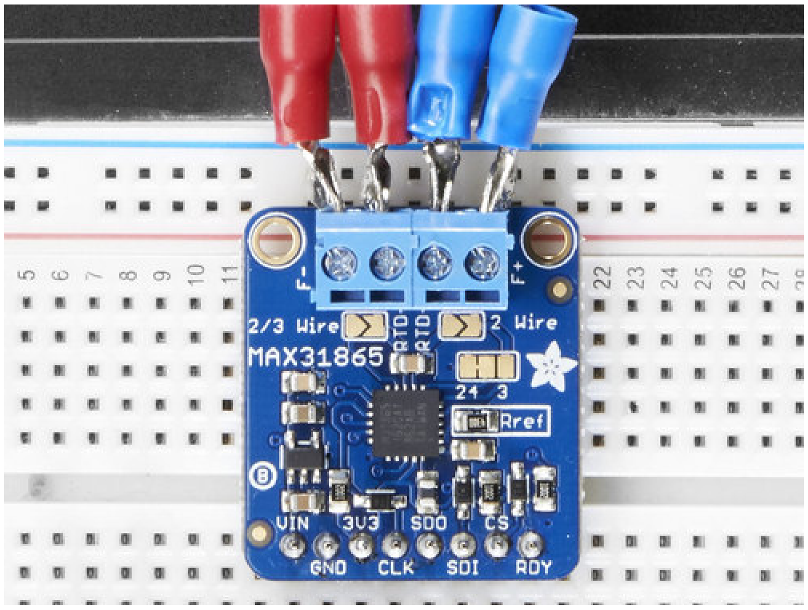
adafruit learning system

Adafruit MAX31865 RTD PT100 or PT1000 Amplifier

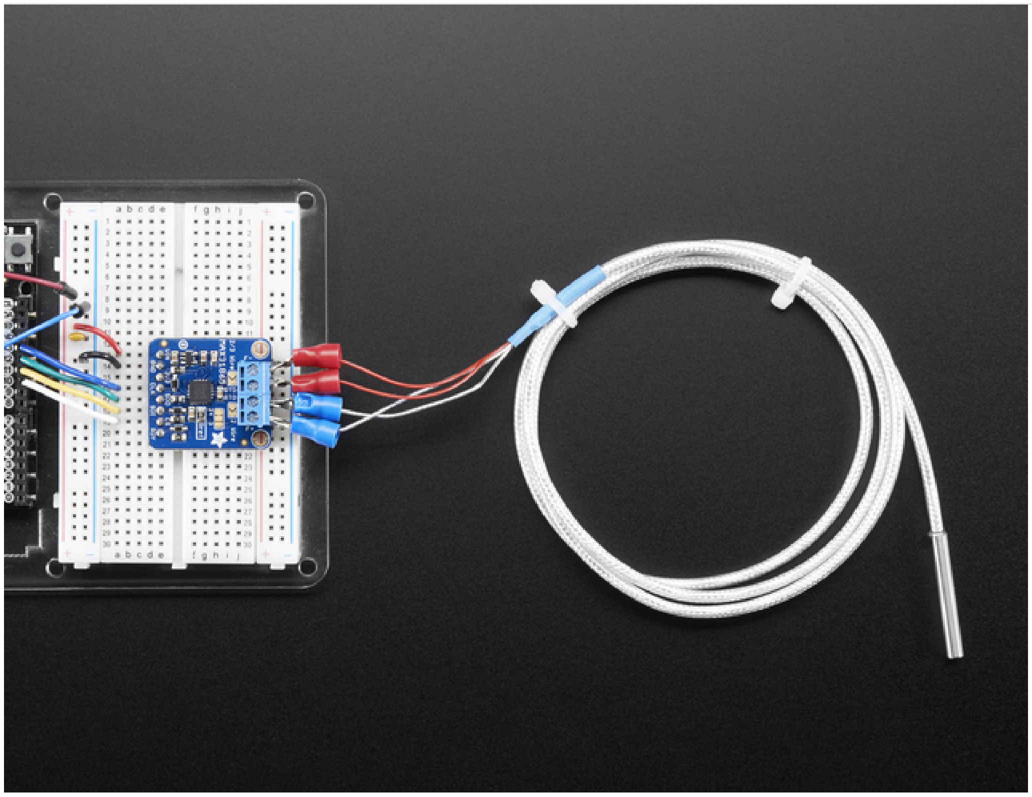
Created by lady ada



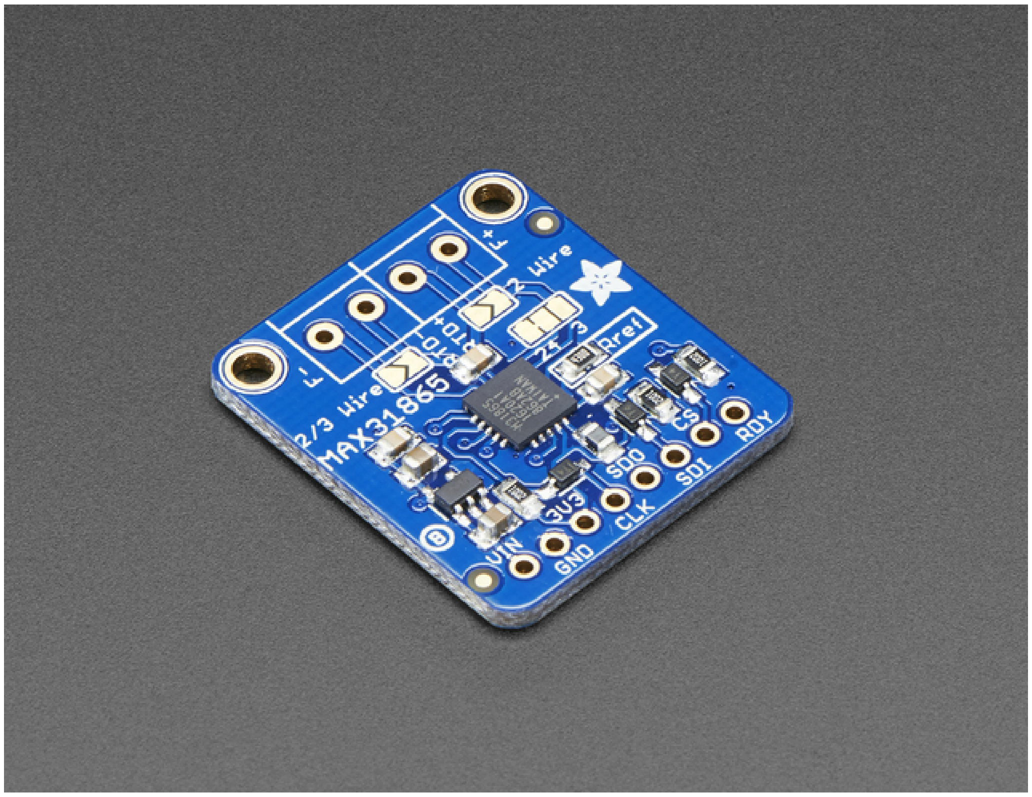
Last updated on 2018-03-05 10:47:36 PM UTC

Guide Contents

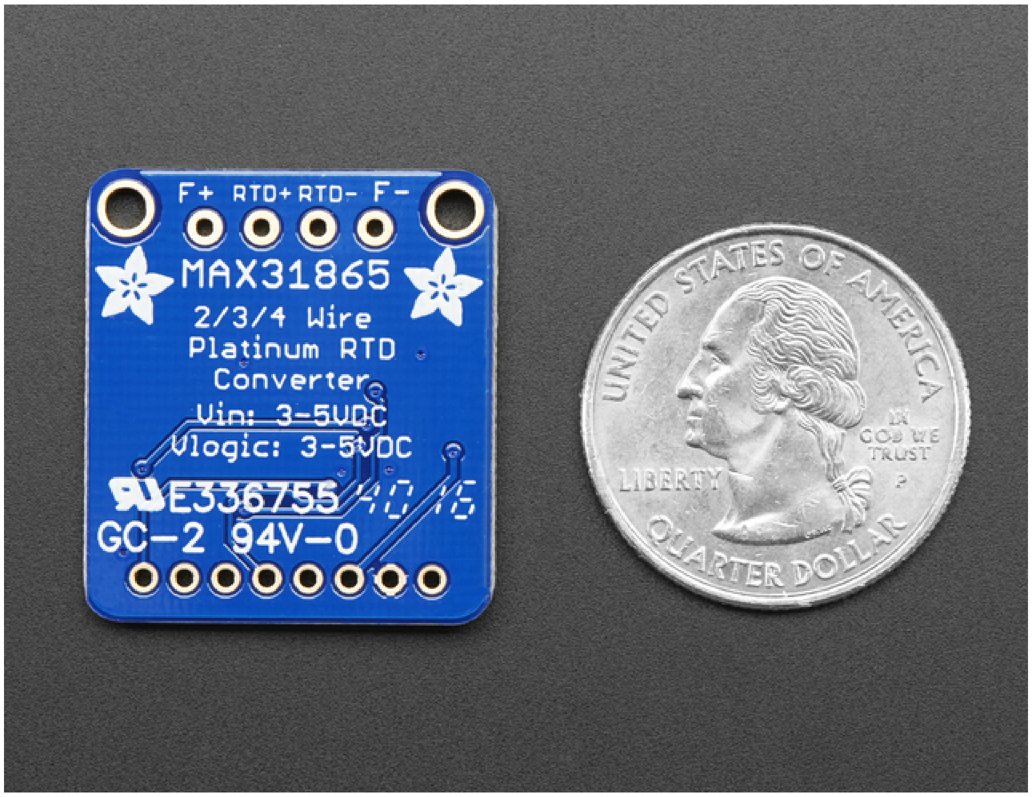
Overview



For precision temperature sensing, nothing beats a Platinum RTD. Resistance temperature detectors (RTDs) are temperature sensors that contain a resistor that changes resistance value as its temperature changes, basically a kind of thermistor. In this sensor, the resistor is actually a small strip of Platinum with a resistance of 100 or 1000 ohms at 0°C, thus the name PT100/PT1000. Compared to most NTC/PTC thermistors, the PT type of RTD is much more stable and precise (but also more expensive) PT's have been used for many years to measure temperature in laboratory and industrial processes, and have developed a reputation for accuracy (better than thermocouples), repeatability, and stability.



