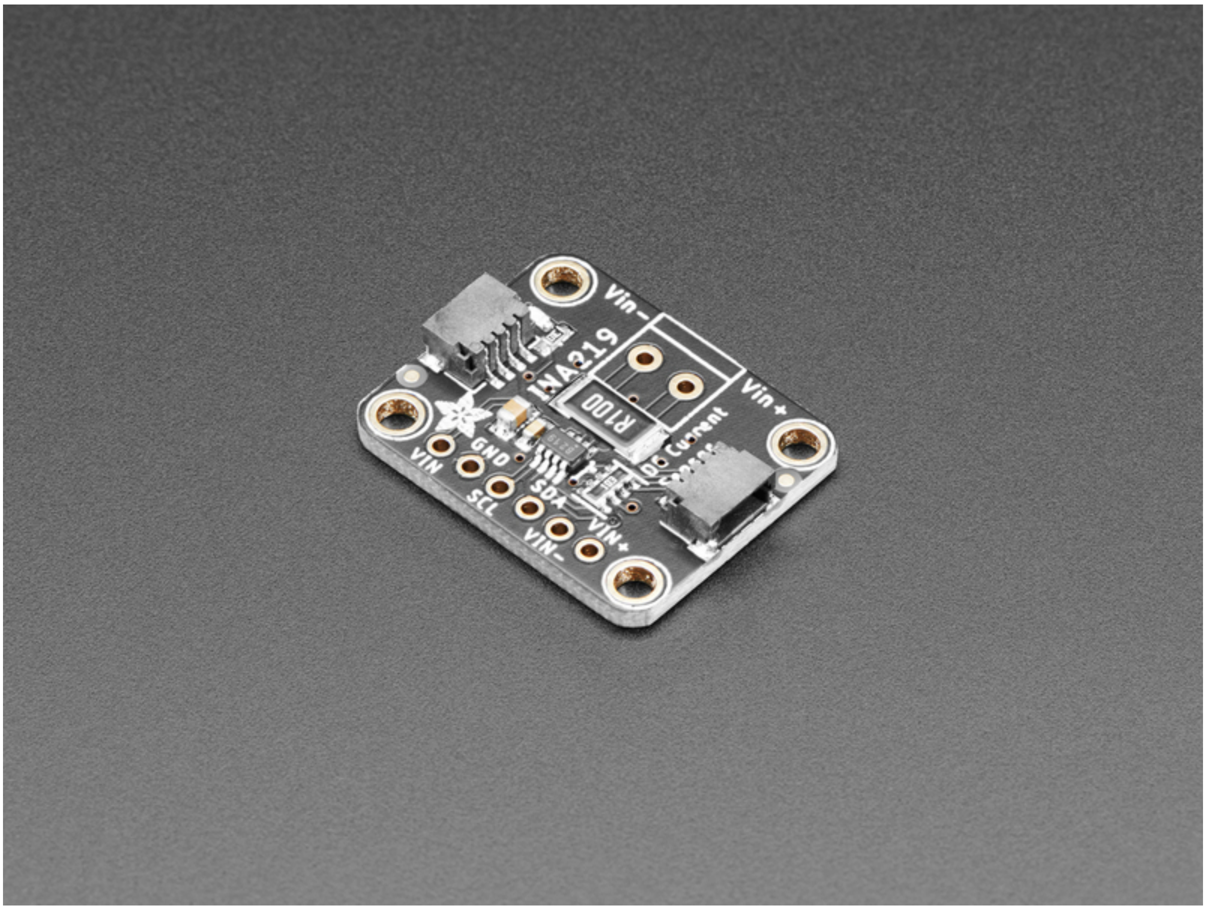
Adafruit INA219 Current Sensor Breakout

Created by lady ada

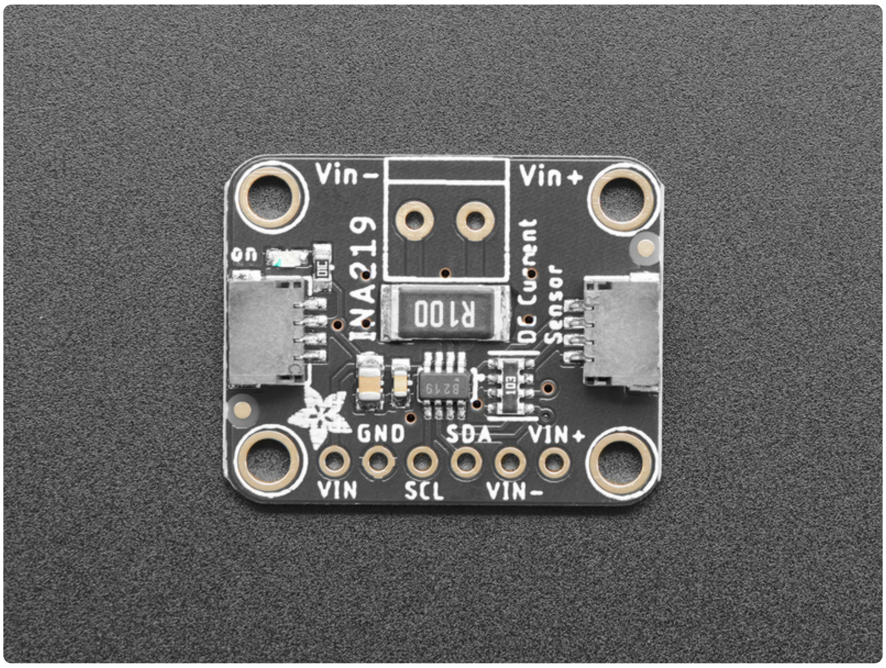


https://learn.adafruit.com/adafruit-ina219-current-sensor-breakout

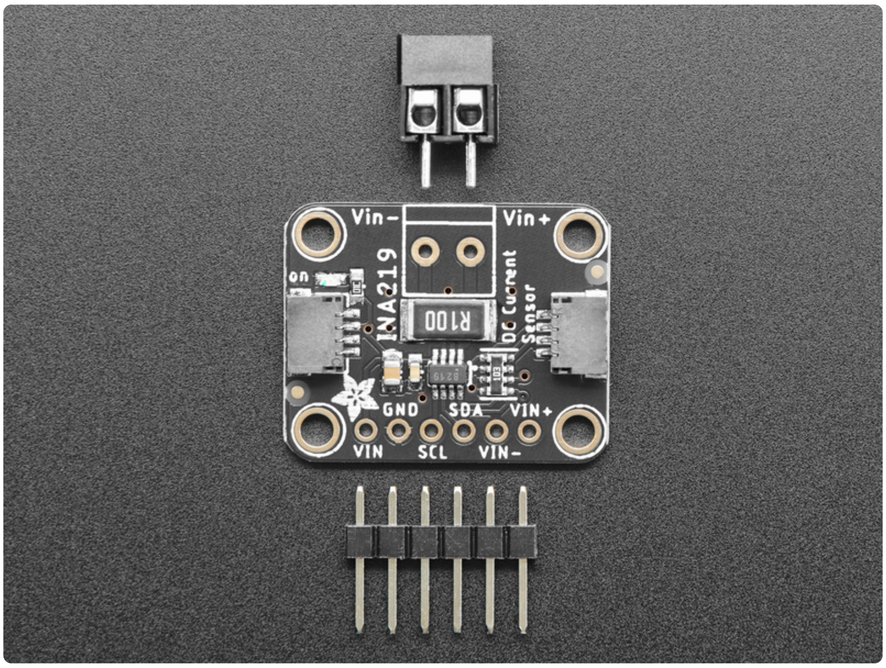
Last updated on 2024-03-08 01:42:01 PM EST

Table of Contents

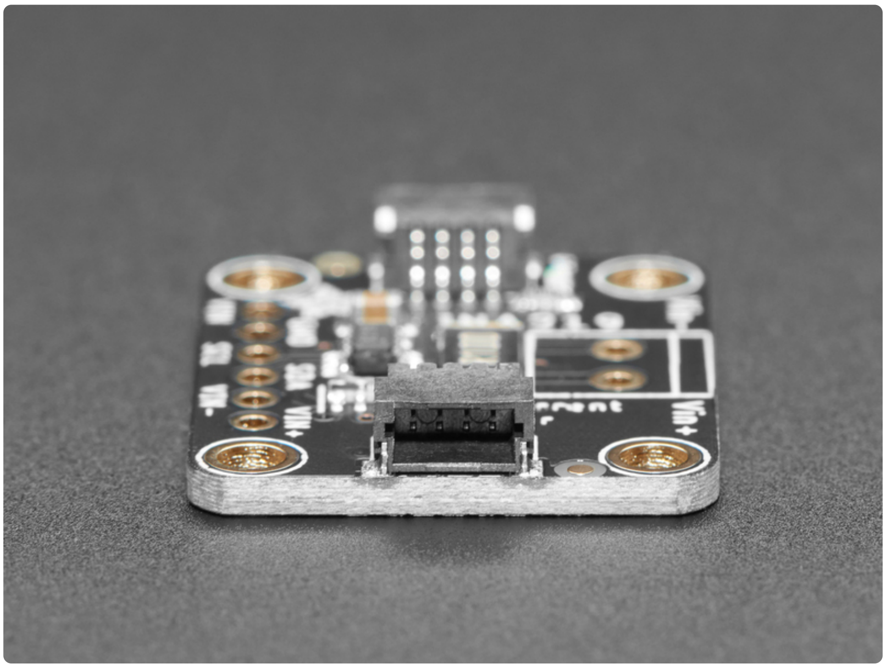
Overview



The INA219B breakout board and the INA219 FeatherWing will solve all your power-monitoring problems. Instead of struggling with two multimeters, you can use this breakout to measure both the high side voltage and DC current draw over I2C with 1% precision.



As if that weren't enough, we've now also added SparkFun qwiic (https://adafru.it/Fpw) compatible STEMMA QT **(https://adafru.it/Ft4)** connectors for the I2C bus **so you don't even need to solder the I2C and power lines.** Just wire up to your favorite micro using a STEMMA QT adapter cable. (https://adafru.it/JnB) The Stemma QT connectors also mean the INA219 can be used with our various associated accessories. (https://adafru.it/Ft6) QT Cable is not included, but we have a variety in the shop (https://adafru.it/17VE).

