SIMON FRASER UNIVERSITY
School of Mechatronic Systems Engineering
MSE 481 - Industrial Control Course Project
I Summer 2019

Project Grading:

This project is worth 15% of the final course grade. Plagiarism will result in a grade of 0%. **Here tasks are expressed as steps.**

The following grading rubric will be used: (graded out of a total of 100).

- Final Report (must be including Title Page with Student Names, Student Numbers, etc.) **50%**
  Report Quality must include the following:
  - Introduction and Description of System (Problem statement) 5%
  - Description of Programming Objectives (Steps 1,3,4,5; 2.5% each) 10%
  - Description of Program Design (Steps 1,3,4,5; 2.5% each) 10%
  - Printouts of Ladder Logic Files, including all Comments (Steps 1,3,4,5; 2.5% each) 10%
  - Results and Conclusions 15%

- Correct/verified operation of PLC ladder logic (Steps 1,3,4,5; 5% each step) **20%**
- Clarity and Completeness of comments in ladder logic ((Steps 1,3,4,5; 5% each step) **20%**
- Video Demonstration **10%**

Project Group Selection:

A total of 36 project groups have been established in Canvas. Maximum of 2 students per group is allowed. Students will form project groups independently, and self-assign to the corresponding group using Canvas.

Project Deliverables:

1. Project group selection (due **June 26th**, selected using Canvas)
2. Submission deadline is **July 7th, 2019**, **late submission will get 10% of total mark deducted**.
3. Submit your project on **CANVAS** as follows:

- Documentations
  - PDF Documents
- Project files
  - PLC ladder logic (.rsl files)
- Video Demonstration (wmv/mp4 etc. files)

SIMON FRASER UNIVERSITY
School of Mechatronic Systems Engineering
MSE 481 - Industrial Control Course Project
I Summer 2019

Project Description:

In this project, you will design and develop a PLC (Ladder Logic) program to regulate the Temperature of a continuous stirred-tank reactor (CSTR) shown in Figure 1. Two control strategies are used in this project including ON/OFF control and PID control. The main objective purpose of this project is to design and implement these two strategies in LogixPro and compare the results qualitatively.

