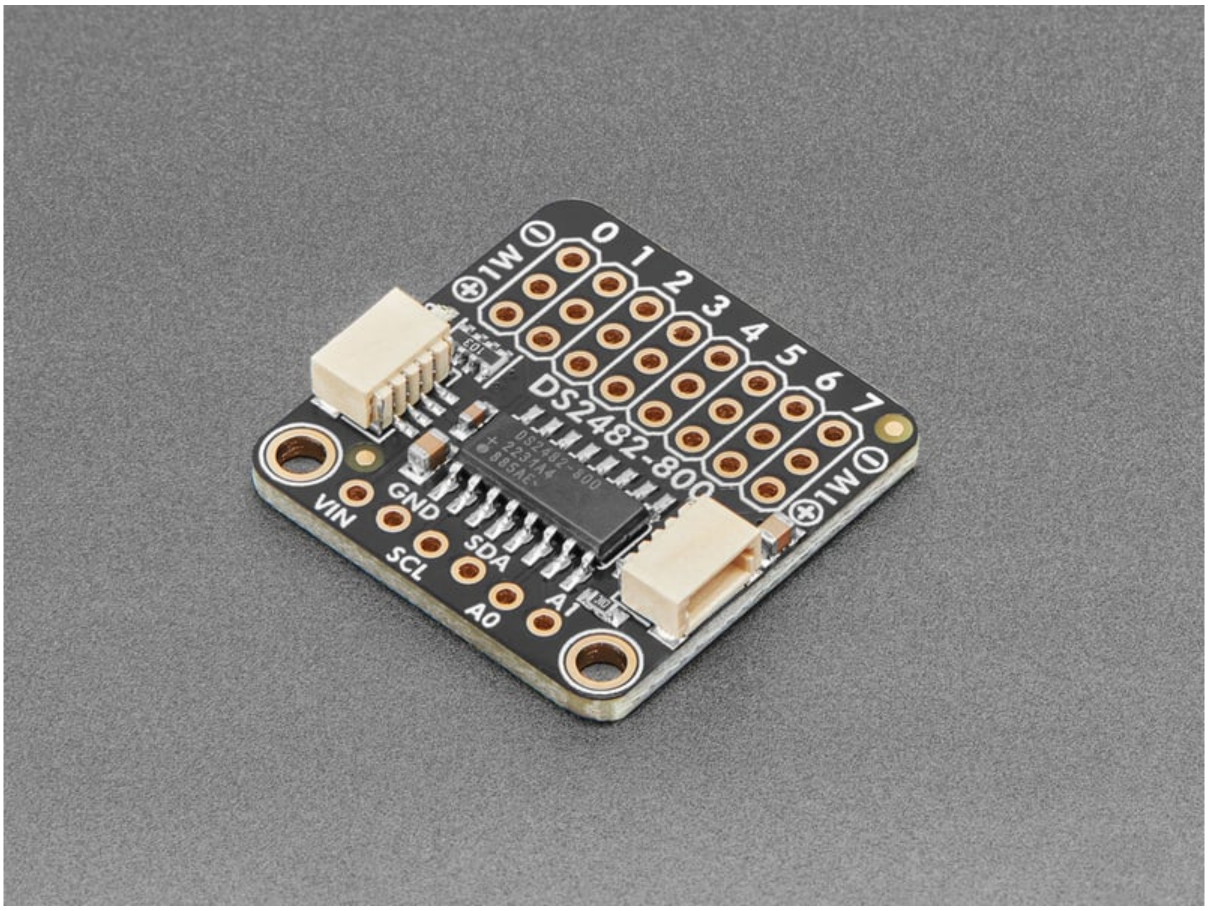
Adafruit DS2482S-800 8 Channel I2C to 1-Wire Bus Adapter

Created by Liz Clark

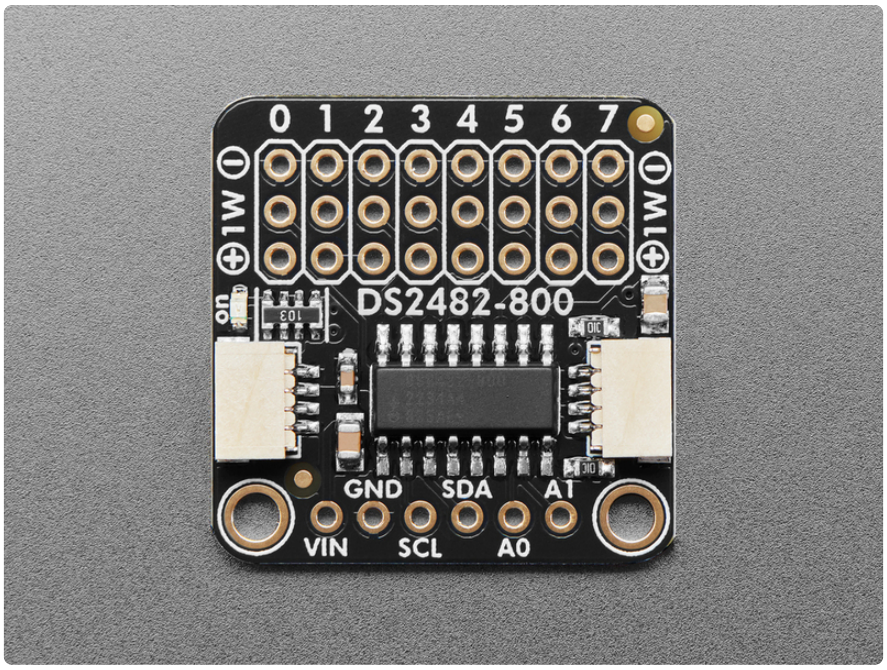


https://learn.adafruit.com/adafruit-ds2482s-800-8-channel-i2c-to-1-wire-bus-adapter

