PID CONTROL FOR HEATING OVEN BY MATLAB-EMBEDDED ARDUINO

Thi-Ai-Van Nguyen*, Minh-Truong Nguyen, Duy-Khanh Nguyen, Minh-Dung Mai, Van-Lam Thieu, Kim-Son Ngo, Van-Son Pham, Thanh-Son Pham, Minh-Huy Pham

Ho Chi Minh city University of Technology and Education (HCMUTE)
Vo Van Ngan street, 01, Ho Chi Minh city, Vietnam
* Corresponding author. E-mail: 20151108@student.hcmute.edu.vn

Abstract: The heating oven is a basic and easy-to-build model for exchanging and heating various types of engine equipment that require the use of thermal energy. In addition, this is also a model used in research in laboratories using basic algorithms such as PID. However, in the past, to control the heating oven, we needed to use expensive PLCs and microprocessors, and it was difficult to write programs. Recently, there are many studies that have succeeded in getting the program into the Arduino using C. Programming and this method requires the operator to have a lot of programming knowledge. To solve this problem, we have implemented and succeeded in using Matlab/Simulink with embedded Arduino and real model.
Keywords: Heating oven, PID control, Matlab/Simulink, Arduino.

1. Introduction

Temperature is the major control parameter in the procedure of industrial manufacture all over the world [1]. Specifically, heating oven are now commonly used in industries such as food and agriculture [2]. Heating oven are designed to perform the function of keeping an object or process at a desired temperature. Proper heating during product manufacturing to ensure quality and prolong service life. This can simplify human work by implementing a control system. PID control is known as a simple and powerful control applied to temperature control applications. According to statistics, more than 97% industrial controller is the same type as PID controller [3]. Instead of executing the instructions to program the microprocessor to control the appropriate temperature [4], in this paper, we will refer to the method of designing a PID controller by embbedding Matlab. This shortens the designing time of the PID controller. By trying many times with different PID parameters, we find the right PID parameter to meet the desired output temperature.

2. Algorithms

2.1. Overview Diagram

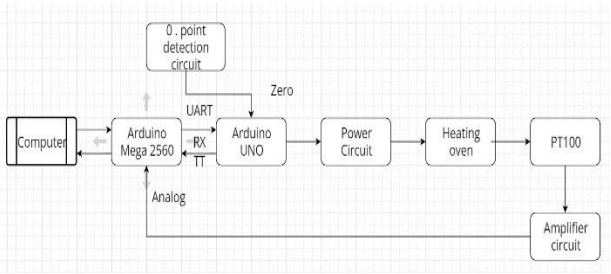

Fig. 1. Overview diagram

The overview diagram is described as follows:

- Heat oven use 230 VAC filament bulbs to generate heat.
- PT100 sensor through the amplifier circuit can adjust the temperature measurement range of the furnace, we replace it with LM35.
- Zero point detection circuit: detect zero point of AC voltage from there and send signals to the microcontroller, control the power circuit to switch the light bulb.
- Use 2 microcontrollers: 1 microcontroller (UNO) to control the power circuit and zero detection circuit, a microcontroller (MEGA) communicates with the sensor
- PT100 and dump controllers like PID, Fuzzy.
- Two microcontrollers will communicate with each other through the UART communication standard.