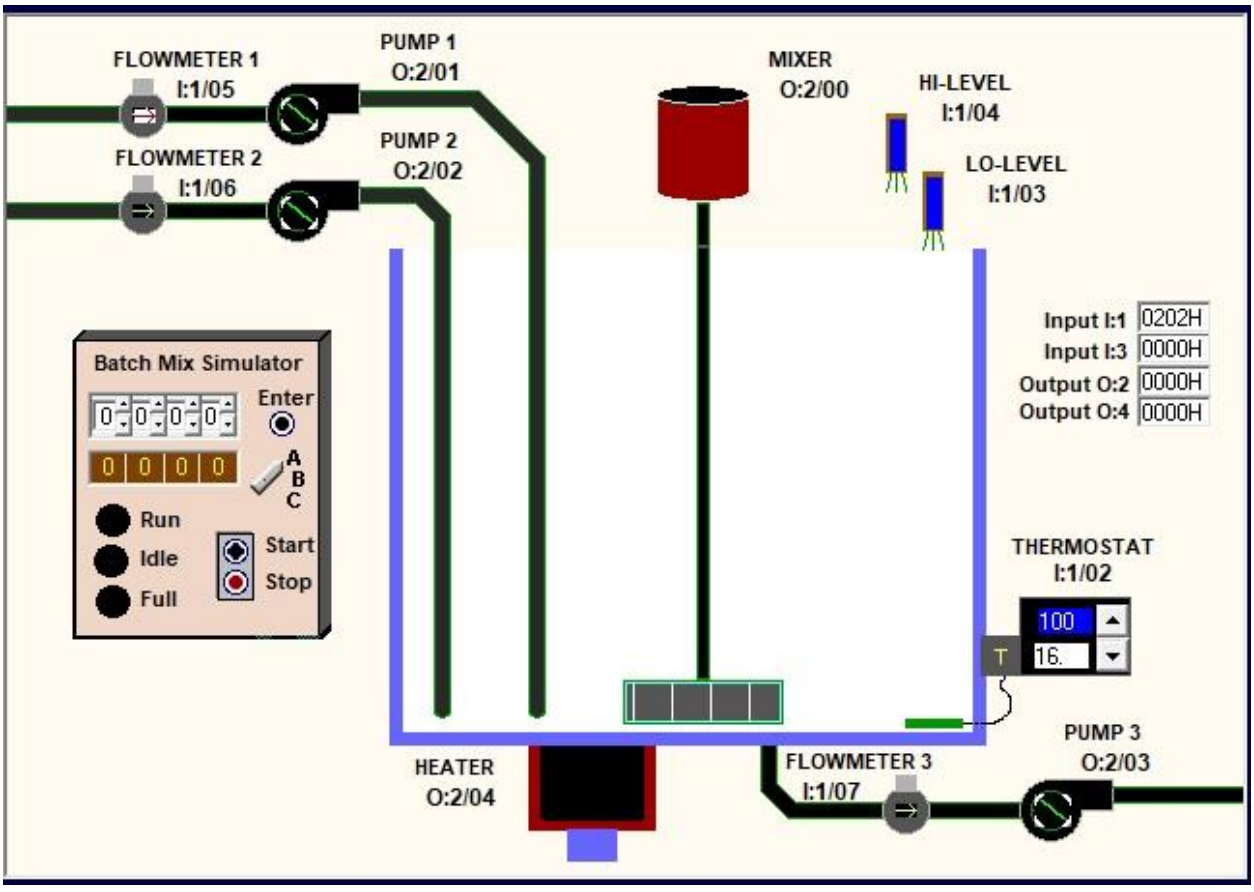

Figure 1. : CSTR simulator (screenshot from LogixPro software)

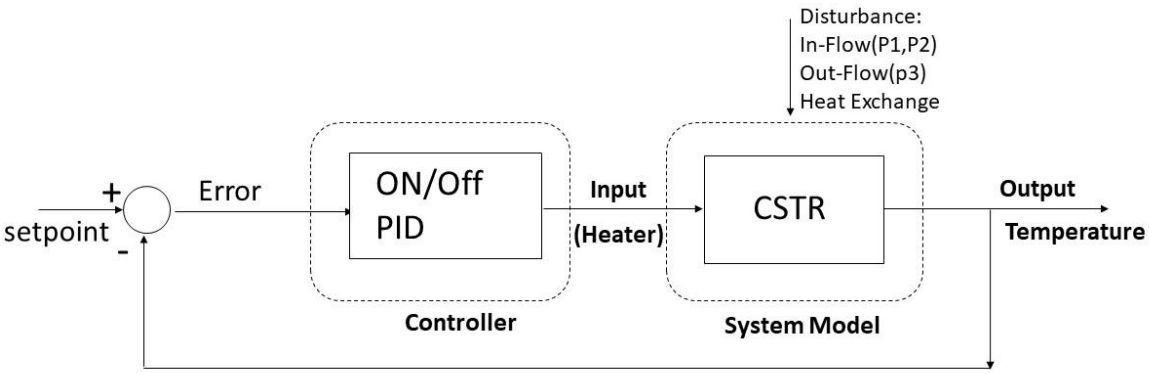

Figure 2. : The system model and control loop

List of Inputs and Outputs:

I:1/00	= Start Pushbutton	(Input)
I:1/01	= Stop Pushbutton	(Input)
I:1/02	= Thermostat	(Input)
I:1/03	= LO-LEVEL sensor	(Input)
I:1/04	= HI-LEVEL sensor	(Input)
I:1/05	= Flowmeter 1	(Input)
I:1/06	= Flowmeter 2	(Input)
I:1/07	= Flowmeter 3	(Input)
I:1/08	= NO Pushbutton (black)	(Input)
I:1/09	= Selector Switch A	(Input)
I:1/10	= Selector Switch B	(Input)
I:1/11	= Selector Switch C	(Input)
I:3	= Temperature Setpoint (BCD)	(Input)
O:2/0	= Mixer	(Output)
O:2/1	= Pump 1	(Output)
O:2/2	= Pump 2	(Output)
O:2/3	= Pump 3	(Output)
O:2/4	= Heater	(Output)
O:2/5	= Run	(Output)
O:2/6	= Idle	(Output)
O:2/7	= Full	(Output)
O:4	= Temperature Reading (Estimation)	(Output)

Design Procedure:

Step 2 in this project is the estimation of temperature that has been developed before, and is given to you as "temperature_reading.rsl". Hence, add your designed program described in the other steps 1,3,4 and 5 to this existing program.

