NANOIMPRINTER SOFTWARE GUIDE

By: Yu Dong (Peter) Feng, Dylan Vogel

Summer 2017

TABLE OF CONTENT

1. Introduction

This document is designed to improve the user's understanding of the software developed for the Nanoimprinter. This document will teach the user how to use the software to run heating and temperature tests as well as provide the background information necessary for the users to modify the program.

2. The Setup

The software is written for Python 3 and the code should be run using a Raspberry Pi. The software also utilizes many python libraries, such as the PyQt 5. These libraries have already been installed on the Pi.

3. File Descriptions

There are 9 files created for the Nanoimprinter so far. This section will be providing a description of the purposes for each file.

3.1. heaterGUI.ui

This is a UI file that controls the design of the GUI. This file can be opened and modified with the Qt Designer program, which should come with the installation of the PyQt library. In Qt Designer, the user can add/remove features, such as buttons and text boxes to the GUI. The user can also change the layout of the GUI.

The UI file can be called in another Python file, where the actions and the processes for the GUI can be defined. For the Nanoimprinter software, the UI file is called in the "controller.py" file. Please refer to section 3.2 for more information about the "controller.py" file.

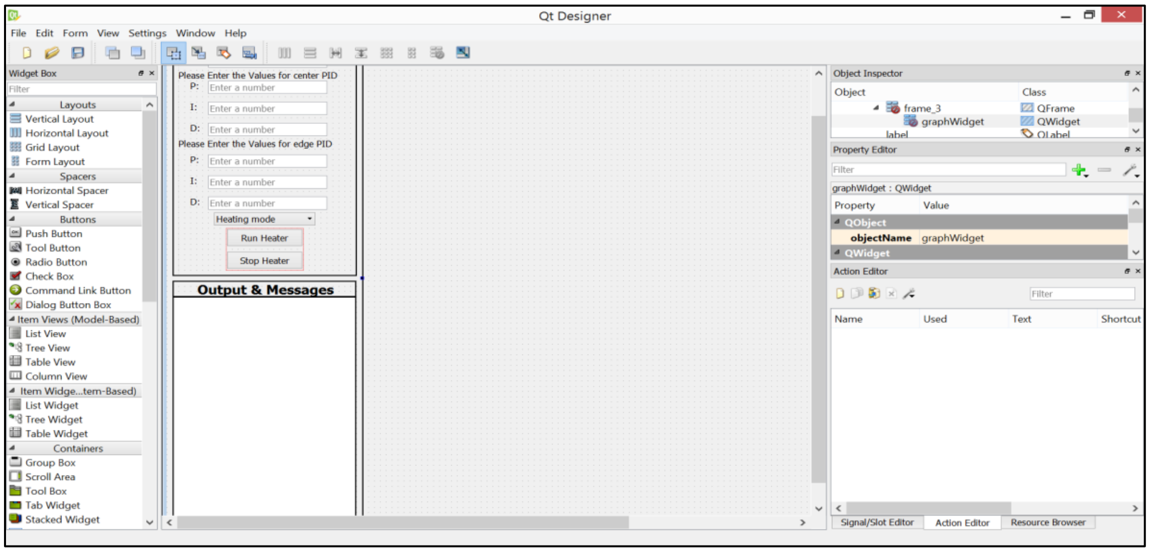


Fig 3.1-1. *"heaterGui.ui" loaded in Qt Designer.*

3.2. controller.py

This file contains the “MainWindow” class, which is a class created for the main window of the GUI. The “MainWindow” class contains functions that define the actions of the GUI. For example, the “handleRun” function contains the code that will be run when the “Run Heater” button has been pressed in the GUI. Some of the code for controlling the PID is also in the “handleRun” function.

3.3. dataLogClass.py

As the name suggests, the file is created for the “dataLog” class. This class is in charge of creating the folder, which will store the data files from each heating test. The “dataLog” class also contains functions that are designed to create & update the data files and the graphs for the heating and the temperature tests. The code that controls the information displayed in the “Output & Messages” textbox is contained in the “dataLogClass.py” file as well.

