Phase 3 Lab Exercise #2 – Using a GUI to Record and Present Data

INSTRUCTIONS

Complete the exercises below and upload them to Canvas as a single MATLAB script file using the naming convention "ENGR131_21S_P3_Lab##_abc123.m", replacing abc123 with your Case ID, and ## with the two-digit lab number.

For example, if Dr. Williams were submitting this lab, it would be ENGR131_21S_P3_Lab02_mrw8.m

For your script, please perform the following:

1. You may use the code and notes from class, the textbook, MATLAB's documentation, and anything you find using Google to solve these problems.
2. Use comments as appropriate to indicate your thoughts and how your code works (or is supposed to work). This is 5 points (9%) of your grade.

ARDUINO HARDWARE SETUP

For this lab, you will be using the DHT11 Temperature and Humidity Sensor that comes in your Elegoo kit. This sensor is a pre-fabricated and calibrated sensing unit complete with it's own microcontroller that collects and transmits data via a single data line. Refer to Figure 1 for a schematic in how to interface this device with the Arduino and Part 2.10 Module on your Elegoo CD for more information on the sensor itself.

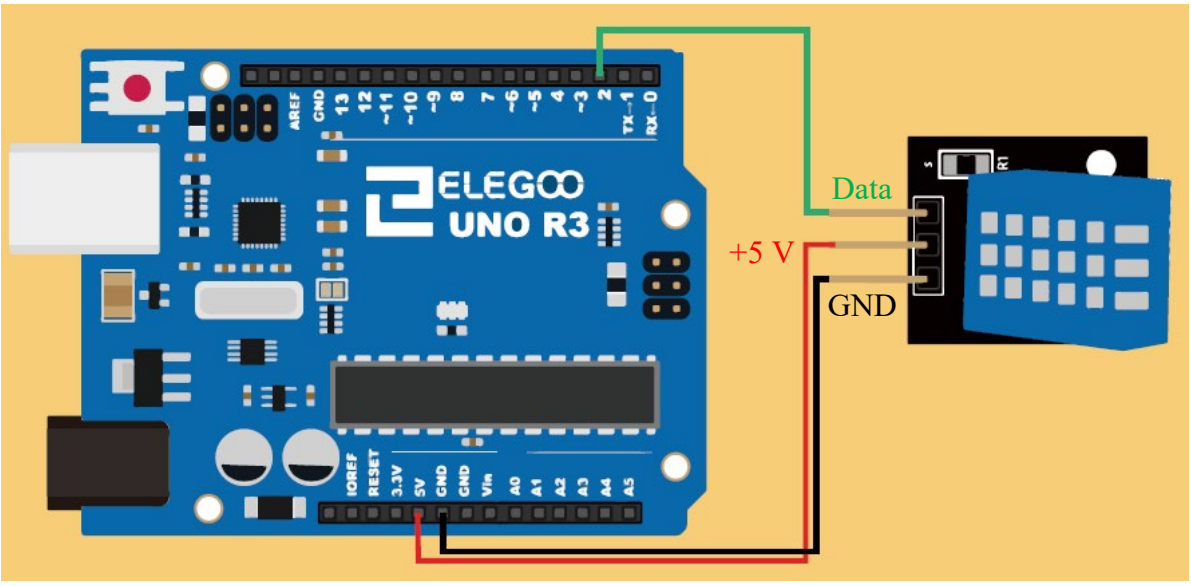


Figure 1. Diagram of how to connect the DHT11 sensor.

QUESTIONS

There are 3 questions for this lab.

1. COLLECT DATA FROM ARDUINO (5 PTS)

The first step in using the Arduino to collect temperature and humidity data is to revise the code that comes with the Elegoo Kit. Do this by:

A. Locating and opening the DHT11_Example in the 2.10 DHT11 Temperature and humidity Sensor module on the Elegoo CD.
B. In the main loop() function, revise the serial output statements to stream out only the data values separated by a comma (remove the data labeling text).
C. Before running, include the DHT library by selecting Add .ZIP Library… in the Arduino IDE as shown in Figure 2. The .ZIP library can be found on the 2.10 DHT11 Temperature and humidity Sensor module on the Elegoo CD.

