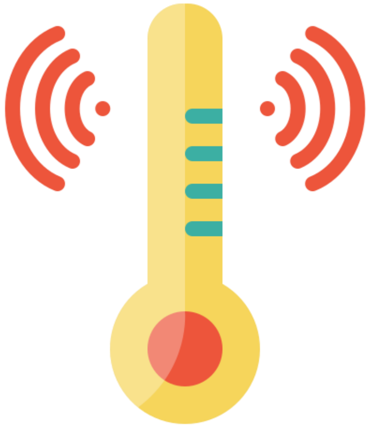
Embedded Systems

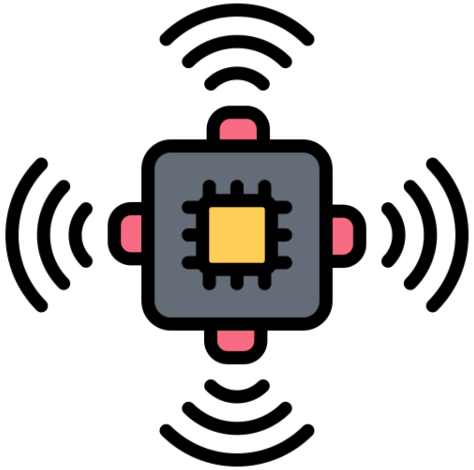
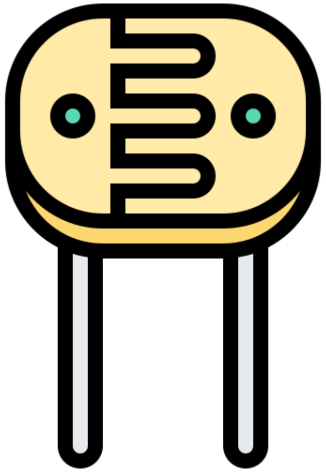
Radar System Prototype

Abdallah El Ghamry

Sensors

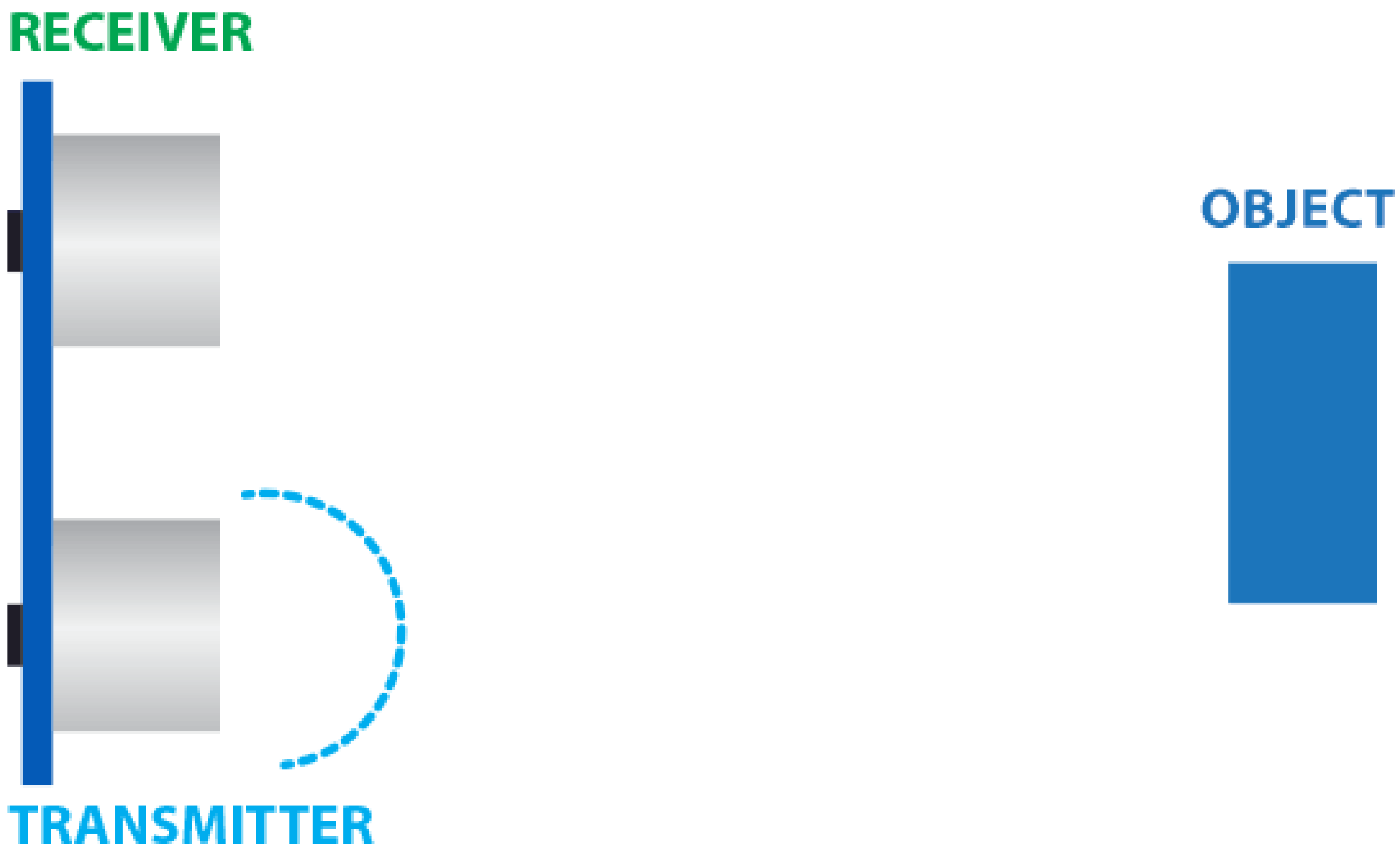
- A sensor is a device that detects some type of input from the physical environment.
- The input can be light, heat, motion, pressure or any number of other environmental phenomena.



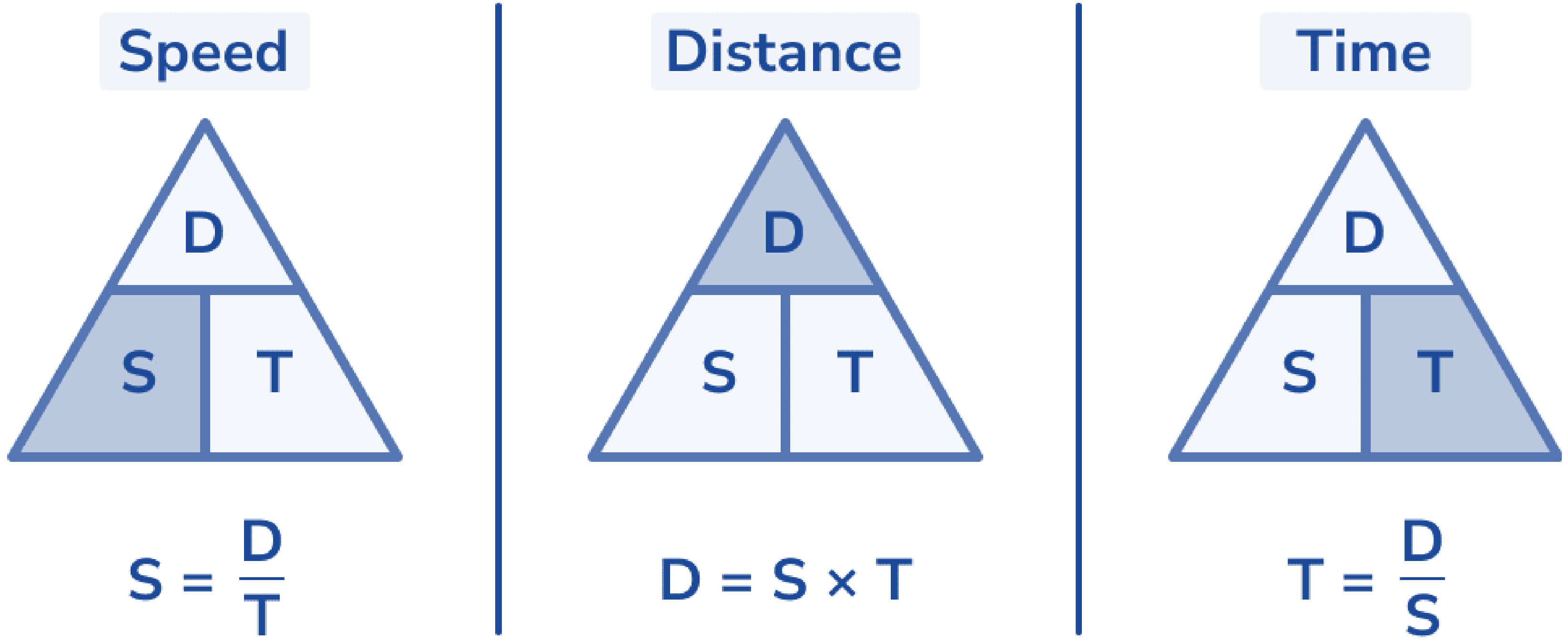
Ultrasonic Sensor

- As the name indicates, ultrasonic sensors measure distance by using ultrasonic waves.



Ultrasonic Sensor: Calculating the Distance

- The width of the received pulse is used to calculate the distance from the reflected object.
- This can be worked out using the simple distance-speed-time equation we learned in high school.

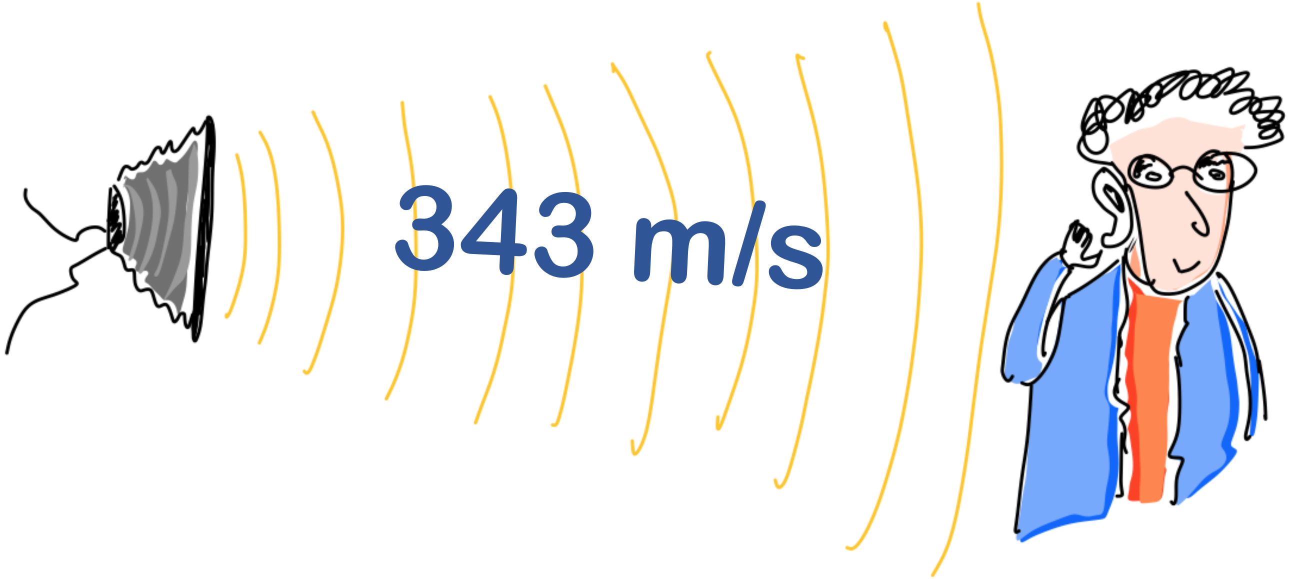


Ultrasonic Sensor: Calculating the Distance

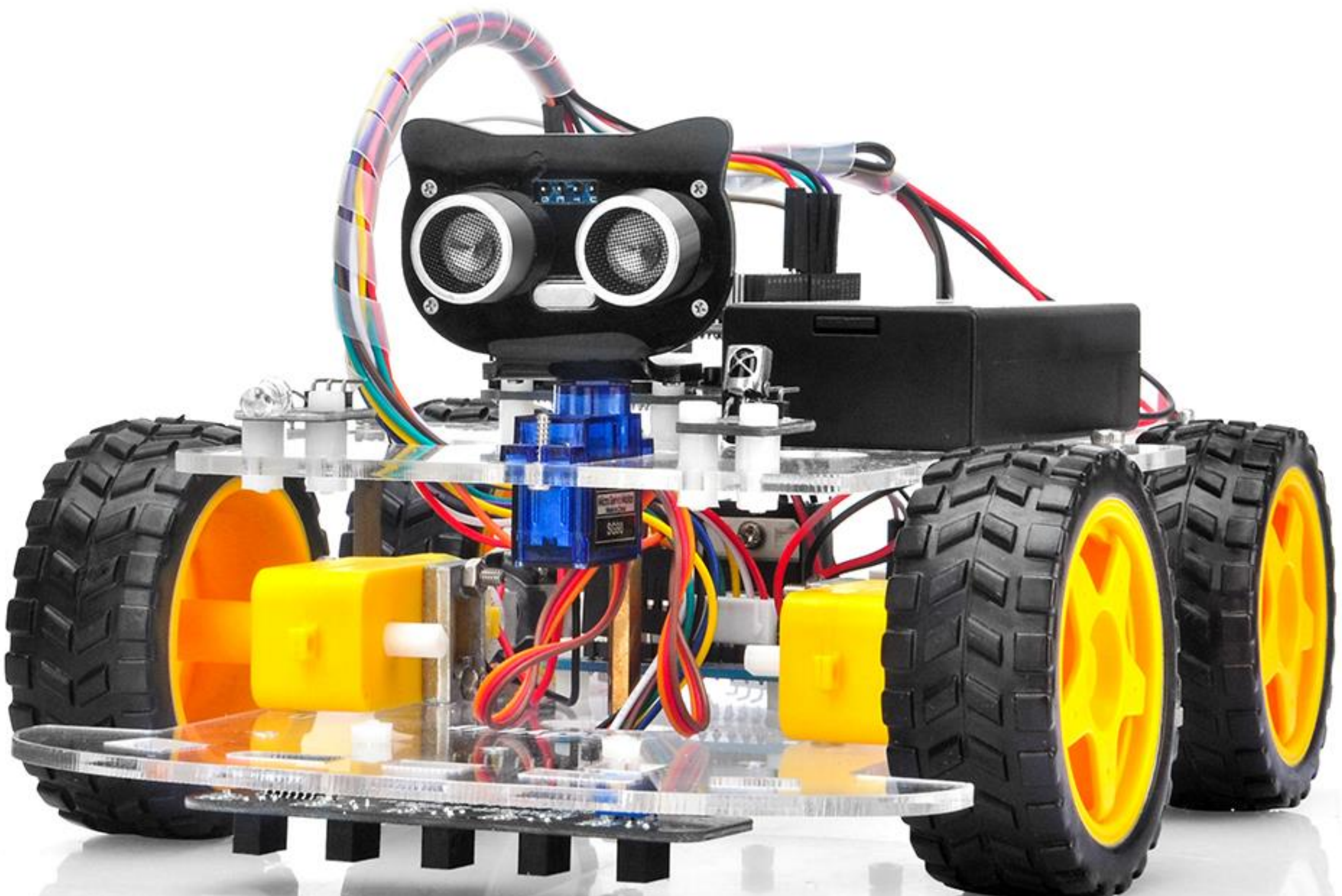
- For the calculation of the object distance, the sensor measures the time taken by the signal to travel between the transmission of the sound by the transmitter to the reflecting back towards the receiver.

$$\text{Distance} = \tfrac{1}{2}\,\text{Time} \times \text{Speed}$$

- The speed of sound in the air at 20ºC is 343 m/s.

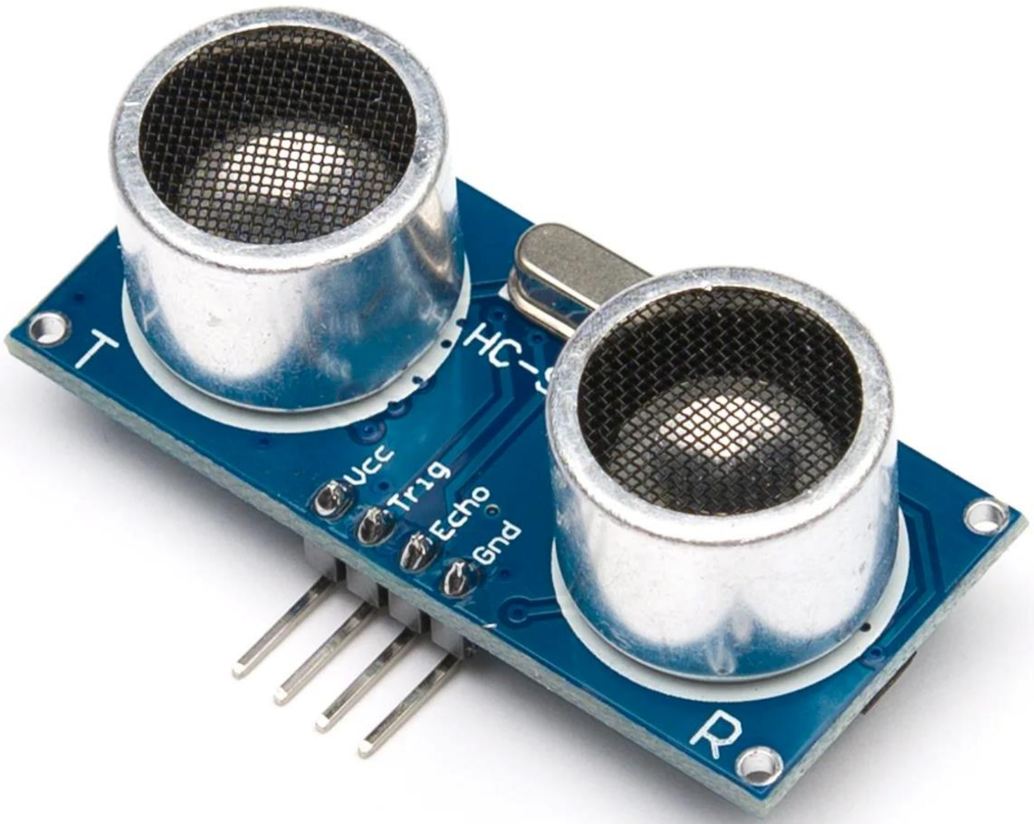


Ultrasonic Sensor: Applications

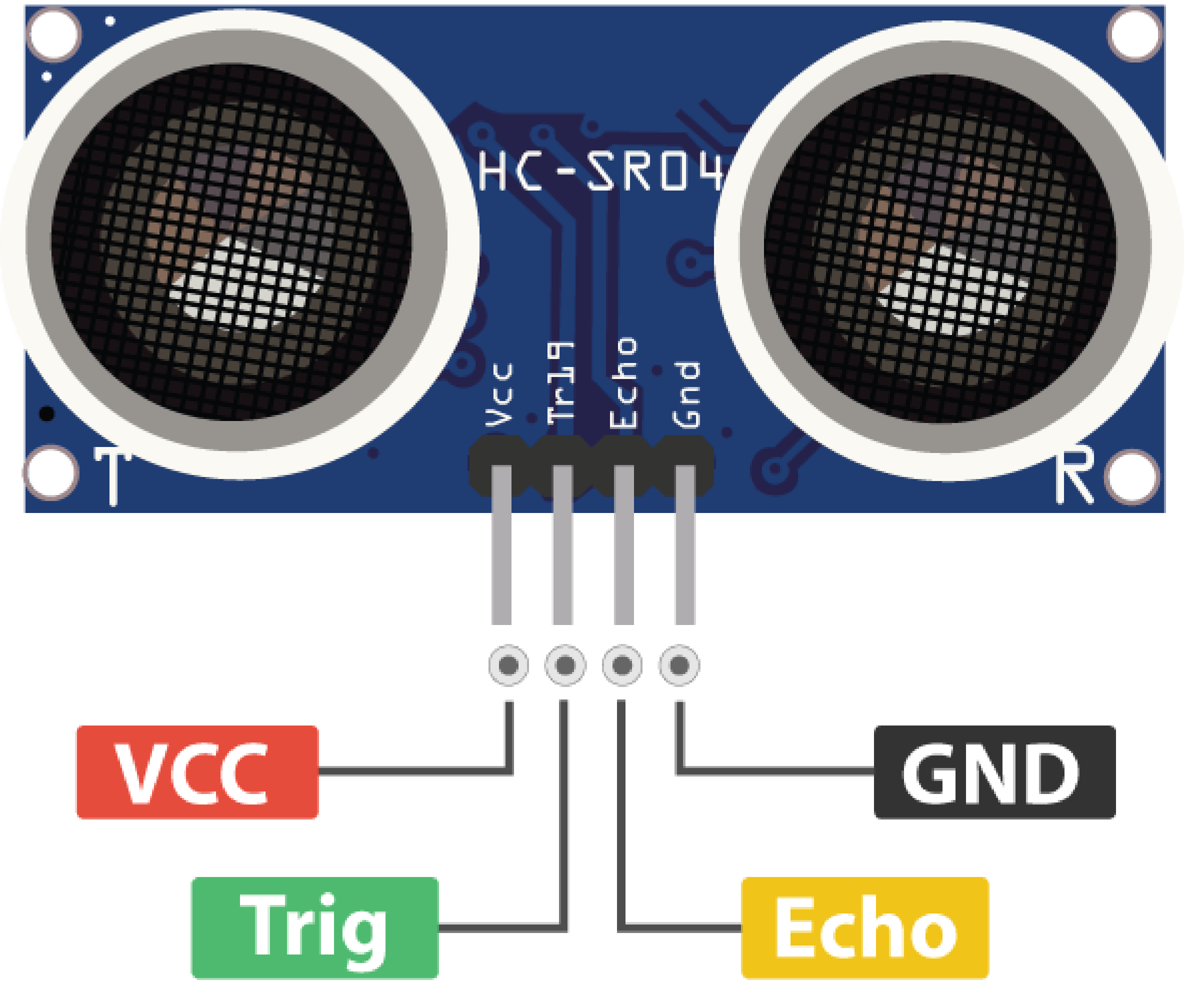


Ultrasonic Sensor: HC-SR04

- The HC-SR04 is an affordable and easy-to-use distance measuring sensor which has a range from 2cm to 400cm.

