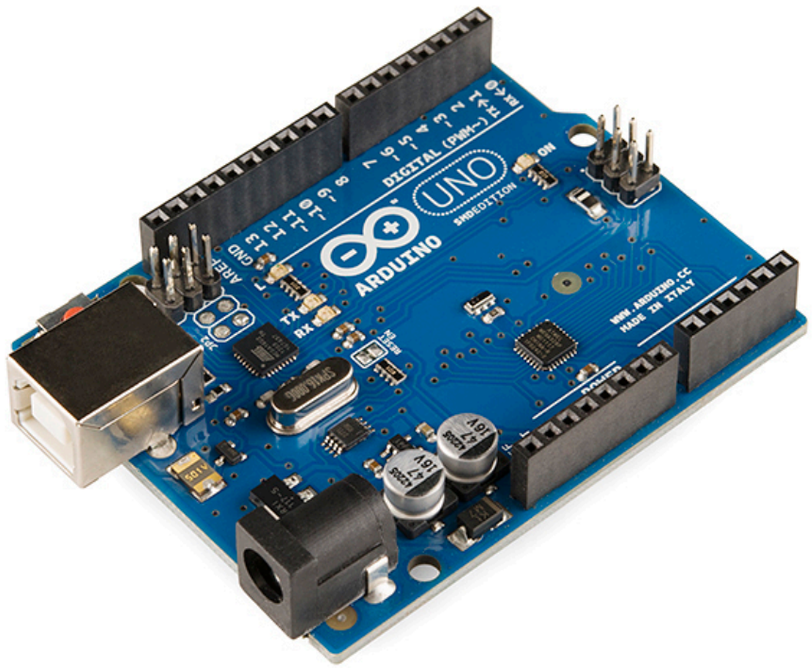
Intro to **Arduino**

DfPI



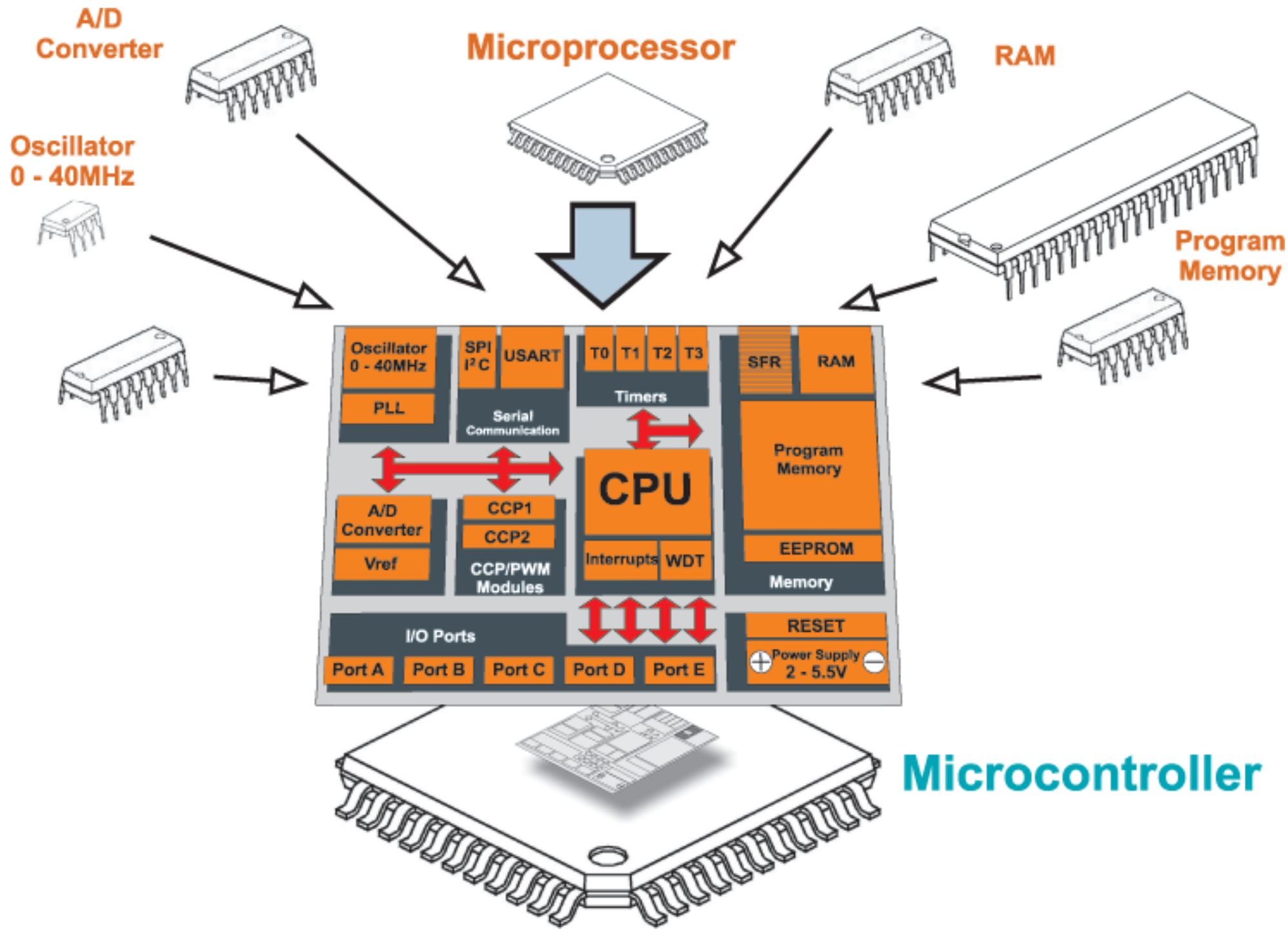
Arduino

- Essentially a mini computer.
- Can control motors, read data from sensors etc…
- Many problems already solved by enormous support community.
- Loads of tutorials online ->

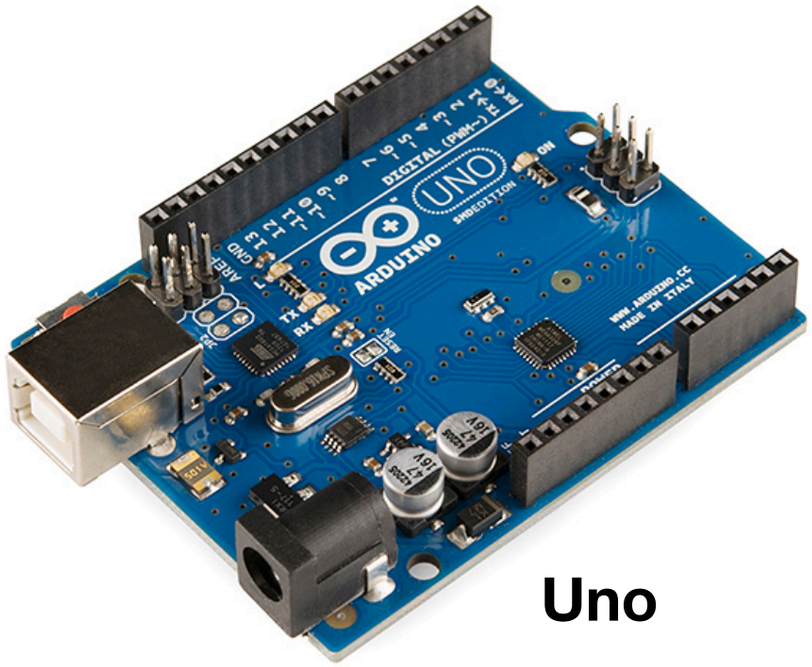
sparkfun™ ELECTRONICS

adafruit

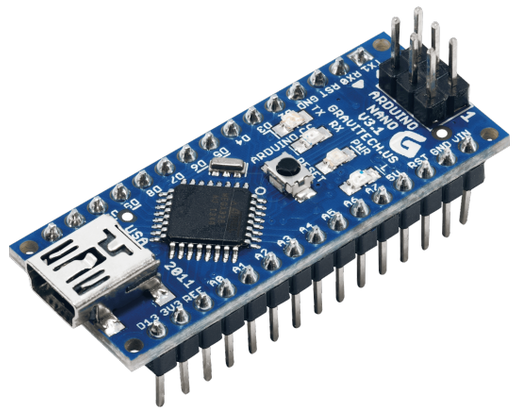
What is a micro controller?



