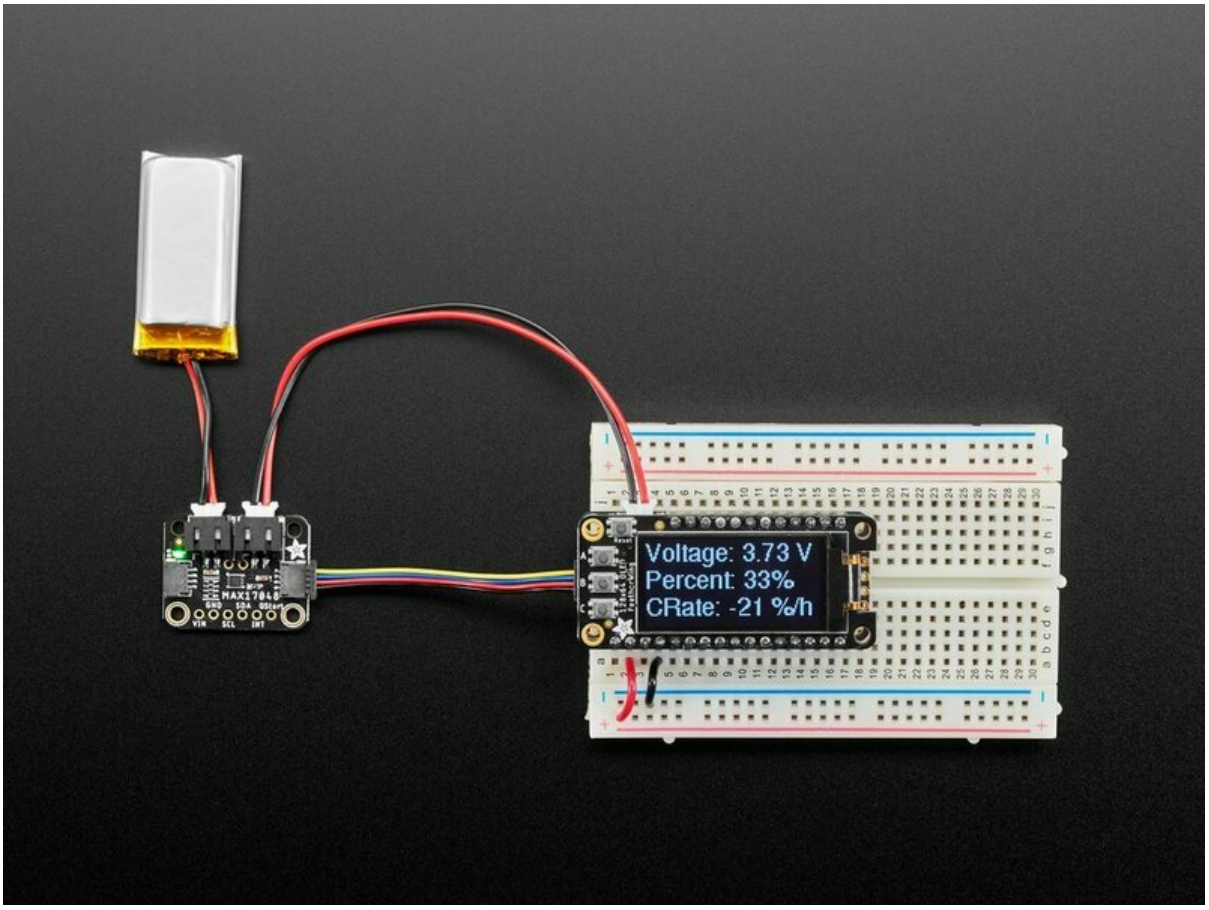
Adafruit MAX17048 LiPoly / LiIon Fuel Gauge and Battery Monitor

Created by Liz Clark

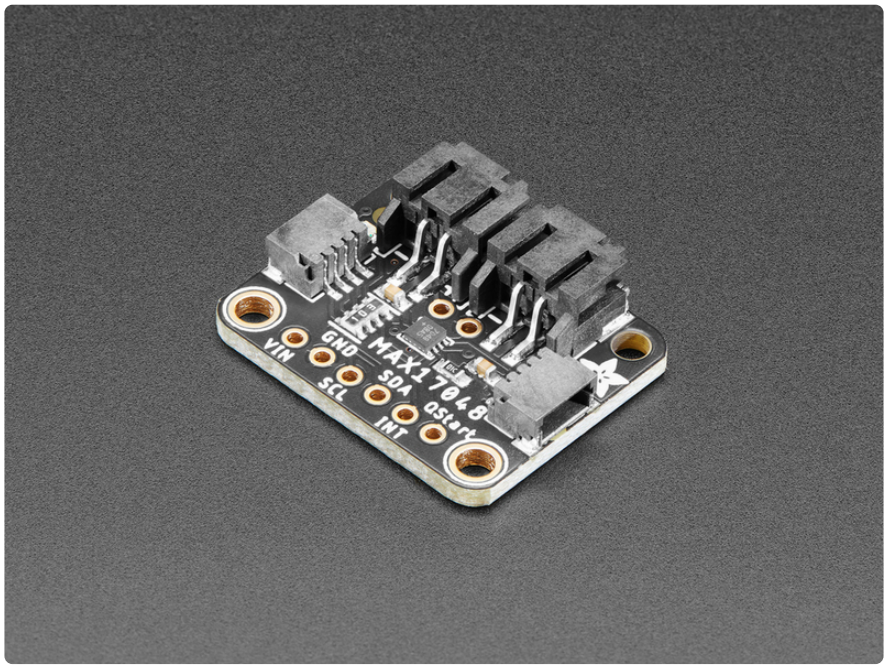


https://learn.adafruit.com/adafruit-max17048-lipoly-liion-fuel-gauge-and-battery-monitor

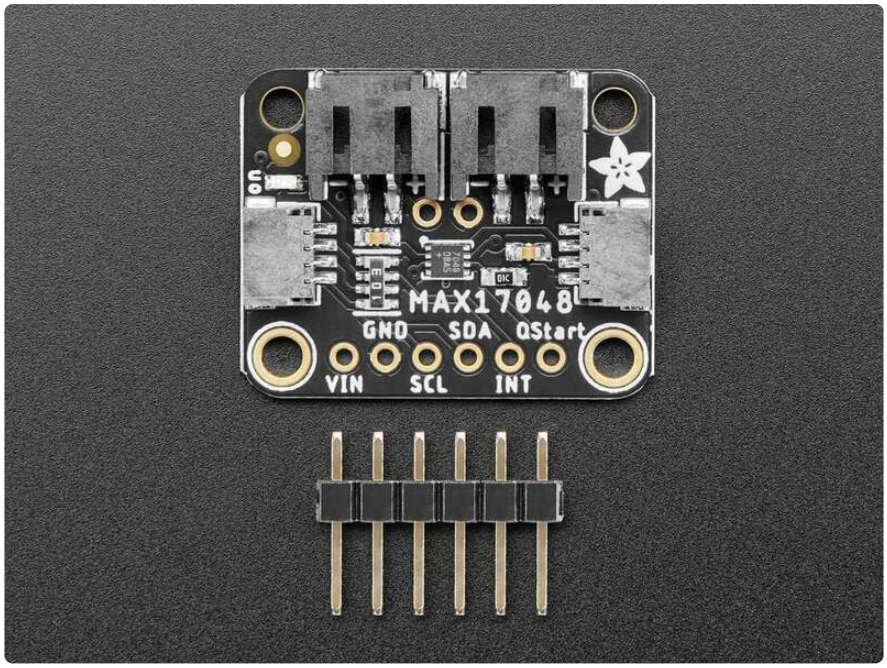
Last updated on 2025-03-04 01:36:31 PM EST

Table of Contents

Overview



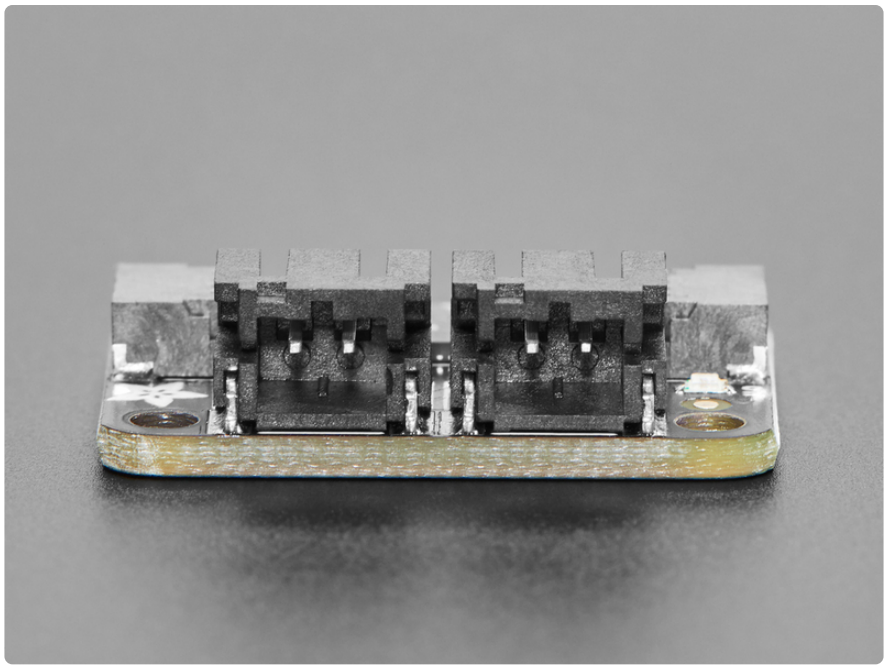
Low cost Lithium Polymer batteries have revolutionized electronics - they're thin, they're light, they can be regulated down to 3.3V and they're easy to charge. On your phone, there's a little image of a battery cell that tells you the percentage of charge - so you know when you absolutely need to plug it in and when you can stay untethered.



