Development of a low-cost PID setup for engineering technology students

Péter Zoltán Csurcsia[1,2], Pujan Bhandari[1], Tim De Troyer[1,2]

[1]Vrije Universiteit Brussel, Department of Engineering Technology
2Brussels Institute for Thermal-fluid systems and clean Energy (BRITE)
Pleinlaan 2, B-1050 Elsene, Belgium (e-mail: peter.zoltan.csurcsia@vub.ac.be)

Abstract: PID controllers are the most frequently used controllers. Theoretical understanding of the system to be controlled and the working principle of PID controllers are crucial for closed-loop control. This fundamental knowledge is - in principle - learned by all engineering students at the undergraduate level. However, understanding the practical aspects of the application of a PID controller and their connections to theoretical concepts must be combined in a clever way. This paper explores the process of the development of a low-cost, easy to manufacture demonstration setup which can act as the bridge between the theoretical and practical world of the control system knowledge. As a reaction to the COVID epidemy, this project is made with the philosophy that the students should be able to take these setups home or even replicate them themselves. The proposed setup comes with an interactive block diagram-based graphical user interface which allows the user to observe the physical changes of the process in -nearly- real-time. The aim of the provided exercises is to comprehend the similarities and the difference between the theoretical and practical aspects of closed-loop control. This paper explains the components of the setup, the development process, the recommended exercises, and feedback from our students.

Keywords: PID-controller, automation, closed-loop control, control theory

1. INTRODUCTION

Control theory is a basic skill that every engineer, regardless of their specialization, must possess. Electrical, mechanical, chemical, or aerospace engineers, all will need to use (control) system theory at some time to address a specific problem. Although the exact application may change from one field to another, the approach and method for solving it will nearly be identical.

The proposed work is a budget-friendly ball balancing closed-loop control demo setup (see Figure 1) that is aimed at undergraduate technical students. The objectives of this work are 1) the implementation of the proposed low-cost setup that mimics all the typical industry-related issues (like process noise, measurement noise), and 2) the reinforcement of the theoretical knowledge in an intuitive and engaging way.

Many open-source demonstration setups that employ the PID controller to achieve the desired controlling action are available. Such examples are, for instance, controlling the shaft rotation of a D.C. motor (Electric Diy Lab, 2018), temperature control of a heater (Arduino Team, 2018), balancing a ball on a beam (Mechatronics Tutorials, 2014) or on a plate (Link, 2018). Many of the demonstration setups are cost-effective (like Arduino) or high-speed complex (like Raspberry PI) microprocessor-based. However, the majority of those demonstration setups use predefined PID libraries – for instance, (Beauregard, 2018). These demonstration setups illustrate how easily a PID controller can be used with the help of the available library but does not explain how the PID controller can be designed in general. Thus, such demonstration setups do not allow to teach fundamental aspects of constructing a PID controller.

Further, some of the demonstration setups cut corners when implementing the controllers in discrete time. That may lead to a problem where the process variable is not adequately controlled. To do it properly, the designer must understand how the controller must be implemented in the software using a suitable discretization technique, z-transform (Aström & Hägglund, 1995). This applies to designing the controllers to control any closed-loop system.

The performance of a controller relies on the used components in the closed-loop system. For instance, a ball balancing demonstration setup can be very precise provided that the high speed and precision components are used. For instance, one needs a high-resolution camera with at least 60-80 FPS (frame per second), motors without a backlash to control the deflection of the beam, a ball with non-reflective coating, and a connecting rod with ball head and fork joints. The cost of a demonstration setup using such components would be comparatively high. Further, the main issue is that it will also be too perfect for our (teaching) application since, the goal of the intended demonstration setup is to mimic the typical problems of an industrial PID control due to measurement and process noise, disturbances, and other imperfections.

This paper discusses the approach where the focus lies on demonstrating the practical challenges which appear throughout the design and control process of a closed-loop system and coping with them using adequate engineering techniques. The content of this paper is organized as follows. In section 2, the components required to develop the physical demonstration setup are introduced and the practical aspects of the proposed PID controller are discussed. In section 3, a few suggested experiments are elaborated. Section 4 elaborates

some of our observations and feedback from our students. Finally, the paper is concluded in section 5.

