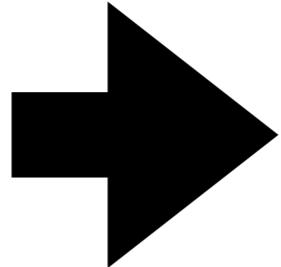


# **Intro to Arduino**

DfPI

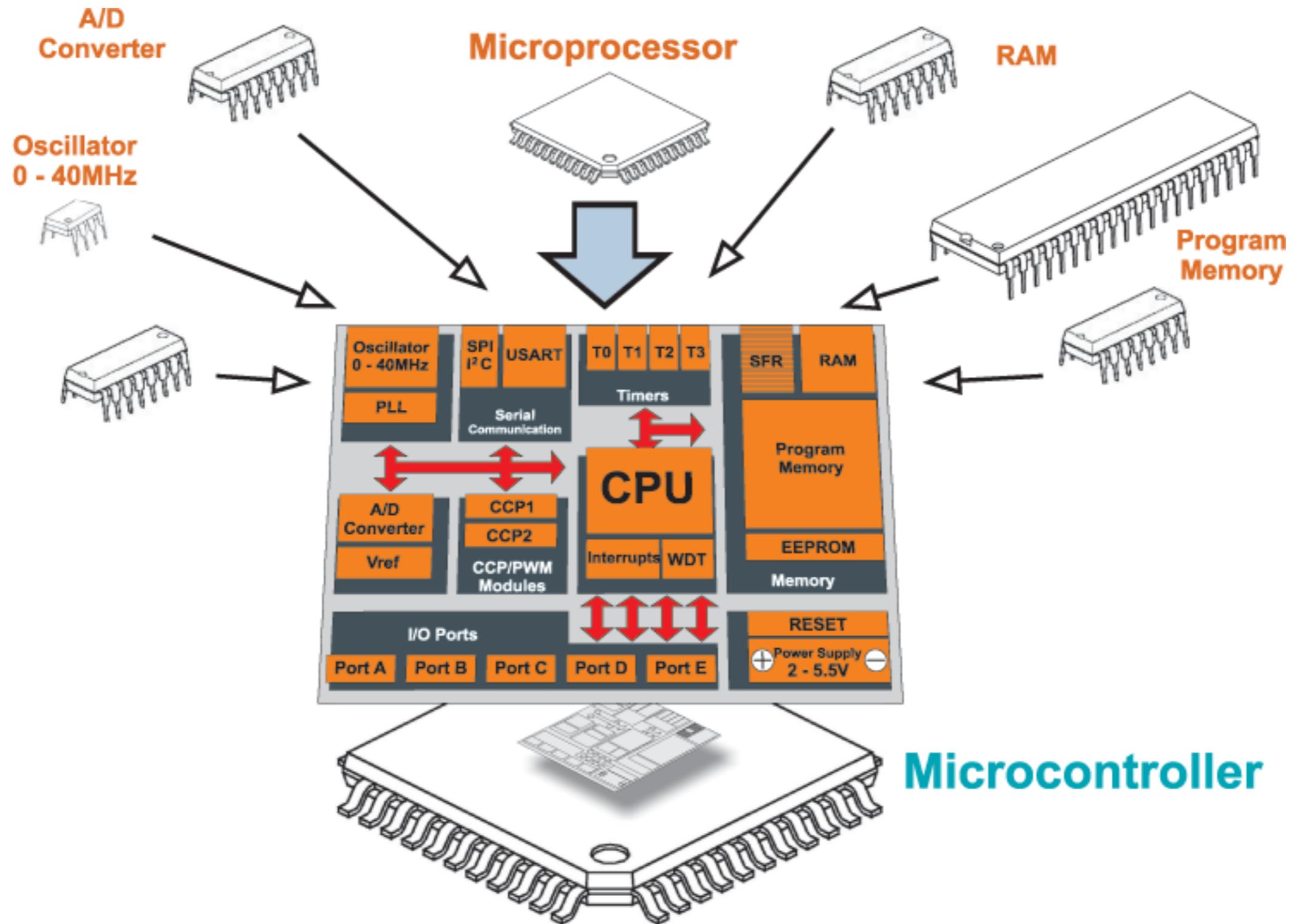


# Arduino

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

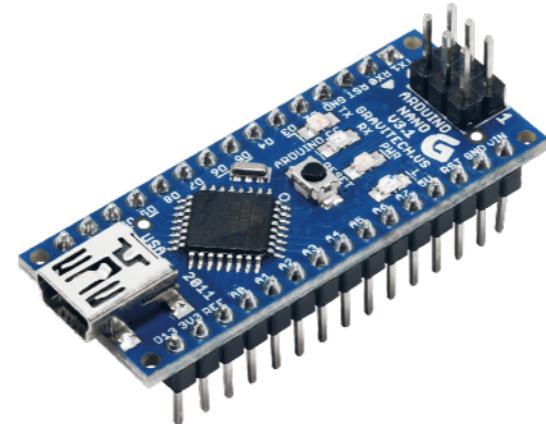


# What is a micro controller?

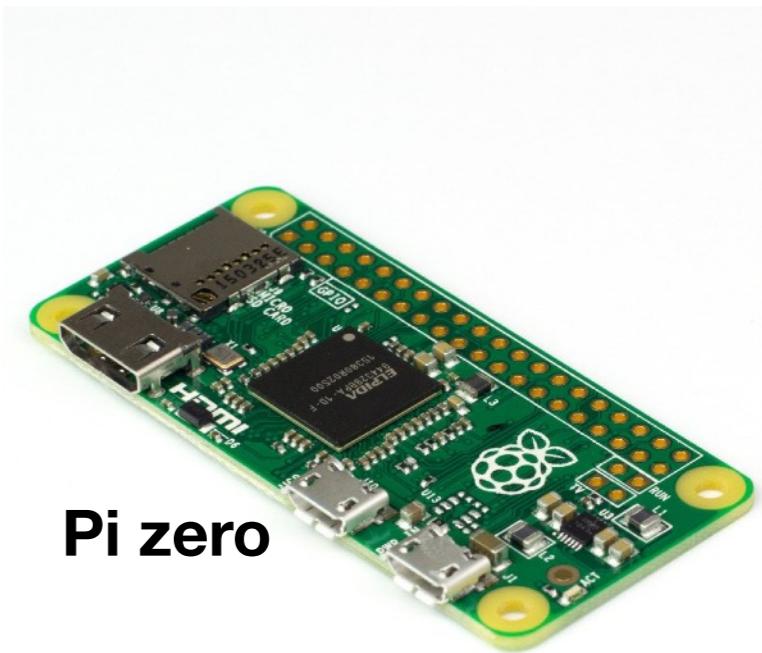




Uno

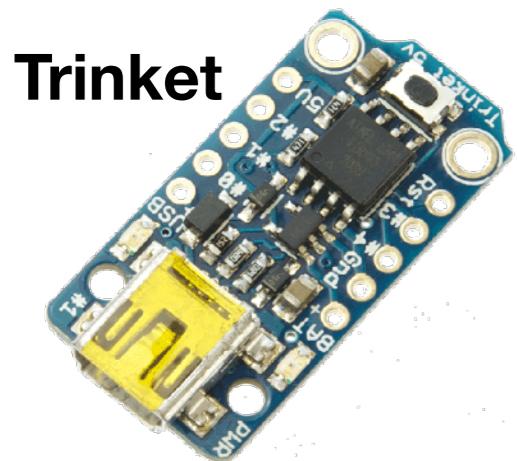


Nano

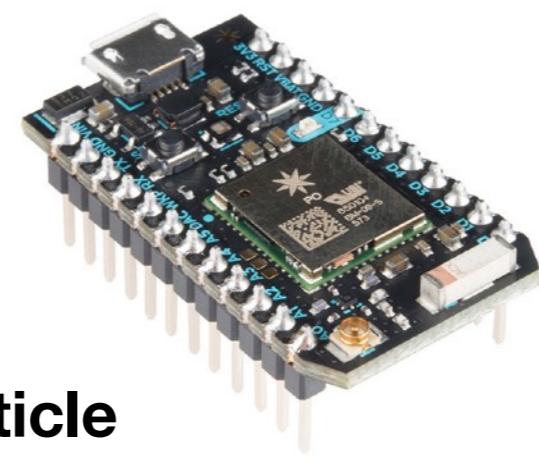


Pi zero

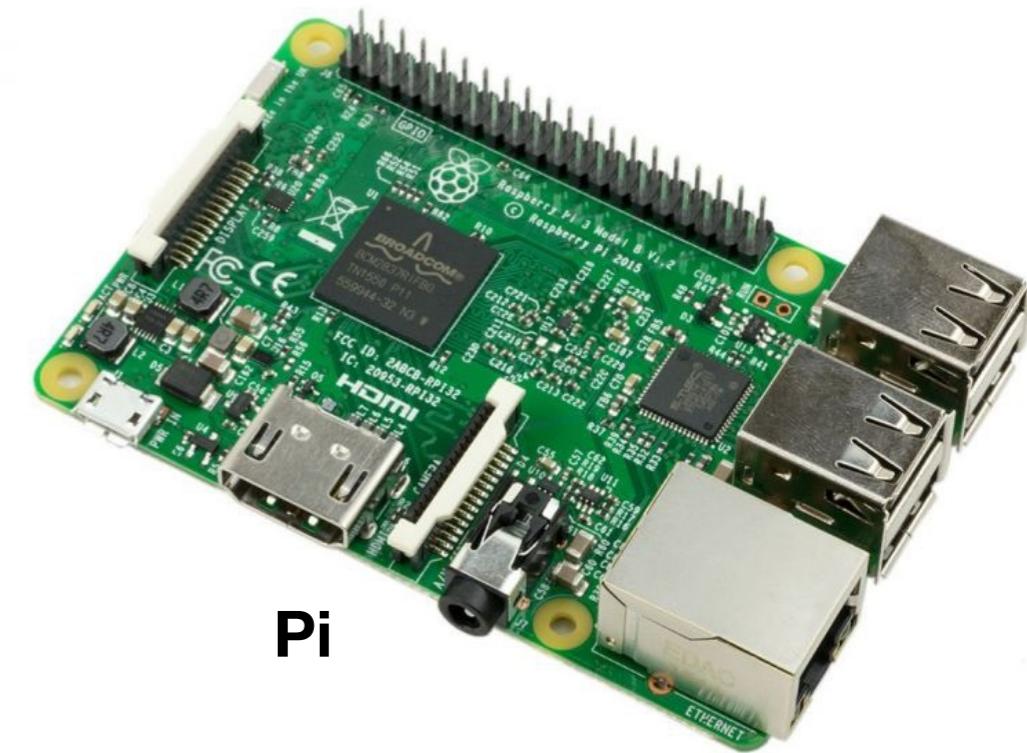
