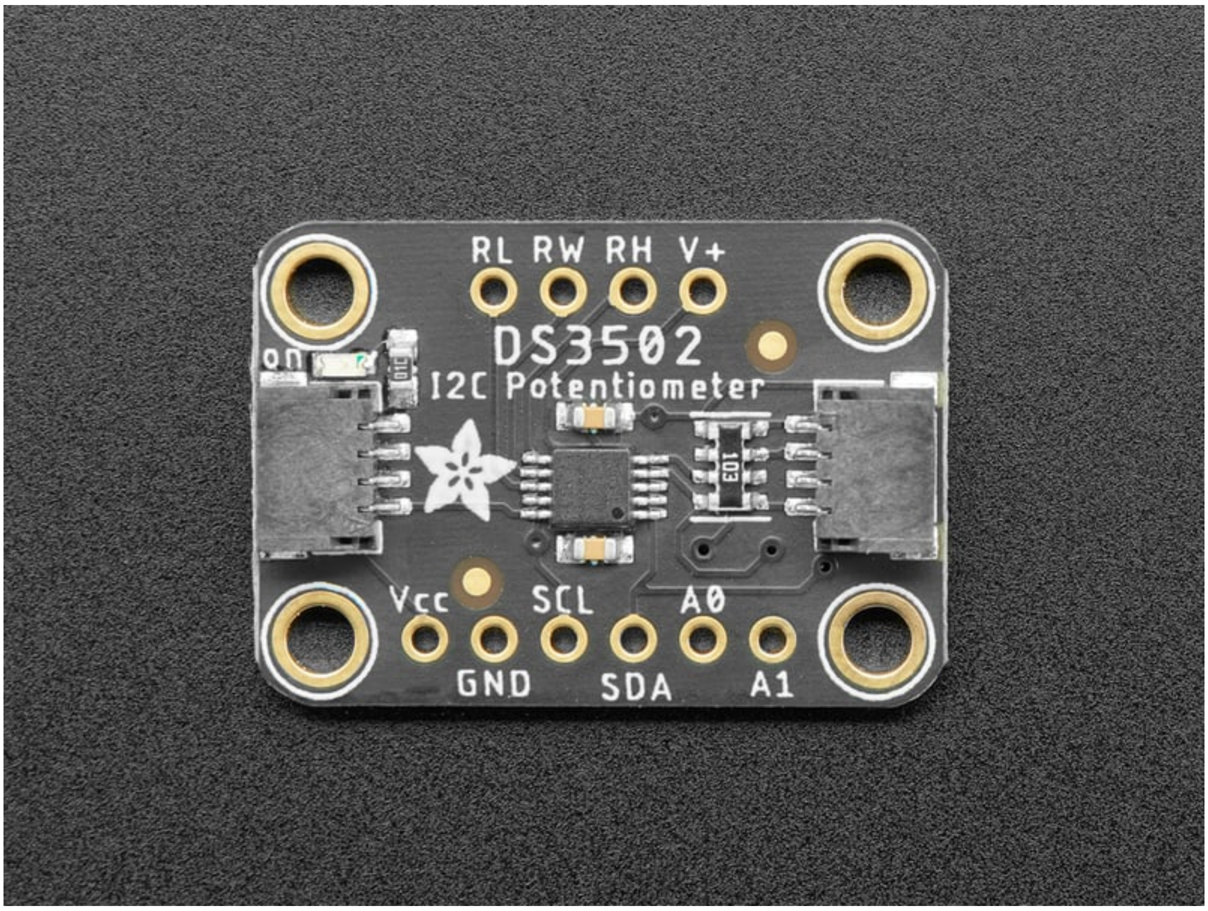
Adafruit DS3502 I2C Digital Potentiometer

Created by Bryan Siepert



https://learn.adafruit.com/ds3502-i2c-potentiometer

Last updated on 2024-03-25 12:12:50 PM EDT

Table of Contents

Overview 3

Pinouts 5

- Power Pins
- I2C Logic Pins
- Resistor Pins
- Extra Pins

Assembly 8

- Prepare the header strip:
- Add the breakout board:
- And Solder!

Arduino 13

- Wiring
- Library Installation
- Load Example
- Example Code

Arduino Docs 16

Python & CircuitPython 16

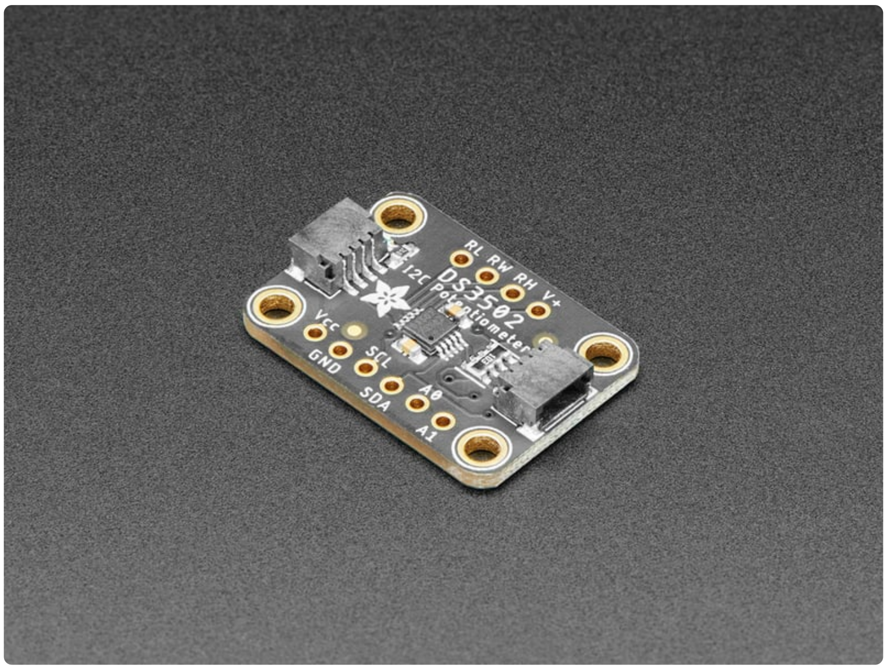
- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- CircuitPython Installation of DS3502 Library
- Python Installation of DS3502 Library
- CircuitPython Usage
- Full CircuitPython Example Code
- Python Usage
- Full Python Example Code