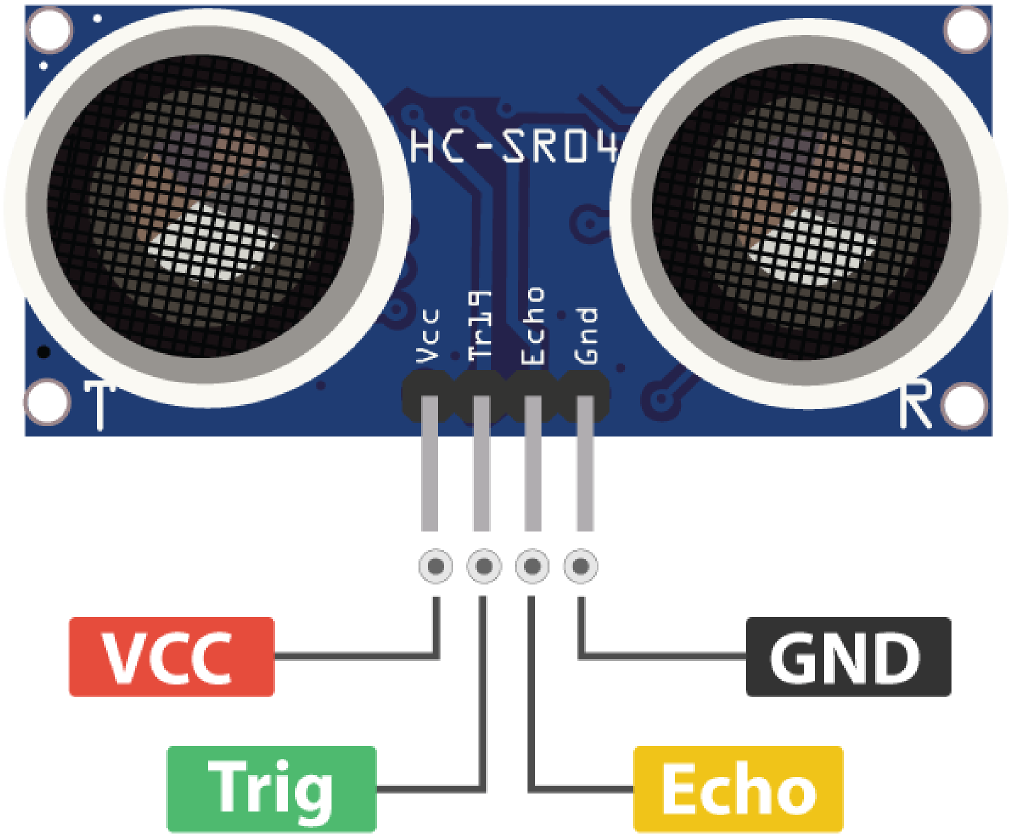


Ultrasonic Sensor: HC-SR04 Pinout



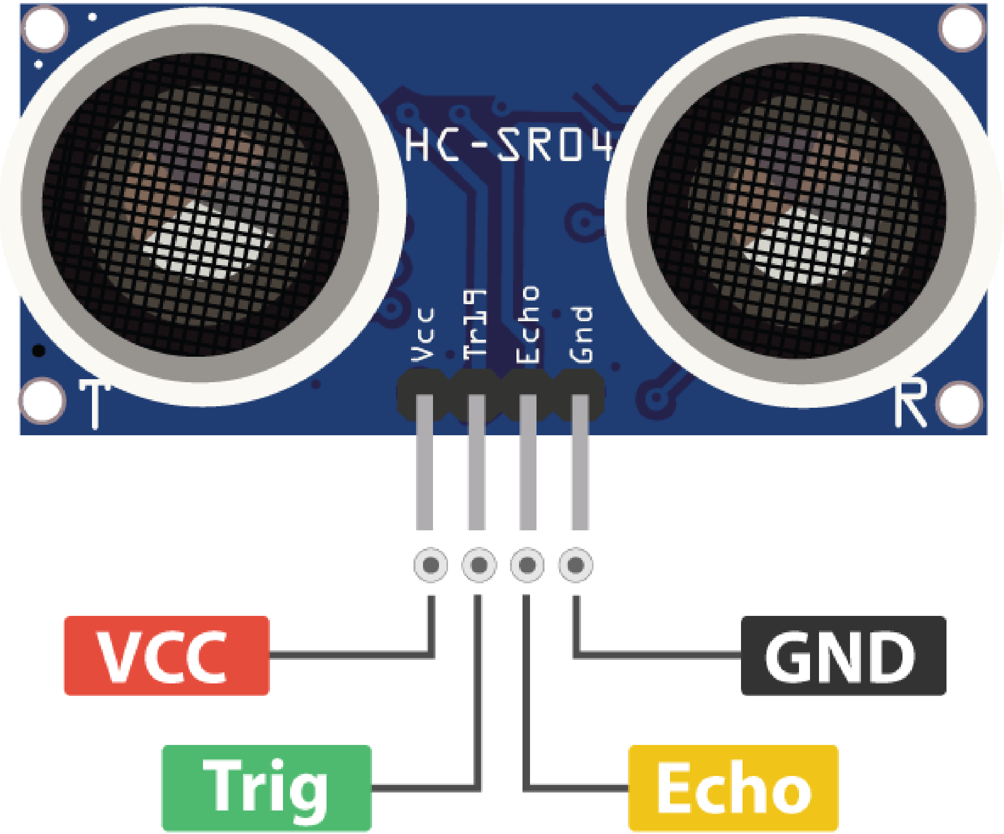
Ultrasonic Sensor: HC-SR04 Pinout

- The VCC supplies power to the HC-SR04 ultrasonic sensor.
- The GND is the ground pin.

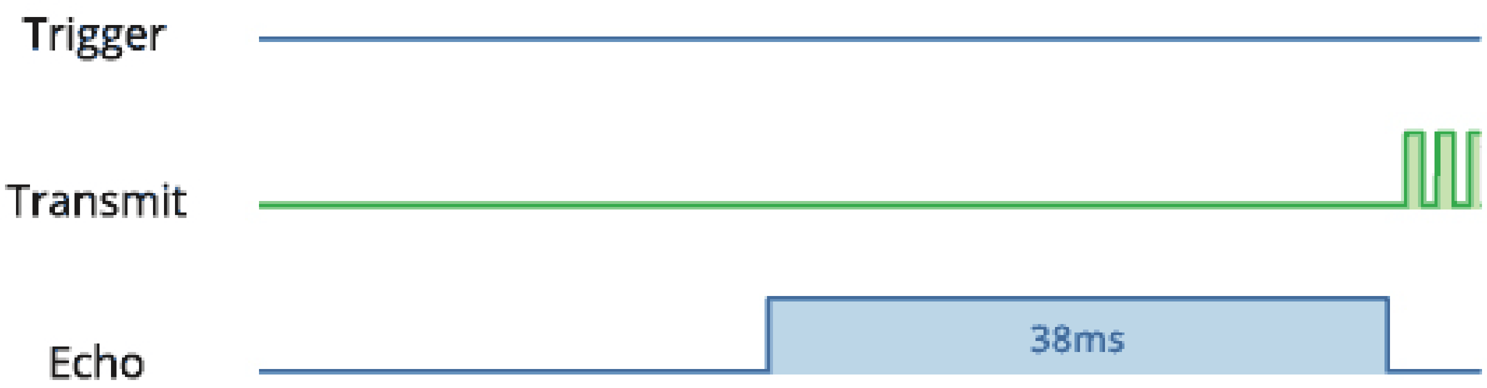


Ultrasonic Sensor: HC-SR04 Pinout

- The Trig pin is used to trigger ultrasonic sound pulses.
- By setting this pin to HIGH for 10µs, the sensor outputs an ultrasonic wave.

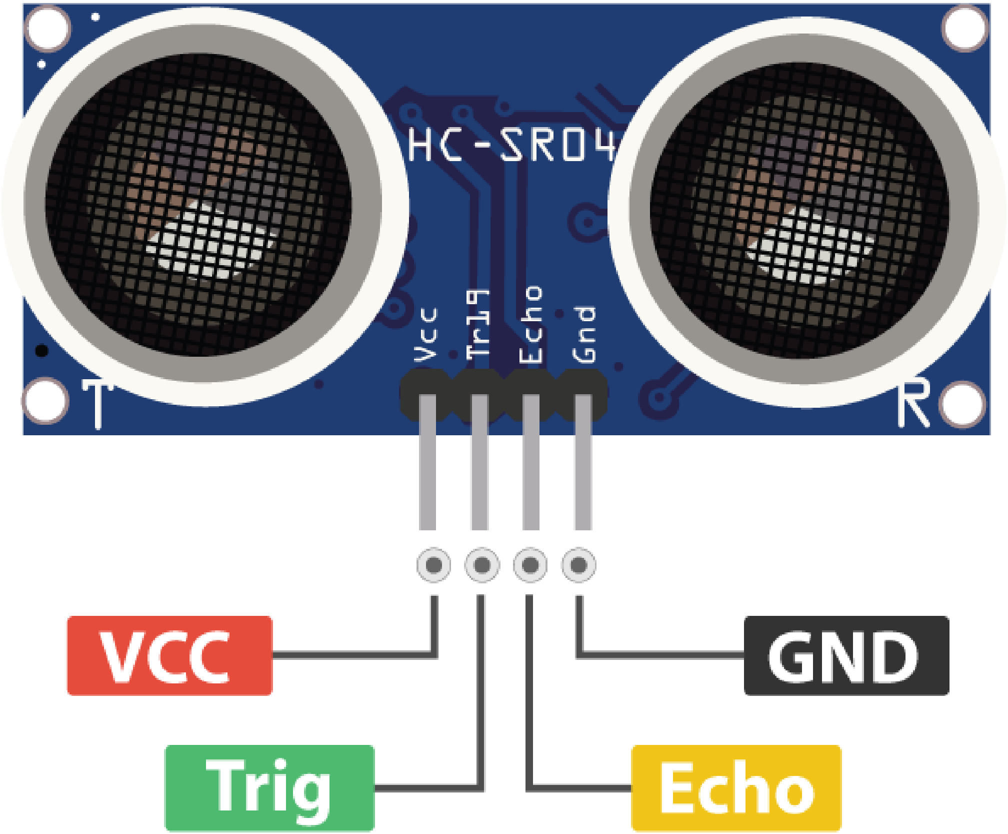


Ultrasonic Sensor: HC-SR04 Pinout

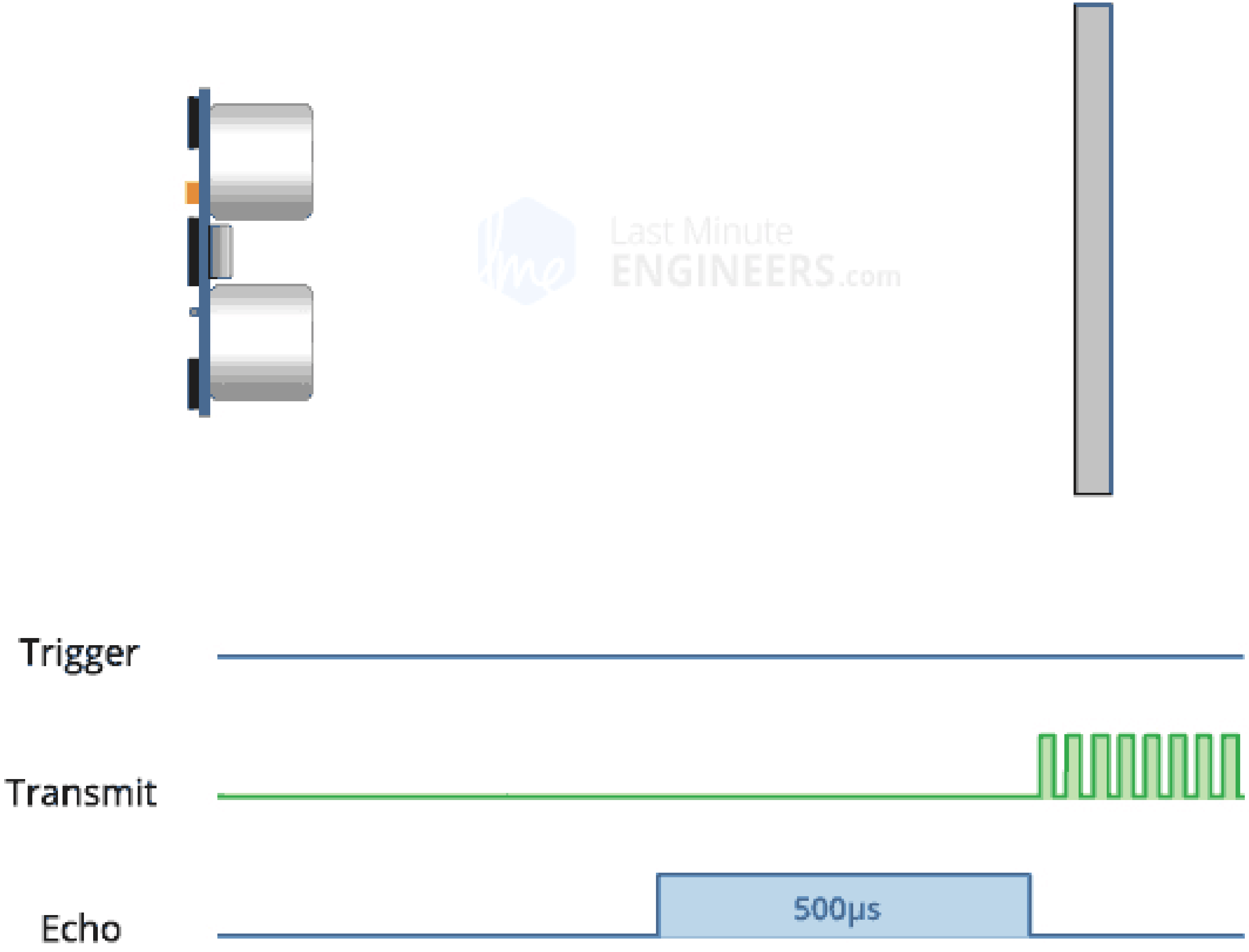


Ultrasonic Sensor: HC-SR04 Pinout

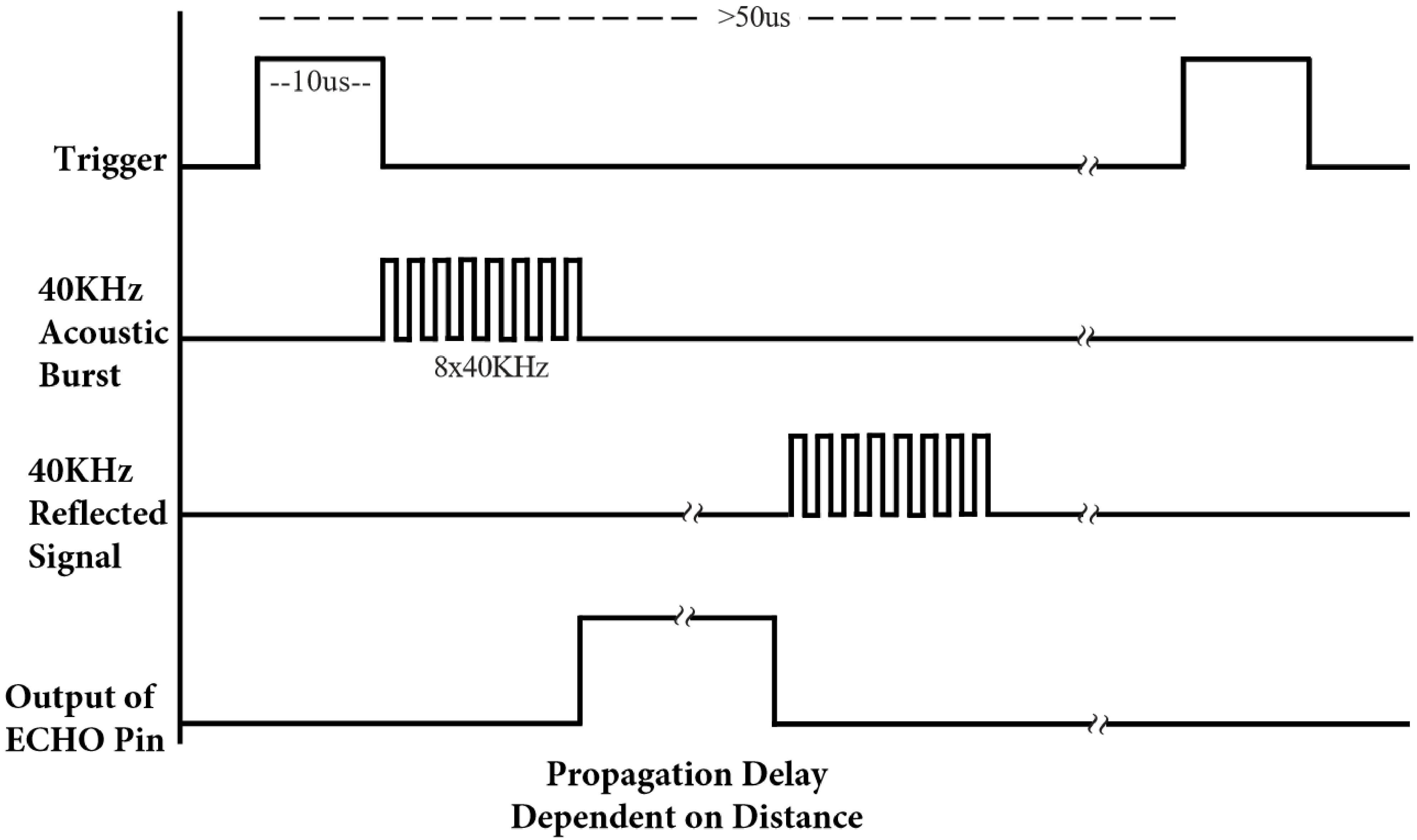
- The Echo pin goes HIGH when the ultrasonic wave is transmitted and remains HIGH until the sensor receives an echo, after which it goes LOW.



