

# **ENGR 3421: Robotics I**

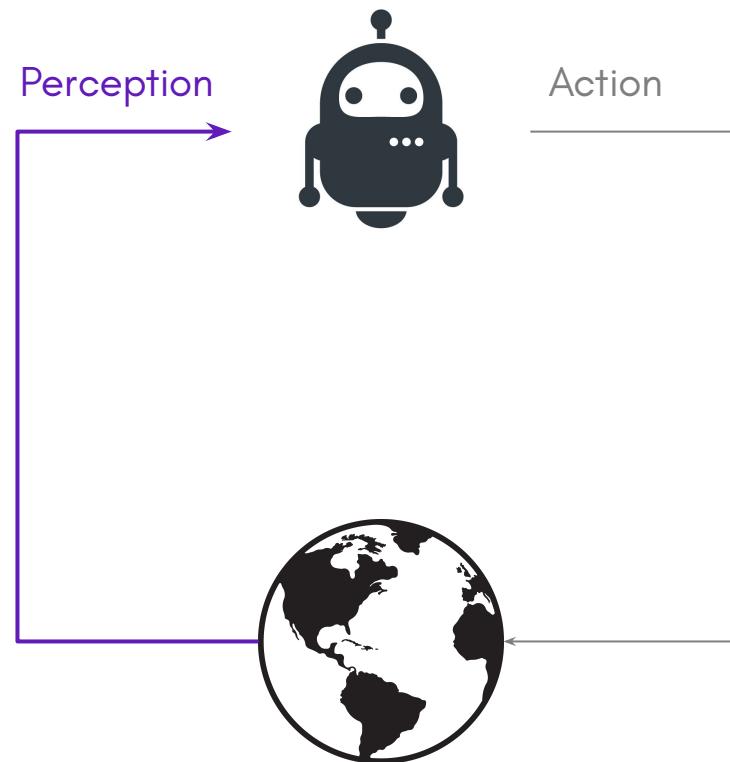
**Ultrasonic Distance Sensor**

**09/25/2025**

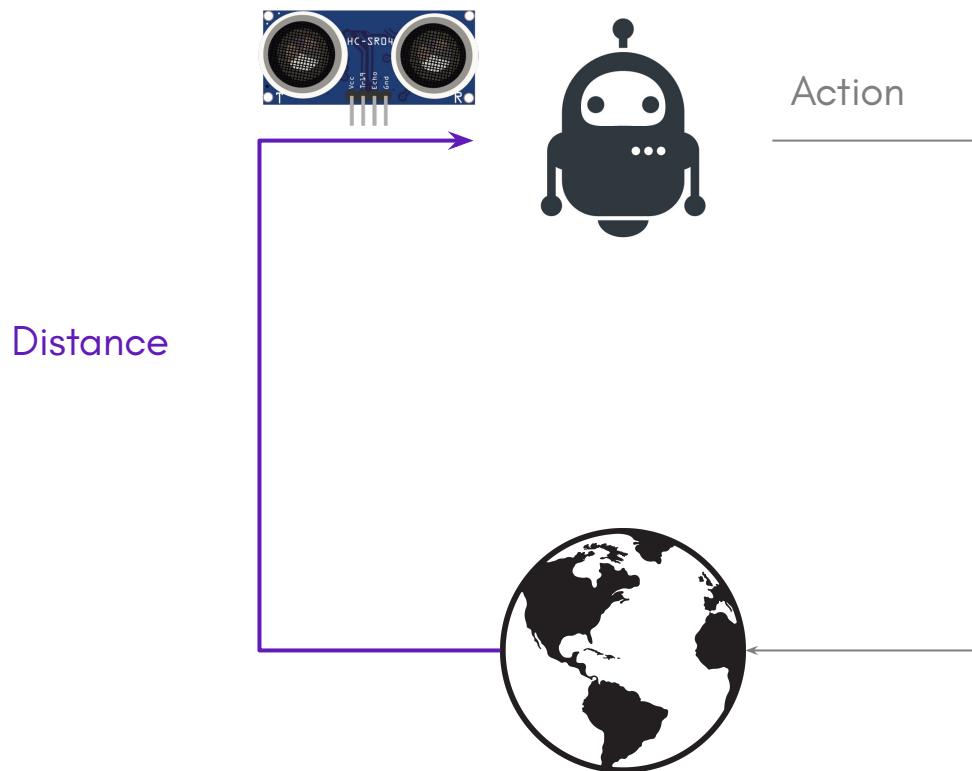
# **Outline**

- Ultrasound
- HC-SR04

