An Arduino-based Low-Cost Hardware for Temperature Control

AMINE DAOUD
Department of Electronics, Faculty of Electrical Engineering
University of Sciences and Technology of Oran -Mohamed Boudiaf-
BP 1505 El Mnaouer Oran 31000
ALGERIA

Abstract: - This paper describes a simple digital temperature control system where the popular LM35DZ sensor is used to sense the temperature of a resistive heating element and to provide a feedback to an Arduino microcontroller board. By subtracting the measured temperature from the desired temperature, a value of error is then processed by a conventional PID controller which provides an adjustment signal and drives a bipolar junction transistor in order to minimize or remove that error by a simple ON/OFF switching of the power to the heating element. The control is also done using logic controllers such as ON/OFF and hysteresis controllers. The evaluation of each controller performance is based on rise time, overshoot, steady state error and the most common performance criterion which is the integral of the absolute value of the error.
Besides that, the arduino integrated development environment (IDE) software is used to write the program code. Moreover, other languages are used to program the Arduino such as Matlab support package for Arduino hardware, Flowcode IDE and mBlock.

Key-Words: - Arduino, LM35, PID, Temperature control, Heating element, Graphical programming, Matlab.

Received: November 11, 2020. Revised: March 1, 2021. Accepted: March 31, 2021. Published: April 10, 2021.

1 Introduction

The subject of automatic control or automation has always played an important role in advancement of modern science and improvement of engineering skills. The applications are found in several fields of engineering and technology, ranging from home appliances to industrial processes control with practical applications such as engines, production machines, transportation systems, power systems, robotics, temperature limiting systems and many more.

Process controller is a key element of any control loop in the process industry. The conventional PID controller remains by far the most employed in automatic process control applications in industry due to its simplicity and ease of implementation. Furthermore, 90% control systems use PID or any of its improved versions as feedback controller [1] to regulate flow, pressure, level, acidity, moisture, temperature, and many other industrial process variables. Thus, temperature control represents a typical application that uses the PID control.

Several methods have been reported in the literature for the control education experiments. A teaching/learning PID control experiment for undergraduate students based on the temperature control is developed in [13]. References [11][14][17] provide fundamental control laboratories based on Arduino tool, which is utilized as a digital controller for the undergraduate control system course. Reference [12] presents a system of wireless temperature and humidity monitoring based on Arduino Uno platform, which has the capability of collecting environmental factors such as temperature and humidity data real-timely and remotely. In order to improve the accuracy of temperature control and reduce overshoot, more advanced approaches have been proposed in [15][16][18][19].

Today, most controllers are implemented digitally or with microcontroller like Arduino which constitutes the brain of thousands of projects. Arduino is an open-source platform, which over the years has become a very popular choice due to its flexibility and low cost. In this scope, an inexpensive Arduino-based PID controller is designed and implemented with different programming environments in order to control the temperature of a resistor as a heating element with an analog temperature sensor.

The research paper is organized as follows. Section 2 presents a brief description of the different conventional control techniques. Section 3 is dedicated to the software description. In section 4, the hardware architecture of a simple thermostat is introduced. The results of the tests are given in section 5, followed by a conclusion in section 6.

2 Control Techniques

In control theory, there are several control laws used, most of them being ON/OFF control method, hysteresis control method, and PID control method and its derivates. Advanced process control strategies include fuzzy control, neuro-fuzzy control, sliding mode control, etc.

2.1 ON/OFF and Hysteresis Control Techniques

The ON/OFF control technique is the simplest form of a feedback mechanism and can be described as follows:

$$u = \begin{cases} u_{max}, & if\ error \geq 0 \\ u_{min}, & if\ error < 0 \end{cases} \quad (1)$$

where u is the actuation command and error is the difference between the set-point (the reference signal) and the output of the process (Fig.1). The ON/OFF control uses only a single reference when input is below the set-point, the output is turned ON, and then it is turned OFF when above the reference.

The ON/OFF control technique is simple, inexpensive and there are no parameters to choose. However, it typically results in a process where the output oscillates around the set-point. In this case, the output of the process will continuously switch around the reference and if the hysteresis band is not correctly set, the deviation from the reference could be quite significant.

ON/OFF and hysteresis controllers are from the logic controller family. The hysteresis controller is common for thermostats, where high and low thresholds are set. If the output of the process is below the lower threshold, the controller turns ON the output and vice versa for the higher threshold, creating a band that output must be kept within. Thus, setting the hysteresis controller is very easy with no need to have a detailed model of the process.

2.2 PID Control Technique

Traditional PID control technique is a conventional control strategy. Because of its simplicity, strong adaptability, robustness over wide range, and ease of implementation, it is still widely used in industrial control. The discrete form of the conventional PID control algorithm is as follows [2][4][8]:

$$u(k) = K_P e(k) + K_I \sum_{i=0}^{k} e(i) + K_D[e(k) - e(k-1)] \quad (2)$$

where u(k) is the output from the controller, e(k) is the error value in kth sampling and k is the sampling number (k=0,1,2,3,..,n). K_P, K_I, and K_D represent the gains of the three controllers (P: proportional controller, I: integral controller, D: derivative controller). As shown in Fig.1 the PID output consists of the sum of the outputs of the three types of controllers.

The performance of the PID controller greatly depends on the tuning of the K_P, K_I, and K_D gains, which represents a substantial limitation. In general, the design procedure for finding suitable values for the gains of the PID controller is a trial and error. There is, however, a wide variety of PID tuning methods that have been proposed in the literature, such as Ziegler-Nichols and others. The Ziegler-Nichols' closed-loop method is a PID tuning method which sets K_I and K_D to zero. After that, K_P is increased from zero continuously until there are sustained oscillations in the process measurement. The gains of the PID are then calculated by measuring the period of the oscillation (T_u) and setting K_P, K_I and K_D to $0.6K_{Pu}$, $1.2K_{Pu}/T_u$ and $3K_{Pu}T_u/40$, respectively. In this case K_{Pu} represents the ultimate gain value of k_p at the stability limit of the system.

In the case of a thermostat, a temperature sensor is used to measure the actual temperature and to provide feedback to the control loop. Based on this feedback, the PID controller calculates the difference between the reference and the actual temperature and uses this value of error to determine how much power to supply to the heater, in order to keep constant the value of the measured temperature.

In this study, Integral of Absolute Error (IAE) criterion is used to measure the increased error in the system. IAE is defined as [8]:

$$IAE = \int_0^{\infty} |e(t)| dt \quad (3)$$

where e is the error of the system and t is the time period.

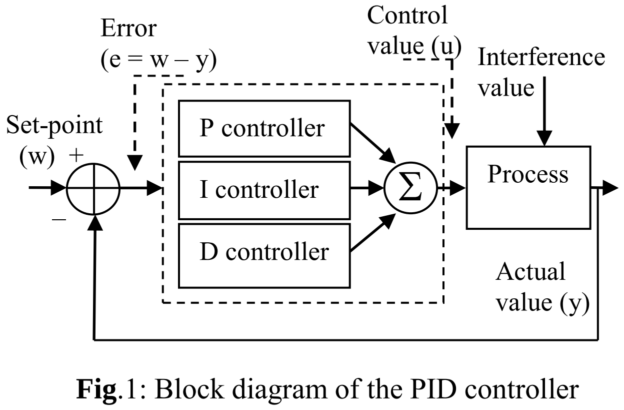


Fig.1: Block diagram of the PID controller

3 Software Description

In the following, a brief description of the main characteristics of the various types of programming tools examined in this study is provided.

