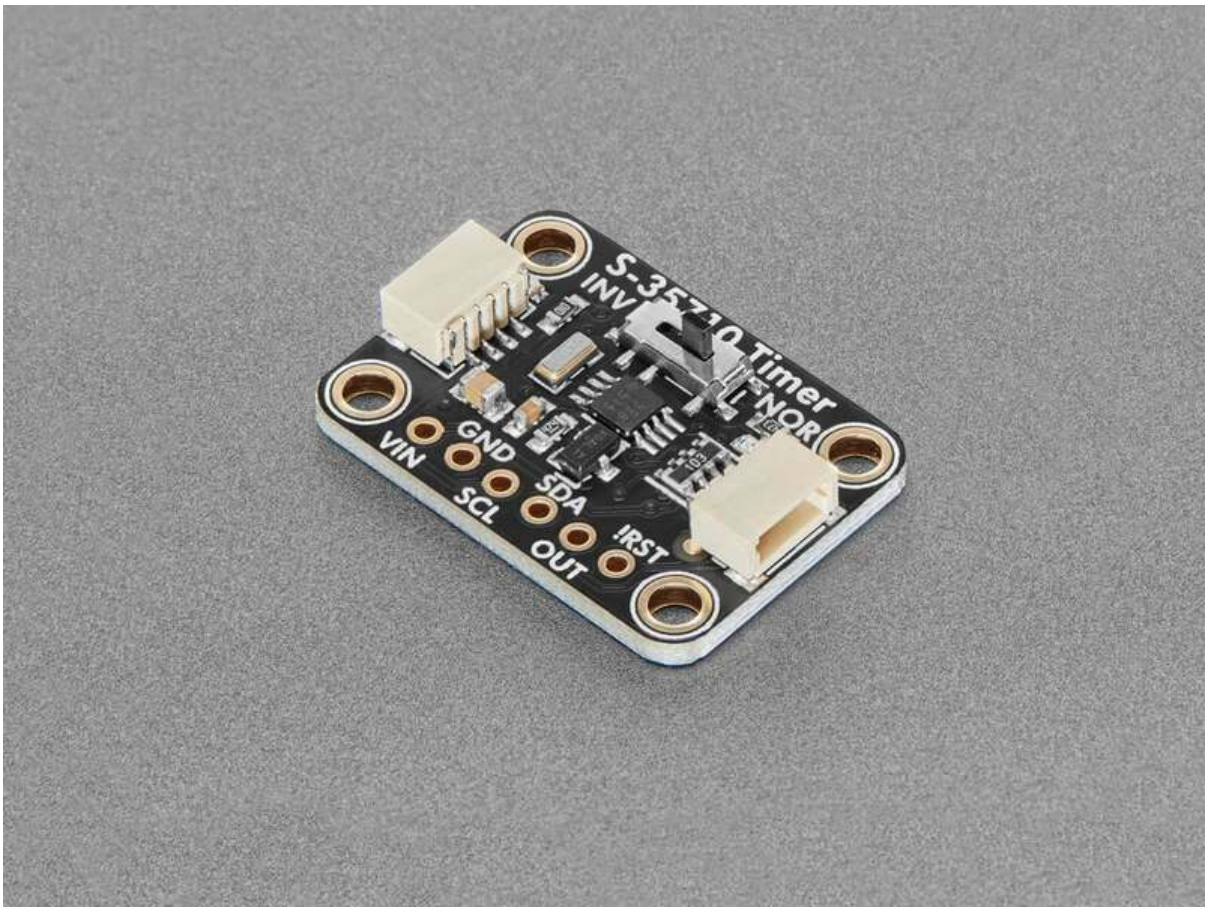
# Adafruit S-35710 Low-Power Wake Up Timer Breakout

