Engineering Technology (ENGR 101)

Arduino and light sensors, Ultrasonic Distance Sensors and dimming LEDs

Mohammad Nekooei

Engineering and Computer Science

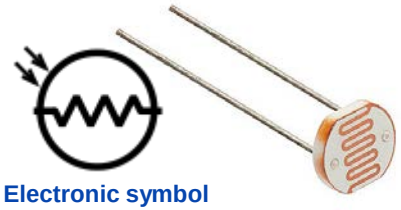
Victoria University of Wellington

Admin

- Lab 7
  - Due date is June 8, 19:00 (Xiamen Time)
  - This lab is in groups.
  - There is a correction

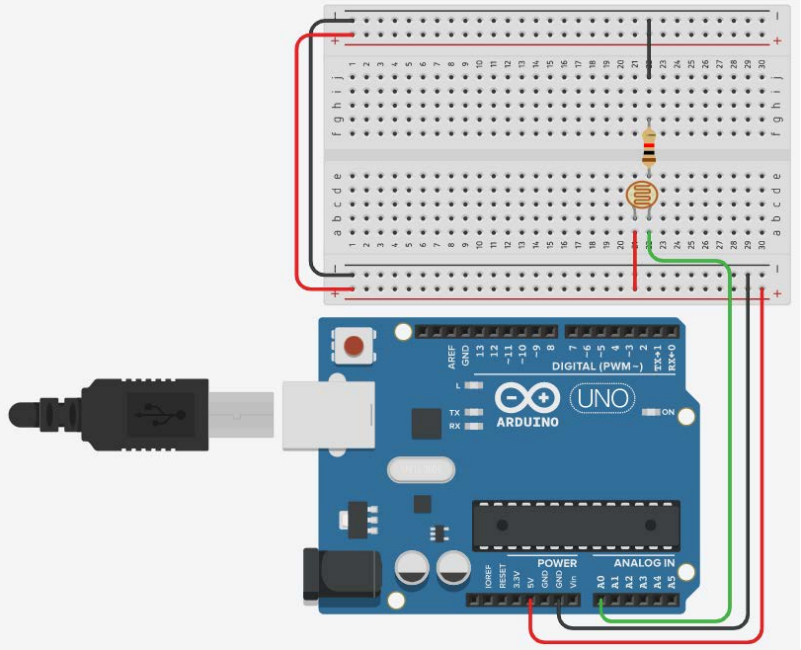
- Students who have not submitted their lab projects
  - Assignments 12% of final grade
  - Labs & project 38% of final grade
  - https://ecs.wgtn.ac.nz/Courses/XMUT101_2021T1/XMUT101CourseOutline