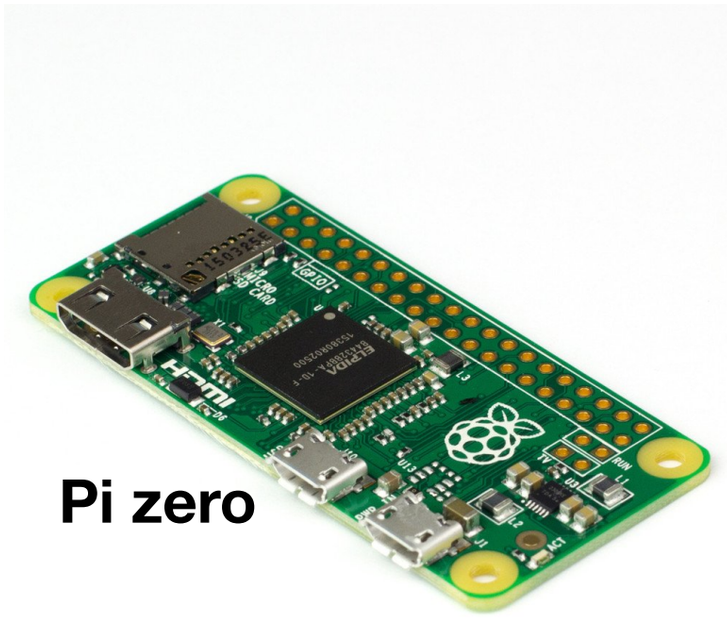
Microcontroller



Uno

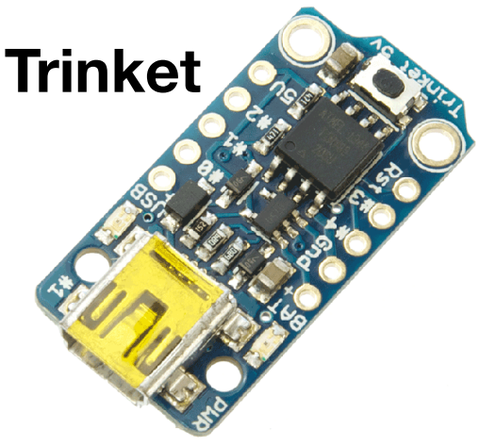


Nano

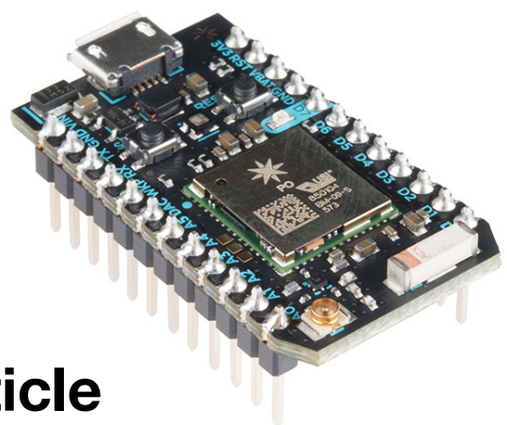


Pi zero

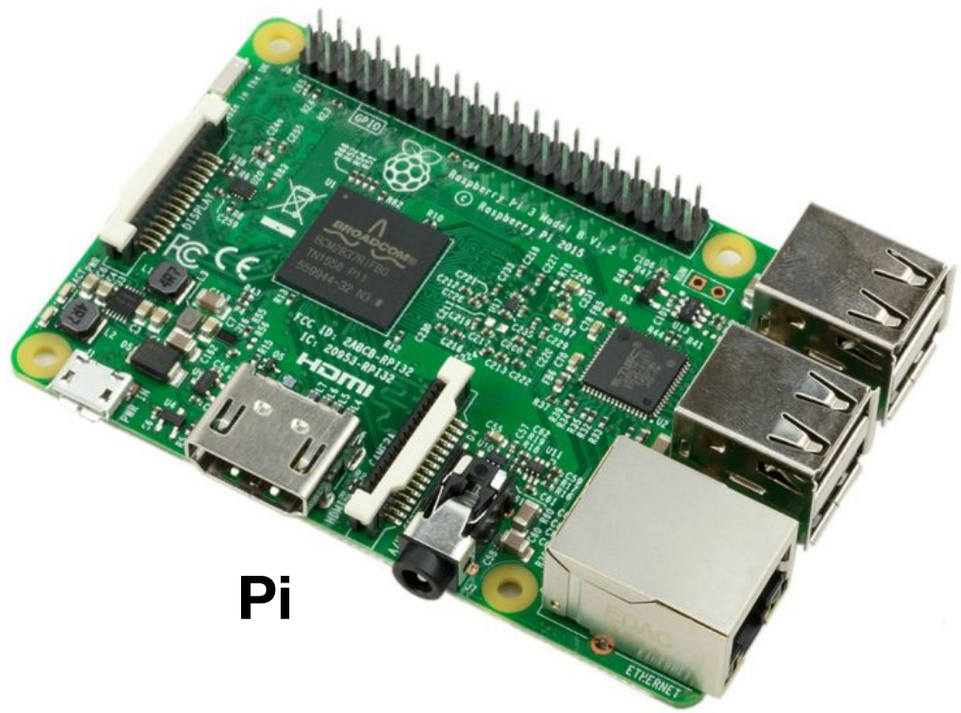
Micro controller Dev. Boards



Trinket

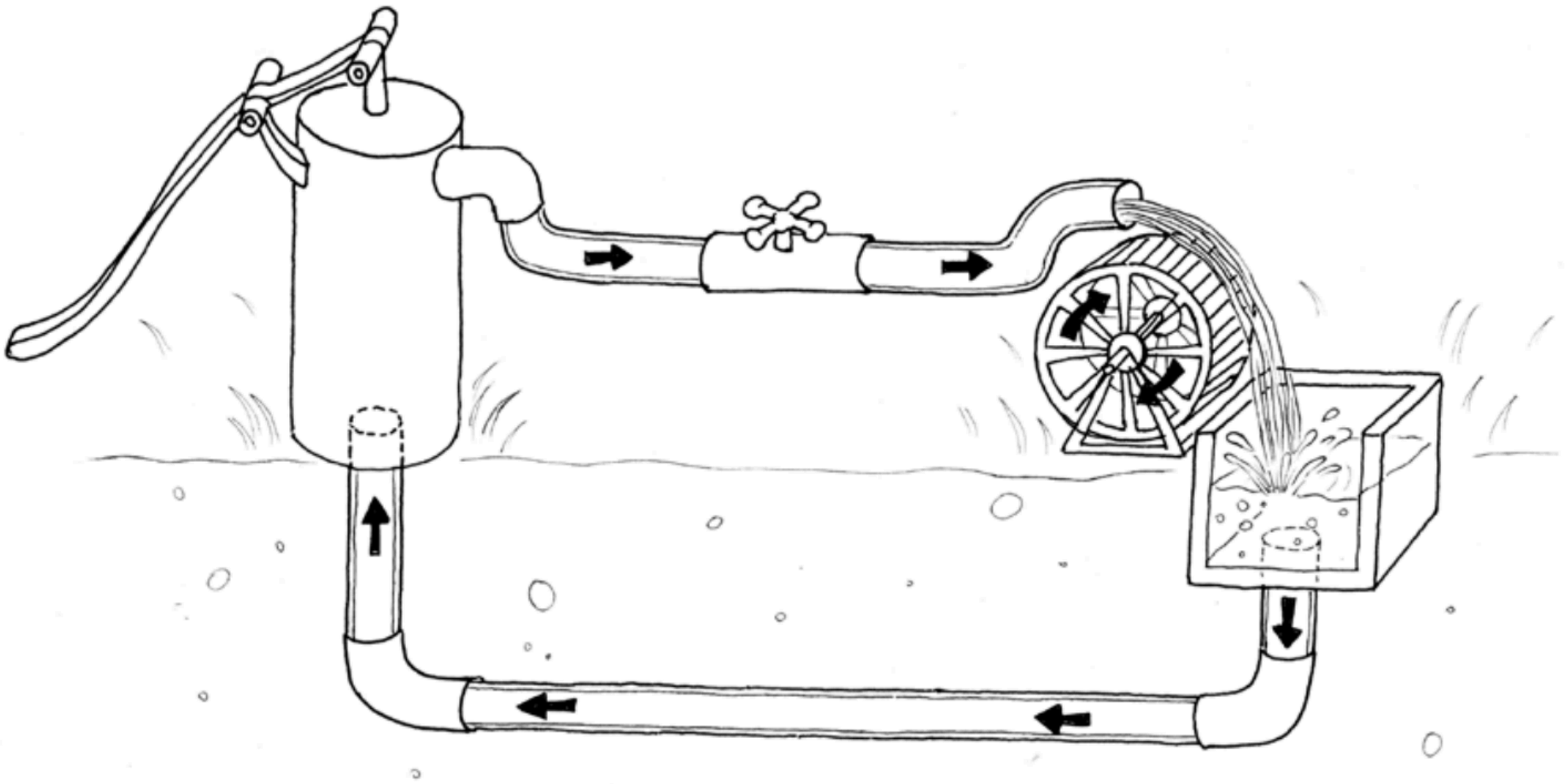


Particle



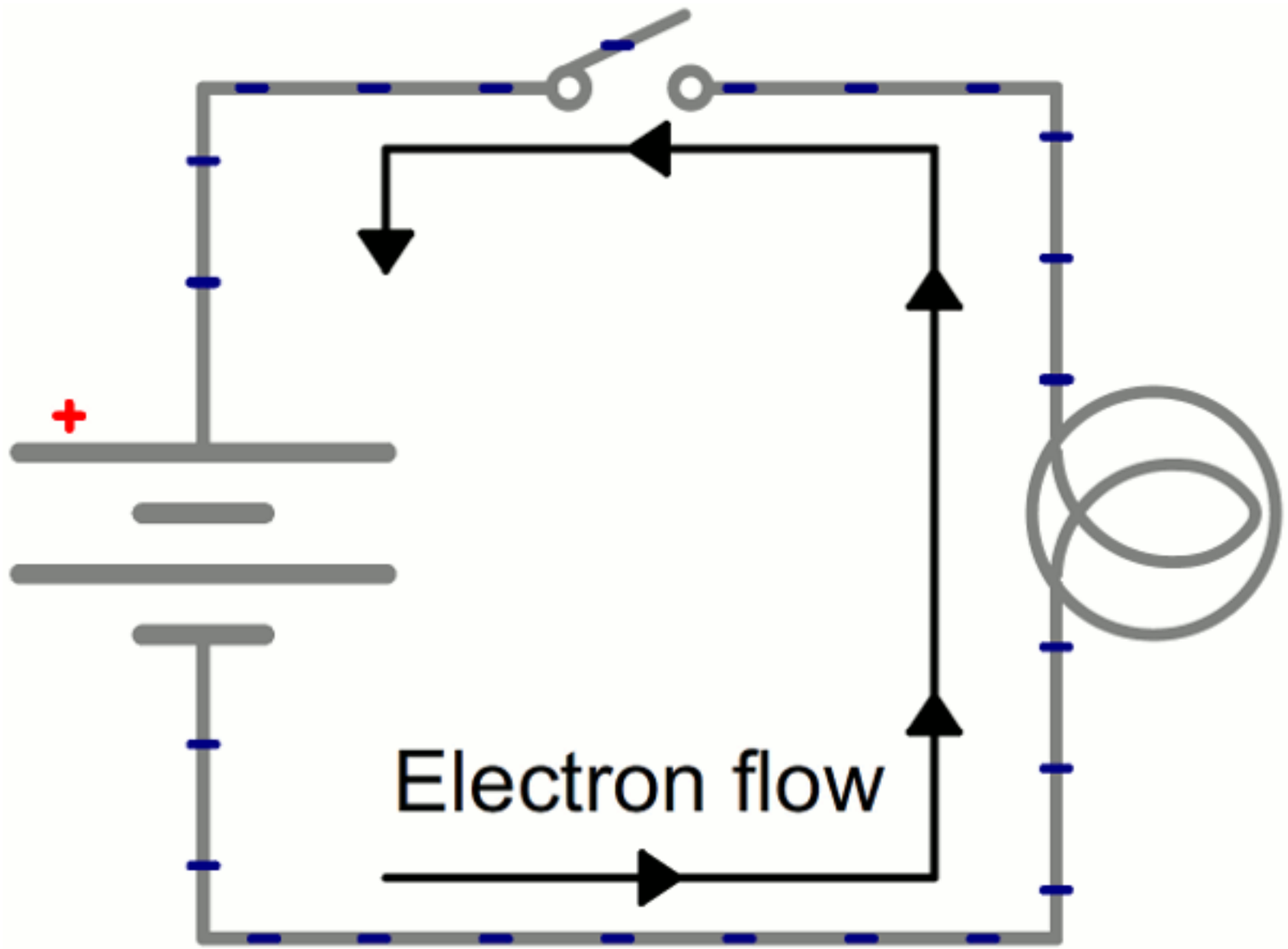
Pi

Electronics revision



There needs to be a complete circuit for current to flow.

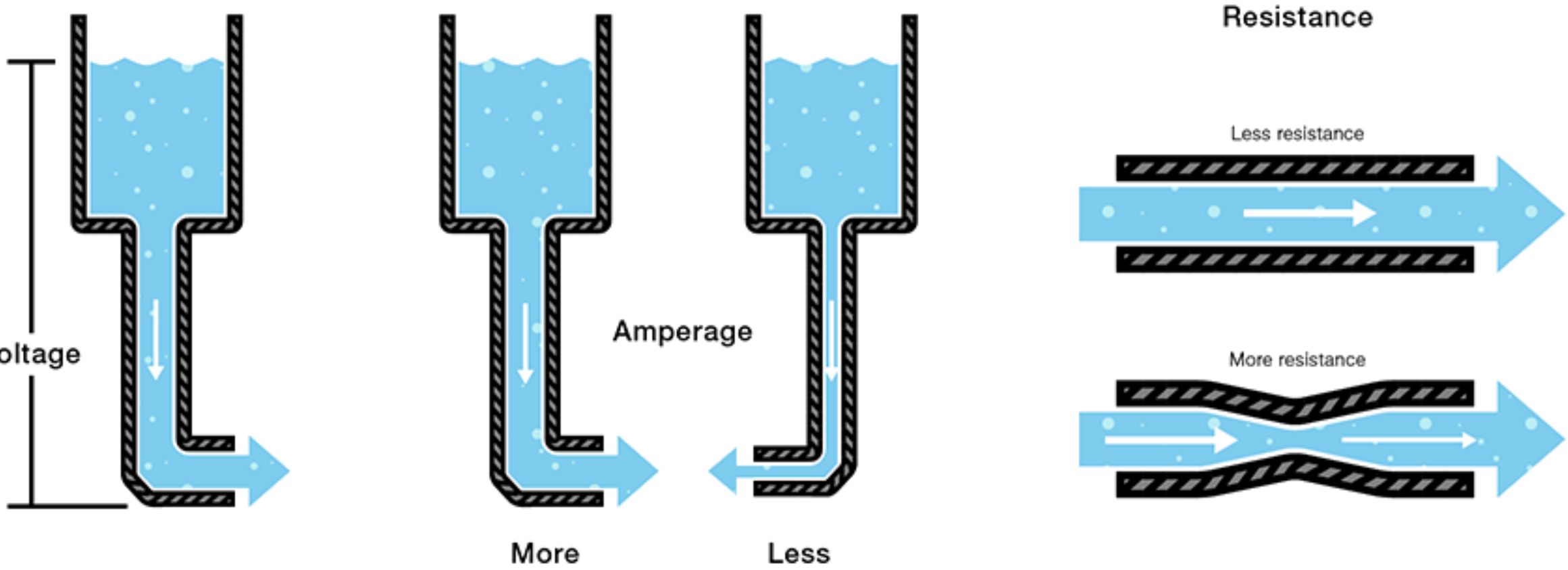
Current flow visualisation



There needs to be a **completed circuit** for current to flow.

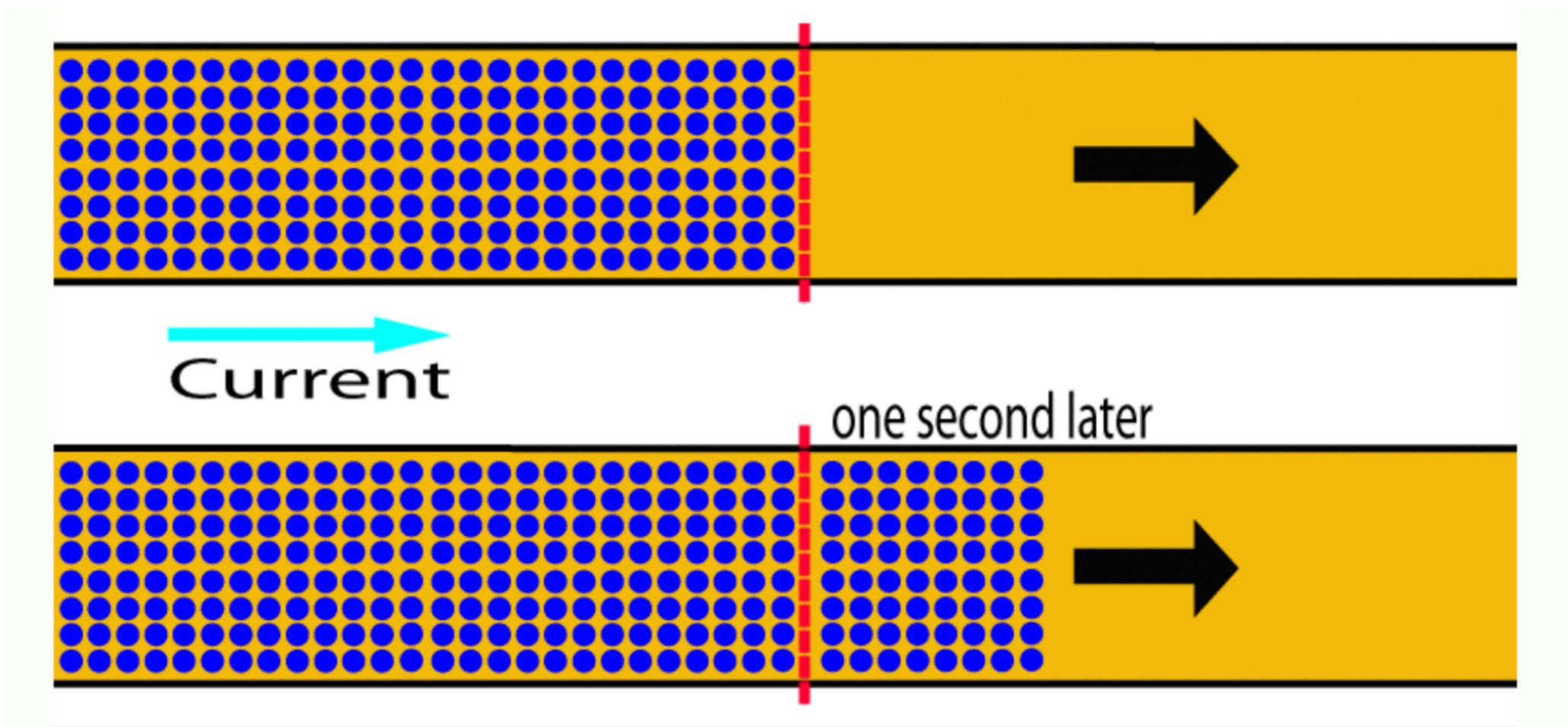
*Does anyone know why the electrons are moving from negative to positive?