Step 1: Setting Up the Pumps and Mixer Operation

Design a program to meet the following requirements:

- When the selector switch (I:1/9) is in A position and the Start switch (I:1/0) is pressed, then pump P1 will be energized and the tank will start to fill. Also, the stop switch (I:1/1) is used, to shot off pump P1 and stop filling the tank. The filling should stop when the HI-LEVEL limit switch (I:1/04) indicates that the tank is full.
- When the selector switch (I:1/10) is in B position and the Start switch (I:1/0) is pressed, then pump P2 will be energized and the tank will start to fill. Also, the stop switch (I:1/1) is used, to shot off pump P2 and stop filling the tank. The filling should stop when the HI-LEVEL limit switch (I:1/04) indicates that the tank is full.
- When the selector switch (I:1/11) is in C position and the Start switch (I:1/0) is pressed, then pump P3 will be energized and the tank will start to drain. Also the stop switch (I:1/1) is used to shot off pump P3 and stop draining the tank. The draining should stop when the LO-LEVEL limit switch (I:1/03) indicates that the tank is empty.
- The mixer motor (O:2/00) should be steered while the tank is not empty, so that the temperature of the liquid is homogeneous everywhere.

Step 2: Simulating the temperature Reading (this step has been done before)

The temperature reading is not available in this simulator. On the other hand the objective of the project is to design a PID controller for the CSTR. Therefore, one should have the temperature for the tank. The temperature of the tank has been modeled using empirical approach and is given to you as a separate .rsl file, where it can be a template file that you can use and add your project on it. Please be advised that the generated temperature reading is an estimation and is not accurate.

Important Note: Throughout this whole project, you will disregard the temperature shown on the simulator's default display 

Instead, read the liquid temperature from the display O:4 (BCD) which is programmed in the .rsl file given to you. 

Due to modelling error the reading may be a bit off from the temperature shown by the simulator.

The obtained empirical model is based on the following observations:

1. Filling the tank using either pump P1 or P2 results in temperature reduction of 0.1 degrees with every 0.1 seconds that either of the pumps are on (t_{p1}, t_{p2}).
2. Emptying the tank using pump P3 results in temperature increase of 0.1 degrees with every 5 seconds that the pump is on (t_{p3}).
3. Turning the heater on causes a temperature increase of 1 degrees for each 0.2 seconds that the heater is on (t_h).
4. The heat exchange between the tank liquid and surroundings causes a decrease in temperature represented by the time passed in idle situation as a 0.1 reduction for each 10 seconds that passes in idle mode (t_k).

The result can be expressed as the following formula:

$$10T = -\frac{1}{0.1}\left(t_{p1} + t_{p2}\right) + \frac{1}{5}\left(t_{p3}\right) + 10\frac{1}{0.2}(t_h) - \frac{1}{10}\,(t_k) + 160 \qquad (1)$$

Note: The actual temperature (T) is multiplied by 10 because the available display (O:4 used to show the temperature does not support floating point, hence by this multiplication, for example the 16.1 temperature is shown 161 on the simulator screen.
The ladder rungs to achieve these specifications are made available to save time.

Step 3: Reading the temperature set point and calculating the error:

Design a program to meet the following requirements:

- Read the temperature setpoint from input (I:3) and store it somewhere in the memory. This input is BCD (binary coded decimal), so don't forget to convert it to the appropriate format. Remember that the input setpoint needs to be the desired setpoint multiplied by 10 to match the temperature reading

and it should not exceed the max temperature determined by (I:1/02) input or it causes system overheat and shutdown.

Important note: The internal overheat alarm in the simulator works with the simulator's default temperature display (not the estimated value designed in step 2). Therefore, to avoid activating the alarm and then plant shutdown, set the maximum temperature value to 100 and choose the simulator's scan time to a value between 50 to 100 scans (that is the number you see on the display not the one you see when hovering your mouse pointer over the scan speed adjuster).

- Calculate the temperature error using the temperature reading from step 2 and the setpoint from step 3.