Created by Liz Clark

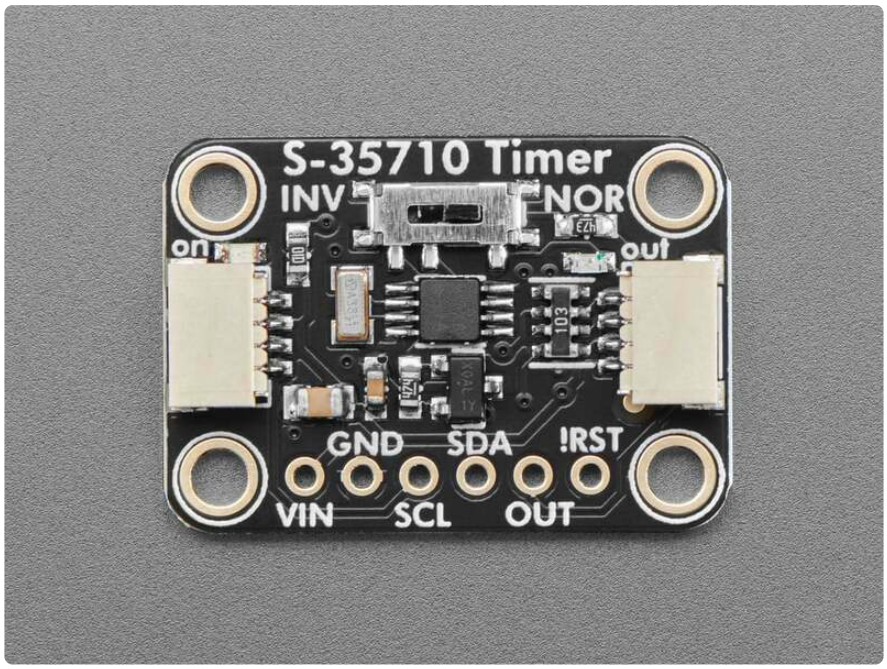



https://learn.adafruit.com/adafruit-s-35710-low-power-wake-up-timer-breakout

Last updated on 2024-06-11 12:12:46 PM EDT

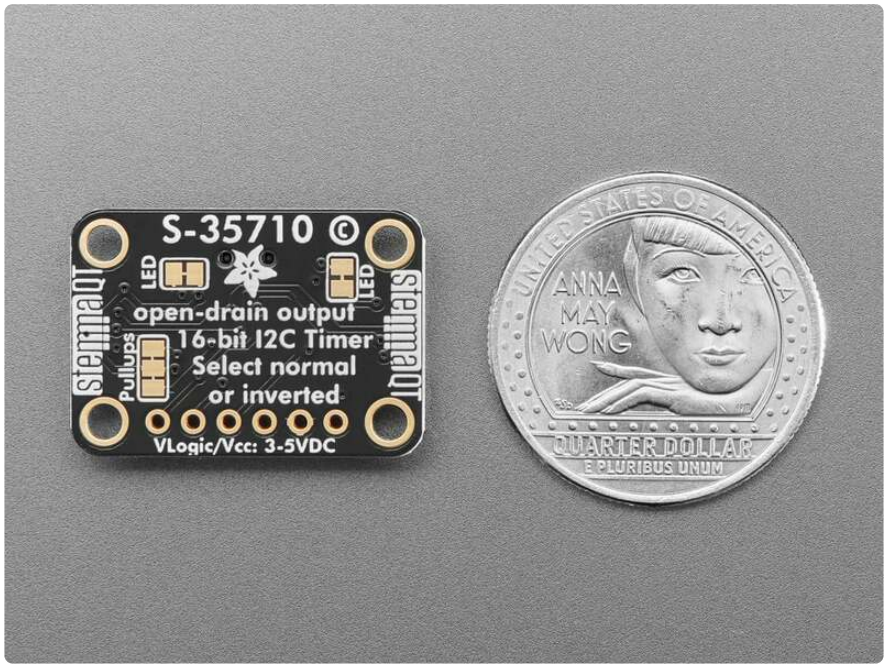
# Table of Contents

# Overview



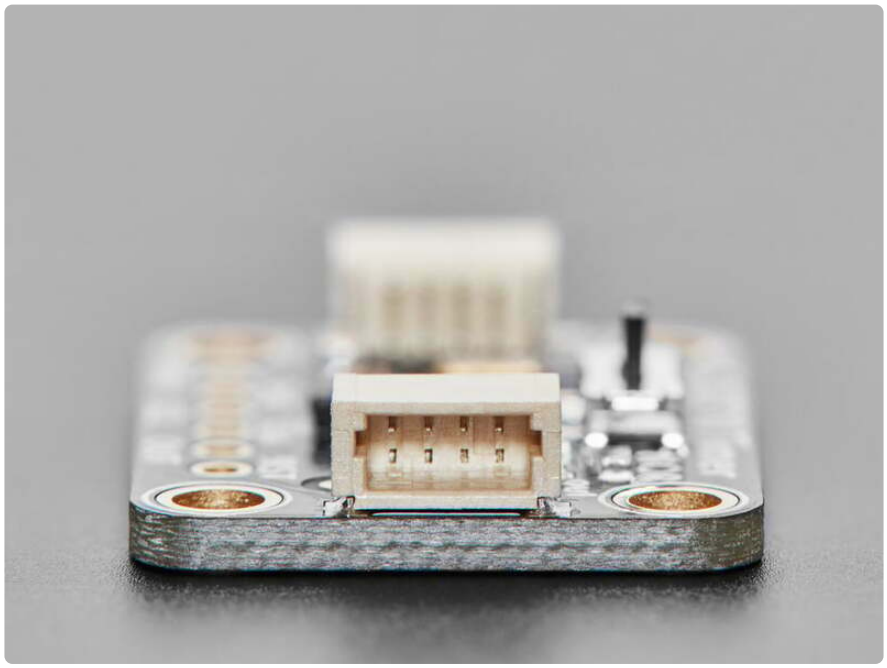

The **Adafruit S-35710 Wake Up Timer** is a low power 'watchdog timer' chip that can be programmed to alert with a digitally-configurable alarm from 1 second up to 194 days, thanks to a 24-bit seconds counter.