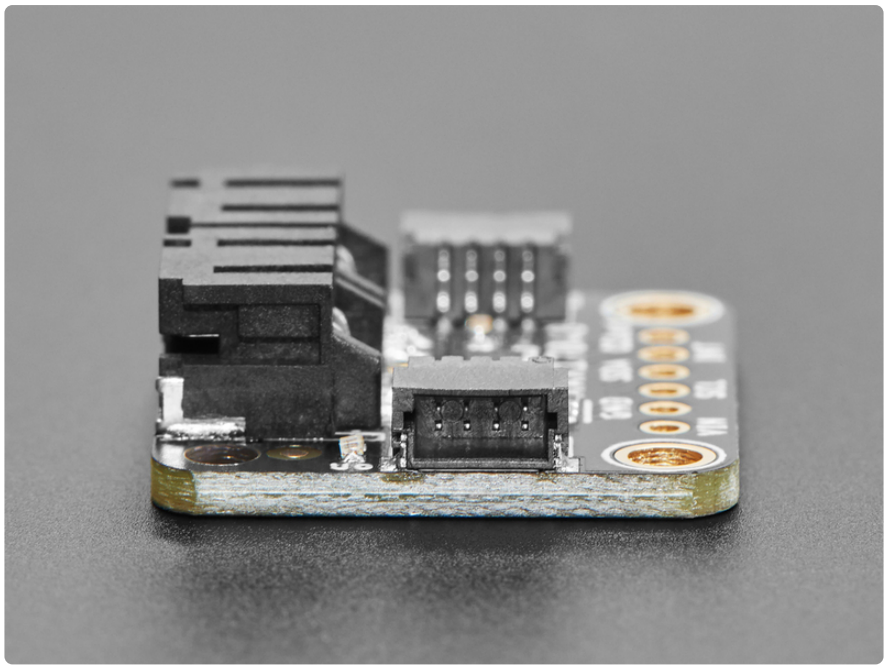
The **Adafruit MAX17048 LiPoly / LiIon Fuel Gauge and Battery Monitor** does the same thing. Connect it to your Lipoly or LiIon battery (https://adafru.it/NdY) and it will

let you know the voltage of the cell, it does the annoying math of decoding the non-linear voltage to get you a valid percentage as well!

Since this nice chip is I2C, it works with any and all microcontroller or microcomputer boards, from the Arduino UNO up to the Raspberry Pi. And you don't have to worry about logic level, as the gauge runs with 3.3V or 5.0V power and logic equally fine.

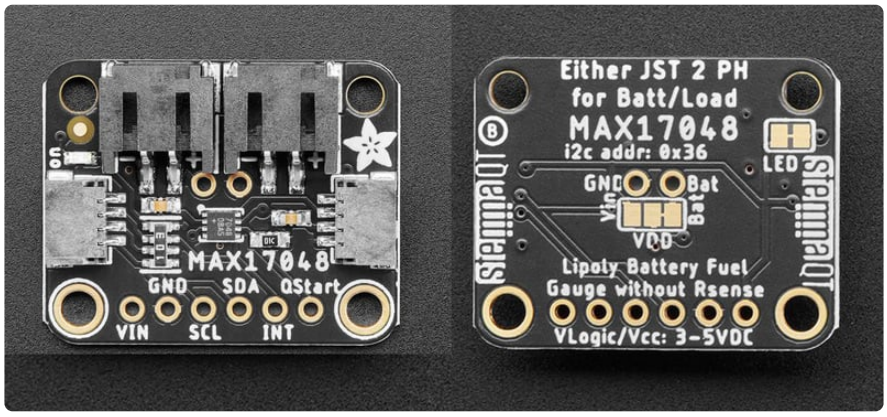


To use, connect a 1S 3.7-4.2V Lithium Ion or Polymer battery (https://adafru.it/NdY) to one of the JST 2 PH ports (either one). Then use a JST PH jumper cable (http://adafru.it/4714) to connect to your boost converter, Feather, whatever! Use the I2C interface and our Arduino (https://adafru.it/10FG) or CircuitPython/Python (https://adafru.it/10RA) library code to read the voltage and percentage whenever you like. There are various alerts, low power modes that can be customized as desired.



To get you going fast, we spun up a custom-made PCB in the **STEMMA QT** form factor (https://adafru.it/LBQ), making it easy to interface with. The STEMMA QT connectors (https://adafru.it/JqB) on either side are compatible with the SparkFun Qwiic (https://adafru.it/Fpw) I2C connectors. This allows you to make solderless connections between your development board and the MAX17048 or to chain it with a wide range of other sensors and accessories using a **compatible cable** (https://adafru.it/JnB). QT Cable is not included, but we have a variety in the shop (https://adafru.it/17VE)

Pinouts



The default I2C address is **0x36**.

Power Pins

- **VIN** - The chip can safely run from 3-5VDC. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V.

- **Bat** - Output from the battery input voltage.
- **GND** - common ground for power and logic.

> The MAX17048 is powered by the connected battery, not by VIN or the STEMMA/QT connector. If no battery is plugged in, or the battery is too low, the MAX17048 will not respond to I2C scans or commands.

I2C Logic Pins

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

JST Ports

Theres are two JST ports, they are equivalent. Connect the battery to either one, then the load/charger to the other. **Watch out for battery polarity if not using an Adafruit battery!**

> The two ports simply connect together, and to the battery. You can use/charge the battery while connected by having the battery on one port, and then connecting a charger/project to the other

Other Pins

- **INT** - Interrupt signal out, you can set this up to pull low when the voltage or percentage drops below a threshold. Pulled up to VIN with a 10K resistor.
- **QStart** - Quick-start input. It allows reset of the MAX17048 through hardware.

Power LED and Jumper

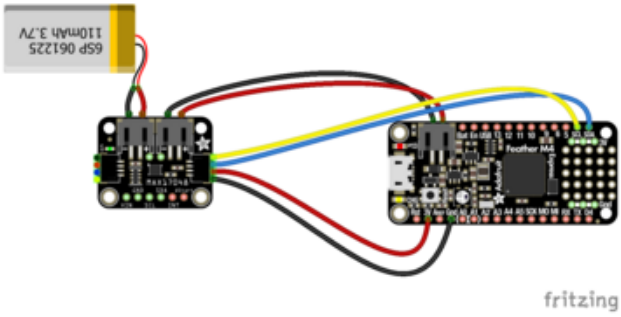
- **Power LED** - This LED is a green LED located on the front of the board. It is labeled **on**.
- **LED jumper** - This jumper is located on the back of the board. Cut the trace on this jumper to cut power to the "on" LED.

Python & CircuitPython

It's easy to use the **MAX17048** with Python or CircuitPython, and the Adafruit_CircuitPython_MAX1704x (https://adafru.it/10RA) module. This module allows you to easily write Python code that reads the values from the **MAX17048**'s battery monitoring and alert functions. 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 MAX17048 to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C using one of the handy **STEMMA QT** (https://adafru.it/Ft4) connectors:



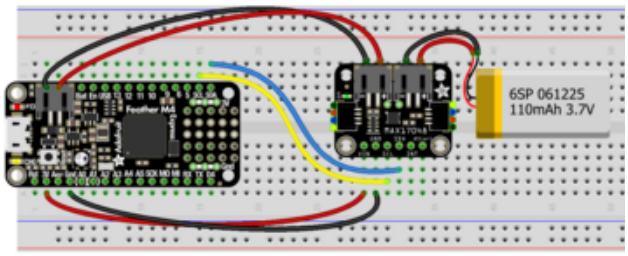
STEMMA

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)**

Battery

Plug a 3.7/4.2V lithium polymer or lithium ion rechargeable battery into **either of the JST battery ports on the board**.
Plug the other **board JST Battery port** into the **Feather JST port** using the cable included with the board.

