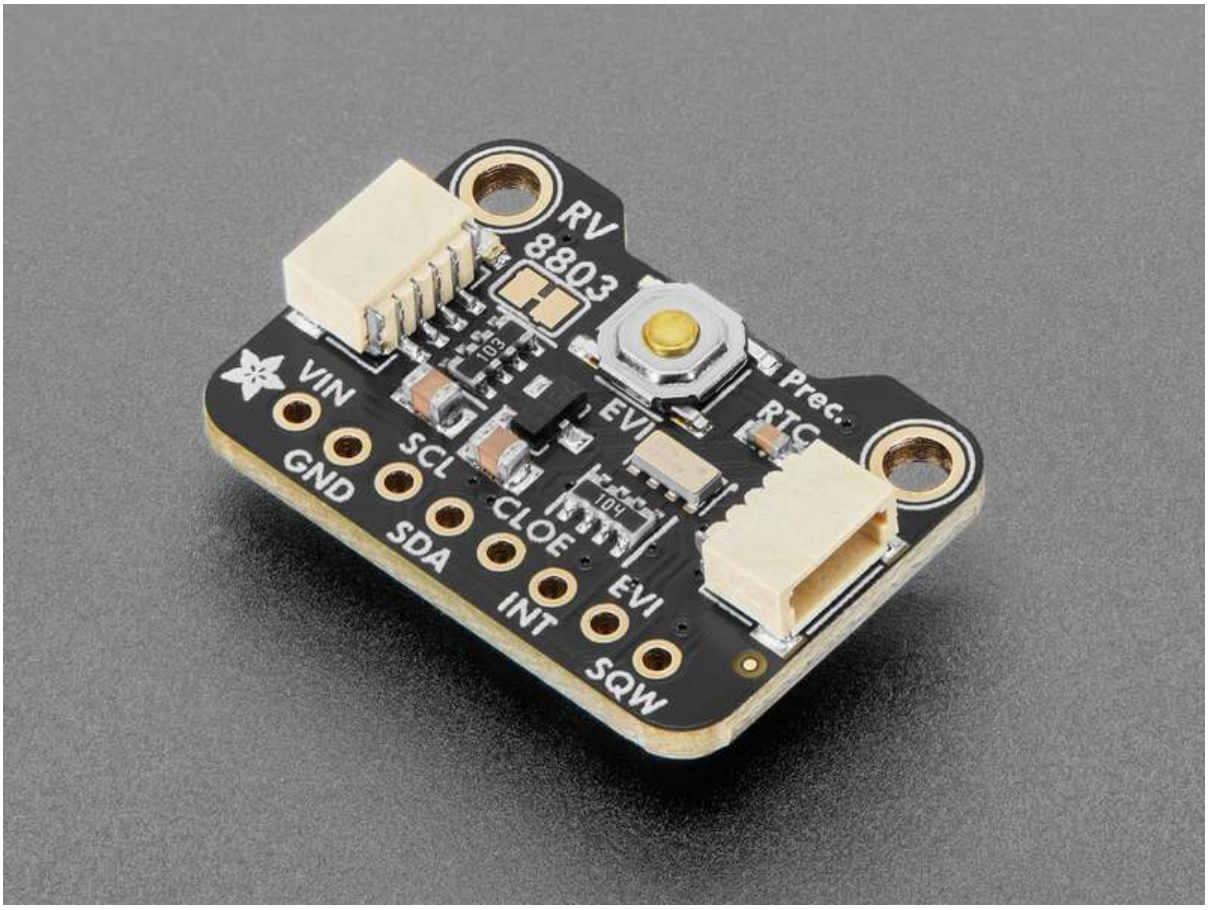
Adafruit RV-8803 High Accuracy Real Time Clock

Created by Liz Clark



https://learn.adafruit.com/adafruit-rv-8803-high-accuracy-real-time-clock

Last updated on 2026-11-09 08:49:51 PM UTC

Table of Contents

Overview 3

Pinouts 6

- Power Pins
- I2C Logic Pins
- Event Input Pin and Button
- Clock Pins
- Interrupt Pin
- Power LED and LED Jumper

CircuitPython and Python 7

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of the RV-8803 Library
- CircuitPython Usage
- Python Usage
- Example Code

