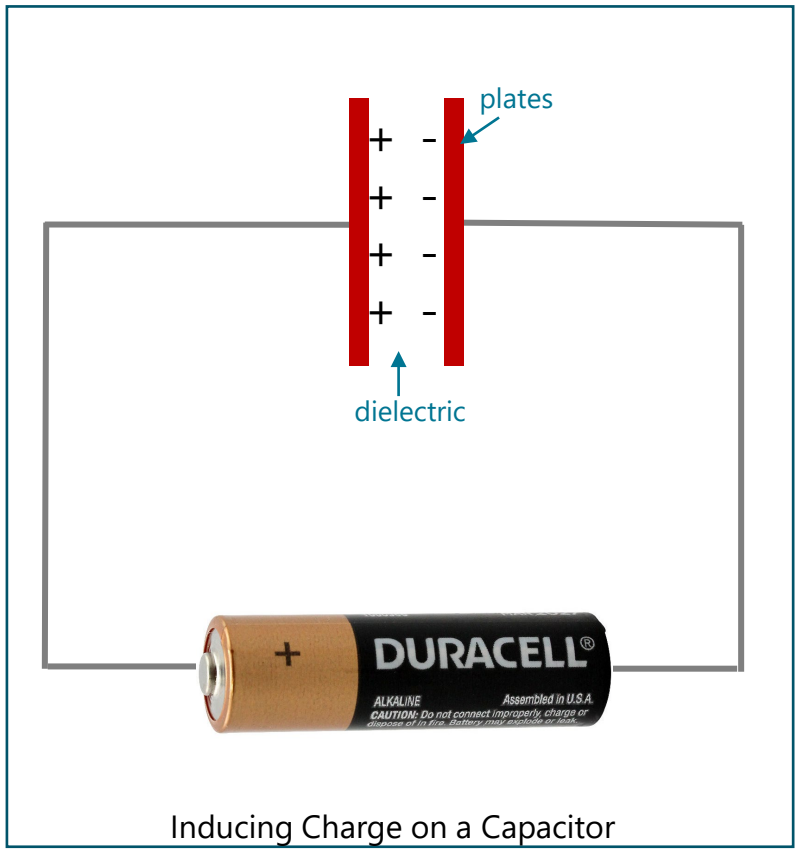

Example Circuit: Storing Charge in a Capacitor

What they do: store charge

Useful for:

- Temporarily holding some charge (like a tiny battery)
- Smoothing out voltage spikes
- Measurement (humidity, pressure, touch sensors)

Capacitors



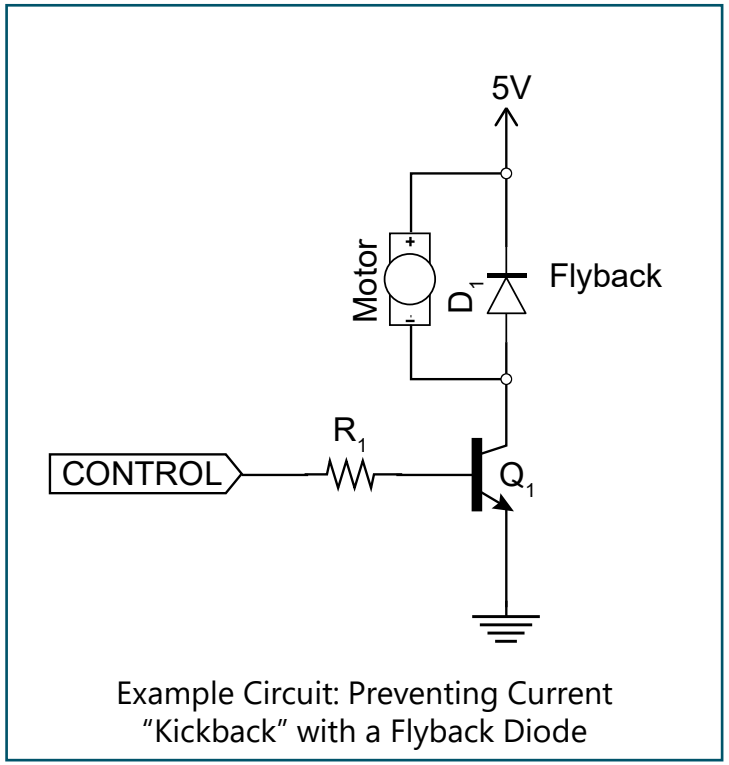
Inducing Charge on a Capacitor

Key metrics:

- **Capacitance:** how much charge it can store (farads)
- **Tolerance:** how accurate their capacitance rating is (%)
- **Maximum voltage:** how much voltage they can handle (before they short out or burn up)
- **Leakage current:** amount of current that leaks through dielectric
  - No "current rating" – you can't have a sustained current through a capacitor aside from leakage
- **Equivalent series resistance (ESR):** capacitors aren't perfect and have tiny amount of resistance (usually less than 0.01Ω)

Diode

Diode



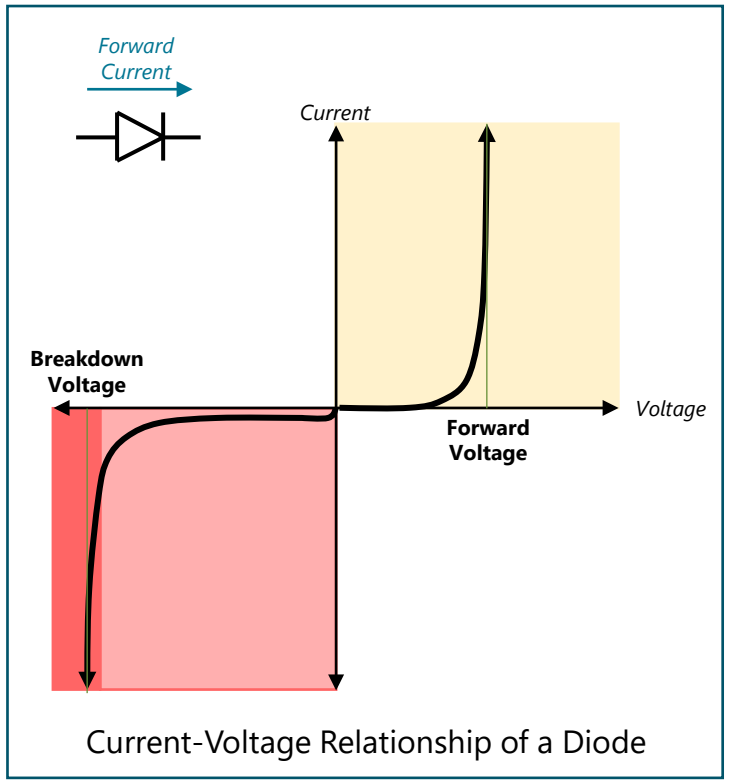
Example Circuit: Preventing Current "Kickback" with a Flyback Diode

What it does: allows current to go in forward direction, but not reverse direction

Useful for:

- Reverse current protection
- "Rectifying" signals (pulling out positive parts)
  - Converting AC to DC
- Measurement (temperature, radiation)

Diode



Current-Voltage Relationship of a Diode

Key metrics:

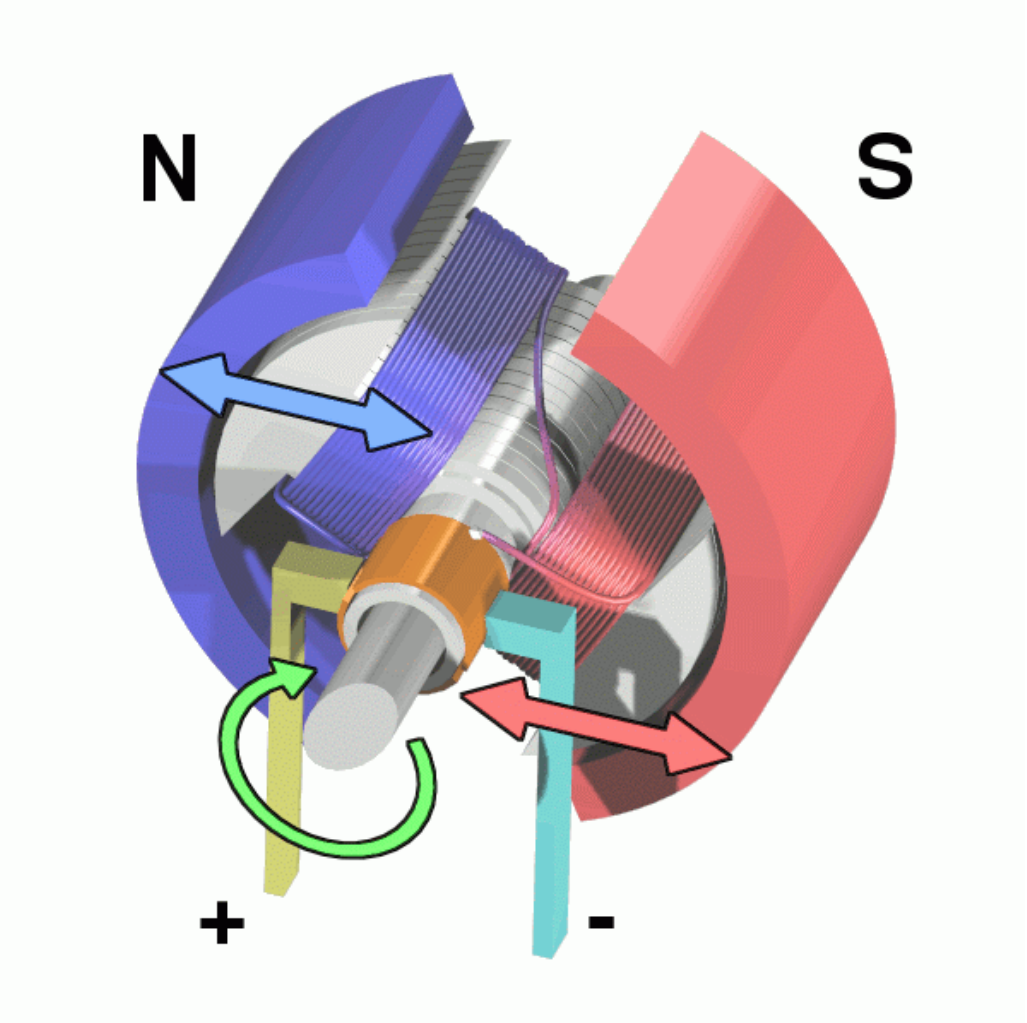
- **Maximum forward current:** how much current can go through in the forward direction (amps)
- **Maximum reverse voltage (breakdown voltage):** how much voltage can be withstood in reverse direction
- **Maximum forward voltage:** voltage difference between input/output when current going through forward direction (ideally should be zero, no resistance to current)

Suppose you want to build… *something that moves*

- Position control surfaces, antennas, conveyors, etc.
- Operating grippers, levers, dispensers, belts, cutters, etc.
- With varying speed, preciseness, torque/power, etc.

Electric Motors

- Converts electricity into motion
- Main idea:
  - Get two big magnets, one north and one south
  - Run electricity through a curved wire
    - Alternatively pushes/pulls against magnets
    - Rotates a mechanical shaft



Common Types of Electric Motors

More Types of Electric Motors

Generator
(run brushed motor via external force to generate current)

Servo
(like stepper, but with position feedback mechanism to control position – more expensive, higher torque/speed)

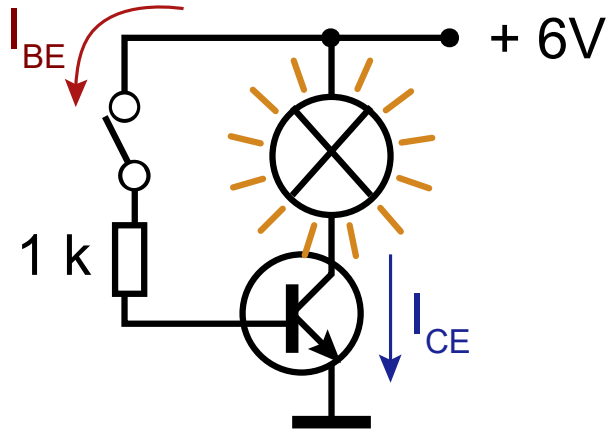
Torque
Can operate indefinitely when stalled, applying constant torque (force feedback games, positioning)

Linear
"Unrolled" motor, produces straight-line torque (maglev trains, pen plotters)

Vibration
Unbalanced weight attached to pole (phones, pagers, toys)

Transistors/Relays

Transistor



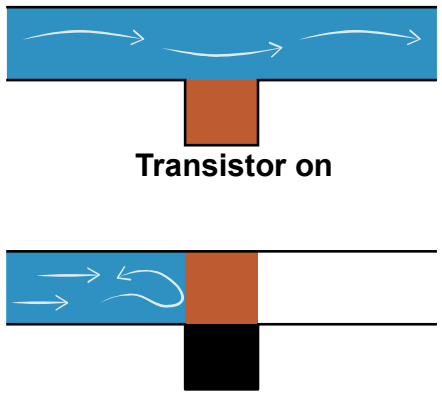
Example Circuit: Controlling a Light with an NPN Transistor

What it does: acts like a switch; when voltage applied to one wire, forms connection between other two wires

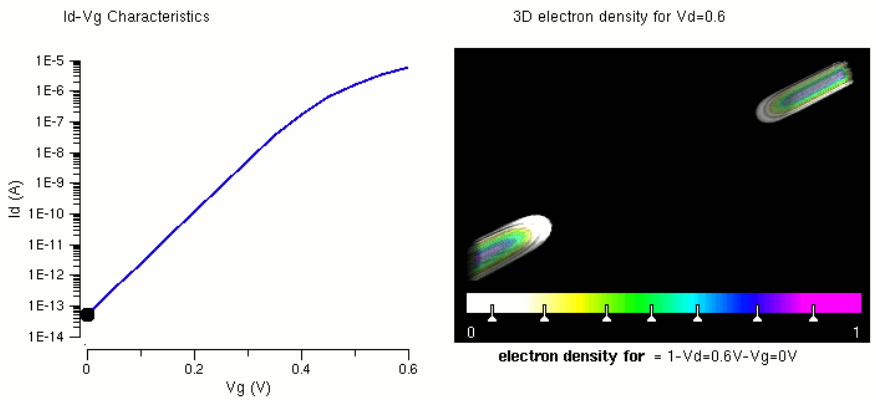
Useful for:

- Acting like a switch
- Amplifying a signal

Transistors/Relays



Water Analogy: Transistors/Relays Stop/Start Flow



Formation of Channel to Make a Connection in a Transistor

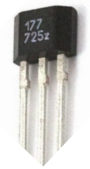
Key metrics (transistors):

- **Type:** does applied current turn on connection (NPN) or turn off connection (PNP)
- **Maximum reverse voltage (breakdown voltage):** how much voltage can be withstood in reverse direction
- **Maximum forward voltage:** voltage difference between input/output when current going through forward direction (ideally should be zero, no resistance to current)

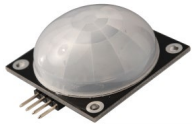
Suppose you want to build… *something that observes*

- Listens, sees, measures
- Sound, light, temperature, humidity, magnetic fields, rotation, acceleration, etc.

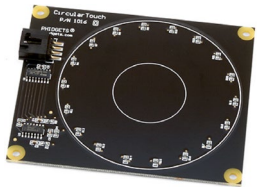
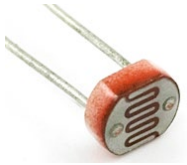
Sensors

Picture	Sensor name	What it detects	Example Applications
	**Color Sensor**	Detects and measures range of visible colors	Test strip reading Sorting by color Ambient light sensing and calibration Color matching
	**Hall Effect Sensor (magnetism)**	Varies output voltage in response to a magnetic field. Can pair with magnet for some applications (e.g, positioning)	Measuring wheel/rotation speed Proximity sensing Positioning/alignment Current sensing
	**Microphone (sound)**	Converts sound into electrical signal	Audio recording Speech recognition/Wake phrase Echolocation
	**Accelerometer**	Measures physical acceleration experienced by an object	Human activities: walking/running/dancing Fall/drop detection, Occupancy detection Detecting which way is down Building/motor health monitoring

Sensors

Picture	Sensor name	What it detects	Example Applications
	**Gyroscope/ Tilt Sensor**	Measures rotation (pitch/yaw/roll)	Human activities: walking/running/dancing 3D motion control Robotics
	**Gas Sensor**	Detection of gases. Variants measure Carbon Monoxide, Methane, Alcohol/Benzine, Propane, etc.	Safety devices Environmental monitoring
	**Passive Infrared Sensor**	Converts infrared light into electrical signal	Motion detection, entry alarm Remote control/communication Maze navigation, boundary sensing
	**Proximity/ Distance Sensor**	Measures distance to moving or stationary objects. Variants: ultrasonic, laser, infrared.	Parking assistant systems Speed measurement Robotic navigation Interactive displays

Sensors

Picture	Sensor name	What it detects	Example Applications
	**Humidity Sensor (hygrometer)**	Measures humidity/water content	Soil monitoring Weather/microclimate monitoring
	**Touch Sensor**	Detects and measures anything that is conductive or has a dielectric different from air	Touchscreens Fingerprint-based identification
	**Photoresistor**	Light intensity	Sleep mode activation Smart building light controls
	**Load Cell**	Measures force/weight	Presence/arrival detection Weight measurement Impact measurement Mechanical strain detection

Sensors

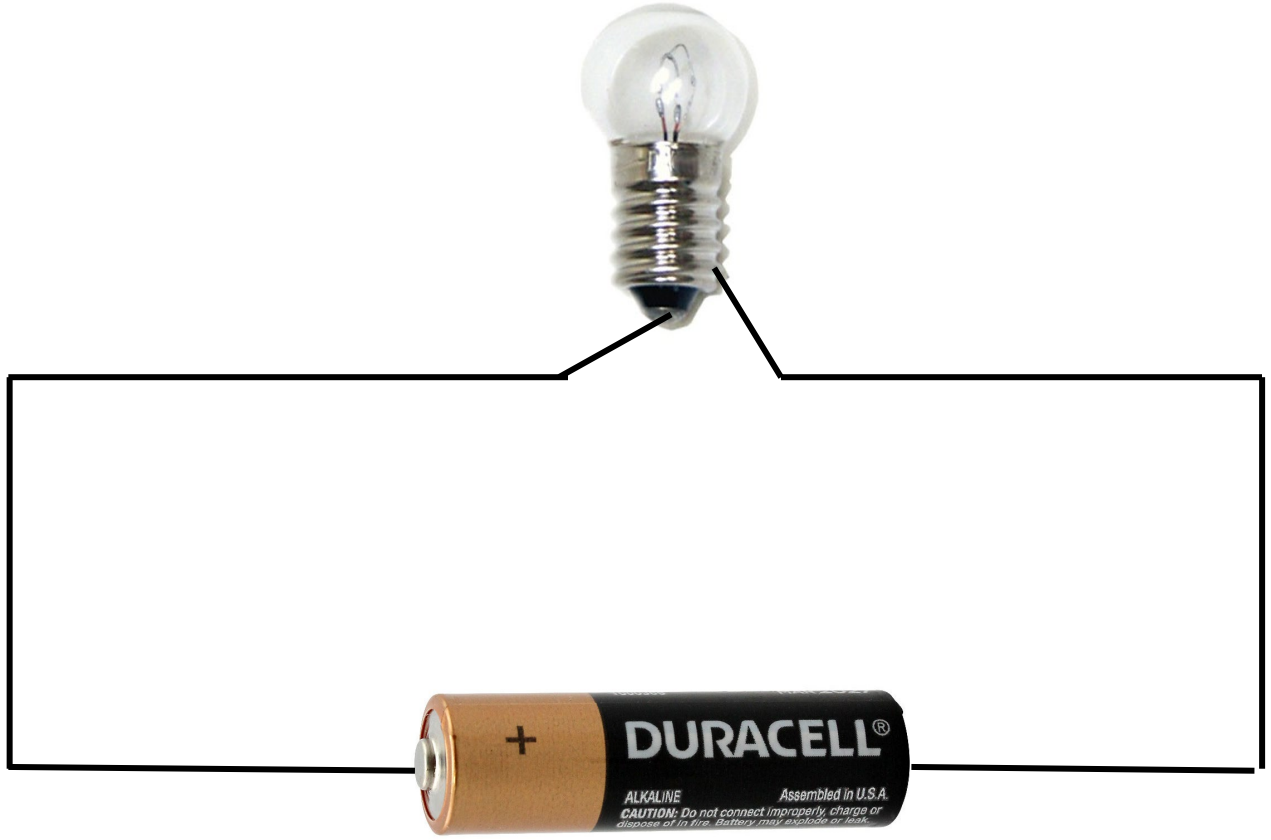
Picture	Sensor name	What it detects	Example Applications
	**Flow Sensor**	Measure flow rate (and thereby volume) of gas or fluids	Industrial process monitoring Farming (eg. water/pesticide applications) Water/gas meters “Green building” resource usage
	**Anemometer**	Measure wind speed. Can pair with vane to measure wind direction. Also see pitot tube.	Weather/microclimate monitoring Structural safety monitoring (bridge, cranes etc)
	**Temperature Sensor**	Measures temperature	Plant health monitoring Industrial machine health monitoring Circuit overheat protection
	**Camera**	Captures video or pictures. Can be paired with computer vision to recognize objects, faces, track motion, etc.	Customer identification Crop Analysis 3D Object recognition “Go home” function

Emerging Sensors

Recently emerging sensor types make IoT more powerful

Picture	Sensor name	What it detects	Example Applications
	**Electronic Nose**	Measures chemical concentrations behind scents/smells	Early detection of health problems (eg COPD) Counter-terrorism Computer smell
	**Tactile Sensor**	Determining texture, stiffness, coefficient of friction, thermal conductivity	Robotics Object recognition
	**Blood Gas/Sugar Sensor**	Measures composition of blood, including amount of arterial gases (eg oxygen, carbon dioxide), pH, sugar, etc.	Smart clothes Home health checkup Early asthma detection
	**Brainwave Sensors**	Measures electrical activity of brain. Can pinpoint activity to 3-D location inside brain.	Mental control over objects Diagnose brain faults and mental disorders Understanding user intent Reading innermost thoughts and desires

Making a Circuit



- Can put components together to build circuits that do things

How to Build Circuits

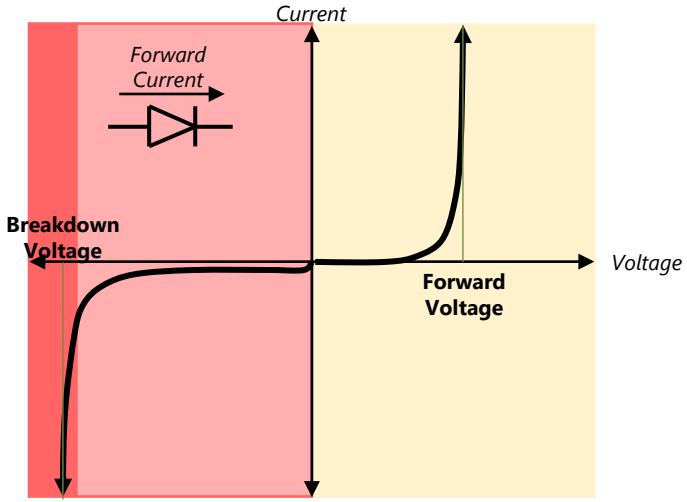
- Like algorithms, common circuits are known
  - Can learn the common ones
- You don’t need to memorize a lot of circuits to do IoT
  - More complicated circuits tend to be componentized anyway
  - Can buy pre-created chips and PCBs
  - Can also look up advanced ones when needed
- Next, we will go through some that are helpful to know for IoT projects
  - Build intuition, use these when you need them

Six Useful Circuits to Know About

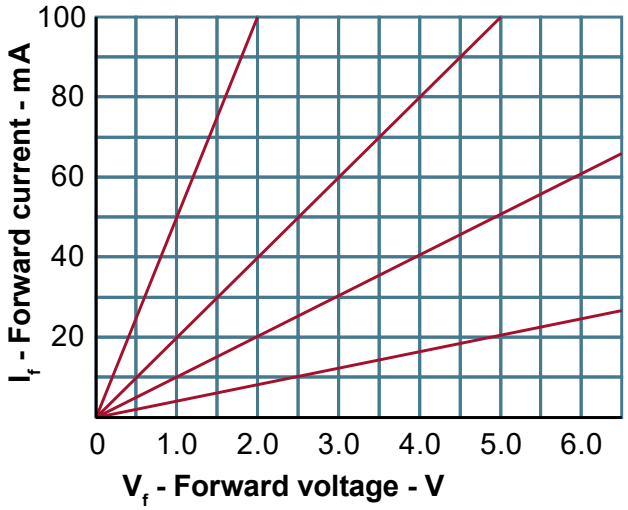
Widely used, demonstrate useful concepts:

1. Current-Limiting Resistor
2. Voltage Shifter
3. Filter Capacitor
4. Pull-Down Resistor
5. Signal Amplifier
6. Transistor Switch

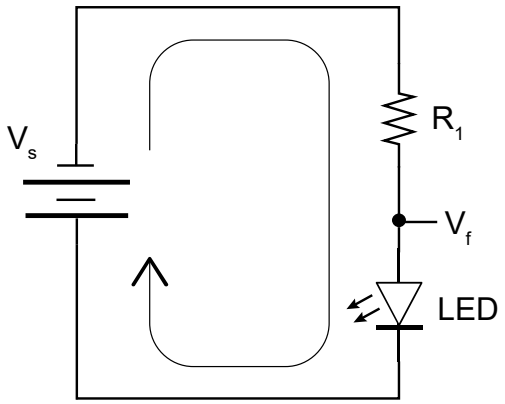
1. Protecting against Current Overload (Resistors)



LEDs have non-linear resistance



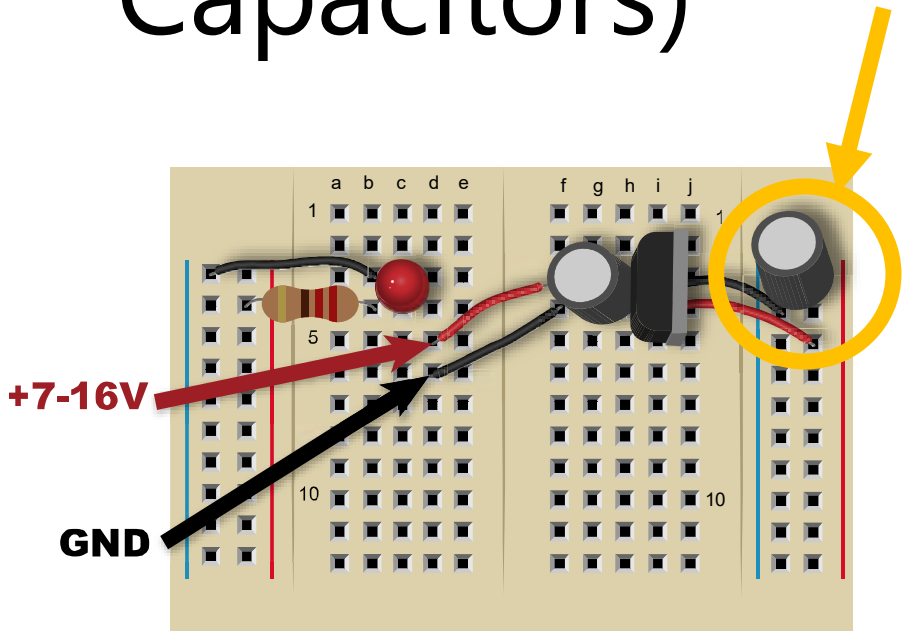
Resistors have linear resistance

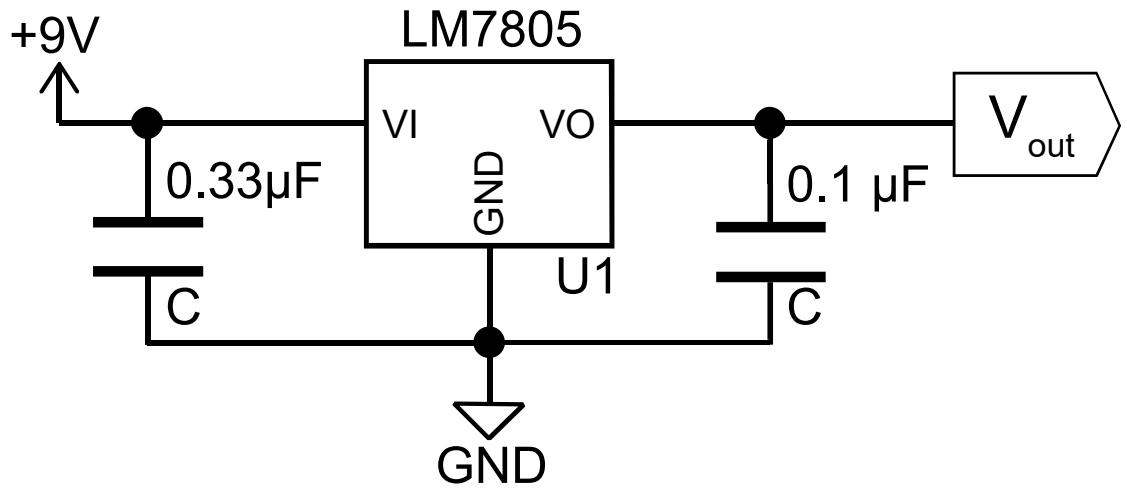


Current-limiting resistor

- Components can accidentally draw too much current
  - Resistors: current increases linearly with voltage
  - LEDs: if you overshoot voltage a little bit, suddenly LOTS of current
- Current-limiting resistors protect your components
  - Prevent too much current from going though your circuit

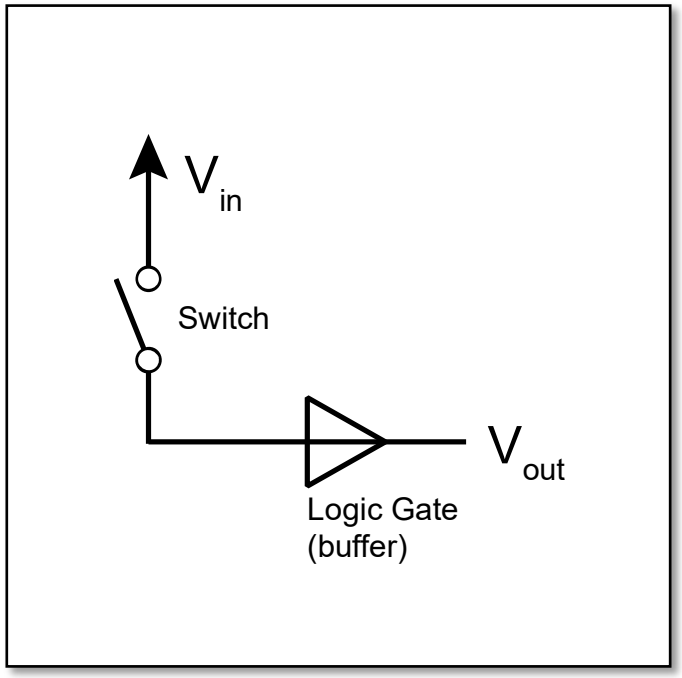
2. Protecting against Spikes (Filter Capacitors)



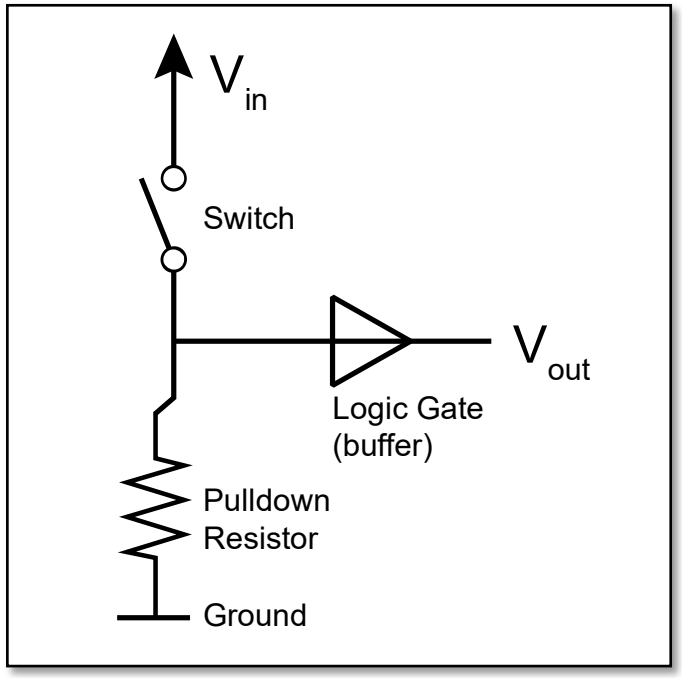


- Place capacitor between terminal and ground to smooth noise at the terminal
- Choose capacitor value based on noise you expect from the system
  - Rule of thumb: 0.1µF-10µF is pretty safe if you're just doing low voltage logic

