Technical Article

Step 2 to Build a Smart Thermostat Using an MCU– It's All about the Sensing

Britta Ruelander

Co-authored by Bhargavi Nisarga , Systems Engineer at TI

Hopefully you are all just waiting to get your hands on the smart thermostat project we outlined in our first blog post. Now, let's get started with the sensing part of the smart thermostat. This blog post will cover how to choose the right temperature sensor and how to use a microcontroller (MCU) to measure the actual temperature value.

Choosing a Temperature Sensor

First and foremost, you will need to pick your sensing element. It is fundamental to put some thought into what sensing and measurement chain you will need to make the right choice for your application.

There are different types of temperature sensing elements you can choose from. The most common types include RTDs (resistance temperature detectors), thermocouples, thermistors and integrated circuit (IC) sensors with digital and analog interfaces. The pros and cons of these temperature sensors based on various factors are comprehensively captured in this blog post. You can also check out the different types of sensors and TI products at TI.com/sensors.

Assuming the smart thermostat is mainly used in your home to monitor and regulate the ambient room temperature, the sensing element is needed to support a narrow temperature range. The resolution and accuracy needed is dependent on the changes in ambient room temperature you want the thermostat to track and what temperature accuracy is acceptable for your application, respectively. A good goal for a smart thermostat would be to support resolution of 1degC and accuracy of at least 0.6degC.

Keep in mind, the inherent sensor characteristics can drive additional circuitry or correction functions for temperature measurement. This in-turn can add to overall cost and complexity of the sensing and measurement chain. A sensor type that is less sensitive requires the signal chain to include operational amplifiers (op amps) and high resolution analog-to-digital converters (ADCs) to be able to measure the small changes at the sensor output. And, a sensor type that is not very linear (that is, a sensor output not proportional to the input over its complete range) requires linearization functions for correcting the non-linearity (e.g. look up table based linearization).

An important consideration that is often missed while selecting the components and putting together the sensing and measurement chain is to factor and accordingly budget the inaccuracies, power consumption and cost across the entire signal chain and not just the sensing element.

As a first approach on the smart thermostat application, you could opt for using either a **thermistor** or an **integrated circuit (IC) temperature sensor** as the sensing element. In either case, depending on the temperature range you're measuring and the sensor sensitivity, the resulting voltage change which needs to be measured can be quite small; thereby, there is a need for an op amp in the signal chain. The op amp will amplify the sensor output signal so that you will be able to make use of the ADCs full input range. This way, you can enhance the overall measurement resolution as more ADC bits are used to represent the measured value.

You can also take a look at the user's guide of the Thermostat Implementation with FRAM Microcontroller TI Design reference design to get an idea on how to get started with a thermistor-based sensing and measurement chain.

How to Select and Set the Sampling Frequency?

Sampling frequency refers to how often the analog input is sampled and converted to digital output for further processing. The sampling frequency should be selected based on how fast or slow the input parameter (in our case, the temperature sensor output) varies. In order to meaningfully reconstruct the continuous-time analog signal from the ADC digital output, the lower limit for the sampling frequency for a given input signal is derived from the Nyquist theorem where, f_sampling > 2f_max , and f_max refers to the highest frequency component of the analog input. And, the upper boundary for sampling frequency is limited by the time needed to sample and convert the input signal, and transfer the conversion results out of the ADC memory.

In our case, we know that the temperature changes are rather slow. So, it is sufficient to sample the sensor a few times a second – depending on how quickly you anticipate the temperature to change and how often you want your thermostat to track and regulate temperature changes.

Given that you know the sampling frequency, the most convenient way to set this up for the ADC is by using one of the general purpose timers already integrated in the MCU. For lower sampling frequency, which is applicable to our case, you can select the low frequency clocks in the MCU (e.g. 32kHz oscillator) as the interval timer clock source. Finally, all you will have to do is set the timer interval period and select this timer as the ADC trigger source. Figure 1 below shows a graphical representation of using an interval timer to trigger ADC sampling of temperature sensor analog output.