Python Docs 12

Arduino 12

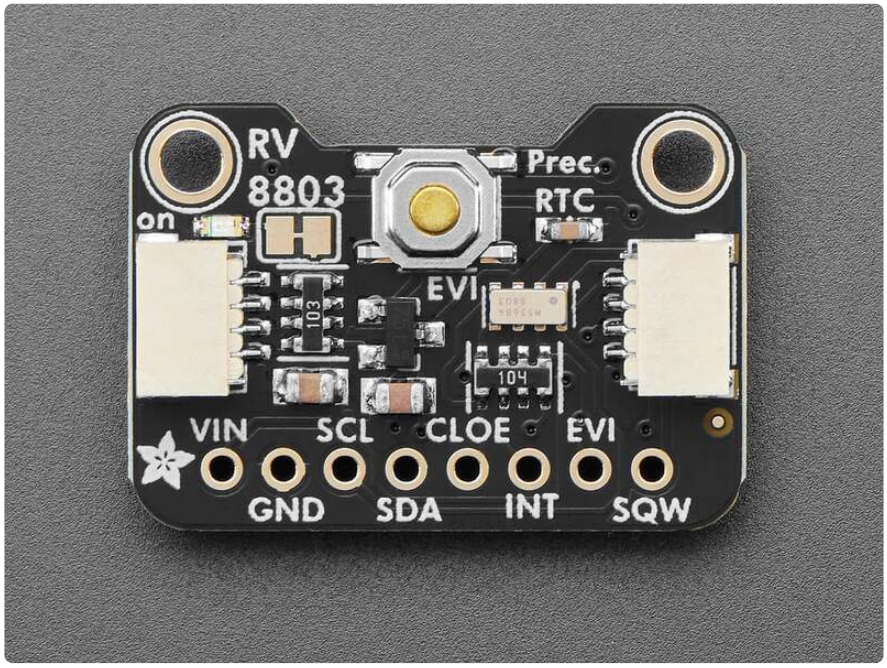
- Wiring
- Library Installation
- Example Code

