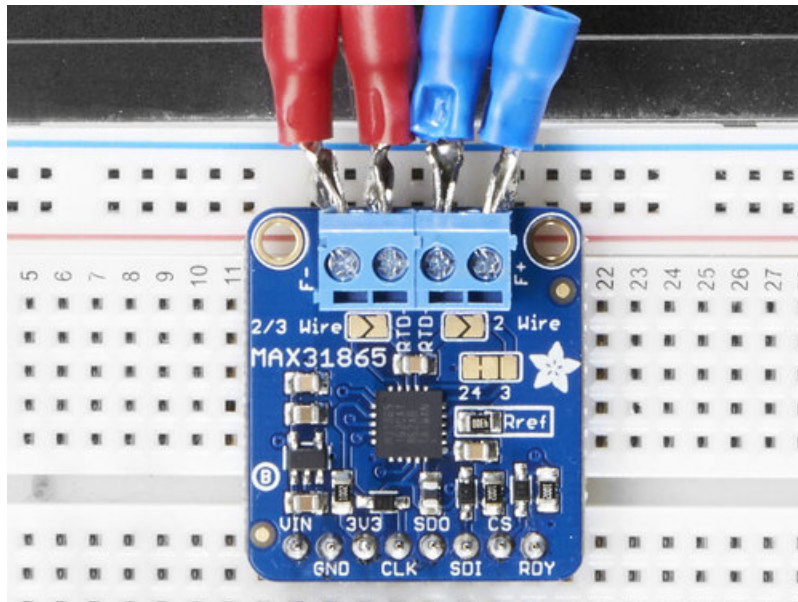


Adafruit MAX31865 RTD PT100 or PT1000 Amplifier

Created by lady ada

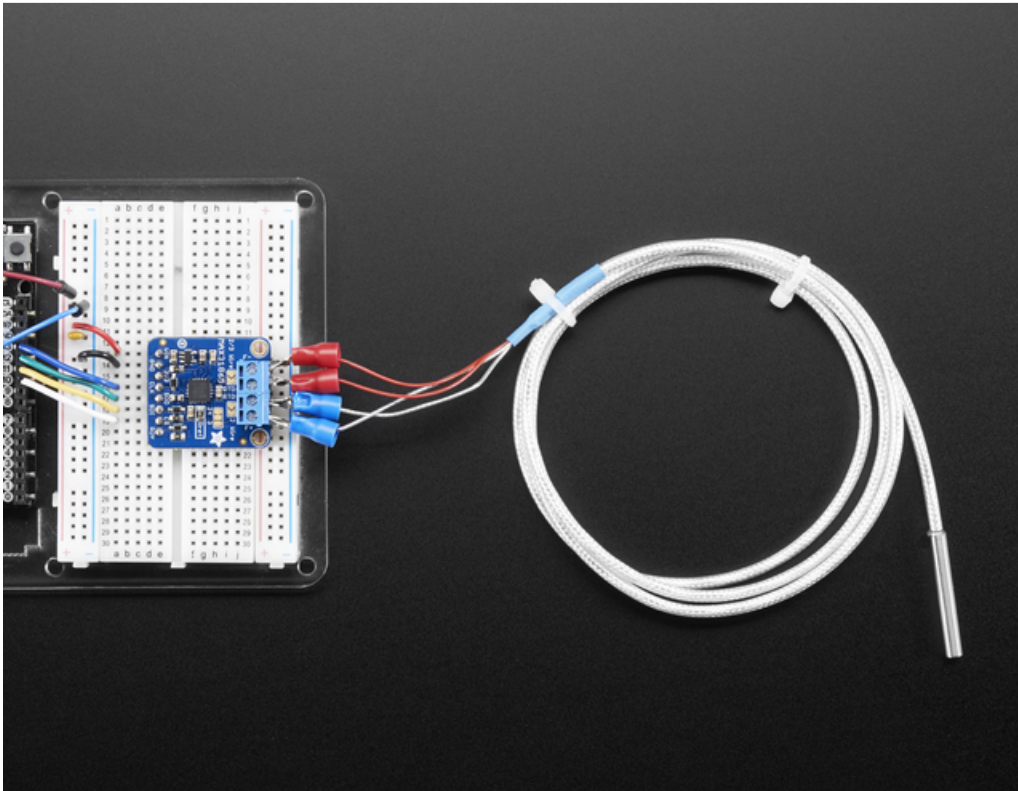


Last updated on 2017-12-02 12:08:40 AM UTC

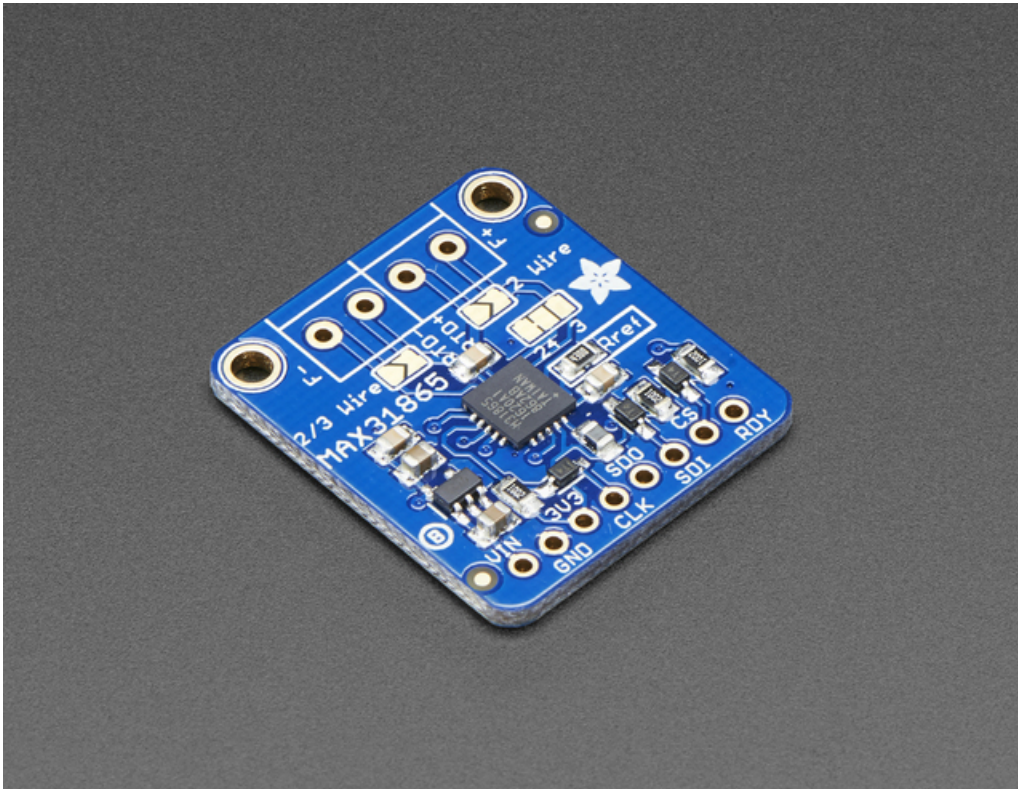
Guide Contents

Guide Contents	2
Overview	3
Pinouts	6
Power Pins:	6
SPI Logic pins:	6
Sensor Terminal Blocks	7
Configuration Jumpers	7
Assembly	9
Prepare the header strip:	10
Solder!	10
RTD Wiring & Config	15
4-Wire RTDs	15
3-Wire RTDs	15
2-Wire RTDs	16
How To Wire Up!	17
4-Wire Sensors	17
3-Wire Sensors	18
2 Wire Sensor	19
Wiring & Test	20
SPI Wiring	20
Download Adafruit_MAX31865 library	20
Attach PT100 or PT1000 RTD	21
Load Demo	21
More Accuracy	23
Library Reference	23
Reading Resistance	23
Calculating Temperature	23
Faults	24
Downloads	25
Files	25
Schematic & Fabrication Print	25

