Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 24, S. 235-239, Nisan 2021

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 24, pp. 235-239, April 2021

Research Article

Design and Realization of Online Auto Tuning PID Controller Based on Cohen-Coon Method

Ali Egemen Taşören[1*]

[1*] Hacettepe University, Faculty of Engineering, Department of Computer Engineering, Ankara, Turkey, (ORCID: 0000-0001-8711-2010), taliegemen@gmail.com

(2nd International Conference on Access to Recent Advances in Engineering and Digitalization (ARACONF)-10–12 March 2021)

(**DOI:** 10.31590/ejosat.897727)

ATIF/REFERENCE: Taşören, A. E. (2021). Design and Realization of Online Auto Tuning PID Controller Based on Cohen-Coon Method. *European Journal of Science and Technology*, (24), 235-239.

Abstract

In this paper, a controller which is capable of automatically obtaining proportional integral derivative (PID) parameters using Cohen-Coon tuning method is developed and tested on a real system consisting of an L298N circuit and a 1.1W brushless DC motor. The main purpose of this paper is to propose a fast, portable, and model-independent automatic controller. ATmega2560 microcontroller is programmed as a PID controller and a Raspberry Pi programmed to collect, compute, send and receive data from ATmega2560 through serial communication. Finally, the system is proven for its performance with satisfactory results.

Keywords: PID, Parameter Tuning, Serial Communication.

Cohen-Coon Tabanlı Otomatik Ayarlayıcılı Online PID Denetleyici Tasarımı ve Gerçeklenmesi

Özet

Çalışmada, Cohen-Coon yöntemini kullanarak otomatik olarak Oransal-İntegral-Türevsel (PID) katsayılarını elde eden denetleyici tasarlanır, L298N devresi ve 1.1 W fırçasız doğru akım motoru içeren gerçek sistemde test edilir. Çalışmada hızlı, taşınabilir ve modelden bağımsız bir otomatik denetleyici önerilir. ATmega2560 mikrodenetleyici PID denetleyici olarak programlanır. ATmega2560 seri haberleşme aracılığıyla veri eldesi, işleme ve gönderimi için programlanmış Raspberry Pi ile haberleşir. Önerilen sistemin performansı, yapılan testler ile kanıtlanır.

Anahtar Kelimeler: PID, Parametre Ayarlama, Seri Haberleşme.

1. Introduction

Proportional-Integral-Derivative (PID) is a prominent linear controller, nearly 95% of industrial processes are controlled by them and most of them are implemented on microcontrollers (Åström & Hägglund, 1995). PID controllers are easy to set up for most systems and they can be used for several different control systems applications namely speed, position, or temperature control (Ogata, 2010). Although PID controllers have only three parameters, the proper approach has to be applied. Without using the appropriate PID parameters, desired control conditions cannot be obtained (Skogestad, 2003). To obtain appropriate PID parameters, Ziegler-Nichols (Ziegler & Nichols, 1942) and Cohen-Coon (Cohen & Coon, 1953) tuning methods are generally used. However, there are several types of PID tuning methods in literature (Sheel & Gupta, 2012). A study on comparing conventional PID tuning methods is recently published and Cohen-Coon method is found to be the best performing among compared methods (Taşören, Örenbaş, & Şahin, 2018).

Tuning a PID controller manually takes the time and effort of the system operator. To minimize human interaction with PID controllers, Automatic Tuning Methods have been developed (Astrom & Hagglund, 2006). Using these methods with MATLAB, a model dependent Adaptive Automatically Tuning PID Control for Vertical Take-off and Landing module has been simulated (Taşören, Gökçen, Şahin & Soydemir, 2020). Also, a comparison has been made for various Auto-Tuning methods are proposed (Hang & Sin, 1993).

The main aim of this paper is to design and realize an open-source model independent online automatically tuning PID controller based on Cohen-Coon tuning method to obtain feasible PID parameters. The design also will be portable, versatile and open sourced [https://github.com/taliegemen/PID-Autotuner].

