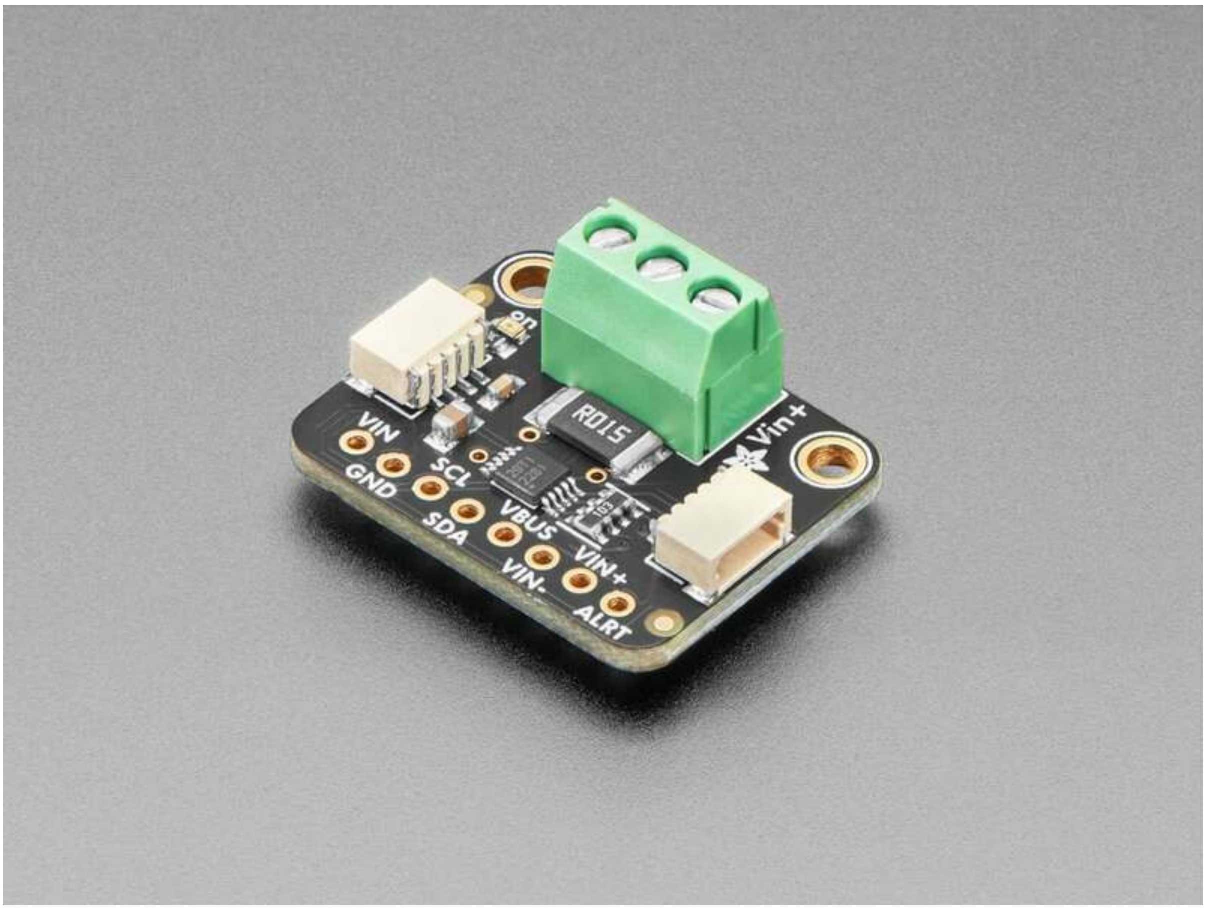
Adafruit INA228 I2C Power Monitor

Created by Liz Clark

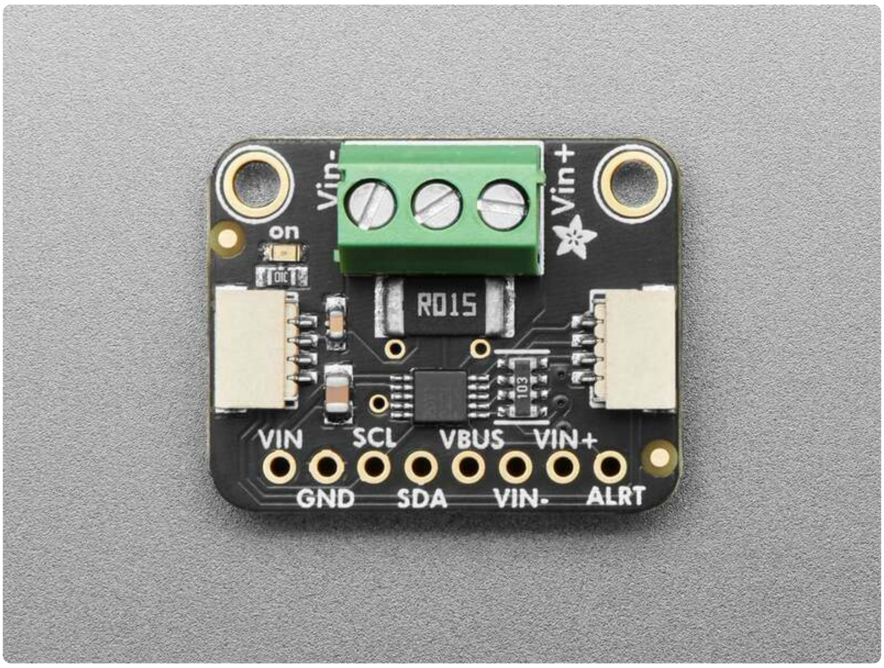


https://learn.adafruit.com/adafruit-ina228-i2c-power-monitor

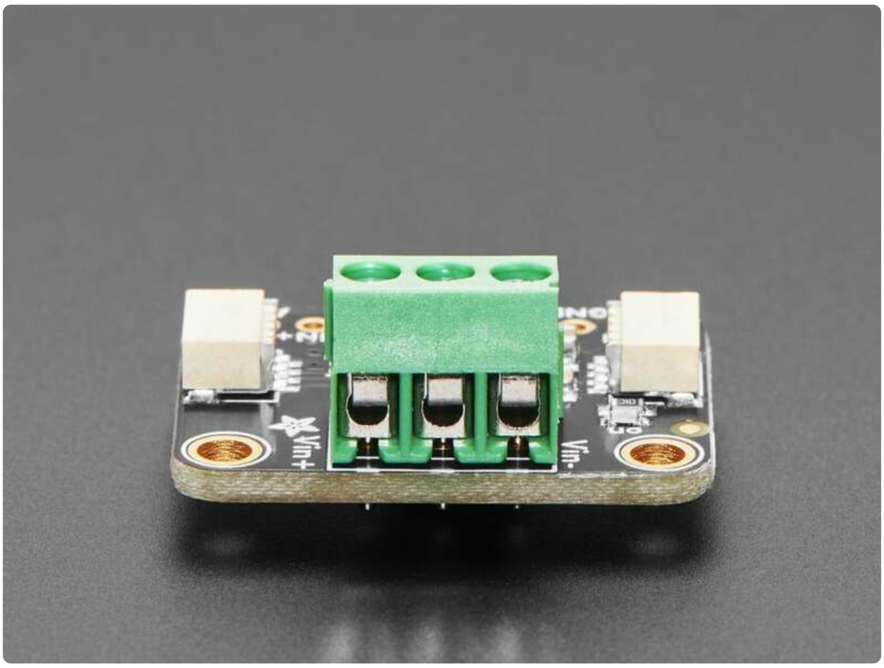
Last updated on 2025-02-10 03:31:22 PM EST

Table of Contents

Overview

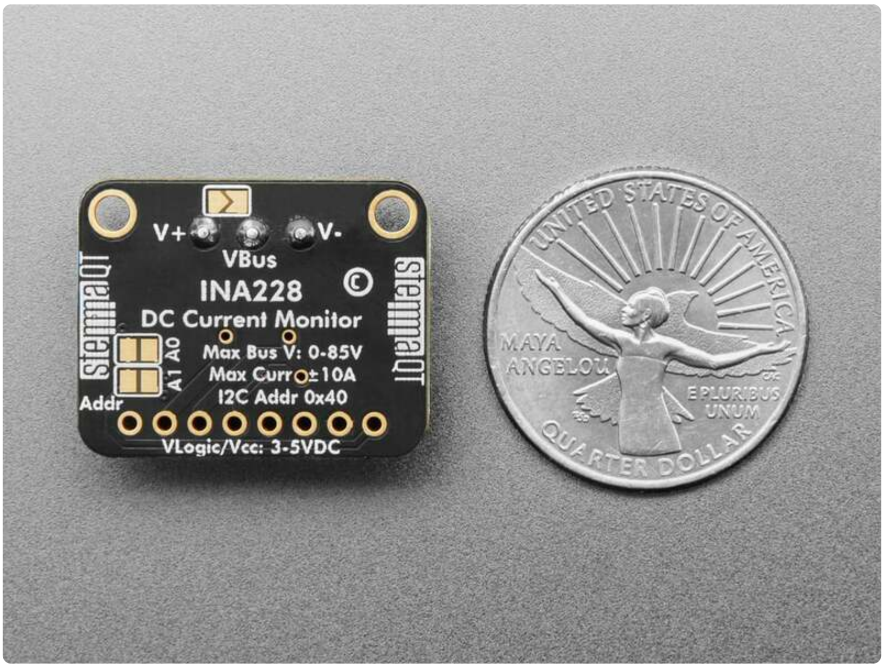


The INA228 is an amazing power monitoring chip, with best-of-everything support: up to 85VDC common-mode, high or low side measurements, 20-bit (!) ADC for precision