Journal of Information Hiding and Multimedia Signal Processing ©2019 ISSN 2073-4212
Ubiquitous International Volume **10**, Number **3**, September **2019**

An Internet of Things Thermostat Sensor Developed with an Arduino Device Using a Recursively Digital Optimization Algorithm

Yeong-Chin Chen*, Mariana Syamsudin and Weide Xu

Department of Computer Science and Information Engineering
Asia University
500 Liufeng Rd., Wufeng, Taichung 41354, Taiwan
ycchenster@gmail.com, ycchenster@asia.edu.tw

Abstract. *Internet of Things (IoT) applications have become increasingly popular along with the technology development of various sensing devices. In this study, a recursively digital Proportional Integral Derivative (PID) algorithm has been applied to develop an Arduino-based IOT thermostat sensor system. By deploying such a low-power device, environmental temperatures are acquired and sent into an Arduino controller to make comparisons with the target temperature variation. By using the Pulse Width Modulation (PWM) control principle, the recursive digital PID calculation is converted to drive the PWM duty-cycle output current to achieve heating and cooling for the thermostat condition. Noting that by using low-cost electronic components with the development of the Arduino controller, PID operation and PWM technology, temperature tracking has been demonstrated by controlling and adjusting a lamp bulb and an electric fan to achieve a complete smart thermostat control. Additionally, the temperature data gathered in the Arduino controller can be uploaded to a cloud database. By using an Android mobile device, which is used as a display and control interface, temperature information can be conveniently accessed, such as temperature curve exporting, real-time monitoring of past temperature changes, and even further large-data analysis results in the cloud database.*

Keywords: IoT thermostat sensor, Arduino, recursive PID, PWM

1. **Introduction.**

A thermostat monitoring [1] system has a wide range of applications throughout the military, industrial and medical fields, household electrical instruments, automotive coolant control, and non-destructive testing and development. In addition to preventing catastrophic failure from extraordinary temperature variation, the thermostat instrument could significantly improve operational efficiency and extend a machines life cycle. The gathered and stored temperature data by thermostat monitoring can further provide environmental temperature control and improvement.

In recent years, the functionality of electronic equipment has been diversified and power-consumption has increased. Moreover, the greater the power consumed, the faster the temperature increases. If the operating instrument environment cannot control the temperature situation, especially to begin immediate cooling, it may cause instrument malfunctions, and these disorders are likely to cause damage or even accidents.

During the normal operation of the power system, the temperature at which the instrument operates represents a normal and efficient operation of the instrument. The stability of the temperature is important for the reliability, safety and economy of the instrument index. Therefore, maintaining the thermostat condition can reduce the risk of loss of instrument operation. This aim become a trigger of a new standard of thermostat which is shifting toward intelligent control [2].

In this work, we have developed an IoT [3] temperature detection and control system, which is based on low-cost components of the Arduino controller [4], temperature sensor, and WiFi transmission module combined with a cloud server. A temperature sensor [5] is connected to the analog input port of the Arduino controller, and the digitized temperature values are transmitted to the cloud server for tracking the gathered temperature, exporting historical data for troubleshooting, as well as further large-data analysis.

2. System Architecture.

The thermostat system architecture and detailed design flow of the sensing device using the Arduino microcontroller are shown in Sect. 2.1. The cooling hardware components accompanied with the Arduino controller are demonstrated in Sect. 2.2. The heating device environment is depicted in Sect. 2.3.

2.1. Thermostat Sensing Device Design.

As shown in Fig. 1 the Arduino controller acts as a major temperature regulator to determine whether to cool down or heat up on the basis of the thermostat setting from the WiFi module and the environment temperature sensed. If heating is needed, after PID computing, it will control the PWM duty cycle to adjust the lamp brightness. If cooling is needed, it will similarly control the PWM duty cycle to spin the fan in order to achieve the desired thermostat condition.

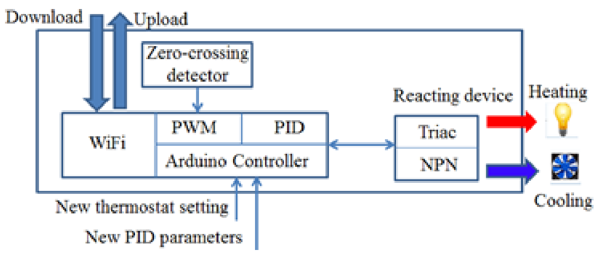


FIGURE 1. System design diagram for thermostat sensing device

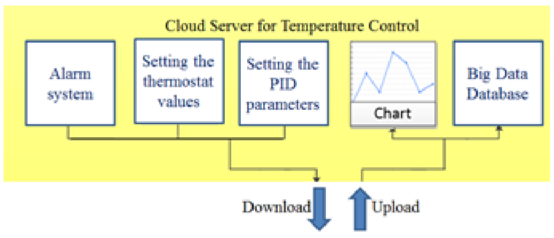

Cloud Server for Temperature Control
Alarm system
Setting the thermostat values
Setting the PID parameters
Chart
Big Data Database
Download
Upload

FIGURE 2. Cloud server for temperature control

PHP [6] was used for the web software design. The cloud system in Fig. 2 is divided into five functions to allow users to set the desired thermostat value in accordance with their needs. The system sends the temperature settings directly to the Arduino controller through the Internet. Users can also choose to modify their own PID parameters. After sending these parameters, the Arduino controller will receive new PID parameters, which are set into the original operation to accomplish the setting. The Arduino controller can also upload the measured temperature of the thermostat sensing devices to the cloud server, where the system will import data to be stored in a large-data database, and create a dynamic graph display in the user interface so that users can monitor the actual temperature which is controlled by the thermostat directly in real time.