Photoresistor

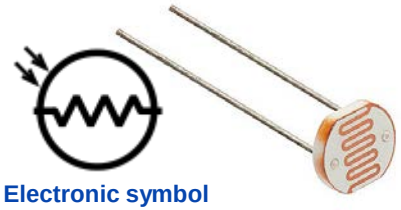


Electronic symbol

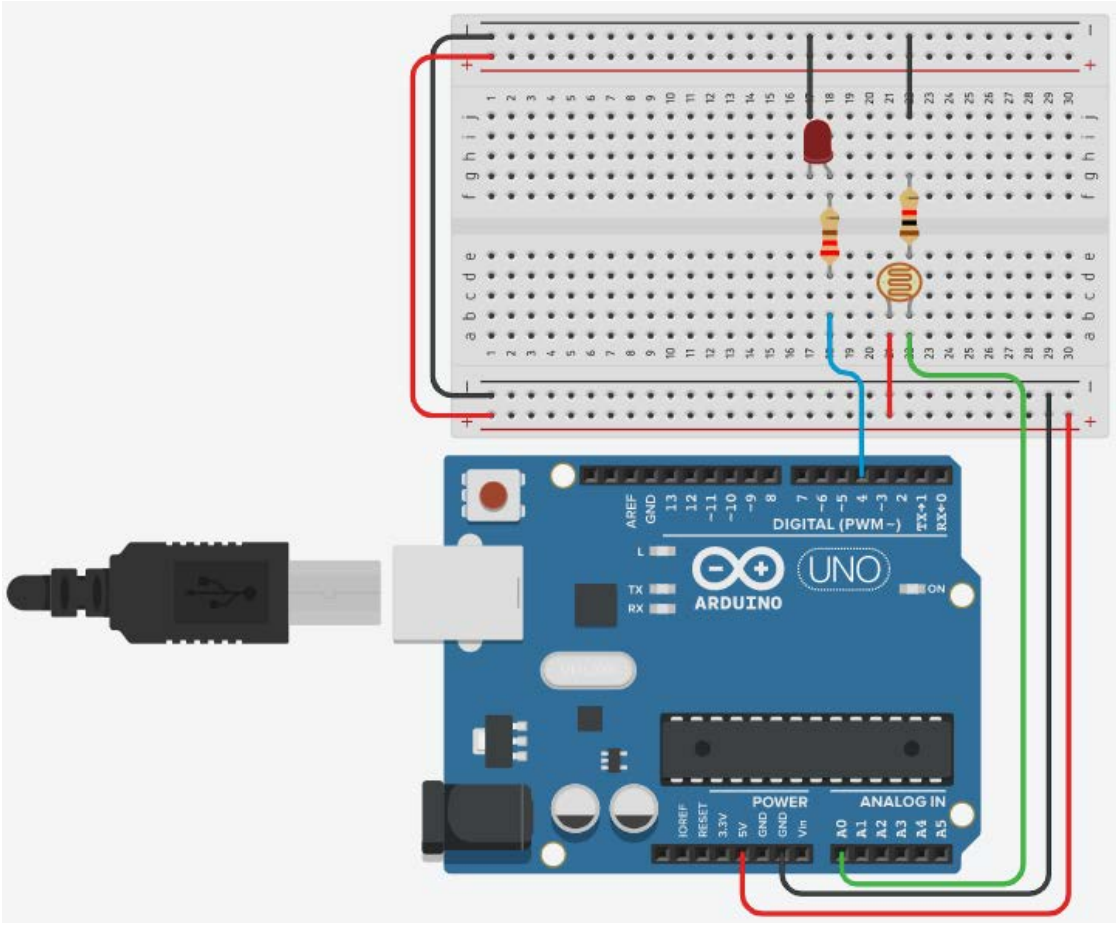
```
void loop() {
    int light_val;
    light_val = analogRead(light_pin);
    Serial.print(light_val);

    // We'll have a few threshholds, qualitatively determined
    if (light_val < 10) {
        Serial.println(" - Dark");
    } else if (light_val < 200) {
        Serial.println(" - Dim");
    } else if (light_val < 500) {
        Serial.println(" - Light");
    } else if (light_val < 800) {
        Serial.println(" - Bright");
    } else {
        Serial.println(" - Very bright");
    }

    delay(1000);
}
```



Photoresistor connected to LED

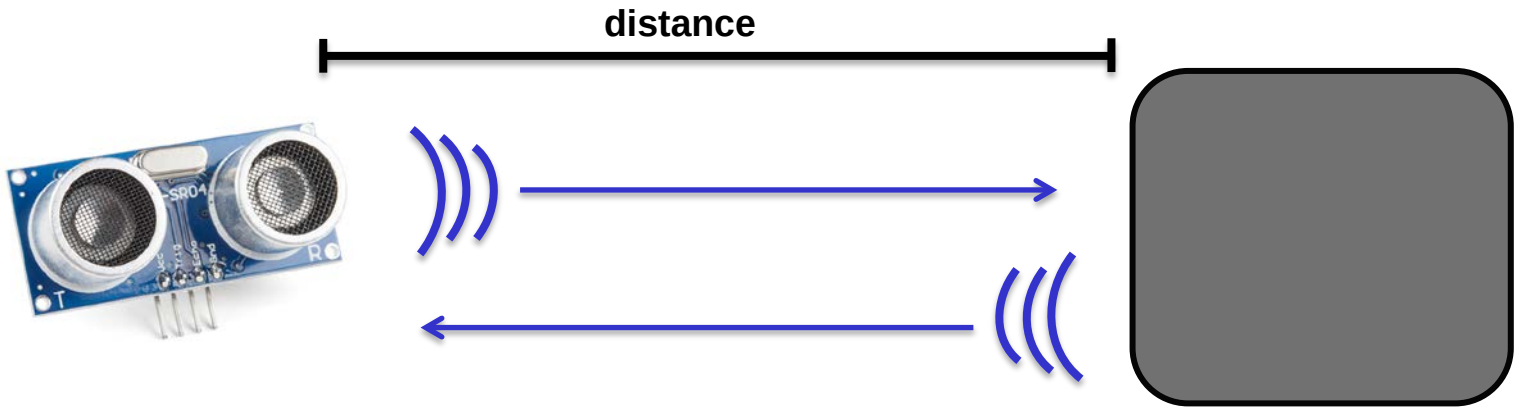

Electronic symbol

```
const int light_pin = A0;
const int led_pin = 4;
// Set darkness threshold
const int threshold = 500;
void setup() {
    Serial.begin(9600);
    pinMode(light_pin, INPUT);
    pinMode(led_pin, OUTPUT);
}
void loop() {
    int light_val;
    light_val = analogRead(light_pin);

    if (light_val < threshold) {
        digitalWrite(led_pin, HIGH);
    }
    else {
        digitalWrite(led_pin, LOW);
    }
    delay(1000);
}
```



