VRIJE
UNIVERSITEIT
BRUSSEL

BALL BALANCING BEAM

PID controller design

Pujan Bhandari, Péter Zoltán Csurcsia

April 7, 2021

Industrial Engineering

Contents

1 Introduction

The goal of this session is to control the position of a ball on a V-shaped beam using a PID controller. The beam is deflected using a servo motor to make the ball roll back and forth on the beam. The computer is used as the controller and a microcontroller is used to drive the servo motor. A USB connected webcam is used as feedback sensor to read the position of the ball. Performing different measurements and analysing the measured data will help to understand the different aspects of the practical implementation of the PID-controller.

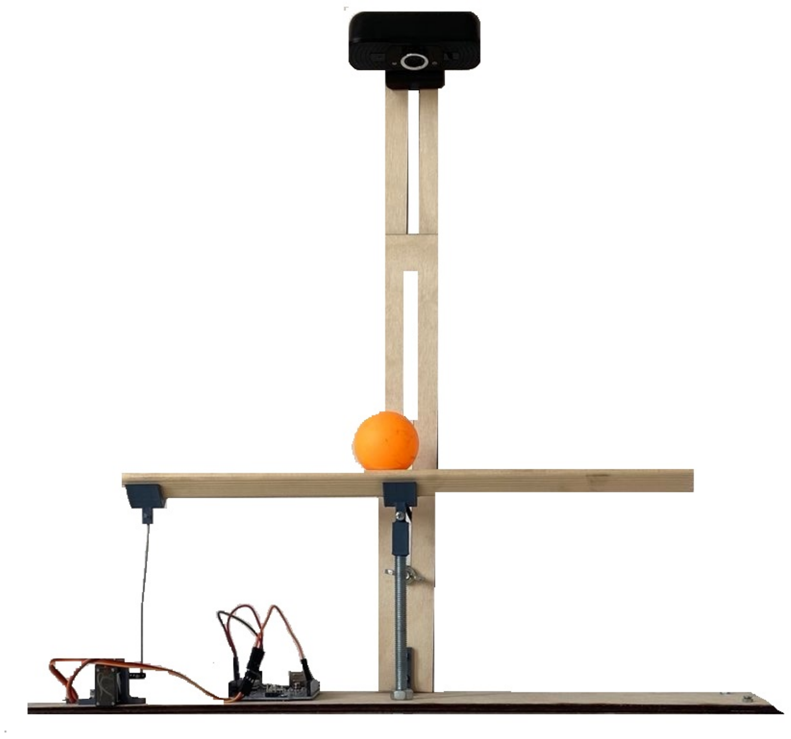


Figure 1: Ball balanicing demo setup.

2 PID - Controller

PID controller is a part of a feedback control system. In a feedback control system, the output of the system, also known as process variable, is measured by a sensor and used to compare with the desired value, also known as set-point. The difference between the desired value and process variable is known as error signal. The main role of the controller is to manipulate the error signal in such a way that the process variable to be equal to the desired value, and the error will ultimately converge to zero. PID stands for Proportional, Integral and Derivative, respectively.

2.1 Algorithm used to design the controller

There exists different types of representation of PID algorithm. Series representation, standard representation and parallel representation are widely known representations. Because of convenience, the algorithm of the ideal/ standard representation is used to design the controller for this setup. This representation is compatible for manual tuning as well as the different tuning techniques like Ziegler-Nichols-open -and closed loop tuning. This is the main reason that ideal representation of PID controller is used in the exercise sessions of the course.

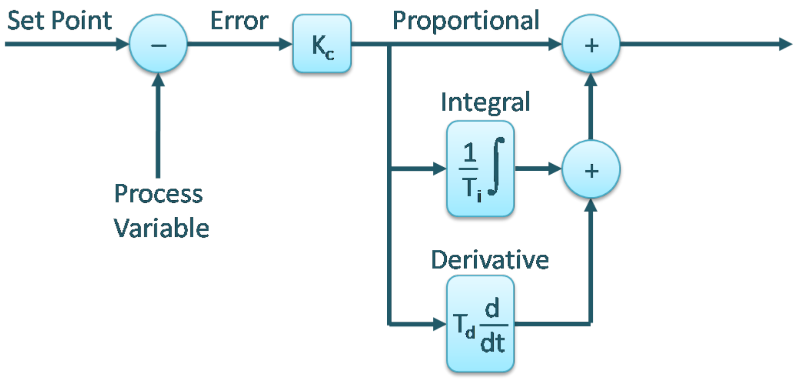


Figure 2: Standard representation of the PID controller.