Arduino Docs 16

Downloads 16

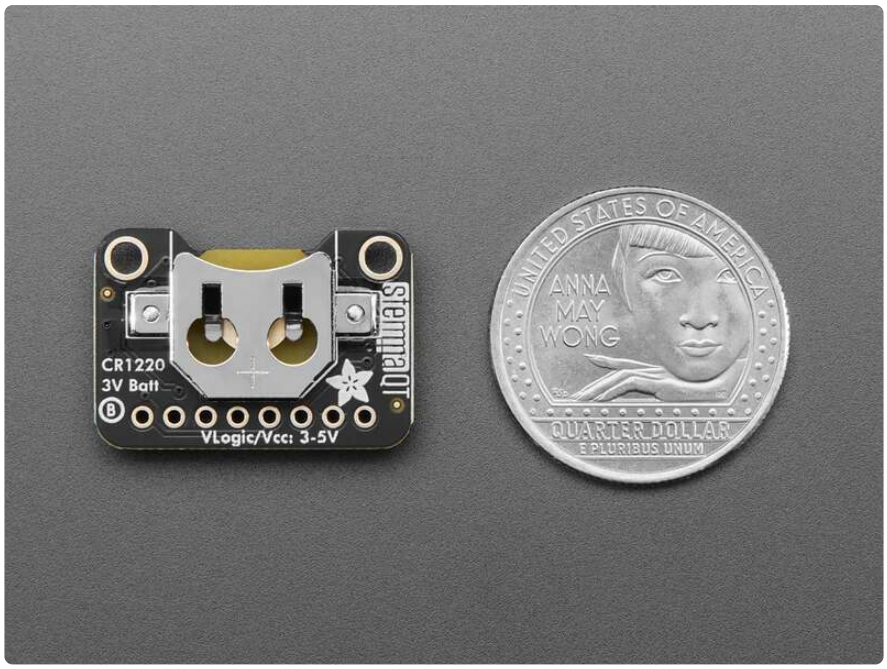
- Files
- Schematic and Fab Print

Overview



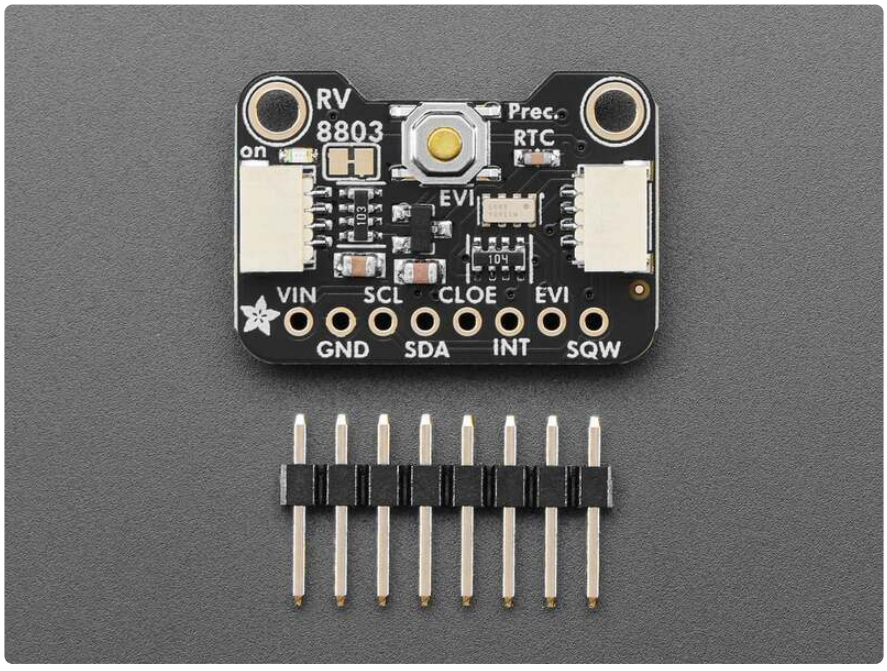
For excellent, accurate time-keeping - when your everyday RTC (Real Time Clock) won't do - the **Adafruit RV-8803 Precision Real Time Clock Breakout Board** is a tick-tock above the rest with excellent ±1.5 ppm (at 0~50°C) accuracy. Thanks to the built-in crystal with temperature compensation management, classic RTC drift is minimized, so you don't have to do any time correction with WiFi or GPS to keep current. Compare it to the popular DS3231 (), with ±2 ppm (0~40°C), you get even tighter spec'd performance, at a better price!

Text emphasized with a yellow exclamation:	Note: This product does not come with a CR1220 coin cell battery. We recommend you purchase a coin cell battery to use with this product.



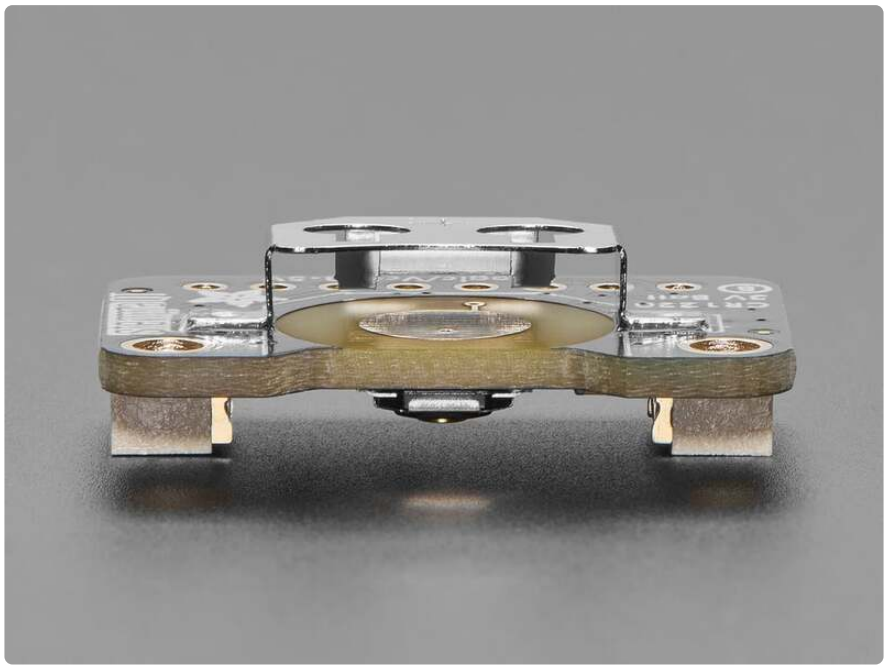
This is a great battery-backed real-time clock (RTC) that allows your microcontroller project to keep track of time, even if it is reprogrammed, or if the power is lost. Perfect for data-logging, clock-building, time stamping, timers, and alarms, etc. Equipped with **RV-8803** RTC - it can run from 3.3V or 5V power & logic - and keep excellent time with under 50 seconds of drift over a whole year.

There's also some nifty extras like square-wave out, timer alarm, event timestamping (you can use the built-in button to activate an event, or use the breakout pads).



This board works great with an Arduino using our RTC library (), with CircuitPython (), or with a Raspberry Pi (or similar single board Linux computer), thanks to built-in kernel support.

