MIT AI2 204

IoT with MIT App Inventor

Fundamental

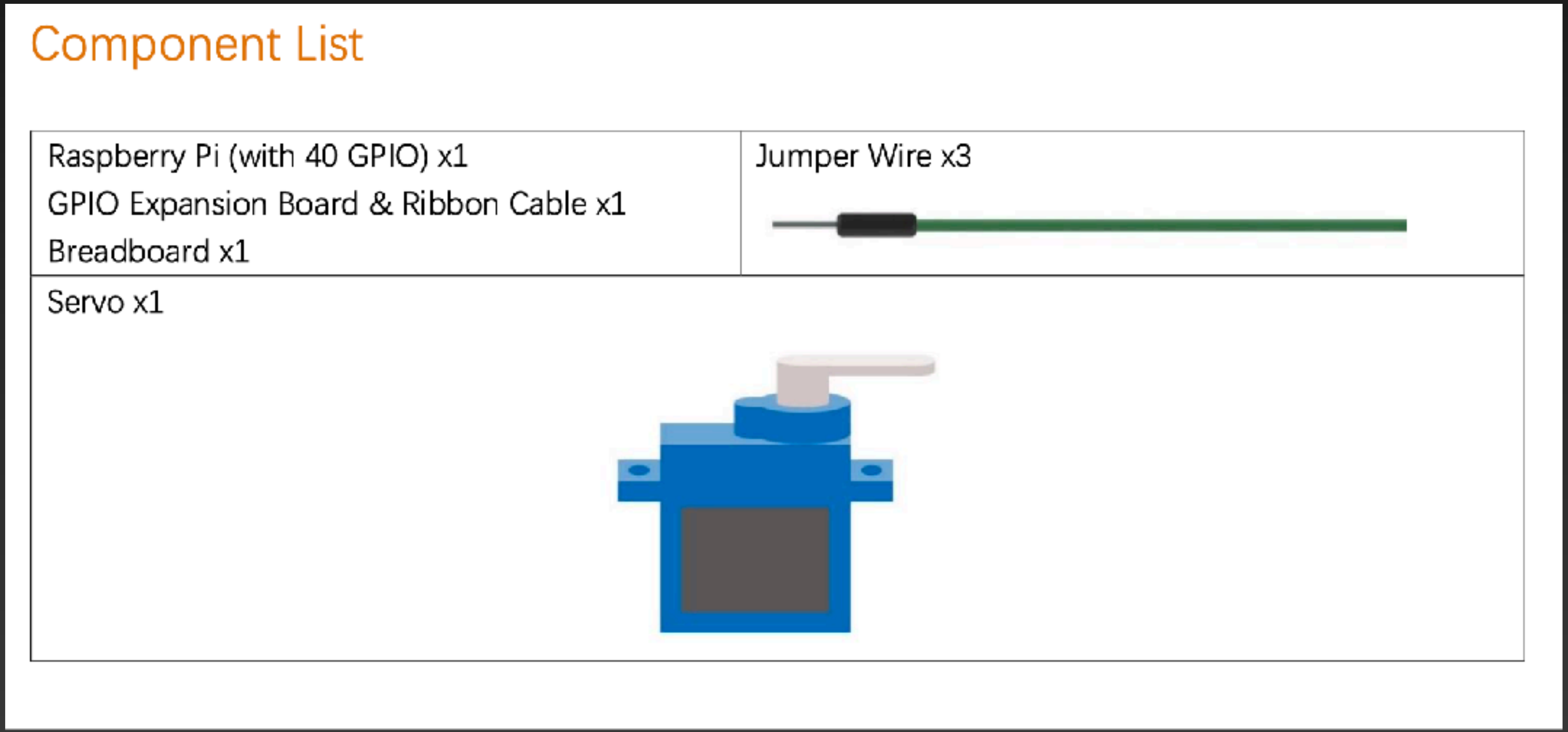
X. Tang

Prepare the Env and Test the Sensor

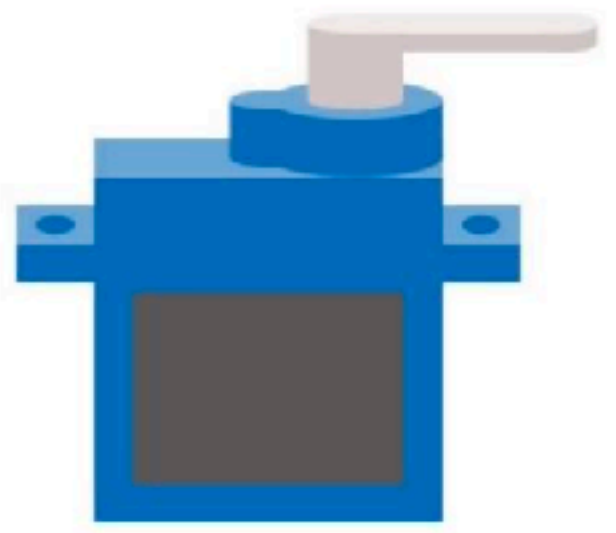
```
git clone --depth 1 https://github.com/freenove/Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi

mv Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi  ~/Freenove_Kit/

