INTERACTIVE VIRTUAL EXPERIMENTAL SETUP FOR CONTROL SYSTEMS ON THE UNITY GAME ENGINE

TANAPHON SRIYIM

TANAPORN SUTHAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN MECHANICAL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2021

INTERACTIVE VIRTUAL EXPERIMENTAL SETUP FOR CONTROL SYSTEMS ON THE UNITY GAME ENGINE

TANAPHON SRIYIM

TANAPORN SUTHAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN MECHANICAL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2021

Year of Thesis 2021

Department of Mechanical Engineering

Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Thesis Title: INTERACTIVE VIRTUAL EXPERIMENTAL SETUP FOR CONTROL SYSTEMS ON THE UNITY GAME ENGINE

Students:

1. Mr. Tanaphon Sriyim Student ID: 61010431
2. Ms. Tanaporn Sutham Student ID: 61010442

_____ Advisor

(Asst. Prof. Dr. Nattawoot Depaiwa)

INTERACTIVE VIRTUAL EXPERIMENTAL SETUP FOR
CONTROL SYSTEMS ON THE UNITY GAME ENGINE

Mr. Tanaphon Sriyim 61010431
Ms. Tanaporn Sutham 61010442
Asst. Prof. Dr. Nattawoot Depaiwa Advisor
Year 2021

ABSTRACT

This thesis contains the development of the control system game by the Unity game engine. The game's objective is to let the players experiment with the ball-beam and water-level control system dynamics and gain enjoyment from playing the game. The preparation starts from studying essential theories, i.e., control system, transfer function, PID controller, and computer software implementation. Then, design the control systems. The ball-beam one has a ball-beam itself, four-bar linkage, and DC-motor. The water-level one contains its plant and flows control pump. Next, determine each system's mathematical model and transfer function, most of the methods come from the literature review chapter. Then simplify them into the block diagram with the PID controller and disturbances. The game development process is done with four modules of the manual control, the PID controller, the disturbances, and the system configurations. Finally, the Unity system is verified by the MATLAB Simulink system with acceptable correctness.

Keywords: control system, the Unity game engine, ball-beam, water-level.

ACKNOWLEDGMENT

After this unfortunate time in the coronavirus disease of the 2019 pandemic, we would like to thank many people for their direct and indirect contributions to this thesis. First, we would like to thank our advisor Asst. Prof. Dr. Nattawoot Depaiwa for his suggestions, comments, and knowledge resources.

Thanks to everyone who made online education content and deployed it to open platforms for everyone to learn. These resources of knowledge not only drive this thesis to success but also drive the education of the whole world, which inspires this thesis to happen.

Finally, our heartfelt thanks to our caring family for their unceasing support and encouragement, without whom this thesis would not be possible.

Mr. Tanaphon Sriyim

Ms. Tanaporn Sutham

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1 Background

This project has begun by the combination of two main ideas, the first one is to build an experimental setup for Control Systems, and the second one is to make an educational purpose video game on the Unity game engine. Together, the concept is to develop a Control Systems experimental setup that is not too hard to play and does not miss any knowledge.

Control Systems is an essential field of study that engineering students should learn and understand because every system is designed to be controllable to acquire reliability. Otherwise, an unexpected problem might cause an unpredictable failure.

The project will contain two classic control systems, i.e., a ball-beam and water-level, excellent for demonstration and observation. The first is for position and balance control, and the second is for level and flow control.

Lately, the Unity game engine is now involved in many industries, not just video games. After the coronavirus disease of 2019 pandemic, the learning method was forced to change into an online platform. Both workshops and laboratories at the university or school are not allowed. Consequently, this project was decided to build on the computer instead of the actual experimental setup.

1.2 Objective

To develop the Control Systems game containing ball-beam and water-level on the Unity game engine that players can learn and experiment with.

1.3 Process of the Project

Step 1: Determine the mathematical model of the systems and prepare ready to utilize equations.

Step 2: Study and review every essential related theory, literature, and game development tool.

Step 3: Design the game objectives and components.

Step 4: Develop the game on the Unity game engine.

Step 5: Make a user manual and instructions for the game.

1.4 Expected Results

This project can provide the enjoyment of playing the game and the knowledge of experimenting with the Control Systems.

CHAPTER 2

LITERATURE REVIEW

2.1 Ball-Beam Control System

Jacob et al. (2011) have derived the mathematical model of the Ball-Beam system by referring to figure 2.1. The equation of motion of the ball-beam system can be found starting from Newton's Law of Motion. Neglecting friction and viscous damping. The sum of forces acting on the ball along with the beam equals:

$$m_b \left(\frac{d^2}{dt^2} x(t) \right) = F_{x,t} - F_{x,r} \tag{2.1}$$

Where m_b is the mass of the ball.

$F_{x,r}$ is the force from the ball's inertia.

$F_{x,t}$ is the translational force generated by gravity can be found as:

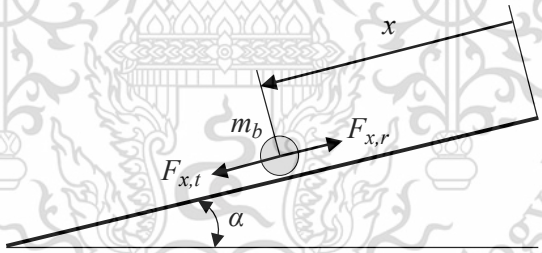

Figure 2.1 Free-body Diagram of Ball and Beam

$$F_{x,t} = m_b \, g \, \sin \alpha(t) \tag{2.2}$$

Then, the force acting on the ball in the x direction from its momentum becomes:

$$F_{x,r} = \frac{J_b \left(\frac{d^2}{dt^2} x(t) \right)}{r_b^2} \tag{2.3}$$

Solving for the linear acceleration gives:

$$\frac{d^2}{dt^2}x(t) = \frac{m_b\, g\, \sin\alpha(t)\, r_b^2}{m_b\, r_b^2 + J_b} \tag{2.4}$$

2.2 Four-Bar Linkage

Ashitava (2010) have studied the Freudenstein Equation by referring to figure 2.2. The analytical approach of Freudenstein obtained a simple scalar equation:

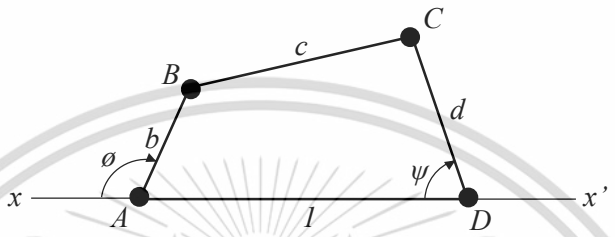

Figure 2.2 Four-Bar Mechanism

$$R_1 \cos ø - R_2 \cos \psi + R_3 = \cos (ø - \psi) \tag{2.5}$$

Where:

$$R_1 = \frac{l}{d} \tag{2.6}$$

$$R_2 = \frac{l}{b} \tag{2.7}$$

$$R_3 = \frac{l + b^2 - c^2 + d^2}{2bd} \tag{2.8}$$

2.3 DC-Motor System

Katsuhiko (2004) have derived the DC motor mathematical model of figure 2.3 by using both Electricals and Mechanical knowledge.

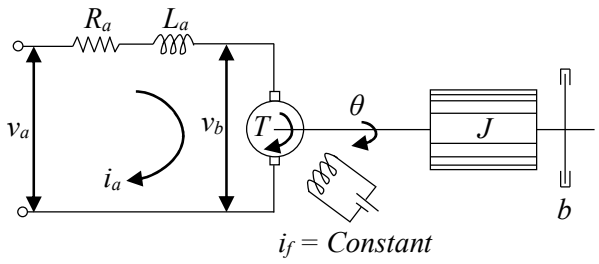

Figure 2.3 Armature-Controlled DC motor

Three equations of the armature-controlled DC motor were derived by the:

$$v_b = K_b \frac{d\theta}{dt} \tag{2.9}$$

$$L_a \frac{di_a}{dt} + R_a i_a + v_b = v_a \tag{2.10}$$

$$J \frac{d^2\theta}{dt^2} + b \frac{d\theta}{dt} = T = Ki_a \tag{2.11}$$

2.4 Water-Level Control System

Katsuhiko (2004) has derived the liquid-level systems mathematical model by using Fluid Mechanics knowledge. The equation represents the relationship of the inlet flowrate as an input and the water level as an output of the system.

$$RC \frac{dh}{dt} + h = Rq_i \tag{2.12}$$

Where R is the resistance for liquid flow in restriction.

C is the capacitance of a tank.

2.5 Pump Flow Control System

Yuqin Wang et al. (2021) have derived the mathematical model of the centrifugal pump flow control system which actuated by electric motor. The transfer function of returning the pump flow rate by giving the actuating signal as an input is shown as follows:

$$G(s) = \frac{k}{(T_L s+1)+(T_S s+1)} \tag{2.13}$$

Where k is pump gain factor.

T_L is the mechanical time constant.

T_S is the electrical time constant.

2.6 PID Controller Tuning Methods

Justin (2007) has done a test on the tuning methods and concluded that if the plant transfer function is known or can be reasonably modeled, then the following recommendations can be followed when tuning systems with pure integrations in their transfer function, the open-loop or closed-loop method be used. When tuning systems of order higher than two, the CHR or Closed Loop method should be used. However, for high-order systems with varying lags, the CHR method should be used.

CHAPTER 3

THEORY

3.1 Control System Definition

A control system contains plans and subsystems capable of returning the desired output with the optimal performance of a given input, as shown in figure 3.1.

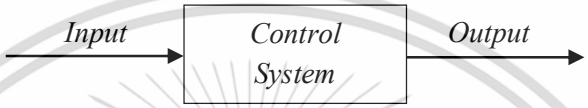

Figure 3.1 Control System

In figure 3.2, the response of the control system is plotted by the time. In this project, there are three characteristics of the system that is considered. The first one is overshoot, the difference between the maximum response and the desired output, which should not exceed the limitation of the system. The second one is the steady-state error, the difference between the steady-state response and the desired output, which needs to be as small as it can. The last one is the settling time. The time before the system is in the steady should not be too long.

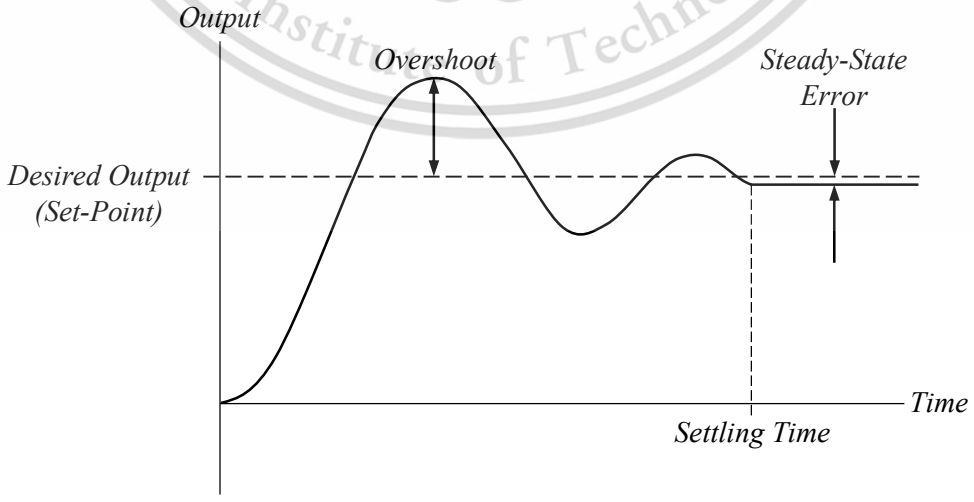

Figure 3.2 Response of the Control Systems

3.2 Control System Configurations

3.2.1 Open-Loop Systems

Open-Loop systems start with the input and then transform it into the input used that for the controller. The controller drives a plant, and other signals are shown added to the controller and process outputs, as shown in figure 3.3.

