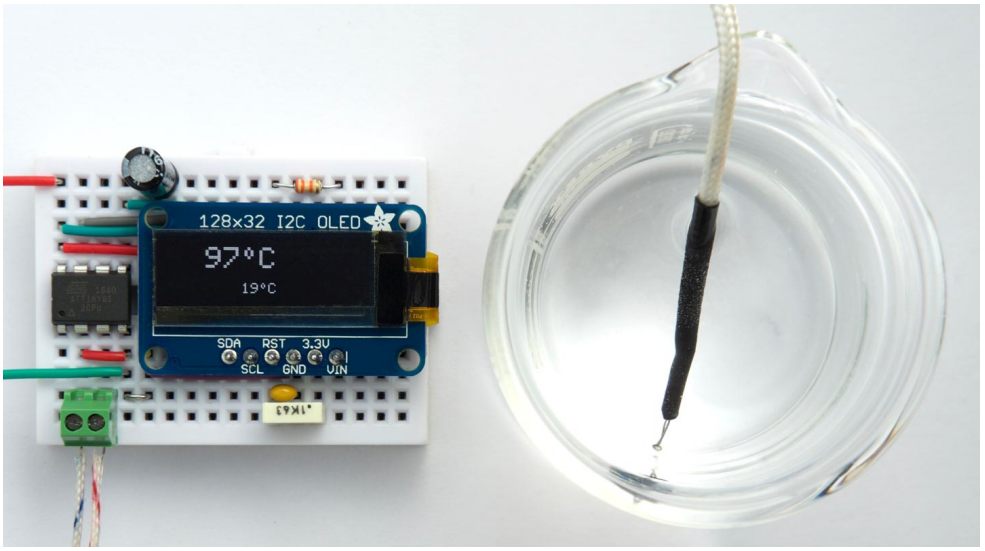
Tiny Thermocouple Thermometer

5th March 2019

This project describes a thermocouple thermometer, capable of measuring temperatures up to +1350°C, using just an ATtiny85 and an OLED display:



Thermocouple thermometer based on an ATtiny85 measuring boiling water.

It uses the ATtiny85's analogue-to-digital converter with a x20 gain option to measure the thermocouple voltage, and the internal temperature sensor to measure the ambient temperature (shown on the bottom line of the display), and it gives readings accurate to better than 5°C. Some possible applications include a cooking thermometer, a soldering-iron temperature monitor, or a wood burning stove temperature display.

Introduction

I've recently wanted to build a circuit capable of measuring temperatures around 250°C, and so the temperature sensors I've used in previous projects, such as the DS12B20, aren't suitable because they usually have an upper temperature limit of 150°C. The sensor of choice here is a thermocouple, which uses the fact that a junction between two different metals generates a small voltage proportional to temperature. The most common type of thermocouple, called the K type, uses chromel (a nickel-chromium alloy) and aluminel (a nickel/aluminium/manganese/silicon alloy).

One can use a custom thermocouple amplifier to translate the small voltage to a digital signal that you can read from a microcontroller; a popular choice is the MAX31855K which provides an SPI output [1].

However, it occurred to me that you could simply read the thermocouple directly using the analogue input on an ATtiny85. The ATtiny85 conveniently provides a 10-bit differential analogue input with a x20 gain option. A K-type thermocouple generates about 41μV/°C, so using the 1.1V voltage reference we can calculate the resolution and maximum range as follows:

- Resolution = $1.1/(1024 \times 20 \times 41 \times 10^{-6}) = 1.31$°C
- Maximum = 1024 x 1.31 = 1341°C.

With oversampling one could get the resolution below 1°C which is definitely enough precision for many applications.

The ATtiny85 is one of a small number of AVR microcontrollers that include a gain option on the ADC inputs; the only others I know of are the ATtiny861, ATtiny167, and ATmega1284P.

Although K-type thermocouples will measure temperatures down to -200°C, for simplicity in this project I have only supported temperatures above room temperature. I may extend this in a future project if there's any interest.

Getting the temperature

How do you get the temperature from a measurement of the thermoelectric voltage?

