Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.115 Microprocessor Project Laboratory Spring 2020

Final Project Tip
Measuring Temperature with the PSoC IDAC

More on why the PSoC is my "desert island part"...

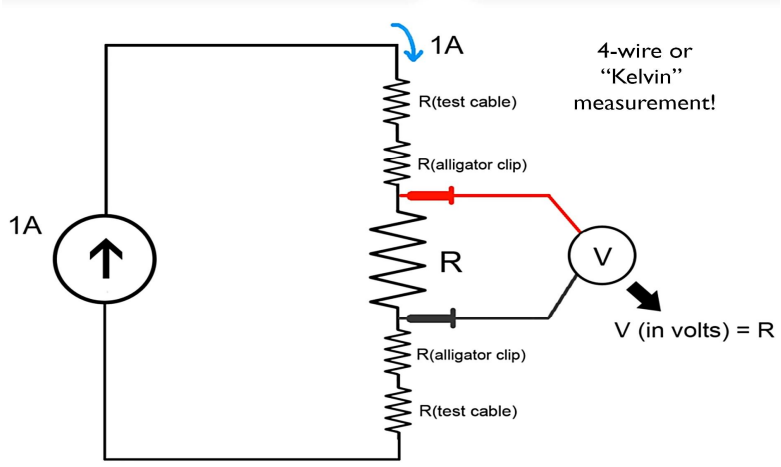
So you are not interested in measuring light levels.... but you do want to measure temperature! How can our "Reality PSoC" alter our desert island world to make this happen? Did you know that an LED (from your Kovid Kit) can be used as a temperature sensor?

As you know, if you put current IN to an LED in the right way, it glows, and also exhibits a "forward voltage" or forward drop. The forward voltage for an LED is typically higher than what we would expect from a "normal" signal diode. Signal diodes might have "textbook" forward voltages of 0.6 or 0.7 volts, while LED's might exhibit 1.7 volts or higher. Always check the specification sheets for correct numbers, not textbook generalizations. But, if you were to check with your multimeter, most of you would probably find that the glowing LED's on your Kovid Konsole exhibit a forward voltage of about 1.7 volts. As with all PN semiconductor junctions, the "forward voltage" is dependent on temperature! This is incredibly annoying when trying to use a diode as a voltage reference. Sometimes people would like the diode's sharp I-V curve to offer a "nearly fixed" voltage for a reference, e.g., in a DAC. Temperature variations in a diode may create unacceptable variations for precision applications, and there have been many patents trying to "fix" this behavior. But, this completely rocks when we actually **want** to use a diode or LED as a temperature sensor!

KEY information: The forward voltage typically changes about -2 millivolts per degree C.

That's right! The forward voltage actually gets lower as the diode gets hotter. You have to account for this if you use a diode as a temperature sensor, i.e., your measurement will get lower as the temperature gets higher. Old computer designs sometimes took advantage of this fact by putting the logic PCBs on top of the (warm) power supply. As the computer operated, it would warm the boards, lower the forward drops in the transistors, and lower the computer's overall power consumption. You didn't want to overwork the horse turning your generator....

