Hardware Workshop:

Arduino

A super fun knowledge share and prototyping workshop for Vigets

viget

Agenda

01 Background

02 Arduino Platform

03 Prototyping Basics

04 Getting Started

05 Helpful Resources

Let’s Build Something!

Background

What is this thing?

Development boards + programming language + IDE

Why is it a thing?

- **A teaching tool** - developed by the Interaction Design Institute Ivrea (Italy) to teach students about electronics and programming.
- **Open source** - open, inexpensive, and relatively easy to use.
- **Widely adopted** - supported by a huge maker-hobbyist community.
- **Resources** - a proliferation of boards and open-source libraries available for use.

Specifics

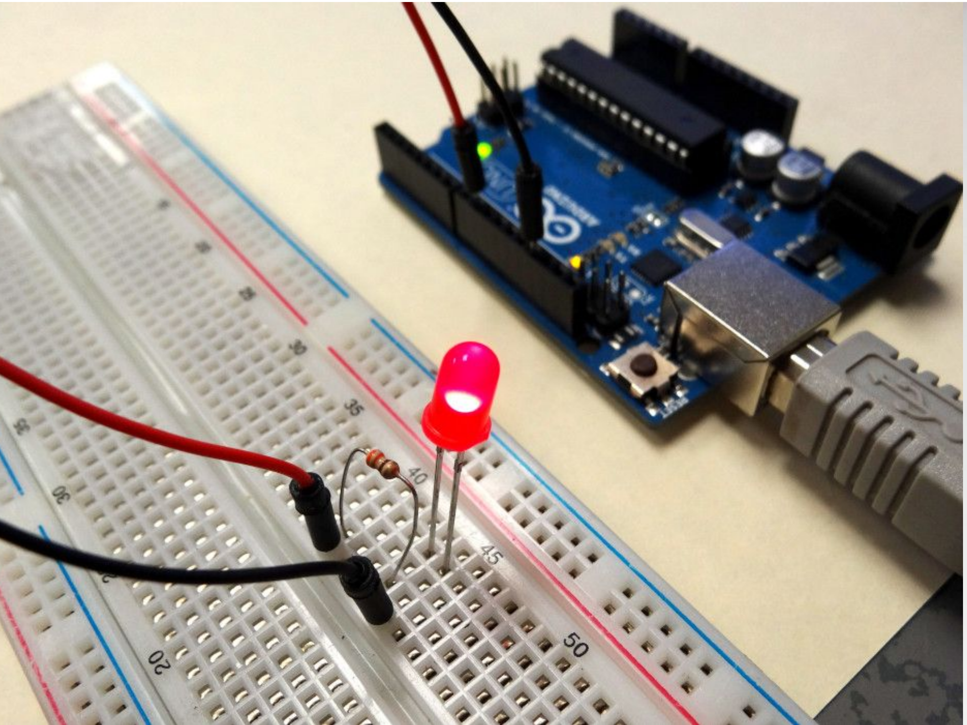
- **IDE** runs on Windows, Mac, and Linux. Free download.
- You can build your own **breakout boards** for controlling peripherals.
- The "OG" **Arduino Uno R3** costs $22; You can find beginner kits from $60-100.

Accessible

Adaptable

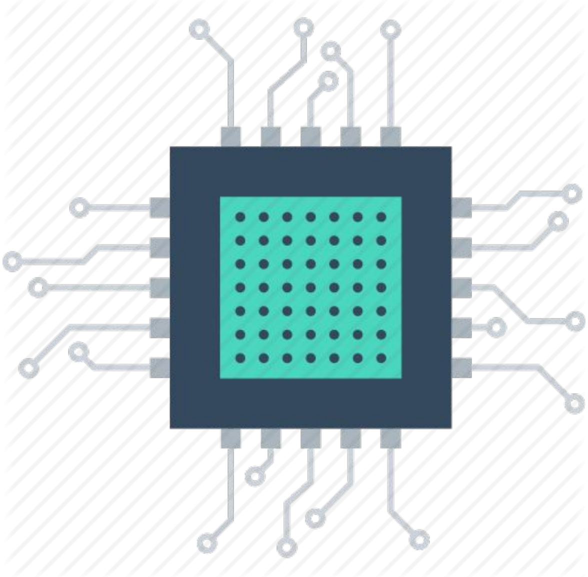
Budget Friendly

What it can do



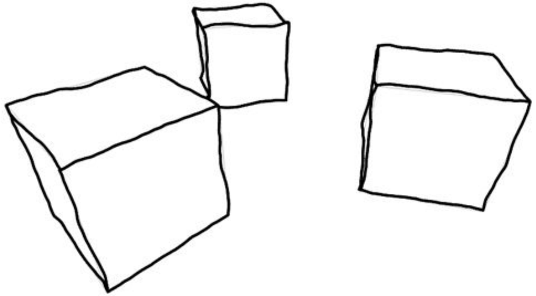
Microcontrollers

- Onboard the Arduino board.
- Integrated Circuit (IC) with five components:
    - CPU
    - RAM
    - Clock
    - ROM
    - I/O
- “Single chip computer”
- Different from a microprocessor, which requires an operating system.



Simple. Efficient. Forever.

- Power the chip, and it does what you tell it to do.
- Can only run one control loop at a time.
- It does that one thing efficiently, forever.



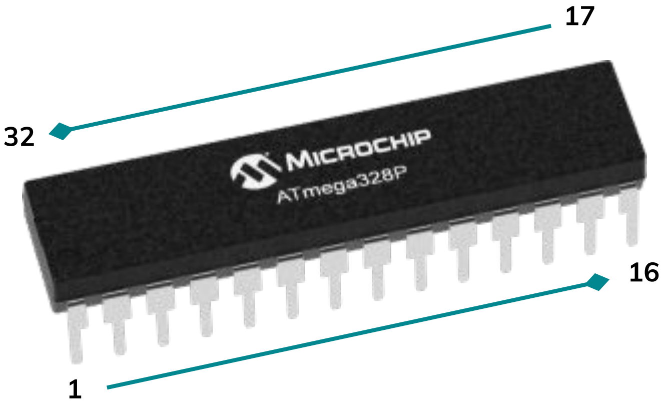
Why should I care?