3. Eliminating Noise from Disconnection (Pull-Down Resistors)

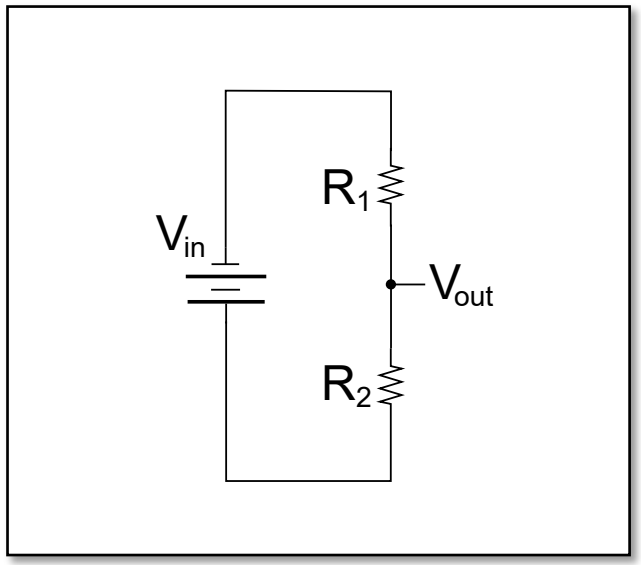


If an input pin is disconnected, its voltage is nondeterministic

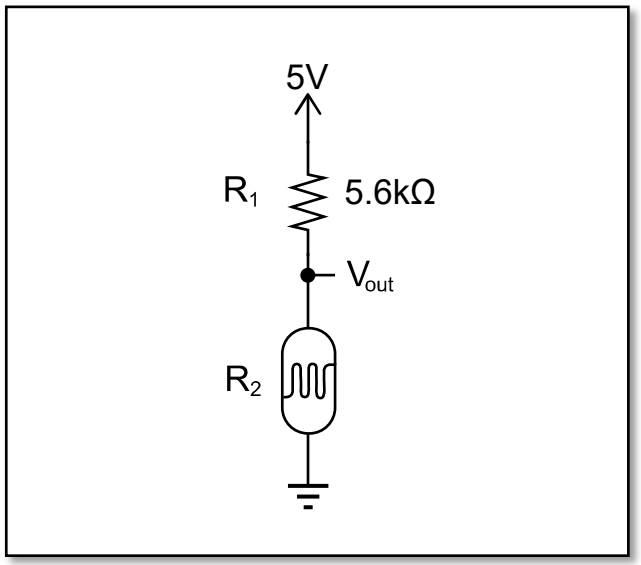


Solution – attach to ground, but add a resistor so you don't short-circuit

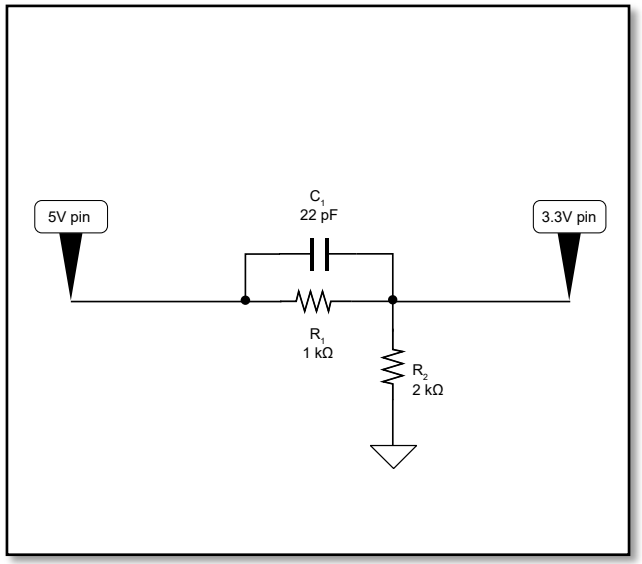
4. Getting the Voltage You Want (Voltage Divider/Regulator/Level Shifter)



Voltage Divider



Reading Resistive Sensor

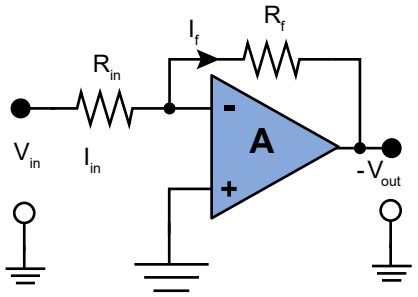


Level Shifter

- May need different voltage from your supply
- Resistors can be used to change voltage
- Transistors can be used to “switch in”

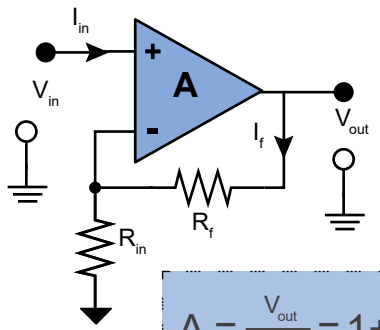
5. Amplifying What You Want to Hear (Operational Amplifiers)

Inverting Op-amp



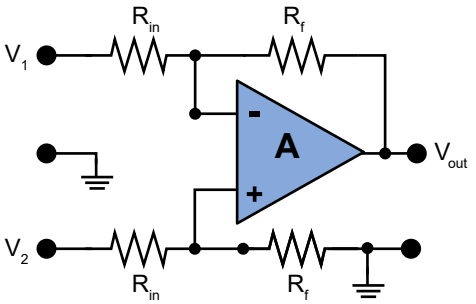
$$A = \frac{V_{out}}{V_{in}} = -\frac{R_f}{R_{in}}$$

Non-inverting Op-amp



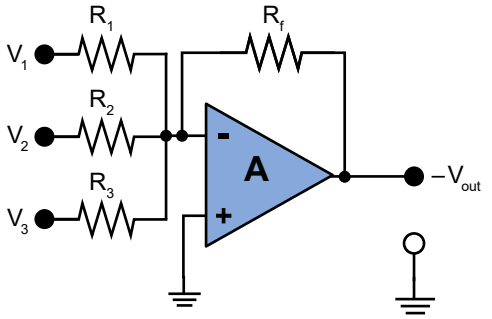
$$A = \frac{V_{out}}{V_{in}} = 1 + \frac{R_f}{R_{in}}$$

Differential Op-amp



$$V_{out} = \frac{R_f}{R_{in}}(V_2 - V_1)$$

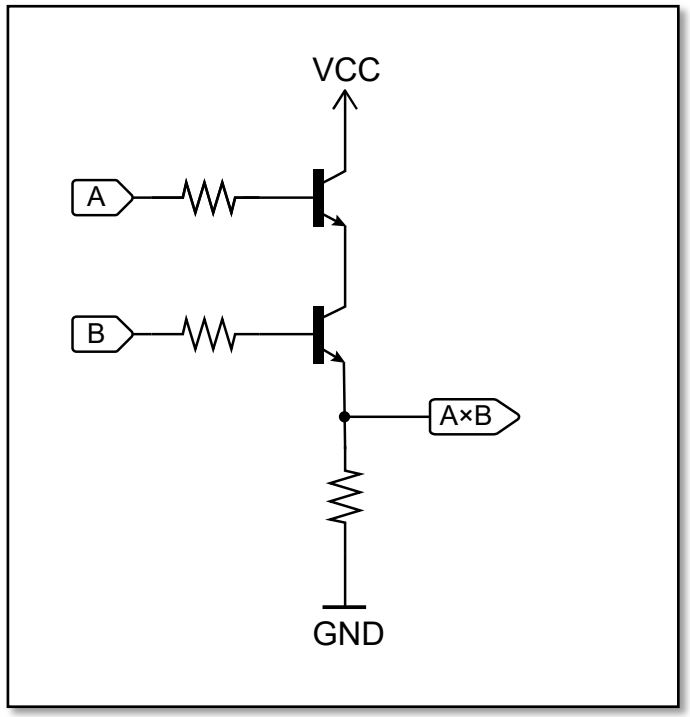
Summing Op-amp



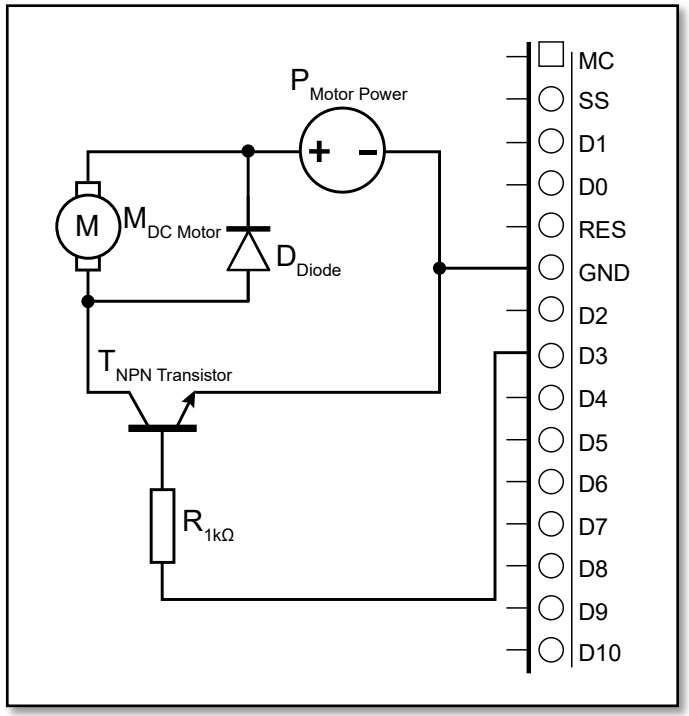
$$V_{out} = -\left(\frac{R_f}{R_{in}}V_1 + \frac{R_f}{R_2}V_2 + \frac{R_f}{R_3}V_3\right)$$

- Two inputs, one output; amplifies difference between inputs
- Good Op-amps reject noise (common-mode voltages)
- Widely used to amplify inputs: audio amplifiers, sensors, radio NICs

6. Transistor Switches (Transistors, Relays)



Logic Gates



Controlling Voltage

- Allow an input to be controlled by another input
- Useful when we need to do conditional logic
- Useful when using logic to drive much larger voltage/current source

Integrated Circuits

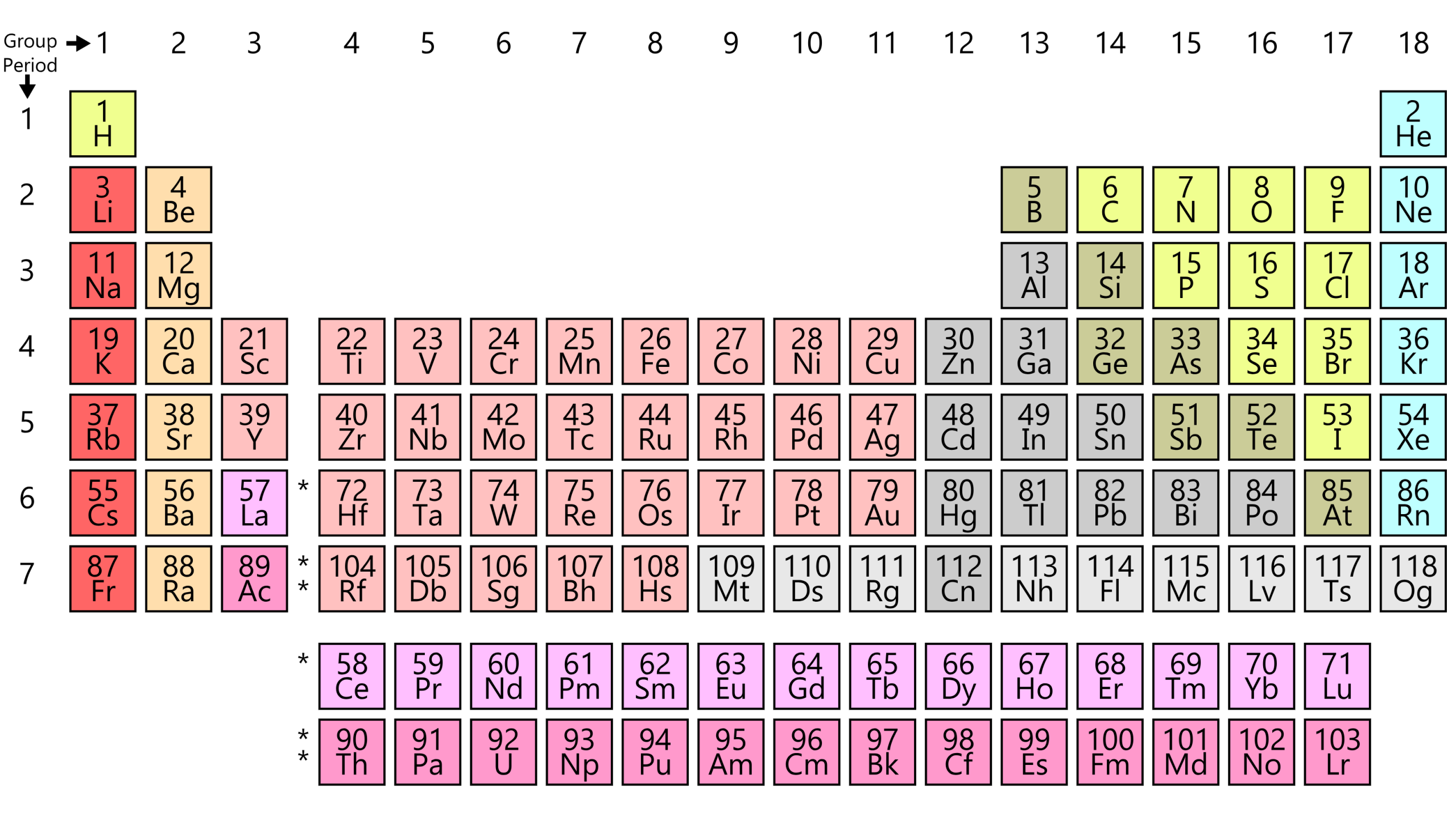
- “Discrete” circuits are great, but limits to how small/cheap you can build them

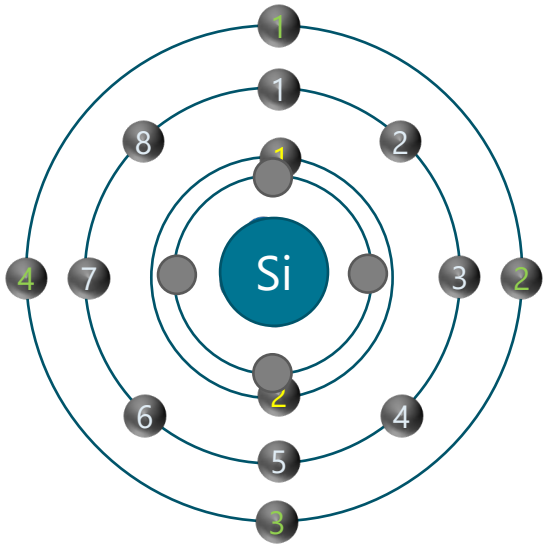
- Discovery in 1960s: you can actually “draw” circuits
  - Cost is low because they are printed en masse
  - Small size and tight proximity → they can switch quickly and use little power

- However, high initial cost to design and fabricate required “photomasks”
  - Practical only for high production volumes

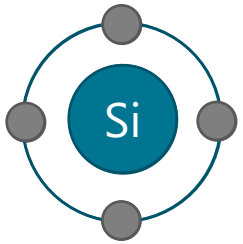
Underlying Technology Behind ICs: Semiconductors

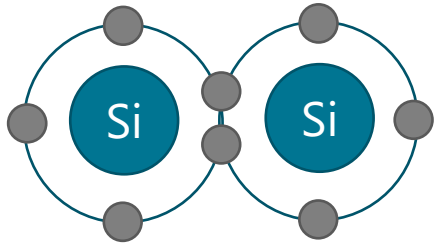
- Want a material you can “draw” on
- Semiconductor: poor insulator
  - Can “dope” it to become conductor
- Can selectively dope it in certain areas
  - Etch tiny wires, components
- Side benefits:
  - Excited electrons can emit light instead of head (LEDs)
  - Resistivity can change with temperature, light, exposure to gas (sensors)
  - High thermal conductivity (good heat dissipation)


Group
Period
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
1 H He
2 Li Be B C N O F Ne
3 Na Mg Al Si P S Cl Ar
4 K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
5 Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe
6 Cs Ba La * Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
7 Fr Ra Ac * Rf Db Sg Bh Hs Mt Ds Rg Cn Nh Fl Mc Lv Ts Og
* Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu
* Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr

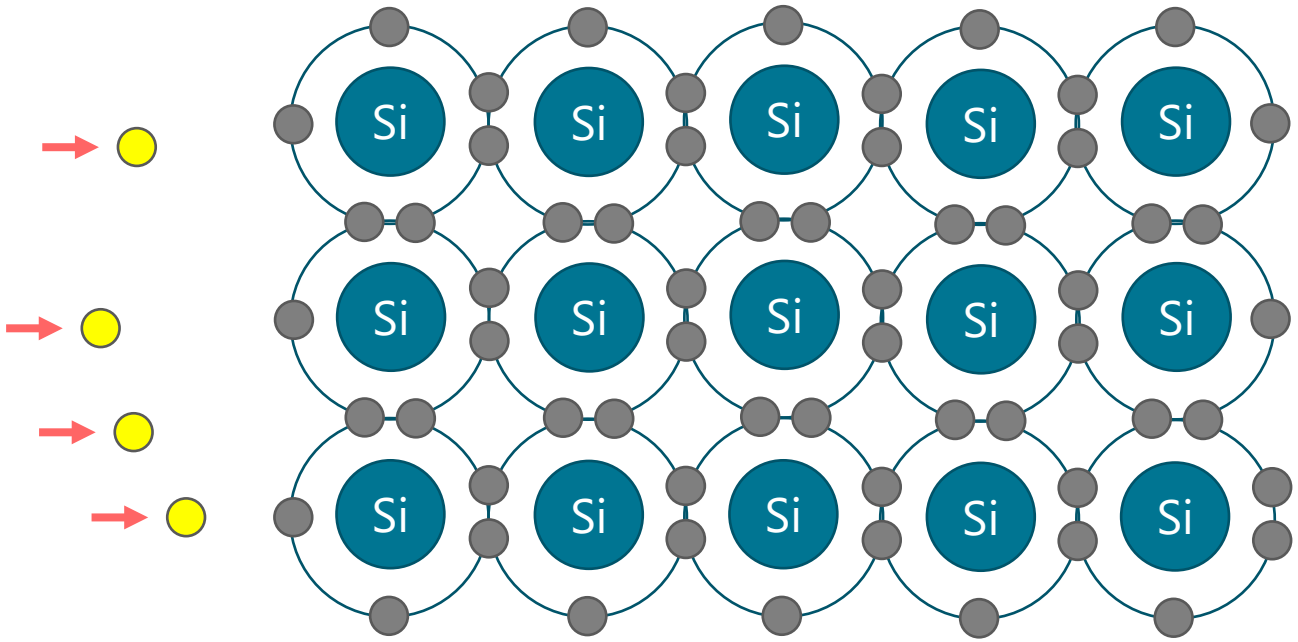


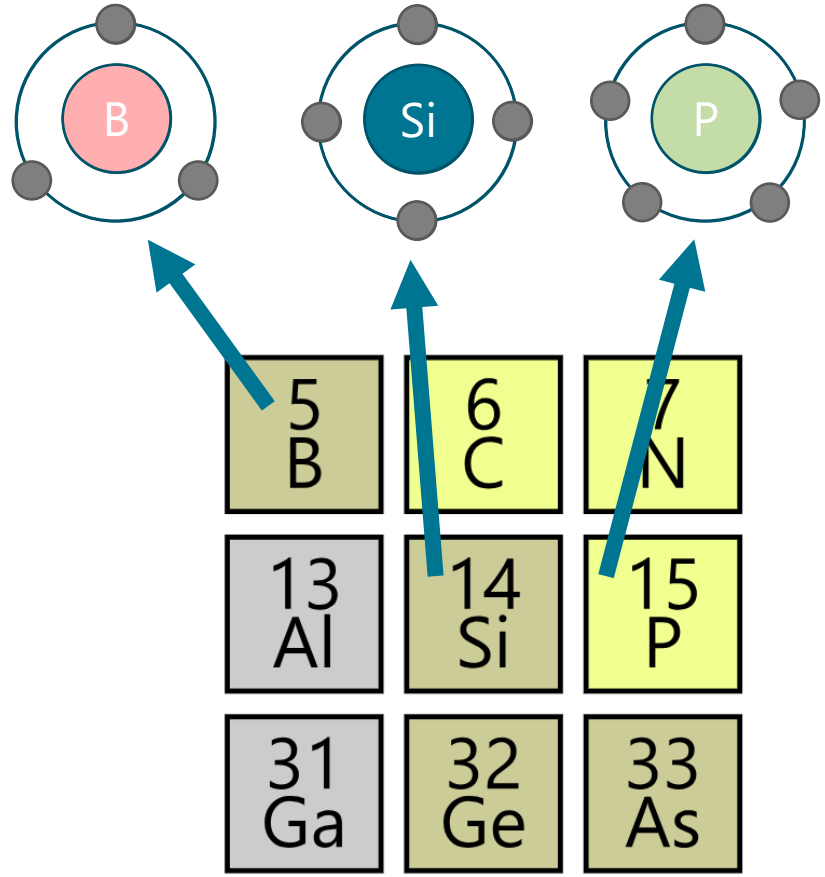
5 B	6 C	7 N
13 Al	14 Si	15 P
31 Ga	32 Ge	33 As