Figure 3: setting the max temperature and the setpoint

Step 4: Controller Design:

a) ON/OFF Controller

With on/off controllers the final control element is either on or off, one for the occasion when the value of the measured variable is above the setpoint and the other for the occasion when the value is below the setpoint.

Design a program to meet the following requirements:

- If the liquid temperature goes below the setpoint (the error is greater than or equal to zero) and the tank is not empty, the heater is turned on.
- When the liquid temperature goes above the setpoint (the error is less than zero), the heater is shut off.

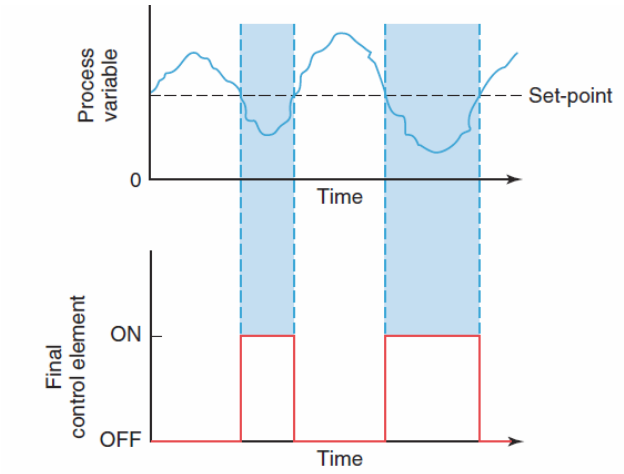

Figure 4: On/Off control action

Discuses about the advantage and disadvantage of this controller.

b) PID Controller

PID control is a feedback control method that combines proportional, integral, and derivative actions. The PID controller is the most widely used type of controller and can reduce the system error to zero faster than any other type.

The original PID control is designed to allow the final control element (here the heater) to take intermediate positions between on and off or analog control of the process depending on how much the value of the measured variable has shifted from the desired value (or the magnitude of the error signal). But what if the final control element is not an analog one and only capable of being turned on and off, is there still a way to utilize the useful properties of the PID controller?

The answer is yes, there is a way of implementing analog control actions by turning the final control element on and off for short intervals. This time proportioning (also known as pulse width modulation) varies the ratio of on time to off. So for example instead of turning the heater on to 50% power (which we are unable to do), we can turn it on to 100% power but for 50% of the time cycle. This will result in a much smoother faster and more accurate control action provided that the on/off cycles are not harmful to the physical components.

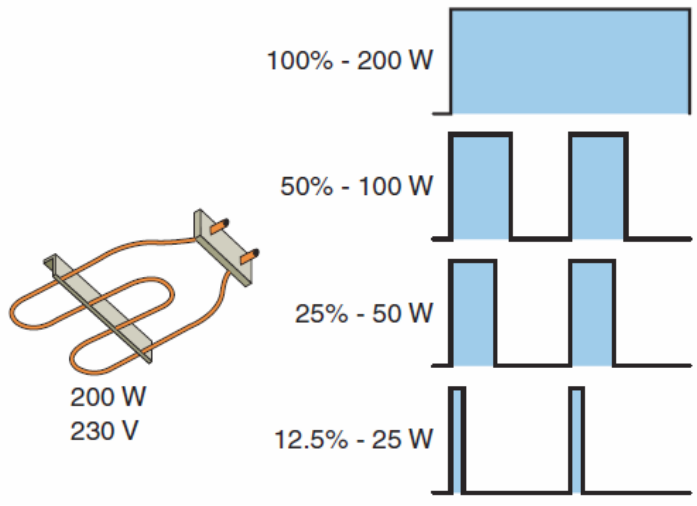

Figure 5: PWM (time proportioning)

Before designing the ladder logic for the controller, let us analyze some theoretical aspects of the control method:

Discretized (Sample based) PID formula:

This is the analog PID equation:

$$u(t) = K_c\left(e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_D\frac{de}{dt}\right) \quad (3)$$

This is not applicable in its current form as we are working with discrete samples rather than continuous signals. This equation corresponds to the following discrete or sample based equation called "the position form":

$$u(k) = K_c\left(e(k) + \frac{1}{T_i}\sum_{j=0}^{k} e(j) + T_D\frac{e(k)-e(k-1)}{\Delta t}\right) \quad (4)$$

The second term of the equation (4) accumulates the error values through time which could result in the integral windup problem, hence, the difference equation form of the PID called the "velocity form" is going to be utilized to avoid integral windup:

$$\Delta u(k) = K_c\left(e(k) - e(k-1) + \frac{\Delta t}{T_i}e(k) + \frac{T_D}{\Delta t}\big(e(k) - 2e(k-1) + e(k-2)\big)\right) \quad (5)$$

The equation (5) represents the change in the control signal in each step rather that the control command value itself.

