Analog and Digital Signals

Todays Topics

- What’s the difference between analog and digital signals
- Review Electricity Equations
- General Purpose Input / Output (GPIO)
- Pulse Width Modulation (PWM)
- Analog to Digital Converters (ADC)
- Microcontrollers and Computer

Electronic Signals

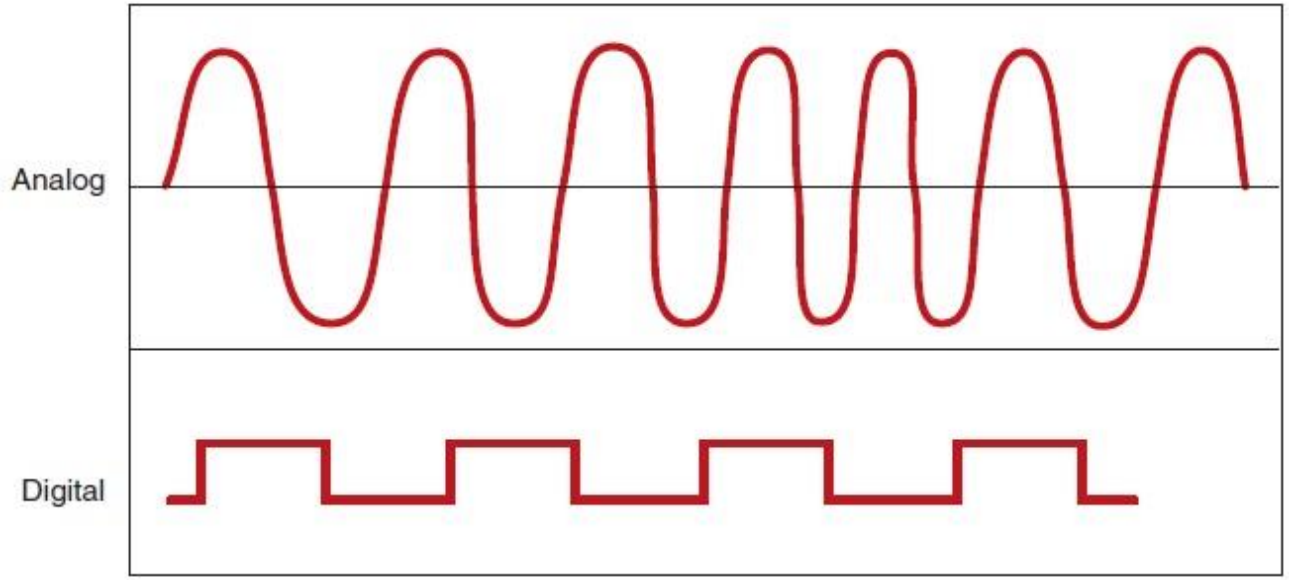


Image from http://autosystempro.com/analog-and-digital-principles/

Analog and Digital

- Digital devices can only store digital signals
  - Using 1 or several bits / sample
- We store an array of numbers
- Parameters
  - Bits per sample
  - Sampling rate
- Example: Audio…

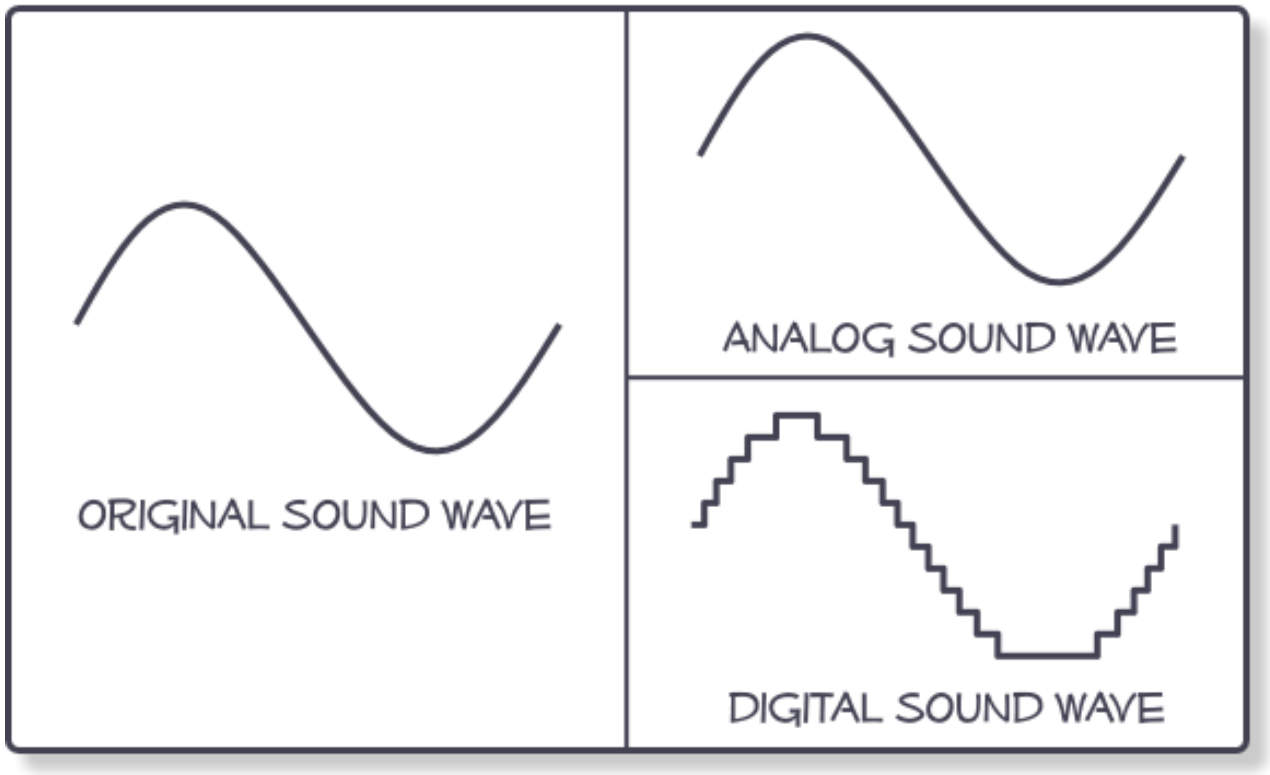
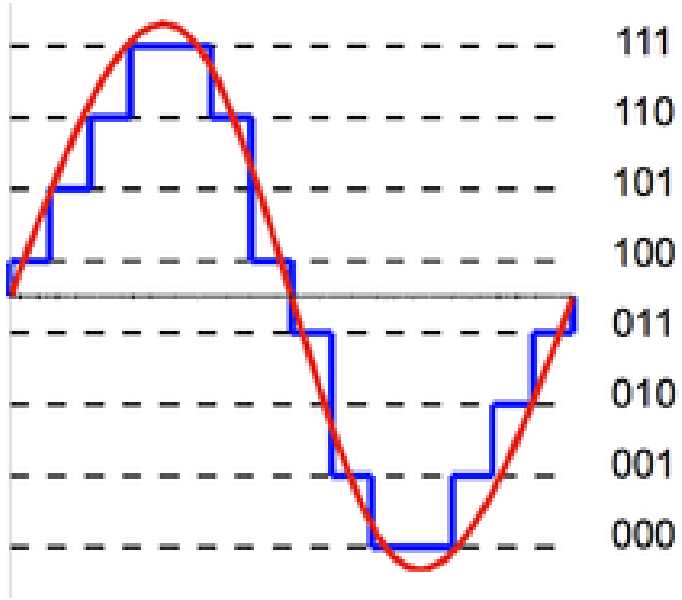