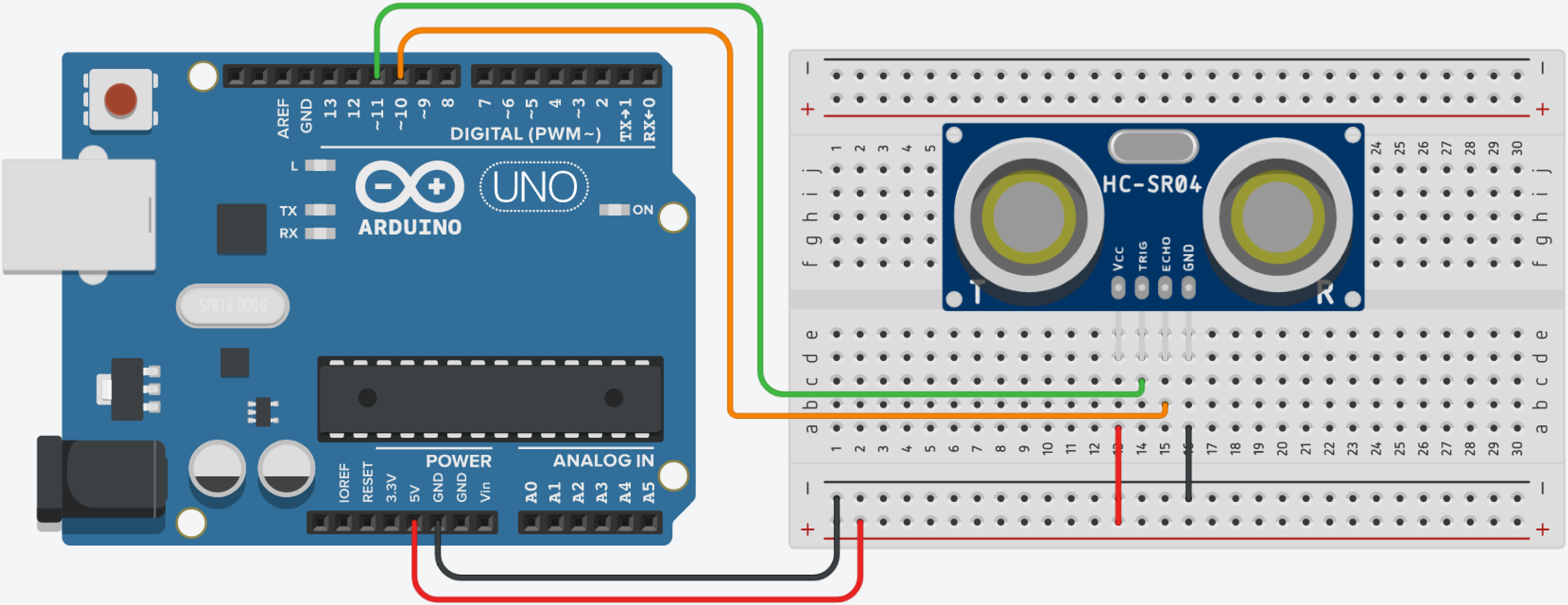
Ultrasonic Sensor: HC-SR04 Pinout



Ultrasonic Sensor: HC-SR04 Working Principle

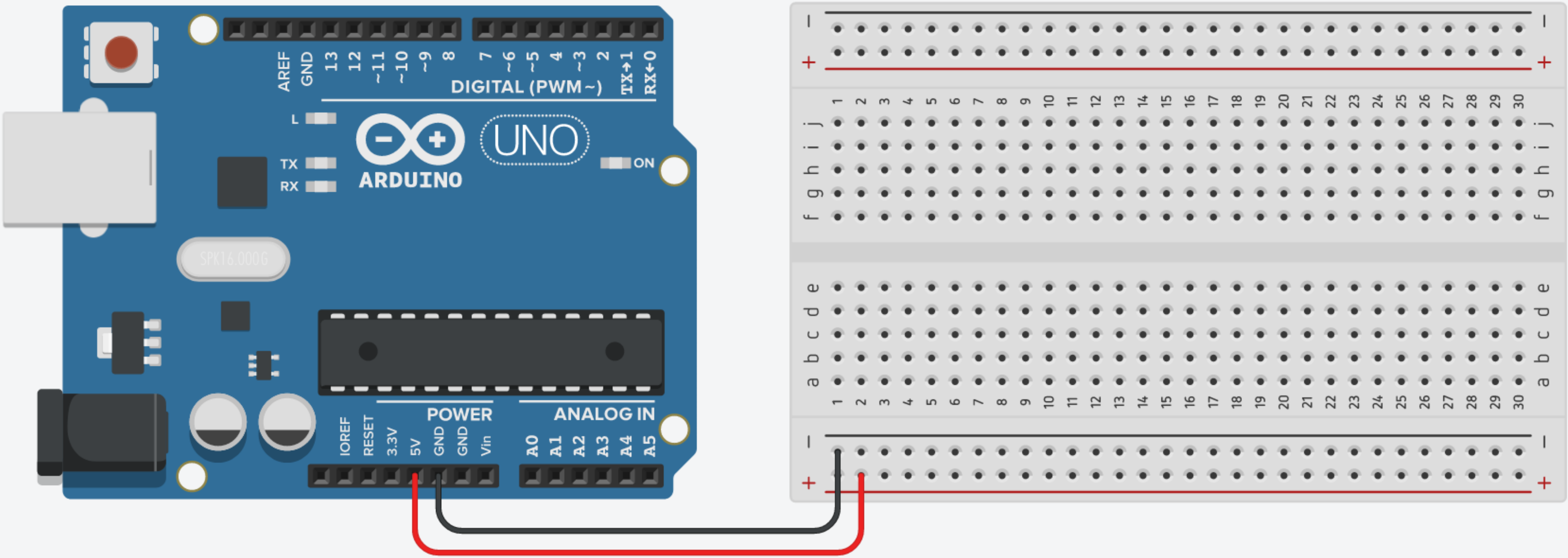


Ultrasonic Sensor: Circuit



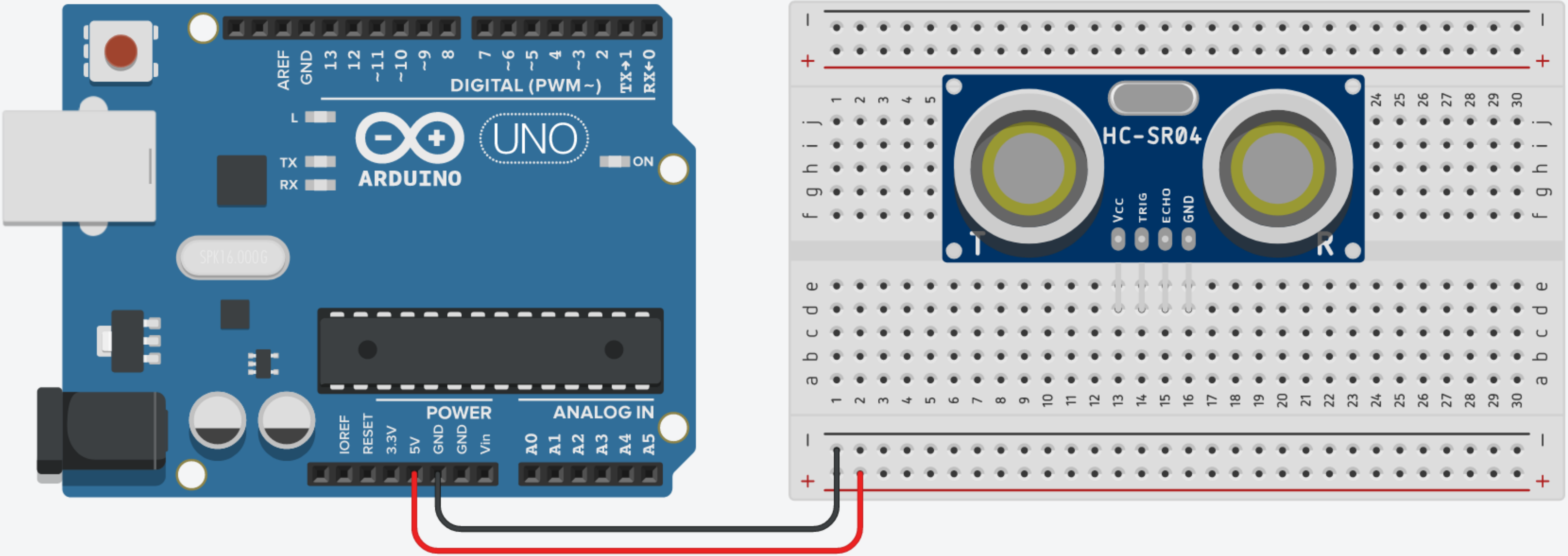
Ultrasonic Sensor: Steps

1. Connect breadboard power (+) and ground (-) rails to Arduino 5V and ground (GND), respectively.



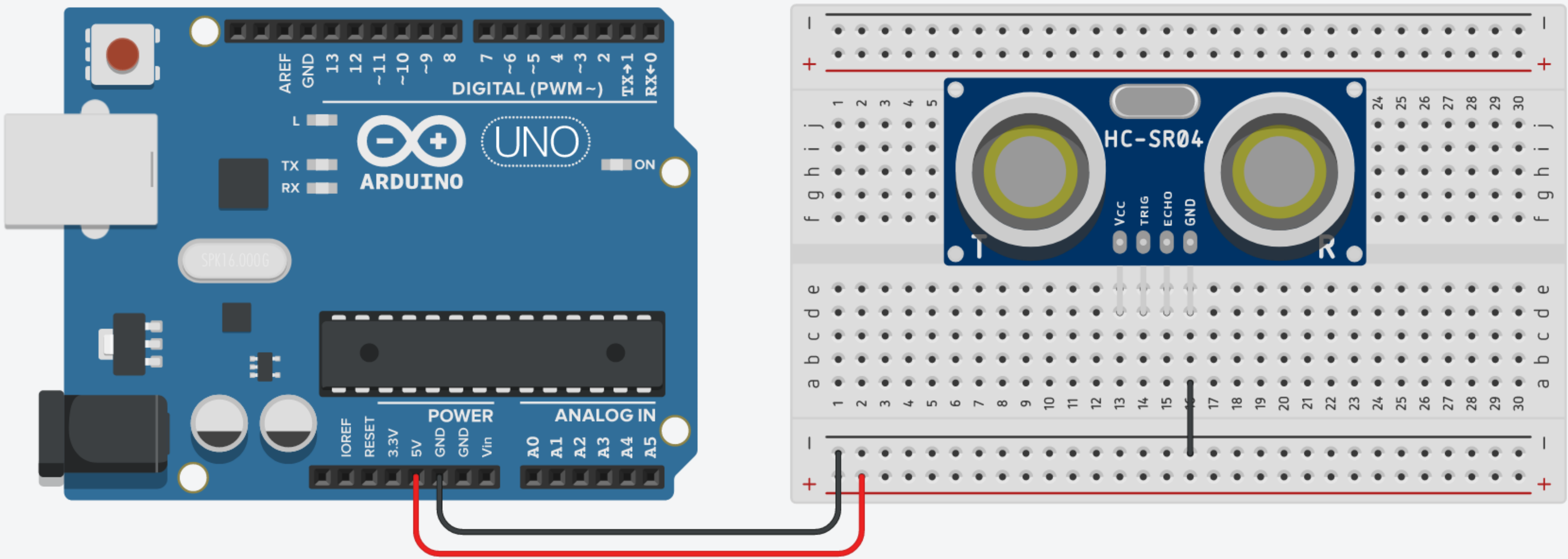
Ultrasonic Sensor: Steps

2. Plug the HC-SR04 Ultrasonic Sensor into the breadboard.



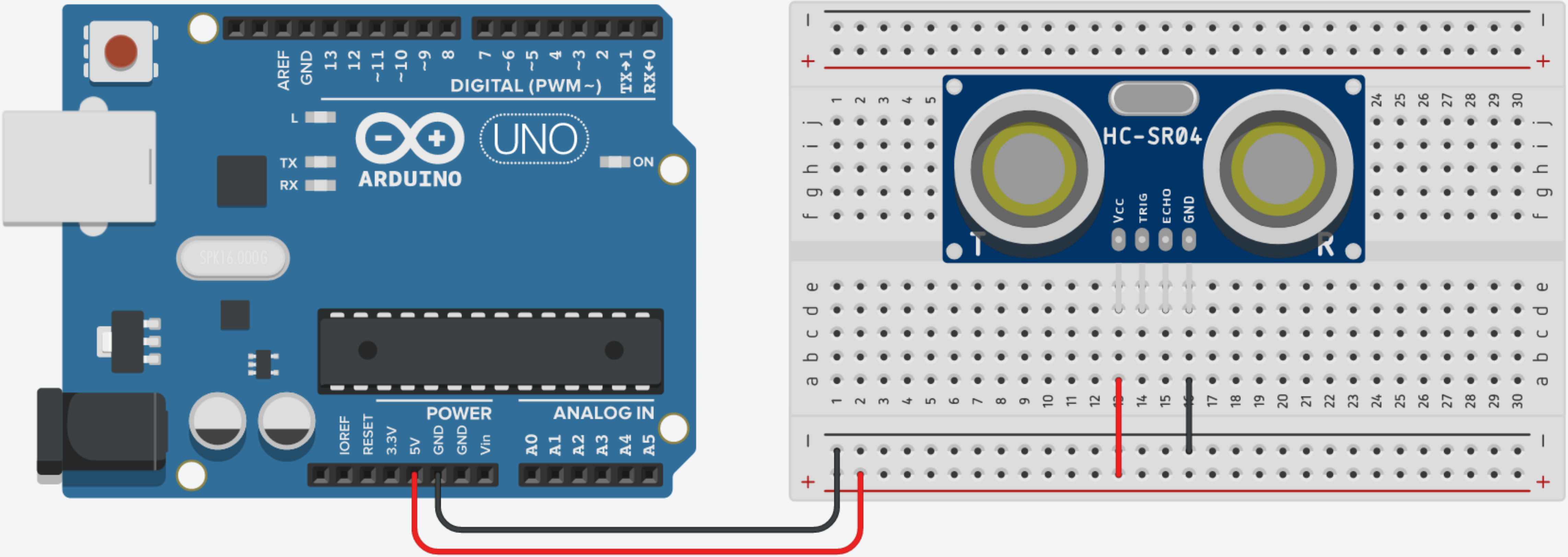
Ultrasonic Sensor: Steps

3. The GND pin of the sensor connects to the ground.



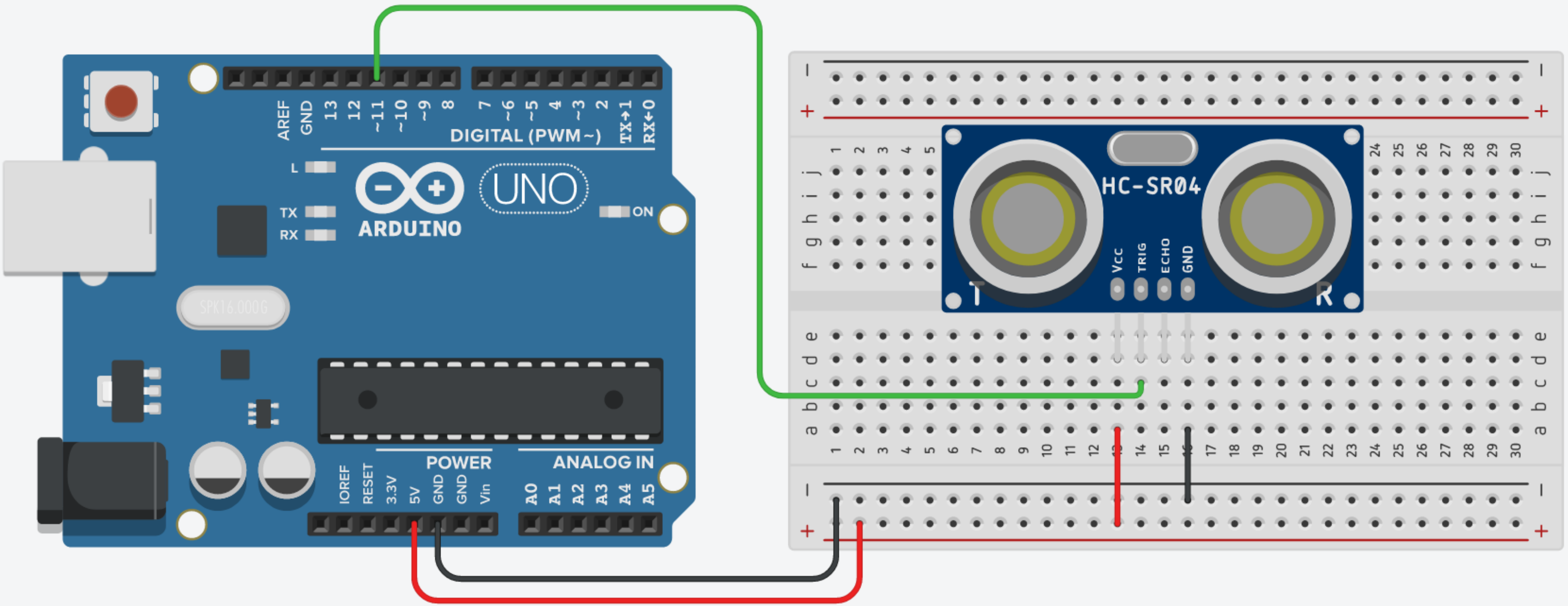
Ultrasonic Sensor: Steps

4. The VCC pin of the sensor connects to the power.



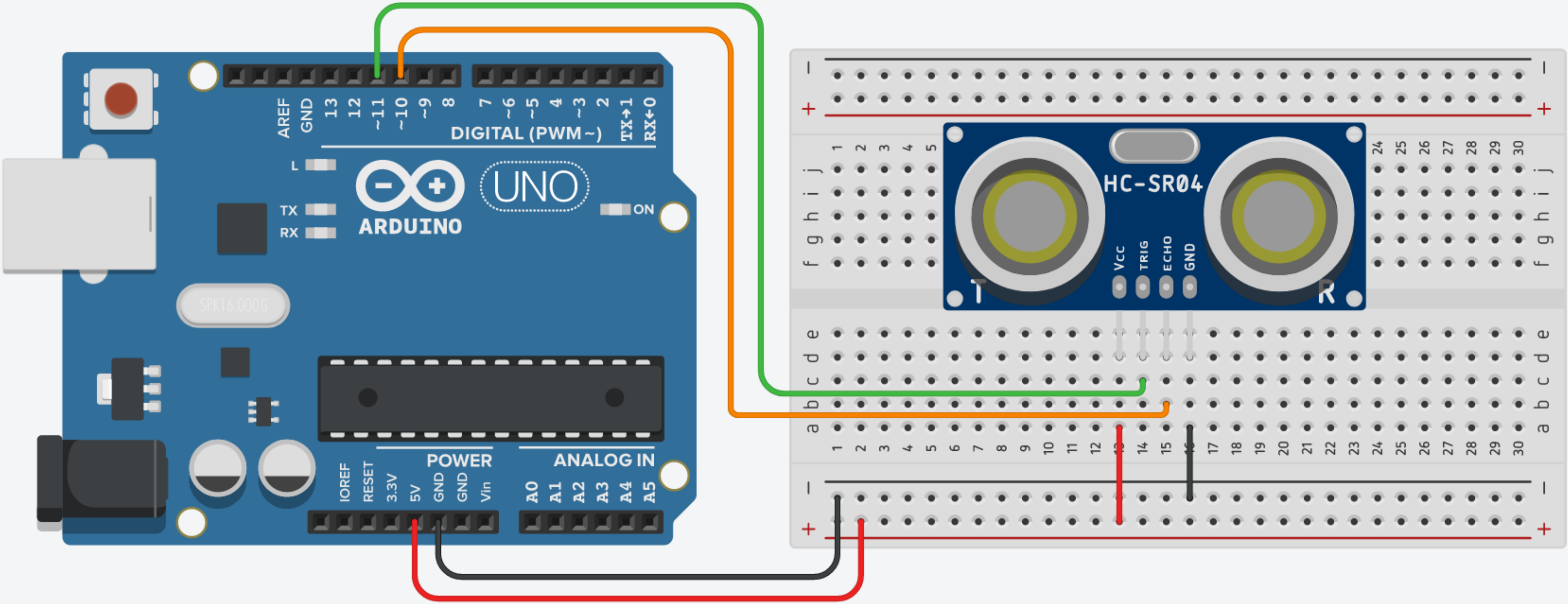
Ultrasonic Sensor: Steps

5. The Trig pin of the sensor connects to pin 11 on Arduino.



Ultrasonic Sensor: Steps

6. The Echo pin of the sensor connects to pin 10 on Arduino.

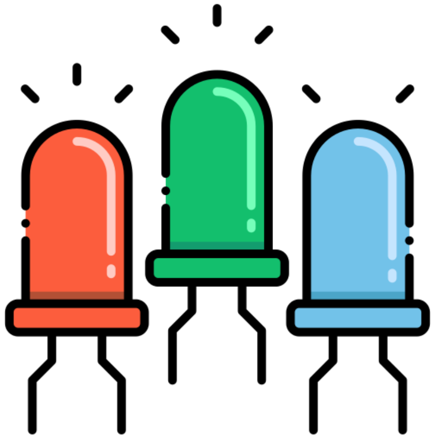


Ultrasonic Sensor: Code

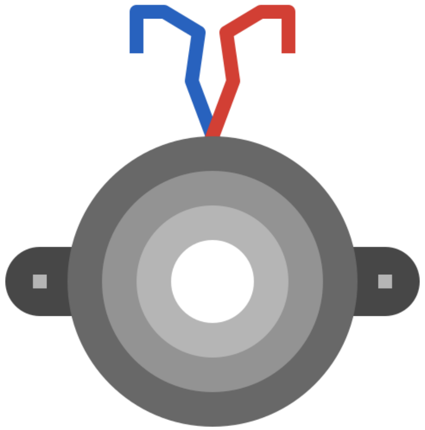
```
#define TRIG_PIN 11                        // Trigger pin of the ultrasonic sensor
#define ECHO_PIN 10                        // Echo pin on the ultrasonic sensor

long t;                                    // Variable to hold the time
int distance;                              // Variable to hold the distance

void setup() {
  Serial.begin(9600);                      // Begin serial communication
  pinMode(TRIG_PIN, OUTPUT);               // Set TRIG_PIN as an output
  pinMode(ECHO_PIN, INPUT);                // Set ECHO_PIN as an input
}

void loop() {
  digitalWrite(TRIG_PIN, LOW);             // Make sure that TRIG_PIN is LOW
  delayMicroseconds(2);                    // for just 2 microseconds
  digitalWrite(TRIG_PIN, HIGH);            // Set the TRIG_PIN to HIGH
  delayMicroseconds(10);                   // for 10 microseconds
  digitalWrite(TRIG_PIN, LOW);             // Set the TRIG_PIN to LOW
  t = pulseIn(ECHO_PIN, HIGH);             // Return the length of pulse in microseconds
  distance = 0.5 * t * 0.0343;             // Calculate the distance (D = 0.5T * S)
  Serial.println(distance);                // Print the distance
  delay(500);                              // Short delay
}
```

Actuators

- Sensors turn a physical input into an electrical output, while actuators do the opposite.
- Actuators take electrical signals from control modules and turn them into physical outputs.