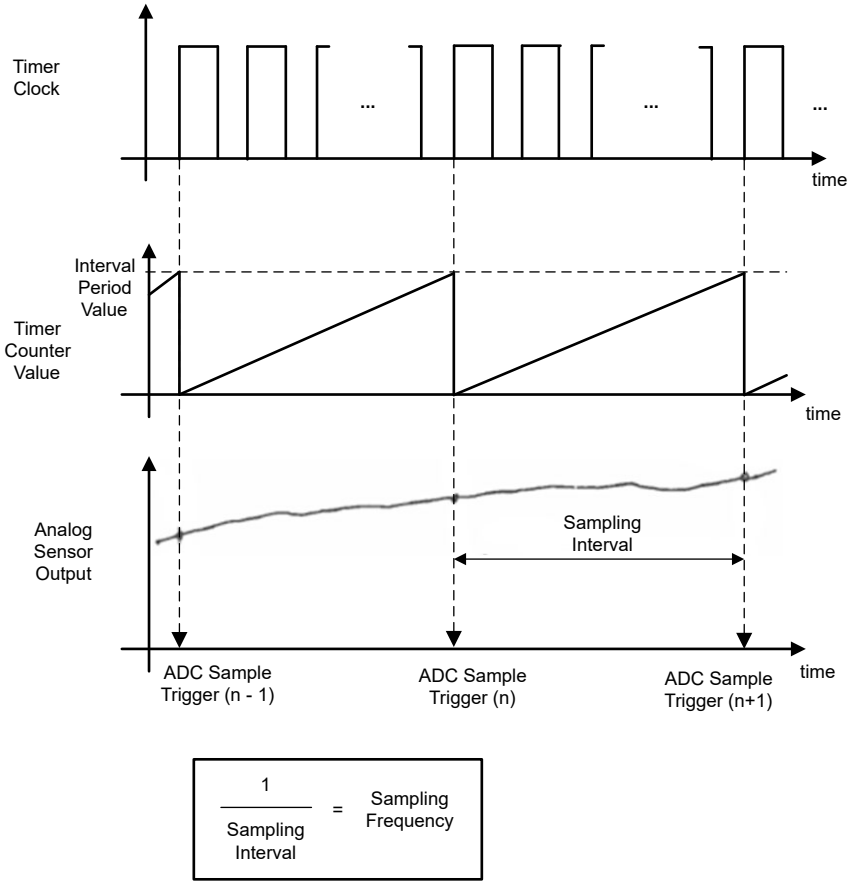

Figure 1. Using an interval time to trigger ADC sampling of temperature sensor output

How to Use the Integrated ADC to Get the Temperature Reading?

Before we proceed with the ADC settings and configuration, you should first do some math to check if the ADC has sufficient resolution to reliably measure the small changes in the sensor output. This two-part blog series "It's in the math: how to convert an ADC code to a voltage" will help you figure this out.

Next, to help you work through the different integrated ADC settings here are four questions we need to address:

1) How to select the ADC sample and hold time?

The time taken by an ADC input source to charge the internal capacitor of the ADC to a stable value in order to get accurate conversion results is referred to as the ADC sample time. The minimum ADC sampling time should be selected based on the output impedance of the sensor and the ADC sampling capacitor and input multiplexer (mux) resistance. You can find these values in the device-specific datasheets. The ADC sampling time is a configurable parameter (in clock cycles) and can be set by selecting appropriate ADC clock source and sample time settings.

With the internal sampling capacitor charged to the input source value at a given instant, the number of clock cycles required to convert this charge to a digital output is referred to as the hold time or the conversion time. This conversion time is typically a fixed number of ADC clock cycles.

2) How to select the ADC sample and hold time?

Integrated ADCs typically support multiple input channels through an input multiplexer. The ADC conversion mode enables you to select if one or multiple channels need to be sampled. The number of conversion channels you'll use and the sequence in which they should be sampled is application dependent. For the smart thermostat application with one input signal, you shall use the single channel mode. If it is required to measure multiple inputs in the application – for example measuring the temperature and the humidity – then, it is required to sample multiple ADC inputs in a sequence and therefore picking the sequence of channels conversion mode.

The ADC conversion mode also enables you to select if you want to trigger single conversion or repeated conversions of your ADC input channel(s). The number of conversions depends on the trigger and application. To build a thermostat you can select the single conversion mode to have each timer trigger initiate one temperature conversion.

3) How to determine the appropriate conversion trigger?

In general, the ADC can be either triggered by software (that is, by setting a register bit) or automatically by a timer. The previous section has discussed how to use the timer trigger and why it is a convenient option in the case of a smart thermostat that needs to periodically measure temperature.

4) How to transfer the conversion result for further processing?

The ADC setup requires allocating ADC memory for the conversion data. Depending on the ADC module offering, the ADC memory allows storage of few conversion results before it is picked by the system or restricted to storing just one conversion result. Take care that you always transfer the ADC conversion results before it is overwritten by new conversion results. We will be discussing more on this front in the next blog post.

In a nutshell, this is what you need to get started with the sensing part of the smart thermostat. But of course you'll find further configuration details in the MCU's user's guide and datasheet.

As a wrap-up, here are the key takeaways of our **second** blog post on smart thermostats:

- Understand the sensing and measurement chain.
- Make sure to pick a temperature sensor that best fits your application needs.
- Keep in mind to budget for the accuracy, power and cost over the entire sensing and measurement chain and not just the individual components in the chain.
- An integrated ADC is highly configurable. Make sure you understand and choose the right settings to make it work for your application.

Now, get started with your project! Check out our next post on data processing.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265