A Framework for Digital PID Controller for Undergraduate Control Systems Laboratory Usage

Pranabendra Prasad Chandra
Department of Electrical Engineering
Jadavpur University
Kolkata, India
pranabendrachandra@gmail.com

***Abstract*—This paper presents an inexpensive and stand-alone framework for digital PID controller for laboratory usage, that is student friendly as well as intuitive to learn and operate. The framework comprises of three hardware modules – input signal conditioning, microcontroller unit and output signal conditioning, supporting both open loop and closed-loop configurations. The use of this framework is threefold. First, students will be able to appreciate the amalgamation of hardware and software in the applications of digital controllers. Secondly, students can customise the different modules and experiment on their own for better understanding. Lastly, but most importantly, this framework can be used as an experiment kit in undergraduate control system laboratories. This setup will encourage students to have a hands-on approach to understand and tune a digital PID controller.**

***Keywords*—PID controller, digital control, Arduino, laboratory experiment, user interface**

I. Introduction

The implementation of this framework was motivated by the fact that, most engineering institutions in India, having a decent undergraduate control systems laboratory lack the experiment exclusively for digital PID control [1]. It is also well-known, that digital controllers have several advantages over analog schemes [2], [3]. The topic of digital PID control is widely covered in undergraduate courses like Process Control and Instrumentation and Digital Control [1], [4], yet in undergraduate control systems laboratory, experiments involving PID control, are only limited to its analog counterpart, i.e., using the PID trainer kits [5], [6]. To teach the applications of digital control, institutions rely exclusively on simulation-based experiments [7], typically done using Simulink®and MATLAB®. Simulations are unparalleled in modern educational process [8], but experiencing a real-time platform in undergraduate, motivates a student more [9].

Recent advancements to include experiments involving digital control is praiseworthy [10]–[12]. However, these experiments need a personal computer (PC) to carry out the full-fledged experiment. This paper presents a standalone framework that will enable the student to tune the controller parameters using an interactive user interface (UI), thereby eliminating the need of a PC during the experiment. Arduino [13] has been used in this framework for both the microcontroller board and the software, which are both open source. Arduino is economical, accessible and platform-agnostic [14]. It can be programmed in Arduino IDE [15], [16], which is supported in open source operating systems (OS), including Raspberry Pi OS [17]. Platforms like Arduino and Raspberry Pi have been showing enormous potential to be used in control system experiments [8], [18], [19]. Given the influence [20] of Arduino and the rapid development of its applications [21], students will feel encouraged to experiment and dig deep into the practical applications of digital control, outside the laboratory.

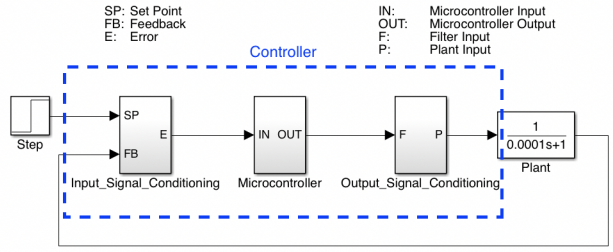


Fig. 1: Simulink®model representing the block diagram of the proposed controller

II. Methodology

A PID controller calculates an error value as the difference between a desired setpoint and a measured process variable. It then applies a correction based on the proportional, integral, and derivative terms. Hence, it enables automatic control. In continuous time domain, the PID equation [22] is given as,

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \qquad (1)$$

where, $e(t)$ is the error signal input to the controller and $u(t)$ is the controller output.

