Engineering Technology (ENGR 101)

LED and user input

Concepts: INPUT vs. OUTPUT

Inputs is a signal / information going into the board.

Examples: Buttons Switches, Light Sensors, Flex Sensors, Humidity Sensors, Temperature Sensors…

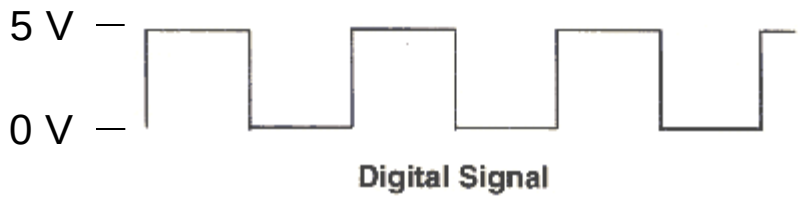
Output is any signal exiting the board.

Examples: LEDs, DC motor, servo motor, a piezo buzzer, relay, an RGB LED

Referenced from the perspective of the microcontroller (electrical board).

Analog vs. Digital

- Microcontrollers are **digital** devices – ON or OFF.



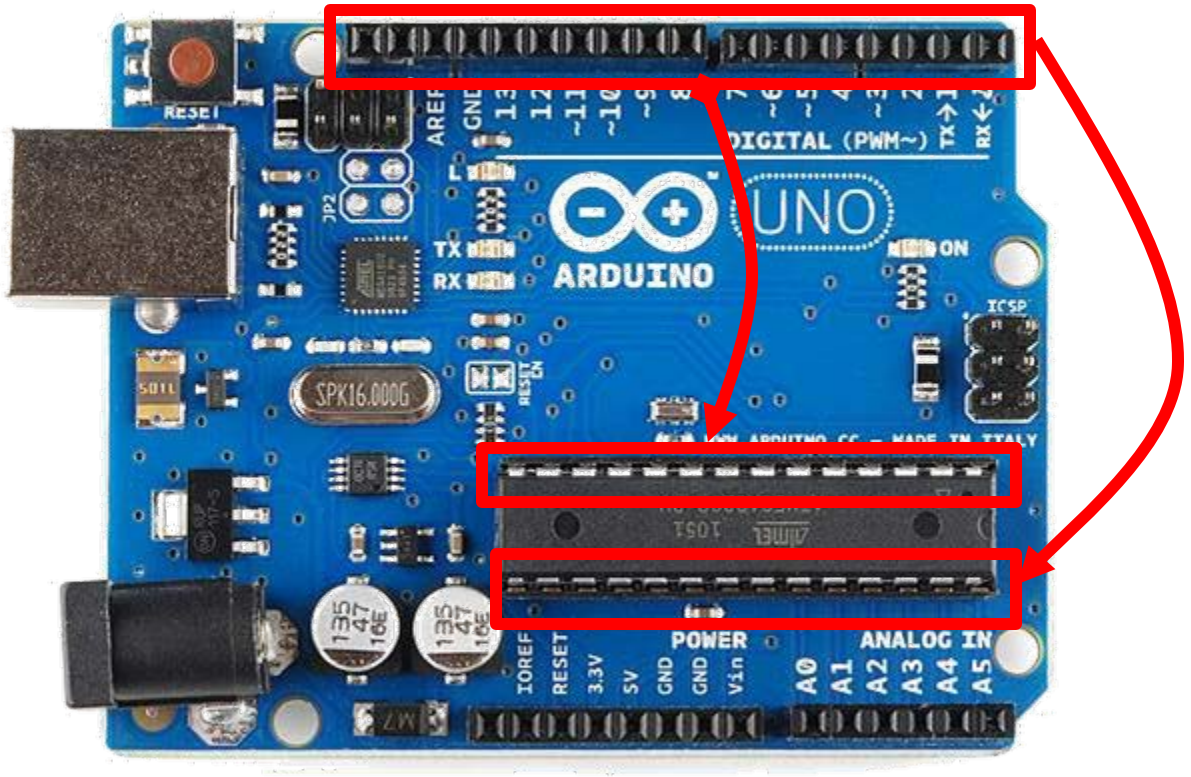
Digital Signal

- **Analog** signals are anything that can be a full range of values.

Analog Signal

Pins

- Pins are wires connected to the microcontroller
- Pins are the interface of the microcontroller
- Pin voltages are controlled by a sketch
- Pin voltages can be read by a sketch



Output Pins

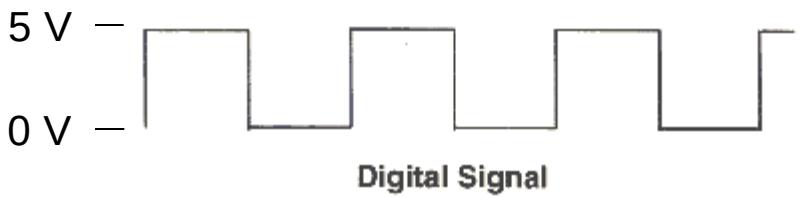
- Output pins are controlled by the Arduino
  - Voltage is determined by your sketch
  - Other components can be controlled through outputs

Input Pins

- Input pins are controlled by other components
- Arduino reads the voltage on the pins
- Allows it to respond to events and data

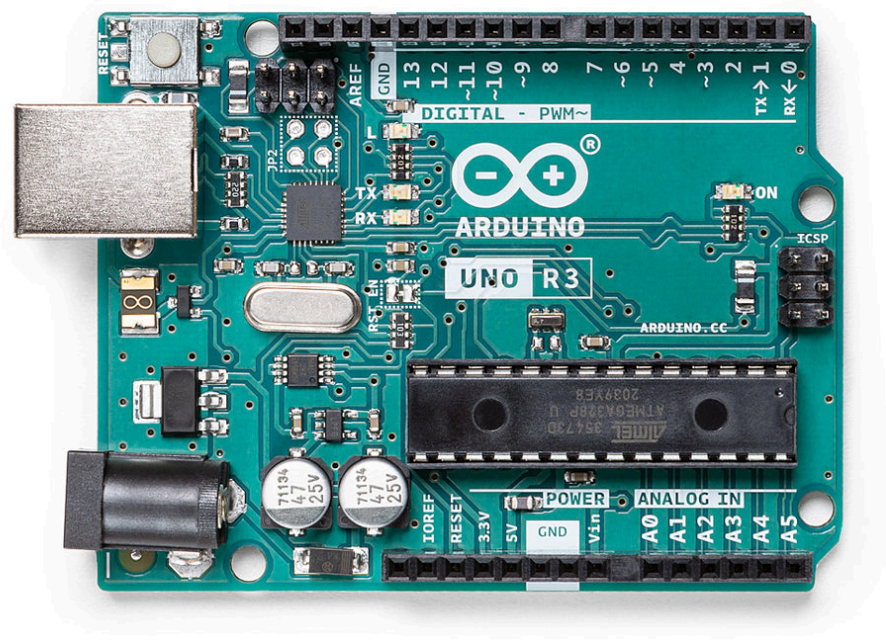
Analog vs. Digital Pins

- Some pins are digital-only
  - Read digital input, write digital output
  - 0 volts or 5 volts



- Some pins can be analogue inputs
  - Can read analogue voltage on the pin
  - Useful for analogue sensors

- Analogue-only pins are clearly labeled (A0 to A5)
- No pin can generate an analogue output