Si

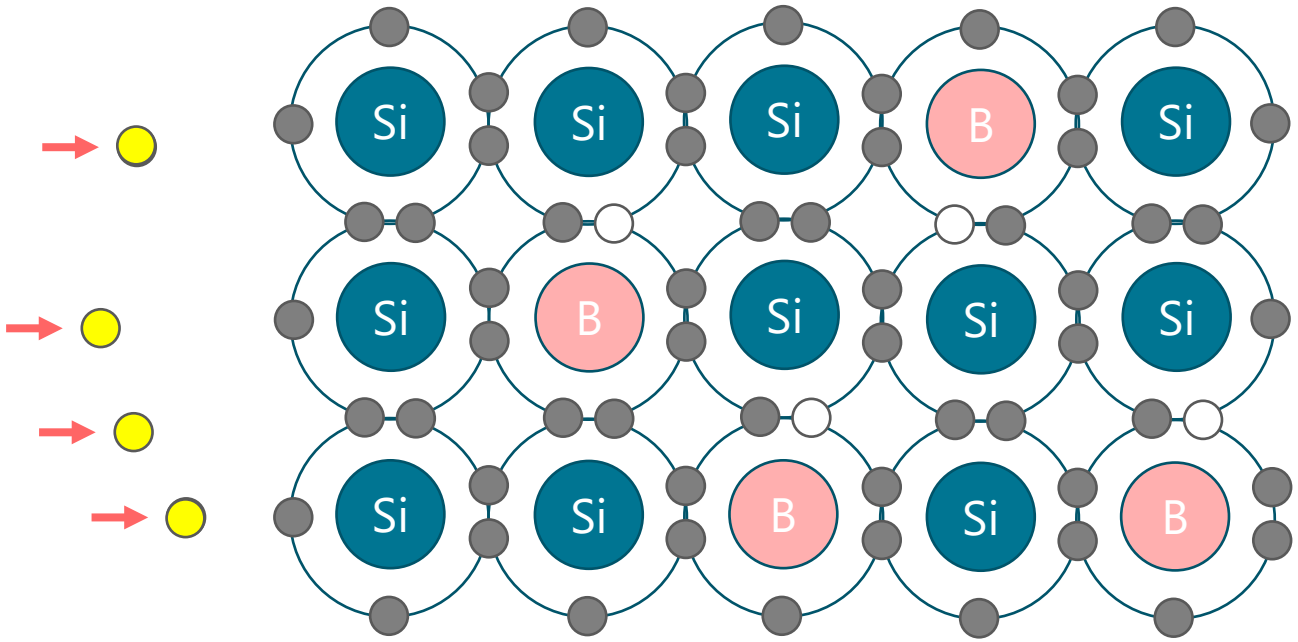

Si
Si

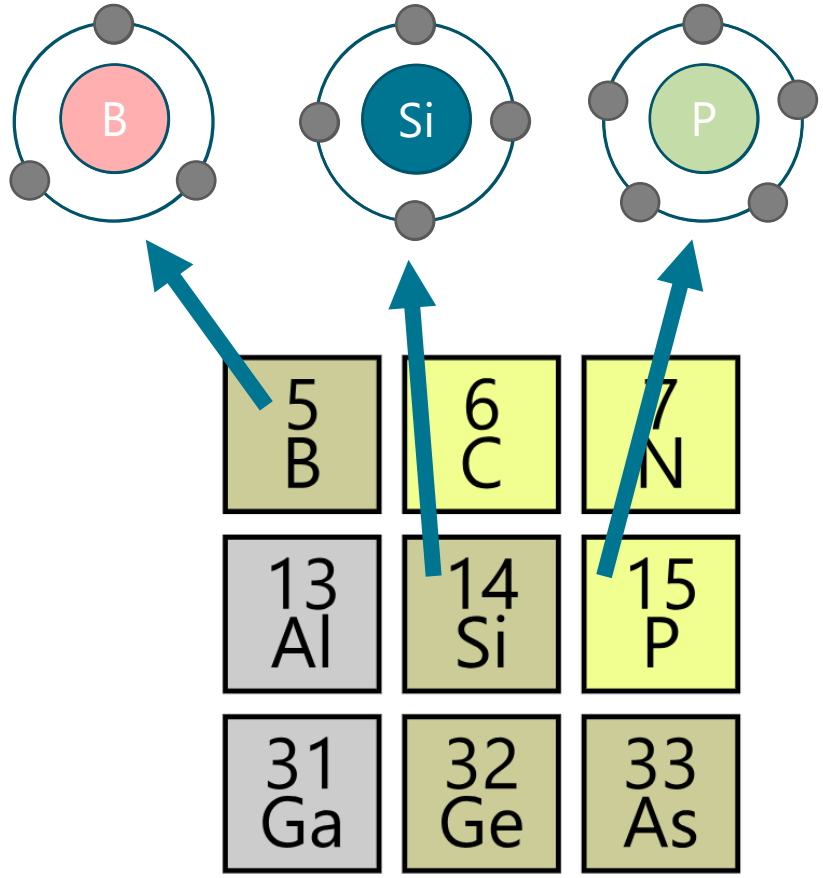
Undoped Semiconductors Are Insulators



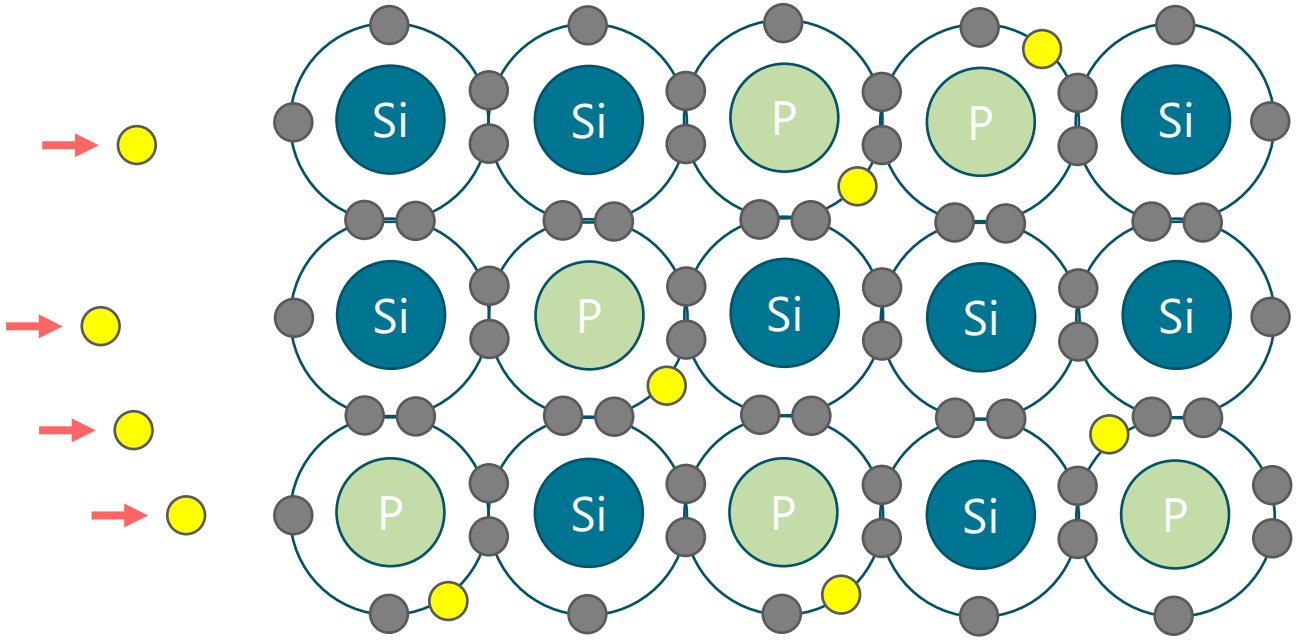


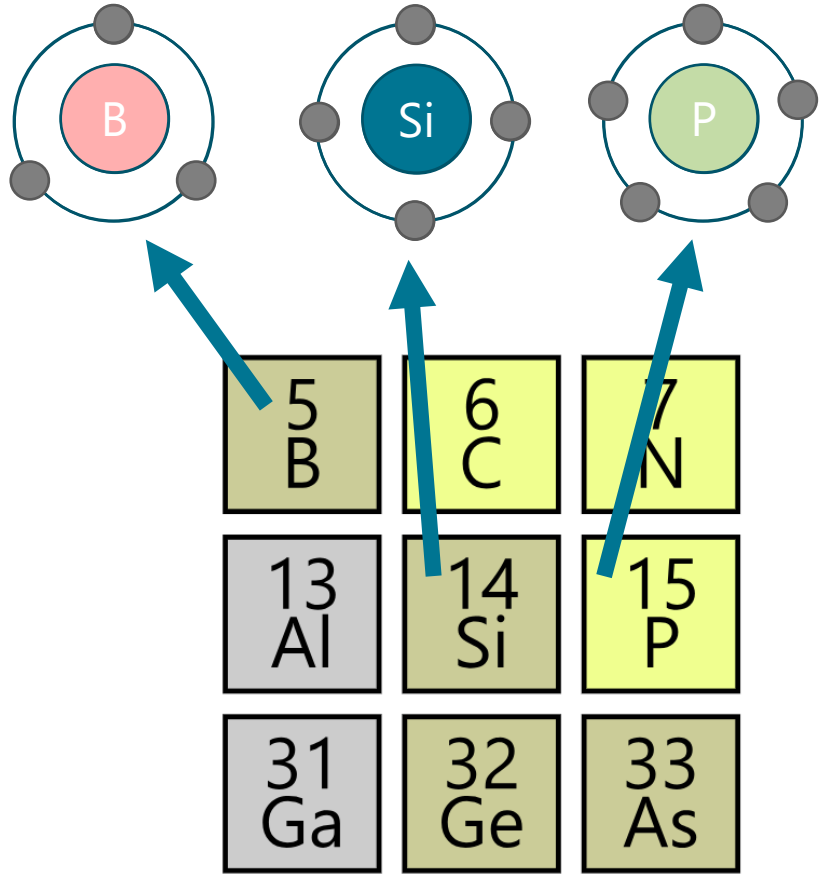
Doping with Boron



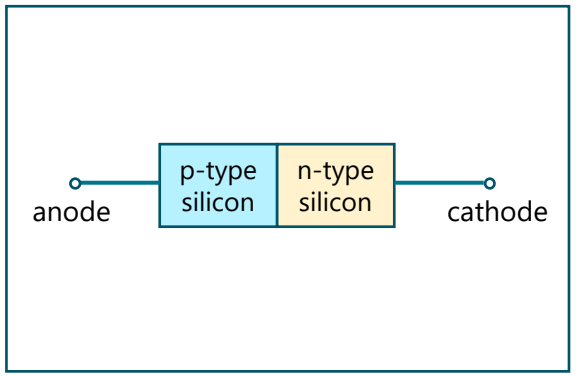


Doping with Phosphorus

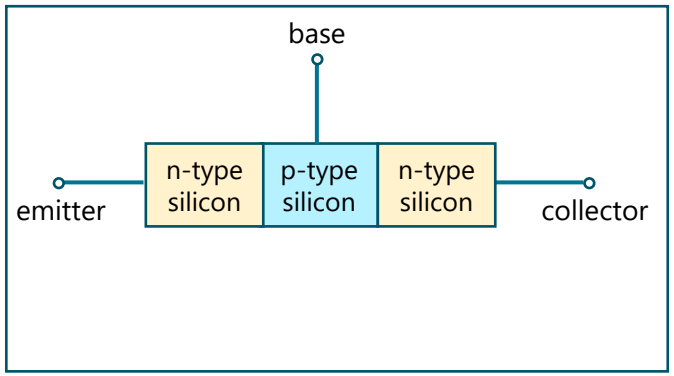




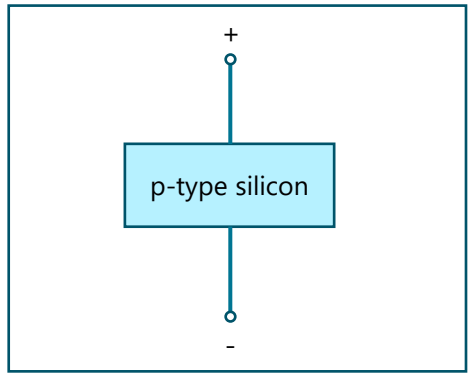
How to "Print" Components



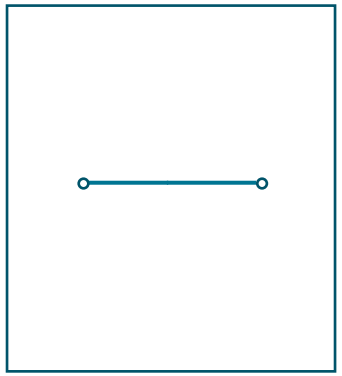
Diode



Transistor



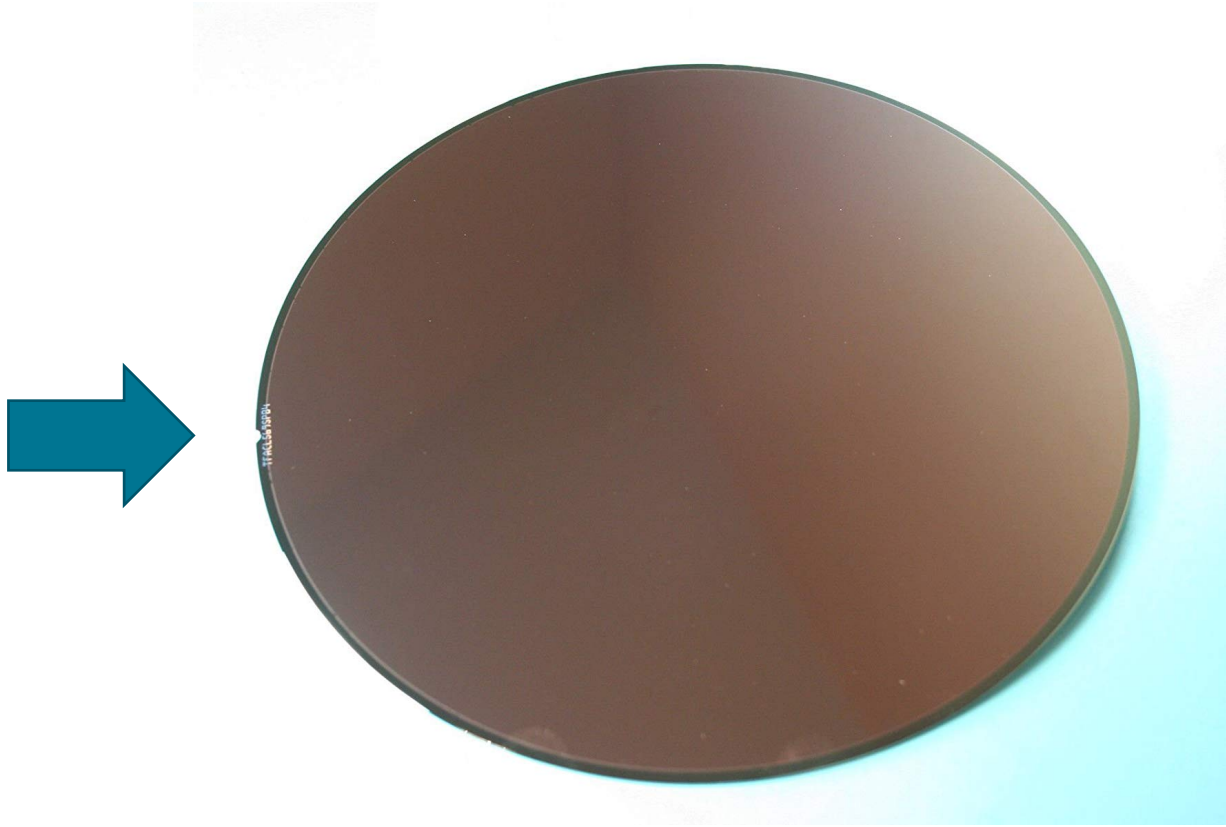
Capacitor

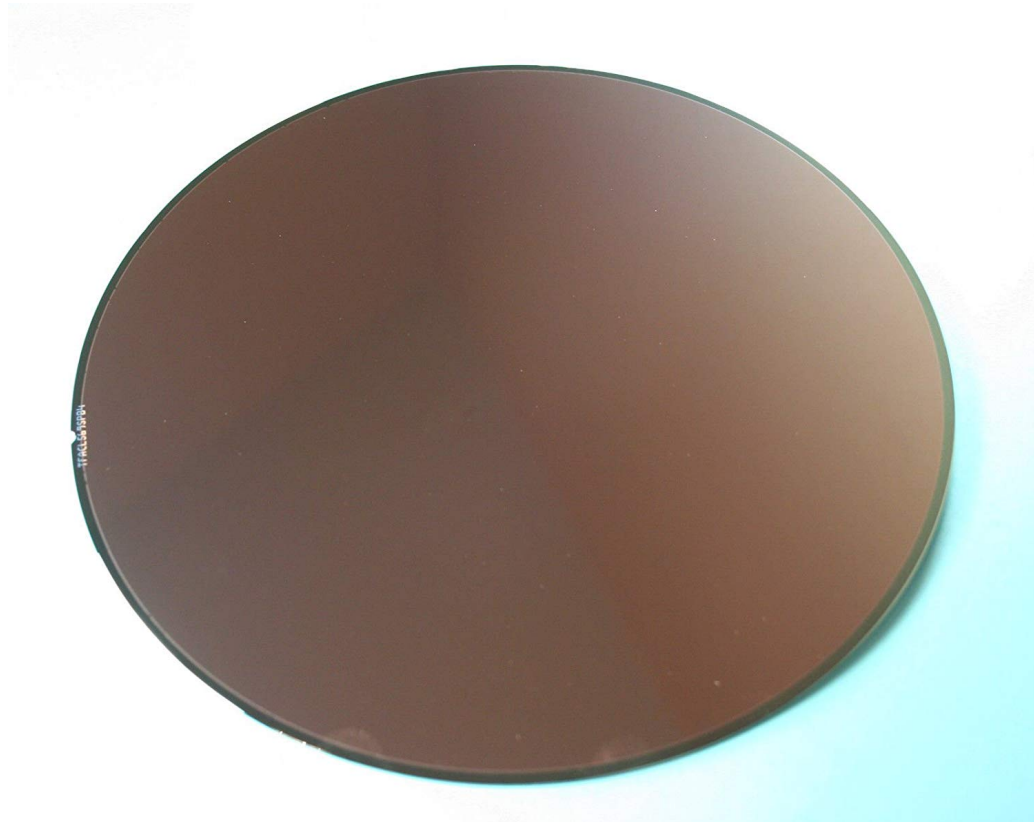


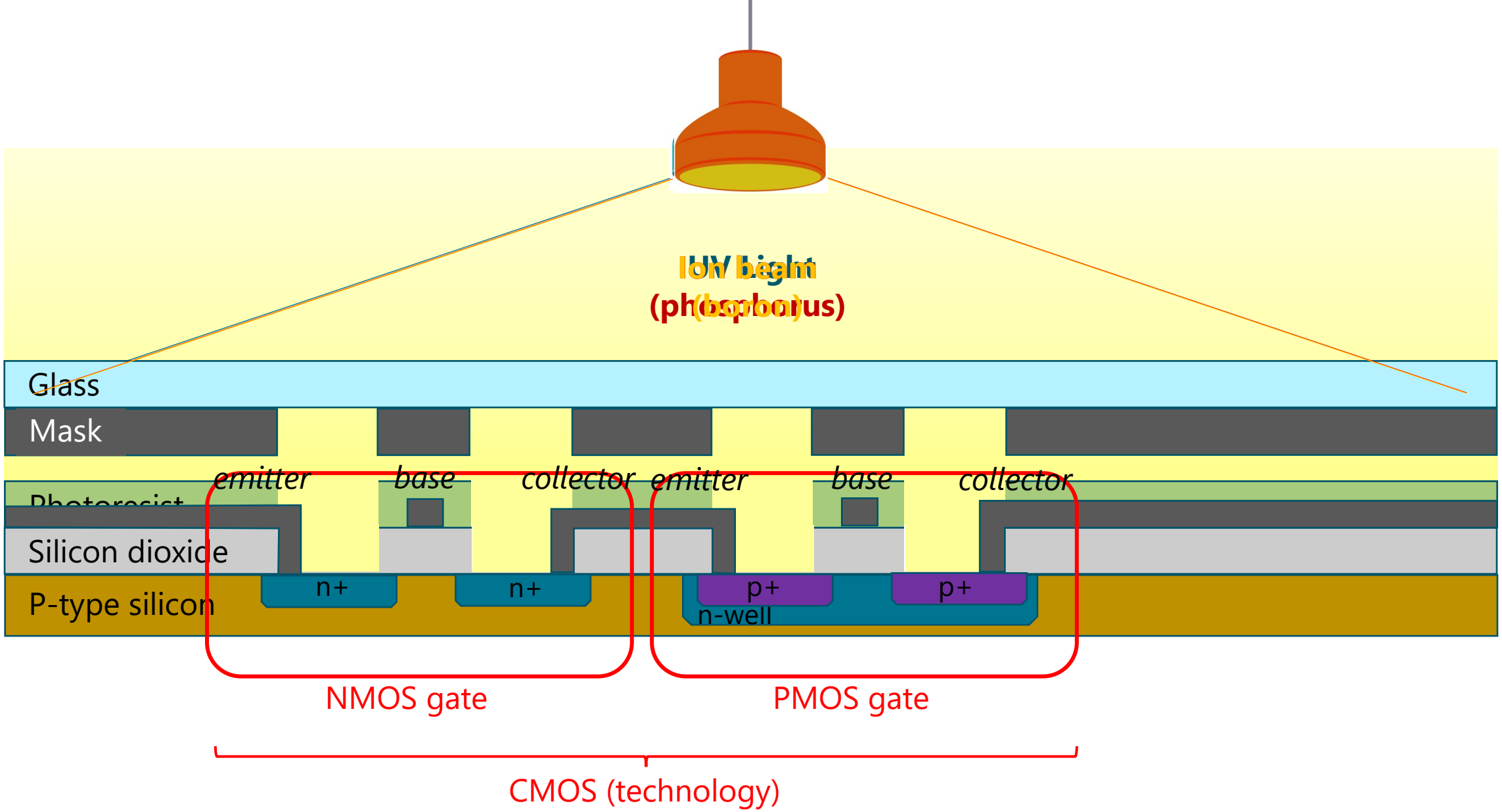
Resistor

- Can "draw" (etch) tiny components using "doping"
  1. Lithography: draw your design onto photomask, apply to surface of silicon wafer
  2. Etching: "carve" your design into the silicon
  3. Deposit metal for connections

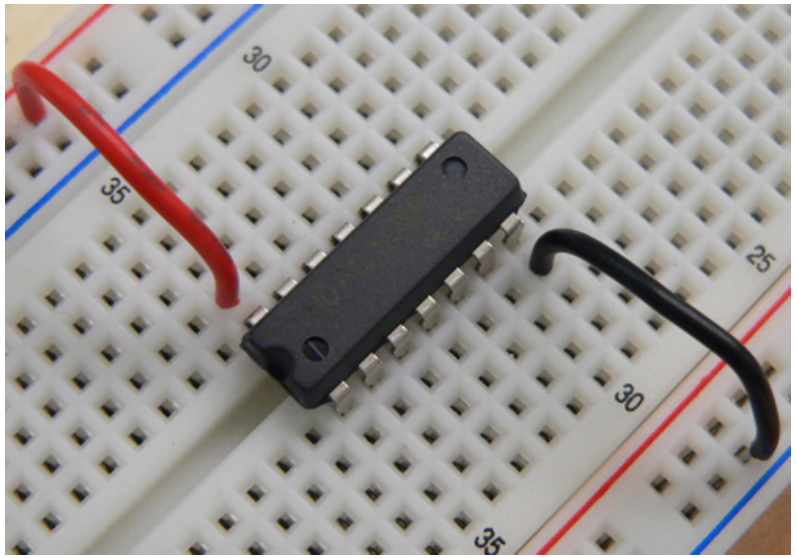
How ICs Are Made

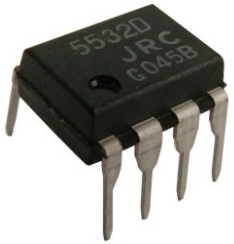




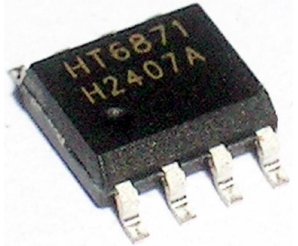

UV light
Ion beam
(phosphorus)
(boron)
Glass
Mask
Photoresist
Silicon dioxide
P-type silicon
emitter
base
collector
emitter
base
collector
n+
n+
p+
p+
n-well
NMOS gate
PMOS gate
CMOS (technology)

Integrated Circuit Packaging





Dual-Inline Package (DIP)

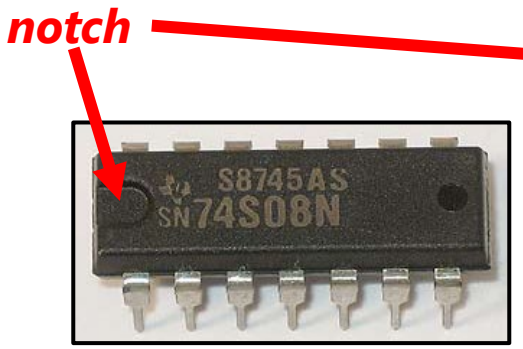


Small-Outline IC (SOIC)

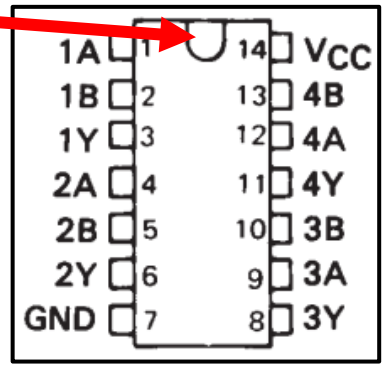
Quad-Flat Package (QFP)

Surface-Mount Device (SMD)

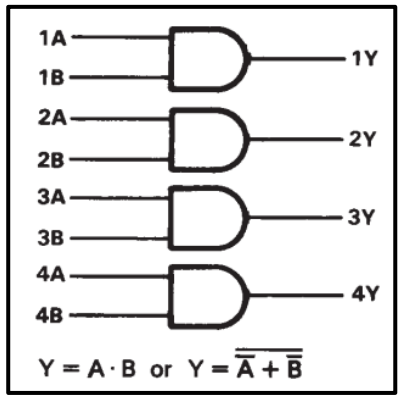
Example IC: Texas Instruments SN74S08N (Quad AND Logic Gates)



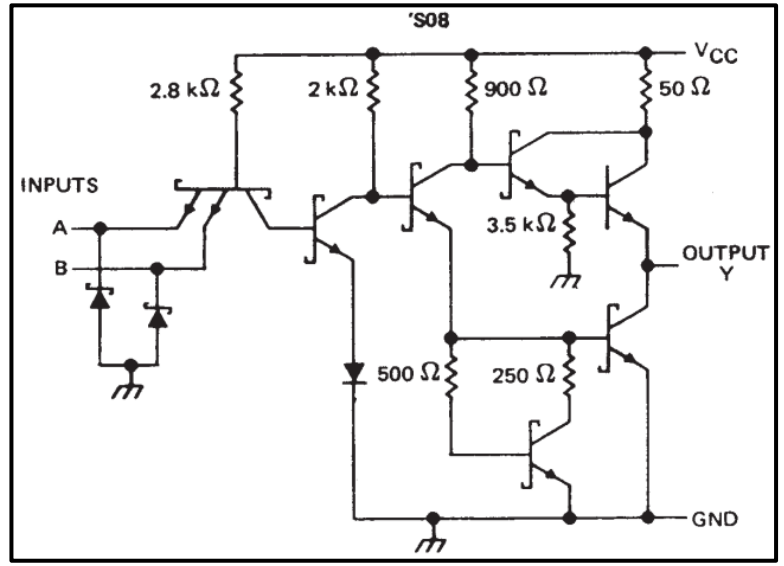
SN74S08N (DIP)



Pinouts



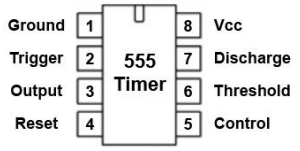
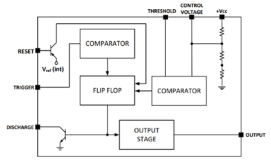
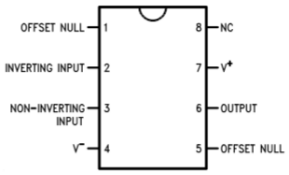
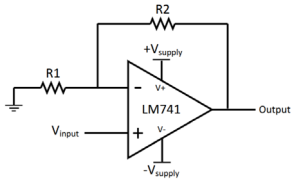
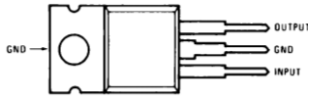
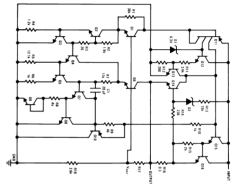
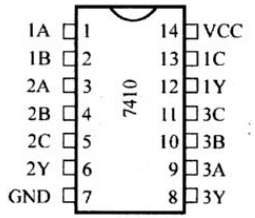
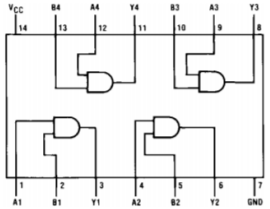
Logic Diagram



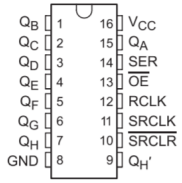
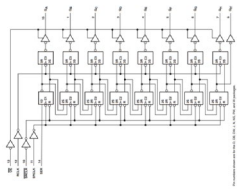
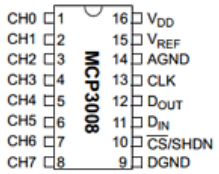
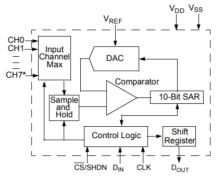
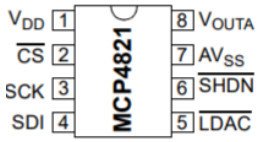
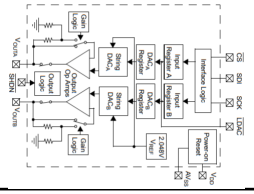
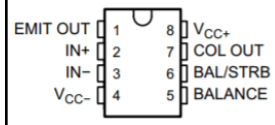
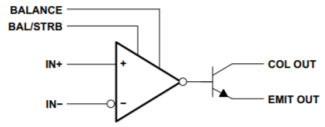
Schematic (each AND gate)

- Datasheet: https://www.ti.com/lit/ds/symlink/sn74ls08.pdf
  - Google "SN74S08N datasheet"
- Price: $1.61 for one, $0.65 each for 5000 (Digikey.com, January 2019)

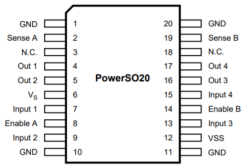
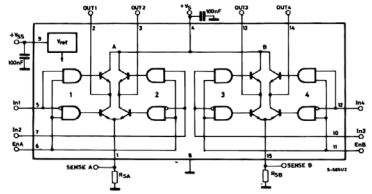
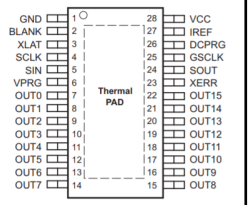
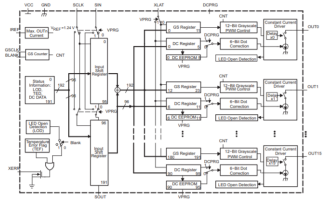
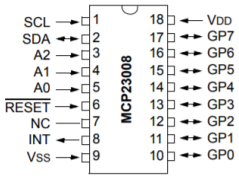
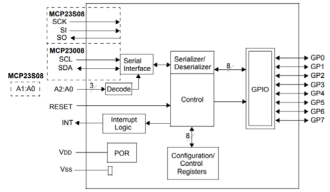
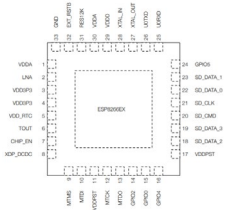
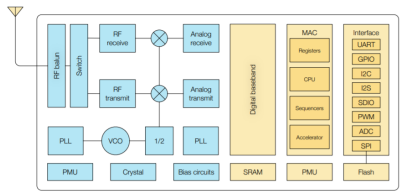
Popular IC Types

IC family	Type	Pinout	Logic diagram	What it does	Alternatives/ Variants
555	Timer			Can configure time and mode. Sends pulse after time period (monostable config) or at regular intervals (astable config). Most popular IC. >1 Billion sold every year.	XR-2206 (also generates sine/sawtooth waves), 8038, CD4046
741	Operational Amplifier (Op-Amp)			Produces output voltage much higher than voltage difference between input terminals	LM324 (has four op-amps in one 14-pin DIP package, doesn't require separate pos/neg power supplies), LM358, TLC272; TL0XX (eg, TL081, TL082, TL084)
78xx	Voltage Regulator			Accepts input voltage within a certain range and produces a constant output voltage. xx represents voltage regulated by chip (e.g., 7805 produces a 5-volt output)	78xxSR, 78xxC, 78xxS, 79xx, LM317, LM2575T-ADJ (output adjustable 1.23V to 37V), MC34063A, XL6009
74xx	Logic Circuits			Digital logic operations. Includes logic gates, flip-flops, counters, buffers, and more.	7408 (AND), 7404 (NOT), 7486 (XOR), 7402 (NOR), 7400 (NAND), 7474 (Flip-flops)

Popular IC Types

IC family	Type	Pinout	Logic diagram	What it does	Alternatives/ Variants
74HC595	Shift Register			Stores an array of bits. Can "shift" bits up, losing (or wrapping) bits off the end. Used to allow many pins to be controlled by 1-2 pins, to multiply by two, etc.	74LS195 (4 bits), 4006 (18 bits), 4014 (8 bit static), 74LS194 (bidirectional 4-bit)
MCP3008	A/D Converter			Converts analog to digital; takes an analog voltage level on one pin, outputs across a set of pins a binary number representing the level of that voltage.	MCP3002. PCF8591,
MCP4822	D/A Converter			Opposite of an A/D converter: takes a binary input across a set of pins, and outputs a voltage corresponding to that number.	MCP4725 (12-bit), MCP4728, MCP4921, AD7390
LM311	Comparator			Compares two input voltages and outputs a digital signal indicating which is larger (output is 1 if V+>V-, 0 otherwise. Can be used to create null detectors (whether input is 0 volts), to generate triangle/square waves (eg car turn signal), etc.	LM312, LM393, LM339

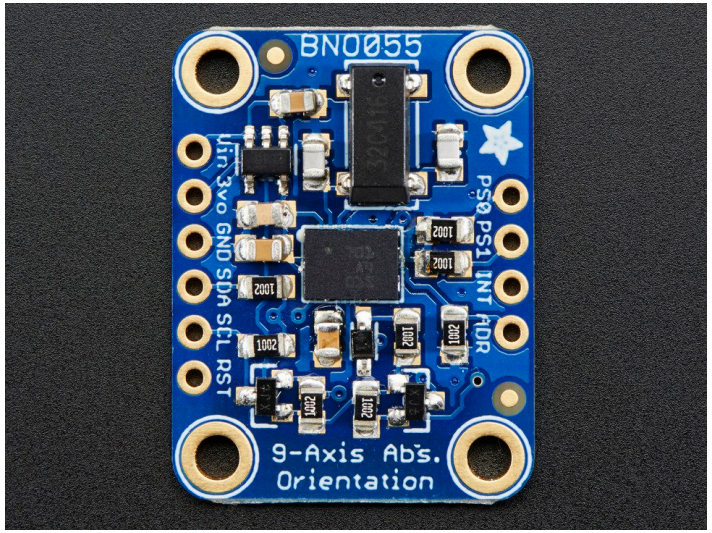
Popular IC Types

IC family	Type	Pinout	Logic diagram	What it does	Alternatives/ Variants
L298D	Motor Driver			Accepts standard TTL logic level voltages, and drives big inductive loads such as electromagnets, stepping motors, valves, pumps, etc.	L298N, L293
TLC5940	LED Driver			16 channels, each has 4096-step PWM brightness control. Brightness controlled by programmable on-chip EEPROM. Reports error information: broken/disconnected LED, overtemperature condition.	TLC5947, LT3741, ADP8861
MCP23008	I/O Expander			Change one serial input to 8 pins of parallel output. Adds 8 pins to your microcontroller.	MCP23017, MCP 23016 (16-bit), SX1509, TCA9535
ESP8266	WiFi module			802.11 b/g/n WiFi with WEP/WPA/WPA2 authentication (or open networks) and full TCP/IP stack. Also contains embedded microprocessor core	ESP8285 (has 1MB built-in flash), ESP32 (successor)

So, can we just make everything an IC?

- ICs are great
  - “Componentize” circuits – don’t need to understand insides so much
  - Reduced cost
- But can’t do everything as an IC
  - High current/voltage – exceeds logic levels
  - Low volume or too new – no ICs exist for it
- Can we componentize non-IC circuits?

Breakout Boards

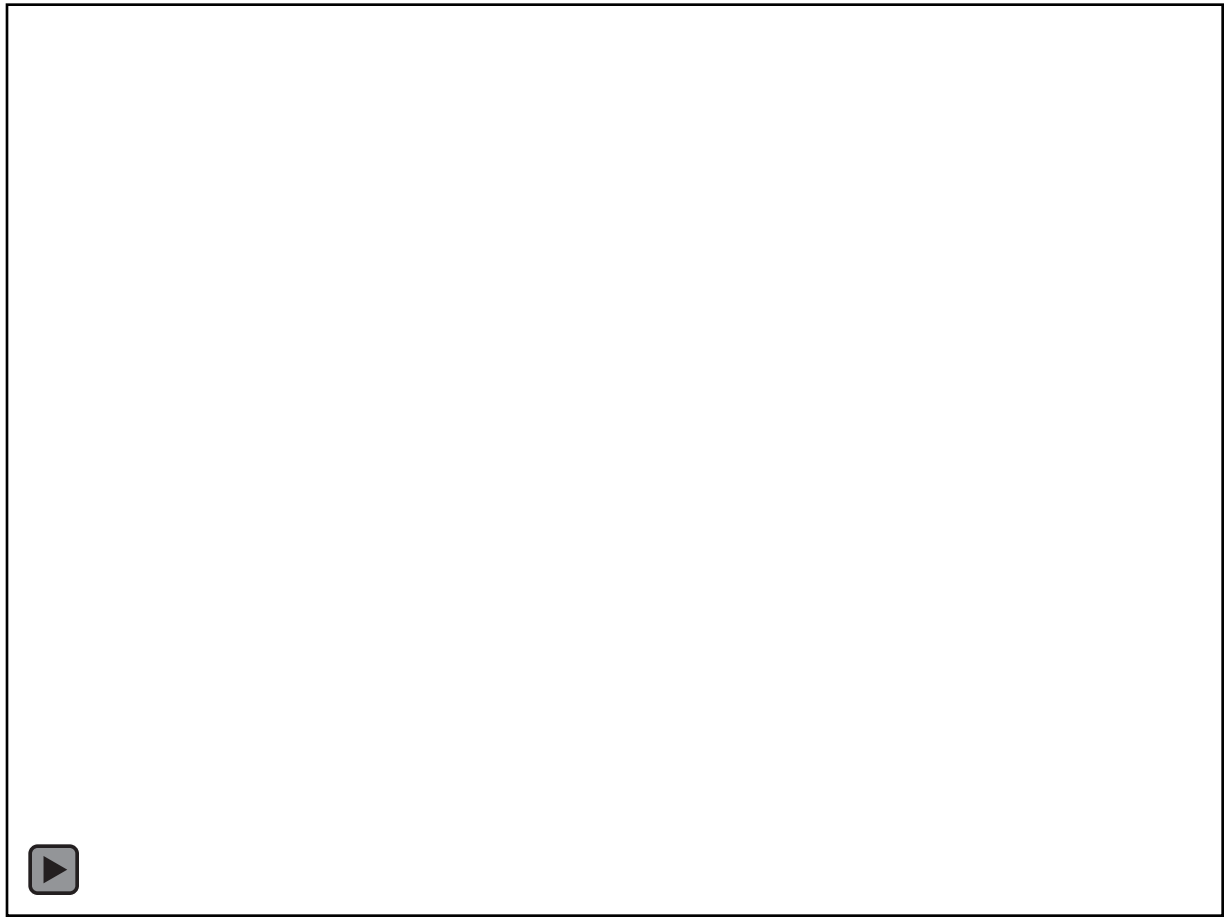


- Standalone board that componentizes a circuit
  - "Breaks out" key pins to simplify prototyping
  - Deals with complicated EE stuff, lets you focus on using the circuit
  - Incorporates collection of components onto a single PC
- Example: Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout – BNO055 = $34.95 (January 2019) (Source: https://www.adafruit.com/product/2472)