cd Freenove_Kit/Code/Python_GPIOZero_Code/21.1.1_DHT11/

python DHT11.py
```

Servo Motor Control

We will learn how to make motor rotate.

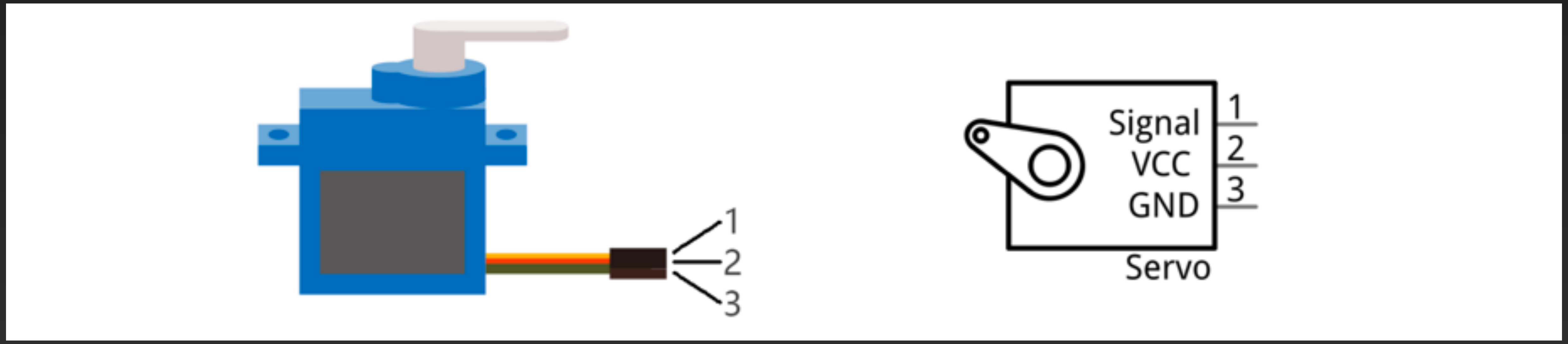


Component List

Raspberry Pi (with 40 GPIO) x1 GPIO Expansion Board & Ribbon Cable x1 Breadboard x1	Jumper Wire x3
Servo x1 	