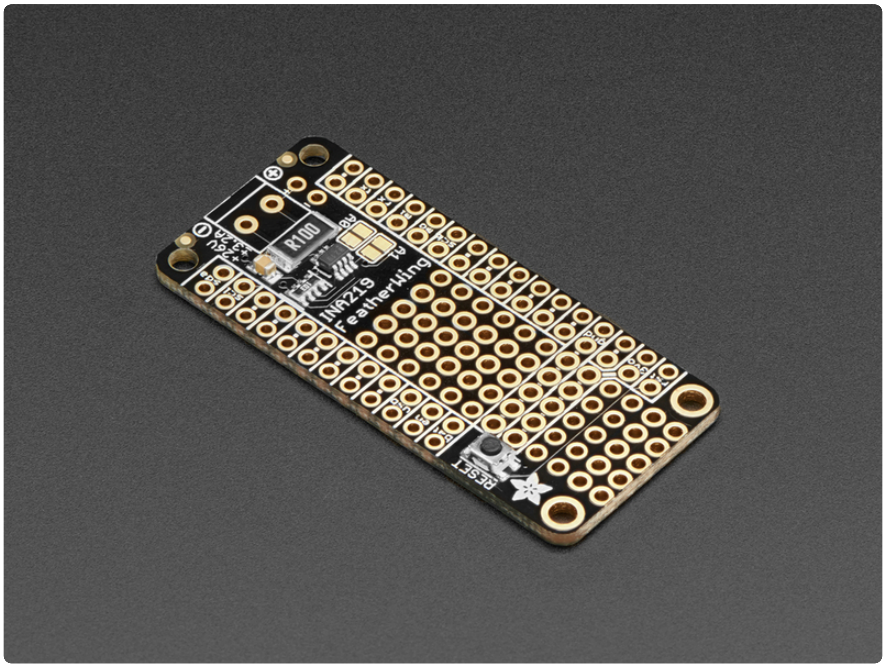


Why the High Side?

Most current-measuring devices such as our current panel meter are only good for low side measuring. That means that unless you want to get a battery involved, you have to stick the measurement resistor between the target ground and true ground.

Since the voltage drop across the resistor is proportional to the current draw, this means that the ground reference will change with varying current. Having a shifting ground reference can cause problems for many circuits.

The INA219B chip is much smarter - it can handle high side current measuring, up to +26VDC, even though it is powered with 3 or 5V. It will also report back that high side voltage, which is great for tracking battery life or solar panels.

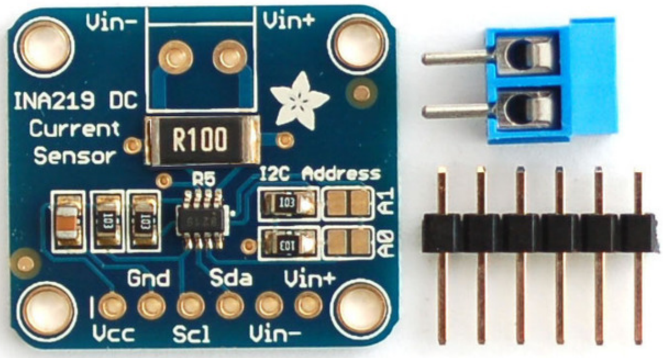


How does it work?

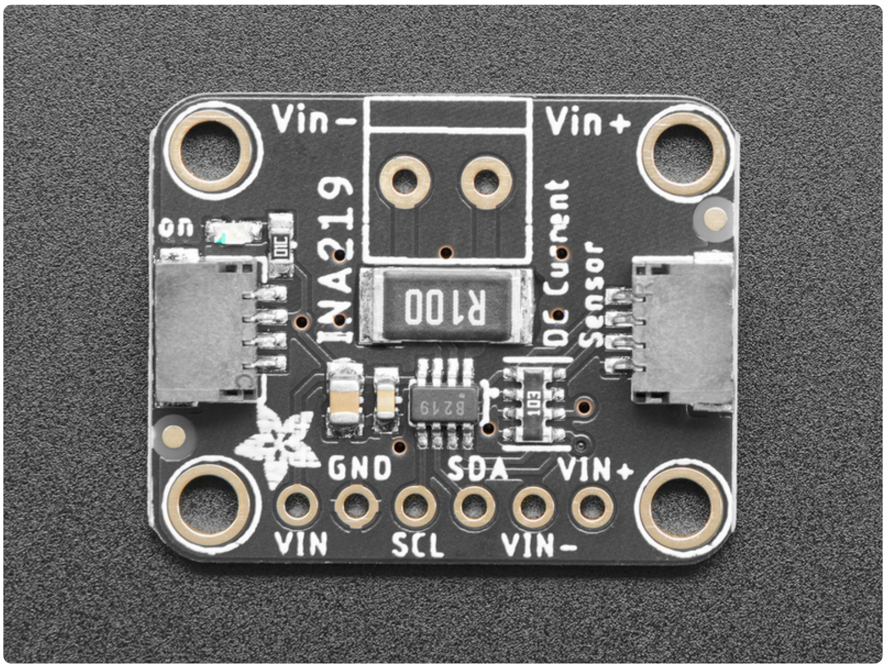
A precision amplifier measures the voltage across the 0.1 ohm, 1% sense resistor. Since the amplifier maximum input difference is ±320mV this means it can measure up to ±3.2 Amps. With the internal 12 bit ADC, the resolution at ±3.2A range is 0.8mA. With the internal gain set at the minimum of div8, the max current is ±400mA and the resolution is 0.1mA. Advanced hackers can remove the 0.1 ohm current sense resistor and replace it with their own to change the range (say a 0.01 ohm to measure up 32 Amps with a resolution of 8mA)

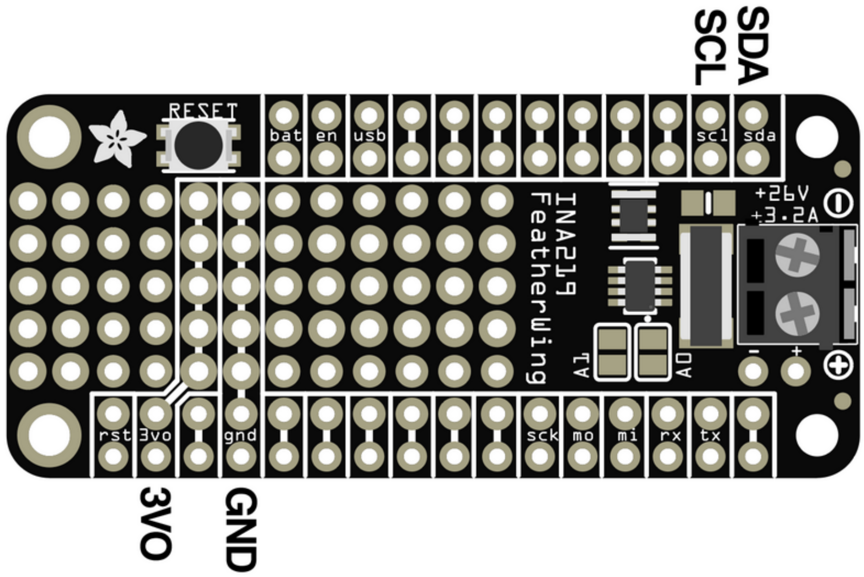
Note that when switching inductive loads, the instantaneous voltage levels may greatly exceed steady-state levels due to inductive kickback. Chip damage can occur if you do not take precautions to protect against inductive spikes.

There are two versions of this board - the STEMMA QT version shown above, and the original header-only version shown below. The code works the same on both!


Vin-
Vin+
INA219 DC
Current
Sensor
R100
R5
I2C Address
A1
A0
Gnd
Sda
Vin+
Vcc
Scl
Vin-

Pinouts


Vin-
Vin+
on
INA219
R100
DC Current
Sensor
GND
SDA
VIN+
VIN
SCL
VIN-



Power Pins

The sensor on the breakout requires between a 2.7V and 5.5V, and can be easily used with most microcontrollers from an Arduino to a Feather or something else.

- **VIN** - This is the power pin. 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, or for a Feather use 3.3V.
- **GND** - This is common ground for power and logic.

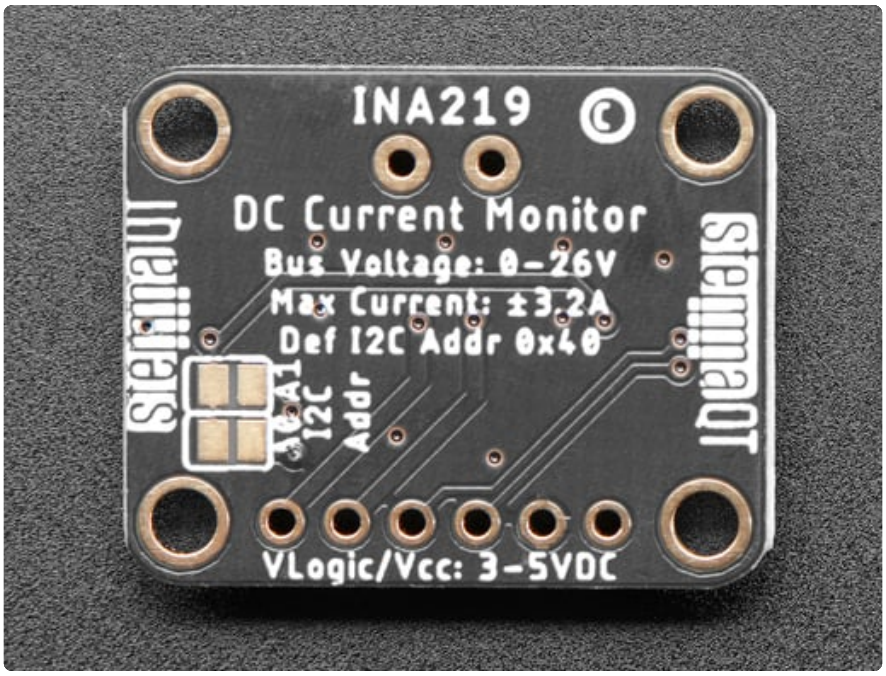
I2C Pins

Default address is **0x40**.

- **SCL** - This is the I2C **clock pin**, connect to your microcontroller's I2C clock line. There's a **10K pullup** on this pin.
- **SDA** - This is the I2C **data pin**, connect to your microcontroller's I2C data line. There's a **10K pullup** on this pin.
- **STEMMA QT** (https://adafru.it/Ft4) - These connectors allow you to connect to development boards with **STEMMA QT** connectors, or to other things, with various associated accessories (https://adafru.it/Ft6).

Other Pins

- **Vin+** is the positive input pin. Connect to supply for high side current sensing or to load ground for low side sensing.
- **Vin-** is the negative input pin. Connect to load for high side current sensing or to board ground for low side sensing
- **A0** and **A1 solder jumpers** - These can be bridged with solder to pull the address pin up to VIN to change the I2C address according to the list below.