measurements from milliamp to Amp, and an I2C interface for easy configuration of alerts, oversampling, gain adjustments and more!

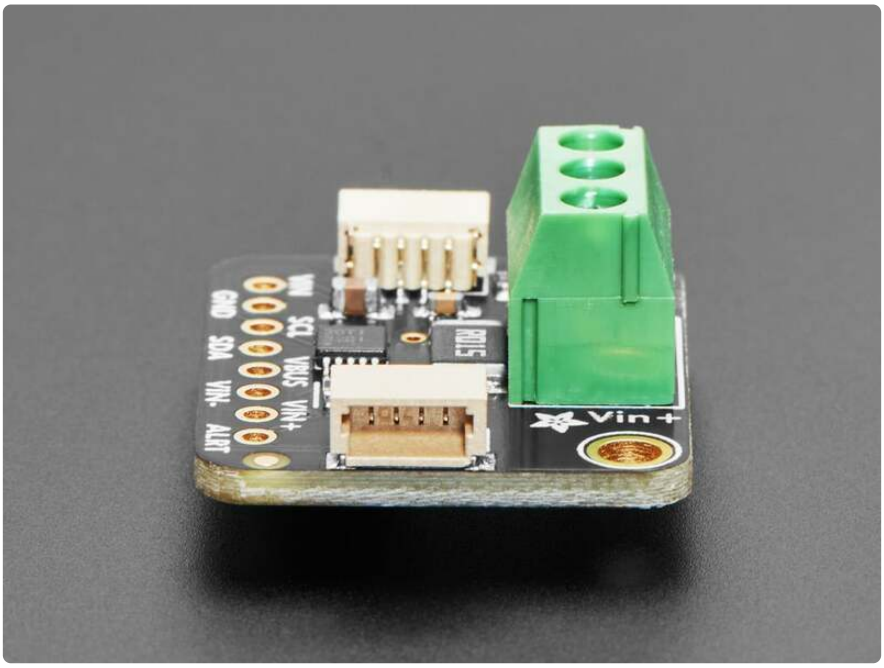


This breakout board may well be the last current sensing solution you ever need to buy. Not only can it do the work of two multimeters, but it can do it with amazing precision and flexibility. With it you can measure high or low side DC current, the bus voltage, and have it automatically calculate the 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.

