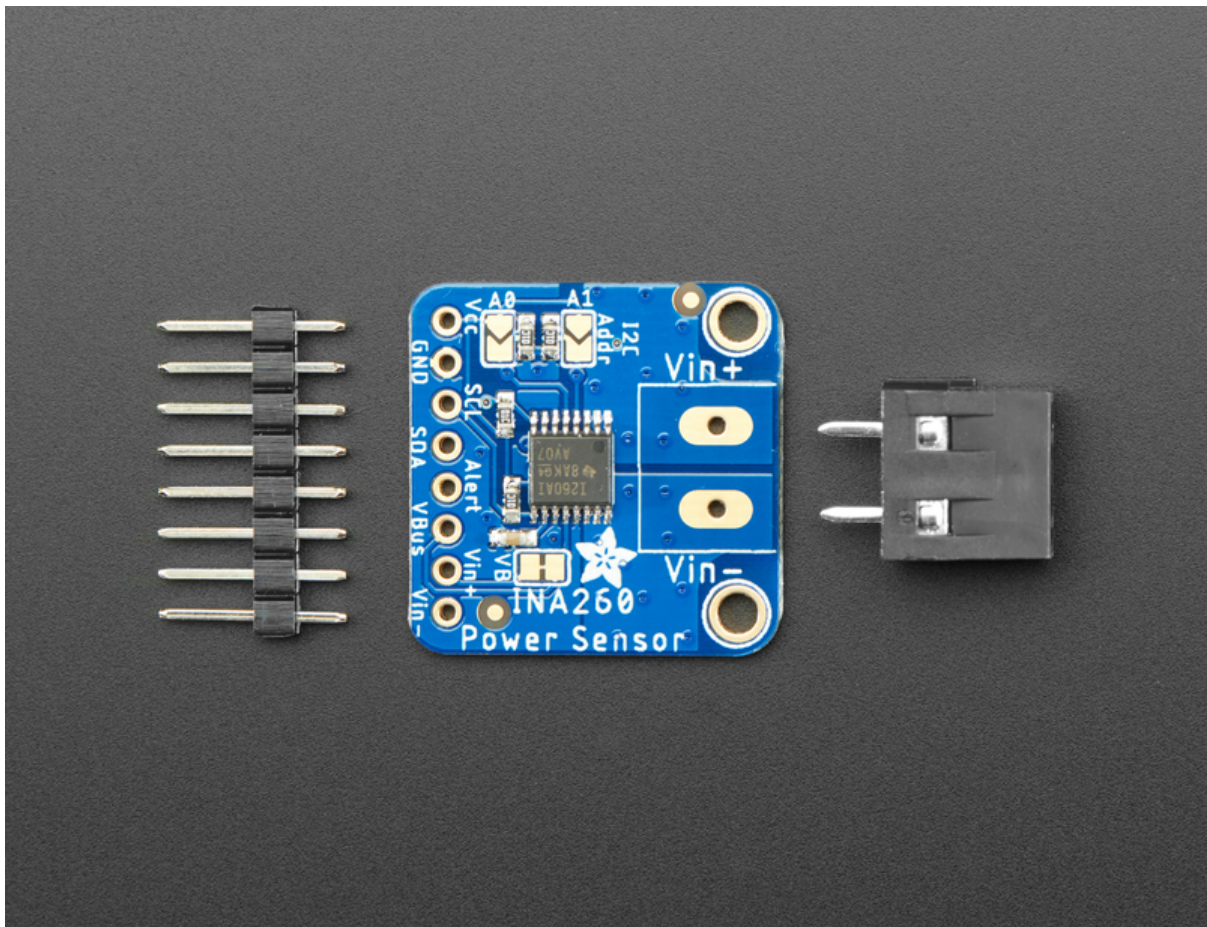




# Adafruit INA260 Current + Voltage + Power Sensor Breakout

Created by Bryan Siepert



<https://learn.adafruit.com/adafruit-ina260-current-voltage-power-sensor-breakout>

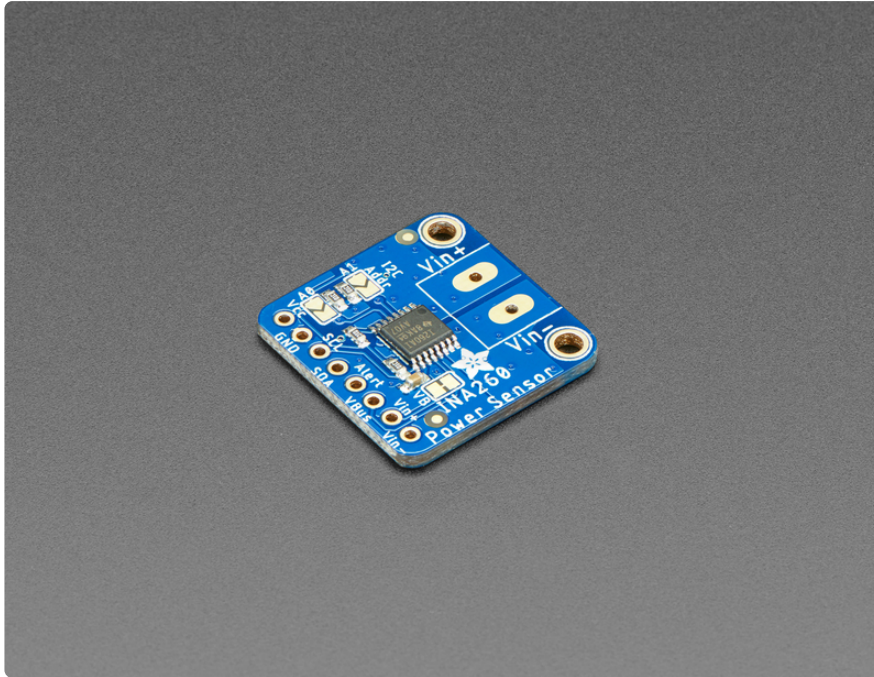
Last updated on 2024-06-03 02:43:41 PM EDT