## Micro controller Dev. Boards



Trinket

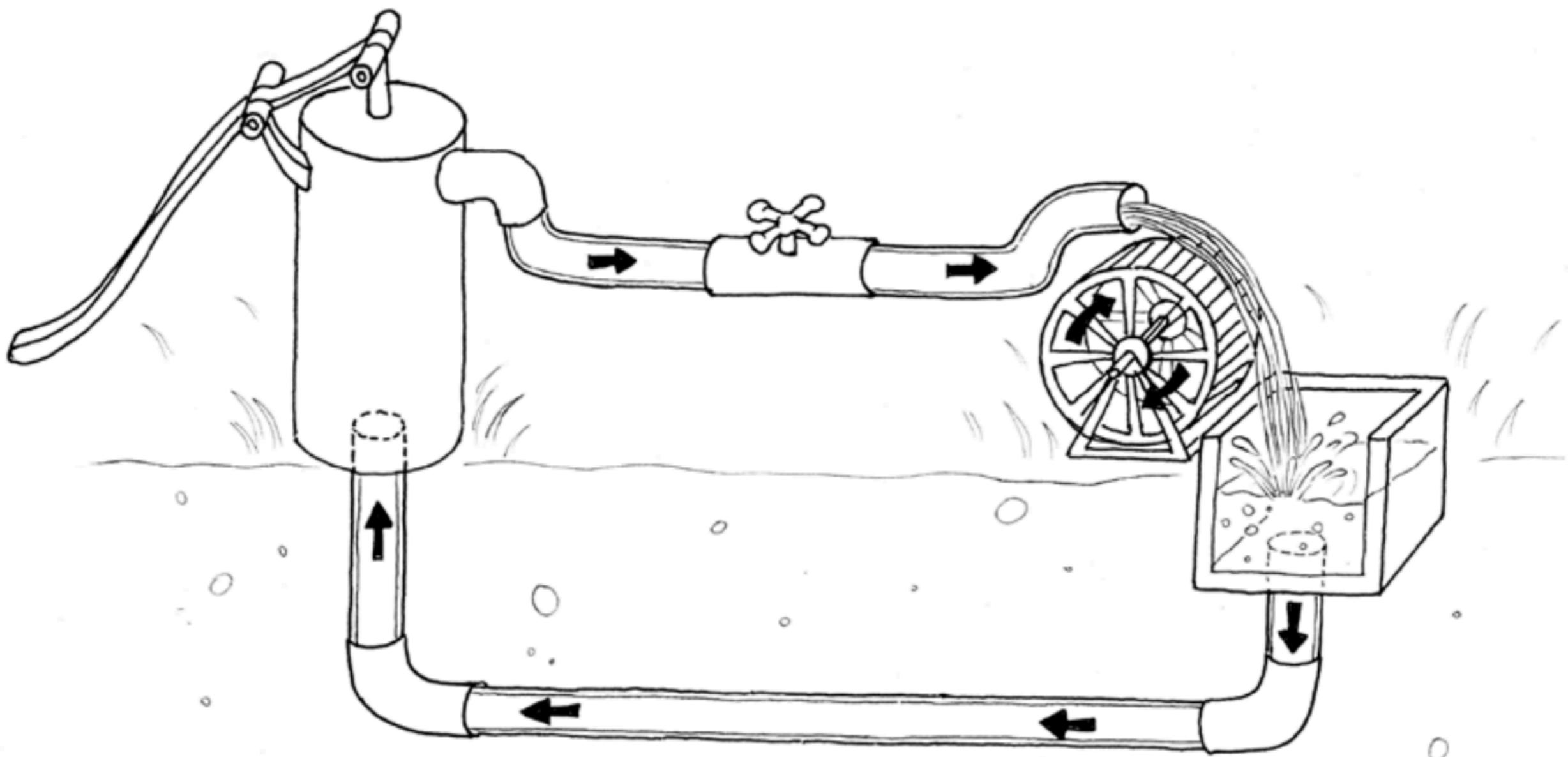


Particle



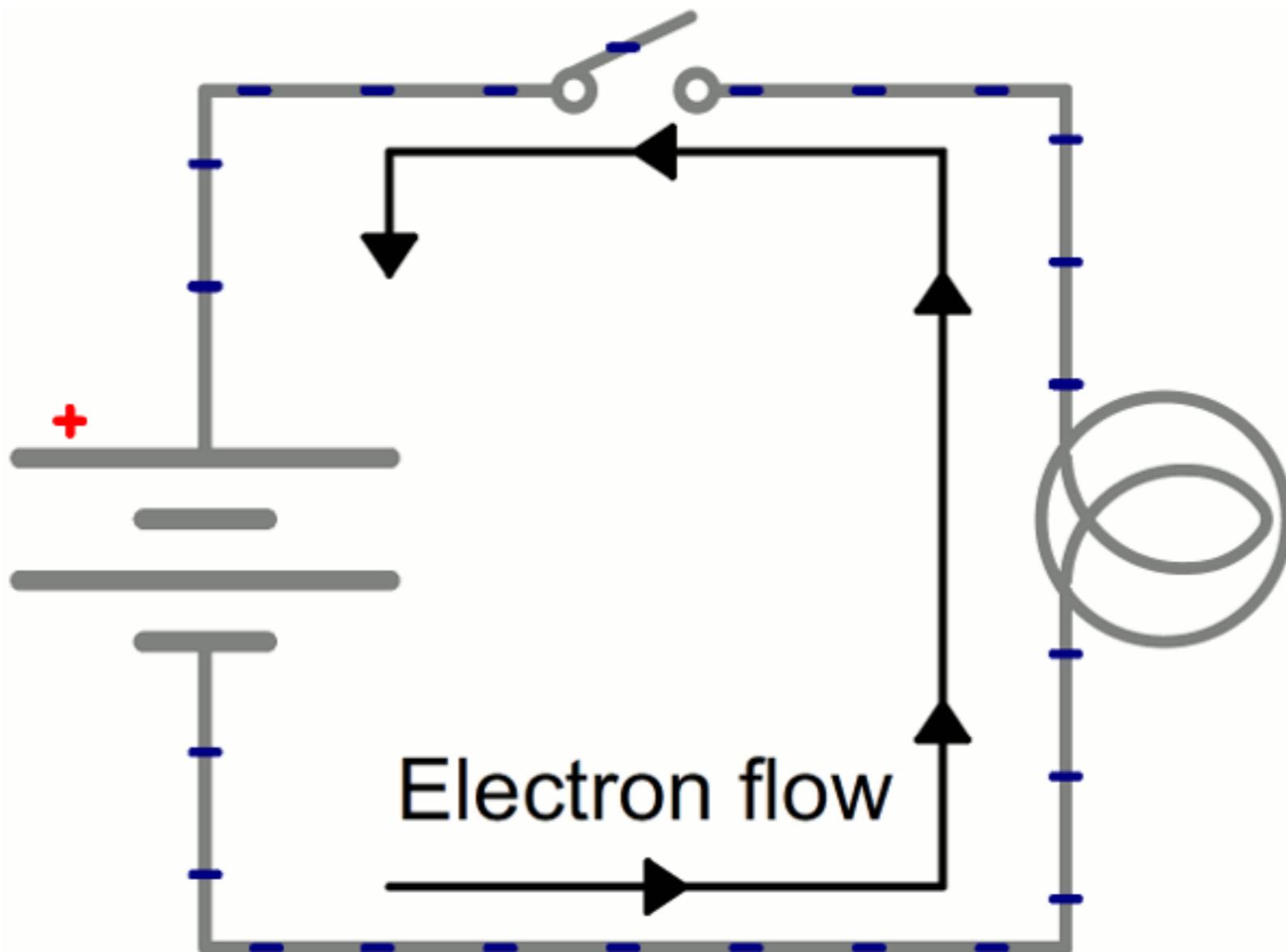
Pi

# Electronics revision



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

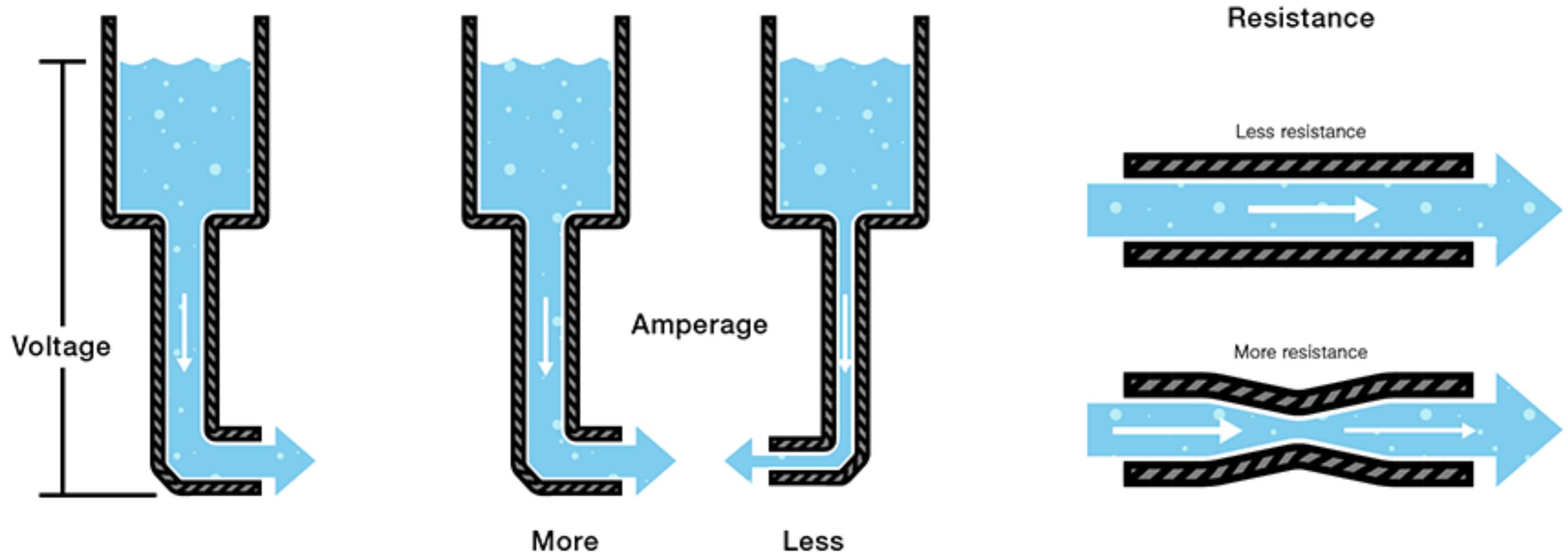
# Current flow visualisation



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

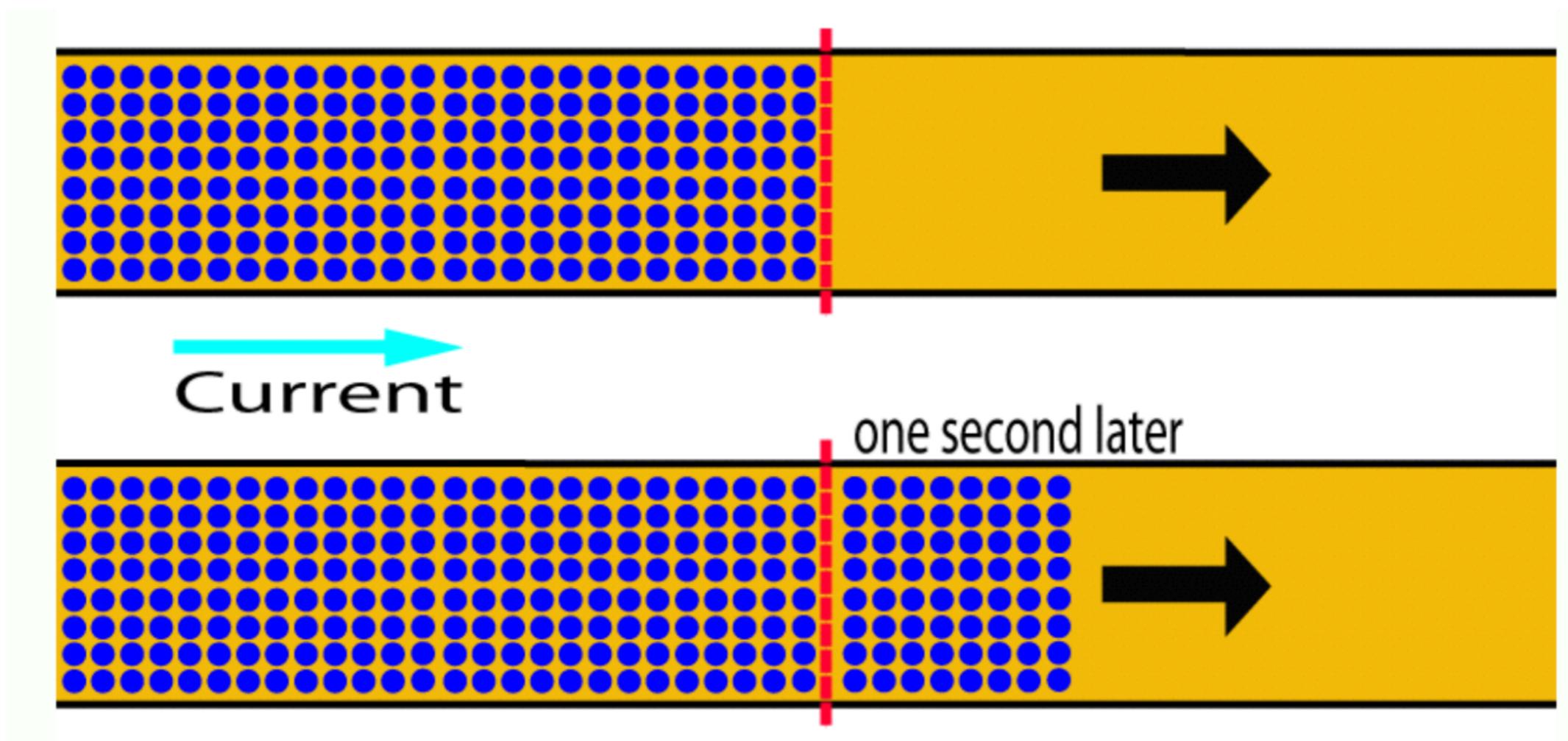
\*Does anyone know why the electrons are moving from negative to positive?