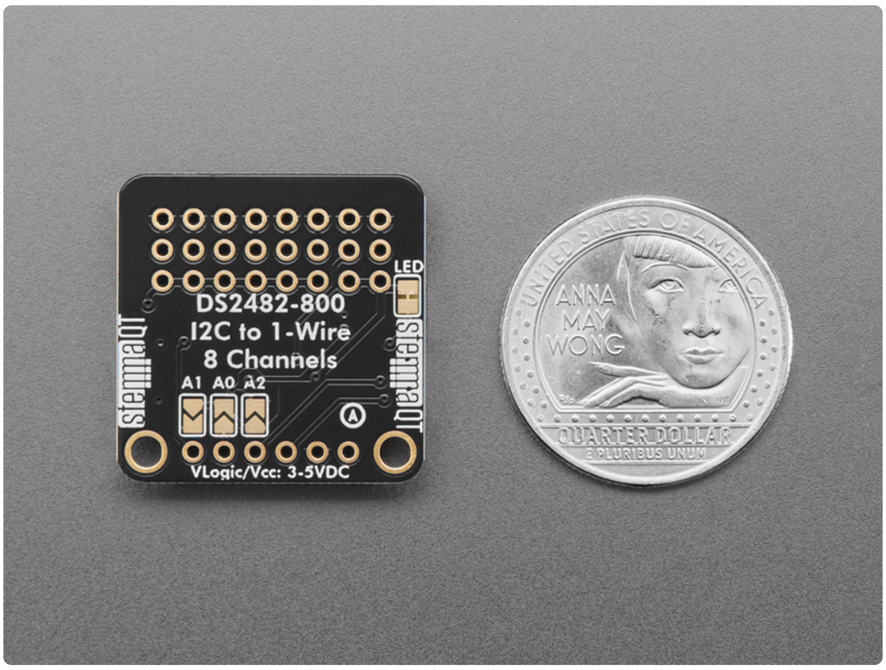
Last updated on 2024-09-18 02:07:15 PM EDT

Table of Contents

Overview

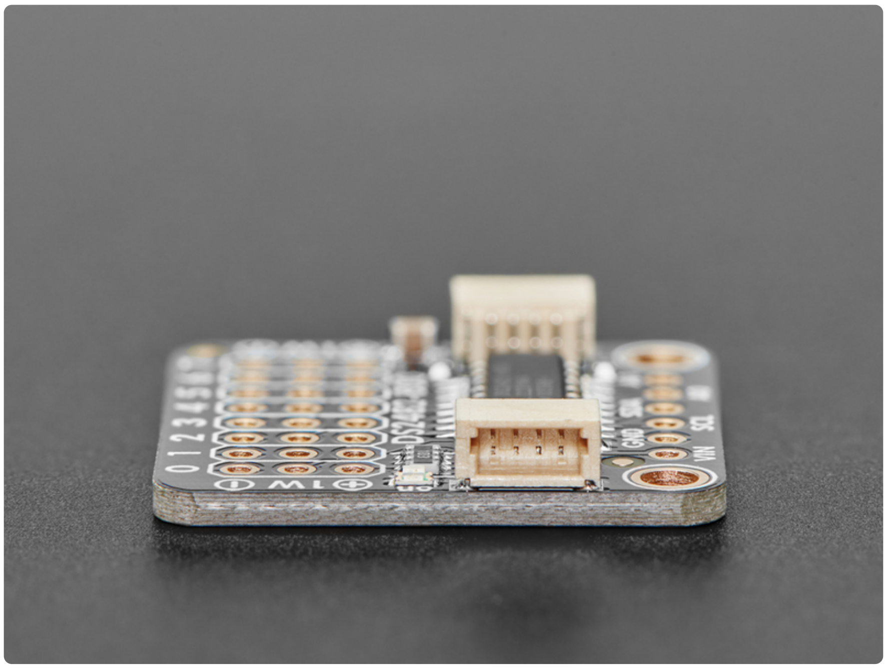


The DS2482S-800 (https://adafru.it/1a7j) is a Stemma QT board that uses a I2C-to-1Wire controller chip, with 8 selectable channels, built-in parasitic-power-pullups, adjustable I2C address and a wide operating voltage range! You can easily connect it to an existing I2C bus and then use the breakout pads to attach multiple DS18B20's, or pair it with our 1-Wire chaining breakouts (http://adafru.it/5971) for fancier experimentation.

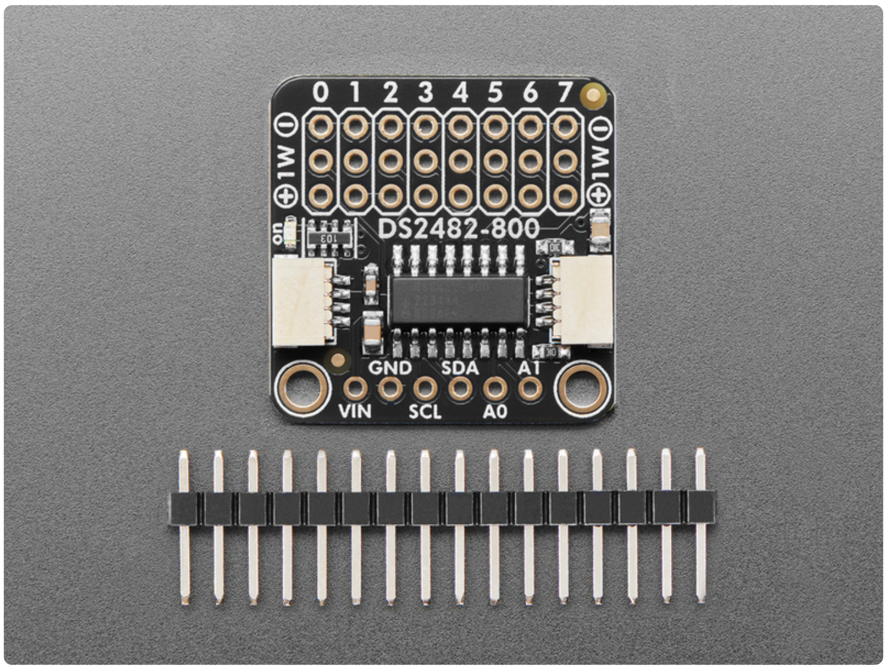


You're probably familiar with the 'top three' electronics protocols: I2C, SPI and UART. Perhaps you're aware of a fourth one, called "1-Wire" which was invented by Dallas