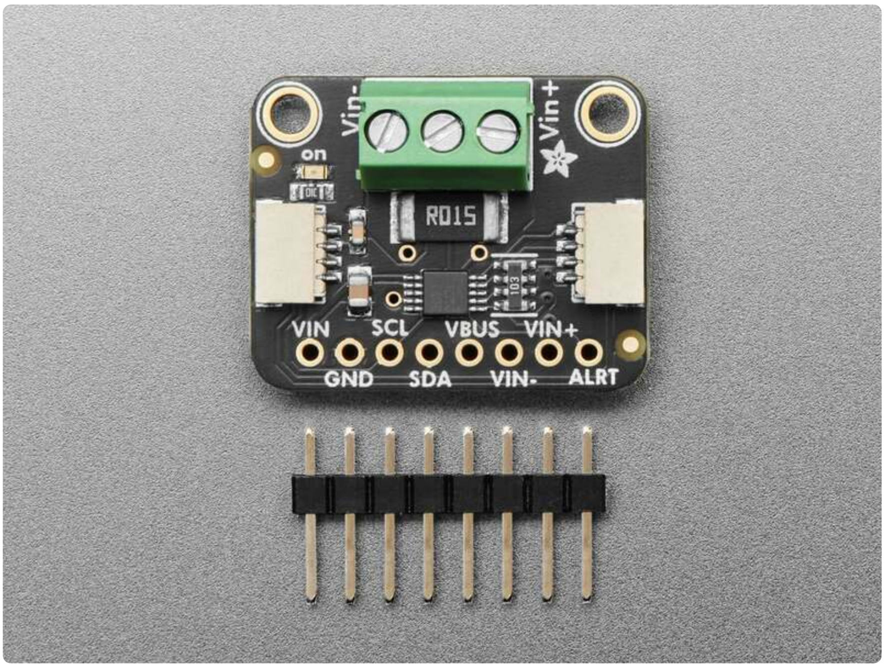
Works great with any microcontroller that is CircuitPython or Arduino compatible, as well as single board computers such as the Raspberry Pi. It is compatible with 3V or 5V logic and can measure bus voltages up to +85VDC. **Note that it is not for use with AC voltages**.



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 its 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 INA228 avoids these limitations, and with the precision 15 milliohm shunt resistor on board, it can be used to measure as much as **+85V** at up to **10A** (~10uA per LSB) or **2.75A** (~2.5uA per LSB) **Continuous** on either the high or low side. Wow!



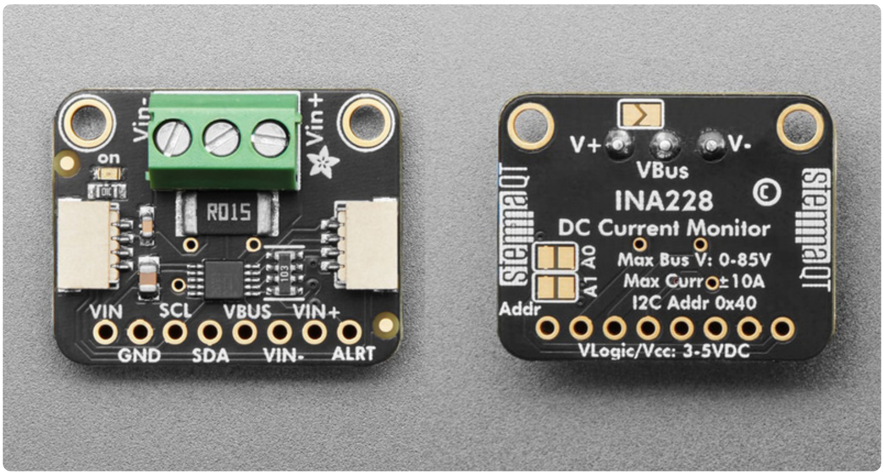
The voltage across the integrated 15 milliohm (.015 ohms!), 0.1% shunt resistor is measured by the internal 20 bit ADC, allowing for measurements over the impressive current range with a resolution of 10uA per LSB in high current measurement mode or 2.5uA per LSB in low current measurement mode.



To measure low-side, connect **VIN-** to ground and **VIN+** to your load's lowest potential. **VBUS** should connect to the highest project voltage, up to 85V. To measure high-side, connect **VIN+** to **VBUS** to the highest project voltage, and **VIN-** to the load's highest potential. In high-side measurement, which is most common, you can simplify connecting **V+** to **VBUS** by soldering closed the back jumper.

This comes as a fully assembled breakout board with a 3.5mm terminal block and header. Some light soldering is required to attach the header for use in a breadboard.

Pinouts



The default I2C address is **0x40**.

Power Pins

- **VIN** - this is the power pin. It can be powered by 3V or 5V. 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 microcontroller's I2C clock line. This pin can use 3-5V logic, and there's a **10K pullup** on this pin.
- **SDA** - I2C data pin, connect to your microcontroller's I2C data line. This pin can use 3-5V logic, and 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** (Qwiic) connectors or to other things with various associated accessories (https://adafru.it/Ft6).

Input Pins

The pins are in a low side measurement configuration by default. They are available along the bottom edge of the board and the terminal block on the top edge of the board.

- **VIN+** - Positive input pin.
- **VIN-** - Negative input pin.

- **VBUS** - Bus voltage input pin.

To measure low-side, Connect **VIN-** to ground and **VIN+** to your load's lowest potential. **VBUS** should connect to the highest project voltage, up to 85V.

To measure high-side, connect **VIN+** to **VBUS** to the highest project voltage, and **VIN-** to the load's highest potential.

VBUS Jumper

On the back of the board, above the **VIN+** and **VBUS** pins, is the **VBUS jumper**. This jumper connects or disconnects **VBUS** from **VIN+**. When the jumper is open (default), it configures the board for low side measurement. When the jumper is soldered closed, it configures the board for high side measurement.

Interrupt Pin

- **ALRT** - The alert/interrupt pin. The default state is active low.

Power LED

- **Power LED -** In the upper left corner, above the STEMMA connector, on the front of the board, is the power LED, labeled **on**. It is the green LED.