# A Robot Needs to Feel

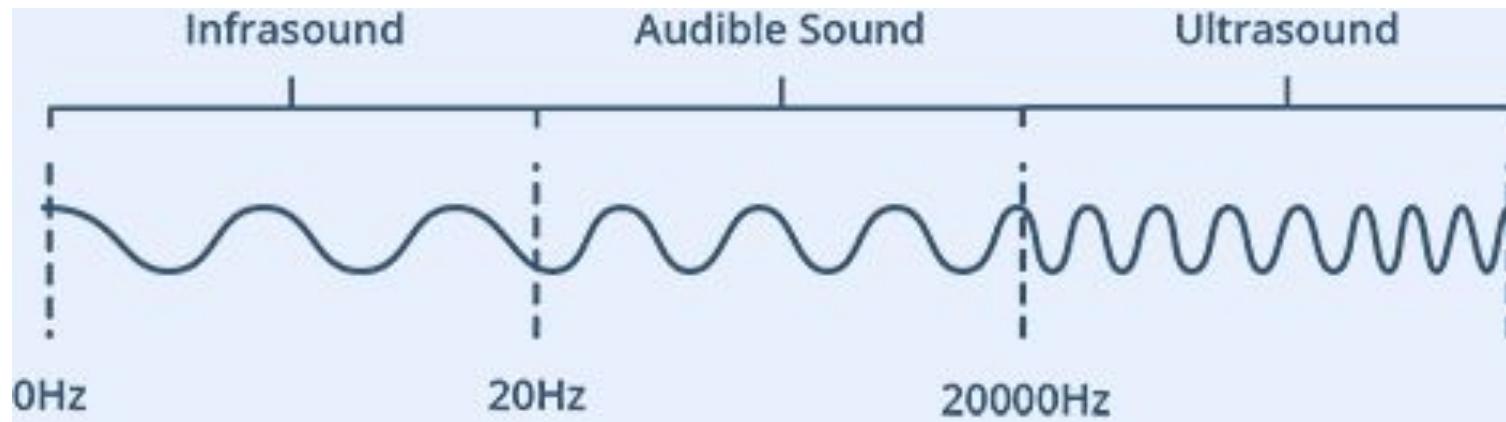


# A Robot Needs to Sense Distance

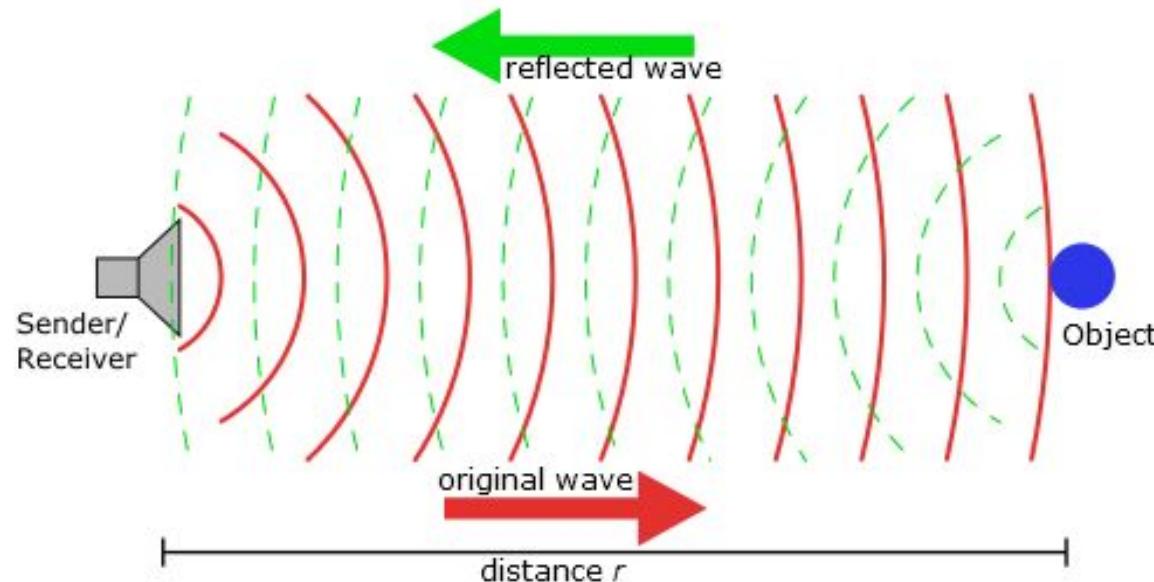


# Ultrasound

Ultrasound is high-pitched sound waves with frequencies higher than the audible limit of human hearing.