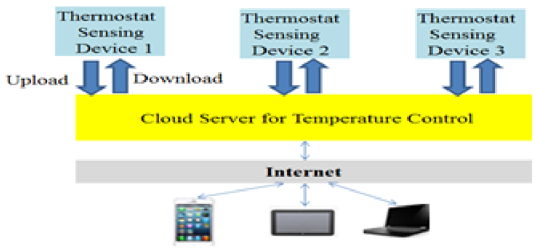

Thermostat Sensing Device 1
Thermostat Sensing Device 2
Thermostat Sensing Device 3
Upload
Download
Cloud Server for Temperature Control
Internet

Figure 3. Simultaneous monitoring for multiple thermostat sensing devices

To achieve a more comprehensive monitoring system, this system supports monitoring for multiple devices at the same time. As shown in Fig. 3, through the cloud server, the user can select one of the thermostat sensing devices to monitor and provide different settings for each individual device.

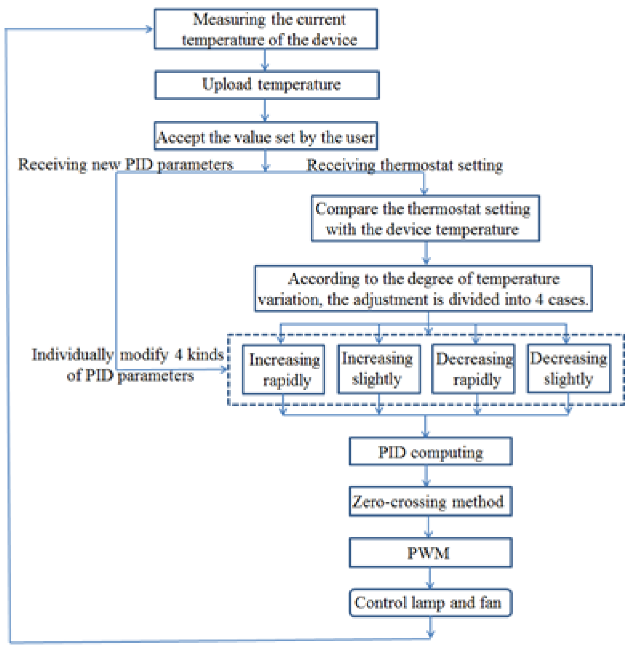

Measuring the current temperature of the device
Upload temperature
Accept the value set by the user
Receiving new PID parameters
Receiving thermostat setting
Compare the thermostat setting with the device temperature
According to the degree of temperature variation, the adjustment is divided into 4 cases.
Individually modify 4 kinds of PID parameters
Increasing rapidly
Increasing slightly
Decreasing rapidly
Decreasing slightly
PID computing
Zero-crossing method
PWM
Control lamp and fan

Figure 4. Design flow chart for the Arduino software program

Figure 4 shows a complete design flowchart for the Arduino program development. After the device temperature was detected, it was uploaded to the cloud server. Additionally, upon receiving the user temperature setting, the computing process is divided into four cases on the basis of the user setting and environmental temperature. More appropriate PID parameters are used for different situations in order to accelerate temperature tracking and regulation. If a new parameter setting is received, the PID parameters will be changed in accordance with the user setting. Next, according to the temperature difference, the PID is calculated by a Zero-crossing method [5] and PWM control to drive the lamp and fan to track the temperature variation.

2.2. **Electric fan hardware circuit.** Since the Arduino maximum output voltage is 5V, our thermostat system uses a 12V DC fan and a 110V AC lamp to control the environmental temperature. Therefore, it requires an external power supply combined with the required transistors, an optical-coupled device, and a bridge rectifier.

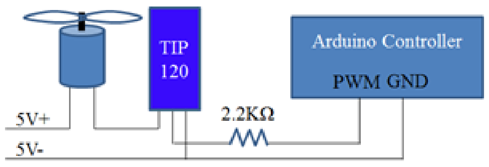


Figure 5. Electric fan circuit diagram

The external AC 110V is directly stepped down to 12V. The transistors like TIP210 are used to amplify the Arduino output signal range from (0 to 5V) to (0 to 12V) for the power supply of the electric fan. The electronic fan used in this work only needs basic voltage to start; the output voltage range of the Arduino is changed from (0 to 5V) to (2 to 5V) to complete the current amplification circuit. Finally, The PWM signal is applied on the base of an amplifier (TIP210) to regulate the driving power and control the fan speed as shown in Fig. 5.

