Specialized

Graduation Project Phase 1

Supervised by:

Dr. Mohamed Ibrahim Awad

Date:31/1/2024

Digital Twin & IIOT In Industry

Industry 4.0

Submitted By:

1-Ahmed Ebrahim ELSayed	1900218
2-Mohamed Abd El Hakeem El Said	1901257
3-Mohamed El Sadek metwally	1901277
4-Omar Hamdy gbr	1900938

1 Elsarayat St., Abbaseya

11517 Cairo, Egypt Fax:

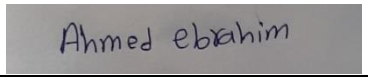
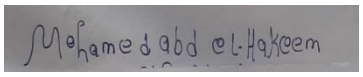
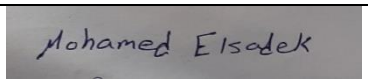
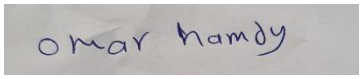
(+20 2) 26850617

www.eng.asu.edu.eg

DECLARATION

We/I hereby certify that this Project submitted as part of our/my partial fulfilment of BSc in ***(Mechatronics Engineering)*** is entirely our/my work, that we/I have exercised reasonable care to ensure its originality, and does not to the best of our/my knowledge breach any copyrighted materials, and have not been taken from the work of others and to the extent that such work has been cited and acknowledged within the text of our/my work.

Signed: by all students

Name: Ahmed Ebrahim El Sayed	Signature: 
Name: Mohamed Abd El Hakeem El Said	Signature: 
Name: Mohamed El Sadek metwally	Signature: 
Name: Omar Hamdy gbr	Signature: 

Date: Thursday, 1st of February

ACKNOWLEDGMENT

We would like to take this opportunity to express our heartfelt gratitude to Prof. Mohamed Ibrahim Awad for allowing us this great opportunity to work on this project and for his constant guidance and support throughout the year. It was a great honour to work under his supervision and have him entrust us. To our parents for their unwavering support and encouragement, for all their motivation and inspirational words and actions throughout this long academic journey. We would also like to extend our gratitude to ENG: Ahmed Salah for their continuous suggestions, and constructive criticism.

ABSTRACT

This report represents the progress throughout the graduation project concerning the SCADA and Digital Twin & IIoT automation of industrial process control. The main target outcome of this project is understanding the system functionality, components, and control design, which was successfully achieved by studying the system flow and the role of each component and tracing the highly complex wire connections inside the control panel and the controller box. After running some tests, some components were found to be damaged either during the moving process of the system or due to being left for a long time without maintenance, therefore they were replaced or repaired. Some issues were only discovered after powering up the system and observing the system's behavior. The report also includes the next phase, which was improving the system by adding PLC, SCADA, and IIoT, the problems that were faced, how it was solved, and step-by-step how to reach the goal. For the PLC part, three PLCs were used. For the SCADA part, due to problems faced in communication, all the PLCs are connected to the same computer at the same time. For the IIoT part, the communication between PLCs with each other and trying to make the master control that can take any action on the system and reading the data from the system and then the many types of protocols that PLCS have them from this side, on another side ESP32 Can it handle any of these Protocols for transferring the data from PLCs to the cloud? , also the little resources available for the idea, and finally trying to find a free cloud to make IIot of our system on it, where the most is the commercia

Table of Contents

Table of Figures

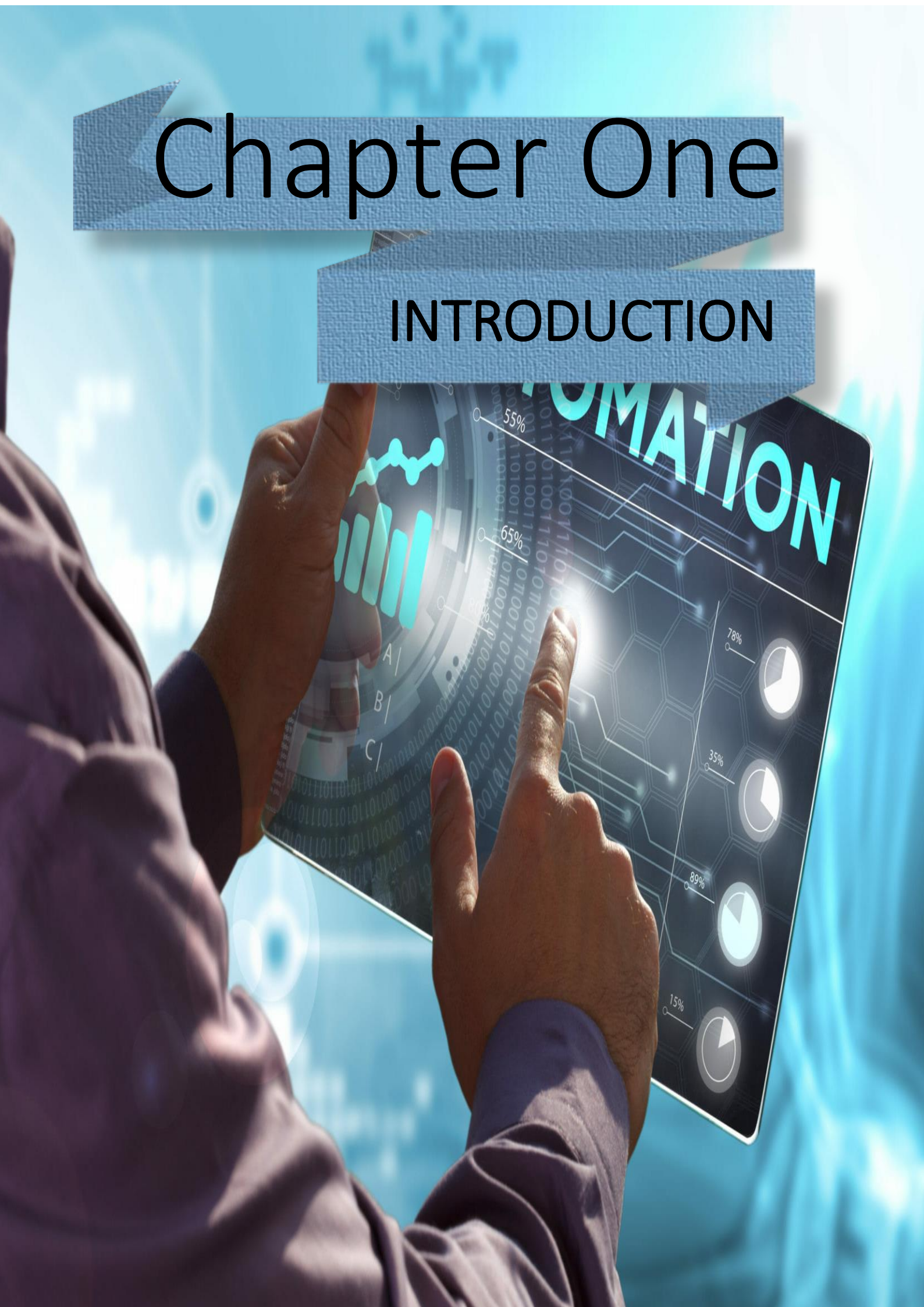
List of Tables

Nomenclature

D1	**Tank**
D2	Feed tank
C1	Column
E1	Heat exchanger
M1	The motor of the heat exchanger fan
M2	Stirrer of the tank
J1	Electrical resistance
TA1	Thermostat for temperature in the tank
TI1	RTD for temperature measurement after cooling
CE1	Conductivity indicator
G1	Centrifugal pump feeding column
G2	Centrifugal pump recirculating back to the tank
TI2	RTD for thermostat
WT1	Two load cells
FT1	Flow-rate transmitter
LT1	Differential-pressure transmitter
PA1	Minimum pressure switch
PA2	Maximum pressure switch
FC1	Pneumatic control valve for flow controls out of the tank
LC1	Pneumatic control valve for the level inside the column
TT1	Temperature transmitter

Chapter One

INTRODUCTION



The multi-process control system, specifically the Mod. UNIPRO/EV model made by Electronica Veneta -an Italian company- encompasses various components and functionalities, designed about 30 years ago. The system comprises a glass column that is supplied with liquid through a centrifugal pump. The pump is interconnected with a tank incorporating an electric resistor for heating. Additionally, another centrifugal pump is responsible for recirculating the water from the column to the tank through an air-cooled heat exchanger.

To regulate and monitor the system, a PID controller known as Digitric 500, which is manufactured by ABB Instrumentation, is employed. Along with a special microprocessor made for the conductivity sensor, and the switches and indicators fixed on the side of the control panel.

And that was it for this super old-fashioned and out-of-date system and it was left rusting and aging with no proper use whatsoever. It needed help and restoration to its original form. Therefore, some modernization was critical for the system to make it more tempting for others to work with, in addition to improving its capabilities and accuracy.

The system's control capabilities have been enhanced by installing three XBCDR30SU programmable logic controllers (i.e., PLC) controllers. The controlling technique can easily be switched between the old controller or the PLCs with a click of a button. One can easily monitor the system -when controlled by the PLC system- using the XP-builder supervisory control and data acquisition (i.e., SCADA) software package. For remote monitoring from anywhere or any device, the TeamViewer application is utilized with Info for a user-friendly experience. Also, as an alternative means of control, the ESP32 WROOM module has been integrated, enabling the utilization of the Industrial

Internet of Things (i.e., IIOT) platform known as Ubidots Cloud for data uploading and monitoring in the cloud.

This system will serve a respected and important mission as it would be used as an educational kit to help other students from different levels with their automation courses and most importantly, save the system with its valuable components from being thrown away.

Figure 1:Whole System

Chapter Two

SYSTEM OVERVIEW

The system is essentially an educational kit equipped with mechanical, electrical, and control elements allowing students to grasp the concept of multi-process control. The system's model dates to the 90s, with very limited documentation available which made understanding the system's operation and functionality quite challenging.

The road for system development has been paved in the previous semester through the following points: Understanding the system flow, control loops, and each component function, Inspection of components, connections, and maintenance/replacement of faulty ones.

1. SYSTEM FLOW AND COMPONENTS

The process starts with a borosilicate glass tank filled with water connected to another small feed tank that contains the solution. The two fluids can be mixed by a stirrer and heated to a specific temperature by electrical resistance.

A conductivity indicator with a probe that can withstand up to 100℃ is used to measure the mixture's conductivity. Both the mixing and heating elements can be controlled by P.C. or a manual switch on the control panel.

The temperature inside the tank is maintained at a specified set point through a thermostat. RTD pt100 sensor is used for measuring the fluid temperature coupled with a temperature transmitter that produces a 4-20 mA output signal.

A centrifugal pump feeds another borosilicate glass column, where the input flow to the column is controlled by a flow-rate transmitter and pneumatic control valve. The column is mounted on two load cells and a differential-pressure transmitter is inserted at the bottom of the column for level measurement.

Another centrifugal pump is placed after the column to recirculate the fluid into the tank. The output flow from the column is controlled by minimum and maximum pressure switches and pneumatic control valves. The fluid is passed by a heat exchanger (motor fan with tachometer generator) for cooling. The output temperature from the heat exchanger is measured by the RTD pt100 sensor.

The following diagram visualizes the process flow in detail:

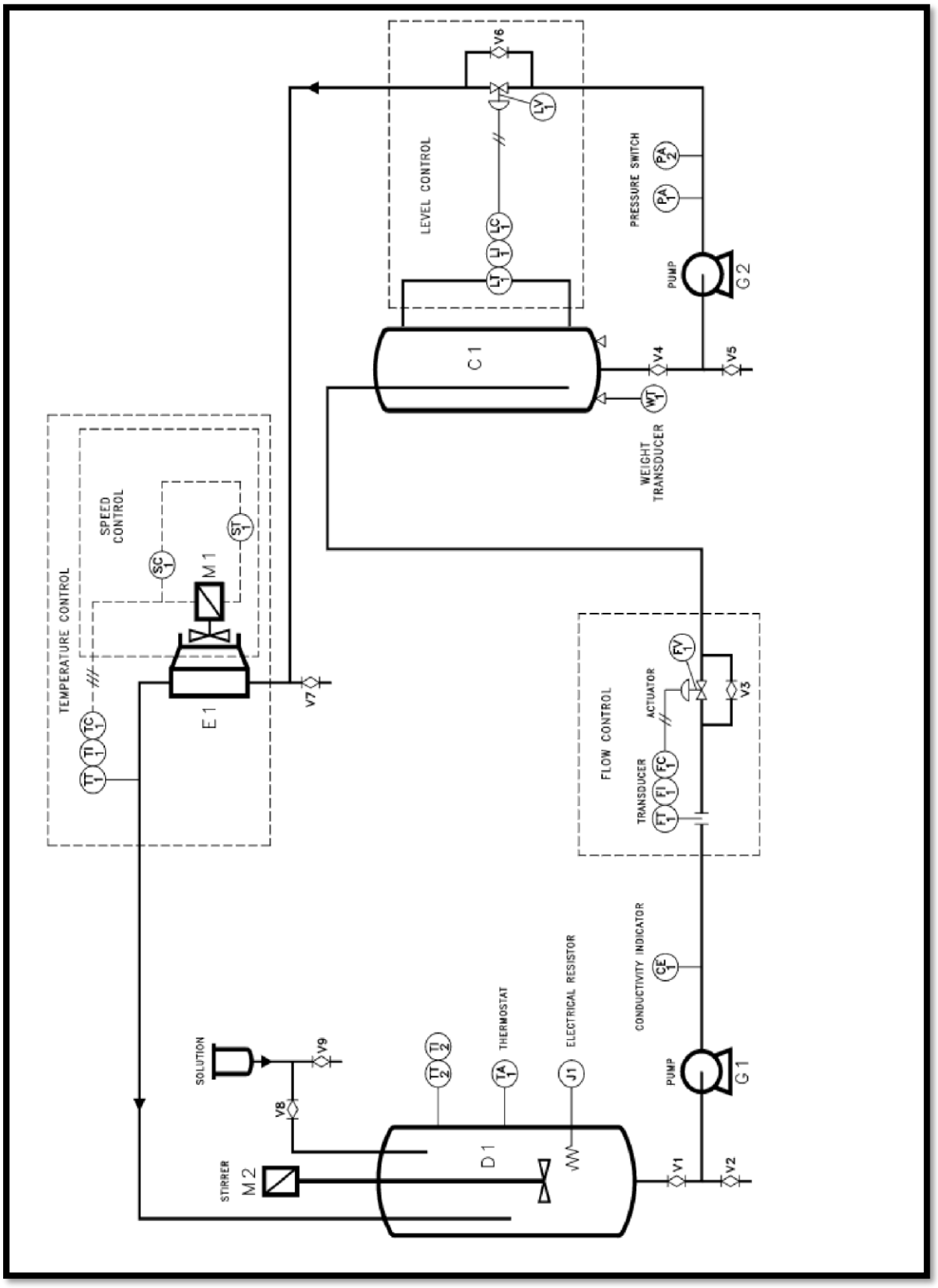


Figure 2: System Flow Diagram

2. WIRING DIAGRAM

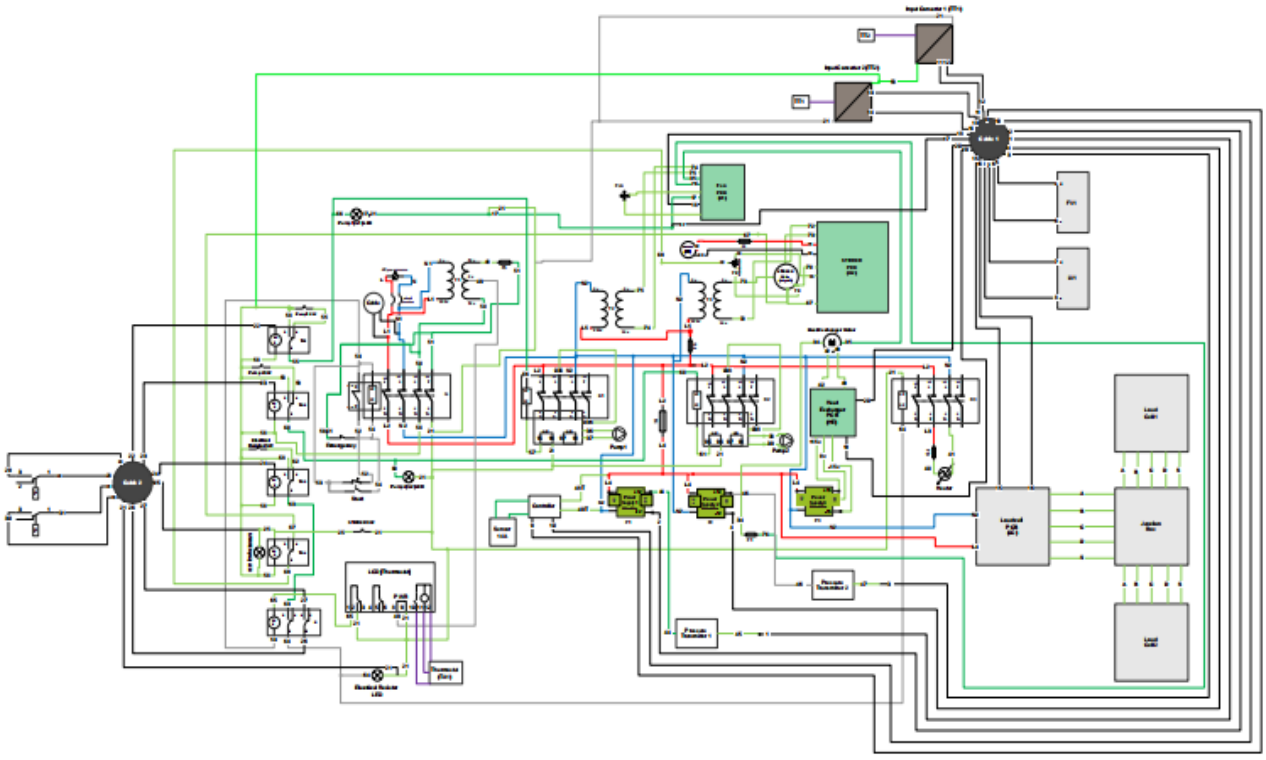


Figure 3:Wiring diagram

3. QR-code for wiring diagram

Part Name	Quantity	Specification
Old System		
Tank	1	Borosilicate Glass (20L Capacity)
Auxiliary Tank (Feed)	1	Borosilicate Glass (1L Capacity)
Column	1	Borosilicate Glass (10L Capacity)
Centrifugal Pump	2	-AISI 304 stainless steel – Qmax = 4.25 m3 /h -Hmax = 32.5 m
Air-cooled heat exchanger	1	- Motor fan speed = 1500 rpm - Tachometer generator and electronic card signal: 0-1300 rpm / 4-20 mA
Electronic flow-rate transmitter	1	- AISI 316 stainless steel - Range: 0-0.6 m3 /h - Output signal: 4-20 mA
Electronic differentialpressure transmit	1	- AISI 316 stainless steel - Range: 0-500 mm H2O - Output signal: 4-20 mA
Pneumatic Control Valve	2	- AISI 316 stainless steel - Cv = 2.5
Electro-pneumatic converters	2	- 4-20 mA / 0.2-1 bar

Load Cell	2	- Range: 0-10 kg - Output signal: 4-20 mA mv/Volt indicator
Electronic Thermostat	1	- Programmable from 0-100 ℃
Conductivity transmitter/indicator	1	- Range: 0-2000 **µ**S - Probe can withstand up to 120℃
Pressure Switch	2	- Minimum: 1.5 bar - Maximum 3.5 bar
RTD Sensor	2	- Pt 100 - AISI 304 stainless steel
Stirrer	1	- AISI 304 stainless steel - Variable speed reducer - Reduction ratio 6:1
Electrical Resistance	1	- AISI 304 stainless steel - P = 3kW
Temperature Transmitter	2	- AISI 304 stainless steel - Output signal: 4-20 mA
Electronic Microprocessor PID Controller	2	- With 4-line LCD - Serial card
Control Panel	1	- IP55
Transformer	3	- 2transformers 230/24 V - 1 transformer 230/12 V
Power Supply	3	- 2 power supplies 12-Vdc - 1 power supply 15-Vdc
PLC&SCADA		
XBC-DR30SU	3	----------

XB0-AD02A	4	----------
XB0-AD02A	2	----------
USB to V3	3	----------
USB to rs485	3	----------
USB hub	1	----------
IIOT		
ESP32	1	----------
PCB	1	----------
Micro USB cable	1	----------
Installation		
Palette	1	
Wires 1mm	Yellow: 20 meters Blue: 20 m Red: 5 m	----------
Wires 2mm	Black: 5 m	----------
Terminal Block	40 2 for power, 6 for communication, and the rest are for the actuators and sensors	----------
Rail	1 meter	----------

Duct	1.5 meter	----------

Table 1:Bill of material

Chapter Three

MAINTENANCE

After powering up the system it was noticed that some components were not functioning as intended and had to be replaced/fixed. The following list summarizes these changes:

1. Replacing some ICs, and fixing the voltage regulator in fan PCB

- We find some problems with this driver when testing it by Avometer.
- there are five circles mentioned to the components that we replace or fix them.
- voltage regulator in Red circle while ICs in orange circle.