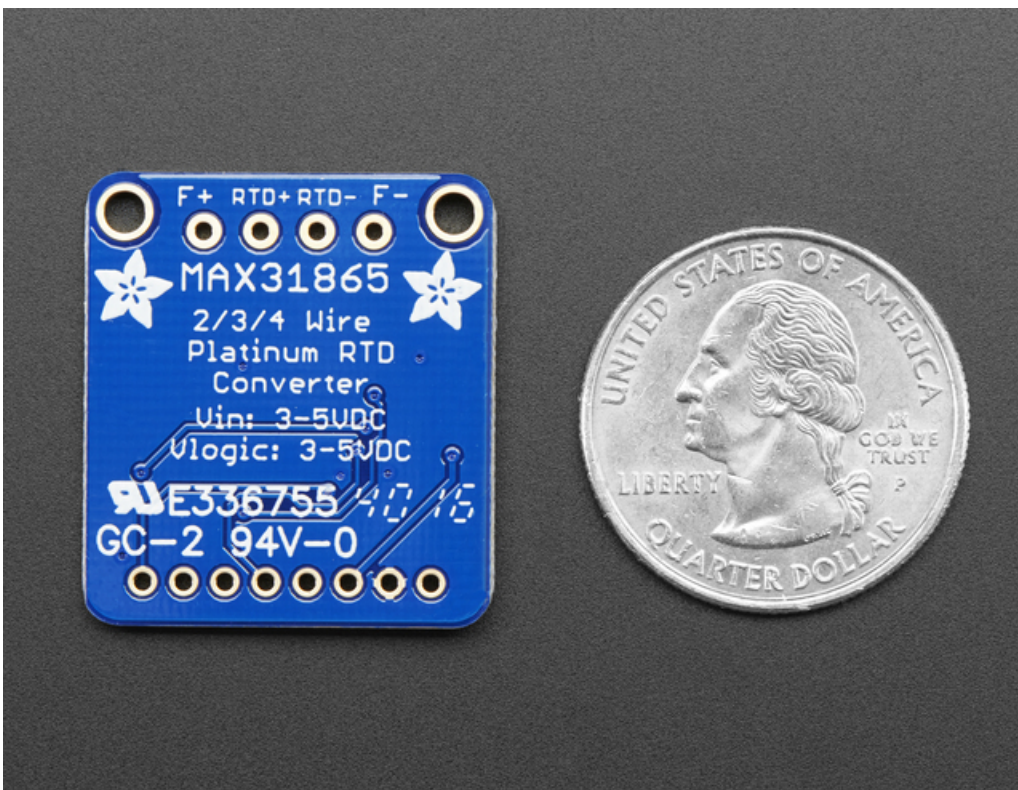
Overview



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.



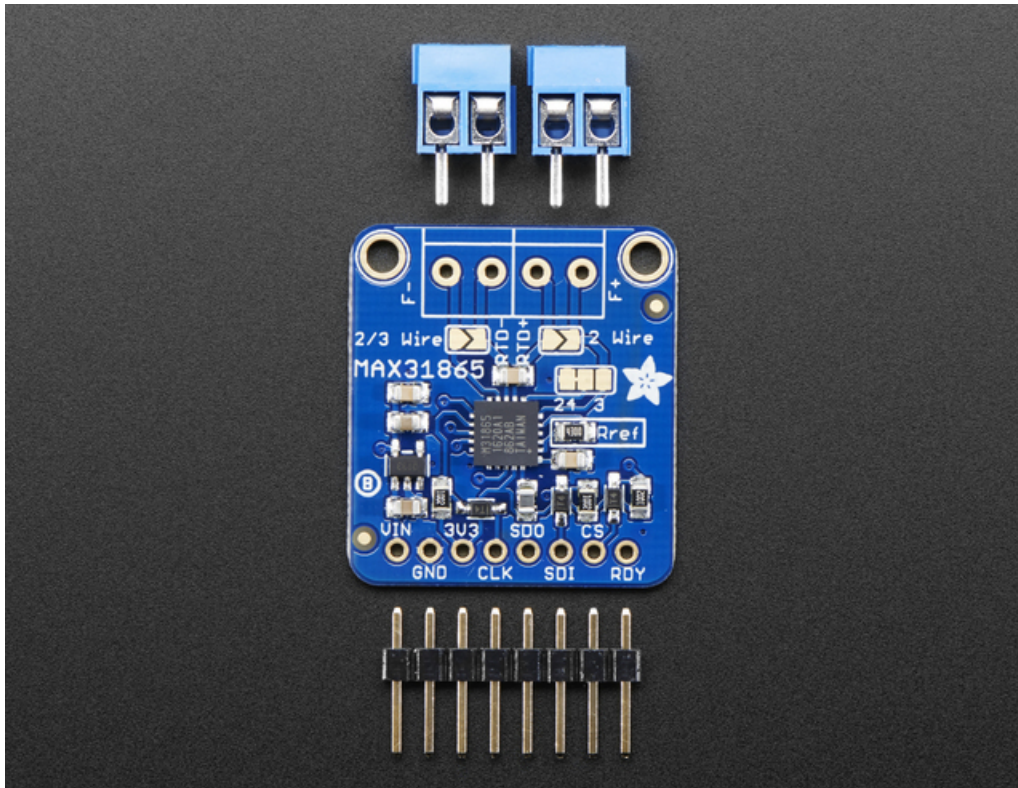
However, to get that precision and accuracy out of your PT100x RTD you must use an amplifier that is designed to read the low resistance. Better yet, have an amplifier that can automatically adjust and compensate for the resistance of the connecting wires. If you're looking for a great RTD sensor, today 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 calculate the temperature based on the resistance for you.

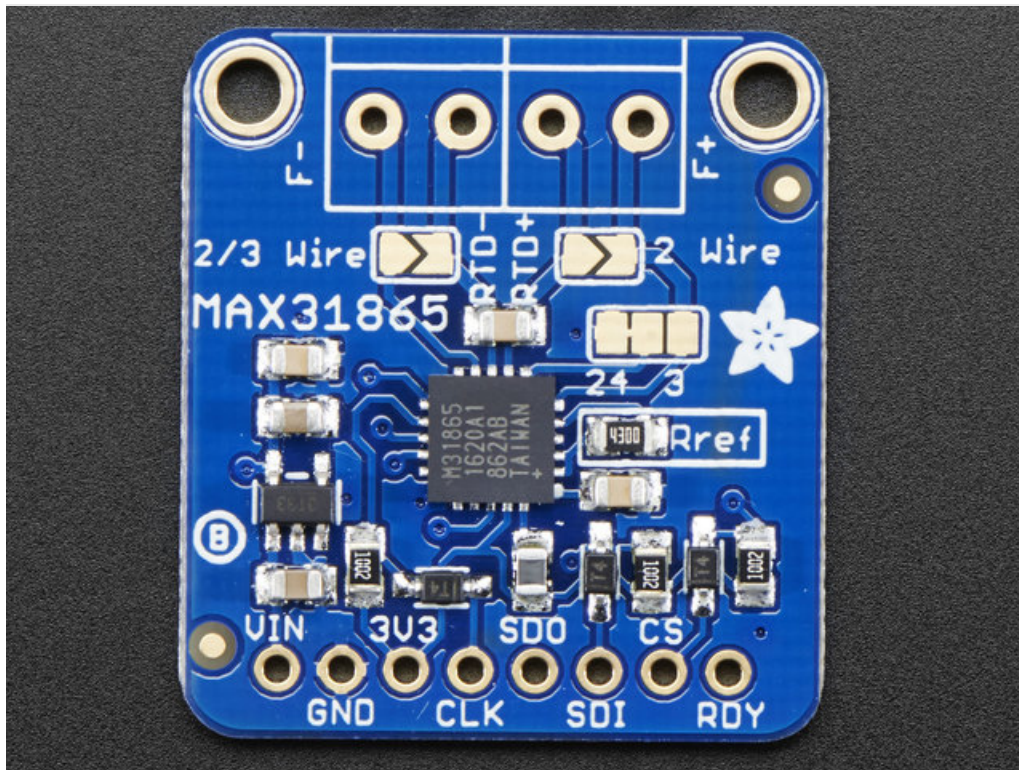
- The PT100 version of the breakout uses **430 Ω**
- The PT1000 version uses **4300 Ω**

