BL40A2910 Automation **19.12.2017**

Jouni Kortelainen **0418271**
Janne Kauppi **0413506**
Lauri Seppänen **0438248**
Paavo Pirinen **0418365**

LTR700 Controller Report

1. Introduction

In this course project a controller was designed for LTR700 “hairdryer” fan-heater system. The aim was to design a controller to keep the temperature of the system at the set reference temperature by adjusting the rotation speed of the fan. The system was tested by externally modifying the heater level. In addition, a user interface was designed with InduSoft software.

The system consisted of the LTR700, Beckhoff PLC and ABB CI511 and CI512 communication interface modules using EtherCAT connection. The controller was implemented in TwinCAT 3 software. Most of the program was written in structured text, but continuous function chart was used for the logic handling the digital I/O for leds, buttons and switches.

2. Controller design

The designed controller was a cascade PI controller. The outer loop of the cascade controls the temperature and the faster inner loop controls the fan speed. The cascade controller allows fast response from the fan speed controller to disturbances as the controller does not need to wait for the temperature inside the system to change – instead, it reacts quickly to the minor changes in the outer temperature loop. The inputs of the controller were the LTR700 temperature and pressure (air flow) sensor readings and the outputs were current signals for the fan and the heater element.

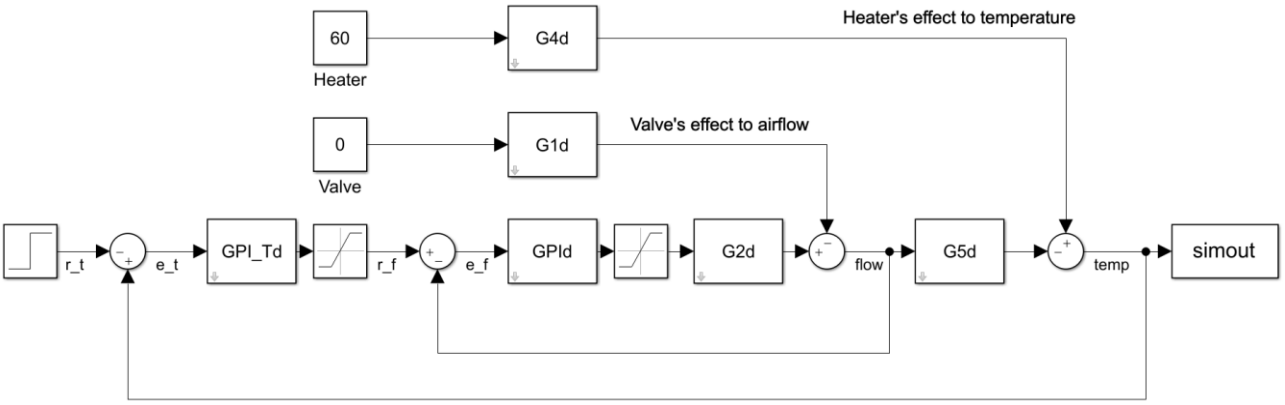

Figure 1. The cascade controller created in Simulink.

The transfer functions shown in the figure 1 are as follows:

$$G1 = e^{0.1}\frac{0.12}{(s+1)},$$

$$G2 = e^{0.1}\frac{0.9}{(1.3s+1)},$$

$$G4 = e^{3.6}\frac{1.05}{(57.5s+1)},$$

$$G5 = e^{5}\frac{0.75}{(50s+1)}.$$

G1 describes flip valve's impact to the air flow, G2 flow control signal's relation to the air flow and G4 is the heater's impact to the temperature. G5 is the main process and it describes the air flow's impact to the temperature.

In addition to the cascade controller, a fuzzy logic controller as shown in figure 2 and 3 was designed. The designed fuzzy controller uses only the temperature feedback.