Current and Voltage



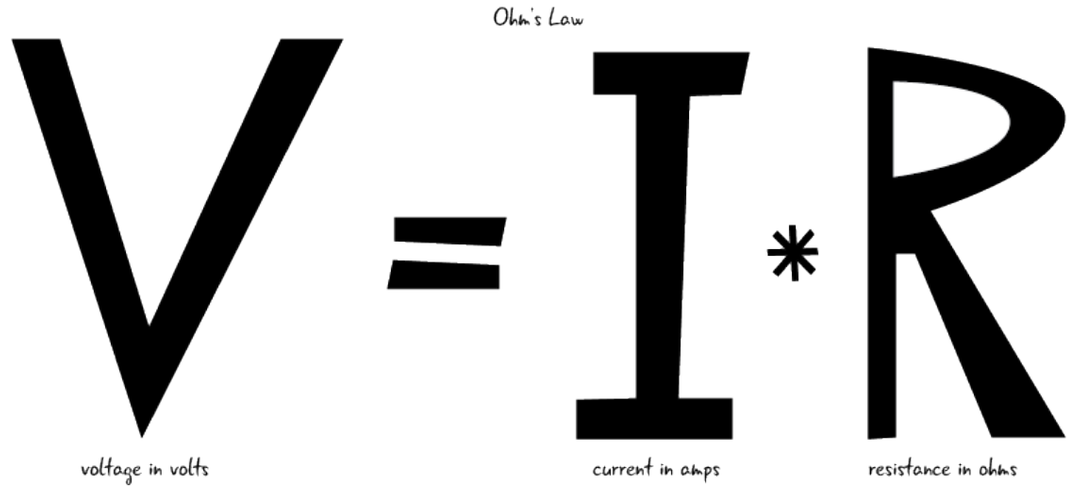
- **Current** - I - Measured in Amperes (A), charge per unit time.
- **Voltage** - V - Measured in Volts (V), Electro motive force, EMF.
- **Resistance** - R - Resistance measured in Ohms, (Omega)

Visualisation of electrical current:

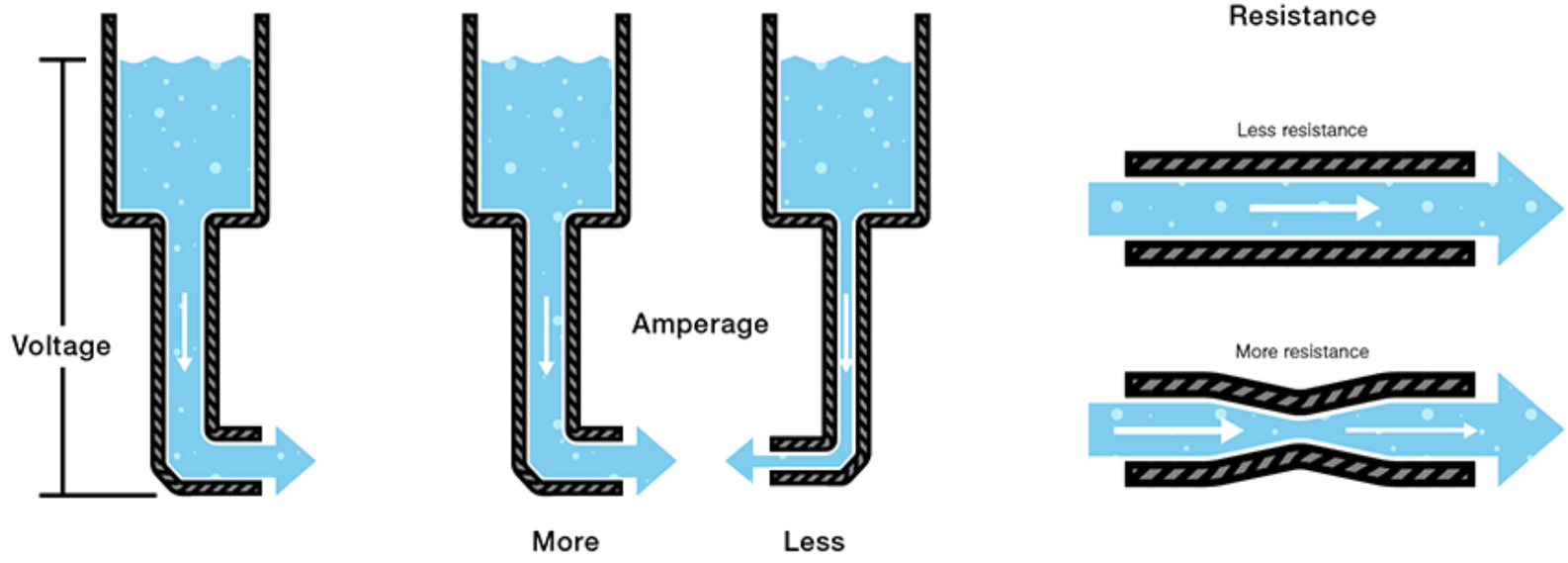


- **Amperes** are a measure of the rate of charge flow through an electrical circuit.

Super important Law!



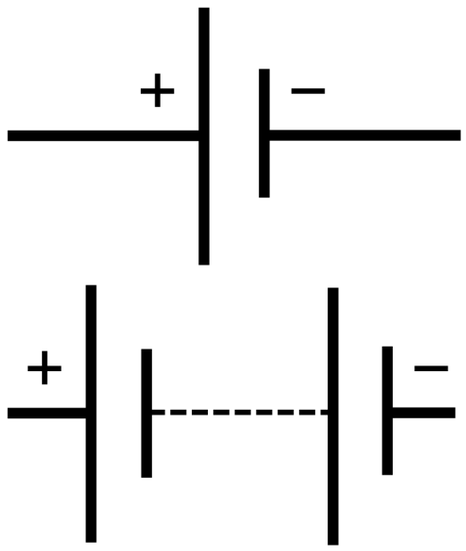
- **V** = Voltage
- **I** = Current
- **R** = Resisance



Usually you will only be varying the **Voltage** or **Resistance** in a circuit, and this will govern how the **Current** behaves.

Schematics & Symbols

Resistor

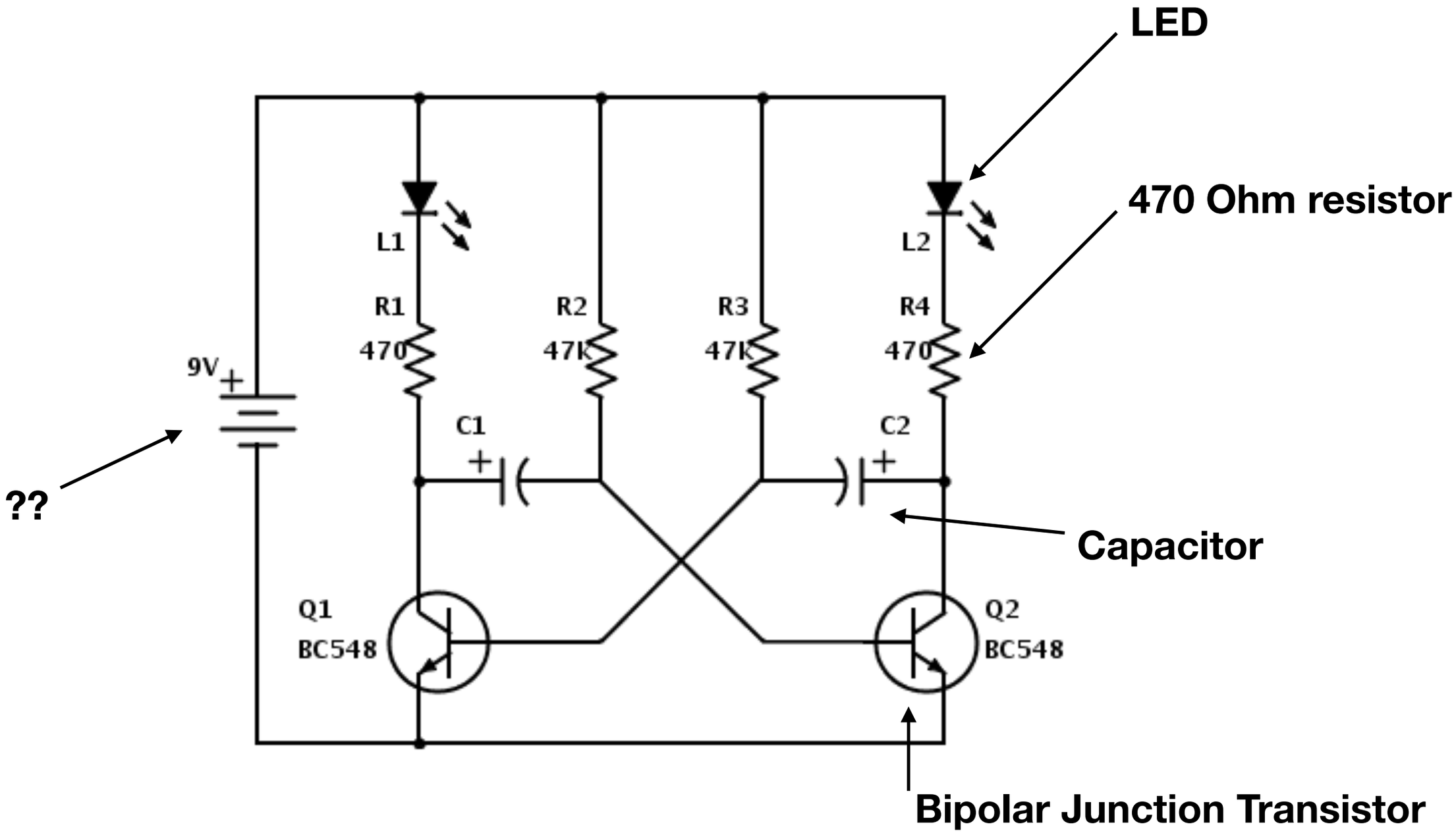


Battery

Ground

These are just a few of the essential symbols

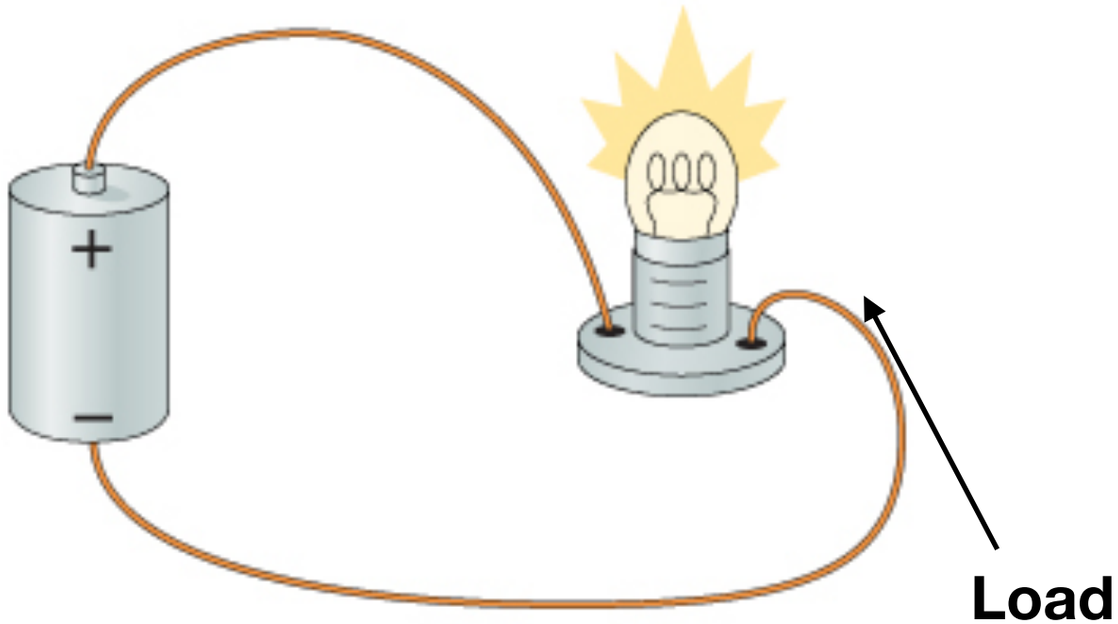
Electrical **Schematics**



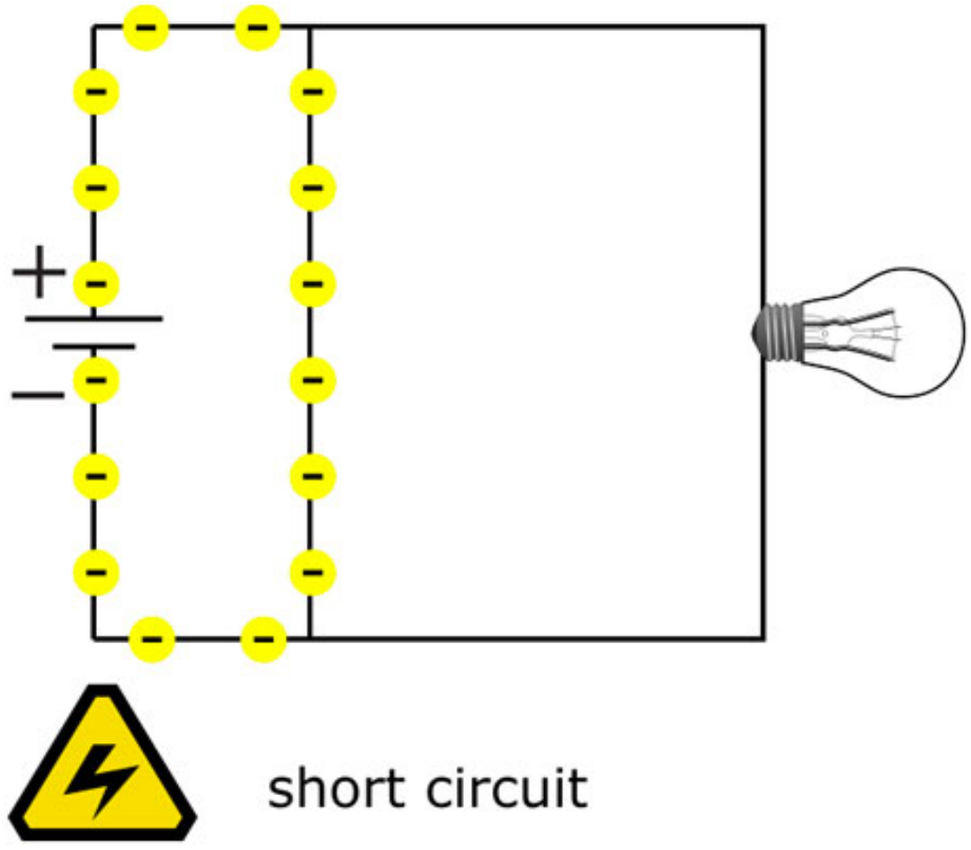
With a bit of practice you start to recognise all the components

Short Circuits

Good



Bad



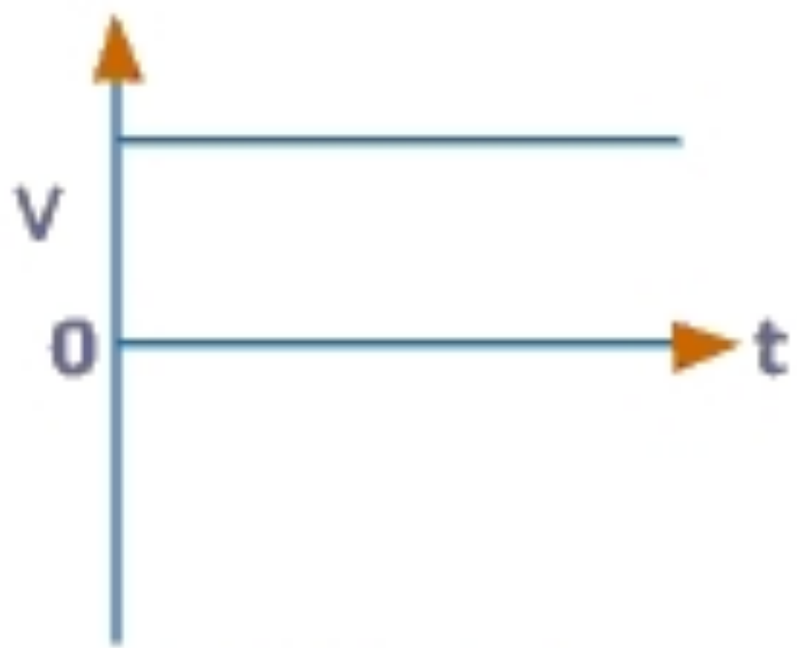
Regular circuit has wires connecting components in a loop, however if there is ever a case of a loop with **NO LOAD** then this is called a short circuit. (Load can be Motor, Light, … etc.)