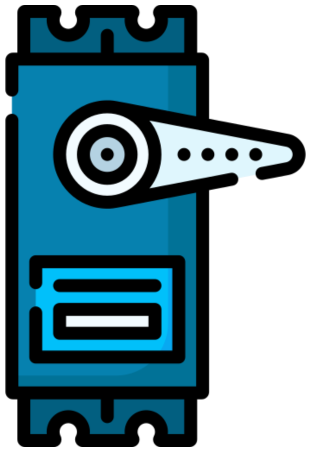


LEDs



Buzzer

DC Fan

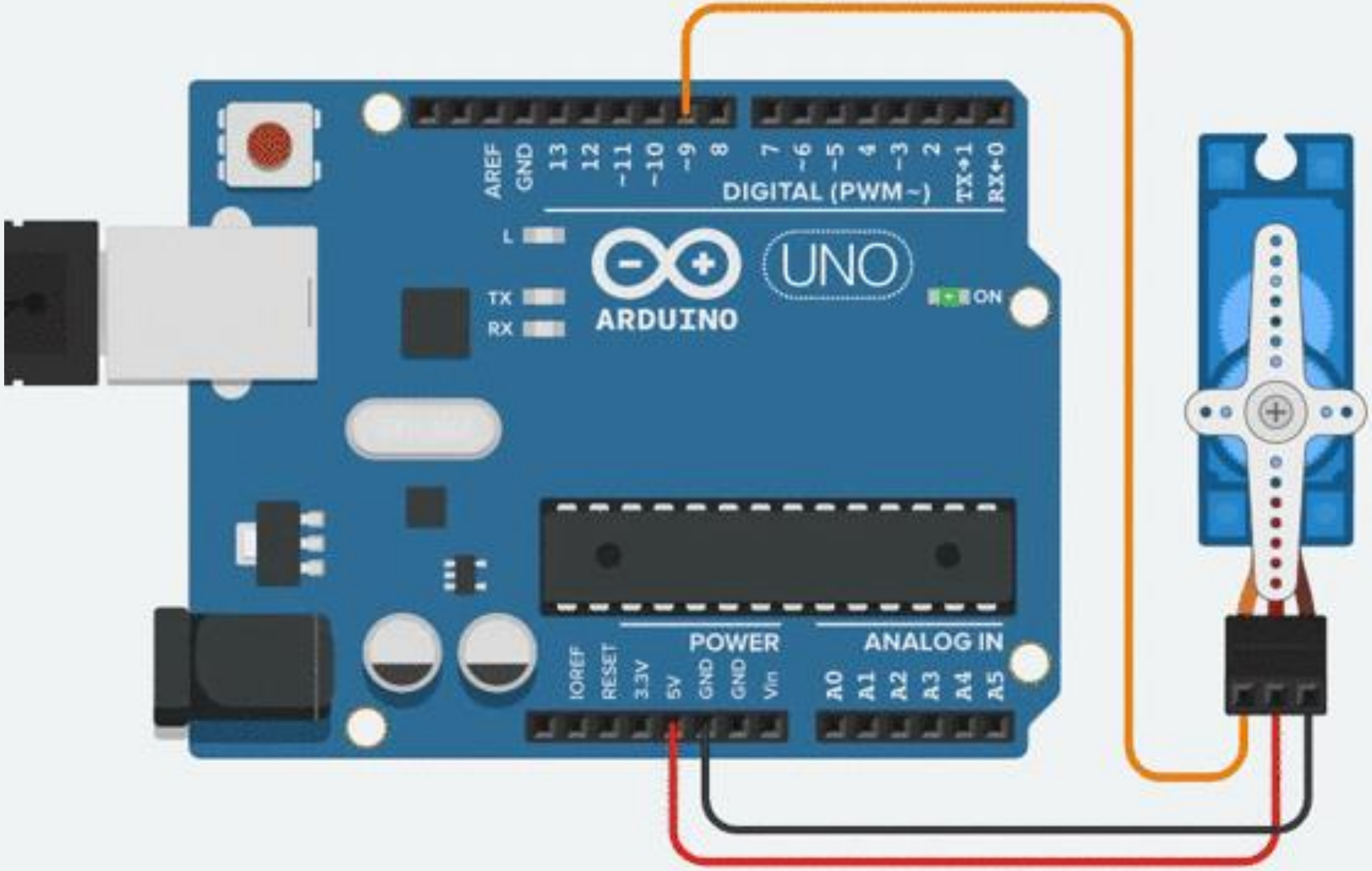


Servo Motor

Servo Motor

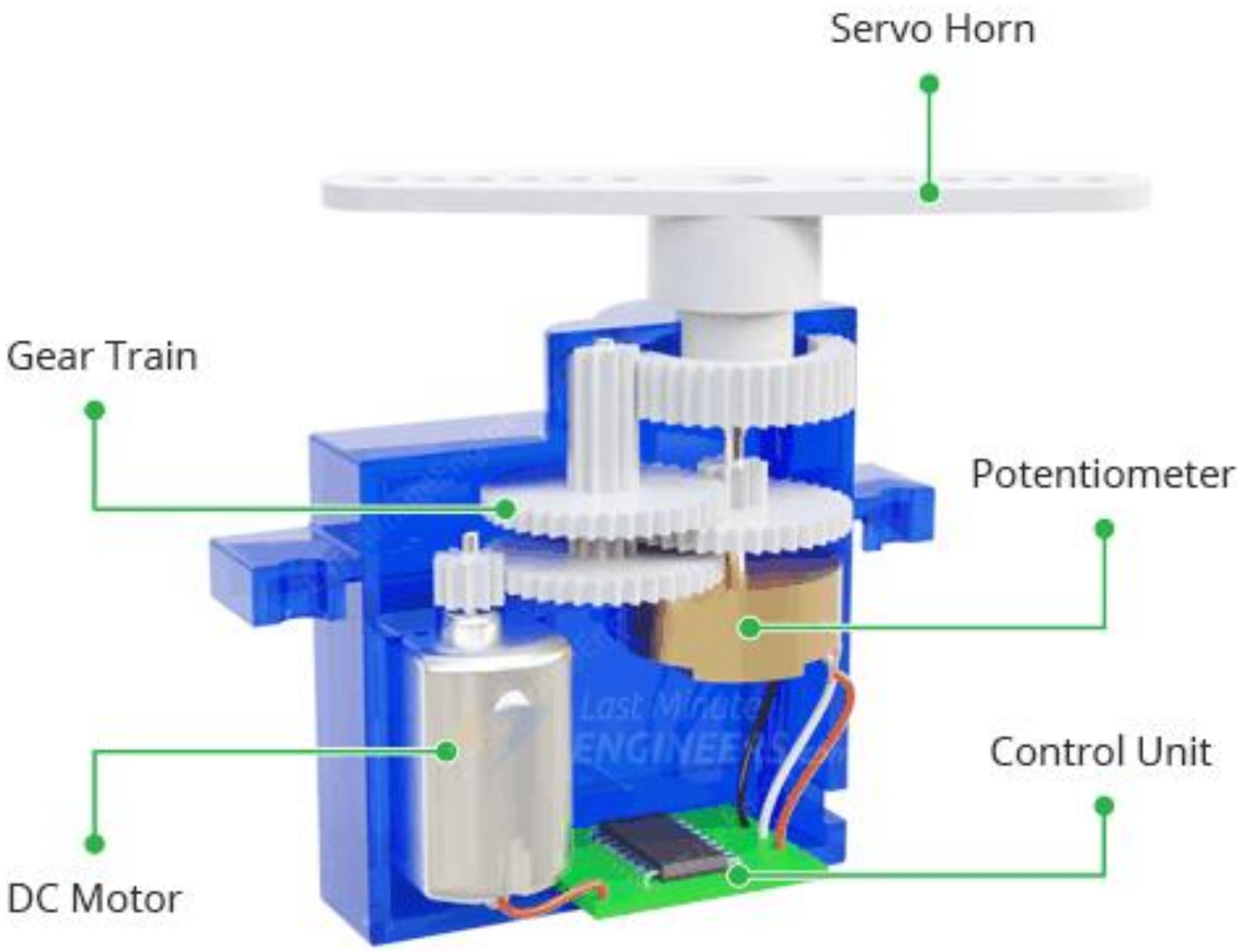
- Servo motors were first used in the Remote Control (RC) world, usually to control the steering of RC cars or the flaps on a RC plane.
- With time, they found their uses in robotics, automation, and of course, the Arduino world.
- There are many motors to pick from, but it's important to pick the right one for the job.
- If your job requires positioning, a servo motor is usually best option.
- A servo motor can turn 180° degrees.

Servo Motor

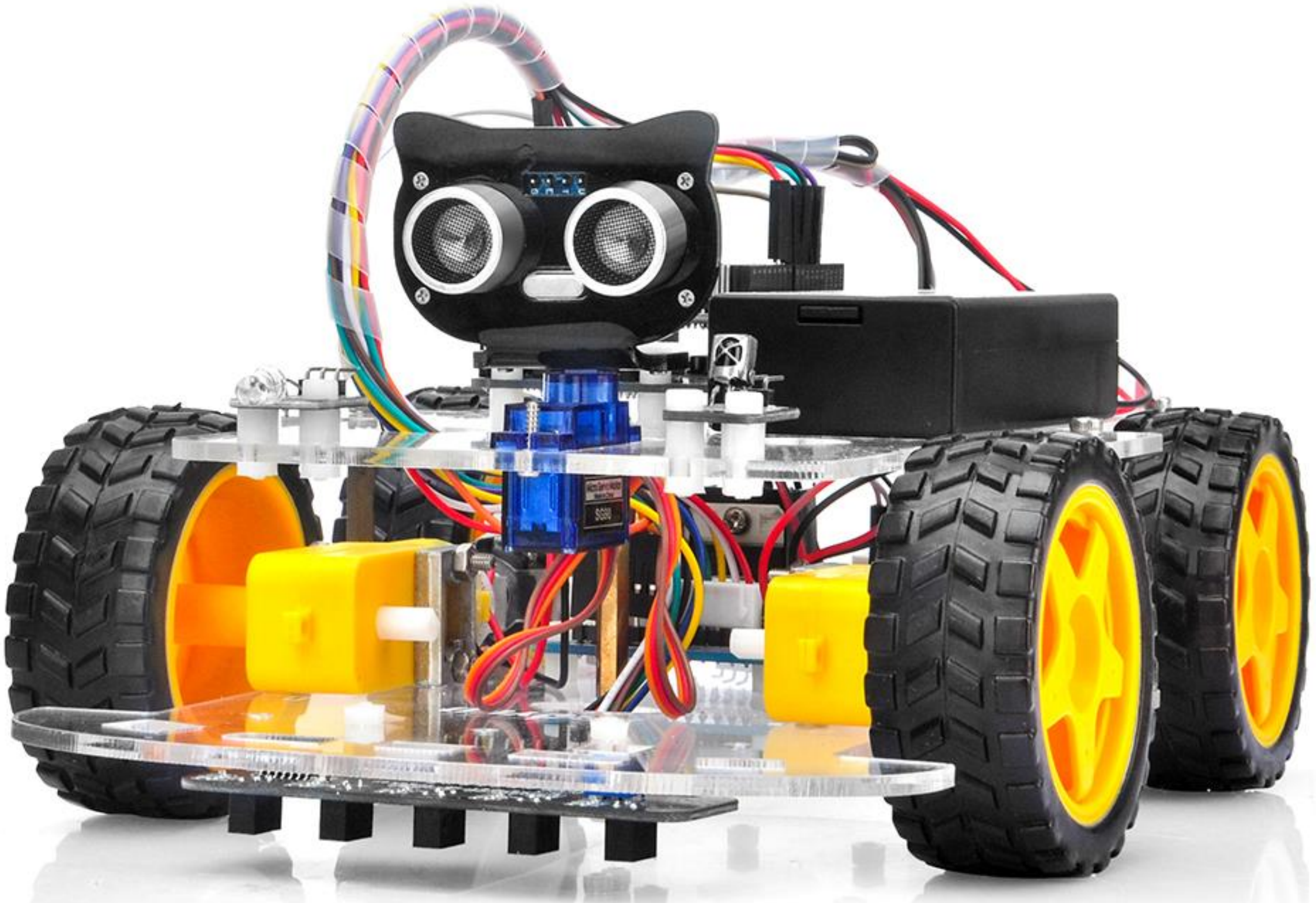


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▸1
RX◂0
DIGITAL (PWM~)
L
TX
RX
UNO
ARDUINO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

