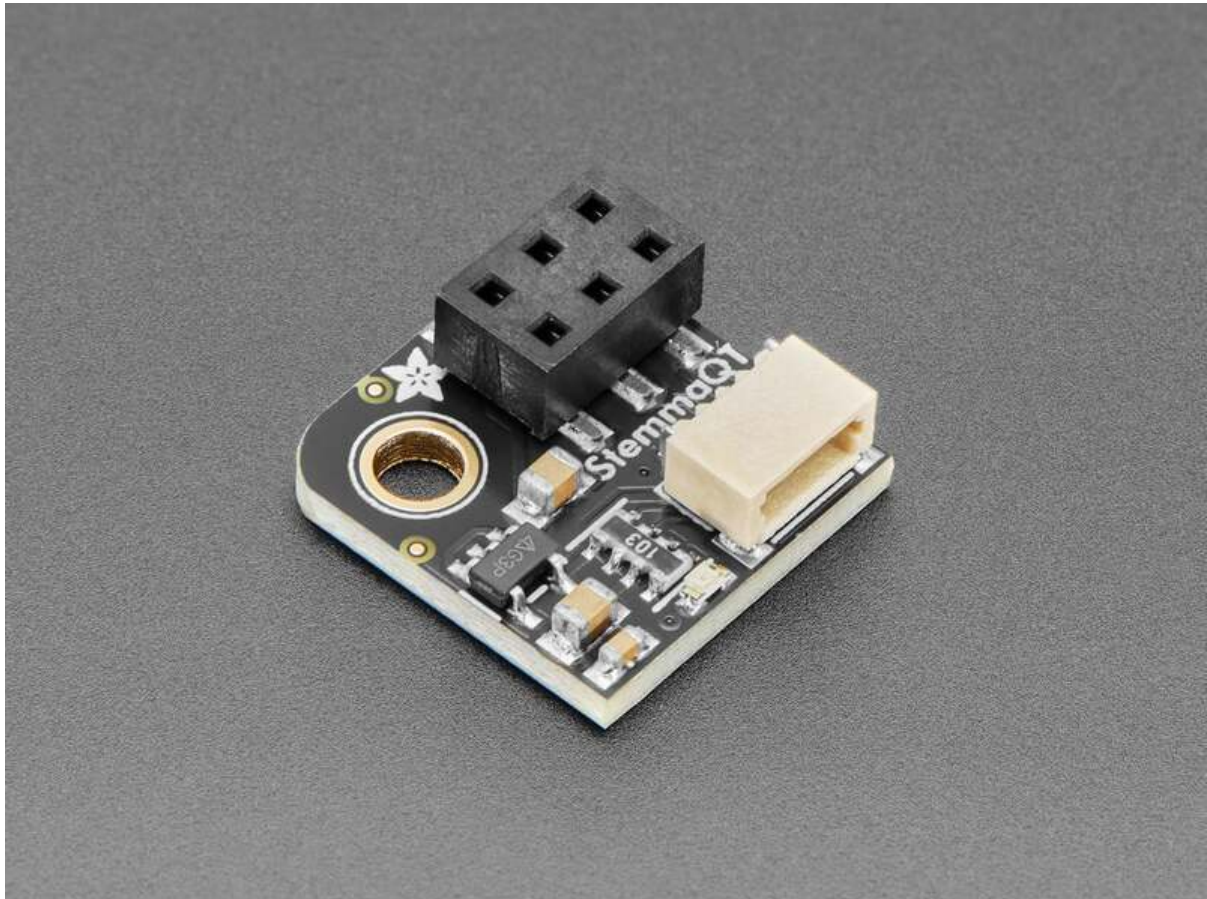




Adafruit Pi Stemma QT Breakout

Created by Liz Clark



<https://learn.adafruit.com/adafruit-pi-stemma-qt-breakout>

Last updated on 2025-07-16 03:42:19 PM EDT

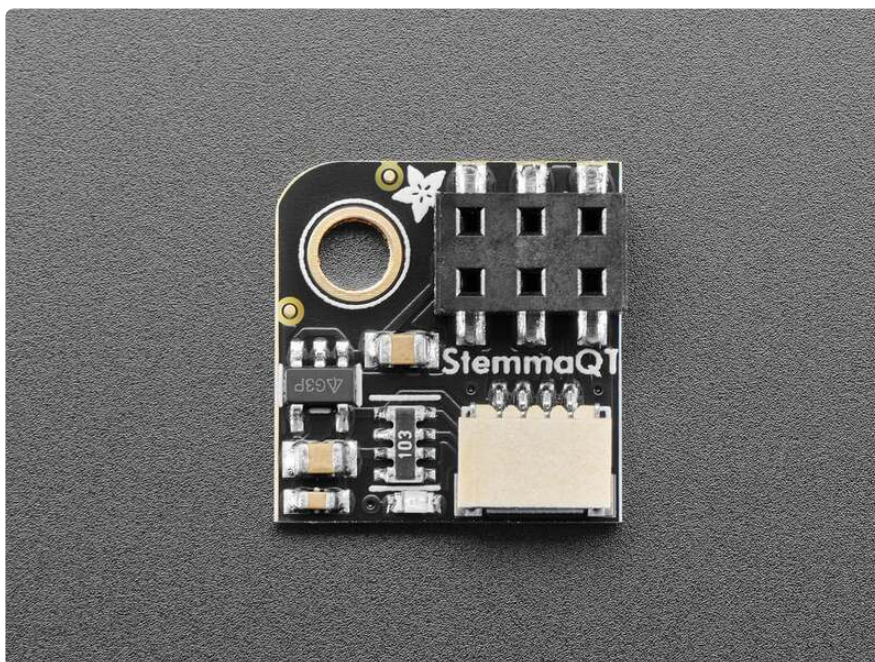
Table of Contents

Overview	3
Pinouts	5
<ul style="list-style-type: none">• 2x3 Header• STEMMA QT Connector• Power LED and Jumper	
I2C Scan - Terminal	6
<ul style="list-style-type: none">• Normal Behavior• Missing Device or Pull Ups	
I2C Scan - Python with Blinka	7
<ul style="list-style-type: none">• Python Computer Wiring• Python Usage• Example Code	
Downloads	9
<ul style="list-style-type: none">• Files• Schematic and Fab Print	

Overview



The **Adafruit Pi Stemma** is a small, easily removable breakout that easily adds a 4-pin JST SH pin (Stemma QT or [Qwiic](https://adafru.it/1anF) (<https://adafru.it/1anF>)) connector to your