Pulse Width Modulation (Time Proportioning)

To impose the PID controller signal $u(k+1) = u(k) + \Delta u(k)$ to the system a Pulse width Modulation signal (PWM) is generated and used. The PWM generates a time proportioning of the analog control command values to implement them using the on/off feature of the final control elements available to us in this setting.

Assume that the length of one cycle is S seconds. Also for simplicity, generate the control signals with 5 values between $0\% - 100\%$ including (0%, 10%, 25%, 50%, 100%). The time proportioned pulse corresponding to each value of the control signal is presented in the following Figure:

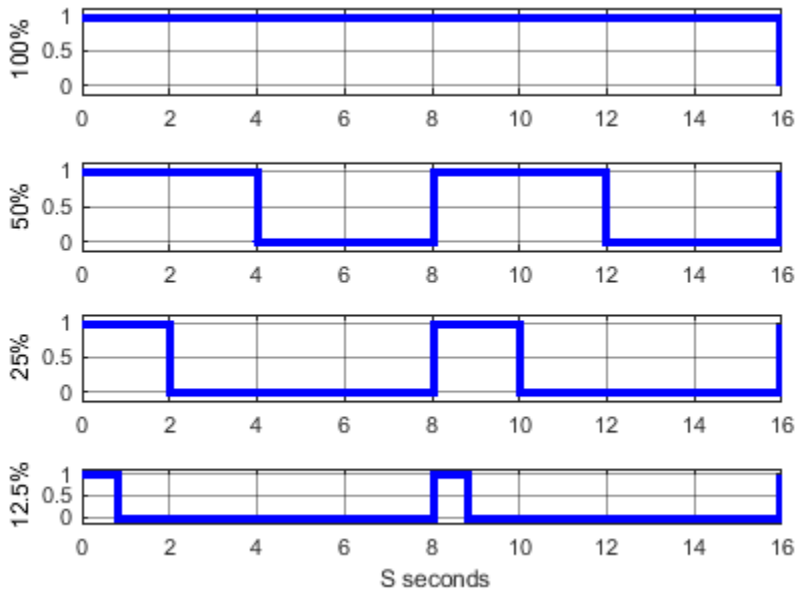

Figure 6: Time proportioning categories

To comply with the constraints of the simulation software, choose $S = 1.6\,seconds$ in your design choose. That is each partition represents $0.1\,seconds$ in your program which can be created using LogixPro.

The control signal u is defined as in the following:

$$0 < u < u_{max} \quad (6)$$

u_{max} is the maximum value that the PID controller can generate. The lower the liquid level is, the smaller the u_{max} is needed to heat it up. Simulations have shown that $u_{max} \approx 1000$ for all of the possible tank level. Then, the controller output is discretized as in the following to generate the percentage of the PWM signals as represented in equation (7) and Figure7.

$$\begin{matrix} u > 1000 \\ 500 < u \leq 1000 \\ 250 < u \leq 500 \\ 125 < u \leq 250 \\ 0 < u \leq 125 \\ u < 0 \end{matrix} \rightarrow \begin{matrix} 100\% \\ 50\% \\ 25\% \\ 12.5\% \\ 0\% \\ 0\% \end{matrix} \qquad (7)$$