978-1-6654-4175-9/21/$31.00 ©2021 IEEE

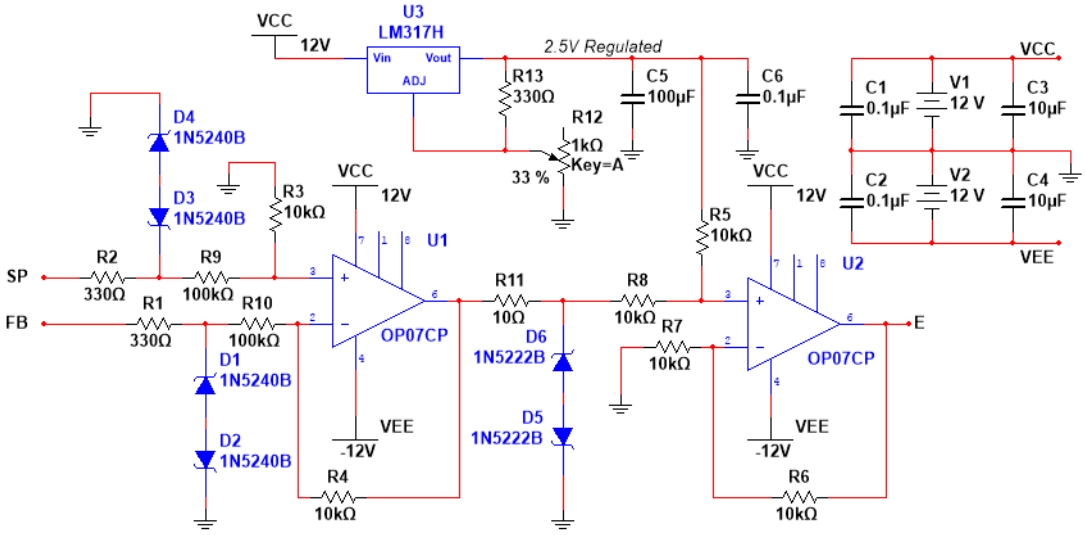


Fig. 2: Circuit realization of the input signal conditioning step

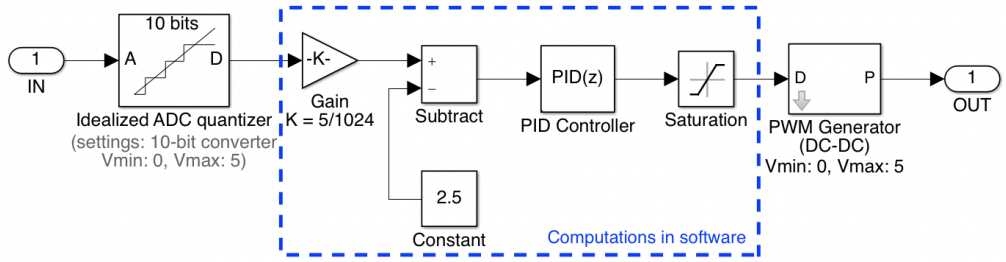


Fig. 3: Simulink®model of the microcontroller operation

In Control Systems Toolbox of MATLAB®, the transfer function of the system governed by Eqn. 1 in discrete-time domain is given in the PID block [23] as,

$$\frac{U(z)}{E(z)} = K_p + K_i T_s \frac{1}{z-1} + K_d \frac{N}{1 + NT_s \frac{1}{z-1}} \tag{2}$$

where, K_p, K_i and K_d are proportional, integral and differential gains respectively. T_s is sampling period. N is derivative filter divisor.

Converting this transfer function into difference equation, we obtain

$$u_n = u_n{}^{(1)} + u_n{}^{(2)} + u_n{}^{(3)} \tag{3}$$

where,

$$u_n{}^{(1)} = K_p e_n \tag{4a}$$
$$u_n{}^{(2)} = u_{n-1}{}^{(2)} + K_i T_s e_{n-1} \tag{4b}$$
$$u_n{}^{(3)} = u_{n-1}{}^{(3)}(1 - NT_s) + K_d N(e_n - e_{n-1}) \tag{4c}$$

III. Implementation

The controller comprises of three modules –

- Input Signal Conditioning - The setpoint (SP) and feedback (FB) signals can be bipolar. This module creates a modified error signal (E), which is unipolar and compatible for a 5V tolerant microcontroller.
- Microcontroller - The microcontroller digitizes the analog error signal and performs the digital PID computations on it, depending on the parameter values set by the user. This module also features an user interface (UI) which makes the system interactive and user-friendly.
- Output Signal Conditioning - The unipolar 5V pulse-width modulated (PWM) signal generated by the microcontroller (F) is converted to bipolar demodulated signal (P), which is fed to the plant.