Image from http://www.centerpointaudio.com/Analog-VS-Digital.aspx

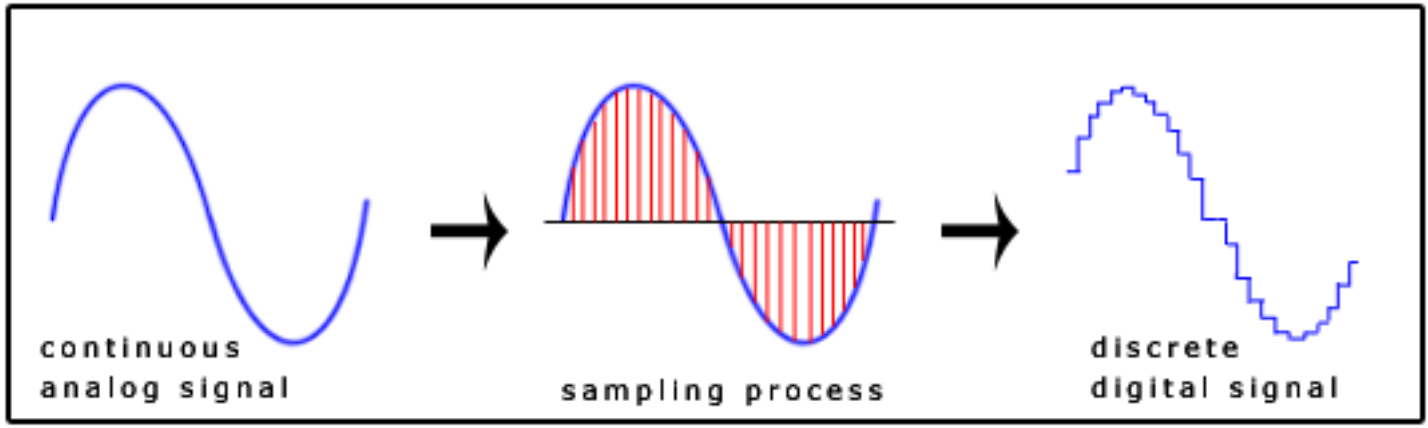
Bits per sample



- Using just 1 bit
  - Just 2 possible values/states: LOW (0) and HIGH (1)
  - digital
- Using >1 bit (n bits where n>1)
  - 2^n possible values/states
- Example: using 3 bits:
  - $2^3 = 8$ possible values/states
- Called Quantisation
- More bits -> higher resolution -> higher accuracy

Sampling

- Sensor signals are analog
  - Temp -> voltage -> value
- Sampling Rate
  - Higher sampling rate(frequency) will give higher the accuracy



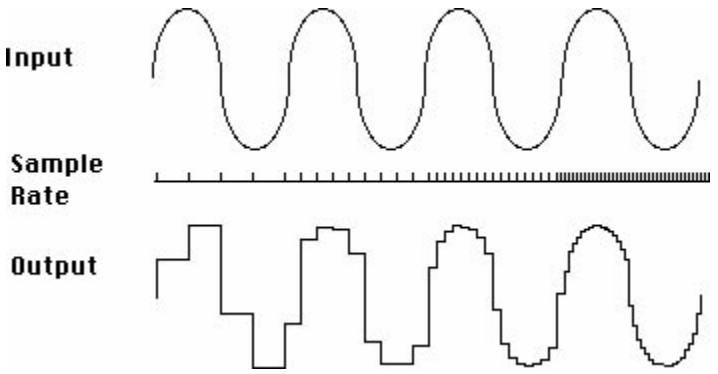
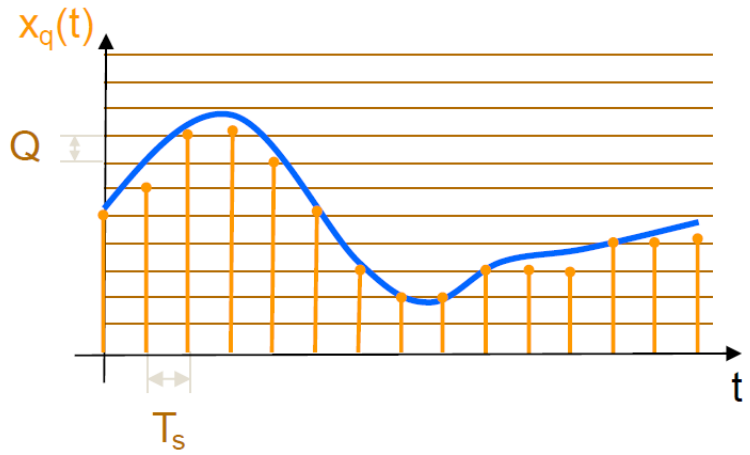
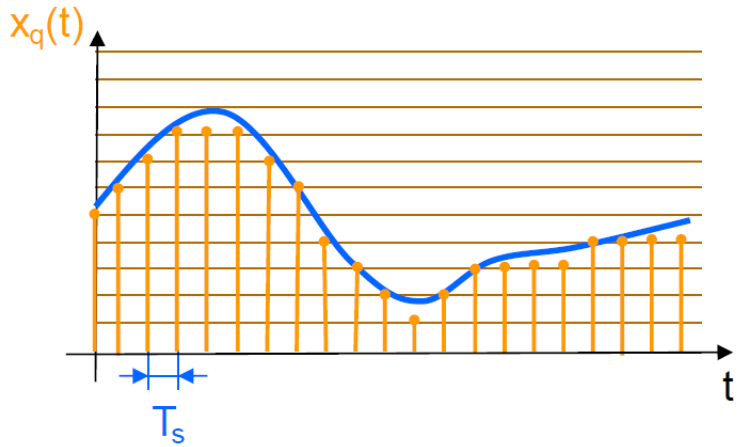