The disadvantage of the open-loop control system is that it only works with the input and output relationship. However, in the real world, some disturbances occur from unexpected circumstances, so the controller will not know what is happening after its actuating signal is sent to the plant.

However, it needs less equipment than the closed-loop system, so if the disturbance of the system is not that large compared to the cost and the complexity of the system, the open-loop system might be chosen.

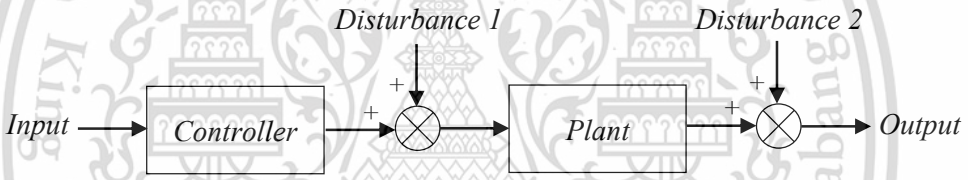

Figure 3.3 Block Diagram of Open-loop System

3.2.2 Closed-Loop Systems

A Closed-Loop system starts with the input then subtracted by the output, and the result is system error used for the controller. The controller drives a plant then the final output with disturbances gets back to the first summing point to calculate the system error to make a closed-loop, as shown in figure 3.4.

On the other side of the open-loop system, the advantage of the closed-loop system is how it can deal with the error even if external disturbances occur. Therefore, it was more widely used in the industrial.

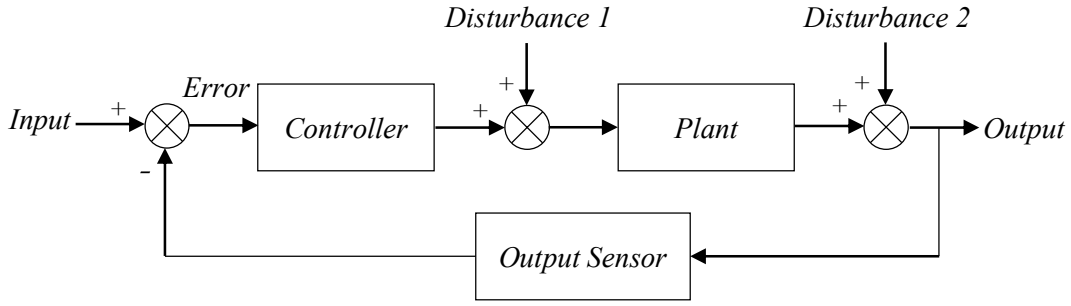

Figure 3.4 Block Diagram of Closed-loop System

3.3 Transfer Function

The more complex of the system, the more difficult of model it as a block diagram. The transfer function represents the precise relationship between input, output, and the system by taking the Laplace transforms with zero initial conditions, as shown in equation 3.1.

$$\frac{C(s)}{R(s)} = G(s) = \frac{(b_m s^m + b_{m-1}s^{m-1} + \ldots + b_0)}{(a_n s^n + a_{n-1}s^{n-1} + \ldots + a_0)} \tag{3.1}$$

Where $G(s)$ is the transfer function.

$C(s)$ is the output of the system.

$R(s)$ is the input of the system.

Note that there are several ways to turn the mathematical equation in the time-domain into the Transfer Function via Laplace transform. However, the easiest way is to do it by hand calculation by the table, and the elementary Laplace transform is shown in table 3.1. Most of the transfer functions can be found by using this table or computer software.

Table 3.1 Elementary Laplace Transforms from William et al. (2017)

$f(t) = \mathcal{L}^{-1}\{F(s)\}$	$F(s) = \mathcal{L}\{f(t)\}$
1. 1	$\frac{1}{s}, \quad s > 0$
2. e^{at}	$\frac{1}{s-a}, \quad s > a$
3. t^n, n a positive integer	$\frac{n!}{s^{n+1}}, \quad s > 0$
4. $t^p, \quad p > -1$	$\frac{\Gamma(p+1)}{s^{p+1}}, \quad s > 0$
5. $\sin(at)$	$\frac{a}{s^2+a^2}, \quad s > 0$
6. $\cos(at)$	$\frac{s}{s^2+a^2}, \quad s > 0$
7. $\sinh(at)$	$\frac{a}{s^2-a^2}, \quad s > \lvert a\rvert$
8. $\cosh(at)$	$\frac{s}{s^2-a^2}, \quad s > \lvert a\rvert$
9. $e^{at}\sin(bt)$	$\frac{b}{(s-a)^2+b^2}, \quad s > a$
10. $e^{at}\cos(bt)$	$\frac{s-a}{(s-a)^2+b^2}, \quad s > a$
11. $t^n e^{at}$, n a positive integer	$\frac{n!}{(s-a)^{n+1}}, \quad s > a$
12. $u_c(t) = \begin{cases} 0 & t < c \\ 1 & t \geq c \end{cases}$	$\frac{e^{-cs}}{s}, \quad s > 0$
13. $u_c(t)\, f(t-c)$	$e^{-cs}F(s)$
14. $e^{ct} f(t)$	$F(s-c)$
15. $f(ct)$	$\frac{1}{c}F\left(\frac{s}{c}\right), \quad c > 0$
16. $(f * g)(t) = \int_0^t f(t-\tau)g(\tau)\, d\tau$	$F(s)G(s)$
17. $\delta(t-c)$	e^{-cs}
18. $f^{(n)}(t)$	$s^n F(s) - s^{n-1}f(0) - \cdots - f^{(n-1)}(0)$
19. $(-t)^n f(t)$	$F^{(n)}(s)$

3.4 PID Controller

PID controller is a feedback control system widely used in industrial controlling because of the adjustability and reliable of it. The PID controller deals with the error, but there are three terms to work with the present, past, and future of the error represented by proportional, integral, and derivative terms, as shown in figure 3.5.

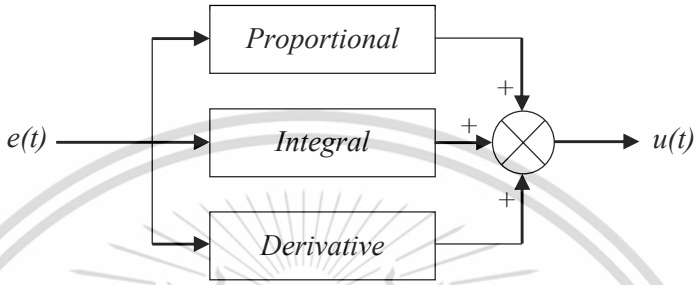

Figure 3.5 Block Diagram of PID Controller

The mathematical form of PID controller is shown in equation 3.2

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{3.2}$$

The transfer function of the PID controller becomes:

$$\frac{E(s)}{U(s)} = K_p + \frac{K_i}{s} + K_d s \tag{3.3}$$

Where $U(t)$ is the output of the controller.

$E(t)$ is the error of the system.

K_p is the proportional gain.

K_i is the integral gain.

K_d is the derivative gain.

Each PID controller gain has its duty. The first one is the proportional gain, which works by multiplication the value to amplify the error to make the actuating signal easier for detection. The second one is integral gain, which works with integration

by controlling the output into set-point, in other words, decreasing the steady-state error. The third one is derivative gain, which improves the transient response or settling time of the system by working with the rate of change. However, every PID term has its disadvantage that is not good for the system, so there are several tuning methods to deal with it.

3.5 Numerical Methods for Calculus

In the Unity game engine, in order to make the system has a good performance which is indicated by framerate, continuous-time domain is not a great choice for the real-time calculation because of its accuracy comes with the complexity, therefore, numerical methods or discrete-time domain will be utilized for this scenario.

3.5.1 Numerical Derivative Approximation

Derivative is the important calculation for the PID controller which is the main controller is this system, the derivative is the rate of change of the function as shown in equation 3.4

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{3.4}$$

In the system implementation, only present and past value can be obtained, so the backwards difference approximation will be used as shown in equation 3.5

$$\frac{df(x)}{dx} \approx \frac{f(x) - f(x-h)}{h} \tag{3.5}$$

The approximation of equation 3.5 can be written as the discrete function for the data array calculation on the coding as follows.

$$\frac{df(x)}{dx} = \frac{f[n] - f[n-1]}{T} \tag{3.6}$$

Where T is the sample time for the discrete-time system.

3.5.2 Numerical Integral Approximation

Another important calculation for the PID controller, numerical integral can be approximated from the Trapezoidal Rule in equation 3.7, where a and b is the interval of the sample time which is the small value between the discrete motion.

$$\int_a^b f(x)dx \approx (b-a)\cdot\frac{1}{2}(f(a)+f(b)) \tag{3.7}$$

The approximation of equation 3.7 can be written as the discrete function for the data array calculation on the coding as follows.

$$\int_a^b f(x)dx = T\cdot\frac{1}{2}(f[n]+f[n+1]) \tag{3.8}$$

3.6 Implementation of a Transfer Function in Game

In the game, the mechanics follow the coding algorithm defined as discrete-time domain array, not continuous like s-domain. However, the s-domain and Laplace is still in use because of the advantage of modeling the system that contains differential equation as a Transfer function.

Z-transform can model the Transfer function in the discrete-time domain and make it ready to utilize for in-game coding or in n-domain of data array. The below example shows how z-domain becomes the discrete-time domain the data array.

First, transform the s-domain Transfer function into the z-domain via hand calculation or computer software, i.e., MATLAB, Octave, and Python. The result of the transform will be in the z-domain:

$$\frac{Y(z)}{X(z)} = \frac{az^{-1}}{bz+cz^{-1}} \tag{3.6}$$

Then, transform the z-domain into n-domain or discrete-time domain:

$$b\,y[n] + c\,y[n-1] = a\,x[n-1] \tag{3.7}$$

Rearrange the equation to turn y into the explicit function of x:

$$y[n] = \frac{a}{b} x[n-1] - \frac{c}{b} y[n-1] \quad (3.8)$$

After the function, table 3.1 shows how data array in coding will be store in the game.

Table 3.2 Example of Discrete-Time domain data array structure.

Time	$n-1$	n	$n+1$
Input	$x[n-1]$	$x[n]$	$x[n+1]$
Output	$y[n-1]$	$y[n]$	$y[n+1]$

3.7 The System Design Process

Every process should be carefully designed because the result is not just the correct equations but needs to work on the game properly, as shown in figure 3.6.

Step 1: Study the systems and gather the requirements.

Before the mathematical things, the physical information should be clearly explained. Every requirement, specification, and limitation are needed.

Step 2: Determine the mathematical model.

This step uses Mechanics to turn the physical problems into the mathematical equations, time-domain specified, to make it clearer to calculate and analyze.

Step 3: Determine the transfer function.

In Control Systems, many design methods tend to be done on the s-domain by applying the Laplace transform to the continuous time-domain because it is easier to solve the differential equation.

Step 4: Design the PID controller.

In this step, the PID gain tuning methods will be chosen, the saturation of the actuator is considered. Compare the result with the uncontrolled system to see how it is improved.

Step 5: Implement the system in code.