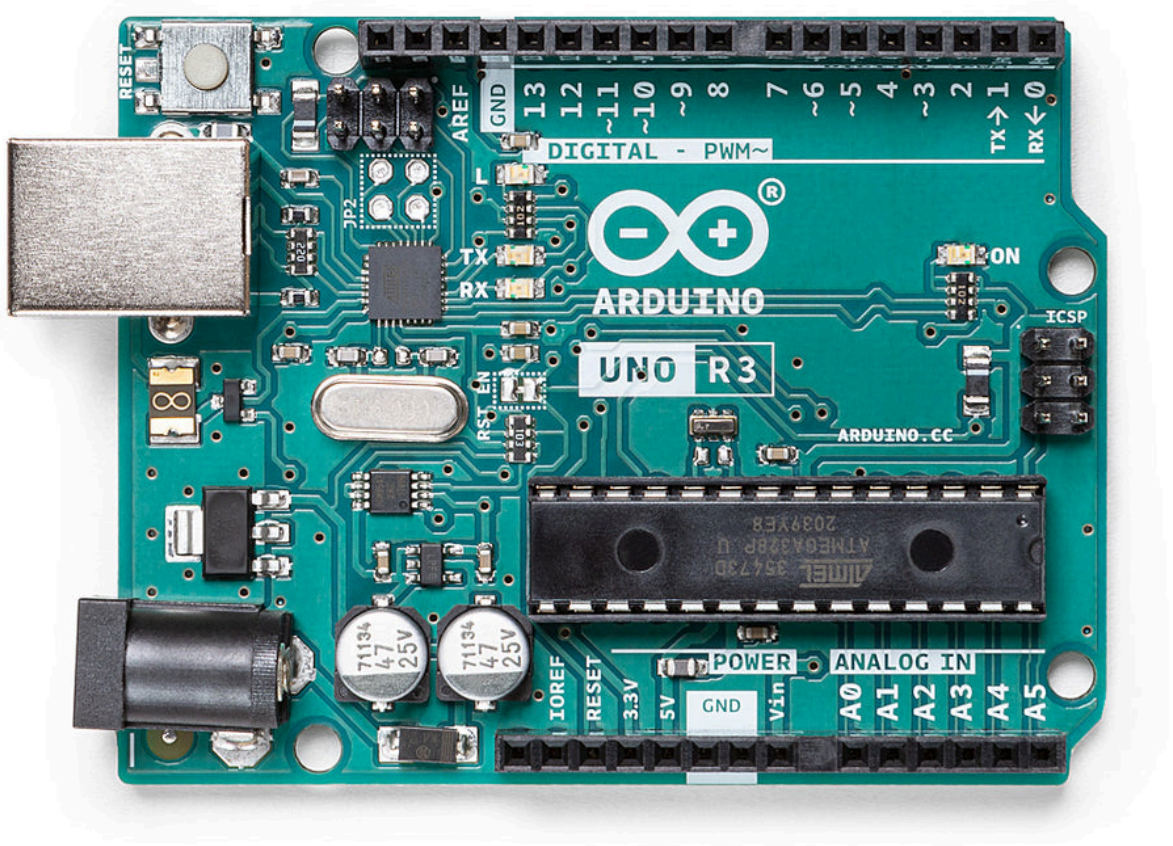
Input/Output (I/O)

- These functions allow access to the pins
  void pinMode(pin , mode)
- Sets a pin to act as either an input or and output
- pin is the number of the pin
  - 0-13 for the digital pins
  - A0 –A5 for the analogue pins
- mode is the I/O mode the pin is set to
  - INPUT
  - OUTPUT

Example:

```
pinMode(3, OUTPUT);
```

Digital pin 3 is set as OUTPUT



Digital Output

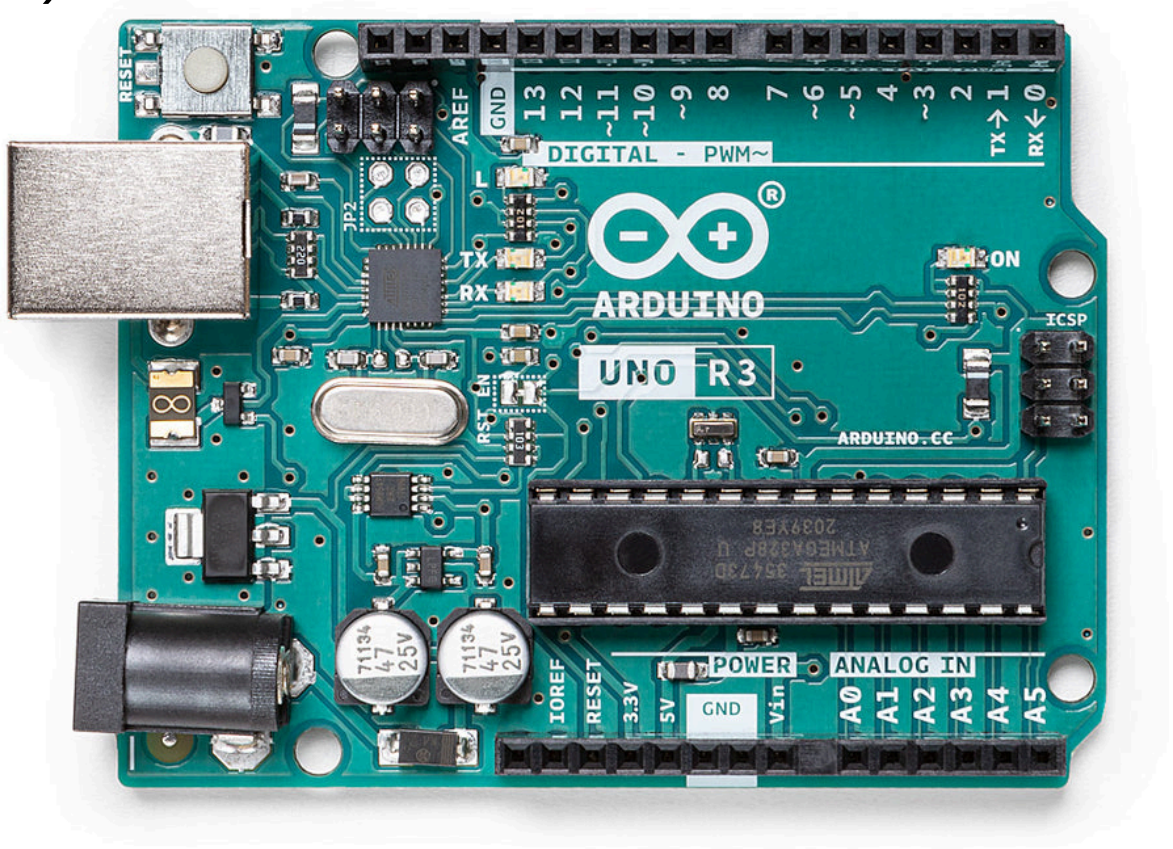
void digitalWrite(pin, value)

- Assigns the state of an output pin
- Assigns either LOW (0 volts) or HIGH (5 volts)

- Example:

```
digitalWrite(3, HIGH);
```

- Digital pin 3 is set HIGH (5 volts)