2.3. **Lamp hardware circuit.** The lamp is powered by the AC 110V, which has a complete AC waveform for the positive half-wave and the negative half-wave. The 0V intersection points of the waveform are called the zero-crossing points [7]. Regardless of the positive or negative half waves, AC output voltage will be controlled on the basis of the zero-crossing points for reference. As shown in Fig. 6, we designed a dimmer system

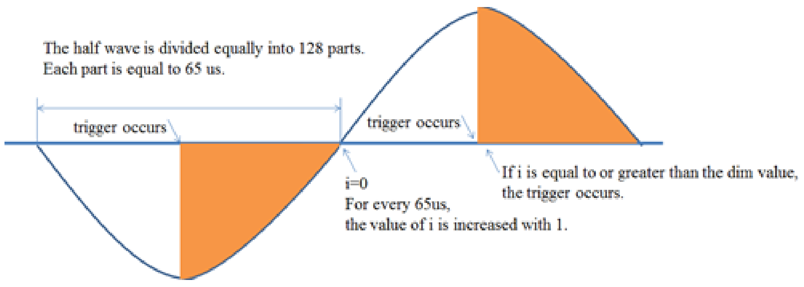

The half wave is divided equally into 128 parts.
Each part is equal to 65 us.
trigger occurs
trigger occurs
i=0
For every 65us,
the value of i is increased with 1.
If i is equal to or greater than the dim value,
the trigger occurs.

Figure 6. Zero-crossing waveform diagram

to divide the half wave into 128 equal parts, each with 65 s. After that, PWM is used to adjust the conduction interval and 50% of the conduction interval is assumed. Dimmers from the 1st to the 64th will not be turned on (delayed trigger), the 65th starts triggering to the next zero-crossing close cycle. Dimmers are adjusted to increase or decrease the period of the delay time which will be subtracted or added by 65s.

The three-pole AC switch (TRIAC) model is BTA12-600B, which was applied for power control applications. When the power control circuit in each half cycle is in the conduction

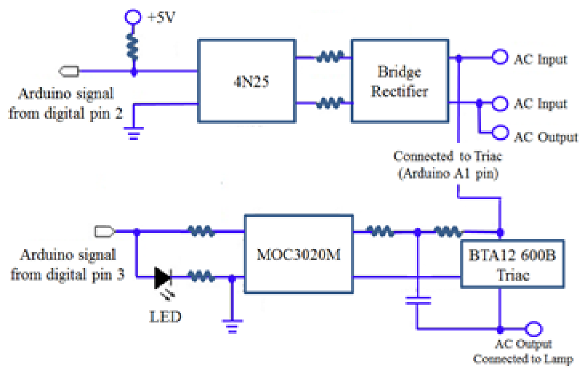

+5V
4N25
Bridge Rectifier
AC Input
AC Input
AC Output
Arduino signal from digital pin 2
Connected to Triac (Arduino A1 pin)
MOC3020M
BTA12 600B Triac
Arduino signal from digital pin 3
LED
AC Output Connected to Lamp

Figure 7. Dimmer circuit diagram

state, the power supplied to the load decreases. A longer on-time conduction state can provide a larger power consumption and a brighter bulb lamp.

Figure 7 is an AC dimmer circuit. The upper half part is a zero-crossing point detection circuit, using a 4N25 element to detect the zero-crossing point. Under normal conditions, the 4N25 usually remains in the low level voltage. Each time the zero-crossing point is detected, the 4N25 will generate a high level output. The lower half is the TRIAC control circuit, where the high-level voltage can trigger TRIAC, but the low-level potential inputs make it under a closed condition. MOC3020M is responsible for the DC and AC isolation and the trigger gate controller of the TRIAC element.

3. **System Design Consideration.**

The main idea is to use the Arduino controller and digital PID algorithm [8] to adjust the pulse duty cycle of the supply power. The Arduino is a low-cost microcontroller based on open-source development, and has a development environment similar to Java and C language. The Arduino hardware core is implemented by the ATMEGA2560 microcontroller family to provide general I/O control functions. The Arduino components used in this paper include a temperature sensor and a WiFi module

We first set the predetermined temperature value in the program. By using the Arduino assembly board to monitor the device temperature, here we use the digital PID control principle [13] to monitor the temperature increase or decrease by heating or cooling the target environment. The PID algorithm is also used to calculate the duty cycle rate of heating or cooling through PWM [14]. In Fig. 8, when the actual temperature $r(t)$ is greater than the setting temperature $c(t)$ for δ or more, the electric fan controlled by PWM will be started for cooling: in contrast, when the actual temperature r(t) is less than the setting temperature c(t) for or more, the lamp bulb will be started for heating, and the brightness of the lamp bulb is controlled by the PWM [15].

3.1. **Pulse Width Modulation (PWM).**

PWM is one of the modulation technologies used to convert an analog signal into a pulse signal. In general, the converted pulse wave period is fixed, but the duty cycle of the pulse wave will depend on the size of the analog signal.

The duty cycle is the ratio of the ON signal output duration of the period signal to the entire cycle of the signal, i.e., duty cycle = (duration of signals not equal to zero) / (duration of signals equal to zero and duration of signals not equal to 0) * 100%.