2. THE DEVELOPMENT

2.1 The setup

The proposed demonstration setup (see Figure 1, Figure 2) is meant to be used by colleges, universities, and technical schools as an introductory practical session to control theory. Thus, there are a few limitations regarding the physical setup. The demonstration setup should be basic enough to be effortlessly replicated, it should also be cost-effective but on the other hand, it should be able to demonstrate all the challenges and choices encountered in the case of the design process of a real-life PID controller.

Figure 1: Ball balancing demonstration setup made up of a wooden base and stand.

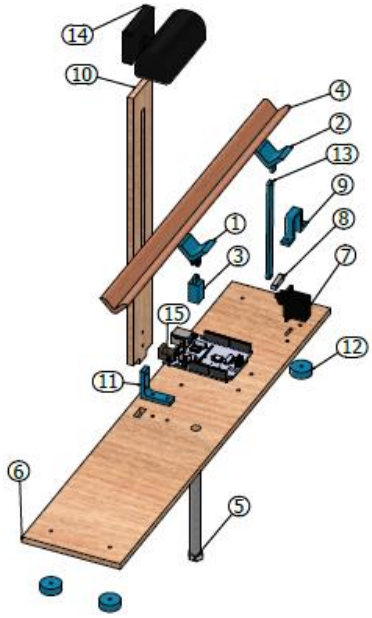

Figure 2: Exploded view of the demonstration setup.1-2: V-piece beam holder, 3: hinge piece, 4: V-shaped beam, 5: 8-mm bolt, 6: base plate, 7: servo motor, 8 servo arm, 9: servo motor holder, 10: camera stand, 11: L-Profile, 12: base leg foot, 13: pushrod, 14: camera, 15: Arduino Uno.

There are various possibilities for making a demonstration setup. Balancing the ball on a platform is a cost-effective and spectacular choice because of its simplicity. The components are enlisted in Figure 2. In the proposed configuration, a servo motor is used for deflecting the platform. The servo motor must be driven by a microcontroller. A budget-friendly Arduino (UNO) is a suitable microcontroller for this application. Furthermore, a sensor is required to determine the position of the ball on the platform. The easiest approach to determine the location of the ball is to utilize a resistive touch sensor on the platform, but given the goal of creating a low-cost system, this would defeat the purpose. A more practical approach to acquire the position of the ball is by using a plug-and-play (USB) camera. This allows utilizing the platform of any material available, like wood, cardboard, or plastic. By tracking the color of the ball, the movement and the position can be easily tracked. This could be accomplished by using any programming language to process the image taken by a camera in (quasi) real-time. We opted for the cheapest available option with the restriction that at least 30 FPS are needed for our application. Note that, according to our experiences, if the FPS is lower than 20, the control setup will be unstable. For simplicity, a V-shaped wooden stick is chosen as the platform where the ball can roll back and forth. The connection interfaces of different elements are created by a (low cost) 3D printer.

The total material price of the project is around € 20 at the time of writing this article. The implemented GUI, manual, programs, and recommended exercises can be found in (Csurcsia, 2020), (Bhandari, Ball Balancing Platform, 2021). The entire project, together with the theoretical description and more detailed analysis can be found in the master thesis (Bhandari, Development of control engineering demonstration setups, 2021) and on YouTube where a few videos illustrate different practical aspects of the PID controller (Bhandari, Control engineering: Demonstration setup, 2021).

2.2 Main components of the proposed PID controller

There are several aspects that must be considered when designing a PID controller for any application. Although they are well explained in theoretical lectures, it can be more interesting for the students to get hands-on experience on those aspects and to understand it in a practical way. The following subsections will explain the most important aspects that are considered in the proposed demonstration setup.

2.2.1 The derivative term

In an idealistic case, a measurement is noise free. However, in real-life situations, there is always a presence of noise, disturbances, imperfections in the measured signals. In most cases, the to-be-controlled systems have low pass filter characteristics. This means that valuable information can be found in the lower frequency band (Aström & Hägglund, 1995). The main issue with the ideal derivative controller is that it amplifies the high-frequency components, which are in typical applications purely noise. Despite the fact that it is discussed in the theory classes, many students were completely unaware that the signals are indeed corrupted by noise and that the performance of the controller can be improved by the use

of a simple low pass filter (with adjustable time-constant) on the derivative term.