Python Docs 22

Downloads 23

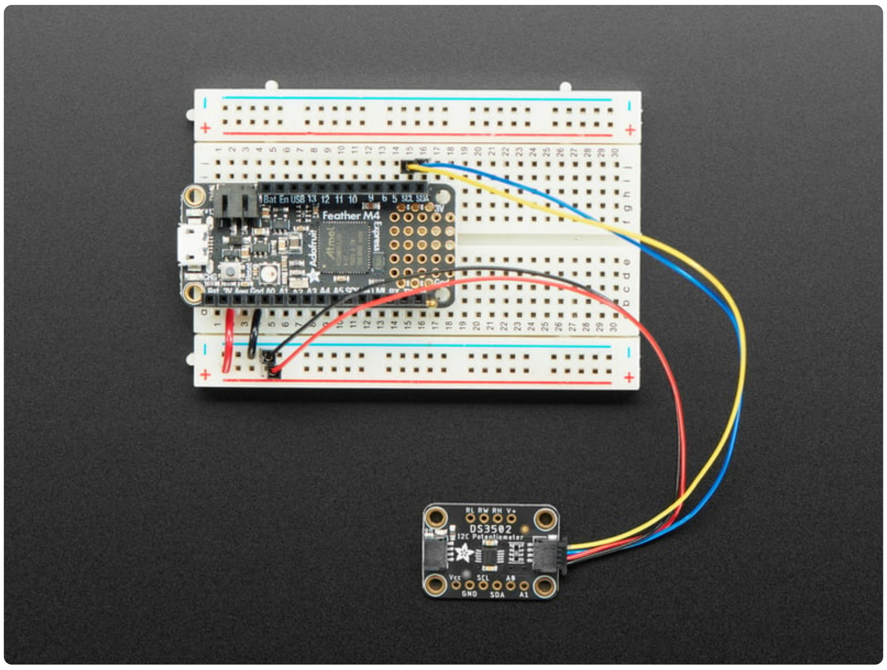
- Files
- Schematic
- Fab Print

Overview



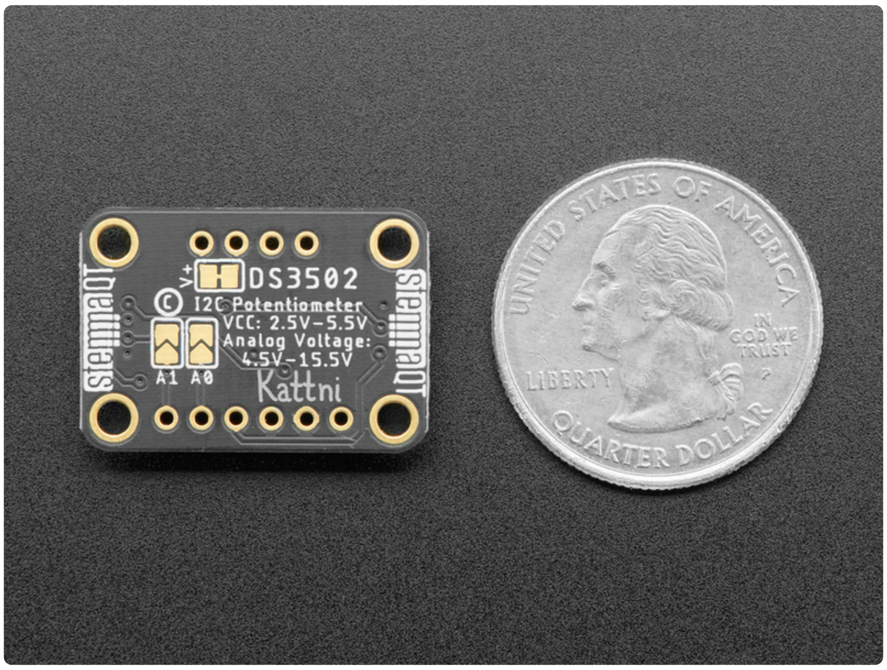
If you're a person like me that gets exhausted turning knobs all day, the DS3502 is just the ticket to calm all your knob-turning related troubles. Instead of having to turn knobs with your HANDS like an ANIMAL, the DS3502 I2C Digital Potentiometer allows you to let your microcontroller adjust the resistance for you!

Now you can free your hands to spin your fidget spinner or or eat a slice of pizza while you're on the phone. Talking over an I2C bus, your Arduino, CircuitPython board, or Python powered computer can talk to the DS3502 and tell it to vary its resistance at your beck and call.



Working with the DS3502 is easy as an I2C controllable pie. We've put it on a breakout PCB with the required support circuitry and SparkFun qwiic (https://adafru.it/Fpw) compatible **STEMMA QT** (https://adafru.it/Ft4) 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.

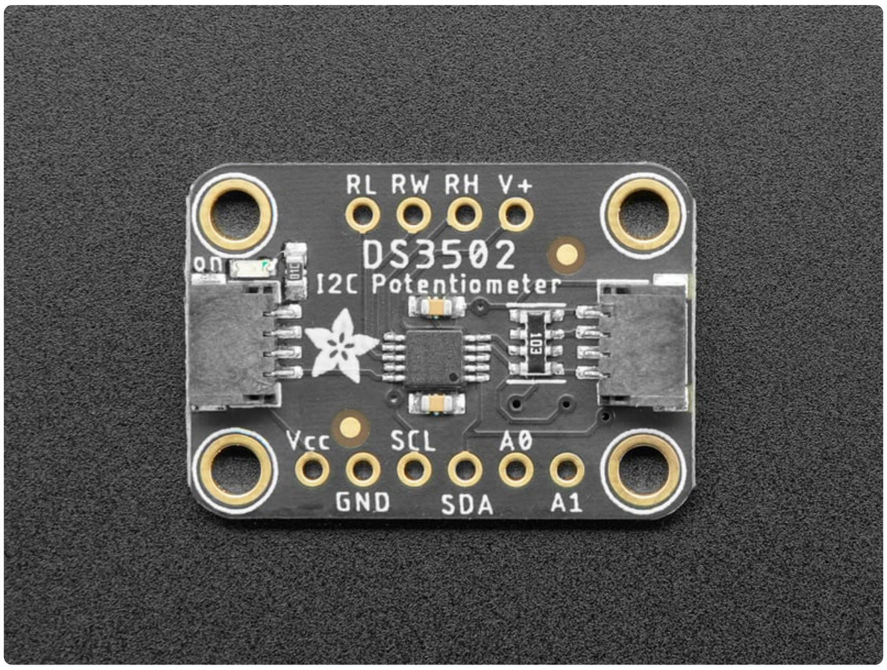
The DS3502 is a simple chip that does one thing well and so it's very easy to work with. Wire it up and set the value for the wiper, that's it. To make things even easier, we've gone and written Arduino and CircuitPython/Python 3 drivers to simplify interfacing with your new knob-replacing friend.



"OK, this thing sounds great. Give me some details" you say. OK then, here you go: The wiper value is a 7-bit number meaning there are **128** possible levels of resistance to choose, from **0-10K ohms**. You can even set a default value that will be set on power up. The analog voltage controlled can be from **4.5-15.5V.**

Additionally you can use the address jumpers or pins to set the I2C address to one of four values, allowing you to have four DS3502s on the same I2C bus.

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.