Watch out for this with practical work!

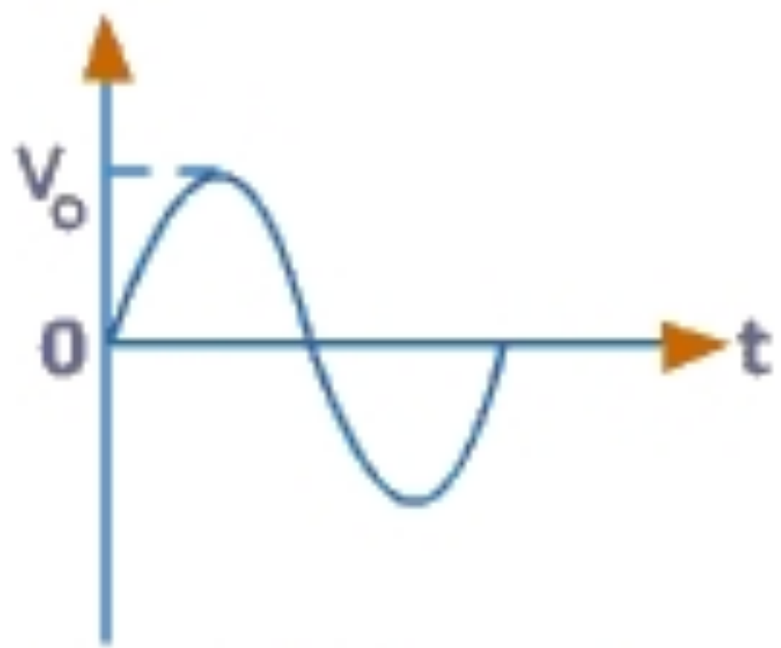
Types of electricity

- **AC** - Alternating current - Voltage supplied varies when supplied to electrical device. For mains and **high power** electronics.

- **DC** - Direct current - Voltage supply is held constant, You will be dealing mainly with **DC** electronics with voltages ranging from 0 - 12V.

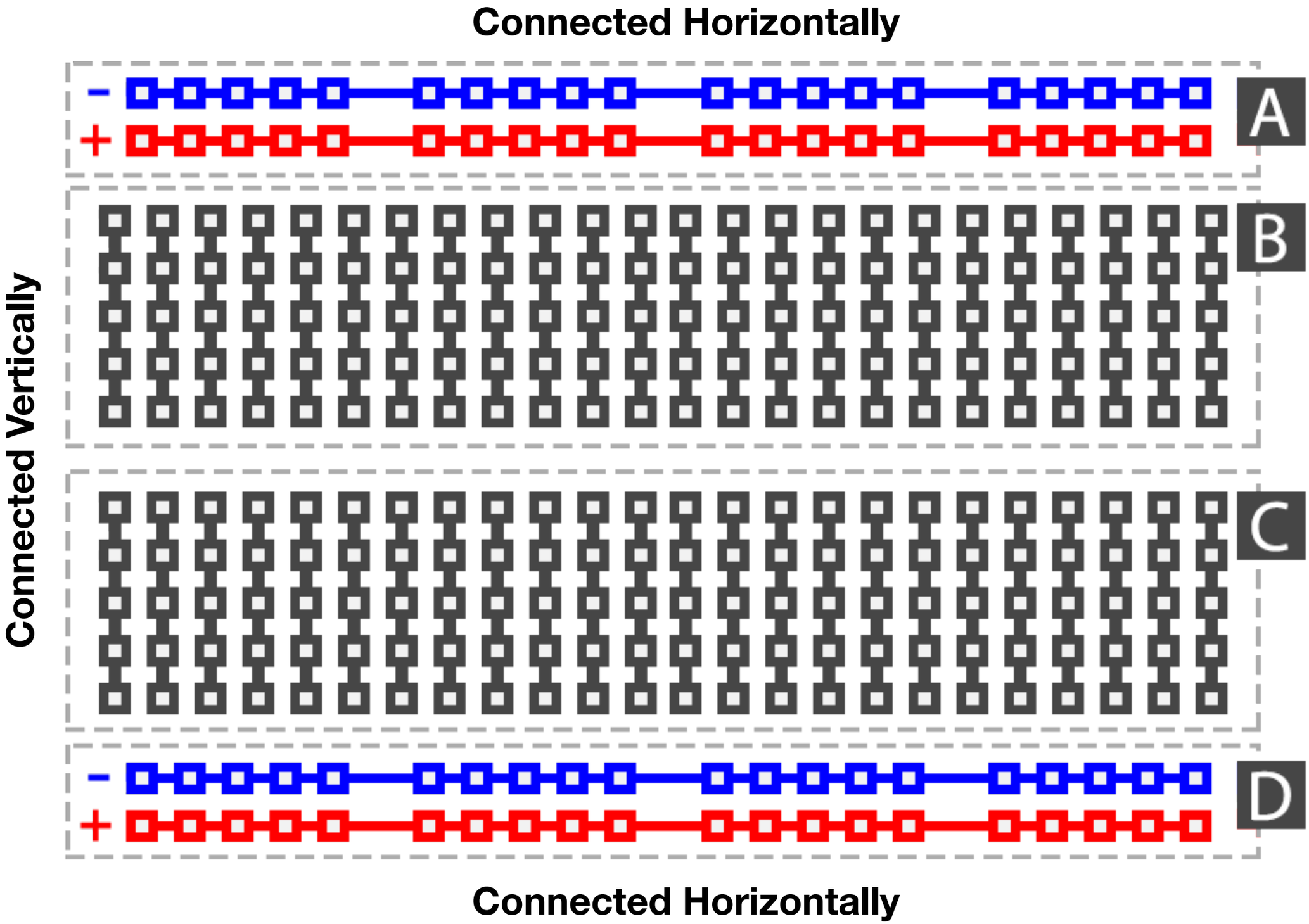


DC Source



AC Source

How a **solderless** breadboard works



Exercise 1.

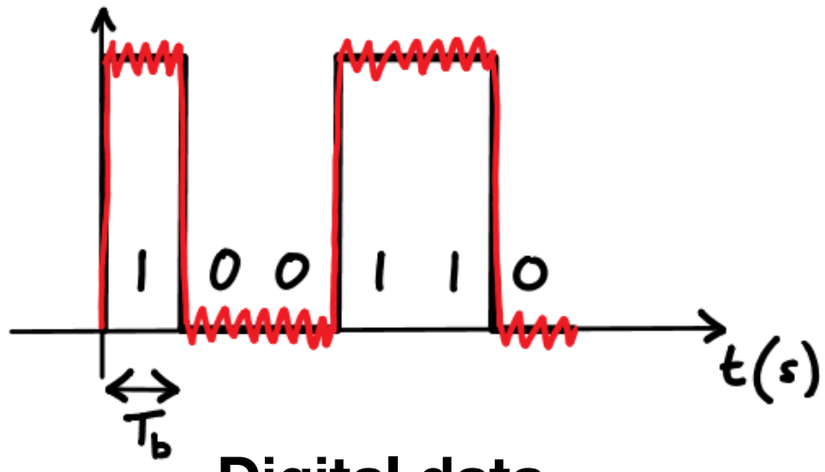
Digital Input - Any Digital Pin on the **Arduino** is capable of recognising a binary change in voltage **Eg:** A change in voltage from **0V** - **5V.**

***Note** Any voltage on the input pin greater than **1.5V** will be recognised as a **Binary 1** or **HIGH,** and anything below will be a **Binary 0** or **LOW.**

Examples of **digital** inputs.

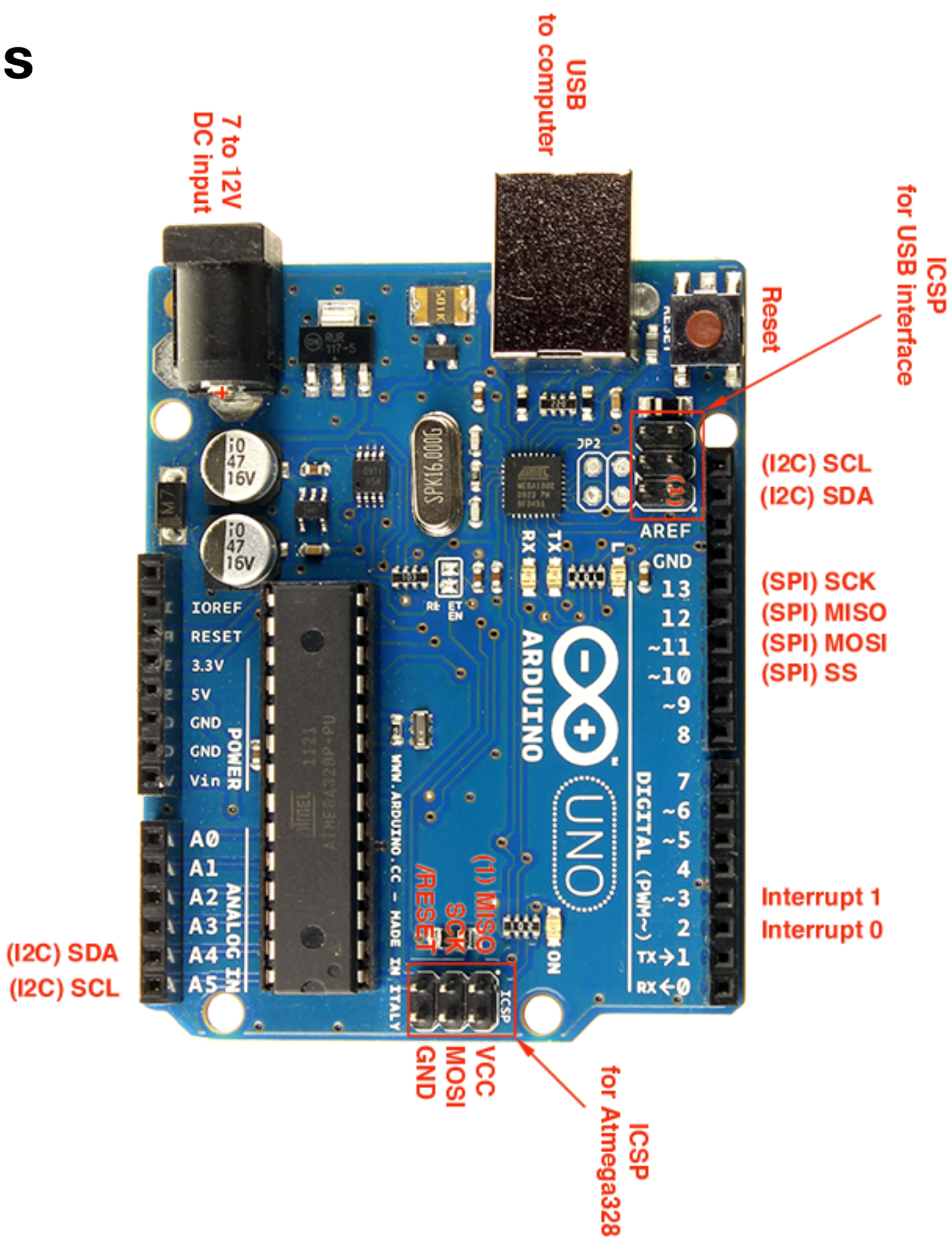
Micro Switch

Push button



Digital data

Arduino Pins



UNO PINOUT

7-12V Depending on current drawn

GND

VIN

2.1mm

ATMEGA 82U/16U2 ICSP

OC0A OC1C PCINT7 PB7 21

PCINT6 PB6 20

19 PB5 PCINT5

18 PB4 PCINT4 T1

USB JACK Type B

DW RESET PC1 24

15 PB1 PCINT1 SCLK

17 PB3 PDO PCINT3 MISO

GND

5V

16 PB2 PDI PCINT2 MOSI

RESET BUTTON

Absolute MAX per pin 40mA
recommended 20mA

Absolute MAX 200mA
for entire package

IOREF provides a logic reference voltage for shields that use it. It is connected to the 5V bus.

R3 Only

NC

IOREF

PCINT14 RESET PC6 1

3V3

5V

GND

GND

VIN

The input voltage to the board when it is running from external power. Not USB bus power.