3.4. heater.py

The “heater” file sets up the GPIO and contains the code that determines the initial heating time for the heating tests. The code for changing the duty of the heater is contained in the “heater” file as well.

3.5. heatingProcess.py

This file is for the class of the same name. The “heatingProcess” class is designed to contain the variables and the set-up functions relevant to running tests. Unlike its name suggests, the “heatingProcess” class is not only used for the heating tests but also for temperature tests as well.

3.6. PID.py

This file contains the “PID” class. The “PID” class contains the functions that help to control the PID values. All the code in this file is from an online source. The author of the code for this file is Caner Durmusoglu.

3.7. pid_setup.py

This file contains the functions that are used for setting up the PID values of the edge and center thermocouples.

3.8. thmcouple.py

The functions for setting up and reading the temperatures of the thermocouples are in this file.

3.9. Main.py

This file is used for running the program.

4. GUI

A GUI has been developed to help user monitor the relevant data during heating and temperature tests. There are currently two modes that the GUI runs in: Heating Mode and Temperature Mode. The mode can be changed through the combo box in the GUI.

NOTE: The user should only change the mode before a test has been started or after a test has been stopped.

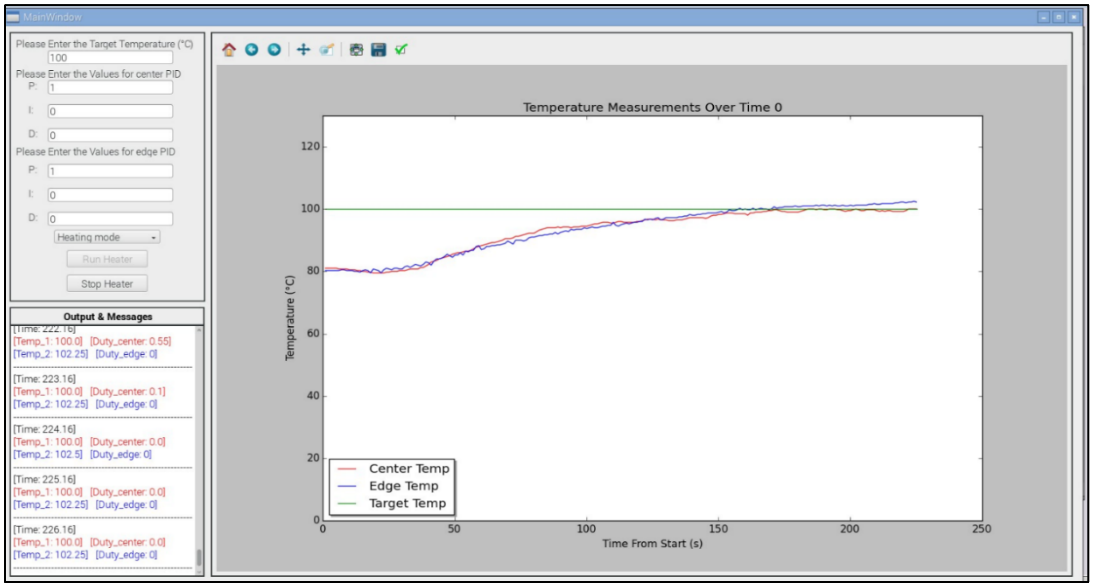
4.1. Heating Mode

In Heating Mode, the user is required to enter a target temperature for the Nanoimprinter to heat up to. The user also needs to enter the initial PID values for the center and edge thermocouples.

Currently, a cap of 200 degrees Celsius has been set for the target temperature. If a user tries to run a heating test with a target temperature above the cap, the program will generate a warning. The cap can be removed/changed in the "checkTemp" function of the "controller.py" file.

After valid temperature and PID values have been entered, the user can start a heating test by clicking the "Run Heater" button. Once the heating test has been started, the current temperatures and duty cycles of the two thermocouples will be in the "Output & Messages" box. A temperature vs time graph will also appear on the GUI and will be updated constantly. The graph contains a green line indicating the target temperature to help the user better visualize how close/far is the Nanoimprinter away from the target temperature.

At any point during the heating test, the user can stop the heating test by clicking the "Stop Heater" button. A picture of the GUI running the heating test is shown below.



Please Enter the Target Temperature (°C) 100
Please Enter the Values for center PID
P: 1
I: 0
D: 0
Please Enter the Values for edge PID
P: 1
I: 0
D: 0

```
[Time: 222.16]
[Temp_1: 100.0]  [Duty_center: 0.55]
[Temp_2: 102.25]  [Duty_edge: 0]
------------------------------------------
[Time: 223.16]
[Temp_1: 100.0]  [Duty_center: 0.1]
[Temp_2: 102.25]  [Duty_edge: 0]
------------------------------------------
[Time: 224.16]
[Temp_1: 100.0]  [Duty_center: 0.0]
[Temp_2: 102.5]  [Duty_edge: 0]
------------------------------------------
[Time: 225.16]
[Temp_1: 100.0]  [Duty_center: 0.0]
[Temp_2: 102.25]  [Duty_edge: 0]
------------------------------------------
[Time: 226.16]
[Temp_1: 100.0]  [Duty_center: 0.0]
[Temp_2: 102.25]  [Duty_edge: 0]
------------------------------------------
```

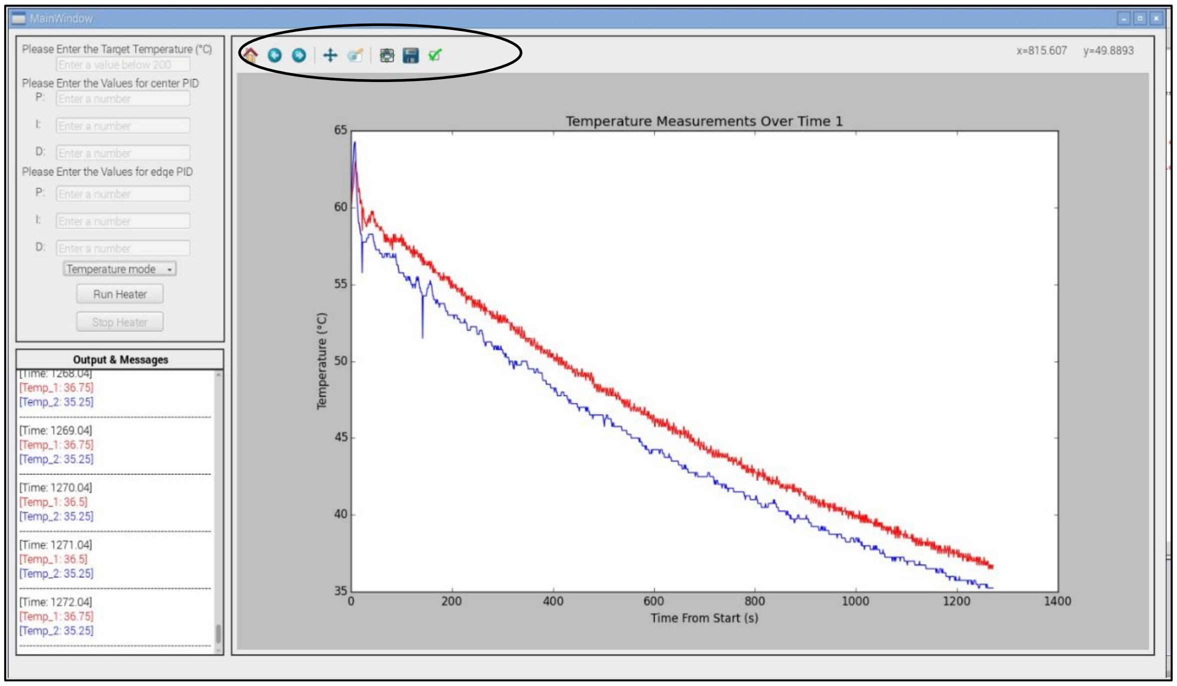
Fig 4.1-1. *The GUI running a heating test.*

4.2. Temperature Mode

In Temperature Mode, the user doesn't have to enter any thing so all the input boxes are inactivated. To start running a temperature test, the user just needs to click the "Run Heater" button.

During a temperature test, the current temperatures of both thermocouples will be displayed in the "Output & Messages" box as well as on a temperature vs time graph. The user can stop the temperature test at any time by clicking the "Stop Heater" button. A picture of the GUI running the temperature tests can be seen below.

NOTE: For both modes of the GUI, the user should use graph features (circled in the figure below) such as zoom only AFTER the tests have been stopped.



```
[Time: 1268.04]
[Temp_1: 36.75]
[Temp_2: 35.25]
-----------------------------------
[Time: 1269.04]
[Temp_1: 36.75]
[Temp_2: 35.25]
-----------------------------------
[Time: 1270.04]
[Temp_1: 36.5]
[Temp_2: 35.25]
-----------------------------------
[Time: 1271.04]
[Temp_1: 36.5]
[Temp_2: 35.25]
-----------------------------------
[Time: 1272.04]
[Temp_1: 36.75]
[Temp_2: 35.25]
-----------------------------------
```

Fig 4.2-1. *The GUI running a temperature test.*

5. HEATING PROCESS

The current Nanoimprinter utilizes 10 heating cartridges. The heating of the Nanoimprinter has been divided into central and edge heating. The heating process has also been divided into various stages.

5.1. Initial heating

The heating process start off with the initial heating stage. In this stage, the software first estimates a suitable initial heating time using the formula: $heating\ time = \frac{m*c*\Delta T}{P}$ (where "m" is the mass being heated, "c" is the heat capacity, "ΔT" is the change in temperature, and “P” is the power supplied to the heaters). The initial heating algorithm can be found in the “heater.py” file.

After initial heating time has been estimated, the software sets the duty values of the central heating cartridges and the edge heating cartridges to predetermined values. The current initial duty values are 20 for the edge cartridges and 100 for the central cartridges (These values can be changed in the “heatingProcess.py” file). The central duty value is set to max to allow for faster initial heating and thereby reducing the overall heating time. The edge duty value is set relatively low to account for the fact that there will be heat conducted from the center to the edge.

5.2. Waiting

After initial heating has been completed, the software stops supplying power to the heating cartridges for a predetermined length of wait time by setting the duty values of all the heaters to 0. This is done to allow heat to conduct to the surface of the sample mounting plate, which will provide thermocouples with readings that are more reflective of the actual temperature. The current wait time is set to 30 seconds. It can be changed in the “heatingProcess.py” file.

5.3. PID control

Once the temperature of the Nanoimprinter has settled, the PID controller will start running. When the PID controller is running, the software will constantly update the PID values based on current and past temperatures. The current algorithm that is used to determine the PID values (located in the PID.py file) is taken from an online source.

5.4. PID Suppression

When the temperature of the Nanoimprinter is close enough to the target temperature, the edge and center PID values are supressed to predetermined values, which are very low. This helps to ensure that the temperature of the Nanoimprinter approaches the target temperature more slowly and prevents the temperature from overshooting/dipping from the target temperature too much.

Currently, the center and edge P values are suppressed when the Nanoimprinter temperature is within 15 degrees Celsius of the target temperature. The suppressed P value is set to 1. The center and edge I and D values are suppressed when the Nanoimprinter temperature is within 1 degree Celsius of the target temperature. The suppressed I and D values are both set to 0.5.

The temperature range within which PID values become suppressed can be changed in the “controller.py” file. The suppressed PID values can be changed in the “heatingProcess.py” file.

6. TEST DATA

Test data from heating and temperature tests are automatically saved as a text file in a folder with the corresponding date after every test. The graphs from the tests are saved automatically as well.

The locations and the formats for the text files and the graphs saved can be changed in the “dataLogClass.py” file. The picture below shows some examples of saved graphs and folders containing saved data files.

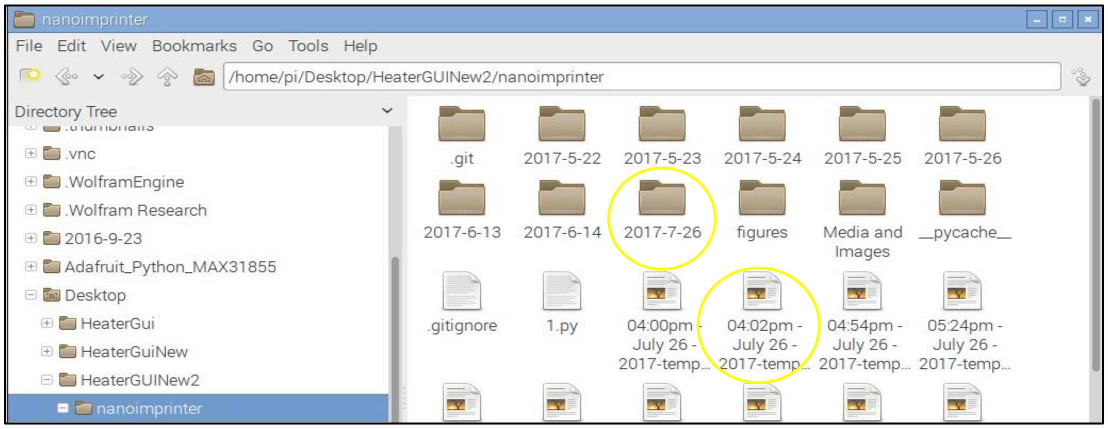


Fig 6-1. Examples of saved graphs and folders containing saved data files are circled in the figure.

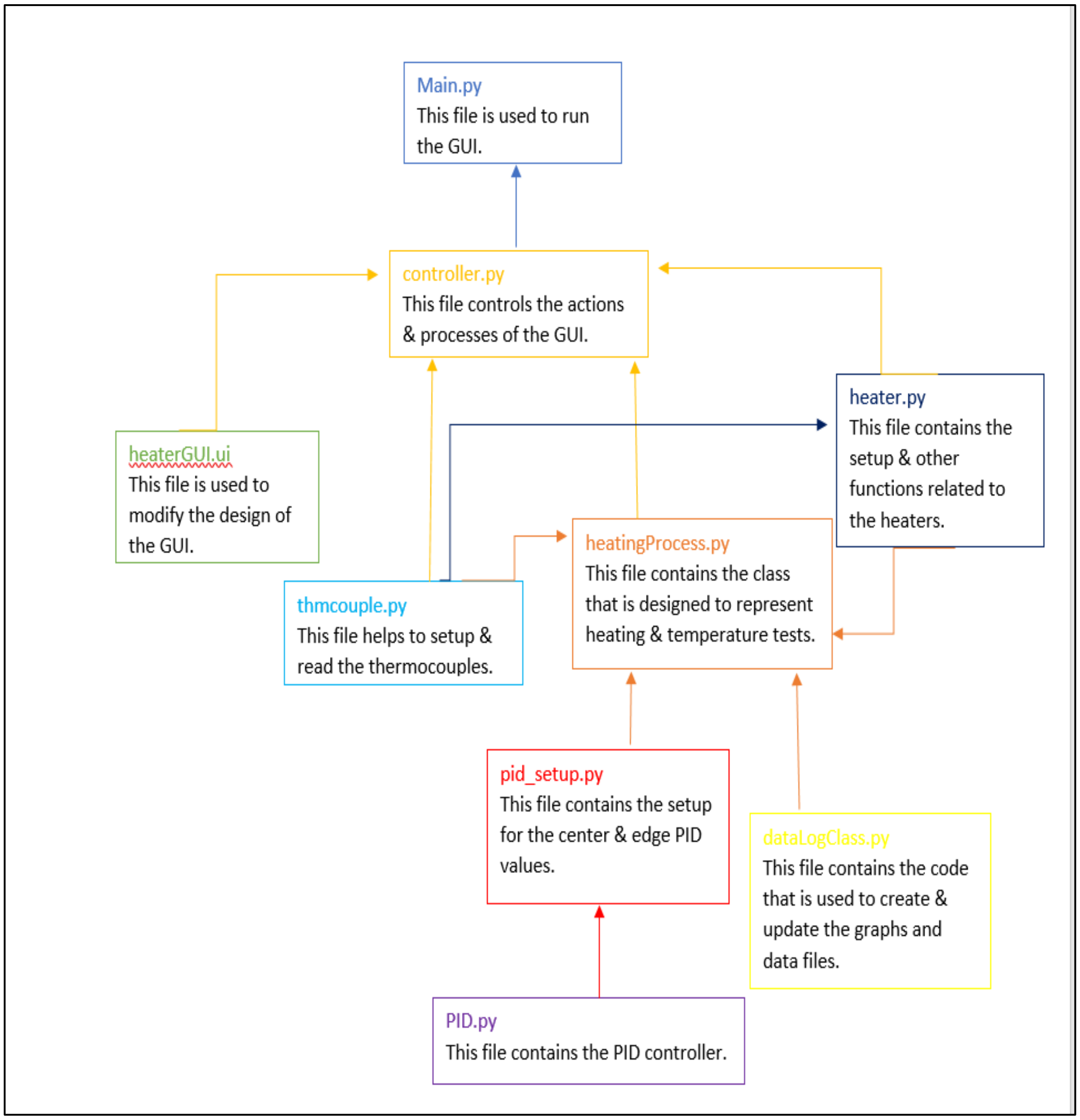
7. CODE INDEX

This section outlines which files should the user modify for several important features of the software.

Features	Associated Files
GUI design	heaterGUI.ui
GUI actions	controller.py
GUI input (temperature, PID) control	controller.py
Thermocouple set up and reading	thmcouple.py
Heating and temperature test parameters, such as data log frequency and heater waiting time	heatingProcess.py
Initial heating algorithm	heater.py
PID set up	pid_setup.py
PID control	PID.py
PID suppression range	controller.py
PID suppression values	heatingProcess.py
Creating, updating, and saving graphs	dataLogClass.py, controller.py
Creating, updating, and saving test data	dataLogClass.py, controller.py

8. CODE MAP

This section contains the code map that shows the relationships between all the files.


Main.py
This file is used to run the GUI.
controller.py
This file controls the actions & processes of the GUI.
heaterGUI.ui
This file is used to modify the design of the GUI.
heater.py
This file contains the setup & other functions related to the heaters.
thmcouple.py
This file helps to setup & read the thermocouples.
heatingProcess.py
This file contains the class that is designed to represent heating & temperature tests.
pid_setup.py
This file contains the setup for the center & edge PID values.
dataLogClass.py
This file contains the code that is used to create & update the graphs and data files.
PID.py
This file contains the PID controller.

9. CURRENT LIMITATIONS & FUTURE IMPROVEMENTS

There are several limitations with the software.

1. The Pi sometimes doesn't display the correct date & time and this can affect the data and graph saved after each test as their names contain date/time. The temporary fix is to check if the Pi is displaying the correct date & time every time it's turned on. If the Pi is not displaying the correct date & time, change to a different time zone and then switch back to Singapore time zone. This should result in the correct date & time most of the time. It may also help to ensure that the Pi is connected to the internet through the ethernet cable rather than through wifi.
2. Currently, there is a "bad file descriptor" bug that appears when a heating test is stopped. This bug requires the GUI to be restarted for each heating test. This bug doesn't seem to be causing any problems other than the restarting of GUI. The bug appears for heating tests only. Temperature tests are not affected.

In addition to eliminating the above limitations, several other improvements can be made to the software.

1. Improve the initial heating algorithm more.
2. Improve the PID control algorithm more.
3. Make the GUI automatically adjust to resizing.

10. RESOURCES

Listed below are links to several useful resources.

1. Qt Designer Manual
    - http://doc.qt.io/qt-5/qtdesigner-manual.html
2. Qt Signals & Slots
    - http://pyqt.sourceforge.net/Docs/PyQt5/signals_slots.html
3. Python Classes
    - https://docs.python.org/3/tutorial/classes.html