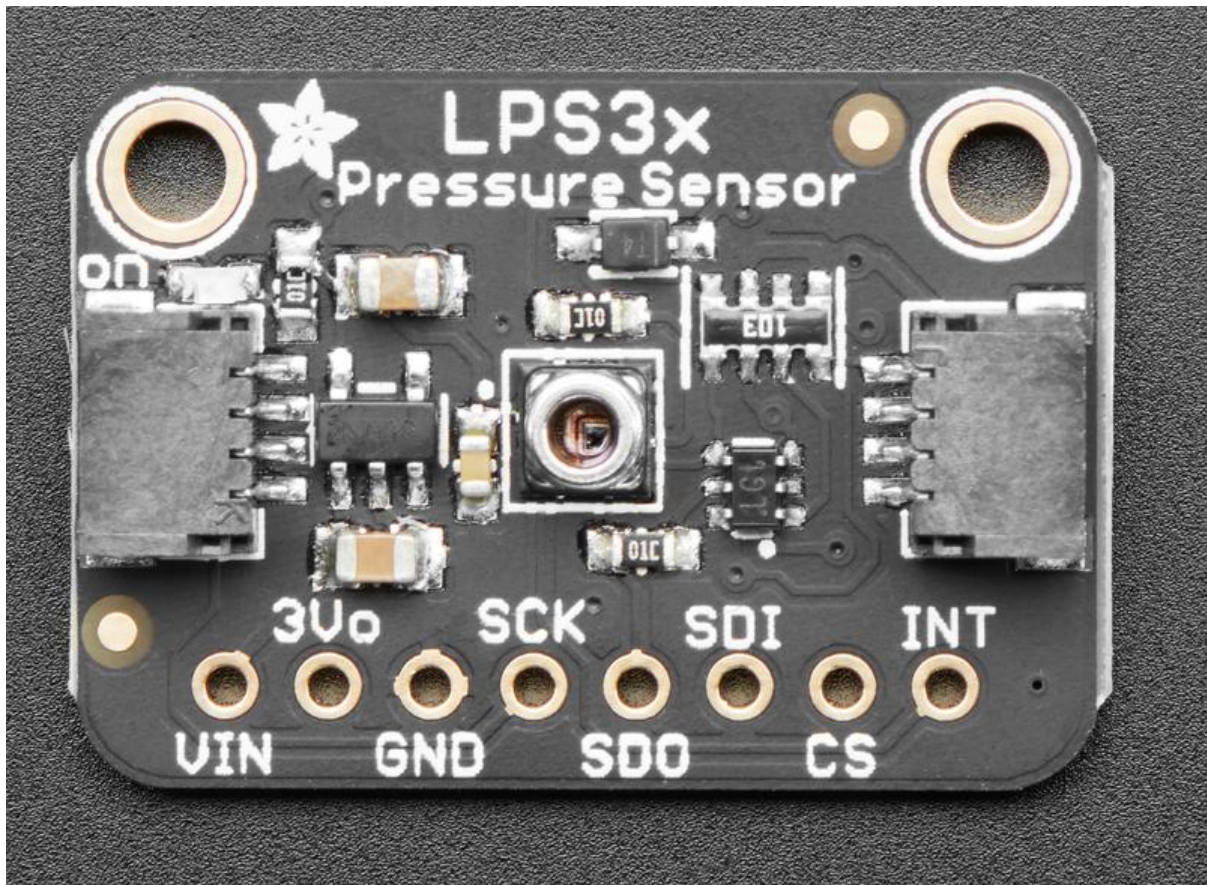




Adafruit LPS33/LPS35 Water Resistant Pressure Sensor

Created by Bryan Siepert



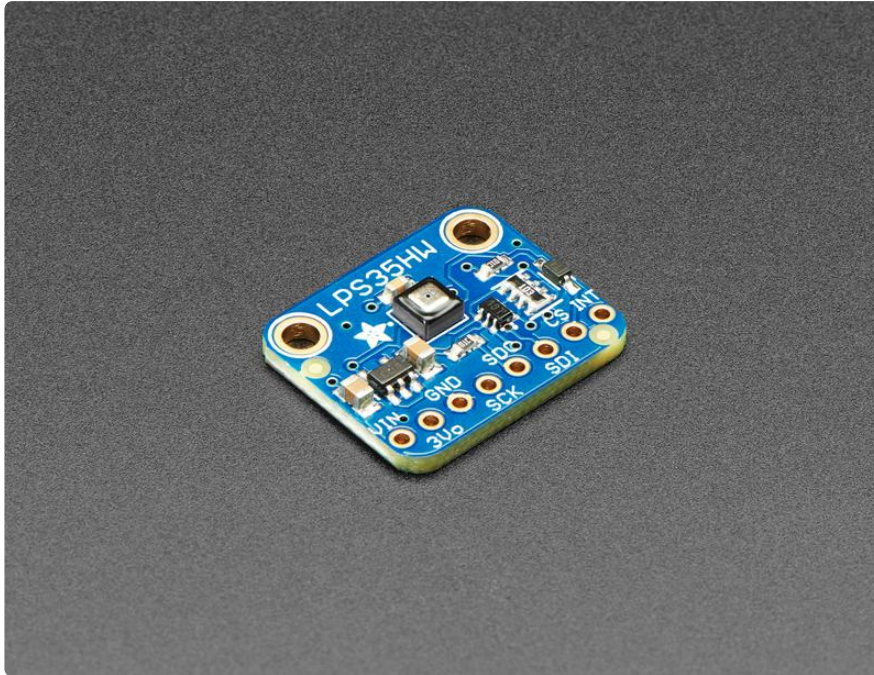
<https://learn.adafruit.com/lps35hw-water-resistant-pressure-sensor>

Last updated on 2023-09-07 09:57:41 AM EDT

Table of Contents

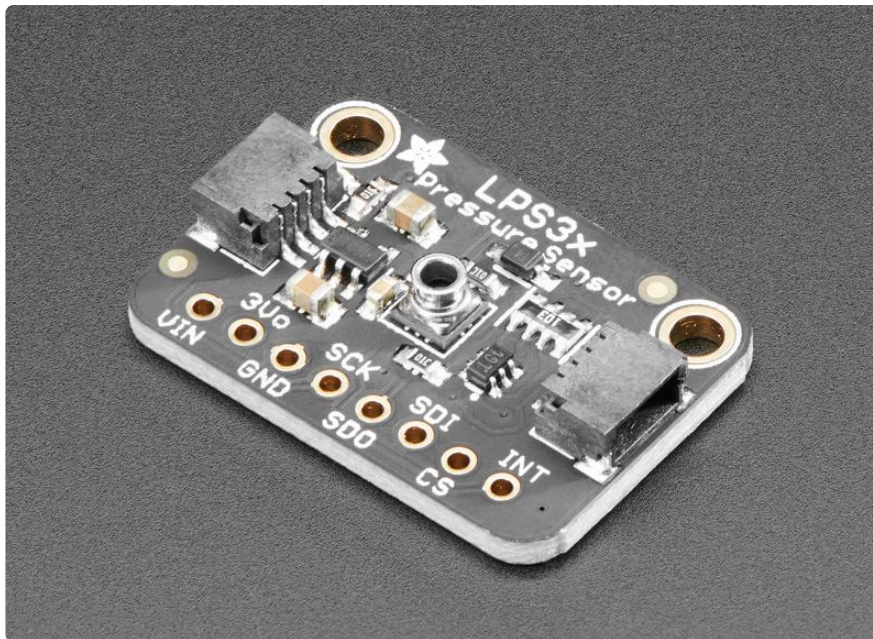
Overview	3
<ul style="list-style-type: none">• Capable sensors• A Family Affair	
Pinouts	8
<ul style="list-style-type: none">• Power Pins• I2C Logic pins:• SPI Logic pins:• Other pins	
Assembly	10
<ul style="list-style-type: none">• Prepare the header strip:• Add the breakout board:• And Solder!	
Arduino	13
<ul style="list-style-type: none">• I2C Wiring• SPI Wiring• Library Installation• Load Example• Example Code	
Arduino Docs	17
Python and CircuitPython	18
<ul style="list-style-type: none">• CircuitPython Microcontroller Wiring• Python Computer Wiring• CircuitPython Installation of LPS35HW Library• Python Installation of LPS35HW Library• CircuitPython & Python Usage• Full Example Code	
Python Docs	22
WipperSnapper	23
<ul style="list-style-type: none">• What is WipperSnapper• Wiring• Usage	
Downloads	30
<ul style="list-style-type: none">• LPS33HW Files• LPS33HW Schematic• LPS33HW Fab Print• LPS35HW Schematic• LPS35HW Fab Print	

Overview



Sometimes you need to sense pressure when it's wet. And sometimes you need to know the relative changes in pressure as well as the absolute pressure. For the times you need to do both (or either), the LPS35HW is the pressure sensor for you. Combining protection from water intrusion with support for high precision relative and absolute measurements, this sensor will do what you need. With drivers for CircuitPython, Arduino, and Raspberry Pi, and support for I2C or SPI (Arduino only SPI support, for now) you'll be measuring pressure in moist situations in no time.

The sensor itself is advertised as Water Resistant but the breakout board for testing out this sensor is not! If you want to use it in wet environments you'll need to pot the rest of the board in a waterproof epoxy!

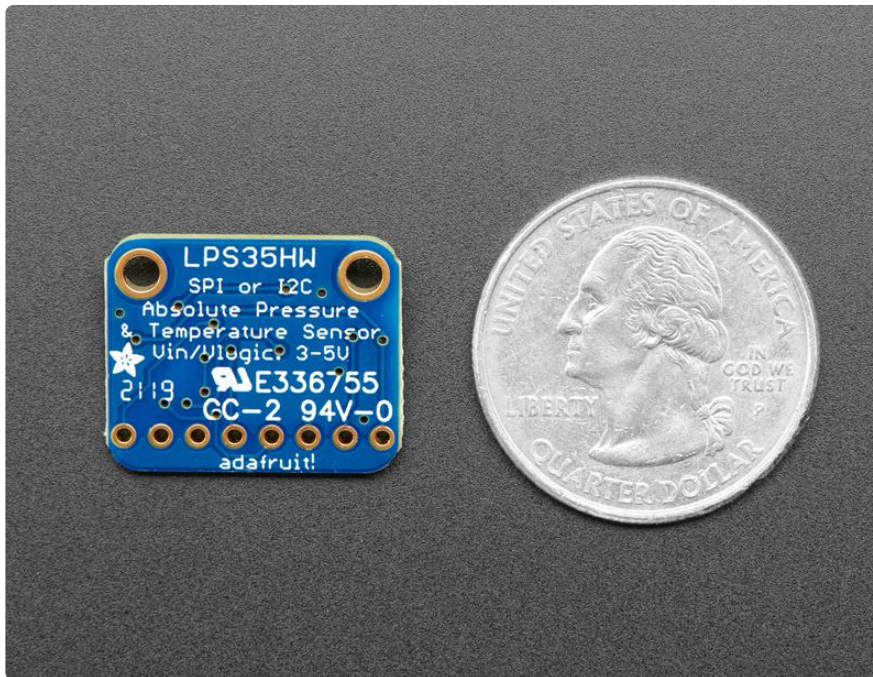
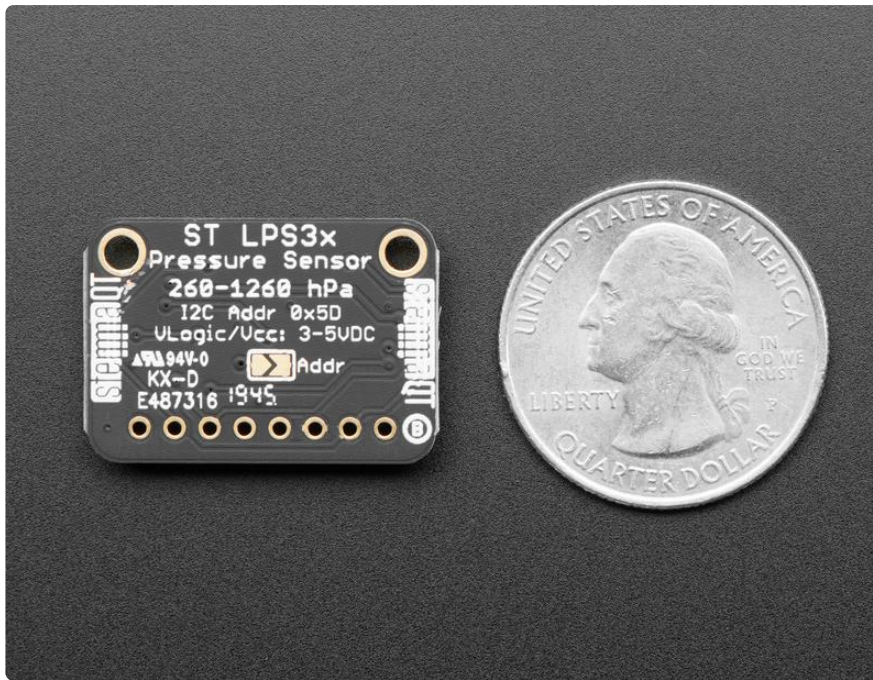


The LPS35HW has a fraternal twin, the LPS33HW. In nearly all respects the LPS33HW is the same as the LPS35HW however the LPS33HW has a bonus feature: a nozzle that can be used to seal it against an enclosure or perhaps a properly fitted tube of some sort. You'll want to make sure to use an appropriately sized o-ring to ensure a good seal. Like the LPS35HW, the LPS33HW chip inside is encased in a protective gel, however in the case of the LPS33HW, the gel is mentioned as being

...designed for and proven to protect electronic components from long-term exposure to harsh environments such as water mixed with chlorine, bromine, commercial washing detergent and fuels, solvents and chemicals.

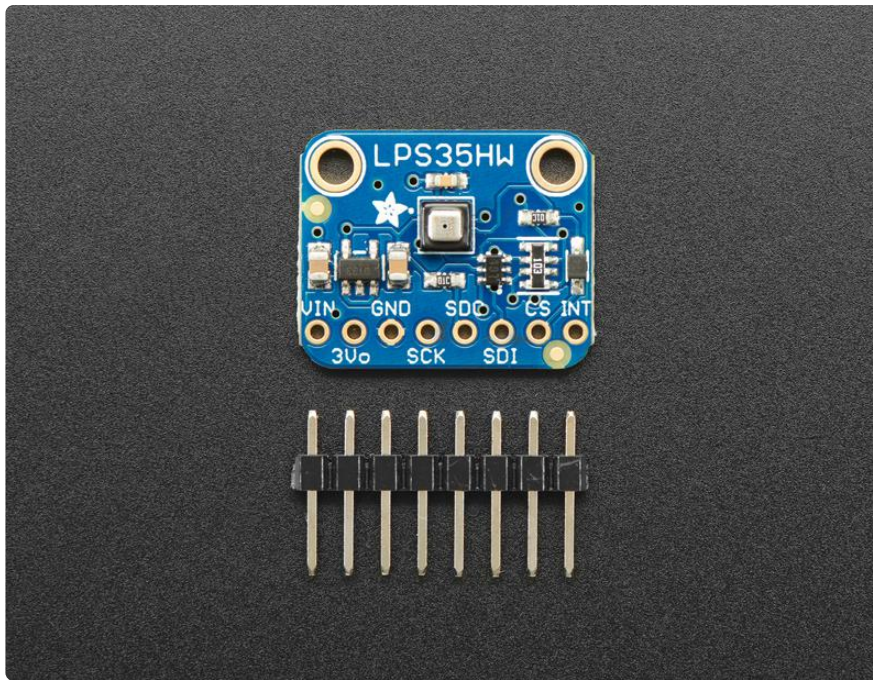
Wow! That's a resilient sensor!

The LPS33HW like its sibling is advertised as being protected from all kinds of horrible things however THE BREAKOUT BOARD IS NOT. For use in icky environments, you'll want to protect the rest of the board with appropriate measures. Since the LPS33HW has a sealable port, you may be able to use the information in the application note on the downloads page to make a suitable enclosure



Capable sensors

The ST LPS35HW is a water resistant barometric pressure and temperature sensor that is also safe to use in wet environments. The sensing element is nestled safely in a ceramic package and is encased in a waterproof gel that prevents water that gets into the sensor from interfering with readings. It does not carry any ratings for resistance to moisture so you probably don't want to take it to the bottom of the Mariana Trench, but it should work well for normal wet situations like weather stations or high humidity.



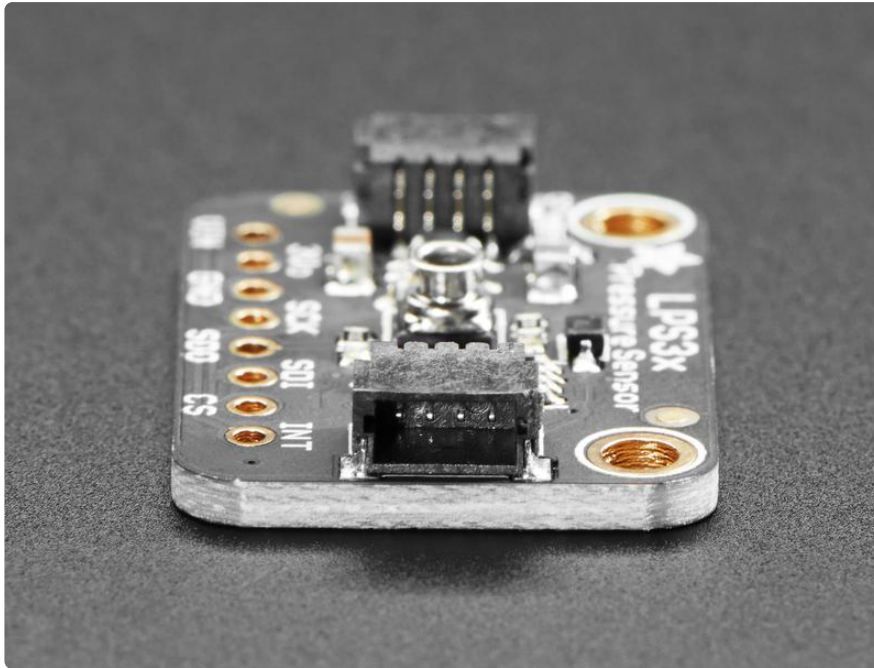
Similarly the LPS33HW is ensconced in a protective gel that protects it from harsh environments. Additionally it is called out by the manufacturer (ST) in the datasheet as being suitable for applications such as weather station equipment or a moist, mouth-adjacent activity that we don't encourage. Let's just say it seems well suited for things like sip-and-puff switches.

Along with not being afraid of getting wet, the LPS35HW and LPS33HW have 24bit pressure data and 16 bit temperature data, allowing it to deliver pressure readings with +/- 0.1% hPa accuracy. It can measure from 260 to 1260 hPa and is able to withstand pressure up to 20 times their measurement range.

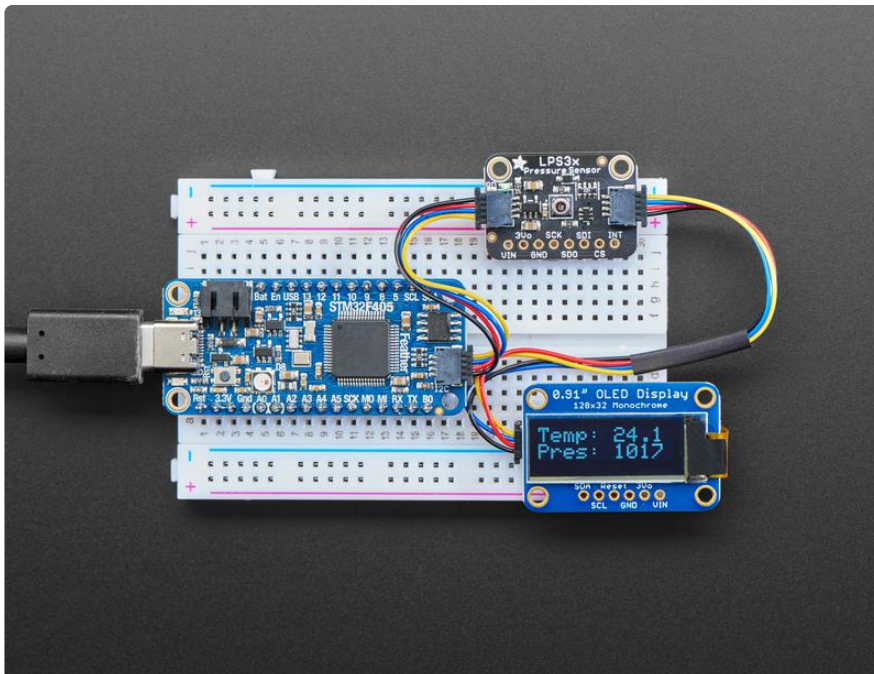
To help you take measurements to your requirements, the sensors also offer an adjustable data rate, as well as a low pass filter to remove noise from the signal. Finally, the onboard temperature compensation makes sure that your readings are always good and won't vary as the temperature changes.

We placed these sensors on a breakout board with a 3.3V regulator and level shifting circuitry so it can be used by 3V or 5V power/logic devices. A small piece of header is also included, so you can solder it in for use with a breadboard.

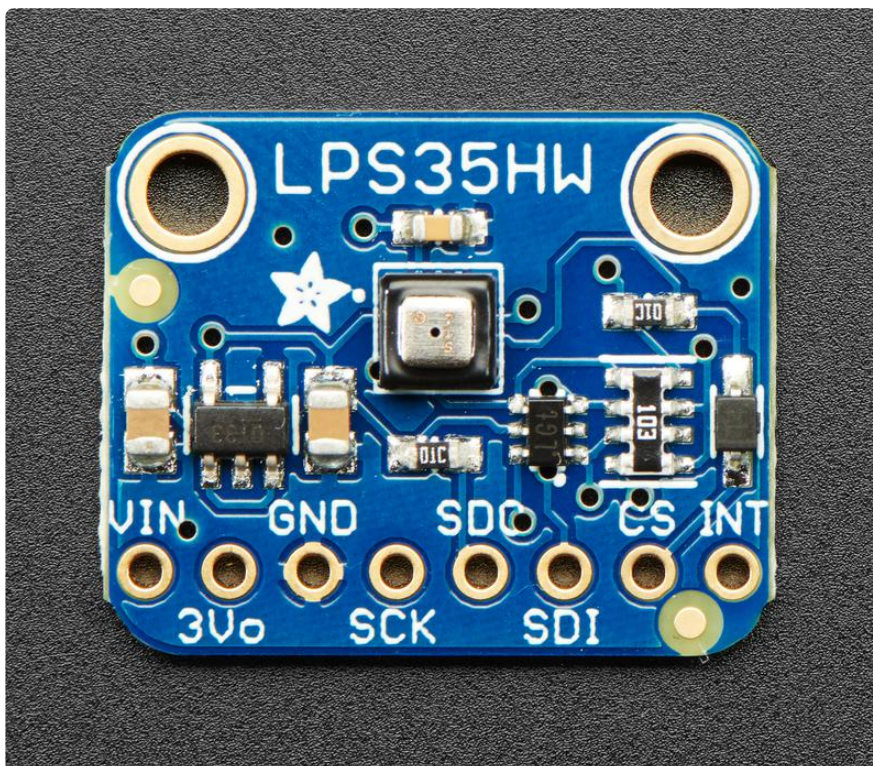
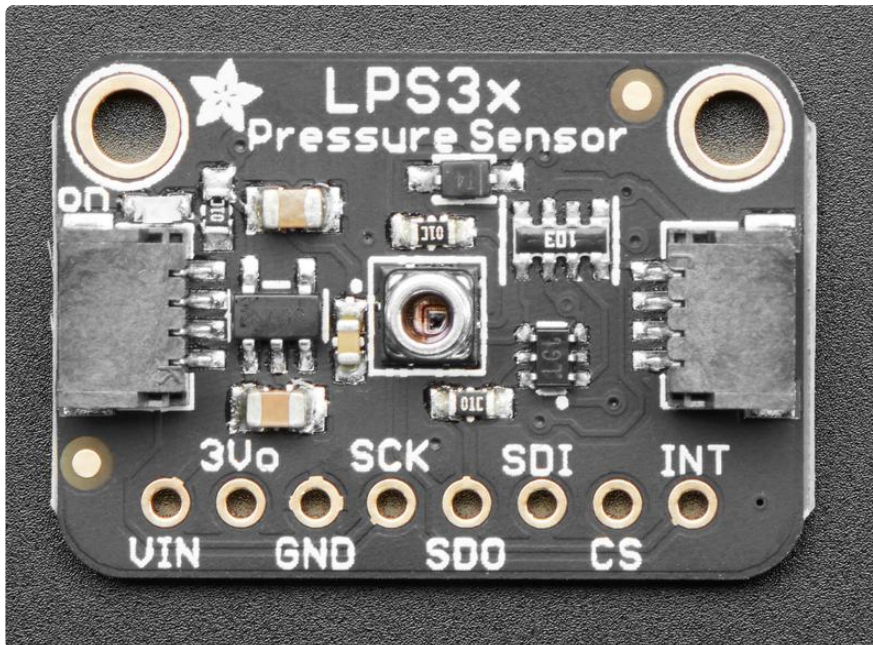
A Family Affair



The LPS33HW also joins our growing family of boards with [SparkFun QUIIC](#) () compatible [STEMMA QT](#) () connectors, allowing you to combine it with all sorts of [other sensors](#) (), and even [displays](#) (), all without needing to solder! Just plug and go/blow!



Pinouts



Power Pins

- Vin - this is the power pin. Since the sensor chip uses 3.3 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V, for a feather use 3.3V

- 3Vo - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like
- GND - common ground for power and logic

I2C Logic pins:

- SCK - this is the I2C clock pin, connect to your microcontrollers I2C clock line.
- SDI - this is the I2C data pin, connect to your microcontrollers I2C data line.

Leave the SDO and CS pins disconnected

SPI Logic pins:

All pins going into the breakout have level shifting circuitry to make them 3-5V logic level safe. Use whatever logic level is on Vin!

- SCK - This is also the SPI Clock pin, its an input to the chip
- SDO - this is the Serial Data Out / Microcontroller In Sensor Out pin, for data sent from the LPS35HW to your processor
- SDI - this is also the Serial Data In / Microcontroller Out Sensor In pin, for data sent from your processor to the LPS35HW
- CS - this is the Chip Select pin, drop it low to start an SPI transaction. Its an input to the chip

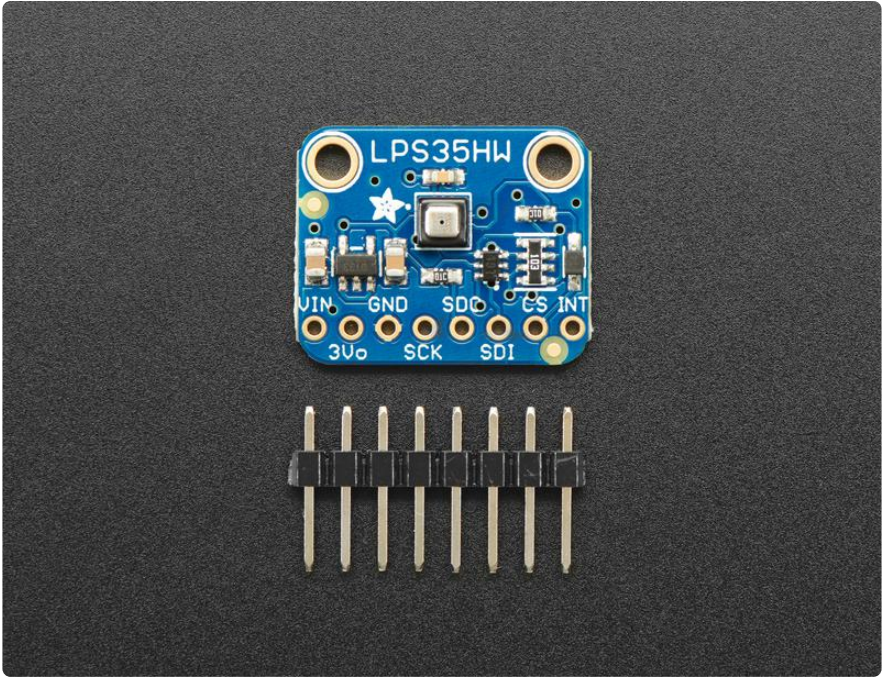
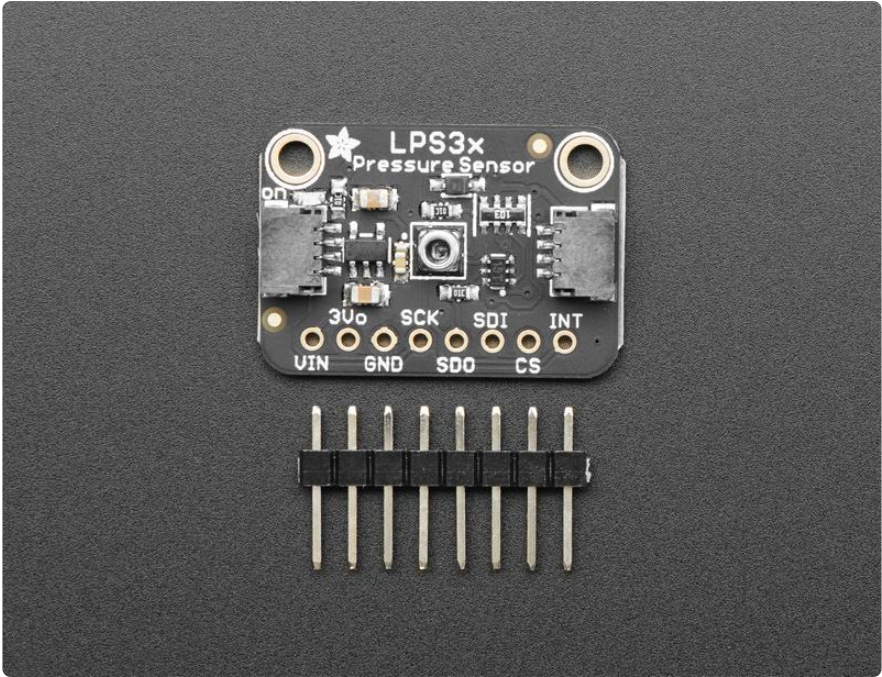
If you want to connect multiple LPS35HW's to one microcontroller, have them share the SDI, SDO and SCK pins. Then assign each one a unique CS pin.

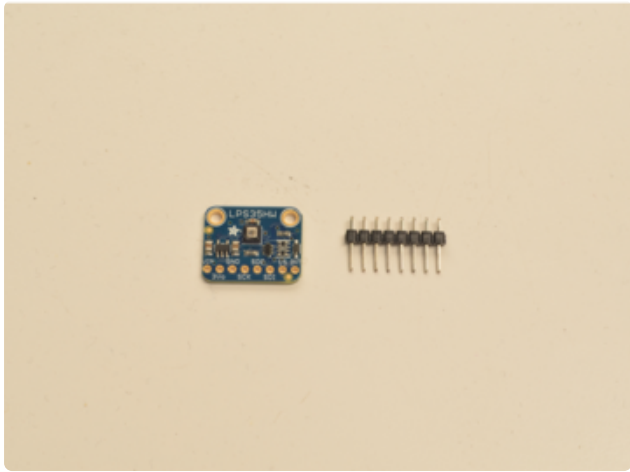
Other pins

- INT is the interrupt output pin. You can configure the interrupt to trigger for various 'reasons' such as going over or under a configured pressure threshold. Voltage level is the same as Vcc.

The low pressure threshold interrupt only works when the LPS33W/LPS35HW are operating in relative mode.

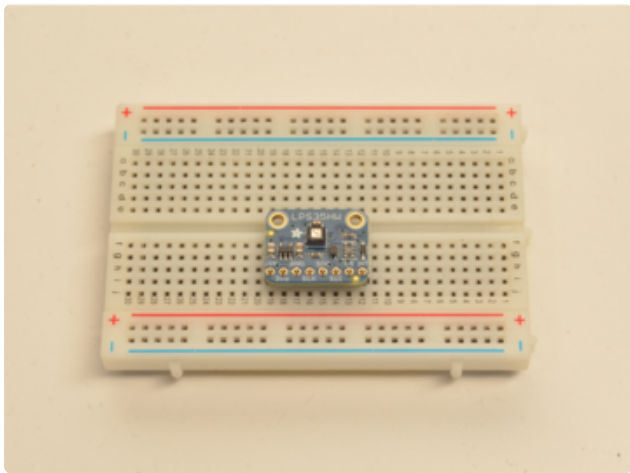
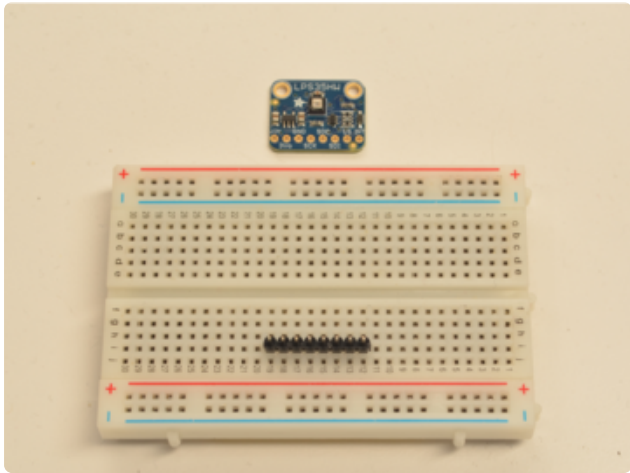
Assembly





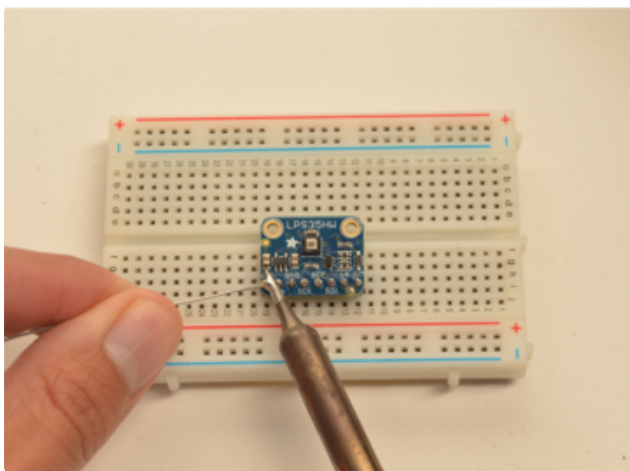
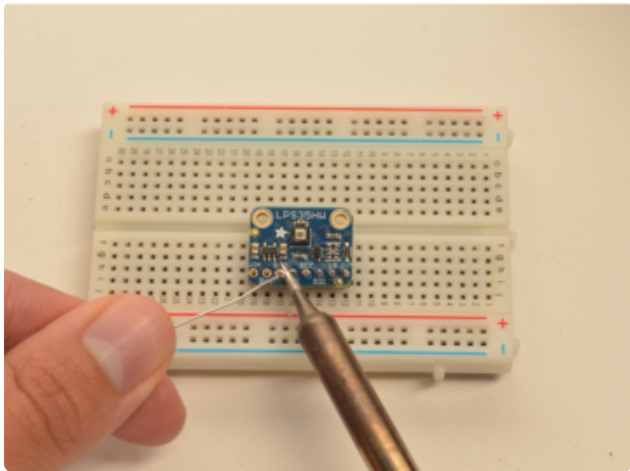
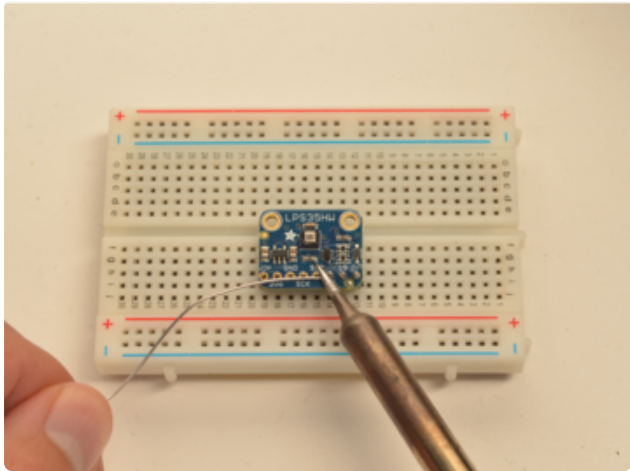
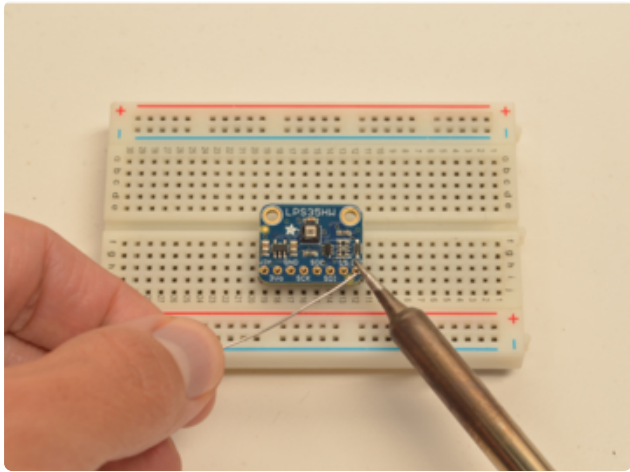
Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - long pins down



Add the breakout board:

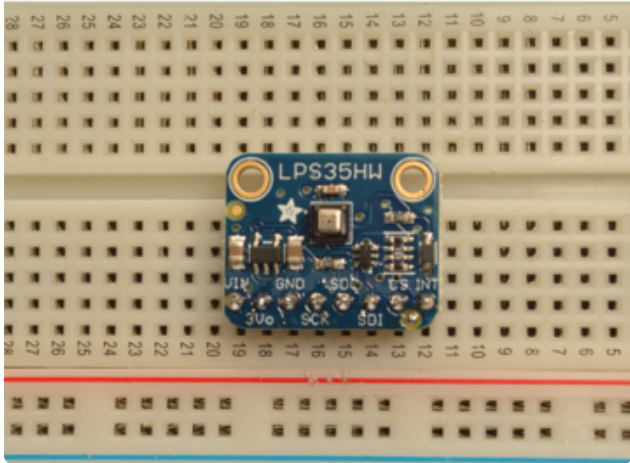
Place the breakout board over the pins so that the short pins poke through the breakout pads



And Solder!

Be sure to solder all 8 pins for reliable electrical contact.

(For tips on soldering, be sure to check out our [Guide to Excellent Soldering \(\)](#)).



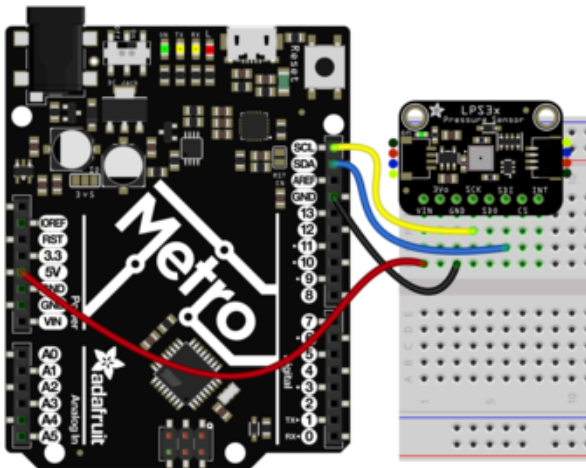
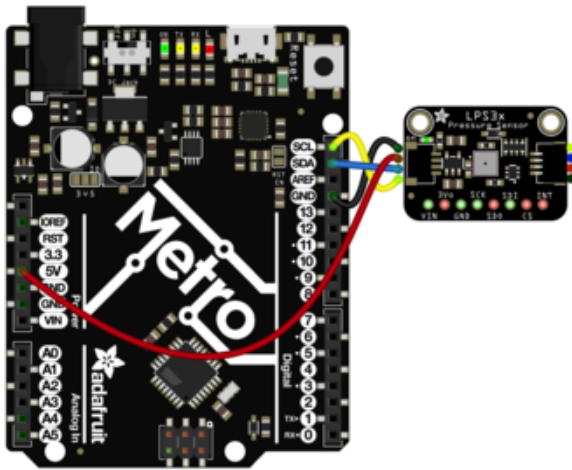
You're done! Check your solder joints visually and continue onto the next steps.

Arduino

I2C Wiring

Use this wiring if you want to connect via I2C interface

By default, the i2c address is 0x5d. If you add a jumper from SDO to GND, the address will change to 0x5c.

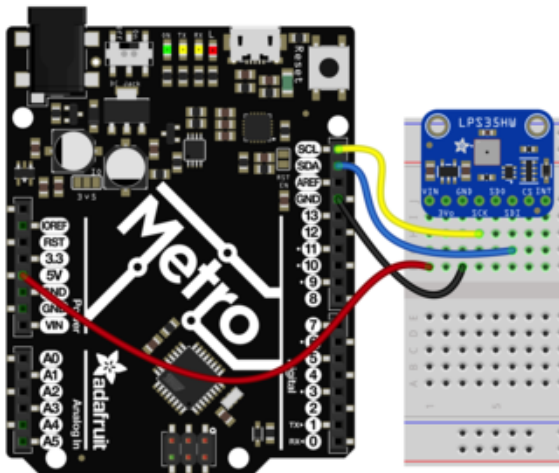


Connect board VCC (red wire) to Arduino 5V if you are running a 5V board Arduino (Uno, etc.). If your board is 3V, connect to that instead.

Connect board GND (black wire) to Arduino GND

Connect board SCL (yellow wire) to Arduino SCL

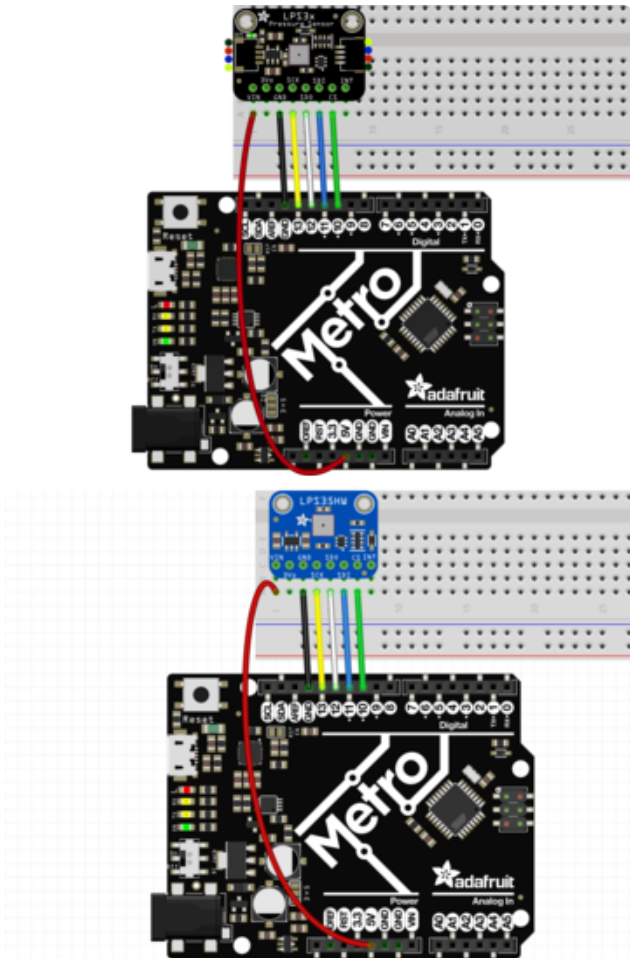
Connect board SDA (blue wire) to Arduino SDA



The final results should resemble the illustration above, showing an Adafruit Metro development board.

SPI Wiring

Since this is a SPI-capable sensor, we can use hardware or 'software' SPI. To make wiring identical on all microcontrollers, we'll begin with 'software' SPI. The following pins should be used:

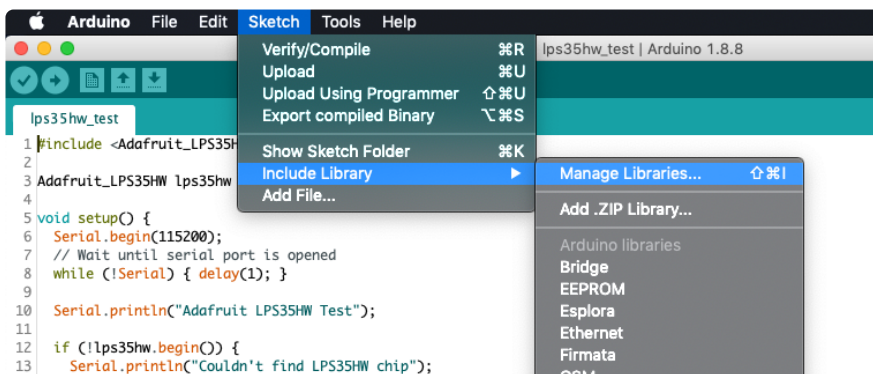


Connect Vin to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of
 Connect GND to common power/data ground
 Connect the SCK pin to Digital #13 but any pin can be used later
 Connect the SDO pin to Digital #12 but any pin can be used later
 Connect the SDI pin to Digital #11 but any pin can be used later
 Connect the CS pin Digital #10 but any pin can be used later

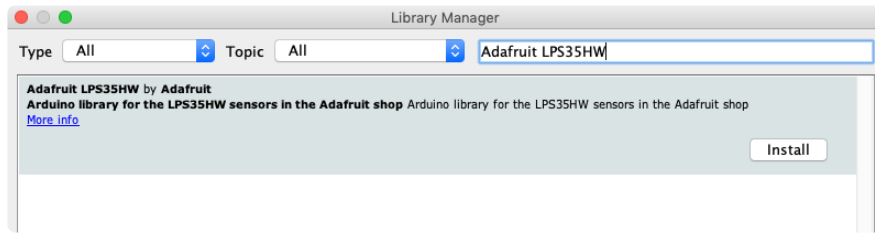
Later on, once we get it working, we can adjust the library to use hardware SPI if you desire, or change the pins to others.

Library Installation

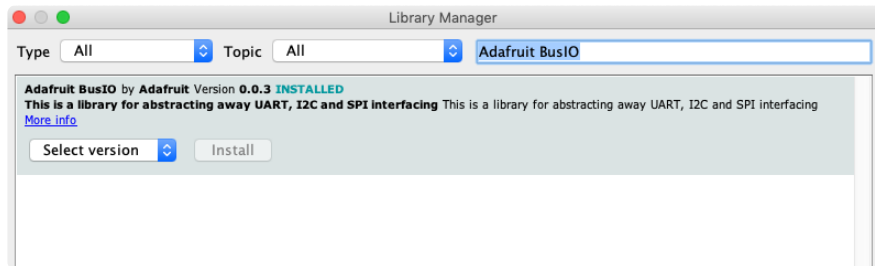
You can install the Adafruit LPS35HW Library for Arduino using the Library Manager in the Arduino IDE. This will work for both the LPS35HW and LPS33HW:



Click the Manage Libraries ... menu item, search for Adafruit LPS35HW, and select the Adafruit LPS35HW library:



Then follow the same process for the Adafruit BusIO library.



Load Example

Open up File -> Examples -> Adafruit LPS35HW -> lps35hw_test and upload to your Arduino wired up to the sensor.

Depending on whether you are using I2C or SPI, change the pin names and comment or uncomment the following lines.

```
if (!lps35hw.begin_I2C()) {  
  //if (!lps35hw.begin_SPI(LPS_CS)) {  
  //if (!lps35hw.begin_SPI(LPS_CS, LPS_SCK, LPS_MISO, LPS_MOSI)) {
```

Once you upload the code, you will see the temperature and pressure being printed when you open the Serial Monitor (Tools->Serial Monitor) at 115200 baud, similar to this:



Temperature is calculated in degrees C, you can convert this to F by using the classic $F = C * 9/5 + 32$ equation.

Pressure is returned in the SI units of Pascals. 100 Pascals = 1 hPa = 1 millibar. Often times barometric pressure is reported in millibar or inches-mercury. For future

reference 1 pascal = 0.000295333727 inches of mercury, or 1 inch Hg = 3386.39 Pascal. So if you take the pascal value of say 100734 and divide by 3386.39 you'll get 29.72 inches-Hg.

Example Code

The following example code is part of the standard library, and illustrates how you can retrieve sensor data from the LPS35HW or LPS33HW for pressure and temperature:

```
#include <Adafruit_LPS35HW.h>

Adafruit_LPS35HW lps35hw = Adafruit_LPS35HW();

// For SPI mode, we need a CS pin
#define LPS_CS 10
// For software-SPI mode we need SCK/MOSI/MISO pins
#define LPS_SCK 13
#define LPS_MISO 12
#define LPS_MOSI 11

void setup() {
  Serial.begin(115200);
  // Wait until serial port is opened
  while (!Serial) { delay(1); }

  Serial.println("Adafruit LPS35HW Test");

  if (!lps35hw.begin_I2C()) {
    //if (!lps35hw.begin_SPI(LPS_CS)) {
    //if (!lps35hw.begin_SPI(LPS_CS, LPS_SCK, LPS_MISO, LPS_MOSI)) {
    Serial.println("Couldn't find LPS35HW chip");
    while (1);
  }
  Serial.println("Found LPS35HW chip");
}

void loop() {
  Serial.print("Temperature: ");
  Serial.print(lps35hw.readTemperature());
  Serial.println(" C");

  Serial.print("Pressure: ");
  Serial.print(lps35hw.readPressure());
  Serial.println(" hPa");

  Serial.println();
  delay(1000);
}
```

Arduino Docs

[Arduino Docs \(\)](#)

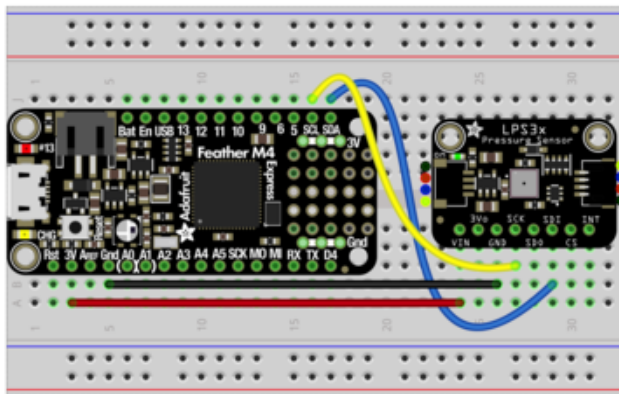
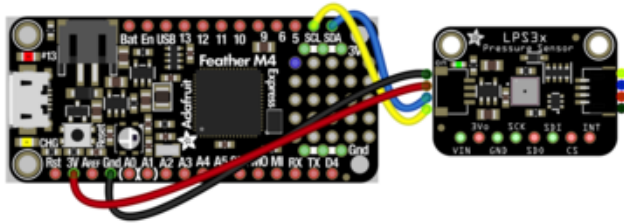
Python and CircuitPython

It's easy to use the LPS33HW or LPS35HW sensors with Python and CircuitPython, and the [Adafruit CircuitPython LPS35HW \(\)](#) module. This module allows you to easily write Python code that reads the pressure and temperature and will work with either sensor.

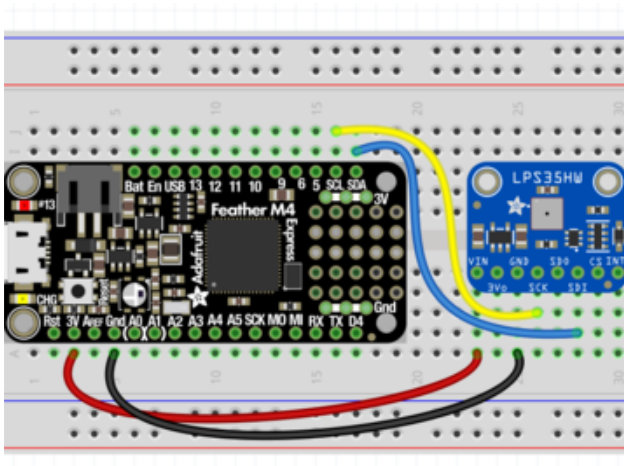
You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python [thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library \(\)](#).

CircuitPython Microcontroller Wiring

First wire up a LPS35HW or LPS33HW to your board for an I2C connection, exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C:



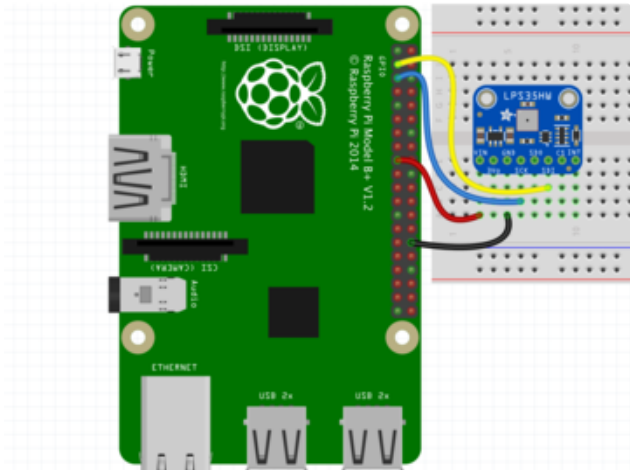
- Board 3V to sensor VIN (red wire)
- Board GND to sensor GND (black wire)
- Board SCL to sensor SCL (yellow wire)
- Board SDA to sensor SDA (blue wire)



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 \(\)](#).

Here's the Raspberry Pi wired with I2C:



Pi 3V3 to sensor VIN
Pi GND to sensor GND
Pi SCL to sensor SCK
Pi SDA to sensor SDI

CircuitPython Installation of LPS35HW Library

Next you'll need to install the [Adafruit CircuitPython LPS35HW \(\)](#) library on your CircuitPython board.

First make sure you are running the [latest version of Adafruit CircuitPython \(\)](#) 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 \(\)](#). Our introduction guide has [a great page on how to install the library bundle \(\)](#) for both express and non-express boards.

Remember for non-express boards like the, you'll need to manually install the necessary libraries from the bundle:

- adafruit_lps35hw.mpy
- adafruit_bus_device
- adafruit_register

You can also download the adafruit_lps35hw.mpy from [its releases page on Github \(\)](#).

Before continuing make sure your board's lib folder or root filesystem has the adafruit_lps35hw.mpy, adafruit_bus_device, and adafruit_register files and folders copied over.

Next [connect to the board's serial REPL \(\)](#) so you are at the CircuitPython >>> prompt.

Python Installation of LPS35HW 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 \(\)!](#)

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

- `sudo pip3 install adafruit-circuitpython-lps35hw`

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

To demonstrate the usage of the sensor we'll initialize it and read the temperature and pressure levels from the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import time
import board
import busio
import adafruit_lps35hw

i2c = busio.I2C(board.SCL, board.SDA)
lps35hw = adafruit_lps35hw.LPS35HW(i2c)
```

```
>>> import time
>>> import board
>>> import busio
>>> import adafruit_lps35hw
>>> i2c = busio.I2C(board.SCL, board.SDA)
>>> lps35hw = adafruit_lps35hw.LPS35HW(i2c)
```

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

- pressure - The barometric pressure in hPa.
- temperature - The temperature in degrees C.

For example to print the pressure and temperature values:

```
print("Pressure: %.2f hPa" % lps35hw.pressure)
print("Temperature: %.2f C"% lps35hw.temperature)
```

```
>>> print("Pressure: %.2f hPa" % lps35hw.pressure)
Pressure: 1008.01 hPa
>>> print("Temperature: %.2f C"% lps35hw.temperature)
Temperature: 18.18 C
```

For more details, check out the [library documentation](#) ().

That's all there is to using the LPS33HW/LPS35HW sensor with CircuitPython!

Full Example Code

```
# SPDX-FileCopyrightText: 2019 Bryan Siepert, written for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense
import time
import board
import adafruit_lps35hw

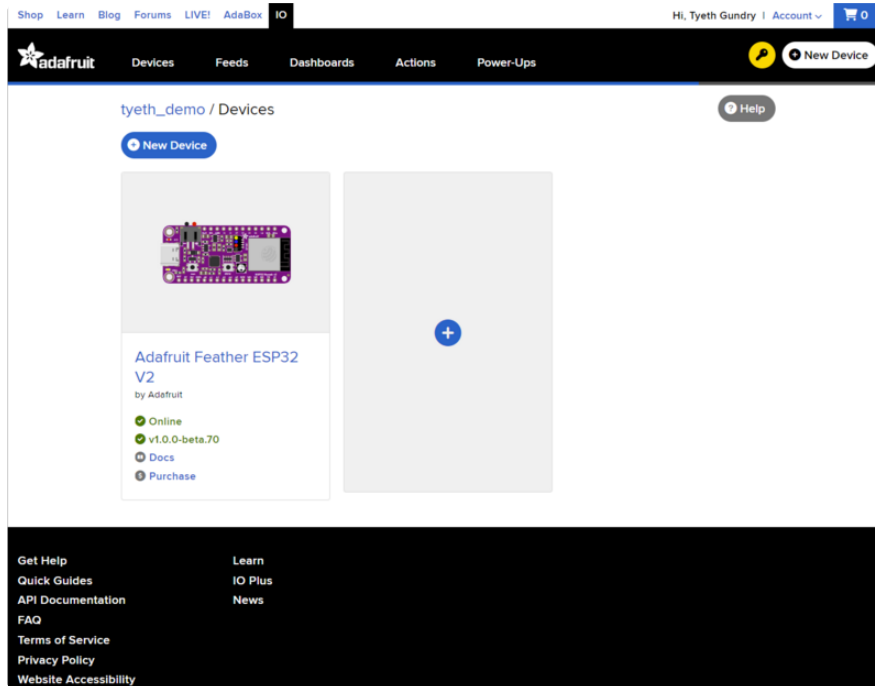
i2c = board.I2C() # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C() # For using the built-in STEMMA QT connector on a
microcontroller
lps = adafruit_lps35hw.LPS35HW(i2c)

while True:
    print("Pressure: %.2f hPa" % lps.pressure)
    print("Temperature: %.2f C" % lps.temperature)
    print("")
    time.sleep(1)
```

Python Docs

[Python Docs](#) ()

WipperSnapper



What is WipperSnapper

WipperSnapper is a firmware designed to turn any WiFi-capable board into an Internet-of-Things device without programming a single line of code. WipperSnapper connects to [Adafruit IO \(\)](#), a web platform designed ([by Adafruit! \(\)](#)) to display, respond, and interact with your project's data.

Simply load the WipperSnapper firmware onto your board, add credentials, and plug it into power. Your board will automatically register itself with your Adafruit IO account.

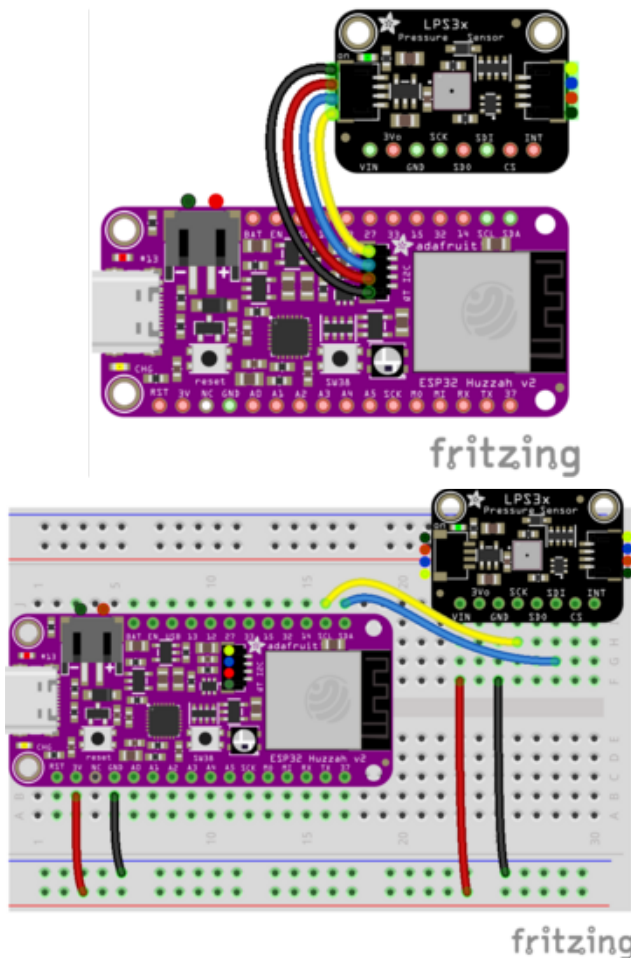
From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

Quickstart: Adafruit IO
WipperSnapper

Wiring

First, wire up an LPS3xHW to your board exactly as follows. Here is an example of the LPS3xHW wired to an [Adafruit ESP32 Feather V2 \(\)](#) using I2C [with a STEMMA QT cable \(no soldering required\) \(\)](#)



Board 3V to sensor VIN (red wire on STEMMA QT)

Board GND to sensor GND (black wire on STEMMA QT)

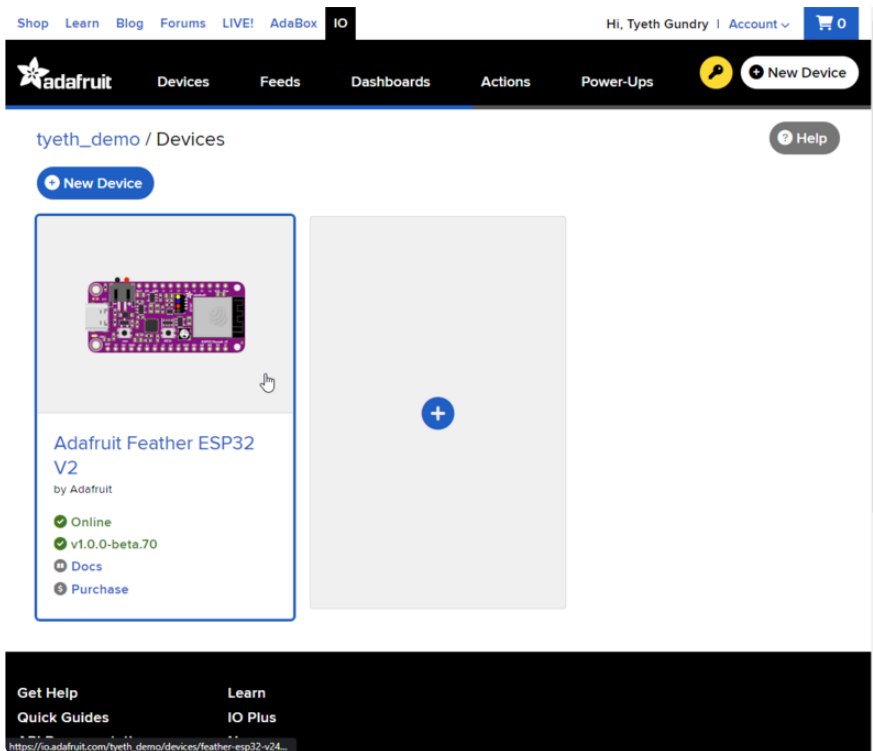
Board SCL to sensor SCK (yellow wire on STEMMA QT)

Board SDA to sensor SDI (blue wire on STEMMA QT)

Usage

Connect your board to Adafruit IO Wippersnapper and [navigate to the WipperSnapper board list \(\)](#).

On this page, select the WipperSnapper board you're using to be brought to the board's interface page.



If you do not see your board listed here - you need [to connect your board to Adafruit IO \(\)](#) first.

Adafruit Feather ESP32 V2

by Adafruit

- ✓ Online
- ✓ v1.0.0-beta.70 ←
- 📖 Docs
- 💰 Purchase

On the device page, quickly check that you're running the latest version of the WipperSnapper firmware.

The device tile on the left indicates the version number of the firmware running on the connected board.

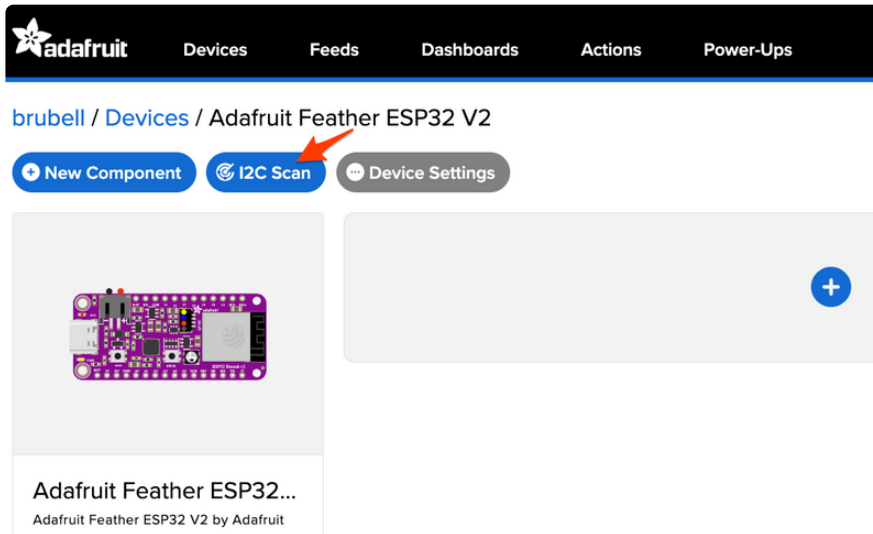
Adafruit Feather ESP32 V2

by Adafruit

- ✓ Online
- ! v1.0.0-beta.68 [Update](#)
- 📖 Docs
- 💰 Purchase

If the firmware version is green with a checkmark - continue with this guide. If the firmware version is red with an exclamation mark "!" - [update to the latest WipperSnapper firmware \(\)](#) on your board before continuing.

Next, make sure the sensor is plugged into your board and click the I2C Scan button.



You should see the LPS3xHW's default I2C address of `0x5d` pop-up in the I2C scan list.

I2C Scan Complete ✕

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00								--	--	--	--	--	--	--	--	--
10	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50	--	--	--	--	--	--	--	--	--	--	--	--	--	5d	--	--
60	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Close Scan Again

I don't see the sensor's I2C address listed!

First, double-check the connection and/or wiring between the sensor and the board.

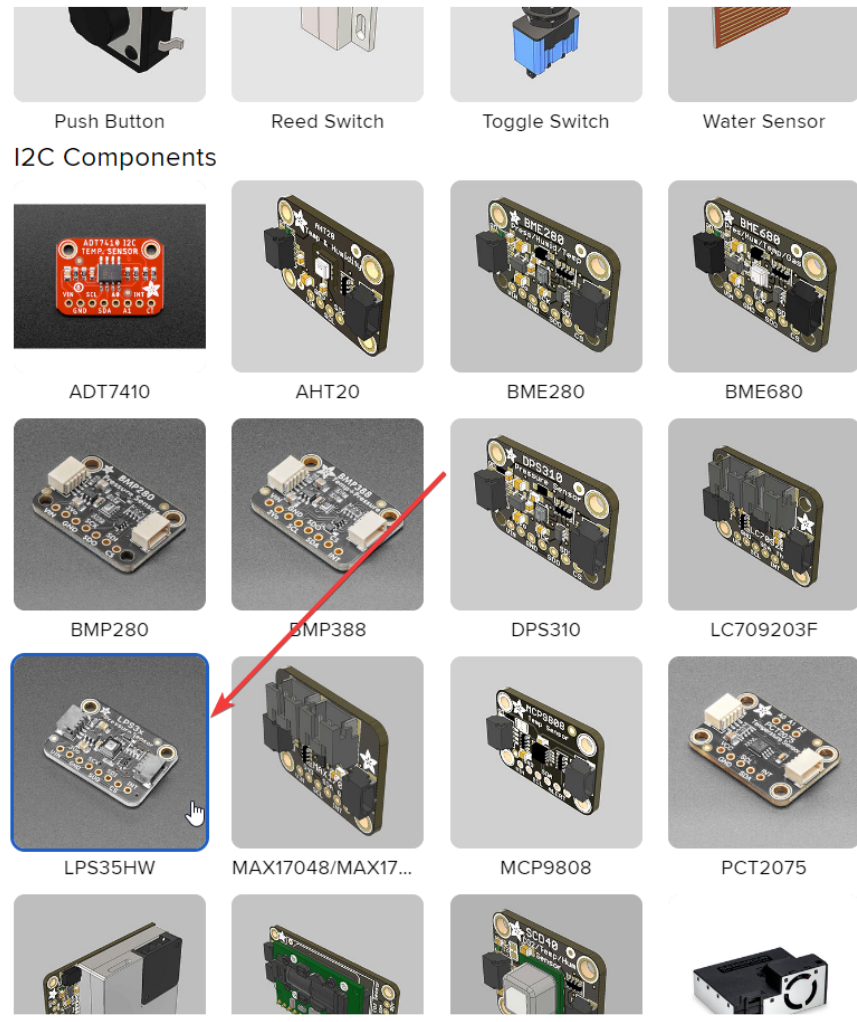
Then, reset the board and let it re-connect to Adafruit IO WipperSnapper.

With the sensor detected in an I2C scan, you're ready to add the sensor to your board.

Click the New Component button or the + button to bring up the component picker.



Select the LPS33HW / LPS35HW from the component picker.



On the component configuration page, the LPS3xHW's sensor address should be listed along with the sensor's settings.

The Send Every option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the LPS3xHW sensor and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

For this example, set the Send Every interval to every 30 seconds.

Create LPS35HW Component ✕

Select I2C Address:

Enable LPS35HW: Temperature Sensor (°C)?
Name:

Send Every:

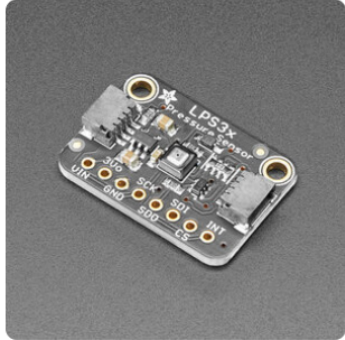
Enable LPS35HW: Temperature Sensor (°F)?
Name:

Send Every:

Enable LPS35HW: Pressure Sensor?
Name:

Send Every:

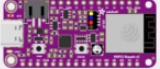
[← Back to Component Type](#) **Create Component**



Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.

tyeth_demo / Devices / Adafruit Feather ESP32 V2 Help

New Component Auto-Config I2C Scan Settings



Adafruit Feather ESP32 V2
by Adafruit

- Online
- v1.0.0-beta.70
- Docs
- Purchase

[Report Bugs](#)

LPS35HW: Pressure Sensor lps35hw.pressure Help

1006.51hPa

[Create Action](#) | [Add to Dashboard](#)

LPS35HW: Temperature Sensor (°C) lps35hw.ambient-temp Help

23.37°C

[Create Action](#) | [Add to Dashboard](#)

LPS35HW: Temperature Sensor (°F) lps35hw.ambient-temp-fahrenheit Help

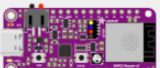
74.10°F

[Create Action](#) | [Add to Dashboard](#)

To view the data that has been logged from the sensor, click on the graph next to the sensor name.

tyeth_demo / Devices / Adafruit Feather ESP32 V2 Help

New Component Auto-Config I2C Scan Settings



Adafruit Feather ESP32 V2
by Adafruit

- Online
- v1.0.0-beta.70
- Docs
- Purchase

[Report Bugs](#)

LPS35HW: Pressure Sensor lps35hw.pressure Help

1006.49hPa

[Create Action](#) | [Add to Dashboard](#)

LPS35HW: Temperature Sensor (°C) lps35hw.ambient-temp Help

23.41°C

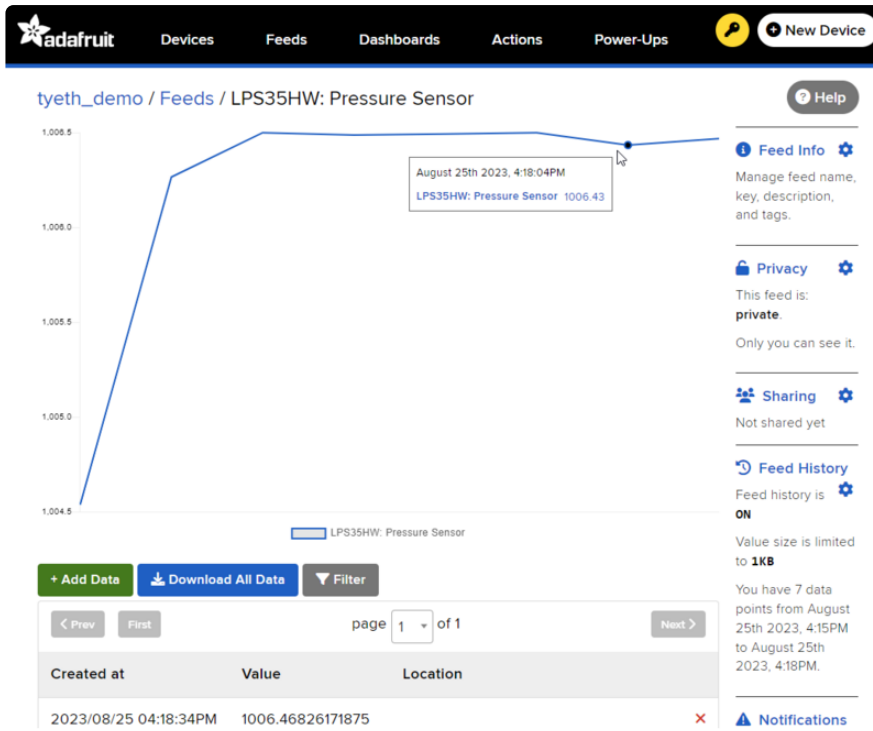
[Create Action](#) | [Add to Dashboard](#)

LPS35HW: Temperature Sensor (°F) lps35hw.ambient-temp-fahrenheit Help

73.96°F

[Create Action](#) | [Add to Dashboard](#)

Here you can see the feed history and edit things about the feed such as the name, privacy, webhooks associated with the feed and more. If you want to learn more about how feeds work, [check out this page](#) ().

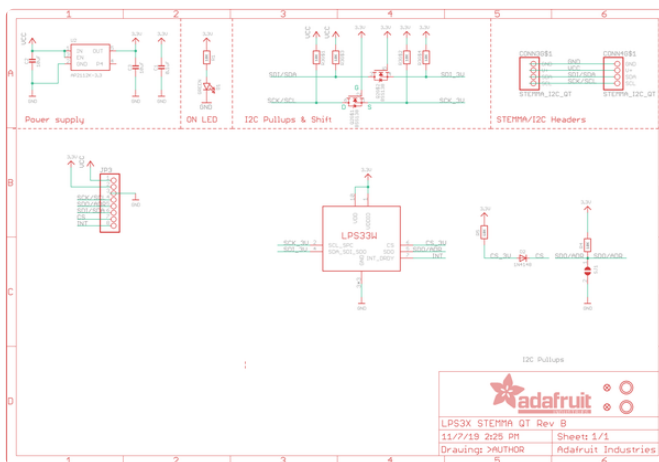


Downloads

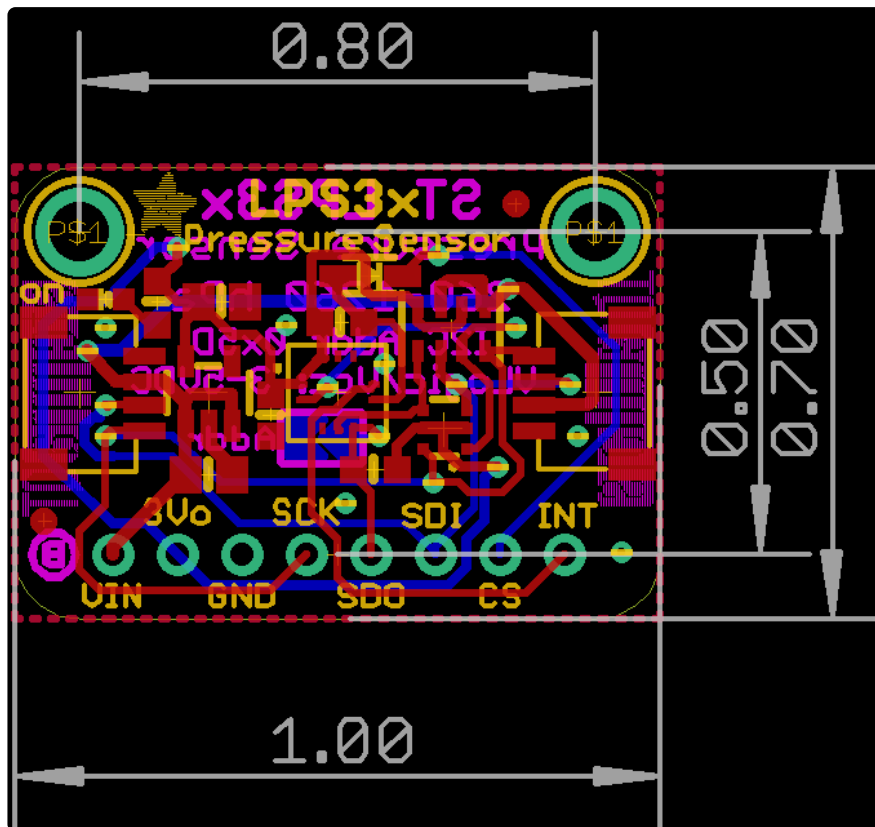
LPS33HW Files

- [LPS35HW Datasheet \(\)](#)
- [LPS33HW Datasheet \(\)](#)
- [LPS33HW System Integration App Note \(\)](#)
- [EagleCAD files on GitHub \(\)](#)
- [LPS33HW Fritzing object from Adafruit Fritzing Library \(\)](#)
- [LPS35HW Fritzing object from Adafruit Fritzing Library \(\)](#)

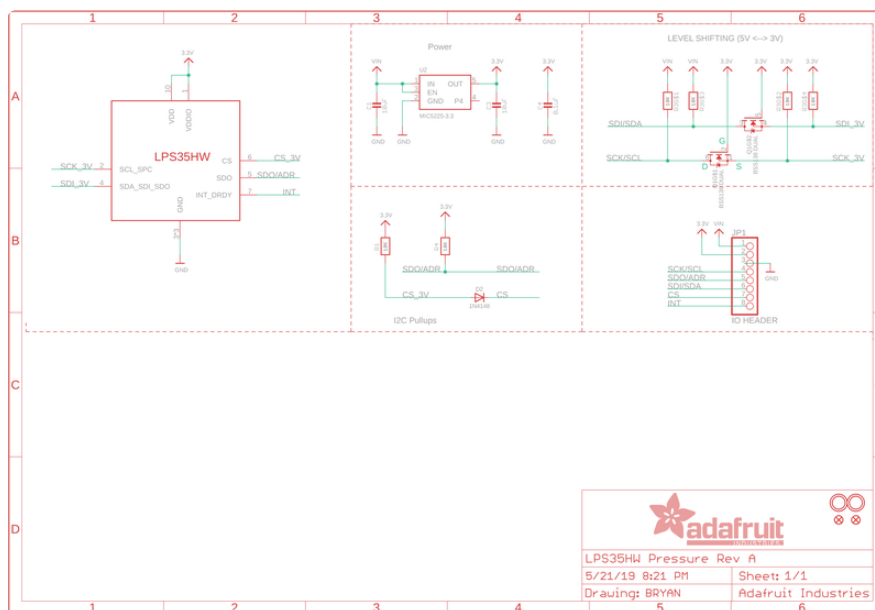
LPS33HW Schematic



LPS33HW Fab Print



LPS35HW Schematic



LPS35HW Fab Print