3.1 Matrix Flowcode

Flowcode is a perfect flowchart language based on graphical programming IDE. This novel programming software is designed to allow those new to microcontroller programming to design programs quickly with little or no knowledge of any higher level languages. Moreover, programs written in C and Assembly code can be easily embedded in Flowcode using the code icon. The C code is not checked by Flowcode, but is passed on to the microcontroller during compilation [10]. Assembly code can be added into the code field in a C assembly code wrapper. Therefore it is possible to take complex and sophisticated programs written in C or assembly and embed them into our designs.

3.2 Arduino IDE

The Arduino IDE is cross-platform application which supports the languages C and C++. It is principally used for coding, compiling and uploading the binary code in the Arduino boards. Program written using Arduino IDE is called sketch. It is based on two main functions: setup() and loop(). The setup function is a sort of pre-setting function which runs only once we run code every time and the loop function is the one which runs continuously throughout the running of the code.

3.3 mBlock

Arduino is designed to be programmed in its own Arduino language, which consists of functions pulled from C/C++. However, other languages can also be used to program an Arduino such as Snap4Arduino, ArduBlock, mBlock, and others. mBlock is a graphical programming language that is used to program the Arduino boards in an easy and interactive way. In mBlock, labeled blocks can be joined to write a complete program, which turns coding into a more visually interesting process.

3.4 MATLAB-Arduino

Arduino I/O package from Mathworks, enables MATLAB to communicate with Arduino hardware over USB cable. This package is mainly based on a server program running on the Arduino board which executes commands from a PC over the bidirectional serial communication (UART).

4. Hardware Architecture

The proposed system includes an Arduino Uno board, a resistor as a heating element, and low-cost analog temperature sensor to implement temperature control techniques based on simple thermostat architecture. The Arduino Uno board is designed around an 8-bit AVR RISC-based microcontroller (ATMEGA328P) with a clock speed of 16MHz provided by a precise crystal oscillator. This board has 32KB flash program memory, 14 digital input/output pins which provide PWM outputs, 06 built-in analog-to-digital channels which convert analog signals in the range of 0-5V with the resolution of 10 bits, a UART serial port, an ICSP header, and a USB serial communication module [2]. Furthermore, the Arduino Uno is a user friendly open-source hardware platform.

The LM35 temperature sensor provides a linear output voltage of 10mV/°C and covers the range of 0°C to 100°C with a typical accuracy of 0.5°C at +25°C. The temperature sensor is connected to analog pin A0 which is used to measure the temperature of the heating resistor [3][5][6]. The error signal is then obtained by calculating the difference between the set point and the measured value received from LM35. This error is processed by the temperature controller and accordingly, the output signal is generated by pin D9 and applied between anode and cathode pins of an opto-isolator (4N25) (Fig.2). The latter drives a bipolar junction transistor (BD137) and hence switches ON or OFF the transistor in order to achieve the desired range of temperature.

A 230V AC to 9V AC center-tap transformer steps down the main voltage. Then, the bridge rectifies it and capacitor filters it. The output across the 1000μF capacitor is given as input to the 7808 voltage regulator IC which produces a regulated 8V DC.

The LM35 temperature reading can be displayed on serial monitor (baud rate: 115200 bps, 8 data bits, no parity, and 1 stop bit) and recorded every second. Also, a 2x16 character LCD module was used to display the sensed temperature (Fig.3). As can be seen from Fig.4, the Arduino IDE serial data plotter is used to visualize data as waveforms.

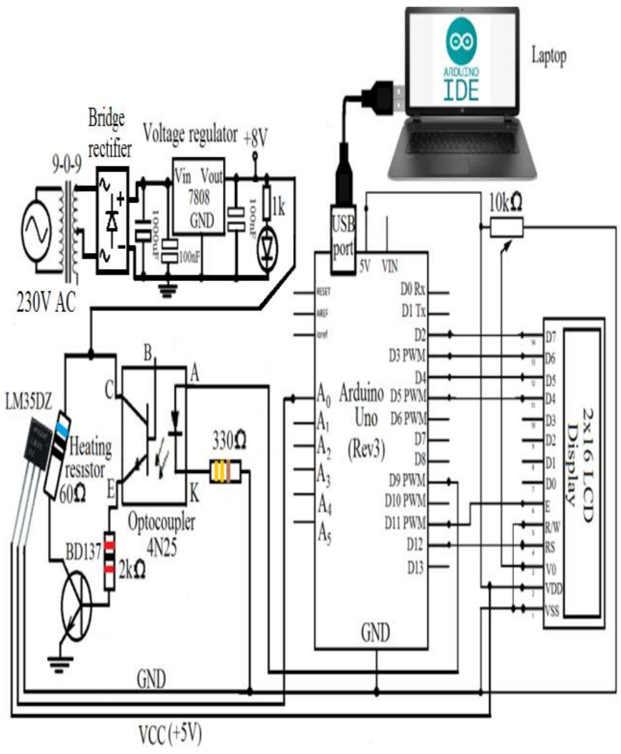


Fig.2: Temperature control circuit schematic.

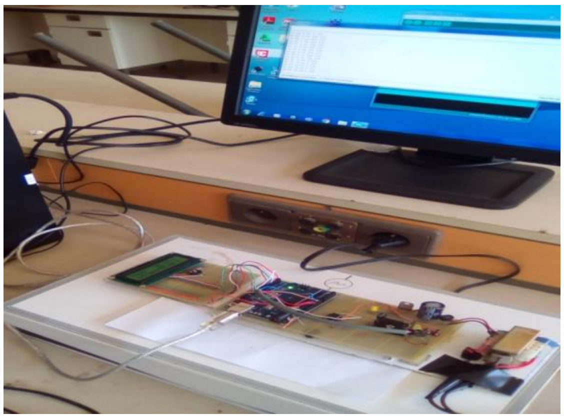
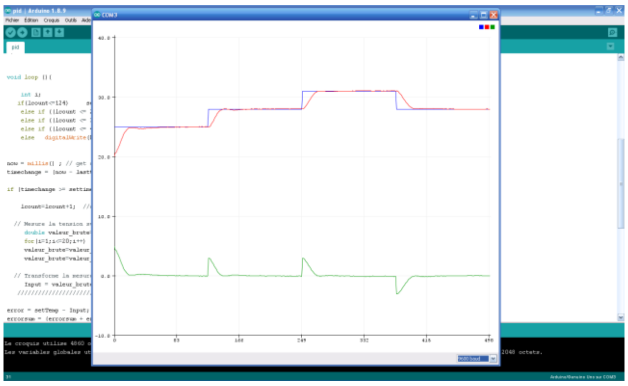


Fig.3: Experimental setup.



```
void loop ()(

   int i;
  if(lcount<=124)     se
   else if ((lcount <=
   else if ((lcount <=
   else if ((lcount <=
   else   digitalWrite(

now = millis() ; // get
timechange = (now - last

if (timechange >= settin

   lcount=lcount+1;  //

  // Mesure la tension s
    double valeur_brute
    for(i=1;i<=20;i++)
    valeur_brute=valeur
    valeur_brute=valeur

  // Transforme la mesur
    Input = valeur_brute
  /////////////////////

error = setTemp - Input;
errorsum = (errorsum + e
```
```
Le croquis utilise 4860 o
Les variables globales ut
```
Fig.4: The Arduibo IDE serial data plotter.

5. Results and Discussion

In this section, basic process control techniques will be implemented on Arduino Uno board for testing purposes in order to validate the proposed control system. An analog sensor detects temperature of resistance heater and provides feedback to the Arduino hardware which runs a control program developed with the Arduino IDE. Fig.5 shows ON/OFF control, hysteresis control, and PID control system responses with the set points of 35°C, 33°C, and 38°C. From the previous figure, the values of rise time, overshoot and IAE of ON/OFF and PID controllers for set point of 35°C can be seen in Table 1. In these tests, there is a slightly larger overshoot produced by on/off control algorithm with oscillations around set point of 35°C. Hysteresis control algorithm produces oscillations of about 1°C higher than that of its final values of 35.5°C and 34.5°C.