Figure 4: Fan driver

- After repairing this board, we were able to adjust the different fan speeds

2. Radiator maintenance and cleaning

- After checking the radiator, we find The issue in the cooling rate so we decided that it need for repairing.
- After maintenance, the cooling rate was increased and it began to operate very efficiently

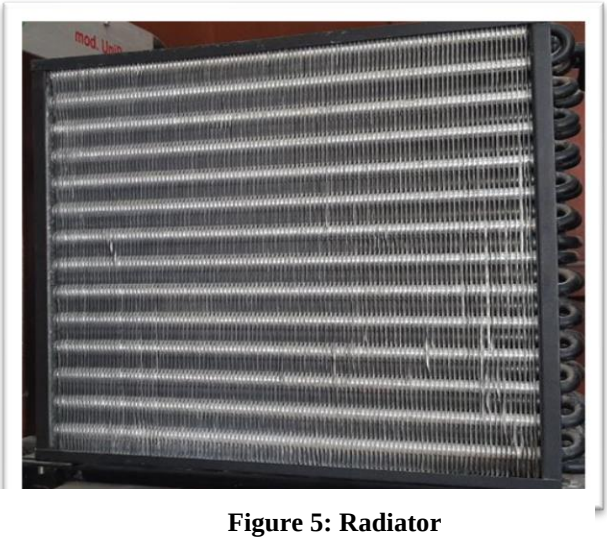


Figure 5: Radiator

3. Stirrer repairing

1. Problem
   - The method of the coupler fixation was not correct or strong enough due to wrong fixation.

2. Solution
   - We disassembled the coupler and considered creating a level surface on the motor shaft. We then attached the screw to it and fixed it between shaft of motor and shaft of the stirrer.

Figure 6:Coupler

4. Tank repairing

- This was perhaps the most challenging part of the hardware maintenance process. Therefore, we tried a lot to implement an effective repairing method.

3. problem

   There is leakage in the bottom of this tank due to break in it.

4. solutions

   1- Make a mechanical drawing of the tank to manufacturing a new one but we find it is too expensive.
   2- Make a connection between tank and pipes made by 3d printing but it wasn't sufficient because there is a leakage again.
   3- Put a Silicon in between this connection and the tank, and between this connection and the pipe but it is not good enough.
   4- We reduce the length of this connection and putting a silicon also and we find it is the best solution until now

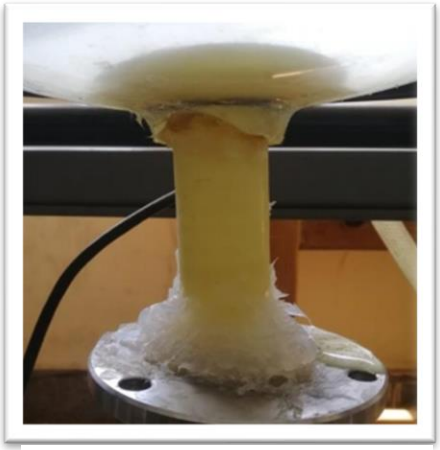


Figure 7:Tank repairing 1

5. Current result

Based on the last solution we reached, we took into consideration not to completely disassemble the tank, as doing so would affect the entire machine's functionality, and we wouldn't be able to test the remaining components required for the project

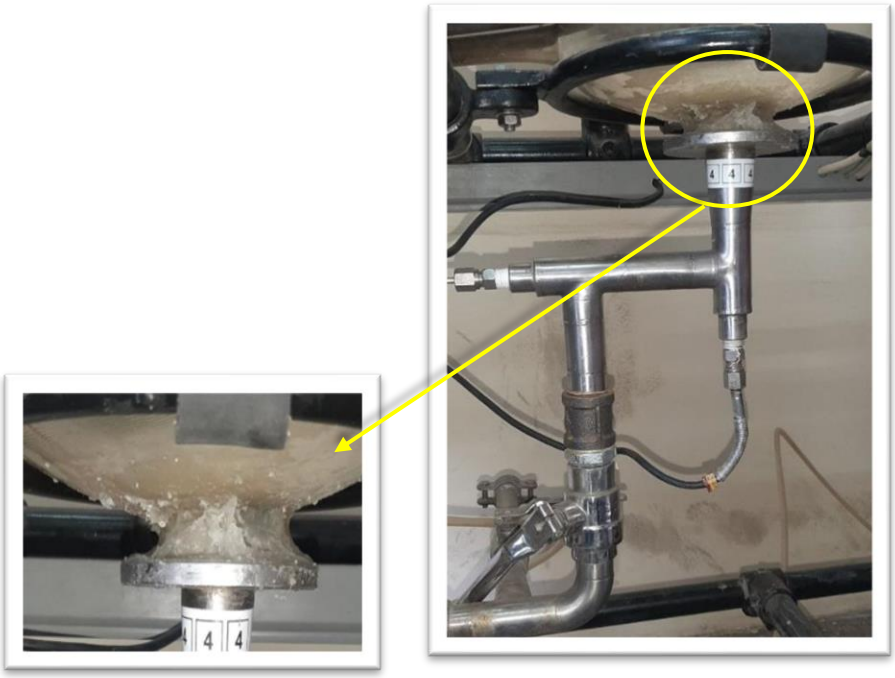


Figure 9:Magnification of Tank repairing 2

Figure 8:Tank repairing 2

Chapter Four

PLC

1. Introduction

At this phase, the system is to be controlled by programmable logic controllers (PLCs) and using Analog Digital Converters [ADC] to communicate with the analog sensors, heat exchanger motor, and the two pneumatic valves. The PLCs from LS Electric of model XBCDR30SU, one of them is the Master and the others are Slaves. Two ADCs connected to each PLC.

A PLC or a programmable logic controller is an industrial computer that automates electromechanical processes like factory assembly lines. PLCs are anticipated to function perfectly for years in hazardous industrial environments. It mainly consists of a CPU, memory, and I/O modules and works with a suitable programming device. Compared to old controlling systems, A PLC has the advantage of a smaller circuit with fewer components, an easier troubleshooting mechanism, and exceptional computation abilities.

2. SYSTEM SPECS

The system requires the following:

- 7 analog input channels (TT1, TT2, FT1, LT1, WT1, ST1, CE1)
- 3 analog output channels (SC1, TC1, FC1, LC1)
- 2 digital input channels (PA1, PA2)
- 4 digital output channels (G1, G2, D1, J1)

The available PLC was a LS-electric product -originated from South Korea-, model XBC-DR30SU which has the following specifications:

Rated Voltage (UL warranty voltage)	AC 100 ∽ 240V
Input voltage range	AC85∽ 264V (-15%, +10%)
Inrush current	50APeak or less
Input current	0.5A or less (220V), 1A or less (110V)
Efficiency	65%
Permitted momentary power failure	10ms or less
Power supply status indication	LED On when the power supply is normal
Rated Output	1.5A @ DC5V; 0.3A @ DC24V
Voltage Ripple	DC5V (±2%)
Cable specification	0.75 ∽ 2 mm2
Digital input	0.18 (AWG24) - 1.5 (AWG16)
Digital output	0.18 (AWG24) - 2.0 (AWG14)
Analogue I/O	0.18 (AWG24) - 1.5 (AWG16)
Communication	0.18 (AWG24) - 1.5 (AWG16)
Main power	1.5 (AWG16) - 2.5 (AWG12)
Protective grounding	1.5 (AWG16) - 2.5 (AWG12)
Terminal Strip Torque	3.7 - 5.1 in/lb

Dimensions in. (LxHxD)	5.3 x 3.5 x 2.5"

Table 2: PLC General Specifications

Operation Voltage Range	20.4–28.8VDC (Within Ripple Rate 5%)
Rated Input Voltage	24VDC
Rated Input Current	4mA (Contact Point 0–1: 16mA, 2–7: 10mA)
Input Resistance	5.6kΩ (P00–P07: 2.7kΩ)
On Voltage/On Current	19VDC or Higher/3mA or Higher
Off Voltage/Off Current	6VDC or Less/1mA or Less
Response Time	
Off ➔ On / On ➔ Off	1/3/5/10/20/70/100ms (Set by CPU Parameter) Default: 3ms
Insulation Voltage	560VAC rms/3 Cycle (Altitude 2000m)
Insulation Resistance	10MΩ or Higher
Relay Outputs	
Rated Load Voltage/Current	24VDC 2A/220VAC 2A, 5A/COM
Min. Load Voltage/Current	5VDC/1mA
Max. Load Voltage	250VAC, 125VDC

Off Leakage Current	0.1mA (220VAC, 60Hz)
Max. On/Off Frequency	3,600 Times/Hour
Response Time	
Off ➔ On	10ms or Less
On ➔ Off	12ms or Less
Service Life	
Mechanical	Rated Load Voltage/Current 100,000 Times or More
Electrical	200VAC/1.5A, 240VAC/1A 100,000 Times or More 200VAC/1A, 240VAC/0.5A 100,000 Times or More 24VDC/1A, 100VDC/0.1A 100,000 Times or More

Table 3: I/O Specifications

Program Control Method	Fixed Scan Time, Constant Scan
I/O Control Method	Scan Synchronous Batch Processing Method (Refresh Method), Directed by Program Instruction
Program Language	Ladder Diagram, Instruction List
Processing Speed	94μs /1K Boolean
Program Capacity	15k steps
Expansion Capability	Expansion Modules and Option Boards (Max. 7 Combined)
Data Area	
P	P0000–P1023F (16,384 points)
M	M0000–M1023F (16,384 points)
K	K00000–K4095F (65,536 points)
L	L00000–L2047F (32,768 points)
F	F000–F1023F (16,384 points)
T	100ms, 10ms, 1ms: T0000–T255 T1023 (1,024 point) (Adjustable by Parameter Setting)

C	C000–C1023 (1,024)
S	S00.00–S127.99
D	D0000–D10239 (10,240 word)
U	U00.00–U0A.31 (Analog Data Refresh Area: 352 words)
Z	Z000–Z127 (128 word)
R	R0000–R10239 (10,240 word)
Total Programs	128
Tasks Available	
Initial Task	1
Cyclic Task	8
I/O Task	8
Internal Device Task	8
Operation Mode	Run, Stop, Debug
Self-Diagnosis Function	Detects Errors of Scan Time, Memory, I/O
Program Ports	Mini DIN 6-Pin RS-232C, USB 1 Channel
Back-up Method	Latch Area Setting in Basic Parameter
Built-in Functions	

PID Control	Control by Instruction, Auto-Tuning, PWM Output, Manual Mode, Adjustable Operation Scan Time Setting, Anti-Windup, Delta MV, SV-Ramp Function, Max. 16 Loops
Serial	
Protocol	Dedicated Protocol, Modbus Protocol, User Defined Protocol
Channel	RS-232C 1 port, RS-485 1 port
High-Speed Counter	
Performance	1 Phase: 100kHz, 2 Channel/20kHz, 6 Channel 2 Phase: 50kHz, 1 Channel/8kHz, 3 Channel
Counter Mode	Based on Input Pulse and INC/DEC method – 1 Phase Pulse Input: Addition/Subtraction Counter 2 Phase Pulse Input: Addition/Subtraction Counter 2 Phase Pulse Input: Addition/Subtraction by Pulse Phase Change? (Encoder)

Function	Internal/External Preset, Latch Counter, Compare Output, No. of Rotation per Unit Time
Positioning	
Control Axis/Method	2 Axis, Position/Speed Control
Control	Pulse, Positioning Data: 80 Data/Axis (Operation Step No. 1–80)
Operation	End/Keep/Continuous, Single, Repeated Operation
Positioning Method	Absolute/Incremental, Address Range: – 2,147,483,648 to 2,147,483,647
Speed	Max. 100kpps (Setting Range 1– 100,000pps)
Acceleration/Deceleration	Trapezoidal Method
Additional Functions	Inching Operation, Speed Synchronizing, Position Synchronizing, Linear Interpolation
Pulse Catch	10µs 2 point (P0000–P0001), 50µs 6 point (P0002–P0007)

Input Filtering Time Setting Range	1, 3, 5, 10, 20, 70, 100ms
External Interrupt	10?: 2 Points (P00000–P00001)
50µs:6 Points (P00002–P00007)	
RTC	Option Board Required

Table 4:performance specification

Since there are 0 analog input or output pins present on the PLC, 4 XBO-AD02A and 2 XBO-AD02A extension modules were used, and as a single PLC can hold up to two modules only it was inevitable to use three PLCs. The communication protocols will later be discussed

Number of Channels	2
Range Voltage	0 to 10VDC (Input Resistance: 1MΩ minimum)
Current	4 to 20mA, 0 to 20mA (Input Resistance 250Ω)
Signal Resolution	12 bit
Unsigned Value	0 to 4000
Signed Value	-2000 to 2000
Precise Value Voltage	0 to 1000 (0 to 10VDC)
Current	400 to 2000 (4-20mA), 0 to 2000 (0 to 20mA)
Percentile Value	0 to 1000

Maximum Resolution Voltage	2.5mV (0 to 10VDC)
Current	5µA (0 to 20mA), 6.25µA (4 to 20mA)
Accuracy	±1% maximum
Maximum Conversion Speed	1ms / Channel + Scan Time
Maximum Absolute Input Voltage	DC±15V
Current	DC±25mA
Average Function	Count Average (2 to 64,000 times)
Gain Adjustment Function	-40 to 40
Insulation Method	None

Table 5:XBO-AD02A Module Specs

Number of Channels	2
Range Voltage	0 to 10VDC (Input Resistance: 1MΩ minimum)
Current	4 to 20mA, 0 to 20mA (Load 450Ω or less)
Setting	In User Program or I/O Parameter per Channel
Signal Resolution	12 bits
Unsigned Value	0 to 4000
Signed Value	-2000 to 2000
Precise Value Voltage	0 to 1000 (0 to 10VDC)
Current	400 to 2000 (4-20mA), 0 to 2000 (0 to

	20mA)
Maximum Resolution Voltage	2.5mV (0 to 10VDC)
Current	5µA (0 to 20mA), 6.25µA (4 to 20mA)
Accuracy	±1% or less
Maximum Conversion Speed	1ms / Channel + Scan Time
Insulation Method	None

Table 6:XBO-DA02A Module Specs

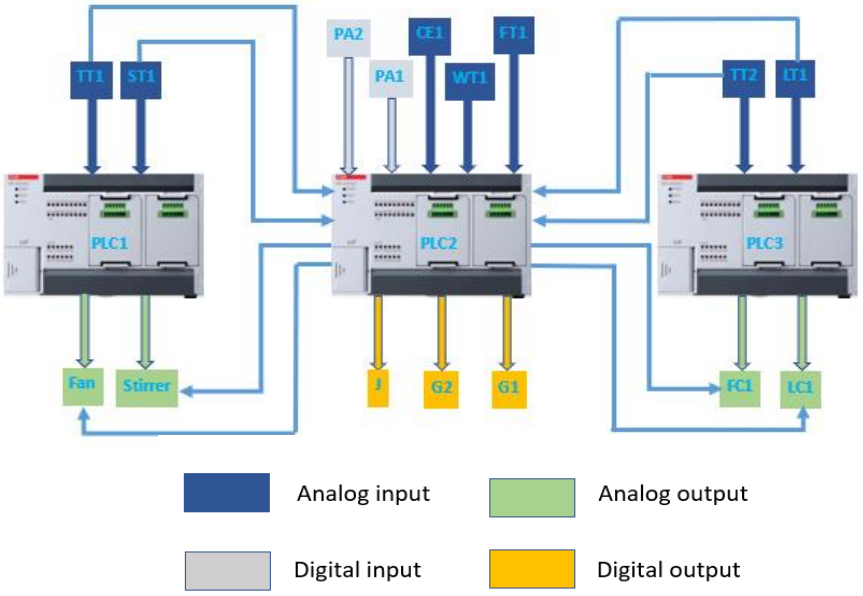


Figure 10:PLCs Inputs &Outputs

Hint: Figure 10 shows the connection between three PLCs and the illustration of how the data transfer between them will explained in detail in the communication chapter.

Sensor symbol	Sensor/Actuator	Sensor name	PLC type	PLC No.	Status
PLC1					
TT1	**Sensor**	Temp. Sensor 1	Analog	1	In
ST1	**Sensor**	Taco meter for fan	Analog	1	In
Fan	**Actuator**	Fan	Analog	1	out
Stirrer	**Actuator**	Stirrer	Analog	1	out
PLC2					
PA1	**Sensor**	Pressure sensor 1	Digital	2	In
PA2	**Sensor**	Pressure sensor 2	Digital	2	In
FT1	**Sensor**	Flow sensor	Analog	2	In
WT1	**Sensor**	Two load cells	Analog	2	In
CE1	**Sensor**	Conductivity indicator	Analog	2	in
G1	**Actuator**	Pump 1 for large tank	Digital	2	Out
G2	**Actuator**	Pump 2 for small tank	Digital	2	Out
J1	**Actuator**	Electrical resistance	Digital	2	Out
PLC3					

TT2	**Sensor**	Temp. sensor 2	Analog	3	In
LT1	**Sensor**	Differential pressure	Analog	3	In
FC1	**Actuator**	Control valve (down)	Analog	3	Out
LC1	**Actuator**	Control valve(up)	Analog	3	out

Table 7: Name of abbreviations for each sensor & actuator

PLC 1							
MODULE 9 (INPUT)				MODULE 10 (OUTPUT)			
Channel 0		Channel 1		Channel 0		Channel 1	
TT1		ST1		Fan (TC1,SC1)		stirrer	
13	14	19	20	17	18	----------	----------
PLC 2							
MODULE 9 (INPUT)				MODULE 10 (INPUT)			
Channel 0		Channel 1		Channel 0		Channel 1	
FT1		WT1		CE1		----------	

1	2	15	16	9	10		
PLC 3							
MODULE 9 (INPUT)				MODULE 10 (OUTPUT)			
Channel 0		Channel 1		Channel 0		Channel 1	
TT2		LT1		FC1		LC1	
11	12	3	4	5	6	7	8

Table 8: analog connections

PLC 1			
No Digital Connections.			
PLC 2			
Inputs:			
Sensor/actuator	PORT	(+) labeling	(-)
PA1	P03	E4P	28
PA2	P04	E4N	31
Outputs:			
G2	P40	23	21
G1	P41	22	21
J1	P42	24	21
PLC 3			
No Digital Connections.			

Table 9:Digital connections

3. PROGRAMMING SOFTWARE

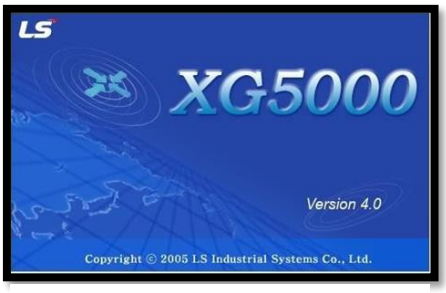


Figure 11:XG500

LS Electric PLCs work with the XG5000 software tool for programming either using ladder diagrams or another language. The XG5000 was also developed by LS Electric to program and troubleshoot the XGT PLC series and has a user-friendly interface for creating, editing, and debugging ladder logic programs. It supports the following programming languages:

- Ladder Diagram (LD)
- Instruction List (IL)
- Function Block Diagram (FBD)
- Structured Text (ST)

XG5000 also features a simulation mode and supports online monitoring and debugging of running programs.

PLC1			
Read		**Write From PLC 2**	
TT1	D03	Fan	M00
ST1	D04	Stirrer	M01
PLC2			
Read		**Write**	
WT1	D14	**To PLC 1**	
FT1	D08	Fan	D09
CE1	D02	Stirrer	D10
PA1	P03	**To PLC 3**	
PA2	P04	FC1	D11
G1	P41		

G2	P40		
J	P42		
From PLC 1		LC1	D12
TT1	M00		
ST1	M01		
From PLC 3			
TT2	M02		
LT1	M03		
PLC3			
Read		**Write**	
TT2	D01	**Write From PLC 2**	
LT1	D09	FC1	M00
		LC1	M01

Table 10: connection between 3 PLCs

Memory	Function
M20	change the mode from automatic to manual
M15	coil indication for modes
M21	manual control for G1
M22	manual control for G2
M23	manual control for J
M24	manual control for fan
M25	manual control for stirrer
M26	Valve1 FC1
M27	Valve 2LC1

Table 11:Manual control for hmi

Memory	Function
M05	change the mode from automatic to manual
M06	Manual_website_ control for G1
M07	Manual_website_control for G2
M08	Manual_website_ control for J
M09	Manual_website_control for fan
M10	Manual_website_ control for stirrer
M11	Manual_website_control for Valve_1(FC1)
M12	Manual_website_control for Valve_2(LC1)

Table 12:Manual control from website

4. Process

5. objective:

   Take samples at different temperatures (40,35,30)

6. challenge:

   The system hasn’t an output source, so the valve used to take the output.

