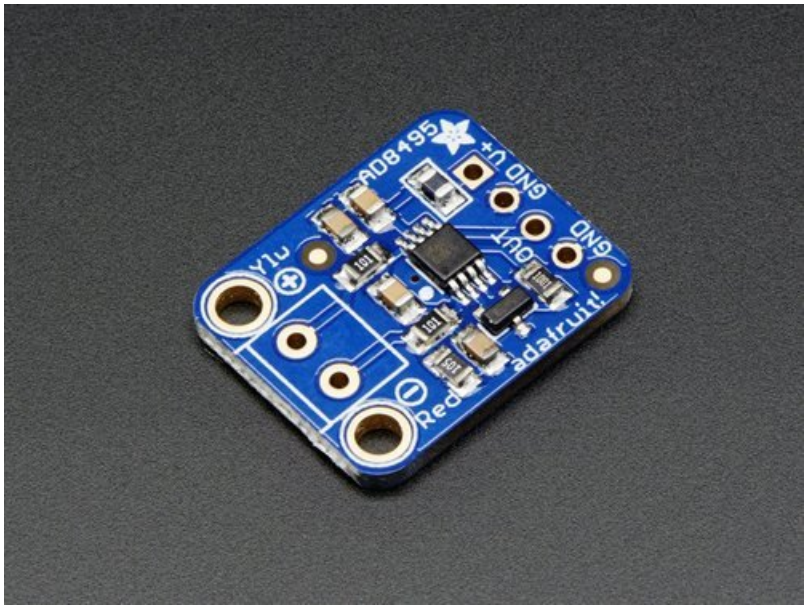
adafruit learning system

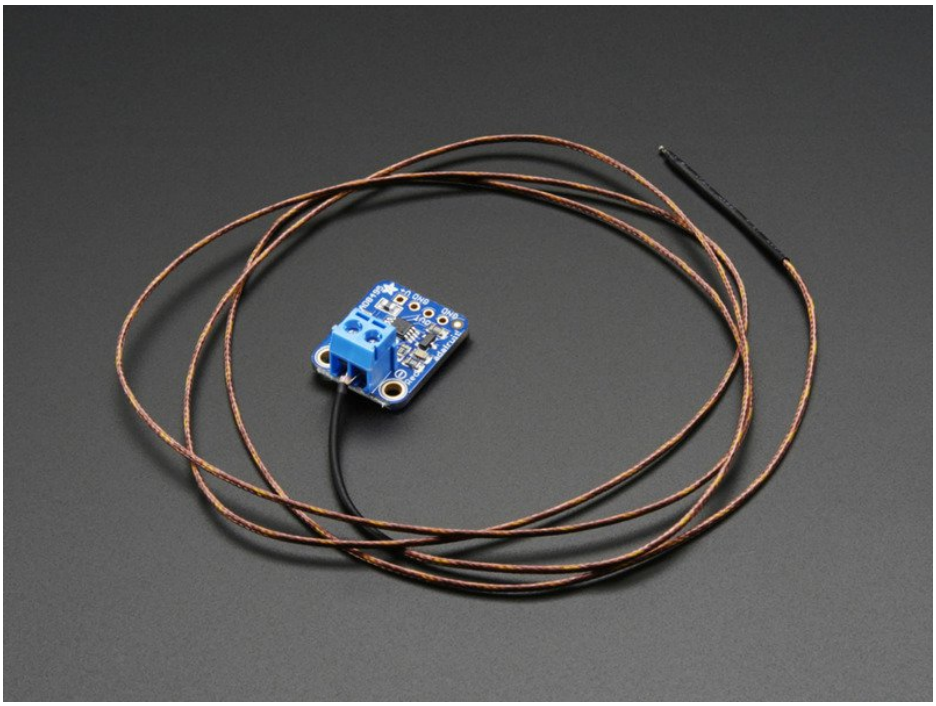
AD8495 Analog Output K-Type Thermocouple Amplifier

