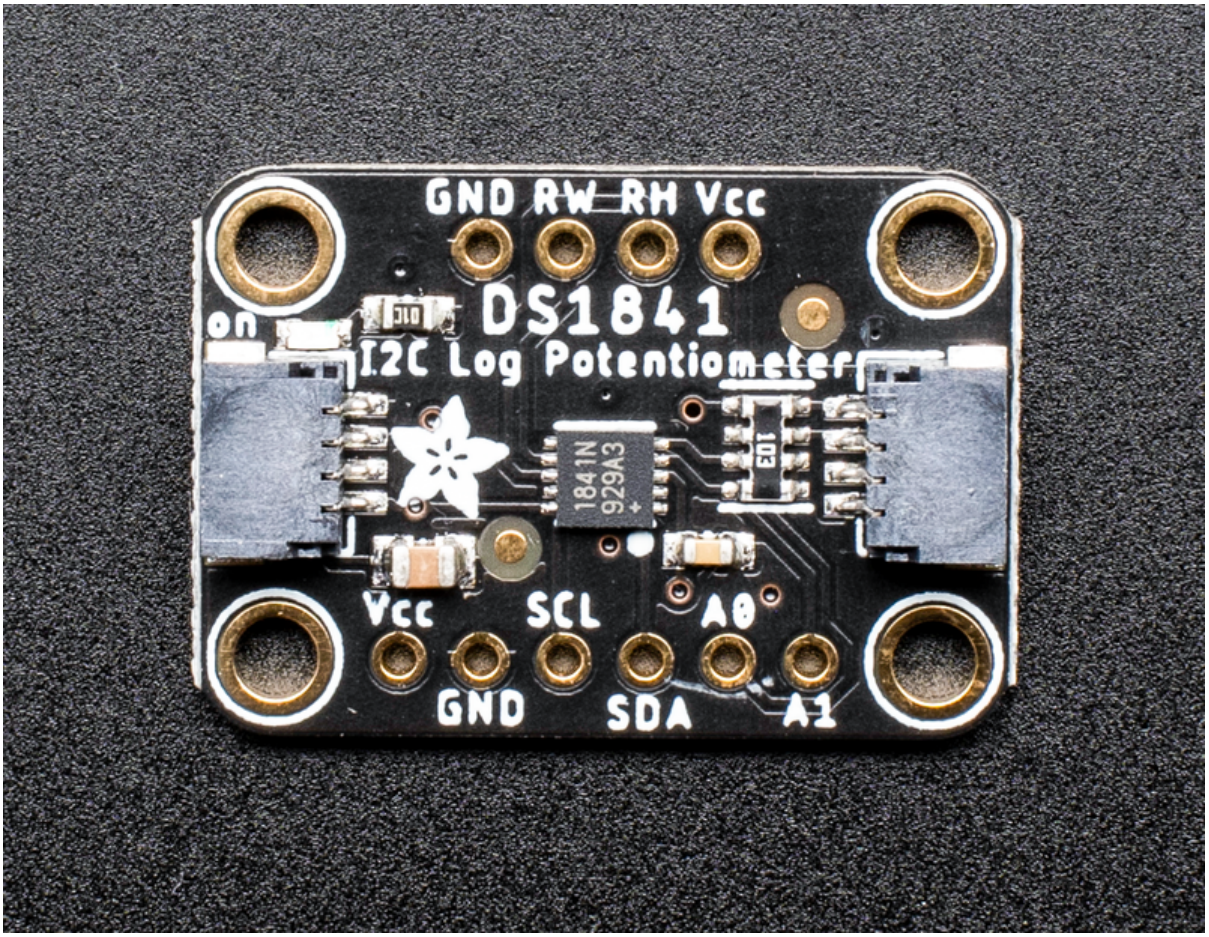
Adafruit DS1841 I2C Logarithmic Resistor

Created by Bryan Siepert



https://learn.adafruit.com/adafruit-ds1841-i2c-logarithmic-resistor

Last updated on 2022-12-01 03:52:39 PM EST

Table of Contents

Overview 3

Pinouts 5

- Power Pins
- I2C Logic Pins
- Resistor Pins
- Extra Pins
- Adjusting the I2C Address