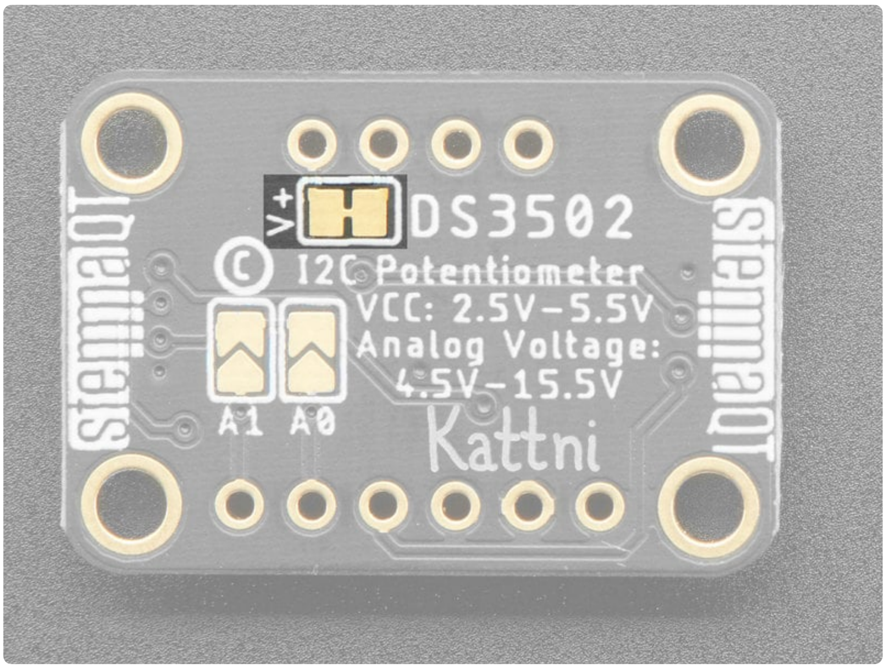
- **Vcc** - this is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
- **GND** - common ground for power and logic

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 (https://adafru.it/Ft4)** - These connectors allow you to connectors to dev boards with **STEMMA QT** connectors or to other things with various associated accessories (https://adafru.it/Ft6)

Resistor Pins

- **RL** is the Low Terminal of the potentiometer, often connected to ground.
- **RW** is the wiper of the potentiometer. As the wiper value is adjusted via I2C, the resistance between **RW** and **RL**/**RH** changes
- **RH** is the High Terminal of the potentiometer, often connected to your high voltage source.
- **V+** is the wiper bias pin and is used to bias the gates of the MOSFETs that are responsible for changing the resistance between **RW** and **RH** or **RL**. If the voltage at **RH** is higher than **VCC**, **V+** must be at the same or higher voltage than **RH**. By default this is connected with a jumper to **RH** but you can cut the solder jumper and wire it directly



Extra Pins

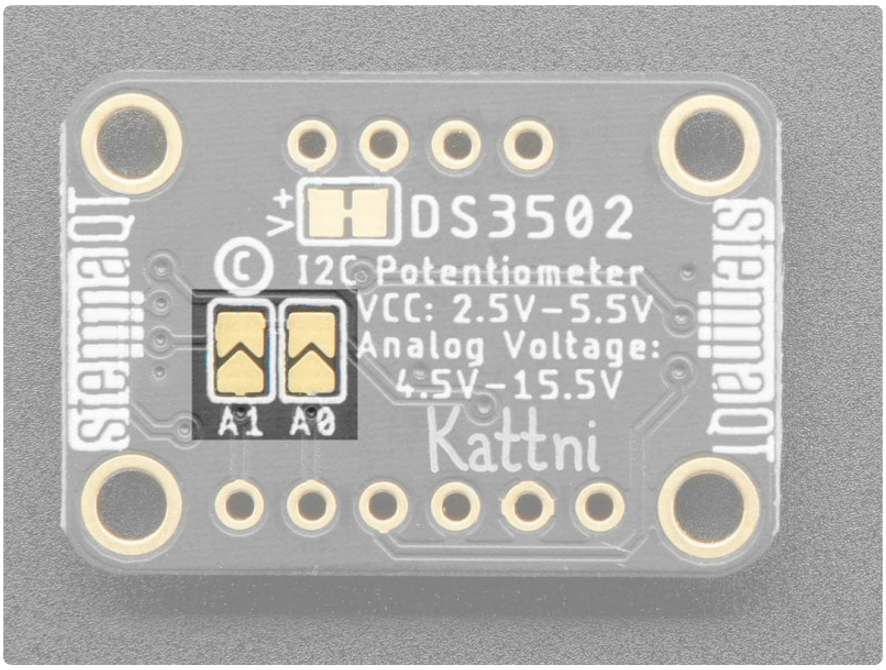
- **A0 and A1** - These are the address select pins.

Since you can only have one device with a given address on an I2C bus, there must be a way to adjust the address if you want to put more than one DS3502 on a shared I2C bus. The **A0/A1** pins set the bottom two bits of the I2C address. There are pull-down resistors on the board so connect them to VDD, you can solder the back jumpers or wire them on a breadboard, to set the bits to '1'. They are read on power up, so de-power and re-power to reset the address

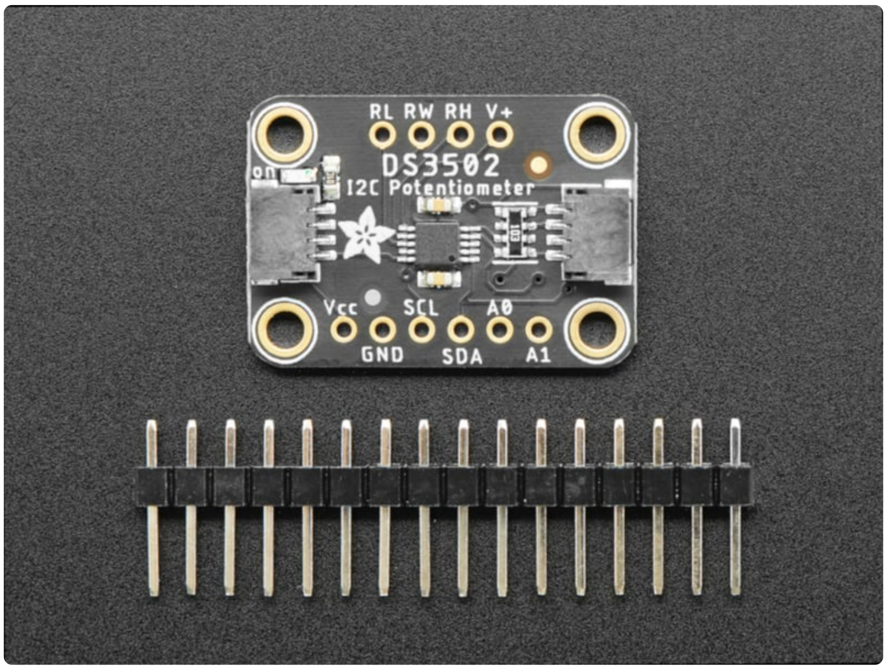
The default address is **0x28** and the address can be calculated by 'adding' the **A0/ A1** to the base of **0x28**