# Table of Contents

<b>Overview</b>	<b>3</b>
<ul style="list-style-type: none"><li>• <a href="#">Choose your side</a></li><li>• <a href="#">How does it work?</a></li></ul>	
<b>Pinouts</b>	<b>5</b>
<ul style="list-style-type: none"><li>• <a href="#">Power Pins</a></li><li>• <a href="#">I2C Logic Pins</a></li><li>• <a href="#">Other Pins</a></li></ul>	
<b>Assembly</b>	<b>7</b>
<ul style="list-style-type: none"><li>• <a href="#">Prepare the header strip:</a></li><li>• <a href="#">Add the breakout board:</a></li><li>• <a href="#">And Solder!</a></li><li>• <a href="#">Prepare the terminal block:</a></li><li>• <a href="#">Solder again!</a></li></ul>	
<b>Arduino</b>	<b>13</b>
<ul style="list-style-type: none"><li>• <a href="#">Wiring</a></li><li>• <a href="#">Install Adafruit_INA260 library</a></li><li>• <a href="#">Example Code</a></li><li>• <a href="#">Load Demo</a></li><li>• <a href="#">Arduino Usage</a></li></ul>	
<b>Python &amp; CircuitPython</b>	<b>17</b>
<ul style="list-style-type: none"><li>• <a href="#">CircuitPython Microcontroller Wiring</a></li><li>• <a href="#">Python Computer Wiring</a></li><li>• <a href="#">CircuitPython Installation of INA260 Library</a></li><li>• <a href="#">Python Installation of INA260 Library</a></li><li>• <a href="#">CircuitPython &amp; Python Usage</a></li><li>• <a href="#">Full Example Code</a></li></ul>	
<b>Python Docs</b>	<b>21</b>
<b>Downloads</b>	<b>21</b>
<ul style="list-style-type: none"><li>• <a href="#">Datasheets</a></li><li>• <a href="#">Schematic</a></li><li>• <a href="#">Fabrication Print</a></li></ul>	
<b>Arduino Docs</b>	<b>23</b>

---

# Overview

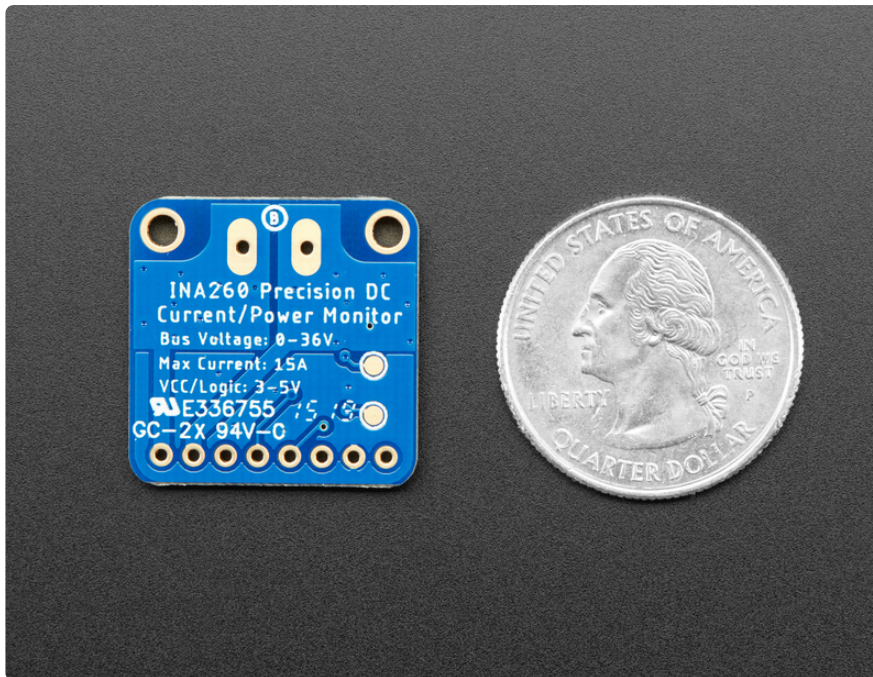


This breakout board may well be the last current sensing solution you every need to buy. Not only can it do the work of two multimeters, but it can do it with amazing precision and flexibility. With it you can measure high or low side DC current, the bus voltage, and have it automatically calculate the power. It can do so over impressive voltage, current, and temperature ranges with better than 1% accuracy, all while delivering the data in an easy to use format over I2C.

Works great with any microcontroller that is CircuitPython or Arduino compatible as well as single board computers such as the Raspberry Pi. It is compatible with 3V or 5V logic and can measure bus voltages up to +36VDC. Not for use with AC voltages.

## Choose your side

Most current-measuring devices operate with some notable constraints that limit what they can be used for. Many are low-side only which can cause issues as the ground reference changes with current. Others like it's little sister the INA219B avoid this by measuring on the high side but need to change their shunt resistor to measure different current ranges. The INA260 avoids these limitations, and with it's integrated precision shut resistor it can be used to measure as much as **+36V** at up to **15A Continuous** on either the high or low side. Wow!



## How does it work?

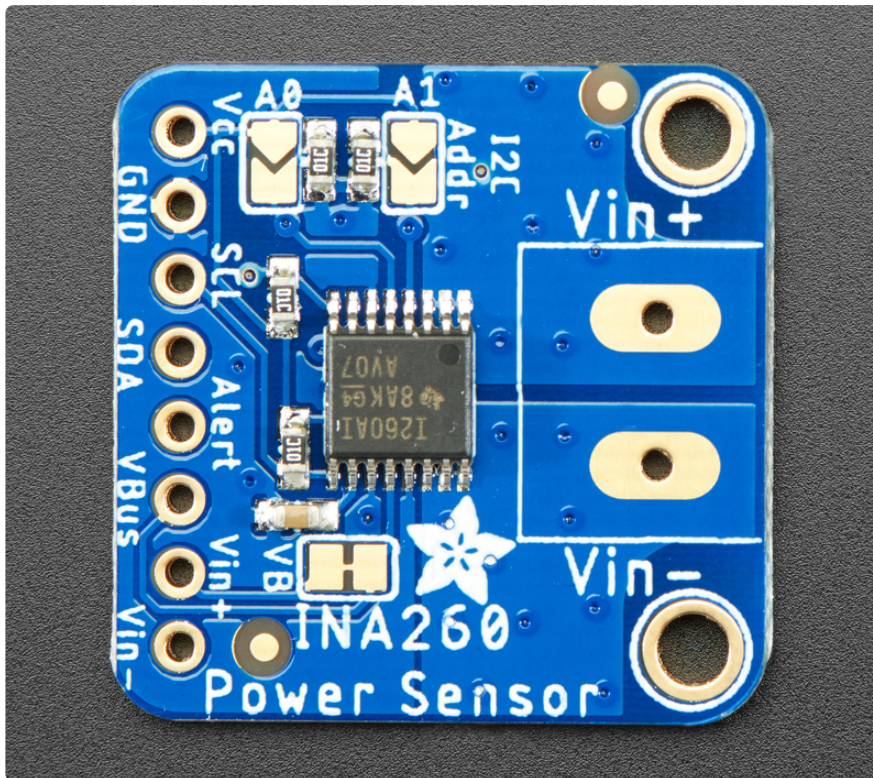
The voltage across the integrated 2 milliohm (.002 ohms!), 0.1% shunt resistor is measured by the internal 16 bit ADC, allowing for measurements over the impressive current range with a resolution of 1.5 mA. (The resistance of the resistor is so low that some multimeters will register it as a short!)