A. Controller Specifications

- Supply voltages: +12V and -12V (obtained from two SMPS).

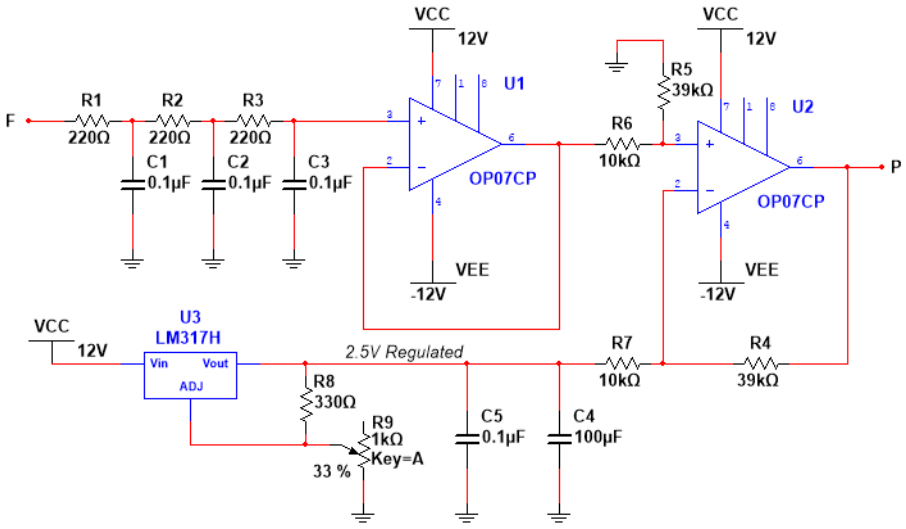


Fig. 4: Circuit realization of the output signal conditioning step

- Voltage of input signal and feedback should be within +10V to -10V.
- Sampling frequency = 5000 Hz.
- 5V tolerant microcontroller
- PWM switching frequency = 100 kHz.

B. Input Signal Conditioning

As mentioned in the controller specifications,

$$-10V \leq V_{SP}, V_{FB} \leq 10V \tag{5a}$$

$$\implies -20V \leq (V_{SP} - V_{FB}) \leq 20V \tag{5b}$$

The modified error,

$$E = \frac{V_{SP} - V_{FB}}{10} + 2.5V \tag{6a}$$

$$\therefore \quad 0.5V \leq E \leq 4.5V \tag{6b}$$

This bound of E ensures that the ADC of the microcontroller is not damaged, as it can only take positive voltages less than or equal to 5V [24].

In Fig. 2, U1 and U2 are precision op-amps, having very low offset voltage, in the order of microvolts [25]. U1 acts as a differential amplifier having gain 0.1, to attenuate $(V_{SP} - V_{FB})$ by 10 times. Zener diodes D1 and D2 are used in back-to-back mode for overvoltage protection. Their breakdown voltage is 10V. This ensures that the signal passing through to R10 remains within +10V and -10V. Else, the Zener diode pair clips the voltage [26]. R1 is used as a current limiting resistance, to protect D1 and D2 from overcurrent. Similar is the case for D3, D4, R2 and R9. D5 and D6 have breakdown voltage of 2.5V. They function similarly, to keep the signal within the limits of -2.5V to +2.5V. U3 is an adjustable positive voltage regulator. R12 is adjusted finely, to set the output regulated voltage to 2.5V. U2 acts as a non-inverting adder, with gain 1. It adds an offset of 2.5V to the bipolar signal, to make it a unipolar signal (lying between 0V to 5V). All the capacitors used in the above circuit (Fig. 2), are decoupling capacitors used with the voltage supplies.