2.2. Connection Diagrams

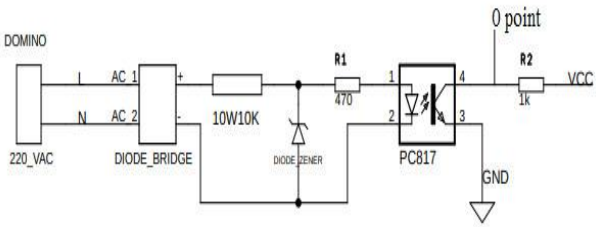

Fig. 2. Point 0 detection block voltage 220 VAC

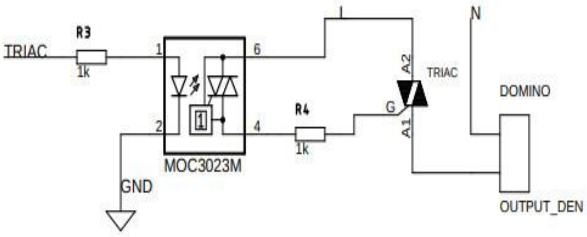

Fig. 3. Triac exciter block

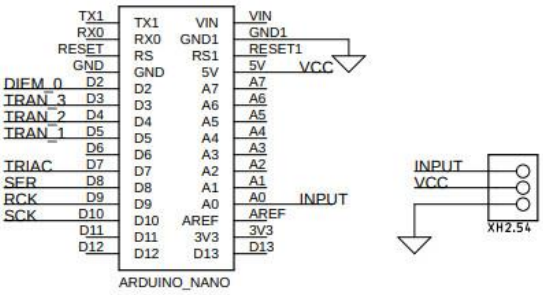

Fig. 4. Arduino and connector

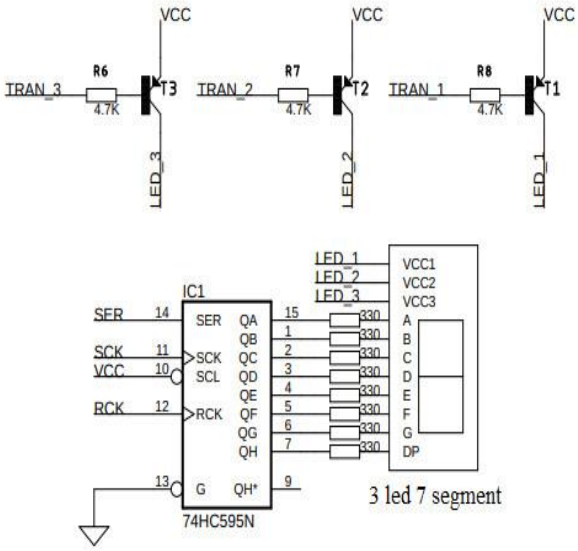

Fig. 5. Scan block 3 LED 7 segment display light time

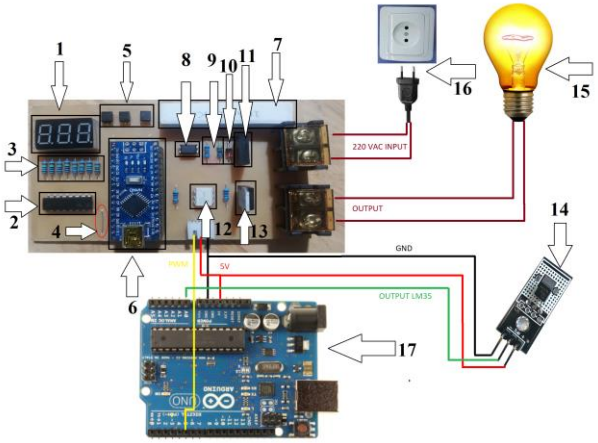

Fig. 6. Connection diagram

Note:

1. led temperature display
2. 74HC595N
3. 330 ohm resistor
4. Wire
5. A1015
6. Arduino Nano
7. Cement resistor 10W10KJ
8. PC817
9. 470 ohm resistor
10. Diode zener 5V
11. Diode bridge 3A
12. Mooc3023
13. BTA16
14. LM35
15. Incandescent bulbs
16. 220V AC power supply
17. Arduino Uno R3

Amplifier circuit for signal processing to increase voltage resolution is shown in