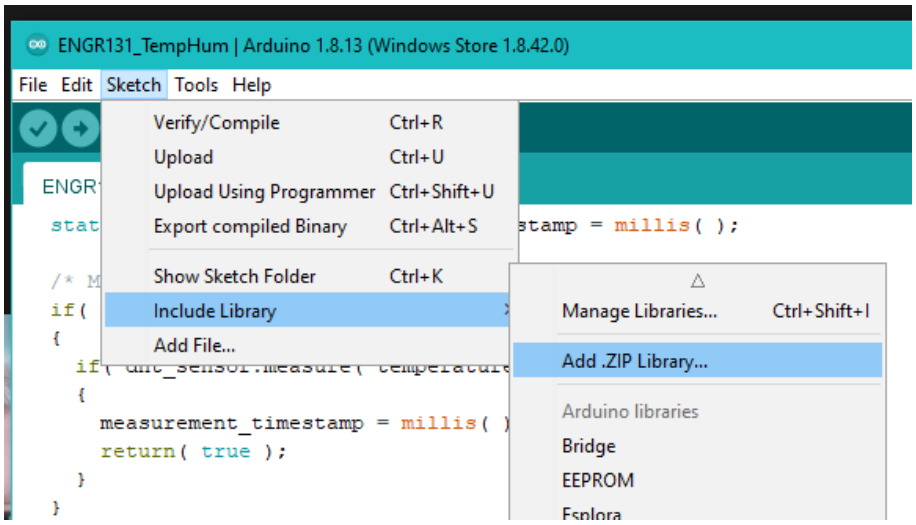


Figure 2. Menu location of how to add libraries. Libraries are pre-defined instructions that allow you to use many third-party devices without the need for writing specific code to govern their operation.

2. CREATE GUI (10 PTS)

Create a GUI to receive and plot the data streamed from the Arduino. The appearance of the GUI is up to you, but it must have the elements listed below. In the Design View mode of the App Designer, add these controls to your GUI by dragging them over from the Component Library (on the left side of app designer) and placing them on the GUI. Be sure to use pertinent names for each control. The required elements include:

A. Recording controls
    a. A knob to set the recording time (15, 30, 45, or 60 seconds)
    b. A button to start the recording
B. User feedback
    a. A gauge to indicate the recording time progress
    b. A lamp to indicate when the system is recording and when it is done
    c. A gauge to indicate the cumulative number of recordings.
C. Plot Controls
    a. Switches for plotting the Temperature and Humidity

b. Pull down menus to set the line color for each plot. Set the values for these the names of the common Matlab plotting colors (Blue, Red, Black, etc.)
c. A text edit field to get the user preference for markers or no markers
d. A button to plot

D. The axes to plot the data
E. A button to save the data to an Excel file

3. Add Functionality to Control Callbacks (40 pts)

Now that you have controls, you'll need to fill in what they will do when activated. Not all of them require a callback – the functional callbacks can get the data they need from other controls. This section will be separated into "Housekeeping", "Recording", "Plotting", and "Saving the Data" functions.

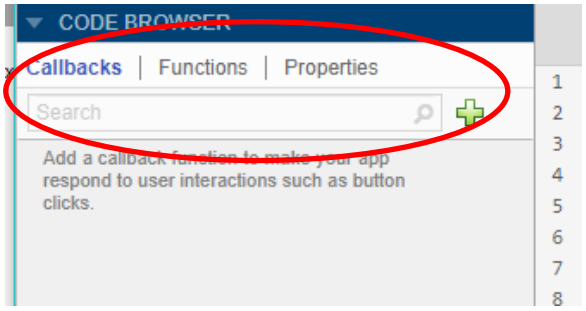
A. Housekeeping

The callbacks in this section are used to set up the GUI for recording and "clean up" the workspace when it is closed. These will all be created in the Code View mode of the App Designer.

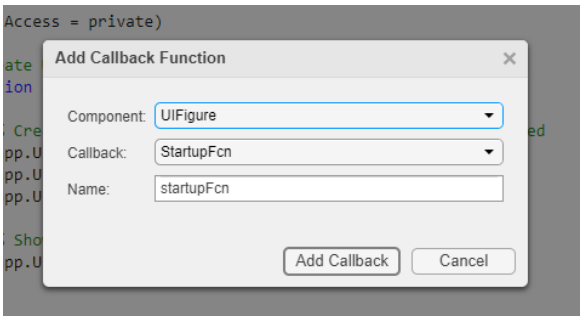
Create Properties (5 Pts)

Remember that properties are essentially global variables that are useable by all functions and callback in the GUI. You will need two – one for the serial object and one for the recorded data. To create these:

a. In the Code Browser section (left hand side), press the Properties control to switch to the Properties mode (Figure 3a).
b. Press the green + sign to add a property to the code
c. Name this property to be whatever you wish to call the serial object
d. Repeat steps b and c to create a property to hold the recorded data.