LM35 is used to read temperature with voltage output of about 10mV for every °C. For default analog reference of 5V, we can only get half °C reading accuracy. Arduino Uno has a built-in internal reference with voltage value equal to 1.1V which provides more accurate temperature readings. Therefore, ADC reference is changed in setup function with the command “analogReference(),” with the argument “INTERNAL”. Moreover, the function “analogWrite()” is used in order to regulate the PWM duty cycle ratio quickly with the default switching frequency of about 490Hz (Fig.6). This function is called by the PID algorithm to write an analog value to pin D9 and its input argument is an integer between 0 and 255 [7][9]. However, for on/off and hysteresis algorithms, the control signals are based on the function “digitalWrite()” which writes a high or a low value to a digital pin D9 (Fig. 7 and Fig.8).

Based on the results given in Fig.9 and Table 1, it can be seen that in terms of both rise time and steady state error performances, PID can have the smallest steady state error values, while both on/off and hysteresis controllers seem to be the fastest and have almost oscillations around the above mentioned set points. In addition, on/off and hysteresis sketches use 13% of program storage space, whereas global variables use 11% and 12% respectively of dynamic memory. While PID sketch uses 15% of program storage space and global variables use 13% of dynamic memory.

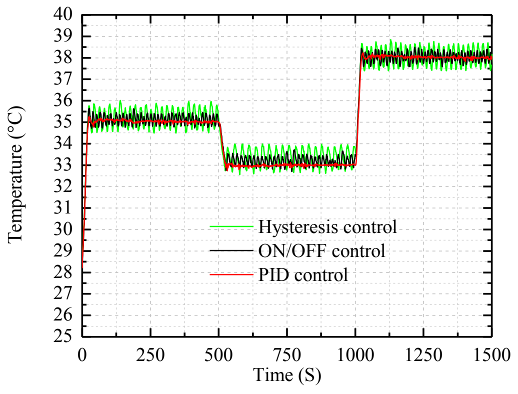


Fig.5: Comparison result at different temperature set points.

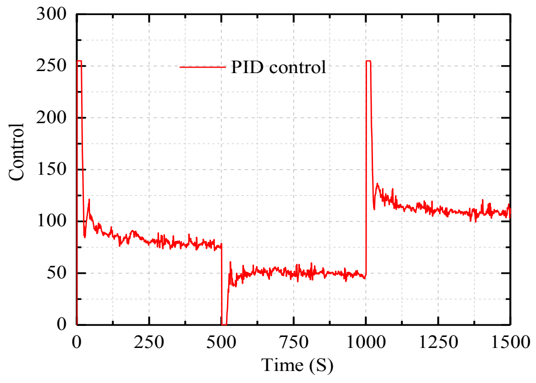


Fig.6: Duty cycle of the PWM signal with PID control method.

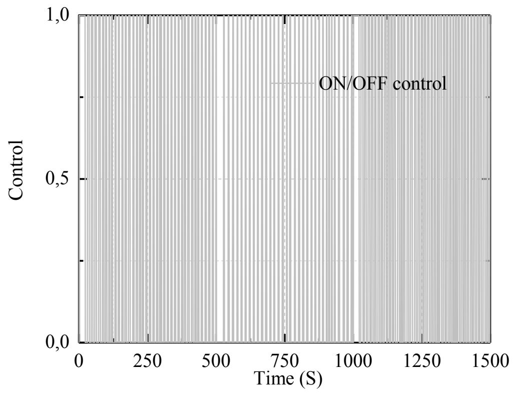


Fig.7: Digital control signal with on/off control method.

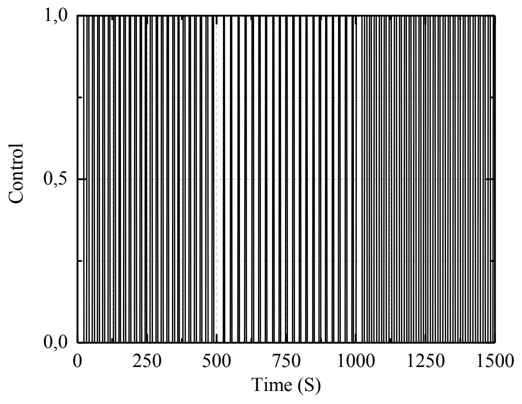


Fig.8: Digital control signal with hysteresis control method.

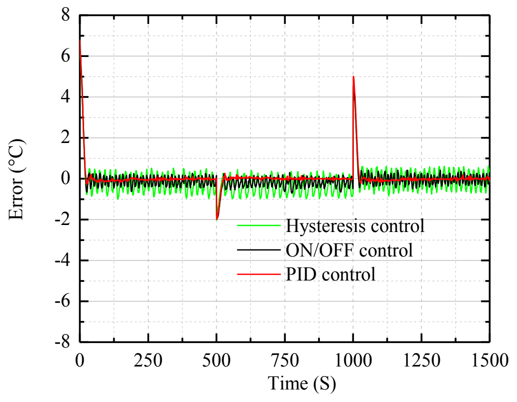


Fig.9: Curves of the tracking errors.

Table 1: Performance comparison of different controllers in the case of Arduino IDE.

	Rise Time (S)	**Overshoot (%)**	**IAE**
on/off control	17.1	0.71	173.86
PID control	17.8	1.85	101.79

From the experimental results in figures 10 and 13, it is easy to see which control technique is the most effective on the basis of rise time, overshoot and steady state error. Furthermore, it can be seen from figures 11 and 14 that the PID controller can successfully reach the set points and have the smallest steady state error values. As shown in figures 12 and 15, the best controller is indicated by the lowest IAE.

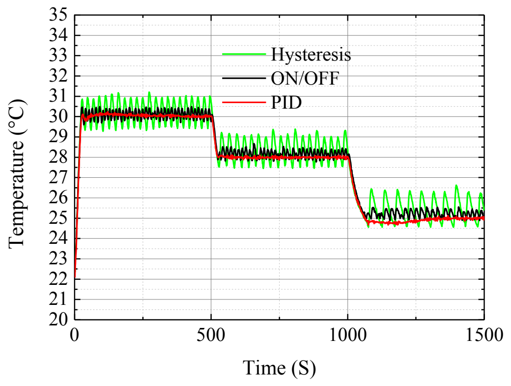


Fig.10: Comparison result at different temperature set points.

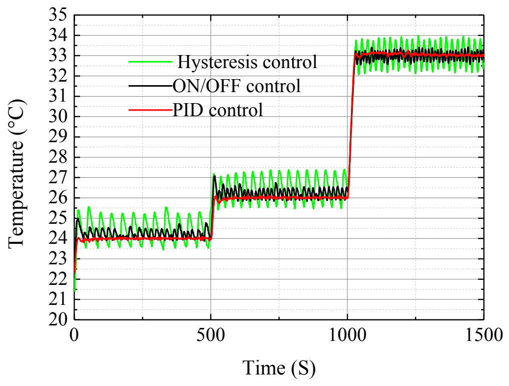


Fig.13: Multistep responses obtained with ON/OFF, hysteresis and PID controllers.

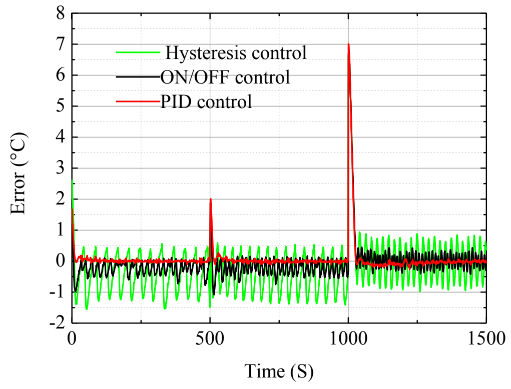


Fig.14: Curves of the tracking errors.

