HIL EXPERIMENT USING PLC AND MATLAB OPC TOOLBOX

A Graduate Project submitted in Partial fulfillment of the requirements

For the degree of Master of Science in

Electrical Engineering

By

Maxim Aloomian

December 2013
The graduate project of Maxim Aloomian is approved:

________________________________________  __________________________________
Dr. Ali Amini  Date:

________________________________________  __________________________________
Dr. Shahnam Mirzaei  Date:

________________________________________  __________________________________
Dr. Kourosh Sedghi Sigarchi, Chair.  Date:

California State University, Northridge
# Table of Contents

Signature Page ................................................................. ii

Table of Contents .............................................................. iii

List of Figures ................................................................. v

List of Tables ................................................................. vii

Abbreviations ................................................................. viii

Abstract ................................................................. x

1 Introduction ......................................................................................... 1

1.1 Overview ......................................................................................... 1

1.2 Objective ......................................................................................... 1

2 Technical Background ........................................................................ 3

2.1 PLC (Programmable Logic Controller) ........................................... 3

2.2 Allen Bradley CompatLogix PLC ................................................... 4

2.3 RSLogix 5000 ................................................................................... 5

2.4 Factory talk view studio ................................................................. 6

2.5 RSLinx Classic ............................................................................... 7

2.6 OPC ......................................................................................... 8

3 Approach ......................................................................................... 11
3.1 Modeling in MATLAB Simulink ............................................................11

3.2 OPC setup in MATLAB and PLC ...........................................................17

3.2.1 OPC Read and OPC write Block .......................................................20

3.3 PLC program and HMI screens ...............................................................24

3.3.1 PID Loops ..........................................................................................27

3.3.2 Automatic/Manual mode selection (H-O-A) .....................................31

3.3.3 Settings screen....................................................................................35

4 Case Study ...................................................................................................... 38

4.1 Water tanks Equations .............................................................................38

4.2 Verification of control system through eigenvalues .................................41

5 Result and Conclusion .................................................................................... 44

6 References ...................................................................................................... 45

7 Appendix: Ladder Logic................................................................................. 46
List of Figures

Figure 2.1 An example screen shot of Ladder Diagram language .................... 4
Figure 2.2 1769-L16ER-BB1B CompactLogix controller used in this project ...... 5
Figure 2.3 Factory Talk view Studio Software .............................................. 7
Figure 2.4 RSLinx Classic environment ......................................................... 8
Figure 2.5 Advantages of OPC in driver development ..................................... 10
Figure 3.1 Model design of three cascaded tank in Simulink ............................ 13
Figure 3.2 Tank 1 model design in Simulink ................................................. 13
Figure 3.3 Tank 1 PID loop block diagram .................................................... 14
Figure 3.4 Tank 2 model design in Simulink ................................................... 14
Figure 3.5 Tank 2 PID loop block diagram .................................................... 15
Figure 3.6 Tank 3 model designs in Simulink ............................................... 15
Figure 3.7 Tank 3 PID loop block diagram .................................................... 16
Figure 3.8 Overview of the system ............................................................... 17
Figure 3.9 Opening ‘OPC configuration’ screen in RSLinx ............................... 18
Figure 3.10 Configuring OPC Topic on ‘OPC/DDE Topic Configuration’ ........ 19
Figure 3.11 OPC configuration block setup .................................................. 20
Figure 3.12 RSLinx OPC Server selection from local host ................................ 20
Figure 3.13 OPC Read Block configuration .................................................... 21
Figure 3.14 Transferring data to subsystem through OPC Read ...................... 22
Figure 3.15 OPC Write Block configuration .................................................. 23
Figure 3.16 Transferring data from subsystem to the PLC through OPC Write .. 23
Figure 3.17 Overview of system in MATLAB Simulink ................................. 24
Figure 3.18 An overview of tags used in this thesis PLC code ....................... 26
Figure 3.19 PID block used for controlling ‘valve 2’ ...................................... 27
Figure 3.20 Scaling PID loop variable for using in HMI and MATLAB .......... 29
Figure 3.21 Tank 1 level tag used in OPC write block in MATLAB ............... 30
Figure 3.22 Tank 1 level tag used in numeric display object HMI .................. 30
Figure 3.23 HOA screen on HMI ................................................................. 31
Figure 3.24 PLC code for ‘Valve 1’ HOA mode ........................................... 32
Figure 3.25 PLC code for opening and indication of ‘Valve 1’ ....................... 33
Figure 3.26 PLC code for changing the status of ‘Valve 1’ in ‘Hand’ mode ..... 34
Figure 3.27 ‘Valve 1’ position selection in ‘Hand’ mode ............................... 34
Figure 3.28 Settings screen on HMI ............................................................ 35
Figure 3.29 Numeric input cursor point in settings screen with assigned tag .. 36
Figure 3.30 Code for reading PID loop 1 set-point value from HMI .......... 37
Figure 4.1 Tank with all parameters ........................................................... 38
List of Tables

Table 4.1 Parameters and Constants description, range and value……………..40

Table 4.2 Sample points of system…………………………………………….40
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Analog Input</td>
<td>4</td>
</tr>
<tr>
<td>AO</td>
<td>Analog Output</td>
<td>4</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
<td>3</td>
</tr>
<tr>
<td>DA</td>
<td>Data Access</td>
<td>17</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
<td>4</td>
</tr>
<tr>
<td>DDE</td>
<td>Dynamic Data Exchange</td>
<td>18</td>
</tr>
<tr>
<td>DI</td>
<td>Digital Input</td>
<td>3</td>
</tr>
<tr>
<td>DO</td>
<td>Digital Output</td>
<td>4</td>
</tr>
<tr>
<td>HDA</td>
<td>Historical Data Access</td>
<td>17</td>
</tr>
<tr>
<td>HIL</td>
<td>Hardware In the Loop</td>
<td>x</td>
</tr>
<tr>
<td>HMI</td>
<td></td>
<td>viii</td>
</tr>
</tbody>
</table>
Human Machine Interface ................................................................. 6