7. Logic of process:

   Steps

- Temperature sensor (TT2) <40℃
  - Turn on heater and stirrer.
- Temperature sensor (TT2) >=40℃
  - Turn off the heater and stirrer
  - Turn on valve2 (FC1) and pump1
- Loadcell sensor (WT1)>=(9L=7.2kg)
  - Turn off valve (FC1) and pump1
  - Turn on valve2 (LC1) and TON Timer
- Timer=2sec
  - Turn off LC1 and turn on pump 2, and fan.
- Load cell< 2L
  - Turn off pump2 and fan
- Then return the process and take second sample at 35℃ after cooling
- Then return the process and take second sample at 30℃ after cooling

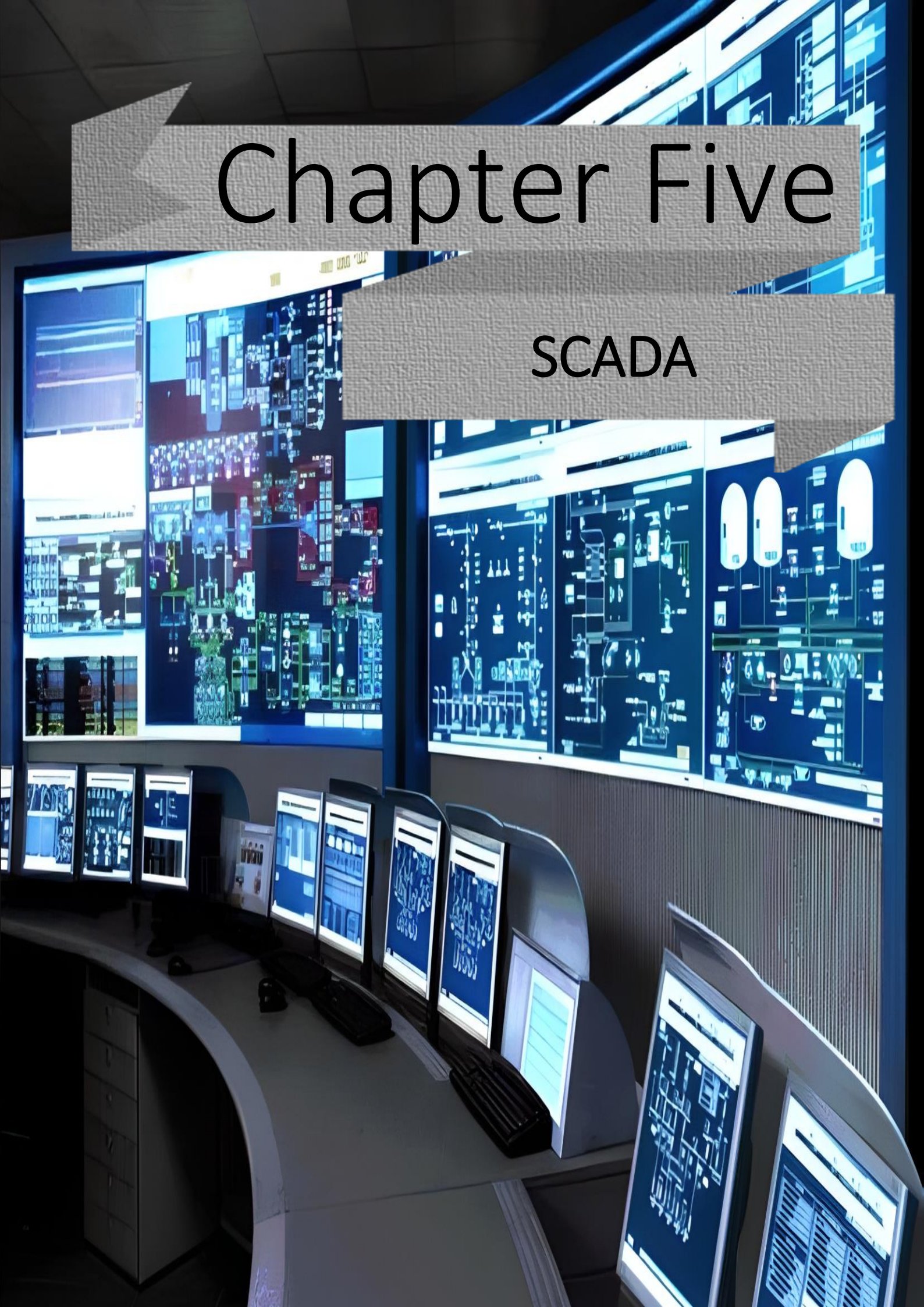
5. SIMULATION

After finishing the ladder diagrams for the 3 PLCs, a simulation was done on XG5000 to ensure the system is working properly. After the simulation, the ladder diagram was modified a bit to reach the best performance

The video in the attached QR shows the simulation process using the updated ladder diagrams.

Chapter Five

SCADA



SCADA, which stands for Supervisory Control and Data Acquisition, is a comprehensive control system architecture used in various industries for monitoring, controlling, and managing complex processes and facilities. It provides a centralized platform that enables real-time data acquisition, visualization, and control of industrial processes.

1. Key components and features of SCADA system

- Supervisory Control: SCADA allows operators and supervisors to monitor and control industrial processes remotely from a central location. This supervision involves overseeing the performance of machinery, processes, and other relevant parameters.

- Data Acquisition: SCADA systems collect data from sensors, instruments, and devices distributed throughout the industrial environment. This data includes information on variables such as temperature, pressure, flow rate, and more.

- Human-Machine Interface (HMI): SCADA systems typically include an HMI that provides a graphical representation of the industrial processes. The HMI allows operators to interact with the system, view real-time data, and respond to alarms or events.

- Control Logic: SCADA systems implement control logic to regulate and automate processes based on the acquired data. This ensures that industrial systems operate efficiently and within specified parameters.

- Communication: SCADA systems use various communication protocols to establish connections between the central system and remote devices or field equipment. This enables the exchange of data and control commands.

- Alarming and Event Logging: SCADA systems monitor for abnormal conditions and trigger alarms when predefined thresholds are exceeded. They also maintain event logs for recording important occurrences and system events.

- Security: Given the critical nature of industrial processes, SCADA systems incorporate security measures to protect against unauthorized access, cyber threats, and ensure the integrity of the data and control commands.

2. XP-Builder

XP-Builder is software that allows you to create and manage projects for machine control devices. You can use XP-Builder to create projects for the XGT Panel. XP-Builder includes multiple features that allow you to design and edit projects conveniently, such as:

- Customizable toolbars and hotkeys
- Customizable tool, project, and editing panes
- Functions to import and export common data
- Tabs for viewing multiple screens easily

- Previews of project screens
- Customizable image and object libraries
- Scripts and advanced functions, such as alarms, and logs.
- Support for multiple languages

3. Building the model

The Scada model was built in XP-builder software as it is an open source software and the next screens explain the steps of building this model:

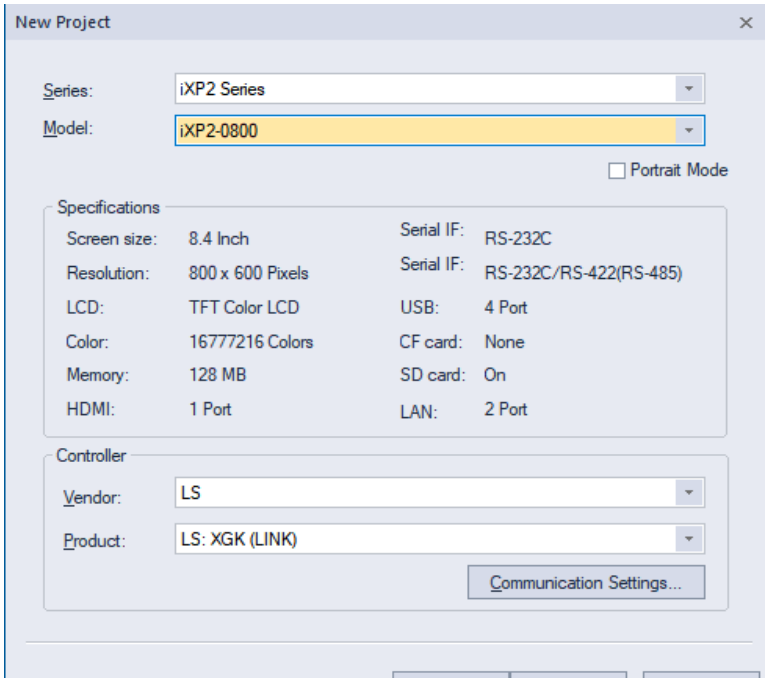


Figure 12:initialization of building Scada model

8. Home screen

This screen contains the buttons that allow us to navigate to the next screens which is: 1-Automatic control 2-Manual control 3-Alarms

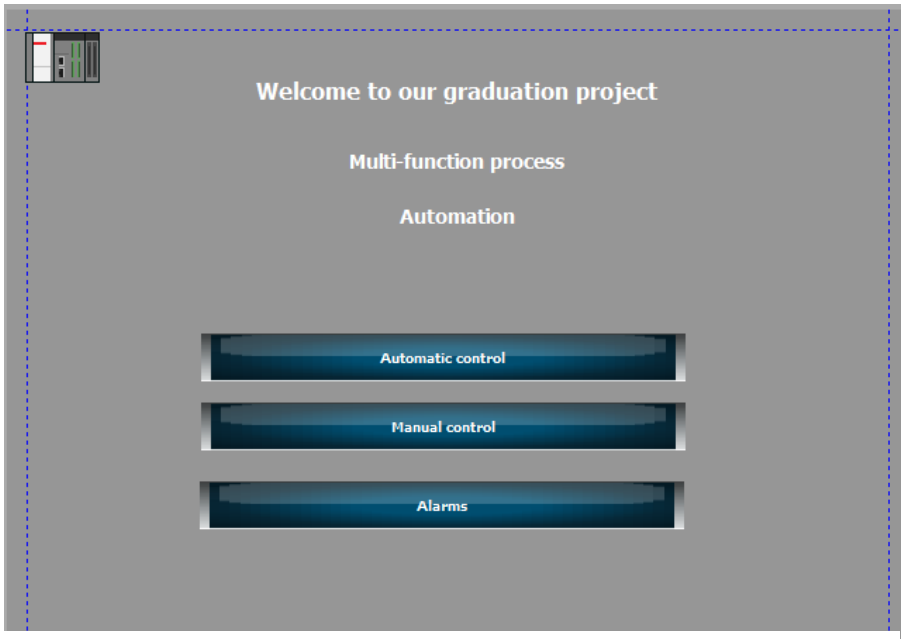


Figure 13:Home screen

9. Automatic control screen

This screen displays all system sensors and actuators, and it is accessed when the automatic process is applied

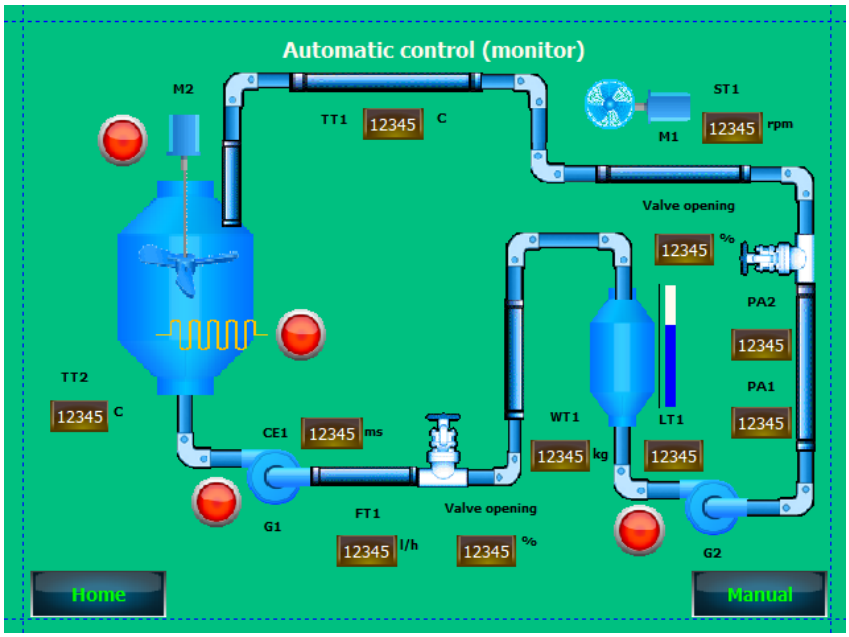


Figure 14:Automatic screen

10.Manual control screen

This screen displays all system sensors and actuators. You access this screen when manual control of the system is necessary. To activate this mode, press the 'Manual Control' button at the bottom screen, thereby switching from automatic control to manual control. This option is designed to facilitate system testing during failures or maintenance needs. In manual control mode, you have the ability to manually adjust and control:

- Pump 1
- Pump 2
- Valve 1
- valve 2
- Heater
- Stirrer
- Fan

Figure 15:Manual mode

11. Alarm screen

We create alarms to safeguard our system from failures and enhance its performance by tracking the frequency of issues. After designing the alarm screen, a table appears at the bottom screen, comprising:

1. Occurrence Time
    - Definition: The occurrence time signifies when a specific alarm condition was initially detected or occurred within the system.
    - Usage: It provides a timestamp for when the alarm event was first identified, aiding in chronological analysis of system events.
2. Message
    - Definition: The message in the alarm table typically contains a brief description or information about the specific alarm condition or event.
    - Usage: It helps operators and users understand the nature of the alarm, facilitating quick and informed decision-making.
3. Group
    - Definition: The group in the alarm table categorizes or groups related alarms together based on common characteristics or causes.
    - Usage: Grouping alarms helps in organizing and prioritizing responses, especially when dealing with multiple alarms simultaneously.
4. Frequency
    - Definition: The frequency in the alarm table indicates how often a particular alarm condition has occurred over a specific period.
    - Usage: It provides insights into the recurring nature of specific issues, aiding in preventive maintenance and system optimization.

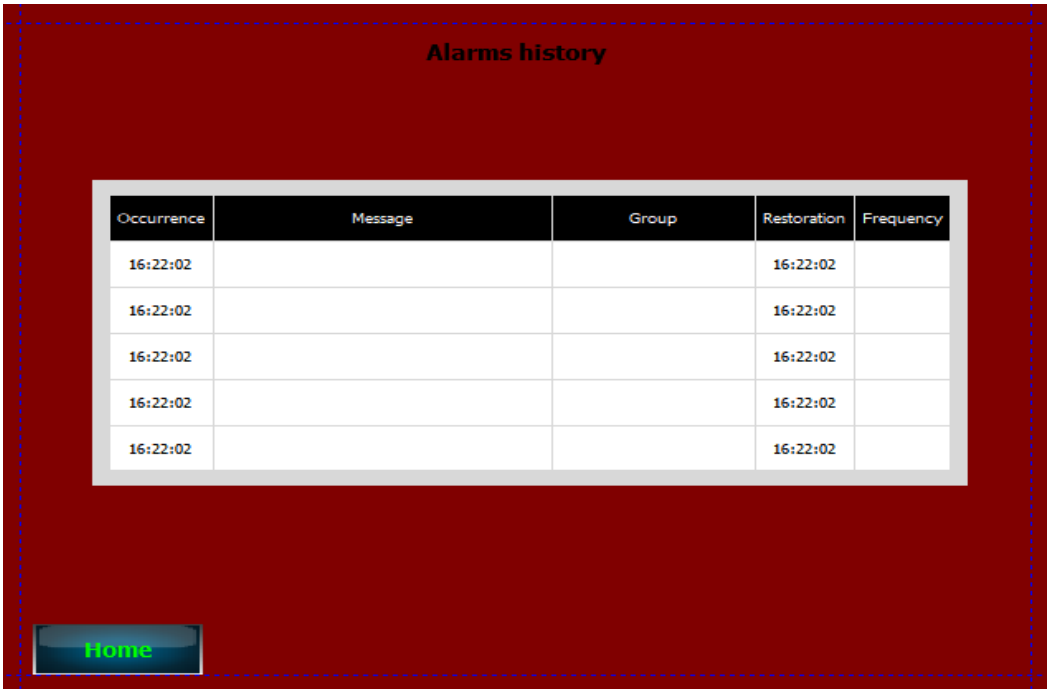
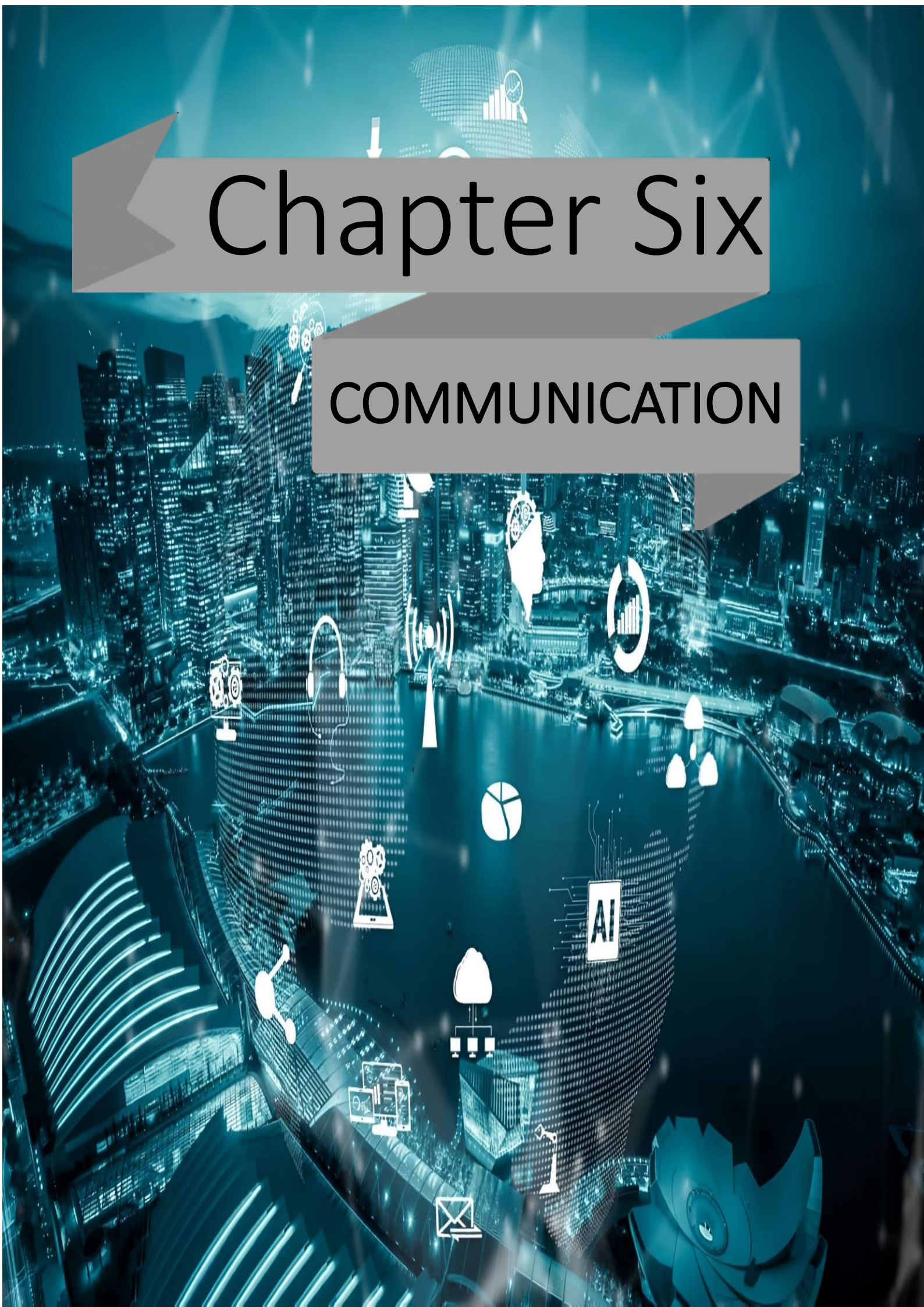


Figure 16:Alarm screen

Memory	Function
M20	change the mode from automatic to manual
M15	coil indication for modes
M21	manual control for G1
M22	manual control for G2
M23	manual control for J
M24	manual control for fan
M25	manual control for stirrer
M26	Valve1 FC1
M27	Valve 2LC1

Table 13:Tags between PLC and XP-Builder

4. Test simulation


Chapter Six
COMMUNICATION
AI

we would remember the challenges that face us for make communication among the plcs with each other and form another side, the system transfers their data to cloud through the used microcontroller for building internet connect with cloud is called “ESP32”, and then we would move to the accurate hardware and connection and diagram for make this communication.

