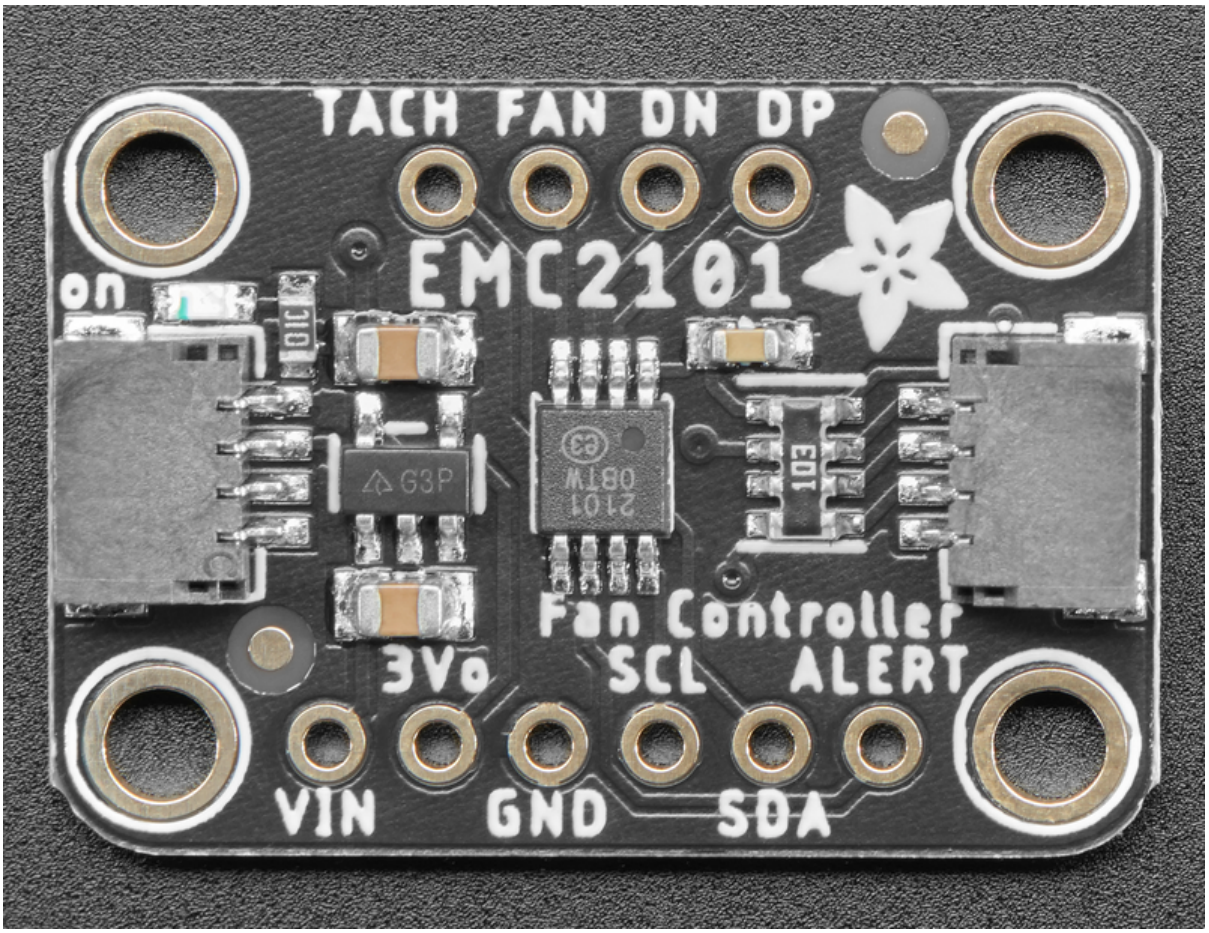
EMC2101 Fan Controller and Temperature sensor

Created by Bryan Siepert



https://learn.adafruit.com/emc2101-fan-controller-and-temperature-sensor

Last updated on 2024-03-08 03:51:53 PM EST

Table of Contents

Overview 3

Pinouts 6

- Power Pins
- I2C Logic Pins
- Other Pins

Arduino 7

- I2C Wiring
- Library Installation
- Load Example
- Example Code

Arduino Docs 11

Python & CircuitPython 11

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- CircuitPython Installation of EMC2101 Library
- Python Installation of EMC2101 Library
- CircuitPython & Python Usage
- Example Code

Python Docs 16

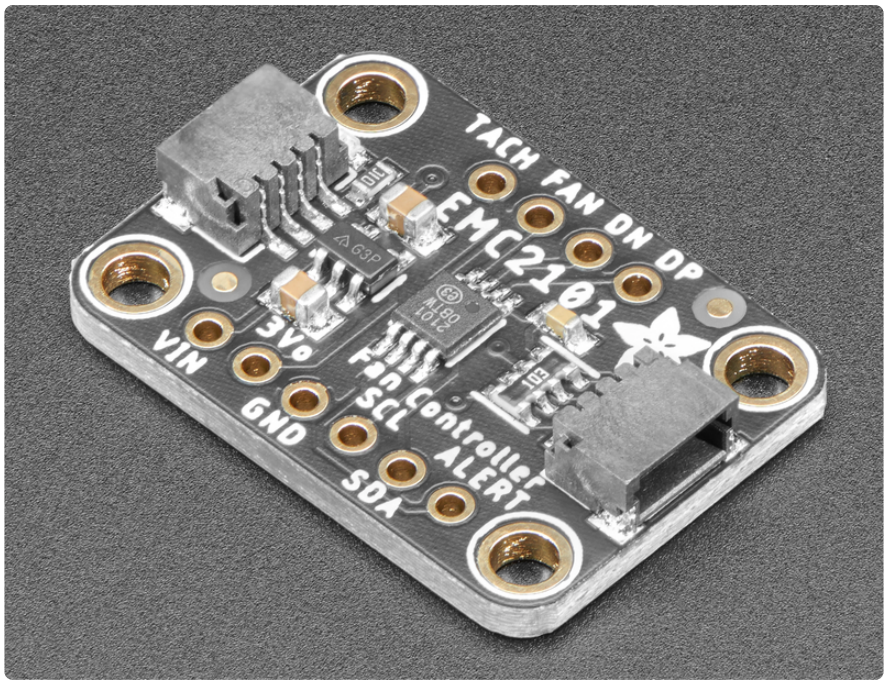
Additional Features 16

- Measurement Frequency
- Internal and External Temperature
- Look Up Table Support
- LUT Hysteresis
- Forcing a Manual Temperature
- PWM Tuning
- PWM vs DAC Output
- Fan Spinup Drive and Time
- Fan Minimum Speed