You can also use standard **0.100" pitch** headers to wire it up on a breadboard:



STEMMA

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)**

Battery

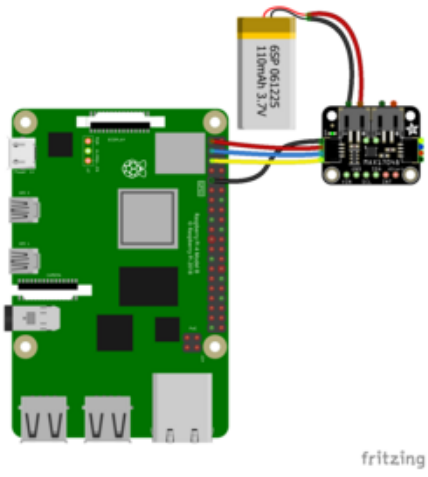
Plug a 3.7/4.2V lithium polymer or lithium ion rechargeable battery into **either of the JST battery ports on the board**.
Plug the other **board JST Battery port** into the **Feather JST port** using the cable included with the board.

Watch out for battery polarity! A reversed battery will damage the monitor. There are + and - symbols on the PCB to indicate which is which.

Python Computer Wiring

Since there's dozens of Linux computers/boards you can use, below shows 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).

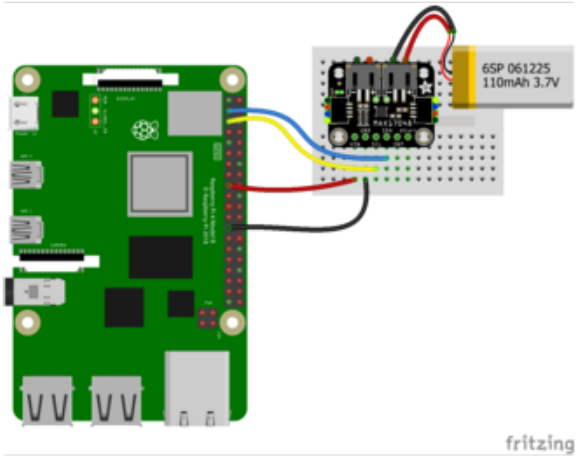
Here's the Raspberry Pi wired to the sensor using I2C and a **STEMMA QT** (https://adafru.it/Ft4) connector:



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)**

Plug a 3.7/4.2V lithium polymer or lithium ion rechargeable battery into either of the JST battery ports.

Finally here is an example of how to wire up a Raspberry Pi to the sensor using a solderless breadboard:



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)**

Plug a 3.7/4.2V lithium polymer or lithium ion rechargeable battery into either of the JST battery ports.

Python Installation of MAX1704X 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-max1704x`

If your default Python is version 3, you may need to run `pip` instead. 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 MAX1704X 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_max1704x.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:

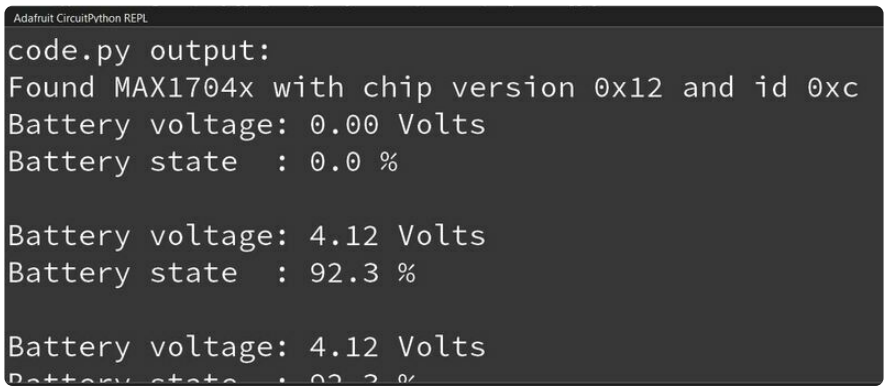
```
python3 code.py
```

Example Code - Simple Test

```
# SPDX-FileCopyrightText: Copyright (c) 2022 ladyada for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

import time
import board
import adafruit_max1704x

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
max17 = adafruit_max1704x.MAX17048(i2c)

print(
    "Found MAX1704x with chip version",
    hex(max17.chip_version),
    "and id",
    hex(max17.chip_id),
)

# Quick starting allows an instant 'auto-calibration' of the battery. However, its
a bad idea
# to do this right when the battery is first plugged in or if there's a lot of load
on the battery
# so uncomment only if you're sure you want to 'reset' the chips charge calculator.
# print("Quick starting")
# max17.quick_start = True

while True:
    print(f"Battery voltage: {max17.cell_voltage:.2f} Volts")
    print(f"Battery state  : {max17.cell_percent:.1f} %")
    print("")
    time.sleep(1)
```

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

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



In the **max1704x_simpletest.py** example, the MAX17048 is instantiated on I2C. The chip version and chip ID is printed to the REPL.

In the loop, the battery's voltage and charge percentage is printed to the REPL every second.

Example Code - Advanced Test

```
# SPDX-FileCopyrightText: Copyright (c) 2022 ladyada for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

import time
import board
import adafruit_max1704x

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
max17 = adafruit_max1704x.MAX17048(i2c)

print(
    "Found MAX1704x with chip version",
    hex(max17.chip_version),
    "and id",
    hex(max17.chip_id),
)

# Quick starting allows an instant 'auto-calibration' of the battery. However, its
a bad idea
# to do this right when the battery is first plugged in or if there's a lot of load
on the battery
# so uncomment only if you're sure you want to 'reset' the chips charge calculator.
# print("Quick starting")
max17.quick_start = True

# The reset voltage is what the chip considers 'battery has been removed and
replaced'
# The default is 3.0 Volts but you can change it here:
# max17.reset_voltage = 2.5
print("MAX1704x reset voltage = %0.1f V" % max17.reset_voltage)

# The analog comparator is used to detect the rest voltage, if you don't think the
```

```
battery
# will ever be removed this can reduce current usage (see datasheet on VRESET.Dis)
print("Analog comparator is ", end="")
if max17.comparator_disabled:
    print("disabled")
else:
    print("enabled")

# Hibernation mode reduces how often the ADC is read, for power reduction. There is
an automatic
# enter/exit mode but you can also customize the activity threshold both as voltage
and charge rate
# max17.activity_threshold = 0.15
print("MAX1704x activity threshold = %0.2f V" % max17.activity_threshold)

# max17.hibernation_threshold = 5
print("MAX1704x hibernation threshold = %0.2f %%" % max17.hibernation_threshold)

# You can also 'force' hibernation mode!
# max17.hibernate()
# ...or force it to wake up!
# max17.wake()

# The alert pin can be used to detect when the voltage of the battery goes below or
# above a voltage, you can also query the alert in the loop.
max17.voltage_alert_min = 3.5
print("Voltage alert minimum = %0.2f V" % max17.voltage_alert_min)
max17.voltage_alert_max = 4.1
print("Voltage alert maximum = %0.2f V" % max17.voltage_alert_max)

print("")
while True:
    print(f"Battery voltage: {max17.cell_voltage:.2f} Volts")
    print(f"Battery state  : {max17.cell_percent:.1f} %")

    # we can check if we're hibernating or not
    if max17.hibernating:
        print("Hibernating!")

    if max17.active_alert:
        print("Alert!")
        if max17.reset_alert:
            print("  Reset indicator")
            max17.reset_alert = False  # clear the alert

        if max17.voltage_high_alert:
            print("  Voltage high")
            max17.voltage_high_alert = False  # clear the alert

        if max17.voltage_low_alert:
            print("  Voltage low")
            max17.voltage_low_alert = False  # clear the alert

        if max17.voltage_reset_alert:
            print("  Voltage reset")
            max17.voltage_reset_alert = False  # clear the alert

        if max17.SOC_low_alert:
            print("  Charge low")
            max17.SOC_low_alert = False  # clear the alert

        if max17.SOC_change_alert:
            print("  Charge changed")
            max17.SOC_change_alert = False  # clear the alert
    print("")
    time.sleep(1)
```

In the **max1704x_advanced.py** example, the MAX17048 is instantiated on I2C. The chip version and chip ID is printed to the REPL along with thresholds for the alerts that are available in the library.

In the loop, the battery's voltage, charge percentage and any active alerts are printed to the REPL every second.