(a)



(b)

Figure 3. Screen capture of Code Browser section with the Callbacks, Functions, Properties, and Add Callback Function controls indicated with a red oval (a) and an example of the Add Callback Function window (b).

Create Startup and Close GUI Callbacks (5 Pts)

These callbacks will be used to create the serial object when the GUI starts as well as clear it when the GUI is closed.

a. In the Code Browser section (left hand side), press the Callback control to switch to the Callbacks mode (Figure 3a).
b. Press the green + sign to open the add callback function window (Figure 3b).
c. Select StartupFcn and add the callback to the code
    i. Use the serialport command to create the serial object using the COM port and baud rate desired. Assign this to the serial object property you defined in step 3.A.c
d. Create another new callback, this time selecting CloseRequestFcn
    i. Use the clearvars command to make sure the workspace is cleared when the GUI is closed.

B. Recording (10 Pts)

a. Right-click on the button you are using to start recording data from the Arduino. Select Add callback as shown in the menu shown in Figure 4. This will add the callback to the code and switch to the Code View mode.

b. Write code that will do the following:

i. Change the lamp color to indicate that the GUI is recording.

ii. Get the recording time from the recording time knob and convert the text to a number

iii. Similar to Phase II Lab 1 and Phase III Lab 1, use a while loop to record a set of data for the specified length of time. Read in the data as a string array.

1. Inside this loop, set the value of the recording progress gauge based on the specified recording time and the timer used in the loop
2. Use the drawnow command to enforce updating the GUI

iv. Change the lamp color to a different color to indicate that the recording is done, pause for a few seconds, and then go back to the "not doing anything" color.

v. Increment the value of the cumulative number of recordings gauge by one.

vi. Convert the recorded string array of data into numeric data

vii. Assign the recorded data to the recorded data property created in step 3.A.d

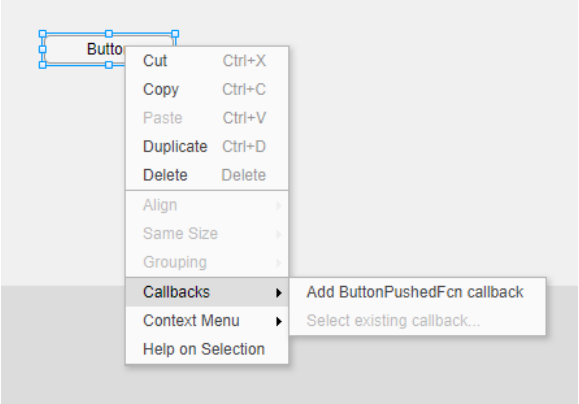


Figure 4. Screen capture showing how to create callback for GUI control

C. Plotting (15 Pts)

a. Similar to step 3.B.a, create a callback for the plotting function (Figure 4).

b. Determine the marker type to use:

i. Assign the value the user entered into the marker text edit field to a variable.

ii. Use the contains command to check whether the value is Y or y and if so, assign marker types for each data series as characters (ex. TempMarker = 'o';)

iii. If the value entered anything else, use marker type 'none'.

c. Create a function to set the plot color specification based on the pulldown menu selection

i. In the Code Browser section (left hand side), press the Functions control to switch to the Functions mode (Figure 3a).

ii. Press the green + sign to add a function to the code

iii. Give the function a useful name and add the input argument to the default app input argument (ex. results = MyAwesomeFcn(app, MyCoolInputvar)

iv. Use a switch statement to convert the color names to color and line specifications (ex. 'r-' for Red, 'b-' for Blue, etc.)

d. For each data series, plot the data if the switch for that data type is in the "on" position

i. Use the contains command to check the value of the switch, if it contains 'On', then proceed.

ii. Call the function you created in 3.C.c to get the desired color spec.

iii. Plot the data using the color and markers specified by the user. Remember that you need to specify the axes to plot on as the first input argument to the plot command (ex. plot(app.UIaxes,…).
iv. Use the hold command, using the axes name and 'on' as input arguments.
v. Get the color for the plot from the pulldown menu for that data series.

e. Set the ylimits and turn the hold off

D. Saving the Data (5 Pts)

a. Create a callback for the save data button similar to steps 3.B.a and 3.C.a (Figure 4).
b. Create a variable that contains a string array with the labels for the data to be save (time, temperature, and humidity).
c. Save this variable to an Excel file using the xlswrite command.
d. Save the numeric data to the same Excel file using xlswrite starting in the row under where the headers were written to.

Revision	Description	Date
A	Original Document	4/6/2021