In a high side configuration the bus voltage measurement and power calculation can be retrieved accurately, however advanced hackers wanting to measure bus voltage in a low side configuration will need to cut the jumper connecting **V+** to **VBUS** and connect the **VBUS** pin to the voltage bus.

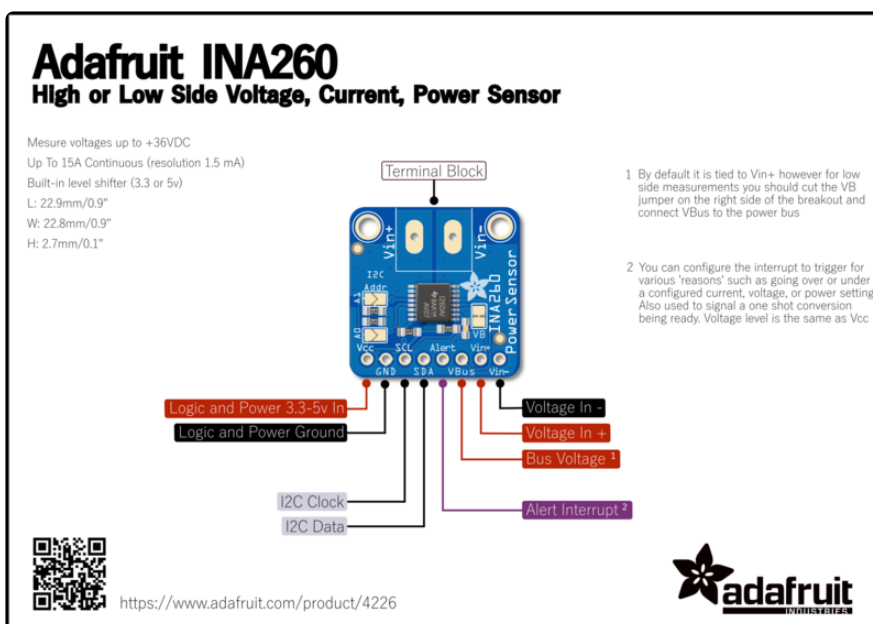
Note that when switching inductive loads, the instantaneous voltage levels may greatly exceed steady-state levels due to inductive kickback. Chip damage can occur if you do not take precautions to protect against inductive spikes. See <https://learn.digilentinc.com/Documents/390> for more information

Comes as a fully assembled breakout board with a 5.08mm terminal block (extra chunky!) and header. Some light soldering is required to attach the header and terminal block.

# Pinouts



The little chip in the middle of the PCB is the actual INA260 sensor that does all the current and voltage sensing. We add all the extra components you need to get started, and 'break out' all the other pins you may want to connect to onto the PCB. For more details you can check out the schematics in the Downloads page.



# 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.

- **Vcc** - 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
- **GND** - common ground for power and logic

# I2C Logic Pins

- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line. The logic level is the same as **Vcc** and it has a 10K pullup already on it.
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line. The logic level is the same as **Vcc**. and it has a 10K pullup already on it.