- **Internet of Things** → a network of internet-connected ("smart") devices that traditionally are not connected to the internet.
- 30 billion devices deployed by 2020.
- Powered by microcontrollers and boards like that of Arduino.
- Affects almost every industry where **physical data can be leveraged**
- **A peek under the hood!**

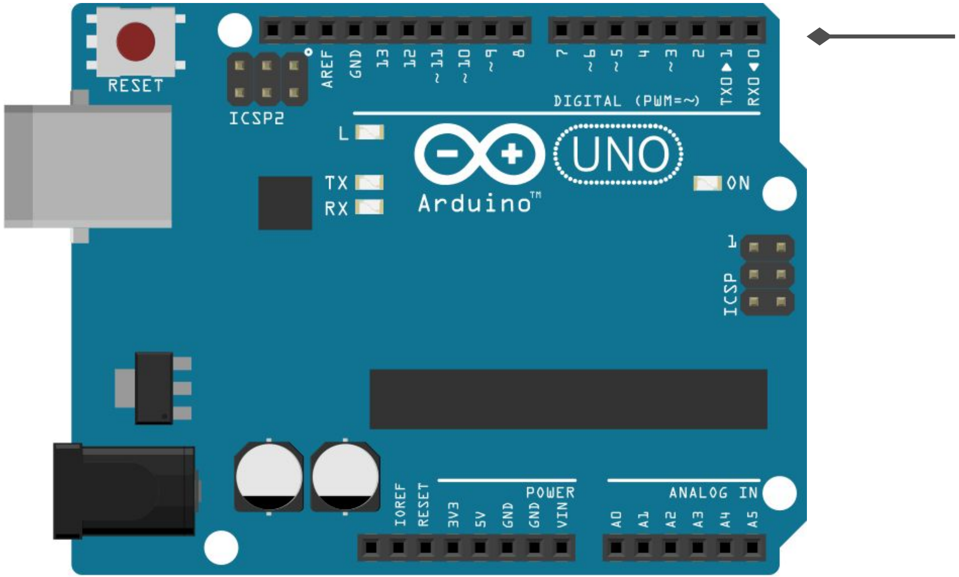
Arduino Platform

Brain

- AtMEGA328P
- 32 pin **microcontroller** from Atmel
  - Each pin serves **different and/or multiple functions**
- 32 KB of **memory**
- **Features:**
  - Analog-Digital Converter
  - Digital I/O
  - Common modes of serial communication