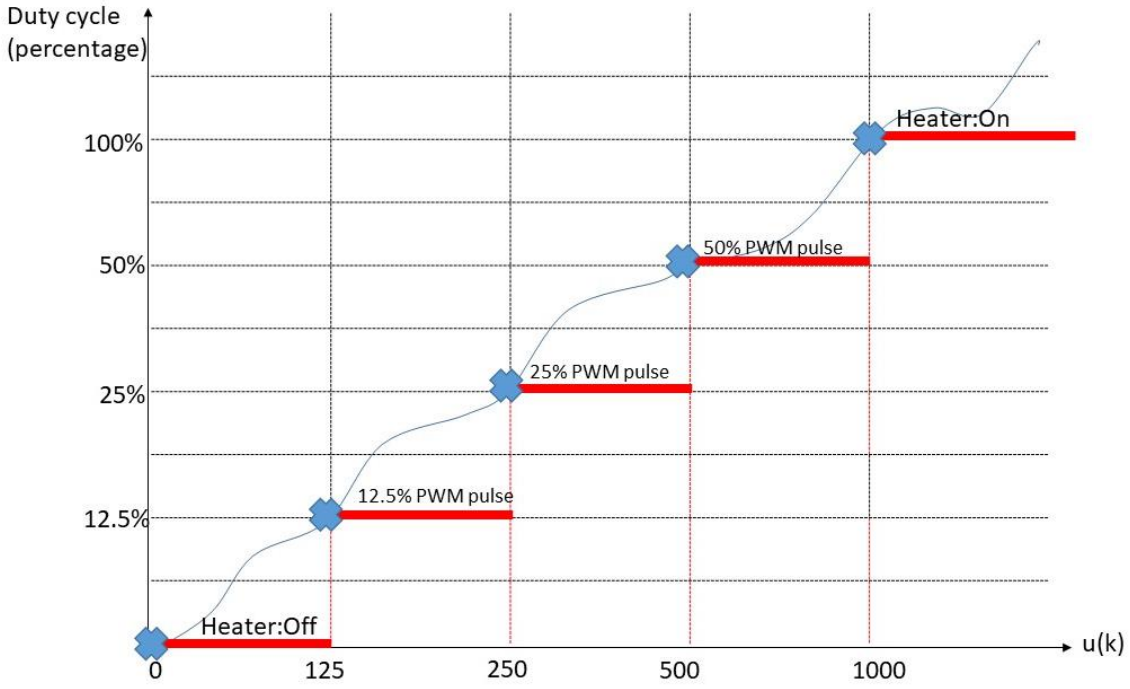

Figure 7: discretization of u(k)

To implement, design a program to meet the following requirements:

- Use the first scan (first pass) bit to initiate u at $u = 0$.
  (Hint: the first scan bit is S:1/15)

According to equation (5), three samples of the error signal are needed at each sampling time to calculate the change in control command at each scan. This means that to calculate $\Delta u(k)$ at the time of scan k, the error samples $e(k)$, $e(k-1)$, $e(k-2)$ are needed.

- Use the first pass bit to initialize $e(k) = e(k-1) = e(k-2) = 0$ in the first scan.
  (Hint: use float variables (F8:0=99) and MOV instruction to define variables and assign them values.)
- From the second scan forward, before updating each $e(k)$ that you have calculated in step 3, move the old $e(k)$ value to $e(k-1)$ and the older $e(k-1)$ to $e(k-2)$.
- Use equation (5) to calculate the increments in the control signal $\Delta u = \Delta u(k)$. Use $\Delta t = 0.1$ as it is the preset time step in all the timers that you will be using and start with the following PID tuning values:

$$k_c = 1, T_i = 0.01, T_D = 1 \quad (8)$$

later, change the PID tuning coefficients and observe and compare the results.

- Scale $\Delta u = \Delta u(k)$ by a factor of 0.1. Since the simulator cannot handle non-integer values, only factorize $\Delta u \geq 10$ values:

$$\begin{cases} \Delta u \geq 10 \rightarrow \Delta u = \Delta u/10 \\ \Delta u < 10 \rightarrow \Delta u = \Delta u \end{cases} \quad (9)$$

This is done to slow down the control process to smaller steps to avoid errors due to overcompensation with too big Δu values at each step.

- Update u using Δu from the previous step:

$$u = u + \Delta u \quad (10)$$

remember that the first value for u was initialized before.

- To prevent integral wind-up in negative control values due to accumulation, add a resetting step to reset u to zero if it is negative.

$$u \leq 0 \rightarrow u = 0 \quad (11)$$

- Write ladder rungs to detect which of the four categories in equation (7) u belongs to, then for each case, write a subroutine and use timers and latch and unlatch instructions to create the PWM-created pulses in Figure(6). (use $S = 1.6\, seconds$). Make sure that none of the control signals is implemented if the tank is empty.

