Custom Program Instructions

Functions in class MyForm

- *__init__(parent = None):* Setting up connections between widgets and functions, or thread and functions. All the visa address are set to be None and the comboBoxes are updated. All the curve widgets are set. Each input string is appended into a list in order, then all the items will be printed and shown in the textEdit box. Essential static variables used in MyForm class are initialized in this function.
- *start_font(status):* This function changes the texts of the pushButton_Start. If the input is string “C”, the pushButton is going to be set to “Check Array”. And if the input is string “S”, the pushButton will be set to “Start”.
- *Update(signal, comboBoxVisa):* Update the exact visa comboBox according to the input signal. For example, if the input signal is “visa1”, then the first visa comboBox will be updated to the latest visa addresses.
- *Select(self, signal, visa_chosen, comboBoxVisa, lineEditVisa, selectClose, baud = None):* Select the exact visa address according to the input signal. The lineEditVisa shows the visa name if the selection is successful, and the selectClose button will be enabled. The baud is default to be None for devices such as Yokogawa and Keithley. However, for some special devices, such as Magnet Model 430, the baud should be set to 115200, otherwise the visa connection will be failed.
- *Ready(inst):* Check if the input visa address has already been chosen or not. If the address has been selected, the function returns False. And if the address has not been selected yet, the function will return True. Here I’m using True and False to represent the visas’ status. They are stored in a two dimensional list, with the corresponding visa names, such as [[visa1, True], [visa2, True], [visa3, False]]. True means that the visa has not been selected while False means the visa has been selected. The purpose of using the visa status list is to avoid our user select the same visa addresses more than once.
- *Mark(inst, bol):* Set the input visa status to True or False (depends on variable bol). For example, the list of visa status is [[visa1, True], [visa2, True], [visa3, True]]. And the function self.Mark(visa1, False) is going to set the status of visa1 to be False. The list then becomes [[visa1, False], [visa2, True], [visa3, True]].

- *Close(signal, visa_chosen, lineEditVisa, selectClose):* Close the certain visa address according to the signal. The lineEditVisa will be cleared (be set to empty string) if the visa is successfully closed.
- *Check(inst):* Check if the visa address is valid or not.
- *make_curveWidgets(curvewidget, color, makerColor, titles):* Set up the curve widget by various parameters, for example color, marker, X-axis name, Y-axis name, plot title.
- *array_builder(start, stop, step):* Return the array list by the input start, stop and step values. If the start value is less than the stop one, the array will be set to be decreasing.
- *start(self):*
    - The button shows “Start”: Generate a thread and send essential values to the Collect_data class to run the measurement. Before sending the values, we first organize them using lists. For example, “curves” is the list including all the curves; “curveWidgets” is the list containing curvewidgets. “inputted_data” saves the string format of value entered by the user, separated by each device. “device_state” saves the status of each device. For example, device_state = [True, False, False] means that only device1 is doing measurements. “array_M” saves the magnet array, similar with array_1, array_2 and array_3. These four arrays are saved together in a big list called “array_lst”. After the organization there are many if statements to make sure the user selects the dynamic saving correctly. The values will be sent to the next class if all the statements are satisfied.
    - The button shows “Check Array”: The magnet array and the other three arrays are shown in the preview tab. Then the button changes its name to “Start”.
- *Check_ready(count):* Return True if the user selects the correct number of devices. Return False otherwise. For example, self.Check_ready(1) means that only one device is doing measurements, and all the possible visa combinations are listed in the if statements. One of these statement has to be satisfied.
- *setup_plot(curve, data):* Curve plot a 1D data array.
- *setup_plot_2(curve_b, curve_r, data):* Curve plot two 1D data arrays together. One is black and the other is red.
- *curvePlots_update(curveInfo):* Update the curve plot.

- *Print_data(device, text):* Print out the measurement information for each device dynamically. The basic information includes total time, time step and magnet field. Input voltages, current and resistivity are shown for each device individually.
- *Switch_scale(num):* Return a list with the format of [scale, unit] according to the input num.
- *Pre_dsave():* Enable or disable the widgets in the dynamic saving tab.
- *Dsave_browse():* Browse the dynamic saving directory.
- *Dsave_filetype():* Select the dynamic saving file type.
- *Dsave_username():* Select the dynamic saving user name.
- *closeEvent(event):* Close the window with an extra message of "Do you want to quit the program?"

