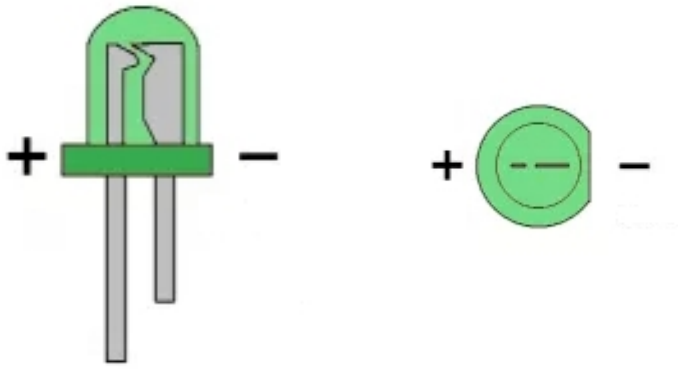
Diode is a device that allow Electric Current to flow in one direction.

LED is similar to a regular diode, except that it can glow. Because of this glowing property, LED is often an Output Device (apart from its function as a diode - to block electricity to flow at the wrong direction)

The LED is made from some chemical material packaged inside a clear plastic package. When Electric Current flow into those chemical, they will “glow”. It can glow in different colors depends on type chemical material used. Sometimes different plastic package color is used to get color effects.

LED comes with many shape and sizes, all of them are similar they have Negative and Positive terminals.



Because the LED only allows Electric Current to flow at one direction, the terminals must be correctly connected to the Power Supply.

- Positive Terminal from LED must be connected to **Positive(or VCC) Terminal of Power Supply**

- Negative Terminal from LED must be connected to **Negative(or GND) Terminal of Power Supply**

The picture above is the “bulb” type LED, for LED in other shapes, we need to find out their Positive and Negative terminal before connecting it to the Power Supply terminals.

The LED requires a minimum Voltage before it can operate. If the LED receives insufficient Voltage, the LED will not glow.

It also have a maximum limit for Electric Current. Do not exceed the limit, otherwise we will risk the LED being damaged.

The LED often have very small maximum Current allowed, most of them are less than 30mA (mili-Ampere). Our Power Supply will normally have more than 30mA flowing, therefore we normally **add Resistor** to the LED to reduce the Current at the LED, **protecting the LED from being damaged.**

LED Resistor Value Calculation

RESISTOR FOR regular 5mm "bulb" LED used with Arduino Uno

220 Ohm Resistor is often used together a regular 5mm "bulb" LED in a 5V electrical circuit (example, the Arduino Uno board). This is important because a regular 5mm "bulb" LED will be damaged if used without a Resistor in this setup.

Why 220 Ohm Resistor ?

We have following information from our circuit and devices:
1.Total Supply Voltage = 5V (Arduino Uno board)

2.Regular 5mm "bulb" LED (information from manufacturer, datasheet)
2.1.LED Voltage Drop = 1.8V
2.2.Electric Current Limit = 20mA (or 0.02A)

These two (3. and 4.), we need to find out on our own
3.Resistor Voltage Drop (RVD) = ?
4.Resistor Value (RV) to get 20mA = ?

3.Resistor Voltage Drop (RVD) = ?

Kirchhoff's Voltage Law,
Total Supply Voltage = Total Voltage Drop

5V(Arduino Uno) = 1.8V(LED) + RVD(Resistor)

RVD = 5V-1.8V = 3.2V (Voltage at our Resistor)

4.Resistor Value (RV)to get 20mA(0.02A) = ?

20mA is not available yet, we want to achieve the 20mA at the LED with our Resistor because we know that is the limit (data provided by the manufacturer of LED). Now we want to find out what Resistor can help to achieve that.

