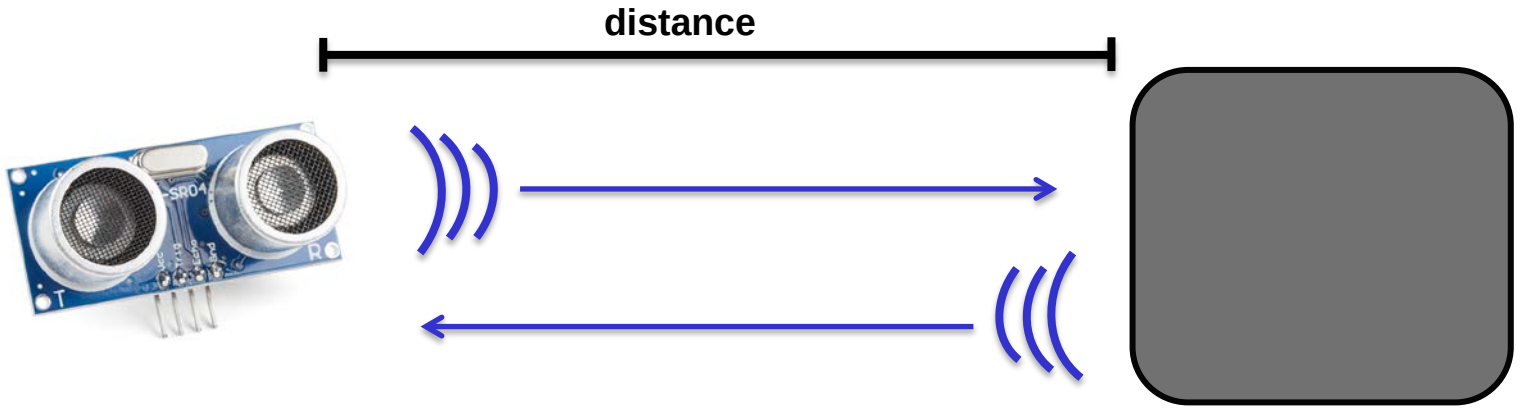
Engineering Technology (ENGR 101)

Arduino and Ultrasonic Distance Sensors

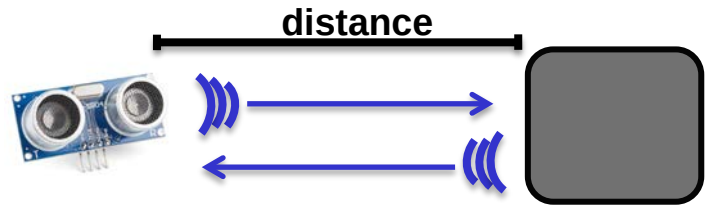
Ultrasonic Distance Sensor

- 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 (10 cm to 3 m).
- Most ultrasonic distance sensors can use a digital electrical pulse to calculate the distance of an object



Timing Diagram

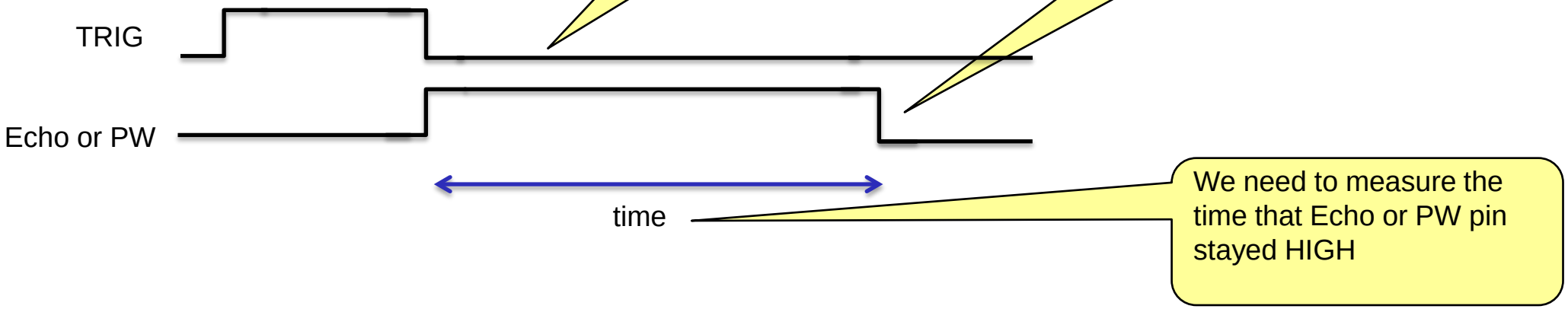
- There are two pins on the sensor
  - Trigger pin: We can control it to tell the sensor it should send out a sound pulse.
  - Echo or pulse width (PW) pin: we measure the electrical pulse width on the PW pin to determine the distance to the object.