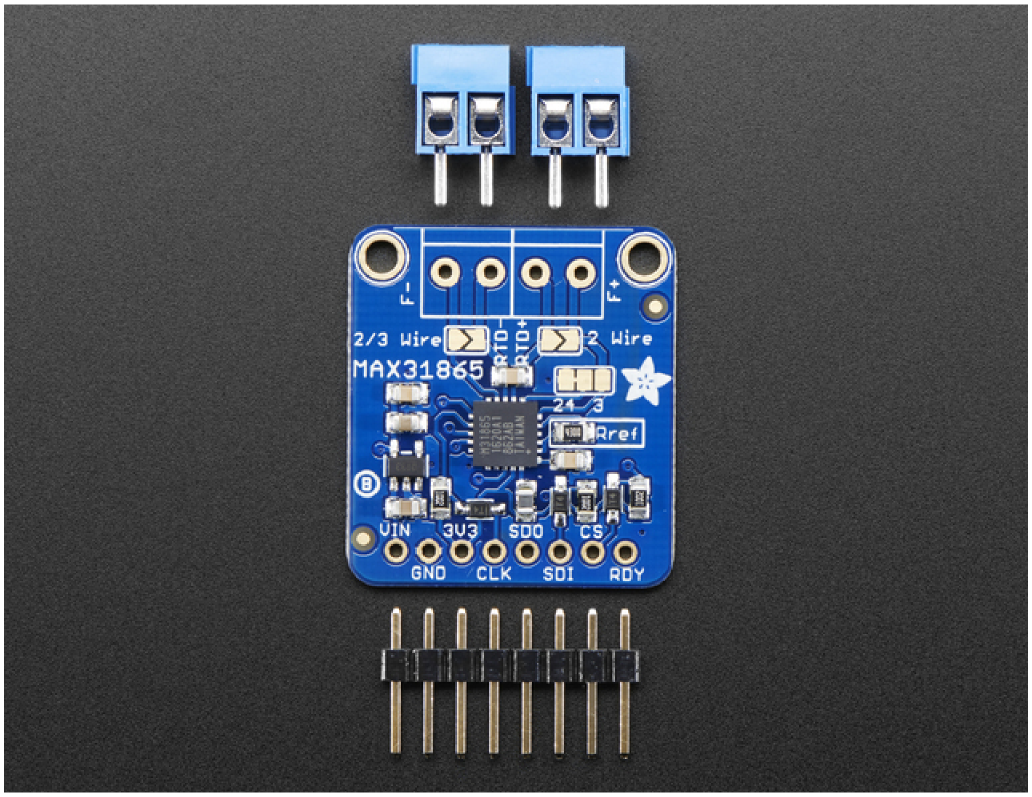
However, to get that precision and accuracy out of your PT100x RTD you must use an amplifier that is designed to read the low resistance. Better yet, have an amplifier that can automatically adjust and compensate for the resistance of the connecting wires. If you're looking for a great RTD sensor, today is your lucky day because we have a lovely Adafruit RTD Sensor Amplifier with the MAX31865 sensor.



We've carried various MAXIM thermocouple amplifiers and they're great - but thermocouples don't have the best accuracy or precision, for when the readings must be as good as can be. The MAX31865 handles all of your RTD needs, and can even compensate 3 or 4 wire RTDs for better accuracy. Connect to it with any microcontroller over SPI and read out the resistance ratio from the internal ADC. We put a 0.1% resistor as a reference resistor on the breakout. We have some example code that will calcuate the temperature based on the resistance for you.

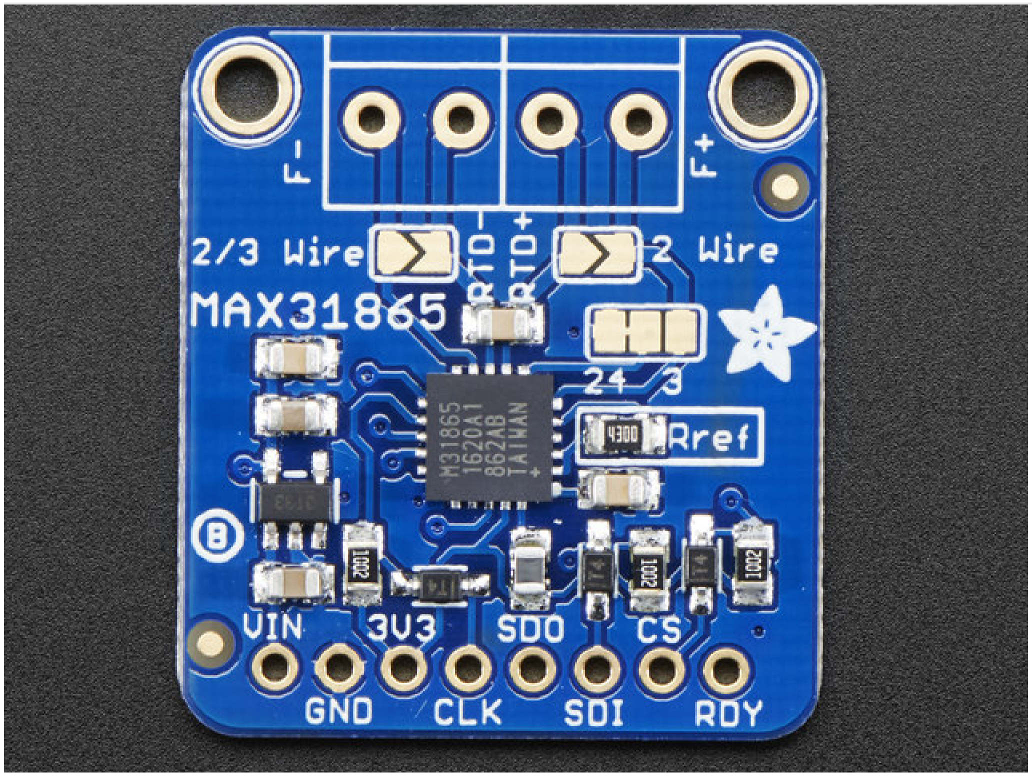
- The PT100 version of the breakout uses **430Ω**
- The PT1000 version uses **4300Ω**

We even made the breakout 5V compliant, with a 3.3V regulator and level shifting, so you can use it with any Arduino or microcontroller



Each order comes with one assembled RTD amplifier breakout board. Also comes with two 2-pin terminal blocks (for connecting to the RTD sensor) and pin header (to plug into any breadboard or perfboard). **A required PT100 or PT1000 RTD is not included!** (But we stock them in the shop). Some soldering is required to solder the headers and terminal blocks to the breakout, but it's an easy task with soldering tools.

Pinouts



The MAX31865 is a tiny surface mount chip, and it needs a lot of other parts to make it work, so we've got it on a nice breakout board for you. You can control the chip and read data from it using the breakouts at the bottom. Let's go thru these!

Power Pins:

- **Vin** - this is the power pin. Since the 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 micro 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

SPI Logic pins:

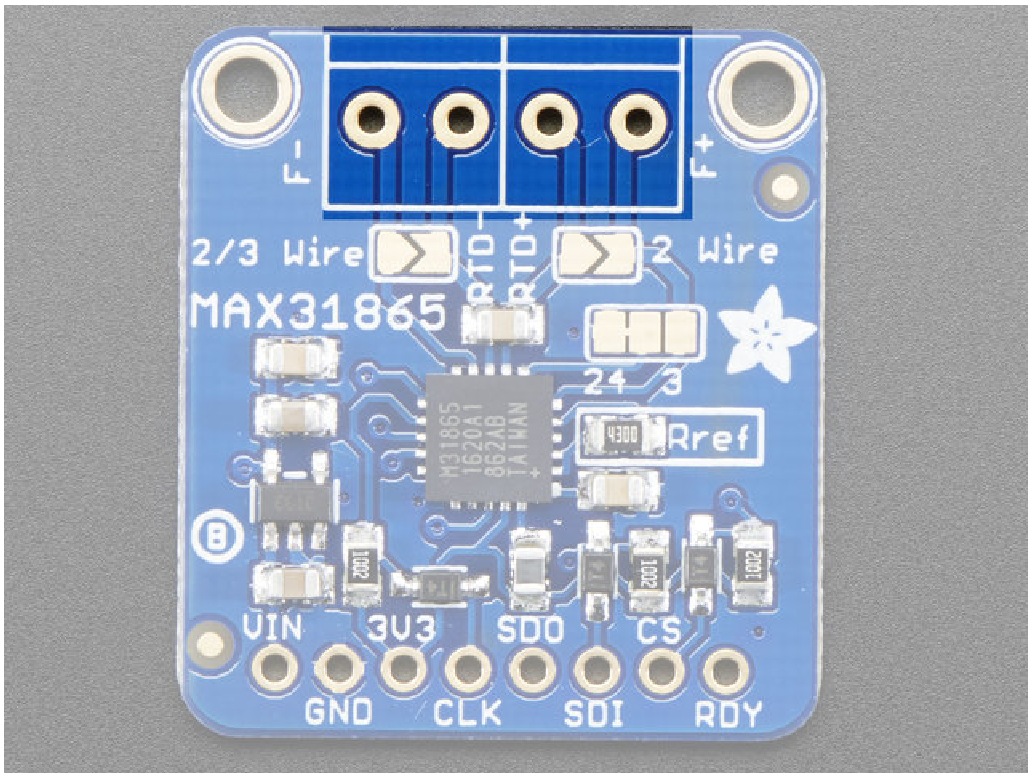
All pins going into the breakout have level shifting circuitry to make them 3-5V logic level safe. Use whatever logic level is on **Vin!**

- **SCK** - This is the **S**PI **C**loc**k** pin, its an input to the chip
- **SDO** - this is the **S**erial **D**ata **O**ut / **M**aster **I**n **S**lave **O**ut pin, for data sent from the MAX31865 to your processor
- **SDI** - this is the **S**erial **D**ata **I**n / **M**aster **O**ut **S**lave **I**n pin, for data sent from your processor to the MAX31865
- **CS** - this is the **C**hip **S**elect pin, drop it low to start an SPI transaction. Its an input to the chip

If you want to connect multiple MAX31865's to one microcontroller, have them share the SDI, SDO and SCK pins. Then assign each one a unique CS pin.

- **RDY** (Ready) - is a data-ready indicator pin, you can use this pin to speed up your reads if you are writing your own driver. Our Arduino driver doesn't use it to save a pin

Sensor Terminal Blocks

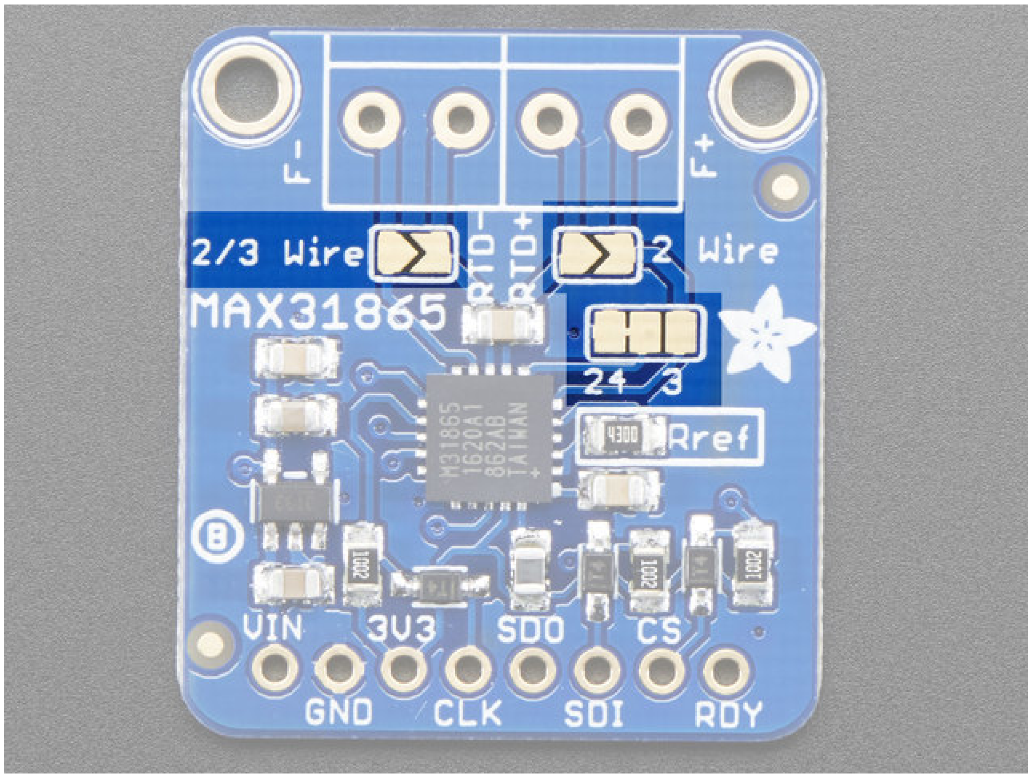


If you have an RTD sensor, you need to connect it somehow! the terminal block area is where you can clamp down to the sensor wires.