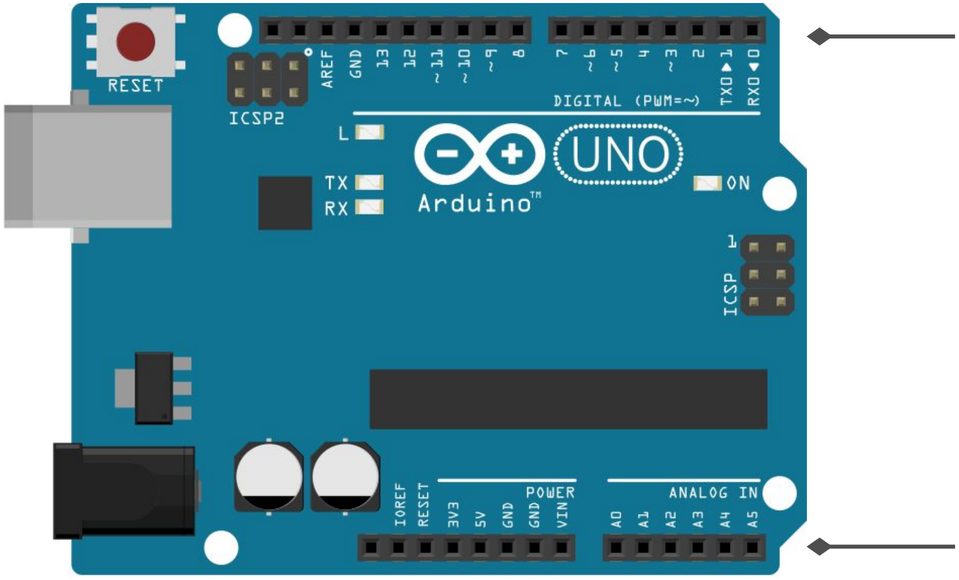


Board



Digital I/O pins

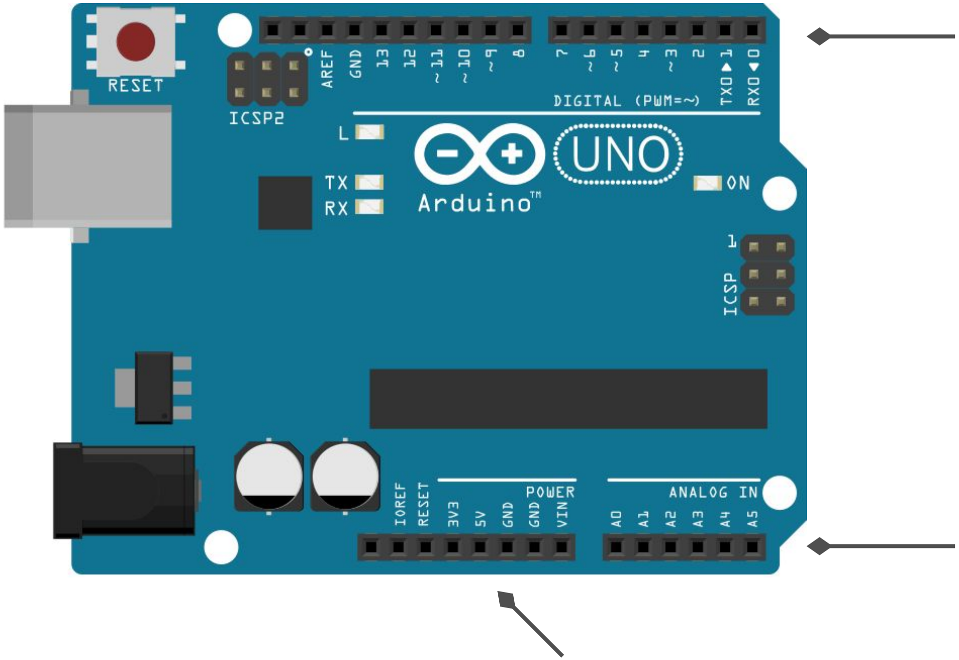
Board



Digital I/O pins

Analog Input pins

Board



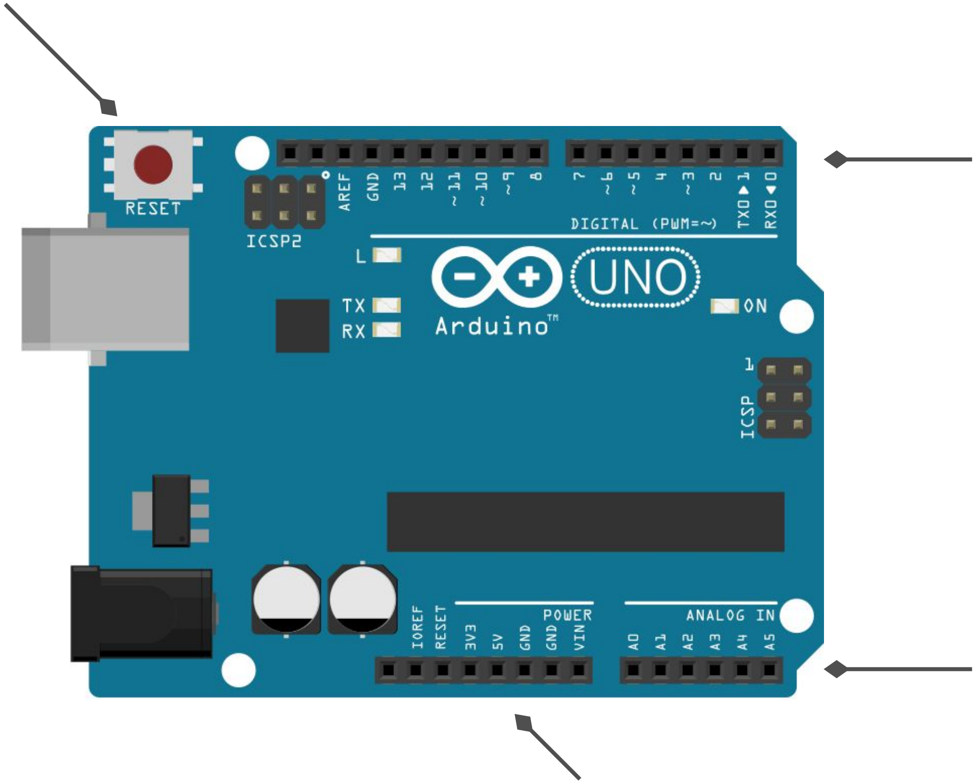
Digital I/O pins

Analog Input pins

Power Connections

Board

RESET button

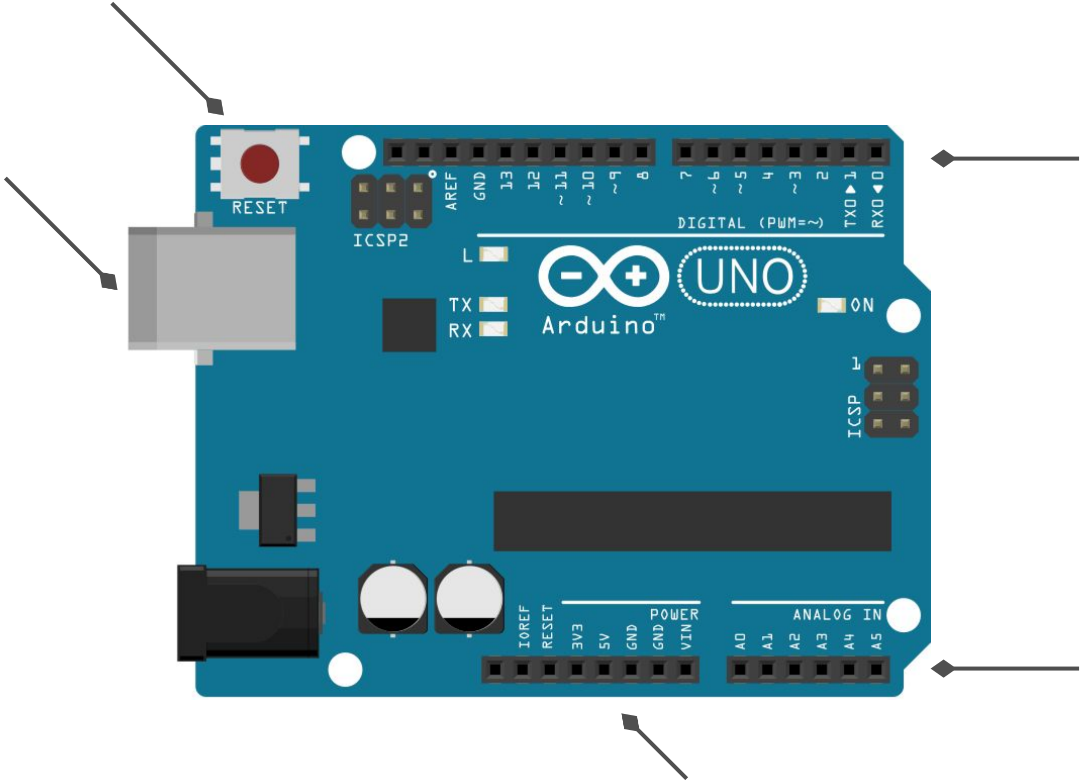


Digital I/O pins

Analog Input pins

Power Connections

Board



RESET button

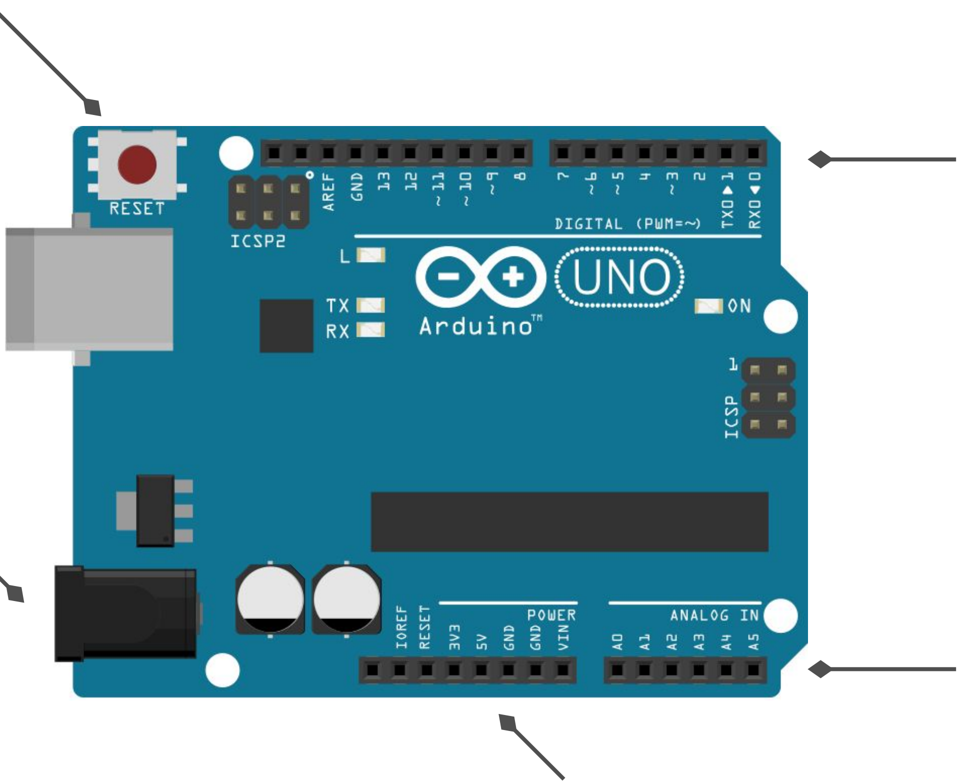
USB-B port

Digital I/O pins

Analog Input pins

Power Connections

Board



RESET button

USB-B port

Power supply jack

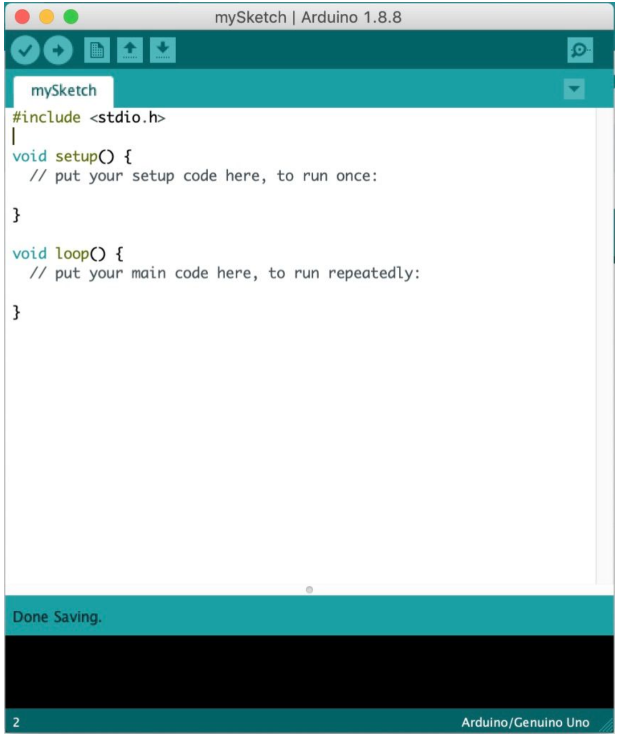
Digital I/O pins

Analog Input pins

Power Connections

IDE + Firmware

- C++ with helpful, built-in, libraries that abstract away the details
- Compiles your code and programs your board via USB → with a click of a button!
- Highly intuitive and sanitized embedded programing experience



```
#include <stdio.h>

void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Done Saving.

Firmware

```
#include <MyLibrary.h> //   include a library
#define MYCONSTANT 0   //   define a constant
int counter;           //   define counter, a global variable

void setup()  {
    // put your setup code here, to run once:
    counter = 0;       //   initialize counter
}

void loop()   {
    // put your main code here, to run repeatedly:
    counter = counter+1;    //  increment i every iteration

    if (counter > 100)  {   //  reset counter at 100 with MYCONSTANT
        counter = MYCONSTANT;
    }
}
```

Firmware

```
#include <MyLibrary.h> //   include a library
#define MYCONSTANT 0   //   define a constant
int counter;           //   define counter, a global variable

void setup()  {
     // put your setup code here, to run once:
     counter = 0;      //   initialize counter
}

void loop()   {