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:

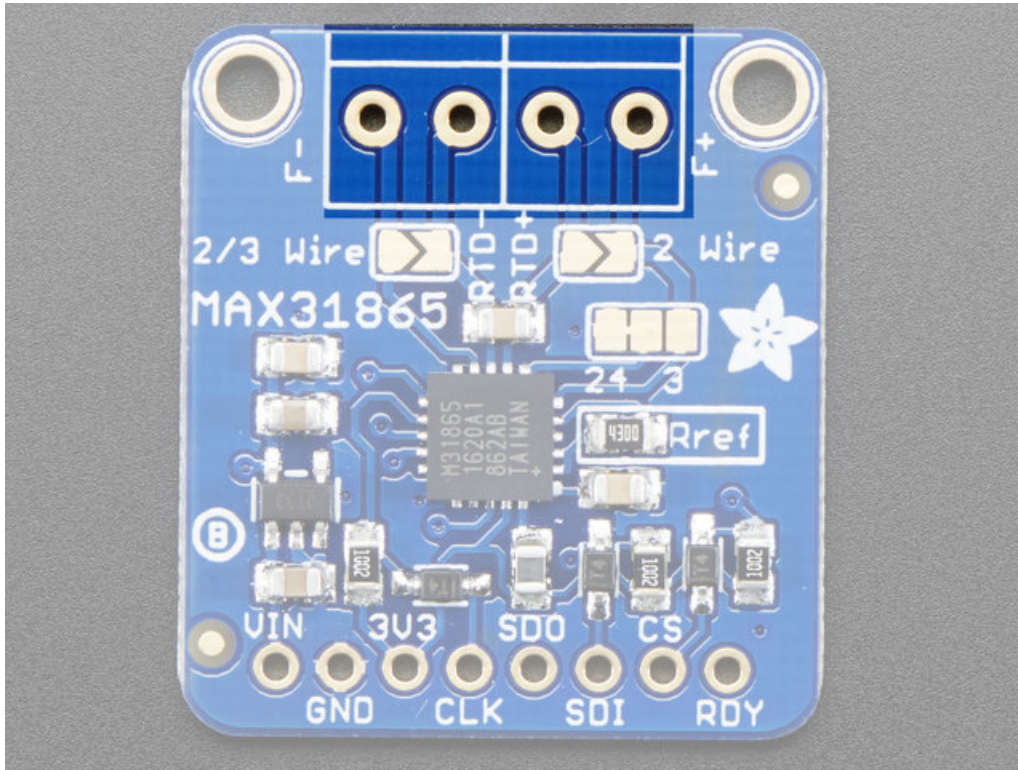
All pins going into the breakout have level shifting circuitry to make them 3-5V logic level safe. Use whatever logic level is on **Vin**!

- **SCK** - This is the **SPI Clock** pin, its an input to the chip
- **SDO** - this is the **Serial Data Out / Master In Slave Out** pin, for data sent from the MAX31865 to your processor
- **SDI** - this is the **Serial Data In / Master Out Slave In** pin, for data sent from your processor to the MAX31865
- **CS** - this is the **Chip Select** 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

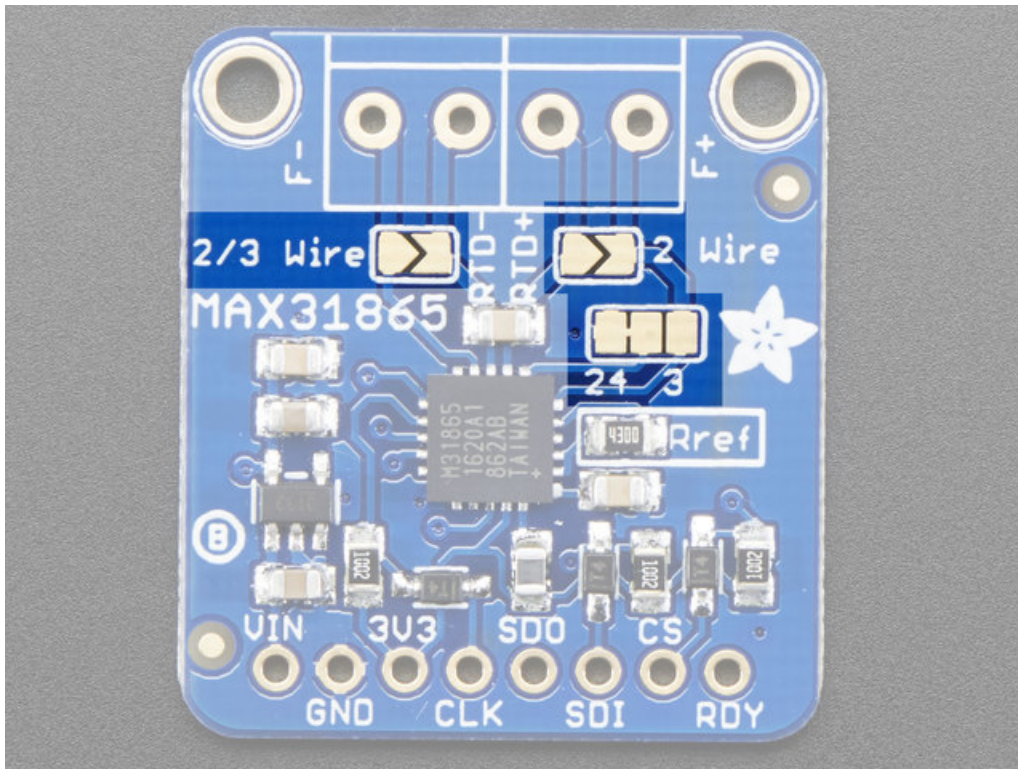


If you have an RTD sensor, you need to connect it somehow! the terminal block area is where you can clamp down to the sensor wires.

There are *four* contacts, but you can use 2, 3 or 4 wire sensors. You may need to solder or jumper some pads depending 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 don't 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.