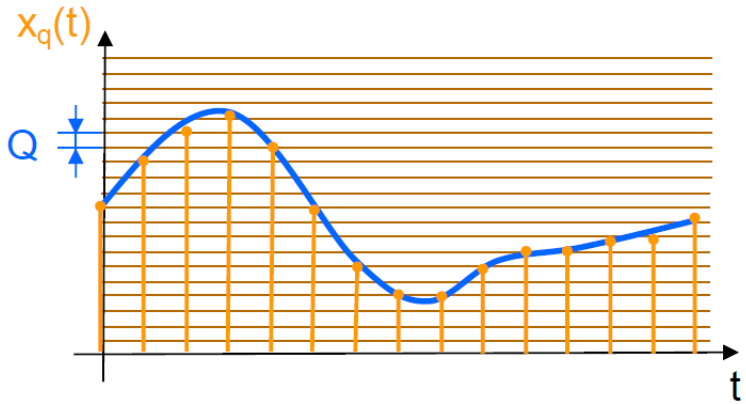
Image from http://www.snotmonkey.com/work/school/405/overview.html
Image from http://www.jazzpoparkisto.net/audio/audio32.html



Higher Sampling rate



Higher Resolution



Analog and Digital Convertsion

- Measure voltage
- Parameters
  - Bits per sample
  - Sampling rate
- For 1 bit we have …
  - GPIO Input
- For n bits we have …
  - $0 – 2^n-1$