There are *four* contacts, but you can use 2, 3 or 4 wire sensors. You may need to solder or jumper some pads deending on how many wires you want to use. You can also use a 3 or 4 wire sensor as a 3-wire or 2-wire sensor (just dont connect the extra wires).

Check the RTD wiring page for details on how to connect the sensor you've got!

Configuration Jumpers



By default the sensor is wired up for 4-wire RTD usage but can be set up for 2 or 3 wire very easily.

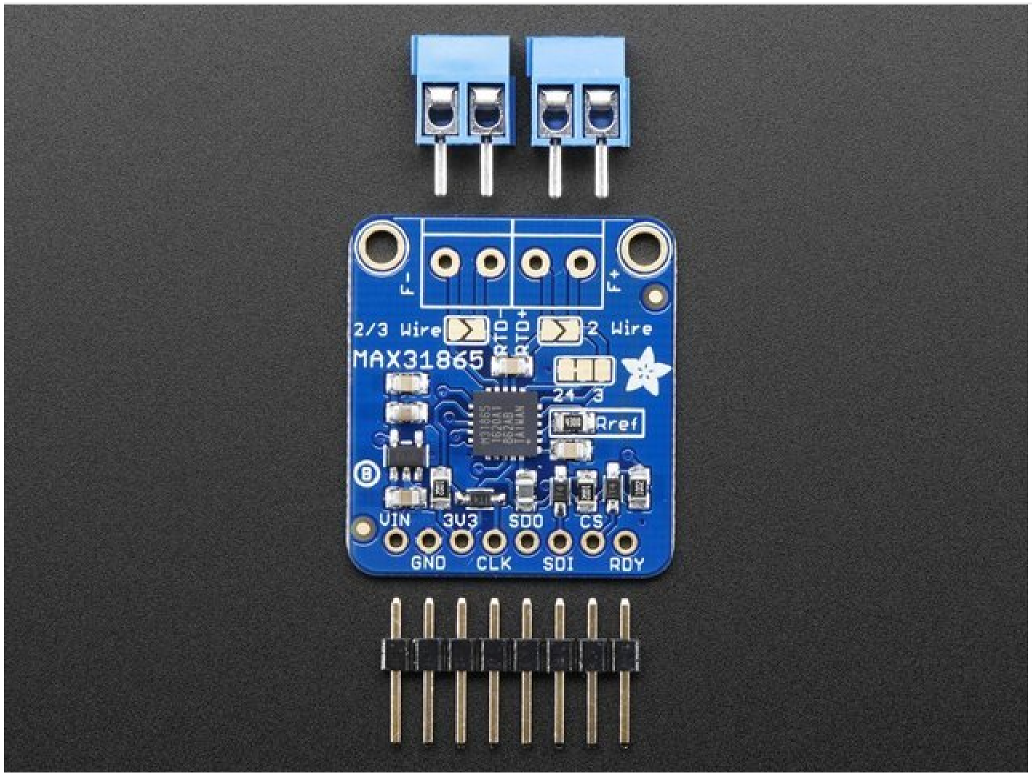
For 4-wire usage, do nothing with the jumpers!

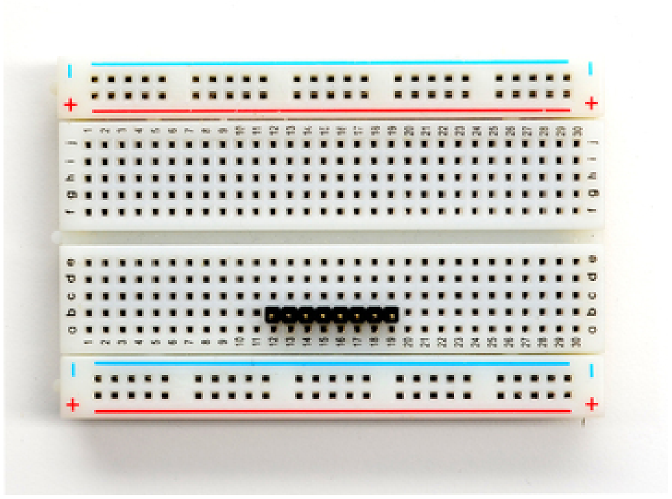
For 3-wire usage. Solder closed the jumper labeled **2/3 Wire** and cut the wire connecting the left side of the 2-way jumper right above Rref. Then solder closed the right side labeled **3**

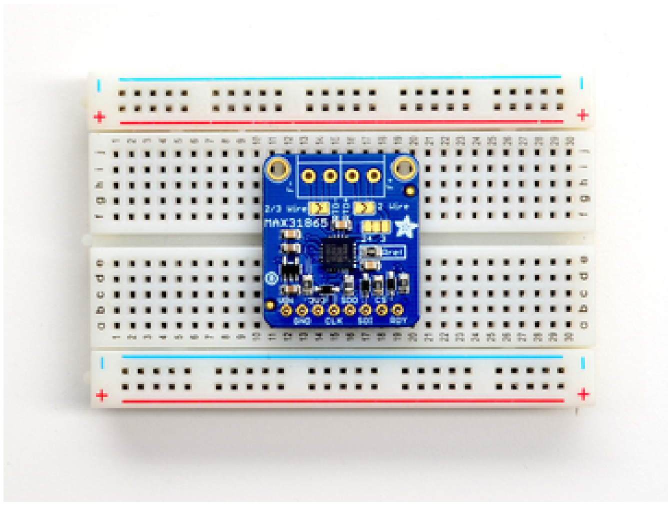
For 2-wire usage, solder closed the two triangular jumpers below the terminal blocks (or put short wire jumpers between the two terminal blocks on either side (essentially jumpering the two right side terminal holes together, and same for left side)

Check the RTD wiring page for details on how to connect the sensor you've got!

Assembly


F-
F+
2/3 Wire
RTD-
RTD+
2 Wire
MAX31865
24
3
Rref
VIN
3V3
SDO
CS
GND
CLK
SDI
RDY





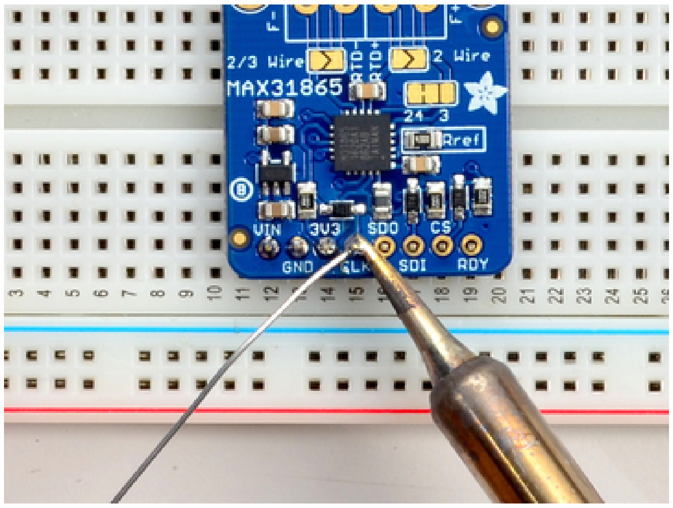
Prepare the header strip:

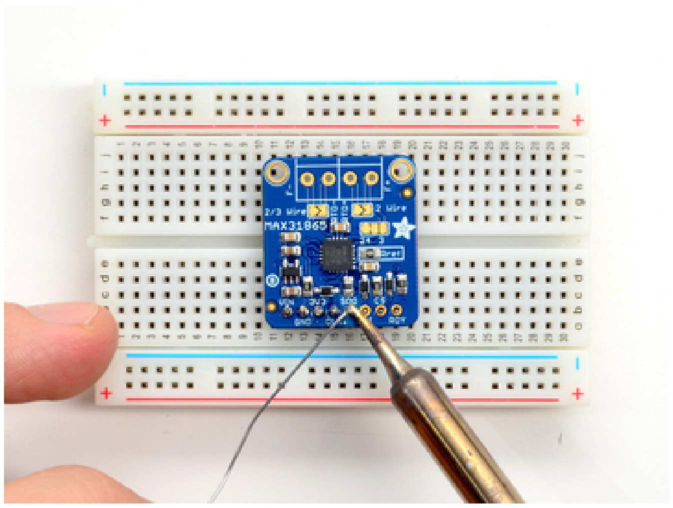
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

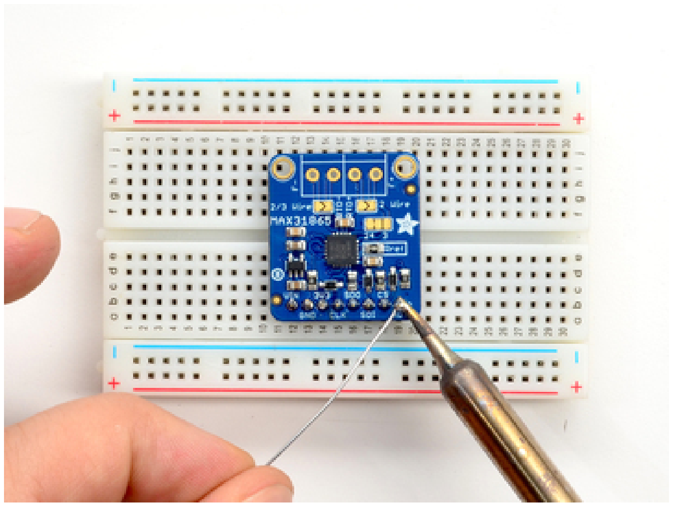
Solder!

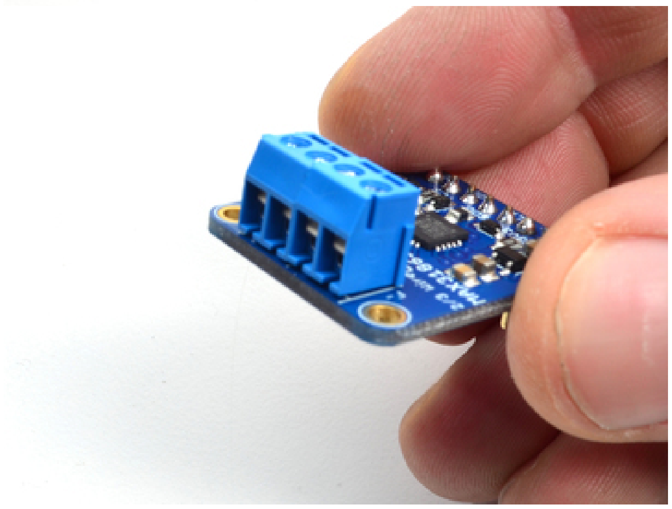
Be sure to solder all pins for reliable electrical contact.