     // put your main code here, to run repeatedly:
     counter = counter+1;   //  increment counter every iteration

     if (counter > 100)  {  //  reset counter after 100
         counter = 0;
     }
}
```

Firmware

```
#include <MyLibrary.h> //   include a library
#define MYCONSTANT 0    //   define a constant
int counter;            //   define counter, a global variable

void setup()  {
    // put your setup code here, to run once:
    counter = 0;        //   initialize counter
}

void loop()   {
    // put your main code here, to run repeatedly:
    counter = counter+1;    //   increment counter every iteration

    if (counter > 100)  {   //   reset counter at 100 w/ MYCONSTANT
        counter = MYCONSTANT;
    }
}
```

Firmware

```
#include <MyLibrary.h>  //   include a library
#define MYCONSTANT 0    //   define a constant
int counter;            //   define counter, a global variable

void setup()  {
     // put your setup code here, to run once:
     counter = 0;       //   initialize counter
}

void loop()   {
     // put your main code here, to run repeatedly:
     counter = counter+1;    //   increment counter every iteration

     if (counter > 100)  {   //   reset counter at 100 w/ MYCONSTANT
         counter = MYCONSTANT;
     }
}
```

Firmware

```
#include <MyLibrary.h>  //   include a library
#define MYCONSTANT 0    //   define a constant
int counter;            //   define counter, a global variable

void setup()  {
     // put your setup code here, to run once:
     counter = 0;       //   initialize counter
}

void loop()   {
     // put your main code here, to run repeatedly:
     counter = counter+1;    //  increment counter every iteration

     if (counter > 100)  {   //  reset counter at 100 w/ MYCONSTANT
          counter = MYCONSTANT;
     }
}
```

Firmware