Breakout Boards

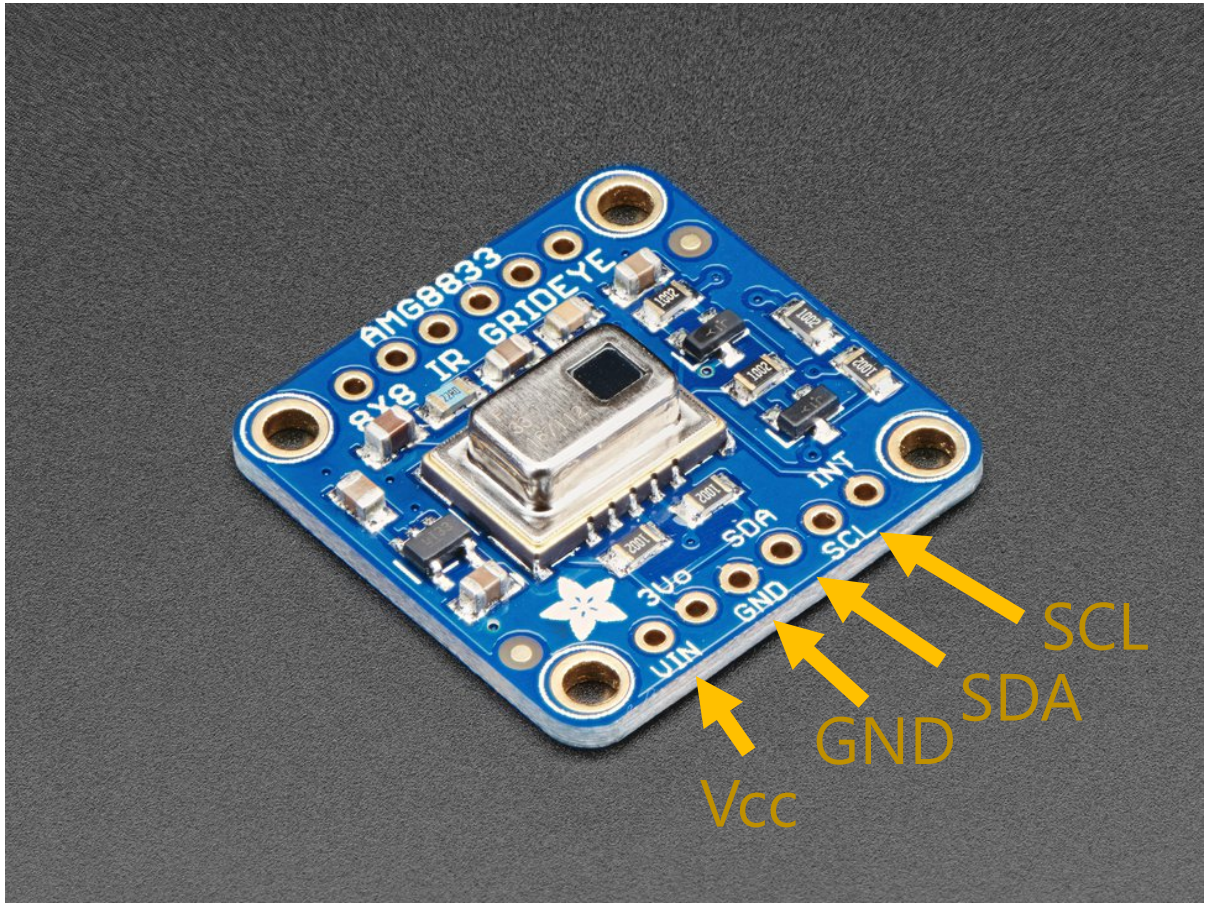
- Benefits: reduces costs, saves space, improves efficiency/reliability, simplifies reuse, clear pinouts, good documentation
  - Lets you do a lot with less domain knowledge
  - Speeds prototyping
  - Saves you soldering work
- Challenges: may not be interchangeable/compatible (e.g., may target a specific platform like Arduino), may need to solder or install header pins, need to be careful about supply voltage and other inputs

Example: Adafruit AMG8833 IR Thermal Camera Breakout



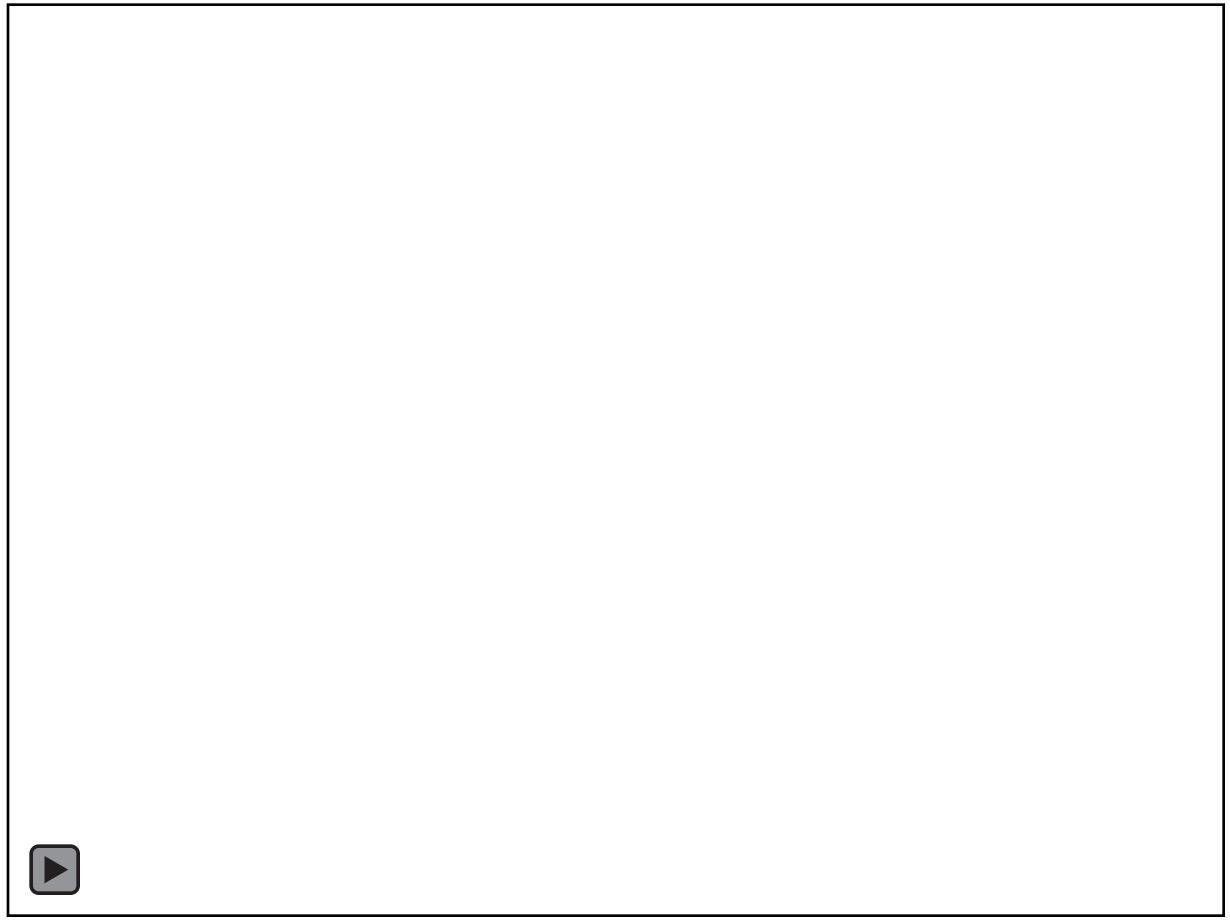
- 8x8 array of thermal IR sensors
  - Can detect human 23 feet away
  - 10 Hz frame rate
- **Uses:** human detector, thermal camera, intruder alert
- **Incorporates:** Panasonic AMG8833 thermal IR sensor, 3.3V regulator, misc. resistors/capacitors

Example: Adafruit AMG8833 IR Thermal Camera Breakout



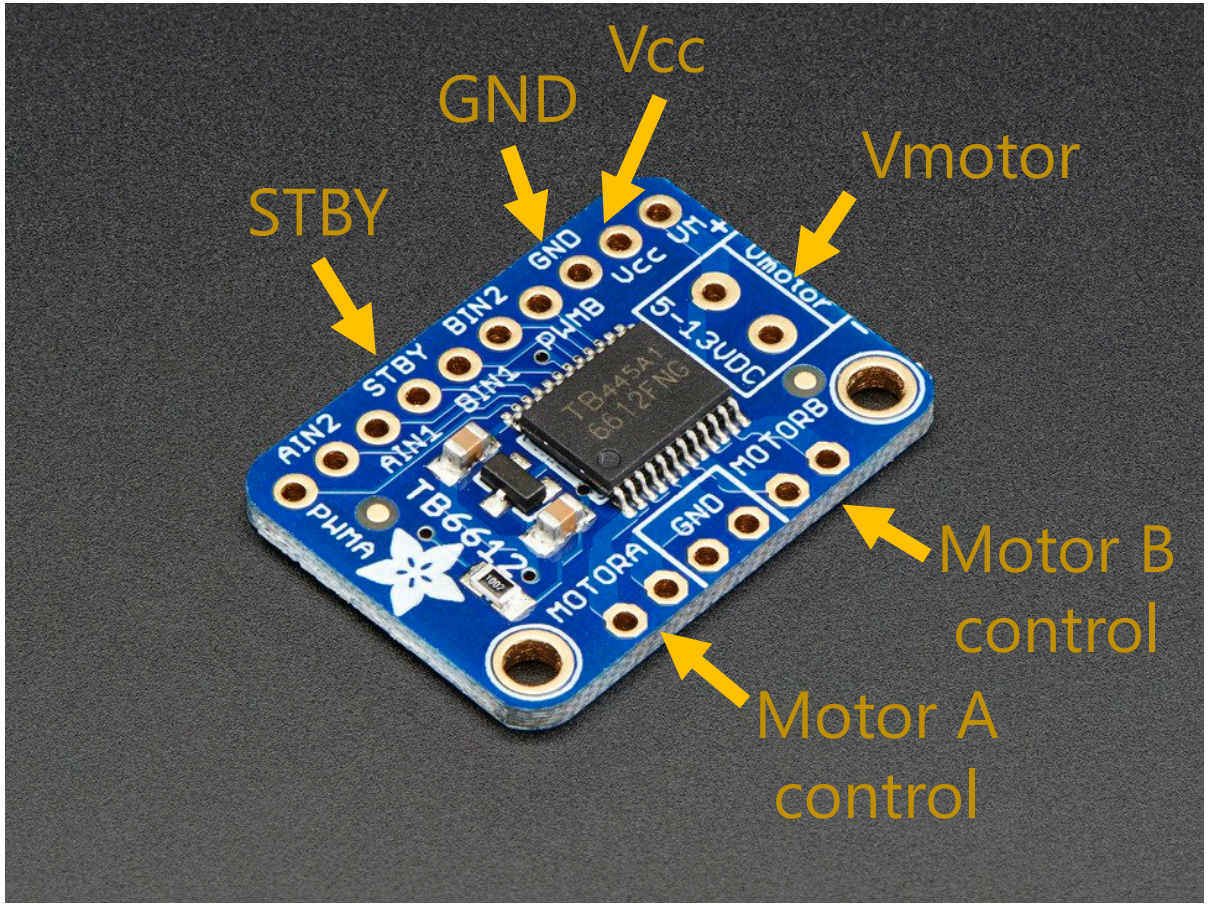
- **Pinout:**
  - Input voltage (Vin)
  - Output voltage in case you can use it (3Vo)
  - Ground (GND)
  - Serial clock – output (SCL)
  - Serial data – output (SDA)
  - Interrupt output – when something changes in vision path (INT)
  - (Pins at top for stability, unused)

Example: Adafruit TB6612 1.2A DC/Stepper Motor Driver Breakout Board



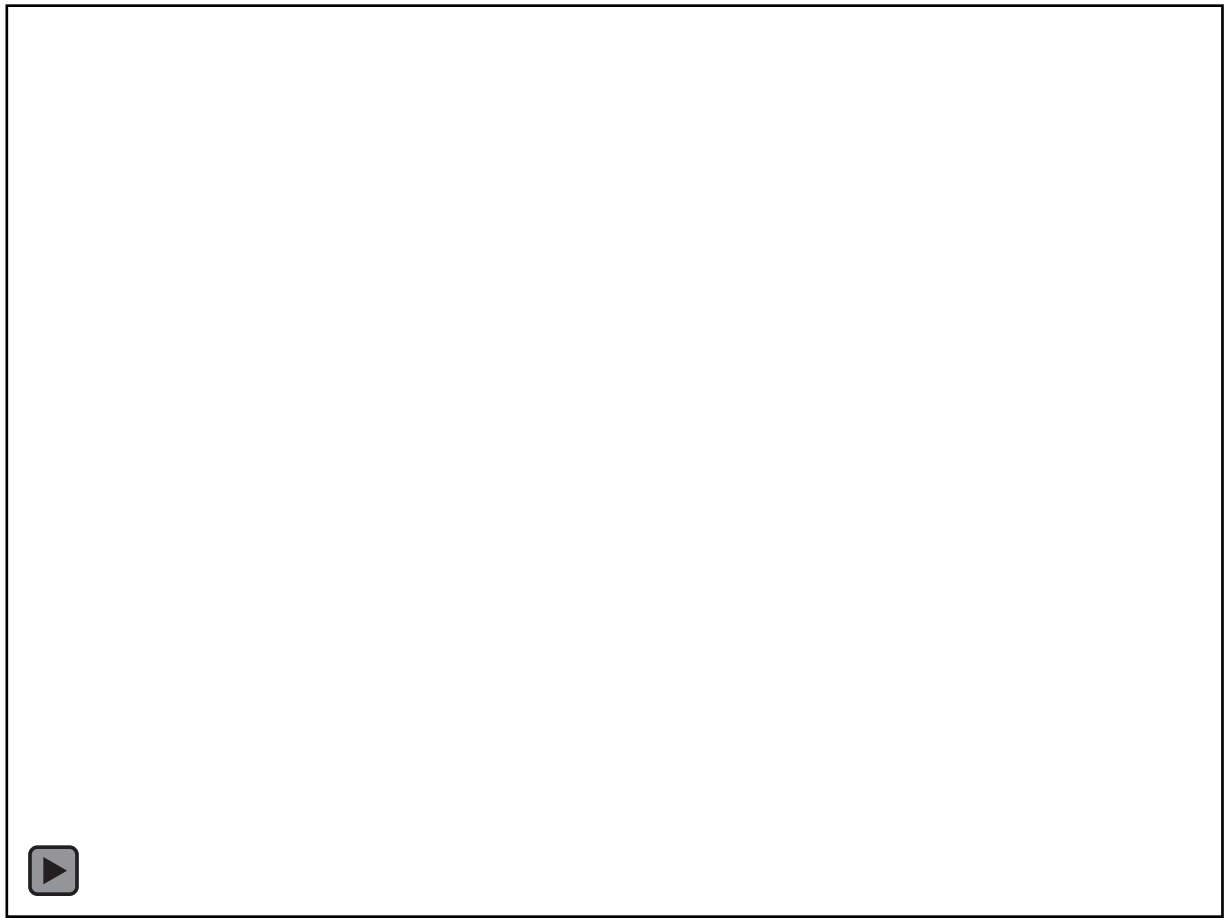
- Driver for stepper motor
  - Can drive 1.2A sustained, 3A peak (20ms)
  - Control direction and speed, stop/brake
  - Motor voltage isolated from logic voltage
- **Uses:** robot, pan/tilt housing, automotive gauge/dial, toy car
- **Incorporates:** a Toshiba TB6612 motor driver IC, two H-bridges, polarity protection FET, pullup resistor for inputs, kick-back diodes

Example: Adafruit TB6612 1.2A DC/Stepper Motor Driver Breakout Board



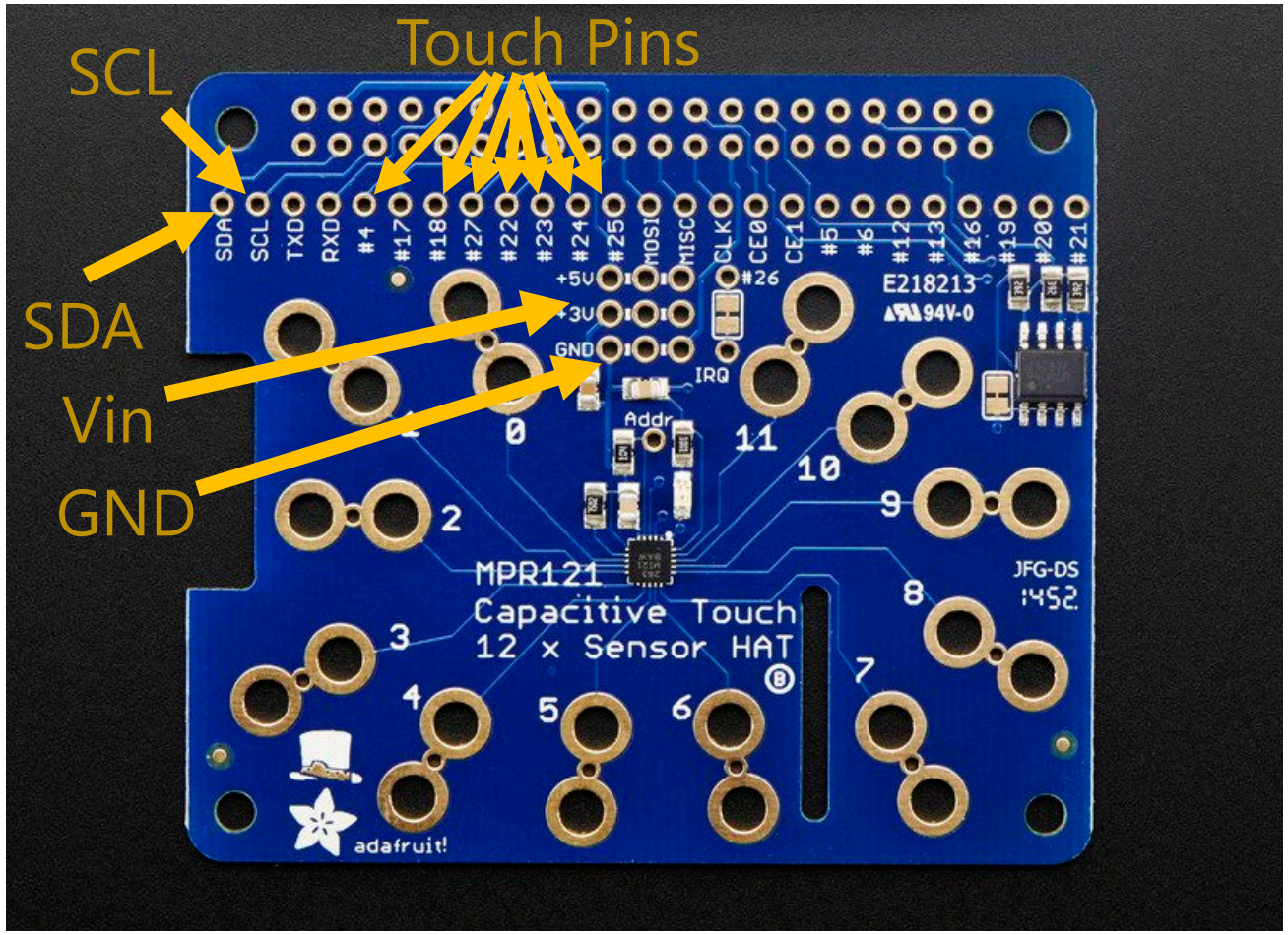
- **Pinout:**
  - Input voltage for logic levels (Vcc)
  - Voltage to drive motor (Vmotor)
  - Ground (GND)
  - Motor control outputs:
    - INA1, INA2, PWMA: outputs for motor A's H-bridge
    - INB1, INB2, PWMB: outputs for motor B's H-bridge
    - STBY: quickly disable both motors

Example: Adafruit Capacitive Touch HAT for Raspberry Pi – Mini Kit – MPR121



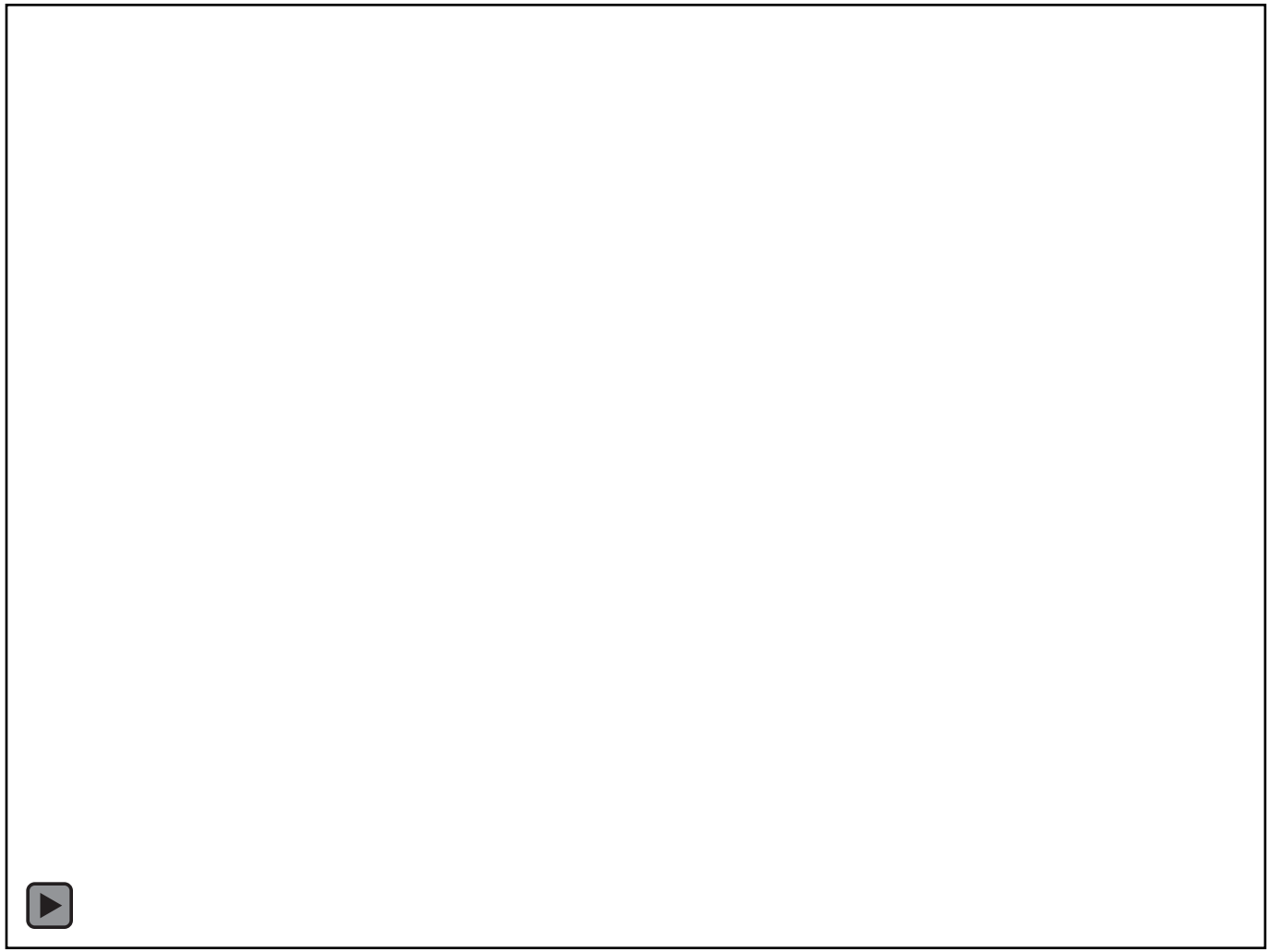
- Driver for detecting touch
  - Attach pins to something conductive (metal, something containing water)
  - “HAT” simplifies connecting to Raspberry Pi platform
  - Comes with python library
- **Uses:** touch-reactive tablets, control panels, and other objects
- **Incorporates:** a MPR121 12-key touch sense microcontroller

Example: Adafruit Capacitive Touch HAT for Raspberry Pi – Mini Kit – MPR121



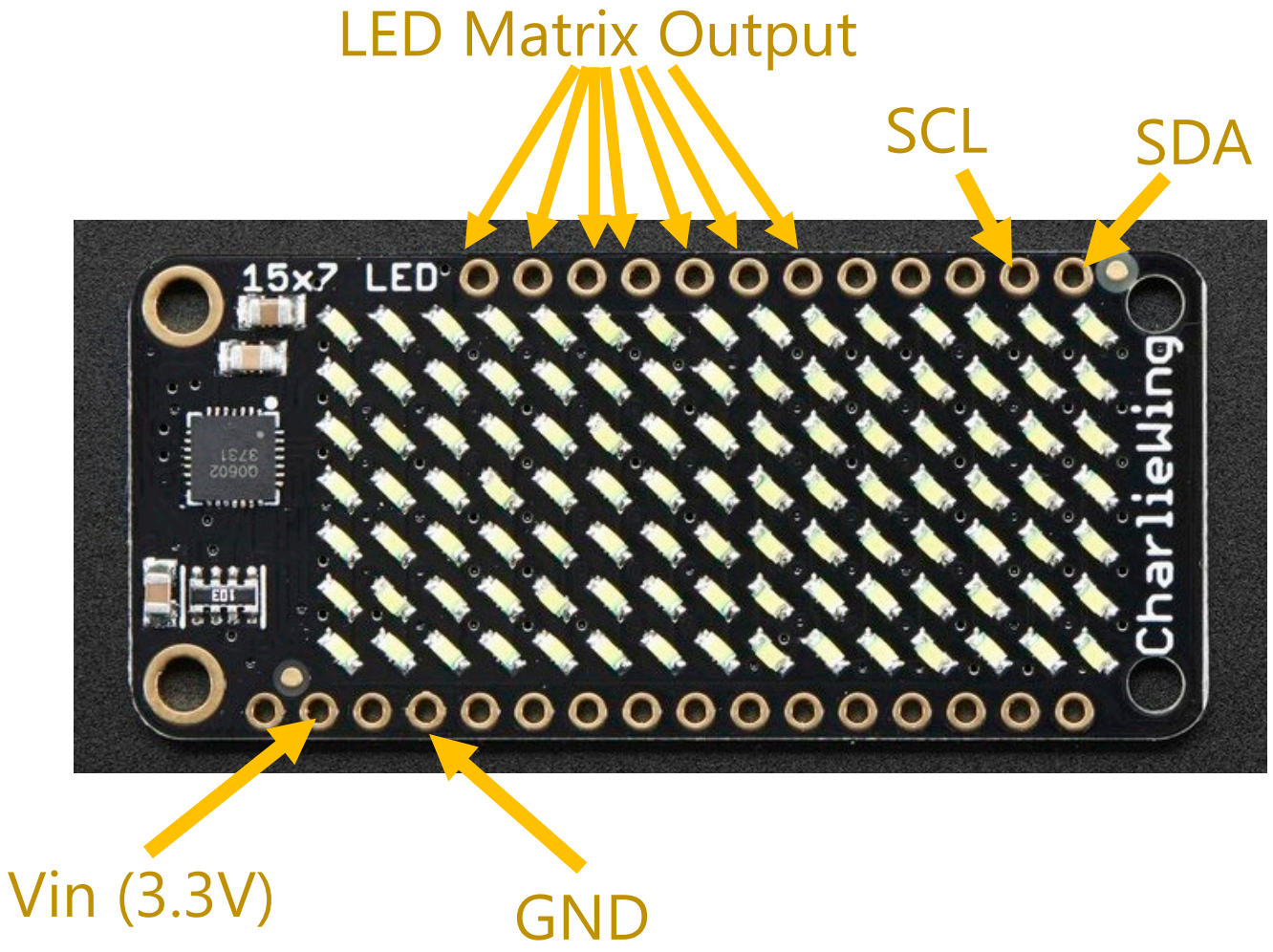
- **Pinout:** (look familiar?)
  - Input voltage (Vin)
  - Ground (GND)
  - Serial clock – output (SCL)
  - Serial data – output (SDA)
  - Touch-capacitive pins (#)

Example: Adafruit 15x7 CharliePlex LED Matrix FeatherWing – Cool White



- Drives matrix of LEDs
  - Tell chip which LEDs you want lit and their brightness, and it will PWM them for you
  - RAM for eight separate frames, can animate through them with single command
  - Comes with python graphics library
- **Uses:** displays, status indicators, human-machine interfaces
- **Incorporates:** a IS31FL3731 I2C LED driver chip

Example: Adafruit 15x7 CharliePlex LED Matrix FeatherWing – Cool White



- **Pinout:** (look familiar?)
  - Input voltage (Vin)
  - Output voltage in case you can use it (3Vo)
  - Ground (GND)
  - Serial clock – input (SCL)
  - Serial data – input (SDA)
  - LED matrix output pins

Encoding

Sending Bits on Wires

- Sometimes we need to take data and send it on wires
  - Between chips on a PCB, between different PCBs within a device, between devices in a home, between devices in different buildings
- Need some way to “encode” bits
- Idea: change some properties of the wire that can be read by the other side

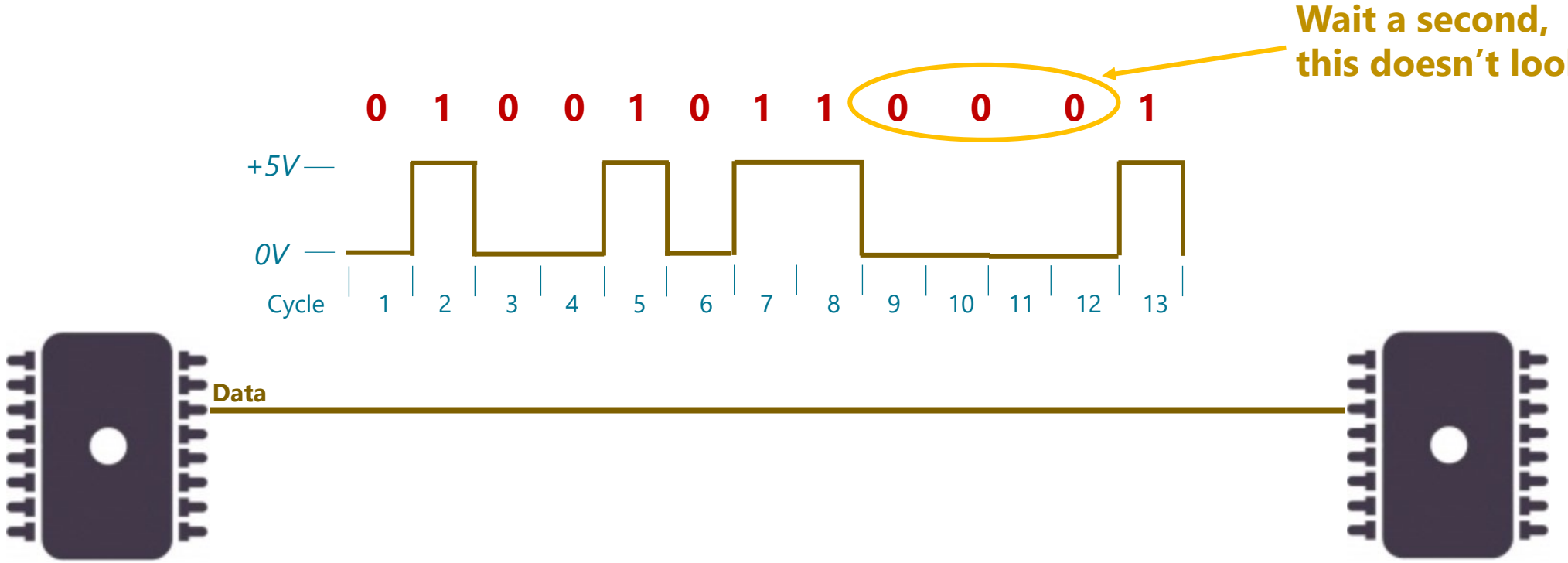
What Properties to Change?

- Really only two options: **current** and **voltage**
- **Voltage** turns out to be much more efficient
  - Current changes over wire length, requires more electricity

What Voltage Levels Should We Use?