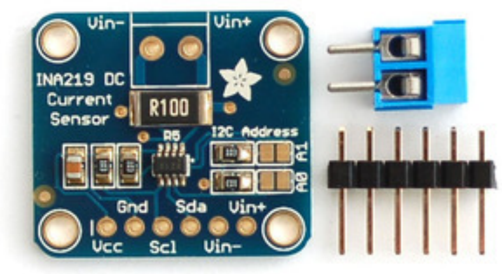


I2C Addresses Based on Jumpers

- **Default** = 0x40
- **A0 soldered** = 0x41
- **A1 soldered** = 0x44
- **A0 and A1 soldered** = 0x45

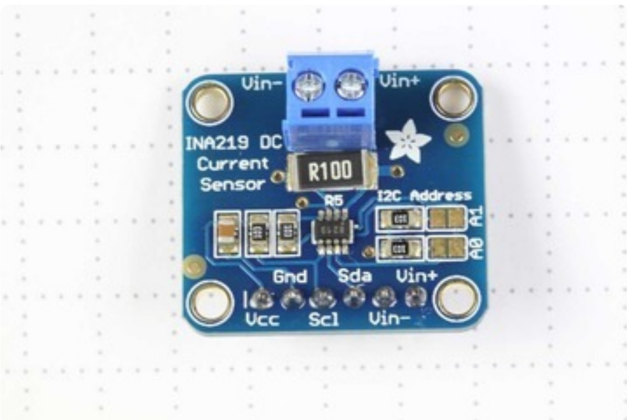
Assembly

Breakout Assembly



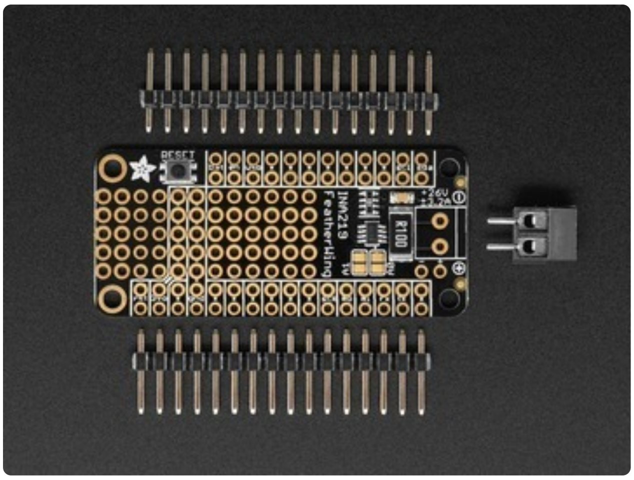
The board comes with all surface-mount components pre-soldered. Additional parts are included to help integrate the INA219 breakout board into your project.

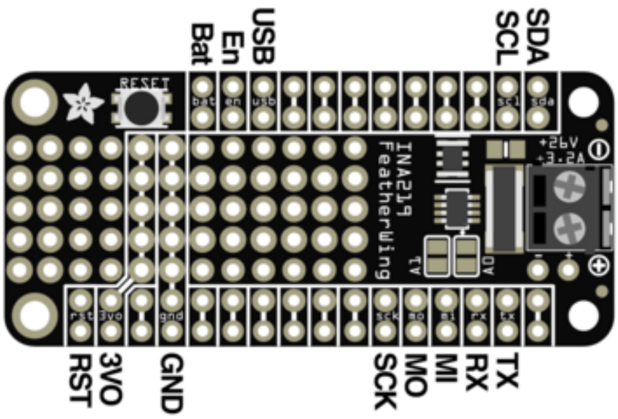
Wires can be soldered directly to the holes on the edge of the board. But for breadboard use, you will want to solder on the included 6-pin header.



The load can be connected via the header, or using the included 2-pin screw-terminal.

FeatherWing Assembly





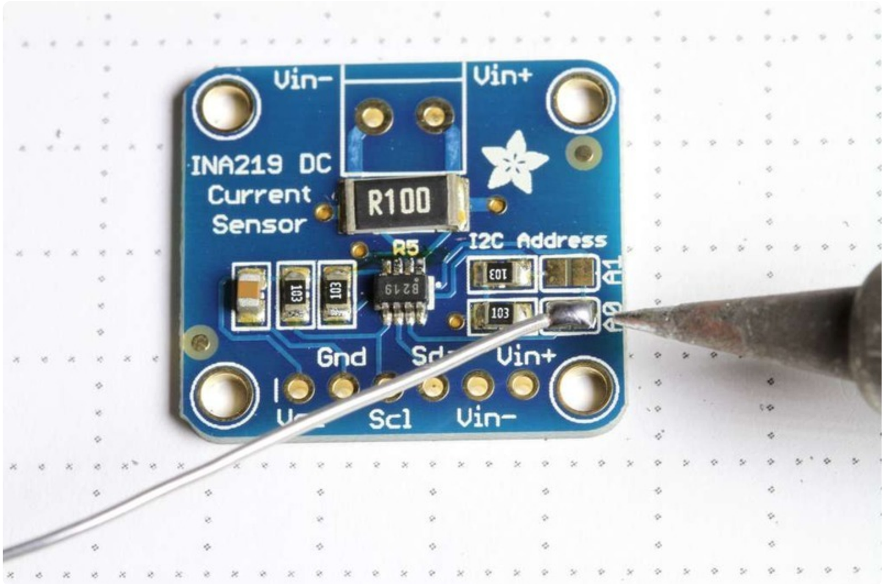
Solder the headers and the screw terminal to the board in the appropriate locations.

The pin labels are small, so here's an annotated diagram.

Addressing the Boards

If more than one INA219 breakout board is used, each board must be assigned a unique address. This is done with the address jumpers on the right edge of the board. The I2C base address for each board is 0x40. The binary address that you program with the address jumpers is added to the base I2C address.

To program the address offset, use a drop of solder to bridge the corresponding address jumper for each binary '1' in the address.

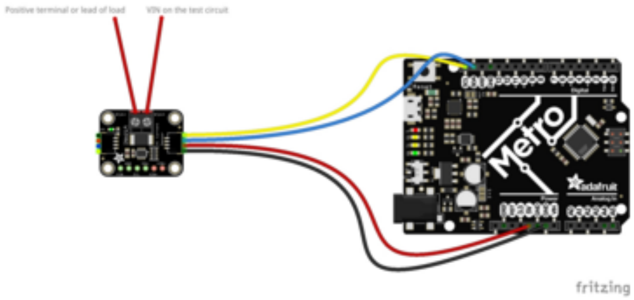


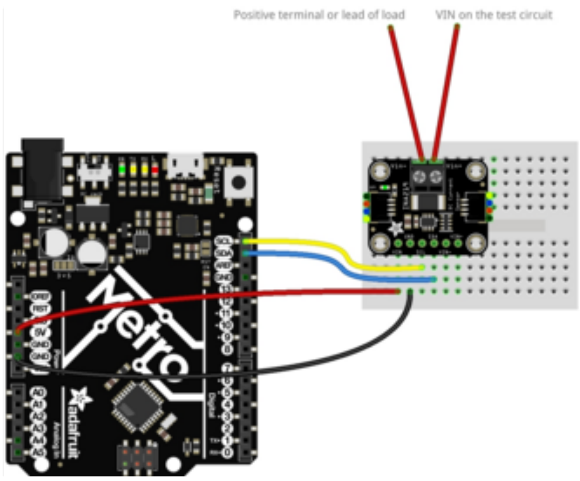
Up to 4 boards may be connected. Addressing is as follows:
Board 0: Address = 0x40 Offset = binary 00000 (no jumpers required)
Board 1: Address = 0x41 Offset = binary 00001 (bridge A0 as in the photo above)
Board 2: Address = 0x44 Offset = binary 00100 (bridge A1)
Board 3: Address = 0x45 Offset = binary 00101 (bridge A0 & A1)

Wiring

The INA219 breakout board can be powered by the 5V or 3V pin on your Arduino and communicates via I2C.

To wire up the STEMMA QT version:





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

Connect **board GND (black wire)** to **Arduino GND**

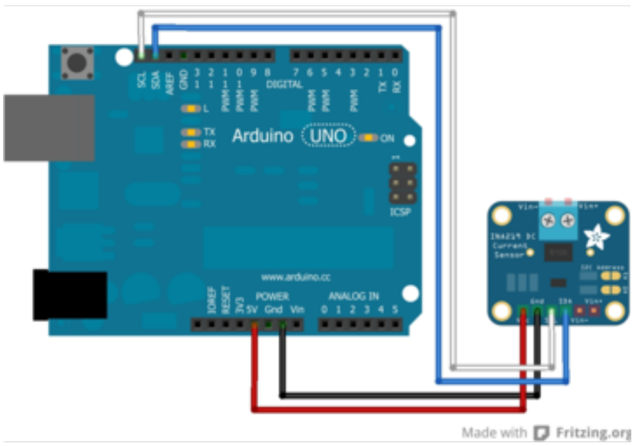
Connect **board SCL (yellow wire)** to **Arduino SCL**

Connect **board SDA (blue wire)** to **Arduino SDA**

Connect **Vin+** to the **positive terminal of the power supply for the circuit under test**

Connect **Vin-** to the **positive terminal or lead of the load**

To wire up the original version:



Connect to the microcontroller

Connect **GND** to **GND**

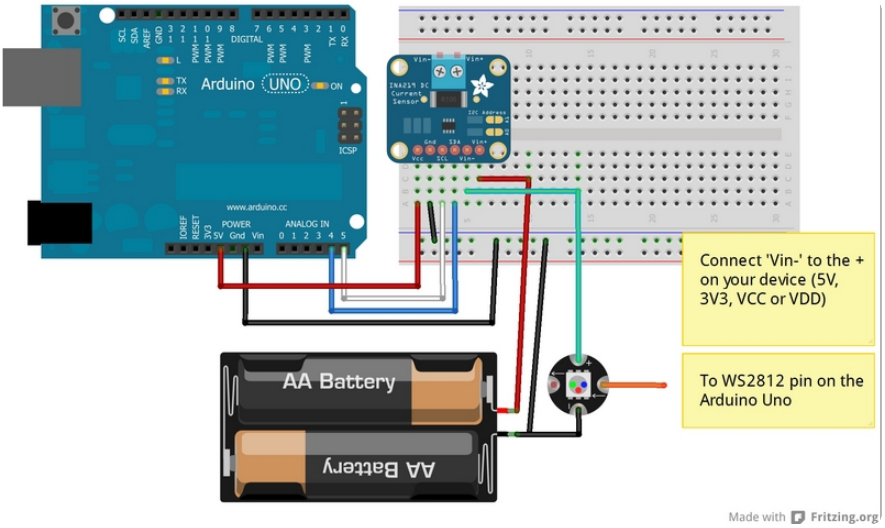
Connect **VCC** to **5v**

Then connect **SDA** to **SDA** (Analog pin 4 on pre-R3 Arduinos)

And connect **SCL** to **SCL** (Analog pin 5 on pre-R3 Arduinos)

On R3 and later Arduinos, you can connect to the new dedicated SDA & SCL pins next to the AREF pin. On pre-R3 Megas, SDA & SCL are on pins 20 & 21.