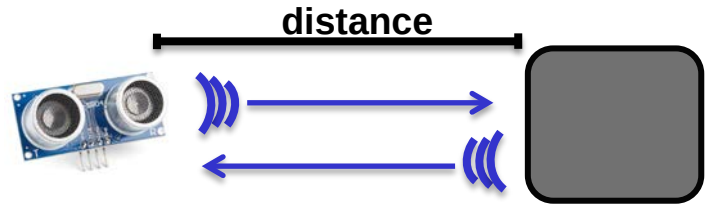
Ultrasonic Distance Sensor

- The ultrasonic sensor works by sending out an ultra high frequency sound pulse.
  - When the sound pulse hits an object, the distance sensor reports the time it takes from sending the pulse and receiving it.
- It works well for medium range applications.
- Most of ultrasonic distance sensors can use a digital electrical pulse to calculate the distance of an object



Timing Diagram

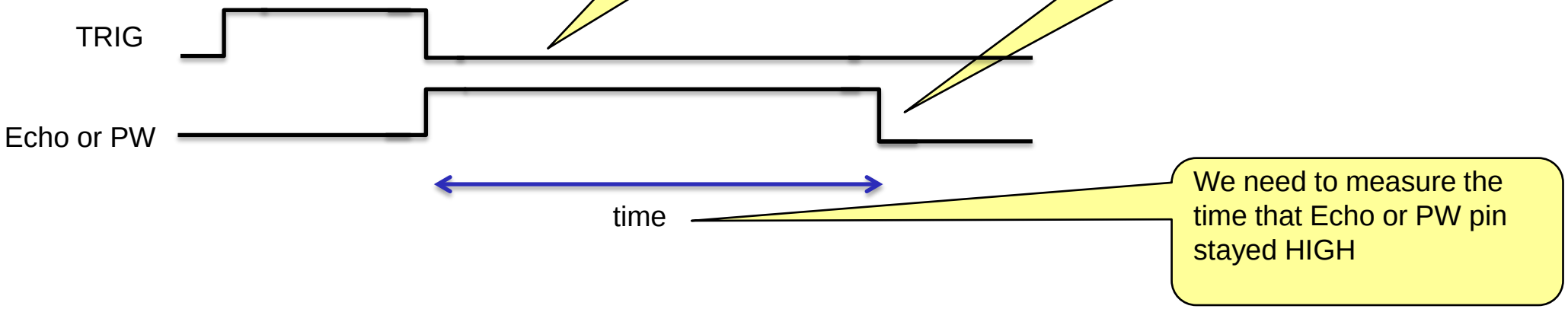
- There are two pins on the sensor
  - Trigger pin: We can control it to tell the sensor it should send out a sound pulse.
  - Echo or pulse width (PW) pin: we measure the electrical pulse width on the PW pin to determine the distance to the object.



1) Set the TRIG to HIGH for specific amount of time, which depends on the sensor (3µs -100µs)

2) Set the TRIG to LOW. The sensor sends out a pulse and sets the Echo or PW pin to HIGH.

3) When the sensor detects the reflected pulse, it sets the Echo or PW pin to Low.



We need to measure the time that Echo or PW pin stayed HIGH

Calculation of distance

$$speed\ of\ sound: 340\ m/s$$

$$speed\ of\ sound: \quad \frac{1s}{340m} \times \frac{1m}{100cm} \times \frac{1000000\mu s}{1s} = 29.4\ \mu s/cm$$

The time that Echo or PW pin stayed HIGH, it counts for the travel time to and from the object.