1.Challenges:

- Current Mirror Issue
- No Synchronoss
- No Mastering

1.Current Mirror:

1. Introduction:

For the PLC and the IIoT to be used simultaneously the output current signal of all the sensors should be passed to both the PLC and the controller of the IIoT. Hence, there need to be two copies of each signal. The current mirroring circuit is used to copy the flow of current in one active device and control the flow of current in another device by maintaining the output current stable instead of loading. Theoretically, a perfect current mirror is an inverting current amplifier. The main function of this amplifier is to reverse the direction of the flow of current. The main function of the current mirror is to provide active loads as well as bias currents to circuits and is also used to form a more practical current source.

Several methods are available to implement a current mirror circuit. Two methods were tried to implement the circuit

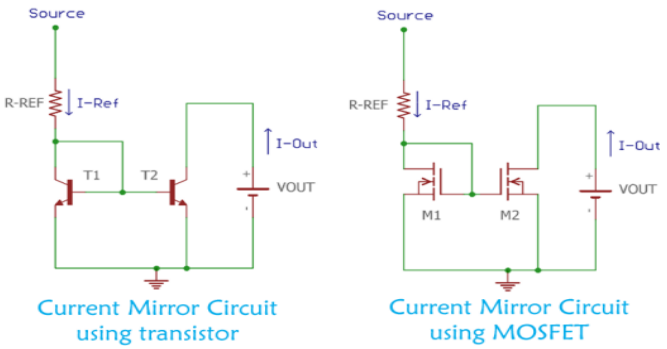


Figure 17:Current Mirror

2. USING MOSFETs:

There are 4 MOSFETs in total in the circuit, two P-type and two N-type having similar properties. Here's a brief explanation of how it works:

In a basic MOSFET current mirror circuit, the input current is applied to the gate of a MOSFET, called the input or reference MOSFET. The drain of the input MOSFET is connected to the gate of a second MOSFET called the output MOSFET. The drain of the output MOSFET is connected to a current source or load resistor.

When a voltage is applied to the gate of the reference MOSFET, it establishes a current flowing through the MOSFET from the drain to the source. This current is mirrored in the output MOSFET and flows through the drain and source of the output MOSFET. The ratio of the two currents is determined by the size (W/L ratio) of the two MOSFETs and their operating conditions.

By adjusting the size and bias of the MOSFETs, the current mirror can be made to operate in a variety of configurations, such as cascade, folded cascade, or improved output impedance.

The simulation is done to check the output and the error for both 4 and 20 mA input and the results are the following:

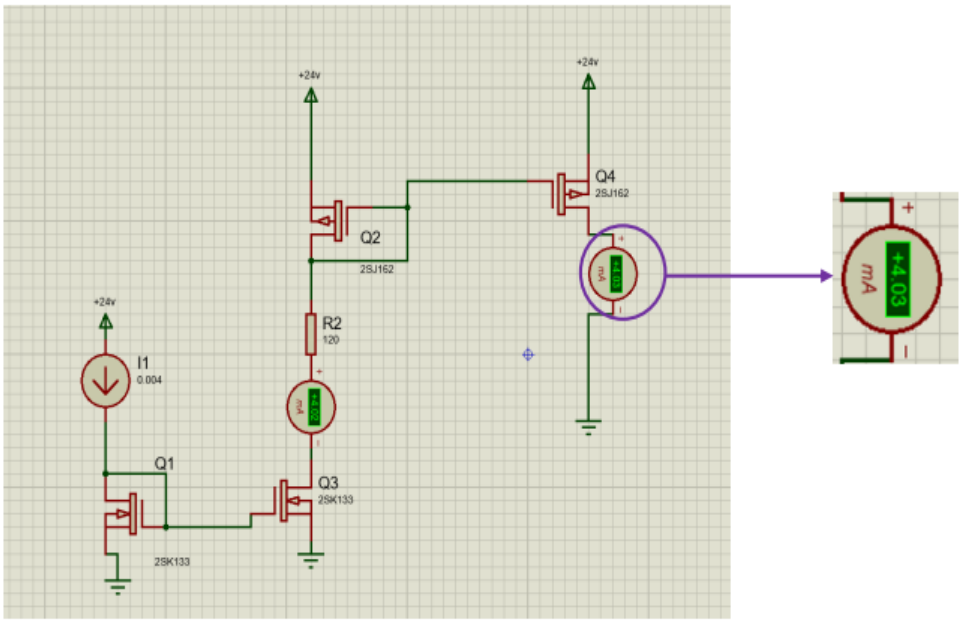


Figure 18:Current Mirror Connections 1

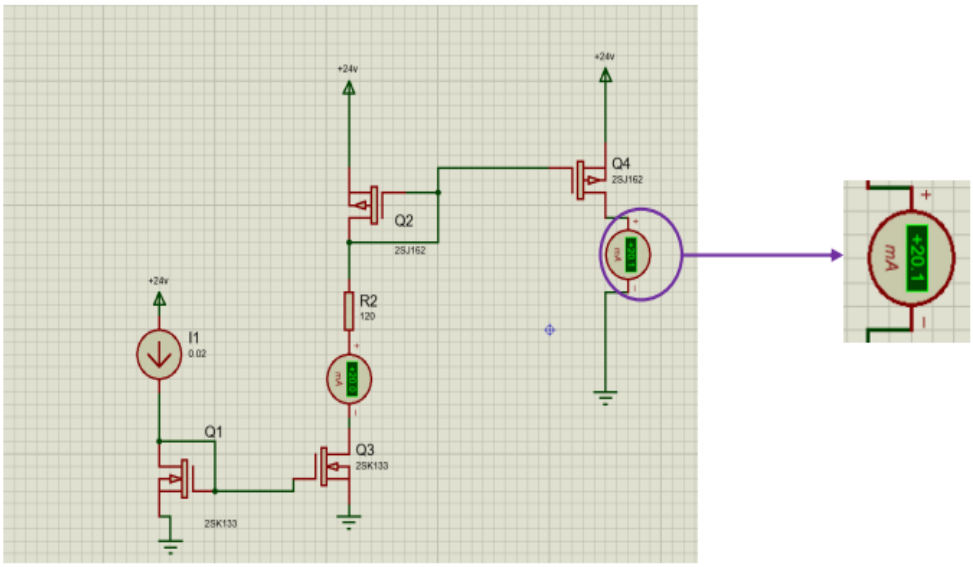


Figure 19:Current Mirror Connection 2

3. Issues Encountered:

- The shortage in the P-type MOSFETs in the market.
- Wide variety in prices.
- Hard to find P-type and N-type MOSFETs with similar characteristics.

The Previous issues made the signals that would esp32 receive them be noisy and distorted and thus the plcs and then this method was not very efficient in communication process.

2.No Synchronous & No Mastering:

1. .Introduction

The current mirror circuit is used to make the Esp32 take the signals of sensors without there is mastering or controlling by PLCs. In this case, the esp32 is used only for transferring the data from the system to the cloud without taking action from the cloud to control and manage the system.

While the Plc receives data and takes action regardless of whether the manager wants to take action as an alarm action or not, then the problems of synchronization and mastering must be taken into consideration for accurate communication.

Brainstorming

Now we find the current mirror and making esp32 is separated of the plcs in transferring data would make the communication is inefficient, then we thought that to solve these problems we make master control in all processes from first the communication between PLCs getting to communication with the Cloud.

the solution that we found out that would make ESP32 Master, PLCs Slaves would be explained in detail in the part Esp32 Master, but before speaking about the Mastering of ESP32, let us speak more about the types of available protocols that plcs have and what would be the best types of protocols for applying this idea.

2.PLC Protocols:

- Ethernet/IP: An advanced version of standard Ethernet, developed by Rockwell Automation.
- Profibus: A protocol owned by Siemens Automation.
- Modbus: A protocol used for transmitting information over serial lines [Modbus RTU] or Ethernet [Modbus Tcp], developed by Mod icon (now Schneider Electric).

1. Ethernet/IP

-In This Part from Topic, we thought to make the plc Is Master and synchroniser through Communicating with cloud directly without needing to any helper as esp32.

XGB Fast Enet I/F module supports open Enet. It provides network configuration that connects LSIS and other company PLC, PC on network.

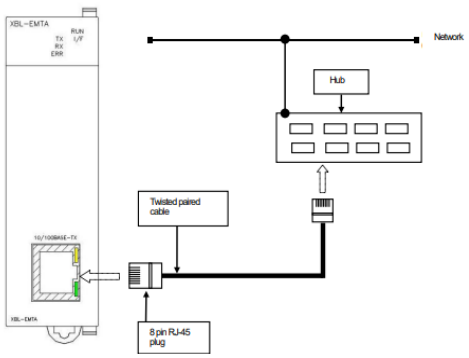


Figure 20:Ethernet/Ip

1. Examples of System Configuration:

1.COMBINATION NETWORK CONFIGURATION:

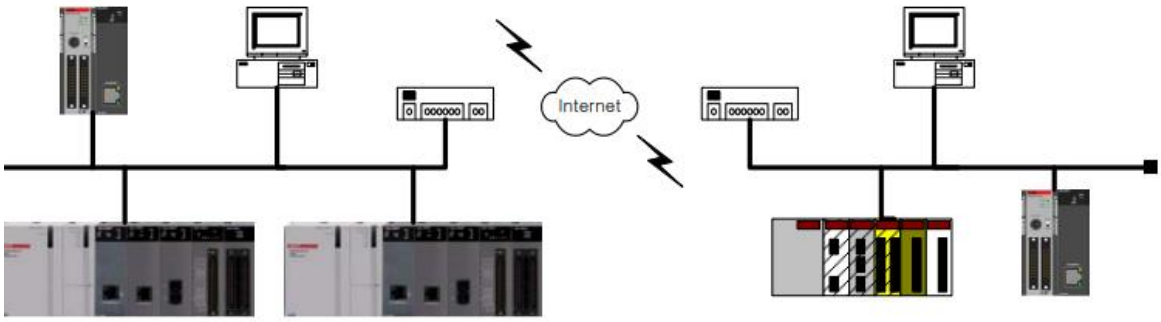


Figure 21:Combination Network configuration

XGB Fast Enet I/F module provides system configuration by using main communication, Modbus TCP/IP, user define frame, HS link communication connecting LSIS PLC with other LSIS PLC, PC on network.

2. Network configuration using XGB

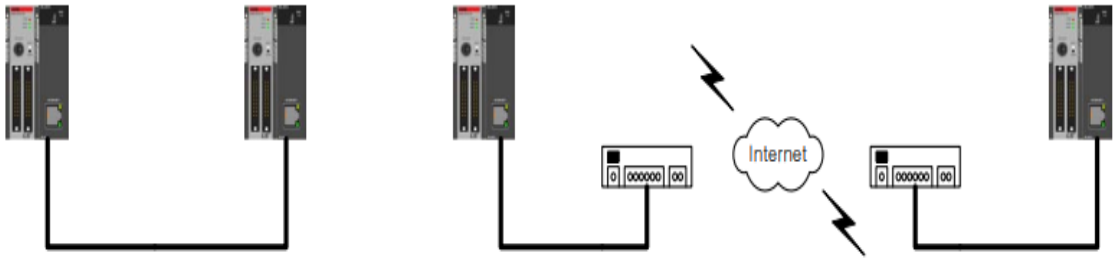


Figure 22:Network Configuration XGB Driver

Communication between XGB Fast Enet I/F modules is available to perform 1:1 communication by using cross cable or 1: N communication by connecting network. It provides data sending/receiving by using the dedicated service, Modbus TCP/IP, user define frame and HS link communication.

And there are many examples that certain the high ability of plc for communicating by Ethernet Protocol, but the encountered issue is that LSIS cannot handle the signals with web through http protocols, so we moved to the Modbus protocol that is the most famous in Automation World to be the best solution for our mission.

2. Modbus Protocol:

1. Introduction

Modbus protocol is specified open protocol used between client-server, which executes reading/writing data according to function code. Communication between devices that use Modbus protocol uses Client-server function in which only one client processes the data.

2. Kind of Modbus protocol

There are two communication modes of Modbus, ASCII and RTU.

Characteristic		ASCII mode	RTU mode
Coding method		ASCII code	8 bit binary code
No. of data per one character	Start bit	1	1
	Data bit	7	8
	Parity bit	Even,Odd,None	Even,Odd,None
	Stop bit	1 or 2	1 or 2
Error check		LRC(Longitudinal Redundancy Check)	CRC (Cyclical Redundancy Check)
Start of frame		Colon (:)	3.5 Character no response time

Table 14:Types of Modbus

3. Data and expression of address

To express data and address of Modbus protocol, the characteristic is as follows. (1) It used hexadecimal as basic form. (2) In the ASCII mode, Hex data is converted into ASCII code. (3) RTU mode uses Hex data. (4) Each function code has the following meaning.

Code(Hex)	Purpose	Used area	address	Max. response data
01	Read Coil Status	Bit output	0XXXX	2000bit
02	Read Input Status	Bit input	1XXXX	2000bit
03	Read Holding Registers	Word output	4XXXX	125word
04	Read Input Registers	Word input	3XXXX	125word
05	Force Single Coil	Bit output	0XXXX	1bit
06	Preset Single Register	Word output	4XXXX	1word
0F	Force Multiple Coils	Bit output	0XXXX	1968bit
10	Preset Multiple Registers	Word output	4XXXX	120word

Table 15: expression of address

- and Because of Modbus RTU is often considered better than Modbus ASCII for several reasons:

Efficiency: Modbus RTU uses binary encoding, which allows for more compact and efficient data transmission compared to Modbus ASCII. The binary nature of Modbus RTU reduces the transmission overhead, enabling faster and more efficient communication.

Transmission Speed: Due to its binary encoding and simpler frame structure, Modbus RTU can achieve higher transmission speeds compared to Modbus ASCII. This is particularly beneficial in applications that require fast and time-sensitive communication.

Compatibility: Modbus RTU enjoys broader support and compatibility with devices and software applications compared to Modbus ASCII. Many industrial devices and communication modules are specifically designed to work with Modbus RTU, making it a more practical choice in most industrial automation environments.

Error Checking: Both Modbus RTU and Modbus ASCII include error checking mechanisms, such as CRC (Cyclical Redundancy Check). However, the binary nature of Modbus RTU allows for more robust error checking, resulting in better data integrity and error detection capabilities.

Noise Immunity: Modbus RTU, being a binary-based protocol, is more resistant to noise and interference compared to Modbus ASCII. The binary nature of Modbus RTU signals allows for better noise immunity, making it suitable for industrial environments with high levels of electrical noise.

Implementation Simplicity: Modbus RTU has a simpler frame structure compared to Modbus ASCII, which includes additional character framing. This simplicity makes it easier to implement and troubleshoot Modbus RTU-based systems.

Performance: Due to its efficiency, higher transmission speed, and better error checking, Modbus RTU generally offers better overall performance compared to Modbus ASCII.

So, we used the Modbus RTU for communicating among the plcs with each other and esp32 with them. but how we would make this idea.

We would make the esp32 master Modbus RTU while the PLCS would be the slaves Modbus RTU ,then the esp32 would have ability to request from any PLC and then PLC response to it ,in this way we can make synchronization and mastering in transferring data without any noisy or until distortion data and from the another side , esp32 would be bridge between the PLCS and Cloud ,this diagram is mentioned below is the description of the solution.

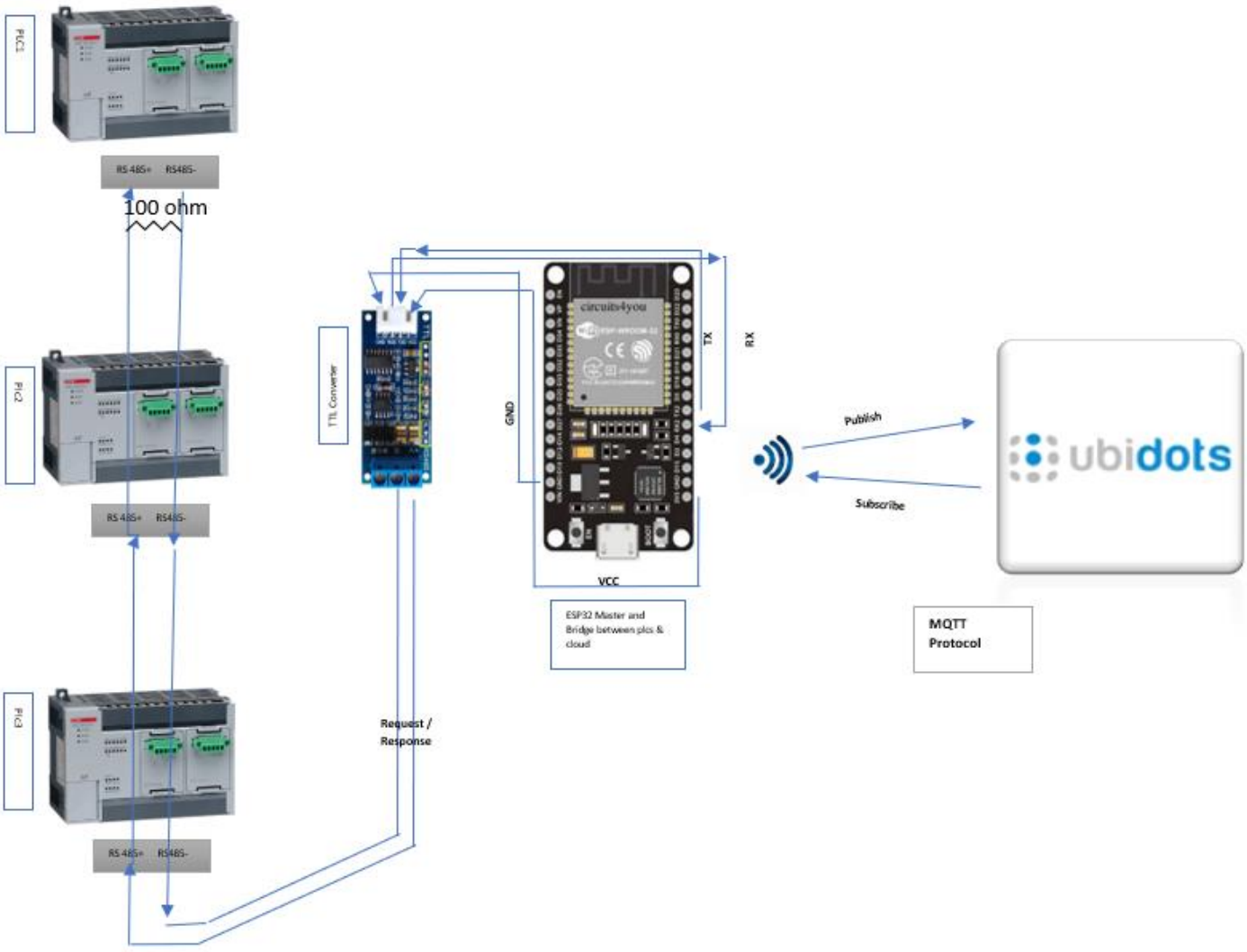


Figure 23:Communication Diagram

Now we will explain how to make PLCs as Slaves MODBUS RTU, ESP32 as Master Modbus RTU.

3. Slave Modbus RTU

First, Modbus RTU can communicate over several types of physical media, two of which are built-in the CPU module of the PLC model used in the project: RS232 and RS485. Modbus used to run on RS232 communication predominantly in the past but nowadays it is more commonly used with RS485.

1.RS485 vs RS232:

RS485 can be considered the more advanced version of RS232 as it solves most of the problems encountered using RS232.

RS232 / RS485 Comparison		
P.O.C	***RS232***	***RS485***
Speed	Slower (20 kb/s)	Faster (10 Mb/s)
Distance	50 ft.	4000 ft.
No. of devices	1	32
Noise	More susceptible (ground reference)	Less susceptible
Complexity	Simple	Complex
Mode of operation	Simplex or full duplex	Simplex or half/full duplex

Table 16:comparsion between RS232 and RS485

Due to the previous reasons and the fact that the project has three PLCs running simultaneously, RS485 seemed like the better solution for the process to run fast and efficiently.

2. Modbus memory mapping:

Modbus has four basic memory areas depending on the type and direction of data, they can be divided into the following:

Modbus Memory Areas	
Start Address	Description
1xxxxx	Bit read area
0xxxxx	Bit write area
3xxxxx	Word read area
4xxxxx	Word write area

Table 17:Modbus memory areas

3. Slave Modbus RTU

The Modbus settings need to be adjusted in each PLC to be slave through network configuration, where the PLC registers are mapped into Modbus addresses.