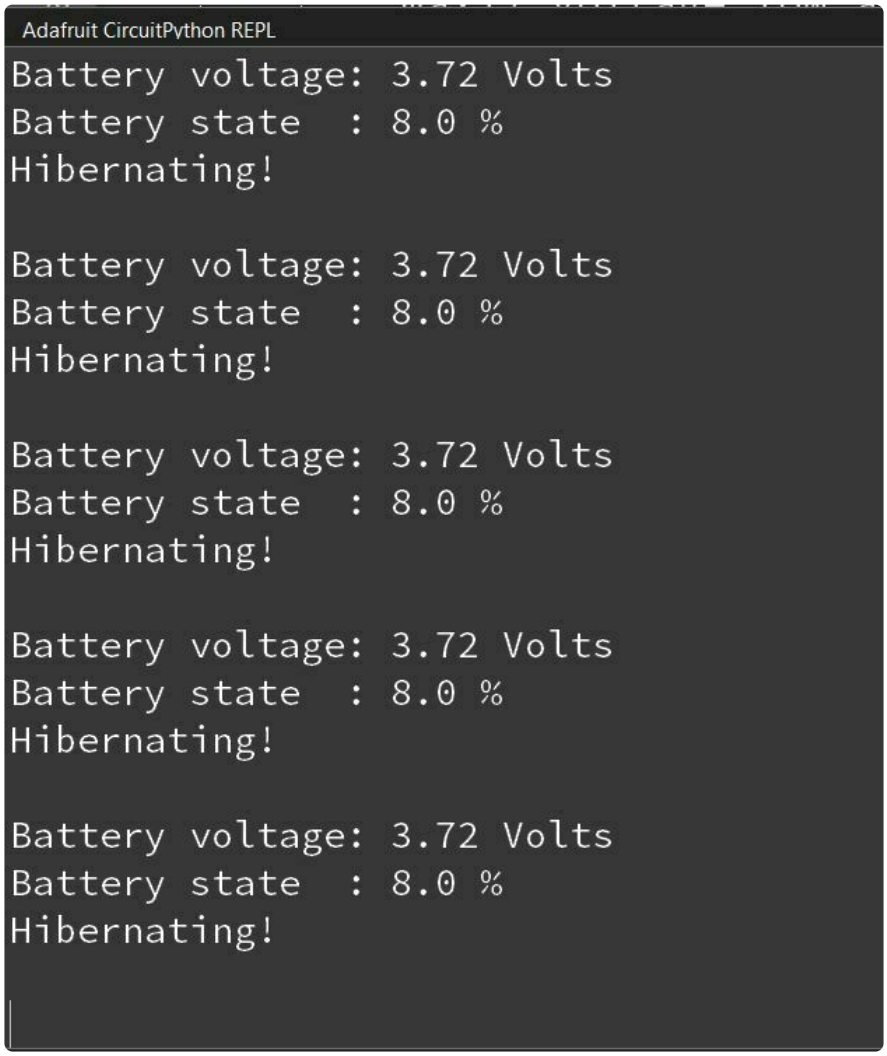
Voltage High Alert:

```
Adafruit CircuitPython REPL
Found MAX1704x with chip version 0x12 and id 0xc
MAX1704x reset voltage = 3.0 V
Analog comparator is enabled
MAX1704x activity threshold = 0.06 V
MAX1704x hibernation threshold = 26.62 %
Voltage alert minimum = 3.50 V
Voltage alert maximum = 4.08 V

Battery voltage: 4.13 Volts
Battery state  : 0.0 %
Alert!
  Voltage high

Battery voltage: 4.12 Volts
Battery state  : 92.3 %
Alert!
  Voltage high
```

Voltage Low Alert:

```
Adafruit CircuitPython REPL
code.py output:
Found MAX1704x with chip version 0x12 and id 0xc
MAX1704x reset voltage = 3.0 V
Analog comparator is enabled
MAX1704x activity threshold = 0.06 V
MAX1704x hibernation threshold = 26.62 %
Voltage alert minimum = 3.50 V
Voltage alert maximum = 4.08 V

Battery voltage: 3.13 Volts
Battery state  : 0.0 %
Alert!
  Voltage low

Battery voltage: 3.13 Volts
Battery state  : 0.0 %
Alert!
  Voltage low
```

Hibernating Alert:



Python Docs

Python Docs (https://adafru.it/10FF)

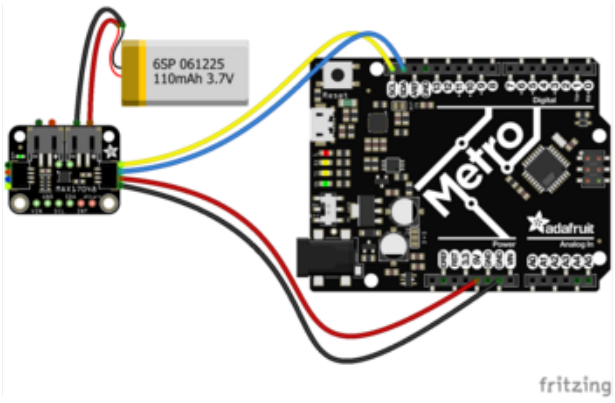
Arduino

Using the MAX17048 with Arduino involves wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit_MAX1704X (https://adafru.it/10FG) 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 MAX17048 VIN.

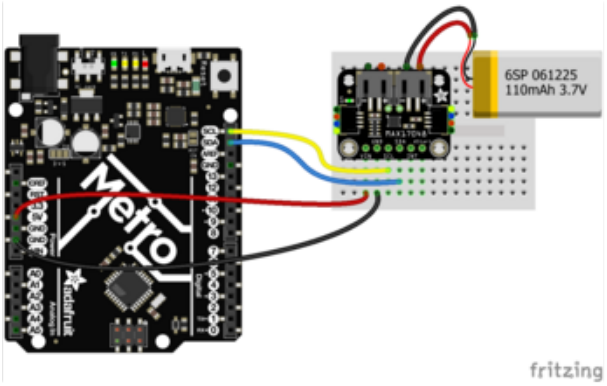
Here is an Adafruit Metro wired up to the MAX17048 using the STEMMA QT connector:



Board 5V 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)**

Plug a 3.7/4.2V lithium polymer or lithium ion rechargable battery into either of the JST battery ports.

Here is an Adafruit Metro wired up using a solderless breadboard:

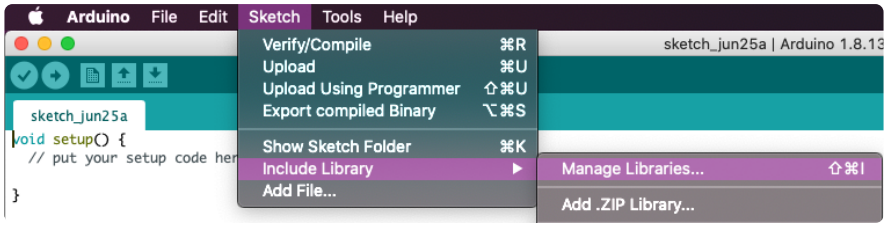


Board 5V to **sensor VIN (red wire)**
Board GND to **sensor GND (black wire)**
Board SCL to **sensor SCL (yellow wire)**
Board SDA to **sensor SDA (blue wire)**

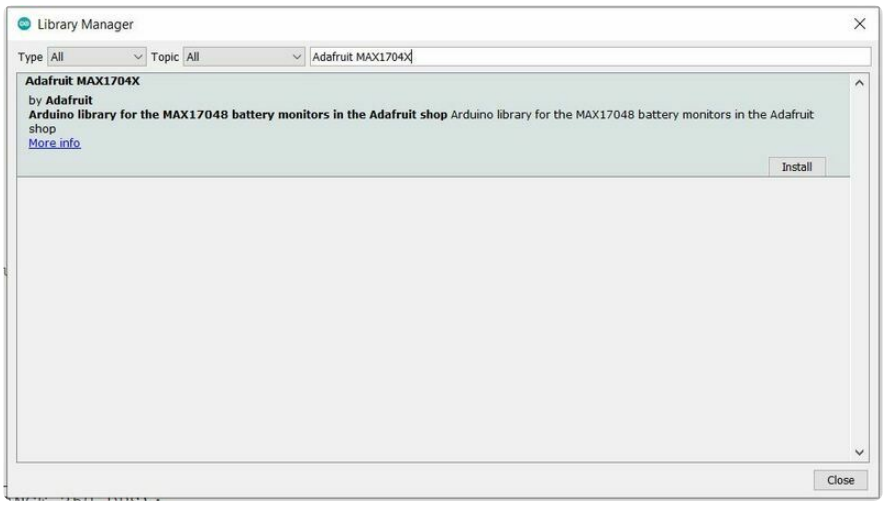
Plug a 3.7/4.2V lithium polymer or lithium ion rechargable battery into either of the JST battery ports.