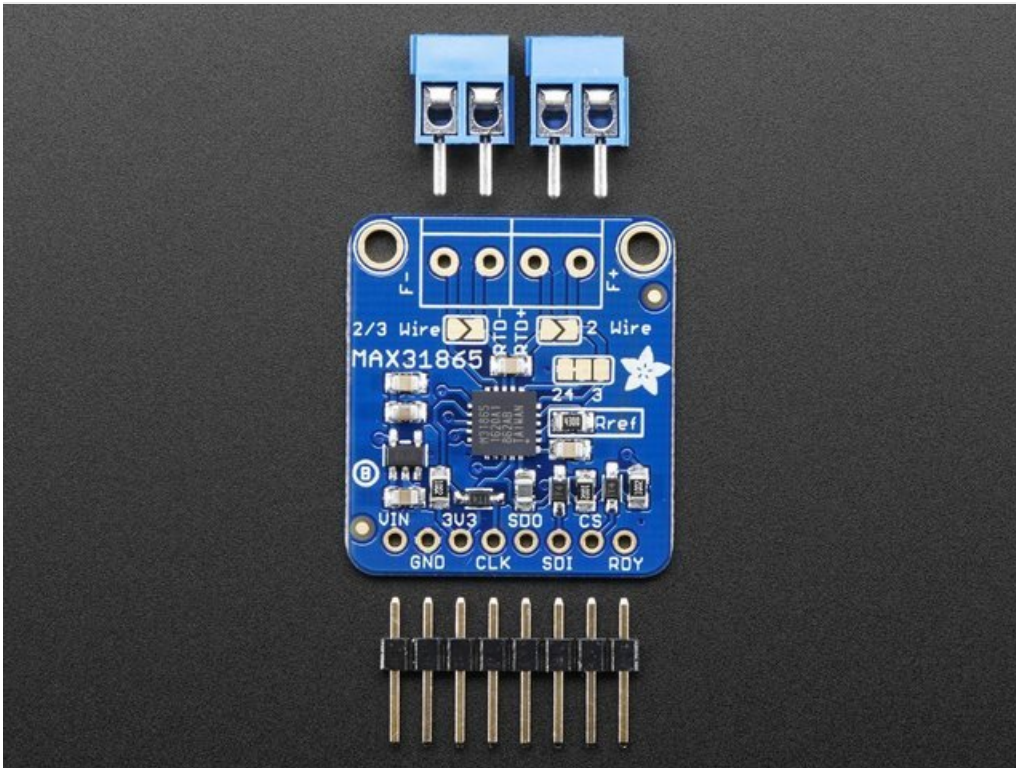
For 4-wire usage, do nothing with the jumpers!

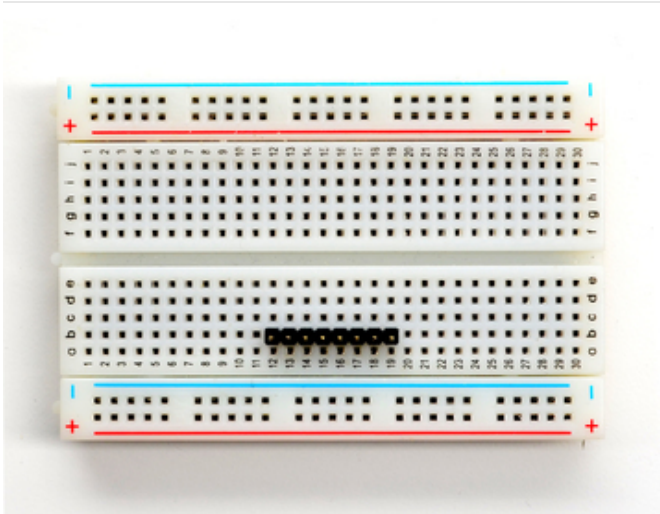
For 3-wire usage. Solder closed the jumper labeled **2/3 Wire** and cut the wire connecting the left side of the 2-way jumper right above Rref. Then solder closed the right side labeled **3**

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!

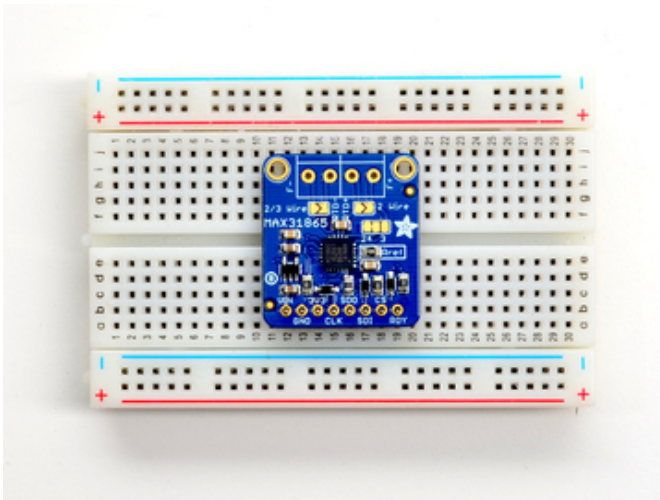
Assembly





Prepare the header strip:

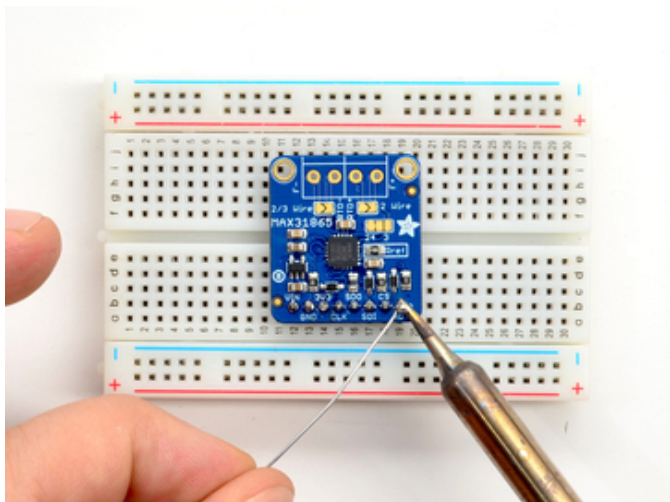
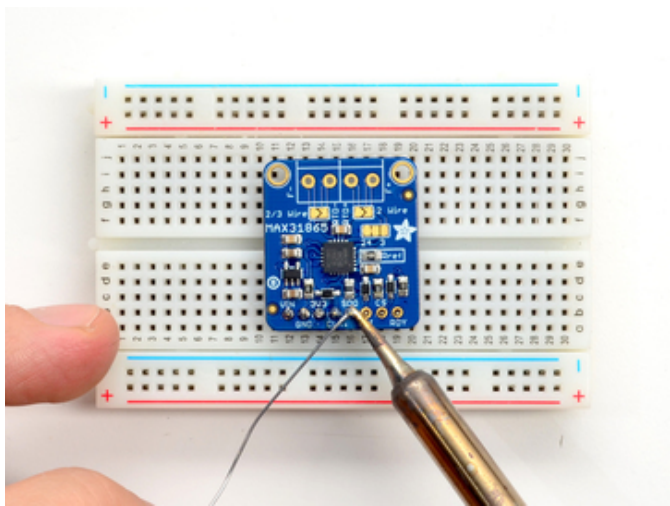
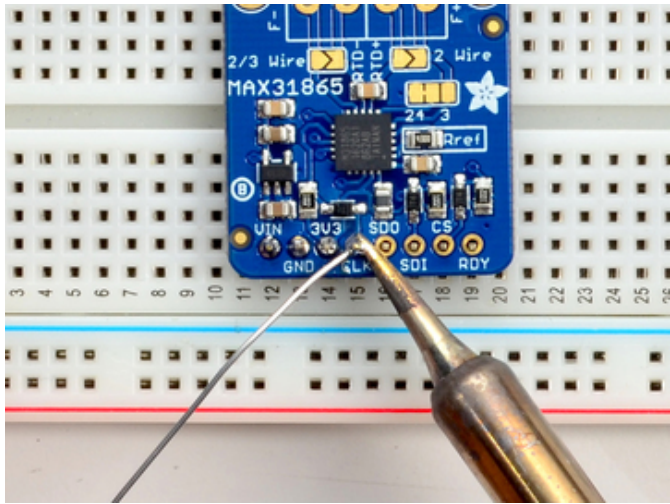
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

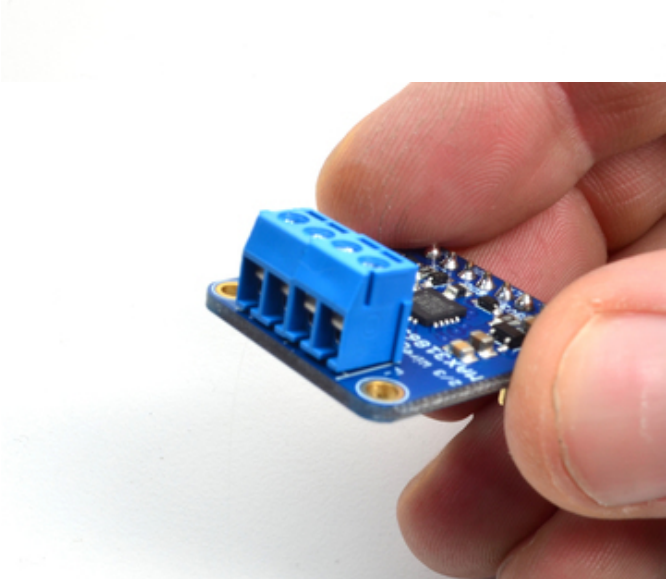


Solder!