Delay

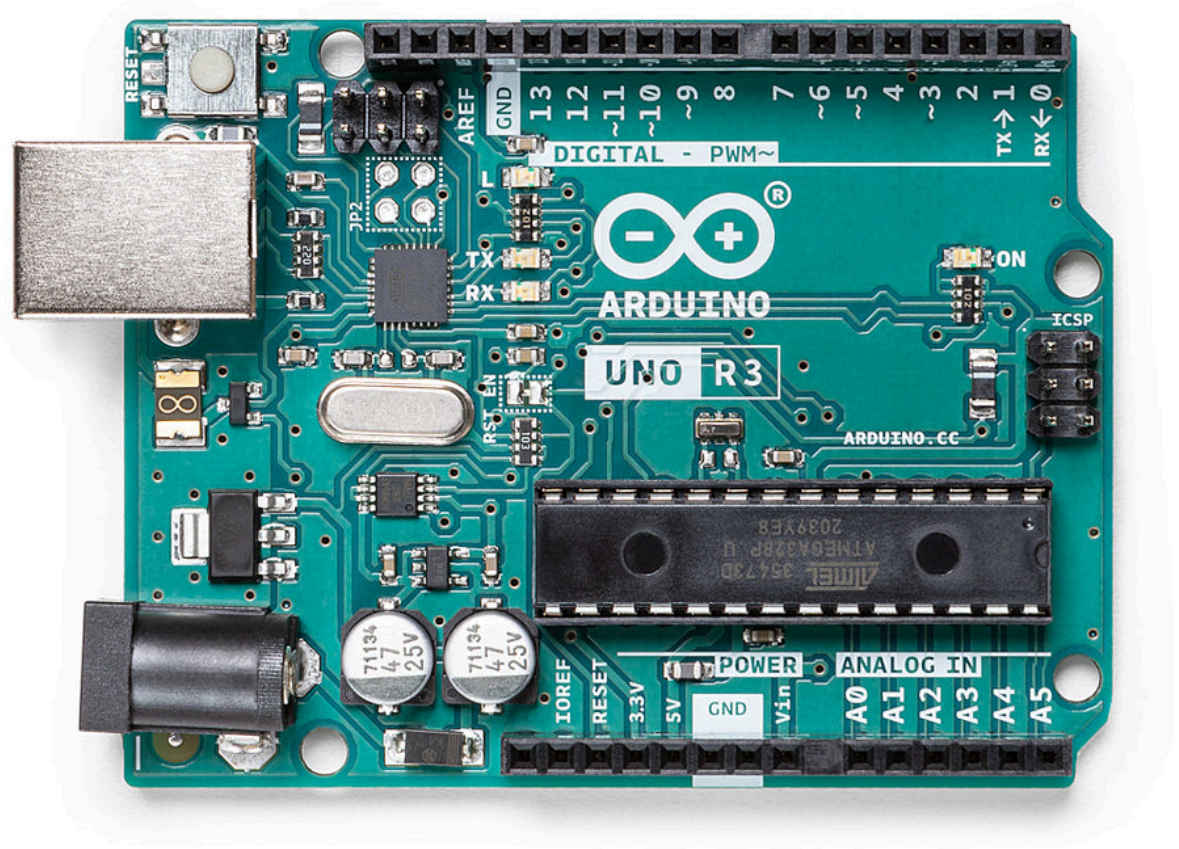
void delay(msec)

- Pauses the program for msec milliseconds
- Useful for human interaction

- Example:

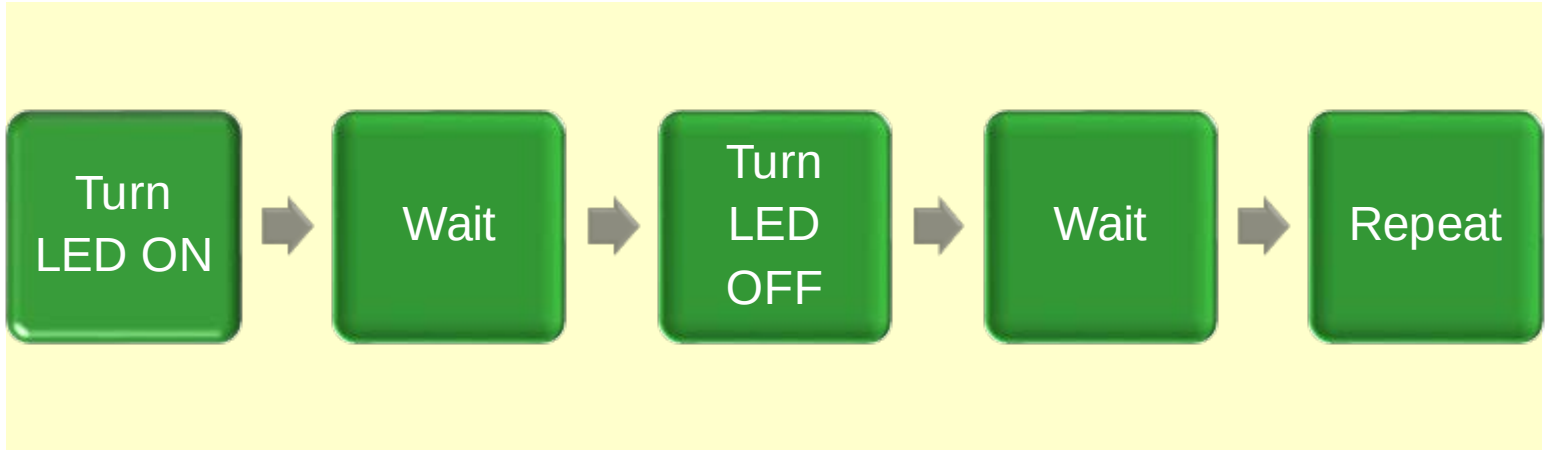
```
digitalWrite(3, HIGH);
delay(1000);
digitalWrite(3, LOW);
```

- Pin 3 is HIGH for 1 second



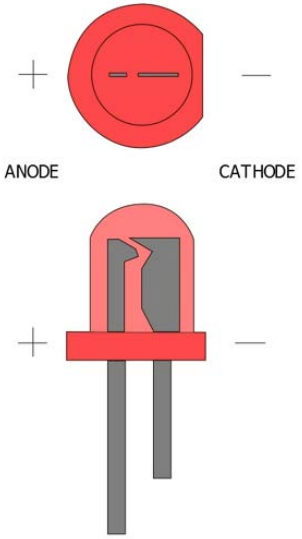
Blink LED

- The specific sketch you want to use here is called Blink.
- It's about the most basic sketch you can write, a sort of "Hello, world!" for Arduino.



Blink LED

```
void setup() {
    pinMode(13,OUTPUT);
}
void loop() {
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
}
```



Connected to pin 13

- Sending 5v to the LED
- Waiting a second
- Sending 0v to the LED
- Waiting a second

Common voltages to turn on LEDs

LED Color	Forward Voltage (V_F)
Infrared	1.5 - 1.7 V
Red	1.8 - 2.2 V
Yellow	2.1 - 2.2 V
Green	2.0 - 3.4 V
Blue	3.2 - 3.8 V