Library Installation

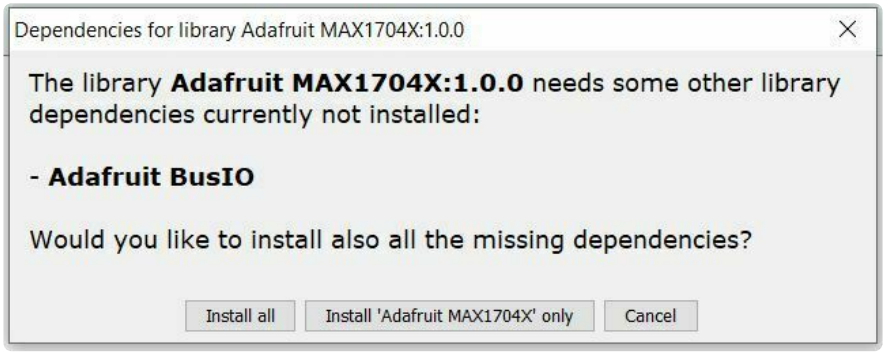
You can install the **Adafruit MAX1704X** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit MAX1704X**, and select the **Adafruit MAX1704X** library:



If asked about dependencies, click "Install all".



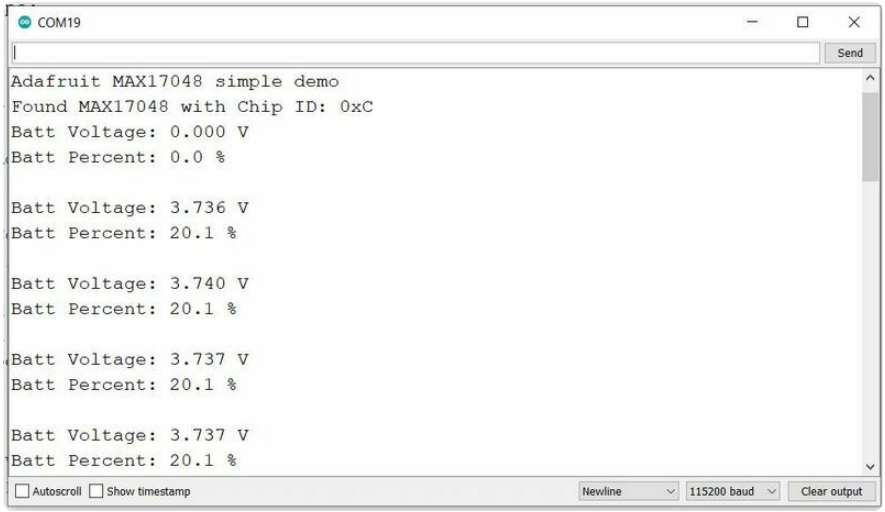
If the "Dependencies" window does not come up, then you already have the dependencies installed.

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

Example Code - Basic

```
#include "Adafruit_MAX1704X.h"

Adafruit_MAX17048 maxlipo;

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);    // wait until serial monitor opens

  Serial.println(F("\nAdafruit MAX17048 simple demo"));

  while (!maxlipo.begin()) {
    Serial.println(F("Couldnt find Adafruit MAX17048?\nMake sure a battery is
plugged in!"));
    delay(2000);
  }
```

```
  Serial.print(F("Found MAX17048"));
  Serial.print(F(" with Chip ID: 0x"));
  Serial.println(maxlipo.getChipID(), HEX);
}

void loop() {
  float cellVoltage = maxlipo.cellVoltage();
  if (isnan(cellVoltage)) {
    Serial.println("Failed to read cell voltage, check battery is connected!");
    delay(2000);
    return;
  }
  Serial.print(F("Batt Voltage: ")); Serial.print(cellVoltage, 3); Serial.println("
V");
  Serial.print(F("Batt Percent: ")); Serial.print(maxlipo.cellPercent(), 1);
Serial.println(" %");
  Serial.println();

  delay(2000);  // dont query too often!
}
```



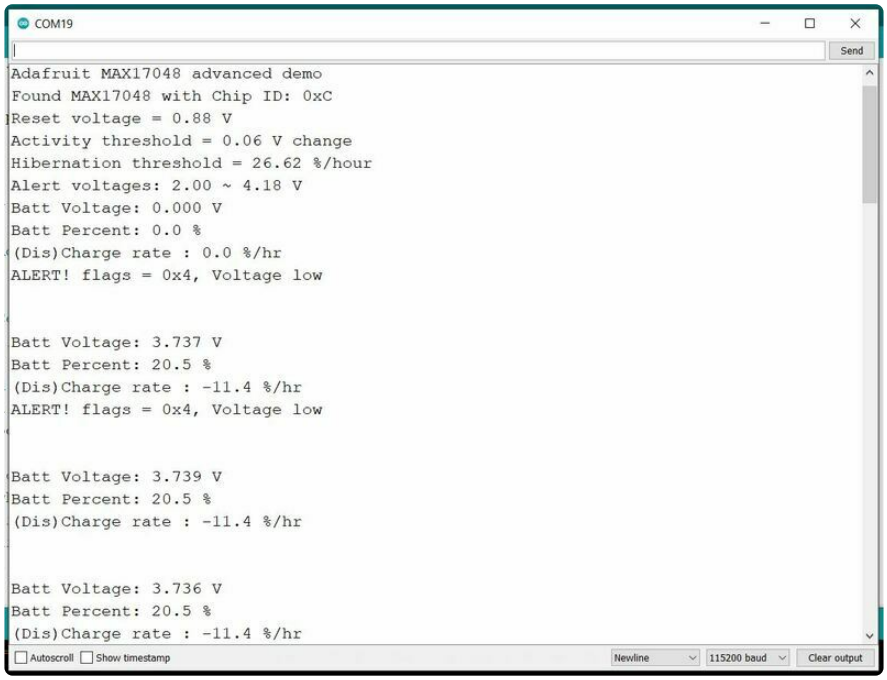
Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You should see the battery's voltage and charge percentage being printed out. You'll see the values change depending on the battery's status.

Example Code - Advanced

```
#include "Adafruit_MAX1704X.h"

Adafruit_MAX17048 maxlipo;

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);    // wait until serial monitor opens

  Serial.println(F("\nAdafruit MAX17048 advanced demo"));

  while (!maxlipo.begin()) {
    Serial.println(F("Couldnt find Adafruit MAX17048?\nMake sure a battery is
plugged in!"));
    delay(2000);
  }
  Serial.print(F("Found MAX17048"));
  Serial.print(F(" with Chip ID: 0x"));
  Serial.println(maxlipo.getChipID(), HEX);
```

```
  // Quick starting allows an instant 'auto-calibration' of the battery. However,
its a bad idea
  // to do this right when the battery is first plugged in or if there's a lot of
load on the battery
  // so uncomment only if you're sure you want to 'reset' the chips charge
calculator.
  // Serial.println("Quick starting");
  // maxlipo.quickStart();

  // The reset voltage is what the chip considers 'battery has been removed and
replaced'
  // The default is 3.0 Volts but you can change it here:
  //maxlipo.setResetVoltage(2.5);
  Serial.print(F("Reset voltage = "));
  Serial.print(maxlipo.getResetVoltage());
  Serial.println(" V");

  // Hibernation mode reduces how often the ADC is read, for power reduction. There
is an automatic
  // enter/exit mode but you can also customize the activity threshold both as
voltage and charge rate

  //maxlipo.setActivityThreshold(0.15);
  Serial.print(F("Activity threshold = "));
  Serial.print(maxlipo.getActivityThreshold());
  Serial.println(" V change");

  //maxlipo.setHibernationThreshold(5);
  Serial.print(F("Hibernation threshold = "));
  Serial.print(maxlipo.getHibernationThreshold());
  Serial.println(" %/hour");

  // You can also 'force' hibernation mode!
  // maxlipo.hibernate();
  // ...or force it to wake up!
  // maxlipo.wake();

  // The alert pin can be used to detect when the voltage of the battery goes below
or
  // above a voltage, you can also query the alert in the loop.
  maxlipo.setAlertVoltages(2.0, 4.2);

  float alert_min, alert_max;
  maxlipo.getAlertVoltages(alert_min, alert_max);
  Serial.print("Alert voltages: ");
  Serial.print(alert_min); Serial.print(" ~ ");
  Serial.print(alert_max); Serial.println(" V");
}

void loop() {
  float cellVoltage = maxlipo.cellVoltage();
  if (isnan(cellVoltage)) {
    Serial.println("Failed to read cell voltage, check battery is connected!");
    delay(2000);
    return;
  }
  Serial.print(F("Batt Voltage: ")); Serial.print(cellVoltage, 3); Serial.println("
V");
  Serial.print(F("Batt Percent: ")); Serial.print(maxlipo.cellPercent(), 1);
Serial.println(" %");
  Serial.print(F("(Dis)Charge rate : ")); Serial.print(maxlipo.chargeRate(), 1);
Serial.println(" %/hr");

  // we can check if we're hibernating or not
  if (maxlipo.isHibernating()) {
    Serial.println(F("Hibernating!"));
  }
```

```
  if (maxlipo.isActiveAlert()) {
    uint8_t status_flags = maxlipo.getAlertStatus();
    Serial.print(F("ALERT! flags = 0x"));
    Serial.print(status_flags, HEX);

    if (status_flags & MAX1704X_ALERTFLAG_SOC_CHANGE) {
      Serial.print(", SOC Change");
      maxlipo.clearAlertFlag(MAX1704X_ALERTFLAG_SOC_CHANGE); // clear the alert
    }
    if (status_flags & MAX1704X_ALERTFLAG_SOC_LOW) {
      Serial.print(", SOC Low");
      maxlipo.clearAlertFlag(MAX1704X_ALERTFLAG_SOC_LOW); // clear the alert
    }
    if (status_flags & MAX1704X_ALERTFLAG_VOLTAGE_RESET) {
      Serial.print(", Voltage reset");
      maxlipo.clearAlertFlag(MAX1704X_ALERTFLAG_VOLTAGE_RESET); // clear the alert
    }
    if (status_flags & MAX1704X_ALERTFLAG_VOLTAGE_LOW) {
      Serial.print(", Voltage low");
      maxlipo.clearAlertFlag(MAX1704X_ALERTFLAG_VOLTAGE_LOW); // clear the alert
    }
    if (status_flags & MAX1704X_ALERTFLAG_VOLTAGE_HIGH) {
      Serial.print(", Voltage high");
      maxlipo.clearAlertFlag(MAX1704X_ALERTFLAG_VOLTAGE_HIGH); // clear the alert
    }
    if (status_flags & MAX1704X_ALERTFLAG_RESET_INDICATOR) {
      Serial.print(", Reset Indicator");
      maxlipo.clearAlertFlag(MAX1704X_ALERTFLAG_RESET_INDICATOR); // clear the alert
    }
    Serial.println();
  }
  Serial.println();
  Serial.println();

  delay(2000);  // dont query too often!
}
```