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://adafruit.it/aTk) (<https://adafruit.it/aTk>)).

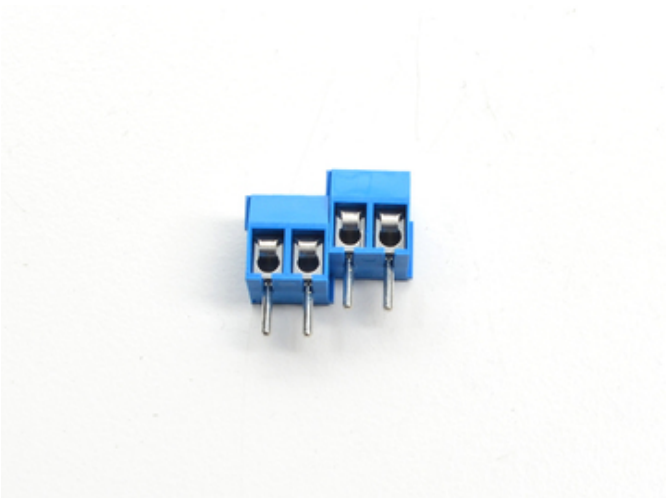
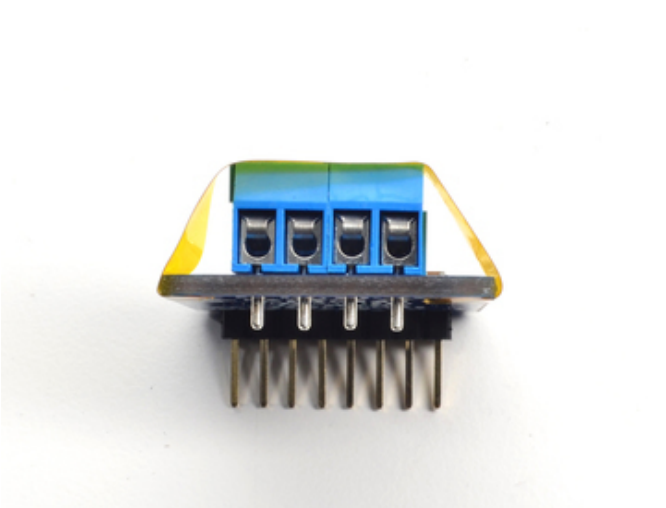




Next we will solder in the two 3.5mm terminal blocks used to connect power & the motor to the breakout board.

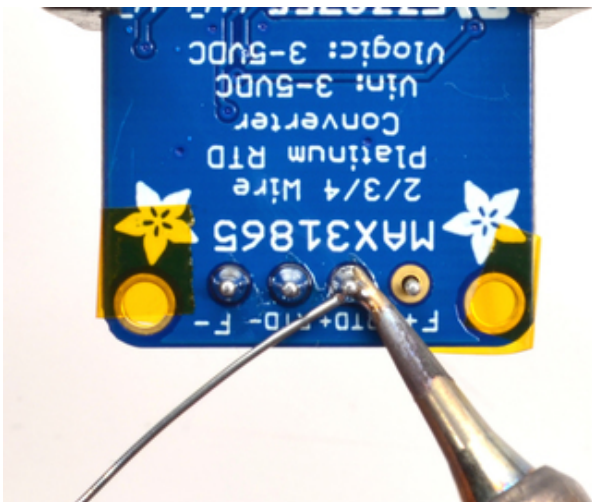
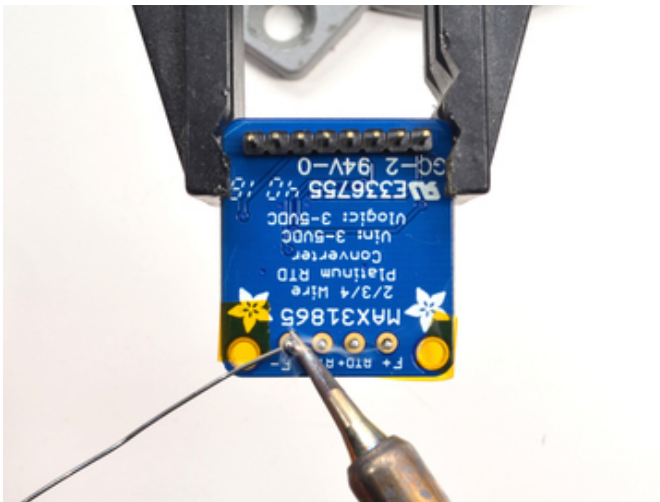
Make sure the open parts of the terminals face outwards so you can easily connect wires

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!



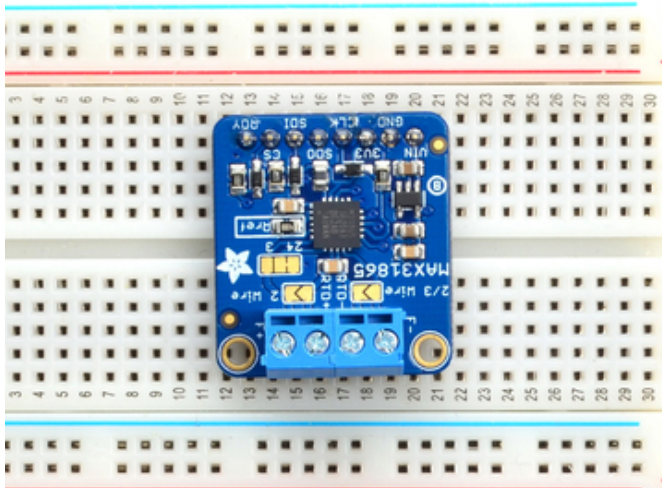
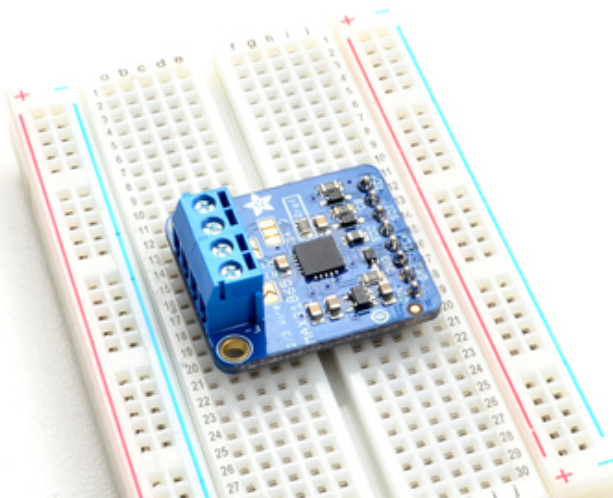


Solder in both pins of each terminal block. You can remove the tape when done.





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.