# 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.

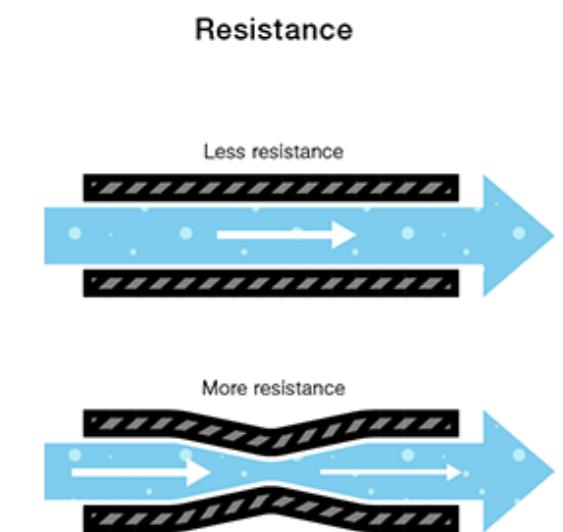
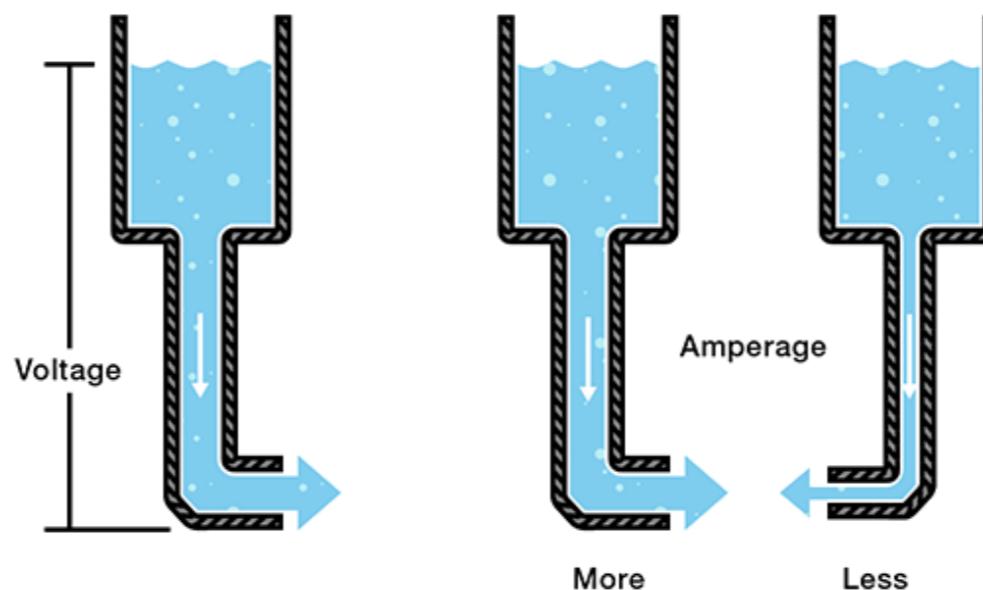
# Super important Law!

Ohm's Law

$$V = I * R$$

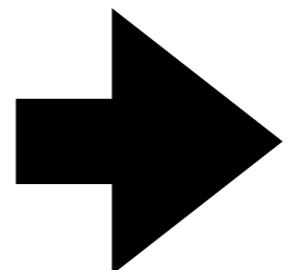
voltage in volts                          current in amps                          resistance in ohms

- $V$  = Voltage
- $I$  = Current
- $R$  = Resistance

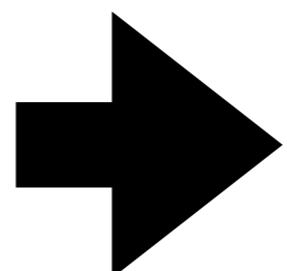
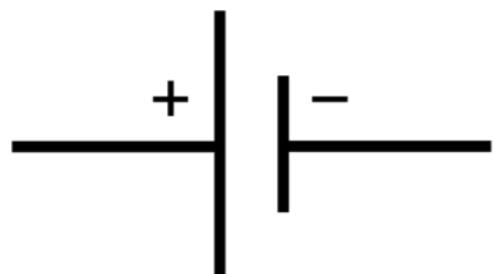


Usually you will only be varying the **Voltage** or **Resistance** in a circuit, and this will govern how the **Current** behaves.

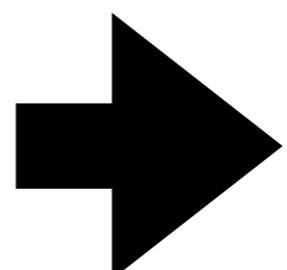
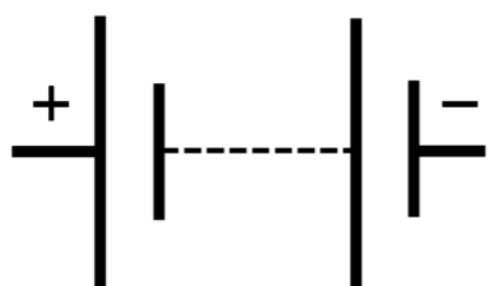
# Schematics & Symbols