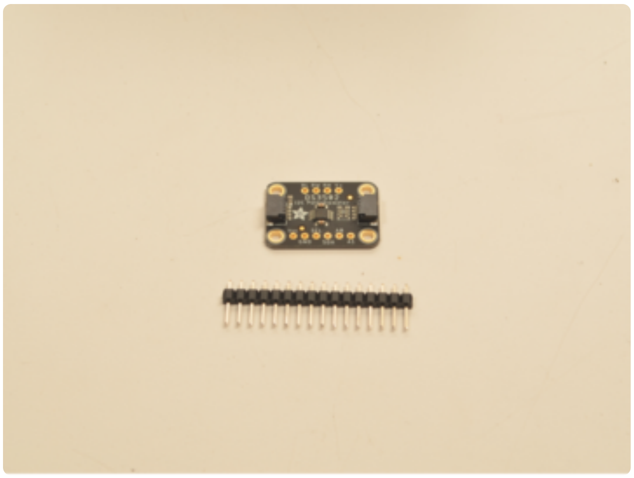
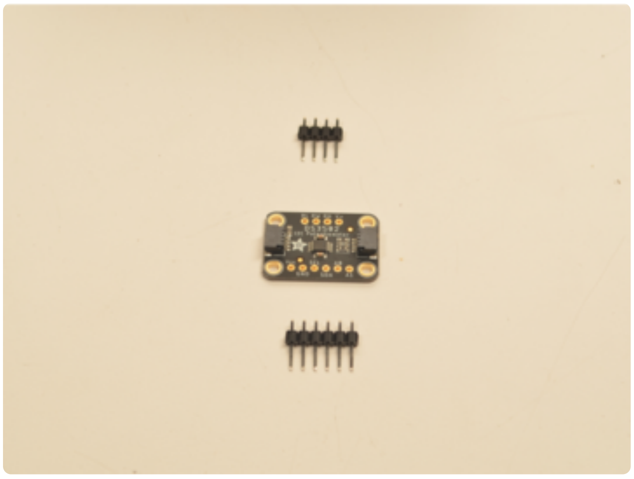
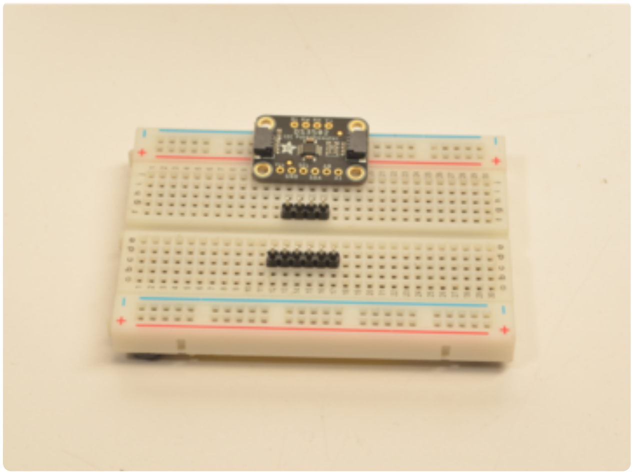
A0 sets the lowest bit with a value of **1**, and **A1** sets the middle bit with a value of **2**. The final address is **0x28** + **A1** + **A0**.

- So for example if only **A0** is tied to VDD, the address is **0x28** + **1** = **0x29**
- If only **A1** is tied to VDD, the address is **0x28** + **2** = **0x2A**
- If **A1** is tied to VDD and **A0** is tied to VDD, the address is **0x28** + **2** + **1** = **0x2B.**



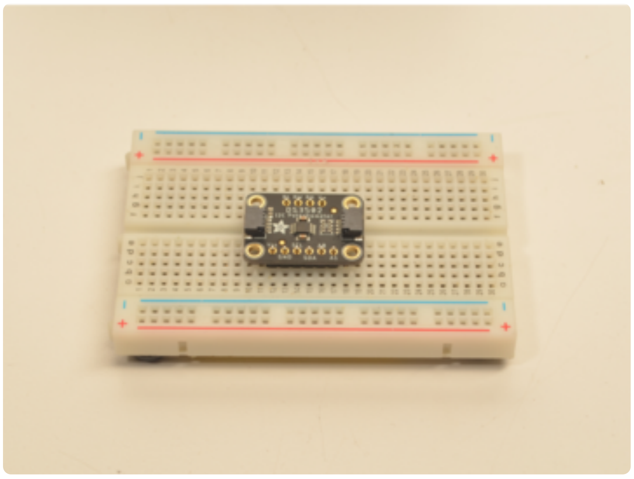
Assembly



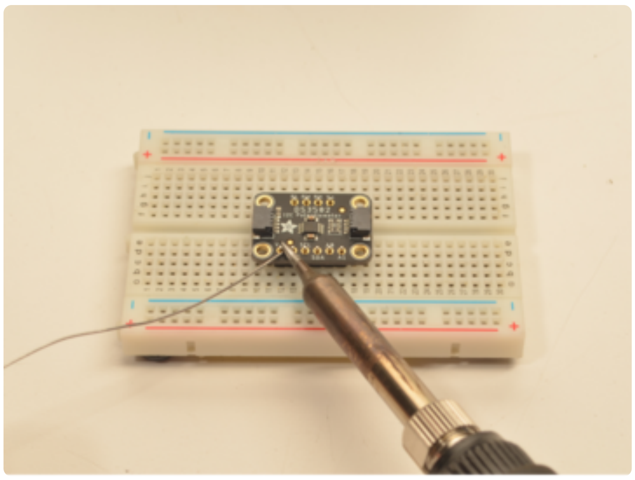
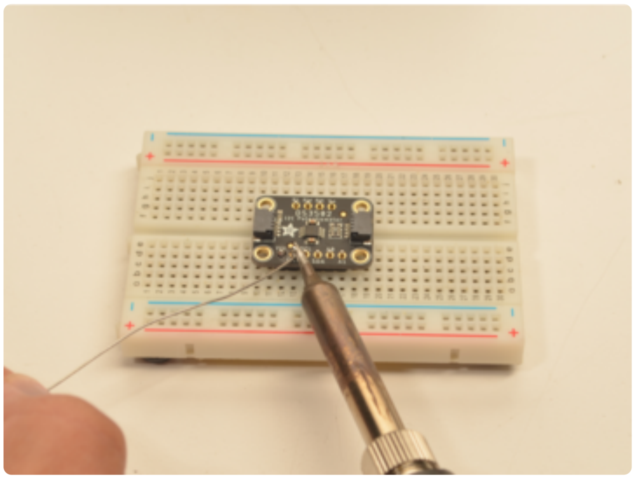
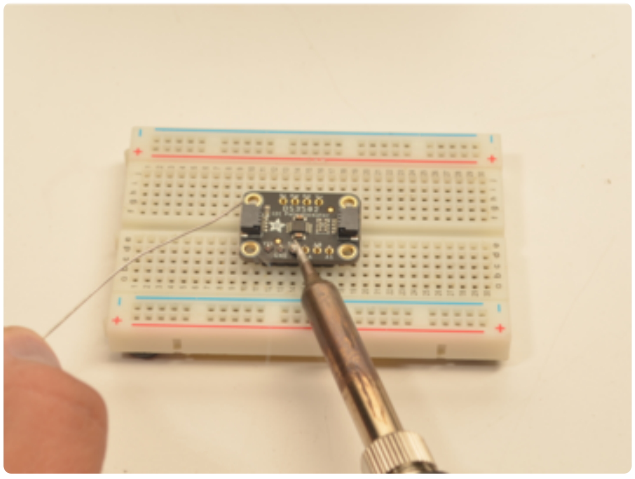
Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**



Add the breakout board:

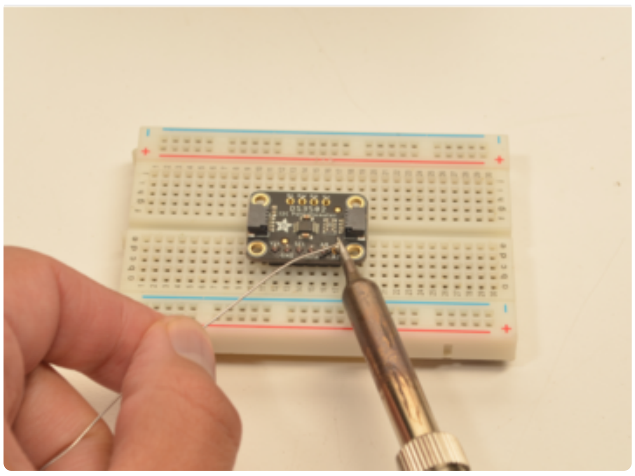
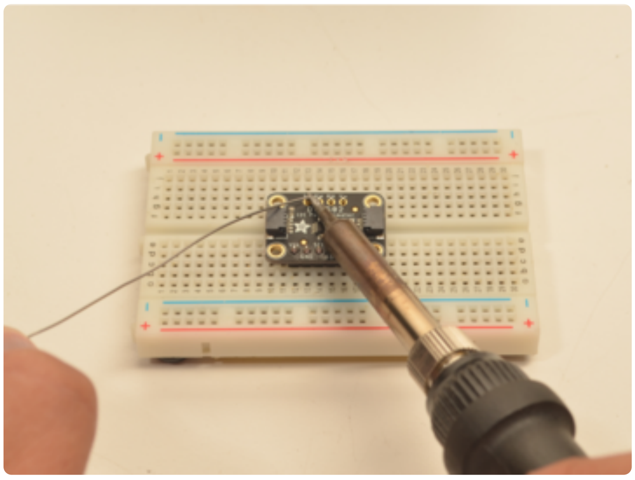
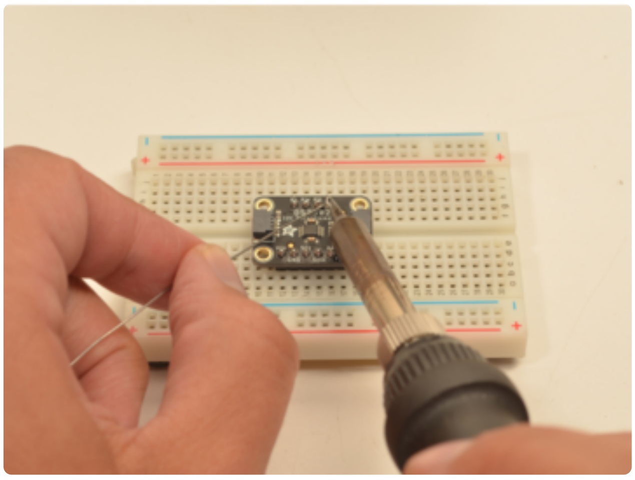
Place the breakout board over the pins so that the short pins poke through the breakout pads

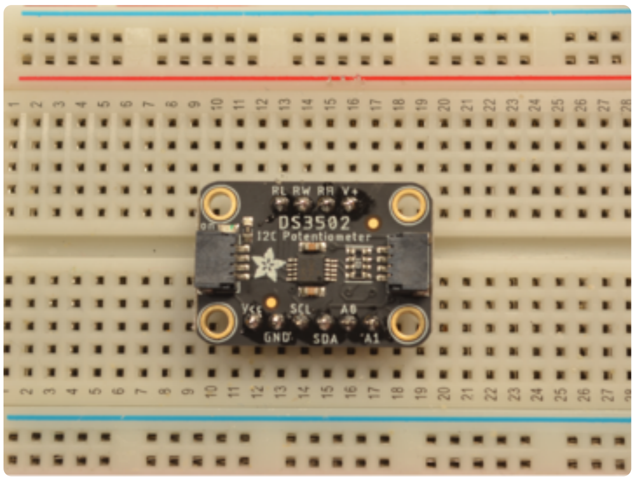




And Solder!

Be sure to solder all 10 pins for reliable electrical contact.

(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk)).

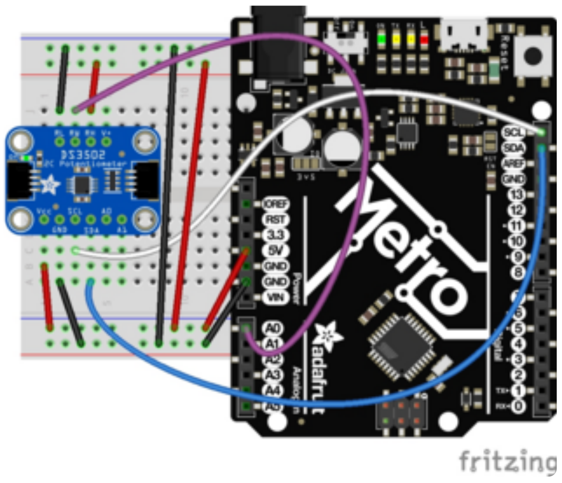


You're done! Check your solder joints visually and continue onto the next steps.

Arduino

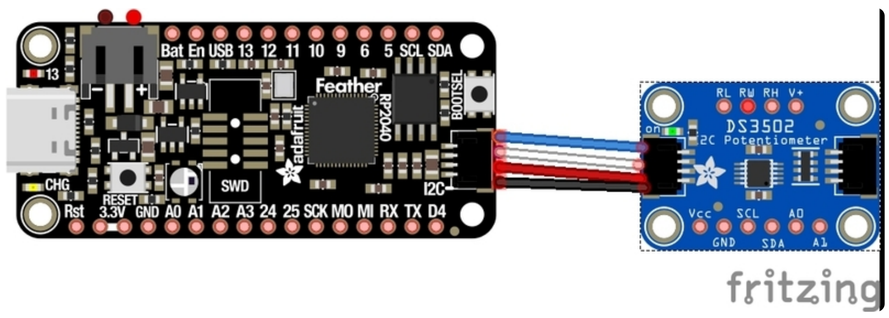
Wiring

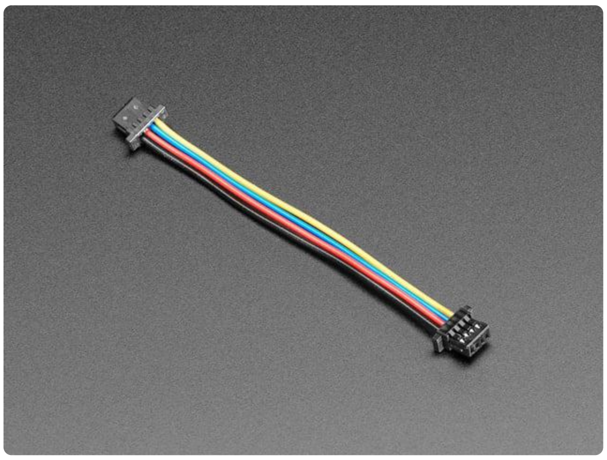
Wiring the DS3502 to communicate with your microcontroller is straight forward forward thanks to the I2C interface. For these examples we can use the Metro or Arduino to measure the voltage changes as the DS3502 adjusts its resistance. The instructions below reference a Metro (http://adafru.it/2488), but the same applies to an Arduino



Connect the **Metro 5V** to **Vcc** on the DS3502
Connect **GND** on the Metro to **GND** on the DS3502
Connect the **SCL** pins on the Metro and DS3502
Connect the **SDA** pins on the Metro and DS3502
Connect **RL** to **GND**
Connect **RH** to **Metro 5V**
Connect **RW** to the **A0** pin on the **Metro**, to allow us to measure the voltage

Using a STEMMA QT cable (http://adafru.it/4399) makes it super easy. Pick a cable length and plug one into a STEMMA QT capable controller and the other end into the DS3502.





