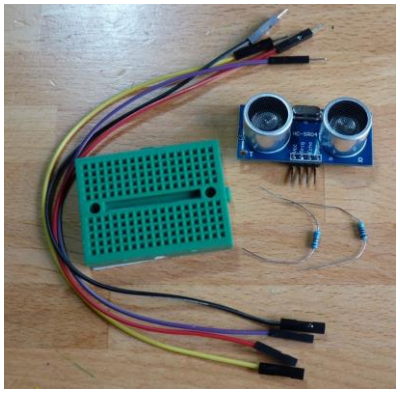
Raspberry Pi Python GPIO Zero Distance Sensor

Tutorial by Jonathan Teague - Public Domain
28th Jan 2017 - www.cotswoldjam.org

This tutorial will cover using the popular HC-SR04 ultrasonic distance sensor, the Raspberry Pi and the gpiozero library to measure distances.

The Kit



In your bag you will find the following components:

- 4 x M-F Jumper leads (pin to socket)
- 1 x R1 680Ω resistor
- 1 x R2 470Ω resistor
- 1 x Mini Breadboard
- 1 x HC-SR04 Ultrasound sensor

Putting it together:

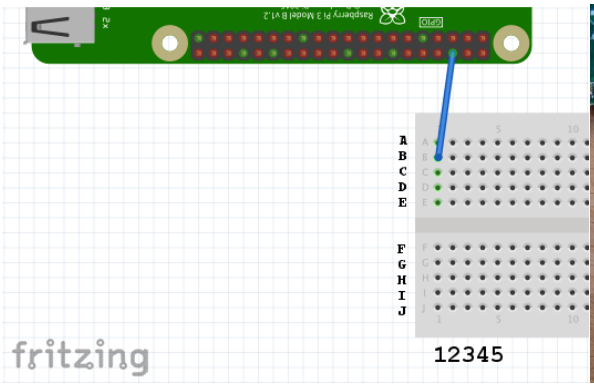
WARNING: DO NOT TURN ON THE RASPBERRY Pi UNTIL YOU HAVE FINISHED YOUR BUILD AND HAD IT CHECKED BY A TUTOR!!

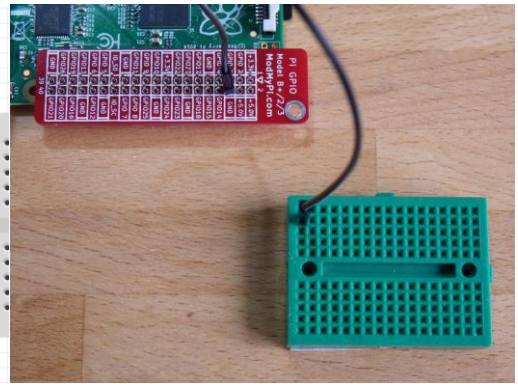
The Raspberry Pi is a 3.3V device, the HC-SR04 is a 5V device, connecting this up wrong can damage the Raspberry Pi

The colour of the jumper wires in your kit is not important – where you put them is VERY important! Breadboard holes are numbered – make sure you connect to the correct hole.

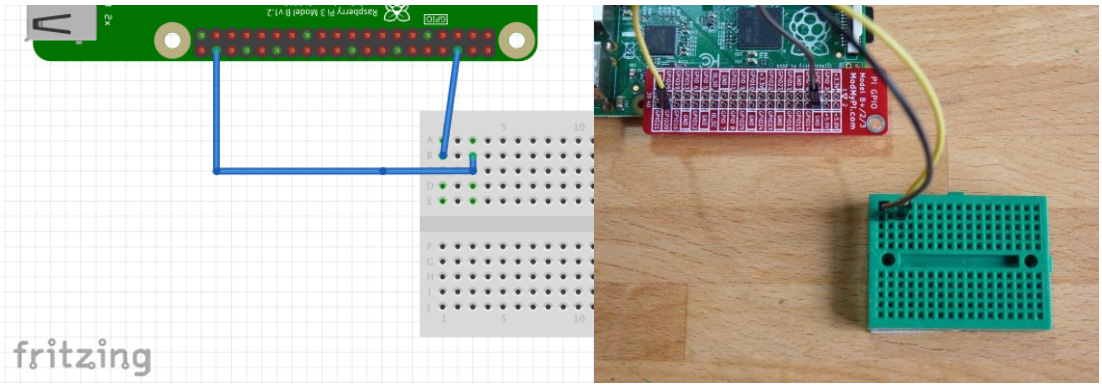
Step 0: Make sure your Pi is shut down and the power lead disconnected.

Step 1: Take 1 jumper lead and plug the pin (male end) into B1 on the breadboard and the socket (female end) into GND(Pin6) on the Pi.