Created by Kattni Rembor

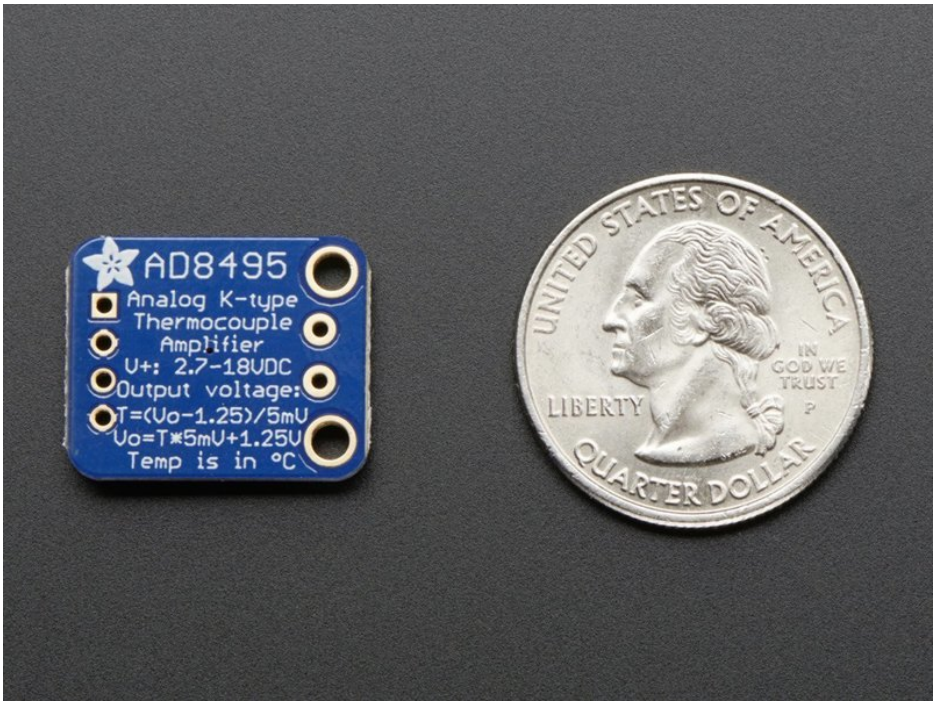


Last updated on 2019-05-03 05:28:45 PM UTC

Overview

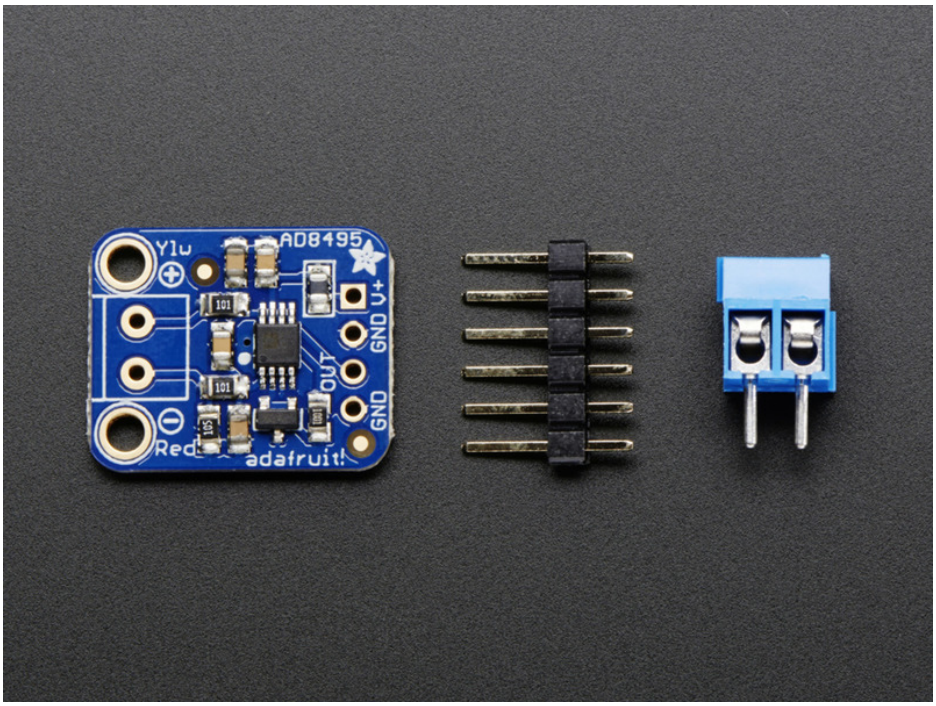


Thermocouples are very sensitive, requiring a good amplifier with a cold-compensation reference. We have a couple digital thermocouple amplifiers in the shop already from Maxim. Now we're happy to introduce an excellent analog-output amplifier. This is a very simple sensor to use, and if your microcontroller has analog input capability, you'll be ready to go really fast!



The AD8495 K-type thermocouple amplifier from Analog Devices is so easy to use, we documented the whole thing on the back of the tiny PCB. Power the board with 3-18VDC and measure the output voltage on the **OUT** pin. You can

easily convert the voltage to temperature with the following equation: Temperature = (Vout - 1.25) / 0.005 V. So, for example, if the voltage is 1.5VDC, the temperature is (1.5 - 1.25) / 0.005 = 50°C.

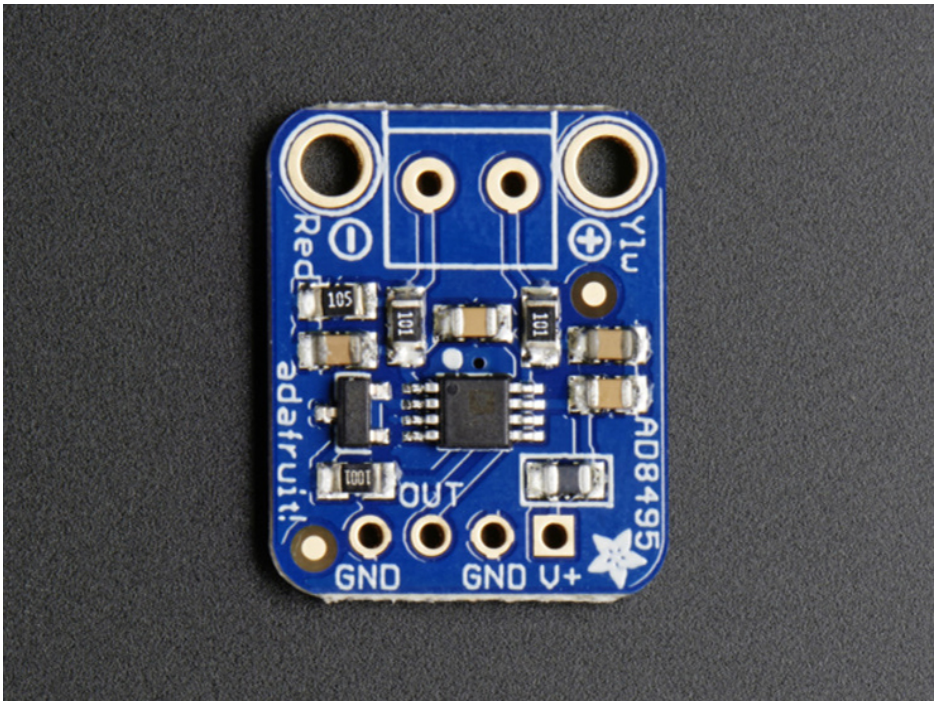


Each order comes with a 2 pin terminal block (for connecting to the thermocouple), a fully assembled PCB with the AD8495 + TLVH431 1.25V precision voltage reference, and pin header (to plug into any breadboard or perfboard). Goes great with our 1m K-type thermocouple (not included) (http://adafru.it/270). Not for use with any other kind of thermocouple, K type only!