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.

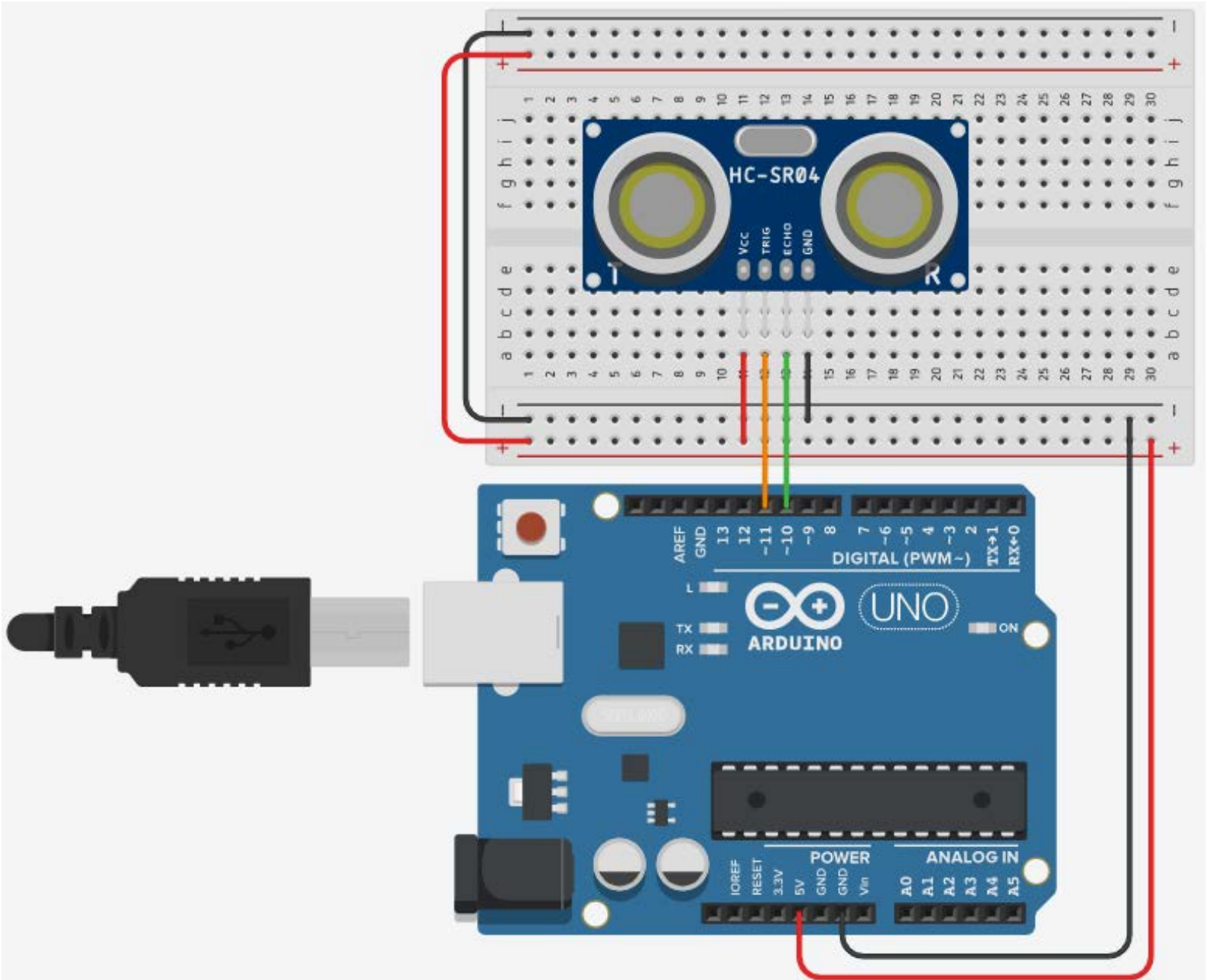
$$\frac{time}{29.4\mu s/cm} = 2 \times distance$$

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

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 = 10;
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.