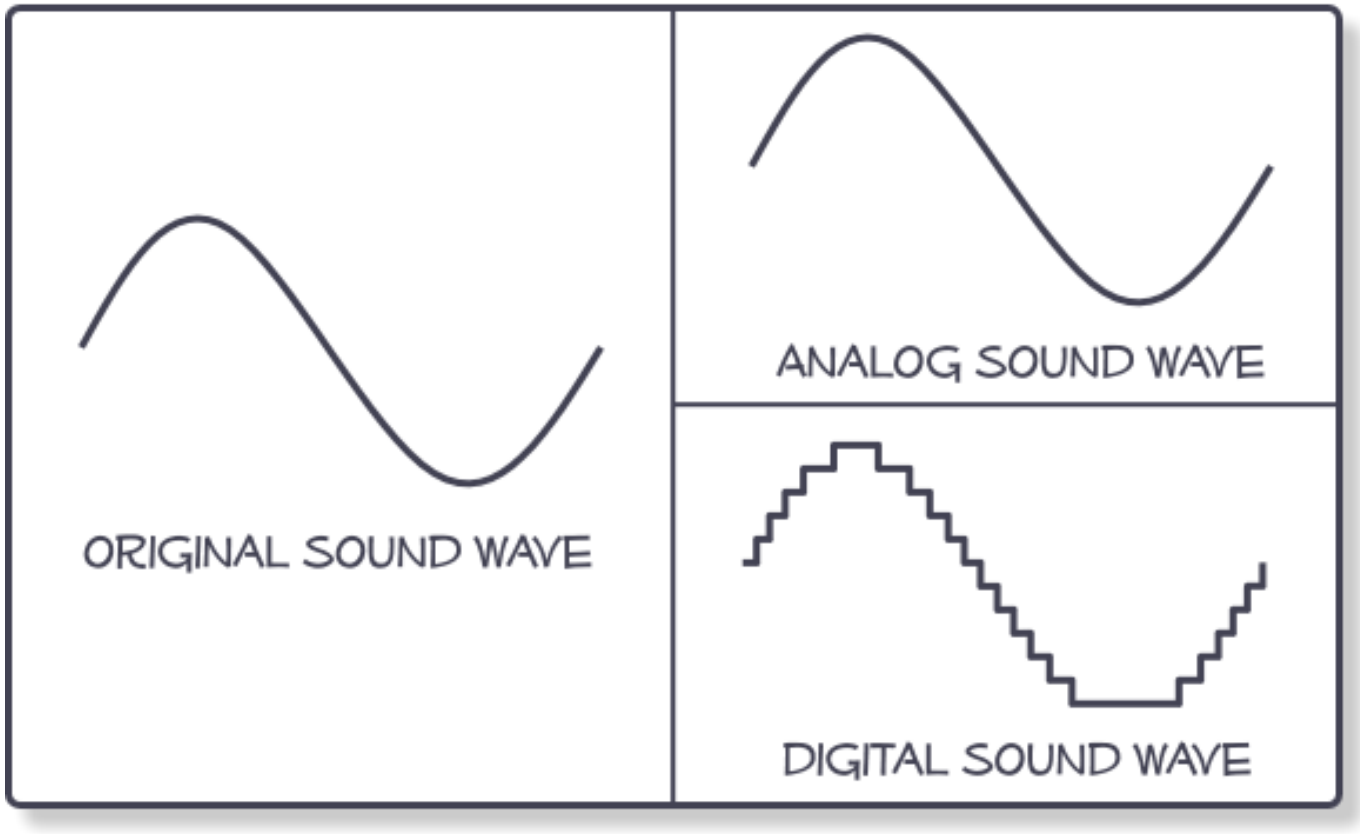


Image from http://www.centerpointaudio.com/Analog-VS-Digital.aspx

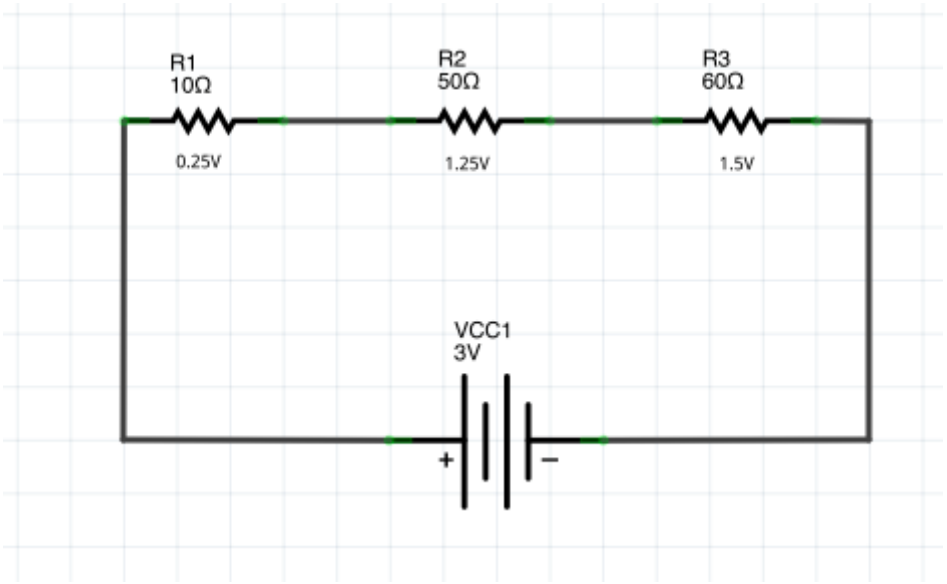
Electricity Equations

$$I = \frac{V}{R}$$

$$\sum_k i_k = 0$$

$$\sum_k E_k = \sum_k R_k I_k$$

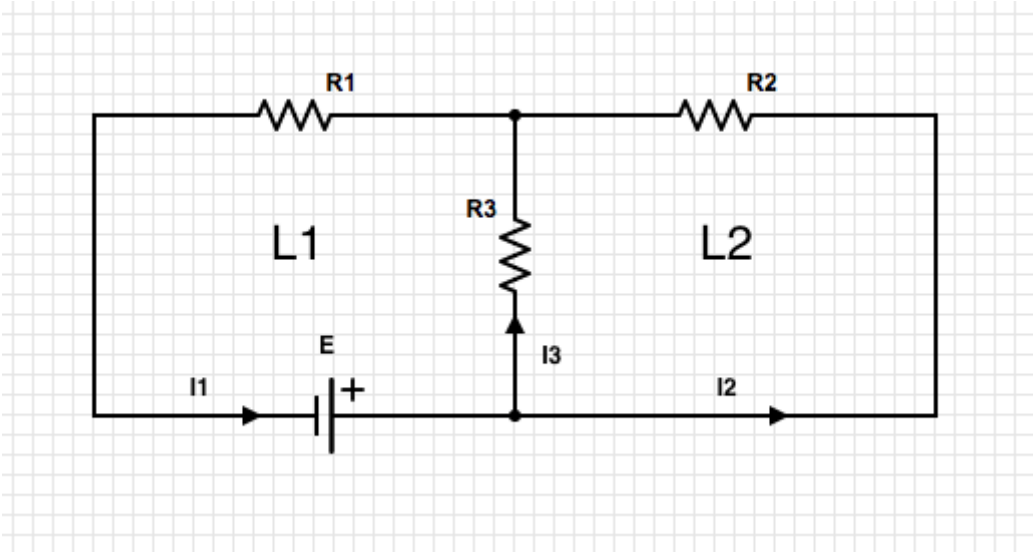
Ohm’s Law



Kirchhoff Law I

$$\sum_k i_k = 0$$

$$-I_1 + I_2 + I_3 = 0$$

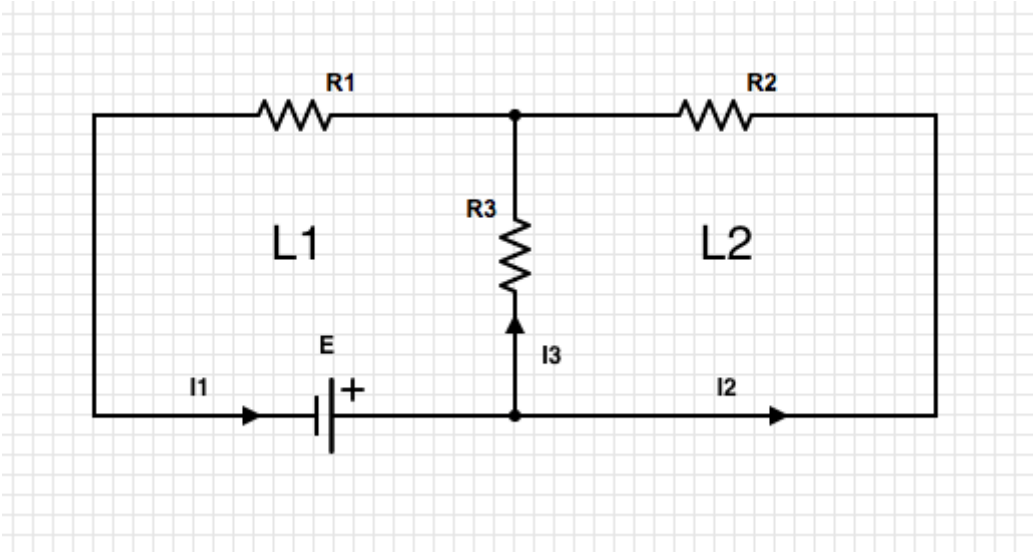


Kirchhoff's Law II

$$\sum_k E_k = \sum_k R_k I_k$$

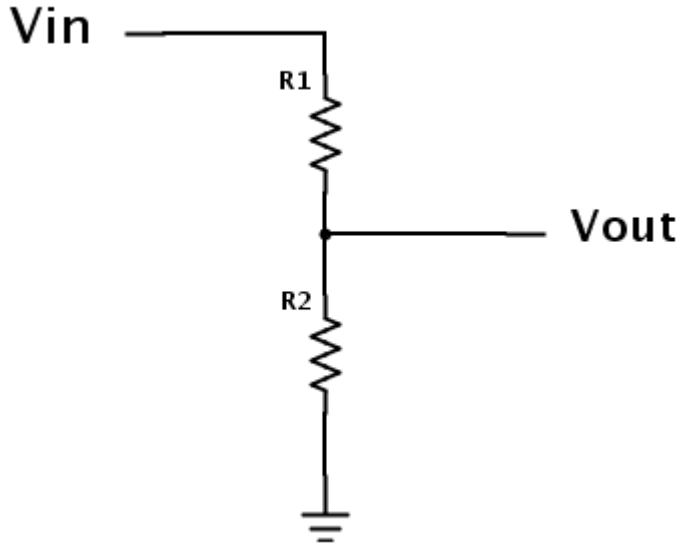
$$L_1 : E = I_1 R_1 + I_3 R_3$$

$$L_2 : 0 = I_2 R_2 - I_3 R_3$$



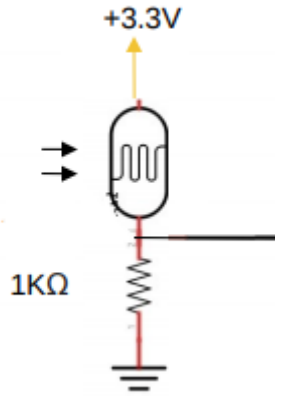
Voltage Divider

$$V_1 + V_2 = V_{in}$$
$$V_1 = I\ R_1$$
$$V_2 = V_{out} = I\ R_2$$



If current in output wire is 0, then:

$$Vout = \frac{R2}{R1 + R2}\text{Vin}$$

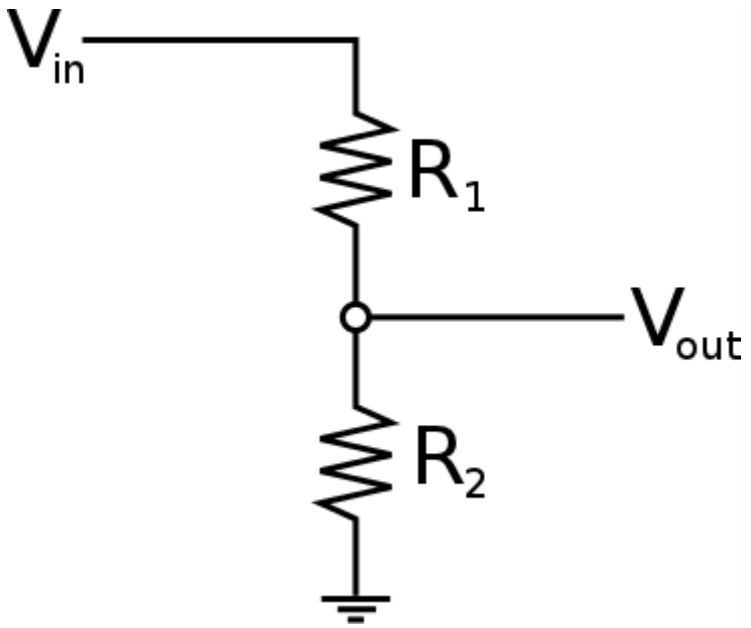


Voltage Divider

$$R_1 = 0$$

$$V_{out} = ?$$

$$Vout = \frac{R2}{R1 + R2}\text{Vin}$$

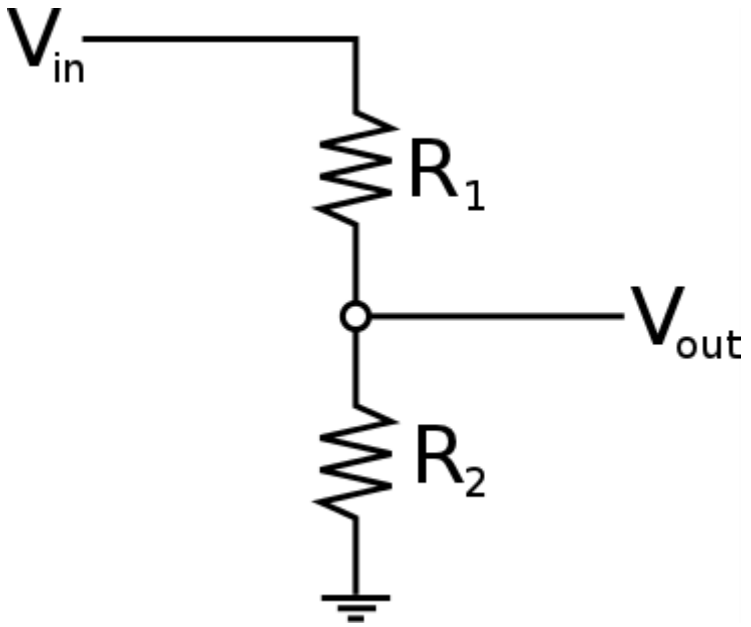


Voltage Divider

$$R_1 = ¥$$

$$V_{out} = ?$$

$$Vout = \frac{R2}{R1 + R2}\text{Vin}$$

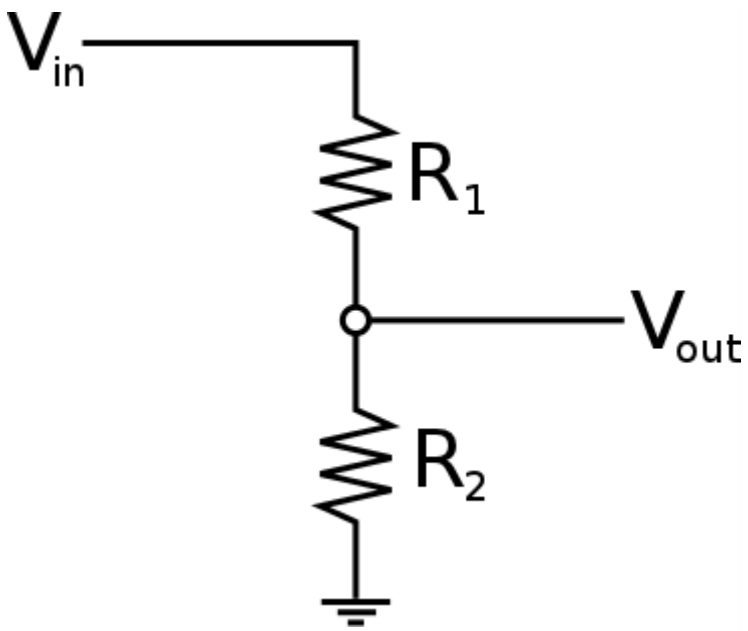


Voltage Divider