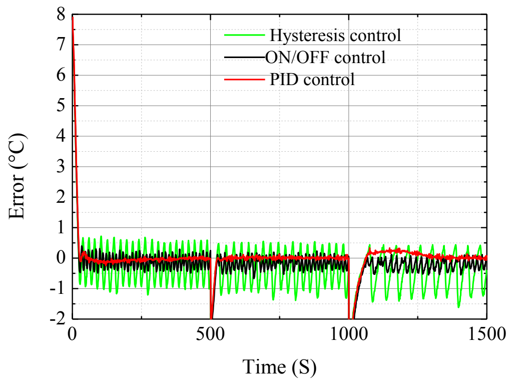


Fig.11: Curves of the tracking errors.

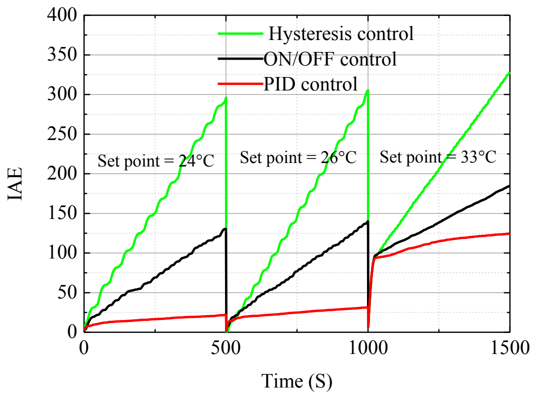


Fig.15: Curves of the integral of the absolute error.

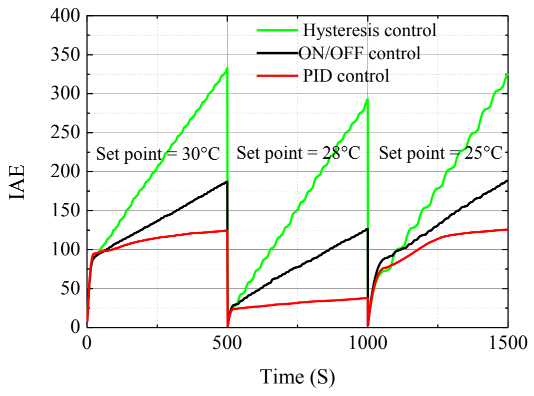


Fig.12: The integral curves of the absolute value of the error.

The second tool discussed in this paper comes from the Matrix Technology Solutions Limited. It is based on visual programming language that can facilitate learning by manipulating program elements graphically on a flowchart as it is depicted in Fig. 18, Fig. 19 and Fig.20. From the output waveforms in Fig.16, it can be seen that PID controller has emerged to be best in terms of performance with very small steady state errors as it is reported in Fig. 17 and Table 2. Control signals and PWM duty cycle ratio are illustrated in Fig.21, Fig. 22 and Fig. 23, respectively. In terms of memory size, on/off and hysteresis sketches use 15% of program storage space. PID sketch uses 17% of program storage space.

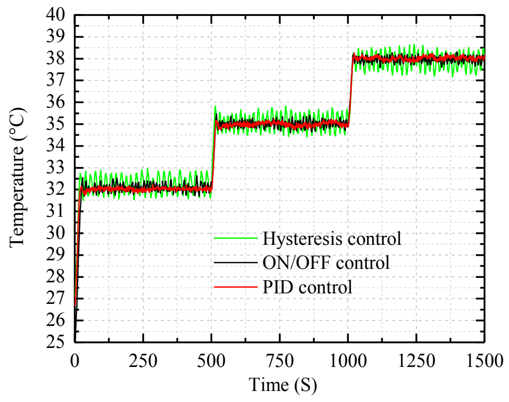


Fig.16: Multistep responses obtained with ON/OFF, hysteresis and PID controllers.

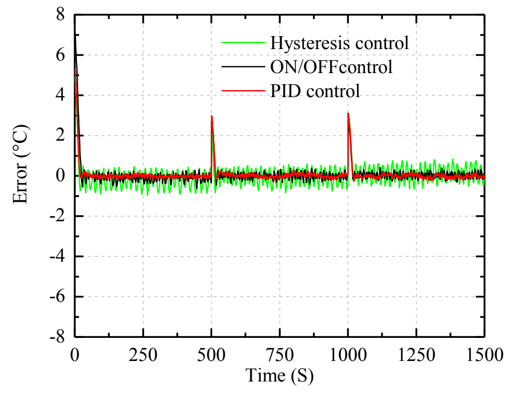


Fig.17: Curves of the tracking errors.

Table 2: Performance comparison proposed control methods controllers in the case of flowcode IDE.

	Rise Time (S)	**Overshoot (%)**	**IAE**
on/off control	11.2	1.71	161.79
PID control	11.7	0.46	82.33

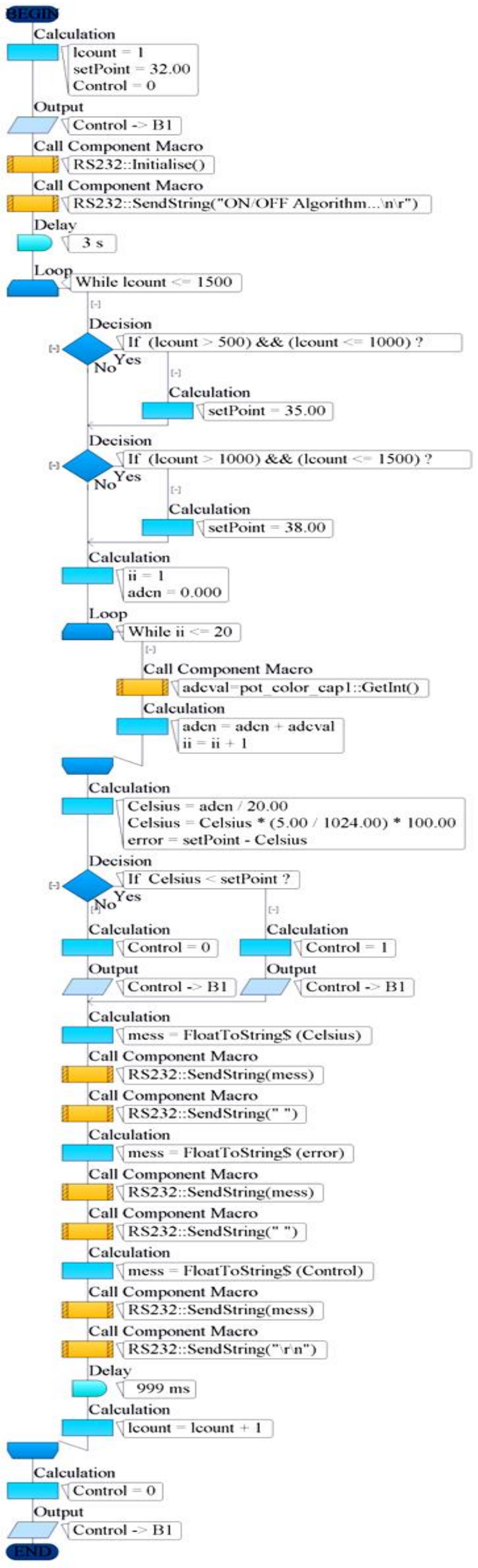


Fig.18: Flowchart of the ON/OFF control method.