**Resistor**



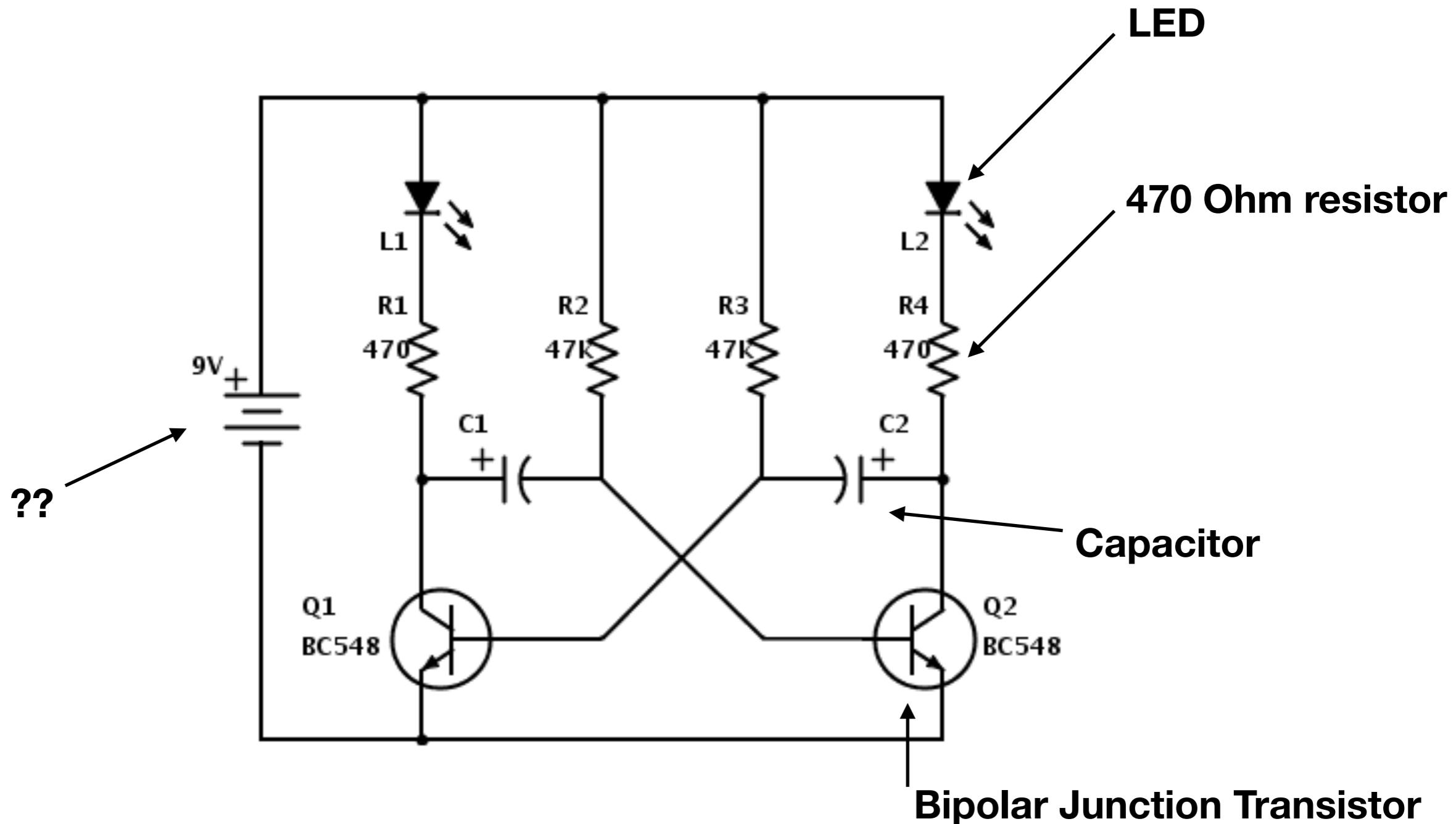
**Battery**



**Ground**

These are just a few of the essential symbols

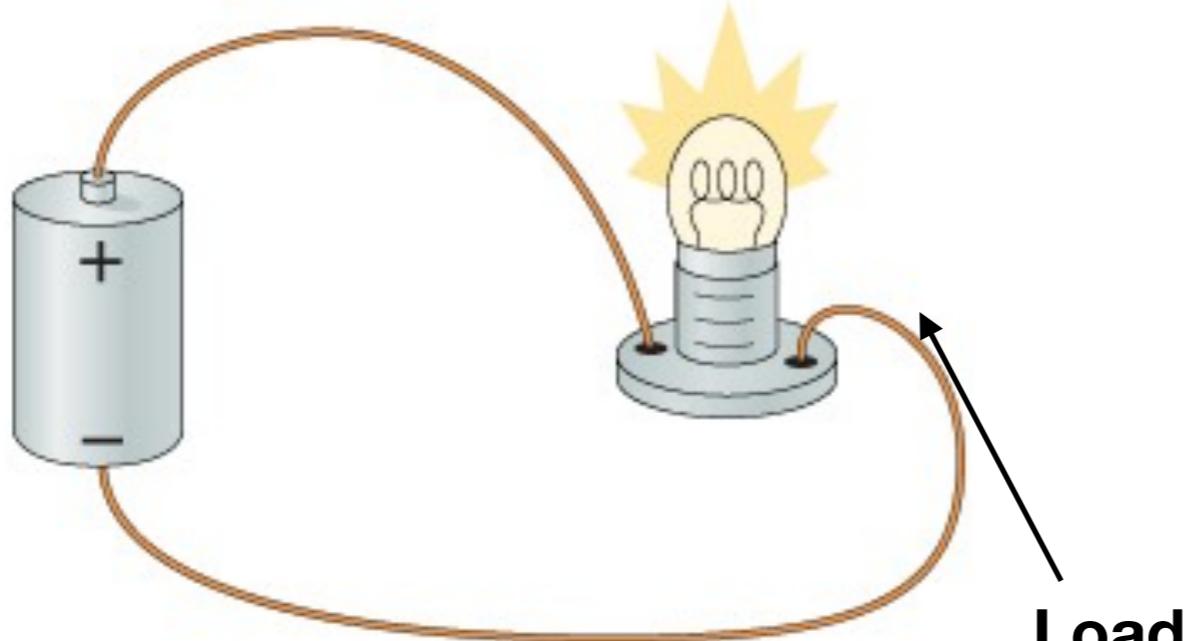
# Electrical Schematics



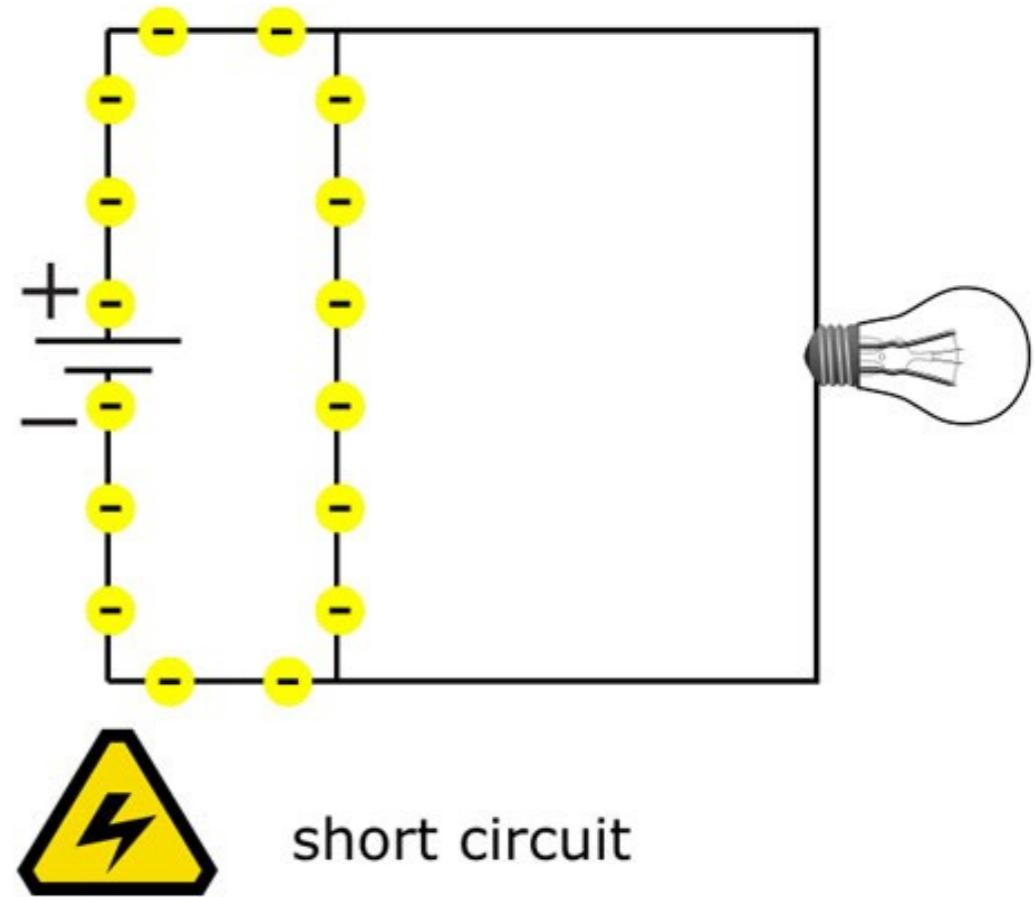
With a bit of practice you start to recognise all the components

# Short Circuits

Good



Bad

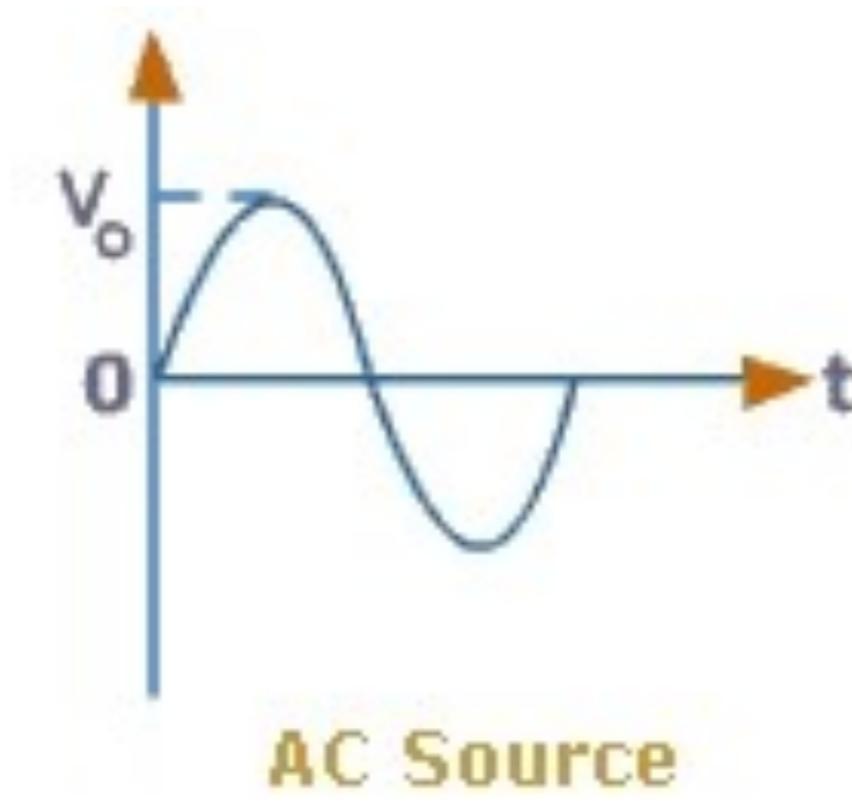
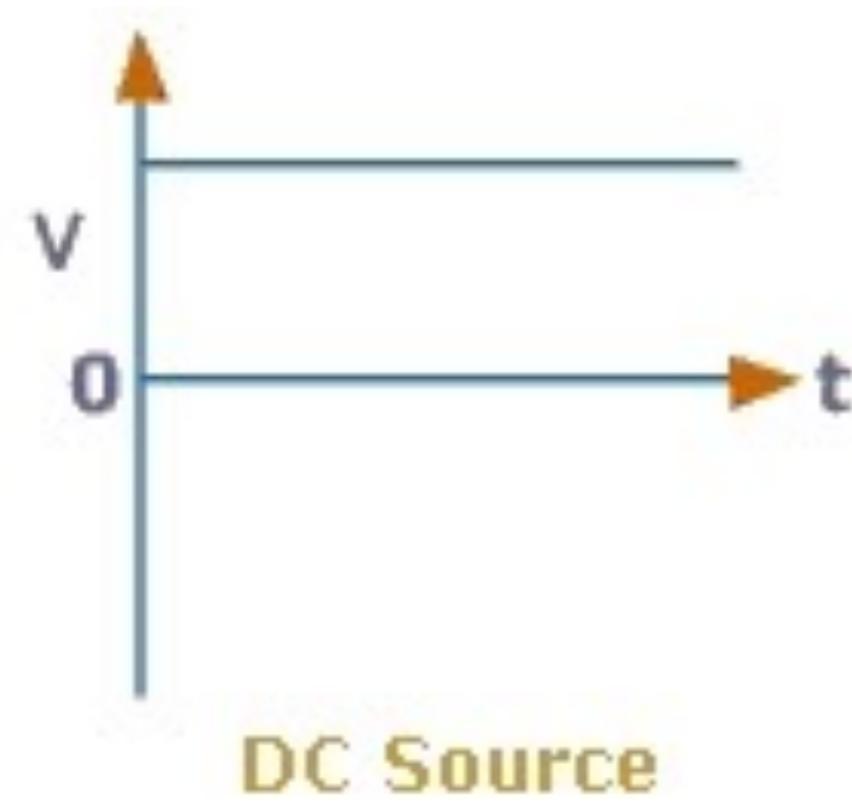


Regular circuit has wires connecting components in a loop, however if there is ever a case of a loop with **NO LOAD** then this is called a short circuit. (Load can be Motor, Light, ... etc.)

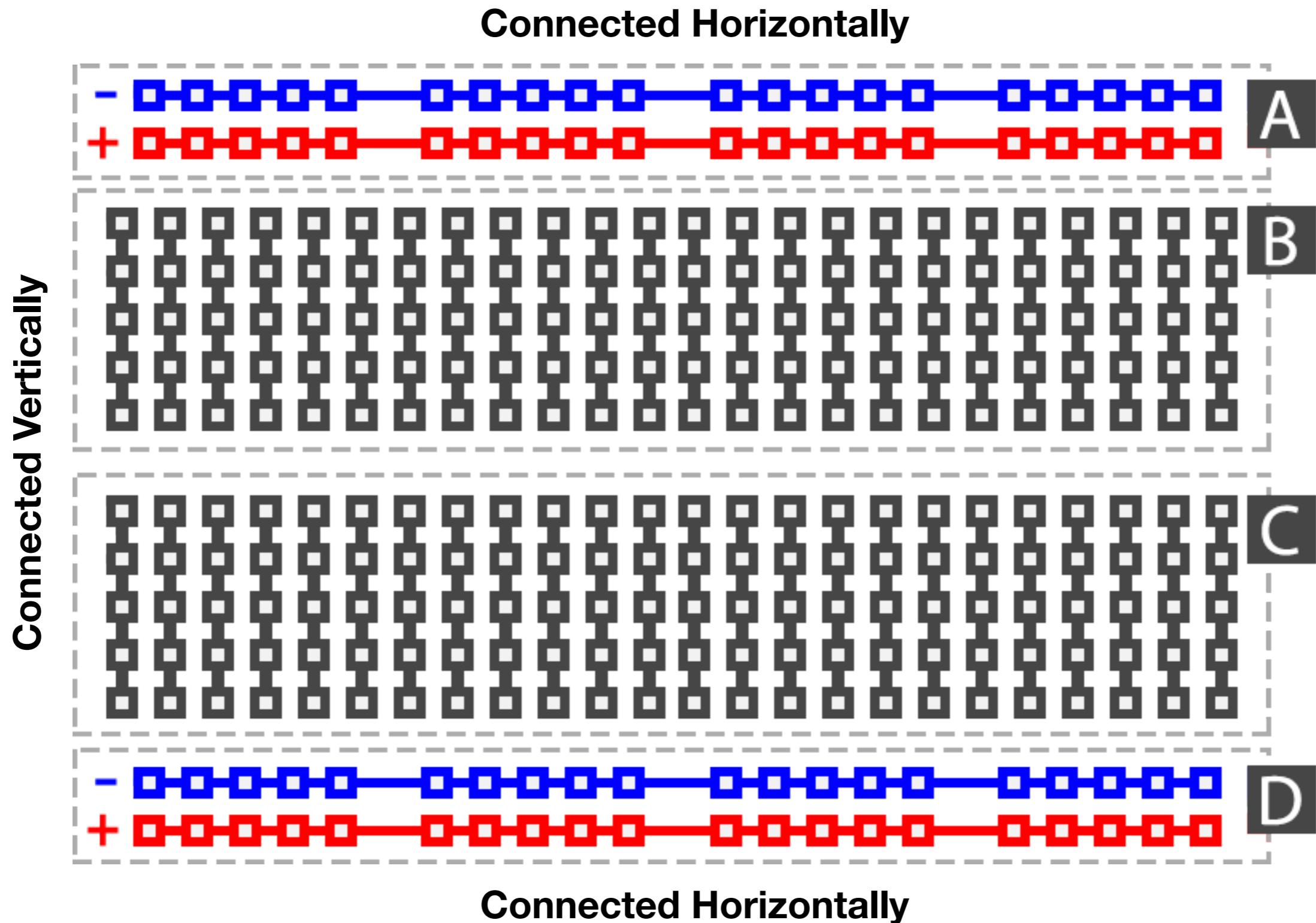
**Watch out for this with practical work!**

# 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.



# 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**.

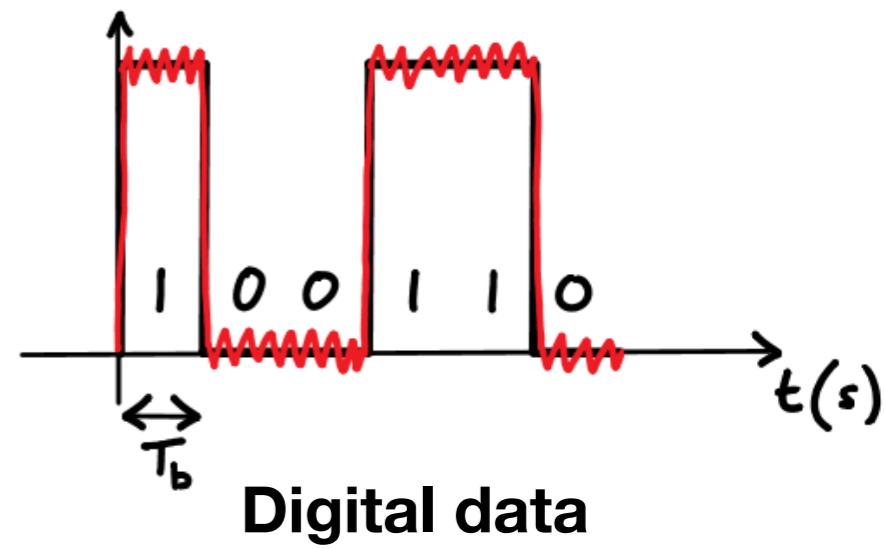
Examples of **digital** inputs.