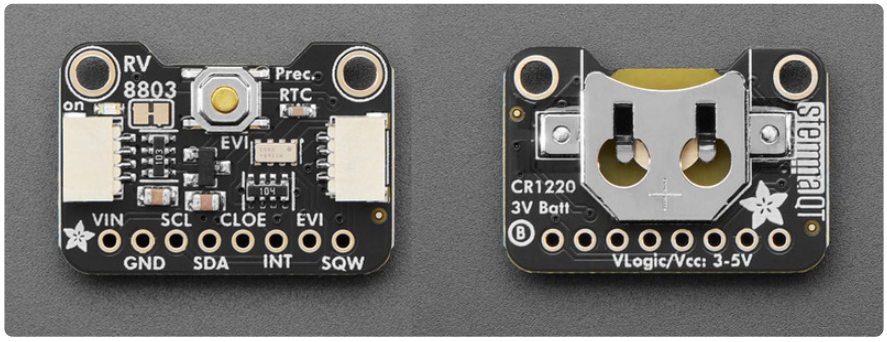
Note: This product does not come with a CR1220 coin cell battery. We recommend you purchase a coin cell battery to use with this product. (http://adafru.it/380) The battery is required in order to keep time when power is lost, and will last at least 5 years before needing to be changed.



We've taken the RTC and put it onto a breakout PCB along with support circuitry to let you use it with 3.3V (Feather/Raspberry Pi) or 5V (Arduino/ Metro328) logic levels. Additionally, since it speaks I2C, you can easily connect it up with two wires (plus power and ground!). We've even included SparkFun qwiic () compatible **STEMMA QT** () connectors for the I2C bus so **you don't even need to solder! QT Cable is not included**, but we have a variety in the shop (). Just wire up to your favorite micro and you can use our CircuitPython/Python or Arduino drivers () to easily interface with the RV-8803.

This comes as a fully assembled and tested breakout plus a small piece of header. You can solder header in to plug it into a breadboard, or solder wires directly.

Pinouts



The default I2C address is **0x32**.

Power Pins

- **VIN** - this is the power pin. Since the RTC can be powered from 1.5V to 5.5V power, you do not need a regulator or level shifter for 3.3V or 5V logic/power. 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

Text emphasized with a yellow exclamation:	Note: This product does not come with a CR1220 coin cell battery. We recommend you purchase a coin cell battery to use with this product.

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** () - These connectors allow you to connectors to dev boards with **STEMMA QT** connectors or to other things with various associated accessories ().

Event Input Pin and Button

- **EVI** - This is the event input pin. This pin is broken out along the edge of the board. The button in the center of the board is also connected to the pin. Both are labeled **EVI** on the board silk. You can press the button or toggle the pin to **GND** to trigger an event interrupt or time stamp.

Clock Pins

- **CLOE** - This is the clock enable pin. When this pin is high, the clock output pin (**SQW**) is in output mode. When it is low, the clock output is stopped and is high impedance/tri-state.
- **SQW** - This is the clock output pin. When the clock enable pin (**CLOE**) is high, **SQW** drives a square wave of 32.768 kHz, 1.024 kHz or 1 Hz (32.768 kHz is the default). When **CLOE** is low, the square wave is stopped and the pin is high impedance/tri-state.

Interrupt Pin

- **INT** - This is the interrupt pin. It is an open-drain output. It is used to output alarm, periodic countdown timer, periodic time update and external event interrupt signals.

Power LED and LED Jumper

- **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 a green LED.
- **LED jumper** - This jumper is located on the front of the board, to the right of the power LED. Cut the trace on this jumper to cut power to the "on" LED.

CircuitPython and Python

It's easy to use the **RV-8803** with Python or CircuitPython, and the Adafruit_CircuitPython_RV8803 () module. This module allows you to easily write Python code to read data from the sensor.

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 ().

CircuitPython Microcontroller Wiring

First wire up a RV-8803 to your board exactly as shown below. Here's an example of wiring a Feather RP2040 to the RTC with I2C using one of the handy **STEMMA QT** () connectors:

- **Board STEMMA 3V** to **RTC VIN (red wire)**
- **Board STEMMA GND** to **RTC GND (black wire)**
- **Board STEMMA SCL** to **RTC SCL (yellow wire)**
- **Board STEMMA SDA** to **RTC SDA (blue wire)**

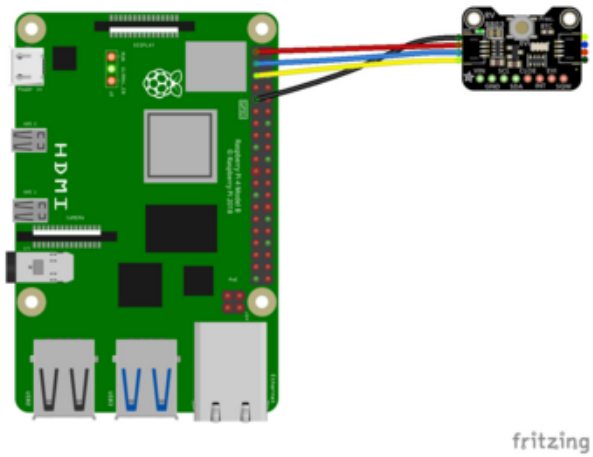
The following is the RV-8803 wired to a Feather RP2040 using a solderless breadboard:

- **Board 3V** to **RTC VIN (red wire)**
- **Board GND** to **RTC GND (black wire)**
- **Board SCL** to **RTC SCL (yellow wire)**
- **Board SDA** to **RTC SDA (blue wire)**

Python Computer Wiring

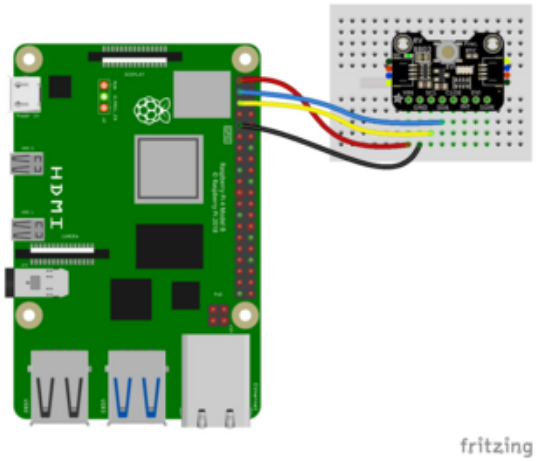
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported]() ().

Here's the Raspberry Pi wired with I2C using the STEMMA connector:



Pi 3V to **RTC VIN (red wire)**
Pi GND to **RTC GND (black wire)**
Pi SCL to **RTC SCL (yellow wire)**
Pi SDA to **RTC SDA (blue wire)**

Here's the Raspberry Pi wired with I2C using a solderless breadboard:



Pi 3V to **RTC VIN (red wire)**
Pi GND to **RTC GND (black wire)**
Pi SCL to **RTC SCL (yellow wire)**
Pi SDA to **RTC SDA (blue wire)**

Python Installation of the RV-8803 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 ()!

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

- `pip3 install adafruit-circuitpython-rv8803`

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_RV8803** 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_rv8803.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 () to see the data printed out!

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

```
# SPDX-FileCopyrightText: Copyright (c) 2026 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time

import board

import adafruit_rv8803

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a microcontroller
rtc = adafruit_rv8803.RV8803(i2c)

# Lookup table for names of days (nicer printing).
days = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

```

```
18
19  if False:  # change to True if you want to set the time!
20      #                year, mon, date, hour, min, sec, wday, yday, isdst
21      t = time.struct_time((2026, 10, 29, 11, 27, 0, 3, -1, -1))
22      # you must set year, mon, date, hour, min, sec and weekday
23      # yearday is not supported, isdst can be set but we don't do anything with it at this time
24      print("Setting time to:", t)  # uncomment for debugging
25      rtc.datetime = t
26      print()
27
28  # Main loop:
29  while True:
30      t = rtc.datetime
31      # print(t)     # uncomment for debugging
32      print(f"The date is {days[int(t.tm_wday)]} {t.tm_mday}/{t.tm_mon}/{t.tm_year}")
33      print(f"The time is {t.tm_hour}:{t.tm_min:02}:{t.tm_sec:02}")
34      time.sleep(1)  # wait a second
```

https://github.com/adafruit/Adafruit_CircuitPython_RV8803/blob/main/examples/rv8803_simpletest.py

In the main loop the date and time from the RTC are printed to the serial monitor every second.

```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
The date is Thursday 29/10/2026
The time is 17:26:22
The date is Thursday 29/10/2026
The time is 17:26:23
The date is Thursday 29/10/2026
The time is 17:26:24
The date is Thursday 29/10/2026
The time is 17:26:25
```

Set the Clock

Before the loop, you can change `if False:` to `if True:` to reset the date and time stored on the RTC. After running it once, set it back to `False` so that the RTC won't continually reset itself.

```
1  if False:  # change to True if you want to set the time!
2      #                year, mon, date, hour, min, sec, wday, yday, isdst
3      t = time.struct_time((2026, 10, 29, 11, 27, 0, 3, -1, -1))
4      # you must set year, mon, date, hour, min, sec and weekday
5      # yearday is not supported, isdst can be set but we don't do anything with it at this time
6      print("Setting time to:", t)  # uncomment for debugging
7      rtc.datetime = t
8      print()
```

Python Docs

Python Docs ()

