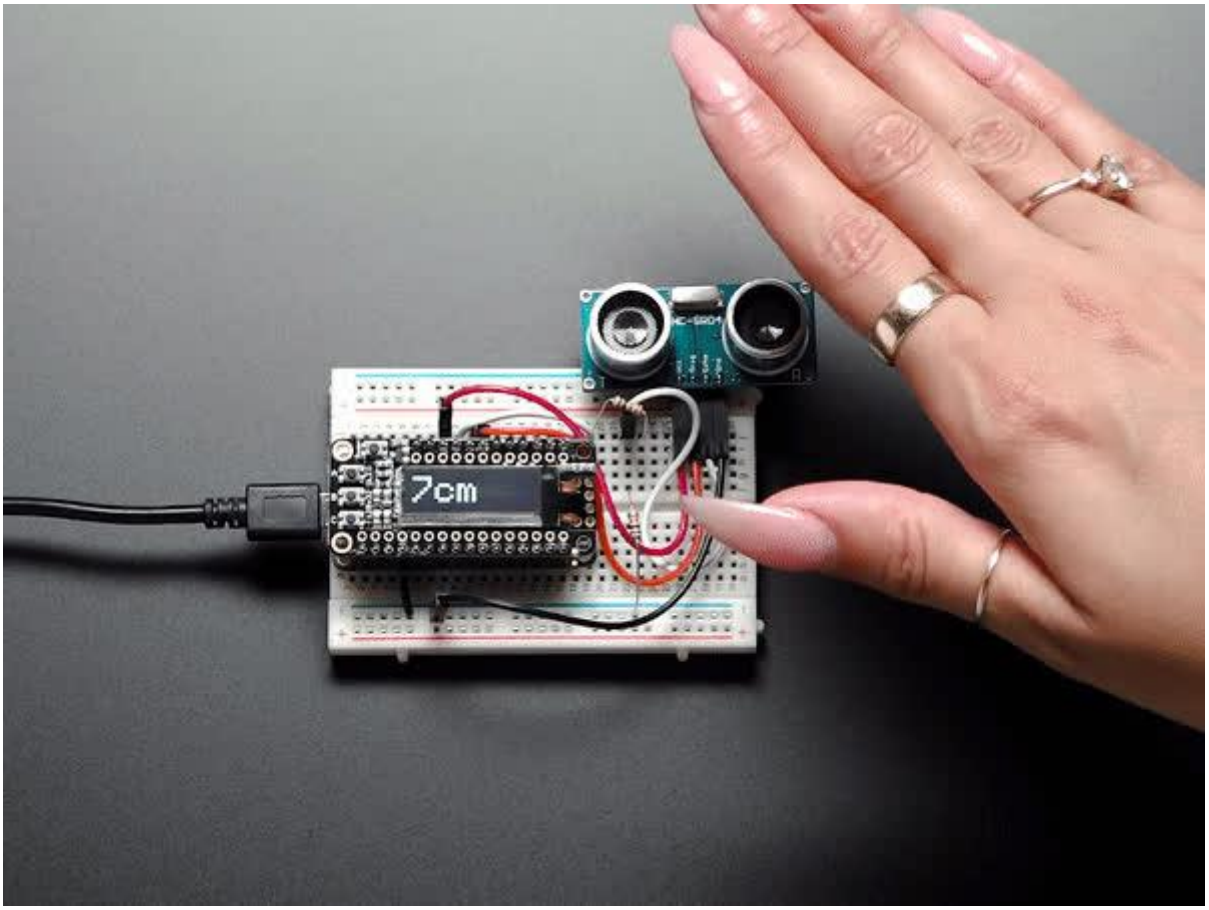
Ultrasonic Sonar Distance Sensors

Created by Kattni Rembor

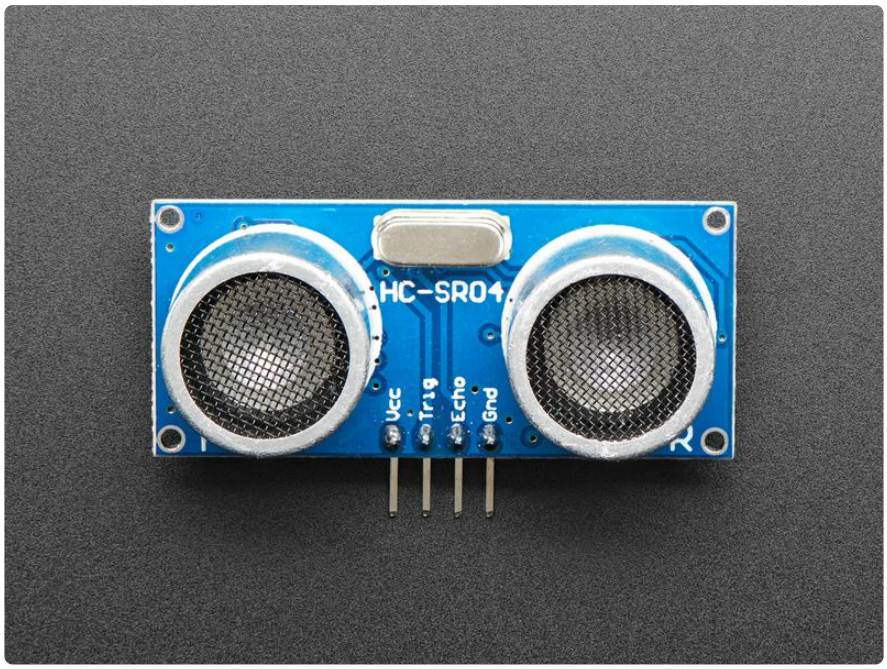


https://learn.adafruit.com/ultrasonic-sonar-distance-sensors

Last updated on 2021-11-15 07:52:30 PM EST

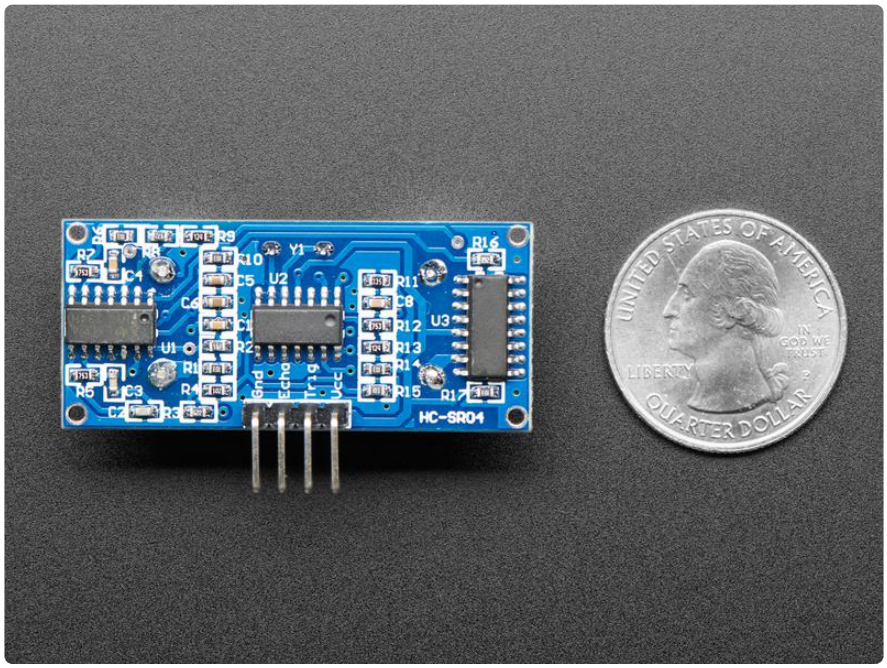
Table of Contents

Overview

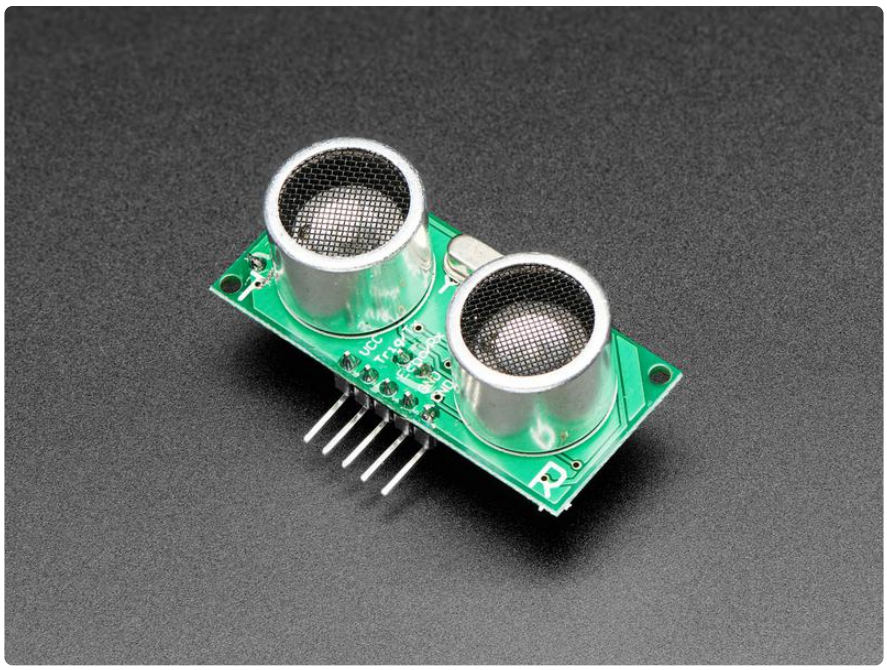


If you're like me, you've dreamed of being a dolphin - smoothly gliding through the water. Using