Raspberry Pi. The 2x3 socket design allows you to plug securely into the Pi's I2C bus with no soldering required. It works great if you want to add a [QT connector](https://adafru.it/1anG) (<https://adafru.it/1anG>) for quick I2C sensor and devices integration without any header-fiddling and no soldering.



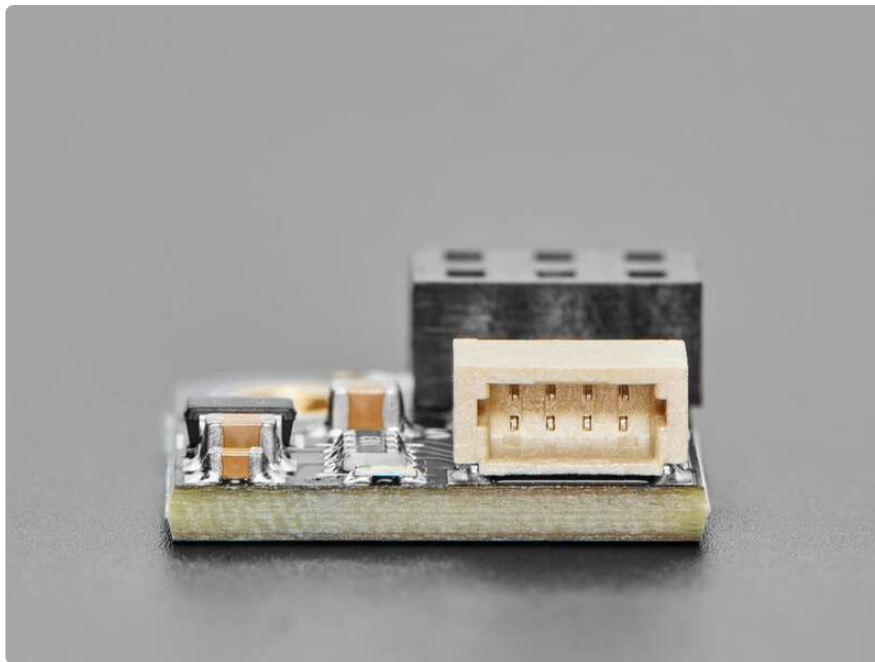
Pi Stemma has a 2x3 header on top that connects to the power and I2C pins, simply line up with the edge of the 2x20 connector on your Pi or Pi Compatible. It has a

single JST SH connector that connects to your Pi's I2C bus (SDA, SCL, 3V, and Ground). It also features a 3.3V regulator, allowing it to work with all Qwiic or Stemma QT devices without risking a brownout dip. A green LED lets you know it's powered correctly. We also added two 10K pullup resistors since there are some Pi compatible boards that don't have them included.