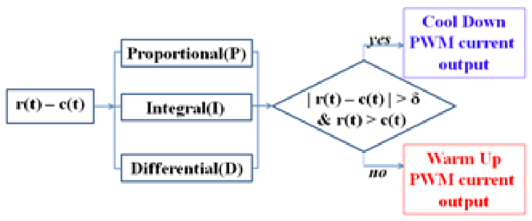

r(t) – c(t)
Proportional(P)
Integral(I)
Differential(D)
| r(t) – c(t) | > δ & r(t) > c(t)
yes
no
Cool Down PWM current output
Warm Up PWM current output

FIGURE 8. Thermostat adjustment for temperature tracking principle

There are five different PWM signals : 0%, 25%, 50%, 75% and 100% duty cycle outputs are shown in Fig. 9. These five PWM signals can be separately coded with five different analog signal values, and these analog signals values were 0%, 25%, 50%, 75% and 100% of the maximum signal values, respectively. For example, when the voltage supply is 9V and the output duty cycle is 25% of the PWM signal, it represents a 2.25V analog signal value.

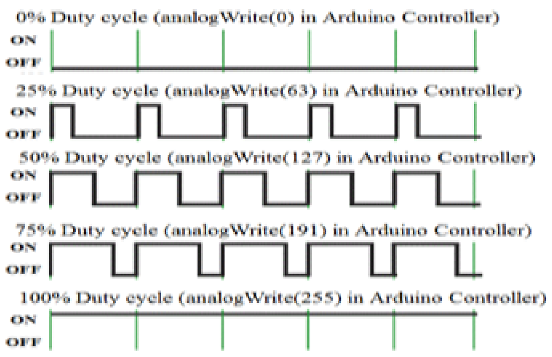

0% Duty cycle (analogWrite(0) in Arduino Controller)
25% Duty cycle (analogWrite(63) in Arduino Controller)
50% Duty cycle (analogWrite(127) in Arduino Controller)
75% Duty cycle (analogWrite(191) in Arduino Controller)
100% Duty cycle (analogWrite(255) in Arduino Controller)
ON
OFF

FIGURE 9. PWM signals with different duty cycles

In Fig. 10, when the temperature increases, the heating rate calculated on the basis of the PID control principle has a high value and therefore the PWM duty cycle becomes wide. On the other hand, when the temperature approaches the predetermined temperature, the PWM duty cycle becomes narrow and the heat amount of electric-heated rods will decrease to make the temperature more stably close to the predetermined temperature.

3.2. **PID Controls for proportional integral and proportional derivative.** The PID control system in Fig. 11 is a linear controller which is based on the closed-loop control system for the given input value r(t) and the actual output value $c(t)$

$$e(t) = r(t) - c(t). \tag{1}$$

The PID controller process is shown in the following eq. 2 The proportional, integral, and differential parts of eq. (2) are constituted by a linear combination of the control amount used to track the target control system.

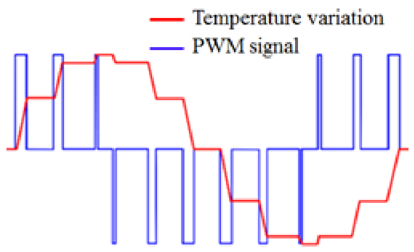


FIGURE 10. Heating and cooling rates using PWM signal

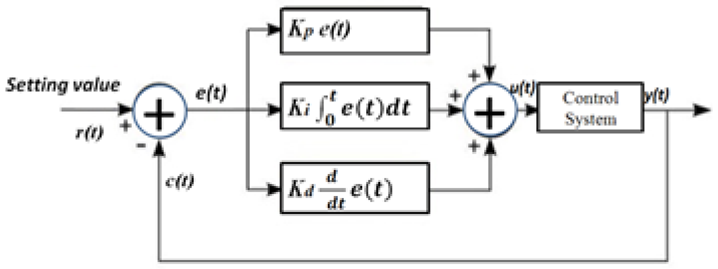


FIGURE 11. PID control system

$$u(t) = K_P \left[e(t) + \frac{1}{T_I} \int_0^t e(t)dt + T_D \frac{d}{dt} e(t) \right] \tag{2}$$

Here, K_p is a proportional coefficient, T_I is an integral coefficient, and T_D is a differential coefficient.

In the following, the digital PID controller is derived. Eq. 1 and 2 can be reformulated, respectively, as

$$e(n) = r(n) - c(n), \tag{3}$$

$$u(n) = K_P \Big\{ e(n) + \frac{T}{T_I} \sum_{i=1}^{n} e(n) + \frac{T_D}{T} \left[e(n) - e(n-1) \right] \Big\} + u_0, \tag{4}$$

where the n-th samplings in $r(t), e(t), u(t), c(t)$ are represented by $r(n), e(n), u(n), c(n)$, respectively. When the sampling period T is very small, dt can be replaced by approximation of T, $de(t$ can be replaced by approximation of $e(n) - e(n-1)$, and "integration" is replaced by the approximation of "summation.