your echo-location abilities to detect tasty fish treats. Until genetic modifications catches up with our desires, we'll just have to make do with these handy HC-SR04 and US100 Ultrasonic Sonar Distance Sensors and a pair of flippers.

These ubiquitous sensors are really common in robotics projects, but they can also be used for automation, interactive art and motion sensing. They work at about 2cm to 400cm away, but we think 10cm-250cm will get you the best results.

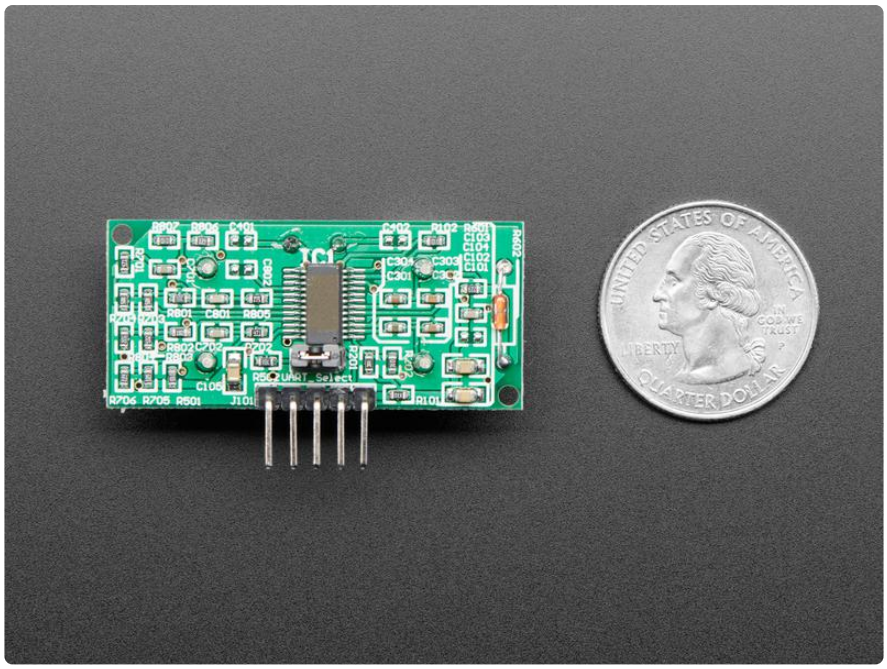


The HC-SR04 sensors are fast, fairly easy to use, and low cost. They do require powering from 5V for best results: connect GND (ground) and VCC to 5V power. While the Trig signal can be 3V or 5V, the 'return' Echo signal is 5V logic. For that reason, we include two 10K resistors, use these as a divider to convert the 5V logic level to a safe 2.5V that you can ready with your 3V device.