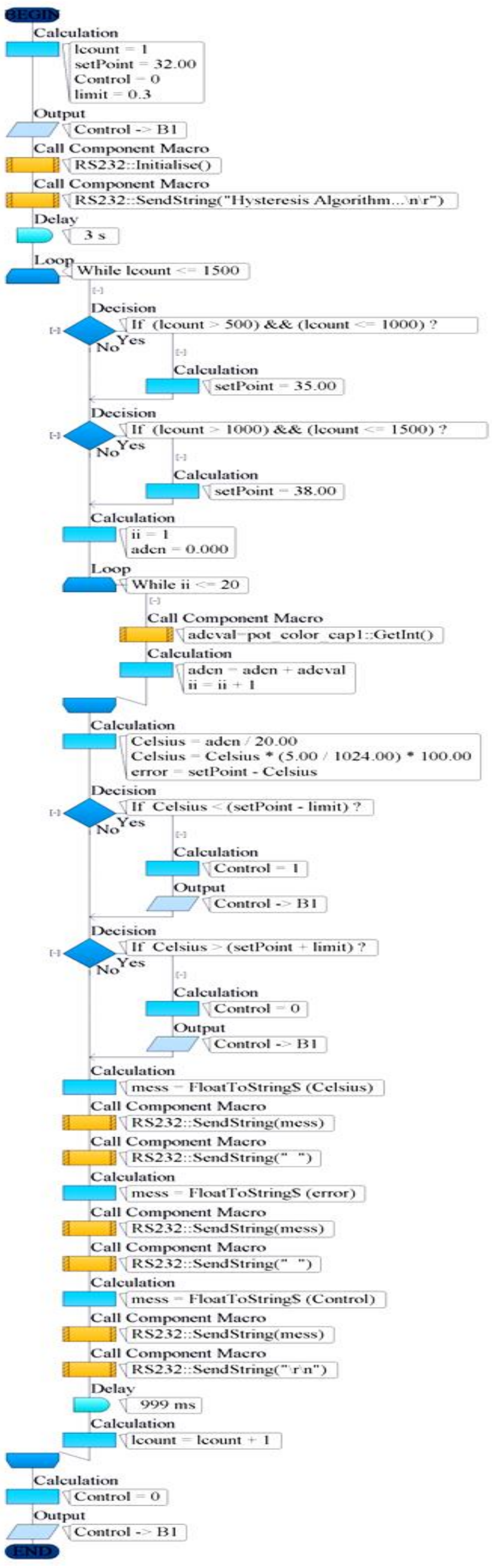


Fig.19: Flowchart of the hysteresis control method.

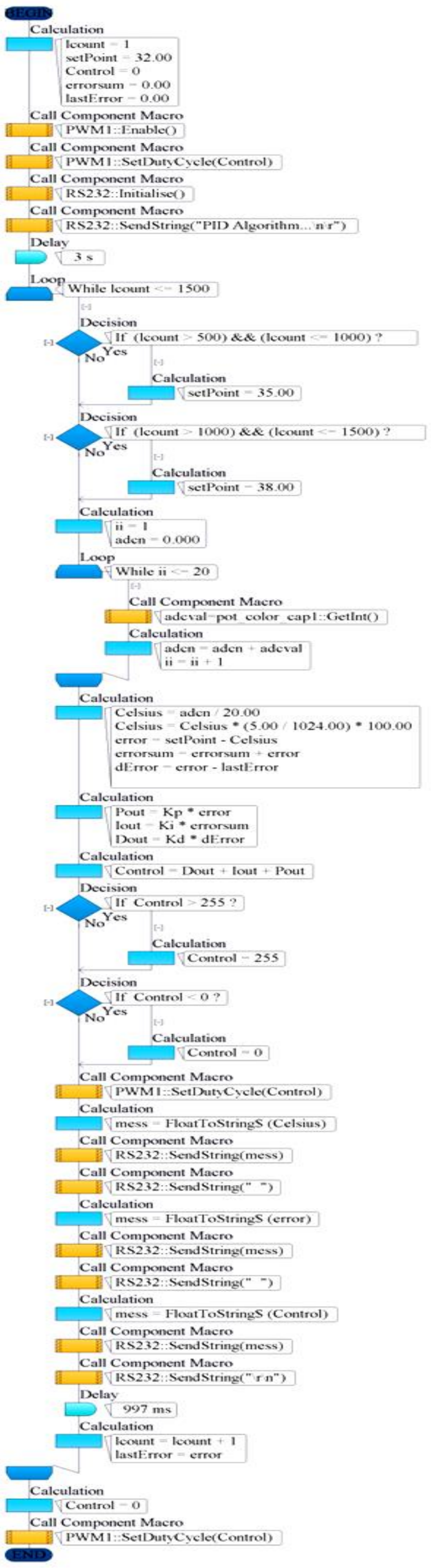


Fig.20: Flowchart of the PID control method.

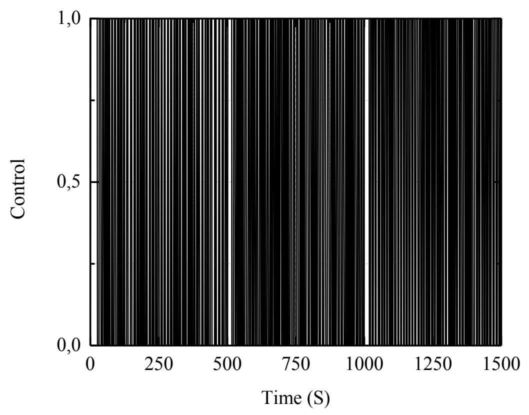


Fig.21: Digital control signal with on/off control method.

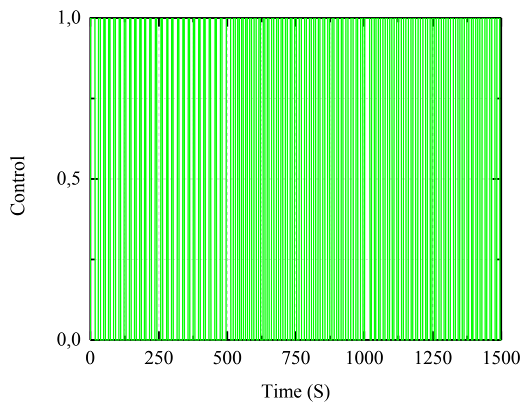


Fig.22: Digital control signal with hysteresis control method.

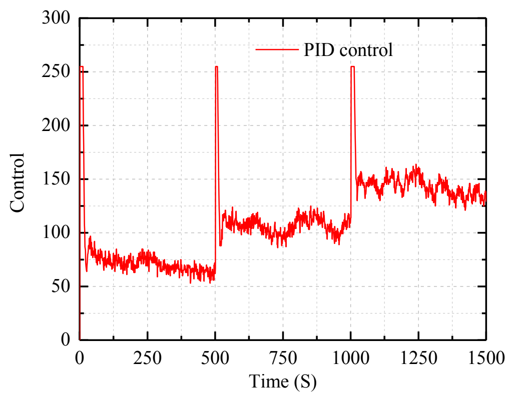
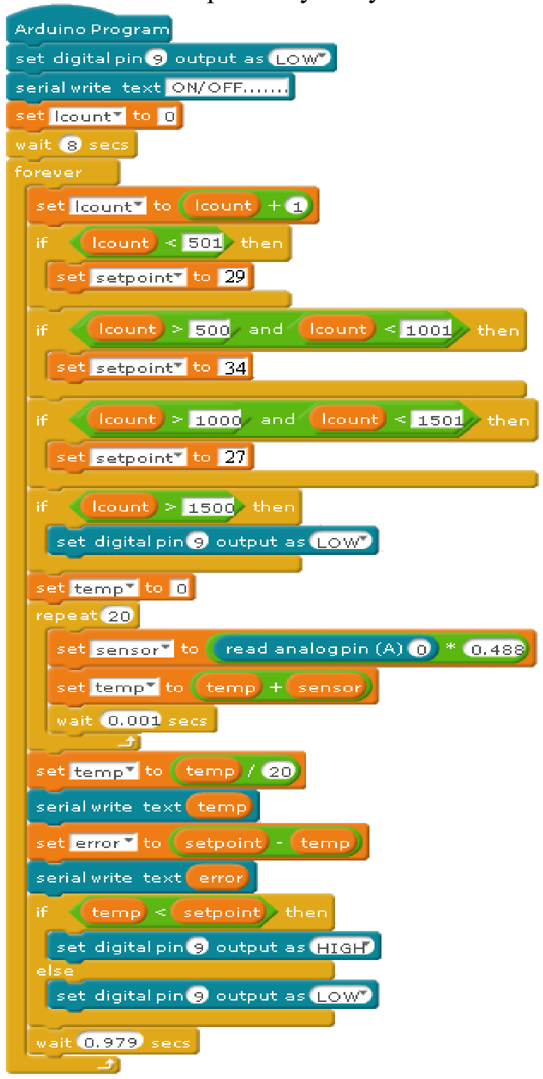