Downloads 21

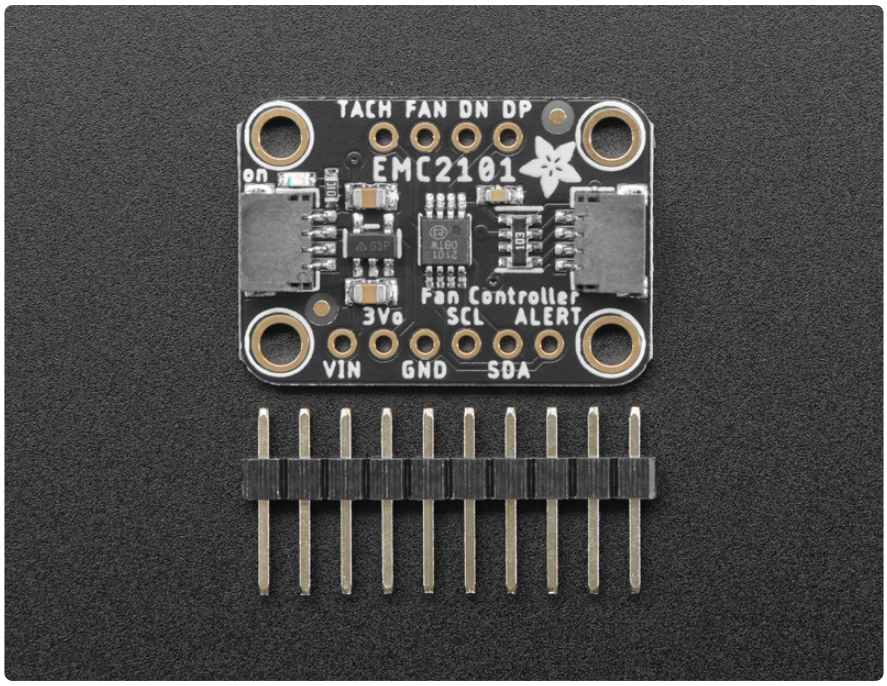
- Files
- Schematic
- Fab Print

Overview



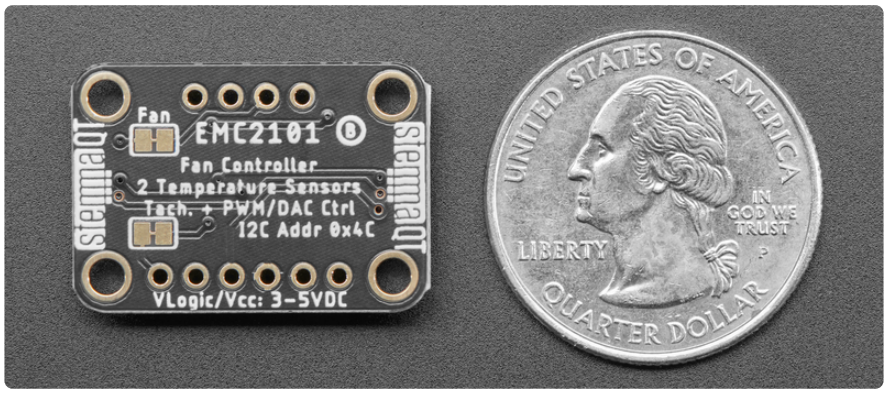
Cooling fans...They're everywhere, and they serve the important purpose of keeping things cool, generally electronics. One might rightfully think: "these fans are pretty good at moving air to keep things cool; maybe I can use one of these neat computer fans to keep my widget frosty" followed closely by throwing up one's hands in confusion at the sight of a 3 or even 4-pin connector. OK, power takes two pins but what are these other pins for and how do I use them to convince this fan to keep my things chilly?

The **EMC2101 from Microchip/SMSC** is **fan controller with temperature monitoring** and it will take care of all of that for you. With programmable PWM output and tachometer input, with both internal and external temperature sensing, with a 1°C accuracy, it's a perfect friend for any 3 or 4-pin PC fan you may find.



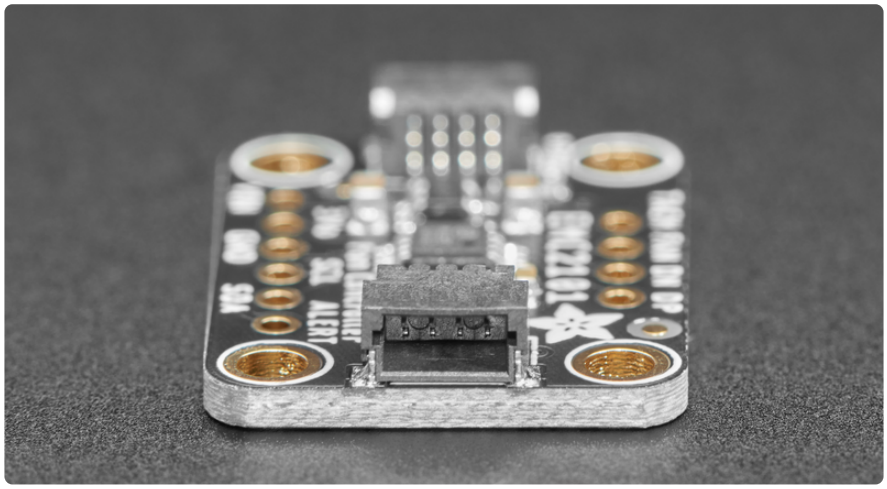
Four pin PC fans have a power and ground pin (those are often red and black) and then two more pins. One of those extra pins, the PWM pin, allows you to set a the speed of the fan. The last TACH pin is for a tachometer output that allows the EMC2101 to monitor the speed of the fan to make sure it's working as expected. Instead of using an PWM output and counter input on your microcontroller, this chip will take care of that all for you, all over every-day I2C.

In addition to allowing you to control a fan, the EMC2101 includes an internal temperature sensor, as well as connections for an **external temperature sensing diode**. If you use an external temperature sensor, you can even configure a **look up table (LUT)** that allows you to set different fan speeds depending on the temperature, and the EMC2101 will **automatically adjust the speed depending on the temperature**.