$$Vout = \frac{R2}{R1 + R2} \text{Vin}$$

$$R_2 = 0$$

$$V_{out} = ?$$

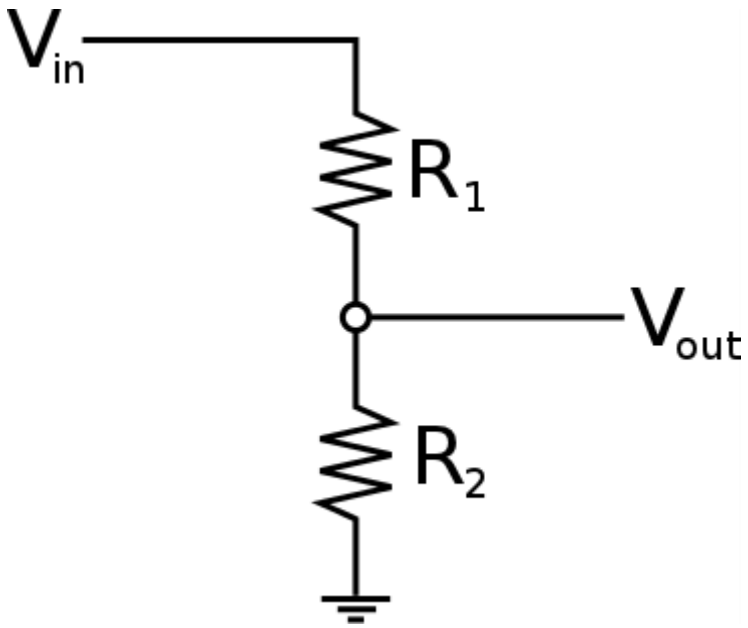


Voltage Divider

$$Vout = \frac{R2}{R1 + R2}\text{Vin}$$

$R_2 = ¥$

$V_{out} = ?$



Voltage Divider

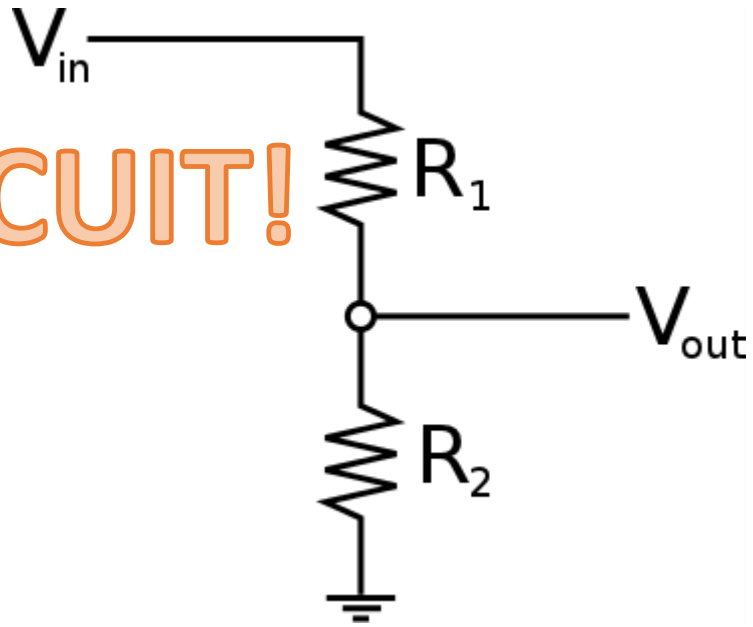
$$Vout = \frac{R2}{R1 + R2}\text{Vin}$$

$R_1 = 0$

$R_2 = 0$

$V_{out} = ?$

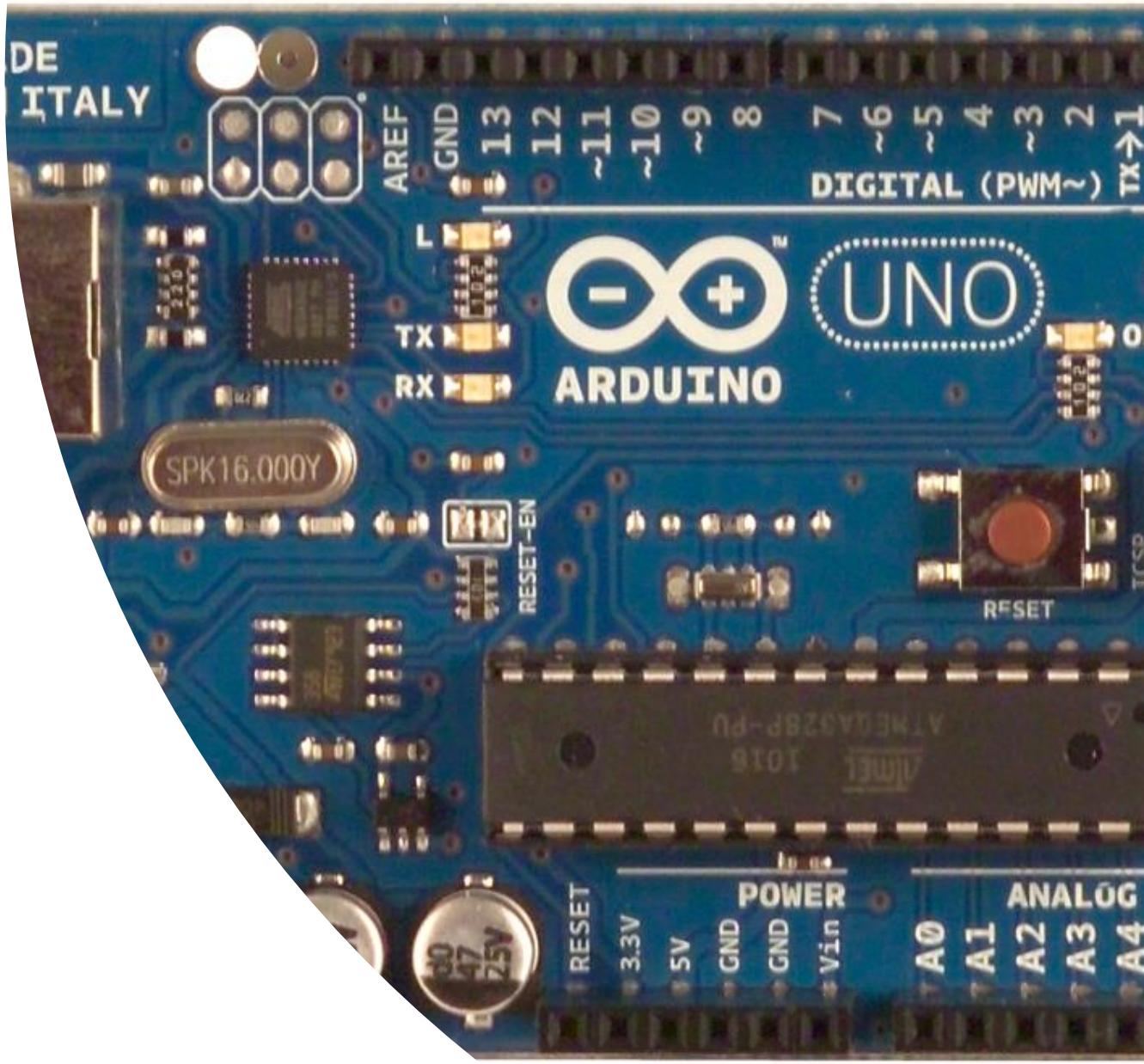
SHORT CIRCUIT!