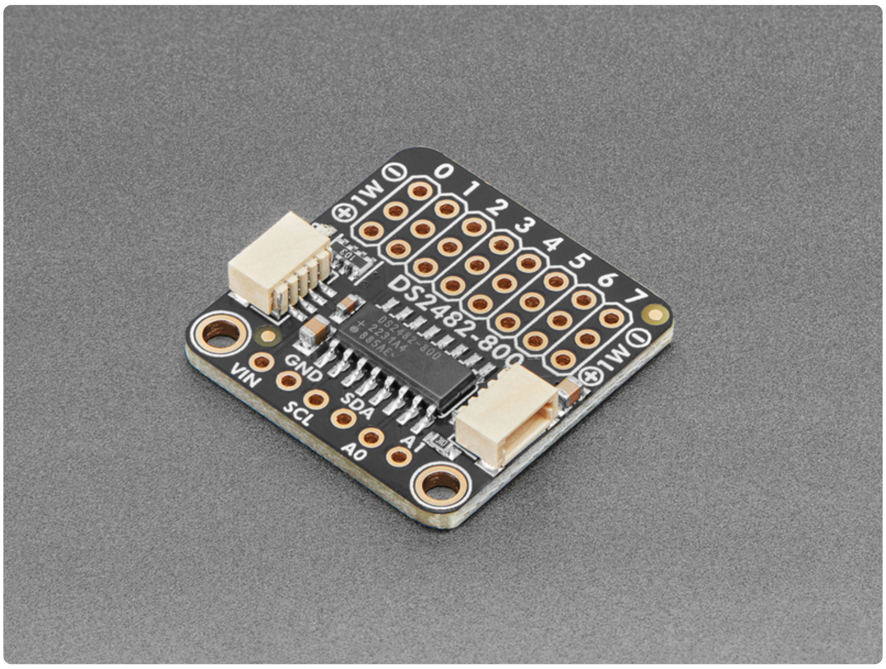
Semiconductor (https://adafru.it/1a2Z) (which became Maxim, which became Analog Devices). As you may expect, this protocol uses a single data wire plus a ground wire (and an optional power wire) to connect to any number of sensors or memory chips that all share the same bus.



In theory, there's a lot of different 1-Wire devices out there, but in reality almost everyone uses 1-Wire for DS18B20 temperature sensors. the long wire lengths and ease of 'chaining' by sharing a single bus wire makes it perfectly fine for this purpose. You can bit-bang 1-Wire on most microcontrollers, and some SBCs like Raspberry Pi have kernel module support (https://adafru.it/1a30). But are lots of chips or firmware stacks without 1-Wire capability, or maybe you want to use 1-Wire devices on your desktop computer or other SBC with I2C.



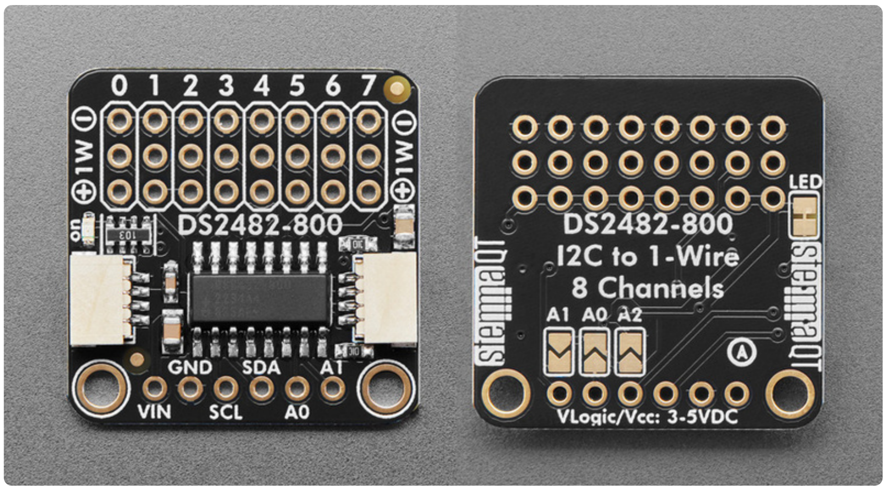
You might be wondering: if you can connect multiple 1-Wire devices on a single wire, then why have 8 channels on this breakout? Reason is: if you have 8 temperature sensors, there's no easy way to know which one is which other than connecting them one at a time and reading the unique ROM identifier. With this breakout you can select the channel so you know for sure which location is being measured, even if the sensor is swapped out later.



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. The STEMMA QT connectors (https://adafru.it/JqB) on either side are compatible with the SparkFun Qwiic (https://adafru.it/Fpw) I2C connectors.

STEMMA QT allows you to make solderless connections between your development board and the DS2482S-800 or to chain it with a wide range of other sensors and accessories using a **compatible cable** (https://adafru.it/JnB). You can power the board directly from the STEMMA QT cable, whether 3V or 5V, that voltage will also be the power output for the 1-Wire peripherals.

Pinouts



The default I2C address is **0x18**.

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 development boards with **STEMMA QT** (Qwiic) connectors or to other things with various associated accessories (https://adafru.it/Ft6).

1-Wire Header Pins

At the top edge of the board are 8 1x3 header pins. These pins let you connect 8 different 1-wire sensors to the breakout board. Each set of 3 pins (+, **1W** and -) are labeled 0-7. These labels denote the channel number that you'll use in the code.

- + - the power connection for the 1-wire sensor. It shares the same voltage level as **VIN**.

- **1W** - the signal connection for the 1-wire line
- **-** - the ground connection for the 1-wire sensor

Address Pins

On the back of the board are **three address jumpers**, labeled **A0**, **A1**, and **A2**. These jumpers allow you to chain up to 8 of these boards on the same pair of I2C clock and data pins. To do so, you solder the jumpers "closed" by connecting the two pads.

On the front of the board are **two address pins**, labeled **A0** and **A1**. Just like the jumpers, these pins allow you to change the I2C address to connect multiple boards by connecting them to **VIN**. **There is no broken out pin for A2**.

The default I2C address is **0x18**. The other address options can be calculated by "adding" the **A0/A1/A2** to the base of **0x18**.

A0 sets the lowest bit with a value of **1**, **A1** sets the next bit with a value of **2** and **A2** sets the next bit with a value of **4.** The final address is **0x18** + **A2** + **A1** + **A0** which would be **0x1F**.