Fig.23: Duty cycle of the PWM signal with PID control method.

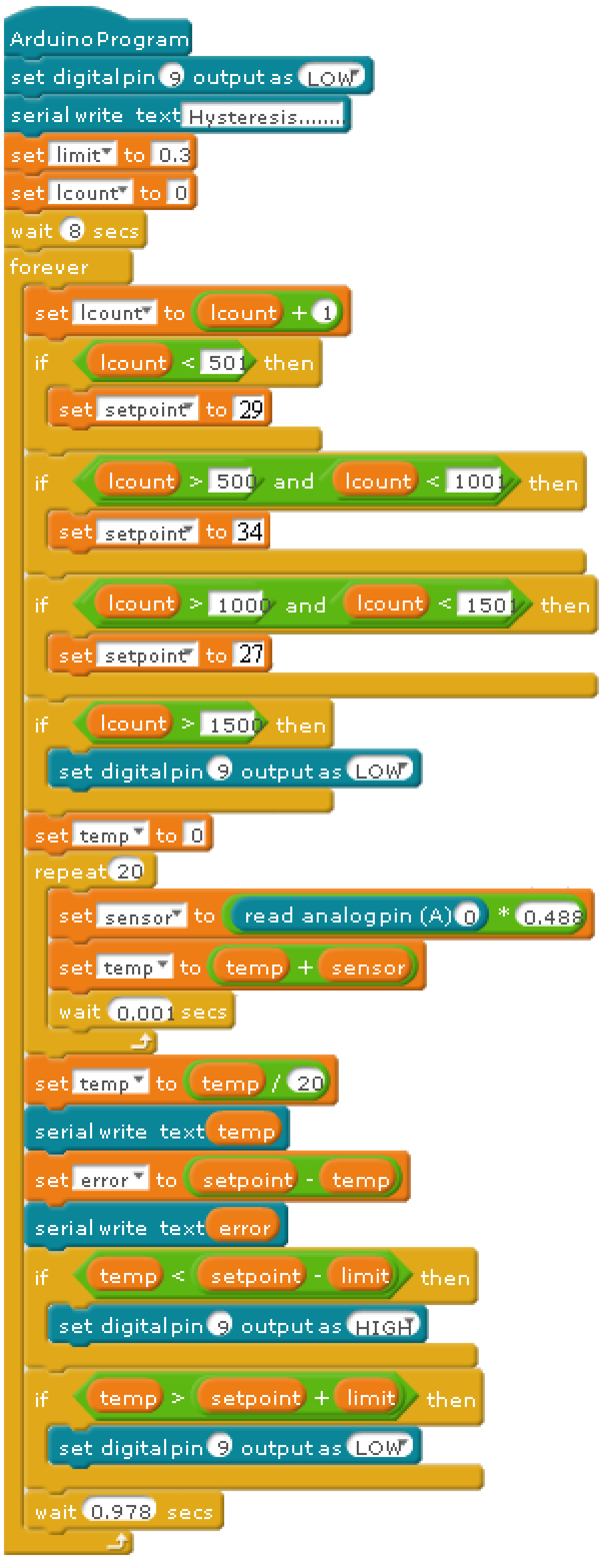
The third tool presented in this paper is the mBlock 3, developed by Makeblock Co., Limited. Thus, the Arduino Uno board is programmed using the mBlock software through block-based coding platform, which turns coding into a visually scratch programming language. As shown in Fig.24, Fig.25, and Fig.26, the proposed control techniques are represented in the form of blocks using the scratch programming tool. From Fig.27 and Fig.28, the results show that the PID controller can guarantee good performance with small steady-state errors and a good tracking capability (Table 3). Duty cycle ratio and control signals are given in Fig.29, Fig.30 and Fig.31 respectively. In terms of memory size, PID sketch uses 21% of program storage space and global variables use 26% of dynamic memory. While on/off and hysteresis sketches use 19% of program storage space, whereas global variables use 24% and 25% respectively of dynamic memory.

ON/OFF control



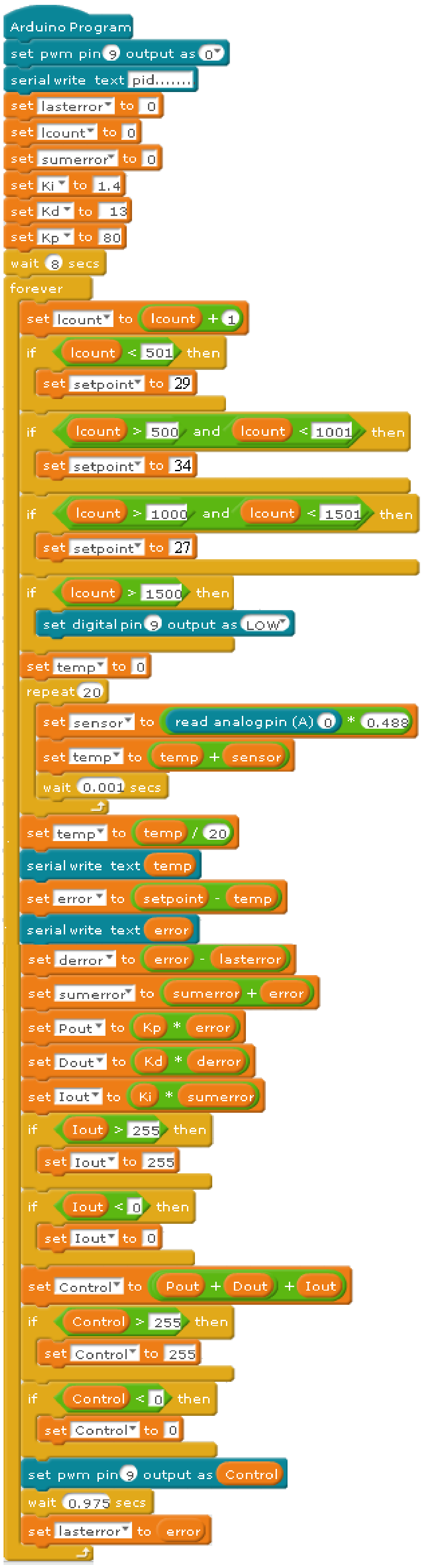
```
Arduino Program
set digital pin 9 output as LOW
serial write text ON/OFF......
set Icount to 0
wait 8 secs
forever
    set Icount to Icount + 1
    if Icount < 501 then
        set setpoint to 29
    if Icount > 500 and Icount < 1001 then
        set setpoint to 34
    if Icount > 1000 and Icount < 1501 then
        set setpoint to 27
    if Icount > 1500 then
        set digital pin 9 output as LOW
    set temp to 0
    repeat 20
        set sensor to read analogpin (A) 0 * 0.488
        set temp to temp + sensor
        wait 0.001 secs
    set temp to temp / 20
    serial write text temp
    set error to setpoint - temp
    serial write text error
    if temp < setpoint then
        set digital pin 9 output as HIGH
    else
        set digital pin 9 output as LOW
    wait 0.979 secs
```