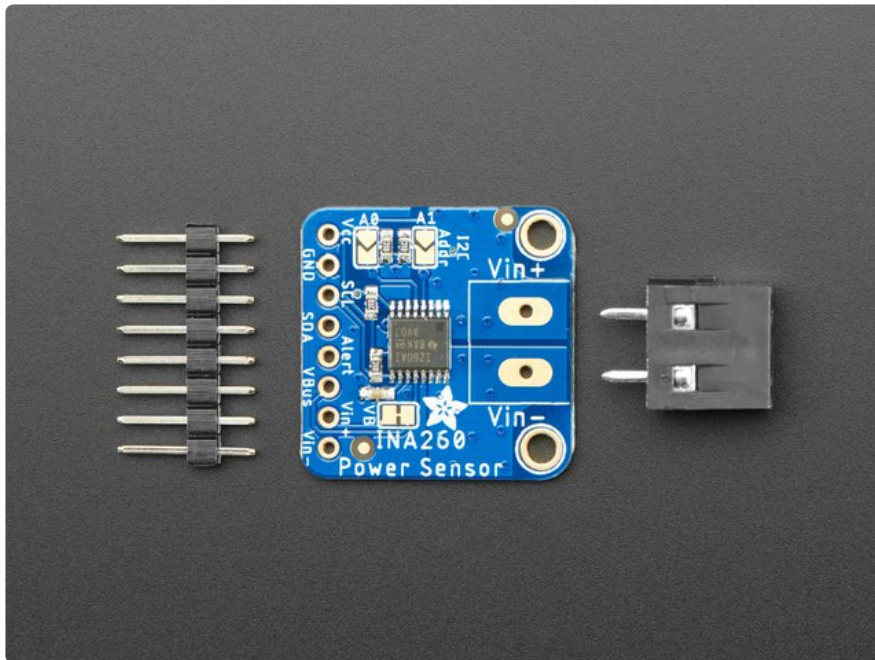
# Other Pins

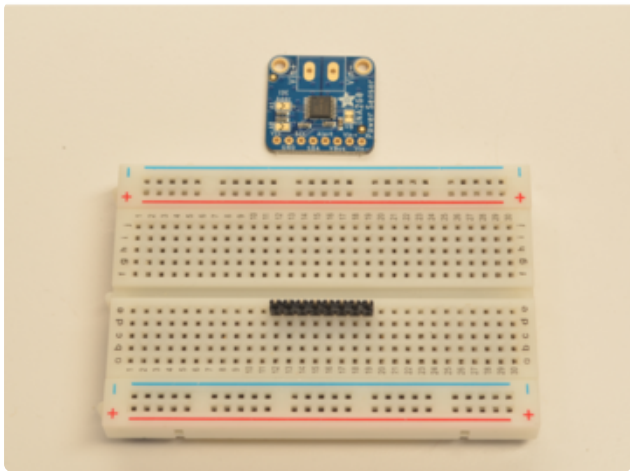
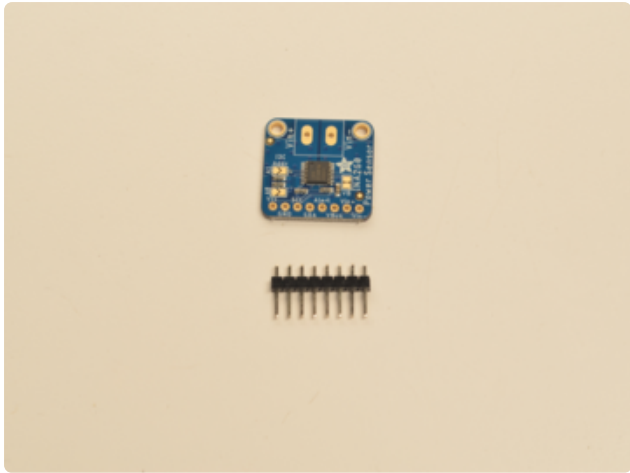
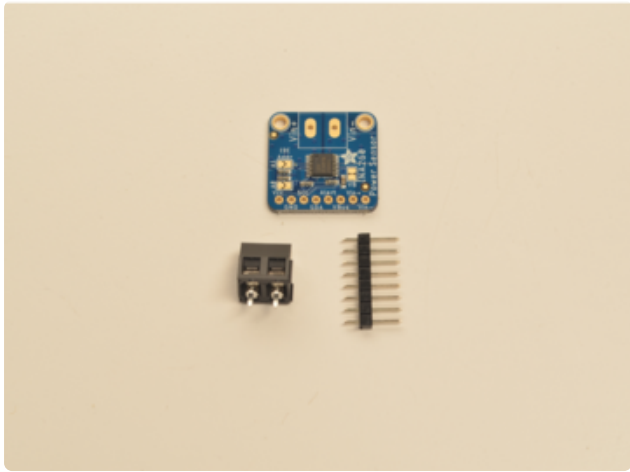
- **Vin+** is the positive input pin. Connect to supply for high side current sensing or to load ground for low side sensing.
- **Vin-** is the negative input pin. Connect to load for high side current sensing or to board ground for low side sensing
- **Alert** is the interrupt output pin. You can configure the interrupt to trigger for various 'reasons' such as going over or under a configured current, voltage, or power setting. Also used to signal a one shot conversion being ready. Voltage level is the same as **Vcc**.
- **VBus** is the bus voltage. By default it is tied to **Vin+** however for low side measurements you should cut the **VB** jumper on the right side of the breakout and connect **VBus** to the power bus so it can accurately calculate the bus voltage and total power ( $V_{Bus} * Current$ ) draw.
- **A0** and **A1** solder jumpers - These can be bridged with solder to pull the address pin up to **VCC** to change the I2C address according to the table below

A1	A0	I2C Address (hex)
GND	GND	<b>0x40</b> (Default)