Arduino 7

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

Arduino Docs 10

Python & CircuitPython 10

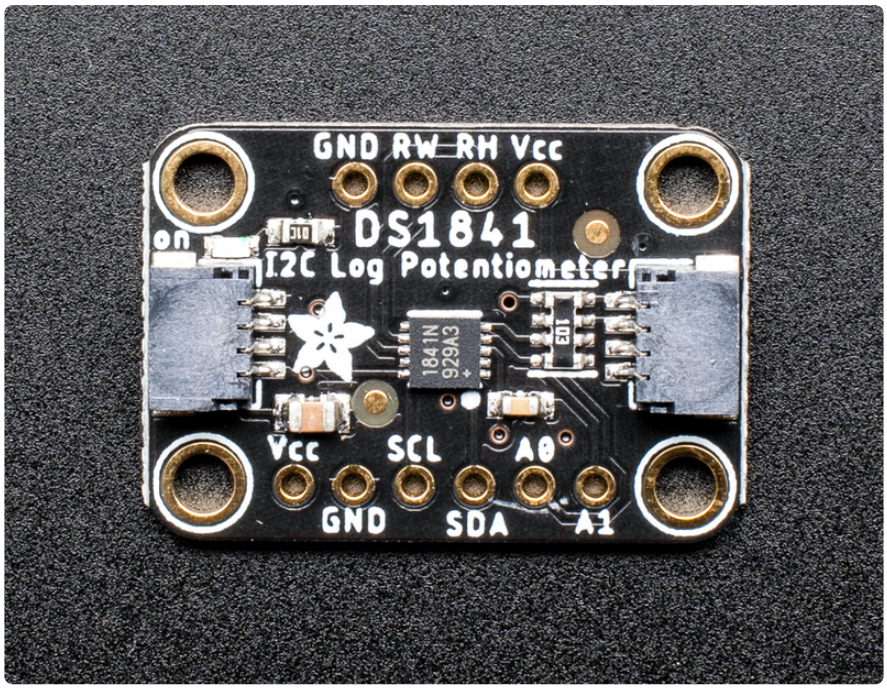
- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- CircuitPython Installation of DS1841 Library
- Python Installation of DS1841 Library
- CircuitPython Usage
- Full CircuitPython Example Code
- Python Usage
- Full Python Example Code

Python Docs 17

Downloads 17

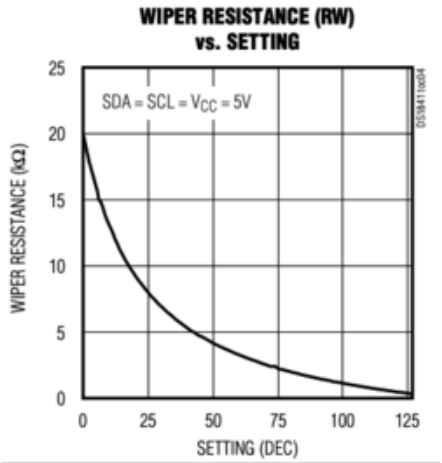
- Files
- Schematic
- Fab Print

Overview



Potentiometers are the perfect tool when you want to change your circuit by turning a knob. Turns out, there are times when you want to adjust your circuit without manually turning a knob, and the DS1841 I2C Logarithmic Resistor from Maxim can do just that. It's a programmable resistor, similar to an I2C potentiometer like the the DS3502 I2C Potentiometer (), so why another?

The big difference between the two is how the resistance changes in relation to changes made to the wiper. The DS3502's resistance has a linear relationship to the wiper's setting. Each time you change the wiper by a given amount, the resistance will change by the same amount. With the DS1841, the relationship between resistance and the wiper setting is logarithmic. This means that as the wiper setting changes, the amount of resistance will depend on where in the wiper's range the current setting is.

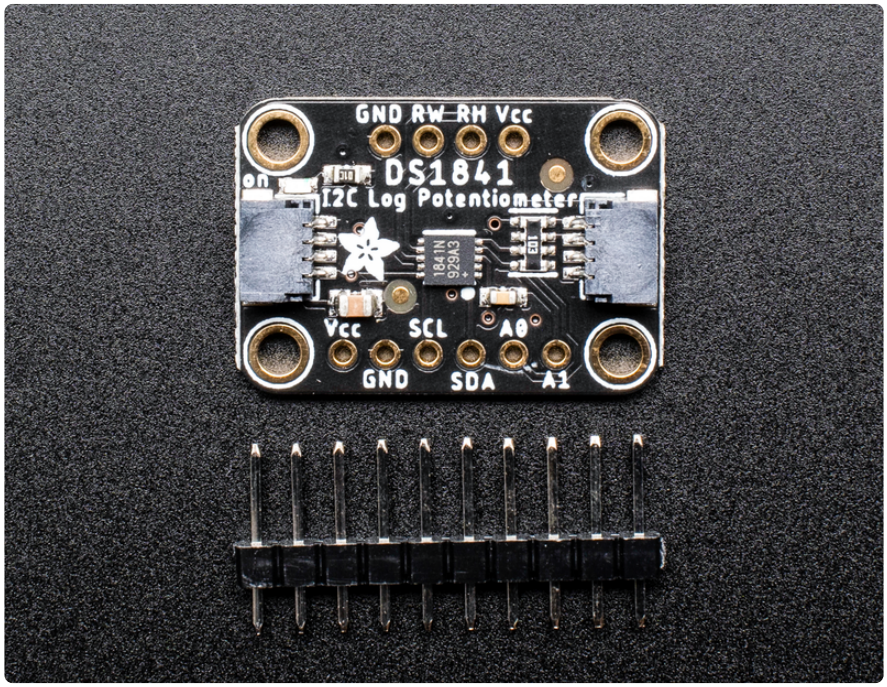


This graph shows the relationship between the wiper setting and the resulting resistance. At the start, the changes are relatively large, but as the values increase, the resulting change in resistance gets smaller and smaller.