# Ultrasound Distance Sensing



$$\text{distance} = \frac{\text{speed} \times \text{time}}{2}$$

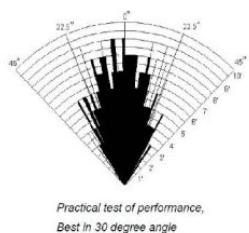
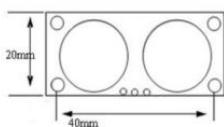
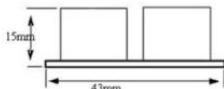
$$r = \frac{v \times t}{2}, \text{ in air: } v = 340 \text{ m/s}$$

# **HC-SR04 Ultrasonic Distance Sensor**

- Consists of a transmitter and a receiver.
- Transmitter broadcasts ultrasound at 40kHz.
- Receiver listens to the transmitted ultrasonic waves.

# HC-SR04

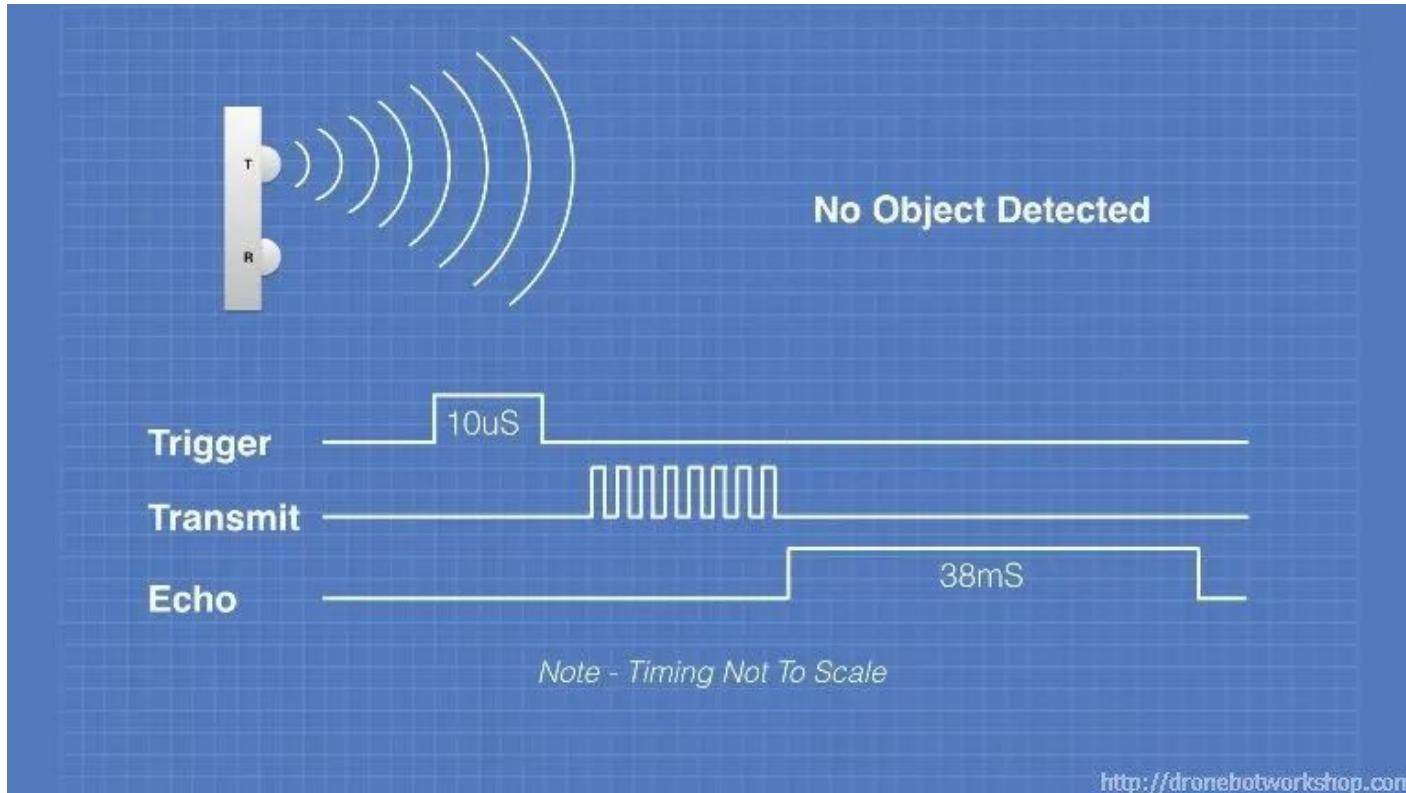
## HC-SR04 Pinout



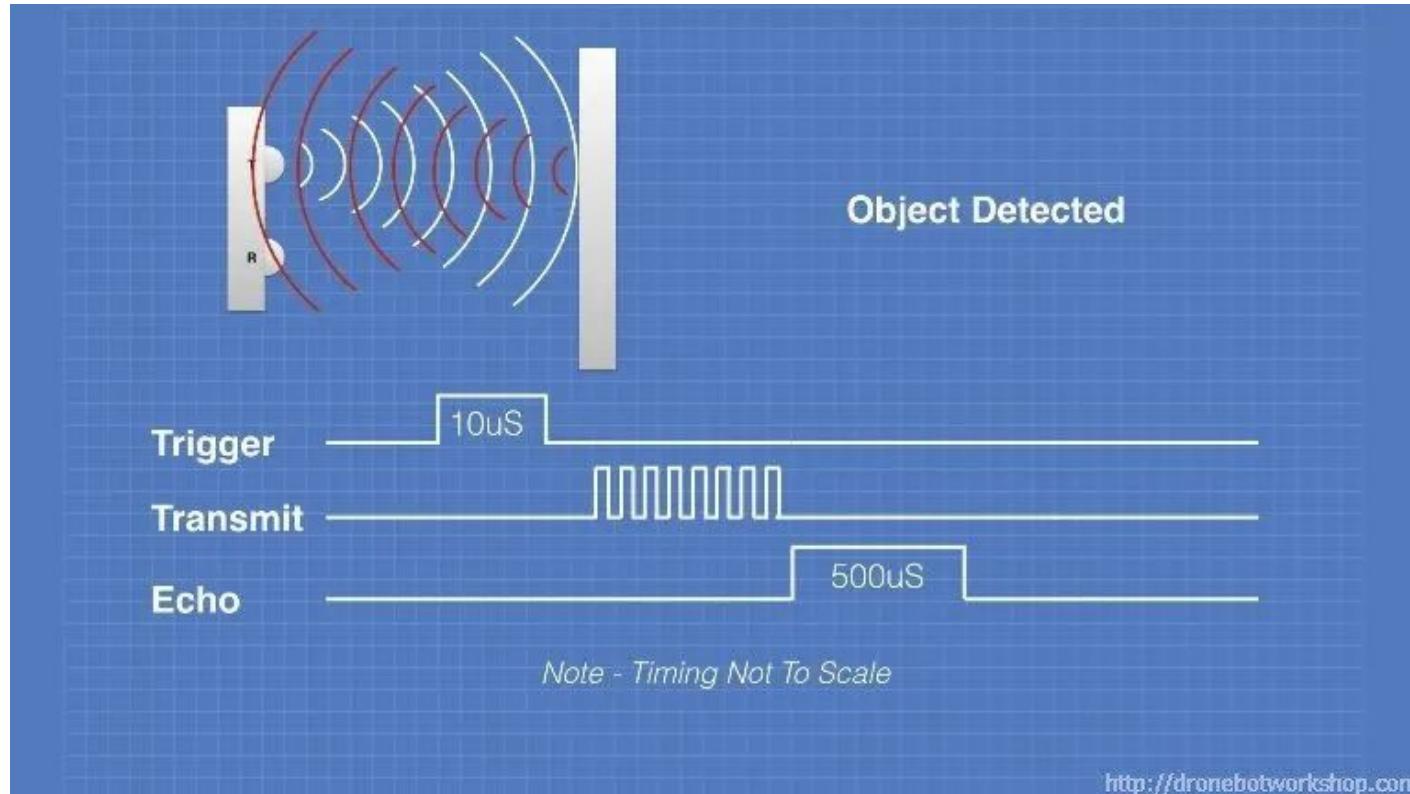
Operating Voltage	5V
Operating Current	15mA
Ultrasound Frequency	40kHz
Max. Linear Range	4 m
Min. Linear Range	0.02 m
Measuring Angle	15 deg
Measuring Accuracy	3 mm