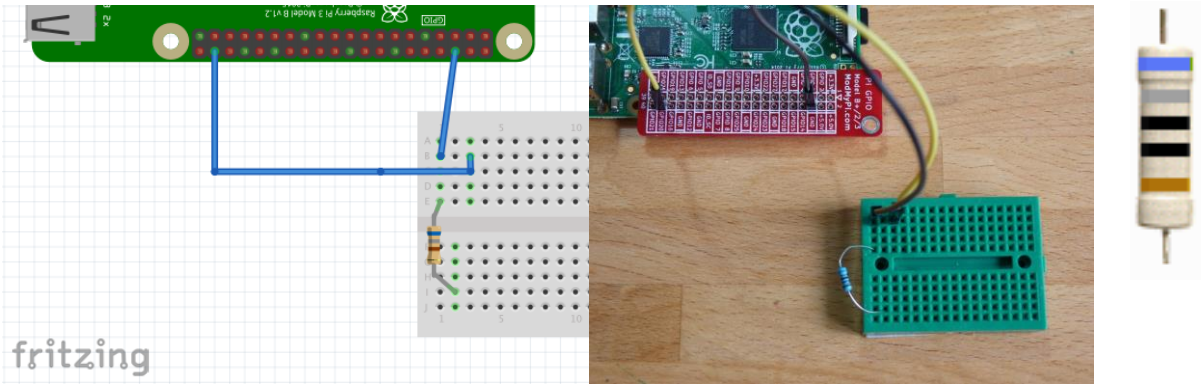




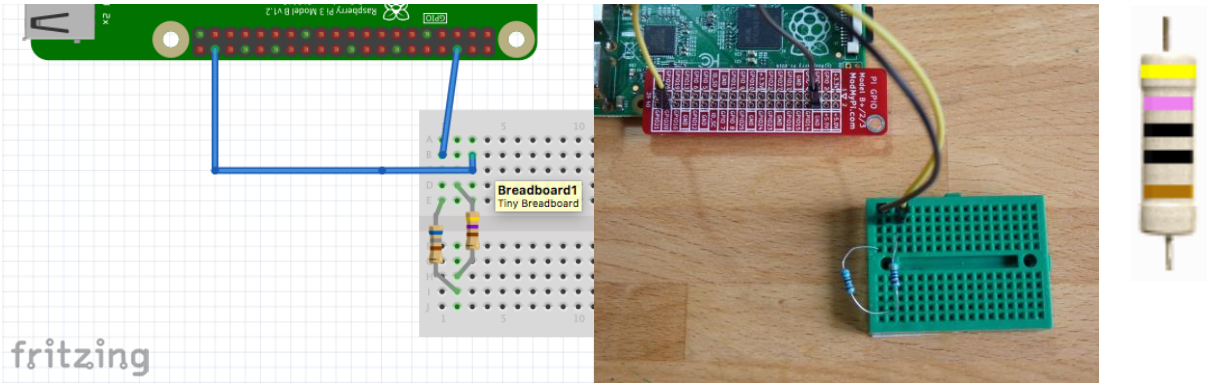
Step 2: Take 1 jumper lead and plug the pin into B3 on the breadboard and the socket into GPIO20 on the Pi. This is the connector that the Pi will use to trigger the distance measurer to take a measurment.



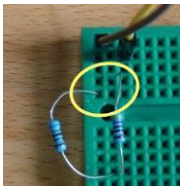
Step 3: Take the R1 680Ω resistor – the one with a blue colour line on and insert the legs into holes E1 and I2 on the breadboard – it doesn't matter which way round the resistor is.



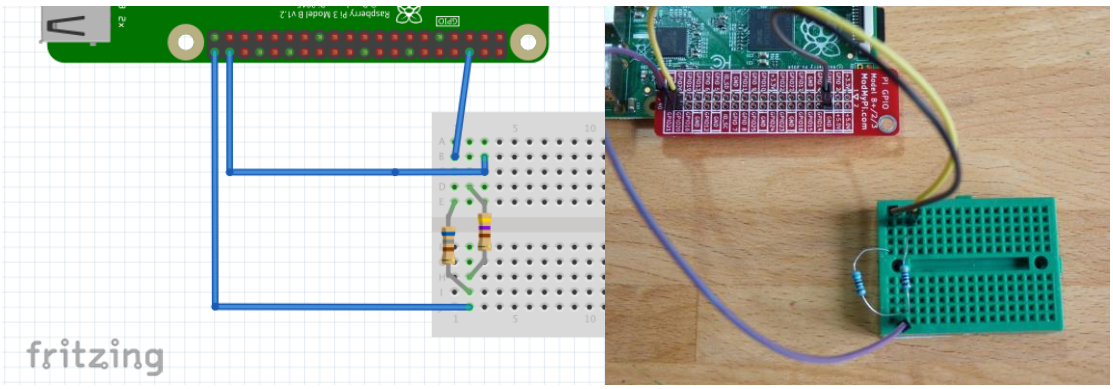
Step 4: Take the other resistor, the R2 470Ω – with a yellow colour line on and insert the legs into holes D2 and H2 on the breadboard – again it doesn't matter which way round.