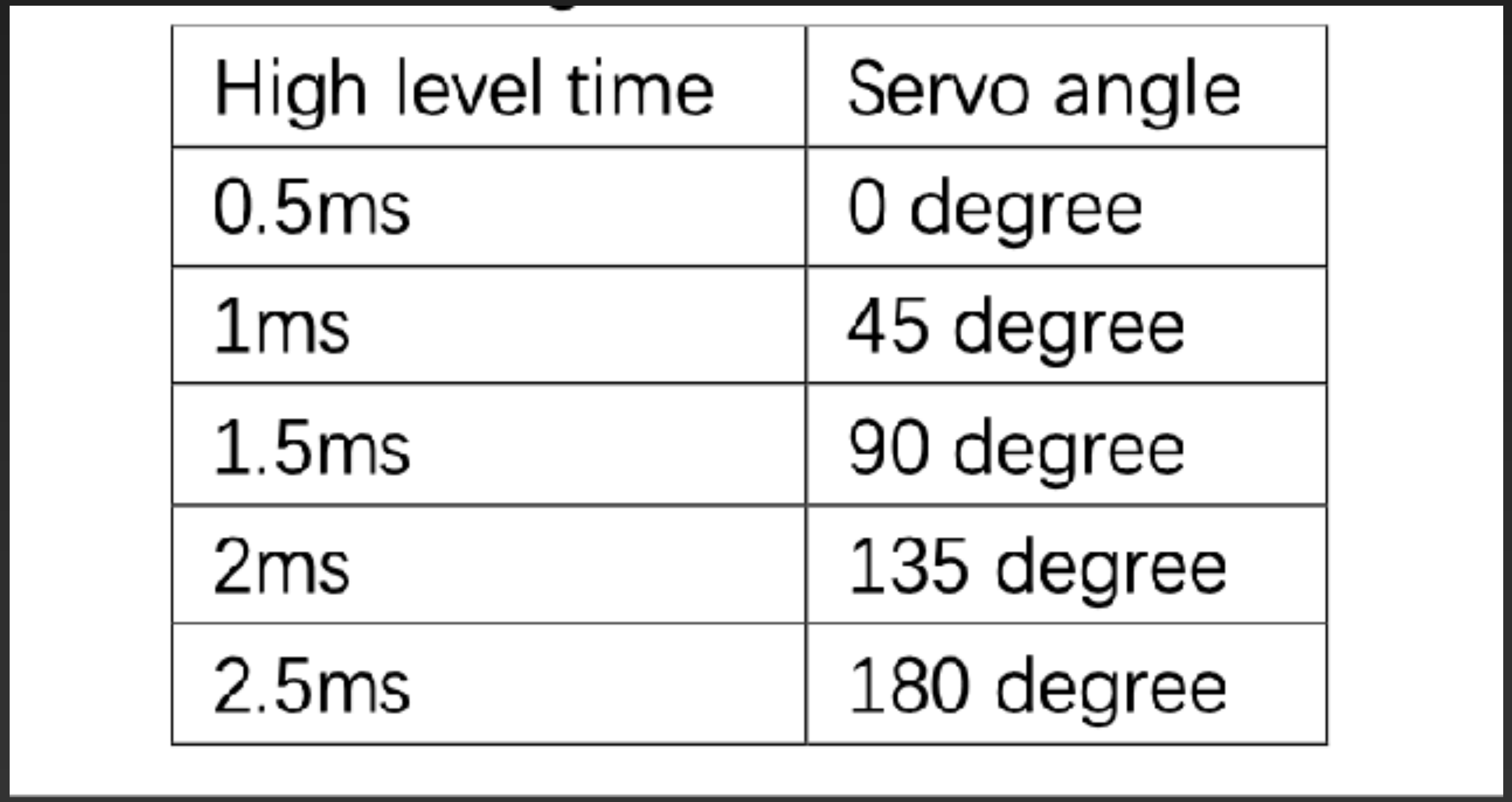
Servo Motor Control

Servo is a compact package which consists of a DC Motor, a set of reduction gears to provide torque, a sensor and control circuit board. Most Servos only have a 180-degree range of motion via their “horn”. Servos can output higher torque than a simple DC Motor alone and they are widely used to control motion in model cars, model airplanes, robots, etc. Servos have three wire leads which usually terminate to a male or female 3-pin plug. Two leads are for electric power: Positive (2-VCC, Red wire), Negative (3-GND, Brown wire), and the signal line (1-Signal, Orange wire) as represented in the Servo provided in your Kit.