Ohms Law,
Resistance(RV) = Voltage / Current

RV = 3.2V (from our **RVD**) / 0.02A
RV = 160 Ohm

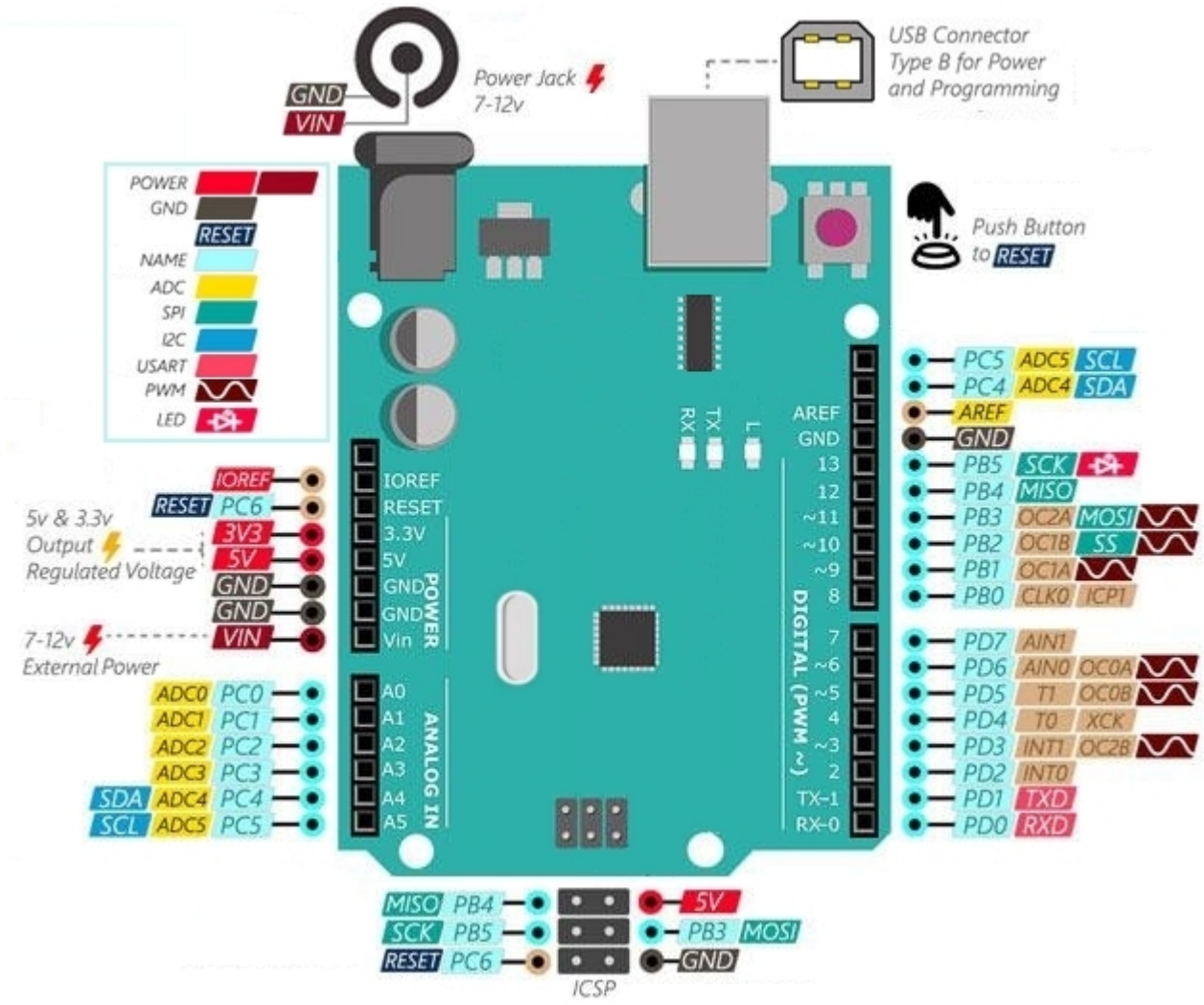
The ideal Resistor to be used with the regular 5mm LED (based on our specs above) on a 5V circuit will be the 160 Ohm Resistor because it allows the exact 20mA Electric Current into the LED.