A further user choice is the use of derivative kick. While controlling a system, a reference value is set, and the controller drives the process variable towards the reference value. Throughout this process, the reference value is constant and plays no role in the derivative controller. However, when the reference value is set to a new value, then there will be a sudden change in the error signal which triggers the derivative path to react to this sudden change. This is called derivative kick. To avoid this effect, the derivative path can be subjected only to the process variable instead of the error signal – as the error signal is the difference between the reference value and process variable. Students could by changing the setpoint realize that the erratic 'spiky' behavior of the ball is indeed related to the derivative kick effect.

2.2.2 Integral wind up

A physical actuator saturates beyond its working region. Suppose there is a disturbance that causes a constant error, for instance, blocking the ball using a finger to roll towards the reference value. As the result, the I controller branch keeps integrating the error signal. The actuator will respond until it reaches its saturation region and beyond that the actuator saturates because of the physical limitations. Once the disturbance causing the constant error is dissipated, the process variable will change by which error decreases and the control signal from the controller also decreases. However, the actuator will not immediately react to the decreasing actuating signal. This time is taken to cancel out the accumulated controlled signal beyond the saturation region. This is called integrator wind up (Aström & Hägglund, 1995). To avoid this problem, the integral controller should be turned off when the actuator saturates by limiting the output of the controller with the help of a saturation check. This functionality can be activated and observed by the students.

2.2.3 Actuator and the controller algorithm

The governing equation of a PID controller is mainly represented in three forms: series, parallel and standard representation. The standard representation is widely used because of its flexibility while tuning the controller (Nise, 2020). We make use of the standard form as:

$$C(s) = K_C \cdot E(s)\left[1 + \frac{1}{s \cdot \tau_I} + s \cdot \tau_D\right] \quad (1)$$

Here, adjusting the gain K_C will impact all branches equally. Students will perceive this during manual tuning of the controller where they just have to focus on the time constant of integral and derivative controller and a controller gain. Further, they also observe that this type of representation is straightforward for various tuning techniques that can be used to extract the controller's parameters.

It is important to understand the behavior of each component used in the closed-loop system. This is usually done by analyzing if there is any manipulation on the signal passing through a component. The manipulation can be amplification, integration, derivation, or delay of the observed signal. In presence of a component that manipulates the signal, the designer of the PID controller must know its influence on the process variable of the system and treat it in a proper manner.

In this application, the servo motor works as an integrator. This means the output of the servo motor is the integration of the controlled signal. To cancel out the integration effect, the input to the servo motor must be derived before feeding it into the servomotor. This is usually done by taking the derivative of the velocity form of equation (Aström & Hägglund, 1995) (Åström & Wittenmark, 1984):

$$C(s) = K_c \cdot E(s)\left[s + \frac{1}{\tau_I} + s^2 \cdot \tau_D\right] \quad (2)$$

In the software to control the demonstration setup, the controller equation is not derived, instead, a separate block is used as a signal manipulator in-between the controller and the actuator. The signal from the controller is derived with this signal manipulator block. Following, the servomotor integrates this signal and cancels out the derivation action. Using the separate block to derivate the controlled signal allows the student to understand the correct way of handling the actuator which works as an integrator. Since the signal manipulator block contains multiple options, this also lets the students observe what happens when the component manipulating the signal is not treated properly. This concept was hard to grasp for our students, to guide them, we created Figure 3 (Csurcsia, 2020).

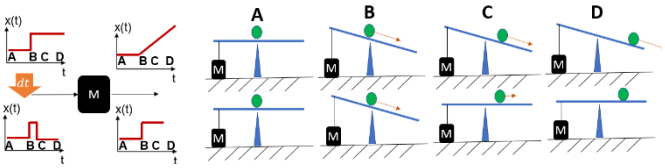

Figure 3: Illustration of the servomotor integrator behavior.