Servo Motor Control

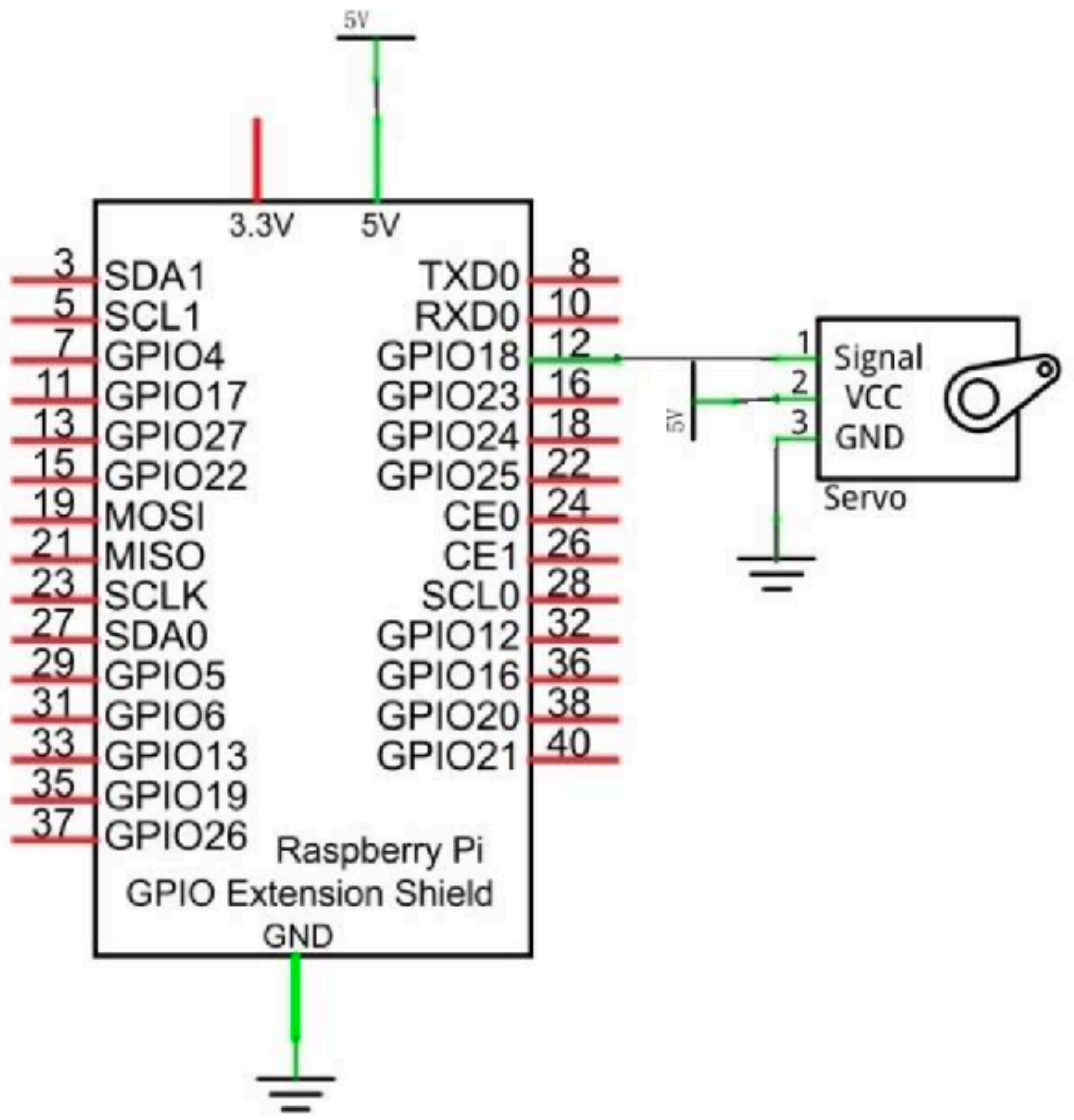
We will use a 50Hz PWM signal with a duty cycle in a certain range to drive the Servo. The lasting time 0.5ms- 2.5ms of PWM single cycle high level corresponds to the Servo angle 0 degrees - 180 degree linearly. Part of the corresponding values are as follows:
Note: the lasting time of high level corresponding to the servo angle is absolute instead of accumulating. For example, the high level time lasting for 0.5ms correspond to the 0 degree of the servo. If the high level time lasts for another 1ms, the servo rotates to 45 degrees.