**Micro Switch**

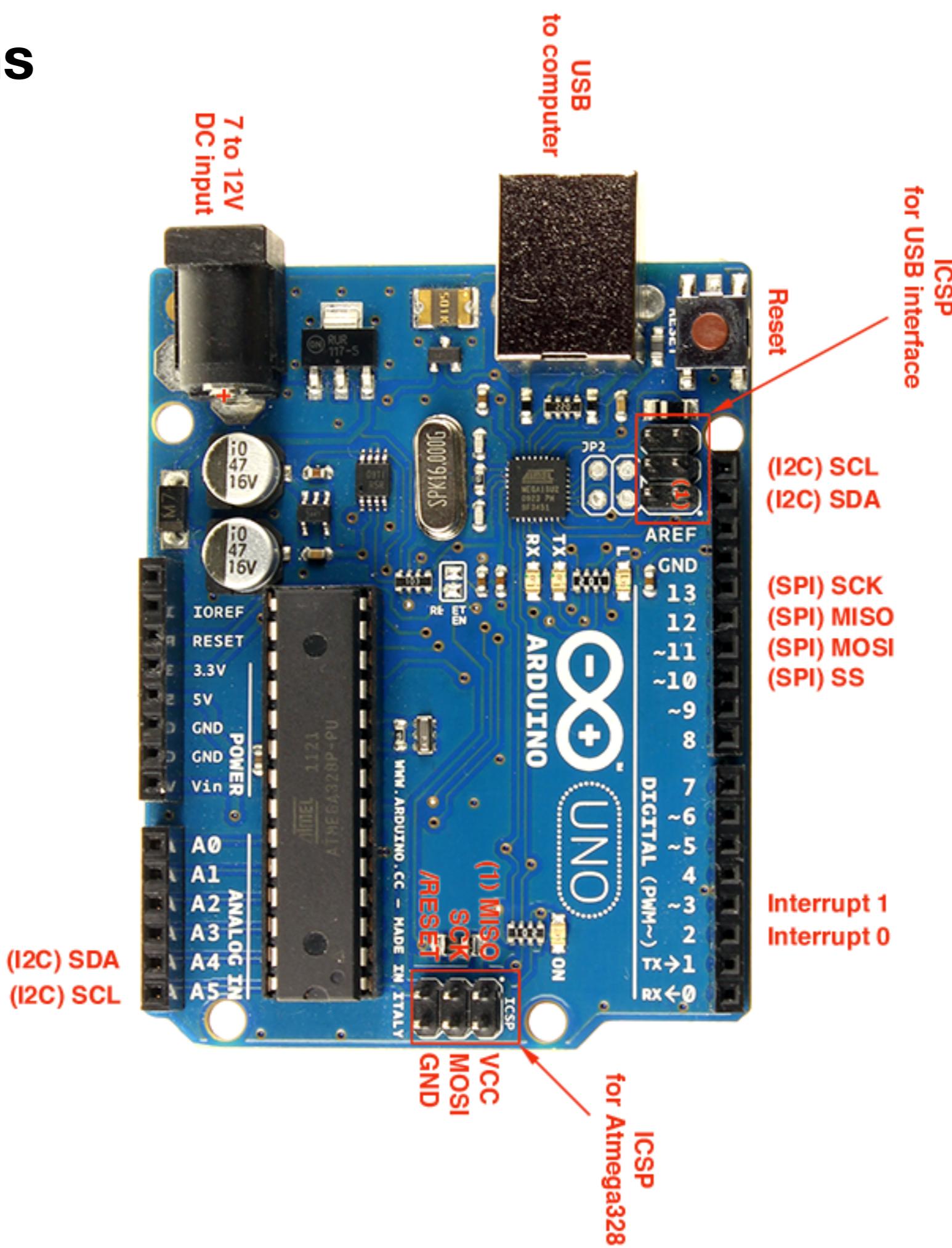


**Push button**

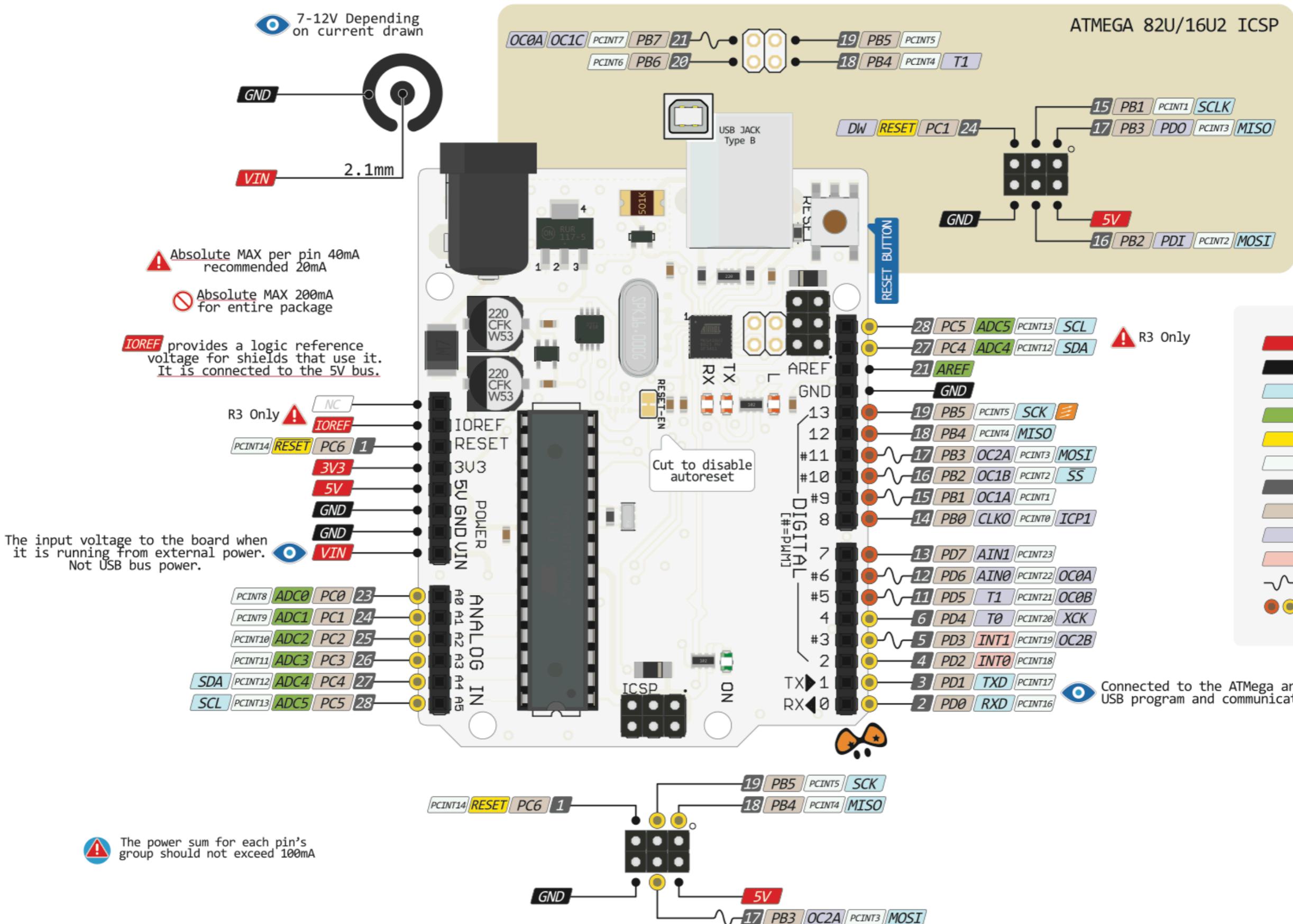


**Digital data**

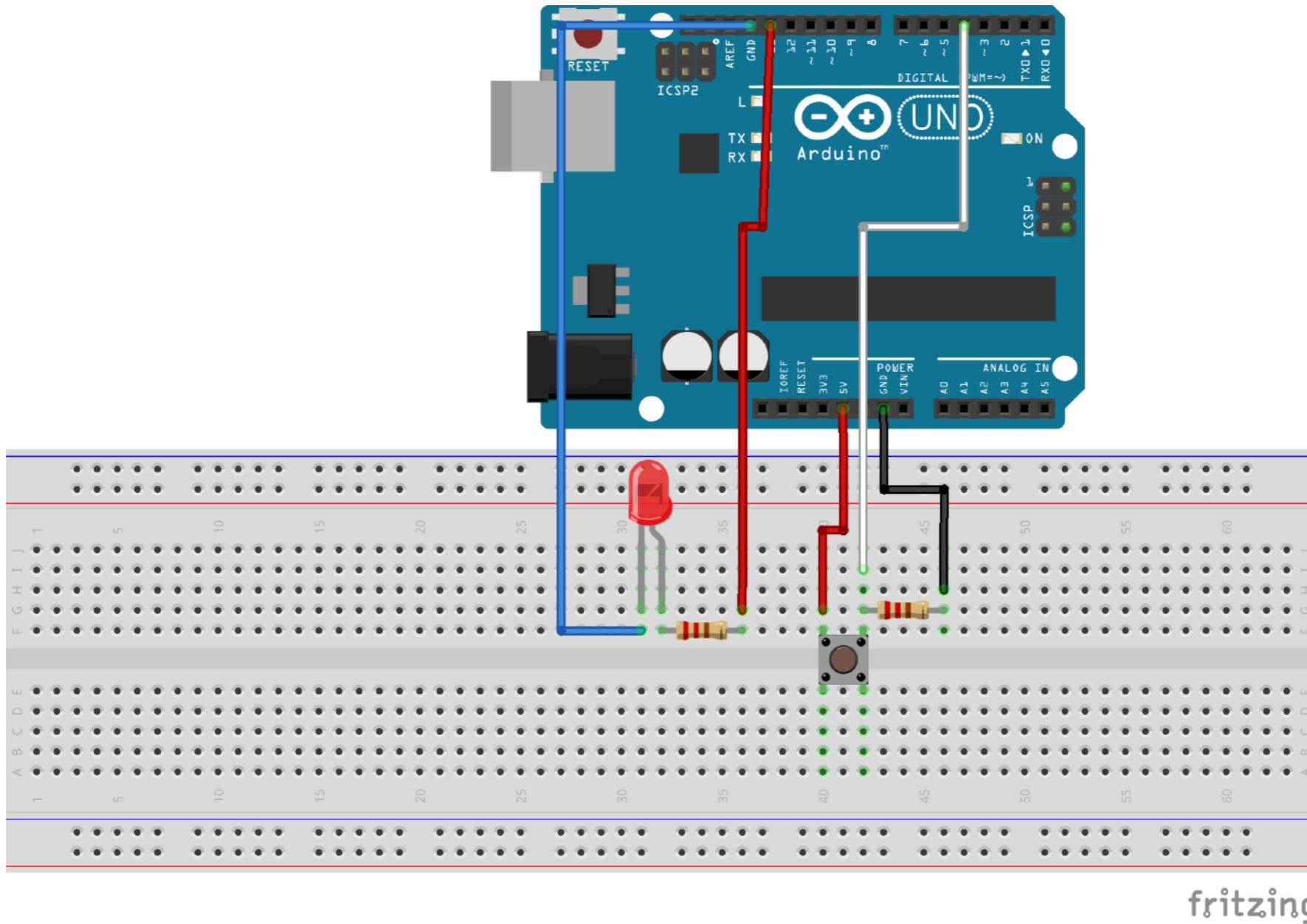
# Arduino Pins



# UNO PINOUT



# The Circuit

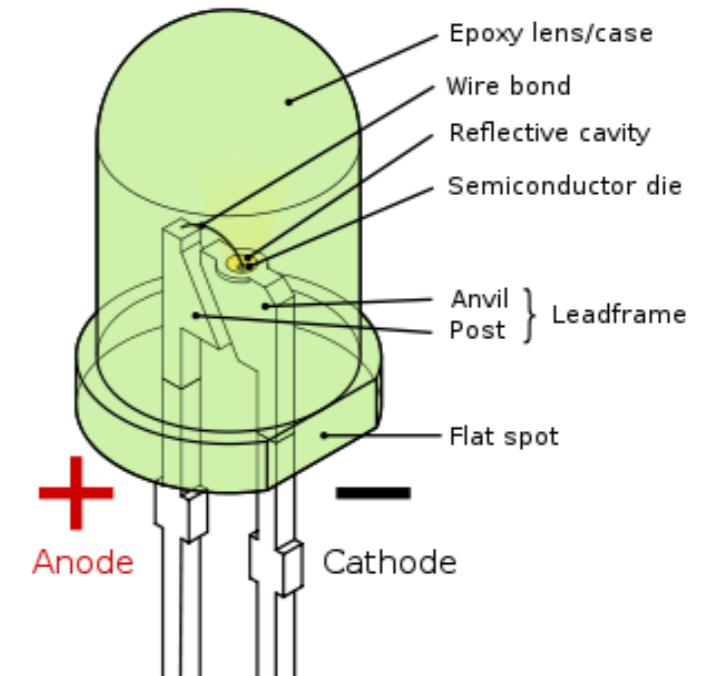


Push button as **Digital input**, and LED as **Digital output**.