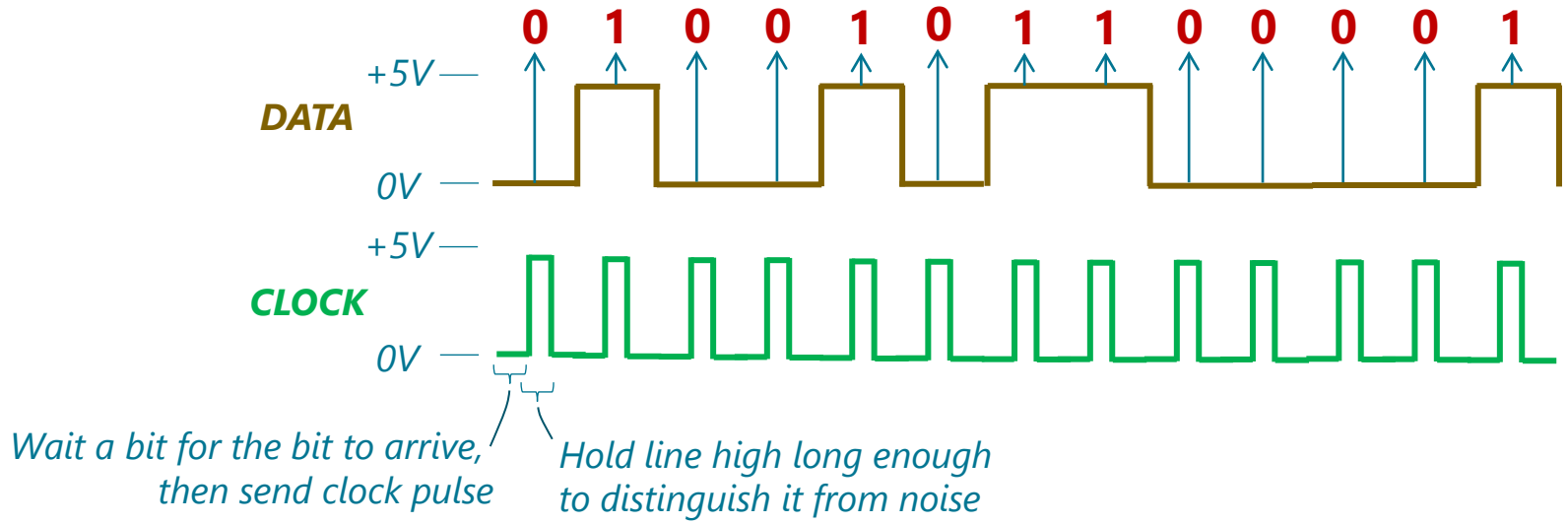
- Lower levels reduce power consumption, improve safety
- Higher levels reduce sensitivity to noise, loss over distances
- Specific values (15V, 12V, 5V, 3.3V) are common for historical reasons
  - Large existing market reduces component cost

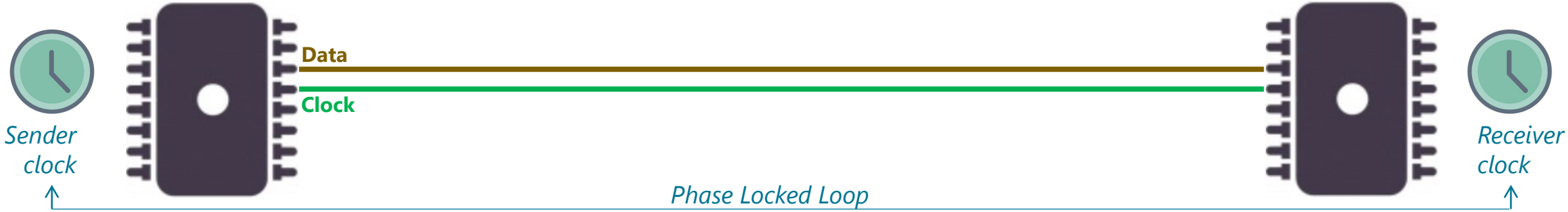
Two Chips Can Vary Voltage to Send Data



- One approach: raise voltage for a 1, lower it for a 0
  - Other side decodes: periodically sample voltage on wire
- Problem: clock synchronization

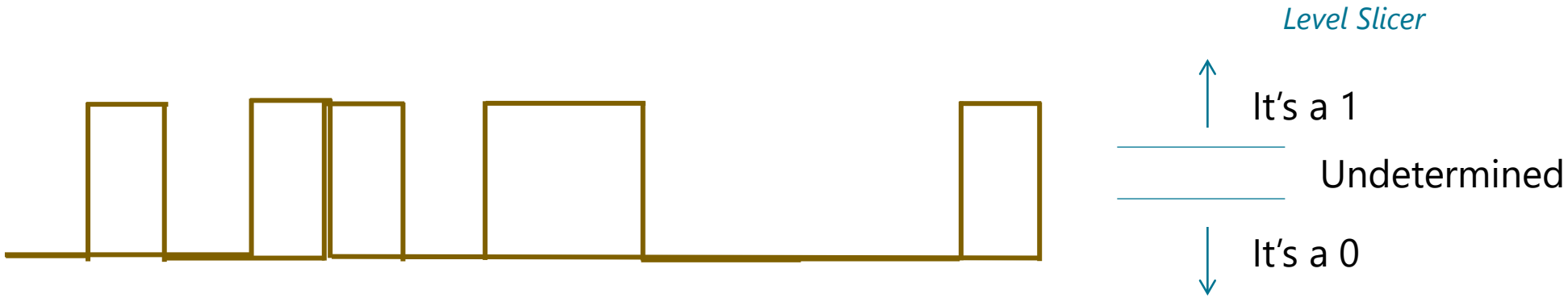
Two Chips Can Vary Voltage to Send Data





- **Another approach:** have a separate clock line
  - Use pulses to signal when receiver should sample line
- **Problems:** need additional wire (increases costs, increases faults because faults/noise in *either* wire cause loss)
  - Also need faster components (if clock transition rate exceeds that of data)

A Few Other Issues: DC Balance

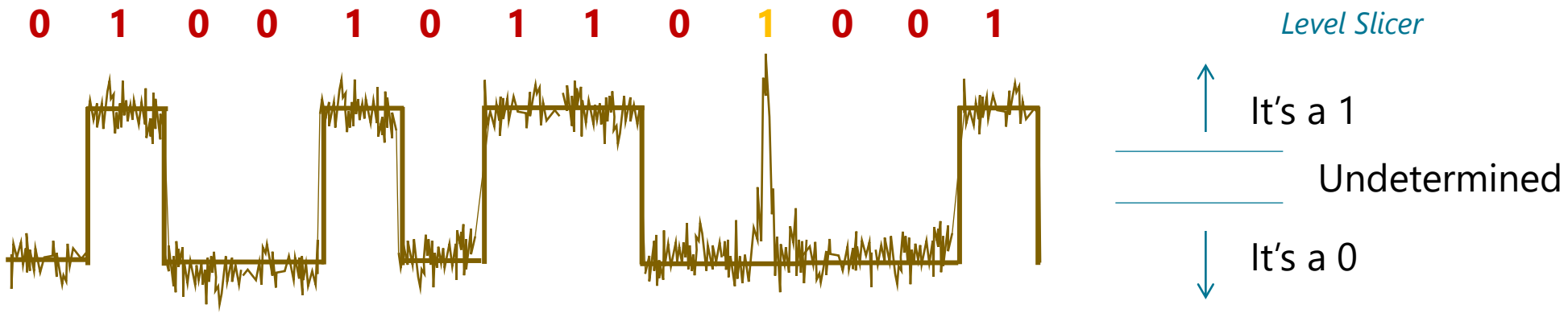


Unequal numbers of 0s and 1s can introduce bias in level detection circuitry

- In unbalanced channel, DC component shifts cause slicer to respond at level different to data average

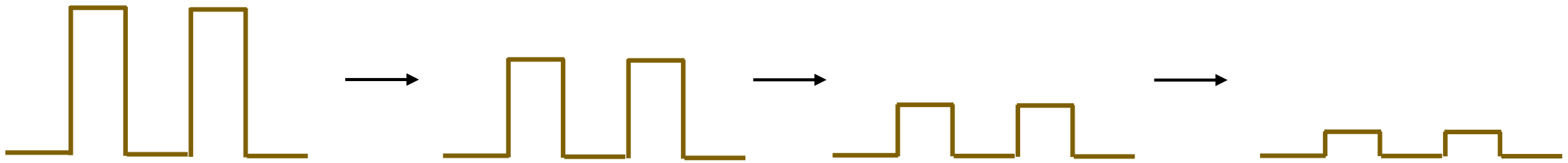
- Also makes it hard to galvanically isolate (separating two circuits due to noise/safety, e.g., with capacitance, EM waves, etc.)
- Also requires more transmission power (need to send DC to receiver)

A Few Other Issues: Electronic Noise



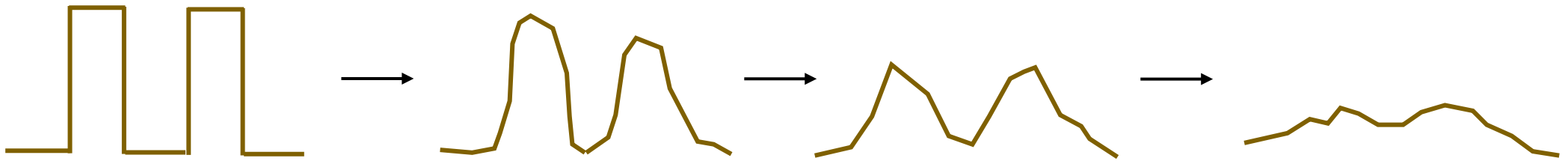
- May introduce false readings
- **Sources:**
  - Thermal noise: free electrons vibrate more at higher temperatures
  - Shot/flicker/burst/transit-time noise: noise from electrons traversing components
  - Coupled noise: noise leaked from adjacent wires/atmosphere/cosmic
  - Leakage noise: motors/speakers/bulbs/etc. induce voltage spikes), etc.

A Few Other Issues: Loss/Attenuation



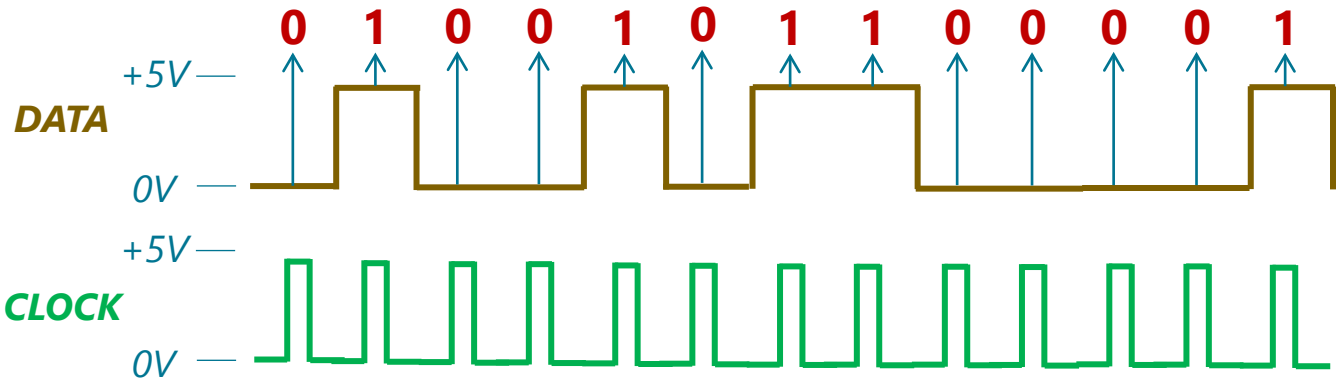
- Wires/components have resistance (impedance)
  - Reduces current and voltage. Amount proportional to length of wire
  - Larger-diameter cables/wires have less resistance per foot
  - Components also have impedance, more you traverse the worse it gets
- Impedances sum when connected in series, inversely sum with inverses when connected in parallel

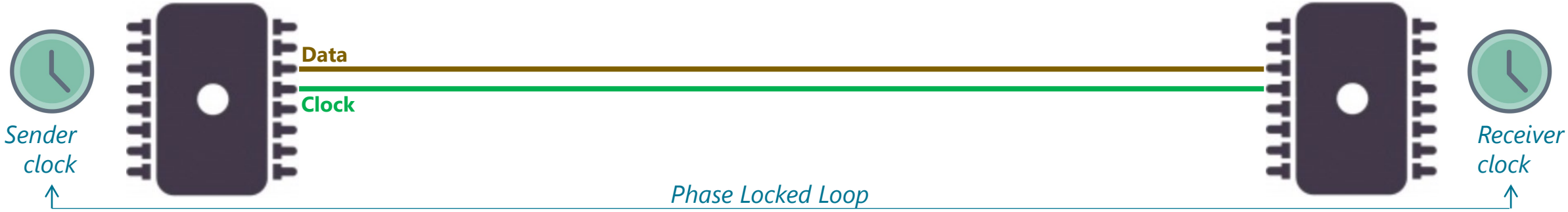
A Few Other Issues: Dispersion



- Not all electrons in the pulse go the same speed
- Bits melt into each other, making them hard to distinguish
- Some components have more dispersion than others (e.g., low-pass filters)

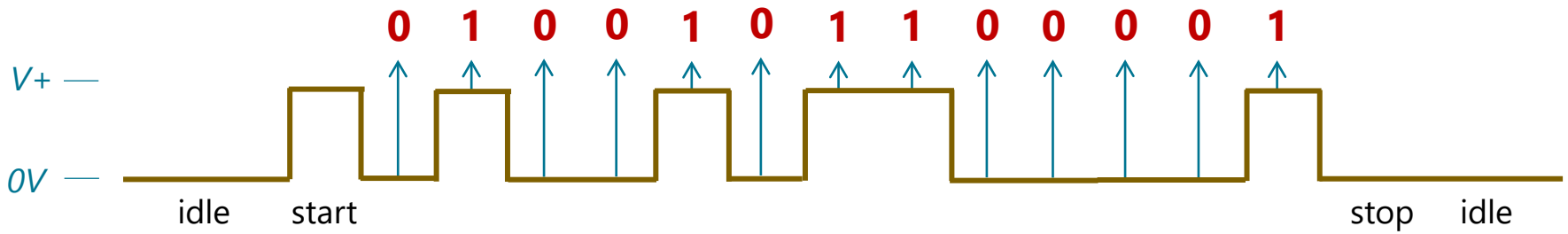
Ok, so can we at least get rid of the extra wire?





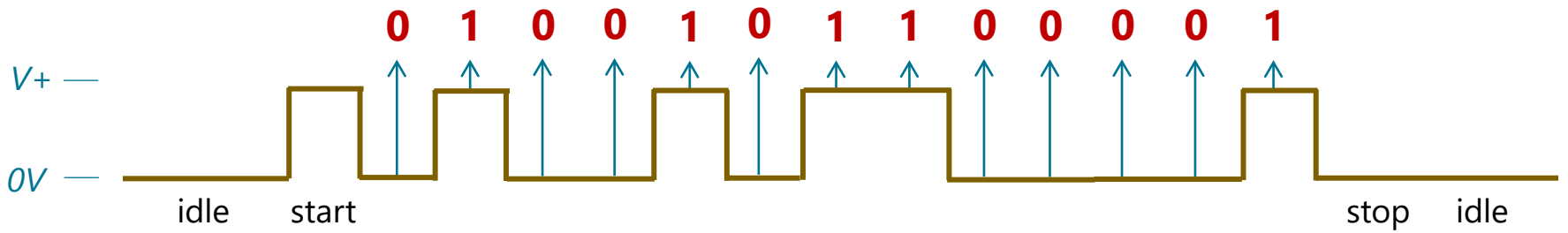
- Idea: receiver can sync clock off transitions
- Problem: What if not enough transitions?
  - Long runs of 1s or 0s
  - Clocks could get out of sync

Idea: Re-sync on Each Short Message (Asynchronous Communication)



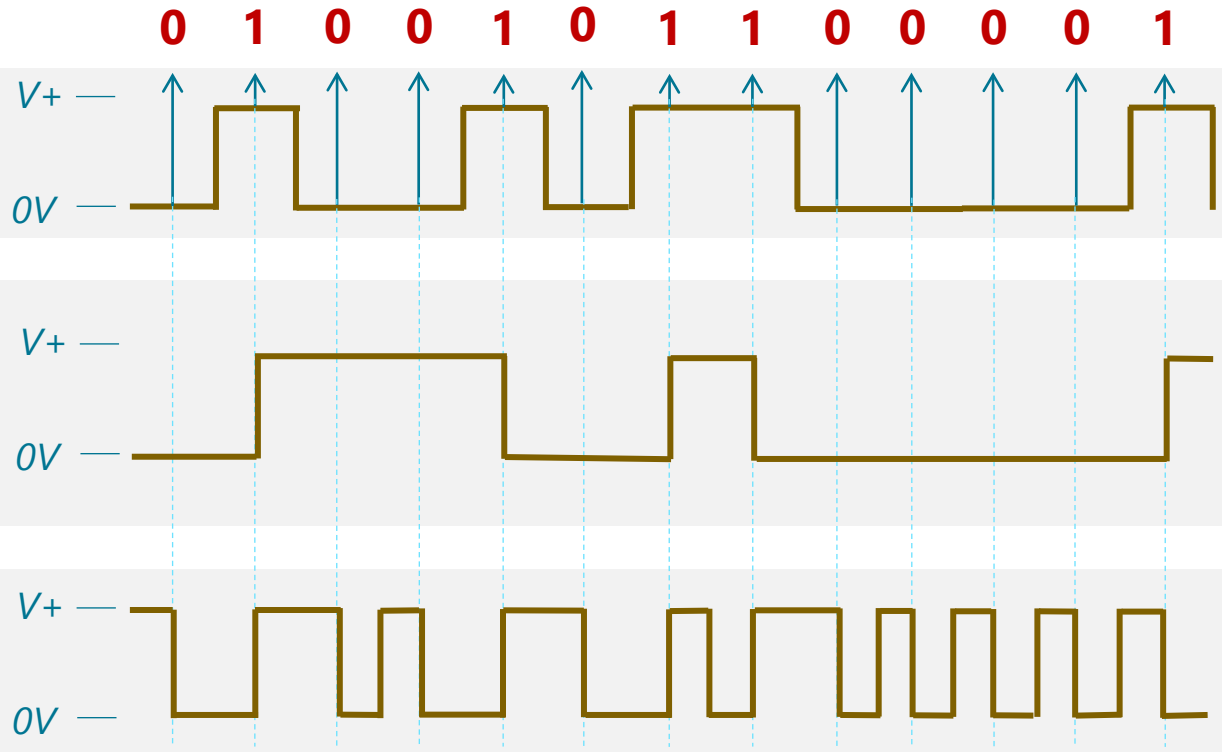
- Start and stop signals appended before/after each unit of transmission
  - Start signal used to set receivers' clock
  - Need to agree on parameters in advance (full vs. half-duplex, # bits per character, endianness, baud rate, whether to use parity, odd vs. even parity, # of stop bits)

Idea: Re-sync on Each Short Message (Asynchronous Communication)



- Advantages: doesn't require continuous synchronization → cheaper hardware, faster session setup
  - Good for applications where messages are generated at irregular intervals: user input (e.g., keyboards), sensors, etc.
- Downsides: large overhead (many bits just used for control)
- Used in: RS-232/UART/PPP/etc.

Another Idea: Force There to Be More Transitions



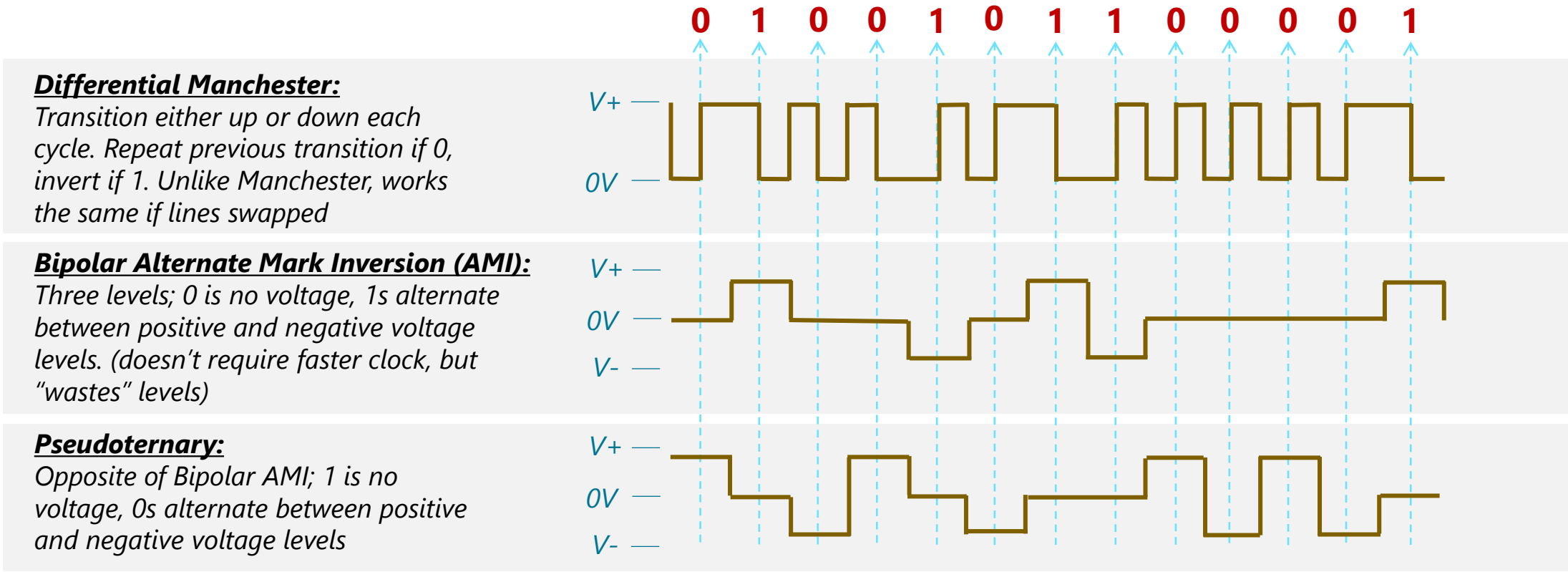
NRZ [Non-Return to Zero]:
High voltage for 1, low voltage for 0 (what we've been seeing)

NRZI [Non-Return to Zero Inverted]:
Transition for 1, no transition for 0. NRZI is a "differential encoding," as value depends on preceding bits (increases transition density)

Manchester (IEEE 802.3):
Up transition for 1, down transition for 0 (further increases transition density, but requires faster clock)

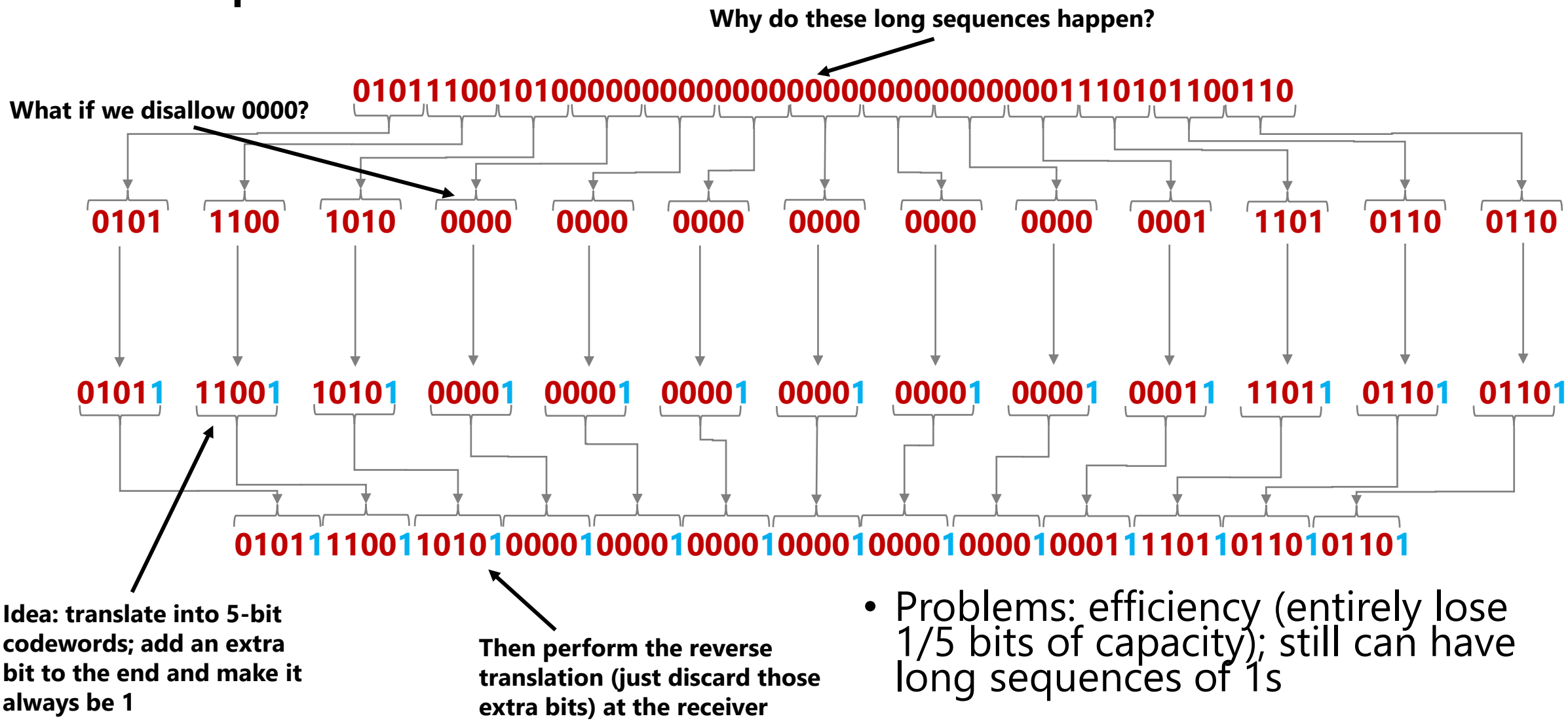
- Use transitions rather than voltage levels to encode data
  - Cause more cycles → improve clock synchronization
  - Transitions are also easier to detect than levels

A Few Other Approaches



- These are pretty good but can we do better?
  - Manchester/Differential Manchester requires 2x clock
  - Bipolar AMI/Pseudoternary requires more levels

Idea: Map onto Codewords with Nice Properties



- Problems: efficiency (entirely lose 1/5 bits of capacity); still can have long sequences of 1s

Better Approach: 4B/5B (Choose Codewords Smarter)

- Encode every 4 consecutive bits as a 5-bit word

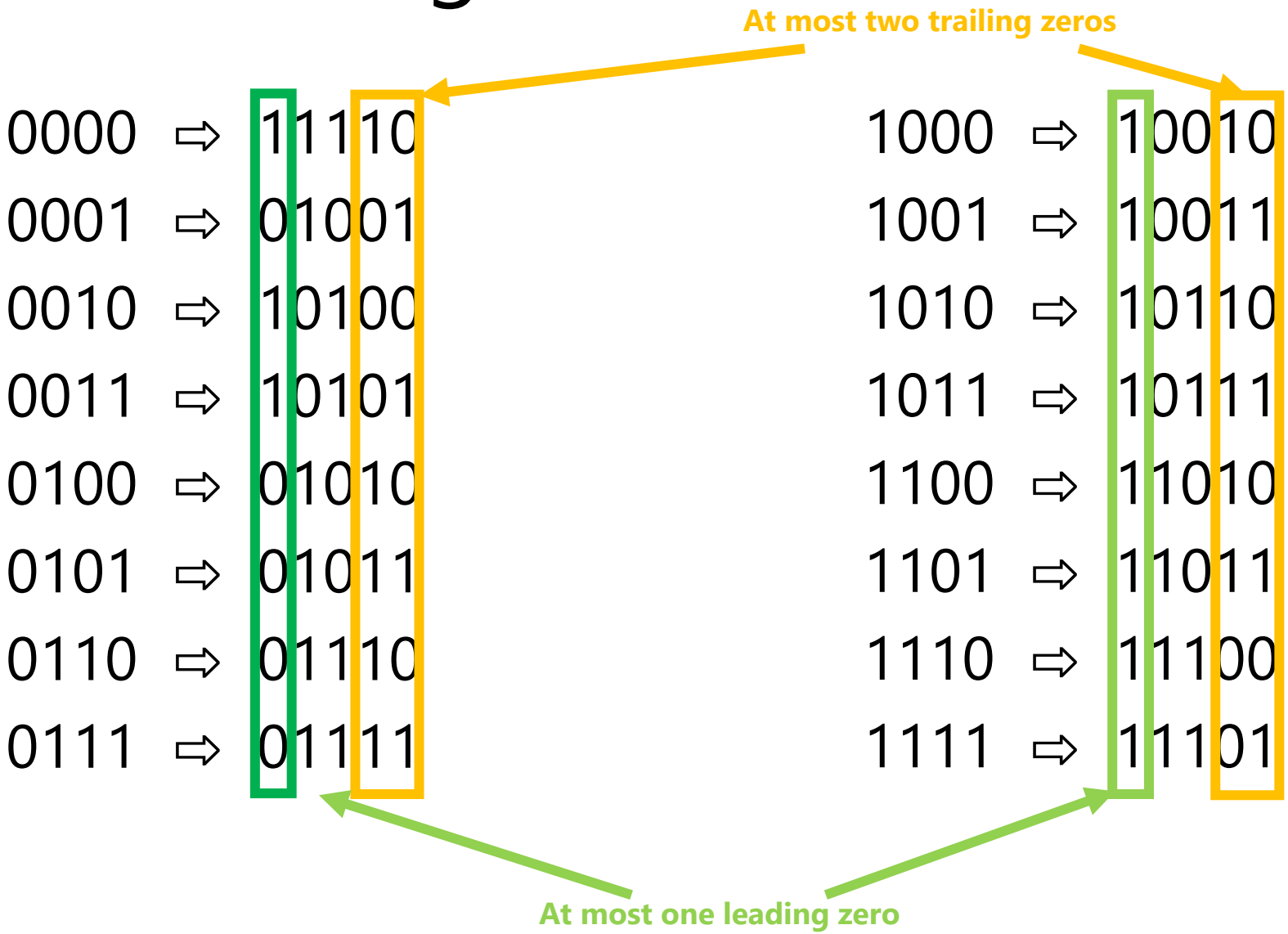
- Codewords
  - At most one leading 0, at most two trailing 0s
  - Never more than three consecutive 0s
  - Transmit with NRZI

Better Approach: 4B/5B (Choose Codewords Smarter)

- 16 of 32 possible codes used for data, others used for command characters
  - Ensures at least two transitions for each code

- Variants
  - 8B/10B (used in digital audio – compact discs, DAT tapes, etc.)
  - 64B/66B (used in 10Gig Ethernet)
  - 128B/130B, 128B/132B (used in PCI Express 3.0 and USB 3.1)

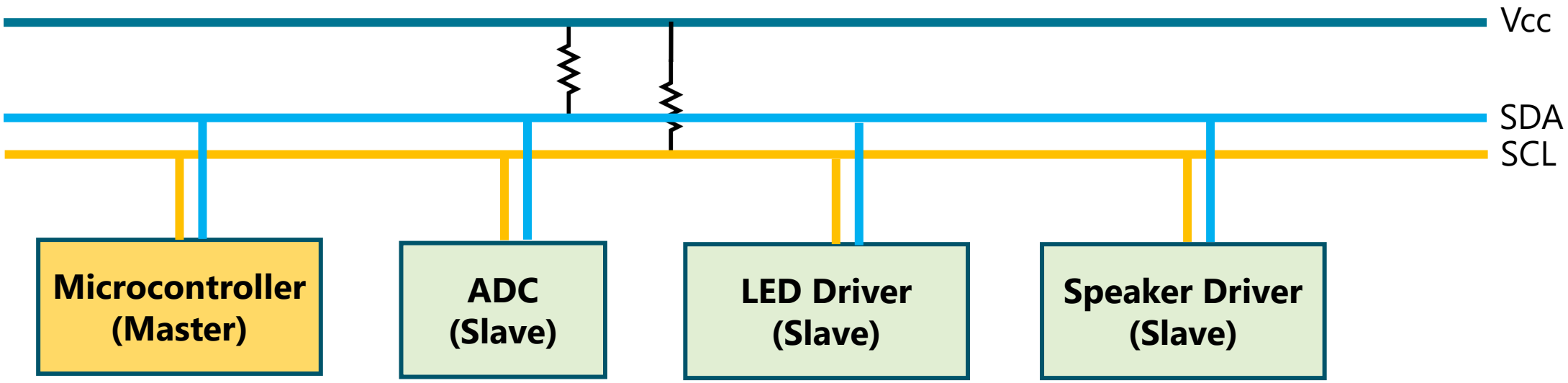
4B/5B Encoding Table



Where is this stuff used?

- Different encoding schemes are used in different settings
- Advanced/Complex encodings used for high-speed links between computers
  - Where it's worth investing in more complex circuitry
  - See more Manchester, 64B/66B, etc.
- Between chips on a PCB you care more about low power and cost
  - Simpler techniques, easier to run shared clock lines everywhere
  - See more asynchronous, clocked encodings, etc.