Address Jumpers

On the back of the board are **two address jumpers**, labeled **A0** and **A1**, above the **Addr** label on the board silk. These jumpers allow you to chain up to 4 of these boards on the same pair of I2C clock and data pins. To do so, you solder the jumpers "closed" by connecting the two pads.

The default I2C address is **0x40**. The other address options can be calculated by “adding” the **A0/A1** to the base of **0x40**.

A0 sets the lowest bit with a value of **1** and **A1** sets the next bit with a value of **2**. The final address is **0x40** + **A1** + **A0** which would be **0x43**.

So for example if **A1** is soldered closed and **A0** is soldered closed, the address is **0x40** + **2** + **1** = **0x43**.

If only **A0** is soldered closed, the address is **0x40** + **1** = **0x41**

If only **A1** is soldered closed, the address is **0x40** + **2** = **0x42**

The table below shows all possible addresses, and whether the pin(s) should be high (closed) or low (open).

ADDR	A0	A1
0x40	L	L
0x41	H	L
0x42	L	H
0x43	H	H

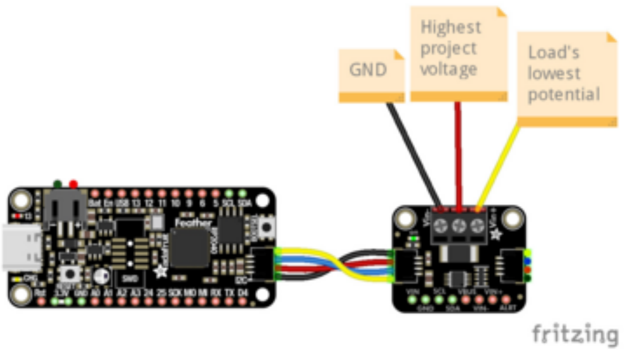
CircuitPython and Python

It's easy to use the **INA228** with Python or CircuitPython, and the Adafruit_CircuitPython_INA228 (https://adafru.it/1ae7) module. This module allows you to easily write Python code to monitor high or low side power measurements.

You can use this driver 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).

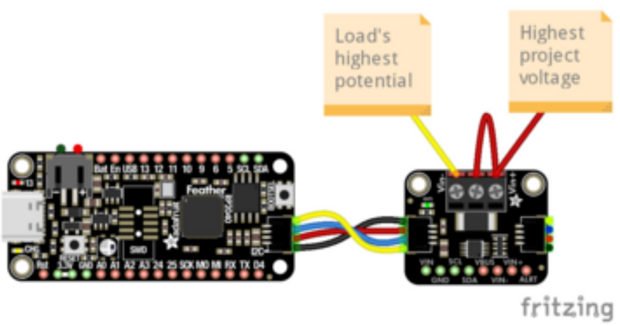
CircuitPython Microcontroller Wiring

First wire up the monitor to your board exactly as follows. The following is the monitor wired to a Feather RP2040 using the STEMMA connector for low side monitoring:



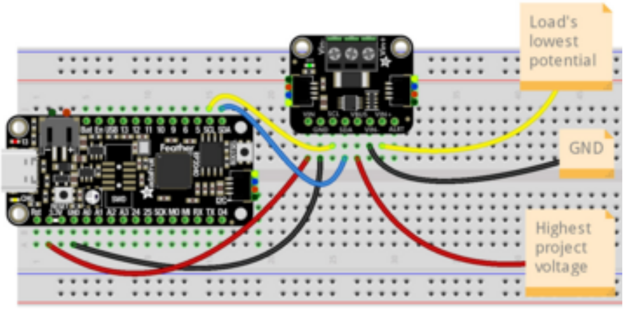
- Board STEMMA 3V to breakout VIN (red wire)
- Board STEMMA GND to breakout GND (black wire)
- Board STEMMA SCL to breakout SCL (yellow wire)
- Board STEMMA SDA to breakout SDA (blue wire)
- Breakout Vin- to GND (black wire)
- Breakout VBus to highest potential voltage (red wire)
- Breakout Vin+ to load's lowest potential (yellow wire)

Here is the monitor wired to a Feather RP2040 using the STEMMA connector for high side monitoring:



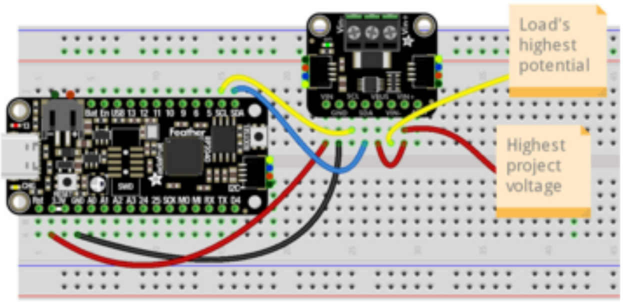
- Board STEMMA 3V to breakout VIN (red wire)
- Board STEMMA GND to breakout GND (black wire)
- Board STEMMA SCL to breakout SCL (yellow wire)
- Board STEMMA SDA to breakout SDA (blue wire)
- Breakout Vin- to load's highest potential (yellow wire)
- Breakout VBus to breakout Vin+ (red wire)
- Breakout Vin+ to highest project voltage (red wire)

The following is the monitor wired to a Feather RP2040 using a solderless breadboard for low side monitoring:



Board 3.3V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**
Breakout VIN- to **GND (black wire)**
Breakout VBUS to **highest potential voltage (red wire)**
Breakout VIN+ to **load's lowest potential (yellow wire)**

Here is the monitor wired to a Feather RP2040 using a solderless breadboard for high side monitoring:



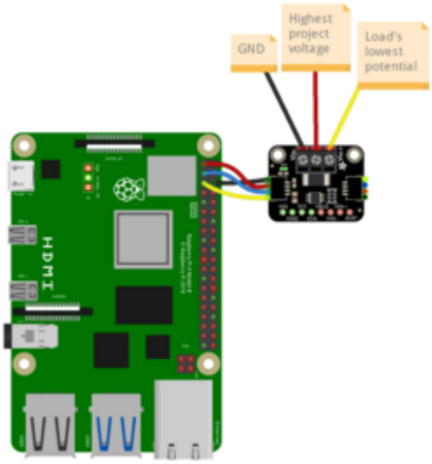
Board 3.3V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**
Breakout VIN- to **load's highest potential (yellow wire)**
Breakout VBUS to **breakout VIN+ (red wire)**
Breakout VIN+ to **highest project voltage (red wire)**

For high side monitoring, you can solder the VBus jumper closed on the back of the breakout to simplify your wiring.