This neat little chip can easily be used to help provide cooling or ventilation to your next project. You might even be able to use it to help quiet down an existing project or hack a noisy piece of electronics that has a loud fan. By allowing you to run a fan at

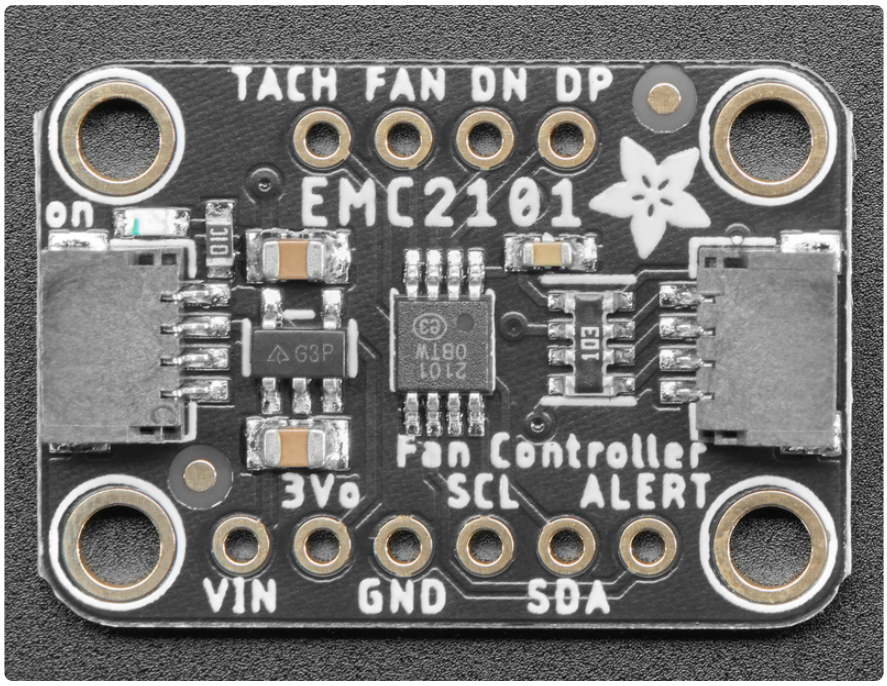
slower speeds when cooling needs are moderate, the EMC2101 can help ease the vibration noise caused by fans running at full speed.



Delivered to you on our custom Stemma QT form factor PCB, the Adafruit EMC2101 breakout is ready to help you integrate a fan into your project by making it easy to use with a range of devices. A voltage regulator and 5V tolerant pins allow you to use it with 3.3V or 5V microcontrollers or single board computers. The included included SparkFun qwiic (https://adafru.it/Fpw) compatible STEMMA QT (https://adafru.it/Ft4) connectors for the I2C bus allow you to make easy solderless connections to your controlling device, and the standard headers make breadboard prototyping easy.

Our libraries, wiring diagrams, and example code for Arduino, CircuitPython and Python complete the package. **How fan-tastic!**

Pinouts



Power Pins

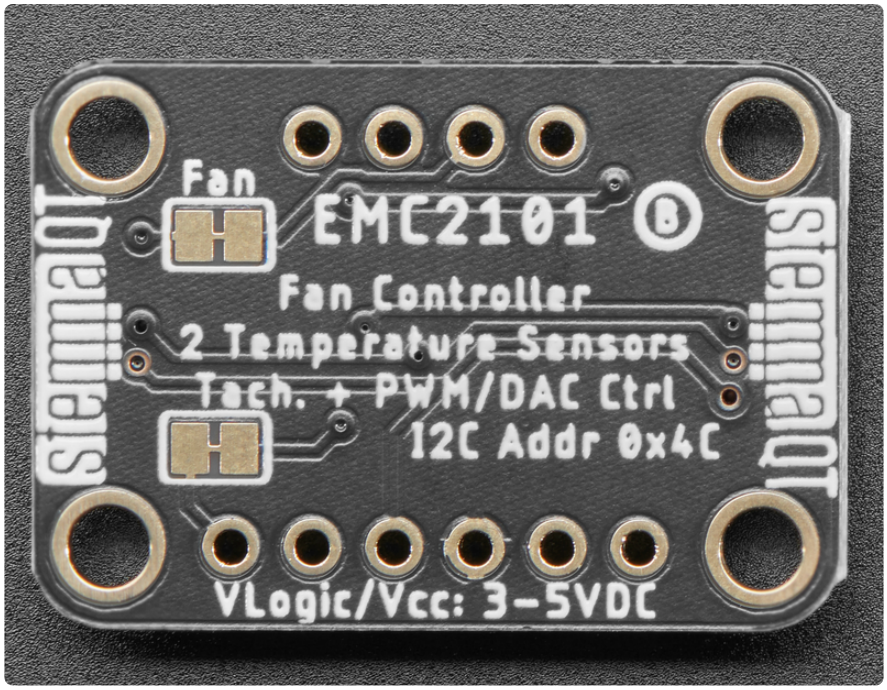
- **VIN** - this is the power pin. Since the sensor chip uses 3 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V
- **3Vo** - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like
- **GND** - common ground for power and logic

I2C Logic Pins

- **SCL** - I2C clock pin, connect to your microcontroller I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **SDA** - I2C data pin, connect to your microcontroller I2C data line. This pin is level shifted so you 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 dev boards with **STEMMA QT** or **Qwiic** connectors or to other things with various associated accessories (https://adafru.it/Ft6)

Other Pins

- **TACH** -Tachometer input for 4-pin fans or alert/interrupt pin
- **ALERT** - This is the same pin as TACH, it can also be used for interrupts
- **FAN** -The fan control output pin for 3 or 4 pin fans. **Defaults to PWM output** but can be configured to use a DAC to output a DC voltage
- **DN** - External temperature diode negative pin
- **DP** - External temperature diode positive pin



Jumpers

- **FAN** - Cut this jumper to disable the built-in pullup on the **Fan** pin when using the DAC/DC voltage output
- **TACH** - Cut this jumper to disable the pullup on the Tach pin when using the **Tach** pin for alerts

Either jumper can be bridged with solder later to re-enable the pullup after cutting the jumper