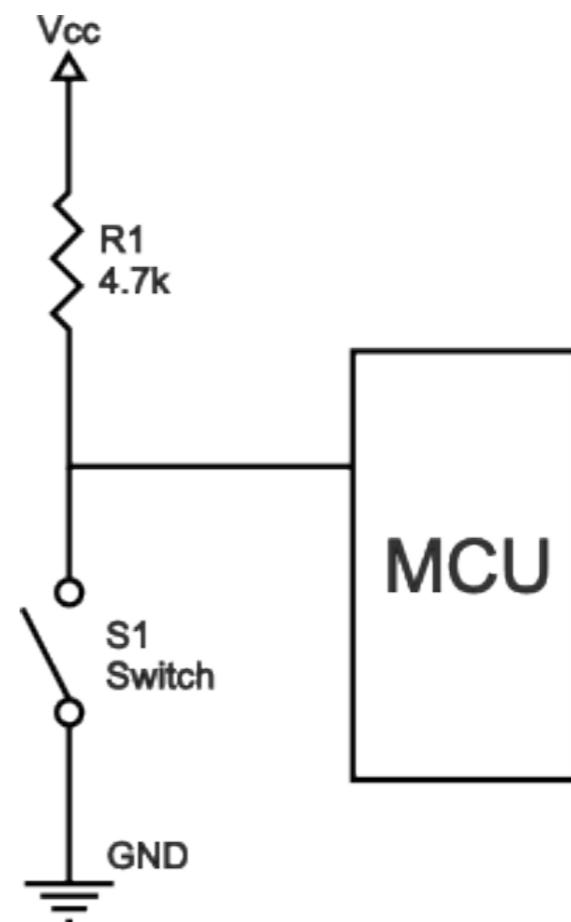
Does anyone know what the **resistors** are for?

\*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.





## 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; 1.
```

```
// constants won't change. Used here to set a pin number.  
const int ledPin = 13; // the number of the LED pin 2.
```

```
// the setup routine runs once when you press reset.
```

```
void setup() { 3.
```

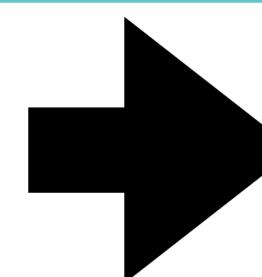
```
    // initialize serial communication at 9600 bits per second.  
    Serial.begin(9600);
```

```
    // 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)



## 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.

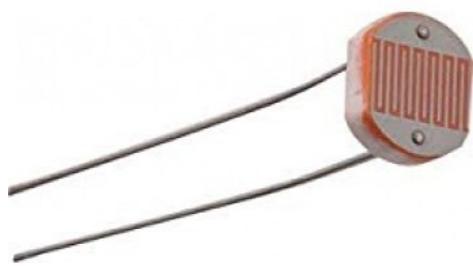
Wait for a little bit, Delay( $t$ ) where “ $t$ ” is in milliseconds.

## Exercise 2.

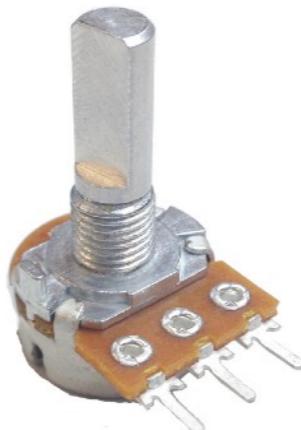
**Analog Input** - Any **Analog Pin** on the **Arduino** is capable of registering a Voltage variation within the range of **0V - 5V** with a resolution of 12 bits (ie. 0 - 1023 integer range).

**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**



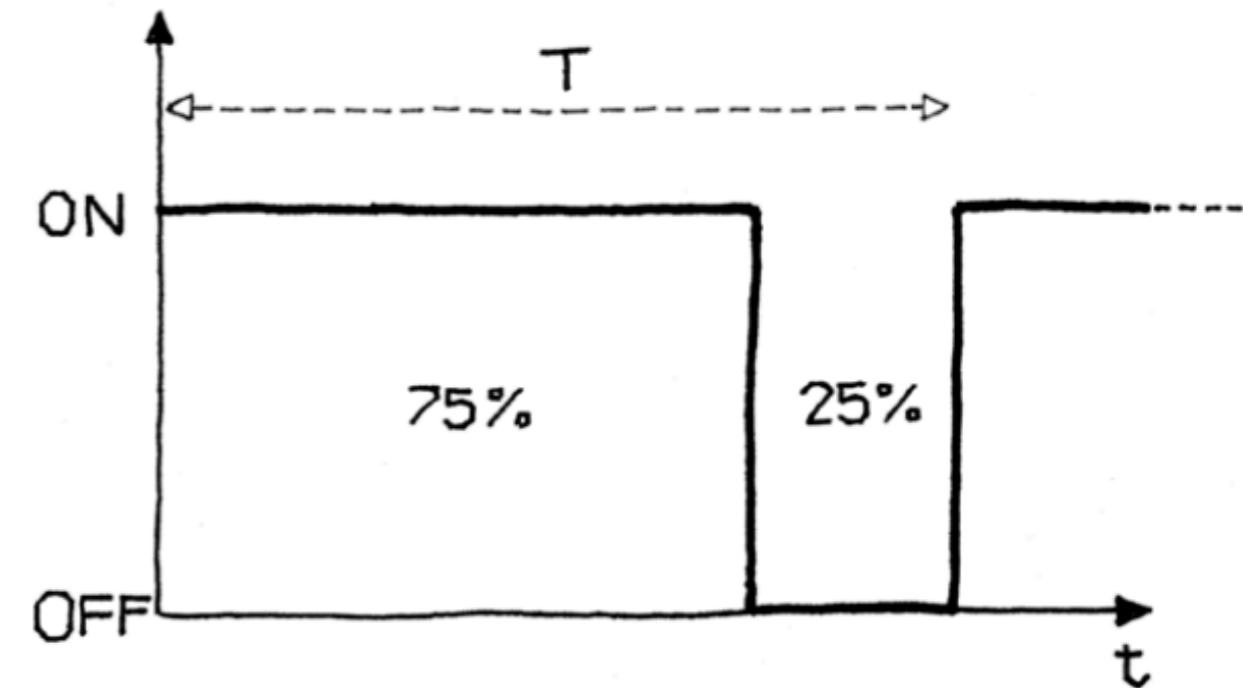
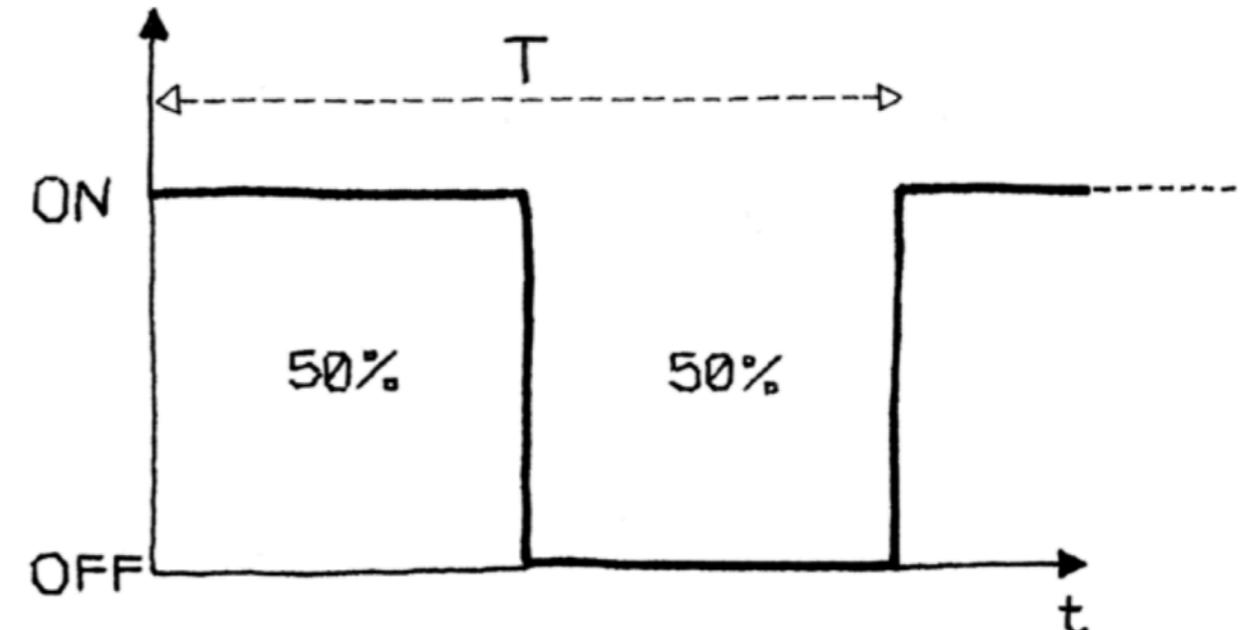
**Microphone**

# PWM - Pulse Width Modulation

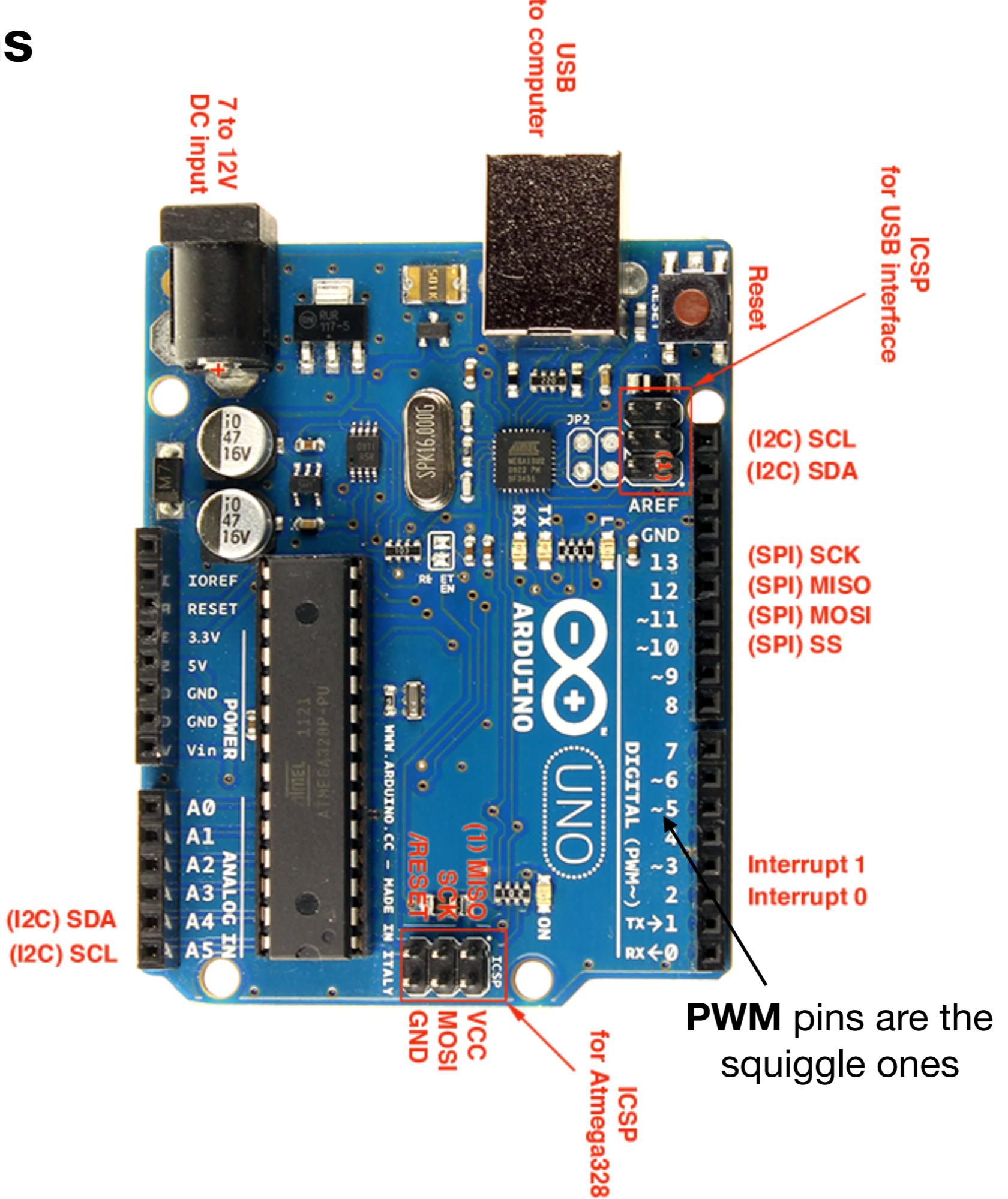
Effectively the **average voltage** over time is dependant on the Ontime:Offtime ratio of the signal.