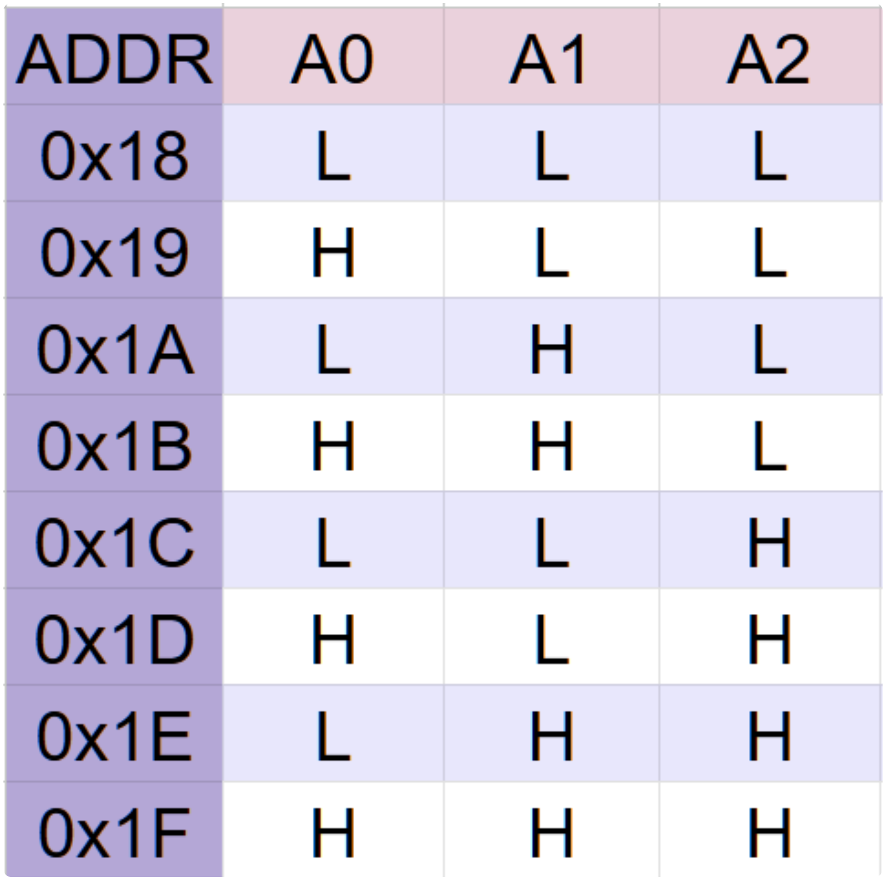
So for example if **A2** is soldered closed and **A0** is soldered closed, the address is **0x18** + **4** + **1** = **0x1D**.

If only **A0** is soldered closed, the address is **0x18** + **1** = **0x19**

If only **A1** is soldered closed, the address is **0x18** + **2** = **0x1A**

If only **A2** is soldered closed, the address is **0x18** + **4** = **0x1C**

The table below shows all possible addresses, and whether the pin(s) should be high (closed) or low (open).



ADDR	A0	A1	A2
0x18	L	L	L
0x19	H	L	L
0x1A	L	H	L
0x1B	H	H	L
0x1C	L	L	H
0x1D	H	L	H
0x1E	L	H	H
0x1F	H	H	H

Power LED and Jumper

- **Power LED** - on the left side of the board, above the STEMMA QT connector, is the power LED, labeled **on**. It is the green LED.
- **LED jumper** - on the back of the board is a jumper for the power LED. It is labeled **LED** on the board silk. If you want to disable the power LED, cut the trace on this jumper.

CircuitPython and Python

It's easy to use the **DS2482S-800** with Python or CircuitPython, and the Adafruit_CircuitPython_DS248x (https://adafru.it/1a31) module. This module allows you to easily write Python code to read 1-Wire sensors over I2C.

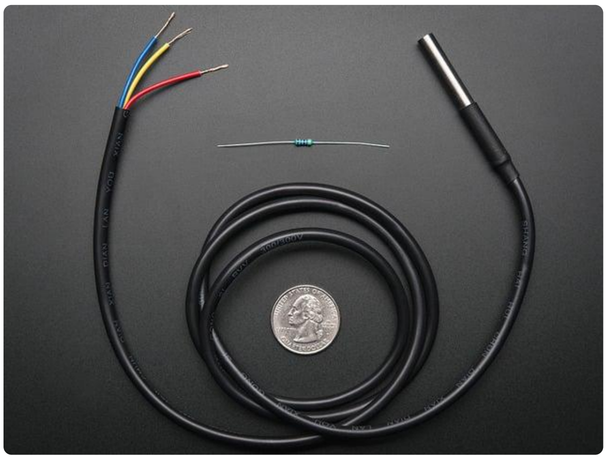
You can use this driver with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

You'll need two DS18B20 sensors to use this example with the breakout:

DS18B20 Digital temperature sensor + extras

These 1-wire digital temperature sensors are fairly precise (±0.5°C over much of the range) and can give up to 12 bits of precision from the onboard digital-to-analog...

https://www.adafruit.com/product/374



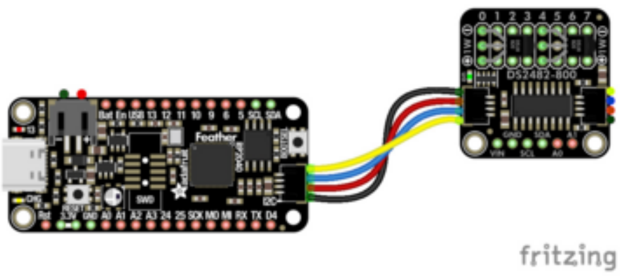
Waterproof 1-Wire DS18B20 Digital temperature sensor

This is a pre-wired and waterproofed (with heat shrink) version of a 1 Wire DS18B20 sensor. Handy for when you need to measure something far away, or in wet conditions. While the...

https://www.adafruit.com/product/381

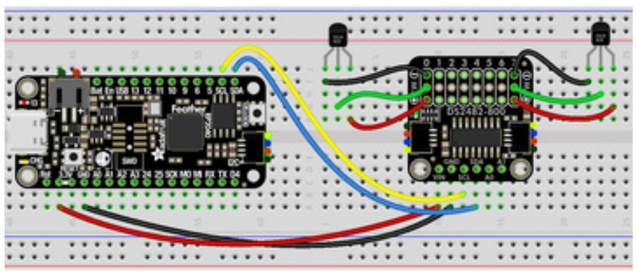
CircuitPython Microcontroller Wiring

First, wire up two DS18B20 sensors to the breakout exactly as follows. Then, wire up the breakout to your board. The following is the breakout wired to a Feather RP2040 with two DS18B20 sensors using the STEMMA connector. The DS18B20 sensor are connected directly to the breakout.



Breakout STEMMA VIN to Feather STEMMA 3.3V (red wire)
Breakout STEMMA GND to Feather STEMMA GND (black wire)
Breakout STEMMA SDA to Feather STEMMA SDA (blue wire)
Breakout STEMMA SCL to Feather STEMMA SCL (yellow wire)
Breakout channel 0 - to DS18B20 GND
Breakout channel 0 1W to DS18B20 signal
Breakout channel 0 + to DS18B20 VIN
Breakout channel 4 - to DS18B20 GND
Breakout channel 4 1W to DS18B20 signal
Breakout channel 4 + to DS18B20 VIN

The following is the breakout wired to a Feather RP2040 and two DS18B20 sensors using a solderless breadboard:

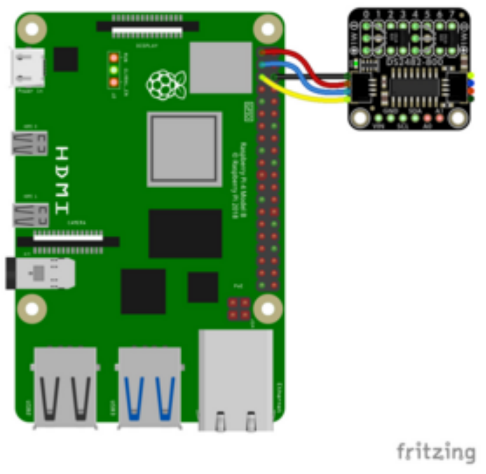


Breakout VIN to Feather 3.3V (red wire)
Breakout GND to Feather GND (black wire)
Breakout SDA to Feather SDA (blue wire)
Breakout SCL to Feather SCL (yellow wire)
Breakout channel 0 - to DS18B20 GND (black wire)
Breakout channel 0 1W to DS18B20 signal (green wire)
Breakout channel 0 + to DS18B20 VIN (red wire)
Breakout channel 4 - to DS18B20 GND (black wire)
Breakout channel 4 1W to DS18B20 signal (green wire)
Breakout channel 4 + to DS18B20 VIN (red wire)

Python Computer Wiring

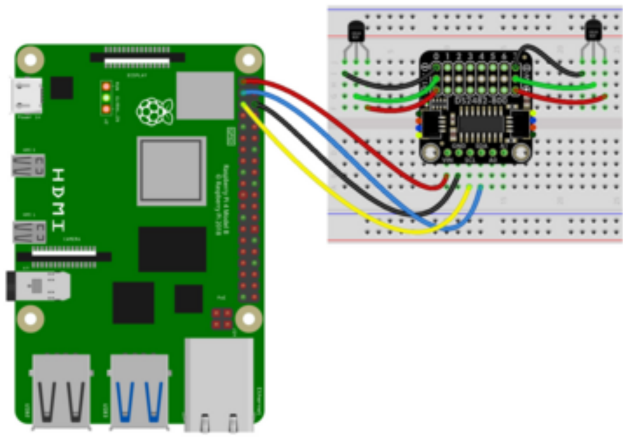
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C using the STEMMA connector. The DS18B20 sensors are connected directly to the breakout on channel 0 and channel 4.



Breakout STEMMA VIN to **Pi 3.3V (red wire)**
Breakout STEMMA GND to **Pi GND (black wire)**
Breakout STEMMA SDA to **Pi SDA (blue wire)**
Breakout STEMMA SCL to **Pi SCL (yellow wire)**
Breakout channel 0 - to **DS18B20 GND**
Breakout channel 0 1W to **DS18B20 signal**
Breakout channel 0 + to **DS18B20 VIN**
Breakout channel 4 - to **DS18B20 GND**
Breakout channel 4 1W to **DS18B20 signal**
Breakout channel 4 + to **DS18B20 VIN**

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



Breakout VIN to **Pi 3.3V (red wire)**
Breakout GND to **Pi GND (black wire)**
Breakout SDA to **Pi SDA (blue wire)**
Breakout SCL to **Pi SCL (yellow wire)**
Breakout channel 0 - to **DS18B20 GND (black wire)**
Breakout channel 0 1W to **DS18B20 signal (green wire)**
Breakout channel 0 + to **DS18B20 VIN (red wire)**
Breakout channel 4 - to **DS18B20 GND (black wire)**
Breakout channel 4 1W to **DS18B20 signal (green wire)**
Breakout channel 4 + to **DS18B20 VIN (red wire)**

Python Installation of DS248x 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-ds248x`

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!

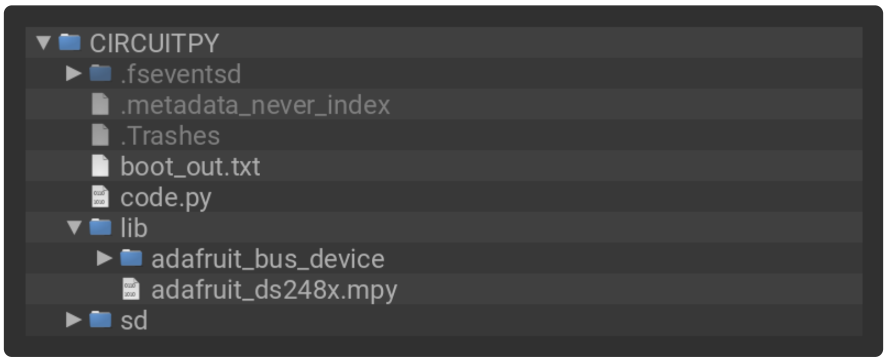
CircuitPython Usage

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_DS248x** 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 folder and file:

- **adafruit_bus_device/**
- **adafruit_ds248x.mpy**



```
▼ CIRCUITPY
    ▶ .fseventsd
      .metadata_never_index
      .Trashes
      boot_out.txt
      code.py
    ▼ lib
        ▶ adafruit_bus_device
          adafruit_ds248x.mpy
    ▶ sd
```

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`

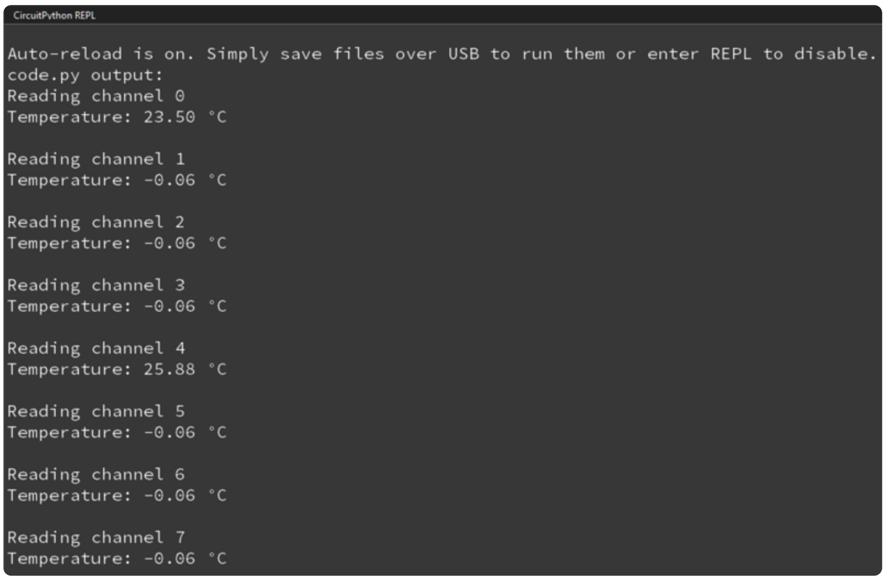
Example Code

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

"""Adafruit DS2482S-800 8-Channel DS18B20 Example"""

import time
import board
from adafruit_ds248x import Adafruit_DS248x

# Initialize I2C bus and DS248x
i2c = board.STEMMA_I2C()
ds248x = Adafruit_DS248x(i2c)

while True:
    for i in range(8):
        ds248x.channel = i
        print(f"Reading channel {ds248x.channel}")
        temperature = ds248x.ds18b20_temperature()
        print(f"Temperature: {temperature:.2f} °C")
        print()
        time.sleep(1)
```

First, the DS2482S-800 is instantiated over I2C. In the loop, each of the 8 channels is selected and read. The selected channel and its temperature reading are printed to the serial console. If you don't have a DS18B20 sensor attached to a channel that is read, a negative value of -0.06 will be printed to the console. The print out below shows data on channels 0 and 4.