STEMMA QT / Qwiic JST SH 4-Pin Cable - 50mm Long

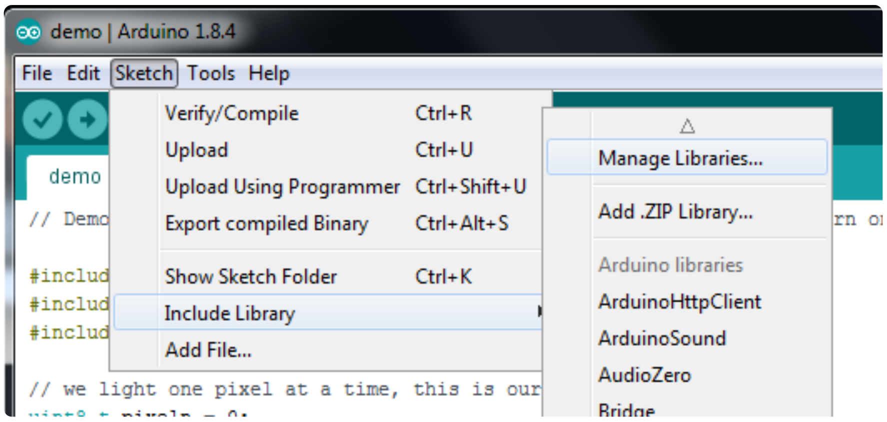
This 4-wire cable is 50mm / 1.9" long and fitted with JST SH female 4-pin connectors on both ends. Compared with the chunkier JST PH these are 1mm pitch instead of 2mm, but...

https://www.adafruit.com/product/4399

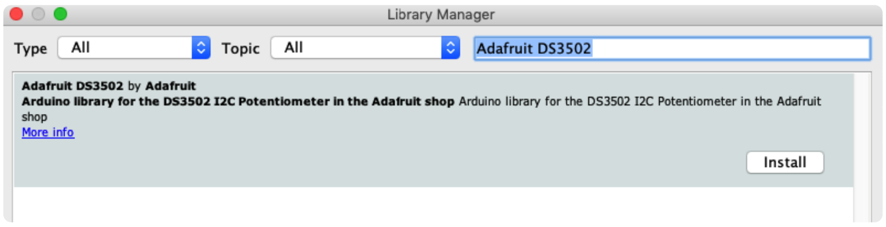
Library Installation

Once wired up, to start using the DS3502, you'll need to install the Adafruit_DS3502 library (https://adafru.it/Ft7). The library is available through the Arduino library manager so we recommend taking that approach.

From the Arduino IDE, open up the Library Manager:

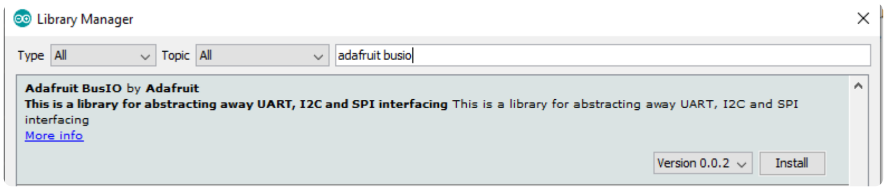


Click the **Manage Libraries ...** menu item, search for **Adafruit DS3502**, and select the **Adafruit DS3502** library and click **Install**:



Adafruit DS3502 by Adafruit
Arduino library for the DS3502 I2C Potentiometer in the Adafruit shop Arduino library for the DS3502 I2C Potentiometer in the Adafruit shop
More info

Follow the same process to install the **Adafruit BusIO** library.



Adafruit BusIO by Adafruit
This is a library for abstracting away UART, I2C and SPI interfacing This is a library for abstracting away UART, I2C and SPI interfacing
More info

Load Example

Open up **File -> Examples -> Adafruit DS3502 -> ds3502_test** and upload to your Metro wired up to the DS3502 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 115200 baud, similar to this:

```
Adafruit DS3502 Test
Found DS3502 chip
Wiper voltage with wiper set to 0: 0.00 V

Wiper voltage with wiper set to 63: 2.48 V

Wiper voltage with wiper set to 127: 5.00 V
```

Example Code

The following code is part of the standard library and illustrates the basic function of using the variable resistance of the DS3502 to change the voltage measured at pin A0:

```
#include <Adafruit_DS3502.h>

Adafruit_DS3502 ds3502 = Adafruit_DS3502();
/* For this example, make the following connections:
    * DS3502 RH to 5V
    * DS3502 RL to GND
    * DS3502 RW to the pin specified by WIPER_VALUE_PIN
*/

#define WIPER_VALUE_PIN A0

void setup() {
  Serial.begin(115200);
  // Wait until serial port is opened
  while (!Serial) { delay(1); }

  Serial.println("Adafruit DS3502 Test");

  if (!ds3502.begin()) {
    Serial.println("Couldn't find DS3502 chip");
    while (1);
  }
  Serial.println("Found DS3502 chip");
}

void loop() {
  Serial.print("Wiper voltage with wiper set to 0: ");
  ds3502.setWiper(0);
  float wiper_value = analogRead(WIPER_VALUE_PIN);
  wiper_value *= 5.0;
  wiper_value /= 1024;
  Serial.print(wiper_value);
  Serial.println(" V");

  Serial.println();
  delay(1000);

  Serial.print("Wiper voltage with wiper set to 63: ");
  ds3502.setWiper(63);
  wiper_value = analogRead(WIPER_VALUE_PIN);
  wiper_value *= 5.0;
  wiper_value /= 1024;
  Serial.print(wiper_value);
  Serial.println(" V");

  Serial.println();
  delay(1000);

  Serial.print("Wiper voltage with wiper set to 127: ");
  ds3502.setWiper(127);
  wiper_value = analogRead(WIPER_VALUE_PIN);
  wiper_value *= 5.0;
  wiper_value /= 1024;
  Serial.print(wiper_value);
  Serial.println(" V");

  Serial.println();
  delay(1000);
}
```