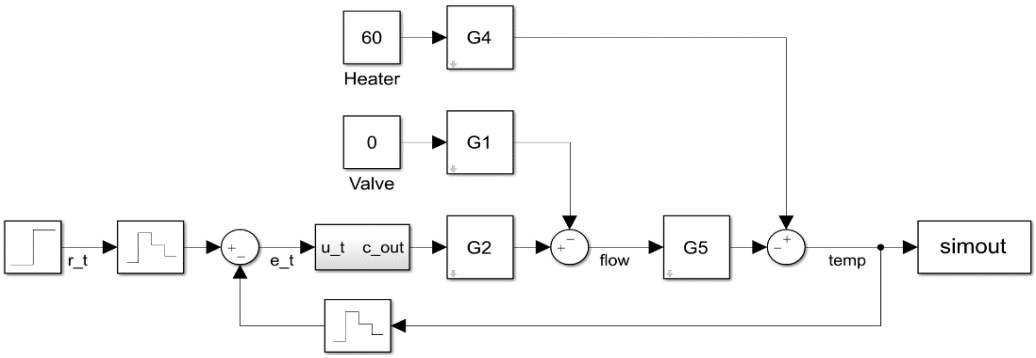

Figure 2. The system with fuzzy logic controller created in Simulink.

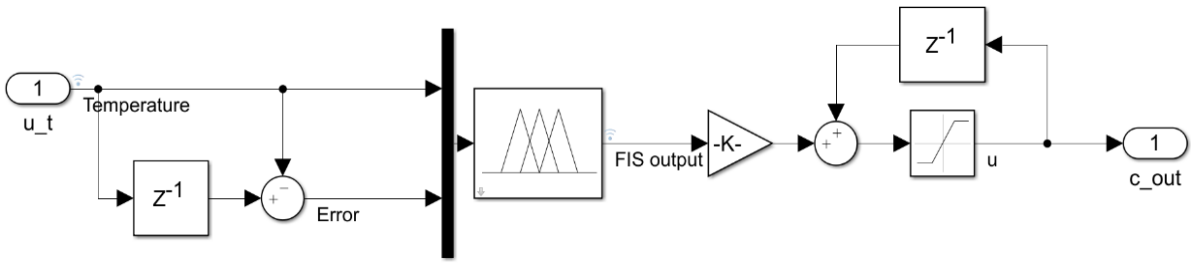

Figure 3. The fuzzy logic.

From the feedback temperature, the controller calculates the current temperature error and the derivative of the error. For making the rules more readable, the temperature error was named just temperature and the derivative of the error just error.

The temperature was considered to be hot, warm, cool or cold. Hot and cold were trapezoidal and warm and cool were triangular. The error could either be decreasing or increasing, both were trapezoidal. The output u is either slowdown or speedup, both are triangular.

The rules are as follows:

1. IF temperature is hot THEN u is speedup
2. IF temperature is warm AND error is increasing THEN u is speedup
3. IF temperature is warm AND error is decreasing THEN u is slowdown
4. IF temperature is cool AND error is increasing THEN u is speedup
5. IF temperature is cool AND error is decreasing THEN u is slowdown
6. IF temperature is cold THEN u is slowdown

Minimum was used as the and operator and the defuzzification was done using the mean of max method.