---

Arduino

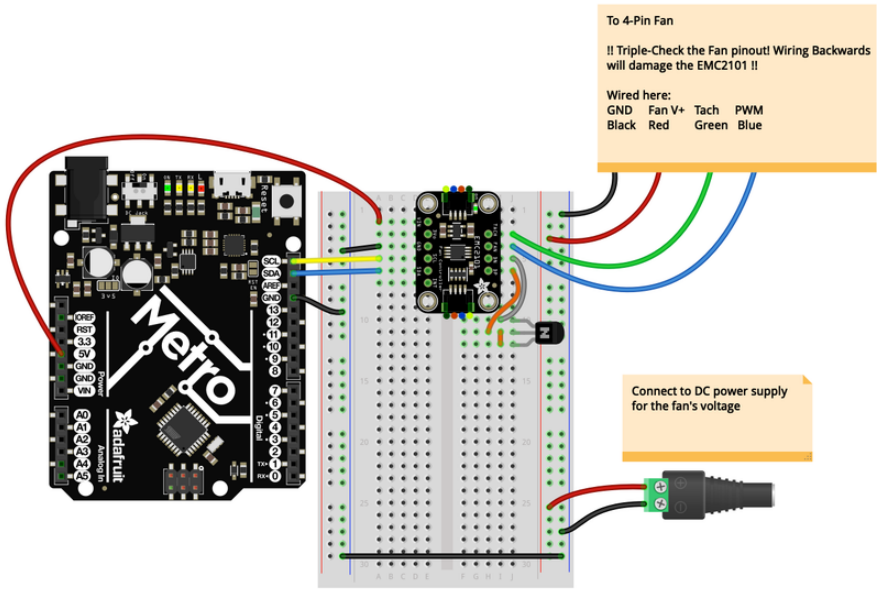
Using the EMC2101 with Arduino is a simple matter of wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit EMC2101 (https://adafru.it/PeJ) library we've written, and running the provided example code.

I2C Wiring

Use this wiring if you want to connect via I2C interface. The I2C address for the EMC2101 is **0x4C.**

Here is how to wire the sensor to a board using a breadboard, using a NPN transistor as a makeshift temperature diode. The examples show a Metro but wiring will work the same for an Arduino or other compatible board.

Carefully check the fan connections. Accidentally wiring the Fan V+ to the EMC2101 or Metro can permanently damage the boards

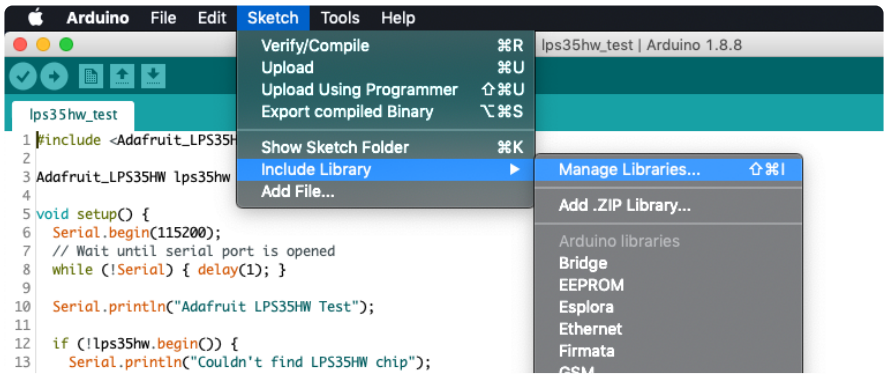


- Connect **board VIN (red wire)** to **Arduino 5V** if you are running a **5V** board Arduino (Uno, etc.). If your board is **3V,** connect to that instead.
- Connect **board GND (black wire)** to **Arduino GND**
- Connect **board SCL (yellow wire)** to **Arduino SCL**
- Connect **board SDA (blue wire)** to **Arduino SDA**
- Connect **board DP (orange wire)** to **Transistor Base**
- Connect **board DN (gray wire)** to **Transistor Emitter**
- Connect **transistor Collector (orange wire)** to **Transistor Base**
- Connect **board FAN** to fan PWM input
- Connect **board TACH** to fan Tach output
- Connect **DC jack positive** pin to **Fan V+** input
- Connect **DC jack GND** to **Fan V-/GND** input

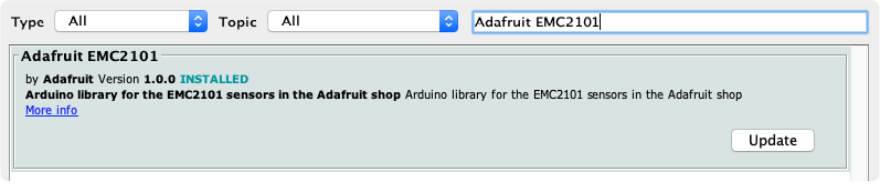
- Connect **DC jack GND** to **Arduino GND**

Library Installation

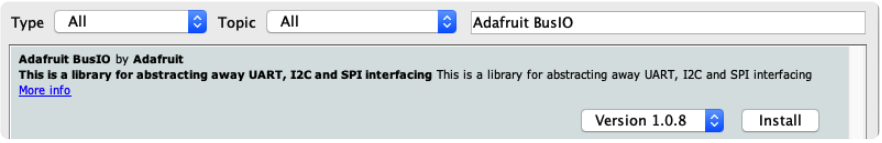
You can install the **Adafruit EMC2101** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit EMC2101** , and select the **Adafruit EMC2101** library:

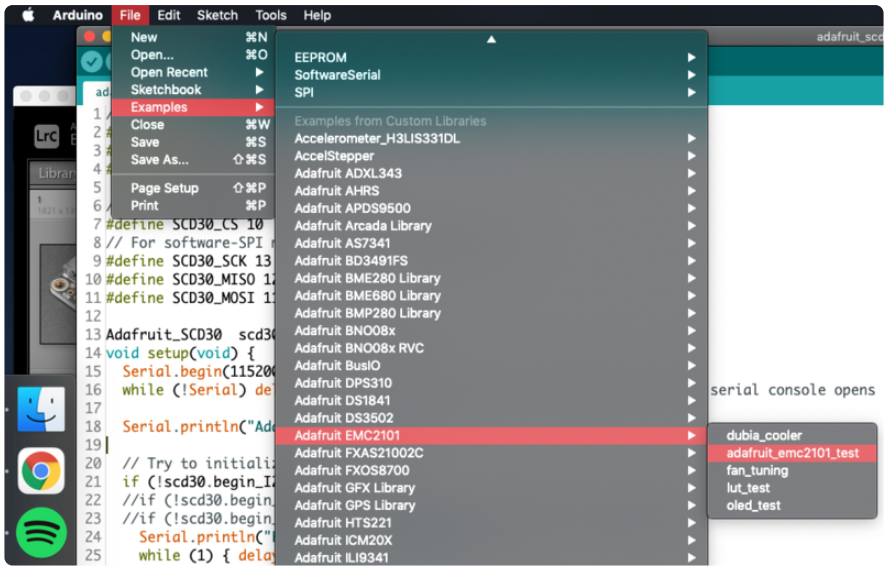


Follow the same process for the **Adafruit BusIO** library.