In this paper, designed hardware and software which are used to automatically tune a BLDC system using the Cohen-Coon method has online properties, which aims to quickly tune a plant with close to no human interaction. A brief explanation of the hardware and software of the system that's used to create Automatic Tuner is, information about finding proper tuning parameters for Cohen-Coon without using any transfer function, and the algorithm of the system, and performance information about the Automatic Tuner system is given in Section 2. The third section includes discussions about the obtained results meanwhile the fourth section concludes the paper also offers topics for future research.

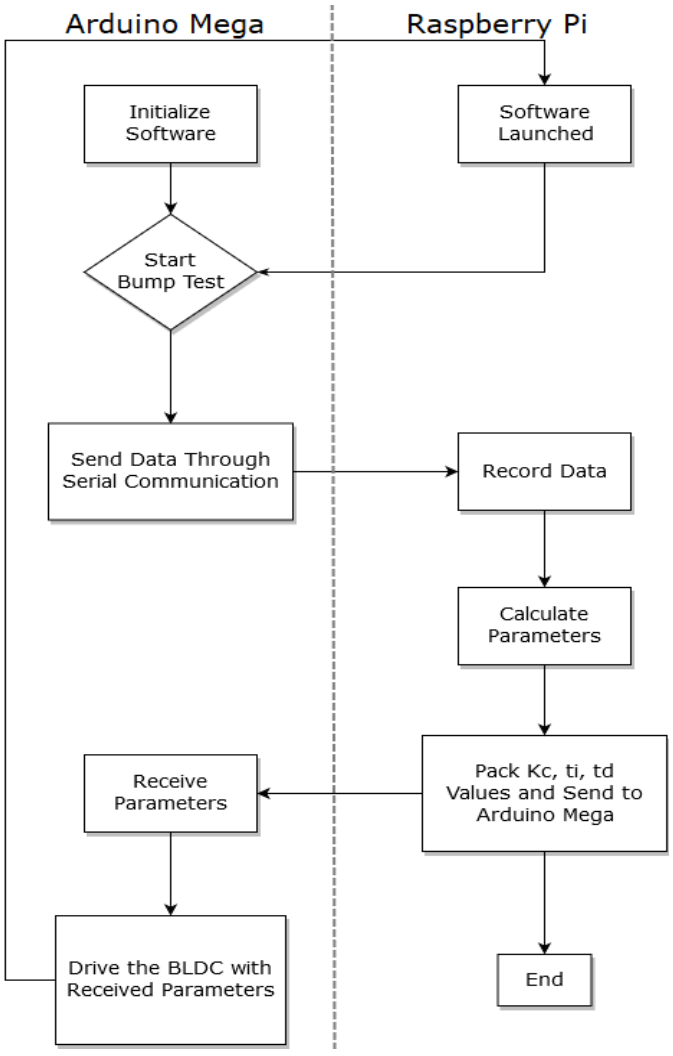
2. Description of Setup

2. 1. Hardware Design of Plant

The designed test plant consists of a Raspberry Pi 3B+ (A computer which has a 1.4 GHz quad-core processor and 1 GB LPDDR2 SDRAM) with Ubuntu Mate 16.04 operating system, an ATmega2560 based microcontroller unit (Arduino Mega) which is programmed as a PID Controller, an L298N h-bridge transistor circuit and a Maxon 344515 BLDC Motor with Encoder and 1:16 reducer.

Since there can be an interruption from the operating system of Raspberry Pi, it cannot be used as a standalone PID Controller also, an Arduino Mega cannot be used for a standalone device for auto-tuning operation since it struggles with calculations of big arrays, thus it cannot drive the motor properly and calculate the parameters at the same time. For better performance, instead of the approaches mentioned above, Arduino Mega and Raspberry Pi are connected with a USB cable which allows serial communication between devices. This configuration will allow Raspberry Pi to handle array operations and Arduino will be responsible for controlling the BLDC with calculated gain parameters which are found after the bump test.

Figure 1. Software Flowchart



There is data that cannot be able to reach from Arduino Mega to Raspberry Pi because of operating system interruption but since this only happens once per 7-8 seconds and it takes few microseconds, there is no loss of big data in the process. Also, the hardware of Arduino Mega only can run the software uploaded for this process only once per 1-2 milliseconds, which sets our sampling rate.