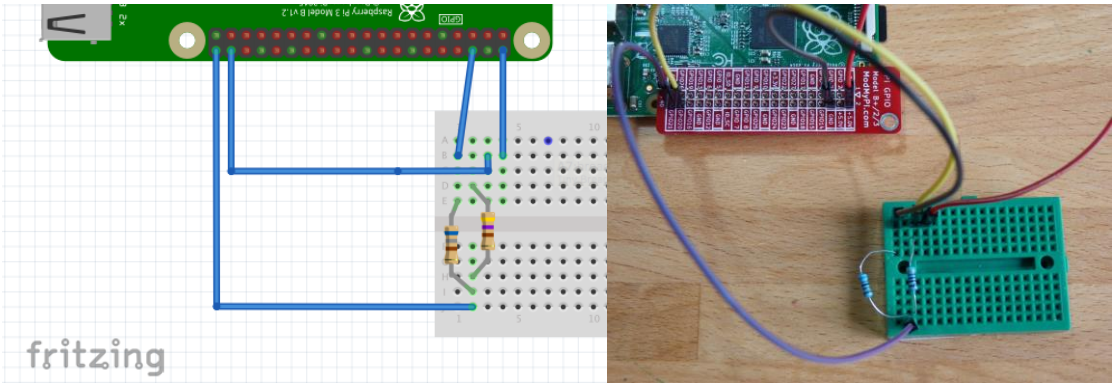
Step 5: Gently bend the R1 resistor away from R2 so the legs in E1 and D2 won't touch each other (if they do it won't damage anything, it just won't work).



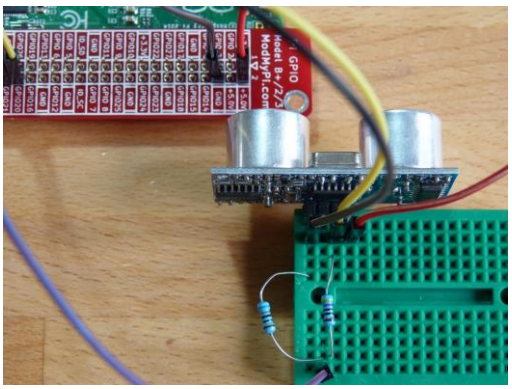
Step 6: Take 1 jumper lead and plug the pin into J2 on the breadboard and the socket into GPIO21 on the Pi. This is the connector that the Pi will use to listen for the measurement.



Step 7: Take the last jumper lead and plug the pin into B4 on the breadboard and the socket into 5V(Pin 2)



Step 8: Finally, plug in the HC-SR04. It has 4 pins and these need to go into A1, A2, A3 and A4. The back of the sensor with all the black ICs on should be facing the breadboard and the two silver sensors pointing away from the breadboard.



Now, no matter how confident you are that you've got it all right, call over a tutor and get your wiring checked. Once you're given the OK continue below to get programming.

The program

Power up your Raspberry Pi. From the desktop menu, select Programming - Python 3 (IDLE). Then use File, New File to create a new program.

Type in the following program then use File, Save to save your program as the name of your choice (don't forget to put `.py` on the end) **in the `~/python/distance` folder** and then run it with Run menu, Run Module.

NB: If you really don't want to type the program in you can use File, Open to open the already prepared `gpiozero_DistanceSensor.py` program in the `~/python/distance` folder.

```
#!/usr/bin/python

from gpiozero import DistanceSensor
from time import sleep

sensor = DistanceSensor(echo=21, trigger=20, max_distance=2.0)

while True:

  distance = sensor.distance * 100
  print("Distance : %.1f" % distance)
  sleep(1)
```

What does the program do?

```
#!/usr/bin/python
```

The first line is a comment (it starts with #) and tells the operating system what langauge the program is written in, as you will run the progam from Python yourself this is redundant (but good practice).

```
from gpiozero import DistanceSensor
from time import sleep
```

The "`from`" lines tell the computer to learn about new things. Computers can learn from programs that other people have written; we call these other programs "libraries". Our program needs the function called `DistanceSensor` from the `gpiozero` library which it will use to talk to the sensor and the function `sleep` from the `time` library so that we can insert pauses. A function is just some code located somewhere else that we can make use of to do something.

```
sensor = DistanceSensor(echo=21, trigger=20, max_distance=2.0)
```

We then create an object we call "`sensor`" using the imported function DistanceSensor. The parameters that we pass say that the "`trigger`" to invoke the sensor will be on GPIO pin 20; the "`echo`" to listen for the reply will be on GPIO pin 21 and that the maximum distance that should be returned is 2.0 metres.

```
while True:
```

This "`while True:`" tells the program to run forever in a loop.

```
  distance = sensor.distance * 100
  print("Distance : %.1f" % distance)
  sleep(1)
```

Then we get the current sensor reading by calling `sensor.distance`, multiply this by 100 (as the returned value is in mm); print out the value with only one digit after the decimal point; and finally go to sleep for a second before the "`while True:`" makes us go around again.