POWER

IOREF
RESET
3V3
5V GND GND VIN

ANALOG IN

A0 A1 A2 A3 A4 A5

PCINT8 ADC0 PC0 23

PCINT9 ADC1 PC1 24

PCINT10 ADC2 PC2 25

PCINT11 ADC3 PC3 26

SDA PCINT12 ADC4 PC4 27

SCL PCINT13 ADC5 PC5 28

Cut to disable autoreset

ICSP

ON

AREF
GND
13
12
#11
#10
#9
8

DIGITAL [#=PWM]

7
#6
#5
4
#3
2
TX 1
RX 0

28 PC5 ADC5 PCINT13 SCL

27 PC4 ADC4 PCINT12 SDA

21 AREF

GND

19 PB5 PCINT5 SCK

18 PB4 PCINT4 MISO

17 PB3 OC2A PCINT3 MOSI

16 PB2 OC1B PCINT2 SS

15 PB1 OC1A PCINT1

14 PB0 CLKO PCINT0 ICP1

13 PD7 AIN1 PCINT23

12 PD6 AIN0 PCINT22 OC0A

11 PD5 T1 PCINT21 OC0B

6 PD4 T0 PCINT20 XCK

5 PD3 INT1 PCINT19 OC2B

4 PD2 INT0 PCINT18

3 PD1 TXD PCINT17

2 PD0 RXD PCINT16

R3 Only

Connected to the ATMega and used for USB program and communicating with it

Power
GND
Serial Pin
Analog Pin
Control
INT
Physical Pin
Port Pin
Pin function
Interrupt Pin
PWM Pin
Port Power

PCINT14 RESET PC6 1

19 PB5 PCINT5 SCK

18 PB4 PCINT4 MISO

GND

5V

17 PB3 OC2A PCINT3 MOSI

The power sum for each pin's group should not exceed 100mA

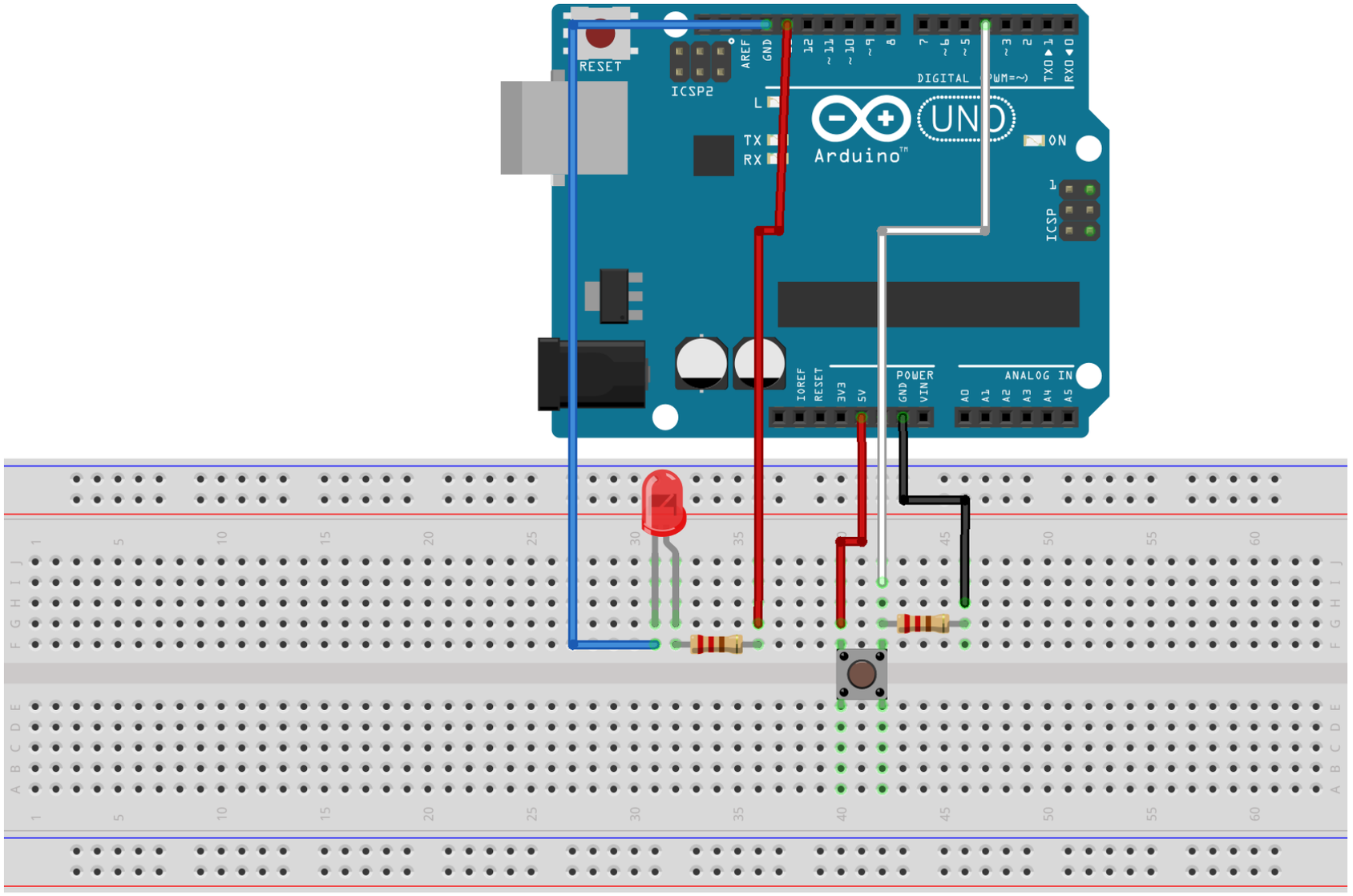
bq

www.bq.com

17 JUL 201

ver 3 rev

The Circuit

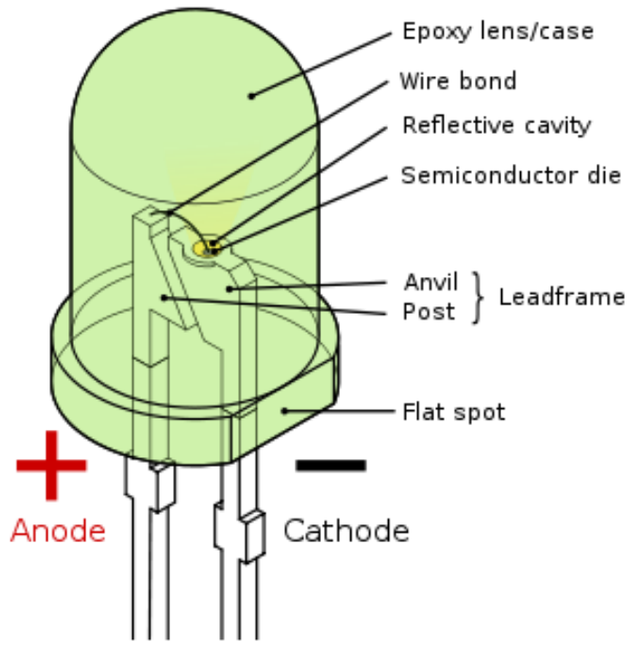


Push button as **Digital input,** and LED as **Digital output.**

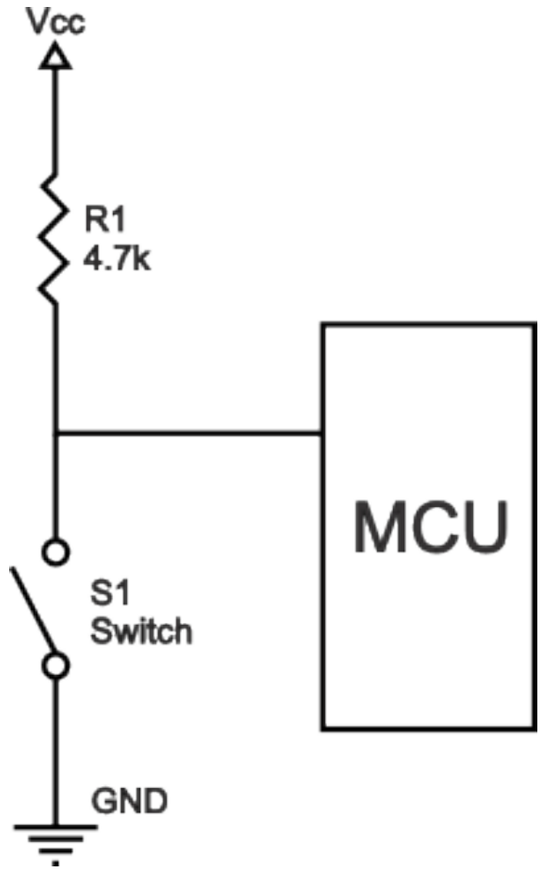
Does anyone know what the **resistors** are for?

*Anyone want to try the challenge?

Current limiting for LED - LED’s will draw current until they explode if you do not have this resistor. Also you can’t plug them in **Backward!**



Pull up resistor - Used to define the digital input value when the button is left as an open switch. If you were to leave the input pin **“floating”** the voltage can fluctuate and cause weird behaviours in your circuit.



Arduino File Edit Sketch Tools Help 100%

Digital_Read_Example | Arduino 1.6.10

Digital_Read_Example

```
//DfPI Tutorial Code
//Sean Malikides
//Oct 2017

/*
Code Function Description

  - Reads a digital input on pin 2, prints the result to the serial monitor.
  - Uses push button to turn a light on and off.

*/

// declare variable "pushButton" as an integer, and assign it the value 2, for later pin assignment.
int pushButton = 2;

// constants won't change. Used here to set a pin number.
const int ledPin =  13;      // the number of the LED pin

// the setup routine runs once when you press reset.
void setup() {

  // initialize serial communication at 9600 bits per second.
  Serial.begin(9600);

  // make the pushbutton's pin an input.
  pinMode(pushButton, INPUT);

  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);

}

// the loop routine runs over and over again forever.
```

Computers **usually** read through your code sequentially.

Comments (For Humans)

1.

2.

3.

Arduino File Edit Sketch Tools Help

Digital_Read_Example | Arduino 1.6.10

Digital_Read_Example

```
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);

}

// the loop routine runs over and over again forever.
void loop() {

  // read the input pin.
  int buttonState = digitalRead(pushButton);

  // print out the state of the button, So you can see what the state of the pushbutton is.
  Serial.println(buttonState);

  if (buttonState == HIGH) {
    //Turn the LED on.
    digitalWrite(ledPin, HIGH);

  } else {

    //Turn the LED off.
    digitalWrite(ledPin, LOW);

  }

  //just a little delay for stability.
  delay(10);

}
```

This **Loop** section of code runs forever!

If statement like a logical switch.

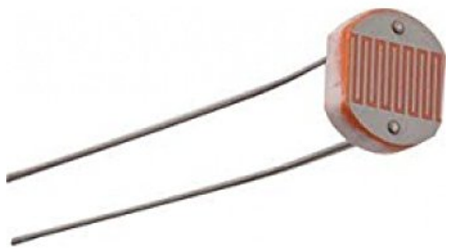
Wait for a little bit, Delay(t) where “t” is in milliseconds.

Exercise 2.

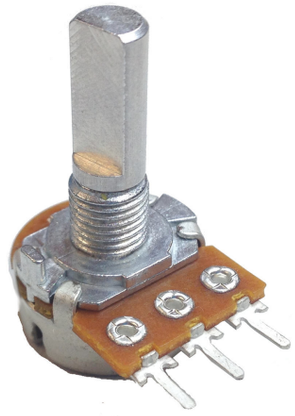
Analog Input - Any **Analog Pin** on the **Arduino** is capable of registering a Voltage variation within the range of **0V - 5V** with a resolution of 12 bits (ie. 0 - 1023 integer range).

Analog output - On the Arduino **Uno** you cannot generate a pure analog output. Instead you fake it, by using **PWM.**

Examples of **Analog** inputs.



Light Dependant Resistor



Potentiometer

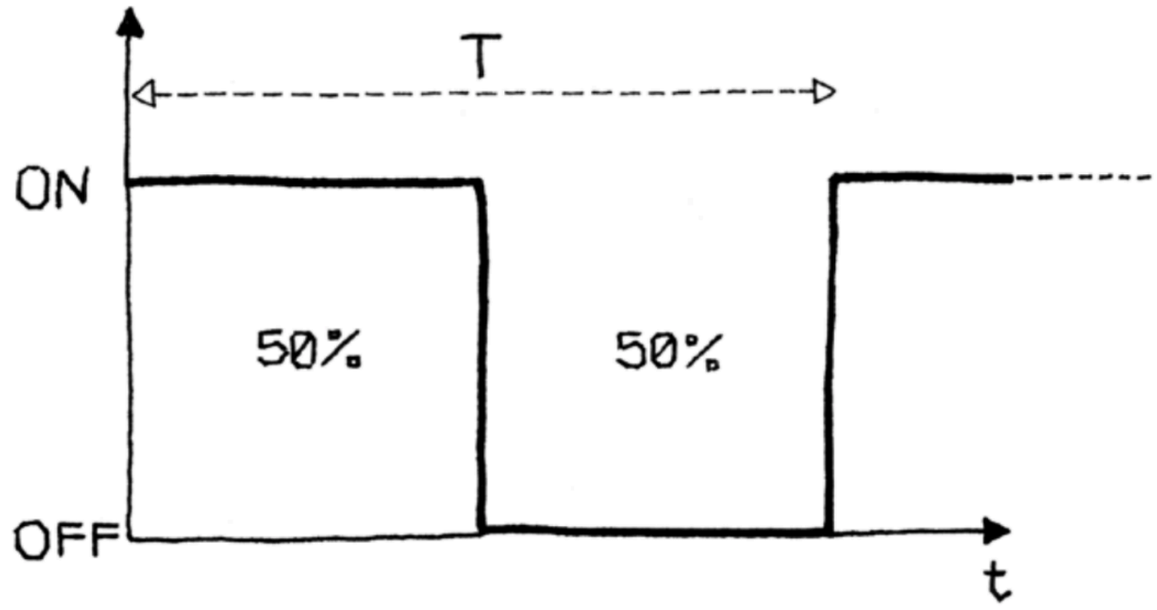
Microphone

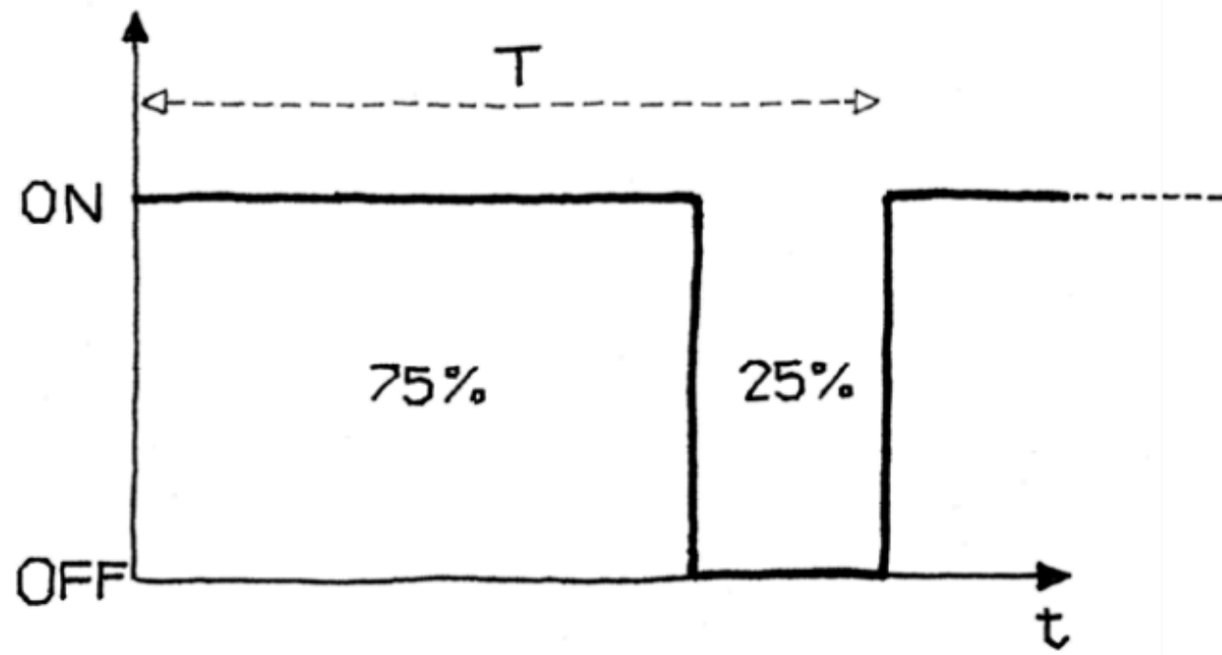
PWM - Pulse Width Modulation

Effectively the **average voltage** over time is dependant on the Ontime:Offtime ratio of the signal.