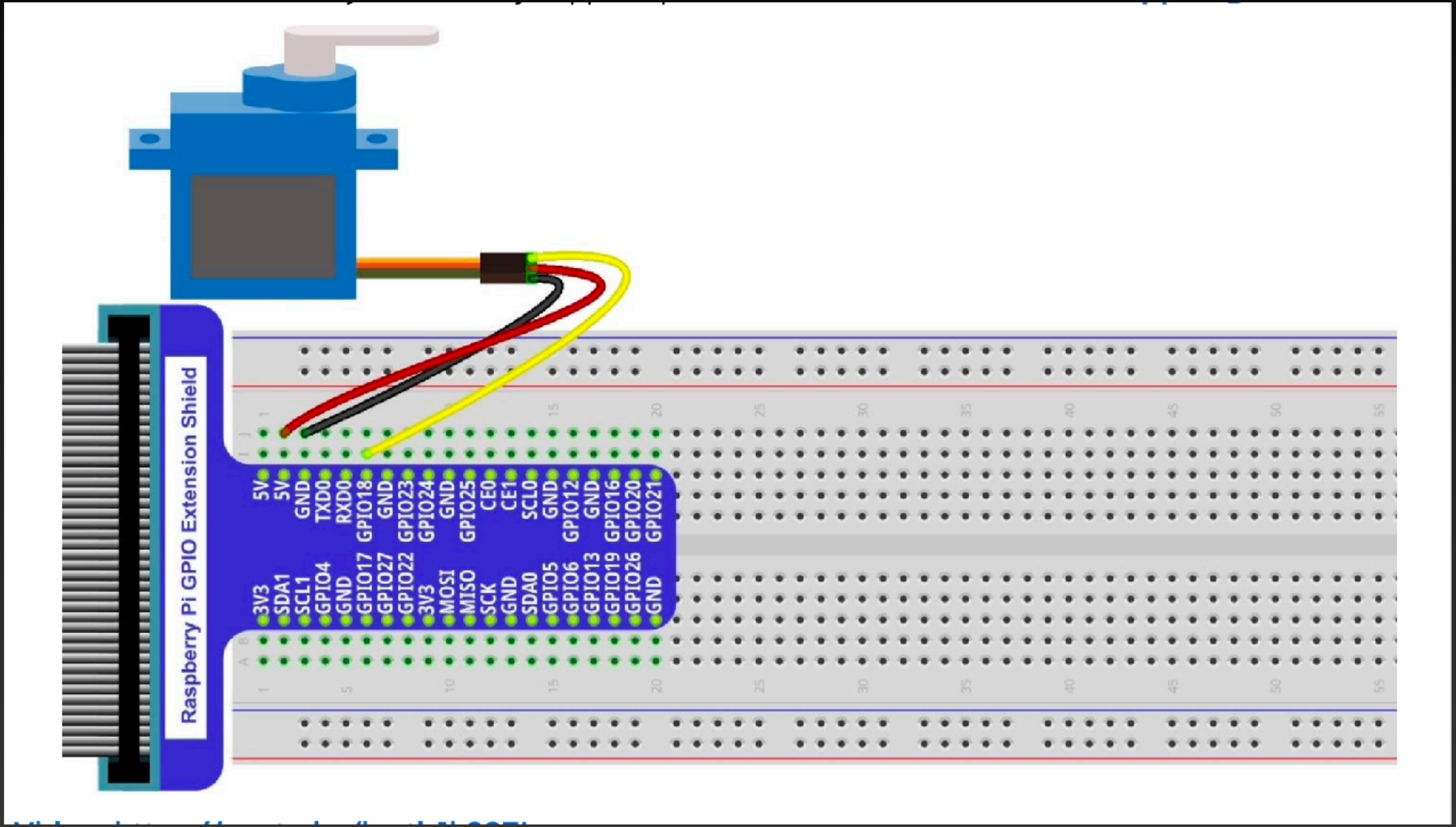
High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	90 degree
2ms	135 degree
2.5ms	180 degree

Servo Motor Control

Schematic diagram



Servo Motor Control



Servo Motor Control

```
import RPi.GPIO as GPIO

import time


OFFSE_DUTY = 0.5  #define pulse offset of servo

SERVO_MIN_DUTY = 2.5 + OFFSE_DUTY  #define the minimum angle of servo

SERVO_MAX_DUTY = 12.5 + OFFSE_DUTY #define the maximum angle of servo

servoPin = 12


def map(value, fromLow, fromHigh, toLow, toHigh):

    return (toHigh-toLow)*(value-fromLow)/(fromHigh-fromLow) + toLow
```

Servo Motor Control

```
def setup():
    global p
    GPIO.setmode(GPIO.BOARD)   #use PHYSICAL GPIO numbering
    GPIO.setup(servoPin, GPIO.OUT) #set servoPin to OUTPUT mode
    GPIO.setup(servoPin, GPIO.LOW) #make servoPin output to LOW level

    p = GPIO.PWM(servoPin, 50) #set Freq to 50Hz
    p.start(0)                 #set initial duty cycle to 0
```

Servo Motor Control

```
def servoWrite(angle):
    if(angle<0):
        angle = 0
    elif(angle > 180):
        angle = 180
    p.ChangeDutyCycle(map(angle,0,180,SERVO_MIN_DUTY,SERVO_MAX_DUTY)) #map the
angle to duty cycle and output it
```

Servo Motor Control

```
def loop():
    while True:
        for dc in range(0, 181, 1):  # make servo rotate from 0 to 180
            servoWrite(dc)   #write dc value to servo
            time.sleep(0.001)
        time.sleep(0.5)

        for dc in range(180, -1, -1): # make the servo rotate from 180 to 0 deg
            servoWrite(dc)
            time.sleep(0.001)
        time.sleep(0.5)
```

Servo Motor Control

```
def destroy():
    p.stop()
    GPIO.cleanup()

if __name__ == '__main__':  # program entrance
    print('Program is starting...')
    setup()
    try:
        loop()
    except KeyboardInterrupt: #press ctrl-c to end the program
        destroy()
```