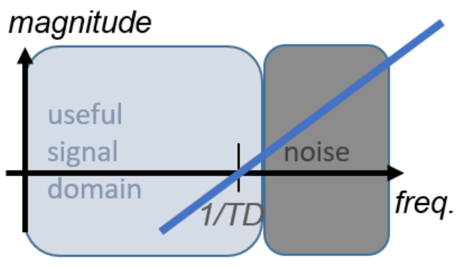
$$U(s) = K_c[1 + \frac{1}{\tau_{I \cdot s}} + \tau_D \cdot s] \cdot E(s) \tag{1}$$

In the proportional branch of a PID controller the error signal is multiplied with some gain: $U(s) = K_c \cdot E(s)$.
Integral controller keeps summing up the error and keep correcting until it fades away: $U(s) = K_c \cdot \frac{1}{\tau_I \cdot s} \cdot E(s)$.
In the derivative branch the rate of change of the error signal is multiplied with the constant τ_D. The faster the error signal is changing the larger the output of the derivative path: $U(s) = K_c \cdot \tau_D \cdot s \cdot E(s)$.

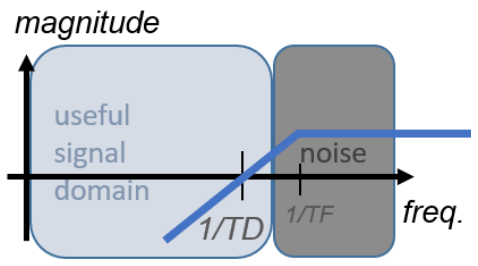
3 Aspects to consider while implementing PID controller in practice

3.1 Ideal derivative vs practical derivative

In case of ideal PID controller [1] , it is considered that the measurement is noise free. In case of practical application, it is not true because of presence of noise, disturbances, imperfections in the measured signal. Noise is a random disturbance which is unavoidable in real-life measurements. In most cases, the system have low-pass filter characteristics with valuable information in the lower band. The ideal derivative controller amplifies the high frequency components, which are in typical application only noise. That is why there must be a Low Pass Filter present along with the ideal derivative to filter all high frequency noises which may cause unwanted impact on the controlled output.



(a) ideal D controller shown in the frequency domain.

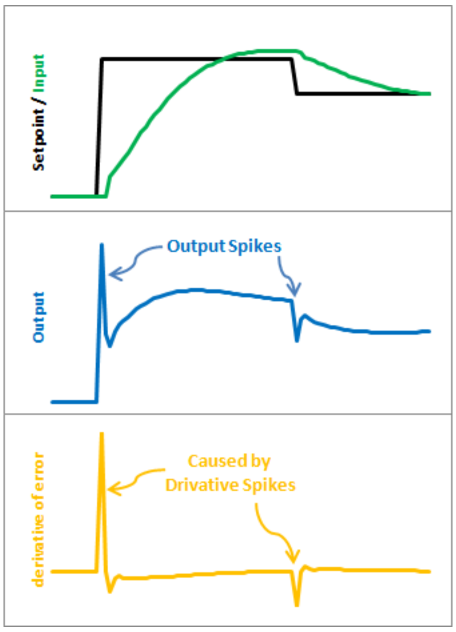


(b) ideal D controller combined with a low pass filter.

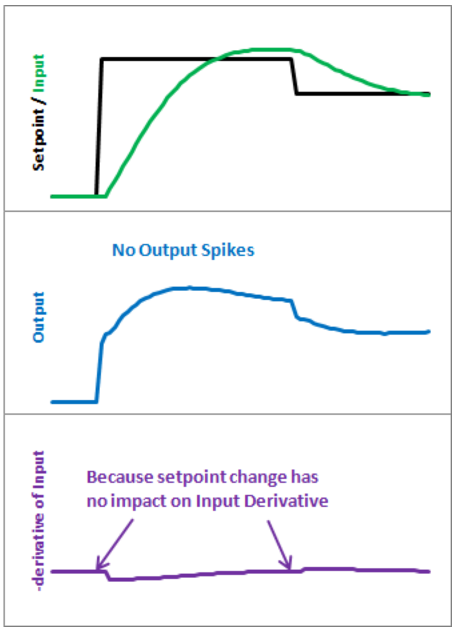
Figure 3: Derivative controller without -and with LPF.