Log potentiometers are used in audio applications for things like volume control because they better match the response of human ear to sound.

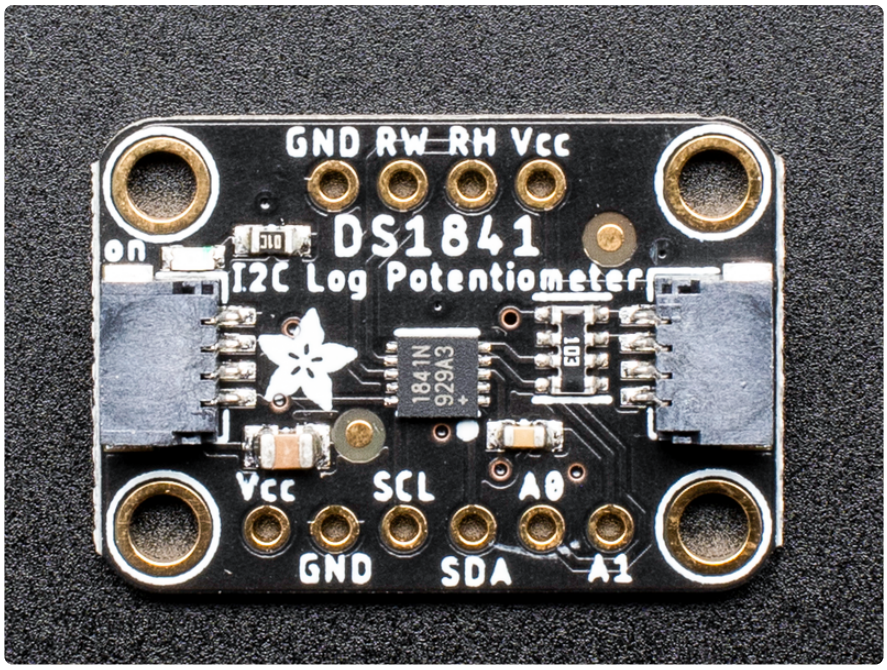
The DS1841's resistance ranges from 22kOhms to 3.7 kOhms and has 128 tap points. Even more interestingly, the DS1841 can be configured to adjust its resistance based on temperature with hysteresis to keep things from jumping around.

Additionally the temperature compensation can be adjusted by using the LUT (Look Up Table) built into the DS1841. This table allows you to specify the wiper setting for each of 70 temperature increments between -39 and 100 degrees C, plus one each for above or below that range. You can even manually set the wiper to one of the entries in the LUT.



Working with the DS1841 is easy. We've put it on a breakout PCB with the required support circuitry and SparkFun qwiic () compatible STEMMA QT () connectors to allow you to use it with other similarly equipped boards without needing to solder. This handy little helper can work with 3.3V or 5V micros, so it's ready to get to work with a range of development boards. To make things even easier we've gone and written Arduino and CircuitPython/Python 3 drivers to simplify interfacing with your new knob-replacing friend.

Pinouts



Power Pins

The sensor on the breakout requires between a 2.7V and 5.5V, and can be easily used with most microcontrollers from an Arduino to a Feather or something else.

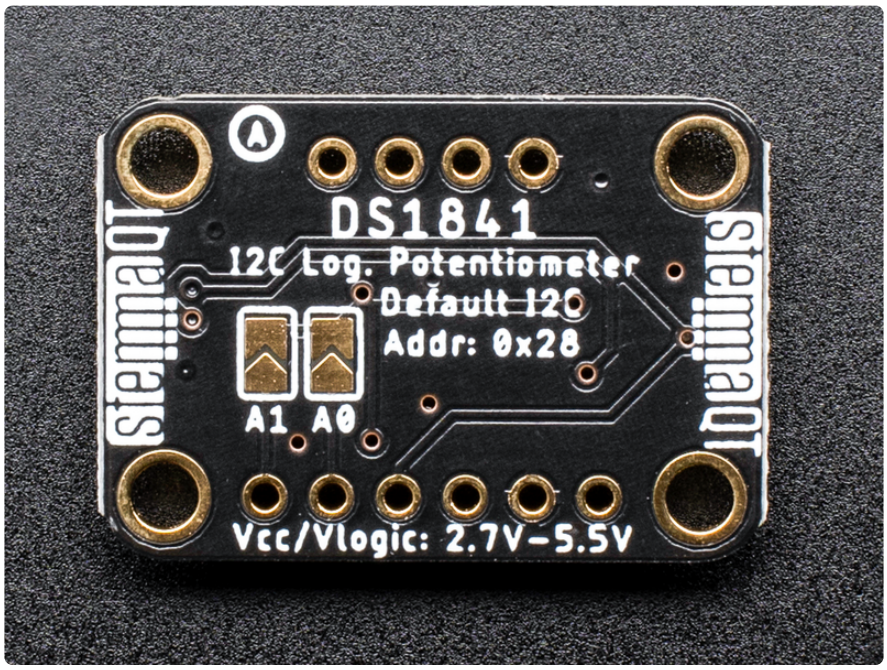
- Vcc - this is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
- GND - common ground for power and logic