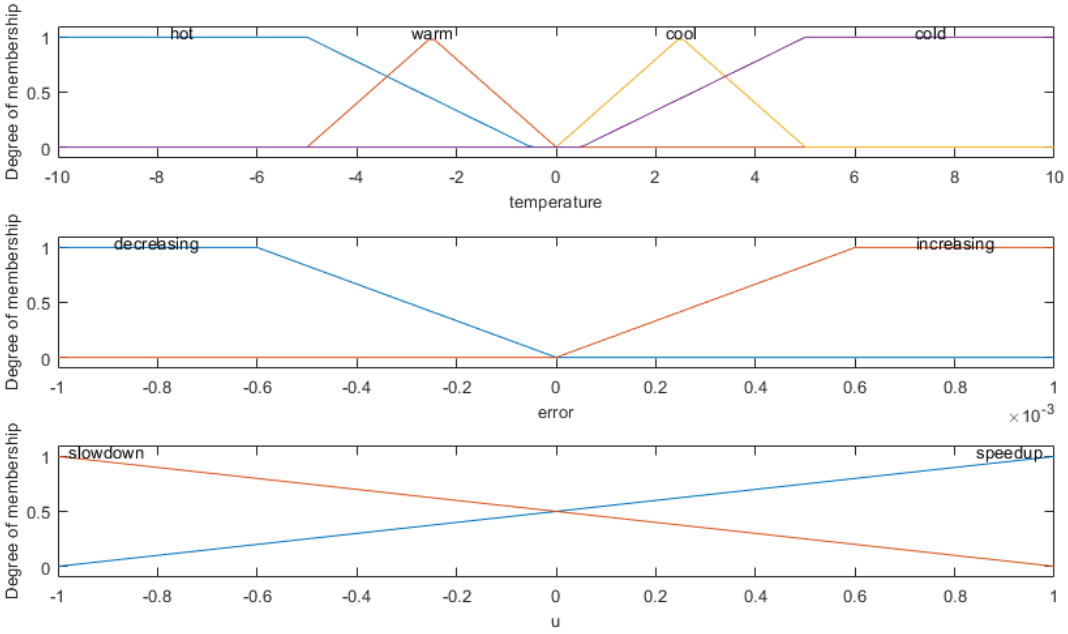

Figure 4. The fuzzy logic rules.

The sampling time of the system had to be changed from 10 ms to 100 ms because of the accuracy of the real-world system. The error between two samples was so small that the rules 2 to 5 were not working correctly. After the change, the rules were making some changes to the output.

3. User interface

User interface was created with InduSoft and is shown in figure 5.

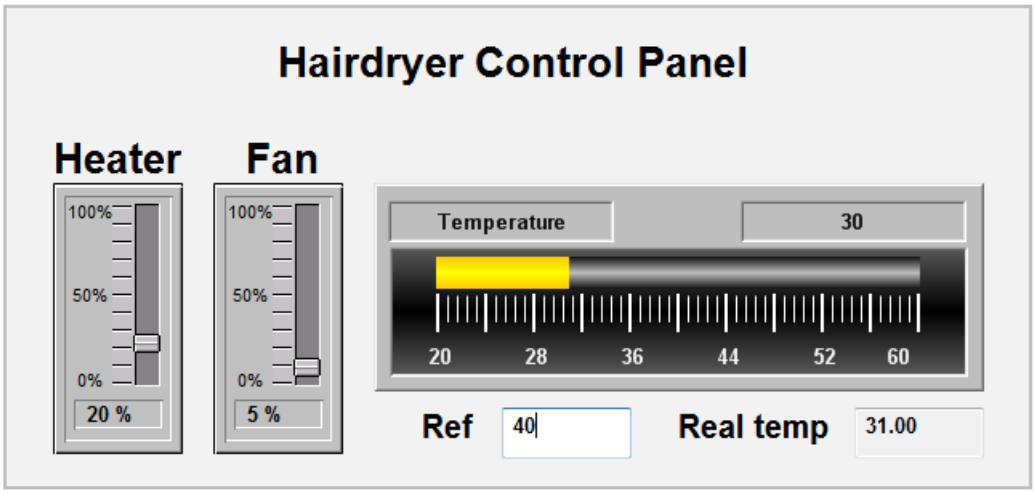

Figure 5. User interface created with InduSoft.

The user interface displays the temperature of the system on the bar graph as well as with a number "Real temp". It also shows the heater and fan operating levels in percentage. The interface allows modifying the heater level with the slider and changing the reference temperature "Ref".

Physical interface was also used, in form of leds, buttons and switches. The red led was used to indicate, that the controller was off, green that the fan was on and yellow that the system was in steady-state region. With a press of a buttons users can increment and decrement the reference value. Switches were used for turning the control on and off and switching between fuzzy and the PI controllers.

4. The program

The logic for digital I/O was done with continuous function chart and the functionality was described in section 3. User interface. The other functionality was written in structured text.

The main program calls the logic for digital I/O and desired controller. It also handles the GUI's bar graphs by sets the input value to the heater and calculating the fan usage ratio. As the last step, the main program gets the current temperature reading and stores it.

The PI controller was done using three function blocks. One of the blocks is used to call the other two, which are the actual controllers. The main block is used to set the fan input value and setting the fan input value to zero, if the control was turned off.

Fuzzy controller was done using a function block and two functions. Functions were used for fuzzyfication of tringle and trapezoidal membership functions. The function block contains membership functions, rules, defuzzification and the fan input value setting. To make handling membership functions easier, two structures, Trig and Trapez were made.

Utility function blocks were also used throughout the code. fbB2U for turning binary number into integer, fbU2B for turning integer into binary number and fbScale for turning analog input values

into real value and the other way around. FbSplit and fbIfElse were used in continuous function chart to split input into two outputs and for simple if-else-structure.

5. Results

The operation of the cascade PI-controller can be seen from the figure 6 and the fuzzy logic controller in the figure 7.

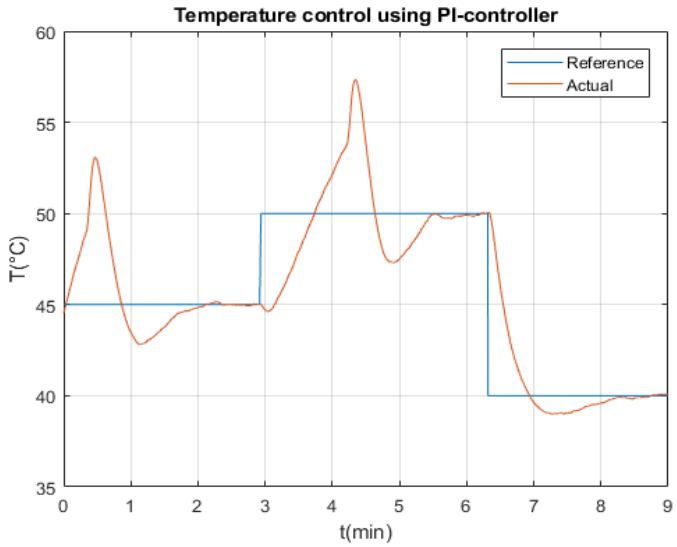

Figure 6. Temperature control using PI-controller.

From the figure 6 it can be seen that there is a 10 degree overshoot, followed by a 5 degree negative overshoot and an approximately 2 minute settling time. There are high spikes in the first overshoots at 0.5 minute and 4.5 minute marks that are caused by the fan turning on again and pushing the hot air to the temperature sensor after the reference value had been raised (and the fan had turned off to allow the temperature to rise).

Once settled, the controller follows the reference temperature accurately with approximately 0.05 degree error.

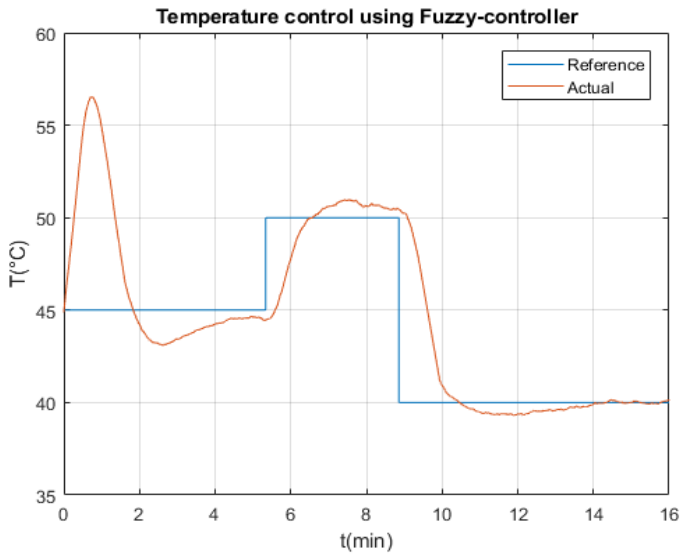

Figure 7. Temperature control using Fuzzy-controller.

From the figure 7 it can be seen, that the fuzzy controller is much slower than the PI controller and the initial temperature reaches 57 degrees. Because of the slowness, the fan doesn't completely stop when the reference is increased from 45 degrees to 50 and temperature doesn't overshoot as in figure 6.

Rules 2 to 5 have hard time correcting the steady-state error, as seen from the second step, 6 to 9 minutes in. Given enough time, the controller does correct the error, as seen at 14-minute mark.

6. Conclusion

The cascade PI and the fuzzy logic controllers were successfully designed and implemented. Both controllers were able to accurately follow the reference value, once settled. The system is very slow which can be seen in the results, and there is room for improvement regarding the overshooting and settling time.