Each PLC also needs to have a unique station number and the baud rate need to be specified. The following are the Modbus settings of each PLC:

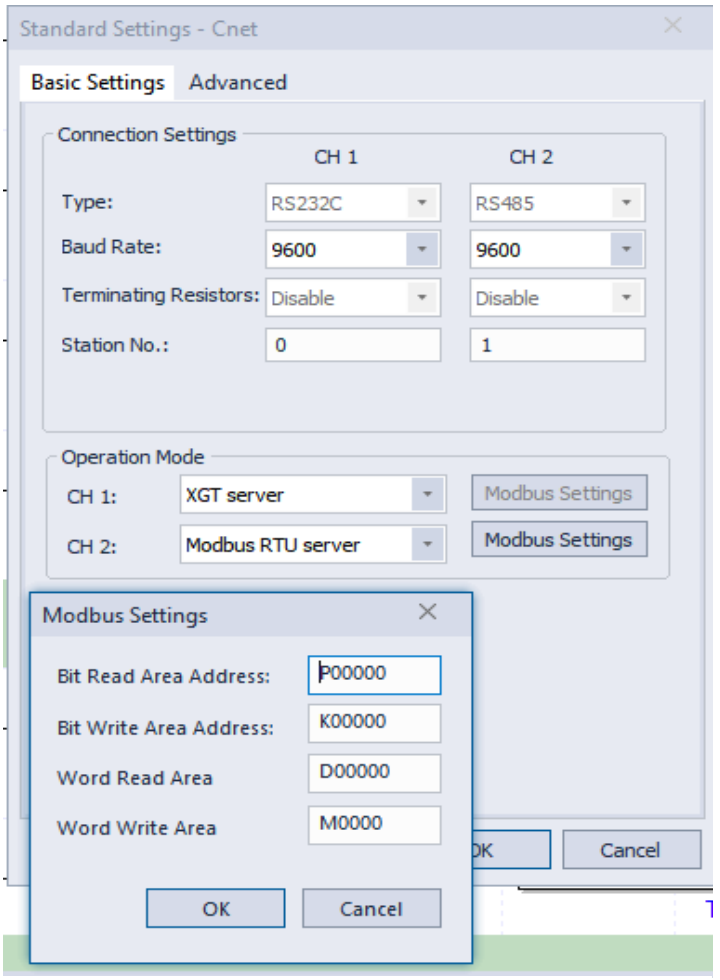


Figure 24: PLC1 Modbus Settings

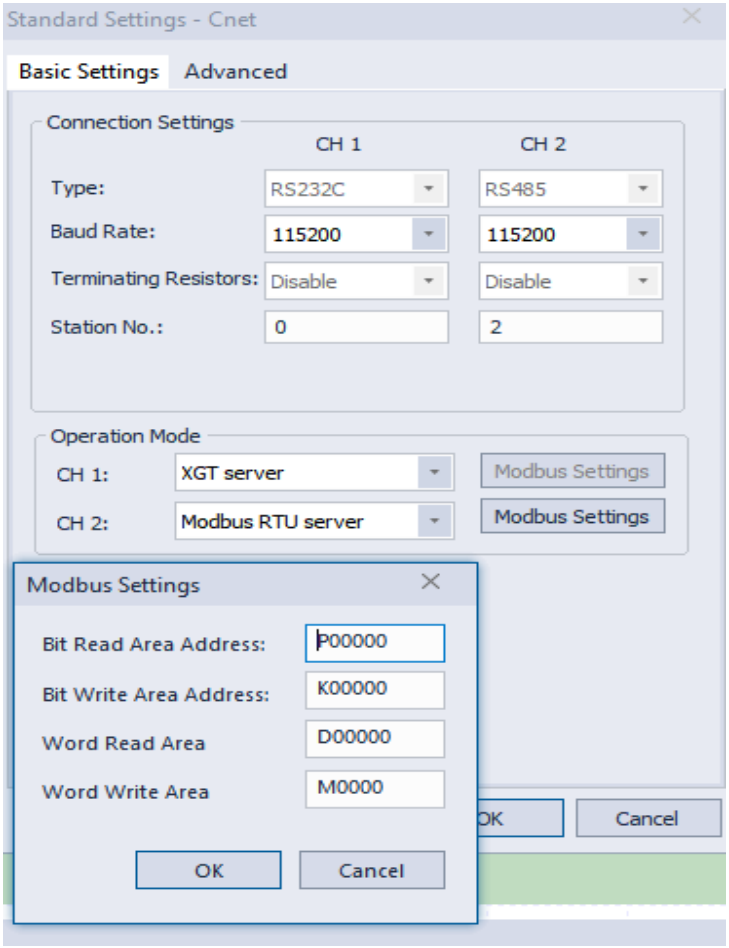


Figure 26:PLC2 Modbus Settings

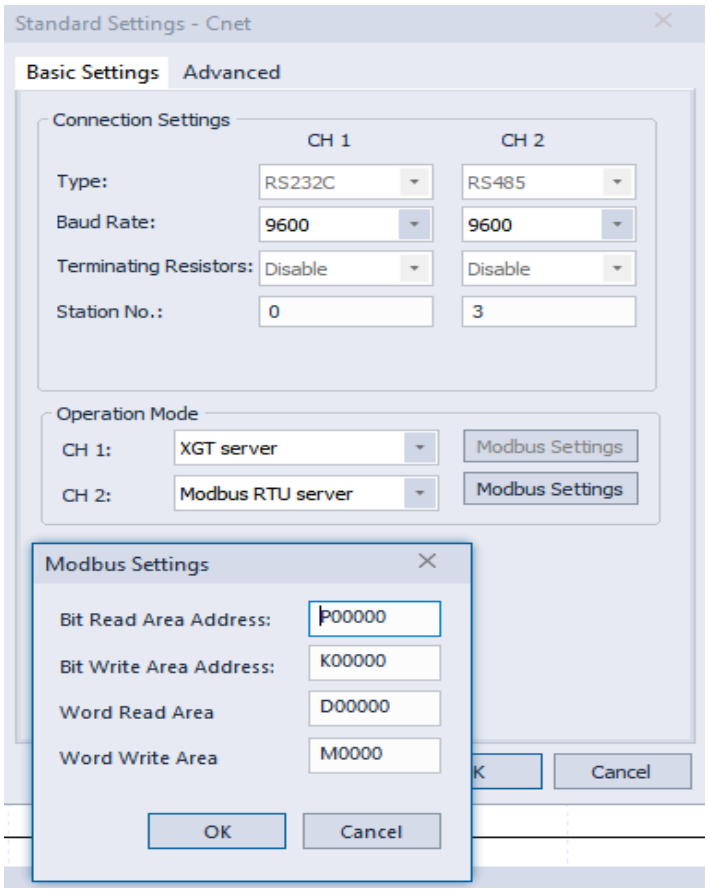


Figure 25:PLC3 Modbus Settings

4. Master Modbus RTU:

To make the esp32 master Modbus RTU that can handle data through serial communication RS 485, we needed to converter called “RS 485 TTL CONVERTER” That achieves this mission distinctively, now we would talk about TTL CONVERTER in some detail.

Figure 27:RS 485 TTL

1. Introduction:

This is a module that allows the TTL interface of the microcontroller to be transferred to the RS485 module. It is generally used for industrial automation.

This module adds lightning protection design and anti-jamming design. When we use it in the field and carry out long-distance transmission, we can Connect the GND end of the module to the ground so that lightning protection and anti-interference can be achieved.

It uses a standard 2.54 mm pitch design. If you want to make the second development will be more convenient. It has 120-ohm matching resistance, and shorted RO can, it is recommended that users in the long-distance transmission shorted.

It supports communication between multiple microcontrollers, allowing up to 128 devices on the bus. It can be hot-swappable, to prevent signal interference, it carries out a large area of copper, and the effect is very good.

2. Features:

1. 3.3V perfectly compatible with 5.0V power supply

2. Absolute using imported chips, industrial design, anti-interference ability, while using more effective mine design can be used in the industrial field and harsh field environments, the working temperature of -40 °C to + 85 °C transmission distance up to 1 kilometer (made with 850 meters of cable F1.5 tests recommended in the 800 meters, more than 800 m, please add repeaters).
3. Transmission distance up to kilometers (with 850 meters of 2* 1.5 cable to do the test, it is recommended to use in the 800 meters, more than 800 meters, please add repeaters)
4. Semi-hole process design, a thickness of 0.8mm, makes it easy to use as a combo board, it can also be welded for terminal use.
5. Has RXD, and TXD signal lights to observe send and receive status.
6. The module is fully considered mine design, long-distance transmission signal 485 in the wild, the module "Connected to the ground" terminal is connected to the earth, can play a very good anti-jamming and anti-lightning effect, more secure; it cannot connect with the ground when the indoor short-distance transmission.
7. Using the standard 2.54 pitch design, easy secondary development.
8. Has a 120-ohm termination resistor, shorted Roto enables the termination resistor.
9. Support multi-machine communication, allowing access to up to 128 devices on the bus.
10. The module can be hot-swapped, and the signal does not appear dead phenomenon.
11. A large area of copper to prevent interference.

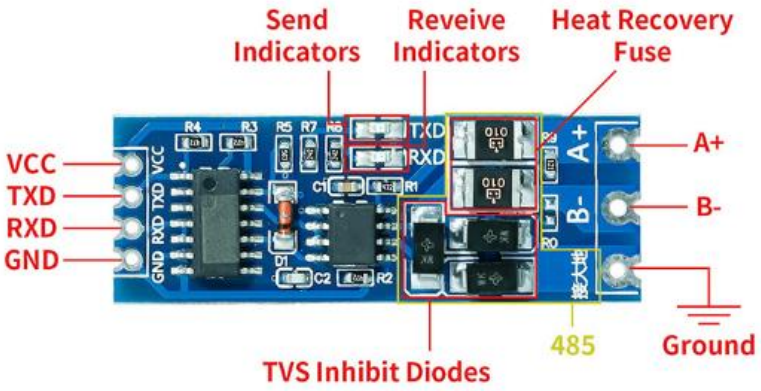


Figure 28:TTL CONVERTER DETAILED

Finally, we need to implement code on Arduino ide to make the esp32 master Modbus RTU which is achieved by a library called “<Modbus Master. H>”, while we need to make the esp32 send data to the cloud “ubidots” that would be explained in detail in “chapter seven” and this is achieved by library is called "UbidotsEsp32Mqtt.h" that handle with MQTT protocol is very efficient in internet of things.

5. Testing

Chapter Seven

IIOT

1. Platform

IoT and Cloud tools to drive digital transformation, Merge the physical and digital worlds into deployable, end-user ready Applications that deliver insights and solve problems.

Figure 29:ubidots platform

2. Device friendly API and SDKs

Connect hardware to Ubidots cloud easily with more than 200 user-proven libraries, SDKs, and tutorials to guide your integration over HTTP, MQTT, TCP, UDP, or by parsing custom/industrial protocols.

Whether onboarding 1 or 1,000 devices, it takes the same amount of effort with Ubidots Device Types. Replicate creating the new device in Ubidots automatically configuring variables, device properties, and appearance each time a new piece of hardware is detected.

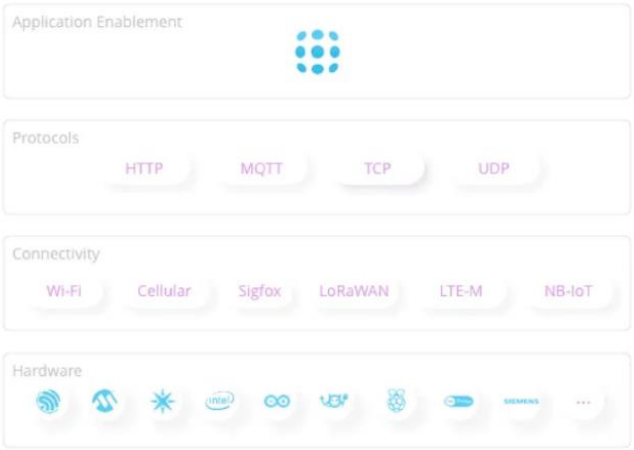


Figure 30:connection hardware to ubidots

3. Synthetic Variables

Transform raw data into insights with Synthetic Variables that compute complex math formulas and statistical expressions.

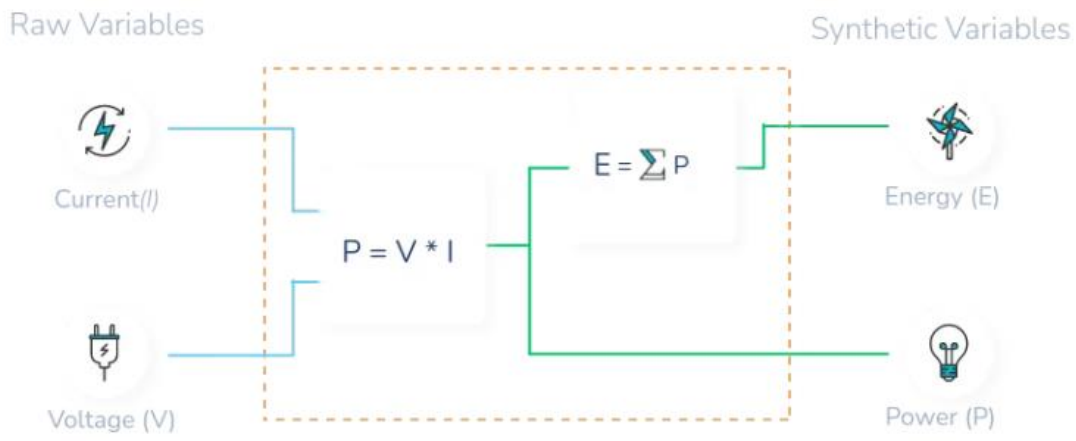


Figure 31:synthetic variables

4. Two-year time-series backend and storage

Whether your data is needed every day or every second, Ubidots time-series infrastructure is optimized to receive, compute, and return millions of data points each second across the globe and, with 2 years rolling data retention standard, Ubidots gives business managers and operation analysts a place to store and mine data for added insights and analytic overlays like anomaly detection or predictive maintenance.

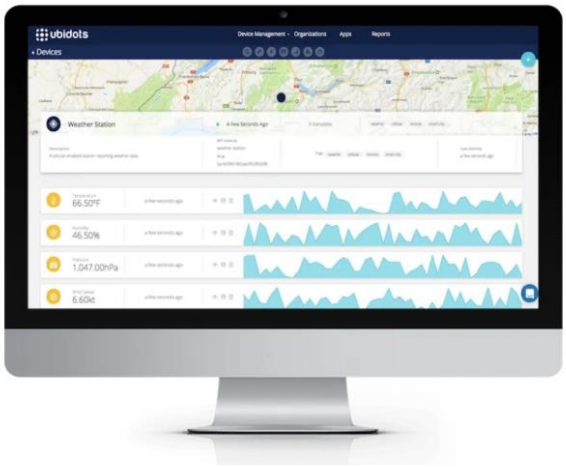


Figure 32:time -series backend and storage

5. Live Dashboard:

using Ubidots point-and-click application development tools, create real-time dashboards to analyze data and control devices.

Visualize data with Ubidots stock graphs, charts, tables, indicators, maps, metrics, and control widgets — or even develop your own using the HTML Canvas and your own code.

SHARE YOUR DATA THROUGH PUBLIC LINKS OR BY EMBEDDING DASHBOARDS OR WIDGETS INTO PRIVATE WEB AND MOBILE APPLICATIONS

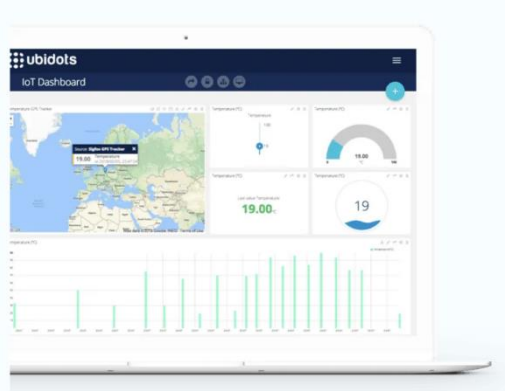


Figure 33:Live dashboard

6. Ubidots IoT

Ubidots gives decision makers the exact information needed for critical decisions in real-time, regardless of where the users are. Whether in the office, on the production floor, at home, or on the road, Ubidots keeps the users informed and empowered to consistently make data-driven decisions for improving production.

Today, data at your fingertips is the norm and Ubidots agrees. Access your applications from any Android device and stay informed when on the go.

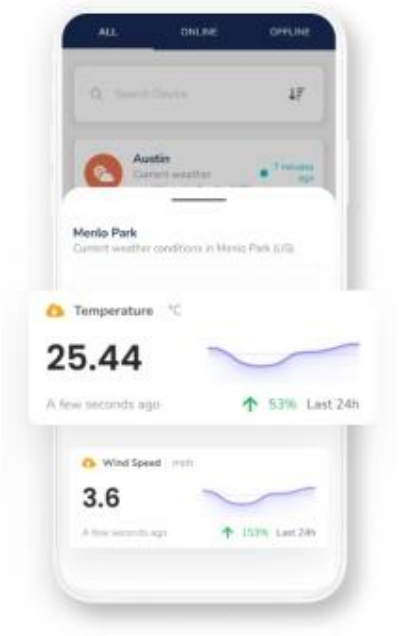


Figure 34:Ubidots iot

7. The Ubidots Advantage

1. **Global Reliability**

Scale from one device to 1,000s with confidence in Ubidots 99.9% uptime SLA made possible by automated failover and replication across IBMs global data centers

2. **Usability Comes First**

Intuitive interfaces and cloud development tools that make Ubidots a pleasure for developers, management, and operators to work with.

3. **Low Barriers to Entry**

Ubidots tools and pricing are designed from the bottom-up to help your business scale its cloud solutions from pilot projects to turn-key solutions.

8. Pricing:

Flexible pricing to launch and scale your IoT project

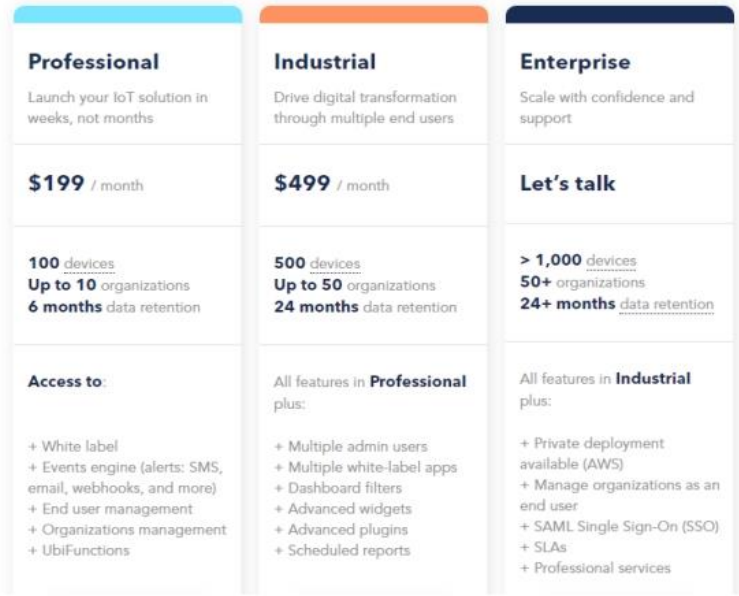
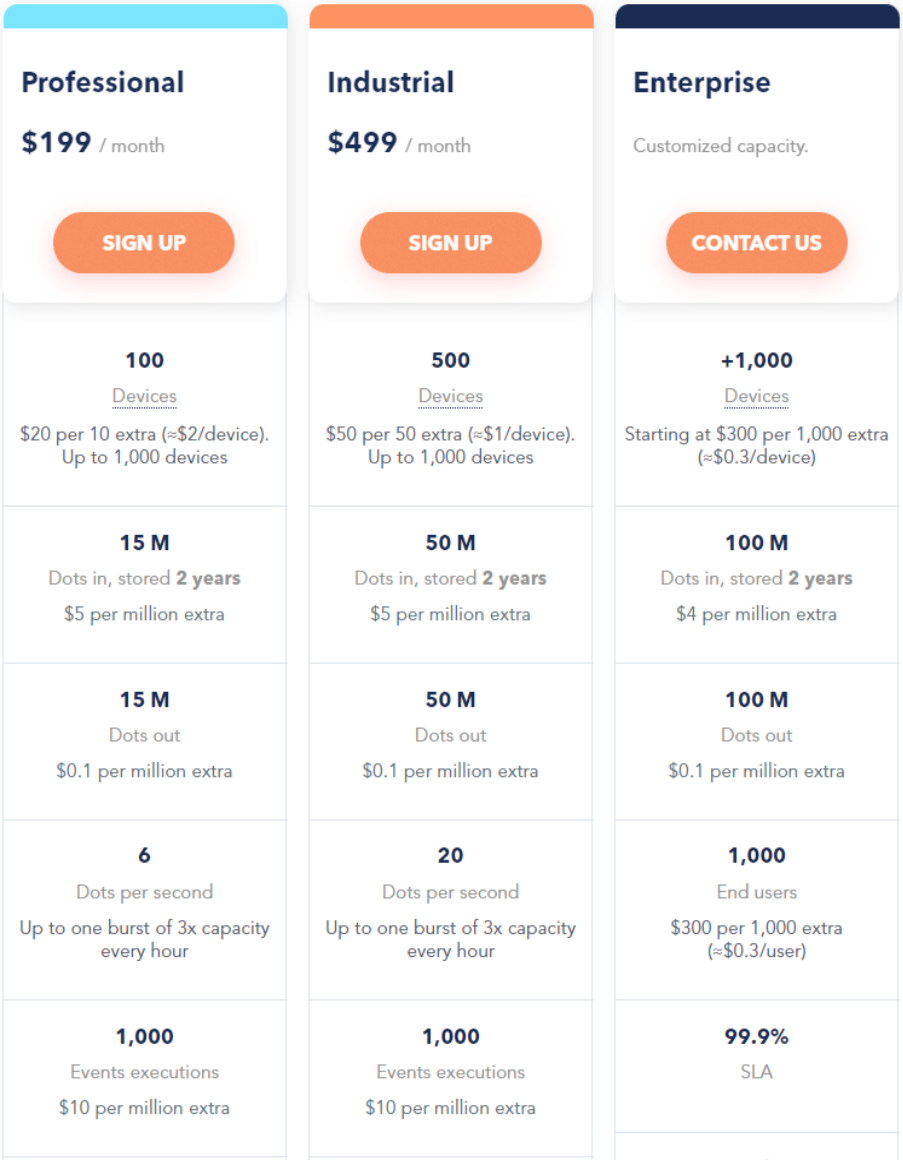