Next we must insert the INA219 current sensor into the circuit we want to measure:



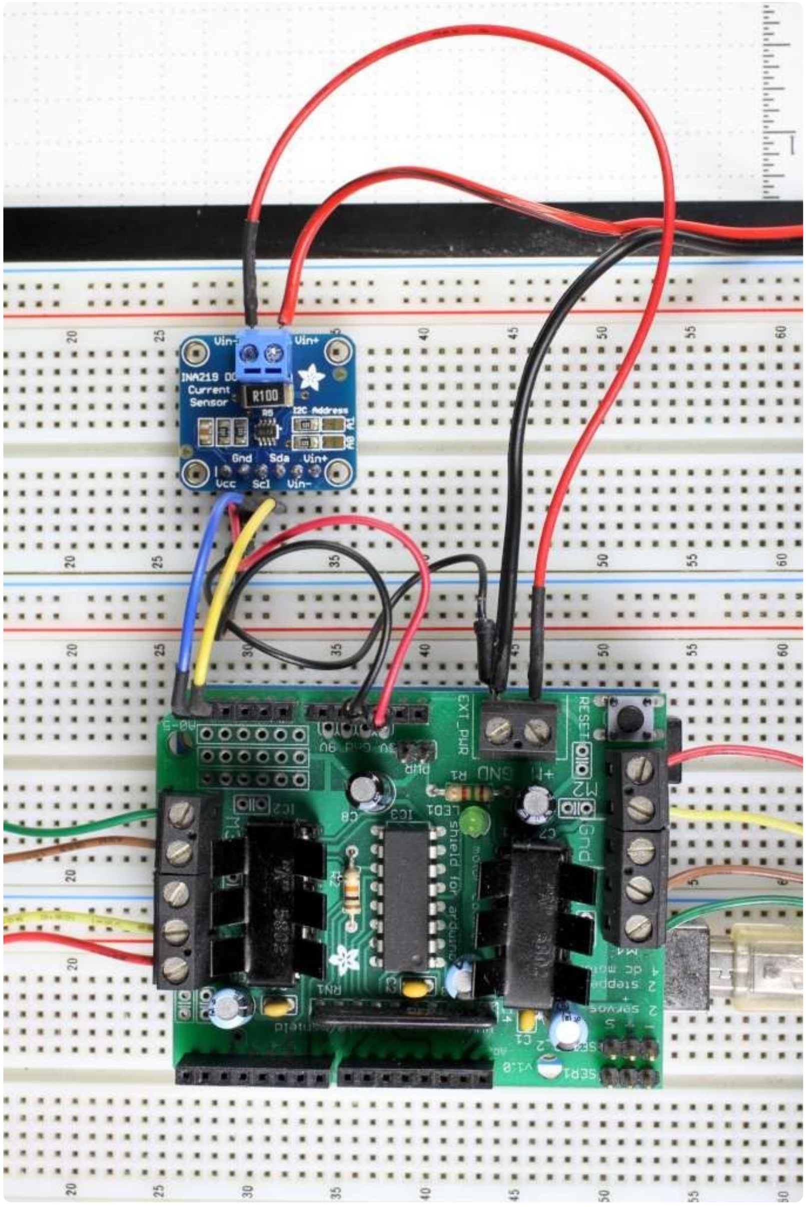
Be careful inserting noisy loads that can cause a sharp current draw, such as DC motors, since they can cause problems on the power lines and may cause the INA219 to reset, etc. When using a DC motor or a similar device, be sure to include a large capacitor to decouple the motor from the power supply and use a snubber diode to protect against inductive spikes.

Connect to the circuit

- Connect V+ to the positive terminal of the power supply for the circuit under test.
- Connect V- to the positive terminal or lead of the load. This puts the sense resistor in-line with the circuit.
- Finally, connect a wire from the negative terminal of the power supply to GND. This allows the sensor to measure the load voltage as well as the load current.

The photo below shows an INA219 breakout board configured to measure the motor supply current on an Adafruit Motor Shield.

All voltages are measured relative to the processor ground, so a common ground connection is necessary to accurately measure the load voltages.

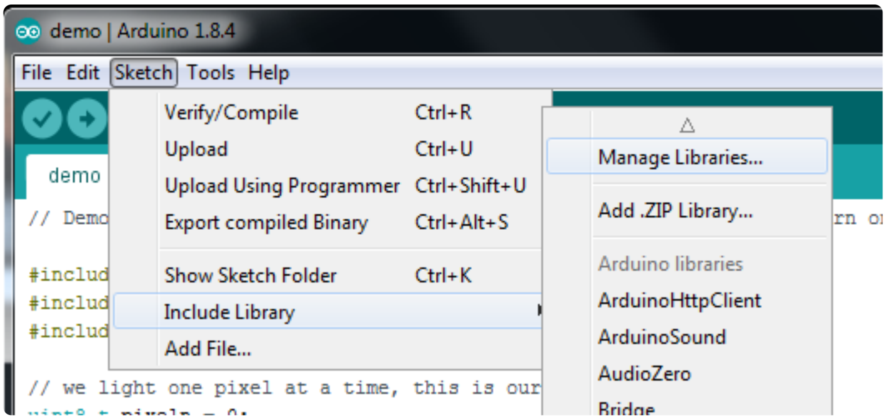


Arduino Code

Programming the Adafruit INA219 breakout board is simple using our library:

Install the Library

Open up the Arduino library manager:

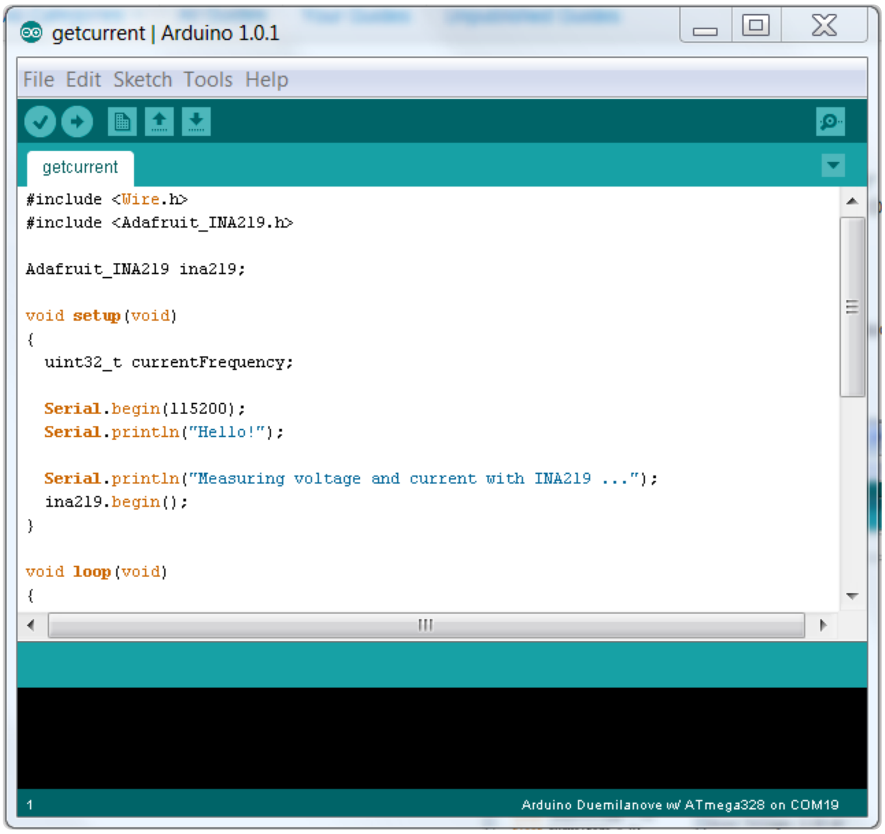


Search for the **Adafruit INA219** library and install it

We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (https://adafru.it/aYM)

Load the Example

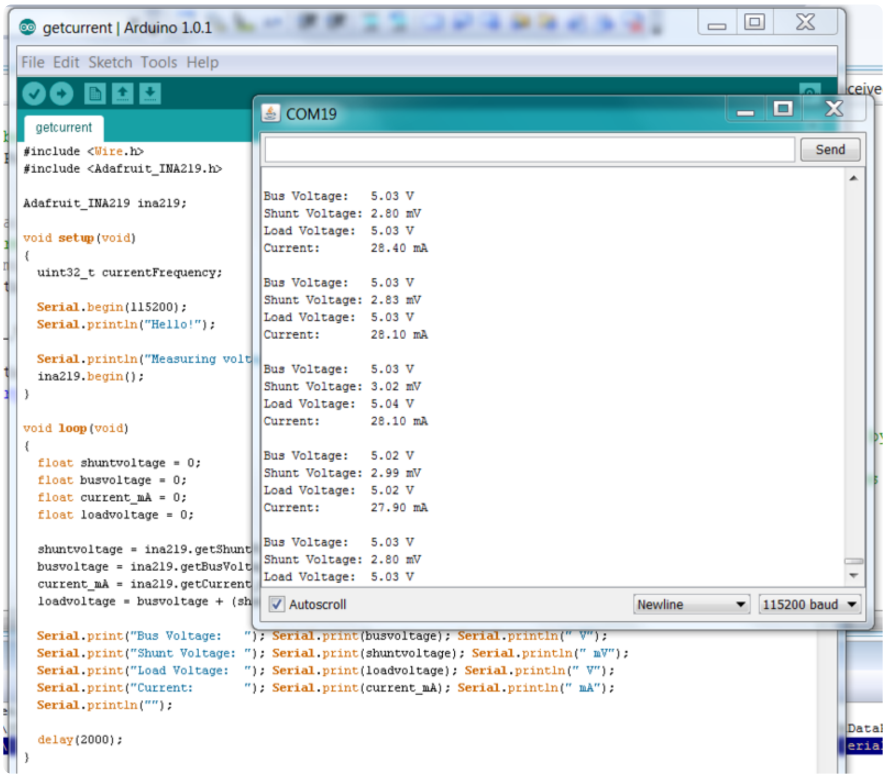
- **Select** "File -> Examples -> Adafruit_INA219 -> getcurrent"
- The "getcurrent" example code should open in a new IDE window.



Run it

- **Click** on the upload button in the IDE. When it is "done uploading"
- **Open** your Serial Monitor and set the speed to 115200 baud.

The output should appear similar to the sample below:



Customize it

You can adapt, expand or modify the example code to suit your project requirements. For a detailed description of the available library functions, see the **Library Reference** on the next page.

Library Reference

Construction and Initialization Functions:

`Adafruit_INA219(uint8_t addr = INA219_ADDRESS);`

Constructs an instance of the `Adafruit_INA219`. If no address is specified, the default address (0x40) is used. If more than one INA219 module is connected, it should be addressed as shown on the Assembly page and the configured address passed to the constructor.

`void begin(void);`

Initializes I2C communication with the `Adafruit_INA219` device using the default configuration values.

Example:

```
#include &lt;Wire.h&gt;
#include &lt;Adafruit_INA219.h&gt;

Adafruit_INA219 ina219_A;
Adafruit_INA219 ina219_B(0x41);


void setup(void)
{
  ina219_A.begin();  // Initialize first board (default address 0x40)
  ina219_B.begin();  // Initialize second board with the address 0x41
}
```

Sensor Reading Functions:

`float getBusVoltage_V(void);`

Reads the voltage between GND and V-. This is the total voltage seen by the circuit under test. (Supply voltage - shunt voltage).