I2C Logic Pins

- SCL - I2C clock pin, connect to your microcontrollers I2C clock line. The logic level is the same as Vcc and it has a 10K pullup already on it.
- SDA - I2C data pin, connect to your microcontrollers I2C data line. The logic level is the same as Vcc. and it has a 10K pullup already on it.
- STEMMA QT () - These connectors allow you to connectors to dev boards with S TEMMA QT connectors or to other things with various associated accessories ()

Resistor Pins

- RW is the wiper of the potentiometer. As the wiper value is adjusted via I2C, the resistance between RW and GND changes
- RH is the High Terminal of the potentiometer, often connected to your high voltage source. The DS1841 has a fixed resistor between RH and RW which forms a voltage divider when RH is connected to VCC, with the voltage between RW and GND varying with the wiper setting



Extra Pins

- A0 and A1 - These are the address select pins which set the two least significant bits of the I2C address. The are pulled to GND with two resistors which sets the address to 0x28 by default.

Adjusting the I2C Address

The default address is 0x28 and the address can be calculated by 'adding' the A0/A1 t o the base of 0x28. You can change the values of one or both A0 and A1 by bridging the solder jumpers on the back of the board with solder. You can also set A0 or A1 by using a jumper wire to connect them to Vcc.

A0 sets the lowest bit with a value of 1, and A1 sets the next highest bit that has a value of 2. The final address is 0x28 + A1 + A0.

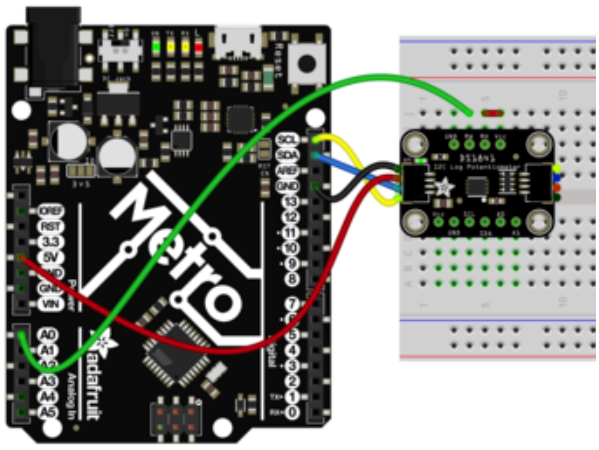
- For example if only A0 is tied to Vcc and A1 is left connected to GND, the address is 0x28 + 0 + 1 = 0x29
- If only A1 is tied to Vcc, the address is 0x28 + 2 + 0 = 0x2A (A is 10 in hexadecimal)
- If both A0 and A1 are connected to Vcc, the address is 0x28 + 2 + 1 = 0x2B (B is 11 in hexadecimal)

Arduino

I2C Wiring

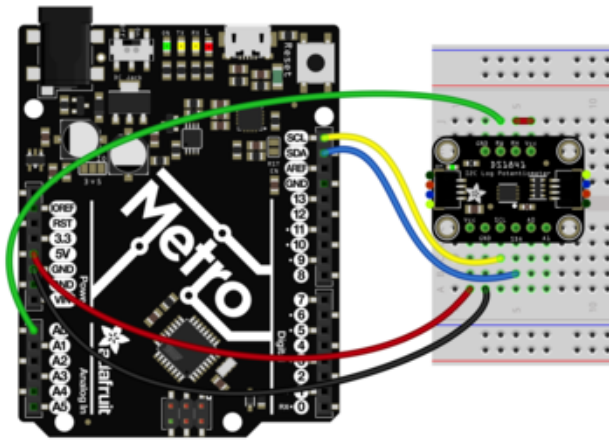
Wiring the DS1841 to communicate with your microcontroller is straight forward forward thanks to the I2C interface. For these examples we can use the Metro or Arduino to measure the voltage changes as the DS1841 adjusts its resistance. The instructions below reference a Metro (), but the same applies to an Arduino

STEMMA QT Wiring



Connect board VIN (red wire) to Metro 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 Metro GND
Connect board SCL (yellow wire) to Metro SCL
Connect board SDA (blue wire) to Metro SDA
Connect RH to Metro 5V
Connect RW to the A0 pin on the Metro, to allow us to measure the voltage