C. Microcontroller Operation

The microcontroller used in the project is Arduino MEGA 2560 [27]. It is a 5V tolerant, having 10-bit ADC and PWM output. As shown in Fig. 3, the modified error signal from the previous module, is converted to a digital signal. The input voltages between 0 and 5V are mapped to integer values between 0 and 1023.

The microcontroller has been programmed in Arduino Programming Language, which is based on C++ [16], to perform the computations needed henceforth. The software stage converts the integer values, back to the voltage value, by multiplying a gain of 5/1024. A constant of 2.5V is subtracted from it, to virtually eliminate the offset in the signal, making it bipolar. Hence, the voltage value lies between -2.5V and +2.5V. This voltage value, which is discrete in time, is fed into the PID difference equations (Eqn. 4a, 4b, 4c) as e_n. Finally, u_n is computed (Eqn. 3), which controls the duty cycle **D** of the PWM generator. Therefore, tuning the PID parameters K_p, K_i, K_d directly impacts the duty cycle. As a protective measure, the computed duty cycle is limited between 0 and 1.

$$D = \min(1, \max(0, D)) \tag{7}$$

D. Output Signal Conditioning

This module reverses the transformations done by the input signal conditioning module, to make it compatible with the input signal and the feedback signal. In Fig. 4, the PWM output from the microcontroller is passed through a three-stage cascaded RC low pass filter (cut-off frequency of approximately 1.5kHz), to obtain the demodulated controller output. U1 acts as a buffer to avoid loading effect. U2 as a differential amplifier, with gain 3.9 (approx. 4). The demodulated controller output is cleared of any offset, by subtracting 2.5V from it. Therefore, it converts the unipolar signal, to a bipolar signal. U3 is an adjustable positive voltage regulator. R9 is adjusted finely, to set the output regulated voltage to 2.5V. Capacitors C4 and C5 are used as decoupling capacitors along with the voltage supply.

E. Modes of Operation

The controller has two independent modes of operation – SET and RUN mode. In SET mode, the user can set the PID parameters K_p, K_i and K_d. Each of the three parameters can range from 0.00001 to 99999.99999 and a cursor is present to set a particular integer at a particular location. When the controller is powered, it is initialized to SET mode and all the PID parameters are initialized to zero. Even in SET mode, the controller does all the computations and generates corresponding output, albeit with the parameter values that is already initialized. When the transition is made from SET mode to RUN mode, only then the new values of the PID parameters are updated in the program.

F. User Interface

To facilitate the setting of PID parameters in an interactive and user-friendly manner, an LCD screen displays the parameter values and the mode of operation (Fig. 6). An arrow symbol '–>' and a cursor are used in SET mode to guide the user to set the PID parameters. The cursor blinks at 1Hz frequency and 0.5 duty cycle. The SELECT button helps to alternate among K_p, K_i and K_d. The cursor can be moved horizontally for the selected parameter, using LEFT and RIGHT buttons. The digit at the cursor position can be increased or decreased, using UP and DOWN buttons (for eg., 9 to 0, or 0 to 1). All the parameters in the LCD can be cleared to zero, using the CLEAR button. It has no impact on the output as it functions only in SET mode. The RESET button on the other hand, resets the microcontroller and sets all the parameter values to zero in the program as well as the LCD.

As shown in Fig. 5, the user can interact with the controller, using a push button array. The individual push buttons perform their respective functions as listed in Table I.