Figure 35:pricing of ubidots

1. Plan Capacity



Professional	Industrial	Enterprise
$199 / month	$499 / month	Customized capacity.
SIGN UP	SIGN UP	CONTACT US
100 Devices $20 per 10 extra (≈$2/device). Up to 1,000 devices	**500** Devices $50 per 50 extra (≈$1/device). Up to 1,000 devices	**+1,000** Devices Starting at $300 per 1,000 extra (≈$0.3/device)
15 M Dots in, stored **2 years** $5 per million extra	**50 M** Dots in, stored **2 years** $5 per million extra	**100 M** Dots in, stored **2 years** $4 per million extra
15 M Dots out $0.1 per million extra	**50 M** Dots out $0.1 per million extra	**100 M** Dots out $0.1 per million extra
6 Dots per second Up to one burst of 3x capacity every hour	**20** Dots per second Up to one burst of 3x capacity every hour	**1,000** End users $300 per 1,000 extra (≈$0.3/user)
1,000 Events executions $10 per million extra	**1,000** Events executions $10 per million extra	**99.9%** SLA

Figure 36:Plan capacity

100 Email and Telegram alerts $2 per thousand extra	**100** Email and Telegram alerts $2 per thousand extra	**24h** Guaranteed response time
10 SMS and Voice call alerts Pricing based on **receiving country**	**10** SMS and Voice call alerts Pricing based on **receiving country**	**1** Assigned Technical Account Manager
Up to 1 White-labeled app (Max. 1 App, 20 Organizations, and 20 End users)	**1+** White-labeled app $149 per extra white-labeled app $49 per extra gray-labeled app	**Enhanced security** Single sign-on Log in to Ubidots using the identity provider of your choice
Up to 20 Organizations	**Up to 100** Organizations	**Full customization** White-labeled API, API docs, and mobile app Offer your users a fully branded experience
1,000 UbiFunctions executions $5 per million extra	**1,000** UbiFunctions executions $5 per million extra	**Tailored solutions** Professional services Get purpose-built feature requests on top of the Ubidots platform
Up to 20 End users	**200** End users $25 per 50 extra	**Increased capacity** Provision the right capacity based on your expected amount of devices, variables, dots per second, events, organizations, and users
	10 Admin Users	

9. QR for ubidots website:

10. Upidots Objects

1. Sensors used in our process:
   - load_cell-(wt-1)
   - Flow_Sensor (FT-1)
   - Diff_Pressure (LT-1)
   - Taco_Meter (ST-1)
   - temp_sensor-1- (TT-1)
   - temp_sensor-2-(TT

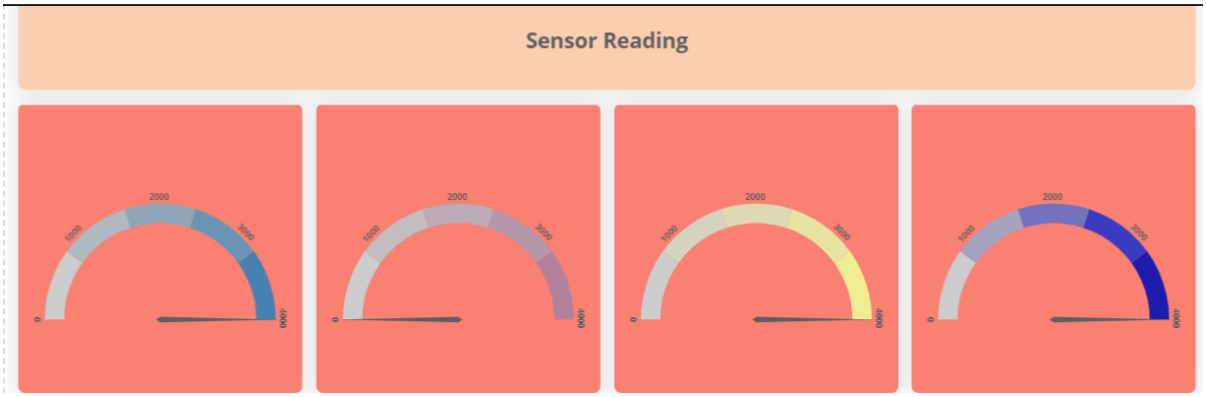


Figure 37:Sensor reading

2. Actuators used in our process:
   - Switch Mode (Supervisor-Switch)
   - pump_1-(G1)
   - Pump_2-(G2)
   - Control Valve "U" (LC-1)
   - Control Valve "D" (FC-1)
   - heater-(j_1)
   - Stirrer Motor (M2)-A
   - Fan Motor (M1)-A

Figure 38:Controlling actuators

- When the control mode button is pressed, the automatic process fails, and control switches to the website. This allows us to have control over any actuator as needed, enabling remote system control

11. Analysis for Reading Temperature sensor 2

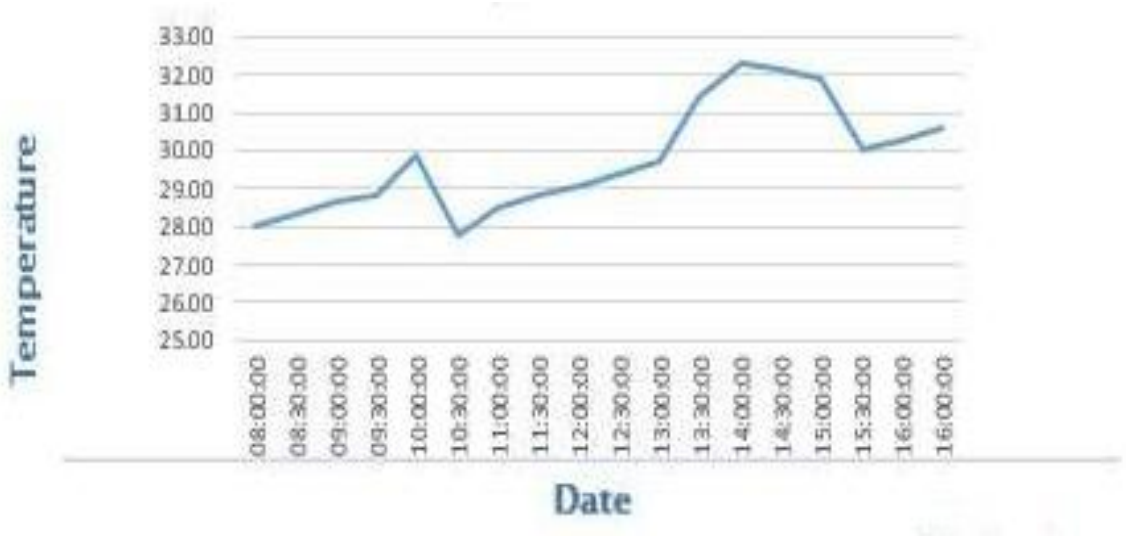


Figure 39:Analysis for temp. sensor 2

12. Communication between Upidots and plc

the communication between the PLCs and upidots was performed using esp32 and TTL converter and this connection was explained in the communication chapter.

13. Test

Chapter Nine

CONCLUSION

In general, LSIS Programmable Logic Controllers (PLCs) are not as extensively utilized in industry compared to other PLC types. This has led to a significant scarcity of resources beyond the manuals provided by the company, making hardware connections, software configuration, and programming more challenging. Specifically, establishing PLC connections with other PLCs has proven to be a complex task due to the limited availability of resources and tutorials on this particular topic.

The need for connections between the three PLCs could have been mitigated if communication modules, such as Ethernet modules, or analog input and output expansion modules were available. However, due to their exorbitant prices and predominantly overseas availability, resorting to software-based connections became inevitable.

The functionality of the system can be enhanced through enabling remote controlling and monitoring capabilities through the Industrial Internet of Things (IIoT), implementation of control and monitoring by a more advanced ESP controller or the adoption of an alternative microcontroller. Also, introducing digital twin technology to the system can be a great asset.

APPENDIX A

References

[1] Mod. UNIPRO/EV User Guide. Italy.

[2] Mod. UNIPRO/EV Brochure. Italy.

[3] LSIS. XGB Series Catalogue. LS Industrial Systems (Version 2018). South Korea.

[4] Release Note XG5000. LS Industrial Systems (Version 4.72). South Korea.

[5] LSIS. XGB Standard/Economic Type Main Unit. LS Industrial Systems (Version 2018). South Korea.

[6] Factory Automation. LS Industrial Systems (Version 2020). South Korea.

[7] RealPars, "What is Modbus and How does it Work?" Dec. 3 2018, [Accessed 16 June 2023].

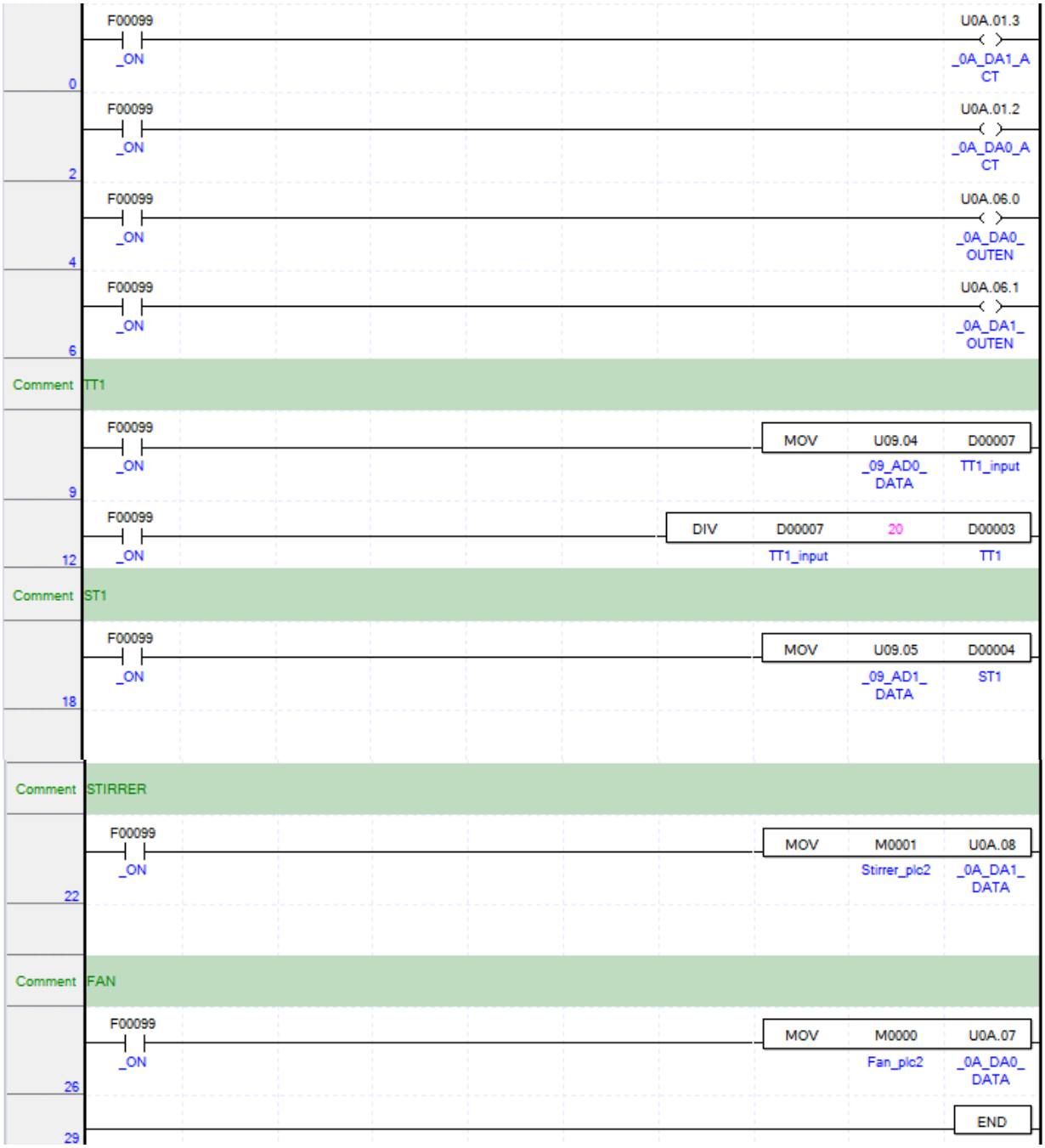
[8] RealPars, "How does Modbus Communication Protocol Work?" Dec 17, 2018 [Accessed 16 June 2023].

APPENDIX B

PLC1

View Variable | Device | View Flag

	Variable	Type	Device	Used	HMI	Comment
1	fan	WORD	D00002	☐	☐	fan
2	Fan_plc2	WORD	M0000	☑	☐	Fan is received from plc 2
3	J1	BIT	K00003	☐	☐	
4	LC1_Virtual	WORD	D00005	☑	☐	LC1 is written to PIC3
5	ST1	WORD	D00004	☑	☐	analog input ST1
6	stirrer	WORD	D00001	☐	☐	stirrer
7	Stirrer_plc2	WORD	M0001	☑	☐	Stirrer is received from plc 2
8	TT1	WORD	D00003	☑	☐	TT1
9	TT1_input	WORD	D00007	☑	☐	TT2
10	_09_AD0_ACT	BIT	U09.01.0	☐	☐	Analog Input Option Board: AD0 Activation Status
11	_09_AD0_DATA	WORD	U09.04	☑	☐	Analog Input Option Board: AD0 Digital Output Data
12	_09_AD0_ERR	BIT	U09.01.8	☐	☐	Analog Input Option Board: AD0 Error Code
13	_09_AD0_IDD	BIT	U09.01.4	☐	☐	Analog Input Option Board: CH0 Input Disconnection F
14	_09_AD1_ACT	BIT	U09.01.1	☐	☐	Analog Input Option Board: AD1 Activation Status
15	_09_AD1_DATA	WORD	U09.05	☑	☐	Analog Input Option Board: AD1 Digital Output Data
16	_09_AD1_ERR	BIT	U09.01.9	☐	☐	Analog Input Option Board: AD1 Error Code
17	_09_AD1_IDD	BIT	U09.01.5	☐	☐	Analog Input Option Board: CH1 Input Disconnection F
18	_09_ERR	BIT	U09.00.0	☐	☐	Analog Input Option Board: Error
19	_09_RDY	BIT	U09.00.F	☐	☐	Analog Input Option Board: Ready
20	_0A_DA0_ACT	BIT	U0A.01.2	☑	☐	Analog Output Option Board: DA0 Activation Status
21	_0A_DA0_DATA	WORD	U0A.07	☑	☐	Analog Output Option Board: DA0 Digital Input Data
22	_0A_DA0_ERR	BIT	U0A.01.A	☐	☐	Analog Output Option Board: DA0 Error Code
23	_0A_DA0_OUTEN	BIT	U0A.06.0	☑	☐	Analog Output Option Board: DA0 Output Enable
24	_0A_DA1_ACT	BIT	U0A.01.3	☑	☐	Analog Output Option Board: Output CH1 Activation St
25	_0A_DA1_DATA	WORD	U0A.08	☑	☐	Analog Output Option Board: DA1 Digital Input Data
26	_0A_DA1_ERR	BIT	U0A.01.B	☐	☐	Analog Output Option Board: DA1 Error Code
27	_0A_DA1_OUTEN	BIT	U0A.06.1	☑	☐	Analog Output Option Board: DA1 Output Enable
28	_0A_ERR	BIT	U0A.00.0	☐	☐	Analog Output Option Board: Error
29	_0A_OUTEN	WORD	U0A.06	☐	☐	Analog Output Option Board: Output Enable
30	_0A_RDY	BIT	U0A.00.F	☐	☐	Analog Output Option Board: Ready

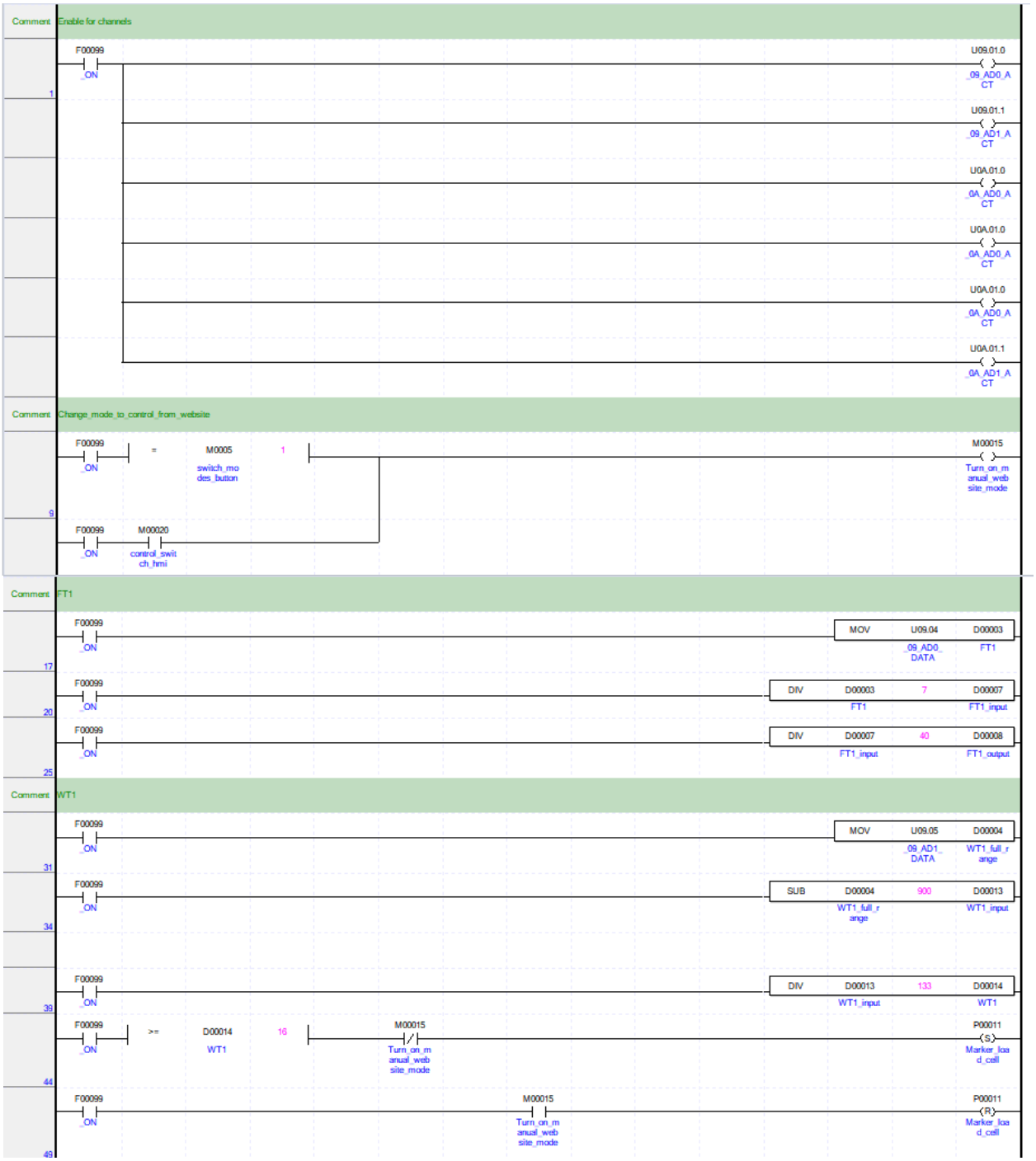

0
F00099
_ON
U0A.01.3
_0A_DA1_ACT
2
F00099
_ON
U0A.01.2
_0A_DA0_ACT
4
F00099
_ON
U0A.06.0
_0A_DA0_OUTEN
6
F00099
_ON
U0A.06.1
_0A_DA1_OUTEN
Comment TT1
9
F00099
_ON
MOV U09.04 D00007
_09_AD0_DATA TT1_input
12
F00099
_ON
DIV D00007 20 D00003
TT1_input TT1
Comment ST1
18
F00099
_ON
MOV U09.05 D00004
_09_AD1_DATA ST1
Comment STIRRER
22
F00099
_ON
MOV M0001 U0A.08
Stirrer_plc2 _0A_DA1_DATA
Comment FAN
26
F00099
_ON
MOV M0000 U0A.07
Fan_plc2 _0A_DA0_DATA
29
END