After all the equations are already gathered, it needs to be transformed into the discrete-time domain to implement in the code of the game. This step will consider the performance of the code, so the algorithm should not be too complex to cause a long calculation time.

Step 6: Verify and validate the system.

Like every simulation, the system must be verified to check if the result is correct and validated to ensure the system is modeled in the right way. This step will compare the result of the game with the theoretical result because the game cannot be misleading any control system concepts.

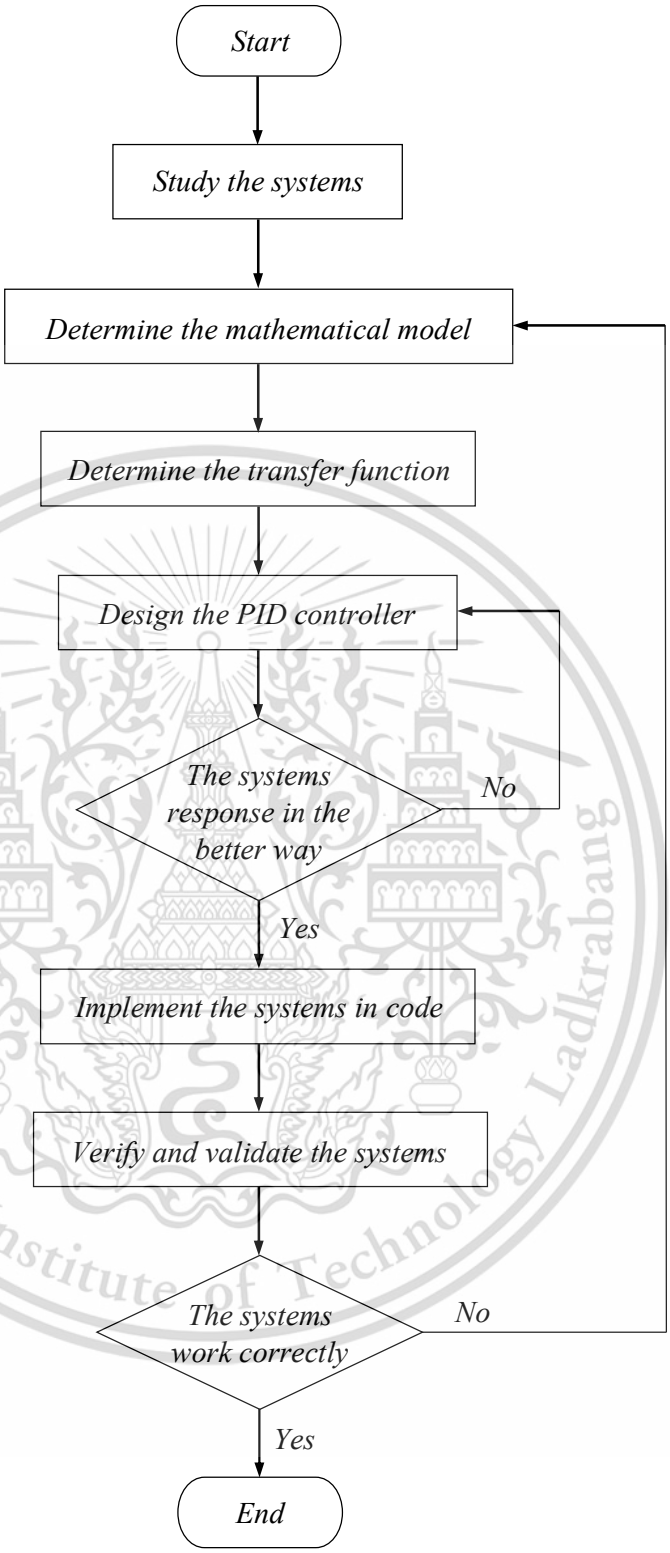

Figure 3.6 Flowchart of the System Design Process

3.8 Computer Software

3.8.1 The Unity Game Engine

The Unity game engine is a tool for developing interactive media, i.e., video games, mobile applications, robotic simulation, and VR media. Its first announced in June 2005 and spread used in an indie game, low budget game, because it has a free version with a powerful performance as shown in figure 3.7.

In this project, the Unity game engine is the main pipeline of the visualizing and interacting workflow. Everything is designed to be put together in this software.

3.8.2 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment for coding and developing software as shown in figure 3.8.

In this project, Visual Studio was involved in developing C# language and Python language, which are the main scripts of the game. Every code will be done it this software.

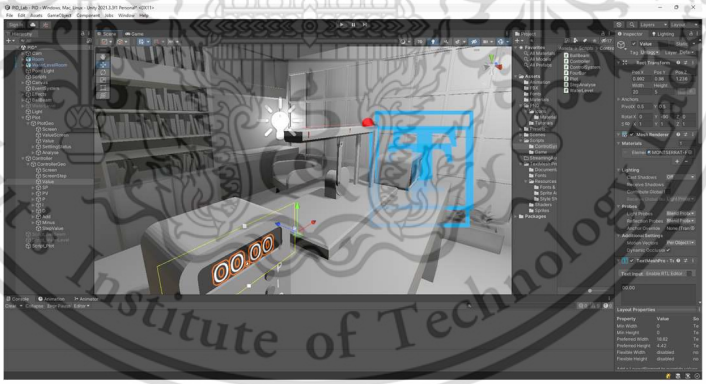

Figure 3.7 Unity Game Engine Workspace

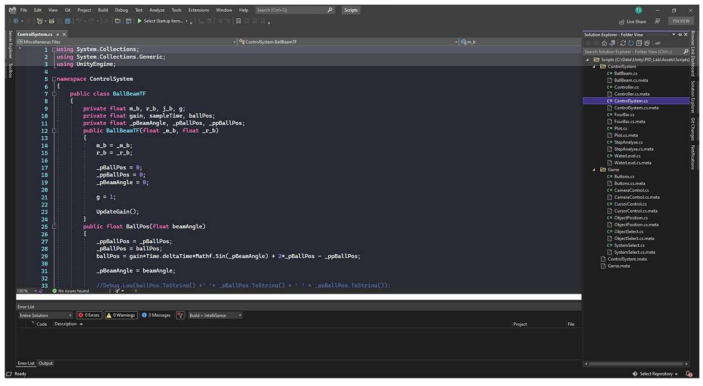

Figure 3.8 Microsoft Visual Studio Workspace

3.8.3 Blender

Blender is an open-source and free 3D modeling, video editing, and creating animated films software. With the large community of users, everything about computer graphics can be done with Blender as shown in figure 3.9.

In this project, Blender will be used for all the 3D modeling imported into the Unity game engine.

3.8.4 MATLAB

MATLAB is a scientific language and a numeric calculation software developed by MathWorks as shown in figure 3.10.

In this project, the responsibility of the MATLAB Simulink is to calculate the mathematical model theoretically to be used for the comparison with the game result.

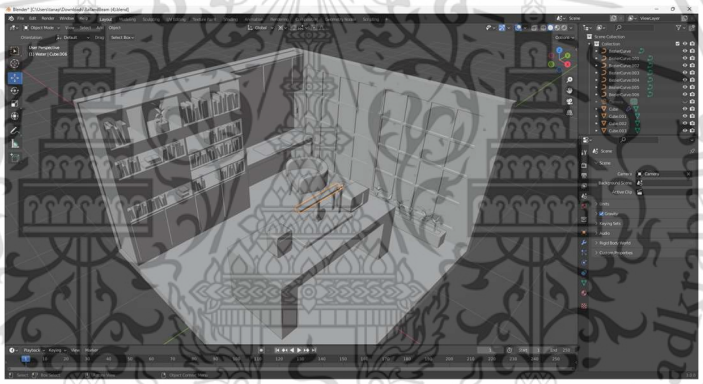

Figure 3.9 Blender Workspace

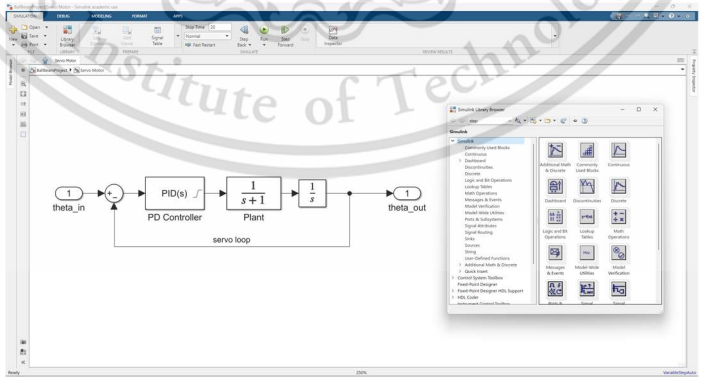

Figure 3.10 MATLAB Simulink Workspace

CHAPTER 4

CONTROL SYSTEMS DESIGN

4.1 Ball and Beam Control System Design

A ball-beam system in this project, as shown in figure 4.1, aims the have a clear visual of system input, output, and mechanism between them. Users can play with the disturbances of moving ball or beam.

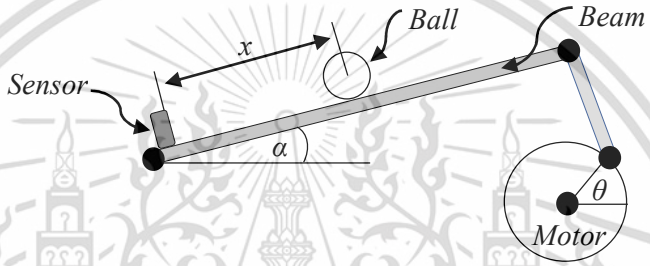

Figure 4.1 Ball and Beam Control System

4.1.1 System Requirements

All the essential information and specifications for ball-beam system needed for system design process are shown in table 4.1.

Table 4.1 Requirements of Ball and Beam Control System

Input	Desired ball position along the beam
Output	Actual ball position along the beam
Actuator	DC Motor
Sensor	Displacement sensor of ball position
Disturbances	Moving ball position
	Rotating beam angle
Limitations	Ball position must not exceed beam length
	Maximum angle and speed of the DC motor

4.1.2 Mathematical Model and Transfer Function

A ball-beam system contains three subsystems. The first one is the ball and beam, the plant of the system. The second one is the four-bar linkage mechanism, the link between the beam and the rotor disk. The third one is the DC motor because the rotating of the motor will be shown in the game, so the motor modeling needs to be correct.

4.1.2.1 Ball and Beam

This section aims to find the equation that describes the ball dynamics by giving the beam angle as an input.

Referring to equation 2.4. The linear acceleration equation contains the sine of the angle function term, which is complex to transform into a transfer function. The working angle of the beam in the game is narrow, so small-angle approximation takes place by simplifying the sine function of the small angle:

$$\sin\theta \approx \theta \tag{4.1}$$

Then, linearize the ball acceleration by equation 4.1:

$$\frac{d^2}{dt^2}x(t) = \frac{m_b g\,\alpha(t)\,r_b^2}{m_b r_b^2 + J_b} \tag{4.2}$$

After that, equation 4.2 is ready to take a Laplace transform:

$$\frac{X(s)}{A(s)} = \frac{m_b g r_b^2}{(m_b r_b^2 + J_b)s^2} \tag{4.3}$$

The transfer function of ball and beam system is the second order system with two poles at the origin, consequently, when the step function is an input, the system will be unstable.

Note that the beam angle is limited, and the ball position must not exceed the beam length.

4.1.2.2 Four-Bar Linkage

This section is responsible of transferring a DC motor angle to a beam angle by the relationship of the four-bar linkage.