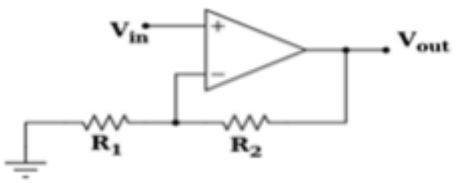

Fig. 7. Amplifier circuit

Then, the microcontroller will read the signal from pin V_{out} of the amplifier and signal processing circuit suitable for triggering the G pin of the triac to make the lamp shining. The change of the jack angle α determines the open power of the triac. When the angle α changes, the output voltage of the triac will also change. Corner size α=0 capacity reaches 100% triac fully open, α at power reaches 50% triac half open, α at power reaches 0% triac stop conduction. The author can control the excitation α when finding the zero point, that is, the intersection point between negative and positive periods of AC voltage

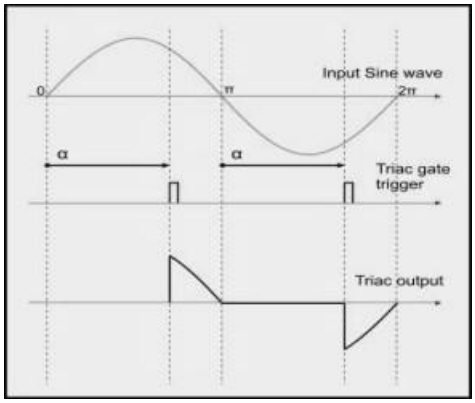

Fig. 8. AC voltage operating cycle

When applying 230V AC voltage through the diode bridge, we get a voltage with positive half-life oscillation with frequency f = 50 Hz, ie oscillation period T = 0.02 s = 20 ms. So the half-period of oscillation is T_{haft}=T/2= 10 ms. The current through the diode bridge to the zener diode has V_Z=4.7V. When V_{bridge}<V_Z then the voltage flows through the resistor 470 Ω.

At this point, the zener diode conducts forward and is seen like a normal semiconductor diode. When the zenner diode conducts inversely and is seen as a 4.7 V power supply voltage for led in opto PC817 glows lead pin 3 and 4 with together. Then the signal pin connected to the microcontroller control is connected to ground. Based on this point, we use the microcontroller's external interrupt to receive know the synchronous signal of AC voltage, calculate the opening angle of the triac. I only control time time period from 1 ms to 9 ms. Because voltage takes time delay to rise 5 V and drops to 5 V.

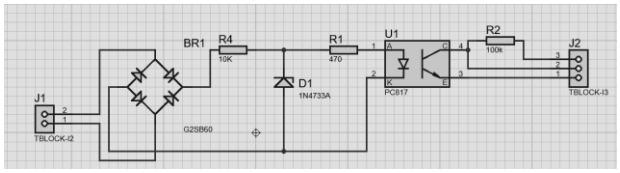

Fig. 9. Zero point detection circuit

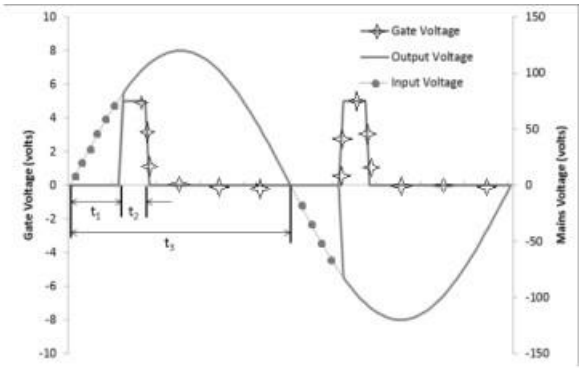

Fig. 10. Relation of triac opening angle and time

Inside:

- t1 corresponds to the opening angle of the triac;
- t2 impulse time of pin G to trigger;
- t3 the time during which the current is switched on and off in half cycle.

When there is a voltage signal since the microcontroller detects 0 point, led of the MOC3023 will light up and lead the pin 4 and pin 6 so there will be a trigger voltage on the G pin of the triac let the light on.