Search

Recent posts

▼ 2022
I2C SD-Card Module
Monochrome Low-Power Display Library
Three-Channel Chart Plotter
Adding File Storage to an Arduino
Universal TFT Display Backpack
Tiny TFT Graphics Library 2
On Bytes and Pins
Tiny I2C Routines for all AVR Microcontrollers
Minimal RP2040 Board
Printing to a Serial LED Display
16 LEDs Kishi Puzzle Solution
Twinkling Pendant
Morse Code Message Pendant
Controlling RGB LED Strips with a Single Function
16 LEDs Solution and a New Puzzle
► 2021
► 2020
► 2019
► 2018
► 2017
► 2016
► 2015
► 2014

Topics

► Games
► Sound & Music
► Watches & Clocks
► GPS
► Power Supplies
► Computers
► Graphics
► Thermometers
► Wearables
► Test Equipment
► Tutorials
► PCB-Based Projects

By processor

AVR ATtiny

► ATtiny10
► ATtiny2313
► ATtiny84
► ATtiny841
► ATtiny85
► ATtiny861
► ATtiny88

AVR ATmega

The simplest way is to assume that the relationship between voltage and temperature is linear at 41µV/°C. For small temperature ranges this is a good approximation; however, over larger ranges there is a significant deviation from linear behaviour [2].

For maximum accuracy you can solve a ninth-order polynomial, using coefficients provided by the National Institute of Standards and Technology (NIST) [3]. However, this approach isn't necessary for the sort of accuracy I was aiming for.

The approach I used is a piece-wise linear model, approximating the standard response curve with a series of straight line segments. I looked up the temperatures for a series of fixed points, corresponding to ADC readings that are multiples of 128, using an online thermocouple temperature calculator [4]:

ADC value	Voltage (mV)	Temperature (°C)
0	0	0
128	6.875	168.36
256	13.750	337.02
384	20.625	499.55
512	27.500	661.26
640	34.375	826.91
768	41.250	999.34
896	48.125	1180.53
1024	55.000	1375.09

These were then coded as the following array of fixed points, in tenths of a degree, for linear interpolation of the temperature:

```
int Temperature[9] = { 0, 1684, 3370, 4995, 6613, 8269, 9993, 11805, 13751 };
```

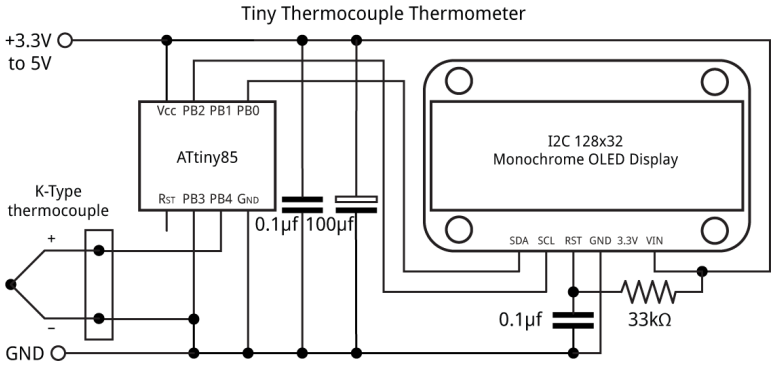
The cold junction

The voltage generated by a thermocouple doesn't give you the absolute temperature, but the difference between the temperature of the probe and the temperature of the connection between the thermocouple and your circuit. This is traditionally referred to as the "cold junction" (even though if you're measuring subzero temperatures it will actually be warmer than the probe). All thermocouple amplifiers therefore incorporate a sensor to measure their internal temperature.

Fortunately the ATtiny85 includes an internal temperature sensor, so the thermocouple thermometer uses this to calculate the cold junction temperature.

The circuit

Here's the circuit:



Circuit of the thermocouple thermometer, based on an ATtiny85.

For the display I chose the I2C 128x32 OLED display available from Adafruit [5] or Pimoroni in the UK [6]. The 33kΩ resistor and 0.1µF capacitor ensure that the display is reset correctly when power is first applied, although you may not need them. Alternatively you could use an equivalent display from AliExpress [7], or an I2C 64x32 OLED display.

I used a K-Type thermocouple bought from eBay [8], but a wide range of probes is available from different suppliers.

► ATmega328
► ATmega1284

AVR 0-series and 1-series

► ATtiny1614
► ATtiny3216
► ATtiny402
► ATtiny404
► ATtiny414
► ATmega4809

AVR DA/DB-series

► AVR128DA28
► AVR128DA48
► AVR128DB28

ARM

► ATSAMD21
► RP2040

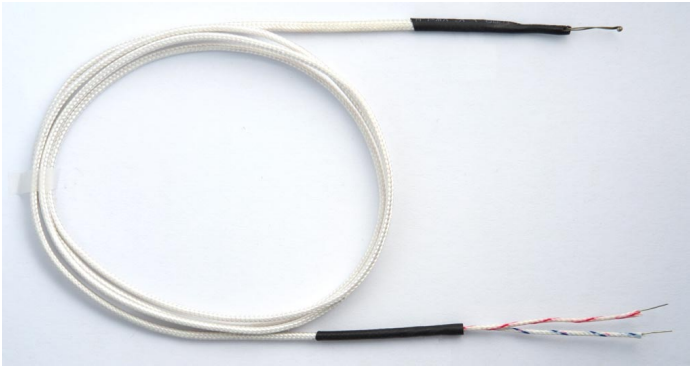
About me

About me
Contact me

Follow @technoblogy

Feeds

RSS feed



A typical K-Type thermocouple probe.