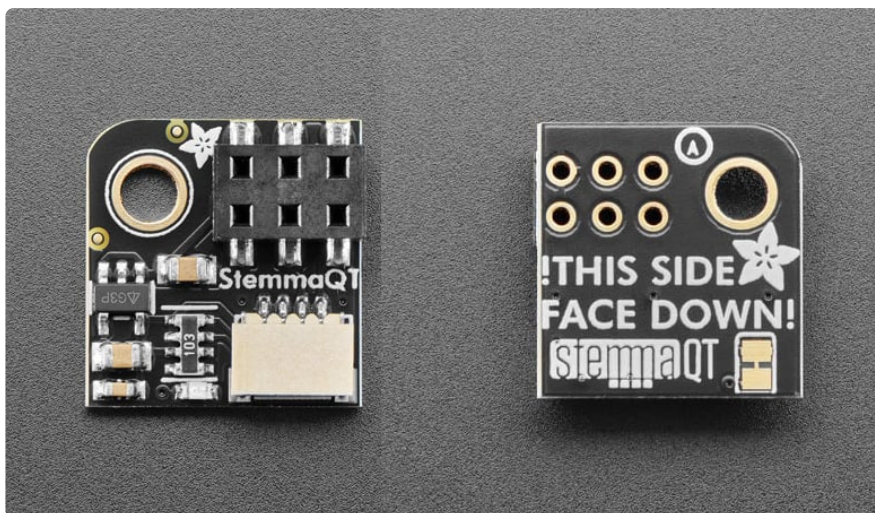


You can use it with any Linux single board computer or microcontroller that has '2x20 Pi-compatible pinout' (there are a lot!) and then [check out Blinka, our Python interface library](https://adafru.it/EA8), (<https://adafru.it/EA8>) so [you can use our huge collection of CircuitPython libraries on SBC computers using Python](https://adafru.it/Tra) (<https://adafru.it/Tra>)!

[Check out all our STEMMA QT offerings here, including a wide range of lengths and connectors](https://adafru.it/1anH) (<https://adafru.it/1anH>)



Pinouts



2x3 Header

At the top edge of the breakout is a 2x3 header that connects to the power and I2C pins on your Raspberry Pi or Pi Compatible. Line up with the edge of your chosen single board computer 2x20 GPIO connector to plug it in.

STEMMA QT Connector

- [STEMMA QT \(https://adafru.it/Ft4\)](https://adafru.it/Ft4) - This connector, on the bottom edge of the breakout, allows you to connect to sensors and breakout boards with STEMMA QT / Qwiic connectors.
- **SDA/SCL** - The I2C pins for the STEMMA QT connector are connected to the default I2C port on the Pi (GPIO 2 and 3).

- **3.3V/GND** - The power for the STEMMA QT connector is 3.3V. Ground is the common ground for power and logic.

Power LED and Jumper

- **Power LED** - To the left of the STEMMA QT port along the bottom edge of the board is the power LED. It is a green LED.
- **LED jumper** - This jumper is located on the back of the board and is outlined in white on the board silk. Cut the trace on this jumper to cut power to the power LED.

I2C Scan - Terminal

To do an I2C scan on a Raspberry Pi the `i2cdetect` command is used. If not already done, be sure to enable I2C on the Raspberry Pi via `raspi-config`. If the `i2cdetect` command is not found, install it with:

```
sudo apt-get install i2c-tools
```