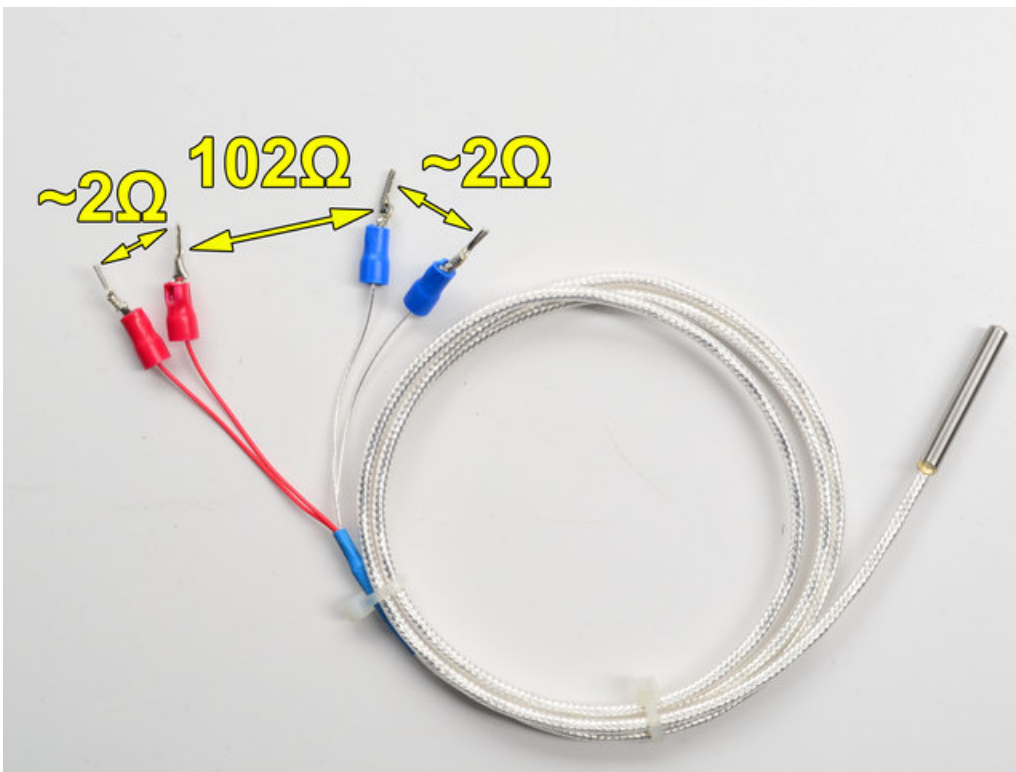
4-Wire RTDs

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 $^\circ\text{C}$! 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.

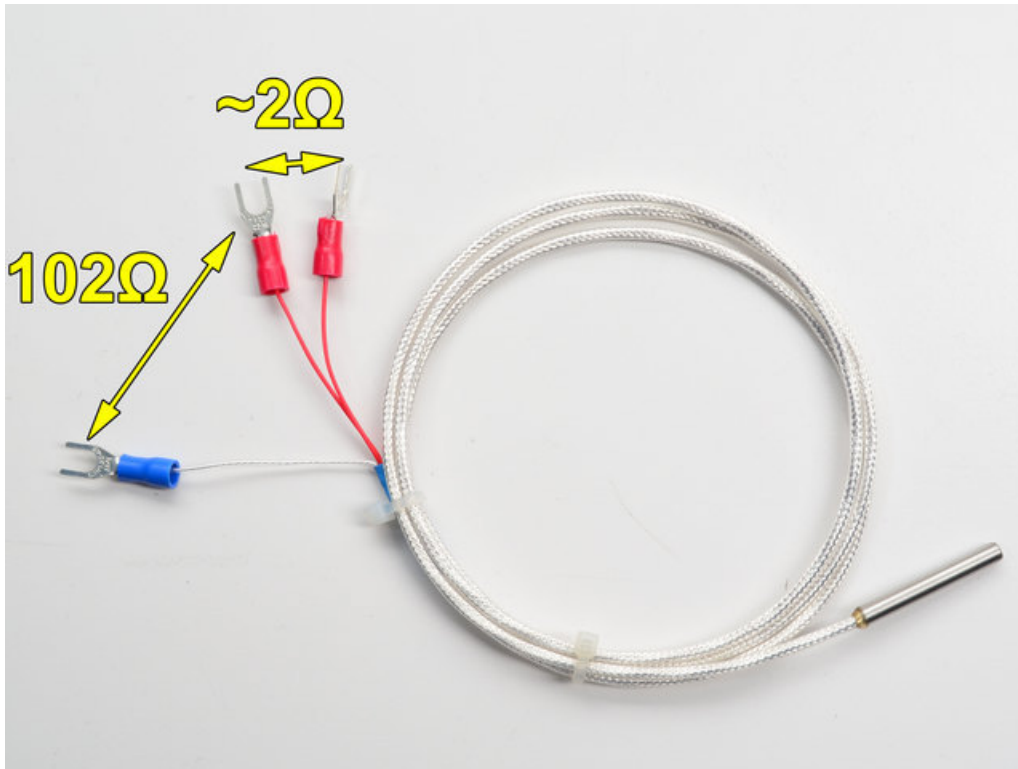
For example, here's the approximate resistances of a 4-Wire PT100 RTD at 0°C (for a PT1000, the middle resistance would be 1002Ω rather than 102Ω)



(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 $102 - 1 - 1 = 100\Omega$

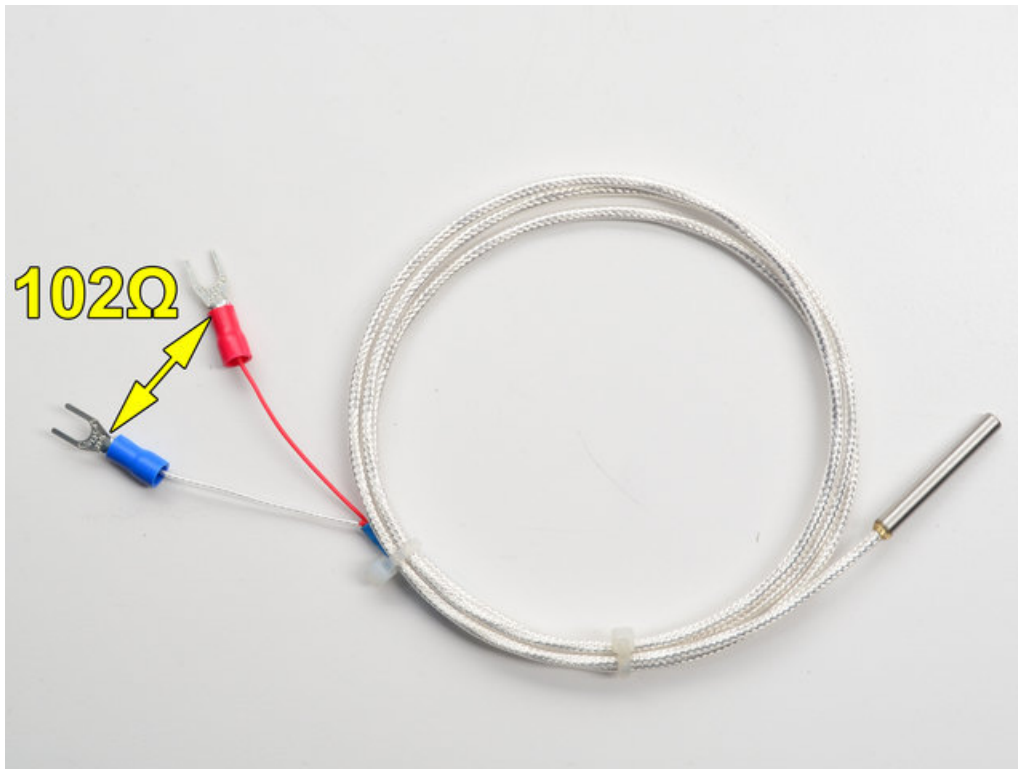
3-Wire RTDs

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!