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”**.

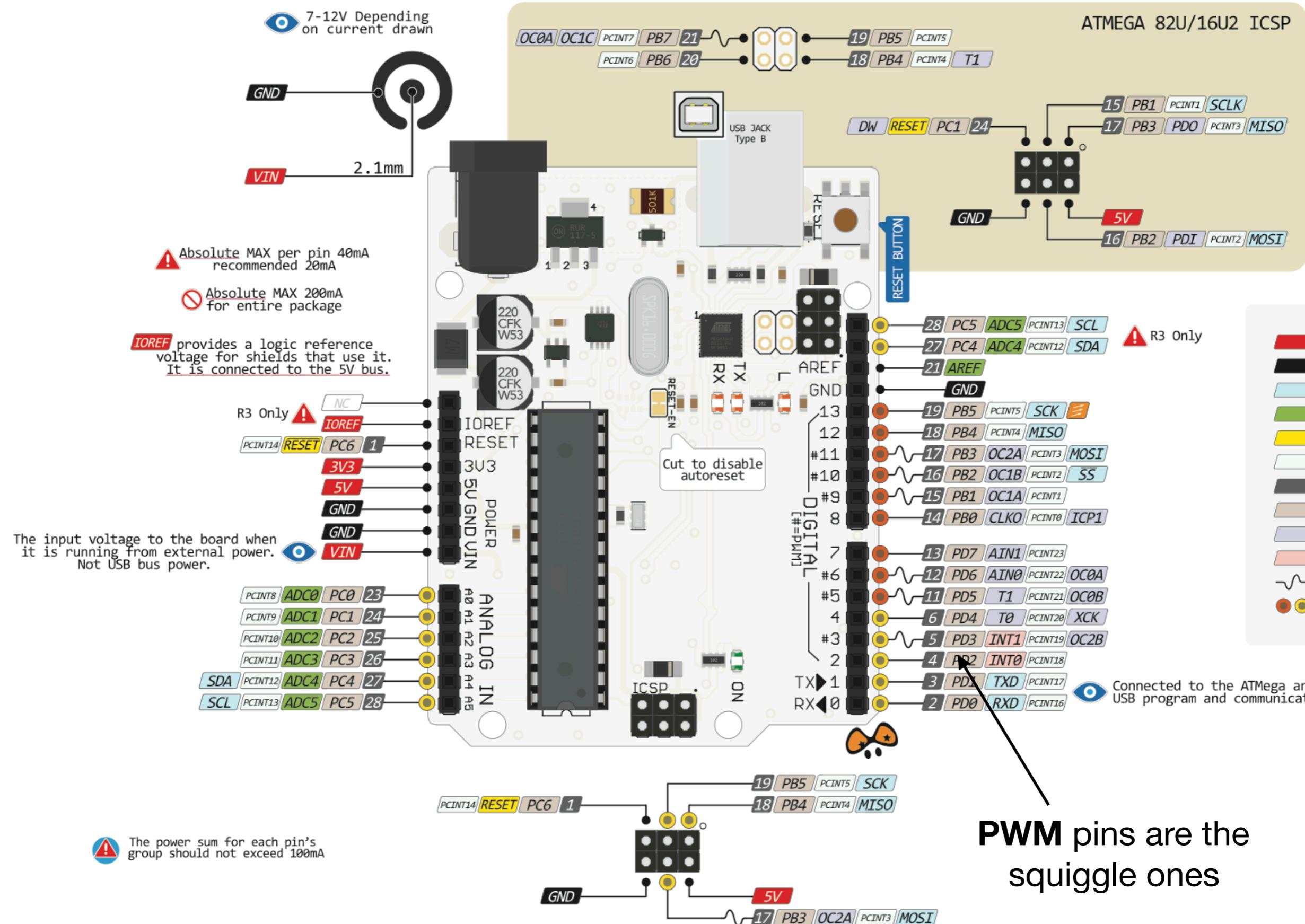
Only specific pins support PWM on the Arduino Uno.



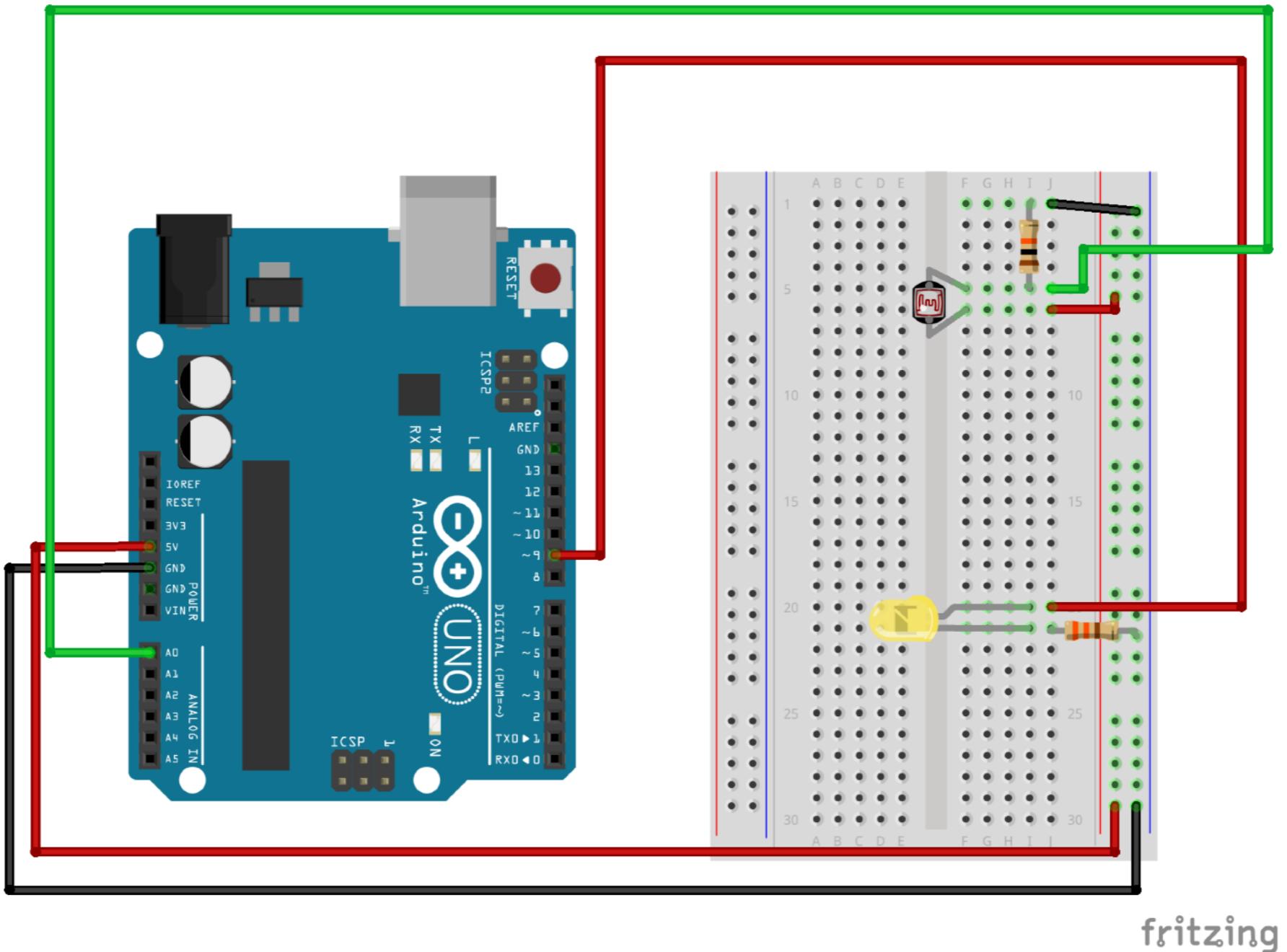
# Arduino Pins



# UNO PINOUT



# The Circuit



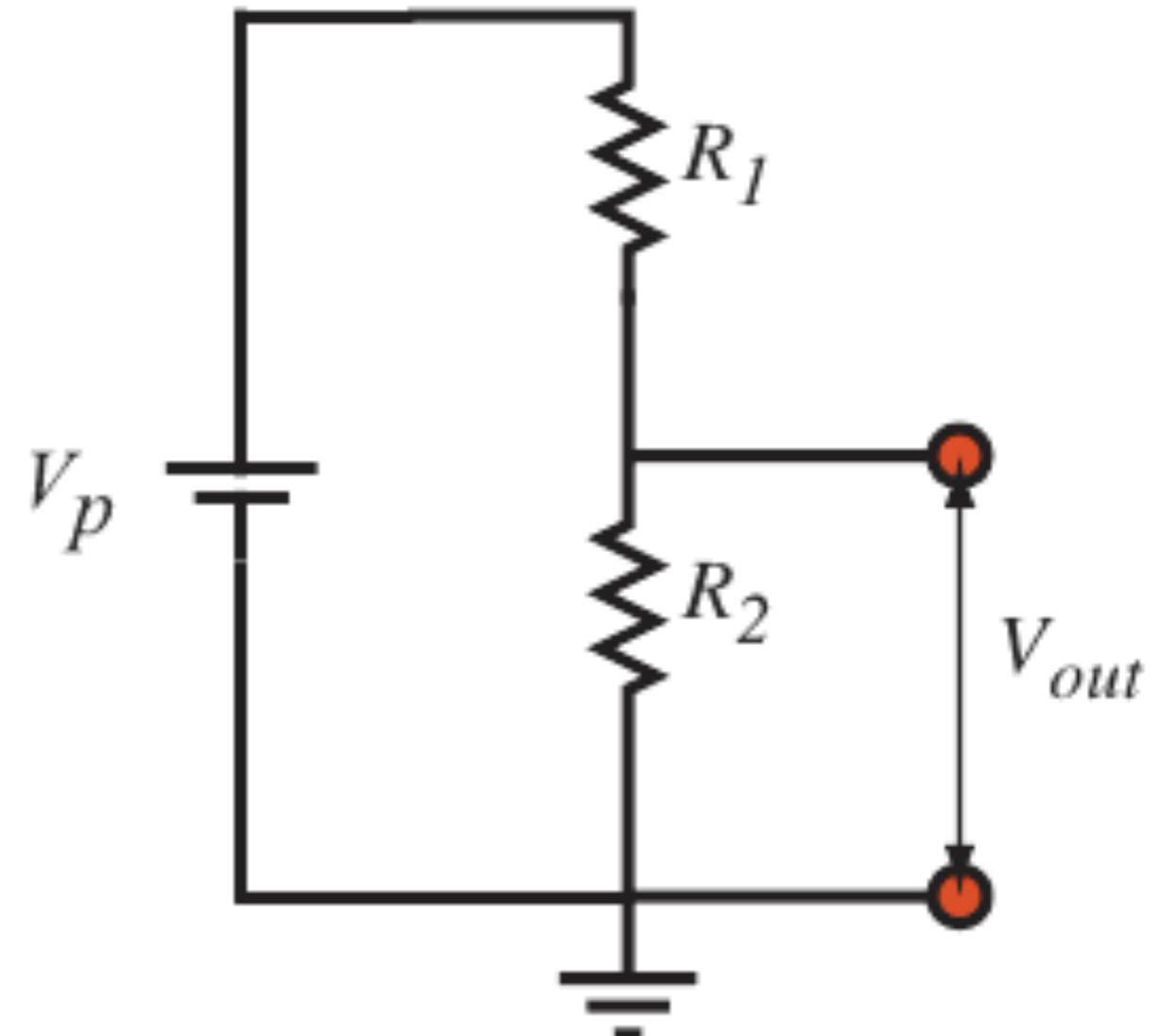
**Light dependant resistor, LDR as Analog input, and LED as Analog output.**

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 Resistor.**

\*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
```

```
//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);  
  
    //outputting 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.