It's an interesting alternative to a real time clock or internal sleep timer and might be useful for some ultra-low-power projects that want to have a separate (and possibly separately-powered) chip to handle time-keeping and alarms. We covered this component on EYE ON NPI and thought it would make for a nice breakout in the shop (https://adafru.it/1a0c).



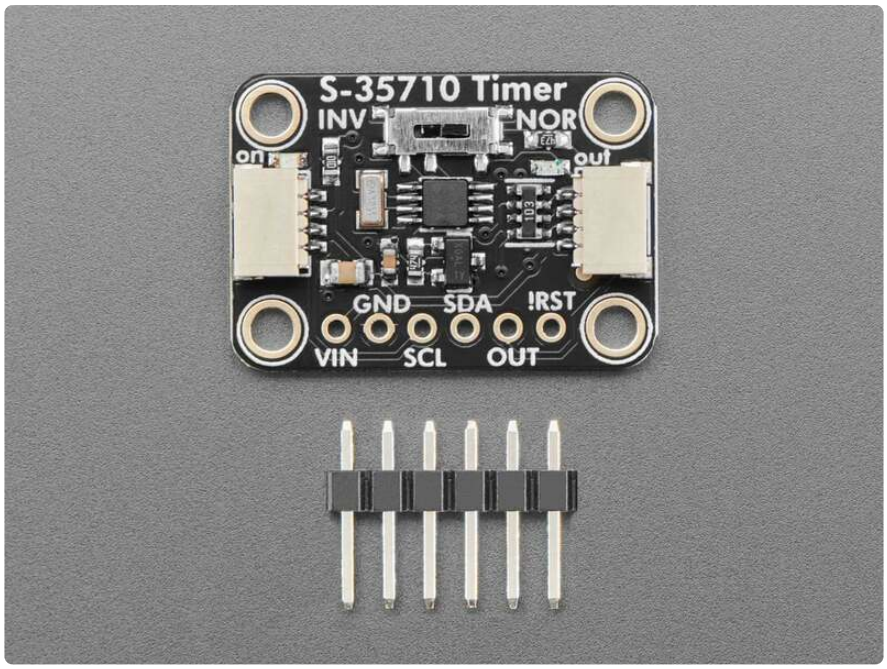
Usage is pretty simple: on I2C write of the timer register, the chip starts counting from 0 seconds up and the OUT pin goes high. When the count matches the timer register, the OUT pin goes low.

Simple, and like we mentioned, it's very low power, drawing only 0.2uA. The output is open-drain, and if you want to have it be inverted, there's a switch that will put the signal through an N-channel FET so the polarity goes the other way.



One thing to note is that on power-up the OUT line is default low, which means its not good for a low-power sleep manager, only for watch-dog-like timings. That is to say, you cannot connect this to your microcontroller's reset or enable line to self-depower because it will be disabled/reset when powered up.

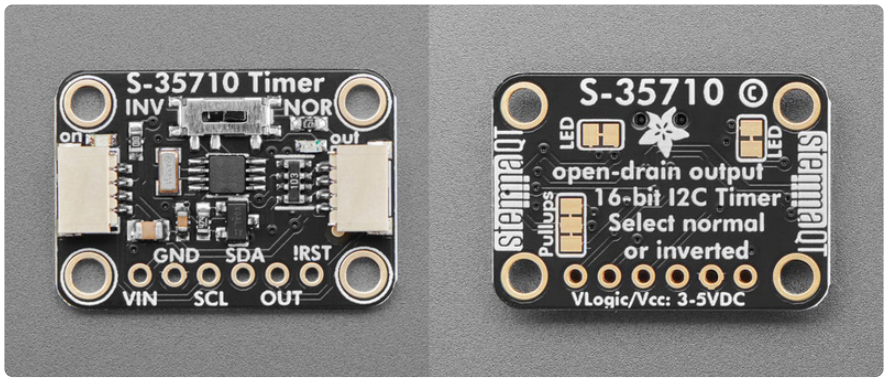
Still, maybe a useful chip for some folks who want to signal after a long delay or have a more complex power management scheme.




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 S-35710 or to chain it with a wide range of other sensors and accessories using a **compatible cable** (https://adafru.it/JnB).

---

# Pinouts




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

## Power Pins

- **VIN** - this is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V.
- **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 to dev boards with **STEMMA QT** (Qwiic) connectors or to other things with various associated accessories (https://adafru.it/Ft6).

## Other Pins

- **OUT** - This is the output pin from the S-35710. When the S-35710 is actively counting, the pin is high. When the count matches the timer register, the pin goes low. One thing to note, is that on power-up **OUT** is default low, which means it's not good for a low-power sleep manager, only for watch-dog-like timings.
- **!RST** - This is the reset pin. If this pin is pulled low, it will reset the timer.