The LCD screen is the manifestation of a 20x4 LCD module with an I2C module. The I2C module reduces the number of I/O pins required to connect to the microcontroller [28]. Each

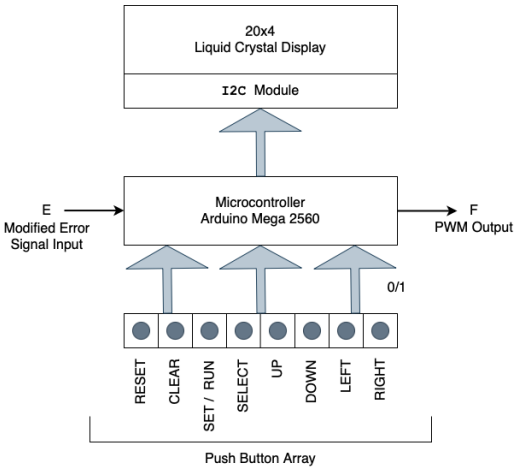


Fig. 5: Block diagram of the user interface comprising of an LCD screen and a push button array

(a) LCD screen during the low state of blinking cursor

(b) LCD screen during the high state of blinking cursor

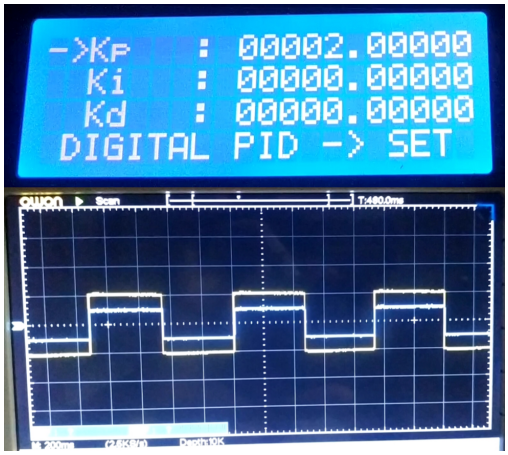
(c) LCD screen during RUN mode

Fig. 6: The different states of the LCD screen when the controller is operational

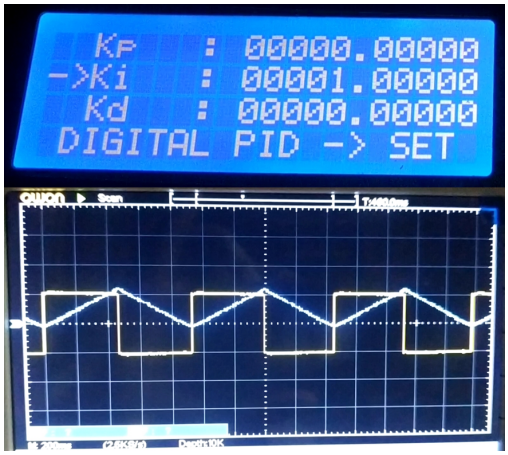
push button is associated with a pull-down resistor. So, when the push button is pressed, the microcontroller reads HIGH. Otherwise, it reads LOW.

G. Software Routines

- Reading the states of the push buttons
  Whenever the program enters the entry point, the states of the eight push buttons are read and stored. If any button is



(a) P controller: K_p = 2



(b) I controller: K_i = 1

(c) D controller: K_p = 0.01

Fig. 7: Controller output in open-loop configuration. The yellow waveform is the input signal and the blue waveform is the output signal.

TABLE I: Functions of individual push buttons in the user interface

Button	Function	SET Mode	RUN Mode
RIGHT	Moves the cursor right	✓	
LEFT	Moves the cursor left	✓	
UP	Increments the current digit MOD 10	✓	
DOWN	Decrements the current digit MOD 10	✓	
SELECT	Selects parameter (K_p, K_i , K_d) to be set	✓	
SET/RUN	To change parameter value and update to run	✓	✓
CLEAR	Clears the parameter value to all zeros	✓	
RESET	Resets the microcontroller	✓	✓

pressed, the microcontroller will read HIGH, else it will read LOW.