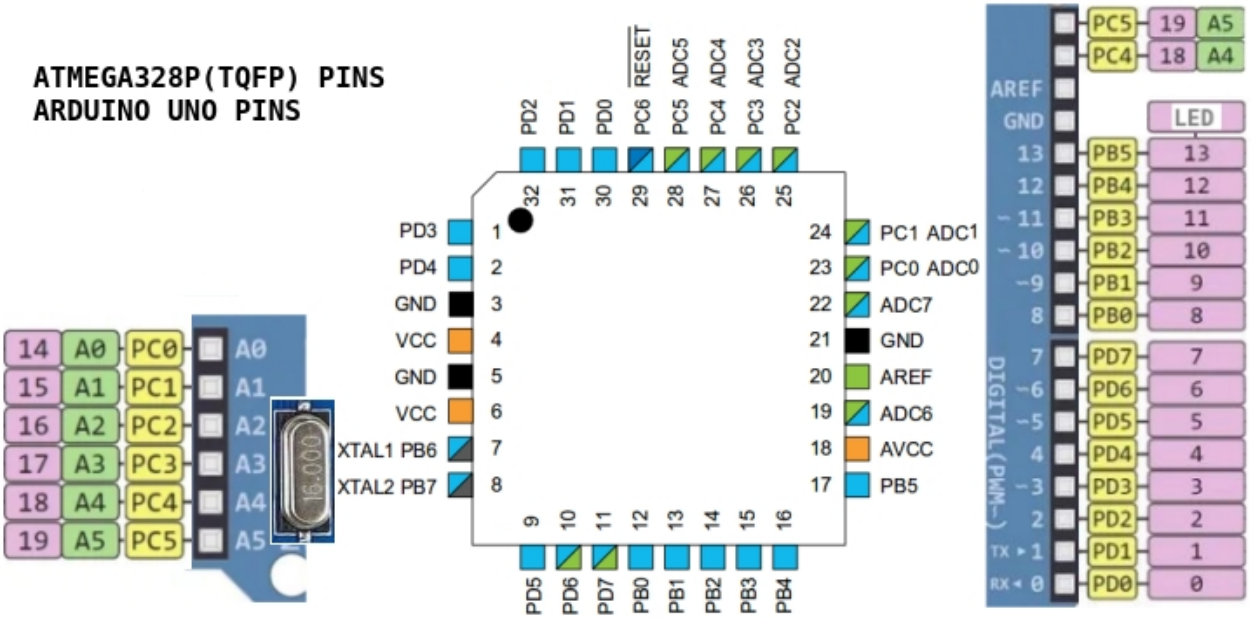
Anything smaller than 160 Ohm Resistor will let more than 20mA Electric current flowing into the LED, we risk destroying the LED.

Sometimes because of variations/defects in manufacturing process, the LED will still survive when used with Resistors lower than 160 Ohm, sometimes the LED gets damaged with 160 Ohm Resistor.

So, we just use something safer and common, "220 Ohm Resistor". We can also use Resistor with bigger resistance value safely with our LED, for example, the "1,000 Ohm or 10,000 Ohm resistor" (the LED will still work, just dimmer because the LED gets less Electric Current)

This Resistor Calculation example is for an LED, this same calculation can apply for other devices. LED is a very good tool for learning this.