Putting up resistance

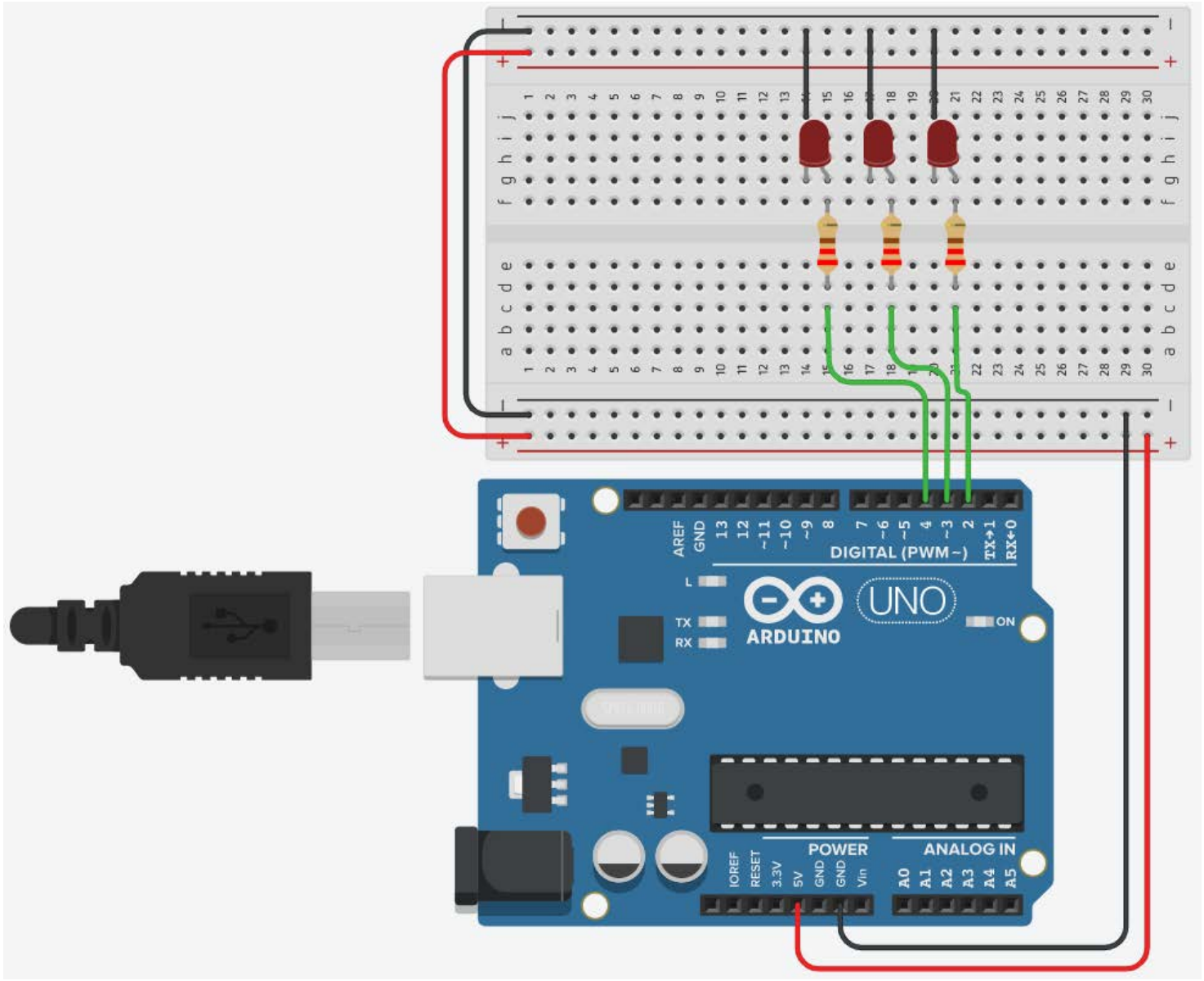
- Calculating the correct resistance is important for a safe circuit
- A digital pin supplies the LED with a source of 0 or 5V (volts)
- A typical red LED has an approximate maximum forward voltage of 2.1V (volts),
- Therefore, a resistor is needed to protect the LED.
- The LED draws a maximum current of approximately 25mA (milliamps).

$$R = \frac{V_{SUPPLY} - V_{FORWARD}}{I}$$

$$R = \frac{(5 - 2.1)}{0.025} = 116\ ohms$$

- Suitable values from various kits include 220Ω, 330Ω, and 560Ω.

Three Blinking LED



Three Blinking LED

- Setting each Arduino pin number, which is connected to LEDs, to mode of OUTPUT

```
//Defining LED pin numbers
const int led_1_Pin = 2;
const int led_2_Pin = 3;
const int led_3_Pin = 4;

void setup()
{
  // Setting the LED pins mode as OUTPUT
  pinMode(led_1_Pin, OUTPUT);
  pinMode(led_2_Pin, OUTPUT);
  pinMode(led_3_Pin, OUTPUT);
}
```

Three Blinking LED

- To turn on and off the LED once, you need to repeat **`digitalWrite()`**function twice

```
void loop()
{
  // Make LED 1 blink once
  digitalWrite(led_1_Pin, HIGH);
  delay(500); // Wait for 500 millisecond(s)
  digitalWrite(led_1_Pin, LOW);
  delay(500); // Wait for 500 millisecond(s)

   …
}
```

Three Blinking LED

- To turn on and off the LED twice, you need to repeat **`digitalWrite()`**function four times

```
void loop()
{
   …
  // Make LED 2 blink Twice
  digitalWrite(led_2_Pin, HIGH);
  delay(500); // Wait for 500 millisecond(s)
  digitalWrite(led_2_Pin, LOW);
  delay(500); // Wait for 500 millisecond(s)

  digitalWrite(led_2_Pin, HIGH);
  delay(500); // Wait for 500 millisecond(s)
  digitalWrite(led_2_Pin, LOW);
  delay(500); // Wait for 500 millisecond(s)

   …
}
```

Three Blinking LED

- To turn on and off the LED three times, you need to repeat **`digitalWrite()`**function six times
- It is much easier to use for-loop to repeat **`digitalWrite()`**function six times.

```
void loop()
{
  ...
  // Make LED 3 blink Three times
  for(int i = 0; i <3; i++){
     digitalWrite(led_3_Pin, HIGH);
     delay(500); // Wait for 500 millisecond(s)
     digitalWrite(led_3_Pin, LOW);
     delay(500); // Wait for 500 millisecond(s)
  }
...
}
```

Color-Coding

- It becomes even more important as you progress to more complex circuits.

- For example: traffic lights are colour-coded to give drivers a clear message of what to do:
  - Green means proceed.
  - Yellow means prepare to stop.
  - Red means stop.

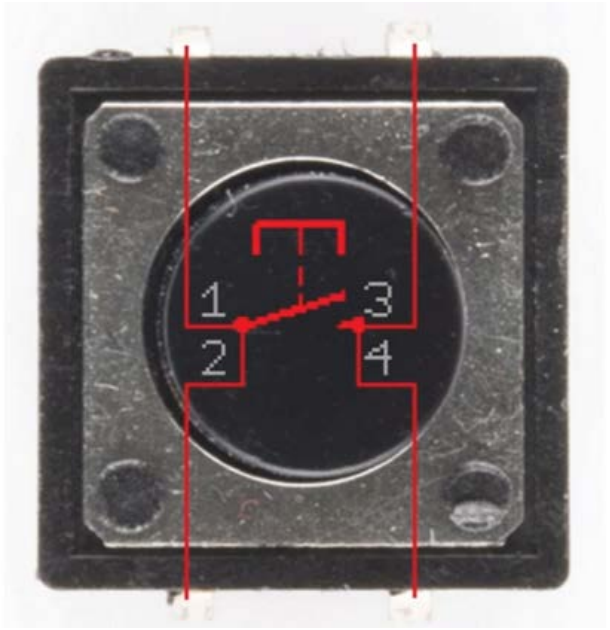
- There are a few conventions that can help you and others to understand your circuit:
  - Red is positive (+).
  - Black is negative (–).
  - Different colours are used for different signal pins.