Eq. 4 is the positional control formula and its operation mode needs to accumulate n times of the deviation e(i), not only taking more storage units, but also not easy to write a program, resulting in improving eq. 4. According to eq. 4, it is easy to see that $u(n-1)$ is expressed as

$$u(n-1) = K_P\Big\{e(n-1) + \frac{T}{T_I}\sum_{i=1}^{n-1} e(n) + \frac{T_D}{T}\big[e(n-1) - e(n-2)\big]\Big\} + u_0. \tag{5}$$

After subtracting Eq. 5 and 5, the digital incremental PID control equation is

$$\begin{aligned}\triangle u(n) &= u(n) - u(n-1)\\ &= K_P\big[e(n) - e(n-1)\big] + K_I e(n) + K_D\big[e(n) - 2e(n-1) + e(n-2)\big].\end{aligned} \tag{6}$$

From the above equations, we can obtain the digital PID position type control equation is

$$u(n) = K_P\big[e(n) - e(n-1)\big] + K_I e(n) + K_D\big[e(n) - 2e(n-1) + e(n-2)\big] + u(n-1). \tag{7}$$

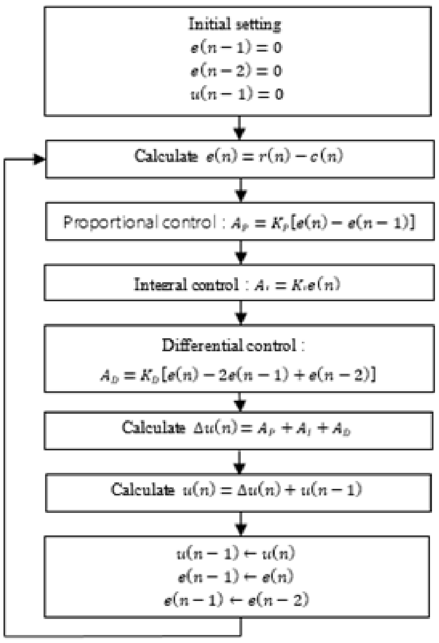

Initial setting
$e(n-1)=0$
$e(n-2)=0$
$u(n-1)=0$
Calculate $e(n)=r(n)-c(n)$
Proportional control : $A_P = K_P[e(n)-e(n-1)]$
Integral control : $A_I = K_I e(n)$
Differential control :
$A_D = K_D[e(n)-2e(n-1)+e(n-2)]$
Calculate $\Delta u(n) = A_P + A_I + A_D$
Calculate $u(n) = \Delta u(n) + u(n-1)$
$u(n-1) \leftarrow u(n)$
$e(n-1) \leftarrow e(n)$
$e(n-1) \leftarrow e(n-2)$

FIGURE 12. Flow chart for the recursively digital PID program

In Eq. 7, K_P is the proportional gain; $K_I = K_P\frac{T}{T_I}$ is the integral coefficient; $K_D = K_P\frac{T_D}{T}$ is the differential coefficient. After Eq. 7 is applied to the Arduino controller, the PID control function can be achieved as shown in Fig. 12.

3.3. **PID parameter tuning method.** In the operation of the PID control system, the PID parameters are appropriately tuned according to the system and the environmental factors to be controlled to ensure the controller is optimized. This is one of the main factors for the controller to achieve optimal control. The optimized tuning of the PID parameters is as follows: First, the integral coefficient (K_i) and the differential coefficient (K_d) are tuned so that they are all set to zero. When the control system is put into closed-loop operation, the proportional coefficient (K_p) is increased from small to large, so that the disturbance signal changes stepwise and the control process is observed until a satisfactory control process is obtained.

Second, the integral coefficient K_i is tuned. We replace the proportional coefficient, K_P, by the previously adjusted K_p value multiplied by 0.83; and increase the integral coefficient, K_i, from small to large. Similarly, let the disturbance signal change through this step until a satisfactory control process is obtained. If the integral factor K_i remains unchanged, change the proportional factor K_P to see if the control process improves. If it improves, it will continue to tune until it is satisfactory. Otherwise, the original ratio K_P increases slightly, and then the integral factor K_i is tuned to improve the control process. This test is repeated until a satisfactory scale factor K_P and an integral factor K_I are found.

In the last step, the appropriate actual differential coefficient K_D and the actual derivative time T_D are introduced. At this time, the proportional coefficient K_P and the integral coefficient K_I can be appropriately increased. As with the previous steps, the setting of the differential time also needs to be adjusted repeatedly until the control process is satisfied.

3.4. **Time optimal control.** The optimal time control of the system means to approach the shortest transition time experienced from an initial state to another state. This type of optimal switching system is also referred as a switching control (bang-bang) system.

$$u_k = \begin{cases} u_{\max} & \text{, if } e(k) > 0 \\ 0 & \text{, if } e(k) \leq 0 \end{cases} \tag{8}$$

In Eq. 8, u_k is the current output of the controller, $u_{\max}$ is the maximum output of the system, and $e(k)$ is the difference between the temperature reference value and the measured value.