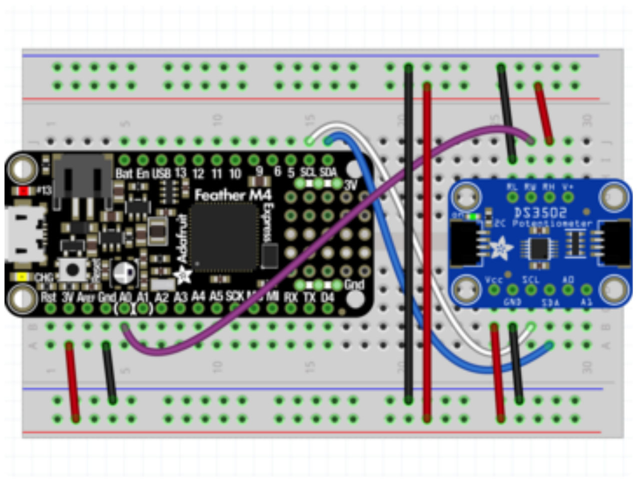
Arduino Docs

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

Python & CircuitPython

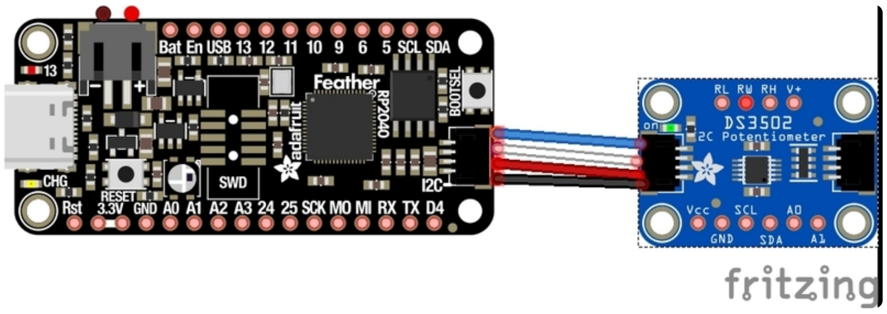
CircuitPython Microcontroller Wiring

Wiring the DS3502 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 DS3502 adjusts its resistance. The instructions below reference a Feather (http://adafru.it/3857), but the same applies to a Metro.



Connect the **Feather 3.3V** to **Vcc** on the DS3502
Connect **GND** on the Feather to **GND** on the DS3502
Connect the **SCL** pins on the Feather and DS3502
Connect the **SDA** pins on the Feather and DS3502
Connect **RL** to **GND**
Connect **RH** to **Feather 3.3V**
Connect **RW** to the **A0** pin on the **Feather**, to allow us to measure the voltage

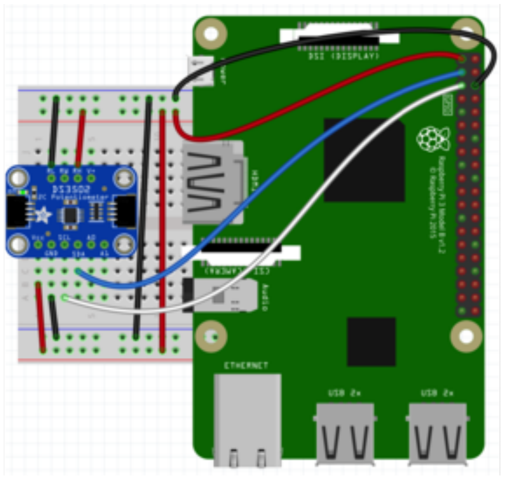
Using a STEMMA QT cable (http://adafru.it/4399) makes it super easy. Pick a cable length and plug one into a STEMMA QT capable controller and the other end into the DS3502.



Python Computer Wiring

Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi (http://adafru.it/3055). 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:



Pi 3V3 to **sensor VIN**
Pi GND to **sensor GND**
Pi SCL to **sensor SCL**
Pi SDA to **sensor SDA**
Pi GND to **sensor RL**
Pi 3V3 to **sensor RH**
Multimeter Positive Lead to **sensor RW**
Multimeter Negative Lead to **sensor GND**

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 DS3502's **RW** pin, you will need to use a multimeter to measure the voltage between the **RW** pin and **GND.**

CircuitPython Installation of DS3502 Library

You'll need to install the Adafruit CircuitPython DS3502 (https://adafru.it/Ft8) 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/uap). Our CircuitPython starter guide has a great page on how to install the library bundle (https://adafru.it/ABU).

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

- **adafruit_ds3502.mpy**
- **adafruit_bus_device**
- **adafruit_register**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_ds3502.mpy, adafruit_bus_device**, and **adafruit_register** files and folders copied over.

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

Python Installation of DS3502 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-ds3502`

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!

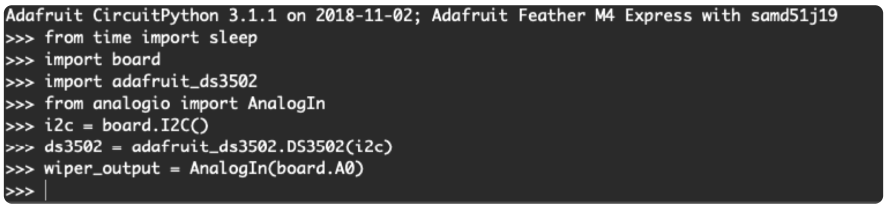
CircuitPython Usage

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:

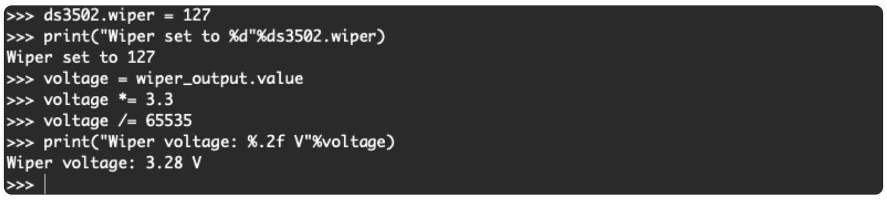
```
from time import sleep
import board
import adafruit_ds3502
from analogio import AnalogIn

i2c = board.I2C()
ds3502 = adafruit_ds3502.DS3502(i2c)
wiper_output = AnalogIn(board.A0)
```



```
Adafruit CircuitPython 3.1.1 on 2018-11-02; Adafruit Feather M4 Express with samd51j19
>>> from time import sleep
>>> import board
>>> import adafruit_ds3502
>>> from analogio import AnalogIn
>>> i2c = board.I2C()
>>> ds3502 = adafruit_ds3502.DS3502(i2c)
>>> wiper_output = AnalogIn(board.A0)
>>> 
```

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

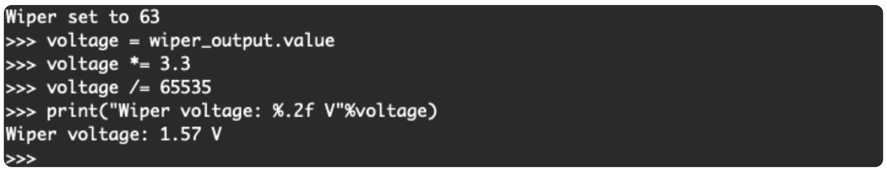
```
ds3502.wiper = 127
print("Wiper set to %d"%ds3502.wiper)
voltage = wiper_output.value
voltage *= 3.3
voltage /= 65535
print("Wiper voltage: %.2f V"%voltage)
```



```
>>> ds3502.wiper = 127
>>> print("Wiper set to %d"%ds3502.wiper)
Wiper set to 127
>>> voltage = wiper_output.value
>>> voltage *= 3.3
>>> voltage /= 65535
>>> print("Wiper voltage: %.2f V"%voltage)
Wiper voltage: 3.28 V
>>> 
```