Equation 2.5 has high accuracy when applied to the game. On the other hand, it is too complex for the Laplace Transform, so it will not be used in the mathematical model process.

Another approach is the small-angle approximation, and the equation describes the relationship between two angles by approximation method, so it will not be used in the game.

$$\alpha \approx \frac{d}{L}\theta \tag{4.4}$$

The equation becomes a linear relationship of two angles:

$$\alpha(t) = \frac{d}{L}\theta(t) \tag{4.5}$$

Where d is the radius of the motor disk.

L is the length of the beam.

Then, take a Laplace transform to the equation 4.5:

$$\frac{A(s)}{\theta(s)} = \frac{d}{L} \tag{4.6}$$

4.1.2.3 DC Motor

DC motor is the system's actuator, transforming the signal from the controller into an output disk that drives the beam.

From equation 2.9, 2.10, and 2.11 assuming that all are zero initial conditions, then taking the Laplace transforms and rewriting into gain and time constant form:

$$\frac{\theta(s)}{E_a(s)} = \frac{K_m}{s\,(T_m s+1)} \tag{4.7}$$

Which each constants defined as:

$$K_m = \frac{K}{R_a b + KK_b} = motor\ gain\ constant \tag{4.8}$$

$$T_m = \frac{R_a J}{R_a b + KK_b} = motor\ time\ constant \tag{4.9}$$

Where R_a is the armature resistance.

b is the armature inductance.

K is a motor-torque constant.

K_b is the back-emf constant.

J is the moment of inertia of the motor and load.

The transfer function of DC motor system is the second order system with one pole at the origin and another pole at negative of s-plane, consequently, when the step function is an input, the system will be unstable.

From the requirements of the system, the angle or speed output of the DC motor is limited by its specification.

4.1.3 Block Diagram of Ball and Beam

All transfer functions are ready for the block diagram. The first block of the system that gets the actuating signal from the controller is the DC motor, represented by equation 4.7. The second block which connects the actuator and the plant is the four-bar linkage is represented by equation 4.6. The last block or the plant, represented by equation 4.3, is the ball and beam relationship which starts with the disturbance of the beam angle and ends with the disturbance of the ball position. Then complete the loop with the PID controller block, as shown in figure 4.2.

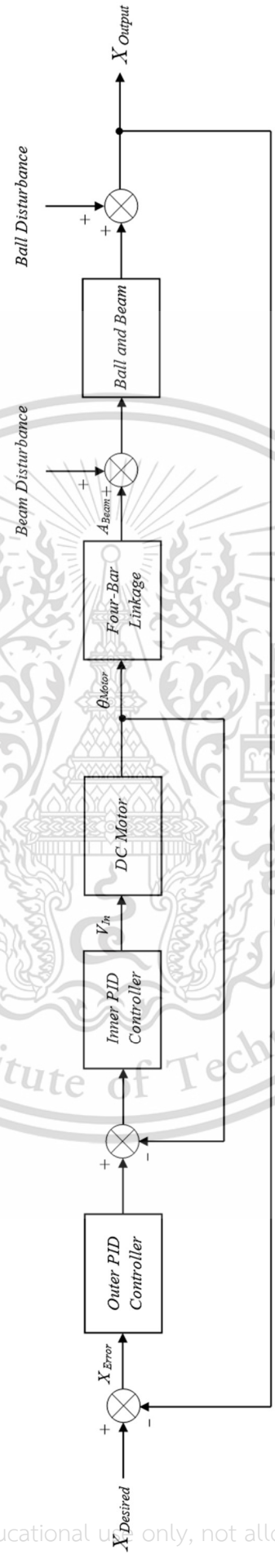

Figure 4.2 Block Diagram of Ball and Beam

4.2 Water-Level Control System Design

A water-level control system in this project, as shown in figure 4.3, is designed to let users play around with the mechanics of the classic liquid-level control system with one disturbance of remove or add the water of the system.

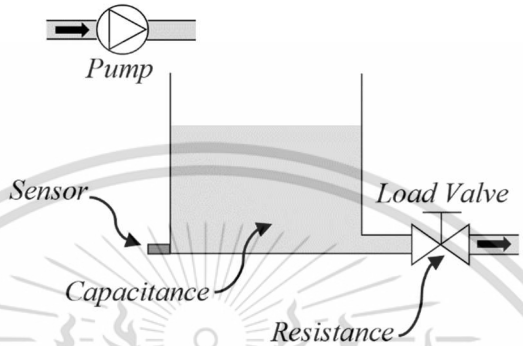

Figure 4.3 Water-level Control System

4.2.1 System Requirements

All the important information and specifications for the water-level control system that needed for system design process is shown in table 4.2.

Table 4.2 Requirements of Water-Level Control System

Input	Desired water level
Output	Actual water level
Actuator	Water Pump
Sensor	Hydrostatic pressure sensor
Disturbances	Filling or leaking of the water
Limitations	Water-level must not exceed tank height
	Maximum inlet flowrate of pump and control valve

4.2.2 Mathematical Model and Transfer Function

Similar to the ball-beam control system, the mechanics of the water pump is the actuator like the DC motor to build the flow up for the water tank plant. So, the actuator signal from the controller will input with the proportional control valve gain to the plant and become the output of water-level.

First, the plant dynamics are derived in equation 2.12 which is the differential equation, then the transfer function can be found by the Laplace transform:

$$\frac{H(s)}{Q_i(s)} = \frac{R}{RCs+1} \tag{4.10}$$

According to the system requirements, the limitation of the transfer function of equation 4.10 is the output must not exceed the height of the water tank.

4.2.3 Block Diagram of Water-Level

After finishing the transfer function, the block diagram of the water-level system starts with the control valve, transforms the actuating signal from the PID controller into the inlet flow rate, which is defined by equation 4.11, then goes to the plant of the water-level system, which defined by equation 4.10. Lastly, add the disturbance of the water's level in the final output, then close the loop by returning the error into the controller, as shown in figure 4.4.

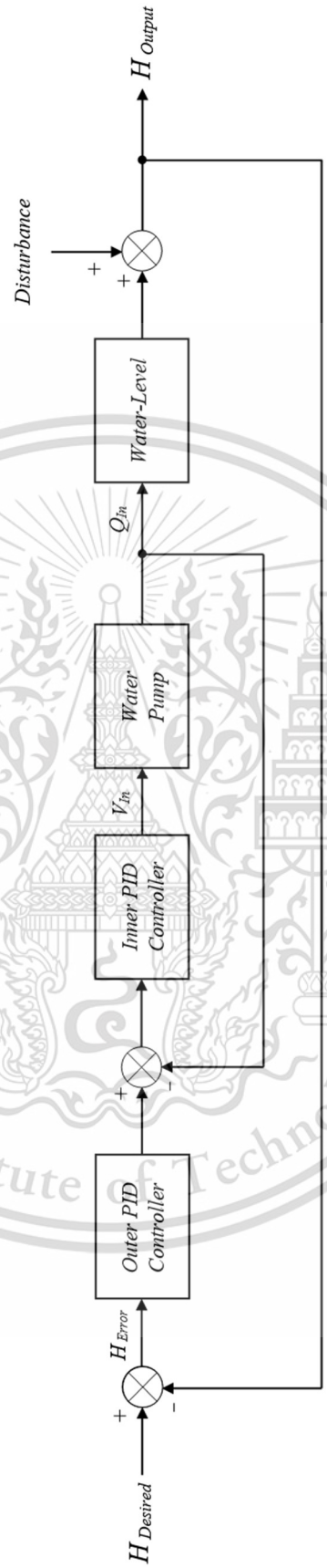

Figure 4.4 Block Diagram of Water-Level

CHAPTER 5

EXPERIMENTAL SETUP DEVELOPMENT

5.1 Experimental Setup Components

In the development process, each of the components was created by Blender software and then put together with the working scripts on the Unity game engine. In this section, only Control System-related components are discussed; therefore, all the decorations are excluded.

The virtual experimental setup of this project contains four main components as follows: a plant for demonstrating the mechanics of the system, an actuator for driving the plant, a controller for response control, and a response monitor for plotting the real-time response of the system.

5.1.1 Plant

The plant of the system is the component that describes the character of each system. Both plants are developed by following the design requirements in chapter 4.

The first plant is ball and beam system consisting of its plant and four-bar mechanism. According to figure 5.1, a plant has a ball that is supported by a beam which jointed with a DC motor disk with a four-bar mechanism. Comparing the distance with the sensor on the right-hand side, the output of the system is returned to calculate the error and monitor on the plotting screen.

The second plant is water-level control system. Referring to figure 5.2, the system consists of its plant on the cylindric tank and the control valve in the outlet which represents the characteristics of the system. The water level is measured by a hydrostatic pressure sensor.

Note that all the plant sensors are pseudo-objects, the output data comes from the mathematical calculation of the plant transfer function.

5.1.2 Actuator

Without the proper actuator, the overall system will look clearly unnatural. In this demonstration, the dynamics of the system are focused on, and all the displacement, speed, and limitations of the actuator must be obviously shown.

The first actuator is a DC motor of a ball and beam system in figure 5.1, a rotor disk represents a motor rotation for both displacement and speed.

The second actuator is a water pump of a water level system in figure 5.2, the light indicator is attached to the pump to show the working of the pump.

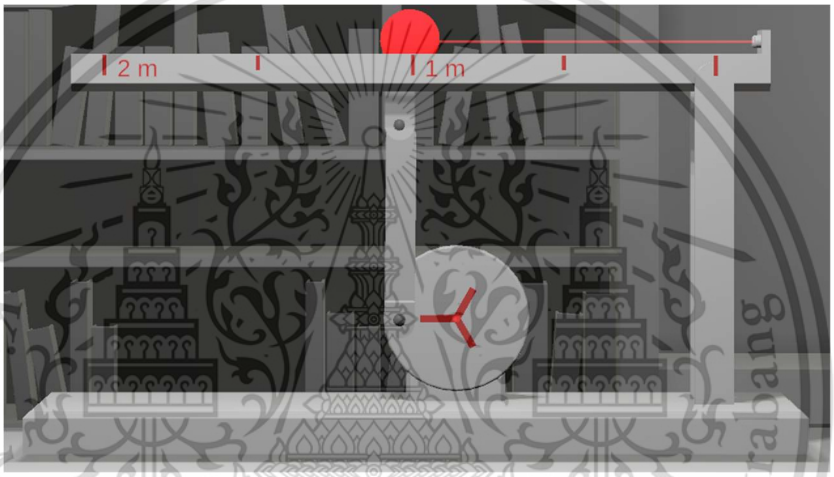

Figure 5.1 Plant and Actuator of Ball and Beam System

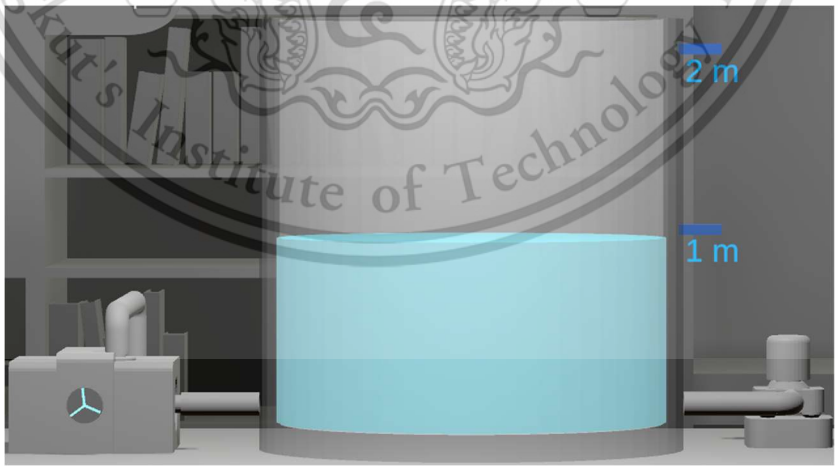

Figure 5.2 Plant and Actuator of Water-Level Control System

5.1.3 PID Controller

The main controller of both systems, and the main objective that is important for users to understand, so the component must not be too complex, but all the functions must not miss as well.