Breadboard Wiring

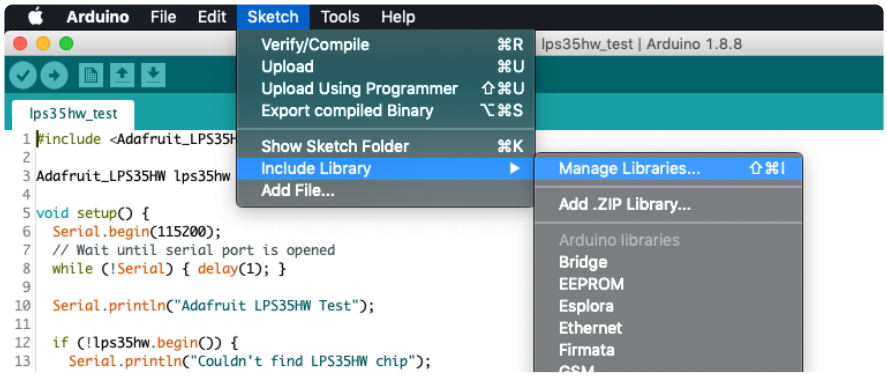


Connect board VIN (red wire) to Metro 5V if you are running a 5V board Arduino (Uno, etc.). If your board is 3V, connect to that instead.
Connect board GND (black wire) to Metro GND
Connect board SCL (yellow wire) to Metro SCL
Connect board SDA (blue wire) to Metro SDA
Connect RH to Metro 5V
Connect RW to the A0 pin on the Metro, to allow us to measure the voltage

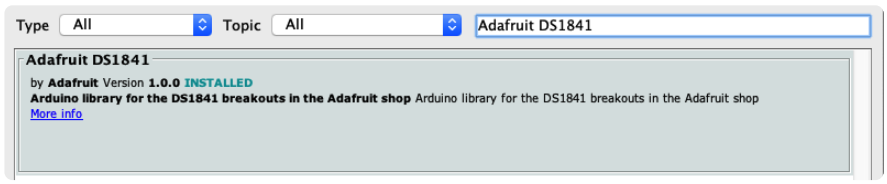
Library Installation

Once wired up, to start using the DS1841, you'll need to install the Adafruit_DS1841 library (). The library is available through the Arduino library manager so we recommend taking that approach.

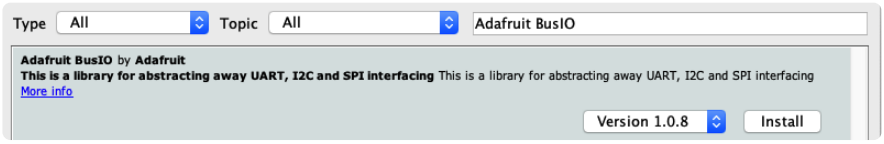
From the Arduino IDE, open up the Library Manager:



Click the Manage Libraries ... menu item, search for Adafruit DS1841, and select the A dafruit DS1841 library and click Install:



Follow the same process to install the Adafruit BusIO library.



Load Example

Open up File -> Examples -> Adafruit DS1841 -> ds1841_test and upload to your Metro wired up to the DS1841 breakout as shown above. Once you upload the code, you will see the wiper settings and measured voltage being printed when you open the Serial Monitor (Tools->Serial Monitor) at 15200 baud, similar to this:

```
Adafruit DS1841 test!
DS1841 Found!
Current VCC Voltage:5094.40mV
Temperature: 25 degrees C
Voltage:3.95
Temperature: 25 degrees C
Wiper: 10 LSB

Voltage:0.65
Temperature: 25 degrees C
Wiper: 120 LSB
```

Example Code

The following code comes with the library and illustrates the basic function of using the variable resistance of the DS1841 to change the voltage measured at pin A0:

```
// Basic demo for readings from Adafruit DS1841
#include <Wire.h>
#include <Adafruit_DS1841.h>
#define VOLTAGE_DIV_PIN A0
float wiperVoltage(void);
float wiper_voltage;

Adafruit_DS1841  ds;
void setup(void) {
  Serial.begin(115200);
  while (!Serial) delay(10);     // will pause Zero, Leonardo, etc until serial
console opens

  Serial.println("Adafruit DS1841 test!");

  // Try to initialize!
  if (!ds.begin()) {
    Serial.println("Failed to find DS1841 chip");
    while (1) { delay(10); }
  }
  Serial.println("DS1841 Found!");
  Serial.print("Current VCC Voltage:"); Serial.print(ds.getVoltage());
Serial.println("mV");
  Serial.print("Temperature: ");Serial.print(ds.getTemperature());Serial.println("
```

```
degrees C");

  pinMode(VOLTAGE_DIV_PIN, INPUT);

}

void loop() {
  ds.setWiper(10);
  delay(1000);
  wiper_voltage = wiperVoltage();
  Serial.print("Voltage:");Serial.println(wiper_voltage);
  Serial.print("Temperature: ");Serial.print(ds.getTemperature());Serial.println("
degrees C");
  Serial.print("Wiper: ");Serial.print(ds.getWiper());Serial.println(" LSB");
  Serial.println("");

  ds.setWiper(120);
  delay(1000);
  wiper_voltage = wiperVoltage();
  Serial.print("Voltage:");Serial.println(wiper_voltage);
  Serial.print("Temperature: ");Serial.print(ds.getTemperature());Serial.println("
degrees C");
  Serial.print("Wiper: ");Serial.print(ds.getWiper());Serial.println(" LSB");
  Serial.println("");
}

float wiperVoltage(void) {
  float wiper_value = analogRead(VOLTAGE_DIV_PIN);
  wiper_value *= 5.0;
  wiper_value /= 1024;
  return wiper_value;
}
```