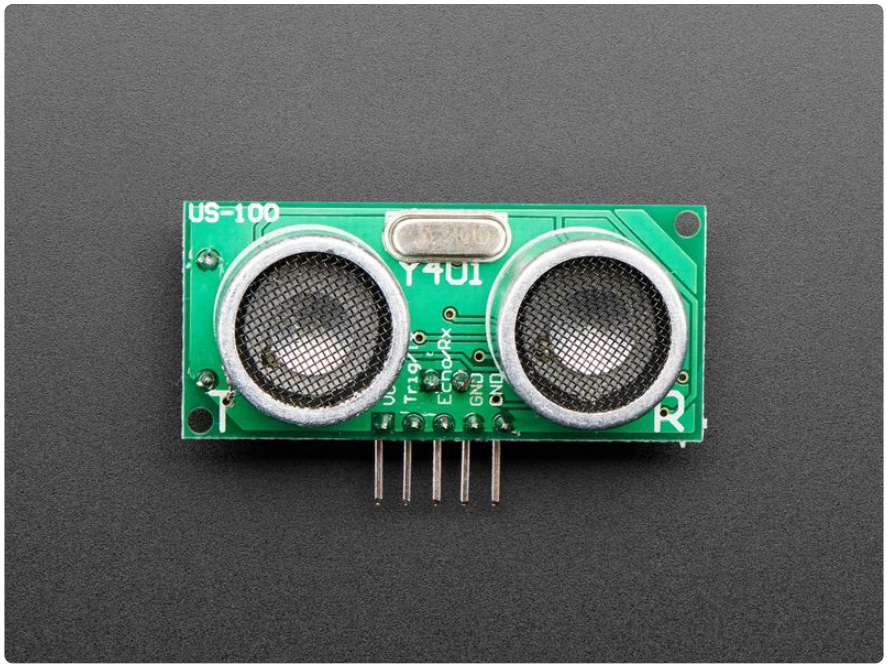


The US-100 is very similar to the popular HC-SR04 ultrasonic sensors, and even looks the same, but has a few extra tricks

- This sensor can run from 3-5V so you don't need any logic level shifters or dividers. Just power from whatever your microcontroller provides
- You can use in "HC-SR04" mode or in "Serial UART" mode.



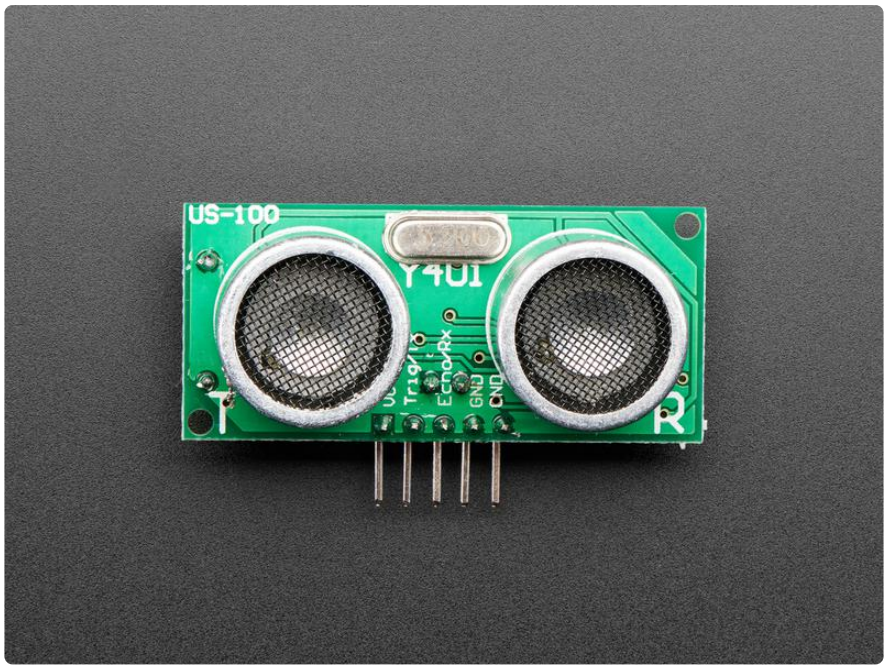
When the jumper on the back is removed, it acts like an HC-SR04 with a trigger and echo pin. When the jumper is in place, you use 9600 baud UART to communicate with the sensor. In UART mode, send 0x55 and read back two bytes (16 bit value) that is mm distance, or 0x50 to read the temperature in degrees C. Handy if you want to use with a computer and a USB-serial converter, or some other device that can't do the special timing needed for the HC-SR04 trigger/echo.