Connecting ICs with the I2C Protocol

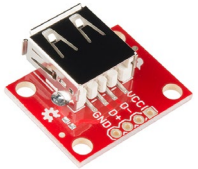
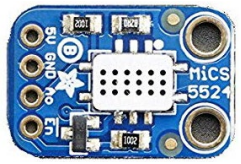


- Easy to use, very expandable, serial bus widely used to connect ICs
  - Reduces pinouts, transfers data/commands
- Address-oriented scheme, where addresses are part of packets that you send
  - Handles many devices; each device has an address (can solder jumpers to change address)
  - Message-based communication (unlimited length), 0.1-5Mbps, synchronous
- Multi-master/Multi-slave architecture: master generates clock, initiates communications

Other Kinds of Buses

- **Serial Peripheral Interface (SPI)**
  - Max # devices very limited (chip-select for each chip on the bus)
  - Significantly faster than I2C (25Mbps), works over longer cable runs
  - Used for, eg. Connecting microcontroller to SD/microSD card
- **UART** (RS-232/422/485/etc., aka Serial Port)
  - Asynchronous (no shared clock)
  - Parameters: baudRate, parity, stopBits, dataBits (e.g., 115200, none, 1, 8)
  - Max speed 8Mbps
- **GPIO**
  - Not really a protocol – turning pins on and off, defines threshold voltages
  - Used for switches, buttons, motion detectors, LEDs, turning on/off appliances
  - Max speed 1kHz

Example Breakout Boards

Type	Name	Picture	What it does	Internal Components	Other boards in this category
Actuator	Allegro A4988 Stepper Motor Driver		Takes in logic input and uses that to control a higher-amp input for driving a stepper motor. Includes current limiting, short-circuit/over-current/over-temperature protection. Operates from 8-35V and supports up to 1A without cooling (2A with sufficient additional cooling). Five different resolutions (full, half, quarter, 1/8, 1/16).	A4988 IC	Clock generators, Audio FX sound boards, storage/microSD, NVRAM
Display	SparkFun Color LCD Breakout Board		Uses logic input to write graphics onto and control backlight of color LCD display. Display is 132x132 pixels @12 bit color, same display from cell phones.	Nokia 6100 color LCD, Philips PCF8833 graphics controller, status LEDs, two pushbuttons.	LED matrix backpacks, LED strip controllers, LCD displays, touchscreen controllers
Connector	SparkFun USB Type A Female Breakout		Splits out female USB type A connector's VCC, D-, D+, and GND pins to a standard 0.1"-width header. Good if you need USB but don't want to deal with soldering tiny connectors.	USB Type A female connector	Other types of USB (microB, C, etc), audio connectors, HDMI/DVI coders, USB mouse/keyboard controller Wireless 802.11/LoRA/etc.
Sensor	Adafruit MiCS5524 CO, Alcohol and VOC Gas Sensor Breakout		MEMS sensor breakout to monitor levels of various sorts of gases. Useful for indoor carbon monoxide and natural gas leakage, alcohol breath checker, early fire detection. Sensitive to 1-1000ppm. Can't tell you which gas it has detected.	SGX NICS-5524 gas sensor	Sensors for Oxygen, CO2, methane, PM2.5/PM10, humidity, barometric/altitude, windspeed, temperature, human breath analysis

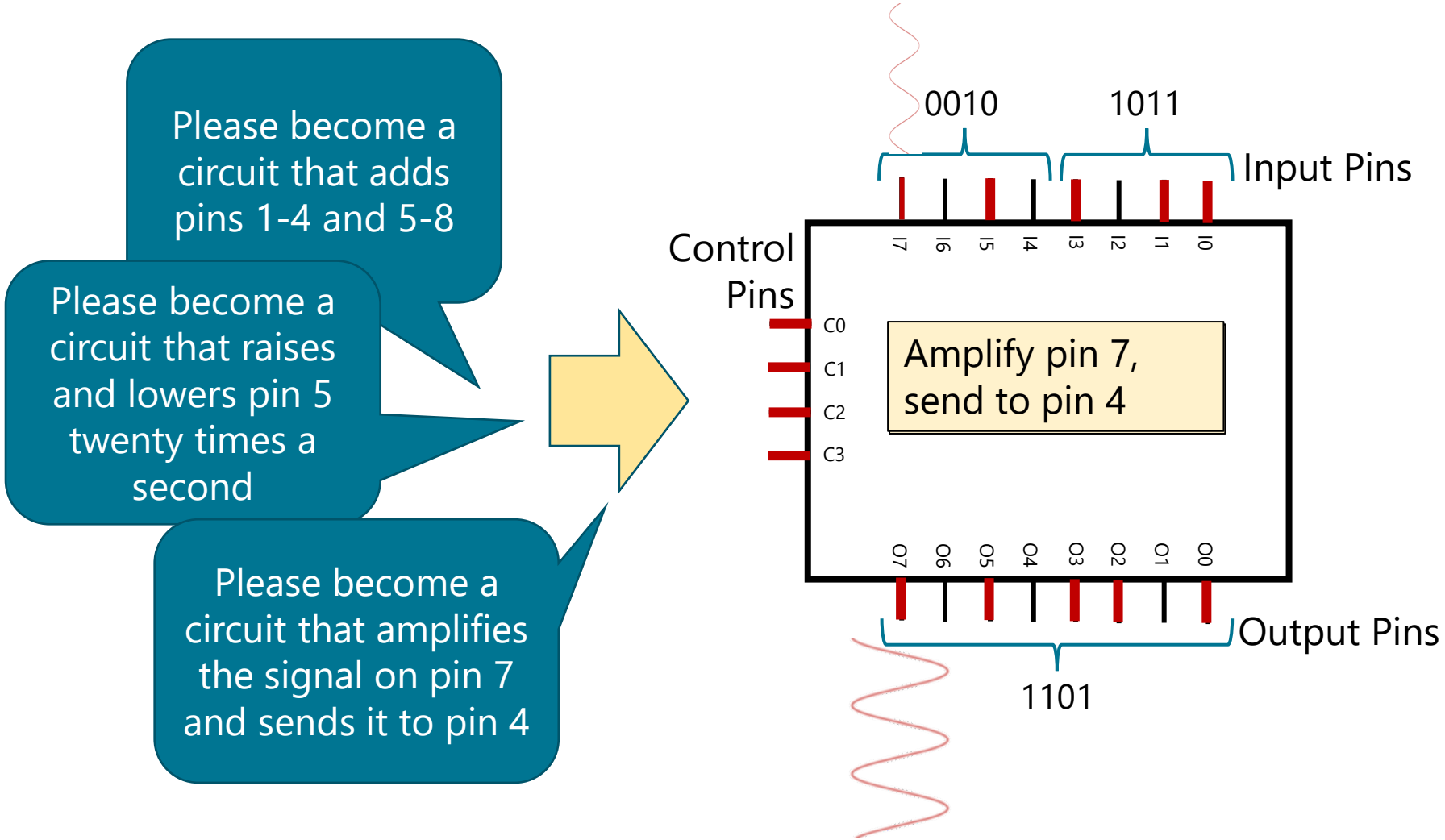
What if we want to build something flexible?

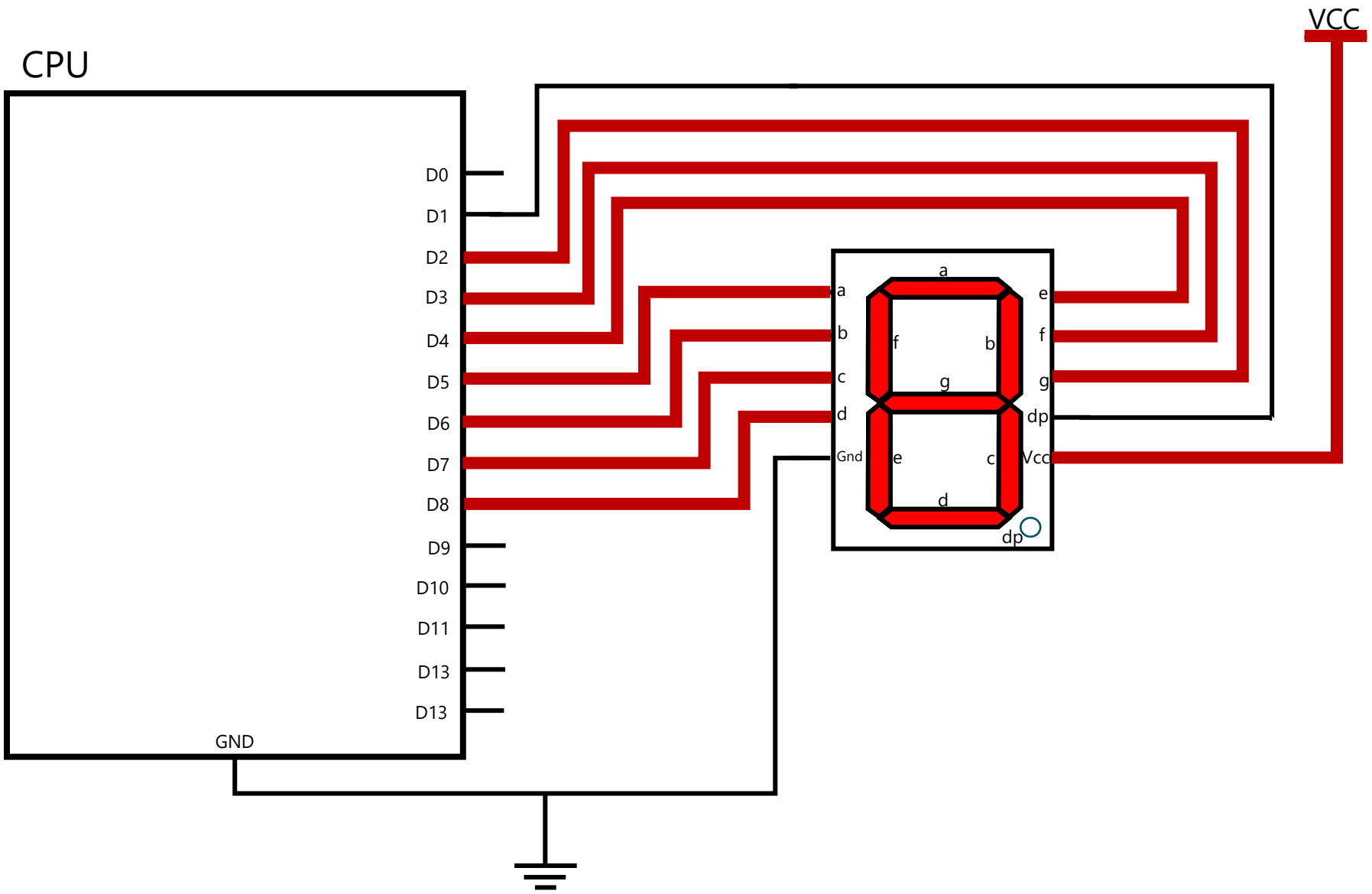
- Circuits are good for very specific, single-purpose functions
  - Increase voltage from 3.3V to 5V, drive a motor, etc.

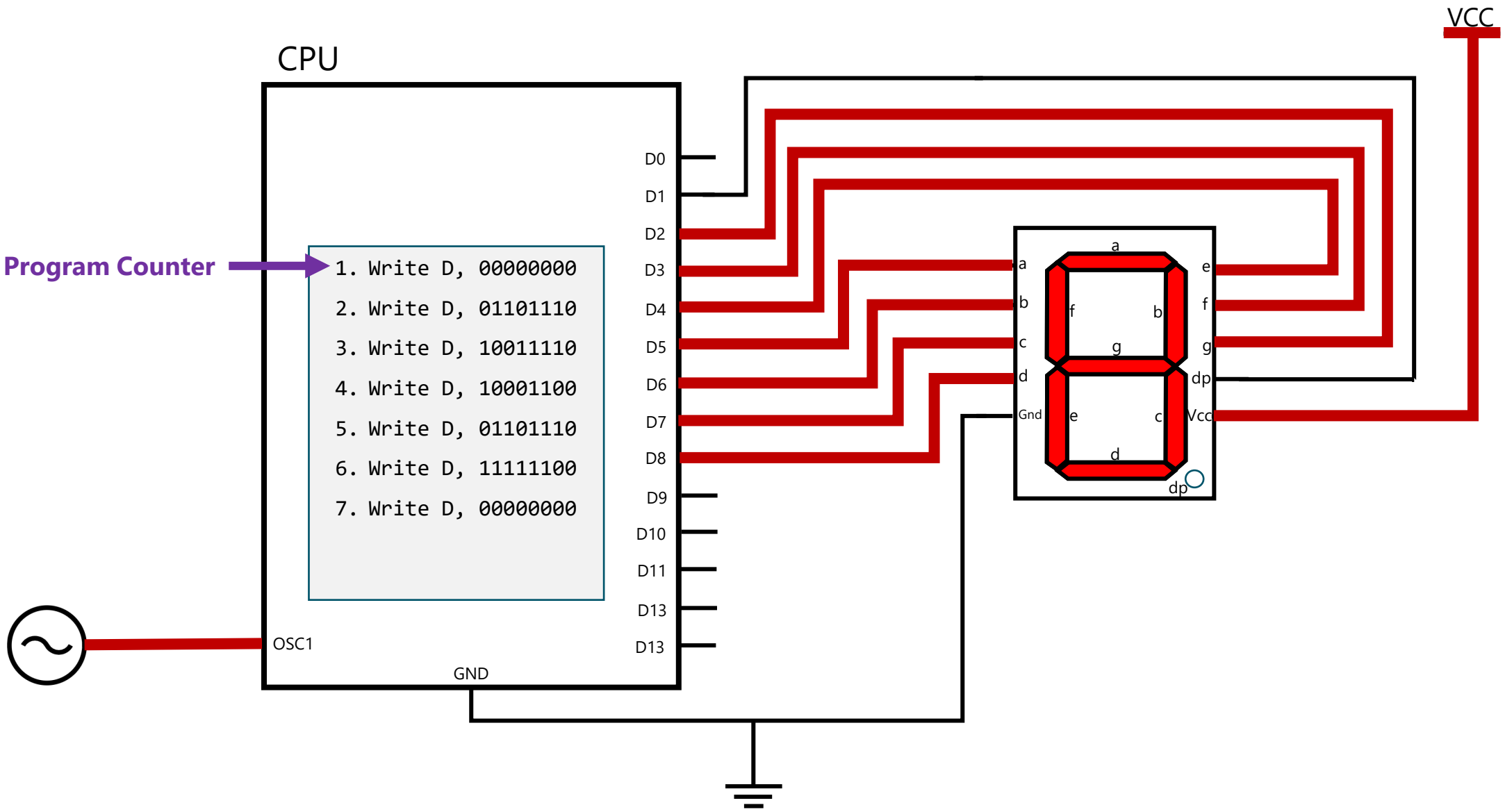
- But what about
  - A car that recognizes human shapes?
  - A drone that flies a programmable path?
  - A thermostat that displays updatable pictures?

- Problem: need to deal with many contingencies

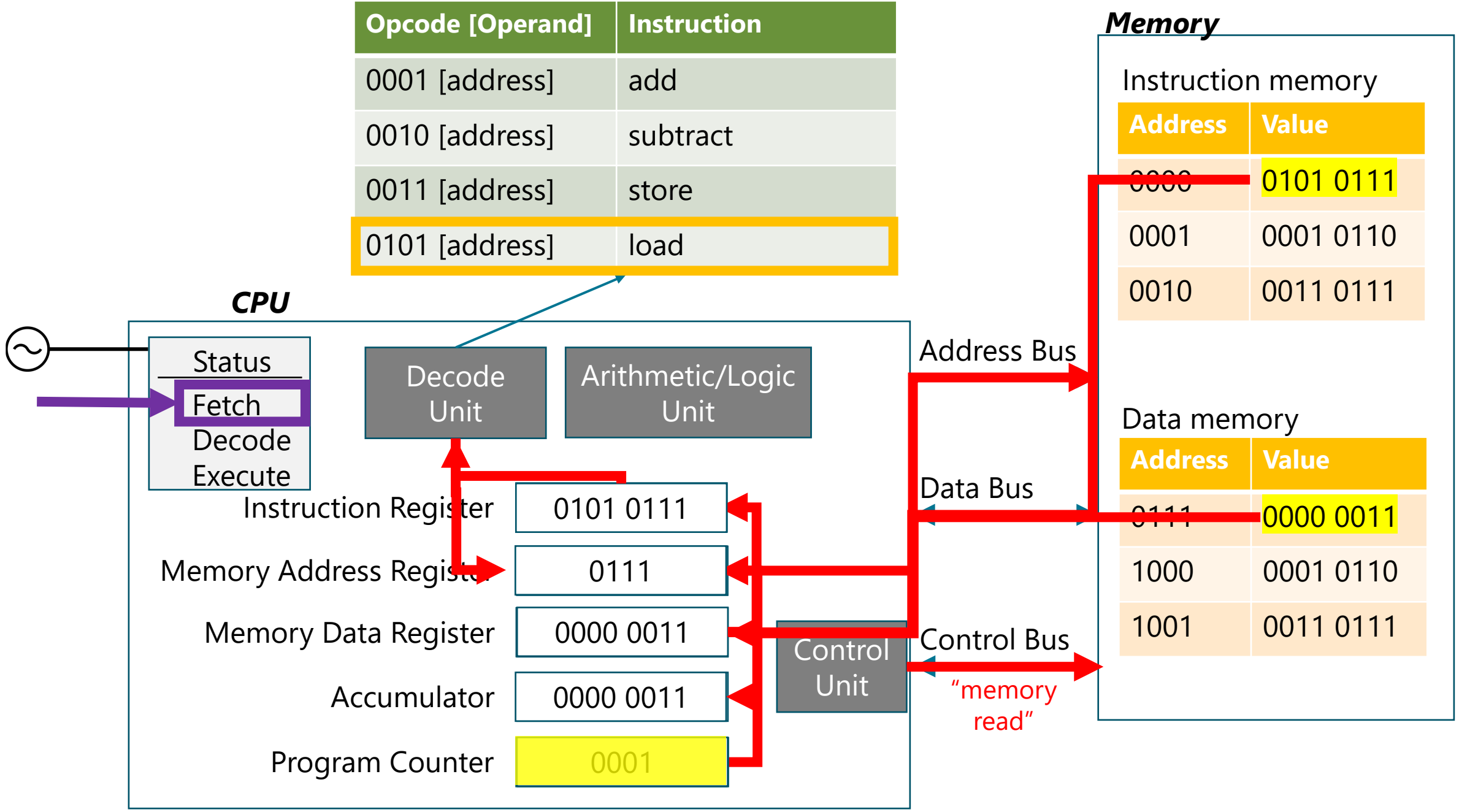
Idea: A “Programmable IC”




CPU
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D13
D13
GND
VCC
a
b
c
d
Gnd
e
f
g
dp
Vcc


CPU
Program Counter
1. Write D, 00000000
2. Write D, 01101110
3. Write D, 10011110
4. Write D, 10001100
5. Write D, 01101110
6. Write D, 11111100
7. Write D, 00000000
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D13
D13
OSC1
GND
VCC
a
b
c
d
Gnd
e
f
g
dp
Vcc

Opcode [Operand]	Instruction
0001 [address]	add
0010 [address]	subtract
0011 [address]	store
0101 [address]	load



Benefits of CPUs

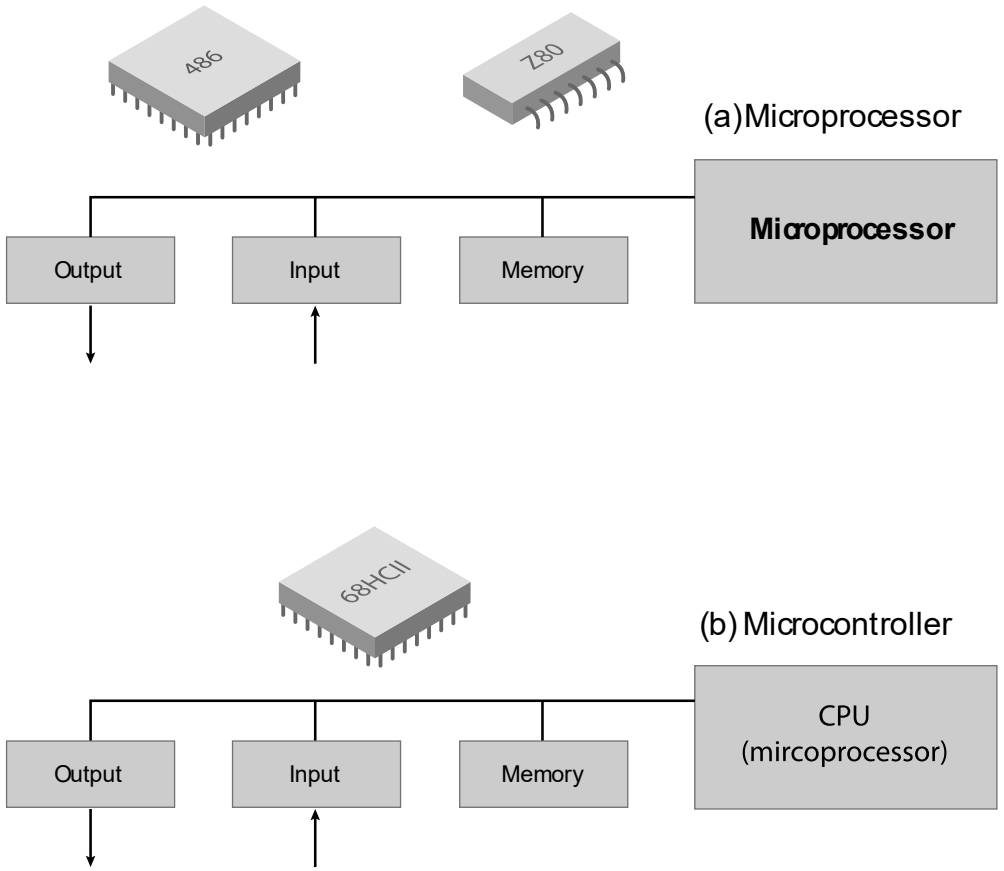
- Programmable circuits allow highly customizable logic in hardware
  - Enables explosion of sophistication
- Focused on digital applications
  - Easier to consolidate into a compact unit
- CPUs are great for many applications
  - Extremely flexible, high-speed
  - Can run operating systems, machine vision, NLP, etc.

Downsides of CPUs

- However, hard to make them very cheaply
  - Hard to make general, powerful platforms also cheap
  - What do we do for ultra-cheap applications?

- Idea: make special-purpose CPUs
  - Architectures targeted toward specific uses
  - Only the capabilities you need → reduced costs

Microcontrollers



A microcontroller is like a CPU, but:

1. Lower-end design
   - Lower-performance (lower power usage/clockspeed)
   - Simpler (limited instruction set)
2. Consolidated design
   - Memory, oscillator, I/O controllers, watchdog timers, are within chipset
3. Special on-chip functions (peripherals)
   - Onboard DAC/ADC, PWM, NCO, programmable logic

How to Choose a Microcontroller

- **Core design:** vendor-proprietary vs. third-party
  - 32-bit ARM core becoming ubiquitous, Intel 8051 also used
- **High- vs. low-end:**
  - Microcontrollers come in “families”: some have hundreds of models to choose from, some have much fewer. Smaller: economy of scale; larger: select what you need
  - Sunken cost in choosing a family
- **Package availability**
  - Wafer-level CSP package that’s 2x2mm? Larger DIP package?

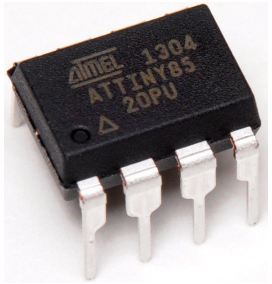
How to Choose a Microcontroller

- **Peripherals:** multiple interrupt channels, DMA, internal clocks, power configuration control options, etc.

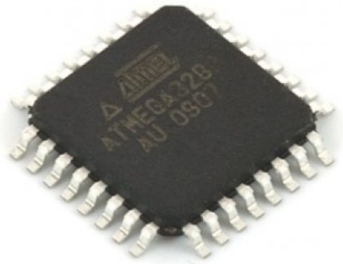
- **Development experience:**
  - What kinds of IDEs/SDKs/compilers/emulators/dev boards/debuggers are available? How much do they cost? Are they cross-platform?

- **Performance:** power efficiency, clock cycle efficiency, code size

Popular Microcontroller Cores



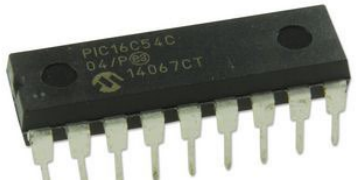
ATMEL TINYAVR
(low-end)



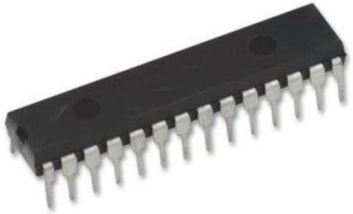
ATMEL MEGAAVR
(high-end)

ATMEL AVR

- RISC design. many single-cycle operations, 2-cycle multiply
- Strengths: many registers (32), good cycle efficiency
- Downsides: only one interrupt priority, slow clock speed compared to competition (up to 32 Mhz)
- Example chips: ATtiny85 (Arduino), ATtiny1616, ATmega328



Microchip PIC16
(lower-end)

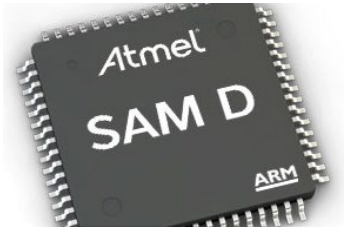


Microchip PIC24
(higher-end)

MICROCHIP PIC

- RISC design. 4T machine - 4 cycles fetch (plus 4 cycles execute)
- Strengths: many peripherals, historically very popular
- Downsides: only one register, no stack, lower-end chips have no on-chip debugging support, slow flash-load times. Falling out of favor for students/hobbyists.
- PIC32 uses 3rd party core (MIPS M4K)
- Example chips: PIC10, PIC12, PIC16, PIC18, PIC24, PIC32

Popular Microcontroller Cores



ATMEL SAM D10

Infineon XMC

ARM Cortex-M0

- 32-bit RISC. Entry level to ARM architecture. Rapidly gaining market share in <$1 market
- Strengths: 32 interrupt vectors, 4 interrupt priorities, full support for runtime exceptions. 13 registers.
- Downsides: low code density, lack of compatibility across vendor peripherals, complex peripherals, slow (12 cycle) interrupt latency
- Example chips: Infineon XMC-1100, Cypress PSOC 4000S

Nuvoton 8051

Infineon XC8XX

INTEL 8051

- Variable-length CISC instruction set, 32 registers, 2 interrupt priorities, 64KB program/RAM addressability. Widely-cloned architecture.
- Strengths: fast interrupts (good for real-time eg USB webcams and audio DSPs), fast clock speeds, historicallyextremely popular
- Downsides: only 8-bits, hard to compete with cheap 32-bit controllers. Simplicity can be a detriment with modern compilers. Falling out of favor.
- PIC32 uses 3rd party core (MIPS M4K)
- Example chips: STCmicro STC8, Silicon Labs EFM8, Nuvoton N76

Other Kinds of Programmable Circuits

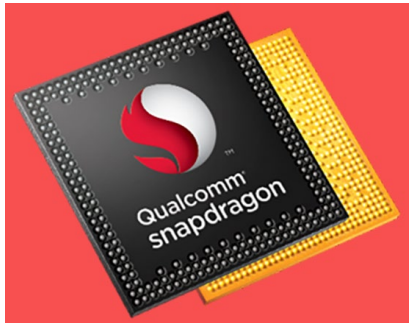
CPUs

Microcontrollers

FPGAs

ASICs

GPUs



System on Chip (SoC)

FPGAs (Field-Programmable Gate Arrays)

- FPGAs consist of a bunch of:
  - Programmable logic blocks – can be configured into logic gates (AND/XOR/NOT/etc.) or complex combinatorial functions
  - Interconnects to hook them together
- Many FPGAs can be reprogrammed after deployment (in the "field")
  - Enables flexible "reconfigurable computing"
  - Can fix vulnerabilities/bugs post-deployment

FPGAs (Field-Programmable Gate Arrays)

- Can write whatever circuits you want → parallel execution
  - Can be more power efficient than single-threaded microcontrollers for some applications (ML/AI/data processing)
- However, current tech targets high end market → expensive (>$500)
- Key players: Xilinx (49% of market), Altera (40% of market)
  - Intel (see Cyclone 10)

Programmable Circuits

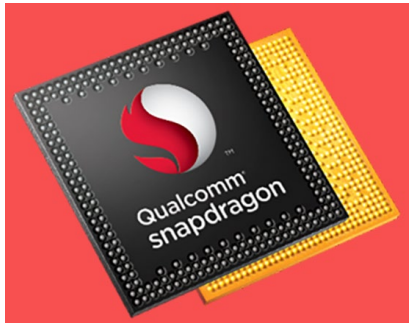
CPUs

Microcontrollers

FPGAs

ASICs

GPUs



System on Chip (SoC)

ASICs (Application-Specific Integrated Circuit)

- Burn your design into silicon
  - 100s of millions of logic gates
  - Can include ROM, RAM, EEPROM, flash memory, and other large building blocks
- Core logic not field-programmable, hard to update designs
    - May contain some reconfigurable logic

ASICs (Application-Specific Integrated Circuit)

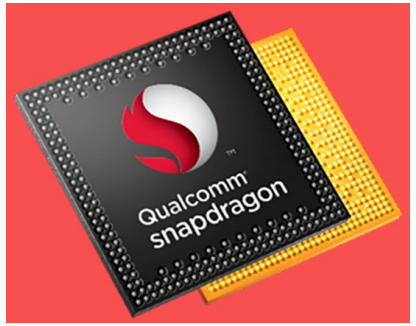
- ASICs (and FPGAs) used to implement CPUs/microcontrollers/etc.
  - Typical design: IP core CPU, digital signal processor, communication interfaces, memory
- Non-recurring engineering (NRE) of millions of dollars
  - Best for large production volumes, shared functionalities
- Key players: TSMC, GlobalFoundries, Samsung Semiconductor, Texas Instruments

GPUs (Graphics Processing Unit)

- Designed to rapidly manipulate and process images and video
  - Many threads, like a highly parallel CPU (e.g., 480-1536 stream processors)
- Commonly used in mobile phones, personal computers, game consoles, self-driving cars, drones, automated surveillance, etc.

GPUs (Graphics Processing Unit)

- Contains hardware implementations of important graphics functions
  - Texture mapping, rendering polygons, rotation/translation of vertices, shading, special effects (chroma keying, bokeh, volumetric lighting)
- Non-graphics applications
  - Applications that can be parallelized work well: processing hashes (bitcoin mining), matrix operations, audio processing, machine learning
- Key players: NVIDIA, Intel, AMD



SoCs (System on Chip)

- SoC=system on chip
- Basically pairs a CPU/microcontroller with other functions, on one chip
  - Tighter integration → better reliability, power savings, lower cost
- Paired functions: AI acceleration, machine vision, wireless communications, digital camera hardware/firmware, etc.

Emerging Programmable Circuits

- **Vision Processing Unit (VPU)** – accelerates computer vision – Microsoft HoloLens, Movidius Myriad X, NVIDIA VLIW Vision Processor

- **Neural Network Processor (NNP)** – accelerates neural network machine learning – Google (Tensor Processing Unit), Intel (Nervana), AWS (Inferentia), Apple (Neural Engine)

- **Machine Learning Processor (MLP)** – general programmable platform for machine learning – ARM (Trillium), IBM (Power9)

Open-Source Hardware Logic

- Don’t create complex hardware designs from scratch!

- Highly advanced complexity in hardware → takes a lot of time and investment to create
  - Re-use cuts product development times and creates better products
  - Open-source organizations provide free IP cores
    - E.g., www.opencores.org

Open-Source Hardware Logic

- **Cell libraries:** libraries of logical sub-components that can be plugged together

- **IP cores:** primitives purchased from a third-party
  - May be provided as hardware-description language ("**soft macro**")
  - May be provided as a fully routed design ("**hard macro**")
  - ARM *only* sells IP cores ("**fabless manufacturer**")

How to decide which to use?