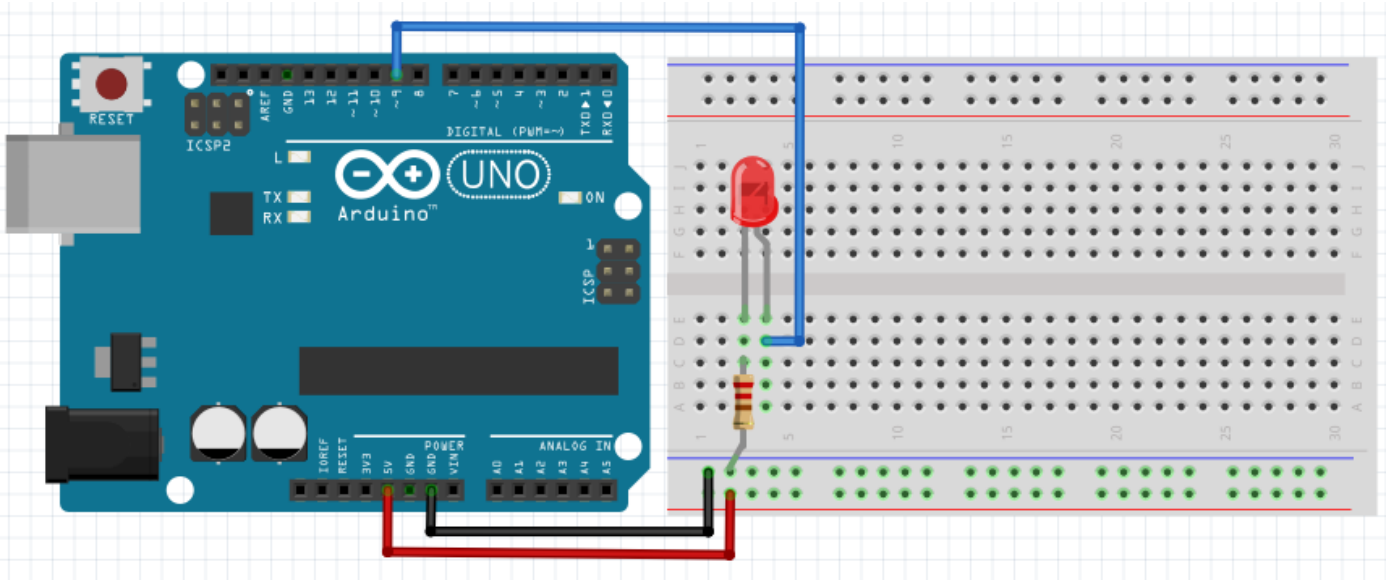

Power Jack
7-12v
GND
VIN
USB Connector
Type B for Power
and Programming
Push Button
to RESET
POWER
GND
RESET
NAME
ADC
SPI
I2C
USART
PWM
LED
IOREF
RESET
PC6
5v & 3.3v
Output
Regulated Voltage
3V3
5V
GND
GND
7-12v
External Power
VIN
ADC0 PC0
ADC1 PC1
ADC2 PC2
ADC3 PC3
SDA ADC4 PC4
SCL ADC5 PC5
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
PC5 ADC5 SCL
PC4 ADC4 SDA
AREF
GND
PB5 SCK
PB4 MISO
PB3 OC2A MOSI
PB2 OC1B SS
PB1 OC1A
PB0 CLK0 ICP1
PD7 AIN1
PD6 AIN0 OC0A
PD5 T1 OC0B
PD4 T0 XCK
PD3 INT1 OC2B
PD2 INT0
PD1 TXD
PD0 RXD
MISO PB4
SCK PB5
RESET PC6
5V
PB3 MOSI
GND
ICSP


ATMEGA328P(TQFP) PINS
ARDUINO UNO PINS
PD2 PD1 PD0 RESET PC6 ADC5 ADC4 ADC3 ADC2
32 31 30 29 28 27 26 25
PD3 1
PD4 2
GND 3
VCC 4
GND 5
VCC 6
XTAL1 PB6 7
XTAL2 PB7 8
24 PC1 ADC1
23 PC0 ADC0
22 ADC7
21 GND
20 AREF
19 ADC6
18 AVCC
17 PB5
9 10 11 12 13 14 15 16
PD5 PD6 PD7 PB0 PB1 PB2 PB3 PB4
14 A0 PC0
15 A1 PC1
16 A2 PC2
17 A3 PC3
18 A4 PC4
19 A5 PC5
16.000
PC5 19 A5
PC4 18 A4
LED
PB5 13
PB4 12
PB3 11
PB2 10
PB1 9
PB0 8
PD7 7
PD6 6
PD5 5
PD4 4
PD3 3
PD2 2
PD1 1
PD0 0

Parts Required

1 x Arduino UNO with USB 2.0 type A/B Data Cable connected to Computer with Arduino IDE Software installed
1 x Half-size Solderless Breadboard
2 x Jumper Wire (M-M)