Pinouts

The HC-SR04 sensors are simple to use. Let's take a look!

- Vcc - Power pin - this sensor requires 5V power for best results.
- Trig - Signal can be 3V or 5V. Setting the Trig pin to high for 10µs causes the sensor to initiate an ultrasonic burst.
- Echo - "Return" signal is 5V logic - the sensor comes with two 10k resistors to use as a divider to convert the 5V logic level to a safe 2.5V that you can read with your 3V device. The Echo pin goes high when the ultrasonic burst is transmitted (in response to Trig) and stays high until the sensor receives an echo of its burst, at which point it goes low. By measuring the time the Echo pin is high, the distance can be computed.
- Gnd - Ground pin.



The US100 is nearly the same except it is capable of UART.

- Vcc - Power pin - this sensor can run at 3V or 5V.
- Trig/TX - Signal can be 3V or 5V in HC-SR04 compatibility mode.. Works as TX in UART mode. Setting the Trig pin to high for 10µs causes the sensor to initiate an ultrasonic burst.
- Echo/RX - Signal can be 3V or 5V in HC-SR04 compatibility mode. Works as RX in UART mode. The Echo pin goes high when the ultrasonic burst is transmitted (in response to Trig) and stays high until the sensor receives an echo of its burst, at which point it goes low. By measuring the time the Echo pin is high, the distance can be computed.
- Gnd - Ground pin.
- Gnd - Ground pin.

Python & CircuitPython

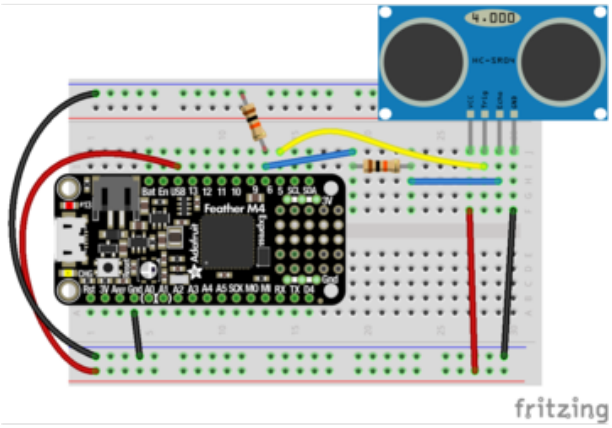
It's easy to use these ultrasonic distance sensors with Python and CircuitPython. These modules allow you to easily use Python code to read the distance from the sensor.

You can use the US-100 sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

First wire up your sensor as shown below.

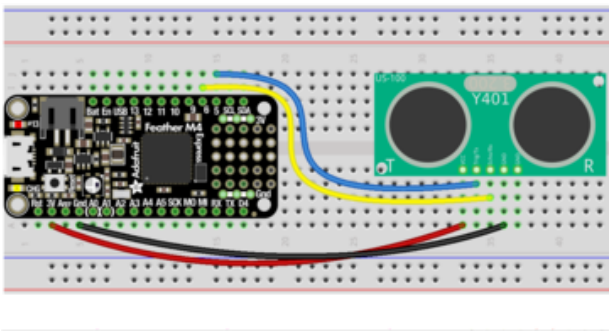
Here is an example of the HC-SR04 wired up to a Feather M4:



- Feather USB or VBat to sensor VCC
- Feather D5 to sensor trig
- Feather D6 to 10k resistor - other side to GND
- Sensor echo to 10k resistor - other side to D6
- Feather and Sensor GND to breadboard ground rail

The HC-SR04 requires a voltage divider to allow the 5V logic to work with a 3V microcontroller.