<https://www.amazon.com/dp/B07YXX52SC/>

# HC-SR04 Detection



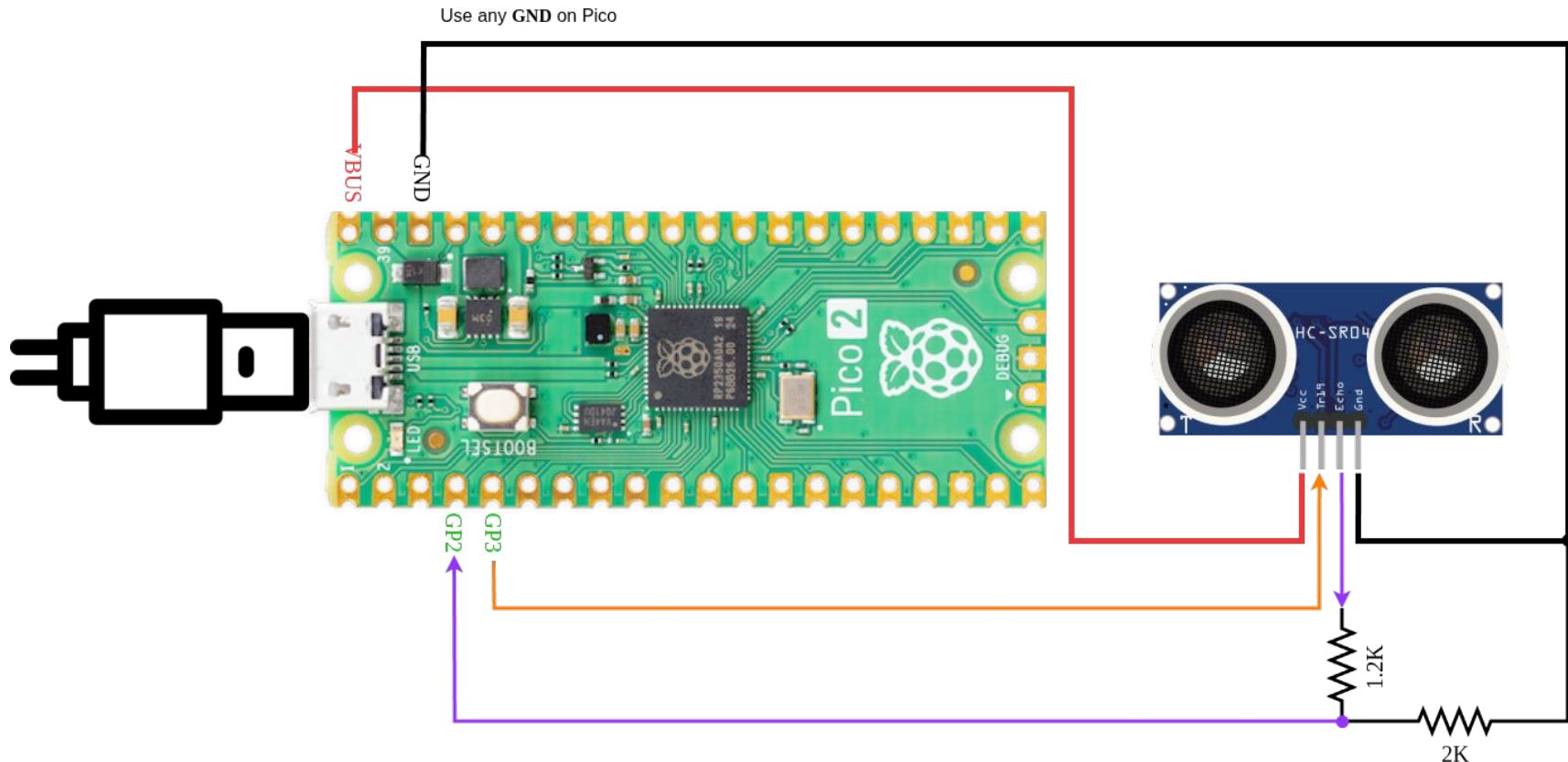
# HC-SR04 Detection



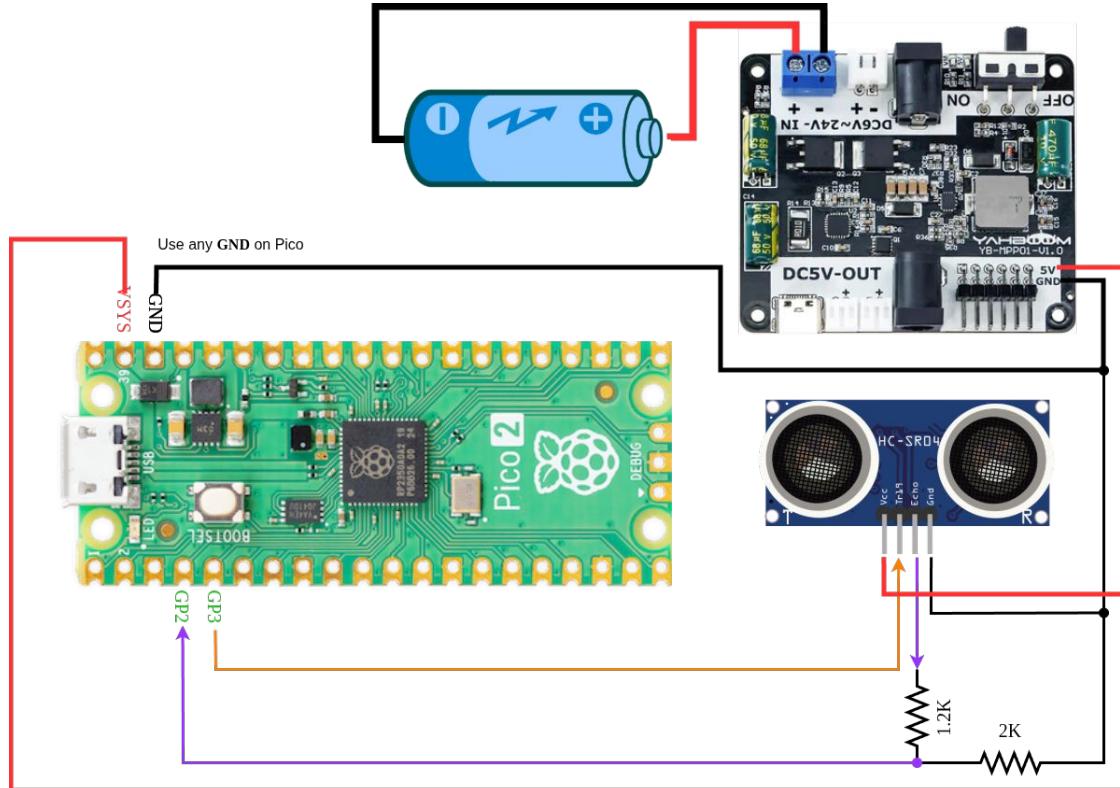
# HC-SR04 Workflow

1. Send a 10 microseconds pulse at 5 volt to the “Trigger” pin.
2. The transmitter bursts of 8 pulses at 40 KHz. This 8-pulse pattern makes the “ultrasonic signature” from the device unique, allowing the receiver to discriminate between the transmitted pattern and the ultrasonic background noise.
3. As soon as the 8-pulse ultrasonic wave is transmitted, the “Echo” pin goes high.
4. If the receiver DOES NOT hear the 8-pulse signal. The “Echo” pin goes low after 38 milliseconds.
5. If the 8-pulse signal is received before the Echo signal timed out, the “Echo” pin goes low immediately. This produces a pulse whose width varies between 150 uS to 25 mS.
6. The width of the received pulse is used to calculate the distance to the reflected object.

# Wiring - Powered by USB



# Wiring - Powered by Voltage Regulator



# Hand-Crafted Examples

```
from machine import Pin                                # LOOP
from time import sleep_us, sleep_ms, ticks_us

# SETUP
trig = Pin(3, Pin.OUT)
echo = Pin(2, Pin.IN, Pin.PULL_DOWN)
# Config IRQ
tic, toc, distance = None, None, None
def resolve_status(pin):
    global tic, toc, distance
    if pin.value():
        tic = ticks_us()
    else:
        dt = ticks_us() - tic
        if dt < 100:
            distance = 0.
        elif 100 <= dt < 38000:
            distance = dt / 58 / 100
        else:
            distance = None
echo.irq(handler=resolve_status)

while True:
    trig.on()
    sleep_us(10)
    trig.off()  # 10 us pulse to trigger
    print(f"distance: {distance} m")
    sleep_ms(60)
```

# Picozero Examples

```
from picozero import DistanceSensor
from time import sleep

ds = DistanceSensor(echo=2, trigger=3)

while True:
    print(ds.distance)
    sleep(0.1)
```