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


2/3 Wire
2 Wire
MAX31865
RTD-
RTD+
Rref
VIN
3V3
SDO
CS
GND
CLK
SDI
RDY



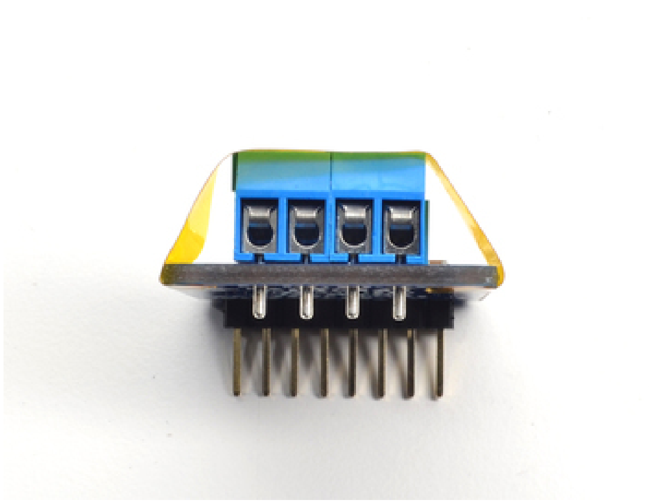


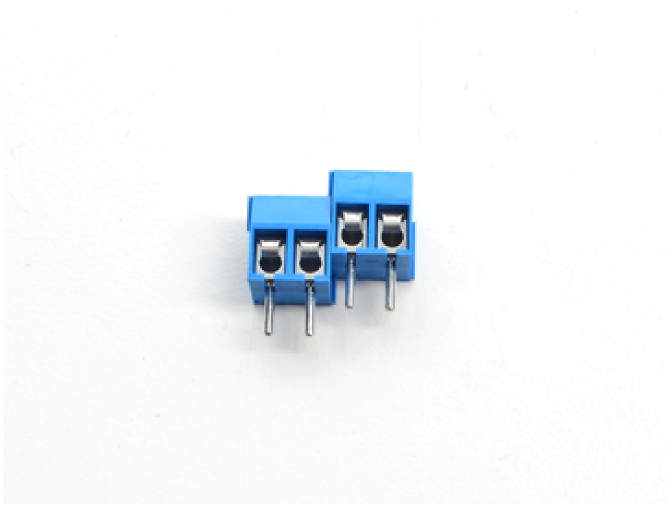


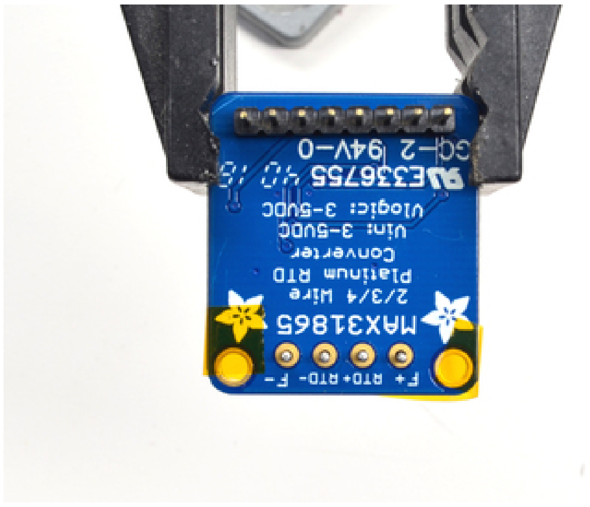
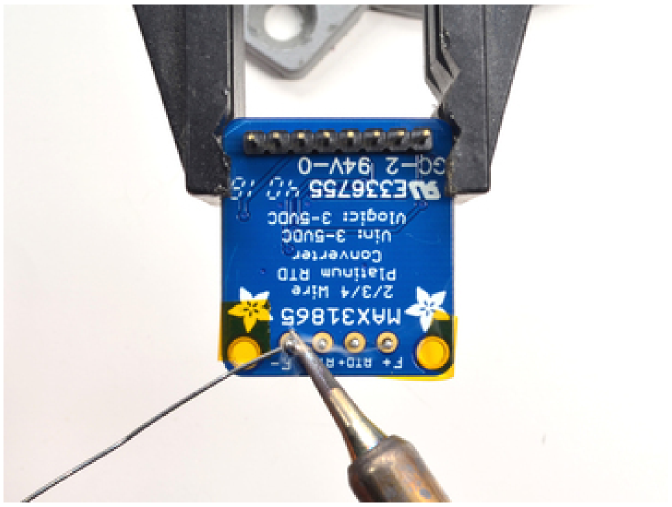
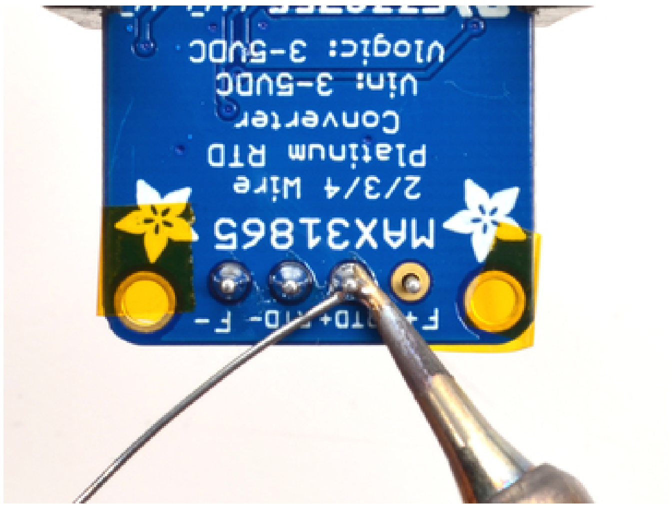
Next we will solder in the two 3.5mm terminal blocks used to connect power & the motor to the breakout board.

Make sure the open parts of the terminals face outwards so you can easily connect wires

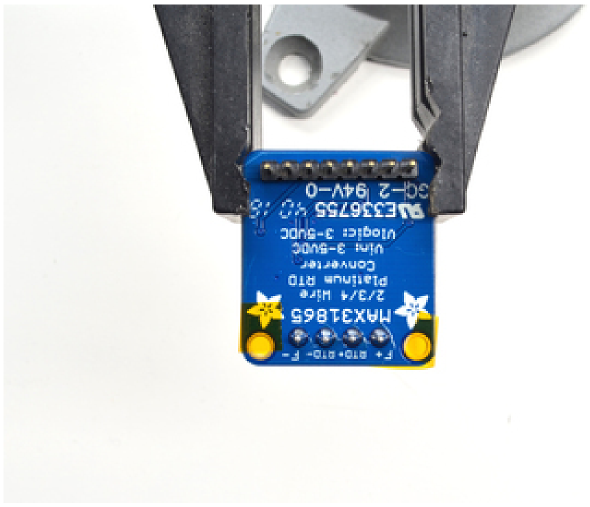
To make it easier to keep these in place, you can use some tape to hold down the two header pieces. Tacky clay also works, whatever you've got handy!

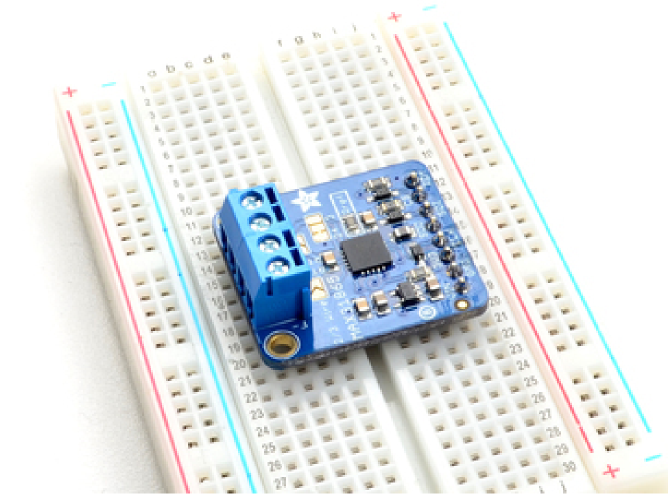


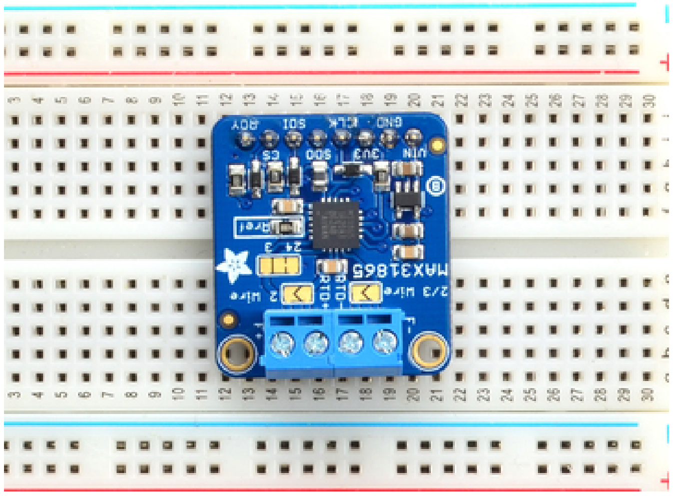


Solder in both pins of each terminal block. You can remove the tape when done.







OK You're done!

RTD Wiring & Config

RTDs are really very simple devices: just a small strip of Platinum that measures 100Ω or 1000Ω exactly at 0°C. Bonded to the PT100/PT1000 are 2, 3 or 4 wires.