```
#include <MyLibrary.h>  //   include a library
#define MYCONSTANT 0    //   define a constant
int counter;            //   define counter, a global variable

void setup()  {
    // put your setup code here, to run once:
    counter = 0;        //   initialize counter
}

void loop()   {
    // put your main code here, to run repeatedly:
    counter = counter+1;     //  increment counter every iteration

    if (counter > 100)   {   //  reset counter at 100 w/ MYCONSTANT
        counter = MYCONTANT;
    }
}
```

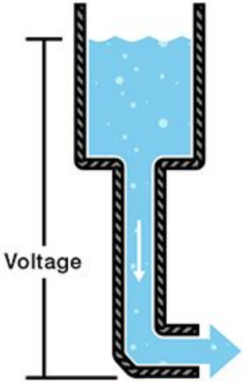
Prototyping Basics

Some basic electronics

- Resistors? Capacitors? LEDs? Oh my.
- The key ideas (for now):
    - Voltage (V) is the source of electricity
    - Current (I) is the flow of electricity
    - Electricity flows from high voltage to low voltage (+ to -)
    - mass resists the flow of electricity + the amount a given object resists that flow is called its Resistance (R)

Water Tank Analogy*

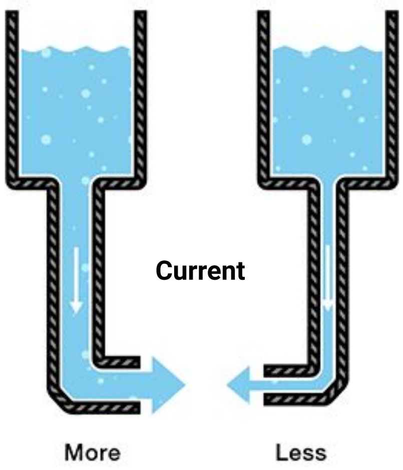
- Voltage = amount of water. Current = flow rate. Resistance = pipe size.



*credit to SparkFun

Water Tank Analogy*

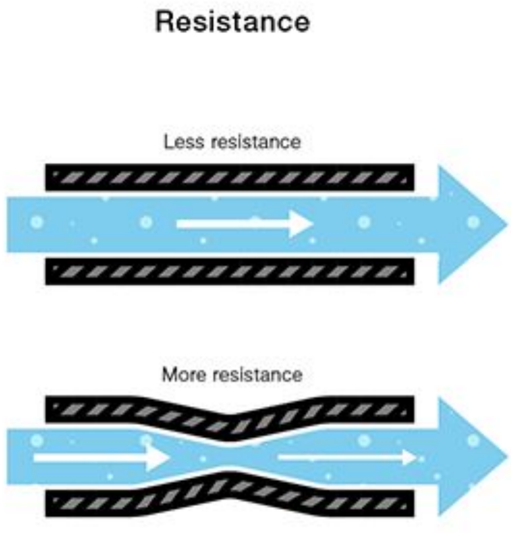
- Voltage = amount of water. Current = flow rate. Resistance = pipe size.



*credit to SparkFun

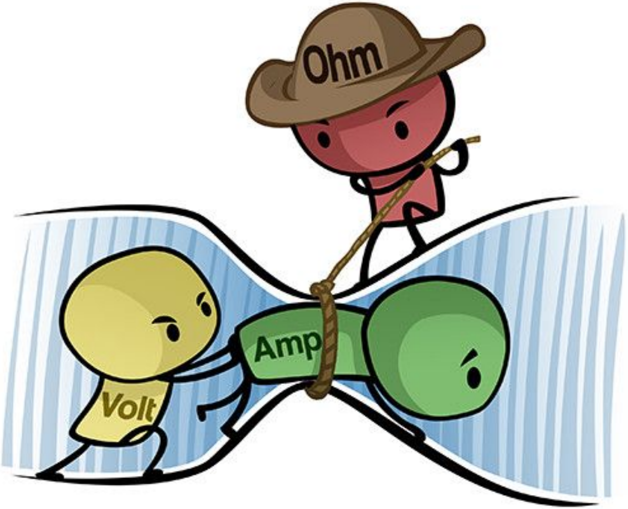
Water Tank Analogy*

- Voltage = amount of water. Current = flow rate. Resistance = pipe size



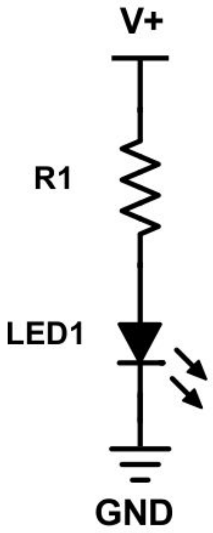
*credit to SparkFun

Another analogy…



*credit to the internet (?)

Basic Circuit



Getting Started

Arduino.cc

- **https://www.arduino.cc/**
- Arduino starter kit ~ $80
  - Comes w/ servo motors, LEDs, resistors, capacitors, start-up guide, wires etc.