Maxon Motor 344515 has 16000 rpm speed without any loads, and it runs at 1000 rpm with the reductor on the motor. It also has a 2-channel encoder which generates 770 pulses per one revolution of the motor. L298N H-Bridge Transistor circuit is used to control the revolution direction of the BLDC mentioned above.

2. 2. Software Design of Plant

A Raspberry Pi and an Arduino Mega are properly coded separately to enable serial communication between devices and PID Control of BLDC. For this setup, the Ideal PID algorithm has been used for Arduino Mega since the Ideal PID algorithm is the most widely used PID algorithm for industrial control purposes (Smuts, 2011). Principles of Ideal PID control algorithms are discussed in the previous study (Taşören, Örenbaş & Şahin, 2018). Software on Arduino Mega enables Arduino to send position and time data of the BLDC to Raspberry Pi using serial communication. The software on Raspberry Pi enables Raspberry Pi to do calculations on data gathered from Arduino Mega, and then send the PID parameters obtained from these calculations to Arduino Mega with serial communication. A basic scheme of the algorithm can be found in Figure 1.

Before further software information, the method for calculating proper PID Control parameters has to be discussed.

2. 2. 1. Obtaining PID Parameters

Finding suitable PID controller parameters is a worthy task for most systems. To find feasible PID tuning parameters some calculations have to be made, and these calculations have to be repeated when there are changes in the system. In this section, information about the system which automates this process will be given.

The ideal PID control algorithm scheme consists of 3 parameters. One of the parameters affects the other two directions. But these two parameters do not affect other operations. To be precise, the integral tuning parameter only affects the integral operation of the algorithm meanwhile derivative tuning parameter only affects the derivative operation of the algorithm, and the proportional gain parameter affects all of the operations in the ideal PID algorithm. Ideal PID control algorithm is also known as noninteractive, ISA, and standard PID control algorithm. There are many Tuning Methods proposed for this method namely, Lambda, integral of the absolute error, and Cohen-Coon Tuning Methods (Smuts, 2011). A mathematical expression of the ideal PID algorithm can be found in Equation 1.

$$K_c \left[e + \left(\frac{1}{T_i}\right) \int e dt + T_d \left(\frac{de}{dt}\right) \right] = C_o \qquad (1)$$

Where K_c is the proportional gain parameter, T_i is the integral tuning parameter, T_d is the derivative tuning parameter, e is error, t is time and C_o is controller output.

Arduino Mega is programmed in a fashion that the software on it will calculate the C_o continuously upon receiving input from serial communication as time passes using Equation 1. The first input mentioned in the third row of Algorithm 1 is the input that starts the Automatic Tuning operation.

At first, the proper parameters have to be calculated to tune a BLDC and this can be done by using a method called Bump Test (Smuts, 2011). Bump Test is rather useful when there are no specifications for the system which will be tuned. This method does not require any kind of specification knowledge of a system, rather it only measures the response of the system to the output of the controller without any amplification, so it can be used for practical PID tuning operations easily.

There is an initialization process on Arduino Mega which has to be launched at first. This initialization process settles K_c to 0 to ensure the software won't work without serial communication input and it starts the serial communication. The first input which is given by Raspberry Pi to Arduino Mega starts the Bump Test of the motor by setting PID parameters on Arduino Mega. K_c is taken as 1, t_d has to be 0 and t_i should be a high number to ensure the integral and derivative part of the PID controller does not interrupt with the Bump Test. Also, K_c is settled 1 since there is no need for gain in Bump Test.

After the Bump Test has been made, all the position and time data of the test are collected by Raspberry Pi and recorded into a Comma Separated Values (CSV) file. After that operation, the software on Raspberry Pi opens the CSV file and makes the calculations that are needed to find PID controller parameters. 3 specific values have to be found to calculate PID parameters. These values are τ, dead time (t_d), and Process Gain (G_p). To explain briefly, τ is the time that passes between the end of dead time and the moment of 0.63 change of motor position, the dead time is the time that needs to pass to the motor to rotate, and gain is the percentage change of system compared to the reference (Equation 2). To find these values, all of the measured position and time values from the Bump Test have been sent to Raspberry Pi via serial communication. Since Arduino Mega cannot handle both controlling the motor and making calculations at the same time, calculations are made by Raspberry Pi in this way. Raspberry Pi has calculated these values with using Equation 2 and Figure 2 as $G_p = 0.9$, $t_d = 14$ ms and $\tau = 47.35$.

$$\frac{\Delta P_{system}}{\Delta P_{reference}} = G_p \qquad (2)$$

Where ΔP_{system} is the position change in the system, $\Delta P_{reference}$ is the position change in the reference, G_p is the process gain.