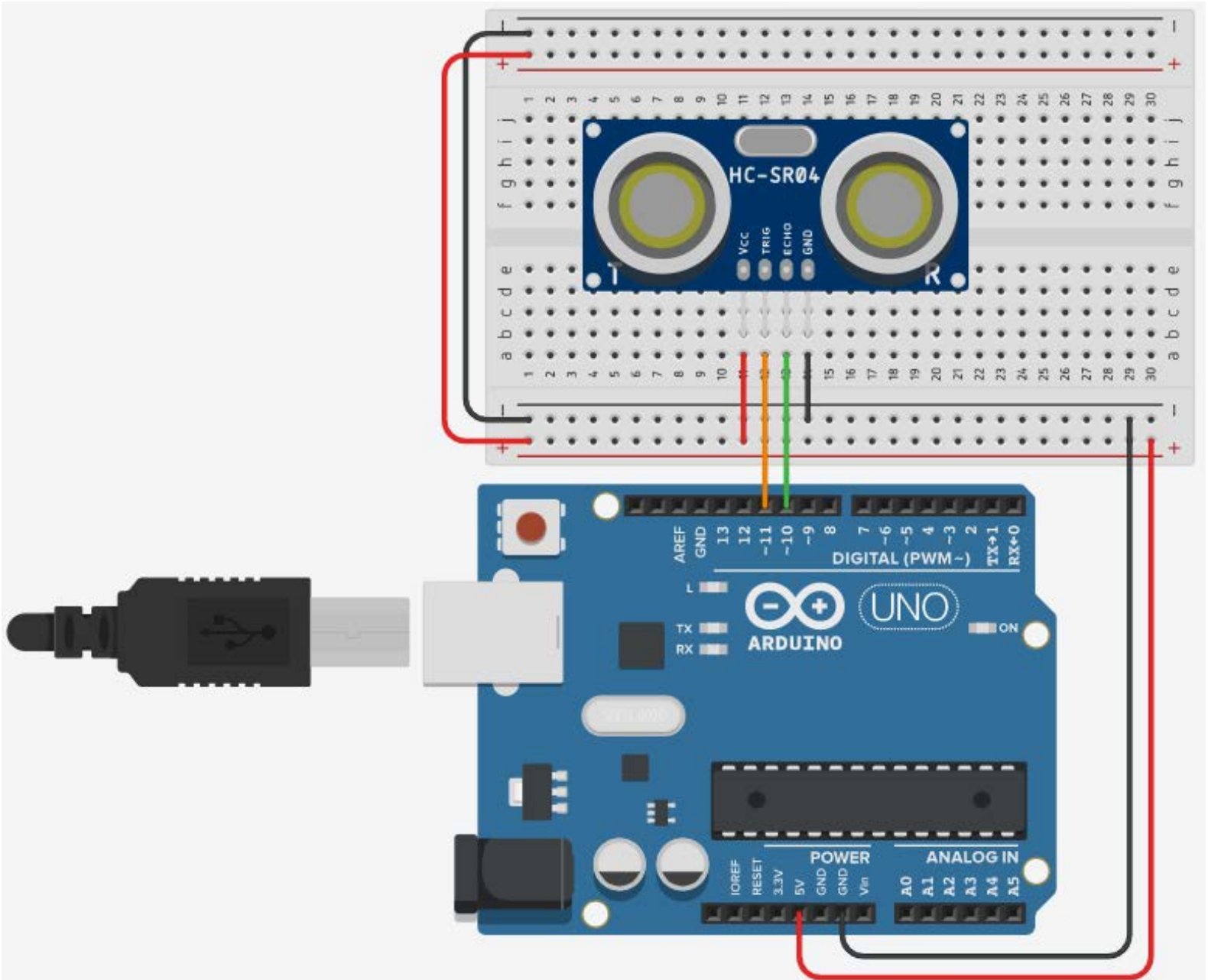
$$\frac{time}{29.4\mu s/cm} = 2 \times distance$$

$$distance = \frac{time}{2 \times 29.4\mu s/cm} = \frac{time}{58.8\ \mu s/cm}$$

We can compute the distance by dividing the time PW or echo pin held HIGH in microseconds by 58.8

Ultrasonic Distance Sensor example

```
const int trig_pin = 11;
const int echo_pin = 10;
const int trig_delay = 25;
void setup() {
    Serial.begin(9600);
    pinMode(trig_pin, OUTPUT);
    pinMode(echo_pin, INPUT);
}
```