The algorithm is simple and easy to achieve, but when the deviation approaches zero, the system is prone to oscillation. Therefore, the PID control algorithm and time optimal are combined as a dual-mode control, and the control rule is

$$\begin{cases} \text{Time Optimal Control} & |e(k)| \geq \epsilon \\ \text{PID Control} & |e(k)| \leq \epsilon \end{cases} \tag{9}$$

When the deviation is large, optimal time control is implemented, and PID control is implemented within a predetermined threshold value (ϵ). In this way, it can not only maximize the advantages of switching control to rapidly eliminate large deviations, but also utilize a PID control of high precision and the advantages of a small overshoot. Therefore, the performance index for this temperature control process, such as short rise time, minimum overshot and minimum setting tine will be approached as shown as in Fig. 13.

4. **Experiments and User Operation Interface.** The webpage was designed as shown in Fig. 14, where the device to be monitored can be selected conveniently. Moreover, the real-time temperature line graph is depicted. Users can directly download past historical data, set the thermostat condition or the PID parameters, and restore the default thermostat status.

In Fig. 15, the current value of the thermostat setting is shown to let the administrator understand the target temperature. The default setting range is from 23° Celcius to 50° Celcius. Once the setting is out of range, The Arduino thermostat device will not be actuated.

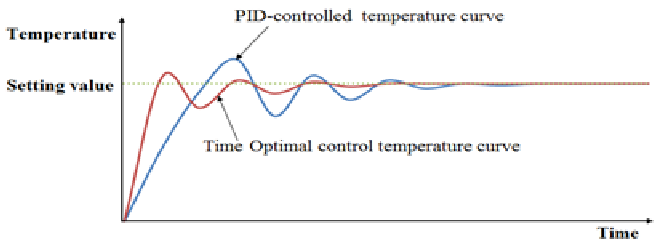
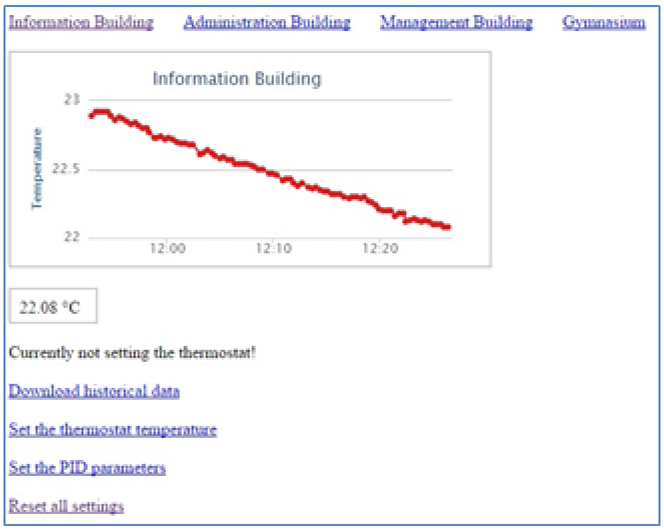


FIGURE 13. Temperature variation comparison for PID control and optimal time control



22.08 °C

Currently not setting the thermostat!

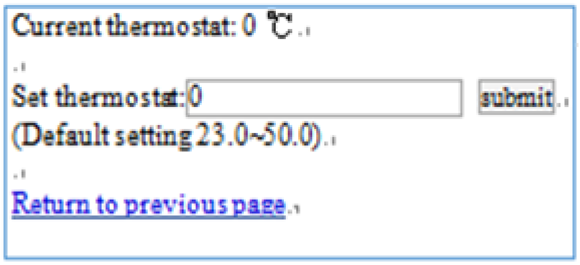
Download historical data

Set the thermostat temperature

Set the PID parameters

Reset all settings

FIGURE 14. Home page for temperature status report


Current thermostat: 0 ℃

Set thermostat: 0 submit

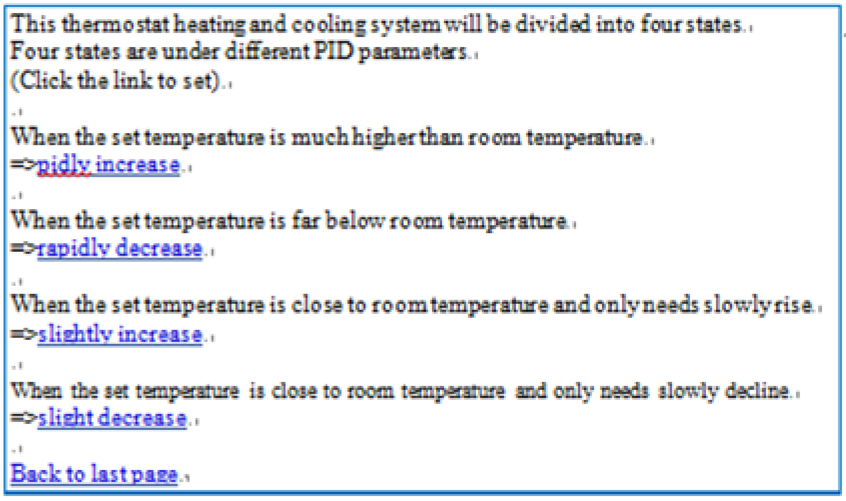
(Default setting 23.0~50.0)

Return to previous page

FIGURE 15. Web page for setting the thermostat temperature

The PID based system is divided into four states. For different situations, different PID parameters are used, so during modification, similarly, the PID parameters under four conditions are distinguished to allow users to make a selection suitable to their requirements as shown in Fig. 16. Upon entering the PID settings page, the original setting is displayed to let the user know how to conveniently evaluate and then set the required PID parameters as shown in Fig. 17.