After these calculations, Raspberry Pi obtains the PID parameters by using the Cohen-Coon Tuning Method formulas which can be seen in Table I (Smuts, 2011), and the obtained parameters can be found in Table II.

COHEN-COON TUNING FORMULAS

	K_c	T_i	T_d
PID	$\frac{1.35\tau}{G_p \times t_d} + \frac{1}{4G_p}$	$\frac{t_d(0.4625t_d + 2.5\tau)}{\tau + 0.611t_d}$	$\frac{0.37\tau t_d}{0.185t_d + \tau}$

EVALUATED PID PARAMETERS

PID	K_c	T_i	T_d
Cohen-Coon	5.35	31.27	4.91

2. 2. 2. Online PID Control

After the calculations, Raspberry Pi sends the found PID parameters to Arduino Mega via serial communication by packaging them into bytes, and Arduino Mega reads these parameters and starts to drive the BLDC with the parameters that have been provided by Raspberry Pi. If it is wanted, the algorithm can be started again just by relaunching the software on Raspberry Pi. The algorithm of Arduino Mega and Raspberry Pi can be found in Algorithm 1 and 2 respectively.

Algorithm 1: Software on Arduino Mega

start: the initial setup process
end: the initial setup process
if there is a signal from the serial communication
  set: K_c to 1, t_i to 999999, t_d to 0,
  reference position to 100
end
while
  read: encoder and timer data
  record: encoder data as present position data,
  record timer data as present time data
  calculate: the time difference by subtracting
  previous data from present data
  calculate: error by subtracting the reference from
  encoder position data and C_o
  If the C_o is positive,
  drive the BLDC motor by sending PWM
  to port number 9
  end
  If the C_o is negative,
  drive the BLDC motor by sending PWM
  to port number 10
  end
  record: the present position time as previous
  position data and present time data as the previous
  time data
  if there is incoming serial data
    read: the packaged data and record it as K_c, t_i, t_d
  end
break
end

Algorithm 2: Software on Raspberry Pi

start: serial communication with intended baud rate.
send: start command to Arduino Mega
If the position from serial communication is
not the same for the given time variable
(this time can be changed based on the system
that's being used)
  receive: position and time information
  from Arduino Mega
  write: position and time data to the output file
end
convert: data on the output file to 2 different arrays
find: dead time, tau, and Process Gain
calculate: K_c, t_i, t_d
prepare: convert the found PID parameters to bytes
since bytes can be sent via serial communication
send: PID parameters to Arduino Mega
end: stop the software

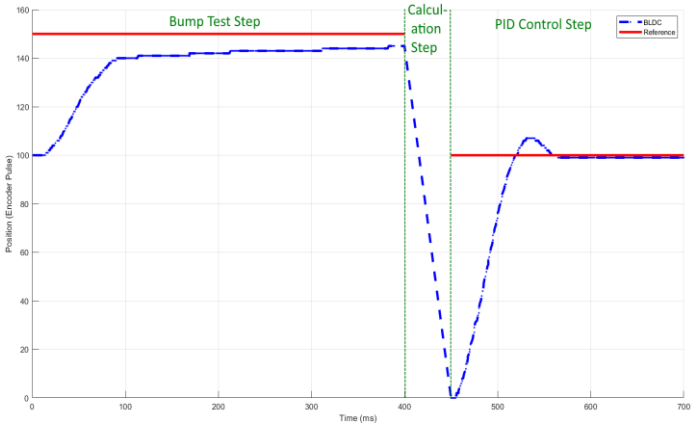
3. Results and Discussion

To measure the performance, a little change to the algorithm can be made to the software in Raspberry Pi to continue obtaining position and data values from serial communication. With this addition, the graph in Figure 2 has been presented from the data obtained by Raspberry Pi.