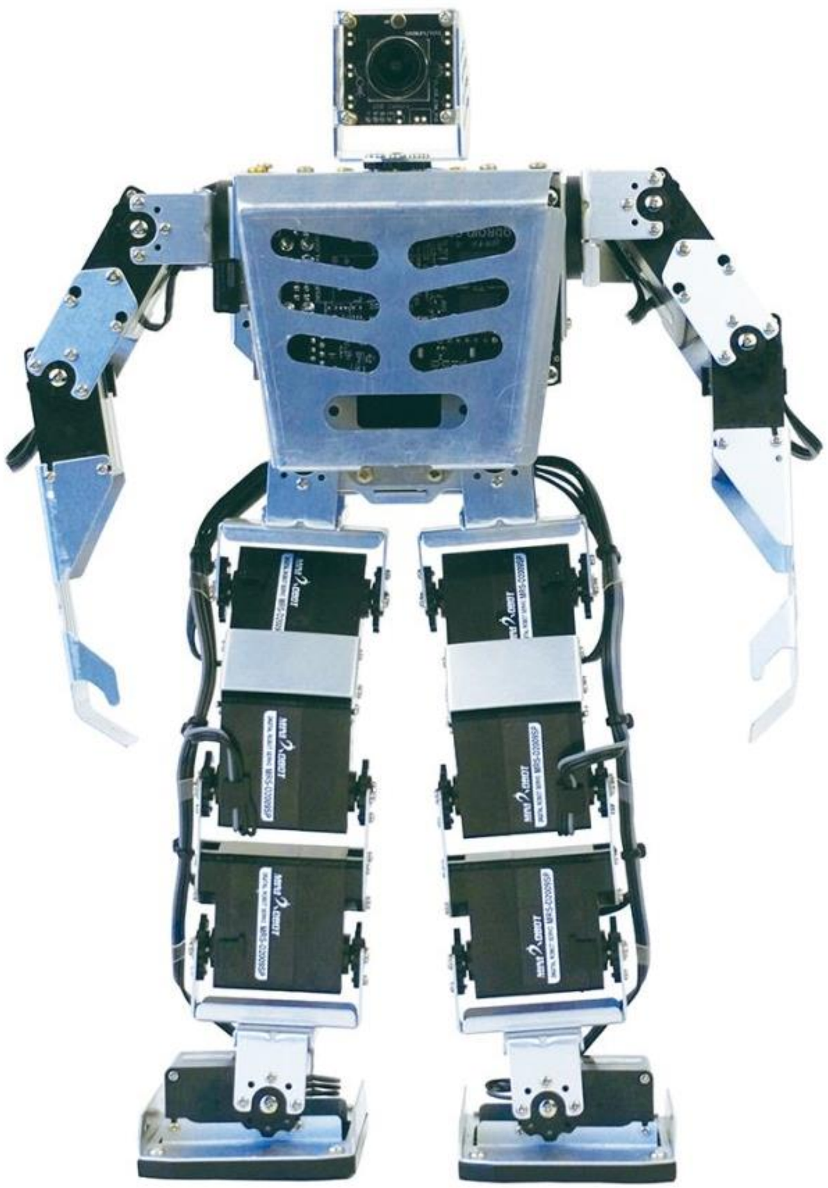
Servo Motor: Internal Structure



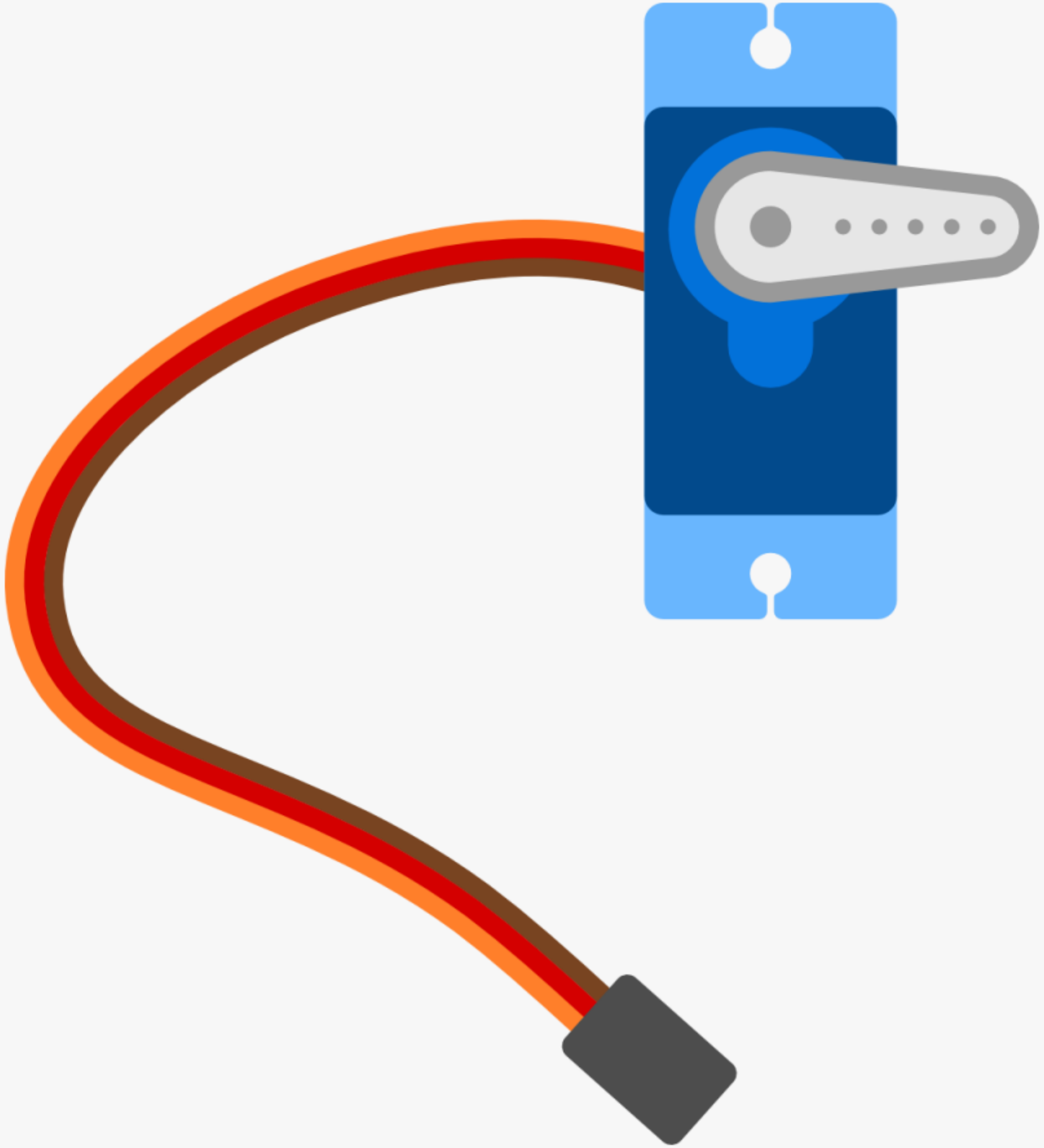
Servo Motor: Applications



Servo Motor: Applications

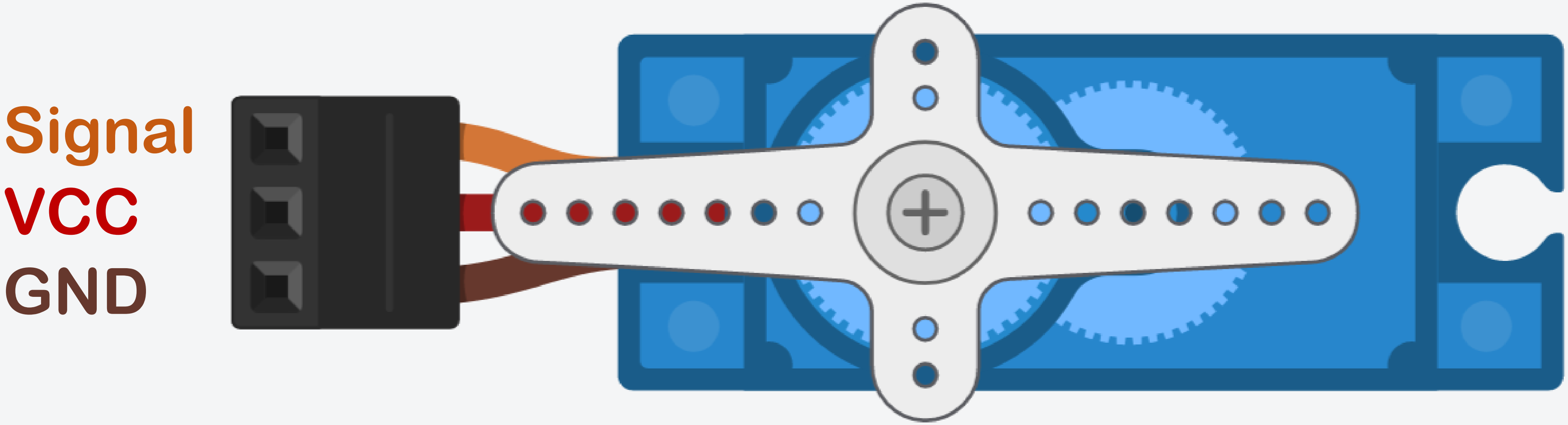


Servo Motor: Pinout

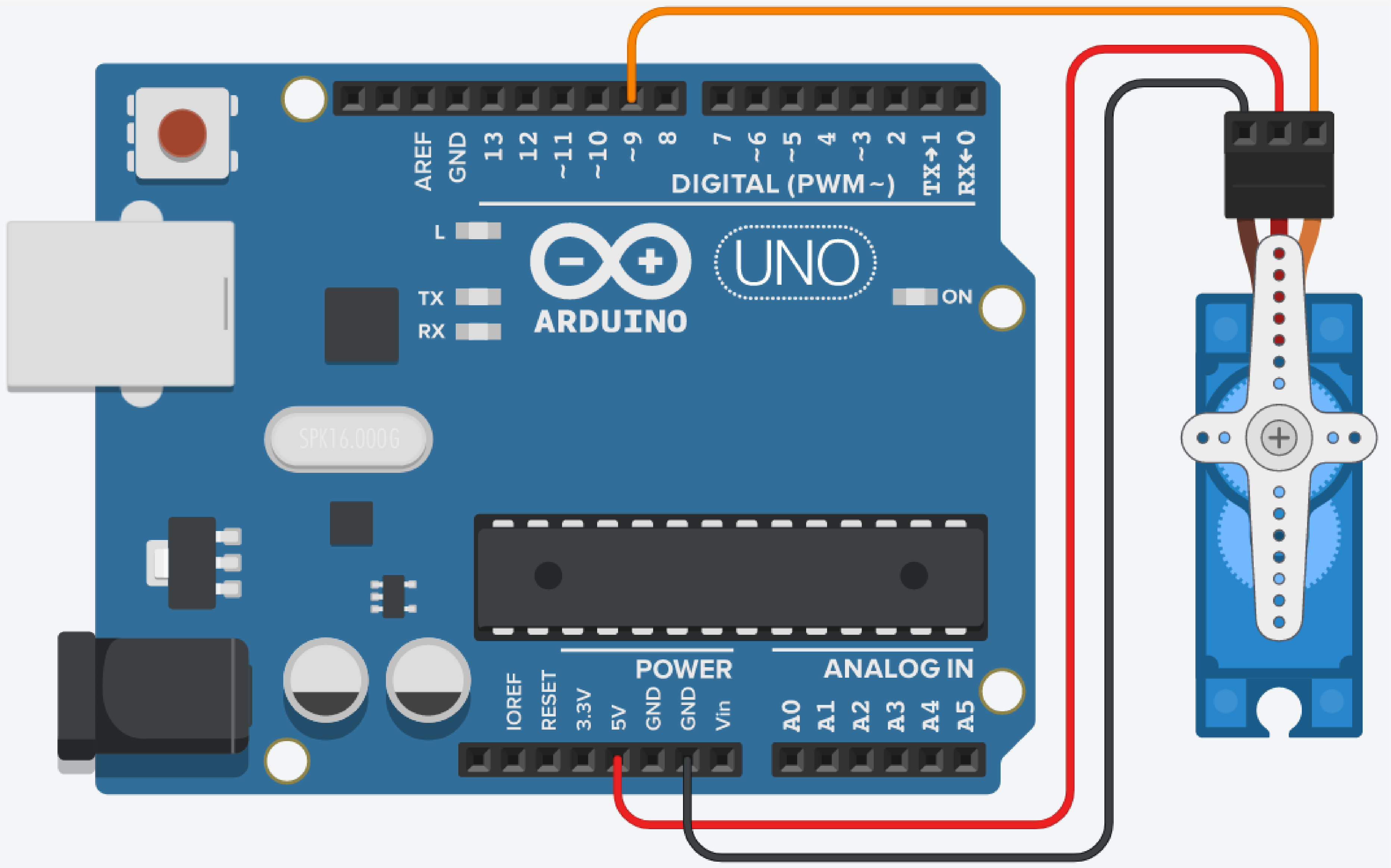


Servo Motor: Pinout

- As you can see, Servo has 3 pins: Signal, VCC and GND.
- These pins can be recognized based on the color of the wire.
- Usually Orange is signal wire, Red is VCC and Black/Brown is GND.