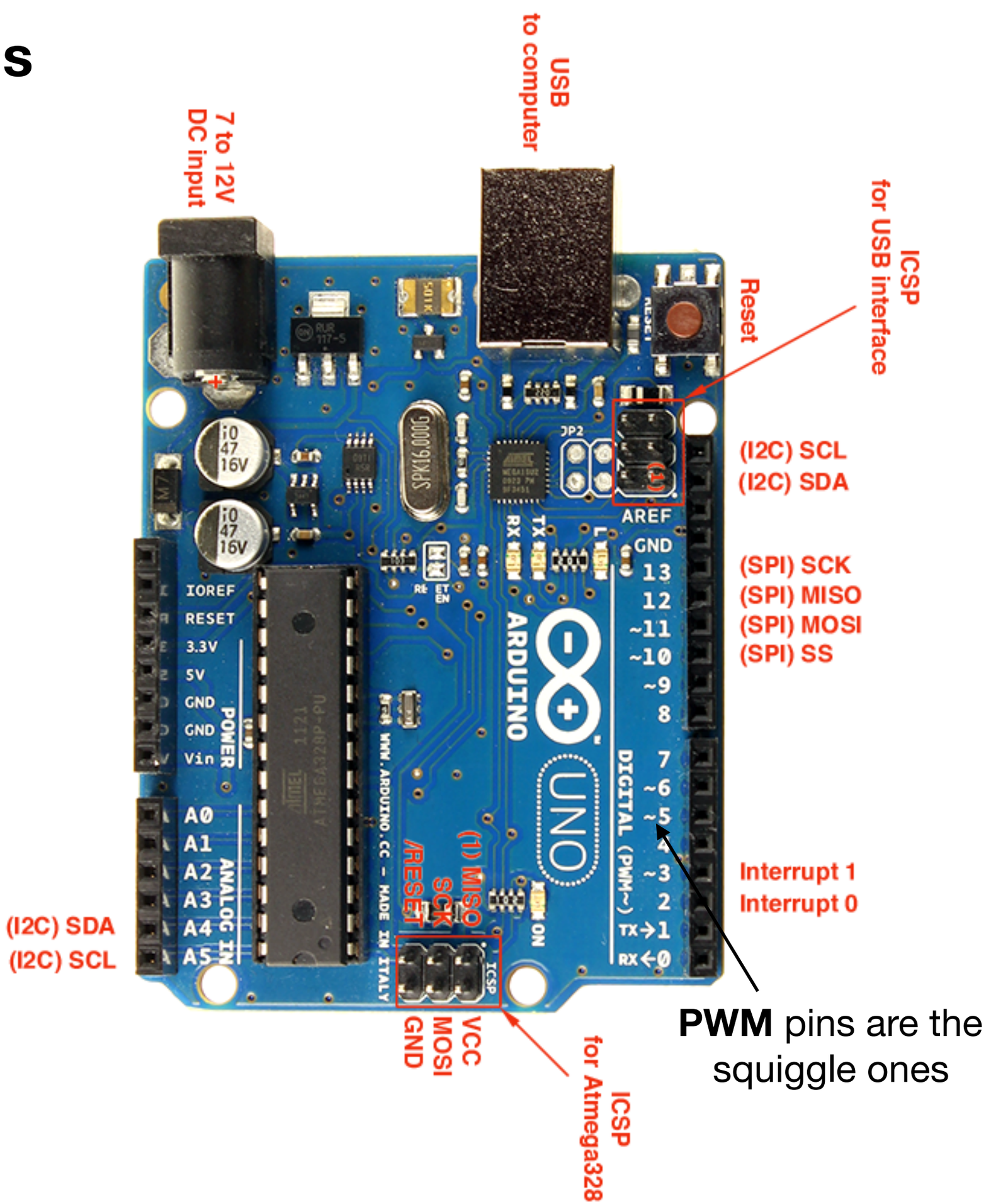
This effect is obvious when used on something like an **LED,** where you can dim the output brightness of the **LED** by changing the PWM **“Duty Cycle”.**

Only specific pins support PWM on the Arduino Uno.





Arduino Pins



UNO PINOUT

7-12V Depending on current drawn

GND

VIN

2.1mm

ATMEGA 82U/16U2 ICSP

OC0A OC1C PCINT7 PB7 21

PCINT6 PB6 20

19 PB5 PCINT5

18 PB4 PCINT4 T1

USB JACK Type B

DW RESET PC1 24

15 PB1 PCINT1 SCLK

17 PB3 PDO PCINT3 MISO

GND

5V

16 PB2 PDI PCINT2 MOSI

RESET BUTTON

Absolute MAX per pin 40mA recommended 20mA

Absolute MAX 200mA for entire package

IOREF provides a logic reference voltage for shields that use it. It is connected to the 5V bus.

R3 Only

NC

IOREF

PCINT14 RESET PC6 1

3V3

5V

GND

GND

VIN

The input voltage to the board when it is running from external power. Not USB bus power.

POWER

IOREF RESET 3V3 5V GND GND VIN

ANALOG IN

A0 A1 A2 A3 A4 A5

PCINT8 ADC0 PC0 23

PCINT9 ADC1 PC1 24

PCINT10 ADC2 PC2 25

PCINT11 ADC3 PC3 26

SDA PCINT12 ADC4 PC4 27

SCL PCINT13 ADC5 PC5 28

Cut to disable autoreset

ICSP

ON

TX

RX

AREF

GND

DIGITAL [#=PWM]

13 12 #11 #10 #9 8

7 #6 #5 4 #3 2 TX 1 RX 0

28 PC5 ADC5 PCINT13 SCL

27 PC4 ADC4 PCINT12 SDA

21 AREF

GND

19 PB5 PCINT5 SCK

18 PB4 PCINT4 MISO

17 PB3 OC2A PCINT3 MOSI

16 PB2 OC1B PCINT2 SS

15 PB1 OC1A PCINT1

14 PB0 CLKO PCINT0 ICP1

13 PD7 AIN1 PCINT23

12 PD6 AIN0 PCINT22 OC0A

11 PD5 T1 PCINT21 OC0B

6 PD4 T0 PCINT20 XCK

5 PD3 INT1 PCINT19 OC2B

4 PD2 INT0 PCINT18

3 PD1 TXD PCINT17

2 PD0 RXD PCINT16

R3 Only

Connected to the ATMega and used for USB program and communicating with it

Power

GND

Serial Pin

Analog Pin

Control

INT

Physical Pin

Port Pin

Pin function

Interrupt Pin

PWM Pin

Port Power

PCINT14 RESET PC6 1

19 PB5 PCINT5 SCK

18 PB4 PCINT4 MISO

GND

5V

17 PB3 OC2A PCINT3 MOSI

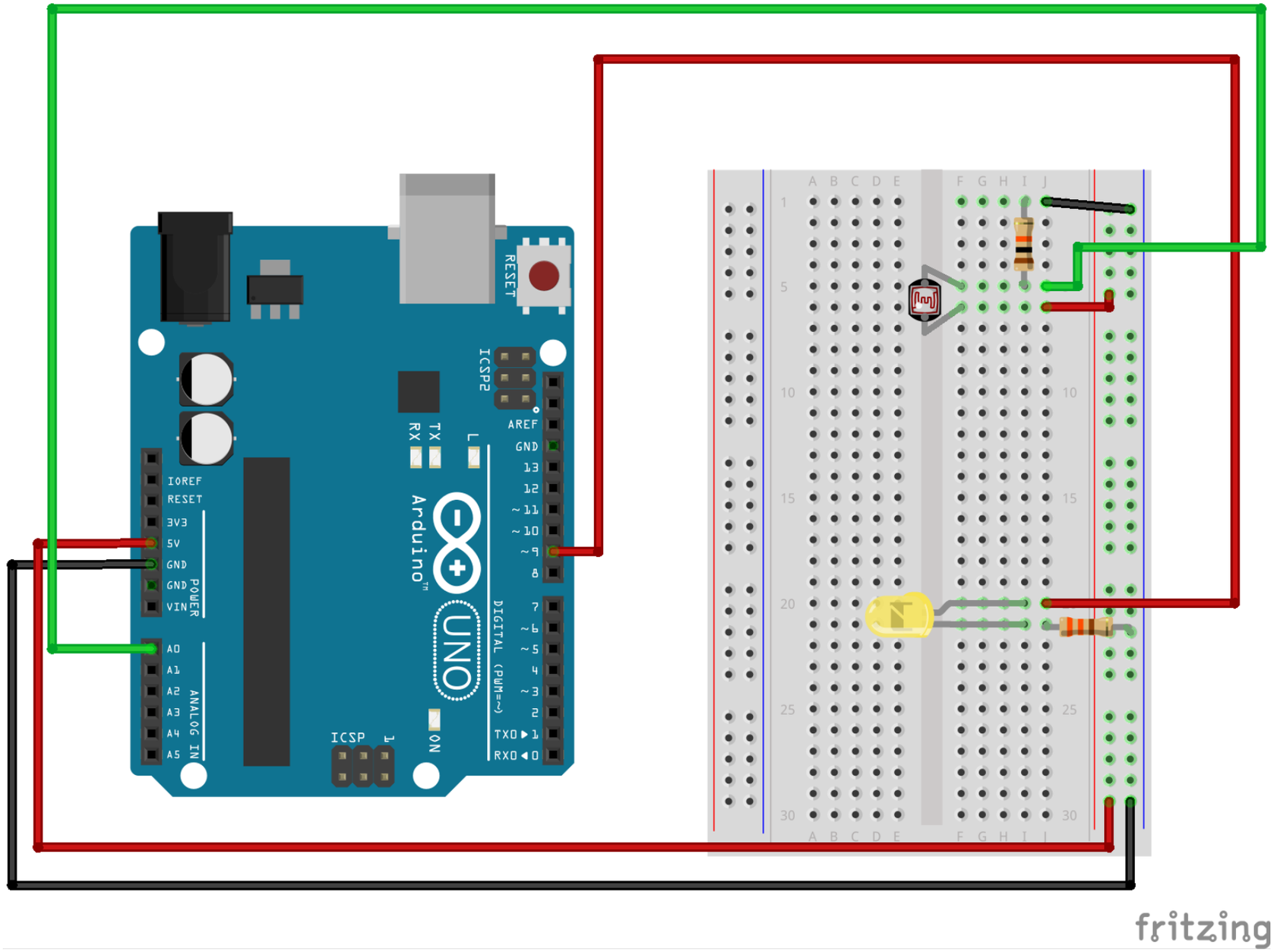
The power sum for each pin's group should not exceed 100mA

PWM pins are the squiggle ones

17 JUL 201

ver 3 rev

The Circuit



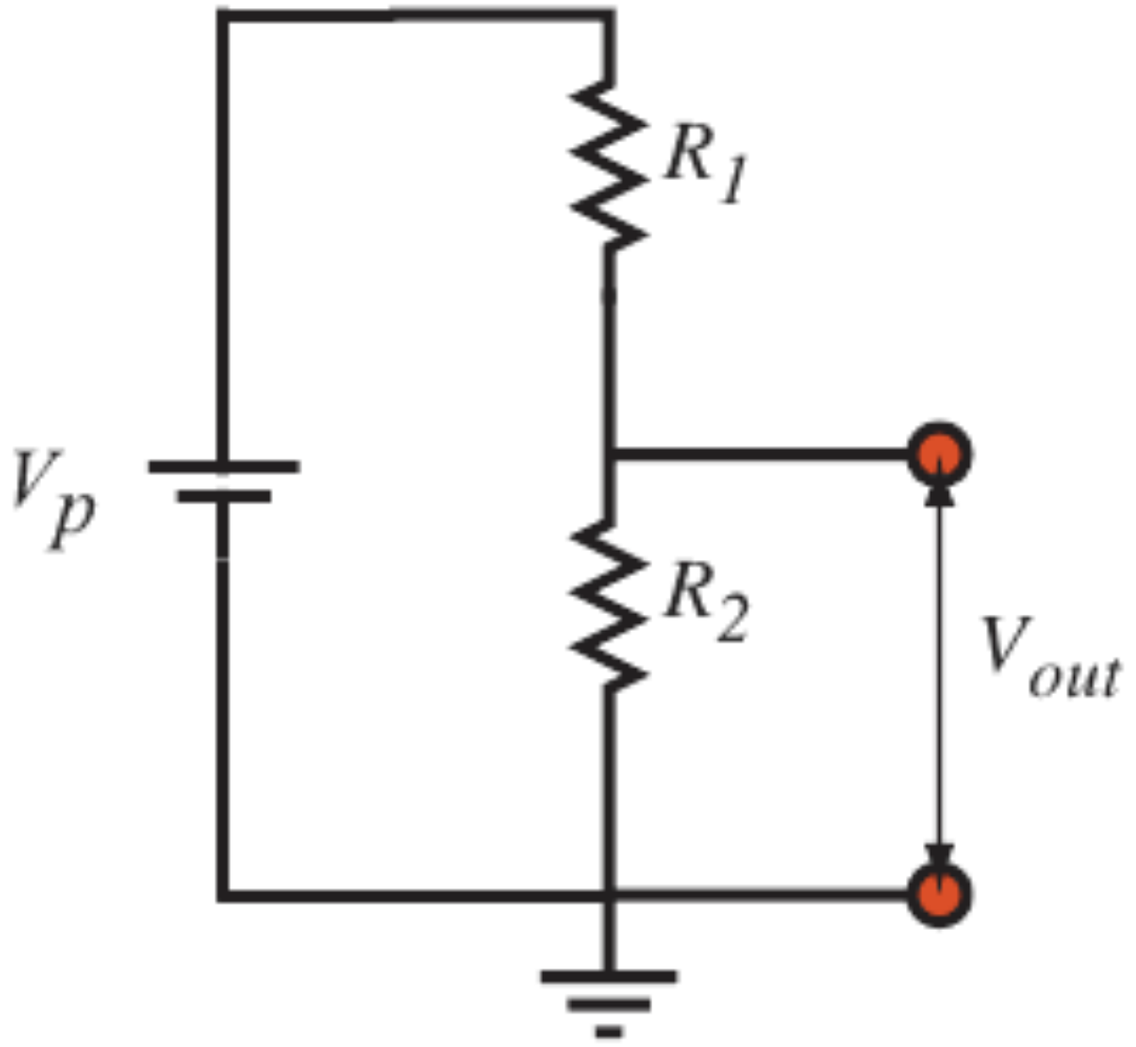
Light dependant resistor, LDR as **Analog input,** and LED as **Analog output.**

Does anyone know what the **resistors** are for?