- Arduino forum and blog
- Start small!
  - more equipment = **more capabilities** = **more money**

Helpful Resources

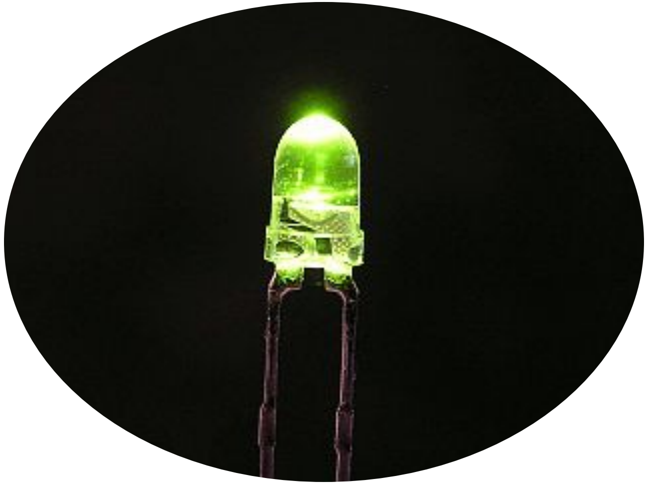
Maker Community

- A vibrant online community to help you **level up**
- **Our favorite resources:**
    - Youtube!
        - Hacker Shack
        - How To Mechatronics
    - Hackaday
        - And its dev community: hackady.io
    - Reddit
    - StackExchange

Let’s Build Something

Today’s Build:

- Step 1 → Toggle 1 LED
- Step 2 → Cycle 3 LEDs, “traveling light” functionality
- Step 3 → When motion sensor is flicked, flash all 3 LEDs 3 times

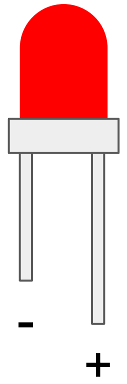


Setup

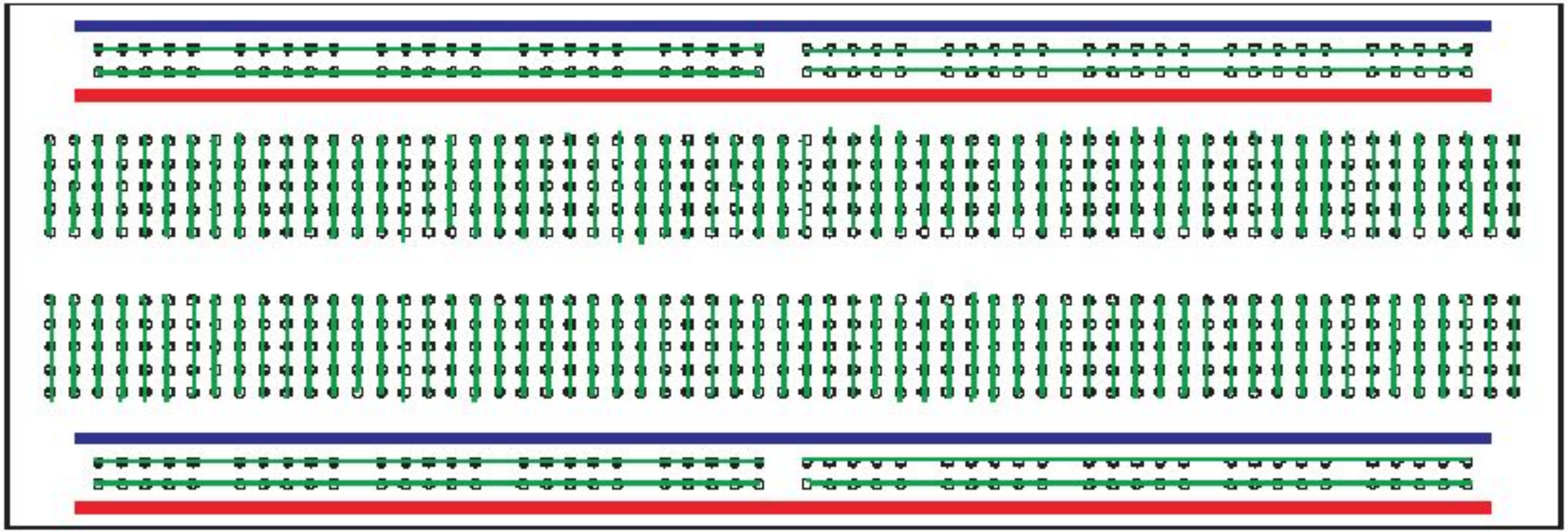
- Get into pairs
- Gather the following items:
  - 1 Arduino Uno
  - 1 USB-B cable
  - 3 LEDs
  - 3 resistors (1kΩ)
  - 7 wires
  - 1 vibration sensor