Technology	When to use	Example applications
Microcontrollers	When you need to get really small, really power-efficient, or really low-cost, and don't need much compute power.	Main compute for Apple Watch Series 3 (ST Microelectronics ST33G1M2 32 bit MCU)
CPUs	When you need to run diverse applications or deal with larger memory or compute challenges.	Nest thermostat (Arm Cortex A8), Amazon Echo Show (Intel Atom x5)
FPGAs	When you benefit from parallelism or have need to reprogram logic in the field, and are at low production volumes. More expensive than ASICs.	Many (proprietary); smart automotive (eg plug-in hybrid), aerospace applications, prototyping ASICs.
ASICs	When you need and can afford to pay millions to fabricate your own chip. High production volumes.	Many (proprietary); edge devices, sensor/actuator control, etc.
GPUs	When your application is data-intense, and graphics-oriented, or can be transformed into a highly-parallel many-core problem (ML/AI).	Tesla Autopilot 2.0 ECU (Nvidia PX2 platform, which uses NVIDIA GP106 GPU)
SoC	When you need general and application-specific compute but also a compact size, your domain is one where there is an SoC and prices are low.	Samsung Galaxy smartphone (Exnyos 8 Octa SoC), Apple iPhone XS (Apple A12 Bionic ARM-Based)

Example Products

CPUs

Nike Hyperadapt
(self-lacing bluetooth shoes)
ARM Cortex M4 CPU

LG E8
(55-65" OLED TV)
LG α9 CPU

Amazon Echo Show
(smart speaker with screen)
ARM Cortex A8 CPU

Moverio BT-300
(smart glasses)
Intel Atom x5 CPU

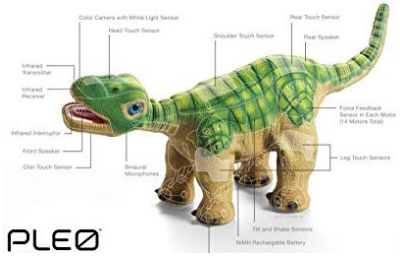
Fitbit Ionic
(GPS smartwatch)
ARM Cortex A8 CPU

Microcontrollers

HTC Vive
(VR Goggles)
ARM Cortex M0

Parrot AR.Drone 2.0
(Drone)
Microchip PIC24

Honeywell Water Alarm
(Freeze/Leak detector)
Microchip PIC12



Pleo Smart Dinosaur
(AI Toy)
Toshiba TMP86FH47 8-bit

Amazon Fire TV
(digital media player)
TI MSP430F5435A

Memory/Storage

- No “universal memory” (yet)
  - So best way to build something is use a mix of technologies
- Usage characteristics of memories:
  - May be easily writeable (RAM) or hard to write more than once (ROM)
  - May lose contents when not powered (volatile), or may retain contents when powered off (non-volatile)
- Performance characteristics of memories:
  - Max erase cycles, cost per byte, speed, density, heat, energy efficiency

Memory Technologies

Type	Volatile?	Writeable?	Speed	Cost per byte	Notes
Static random-access memory (SRAM)	Yes	Yes	Fast	Expensive	Commonly used in CPU caches. Can reduce power consumption when idle.
Dynamic random-access memory (DRAM)	Yes	Yes	Moderate	Moderate	Requires constant power to for external memory refresh circuit.
Masked ROM (Masked read-only memory)	No	No	Fast	Cheap	Contents programmed by IC manufacturer during lithography process. More compact than any other kind of memory, but very high 1-time masking cost.
Electrically-Erasable programmable ROM (EEPROM)	No	Yes	Fast to read, slow to write	Expensive	Commonly used as firmware. Common to use EEPROM during development and switch to Mask ROM when design is finalized.
Flash memory	No	Yes	Fast to read, slow to write	Moderate	Type of NVRAM/EEPROM. Found in memory cards, flash drives, SSDs. Limited write cycles.
NVRAM	No	Yes	Fast	Expensive	Ferroelectric RAM is main type to enter production.

IoT Platform Design

How can we design things out of these components?

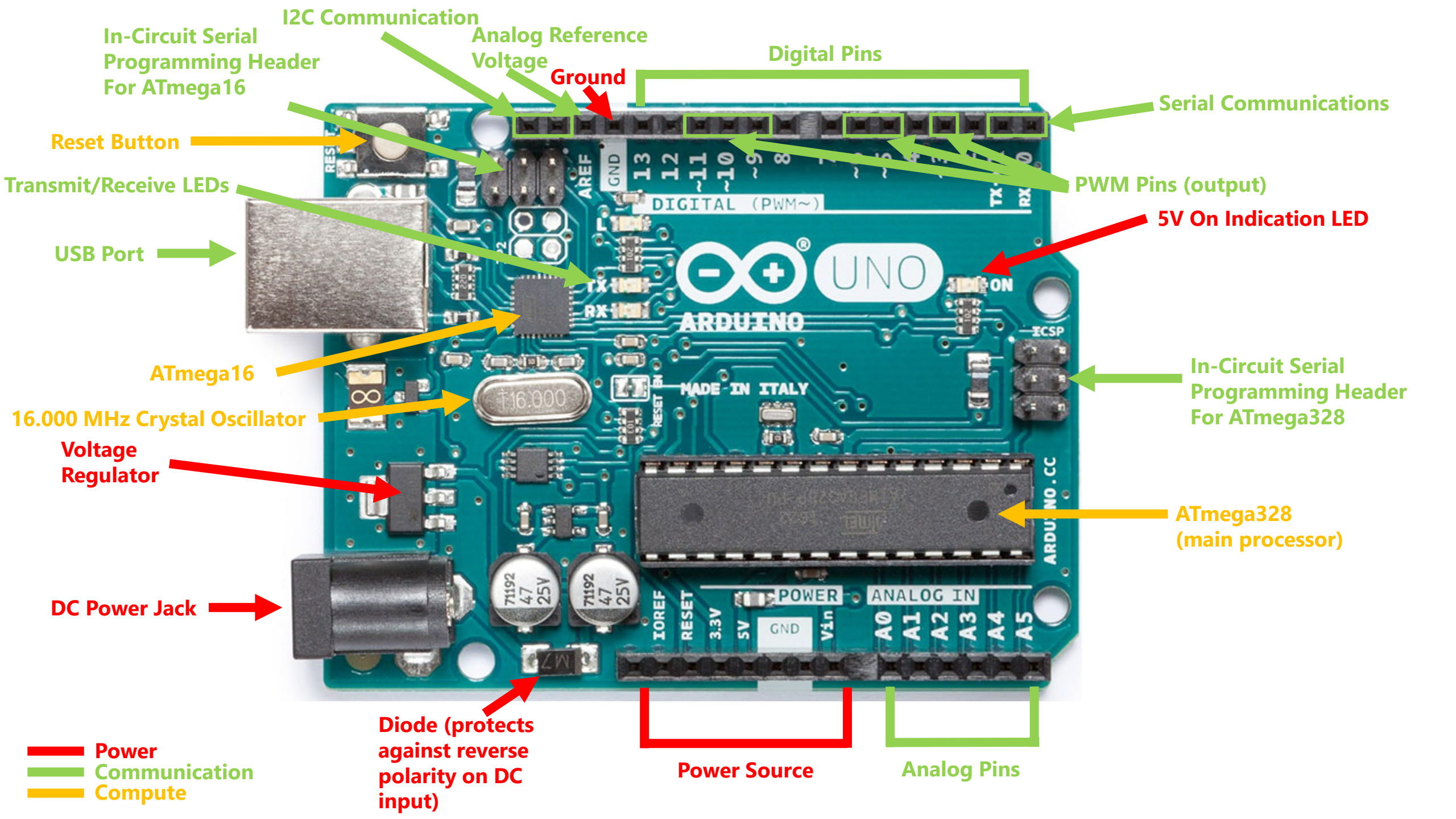
- You now know many components you can use
- You now know how to connect them together with wires
- But how do you build actual physical things?
- Next: let’s study some real platforms to get a sense of that
  - Let’s start with the Arduino platform

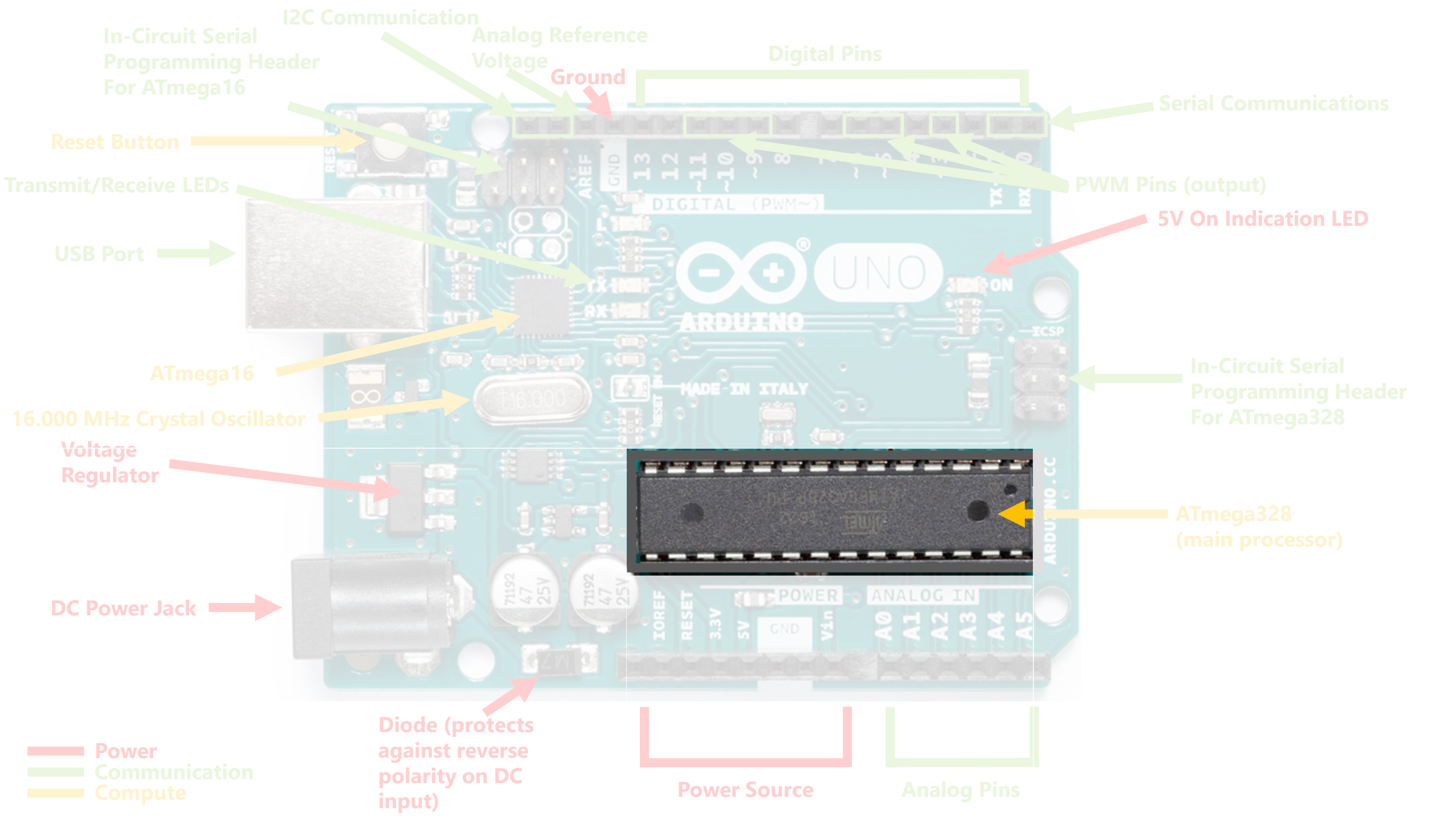
Arduino Platform

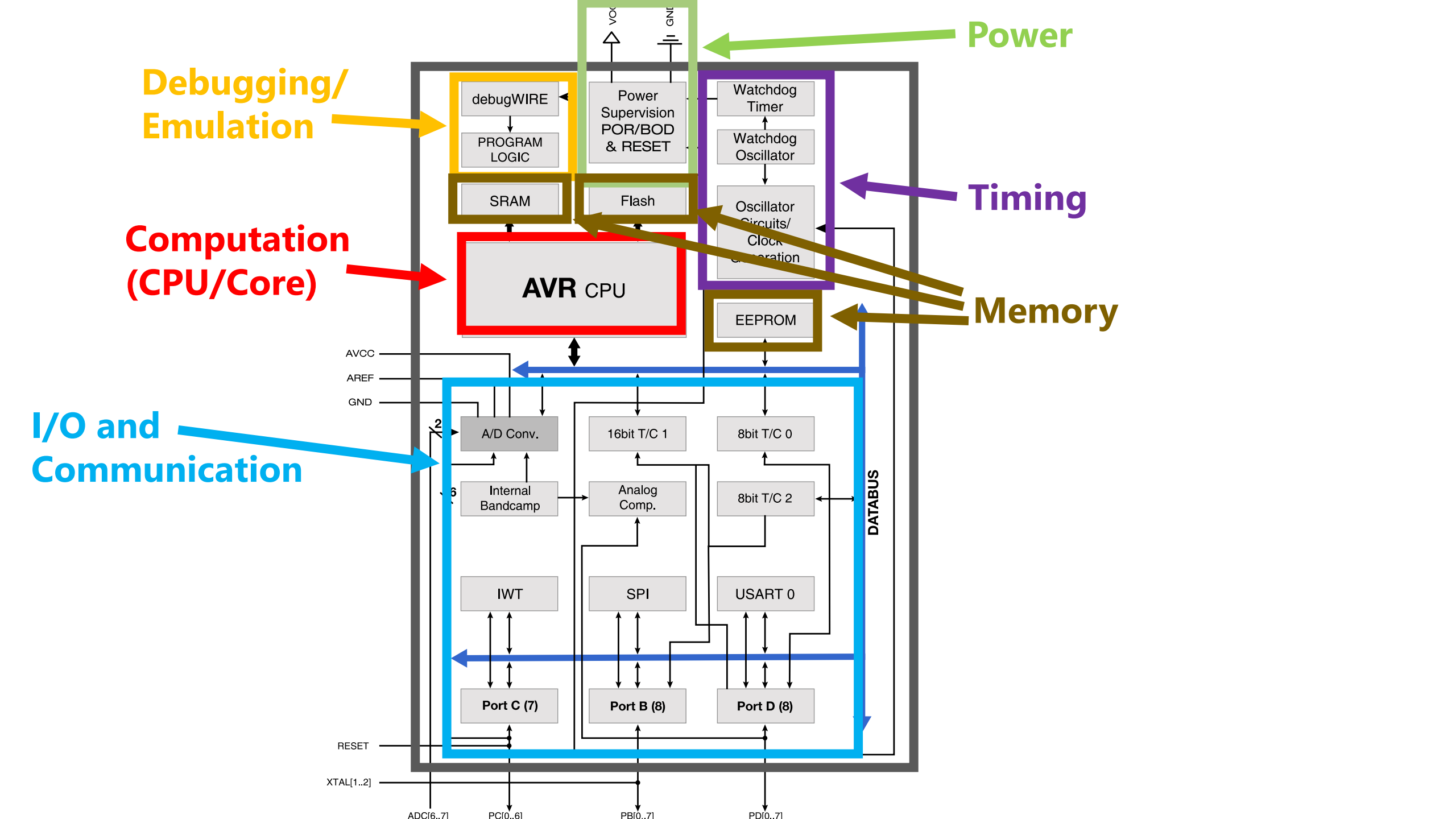
- Hobbyist prototyping platform for IoT devices
  - Simplifies some aspects of programming and circuit development
- You can do so many things with it
  - Can plug in virtually unlimited sensors/indicators/displays/motors/etc.
- Started as a research project at Interaction Design Institute in Ivrea, Italy
  - Name comes from a bar in Italy where founders used to meet
  - Project goal: develop a cheap and easy way for novices to create devices that interact with their environment using sensors/actuators

Arduino Technical Components

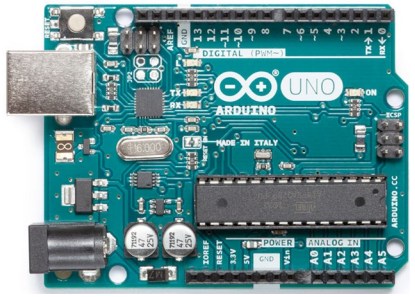
- Arduino provides software to simplify development
  - Interactive development environment
  - Collection of libraries to simplify code
- Code written in C++
  - Program is called a “sketch,” saved with extension “.ino”
- Multiple board designs, using various microprocessors/controllers
  - Both hardware (Creative Commons) and software (GPL) is open-source


I2C Communication
In-Circuit Serial Programming Header For ATmega16
Analog Reference Voltage
Ground
Digital Pins
Serial Communications
Reset Button
Transmit/Receive LEDs
PWM Pins (output)
5V On Indication LED
USB Port
ATmega16
In-Circuit Serial Programming Header For ATmega328
16.000 MHz Crystal Oscillator
Voltage Regulator
ATmega328 (main processor)
DC Power Jack
Diode (protects against reverse polarity on DC input)
Power Source
Analog Pins
Power
Communication
Compute
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
MADE IN ITALY


I2C Communication
In-Circuit Serial Programming Header For ATmega16
Analog Reference Voltage
Ground
Digital Pins
Serial Communications
Reset Button
Transmit/Receive LEDs
PWM Pins (output)
5V On Indication LED
USB Port
ATmega16
In-Circuit Serial Programming Header For ATmega328
16.000 MHz Crystal Oscillator
Voltage Regulator
ATmega328 (main processor)
DC Power Jack
Diode (protects against reverse polarity on DC input)
Power Source
Analog Pins
Power
Communication
Compute


Power
Debugging/
Emulation
Timing
Computation
(CPU/Core)
Memory
I/O and
Communication
debugWIRE
PROGRAM
LOGIC
Power
Supervision
POR/BOD
& RESET
Watchdog
Timer
Watchdog
Oscillator
SRAM
Flash
Oscillator
Circuits/
Clock
AVR CPU
EEPROM
AVCC
AREF
GND
A/D Conv.
16bit T/C 1
8bit T/C 0
Internal
Bandcamp
Analog
Comp.
8bit T/C 2
DATABUS
IWT
SPI
USART 0
Port C (7)
Port B (8)
Port D (8)
RESET
XTAL[1..2]
ADC[6..7]
PC[0..6]
PB[0..7]
PD[0..7]

Official Arduino Boards



Arduino Uno Rev3
(most popular Arduino board)

ATmega328P@16MHz

14 digital I/O pins (6 PWM), 6 analog inputs

USB, power jack, ICSP header

32KB Flash, 2KB SRAM, 1KB EEPROM

2.7"x2.1", 25 grams

Arduino Uno Wi-Fi Rev2
(Similar to Uno, but has Wi-Fi)

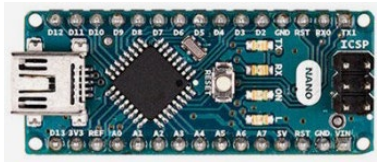
ATmega4809@16MHz
ECC608 crypto co-processor

14 digital I/O pins (5 PWM), 6 analog inputs

USB, power jack, ICSP header

48KB Flash, 6KB SRAM, 0.25KB EEPROM

2.7"x2.1", 25 grams



Arduino Nano
(smaller Arduino board)

ATmega328

22 digital I/O pins (6 PWM), 8 analog inputs

USB, ICSP header

32KB Flash, 2KB SRAM, 1KB EEPROM

0.7"x1.7", 7 grams

Arduino Mega
(larger Arduino board)

ATmega2560

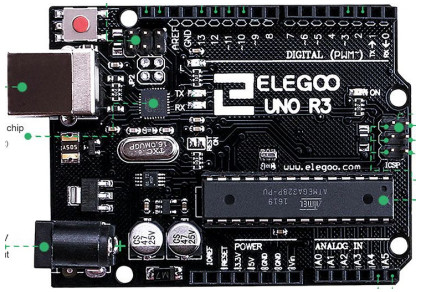
54 digital I/O pins (15 PWM), 16 analog inputs

256KB Flash, 8KB SRAM, 4KB EEPROM

USB, power jack, ICSP header

4"x2.1", 37 grams

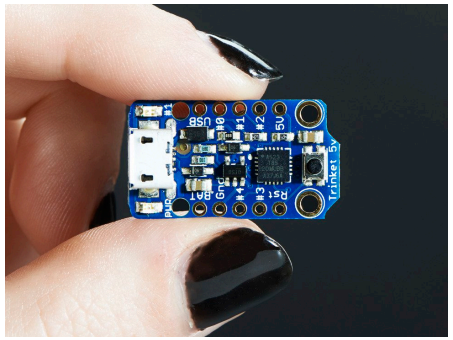
Third-Party Arduino-Compatible Boards



Elegoo Uno
(lower-cost Arduino)

Fully compatible; uses faster control processor (ATMega16U2)

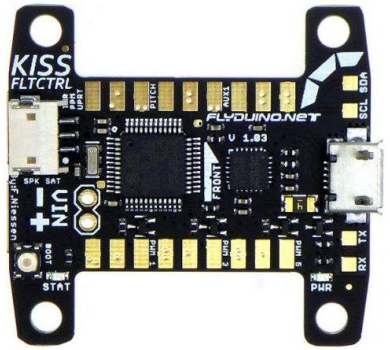
Other boards exist at <$3



Adafruit Trinket
(small Arduino)

Small (1.2"x0.6"), light (1.85 grams). Good for wearables (e.g., necklaces), dog tracking, covert computing, etc.

Attiny85@16MHz, 8KB Flash, 512bytes SRAM, 512bytes EEPROM



FlyDuino KISS FC
(domain-specific Arduino)

Arduino-compatible board designed for autonomous navigation and auto-piloting of aircraft.

Flight control firmware monitors and performs telemetry of motor speed (ESCs). Stabilizes flight and automates movement and tracking.

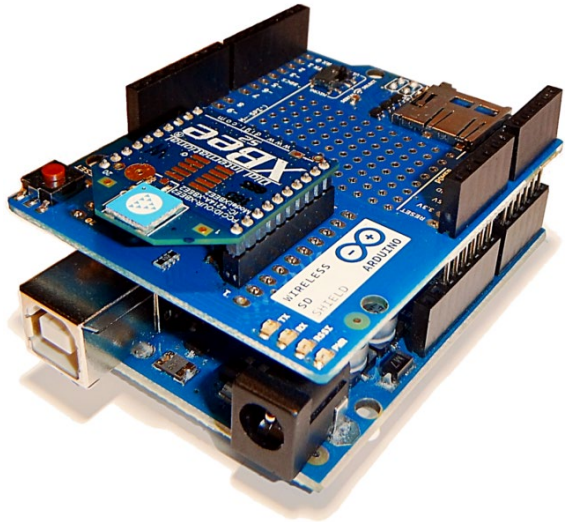
Ardusat
(domain-specific Arduino)

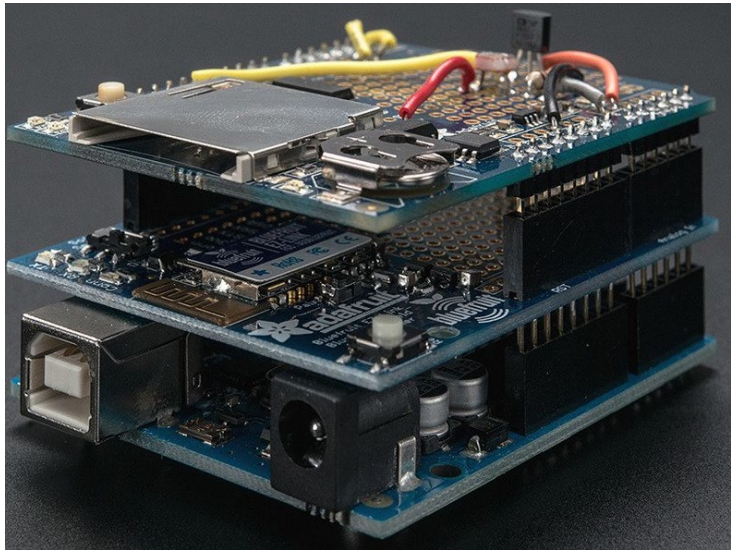
Nanosatellite based on CubeSat standard

Public allowed to use platform via ArduSatSDK while in space

Includes a 3-axis magnetometer, a 3-axis gyroscope, a 3-axis accelerometer, an infrared temperature sensor, four temperature sensors, two luminosity sensors (infrared and vis light), two Geiger counters, one optical spectrometer one 1.3MP camera

Arduino Shields





Arduino Uno + Xbee Module + SD Card Storage

Many shields are out there:

Motor drivers, cellular (e.g., GSM GPRS), data logging, GPS, blank (for prototyping), LCD/Displays, Cameras, Speakers/buzzers, sensors (e.g., temperature/humidity), pushbuttons, capacitive touchpads, IR receivers, medical sensors, biometrics, wired ethernet, relays (high voltage), lasers, smoke detectors, joystick/game controller, MP3 players, etc.

How do we program an Arduino?

- Usually people program microcontrollers in high-level languages

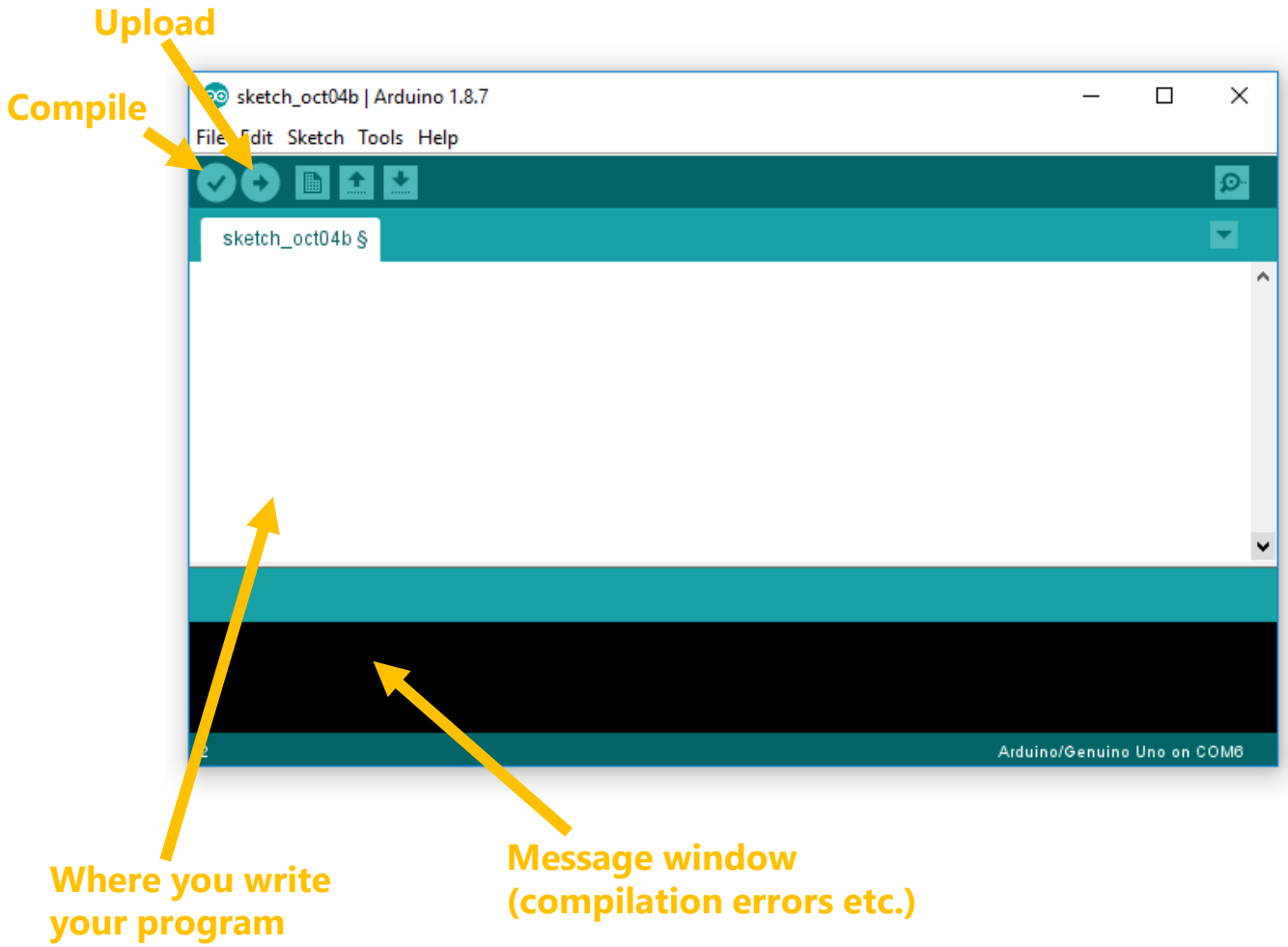
- Arduino platform makes this process even easier
  - Simple IDE
  - Libraries, objects (some from Arduino, some from Atmel/AVR)

How do we program an Arduino?

- Arduino sketches are C++ programs calling some extra supporting libraries

- Arduino programming is similar to other kinds of embedded programming
  - Approaches very generalizable to other microcontrollers, platforms, etc.