Arduino

Using the RV-8803 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the Adafruit_RV8803 () 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 RTC using the STEMMA QT connector:

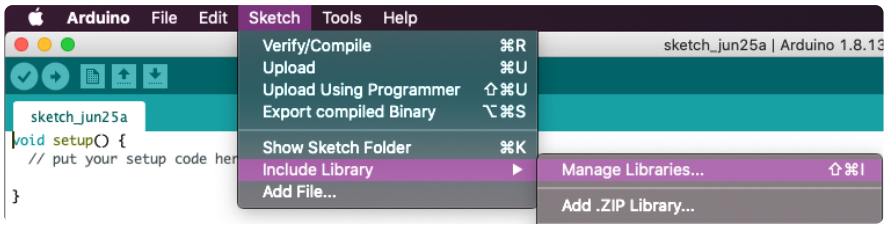
- **Board 5V** to **breakout VIN (red wire)**
- **Board GND** to **breakout GND (black wire)**
- **Board SCL** to **breakout SCL (yellow wire)**
- **Board SDA** to **breakout SDA (blue wire)**

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

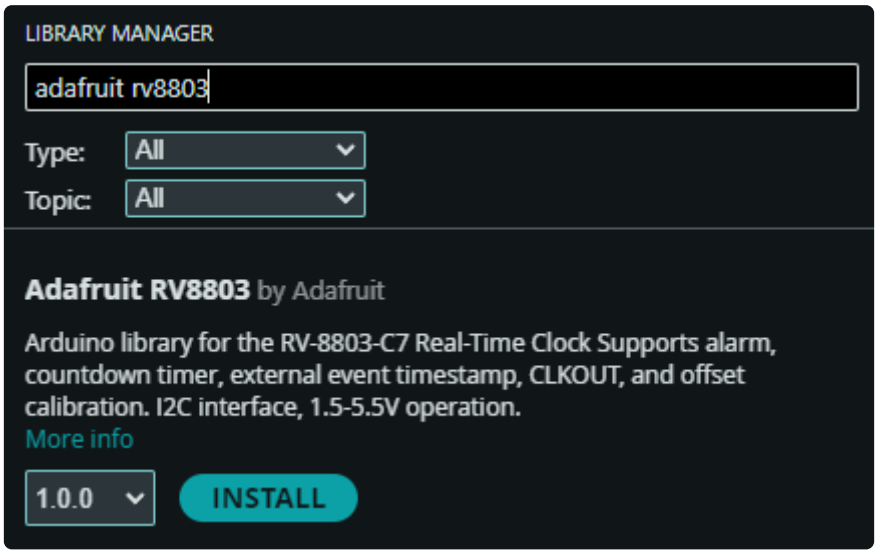
- **Board 5V** to **breakout VIN (red wire)**
- **Board GND** to **breakout GND (black wire)**
- **Board SCL** to **breakout SCL (yellow wire)**
- **Board SDA** to **breakout SDA (blue wire)**

Library Installation

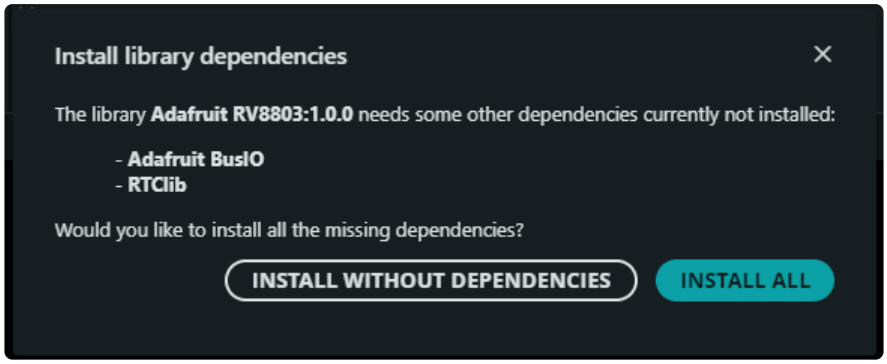
You can install the **Adafruit_RV8803** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit RV8803**, and select the **Adafruit RV8803** library:



If asked about dependencies, click "Install all".



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

Text emphasized with a yellow exclamation:

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

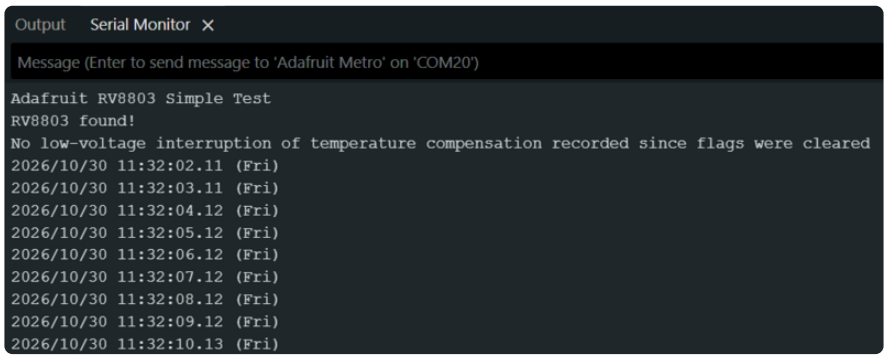
Example Code

```
/*!
 * @file simpletest.ino
 *
 * Simple example for the Adafruit RV8803 RTC library.
 * Reads and prints the time every second.
 *
 * Written by Limor 'ladyada' Fried with assistance from Claude Code
 */

#include <Adafruit_RV8803.h>

Adafruit_RV8803 rtc;

void setup() {
  Serial.begin(115200);
  while (!Serial)
    delay(10); // Wait for serial on native USB boards

  Serial.println(F("Adafruit RV8803 Simple Test"));

  if (!rtc.begin()) {
    Serial.println(F("Couldn't find RV8803"));
    while (1)
      delay(10);
  }
  Serial.println(F("RV8803 found!"));

  // Report retained status before setting the time clears the power flags.
  uint8_t flags = rtc.readFlagRegister();
  if (flags == RV8803_READ_ERROR) {
    Serial.println(F("Could not read temperature-compensation status"));
  } else if (flags & RV8803_FLAG_TEMP_COMP_STOPPED) {
    Serial.println(F("Low voltage interrupted temperature compensation; clock accuracy may have been
affected"));
  } else {
    Serial.println(F("No low-voltage interruption of temperature compensation recorded since flags were
cleared"));
  }

  // Check if RTC lost power and needs time set
  if (rtc.lostPower()) {
```

```
    Serial.println(F("RTC lost power, setting time to compile time"));
    // Set to compile time
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  // Clear any power flags
  rtc.clearPowerFlags();
}

void loop() {
  DateTime now = rtc.now();

  // Print date
  Serial.print(now.year(), DEC);
  Serial.print(F("/"));
  if (now.month() < 10)
    Serial.print(F("0"));
  Serial.print(now.month(), DEC);
  Serial.print(F("/"));
  if (now.day() < 10)
    Serial.print(F("0"));
  Serial.print(now.day(), DEC);

  Serial.print(F(" "));

  // Print time
  if (now.hour() < 10)
    Serial.print(F("0"));
  Serial.print(now.hour(), DEC);
  Serial.print(F(":"));
  if (now.minute() < 10)
    Serial.print(F("0"));
  Serial.print(now.minute(), DEC);
  Serial.print(F(":"));
  if (now.second() < 10)
    Serial.print(F("0"));
  Serial.print(now.second(), DEC);

  // Print hundredths
  Serial.print(F("."));
  uint8_t hundredths = rtc.getHundredths();
  if (hundredths < 10)
    Serial.print(F("0"));
  Serial.print(hundredths, DEC);

  // Print day of week
  const char* days[] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};
  Serial.print(F(" ("));
  Serial.print(days[now.dayOfTheWeek()]);
  Serial.println(F(")"));

  delay(1000);
}
```

https://github.com/adafruit/Adafruit_RV8803/blob/main/examples/simpletest/simpletest.ino

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the RV-8803 recognized over I2C. Then, it will list if any flags were cleared. In the loop, the timestamp is printed every second.



Arduino Docs

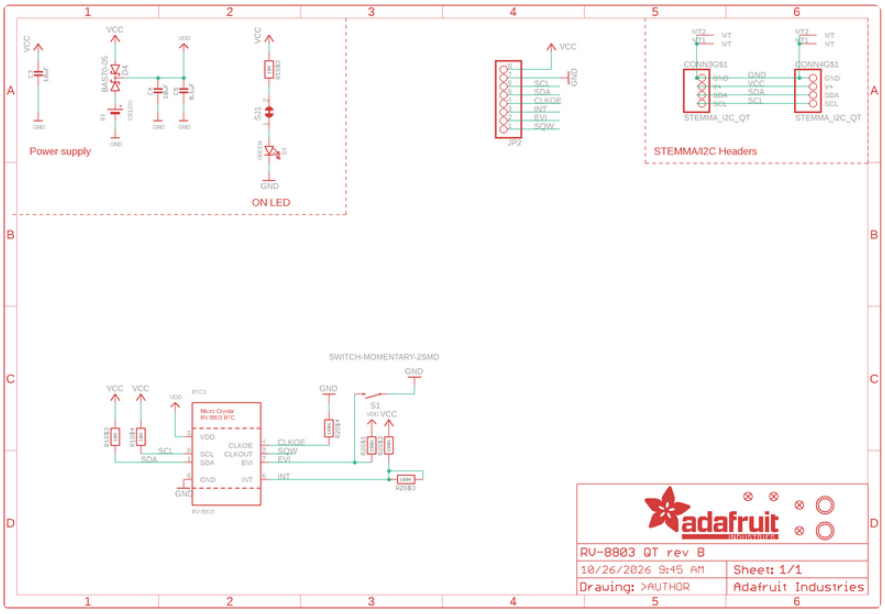
Arduino Docs ()

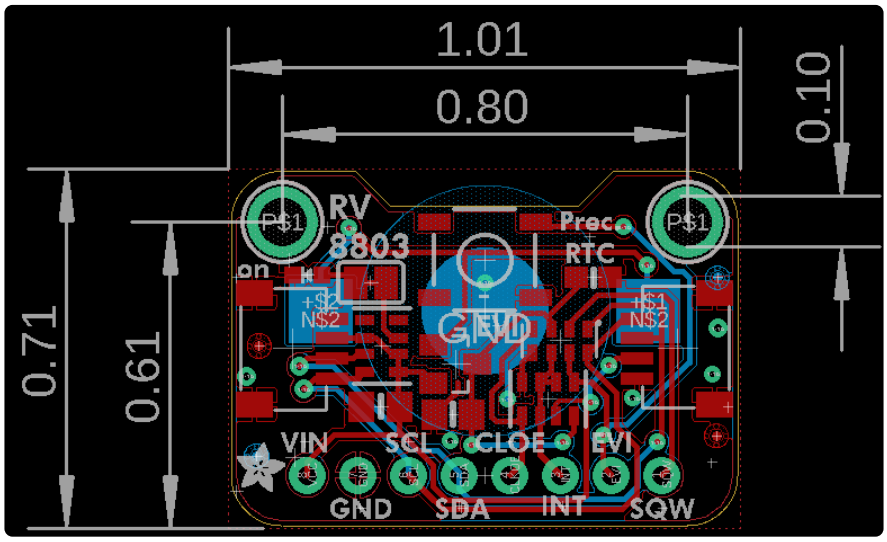
Downloads

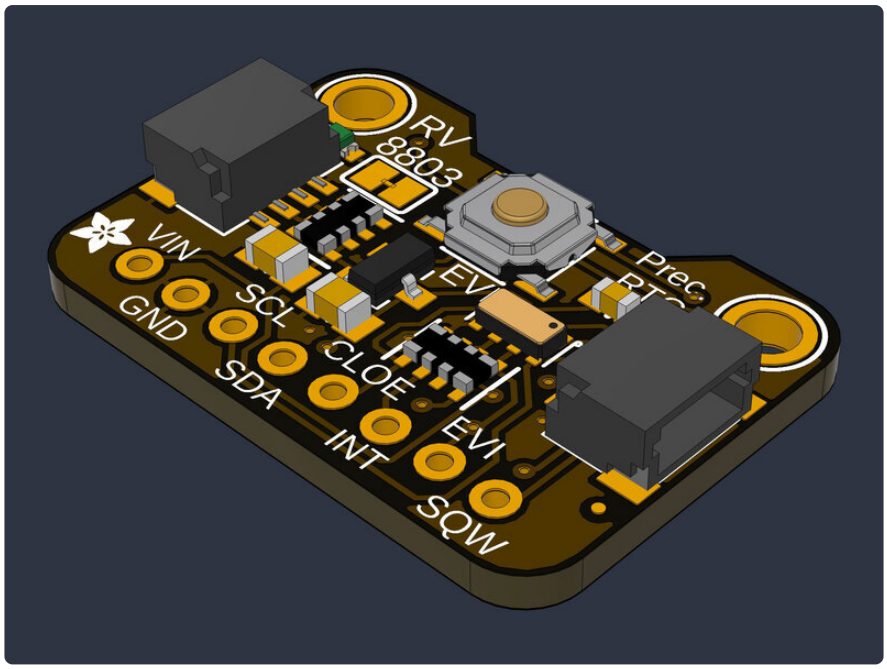
Files

- RV-8803 Datasheet ()
- RV-8803 Application Manual ()
- EagleCAD PCB files on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()
- 3D models on GitHub ()

Schematic and Fab Print






RV
8803
VIN
GND
SCL
SDA
CLOE
INT
EVI
SQW
EVI
Prec