Load Example

Open up **File -> Examples -> Adafruit EMC2101 -> adafruit_emc2101_test**



After opening the demo file, upload to your Arduino wired up to the EMC2101. Once you upload the code, you will see the Internal and External **Temperature, Fan speed and PWM duty cycle** values being printed when you open the Serial Monitor (**Tools->Serial Monitor**) at **115200 baud**, similar to this:

```
Adafruit EMC2101 test!
EMC2101 Found!
Data rate set to: 32 HZ
External Temperature: 20.12 degrees C
Internal Temperature: 21 degrees C
Duty Cycle: 76% / Fan RPM: 1470 RPM

External Temperature: 20.25 degrees C
Internal Temperature: 20 degrees C
Duty Cycle: 76% / Fan RPM: 1470 RPM

External Temperature: 20.12 degrees C
Internal Temperature: 21 degrees C
Duty Cycle: 76% / Fan RPM: 1475 RPM
```

Example Code

```
// Basic demo for readings from Adafruit EMC2101
#include <Adafruit_EMC2101.h>

Adafruit_EMC2101  emc2101;

void setup(void) {
  Serial.begin(115200);
  while (!Serial) delay(10);     // will pause Zero, Leonardo, etc until serial
console opens

  Serial.println("Adafruit EMC2101 test!");

  // Try to initialize!
  if (!emc2101.begin()) {
    Serial.println("Failed to find EMC2101 chip");
    while (1) { delay(10); }
  }
  Serial.println("EMC2101 Found!");
```

```
  //  emc2101.setDataRate(EMC2101_RATE_1_16_HZ);
  Serial.print("Data rate set to: ");
  switch (emc2101.getDataRate()) {
    case EMC2101_RATE_1_16_HZ: Serial.println("1/16_HZ"); break;
    case EMC2101_RATE_1_8_HZ: Serial.println("1/8_HZ"); break;
    case EMC2101_RATE_1_4_HZ: Serial.println("1/4_HZ"); break;
    case EMC2101_RATE_1_2_HZ: Serial.println("1/2_HZ"); break;
    case EMC2101_RATE_1_HZ: Serial.println("1 HZ"); break;
    case EMC2101_RATE_2_HZ: Serial.println("2 HZ"); break;
    case EMC2101_RATE_4_HZ: Serial.println("4 HZ"); break;
    case EMC2101_RATE_8_HZ: Serial.println("8 HZ"); break;
    case EMC2101_RATE_16_HZ: Serial.println("16 HZ"); break;
    case EMC2101_RATE_32_HZ: Serial.println("32 HZ"); break;
  }

  emc2101.enableTachInput(true);
  emc2101.setPWMDivisor(0);
  emc2101.setDutyCycle(50);
}



void loop() {
  Serial.print("External Temperature: ");
  Serial.print(emc2101.getExternalTemperature());
  Serial.println(" degrees C");

  Serial.print("Internal Temperature: ");
  Serial.print(emc2101.getInternalTemperature());
  Serial.println(" degrees C");

  Serial.print("Duty Cycle: ");
  Serial.print(emc2101.getDutyCycle());
  Serial.print("% / Fan RPM: ");
  Serial.print(emc2101.getFanRPM());
  Serial.println(" RPM");
  Serial.println("");

  delay(100);
}
```

Arduino Docs

Arduino Docs (https://adafru.it/OeX)

Python & CircuitPython

It's easy to use the **EMC2101** with Python or CircuitPython, and the Adafruit CircuitPython EMC2101 (https://adafru.it/PeK) module. This module allows you to easily write Python code that reads temperature and fan speed from the **EMC2101** sensor, as well as setting the fan speed.

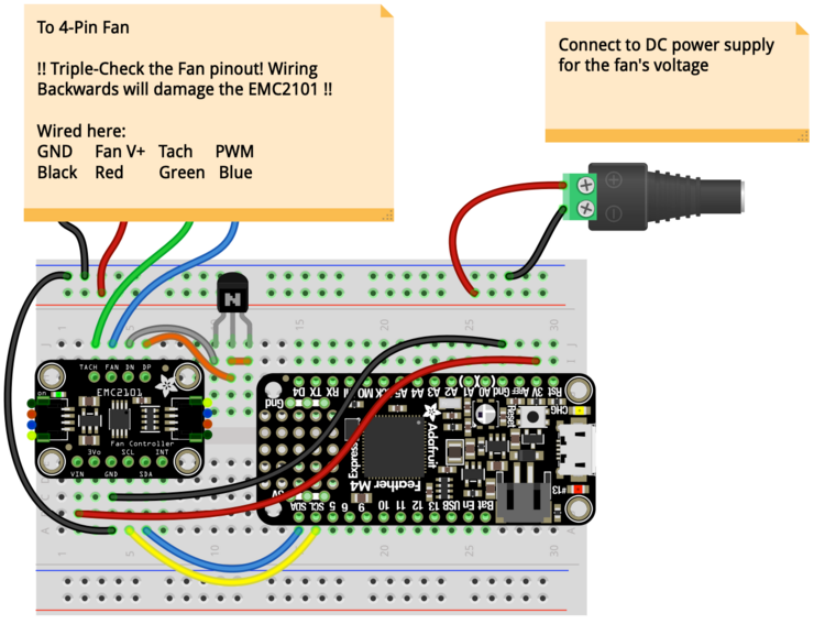
You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

We will show how to wire up the EMC2101 to a microcontroller and fan, along with a NPN Bipolar Junction Transistor (BJT) as a makeshift temperature diode.

