20 January 2019

Temperature Sensor

The Kinetis Cortex-M processor on the Teensy 3 boards has a built-in temperature sensor. However, it does not return temperature directly, but a voltage that must be converted to a temperature. This library provides functions to do that conversion.

Note: the library supports both Celsius and Fahrenheit temperatures. In this document, if not specified, temperatures are in Fahrenheit.

Based on my five processor samples (1 of each Teensy 3 type), the uncalibrated temperature sensor has an accuracy of about +/-4ºF (2ºC) at room temperature. The 3.6 specification is +/-6ºC at 25ºC. After calibration, I was able to get about 1ºF accuracy. Don't expect anything better than that from this sensor.

This temperature is the temperature of the chip itself. The chip runs hotter at faster speeds. See Figure 1 for a plot of this relationship for Teensy's running a nominal sketch. CPU-intensive sketches will run hotter.

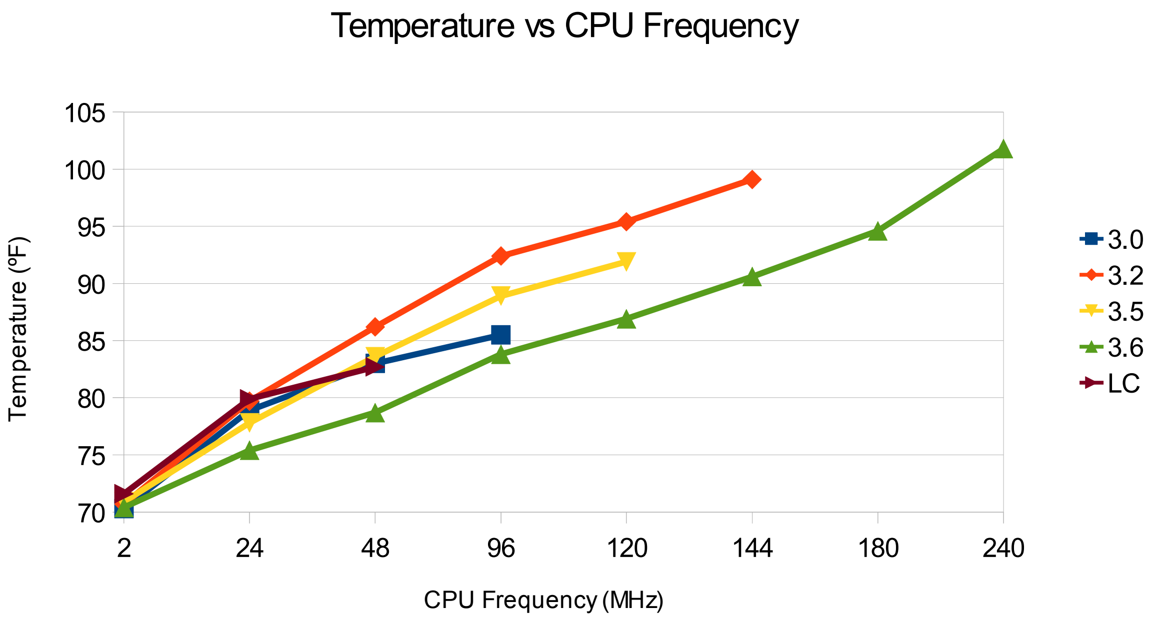

Figure 1: Temperature vs CPU Frequency

Library Usage

If you don't need calibration, using the library is as simple as:

```
#include <InternalTemperature.h>

// create instance of the class
InternalTemperature internalTemperature;
float temperature;

void Setup() {

  // initialize the library
  internalTemperature.begin();

  // read the temperature - C for Celsius, F for Fahrenheit
  temperature = internalTemperature.readTemperatureC();
}
```

By default, calling 'begin' will set up the Analog-to-Digital Converter (ADC) to have the maximum accuracy. If your application is also using the ADC for other readings, this may not be desirable. In this case, call begin (TEMPERATURE_NO_ADC_SETTING_CHANGES). This will not change any ADC settings, but temperature readings may be less accurate.