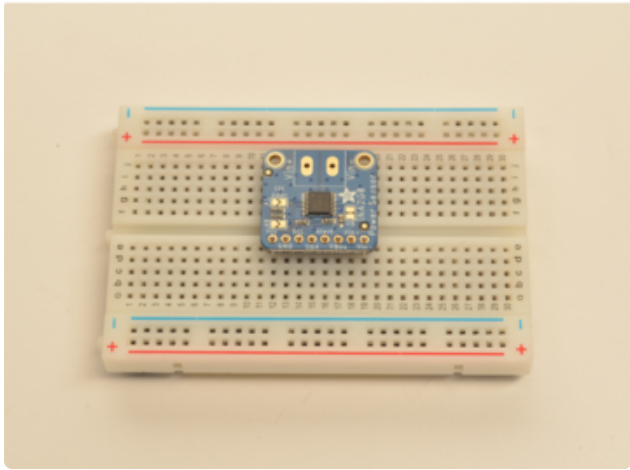
GND	VCC	0x41
VCC	GND	0x44
VCC	VCC	0x45

## Assembly



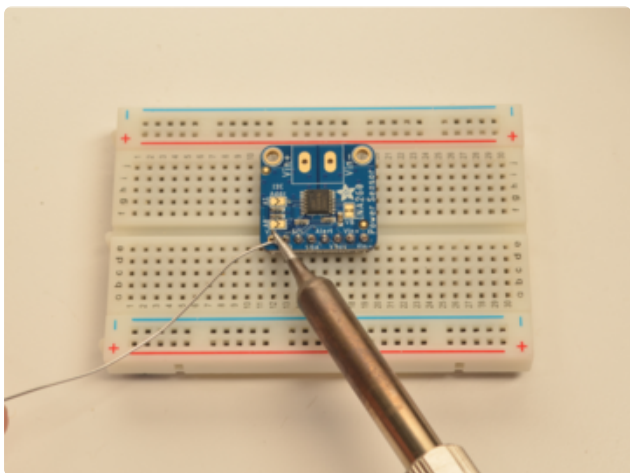
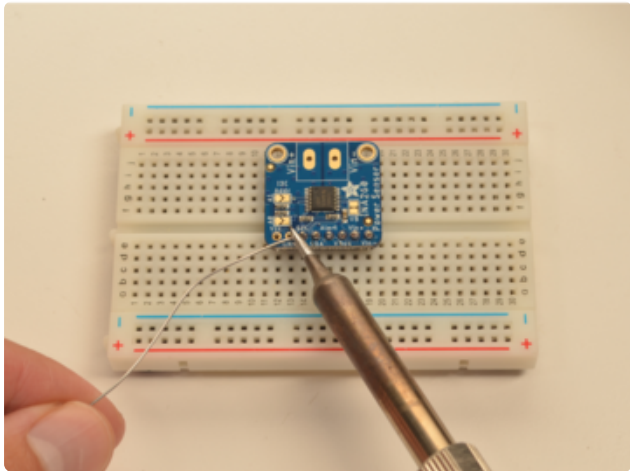
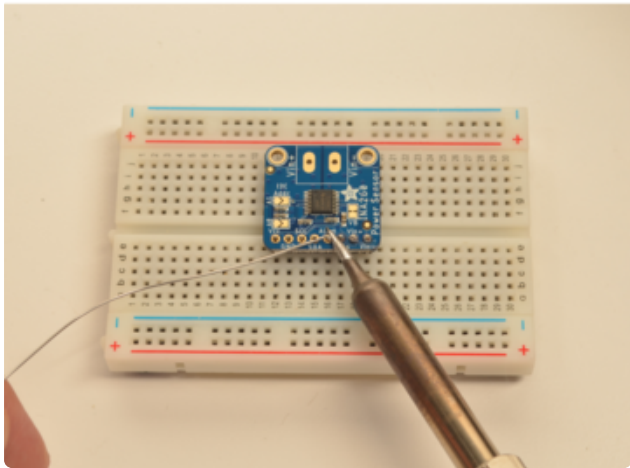
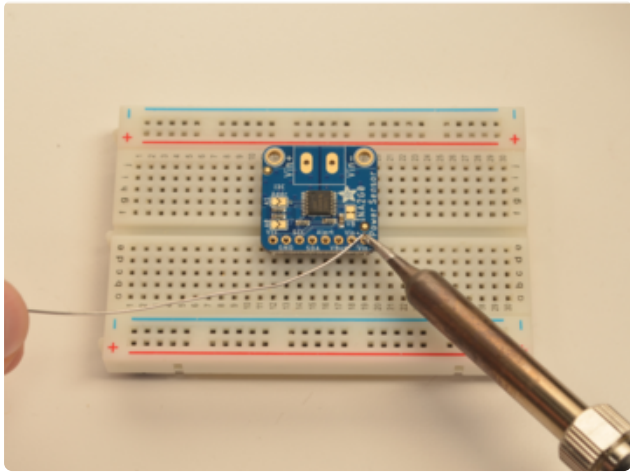


**Prepare the header strip:**  
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**.



## Add the breakout board:

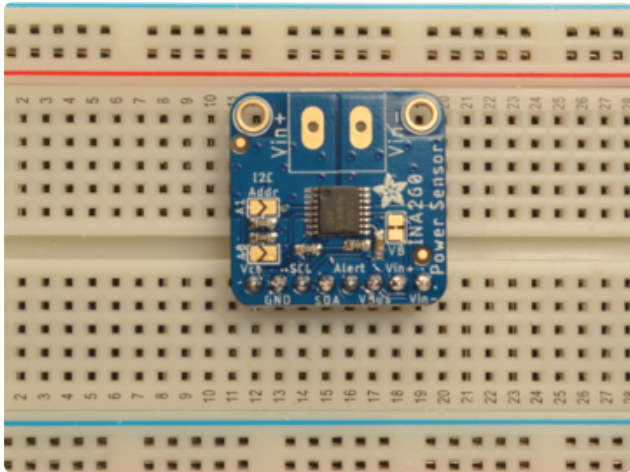
Place the breakout board over the pins so that the short pins poke through the breakout pads.



## And Solder!

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

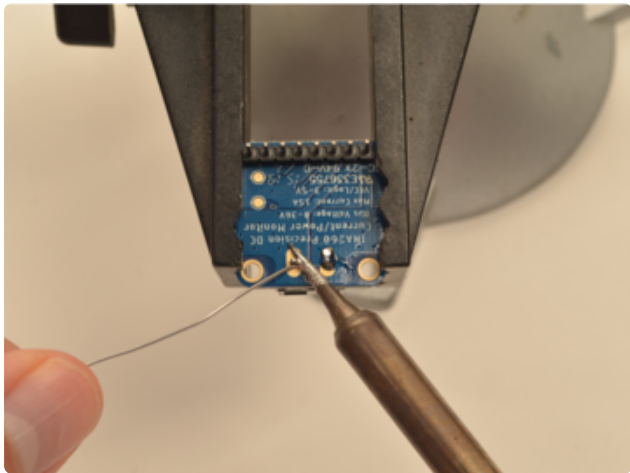
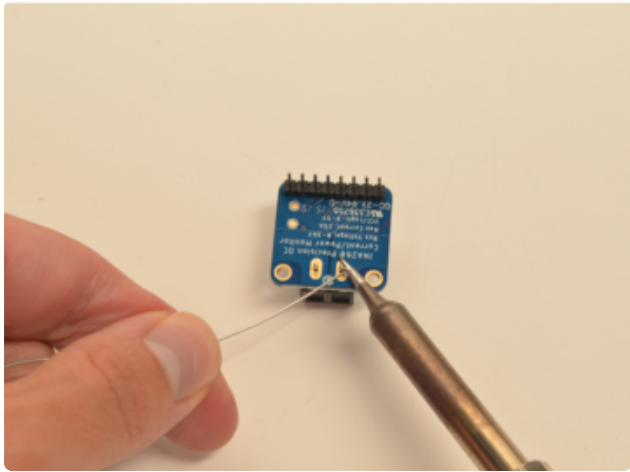
(For tips on soldering, be sure to check out our [Guide to Excellent Soldering \(https://adafru.it/aTk\)](https://adafru.it/aTk)).



You're done with the header strip! Check your solder joints visually.

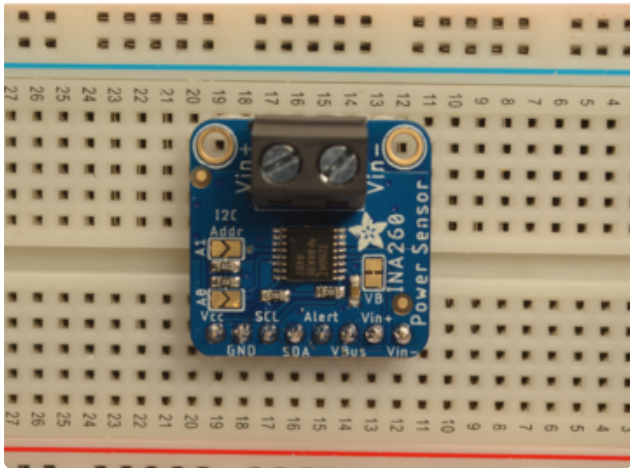
## Prepare the terminal block:

Place the terminal block inside the board. Make sure it's facing out! Turn the board over so the terminal pins are facing upwards.



## Solder again!

First solder one pin by using the soldering iron to prop up the board. Then use a [vice](http://adafru.it/151) or helper hands to hold the board in place while soldering the other pin.



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

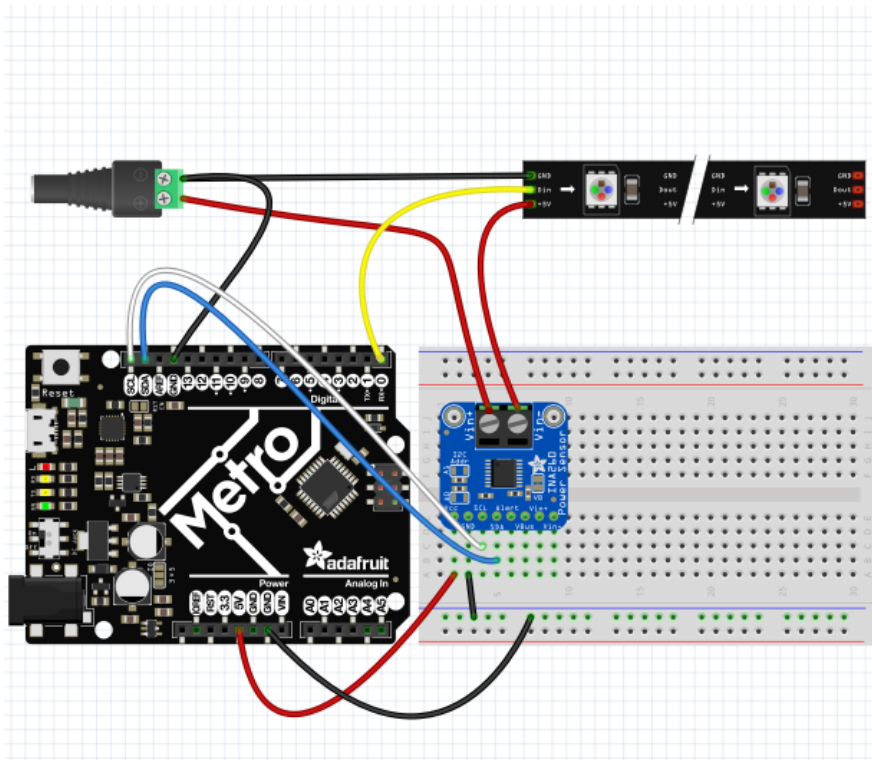
---

## Arduino

### Wiring

Connecting the INA260 to your Feather or Arduino is easy:

- If you are running a Feather (3.3V), connect **Feather 3V** to board **VIN**
- If you are running a 5V Arduino (Uno, etc.), connect **Arduino 5V** to board **VIN**
- Connect **Feather or Arduino GND** to board **GND**
- Connect **Feather or Arduino SCL** to board **SCL**
- Connect **Feather or Arduino SDA** to board **SDA**
- Connect **Vin+** to supply for high side current sensing or to load ground for low side sensing.
- Connect **Vin-** to load for high side current sensing or to board ground for low side sensing

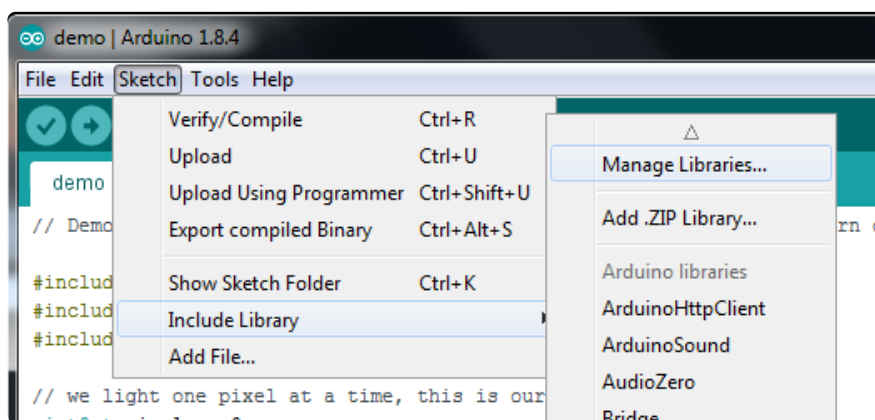


The final results should resemble the illustration above, showing an Adafruit Metro development board.

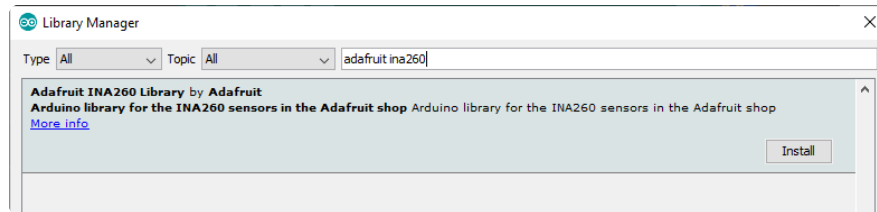
## Install Adafruit\_INA260 library

To begin reading sensor data, you will need to [install the Adafruit\\_INA260 library \(code on our github repository\)](https://adafru.it/EGH) (<https://adafru.it/EGH>). It is available from the Arduino library manager so we recommend using that.

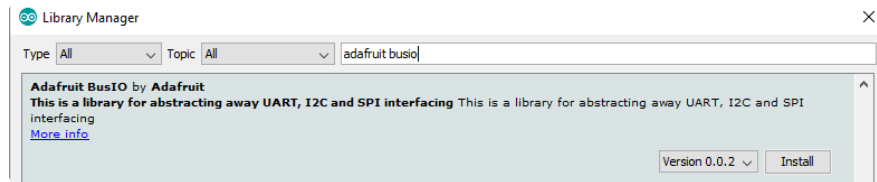
From the IDE open up the library manager...