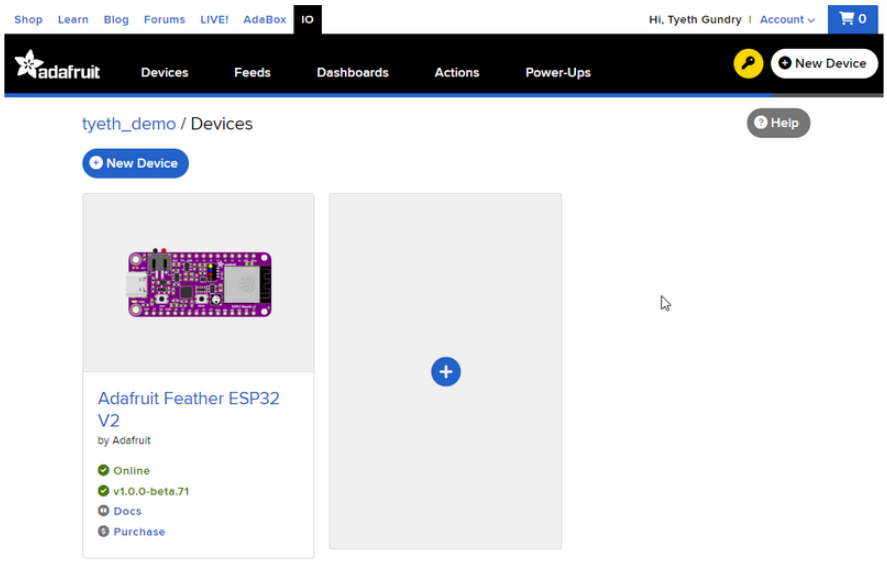
Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You should see the battery's voltage, charge percentage,

charge/discharge rate and any alert status messages being printed out. You'll see the values change depending on the battery's status.

Arduino Docs

Arduino Docs (https://adafru.it/10FG)

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 quick start guide before continuing.

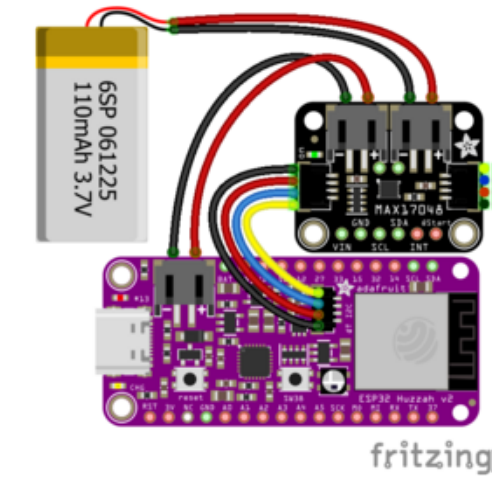
Quickstart: Adafruit IO WipperSnapper

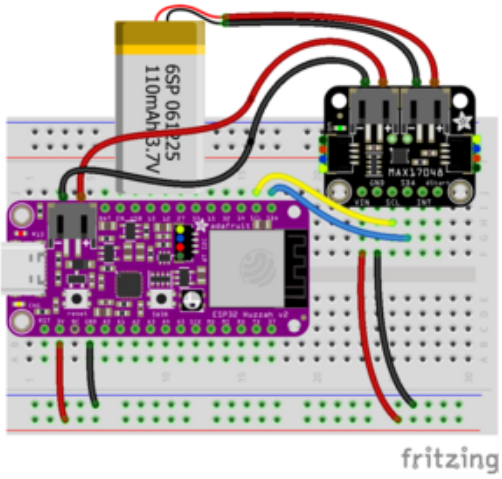
https://adafru.it/Vfd

Wiring

First, wire up a MAX17048 to your board exactly as follows (note that some boards might already have the sensor integrated on the PCB).

Here is an example of the MAX17048 wired to an Adafruit ESP32 Feather V2 (http://adafru.it/5400) using I2C with a STEMMA QT cable (no soldering required) (http://adafru.it/4210)



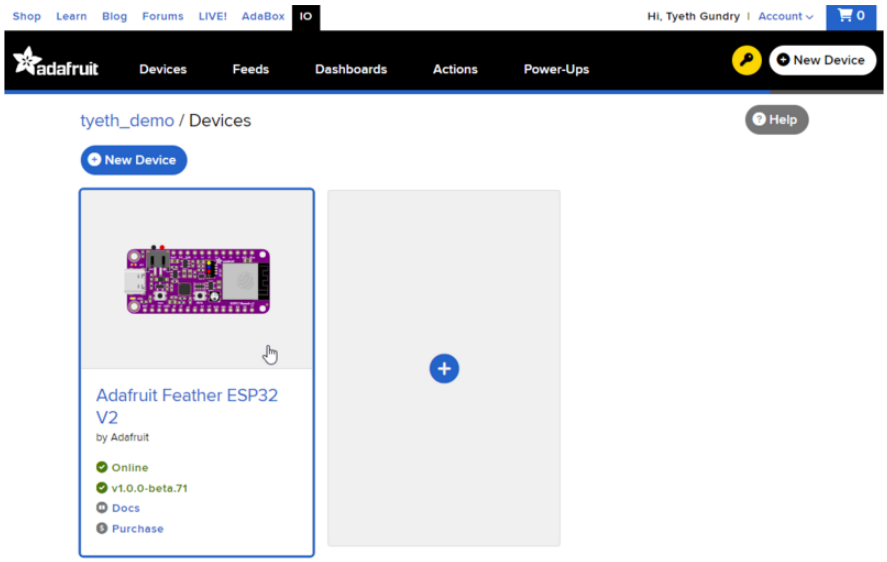


Board 3V to **sensor VIN (red wire on STEMMA QT)**
Board GND to **sensor GND (black wire on STEMMA QT)**
Board SCL to **sensor SCL (yellow wire on STEMMA QT)**
Board SDA to **sensor SDA (blue wire on STEMMA QT)**