The program

For both the internal temperature measurement and the differential voltage measurement I do the analogue-to-digital conversions in sleep mode, as recommended by the datasheet to minimise noise from the processor and peripherals.

Driving the display

The display code is similar to the code I used in earlier projects that used the same 128x32 I2C OLED display, such as Tiny Machine-Code Monitor. In this project I only needed character-set definitions for the digits 0 to 9, and a few extra characters such as "-", "C", and a degree symbol.

The program allows you to plot double-sized characters; for details of how this works see Big Text for Little Display.

Measuring the internal temperature

The routine to set up the ADC for the internal temperature measurement is:

```
void SetupInternal () {
  ADMUX = 0<<REFS2 | 2<<REFS0 | 15<<MUX0;  // Temperature and 1.1V reference
  ADCSRA = 1<<ADEN | 1<<ADIE | 4<<ADPS0;   // Enable ADC, interrupt, 62.5kHz clock
  ADCSRB = 0<<ADTS0;                       // Free running
  set_sleep_mode(SLEEP_MODE_ADC);
}
```

The **ReadInternal()** routine takes a reading; it simply puts the processor to sleep. The ADC generates an interrupt to wake up the processor when the conversion is finished, and the routine then reads and returns the ADC register value:

```
unsigned int ReadInternal () {
  sleep_enable();
  sleep_cpu();                             // Start in ADC noise reduction mode
  return ADC;
}
```

Measuring the thermocouple voltage

A similar routine sets up the ADC for the thermocouple measurement:

```
void SetupThermocouple () {
  ADMUX = 0<<REFS2 | 2<<REFS0 | 7<<MUX0;   // PB4 (ADC2) +, PB3 (ADC3) was 7
  ADCSRA = 1<<ADEN | 1<<ADIE | 4<<ADPS0;   // Enable ADC, interrupt, 62.5kHz clock
  ADCSRB = 0<<ADTS0;                       // Free running
  set_sleep_mode(SLEEP_MODE_ADC);
}
```

The routine **ReadThermocouple()** actually takes the reading:

```
unsigned int ReadThermocouple () {
  sleep_enable();
  sleep_cpu();                             // Start in ADC noise reduction mode
  return ADC;
}
```

Converting the thermoelectric voltage to a temperature

The ADC reading from `ReadThermocouple()` is converted to a temperature using piece-wise linear interpolation, as described in `Getting the temperature` above. For greater accuracy I used the sum of four successive ADC readings; the routine `Convert()` takes this value and uses the values in the array `Temperature[]` to convert it to a temperature, in degrees:

```
int Temperature[9] = { 0, 1684, 3370, 4995, 6613, 8269, 9993, 11805, 13751 };

int Convert (int adc) {
  int n = adc>>9;
  unsigned int difference = Temperature[n+1] - Temperature[n];
  unsigned int proportion = adc - (n<<9);
  unsigned int extra = ((unsigned long)proportion * difference)>>9;
  return (Temperature[n] + extra + 5)/10;
}
```

If the parameter to `Convert()` is an exact multiple `n` of 512 the temperature is `Temperature[n]`/10. Otherwise the temperature is interpolated between `Temperature[n]` and `Temperature[n+1]`.

Displaying the readings

Finally, the main loop in `loop()` reads the internal temperature and thermocouple temperature every second, and displays the readings on the display. For each measurement I take the average of 16 successive readings, to reduce the noise:

```
  SetupThermocouple();
  ReadThermocouple();                     // Throw away first reading
  int reading = 0;
  for (int i=0; i<16; i++) reading = reading + ReadThermocouple();
  reading = Convert(max((reading>>2) + ADCOffset*4, 0));
  Scale = 2;
  PlotTemperature(reading + internal, 0, 0);
```

Improving accuracy

There are two factors that influence the accuracy of the thermometer, and you can calibrate each of these to improve the accuracy.

Internal temperature sensor

If the internal sensor is incorrect, all readings will be offset by this error. It's therefore a good idea to calibrate the sensor at room temperature against a known thermometer as follows:

- Check the internal temperature reading on the bottom line of the display.
- Set the constant `InternalOffset` in the program to the required adjustment to make this equal to the measured ambient temperature.