Arduino Docs

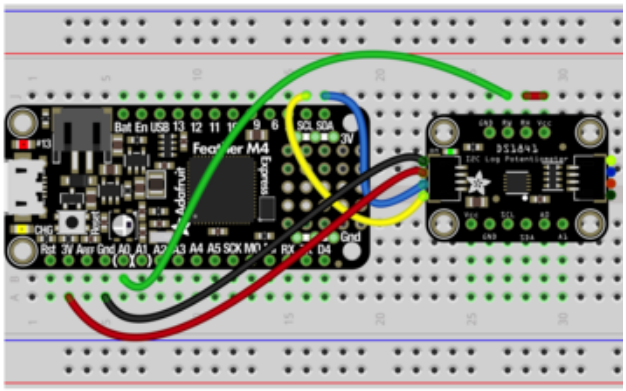
Arduino Docs ()

Python & CircuitPython

CircuitPython Microcontroller Wiring

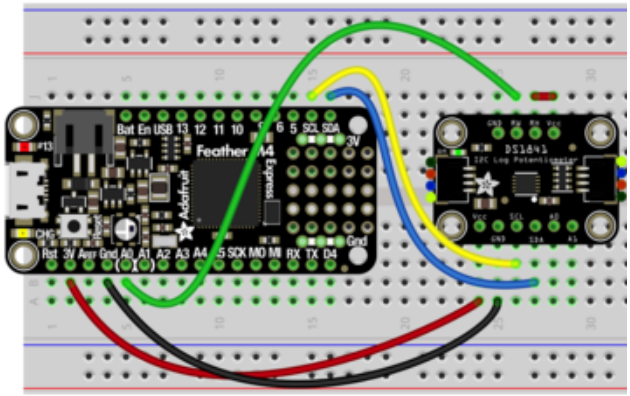
Wiring the DS1841 to communicate with your microcontroller is straightforward thanks to the I2C interface. For these examples, we can use a Metro or a Feather to measure the voltage changes as the DS1841 adjusts its resistance. The instructions below reference a Feather (), but the same applies to a Metro.

STEMMA QT Wiring



Connect board Vcc (long red wire) to Feather 3.3V
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 RH to Vcc (short red wire)
Connect RW to the A0 pin on the Feather, to allow us to measure the voltage

Breadboard Wiring



Connect board Vcc (long red wire) to Feather 3.3V
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 RH to Vcc (short red wire)
Connect RW to the A0 pin on the Feather, to allow us to measure the voltage

Python Computer Wiring

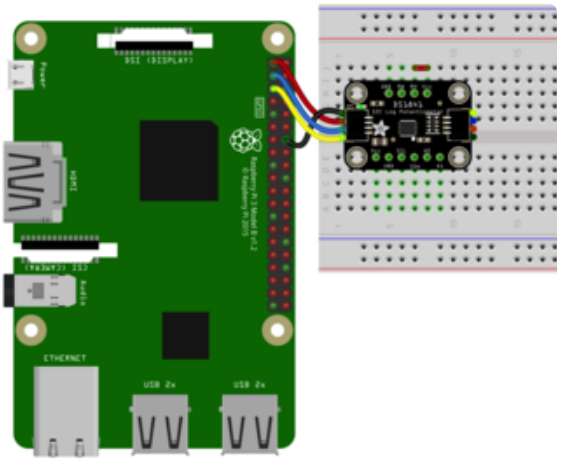
Since there's dozens of Linux computers/boards you can use we will show wiring for R aspberry Pi (). For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported ().

Here's the Raspberry Pi wired with I2C:

Note that because the Raspberry Pi does not include any pins with analog to digital converters (ADCs) to read the voltage that will change on the DS1841's RW pin, you will need to use a multimeter to measure the voltage between the RW pin and GND.