HOA
Hand Off Auto ............................................................................... 30

I/O
Input/Output .................................................................................. 3

ID
Item Definition ............................................................................... 21

LD
Ladder Diagram ........................................................................... 4

OLE
Object Linking and Embedding .................................................... 8

OPC
OLE for Process Control ............................................................. x

PID
Proportional Integral Derivative .................................................. 11

PLC
Programmable Logic Controller ................................................ x

RAM
Random Access Memory .............................................................. 3

ROM
Read Only Memory ....................................................................... 3
Abstract

HIL EXPERIMENT USING PLC AND MATLAB OPC TOOLBOX

Maxim Aloomian

Master of Science in Electrical Engineering

The objective of this thesis is to perform a HIL simulation with PLC and MATLAB Simulink and to accomplish real time communication between them through OPC. PLC is being widely used in control systems for industrial automation, and testing and verifying written code for every control system is an essential and unavoidable task. Since testing on physical plant is time consuming and risky in case of damaging equipments, so simulation and HIL is very useful and desirable. In this method, a real plant is being replaced by a model in software, and interacts with the real control system. Every plant or system consists of inputs and outputs which in a control system is represented with various data values and type, so it’s possible to model almost every system with mathematical equations in MATLAB. The main barrier in this method is to connect these two separate systems which are implemented in two different environments. The solution for this problem requires recruiting a translator to exchange data between these two objects. OPC (OLE for Process Control) has been created for this purpose, it’s a software based technology that creates a database, and exchange data
between different objects from different vendors and integrators. Every PLC in the market has OPC server and this makes possible for researchers and students to accomplish real time communication between PLC and MATLAB. One of the goals of this project is to create a structure for students to use in future experiments and researches to implement simulation.
1 Introduction

This chapter represents the basic data for understanding content of this thesis. It includes an overview of the theory used in thesis, detailed problem description, and the reasons for conducting this project.

1.1 Overview

Using automated workstations in modern industrial production lines is considered an important element. Its advantages include higher production rate, lower labor cost, improved product quality, and increase the efficiency and safety of the system. These systems consist of controlling part that needs to be designed and modify with respect to production line and their specifications. So it’s essential to design control systems for these workstations, and to test them before starting their real production line. Testing and verification of the control program in the early stages of implementing a production line before real commissioning is an important element. This test can be less expensive and free of any damage to equipment if it’s done in a virtual environment with modeling and simulation.

1.2 Objective

The main objective is to accomplish an HIL (Hardware In the Loop) simulation using a real time communication through an OPC standard. Students and researchers can directly create complex systems and related control system on MATLAB or any simulation software. This approach has two problems; one is about the proximity of a MATLAB based control system to the real world, and the other one about difficulty of
implementing of that control system in real world. Recently, PLCs are mostly used to control many industrial applications, and that’s the reason I decided to use PLC as a control system to reflect the real control hardware in this paper. In my research about HIL, I found out that this approach has been done with PLC with a main different that real plant has been modeled in software other than MATLAB and which limited software and not capable of extending to every application [7]. One weak point for mentioning approach was about communication which wasn’t an easy task to establish, and it’s not a stable, secure and reliable connection. The advantage of using MATLAB is that every researcher and student has access to this software and it’s capable to create almost every complex system. One of the other advantages of using MATLAB is about using the OPC Toolbox for real time communication between PLC and MATLAB which is a reliable and secure communication. This paper can be used as a sample for future students who want to control their model designed on MATLAB with PLC, with an established real time communication through OPC technology.
2 Technical Background

The following pages will provide information about PLC hardware and software. The type and brand of PLC and required software for programming this hardware is being described. OPC which plays a key role in this project for connecting control system and plant model to each other will be explained.

2.1 PLC (Programmable Logic Controller)

PLC (Programmable Logic Controller) is a computer which is designed for control solution purpose in the industry. It’s widely used in any application in industry which needs a sort of automation like machine controls, food processing, factory assembly lines, power plants, waste water treatments, and etc. Basically it replaces the relay circuit design control panel to control equipment and machines. The main difference between PLCs and other types of computers and microprocessor is about their capability to work in harsh environments in different applications. They are highly resistant to various noises such as sound, vibration, dirt, dust and high temperature and etc., and that’s why makes them so favorable for being used widely in industry.

PLC hardware typically consists of two main parts, base part and I/O (Input/output) modules. Base part consists of a CPU (Central Processing Unit), RAM (Random Access Memory) and ROM (Read Only Memory), communication ports, power supply, and sometimes embedded I/O (Input/output) ports. In order to communicate and monitor process variables and actuate external systems it’s required to have input/output (I/O) ports which are designed in I/O modules. I/O modules are four types DI (Digital
Ladder Diagram (LD) is a popular language for writing PLC code, because of its similarity to relay logic circuits and simple graphical features. In this paper the program for PLC is written in Ladder Diagram (LD) language. There is an example in Figure 2.1.

![Ladder Diagram language example](image)

**Figure 2.1** An example screen shot of Ladder Diagram language

### 2.2 Allen Bradley CompatLogix PLC

Allen Bradley is the brand name of a line of automation equipment manufacturer Rockwell Automation. Allen Bradley PLC is most used and famous PLC brand that is used in North America. It has several generations of controllers like PLC5, MicroLogix, CompactLogix, ControlLogix and etc. In this paper CompactLogix 5370 L1 series controller with embedded point I/O will be used. It has 16 DC inputs and 16 DC outputs, USB port for firmware download and programming, dual Ethernet/IP ports for ring topologies, and a built in 24VDC power supply [6].
2.3 **RSLogix 5000**

RSLogix 5000 is software to configure and program Logix5000 controller like CompactLogix and ControlLogix. It’s used to create and manage projects for PLC and stores project’s Logic, configuration, data and documentation. RSLogix 5000 is capable of creating and editing projects in ladder logic function block diagram, structured text and sequential function chart. With RSLogix 5000 you can have an access to real-time information, and makes it easier to create and develop a program for any complex logic and application [6].