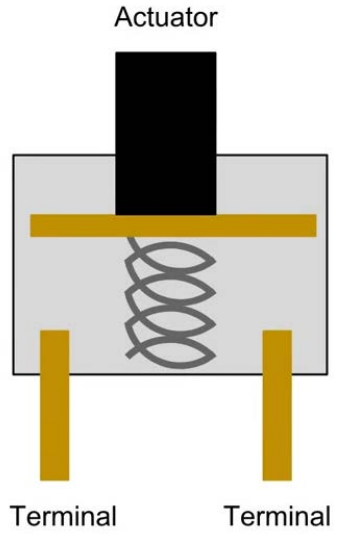
Input

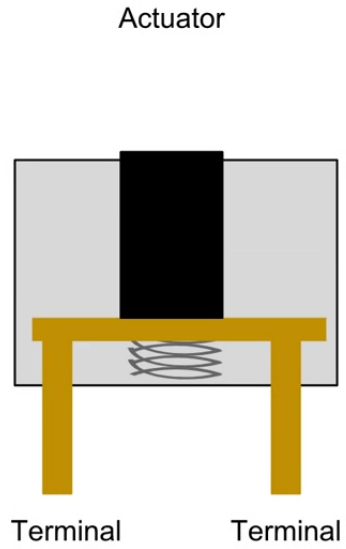
- Input is any signal entering an electrical system
- Both digital and analog sensors are forms of input
- Input can also take many other forms: Keyboards, a mouse, infrared sensors, biometric sensors, or just plain voltage from a circuit

Pushbutton Switch

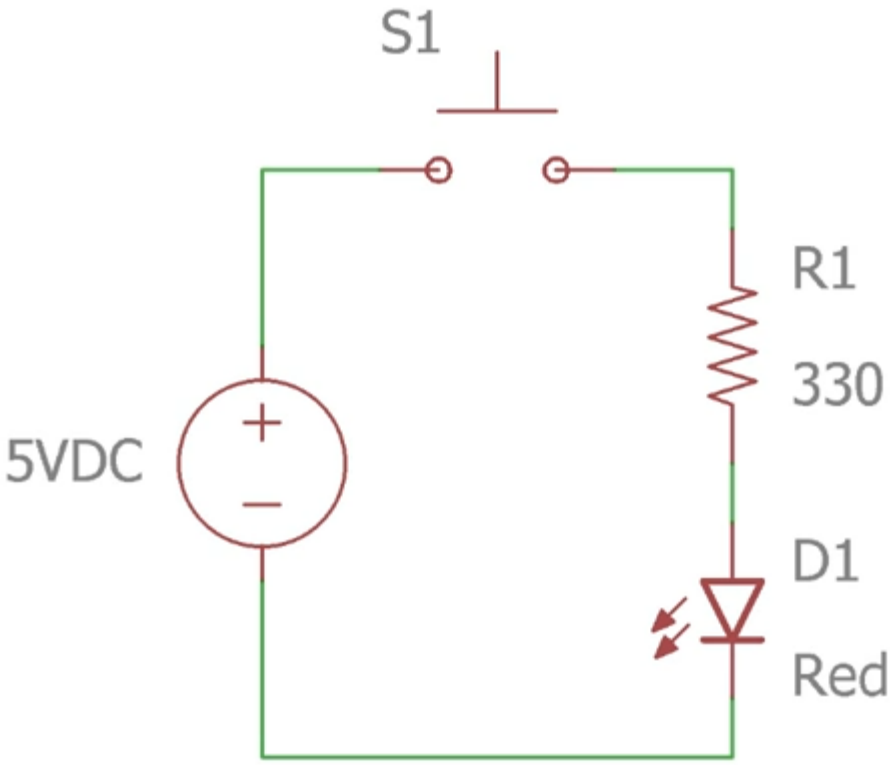
- Be careful with the orientation of the 4 legs
- Legs 1 and 2 are always connected
- Legs 3 and 4 are always connected
- Legs 1 and 3 are connected only when you press the button
- Legs 2 and 4 are connected only when you press the button



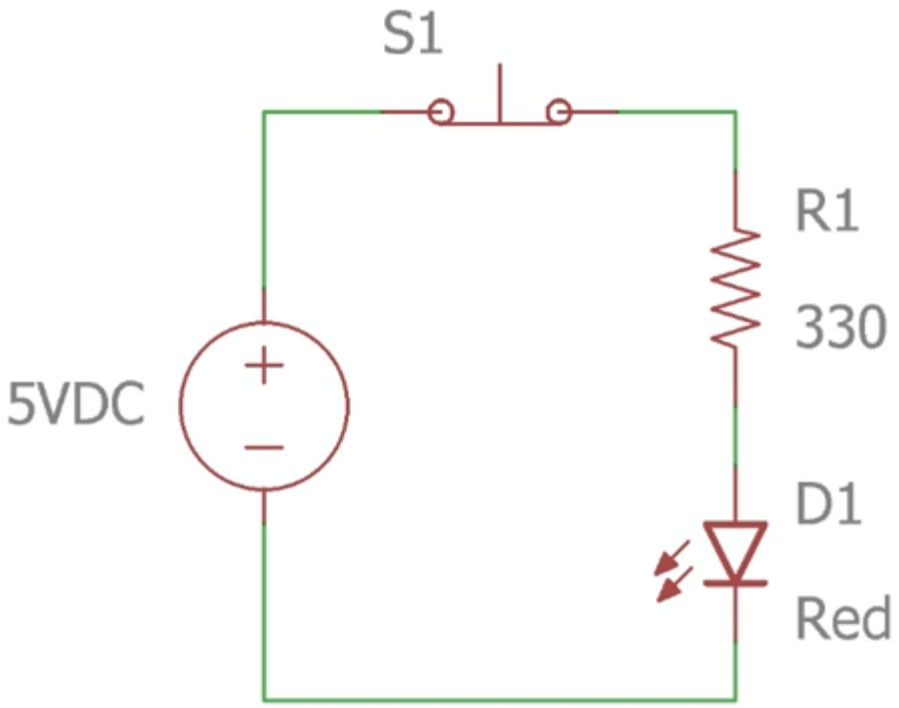




Pushbutton Switch



Normally Open (NO)



Normally Closed (NC)

Input Pins

- Input pins are controlled by other components
- Arduino reads the voltage on the pins
- Allows it to respond to events and data
- pinMode(pin , INPUT) sets a pin to INPUT
  - pin is the number of the pin
    - 0-13 for the digital pins (Digital signal only)
    - A0 –A5 for the analogue pins (Digital and analogue signals)

Digital Input

int digitalRead(pin)

- Returns the state of an input pin
- Returns either LOW (0 volts) or HIGH (5 volts)

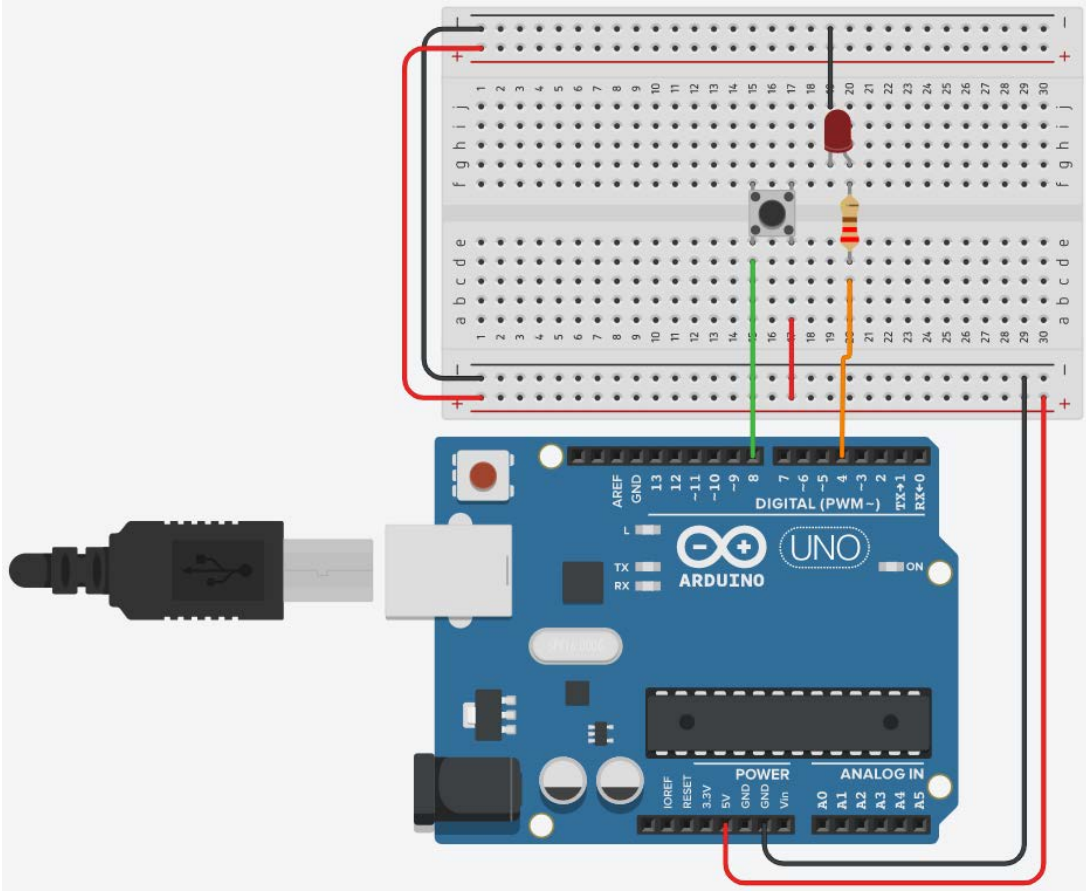
- Example:
  - int pin_val;
  - pin_val= digitalRead(8);