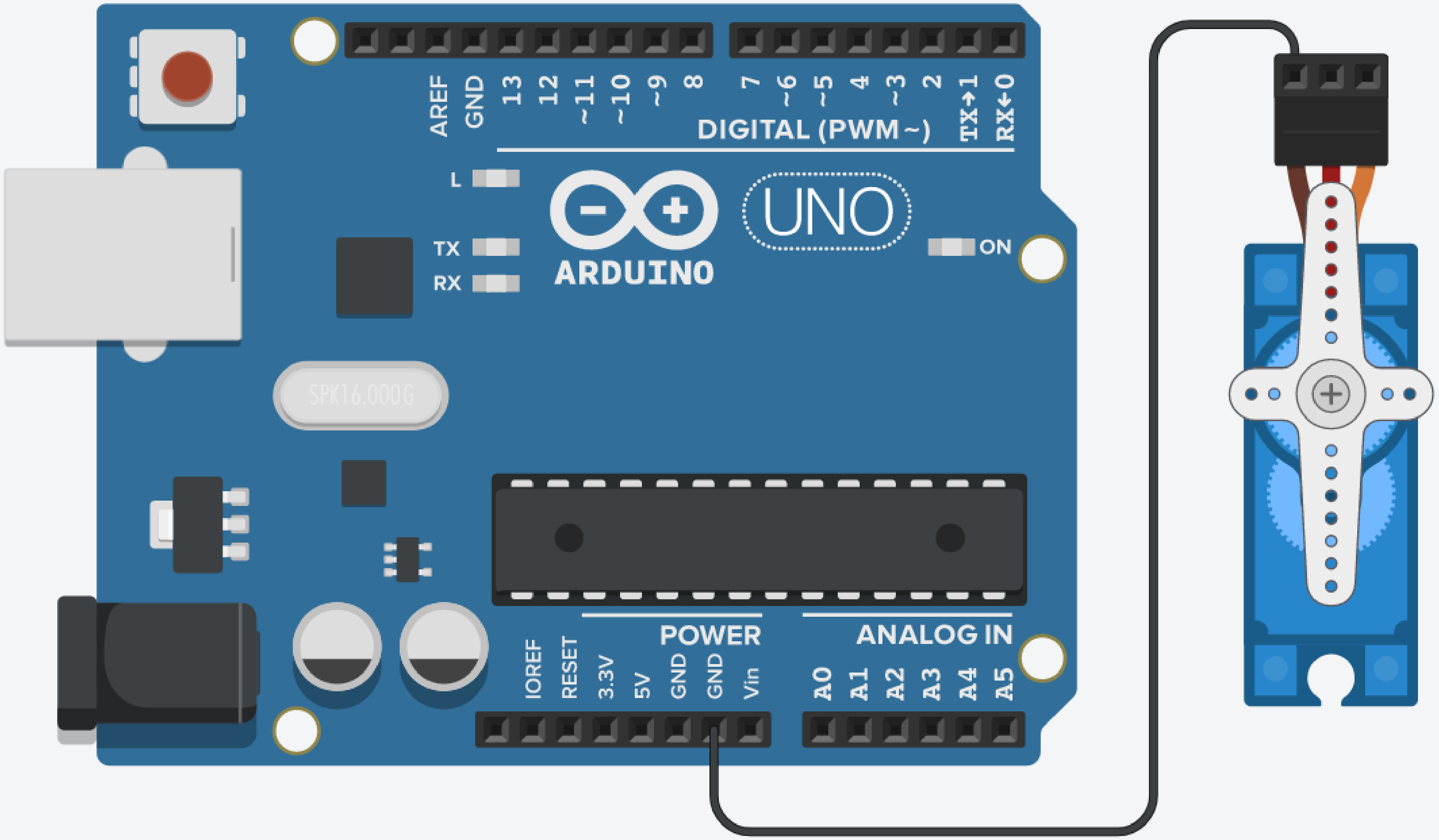


Servo Motor: Circuit



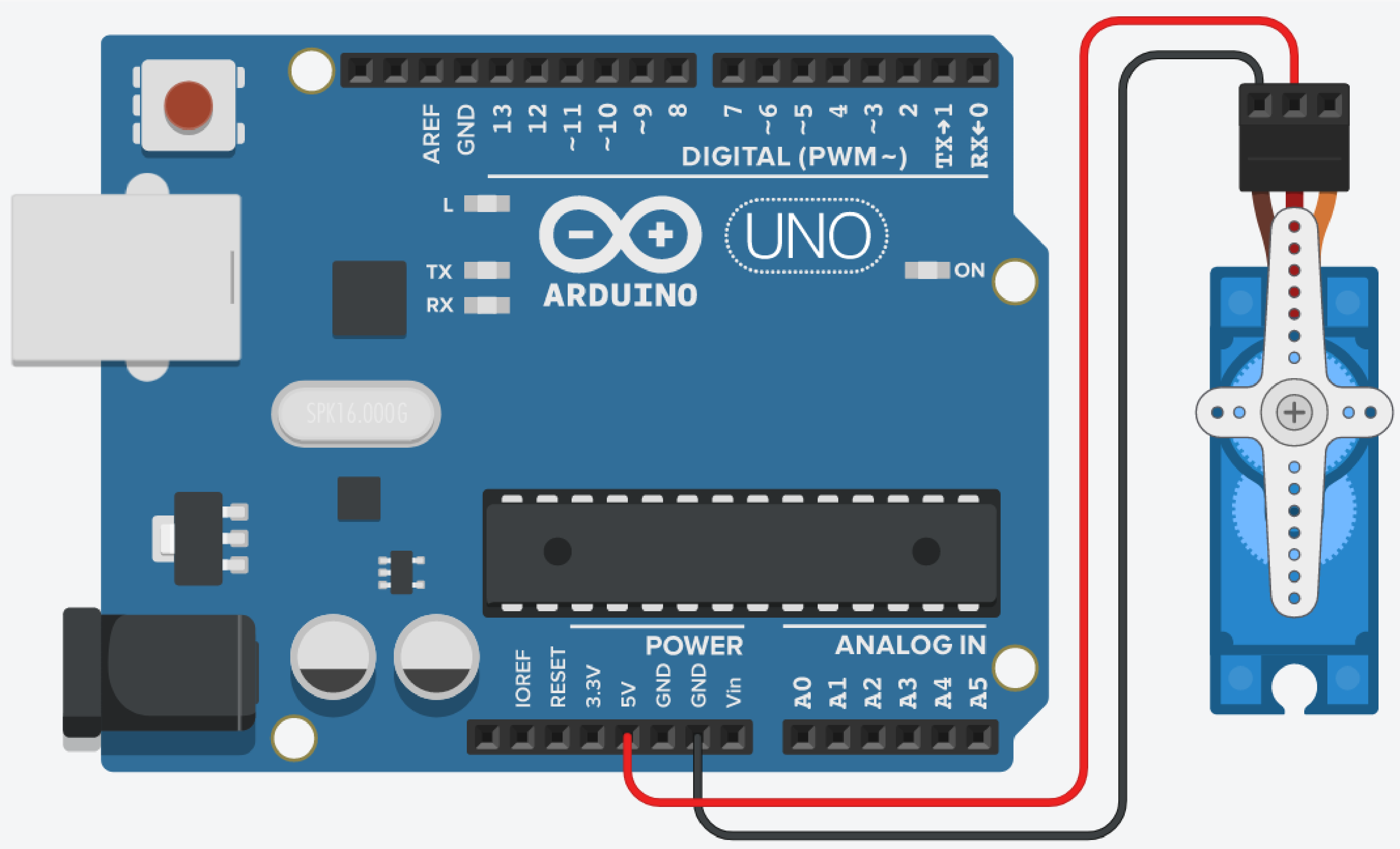
Servo Motor: Steps

1. The GND pin of the servo connects to the ground on Arduino.



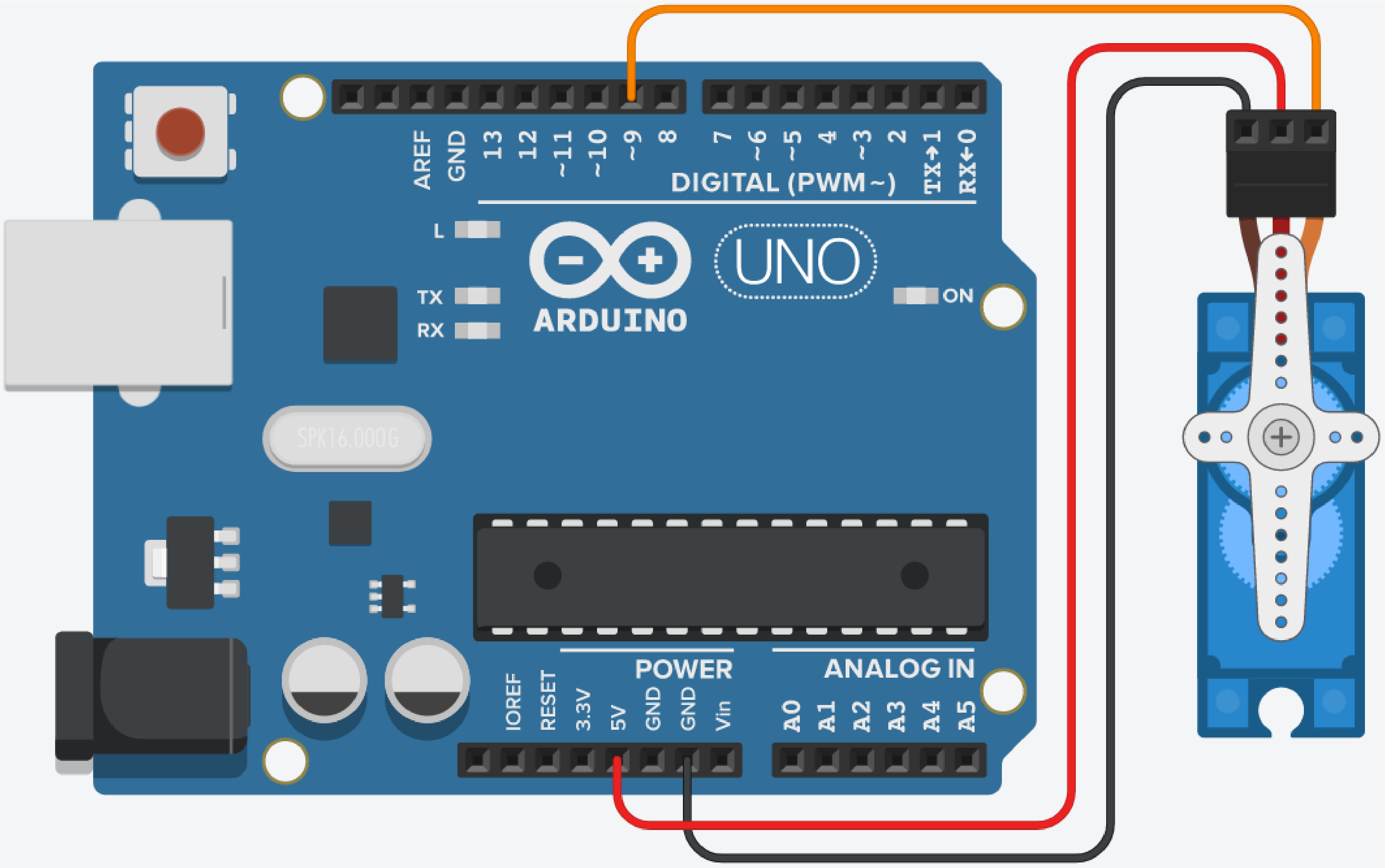
Servo Motor: Steps

2. The VCC pin of the servo connects to the 5V on Arduino.



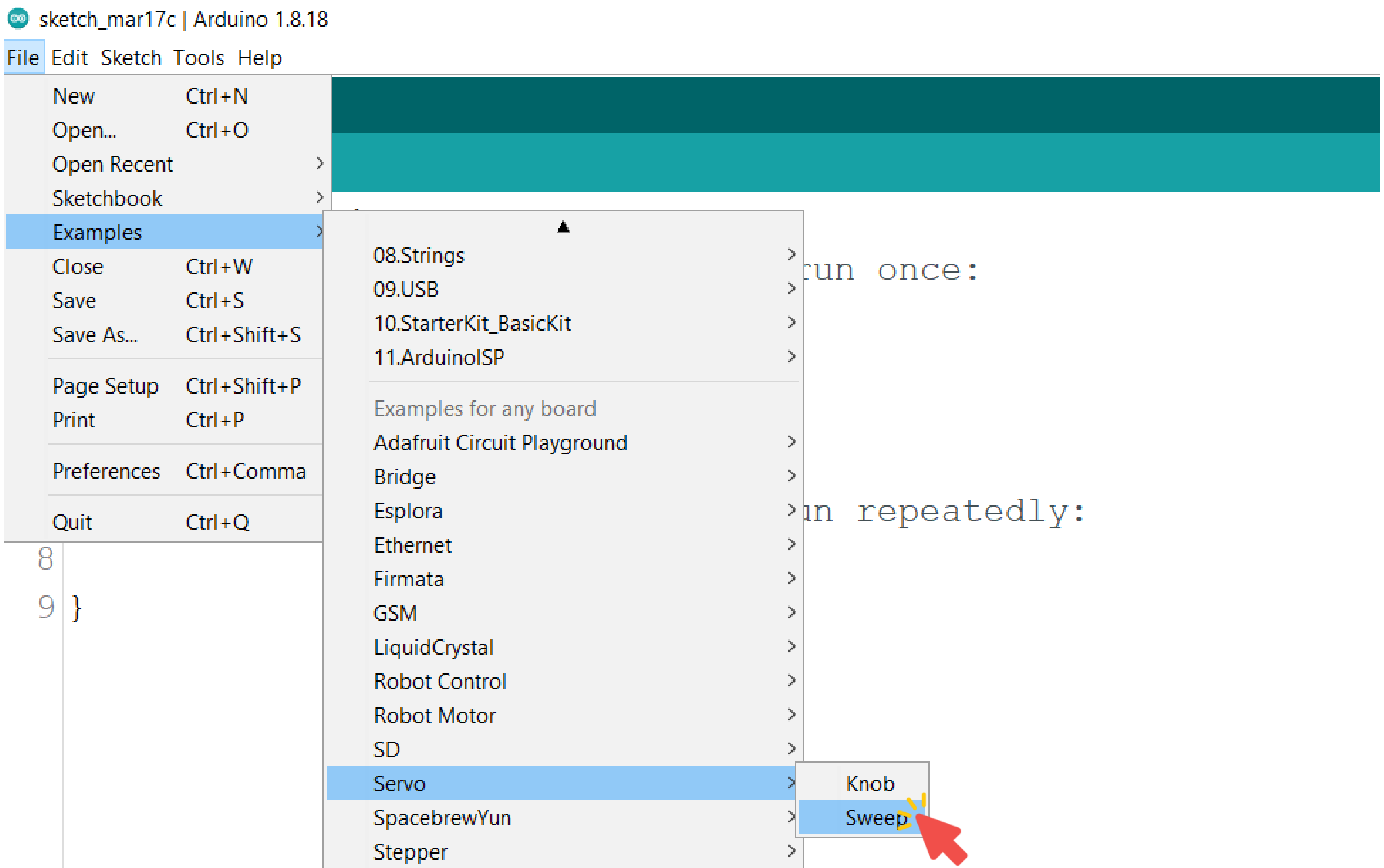
Servo Motor: Steps

3. The Signal pin of the servo connects to pin 9 on Arduino.



Servo Motor: Servo Library

Go to File → Examples → Servo → Sweep



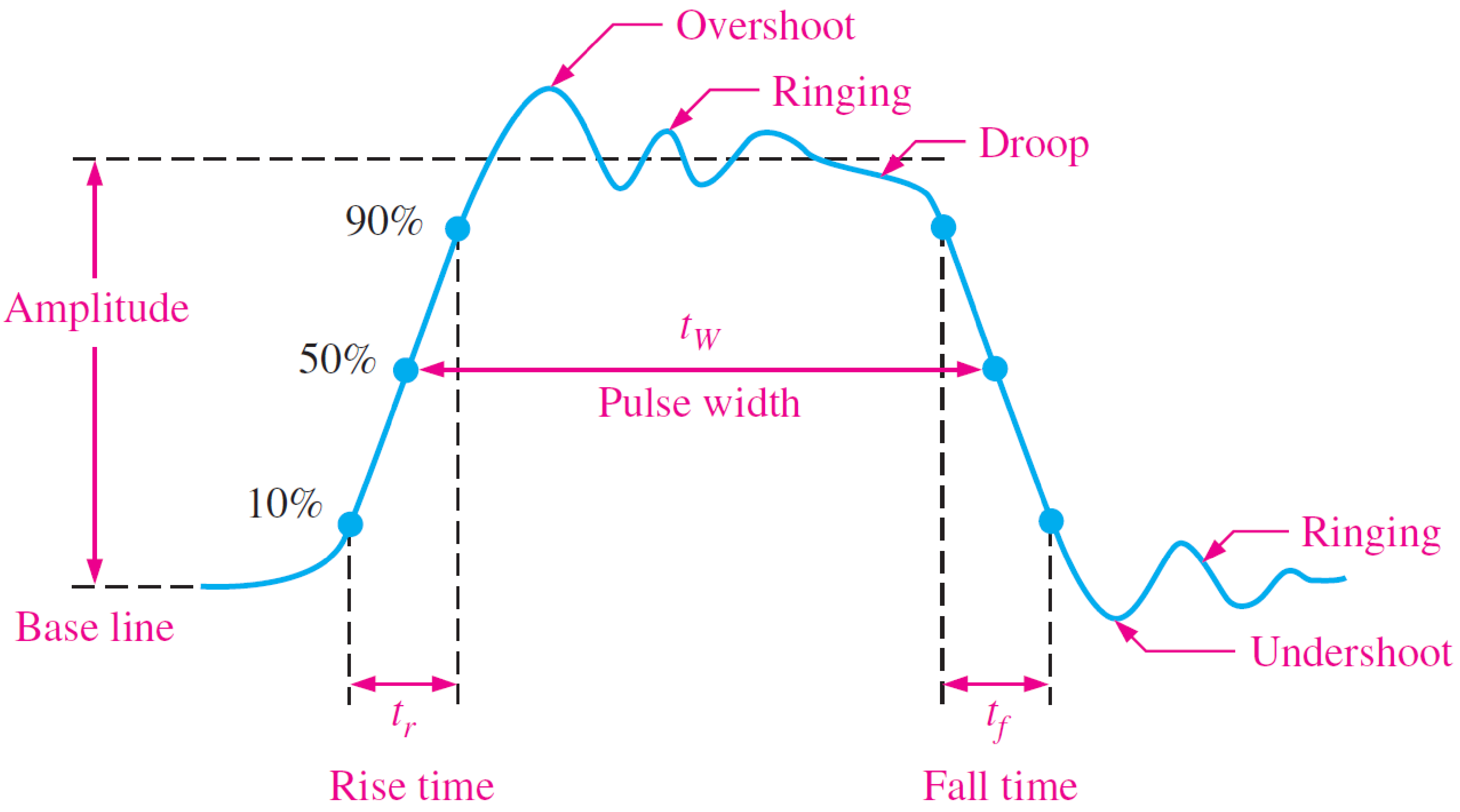
Servo Motor: Code

```
#include <Servo.h>                              // Include the Servo library

Servo myservo;                                  // Create servo object to control a servo
int pos = 0;                                    // Variable to store the servo position

void setup() {
  myservo.attach(9);                            // Attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) {         // Goes from 0 to 180 degrees in steps of 1 degree
    myservo.write(pos);                         // Tell servo to go to position in variable 'pos'
    delay(15);                                  // Waits 15 ms for the servo to reach the position
  }

  for (pos = 180; pos >= 0; pos -= 1) {         // Goes from 180 to 0 degrees in steps of 1 degree
    myservo.write(pos);                         // Tell servo to go to position in variable 'pos'
    delay(15);                                  // Waits 15 ms for the servo to reach the position
  }
}
```

Pulse Width Modulation (PWM)

- Pulse Width Modulation (PWM) is a technique for getting analog results with digital means.
- Digital control is used to create a square wave, a signal switched between LOW and HIGH.
- This LOW-HIGH pattern can simulate voltages in between the HIGH (5V) and LOW (0V) by changing the portion of the time the signal spends HIGH versus the time that the signal spends LOW.
- The duration of “HIGH time” is called the pulse width.

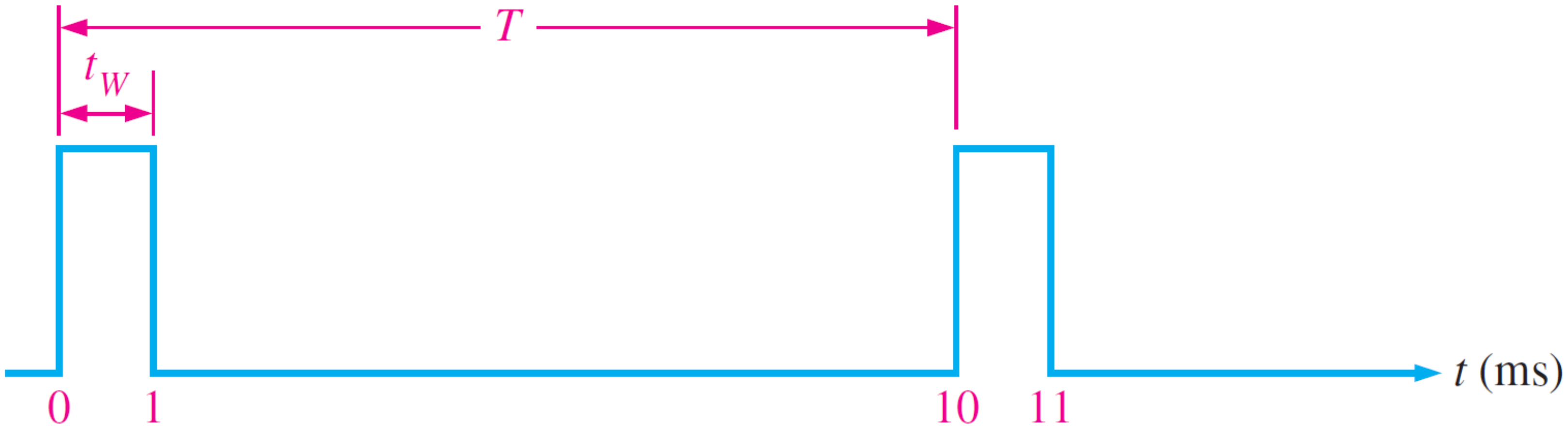
Pulse Width Modulation (PWM)



Pulse Width Modulation (PWM)

- An important characteristic of a periodic digital waveform is its duty cycle, which is the ratio of the pulse width (t_w) to the period (T).

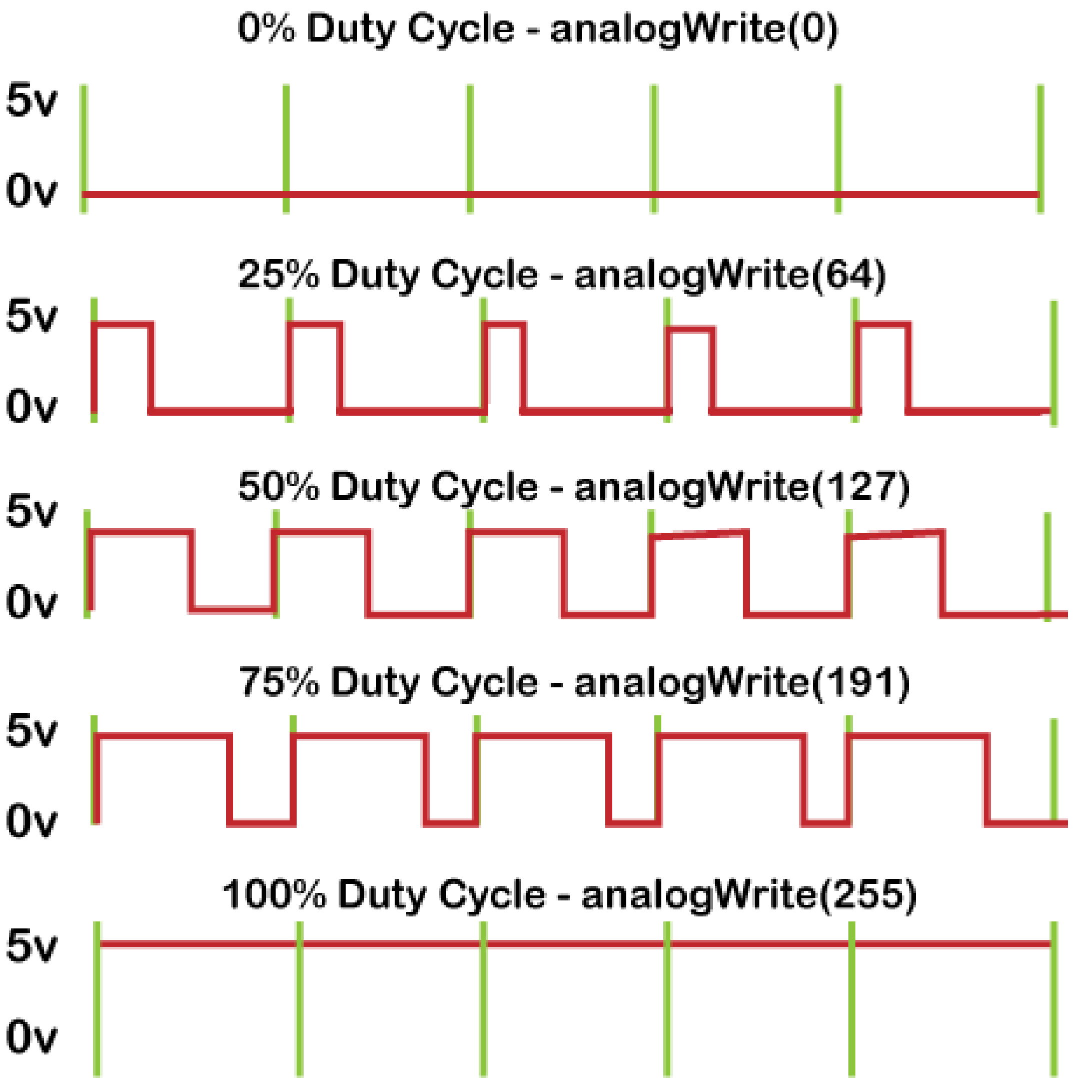
$$\textbf{Duty cycle} = \left(\frac{t_W}{T}\right)100\%$$



$$\text{Duty cycle} = \left(\frac{t_W}{T}\right)100\% = \left(\frac{1\text{ ms}}{10\text{ ms}}\right)100\% = \mathbf{10\%}$$

Pulse Width Modulation (PWM)

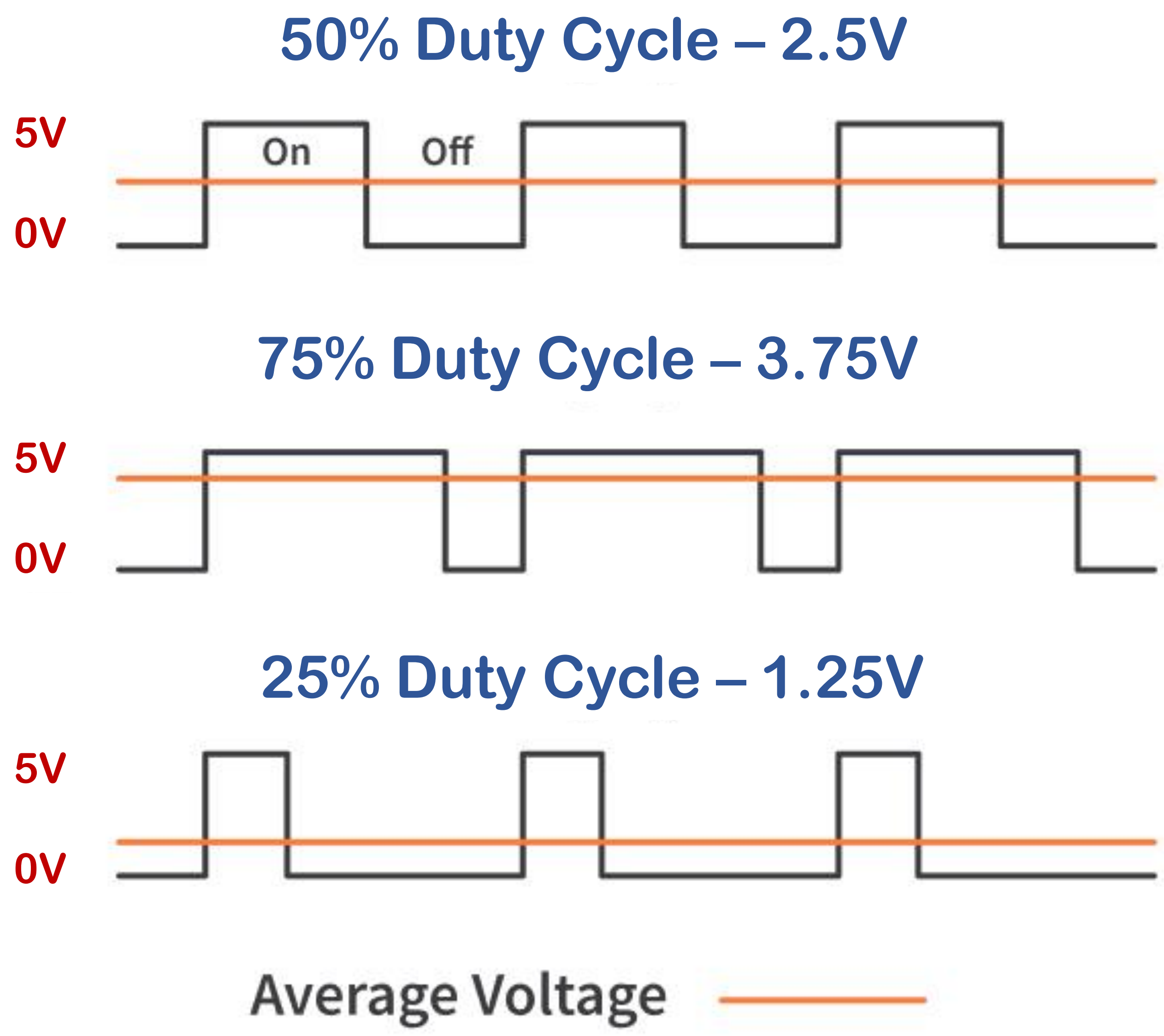
- To get varying analog values, you change, or modulate, that pulse width.



Pulse Width Modulation (PWM)

- The function `analogWrite()` writes an analog value (PWM wave) to pin.
- It can be used to light a LED at varying brightnesses or drive a motor at various speeds.
- The `analogWrite(0)` gives a signal of 0% duty cycle (0V output).
- The `analogWrite(50)` gives a signal of 20% duty cycle (1V output).
- The `analogWrite(63)` gives a signal of 25% duty cycle (1.25V output).
- The `analogWrite(127)` gives a signal of 50% duty cycle (2.5V output).
- The `analogWrite(255)` gives a signal of 100% duty cycle (5V output).

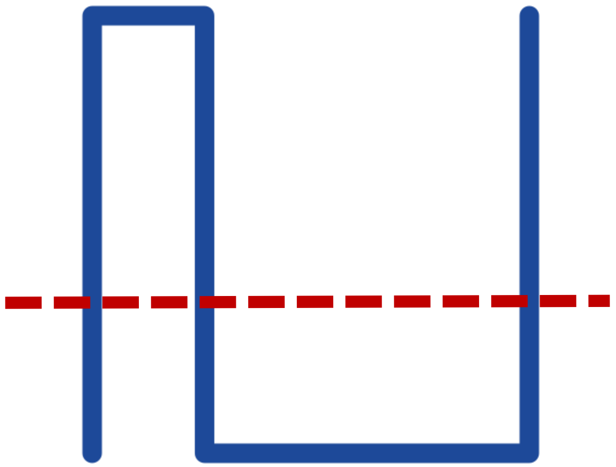
Pulse Width Modulation (PWM): Averaging Operator



Pulse Width Modulation (PWM): Averaging Operator

- If the signal $x(t)$ is periodic with period T_0, its time average can be computed as

$$\langle x(t) \rangle = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} x(t)\; dt$$