Carefully check the fan connections! Accidentally connecting Fan V+ to the EMC or Feather may damage the boards

CircuitPython Microcontroller Wiring

First wire up a EMC2101 to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C using standard **0.100" pitch** headers to wire it up on a breadboard:

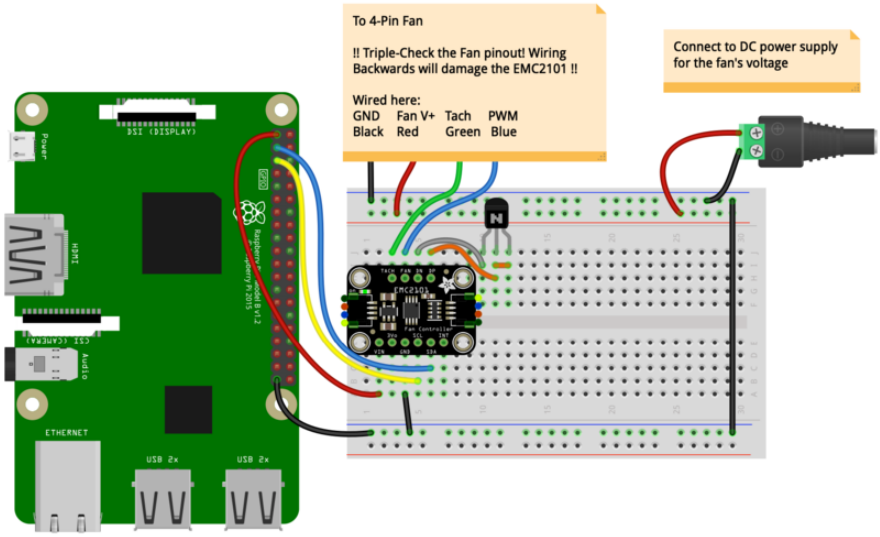


- Connect **board VIN (red wire)** to **Feather 3V**
- Connect **board GND (black wire)** to **Feather GND**
- Connect **board SCL (yellow wire)** to **Feather SCL**
- Connect **board SDA (blue wire)** to **Feather SDA**
- Connect **board DP (orange wire)** to **Transistor Base**
- Connect **board DN (gray wire)** to **Transistor Emitter**
- Connect **transistor Collector (orange wire)** to **Transistor Base**
- Connect **board FAN (blue wire)** to fan **PWM input**
- Connect **board TACH (green wire)** to fan **Tach output**
- Connect **DC jack positive** pin to **Fan V+** input
- Connect **DC jack GND** to **Fan V-/GND** input
- Connect **DC jack GND** to **Feather GND**

Python Computer Wiring

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

Here is an example of how to wire up a Raspberry Pi to the EMC2101 and fan, along with a NPN Bipolar Junction Transistor (BJT) as a makeshift temperature diode.



- Connect **board VIN (red wire)** to **RPi 3V**
- Connect **board GND (black wire)** to **RPi GND**
- Connect **board SCL (yellow wire)** to **RPi SCL**
- Connect **board SDA (blue wire)** to **RPi SDA**
- Connect **board DP (orange wire)** to **Transistor Base**
- Connect **board DN (gray wire)** to **Transistor Emitter**
- Connect **transistor Collector (orange wire)** to **Transistor Base**
- Connect **board FAN (blue wire)** to fan **PWM input**
- Connect **board TACH (green wire)** to fan **Tach output**
- Connect **DC jack positive** pin to **Fan V+** input
- Connect **DC jack GND** to **Fan V-/GND** input
- Connect **DC jack GND** to **RPi GND**

CircuitPython Installation of EMC2101 Library

You'll need to install the Adafruit CircuitPython EMC2101 (https://adafru.it/PeK) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/ENC). Our CircuitPython starter guide has a great page on how to install the library bundle (https://adafru.it/ABU).

Before continuing make sure your board's **lib** folder or root filesystem has the **adafruit_EMC2101.mpy** file as well as the **adafruit_register and adafruit_bus_device** folders.

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

Python Installation of EMC2101 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:

- `sudo pip3 install adafruit-circuitpython-emc2101`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython & Python Usage

> These examples assume the use of a standard 4-pin computer fan. 3-pin fans can be used as well, but won't be able to read the RPM of the fan due to the lack of a Tach output from the fan.

To demonstrate the usage of the sensor we'll initialize it to set and read the fan speed and read the external and internal temperatures from the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import board
from adafruit_emc2101 import EMC2101

i2c = board.I2C()

emc = EMC2101(i2c)
```

```
>>> import board
>>> from adafruit_emc2101 import EMC2101
>>> i2c = board.I2C()
>>> emc = EMC2101(i2c)
```

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

- **internal_temperature** - The temperature of the EMC2101 itself, a value in degrees Celsius.
- **external_temperature** - The temperature measured by the external sensor, a value in degrees Celsius.

```
print("External temperature:", emc.external_temperature, "C")
print("Internal temperature:", emc.internal_temperature, "C")
```

```
>>> print("External temperature:", emc.external_temperature, "C")
External temperature: 21.25 C
>>> print("Internal temperature:", emc.internal_temperature, "C")
Internal temperature: 20 C
```

And of course you can also use the corresponding properties to read and set the fan speed:

- **manual_fan_speed** - The speed of the fan when not controlled by the LUT
- **fan_speed** - The speed of the fan in **R**evolutions **P**er **M**inute (RPM)

```
emc.manual_fan_speed = 25
print("Fan speed:", emc.fan_speed, "RPM")
```

```
>>> emc.manual_fan_speed = 25
>>> print("Fan speed:", emc.fan_speed, "RPM")
Fan speed: 718.946 RPM
```

Example Code

```
# SPDX-FileCopyrightText: 2020 Bryan Siepert, written for Adafruit Industries
#
# SPDX-License-Identifier: MIT
import time
```