Usage

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

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



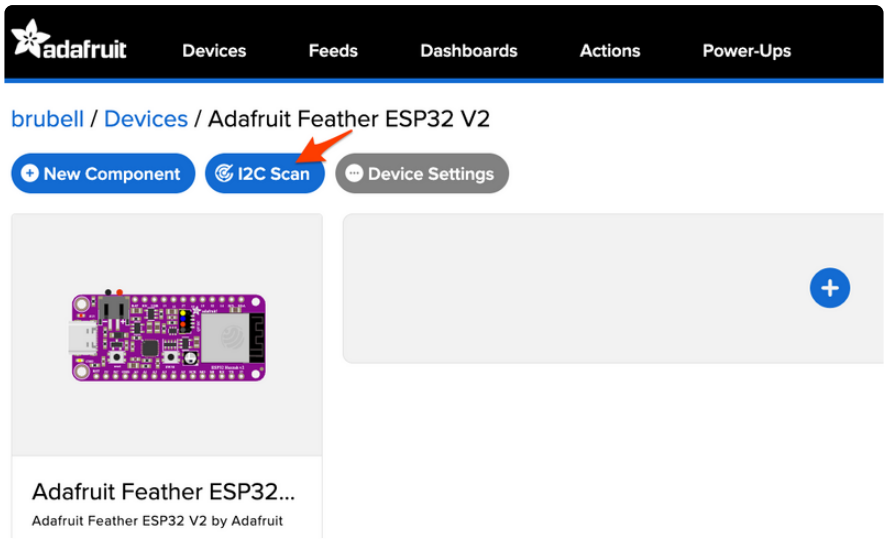
If you do not see your board listed here - you need to connect your board to Adafruit IO (https://adafru.it/Vfd) first.

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

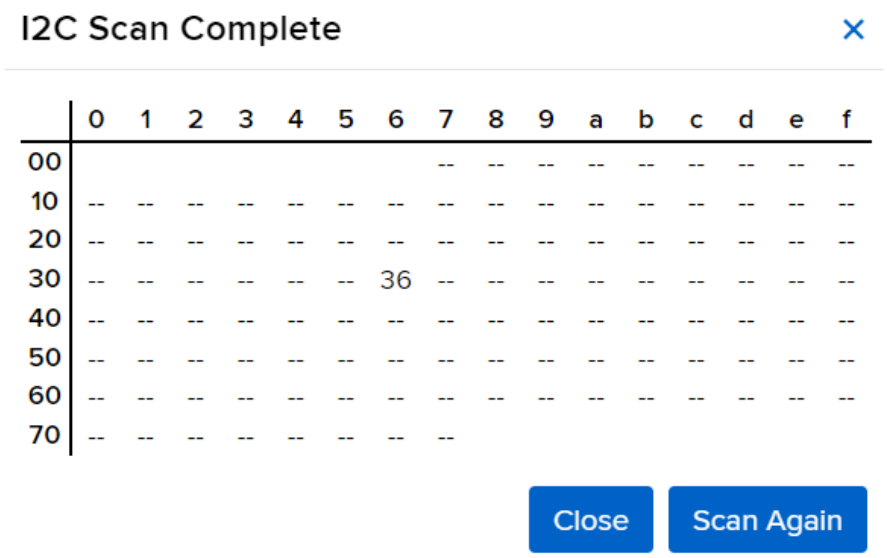
The device tile on the left indicates the version number of the firmware running on the connected board.

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) 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 MAX17048's default I2C address of `0x36` pop-up in the I2C scan list.



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

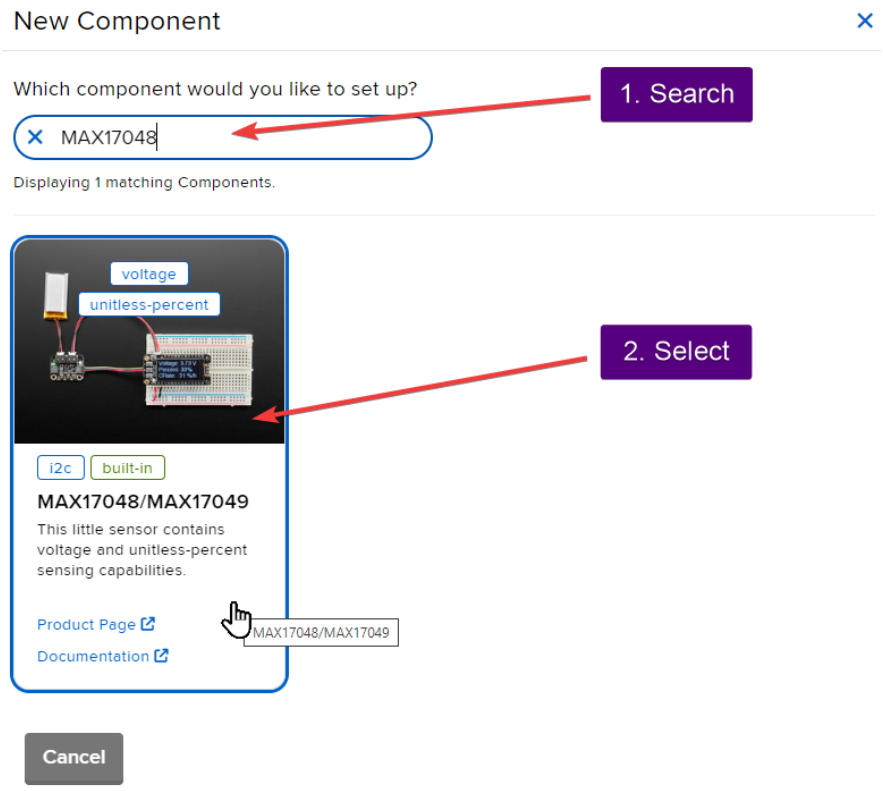
First, double-check the connection and/or wiring between the sensor and the board.

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.

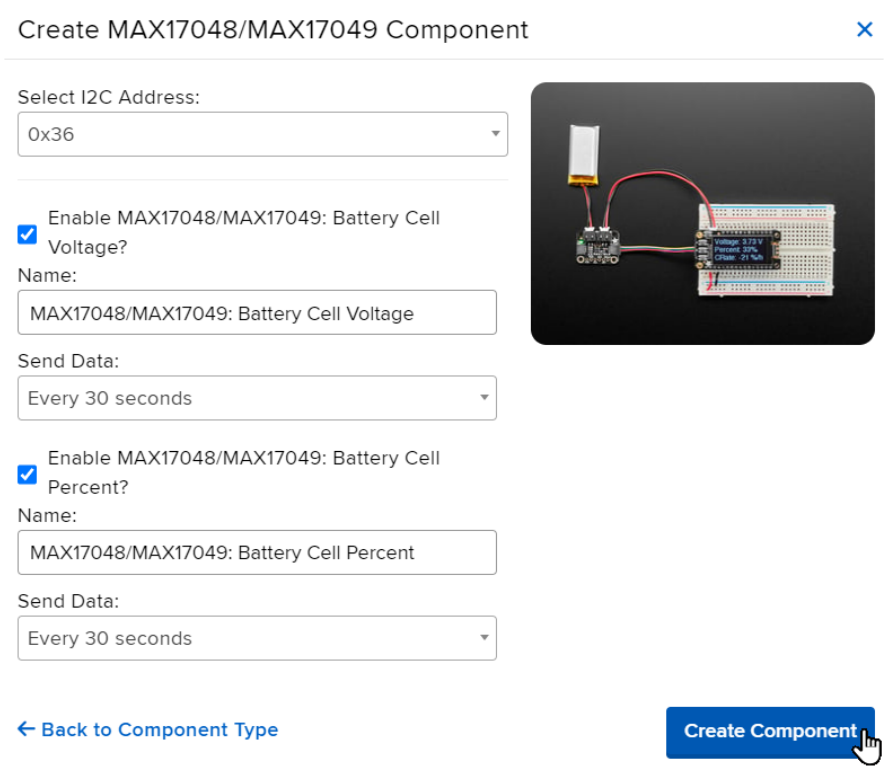
Adafruit IO supports a large amount of components. To quickly find your sensor, type `MAX17048` into the search bar, then select the **MAX17048** component.