Figure 18 shows the temperature variation before the PID parameters are optimized. When the thermostat system is directly adjusted by the PID parameters, it is necessary to


This thermostat heating and cooling system will be divided into four states.
Four states are under different PID parameters.
(Click the link to set).

When the set temperature is much higher than room temperature.
=>pidly increase

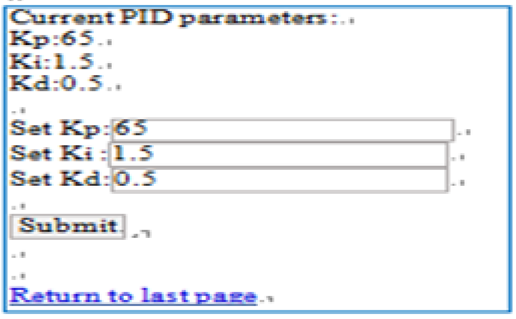
When the set temperature is far below room temperature.
=>rapidly decrease

When the set temperature is close to room temperature and only needs slowly rise.
=>slightly increase

When the set temperature is close to room temperature and only needs slowly decline.
=>slight decrease

Back to last page

FIGURE 16. Web page for deciding the temperature control policy


Current PID parameters:
Kp:65
Ki:1.5
Kd:0.5

Set Kp: 65
Set Ki : 1.5
Set Kd: 0.5

FIGURE 17. Web page for PID parameters modification

accelerate the temperature increase or decrease to reach the temperature setting. Moreover, when the temperature reaches the set value, the thermostat system must maintain a constant temperature and reduce the steady-state error. After optimizing the PID parameters in Fig. 19, it shows a better result when compared with the oscillating stability and steady-state error in Fig. 18. The default action time was set to 15 min for comparison. The heating and cooling speeds are slower before optimization. Additionally, after entering the thermostat temperature, the steady-state temperature error value became larger than the error value after optimization. The temperature variation after the PID parameter optimization can be made to quickly approach the temperature setting effectively.

After many trial tests, the system results that every set temperature could be stably reached within no more than 6 minutes, and the steady-state error does not exceed ± 0.2 degrees. Based on the statistical results, we also build set up an alert device to achieve a complete intelligent temperature control system.

For example, if the system is abnormal, it means every time, if the set temperature has been changed for more than 6 minutes, we check the temperature error between the set temperature and the actual temperature that the temperature sensor reads from the test field. If the temperature error exceeds 0.2 degrees, the system will be announced as being

