Experiment C1
Introduction to Electronics and LabView
Procedure

Deliverables: Checked lab notebook, Brief Tech Memo

Part I: Programming in LabView

For most of the labs in this course will use LabView to implement your control algorithm and acquire data. Please do the following tutorials. Take turns with your lab partner for writing the code. Demonstrate each one to the lab instructor when you are finished.

Code Library

Throughout the semester, you will build a "code library" containing all of the LabView programs, Arduino Sketches, and data sets that you generate. You must keep a multiple copies of the library on the lab computer, a thumb drive, and on the cloud.

1. Create a folder on the desktop titled "AME40453_YourName".
2. Create the following sub-directories (folders inside the folder you just created): "C1", "C2", "C3", "C4", "C5", "C6", "C7", "C8", and "Final_Project".
3. Save multiple versions of your codes and data sets in the appropriate folder.
4. At the end of each lab, you must create back-up copies of this on your thumb drive and the cloud (i.e. Google drive or Box).

Analog Input

You will use the USB-6341 DAQ device to measure an analog voltage.

1. Everything you need for the first four labs is in a large plastic tote box. Remove the breadboard, tackle box, wire box, and USB-6341 DAQ from the tote.
2. Make sure the USB-6341 DAQ device is connected to the computer.
3. Open LabView 2017 and create a blank VI. (VI stands for "virtual instrument".) Save it as "analog_input_YourName.vi" in the C1 folder in your code library.
4. There are two main windows for the VI. The "block diagram" is where you write the LabView "code". The "front panel" displays data and allows user to interact with the I/O device. Press ctrl + E to toggle between the block diagram and front panel.
5. In the block diagram, right click the white background. In the pop-up "pallet" menu, select "Express" > "Input" > "DAQ Assist" and place the icon on the block diagram.
6. In the window that appears, select "Acquire Signals" > "Analog Input" > "Voltage" > "ai0" and press finish. The DAQ Assistant window should appear.
7. Make sure the breadboard is turned off. Connect the ground and +15V variable supply ports to the horizontal bus lines.

8. Cut off a foot each of red and black wire from the spools and strip the ends. Insert them into the "AI" (analog input) screw terminals of the USB-6341. Black should be connected to ground and red should be connected to "ai0". Refer to the pin-out in on the inside cover of the USB-6341.
9. Connect the other ends of the wires to the ground and +15V horizontal bus lines that you wired up in the previous step.
10. Ask the lab instructor to check your set up. After the instructor approves, turn on the breadboard.
11. In the DAQ Assistant window, select "Acquistion Mode" > "1 Sample (On Demand)" and set the "Terminal Configuration" to "RSE". This will make the voltage measurement be relative to ground.
12. Press the "Run" button on top of the screen. Twiddle the +15V knob on the breadboard. You should see the voltage go up and down in the DAQ Assistant window.
13. Press OK to return to the block diagram. Wait for LabView to build the Express VI.
14. Press ctrl + h to bring up the "context help" window. Place the cursor over the object you created, and the help window will show you a pinout.
15. Right click out "data" pin and select "Create" > "Numeric Indicator". Double click the newly created indicator, and LabView will show you where it is in the front panel. Rename the indicator to be "Analog voltage ai0"
16. In the front panel, press the "Run Continuously" button (looks like a refresh or recycle sign).
17. Twiddle the +15V knob on the breadboard. You should see the voltage go up and down on the indicator in the front panel.
18. Press the stop button on the front panel. Disconnect the wires. Save and close the VI.

Analog Output

You will use the USB-6341 I/O device to create analog voltage that you can control from the computer.

1. Create a New VI, and save it as "analog_output_YourName.vi" in the C1 folder.
2. In the block diagram, right click the white background. In the pop-up menu, select "Express" > "Output" > "DAQ Assist. Place the icon in the block diagram.
3. In the window that appears, select "Generate Signals" > "Analog Output" > "Voltage" > "ao0" and press finish. The DAQ Assistant window should appear.
4. In the DAQ Assistant window, select "Generation Mode" > "1 Sample (On Demand)" and set the "Terminal Configuration" to "RSE". This will make the voltage output be relative to ground.
5. Cut off a foot each of red and black wire from the spools and strip the ends. Insert them into the "AO" (analog output) screw terminals of the USB-6341. Black should be connected to ground and red should be connected to "ao0". Refer to the pin-out inside the cover of the USB-6341.

6. Connect the other ends of the wires to the Extech handheld DMM.
7. In the DAQ Assistant window, press the "Run" button and adjust the "VoltageOut" value. Use the DMM to measure the voltage as you change the value.
8. Press OK to return to the block diagram. Wait for LabView to build the Express VI.
9. Press ctrl + h to bring up the "context help" window. Place the cursor over the object you created, and the help window will show you a pinout.
10. Right click the "data" pin and select "Create" > "Control". Double click the newly created control block, and LabView will show you where it is in the front panel. Rename is to be "Analog output ao0".
11. In the front panel, press the "Run Continuously" button (looks like a refresh or recycle sign).
12. Adjust the voltage value in the font panel. You should see the voltage on the DMM change accordingly.
13. Press the stop button on the front panel. Disconnect the wires. Save and close the VI.

While Loop

1. Create a blank VI, and save it as "While_loop_YourName.vi" in the C1 folder.
2. In the block diagram, right click the white background. In the pop-up menu, select "Structures" > "While loop".
3. Use the cursor to draw a big box on the screen. The box is a while loop, and everything inside will be looped.

Pro-Tip: Ctrl + b will remove any broken wires from the block diagram.

4. Right click the stop sign button in the bottom right corner of the while loop. Select "Create Control" and create a stop button. This button will allow you to stop the while loop.
5. In the block diagram, right click the white background. In the pop-up menu, select "Timing" > "Wait Until Next ms Multiple" (looks like an metronome) and place it in the loop.
6. Right click the input terminal of the metronome block and select "Create" > "Constant", and enter a value of 2000. This will make the while loop wait at least 2000ms before the next iteration.
7. Right click the border of the while loop and create a "shift register". Shift registers allow you to pass values from the previous iteration of the loop to the next iteration.
8. Use the array pallet under programming structures to create the program shown in **Appendix A**. This will save the clock values into an array. You will need to use "Build Array" and "Initialize Array" from the Array pallet and "Multiply" from the Numeric pallet.
9. Activate the "Highlight Execution" button (looks like a light bulb) at the top of the block diagram.

10. Press the "Run" button in the top left corner. With "Highlight Execution" activated, you can see how information is being passed between terminals in the block diagram. This is an extremely useful tool for debugging code.
11. Stop the program. Deactivate the "Highlight Execution" button. Save but do NOT close the VI.

Collecting and Saving Data

1. Select "Save as" and save the while loop via under a new name: "clock_data_YourName.vi".
2. In the block diagram, right click the white background. In the pop-up menu, select "Structures" > "Flat Sequence".
3. Use the cursor to draw a big box around the while loop. The box is called a "frame", similar to a frame in a movie.
4. Right click the border of the first frame and select "add frame after".
5. Right click a blank spot in the new frame and select "Express" > "Output" > "Write Meas File". Place the icon in the second frame.
6. In the "Configure" window, select "Ask user to choose file" > "Ask only once". Press OK.
7. Connect the array from the first frame to the "Signals" input in the second frame.
8. Press the run button in the front panel and wait about 10 seconds.
9. Press the "STOP" button that you created. NOT the stop sign button on the top.
10. The program should prompt you to save the data. Save it in your C1 folder with an appropriate file name.
11. Open the file with WordPad and inspect the data. Does it look correct?
12. Run the program again. This time, use the stop sign on the tool bar button to stop the program. Did it ask you to save the data? …Oops! The data is gone!
13. Save and close the VI.

Part II: Measuring Temperature

Background

You will now use a thermistor in a voltage divider circuit to measure temperature, as shown in Fig. 1, and write a LabView program to display and save the measured temperatures. The thermistor has a negative temperature coefficient (NTC), which means the resistance decreases with increasing temperature. The temperature T is related to the thermistor resistance R_s via the Steinhart interpolation formula

$$T(R_s) = \left[A + B\ln\left(\frac{R_S}{R_{ref}}\right) + C\ln\left(\frac{R_S}{R_{ref}}\right)^2 \right]^{-1}, \quad (1)$$

where $R_{ref} = 10\text{k}\Omega$, $A = 3.354\times10^{-3}$ K^{-1}, $B = 2.57\times10^{-4}$ K^{-1}, and $C = 2.62\times10^{-6}$ K^{-1}. The output voltage V_S of the circuit shown in Fig. 1 is given by the voltage divider equation

$$V_S = \left(\frac{R_2}{R_2 + R_S}\right) 12V, \quad (2)$$

where $R_2 = 4.7\text{k}\Omega$. You will write a LabView program to digitally record V_S and convert the measured voltages to temperatures.

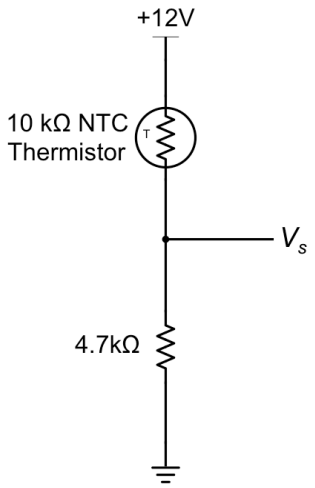


Figure 1 – A thermistor is wired up in a voltage divider circuit. Equations (1) and (2) can be combined to compute the temperature from the voltage output V_S.

Procedure

1. Sketch the circuit shown in Fig. 1 in your lab notebook.
2. Construct the circuit on the breadboard. Use the variable +15V power supply to provide the +12VDC.
3. Test the circuit. Use the handheld DMM to measure V_S. Pinch the thermistor, and you should see the voltage change as it warms in your hand. Does it increase or decrease? Based on the pre-lab assignment, is this what you expect?
4. Modify the final VI from Part I to record data from the circuit. (Be sure to "save as" a new

file name!) The program should do the following:

a. Have an indicator that displays the voltage V_S.

b. Convert the voltages to temperatures (degrees K) and save them into another array. In the "Numeric", create a "Expression Node" and enter the formulae you derived in the pre-lab assignment. (It is easiest to first convert the voltage to resistance in one node, then use a second node in series to convert the resistance to temperature.)

c. Save the temperature into an array at a rate of **4 samples/second**. You can do this by changing the "wait" time between loop iterations.

d. Convert the time data to be in seconds instead of milliseconds.

e. A graph of temperature vs. time that updates in real time. Right click anywhere in the **front panel** and select "Express" > "Graph Indicators" > "XY Graph", and place it somewhere in the front panel. In the block diagram, move it into your loop and connect it to the thick orange lines containing the temperature and times arrays. Make sure the axes in the front panel are properly labeled with units.

f. Have a stop button that, when pressed, will terminate the program and prompt the user to save the temperature and time arrays as a .lvm file. To save both the time and temperature arrays, you will need to right click the "data" wire going into the DAQ Assistant icon in the second frame and select "Signal Manipulation Palette" > "Merge Signals".

5. Test the LabView program. Pinch the thermistor with your fingertips for about a minute, so the temperature on the graph increases to steady state. Then, let go so the temperature decreases to its steady state room temperature.

6. Let it sit at room temperature for at least a minute. Does the temperature remain constant, or is there some noise causing it to randomly fluctuate? Stop the program and save the data.

7. Take screen shots of your front panel and block diagram code. Save a back-up of the VI to your email or Google drive.

8. Disassemble your circuit and disconnect your cables. Return all components to their appropriate bins. Carefully return everything to the large plastic tote box, and place it the cabinet above your lab bench.

Data Analysis and Deliverables

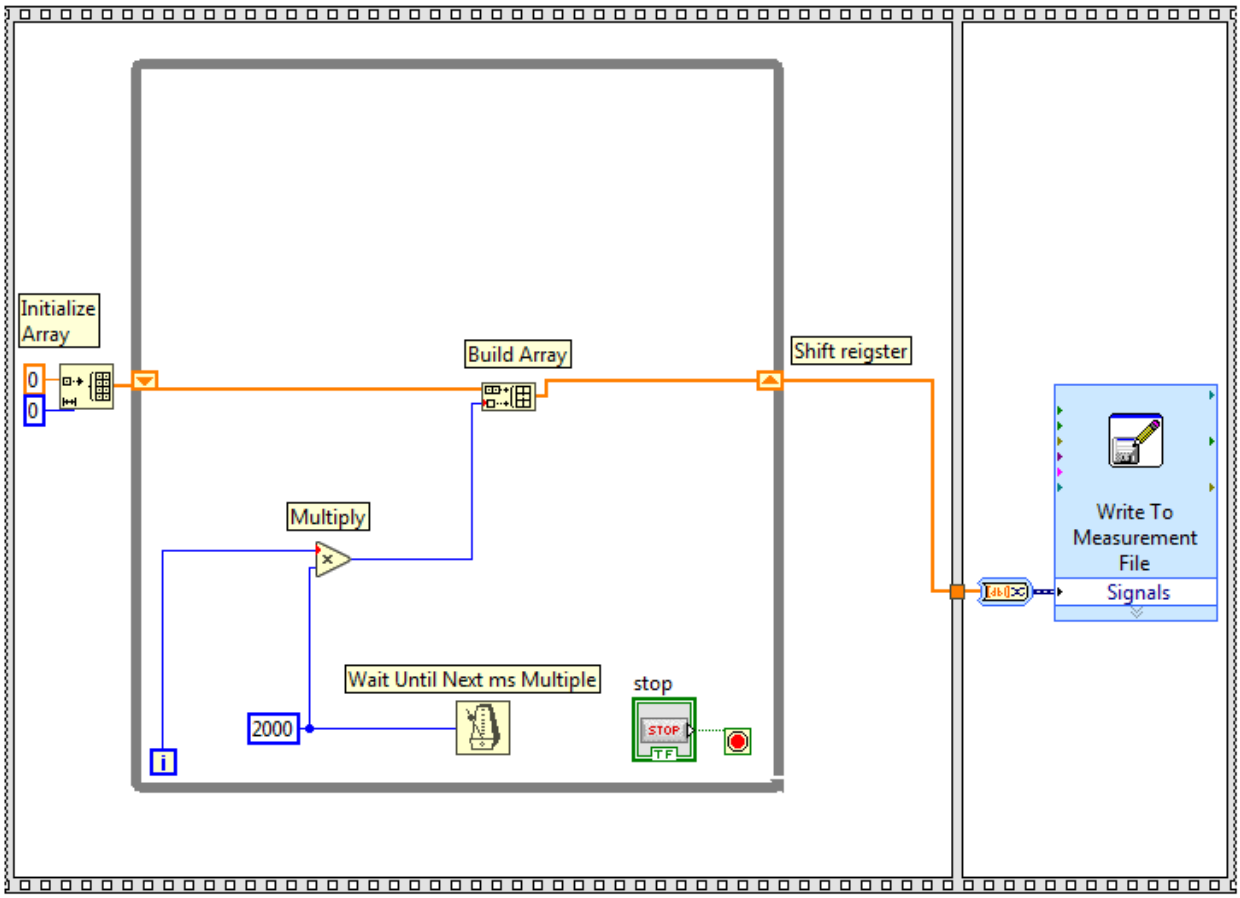
Using LaTeX or MS Word, make the following items and give them concise, intelligent captions. Make sure the axes are clearly labeled with units. Plots with multiple data sets on them should have a legend. **Additionally, write 1 – 3 paragraphs separate from the caption describing the plots/tables. Any relevant equations should go in these paragraphs.**

1. Screen shots of the front panel and block diagram code from your LabView program from Part II.
2. Using Matlab, make a plot of the temperature vs. time data recorded by your LabView program in Part II.

Talking Points – Discuss these in your paragraphs.

- Comment on the accuracy of the temperature measurement. Estimate the signal-to-noise ratio.

Appendix A


Initialize Array
0
0
Build Array
Shift reigster
Multiply
Wait Until Next ms Multiple
2000
stop
STOP
TF
Write To Measurement File
Signals

Appendix B

Equipment

Part I

- USB-6341 DAQ
- PC computer with LabView

Part II

- Vishay NTCLE100E3103JB0, NTC Thermistor 10k Bead (Digikey part # BC2301-ND)
- 4.7kΩ resistor
- Breadboard
- 22 gauges wire