We can then change the wiper value and take another reading to see how the resulting voltage on the **RW** pin has changed

```
ds3502.wiper = 63
print("Wiper set to %d"%ds3502.wiper)
voltage = wiper_output.value
voltage *= 3.3
voltage /= 65535
print("Wiper voltage: %.2f V"%voltage)
```



```
Wiper set to 63
>>> voltage = wiper_output.value
>>> voltage *= 3.3
>>> voltage /= 65535
>>> print("Wiper voltage: %.2f V"%voltage)
Wiper voltage: 1.57 V
>>>
```

Full CircuitPython Example Code

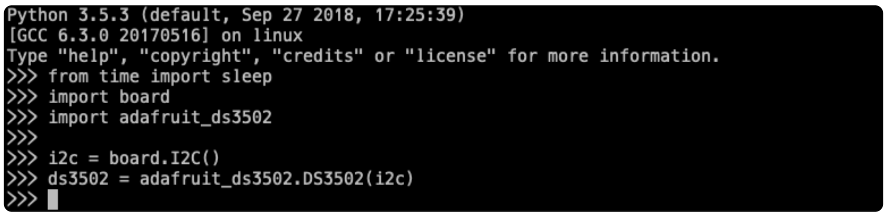
```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

from time import sleep
import board
from analogio import AnalogIn
import adafruit_ds3502

####### 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 "ds3502_blinka_simpletest.py" example

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
ds3502 = adafruit_ds3502.DS3502(i2c)
wiper_output = AnalogIn(board.A0)

while True:
    ds3502.wiper = 127
    print("Wiper set to %d" % ds3502.wiper)
    voltage = wiper_output.value
    voltage *= 3.3
    voltage /= 65535
    print("Wiper voltage: %.2f V" % voltage)
    print("")
    sleep(1.0)

    ds3502.wiper = 0
    print("Wiper set to %d" % ds3502.wiper)
    voltage = wiper_output.value
    voltage *= 3.3
    voltage /= 65535
    print("Wiper voltage: %.2f V" % voltage)
    print("")
    sleep(1.0)

    ds3502.wiper = 63
    print("Wiper set to %d" % ds3502.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

```
from time import sleep
import board
import adafruit_ds3502

i2c = board.I2C()
ds3502 = adafruit_ds3502.DS3502(i2c)
```



```
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from time import sleep
>>> import board
>>> import adafruit_ds3502
>>>
>>> i2c = board.I2C()
>>> ds3502 = adafruit_ds3502.DS3502(i2c)
>>>
```

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

```
ds3502.wiper = 127
```

```
>>> ds3502.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 the voltage on the **RH** pin which will be the voltage level of your device, most likely **3.3V**

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

```
ds3502.wiper = 63
```

```
>>> ds3502.wiper = 63
```

The voltage on the **RW** pin should be approximately half of the voltage at the **RH** pin, likely around **1.6V**

Full Python Example Code

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

from time import sleep
import board
import adafruit_ds3502

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
ds3502 = adafruit_ds3502.DS3502(i2c)

# 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.
while True:
    ds3502.wiper = 127
    print("Wiper value set to 127")
    sleep(5.0)

    ds3502.wiper = 0
    print("Wiper value set to 0")
    sleep(5.0)

    ds3502.wiper = 63
    print("Wiper value set to 63")
    sleep(5.0)
```

Python Docs

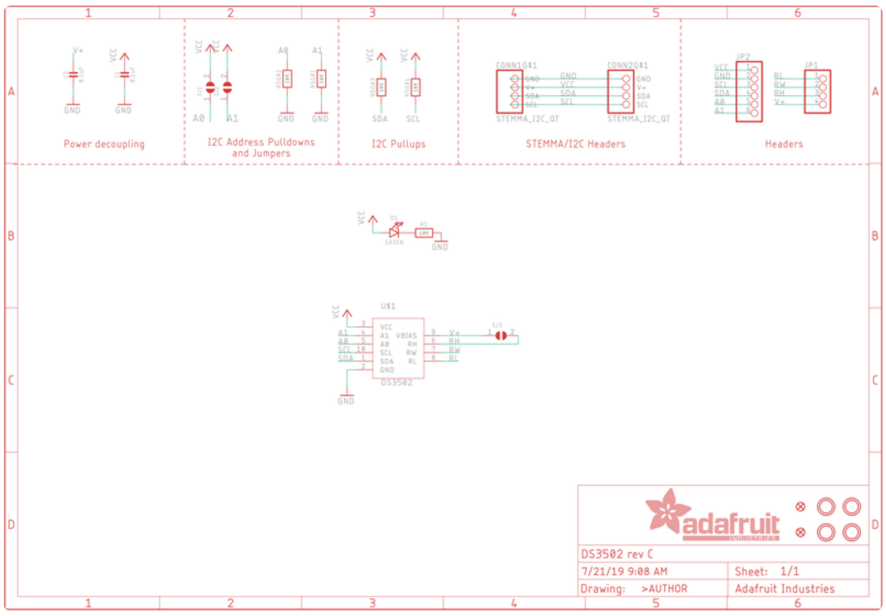
Python Docs (https://adafru.it/Oka)

Downloads

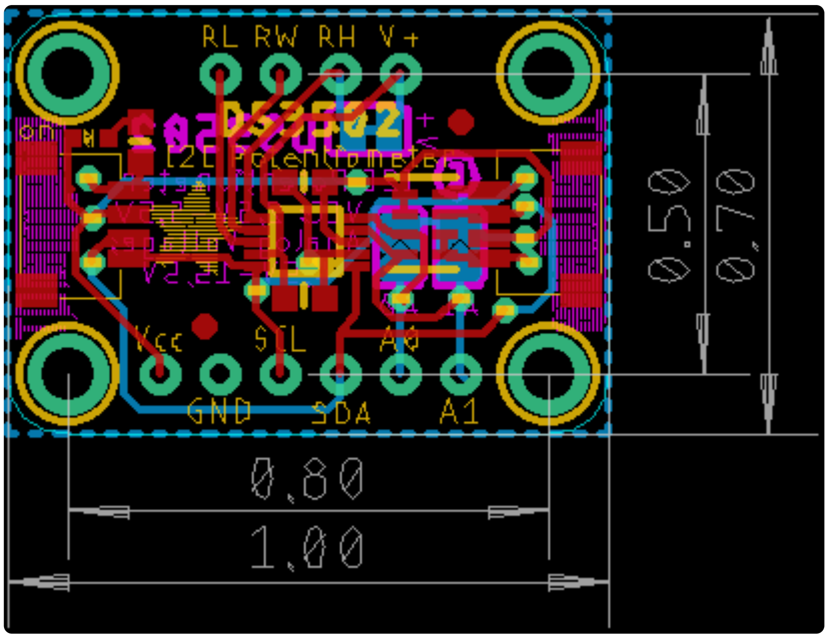
Files

- DS3502 Datasheet (https://adafru.it/Ft9)
- EagleCAD files on GitHub (https://adafru.it/Fta)
- 3D CAD files on GitHub (https://adafru.it/FGL)
- Fritzing object in Adafruit Fritzing Library (https://adafru.it/Ftb)

Schematic



Fab Print


RL RW RH V+
DS3502
I2C Potentiometer
Vcc
SCL
A0
GND
SDA
A1
0.50
0.70
0.80
1.00