Figure 2.2 1769-L16ER-BB1B CompactLogix controller used in this project
2.4 Factory talk view studio

Factory talk view studio is software for creating and developing applications for HMI (Human Machine Interface). HMI is a device that provides a graphic-based visualization of an industrial control and monitoring system. It provides communication with machines and control system and informs user from states of the control system and machinery. HMI by providing real time data acquisitions, alerts and alarms, commands and other tools, connects the user with the process being controlled process in order to give him or her to react appropriately [7].

Factory Talk view Studio allows users to create screens that represent plant floor and the process is being controlled. It has a complete library for presenting control instrument and equipment on HMI and assign tags to be aware of every state of the process. By adding virtual pushbuttons, selector switches, numeric input displays, gauge bars, navigation switches and etc. in this software, users will be able to view the status of the process, send commands to system, and receive feedbacks and alarms to monitor control system. Figure 2.3 displays environment of Factory Talk View Studio software.
2.5 RSLinx Classic

RSLinx Classic is a comprehensive communications solution for all Rockwell Automation networks and devices. RSLinx Classic allows browsing and viewing of devices across multiple Rockwell Automation networks [5]. With RSLinx Classic you can view all your active networks through a single window and can run any combination of supported applications simultaneously, through same or different communication interfaces. RSLinx Classic is a complete communication server that makes possible to find available Allen Bradley devices in your network, and manages communication and exchanging data between PLC and HMI (Human Machine Interface) and other Rockwell Automation devices on network [7].

RSLinx have capabilities to support OPC connectivity to maximize the
interoperability between clients and servers. It’s an OPC compliant server that enables data interchange between HMIs and other OPC Clients and Allen Bradley PLCs. This feature enables us to provide communication between PLC and MATLAB [5]. Figure 2.4 displays environment of RSLinx Classic with its various drivers for detecting Allen Bradley devices on different networks.

![Figure 2.4 RSLinx Classic environment](image)

2.6 OPC

OLE (Object Linking and Embedding) for Process Control (OPC) is a software interface standard used to ease data exchange between different vendors of process control equipment vendors.
In the past few years every manufacturer of industrial automation device had to develop a specific communication software or driver to connect to each industrial device they wished to interface. For example, each vendor of HMI had to write many specific codes for communicating to every available brand of industrial devices (PLC, Distributed Control Systems, etc.) [1]. This is a problem same as what every software developer company faced to write a printer driver for every printer type in the market, like AutoCAD that had to write a driver for each brand like Epson, Canon, HP, and on and on. Windows solved the printer driver problem by incorporating printer support into the operating system [1].

For solving this problem and reducing the amount of time and effort OPC has been created. In this case industrial device providers don’t need to write a driver for every other vendor devices and every one of them only needs to develop an OPC client and/or server for their product [5]. Figure 2.5 shows the advantage of OPC to reduce the number of required drivers. This OPC client would then communicate with other OPC servers designed and sold by the manufacturers of the other networks and controllers. In this project RSLinx is as an OPC server and MATLAB OPC toolbox is an OPC client.
Figure 2.5 Advantages of OPC in driver development
3 Approach

This chapter describes the method for implementing a HIL simulation for specified plant with the help of the tools described in previous pages. The model design in MATLAB Simulink and associated control system with PLC will be presented to make a complete HIL simulation. In Flowchart 3.1, the order of tasks for completing this HIL simulation is displayed.

3.1 Modeling in MATLAB Simulink

Model-Based design is a way to transfer engineering work from the field to the desktop. This approach increases the quality of products and reduces labor hours [11]. Here, I used the model for a process which is three cascaded tanks, and I connected this model to PLC and read from and write to PLC values associated with this process. In following pictures and notes, it will be displayed in detail.

In Figure 3.1 on the next page the model of three cascaded tanks is shown in MATLAB. The input of the first tank is a constant flow which is adjusted by a manual valve, and the output of each part or tank in input of next one. Figure 3.2 through Figure 3.7 show individual tanks with the related Simulink model, and PID loop block diagram of each.

In this model every tank has a separate set-point, and control variable of every tank is process variable of the other one. Hence, we designed every tank with one separate PID loop in the PLC, the control loop block diagram for every tank is shown after every tank model design in the following pages.
Flowchart 3.1 An order of tasks should be done to complete this HIL simulation
Figure 3.1 Model design of three cascaded tank in Simulink

Figure 3.2 Tank 1 model design in Simulink
Figure 3.2 designed in Simulink shows tank1, the input of this system is ‘In Flow’ and output or process variable is ‘h1’ and PID loop changes control variable ‘valve_out_1’ to get to desired ‘h1’. These two parameters ‘valve_out_1’ and ‘h1’ determine the output flow of system ‘flow2’ which is an input flow of second system shown in Figure 3.4.

Figure 3.3 shows a block diagram of control system which is written in PLC and for better demonstration is shown in this way. ‘valve_out_1’ is control variable which we talked about in the previous paragraph, and ‘h1’ is processed variable.

Figure 3.3 Tank 1 PID loop block diagram

Figure 3.4 Tank 2 model design in Simulink
Figure 3.4 displays a model of tank 2 in Simulink, the input of this system is ‘flow2’ and output or process variable is ‘h2’ and PID loop changes control variable ‘valve_out_2’ to get to desired ‘h2’. These two parameters ‘valve_out_2’ and ‘h2’ determine the output flow of system ‘flow3’ which is input flow of third system shown in Figure 3.4.

Figure 3.5 shows a block diagram of control system which is written in PLC and for better demonstration is shown in this way. ‘valve_out_2’ is control variable which we talked about in the previous paragraph, and ‘h2’ is processed variable.

![Figure 3.5 Tank 2 PID loop block diagram](image)

![Figure 3.6 Tank 3 model designs in Simulink](image)
Figure 3.6 displays a model of tank 3 in Simulink, the input of this system is ‘flow3’ and output or process variable is ‘h3’ and PID loop changes control variable ‘valve_out_3’ to get to desired ‘h3’. These two parameters ‘valve_out_3’ and ‘h3’ determine the output flow of system ‘flow3’ which is an input flow of third system shown in Figure 3.7.