Ultrasonic Distance Sensor example

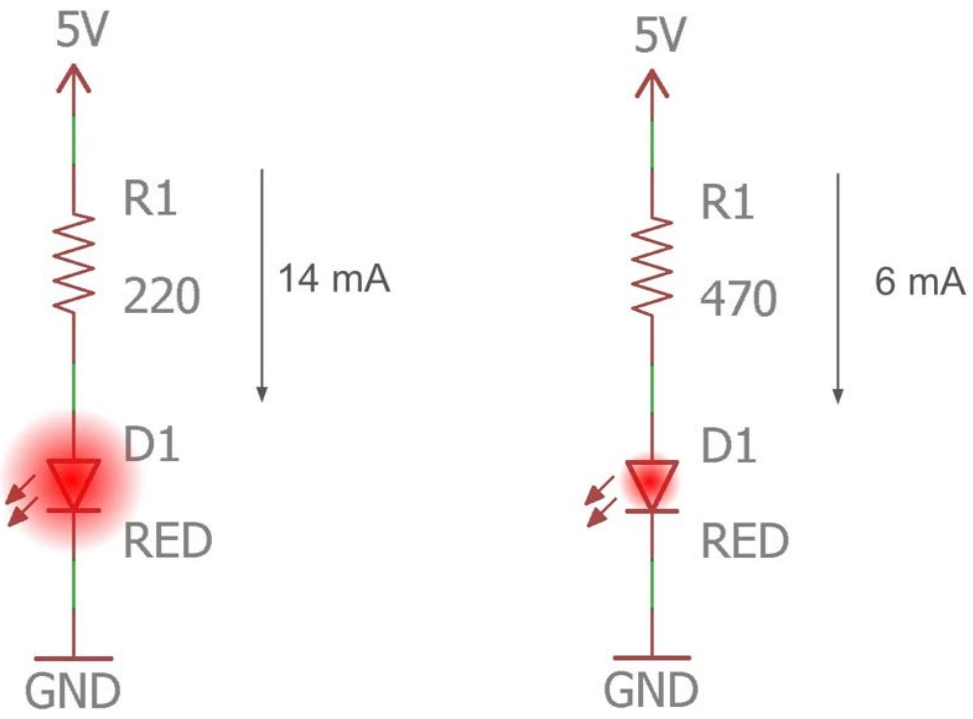
```
void loop() {
    long duration;
    float cm;
    // Tell distance to send out a pulse
    digitalWrite(trig_pin, LOW);
    delayMicroseconds(10);
    digitalWrite(trig_pin, HIGH);
    // Holds trig_pin high for necessary amount of time
    delayMicroseconds(trig_delay);
    digitalWrite(trig_pin, LOW);

    // Measure time of pulse on echo_pin pin
    duration = pulseIn(echo _pin, HIGH);

    // Convert time to dictance
    cm = duration / 58.8;
    Serial.print(cm);
    Serial.println(" cm");
    delay(1000);
}
```

We want to measure how long echo_pin is held high, so we set the second parameter of `pulseIn(…)` function to HIGH

- **`pulseIn(pin, value)`**
  - Reads a pulse (either HIGH or LOW) on a pin.
  - For example, if value is HIGH, **`pulseIn(…)`** waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing.
  - Returns the length of the pulse in microseconds or gives up and returns 0 if no complete pulse was received within the timeout.