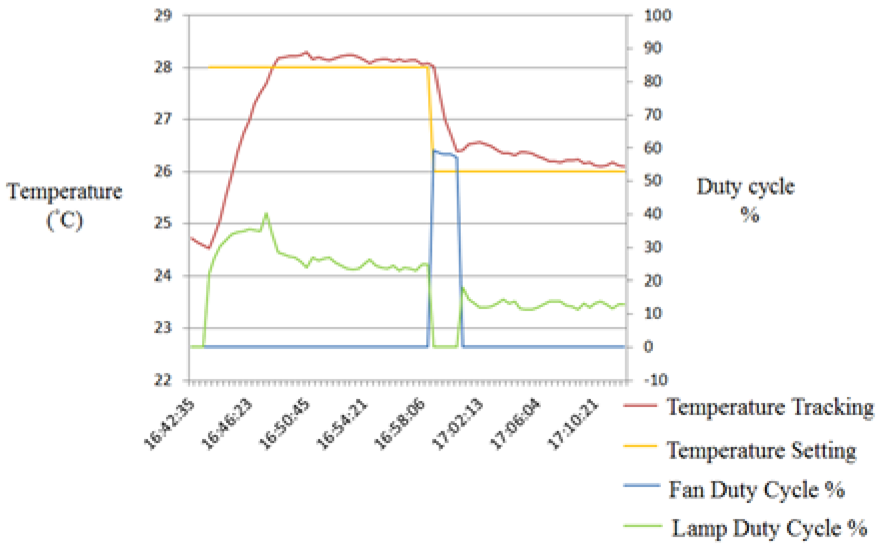


Figure 18. Web page for PID parameters modification

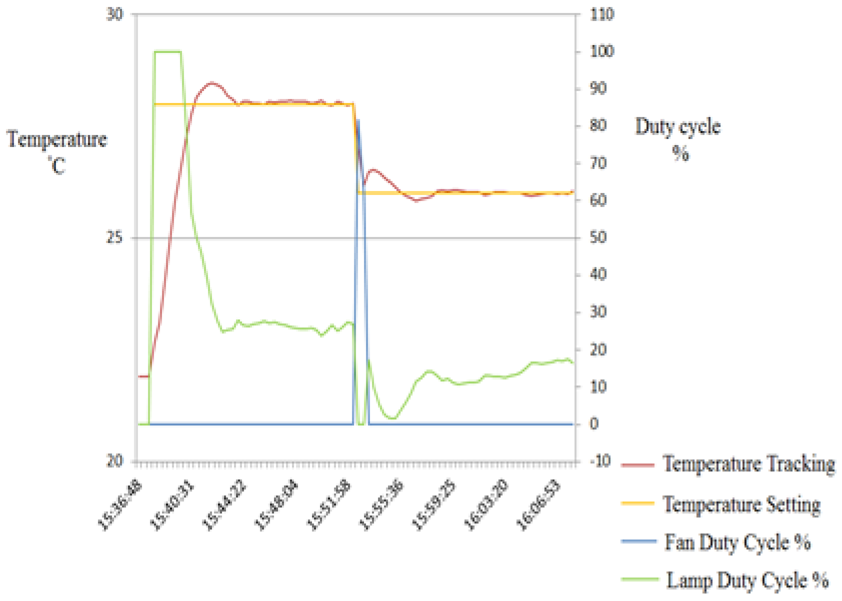


Figure 19. Web page for PID parameters modification

abnormal and the message will automatically be sent to the maintenance personnel in the operation field or the system manager.

5. **Conclusions.** Smart thermostats devices play an important role in many environments such as incubators for premature infants or high-end electronic devices. In this work, PID control technology has been applied to intelligently accomplish temperature regulation. The Arduino developer kit implanted with the developed PID algorithm is used to work with low-cost electronic components to build an IOT temperature regulation system. A research scenario is demonstrated by controlling and adjusting the lamp bulb and electric fan approves our study, which achieves a complete smart thermostat control.

Additionally, by way of establishing a cloud server and using a mobile web page, we have performed a complete monitoring system that allows the user or system manager the ability to directly monitor the temperature. Temperature information can be automatically gathered and stored in a cloud database so that the data can be further analyzed for optimization of the PID parameters under different conditions.

Acknowledgment. This work is supported by the Ministry of Science and Technology of Taiwan (106-2221-E-468 -011 -MY2). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] C. Wang, Thermostat fuzzy control system based on MCU, *Intelligent Information Technology Application, IITA 2009.* vol.1, pp. 462-463, 2009.

[2] A. Matthew, B. Sam, R. Kevin, Smart Thermostat, *Electrical Engineering Senior Theses 16*, Santa Clara University, https://scholarcommons.scu.edu/elec_senior/16, 2015.

[3] M. Soliman, T. Abiodun, T. Hamouda, J. Zhou; C. H. Lung, Smart home: integrating internet of things with web services and cloud computing, *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference*, vo. 2, pp. 317-320, 2013.

[4] M. Schmidt, Arduino: a quick-start guide, *Pragmatic Bookshelf*, 2011.

[5] E. H. Goud, A. Harshika, G. Akhil, D. Charishma, K. Bhupathi, I. K. Swamy: real time based temperature control using arduino, *International Journal of Innovations in Engineering and Technology (IJIET)*, vol. 8, pp. 209-216, http://dx.doi.org/10.21172/ijiet.82.030, 2017.

[6] H. Z. Chen and J. R. Chen, PHP7 and MySQL for Web Development, (Gotop Information Co., Ltd., Taipei, 2014).

[7] M. Kando, Partial discharges using a zero cross AC voltage, *Sixth International Conference on Dielectric Materials, Measurements and Applications*, pp. 354-357, 1992.

[8] H. Xiang, K. Wang and Z. Li, Monitored control system of temperature/humidity for ammunition storehouse based on LabVIEW, *First International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC)*, pp. 172-175, 2011.

[9] Q. M. Ashraf, M. I. M. Yusoff , A. A. Azman, N. M. Nor, N. A. A. Fuzi, M. S. Saharedan and N. A. Omar, Energy monitoring prototype for Internet of Things: Preliminary results, *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum*, pp. 15, 2015.

[10] Y. J. Zhao, Interactive and Graphical Design Entry for Arduino, (Flag Publishing Co., Taipei, 2013).

[11] A. Visioli, Tuning of PID controllers with fuzzy logic,*IEEE Proc. on Control Theory and Applications*, 1, pp. 1-8, 2001.

[12] S.Wang, H.Cao, Temperature control system research for double roller heat sealing machine based on the Fuzzy-PID control, *2nd International Conference on Electrical and Electronics: Techniques and Applications (EETA 2017)*, pp. 167-172, 2017.

[13] D. Wang , X. J. Dai: new digital thermostat development, *International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pp. 376-379, 2015.

[14] G. C. Hsieh , J. F. Yan, Group-asymmetrical PWM controller for dimmable fluorescent lamp ballast without striation and thermostat effect, *Industrial Electronics Society, IECON 2005. 31st Annual Conference of IEEE, Raleigh*, 2005.

[15] S. Wei , B. Wu, F. Li and C. Liu, A general space vector PWM control algorithm for multilevel inverters, *Applied Power Electronics Conference and Exposition, Eighteenth Annual IEEE*, 1, pp. 562-568, 2003.