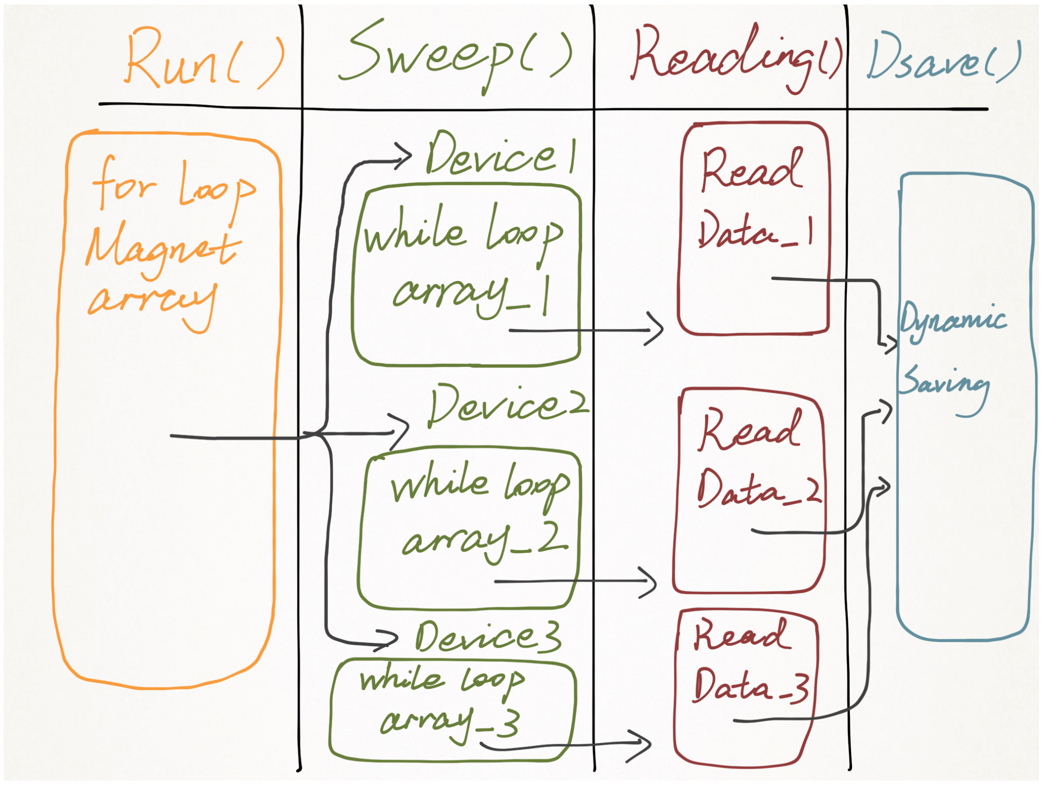
Functions in class Collect_data(QThread)

- *input(ui, instruments, curves, curveWidgets, go_on, inputted_data, array_lst, device_state, count, dsave, input_data):* Receive the data sent from the start function in the previous MyForm class. Initialize all the necessary static variables used in the functions below and set up the curve plot. In the end, self.start() will automatically jump to function run().
- *stop():* Set self.stop_collecting to be True. The main while loop in the function run() keeps checking self.stop_collecting. Once it is True, the loop holds on until it turns to be False or the loop is paused.
- *pause():* Break the main loop in the function run().
- *run():*
  - *Yokogawa_Magnet:* Using the Yokogawa to control the magnet field. The for loop goes through the required magnet sweep array.
  - *Magnet:* Directly using the American Magnetics to output the magnet field. The for loop goes through the required magnet sweep array.
  - *Non-magnet:* Skipping the magnet sweeping.

- *Sweep(input_M, Round):* Under each step of magnet field, the three devices go through their individual sweeping arrays. The sweeping arrays for three devices may not be the same, which means one of it may be shorter or longer than the other. Therefore, in each of the while loop, we need to make sure that the device which has the longest sweeping array is able to complete its measurement. And the devices which have the short arrays have to wait for the longest one to be complete.

- *Reading(device, input_volt, input_M, num, Round):* This function reads back the data from the Yokogawa and Aglient. It also do simple analysis on the raw data and save them in order (in the list). The curve plot gets update according to the list. The data is also sent back to the Print_data function in the My_Form class to show the data on screen dynamically. In the end, it sends essential data to the Pre_dynamic_save function.

- *Pre_dynamic_save(self, num, is_last, data_lst, i, Round):* Receive the data from the Reading function. This data will be written to the file dynamically. The biggest advantage is that the data will be saved in the file whenever the program crashes or the user exits it. The script is in the Sub_Scripts folder.



Relationship of Run, Sweep, Reading and Pre_dynamic_save functions

- *setup_plot(self, curveWidget, curve, data, titles):* Send a thread called “curve_plot” back to the My_Form class and trigger the curvePlots_update function, which is also in the My_Form. The plot command is put in the main class, in order to increase the speed and efficiency of the program.