## Output Inverter Switch

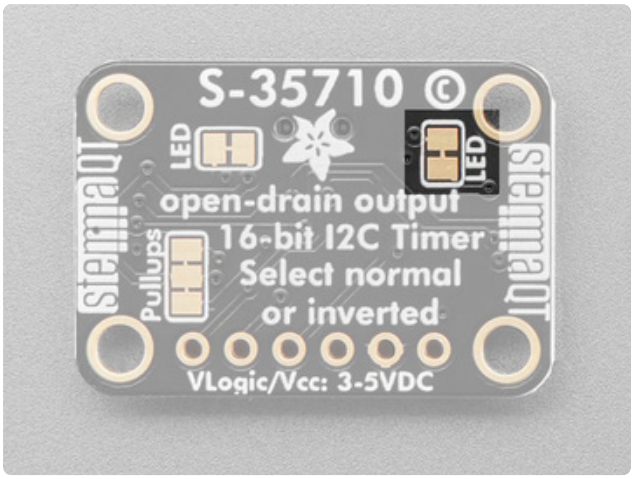
At the top of the board is a slide switch. This switch determines whether the **OUT** pin is the default open-drain (high when counting, low otherwise) or open-collector (low when counting, high otherwise).

To keep the **OUT** as the default, select **NOR** on the switch for normal. To change the **OUT** to inverted (still open-drain), select **INV** for inverted on the switch.

## I2C Pullup Jumpers

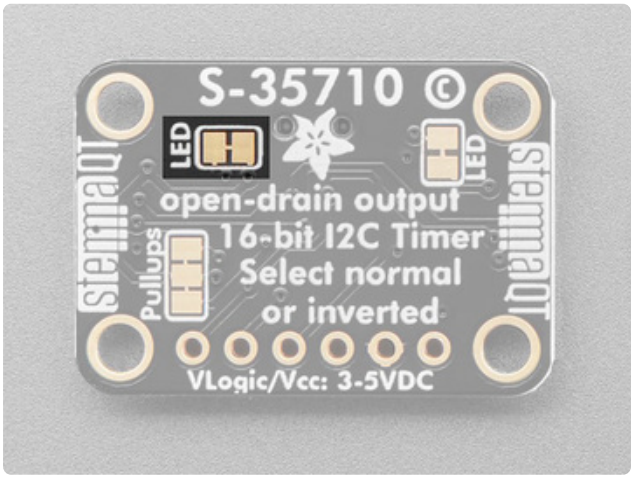
On the back of the board are three connected jumper pads labeled **Pullups** on the board silk. These jumpers connect the 10K pullups to the SDA and SCL lines. Cut these jumpers if you want to disconnect the 10K pullups from the I2C clock and data pins.

## 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 back of the board, to the right of the Adafruit logo on the board silk. Cut the trace on this jumper to cut power to the "**on**" LED.

## Output LED and LED Jumper




**Output LED -** In the upper right corner, behind the STEMMA connector, on the front of the board, is the output LED, labeled **out**. It is a red LED. This LED will be **on when the OUT pin is low** and **off when the OUT pin is high**.
**LED jumper** - This jumper is located on the back of the board, to the left of the Adafruit logo on the board silk. Cut the trace on this jumper to cut power to the "**out**" LED.

---

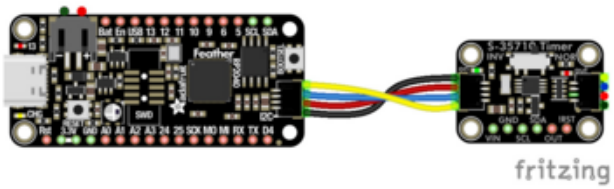
# CircuitPython & Python

It's easy to use the **S-35710** with Python or CircuitPython, and the Adafruit_CircuitPython_S35710 (https://adafru.it/1a2m) module. This module allows you to easily write Python code to set and monitor the timer alarm.

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).

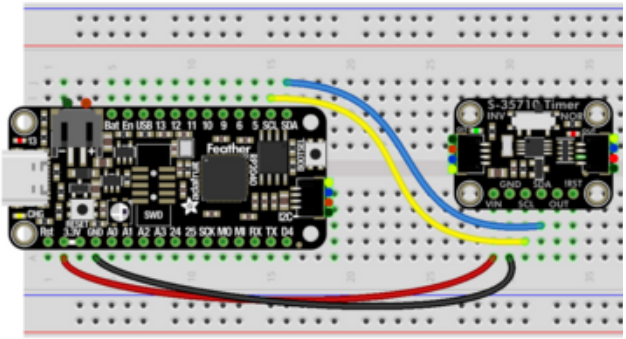
## CircuitPython Microcontroller Wiring

First wire up the breakout to your board exactly as follows. The following is the breakout wired to a Feather RP2040 using the STEMMA connector:




**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)**

The following is the breakout wired to a Feather RP2040 using a solderless breadboard:

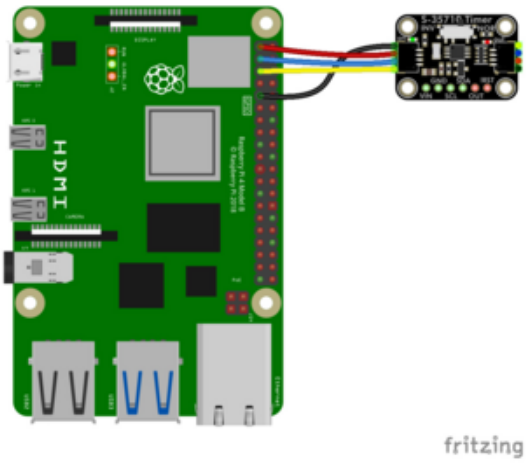


**Board 3V** to **breakout VIN (red wire)**
**Board GND** to **breakout GND (black wire)**
**Board SCL** to **breakout SCL (yellow wire)**
**Board SDA** to **breakout SDA (blue wire)**

## Python Computer Wiring

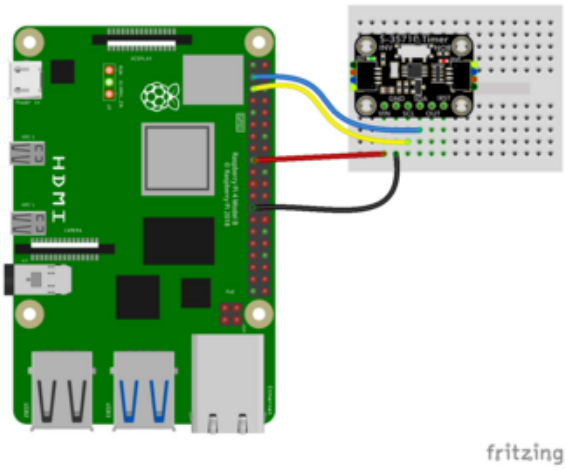
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

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




**Pi 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)**

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




**Pi 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)**

## Python Installation of S-35710 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-s35710`

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!

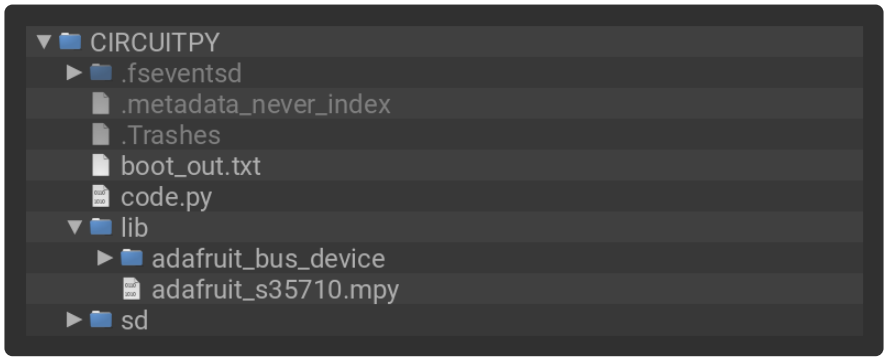
## CircuitPython Usage

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_S35710** 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_s35710.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
```

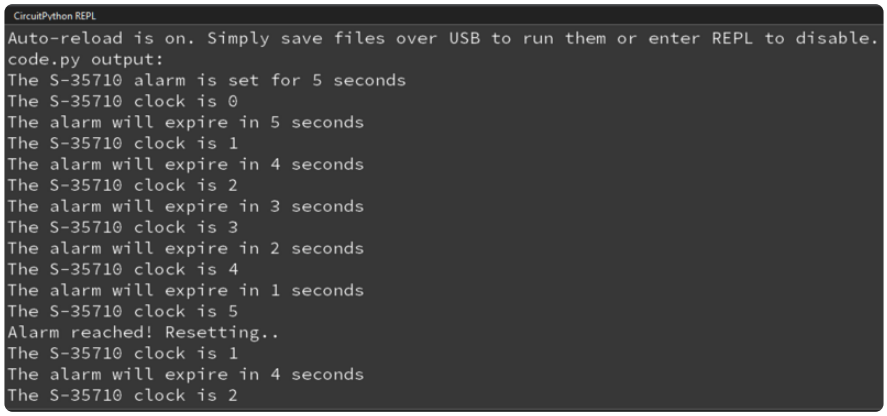
## Example Code

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

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

```
# SPDX-FileCopyrightText: Copyright (c) 2024 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_s35710

i2c = board.I2C()

timer = adafruit_s35710.Adafruit_S35710(i2c)

timer.alarm = 5
print(f"The S-35710 alarm is set for {timer.alarm} seconds")

countdown = timer.alarm - timer.clock

while True:
    print(f"The S-35710 clock is {timer.clock}")
    countdown = timer.alarm - timer.clock
    if countdown == 0:
        timer.alarm = 5
        print("Alarm reached! Resetting..")
    else:
        print(f"The alarm will expire in {countdown} seconds")
    time.sleep(1)
```

First, the timer is instantiated over I2C. The alarm is set for 5 seconds. Then, in the loop, the clock from the timer is printed to the serial monitor, followed by the time remaining until the alarm ends. When the alarm is reached, it is reset to begin again for 5 seconds.




# Python Docs

Python Docs (https://adafru.it/1a03)

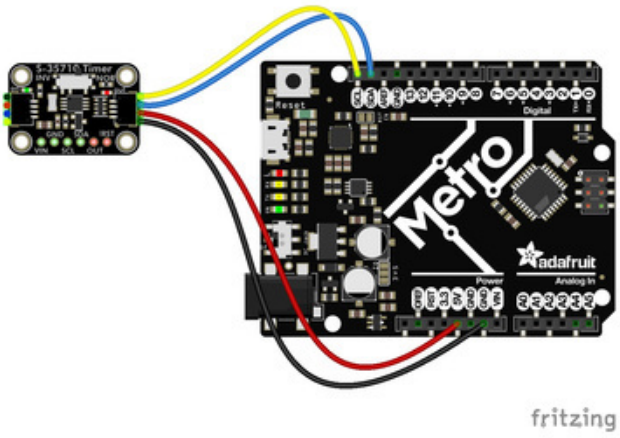
# Arduino

Using the S-35710 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the Adafruit_S-35710 (https://adafru.it/1a0e) library, and running the provided example code.

## Wiring

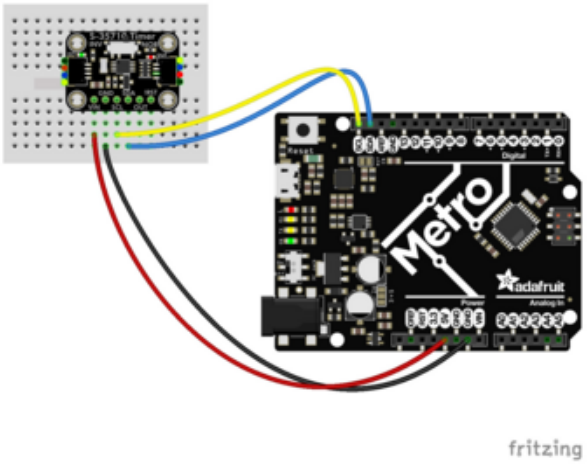
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the breakout VIN.

Here is an Adafruit Metro wired up to the breakout 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)**

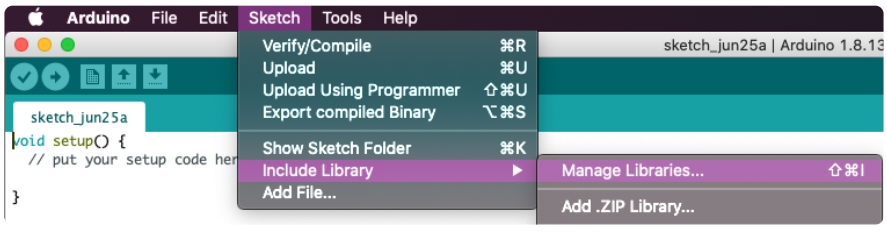
Here is an Adafruit Metro wired up using a solderless breadboard:



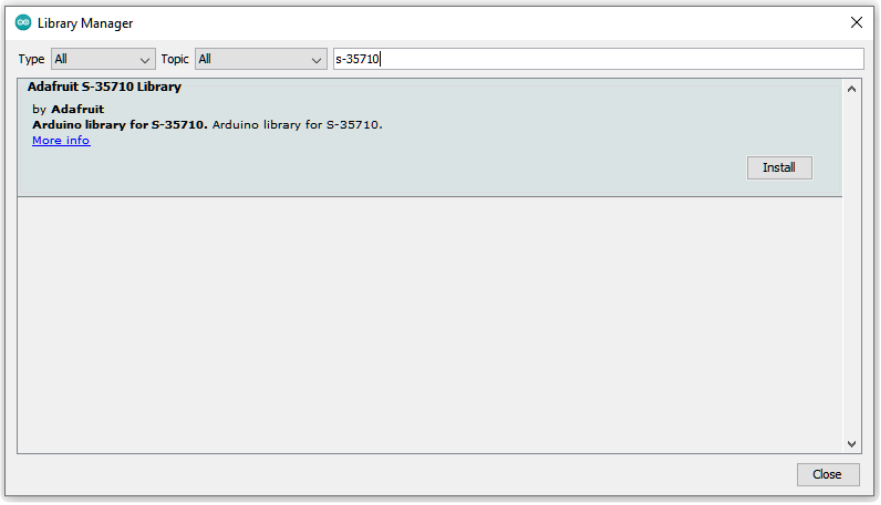

**Board 5V** to **breakout VIN (red wire)**
**Board GND** to **breakout GND (black wire)**
**Board SCL** to **breakout SCL (yellow wire)**
**Board SDA** to **breakout SDA (blue wire)**

## Library Installation

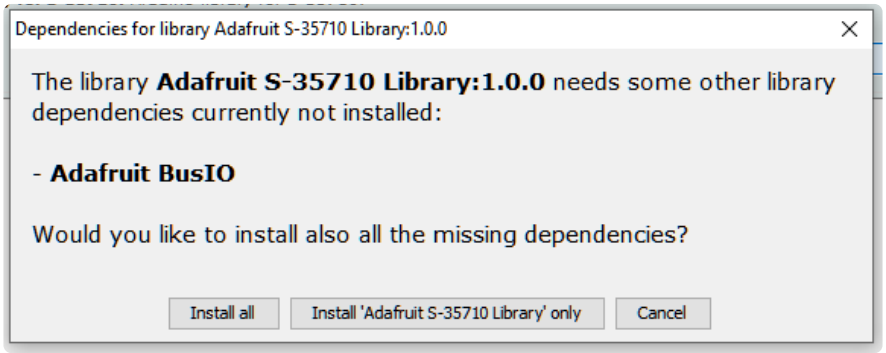
You can install the **Adafruit_S-35710** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit_S-35710**, and select the **Adafruit S-35710** library:




If asked about dependencies, click "Install all".




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

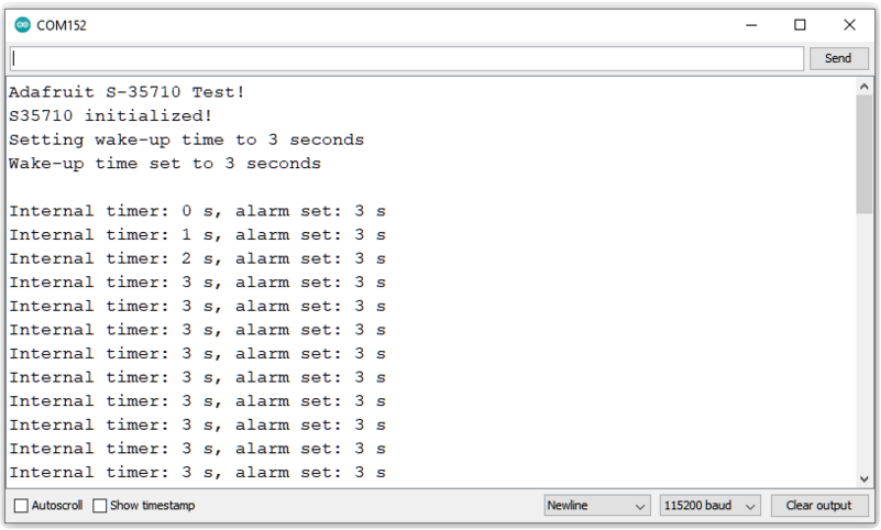
If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

## Example Code