In Figure 2, there is a calculation step that takes 50 ms, and there is a Bump Test which is mentioned detailly at the PID parameter calculation part of the paper, after these steps BLDC is controlled with calculated PID parameters which are calculated with Cohen-Coon Tuning method.

With Automatic Tuner System, BLDC Plant has been controlled with %7 overshoot, 103 milliseconds of settling time, 46 milliseconds of rising time, and 1% steady-state error. This result is very satisfactory considering no calculations are made by humans for this PID tuning operation.

Figure 2. BLDC Response Graph



The motors which are used for various kind of systems have to be tuned again each time when their friction, load, or position changes since these changes affect motor transfer function as well. To apply automatic tuning, there is no need for the model of the motor which is being used with the system because proposed algorithm works independent from the model of the system.

There are lots of required calculations for the process of getting appropriate tuning parameters. Each of the changes that has been made to the system also requires new parameters, with Automatic Tuner Raspberry Pi will do all of the calculations thus the tuning process will be faster, and easier.

The mentioned Automatic Tuner system is very portable since it is made up of a small-sized computer (Raspberry Pi), an H-Bridge Transistor circuit (L298N), and an ATmega2560 based microcontroller (Arduino Mega) is used. With these components Automatic Tuner can be used in different places easily, the proposed controller is also easy to set up with motors that have 2-channel encoders. With a single Automatic Tuner, different systems can be tuned since there are no required information about the plant.

4. Conclusion

In this study, an Automatic Tuner System has been designed to Tune PID systems quickly, reducing the human effort that's put into calculations and observations, and with a design as portable as possible. This design is carried out by using an ATmega2560 and a Raspberry Pi. ATmega2560 is coded in a way that is feasible for noninteractive PID control, also the software on the microcontroller allows it to serially communicate with other devices. Raspberry Pi is programmed in a way that allows it to make serial communication with ATmega2560. Moreover, it records the serial communication data which is received and makes calculations on these values using Cohen-Coon parameters and delivers the found parameters to the ATmega2560, allowing ATmega2560 to drive the BLDC with appropriate parameters.

In future work, the automatic tuner system which is developed in this paper can be improved with machine learning to obtain optimal controller performance.

References

Åström, K. J., & Hägglund, T. (1995). PID controllers: theory, design, and tuning (Vol. 2). Research Triangle Park, NC: Instrument society of America.

Ogata, K. (2010). Modern control engineering. Prentice hall.

Skogestad, S. (2003). Simple analytic rules for model reduction and PID controller tuning. Journal of process control, 13(4), 291-309.

Ziegler, J. G., & Nichols, N. B. (1942). Optimum settings for automatic controllers. trans. ASME, 64(11).

Cohen, G., & Coon, G. A. (1953). Theoretical consideration of retarded control. Trans. Asme, 75, 827-834.

Sheel, S., & Gupta, O. (2012). New techniques of PID controller tuning of a DC motor—development of a toolbox. MIT International Journal of Electrical and Instrumentation Engineering, 2(2), 65-69.

Åström, K. J., & Hägglund, T. (2006). PID control. IEEE Control Systems Magazine, 1066(033X/06). Chicago

Hang, C. C., & Sin, K. K. (1991). A comparative performance study of PID auto-tuners. IEEE Control Systems Magazine, 11(5), 41-47.

Smuts, J. F. (2011). Process Control for Practitioners: How to Tune PID Controllers and Optimize Control Loops. OptiControls.

O'dwyer, A. (2009). Handbook of PI and PID controller tuning rules. World Scientific.

Taşören, A. E., Örenbaş, H., & Şahin, S. (2018, October). Analyze and comparison of different PID tuning methods on a brushless DC motor using ATmega328 based microcontroller unit. In 2018 6th International Conference on Control Engineering & Information Technology (CEIT) (pp. 1-4). IEEE.

Taşören, A. E., Gökçen, A., Soydemir, M. U., Şahin, S. (2020). Artificial Neural Network-Based Adaptive PID Controller Design for Vertical Takeoff and Landing Model. European Journal of Science and Technology, (Special Issue), 87-93.