Examples

The library contains the following example programs:

simpleCelsius - periodically prints the current temperature to the USB serial port

simpleFahrenheit - periodically prints the current temperature to the USB serial port

tftThermometer - periodically displays the temperature to a TFT display using the ILI9341_t3 library. Also uses the Snooze library to run in low power mode to get room temperature.

calibration_single_point - shows how to input and use calibration data

Calibration

Uncalibrated temperatures are fine for comparison purposes. But for measuring absolute temperatures, calibration is recommended. There are 2 calibration modes supported in the library - single point and dual point. A single point calibration is good for a range of about +/-10ºC around the calibration point. A dual point calibration covers the entire temperature range between the points plus a good range beyond each point.

Note: For best temperature measurements, a dedicated temperature chip should be used. In these examples, my truth data was an MPC9808, which has an accuracy of 0.25ºC typical, 0.5ºC max. I also used the MPL3115A2 (1ºC accuracy at 25ºC) on the prop shield for comparison.

I wanted to measure air temperature, so I calibrated each Teensy in 2 MHz low power mode.

Theory

The voltage measurement decreases as the temperature increases. It is a linear function and is modeled as a straight line with a slope and intercept. The intercept point is arbitrarily chosen as 25ºC. See Figure 2 for the nominal specified slope and intercept.

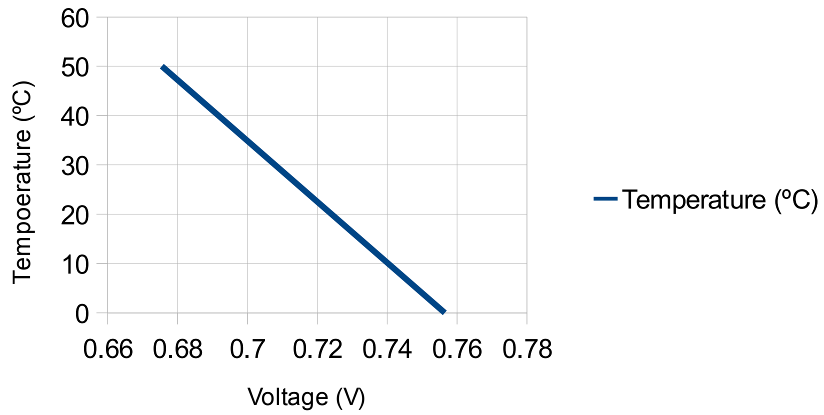

Figure 2: Voltage vs Temperature

Figure 3 shows some of my measured values versus the nominal value.