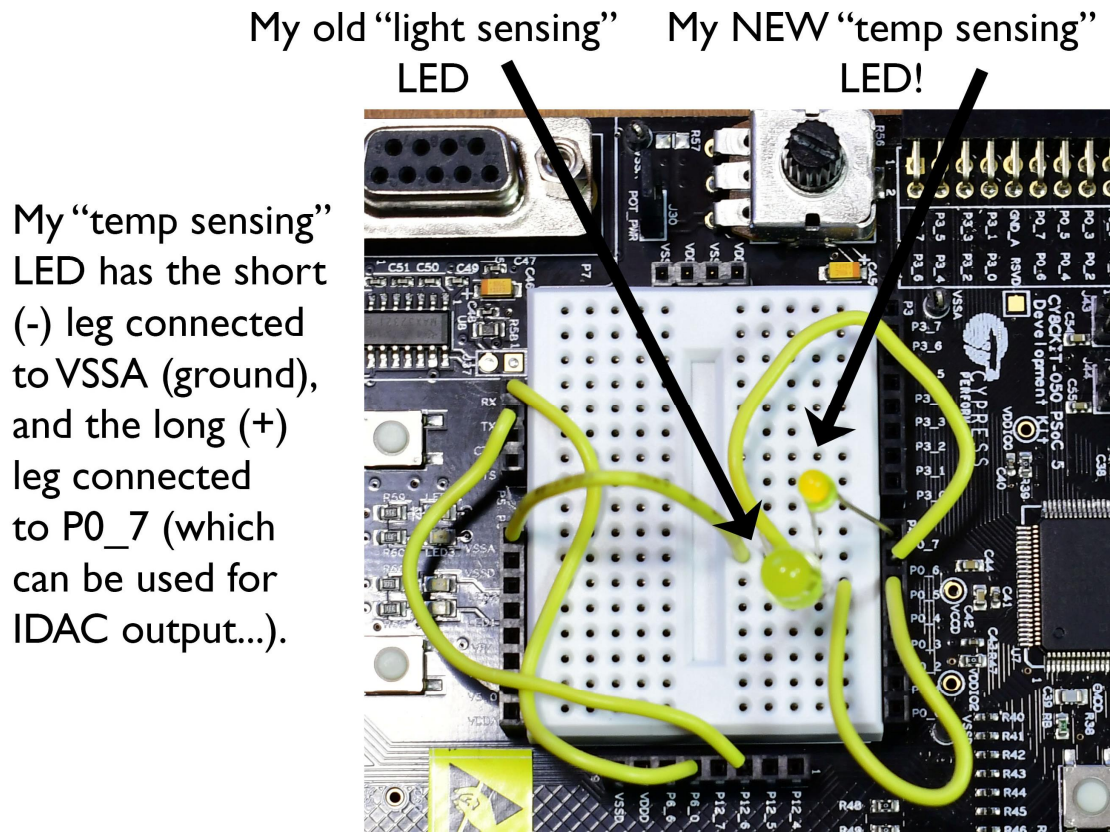
So we seek to measure fairly small changes in voltage (compared to other signals we have examined) when the LED is carrying current. How do we set up this measurement? We will use a (crude) version of a measurement technique developed by that irrepressible genius, Lord Kelvin (the temperature Dude):