Pulse Width Modulation (PWM): 25% Duty Cycle

$$\langle x(t) \rangle = \frac{1}{T_0} \int_0^{T_0} x(t)\, dt$$

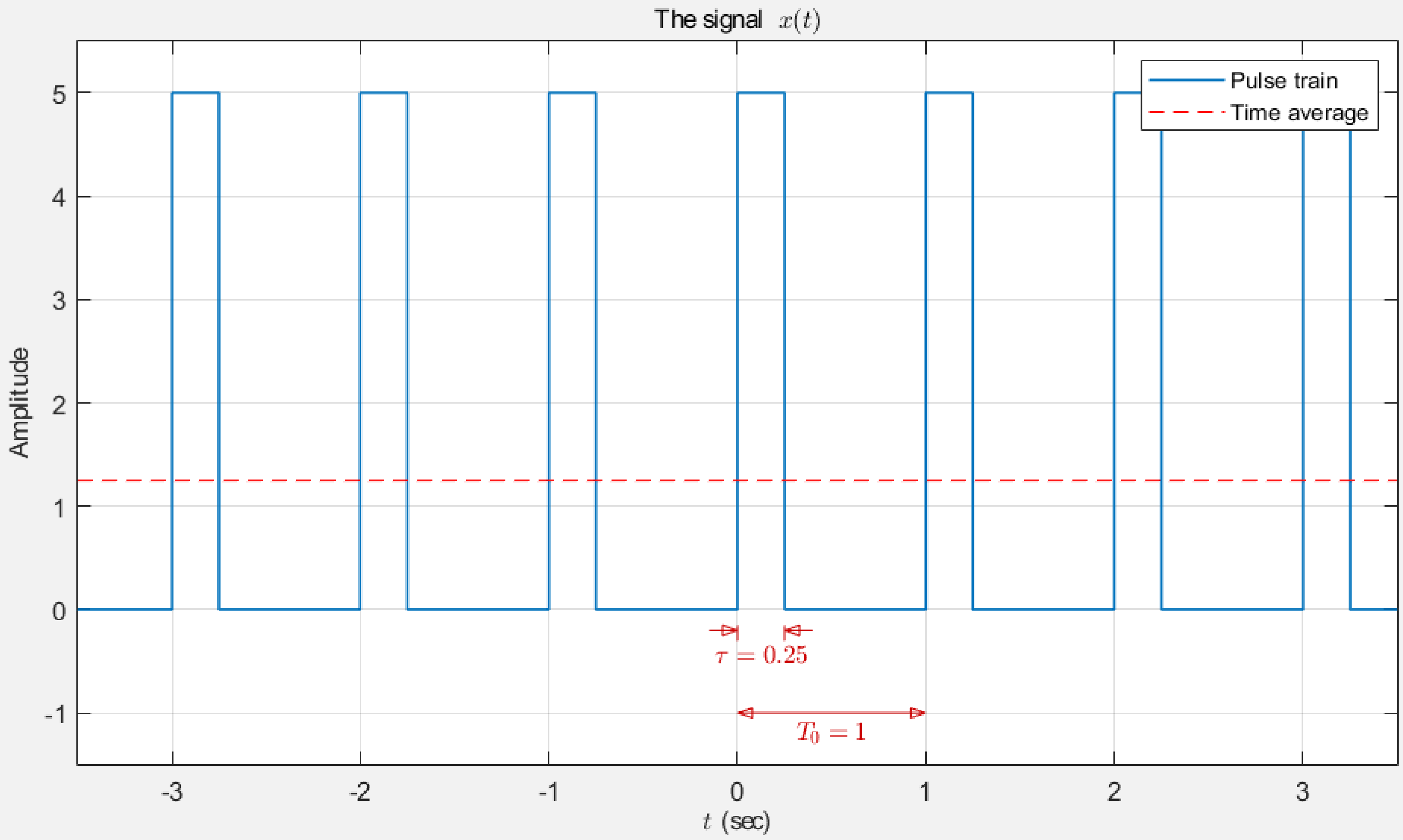
$$= A\,d = 1.25$$

Pulse amplitude (A): 5

Duty cycle (d): 0.25

Period (T0) in seconds: 1

☑ Display annotations



Pulse Width Modulation (PWM): 50% Duty Cycle

$$\langle x(t) \rangle = \frac{1}{T_0} \int_0^{T_0} x(t)\ dt$$

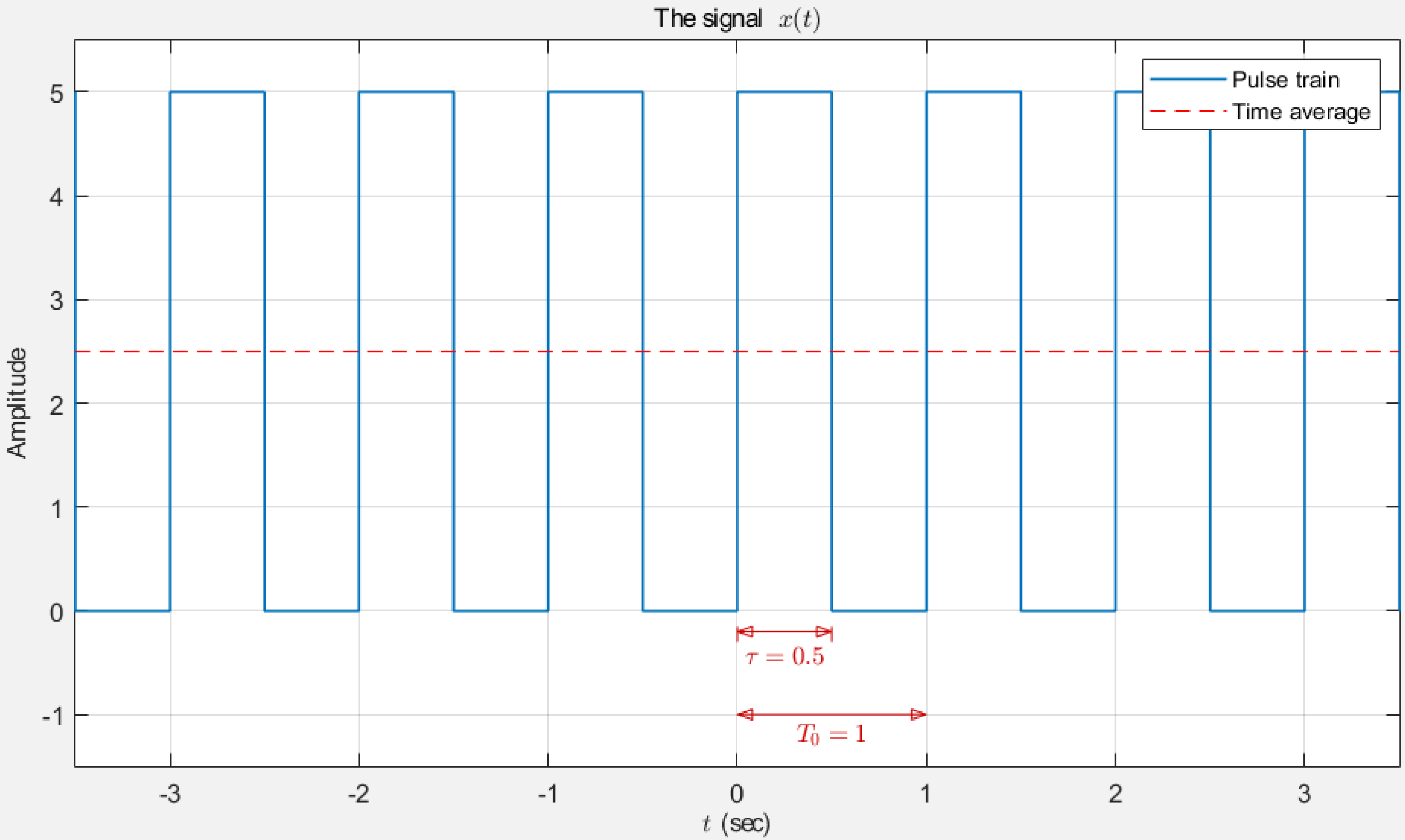
$$= A\,d = 2.5$$

Pulse amplitude (A): 5

Duty cycle (d): 0.5

Period (T0) in seconds: 1

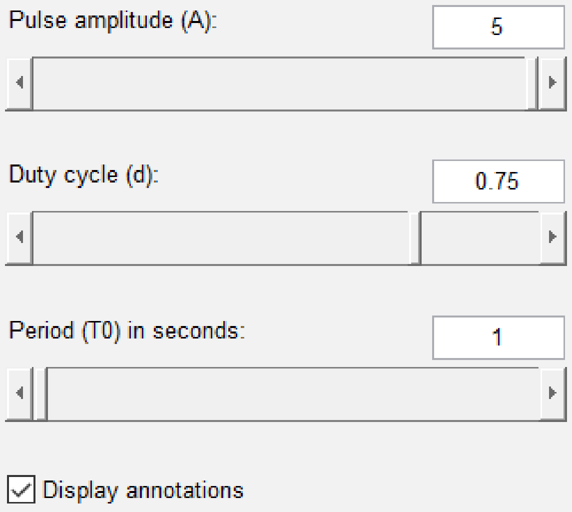
☑ Display annotations

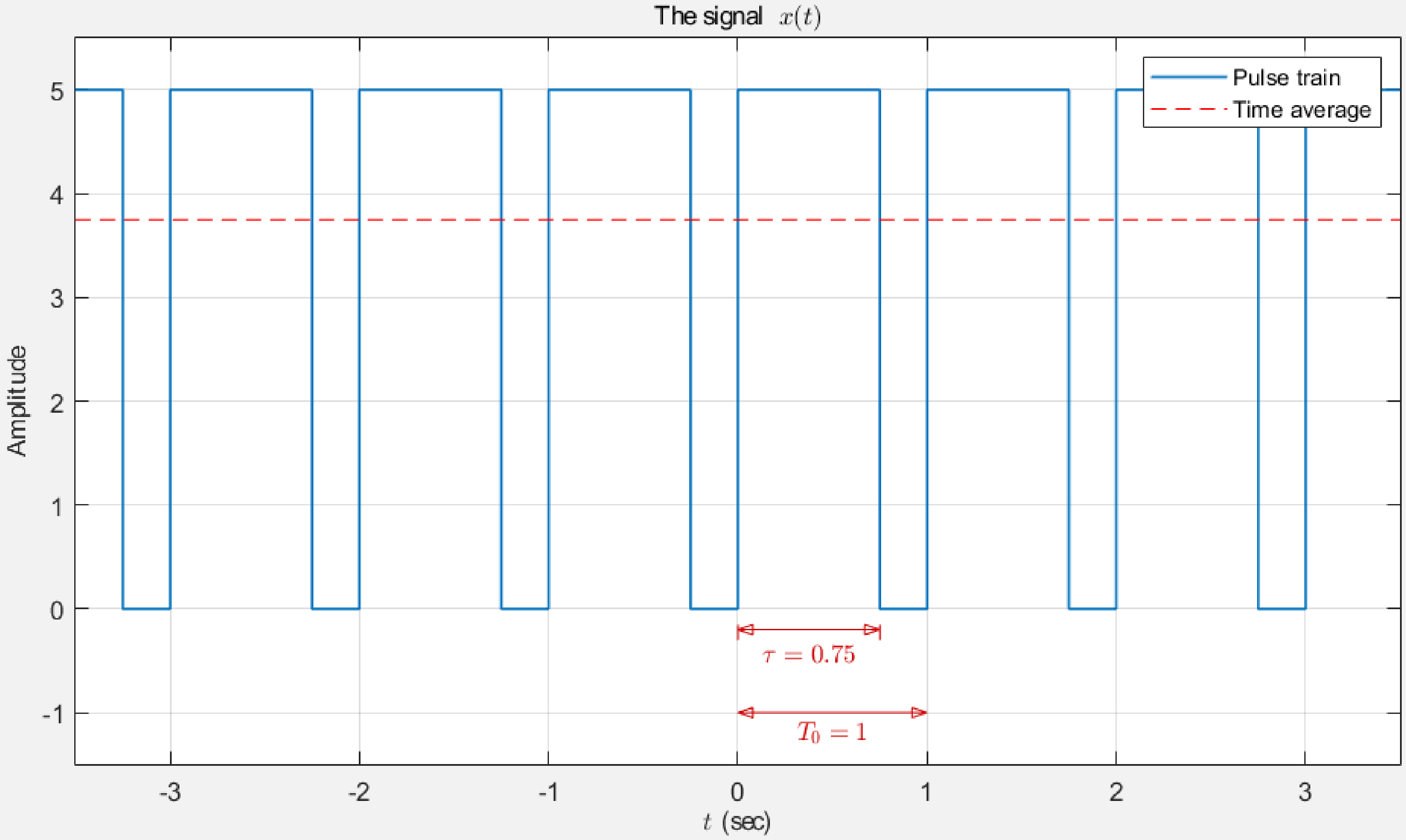


Pulse Width Modulation (PWM): 75% Duty Cycle

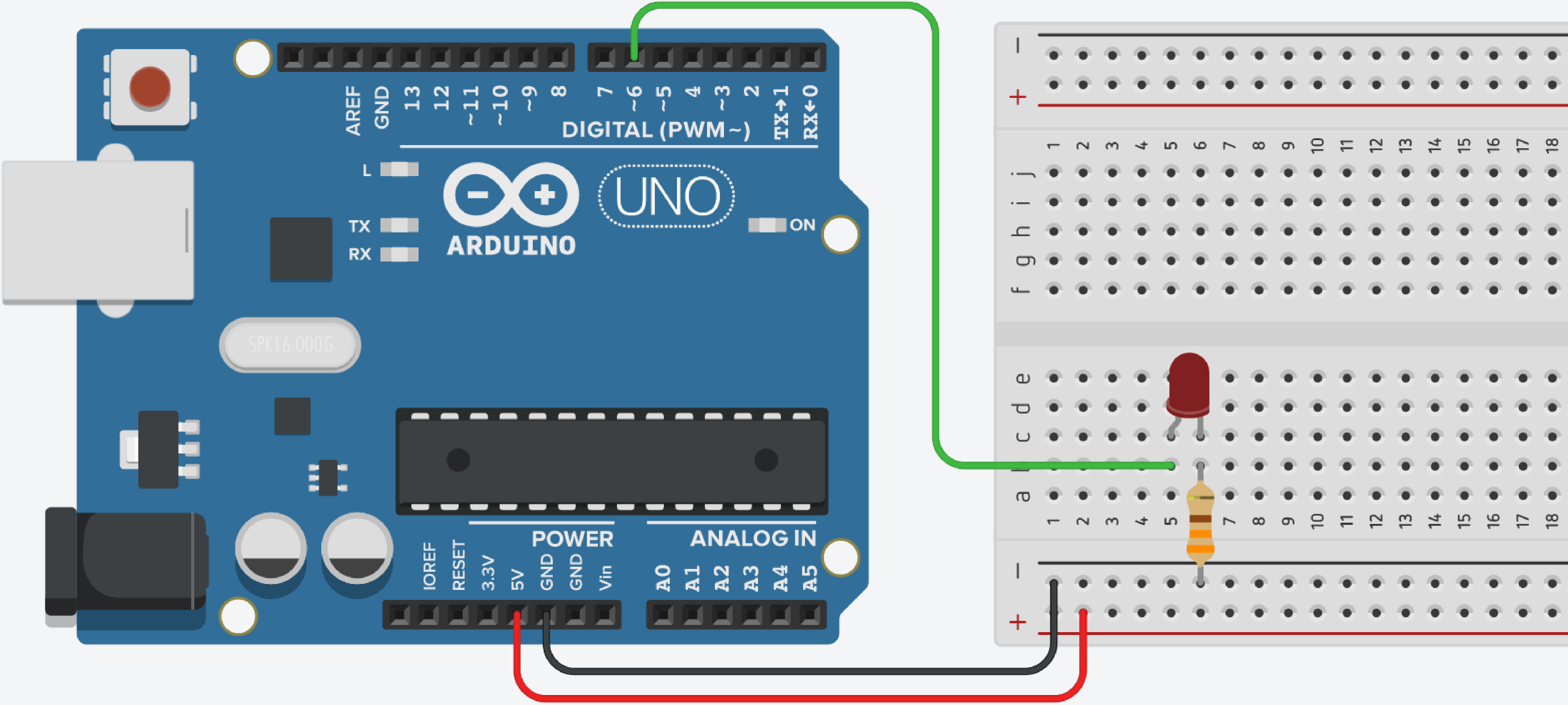
$$\langle x(t) \rangle = \frac{1}{T_0}\int_0^{T_0} x(t)\, dt$$

$$= A\,d = 3.75$$





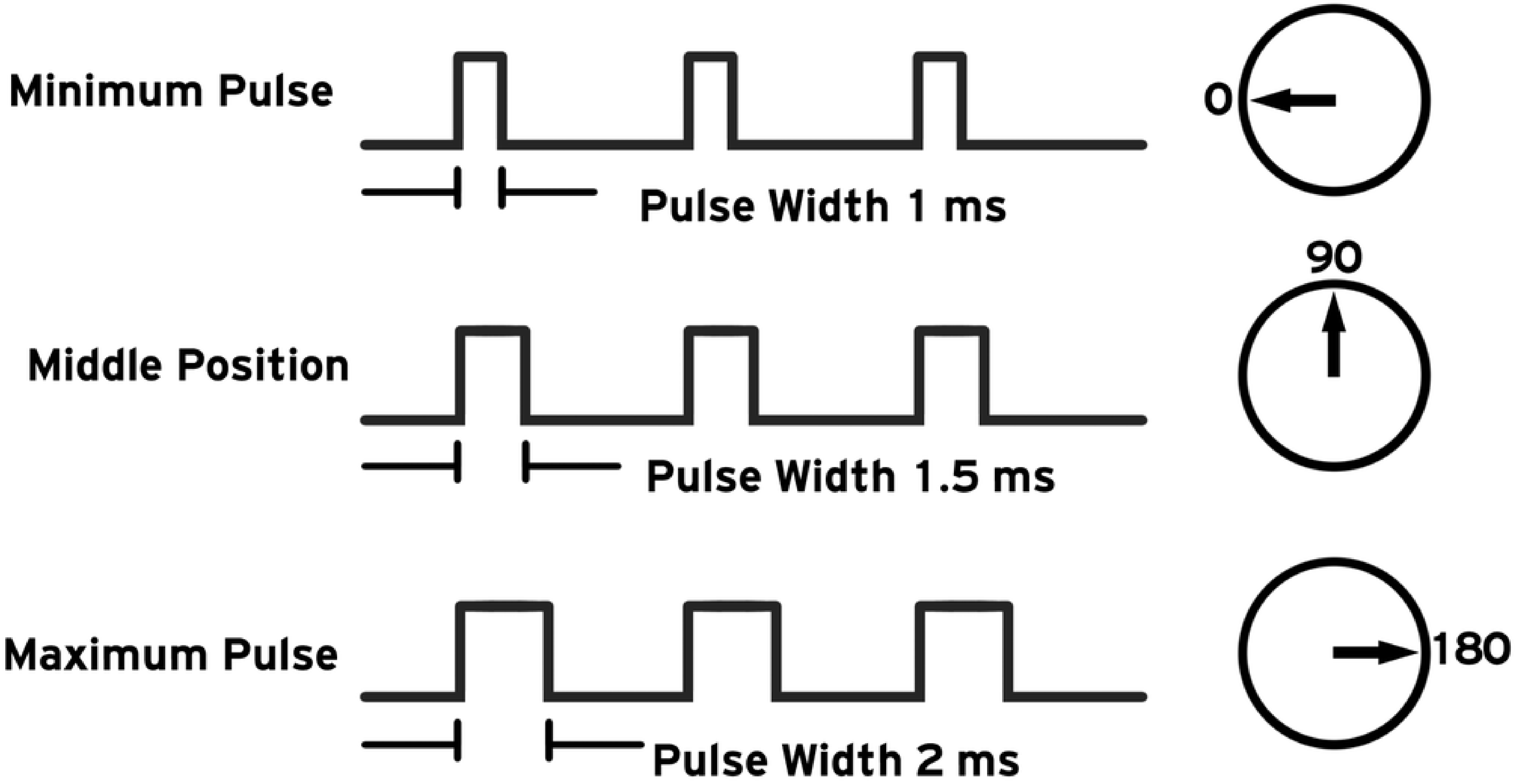
Pulse Width Modulation (PWM): LED Brightness



Pulse Width Modulation (PWM): LED Brightness

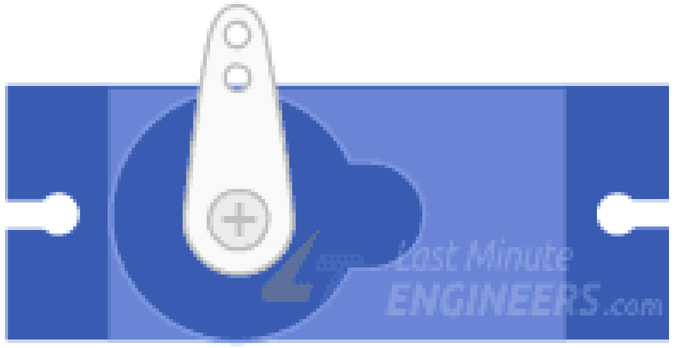
```
#define LED 6                                       // LED pin (~)

void setup() {
  pinMode(LED, OUTPUT);                             // Set pin 6 as an output
}

void loop() {
  // Increase the brightness of LED from zero brightness to the fullest
  for(int i = 0; i <= 255; i++)
  {
    analogWrite(LED, i);                            // PWM
    delay(10);                                      // Short delay
  }

  // Decrease the brightness of the LED from the fullest to the off state
  for(int i = 255; i >= 0; i--)
  {
    analogWrite(LED, i);                            // PWM
    delay(10);                                      // Short delay
  }
}
```

Pulse Width Modulation (PWM): Servo Motor



Pulse Width Modulation (PWM): Servo Motor

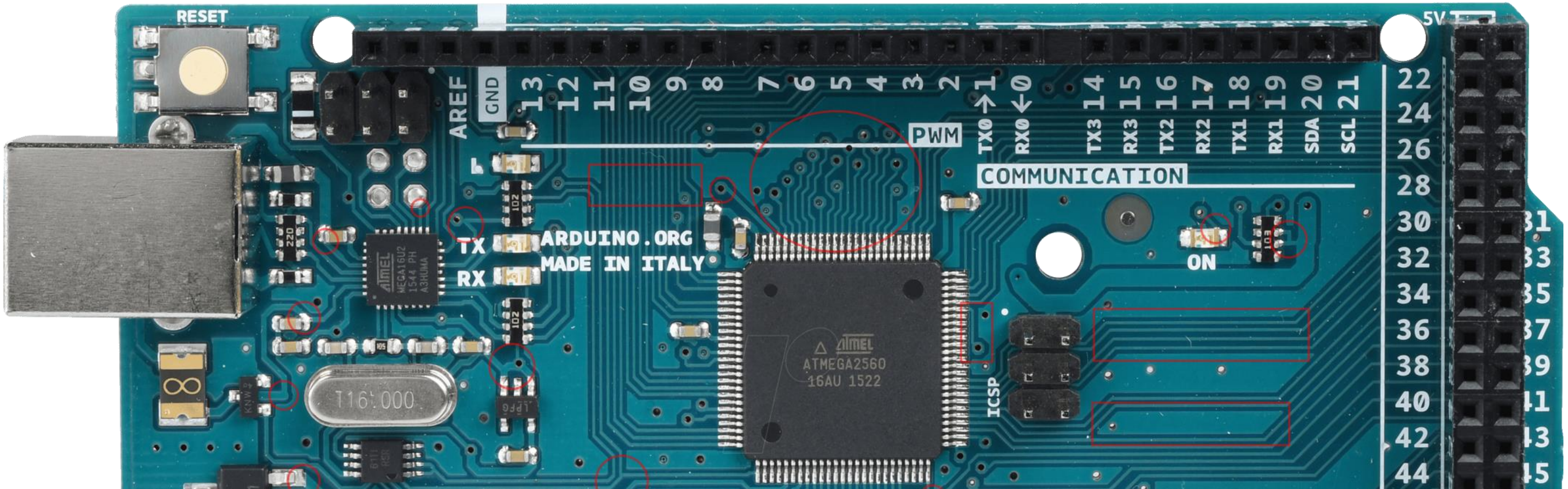
- Servo motor can be rotated from 0 to 180 degrees.
- This degree of rotation can be controlled by applying the electrical pulse of proper width, to its control pin.
- The pulse of 1 millisecond width can rotate the servo to 0 degrees.
- The pulse of 1.5 milliseconds width can rotate it to 90 degrees.
- The pulse of 2 milliseconds width can rotate it to 180 degrees.