Click the **Manage Libraries ...** menu item, search for **Adafruit INA260**, and select the **Adafruit INA260** library and click **Install**:



Then follow the same process for the **Adafruit BusIO** library.



## Example Code

The following example code is part of the standard library, but illustrates how you can retrieve sensor data from the INA260 for the Current, Voltage, and Power.

## Load Demo

Open up **File->Examples->Adafruit\_INA260 Library->ina260\_test** and upload to your Arduino wired up to the sensor.

Additionally you will want to add code to turn the neopixel strip on so that there is some current to measure! If you're not familiar with using the NeoPixel library, please consult the [excellent NeoPixel Überguide page on the subject](https://adafru.it/nBF). (<https://adafru.it/nBF>)

Upload the sketch to your board and open up the Serial Monitor (**Tools->Serial Monitor**). You should see the the values for Current, Voltage, and Power.

```
#include <Adafruit_INA260.h>

Adafruit_INA260 ina260 = Adafruit_INA260();

void setup() {
  Serial.begin(115200);
  // Wait until serial port is opened
  while (!Serial) { delay(10); }

  Serial.println("Adafruit INA260 Test");

  if (!ina260.begin()) {
    Serial.println("Couldn't find INA260 chip");
    while (1);
  }
  Serial.println("Found INA260 chip");
}
```

```

void loop() {
  Serial.print("Current: ");
  Serial.print(ina260.readCurrent());
  Serial.println(" mA");

  Serial.print("Bus Voltage: ");
  Serial.print(ina260.readBusVoltage());
  Serial.println(" mV");

  Serial.print("Power: ");
  Serial.print(ina260.readPower());
  Serial.println(" mW");

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

```

You should get something resembling the following output when you open the Serial Monitor at 115200 baud:

```

COM9
Adafruit INA260 Test
Found INA260 chip
Current: 148.75 mA
Bus Voltage: 4962.50 mV
Power: 820.00 mW

Current: 170.00 mA
Bus Voltage: 4980.00 mV
Power: 790.00 mW

Current: 167.50 mA
Bus Voltage: 4975.00 mV
Power: 830.00 mW

```

## Arduino Usage

Here we'll explain the different calls to the INA260 library and their use

First we include the library and create an Adafruit\_INA260 object to use in the rest of the sketch.

```

#include <Adafruit_INA260.h>

Adafruit_INA260 ina260 = Adafruit_INA260();

```

Next, in the setup we call the INA260 object's `begin` function to initialize the driver and prepare it to read measurements from the sensor.

The `begin` function will return false if it is unable to make a connection to an INA260 sensor. If this happens, double check your power and I2C wiring.

```

if (!ina260.begin()) {
  Serial.println("Couldn't find INA260 chip");
}

```

```
    while (1);  
}
```

Finally we can take some readings! `readCurrent`, `readBusVoltage`, and `readPower` all read and return the given measurements in milliamps, millivolts, and milliwatts respectively.

```
Serial.print("Current: ");  
Serial.print(ina260.readCurrent());  
Serial.println(" mA");  
  
Serial.print("Bus Voltage: ");  
Serial.print(ina260.readBusVoltage());  
Serial.println(" mV");  
  
Serial.print("Power: ");  
Serial.print(ina260.readPower());  
Serial.println(" mW");
```

By default power and voltage measurements are only accurate for high side current sensing

To get accurate measurements when calling `readVoltage` and `readPower` (for low-side sensing), you will need to cut the **VB** jumper on the right side of the breakout and connect the **VBus** pin to your bus.

---

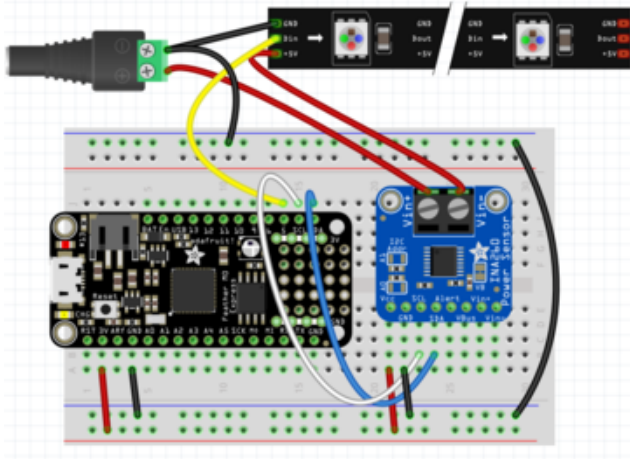
## Python & CircuitPython

It's easy to use the INA260 sensor with Python and CircuitPython, and the [Adafruit CircuitPython INA260 \(https://adafru.it/EGK\)](https://adafru.it/EGK) module. This module allows you to easily write Python code that reads the current, voltage and power 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 \(https://adafru.it/BSN\)](https://adafru.it/BSN).

## CircuitPython Microcontroller Wiring

First wire up a INA260 to your board exactly as follows. Here is an example of the INA260 wired to a Feather using I2C:



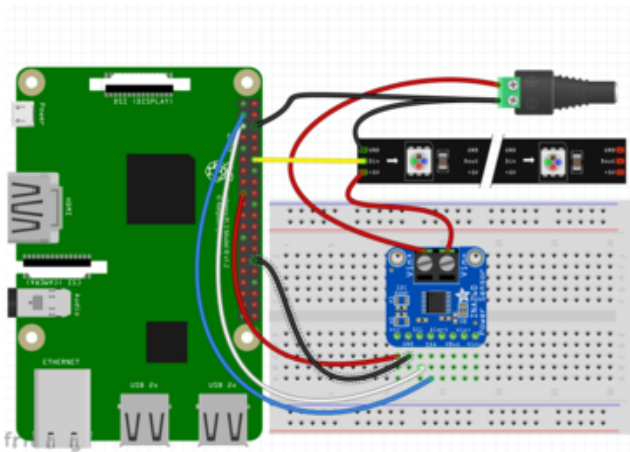
- Board 3V to sensor Vin
- Board GND to sensor GND
- Board SCL to sensor SCL
- Board SDA to sensor SDA
- Sensor Vin+ to supply for high side current sensing or to load ground for low side sensing.
- Sensor Vin- to load for high side current sensing or to board ground for low side sensing

Note: This breakout includes pullup resistors on the I2C lines, no external pullups are required.