Fig.24: Block-based coding of the ON/OFF control method.



```
Arduino Program
set digital pin 9 output as LOW
serial write text Hysteresis.......
set limit to 0.3
set lcount to 0
wait 8 secs
forever
    set lcount to lcount + 1
    if lcount < 501 then
        set setpoint to 29
    if lcount > 500 and lcount < 1001 then
        set setpoint to 34
    if lcount > 1000 and lcount < 1501 then
        set setpoint to 27
    if lcount > 1500 then
        set digital pin 9 output as LOW
    set temp to 0
    repeat 20
        set sensor to read analog pin (A) 0 * 0.488
        set temp to temp + sensor
        wait 0.001 secs
    set temp to temp / 20
    serial write text temp
    set error to setpoint - temp
    serial write text error
    if temp < setpoint - limit then
        set digital pin 9 output as HIGH
    if temp > setpoint + limit then
        set digital pin 9 output as LOW
    wait 0.978 secs
```

Fig.25: Block-based coding of the hysteresis control method.



```
Arduino Program
set pwm pin 9 output as 0
serial write text pid.......
set lasterror to 0
set lcount to 0
set sumerror to 0
set Ki to 1.4
set Kd to 13
set Kp to 80
wait 8 secs
forever
    set lcount to lcount + 1
    if lcount < 501 then
        set setpoint to 29
    if lcount > 500 and lcount < 1001 then
        set setpoint to 34
    if lcount > 1000 and lcount < 1501 then
        set setpoint to 27
    if lcount > 1500 then
        set digital pin 9 output as LOW
    set temp to 0
    repeat 20
        set sensor to read analog pin (A) 0 * 0.488
        set temp to temp + sensor
        wait 0.001 secs
    set temp to temp / 20
    serial write text temp
    set error to setpoint - temp
    serial write text error
    set derror to error - lasterror
    set sumerror to sumerror + error
    set Pout to Kp * error
    set Dout to Kd * derror
    set Iout to Ki * sumerror
    if Iout > 255 then
        set Iout to 255
    if Iout < 0 then
        set Iout to 0
    set Control to Pout + Dout + Iout
    if Control > 255 then
        set Control to 255
    if Control < 0 then
        set Control to 0
    set pwm pin 9 output as Control
    wait 0.975 secs
    set lasterror to error
```

Fig.26: Block-based coding of the PID control method.

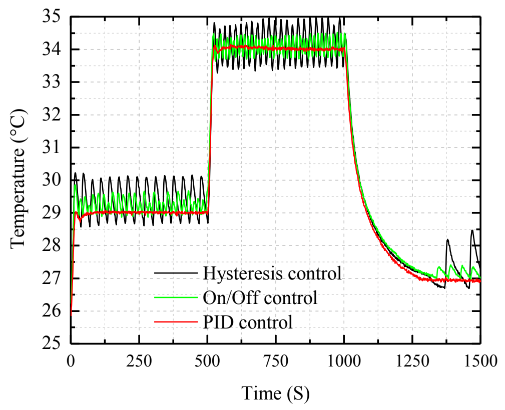


Fig.27: Comparison result at different temperature set points.

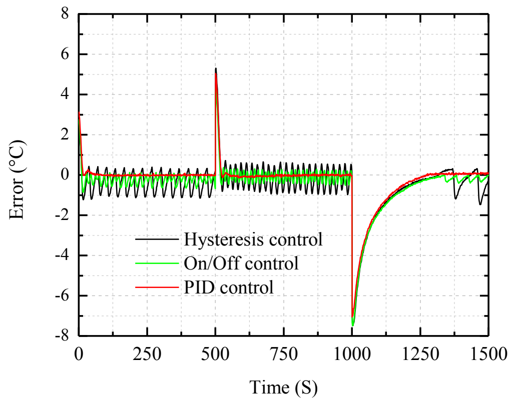


Fig.28: Curves of the tracking errors.

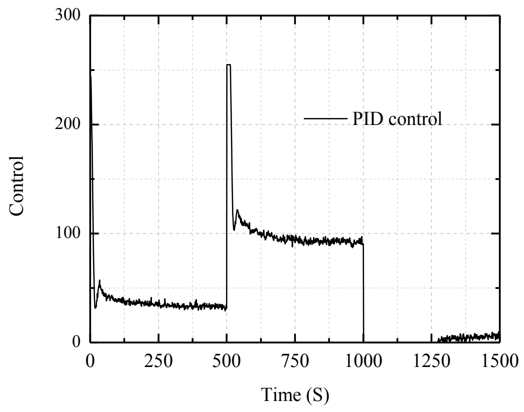


Fig.29: Duty cycle of the PWM signal with PID control method.

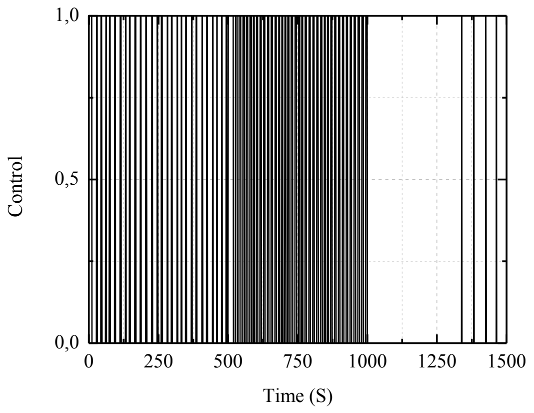


Fig.30: Digital control signal with ON/OFF control method.

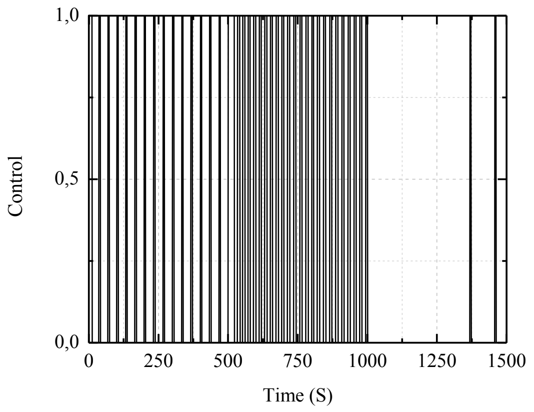


Fig.31: Digital control signal with hysteresis control method.

Table 3: Performance comparison between PID and ON/OFF controllers in the case of mBlock

	Rise Time (S)	**Overshoot (%)**	**IAE**
on/off control	8.5	2.96	152.74
PID control	9.1	0.1	37.63

The last tool tackled in this paper is the Matlab, developed by Mathworks. Matlab is an advanced programming environment and a high level interpreted language. In this study, Matlab support package for Arduino hardware is used to perform the following tasks: acquire analog sensor data from Arduino board, generate digital or analog (PWM form) output for a temperature-controlled heating element, and plot live data collected from analog sensor connected to Arduino board. Fig.32 represents multistep responses obtained with on/off, hysteresis and PID controllers. The main advantage of logic controllers is fast dynamic responses, but they are usually associated with oscillations around set points. Therefore, as can be seen from Fig.33, the PID controller can achieve very good tracking performance with small steady-state tracking errors (Table 4). Fig.34 represents the duty cycle ratio of the PWM signal which is generated by PID controller. Furthermore, with Matlab support package for Arduino hardware, it is only possible to see results from input/output instructions immediately, without having to compile.