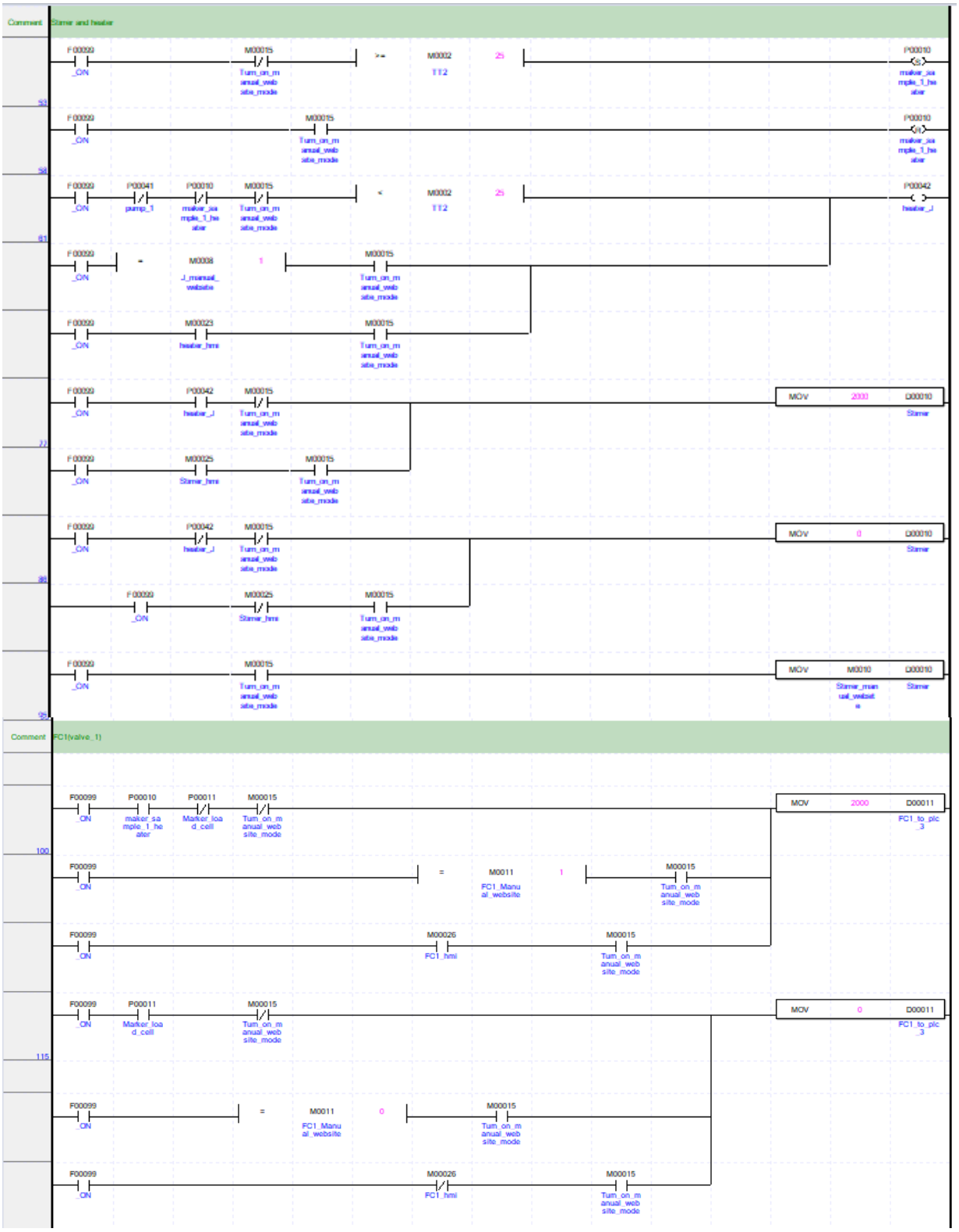
PLC2

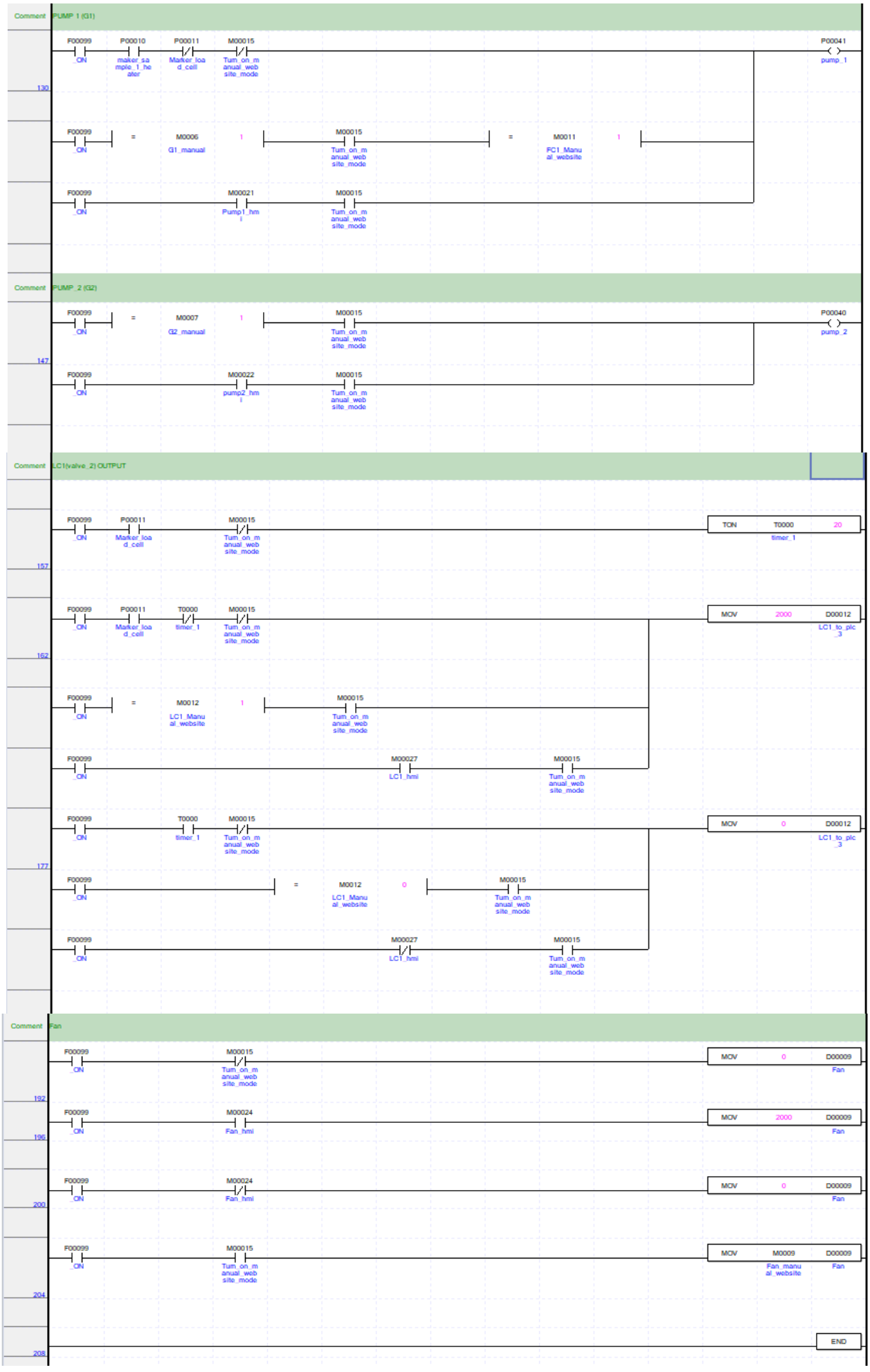
View Variable | Device | View Flag

	Variable	Type	Device	Used	HMI	Comment
1	Alarm_high_TT2	BIT	M00030	☑	☐	
2	CE1	WORD	D00002	☐	☑	
3	CE1_input	WORD	D00001	☐	☑	
4	control_switch_hmi	BIT	M00020	☑	☐	
5	Fan	WORD	D00009	☑	☑	Fan is send to plc 1
6	Fan_hmi	BIT	M00024	☑	☐	
7	Fan_manual_website	WORD	M0009	☑	☑	Manual control for Fan
8	FC1	BIT	D00011.0	☐	☐	valve_1 is send to plc 3
9	FC1_hmi	BIT	M00026	☑	☐	
10	FC1_Manual_website	WORD	M0011	☑	☐	
11	FC1_to_plc_3	WORD	D00011	☑	☐	
12	FT1	WORD	D00003	☑	☑	FT1
13	FT1_input	WORD	D00007	☑	☑	
14	FT1_output	WORD	D00008	☑	☑	
15	g1_input	WORD	D00005	☐	☐	
16	G1_manual	WORD	M0006	☑	☑	Manual control for G1
17	G2_manual	WORD	M0007	☑	☑	Manual control for G2
18	heater_2off	BIT/WORD	C0004	☐	☐	
19	heater_hmi	BIT	M00023	☑	☐	
20	heater_J	BIT	P00042	☑	☑	Heater
21	J_manual_website	WORD	M0008	☑	☑	J for manual control
22	LC1	BIT	D00012.0	☐	☐	valve_2 is send to plc3
23	LC1_hmi	BIT	M00027	☑	☐	
24	LC1_Manual_website	WORD	M0012	☑	☐	
25	LC1_to_plc_3	WORD	D00012	☑	☐	
26	LT1	WORD	M0003	☐	☐	LT1 is received from PLC 3
27	maker_sample_1_heater	BIT	P00010	☑	☐	
28	Marker_load_cell	BIT	P00011	☑	☐	
29	marker_pump2_on	BIT	P00012	☐	☐	
30	marker_stirrer	WORD	M0014	☐	☐	marker_stirrer
31	PA1	BIT	P00003	☐	☑	PA1
32	PA2	BIT	P00004	☐	☑	PA2
33	Pump1_hmi	BIT	M00021	☑	☐	
34	pump2_hmi	BIT	M00022	☑	☐	
35	pump_1	BIT	P00041	☑	☑	pump_1
36	pump_2	BIT	P00040	☑	☑	Pump_2
37	Pump_2_timer	BIT/WORD	T0003	☐	☐	
38	reset_heater_After3samples	BIT	K00003	☐	☐	
39	sampe2_WT1	BIT	K00001	☐	☐	
40	Second_sample	BIT	K00000	☐	☐	
41	ST1	WORD	M0001	☐	☐	ST1 is received from PLC 1
42	Stirrer	WORD	D00010	☑	☑	stirrer is send to plc1
43	Stirrer_hmi	BIT	M00025	☑	☐	
44	Stirrer_manual_website	WORD	M0010	☑	☑	MAnual control for stirrer
45	switch_modes_button	WORD	M0005	☑	☑	Switch between manual and automatic
46	timer_1	BIT/WORD	T0000	☑	☑	
47	timer_2	BIT/WORD	T0001	☐	☑	
48	TT1	WORD	M0000	☐	☐	TT1 is received from PLC 1
49	TT2	WORD	M0002	☑	☐	TT2 is received from PLC 3
50	Turn_on_manual_website_mode	BIT	M00015	☑	☐	
51	WT1	WORD	D00014	☑	☑	WT1
52	WT1_counter	BIT/WORD	C0000	☐	☐	
53	WT1_counter_2	BIT/WORD	C0003	☐	☐	
54	WT1_full_range	WORD	D00004	☑	☑	WT1_full_range
55	WT1_input	WORD	D00013	☑	☑	WT1_input
56	_09_AD0_ACT	BIT	U09.01.0	☑	☐	Analog Input Option Board: AD0 Activation Status
57	_09_AD0_DATA	WORD	U09.04	☑	☐	Analog Input Option Board: AD0 Digital Output Data
58	_09_AD0_ERR	BIT	U09.01.8	☐	☐	Analog Input Option Board: AD0 Error Code
59	_09_AD0_IDD	BIT	U09.01.4	☐	☐	Analog Input Option Board: CH0 Input Disconnection F

60	_09_AD1_ACT	BIT	U09.01.1	☑	☐	Analog Input Option Board: AD1 Activation Status
61	_09_AD1_DATA	WORD	U09.05	☑	☐	Analog Input Option Board: AD1 Digital Output Data
62	_09_AD1_ERR	BIT	U09.01.9	☐	☐	Analog Input Option Board: AD1 Error Code
63	_09_AD1_IDD	BIT	U09.01.5	☐	☐	Analog Input Option Board: CH1 Input Disconnection F
64	_09_ERR	BIT	U09.00.0	☐	☐	Analog Input Option Board: Error
65	_09_RDY	BIT	U09.00.F	☐	☐	Analog Input Option Board: Ready
66	_0A_AD0_ACT	BIT	U0A.01.0	☑	☐	Analog Input Option Board: AD0 Activation Status
67	_0A_AD0_DATA	WORD	U0A.04	☐	☐	Analog Input Option Board: AD0 Digital Output Data
68	_0A_AD0_ERR	BIT	U0A.01.8	☐	☐	Analog Input Option Board: AD0 Error Code
69	_0A_AD0_IDD	BIT	U0A.01.4	☐	☐	Analog Input Option Board: CH0 Input Disconnection F
70	_0A_AD1_ACT	BIT	U0A.01.1	☑	☐	Analog Input Option Board: AD1 Activation Status
71	_0A_AD1_DATA	WORD	U0A.05	☐	☐	Analog Input Option Board: AD1 Digital Output Data
72	_0A_AD1_ERR	BIT	U0A.01.9	☐	☐	Analog Input Option Board: AD1 Error Code
73	_0A_AD1_IDD	BIT	U0A.01.5	☐	☐	Analog Input Option Board: CH1 Input Disconnection F
74	_0A_ERR	BIT	U0A.00.0	☐	☐	Analog Input Option Board: Error
75	_0A_RDY	BIT	U0A.00.F	☐	☐	Analog Input Option Board: Ready
76				☐	☐	


Comment
Enable for channels
F00099
_ON
U09.01.0
_09_AD0_ACT
U09.01.1
_09_AD1_ACT
U0A.01.0
_0A_AD0_ACT
U0A.01.0
_0A_AD0_ACT
U0A.01.0
_0A_AD0_ACT
U0A.01.1
_0A_AD1_ACT
Comment
Change_mode_to_control_from_website
F00099
_ON
=
M0005
switch_modes_button
1
M00015
Turn_on_manual_website_mode
F00099
_ON
M00020
control_switch_hmi
Comment
FT1
F00099
_ON
MOV
U09.04
_09_AD0_DATA
D00003
FT1
F00099
_ON
DIV
D00003
FT1
7
D00007
FT1_input
F00099
_ON
DIV
D00007
FT1_input
40
D00008
FT1_output
Comment
WT1
F00099
_ON
MOV
U09.05
_09_AD1_DATA
D00004
WT1_full_range
F00099
_ON
SUB
D00004
WT1_full_range
900
D00013
WT1_input
F00099
_ON
DIV
D00013
WT1_input
133
D00014
WT1
F00099
_ON
>=
D00014
WT1
16
M00015
Turn_on_manual_website_mode
P00011
Marker_load_cell
F00099
_ON
M00015
Turn_on_manual_website_mode
P00011
Marker_load_cell

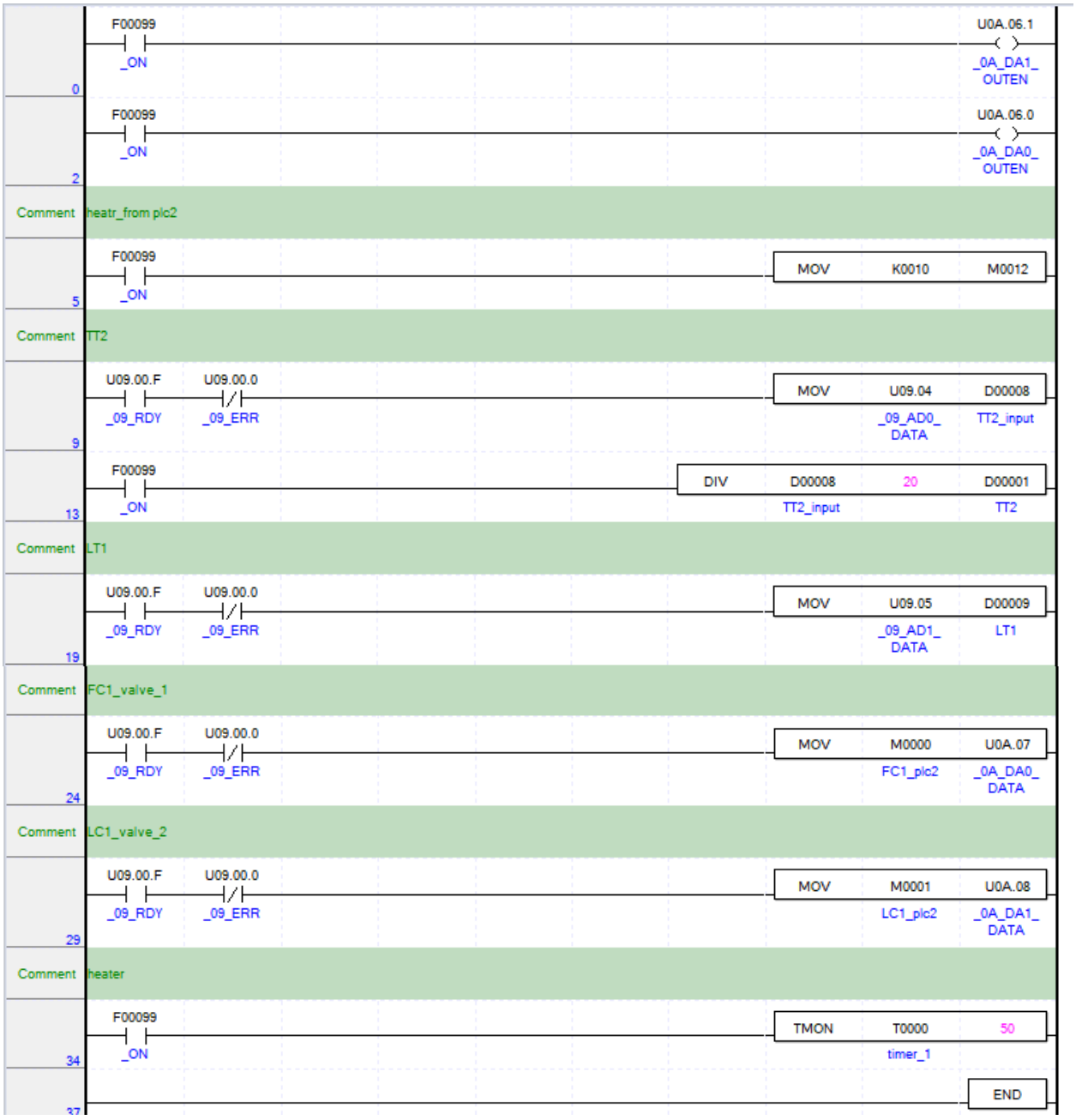

Comment
Stirrer and heater
F00000
_ON
M00015
Turn_on_manual_website_mode
>=
M0002
TT2
25
P00010
maker_sample_1_heater
P00041
pump_1
P00042
heater_1
M0008
J_manual_website
1
M00023
heater_hmi
M00025
Stirrer_hmi
MOV
2000
0
D00010
Stirrer
M0010
Stirrer_manual_webset
Comment
FC1(valve_1)
F00099
P00011
Marker_load_cell
M0011
FC1_Manual_website
M00026
FC1_hmi
D00011
FC1_to_plc_3


Comment
PUMP 1 (G1)
F00099 _ON
P00010 maker_sample_1_heater
P00011 Marker_load_cell
M00015 Turn_on_manual_website_mode
P00041 pump_1
130
M0006 G1_manual
M0011 FC1_Manual_website
M00021 Pump1_hmi
Comment
PUMP_2 (G2)
M0007 G2_manual
P00040 pump_2
147
M00022 pump2_hmi
Comment
LC1(valve_2) OUTPUT
157
TON T0000 20 timer_1
162
MOV 2000 D00012 LC1_to_plc_3
M0012 LC1_Manual_website
M00027 LC1_hmi
177
MOV 0 D00012 LC1_to_plc_3
Comment
Fan
192
MOV 0 D00009 Fan
196
MOV 2000 D00009 Fan
M00024 Fan_hmi
200
204
MOV M0009 D00009 Fan_manual_website Fan
208
END