## 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>).

Here's the Raspberry Pi wired with I2C:



- Pi 3V3 to sensor VIN
- Pi GND to sensor GND
- Pi SCL to sensor SCL
- Pi SDA to sensor SDA

## CircuitPython Installation of INA260 Library

You'll need to install the [Adafruit CircuitPython INA260](https://adafru.it/EGK) (<https://adafru.it/EGK>) library on your CircuitPython board.

First make sure you are running the [latest version of Adafruit CircuitPython \(https://adafru.it/Amd\)](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 \(https://adafru.it/uap\)](https://adafru.it/uap). Our CircuitPython starter guide has [a great page on how to install the library bundle \(https://adafru.it/ABU\)](https://adafru.it/ABU).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- `adafruit_ina260.mpy`
- `adafruit_bus_device`
- `adafruit_register`

Before continuing make sure your board's lib folder or root filesystem has the `adafruit_ina260.mpy`, `adafruit_bus_device`, and `adafruit_register` files and folders copied over.

Next [connect to the board's serial REPL \(https://adafru.it/Awz\)](https://adafru.it/Awz) so you are at the CircuitPython `>>>` prompt.

## Python Installation of INA260 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 \(https://adafru.it/BSN\)](https://adafru.it/BSN)!

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

- `sudo pip3 install adafruit-circuitpython-ina260`

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 voltage 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_ina260

i2c = busio.I2C(board.SCL, board.SDA)
ina260 = adafruit_ina260.INA260(i2c)
```

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

- **current** - The current in milliamps.
- **voltage** - The voltage in volts.
- **power** - The power in milliwatts

For example to print current, voltage, and power levels:

```
print("Current:", ina260.current)
print("Voltage:", ina260.voltage)
print("Power:", ina260.power)
```

```
>>> print("Current:", ina260.current)
Current: 198.75
>>> print("Voltage:", ina260.voltage)
Voltage: 3.23875
>>> print("Power:", ina260.power)
Power: 640
```

For more details, check out the [library documentation \(https://adafru.it/EGM\)](https://adafru.it/EGM).

That's all there is to using the INA260 sensor with CircuitPython!

## Full Example Code

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

import time
import board
import adafruit_ina260
```

```
i2c = board.I2C() # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C() # For using the built-in STEMMMA QT connector on a
microcontroller
ina260 = adafruit_ina260.INA260(i2c)
while True:
    print(
        "Current: %.2f mA Voltage: %.2f V Power: %.2f mW"
        % (ina260.current, ina260.voltage, ina260.power)
    )
    time.sleep(1)
```

By default power and voltage measurements are only accurate for high side current sensing

To get accurate measurements when calling `readVoltage` and `readPower` for low side sensing you will need to cut the **VB** jumper on the right side of the breakout and connect the **VBus** pin to your bus.

---

## Python Docs

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

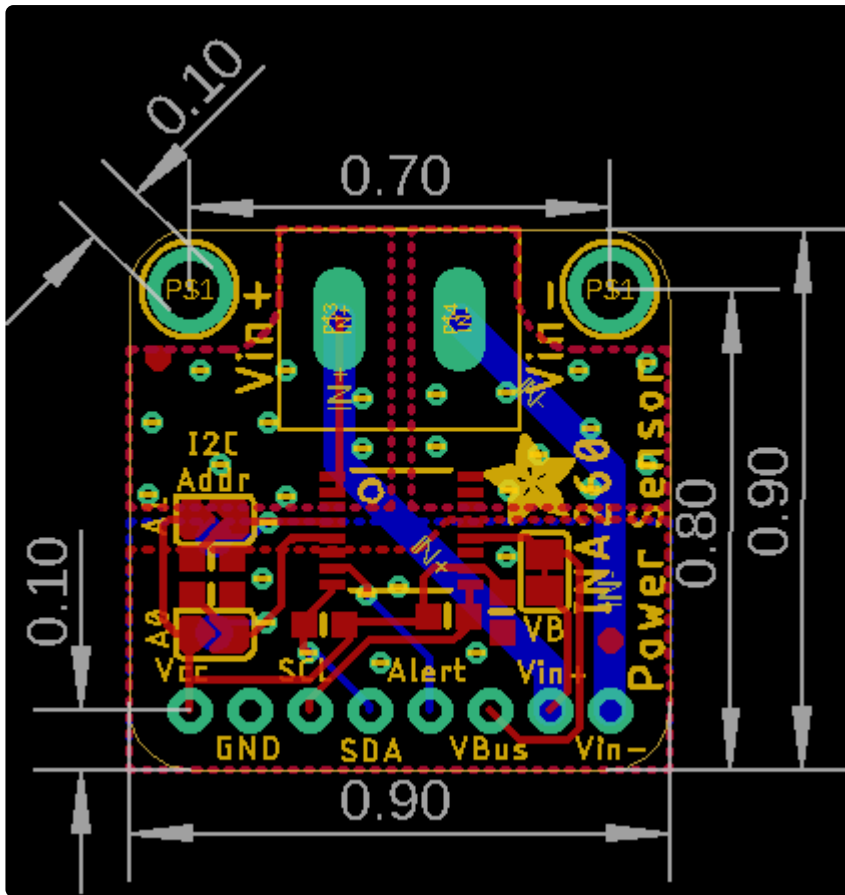
---

## Downloads

## Datasheets

- [INA260 datasheet \(https://adafru.it/EGN\)](https://adafru.it/EGN)
- [TI design resources \(https://adafru.it/EGO\)](https://adafru.it/EGO)
- [Fritzing object available in the Adafruit Fritzing Library \(https://adafru.it/aP3\)](https://adafru.it/aP3)
- [EagleCAD PCB files on GitHub \(https://adafru.it/EGP\)](https://adafru.it/EGP)





## Arduino Docs

[Arduino Docs \(https://adafru.it/Esz\)](https://adafru.it/Esz)