3.2 Derivative kick

While controlling a system, a reference value is set and the controller drives the process variable towards reference value. Throughout this process the reference value is constant and plays no role in derivative controller. But if the reference value is set to new value than there will be a sudden change which triggers the derivative path to react to this sudden change abnormally. In the derivative, this is called derivative kick. To avoid this derivative kick, derivative path can be subjected to the measured signal instead of the error signal as error signal is the difference between the reference value and measured value.



(a) Presence of derivative kick



(b) Absence of derivative kick

Figure 4: Avoiding the derivative kick by taking the measurement as input for derivative controller.

3.3 Integral windup

A physical actuator works only within the range of its working region, beyond it, it can saturate. For example, there exists certain minimum and maximum angle of the servo arm. Beyond this point, it can saturate. While designing the controller one must consider the fact that the controller is not aware about the saturation region of the actuator. If there is a constant error present in the system due to various reasons, the integral path will accumulate the signal beyond the saturation region of the actuator due to which it may take time to decline back to the working region of the actuator. When the signal from the integral path reaches above the saturation region, it dominates the controller and block the effects of the other (P and D) controllers. To avoid this effect, the integrator should be turned off whenever the actuator saturates.

3.4 Deadband

Deadband is a range of input value where the output of the system is set to zero. Here deadband is applied on twee signals. When the ball reaches certain error value, the error signal is set to zero. This prevents the servo motor from frequent activation and deactivation (hunting) of the servo motor around the desired value.

For example, if the dead band of error signal is from 0 till 0.5, than the servo motor will not act with in the region which makes ball more stable around the set point. Further the deadband is applied on the actuating signal. We can notice that a small change in angle of servo arm does not leads to the change in ball position. So, to make sure that the non zero output changes the position of the ball dead band is applied.

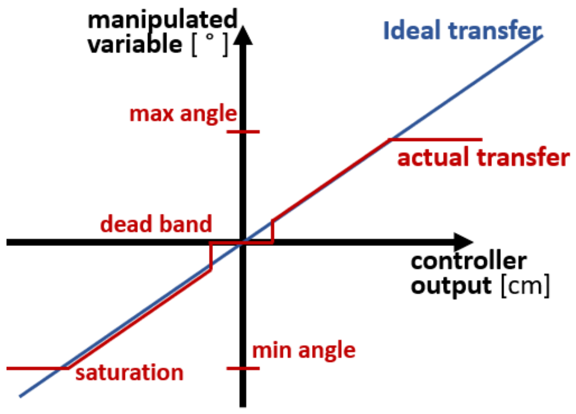


Figure 5: Change in the transfer function after applying deadband.

3.5 Discretization of PID equation

It is important to convert the PID equation from continuous domain to discrete domain in order to be able to implement in a software. This is done by transforming laplace domain equation to z-domain by using bi-stable/ Tustin formula, with ΔT as sampling time: $s = \frac{2}{\Delta T} \cdot \frac{z-1}{z+1}$, $X[n] \cdot z = X[n+1]$

4 Introducing the measurement setup

4.1 Acquiring the position of the ball using camera and computer.

The open source library, opencv2, is used in python to track the ball by detecting the colour of the ball. The colour of the ball is represented in HSV (Hue, Saturation and Value) colour space which is widely used in computer vision and image analysis. Hue represents the dominant colour observed through the eye. Saturation indicates the amount of the white light mixed with Hue. Value represents the darkness of the colour the lower the value the darker the colour. Once the HSV colour range of the ball is known, the ball moving on the beam can be tracked.

4.2 Using the computer as a controller

Once the actual position of the ball and the desired position of the ball is known, the error signal is calculated and used as the input to the controller. There are various options available to activate the different path of PID controller.

4.3 Using the Arduino as the driver

The Arduino is used to take the value from the serial port and sent it to the servo motor as it is straightforward to drive servomotor. With the help of the servo library of Arduino, it is possible to send the desired angle to the servo motor using one of the PWM pin of the Arduino.