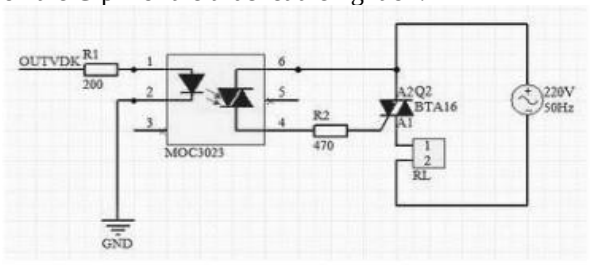

Fig. 11. Triac open angle driver circuit

3. Simulation

3.1. PID algorithm

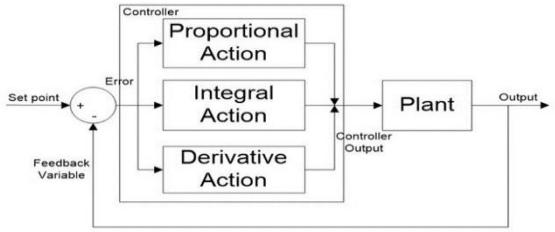

Fig. 12. PID controller

PID is a closed-loop feedback controller, which is a combination of three proportional, integral, and differential controllers. The error between temperature of system and the expected temperature is input of PID controller. And, this controller, through its algorithms, minimizes this error. PID controller has ability to suppress setting error, increase response speed, and reduce overshoot if the controller's parameters are selected appropriately.

The influence of PID parameters is shown in Tab. 1 below.

Tab. 1. Influence of PID parameters to response of system

Input response	Rise time	Overshoot	Transient time	Setting error
K_P	Decrease	Increase	Small change	Decrease
K_D	Decrease	Increase	Increase	Eliminate
K_I	Small change	Decrease	Decrease	Small change

3.2. Simulation Diagram

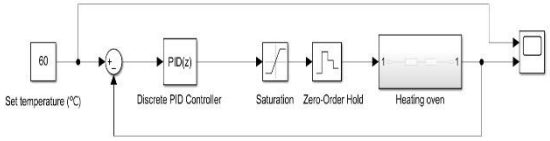

Fig. 13. Schematic of the simulation of heating oven using a PID controller

Fig. 14. Heating oven block

3.3. Simulation Diagram

3.3.1. Standard PID

Select a set of PID parameters to control the system stably:

With a set temperature of 60 degrees C

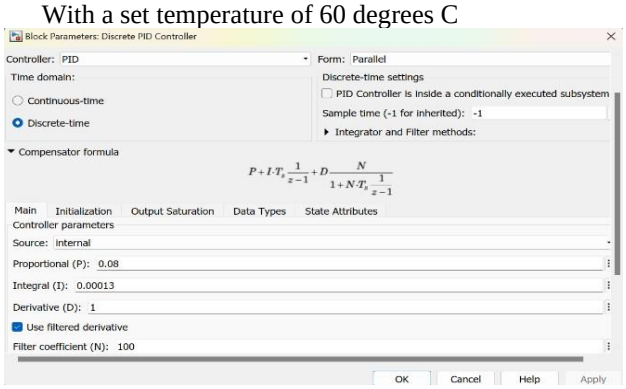

Fig. 15. Stable PID value

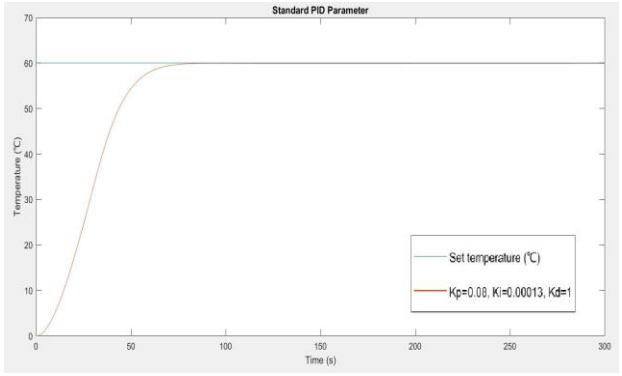

Fig. 16. Result of standard PID value

Comment: We see that the output response has no overshoot, the time for the system to reach the set temperature of 60 degrees Celsius is about 90s and the setting error is zero.

3.3.2. Change K_P value

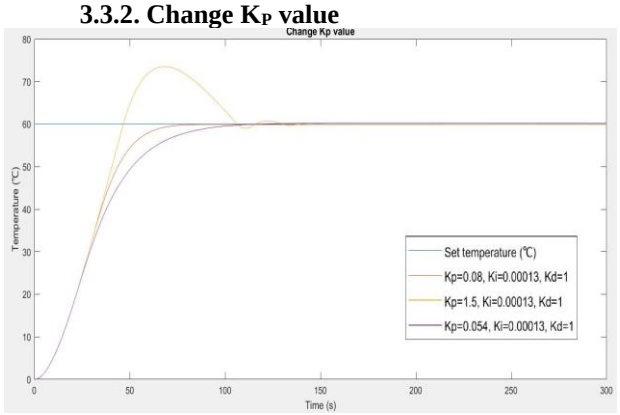

Fig. 17. Result of change in KP value

- *Increase in K_P value*

$K_P = 1.5$

Comment: When increasing K_P, we see that the output response has an overshoot of POT=23%, the time for the system to reach the set temperature is about 150s slower than the standard PID parameter, and the setting error is zero.

- *Decrease in K_P value*

$K_P = 0.054$

Comment: : When reducing K_P, we see that the output response has no overshoot, but the time for the system to reach 60 degrees Celsius is very slow, about 110s, the error is zero.

3.3.3. Change K_I value

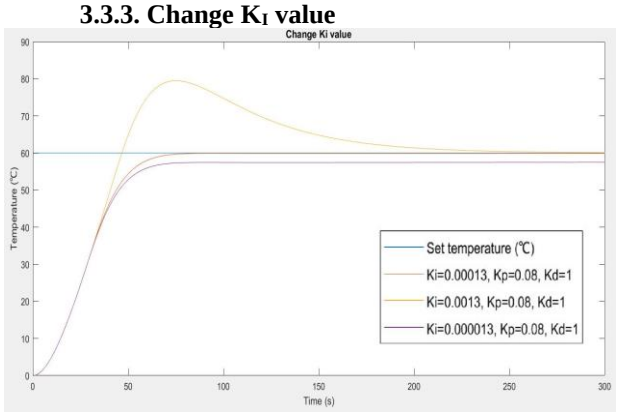

Fig. 18. Result of change in K_I value

- *Increase in K_I value*

$K_I = 0.0013$

Comment: When increasing K_I, the system output has a high overshoot POT=33% and the time for the system to stabilize at 60 degrees Celsius is about 260s, the setting error is very small, close to zero.

- *Decrease in K_I value*

$K_I = 0.000013$

Comment: : Reducing K_I we see that the system has no overshoot, reaching a temperature of 60 degrees Celsius very slowly.

3.3.4. Change K_D value

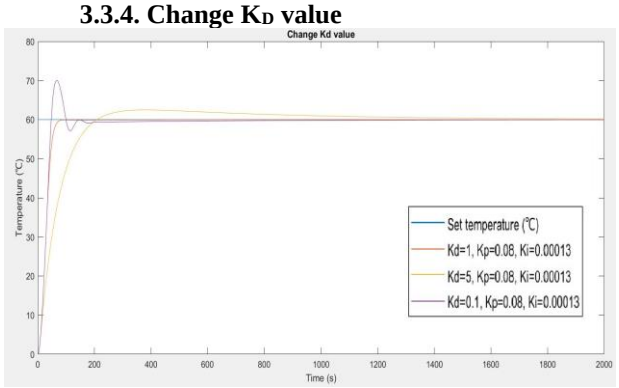

Fig. 19. Result of change K_D value

- *Increase in K_D value*

$K_D = 5$

Comment: When increasing Kd, the system starts to lose stability, the system output has a overshoot POT=7%, the time for the system to stabilize at 60 degrees Celsius is about 1600s

- *Decrease in K_D value*

$K_D = 0.1$

Comment: When reducing Kd, the system gradually overshoots POT=17%, the time for the system to stabilize is slower than the standard PID parameter about 600s, the setting error is almost zero.

4. Experimental Model

The model of the furnace in the article includes 1 box mica and 3D printed joints for assembly. Beside In the oven, a layer of insulation is placed to keep the heat in and temperature sensor LM35 placed inside to measure furnace temperature. Realistic oven model as Figure 20 and Figure 21.

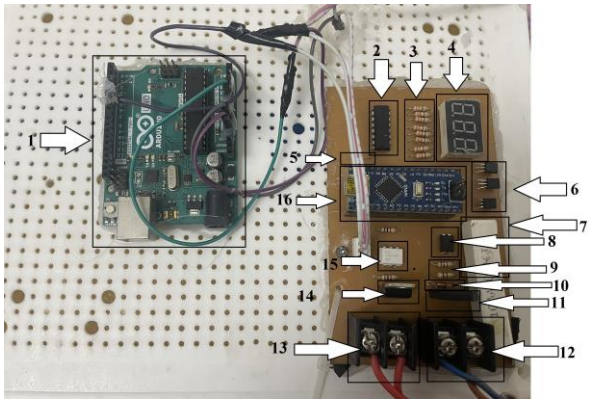

Fig. 20. Outside the heating oven

Note:

1. Arduino Uno R3
2. 74HC595N
3. 330 ohm resistor
4. Led temperature display
5. Wire
6. A1015
7. Cement resistor 10W10KJ
8. PC817
9. 470 ohm resistor
10. Diode zener 5V
11. Diode bright 3A
12. 220V AC power supply
13. Incandescent bulls
14. BTA 16
15. Mooc 3023
16. Arduino Nano

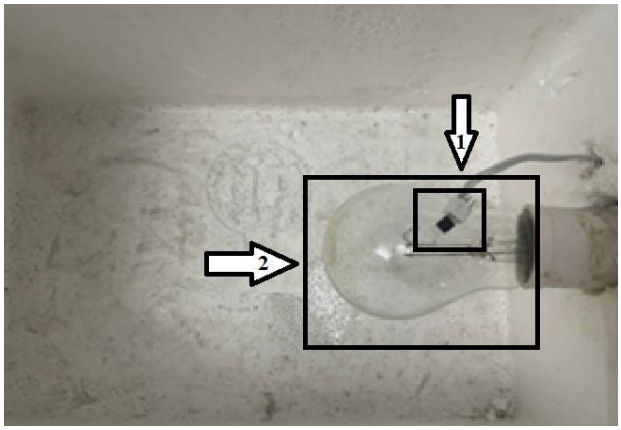

Fig. 21. Inside the heating oven

Note:

1. Temperature sensor (LM35)
2. Incandescent bulbs

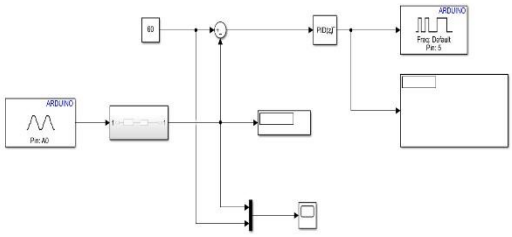

Fig. 22. PID Matlab-embbed program

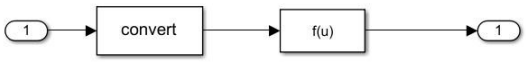

Fig. 23. Subsystem blocks

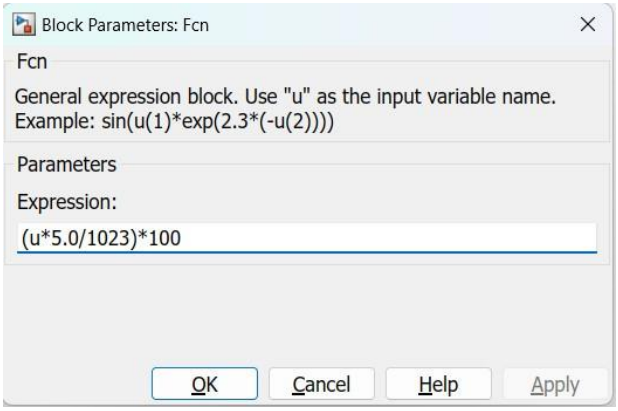

Fig. 24. Expression for FCN

4.1. Standard PID

Select a set of PID parameters to control the system stably:

With a set temperature of 60 degrees C:

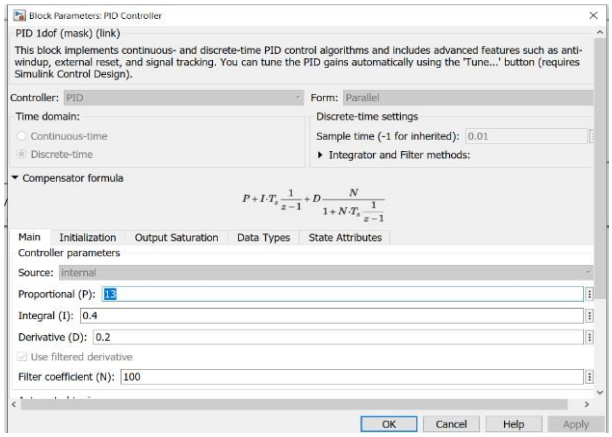

Fig. 25. Stable PID values

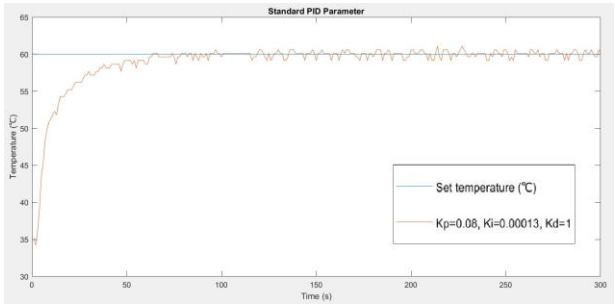

Fig. 26. Result of standard PID value

Comment: We see that the output response has no overshoot, the time for the system to reach the set temperature of 60 degrees Celsius is about 100s and the setting error is zero.

4.2. Change K_P value

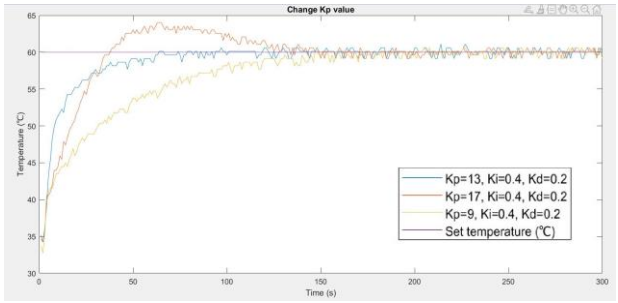

Fig. 27. Result of change in K_P value

- *Increase in K_P value*

$K_P = 17$

Comment: When increasing K_P, we see that the output response has an overshoot of POT= 6.7%, the time for the system to reach the set temperature is about 175s slower than the standard PID parameter, and the setting error is zero.

- *Decrease in K_P value*

$K_P = 9$

Comment: When reducing K_P, we see that the output response has no overshoot, but the time for the system to reach 60 degrees Celsius is very slow, about 200s, the error is zero.

4.3. Change K_I value

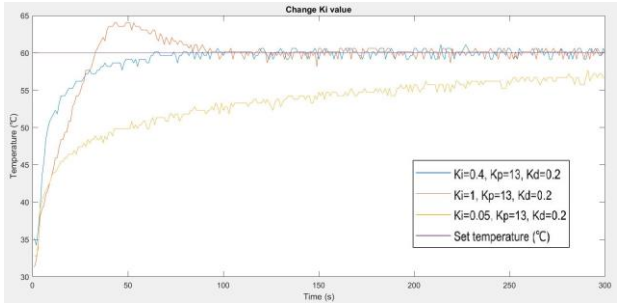

Fig. 28. Result of change in K_I value

- *Increase in K_I value*

$K_I = 1$

Comment: When increasing K_I, the system output has a high overshoot POT=7.5% and the time for the system to stabilize at 60 degrees Celsius is about 180s, the setting error is very small, close to zero.

- *Decrease in K_I value*

$K_I = 0.05$

Comment: Reducing K_I we see that the system has no overshoot, reaching a temperature of 60 degrees Celsius very slowly, after 300s, the system begins to stabilize.

4.4. Change K_D value

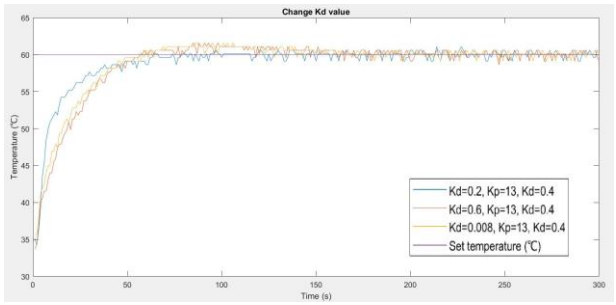

Fig. 29. Result of change in K_D value

- *Increase in K_D value*

$K_D = 0.6$

Comment: When increasing K_D, we see that the time for the system to reach the set temperature of 60 degrees Celsius is improved by about 75s, then the system starts to stabilize, the overshoot and the setting error are almost zero.

- *Decrease in K_D value*

$K_D = 0.008$

Comment: When reducing K_D, the system gradually overshoots POT=3.3%, the time for the system to stabilize is slower than the standard PID parameter about 200s, the setting error is almost zero.

5. Conclusions

About the PID calibration, we obtain this information:

- If K_P is larger, we obtain the faster the response speed of output of system.
- If K_P is larger, we obtain smaller steady-state error (but with only K_P calibration, this error cannot be eliminated). In this case, poles of the system tend to move away from the real axis. Thence, output fluctuates more, overshoot is higher and system can be uncontrollable if K_P is too large.
- Transient response is slower and steady state error is larger if we decrease K_I.
- We can eliminate the settling error by an increasing of K_I. However, too big K_I can cause higher overshoot and the instability of system.
- If we increase K_D, transient response of system become more faster and we also obtain less overshoot. But, too big value of K_D makes system more sensitive to noise which often has high frequency.
- K_P is most important. K_I can be used to combine with K_P. And, K_D can be considered less important in these three components.

6. Acknowledgement

This work belongs to the project SV2023-58. This project is funded by Ho Chi Minh city University of Technology and Education (HCMUTE), Vietnam. We also want to give thanks to Ms. Eng. Thi-Hong-Lam Le and PhD. Van-Dong-Hai Nguyen due to their sincere help in operating intruments in lab for the experiments.

7. References

[1] Qin G., Ma Y., Zhang X., Zhang M.: “Design of Fuzzy Adaptive PID Temperature Controller Based on FPGA”, TELKOMNIKA, Vol. 11, No. 10, pp. 6008 ~ 6016 ISSN: 2302-4046, 2013.

[2] Palaniyappan T.K., Yadav V., Ruchira, Tayal V.K., Choudekar P.: "PID Control Design for a Temperature Control System," 2018 International Conference on Power Energy, Environment and Intelligent Control (PEEIC), Greater Noida, India, pp. 632-637, 2018.

[3] Astrom K., Hagglund T.: “Revisiting the Ziegler-Nichols step response method for PID control”, Journal of Process Control, pp. 635-650, 2004.

[4] Nakamori S.: „Arduino-based PID Control of Temperature in Closed Space by Pulse Width Modulation of AC Voltage2, International Journal of Computers and Systems Engineering, Vol. 2, No. 1, 2021.