Referring to figure 5.3, the controller was created as a simple box with a display screen, mode indicator light, and interactive button. Besides the back-end logic behind the box, the user interface let the user input the setpoint, the three terms of PID gains, and the monitor the process valve together with the plot in the response monitor.

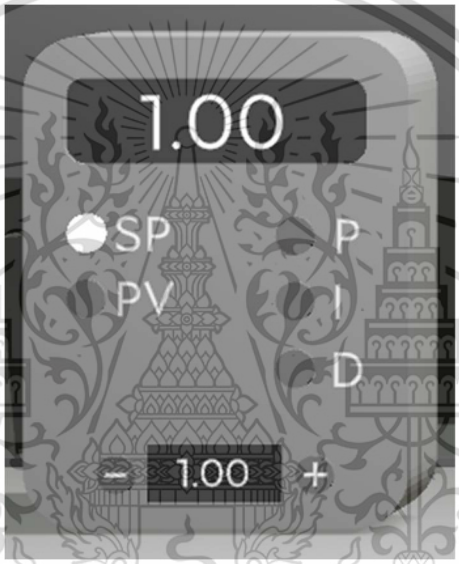

Figure 5.3 PID Controller of the System

5.1.4 Response Monitor

Real-time data inspection is very important in the control system. According to figure 5.4, the response monitor is created as a simple box with a screen size of 2:3 of the object, another portion is for the user input for adjusting the graph color, data axis length, and scrolling through the time-domain.

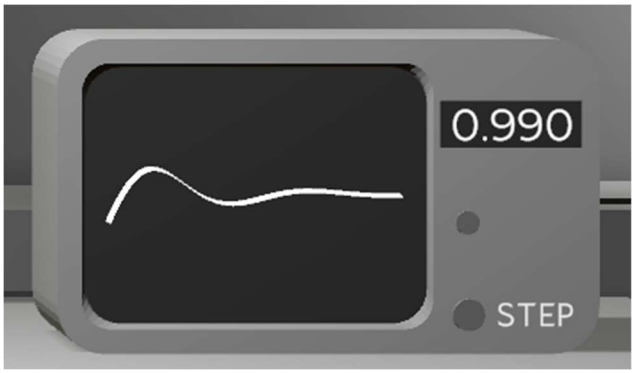

Figure 5.4 Real-Time Response Monitor of the System

5.2 System Implementation on Unity Game Engine

After the components in the Unity have been created, in order to make the system work, the C# scripting takes place in controlling all the system mechanics.

According to the theory of implementation in chapter 2, the procedure starts from derive the transfer functions in the z-domain from the s-domain, then transforming it into the discrete-time domain which is convenient for coding.

The important topic in working with the discrete domain is the sample time. In the Unity game engine, the sample time, T is represented as the time between the frame of the working scene, also known as delta time.

5.2.1 Ball and Beam System

The transfer function of the ball and beam system plant from equation (4.3) can be transformed into the z-domain as shown in the ramp form of equation (5.1).

$$\frac{X(z)}{A(z)} = kT\frac{z^{-1}}{1 - 2z^{-1} + z^{-2}} \tag{5.1}$$

Where k is the ball-beam gain constant.

From the discrete z-domain in equation 5.1, the equation for implementing on data array coding for the better calculation speed can be written as follow,

$$x[n] = kT \cdot \alpha[n-1] + 2 \cdot x[n-1] - x[n-2] \tag{5.2}$$

The equation 5.2 of ball and beam system can be coded in any language with the pseudocode in figure 5.5 of the function that get the beam angle as the input value then return the ball position as the output value.

```
FUNCTION BallPosition( beamAngle )
        previousPreviousBallPosition = previousBallPosition
        previousBallPosition = ballPosition
        ballPosition = gain * sampleTime * sin(previousBeamAngle)
        ballPosition += 2 * previousBallPosition
        ballPosition += -previousPreviousBallPosition
        previousBeamAngle = beamAngle
        RETURN ballPosition
END FUNCTION
```

Figure 5.5 Pseudocode of the Ball and Beam System

The next subsystem is the actuator transfer function of the DC motor from equation (4.7) can be transformed into the z-domain as shown in the asymptotic exponential form of equation (5.2).

$$\frac{\theta(z)}{V_a(z)} = k_m T \frac{cz^{-1}}{1 - (1+c)\,z^{-1} + cz^{-2}} \tag{5.3}$$

The complex exponential form is defined as the constant:

$$c = e^{-\frac{T}{T_m}} \tag{5.4}$$

Where k_m is the motor gain constant.

T_m is the motor time constant.

Referring to the discrete z-domain in equation 5.3, the equation of motion for implementing on data array programming for the better calculation speed can be written as follow,

$$\theta[n] = k_m T \cdot v[n-1] + (1+c) \cdot x[n-1] - c \cdot x[n-2] \tag{5.5}$$

The equation 5.5 of DC motor system can be programmed with the pseudocode in figure 5.6 of the function that get the voltage as the input value then return the motor angle as the output value.

```
FUNCTION MotorAngle( vIn )
        a = 1 / t_m
        previousPreviousMotorAngle = previousMotorAngle
        previousMotorAngle = motorAngle
        motorAngle = k_m / a * (1 – exp(-a*sampleTime)) * previousVIn
        motorAngle += (1 + exp(-a*sampleTime)) * previousMotorAngle
        motorAngle += -exp(-a*sampleTime) * previousPreviousMotorAngle
        previousVIn = vIn
        RETURN motorAngle
END FUNCTION
```

Fig 5.6 Pseudocode of the DC Motor System

5.2.2 Four Bar Linkage Mechanism

The complex mechanism of four bar linkage can not be derived as a transfer function by the conventional method like Laplace transform, but the mechanism motion can be demonstrated on the Unity game engine by apply the equation 2.5 to 2.8 which get the input value of input link which is DC motor angle and return two values of floating link angle which is the link component and output link angle which is the beam component angle as shown in the pseudocode of figure 5.7.

```
FUNCTION FourBarAngle( inputAngle )
        A = b*b + c*c
        B = 2*b*c
        r = d – a*cos(inputAngle)
        s = a*sin(inputAngle)
        f2 = r*r + s*s
        sigma = acos((A – f2)/B)
        g = b – c*cos(sigma)
        h = c*sin(sigma)
        outputAngle = atan2(h*r – g*s, g*r + h*s)
        floatAngle = angle[1] + sigma
        RETURN {outputAngle, floatAngle}
END FUNCTION
```

Figure 5.7 Pseudocode of the Four Bar Linkage Mechanism

5.2.3 Water-Level Control System

The transfer function of the water-level control system plant from equation (4.10) is transformed into the z-domain transfer function in the first-order exponential form of equation (5.3).

$$\frac{H(z)}{Q(z)} = RC \frac{cz^{-1}}{1 - (1+c)z^{-1} + cz^{-2}} \tag{5.6}$$

The complex exponential form is defined as the constant:

$$c = e^{-\frac{T}{T_w}} \tag{5.7}$$

Where T_w is the water level time constant:

According to the discrete z-domain in equation 5.6, the equation of motion for implementing on data array programming for the better calculation speed can be written as shown in equation 5.8.

$$h[n] = RC \cdot q[n-1] + (1+c) \cdot h[n-1] - c \cdot h[n-2] \tag{5.8}$$

The equation 5.8 of water level system can be programmed with the pseudocode in figure 5.8 of the function that get the water flowrate as the input value then return the water height level as the output value.

```
FUNCTION WaterLevel( flowRate )
        previousWaterLevel = waterLevel
        waterLevel = gain*sampleTime*flowRate
        waterLevel += exp(-sampleTime)*previousWaterLevel
        RETURN waterLevel
END FUNCTION
```

Fig 5.8 Pseudocode of the Water Level System

The actuating pump transfer function from equation (2.13) is transformed into the z-domain transfer function in the second-order double exponential form of equation (5.5).

$$\frac{Q(z)}{V(z)} = \frac{k_p(c_1 - c_2)}{T_L - T_S}\left(\frac{z^{-1}}{1 + (c_2 - c_1)\,z^{-1} + c_1 c_2\,z^{-2}}\right) \tag{5.9}$$

The complex exponential form is defined as the constants:

$$c_1 = e^{-\frac{T}{T_L}} \tag{5.10}$$

$$c_2 = e^{-\frac{T}{T_S}} \tag{5.11}$$

Where k_p is the pump gain constant.

T_L is the motor time constant.

T_S is the pump time constant.

According to the discrete z-domain in equation 5.9, the equation of motion for implementing on data array programming for the better calculation speed can be written as shown in equation 5.12.

$$q[n] = \frac{k_p(c_1 - c_2)}{T_L - T_S} \cdot v[n-1] + (c_2 - c_1) \cdot q[n-1] - c_1 c_2 \cdot q[n-2] \tag{5.12}$$

The equation 5.12 of water pump system can be programmed with the pseudocode in figure 5.9 of the function that get the voltage as the input value then return the water flowrate as the output value.

```
FUNCTION FlowRate( vIn )
        a = 1 / (t_L – t_S)
        previousPreviousFlowRate = previousFlowRate
        previousFlowRate = flowRate
        flowRate = k_p / a * (1 – exp(-a*sampleTime)) * previousVIn
        flowRate += (1 + exp(-a*sampleTime)) * previousMotorAngle
        flowRate += -exp(-a*sampleTime) * previousPreviousMotorAngle
        previousVIn = vIn
        RETURN flowRate
END FUNCTION
```

Fig 5.9 Pseudocode of the Water Pump System

5.2.4 PID Controller

Like other systems, the PID controller transfer function from equation (3.3) can be transformed into the z-domain transfer function in equation (5.8).

$$\frac{E(z)}{U(z)} = K_p + \frac{K_i T z}{z - 1} + \frac{K_d (z - 1)}{z} \tag{5.8}$$

Where K_p is the proportional gain.

K_i is the integral gain.

K_d is the derivative gain.

The equation 5.8 of the PID controller system can be programmed with the pseudocode which apply the numerical methods in figure 5.10 of the function that get the error as the input value then return the actuating signal as the output.

```
FUNCTION ActuatingSignal( error )
        actuatingSignal = pGain * error
        actuatingSignal += iGain * (error + previousError) * sampleTime / 2
        actuatingSignal += dGain * (error - previousError) / sampleTime
        previousError = error
        RETURN actuatingSignal
END FUNCTION
```

Fig 5.10 Pseudocode of the PID Controller

5.2.5 Actuator Saturation

When the actuating signal from the controller go to the actuator, not all range of output that will return from the system, every system has their limit also known as the actuator saturation.