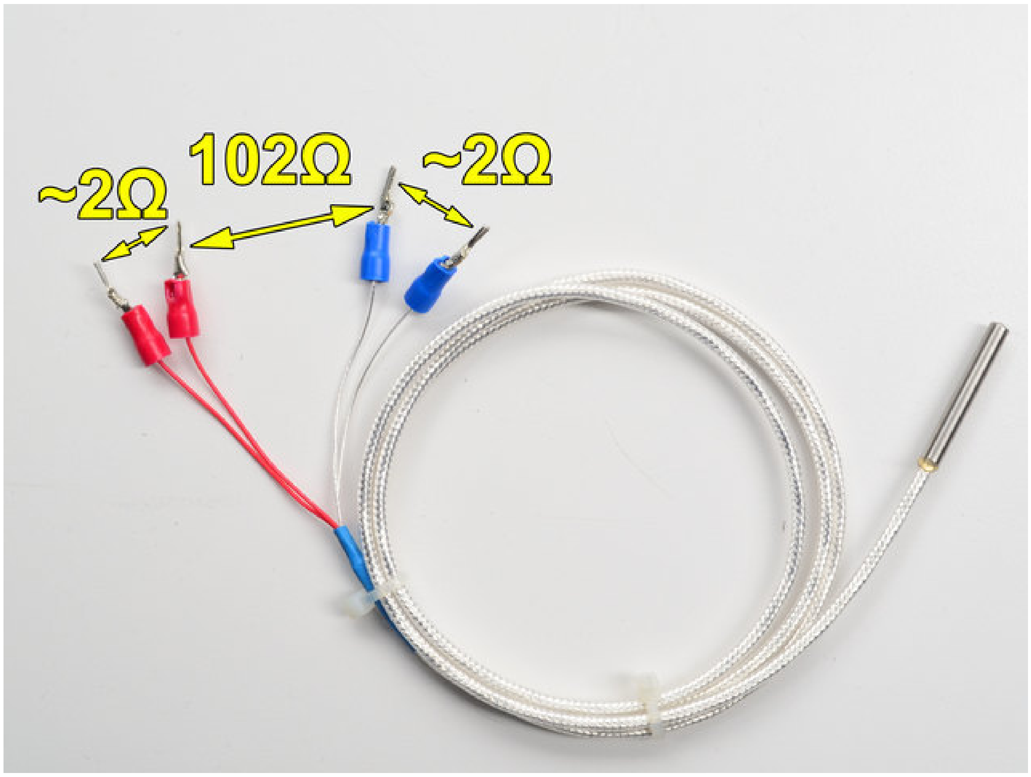
4-Wire RTDs

We'll explain the 4-wire version since that's the most complex. Normally if you want to measure a resistor you just connect your multimeter to each side of the resistor. The multimeter puts a small current through the resistor and measures the voltage generated across it (remember V = I * R). This works great for just about all resistors. However, for *very precise readings* of low-resistance resistors, you *also* have to account for the wires connected! For basic resistors, they are only good to 5% anyways so we don't mind the resistance of the wires.

For RTDs, the wires, especially the 1 meter long ones, are 1, 2 maybe even 4Ω of extra resistance! That can add up to half or even a full °C! No good, we want to make sure that resistance is not included in our measurement

Thus, the 4-wire RTD. Each side of the RTD has *two* wires attached. Each wire is maybe 1Ω of resistance. When connected to the amplifier, the smart amp will measure the voltage across the RTD and also across the wire pairs.

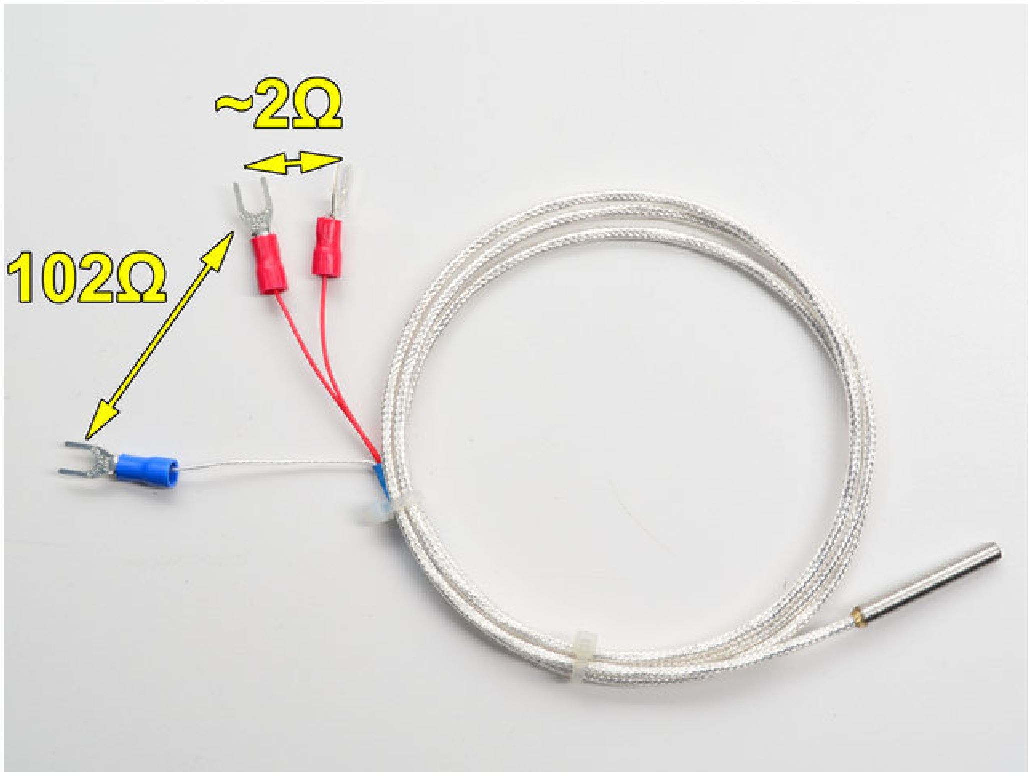
For example, here's the approximate resistances of a 4-Wire PT100 RTD at 0°C (for a PT1000, the middle resistance would be (1002Ω rather than 102Ω)



(Remember that the middle resistance - 102 or 1002 Ω - will vary with temperature, but the 2Ω wires will not) When the amp measures this sensor, it will measure the resistance between one set of red and blue wires. It will then measure the resistances between the red wires and blue wires. Then divide those resistances by half - since there's two wires and we just want the resistance of one wire. The final result is 102 - 1 - 1 = 100Ω

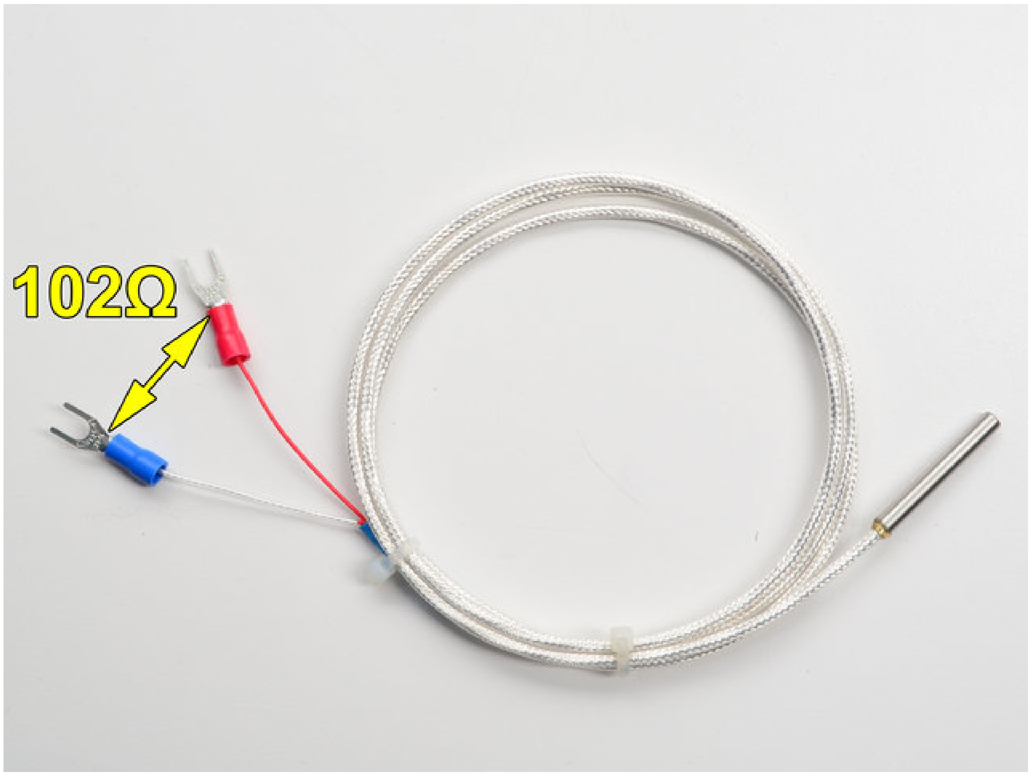
3-Wire RTDs

These are very similar to the 4-wire type but there is only one 'pair' of connected wires. The reasoning for this is that the wires for the RTD are all pretty much the same gauge and length, so rather than having two pairs, the amplifier will just read one pair and use that resistance as the same for both wires.



2-Wire RTDs

These are as simple as it gets, only one wire per side. You may need to calibrate the sensor by putting it in an ice bath to get the resistance at 0°C (say 102Ω) and then subtracting 100Ω to figure out the collective resistance of the connection wires!

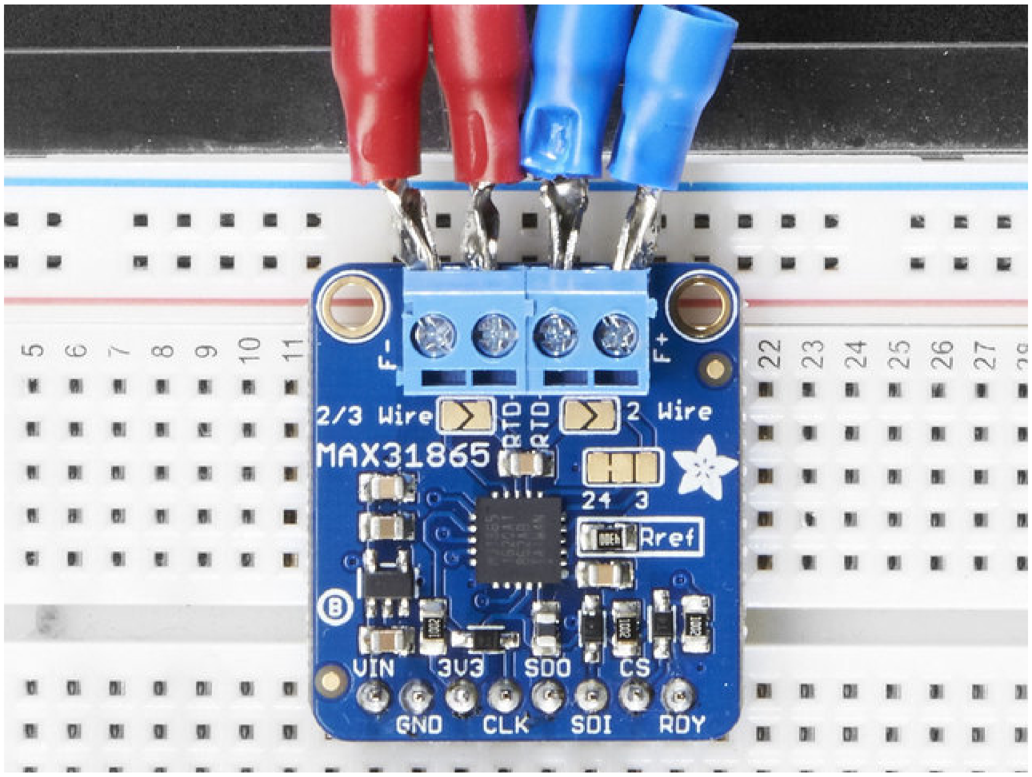


How To Wire Up!