Figure 3.7 shows a block diagram of control system which is written in PLC and for better demonstration is shown in this way. ‘valve_out_3’ is control variable which we talked about in the previous paragraph, and ‘h3’ is processed variable.

Figure 3.7 Tank 3 PID loop block diagram

In Figure 3.8, better demonstration of system is presented. This picture is obtained of HMI of the system developed in the Factory talk view studio. In this picture ‘VALVE2’, ‘VALVE3’ and ‘VALVE4’ are control variables for each tank that were described earlier in previous pages.
3.2 OPC setup in MATLAB and PLC

OPC Toolbox™ provides a connection to OPC DA and OPC HDA servers, giving you access to live and historical OPC data directly from MATLAB® and Simulink®. You can read, write, and log OPC data from devices, such as distributed control systems, supervisory control and data acquisition systems, and PLCs (Programmable Logic Controller) that conform to the OPC Foundation Data Access (DA) standard [4].
In this part of the paper, we will set up OPC communication between PLC and MATLAB.

- In a first step OPC should be configured on PLC, this step should be done in RSLinx Classic. RSLinx continuously browsing network to find present Allen Bradley devices, after finding PLC, we should right click on select ‘Configure New DDE/OPC Topic’ Figure 3.9 illustrates this step.

![Figure 3.9: Opening 'OPC configuration' screen in RSLinx](image)

- Then on ‘OPC/DDE Topic Configuration’ screen, a topic should select for PLC and by highlighting PLC and topic, should press apply and our setting for PLC will be complete in this step.
Figure 3.10 Configuring OPC Topic on ‘OPC/DDE Topic Configuration’

- Then PLC and PC should be on the same network, for this purpose IP address of both should be under the same subnet mask. It means that first 3 parts of the IP address of each should be same for example: 192.168.50.XXX

- Then OPC should set up on MATLAB, for this purpose OPC configuration block should be added to the model from the OPC toolbox in a Simulink model. Then select RSLinx OPC server from the local host. Figure 3.11 and Figure 3.12 illustrate these steps.
3.2.1  **OPC Read and OPC write Block**

OPC toolbox in Simulink has two other blocks, one is OPC Read and the other one is OPC Write. OPC Read is for reading variables’ values from PLC and transfer to the
model. Every variable which is read or written by OPC is called ‘Item’. To access to every item in OPC, item ID should be entered, the item ID is in this format “[OPC server name]variable name”. For example for reading a variable named “analog_Tank1[12]” from “opc_program_1” item ID would be entered as follows: “[opc_program_1] analog_Tank1[12]”. Figure 3.13 illustrates this part.

![OPC Read Block configuration](image.png)

Figure 3.13 OPC Read Block configuration

In Figure 3.14 output variables from PLC are transferred to the OPC Read block in Simulink through OPC Read block, these variables are valve control values which are
controlling the amount of water to run out of each tank, and consequently control level in each tank.

Figure 3.14 Transferring data to subsystem through OPC Read

The OPC Write block is used for transferring values from Simulink to PLC, through this block PLC is having real time communication with the model in Simulink and gets updated in every scan of the model. For setting up this block, it requires to add items ID as is described in earlier paragraphs. Figure 3.15 demonstrates configuring this block.

In this system output value of Simulink model, indicates the status and condition of our plant and that’s like values read from sensors and instruments through PLC. Since there is no real sensor in this model, these values show status of the system and PLC needs those to control the system. Hence, we connect them from subsystem to OPC write block to send to the PLC.
Figure 3.15 OPC Write Block configuration

Figure 3.16 Transferring data from subsystem to the PLC through OPC Write
After establishing communication between PLC and MATLAB Simulink, PLC and HMI program should be developed.

![Figure 3.17 Overview of system in MATLAB Simulink](image)

3.3 **PLC program and HMI screens**

In this part PLC program structure will be reviewed. Some important parts that have been developed in this program code will be discussed, and the reason for having them here will be presented. Flowchart 3.2 displays steps for creating PLC and HMI application.
Create variables or tags list and their corresponding data type in RSLogix5000.

Create various subroutines based on control system logic in RSLogix5000. These subroutines include Reading and writing inputs and outputs, PID loops, HOA, process sequences, and etc.

Create subroutines for reading and writing tags used in HMI in RSLogix5000.

Develop HMI screens in FactoryTalk View Studio.

Create HMI tags for objects used in HMI. Basically these tags are created by importing tags from PLC program.

Assign HMI tags to objects used in screens, these objects include switches, multistate indicators, numeric displays, numeric input displays, valves, pumps, and etc.

Create a connection between HMI and PLC through RSLinx. This connection enables HMI to read value from PLC and change them. It has been used same tags name in both PLC and HMI.

Download written code in RSLogix 5000 to PLC, and run PLC. Monitor logic and values online, and create runtime file of HMI and run it on PC.

Flowchart 3.2 Required steps to create PLC and HMI application
First thing to start writing a code is about creating variables or tags, and their data type. Figure 3.18 represents an overview of tags used in this thesis, these tags consist of tags used in PLC, and shared tags used in HMI and MATLAB. HMI uses tags that are shared between PLC and HMI. HMI tags are same as PLC tags; with only this difference that there should be a shortcut name of PLC precedes every tag. This shortcut name is assigned to PLC in RSLinx when connection between PLC and HMI is being set. For example [clgx] precedes ‘analog_Tank1’ in Figure 3.21. So, every HMI tag used in this thesis starts with [clgx]. These shared tags enable PLC and HMI to exchange data and values through RSLinx. MATLAB also uses same routine to talk to PLC, with only this difference that MATLAB uses OPC toolbox to transfer data to PLC.

![Figure 3.18 An overview of tags used in this thesis PLC code](image-url)
Ladder Logic code used in this project has three main components, which are described in detail in following pages. Development of PID loops, HOA mode selection and setting screen, and their application will be discussed in next pages.

### 3.3.1 PID Loops

Analog output for PLC is used to control proportional valve. Three PID loop is written to control output flow water from each tank, and in result by using proportional valves the level of tank will be controlled. It’s assumed that proportional valves working based on a variable 4 to 20mA current. An analog current 4mA sent to valve means that the valve is fully closed, and it starts changing position from closed to %100 open, by increasing analog current to 20mA.

![Image of PID block used for controlling ‘valve 2’](image)

**Figure 3.19 PID block used for controlling ‘valve 2’**
For performing an analog control for this process, we use PID blocks that are available in RSLogix5000 library. It requires calculating PID gains and applying them to get a robust control for the process. Figure 3.19 shows the PID block used for ‘valve 2’ to control position of ‘valve 2’ based on level in tank 1. In this block there is an option named ‘Process variable’ which is input to PID block and that’s the level in the tank 2. PLC reads this input from MATLAB, and it’s a real type data. Another option used in PID block is ‘Control variable’ which is output of PID block and contains the position of ‘valve 2’. These two option reads and write values through ‘analog_tank1[20]’ and ‘analog_tank1[22]’. These two tags value should be in a specific range, for being used in PID block. For this purpose, these values should be scaled to specific range values to be readable and usable in MATLAB and HMI. Figure 3.20 displays ladder logic code to scale their values to proper ranges in order to be used in HMI and MATLAB. As it’s shown in this picture, tags that are being used in HMI and MATLAB are ‘analog_tank1[10]’ and ‘analog_tank1[12]’. These steps should be repeated same for other tanks level control and their PID blocks.
As you see in Figure 3.21 and Figure 3.22, these tags have been used in MATLAB and HMI. Figure 3.22 shows how tank 1 level tag ‘analog_tank1[10]’ is being assigned to numeric display object in HMI. Figure 3.21 also displays the same tag is being used in MATLAB as level of tank 1. This tag’s value is created in designed model in MATLAB, and is written to PLC through ‘OPC write block’ and transferred to HMI through RSLinx connection with HMI.
Figure 3.21 Tank 1 level tag used in OPC write block in MATLAB

Figure 3.22 Tank 1 level tag used in numeric display object HMI
3.3.2 Automatic/Manual mode selection (H-O-A)

Every equipment or device which is controlled and actuated in a process should have an automatic/manual selection option. This is necessary for maintenance purpose to exit a machine from production line because of its failure or maintenance, and run it in manual mode regardless of the status of the process. Since this paper should reflect a real control system in the industry, it’s essential to add this feature to control the system. The HOA has three states, Hand (H) mode that enables and disables valves or any device manually. Any device on Off (O) mode will turn off regardless of condition and state of the process. In Auto (A) mode according to the process condition and status, each device and equipment would start or stop operating. In following pages, objects used in HOA screen in Figure 3.23 and their corresponding ladder logic will be illustrated.

Figure 3.23 HOA screen on HMI
The object numbered ‘1’in a circle in Figure 3.23 is a multistate pushbutton object in FactoryTalk View Studio, and it has three states of Hand, Off, Auto. It changes its states and also value of assigned tag ‘HOA_Selection[0]’. On Hand mode, its value is ‘1’, so least significant bit is energized and by checking this bit, we can get that it’s in Hand mode. For Off mode, it gets ‘2’, and for Auto mode, it changes to ‘4’, and by checking bit ‘1’ and ‘2’ of this tag, we’ll get the state. Figure 3.24 illustrates this part of code.

The object numbered ‘2’in a circle in Figure 3.23 is a multistate indicator, and it’s assigned with two states of ‘OPEN’ and ‘CLOSED’. It switches states when value of the tag named ‘general_bits[0]’ changes. Whenever this tag’s value is ‘1’, it turns to ‘OPEN’, and turns to ‘CLOSED’ while is ‘0’. Figure 3.25 illustrates this part of PLC code.
The objects numbered ‘3’ and ‘4’ in circles in Figure 3.23 are grouped, and they are visible when ‘Valve 1’ is in ‘Hand’ mode. Object numbered ‘4’ is a numeric input display, and we can enter our desired position for valve through this object. Tag for this object is ‘HMI_Analog[0]’ and it’s shown on Figure 3.27. Object numbered ‘3’ is a momentary pushbutton, and we can open or close this valve in ‘Hand’ mode by clicking this button. Figure 3.26 displays written code for this purpose, and object ‘3’ is shown with tag named ‘HOA_Internal_Bits[3]’.

Figure 3.25 PLC code for opening and indication of ‘Valve 1’
Figure 3.26 PLC code for changing the status of ‘Valve 1’ in ‘Hand’ mode

Figure 3.27 ‘Valve 1’ position selection in ‘Hand’ mode
3.3.3 Settings screen

In settings screen, operator will be able to enter the desired level value as setpoint for each tank. PID loop gains also can be entered in the PID block through settings screen. In this screen numerical input display objects have been used, and on every object a specific tag or variable name has been assigned. PLC reads these values from HMI on every scan of code through RSLinx Ethernet/IP connection. Control system and PID loop will control valves status and position to reach to the setpoint for every tank.

![Settings screen on HMI](image)

Every PID block needs to know four parameters of Setpoint (SP), Proportional gain (KP), Integral gain (KI) and Derivative gain (KD). User should enter these
parameters from HMI, and PLC reads from HMI. For every parameter, one numeric input
cursor point object has been used; Figure 3.29 shows this object with tag used for it. This
tag is being read by PLC in Figure 3.30 and copied to PID block set-point parameter. The
same procedure is used for entering every parameter from HMI to PLC.

In this chapter, ladder logic code structure has been discussed. We break the code
to various subroutines like HOA, PID loop, and settings. Every one of these parts has
been described, and the method for transferring data from this code to HMI has been
illustrated. Basically every code should be divided to various parts to ease understanding
and debugging of the logic.
Figure 3.30 Code for reading PID loop 1 set-point value from HMI
4 Case Study

4.1 Water tanks Equations

For modeling of a water tank, Bernoulli’s law has been used, the principle of conservation of fundamental quantities mass and energy [2].

