ENPM 809T

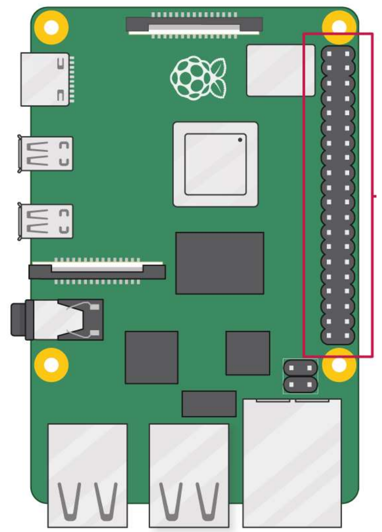
UMCP, Mitchell

GPIO

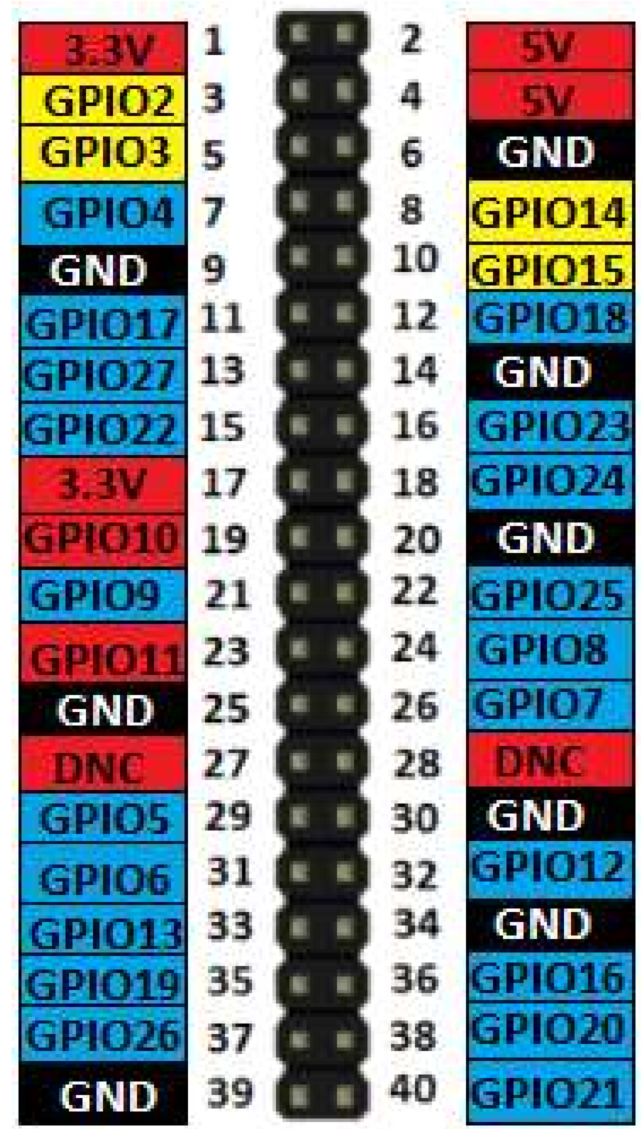
- Uncommitted digital signal pin
- Acts as either input or output
  - **3.3V on Raspberry Pi**
- Controlled by user at run time
- Pulse-width modulation
- I2C
- Serial

GPIO: Raspberry Pi

- Voltages
  - 2x 5V pins
  - 2x 3.3V pins
  - GND 0V pins
- Outputs
  - High (3.3V) or low (0V)
- Inputs
  - Read as high (3.3V) or low (0V)
- Type **pinout** in terminal window

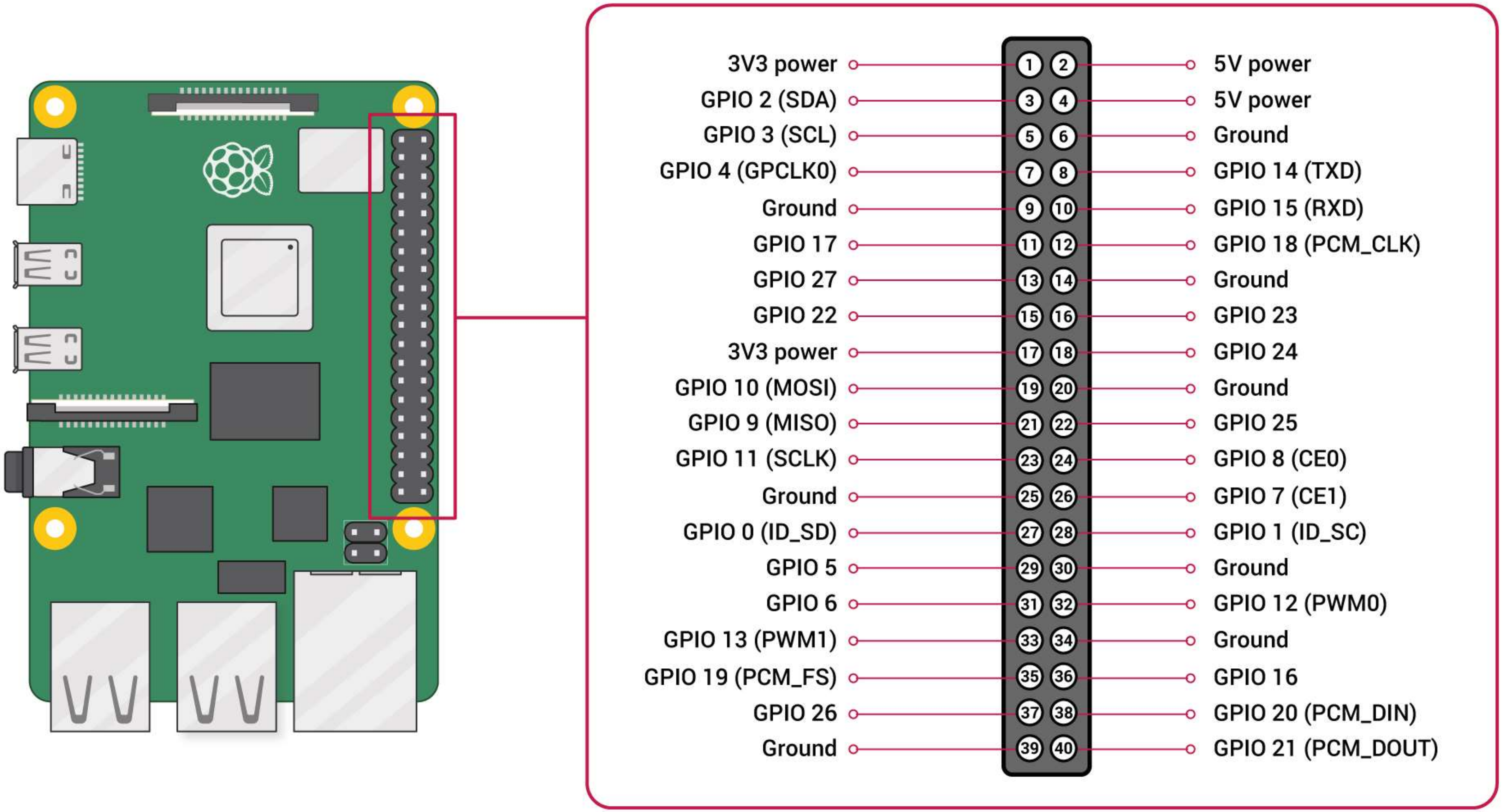


**** Numbering is <u>not</u> in numerical order ****



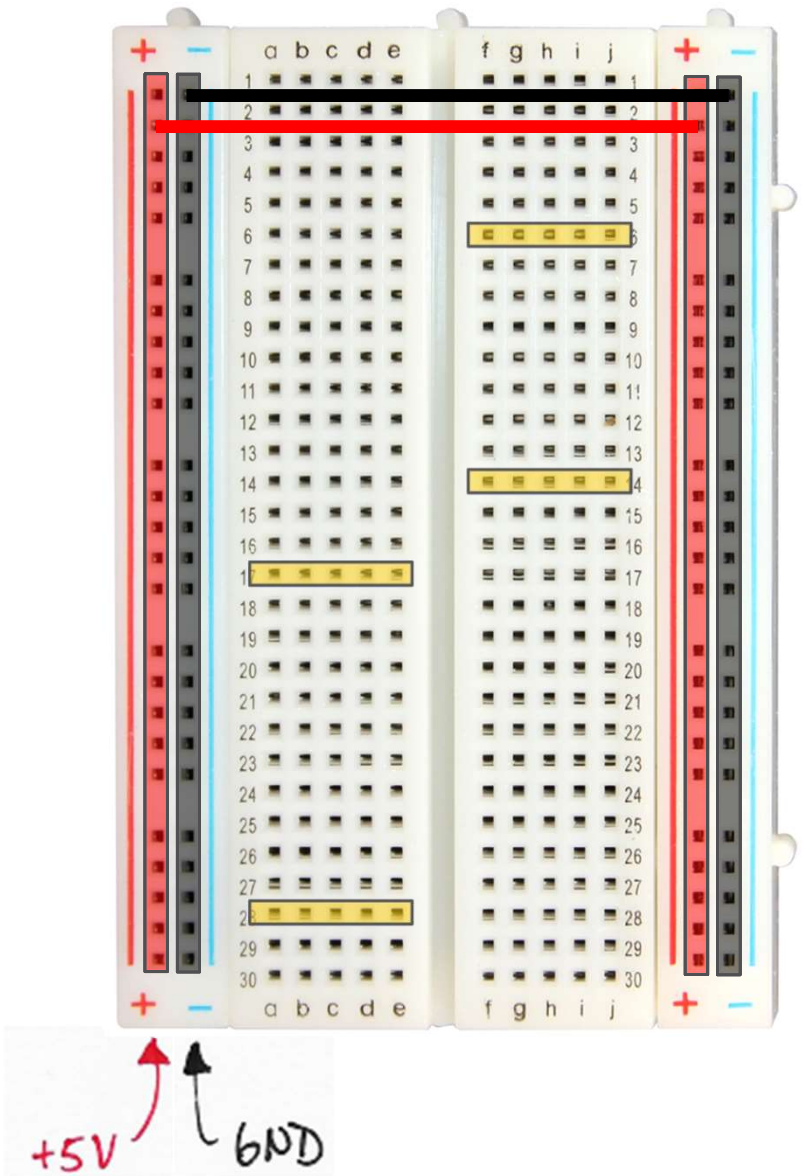
https://www.raspberrypi.org/documentation/usage/gpio/