4-Wire Sensors

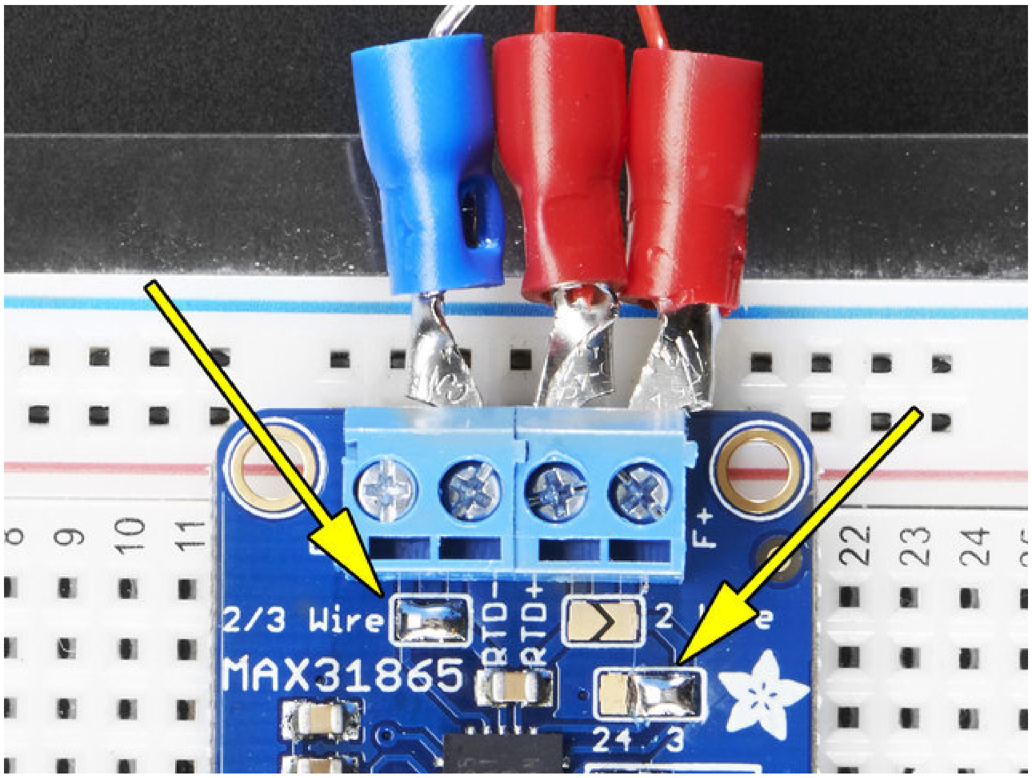
Connect the four wires to each of the pads. Use a multimeter to determine which wires connect together directly (2 ohms or so between them) and which connect through the RTD. Chances are the wires that connect together are the same color. The two pairs connect so that the ones that are connected together go into the two matching terminal blocks on left or right. It doesn't matter which of the matched pair is on the outside or inside. It doesn't matter which of the match pairs are on the left or right.

Do not solder closed any jumpers or cut any jumpers. Use as is!



3-Wire Sensors

Connect the three wires to the three right-most contacts. Use a multimeter to determine which wires connect together directly (2 ohms or so between them) and which connect through the RTD. Chances are the wires that connect together are the same color. The two wires that are connected together should go in the right-most blocks (labeled **F+** and **RTD+**). It doesn't matter which of the *matched* pair is on the outside or inside. The third wire that is on the other side of the RTD connects to the left (marked **F-** or **RTD-**). It doesn't matter which slot it's in!



You will have to cut the thin trace in between the 2-way jumper on the right side of the board, and then solder closed the blob on the right side.

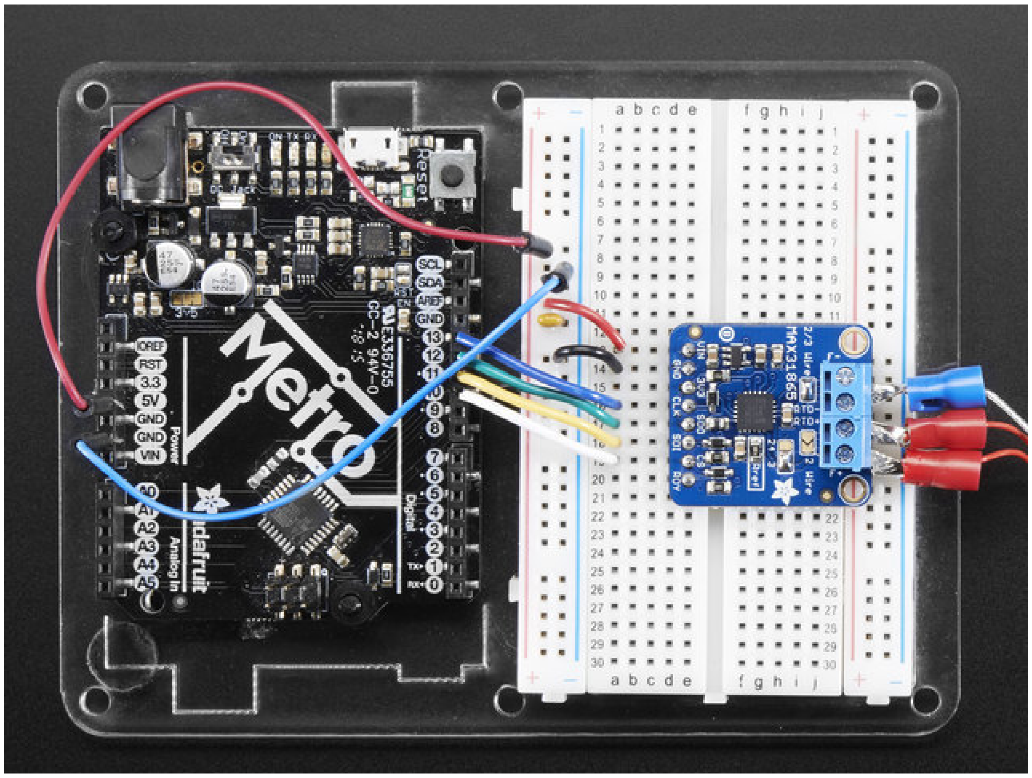
Then next to the terminal block on the left, solder closed that jumper as well. Alternatively you can put a piece of wire into the terminal blocks to 'short' them

2 Wire Sensor

This is the easiest wiring, you can just use either terminal block slot on the sides for each wire. Then either solder closed the jumpers next to the RTD terminal block or put little wires in the right and left terminal blocks to short them together.

Arduino Code

You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, as long as you have 4 available pins it is possible to 'bit-bang SPI' or you can use hardware SPI if you like. Just check out the library, then port the code.



SPI Wiring

Since this is a SPI-capable sensor, we can use hardware or 'software' SPI. To make wiring identical on all Arduinos, we'll begin with 'software' SPI. The following pins should be used:

- Connect **Vin** to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect **GND** to common power/data ground
- Connect the **SCK** pin to **Digital #13** but any pin can be used later
- Connect the **SDO** pin to **Digital #12** but any pin can be used later
- Connect the **SDI** pin to **Digital #11** but any pin can be used later
- Connect the **CS** pin **Digital #10** but any pin can be used later

Later on, once we get it working, we can adjust the library to use hardware SPI if you desire, or change the pins to other

Download Adafruit_MAX31865 library

To begin reading sensor data, you will need to download Adafruit_MAX31865 from our github repository. You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

Download latest Adafruit MAX31865 Arduino

Library

https://adafru.it/swB

Rename the uncompressed folder **Adafruit_MAX31865** and check that the **Adafruit_MAX31865** folder contains **Adafruit_MAX31865.cpp** and **Adafruit_MAX31865.h**

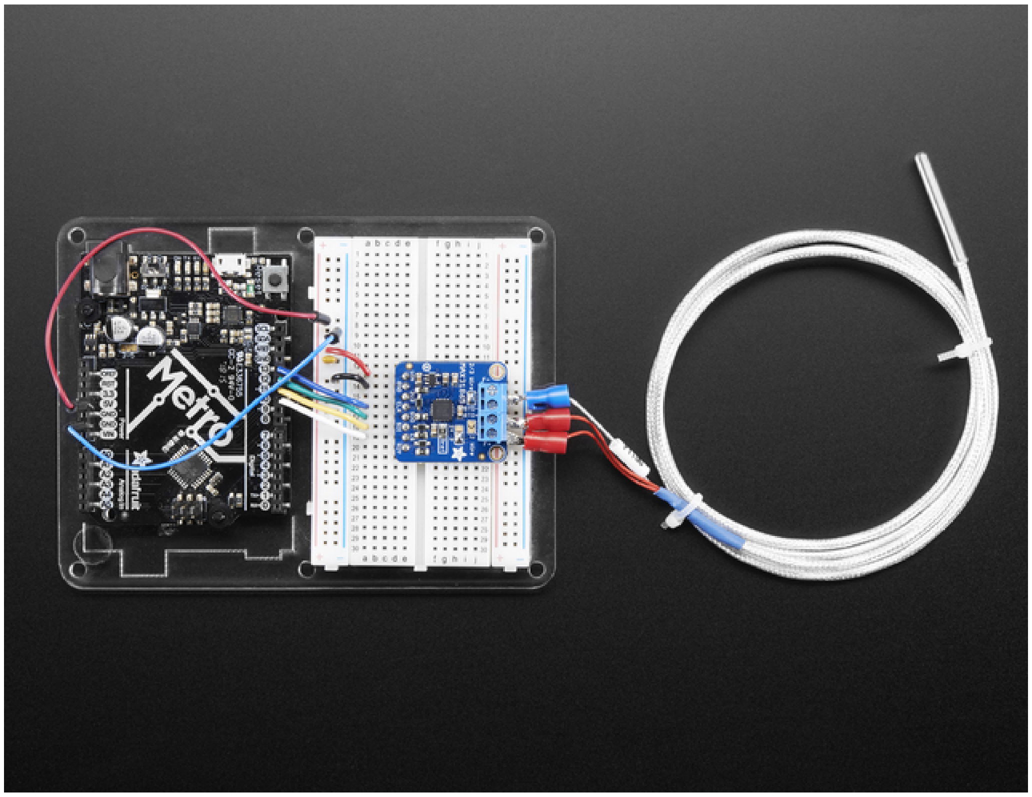
Place the **Adafruit_MAX31865** library folder your **arduinosketchfolder/libraries/** folder.
You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use

Restart the IDE

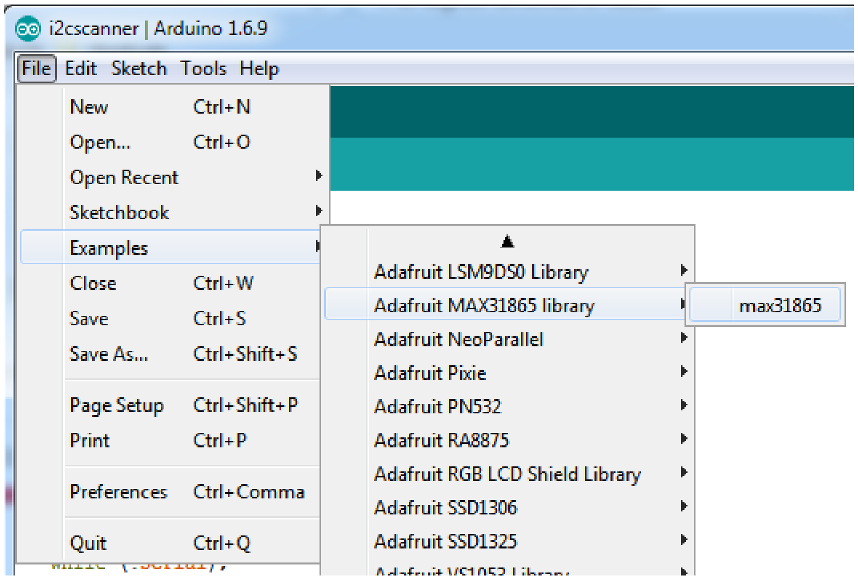
Attach PT100 or PT1000 RTD

You'll need to attach an RTD, for this demo we'll be using a 3-wire 1 meter long one but you can adjust the demo if you have a 2 or 4 wire. Check the RTD wiring page for the jumpers and wiring requirements!