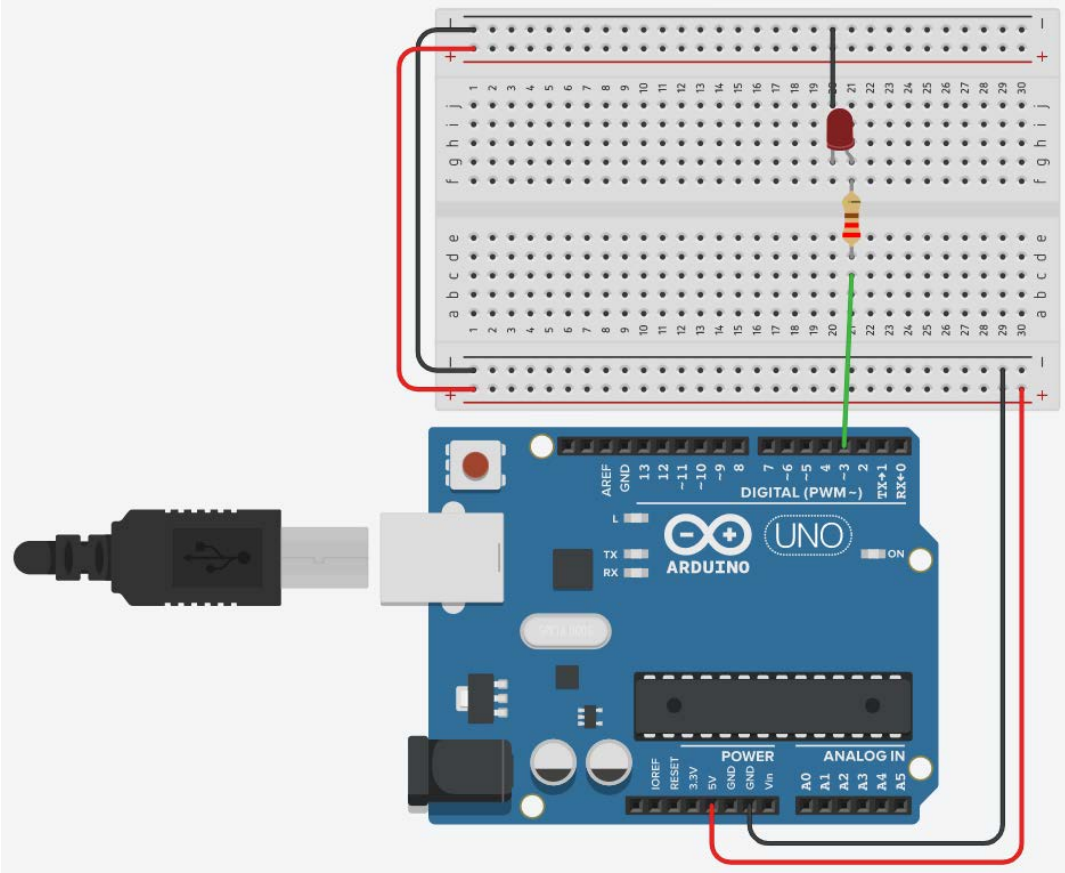
Dimming an LED

- Dimming an LED consists of changing the a mount of current flowing through it
- It needs complicated circuits
- There is an alternative way to fool our eyes of think an LED is dimmer without using an extra hardware



Dimming an LED

```
const int led_1_pin = 3;

void setup() {
    pinMode(led_1_pin,OUTPUT);
}
void loop() {
    digitalWrite(led_1_pin, HIGH);
    delay(500);
    digitalWrite(led_1_pin, LOW);
    delay(500);
}
```



- Blinking the LED every 500 milli seconds

Dimming an LED

```
void loop() {
   digitalWrite(led_1_pin, HIGH);
   delay(50);
   digitalWrite(led_1_pin, LOW);
   delay(50);
}
```

The LED should blink fast

```
void loop() {
   digitalWrite(led_1_pin, HIGH);
   delay(5);
   digitalWrite(led_1_pin, LOW);
   delay(5);
}
```

The LED blinks too fast that your eyes cannot see it. Then the LED appears simply on.

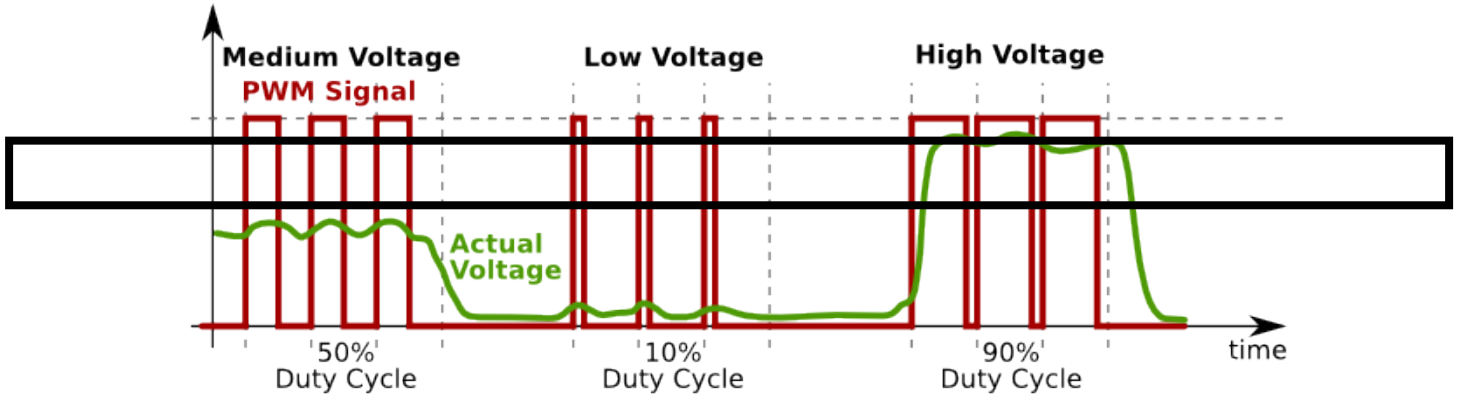
$$\frac{5ms\ (on)}{10ms\ (total\ time)} = 50\%\ duty\ cycle$$

```
void loop() {
   digitalWrite(led_1_pin, HIGH);
   delay(1);
   digitalWrite(led_1_pin, LOW);
   delay(9);
}
```

$$\frac{1ms\ (on)}{10ms\ (total\ time)} = 10\%\ duty\ cycle$$

Fading in and Fading Out (Analog or Digital?)