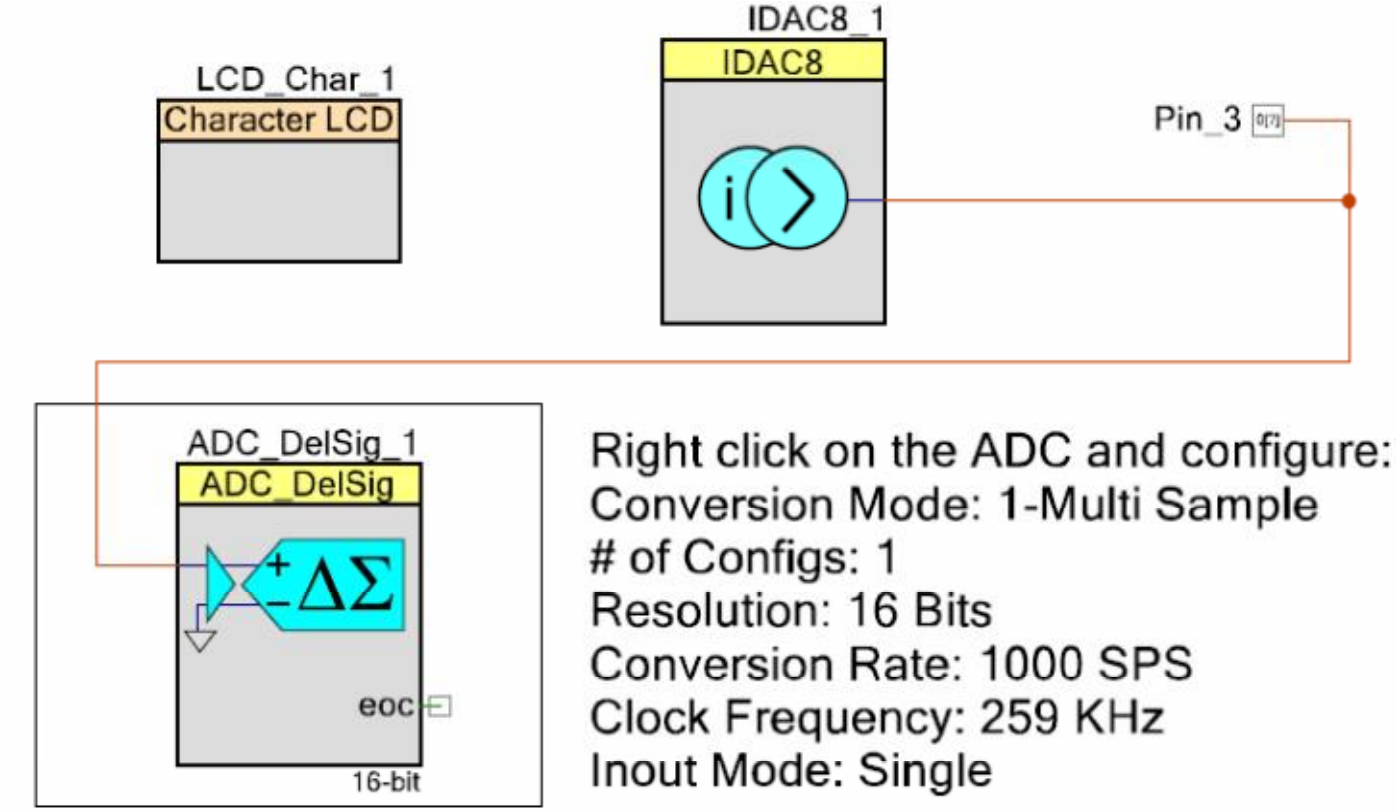
Lord Kelvin developed the "4-wire" technique for making measurements of very small resistances when he was working with strain gages, i.e., wires whose resistance changes when they undergo elastic deformation. I mention that because the techniques we're looking at in this note for temperature measurement also apply in part for strain gage measurements. Here's the idea: Suppose you want to measure the resistance R in the figure above. If R is relatively small, it might be difficult to measure without also including annoying errors from parasitics like the resistances in the test cables and clips. So, the measurement is conducted by two pairs of leads, hence the name "4-wire" technique. One pair of leads injects a test current, shown as one amp in the figure. This current flows through all the resistances, both the one of interest and also the parasitics. The measurement of voltage is made directly across the terminals of the device of interest, in this case the resistance R. The voltage meter ideally draws no current, so it gets a clean, crisp measurement of the voltage across the device under test (**DUT**). You can divide this voltage by the current you injected and you now know R. This technique works for surprisingly small values (milliOhms) compared to what you can measure with the resistance setting of your conventional multimeter! Pretty clever....

We will scam the 4-wire measurement idea because PSoC makes it easy! I'm being cavalier in my demonstration here, because I'm not worried about accuracy too much in my temperature measurement. If you were worried, you would want to carefully implement the full 4-wire measurement technique. Here's my "quick and dirty" version:

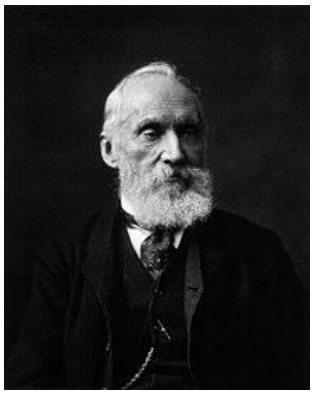
1. First, I added another LED from my Kovid Kit to my PSoC 5LP "Big Board". Again, be careful!