In the implementation process, the limitation of the input signal is worked by the mathematical clamp function as shown in figure 5.11 which required upper limit and lower limit value then return the clamped value to the actuating signal.

```
FUNCTION Clamp( value, upperLimit, lowerLimit )
    IF ( value >= upperLimit )
        value = upperLimit
    END IF
    IF ( value <= lowerLimit )
        value = lowerLimit
    END IF
    RETURN value
END FUNCTION
```

Fig 5.11 Pseudocode of the Actuator Saturation Clamp Function

5.2.6 System Response Analysis

The most important thing in Control Systems is to understand the system response characteristics which are overshoot as explained in section 3.1, steady-state error, and settling time. All of the calculation uses the 5% error which is better in the demonstrate purpose comparing to the more accuracy one of 2% because of the high order system are quite hard to stabilize.

The first step is to collect the step response dataset array that is needed for the calculation which is start from the first data point when user hit the step button and stop collecting with the final data point when user observe that the system is stable then hit the step button again to finish. After that, the step response data array will be sent to the three main function of the system response analysis as follows.

Overshoot calculation is done by the concept of the maximum value that exceed the steady-state value which are roughly estimate as the set point value of the system then comparing the difference in percentage to return the overshoot percentage value as displayed in pseudocode of the figure 5.12.

```
FUNCTION Overshoot( stepResponseData )
        maxValue = max(stepResponseData)
        overshoot = (maxValue - setPoint) / setPoint * 100
        RETURN overshoot
END FUNCTION
```

Fig 5.12 Pseudocode of the Overshoot Calculation Function

Steady-State error calculation is done by the idea of the difference between set point and the final value when the system is in the stable region which is configured as 5%. If the value is not in the stable point, the function will return the status for informing users as shown in the pseudocode of the figure 5.13.

```
FUCNTION SteadyStateError( stepResponseData )
        stepCount = count(stepResponseData)
        steadyStateError = stepResponseData[stepCount - 1]
        IF ( (setPoint – steadyStateError) / setPoint > 0.05 )
                steadyStateError = NULL
        END IF
        RETURN steadyStateError
END FUNCTION
```

Fig 5.13 Pseudocode of the Steady-State Error Calculation Function

Settling time calculation is done by the concept of the time when the system response getting into steady state of 5% error, the process start with finding the stable point of the response from the final value to the first value which will get the first stable point then calculate the sample time with the amount of the data size before the system is stable before breaking the loop to return the settling time value as shown in the figure 5.14.

```
FUNCTION SettlingTime( StepResponseData )
        settlingTime = 0
        stepCount = count(stepResponseData)
        FOR ( i = stepCount - 1; i > 0; I -= 1 )
                IF ( abs((setPoint – stepResponseData[i]) / setPoint) > 0.05 )
                        settlingTime = (stepCount - i) * sampleTime
                        BREAK
                END IF
        END FOR
        RETURN settlingTime
END FUNCTION
```

Fig 5.14 Pseudocode of the Settling Time Calculation Function

5.3 User Interface and User Experience

The UI/UX of the software must be clear and easy to use. It contains an overall view and experiment view.

The overall view of the control system, as shown in figures 5.10 and 5.11, shows the whole room for the preparation phase before taking into detail the control system. In this view, the user can access the main menu which consists of the system selection, the user manual document, game option, and exit the software.

The second view will be activated when user selects the system on the main menu in the previous view. Referring to figures 5.12 and 5.13, this experiment view can fully access the control system components. Users can interact with the controller to tune the system, the monitor to adjust the response inspection, and the plant to add the disturbance into the system or change the parameters of the subsystem.

Figure 5.15 Overall View of the Ball and Beam System

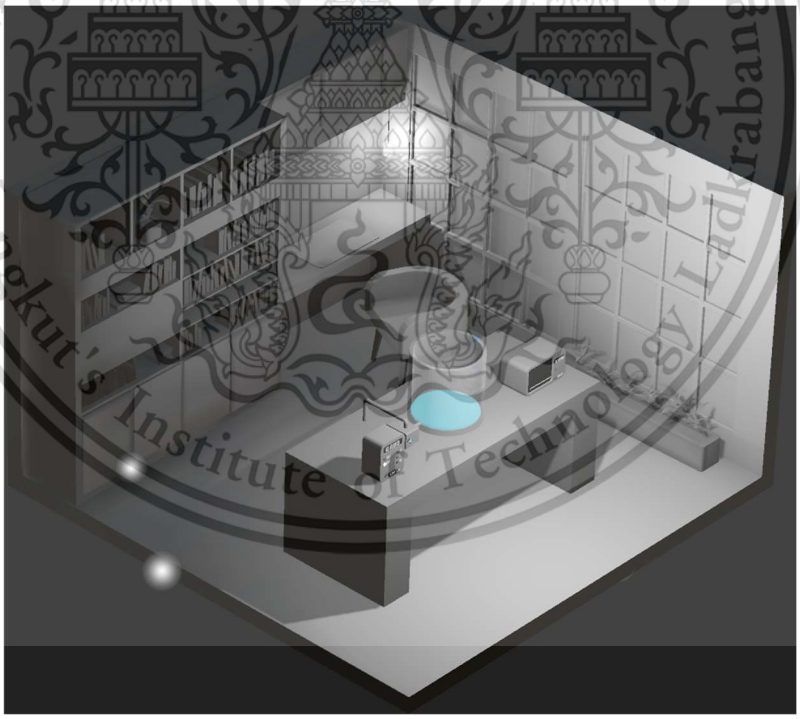

Figure 5.16 Overall View of the Water Level Control System

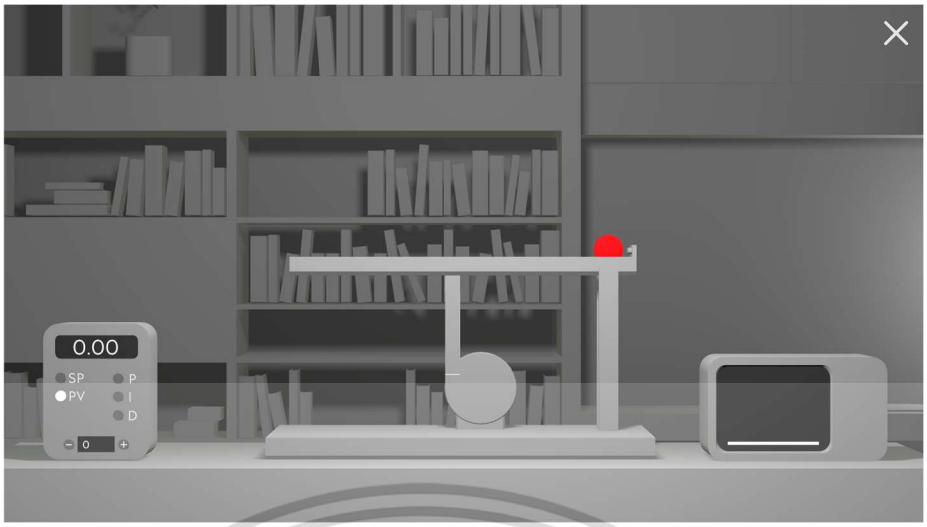

Figure 5.17 Experiment View of the Ball and Beam System

Figure 5.18 Experiment View of the Water Level Control System

5.4 Experiment Modules

5.4.1 Manual Control

Before getting into the automation control, the user needs to know the nature of the system by manual control of the actuator themselves as shown in figure 5.14.

The objective of this module is to control the output of the system into the desired output by mouse and keyboard without the PID controller active.

The completion of this module is when the user can control the process value into the setpoint and be able to hold it for a given time.

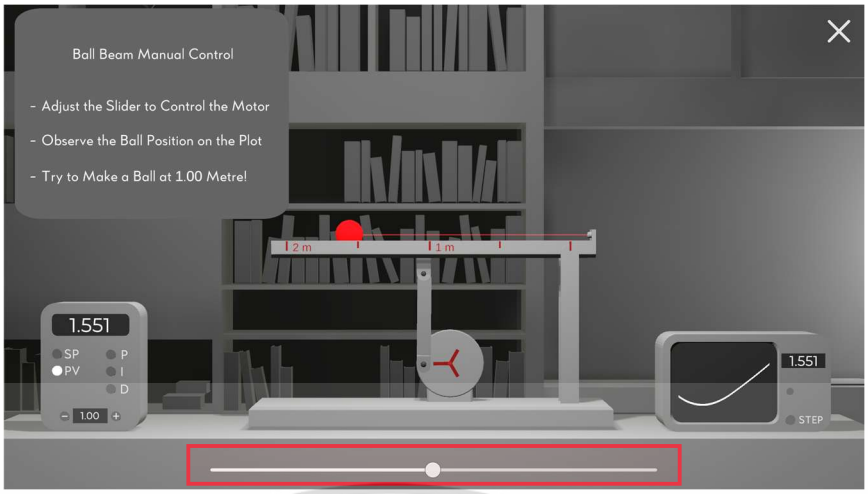

Figure 5.19 Experimental View of the Manual Control Module

5.4.2 PID Controller

After understanding the principle of the simple control by manual, the user will have the opportunity to enable the PID controller and stabilize the system.

The objective of this module is to control and tune the system into the desired response with the assistance of the PID controller as displayed in figure 5.15.

The completion of this module is when the user can stabilize the system which returns the desired output with the given steady-state error, overshoot percentage, and settling time.

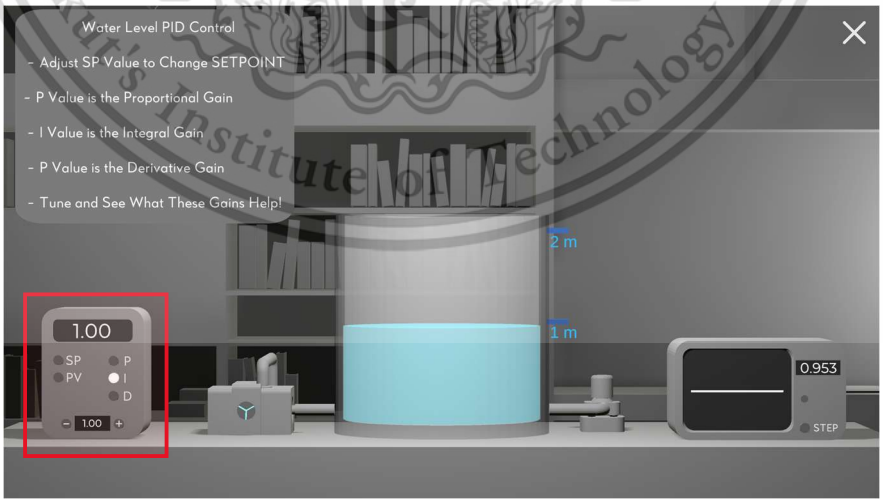

Figure 5.20 Experimental View of the PID Controller Module

5.4.3 Disturbances

After finishing the tuning, the user needs to ensure that the tuned system will be stable even when the system is disturbed by unexpected events.

The objective of this module is to add the disturbance to the system by manual mouse and keyboard to make the system offset from the setpoint as illustrated in figure 5.16.

The completion of this module is when the system response goes to the given offset percentage.

Figure 5.21 Experimental View of the Disturbances Module

5.4.4 System Configurations

The additional module to improve the understanding of the systems.

The objective of this module is to manually adjust the plant or actuator parameters in order to make the system response with the two buttons like the disturbances module in figure 5.16 without re-tune the PID controller.

The completion of this module is when the system output responds differently from the old system parameters.

CHAPTER 6

SYSTEM VERIFICATION

6.1 System Simulation on MATLAB Simulink

This chapter aims to verify the Unity system with the MATLAB Simulink response. The same models are created in two different software; therefore, the more reliable scientific software like MATLAB Simulink is the reference system of the Unity system for the verification.