PLC3

View Variable | Device | View Flag

	Variable	Type	Device	Used	HMI	Comment
1	FC1_plc2	WORD	M0000	☑	☐	FC1 is received from PLC2
2	heater	BIT	P00040	☐	☐	
3	heater_lamp	BIT	P00043	☐	☐	
4	J1	BIT	K00000	☐	☐	
5	LC1_plc2	WORD	M0001	☑	☐	LC1 is received from PLC2
6	LC1_Virtual	WORD	M0002	☐	☐	LC1 is received from PLC 1
7	LT1	WORD	D00009	☑	☐	LT1
8	timer_1	BIT/WORD	T0000	☑	☐	
9	TT2	WORD	D00001	☑	☐	TT2
10	TT2_input	WORD	D00008	☑	☐	TT2_input
11	_09_AD0_ACT	BIT	U09.01.0	☐	☐	Analog Input Option Board: AD0 Activation Status
12	_09_AD0_DATA	WORD	U09.04	☑	☐	Analog Input Option Board: AD0 Digital Output Data
13	_09_AD0_ERR	BIT	U09.01.8	☐	☐	Analog Input Option Board: AD0 Error Code
14	_09_AD0_IDD	BIT	U09.01.4	☐	☐	Analog Input Option Board: CH0 Input Disconnection Flag
15	_09_AD1_ACT	BIT	U09.01.1	☐	☐	Analog Input Option Board: AD1 Activation Status
16	_09_AD1_DATA	WORD	U09.05	☑	☐	Analog Input Option Board: AD1 Digital Output Data
17	_09_AD1_ERR	BIT	U09.01.9	☐	☐	Analog Input Option Board: AD1 Error Code
18	_09_AD1_IDD	BIT	U09.01.5	☐	☐	Analog Input Option Board: CH1 Input Disconnection Flag
19	_09_ERR	BIT	U09.00.0	☑	☐	Analog Input Option Board: Error
20	_09_RDY	BIT	U09.00.F	☑	☐	Analog Input Option Board: Ready
21	_0A_DA0_ACT	BIT	U0A.01.2	☐	☐	Analog Output Option Board: DA0 Activation Status
22	_0A_DA0_DATA	WORD	U0A.07	☑	☐	Analog Output Option Board: DA0 Digital Input Data
23	_0A_DA0_ERR	BIT	U0A.01.A	☐	☐	Analog Output Option Board: DA0 Error Code
24	_0A_DA0_OUTEN	BIT	U0A.06.0	☑	☐	Analog Output Option Board: DA0 Output Enable
25	_0A_DA1_ACT	BIT	U0A.01.3	☐	☐	Analog Output Option Board: Output CH1 Activation Status
26	_0A_DA1_DATA	WORD	U0A.08	☑	☐	Analog Output Option Board: DA1 Digital Input Data
27	_0A_DA1_ERR	BIT	U0A.01.B	☐	☐	Analog Output Option Board: DA1 Error Code
28	_0A_DA1_OUTEN	BIT	U0A.06.1	☑	☐	Analog Output Option Board: DA1 Output Enable
29	_0A_ERR	BIT	U0A.00.0	☐	☐	Analog Output Option Board: Error
30	_0A_OUTEN	WORD	U0A.06	☐	☐	Analog Output Option Board: Output Enable
31	_0A_RDY	BIT	U0A.00.F	☐	☐	Analog Output Option Board: Ready


0
F00099
_ON
U0A.06.1
_0A_DA1_
OUTEN
2
F00099
_ON
U0A.06.0
_0A_DA0_
OUTEN
Comment
heatr_from plc2
5
F00099
_ON
MOV
K0010
M0012
Comment
TT2
9
U09.00.F
_09_RDY
U09.00.0
_09_ERR
MOV
U09.04
_09_AD0_
DATA
D00008
TT2_input
13
F00099
_ON
DIV
D00008
TT2_input
20
D00001
TT2
Comment
LT1
19
U09.00.F
_09_RDY
U09.00.0
_09_ERR
MOV
U09.05
_09_AD1_
DATA
D00009
LT1
Comment
FC1_valve_1
24
U09.00.F
_09_RDY
U09.00.0
_09_ERR
MOV
M0000
FC1_plc2
U0A.07
_0A_DA0_
DATA
Comment
LC1_valve_2
29
U09.00.F
_09_RDY
U09.00.0
_09_ERR
MOV
M0001
LC1_plc2
U0A.08
_0A_DA1_
DATA
Comment
heater
34
F00099
_ON
TMON
T0000
timer_1
50
37
END

APPENDIX C

ESP32 Full Code [Modbus RTU & Ubidots]:

```
#include "UbidotsEsp32Mqtt.h"
#include <ModbusMaster.h>
#include <SoftwareSerial.h>

/*|||||||||||||>Define the Modbus communication parameters|||||||||||||||*/
#define BAUD_RATE 115200
#define PARITY 0
#define DATA_BITS 8
#define STOP_BITS 1
#define SerialRS485_RX_PIN 16
#define SerialRS485_TX_PIN 17
/*|||||||||||||Define the Modbus communication parameters<|||||||||||||||*/




/*|||||||||||||>Define the slave addresses for the two devices|||||||||||*/
#define PLC_SLAVE1_Station 1
#define PLC_SLAVE2_Station 2
#define PLC_SLAVE3_Station 3

SoftwareSerial Serial_mod(SerialRS485_RX_PIN, SerialRS485_TX_PIN);
/*|||||||||||||Define the slave addresses for the two devices<|||||||||||*/
```

```
/*|||||||||||||||||||||||>PLCS_VARIABLE|||||||||||||||||||||||||||||||*/
uint16_t TT1_PLC1 = 0;
uint16_t ST1_PLC1 = 0;
uint16_t LC1_Virtual=0;
uint16_t FT1_PLC2 = 0;
uint16_t CE1_PLC2 = 0;
uint16_t WT1_PLC2 = 0;
uint16_t PA1_PLC2 = 0;
uint16_t PA2_PLC2 = 0;
uint16_t Fan=0;
uint16_t Stirrer=0;
uint16_t FC1=0;
uint16_t LC1=0;
uint16_t J1=0;
uint16_t TT2_PLC3 = 0;
uint16_t LT1_PLC3 = 0;
/*|||||||||||||||||||||||PLCS_VARIABLE<|||||||||||||||||||||||||||||||*/

/*|||||||||||||||||||||||>WEBS_VARIABLE|||||||||||||||||||||||||||||||*/
const char *UBIDOTS_TOKEN = "BBUS-olb3W0DF86neBdwjUwMH0zhtJzjpUK";           // Put here your Ubidots TOKEN
const char *WIFI_SSID = "ASUENG wi-fi";               // Put here your Wi-Fi SSID
const char *WIFI_PASS = "ASUeng_Demo11";               // Put here your Wi-Fi password
const char *PUBLISH_DEVICE_LABEL = "sensors";    // Put here your Device label to which data  will be published
const char *SUBSCRIBE_DEVICE_LABEL = "actuators";   // Replace with the device label to subscribe to
float Switch_mode=0;
float pump1_web=0;
float pump2_web=0;
float FC_web=0;     // Valve 1  to small tank
float LC_web=0;     //Valve 2  output valve
float Fan_web=0;
float Heater_web=0;
float Stirrer_web=0;
/*|||||||||||||||||||||||WEBS_VARIABLE<|||||||||||||||||||||||||||||||*/
```

```
/*|||||||||||||>Create an instances of the ModbusMaster class|||||||||||||||*/
ModbusMaster Slave1;
ModbusMaster Slave2;
ModbusMaster Slave3;
/*|||||||||||||Create an instances of the ModbusMaster class<|||||||||||||||*/


/*|||||||||||||>Create an instances of the Ubidots class|||||||||||||||||||*/

Ubidots ubidots(UBIDOTS_TOKEN);

/*|||||||||||||Create an instances of the Ubidots class<|||||||||||||||||||*/


/*|||||||||||||||||||||||>CALLBACK_FUNCTION|||||||||||||||||||||||||||||||*/
/*||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*/


void callback(char *topic, byte *payload, unsigned int length) {
  // Convert the payload to a float value
  payload[length] = '\0';
  String str_payload = String((char *)payload);
  float value = str_payload.toFloat();

  if(strcmp(SUBSCRIBE_DEVICE_LABEL, "actuators") == 0 &&
      strcmp(topic, "/v2.0/devices/actuators/pump_1-g1/lv") == 0) {
      pump1_web = value;
    Serial.println(pump1_web);
  }
```

```
  if(strcmp(SUBSCRIBE_DEVICE_LABEL, "actuators") == 0 &&
      strcmp(topic, "/v2.0/devices/actuators/pump_2-g2/lv") == 0) {
      pump2_web = value;
    Serial.println(pump2_web);
  }

  if (strcmp(SUBSCRIBE_DEVICE_LABEL, "actuators") == 0 &&
      strcmp(topic, "/v2.0/devices/actuators/control_valve-d-fc-1/lv") == 0) {
    FC_web = value;
    Serial.println(FC_web);
  }

  if(strcmp(SUBSCRIBE_DEVICE_LABEL, "actuators") == 0 &&
      strcmp(topic, "/v2.0/devices/actuators/control_valve-u-lc-1/lv") == 0) {
      LC_web = value;
    Serial.println(LC_web);
  }

  if (strcmp(SUBSCRIBE_DEVICE_LABEL, "actuators") == 0 &&
      strcmp(topic, "/v2.0/devices/actuators/fan-motor-m1/lv") == 0) {
    Fan_web = value;
    Serial.println(Fan_web );
  }

  if(strcmp(SUBSCRIBE_DEVICE_LABEL, "actuators") == 0 &&
      strcmp(topic, "/v2.0/devices/actuators/heater-j_1/lv") == 0) {
      Heater_web = value;
    Serial.println(Heater_web);
  }
```

```
 if (strcmp(SUBSCRIBE_DEVICE_LABEL, "actuators") == 0 &&
        strcmp(topic, "/v2.0/devices/actuators/fan-motor-m1/lv") == 0) {
      Fan_web = value;
      Serial.println(Fan_web );
    }

 if(strcmp(SUBSCRIBE_DEVICE_LABEL, "actuators") == 0 &&
        strcmp(topic, "/v2.0/devices/actuators/heater-j_1/lv") == 0) {
        Heater_web = value;
      Serial.println(Heater_web);
    }

 if (strcmp(SUBSCRIBE_DEVICE_LABEL, "actuators") == 0 &&
        strcmp(topic, "/v2.0/devices/actuators/stirrer-motor-m2/lv") == 0) {
      Stirrer_web = value;
      Serial.println(Stirrer_web);
    }

 if (strcmp(SUBSCRIBE_DEVICE_LABEL, "actuators") == 0 &&
        strcmp(topic, "/v2.0/devices/actuators/switch_mode/lv") == 0) {
      Switch_mode = value;
      Serial.println(Switch_mode);
    }


}


/*|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*/
/*|||||||||||||||||||||||||CALLBACK_FUNCTION<|||||||||||||||||||||||||||||||||||*/

```

```
void setup() {
  /*|||||||||||||||||Initialize Speed Rate Of Communication|||||||||||||||||*/
  Serial.begin(115200);
  Serial_mod.begin(BAUD_RATE);

  /*|||||||||||||||||||||||||||>Slaves BEGINS|||||||||||||||||||||||||||||||*/
  // Modbus slave1 device
  Slave1.begin(PLC_SLAVE1_Station, Serial_mod);
  //  Modbus slave2 device
  Slave2.begin(PLC_SLAVE2_Station, Serial_mod);
  //Modbus slave3 device
  Slave3.begin(PLC_SLAVE3_Station, Serial_mod);
  /*|||||||||||||||||||||||||||Slaves BEGINS<|||||||||||||||||||||||||||||||*/

  /*|||||||||||||||||||||>UBIDOTS Initializations|||||||||||||||||||||||||*/
  /*|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*/

  ubidots.connectToWifi(WIFI_SSID, WIFI_PASS);
  ubidots.setCallback(callback);
  ubidots.setup();
  ubidots.reconnect();
  ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"pump_1-g1" ); // Insert the device and variable's Labels, respectively
  ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"pump_2-g2" ); // Insert the device and variable's Labels, respectively
  ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"control_valve-d-fc-1" ); // Insert the device and variable's Labels, respectively
  ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"control_valve-u-lc-1" ); // Insert the device and variable's Labels, respectively
  ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"fan-motor-m1" ); // Insert the device and variable's Labels, respectively
  ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"heater-j_1" ); // Insert the device and variable's Labels, respectively
  ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"stirrer-motor-m2" ); // Insert the device and variable's Labels, respectively
  ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"switch_mode" ); // Insert the device and variable's Labels, respectively
  /*|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*/
  /*|||||||||||||||||||||>UBIDOTS Initializations|||||||||||||||||||||||||*/
}
```

```
void loop()
{
  /*||||||||||||||||||||||>UBIDOTS Reconnecting|||||||||||||||||||||||||*/
  /*||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*/

  if (!ubidots.connected())
  {
    ubidots.reconnect();
    ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"pump_1-g1" ); // Insert the device and variable's Labels, respectively
    ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"pump_2-g2" ); // Insert the device and variable's Labels, respectively
    ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"control_valve-d-fc-1" ); // Insert the device and variable's Labels, respectively
    ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"control_valve-u-lc-1" ); // Insert the device and variable's Labels, respectively
    ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"fan-motor-m1" ); // Insert the device and variable's Labels, respectively
    ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"heater-j_1" ); // Insert the device and variable's Labels, respectively
    ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"stirrer-motor-m2" ); // Insert the device and variable's Labels, respectively
    ubidots.subscribeLastValue(SUBSCRIBE_DEVICE_LABEL,"switch_mode" ); // Insert the device and variable's Labels, respectively

  }

  /*||||||||||||||||||||||>UBIDOTS Reconnecting|||||||||||||||||||||||||*/
  /*||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*/

  /*||||||||||||||||||||Read Data from PlCS to PLCS|||||||||||||||||||||||*/
  /*|||||||||||| Read analog sensors signals  from PLC1|||||||||||||*/
    //Read TT1 and St1  from register D0003 & D0004

  uint8_t result_0 = Slave1.readInputRegisters(3, 3);
  if (result_0 == Slave1.ku8MBSuccess) {
    TT1_PLC1 = Slave1.getResponseBuffer(0);
    ST1_PLC1 = Slave1.getResponseBuffer(1);
    LC1_Virtual= Slave1.getResponseBuffer(2);
  }
```

```
  delay(25);
  /*|||||||||||||||||||| Read data from PLC2|||||||||||||||||||||||||*/
  /*|||||||||||| Read analog sensors signals from plc2|||||||||||*/
    //Read Wt1 from register D00014

  uint8_t result_1 = Slave2.readInputRegisters(14, 1);
  if (result_1 == Slave2.ku8MBSuccess) {
    WT1_PLC2 = Slave2.getResponseBuffer(0);
  }

  // Read FT1 from register D0008
  uint8_t result_2 = Slave2.readInputRegisters(8, 1);
  if (result_2 == Slave2.ku8MBSuccess) {
    FT1_PLC2 = Slave2.getResponseBuffer(0);
  }

  // Read CE1 from register D0002
  uint8_t result_3 = Slave2.readInputRegisters(2, 1);
  if (result_3 == Slave2.ku8MBSuccess) {
    CE1_PLC2 = Slave2.getResponseBuffer(0);
  }

  /*|||||||||||| Read analog actuators signals from plc2 |||||||||||*/
  // Read  Fan=0; Stirrer=0; FC1=0; LC1=0  from  (9-12)
  uint8_t result_4 = Slave2.readInputRegisters(9, 4);
  if (result_4 == Slave2.ku8MBSuccess) {
    Fan = Slave2.getResponseBuffer(0);
    Stirrer = Slave2.getResponseBuffer(1);
    FC1 = Slave2.getResponseBuffer(2);
    LC1 = Slave2.getResponseBuffer(3);
   }

  delay(25);
```

```
/*|||||||||||||||||||| Read data from PLC3|||||||||||||||||||||||*/
/*|||||||||||| Read analog sensors signals from plc3 ||||||||||||*/
// Read TT2 from register D0001

uint8_t result_5 = Slave3.readInputRegisters(1, 1);
if (result_5 == Slave3.ku8MBSuccess) {
  TT2_PLC3  = Slave3.getResponseBuffer(0);
}

// Read LT1 from register D0009
uint8_t result_6 = Slave3.readInputRegisters(9, 1);
if (result_6 == Slave3.ku8MBSuccess) {
  LT1_PLC3  = Slave3.getResponseBuffer(0);
}

delay(25);

/*||||||||||||||||||||Write Data from PlCS to PLCS|||||||||||||||||||||*/
/*|||||||||||||||Write analog actuator to PLC 1|||||||||||||||||*/
/*_____________________________________________*/
// Wrtie  [Fan]  to [PLC1 in M0000]:
//Wrtie  [ Stirrer]  to [PLC1 in M0001]:

uint8_t result_7 = Slave1.writeSingleRegister(1 , Stirrer);
delay(25);
uint8_t result_8 = Slave1.writeSingleRegister(0 , Fan);
delay(25);
```

```
/*||||||||||||||||||||Write analog sensors data to PLC 2|||||||||||||||||||||*/
/*_____________________________________________*/
  //Wrtie  [TT1_PLC1]  to [PLC2 in M0000]:
  //Wrtie  [ ST1_PLC1]  to [PLC2 in M0001]:
  //Wrtie  [TT2_PLC3]  to [PLC2 in M0002]:
  //Wrtie  [ LT1_PLC3]  to [PLC2 in M0003]:

uint8_t result_9 = Slave2.writeSingleRegister(0 ,TT1_PLC1 );
delay(25);
uint8_t result_10 = Slave2.writeSingleRegister(1  , ST1_PLC1 );
delay(25);
uint8_t result_11 = Slave2.writeSingleRegister(2  , TT2_PLC3 );
delay(25);
uint8_t result_12 = Slave2.writeSingleRegister(3  , LT1_PLC3 );
delay(25);


/*_____________________Write Web Action To PLC2_____________________*/
  /*
  Switch_mode=0;
  pump1_web=0;
  pump2_web=0;
  FC_web=0;     // Valve 1  to small tank
  LC_web=0;     //Valve 2  output valve
  Fan_web=0;
  Heater_web=0;
  Stirrer_web=0;
  */

uint8_t result_13 = Slave2.writeSingleRegister(5 ,Switch_mode );
delay(25);
uint8_t result_14 = Slave2.writeSingleRegister(6 ,pump1_web );
delay(25);
```

```
322     uint8_t result_15 = Slave2.writeSingleRegister(7 ,pump2_web );
323     delay(25);
324     uint8_t result_16 = Slave2.writeSingleRegister(8 ,Heater_web);
325     delay(25);
326     uint8_t result_17= Slave2.writeSingleRegister(9  ,Fan_web);
327     delay(25);
328     uint8_t result_18 = Slave2.writeSingleRegister(10  , Stirrer_web);
329     delay(25);
330     uint8_t result_19 = Slave2.writeSingleRegister(11  ,FC_web);
331     delay(25);
332     uint8_t result_20= Slave2.writeSingleRegister(12 ,LC_web);
333     delay(25);
334
335
336
337   /*||||||||||||||||||||Write  analog actuator  to PLC 3|||||||||||||||||||||||*/
338   /*_____________________________________________*/
339     //Wrtie [FC1]  to [PLC3 in M0000]:
340     //Wrtie  [LC1]  to [PLC3 in M0001]:
341
342   uint8_t result_21 = Slave3.writeSingleRegister(1  , LC1 );
343   delay(25);
344   uint8_t result_22 = Slave3.writeSingleRegister(0  , FC1 );
345   delay(25);
346
347
348
```

```
348   /*||||||||||||||||||||||>Publishing For Cloud|||||||||||||||||||||||||||||||*/
349   /*||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*/
350
351     //float value_1 = 205.1458;
352     ubidots.add("temp_sensor-1-tt-1",TT1_PLC1);
353     ubidots.publish(PUBLISH_DEVICE_LABEL);
354     ubidots.add("temp_sensor-2-tt-2", TT2_PLC3);
355     ubidots.publish(PUBLISH_DEVICE_LABEL);
356     ubidots.add("diff_pressure-lt-1", LT1_PLC3);
357     ubidots.publish(PUBLISH_DEVICE_LABEL);
358     ubidots.add("flow_sensor-ft-1", FT1_PLC2);
359     ubidots.publish(PUBLISH_DEVICE_LABEL);
360     ubidots.add("load_cell-wt-1", WT1_PLC2);
361     ubidots.publish(PUBLISH_DEVICE_LABEL);
362     //ubidots.add("pressure_sensor-pa_1",PA1_PLC2);
363     //ubidots.publish(PUBLISH_DEVICE_LABEL);
364     //ubidots.add("pressure_sensor-pa_2", PA2_PLC2);
365     //ubidots.publish(PUBLISH_DEVICE_LABEL);
366     ubidots.add("taco_meter-st-1", ST1_PLC1);
367     ubidots.publish(PUBLISH_DEVICE_LABEL);
368
369     ubidots.loop();
370
371
372   /*||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*/
373   /*||||||||||||||||||||||||Publishing For Cloud<|||||||||||||||||||||||||||||||*/
374
375     delay(100);
376
377 }
```

QR Code for full code: