METHODOLOGY

1 Temperature Control

Temperature of the system is proposed to be controlled by either bang-bang control method or PID control method. In order to design the control system first the plant must be modelled accurately. The alternative method is to tackle the problem head on and manually calibrate the PID parameters which might work but is more time consuming and requires more manual labor.

Hence the first step to design a control system is to know the dynamic behavior of the system and to construct the transfer function that accurately represents the real system.

The first method is to derive using first principles and using underlying physics behind it which might be complicated hence using its alternative i.e. data driven method should be more appropriate. The necessary steps to extract transfer function can be listed below.

1. **Gathering of data**
    a. Finding the open loop response of the system. In this project input is the step function as input is given by a switching relay which is either on or off i.e. high or low whereas the output is the temperature of the plant.
    b. The data should ideally cover the entire operating range of the system to accurately capture the dynamics.

2. **Importing Data into MATLAB**
    a. Load the captured data into MATLAB. The data can be in the format of csv, xls, or. mat.
    b. Creating iddata object in order to organize the data into input output and sampling time.
    data = iddata(output_signal, input_signal, sampling_time)

3. **Preprocessing the Data**
    a. If the data has a trend or offset, you can remove it using the detrend function.

4. **Split Data into Estimation and Validation Sets**
    a. It is ideal to split the data into two sets: one for estimating the model and another for validating it.
    [data_est, data_val] = split(data);

5. **Choosing a Model Structure**
    a. Selecting a Transfer Function Model i.e. order of the transfer function, number of poles and zeros.

6. **Estimating the Transfer Function**
    a. The tfest function will estimate the transfer function based on the input-output data.

7. **Validating the Model**
    a. Using the compare function to validate the estimated model against the validation dataset.
    b. compare (data_val, sys);
    This function plots the measured output against the model's output to visually assess the model's accuracy.

Alternatively, one can use the System Identification Tool GUI in the command window and interactively selecting a transfer function model.

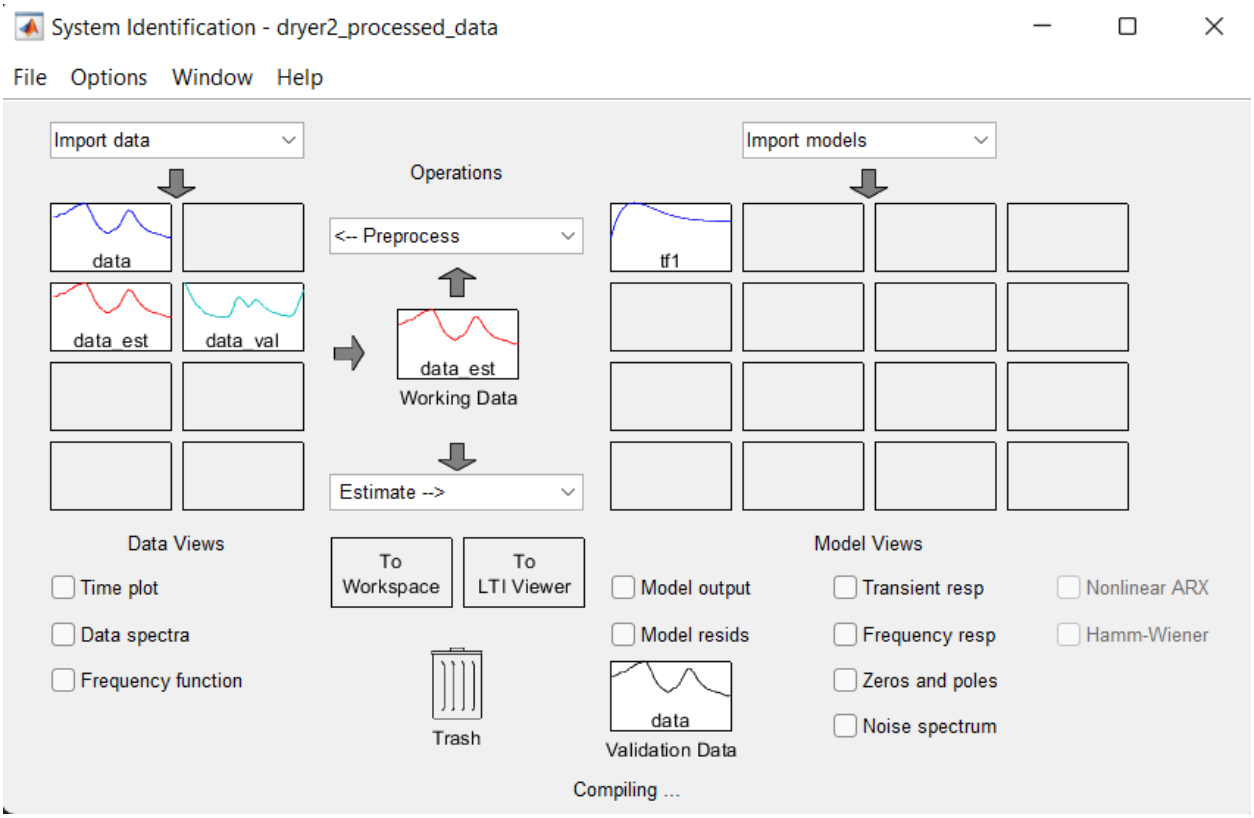

Figure 1-1: System Identification Toolbox

2 Phase Protection

Having simple interface and noise reduction mechanism using AC voltage sensor module ZMPT101B probably might be the best option. The ZMPT101B module is a voltage sensor designed for accurate AC voltage measurement. It features a ZMPT101B voltage transformer and an operational amplifier circuit that amplifies the small AC signals. The module includes a potentiometer to adjust the gain of the amplifier, allowing for precise scaling of the output voltage. This output, which represents the AC voltage, can be connected to an analog-to-digital converter (ADC) on a microcontroller for further processing. Typically powered by a 5V supply, the ZMPT101B module is used in applications such as phase detection, RMS voltage measurement, and energy monitoring systems.

Following algorithm is proposed to detect phase sequence and rms value of each phase.

1. Initialization
    a. Analog pins for each phase are assigned (e.g., Phase A, Phase B, and Phase C). A threshold value is set, typically at the midpoint of the ADC range (e.g., 2048 for a 12-bit ADC).
    b. Variables are initialized to keep track of previous analog values, accumulated sums of squares, sample counts, and zero-crossing flags for each phase.

2. Continuous Monitoring (Main Loop)
    a. Continuously sample the analog input signals from each phase.

3. Phase Detection
    a. For each phase (A, B, and C) Reading the current analog value from the analog input pin.
    b. Determine if a zero crossing has occurred by comparing the current value with the previous value:
    c. If a change in sign across the threshold is detected, update the zero-crossing flag for that phase to true.
    d. If the zero-crossing flag is not set:
    e. Accumulate the square of the current analog value into the sum of squares for that phase.
    f. Increment the sample count for that phase.
    g. Update the previous analog value to the current value.

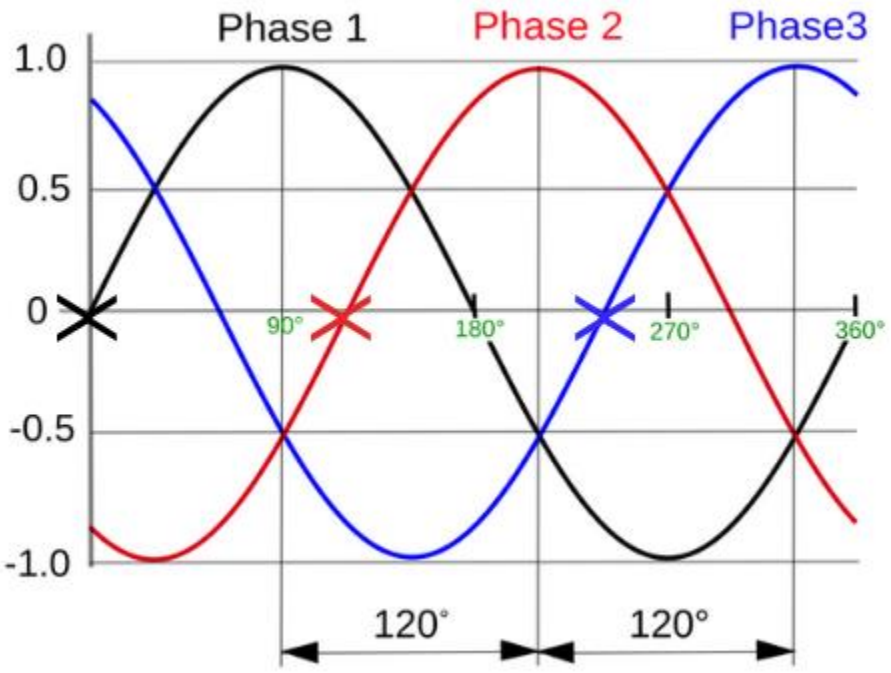

Figure 2-1: Zero Crossing of Each Phases

4. RMS Calculation:
    a. For each phase (A, B, and C). If the zero-crossing flag is true computing RMS as square root average of sum of squares.