2.3.4. Dead band

Due to imperfections, the steady-state process variable and the reference value may not be entirely equal even with the appropriate PID controller parameters. However, the PID controller tries continuously to achieve zero error. This causes the ball to roll back and forth around the reference value. To avoid this movement, a band is introduced as a dead band where the controller output is deliberately set to zero. This allows the controller to settle the process variable within the selected band of a static error. In this closed-loop system, the dead band is applied to two signals. One on the error signal and the other on the manipulated signal. The dead band on the error signal means that some error within the band is allowed during the controlling action. Further, the dead band is applied to the manipulated signal. It is necessary because it can be noticed that a small change in the angle of the servo arm does not lead to a change in the position of the ball. A dead band is applied to make sure that the non-zero input signal to the servo motor changes the position of the ball.

2.3.4. Software implementation

We chose the Python programming language as the basis for the GUI program because our students are familiar with it. It is used to access the USB- webcam and process the image in

real-time to acquire the position of the ball (Nelson, 2019), (Rosebrock, 2015). Implementation of the controller and the development of GUI is also done in python (Franklin, Workman, & Powell, 2006). Further, the communication between the program in python and Arduino is developed (Bell, 2020). In each stage, some improvements have been made to make the demonstration as user friendly as possible (Csurcsia, 2020).

3. RECOMMENDED EXERCISES

3.1 *Introduction*

In our situation, engineering technology students have a more practical background. We provide many examples to help the students develop an understanding of the influence of each step in the control theory. In practical sessions, we have used Simulink to simulate various control situations. Even though in our practical sessions, we provided many exercises, students still had problems with the application of their knowledge to real-life situations. Therefore, this exercise comes in handy. To make sure that all students are well prepared, a detailed user's manual is provided describing the calibration process and a short introduction to the practical aspects applied in the demonstration setup. Depending upon the profile of the students, we recommend devoting 3-4 hours to this exercise. The demonstration setup must undergo a calibration step before starting the (controlling) action. This is necessary because each setup has been deliberately made slightly different. With this difference, we can avoid the copy-paste-not thinking over mechanism. After calibrating the setup, the position of the ball can be controlled. Because the students must roughly estimate the controller parameters based on the observation from the calibration process, the first tuning of the controller is done manually. This also allows the students to analyze the effect of different variables on the closed-loop system. After the manual tuning, the students have to perform the Ziegler-Nichols closed-loop tuning method (Ziegler & Nichols, 1993) to illustrate how the controller parameters can be extracted using some simple measurements. Upon completion of each main exercise, a short discussion with the lab instructor is foreseen.

3.2 *Calibration*

The calibration is needed to be done because each demonstration setup has slightly different driving properties. There are some aspects that must be considered during calibration. For the camera, the color of the ball must be distinct within its field of view. The transfer characteristic of a servo motor is also not linear throughout its working range. Thus, the minimum and the maximum angle of deflection should be estimated, to be in the linear region (Blanco, Csurcsia, Peeters, Janssens, & Desmet, 2018). Further, the time necessary for the servo arm to rotate a certain degree must be determined with the help of the datasheet of the servo motor. Students were asked to write every relevant calibration value of the process in their report.

Once the calibration is performed, the correct signal manipulator of the servo motor (linear/integrator/derivative) must be chosen before starting the tuning exercises. Here, despite the provided detailed information on the servomotors, we have encountered serious difficulties. To cope with this issue, students were asked to draw an (approximate) waveform to step excitation at each part of the controlled process. This helped them understand better the whole process. The overview of this process can be seen in Figure 4.

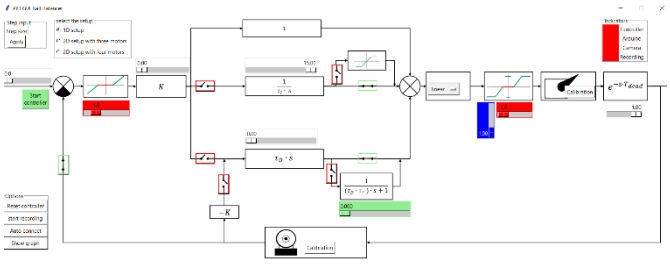

Figure 4: Interactive block-diagram based GUI of the setup.

3.3 *Manual tuning*

The aim of manual tuning is to understand the influence of each branch (and each setting) of the controller individually on the output of the closed-loop system. Further, the influence of different non-idealities present on the components can also be observed. Three different types of balls are used to perform step response experiments. The balls are beer pong ball, ping pong ball, and rubber ball. The beer pong ball is the lightest and the ping pong ball is the heaviest among the three. The heaviest ball, thus, the rubber ball, is the easiest ball to control. This can be observed in Figure 5, Figure 6, and Figure 7 where the settling time of the green curve is lowest. This is because the heavier ball comes easier to rest from motion.

3.2.1. P controller

Figure 5 shows the step response of different balls. For the same step size, it can be observed that the three types of balls have different responses. The ball with the lightest weight, the beer pong ball, has the greatest settling time. The settling time for the system controlling the beer-pong ball, ping-pong ball and the rubber ball are 21.9, 2.6 and 1.16 seconds respectively. The difference between the settling time of the beer pong ball with the settling time of two balls is remarkable because it is difficult to get the beer-pong ball back to rest once it is in motion. This is also the reason that the beer-pong ball has the highest overshoot/undershoot. It can also be observed that on the graph that the system's response does not progress towards the reference signal.

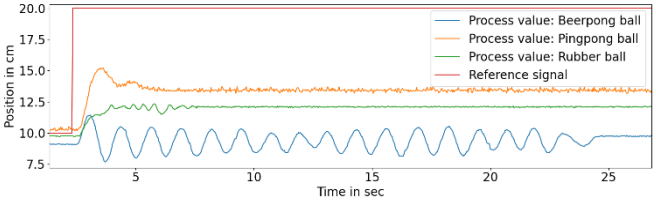

Figure 5: Step response of the system with P-controller using different balls.

3.2.2. PID controller

With the PID-controller, the output of the system converges towards the reference signal (see Figure 6). Concerning the dynamics of different balls, the settling time of the beer pong

ball is still the highest among the three different balls. On the graph, the effect of the two additional controllers, integral and derivative controller, can be clearly observed. It is known that the integral controller reacts to the static error present on the controller's steady-state response to eliminate it. Due to various non-idealities present on the different components, for instance, the color filter of camera, friction between the ball and the platforms etc. the static error is not always fully eliminated. However, the static error will be within the dead band when a dead band is applied. The response of the ping pong ball and the rubber ball were considerably better than the beer pong ball. The effect of the derivative controller can be observed by comparing the transient response of the system controlling the beer pong ball with P controller and PID controller. The settling time with P-controller was 21.9 seconds whereas it is reduced to 6.72 seconds with PID controller.

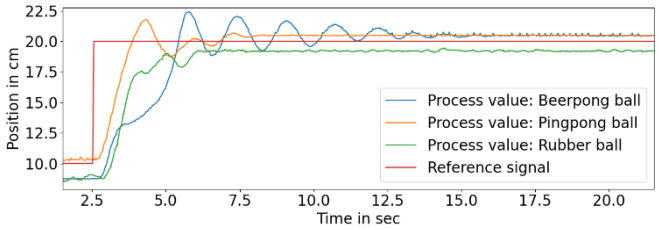

Figure 6: Step response with manually tunned PID-controller.

3.4 *Ziegler-Nichols closed loop tuning*

One of the most popular methods to determine the PID controller parameters using experimental data is proposed by Ziegler-Nichols in their paper (Ziegler & Nichols, 1993). The paper provides methods to help control engineers to have an initial parameter estimation of the controller parameters. They purposed two methods that theoretically provide the same controller parameters and can be applied according to the setup. The open-loop tuning technique is based on the open-loop step response of the system with a P-controller with unity gain. The closed-loop tuning technique is based on the closed-loop frequency response of the system with a P-controller with critical gain where the system is marginally stable. As the step response of the system in open loop does not provide a curve as expected from theory, we recommend the students to use the closed-loop tuning technique where the curve is much clearer, and it is easier to extract the necessary parameters. Three different balls are used to obtain their respective frequency response of the system. The controller parameters are then extracted by using the empirical formula provided by Ziegler-Nichols in their paper. The closed-loop method can be summarized as follows. First, only the proportional term should be activated and varied till marginal stability is achieved i.e. when the process variable oscillates. The gain at which the process variable oscillates is the critical gain of the system and the period of this oscillation is the critical period of the system. By using critical gain and critical period, the controller's parameter can be calculated easily.