Python Computer Wiring

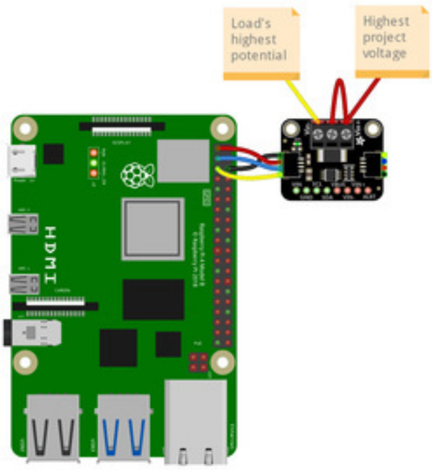
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C using the STEMMA connector for low side monitoring:



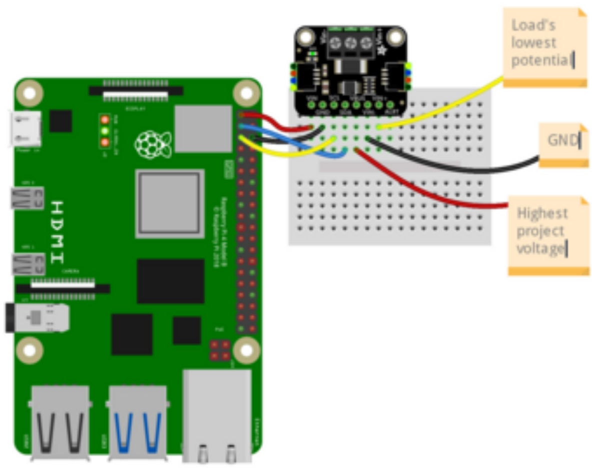
Pi 3.3V to **breakout VIN (red wire)**
Pi GND to **breakout GND (black wire)**
Pi SCL to **breakout SCL (yellow wire)**
Pi SDA to **breakout SDA (blue wire)**
Breakout VIN- to **GND (black wire)**
Breakout VBUS to **highest potential voltage (red wire)**
Breakout VIN+ to **load's lowest potential (yellow wire)**

Here's the wiring using the STEMMA connector for high side monitoring:



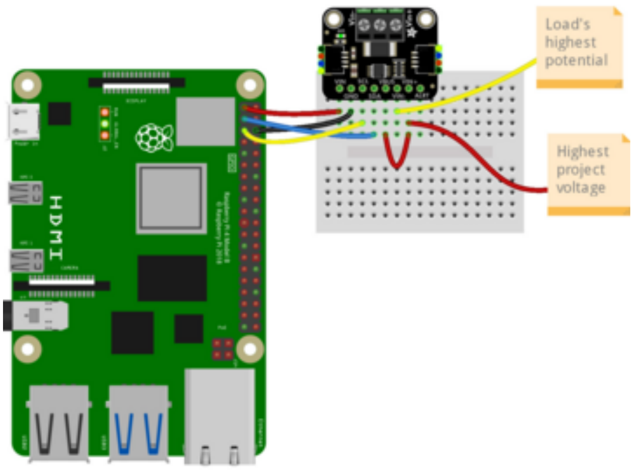
Pi 3.3V to **breakout VIN (red wire)**
Pi GND to **breakout GND (black wire)**
Pi SCL to **breakout SCL (yellow wire)**
Pi SDA to **breakout SDA (blue wire)**
Breakout Vin- to **load's highest potential (yellow wire)**
Breakout VBus to **breakout Vin+ (red wire)**
Breakout Vin+ to **highest project voltage (red wire)**

Here's the Raspberry Pi wired with I2C using a solderless breadboard for low side monitoring:



Pi 3.3V to **breakout VIN (red wire)**
Pi GND to **breakout GND (black wire)**
Pi SCL to **breakout SCL (yellow wire)**
Pi SDA to **breakout SDA (blue wire)**
Breakout VIN- to **GND (black wire)**
Breakout VBUS to **highest potential voltage (red wire)**
Breakout VIN+ to **load's lowest potential (yellow wire)**

Here's the wiring using a solderless breadboard for high side monitoring:



Pi 3.3V to **breakout VIN (red wire)**
Pi GND to **breakout GND (black wire)**
Pi SCL to **breakout SCL (yellow wire)**
Pi SDA to **breakout SDA (blue wire)**
Breakout Vin- to **load's highest potential (yellow wire)**
Breakout VBus to **breakout Vin+ (red wire)**
Breakout Vin+ to **highest project voltage (red wire)**

For high side monitoring, you can solder the VBus jumper closed on the back of the breakout to simplify your wiring.

Python Installation of INA228 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:

- `pip3 install adafruit-circuitpython-ina228`

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!