```
import board
from adafruit_emc2101 import EMC2101

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
emc = EMC2101(i2c)
while True:
    print("Setting fan speed to 25%")
    emc.manual_fan_speed = 25
    time.sleep(2)  # longer sleep to let it spin down from 100%
    print("Fan speed", emc.fan_speed)
    time.sleep(1)

    print("Setting fan speed to 50%")
    emc.manual_fan_speed = 50
    time.sleep(1.5)
    print("Fan speed", emc.fan_speed)
    time.sleep(1)

    print("Setting fan speed to 75%")
    emc.manual_fan_speed = 75
    time.sleep(1.5)
    print("Fan speed", emc.fan_speed)
    time.sleep(1)

    print("Setting fan speed to 100%")
    emc.manual_fan_speed = 100
    time.sleep(1.5)
    print("Fan speed", emc.fan_speed)
    time.sleep(1)

    print("External temperature:", emc.external_temperature, "C")
    print("Internal temperature:", emc.internal_temperature, "C")

    print("")
    time.sleep(0.5)
```

Python Docs

Python Docs (https://adafru.it/Of4)

Additional Features

Measurement Frequency

Depending on your application, you may not need to measure the temperature continuously if for example it doesn't change quickly. You can specify the EMC2101's temperature conversion/measurement rate which will affect the power usage of the EMC2101; more frequent measurements will use more power.

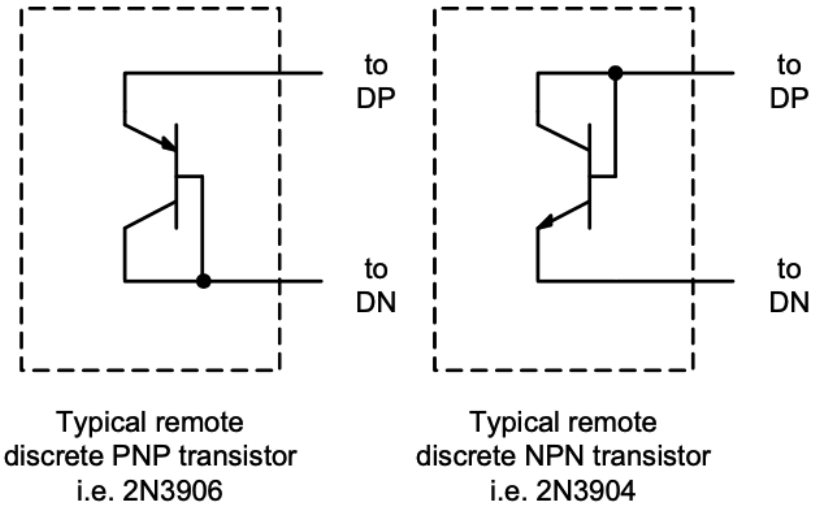
Python - Documentation (https://adafru.it/PfM)

Arduino - Documentation (https://adafru.it/PfN)

Arduino - Example (https://adafru.it/PfO)

Internal and External Temperature

The EMC2101 supports measuring both the temperature of the EMC2101 itself as the internal temperature, as well as the temperature of an externally connected "diode".



I've put diode in quotes above because we've only been able to successfully measure external temperature using a NPN BJT transistor with the collector and base tied together which acts like a diode. PNP BJT's should also work with the base and emitter connected. The diagram from the datasheet above shows how connecting the pins of a BJT lets it act like a diode.

Python - Documentation (https://adafru.it/PfM)

Python - Example Code (https://adafru.it/PfP)

Arduino - Documentation (https://adafru.it/PfN)

Arduino - Example (https://adafru.it/PfO)

Note: The DAC output is not meant to directly drive a Fan! The output current is limited to 1mA, well below what any fan will need. Instead, the DAC output should be used as an input signal to other control circuitry.

Python - Documentation (https://adafru.it/PfM)

Arduino - Documentation (https://adafru.it/PfN)

Look Up Table Support

As an alternative to adjusting the speed directly with your code, the EMC2101 allows you to set up a Look Up Table (LUT) that specifies how fast the fan should spin for a given temperature. One important note is that the LUT only works automatically based on the external temperature.

Python - LUT Documentation (https://adafru.it/PfM)

Python - LUT Example (https://adafru.it/PfP)

Arduino - Documentation (https://adafru.it/PfN)

Arduino - LUT Example (https://adafru.it/PfQ)

LUT Hysteresis

If the current temperature is very close to a LUT temperature, the fan may "thrash" by changing the fan speed excessively as the temperature fluctuates between above and below the LUT temperature.

To prevent thrashing, the EMC2101 applies a hysteresis value to the measured temperature which is an additional number of degrees below the LUT temperature that the measurement must be before switching to the lower LUT entry.

Python - Documentation (https://adafru.it/PfM)