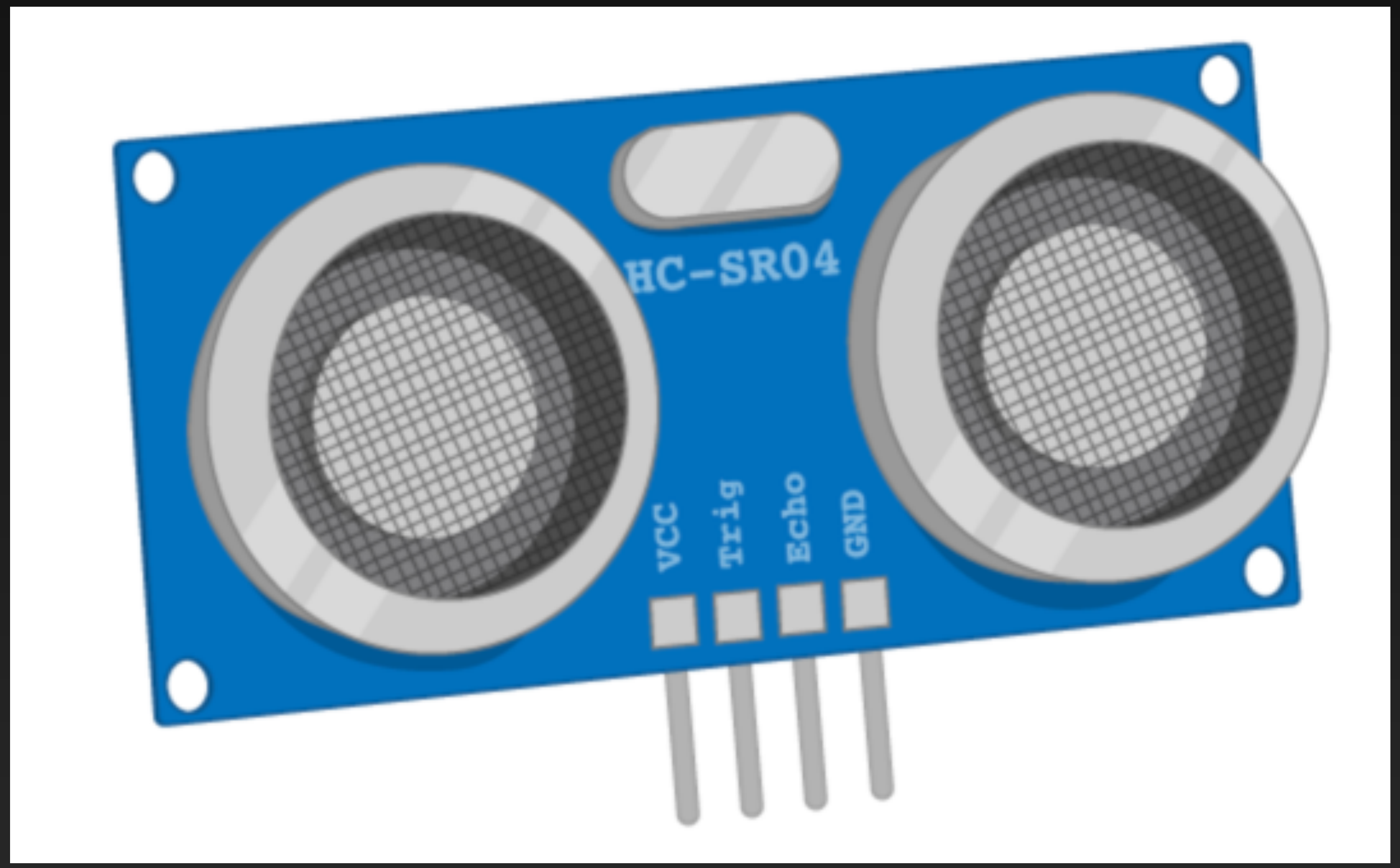
Setup Distance Sensor

An ultrasonic distance sensor is a device that sends out pulses of ultrasonic sound, and measures the time they take to bounce off nearby objects and be reflected back. They can measure distances fairly accurately (up to about a meter).