0 Degrees

Pulse Width Modulation (PWM): Arduino Pins

- Arduino has a limited number of pins that can be used for PWM output.
- On the Arduino Uno and compatible boards based on the ATmega328, you can use pins 3, 5, 6, 9, 10, and 11.
- On the Arduino Mega board, you can use pins 2 through 13 and 44 through 46 for PWM output.

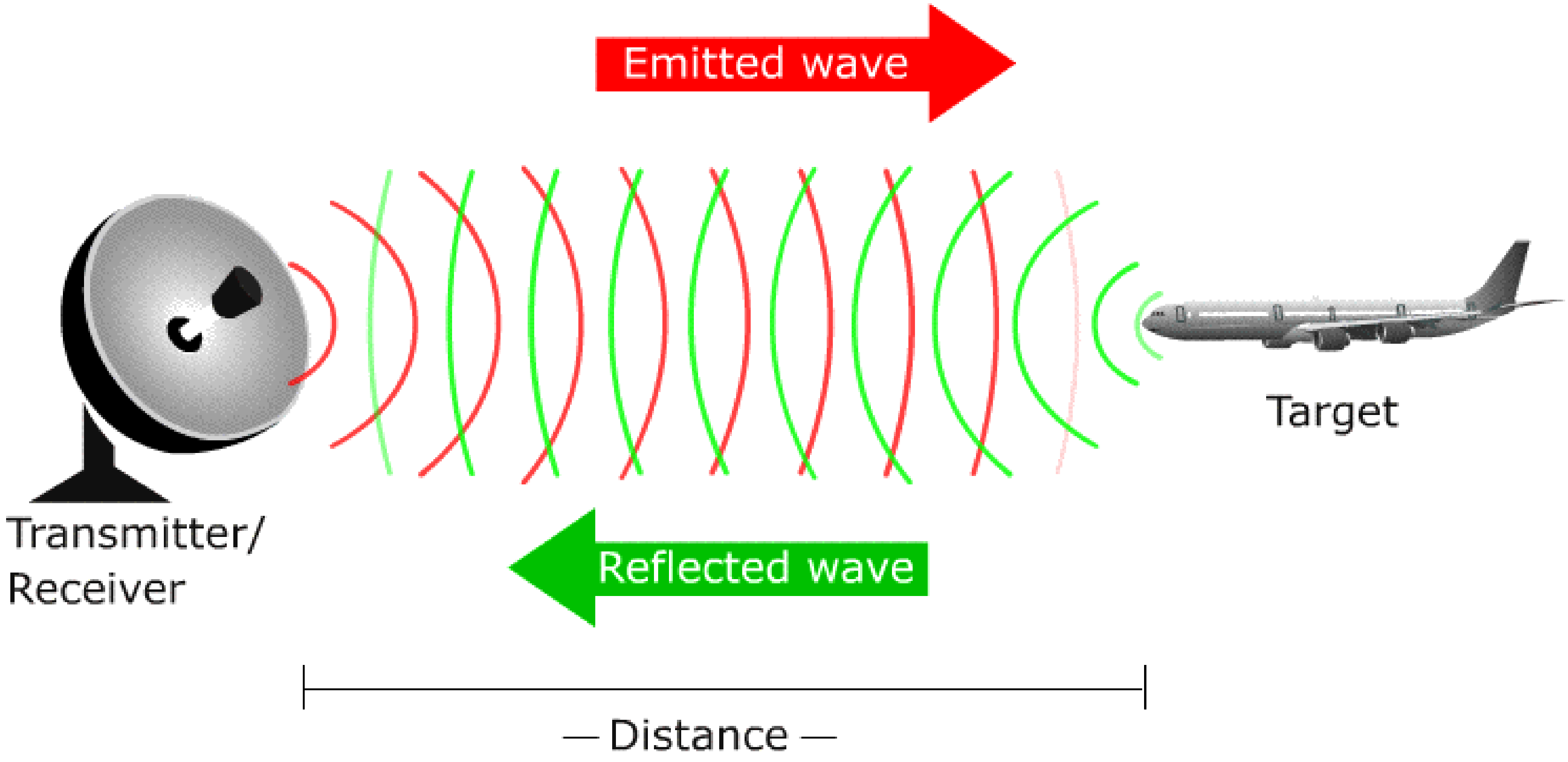


Radar

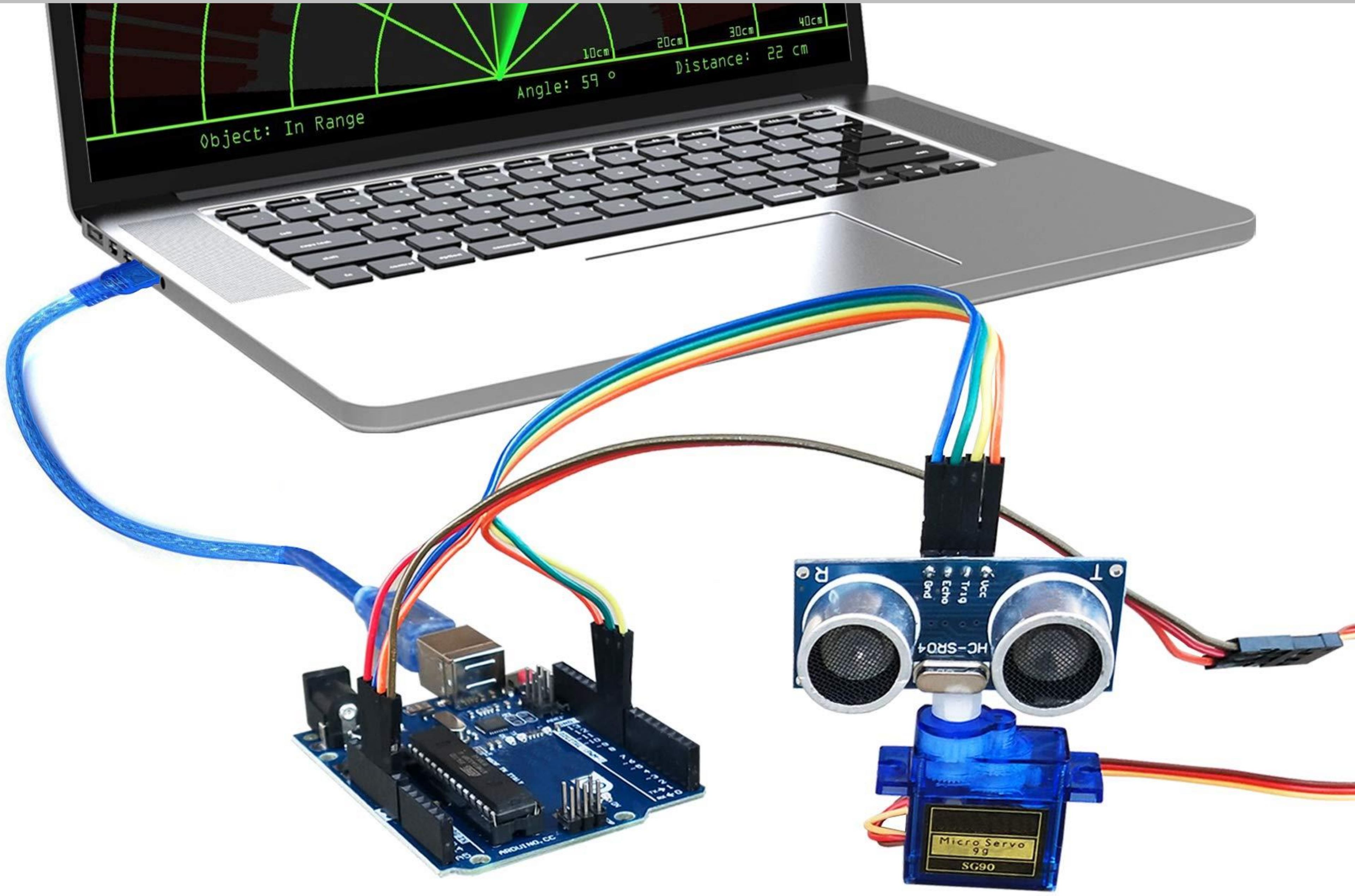
- The word RADAR means Radio Detection and Ranging.
- Radar is an object detection system that uses microwaves to determine the range, altitude, direction, and speed of objects within about a 100-mile radius of their location.

Radar

- The radar antenna transmits radio waves or microwaves that bounce off any object in its path.
- Due to this, we can easily determine the object in the radar range.

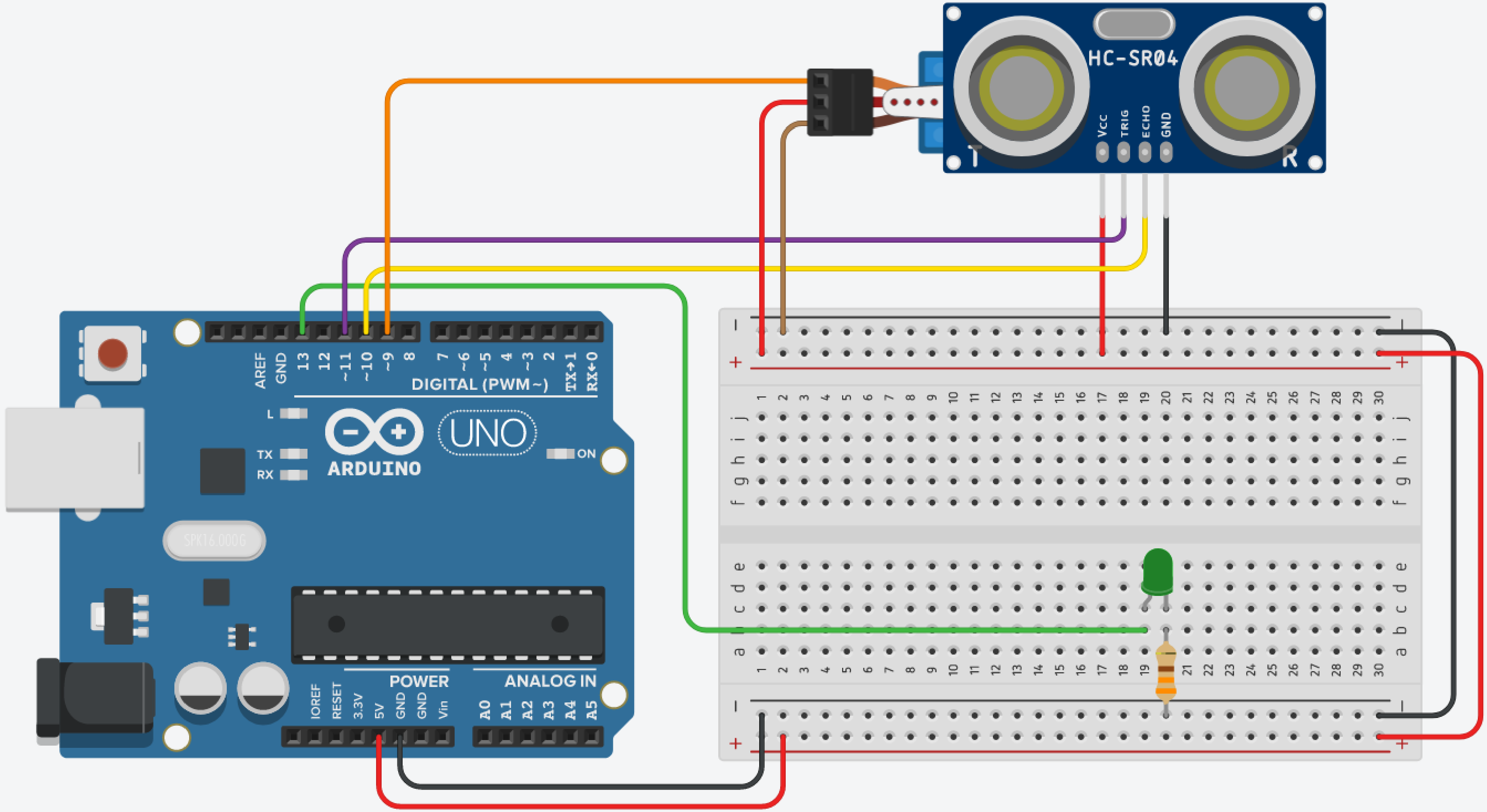


Radar System Prototype



Radar System Prototype: Circuit

- We will use the same connections.



Radar System Prototype: Code

```
#include <Servo.h>                  // Include the Servo library

#define LED_PIN 13                  // Flash LED pin
#define TRIG_PIN 11                 // Trigger pin of the ultrasonic sensor
#define ECHO_PIN 10                 // Echo pin on the ultrasonic sensor
#define SERVO_PIN 9                 // Servo signal pin

Servo myservo;                      // Create servo object to control a servo
int pos;                            // Variable to store the servo position

long t;                             // Variable to hold the time
int distance;                       // Variable to hold the distance

void setup() {
  Serial.begin(9600);               // Begin serial communication
  myservo.attach(SERVO_PIN);        // Attaches the servo on pin 9

  pinMode(LED_PIN, OUTPUT);         // Set LED_PIN as an output
  pinMode(TRIG_PIN, OUTPUT);        // Set TRIG_PIN as an output
  pinMode(ECHO_PIN, INPUT);         // Set ECHO_PIN as an input
}
```

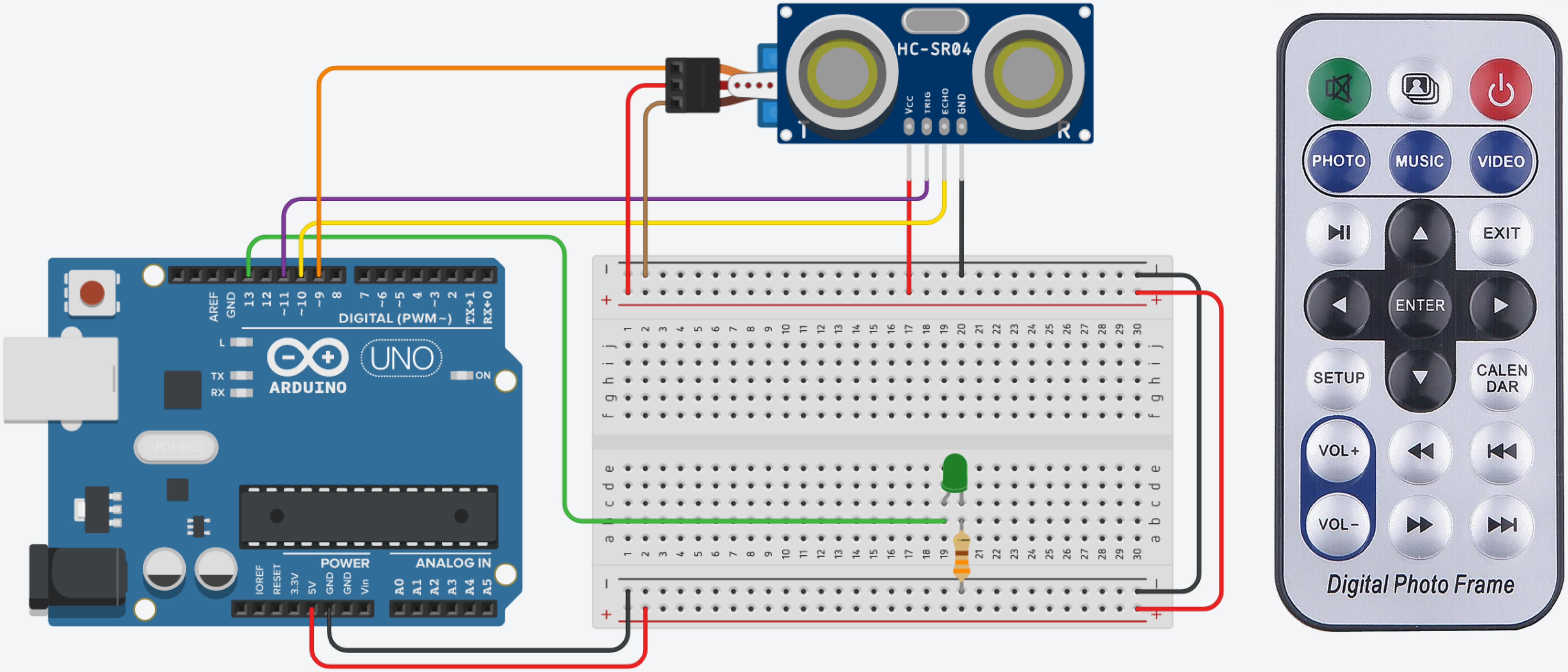
Radar System Prototype: Code

```
void loop() {
  for (pos = 0; pos <= 180; pos += 1) {       // Goes from 0 to 180 degrees
    myservo.write(pos);                       // Tell servo to go to position
    delay(100);                               // Short delay
    radar();                                  // Call radar function
  }

  for (pos = 180; pos >= 0; pos -= 1) {       // Goes from 180 to 0 degrees
    myservo.write(pos);                       // Tell servo to go to position
    delay(100);                               // Short delay
    radar();                                  // Call radar function
  }
}

// Calculate distance function
int calculate_distance(){
  digitalWrite(TRIG_PIN, LOW);                // Make sure that TRIG_PIN is LOW
  delayMicroseconds(2);                       // for just 2 microseconds

  digitalWrite(TRIG_PIN, HIGH);               // Set the TRIG_PIN to HIGH
  delayMicroseconds(10);                      // for 10 microseconds
  digitalWrite(TRIG_PIN, LOW);                // Set the TRIG_PIN to LOW

  t = pulseIn(ECHO_PIN, HIGH);                // Return the length of pulse in microseconds
  return (0.5 * t * 0.0343);                  // Return the distance (D = 0.5T * S)
}
```

Radar System Prototype: Code

```
// Radar function
void radar(){
  distance = calculate_distance();         // Calculate the distance
  Serial.println(distance);                // Print the distance

  if(distance >= 10 && distance <= 20){    // If distance is between 10cm and 20cm
    digitalWrite(LED_PIN, HIGH);           // Object is detected!
    delay(100);                            // Blink the LED every 100 seconds
    digitalWrite(LED_PIN, LOW);
    delay(100);
  }
  else if(distance < 10){                  // If distance is less than 10cm
    digitalWrite(LED_PIN, HIGH);           // Object is detected!
    delay(20);                             // Blink the LED every 20 seconds
    digitalWrite(LED_PIN, LOW);
    delay(20);
  }
  else
    digitalWrite(LED_PIN, LOW);            // If no object is detected, Turn off LED
}
```

Assignment 04: Control Radar System Using IR Remote



References

- How HC-SR04 Ultrasonic Sensor Works
- Ultrasonic Sensor HC-SR04 and Arduino
- Complete Guide for Ultrasonic Sensor HC-SR04
- How To Control Servo Motor Using Arduino
- How to Control Servo Motors with Arduino
- How Servo Motor Works & Interface It With Arduino
- Arduino Servo Motors
- Basics of PWM (Pulse Width Modulation)
- analogWrite()
- Arduino - PWM