Lab 9 Temperature Data Acquisition System

This laboratory assignment accompanies the book, Embedded Systems: Real-Time Interfacing to ARM Cortex M Microcontrollers, ISBN-13: 978-1463590154, by Jonathan W. Valvano, copyright © 2017.

Goals
- Study ADC conversion and the Nyquist Theorem
- Develop a temperature measurement system using a thermistor,

Review
- Operation of the ADC system in the TM4C123 data sheet,
- Data sheets on the OPA2350 dual op amp IC, and INA122P instrumentation amp
- Valvano Chapters 8 and 10 on thermistors, analog amplifiers, analog low pass filters, and ADC's, data acquisition systems.

Starter files
- Design templates: **therm12.xls**, **FFT16.xls**, **lpf.xls Lab09_Artist.sch** at http://users.ece.utexas.edu/%7Evalvano/EE445L/labs.htm
- TI **FilterPro,** filterpro_design_program_download-h.exe (version 2) at http://users.ece.utexas.edu/%7Evalvano/EE445L/downloads.htm
- **PeriodicTimer0AInts_4C123, ADCT0ATrigger_4C123,** at http://users.ece.utexas.edu/~valvano/arm/
- **ST7735_4C123** LCD driver used in Labs 1, 2, and 3,

Requirements

This experiment will use an ADC converter on the TM4C123 to construct a digital thermometer. The temperature range is 10 to 40 ˚C. (You may adjust the temperature range to other values as you see fit). The measured temperature data will be displayed numerically as a fixed-point number on the LCD, using a 0.01 fixed-point format. The temperature versus time data will also be graphically plotted. Your temperature measurement resolution should be 0.1 ˚C or better. The average temperature accuracy should be 1 ˚C or better. The frequency components of the signal are 0 to 10 Hz.

Required Hardware

EK-TM4C123GXL www.ti.com	$12.99
Sitronix ST7735 Color LCD, http://www.adafruit.com/products/358	$19.99

Required Parts

Parts	www.Mouser.com	www.element14.com	www.digikey.com	Price
~50k thermistor	**594-NTCLE100E3503GB0**	**NTCLE100E3473JB0**	**BC2432-ND**	**$1.00**
LM4041CILPR shunt diode	**595-LM4041CILPR**	**LM4041CILPR**	**296-22173-1-ND**	**$1.00**
OP2350PA op amp	**595-OPA2350PA**	**OPA2350PA**	**OPA2350PA-ND**	**$5.18**
INA122PA, amp	**595-INA122PA**	**74K3813**	**INA122PA-ND**	**$6.50**

You will need some resistors and capacitors. An AD627AN or AD627ANZ can be substituted for the INA122. A MAX492CPA TLC2272CP or TLC2274CN can be substituted for the OP2350.

Approach and Constraints

Figure 9.1 shows the data-flow graph of the temperature data acquisition system. The thermistor converts temperature to resistance, the bridge converts resistance to a small voltage, the analog amp converts a small voltage into the 0 to +3.3V range, the analog filter removes high frequency noise (improves SNR), and the ADC converts analog voltage to a 12-bit integer. The periodic timer triggers the ADC, and the ISR executes when the ADC is complete. Because the timer starts the ADC, there is no sampling jitter. The system is real time as long as the software is fast enough to keep up (no lost data). Data are passed to the foreground using any appropriate data structure (such as a mailbox or FIFO, shown as **FIFO** in Figure 9.1). The main program converts integer to fixed-point data using a table lookup-interpolation scheme. The results are displayed on the color LCD.

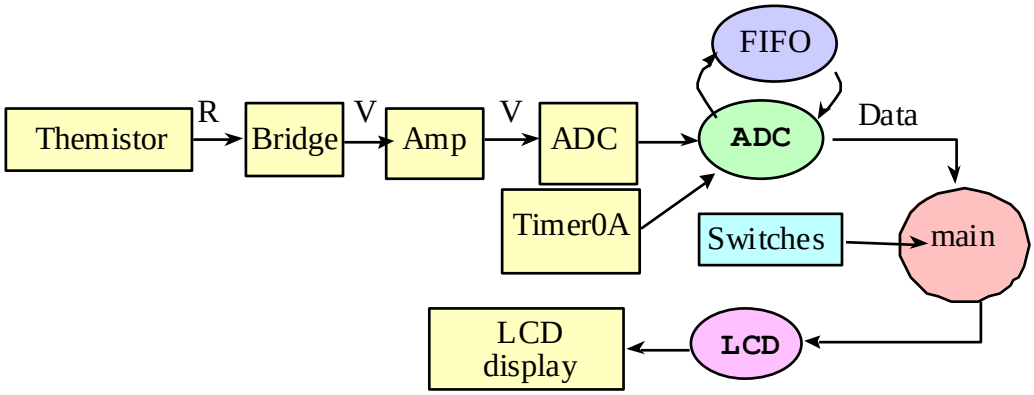

Figure 9.1. Data-flow graph of the data acquisition system (R means resistance, V means voltage).

Figure 9.2 shows one possible call-graph. Each of the modules (**LCD**, **ADC**, **FIFO**) has separate header and implementation files. Notice the main program does not directly access the **ADC, Timer** or **LCD** I/O port registers. The module **FIFO** can employ any appropriate method to pass data from background to foreground.

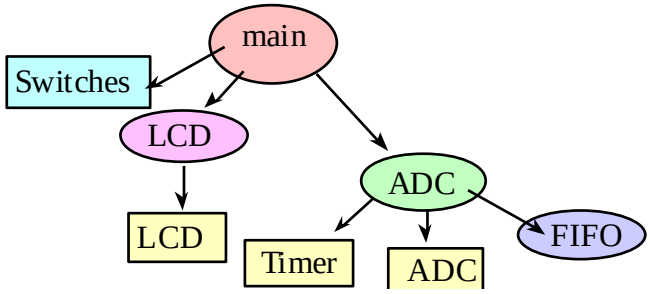

Figure 9.2. One possible call-graph of the data acquisition system.

This data acquisition system includes a thermistor, which converts temperature into a resistance. One possible way to convert resistance into voltage is to use a bridge. A second possibility is to use a constant current source. In either case, an analog amplifier boosts the voltage so the full-scale temperature range (e.g., 0 to 40 ˚C) maps into the full-scale range of the ADC (0 to 3.3 V). A low-pass analog filter, with a cutoff frequency of about ½ the sampling rate, removes high frequencies that might otherwise cause aliasing. A low-pass analog filter must be implemented in this lab. Choose the LPF cutoff frequency greater than or equal to 20 Hz, and less than ½ f_s.

1. Temperature-Resistance Calibration of the Thermistor:

The thermistor resistance varies nonlinearly with its temperature. It is very important to use temperature units of Kelvin in this equation and not ˚C.

$$R = R_0 e^{+\beta/T}$$ (where T is temperature in degrees Kelvin)

The thermistors in this lab have a resistance of between 30 kΩ and 100 kΩ at 25 ˚C. If you want to put the probe into water, you will have to water-proof the tip of the probe. One water-proof method is to lightly paint the bare wires with paint or epoxy. Do not use electrical tape. Do not solder/epoxy/paint thermistors from checkout, because devices from checkout need to be returned.

2. Choose a sampling rate

In this lab, we will process temperature signals (0 to 10 Hz). According to the Nyquist Theorem, we need a sampling rate greater than 20 Hz. You are free to choose any sampling rate (f_s) greater than 20 Hz. Periodic timer hardware will trigger the ADC sample, and an ADC interrupt occurs when the sample is complete. The ADC ISR will read the 12-bit result and store it into an appropriate data structure. This interrupt needs to be high enough priority to guarantee no samples are lost. This selection of sampling frequency will affect the design of analog and digital filters. So, when you change the sampling rate, you will have to redesign the filters.

Nyquist Theorem: If f_{max} is the largest frequency component of the analog signal, then you must sample more than twice f_{max} in order to faithfully represent the signal in the digital samples. For example, if the analog signal is

$$A + B \sin(2\pi ft + \phi)$$

and the sampling rate is greater than $2f$, you will be able to determine A, B, f, and ϕ from the digital samples.

Valvano Postulate: If f_{max} is the largest frequency component of the analog signal, then you must sample more than ten times f_{max} in order for the reconstructed digital samples to look like the original signal when plotted on a voltage versus time graph.

3. Hardware Interface

Figure 9.3 shows one possible block diagram for the analog electronics of the digital thermometer. You must add an analog filter in this lab. Choose the cutoff frequency of the LPF to be about ½ the sampling rate. The amplifier should convert the entire temperature range into the 0 to +3.3 V ADC range. Good design practice is to map the temperature range into 0.2 to 3.1 V, because the ADC may have more error at its extremes. Because you are using rail-to-rail op amps, the entire system can be powered by a single +3.3 V supply. PLEASE DO NOT USE +12 OR -12 V SUPPLIES IN THIS LAB. If you have received free samples from TI in class, you should design your system using state of the art components (if you did not come to class that day, you can build the circuit using TLC2274 from ECE lab checkout). The gain of the INA122P amplifier is determined by the resistor R_G. Go to TI.com and download their design tool called **FilterPro**. I like version 2 of this design tool, which is posted on the EE445M lab page http://users.ece.utexas.edu/~valvano/EE345M/filterpro_design_program_download-h.exe

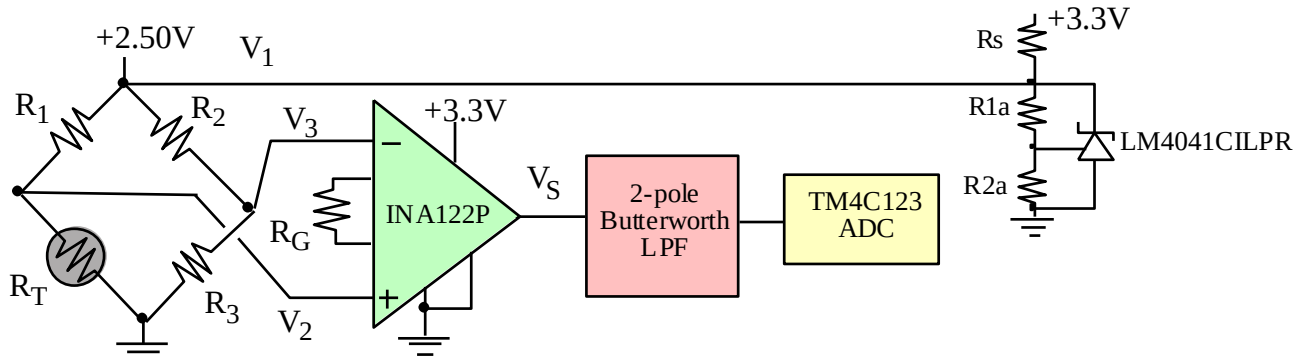

Figure 9.3. Possible thermistor interface (easy to construct, but more expensive).

We need an instrumentation amp (premade or built) because of its high input impedance and good CMRR. This semester, TI has an INA122P instrumentation amplifier that works well for this lab. If you do not have an INA122, then you can build your own instrumentation amplifier using 3 op-amps, as shown in Figure 9.4. The gain of the 3-op-amp instrumentation amplifier is a function of the resistors R_3 R_4 R_5 and R_6. The TLC2274 operates rail-to-rail, which means its output can swing all the way from 0 to +3.3 V. Any of the rail-to-rail op amps mentioned in class can be used in this lab: AD8032, TLC2274, or OPA2350. These op amps can operate on a single +3.3V supply. *In fact, if you connect the TLC2274 up to the usual +12 –12 V supplies, you will damage the device and damage your microcontroller.*

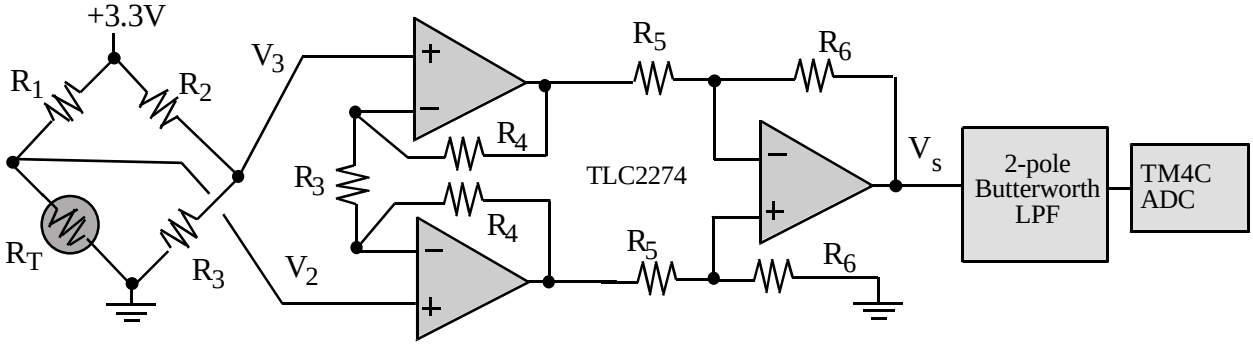

Figure 9.4. A good thermistor interface (harder to construct, but cheaper to build).

The advantage of using an instrumentation amp (with its high Z_{in}) is you can analyze the transfer function from R_T to voltage V_1 considering only the bridge input (+2.50 V), R_1 and R_T. Normally, the R_1 resistor in the bridge is chosen large enough to prevent self-heating the thermistor. Assume the dissipation constant to be about 1 mW/˚C. Limit the thermistor power to 0.1 mW so that the self-heating error is below 0.1 ˚C. The R_3 resistor in the bridge establishes one of the extreme values of the temperature range. For example, when the thermistor resistance, R_T, equals R_3, then the bridge output voltage is 0. The gain of the differential amplifier, along with the ADC range (0 to +3.3 V in our case) will determine the temperature range of the system.

4. Software Conversion:

Using the calibration data, the nonlinear thermistor equation, the characteristics of your analog circuit and the response of the microcontroller ADC, determine the ADC output sample for each temperature for about 5 to 20 temperature points within the 0 to 40 ˚C temperature range. (Again, you are free to adjust the temperature range, as you see fit.) There is an Excel spreadsheet to assist you called **Therm12.xls**. Show both, a table of figures and a plot of this data. Include appropriate intermediate voltages in the table (e.g., thermistor resistance, bridge output, and analog circuit output.) Design a software conversion routine that calculates temperature from the ADC sample. You should consider various methods:

a) linear equation (don't use because it has errors too large),
b) nonlinear equation,
c) large table lookup (one entry for each ADC value, i.e., 4096 entries),
d) small table lookup (≈50 entries) with linear interpolation in between.

Please consider dropout and overflow when implementing the conversion software.

5. Real-time ADC sampling:

The program will continuously sample the ADC at a sampling frequency, f_S, selected in part 2. The ADC must be implemented using interrupts. There are two good approaches to implementing real-time ADC sampling. The first approach is to implement a periodic interrupt using SysTick or a timer at the desired sampling rate. In the ISR you start the ADC conversion, wait for it to complete, read the ADC result, and pass the data from the ISR to the main program using a mailbox or FIFO queue. There is some sampling jitter to this approach due to the variable delay between interrupt trigger and the execution of the ISR (**ADCSWTrigger_xxx.zip**).

The second approach is to create a hardware timer period with a frequency matching the desired sampling rate. The timer is internally connected to the ADC to create a hardware trigger. The ADC is configured to start a conversion periodically by the hardware timer. The ADC is armed to generate an interrupt on completion. In this second approach, the ADC ISR does not need to start the conversion. In fact, the ADC completion flag triggers the interrupt. This ISR simply reads the data, acknowledges the interrupt and passes the data to the foreground using a mailbox or FIFO queue. There is no sampling jitter to this approach (**ADCT0ATrigger_xxx.zip**).

The main or foreground thread will get data from the mailbox or FIFO, calculate temperature from the ADC sample, and then display the measured temperature both numerically and graphically on the LCD display.

Preparation (do this before your lab period)

1. **Crude calibration.** The thermistor resistance varies nonlinearly with its temperature. Perform a very crude temperature calibration experiment with two points somewhere in your temperature range of 0 to 40 ˚C. We expect most students to use room temperature and axilla temperature for calibration. Your skin temperature in the axilla region (arm pit) is about 36 ˚C. The room temperature can be measured any thermometer. Use the Excel spreadsheet to determine R_0 and β from the two calibration points. In particular, **Therm12.xls**. This quick calibration will only be used to choose resistors in the circuit. A real calibration will be performed as procedure 5.

2. Review the technical information on the ADC system of TM4C123. List three ways you could use to initiate the ADC conversion process. What is the way to know when the conversion process has been completed? Place the answers to these two questions in the beginning comment section of your main program. You may use the example ADC programs, but it is your responsibility to understand the modes.

3. Choose one of the options as discussed in hardware section and design the appropriate thermistor amplifier. Be prepared during checkout to discuss the reasons for your choice of design. You must add a two-pole Butterworth low-pass anti-aliasing analog filter. Show name and number of all the pins involved including power. Add bypass capacitors on all chips. Why is it is important to connect bypass capacitors across the power pins for the analog IC components? Label all resistance and capacitance values and types. For example, 1kΩ 5% carbon, or 0.01μF 10% X7R ceramic. You do not need to show circuits on the evaluation board, such as switches and LEDs. Draw the circuit using the CAD tool like PCB Artist or Eagle.

4. This program is not used to measure temperature. Rather it is used to study the Nyquist Theorem as described in Procedure 1). Create a real-time data acquisition system that takes 100 samples at 1000 Hz, then prints out the data via the UART (**ADCPrintResults_xxx.zip**). You should first collect the data, next print it out, and then stop.

A "syntax-error-free" hardcopy listing for the software is required as preparation. The TA will check off your listing at the beginning of the lab period. You are required to do your editing before lab. The debugging will be done during lab. Document clearly the operation of the routines. The ADC and ADC trigger (e.g., Timer0A) modules must be written at a low level, like the book, without calling TivaWare driver code. Other code can use TivaWare driver code.

Procedure (do this during your lab period)

1. *Basic understanding*: The purpose of this section is to verify the **Nyquist Theorem** and the **Valvano Postulate**. Generate a continuous waveform (0.5 to +3V) with an adjustable frequency from 10 Hz to 10 kHz. Use a function generator but make sure the output is 0.5 to +3V. Please connect the analog waveform to a scope and verify the voltage range is between 0 and +3.3V before connecting to the microcontroller. VOLTAGES OUTSIDE THIS RANGE WILL DAMAGE THE TM4C. Next connect the signal to an analog input. In this part, we will not be using the thermistor or analog amplifier. Use the software system from Preparation 4) to capture 100 data points at 1 kHz sampling. Collect data for frequencies about 100 Hz (Valvano Postulate), 500 Hz (Nyquist), and 2 kHz (aliased). Use the UART output to transfer data to the PC. Plot the results by connecting the data points with a straight line. Describe the concepts of Nyquist Theorem, Valvano Postulate, and aliasing using this specific data. Be prepared to explain your results during checkout.

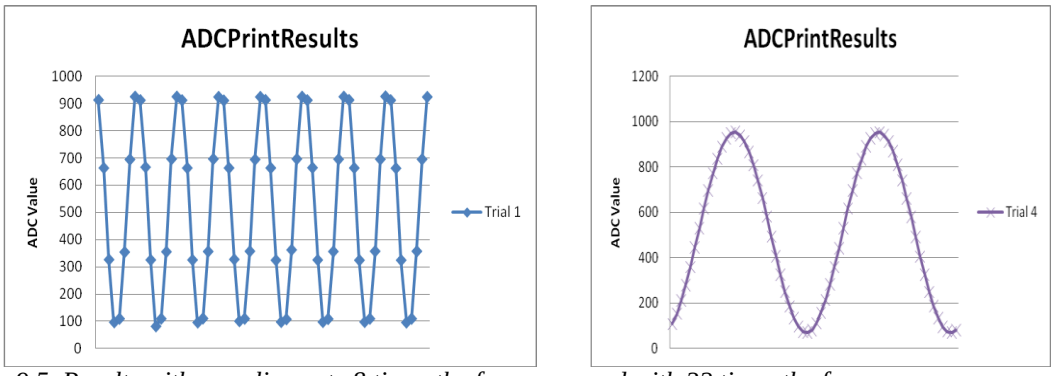

Figure 9.5. Results with sampling rate 8 times the frequency and with 32 times the frequency.

2. *Static analog circuit test*: Perform these tests before connecting the circuit to the TM4C123. Construct and evaluate the thermistor circuit. One by one replace the thermistor with 4 regular resistors that have resistances within the typical range of your thermistor. One test resistor should have a resistance equal to the resistance of the thermistor near the maximum temperature, and another test resistor should have a resistance equal to the resistance of the thermistor near the minimum temperature. Record the voltage values at strategic places in your analog circuit. What voltage output do you get when the thermistor is disconnected? What voltage output do you get when the thermistor wires are shorted? These measurements should match the numbers in the **therm12.xls** design template. You should modify your temperature measurement software to output specific error conditions if the thermistor is shorted or disconnected.

3. *Dynamic analog circuit test*: Again, perform these tests before connecting the circuit to the TM4C123. Disconnect the thermistor, and connect a sine-wave signal generator in its place. Make sure the voltage level of the signal generator is within range, so that the inputs and outputs of your analog circuit are not saturated. Record the sine-wave amplitudes of the input and output voltages. Start at about 1 Hz and collect measurements at ten different frequencies. Make sure you choose frequencies large enough to see the gain roll off. Calculate the gain at each frequency. Plot the gain versus frequency response of your circuit. In systems where the shape of the signal is important, such as audio or video, the phase versus frequency response is important. You do not have to measure the phase response of your analog circuit.

4. *Software system to measure temperature:* Write the software system that samples the ADC at a rate at or above 20 Hz. Pass the data from background into the foreground using a mailbox or FIFO queue. In the main program convert the ADC data to fixed-point temperature and display it on the LCD both numerically and

graphically. Also include on the LCD display, the ADC sample as 0 to 4095. Connect the output of your thermistor amplifier to the input of the ADC system.

5. *Calibration*. In this section your software will output the ADC sample as a decimal number. We define the temperature as measured by the Fluke 87V multimeter as reference truth. There is a K-type thermocouple that plugs into the Fluke allowing it to measure temperature. Its range is -40 to 260 °C (-40 to 500 °F), and its accuracy is 2.2 °C or 2 %. There are two of these meters on UTA basement lab, so record with which Fluke meter you calibrated. You are allowed to use another temperature reference, as long as it is as good as this Fluke. Place your thermistor and the reference thermometer at the same temperature (room temperature or an insulated cup). Wait for both your ADC measurement and the reference thermometer to stabilize. Record both the true temperature and the ADC sample as measured by your system. Record both the true temperature and the ADC sample as measured by your system. Incorporate this calibration data into a header file called **`calib.h`**. In particular, if you change thermistors or recalibrate, only changes to this header file will be required. You can use or not use the spreadsheet **Therm12.xls** as long as you incorporate the nonlinearity of the thermistors in an appropriate manner. One possible solution is to use the 10 and 40 ˚C to create a 51-point table from Therm12.xls. Then, use the calibration point as a temperature offset to adjustment. I.e., sample the ADC, use the therm12.xls lookup tables to convert ADC into temperature, then add a temperature correction as determined by the one-point calibration.

Figure 9.6. The Fluke 87V can measure temperature.

6. *Accuracy*. **Accuracy** is defined as the absolute difference between the true temperature and the value measured by your device. Accuracy is dependent on the same parameters as resolution, but in addition it is also dependent on the stability of the transducer and the quality of the calibration procedure. Please use the same Fluke meter as you used in calibration. In this section your software will output the temperature as a fixed-point number with a 0.01 ˚C format. Collect 5 measurements at one temperature: either in room air, or in an insulated container, creating a table showing the true temperature (x_{ti} as determined by the Fluke meter), and measured temperature (x_{mi} using your device). Calculate average accuracy by calculating the average difference between truth and measurement,

Average accuracy (with units in ˚C) = ____

7. *Reproducibility*: Place the thermistor in either room air, or in an insulated container, and record 10 independent temperature measurements. Calculate the standard deviation of these data and report S (estimation of σ) as reproducibility.

Deliverables (exact components of the lab report)

A) Objectives (1/2 page maximum)

B) Hardware Design

Circuit diagram of the thermistor interface

C) Software Design (a hardcopy software printout is due at the time of demonstration)

1) Calibration data (procedure 5 and the **`calib.h`** file)
2) Low level ADC interface (**`ADC.c`** and **`ADC.h`** files)
3) Main program used to measure temperature

D) Measurement Data

1) Sketch three waveforms (procedure 1)
2) Static circuit performance (procedure 2)
3) Dynamic circuit performance (procedure 3)
4) Accuracy (procedure 6)
5) Reproducibility (procedure 7)

E) Analysis and Discussion (give short answers to these questions)

1) What is the Nyquist theorem and how does it apply to this lab?
2) Explain the difference between resolution and accuracy?
3) Derive an equation to relate reproducibility and precision of the thermometer.
4) What is the purpose of the LPF?

5) If the R versus T curve of the thermistor is so nonlinear, why does the voltage versus temperature curve look so linear?

6) There are four methods (a,b,c,d) listed in the **4) Software Conversion** section of methods and constraints. For one of the methods you did not implement, give reasons why your method is better, and give reasons why this alternative method would have been better.

Checkout (show this to the TA)

You should be able to demonstrate the proper operation of digital thermometer. We expect your accuracy to be at least 1 ˚C and your reproducibility to be less than 0.1 ˚C. Have a Fluke thermometer available during checkout/

Follow the submission directions given to you by your TA. Software style will be graded.

Hints

1) This is a long lab with many parts, so start early.
2) Don't try to complete the experiment in one full swoop. Run the hardware test program given above before testing your software. Debug the system in an analytical, step-wise manner.
3) Please do not use regular op amps that require ±12 V supplies. An advantage of the single supply op amp with rail-to-rail outputs is that the output of your analog amplifier will remain within the 0 to +3.3V ADC range.
4) Circuits built with either the INA122P or AD627 are very easy to implement and have much improved performance over instrumentation amps built with three op amps.
5) The underlined sections identify components that must be performed and included in the lab report.
6) How to connect the UART to a PC.

A) First, you plug the TM4C into the PC

B) Open the device manager in Windows to see which COM port it is. It will be somewhere in the range COM1 to COM99.

C) I like PuTTy; you can get putty at http://www.putty.org/. Select the COM port, 119200 bits/sec, 8 bit data, no parity, no hardware flow control.

D) Open the ADCPrintResults_xxx.zip example, compile download and run.

7) Here is one way to use the ST7735 graphics routines as a sweeping graph, where new data overwrites existing data. Temperature is plotted versus time. *N* will be the number data points/pixel along the time axis. Assume *N* is a power of 2. There will be 128**N* data points per sweep. Because of the ST7735 hardware there will be exactly 128 pixels along time axis. If the sampling rate is f_s, then it will take 128**N*/ f_s time to sweep across once.

A) Call **`ST7735_PlotClear`** once. For example, if the minimum temperature is 1000 and maximum is 4000, we initialize the plot by calling:

```
ST7735_SetCursor(0,0); ST7735_OutString("My awesome machine");
ST7735_PlotClear(1000,4000);  // range from 0 to 4095
ST7735_SetCursor(0,1); ST7735_OutString("N=");
ST7735_SetCursor(0,2); ST7735_OutString("T="); ST7735_sDecOut2(2500);
                       ST7735_OutString(" C");
```

B) Let **`j`** be a local variable that counts the number of data points collected. Let **`fs`** be the sampling rate in Hz. Let **`Data`** be the current 12-bit ADC sample. Let **`Temperature`** is a fixed point variable with units 0.01 C. **`ST7735_sDecOut2`** outputs the temperature in 0.01 C in a manner similar to **`ST7735_sDecOut3`** from Lab 1. Perform these functions for every sample. Basically this code runs in the main program, but executed at the sampling rate (using a mailbox or FIFO to collect data from the ISR).

```
ST7735_PlotPoint(Temperature);  // Measured temperature
if((j&(N-1))==0){           // fs sampling, fs/N samples plotted per second
  ST7735_PlotNextErase();  // overwrites N points on same line
}
if((j%fs)==0){    // fs sampling, 1 Hz display of numerical data
  ST7735_SetCursor(3,1); ST7735_OutUDec(Data);            // 0 to 4095
  ST7735_SetCursor(3,2); ST7735_sDecOut2(Temperature/10); // 0.01 C
}
j++;                        // counts the number of samples
```

Temperature Resolution (skip this section):

To measure temperature resolution, we use the student's t-test to determine if the system is able to detect the change. To use the student's t test we need to make the following assumptions:

1) the errors in one data set are independent (not correlated to) the errors in the other data set;
2) the errors in each data sample are independent (not correlated to) the errors in other data within that set;
3) the errors are normally distributed;
4) the variance is unknown;
5) the variances in the two sets are equal.

If a random variable, X, is normally distributed with a mean is μ and a standard deviation of σ, then the probability that it falls between $\pm 1\ \sigma$ is 68 %. I.e.,

$P(\mu-\sigma < X < \mu+\sigma) = 0.68$

Similarly,

$P(\mu-1.96\sigma < X < \mu+1.96\sigma) = 0.95$
$P(\mu-2\sigma < X < \mu+2\sigma) = 0.954$
$P(\mu-2.58\sigma < X < \mu+2.58\sigma) = 0.99$
$P(\mu-3\sigma < X < \mu+3\sigma) = 0.9997$

The square of the standard deviation is called variance, σ^2. In most situations, we do not know the mean and standard deviation, so we collect data and estimate them. In particular, we take multiple measurements assuming the temperature is constant. Let $\boldsymbol{X_i}$ be repeated measurements under the same conditions, and $\boldsymbol{N}$ is the number of measurements (e.g., $\boldsymbol{N} = 10$).

$$\overline{X} = \frac{1}{N}\sum_i X_i \qquad S^2 = \frac{1}{N-1}\sum_i (X_i - \overline{X})^2$$

The N-1 term is used in the calculation of S because there are N-1 degrees of freedom. These expressions are unbiased estimates of μ and σ, meaning as the sample size increases the estimates approach truth. Formally, we say the expected value of $\overline{X}$ is μ, or $E(\overline{X}) = \mu$. Similarly, the expected value of S^2 is σ^2, or $E(S^2) = \sigma^2$.

For example, we collect two sets of data (e.g., 10 measurements in each set, $\boldsymbol{N} = 10$), and we want to know if the means of two sample sets are different. Consider the measurements in the two data sets as the sum of the true value plus an error:

$X_{01} = \mu_0 + e_{0i}$
$X_{i1} = \mu_1 + e_{1i}$

Assumption 1 states that e_{0i} are not correlated to e_{1i}. Assumption 2 states that e_{0i} are not correlated to e_{0j} and e_{1i} are not correlated to e_{1j}. Thermal noise will satisfy these assumptions. We employ a test statistic to test the hypothesis H_0: $\mu_0 = \mu_1$. First, we estimate the means and variances of the data (assuming equal sized samples)

$$\overline{X_0} = \frac{1}{N}\sum_i X_{0i} \qquad S_0^2 = \frac{1}{N-1}\sum_i (X_{0i} - \overline{X_0})^2$$

$$\overline{X_1} = \frac{1}{N}\sum_i X_{1i} \qquad S_1^2 = \frac{1}{N-1}\sum_i (X_{1i} - \overline{X_1})^2$$

From these, we calculate the test statistic **t**:

$$t = \frac{\overline{X_1} - \overline{X_0}}{\sqrt{S_0^2/N + S_1^2/N}}$$

The two sets of data, together, have 2N-2 degrees of freedom. The student's t table, shown as Table 9.1, has two dimensions. In the vertical direction, we specify the degrees of freedom, **df**. For example, if there are 10 data points in each data set, then **df** equals 18. In the horizontal direction we select the probability of being correct. For example, if we wish to be 99% sure of the test, then we select the 99% column. Selecting the row and the column allows us to pick a number threshold. For example, the number in the **df**=18 row, **confidence**=99% column is 2.878. This means if H_0 is true, then

Probability of $\mathbf{t} < -2.878 = 0.005$ and Probability of $\mathbf{t} > 2.878 = 0.005$

Therefore

Probability of $-2.878 < \mathbf{t} < 2.878 = 0.99$ (confidence interval of 99%)

If we collect data and calculate **t** such that the test statistic **t** is greater than 2.878 or less than -2.878, then we claim "we reject the hypothesis H_0". If the test statistic **t** is between -2.878 and 2.878 we do not claim the hypothesis to be true. In other words we have not proven the means to be equal. Rather, we say "we do not reject the hypothesis H_0".

confidence	80%	90%	98%	**99%**	99.8%	99.9%
df p=	0.10	0.05	0.01	**0.005**	0.001	0.0005
8	1.397	1.860	2.896	**3.355**	4.501	5.041
9	1.383	1.833	2.821	**3.250**	4.297	4.781
10	1.372	1.812	2.764	**3.169**	4.144	4.587
11	1.363	1.796	2.718	**3.106**	4.025	4.437
12	1.356	1.782	2.681	**3.055**	3.930	4.318
13	1.350	1.771	2.650	**3.012**	3.852	4.221
14	1.345	1.761	2.624	**2.977**	3.787	4.140
15	1.341	1.753	2.602	**2.947**	3.733	4.073
16	1.337	1.746	2.583	**2.921**	3.686	4.015
17	1.333	1.740	2.567	**2.898**	3.646	3.965
18	**1.330**	**1.734**	**2.552**	**2.878**	**3.610**	**3.922**
19	1.328	1.729	2.539	**2.861**	3.579	3.883
20	1.325	1.725	2.528	**2.845**	3.552	3.850
21	1.323	1.721	2.518	**2.831**	3.527	3.819
22	1.321	1.717	2.508	**2.819**	3.505	3.792
23	1.319	1.714	2.500	**2.807**	3.485	3.767
24	1.318	1.711	2.492	**2.797**	3.467	3.745
25	1.316	1.708	2.485	**2.787**	3.450	3.725
26	1.315	1.706	2.479	**2.779**	3.435	3.707
27	1.314	1.703	2.473	**2.771**	3.421	3.690
28	1.313	1.701	2.467	**2.763**	3.408	3.674
29	1.311	1.699	2.462	**2.756**	3.396	3.659
30	1.310	1.697	2.457	**2.750**	3.385	3.646
40	1.303	1.684	2.423	**2.704**	3.307	3.551
50	1.299	1.676	2.403	**2.678**	3.261	3.496
60	1.296	1.671	2.390	**2.660**	3.232	3.460
80	1.292	1.664	2.374	**2.639**	3.195	3.416
100	1.290	1.660	2.364	**2.626**	3.174	3.390
120	1.289	1.658	2.358	**2.617**	3.160	3.373
∞	1.282	1.645	2.326	**2.576**	3.090	3.291

Table 9.1. Student's t distribution table.

Resolution: Take 10 measurements at 37˚C . Next, set the temperature to 37.1 ˚C, and record 10 more independent temperature measurements. Again, calculate the standard deviation of these data. Next calculate the student's t statistic and use it to determine if your system as a temperature resolution of 0.1 ˚C or better.

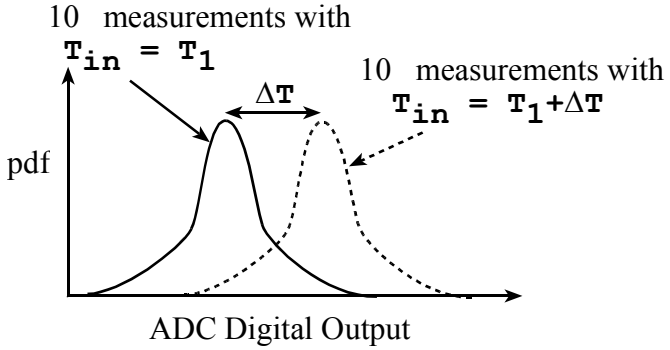

Figure 9.7. Resolution means if the temperature increases by ΔT, the system will probably notice.