Voltage divider - **Super important concept**

Often used to convert a sensor value that varies in resistance into a change in Voltage, such as in our case with the **Light Dependant Resister.**

*The circuit we are using is essentially the same just replacing one of the resistors with an **LDR.**



$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$

Analog_Read_Example | Arduino 1.6.10

Analog_Read_Example

```
//setting output pin for LED, ensuring PWM pin.
const int ledPin =  6;

// the setup routine runs once when you press reset:
void setup() {

  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);

  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);

}

// the loop routine runs over and over again forever:
void loop() {

  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);

  // print out the value you read:
  Serial.println(sensorValue);

  //map sensor value to a value between 0 and 255.
  int brightness = map(sensorValue, 0,1023,0,255);

  //outputing brightness with PWM, must be a value between 0 and 255.
  analogWrite(ledPin,brightness);
  delay(10);        // delay in between reads for stability

}
```

Read the sensor voltage

map() function is like a linear interpolation function.

PWM output

Done Saving.

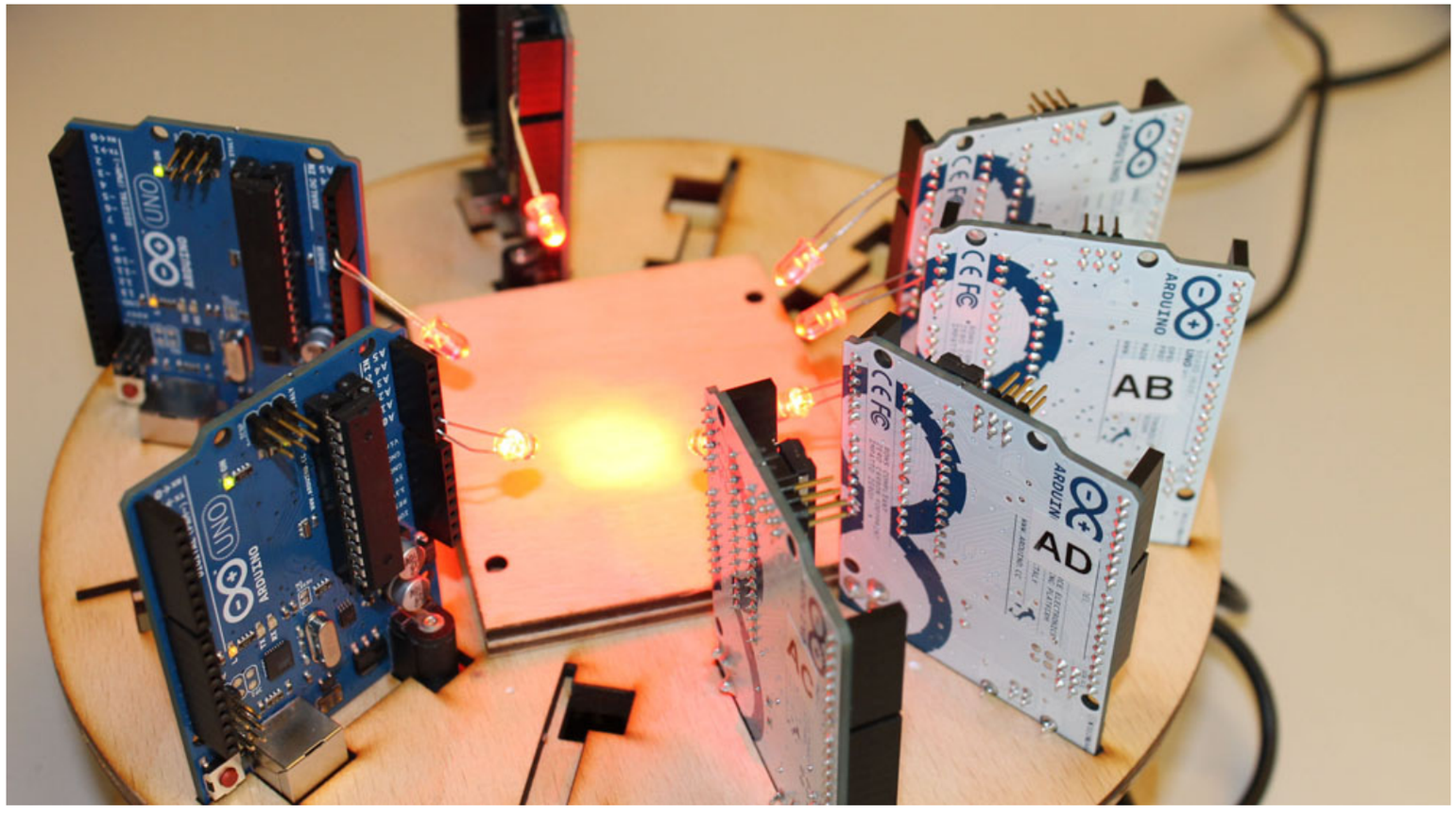
```
Invalid library found in /Users/seanmalikides/Documents/Arduino/libraries/jeroendoggen-Arduino-signal-filtering-library-ef8b188: /Users/seanmalik
Invalid library found in /Users/seanmalikides/Documents/Arduino/libraries/jeroendoggen-Arduino-signal-filtering-library-ef8b188: /Users/seanmalik
```

PanGenerator

Challenge

With the knowledge from the previous 2 examples attempt to get your Arduino **respond** to your partners Arduino with **Light!** Like the modules in the video.

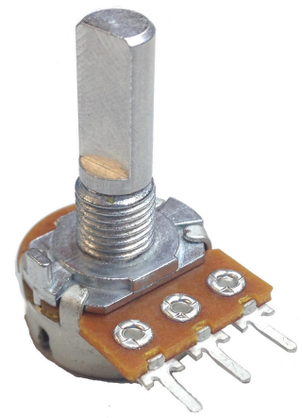
We would like to put all the modules in a swarm together at the end and get them all to speak to each other.



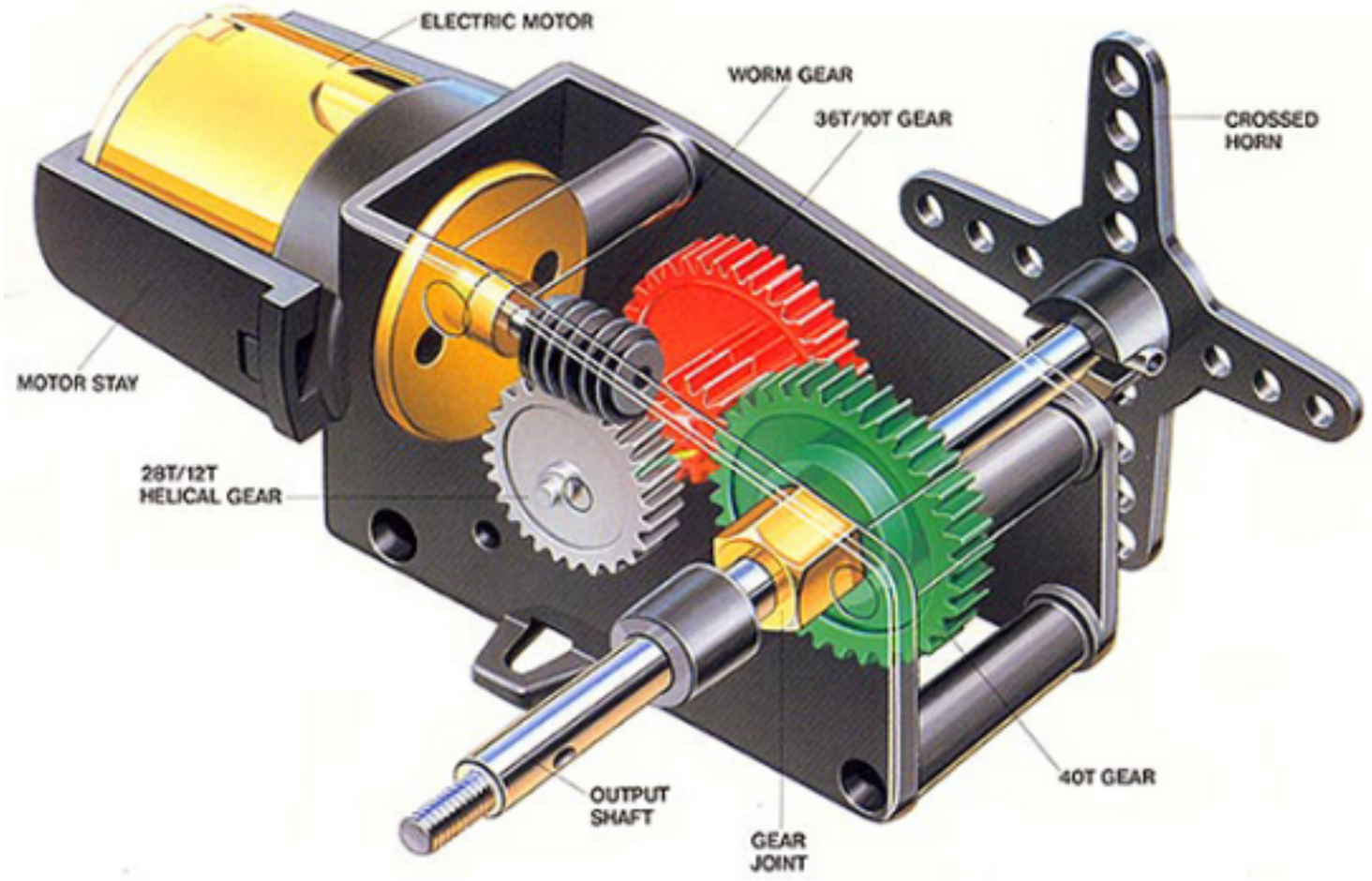
Add features

- Try adding multiple light sensors so you can sense a light pulse from multiple directions. You should be able to simply use the circuit from the previous examples.

- Try adding a potentiometer to enable you to adjust the light threshold for which your device responds to light.



Gearbox Motors - Giving your projects motion

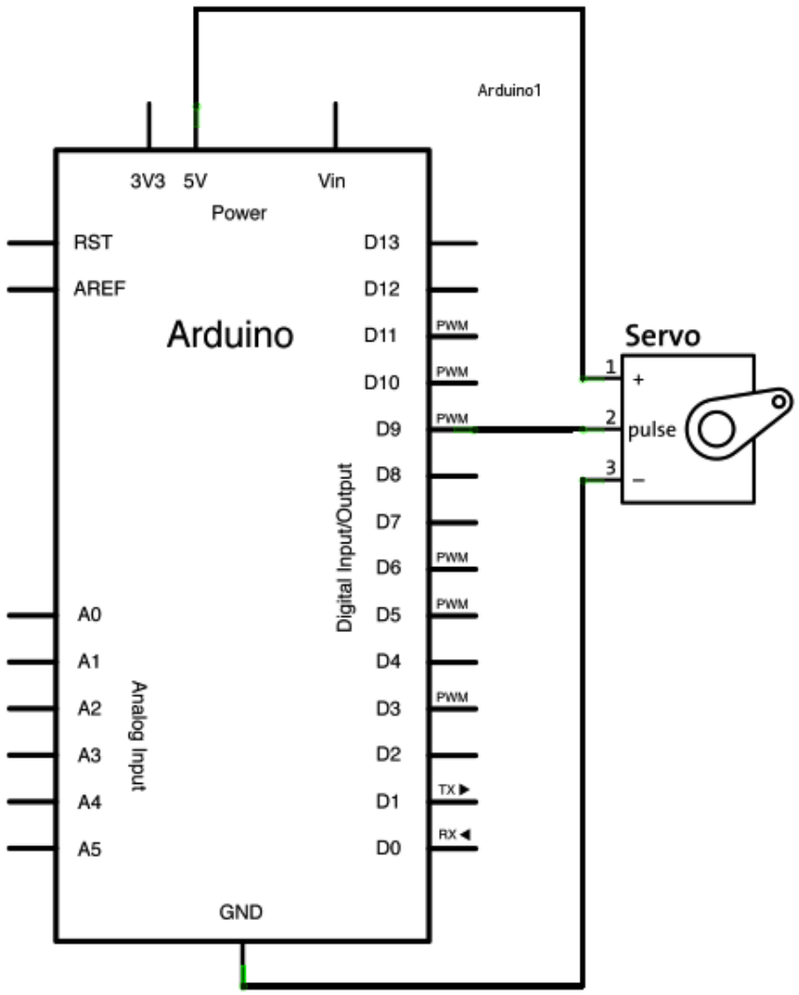


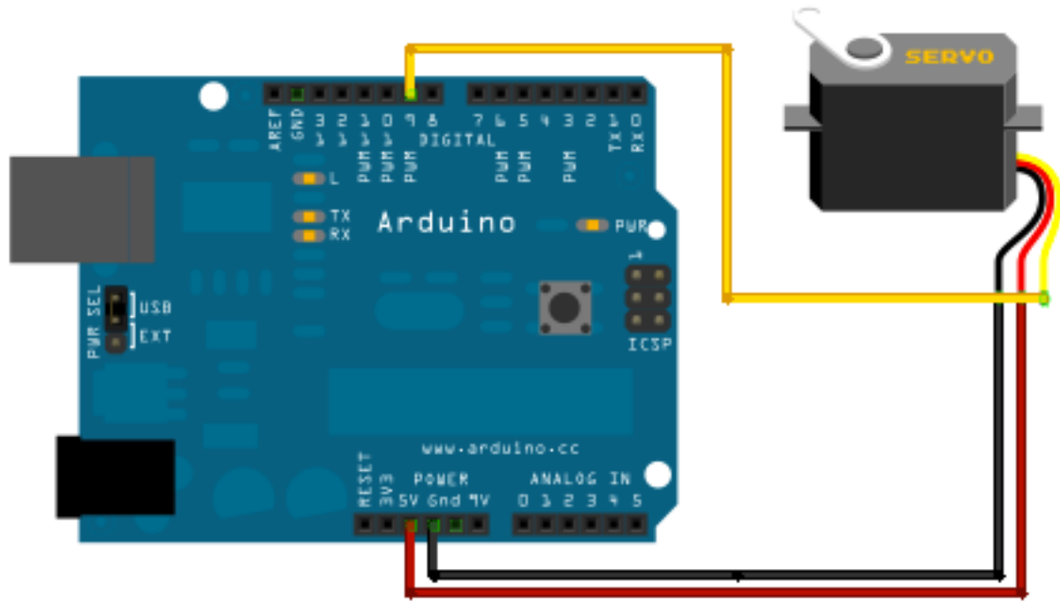
- Super easy to use.
- Can be super strong!