4.4 Servomotor as an actuator

The servomotor is used to tilt the beam up and down to let the ball move back and forth. A servo works as an integrator. This means that the servo motor integrates the input signal. If the input signal of the motor is position, the output signal will be displacement. If the desired output is for instance position, than the input signal should be the derivative of the position.

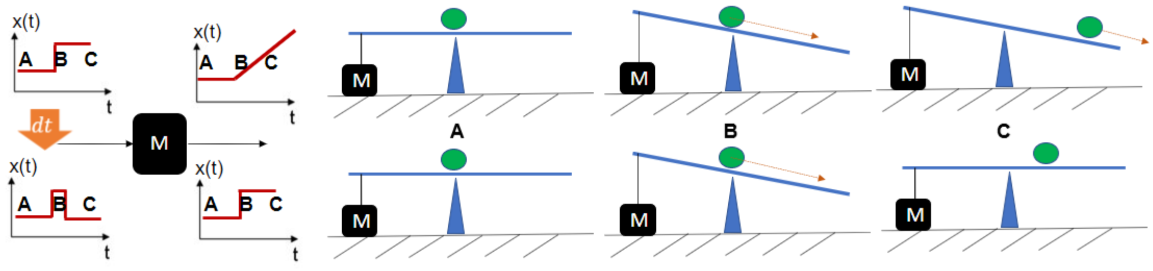


Figure 6: Position as an input signal to servo motor gives displacement as output signal (above). Derivation of position as an input signal to the servo motor results in position output due to the integrator nature of the servomotor (below).

4.5 Signals throughout the closed loop

The reference value is given by the user which will obviously be in the unit of length: cm. The position of the ball is measured in pixels by the camera. The length of the beam should be measured in cm and it should be used to convert the unit of position of the ball in cm. As both reference value and process variable are in cm the error signal will be in cm. The error signal is fed to the controller and the controlled output will be in cm as well. The controller output should be fed to the actuator. Here the servomotor is the actuator thus the signal in the unit of length should be converted into the angle of the servo arm (°). When the signal is position and fed into the actuator, it should be differentiated (see figure6). This should be done because the servo motor works as an integrator.

5 Installation

1. Installing Arduino IDE and uploading the code to the Arduino. This may be an optional step to perform. Consult with your instructor before you proceed.

   - Download the Arduino IDE suitable for your computer from the given link below:

     https://www.Arduino.cc/en/software
   - Open the provided Arduino code with Arduino IDE.
   - Compile the code by clicking the tick button on the top left side of the screen.
   - If the code is successfully compiled, click on item menu and click on ports. Select an available com port. The selected port is the port where the Arduino is connected. Note the port name, for example com3, com5 etc, it will be needed in further steps. If the option ports are inaccessible, try troubleshooting steps provided on the appendix of this document.
   - Once the port is selected, upload the code by clicking the arrow button pointing towards right next to the compile button. The code is successfully uploaded to the Arduino board if the message "Done Uploading" appears.

2. Check the suitable frame per second for the webcam by using a external website given below:

   https://webcamtests.com/fps
   After checking the FPS of the camera, make sure that the website is not possessing the camera anymore.

6 Exercises

Your lab instructor will provide a printed template and further instructions for completing these exercises. The block diagram that you will see in the GUI represents the ball balancing demo setup which is needed to be tuned. Follow the following steps to be able to start the controller successfully.

6.1 Calibrating the drive system

1. Click on the button calibration next to the icon of servomotor.
2. Click on the button refresh com ports. Choose the com port connected to the Arduino from the list. If you are not sure which port is assigned to Arduino:
   - Open the device manager (or run the command: devmgmt.msc)
   - Open the Ports (COM & LPT) list, look for the item ©USB-SERIAL CH340 ©.