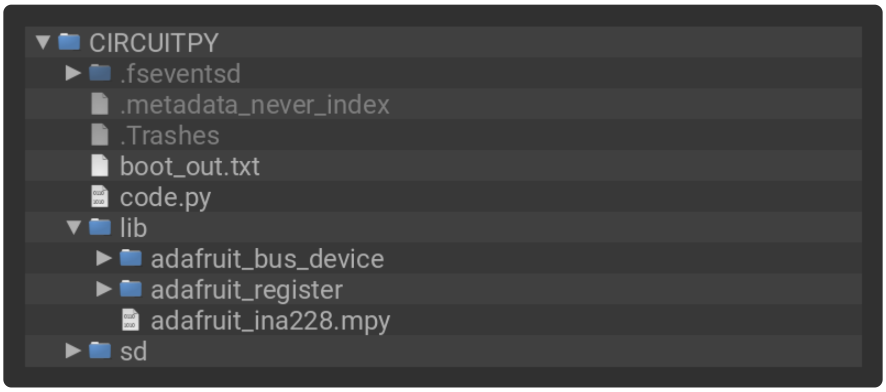
CircuitPython Usage

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_INA228** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_bus_device/**
- **adafruit_register/**
- **adafruit_ina228.mpy**



Python Usage

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

`python3 code.py`

Example Code

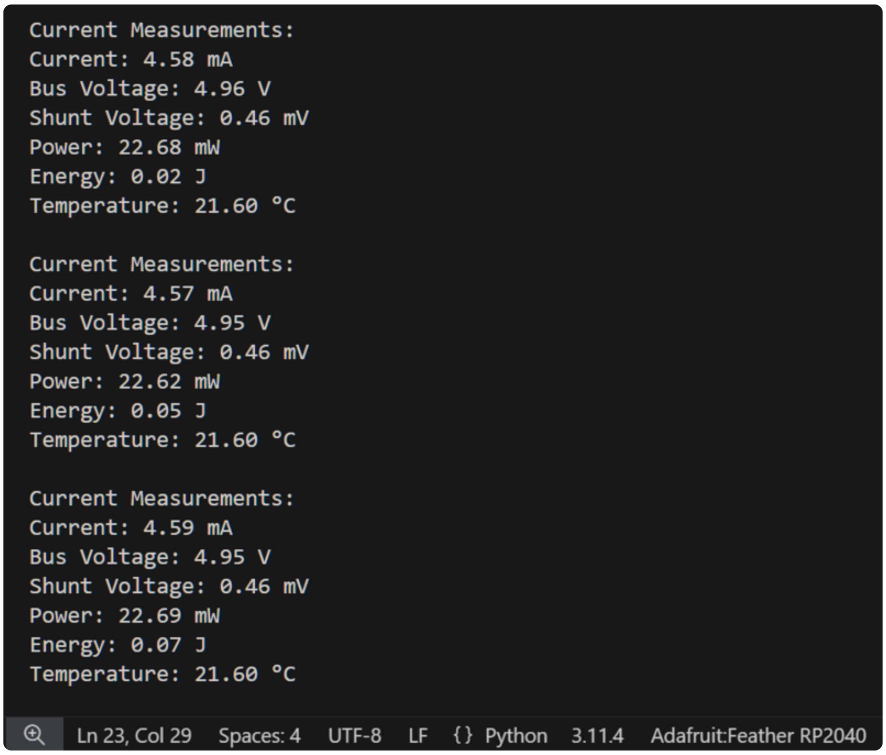
If running CircuitPython: Once everything is saved to the **CIRCUITPY** drive, [connect to the serial console](https://adafru.it/Bec) (https://adafru.it/Bec) to see the data printed out!

If running Python: The console output will appear wherever you are running Python.

```
# SPDX-FileCopyrightText: Copyright (c) 2025 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time

import board

import adafruit_ina228

i2c = board.I2C()
ina228 = adafruit_ina228.INA228(i2c)
print("Adafruit INA228 Test")

print(f"Bus conversion time: {ina228.conversion_time_bus} microseconds")
print(f"Shunt conversion time: {ina228.conversion_time_shunt} microseconds")
print(f"Samples averaged: {ina228.averaging_count}")

while True:
    print("\nCurrent Measurements:")
    print(f"Current: {ina228.current:.2f} mA")
    print(f"Bus Voltage: {ina228.voltage:.2f} V")
```

```
    print(f"Shunt Voltage: {ina228.shunt_voltage*1000:.2f} mV")
    print(f"Power: {ina228.power:.2f} mW")
    print(f"Energy: {ina228.energy:.2f} J")
    print(f"Temperature: {ina228.temperature:.2f} °C")
    time.sleep(1)
```

First, the sensor is instantiated over I2C. Then, in the loop, the measurements for current, bus voltage, shunt voltage, power, energy and temperature are printed to the serial console.



Python Docs

Python Docs (https://adafru.it/1ae0)

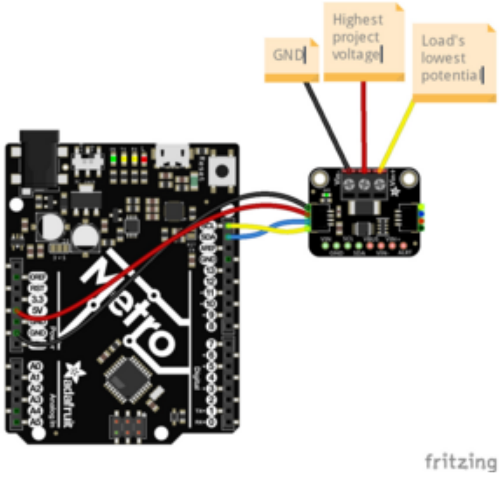
Arduino

Using the INA228 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the Adafruit_INA228 (https://adafru.it/1ae8) library, and running the provided example code.

Wiring

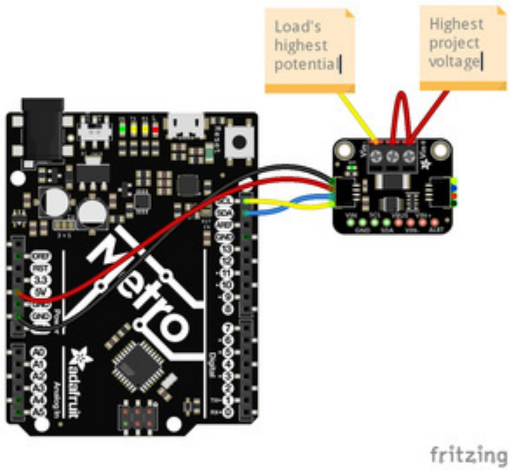
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the sensor VIN.