```
#include "Adafruit_S35710.h"
#include <Wire.h>

// Optional reset pin connected to the S35710
const uint8_t RESET_PIN = -1;

// Create the S35710 with the reset pin
Adafruit_S35710 s35710 = Adafruit_S35710(RESET_PIN);
```

```
void setup() {
  // Start the serial communication
  Serial.begin(115200);
  while (!Serial)
    ; // Wait for serial port to connect

  Serial.println("Adafruit S-35710 Test!");

  if (!s35710.begin(&Wire, S35710_I2C_ADDRESS)) {
    Serial.println("Failed to initialize S-35710!");
    while (1)
      ;
  }
  Serial.println("S35710 initialized!");

  uint32_t wakeupTimeValue = 3; // 3 seconds

  Serial.print("Setting wake-up time to ");
  Serial.print(wakeupTimeValue);
  Serial.println(" seconds");
  if (!s35710.setWakeUpTimeRegister(wakeupTimeValue)) {
    Serial.println("Failed to set wake-up time register!");
    while (1)
      ;
  }

  // Test getting the wake-up time register
  int32_t readWakeupTimeValue = s35710.getWakeUpTimeRegister();
  if (readWakeupTimeValue == -1) {
    Serial.println("Failed to read wake-up time register!");
    while (1)
      ;
  }
  Serial.print("Wake-up time set to ");
  Serial.print(readWakeupTimeValue);
  Serial.println(" seconds");

  Serial.println();
}

void loop() {
  int32_t readTimeValue = s35710.getTimeRegister();
  int32_t readWakeupTimeValue = s35710.getWakeUpTimeRegister();
  if (readTimeValue == -1 || readWakeupTimeValue == -1) {
    // Hmm failed to read? wait & retry
    delay(1000);
    return;
  }

  Serial.print("Internal timer: ");
  Serial.print(readTimeValue);
  Serial.print(" s, alarm set: ");
  Serial.print(readWakeupTimeValue);
  Serial.println(" s");

  delay(1000);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the S-35710 recognized over I2C. Then, an alarm is set for 3 seconds. In the loop, the internal timer is printed out every second along with the alarm. Once the internal timer reaches 3 seconds, the alarm expires.