- Works with any K type thermocouple
- Will not work with any other kind of thermocouple other than K type
- Easy to use analog output
- Temp range with 5V power: **-250°C to +750°C output** (0 to 5VDC) as long as the thermocouple can handle that range
- Temp range with 3.3V power: **-250°C to +410°C output** (0 to 3.3VDC) as long as the thermocouple can handle that range
- For higher temperatures you'll need to power with a higher voltage, so you can get the analog reading out.

Note: The terminal blocks included with your product may be blue or black.

Pinouts



Power Pins

- **V+** - This is the power pin. This board works with 3.3V and 5V power. The temp range with 5V power is -250°C to +750°C output (0 to 5VDC), and with 3.3V power is -250°C to +410°C output (0 to 3.3VDC), as long as the thermocouple can handle the range.
- **GND** - Common ground pins

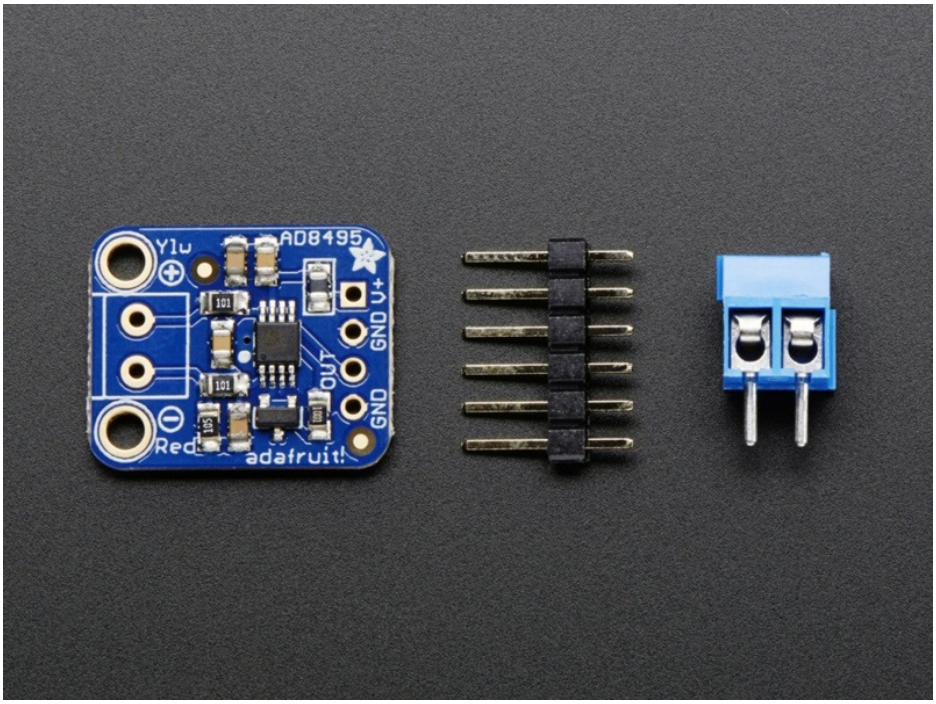
Analog Pin

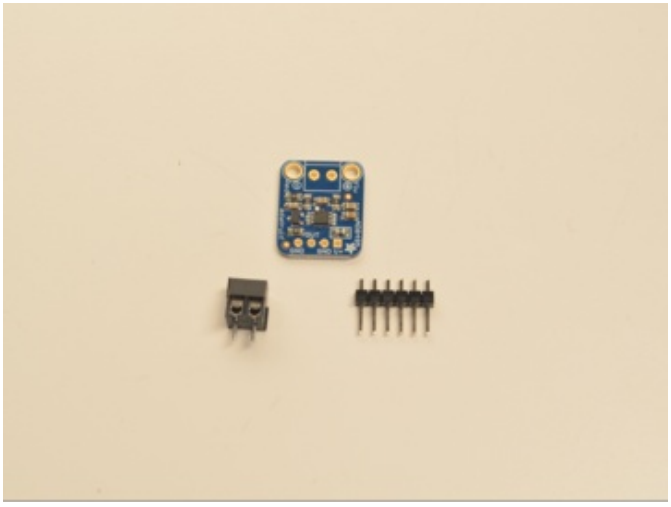
- **OUT** - This is the output voltage pin. Read using an analog input.

Thermocouple Terminal

- **Red-** - Connect the red or - wire on your thermocouple to this side of the screw terminal
- **Ylw+** - Connect the yellow or + wire on your thermocouple to this side of the screw terminal