- Pin_val is set to the state of digital pin 8

Pushbutton Switch

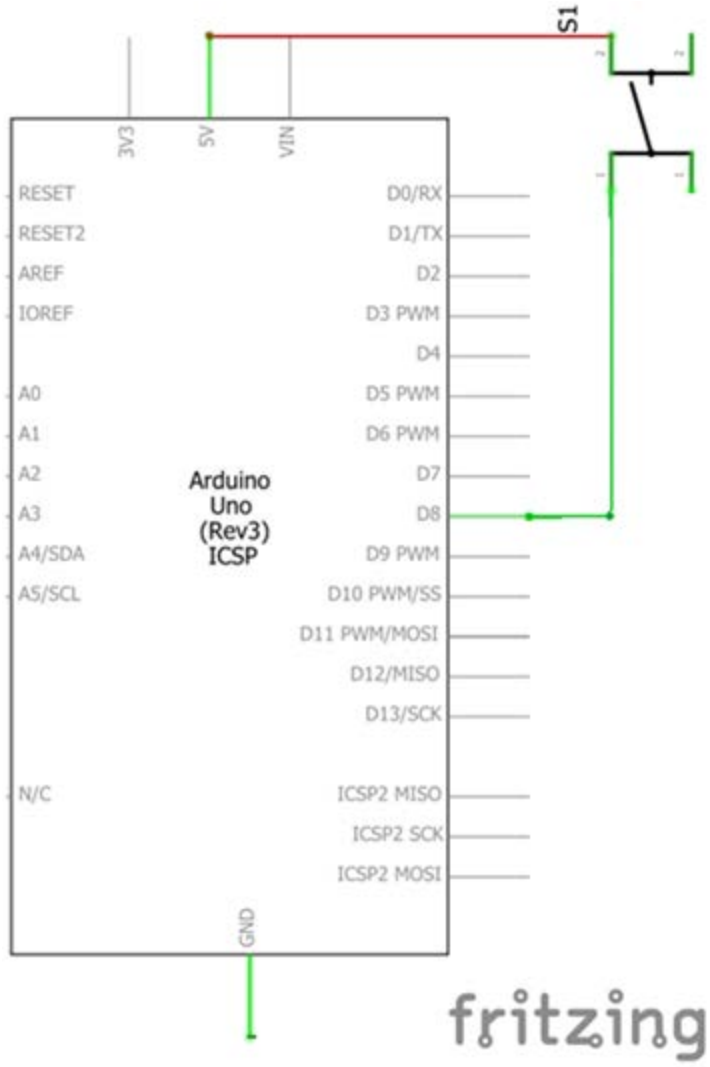
```
const int switchPin = 8;
const int LEDpin = 4;
int switchPressed = 0;
void setup() {
    pinMode(LEDpin,OUTPUT);
    pinMode(switchPin,INPUT);
}

void loop() {
    // Get the input
    switchPressed = digitalRead(switchPin);
    //Change the output
    if (switchPressed == HIGH ) {
       digitalWrite(LEDpin, HIGH);
    }
    else if (switchPressed == LOW ) {
       digitalWrite(LEDpin, LOW);
    }
}
```

- Make sure to use ALL CAPS for ***INPUT*** or ***OUTPUT***
- Digital Input values are only **HIGH** (On) or **LOW** (Off)



Pushbutton Switch connected to Arduino

- When the switch is closed, it creates a direct connection to V_{CC}, but when the switch is open, the rest of the circuit would be left floating.

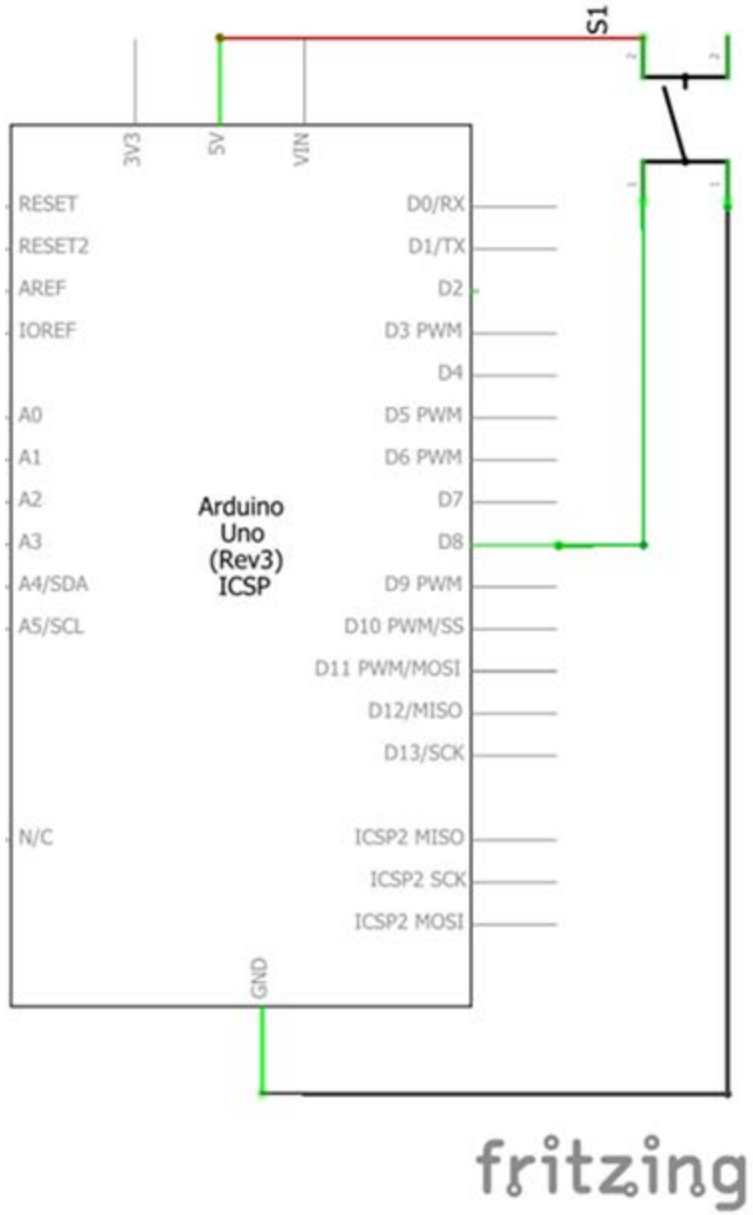


"Floating" pin

Voltage when switch open = ?