The return value is in Volts.

`float getShuntVoltage_mV(void);`

Reads the voltage between V- and V+. This is the measured voltage drop across the shunt resistor.

The return value is in Milivolts.

`float getCurrent_mA(void);`

Reads the current, derived via Ohms Law from the measured shunt voltage.

The return value is in Milliamps.

Example:

```
  float shuntvoltage = 0;
  float busvoltage = 0;
  float current_mA = 0;
  float loadvoltage = 0;

  shuntvoltage = ina219.getShuntVoltage_mV();
  busvoltage = ina219.getBusVoltage_V();
  current_mA = ina219.getCurrent_mA();
  loadvoltage = busvoltage + (shuntvoltage / 1000);
```

```
  Serial.print("Bus Voltage:   "); Serial.print(busvoltage); Serial.println(" V");
  Serial.print("Shunt Voltage: "); Serial.print(shuntvoltage); Serial.println("
mV");
  Serial.print("Load Voltage:  "); Serial.print(loadvoltage); Serial.println(" V");
  Serial.print("Current:       "); Serial.print(current_mA); Serial.println(" mA");
  Serial.println("");
```

Arduino Library Docs

Arduino Library Docs (https://adafru.it/DYv)

Python & CircuitPython

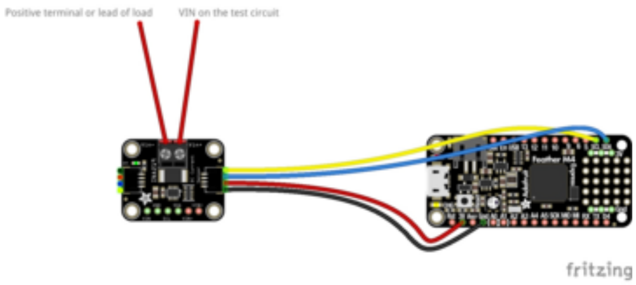
It's easy to use the INA219 sensor with Python and CircuitPython, and the Adafruit CircuitPython INA219 (https://adafru.it/BHQ) module. This module allows you to easily write Python code that reads the current and more from the sensor.

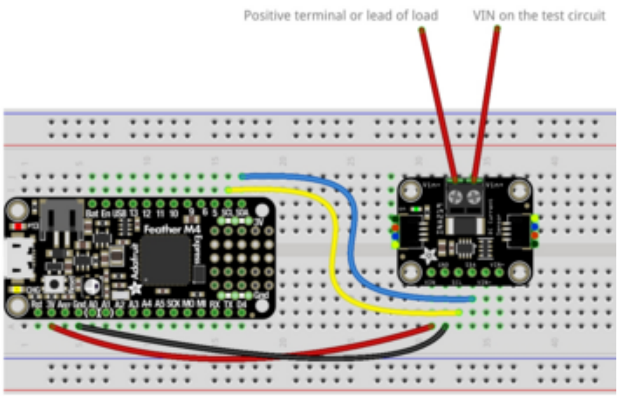
You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

First wire up a INA219 to your board exactly as shown on the previous pages for Arduino using an I2C interface. In addition connect some load to measure the current from in series to the sensor's **Vin-** and **Vin+** pins as mentioned on the wiring page (https://adafru.it/BHR).

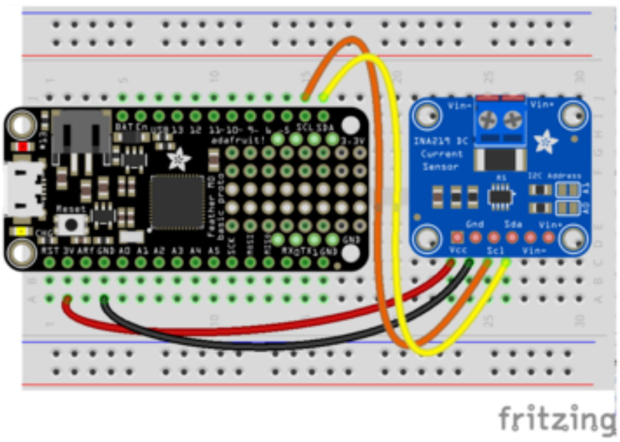
Here is an example of the STEMMA QT version connected to a Feather M4:





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)**
Connect **Vin+** to the **positive terminal of the power supply for the circuit under test**
Connect **Vin-** to the **positive terminal or lead of the load**

Here's an example of the original version of the sensor wired up to a Feather M0:



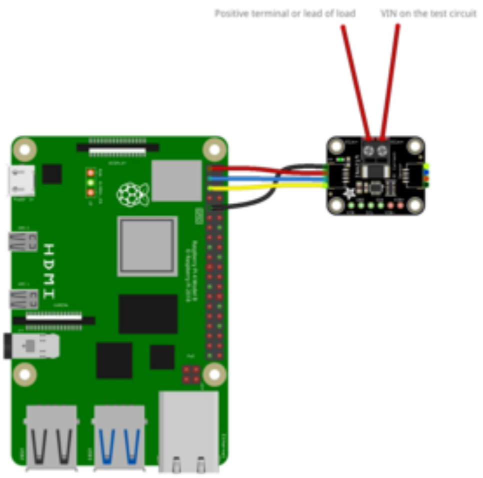
Board 3V to **sensor Vcc**
Board GND to **sensor GND**
Board SCL to **sensor SCL**
Board SDA to **sensor SDA**

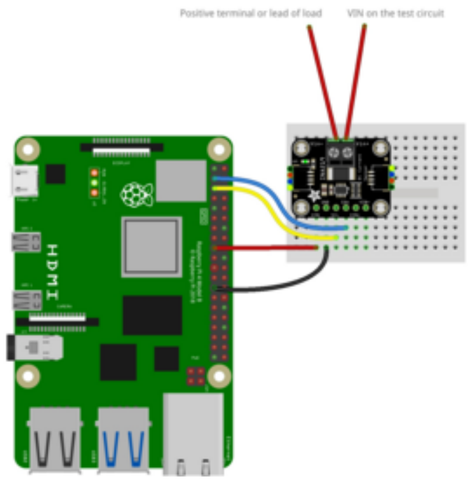
Python Computer Wiring

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).

In addition connect some load to measure the current from in series to the sensor's **Vin-** and **Vin+** pins as [mentioned on the wiring page](https://adafru.it/BHR) (https://adafru.it/BHR).

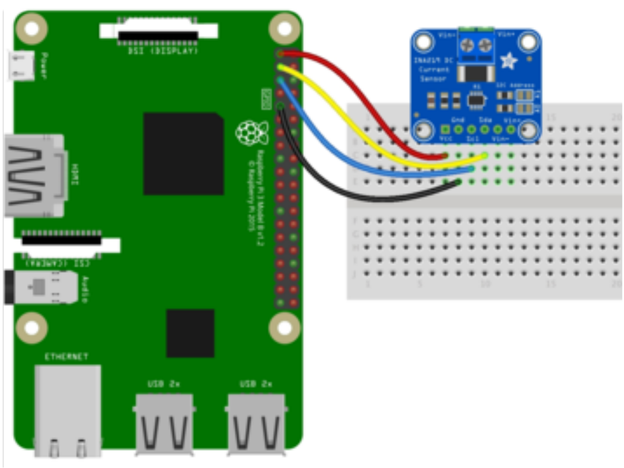
Here's the Raspberry Pi wired to the STEMMA QT version of the sensor:





Pi 3V to **sensor VIN (red wire)**
Pi GND to **sensor GND (black wire)**
Pi SCL to **sensor SCL (yellow wire)**
Pi SDA to **sensor SDA (blue wire)**
Connect **Vin+** to the **positive terminal of the power supply for the circuit under test**
Connect **Vin-** to the **positive terminal or lead of the load**

Here's the Raspberry Pi wired to the original version of the sensor with I2C:



Pi 3V3 to **sensor Vcc**
Pi GND to **sensor Gnd**
Pi SCL to **sensor Scl**
Pi SDA to **sensor Sda**

CircuitPython Installation of INA219 Library

Next you'll need to install the Adafruit CircuitPython INA219 (https://adafru.it/BHQ) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/tBa) 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). For example the Circuit Playground Express 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 Trinket M0, Gemma M0, and Feather/Metro M0 basic you'll need to manually install the necessary libraries from the bundle:

- **adafruit_ina219.mpy**
- **adafruit_bus_device**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_ina219.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 INA219 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-ina219`

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 current and more from the board's Python REPL. Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import board
import busio
import adafruit_ina219
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_ina219.INA219(i2c)
```

Now you're ready to read values from the sensor using any of these functions:

- **shunt_voltage** - The shunt voltage in volts.
- **bus_voltage** - The bus voltage in volts.
- **current** - The current in milliamps.

```
print("Bus Voltage:   {} V".format(ina219.bus_voltage))
print("Shunt Voltage: {} mV".format(ina219.shunt_voltage / 1000))
print("Current:       {} mA".format(ina219.current))
```

That's all there is to using the INA219 with CircuitPython!

Here's a full example to print the voltage and current every second. Save this as **code.py** on your board's filesystem and check the output from the serial REPL.

Full Example Code

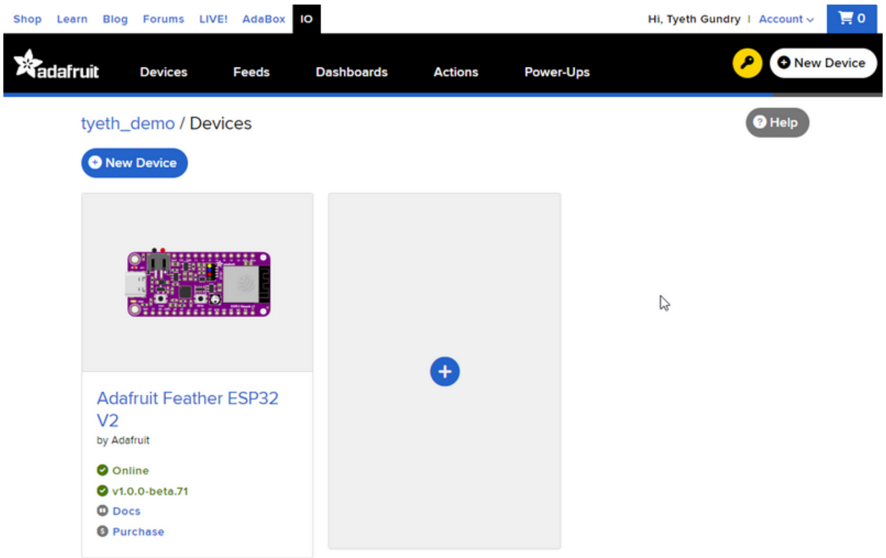
```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""Sample code and test for adafruit_ina219"""

import time
import board
from adafruit_ina219 import ADCResolution, BusVoltageRange, INA219


i2c_bus = board.I2C()  # uses board.SCL and board.SDA
# i2c_bus = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller

ina219 = INA219(i2c_bus)

print("ina219 test")
```

```
# display some of the advanced field (just to test)
print("Config register:")
print("  bus_voltage_range:    0x%1X" % ina219.bus_voltage_range)
print("  gain:                 0x%1X" % ina219.gain)
print("  bus_adc_resolution:   0x%1X" % ina219.bus_adc_resolution)
print("  shunt_adc_resolution: 0x%1X" % ina219.shunt_adc_resolution)
print("  mode:                 0x%1X" % ina219.mode)
print("")