“Hobby” Servo Motors — We will work with these

- Super easy to use.
- Provide smooth accurate motion.
- Good for making small things move

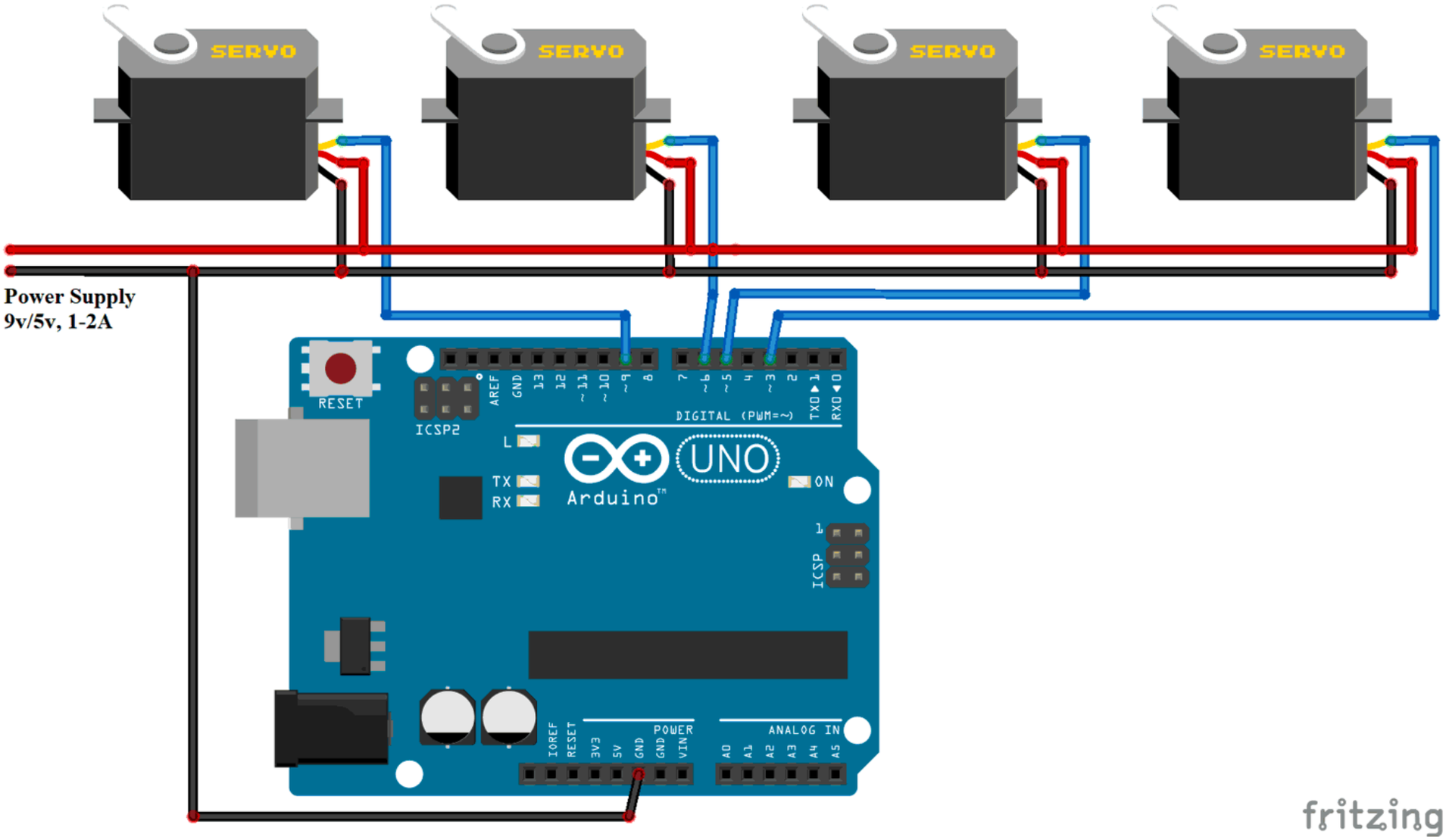
Control Circuit





With Arduino you only need to use a **single** pin for control, you do however need to power the servo from an extra power supply. **WHY IS THIS?**

Multiple Servo Control Circuit



With Arduino you only need to use a **single** pin for control, you do however need to power the servo from an extra power supply. **WHY IS THIS?**

Sweep | Arduino 1.8.5

Sweep

```
#include <Servo.h> // use a Servo Library which simplifies servo control

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

Function to initialise servo

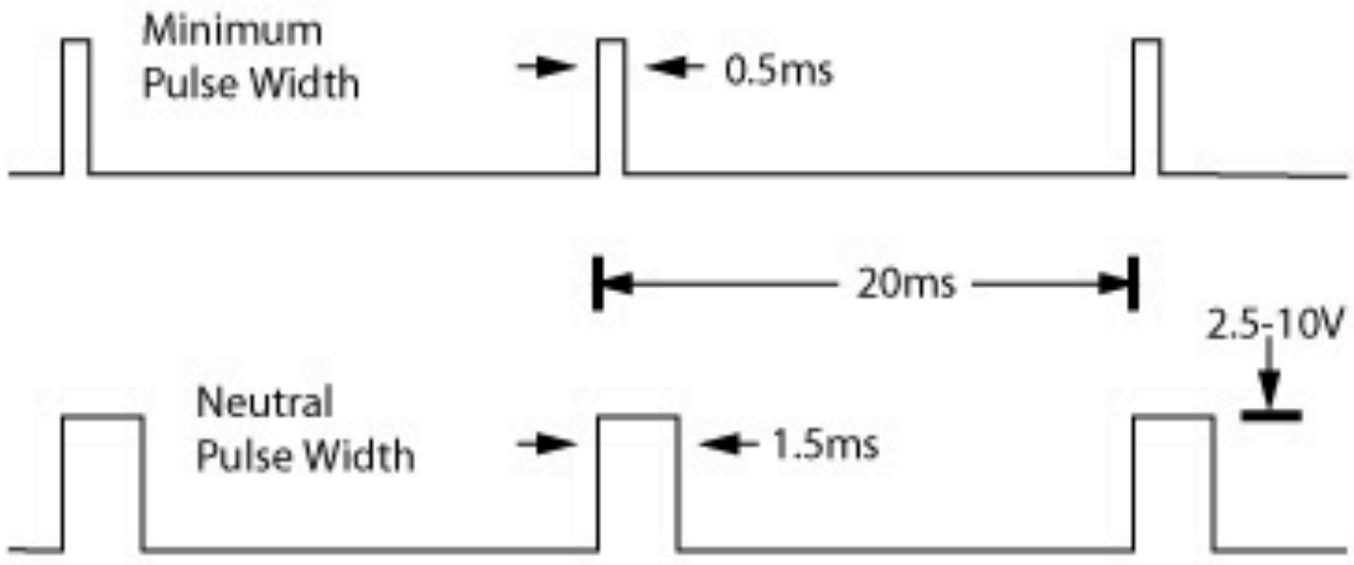
Function to set servo angle, 0 - 180 degrees

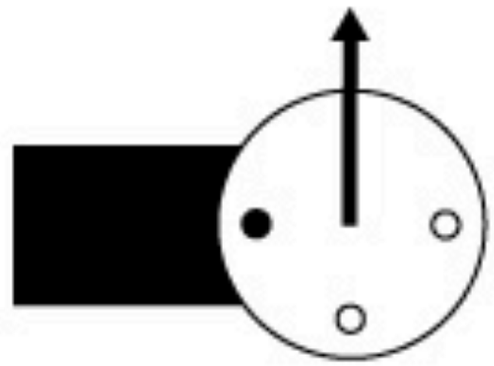
Done Saving.

2

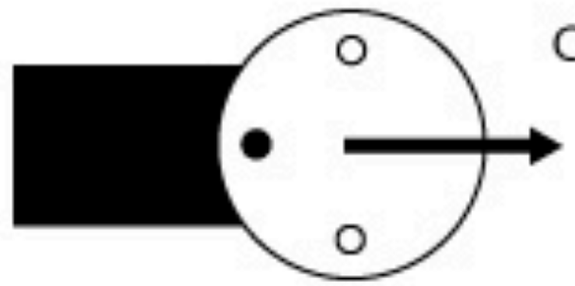
Arduino/Genuino Uno on /dev/cu.usbmodem51

R/C Control Signal Theory

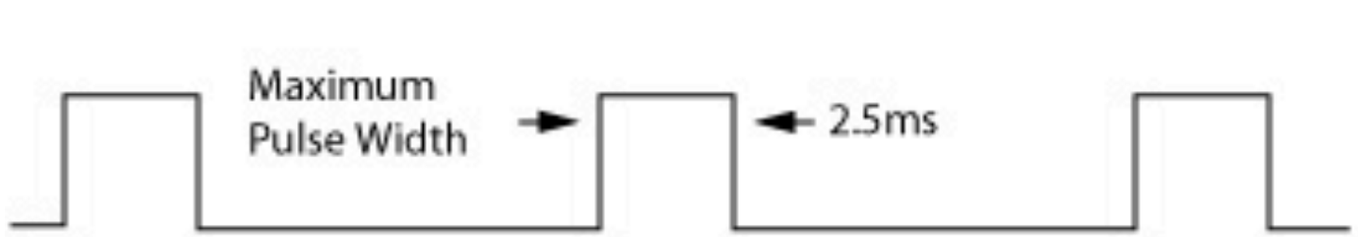


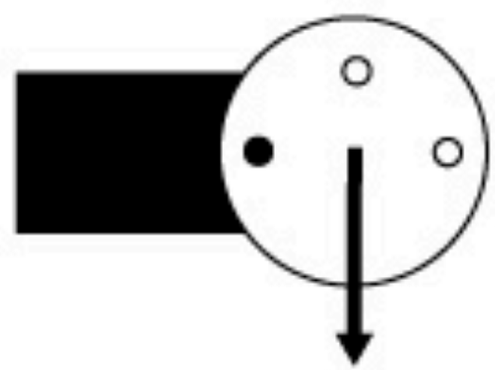


Maximum left throw. 45-90 degrees.



Center Position





Maximum right throw. 45-90 degrees.

The minimum and maximum pulse width for different manufacturers can vary considerably; however, the neutral position is generally quite near 1.5ms regardless of manufacturer. Typical variance for the minimum pulse width is from 0.5ms to 0.8ms, and the typical variance for the maximum pulse width is from 2.5ms to 3.0ms. The frequency of the signal is generally near 50Hz; however, it can range from 30Hz to 200Hz. The output voltage can vary from 2.5V to as much as 10V.

homemadeServoLibrary0 | Arduino 1.8.8

homemadeServoLibrary0

```
// Servo Control without library

int servo = 9;// set servo pin

void setup() {
  pinMode(servo, OUTPUT); // set pin as output
}

void loop() {
  servoPulse(servo, 1500); // continously pulse 1500 = 90 degrees
  //servoPulse(servo,500); // continously pulse 500 = 0 degrees
  //servoPulse(servo,2500); // continously pulse 2500 = 180 degrees
  delay(20);
}

void servoPulse(int servoPin, int pulse) {
  digitalWrite(servoPin, HIGH);
  delayMicroseconds(pulse);
  digitalWrite(servoPin, LOW);
}
```

Why can you can only pulse on of these at a time? Because the Servos need time to get to position

Our own servoPulse Function

Done Saving.

/Users/ruairiglynn/Desktop/Sweep/Sweep.ino

13 Arduino/Genuino Uno on /dev/cu.usbmodem51

homemadeServoLibrary0 | Arduino 1.8.8

homemadeServoLibrary0 §

```
// Servo Control without library

int servo = 9;// set servo pin
int pos = 500;

void setup() {
  pinMode(servo, OUTPUT); // set pin as output
}

void loop() {
  for (pos = 500; pos <= 2500; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    servoPulse(servo, pos);             // tell servo to go to position in variable 'pos'
    delay(15);                          // waits 15ms for the servo to reach the position
  }
  for (pos = 2500; pos >= 500; pos -= 1) { // goes from 180 degrees to 0 degrees - using pulse vals
    servoPulse(servo, pos);             // tell servo to go to position in variable 'pos'
    delay(15);                          // waits 15ms for the servo to reach the position
  }
}

void servoPulse(int servoPin, int pulse) {
  digitalWrite(servoPin, HIGH);
  delayMicroseconds(pulse);
  digitalWrite(servoPin, LOW);
}
```

Servo Sweep without Library

Done compiling.

```
Sketch uses 1100 bytes (3%) of program storage space. Maximum is 32256 bytes.
Global variables use 11 bytes (0%) of dynamic memory, leaving 2037 bytes for local variables. Maximum is
```

18 Arduino/Genuino Uno on /dev/cu.usbmodem51