2. Next, I laid out my TopDesign sheet to include a mighty new Infinity Component from Creator, the IDAC! The IDAC is similar to a regular DAC you have already used, e.g., the AD558J. But, instead of creating an output voltage, the IDAC responds to the input data byte by creating a level of output current - just what Lord Kelvin ordered. We can use this to drive a well-controlled current into the LED. If you set the IDAC current high enough, you can actually make the LED glow, and control its brightness directly, no current-limiting series resistor needed. Here's my TopDesign sheet:



Note that Lord Kelvin is not impressed:



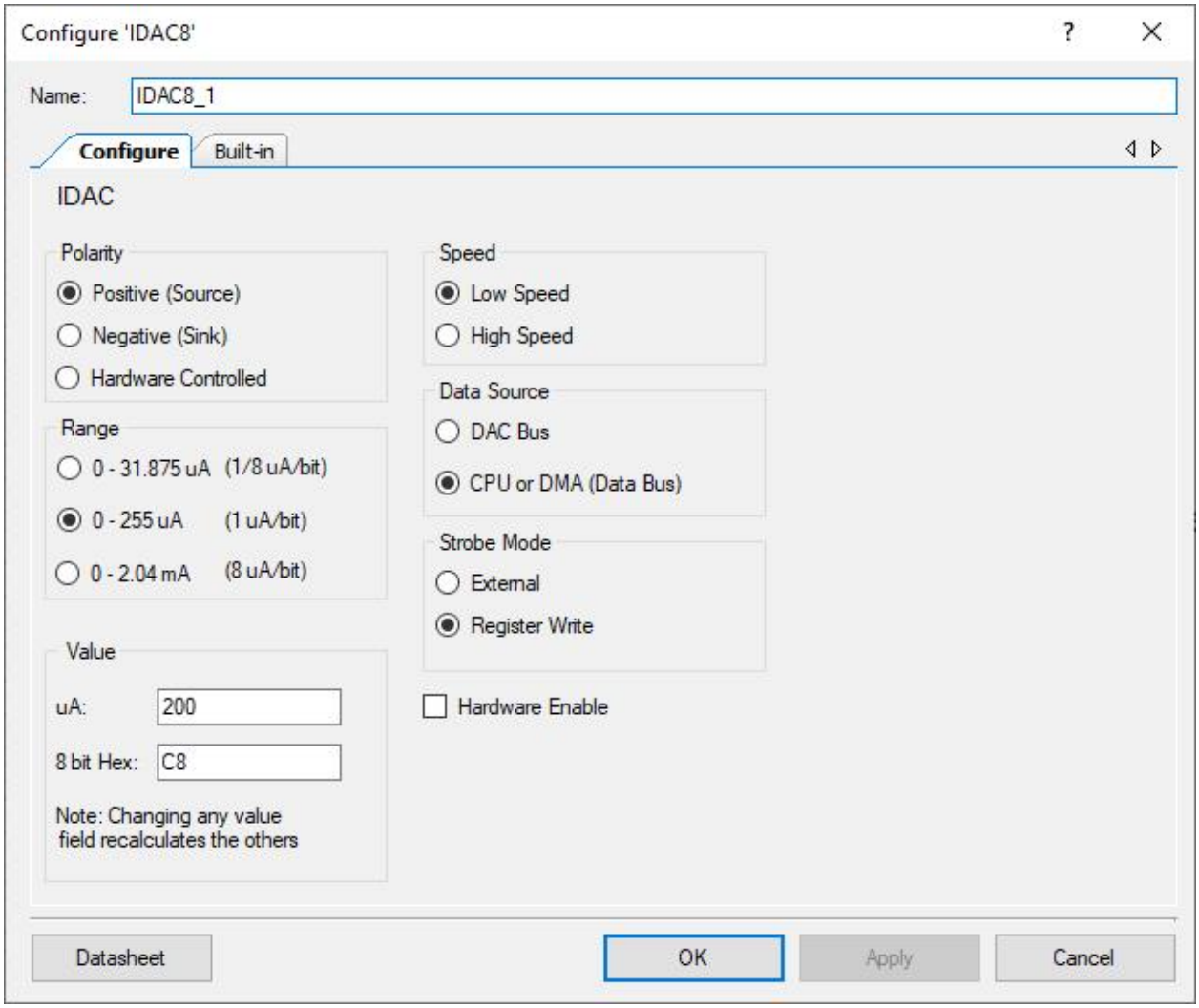
Your Instructor fails to impress. But PSoC is indeed astonishing....