Figure 7 shows the step response of the system controlled by the PID controller with the parameters obtained from the closed-loop tuning technique.

The result obtained from the closed-loop tuning is as good as the result obtained from the manual tuning. The time needed for the students to perform the tuning technique is much lower, less than 30 minutes, compared to the manual tuning case (more than an hour). For most students, the results obtained from the tunning method are better than the manual tuning.

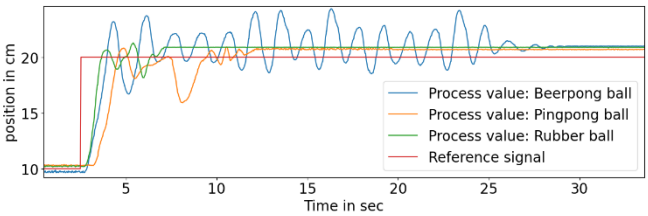

Figure 7: Step response with a ZN tuned PID controller.

4. DEVELOPMENT AND LEARNING PROCESS

4.1 Overview of the test phase

The development of the physical setup and the software was done in multiple stages. First, the different components have been individually tested to make sure every component works as expected. For instance, acquiring the position of the ball using the camera, communication between Python and Arduino, designing the GUI, etc. Then, everything was combined to make a complete software. This demonstration setup has been used as one of the practical sessions by our undergraduate students of engineering technology taking the control engineering course at Vrije Universiteit Brussel. Two small groups of students volunteered to partake in try-out sessions. These sessions were organized to see what difficulties they experience while using the setup and what they expect to be improved for the real lab session. For instance, most of them had to re-calibrate the setup when restarting the Python program as there was no auto-connect function to connect the setup with the software using the latest calibration data. This functionality was later added to the software. Further, there were also some problems while loading the code to the Arduino and downloading the necessary libraries. After the improvements, a manual with a short introduction to the demonstration setup – detailing the start-up and calibration process, and troubleshooting – was made available. At this stage, we finalized the project such that all the students could participate.

4.2 *Interaction with the students*

The students were divided into groups of two. A detailed user manual was provided to all the students, and they were expected to be well prepared. As this practical session was mainly focused on applying the theoretical knowledge in real-life application, various mistakes and confusion showed up. Those mistakes, issues were discussed with the student which helped them to understand certain concepts in a better way. In this session, some of the most typical observations are shared.

Most of the students were unaware of how the PID controller was implemented in the software. It was necessary for the students to explain why the controller equation typically found in the textbooks is not suitable to directly implement in the

software. After recalling the concept of the z-transformation, the students realized the necessity of the Tustin formula (Aström & Hägglund, 1995) to discretize the equation before implementing it as a controller in the software.

The first source of trouble was the appropriate choice of the manipulator of the servo motor signal (linear, integral, derivative). When the manipulator is not chosen correctly, the controller does not perform the task as expected. Even though the students have used the servomotors during different projects, it seems that the students are unaware about the influence of the servomotor on the signal flowing through it. It was necessary to explain how the servo motor works as an integrator and what it does to the signal.

None of the group initially applied the approach to tune the controller based on the time constants of the components, despite the fact, that they were similar Simulink exercises before this session. These constants could directly be derived by analyzing the results of the first exercise: the calibration process. This allows to estimate roughly the integration and derivative time constants of the controller by using the first and second dominant time constant. The lack of applying this knowledge of initial time constant estimation was the main reason that it took more than one hour for every to perform the manual tuning experiment.

5. CONCLUSION

In this work, a cost-effective and engaging PID control setup is built. The students can use the demonstration setup on their own to control different balls. During the calibration and both manual tuning and Ziegler-Nichols closed-loop tuning, different practical challenges are encountered by the students where they can associate those problems with the theoretical lectures where various aspects are assumed ideal. Even though the proposed setup is simple, it mimics the behavior of a real-life, industrial controller setup: imperfections, process and measurement noise, choice of appropriate tunning techniques. Hence, the practical session with this demonstration setup builds the insight for the students to take the correct approach to control any closed-loop system with a PID controller either by using manual tunning, tuning based on the time constant of the components, or one of the most popular tuning techniques.

REFERENCES

Arduino Team. (2018, April 16). *PID Temperature Control with Arduino*. Retrieved October 19, 2020, from Arduino: https://blog.arduino.cc/2018/04/16/pid-temperature-control-with-arduino/

Aström, K. J., & Hägglund, T. (1995). *PID controllers: theory, design, and tuning.* Research Triangle Park, N.C., International Society for Measurement and Control.

Åström, K. J., & Wittenmark, B. (1984). *Computer controlled systems: theory and design.* London: Prentice-Hall.

Beauregard, B. (2018). *Arduino Playground - PIDLibrary.* Retrieved November 28, 2020, from Arduino: https://playground.arduino.cc/Code/PIDLibrary/

Bell, C. A. (2020). *Beginning sensor networks with XBee, Raspberry Pi, and Arduino: sensing the world with Python and MicroPython.*

Bhandari, P. (2021, October). *Ball Balancing Platform*. Retrieved October 30, 2021, from Github: https://github.com/PujanBhandari/Ball-balancing-platform

Bhandari, P. (2021, October). *Control engineering: Demonstration setup*. Retrieved October 30, 2021, from You Tube: https://www.youtube.com/playlist?list=PLoh7bDqp3hUnENrVe9Ns2mMkPU5cY8QTG

Bhandari, P. (2021). *Development of control engineering demonstration setups.* Vrije Universiteit Brussel.

Blanco, M., Csurcsia, P., Peeters, B., Janssens, K., Desmet, W. (2018). Nonlinearity assessment of MIMO electroacoustic systems on direct field environmental acoustic testing. *Proceedings of ISMA 2018 - International Conference on Noise and Vibration Engineering and USD 2018 - International Conference on Uncertainty in Structural Dynamics.* Leuven, Belgium.

Csurcsia, P. Z. (2020). *Website of Dr. Ir. Péter Zoltán Csurcsia*. Retrieved March 14, 2021, from http://homepages.vub.ac.be/~pcsurcsi/teaching.html

Electric Diy Lab. (2018, November 17). *Arduino PID based DC motor position control system*. Retrieved October 22, 2021, from Youtube: https://www.youtube.com/watch?v=K7FQSS_iAw0

Electronoobs. (2018). Retrieved November 14, 2020, from MAX6675 Arduino Tutorial: http://electronoobs.com/eng_arduino_tut24.php

Electronoobs. (2019, July 14). *PID balance control*. Retrieved November 5, 2020, from Electronoobs: http://www.electronoobs.com/eng_arduino_tut100.php

Franklin, G. F., Workman, M. L., & Powell, J. D. (2006). *Digital control of dynamic systems.* Half Moon Bay, Calif, Ellis-Kagle Press.

Link, J. (2018, December 24). *Ball Balancing PID System*. Retrieved October 17, 2020, from Youtube: https://www.youtube.com/watch?v=57DbEEBF7sE

Mechatronics Tutorials. (2014, July 2). Retrieved October 17, 2020, from http://mechatronicstutorials.blogspot.com/2014/07/balancing-of-ball-on-beam-using-arduino.html

Nelson, A. (2019). *OpenCV computer vision examples with Python: a complete guide for dummies.*

Nise, N. S. (2020). *Control Systems Engineering.* [S.l.], JOHN WILEY.

Rosebrock, A. (2015, September 14). *OpenCV Track Object Movement-PyImageSearch*. Retrieved October 27, 2020, from Ball Tracking with OpenCV: https://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/

Ziegler, J. G., & Nichols, N. B. (1993). Optimum Settings for Automatic Controllers. *Journal of Dynamic Systems, Measurement, and Control.*, 115, 220-222.