Setup Distance Sensor

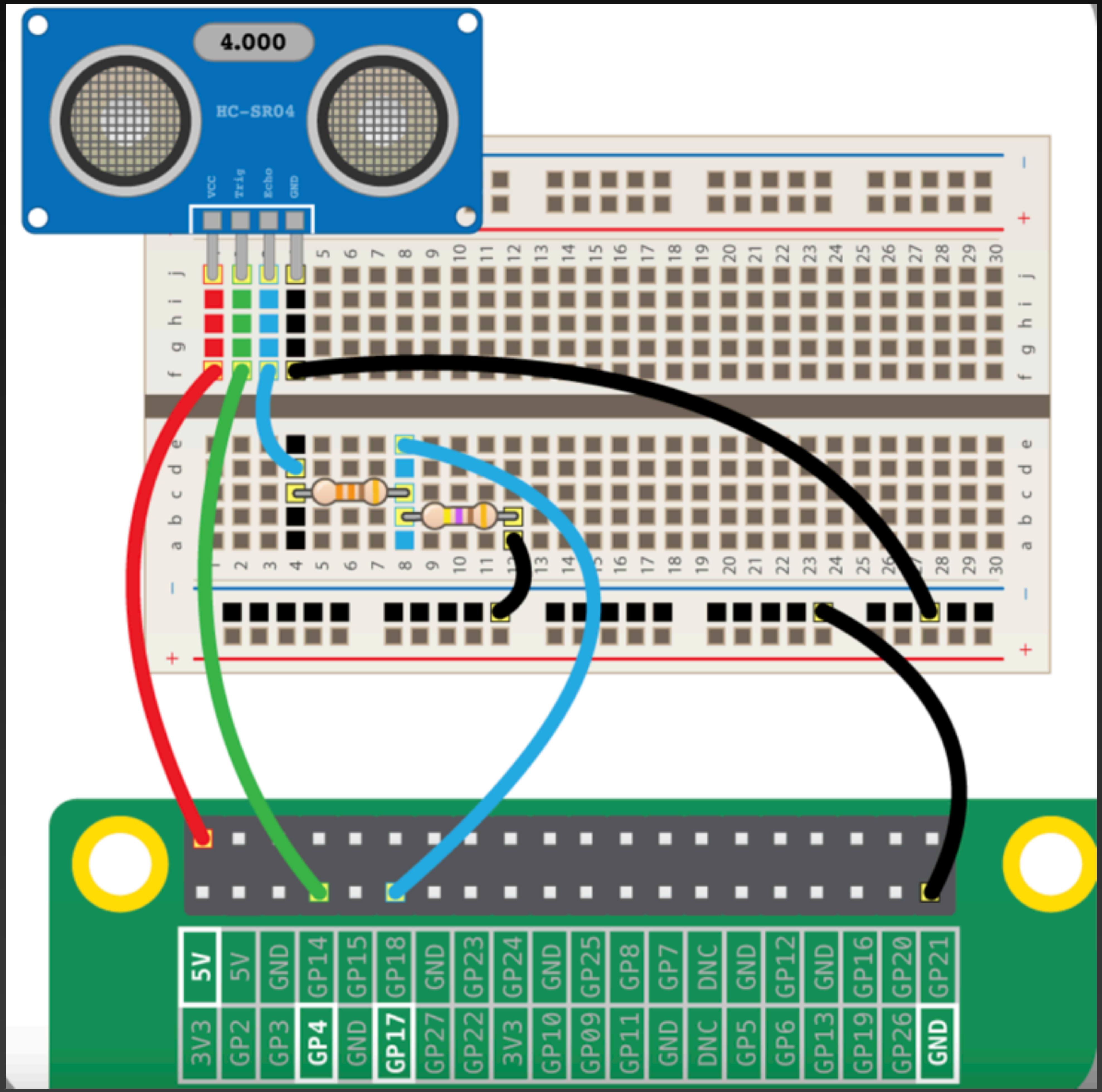
An ultrasonic distance sensor has four pins. They are called Ground (Gnd), Trigger (Trig), Echo (Echo), and Power (Vcc).

To use an ultrasonic distance sensor, you need to connect the Gnd pin to the ground pin on the Raspberry Pi, the Trig and Echo pins to GPIO pins on the Raspberry Pi, and the Vcc pin to the 3V3 pin on the Raspberry Pi.



Setup Distance Sensor

The circuit connects to two GPIO pins (one for echo, one for trigger), the ground pin, and a 5V pin. You'll need to use a pair of resistors (220Ω) as a potential divider:

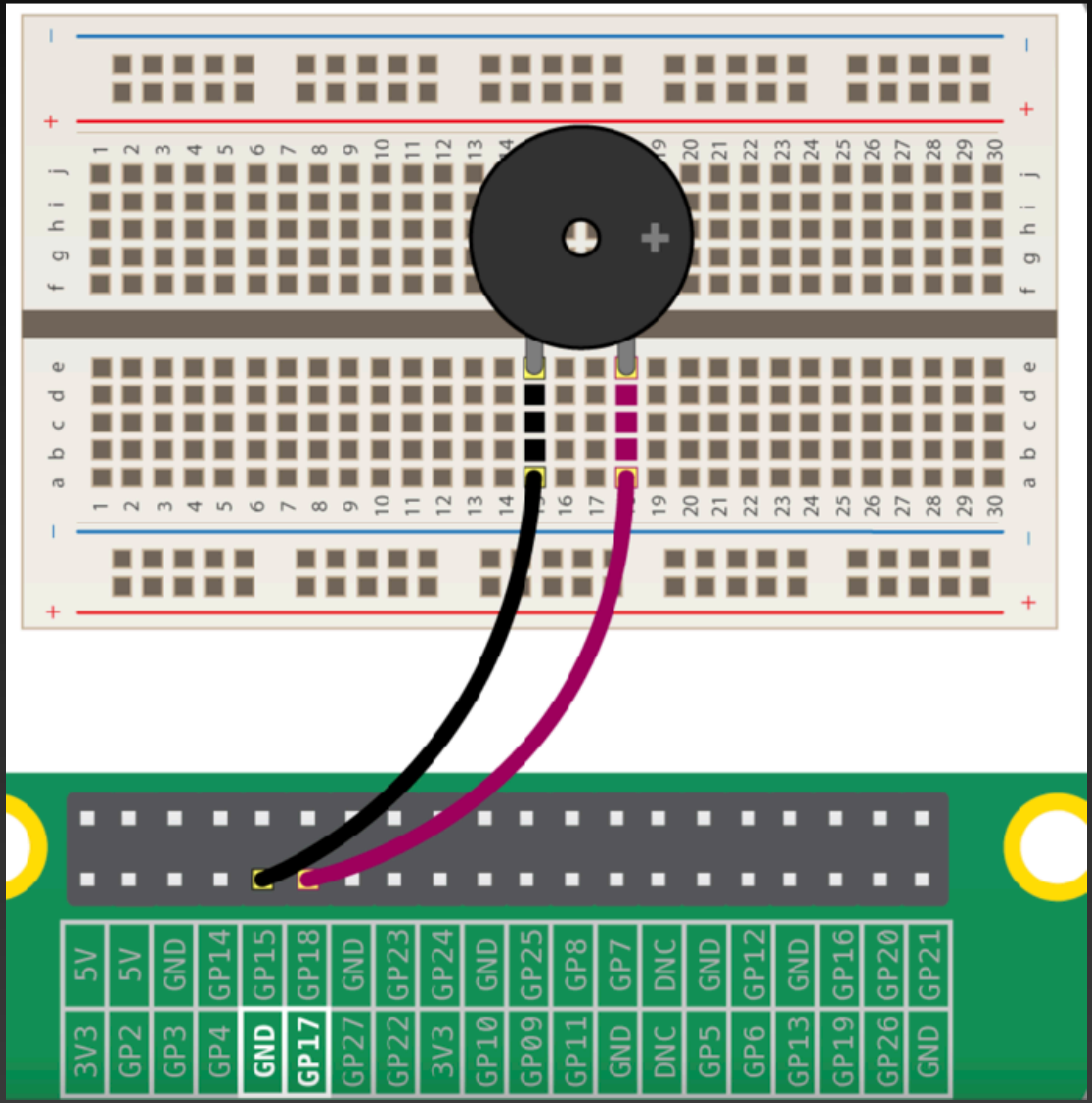


Setup Distance Sensor

```python
from gpiozero import DistanceSensor
ultrasonic = DistanceSensor(echo=17, trigger=4)
while True:
    print(ultrasonic.distance)
```

Setup Buzzer

An active buzzer can be connected just like an LED, but as they are a little more robust, you won't be needing a resistor to protect them.



5V
5V
GND
GP14
GP15
GP18
GND
GP23
GP24
GND
GP25
GP8
GP7
DNC
GND
GP12
GND
GP16
GP20
GP21
3V3
GP2
GP3
GP4
GND
GP17
GP27
GP22
3V3
GP10
GP09
GP11
GND
DNC
GP5
GP6
GP13
GP19
GP26
GND