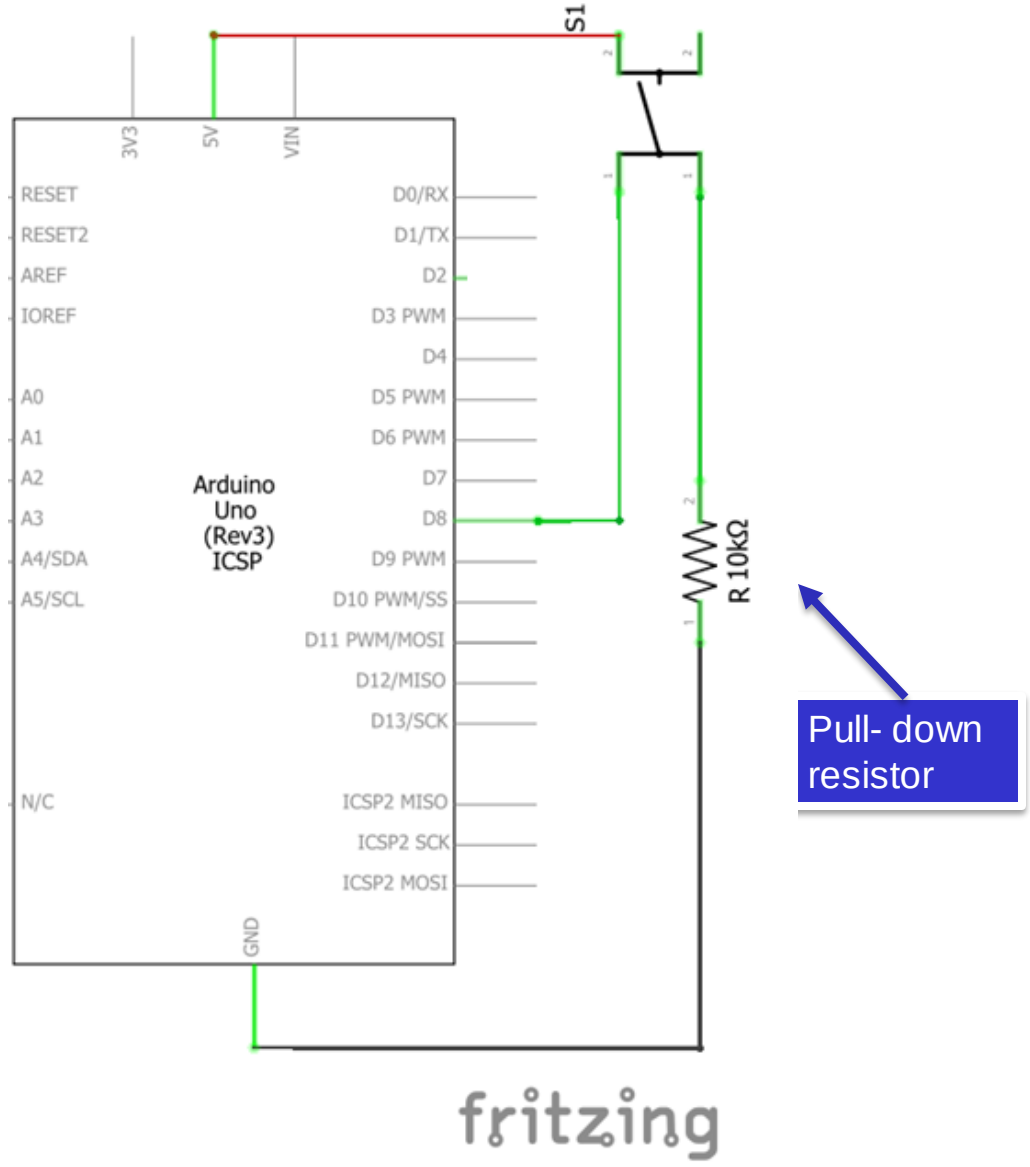
Connecting the floating Pin to Ground: Short Circuit

- It is possible to connect the switch directly to the ground for avoiding floating pin
- However, when the circuit is closed, it makes a short circuit
  - It might damage our power supply and make components very hot!



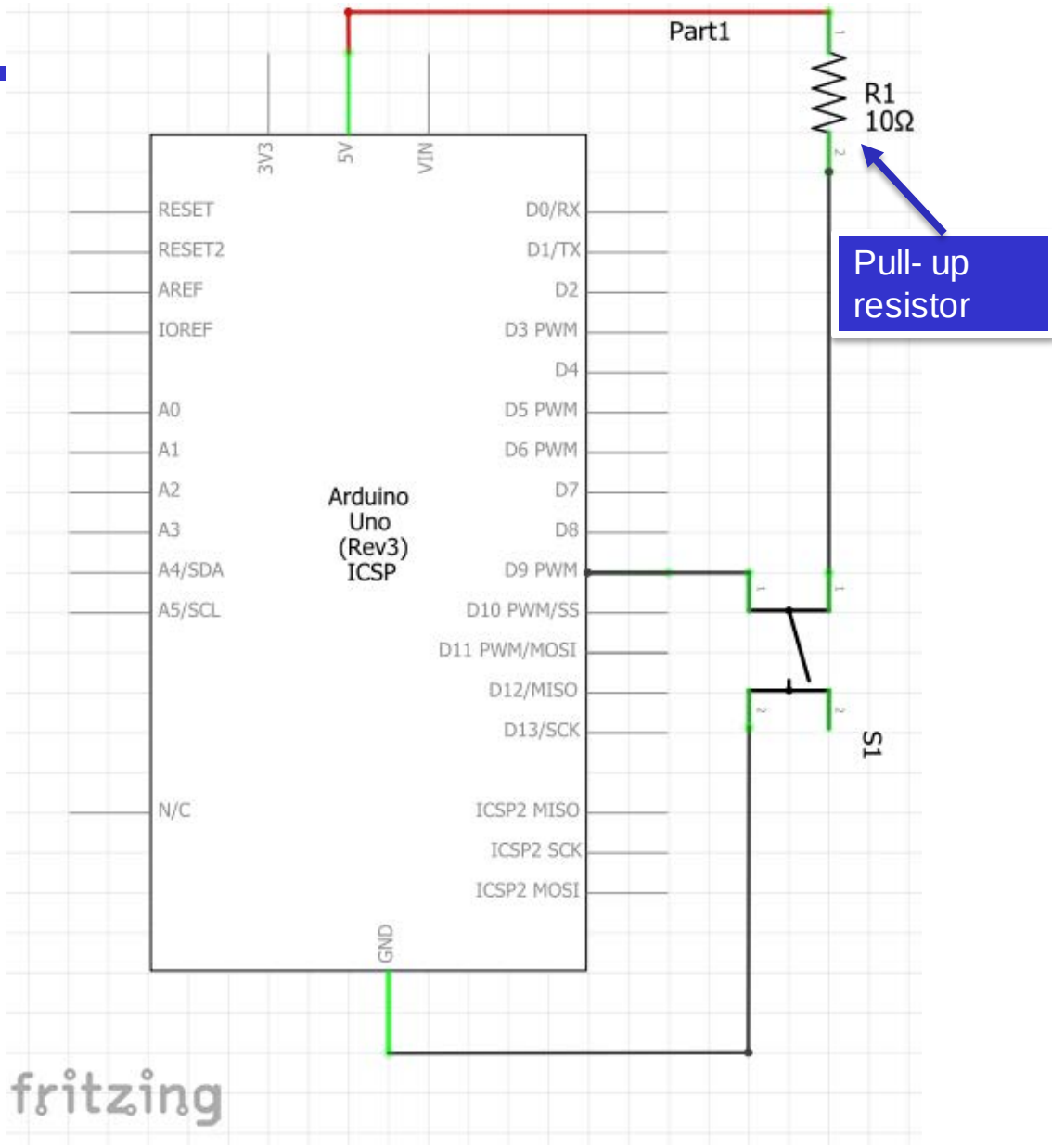
Pull-Down Resistors

- Putting a resistor between switch and ground
- For a switch that connects to V_{CC}, a pull-down resistor ensures a well-defined ground voltage (i.e. logical low) when the switch is open.
- Pull-down resistor pulls the voltage level down to zero volt when the switch is open
  - When you press the switch, the Arduino receives logical 1 or HIGH or 5 volts