Load Demo

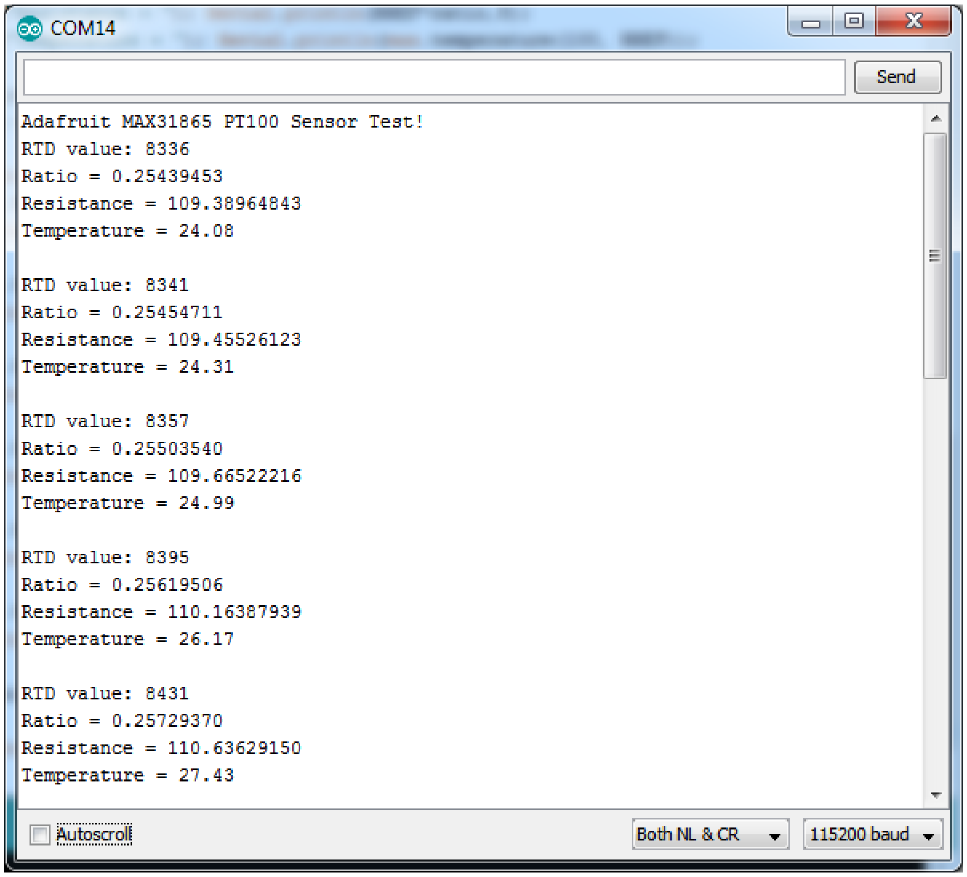
Open up **File->Examples->Adafruit_MAX31865->max31865** and upload to your Arduino wired up to the sensor. Adjust the `max.begin(MAX31865_3WIRE)` line if necessary.



Upload to your Arduino and open up the serial console at 115200 baud to see a print out of the sensors data. The MAX31865 doesn't actually return the resistance it measures. Instead it returns the *ratio* between the resistance measured and the **Rref** reference resistor.

- For the PT100 version of the breakout, this is a 430 ohm 0.1% resistor (marking is **4300** !!!)
- For the PT100 version of the breakout, this is a 4300 ohm 0.1% resistor (marking is **4301** !!!)

You can use that ratio to calculate the resistance and then determine the temperature



```
Adafruit MAX31865 PT100 Sensor Test!
RTD value: 8336
Ratio = 0.25439453
Resistance = 109.38964843
Temperature = 24.08

RTD value: 8341
Ratio = 0.25454711
Resistance = 109.45526123
Temperature = 24.31

RTD value: 8357
Ratio = 0.25503540
Resistance = 109.66522216
Temperature = 24.99

RTD value: 8395
Ratio = 0.25619506
Resistance = 110.16387939
Temperature = 26.17

RTD value: 8431
Ratio = 0.25729370
Resistance = 110.63629150
Temperature = 27.43
```

More Accuracy

Our library is efficient and small and uses an algorithm to calculate temperature. While this works very well, it isn't as accurate as it **could** be. Check out this ITS-90 conforming library from DrHaney that uses a lookup table for better accuracy!

Library Reference

You can start out by creating a MAX31865 object with either software SPI (where all four pins can be any I/O) using

```
// Use software SPI: CS, DI, DO, CLK
Adafruit_MAX31865 max = Adafruit_MAX31865(10, 11, 12, 13);
```

Or you can use hardware SPI. With hardware SPI you *must* use the hardware SPI pins for your Arduino - and each arduino type has different pins! Check the SPI reference to see what pins to use.
In this case, you can use any CS pin, but the other three pins are fixed

```
// use hardware SPI, just pass in the CS pin
Adafruit_MAX31865 max = Adafruit_MAX31865(10);
```

Once started, you can initialize the sensor with one of the following, depending on what kind of RTD you've got connected!

```
max.begin(MAX31865_2WIRE)
max.begin(MAX31865_3WIRE)
max.begin(MAX31865_4WIRE)
```

Reading Resistance

If you want to know the actual *resistance* (not temperature) you can do that fairly easily. You can read *ratio* from the MAX31865 with

```
max.readRTD()
```

This will give you the raw 16-bit unsigned value where 0xFFFF is '1.0 ratio'. Chances are you want to convert it to the resistance. We recommend this code:

```
  Serial.print("RTD value: "); Serial.println(rtd);
  float ratio = rtd;
  ratio /= 32768;
  Serial.print("Ratio = "); Serial.println(ratio,8);
  Serial.print("Resistance = "); Serial.println(RREF*ratio,8);
```

You'll need to define RREF - in this case its **430.0 for PT100** and **4300.0 for PT1000**

Calculating Temperature

Once you have the resistance you can look up in an RTD table or use a calculation to do a best-fit approximation. We use the example from this app note: http://www.analog.com/media/en/technical-documentation/applicationnotes/AN709_0.pdf

It's fast and seems to work very well! We have a one-stop function that does everything for you, just call:

```
max.temperature(100, RREF)
```

Where the first argument is the resistance of the RTD at 0°C (for PT100 that's 100) and the second argument is the value of the reference resistor. This function returns the tempreature in °C

Faults

The MAX31865 has a wide-ranging fault mechanism that can alert you via pin or function when something is amiss. Don't forget to test this functionality before relying on it!

You can read faults with

```
max.readFault()
```

Which will return a uint8_t type with bits set for each of 6 different fault types. You can test for each one with this set of code:

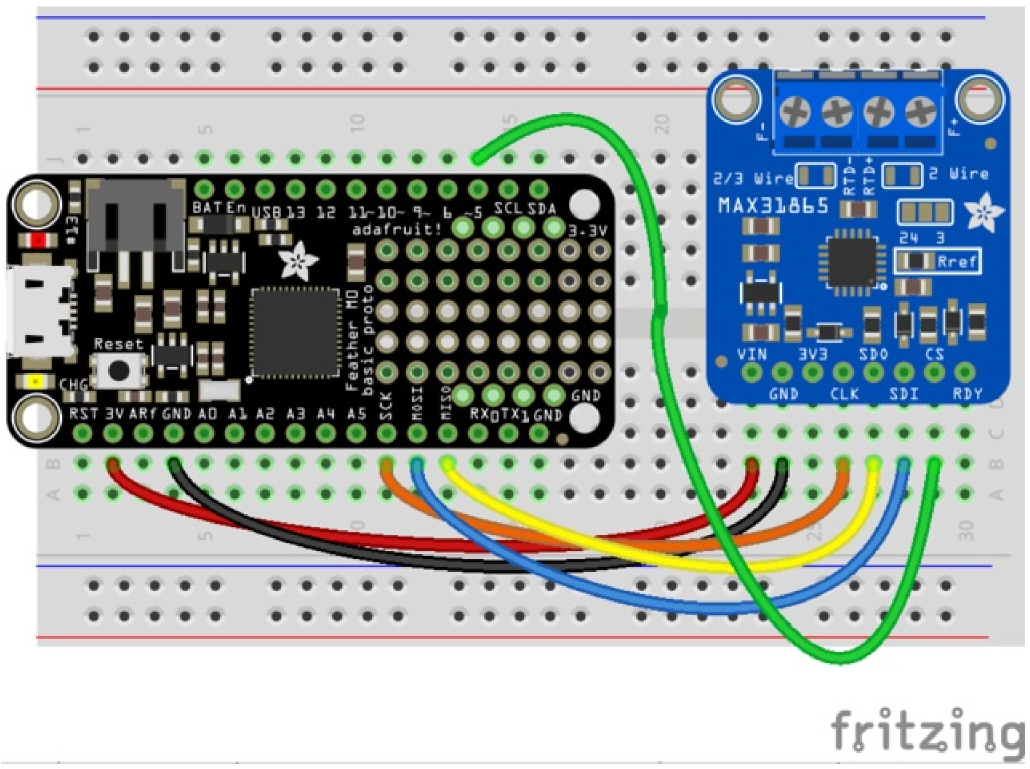
```
  // Check and print any faults
  uint8_t fault = max.readFault();
  if (fault) {
    Serial.print("Fault 0x"); Serial.println(fault, HEX);
    if (fault & MAX31865_FAULT_HIGHTHRESH) {
      Serial.println("RTD High Threshold");
    }
    if (fault & MAX31865_FAULT_LOWTHRESH) {
      Serial.println("RTD Low Threshold");
    }
    if (fault & MAX31865_FAULT_REFINLOW) {
      Serial.println("REFIN- > 0.85 x Bias");
    }
    if (fault & MAX31865_FAULT_REFINHIGH) {
      Serial.println("REFIN- < 0.85 x Bias - FORCE- open");
    }
    if (fault & MAX31865_FAULT_RTDINLOW) {
      Serial.println("RTDIN- < 0.85 x Bias - FORCE- open");
    }
    if (fault & MAX31865_FAULT_OVUV) {
      Serial.println("Under/Over voltage");
    }
    max.clearFault();
  }
```

In particular, the last four are ones that indicate a hardware failure. The first two are threshold faults, we don't have code for setting those thresholds at this time.

CircuitPython Code

It's easy to use the MAX31865 sensor with CircuitPython and the Adafruit CircuitPython MAX31865 module. This module allows you to easily write Python code that reads the range from the sensor.