Here is an Adafruit Metro wired up to the breakout using the STEMMA QT connector for low side monitoring:



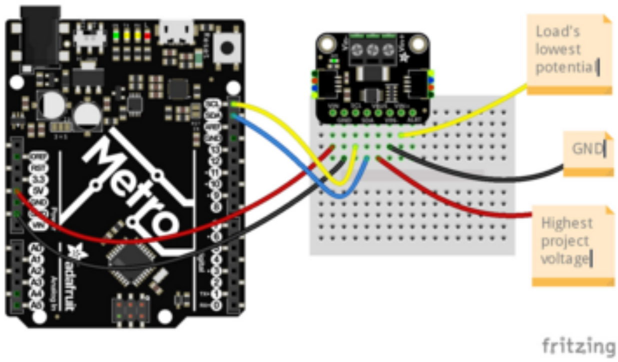
Board 5V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**
Breakout Vin- to **GND (black wire)**
Breakout VBus to **highest potential voltage (red wire)**
Breakout Vin+ to **load's lowest potential (yellow wire)**

Here is an Adafruit Metro wired up to the breakout using the STEMMA QT connector for high side monitoring:



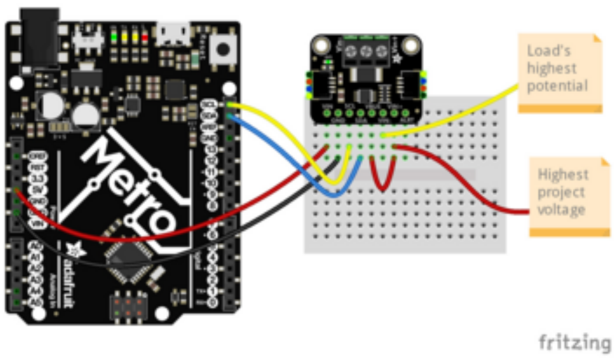
Board 5V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**
Breakout Vin- to **load's highest potential (yellow wire)**
Breakout VBus to **breakout Vin+ (red wire)**
Breakout Vin+ to **highest project voltage (red wire)**

Here is an Adafruit Metro wired up using a solderless breadboard for low side monitoring:



Board 5V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**
Breakout VIN- to **GND (black wire)**
Breakout VBUS to **highest potential voltage (red wire)**
Breakout VIN+ to **load's lowest potential (yellow wire)**

Here is an Adafruit Metro wired up using a solderless breadboard for high side monitoring:

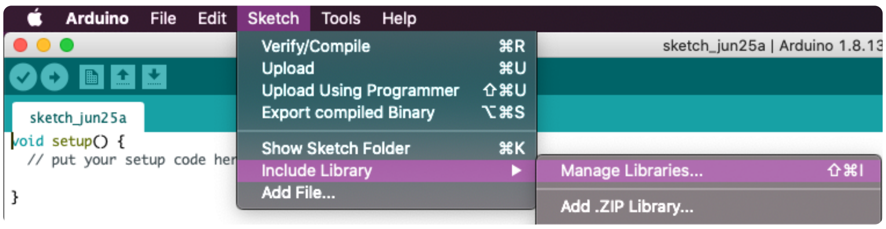


Board 5V to **breakout VIN (red wire)**
Board GND to **breakout GND (black wire)**
Board SCL to **breakout SCL (yellow wire)**
Board SDA to **breakout SDA (blue wire)**
Breakout VIN- to **load's highest potential (yellow wire)**
Breakout VBUS to **breakout VIN+ (red wire)**
Breakout VIN+ to **highest project voltage (red wire)**

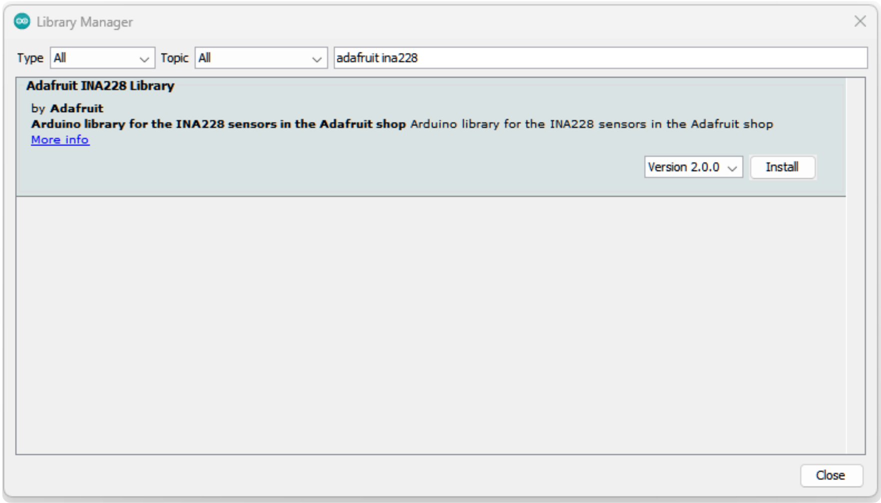
For high side monitoring, you can solder the VBus jumper closed on the back of the breakout to simplify your wiring.

Library Installation

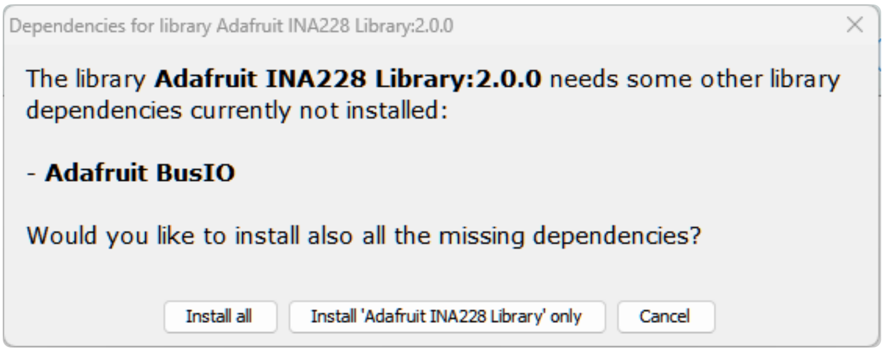
You can install the **Adafruit_INA228** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit_INA228**, and select the **Adafruit INA228** library:



If asked about dependencies, click "Install all".