GPIO: Raspberry Pi

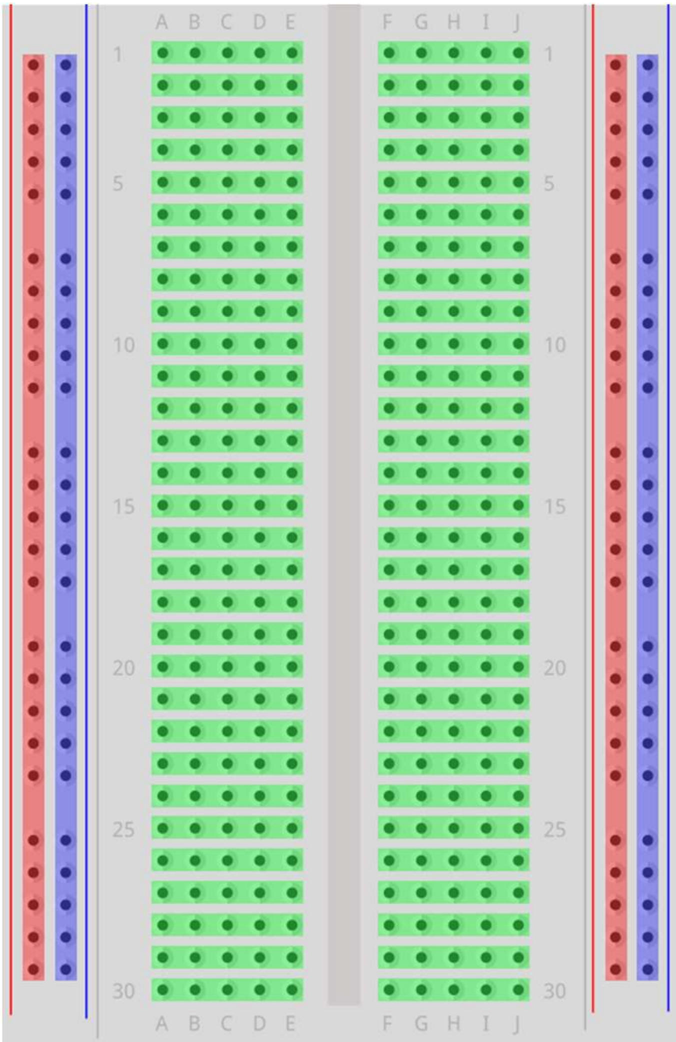


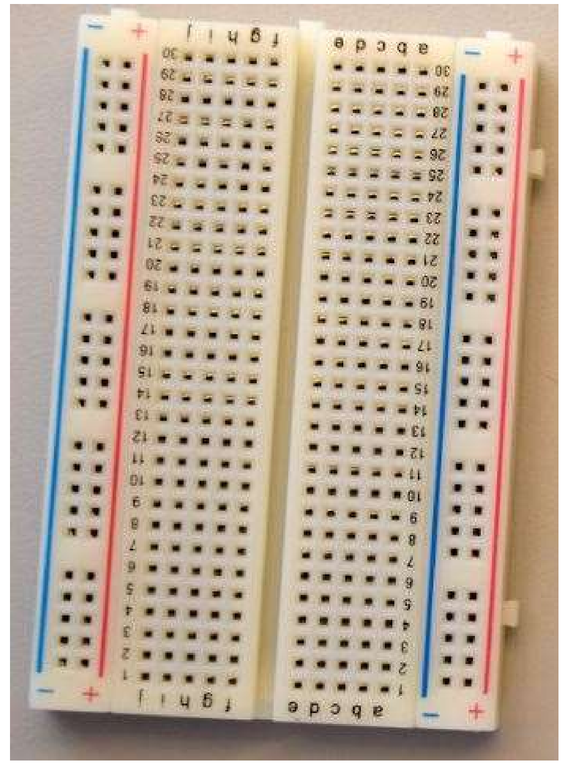
https://www.raspberrypi.org/documentation/usage/gpio/

Solderless breadboards



Solderless breadboards





https://www.w3schools.com/nodejs/nodejs_raspberrypi_gpio_intro.asp

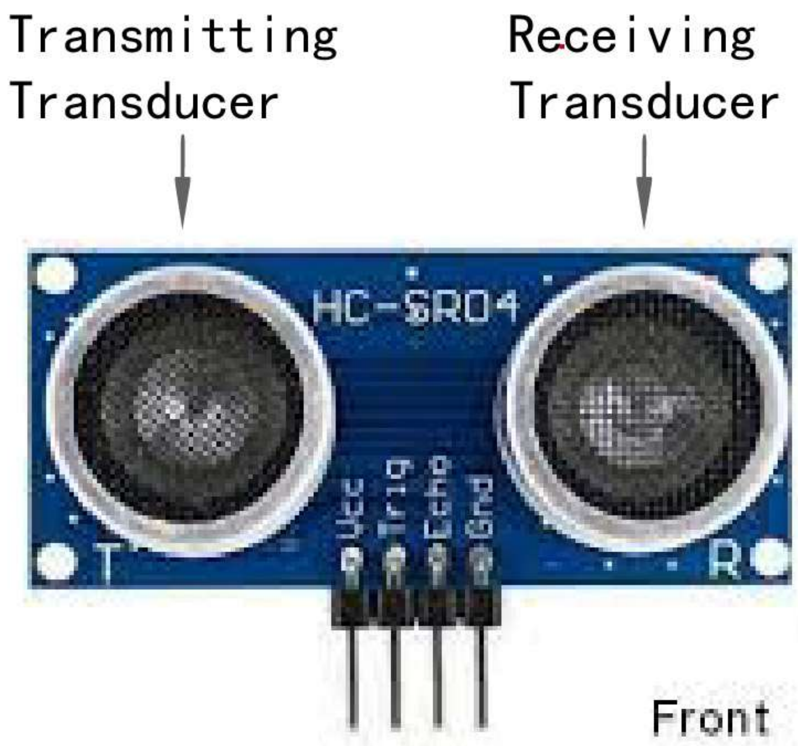
Ultrasonic Range Sensor

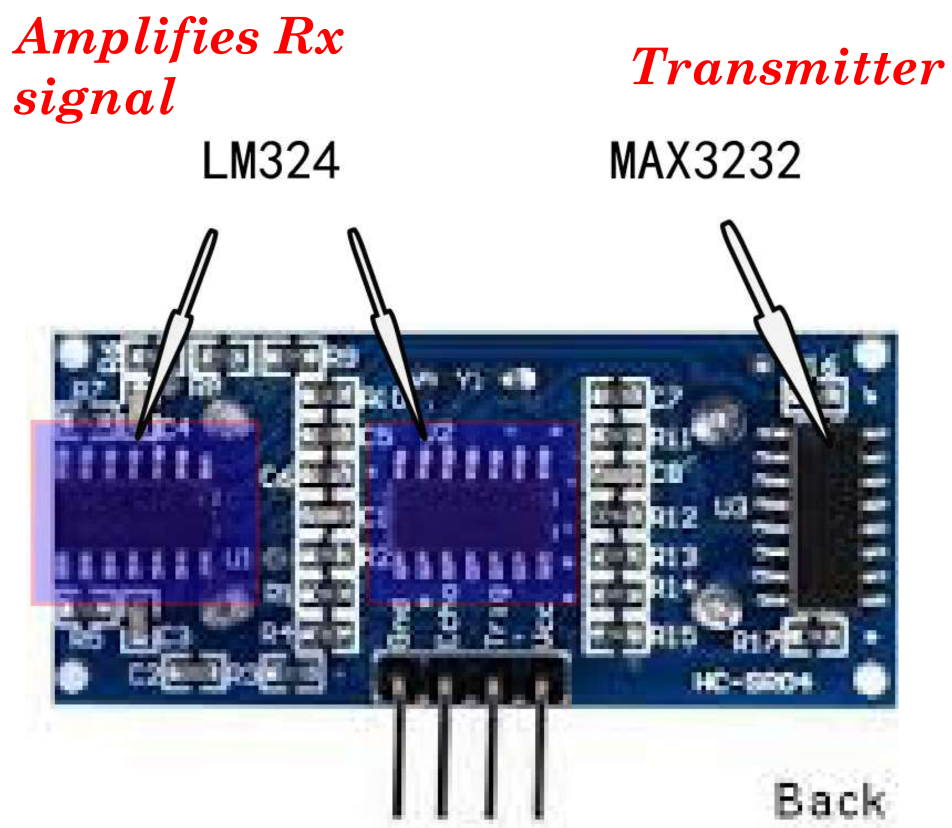
- Transmits high-frequency pulses of sound (pressure) via transducers
- Measure time from transmission to reception of scattered wave
- Typical range 2 - 400 cm (1 - 13 ft)

https://www.maxbotix.com/Ultrasonic_Sensors/MB1010.htm

http://osoyoo.com/2017/07/23/arduino-lesson-ultrasonic-sensor-hc-sr04/

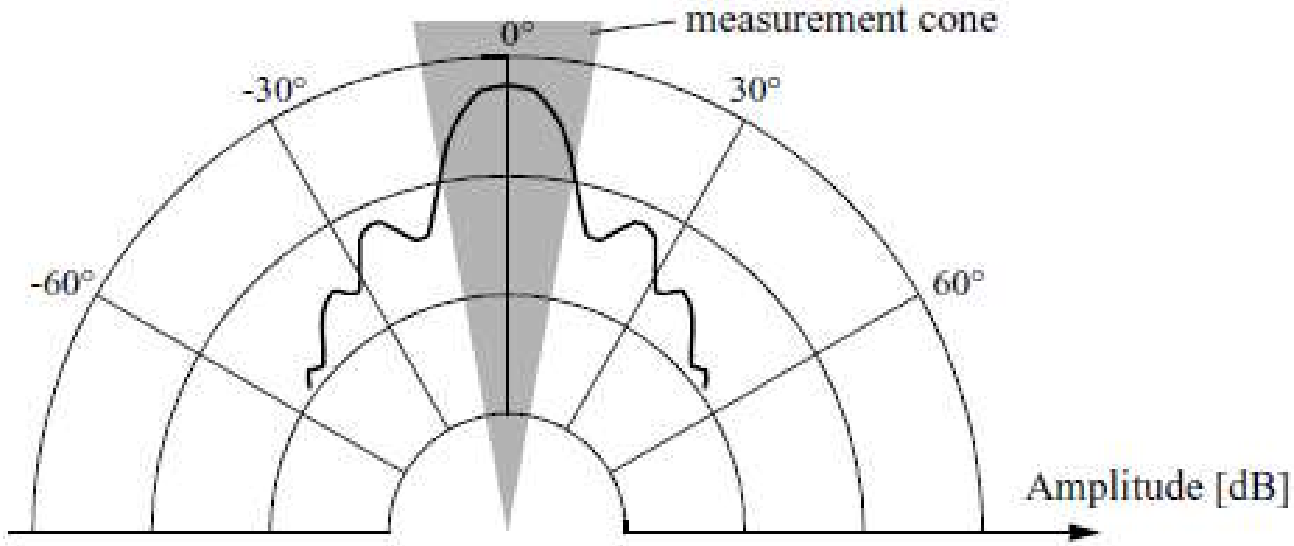
Ultrasonic Range Sensor

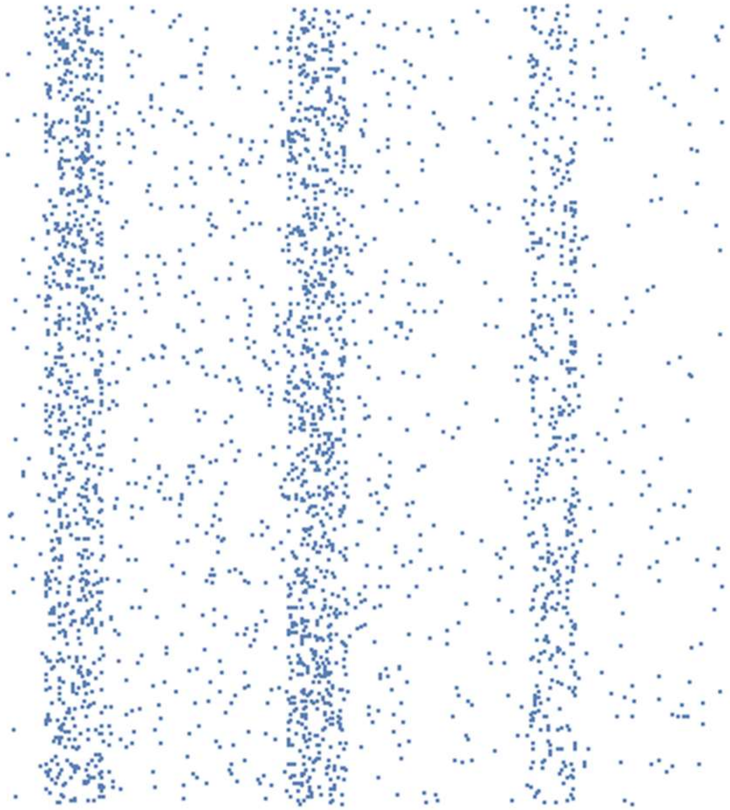




http://osoyoo.com/2017/07/23/arduino-lesson-ultrasonic-sensor-hc-sr04/


T
R
VCC
Trig
Echo
GND


measurement cone
0°
-30°
30°
-60°
60°
Amplitude [dB]



Ultrasonic Range Sensor

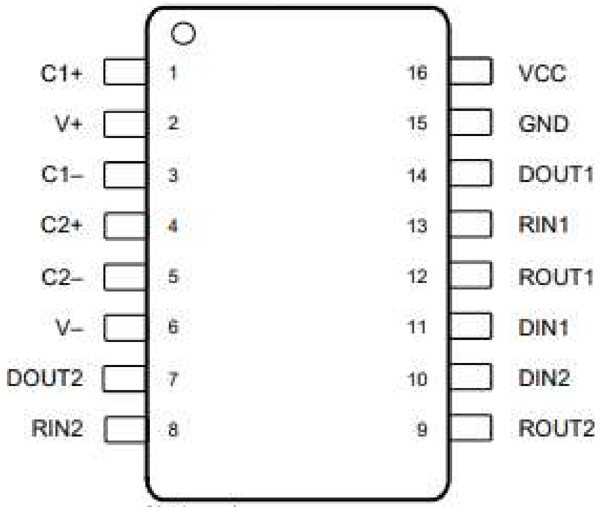
TEXAS INSTRUMENTS

MAX3232

SLLS410N – JANUARY 2000 – REVISED JUNE 2017

MAX3232 3-V to 5.5-V Multichannel RS-232 Line Driver/Receiver With ±15-kV ESD Protection

D, DB, DW, or PW Package
16-Pin SOIC, SSOP, or TSSOP
Top View



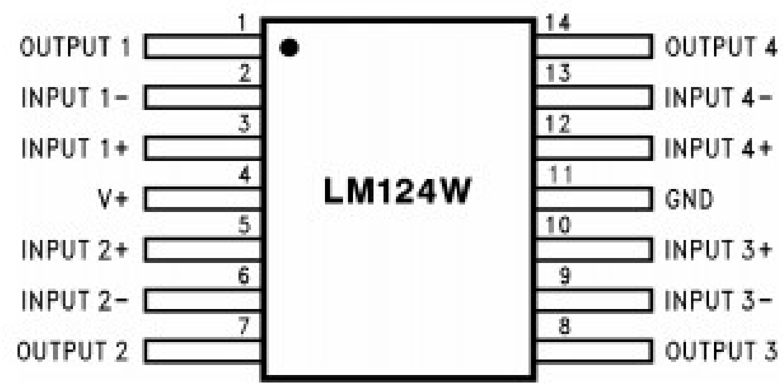
Not to scale

TEXAS INSTRUMENTS

LM124-N, LM224-N
LM2902-N, LM324-N

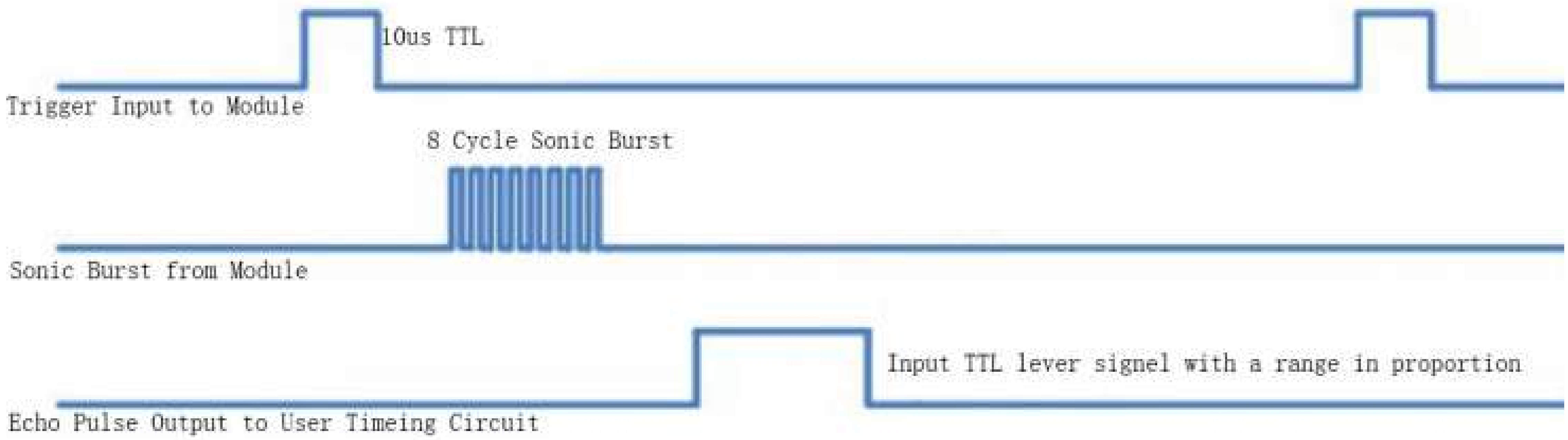
SNOSC16D – MARCH 2000 – REVISED JANUARY 2015

LMx24-N, LM2902-N Low-Power, Quad-Operational Amplifiers



http://www.ti.com/lit/ds/symlink/max3232.pdf *http://www.ti.com/lit/ds/symlink/lm324-n.pdf*

Timing

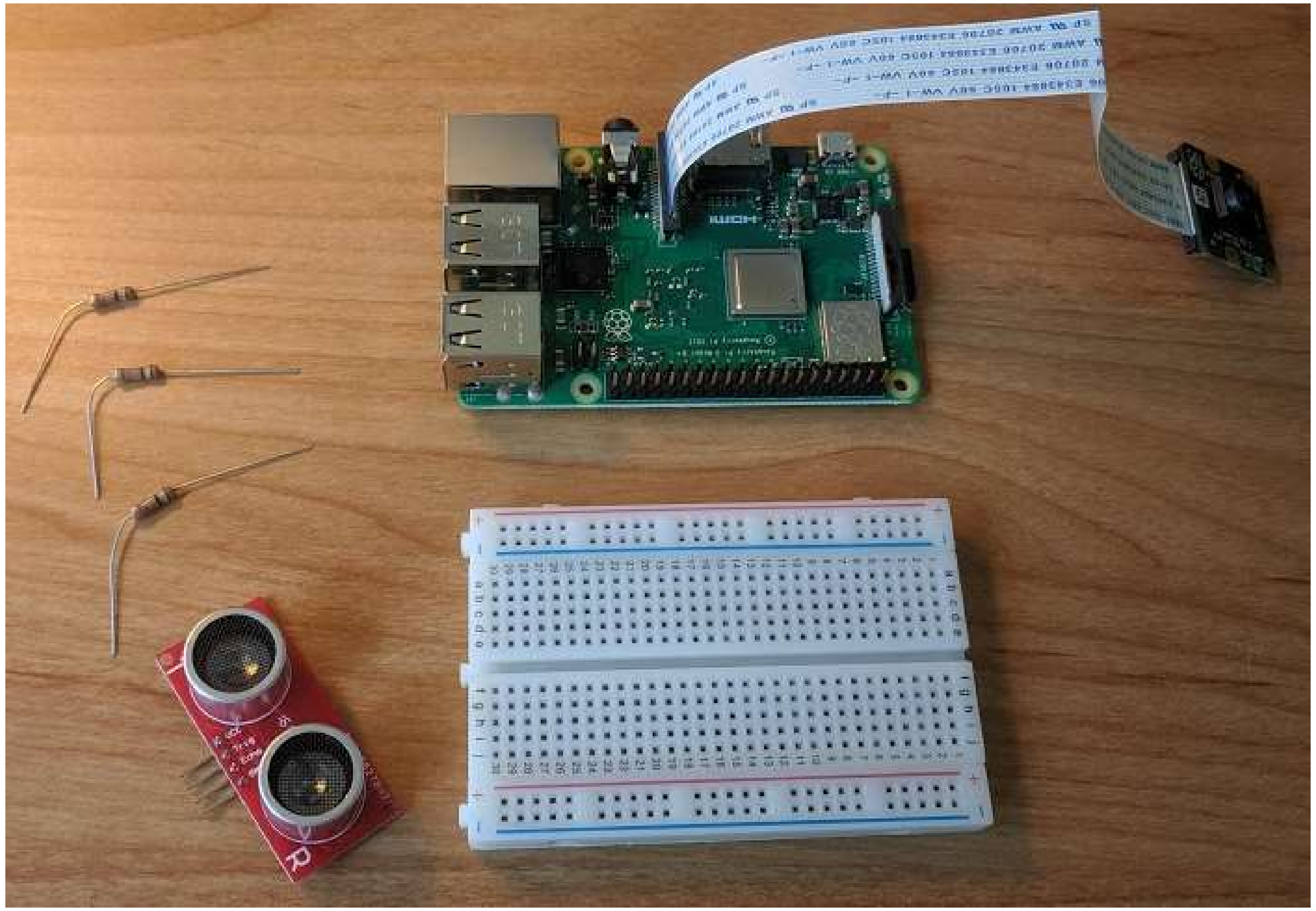


https://cdn.sparkfun.com/assets/b/3/0/b/a/DGCH-RED_datasheet.pdf

Circuit

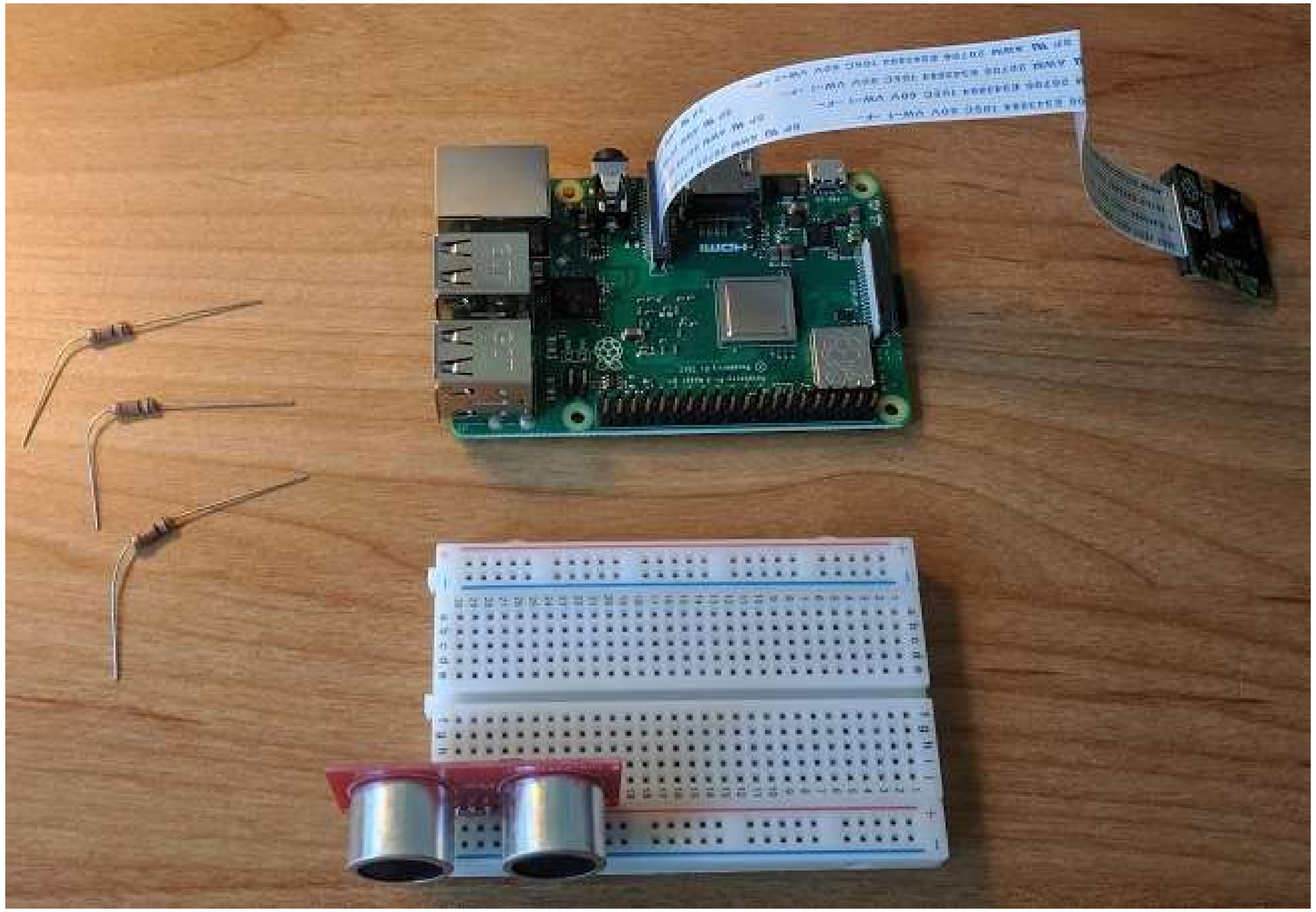
- From parts kit:

1. Raspberry Pi
2. Breadboard
3. Distance sensor
4. Three 1kΩ resistors
5. Jumper wires



Circuit

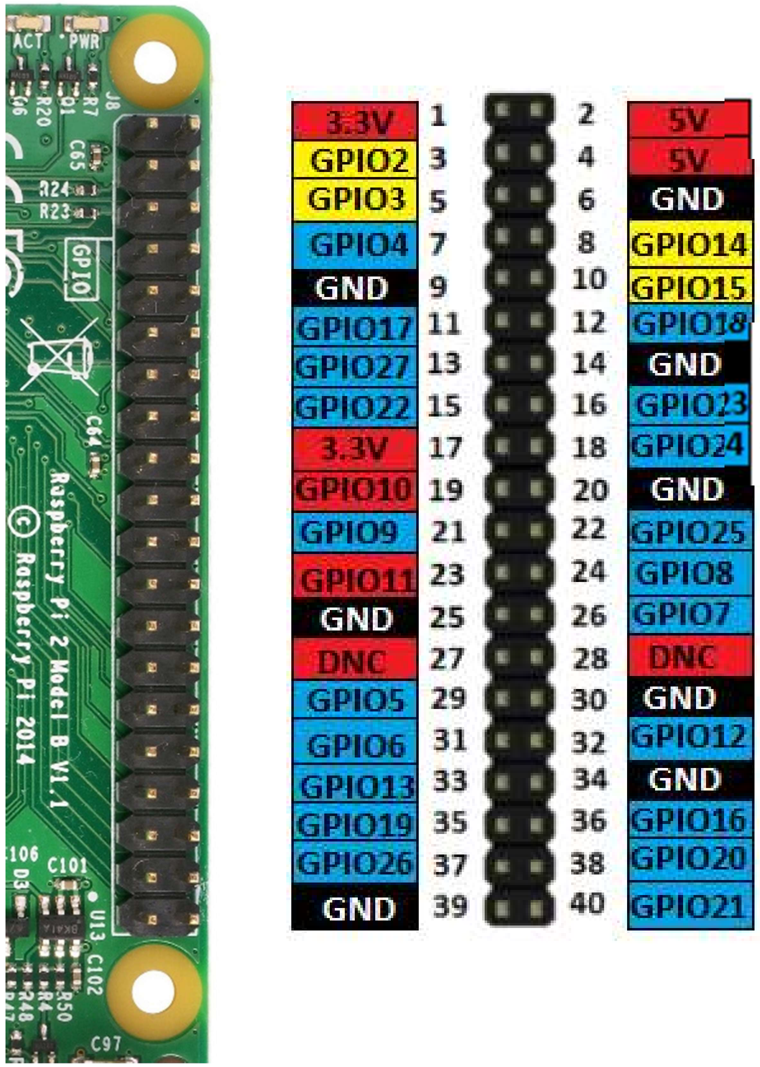
- Plug sensor into end of breadboard



Pin Allocations

Distance sensor

- We will **track** throughout the course

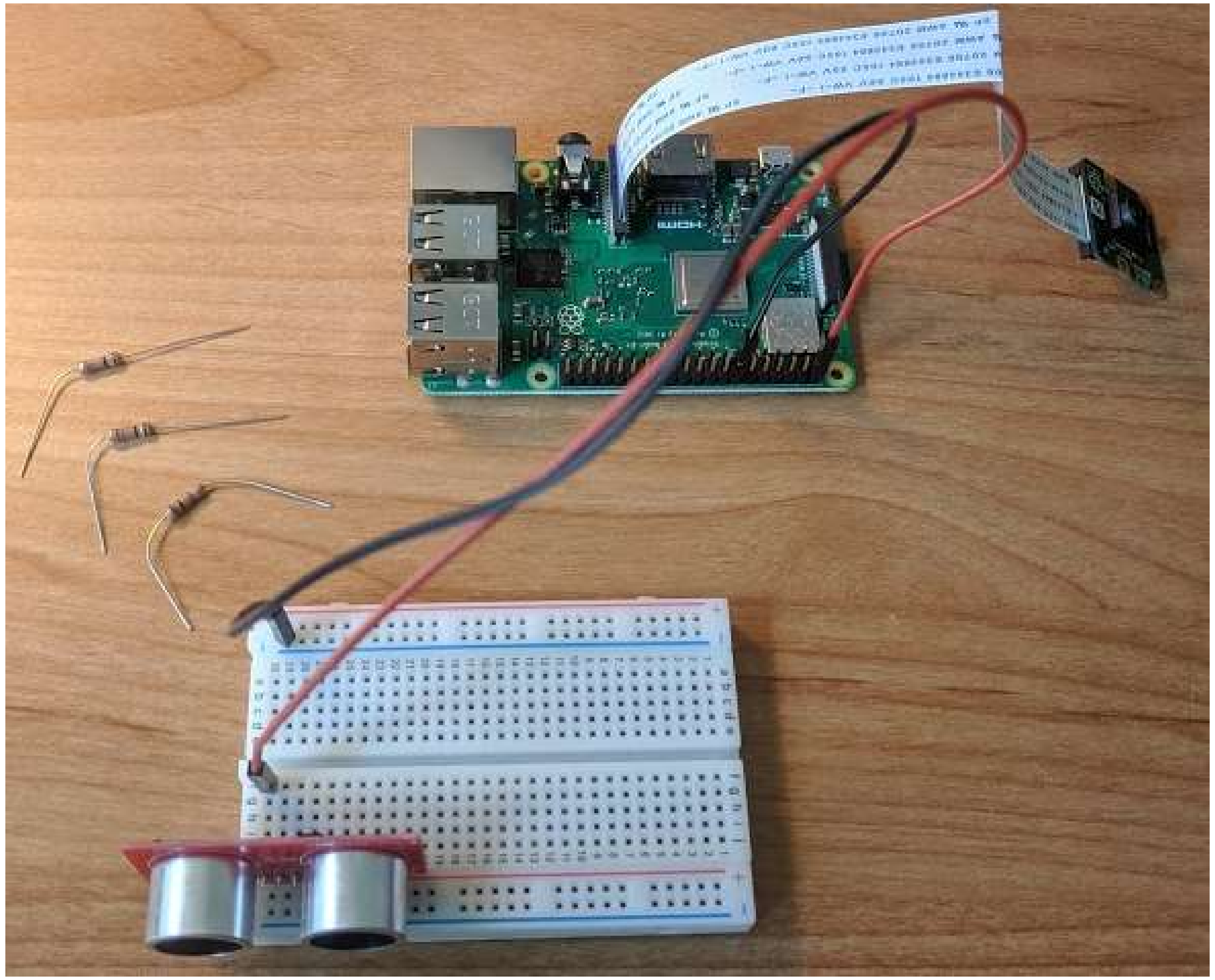


TRIG
ECHO

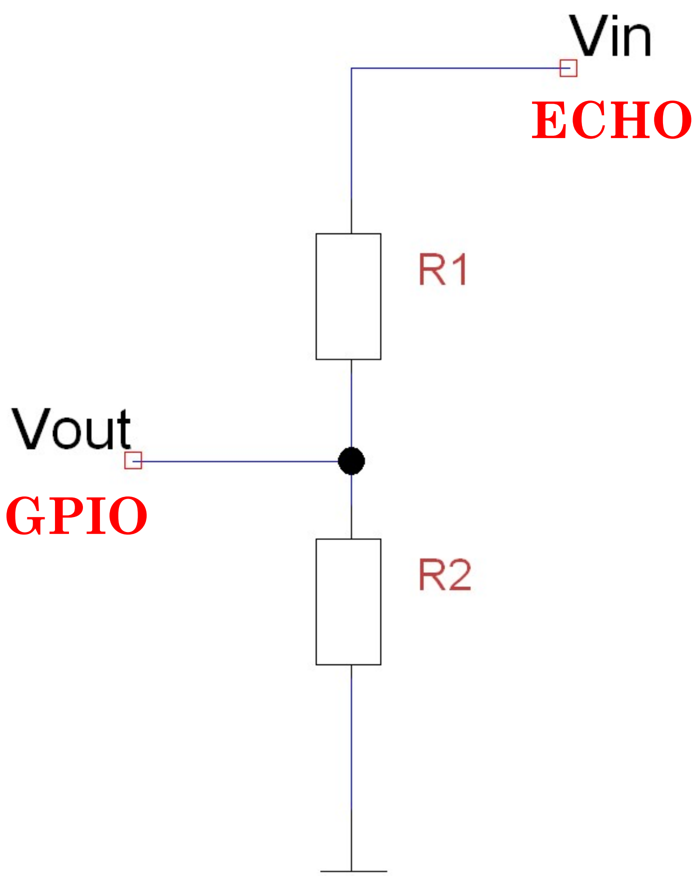
Circuit

- Plug **red** male-female wire into pin 2
  - **5V** Vcc supply
- Plug **black** male-female wire into pin 14
  - **Ground**

- Plug 5V into sensor Vcc
- Plug GND into breadboard ground rail



Circuit



$$Vout = Vin \times \frac{R2}{R1 + R2}$$

$$\frac{Vout}{Vin} = \frac{R2}{R1 + R2}$$

$$\frac{3.3}{5} = \frac{R2}{1000 + R2}$$

$$0.66 = \frac{R2}{1000 + R2}$$

$$0.66(1000 + R2) = R2$$

$$660 + 0.66R2 = R2$$

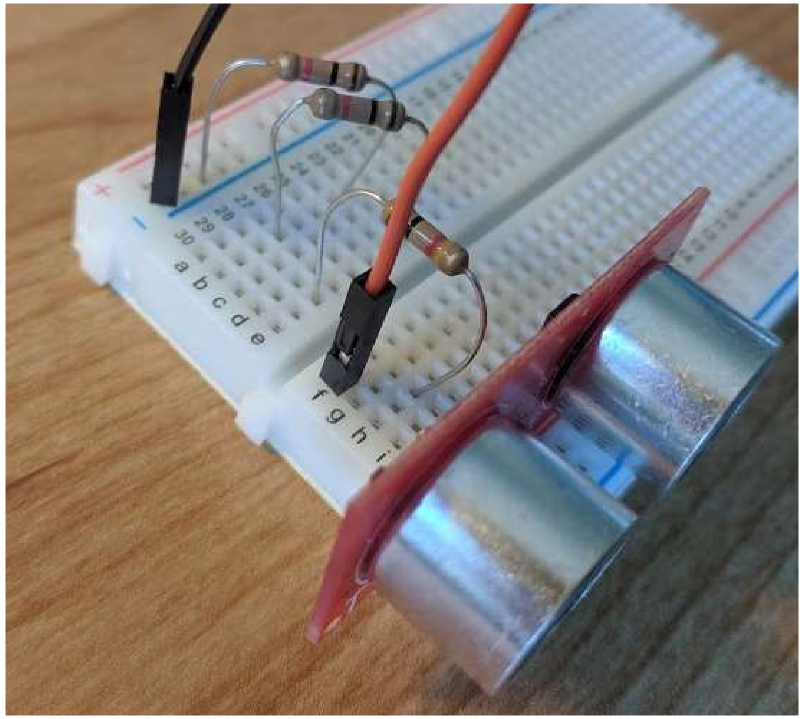
$$660 = 0.34R2$$

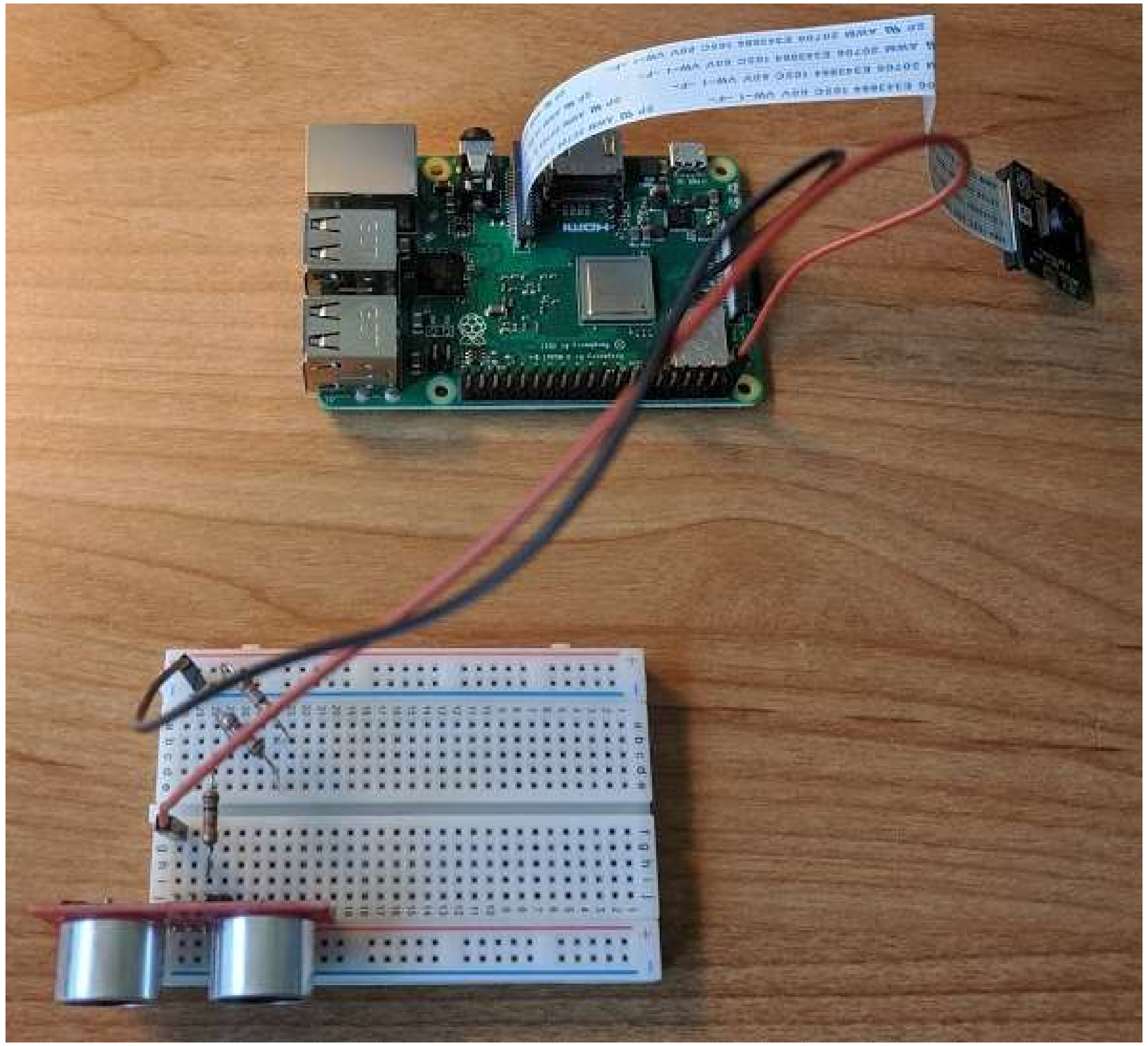
$$1941 = R2$$

https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

Circuit

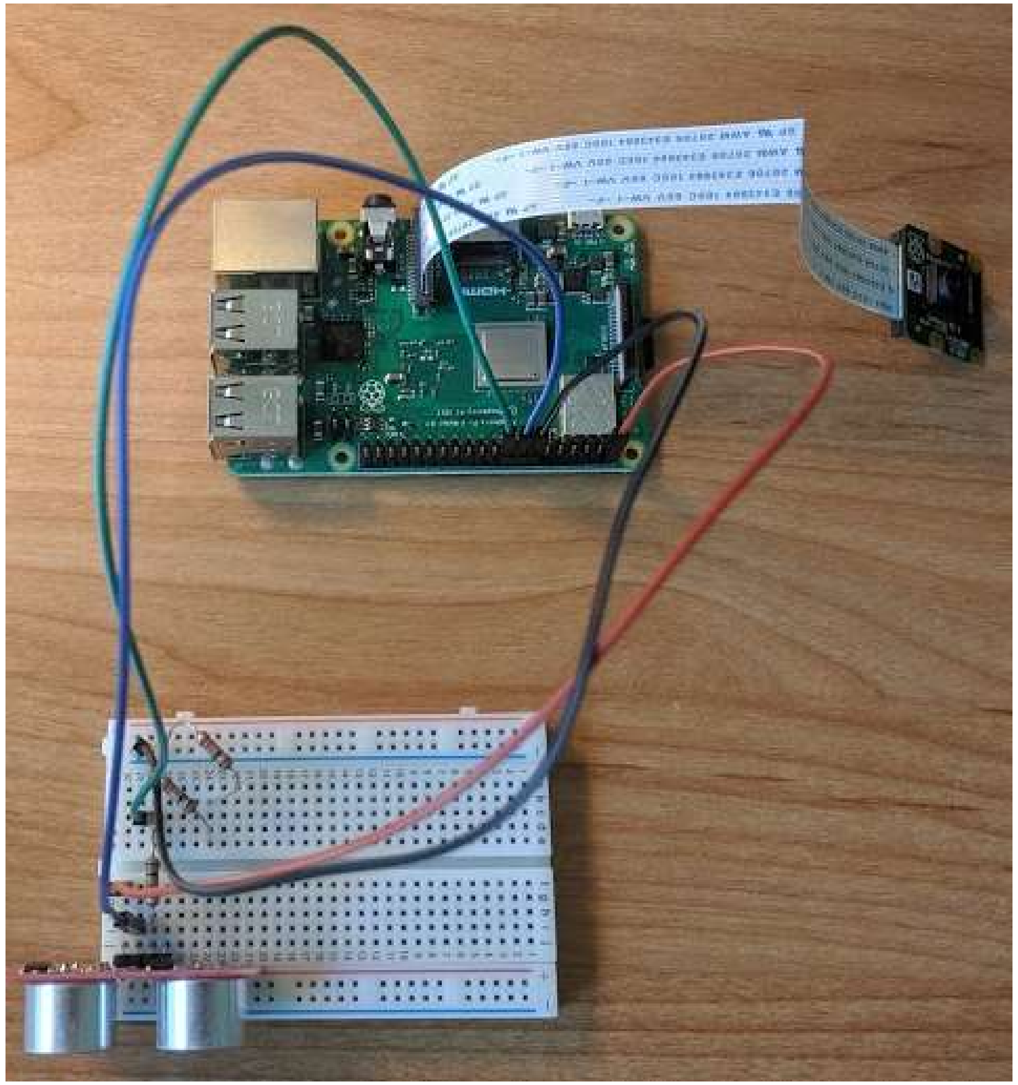
- Create voltage divider using three 1kΩ resistors





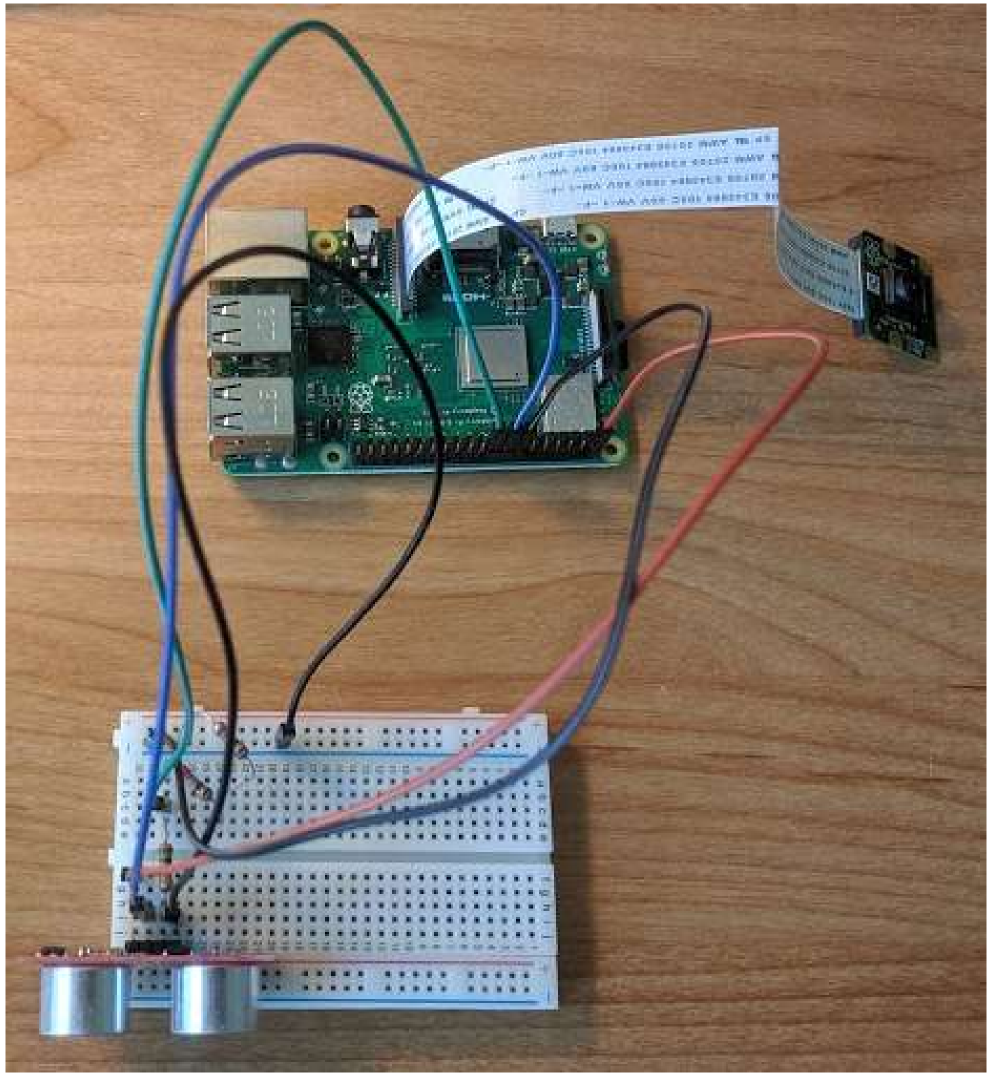
Circuit

- Plug **blue** male-female wire into pin 16
  - GPIO23
  - Sensor **Trig**
- Plug **green** male-female wire into pin 18
  - GPIO24
  - Sensor **Echo**
- Connect each wire on breadboard



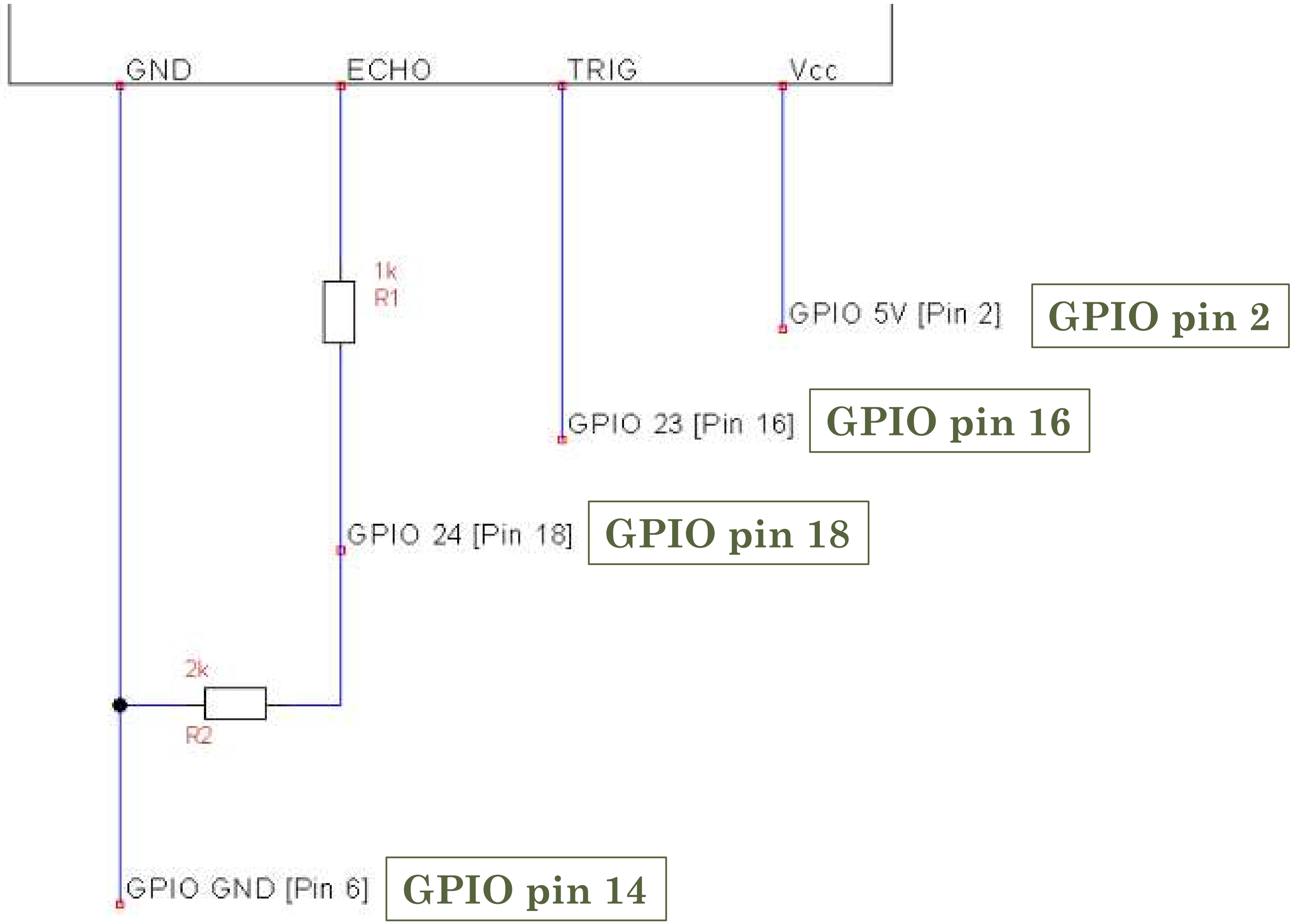
Circuit

- Plug **black** male-male wire between sensor **GND** and breadboard **GND** rail



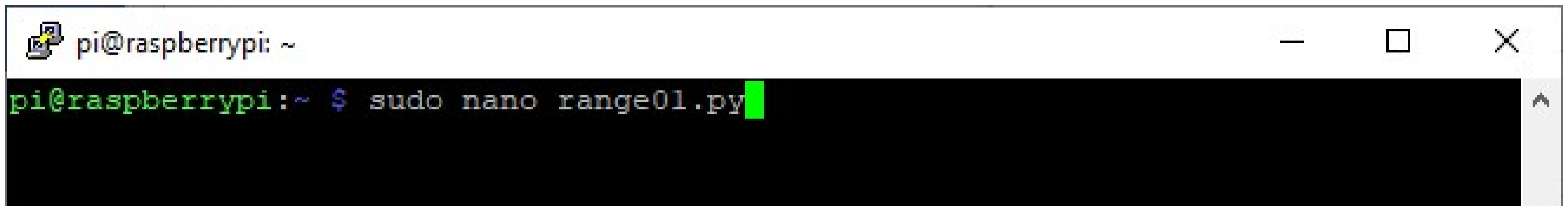
Circuit

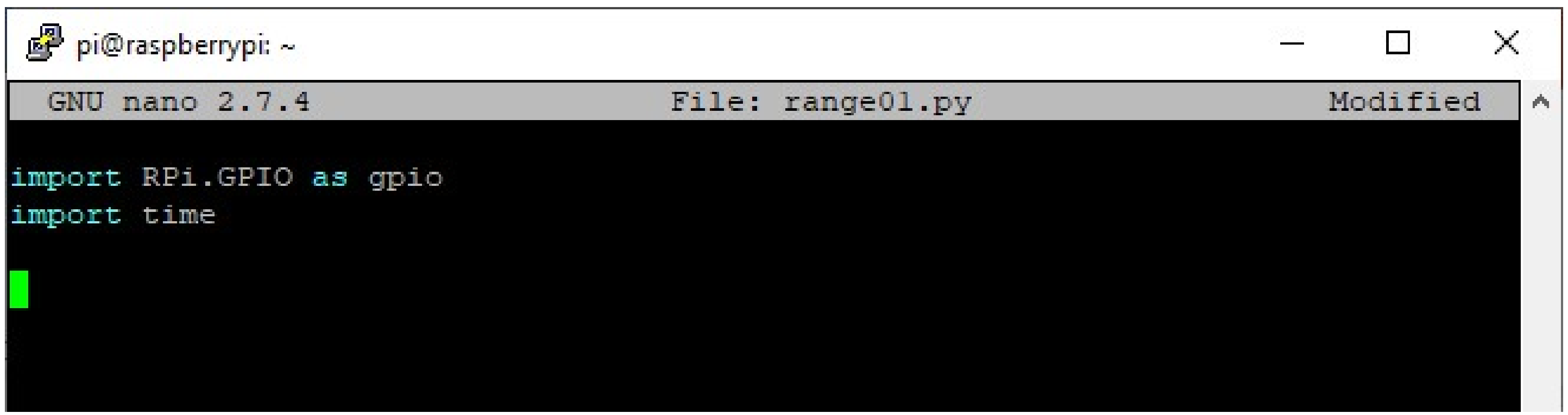
* ***Confirm*** *circuit is* ***properly wired*** *before applying power to the RPi*



Code

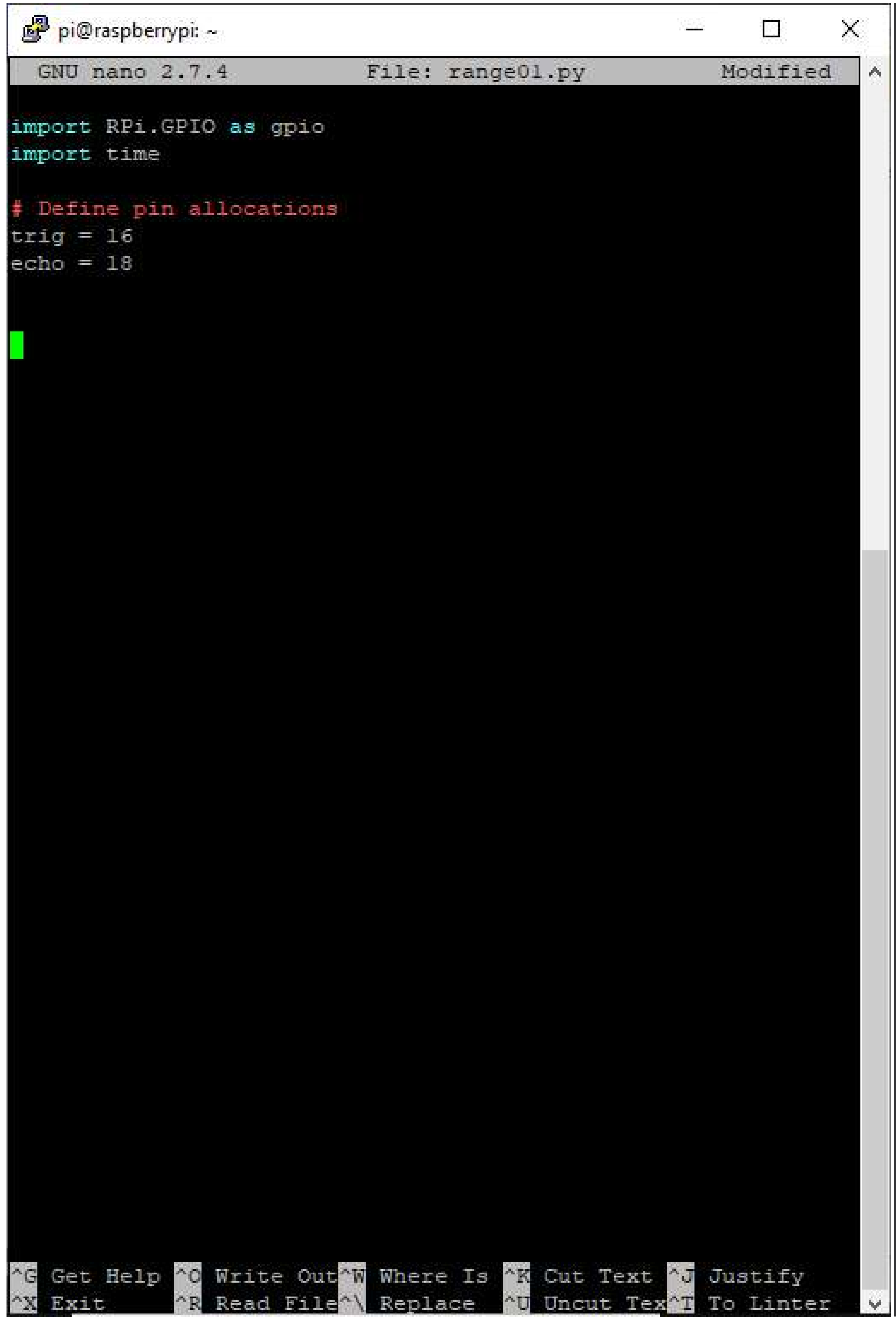
- Create a new .py file: ***range01.py***
- Import **RPIO** and **time** modules





Code

- Define pins for trigger & echo

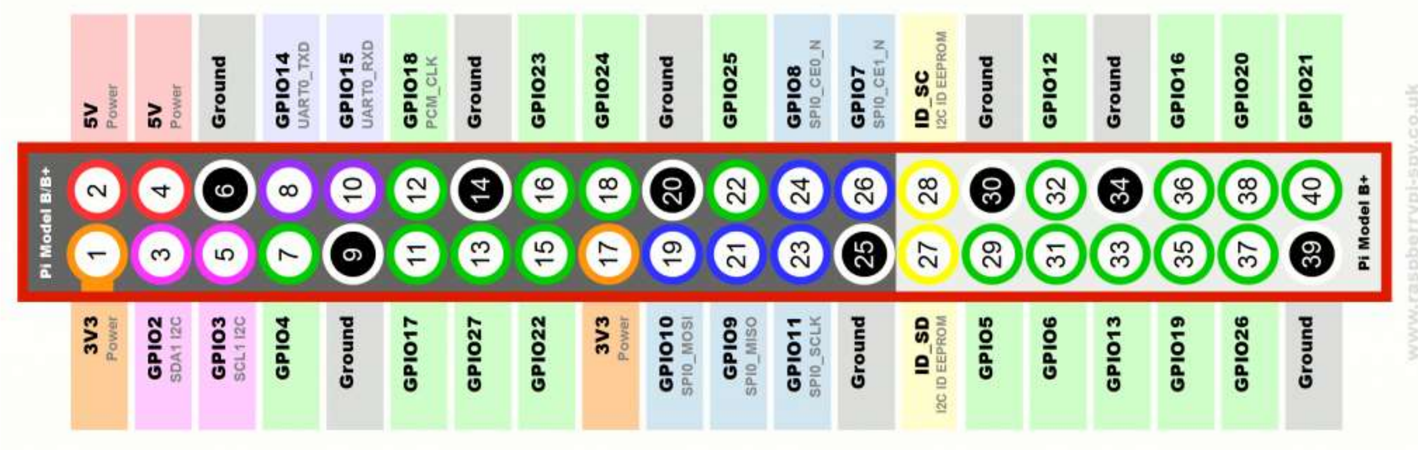


Code

- Create distance( ) function
- Performs all required operations to measure range
- Returns single distance measurement

Code

- Setup board



- Assign GPIO pins as either input or output

pi@raspberrypi: ~

GNU nano 2.7.4 File: range01.py Modified

```python
import RPi.GPIO as gpio
import time

# Define pin allocations
trig = 16
echo = 18

def distance():
        gpio.setmode(gpio.BOARD)
        gpio.setup(trig, gpio.OUT)
        gpio.setup(echo, gpio.IN)
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter

Code

- Set trig pin low

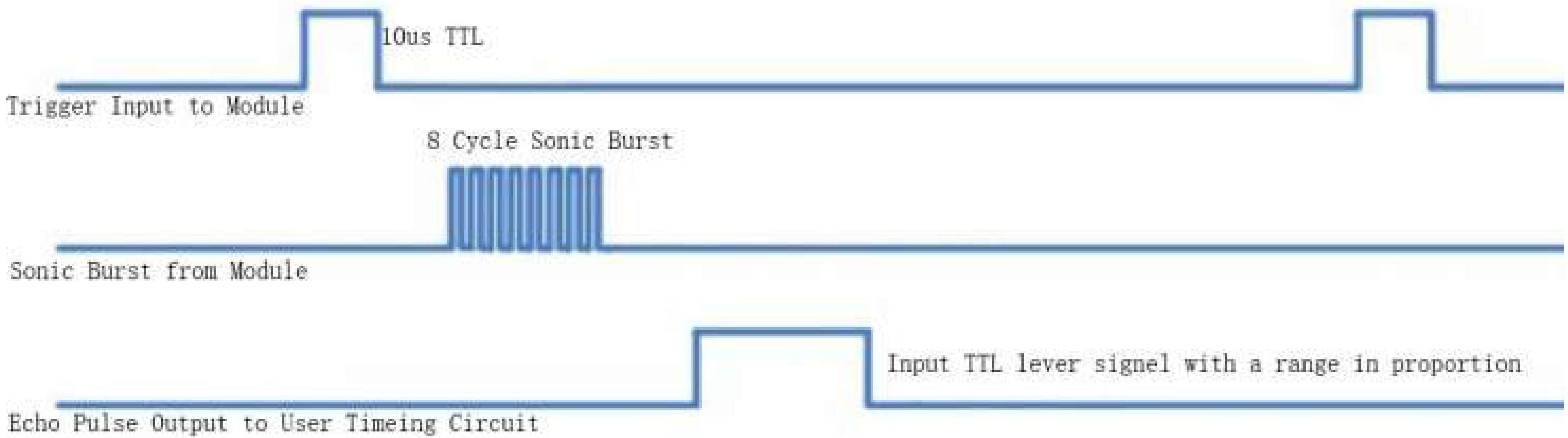
```
pi@raspberrypi: ~
GNU nano 2.7.4          File: range01.py          Modified

import RPi.GPIO as gpio
import time

# Define pin allocations
trig = 16
echo = 18

def distance():
        gpio.setmode(gpio.BOARD)
        gpio.setup(trig, gpio.OUT)
        gpio.setup(echo, gpio.IN)

        # Ensure output has no value
        gpio.output(trig, False)
        time.sleep(0.01)


^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File ^\ Replace   ^U Uncut Tex ^T To Linter
```

Timing

- Each range measurement requires (next slide):

1. Trigger HIGH for 10 microseconds
2. Sensor automatically transmits eight 40 kHz pulses
3. Received signal generates HIGH echo signal
4. Duration of echo HIGH output defines delta time

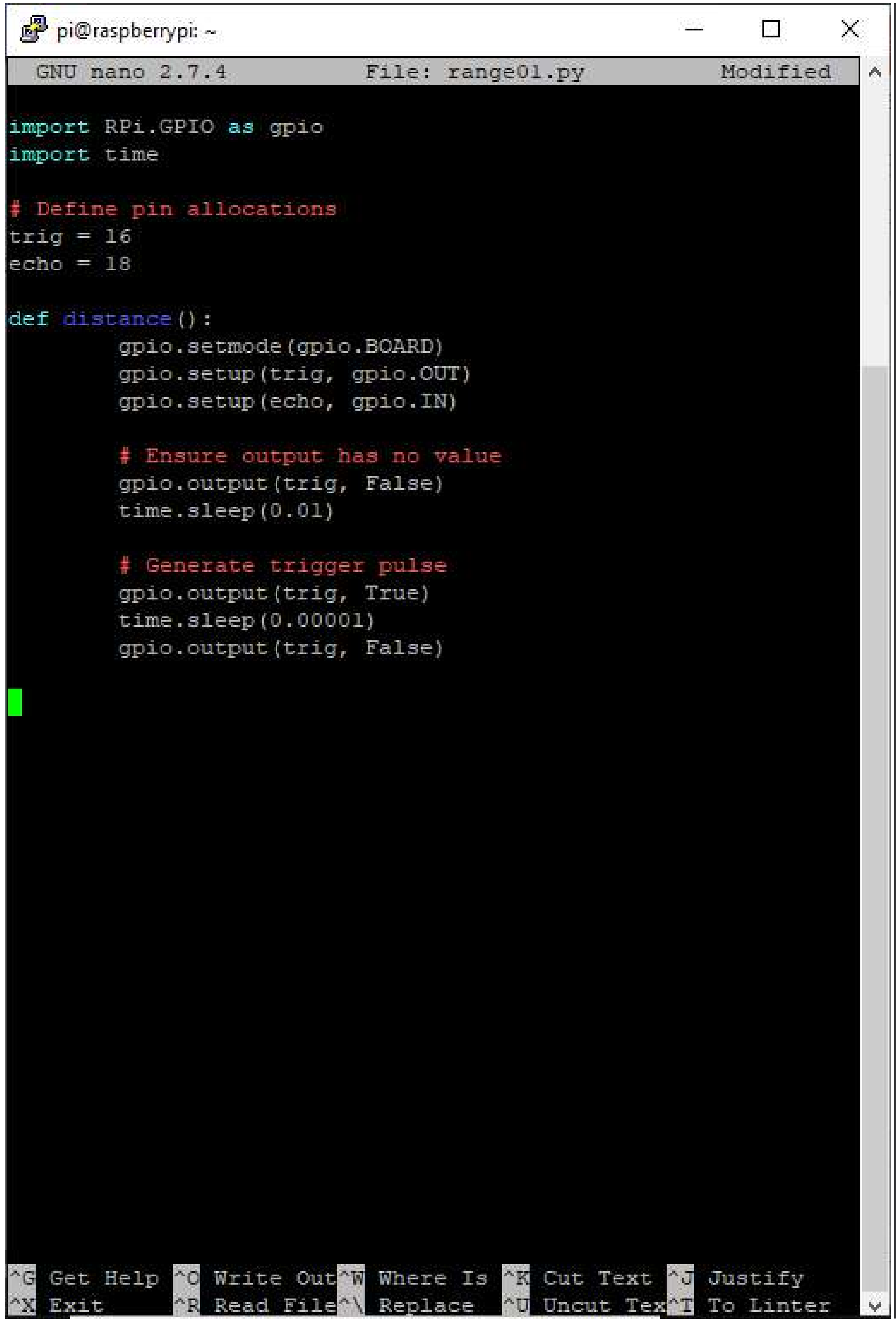
Timing



https://cdn.sparkfun.com/assets/b/3/0/b/a/DGCH-RED_datasheet.pdf

Code

- Generate trigger signal




```
import RPi.GPIO as gpio
import time

# Define pin allocations
trig = 16
echo = 18

def distance():
        gpio.setmode(gpio.BOARD)
        gpio.setup(trig, gpio.OUT)
        gpio.setup(echo, gpio.IN)

        # Ensure output has no value
        gpio.output(trig, False)
        time.sleep(0.01)

        # Generate trigger pulse
        gpio.output(trig, True)
        time.sleep(0.00001)
        gpio.output(trig, False)
```

Code

- Generate echo time signal

```
pi@raspberrypi: ~
GNU nano 2.7.4              File: range01.py              Modified

import RPi.GPIO as gpio
import time

# Define pin allocations
trig = 16
echo = 18

def distance():
        gpio.setmode(gpio.BOARD)
        gpio.setup(trig, gpio.OUT)
        gpio.setup(echo, gpio.IN)

        # Ensure output has no value
        gpio.output(trig, False)
        time.sleep(0.01)

        # Generate trigger pulse
        gpio.output(trig, True)
        time.sleep(0.00001)
        gpio.output(trig, False)

        # Generate echo time signal
        while gpio.input(echo) == 0:
                pulse_start = time.time()

        while gpio.input(echo) == 1:
                pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File ^\ Replace   ^U Uncut Tex ^T To Linter
```

Code

- Convert time measured to distance estimate
- At sea level, sound travels 343 m/sec = 34300 cm/sec

$$Speed = \frac{Distance}{Time}$$

$$34300 = \frac{Distance}{Time/2}$$

$$17150 = \frac{Distance}{Time}$$

$$17150 \times Time = Distance$$

pi@raspberrypi: ~

```
  GNU nano 2.7.4            File: range01.py            Modified

import RPi.GPIO as gpio
import time

# Define pin allocations
trig = 16
echo = 18

def distance():
        gpio.setmode(gpio.BOARD)
        gpio.setup(trig, gpio.OUT)
        gpio.setup(echo, gpio.IN)

        # Ensure output has no value
        gpio.output(trig, False)
        time.sleep(0.01)

        # Generate trigger pulse
        gpio.output(trig, True)
        time.sleep(0.00001)
        gpio.output(trig, False)

        # Generate echo time signal
        while gpio.input(echo) == 0:
                pulse_start = time.time()

        while gpio.input(echo) == 1:
                pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start

        # Convert time to distance
        distance = pulse_duration*17150
        distance = round(distance, 2)



^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File ^\ Replace   ^U Uncut Tex ^T To Linter
```

Code

- Cleanup GPIO pins
  - Resets ports used in program as inputs
  - Prevents shorts/damage
- Return distance estimate from distance( ) function

```
pi@raspberrypi: ~
 GNU nano 2.7.4              File: range01.py              Modified

import RPi.GPIO as gpio
import time

# Define pin allocations
trig = 16
echo = 18

def distance():
        gpio.setmode(gpio.BOARD)
        gpio.setup(trig, gpio.OUT)
        gpio.setup(echo, gpio.IN)

        # Ensure output has no value
        gpio.output(trig, False)
        time.sleep(0.01)

        # Generate trigger pulse
        gpio.output(trig, True)
        time.sleep(0.00001)
        gpio.output(trig, False)

        # Generate echo time signal
        while gpio.input(echo) == 0:
                pulse_start = time.time()

        while gpio.input(echo) == 1:
                pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start

        # Convert time to distance
        distance = pulse_duration*17150
        distance = round(distance, 2)

        # Cleanup gpio pins & return distance estimate
        gpio.cleanup()
        return distance


^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File ^\ Replace   ^U Uncut Tex^T To Linter
```

https://raspi.tv/2013/rpi-gpio-basics-3-how-to-exit-gpio-programs-cleanly-avoid-warnings-and-protect-your-pi

Code

- Print distance in terminal window

pi@raspberrypi: ~

```
  GNU nano 2.7.4              File: range01.py

import RPi.GPIO as gpio
import time

# Define pin allocations
trig = 16
echo = 18

def distance():
        gpio.setmode(gpio.BOARD)
        gpio.setup(trig, gpio.OUT)
        gpio.setup(echo, gpio.IN)

        # Ensure output has no value
        gpio.output(trig, False)
        time.sleep(0.01)

        # Generate trigger pulse
        gpio.output(trig, True)
        time.sleep(0.00001)
        gpio.output(trig, False)

        # Generate echo time signal
        while gpio.input(echo) == 0:
                pulse_start = time.time()

        while gpio.input(echo) == 1:
                pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start

        # Convert time to distance
        distance = pulse_duration*17150
        distance = round(distance, 2)

        # Cleanup gpio pins & return distance estimate
        gpio.cleanup()
        return distance

print("Distance: ", distance(), " cm")

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File ^\ Replace   ^U Uncut Tex ^T To Linter
```

Code

- Run program to confirm proper operation

pi@raspberrypi: ~

```
pi@raspberrypi:~ $ sudo nano range01.py
pi@raspberrypi:~ $ python3 range01.py
Distance:  34.51  cm
pi@raspberrypi:~ $
```

Code

- Modify your code to measure and provide range estimates of 10 successive range measurements at a rate of 1 Hz
- Print each successive range measurement to the terminal window

pi@raspberrypi: ~

```
pi@raspberrypi:~ $ python3 range02.py
Distance:  105.81  cm
Distance:  103.53  cm
Distance:  53.09  cm
Distance:  53.54  cm
Distance:  40.1  cm
Distance:  41.07  cm
Distance:  24.45  cm
Distance:  23.4  cm
Distance:  18.44  cm
pi@raspberrypi:~ $
```

In-Class Exercise

- Place an object ~0.5 m from the Rpi
- Write a Python script to perform the following:

1. Record an image of the scene using raspistill
2. Record 10 successive distance measurements from the RPi to the object
3. Calculate & print the average of the 10 measurements onto the image using OpenCV

```
import numpy as np
import cv2
import imutils
import RPi.GPIO as gpio
import time
import os
```

```
# Record image using Raspistill
name = "lecture4inclass.jpg"
os.system('raspistill -w 640 -h 480 -o ' + name)
```

References

- *Introduction to Autonomous Mobile Robots*, Siegwart
  - Chapter 4
- Raspberry Pi GPIO Setup
  - https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/overview
- HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi
  - https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi
- Arduino lesson – Ultrasonic Sensor HC-SR04
  - http://osoyoo.com/2017/07/23/arduino-lesson-ultrasonic-sensor-hc-sr04/