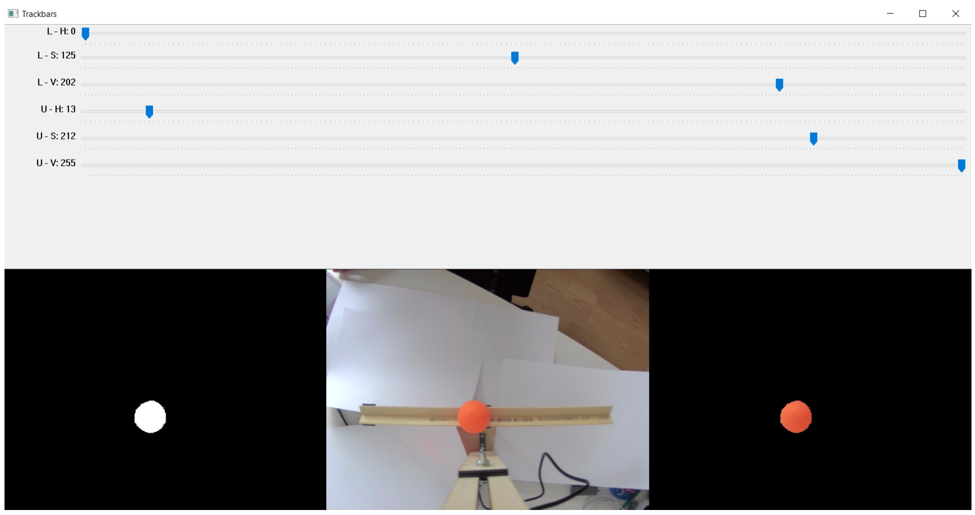
   If the Arduino is not recognized, on the homepage of the course there is a windows driver available for download.
3. Click on the button Connect Arduino and wait till the message Arduino connected appears.
4. It is important to have the beam horizontal before starting the controller. Servo motor rotates from 0 to 180 degrees and ideally at 90°the beam should be horizontal. Check if the beam is horizontal and make adjustment by giving the angle around 90°(± 20°) via the text box and sending it to Arduino by clicking on the button apply.
   If the neutral angle is not within the 90°(± 20°) interval, disconnect the sting at servoarm, apply 90°angle and connect the servoarm such that it remains almost horizontal.
5. After the neutral angle is obtained, check what is the maximum and minimum effective deflection angle by using same text box and button to send an angle to servomotor. Use ruler to check distance between the base plate and the beam when it is deflected. When the beam does not move significantly further, than you have reached the maximum angle.
6. Enter the minimum angle and maximum angle.
7. Click on the button OK. The window will shut down if everything goes right. If the error message appears than check if the inputs are given correctly and try again.

8. Now, the drive system is calibrated. The following step will be calibrating the camera.

6.2 Calibrating the camera

1. Click on camera calibration button to open a new window to connect the camera. Input for the camera number is an index number of the camera, starting from 0. Using the next step (6.1.3), find out what is the right camera index (start with 0, then 1, etc.)

2. Click on Open HSV window. This will open the window where there are six sliders on the top of the window and three frames at the bottom of the window. The goal is to find the HSV colour range of the ball. By sliding the slider of H,S and V for lower and upper bound you should try to eliminate all colours except the colour of the ball. Only the ball should be white on the left frame and only the ball should be seen on the right frame. First of all, try to find the range of the color seen by our eyes. This is done by sliding L-H slider and U-H slider. The lightness and darkness can be than adjusted by slider L-S,U-S and slider L-V,U-V respectively. Once this is done, click on X button to close the window. This will save the determined HSV value to a file which will be accessed by the program while connecting camera on further steps.
   Be advised: the camera can easily pick up locations of similarly colored objects (e.g. cables, LED, etc). If it happens, we recommend to use a piece of paper to mask the area where the problem occurs.



L - H: 0
L - S: 125
L - V: 202
U - H: 13
U - S: 212
U - V: 255

Figure 7: Successfully finding the HSV range of an object.

3. Click on the button Connect camera and wait till the message Camera connected appears. If the message is Camera Connection Failed, than re-check the steps and make sure that all inputs, i.e. camera number is given correctly.

4. By clicking on button show video, it can be checked if the camera is connected and the ball is tracked properly. It is recommended to keep the video window open because issues with the tracking (e.g. flickering, choppy video stream) can be easily identified. If it happens, we recommend to check your settings or restart your computer.