Equation (4.1) shows the relation between levels and input and output flow in the tank. ‘\( A \)’ is the cross section area of the tank ‘\( h \)’ is the level of the water in the tank , ‘\( F_{in} \)’ is input flow to the tank and ‘\( F_{out} \)’ is output flow from the tank.

\[
A \frac{dh}{dt} = F_{in} - F_{out} \tag{4.1}
\]
\[ F_{out} = k \times a \sqrt{2 \times g \times h} \] (4.2)

Equation (4.2) shows the relation between output flow and water level in tank. ‘\(a\)’ is the cross section area of orifice, ‘\(g\)’ is gravitational acceleration and ‘\(k\)’ is a value between 0 to 1 to demonstrate the portion of the valve is open. In this system proportional valve is used and it gets controlled through Analog output of PLC. 0 indicates that the valve is fully closed and 1 means fully open, and any value like 0.25 means a portion of the valve (here %25) is open. For better demonstration in HMI display it scaled up between 0 to %100.

So from (4.1) and (4.2),

\[ A \frac{dh_i}{dt} = F_{in_i} - k_i \times a_i \sqrt{2 \times g \times h_i} \] (4.3)

So,

\[ \frac{dh_i}{dt} = \frac{F_{in_i}}{A_i} - k_i \times a_i \sqrt{2 \times g \times h_i} \] (4.4)

Since there are three tanks cascaded to each other output flow of tank 1 is input flow of tank 2 and so on.
\[ F_{out_1} = F_{in_2} \] (4.5)

\[ F_{out_2} = F_{in_3} \] (4.6)

For tank 1,

\[
\frac{dh_1}{dt} = \frac{F_{in_1}}{A_1} - \frac{k_1 \times a_1 \sqrt{2 \times g \times h_1}}{A_1} \] (4.7)

For tank 2, from (4.4) and (4.5),

\[
\frac{dh_2}{dt} = \frac{F_{in_2}}{A_2} - \frac{k_2 \times a_2 \sqrt{2 \times g \times h_2}}{A_2} \] (4.8)

From (4.5) and (4.8),

\[
\frac{dh_2}{dt} = \frac{k_1 \times a_1 \sqrt{2 \times g \times h_1}}{A_1 \times A_2} - \frac{k_2 \times a_2 \sqrt{2 \times g \times h_2}}{A_2} \] (4.9)

And for tank 3,

\[
\frac{dh_3}{dt} = \frac{k_2 \times a_2 \sqrt{2 \times g \times h_2}}{A_2 \times A_3} - \frac{k_3 \times a_3 \sqrt{2 \times g \times h_3}}{A_3} \] (4.10)
Table 4.1 demonstrates parameters and constants with associated values and ranges. Some of values and ranges are scaled and weighted to make model analysis and controller design simpler and easier. Variables should be scaled to get a maximum variety of input and outputs.

Table 4.1 Parameters and Constants description, range and value

<table>
<thead>
<tr>
<th>Constant and Parameter</th>
<th>Description</th>
<th>Values with units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1, A_2, A_3$</td>
<td>Cross section area of tanks</td>
<td>$100 \text{ cm}^2$</td>
</tr>
<tr>
<td>$a_1, a_2, a_3$</td>
<td>Cross section area of orifices</td>
<td>$5 \text{ cm}^2$</td>
</tr>
<tr>
<td>$h_1, h_2, h_3$</td>
<td>Level in tanks</td>
<td>0 to 10 cm</td>
</tr>
<tr>
<td>$k_1, k_2, k_3$</td>
<td>Valves output</td>
<td>0 to 1</td>
</tr>
<tr>
<td>$g$</td>
<td>Gravitational acceleration</td>
<td>$9.8 \frac{m}{s^2}$</td>
</tr>
</tbody>
</table>

4.2 Verification of control system through eigenvalues

In

Table 4.2 operating points of system in a specific state has been recorded. We will use these values or points in Jacobian Matrix, and then we’ll find Eigenvalues, and it will determine system is stable or not.

Table 4.2 Sample points of system

<table>
<thead>
<tr>
<th>Tank</th>
<th>$h$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank 1</td>
<td>$h_1 = 8$</td>
<td>$k_1 = 0.14$</td>
</tr>
<tr>
<td>Tank 2</td>
<td>$h_2 = 5$</td>
<td>$k_2 = 0.21$</td>
</tr>
<tr>
<td>Tank 3</td>
<td>$h_3 = 3$</td>
<td>$k_3 = 0.26$</td>
</tr>
</tbody>
</table>
In the previous part, we observed that water tank level equations are nonlinear, so for linearization of the system we have to create below equation and substitute operating points on that.

\[
\begin{bmatrix}
\Delta \dot{x}_1 \\
\Delta \dot{x}_2 \\
\Delta \dot{x}_3
\end{bmatrix} =
\begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\
\frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3}
\end{bmatrix}
\begin{bmatrix}
\delta x_1 \\
\delta x_2 \\
\delta x_3
\end{bmatrix} +
\begin{bmatrix}
\frac{\partial f_1}{\partial u} \\
\frac{\partial f_2}{\partial u} \\
\frac{\partial f_3}{\partial u}
\end{bmatrix}
\begin{bmatrix}
\delta u_1 \\
\delta u_2 \\
\delta u_3
\end{bmatrix}
\] (4.11)

After creating above equation, it represents following state space form of the system.

\[
\dot{X} = AX + BV
\] (4.12)

\[
V = -KX
\] (4.13)

And

\[
\dot{X} = AX - BHX = (A - BK)X
\] (4.14)

By finding derivative and creating above matrices in operation points, we reach following matrices.
\[ A - BK = \begin{bmatrix} -0.54 & 0 & 0 \\ 0.54 & -1.04 & 0 \\ 0 & 1.04 & -1.66 \end{bmatrix} \]

And, eigenvalues are: \( \lambda_1 = -1.66, \lambda_2 = -1.04, \lambda_3 = -1.04 \).