Development Boards

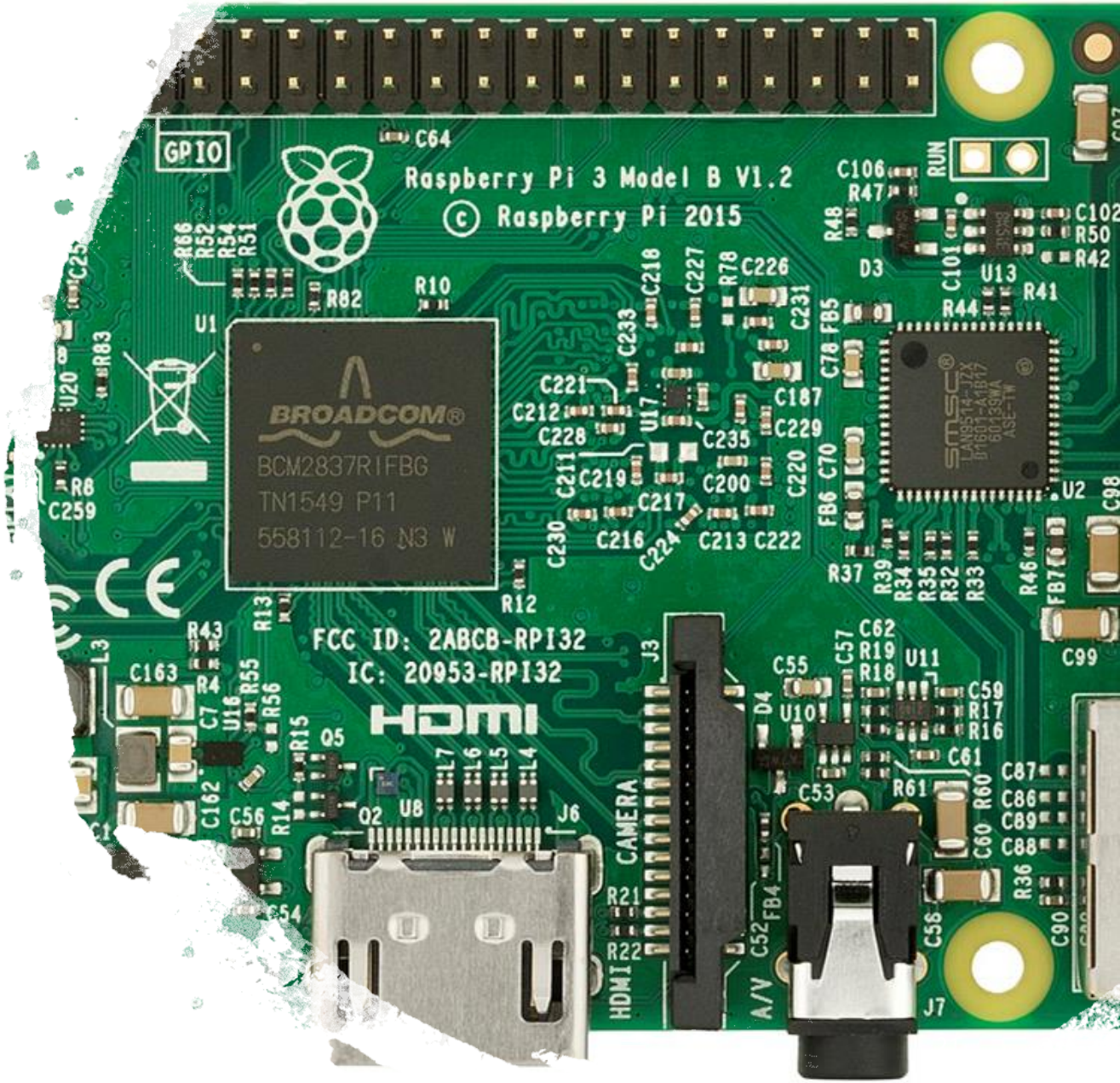
Microcontrollers (e.g. Arduino)

- Simple, low cost
- Runs one software program
- Connect sensors/hardware via GPIO
- Constrained Resources
  - Low speed
  - small memory
  - (usually) no disk
  - No general audio/video/networking (added as needed)
- PWM
- Can have built in Analog to Digital Conversion.



Single Board Computers (e.g. Rpi)

- CPU
- Memory and Storage
- General interfaces for audio/video
- Operating System
- General Purpose Input and Output
- Usually no built in ADC.