```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Reading channel 0
Temperature: 23.50 °C

Reading channel 1
Temperature: -0.06 °C

Reading channel 2
Temperature: -0.06 °C

Reading channel 3
Temperature: -0.06 °C

Reading channel 4
Temperature: 25.88 °C

Reading channel 5
Temperature: -0.06 °C

Reading channel 6
Temperature: -0.06 °C

Reading channel 7
Temperature: -0.06 °C
```

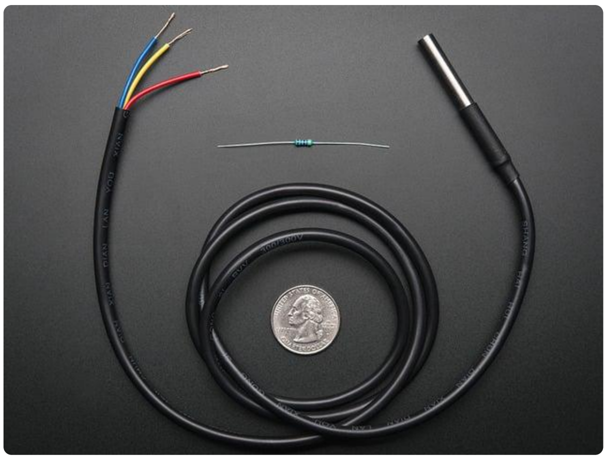
Python Docs

Python Docs (https://adafru.it/1a2U)

Arduino

Using the DS2482S-800 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller with two DS18B20 sensors, installing the Adafruit_DS248x (https://adafru.it/1a32) library, and running the provided example code.

DS18B20 Digital temperature sensor + extras
These 1-wire digital temperature sensors are fairly precise (±0.5°C over much of the range) and can give up to 12 bits of precision from the onboard digital-to-analog...
https://www.adafruit.com/product/374

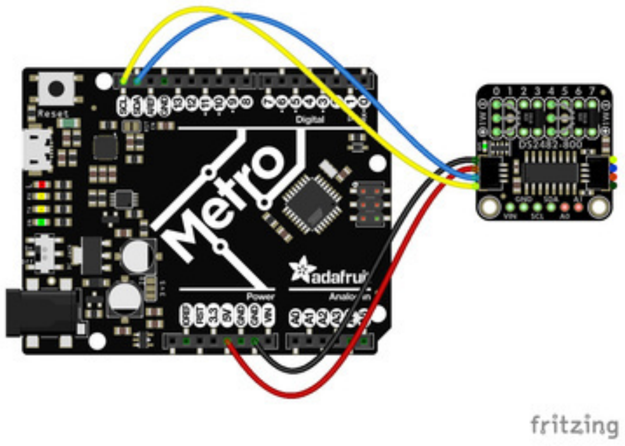


Waterproof 1-Wire DS18B20 Digital temperature sensor
This is a pre-wired and waterproofed (with heat shrink) version of a 1 Wire DS18B20 sensor. Handy for when you need to measure something far away, or in wet conditions. While the...
https://www.adafruit.com/product/381

Wiring

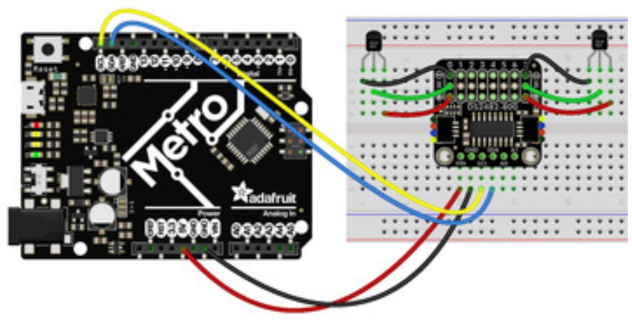
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the breakout VIN.

Here is an Adafruit Metro wired up to the breakout with two DS18B20 sensors using the STEMMA QT connector. The DS18B20 sensors are connected directly to the breakout on channel 0 and channel 4.



Breakout STEMMA VIN to **Metro 5V (red wire)**
Breakout STEMMA GND to **Metro GND (black wire)**
Breakout STEMMA SDA to **Metro SDA (blue wire)**
Breakout STEMMA SCL to **Metro SCL (yellow wire)**
Breakout channel 0 - to **DS18B20 GND**
Breakout channel 0 1W to **DS18B20 signal**
Breakout channel 0 + to **DS18B20 VIN**
Breakout channel 4 - to **DS18B20 GND**
Breakout channel 4 1W to **DS18B20 signal**
Breakout channel 4 + to **DS18B20 VIN**

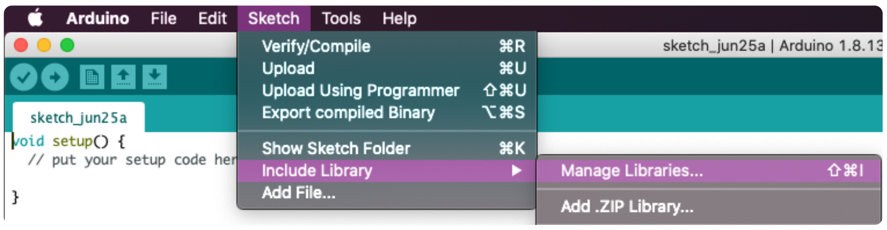
Here is an Adafruit Metro wired up using a solderless breadboard:



Breakout VIN to **Metro 5V (red wire)**
Breakout GND to **Metro GND (black wire)**
Breakout SDA to **Metro SDA (blue wire)**
Breakout SCL to **Metro SCL (yellow wire)**
Breakout channel 0 - to **DS18B20 GND (black wire)**
Breakout channel 0 1W to **DS18B20 signal (green wire)**
Breakout channel 0 + to **DS18B20 VIN (red wire)**
Breakout channel 4 - to **DS18B20 GND (black wire)**
Breakout channel 4 1W to **DS18B20 signal (green wire)**
Breakout channel 4 + to **DS18B20 VIN (red wire)**

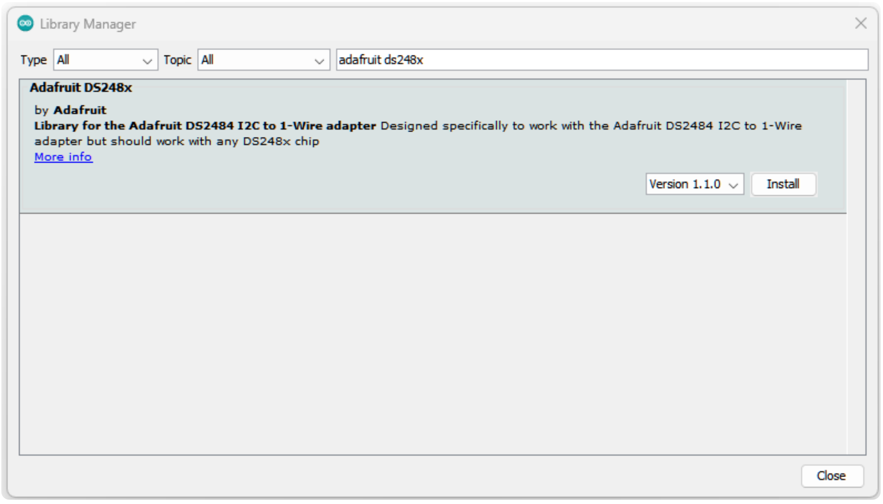
Library Installation

You can install the **Adafruit_DS248x** library for Arduino using the Library Manager in the Arduino IDE.



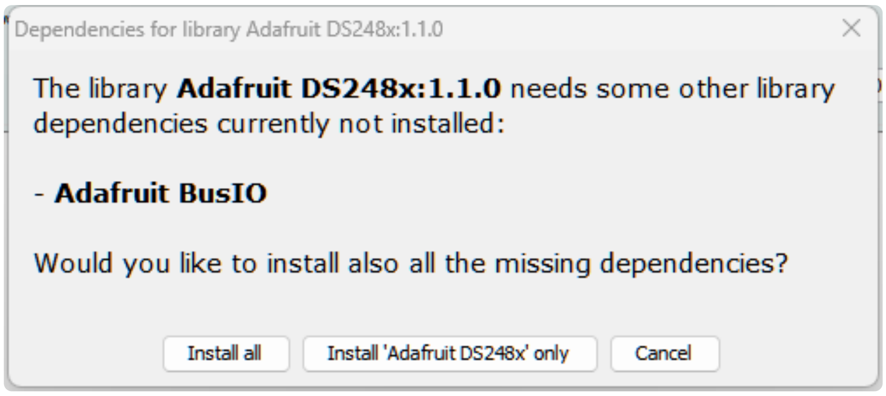
```
void setup() {
  // put your setup code her
}
```

Click the **Manage Libraries ...** menu item, search for **Adafruit_DS248x**, and select the **Adafruit DS248x** library:



Adafruit DS248x
by **Adafruit**
Library for the Adafruit DS2484 I2C to 1-Wire adapter Designed specifically to work with the Adafruit DS2484 I2C to 1-Wire adapter but should work with any DS248x chip
More info

If asked about dependencies, click "Install all".


Dependencies for library Adafruit DS248x:1.1.0

The library **Adafruit DS248x:1.1.0** needs some other library dependencies currently not installed:

- **Adafruit BusIO**

Would you like to install also all the missing dependencies?

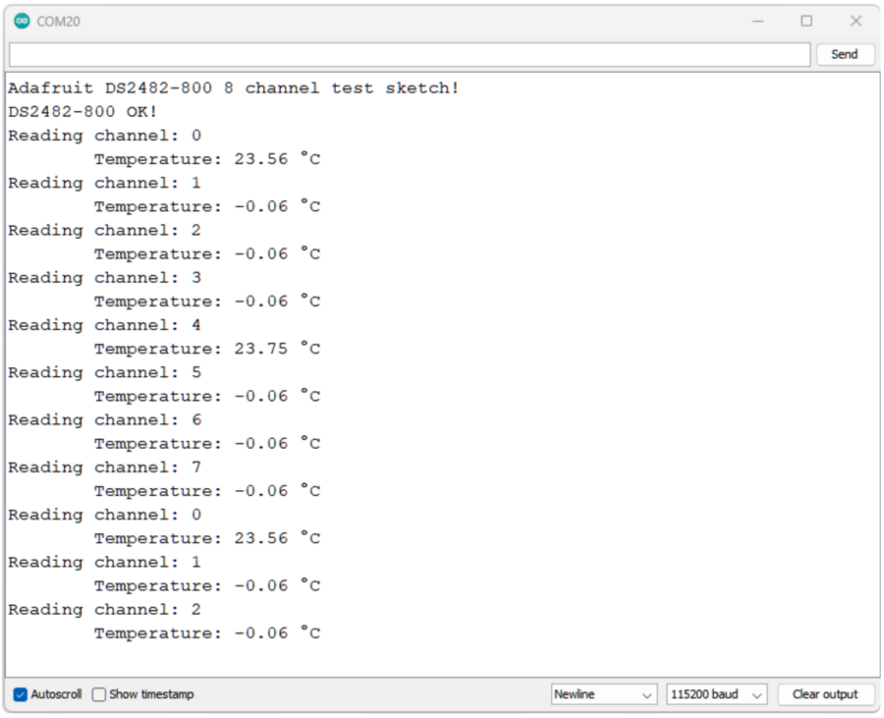
If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Example Code

```
#include "Adafruit_DS248x.h"

#define DS18B20_CMD_SKIP_ROM 0xCC
#define DS18B20_CMD_CONVERT_T 0x44
#define DS18B20_CMD_READ_SCRATCHPAD 0xBE

Adafruit_DS248x ds248x;

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);

  Serial.println("Adafruit DS2482-800 8 channel test sketch!");
  if (!ds248x.begin(&Wire, DS248X_ADDRESS)) {
    Serial.println(F("DS2482-800 initialization failed."));
    while (1);
  }
  Serial.println("DS2482-800 OK!");
}


void loop() {

  for (int i = 0; i < 8; i++) {
    ds248x.selectChannel(i);
    Serial.print("Reading channel: ");
    Serial.println(i);
    float temperature = readTemperature(i);
    Serial.print("\tTemperature: ");
    Serial.print(temperature);
    Serial.println(" °C");
    delay(1000);
  }