The following figure illustrates the steps and how they are connected to each other and the order in which they need to be implemented.

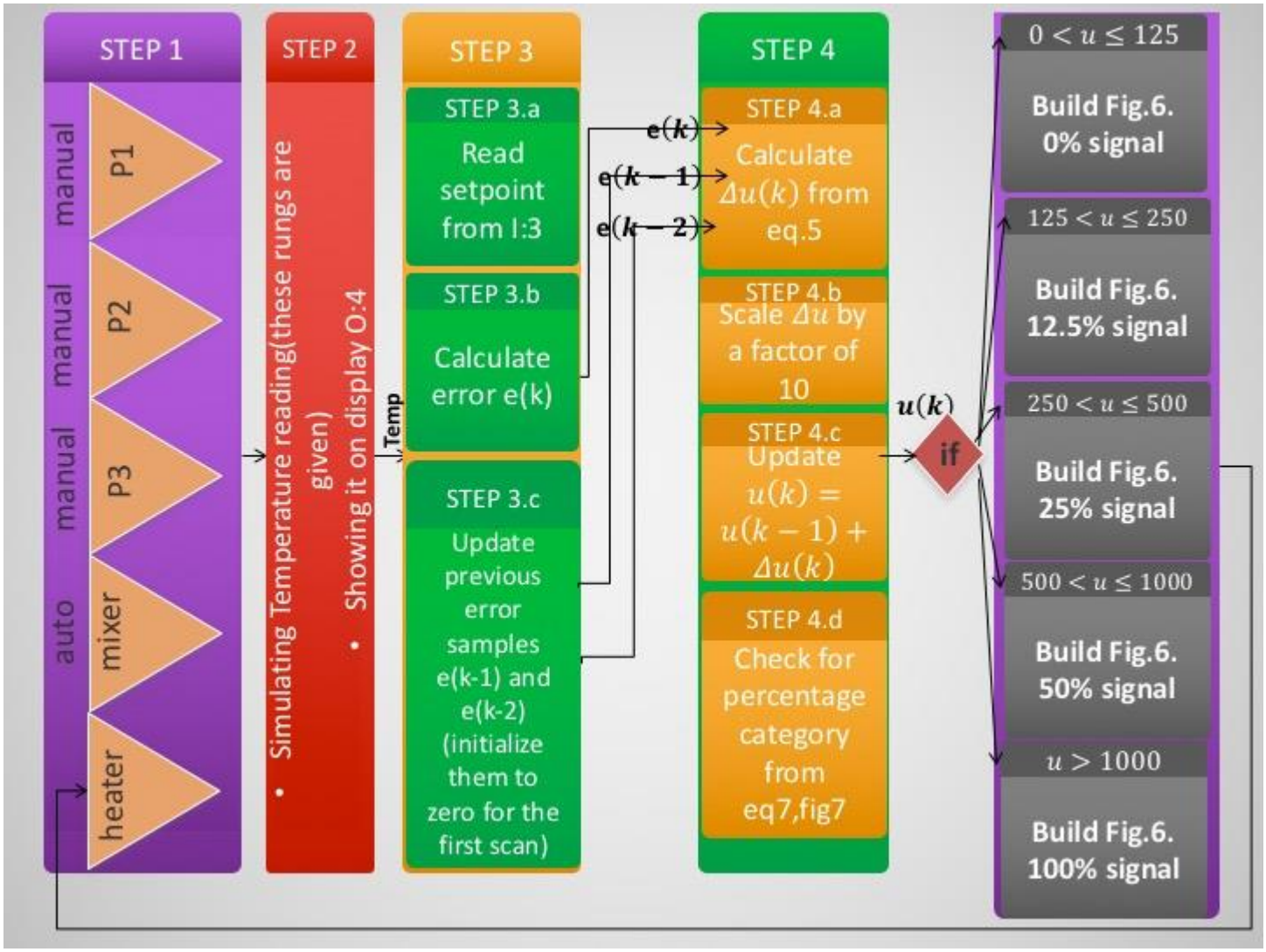

Figure 8: A signal flow graph describing how to organize steps of the process