# optional : change configuration to use 32 samples averaging for both bus voltage
and shunt voltage
ina219.bus_adc_resolution = ADCResolution.ADCRES_12BIT_32S
ina219.shunt_adc_resolution = ADCResolution.ADCRES_12BIT_32S
# optional : change voltage range to 16V
ina219.bus_voltage_range = BusVoltageRange.RANGE_16V

# measure and display loop
while True:
    bus_voltage = ina219.bus_voltage  # voltage on V- (load side)
    shunt_voltage = ina219.shunt_voltage  # voltage between V+ and V- across the
shunt
    current = ina219.current  # current in mA
    power = ina219.power  # power in watts

    # INA219 measure bus voltage on the load side. So PSU voltage = bus_voltage +
shunt_voltage
    print("Voltage (VIN+) : {:6.3f}   V".format(bus_voltage + shunt_voltage))
    print("Voltage (VIN-) : {:6.3f}   V".format(bus_voltage))
    print("Shunt Voltage  : {:8.5f} V".format(shunt_voltage))
    print("Shunt Current  : {:7.4f}  A".format(current / 1000))
    print("Power Calc.    : {:8.5f} W".format(bus_voltage * (current / 1000)))
    print("Power Register : {:6.3f}   W".format(power))
    print("")

    # Check internal calculations haven't overflowed (doesn't detect ADC overflows)
    if ina219.overflow:
        print("Internal Math Overflow Detected!")
        print("")