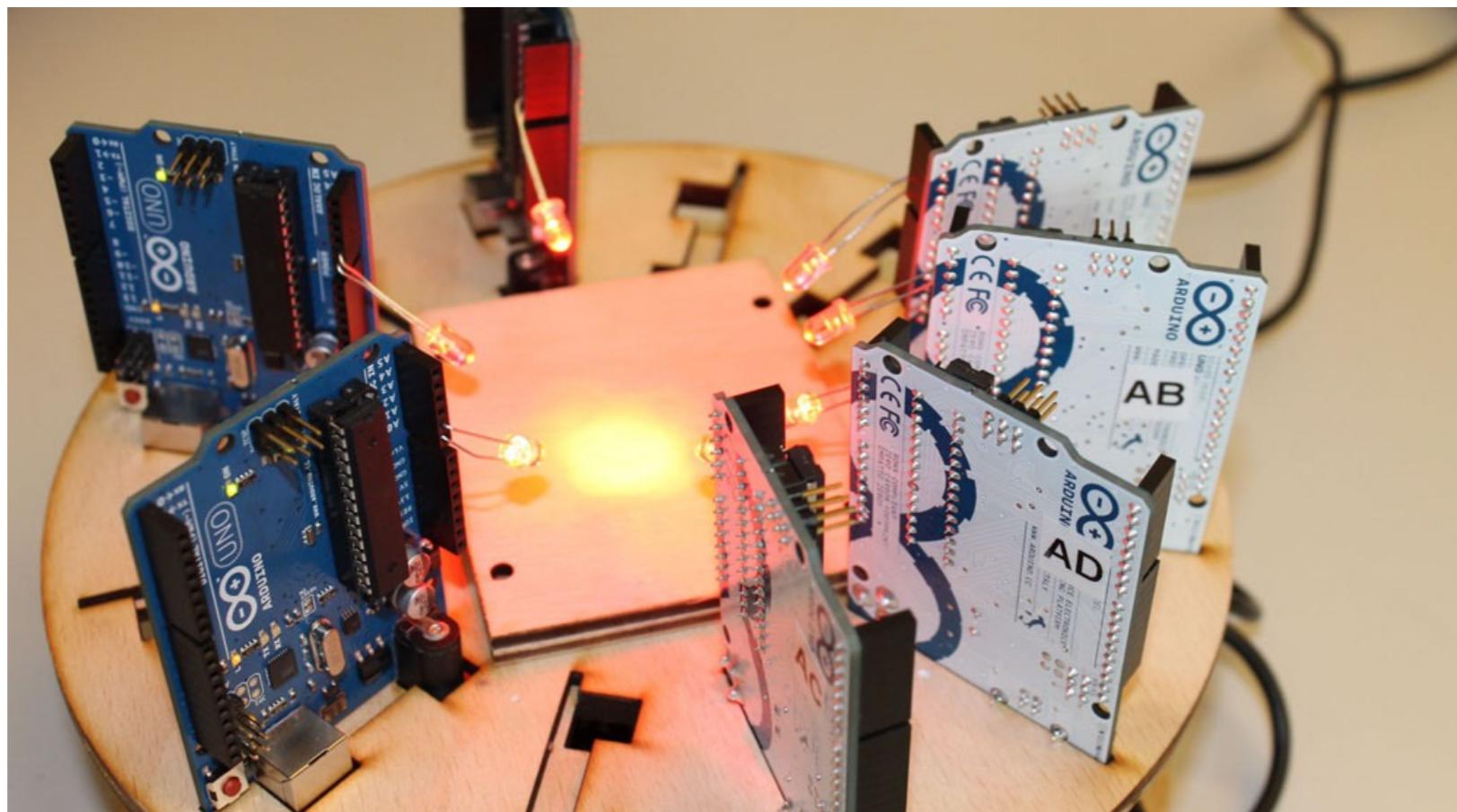


PanGenerator

# Challenge

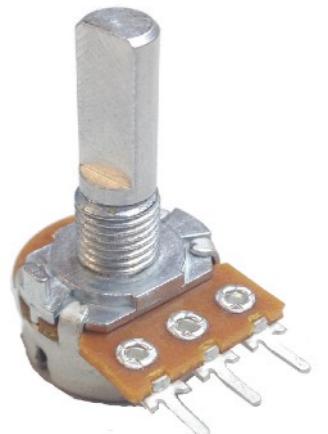
With the knowledge from the previous 2 examples attempt to get your Arduino **respond** to your partners Arduino with **Light!** Like the modules in the video.

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

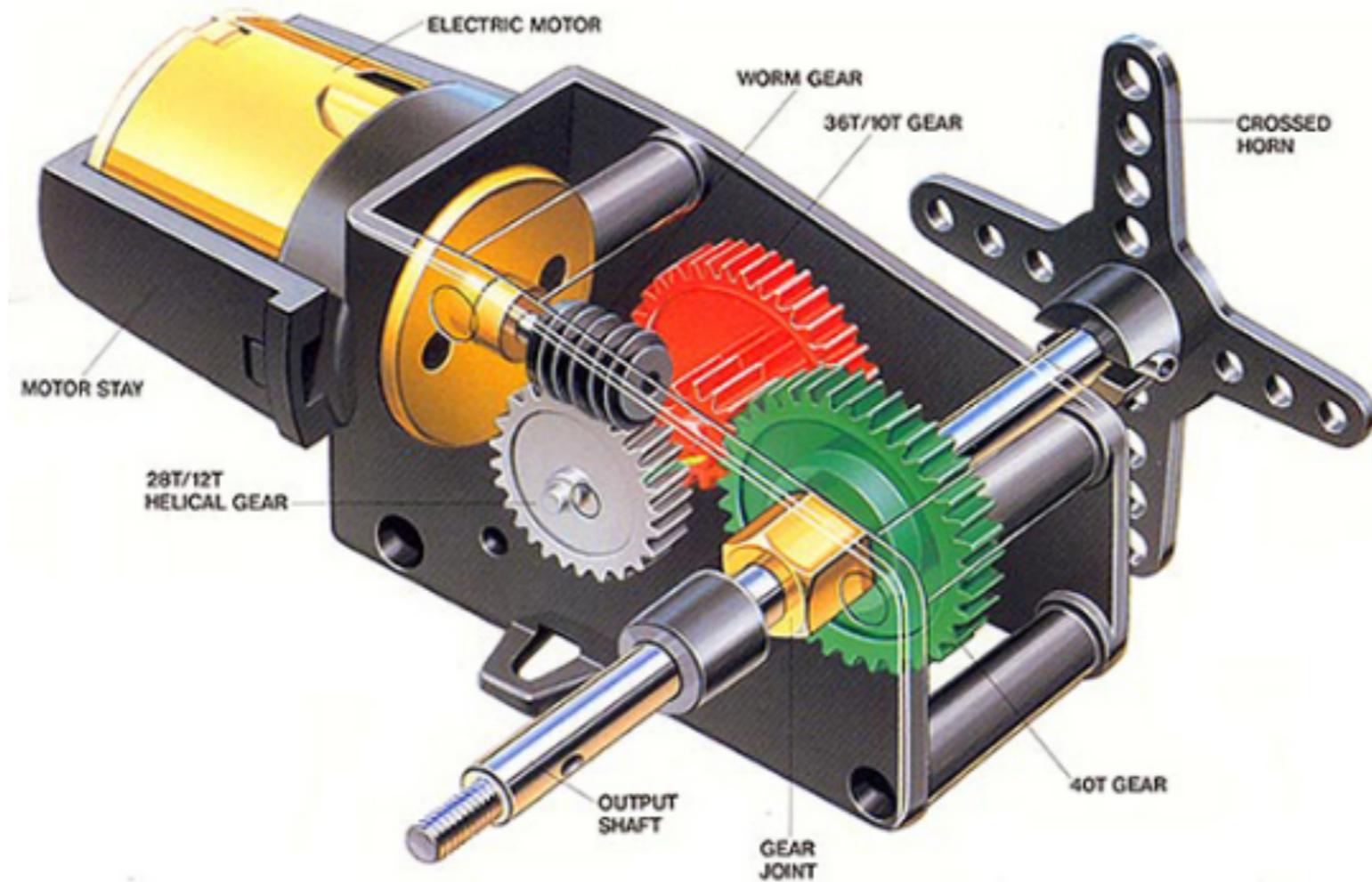


# Add features

- 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.



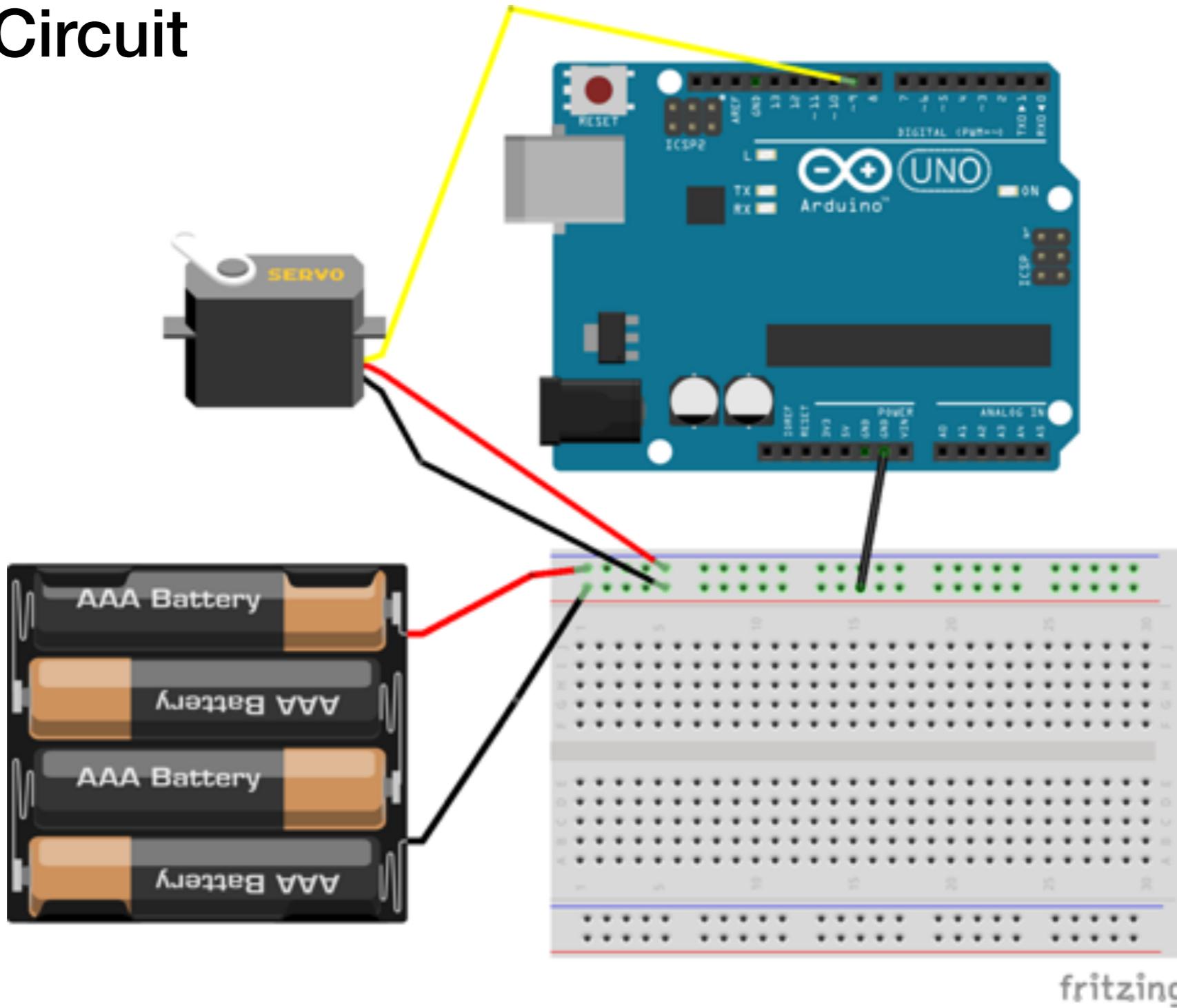
# Servo Motors - Giving your projects motion



- Super easy to use.
- Provide smooth accurate motion.
- Can be super strong!



# 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?**



## Servo\_Control

```
//variable used to define servos angle.  
float pos = 0;  
  
// variable to store the servo position  
float ang = 0;  
  
// Servos sweep speed.  
float vel = 0.1;  
  
void setup() {  
  
    //command for attaching a servo to a specific pin.  
    myservo.attach(3); // attaches the servo on pin 9 to the servo object  
  
}  
  
void loop() {  
  
    //do some maths  
    pos = 90 + 20*sin(ang);  
  
    //increase angle variable.  
    ang += vel;  
  
    //command to send servo to specific angle  
    myservo.write(pos);  
  
    //wait a little bit to let the servo reach its goal  
    delay(50);
```

Function to initialise servo

Function to set servo angle, 0 - 180 degrees

{