6.1.1 Ball and Beam System Simulation

The ball and beam system model is created from the block diagram in figure 4.2. The ball and beam system is the high order system, which is hard to stabilize with only one controller, so the cascade of the inner loop and the outer loop control are shown in figures 6.1, 6.2, 6.3, and 6.4.

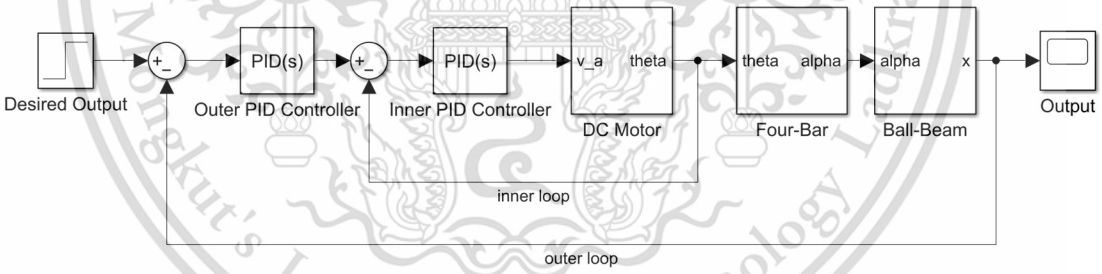

Figure 6.1 Ball and Beam System Model on MATLAB Simulink

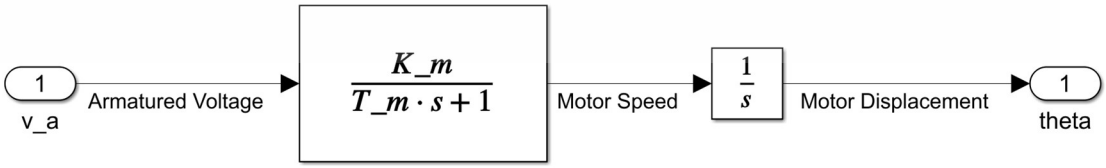

Figure 6.2 DC Motor Transfer Function

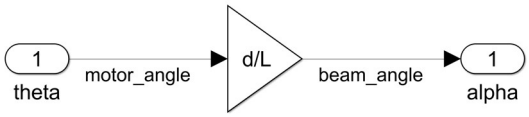

Figure 6.3 Four Bar Mechanism Transfer Function

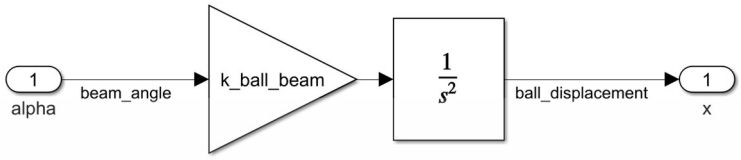

Figure 6.4 Ball and Beam System Plant Transfer Function

After all the blocks are created, the ball and beam system step response with the tuned PID controller is shown in figure 6.5 with all the requirements verified.

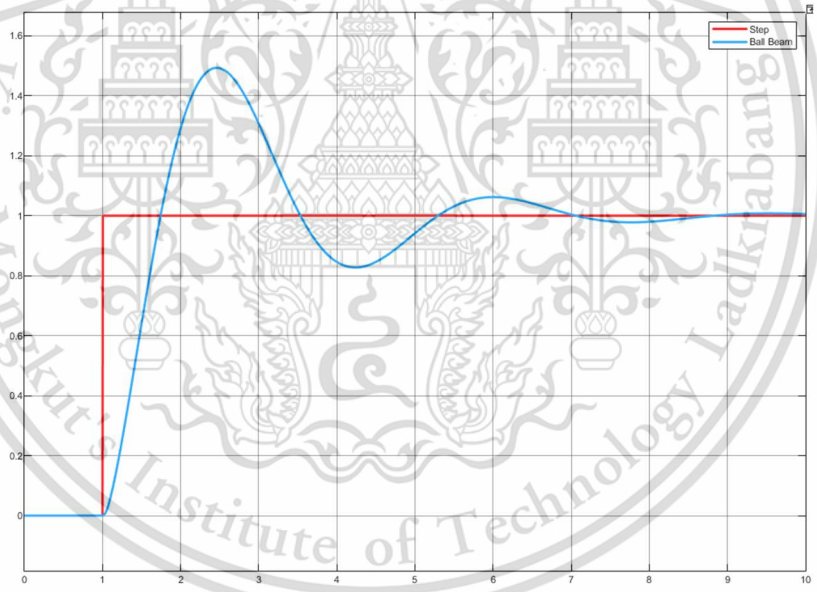

Figure 6.5 Step Response of the Ball and Beam System from MATLAB Simulink

6.1.2 Water-Level System Simulation

The water-level system model is created from the block diagram in figure 4.4. The water level system is a high order system like the ball and beam system, which is too hard to stabilize with only one controller, so the cascade of the inner loop and the outer loop control are shown in figures 6.6, 6.7, and 6.8.

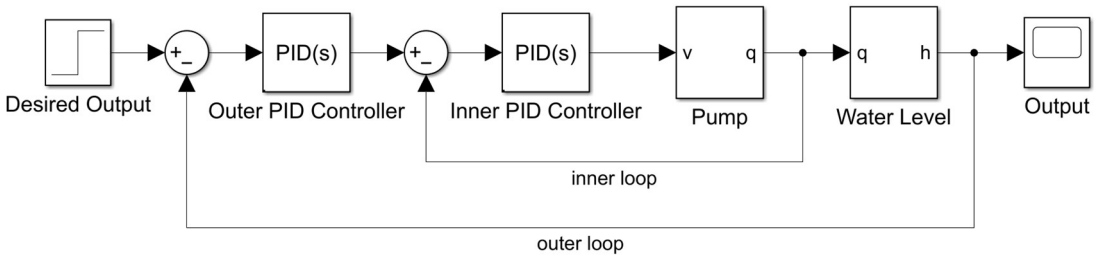

Figure 6.6 Water Level System on MATLAB Simulink

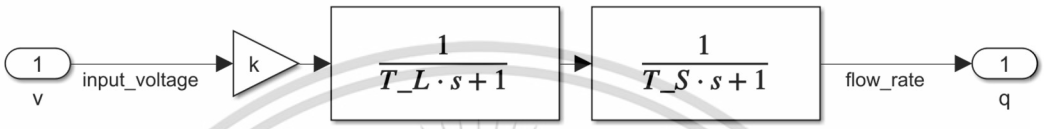

Figure 6.7 Water Pump System Transfer Function

Figure 6.8 Water Level System Plant Transfer Function

After all the blocks are created, the water level system step response with the tuned PID controller is shown in figure 6.9 with all the requirements verified.

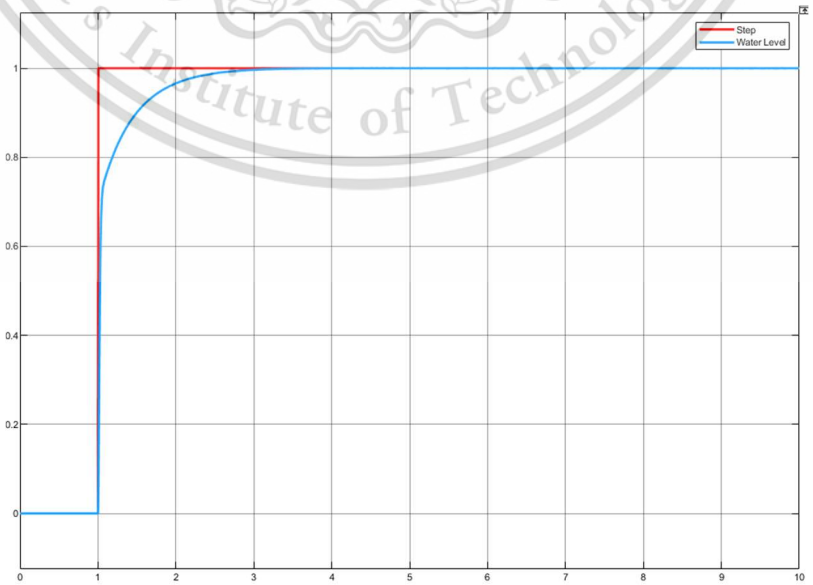

Figure 6.9 Step Response of the Water Level Control System from MATLAB Simulink

6.2 Verification of the Unity System with MATLAB Simulink

Ball and beam system verification has 4 parameter sets to ensure that the accuracy between MATLAB Simulink system and the Unity game engine is satisfy comparing to theoretical value as shown in table 6.1 and figure 6.10 to 6.13, the most significant maximum error is 0.0621 m from the system 2 is properly in term of does not miss the characteristic of ball and beam system.

Table 6.1 System Parameters of Ball and Beam System

System	Mass of the Ball (kg)	Radius of the Ball (m)	Maximum Error (m)
1	0.500	0.100	0.0534
2	0.500	0.200	0.0621
3	0.400	0.100	0.0463
4	0.600	0.200	0.0202