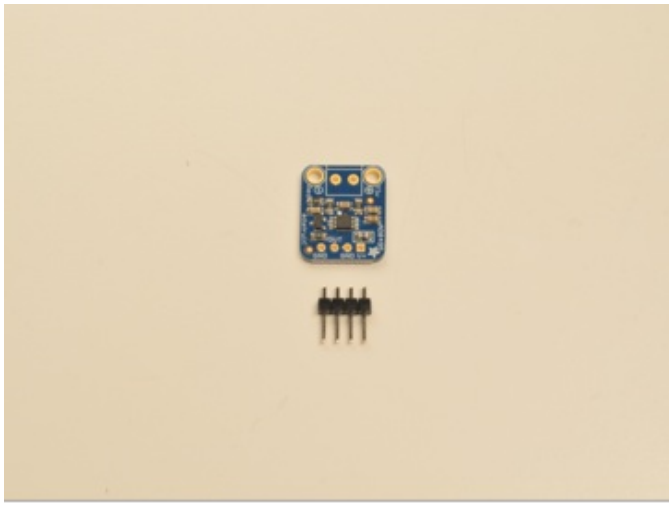
Assembly

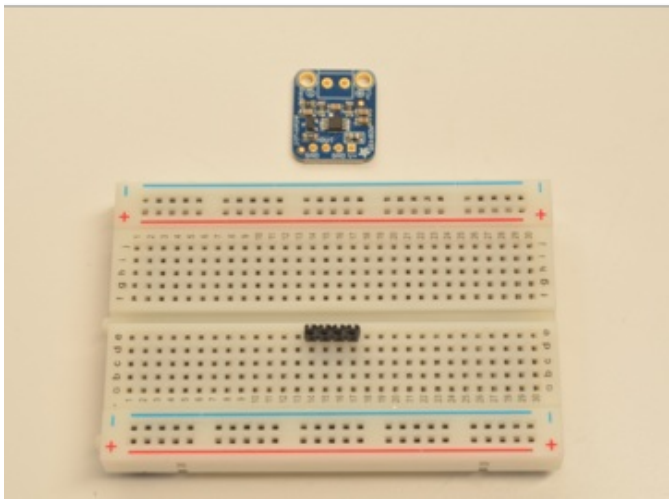


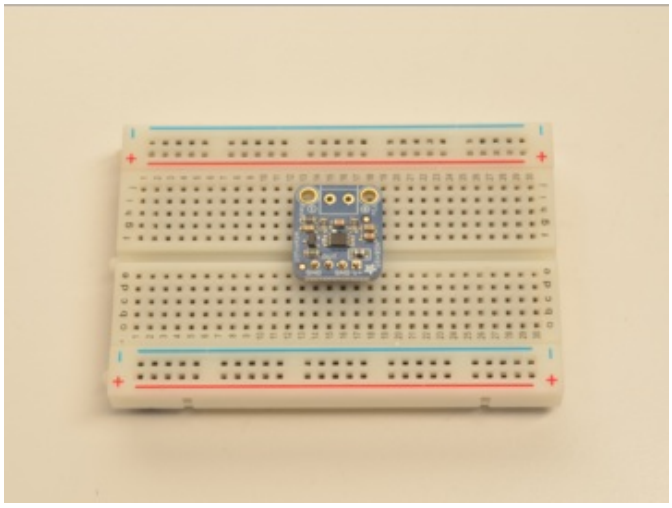


Prepare the header strip:

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







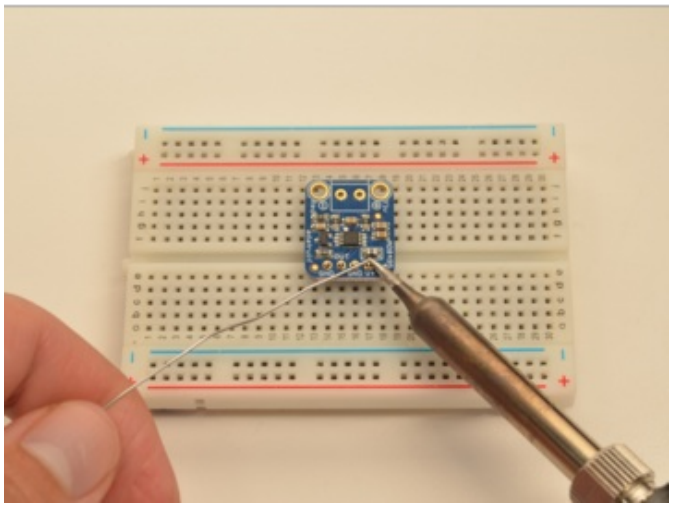
Add the breakout board:

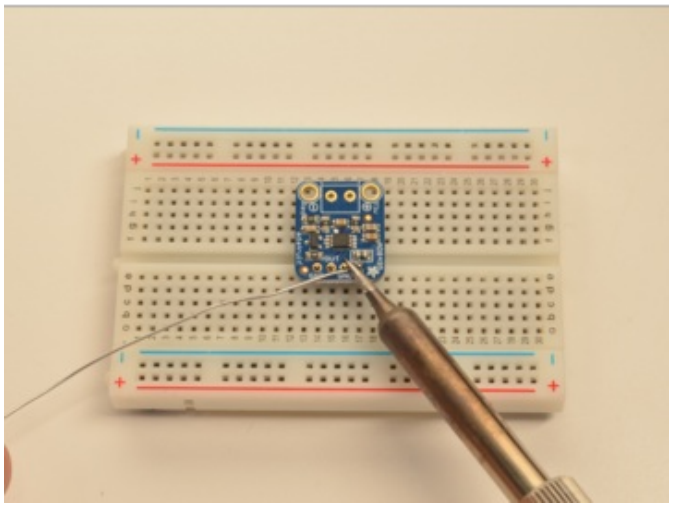
Place the breakout board over the pins so that the short pins poke through the breakout pads.

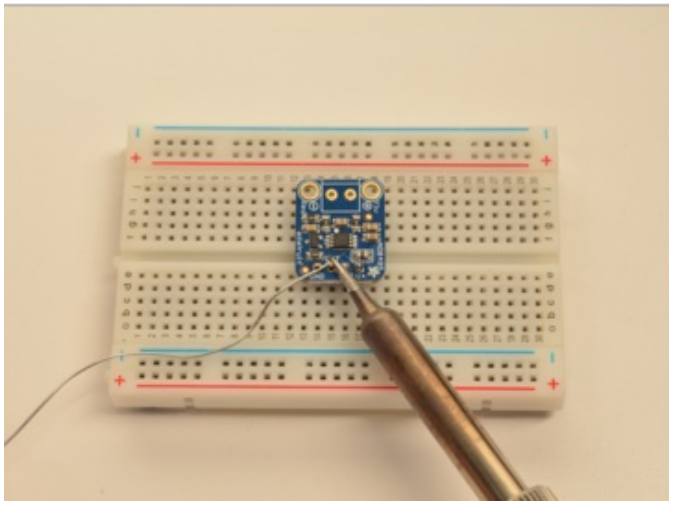
And Solder!