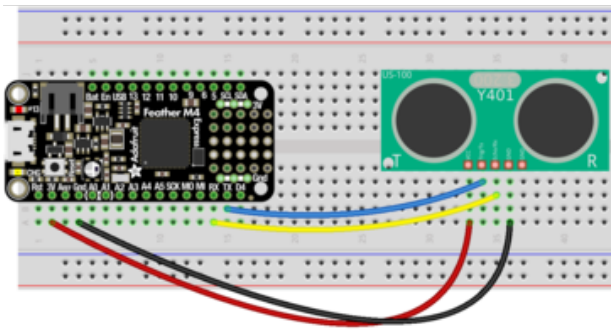
Here is an example of the US-100 wired up to a Feather M4 in HC-SR04 compatibility mode:



- Feather 3V to sensor VCC
- Feather GND to sensor GND
- Feather D5 to sensor trig/TX
- Feather D6 to sensor echo/RX
- Jumper removed from back of sensor

Here is an example of the US-100 wired up to a Feather M4 in UART mode:

Note: Typically with UART, you connect TX to RX and RX to TX, this is not the case with this sensor!

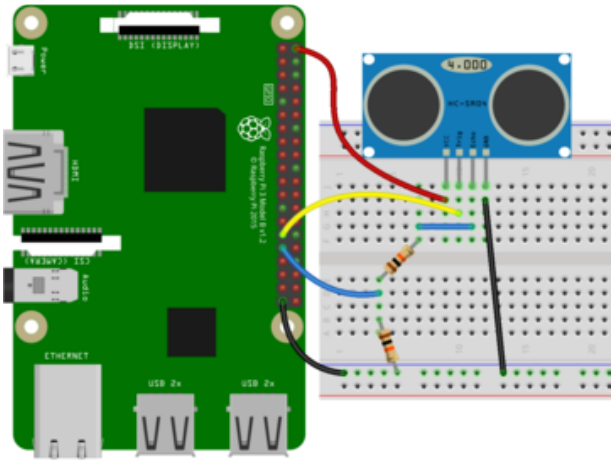


- Feather 3V to sensor VCC
- Feather GND to sensor GND
- Feather TX to sensor trig/TX
- Feather RX to sensor echo/RX
- Jumper present on back of sensor

Python Computer Wiring

Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported](https://adafru.it/BSN) (https://adafru.it/BSN).

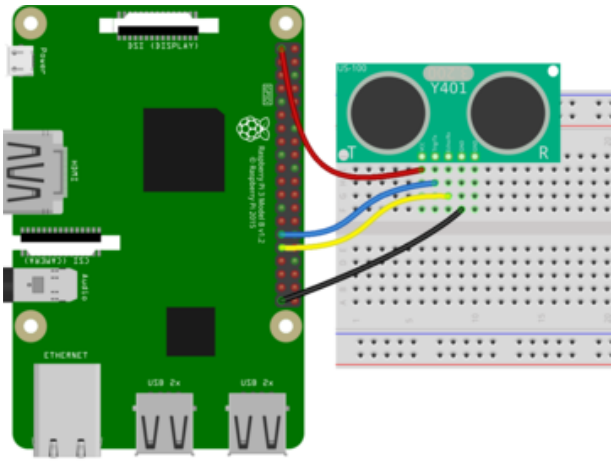
Here is an example of the HC-SR04 wired up to a Raspberry Pi:



- Pi 5v to sensor VCC
- Pi D5 to sensor trig
- Pi D6 to 10k resistor - other side to GND
- Sensor echo to 10k resistor - other side to D6
- Pi GND to breadboard ground rail
- Sensor GND to breadboard ground rail

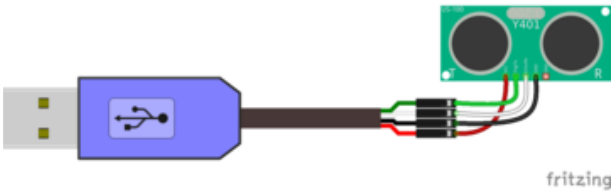
The HC-SR04 requires a voltage divider to allow the sensor 5V logic to work with 3V logic.

Here's an example of the US-100 in HC-SR04 compatibility mode wired up to a Raspberry Pi:



- Pi 3.3V to sensor VCC
- Pi GND to sensor GND
- Pi D5 to sensor trig/TX
- Pi D6 to sensor echo/RX
- Jumper removed from back of sensor

You have two options for the US100 in UART mode: An external USB-to-serial converter, or the built-in UART on the Pi's TX/RX pins. Here's an example of wiring up the US100 to a USB-to-serial converter (https://adafru.it/dDd):

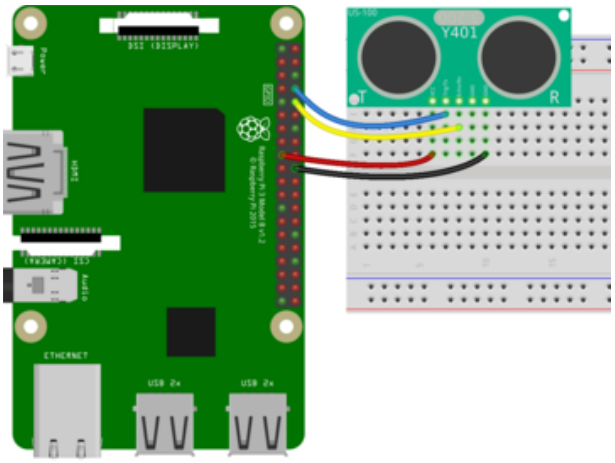


- USB serial 5V to sensor VCC
- USB serial GND to sensor GND
- USB serial TX to sensor trig/TX
- USB serial RX to sensor echo/RX

Note: Typically with UART, you connect TX to RX and RX to TX, this is not the case with this sensor!

For single board computers other than the Raspberry Pi, the serial port may be tied to the console or not be available to the user. Please see the board documentation to see how the serial port may be used

Here's an example with the US100 using the Pi's built-in UART:



- Pi 3V to sensor VCC
- Pi GND to sensor GND
- Pi TX to sensor trig/TX
- Pi RX to sensor echo/RX

Note: Typically with UART, you connect TX to RX and RX to TX, this is not the case with this sensor!

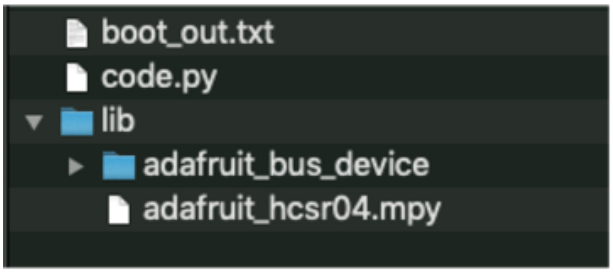
If you want to use the built-in UART, you'll need to disable the serial console and enable the serial port hardware in raspi-config. See the UART/Serial section of the CircuitPython on Raspberry Pi guide (https://adafru.it/CEk) for detailed instructions on how to do this.

CircuitPython Installation of HC-SR04 / US-100 Library

To use the HC-SR04, you'll need to install the Adafruit CircuitPython HCSR04 (https://adafru.it/CTc) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

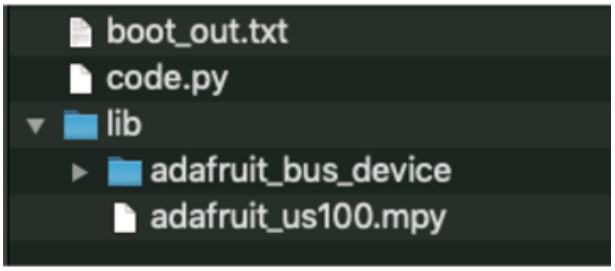
Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/ENC). Our introduction guide has a great page on how to install the library bundle (https://adafru.it/ABU) for both express and non-express boards.



For the HCSR04 - You'll want to copy the necessary libraries from the bundle to your lib folder on your CIRCUITPY drive:

- adafruit_hcsr04.mpy
- adafruit_bus_device

Before continuing make sure your board's lib folder has the adafruit_hcsr04.mpy, and adafruit_bus_device files and folders copied over.