- Latching of button press
  Whenever a button is pressed, it will be latched, but no HIGH state will be registered. During that time, if any other button is pressed, they will not be active to register their states. Only when the initial button is released, the falling edge is detected and HIGH is registered for that particular button only.
- Detecting which button is pressed
  From the previous state and the present state of push buttons, the button that is pressed, is detected. For that particular button, the corresponding function is executed as a result of the button press.
- Display LCD in SET mode
  When the controller is in SET mode, the parameter pointer '−>' blinks and the cursor blinks for the user to change or update the value.
- Display LCD in RUN mode
  When the controller is in RUN mode, the LCD becomes static and shows the current value of the parameters.
- Reading the error value
  The modified error signal is read and it is multiplied by a gain of 5/1024 to bring it back to the voltage level compatible with the microcontroller. Then the offset is removed to get the attenuated error value.
- Computing the controller output
  The previous controller output, the present sample and the previous sample of the error is passed through a set of difference equations (Eqn. 4a, 4b, 4c) to obtain the current controller output (Eqn. 3).
- Changing the duty cycle of PWM
  The current controller output value determines the duty cycle of the PWM, which is always running at constant switching frequency.

IV. Experimental Results and Discussions

Fig. 7 shows the open loop characteristics of the controller. The setpoint signal is a bipolar square wave of time period 920ms and 10V pk-pk. Fig. 8 shows the closed loop char-

Fig. 8: Controller output in closed-loop configuration

acteristics of the controller, with the plant having a transfer function,

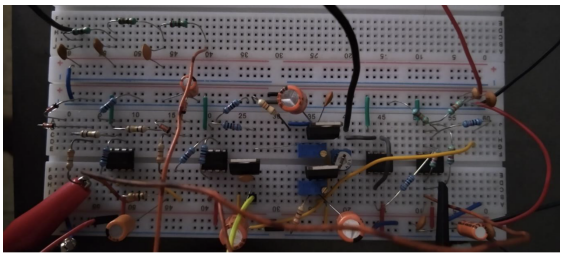
$$G(s) = \frac{1}{0.0001s + 1} \tag{8}$$

In both Fig. 7 and Fig. 8, the LCD screens shown are in SET mode, whereas the corresponding waveforms are seen when transitioned from SET mode to RUN mode.

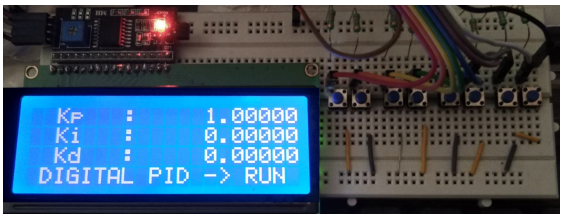
Due to the use of passive RC filter to demodulate the controller output from PWM, the controller performs poorly if the set point signal frequency becomes comparable to the cut off frequency of the filter. Also, the system time constant increases slightly. It has also been seen, that the digital PID controller works best for low frequencies of the setpoint signal ($\leq 100Hz$). The output starts to exhibit distortions, as the frequency of the setpoint signal is gradually increased ($> 100Hz$). These effects can be largely eliminated, if a microcontroller with an inbuilt DAC is used.

V. Conclusion

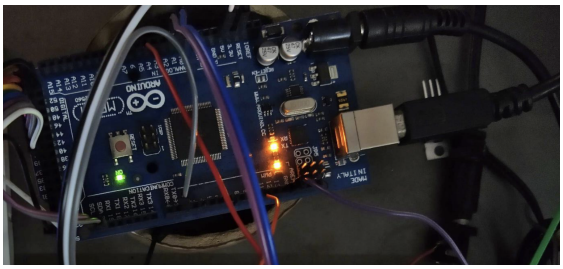
1) This paper presents a cost-effective, student-friendly framework for digital PID controller that can be adopted by engineering institutions as a standard experiment in their undergraduate control systems laboratory course-work. The novelty of the paper lies in the fact that it offers a complete hardware set up of the digital PID controller with off-the-shelf components, where the experimental model is working well at par with commercial systems. It is even inexpensive compared to the existing digital control system trainer kits in the market [29].
2) The framework presented can be encapsulated as a training kit, for laboratory usage as shown in Fig. 10. This setup when used in tandem with a signal generator and an oscilloscope, will serve as a full-fledged laboratory experiment system. Students can vary the input signal (sine, square or triangle wave) by changing its amplitude and frequency, change the PID parameters via the user interface and see the open loop and closed loop characteristics. They can also see the same by experimenting with different plant transfer functions.
3) Students can also experiment using this framework outside the laboratory. They can build the setup from