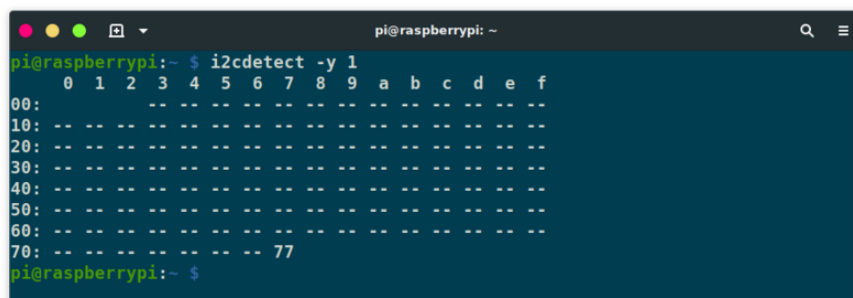
And then to run a scan, use `i2cdetect` with the following command line parameters:

```
i2cdetect -y 1
```

On modern Raspberry Pi OS releases, you do **not** need to run the command with `sudo`. The `-y` disables interactive mode, so it just goes ahead and scans. The `1` specifies the I2C bus.

Normal Behavior

If all goes well, you should get a list of addresses for each device found. In the example below, an [Adafruit BMP280 breakout](http://adafru.it/2651) (<http://adafru.it/2651>) is attached to a Raspberry Pi 4.

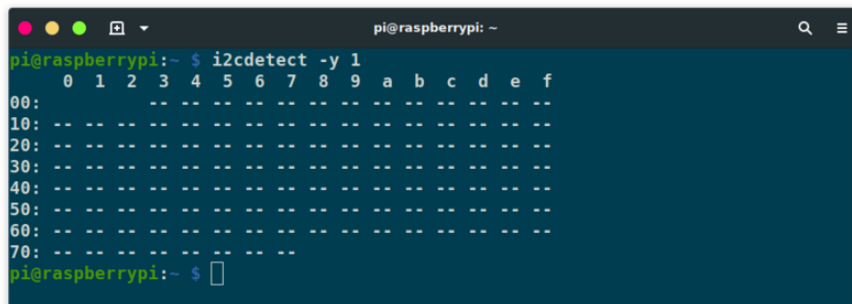


```
pi@raspberrypi: ~  
pi@raspberrypi:~$ i2cdetect -y 1  
  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
70:  --  --  --  --  --  --  --  --  --  --  77  --  --  --  --  
pi@raspberrypi:~$
```

The BMP280's I2C address of `0x77` shows up as expected.

Missing Device or Pull Ups

If the device is disconnect and/or the pull up resistors are missing, the results will look like this:



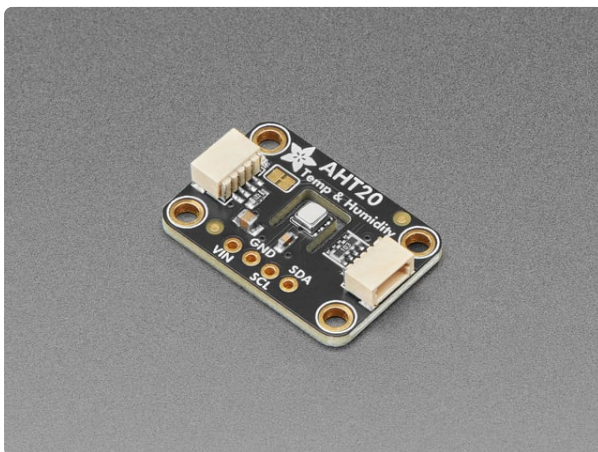
```
pi@raspberrypi:~$ i2cdetect -y 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi:~$
```

The scan runs fine, but no addresses are shown. Keep in mind that Raspberry Pi's include pull up resistors on the SCL and SDA pins.

I2C Scan - Python with Blinka

It's easy to use the **Pi Stemma QT Breakout** with Python and Blinka. To demo the I2C functionality, you can run an I2C scan with one of many STEMMA QT / Qwiic compatible boards like the AHT20 STEMMA QT breakout below.

You can use this breakout with a computer that has GPIO and Python [thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library \(https://adafru.it/BSN\)](https://adafru.it/BSN).



Adafruit AHT20 - Temperature & Humidity Sensor Breakout Board

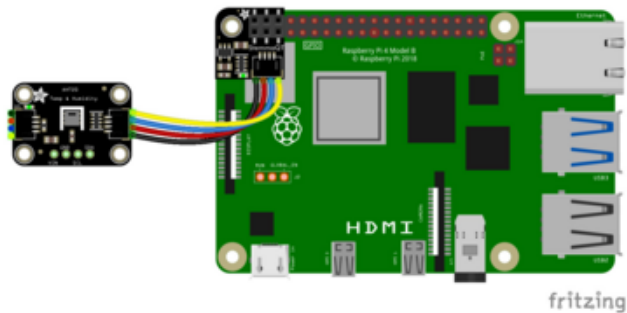
The AHT20 is a nice but inexpensive temperature and humidity sensor from the same folks that brought us the DHT22. You can take...