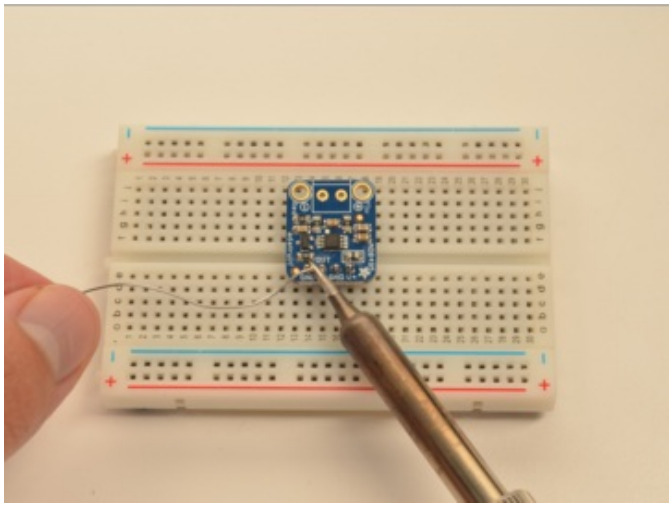
Be sure to solder all 4 pins for reliable electrical contact.

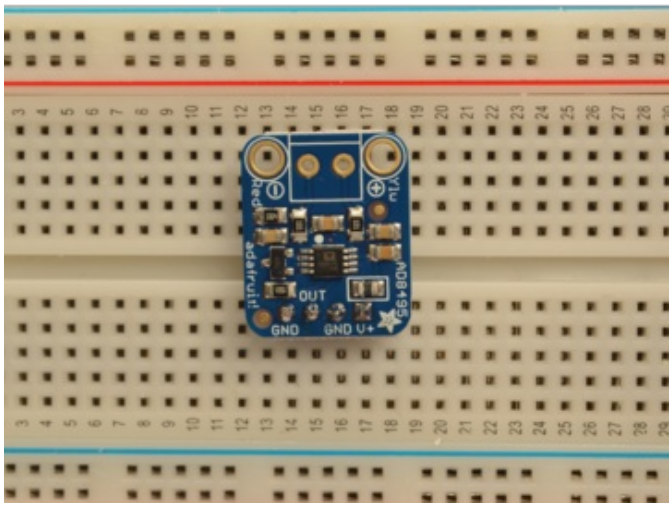
(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk).



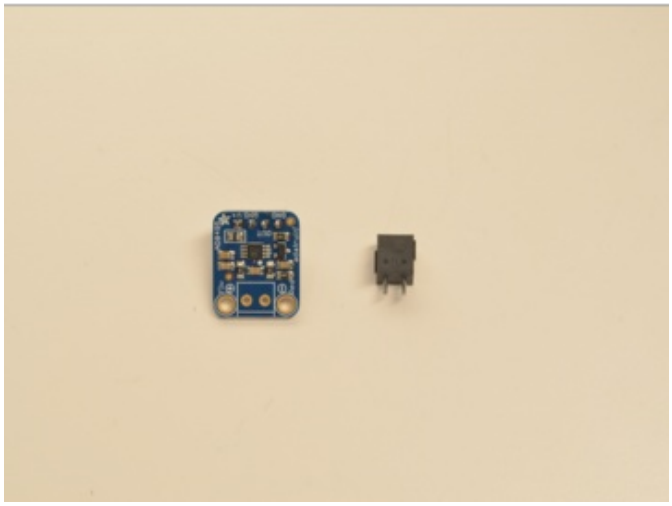






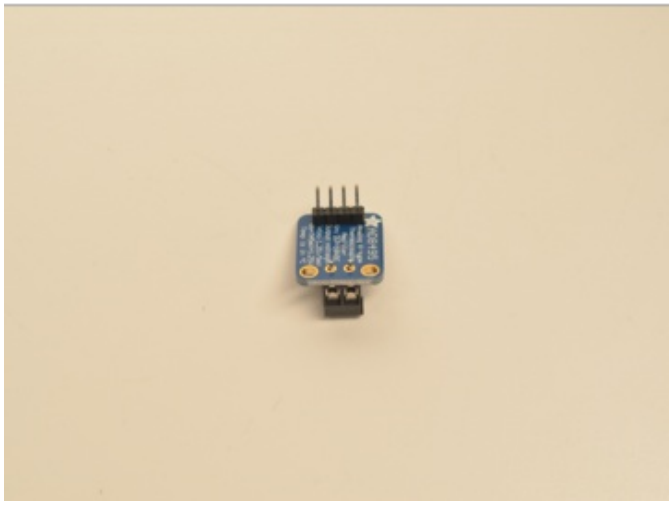


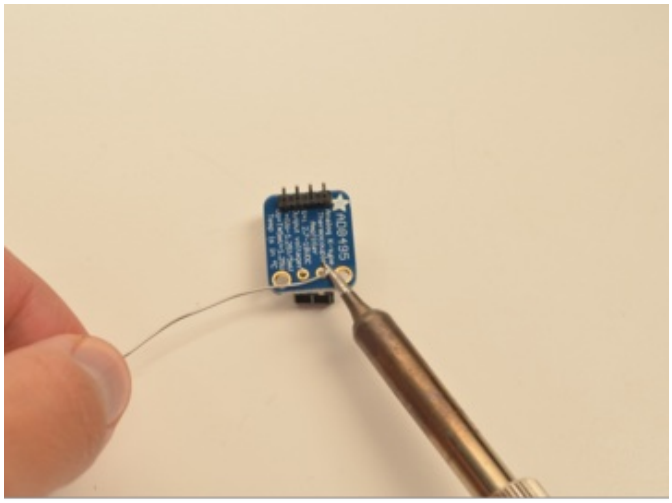
You're done with the header strip! Check your solder joints visually.



Prepare the terminal block:

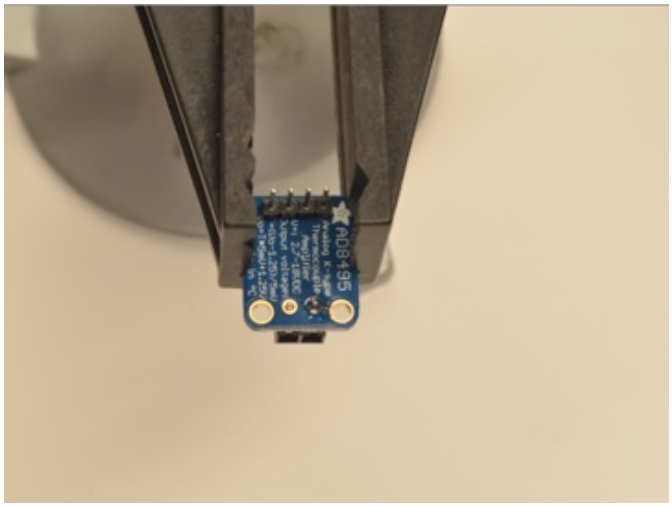
Place the terminal block inside the board. Make sure it's facing out! Turn the board over so the terminal pins are facing upwards.

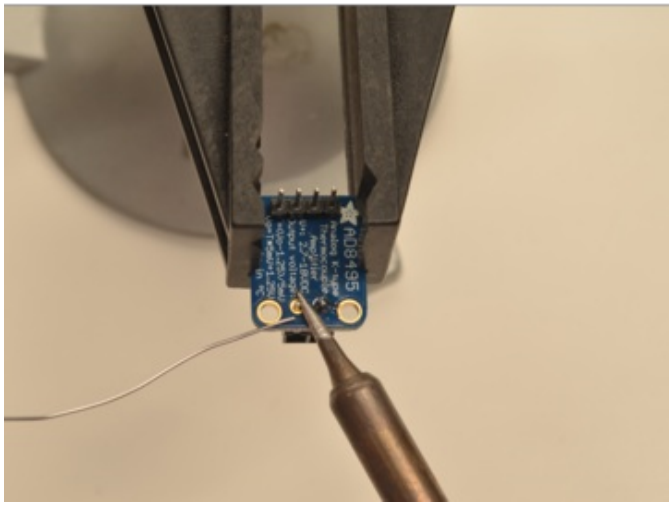


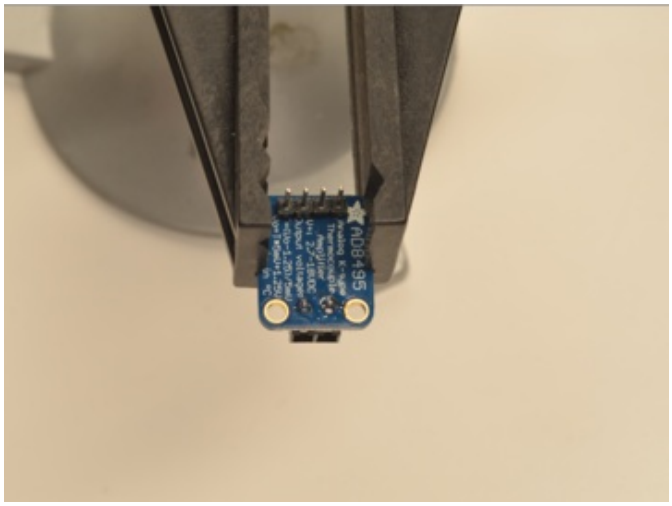


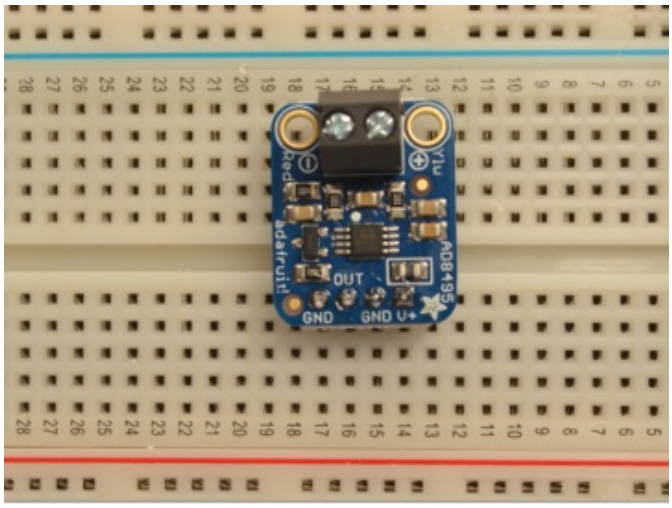
Solder again!

First solder one pin by using the soldering iron to prop up the board. Then use a vice (https://adafru.it/dDJ) or helper hands to hold the board in place while soldering the other pin.


AD8495


AD8495


AD8495



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

Arduino

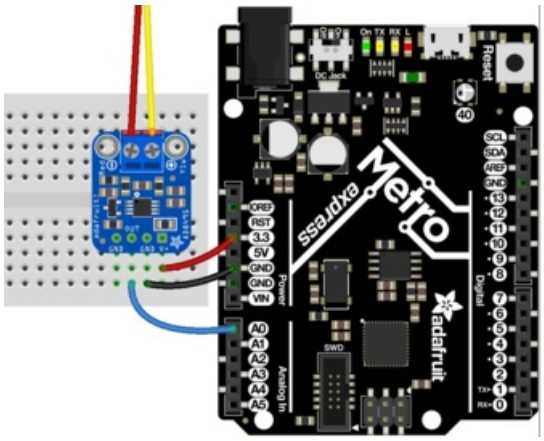
It's easy to wire this breakout to a microcontroller to use with the Arduino IDE. Keep in mind the output sensitivity is only 0.005 volts per degree Celsius. This is due to the overall large range of temperatures that can be measured.

If you are using a low resolution ADC (like 10 bits or less) and trying to discern small temperature differences (like only a couple of degrees), it may be difficult to pull that signal out of the noise.