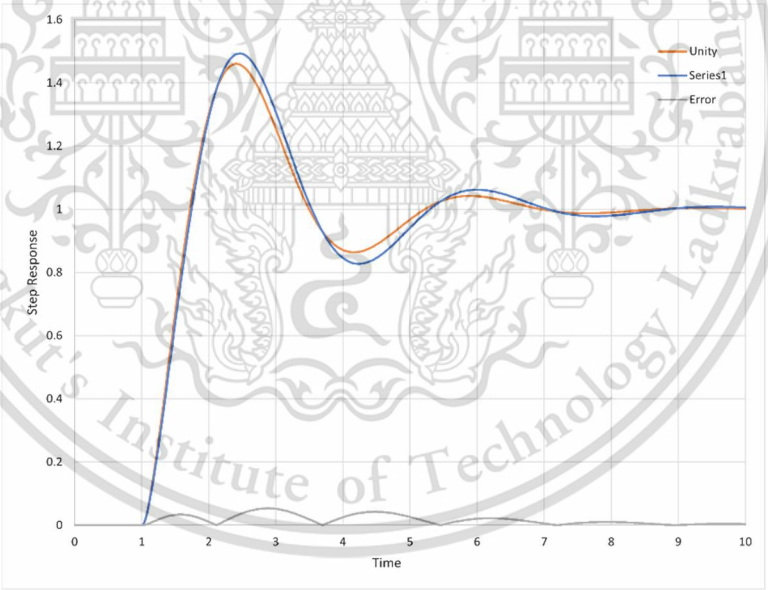

Figure 6.10 Step Response of Ball and Beam System 1 on Simulink and Unity

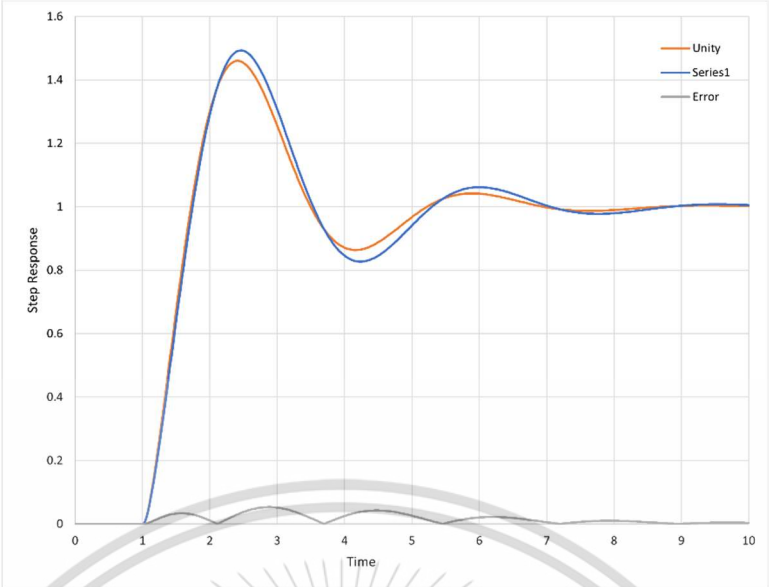

Figure 6.11 Step Response of Ball and Beam System 2 on Simulink and Unity

Figure 6.12 Step Response of Ball and Beam System 3 on Simulink and Unity

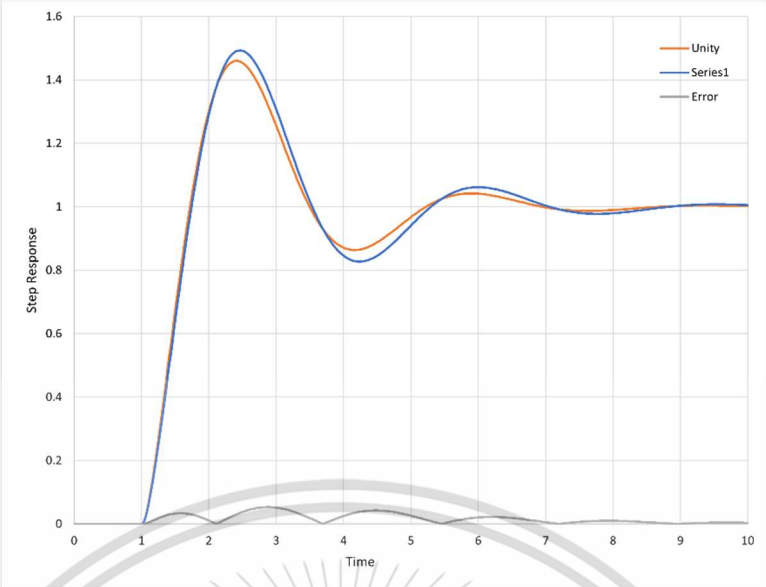

Figure 6.13 Step Response of Ball and Beam System 4 on Simulink and Unity

Water level system verification has 4 parameter sets to ensure that the accuracy between MATLAB Simulink system and the Unity game engine is satisfy comparing to theoretical value as shown in table 6.2 and figure 6.14 to 6.17, the most significant maximum error is 0.0249 m from the system 4 is properly in term of does not miss the characteristic of the water level system.

Table 6.2 System Parameters of Water-Level Control System

System	Tank Capacitance (m^2)	Flow Resistance (s/m^2)	Maximum Error (m)
1	1.00	500	0.0105
2	1.00	1000	0.0188
3	0.800	500	0.0127
4	1.20	1000	0.0249

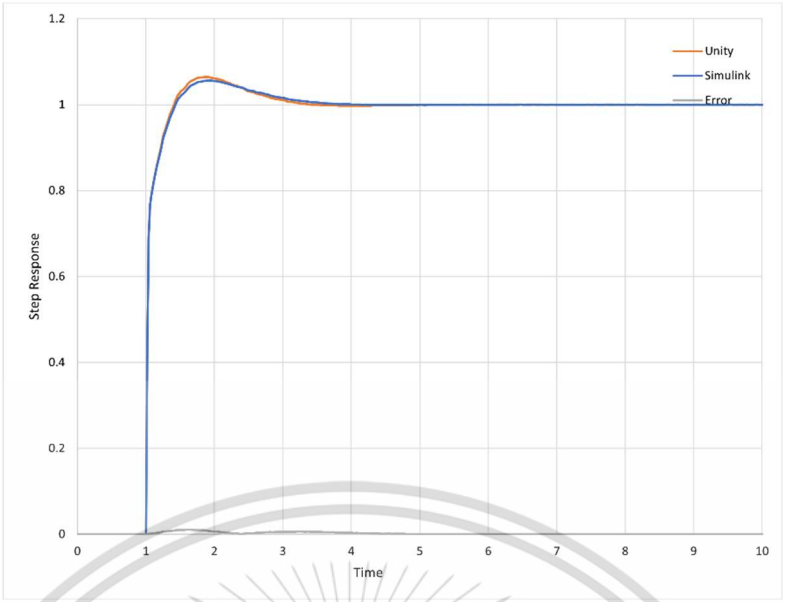

Figure 6.14 Step Response of Water Level System 1 on Simulink and Unity

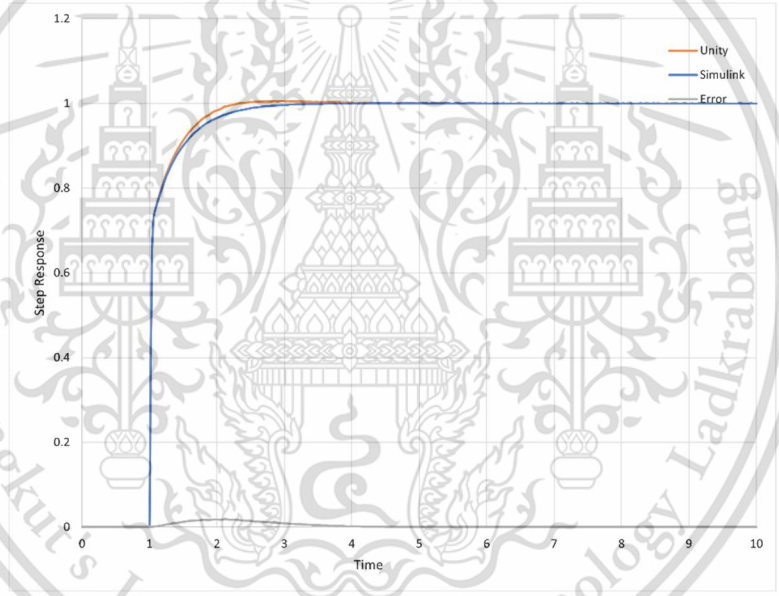

Figure 6.15 Step Response of Water Level System 2 on Simulink and Unity

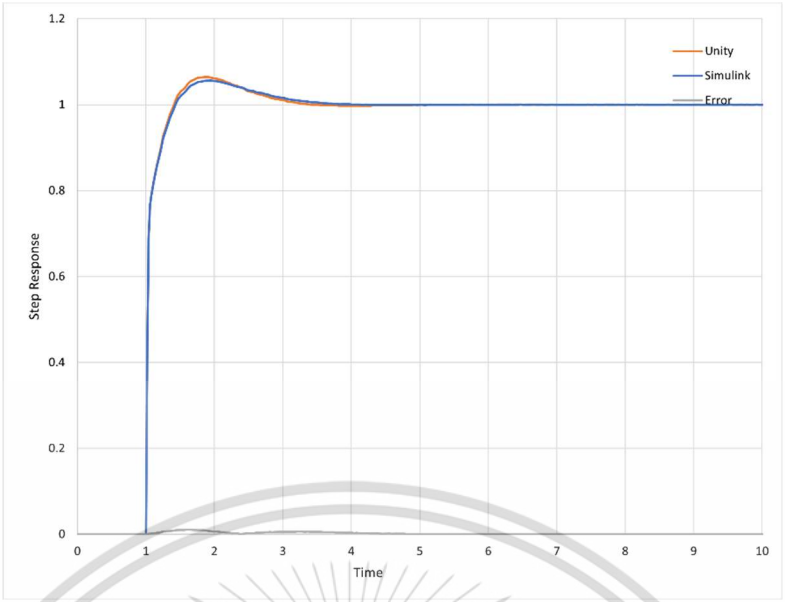

Figure 6.16 Step Response of Water Level System 3 on Simulink and Unity

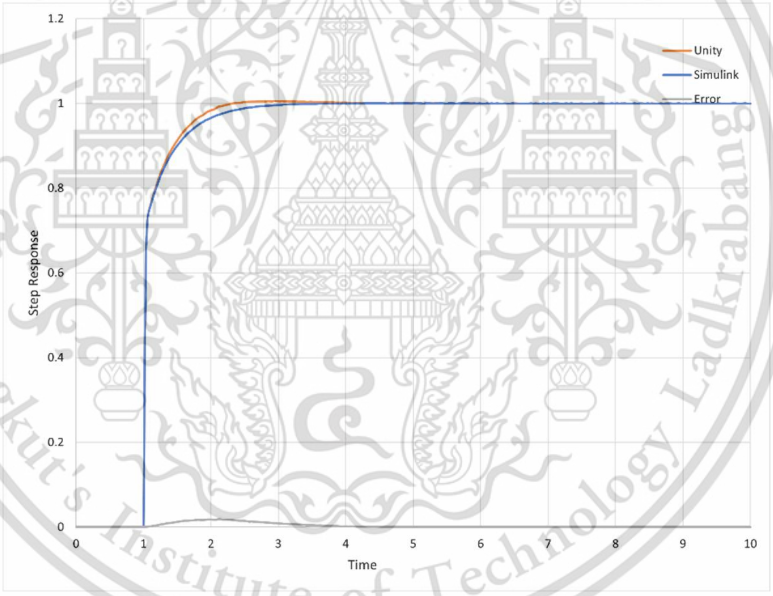

Figure 6.17 Step Response of Water Level System 4 on Simulink and Unity

CHAPTER 7

CONCLUSION

7.1 Results Discussion

This experimental setup is completely developed and ready for users to use and learn and contains the ball beam system and the water level system. The evaluation is divided into four modules, the manual control to learn the nature of the system, the PID controller to learn the tuning of the feedback controller, the disturbances to be ready for the unexpected events, and the system configurations to fulfill the understanding in the system plant.

The user can interact with the control system with their mouse and keyboard with the virtual experience of PID tuning with the real-time response monitor. This result satisfies the thesis objective of the enjoyment of the experiment.

The accuracy of the Unity system is verified by the MATLAB Simulink, which is reliable scientific software with low and insignificant error.

7.2 Future Works

This thesis is working with only one field of study which is control systems. There are many subjects and industrial fields that can do the interactive virtual experience like this to improve the learning system.

The more realistic the system, the more experience the people will get. Not just the graphical visualization, the input system like virtual-reality and mixed-reality are the key to improving the user experience.

From both paragraphs above, the future works can be both increasing the system in the game and improving the user experience system.

REFERENCES

Jacob Apkarian, Michel Lévis, and Hakan Gurocak. (2011). Ball and Beam Experiment for MATLAB®/Simulink® Users. Ontario: *Quanser, Inc.*

Ashitava Ghosal. (2010). The Freudenstein Equation Design of Four-Link Mechanism, Vol. 15, pp. 699-710. *Resonance*

Katsuhiko Ogata. (2004). System Dynamics. 4th Edition. Minnesota: *Pearson Prentice Hall ®*

Justin Youney. (2007). A Comparison and Evaluation of Common PID Tuning Methods. Florida: *University of Central Florida*

William E. Boyce, Richard C. Diprima, and Douglas B. Meade. (2017). Elementary Differential Equations and Boundary Value Problems. 11th Edition. *John Wiley & Sons, Inc.*

Yuqin Wang, Haodong Zhang, Zhibo Han, and Xiaoqiang Ni. (2021). Optimization Design of Centrifugal Pump Flow Control System Based on Adaptive Control. China: *Chaohu University*