Specifically, Lord Kelvin is disappointed that I failed to fully deploy his brilliant invention. If I use his idea correctly, I would have the IDAC drive my DUT (our LED in this case) through a pin. I would then commit two **additional** pins to make the voltage measurement with the ADC configured for a fully differential measurement across the LED directly. Instead, I have been lazy for expedience and internally connected a single-ended ADC measurement to the **same** PSoC pin used for the IDAC output. So I am picking up some small measurement error due to the impedances of the wires that will connect the LED to the PIN_3 connector and ground.

(Standards for Tenure have dropped over the last 100 years...)

But in my defense, we are on a desert island, and wire and pins are at a premium. Lord Kelvin's funding was a little better than mine....

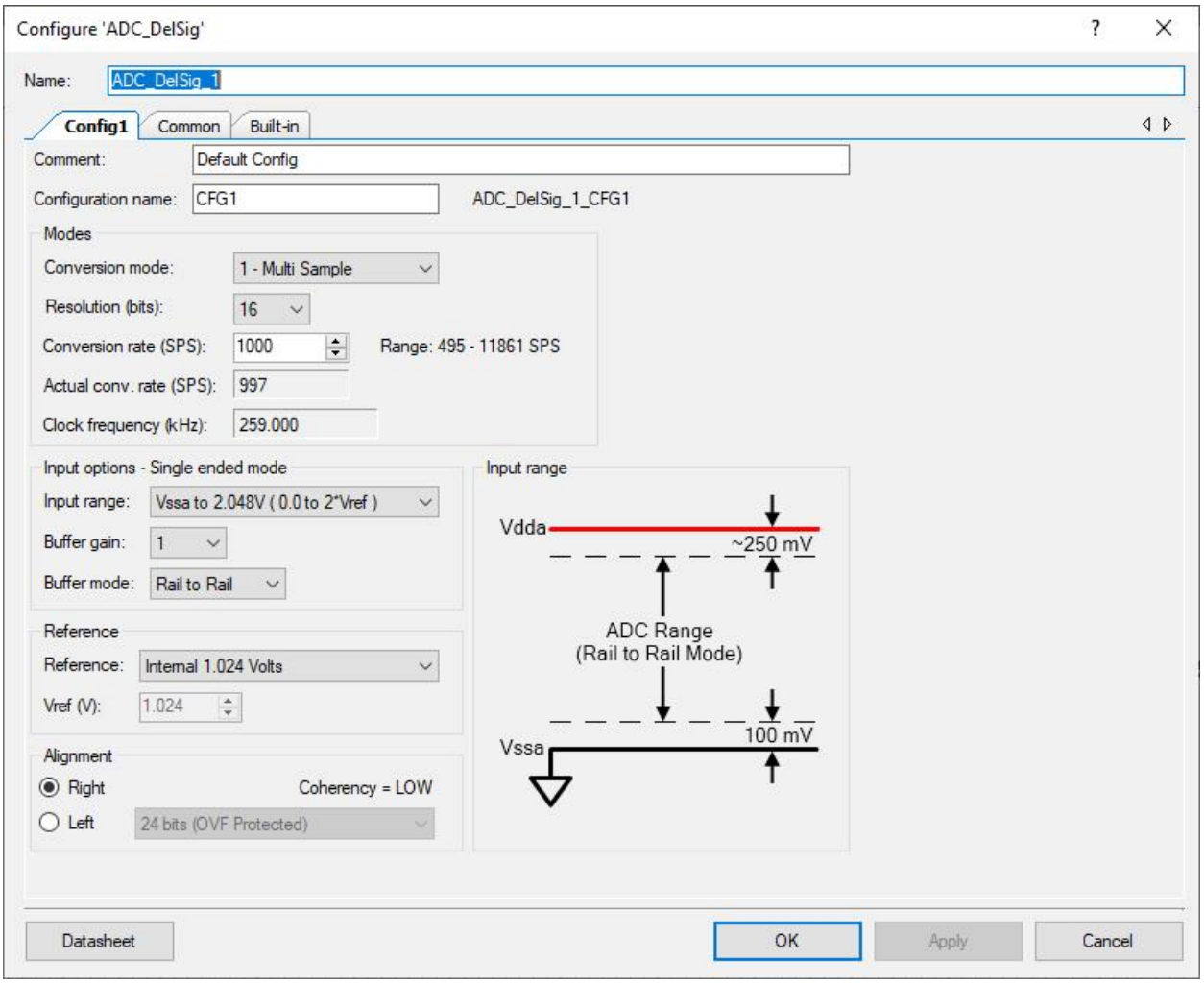
3. Ignoring my shameful laziness and the pauper status of my bunker, I configured the IDAC as shown below:



Note that you can choose the IDAC output current range, analogous to the way we could set the AD558 output voltage range by changing the configuration of its output op-amp. We alway get 8 bits of **resolution** with this particular IDAC component, but we can distribute the resolution over three different choices of **range**. If I chose the "biggest" range, which can deliver up to 2.04 mA of current, I could have used this IDAC to "light up" an LED from my Kovid Kit. My LEDs actually glow at this level.

For the temperature sensing application, I do **not** want the LED to glow. That would just add heat to the system that would interfere with my measurement. So, I chose a lower current range, with better resolution, and pre-set the IDAC output current to 200 microamps. There are API commands (read the datasheet for the IDAC) that will let you change the current level in the C code if you want. This fixed current will be fine for my demonstration.

4. Continuing on in my workmanly manner, I configured the ADC as shown below:

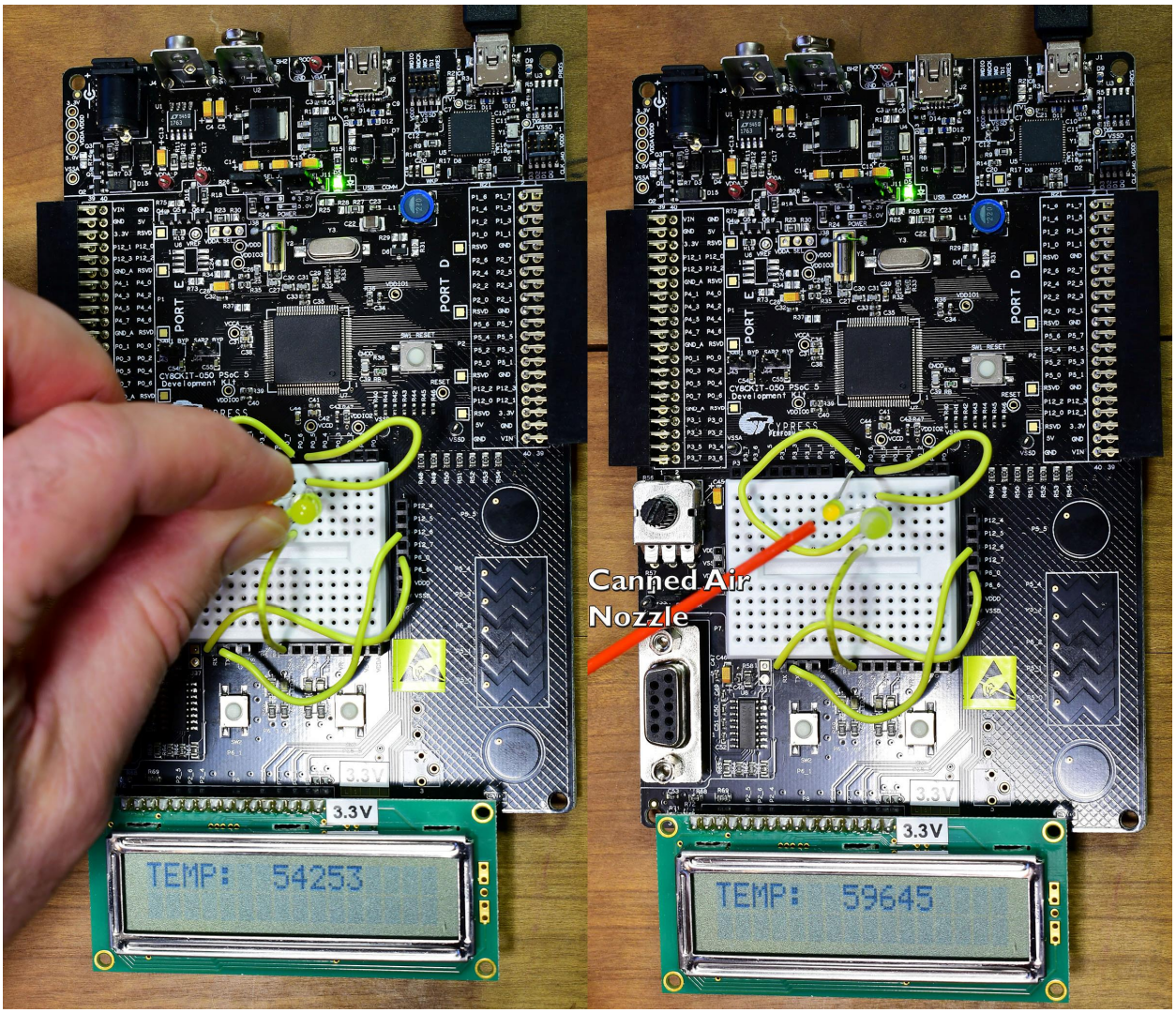


I selected a 16-bit resolution ADC with a range spanning 0 to 2.048 volts. In principle, this gives an ability to resolve voltage differences as small as 31 microvolts. It will not be that good in practice because of noise and my poor wiring. But that's still plenty fine for believably resolving temperatures changes of one degree Celsius (or Kelvin!) if the LED forward drop changes by 2 millivolts per degree.

5. Now, I just need C-code in my main.c file to fire up the whole system, read the LED voltage using the ADC, and display the 16-bit return value (between 0 and 65,535) on the LCD. My temperature display is "no frills," just a report of the 16-bit ADC integer value. Note, also, that the number will go down as the temperature goes up, because of the LED physics! A quality program would convert the answer to a proper temperature display on the LCD. My solution:

```
#include <device.h>

uint16 adcResult = 0;

void main()
{
      unsigned char j = 255;                    // milliseconds delay


      LCD_Char_1_Start();                              // initialize lcd
      LCD_Char_1_ClearDisplay();
      LCD_Char_1_PrintString("TEMP: ");

      ADC_DelSig_1_Start();          // start the ADC_DelSig_1
      ADC_DelSig_1_StartConvert();   // start the ADC_DelSig_1 conversion


    IDAC8_1_Start();    // Fire up the IDAC.

    for(;;)
    {
            if( ADC_DelSig_1_IsEndConversion(ADC_DelSig_1_WAIT_FOR_RESULT) )
            {
                  LCD_Char_1_Position(0, 7);
                  LCD_Char_1_PrintString("      "); // clean up the display
                  LCD_Char_1_Position(0, 7);
                  adcResult = ADC_DelSig_1_GetResult16(); // read the adc
                  LCD_Char_1_PrintNumber(adcResult);
                  CyDelay(j);              // create a 3/4 sec delay
                  CyDelay(j);   // Temperature doesn't change that fast
                  CyDelay(j);   // so this slow update is fine.

            }
    }
}
```

6. All that remains is to program the PSoC Big Board in Creator. After programming, I tested my board as shown below. The picture on the left shows the temperature readout when I warm up the LED by holding it between my fingers. (My Creator Gauntlet is removed in the picture for clarity.) The picture on the right shows the temperature readout when I spray "canned air" at the LED to cool it (you can see the air nozzle in the picture). Note that it makes sense that the "nominal" temperature integer report is around 55000 or so. With a 16-bit ADC and a range of 0 to 2.048 volts, a report near 55000 corresponds to a forward voltage around 1.7 volts.



Never turn around before you reach the top of the hill.