}
```

```
float readTemperature(uint8_t channel) {
    // Select the channel on the DS2482-800
    if (!ds248x.selectChannel(channel)) {
        // Handle error if channel selection fails
        Serial.println("Failed to select channel");
        return NAN; // Return 'Not a Number' to indicate an error
    }

    // Start temperature conversion
    ds248x.OneWireReset();
    ds248x.OneWireWriteByte(DS18B20_CMD_SKIP_ROM); // Skip ROM command
    ds248x.OneWireWriteByte(DS18B20_CMD_CONVERT_T); // Convert T command
    delay(750); // Wait for conversion (750ms for maximum precision)

    // Read scratchpad
    ds248x.OneWireReset();
    ds248x.OneWireWriteByte(DS18B20_CMD_SKIP_ROM); // Skip ROM command
    ds248x.OneWireWriteByte(DS18B20_CMD_READ_SCRATCHPAD); // Read Scratchpad command

    uint8_t data[9];
    for (int i = 0; i < 9; i++) {
        ds248x.OneWireReadByte(&data[i]);
    }

    // Calculate temperature
    int16_t raw = (data[1] << 8) | data[0];
    float celsius = (float)raw / 16.0;

    return celsius;
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the DS2482S-800 recognized over I2C. In the loop, each of the 8 channels are selected and read one by one. Each channel's temperature data is printed to the Serial Monitor. If you do not have a sensor attached to a channel, then you'll see negative data print out for it. In the print out below, you'll see data from sensors on channel 0 and channel 4.



```
Adafruit DS2482-800 8 channel test sketch!
DS2482-800 OK!
Reading channel: 0
        Temperature: 23.56 °C
Reading channel: 1
        Temperature: -0.06 °C
Reading channel: 2
        Temperature: -0.06 °C
Reading channel: 3
        Temperature: -0.06 °C
Reading channel: 4
        Temperature: 23.75 °C
Reading channel: 5
        Temperature: -0.06 °C
Reading channel: 6
        Temperature: -0.06 °C
Reading channel: 7
        Temperature: -0.06 °C
Reading channel: 0
        Temperature: 23.56 °C
Reading channel: 1
        Temperature: -0.06 °C
Reading channel: 2
        Temperature: -0.06 °C
```

Arduino Docs

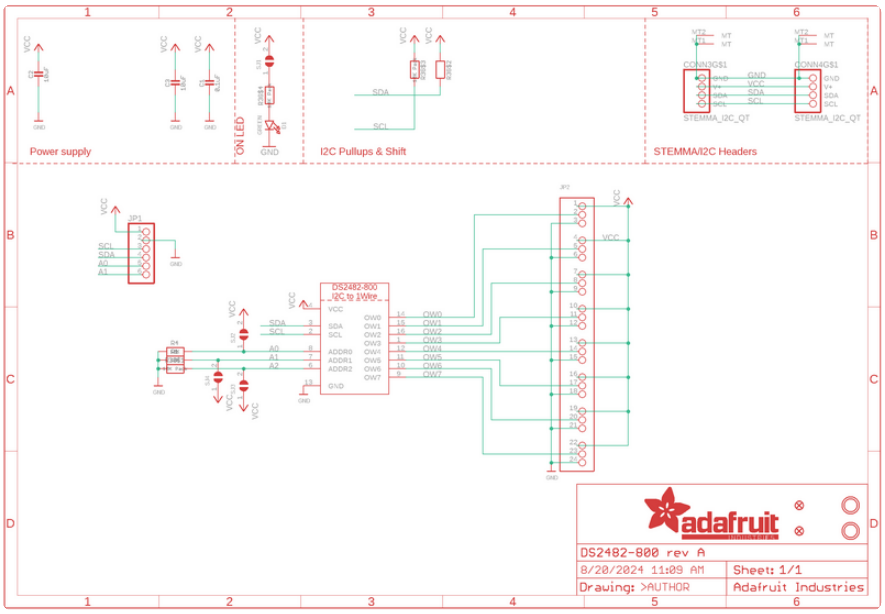
Arduino Docs (https://adafru.it/1a2M)

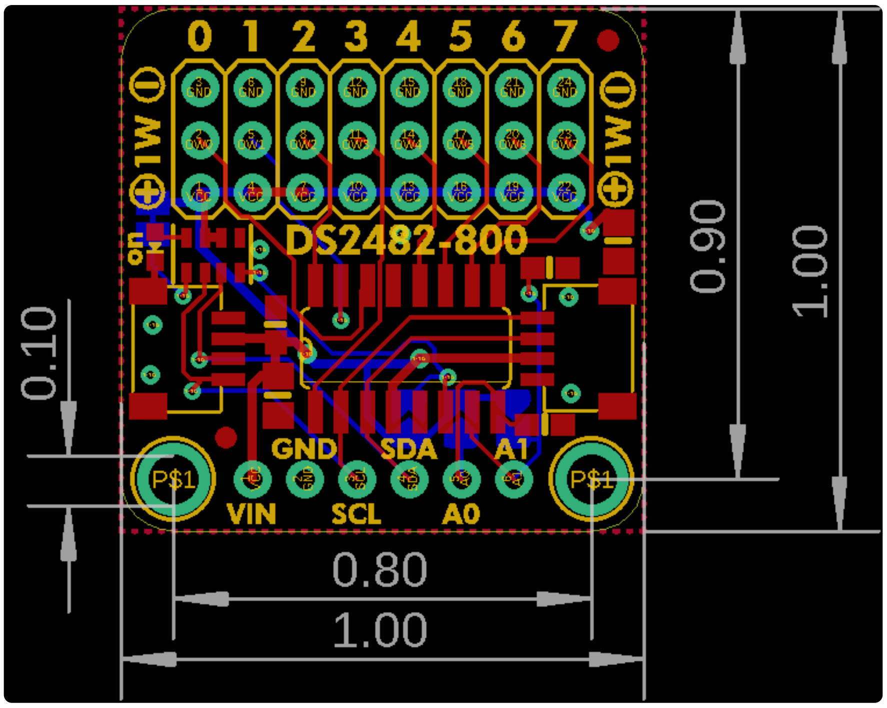
Downloads

Files

- DS2482S-800 Datasheet (https://adafru.it/1a7k)
- EagleCAD PCB Files on GitHub (https://adafru.it/1a7l)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1a7m)

Schematic and Fab Print




0 1 2 3 4 5 6 7
1W
1W
DS2482-800
on
GND
SDA
A1
VIN
SCL
A0
P$1
P$1
0.10
0.90
1.00
0.80
1.00