This determines that our system with selected gains are stable.
5 Result and Conclusion

In this project, a process has been implemented successfully in MATLAB based on physics equations. And the control system has been developed in PLC with ladder logic code. This code has been tested and verified through an OPC based connection between these two objects, MATLAB and PLC. One of the advantages of this kind of experiment is about that whoever is developing the code can get a better understanding and perception of system behavior, and it ends to implement optimized and efficient control design. HIL (Hardware In Loop) simulation is a type of virtual simulation that the real control system is used to control virtual model of a real plant. Virtual simulation demand is increasing rapidly because of safety operations and low cost. Using OPC communication for connecting MATLAB to Allen Bradley PLC is a modern approach that makes a reliable and stable connection. It makes possible to do academic level experiments on every industrial complex project using virtualization without creating even any small scale hardware.
6 References


7 Appendix: Ladder Logic

This part is calling subroutines on every scan of PLC code.

This part is for digital command of valves to open, and indication on HMI, when output is ‘1’ valves turns to green (open), and when it’s ‘0’ turns to red (closed).
This is digital command to open "Valve2" and also indication of it on HMI
When general_bit[1] is on "Valve2" color changes to green
When general_bit[1] is off "Valve2" color changes to red

Valve 2
Hand Mode
HOA_Internal_Bits[10]
On/Off bits of
"Valve2" in Hand
mode
HOA_Internal_Bits[16]
"Valve2" Analog
position 0-100%
GRT
Greater Than (A>B)
Source A  analog_Tank1[13]
0.0+
Source B
0
Valve 2
Auto Mode
HOA_Internal_Bits[12]

This is digital command to open "Valve3" and also indication of it on HMI
When general_bit[2] is on "Valve3" color changes to green
When general_bit[2] is off "Valve3" color changes to red

Valve 3
Hand Mode
HOA_Internal_Bits[20]
On/Off bits of
"Valve3" in Hand
mode
HOA_Internal_Bits[26]
"Valve3" Analog
position 0-100%
GRT
Greater Than (A>B)
Source A  analog_Tank2[12]
0.0+
Source B
0
Valve 3
Auto Mode
HOA_Internal_Bits[22]

This is digital command to open "Valve4" and also indication of it on HMI
When general_bit[3] is on "Valve4" color changes to green
When general_bit[3] is off "Valve4" color changes to red

Valve 4
Hand Mode
HOA_Internal_Bits[30]
On/Off bits of
"Valve4" in Hand
mode
HOA_Internal_Bits[36]
"Valve4" Analog
position 0-100%
GRT
Greater Than (A>B)
Source A  analog_Tank3[12]
0.0+
Source B
0
Valve 4
Auto Mode
HOA_Internal_Bits[32]

General_bits[1]
General_bits[2]
General_bits[3]
The following part shows the HOA related logic. With this logic, valves modes change between three modes Hand-Off-Auto, and makes possible to turn on in Hand mode.
This part displays logic for HOA
By pressing push button for 'Valve2' on HMI
State of 'Valve2' from Hand-to-Off-to-Auto
Valve 2 HOA

Valve 2
Hand Mode
HOA_Selection[1].0 HOA_Internal_Bits[10]

Valve 2
Off Mode
HOA_Selection[1].1 HOA_Internal_Bits[11]

Valve 2
Auto Mode
HOA_Selection[1].2 HOA_Internal_Bits[12]

This part displays logic for HOA
By pressing push button for 'Valve3' on HMI
State of 'Valve3' from Hand-to-Off-to-Auto
Valve 3 HOA

Valve 3
Hand Mode
HOA_Selection[2].0 HOA_Internal_Bits[20]

Valve 3
Off Mode
HOA_Selection[2].1 HOA_Internal_Bits[21]

Valve 3
Auto Mode
HOA_Selection[2].2 HOA_Internal_Bits[22]
This part displays logic for HOA
By pressing push button for ‘Valve4’ on HMI
State of ‘Valve4’ from Hand-to-Off-to-Auto
Valve 4 HOA

Valve 4
Hand Mode
HOA_Selection[3].0 HOA_Internal_Bits[30]

Valve 4
Off Mode
HOA_Selection[3].1 HOA_Internal_Bits[31]

Valve 4
Auto Mode
HOA_Selection[3].2 HOA_Internal_Bits[32]
In following pages, analog values from PID blocks and scaling part are moved to specific registers to be used in MATLAB Simulink.
In this part Analog value of "Valve2" is sent to MATLAB Simulink and HMI
Based on H-O-A mode

```
Valve 2
Hand Mode
HOA_Internal_Bits[10]

'Move'
Source  HMI_Analog[1]
Dest  analog_Tank1[11]
0.0*

'Move'
Source  analog_Tank1[12]
Dest  analog_Tank1[11]
0.0*

Clear
Dest  HMI_Analog[1]
0.0*

Clear
Dest  analog_Tank1[11]
0.0*
```

Valve 2
Auto Mode
HOA_Internal_Bits[12]
In this part, analog value of "Valve3" is sent to MATLAB Simulink and HMI based on H-O-A mode.

- **Valve3** Analog to position 0-100% To MATLAB Simulink
  - MOV
  - Source: HMI_Analog[2]

- **Valve3** Analog to position 0-100% To MATLAB Simulink
  - MOV
  - Source: analog_Tank2[12]

- **Valve3** position in Hand mode coming from HMI
  - CLR
  - Dest: HMI_Analog[2]

- **Valve3** Analog to position 0-100% To MATLAB Simulink
  - CLR
In this part Analog value of "Valve4" is sent to MATLAB Simulink and HMI Based on H-O-A mode

Valve 4
Hand Mode
HOA/Internal Bits[30]

Valve 4
Auto Mode
HOA/Internal Bits[32]

Valve 4
Hand Mode
HOA/Internal Bits[30]

Valve 4
Off Mode
HOA/Internal Bits[31]