- Where the analogue input pins are designed to read analogue sensors (input), the Pulse-Width Modulation (PWM) pins are designed for output.
- PWM is a technique for obtaining analogue results with digital output.
- Since a digital output can be either on or off, to obtain the analogue output the digital output is switched between HIGH and LOW rapidly.
- The percentage of the time that the signal is high is called the **duty cycle**.



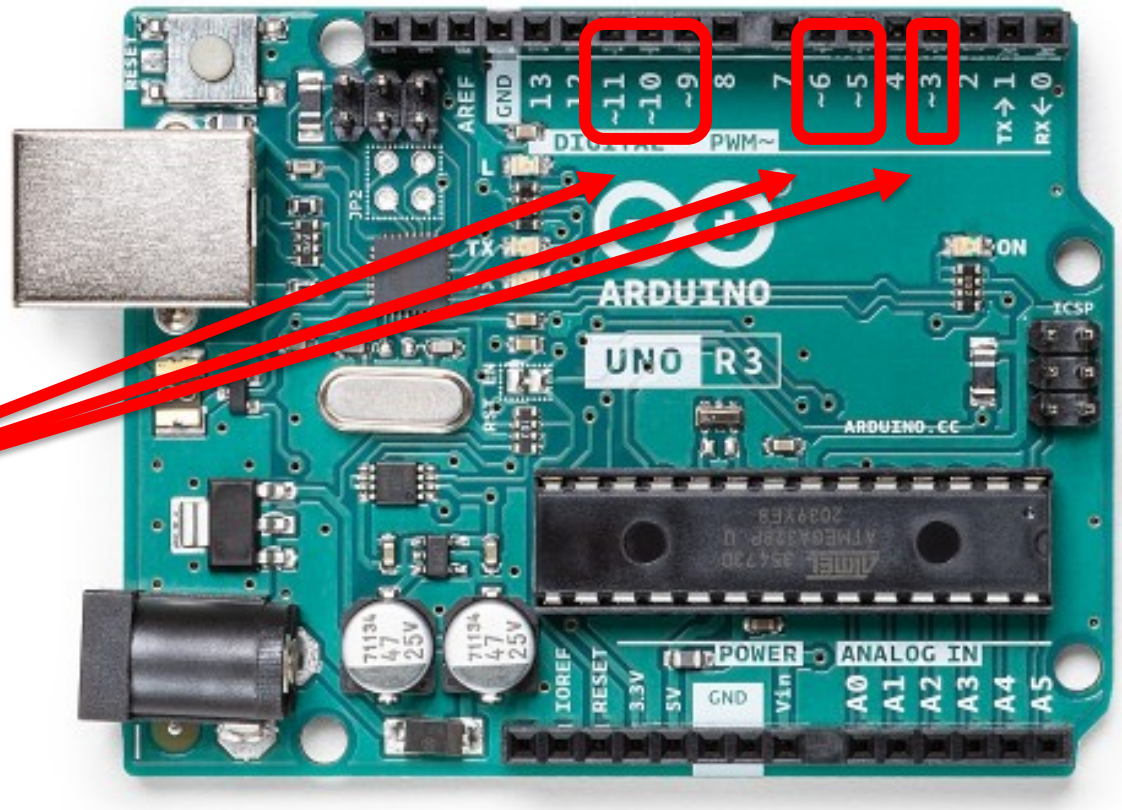
Analogue Output

int analogWrite(pin, value)

- Assigns the state of a Pulse-Width Modulation (PWM) pin labelled with the tilde (~)
- Assigns an integer from 0 to 255

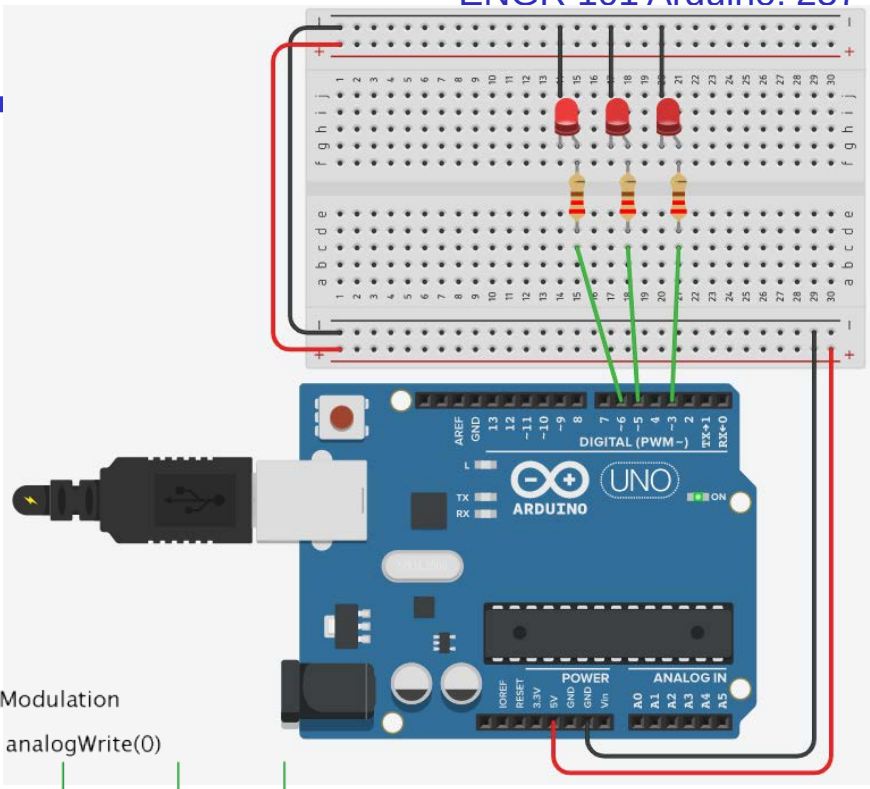
- Example:
  - `analogWrite(9, 255);`

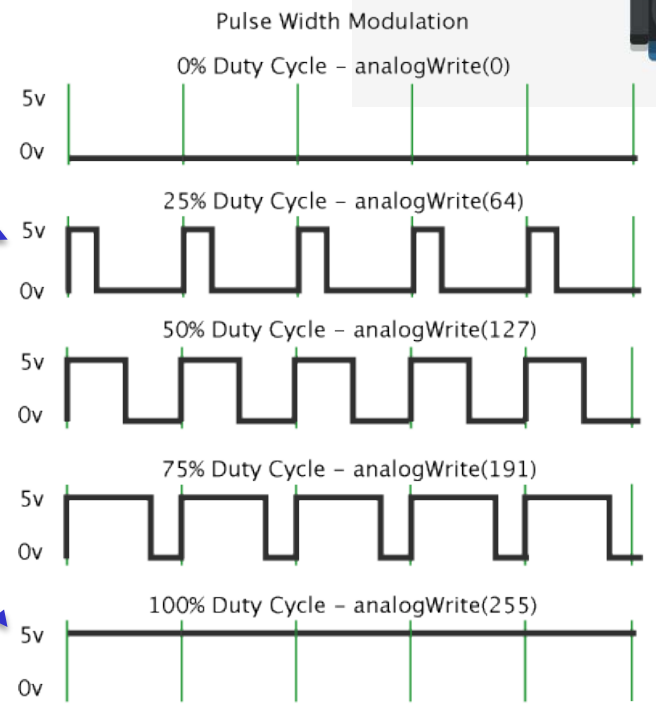
- pin must be a PWM pin (~)



Dimming three LEDs

```
const int led_1_pin = 3;
const int led_2_pin = 5;
const int led_3_pin = 6;
void setup()
{
  pinMode(led_1_pin, OUTPUT);
  pinMode(led_2_pin, OUTPUT);
  pinMode(led_3_pin, OUTPUT);

  analogWrite(led_1_pin, 64);
  analogWrite(led_2_pin, 127);
  analogWrite(led_3_pin, 255);
}
void loop() {
}
```





Fading LED

```
int LED_pw_pin = 5;
int brightness = 0;
int fadeAmount = 5;
void setup() {
   pinMode(LED,OUTPUT);
}

void loop() {
   analogWrite(LED_pw_pin, brightness);
   brightness += fadeAmount;
   if(brightness<= 0|| brightness>= 255){
      fadeAmount = fadeAmount * -1;
   }
   delay(30);
}
```