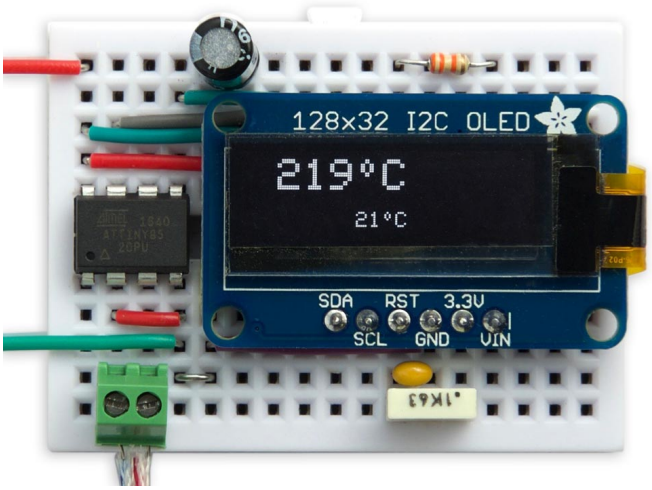
ADC offset

With no input on the x20 differential input there is usually a small offset, which is added to every reading. To measure this:

- Remove the thermocouple, and put a wire link between the two differential inputs, PB3 and PB4, to give a zero input.
- Note the difference between the thermocouple reading and the internal temperature reading.
- Set the constant `ADCOffset` in the program to the required correction to make this difference zero.

Testing

Note that before using the thermocouple thermometer in a critical application you should test it against known temperatures. I tested the thermometer against a commercial thermocouple thermometer with boiling water at 100°C, and with olive oil at 220°C, and the readings were within 5°C in each case.



Thermocouple thermometer based on an ATtiny85.

To measure the temperature of a solid object you can attach the thermocouple to the object using polyimide tape, available from SparkFun [9] or Proto-PIC in the UK [10].

Compiling the program

I compiled the program using Spence Konde's ATTiny Core [11]. Choose the **ATtiny25/45/85** option under the **ATTinyCore** heading on the **Board** menu. Then check that the subsequent options are set as follows (ignore any other options):

Chip: "ATtiny85"
Clock: "1 MHz (internal)"
B.O.D: "B.O.D. Disabled"
Timer 1 Clock: "CPU"

These are the default fuse settings for a new ATtiny85, but if you've previously used the ATtiny85 with different settings choose **Burn Bootloader** to set the fuses appropriately.

Then upload the program using ISP (in-system programming); I used Sparkfun's Tiny AVR Programmer Board; see ATtiny-Based Beginner's Kit.

Here's the Tiny Thermocouple Thermometer program: Tiny Thermocouple Thermometer Program.

Update

To display the temperatures in degrees Fahrenheit change the calls to **PlotTemperature()** in **loop()** to:

```
PlotTemperature(internal*9/5+32, 3, 6);
```

and

```
PlotTemperature((reading + internal)*9/5+32, 0, 0);
```

You'll probably also want to change the character definition of the "C" to an "F" in the **CharMap[][]** table:

```
{  0x7F, 0x09, 0x09, 0x09, 0x01, 0x00  }, // F
```

---

1. ^ Thermocouple amplifier MAX31855 breakout board on Adafruit.
2. ^ Calibrating Thermocouple Sensors on mstarlabs.com.
3. ^ Inverse coefficients for K type thermocouple on NIST.
4. ^ Thermocouple Voltage to Temperature Calculator on Fluke.
5. ^ Monochrome 128x32 I2C OLED graphic display on Adafruit.
6. ^ Adafruit Monochrome 128x32 OLED graphic display on Pimoroni.
7. ^ 0.91 inch 128x32 I2C IIC Serial OLED LCD Display Module on AliExpress.
8. ^ 1M K-Type Multimeter Temperature Thermocouple Probe on eBay.
9. ^ High Temperature Adhesive Tape on SparkFun.
10. ^ High Temperature Adhesive Tape on Proto-PIC.
11. ^ ATTinyCore on GitHub.

Next: Minimal ATSAMD21 Computer

Previous: Twelve PWM Outputs from an ATtiny85

Oleg • 2 years ago
А можно видео увидеть всего этого чуда
Reply • Share ›

Mileu • 3 years ago
Hello,
Can you please tell me what needs to be modified for a 128x64 OLED display (SSD1306 controller) ?
Also,how can the position of the reading can be change on the display ?