For example, the Microchip SAMD21 M0 boards such as the Adafruit Metro M0 Express (https://adafru.it/EyR) offer 12 bit analog inputs.

Wiring

Wire up the AD8495 as follows. This example uses a Metro M0 Express:



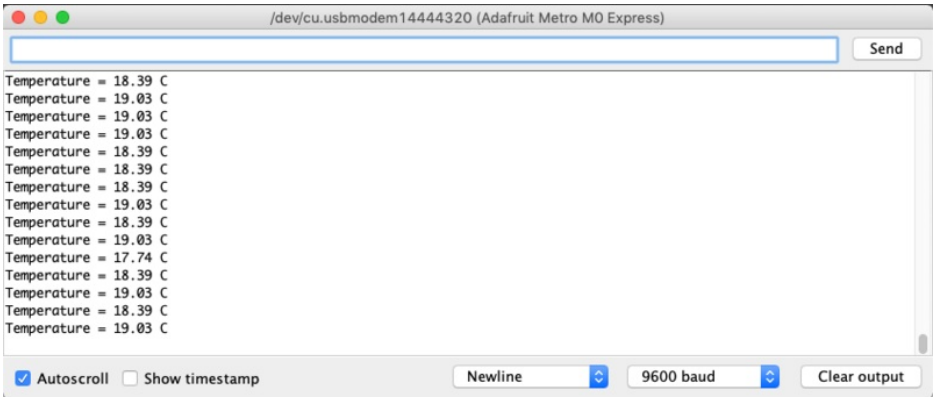
- **Board V+** to **Metro 3.3V**
- **Board GND** to **Metro GND**
- **Board OUT** to **Metro A0**
- **Board RED-** to **thermocouple red- wire**
- **Board YLW+** to **thermocouple yellow+ wire**

Load Demo

Download the following file and save it to your computer.

```
#define TC_PIN A0          // set to ADC pin used
#define AREF 3.3           // set to AREF, typically board voltage like 3.3 or 5.0
#define ADC_RESOLUTION 10  // set to ADC bit resolution, 10 is default

float reading, voltage, temperature;

float get_voltage(int raw_adc) {
  return raw_adc * (AREF / (pow(2, ADC_RESOLUTION)-1));
}

float get_temperature(float voltage) {
  return (voltage - 1.25) / 0.005;
}

void setup() {
  Serial.begin(9600);
}

void loop() {
  reading = analogRead(TC_PIN);
  voltage = get_voltage(reading);
  temperature = get_temperature(voltage);
  Serial.print("Temperature = ");
  Serial.print(temperature);
  Serial.println(" C");
  delay(500);
}
```

Open the file you downloaded in the Arduino IDE, and upload it to your Arduino. Once uploaded to your Arduino, open up the serial console at 9600 baud speed to see data being printed out.



Usage

First, set the ADC pin to be used, choose the voltage you'll be connecting to, set the ADC resolution and set `reading`, `voltage` and `temperature` to floats.

```
#define TC_PIN A0          // set to ADC pin used
#define AREF 3.3           // set to AREF, typically board voltage like 3.3 or 5.0
#define ADC_RESOLUTION 10  // set to ADC bit resolution, 10 is default

float reading, voltage, temperature;
```

Next are two functions: one to take the raw ADC value and turn it into voltage, and one to turn the voltage into temperature.

```
float get_voltage(int raw_adc) {
  return raw_adc * (AREF / (pow(2, ADC_RESOLUTION)-1));
}

float get_temperature(float voltage) {
  return (voltage - 1.25) / 0.005;
}
```

The `get_voltage` function uses the raw ADC value, and the board voltage and resolution specified above to calculate the output voltage. Check out this Arduino tutorial (https://adafru.it/EuH) for more information.

The `get_temperature` function takes the output voltage from the `get_voltage` function and uses it to calculate the temperature.

In the main loop, we take the analog `reading`, apply it to `get_voltage` to get `voltage`, and then apply the `voltage` to `get_temperature` to get `temperature`. Then we print the temperature to the serial monitor every 0.5 seconds.

```
  reading = analogRead(TC_PIN);
  voltage = get_voltage(reading);
  temperature = get_temperature(voltage);
  Serial.print("Temperature = ");
  Serial.print(temperature);
  Serial.println(" C");
  delay(500);
```

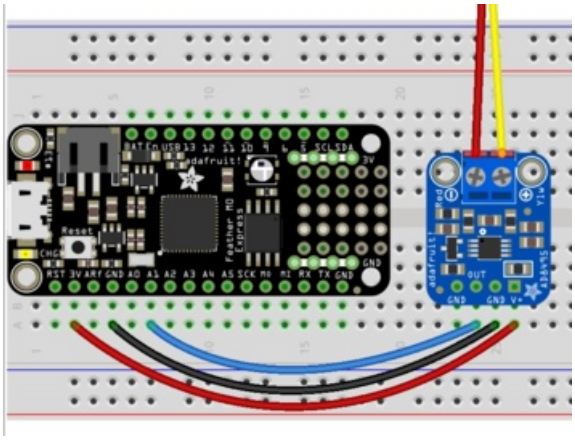
Python & CircuitPython

It's easy to use the AD8495 sensor with CircuitPython. This board does not require a separate library. The built in `analogio` module lets you easily read voltage from a thermocouple and use a simple equation to turn it into temperature.

Raspberry Pi does not have analog inputs, and therefore does not work with this breakout. Consider an I2C or SPI thermocouple amplifier if your project involves Raspberry Pi!

CircuitPython Microcontroller Wiring

First, wire up your AD8495. Here is an example of it wired up to a Feather M0:



- **Board V+** to **Feather 3V**
- **Board GND** to **Feather GND**
- **Board OUT** to **Feather A1**
- **Board RED-** to **thermocouple red- wire**
- **Board YLW+** to **thermocouple yellow+ wire**

CircuitPython Usage

To demonstrate use of this board, we'll initialise it and read the voltage and use the equation to turn it into temperature using the board's Python REPL.

First, run the following code to import the necessary modules and initialise the board on pin **A1**:

```
import board
import analogio
ad8495 = analogio.AnalogIn(board.A1)
```

Next, we'll need a helper function to take the raw data from the thermocouple and turn it into voltage.

```
def get_voltage(pin):
    return (pin.value * 3.3) / 65536
```

Next, we set `temperature` equal to the equation necessary to convert the voltage from the thermocouple into temperature, using the `get_voltage` helper.

The equation is `(voltage - 1.25) / 0.005`.

```
temperature = (get_voltage(ad8495) - 1.25) / 0.005
```

Now we can print the current temperature in C.

```
print(temperature)
```

```
>>> temperature = (get_voltage(ad8495) - 1.25) / 0.005
>>> print(temperature)
19.0917
```

That's all there is to using the AD8495 with CircuitPython!

Full Example Code

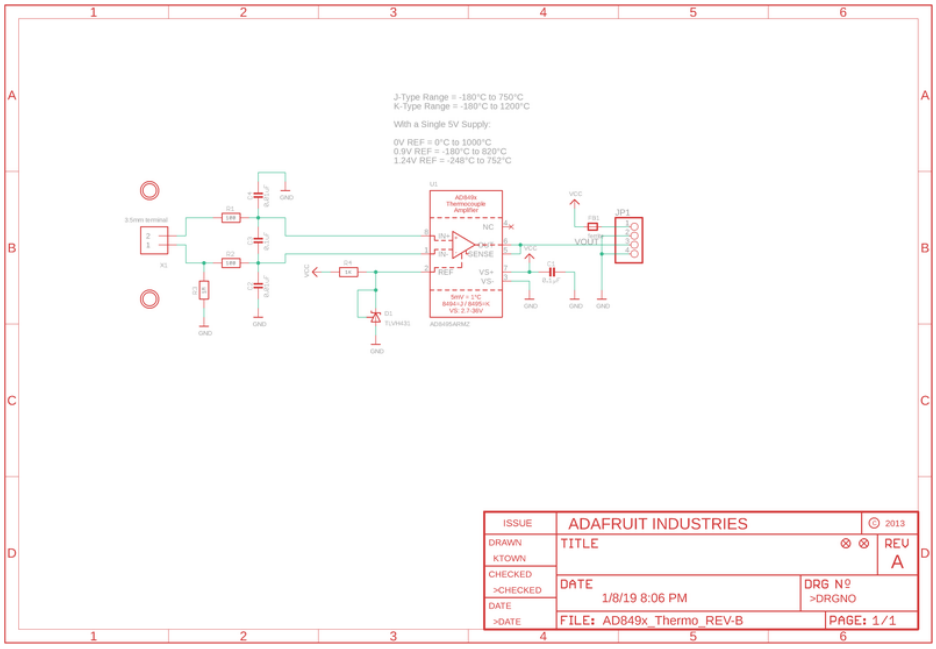
```
import time
import analogio
import board

ad8495 = analogio.AnalogIn(board.A1)


def get_voltage(pin):
    return (pin.value * 3.3) / 65536


while True:
    temperature = (get_voltage(ad8495) - 1.25) / 0.005
    print(temperature)
    print(get_voltage(ad8495))
    time.sleep(0.5)
```

Downloads

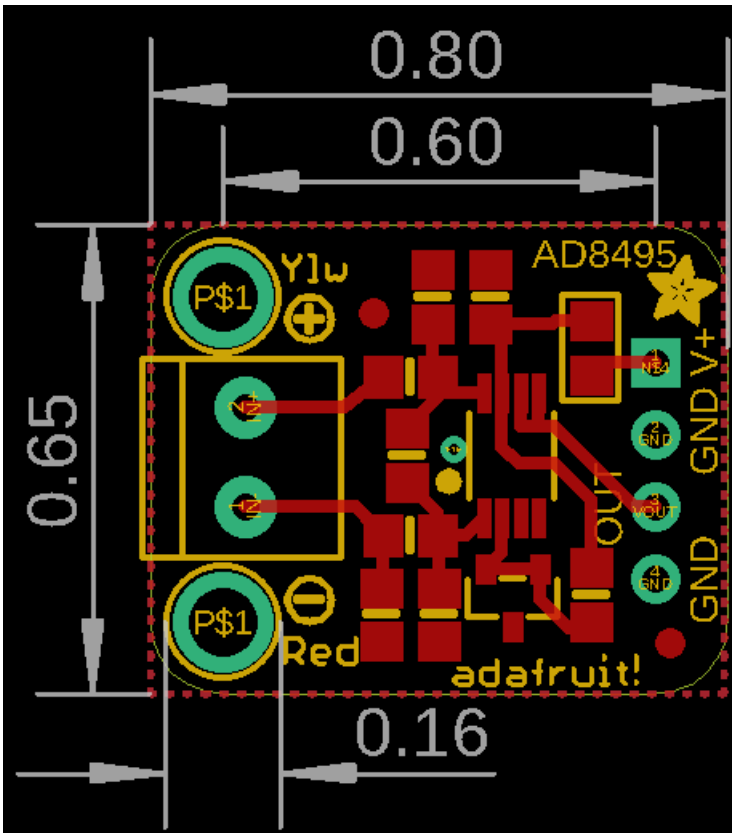
Files

- AD8495 Datasheet (https://adafru.it/EuI)
- EagleCAD PCB Files on GitHub (https://adafru.it/EuJ)
- Fritzing object in Adafruit Fritzing Library (https://adafru.it/EuK)

Schematic



Fab Print


0.80
0.60
0.65
0.16
AD8495
Ylw
Red
P$1
P$1
GND
V+
GND
OUT
adafruit!