If the "Dependencies" window does not come up, then you already have the dependencies installed.

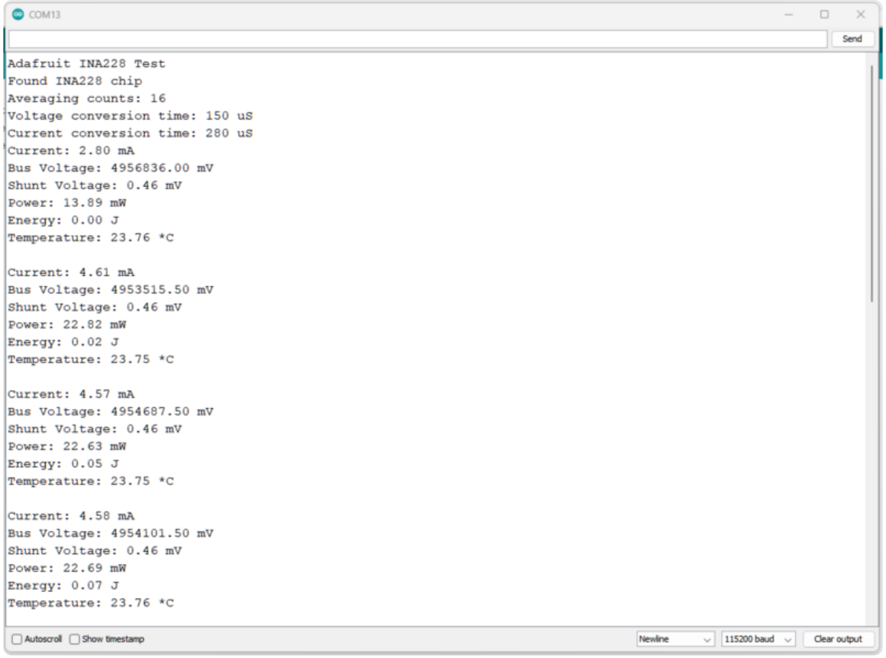
If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Example Code

```
#include <Adafruit_INA228.h>

Adafruit_INA228 ina228 = Adafruit_INA228();

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

  Serial.println("Adafruit INA228 Test");

  if (!ina228.begin()) {
    Serial.println("Couldn't find INA228 chip");
```

```
    while (1)
      ;
  }
  Serial.println("Found INA228 chip");
  // we need to set the resistance (default 0.1 ohm) and our max expected
  // current (no greater than 3.2A)
  ina228.setShunt(0.1, 1.0);

  ina228.setAveragingCount(INA228_COUNT_16);
  uint16_t counts[] = {1, 4, 16, 64, 128, 256, 512, 1024};
  Serial.print("Averaging counts: ");
  Serial.println(counts[ina228.getAveragingCount()]);

  // set the time over which to measure the current and bus voltage
  ina228.setVoltageConversionTime(INA228_TIME_150_us);
  Serial.print("Voltage conversion time: ");
  switch (ina228.getVoltageConversionTime()) {
  case INA228_TIME_50_us:
    Serial.print("50");
    break;
  case INA228_TIME_84_us:
    Serial.print("84");
    break;
  case INA228_TIME_150_us:
    Serial.print("150");
    break;
  case INA228_TIME_280_us:
    Serial.print("280");
    break;
  case INA228_TIME_540_us:
    Serial.print("540");
    break;
  case INA228_TIME_1052_us:
    Serial.print("1052");
    break;
  case INA228_TIME_2074_us:
    Serial.print("2074");
    break;
  case INA228_TIME_4120_us:
    Serial.print("4120");
    break;
  }
  Serial.println(" uS");

  ina228.setCurrentConversionTime(INA228_TIME_280_us);
  Serial.print("Current conversion time: ");
  switch (ina228.getCurrentConversionTime()) {
  case INA228_TIME_50_us:
    Serial.print("50");
    break;
  case INA228_TIME_84_us:
    Serial.print("84");
    break;
  case INA228_TIME_150_us:
    Serial.print("150");
    break;
  case INA228_TIME_280_us:
    Serial.print("280");
    break;
  case INA228_TIME_540_us:
    Serial.print("540");
    break;
  case INA228_TIME_1052_us:
    Serial.print("1052");
    break;
  case INA228_TIME_2074_us:
    Serial.print("2074");
    break;
  case INA228_TIME_4120_us:
```

```
    Serial.print("4120");
    break;
  }
  Serial.println(" uS");

  // default polarity for the alert is low on ready, but
  // it can be inverted!
  // ina228.setAlertPolarity(1);
}

void loop() {

  // by default the sensor does continuous reading, but
  // we can set to triggered mode. to do that, we have to set
  // the mode to trigger a new reading, then wait for a conversion
  // either by checking the ALERT pin or reading the ready register
  // ina228.setMode(INA228_MODE_TRIGGERED);
  // while (!ina228.conversionReady())
  //  delay(1);

  Serial.print("Current: ");
  Serial.print(ina228.readCurrent());
  Serial.println(" mA");

  Serial.print("Bus Voltage: ");
  Serial.print(ina228.readBusVoltage());
  Serial.println(" mV");

  Serial.print("Shunt Voltage: ");
  Serial.print(ina228.readShuntVoltage());
  Serial.println(" mV");

  Serial.print("Power: ");
  Serial.print(ina228.readPower());
  Serial.println(" mW");

  Serial.print("Energy: ");
  Serial.print(ina228.readEnergy());
  Serial.println(" J");

  Serial.print("Temperature: ");
  Serial.print(ina228.readDieTemp());
  Serial.println(" *C");

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



Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the INA228 recognized over I2C. Then, your connected project's current, bus voltage, shunt voltage, power and energy measurements are printed out every second, along with the INA228's temperature reading in Celsius.

Arduino Docs

Arduino Docs (https://adafru.it/1adZ)

Downloads

Files

- INA228 Datasheet (https://adafru.it/1ae9)
- EagleCAD PCB Files on GitHub (https://adafru.it/1aea)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1aeb)

Schematic and Fab Print