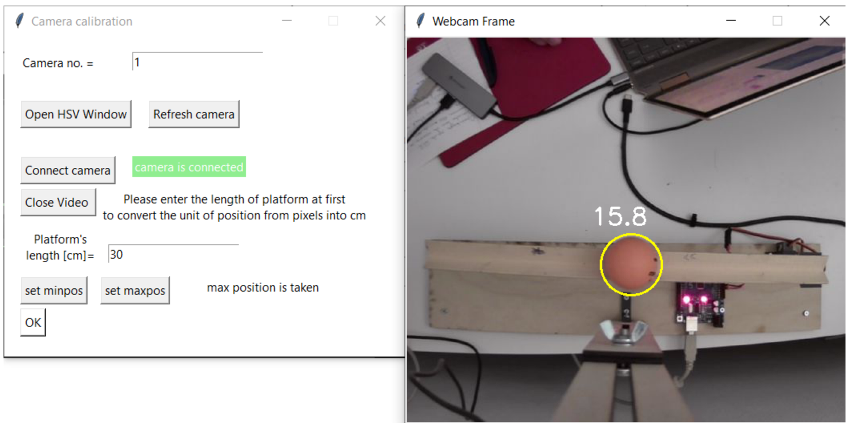
5. Measure the length of the beam that the camera can cover if the ball rolls on it. Give the length in the textbox.

6. Place the ball on the minimum position that the camera can cover and click on the button set minpos. "Min pos is set" will appear after successfully reading the minimum position.

7. Place the ball on the maximum position that the camera can cover and click on the button set maxpos. "Max pos is set" will appear after successfully reading the maximum position.

8. Click the button OK. The window will shut down if everything goes right. If the error message appears than check if the inputs are given correctly and try again.

9. The camera is connected, The controller is now ready to start.



Camera no. = 1

camera is connected

Please enter the length of platform at first
to convert the unit of position from pixels into cm

Platform's length [cm]= 30

max position is taken

Figure 8: GUI showing the position of the ball[cm] after camera calibration.

The system is now ready to start. The goal is to find the tuning parameters by using different methods. There are nine sliders available to change the reference point, deadband for error, K, τ_I, τ_D, τ_F, maximum deflection angle, deadband for the angle and servo deadtime respectively from left to right. Dead time servo means the time between two servo signal. Servo typically takes 0.1 sec to rotate about 60° angle. Thus, estimate a suitable dead time based on the range of the operating angle that you calculated while calibrating the system. Different buttons are available which act as switches to connect or disconnect any branch. There is also a drop down box available with various options from which you should be able to choose the right one. There is also a button called record data available to save the data as a .csv file which can later be used to make necessary calculations. There is a png image available on the homepage of the course which gives an overview of different objects used on the GUI.

The goal of this session is to understand the role of each blocks representing the part of the closed loop by analysing the signal flowing through it. For example, what happens when the signal flows through ideal derivative controller and what happens when the LPF is activated.

6.3 Manual tuning

Manual tuning is performed by changing the controller parameters and observing the response of the system. Some practical observations:

1. Proportional controller multiplies the error signal with some constant K. When the error is large, the output signal from the proportional controller is also large but when the error gradually degrade the output signal is also small. The system comes to an equilibrium with some static error. Increasing the gain decreases the static error but the overshoot increases and the stability of the system decreases.

2. Integral controller is used to eliminate the static error from the system. It is done by summing up the error over time which accumulates the signal.Thus even the tiniest error presence in the system will eventually force the controller to correct it. On the other hand the integral controller will cause an overshoot and the stability of the system decreases.

3. Derivative controller reacts to the rate of change of the error. Thus increasing the parameter τ_D will help to decrease the overshoot.