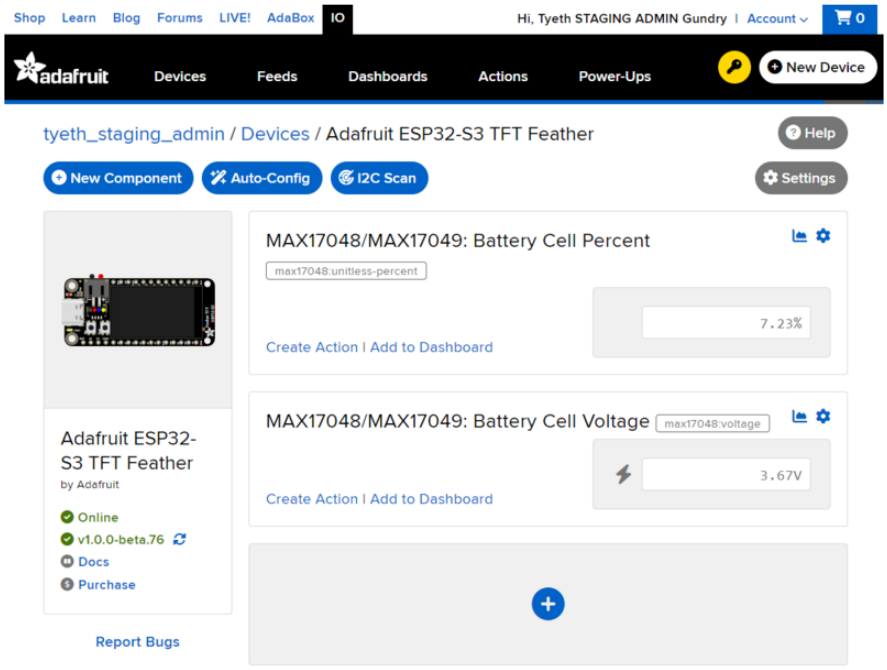
On the component configuration page, the MAX17048's sensor address should be listed along with the sensor's settings.

The **Send Every** option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the MAX17048 sensor and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

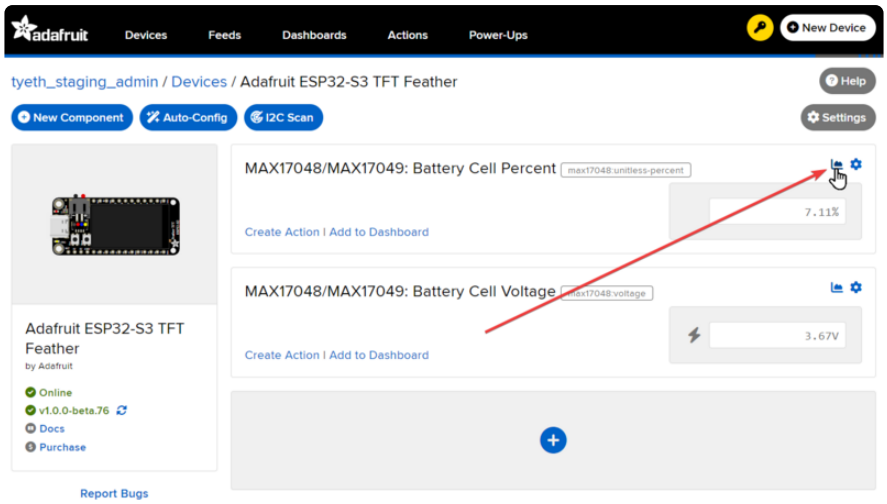
For this example, set the **Send Every** interval to every 30 seconds.



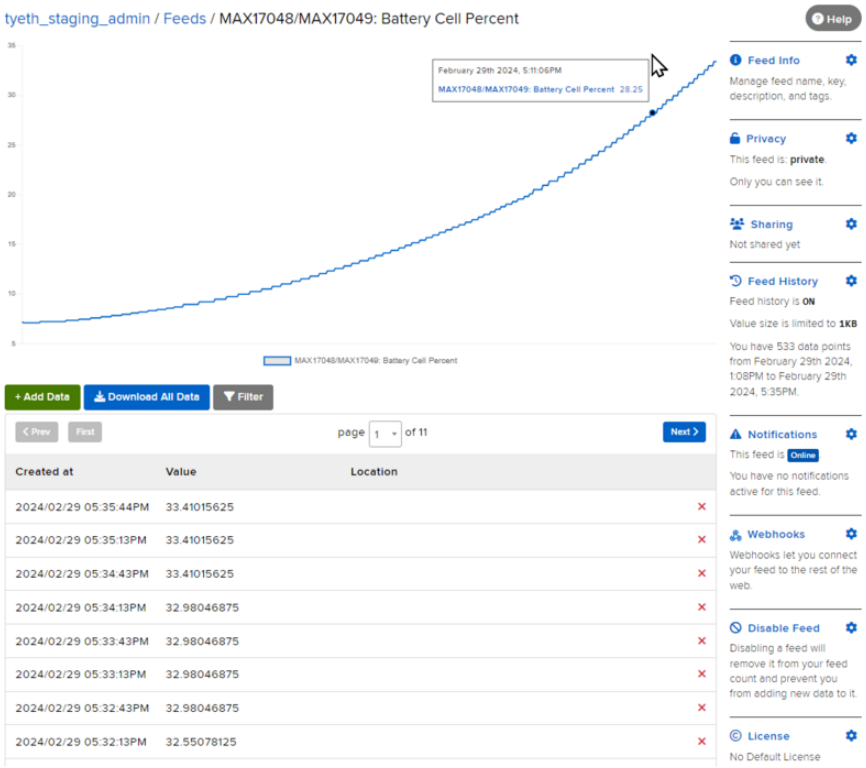
Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.



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).

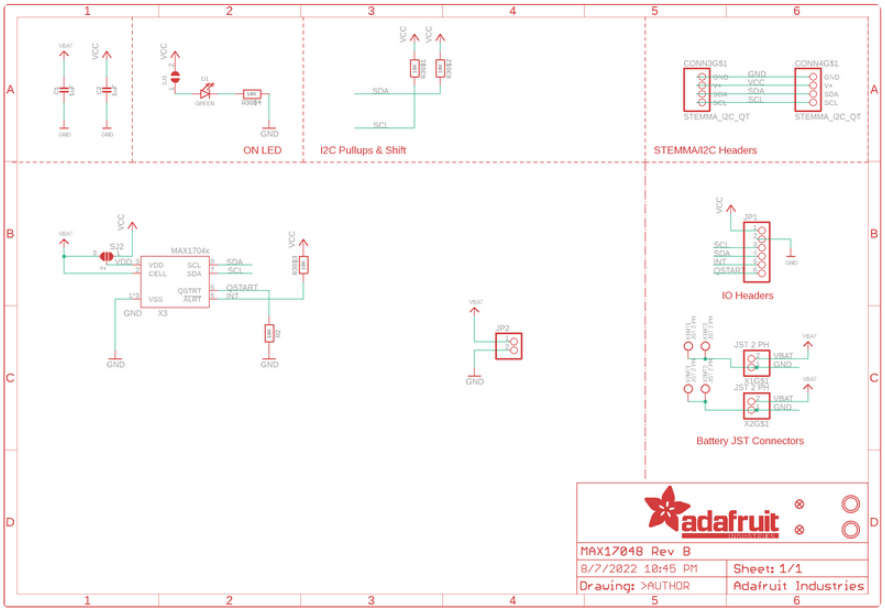


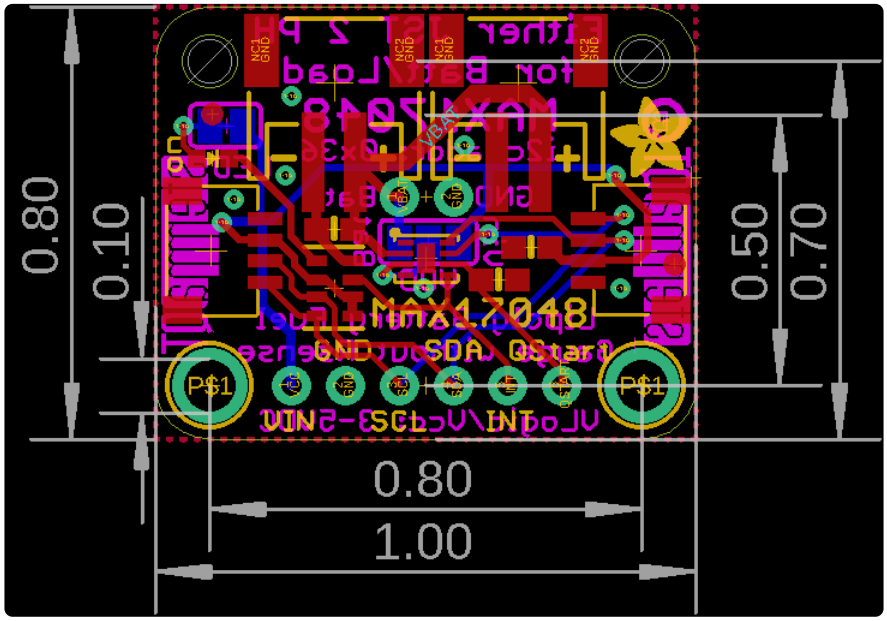
Downloads

Files

- MAX17048 Datasheet (https://adafru.it/10RB)
- EagleCAD PCB files on GitHub (https://adafru.it/10RC)
- 3D models on GitHub (https://adafru.it/112c)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/10RD)

Schematic and Fab Print





3D Model