    time.sleep(2)
```

Python Docs

[Python Docs](https://adafru.it/C4O) (https://adafru.it/C4O)

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 (https://adafru.it/fsU), a web platform designed (by Adafruit! (https://adafru.it/Bo5)) 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.

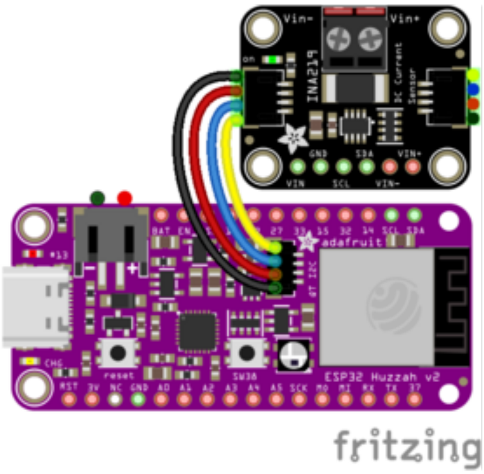
If you've never used WipperSnapper, click below to read through the quickstart guide before continuing.

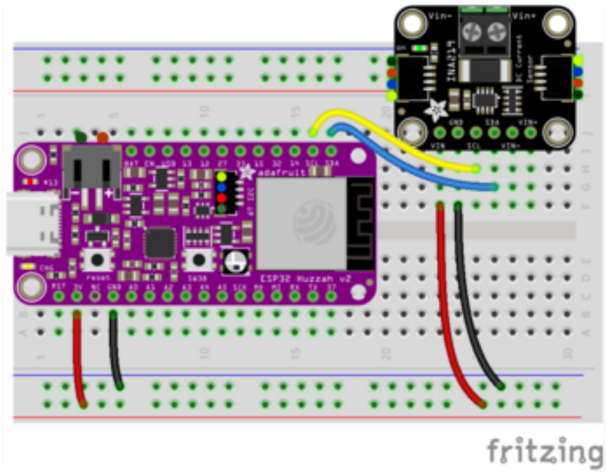
Quickstart: Adafruit IO WipperSnapper

https://adafru.it/Vfd

Wiring

First, wire up an **INA219** to your board exactly as follows. Here is an example of the **INA219** wired to an [Adafruit ESP32 Feather V2](http://adafru.it/5400) (http://adafru.it/5400) using I2C [with a STEMMA QT cable (no soldering required)](http://adafru.it/4210) (http://adafru.it/4210)



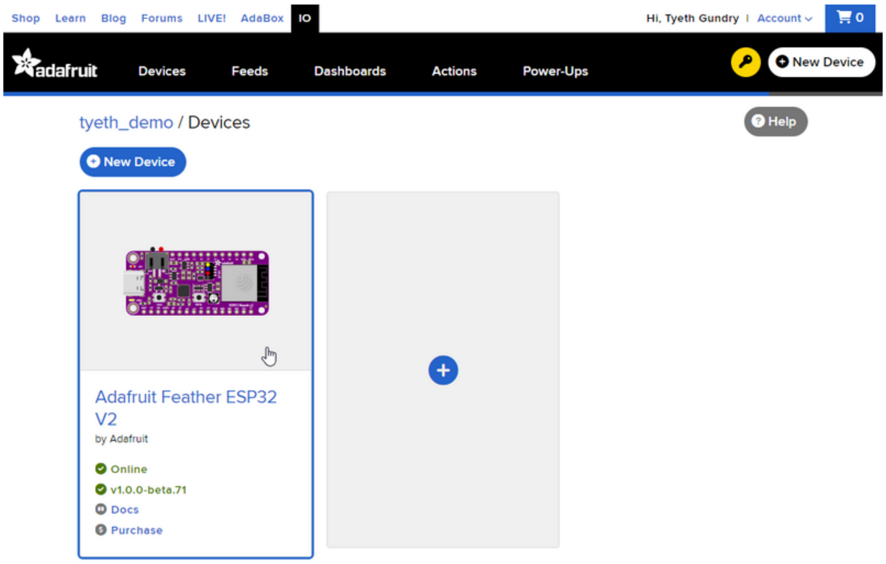


Board 3V to **sensor VIN (red wire on STEMMA QT)**
Board GND to **sensor GND (black wire on STEMMA QT)**
Board SCL to **sensor SCL (yellow wire on STEMMA QT)**
Board SDA to **sensor SDA (blue wire on STEMMA QT)**

Usage

Connect your board to Adafruit IO Wippersnapper and [navigate to the WipperSnapper board list](https://adafru.it/TAu) **(https://adafru.it/TAu).**

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



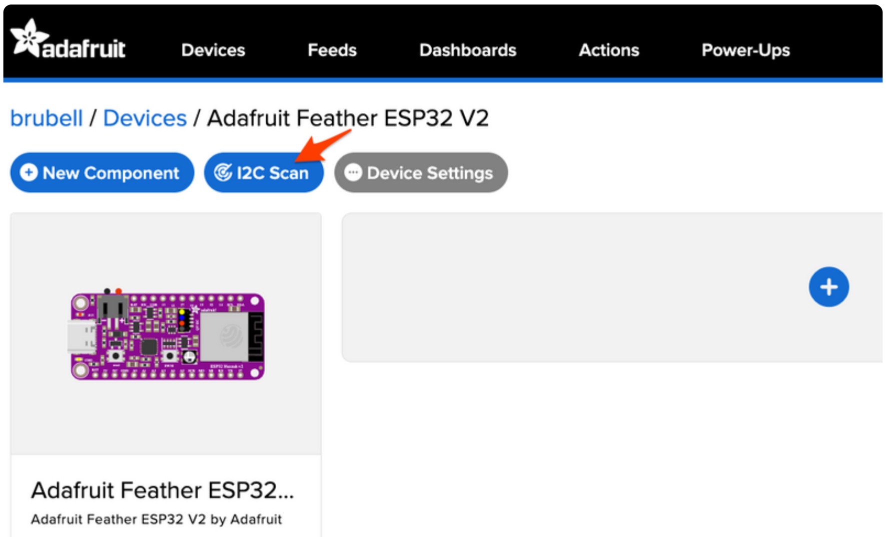
If you do not see your board listed here - you need [to connect your board to Adafruit IO (https://adafru.it/Vfd)](https://adafru.it/Vfd) first.

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

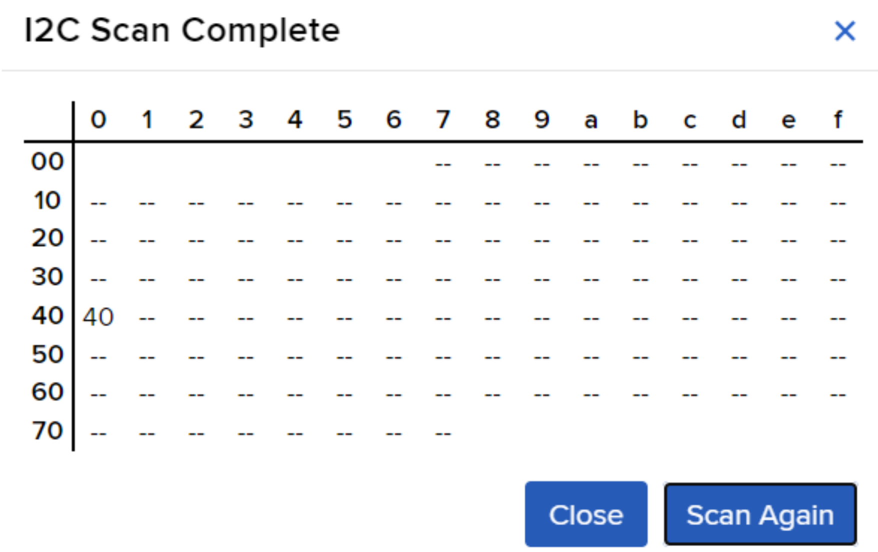
The device tile on the left indicates the version number of the firmware running on the connected board.

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 (https://adafru.it/Vfd)](https://adafru.it/Vfd) 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 **INA219**'s default I2C address of `0x40` pop-up in the I2C scan list.



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

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

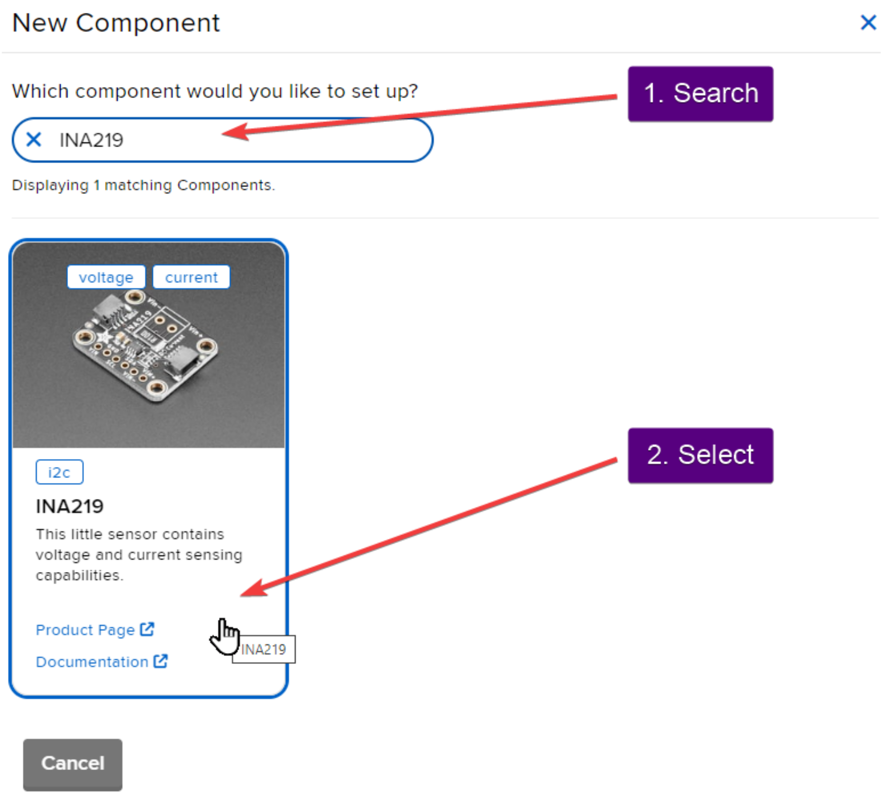