1 x Light Emitting Diode 5mm (LED)
1 x Resistor (220 Ohm)
1 x Jumper Wire (M-M)



```
// program name = led_digital_pin
//
#define LED_PIN 9

void setup() {
  pinMode(LED_PIN,OUTPUT);
}

void loop() {
  digitalWrite(LED_PIN, HIGH);
  delay(500);
  digitalWrite(LED_PIN, LOW);
  delay(500);
}
```

```
Watch the LED
It goes ON, then OFF then repeats the process.
```

```
// program name = led_digital_pwm
//
#define LED_PIN 9

int pwmValue = 0; // value = 0 to 255
int pwmDir   = 1;

void setup() {
  pinMode(LED_PIN,OUTPUT);
  for (int i=0;i<5;i++) {
    digitalWrite(LED_PIN,HIGH);
    delay(500);
    digitalWrite(LED_PIN,LOW);
    delay(500);
  }
}

void loop() {
  switch (pwmValue) {
    case 0:
      delay(800);pwmDir = 1;
      break;
    case 255:
      delay(800);pwmDir = -1;
      break;
    default:break;
  }
  pwmValue = pwmValue+pwmDir;
  analogWrite(LED_PIN,pwmValue);
  if (pwmValue > 50) {
    delay(20);
  } else {
    delay(150);
  }
}
```

Watch the LED
It goes brighter then becomes dimmer then repeats the process.

PWM (Pulse wave Modulation) is a process where power supply is turn “ON” or “OFF” (alternating between the two state) with a particular pattern within a period of time.

Within the time,
PWM at 100% means, the pin is switched “ON”=100%
PWM at 50% means, the pin is alternating between “ON”=50% and “OFF”=50%
PWM at 25% means, the pin is alternating between “ON”=25% and “OFF”=75%
PWM at 75% means, the pin is alternating between “ON”=75% and “OFF”=25%
PWM at 0% means, the pin is switched “OFF”=100%

Arduino library function, analogWrite() peforms PWM (pulse wave modulation) effect on the LED.

When programming with the Arduino Library analogWrite() function,
when the PWM value = 255, means PWM at 100%
when the PWM value = 0, means PWM at 0%

the PWM values can be calculated, for example to get 50% PWM,
Means 255/2 = 127.5 about 127 (it has to be a whole number)

International Morse Code

1 Dash = 3 Dots
Space between parts of same letter = 1 Dot
Space between letters = 3 Dots
Space between words = 7 Dots


```
// program name = led_morse_SOS
//
#define LED_PIN          9
#define MC_DOT           300
#define MC_DASH          900  // DASH = DOT x 3
#define MC_PART_SPACE    300  // PART SPACE = DOT x 1
#define MC_LETTER_SPACE 900  // LETTER SPACE = DOT x 3
#define MC_WORD_SPACE    2100 // WORD SPACE = DOT x 7

void setup() {
 pinMode(LED_PIN,OUTPUT);
}

void loop() {
  mcLetter_S();digitalWrite(LED_PIN,LOW);delay(MC_LETTER_SPACE);
  mcLetter_O();digitalWrite(LED_PIN,LOW);delay(MC_LETTER_SPACE);
  mcLetter_S();digitalWrite(LED_PIN,LOW);delay(MC_WORD_SPACE);
}

void mcLetter_S() {
  digitalWrite(LED_PIN,HIGH); delay(MC_DOT);
  digitalWrite(LED_PIN,LOW);  delay(MC_PART_SPACE);
  digitalWrite(LED_PIN,HIGH); delay(MC_DOT);
  digitalWrite(LED_PIN,LOW);  delay(MC_PART_SPACE);
  digitalWrite(LED_PIN,HIGH); delay(MC_DOT);
}

void mcLetter_O() {
  digitalWrite(LED_PIN,HIGH); delay(MC_DASH);
  digitalWrite(LED_PIN,LOW);  delay(MC_PART_SPACE);
  digitalWrite(LED_PIN,HIGH); delay(MC_DASH);
  digitalWrite(LED_PIN,LOW);  delay(MC_PART_SPACE);
  digitalWrite(LED_PIN,HIGH); delay(MC_DASH);
}
```

Watch the LED sending out S-O-S in Morse Code