First wire up a MAX31865 to your board exactly as shown on the previous pages for Arduino.   Here's an example of wiring a Feather M0 to the sensor with a SPI connection:



- **Board 3V** to **sensor VIN**
- **Board GND** to **sensor GND**
- **Board SCK** to **sensor CLK**
- **Board MOSI** to **sensor SDI**
- **Board MISO** to **sensor SDO**
- **Board D5** to **sensor CS** (or any other digital I/O pin)

Next you'll need to install the Adafruit CircuitPython MAX31865 library on your CircuitPython board. **Remember this module is for Adafruit CircuitPython firmware and not MicroPython.org firmware!**

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. For example the Circuit Playground Express guide has a great page on how to install the library bundle for both express and non-express boards.

Remember for non-express boards like the Trinket M0, Gemma M0, and Feather/Metro M0 basic you'll need to manually install the necessary libraries from the bundle:

- **adafruit_max31865.mpy**
- **adafruit_bus_device**

You can also download the **adafruit_max31865.mpy** from its releases page on Github.

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_max31865.mpy,** and **adafruit_bus_device** files and folders copied over.

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

Usage

To demonstrate the usage of the sensor we'll initialize it and read the range and more from the board's Python REPL.

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

```
import board
import busio
import digitalio
import adafruit_max31865
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
cs = digitalio.DigitalInOut(board.D5)  # Chip select of the MAX31865 board.
sensor = adafruit_max31865.MAX31865(spi, cs)
```

Notice you need to explicitly define the chip select digital I/O pin--be sure to use the same pin as your wiring (D5 if following this example exactly).

By default the MAX31865 class assumes a **2 wire** sensor, however you can change this by setting the **wires** keyword argument in the initializer. Set this to the number of wires in your sensor (2, 3, or 4). For example to create a 3 wire sensor:

```
import board
import busio
import digitalio
import adafruit_max31865
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
cs = digitalio.DigitalInOut(board.D5)  # Chip select of the MAX31865 board.
sensor = adafruit_max31865.MAX31865(spi, cs, wires=3)
```

Be sure to set **wires** to the appropriate value for your sensor!

In addition you can specify the nominal resistance and reference resistance of the RTD with these optional keyword arguments of the initializer:

- **rtd_nominal** - This is the resistance value in Ohms of the RTD at a nominal value (typically 0 degrees Celsius). This defaults to 100 for a PT100 sensor. **For a PT1000 change it to 1000 Ohms.**
- **ref_resistor** - The reference resistor value in Ohms. The default is 430 Ohms which matches the PT100 version of the breakout. **For a PT1000 breakout change this to 4300 Ohms.**

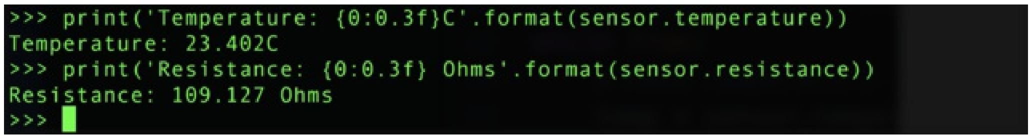
For example here's how to create a 2-wire PT1000 sensor instance with 1000 Ohm nominal resistance and 4300 Ohm reference resistance:

```
import board
import busio
import digitalio
import adafruit_max31865
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
cs = digitalio.DigitalInOut(board.D5)  # Chip select of the MAX31865 board.
sensor = adafruit_max31865.MAX31865(spi, cs, rtd_nominal=1000.0, ref_resistor=4300.0)
```

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

- **temperature** - The temperature measured by the sensor in degrees Celsius.
- **resistance** - The resistance of the RTD in Ohms.

```
print('Temperature: {0:0.3f}C'.format(sensor.temperature))
print('Resistance: {0:0.3f} Ohms'.format(sensor.resistance))
```



```
>>> print('Temperature: {0:0.3f}C'.format(sensor.temperature))
Temperature: 23.402C
>>> print('Resistance: {0:0.3f} Ohms'.format(sensor.resistance))
Resistance: 109.127 Ohms
>>> 
```

See the simpletest.py example for a complete demo of printing the range every second. Save this as **main.py** on the board and examine the REPL output to see the temperature printed every second.

```
# Simple demo of the MAX31865 thermocouple amplifier.
# Will print the temperature every second.
import time

import board
import busio
import digitalio

import adafruit_max31865


# Initialize SPI bus and sensor.
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
cs = digitalio.DigitalInOut(board.D5)  # Chip select of the MAX31865 board.
sensor = adafruit_max31865.MAX31865(spi, cs)
# Note you can optionally provide the thermocouple RTD nominal, the reference
# resistance, and the number of wires for the sensor (2 the default, 3, or 4)
# with keyword args:
#sensor = adafruit_max31865.MAX31865(spi, cs, rtd_nominal=100, ref_resistor=430.0, wires=2)

# Main loop to print the temperature every second.
while True:
    # Read temperature.
    temp = sensor.temperature
    # Print the value.
    print('Temperature: {0:0.3f}C'.format(temp))
    # Delay for a second.
    time.sleep(1.0)
```

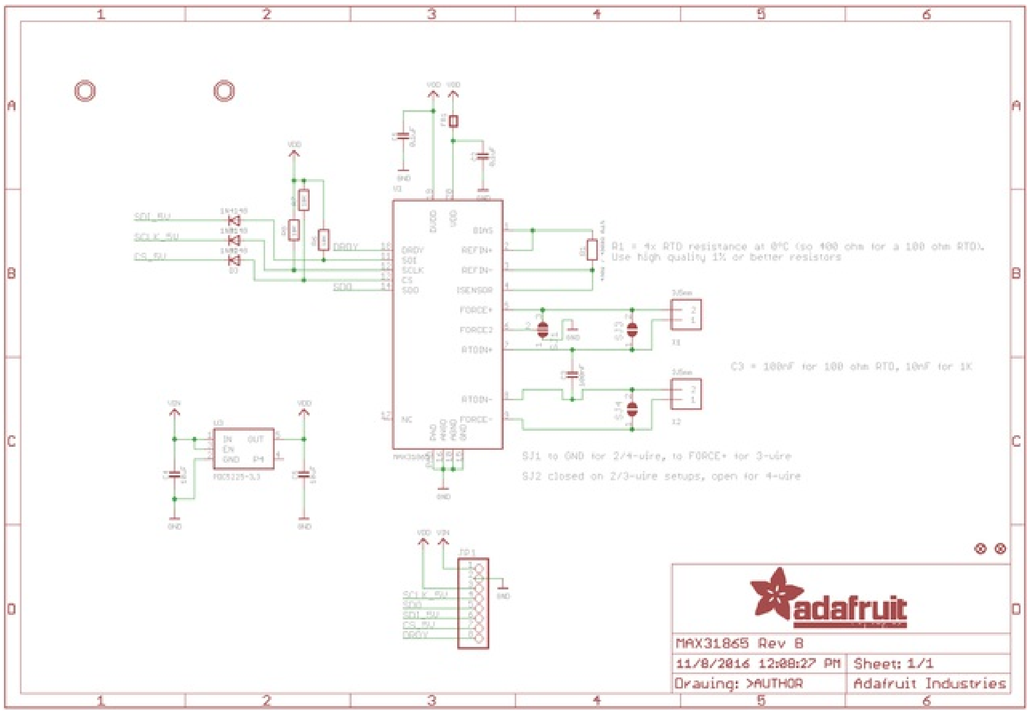
That's all there is to using the MAX31865 with CircuitPython!

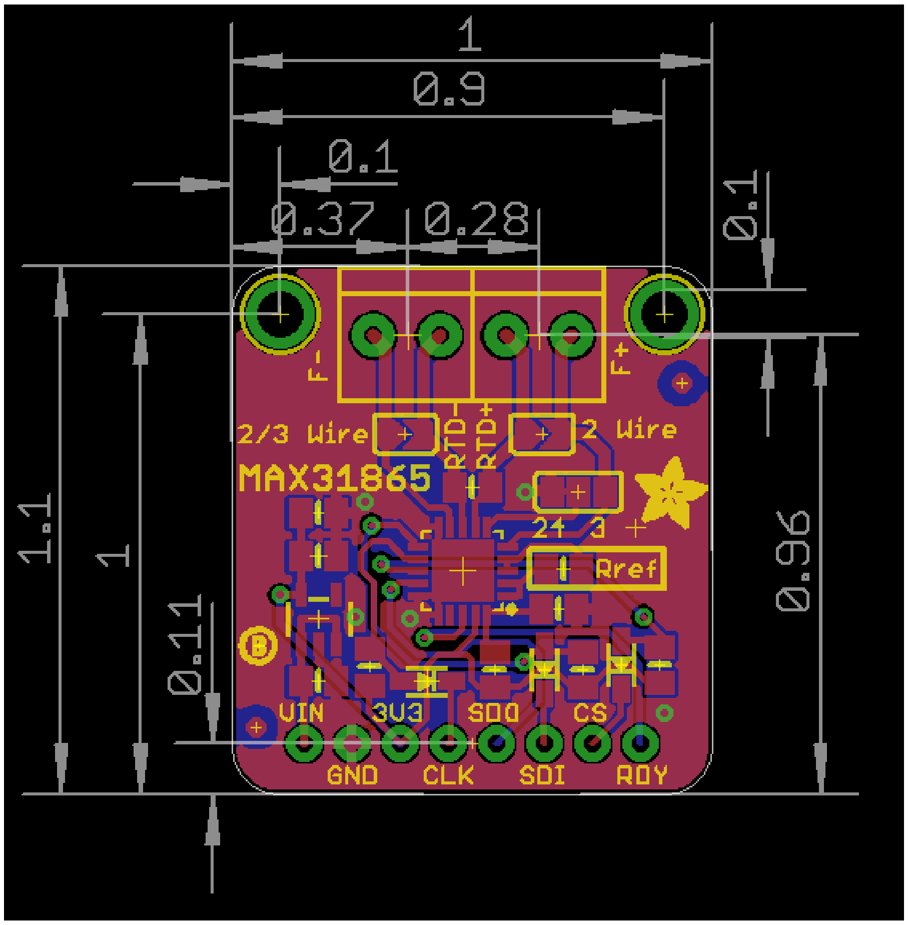
Downloads

Files

- Fritzing object in Adafruit Fritzing library
- EagleCAD PCB files on GitHub
- Library on GitHub
- MAX31865 Datasheet
- A "lookup table" based library is bigger but more precise since it doesn't do an approximation of the temperature - for advanced users!

Schematic & Fabrication Print


R1 = 4x RTD resistance at 0°C (so 400 ohm for a 100 ohm RTD).
Use high quality 1% or better resistors
C3 = 100nF for 100 ohm RTD, 10nF for 1K
SJ1 to GND for 2/4-wire, to FORCE+ for 3-wire
SJ2 closed on 2/3-wire setups, open for 4-wire
MAX31865 Rev B
11/8/2016 12:08:27 PM
Sheet: 1/1
Drawing: >AUTHOR
Adafruit Industries


1
0.9
0.1
0.37
0.28
0.1
1.1
1
0.11
0.96
F-
F+
2/3 Wire
2 Wire
RTD-
RTD+
MAX31865
24
3
Rref
VIN
3V3
SDO
CS
GND
CLK
SDI
RDY