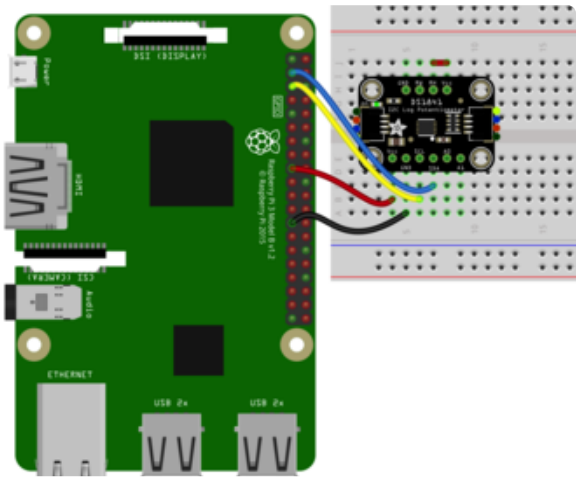
If you are new to using a multimeter to measure voltage, check out ladyada's guide to multimeters () which has a section on using your multimeter to measure voltage ()

STEMMA QT Wiring



Connect board Vcc (long red wire) to Pi 3.3V
Connect board GND (black wire) to Pi GND
Connect board SCL (yellow wire) to Pi SCL
Connect board SDA (blue wire) to Pi SDA
board Vcc to board RH (short red wire)
Multimeter Positive Lead to sensor RW
Multimeter Negative Lead to sensor GND

Breadboard Wiring



Connect board Vcc (long red wire) to Pi 3.3V
Connect board GND (black wire) to Pi GND
Connect board SCL (yellow wire) to Pi SCL
Connect board SDA (blue wire) to Pi SDA
board Vcc to board RH (short red wire)
Multimeter Positive Lead to sensor RW
Multimeter Negative Lead to sensor GND

CircuitPython Installation of DS1841 Library

You'll need to install the Adafruit CircuitPython DS1841 () library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython () 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

(). Our CircuitPython starter guide has a great page on how to install the library bundle ().

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- adafruit_ds1841.mpy
- adafruit_bus_device
- adafruit_register

Before continuing make sure your board's lib folder or root filesystem has the adafruit_ds1841.mpy, adafruit_bus_device, and adafruit_register files and folders copied over.

Next connect to the board's serial REPL ()so you are at the CircuitPython >>> prompt.

Python Installation of DS1841 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:

- `sudo pip3 install adafruit-circuitpython-ds1841`

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

Because of hardware differences, Python Computer users should follow the "Python Usage" examples in the next section

To demonstrate the usage of the potentiometer we'll initialize it and set the wiper value to change the voltage on the WH pin. We will then use the A0 pin to take an analog reading of the voltage on the WH pin.

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

```
import board
import busio
from analogio import AnalogIn
import adafruit_ds1841
i2c = busio.I2C(board.SCL, board.SDA)
ds1841 = adafruit_ds1841.DS1841(i2c)
wiper_output = AnalogIn(board.A0)
```

```
>>> import board
>>> import busio
>>> from analogio import AnalogIn
>>> import adafruit_ds1841
>>> i2c = busio.I2C(board.SCL, board.SDA)
>>> ds1841 = adafruit_ds1841.DS1841(i2c)
>>> wiper_output = AnalogIn(board.A0)
```

With the driver initialized, we can the set the wiper value using the wiper property. Then we can take an ADC reading to measure the voltage, and convert the raw ADC value to a human-readable value

```
ds1841.wiper = 127
print("Wiper set to %d" % ds1841.wiper)
voltage = wiper_output.value
voltage *= 3.3
voltage /= 65535
print("Wiper voltage: %.2f V" % voltage)
```

```
>>> ds1841.wiper = 127
>>> print("Wiper set to %d" % ds1841.wiper)
Wiper set to 127
>>> voltage = wiper_output.value
>>> voltage *= 3.3
>>> voltage /= 65535
>>> print("Wiper voltage: %.2f V" % voltage)
Wiper voltage: 0.32 V
```

We can then set the wiper to a different value and see how it changes the measured wiper voltage

```
ds1841.wiper = 0
print("Wiper set to %d" % ds1841.wiper)
voltage = wiper_output.value
voltage *= 3.3
voltage /= 65535
print("Wiper voltage: %.2f V" % voltage)
```

```
>>> ds1841.wiper = 0
>>> print("Wiper set to %d" % ds1841.wiper)
Wiper set to 0
>>> voltage = wiper_output.value
>>> voltage *= 3.3
>>> voltage /= 65535
>>> print("Wiper voltage: %.2f V" % voltage)
Wiper voltage: 2.82 V
```

Full CircuitPython Example Code

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

from time import sleep
import board
import busio
from analogio import AnalogIn
import adafruit_ds1841

####### NOTE ################
# this example will not work with Blinka/rasberry Pi due to the lack of analog pins.
# Blinka and Raspberry Pi users should run the "ds1841_blinka_simpletest.py" example

# WIRING:
# 1 Wire connecting  VCC to RH to make a voltage divider using the
#   internal resistor between RH and RW
# 2 Wire connecting RW to A0

# setup of the i2c bus giving the SCL (clock) and SDA (data) pins from the board
i2c = busio.I2C(board.SCL, board.SDA)
# create the ds1841 instance giving the I2C bus we just set up
ds1841 = adafruit_ds1841.DS1841(i2c)

# set up an analog input, selecting the A0 pin
wiper_output = AnalogIn(board.A0)

while True:

    # set th
    ds1841.wiper = 127
    print("Wiper set to %d" % ds1841.wiper)
    voltage = wiper_output.value
    voltage *= 3.3
    voltage /= 65535
    print("Wiper voltage: %.2f V" % voltage)
    print("")
    sleep(1.0)

    ds1841.wiper = 0
    print("Wiper set to %d" % ds1841.wiper)
    voltage = wiper_output.value
    voltage *= 3.3
    voltage /= 65535
    print("Wiper voltage: %.2f V" % voltage)
    print("")
    sleep(1.0)

    ds1841.wiper = 63
    print("Wiper set to %d" % ds1841.wiper)
    voltage = wiper_output.value
    voltage *= 3.3
    voltage /= 65535
    print("Wiper voltage: %.2f V" % voltage)
    print("")
    sleep(1.0)
```

Python Usage

Because the Raspberry Pi and many other similar devices do not include the hardware for measuring analog voltages, you will have to use a multimeter to measure the voltage on the RW pin.

To start, similar to above we will import the needed modules and initialize the driver

```
import board
import busio
import adafruit_ds1841

i2c = busio.I2C(board.SCL, board.SDA)
ds1841 = adafruit_ds1841.DS1841(i2c)
```

```
Python 3.7.5 (default, Jan 15 2020, 14:43:37)
[Clang 10.0.1 (clang-1001.0.46.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import board
>>> import busio
>>> import adafruit_ds1841
>>>
>>> i2c = busio.I2C(board.SCL, board.SDA)
>>> ds1841 = adafruit_ds1841.DS1841(i2c)
```

Once the driver has been initialized, we can use it to set the value of the wiper

```
ds1841.wiper = 127
```

```
>>> ds1841.wiper = 127
```

You can then use your multimeter or other voltage measuring device to check the voltage across GND and RW. It should be the same as shown in the CircuitPython example above, approximately 0.3V

Next, change the wiper value again and measure the voltage on the RW pin

```
ds1841.wiper = 0
```

```
>>> ds1841.wiper = 0
```

The voltage on the RW pin should be the same as shown in the CircuitPython example above, this time approximately 2.8V

Full Python Example Code

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

####### NOTE ################
# This example is meant for use with Blinka/rasberry Pi due to the lack of analog
pins.
# CircuitPython board users should run the "ds1841_simpletest.py" example

# WIRING:
# 1 Wire connecting  VCC to RH to make a voltage divider using the
#   internal resistor between RH and RW

# As this code runs, measure the voltage between ground and the RW (wiper) pin
# with a multimeter. You should see the voltage change with each print statement.
from time import sleep
import board
import busio
import adafruit_ds1841

i2c = busio.I2C(board.SCL, board.SDA)
ds1841 = adafruit_ds1841.DS1841(i2c)

while True:
    ds1841.wiper = 127
    print("Wiper value set to 127")
    sleep(5.0)

    ds1841.wiper = 0
    print("Wiper value set to 0")
    sleep(5.0)

    ds1841.wiper = 63
    print("Wiper value set to 63")
    sleep(5.0)
```

Python Docs

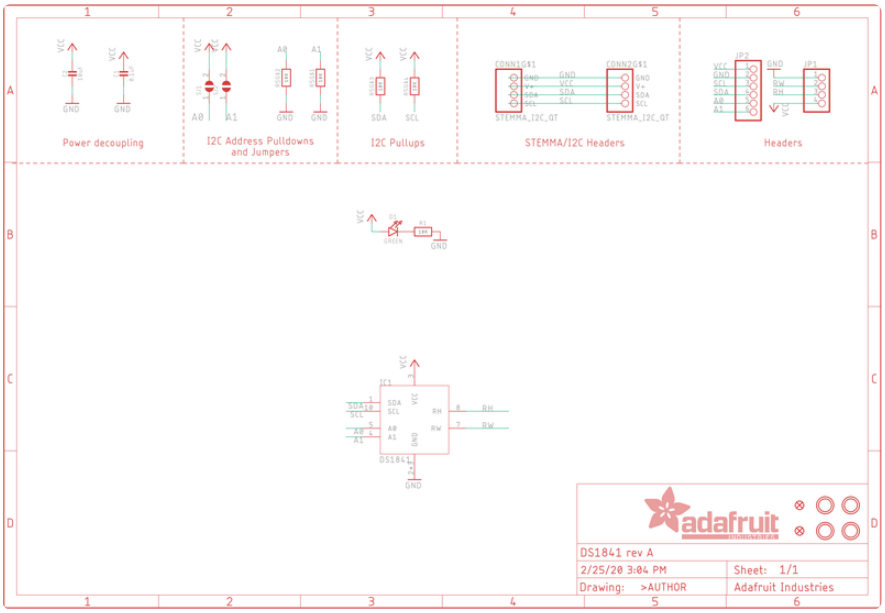
Python Docs ()

Downloads

Files

- DS1841 Datasheet ()
- EagleCAD files on GitHub ()
- Fritzing object in Adafruit Fritzing Library ()

Schematic



Fab Print