Thank you.
Reply • Share ›

Fred • 3 years ago
Hi. Nice design. What does the code need to change to read in Fahrenheit? This would make a good sensor for charging AC units but all the manufactures specs are in F in the US. Thanks for sharing.
Reply • Share ›

johnsondavies Mod ➚ Fred • 3 years ago
I've added an update at the end to explain how to do this. I haven't tested it - let me know if it works!
Reply • Share ›

Fred ➚ johnsondavies • 3 years ago
Hi. Thank you for the quick reply. Not use to that on the net. Thanks.

Been working on a proto but cannot find this statement anywhere in the sketch:
{ 0x7F, 0x09, 0x09, 0x09, 0x01, 0x00 }

Found:
{ 0x3E, 0x41, 0x41, 0x41, 0x22, 0x00 }, // C

Should this whole line be changed to:
{ 0x7F, 0x09, 0x09, 0x09, 0x01, 0x00 }, // F

The way I understood your instructions was to find the:
{ 0x3E, 0x41, 0x41, 0x41, 0x22, 0x00 }, // C and just change the C to an F.

Thanks. Really appreciate your help.
Reply • Share ›

johnsondavies Mod ➚ Fred • 3 years ago
Yes, change the whole line:

```
{ 0x3E, 0x41, 0x41, 0x41, 0x22, 0x00 }, // C
```

to:

```
{ 0x7F, 0x09, 0x09, 0x09, 0x01, 0x00 }, // F
```

Reply • Share ›

Fred ➚ johnsondavies • 3 years ago
Thanks so much again for your gracious help.

Made the changes including the I2C address unique to my display but nothing on the screen. I do not have the adafruit part but have this one:
https://www.ebay.com/itm/Di....
Is that my issue since it is 128X64?

Also there are no mV readings on the thermocouple leads but voltage and continuity to all the other connections.

Thank you.

Reply • Share ›

johnsondavies Mod ➚ Fred • 3 years ago

Your display should work fine. Try changing setup() and loop() to:

```
void setup() {
  Wire.begin();
  InitDisplay();
}

void loop () {
}
```

You should get a random dot pattern from the display memory's initial contents.

Reply • Share ›

Fred ➚ johnsondavies • 3 years ago • edited

Thanks so much again.
It works now! Only one issue left. Reads 87F in large text at top of display and 31F at bottom in small text. I assume this is supposed to be "C" on the small text. How do I change that?

Did not have to make the last changes you sent. The problem was the I2C address. The address printed on the back of the display was incorrect. I used the below sketch on an Uno to query the actual I2C address. Once I had the correct address it works great other than the small temperature which prints "F" instead of "C".

I'm going to but this in a nice case and use it to recharge AC units. Pressure measurements are critical but only at a know temperature.

Thank very much for your numerous help. Once complete I can send you some photo's of the finished product and the eagle board files if you are interested.

Cheers

Reply • Share ›

johnsondavies Mod ➚ Fred • 3 years ago

Sorry, I forgot about the internal temperature. I've modified the update to include that. Look forward to the photos!

Reply • Share ›

Fred ➚ johnsondavies • 3 years ago

Hi. Thanks again for the help.

Here's a new interesting twist. Have everything working in F but temperature below room temperature does not change. In ice the temperature stays the same as room temperature. It does read an increase in temperature above room temperature. Trying to isolate the issue I used 3 different thermocouples, 2 different attinys, and ran your Celsius sketch unedited. Same results irrespective of the test case. Temperature does not change below room temperature but does rise with an increase in room temperature. Any ideas appreciated.

Thank you.

Reply • Share ›

johnsondavies Mod ➚ Fred • 3 years ago

Sorry, the project doesn't currently support temperatures below room temperature. I think I mentioned that in the article.

Reply • Share ›

slazare • 3 years ago

Thank you to let me discover this differential input. I wonder if it is a feature specific to ATtiny85 or is it also available with ATmega328?

Reply • Share ›

johnsondavies Mod ➚ slazare • 3 years ago

It's also available in the ATtiny861, ATtiny167, and ATmega1284P, but not the ATmega328.

Reply • Share ›

slazare johnsondavies • 3 years ago

Many thanks!

Reply • Share ›

Phil-S • 3 years ago

Great stuff.
As you say, opens up quite a few possibilities around the home with heating monitoring and control.
I like the soldering iron idea and could give a new lease of life and better use of my old Weller TP soldering iron. All I know at the moment is the solder does or doesn't melt.

Reply • Share ›

johnsondavies Mod Phil-S • 3 years ago

Thanks! Let me know if you find a good application.