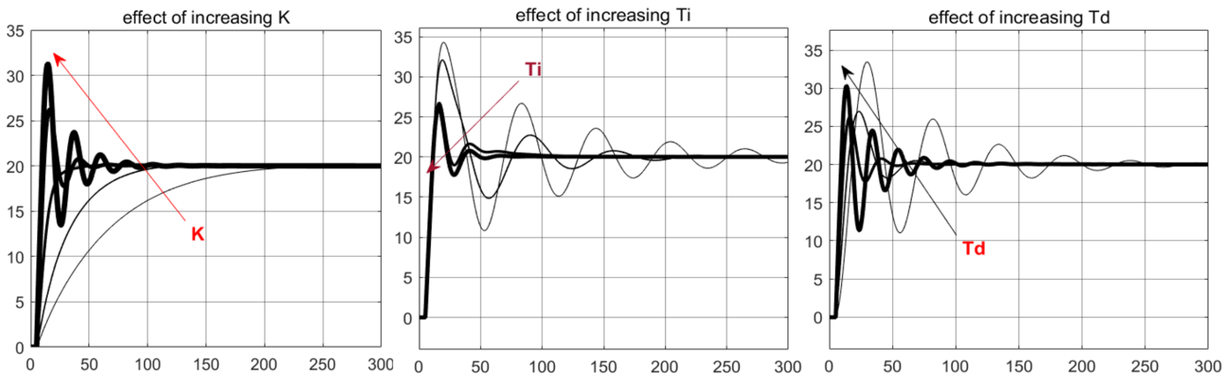


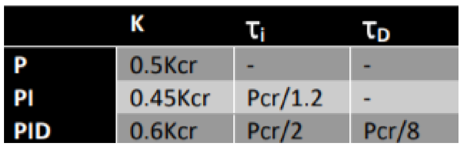
Figure 9: Effects of increasing K ,τ_I and τ_D on the PID controller.

6.4 Ziegler–Nichols method

6.4.1 Ziegler-Nichols closed loop method

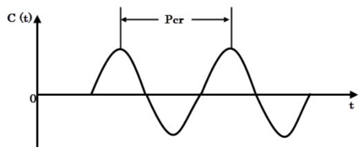
This method works for the closed loop systems which oscillates at certain gain level.

1. Turn off any integrator and differentiator. Set K to a low value.
2. Measure the step response of the closed loop system.
3. Gradually increase K until the system just starts to oscillate, this value is the K_{cr} – critical gain.
4. Estimate the period time of the oscillation, see figure.
5. Apply the following controller parameters:



	K	τi	τD
P	0.5Kcr	-	-
PI	0.45Kcr	Pcr/1.2	-
PID	0.6Kcr	Pcr/2	Pcr/8

(a) Table: ZN-closed loop tuning



(b) Oscillating response of the system.

Figure 10: ZN closed loop tuning.

6.4.2 Ziegler-Nichols open loop method

1. Turn off any integrator and differentiator. Set K (gain) to 1.

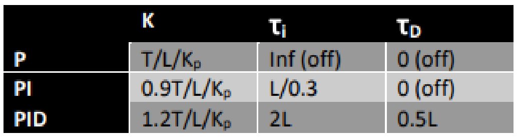
2. Measure a step response of the underlying open loop system. Using unit step excitation, you may not be able to see the significant change in the process variable because of the small step. In this case you can use larger step and calculate the static gain accordingly.

$$K_p = \frac{y_{final} - y_{start}}{u_{final} - u_{start}} \tag{2}$$

   You may notice that the response is not ideal as given in the figure below. The goal of this exercise is to approximate the parameters by using obtained step response of the system.

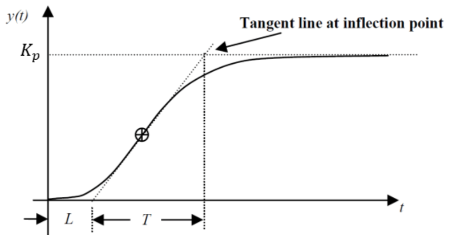
3. Estimate the gain K_p, time delay L and the dominant time constant T.

4. Apply the following controller parameters:



	K	τ_I	τ_D
P	$T/L/K_p$	Inf (off)	0 (off)
PI	$0.9T/L/K_p$	L/0.3	0 (off)
PID	$1.2T/L/K_p$	2L	0.5L

(a) Table: ZN-open loop tuning



(b) Unit step response of the system

Figure 11: ZN open loop tuning.

Appendix

7 Troubleshooting

If your computer is not reading a USB port, it may be caused by the absence of the driver on your computer. Please surf to the following link to install or update the driver:
https://learn.sparkfun.com/tutorials/usb-serial-driver-quick-install-/all
If the steps on the link doesnot work, try following:

- Find arduino.inf file on your computer. It should be located at
  C:/Program Files (x86)/Arduino/drivers
- Right click and select install.
- After completing the installation, go to device manager and find ports. You can now find the arduino uno and the connected com port.