<https://www.adafruit.com/product/4566>

Python Computer Wiring

Since there are 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's the Raspberry Pi wired with the Pi Stemma QT Breakout and an AHT20 breakout:



Plug the Pi Stemma QT Breakout header into the Raspberry Pi GPIO. Then, plug the AHT20 sensor into the breakout with a STEMMA QT cable.

No soldering required, making it quick and easy.

Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing `code.py` with whatever you named the file:

```
python3 code.py
```

Example Code

The console output will appear wherever you are running Python.

```
# SPDX-FileCopyrightText: 2017 Limor Fried for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# pylint: disable=broad-except, eval-used, unused-import

"""CircuitPython I2C Device Address Scan"""
import time
import board
import busio

# List of potential I2C busses
ALL_I2C = ("board.I2C()", "board.STEMMA_I2C()", "busio.I2C(board.GP1, board.GP0)")

# Determine which busses are valid
found_i2c = []
for name in ALL_I2C:
    try:
        print("Checking {}".format(name), end="")
        bus = eval(name)
        bus.unlock()
        found_i2c.append((name, bus))
        print("ADDED.")
    except Exception as e:
        print("SKIPPED:", e)

# Scan valid busses
if len(found_i2c):
    print("-" * 40)
```



```

print("I2C SCAN")
print("-" * 40)
while True:
    for bus_info in found_i2c:
        name = bus_info[0]
        bus = bus_info[1]

        while not bus.try_lock():
            pass

        print(
            name,
            "addresses found:",
            [hex(device_address) for device_address in bus.scan()],
        )

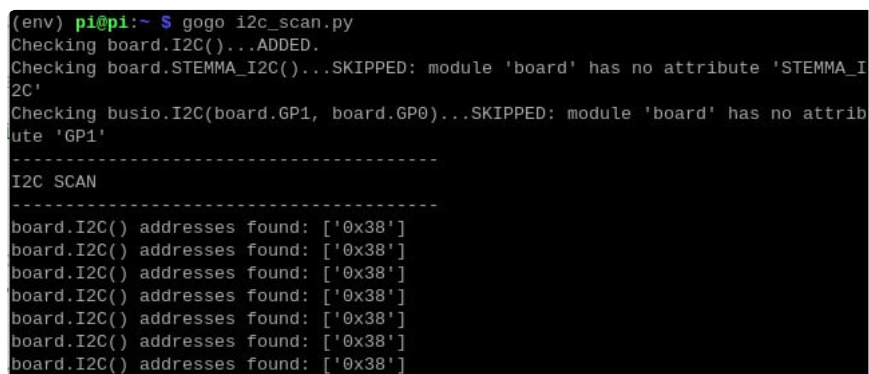
        bus.unlock()

    time.sleep(2)
else:
    print("No valid I2C bus found.")

```

First, the I2C bus is found. Then, in the loop, the connected I2C device addresses are printed to the console. The AHT20 is on address **0x38**. If you have that breakout connected, then you should see that address printed repeatedly.

If you have a different breakout, a different address specific to that board would typically be printed.



```

(env) pi@pi:~$ gogo i2c_scan.py
Checking board.I2C()...ADDED.
Checking board.STEMMA_I2C()...SKIPPED: module 'board' has no attribute 'STEMMA_I2C'
Checking busio.I2C(board.GP1, board.GP0)...SKIPPED: module 'board' has no attribute 'GP1'
-----
I2C SCAN
-----
board.I2C() addresses found: ['0x38']
board.I2C() addresses found: ['0x38']
board.I2C() addresses found: ['0x38']
board.I2C() addresses found: ['0x38']
board.I2C() addresses found: ['0x38']
board.I2C() addresses found: ['0x38']
board.I2C() addresses found: ['0x38']

```

Downloads

Files

- [EagleCAD PCB files on GitHub \(https://adafru.it/1anI\)](https://adafru.it/1anI)
- [3D models on GitHub \(https://adafru.it/1anJ\)](https://adafru.it/1anJ)
- [Fritzing object in the Adafruit Fritzing Library \(https://adafru.it/1anK\)](https://adafru.it/1anK)

Schematic and Fab Print

