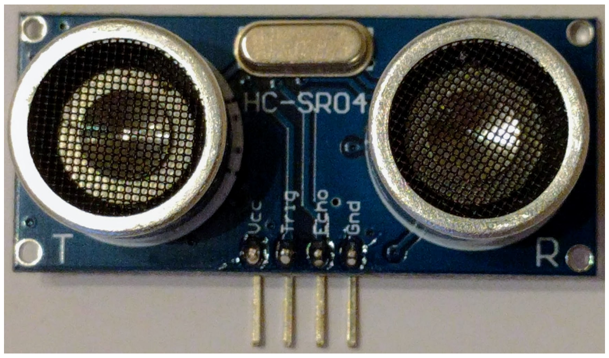
Adafruit HCSR04 Library Documentation

Release 0.2

Mike Mabey

Oct 02, 2018

Contents



The HC-SR04 is an inexpensive solution for measuring distances using microcontrollers. This library provides a simple driver for controlling these sensors from CircuitPython.

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle.

CHAPTER 2

Usage Example

Warning: The HC-SR04 uses 5V logic, so you will have to use a level shifter between it and your CircuitPython board (which uses 3.3V logic).

Note: If you want to use an HC-SR04 with MicroPython, I recommend checking out this library.

You'll need to dedicate two pins to communicating with the HC-SR04. The sensor communicates in a very rudimentary manner, so it doesn't matter which pins you choose, as long as they're digital IO pins (pins that start with "`D`" are digital).

There are two ways of instantiating a `HCSR04` object: with or without using a context manager.

Note: It is technically possible to communicate with the HC-SR04 using only one wire since the trigger and echo signals aren't ever active at the same time. Once I have a chance to determine a safe way to do this, I plan to add this as a feature to the library.

See also:

Adafruit's guide on Lifetime and ContextManagers Gives more info on using context managers with CircuitPython drivers.

`board` A list of pins available on your device. To view this list, first get a REPL (the guide linked was written for the pyboard, but it still works), then input the following:

```python
import board
dir(board)
```

2.1 Without a Context Manager

In the example below, we create the `HCSR04` object directly, get the distance every 2 seconds, then de-initialize the device.

```
from hcsr04 import HCSR04
sonar = HCSR04(trig, echo)
try:
    while True:
        print(sonar.dist_cm())
        sleep(2)
except KeyboardInterrupt:
    pass
sonar.deinit()
```

2.2 With a Context Manager

In the example below, we use a context manager (the `with` statement) to create the `HCSR04` instance, again get the distance every 2 seconds, but then the context manager handles de-initializing the device for us.

```
from hcsr04 import HCSR04
with HCSR04(trig, echo) as sonar:
    try:
        while True:
            print(sonar.dist_cm())
            sleep(2)
    except KeyboardInterrupt:
        pass
```

CHAPTER 3

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 4

Building locally

4.1 Zip release files

To build this library locally you'll need to install the circuitpython-build-tools package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-hcsr04 --library_
↪location .
```

4.2 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the index.html in your browser to view them. It will also (due to -W) error out on any warning like Travis will. This is a good way to locally verify it will pass.

CHAPTER 5

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/hcsr04_simpletest.py

```python
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)
```

5.2 adafruit_hcsr04

A CircuitPython library for the HC-SR04 ultrasonic range sensor.

The HC-SR04 functions by sending an ultrasonic signal, which is reflected by many materials, and then sensing when the signal returns to the sensor. Knowing that sound travels through dry air at 343.2 meters per second (at 20 °C), it's pretty straightforward to calculate how far away the object is by timing how long the signal took to go round-trip and do some simple arithmetic, which is handled for you by this library.

Warning: The HC-SR04 uses 5V logic, so you will have to use a level shifter or simple voltage divider between it and your CircuitPython board (which uses 3.3V logic)

- Authors:
  - Mike Mabey
  - Jerry Needell - modified to add timeout while waiting for echo (2/26/2018)
  - ladyada - compatible with *distance* property standard, renaming, Pi compat

class adafruit_hcsr04.**HCSR04**(*trigger_pin*, *echo_pin*, ***, *timeout=0.1*)

Control a HC-SR04 ultrasonic range sensor.

Example use:

```
import time
import board

import adafruit_hcsr04

sonar = adafruit_hcsr04.HCSR04(trigger_pin=board.D2, echo_pin=board.D3)


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

Parameters

- **trigger_pin** – The pin on the microcontroller that's connected to the `Trig` pin on the HC-SR04.
- **echo_pin** (*microcontroller.Pin*) – The pin on the microcontroller that's connected to the `Echo` pin on the HC-SR04.
- **timeout** (*float*) – Max seconds to wait for a response from the sensor before assuming it isn't going to answer. Should *not* be set to less than 0.05 seconds!

deinit()

De-initialize the trigger and echo pins.

distance

Return the distance measured by the sensor in cm.

This is the function that will be called most often in user code. The distance is calculated by timing a pulse from the sensor, indicating how long between when the sensor sent out an ultrasonic signal and when it bounced back and was received again.

If no signal is received, we'll throw a RuntimeError exception. This means either the sensor was moving too fast to be pointing in the right direction to pick up the ultrasonic signal when it bounced back (less likely), or the object off of which the signal bounced is too far away for the sensor to handle. In my experience, the sensor can detect objects over 460 cm away.

Returns Distance in centimeters.

Return type float

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index