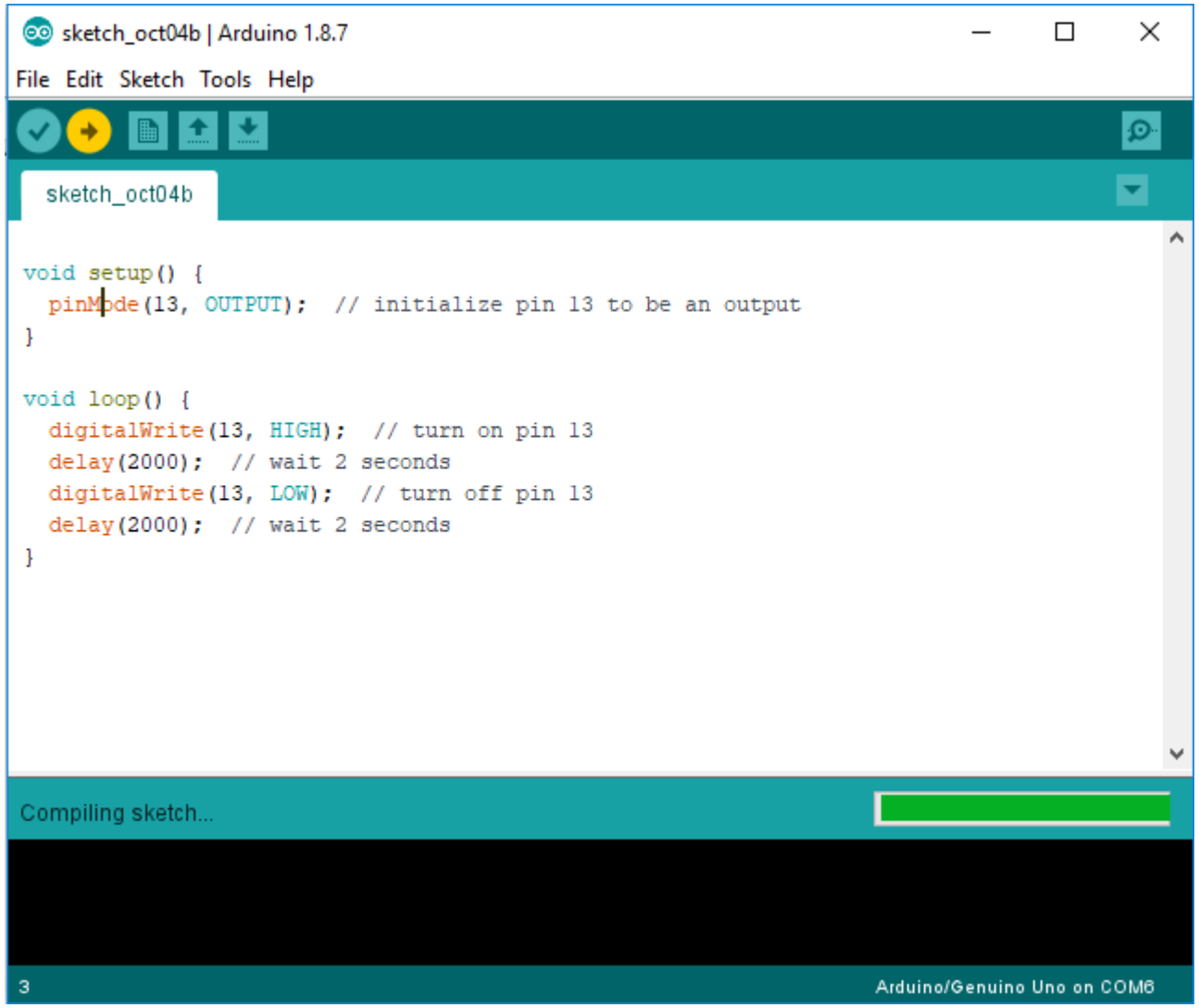
What the IDE Looks Like



Setting up Arduino:

1. To tell IDE how to reach the Arduino, choose Tools::Port: Select COM1 or whatever port your Arduino is plugged into
   - Windows: Device manager, expand Ports, see where Arduino is
   - Mac: type "`ls /dev/*`" in terminal, use port number listed for /dev/tty.usbmodem* or /dev/tty.usbserial*
   - Linux: type "`ls /dev/ttyUSB*`" or "`/dev/ttyACM*`"
2. To tell IDE how to talk to the Arduino, tell it the type of board under Tools:Board
   - For Uno or Uno-compatible board select Arduino/Genuino Uno

Writing an Arduino Program



- Two main functions, executed by bootloader
  - **setup()** – executed once on startup
  - **loop()** – run repeatedly, forever
- Functions you can call (digital I/O):
  - **pinMode**(<pin>,<type>) – configures pin as detecting an input signal (INPUT) or outputting a signal (OUTPUT). Can also enable an internal pullup resistor (INPUT_PULLUP)
  - **digitalWrite**(<pin>,<value>) – sets pin state (HIGH or LOW)
  - <int> = **digitalRead**() – reads pin state (HIGH or LOW)

Uploading/Running an Arduino Program

Three key steps:

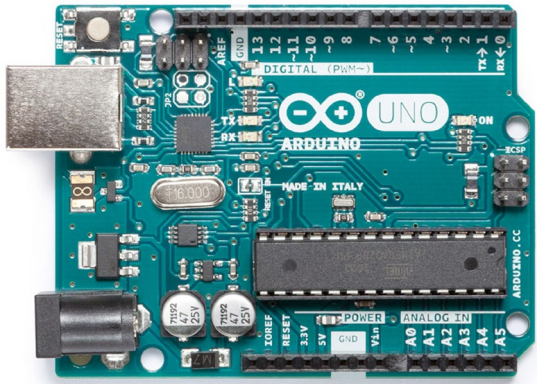
1. Verify/Compile your code
   - Sketch::Verify/Compile, or hit the check mark button
2. Upload bootloader to board
   - Code that runs before main program
   - Arduino bootloader enables re-programming flash memory over serial port
   - Previous step takes care of this for you
3. Upload your code to board
   - Sketch::Upload, or hit right-arrow button

Uploading/Running an Arduino Program

- Program will just start running automatically
- Troubleshooting ideas:
  - Make sure you have right board type and port selected
  - Check for short circuits, hot components, smoky smells
  - Use a multimeter to check output values and current loads
  - Use serial monitor (Serial.println()) to log progress
  - Use a simulator/emulator

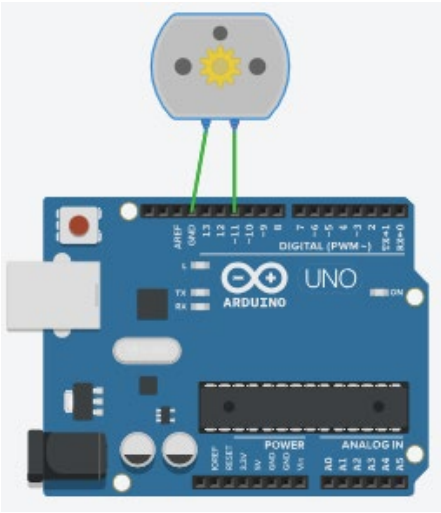
A Few More Example Sketches

Suppose you…

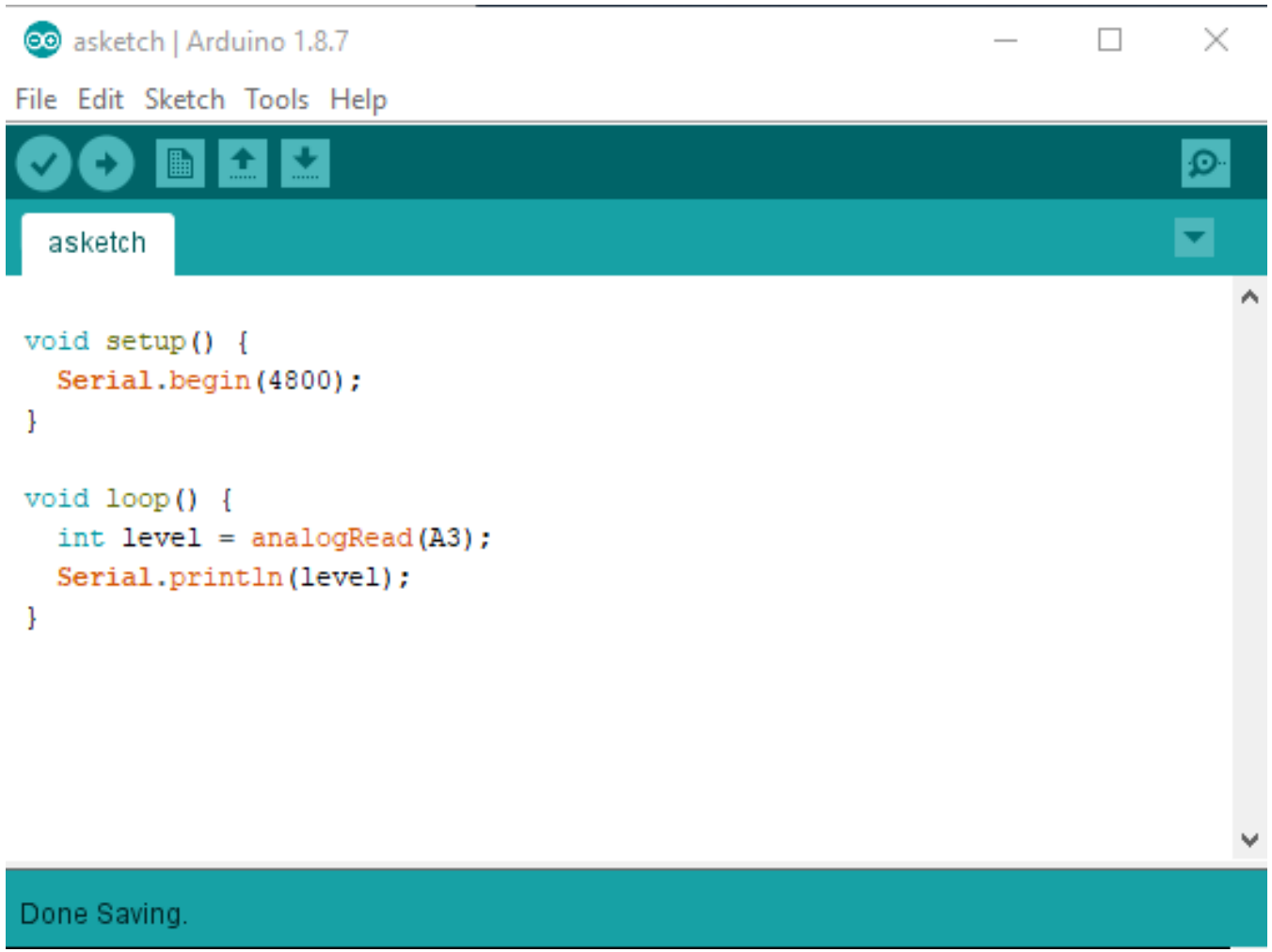


- Need to debug an Arduino
  - How do you know what's going on inside?
  - Can use LEDs but that's kind of hard
  - May need to log and collect data

- Want to measure/send non-digital (analog) values?
  - E.g., read output from a sensor, control speed of motor



Serial Programming, Analog I/O



- Serial functions:
  - **begin**(<speed>) – sets data rate (baud)
  - **println**(<value>) – outputs data as ASCII text to serial port
  - <int>=**read**() – reads one byte from serial port, returns -1 if no data to read
  - Warning: data is written to USB *and* pins 0/1 – will clash with other uses of pins 0/1
  - Also see: hardware debugging (debugWire)

Serial Programming, Analog I/O

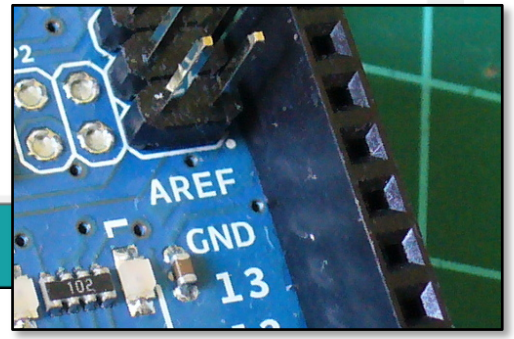
asketch | Arduino 1.8.7

File Edit Sketch Tools Help

asketch

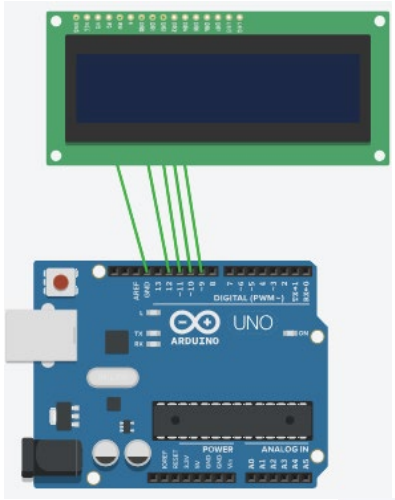
```
void setup() {
  Serial.begin(4800);
}

void loop() {
  int level = analogRead(A3);
  Serial.println(level);
}
```

Done Saving.



- Analog I/O functions:
  - <int>=**analogRead**(<pin>) reads voltage from specified input pin as value between [0,1023]
  - **analogWrite**(<level>) – sends voltage level (PWM) to pin, specified as range from 0 (always off) to 255 (always on)
  - **analogReference**() – sets what 1023 corresponds to for analog input – DEFAULT (5V), INTERNAL (1.1V), EXTERNAL (voltage applied to AREF pin), etc.

Suppose You…



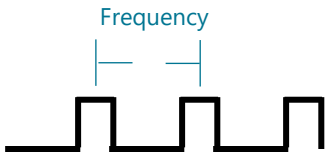
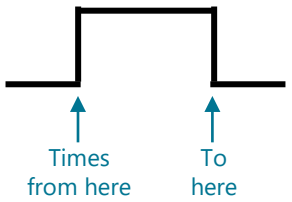
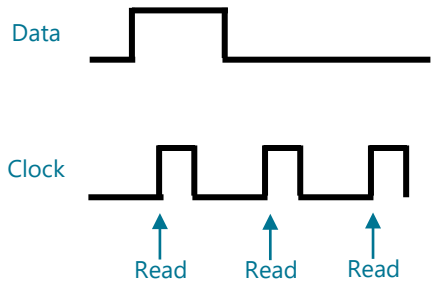
- Want to send or read digital data
  - E.g., you have an external component with a microcontroller

- Measure pulses
  - Measure how long user holds down a button, how long car was blocking highway laser sensor

- Generate tones
  - Alarm sounds when burglar steps on smart floor, make music play when you walk into a room

Digital Streaming I/O Functions

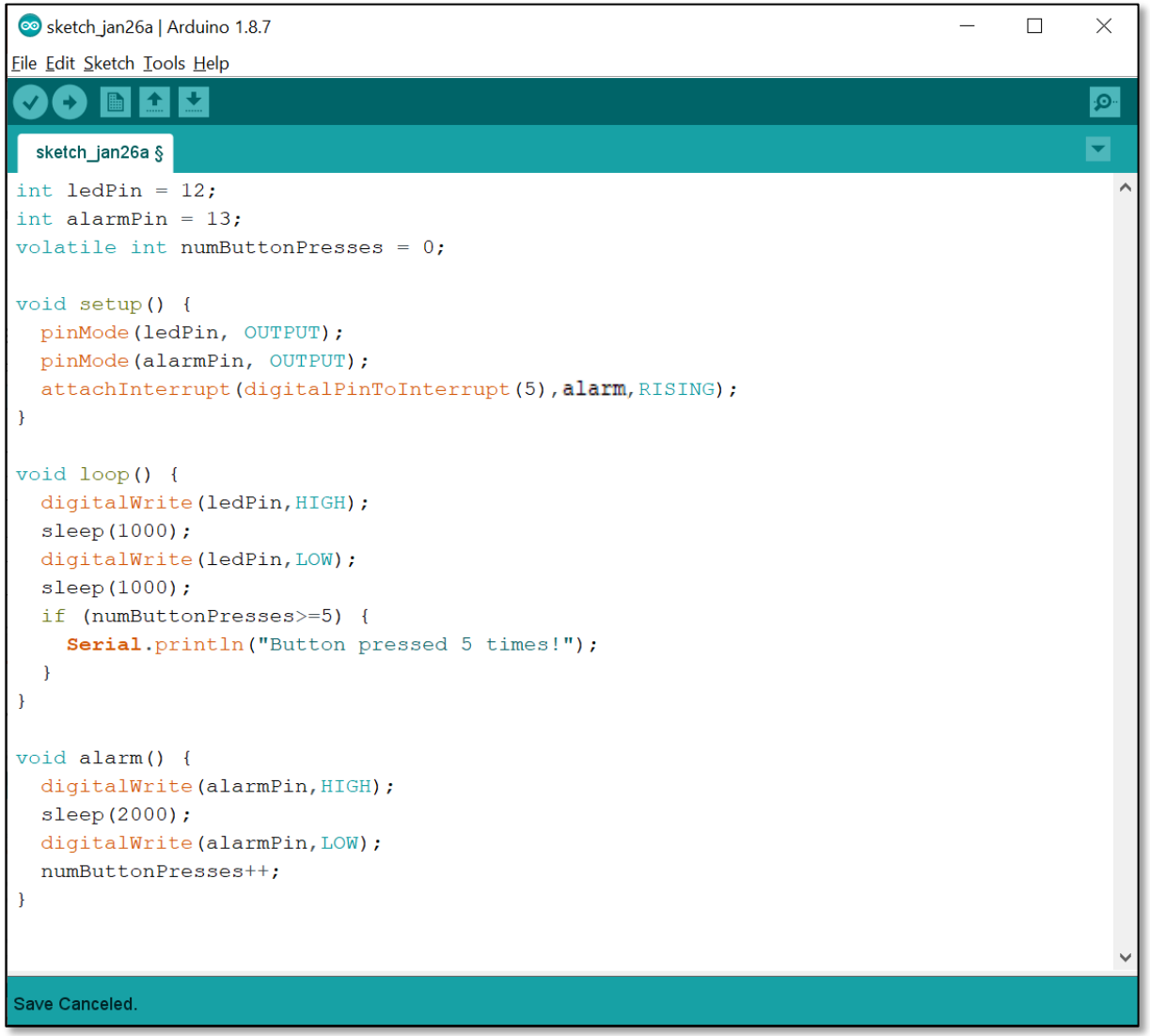
- Functions for shifting data in/out:

Diagram	Function name(s)	What it does	Notes
	**tone**(pin, frequency, [duration]) Example: tone(8,500)	Generates a square wave of the specified frequency in Hz (with 50% duty cycle). Can connect to piezo buzzer or speaker to play tones.	Also see: **noTone**(pin) – stops generation of square wave on specified pin.
	**pulseIn**(pin, value, [timeout]) Example: unsigned long duration = pulseIn(3,HIGH)	Measures duration of next received pulse. E.g. If value is HIGH, pulseIn() waits for pin to transition LOW→HIGH, starts timer, and ends on transition HIGH→LOW.	Returns 0 if no complete pulse was received before timeout. Works on pulses from 10µs → 3 minutes.
	byte incoming = **shiftIn**(dataPin, clockPin, bitOrder) Example: byte value = shiftIn(6,7,MSBFIRST)	Shifts a byte of data in from data pin, with values read on rising edges of clock pin.	Also see: **shiftOut**(dataPin, clockPin, bitOrder, value) – shifts out the byte value, driving both data and clock pins appropriately.

Suppose You…

- Want to do multiple things at once
  - Run autopilot for drone, but user gives you a new command
  - Display camera feed, but pop up character pad when user touches screen

Interrupts



- **attachInterrupt**(pin, function, mode) – associates function pointer to be called on pin level change
  - Modes: LOW, CHANGE, RISING, FALLING
- **digitalPinToInterrupt**(pin) – converts pin # to interrupit #
- **noInterrupts**()/**interrupts**() – enables/disables interrupts
- Notes:
  - Only one ISR can run at a time
    - Next ISR to run selected according to priority
  - Shared variables must be declared volatile

What if something goes wrong?

- Program crashes or hangs in the field
  - It is 10,000 feet in the air or floating in the arctic
  - Your plant watering system locks up when you're on vacation
  - You can't get out there to help it

Watchdog Timer

- "Deadman's switch" hardware-supported timer
  - Counter counts up continuously in hardware
  - If not reset within timeout, board is rebooted
- **wdt_enable**(time) – sets timeout
- **wdt_reset**() – resets timer
- Levels supported by ATmega328: 15ms, 30ms, 60ms, 120ms, 250ms, 500ms, 1s, 2s, 4s, 8s

asketch | Arduino 1.8.7

File Edit Sketch Tools Help

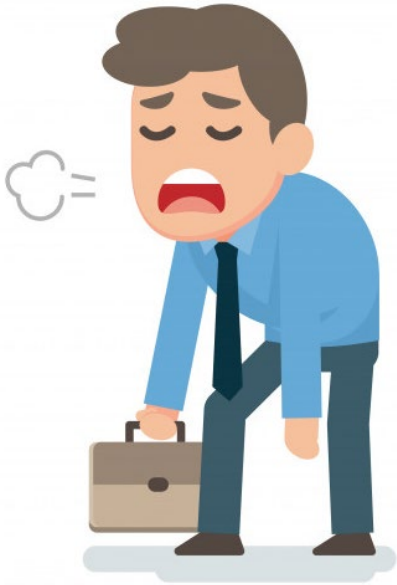
asketch §

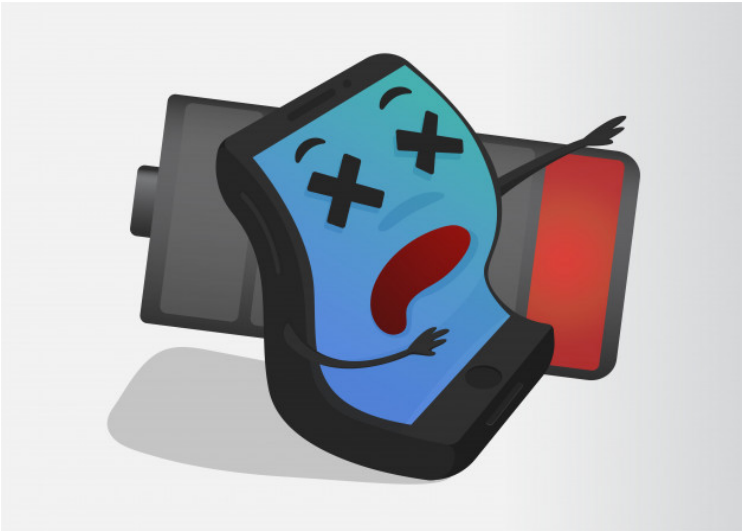
```
#include <avr/wdt.h>

void setup() {
  wdt_enable(WDTO_2S); // wake up the dog
}

void loop() {
  process_data();
  wdt_reset(); // pat the dog
}
```

Done Saving.

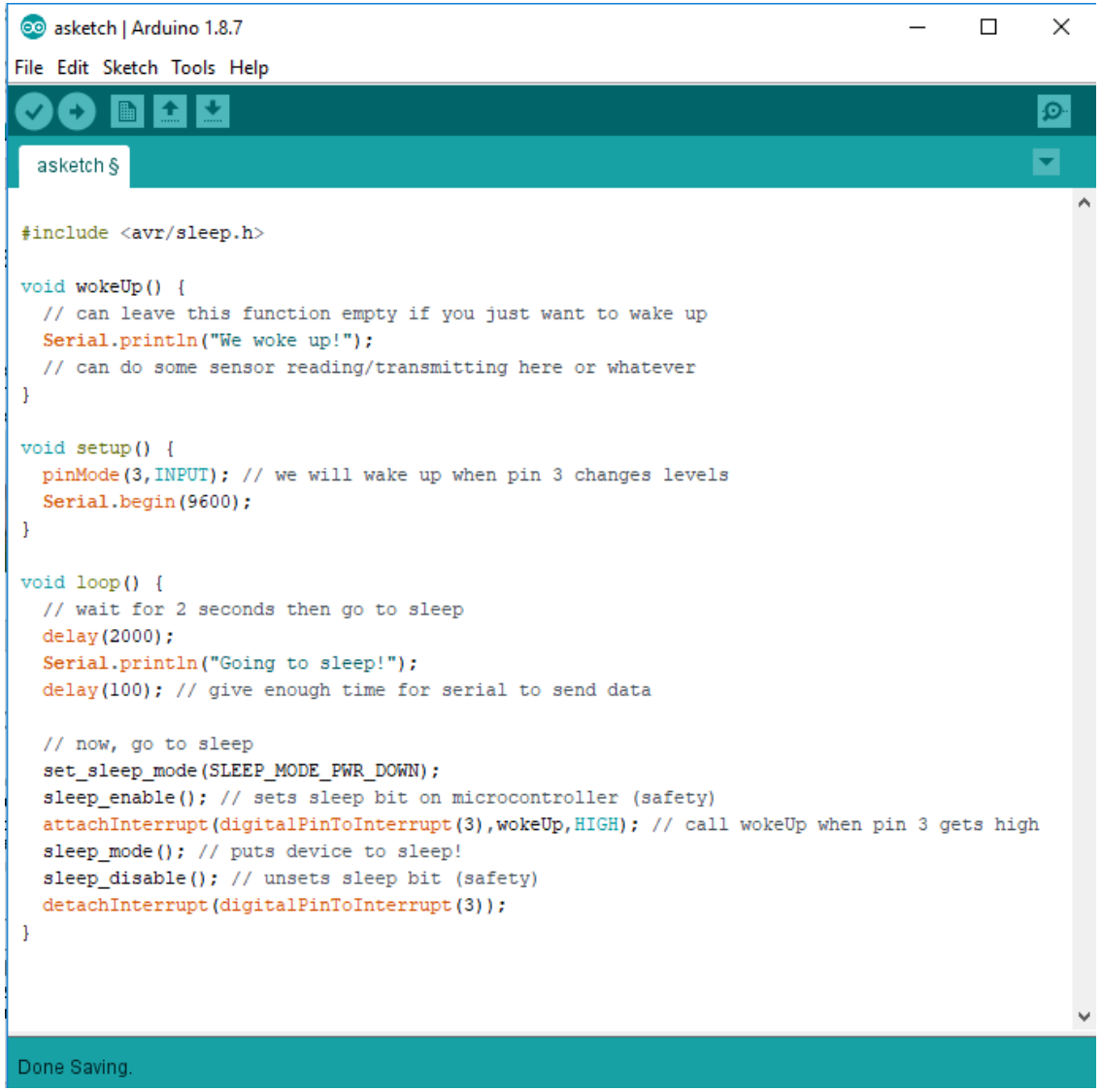
What if you are running out of power?

- May need to operate for long times on low battery
- May be reaching end of battery life
- May not be able to replace battery
- May not even have a battery – energy harvesting and sources drying up



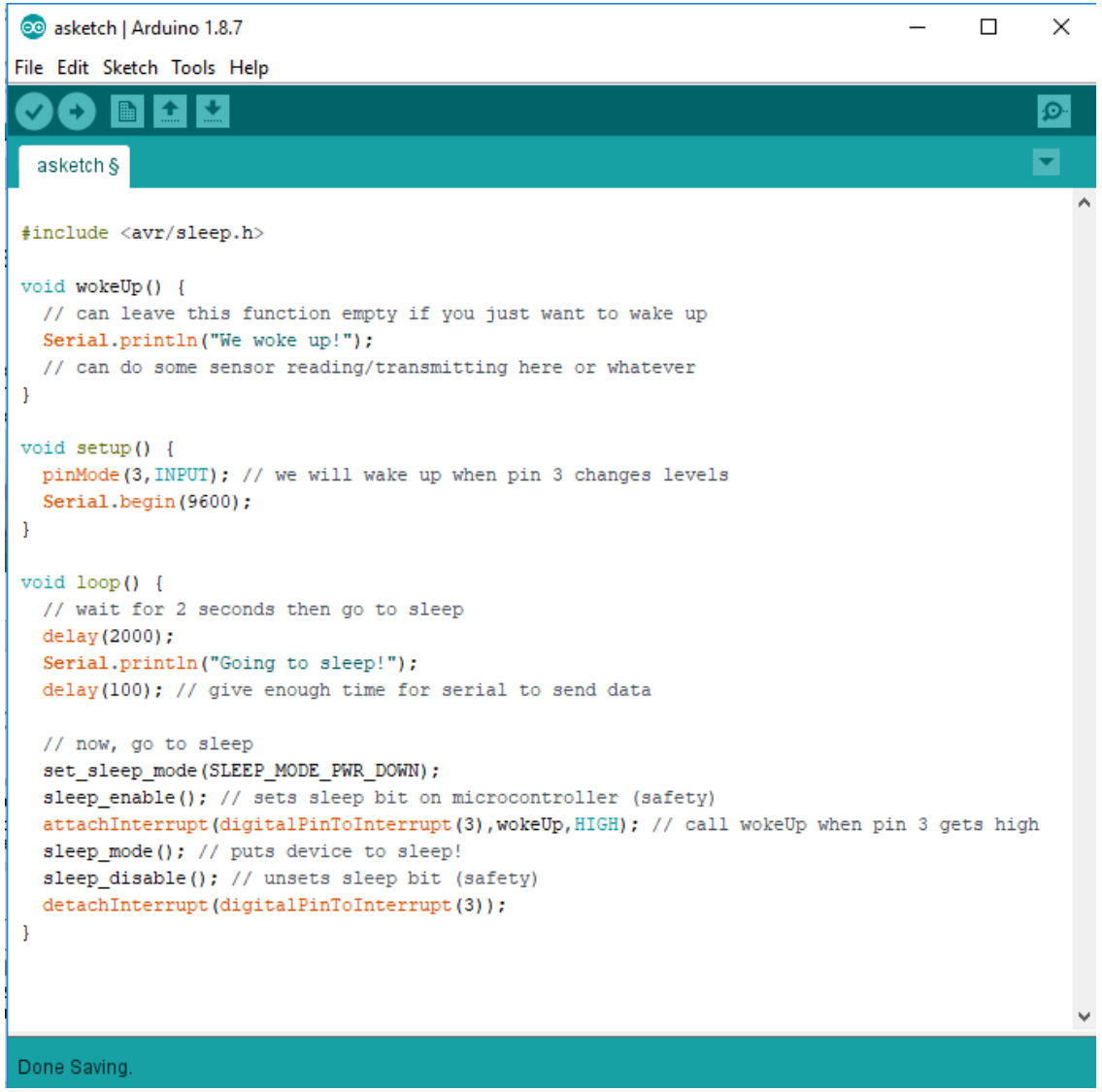


Sleep Mode



- Can be woken up by timer, pin level change
- **set_sleep_mode**(mode) – sets how deeply to sleep
  - Sleep modes: SLEEP_MODE_PWR_DOWN (most savings), SLEEP_MODE_STANDBY, SLEEP_MODE_PWR_SAVE, SLEEP_MODE_ADC, SLEEP_MODE_IDLE (least savings)
  - Disables brown-out detection, timers, analog-digital conversion
- **sleep_enable()/sleep_disable()** – enables/disables safety "sleep" bit on microcontroller
- **sleep_mode()** – triggers sleep

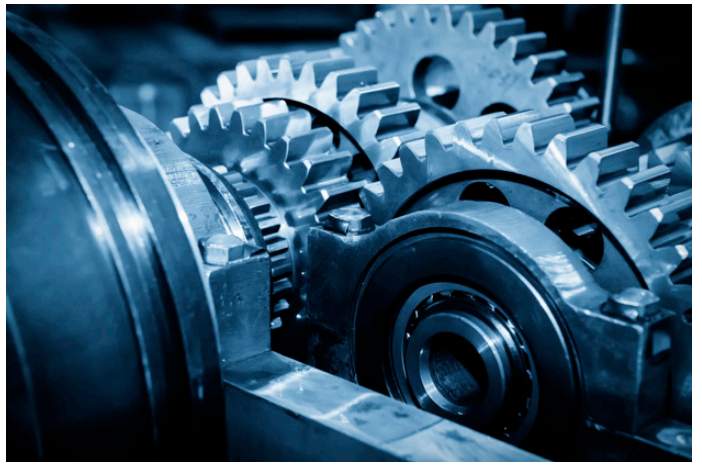
Sleep Mode



- Can replace delay() calls, use when not doing anything important
- Doesn't actually save much power on some Arduinos because of voltage regulator
  - However you can use this concept when programming other platforms
- Other ways to save power: avoid linear regulator, lower voltage (5V to 3.3V reduces ~4mA to ~1mA), reduce clock speed

What if you need to store some information?

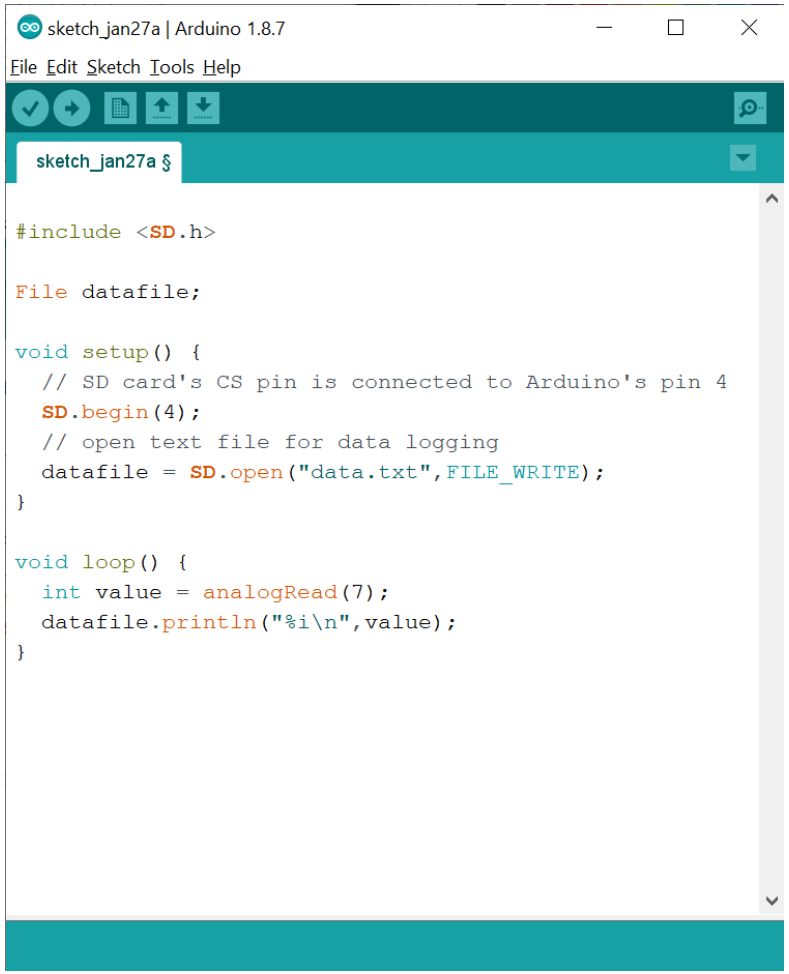
- May need to log data
  - Could send to Serial, but what if not attached to a computer?
  - Could send to Wi-Fi, but what if no clearance to send?
- Need to retain state across power outages/resets
  - Storing configurations and user preferences



Arduino Storage

EEPROM

SD Card



- Can access memory and storage. Examples:
- Write to EEPROM (byte-addressable):
  - read()/write()
  - clear() – clear entire EEPROM
  - crc() – calculates CRC of entire EEPROM contents
- Write to SD card (file-based storage):
  - Similar to POSIX: open()/close()/println()/read()/write()/mkdir()/remove()/rmdir()

References

- Mouser Electronic / EasyEDA
- BatteryJunction.com
- Uxcell
- E-Projects
- NTE Electronics / https://www.ntepartsdirect.com/
- Jameco Electronics / https://www.jameco.com/
- Digi-Key Electronics / https://www.digikey.com/
- Dilson Enterprises
- Regency Semiconductors

References

- Nest
- TickPlant
- Amazon
- Adafruit
- FilipeFlop Componentes Eletrônicos
- Anker
- Transfer Multisort Elektronik
- SparkFun Electronics / https://www.sparkfun.com/
- DJI
- Yale Security
- Open Bionics

References

- Banggood.com
- Active Robots / https://www.active-robots.com/
- Davis Instruments / https://www.davisinstruments.com/
- Honda
- Anki
- Parallax Inc. / https://www.parallax.com/
- HT Sensor / http://htc-sensor.com/
- AIRSENSE Analytics GmbH / http://airsense.com/en
- Integrated Device Technology, Inc. / https://www.idt.com/

References

- Texas Instruments
- National Semiconductor
- SeekIC
- Microchip
- Espressif Systems
- Adafruit / https://www.adafruit.com/
- NikeLG Electronics
- Moverio
- Fitbit
- HTC Vive

References

- Parrot http://www.parrot.com
- Honeywell
- Pleo
- ElectronicSurplus.ca
- Arduino.cc