Arduino - Documentation (https://adafru.it/PfN)

Arduino - LUT Example (https://adafru.it/PfQ)

Forcing a Manual Temperature

For testing the LUT, the EMC2101 allows you to set a temperature that the LUT will use instead of the external temperature. Each time you set a forced/manual temperature, the EMC2101 will consult the LUT to find the corresponding fan speed. The keen eyed may notice that you can use this to hack LUT support for other values than the

external temperature sensor by regularly updating the forced temperature based on another measurement.

Python - Documentation (https://adafru.it/PfM)

Python - LUT Example (https://adafru.it/PfP)

Arduino - Documentation (https://adafru.it/PfN)

Arduino - LUT Example (https://adafru.it/PfQ)

PWM Tuning

It can sometimes be necessary to adjust the PWM frequency to suit your application or prevent a motor from making audible noise. These features allow you to specify the base PWM clock, divisor and frequency.

Python - Documentation (https://adafru.it/PfM)

Python - PWM Example

Arduino - Documentation (https://adafru.it/PfN)

Arduino - Example (https://adafru.it/PfO)

PWM vs DAC Output

By default the EMC2101 sets the speed of the fan by outputting a PWM signal compatible with 4 and 3-pin fans. A transistor in the fan uses this signal to change the amount of its operating voltage.

The EMC2101 can alternatively be configured to output a steady DC voltage instead of a pulsed signal, should your application require it. The voltage range is from **0** to approximately **3.24V** with the same 6-bit resolution as the PWM output.

Fan Spinup Drive and Time

The EMC2101 allows you to specify how the fan should be started from a stop by specifying a spinup speed and time. Sometimes fans have to build up momentum before slowing down to a lower speed, and these settings help with that. Fortunately the default settings are good enough for most cases.

Python - Documentation (https://adafru.it/PfM)

Arduino - Documentation (https://adafru.it/PfN)

Fan Minimum Speed

This setting allows you to specify the minimum speed of your fan. 4-pin fans will often still report a small speed on their tach output when stopped, and this setting allows you to say "anything below this speed is off" so you don't think the fan is still on when it is not.

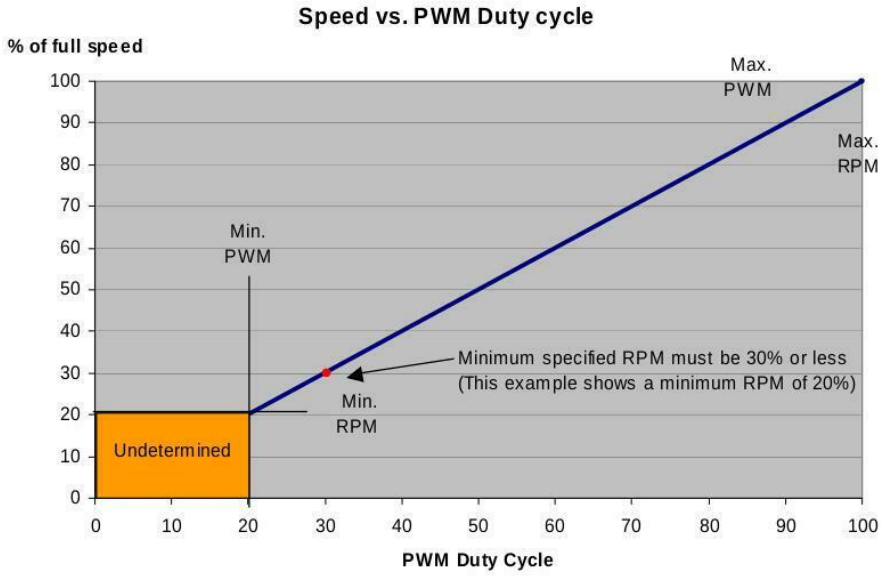
Python - Documentation (https://adafru.it/PfM)

Arduino - Documentation (https://adafru.it/PfN)

Arduino - Fan Tuning Example (https://adafru.it/PfR)

NOTE: PC fan behavior below 20% duty cycle is undetermined. Some fans may not fully stop.

Per the Intel specification "4-Wire Pulse Width Modulation (PWM) Controlled Fans", fan response to a PWM signal below 20% duty cycle is undetermined. As a result, **some PC fans may not fully stop**, even with 0% duty cycle.



This can vary from fan to fan. Ideally, this information will be given in a fan's datasheet. For example, this if from the datasheet for a Delta Electronics AUB0912VH fan:

13. SPEED VS PWM CONTROL SIGNAL: (AT RATED VOLTAGE & PWM FREQUENCY=25KHZ)

DUTY CYCLE (%)	SPEED R.P.M.	CURRENT (A) TYP.
100	3800±10%	0.38
50	1950±10%	0.10
0~20	500~1000	0.02

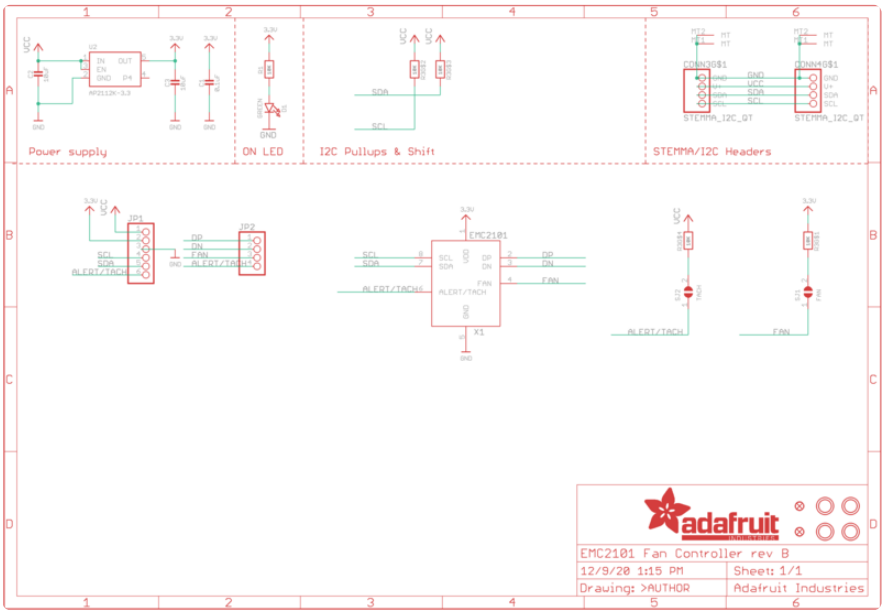
As can be seen, the RPM never goes to 0, even with 0% duty cycle.

Downloads

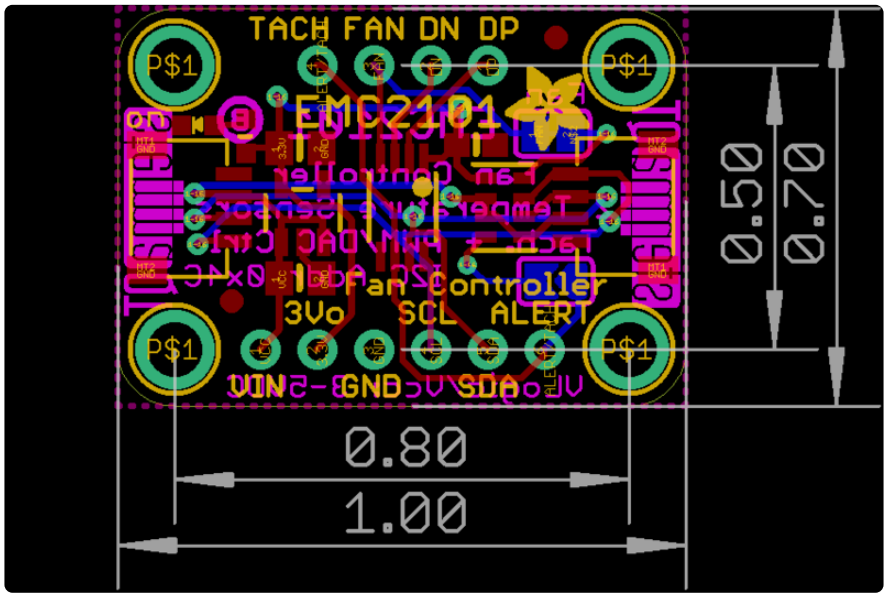
Files

- EMC2101 Datasheet (https://adafru.it/PeL)
- EagleCAD files on GitHub (https://adafru.it/PeM)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/PeN)

Schematic



Fab Print


TACH FAN DN DP
P$1
P$1
EMC2101
Fan Controller
3Vo SCL ALERT
P$1
P$1
VIN GND SDA
0.50
0.70
0.80
1.00