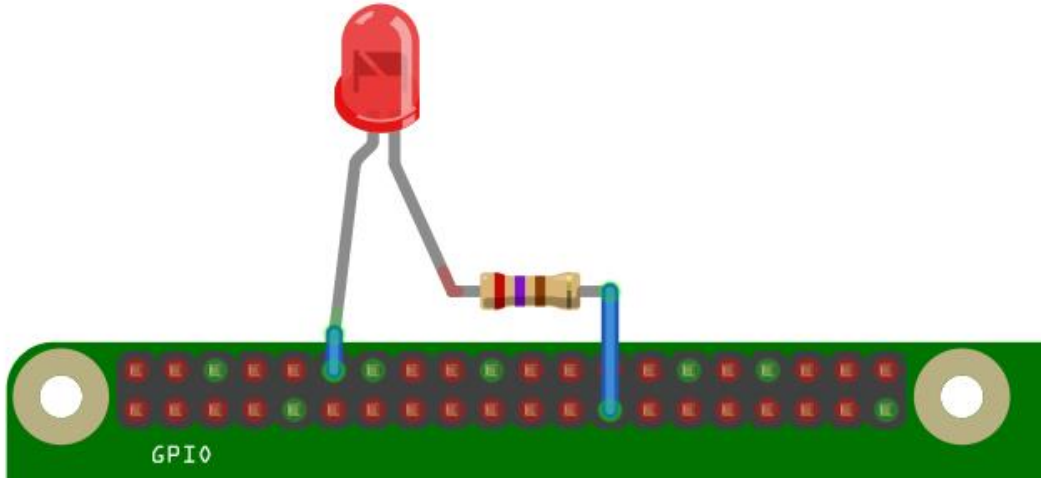


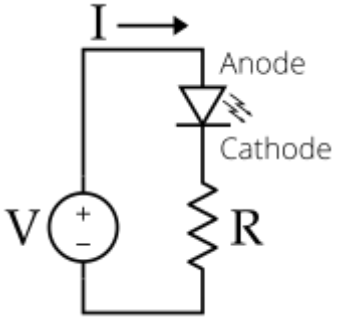
Output and Input on RPi

- Digital Pins
  - Value LOW (0) or HIGH (1)
- Write programs to set pins to Low(0) or High(1)
  - 0V or 3.3V
- Write programs to read pin values
  - High(1)
  - Low(0)

Output

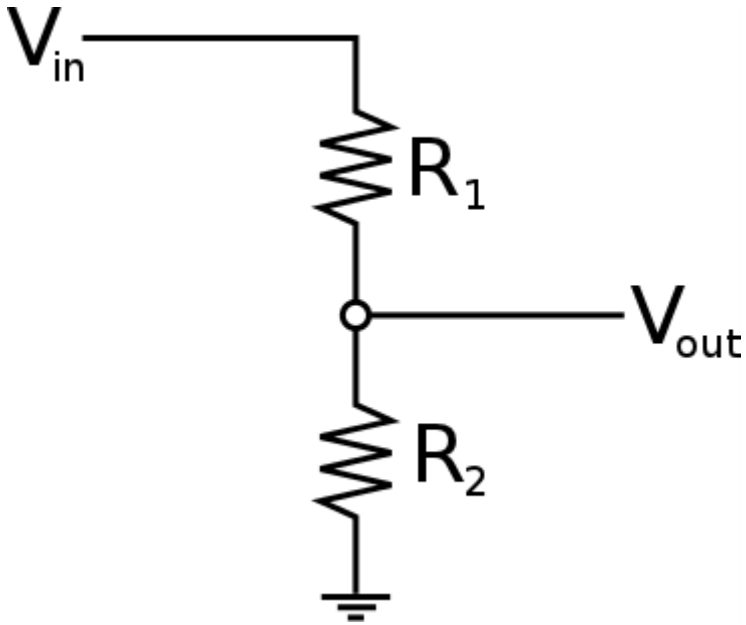
- 3.3V(HIGH) or 0V(LOW)
  - Default 8 mA max per pin.
- Switching a pin High is like connecting a 3.3V battery to device.
- Switching a pin Low is like disconnecting the battery

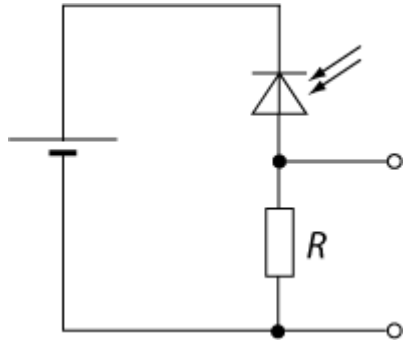




Input

- Can be Voltage Divider Circuit
- Can measure Vout…
  - via analogue input on Arduino
  - Required Analog to Digital converter on RPi. (unless you want it to act like a switch)



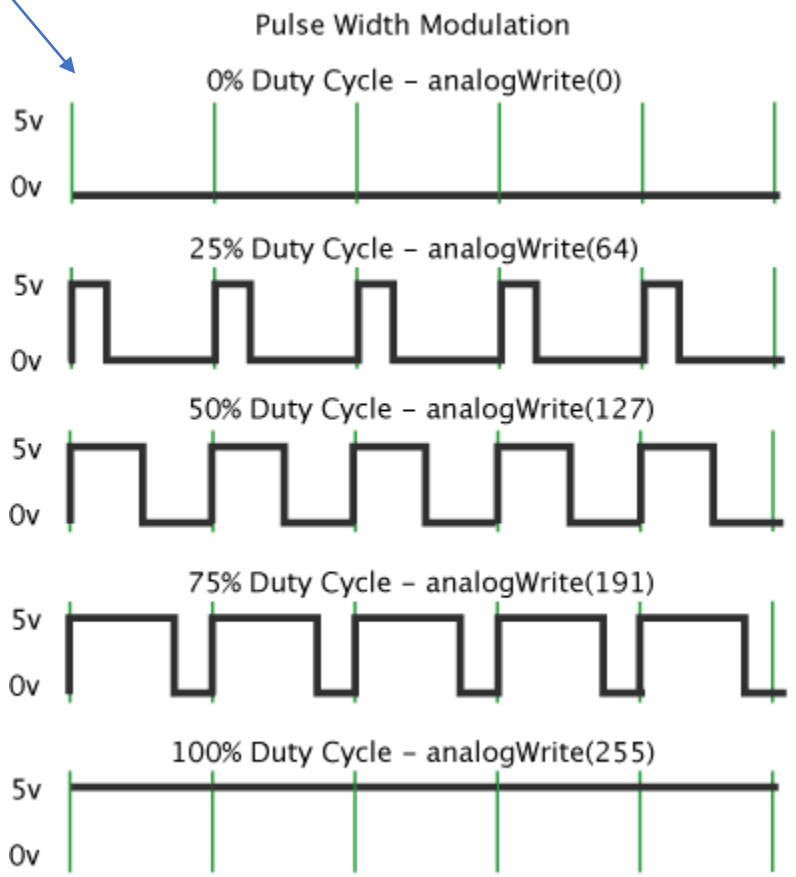


Pulse With Modulation

Pulse Width

- We set the % of “high” cycle
  - 0 – 0%
  - 255 – 100%
    - Typical for Arduino. Depends on the library
- Implementation
  - Hardware
  - Software
- Usage
  - LED dimming
  - Servo Motors

Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each.



Microcontrollers and Computers

Microcontrollers and computers

Firmware

Software