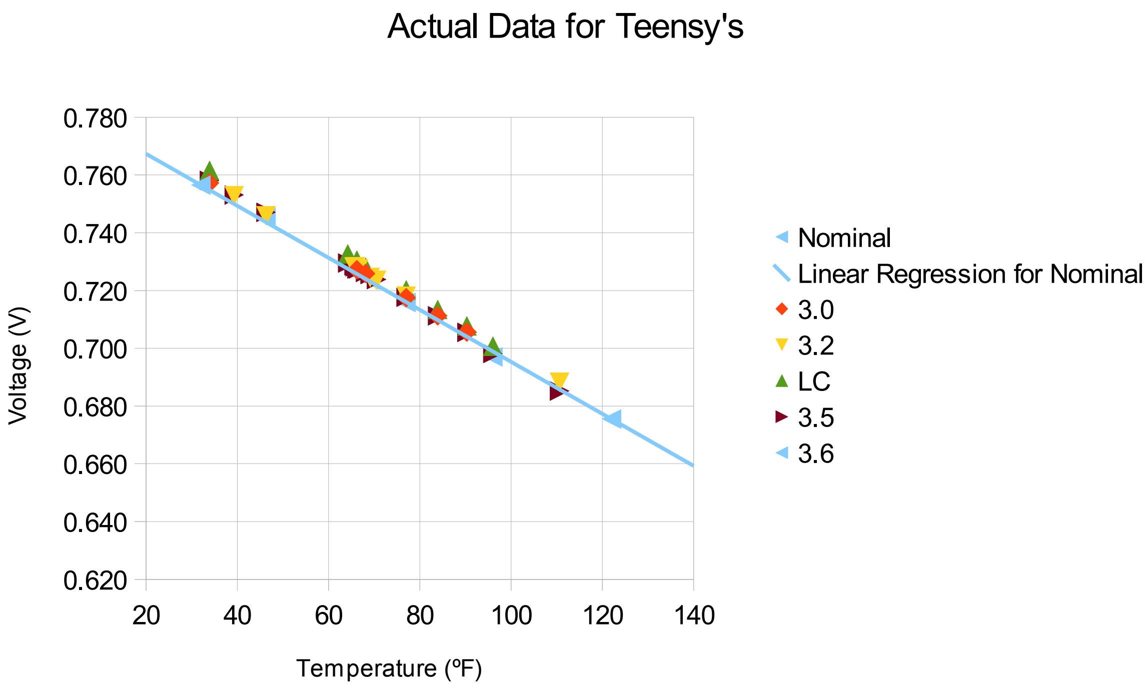

Figure 3: Temperature vs Voltage

Figure 4 shows how the error changes as the temperature changes. Each chip will have its unique error characteristics.

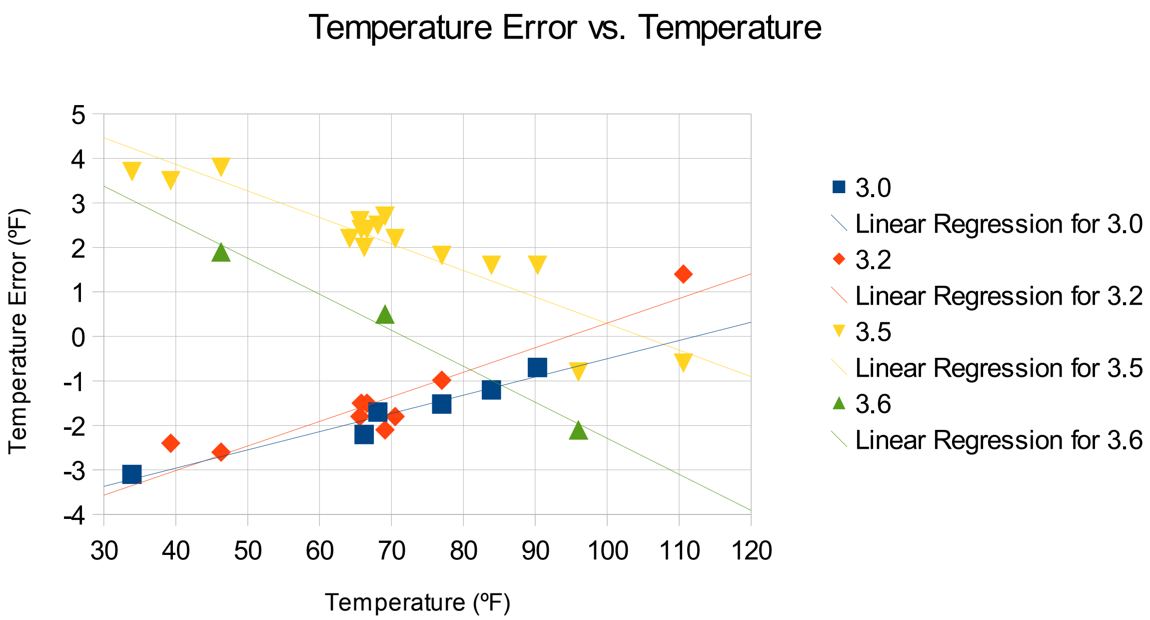

Figure 4: Uncalibrated Temperature Error

Figure 5 shows the temperature errors after performing a dual point calibration on each processor.