(a) Signal conditioning circuits



(b) User Interface - LCD screen and push button array



(c) Microcontroller board (Arduino MEGA 2560)

Fig. 9: Photographs of the hardware setup

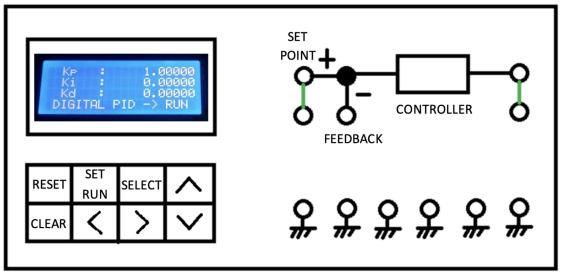


Fig. 10: Artist's impression of the front view of the trainer kit encapsulating the proposed controller for laboratory usage

scratch. They can simply connect an astable multivibrator as input and see the signals in the PC using the Arduino software. This way, they won't need a laboratory-grade signal generator or an oscilloscope.

4) This framework can be extended and improvements can be made on top of it. For eg., a microcontroller with an inbuilt DAC can be used to largely eliminate the effects of the time constant introduced by the cascaded passive RC filters. Also, the two SMPS that has been used as a power source in this work, can be done with a single power source with positive voltage [30].
5) On using this inexpensive, stand-alone, student-friendly and easy-to-use framework, students will be able to appreciate the amalgamation of hardware and software in the applications of digital controllers.

ACKNOWLEDGEMENT

I am immensely grateful to Prof. Tapan Kumar Ghoshal, Prof. Samar Bhattacharya and Prof. Smita Sadhu of Department of Electrical Engineering, Jadavpur University for their humble mentoring on this work, as part of my final year seminar project. Any error is mine and should not tarnish the reputations of these esteemed persons. I want to express my sincere gratitude to my parents for their constant encouragement, care, and extending all kinds of support throughout this work.