'Valve4' Analog to position 0-100% To MATLAB Simulink

Move
Source HMI_Analog[3]
0*
Dest analog_Tank[11]
0.0*

Move
Source analog_Tank[12]
0.0*
Dest analog_Tank[11]
0.0*

'Valve3' position in Hand mode coming from HMI

Clear
Dest HMI_Analog[2]
0*

Clear
Dest analog_Tank[11]
0.0*
In following pages, level setting points for Tank1 from HMI and level actual value from MATLAB are scaled to be used in PID Blocks. And also PID block gains are entered from HMI to here.
Scaling 'Valve2' position from 0-16383 to 0-100

'Valve2' Analog position 0-100%

Compute
Dest analog_Tank1[12] 0.0+
Expression (analog_Tank1[22]*100)/16383

Moving Proportional Gain 'Kp' to PID block

'Valve2' PID
Proportional Gain 'Kp'

Move
Source analog_Tank1[3] 0.0+
Dest Tanks_Level_PID[0].KP 0.0+

Moving Integral Gain 'Ki' to PID block

'Valve2' PID
Integral Gain 'Ki'

Move
Source analog_Tank1[4] 0.0+
Dest Tanks_Level_PID[0].KI 0.0+

Moving Derivative Gain 'Kd' to PID block

'Valve4' PID
Derivative Gain 'Kd'

Move
Source analog_Tank1[5] 0.0+
Dest Tanks_Level_PID[0].KD 0.0+
Moving set-point "SP" to PID block

'Valve2' PID
Setpoint
which is desired
level value in Tank1

Move
Source  analog_Tank1[16]
        0.0
Dest    Tanks_Level_PID[0].SP
        0.0

Valve 2
Auto Mode
HOA_Internal_Bits[12]

'Valve2' PID for
Tank1
Tanks_Level_PID[0].SWM

'Valve2' PID for
Tank1
PID

Proportional Integral Derivative
PID  Tanks_Level_PID[0]
Process Variable  analog_Tank1[20]
Tieback  0
Control Variable  analog_Tank1[22]
PID Master Loop  0
Inhold Bit  0
Inhold Value  0
Setpoint  0.0
Process Variable  0.0
Output %  0.0
In following 2 pages, level setting points for Tank2 from HMI and level actual value from MATLAB are scaled to be used in PID Blocks. And also PID block gains are entered from HMI to here.
Scaling 'Valve3' position from 0-16383 to 0-100

'Valve3' Analog position 0-100%

- Compute
  - Dest: analog_Tank2[12]
    - Expression: \( \text{analog}\_\text{Tank2}[22] \times 100 / 16383 \)

Moving Proportional Gain 'Kp' to PID block

'Valve3' PID Proportional Gain 'Kp'

- Move
  - Source: analog_Tank2[3]
  - Dest: Tanks_Level_PID[1].KP

Moving Integral Gain 'Ki' to PID block

'Valve3' PID Integral Gain 'Ki'

- Move
  - Source: analog_Tank2[4]
  - Dest: Tanks_Level_PID[1].KI

Moving Derivative Gain 'Kd' to PID block

'Valve3' PID Derivative Gain 'Kd'

- Move
  - Source: analog_Tank2[5]
  - Dest: Tanks_Level_PID[1].KD
Moving set-point 'SP' to PID block

'Valve3' PID
Setpoint
which is desired
level value in Tank2

Move
Source: analog_Tank2[16]
0.0*
Dest: Tanks_Level_PID[1].SP
0.0*

Valve 3
Auto Mode
HOA_Internal_Bits[22]

'Valve3' PID for Tank3
Tanks_Level_PID[1].SWM

'Valve3' PID for Tank3
PID

Proportional Integral Derivative
PID: Tanks_Level_PID[1]
Process Variable: analog_Tank2[20]
Tieback: 0
Control Variable: analog_Tank2[22]
PID Master Loop: 0
Inhold Bit: 0
Inhold Value: 0
Setpoint: 0.0*
Process Variable: 0.0*
Output %: 0.0*
In following 2 pages, level setting points for Tank3 from HMI and level actual value from MATLAB are scaled to be used in PID Blocks. And also PID block gains are entered from HMI to here.

Scaling 'Valve4' level set-point from 0-16383 to 0-10

'Valve4' Level set-point scaled from 0-10 to 0-16383 for use in PID block

Compute
Dest analog_Tank3[18]
Expression (analog_Tank3[8]*16383)/10

Scaling 'Valve4' level set-point from 0-10 to 0-16383

'Tank3' level 'h3'
From MATLAB Simulink Scaled to 0-16383

Compute
Dest analog_Tank3[20]
Expression (analog_Tank3[10]*16383)/10
Scaling 'Valve4' position from 0-16383 to 0-100

'Valve4' Analog position 0-100%

Compute CPT
Dest analog_Tank3[12] 0.0 +
Expression (analog_Tank3[22]*100)/16383

Moving Proportional Gain 'Kp' to PID block

'Valve4' PID
Proportional Gain 'Kp'

Move MOV
Source analog_Tank3[3] 0.0 +
Dest Tanks_Level_PID[2].KP 0.0 +

Moving Integral Gain 'Ki' to PID block

'Valve4' PID
Integral Gain 'Ki'

Move MOV
Source analog_Tank3[4] 0.0 +
Dest Tanks_Level_PID[2].KI 0.0 +

Moving Derivative Gain 'Kd' to PID block

'Valve4' PID
Derivative Gain 'Kd'

Move MOV
Source analog_Tank3[5] 0.0 +
Dest Tanks_Level_PID[2].KD 0.0 +
Moving set-point 'SP' to PID block

'Valve4' PID
Setpoint which is desired level value in Tank3

Move
Source analog_Tank3[16] 0.0
Dest Tanks_Level_PID[2].SP 0.0

Valve 4
Auto Mode HOA_Internal_Bits[32]

'Valve4' PID for Tank3
Tanks_Level_PID[2].SWM

'Valve4' PID for Tank3
PID

Proportional Integral Derivative
PID Tanks_Level_PID[2]
Process Variable analog_Tank3[20]
Tieback 0
Control Variable analog_Tank3[22]
PID Master Loop 0
Inhold Bit 0
Inhold Value 0
Setpoint 0.0
Process Variable 0.0
Output % 0.0