LED

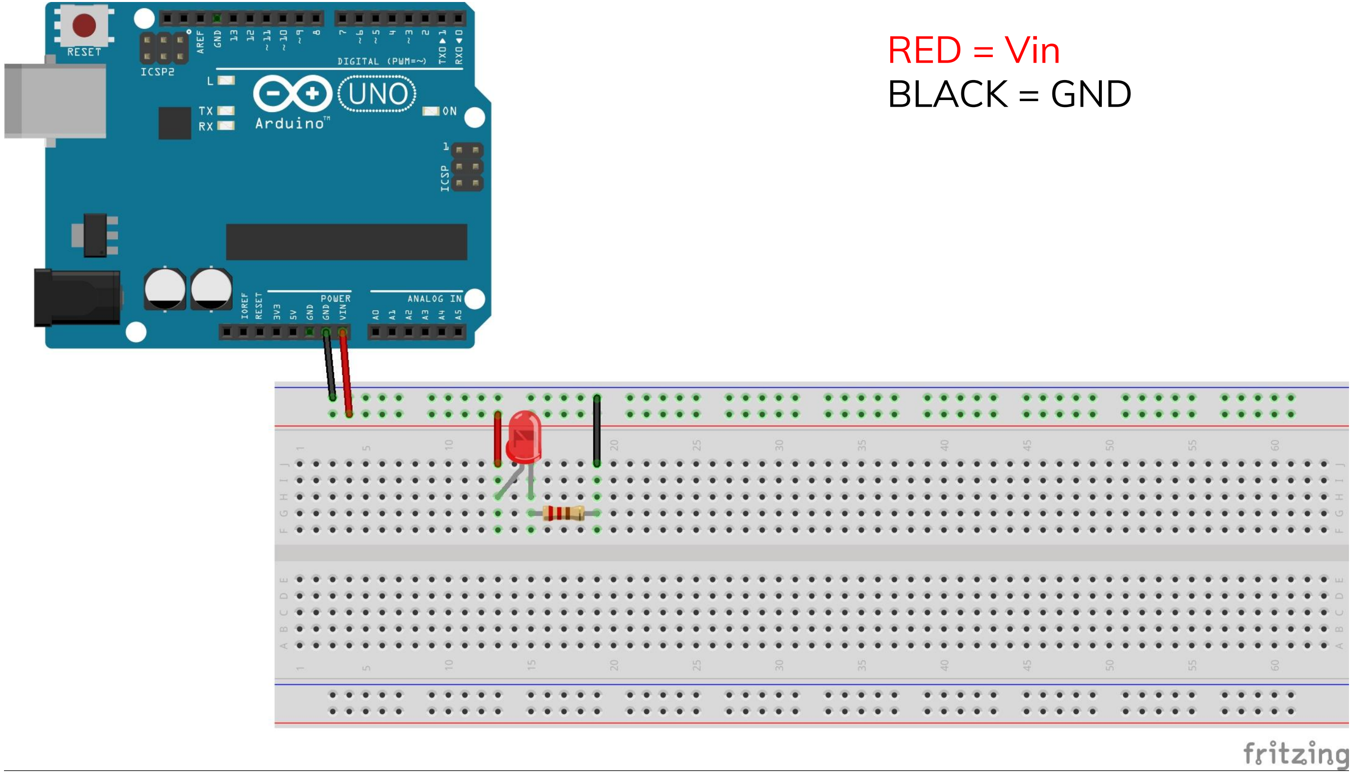
- “Light Emitting Diode”
  - Emits light
  - Current only flows one way
- Two leads: one positive (long), one negative (short)



Breadboard

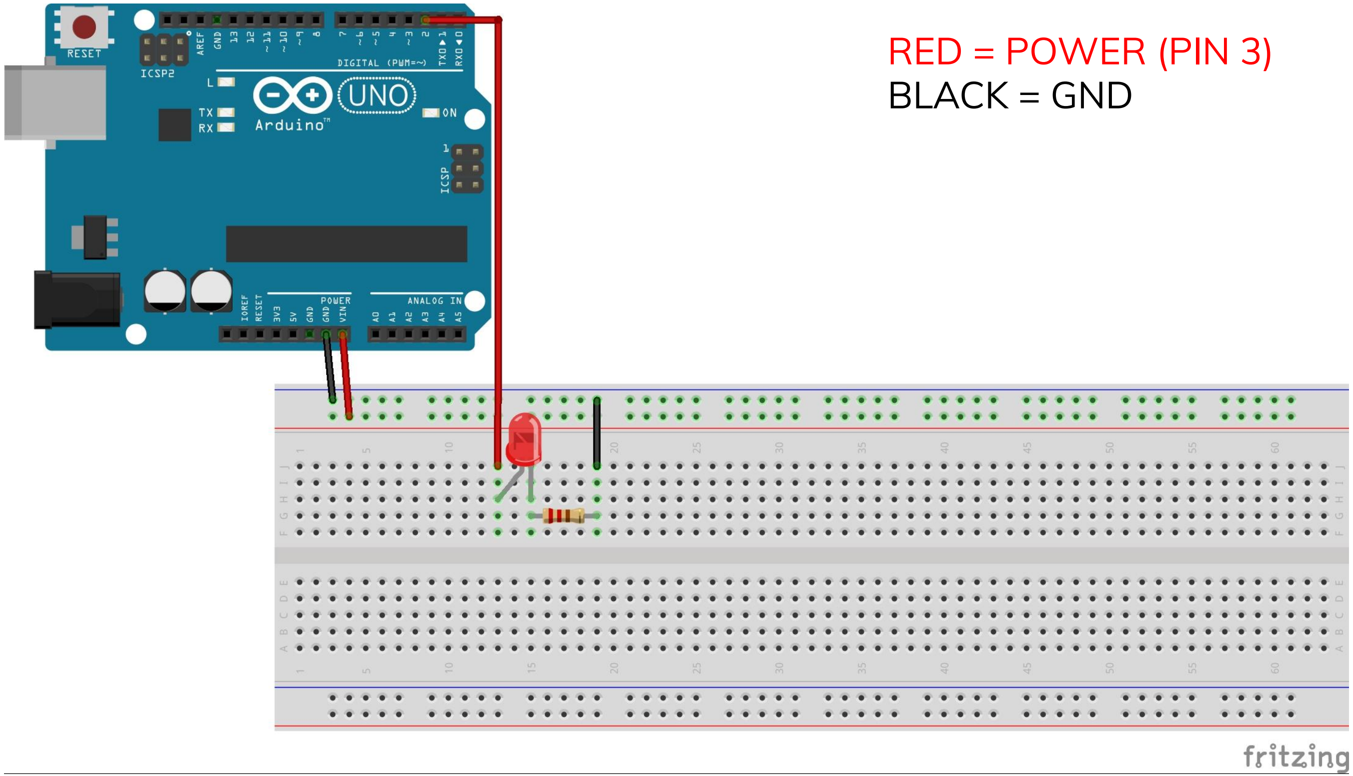


Step 0



RED = Vin
BLACK = GND

Step 1

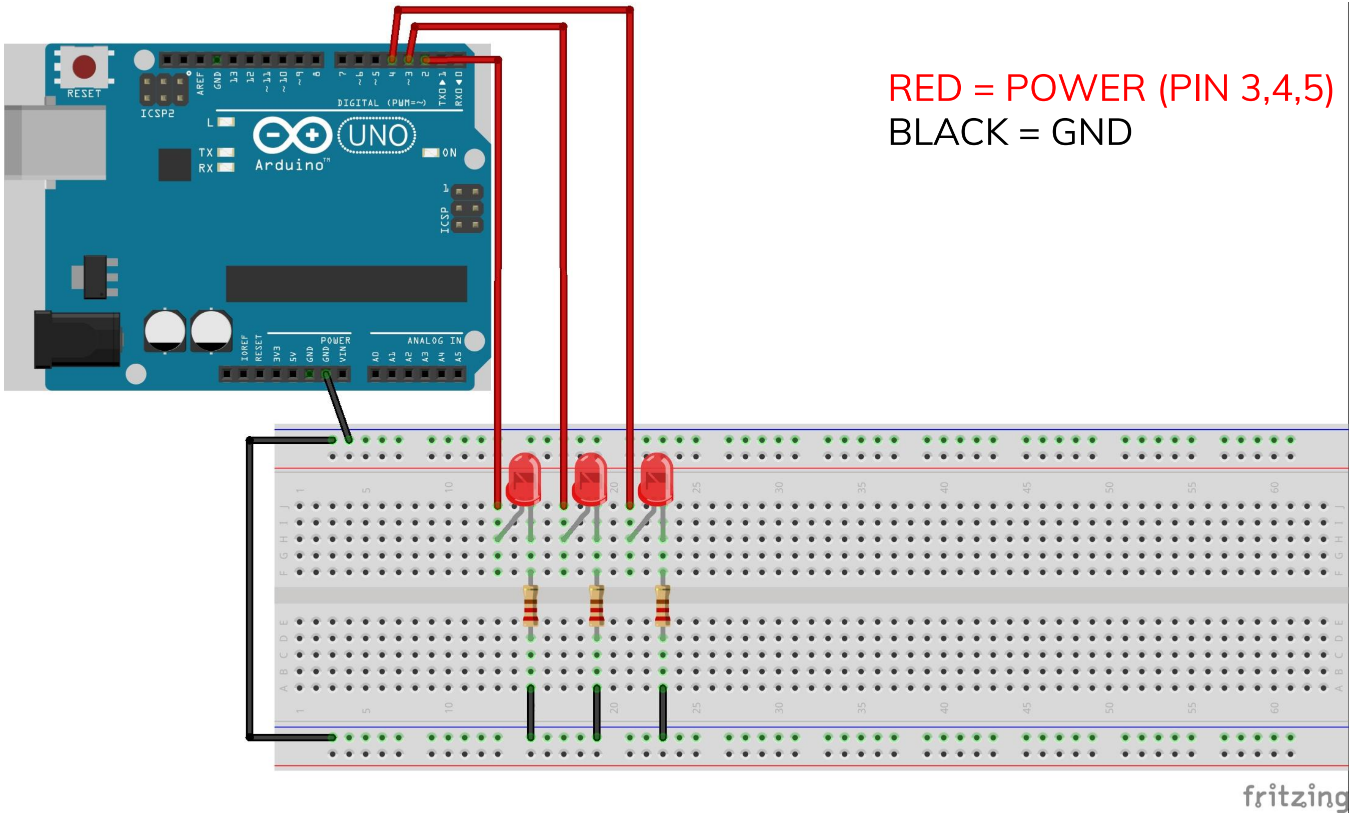


RED = POWER (PIN 3)
BLACK = GND

Debugging

- Arduino has a built-in serial monitor → a luxury
- No way to directly print output
  - simple LED circuit = **your best friend**

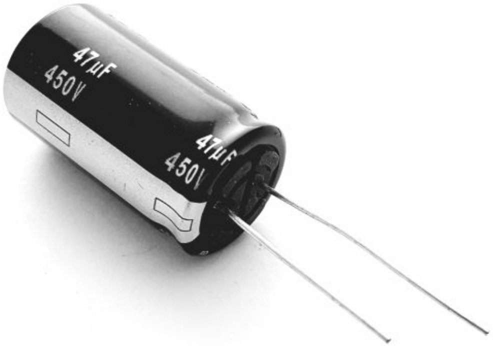
Step 2



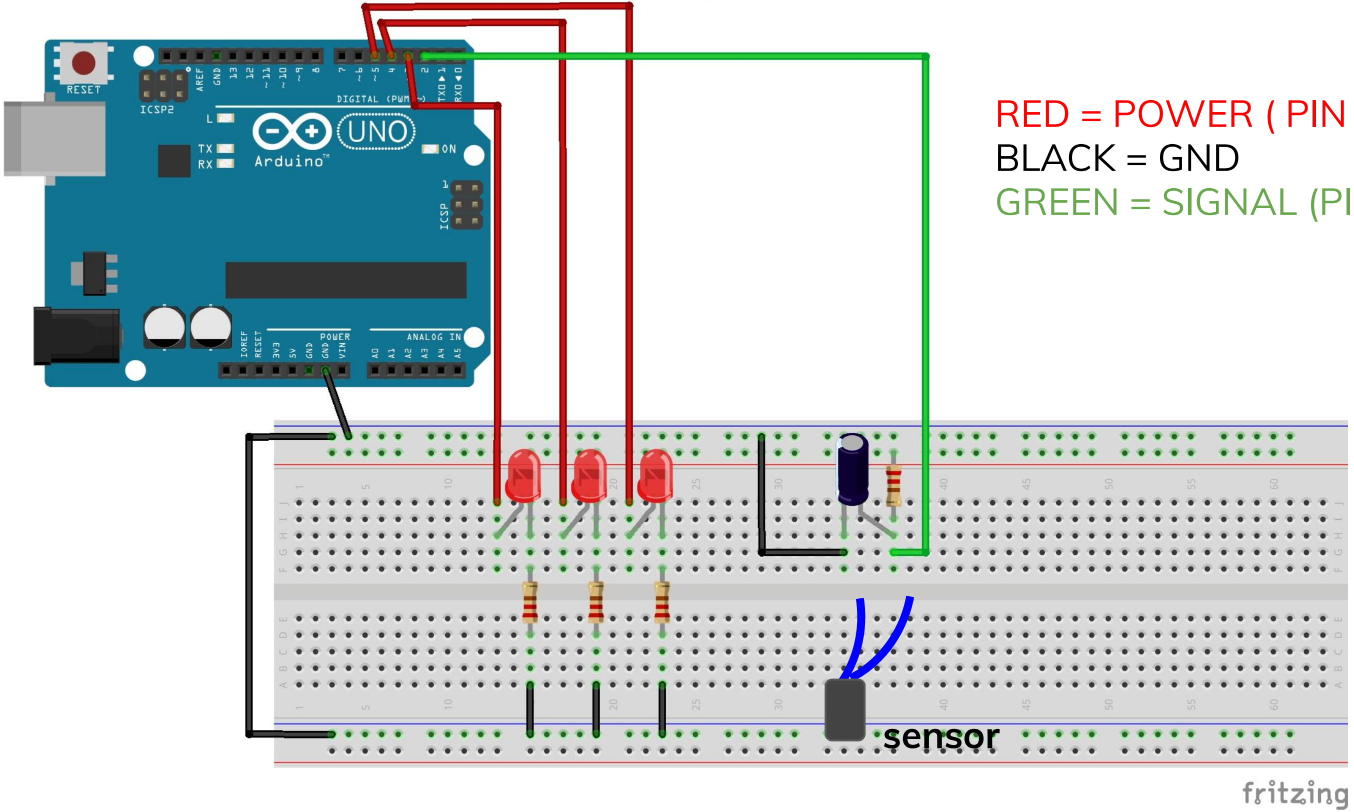
RED = POWER (PIN 3,4,5)
BLACK = GND

Capacitor

- Don’t worry about what it does
- Positive side (long leg), negative side (short leg)
- Why are we using it?
  - To reduce the messy signals



Step 3



RED = POWER (PIN 3,4,5)

BLACK = GND

GREEN = SIGNAL (PIN 2)