First, double-check the connection and/or wiring between the sensor and the board.

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.

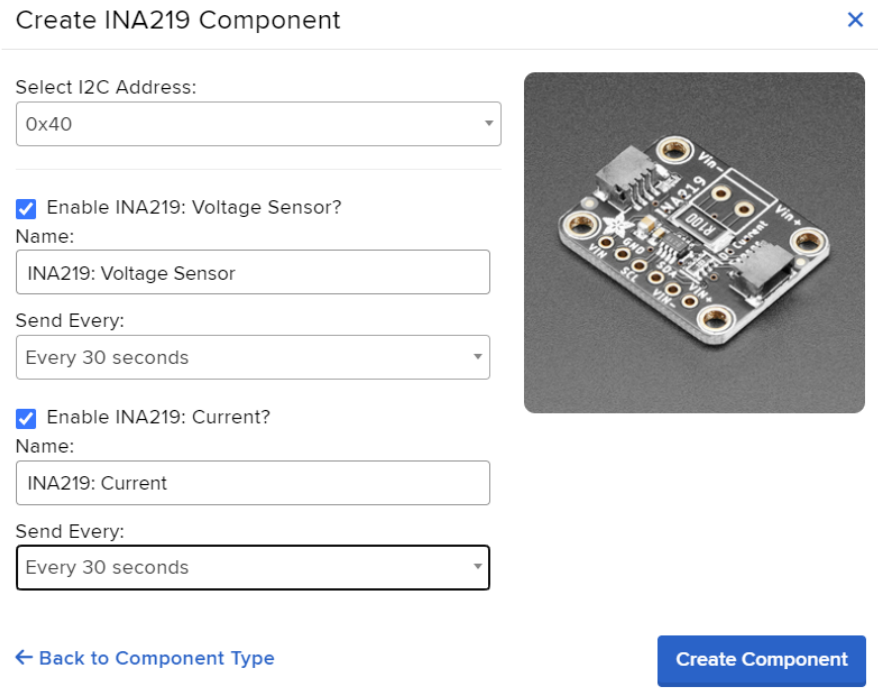
Adafruit IO supports a large amount of components. To quickly find your sensor, type `INA219` into the search bar, then select the **INA219** component.



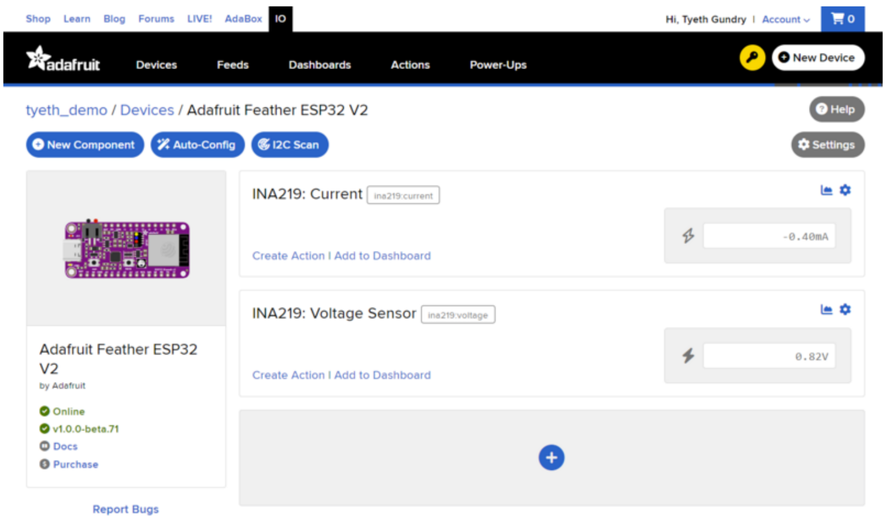
On the component configuration page, the **INA219**'s sensor address should be listed along with the sensor's settings.

The **Send Every** option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the **INA219** sensor and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

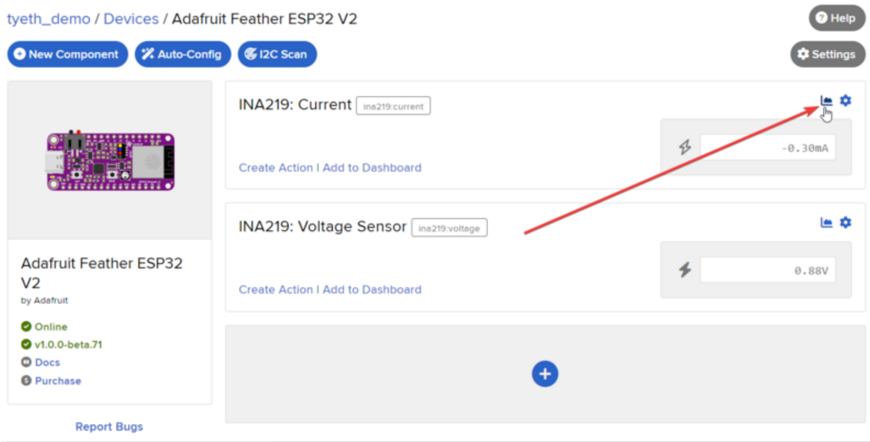
For this example, set the **Send Every** interval to every 30 seconds.



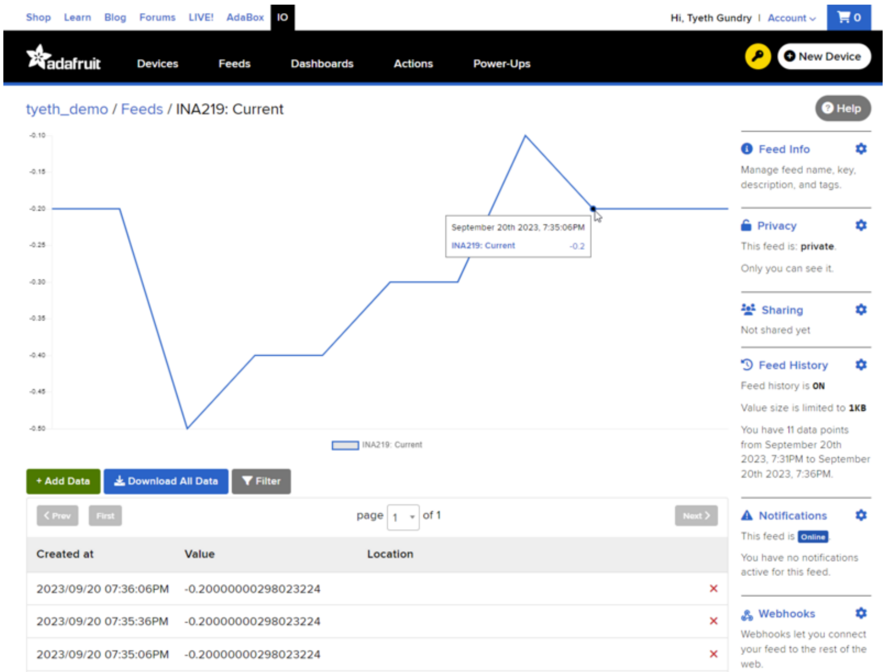
Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.



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



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](https://adafru.it/10aZ) (https://adafru.it/10aZ).

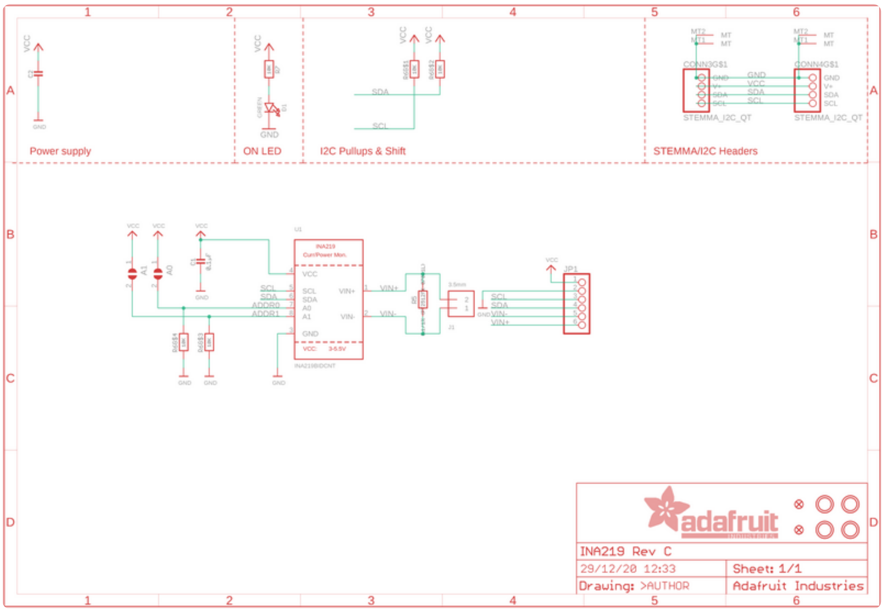


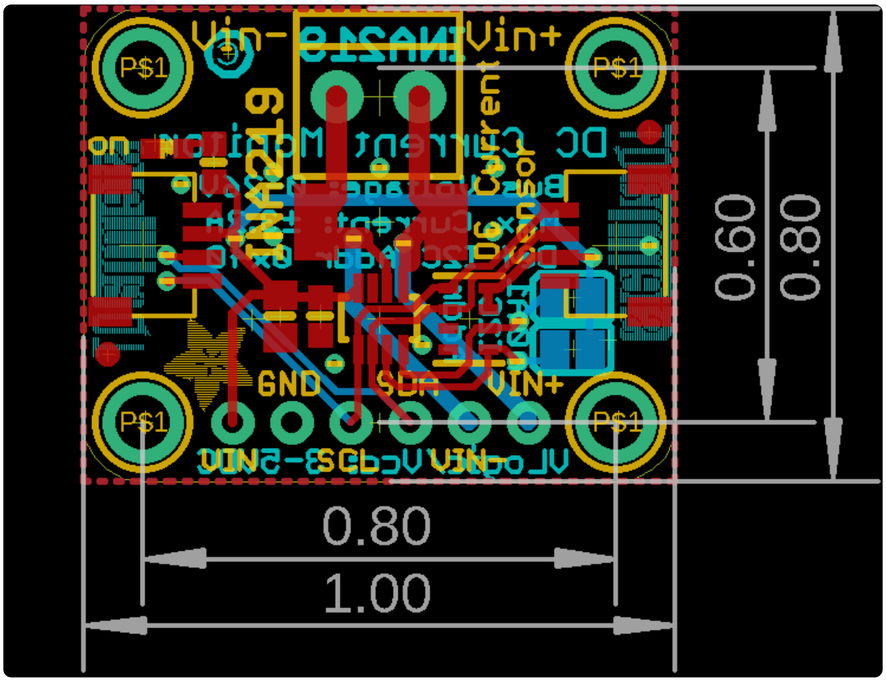
Downloads

Datasheets & Files

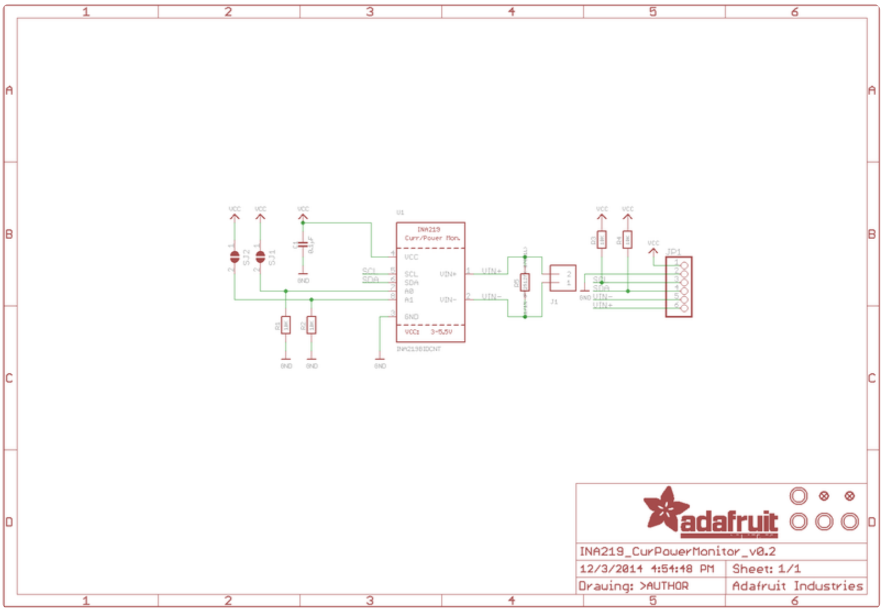
- [Eagle PCB files for the INA219 breakout board](https://adafru.it/aRk) (https://adafru.it/aRk)
- [3D models on GitHub](https://adafru.it/11A5) (https://adafru.it/11A5)
- [Arduino driver library](https://adafru.it/aRj) (https://adafru.it/aRj)
- [Data Sheet for the INA219 chip](https://adafru.it/aRl) (https://adafru.it/aRl)
- [Fritzing object in the Adafruit Fritzing library](https://adafru.it/aP3) (https://adafru.it/aP3)

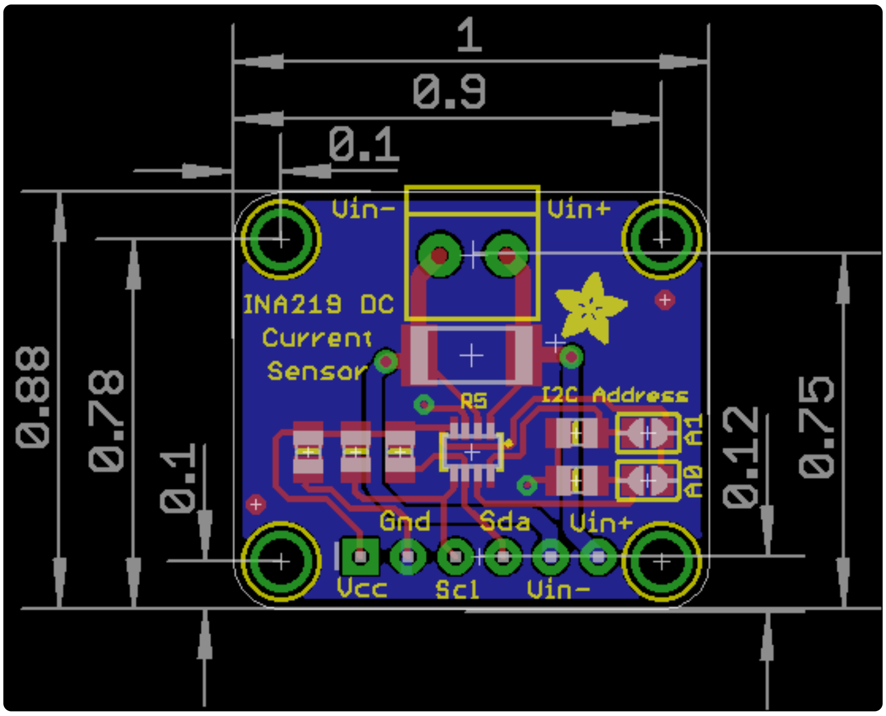
Schematic and Fab Print STEMMA QT Version



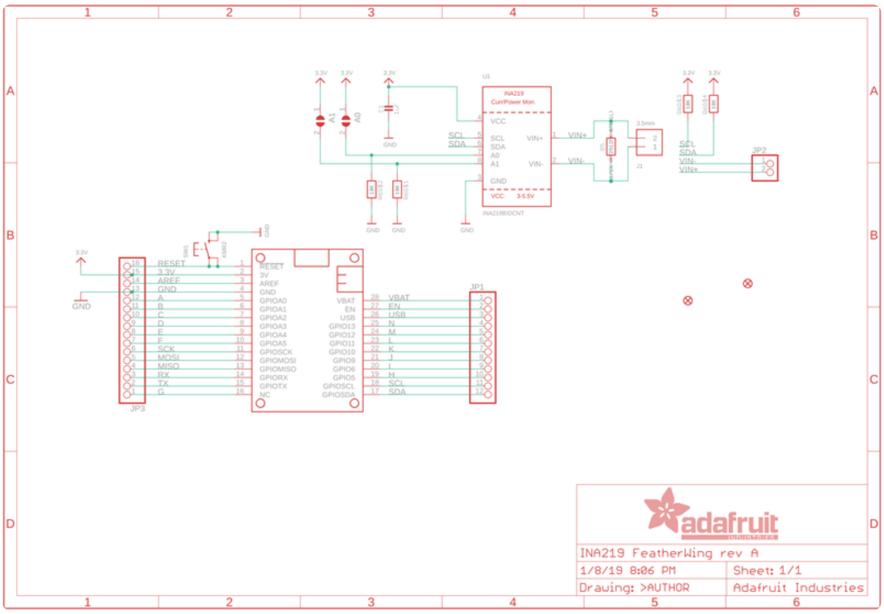


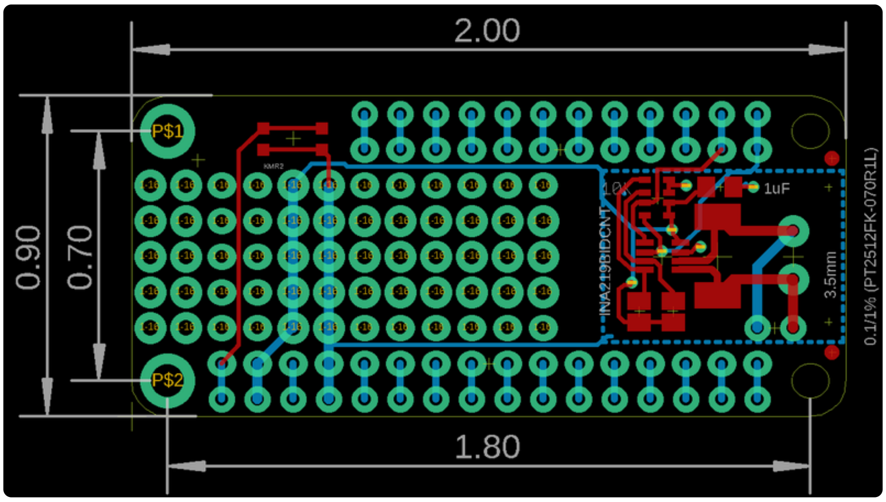
Breakout Schematic & Fabrication Print Original Version

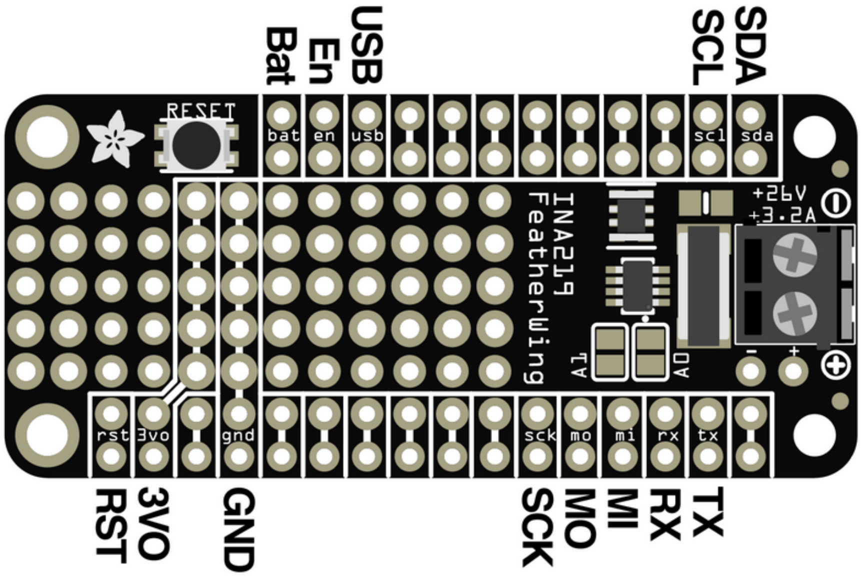




FeatherWing Schematic, Fabrication Print and Pinout






SDA
SCL
USB
En
Bat
RESET
bat
en
usb
scl
sda
INA219
FeatherWing
+26V
+3.2A
A1
A0
rst
3vo
gnd
sck
mo
mi
rx
tx
RST
3VO
GND
SCK
MO
MI
RX
TX