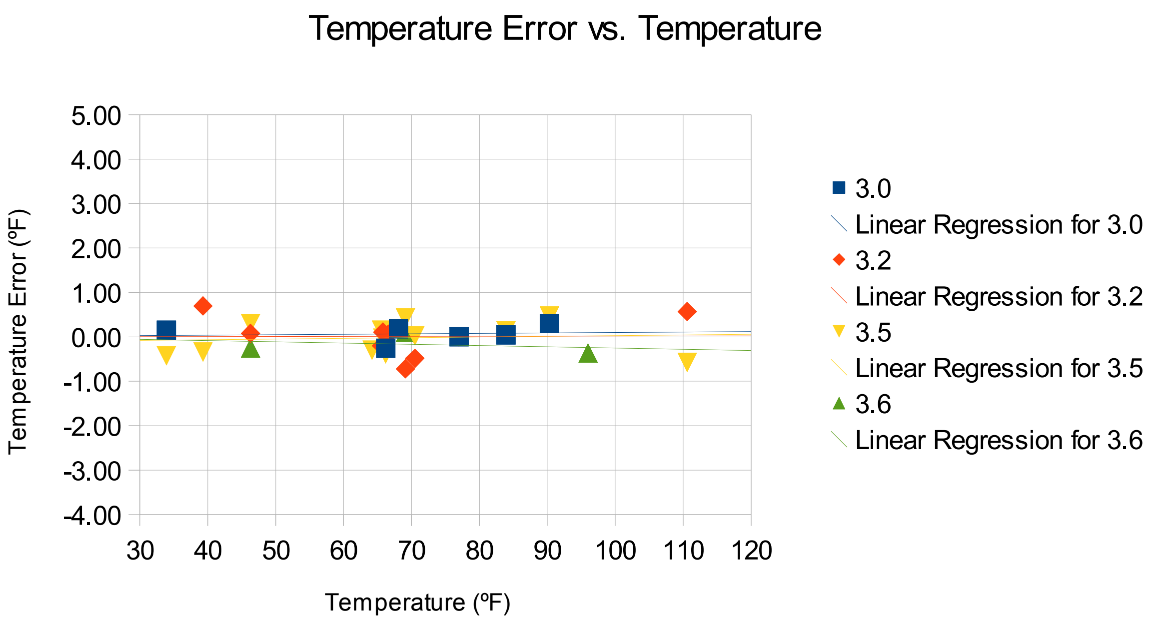

Figure 5: Temperature Error After Dual Point Calibration

Single Point Calibration

Single point calibration just provides a constant offset for each temperature. The temperature is measured at the desired actual temperature and the delta temperature is subtracted. In Figure 6, the error is the difference between the orange line and the blue line. To calibrate at 25 degrees, e.g. you read a temperature of 29 degrees when the actual is 25 degrees, so the library will subtract 4 degrees from each measurement. This moves the conversion from the orange line to the yellow line. This reduces the error (now the difference between yellow and blue line) at the calibration point but the error increases as the temperature moves away from that point.