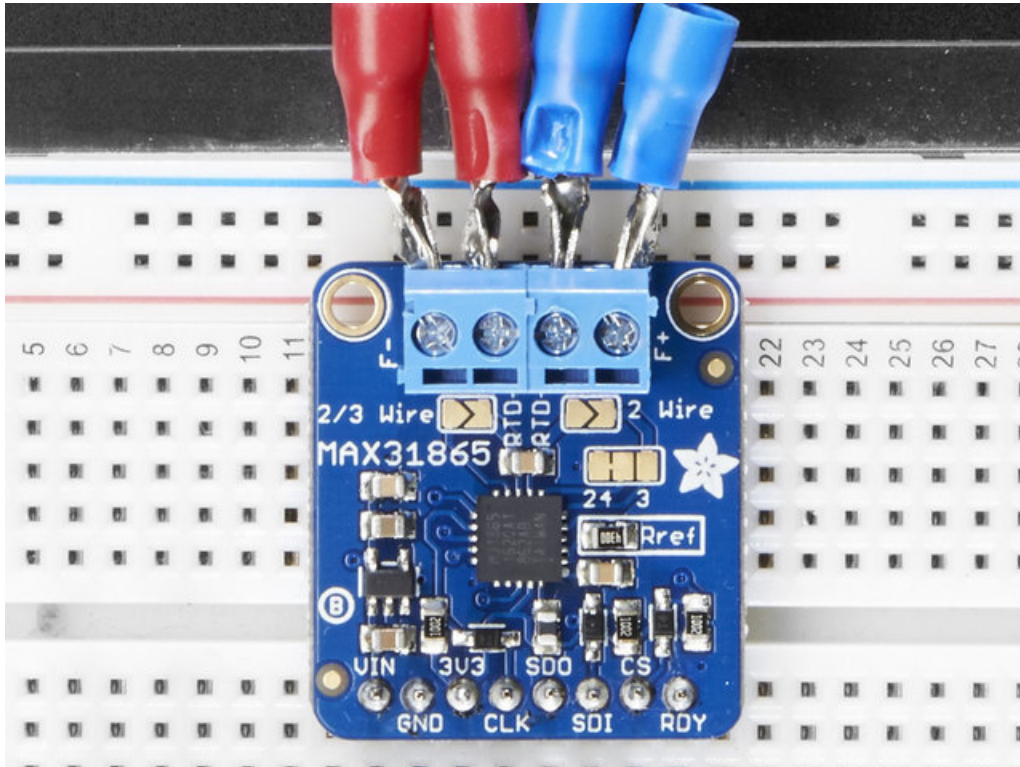


How To Wire Up!

4-Wire Sensors

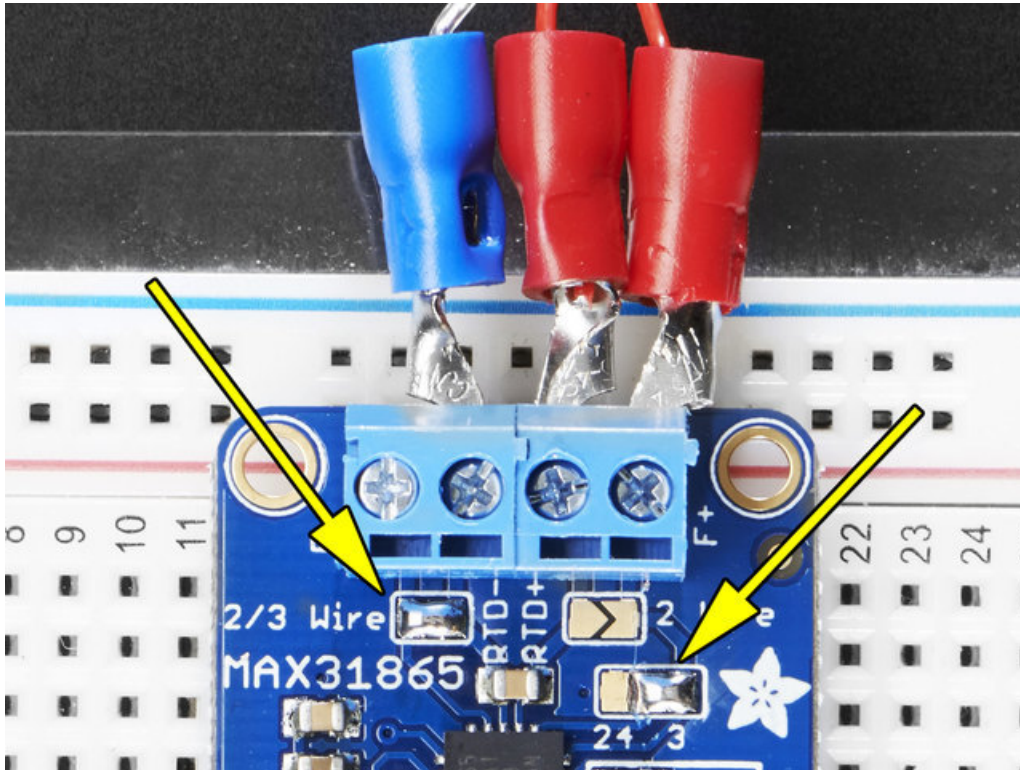
Connect the four wires to each of the pads. Use a multimeter to determine which wires connect together directly (2 ohms or so between them) and which connect through the RTD. Chances are the wires that connect together are the same color. The two pairs connect so that the ones that are connected together 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.

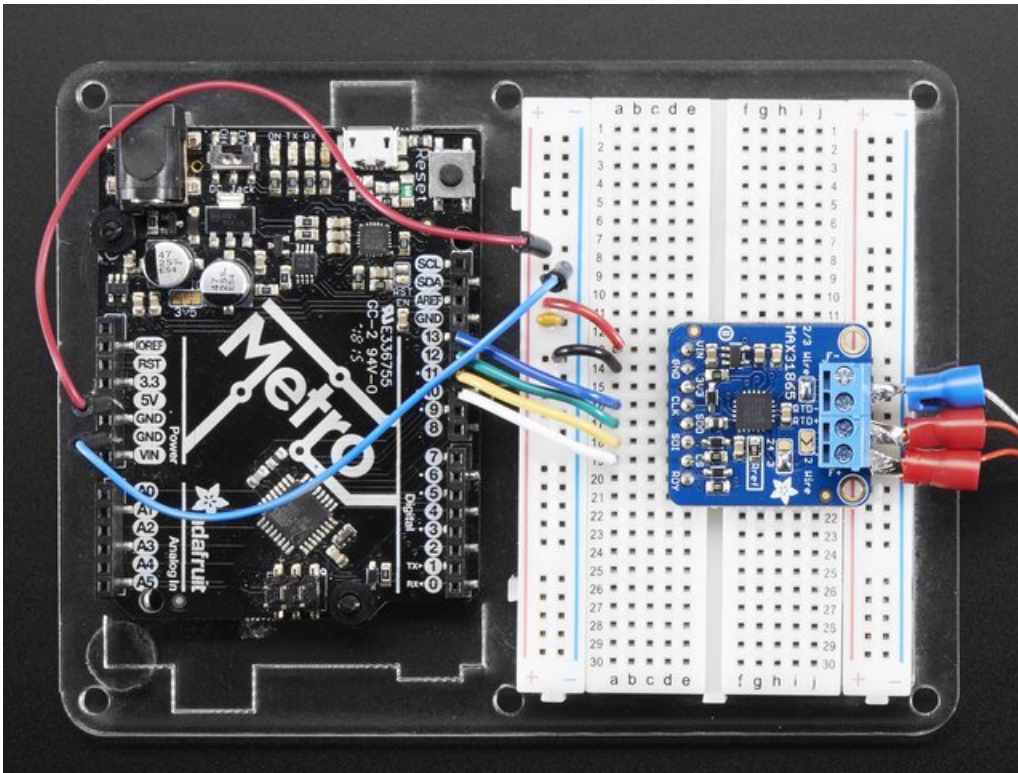
Then next to the terminal block on the left, solder closed that jumper as well. Alternatively you can put a piece of wire into the terminal blocks to 'short' them

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.