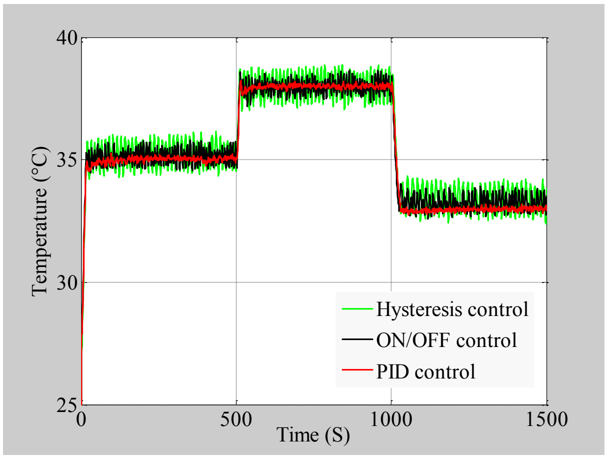


Fig.32: Comparison result at different temperature set points.

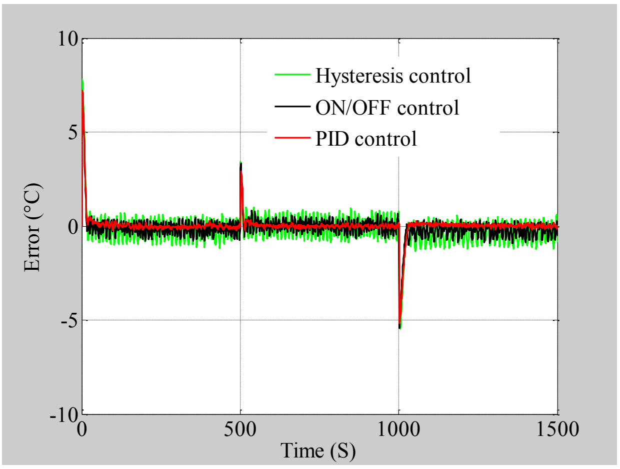


Fig.33: Curves of the tracking errors.

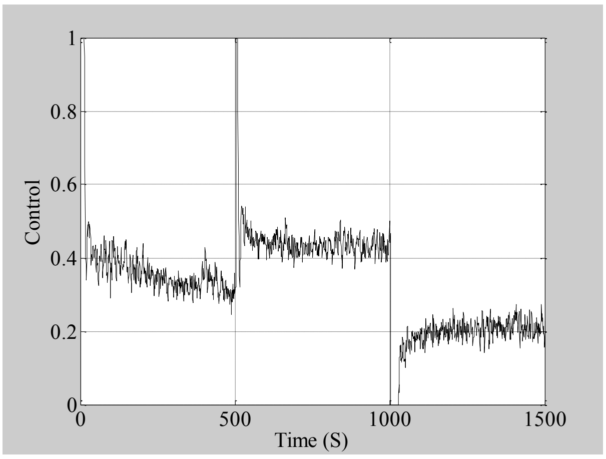


Fig.34: Duty cycle of the PWM signal with PID control method.

Table 4: Performance comparison between PID and ON/OFF controllers

	Rise Time (S)	**Overshoot (%)**	**IAE**
on/off control	13.4	0.8	191.47
PID control	13.8	0.21	110.91

6. Conclusion

In this study a simple thermostat system based on Arduino Uno board has been described. A serial monitor has been used to display the measured temperature value, the set point, the tracking error, and the IAE index. Also, LCD display has been used in order to display the desired and the actual temperature. Four coding environments have been described and used to implement the conventional control techniques and validate the proposed system. PID controller has resulted a good performance with very less percentage of overshoot than other two control techniques.
In comparison with other popular similar coding tools, Arduino IDE offers advantages of ease of coding, code optimization, has a lot of peripheral support (including many third-party libraries), and operates on any operating system. Graphical programming is superior to the conventional programming software in error free results, easy to understand, fast and user-friendliness. Hence, novice designers can perform simple practical experiments anytime and anywhere. In addition, the implementation of conventional control techniques offers to beginners a platform for learning the main concepts of control system and embedded system.

The proposed system is an inexpensive alternative oriented to control education. The future work will focus on the implementation of fuzzy control with a graphical programming tool.

References:

[1] K.Nouman, Z. Asim, K.Qasim, Comprehensive Study on Performance of PID Controller and its Applications, *2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Xi'an, China, 2018, pp.1574-1579.

[2] N. Cameron, *Arduino Applied: Comprehensive Projects for Everyday Electronics*, Apress, 2019.

[3] R. Winters, Arduino PID Temperature Control, *Nuts & Volts*, November 2017, pp. 30-35.

[4] S. Fattahzadeh, *Implementing of an Arduino based Temperature controller with PID algorithm*, Kindle Edition, 2015.

[5] S. Monk, *Make: Action: Movement, Light, and Sound with Arduino and Raspberry PI*, Make Community, 2016.

[6] R. Lacoste, PID Control without Math, *Circuit Cellar*, No. 221, 2008,pp. 54-61.

[7] R. Anderson, D. Cervo, *Pro Arduino*, Apress, 2013.

[8] Ramon Vilanova, Antonio Visioli, *PID control in the Third Millennium*, Springer, 2012.

[9] V. Arun Thomas, K. Srinivasan, Design and implementation of enhanced PID controller in embedded platform for realtime applications, *2nd International Conference on Power and Embedded Drive Control (ICPEDC)*, Chennai, India, August 2019, pp.129-133.

[10] Matrix TSL, C Code Icon Properties – Flowcode Help, https://www.matrixtsl.com/wiki/

[11] P. Reguera, D. García, M. Domínguez, M. A. Prada, S. Alonso, A Low-Cost Open Source Hardware in Control Education Case Study:Arduino-Feedback MS-150, *IFAC-PapersOnLine*, Vol. 48, No. 29, 2015, pp. 117-122.

[12] Y. Wang, Z. Chi, System of Wireless Temperature and Humidity Monitoring Based on Arduino Uno platform, *Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)*, Harbin, China, 2016, pp.770-773.

[13] P. M. Oliveira, J. D. Hedengren, An APMonitor Temperature Lab PID Control Experiment for Undergraduate Students, *International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain, 2019, pp. 790-797.

[14] J. H. Li, Control System Laboratory with Arduino, *International Symposium on Computer, Consumer and Control (IS3C)*, Taichung, Taiwan, 2018, pp. 181-184.

[15] X. Wu, X. Wang & G. He, A Fuzzy Self-tuning Temperature PID Control Algorithms for 3D Bio-printing Temperature Control System, *Chinese Control And Decision Conference (CCDC)*, Hefei, China, 2020, pp. 2846-2849.

[16] Y. He, H. Wang, D. Chen, The application of fuzzy PID control in the process temperature and humidity control of cigarette factory, *9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, Chongqing, China, 2020, pp.1091-1096.

[17] D. Purcaru, A. Purcaru, & V. Radulescu, Temperature measurement and control system for engineering education, *18th International Carpathian Control Conference (ICCC)*, Sinaia, Romania, 2017, pp.258-262.

[18] A. Najmurrokhman, Kusnandar, U. Komarudin, A. Daelami, F. Adiputra, Design and Implementation of Temperature and Humidity Control System in Oyster Mushroom Cultivation using Fuzzy Logic Controller, *International Conference on Computer, Control, Informatics and its Applications (IC3INA)*,Tangerang, Indonesia, 2019, pp. 146-150.

[19] N. A. M. Yunus, M. A. Zainudin, N. Sulaiman, & Z. A. Abbas, Microfluidic Fluid Flow Design with Arduino Relay and Temperature Controller for Processor, *5th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, Songkhla, Thailand, 2018, pp.28-30.