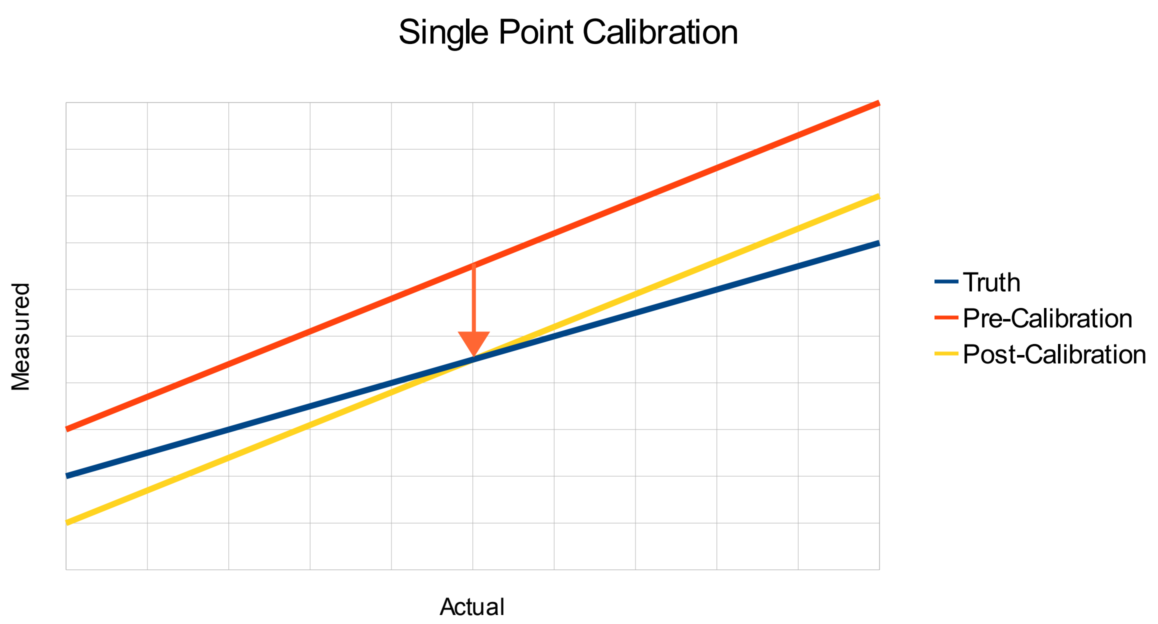

Figure 6: Single Point Calibration

Dual Point Calibration

Dual point calibration adjusts both the offset and the slope of the conversions. Two temperature measurements are made, preferably at each end of the range of desired accuracy. In Figure 7, two measurements are made. This allows adjusting both the slope and offset of the conversion, and now the yellow line is right on top of the blue line. For example, measurement at 10ºC and 25ºC will probably provide good accuracy from 0ºC to 35ºC.

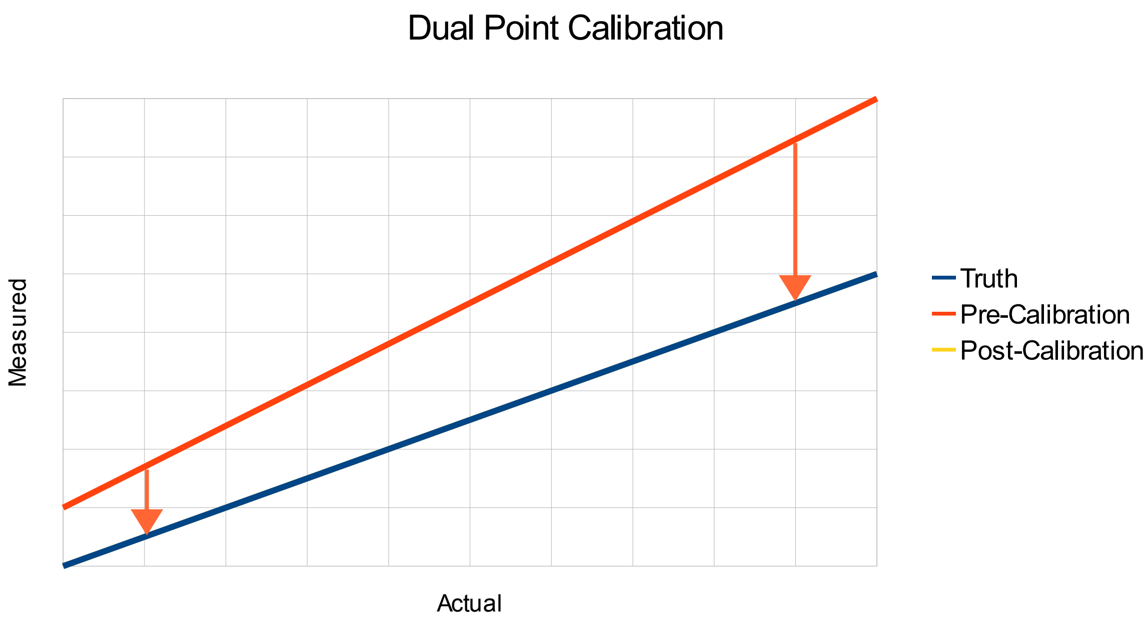

Figure 7: Dual Point Calibration

Calibration Code

See examples of calibration in the example code directories.