Pull-Up Resistors

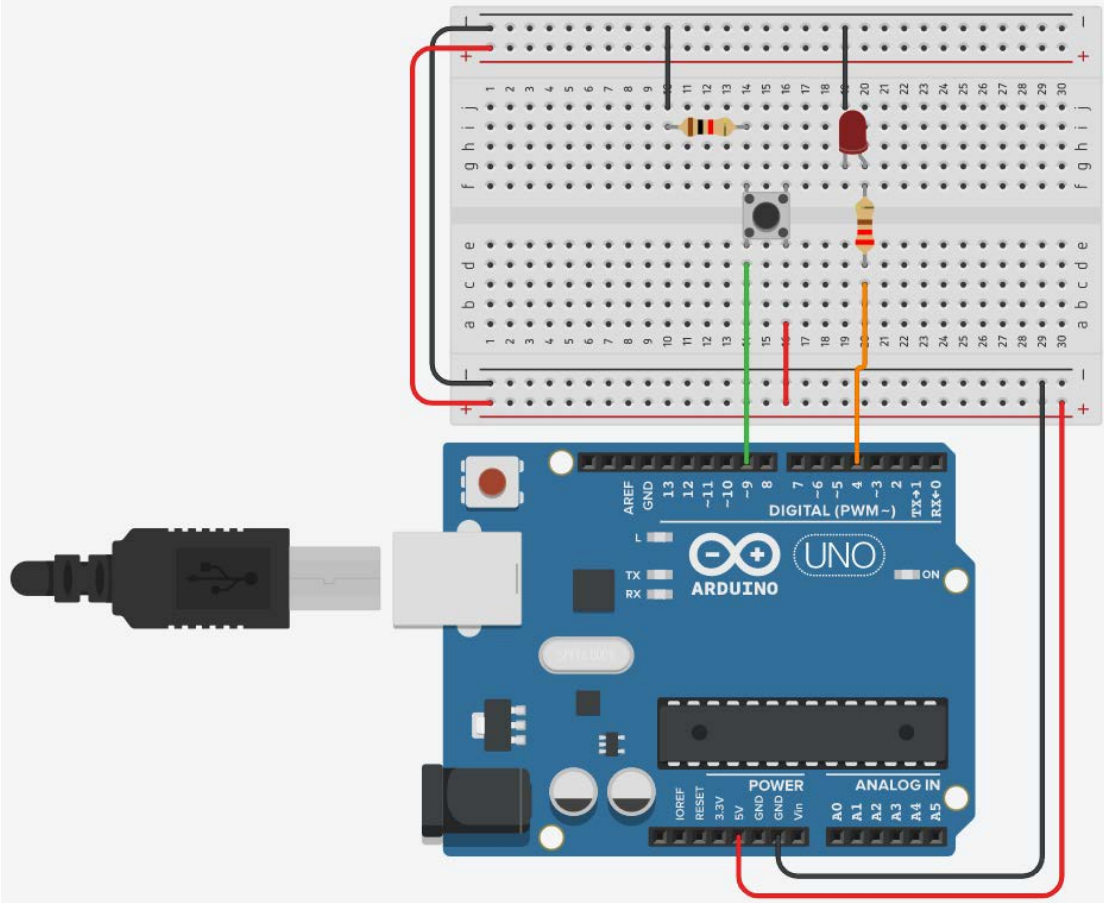
- Putting a resistor between switch and V_{CC}
- For a switch that connects to ground, a pull-up resistor ensures a well-defined voltage (i.e. V_{CC}, or logical high) across the remainder of the circuit when the switch is open.
- Pull-up resistor pulls the voltage level up to 5 volt when the switch is open.
  - When you press the switch, the Arduino receives logical 0 or LOW or 0 volt



Pushbutton Switch

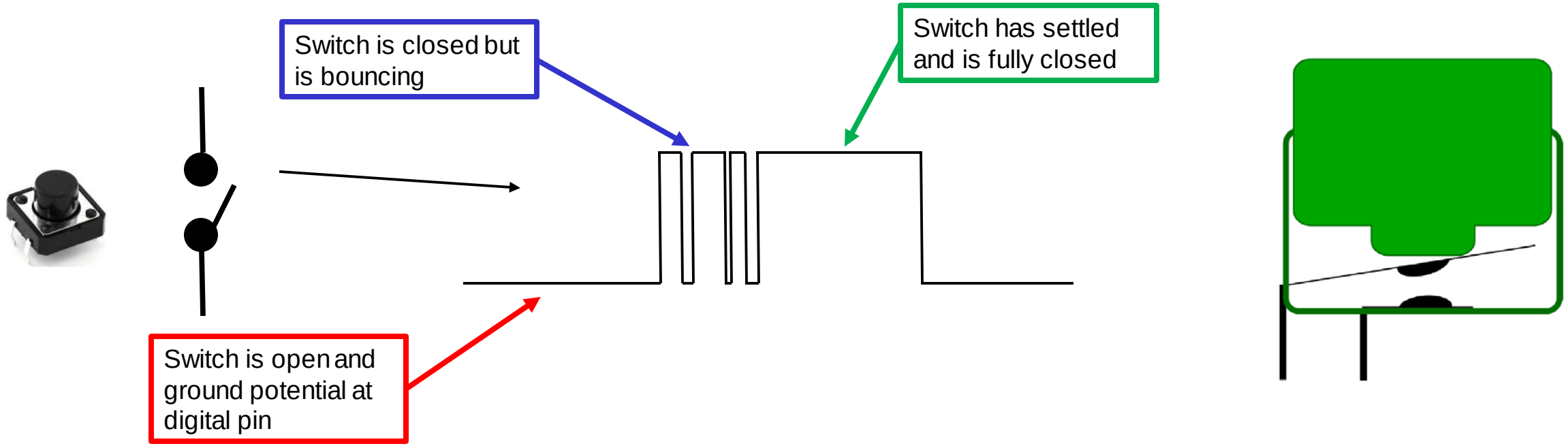
```
const int switchPin = 8;
const int LEDpin = 4;
int switchPressed = 0;
void setup() {
    pinMode(LEDpin,OUTPUT);
    pinMode(switchPin,INPUT);
}

void loop() {
    // Get the input
    switchPressed = digitalRead(switchPin);
    // Change the output
    if (switchPressed == HIGH ) {
        digitalWrite(LEDpin, HIGH);
    }
    else if (switchPressed == LOW ) {
        digitalWrite(LEDpin, LOW);
    }
}
```

- Make sure to use ALL CAPS for ***INPUT*** or ***OUTPUT***
- Digital Input values are only **HIGH** (On) or **LOW** (Off



Debouncing a Switch

- In the previous sketch, when you press the switch, sometimes the LED does not turn on or off.
- When a switch is pressed, the springy nature of the metal used in the contact points can cause the contact points to touch several times.
- This behaviour is called bouncing.
- The switch can be debounced by using a timer.



Debouncing a Switch

```
void loop() {
    switchPressed = digitalRead(switchPin);
    if(switchPressed == HIGH){
        delay(50);
        switchPressed = digitalRead(switchPin);
        if(switchPressed == HIGH){

            ...

        }
    }
}
```

Read switch pin

Debounce time of 50ms

Read switch pin again

Compare switch state again

The signal is stable and remember the switch state