# Arduino Docs

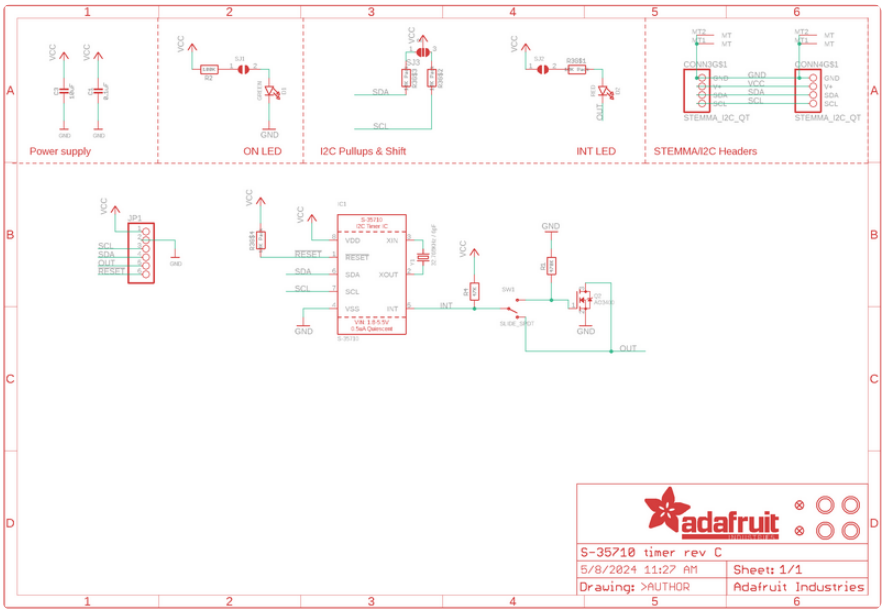
Arduino Docs (https://adafru.it/1a02)

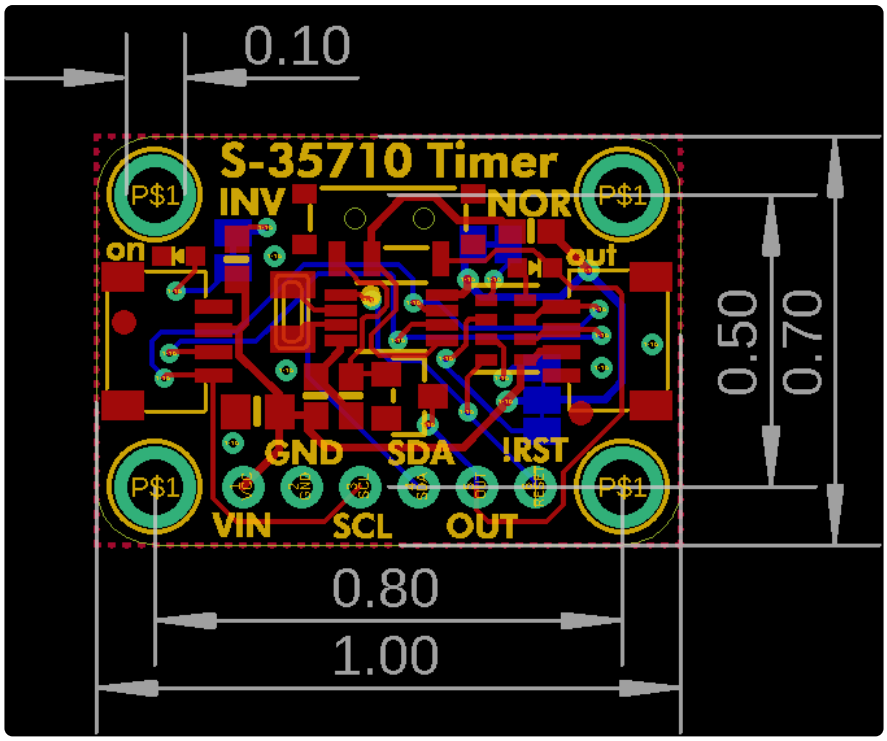
# Downloads

## Files

- S-35710 Datasheet (https://adafru.it/1a0f)
- EagleCAD PCB Files on GitHub (https://adafru.it/1a0g)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1a0h)

## Schematic and Fab Print




0.10
S-35710 Timer
P$1
INV
NOR
P$1
on
out
0.50
0.70
GND
SDA
!RST
P$1
P$1
VIN
SCL
OUT
0.80
1.00

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Adafruit:

5959