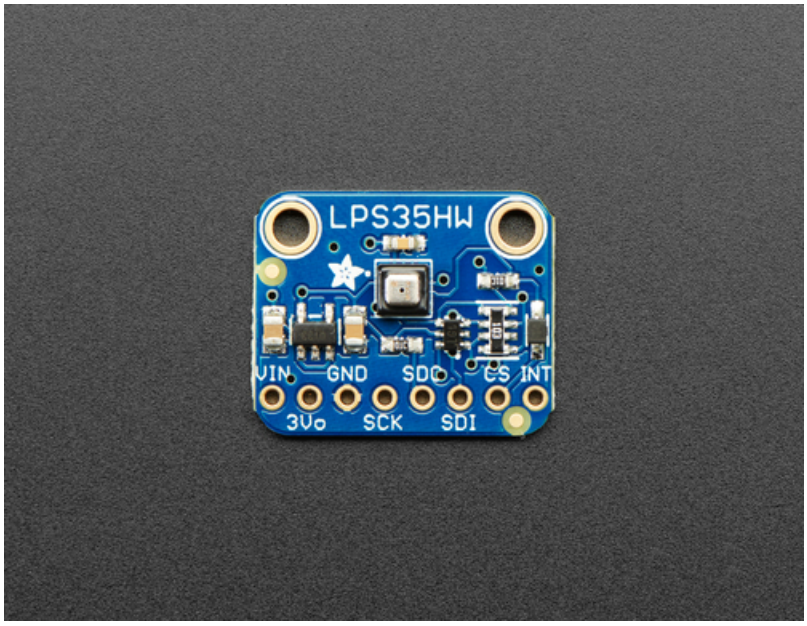
adafruit learning system

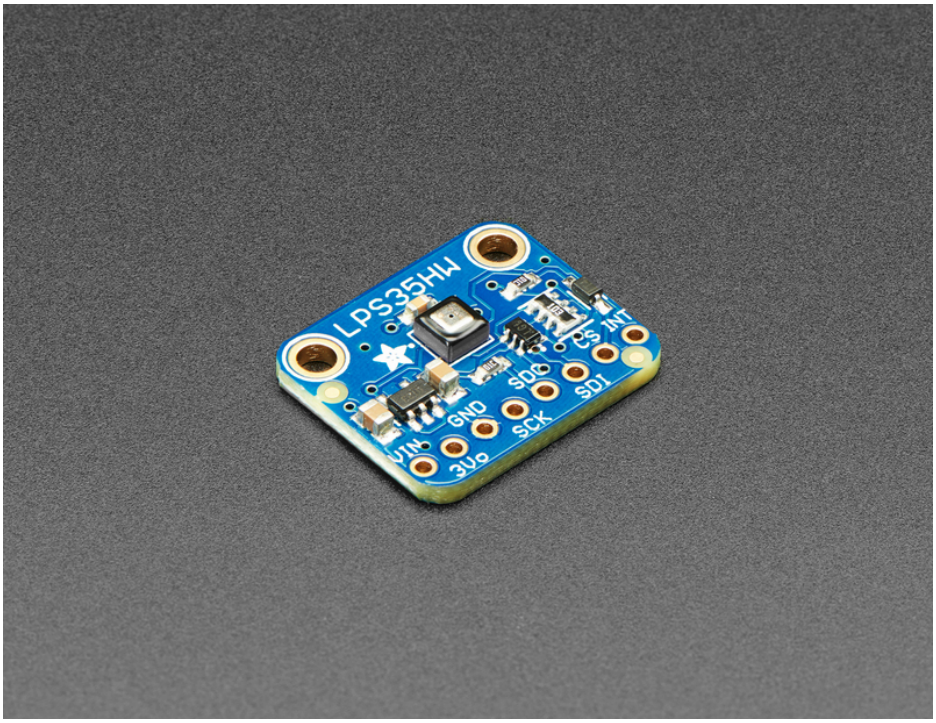
Adafruit LPS35HW Water Resistant Pressure Sensor

Created by Bryan Siepert



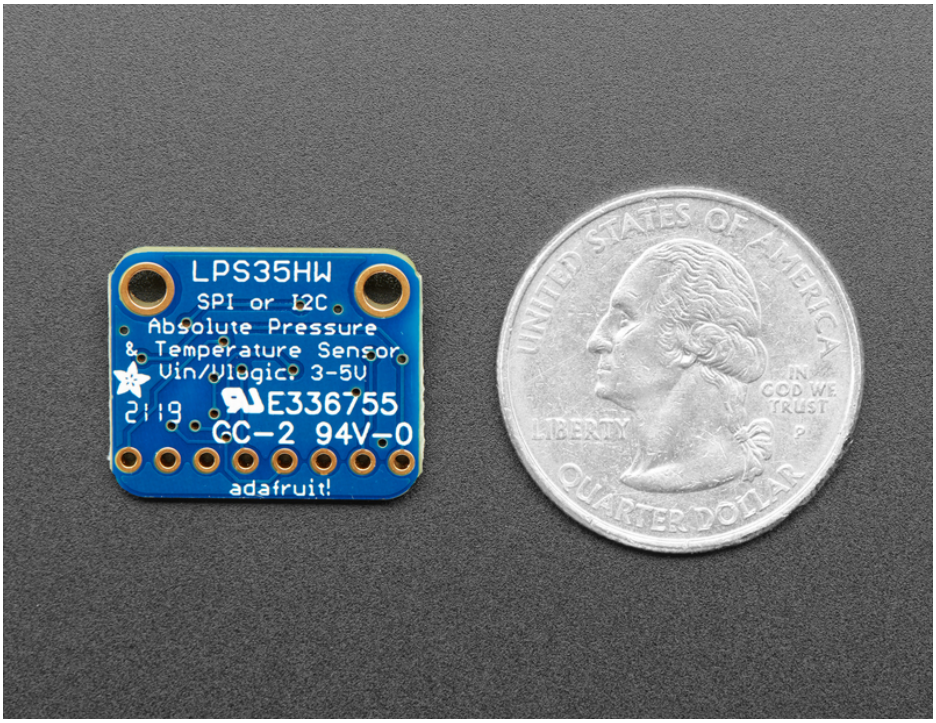
Last updated on 2019-06-21 09:01:57 PM UTC

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 wateproof epoxy!

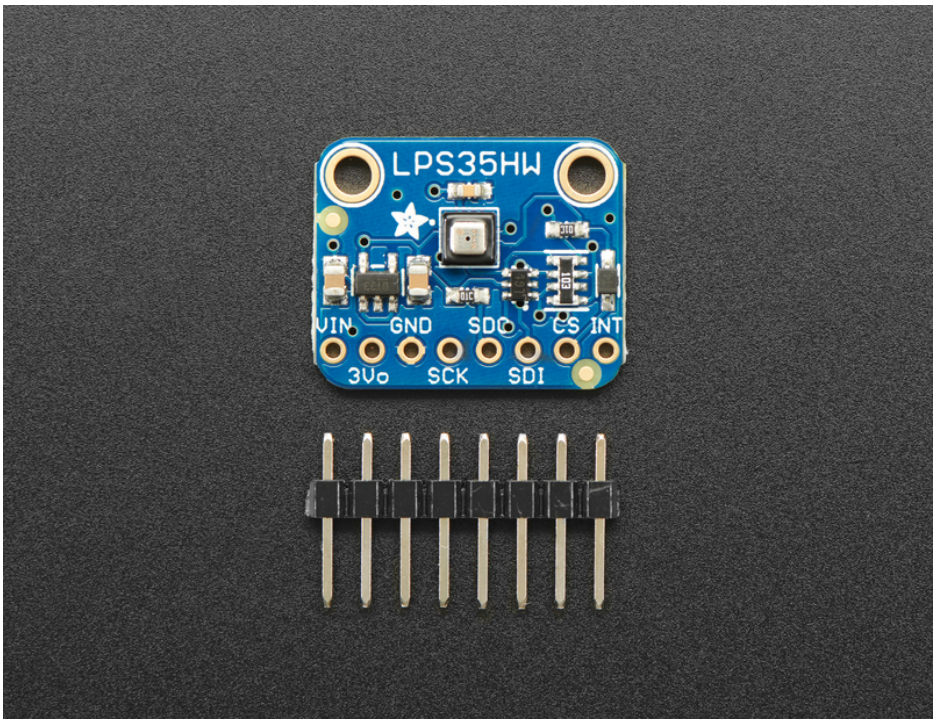


A capable sensor

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.

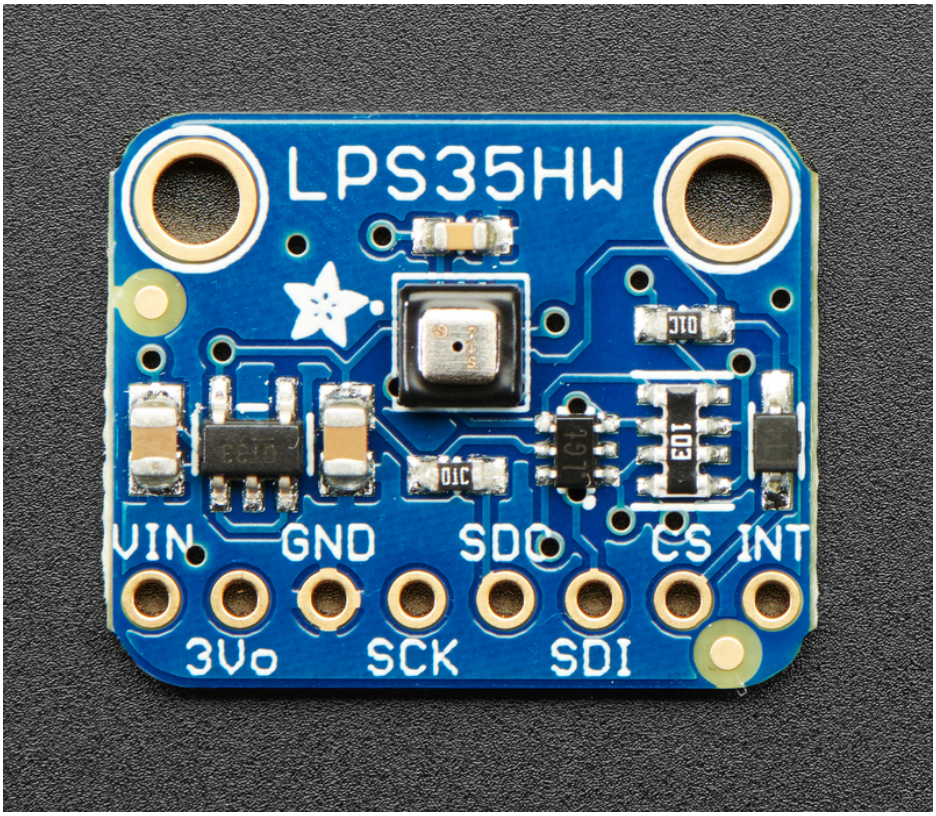
Along with not being afraid of getting wet, the LPS35HW has a 24bit pressure data and 16 bit temperature data, allowing it to deliver pressure readings with +/- 0.1% hPa accuracy. It can measure from 260 to 1260 hPa and is able to withstand pressure up to 20 times its measurement range.

To help you take measurements to your requirements, the LPS35HW also offers 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 this sensor 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.

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 **S**PI **C**lock pin, its an input to the chip
- **SDO** - this is the **S**erial **D**ata **O**ut / **M**aster **I**n **S**lave **O**ut pin, for data sent from the LPS35HW to your processor
- **SDI** - this is *also* the **S**erial **D**ata **I**n / **M**aster **O**ut **S**lave **I**n pin, for data sent from your processor to the LPS35HW
- **CS** - this is the **C**hip **S**elect 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**.

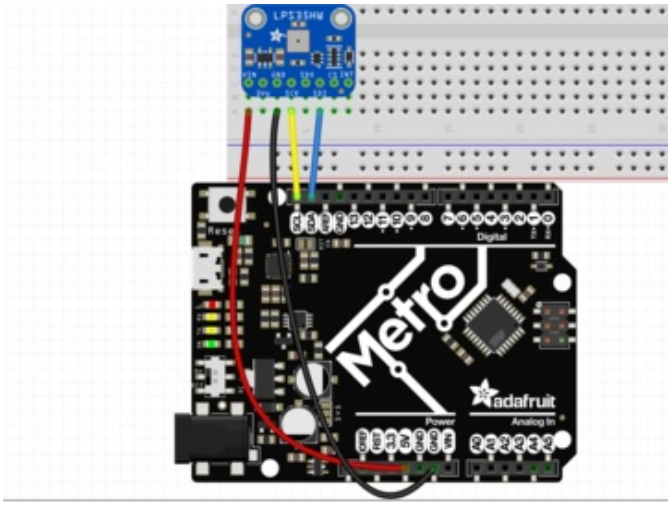
The low pressure threshold interrupt only works when the LPS35HW is operating in relative mode.

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**.

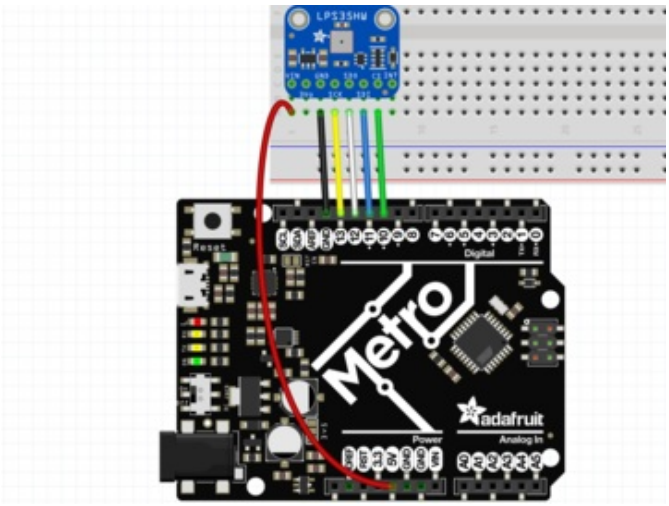


- If you are running a 5V Arduino (Uno, etc.), connect **Arduino 5V** to **board VIN**
- Connect **Arduino GND** to **board GND**
- Connect **Arduino SCL** to **board SCK**
- Connect **Arduino SDA** to **board SDI**

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:

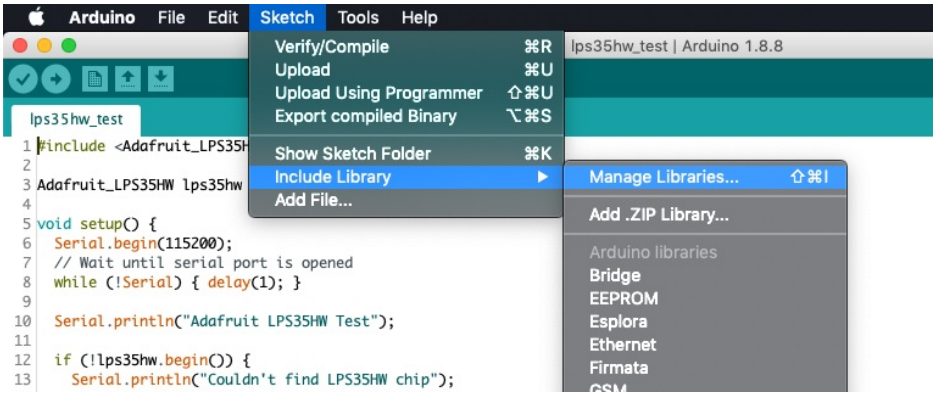


- Connect **Vin** to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of
- Connect **GND** to common power/data ground
- Connect the **SCK** pin to **Digital #13** but any pin can be used later
- Connect the **SDO** pin to **Digital #12** but any pin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

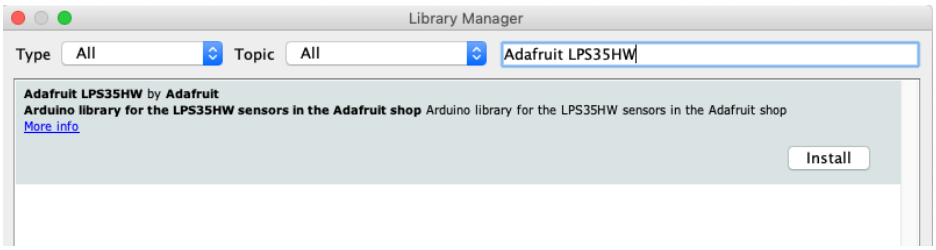
Later on, once we get it working, we can adjust the library to use hardware SPI if you desire, or change the pins to others.

Library Installation

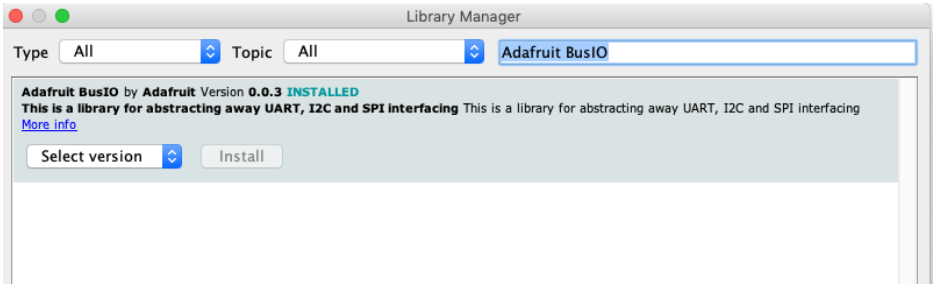
You can install the **Adafruit LPS35HW Library** for Arduino using the Library Manager in the Arduino IDE:



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 9600 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 for pressure and temperature:

```
Temporarily unable to load content:
```

Arduino Docs

Arduino Docs (https://adafru.it/ERq)

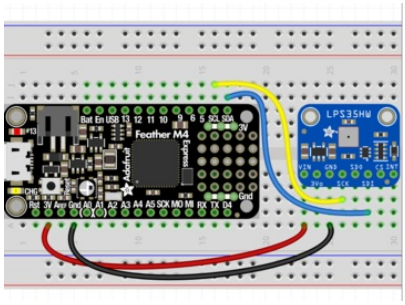
Python and CircuitPython

It's easy to use the LPS35HW sensor with Python and CircuitPython, and the Adafruit CircuitPython LPS35HW (https://adafru.it/ERZ) module. This module allows you to easily write Python code that reads the pressure and temperature from the 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 (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

First wire up a LPS35HW to your board exactly for an I2C connection as shown on the previous pages for Arduino. Here's an example of wiring a Feather M4 to the sensor with I2C:

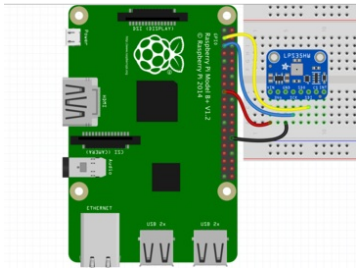


- **Board 3V** to **sensor VIN**
- **Board GND** to **sensor GND**
- **Board SCL** to **sensor SCK**
- **Board SDA** to **sensor SDI**

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).

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 (https://adafru.it/ERZ) 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/zdx). 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.

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 (https://adafru.it/ER-).

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 (https://adafru.it/Awz) 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 (https://adafru.it/BSN)!

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 (https://adafru.it/ES0).

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

Full Example Code

```
import time
import board
import adafruit_lps35hw

i2c = board.I2C()
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

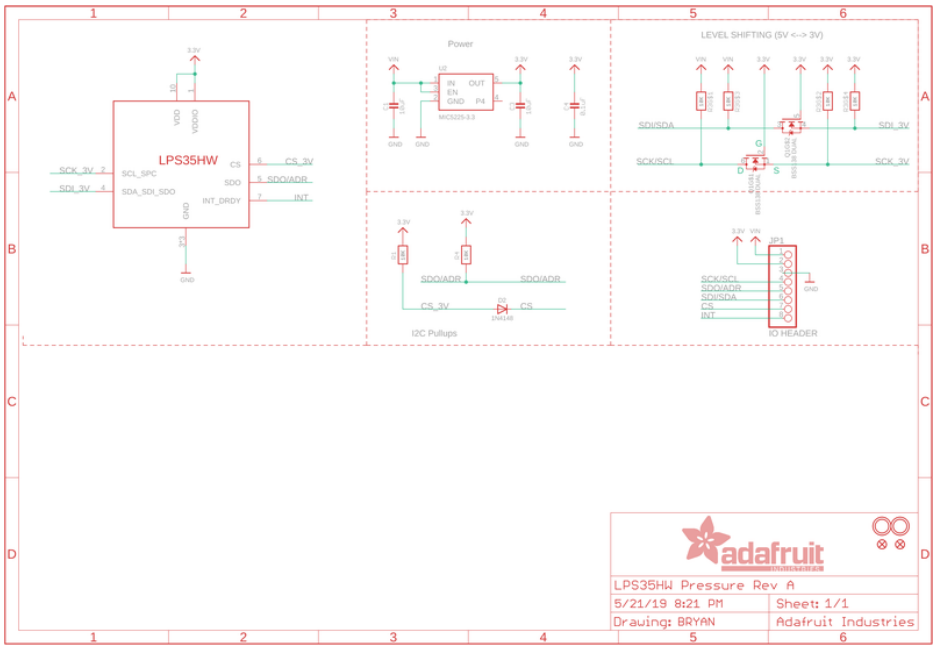
Python Docs (https://adafru.it/ERr)

Downloads

Files

- LPS35HW Datasheet (https://adafru.it/ES1)
- EagleCAD files on GitHub (https://adafru.it/ES2)
- Fritzing object from Adafruit Fritzing Library (https://adafru.it/ES3)

Schematic



Fab Print