REFERENCES

[1] J. University, "Electrical engineering - syllabus." http://www.jaduniv.edu.in/upload_files/course_file/1408616005-1.pdf, Accessed: 2021-07-12.
[2] A. Majumdar, A. Das, R. Alagirusamy, and V. Kothari, *Process control in textile manufacturing*. Elsevier, 2012.
[3] P. Murphy, M. Xie, Y. Li, M. Ferdowsi, N. Patel, F. Fatehi, A. Homaifar, and F. Lee, "Study of digital vs analog control," in *Power Electronics Seminar Proceedings (CPES Center for Power Electronics Systems)*, pp. 203–206, Citeseer, 2002.
[4] G. T. University, "Digital control system." https://www.gtu.ac.in/syllabus/NEW\%20BE/Sem6/2160807.pdf, Accessed: 2021-07-12.
[5] T. R. Babu, "Control systems laboratory manual." http://mist.ac.in/pdfs/LabManuals/EEE/ControlSytemLab.pdf, Accessed: 2021-07-12.
[6] I. of Aeronautical Engineering, "Control systems and simulation lab." https://www.iare.ac.in/sites/default/files/lab1/IARE_Control_Systems_Lab_Manual.pdf, Accessed: 2021-07-12.
[7] J. C. of Engineering, "Laboratory instruction manual, control system 2 lab." https://www.jiscollege.ac.in/ee/pdf/ee-693-lab\%20manual-final-control-system-2.pdf, Accessed: 2021-07-12.
[8] J. Sobota, R. PiŠl, P. Balda, and M. Schlegel, "Raspberry pi and arduino boards in control education," *IFAC Proceedings Volumes*, vol. 46, no. 17, pp. 7–12, 2013.
[9] J. Bertram, J. Moskaliuk, and U. Cress, "Virtual training: Making reality work?," *Computers in Human Behavior*, vol. 43, pp. 284–292, 2015.
[10] M. Gunasekaran and R. Potluri, "Low-cost undergraduate control systems experiments using microcontroller-based control of a dc motor," *IEEE Transactions on Education*, vol. 55, no. 4, pp. 508–516, 2012.
[11] M. Gunasekaran and R. Potluri, "Lab manual for ee380 (control lab)," 2013.
[12] S. Dalapati, "A control systems laboratory experiment on transfer function emulation," in *2020 IEEE Applied Signal Processing Conference (ASPCON)*, pp. 51–55, IEEE, 2020.
[13] Arduino. https://www.arduino.cc/, Accessed: 2021-07-12.
[14] A. A. Galadima, "Arduino as a learning tool," in *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, pp. 1–4, IEEE, 2014.
[15] M. McRoberts, *Beginning Arduino*. Apress, 2011.
[16] S. Monk, *Programming Arduino: getting started with sketches*. McGraw-Hill Education, 2016.
[17] A. D. Deshmukh and U. B. Shinde, "A low cost environment monitoring system using raspberry pi and arduino with zigbee," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 3, pp. 1–6, IEEE, 2016.
[18] R. M. Reck and R. Sreenivas, "Developing an affordable and portable control systems laboratory kit with a raspberry pi," *Electronics*, vol. 5, no. 3, p. 36, 2016.
[19] A. Vostrukhin, E. Vakhtina, *et al.*, "Studying digital signal processing on arduino based platform," in *Proceedings of the 15th International Scientific Conference on Engineering for Rural Development, Jelgava, Latvia*, pp. 25–27, 2016.
[20] J. C. Martinez-Santos, O. Acevedo-Patino, and S. H. Contreras-Ortiz, "Influence of arduino on the development of advanced microcontrollers courses," *IEEE revista iberoamericana de tecnologias del aprendizaje*, vol. 12, no. 4, pp. 208–217, 2017.
[21] M. Matijevic and V. Cvjetkovic, "Overview of architectures with arduino boards as building blocks for data acquisition and control systems," in *2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, pp. 56–63, IEEE, 2016.
[22] K. Ogata, *Discrete-time control systems*. Prentice-Hall, Inc., 1995.
[23] M. Control Systems Toolbox, "Discrete-time proportional-integral-derivative (pid) controllers." https://www.mathworks.com/help/control/ug/discrete-time-proportional-integral-derivative-pid-controller.html.
[24] G. Makan, R. Mingesz, and Z. Gingl, "How accurate is an arduino ohmmeter?," *Physics Education*, vol. 54, no. 3, p. 033001, 2019.
[25] I. Analog Devices, "Ultralow offset voltage operational amplifiers op07," 2002.
[26] R. L. Boylestad and L. Nashelsky, *Electronic devices and circuit theory*. Prentice Hall, 2012.
[27] A. Kurniawan, *Arduino Mega 2560 A Hands-On Guide for Beginner*. PE press, 2012.
[28] O. Akinwole and T. Oladimeji, "Design and implementation of arduino microcontroller based automatic lighting control with i2c lcd display," *J. Electr Electron Syst*, vol. 7, p. 258, 2018.
[29] A. E. Industries, "Digital control system trainer (4039)." https://pdf.indiamart.com/impdf/9840582191/631427/digital-control-system-model.pdf, Accessed: 2021-10-19.
[30] T. Instruments, "Dual output rail power supply with inverting buck boost converter reference design." https://www.ti.com/tool/PMP10090, Accessed: 2021-07-12.