For the US100 you'll want to copy the necessary libraries from the bundle to your lib folder on your CIRCUITPY drive:

- adafruit_us100.mpy
- adafruit_bus_device

Before continuing make sure your board's lib folder has the adafruit_us100.mpy, and adafruit_bus_device files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz) so you are at the CircuitPython >>> prompt.

Python Installation of HC-SR04 / US100 Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

Once that's done, from your command line run the following command

- `sudo pip3 install adafruit-circuitpython-hcsr04 adafruit-circuitpython-us100`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython & Python Usage of HC-SR04

The following works for both the HC-SR04, and the US-100 in HC-SR04 compatibility mode.

To demonstrate usage of these sensors, we will initialise it and read the distance using the board's Python REPL.

Run the following code to import the necessary modules and initialize the connection with the sensor:

```
import time
import board
import adafruit_hcsr04
sonar = adafruit_hcsr04.HCSR04(trigger_pin=board.D5, echo_pin=board.D6)
```

Now you're ready to read the values from the sensor using the following property:

- distance - The distance measured by the sensor in cm.

```
while True:
    try:
        print((sonar.distance,))
    except RuntimeError:
        print("Retrying!")
    time.sleep(0.1)
```

That's all there is to using the HC-SR04 and the US-100 in HC-SR04 compatibility mode with CircuitPython or Python!

CircuitPython & Python Usage of US-100

To demonstrate the usage of this sensor, we will initialise it and read the distance using the board's Python REPL.

For use on a microcontroller, run the following code to import the necessary modules and initialise the connection with the sensor:

```
import board
import busio
import adafruit_us100
uart = busio.UART(board.TX, board.RX, baudrate=9600)
us100 = adafruit_us100.US100(uart)
```

For use on a computer with a USB-to-serial cable, run the following code:

```
import serial
import adafruit_us100
uart = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=1)
us100 = adafruit_us100.US100(uart)
```

For use on Raspberry Pi/Linux using hardware UART, run the following code:

```
import serial
import adafruit_us100
uart = serial.Serial("/dev/ttyS0", baudrate=9600, timeout=1)
us100 = adafruit_us100.US100(uart)
```

Now you're ready to read the values from the sensor using the following properties:

- distance - the distance measured by the sensor in cm.
- temperature - the on-chip temperature in Celsius.

```
print("Distance: ", us100.distance)
print("Temperature: ", us100.temperature)
```

```
>>> print("Distance: ", us100.distance)
Distance:  25.6
>>> print("Temperature: ", us100.temperature)
Temperature:  24
```

Full Example Code

For HC-SR04:

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_hcsr04

sonar = adafruit_hcsr04.HCSR04(trigger_pin=board.D5, echo_pin=board.D6)

while True:
    try:
        print((sonar.distance,))
    except RuntimeError:
        print("Retrying!")
    time.sleep(0.1)
```

For US-100:

Note: The delay between the `temperature` reading and the `distance` reading is required for use on Raspberry Pi/Linux. Due to the increased speed over a microcontroller, the `distance` reading occurs too quickly after the `temperature` reading and the sensor doesn't process the response properly. If you find the code is hanging on the `distance` reading, ensure you have the delay!

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time

# For use with a microcontroller:
import board
import busio
import adafruit_us100

uart = busio.UART(board.TX, board.RX, baudrate=9600)

# For use with USB-to-serial cable:
# import serial
# import adafruit_us100
# uart = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=1)

# For use with Raspberry Pi/Linux:
# import serial
```

```
# import adafruit_us100
# uart = serial.Serial("/dev/ttyS0", baudrate=9600, timeout=1)

us100 = adafruit_us100.US100(uart)

while True:
    print("-----")
    print("Temperature: ", us100.temperature)
    time.sleep(0.5)
    print("Distance: ", us100.distance)
    time.sleep(0.5)
```

Python Docs

Python Docs (https://adafru.it/GED)

Downloads

Files:

- HC-SR04 Datasheet (https://adafru.it/HiC)
- US-100 Datasheet (https://adafru.it/HiD)
- HC-SR04 Fritzing object at Fritzing.org (https://adafru.it/HiE)
- US-100 Fritzing object on GitHub (https://adafru.it/HiF)