Wiring & Test

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

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

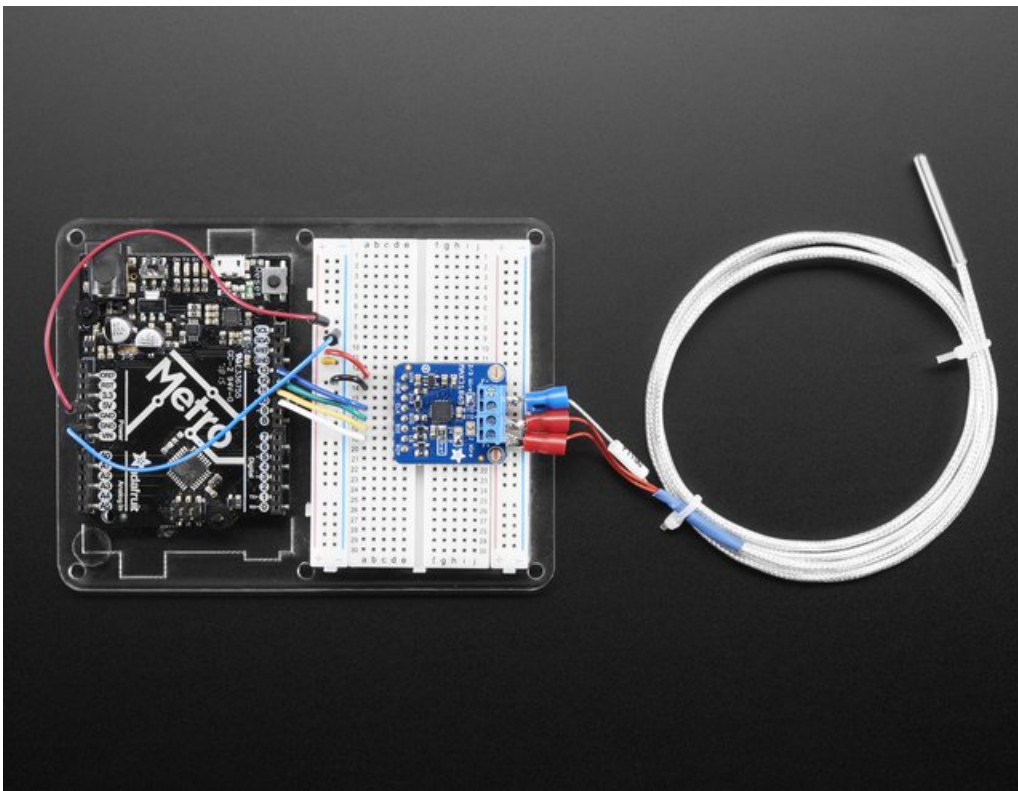
Place the **Adafruit_MAX31865** library folder your **arduinofolder/libraries/** folder.
You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

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

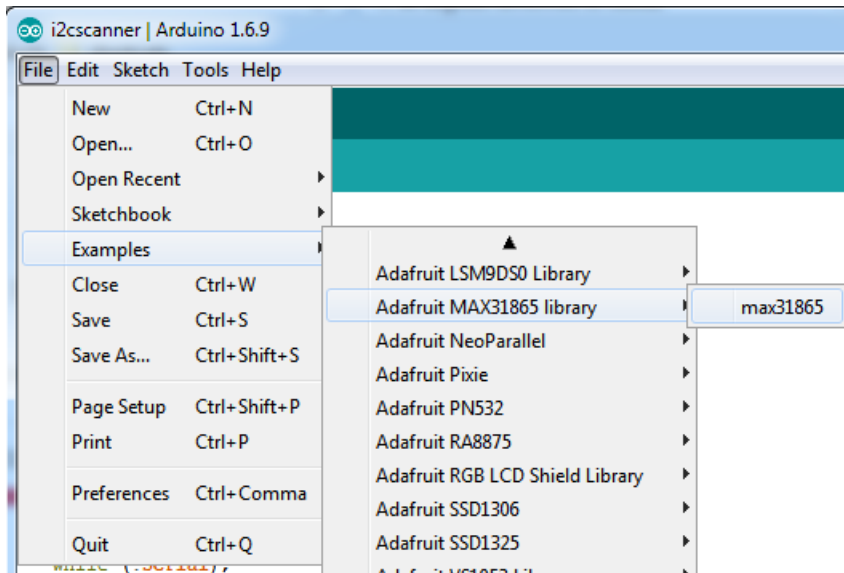
Attach PT100 or PT1000 RTD

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

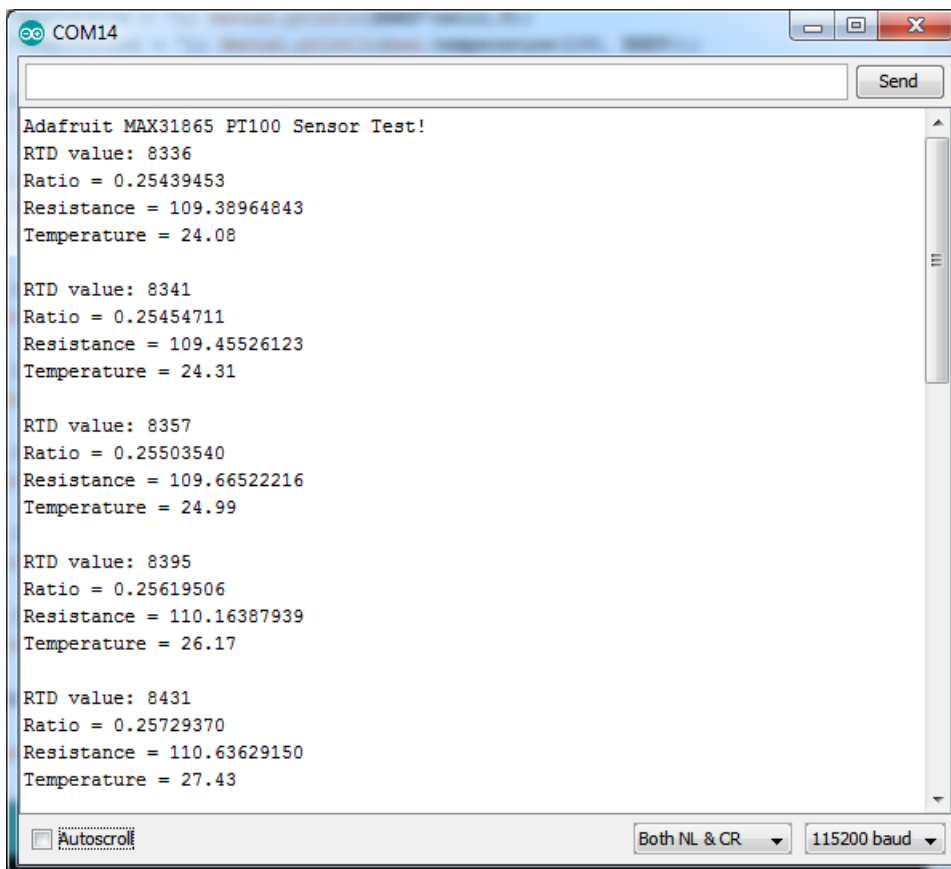
Open up **File->Examples->Adafruit_MAX31865->max31865** and upload to your Arduino wired up to the sensor. Adjust the `max.begin(MAX31865_3WIRE)` line if necessary.



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



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/application->

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 temperature 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.

